ISSN: 0973-6875

Volume: 13, Issue: Special Issue, NSEFCCIC 2026

Special Issue in “International Journal of HIT Transaction on ECCN”

Proceedings of National Symposium on Emerging Frontiers of Cognitive Communication & Intelligent Computing (NSEFCCIC 2026)

Published By

Haldia Institute of Technology, Haldia, West Bengal, India

Funded by: Haldia Institute of Technology

Edited By:

Prof. (Dr.) Tarun Kanti Jana
Prof. (Dr.) Subhankar Joardar
Prof. (Dr.) Chanchal Kumar De
Dr. Ritwik Haldar
Dr. Biplab Bag
Dr. Surajit Mukherjee

Special Issuse in "International Journal of HIT Transaction on ECCN"

CONTENTS

CHIEF PATRON:

Dr. Lakshman Chandra Seth, D.Lit, Hon'ble Chairman, HIT Haldia.

ADVISORY COMMITTEE:

Mr. Sayantan Seth, Vice Chairman, HIT, Haldia.
Mr. Ashis Lahiri, Secretary, ICARE,Haldia.
Prof. (Dr.) Tarun Kanti Jana, Principal, HIT, Haldia.
Dr. Anjan Mishra, Registrar General, HIT, Haldia.
Prof. (Dr.) Subhankar Joardar, Dean, SECI, HIT, Haldia.

PROGRAM CHAIR:

Prof. (Dr.) Chanchal Kumar De, Prof. & Head, Dept. of ECE, HIT Haldia.

CONVENOR

Dr. Ritwik Haldar, Assoc. Prof., Dept. of ECE, HIT, Haldia.
Dr. Himanshu Ranjan Das, Asst. Prof., Dept. of ECE, HIT Haldia.

SECRETARY

Dr. Surajit Mukherjee, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Rana Kumar Jana, Asst. Prof., Dept. of ECE, HIT Haldia.

PUBLICATION CHAIR:

Dr. Biplab Bag, Asst. Prof., Dept. of ECE, HIT Haldia.
Mr. Asim Kuilya, Asst. Prof., Dept. of ECE, HIT Haldia.

FINANCE CHAIR:

Dr. Jagannath Samanta, Assoc. Prof., Dept. of ECE, HIT Haldia
Dr. Dibyendu Chowdhury, Assoc. Prof., Dept. of ECE, HIT, Haldia.

ORGANIZING COMMITTEE:

Dr. Malay Kumar Pandit, Prof., Dept. of ECE, HIT Haldia.
Dr. Kishalaya Charkrabarti, Prof., Dept. of ECE, HIT Haldia.
Dr. Avisankar Roy, Assoc. Prof., Dept. of ECE, HIT, Haldia.
Dr. Tilak Mukherjee, Assoc. Prof., Dept. of ECE, HIT, Haldia.
Dr. Sagarika Das, Assoc. Prof. Dept. of ECE, HIT, Haldia.
Dr. Avishek Das, Assoc. Prof. Dept. of ECE, HIT, Haldia.
Dr. Akinchan Das, Assoc. Prof., Dept. of ECE, HIT, Haldia.
Dr. Himadri Sekhar Das, Assoc. Prof., Dept. of ECE, HIT, Haldia.
Dr. Kalyan Sundar Kola, Assoc. Prof., Dept. of ECE, HIT, Haldia.
Dr. Kushal Roy, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Sudipta Bardhan, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Suman Paul, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Tirthadip Sinha, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Pinaki Satpathy, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Santanu Maity, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Kamanashis Goswami, Asst. Prof., Dept. of ECE, HIT Haldia.
Dr. Dipak Samanta, Asst. Prof., Dept. of ECE, HIT Haldia.
Mrs. Razia Sultana, Asst. Prof., Dept. of ECE, HIT Haldia.
Ms. Sayani Ghosh, Asst. Prof., Dept. of ECE, HIT Haldia.

Mrs. Moumita Jana, Asst. Prof., Dept. of ECE, HIT Haldia.
Mr. Jayanta Kumar Bag, Asst.Prof., Dept. of ECE, HIT Haldia.
Mrs. Pallabi Pahari, Asst.Prof., Dept. of ECE, HIT Haldia.

LOCAL ORGANIZING COMMITTEE:

Mr. Pulak Maity, Asst. Prof, Dept. of ECE, HIT Haldia.
Mr. Sachindeb Jana, Asst. Prof., Dept. of ECE, HIT Haldia.
Mr. Atanu Pradhan, Asst. Prof., Dept. of ECE, HIT Haldia.
Mr. Tapan Maity, Asst. Prof., Dept. of ECE, HIT Haldia.
Mr. Shubhendu Barman, Instructor, Dept. of ECE, HIT Haldia.
Mrs. Ira Samanta, Asst. Prof, Dept. of ECE, HIT Haldia.
Mr. Souvik Devangshi, Demonstrator, Dept. of ECE, HIT Haldia.
Mr. Tirthankar Majee, Demonstrator, Dept. of ECE, HIT Haldia.
Mr. Swapan Jana, General Assistance, Dept. of ECE, HIT Haldia.

INVITED SPEAKERS:

1. **Prof. Abhirup Das Barman** Professor**,** HoD, Institute of Radio Physics and Electronics, University of Calcutta, Kolkata, India.
2. **Prof. Santi Prasad Maity** Professor, IT Department, IIEST, Shibpur, India
3. **Dr. Joydeep Chandra** Associate Professor, CSE Department, IIT PATNA, India
4. **Dr. Mamata Dalui** Associate Professor, CSE Department, NIT Durgapur, India
5. **Dr. Subhrajyoti Deb** Assistant Professor, IT Department, IIIT Bhopal, India

SYMPOSIUM WEBSITE:

https://nsefccic2026.ecehithaldia.in/

HIT WEBSITE

http://hithaldia.ac.in

SYMPOSIUM EMAIL ID:

nsefccic@hithaldia.ac.in

PAPER SUBMISSION EASY CHAIR LINK:

https://cmt3.research.microsoft.com/NSEFCCIC2026

Proceedings of Previous Symposium/Conferences

1. **Visit previous symposium NSIC 2025**
   https://hittransaction.in/journalContents.php?vid=10&sid=NSIC2025
2. **Visit previous conference ICCDC 2017 proceedings in LNEE**
   https://www.springer.com/in/book/9789811085840
3. **Visit previous conference ICCDC 2019 proceedings in LNEE**
   https://www.springer.com/in/book/9789811508288
4. **Visit previous conference ICCDC 2021proceedings in LNEE**
   https://link.springer.com/book/10.1007/978-981-16-9154-6
5. **Visit previous conference ICCDC 2023 proceedings in LNEE**

https://link.springer.com/book/10.1007/978-981-99-2710-4

6. **Visit previous conference ICCDC 2025 proceedings in LNEE**
   https://link.springer.com/book/10.1007/978-3-032-18470-2

Preface

It is with great pleasure that we welcome you to the National Symposium on "Emerging Frontiers of Cognitive Communication & Intelligent Computing" (NSEFCCIC 2026), organized by the Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal, India, on August 27–28, 2026. The symposium provides a forum for researchers, scholars, students, and industry professionals to exchange ideas, address emerging challenges, and explore innovative solutions in intelligent communication and computing systems.

In response to the call for papers, submissions were evaluated through peer review on originality, relevance, technical quality, and significance, and selected papers were accepted for oral presentation. Together they span cognitive communication, intelligent computing, machine learning and deep learning, IoT and edge computing, cybersecurity, 5G/6G technologies, and interdisciplinary applications including healthcare, robotics, smart grids, UAVs, and autonomous systems. All accepted and presented papers will be considered for publication in the conference proceedings as a special issue of the International Journal of HIT Transactions on ECCN (ISSN: 0973-6875), subject to fulfilment of the required registration and publication conditions. Alongside the technical sessions, invited talks by experts from academia and industry offer further opportunity for interaction, with participation in both offline and online modes.

A symposium of this kind would not be possible without the organizing team, editorial board, technical program committee, reviewers, faculty members, invited speakers, authors, and participants. We are especially grateful to the reviewers for their time and constructive suggestions in improving the quality of the submitted papers, and we look forward to making NSEFCCIC 2026 a successful platform for knowledge sharing and research collaboration.

Published by: Haldia Institute of Technology, Haldia, West Bengal, INDIA

Prof. (Dr.) Tarun Kanti Jana
Prof. (Dr.) Subhankar Joardar
Prof. (Dr.) Chanchal Kumar De
Dr. Ritwik Haldar
Dr. Biplab Bag
Dr. Surajit Mukherjee

Message from the Volume Editors

It is with great pleasure that we welcome you to the National Symposium on "Emerging Frontiers of Cognitive Communication & Intelligent Computing" (NSEFCCIC 2026), organized by the Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal, on 27–28 August 2026. The symposium provides a high-quality national academic and industrial exchange forum for experts, scholars, and students in the fields of cognitive communication, intelligent computing, artificial intelligence, IoT, wireless networks, and cybersecurity. Its main objective is to address emerging challenges and explore innovative solutions in intelligent communication and computing systems, spanning sensing and edge devices through to next-generation network and cloud infrastructure.

Following the call for papers, **50** papers were submitted for presentation at the symposium. All submitted papers were sent to external referees, and after refereeing, **24** papers were recommended for publication in the conference proceedings, to be published as a special issue of the *International Journal of HIT Transactions on ECCN* (ISSN: 0973-6875).

Special thanks to the **Haldia Institute of Technology** for their financial support in conducting the symposium.

We are grateful to the speakers, participants, reviewers, sponsors, and Haldia Institute of Technology for their support and help, without which it would have been impossible to organize this symposium. We express our gratitude to the organizing committee members who worked tirelessly behind the scenes, taking care of the details that made it a success.

Prof. (Dr.) Tarun Kanti Jana
Prof. (Dr.) Subhankar Joardar
Prof. (Dr.) Chanchal Kumar De
Dr. Ritwik Haldar
Dr. Biplab Bag
Dr. Surajit Mukherjee

Prof. (Dr.) Tarun Kanti Jana is currently working as Principal, Haldia Institute of Technology, Haldia, India. He received his B. E. (Mechanical) degree from erstwhile Bengal Engineering College (currently IIEST, Shibpur) Howrah, West Bengal, India and M. Prod. E. (Production) from Jadavpur University, Kolkata, India, in the year 1988 and in 1990 respectively. He obtained PhD (Engineering) from Production Engineering, Jadavpur University in 2015. He is a Fellow of Institution of Engineers (India). He has published a number of research papers in international journals and conferences. He is the coauthor of two books: "Engineering Mechanics" and "Engineering Thermodynamics & Fluid Mechanics" both published by McGraw Hill Education. He is also the author of the book titled "Dynamic Scheduling by Agent Based Holonic Manufacturing System: Concepts, Framework, and Execution", published by Lap-Lambert, Academic Publishing, Germany. He has also contributed as an author of book chapter titled: The Role of Nontraditional Machining in Industry 4.0, (Source Title: Machine Learning Applications in Non-Conventional Machining Processes), published by IGI Global. He is a Reviewer of Journal: IEEE Transaction on Systems, Man and Cybernetics: Systems. His active areas of interests are Intelligent Manufacturing, Agent based systems, Industry 4.0 and Industry 5.0, Smart Manufacturing and Cyber Physical Production Systems, Digital Twin, Machine Tool 4.0.

Prof. (Dr.) Subhankar Joardar is presently Professor & Dean in the School of Computer Science, Electronics and Informatics, Haldia Institute of Technology, Haldia-721657, India. He received his Ph.D degree from Birla Institute of Technology, Mesra, Ranchi, India in 2016. He did his masters (M. Tech and MCA) both from BIT, Mesra, Ranchi in 2009 and 2002 respectively. He has published more than 50 technical papers in the referred journals/conferences. He has served as Organizing Chair of international conference (ICITAM 2017). He is also Program Committee member of International Conferences (ICCDC 2019, ICCDC 2023, ICCDC 2025) and Program Chair (ISAI 2022). He is a member of Computer Society of India. His current research interests include Swarm intelligence, algorithm design, Routing in Mobile Ad Hoc Networks, Machine Learning.

Prof. (Dr.) Chanchal Kr. De earned his Ph.D. in Electronics and Communication Engineering (specializing in Wireless Communication & Networking) from NIT Durgapur in 2014. With over 20 years of academic experience, his research areas include Radio Resource Management in Wireless Networks, OFDM Technology, Cooperative Communication, Cognitive Radio Networks, and Full-Duplex Communication. He has authored more than 50 publications in international reputed journals and conference proceedings, along with book chapters. Additionally, he has been guiding multiple Ph.D. students in the field of wireless communication. Dr. De has also held several key administrative roles at Haldia Institute of Technology, serving as Head of the ECE Department, IQAC Member, Convener of the R&D Monitoring Committee, and Convener of the Publication Committee. His outreach and professional contributions extend to membership in Boards of Studies at various universities, serving on editorial boards for conference proceedings, delivering invited talks at numerous conferences and seminars, and contributing as a Subject Matter Expert for MOOC courses.

Dr. Ritwik Haldar is an Associate Professor in the Department of Electronics and Communication Engineering at Haldia Institute of Technology, India. He earned his Ph.D. in Electronics and Communication Engineering from NIT Silchar, with research focused on energy-harvested wireless sensor networks using analytical and stochastic modeling. He completed his M.Tech in ECE with specialization in Microelectronics and VLSI Design from NIT Silchar and his B.Tech in ECE from Haldia Institute of Technology. His expertise spans Internet of Things, wireless communication and sensor networks, machine learning, with strong proficiency in MATLAB and Python. Dr. Haldar has published in reputed SCI and Scopus-indexed journals and patent. With over 7 years of teaching and

research experience, he actively contributes to curriculum development, student mentoring, and outcome-based education.

Dr. Biplab Bag received his Ph.D. in Engineering from the University of Kalyani in 2021. He qualified the UGC-NET examination in 2014. He completed his M.Tech. degree from West Bengal University of Technology, West Bengal, in 2008, and his B.Tech. degree from the same university in 2006. He is presently serving as an Assistant Professor in the Department of Electronics and Communication Engineering at Haldia Institute of Technology (Autonomous), West Bengal, India. Prior to this, he worked for 14 years as an Assistant Professor at Murshidabad College of Engineering and Technology, contributing extensively to teaching, research, and academic development. His research interests focus on planar monopole antennas, MIMO, array antennas, and frequency selective surface (FSS)-based antenna design for modern wireless communication systems. He has published more than 28 papers in reputed peer-reviewed journals and conferences. His teaching interests include control systems, electromagnetic field theory, antenna theory, analog and digital communication, signals and systems, and analog electronics.

Dr. Surajit Mukherjee is an Assistant Professor in the ECE Department at Haldia Institute of Technology, a role he has held since 2010. He earned his Ph.D. from the Central Institute of Technology, Kokrajhar (2025), following an M.Tech from the University of Burdwan and a B.Tech from WBUT. Specializing in RF, microwave, and antenna design, Dr. Mukherjee has authored over 20 publications, including journals and Springer book chapters. His research focuses on ultra-wideband (UWB) antennas, frequency selective surfaces, and patch antennas for X/Ku band applications. A member of the IETE, he actively contributes to the scholarly community as a reviewer for journals such as Physics Open and SLAS Technology. His expertise is widely recognized through his contributions to IEEE-related conferences and a strong presence in Scopus and ORCID databases.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 1-5 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

AI-Powered Climate-Aware Crop Recommendation System using IoT and Machine Learning

[1]Priyanshu Singha Roy, [2]Rishikesh Banerjee, [3]Riddhimoy Mondal, [4]Rupak Raptan, [5]Priyanka Dutta, [6]Sagarika Das

[1,2,3,4,5]UG Students, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[6]Faculty, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Increasing climatic unpredictability has made traditional, experience-based crop selection increasingly unreliable for Indian farmers. This paper proposes an AI-Powered Climate-Aware Crop Recommendation System that combines IoT-based environmental sensing with Machine Learning to deliver real-time, data-driven crop recommendations. The proposed architecture spans an ESP32 microcontroller with DHT22 and soil sensors, a Node.js/PostgreSQL backend, a Random Forest Classifier trained on agricultural crop-recommendation data, and a Next.js dashboard that presents recommendations to the farmer. For this symposium demonstration phase, given hardware and time constraints, the crop-recommendation inference was simulated using Google's Gemini API with manually entered field parameter values in place of live sensor feeds, in order to validate the end-to-end recommendation logic and dashboard pipeline ahead of full hardware and model deployment. Sample simulation results demonstrate the system correctly identifying high-confidence crop matches from a given set of atmospheric and soil-nutrient parameters, indicating strong potential for real-world precision-agriculture deployment.

KEYWORDS: IoT, Machine Learning, Random Forest, Crop Recommendation, Precision Agriculture, Generative AI, Gemini API, ESP32

1. INTRODUCTION

Agriculture remains the backbone of the Indian economy, employing a large share of the rural population and contributing significantly to national GDP. However, increasing climatic unpredictability — irregular monsoons, sudden droughts, untimely floods, and extreme weather events — has made traditional, history-based crop selection increasingly unreliable. Farmers largely continue to rely on seasonal experience and word-of-mouth advice passed down over generations, which is no longer sufficient to cope with the pace of environmental change.

This project proposes an AI-Powered Climate-Aware Crop Recommendation System that combines IoT-based environmental sensing with Machine Learning to deliver real-time, data-driven crop recommendations. By continuously monitoring soil and atmospheric parameters directly from the field and applying a trained predictive model, the system converts raw sensor readings into a clear, actionable recommendation — helping farmers make informed planting decisions without needing any technical expertise themselves.

2. PROBLEM STATEMENT

Farmers today lack access to real-time, localized environmental data and intelligent decision-support tools. Crop selection is still largely based on seasonal experience rather than current soil and atmospheric conditions, which frequently leads to poor yield, wasted inputs, and avoidable economic loss. This gap is especially severe in rural and resource-constrained regions where access to agronomists or advisory services is limited.

*Corresponding Address: sagarikadas1978@gmail.com DOI 10.5281/zenodo.22032998

The key challenges this project addresses are: (i) irregular and unpredictable rainfall causing either water stress or waterlogging, (ii) fluctuating soil moisture directly affecting seed germination and yield, (iii) little to no access to real-time environmental monitoring in most small and marginal farms, (iv) recurring economic losses due to uninformed planting decisions, and (v) the absence of an integrated, low-cost, intelligent system built specifically for agricultural decision support.

3. PROPOSED SYSTEM

We propose a three-tier IoT–Cloud–ML system that spans embedded hardware, backend infrastructure, and a machine learning inference layer. At the sensing layer, an ESP32 microcontroller is interfaced with a DHT22 sensor (temperature & humidity) and a capacitive soil moisture sensor to continuously capture field conditions. These readings are transmitted over Wi-Fi via REST APIs at defined intervals, ensuring the system always reflects near real-time conditions rather than stale historical averages.

At the storage and processing layer, a Node.js/Express backend receives incoming sensor data and persists it in a PostgreSQL database, enabling both live queries and historical trend analysis. A Random Forest Classifier, trained offline on a public agricultural crop-recommendation dataset, is deployed as a REST API microservice to generate a crop prediction along with a confidence score.

Finally, at the presentation layer, a Next.js web dashboard renders the live sensor readings, the recommended crop, its confidence score, and historical trends in a simple, farmer-friendly interface, prioritising clarity over technical detail.

3.1 Simulation Using Generative AI (Gemini API)

Given the limited time available ahead of the symposium, full hardware deployment and offline model training were kept as the next implementation phase. For the purpose of this symposium demonstration, the crop-recommendation inference was simulated using Google's Gemini API through contextual, structured prompting, with field parameter values entered manually in place of live ESP32/DHT22 sensor feeds. This allowed the complete recommendation pipeline — from parameter input to a ranked, confidence-scored crop recommendation with an agronomic rationale — to be validated end-to-end on the existing dashboard, ahead of the planned Random Forest model deployment on real sensor data.

4. SYSTEM ARCHITECTURE

The figure below illustrates the end-to-end data flow of the proposed system, from sensor-level data acquisition to the final recommendation shown to the farmer.

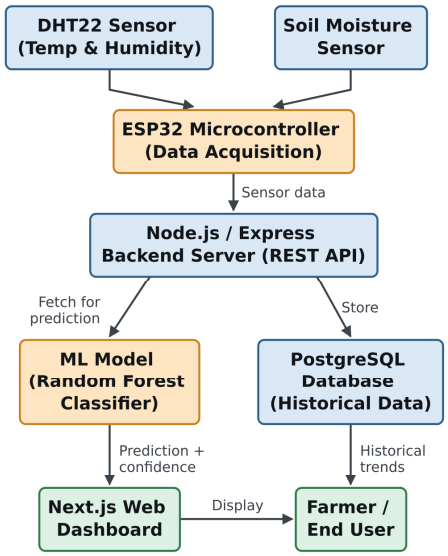


Figure 1: Block diagram of the proposed IoT–Cloud–ML architecture.

The DHT22 and soil sensors feed raw readings to the ESP32, which pushes them to the backend server over REST APIs. The backend persists readings in PostgreSQL for historical logging and feeds the latest reading to the prediction layer, which returns a predicted crop with a confidence score for the Next.js dashboard to render alongside live values and past trends.

5. SIMULATION RESULTS: EXAMPLE INPUT & OUTPUT

A sample interaction with the simulated system is shown below to illustrate how field parameters translate into a crop recommendation.

5.1 Input Parameters

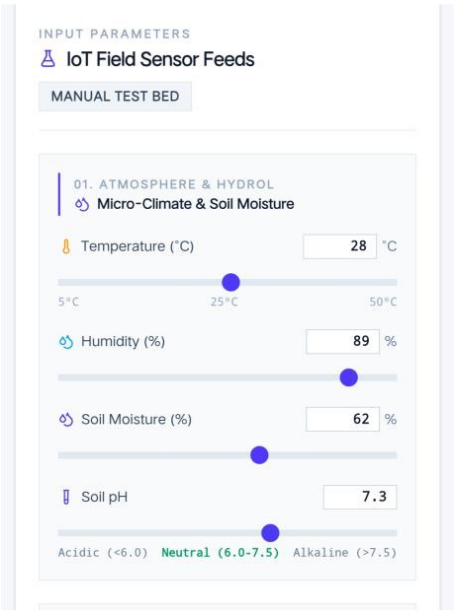


Figure 2a: Manually entered atmospheric and soil-moisture parameters.

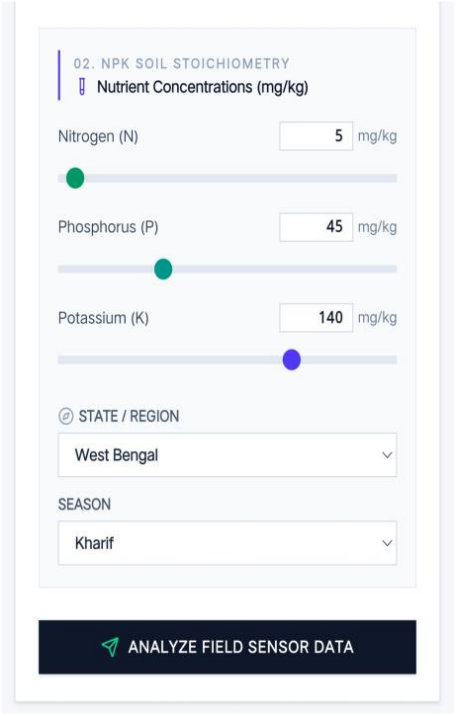


Figure 2b: Manually entered soil NPK and regional/seasonal parameters.

Parameter	*Value*
Temperature	28 °C
Humidity	89 %
Soil Moisture	62 %
Soil pH	7.3 (Neutral)
Nitrogen (N)	5 mg/kg
Phosphorus (P)	45 mg/kg
Potassium (K)	140 mg/kg
Region / Season	West Bengal / Kharif

5.2 Recommendation Output

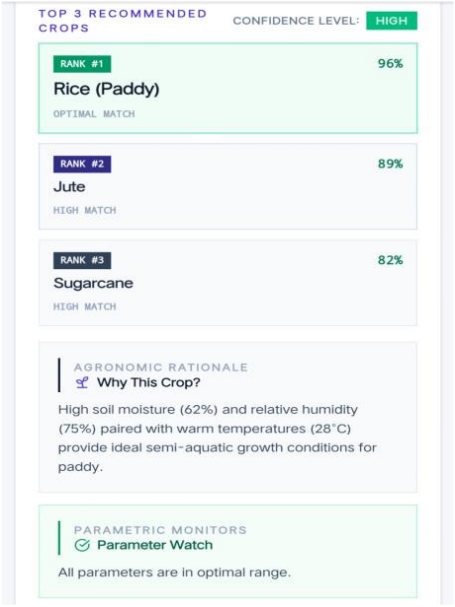


Figure 3: Top-3 recommended crops with confidence scores and agronomic rationale.

Given warm, humid conditions with high soil moisture and NPK levels within a favourable range, the system ranked Rice (Paddy) as the optimal match at 96% confidence, followed by Jute (89%) and Sugarcane (82%), with the accompanying rationale correctly identifying the semi-aquatic growth conditions favouring paddy.

5.3 Soil Nutrient Analysis

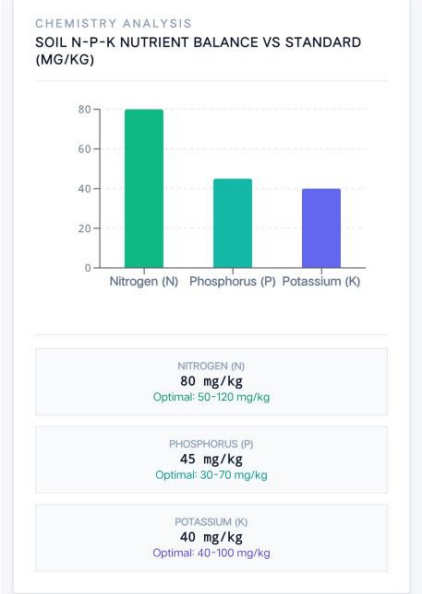


Figure 4: Soil N-P-K nutrient balance compared against optimal reference ranges.

6. IMPLEMENTATION SNAPSHOTS

A few representative code snippets from the system's backend and ML layer illustrate the core data and prediction flow.

6.1 POST Request — Adding Sensor Data to the Database

```
app.post('/api/sensor-data', async (req, res) => {
  const { temperature, humidity, soilMoisture,
          n, p, k } = req.body;
  const result = await db.query(
    `INSERT INTO sensor_readings
     (temperature, humidity, soil_moisture,
      n, p, k, recorded_at)
     VALUES ($1,$2,$3,$4,$5,$6,NOW())
     RETURNING *`,
    [temperature, humidity, soilMoisture, n, p, k]
  );
  res.status(201).json(result.rows[0]);
});
```

6.2 ML Model — Training the Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

X = data[['temperature','humidity','soil_moisture',
          'N','P','K','ph']]
y = data['label']            # crop name
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2)
model = RandomForestClassifier(n_estimators=100,
                               random_state=42)
model.fit(X_train, y_train)
joblib.dump(model, 'crop_model.pkl')
```

6.3 Prediction Endpoint

```
@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    features = [[data['temperature'],
                 data['humidity'],
                 data['soil_moisture'], data['N'],
                 data['P'], data['K'], data['ph']]]
    prediction = model.predict(features)[0]
    confidence = max(
        model.predict_proba(features)[0]) * 100
    return jsonify({'crop': prediction,
                    'confidence': round(confidence,1)})
```

7. ADVANTAGES OVER TRADITIONAL METHODS

Unlike conventional crop selection, which relies on seasonal memory and generalised regional advice, this system grounds every recommendation in the farmer's own field data captured at that moment. Recommendations adapt automatically as weather and soil conditions shift through the season; the inclusion of NPK sensing means nutrient deficiencies are accounted for, not just moisture and temperature; historical data logging allows trend analysis across a full growing season; and the entire stack is built on affordable, open-source hardware and software, keeping the solution realistic to deploy for small and marginal farmers.

8. CONCLUSION

The proposed system demonstrates a practical convergence of embedded systems, cloud infrastructure, and machine learning to address a real agricultural problem. The symposium demonstration, validated through Gemini API-based simulation of the recommendation logic, confirms the viability of the end-to-end pipeline ahead of full hardware and Random Forest model deployment. By replacing static, history-based crop selection with continuous, sensor-driven recommendations that account for both weather and soil nutrient status, the project offers a low-cost, deployable proof-of-concept with clear real-world relevance and strong potential for extension towards genuine precision farming for resource-constrained farmers.

REFERENCES

[1] Chan, J. W. K., (2006). Application of grey relational analysis for ranking material options. *International Journal of Computer Applications in Technology*, 26(4), 210–217. doi: 10.1504/IJCAT.2006.010766

[3] Chan, J. W. K., & Tong, T. K. L., (2007). Multi-criteria material selections and end-of-life product strategy: grey relational analysis approach. *Materials & Design*, 28(5), 1539–1546. doi: https://doi.org/10.1016/j.matdes.2006.02.016

[4] Chatterjee, P., Athawale, V. M., & Chakraborty, S., (2011). Materials selection using complex proportional assessment and evaluation of mixed data methods. *Materials & Design*, 32, 851-860. doi: https://doi.org/10.1016/j.matdes.2010.07.010

[5] Chatterjee, P., & Chakraborty, S. (2012). Material selection using preferential ranking methods. *Materials and Design*, 35, 384–393, doi:10.1016/j.matdes.2011.09.027

[6] Dweiri, F., & Al-Oqla, F. M. (2006). Material selection using analytical hierarchy process. *International Journal of Computer Applications in Technology*, 26(4), 182-189. doi:10.1504/IJCAT.2006.010763

[7] Edwards, K. L. (2011). Materials influence on design: a decade of development. *Materials and Design*, 32, 1073–1080.

[8] Hodgett, R.E., (2016). Comparison of multi-criteria decision-making methods for equipment selection. *The International Journal of Advanced Manufacturing Technology*. 85(5-8), 1145-1157, https://doi.org/10.1007/s00170-015-7993-2

[9] Hwang, C.L. Yoon, K. (1981). *Multiple Attribute Decision Making: Methods and Applications*. New York: Springer-Verlag.

[10] Jahan, A., Ismail, M.Y., Mustapha, F., & Sapuan, S. M. (2010). Material selection based on ordinal data. *Materials and Design*, 31(7), 3180–3187. doi: https://doi.org/10.1016/j.matdes.2010.02.024

[11] Jee, D-H., & Kang, K-J. (2000). A method for optimal material selection aided with decision making theory. *Materials & Design*, 21, 199–206. doi: https://doi.org/10.1016/S0261-3069(99)00066-7

[12] Malczewski, J. (1999). *GIS and multi-criteria decision analysis*. New York: Wiley.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 6-13 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Uncovering Irregularities in National Identity Frameworks: An Algorithmic Approach to Anomaly Detection and LLM Based Predictive Auditing

[1]Sounak Sarkar, [2]Soumyashis Ghosh, [3]Sourodeep Kar, [4]Dr. Tirthadip Sinha

[1,2]UG student, Dept. of Computer Science & Engineering, Haldia Institute of Technology, Haldia, West Bengal

[3]UG student, Dept. of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal

[4]Assistant Professor, Dept. of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal

ABSTRACT

National biometric platforms like India's Aadhaar generate massive daily transaction volumes, making traditional reporting too slow to catch operational anomalies or sudden regional demand surges. To shift from reactive auditing to intelligent real-time monitoring, we introduce a predictive analytics engine using real-world UIDAI operational data (March–December 2025). Built on a decoupled React and FastAPI architecture, the platform features an automated time-series forecasting tournament. Using rolling-origin cross-validation with split-conformal prediction intervals, it delivers reliable capacity forecasts with precise statistical bounds. Concurrently, unsupervised Isolation Forests flag localized anomalies across state and district levels without requiring pre-labeled fraud data. A local LLM translates these technical insights into actionable, plain-language briefs backed by a human-in-the-loop audit trail—enabling swift, targeted field responses at national scale.

KEYWORDS: Isolation Forest, Split-Conformal Prediction, Unsupervised Anomaly Detection, Time-Series Forecasting, LLM, Predictive Auditing, Geospatial Intelligence, Data Governance.

1. INTRODUCTION

Managing a national biometric system involves complex daily hurdles in securing data, distributing resources, and mitigating localized risks. The Unique Identification Authority of India (UIDAI) oversees Aadhaar, the world's largest biometric ID system, which processes millions of daily transactions including new enrolments, biometric updates, and demographic modifications. As the sheer volume and velocity of this data grow, traditional static reporting dashboards become insufficient and inefficient. There is a clear need for an intelligent, scalable system that can surface operational anomalies, predict future workloads with quantified uncertainty, and support transparent data governance.

This paper introduces the Aadhaar Intelligence Engine (AIE), an AI-integrated end-to-end analytical platform built for real-time monitoring and predictive decision-making to address these operational challenges. The system uses a modern, decoupled architecture with a fast frontend and a high-throughput backend. Far beyond a simple visualization dashboard, the backend serves as a dedicated statistical and machine learning engine for the identity infrastructure.

The primary dataset driving this engine was collected from UIDAI, capturing daily operational volumes for new enrolments, demographic updates, and biometric updates across Indian states and districts. The dataset spans the operational period from March 2, 2025 to December 31, 2025. We applied strict data cleaning and normalization pipelines to ensure our statistics were valid. This included the deduplication of anomalous records, normalization of district aliases, reallocation of legacy state jurisdictions, and the quarantining of records with invalid dates or corrupted pin-codes. The system leverages the following primary tech stack and methodologies.

- The backend API is served via FastAPI.
- Data manipulation and numerical operations are handled using Pandas and NumPy.

*Corresponding Address: tirthadip.sinha@gmail.com DOI 10.5281/zenodo.22033304

- Time-series forecasting and anomaly detection models are implemented using Scikit-learn.
- Natural language insight generation is powered by Ollama, serving the Qwen2.5 Large Language Model locally.
- The user interface is built with React and bundled using Vite.
- Data visualization is achieved through Recharts, while complex 2D and 3D geospatial mappings are rendered using Deck. GL layered over a MapLibre base map.

2. LITERATURE REVIEW

The analytical methodologies and infrastructure choices employed in the AIE build directly upon foundational research across machine learning, time-series forecasting, data storage, and large language models.

- Unsupervised Anomaly Detection: To address the challenge of flagging operational deviations without pre-labeled fraud datasets, the system relies on Isolation Forest [1]. Unlike conventional distance or density-based anomaly detection algorithms that compute normal regions, Liu et al. (2008) introduced a tree-structured approach that explicitly isolates anomalies by randomly partitioning feature spaces. Because anomalies are sparse and distinct, they require significantly fewer splits to isolate, resulting in shorter path lengths in the trees. This method provides low computational complexity O(n) and linear memory requirements, making it suitable for national identity transaction logs.
- Time-Series Forecasting & Uncertainty Quantification: Evaluating model performance across geographically diverse regions requires a metric robust to varying data scales. Hyndman and Koehler (2006) introduced Mean Absolute Scaled Error (MASE) to overcome the numerical instabilities of percentage-based metrics (such as MAPE) when encountering values close to zero. By scaling the forecast errors against the in-sample mean absolute error of a naive benchmark, MASE provides a scale-free comparison across diverse datasets [2]. To handle model diversity, the system employs stacked generalization principles [3]. Wolpert's ensemble framework demonstrates that combining distinct learning algorithms via a meta-model minimizes generalization bias and yields superior predictive performance compared to individual base models. Furthermore, traditional point forecasts fail to convey risk without valid confidence limits. Xu and Xie (2023) developed sequential predictive conformal prediction frameworks tailored for non-exchangeable time-series data. By adaptively estimating quantiles over non-conformity scores, split-conformal methods yield distribution-free, statistically guaranteed uncertainty bounds without relying on Gaussian residual assumptions [4].
- Columnar Data Analytics and Local LLM Inference: For high-speed real-time querying, the platform draws on the columnar data storage paradigm originally formalized by Melnik et al. (2010) in the Dremel system (which forms the basis for Apache Parquet). Storing transactional records in a columnar format significantly reduces I/O footprint and speeds up aggregations over large datasets [5]. Finally, generating qualitative briefs while ensuring data privacy is achieved through local large language model deployment. Bai et al. (2023) presented the Qwen technical architecture, establishing high-capability open-weight language models optimized for reasoning and structured text generation. Local inference avoids transmitting sensitive operational data to external APIs while converting deterministic analytical outputs into narrative intelligence [6].

3. DESIGN METHODOLOGY

The AIE is built on a modular architecture, pairing a fast Python backend with a responsive, hardware-accelerated frontend. The methodology driving the system is divided into four primary pipelines: Data Pre-processing, Time-Series Forecasting, Unsupervised Anomaly Detection, and LLM-Driven Governance.

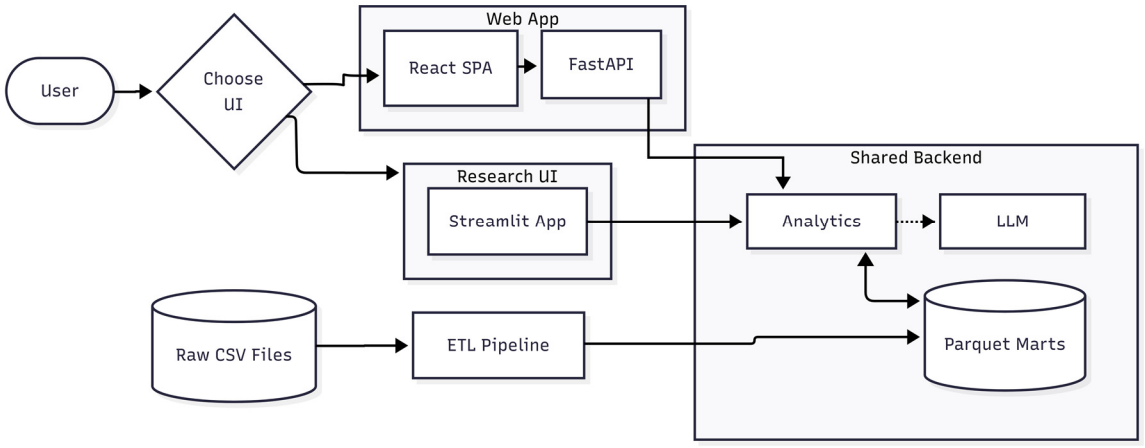


Figure 1: High-level system architecture illustrating the integration of dual user interfaces with the shared analytical engine.

To handle high-velocity data, the system utilizes a strict Extract, Transform, Load (ETL) pipeline resulting in highly optimized Parquet data marts, leveraging high-performance columnar storage principles [5]. Raw CSV shards representing enrolments, demographic updates, and biometric updates are scanned and parsed. Records are grouped by a natural composite key (Date, State, District, Pin-code), and metric duplicates are collapsed via summation to prevent double-counting. The system automatically fixes geographic data errors by applying a strict set of cleaning rules:

- It repairs records where districts were mistakenly entered as states.
- Normalizes thousands of local district aliases to official UIDAI nomenclature.
- Reallocates legacy state borders (e.g., Andhra Pradesh to Telangana splits).

Cleaned DataFrames are persisted as columnar Parquet files (fact_enrol_daily, dim_geo, etc.), allowing the analytics engine to execute in-memory aggregations in milliseconds without re-parsing CSVs.

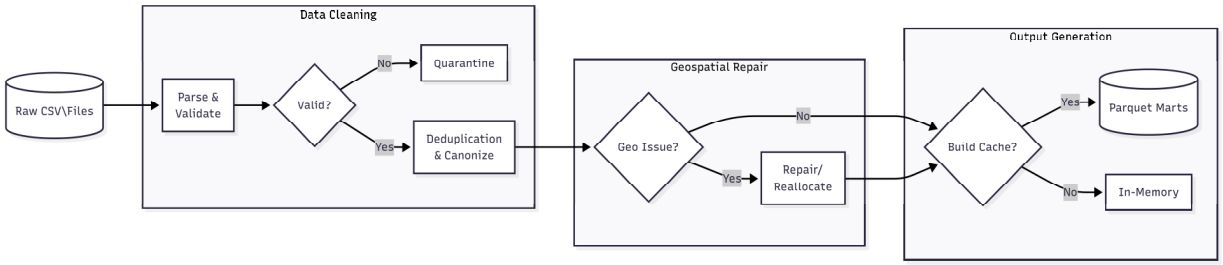


Figure 2: The automated data processing pipeline that cleans, standardizes, and converts raw CSV data into optimized storage.

Risk management relies on proactive, unsupervised machine learning to surface multidimensional outliers. To keep the analysis consistent, all calculations are performed at the state–district level. The engine calculates derived features for each district, including volume volatility (coefficient of variation), demographic-to-enrolment ratios, biometric stress, and week-over-week growth. These features are scaled and fed into an Isolation Forest [1] ensemble. The model flags multidimensional anomalies without needing labeled fraud data, assigning each flagged district a normalized risk score from 10 to 100. The engine aggregates these district-level flags to generate a "State Risk Radar," prioritizing states by their total anomalous volume.

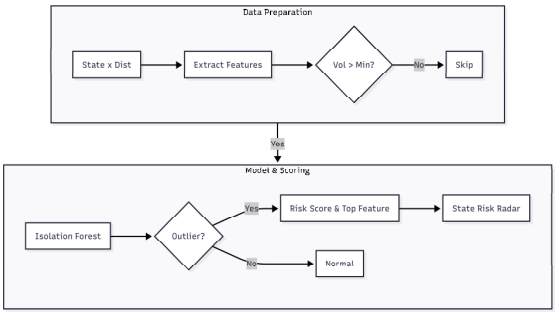


Figure 3: Two-stage anomaly detection process identifying regional outliers without relying on pre-labeled data.

The system includes a data governance service that requires human review while maintaining a continuous record of changes. Rather than mutating the source Parquet or CSV files, administrators review data discrepancies in the UI. Actions such as "Merge" or "Delete" are appended to a JSON patch store and an immutable CSV audit log. These patches are dynamically applied at runtime, ensuring data lineage remains transparent and reproducible.

Capacity planning is driven by a rigorous, rolling-origin cross-validated model tournament. The engine evaluates a diverse registry of models: Moving Averages, damped Drift models, intelligent Ensembles based on stacked generalization concepts [3], and Seasonal Naive baselines (accounting for Indian holidays). Models are back tested using rolling-origin cross-validation. The winning model is selected based on the lowest Mean Absolute Scaled Error (MASE) [2], provided it strictly beats the naive baselines. Instead of assuming Gaussian residuals, the engine applies split-conformal prediction to generate statistically guaranteed uncertainty bounds (upper and lower prediction intervals), classifying them into operational bands (Tight, Directional, or Exploratory).

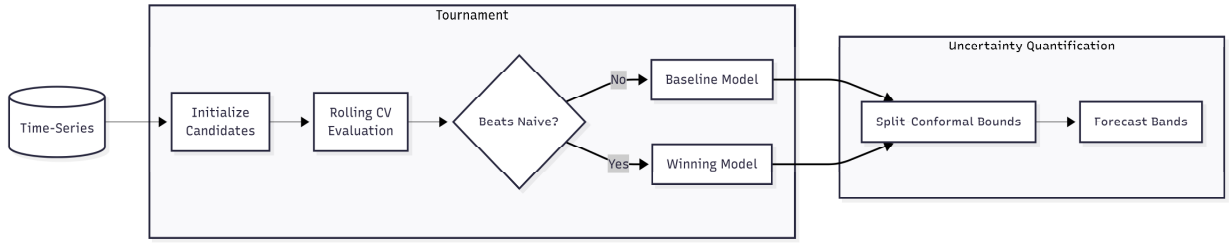


Figure 4: Automated forecasting tournament that pits multiple models against baselines to ensure reliable capacity predictions.

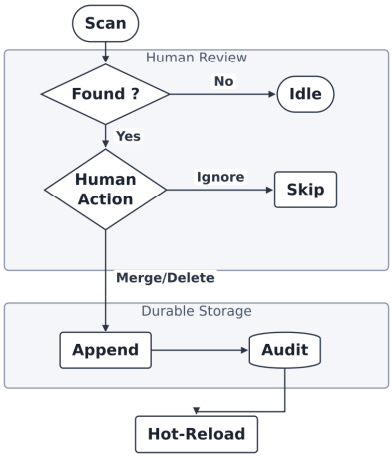


Figure 5: Human-in-the-loop data governance workflow for auditing, resolving, and durably patching operational dataset discrepancies.

To translate complex statistical outputs into actionable intelligence, the engine integrates a local LLM [6]. The Python engine deterministically compiles "evidence" such as winning forecast models, specific anomaly ratios, total volume, etc. The LLM is strictly prompted to process this evidence and generate a structured markdown brief, ensuring it cannot invent or alter underlying mathematical metrics.

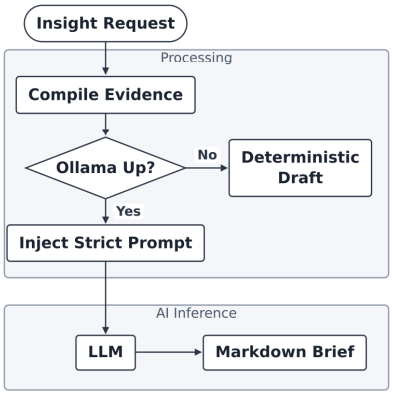


Figure 6: LLM integration framework for transforming quantitative engine outputs into qualitative research briefs.

4. OUTCOMES

A core accomplishment of this work lies in converting massive daily transactional CSV files (Enrolment, Biometric, and Demographic) into an optimized, unified format capable of real-time querying. The automated Extract, Transform, Load (ETL) pipeline reliably handled deduplication, geographic repairs, and Parquet aggregation in sequence with high-speed data processing.

Table 1: Performance of the different processing stages in automated data pipeline, highlighting significant data compression and geographic corrections.

Processing Stage	Action / Metric	Volume (Rows)	Operational Impact
Ingestion	Raw transactions processed	4,938,837	Established the total operational volume.
Deduplication	Natural keys collapsed	738,146	Prevented statistical inflation by summing overlapping metric keys.
Geospatial Repair	District aliases normalized	205,657	Standardized thousands of inconsistent spellings.
Geospatial Repair	Boundary reallocations (AP→TS)	92,491	Corrected legacy state assignments.
Mart Generation	Final aggregated Parquet rows	218,287	Compressed 4.2M rows into a lightweight matrix.

Nearly 300,000 geographic errors were algorithmically corrected upon ingest, ensuring that regional aggregations and geospatial maps remain perfectly accurate for administrators without requiring tedious manual data-entry corrections. Compressing 4.9 million transactions into 218k daily Parquet rows allows the FastAPI backend to load all the necessary analytics data directly into memory. This architectural decision guarantees that all filtering and chart rendering on the React Web App executes with sub-second latency.

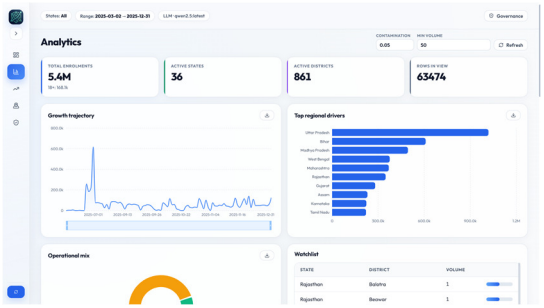


Figure 7: React Web App Dashboard displaying clean, unified State and District demographic data.

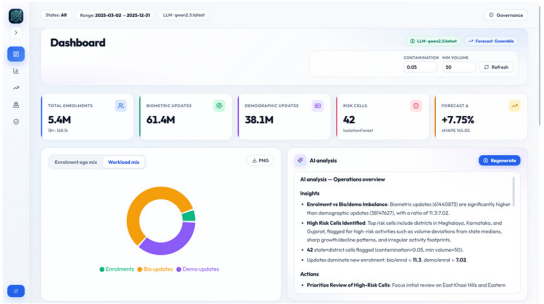


Figure 8: Analytics tab demonstrating sub-second rendering of over 218,000 aggregated Parquet rows.

Rather than relying on rigid, rule-based alerts, the engine detects complex and hidden nationwide anomalies using unsupervised learning specifically the Isolation Forest algorithm. The following radar output aggregates the highest-risk districts, prioritizing state-level interventions based on flagged transaction volume. Karnataka and West Bengal emerged as the highest priority jurisdictions, jointly accounting for over 120,000 anomalous transactions across 9 combined districts.

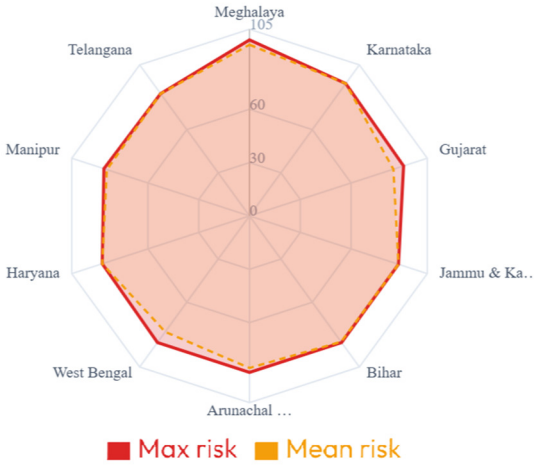


Figure 9: Polar Risk Radar charting the top jurisdictions requiring immediate operational audits.

Table 2: The top 5 highest-risk states automatically flagged by the AIE for immediate operational audits.

State	Flagged Districts	Flagged Volume	Max Risk Score	Mean Risk Score	Primary Anomaly Driver
Karnataka	2	61,274	96.0	87.5	Extreme volume volatility
West Bengal	7	60,139	82.0	77.6	High demographic-to-enrolment ratio
Bihar	2	31,230	84.0	80.5	Extreme volume volatility

State	Flagged Districts	Flagged Volume	Max Risk Score	Mean Risk Score	Primary Anomaly Driver
Telangana	1	30,830	88.0	88.0	High biometric update stress
Meghalaya	3	29,773	99.0	89.3	Volume outlier vs state median

Effective operational planning requires highly reliable capacity forecasting method like Time-Series Forecasting. Instead of relying on a single fixed algorithm, the forecasting backend runs a rolling-origin cross-validation comparison across several statistical models. Models are ranked by Mean Absolute Scaled Error (MASE). More complex models must outperform simple baselines such as Seasonal Naive or Moving Average on the data being examined.

Table 3: Ensemble model outperforms single-architecture baselines.

Rank	Model Architecture	MASE	MAPE (%)	RMSE	Decision Band
1	Ensemble (Median Stack)	0.9010	188.6	34,916	Directional
2	Moving Average	0.9863	182.4	36,185	Directional
3	Drift	0.9945	200.0	36,542	Directional
4	Seasonal Naive (Baseline)	1.0383	190.0	40,658	Directional

The automated Ensemble model utilizing median stacking strategies consistently outperformed all single-architecture baselines, reducing the scaled error (MASE) below 1.0, indicating it actively learned the volume patterns rather than just echoing recent history. Split-conformal prediction intervals applied to the residuals of the winning model give a statistically valid 90 % confidence band around the forecasts. Using the upper prediction limits, administrators can safely allocate servers, update kits, and staff for upcoming traffic spikes without over-allocating the budget. While the Ensemble model dominated at the national aggregate level, the tournament proved crucial at the micro-level. For example, when predicting capacity for Bihar (characterized by extreme volume volatility) and Sikkim (a low-volume, sparser dataset), the system automatically discarded the complex ensemble in favor of a robust drift model. This demonstrates the engine's ability to guard against overfitting by dynamically reverting to simpler linear trendlines when they mathematically out-score complex architectures in noisy regions.

Figure 10: Time-series forecasting shows the Ensemble model's trajectory and 90% confidence bounds.

Deploying the model on-premise safeguards sensitive operational data from third-party exposure, aligning strictly with national security standards. Constraining the LLM to pre-computed JSON payloads eliminates numerical hallucinations in the generated summaries from the final report. Operators interacting with the React Web App instantly receive human-readable, actionable summaries of a state's health

and accelerating response times without requiring dedicated data science expertise.

Table 4: Complex mathematical payloads are shifted into actionable operator directives in processing stages.

Processing Stage	Output Example (Karnataka Enrolments)	Operational Impact
Statistical Engine Output (Deterministic)	{"anomaly": true, "vol": 61274, "risk": 96.0, "driver": "volatility"}	Calculates the exact statistical metrics.
LLM Output Brief (Qualitative)	"Karnataka has been flagged for a critical anomaly (Risk Score: 96.0). Our engine detected extreme volume volatility resulting in 61,274 flagged transactions. An immediate regional audit is recommended."	Translates complex statistics into clear, actionable instructions for administrators.

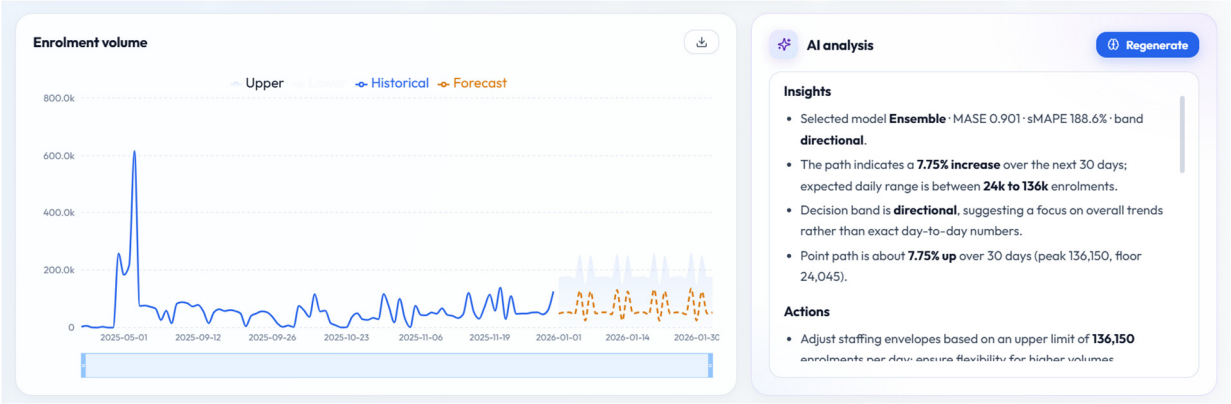


Figure 11: LLM-generated operational brief providing zero-shot narrative insights directly on the dashboard.

5. CONCLUSION

This work marks a significant step forward in national identity management, shifting the paradigm from slow, reactive spreadsheet reporting to an agile, automated, and AI-driven intelligence platform. By seamlessly uniting automated data engineering, unsupervised anomaly detection, and dynamic forecasting, the Aadhaar Intelligence Engine empowers administrators to proactively monitor millions of daily transactions with sub-second responsiveness. Crucially, by running Large Language Models locally on the ground, the system safeguards strict data privacy while instantly translating complex statistical patterns into clear, plain-language directives for everyday operators.

While the current platform demonstrates high operational efficiency, we recognize several realistic boundaries. First, because models currently process data as daily totals, brief intra-day or hourly capacity spikes and bottlenecks can still go undetected. Secondly, the forecasting algorithms rely primarily on historical transaction volumes. As a result, unexpected external events—such as sudden local holidays, severe weather disruptions, or regional internet outages—are not yet factored into workload predictions. Finally, while local LLM deployment ensures absolute data privacy, running these models locally introduces a hardware bottleneck on dashboard performance in environments lacking dedicated GPU hardware. To build a more resilient system, future iterations will focus on our three key enhancements. We plan to integrate external dataset streams such as local infrastructure health indices, census demographics, and regional event calendars, to make workload forecasts significantly more adaptive. Extending geospatial analytics down to the pin-code level will allow administrators to pinpoint under-served areas and deploy physical assets, such as mobile enrolment vans, directly where demand is highest.

Also, transitioning the architecture to a streaming framework (e.g., Apache Kafka) will enable the system to catch transactional irregularities the exact moment they occur.

Acknowledgement

Due to applicable national privacy, security, and data-protection regulations, public-sharing of the raw data is restricted in India. The authors

gratefully acknowledge the Unique Identification Authority of India (UIDAI) for providing the real-world data used in this work.

REFERENCES

[1] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422. https://doi.org/10.1109/ICDM.2008.17

[2] Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

[3] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1

[4] Xu, C., & Xie, Y. (2023). Sequential Predictive Conformal Inference for Time Series. *Proceedings of the 40th International Conference on Machine Learning*, 38707–38727. https://doi.org/10.48550/arXiv.2212.03463

[5] Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). Dremel: Interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2), 330–339. https://doi.org/10.14778/1920841.1920886

[6] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, R., Han, Y., Fei, H., & Zhu, T. (2023). *Qwen Technical Report*. arXiv. https://doi.org/10.48550/arXiv.2309.16609

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 14-18 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Design and Implementation of an Intelligent Machine Learning-Based Weather Prediction Application

[1]Dr.Sagarika Das, [2]Nikita Shaw, [3]Nabarun Naskar, [4]Mith Biswas, [5]Md. Raquif, [6]Mayukh Santra

[1]Faculty , Dept.of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[2,3,4,5,6]UG student, Dept. of Electronics and Communication Engineering , Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Accurate weather forecasting is essential for agriculture, transportation, disaster management, and environmental planning. This study presents a comparative weather forecasting model using three time-series and deep learning approaches: Autoregressive Integrated Moving Average (ARIMA), Seasonal ARIMA (SARIMA), and Long Short-Term Memory (LSTM). Historical weather data containing variables such as temperature, humidity, atmospheric pressure, and rainfall are preprocessed to handle missing values, normalize the data, and identify temporal patterns. ARIMA is applied to capture linear trends and short-term dependencies, while SARIMA incorporates seasonal variations in weather patterns. LSTM, a recurrent neural network architecture, is employed to learn complex nonlinear relationships and long-term temporal dependencies. The models are trained and evaluated using appropriate error metrics, including Mean, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The comparative analysis identifies the most effective approach for accurate weather prediction and demonstrates the potential of machine learning and deep learning techniques for reliable forecasting.

KEYWORDS: ARIMA, SARIMA, LSTM, Time Series Forecasting, Deep Learning, Machine Learning.

1. INTRODUCTION

Weather forecasting is one of the most important as well as a challenging task in the field of science and technology in daily life. Weather conditions such as temperature, rainfall, humidity, wind speed, and pressure directly affect human activities, agriculture, transportation, disaster management, aviation, marine operations, and energy consumption. Due to these variations, predicting future weather accurately is not easy. In many cases, people depend on general weather reports from websites, mobile applications, or news channels. However, for academic, local, and research-based applications, it is useful to build a system that can analyze available weather data and generate predictions automatically.In this project, a smart weather forecasting system is developed using statistical models like ARIMA, SARIMA, and LTSM which collects and uses previous weather data for forecasting future weather. RMSE is used to compare all the models and generate lowest error. This project is useful for understanding the basic concept of weather forecasting using AI/ML. The current scope of the project includes forecasting temperature for the next 10 days. However, the present system is limited to temperature prediction only. It does not forecast other weather parameters such as rainfall, humidity, wind speed, pressure, or cloud condition. In the future, the project can be extended by adding more weather parameters and using a larger dataset. It can also be improved by using real-time weather data from sensors pr online weather APIs.

2. METHODOLOGY

The methodology of this project explains step-by-step procedure to develop the smart weather forecasting system. This project uses AI/ML based models such as ARIMA, SARIMA, LSTM etc. The methodology of this project is extended by adding an application layer after model training and prediction.

2.1 System Architecture:

*Corresponding Address: sagarikadas1978@gmail.com DOI 10.5281/zenodo.22037449

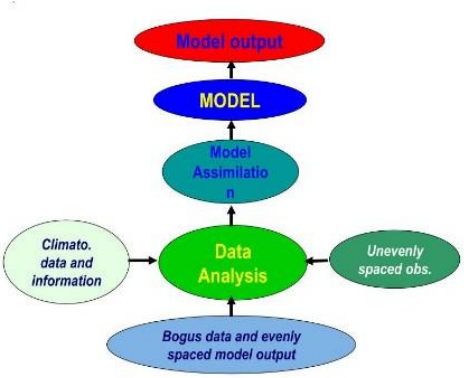


Figure 1: ML and Weather Forecasting Flowchart

The system architecture starts with input weather records and ends with final forecast output. This system reads the data set and automatically detects the date-time column and temperature column. After loading the dataset, the date-time column is converted into proper date-time format. The hourly temperature data is then converted into daily average temperature data.

The system uses three forecasting models: ARIMA, SARIMA, and LSTM. ARIMA is used to identify general trends in the temperature data. SARIMA is used to identify both trend and seasonal patterns. LSTM is used as a deep learning model to learn long-term patterns from previous temperature values. After training all models generates prediction which is evaluated using RMSE. It reduces the prediction error. The model with lowest RMSE is selected as the best model.

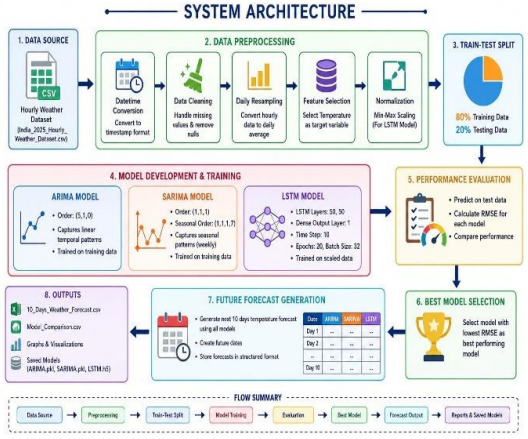


Figure 2: System Architecture Flowchart

2.2 Data Description

The quality and reliability of the weather forecasting depend heavily on the availability on the historical weather observations. In the given code, the dataset file named India_2025_Hourly_Weather_Dataset.csv. is loaded using the Pandas library.

Table-1: Data set Description

Property	Description
Dataset Name	India 2025 Hourly Weather Dataset
File Format	CSV (Comma-Separated Values)
Total Records	8,760
Total Features	5
Data Type	Time-Series Data
Observation Frequency	Hourly
Forecast Target	Temperature
Year Covered	2025

2.3 Feature Engineering

Feature engineering is an important step in this project because it converts raw weather data into a suitable format for machine learning models because hourly weather data used in this project can not be directly used for daily forecasting.

The first step is date-time conversion where the date column is converted into proper date-time format. After that it is set as the index of the dataset. In the second step all the hourly temperature values are converted into daily average temperature. This step is known as daily sampling.

After sampling, some missing values are removed to make the dataset clean. This is called missing value handling. The fourth step is train-test splitting, where the daily temperature data is divided into training and testing parts. For the LSTM model the temperature values are scaled between 0 to 1 using MinMaxScaler. Thus, these overall feature engineering steps make the dataset clean, structured, and suitable for AI/ML models.

2.4 Machine Learning Models

It is the most important part of the whole project where different time-series forecasting models are used to predict future temperature values. As the weather data is time dependent time-series forecasting models are suitable for the weather prediction. In this project, three models are used for forecasting:

2.4.1 ARIMA Model

ARIMA stands for AutoRegressive Integrated Moving Average. It is a commonly used statistical

model useful when the data contains a pattern or trend over time in time-series forecasting. It has three main parts:

A. AR- AutoRegressive Part: It means that the future value is predicted using previous values. The model studies the last few days temperature values and predicts the future values.

B.I- Integrated Part: It is used to make the data more stable. If the data shows increasing or decreasing trend over time ARIMA used differencing to remove all those trends and makes it suitable for forecasting.

C. MA-Moving Average part: The moving average part considers the previous forecasting errors and corrects itself by learning from past prediction mistakes. This improves the quality of the future predictions. The general equation of an ARIMA model is:

$$ARIMA(p, d, q)$$

Where: p = number of past values used, d = number of differencing operations , q = number of past error terms used In the given project code, the ARIMA model is created using the order:

2.4.2 SARIMA Model

SARIMA stands for Seasonal AutoRegressive Integrated Moving Average. SARIMA contains both non-seasonal and seasonal parameters. The non-seasonal part works like ARIMA, while the seasonal part captures repeating patterns in the time-series data. The general equation for the SARIMA model is:

$$SARIMA(p, d, q)(P, D, Q)s$$

where:

- p=autoregressive order
- d=differencing order
- q=moving average order
- P=seasonal autoregressive order
- D=seasonal differencing order
- Q=seasonal moving average order
- S=seasonal period

2.4.3 LTSM Model

LSTM stands for Long Short-Term Memory. It is a deep learning model used for sequence prediction problems. It is a special type of Recurrent Neural Network (RNN) which can learn patterns from previous data and remember important information for longer time. Before using LSTM, the data is scaled using MinMaxScaler into range between 0 and 1. Unlike traditional models, LSTM can capture complex and non-linear patterns in the dataset. As temperature may depend upon previous temperature LSTM learns these relations from historical data. After scaling, the data is converted into sequences. In this project, the sequence length is 10 days. The LSTM architecture used in this project contains:

- First LSTM layer with 50 units
- Second LSTM layer with 50 units
- Dense output layer with 1 neuron

The first LSTM layer uses return_sequences=True, which means it passes sequence output to the next LSTM layer. The second LSTM layer processes this information and sends the final output to the Dense layer. The Dense layer gives the final predicted temperature value.

The model is compiled using:

- Optimizer: Adam
- Loss function: Mean Squared Error

The Adam optimizer helps the model update its internal weights during training. The model is trained for 20 epochs with a batch size of 32. An epoch means one complete training cycle through the dataset. Batch size means the number of samples processed before updating the model weights.

2.5 Model Training

After data processing and feature engineering hourly data set is converted into daily average temperature data. 80% data has been used for training and 20% for testing. ARIMA and SARIMA are trained directly on the daily temperature data. LSTM requires additional preprocessing, such as scaling and sequence creation. After training, all three models generate predictions.

2.6 Model Evaluation using RMSE

The evaluation metric RMSE stands for Root Mean Squared Error. In this step RMS calculates the difference between actual temperature values.

3. RESULT AND IMPLEMENTATION

The implementation of the project includes two parts. The first part is the machine learning implementation, where ARIMA, SARIMA, and LSTM models are trained and tested. The second part is the application implementation, where the saved results are displayed using a Streamlit web application.

Table 2: Performance metrics

Model	**RMSE Value**	**Performance Analysis**
ARIMA	3.1858381427	Suitable for trend-based forecasting
SARIMA	2.3163921426	Suitable for seasonal pattern forecasting
LSTM	0.6074101536	Suitable for long-term sequence learning

Table 3: Temperature forecasting comparison among all models

Date	**ARIMA Forecast**	**SARIMA Forecast**	**LSTM Forecast**
2026-01-01 00:00:00	23.2762	22.8865	19.3321
2026-01-02 00:00:00	23.3138	22.8307	19.3465
2026-01-03 00:00:00	23.2964	22.6838	19.3768
2026-01-04 00:00:00	23.2014	22.5488	19.4028
2026-01-05 00:00:00	23.2250	22.6651	19.4368
2026-01-06 00:00:00	23.2184	22.4612	19.4614
2026-01-07 00:00:00	23.2053	22.3711	19.4781
2026-01-08 00:00:00	23.1815	21.9970	19.5119
2026-01-09 00:00:00	23.1736	21.9487	19.5602
2026-01-10 00:00:00	23.1753	21.7936	19.6037

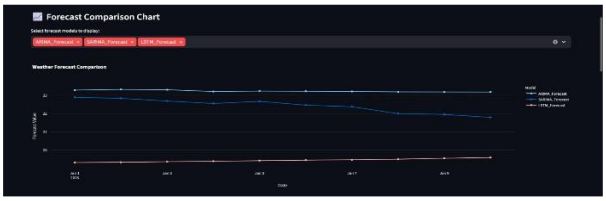


Figure 3: Visualization of performance metrics among all models

Figure 4: Temperature Forecasting using Predictor

4. CONCLUSION & FUTURE WORK

The Smart Weather Forecasting System developed in this project provides a simple and effective method for forecasting future temperature using Artificial Intelligence and Machine Learning techniques. The major conclusions of this project are:

a) Historical data supports future temperature prediction.

b) Time-series models are suitable for weather forecasting.

c) ARIMA, SARIMA, and LSTM offer different forecasting approaches.

d) RMSE identifies the most accurate model.

e) Graphs and CSV outputs simplify result analysis.

The system uses graphs to clearly visualize original data, forecasts, and RMSE comparisons. Future improvements can enhance the system's accuracy, practicality, and real-world usability.

REFERENCES

[1] BRAC University, (2022). Weather Forecasting using Deep Learning. BRAC University Thesis/Project Report, Department of Computer Science and Engineering.

[2] Waqas, M., (2025). Artificial Intelligence and Numerical Weather Prediction. ScienceDirect.

[3] Karevan, Z., & Suykens, J. A. K., (2018). Spatio-temporal Stacked LSTM for Temperature Prediction in Weather Forecasting. arXiv preprint arXiv:1811.06341. doi: 10.48550/arXiv.1811.06341.

[4] International Journal of Advanced Research in Science, Communication and Technology, (2025). Article on machine learning-based weather forecasting. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT).

[5] Hossain, M. L., Shams, S. M. N., & Ullah, S. M., (2025). Time-series and deep learning approaches for renewable energy forecasting in Dhaka: A comparative study of ARIMA, SARIMA, and LSTM models. Discover Sustainability, 6, 775. doi: 10.1007/s43621-025-01733-5.

[6] Rozas Larraondo, P., (2019). Application of Machine Learning Techniques to Weather Forecasting. Doctoral Thesis, University of the Basque Country UPV/EHU, Department of Computer Science and Artificial Intelligence.

[7] Weather Forecasting using Machine Learning Techniques, (2020). Research/technical document on machine learning-based weather forecasting.

[8] Weather Forecasting using Machine Learning, (2026). Google Colab implementation/project. Google Colaboratory.

[9] Sanders, W. S., (2017). Machine Learning Techniques for Weather Forecasting. M.S. Thesis, University of Georgia, 136 pages.

[10] Karevan, Z., & Suykens, J. A. K., (2018). Spatio-temporal Stacked LSTM for Temperature Prediction in Weather Forecasting. arXiv preprint arXiv:1811.06341. doi: 10.48550/arXiv.1811.06341.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 19-25 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

ENERGY EFFICIENT SMART STREET LIGHTING SYSTEM USING ARDUINO

[1]Soham Dey, [1]Srijita Patra, [1]Supratim Datta, [1]Sreya Maity, [2]Dr. Tirthadip Sinha

1UG student, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal
2Assistant Professor, Dept. of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal

ABSTRACT

Conventional Street lights waste significant power by running at maximum output all night, regardless of actual road activity. To solve this, we developed a budget-friendly, adaptive smart lighting system using an Arduino Uno integrated with an LDR, PIR, and ultrasonic sensors. The LDR disables operation during daylight. At night, the system keeps LED lights dimmed at PWM 80. When approaching traffic or pedestrians are detected within a 120 cm range, the Arduino dynamically scales brightness to full capacity (PWM 255) and triggers an alert. Simulation results confirm seamless transition between idle and active states. By aligning illumination with real-time need, this solution slashes energy overhead, reduces operating costs, and enhances municipal safety across urban and residential areas.

KEYWORDS: Energy Efficiency, Intelligent Lighting, Adaptive Illumination, Arduino, Automated Lighting Control.

1. INTRODUCTION

Smart street lighting has become an essential pillar of modern urban infrastructure, playing a crucial role in maintaining road safety and driver visibility while optimizing municipal power usage [1], [2]. Historically, public lighting designs focused primarily on balancing energy savings with visual comfort and strict safety guidelines [3]. Kostic and Djokic emphasized that energy-efficiency measures must not compromise driver visibility or pedestrian safety, advocating for adaptive lighting criteria tailored to real-time road conditions [3]. Despite these recommendations, conventional street lighting networks still largely rely on fixed-intensity lamps operating on static timer schedules regardless of actual traffic density or natural daylight [2], [4]. This static operation leads to excessive energy waste, accelerated hardware degradation, and elevated operational costs [2], [4].

To overcome these traditional limitations, significant research has focused on intelligent, responsive lighting architectures [4], [5]. Early networked approaches, such as the ZigBee-based remote control system developed by Leccese, demonstrated how wireless sensor networks (WSNs) can enable real-time monitoring and centralized lamp control to lower energy consumption [6]. To handle more complex traffic patterns, Zhang *et al.* proposed a group decision-making model that aggregates data across distributed sensors to autonomously regulate streetlights based on collective environmental feedback rather than isolated triggers [7]. Building on these concepts, modern adaptive street lighting frameworks combine low-power microcontrollers with ambient light and motion sensors to deliver full illumination when activity is present and dim during low-demand periods [1], [5], [8], [9].

In practical field observations of secondary roadways, university campuses, and isolated highways, traffic drops significantly during late-night hours. Operating conventional fixtures at maximum brightness during these idle periods represents a continuous drain on power grids [1], [2]. While advanced wireless mesh networks [6] and distributed decision models [7] offer robust capabilities, they often introduce higher installation costs and network overheads. This highlights the need for low-cost, decentralized architectures that provide immediate, adaptive responsiveness for localized deployments [1], [5].

DOI 10.5281/zenodo.22037474

*Corresponding Address: tirthadip.sinha@gmail.com

To address these practical constraints, this paper presents a budget-friendly, energy-efficient smart street lighting system that automatically controls LED intensity using an Arduino Uno microcontroller integrated with Light Dependent Resistors (LDR), Passive Infrared (PIR) sensors, and ultrasonic sensors [1], [8]. During daylight hours, the system keeps the lights completely powered off (PWM = 0). At night, the system defaults to an energy-saving standby mode (PWM = 80). When an approaching vehicle or pedestrian is detected within a designated 120 cm range, the controller rapidly scales illumination to full capacity (PWM = 255) and triggers an alert [1], [8]. By matching light output directly to real-time road demand, the proposed framework significantly cuts electricity consumption while maintaining public safety and visual standards [1], [3].

2. DESIGN METHODOLOGY

This system is developed to control the street light according to the surrounding light condition and the presence of a nearby object. The Arduino Uno R3 is used as the main controller, while different sensors provide the information required for controlling the light. The working process of the system is carried out continuously so that the brightness can change whenever the surrounding condition changes. Arduino Uno R3 is used as the main controller in the system. An LDR detects the ambient light level and controls the day/night operation. A PIR sensor and ultrasonic sensor detect movement and objects within a predefined distance of 120 cm. When an object is detected, the Arduino increases the LED intensity using PWM from 80 to 255. A 16×2 LCD displays the system status, while a buzzer provides an audio indication during detection. Thus, the system combines light sensing, motion detection, distance measurement, and PWM-based control to provide adaptive street lighting and reduce unnecessary energy consumption.

First, the Arduino Uno R3 is connected to the LDR, PIR sensor, ultrasonic sensor, LED street light, 16×2 LCD, and a buzzer. Each component performs a specific function in the system. The sensors provide input to the Arduino, and the Arduino processes these inputs before deciding the required light intensity. The LDR is used to determine the amount of light present around the street light. Its reading is continuously monitored by the Arduino. When the surrounding light is sufficient, the system considers it as daytime and keeps the LED switched OFF. When the light level decreases, the system changes to night operation. After entering night mode, the LED does not immediately operate at its maximum brightness. Instead, it is kept at a lower intensity using PWM. In the developed system, the initial PWM value is 80. This provides basic illumination while avoiding the need to operate the light at full brightness when there is no nearby traffic. During night operation, the PIR sensor is used to sense movement in the surrounding area. The ultrasonic sensor is used along with it to determine whether an object is within the selected detection range. In this work, the distance limit is set to 120 cm. These sensors help the controller identify when a vehicle or moving object approaches the street light.

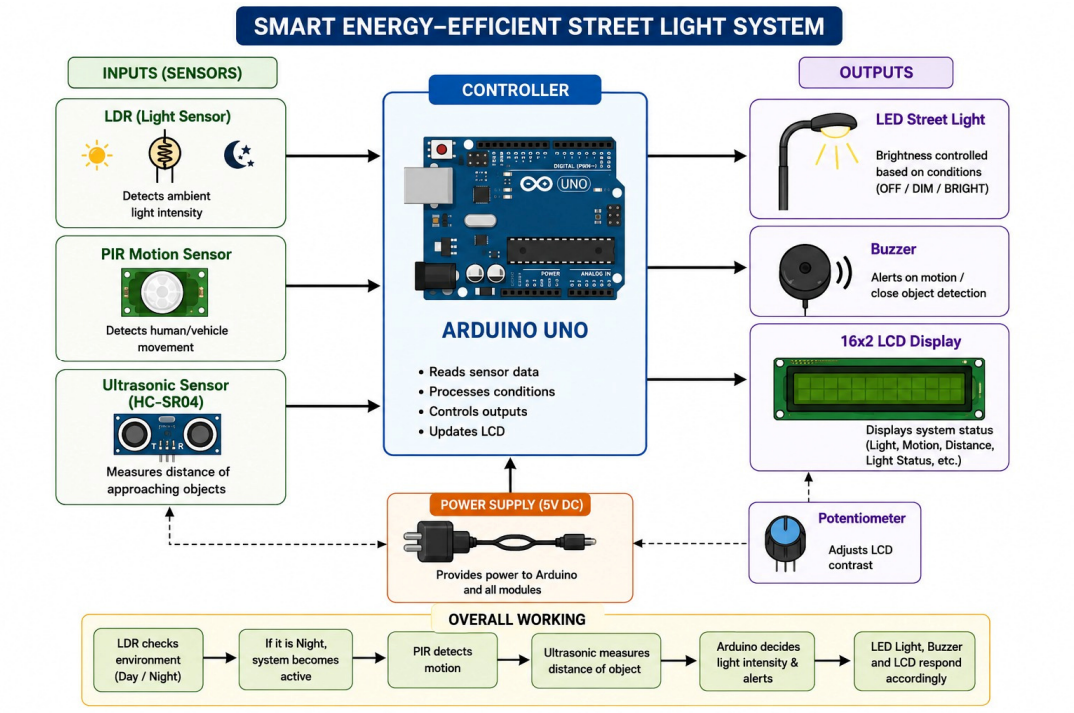


Figure 1: Block Diagram of the Proposed Smart Energy-Efficient Street Light System

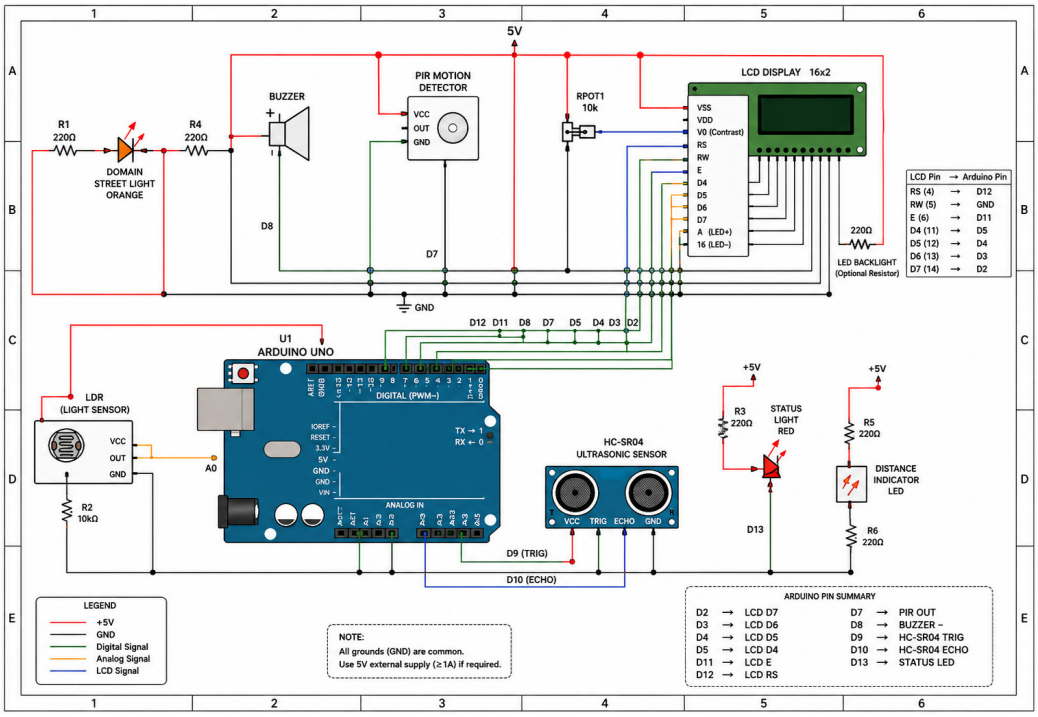


Figure 2: Detailed Circuit Connection Diagram of the Proposed Smart Street Light System

When the sensors detect an object within the specified range, the Arduino changes the PWM value of the LED from 80 to 255. As a result, the street light reaches its maximum brightness and provides better visibility for the detected object. Once the object moves away from the detection range, the system returns the LED to its lower brightness level. The 16×2 LCD is used to show the current condition of the system. It can indicate whether the system is operating in day mode, night mode, or detecting an object. The piezoelectric buzzer provides an additional indication when the required detection condition occurs. This makes it easier to observe the

response of the system during testing. After completing one sensing and control cycle, the Arduino again reads the sensor values and repeats the same process. Therefore, the street light does not depend on a fixed operating schedule. Its condition changes according to the actual readings received from the sensors. Through this continuous process, the system provides higher brightness when required and lower brightness when the road is clear, helping to reduce unnecessary energy consumption.

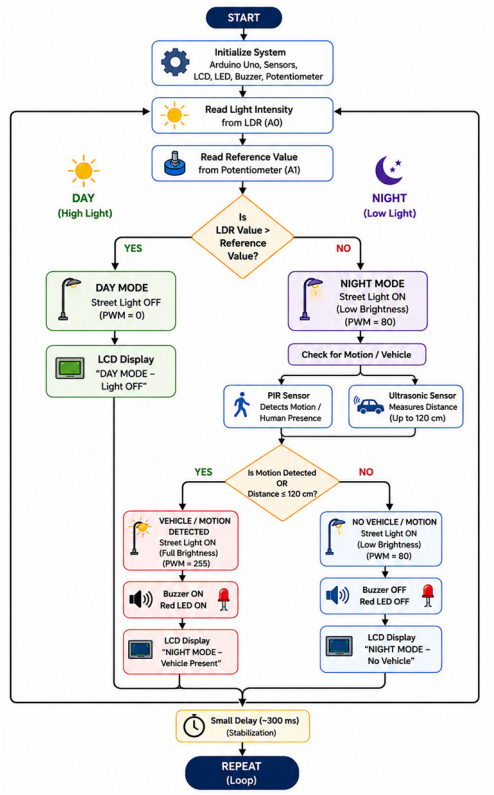


Figure 3: Control Flowchart of the Smart Energy-Efficient Street Light System

The block diagram in Figure 1 shows the main components and signal flow of the proposed smart street light system. The LDR, PIR, and ultrasonic sensors provide the required input to the Arduino Uno for checking light conditions, movement, and object distance. Based on these inputs, the Arduino controls the LED street light using PWM. The 16×2 LCD displays the system status, while the buzzer gives an alert during detection. A 5V supply powers the circuit, and the potentiometer is used for LCD contrast adjustment. Overall, the diagram gives a simple view of how the different parts work together. Figure 2 shows the circuit connections used for our work. The Arduino Uno is connected to the LDR, PIR sensor, HC-SR04 ultrasonic sensor, LCD, buzzer, and LED indicators. The LDR is connected to A0, while the PIR output is connected to D7. The ultrasonic sensor uses D9 and D10 for trigger and echo. The LCD is connected through D2–D6, D11, and D12, and D8 is used for the buzzer. The circuit uses a common 5V supply and ground for the connected components. Figure 3 explains the control sequence used in the proposed street light system. The system first starts all the connected components and takes the LDR and potentiometer readings. These values are used to decide whether the surrounding condition is suitable for day mode or night mode. In day mode, the street light stays OFF. At night, the light is kept at a low level until the PIR or ultrasonic sensor detects a person or vehicle within 120 cm. When detection occurs, the light changes to full brightness (PWM 255) and the buzzer gives an indication. If nothing is detected, the light continues at PWM 80. The LCD also changes its message according to the current condition. After a short delay, the controller checks the sensors again and continues the same operation.

The following components were used to build and test the proposed street lighting system:

- Arduino Uno R3 – Main controller of the system.
- LDR – Detects the surrounding light level.
- 16×2 LCD – Displays the current system condition.
- 10 kΩ Potentiometer – Adjusts the LCD contrast.
- PIR Sensor – Detects movement in the monitored area.
- HC-SR04 Ultrasonic Sensor – Measures the distance of a nearby object.
- Orange LED – Represents the street light.
- Red LED – Provides system status indication.
- Piezo Buzzer – Gives an alert when detection occurs.
- 220 Ω Resistors – Used to limit current through the LEDs.
- 10 kΩ Resistor – Used with the LDR for voltage sensing.

3. RESULTS AND ANALYSIS

The circuit was tested in the simulation by changing the light condition and the position of the detected object. The main things checked were the street-light brightness, LCD message, buzzer response, and distance measured by the ultrasonic sensor. The results for the different conditions are given below.

(i) Day Condition: When enough light falls on the LDR, the system treats the condition as daytime. The street light stays OFF (PWM = 0). The LCD shows the day-mode message. This confirms that the light does not remain ON when natural light is already available.

(ii) Night Condition with No Vehicle: When the LDR reading indicates low light, the system changes to night mode. If there is no vehicle or moving object near the sensor, the street light stays at low brightness (PWM = 80). The buzzer is OFF and the LCD shows the corresponding night-mode status.

(iii) Vehicle Detection: A vehicle or object was placed within the detection range. When the distance became 120 cm or less, the system detected it and increased the street-light output to PWM = 255. At the same time, the buzzer turned ON and the LCD changed to the vehicle-detected message.

(iv) Distance Check: The HC-SR04 was used to check the distance of the object from the sensor. The measured distance was compared with the 120 cm limit set in the program. This allowed the Arduino to decide whether the light should remain dim or change to full brightness.

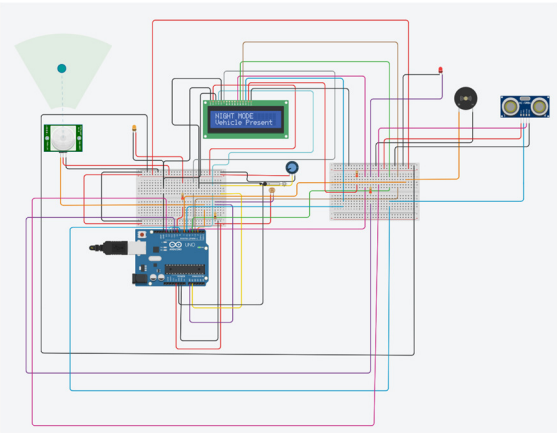


Figure 4: Simulation Result for Vehicle and Motion Detection

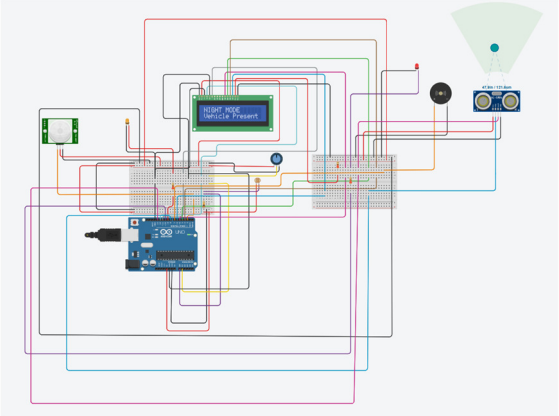


Figure 5: Simulation Result for Vehicle Detection with Distance Measurement

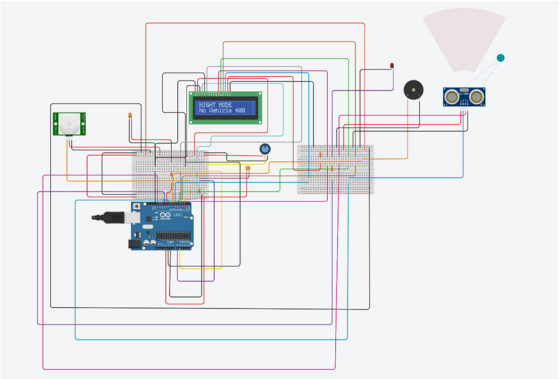


Figure 6: Simulation Result for Night Mode with No Vehicle

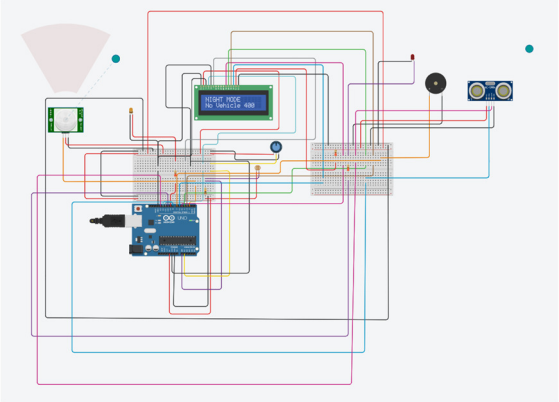


Figure 7: Simulation Result for Night Mode with No Motion Detection

Table 1: Observed Sensor Readings and System Response

SL. No.	*LDR VALUE*	*Distance (cm)*	*Motion*	*Mode*	*LED PWM*	*Buzzer*
1	950	400	No	Day	0	OFF
2	550	400	No	Night	80	OFF

SL. No.	*LDR VALUE*	*Distance (cm)*	*Motion*	*Mode*	*LED PWM*	*Buzzer*
3	400	150	No	Night	80	OFF
4	250	120	Yes	Night	255	ON
5	100	80	Yes	Night	255	ON
6	420	400	No	Night	80	OFF

The Table 1 gives the readings taken during different test conditions. In daylight, the street light stays OFF with PWM 0. At night, when no vehicle is present, the light works at PWM 80. When a vehicle or moving object comes within 120 cm, the light changes to PWM 255 and the buzzer turns ON. The readings show that the system changes the light intensity according to the surrounding condition and object detection. The characteristic curves shown in Fig. 8 are derived from the experimental dataset recorded in Table 1, illustrating how ambient lighting conditions and vehicle proximity influence the street light brightness.

Specifically, Fig. 8(a) highlights the relationship between the analog LDR reading and LED output. Under high ambient light conditions (LDR = 950), daylight is deemed sufficient, keeping the street light completely turned off (PWM = 0). Once ambient light levels drop below the threshold value of 600, the system automatically transitions into night mode, setting the LED to an energy-saving standby intensity (PWM = 80). As ambient light decreases further, detecting motion triggers the system to scale the brightness up to its maximum capacity (PWM = 255).

Complementing this, Fig. 8(b) demonstrates how vehicle distance dictates LED illumination. When an approaching object enters the 120 cm threshold or closer, the Arduino immediately boosts the LED brightness to full capacity (PWM = 255) and activates the piezo-buzzer. Conversely, for distances exceeding 120 cm, the streetlight remains in its dimmed state (PWM = 80) with the buzzer disengaged. The red dashed boundary line in Fig. 8(b) explicitly marks this 120 cm detection cutoff.

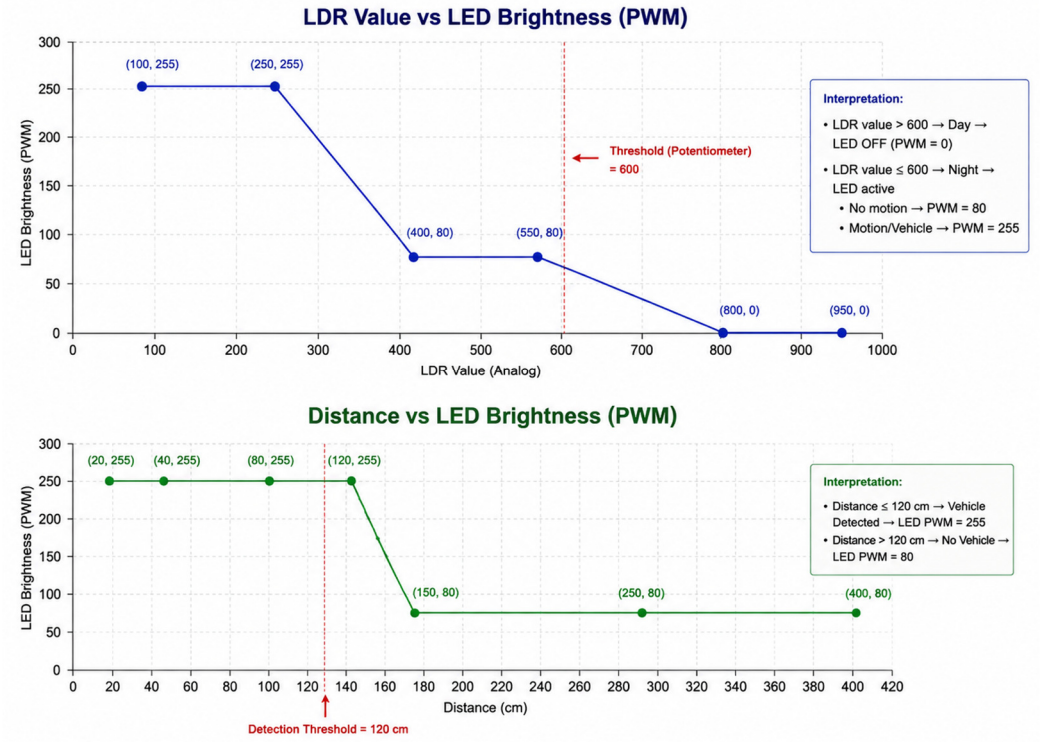


Figure 8: (a) Characteristics curve between variation of LED Brightness with LDR Value and (b) Characteristics curve between variation of LED Brightness with Distance of vehicle.

4. CONCLUSION

This work originated from observing a simple, widespread issue: standard streetlights waste vast amounts of energy by glowing at full power all night, even when roads are empty or daylight is present. To address this inefficiency, we developed an adaptive, smart lighting setup using an LDR for ambient light sensing, along with PIR and ultrasonic sensors for precise motion and proximity detection. Successfully simulated in Tinkercad, the Arduino UNO seamlessly coordinated sensor inputs with the LED, LCD, and buzzer outputs under varying conditions. By dynamically scaling brightness to actual demand, this design significantly cuts electricity waste, lowers operational expenses, and ensures safer, smarter public spaces.

Looking ahead, the system can be enhanced by integrating IoT and wireless protocols for remote monitoring and diagnostics. Incorporating solar panels and rechargeable batteries would enable sustainable off-grid operation. Furthermore, deploying AI-driven algorithms could allow the lights to learn traffic patterns and proactively adjust illumination, while automated fault detection could instantly flag damaged units. Designed for versatility, this solution can easily scale beyond city streets to highways, campuses, parking lots, and parks—paving the way for more resilient and energy-conscious urban infrastructure.

REFERENCES

[1] S. P. Lau, G. V. Merrett, A. S. Weddell, and N. M. White, "A traffic-aware street lighting scheme for Smart Cities using autonomous networked sensors," *Computers & Electrical Engineering*, vol. 45, pp. 192–207, Jul. 2015, doi: 10.1016/j.compeleceng.2015.06.011.

[2] E. Petritoli, F. Leccese, S. Pizzuti, and F. Pieroni, "Smart lighting as basic building block of smart city: An energy performance comparative case study," *Measurement*, vol. 136, pp. 466–477, Mar. 2019, doi: 10.1016/j.measurement.2018.12.095.

[3] M. Kostic and L. Djokic, "Recommendations for energy efficient and visually acceptable street lighting," *Energy*, vol. 34, no. 10, pp. 1565–1572, Oct. 2009, doi: 10.1016/j.energy.2009.06.056.

[4] A. Ożadowicz and J. Grela, "Energy saving in the street lighting control system a new approach based on the EN-15232 standard," *Energy Efficiency*, vol. 10, no. 3, pp. 563–576, Jun. 2017, doi: 10.1007/s12053-016-9476-1.

[5] G. Shahzad, H. Yang, A. W. Ahmad, and C. Lee, "Energy-efficient intelligent street lighting system using traffic-adaptive control," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5397–5405, Jul. 2016, doi: 10.1109/JSEN.2016.2557345.

[6] F. Leccese, "Remote-control system of high efficiency and intelligent street lighting using a ZigBee network of devices and sensors," *IEEE Transactions on Power Delivery*, vol. 28, no. 1, pp. 21–28, Jan. 2013, doi: 10.1109/TPWRD.2012.2212215.

[7] J. Zhang, G. Qiao, G. Song, H. Sun, and J. Ge, "Group decision making based autonomous control system for street lighting," *Measurement*, vol. 46, no. 1, pp. 433–441, Jan. 2013, doi: 10.1016/j.measurement.2012.05.025.

[8] P. Mohandas, J. S. A. Dhanaraj, and X.-Z. Gao, "Artificial neural network based smart and energy efficient street lighting system: A case study for residential area in Hosur," *Sustainable Cities and Society*, vol. 48, p. 101499, Jul. 2019, doi: 10.1016/j.scs.2019.101499.

[9] P. Elejoste et al., "An easy to deploy street light control system based on wireless communication and LED technology," *Sensors*, vol. 13, no. 5, pp. 6492–6523, May 2013, doi: 10.3390/s130506492.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 26-31 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

ScriptGenie AI: A Groq API and LLaMA 3-Based Full-Stack Framework for Automated YouTube Script Generation

[1]Kumar Harsh, [2]MD Tayyeb Hussain, [3]Satyam Kumar, [4]Jayanta Kumar Bag, [5]Tilak Mukherjee

[1234]Dept. of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India
1UG Student, Department of ECE, Haldia Institute of Technology, Haldia, West Bengal
2Assistant Professor, Department of ECE, Haldia Institute of Technology, West Bengal, India

ABSTRACT

Scripting remains one of the most time-consuming stages involving YouTube video production, requiring creators to structure a hook, introduction, main content, and call-to-action while also drafting titles, tags, and descriptions for discoverability. ScriptGenie AI is a full-stack web application that automates this process by converting a single user-supplied topic into a complete, structured video script. The system integrates a vanilla HTML/CSS/JavaScript front end with a Node.js and Express.js back end, which communicates with the Groq inference API so as to query the open-weight LLaMA 3 language model through a structured, JSON-constrained prompt. The present paper details the system architecture, the rationale for selecting an inference-API-based design over local model hosting, the request-response workflow, and the prompt-engineering strategy used to obtain reliably structured output.

KEYWORDS: YouTube Script Generation, Large Language Models, Groq API, LLaMA 3, Prompt Engineering; MERN-based Web Architecture, MongoDB Atlas.

1. INTRODUCTION

The growth of YouTube as a primary medium for information and entertainment has made consistent content output a central challenge for all class of creators. Producing a video typically begins with scripting: framing an attention-grabbing hook, organizing the main content into coherent sections, and closing with a call-to-action, in addition to preparing supporting metadata such as titles, tags, and descriptions for search visibility. Manually performing these sub-tasks for every video is repetitive and time-intensive task, particularly for independent creators managing the entire production pipeline alone.

However, recent advances in large language models (LLMs) have made it feasible to automate substantial portions of this writing workload through natural-language prompting rather than task-specific model training. Cloud-hosted inference APIs now allow small teams to access such models without owning the underlying infrastructure, shifting the engineering effort from model development to system integration, prompt design, and workflow orchestration.

This paper presents ScriptGenie AI, a full-stack tool that accepts a topic, a desired tone, a target length, and an audience type, and returns a structured script object comprising a hook, introduction, main content sections, a call-to-action, title suggestions, tags, and a description.

The system does not train or fine-tune any language model; instead, it is built entirely around the Groq inference API serving the openly released LLaMA 3 model. The contribution of this work is therefore an applied system design and integration study. It documents a kind of architecture, an API-selection rationale, and a prompt-engineering approach for structured content generation, rather than a new model or a benchmarked algorithm.

2. RELATED WORK

*Corresponding Address: mukherjeetilak@gmail.com DOI 10.5281/zenodo.22037535

The use of generative AI in video content creation has catapulted growing research attention and growth. Analysis of a large corpus of YouTube how-to videos and related search reported that creators already use general-purpose LLMs at multiple stages of production, including topic ideation, script drafting, and title generation [1, 2]. Broader surveys of content creators similarly report that generative tools are being adopted for writing and editing tasks, while also raising concerns around originality and disclosure that motivate careful, transparent tool design [2]. Distinct from tools that summarize or repurpose existing long-form media, ScriptGenie AI addresses the earlier stage of the pipeline: producing an original script structure from a bare topic before any recording takes place [1,2].

It needs to be mentioned here that the system relies entirely on prompting a pretrained model rather than on fine-tuning, and its design is informed by the broader literature on prompt engineering. A wide range of prompting techniques for eliciting structured, task-specific behaviour from LLMs without parameter updates have been reported [3], which motivates the JSON-schema-constrained prompting strategy adopted in this work. On the inference side, the system depends on Groq's Language Processing Unit (LPU), a deterministic, statically scheduled accelerator architecture purpose-built for low-latency LLM inference, as distinct from general-purpose GPU pipelines [4, 5]. Finally, the underlying language model, LLaMA 3, is integral part of an openly released model family whose architecture and training methodology are documented by Meta AI [6]. The backend integration pattern used here, an Express.js REST layer over a MongoDB document store, follows established practices for JSON-native web-service architecture systems [7].

3. SYSTEM ARCHITECTURE

ScriptGenie AI follows a three-tier architecture consisting of a presentation layer, layer, and persistence layer application, which are connected through a REST interface.

3.1 Presentation Layer

The client is implemented in plain HTML5, CSS3, and vanilla JavaScript, avoiding a front-end framework dependency. It captures the user's topic, tone (professional, funny, storytelling, motivational, educational, or casual), desired length (shorts, standard, or long form), and target audience; and sends this payload to the backend; and thereby renders the returned script into labelled sections (hook, intro, main content, call-to-action), a metadata tab (tags and description), and a titles tab. The interface also exposes copy-to-clipboard and .txt download actions.

3.2 Application Layer

The backend is built on Node.js with the Express.js framework and is organized using a routes-controllers-services pattern. The route scriptRoutes.js exposes POST /api/scripts/generate, which is handled by scriptController.js. This controller essentially validates the incoming topic length and confirms that the Groq API key is configured before delegating generation to a dedicated service module, openaiService.js, which constructs the model prompt and issues the request to the Groq API. A parallel route, savedRoutes.js, exposes CRUD endpoints for the script library, handled by savedController.js.

3.3 Persistence Layer

Saved scripts are stored in MongoDB Atlas, a cloud-hosted, fully managed instance of MongoDB, and accessed through the Mongoose object-modeling library. A Script schema (Script.js) defines the structure of a persisted script document, enabling later retrieval and deletion through the /api/saved endpoints without requiring a locally administered database server.

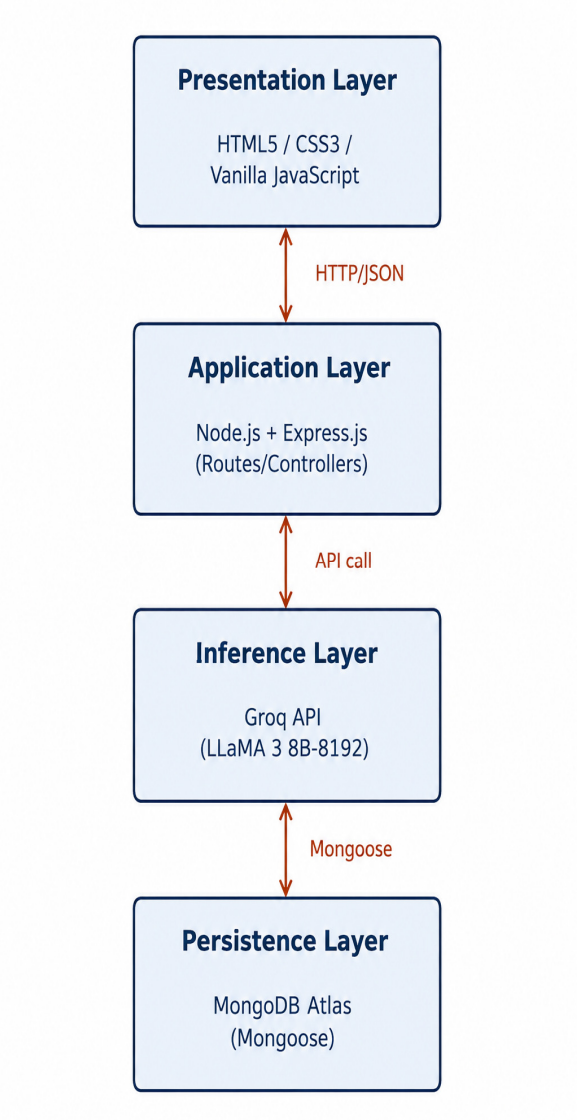


Figure 1: Three-tier system architecture of ScriptGenie AI, showing the presentation, application, inference, and persistence layers and their REST/API communication paths

4. METHODOLOGY AND API INTEGRATION

4.1 Rationale for API-Based Inference

ScriptGenie AI deliberately does not host or fine-tune a language model. Locally hosting an LLM of comparable capability would require dedicated GPU infrastructure and ongoing maintenance, which is impractical for a small student-built application. The Groq API was selected instead of a locally hosted alternative for three reasons drawn directly from the project's requirements:

i. It offers a free tier sufficient for demonstration and iterative development.

ii. It serves the openly released LLaMA 3 (llama3-8b-8192) model, avoiding dependence on a closed-source provider.

iii. It is built on Groq's LPU accelerator, whose statically schedule and deterministic pipeline architecture is reported to substantially reduce per-token inference latency relative to general-purpose GPU serving [4, 5], which is important for an interactive, request-response user experience.

4.2 Request-Response Flow

A generation request follows a linear pipeline: the client validates and submits the topic, tone, length, and audience fields; the Express route forwards the request to scriptController.js, which re-validates the payload server-side; openaiService.js assembles a structured prompt embedding these fields and instructs the model to respond strictly in a predefined JSON schema (hook, intro, mainContent, cta, title, tags, description); this prompt is sent to the Groq /chat/completions-equivalent endpoint targeting llama3-8b-8192 with a temperature of 0.8 and a maximum token budget of 2000; the returned JSON is parsed and returned to the client, which renders it into the appropriate UI sections.

Here the user chooses to save a script, a second, independent request persists the same JSON structure into the Script collection in MongoDB Atlas.

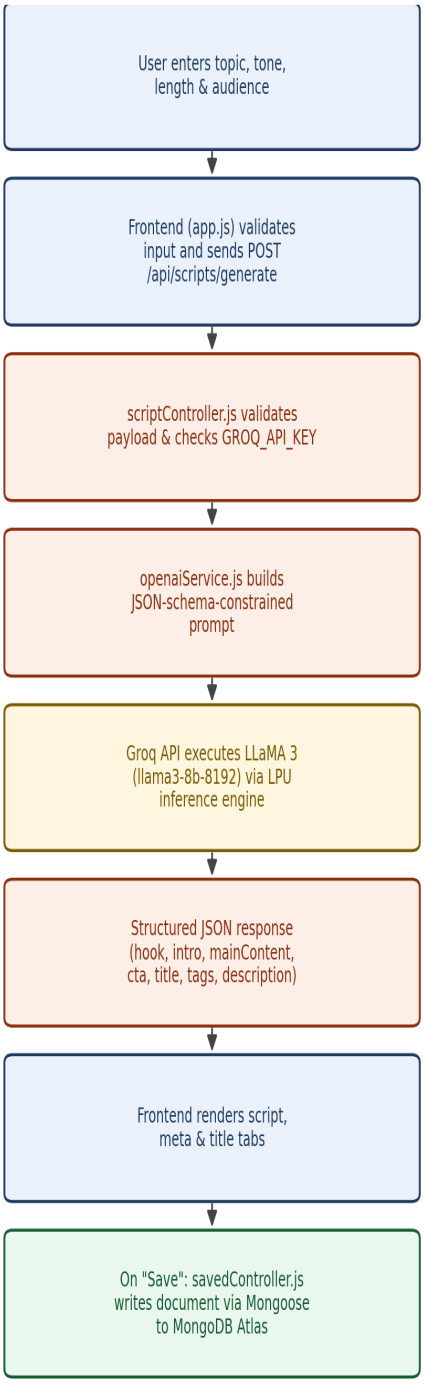


Figure 2: Request-response workflow of a script-generation call, from user input through Groq/LLaMA 3 inference to optional persistence in MongoDB Atlas.

4.3 Prompt Design

Because the system has no control over model weights, output reliability depends entirely on prompt construction. The prompt explicitly role-primes the model as a professional YouTube scriptwriter, supplies the topic, tone, a length-specific structural guide, and audience descriptor, and constrains the response format to a fixed JSON object.

This schema-constrained prompting approach follows established practice for eliciting structured, pursuable output from general-purpose LLMs without any fine-tuning step [3], and allows the front end to deterministically map model output to fixed UI fields.

5. RESULTS AND DISCUSSIONS

As ScriptGenie AI does not train a model or operate on a labelled dataset, no accuracy, precision, or benchmark metrics are applicable or reported. The system was instead evaluated functionally: the generation endpoint was exercised across the six supported tones and three supported length options, and in each case returned a syntactically valid JSON object containing all expected fields, which the front end successfully rendered without manual reformatting. The save, retrieve, and delete operations against the MongoDB Atlas-backed library were similarly verified to behave as CRUD operations should, that is, saved entries reappear correctly on retrieval and are removed on deletion.

Consistent with Groq's documented LPU-based serving characteristics [4, 5], generation requests were observed to return within a short, interactive time window, in line with the project's own documented expectation of roughly one to two seconds end-to-end. This should be read as a qualitative operational characteristic of the deployed system rather than a controlled, statistically validated latency benchmark, since no repeated-trial timing study was conducted. Likewise, the Groq free tier's daily request allowance was found sufficient for iterative development and demonstration use but was not stress-tested under sustained concurrent load, which is noted as a major limitation below.

6. LIMITATIONS AND FUTURE SCOPE

The system's output quality is bounded by the capability of the llama3-8b-8192 model and by prompt design rather than by any mechanism the system itself learns or adapts. Therefore, it inherits general LLM limitations such as occasional factual inaccuracy in generated content, which is not independently fact-checked before being returned to the user. Reliance on a third-party inference API also introduces an external dependency in terms of service availability, rate limits, and pricing policy changes on Groq's side that directly affect system availability. Finally, the current implementation has not been evaluated under multi-user concurrent load or adversarial input conditions.

Several extensions have been planned as future scope of work. Voice-over generation would convert a produced script directly into narrated audio, closing the gap between scripting and recording. Multi-language support would allow script generation in languages beyond English, broadening the tool's creator base. An AI-assisted thumbnail generator would extend the pipeline into the visual domain. A video-SEO optimizer could automate title and tag refinement using performance feedback, while an analytics dashboard could track how generated scripts correlate with published video performance. Direct export integrations with social platforms would additionally streamline distribution of generated titles and descriptions beyond YouTube.

7. CONCLUSION

This paper presented ScriptGenie AI, a full-stack web application that automates YouTube script generation by integrating a vanilla JavaScript front end, an Express.js/Node.js backend, the Groq inference API serving LLaMA 3, and a MongoDB Atlas persistence layer. Rather than proposing a new model or algorithm, the contribution of this work particularly lies in its system architecture, its rationale for choosing an API-based inference strategy over local model hosting, and its schema-constrained prompt design for reliable structured generation.

Functional testing confirmed that the generation, storage, and retrieval pipeline operates as designed, though the evaluation remains qualitative in the absence of a training dataset or benchmarked task. Furthermore, future work shall extend the system toward multilingual, multimedia, and analytics-driven capabilities as outlined above.

REFERENCES

[1] Niu, S., et al. (2025). Making AI-Enhanced Videos: Analyzing Generative AI Use Cases in YouTube Content Creation. Extended Abstracts of the CHI Conference on Human Factors in Computing Systems. arXiv:2503.03134.

[2] Content Creation with Generative AI: How Do Content Creators Responsibly Use Generative AI Tools? (2025). Proceedings of the ACM on Human-Computer Interaction. doi:10.1145/3788080.

[3] Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. arXiv:2402.07927.

[4] Groq Inc. (2024). What is a Language Processing Unit? GroqCloud Technical Documentation. Available: https://groq.com/blog/the-groq-lpu-explained.

[5] AI Accelerators for Large Language Model Inference: Architecture Analysis and Scaling Strategies. (2025). arXiv:2506.00008.

[6] Meta AI, Llama Team. (2024). The Llama 3 Herd of Models. arXiv:2407.21783.

[7] Holmes, & Harber, A. Getting MEAN with Mongo, Express, Angular, and Node.js, 2nd Edition. Manning Publications. (2019).

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 32-37 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

DESIGN, SIMULATION AND FABRICATION OF DUAL-BAND MICROSTRIP PATCH ANTENNA

[1]Dipan Das, [2]Surajit Mukherjee

[1]UG Student, Department of ECE, Haldia Institute of Technology, Haldia, West Bengal
[2]Assistant Professor, Department of ECE, Haldia Institute of Technology, West Bengal, India

ABSTRACT

This paper presents the design, simulation, and prototype realization of a compact dual-band microstrip patch antenna intended for wireless applications. A dual-band dual-mode structure that resonates at 2.7 GHz and 5.3 GHz is obtained by inserting two parallel I-shaped slots into the radiating patch. The slots create the second resonance while preserving a simple, low-profile geometry. The antenna is first dimensioned with the standard microstrip design equations and then refined through parametric sweeps in HFSS/CST on a 1.2 mm thick Arlon substrate (ε_r = 2.2). The optimized design yields return losses of –21.38 dB at 2.7 GHz and –23.04 dB at 5.3 GHz, together with broadside gains of approximately 7.33 dB and 7.38 dB, respectively. Radiation-pattern analysis shows predominantly broadside radiation at the lower frequency and additional lobes at 5.3 GHz arising from higher-order modes. A physical prototype is fabricated by PCB etching, fitted with an SMA connector, and measured. The measured results confirm the simulated performance, demonstrating that the antenna is well suited to compact dual-band use in WLAN, LTE and related wireless systems.

KEYWORDS: Dual-Band Antenna; Microstrip Patch Antenna; I-Shaped Slots; HFSS/CST; WLAN and LTE.

1. INTRODUCTION

The rapid expansion of wireless communication systems continues to drive demand for compact, lightweight, low-cost and efficient antennas capable of dual-band operation. Microstrip patch antennas remain attractive because of their planar geometry, low profile, straightforward fabrication and ease of integration with microwave circuits. Conventional single-patch designs, however, typically offer only a single resonance and limited bandwidth, gain and efficiency. Techniques such as slot loading, stacked patches, parasitic elements and defected ground structures have been widely explored to overcome these limitations. Among them, slot loading is particularly effective for dual-band operation: it perturbs the surface-current distribution and lengthens the effective current path without a significant increase in overall size.

In the present work a compact dual-band microstrip patch antenna for wireless applications is designed, simulated and fabricated. The design begins with the classical microstrip equations, after which two parallel I-shaped slots are introduced, and the geometry is optimized by full-wave parametric simulation. Reflection coefficient, VSWR, bandwidth, gain, efficiency and radiation patterns are evaluated. A prototype is then fabricated by standard PCB techniques and measured with a network analyzer. Agreement between measured and simulated data validates the approach.

2. ANTENNA DESIGN AND GEOMETRY:

The antenna is designed to operate at 2.7 GHz and 5.3 GHz on an Arlon substrate of relative permittivity ε_r = 2.2 and thickness 1.2 mm. The design flow comprises theoretical calculation of the initial patch dimensions, electromagnetic modelling, parametric optimization, fabrication and experimental verification. Standard rectangular-patch equations are first applied at the lower design frequency of 2.7 GHz. Two parallel I-shaped slots are subsequently etched on the

*Corresponding Address: ece.surajit@yahoo.com DOI 10.5281/zenodo.22038093

radiating patch; these slots alter the surface-current paths and introduce an additional resonance at 5.3 GHz while keeping the structure compact and simple.

The complete antenna comprises a rectangular radiating patch, two parallel I-shaped slots, the dielectric substrate, a ground plane and a microstrip feed line. Initial dimensions are refined by repeated full-wave simulations. The principal design equations are given below.

The Patch Width (w_p) is given by:

$$w_p = \frac{c}{2f_0}\sqrt{\frac{2}{\varepsilon_r + 1}}$$

The Effective Dielectric Constant (ε_{eff}) considering fringing effects is:

$$\varepsilon_{eff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left(1 + \frac{12h}{w_p}\right)^{-1/2}$$

The Effective Patch Length is calculated as:

$$L_{eff} = \frac{c}{2f_0\sqrt{\varepsilon_{eff}}}$$

The Fringing Length Extension is:

$$\Delta L = 0.412h\frac{(\varepsilon_{eff}+0.3)(\frac{w_p}{h}+0.264)}{(\varepsilon_{eff}+0.258)(\frac{w_p}{h}+0.8)}$$

Finally, the Actual Patch Length is:

$$l_p = L_{eff} - 2\Delta L$$

Table 1: Final Optimised Antenna Dimensions

Parameter	**Dimension**
patch width (w_p)	39.501 mm
patch length (l_p)	33.009 mm
substrate width(w_s)	89.501 mm
substrate length (l_s)	83.36 mm
feed position (y_0)	12.881 mm
notch width	4.971 mm
feed-line width	2.42 mm
slot width	3 mm
slot length	17 mm
substrate height	1.2 mm

Arlon is chosen for its ready availability, moderate cost and suitability for laboratory fabrication; the 1.2 mm thickness provides a practical compromise between compactness and radiation efficiency. Copper is used for both the patch and the ground plane. Microstrip-line feeding is selected for its planar simplicity and compatibility with printed-circuit processes. The feed position is adjusted until S_{11} remains below –10 dB at both resonances. During optimization, patch size, slot geometry, feed location, substrate outline and ground-plane dimensions are varied systematically to improve return loss, impedance match, bandwidth, gain and pattern stability. The final geometry is realized as a three-dimensional electromagnetic model that incorporates accurate material parameters, finite conductor thickness, precise slot placement and a realistic feed transition before fabrication proceeds.

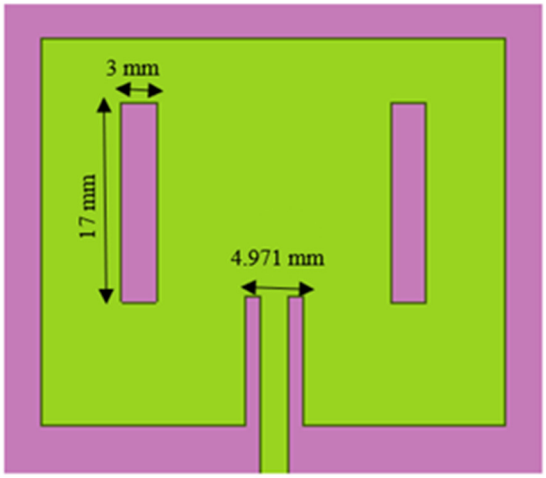


Figure 1: Slot and notch position on the patch

Figure 1 shows the slot and notch locations on the radiating patch.

The structure is enclosed in a radiation box with approximately quarter-wavelength clearance on all sides. Adaptive meshing is concentrated along the

slot edges, the feed gap and the patch perimeter. Two frequency sweeps are employed: a discrete sweep for gain and radiation patterns at the two design frequencies, and an interpolating sweep from 2 GHz to 8 GHz to obtain the broadband return-loss response.

3. WORKING PRINCIPLE:

The generation of higher and lower resonant frequencies in a slotted patch antenna mainly depends on the current path length and the resonance characteristics of the patch.

3.1 Lower-Frequency Generation:

The slots produce a discontinuity in the direct path of the surface current. Consequently, the current is forced to flow around the slot, which increases the effective electrical length of the patch. Since the longer current path produces a lower resonant frequency. Thus, the I-shaped slots help generate the lower-frequency resonance. Figure 2 presents the top view of the dual-band antenna, illustrating the current paths associated with the lower resonance.

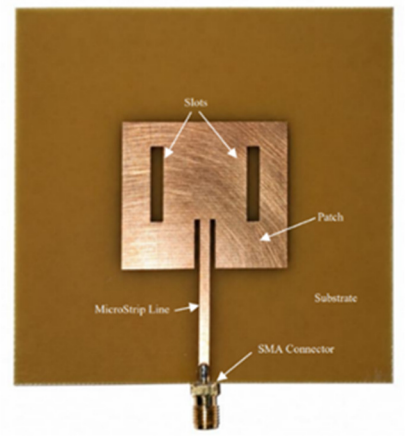


Figure 2: Top view of the "Dual-Band" microstrip patch antenna

3.2 Higher-Frequency Generation:

The portion of the patch that lies outside the slots supports a shorter current path and therefore resonates at the higher frequency of 5.3 GHz. Edge modifications and the slot geometry also excite higher-order modes that contribute to the upper band. Figure 3 shows the corresponding back view of the antenna.

Figure 3: Back view of the "Dual-Band" microstrip patch antenna

3.3 Dual-Band Operation:

The two current paths therefore establish the two operating frequencies: the longer path around the slots yields 2.7 GHz while the shorter residual path yields 5.3 GHz. Proper choice of slot dimensions and feed location enables simultaneous good matching at both frequencies.

4. FABRICATION:

Once simulation results are judged satisfactory, the layout is transferred to a 1.2 mm Arlon substrate with copper cladding on both sides. The pattern is defined by toner-transfer etching, producing the dual-slot patch and microstrip feed. An SMA connector is soldered to the feed line to complete the prototype. Figure 4 shows the isometric view of the simulated antenna.

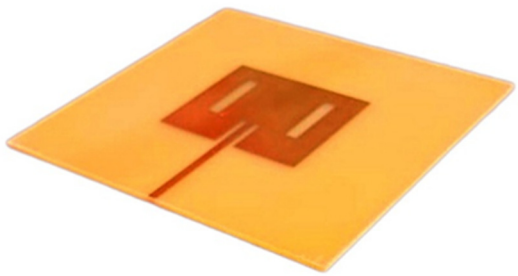


Figure 4: Isometric view of the "Dual-Band" microstrip patch antenna

The pattern is etched using toner transfer methodology following application on Arlon to obtain the required microstrip line with dual slots and fed with an SMA connector in the input feed line, as the desired prototype of the antenna is fabricated in its ready form as displayed in Figure 2 and 3. Photos are given for the Top view of the fabricated antenna, including the dual slots and patch and their feeding line, and for its bottom view displaying a solid ground plane. The slots and feeding line align within the specified limits from their simulated values.

5. RESULT:

The proposed dual-band microstrip patch antenna was designed and analysed using full-wave electromagnetic simulation software (HFSS/CST) for operation at 2.7 GHz and 5.3 GHz. Two parallel rectangular slots were incorporated into the radiating patch to generate dual-band characteristics and increase the effective current path length.

5.1 Return Loss (S_{11}):

The simulated results show two distinct resonances with good impedance matching. At 2.7 GHz, the return loss is approximately −21.38 dB, while at 5.3 GHz, it is approximately −23.04 dB. Both values are below the −10 dB reference level, confirming satisfactory impedance matching and dual-band operation. Figure 5 displays the simulated return-loss curve over 2–8 GHz.

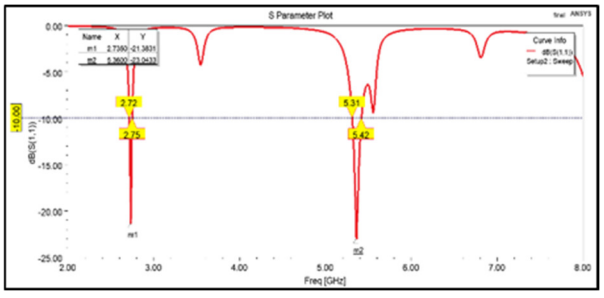


Figure 5: Return Loss (S_{11}) Plot of the Antenna

5.2 Peak Realized Gain:

The antenna provides stable gain at both resonant frequencies. The peak realized gain is approximately 7.33 dB at 2.7 GHz and 7.38 dB at 5.3 GHz, indicating satisfactory radiation performance and effective power transfer. Figure 6 shows the peak-gain response versus frequency.

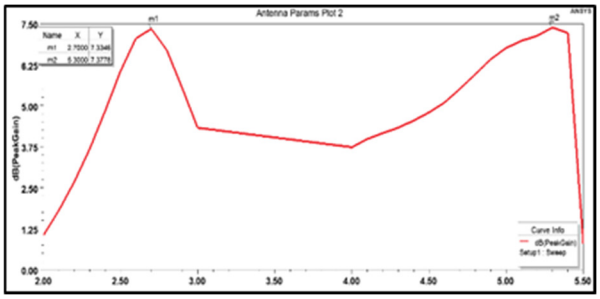


Figure 6: Peak Gain Plot of the Patch Antenna

5.3 Radiation Pattern at 2.7 GHz:

The φ = 0° and φ = 90° planes exhibit broadside radiation with stable gain and effective current concentration around the patch edges and slot regions, confirming efficient resonance at 2.7 GHz. Figures 7 and 8 present the E-plane and H-plane patterns at 2.7 GHz, respectively.

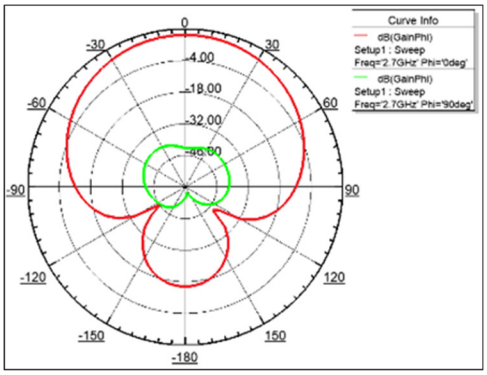


Figure 7: Radiation Pattern at 2.7 GHz (E-Plane)

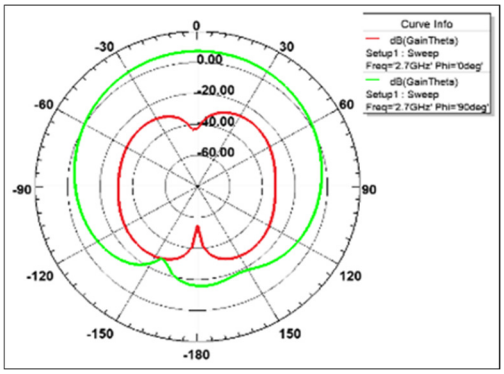


Figure 8: Radiation Pattern at 2.7 GHz (H-Plane)

5.4 Radiation Pattern at 5.3 GHz:

At the higher frequency, additional lobes are observed due to higher-order resonant modes associated with the slot structure. Slight asymmetry in the φ = 0° and φ = 90° planes, along with more noticeable back lobes and nulls, is observed due to surface-current redistribution and scattering effects. Figures 9 and 10 show the corresponding E-plane and H-plane patterns at 5.3 GHz.

5.5 Overall Performance:

The proposed antenna successfully achieves dual-frequency operation at 2.7 GHz and 5.3 GHz with low return loss, stable gain, and acceptable radiation characteristics. The parallel slot configuration provides dual resonance while maintaining a compact, low-profile, simple, and cost-effective structure.

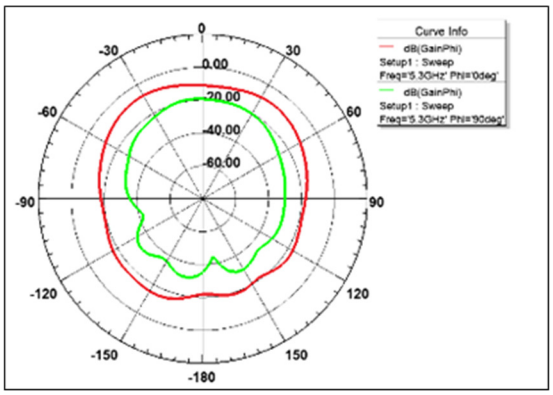


Figure 9: Radiation Pattern at 5.3 GHz (E-Plane)

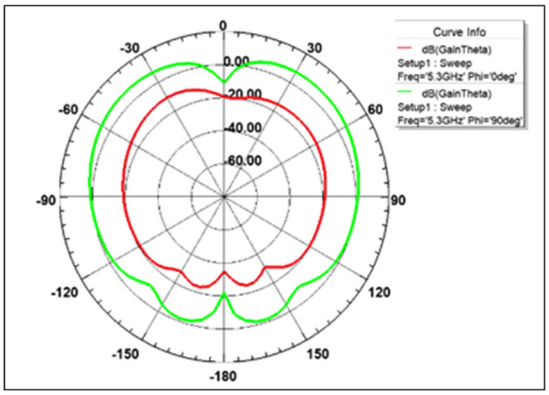


Figure 10: Radiation Pattern at 5.3 GHz (H-Plane)

6. CONCLUSION

A compact dual-band microstrip patch antenna array containing two parallel rectangular slot-etched elements for operation at 2.7 GHz and 5.3 GHz was designed, simulated and developed. Design specifications obtained, such as the return loss being below 19 dB, the gain being approximately 7.3 dB, and broadside radiation being observed mostly, are in accordance with the obtained values from manufactured results to accommodate a specific range of applications like WLAN and LTE.

REFERENCES

[1] C. A. Balanis, (2016). Antenna Theory: Analysis and Design. 4th ed., John Wiley & Sons.

[2] R. Garg, P. Bhartia, I. Bahl and A. Ittipiboon, (2001). Microstrip Antenna Design Handbook. Artech House, 2001.

[3] D. M. Pozar, (1992). Microstrip Antennas. *Proceedings of the IEEE*, 80(1), 79–91.

[4] Y. X. Guo, K. M. Luk and K. F. Lee, (2002). Dual-Band Slot-Loaded Microstrip Patch Antenna. *IEEE Transactions on Antennas and Propagation*, 50(1), 65–67.

[5] K. L. Wong, (2002). Compact and Broadband Microstrip Antennas. John Wiley & Sons.

[6] N. Behdad and K. Sarabandi, (2010). A Multiresonant Single-Element Wideband Slot Antenna. *IEEE Transactions on Antennas and Propagation*, 58(5), 1740–1743.

[7] J. Anguera, C. Puente, C. Borja and J. Soler, (2003). Dual-Frequency Broadband Stacked Microstrip Antenna. *IEEE Antennas and Propagation Magazine*, 45(2), 10–22.

[8] S. Maci and G. B. Gentili, (1997). Dual-Frequency Patch Antennas. *IEEE Antennas and Propagation Magazine*, 39(6), 13–20.

[9] W. C. Liu, C. M. Wu and N. C. Chu, (2010). A Compact CPW-Fed Slotted Patch Antenna for Dual-Band Operation. *IEEE Antennas and Wireless Propagation Letters*, 9, 110–113.

[10] K. F. Tong and T. P. Wong, (2007). Circularly Polarized U-Slot Rectangular Patch Antenna. *IEEE Transactions on Antennas and Propagation*, 55(8), 2382–2385.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 38-48 ISSN: 0973-6875
Available Online at www.hithaldia.in/locate/ECCN

AI-BASED HELMET DETECTION PROJECT USING YOLOv11 AND STREAMLIT

[1]Sams Tabrez, [2]Keshav Kumar Mishra, [3]Sachin Verma, [4]Sachin Kumar, [5]Komal Kusum, [6]Rishav Raj, [7]Himanshu Ranjan Das

[1-6]UG Student, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[7]Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[1]tabrezmtp@gmail.com, [2]Keshavkumarmishra35@gmail.com, [3]sachinverma78760@gmail.com, [4]sachin141ece@gmail.com, [5]komalkusum18062005@gmail.com, [6]rishavraj30420@gmail.com, [7]das.himanshu69@gmail.com

ABSTRACT

In this paper we studied on how Wearing a safety helmet is one of the simplest ways to protect a worker's head at a construction site or factory, but checking this that all worker is wearing helmet by hand is very slow, tedious, and often uncertain. This paper presents an automatic helmet detection system built on the YOLOv11 deep learning model, trained on a labelled image dataset collected from the Roboflow platform. On unseen images, the model is showing precision of 77.7%, a recall of 90.1%, and a mean Average Precision of 91.3%, And for processing a single image it takes almost 11.8 milliseconds. The trained model was connected with web dashboard Streamlit, so that a safety officer with no technical background can upload a photo and instantly see whether the site is safe or not. The results confirm that how an AI model is used in human safety purpose.

KEYWORDS: Helmet Detection, YOLOv11, Deep Learning, Streamlit Dashboard

1. INTRODUCTION

Construction sites, factories, and mines are the places where workers face problem like falling hard object from height, heavy machinery part etc. A safety helmet is one of the simplest and most effective ways to protect a worker's head, that is why safety rules is compulsory in all industries. In practice, however, checking whether every worker is wearing a helmet is usually done by a human guard walking around the site or watching a wall of CCTV screens. This kind of manual checking are boring and exhausted, a man can never watched each corner of site at a time and often misses violations of rules, especially when it comes on large and busy sites **[1]**.

Nowadays using artificial intelligence that lets a computer "see" and understand the pictures of construction sites, earlier needed a programmer to manually write rules for colour and shape to detect an object, and these rules broke down easily whenever the lighting or camera angle changed. So, using Deep learning models, especially Convolutional Neural Networks (CNNs), instantly learn these visual patterns directly from thousands of example images **[2]**, which makes them far easier for a busy, unpredictable construction site.

The goal of this project is to build a complete, working system that looks at a photo from a work site and decides whether the person in it is wearing a helmet or not for worker safety.

2. LITERATURE REVIEW

Before the idea of deep learning, helmet detection was done by using simple image processing tricks. In this System first located a person's head and then checked the colour of that region using techniques such as a Histogram of Oriented Gradients (HOG) with a Support Vector Machine (SVM), or fitted circular shapes using the Hough Transform **[1]**. Sometimes this method facing problem due to

camera angle, lightening effect, shadows, and other brightly coloured objects **[3-20].**

A major problem came from large number of image's classification competitions such as ImageNet **[3]**, deep CNNs like AlexNet **[4]**. For worker safety researchers work on the field of deep learning. Early deep detectors, known as two-stage detectors, first proposed a set of candidate regions and then classified each one separately; the Faster R-CNN model **[5]** improved this approach by sharing computed features between the two steps. Two-stage detectors are good, but their live video analyses is usually slow. To increase the speed of analysing, the idea of single-stage detectors came, which predict an object's location and class in a single pass over the image. The You Only Look Once (YOLO) family, first introduced by Redmon et al. **[6]** and later improved through versions such as YOLO9000 **[7]**, became very popular because it is both fast and more accurately.

3. METHODOLOGY AND SYSTEM DESIGN

3.1 System Architecture

The complete system has two main parts: a backend that runs the trained YOLOv11 model, and a Streamlit dashboard that makes the interface between user and screen. When a photo is uploaded, firstly it stored in a memory and then passed through the trained network in a single forward pass, and the network give returns back a list of bounding boxes with class scores. Overlapping boxes are then removed using Non-Maximum Suppression (NMS), and the surviving boxes are drawn back onto the original photo before it is shown to the user, as summarised in Figure 1.

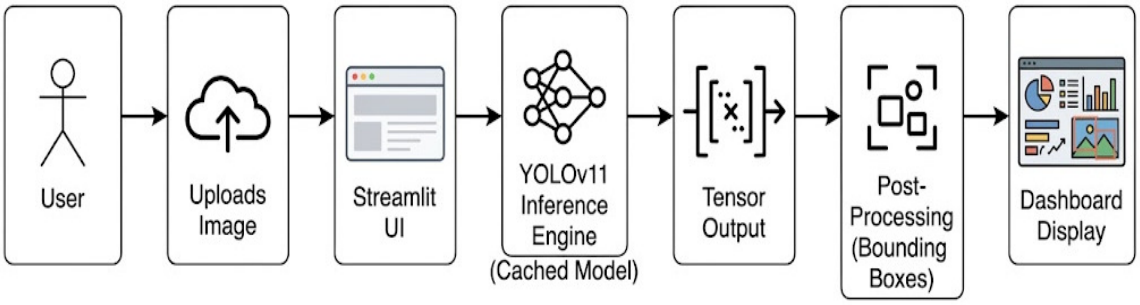


Figure 1: System Architecture Block Diagram.

Table 1. Dataset splitting strategy.

Partition	Number of Images	Percentage of Dataset
Training Set	760	86.36%
Validation Set	120	13.64%
Test Set	0	0.00%
Total	880	100%

3.2 Data Description and Source

A good dataset of images is the foundation of any deep learning project, so the images which is used here were collected from the Roboflow platform under the project workspace helmet__nohelmet. The dataset has two classes: Class 0 for a person who is not wearing helmet, and Class 1 for a person who is wearing a helmet, taken from real construction sites and factory area. Out of 880

labelled images, 760 were used to train the model and 120 were kept aside for validation, as listed in Table 1. Analysing the data of validation set and makes sure the reported accuracy reflects that how the model performs on images it has never seen before.

3.3 Image Pre-processing and Multi-Scale Augmentation

The real camera images differ a lot in lighting and angle, every image was set to a fixed 640×640 pixels before being given to the network. Contrast Limited Adaptive Histogram Equalization (CLAHE) and random brightness changes were applied so that the model learns how to handle both excess sunlight and dark areas. A mosaic augmentation step was also used, which stitches four training images into one single frame, forcing the model to notice presizely, distant helmets in addition to large, close-up ones.

3.4 System Data Flow and Logic Modelling

Before writing any code, two simple Data Flow Diagrams (DFDs) were used to plan the logic of the system. The Level 0 diagram in Figure 2 treats the whole system as a single process and shows that a Site Safety Supervisor uploads an image and receives an annotated image with a safety status in return. The Level 1 diagram in Figure 3 breaks this single process into smaller steps: an upload module, the YOLOv11 inference engine, analytics and rendering module, and an export module, which made it much easier for the team to divide the work between members.

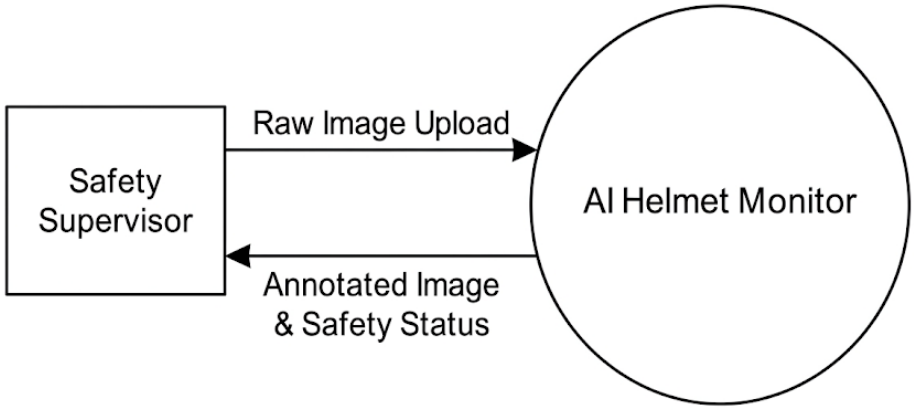


Figure 2: Context-Level Data Flow Diagram (DFD Level 0).

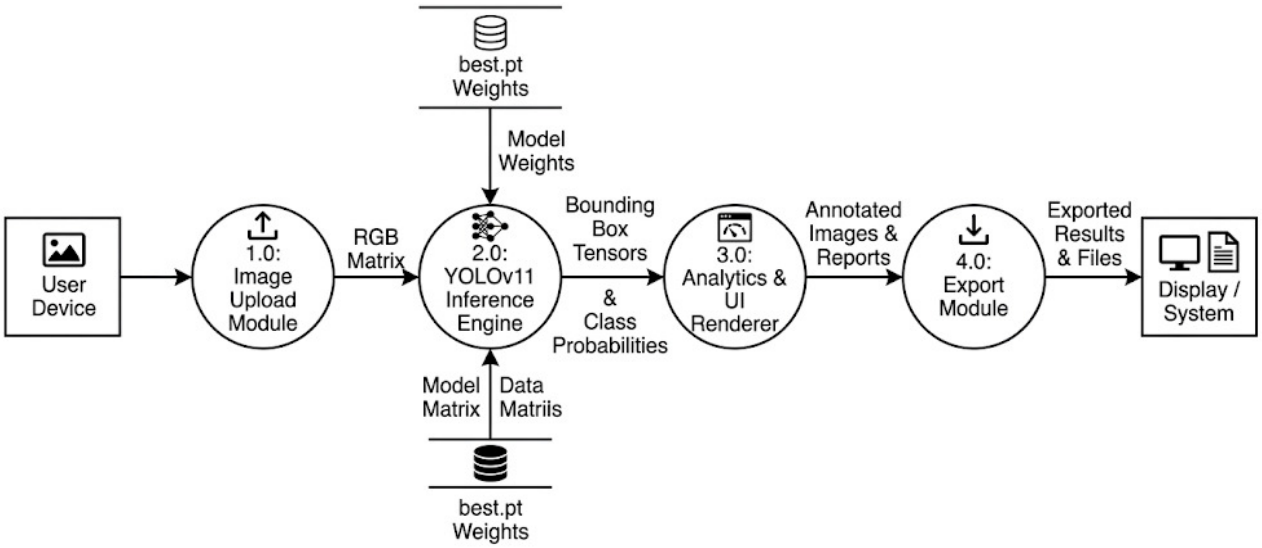


Figure 3: First-Level Data Flow Diagram (DFD Level 1)

3.5 Underlying Algorithm: YOLOv11 Architecture

The detection model of this project is based on YOLOv11, the newest member of the YOLO family. YOLOv11 treats detection as a single, direct regression problem, which is exactly what allows it to run in real time. Its backbone, shown in Figure 4, uses lightweight C3k2 blocks that keep the amount of computation low while still extracting strong features for small and medium-sized objects such as a helmet. The model learns to draw a tight, accurate box around each detected object by minimising a Complete Intersection over Union (CIoU) loss together with a Distribution Focal Loss (DFL); together, these make the predicted box edges sharper, even when a helmet is partly hidden behind scaffolding.

3.6 Tools and Technology Stack

The project was built entirely with free, open-source tools, which are listed in Table 2. Python was the programming language used throughout, the model itself was built and trained with PyTorch through the Ultralytics YOLO package **[19]**, OpenCV handled routine image operations, and Streamlit was used to build the web dashboard. Training was carried out on an NVIDIA Tesla T4 GPU on a cloud notebook, which let the full 60-epoch training run finish in about fifteen minutes. Streamlit's caching feature was especially useful, since it keeps the large trained model loaded in memory instead of reloading it every time a new photo is uploaded.

Table 2: Tools and Technology Stack

Component	Technology Used
Programming Language	Python 3.12
Deep Learning Framework	PyTorch & Ultralytics
Computer Vision Library	OpenCV (cv2)
Frontend UI Framework	Streamlit
Matrix Processing	NumPy & PIL
Hardware Accelerator	NVIDIA Tesla T4 GPU
Code Editor / IDE	Jupyter Notebook & VS Code

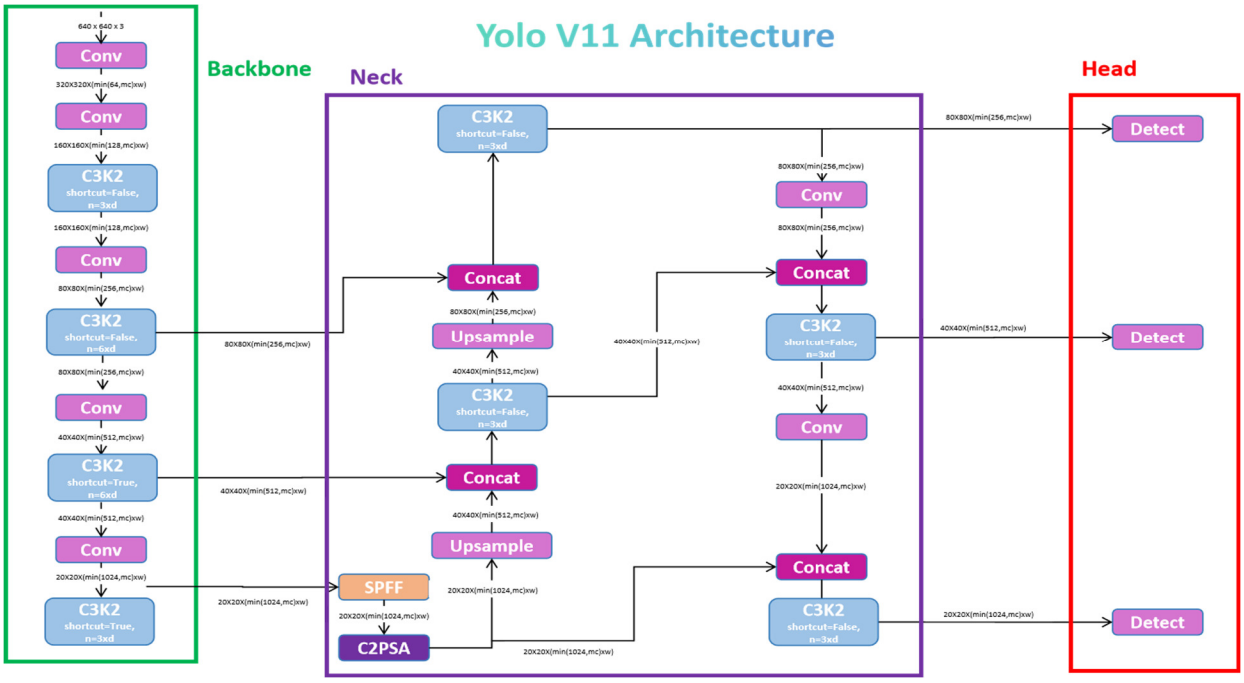


Figure 4: YOLOv11 Backbone, Neck and Head Architecture.

3.7 YOLOv11 Architecture

The YOLOv11 architecture is divided into three main parts: **Backbone, Neck, and Head**. The Backbone extracts important features from the input image using convolutional layers and special blocks that capture patterns at different scales. The Neck then refines and combines these features through up sampling and concatenation, making sure information from multiple resolutions is preserved. Finally, the Head performs the actual object detection at three different scales, allowing the model to identify both small and large objects effectively. In simple terms, YOLOv11 first learns useful details from the image, then enhances them, and finally detects objects with high accuracy.

4. IMPLEMENTATION

4.1 Environment Setup and Hardware Configuration

Model training was carried out on a cloud GPU instance fitted with an NVIDIA Tesla T4 card, whose availability was first checked using the standard NVIDIA system tool before training began. The core Ultralytics package was then installed to provide all the YOLOv11 training and inference commands used through the rest of the project.

4.2 Dataset Acquisition via Roboflow API

Instead of manually downloading and unzipping files, the labelled dataset was pulled directly into the training notebook using the Roboflow Python API. This automatically downloaded the annotated images together with a data.yaml configuration file that tells the training script the class names and file locations to use.

4.3 Model Training Implementation

Training was started from the pre-trained yolo11n.pt weights so that the network did not have to learn basic shapes from zero, a common and effective trick known as transfer learning. The model was then trained for 60 epochs at an image size of 640×640, after which the best-performing weights were automatically saved for later use inside the dashboard.

4.4 Streamlit Dashboard Implementation

The dashboard was built with Streamlit because it turns a normal Python script into an interactive web page with very little extra code. The trained model is loaded once and cached in memory, an uploaded photo is passed to the model, and the returned bounding boxes are drawn back onto the image so the user can view the original and the annotated result side by side.

4.5 System Interface and Operational Walkthrough

Figure 5 shows the landing page of the dashboard, where a user can read a short project description and upload a photo for checking. Figure 6 shows the detection screen: the original photo appears on the left and the annotated result, complete with bounding boxes and confidence scores, appears on the right, together with a clear status card underneath so that even a non-technical viewer can understand the result immediately.

5. RESULT ANALYSIS

5.1 Training Loss Convergence

The model was trained for 60 epochs, and three kinds of loss were tracked throughout: box loss (how well a predicted box matches the true one), class loss (how well the model tells a helmet apart from a bare head), and distribution focal loss (how sharp the predicted box edges are). Table 3 lists these values at sample epochs, and Figure 7 plots them across the complete run. As training progressed, all three losses steadily decreased while the mAP score climbed upward, which is exactly the pattern expected from a model that is genuinely learning rather than memorising the training images.

5.2 Performance Metrics (Precision, Recall and mAP)

On the 120 unseen validation images, the final model reached an overall precision of 77.7%, a recall of 90.1%, an mAP@0.5 of 91.3%, and an mAP@0.5:0.95 of 63.1%. The high recall value matters most for a safety tool, since it means the system rarely misses a worker who is actually present in the frame, keeping false "all-clear" signals to a minimum. Table 4 breaks these numbers down by class: the Helmet class, which had far more training examples, reached a strong 94.0% mAP@0.5, while the smaller No-Helmet class still reached a useful 88.7% mAP@0.5 even with only ten validation examples.

5.3 Confusion Matrix Evaluation

Figure 8 shows the confusion matrix, a simple grid comparing the model's prediction against the true label. Around 90% of both helmet and no-helmet cases landed on the correct diagonal cell, and most of the remaining mistakes were the model predicting "background" instead of a real object, rather than confusing a helmet with a bare head. This is a reassuring result, since it means the two safety classes are rarely swapped with one another.

Table 3: Training Performance Convergence Metrics Across Epochs

Epoch	Box Loss	Class Loss	DFL Loss	mAP@0.5	mAP@0.5:0.95
1	1.2510	2.6680	1.3880	0.2110	0.1180
10	1.2760	1.1450	1.3730	0.8320	0.4520
20	1.1100	0.8362	1.2590	0.8530	0.5190
30	1.0220	0.7013	1.2130	0.8970	0.5610
40	0.9612	0.6191	1.1650	0.9200	0.5730
50	0.8609	0.5340	1.0970	0.9240	0.6220
60	0.7704	0.4180	1.0330	0.8820	0.5970

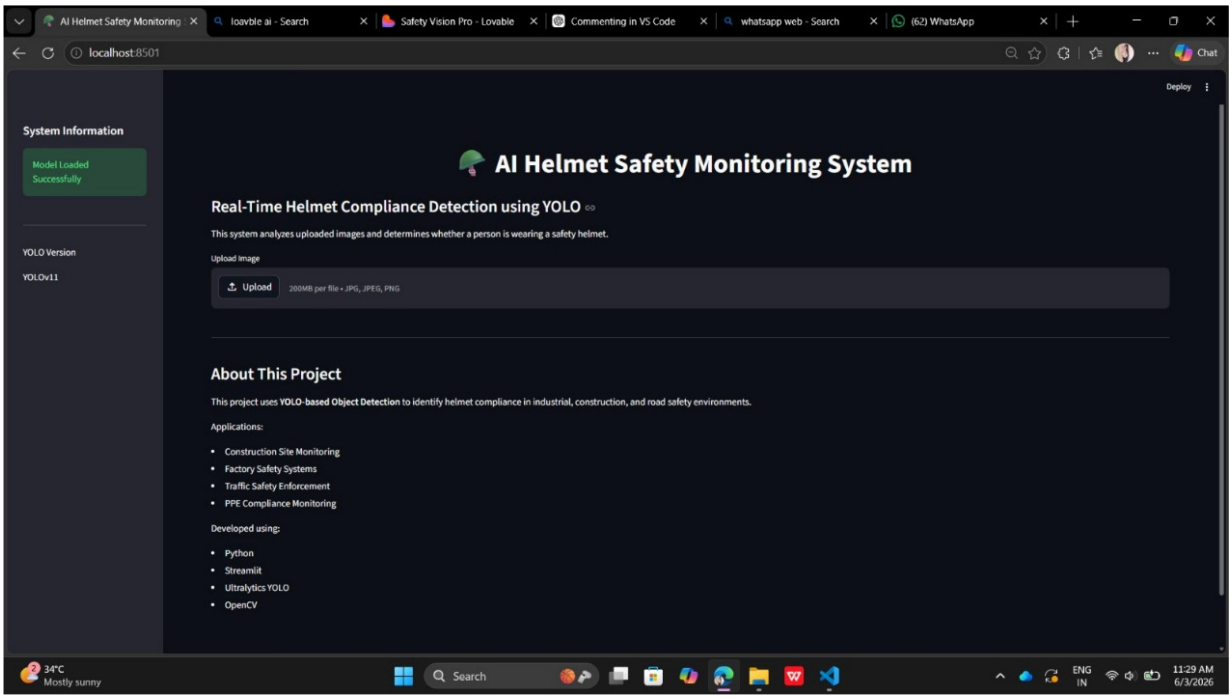


Figure 5: Dashboard Landing Page.

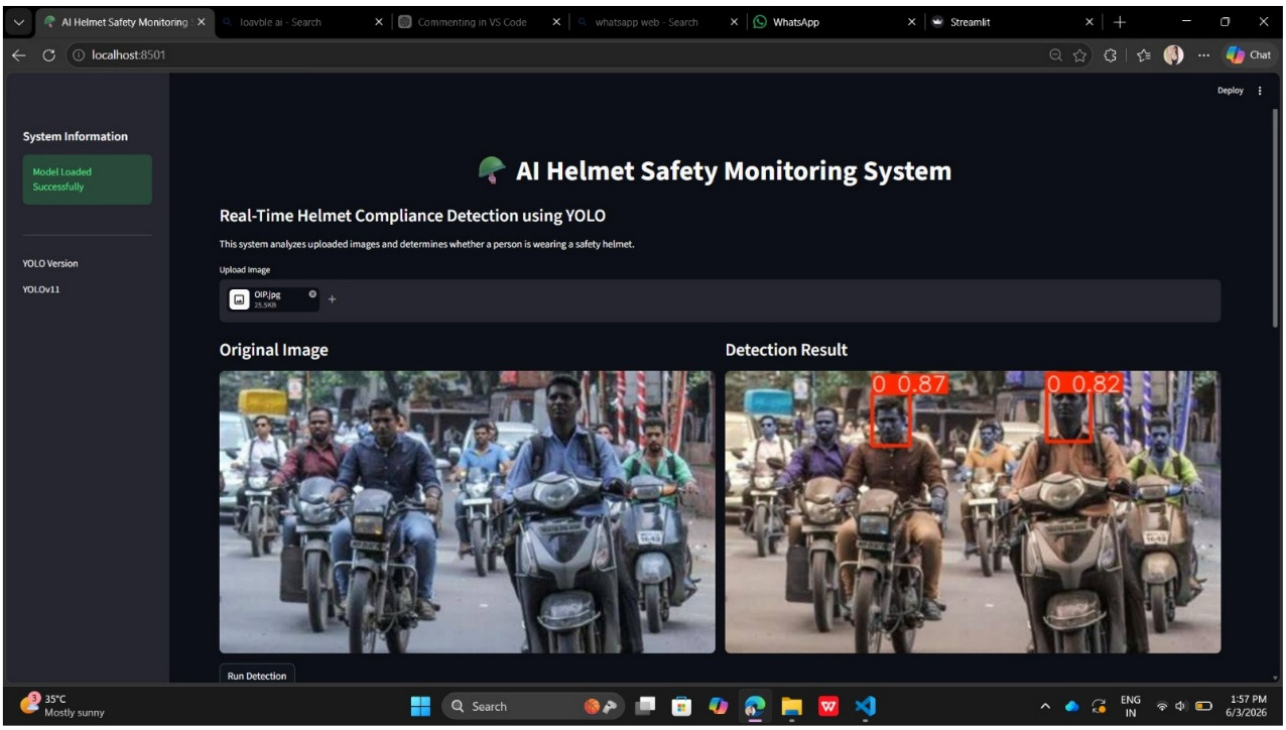


Figure 6: Real-Time Detection Result View.

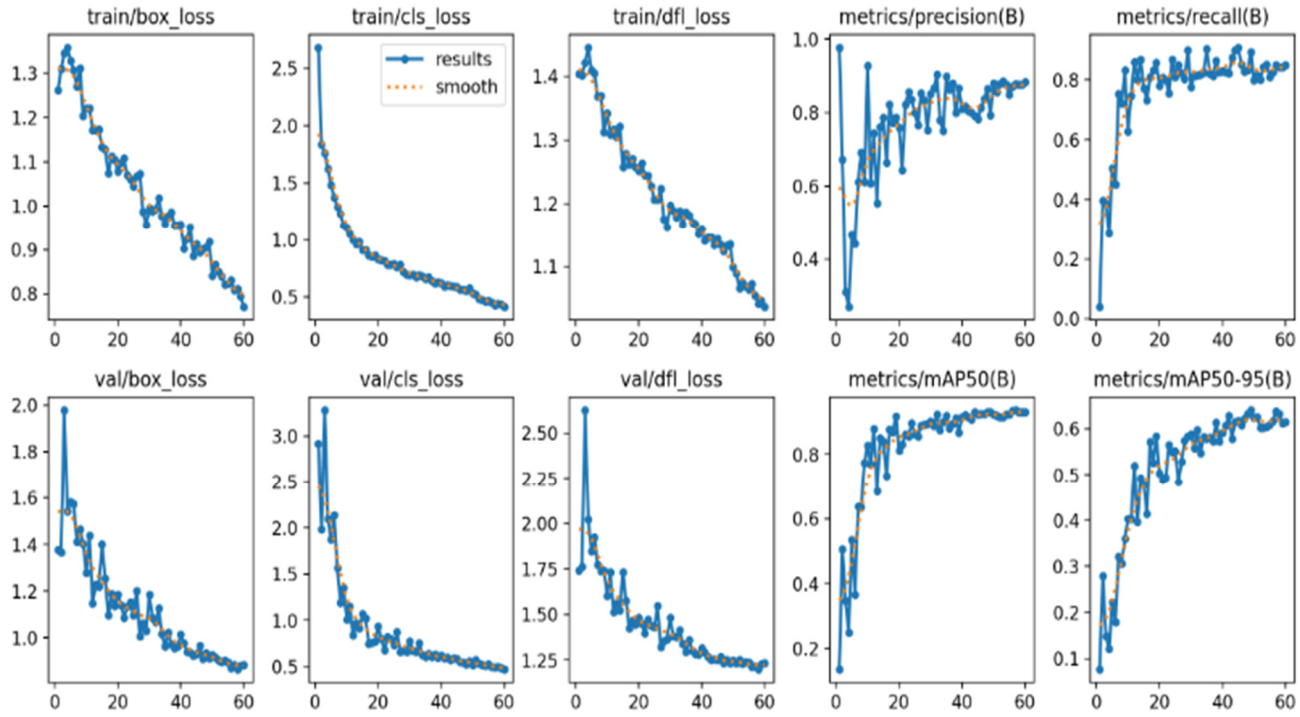


Figure 7: Training and Validation Loss Curves Across 60 Epochs.

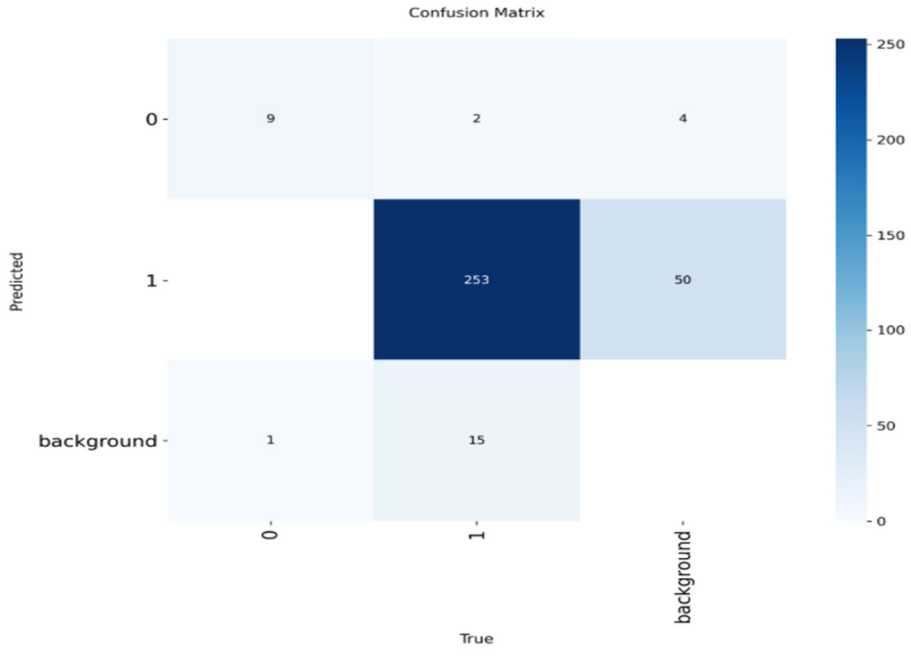


Figure 8: Confusion Matrix on the Validation Set.

Table 4: Class-Wise Validation Performance Metrics

Object Class	Instances	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Class 0 (No Helmet)	10	0.642	0.898	0.887	0.663
Class 1 (Helmet)	270	0.913	0.904	0.940	0.599

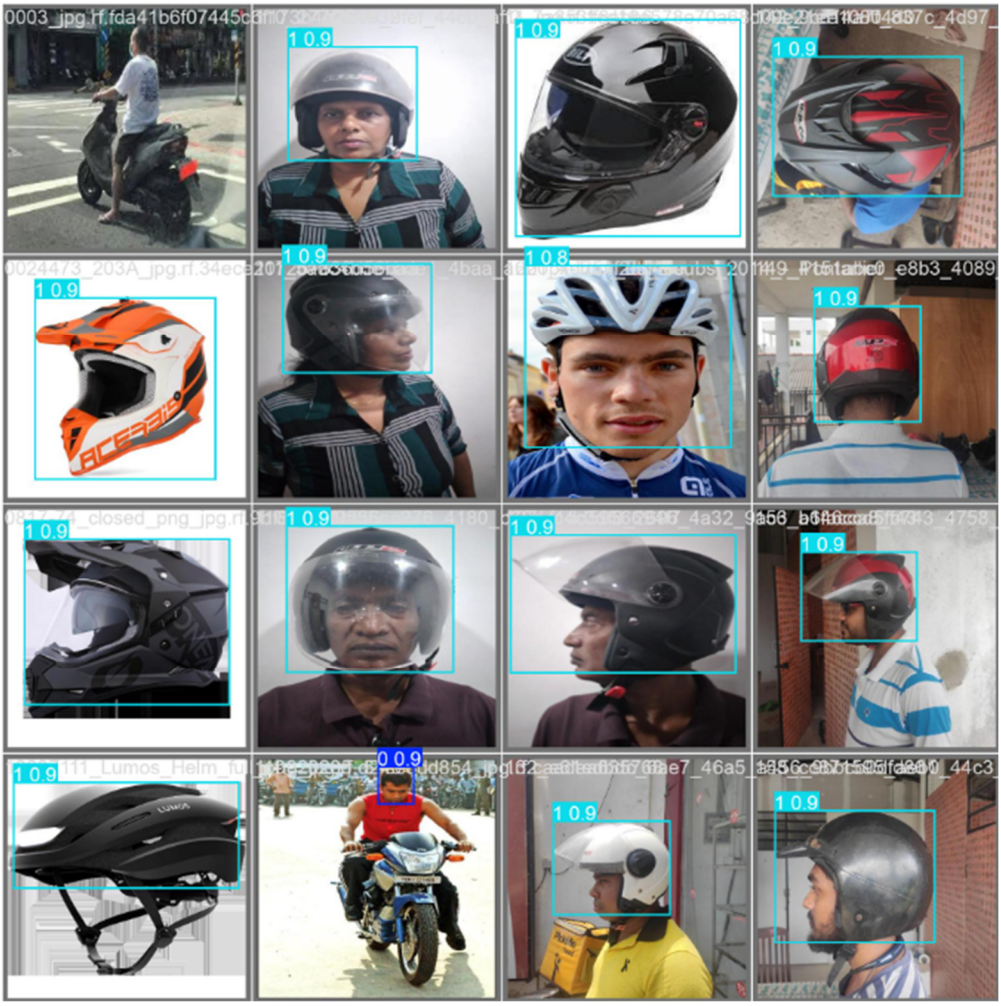


Figure 9: Sample Helmet Detection Results (0: No Helmet, 1: Helmet).

Table 5: Processing Latency Breakdown (Per Frame)

Pipeline Phase	Execution Time
Image Pre-processing	0.3 ms
YOLOv11 Deep Convolutional Inference	5.2 ms
Post-processing (NMS)	6.3 ms
Total Pipeline Latency	11.8 ms

5.4 Detection Result Samples

Figure 9 shows a sample of test images with the model's predicted boxes drawn on them. The system correctly located helmets of different colours and shapes, on people at different distances from the camera, and even inside a busy traffic scene, showing that it generalises reasonably well beyond simple, posed photographs. Therefore, from the pictures shown in the figure, it is clear that utmost detection is possible using the described techniques.

5.5 Inference Latency and Computational Benchmarks

Real-time video needs a processing speed faster than 30 frames per second, which means less than about 33 milliseconds per frame. Table 5 shows that the trained model, with only about 2.6 million parameters, completed the full pipeline (pre-processing, inference, and box clean-up) in around 11.8 milliseconds per image on the Tesla T4 GPU - roughly 85 frames per second, well within the real-time requirement.

5.6 Failure Mode Observations

A few edge cases were also noticed during testing. Workers who were very far from the camera sometimes occupied too few pixels for the model to notice them, and a helmet covered more than about three-quarters by a pipe or another worker was occasionally missed altogether. Strong glare from direct sunlight or a welding arc also sometimes washed out the colour of a helmet enough to push the model's confidence score below the detection threshold.

6. CONCLUSION AND FUTURE WORK

This project developed and successfully tested a real time helmet detection system using YOLOv11. It also worked very fast for live video, at around 85 frames per second. This model is connected with streamlit dashboard. This helps for user who didn’t have any coding background to check weather workers are wearing helmet or not and also this model reduces the time for checking construction sites. It has one drawback that it’s difficult to detect when object is far from the camera, in crowd and when there is strong light or glare. Overall, this project shows that a lightweight deep learning model with a simple web interface can provide a fast and adorable solution for safety monitoring at construction sites.

REFERENCES

[1] K. Dahiya, D. Singh and C. K. Mohan, "Automatic detection of bike riders without helmet using surveillance videos in real-time," in Proc. IJCNN, Vancouver, BC, Canada, July 2016, pp. 3046-3051.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[3] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in Proc. IEEE CVPR, Miami, FL, USA, 2009, pp. 248-255.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in NIPS 25, Lake Tahoe, NV, USA, 2012, pp. 1097-1105.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE TPAMI, vol. 39, no. 6, pp. 1137-1149, June 2017.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE CVPR, Las Vegas, NV, USA, 2016, pp. 779-788.

[7] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in Proc. IEEE CVPR, Honolulu, HI, USA, 2017, pp. 6517-6525.

[8] Tushar Kale, et al., "Helmet Detection and Number Plate Recognition," IJRASET, vol. 11, issue V, pp. 4038-4041, May 2023.

[9] Manisha Pise et al., "Helmet and Number Plate Detection Using YOLOv5," IJARSCT, 2024.

[10] N. Gosavi, et al., "Helfine: Real-Time Helmet Violation Detection with Automated Fine System," IJISRT, vol. 10, issue 2, 2025.

[11] Hari Krishnan, R., "Enhancing Road Safety with Real-Time Helmet Detection and Challan Issuance Using YOLO and OCR," IRJMETS, 2024.

[12] J. Majumder, "Real-time YOLO-based monitoring for helmet and number plate detection," 2022.

[13] D. Singh, C. Vishnu and C. K. Mohan, "Visual big data analytics for traffic monitoring in smart city," in Proc. IEEE ICMLA, Anaheim, CA, USA, 2016, pp. 18-20.

[14] A. S. Kapse, S. Shreevamshi, P. Ravichandra, R. Reddy, and R. Reddy, "A Survey on Helmet Detection by CNN Algorithm," ITM Web of Conferences (ICDSAC 2023), vol. 56, 05004, 2023.

[15] P. Tripathi, et al., "A Review on Helmet and Number Plate Detection," IJRPR, vol. 4, no. 5, pp. 2473-2484, May 2023.

[16] P. Ahuja and K. Bhavsar, "Microcontroller Based Smart Helmet Using GSM & GPRS," in Proc. ICOEI, Tirunelveli, India, 2018, pp. 917-921.

[17] M. Uniyal, H. Rawat, M. Srivastava and V. K. Srivastava, "IOT based Smart Helmet System with Data Log System," in Proc. ICACCCN, Greater Noida, India, 2018, pp. 1-6.

[18] M. Jeong, H. Lee, M. Bae, D. B. Shin, S. H. Lim and K. B. Lee, "Development and Application of the Smart Helmet for Disaster and Safety," in Proc. ICTC, Jeju, Korea, 2018, pp. 1060-1062.

[19] Jocher, G., Stubs, F., et al., "Ultralytics YOLO" (Computer software), GitHub.

[20] Y. Wu, M. Chen, G. Li, W. Wang, and G. Zhang, "A comprehensive survey on the application of deep learning in intelligent transportation systems," IEEE Access, vol. 9, pp. 32812-32837, 2021.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 49-53 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

Smart Home Automation and Security System Using Gesture Recognition, Voice Assistance, IoT, and Machine Learning

[1]Anik Kapat, [1]Aditya Kumar, [1]Honey Yadav, [1]Nanda Kishor Manna, [1]Muskan Sureka,

[2] Tilak Mukherjee

[1]UG student, Department of Electronics of Communication Engineering Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[2]Department of Electronics of Communication Engineering Engineering, Haldia Instituteof Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

In order to offer a touchless and intelligent home control solution, the suggested Smart Home Automation and Security System combines gesture recognition, voice assistance, IoT connectivity, and machine learning. With an ESP32-based architecture, the system allows users to operate appliances with voice requests and hand gestures. Motion detection and facial recognition work together to improve security by capturing evidences during illicit entry and generating alarms. Environmental sensors automate appliance functioning by continuously monitoring temperature and humidity. The system is a scalable and affordable option for contemporary smart homes since it offers real-time monitoring, control, and security upgrades via a centralized web and mobile dashboard.

KEYWORDS: Gesture Recognition; Voice Assistant; Internet of Things (IoT); Machine Learning; Home Automation; Smart Security

1. INTRODUCTION

Home management systems have advanced far beyond basic remote-controlled switching in the modern world due to the increased emphasis on accessibility, convenience, and residential safety. By physically pressing switches, twisting dials, or touching touch screens, the traditional approach of engaging with a home implicitly assumes that the user is present, able-bodied, and unoccupied. In a surprisingly large number of real-world scenarios, this assumption is incorrect. Elderly people might not have the mobility or dexterity to reliably operate ordinary controls. Touch-based interfaces may be completely inaccessible to people with physical disabilities. A system that responds to a gesture or a spoken word instead of requiring a free hand is beneficial even for a fully capable user who is cooking, grocery shopping, or caring for a child. The development of touch-free control mechanisms, particularly hand gesture recognition and voice command processing, which make the home environment more responsive to its occupants rather than requiring adaption from them, is motivated by these practical gaps. The possibilities for home automation have fundamentally changed due to the Internet of Things. IoT architectures make it possible to monitor and control a home from any location at any time using any networked device by connecting household equipment to a shared network and exposing them thru a common communication layer. A caregiver can validate that a medical gadget in a patient's room is functioning properly, and a user traveling overseas can check if a fan was left running. Once restricted to costly proprietary installations, this remote visibility and control capability is now possible with inexpensive microcontrollers and cloud messaging services, making it suitable for routine household deployment [1, 2].

The picture is further enhanced using machine learning. A machine-learning-powered system can identify patterns, such as a particular hand configuration, a spoken phrase in natural language, a familiar face, or an unusual temperature

*Corresponding Address: anikkapatece@gmail.com DOI 10.5281/zenodo.22038177

reading, and react to them meaningfully instead of requiring the user to interact with a fixed menu of discrete commands. As a result, the interface becomes more adaptable and flexible. It is possible to expand a gesture vocabulary without rebuilding the hardware. Without changing the underlying control logic, a voice model can be retrained to identify a new command [3]. Once restricted to costly proprietary installations, this remote visibility and control capability is now possible with inexpensive microcontrollers and cloud messaging services, making it suitable for routine household deployment.

It is impossible for a PIR sensor to determine whether someone crossing the threshold is a resident. This restriction results in frequent false alarms that are set off by pets, passing cars, or regular home movement. Over time, this erodes user confidence in the system and frequently causes alerts to be disregarded or disabled. This flaw is directly addressed by combining motion sensing and facial recognition: the system first determines that movement has taken place before determining if the responsible individual is identified. The system only escalates-disseminating an alert, setting off an alarm, and taking pictures as proof-when the face cannot be matched to an authorized occupant. This multi-layered strategy maintains a dependable barrier against actual incursion while drastically reducing false positives [4, 5].

The capability set is completed with environmental awareness. Among the most direct factors influencing indoor comfort and, in some situations, are temperature and humidity. Unless the system can recognize these variables and react, a home that becomes uncomfortably hot in the afternoon or excessively humid during the monsoon months requires continuous human attention. The backend can automatically turn on a fan, air conditioner, or dehumidifier without the occupant's input by reading from an environmental sensor and comparing measurements against user-configured thresholds. This reduces discomfort and the mental strain of constantly monitoring the home environment [2].

All of these threads are combined into a single, cohesive platform by this initiative. Hands-free appliance control is made possible by voice interaction and gesture detection. Device communication and switching are managed by an IoT-capable microcontroller. Access control is enforced with few false alarms when face recognition and motion sensing are integrated. Automated climate reactions are powered by environmental sensors. Additionally, a single web and mobile interface connects everything, offering environmental readings, security alerts, and real-time status updates while maintaining the user's ability to manually override any automated decision at any moment. The system's components, operation, and experimental validation are all thoroughly explained in the sections that follow.

2. LITERATURE REVIEW

Recent advancements in the Internet of Things (IoT) have significantly transformed home automation systems by enabling remote monitoring and control of household appliances. Traditional IoT-based home automation systems primarily focus on smart phone-controlled switching and environmental monitoring. While these systems improve convenience, they often rely heavily on manual interaction and provide limited accessibility for elderly and physically challenged users [1, 6].

Gesture recognition technology has emerged as a promising alternative to conventional input methods. Computer vision and machine learning techniques allow users to control devices through natural hand movements without physical contact. However, most gesture-based systems focus solely on appliance control and do not integrate security mechanisms [3].

Voice-controlled home automation systems further enhance user convenience by enabling hands-free interaction through speech commands. Modern speech recognition technologies have achieved high accuracy, but voice-only systems may face challenges in noisy environments and often lack robust security features [4]. Similarly, many smart security systems employ motion sensors or surveillance cameras independently. PIR-based motion detection systems frequently generate false alarms because they cannot distinguish between authorized and unauthorized individuals. Face

recognition technology addresses this limitation by providing identity verification capabilities [5].

The proposed system combines gesture recognition, voice assistance, IoT connectivity, environmental monitoring, motion detection, and face recognition into a unified platform. This integration enhances accessibility, security, and automation while reducing the limitations associated with standalone solutions [1-6].

3. SYSTEM ARCHITECTURE

3.1 Input Layer

The input layer is in charge of gathering environmental data and user commands. It has a camera for facial recognition and gesture recognition, a microphone for spoken command acquisition, a PIR sensor for motion detection, and a DHT11 sensor for temperature and humidity monitoring. Data is continuously collected by these devices and sent to the processing layer for analysis [2, 3, 5].

3.2 Processing and Communication Layer

Security validation, environmental analysis, speech processing, and gesture recognition are all carried out by the processing layer. Computer vision algorithms analyze camera frames to recognize allowed faces and predetermined hand motions. Spoken commands are translated into machine-readable instructions by a speech-to-text engine. All decision-making is coordinated by a centralized backend that uses MQTT-based communications to interact with the ESP32 microcontroller [1, 4].

3.3 Device Control Layer

The ESP32 microcontroller, relay modules, alarm unit, and linked appliances make up the device-control layer. The ESP32 reports the status of applications to the dashboard and activates or deactivates them in response to orders received from the backend. Additionally, this layer controls environmental automation and security alerts [6].

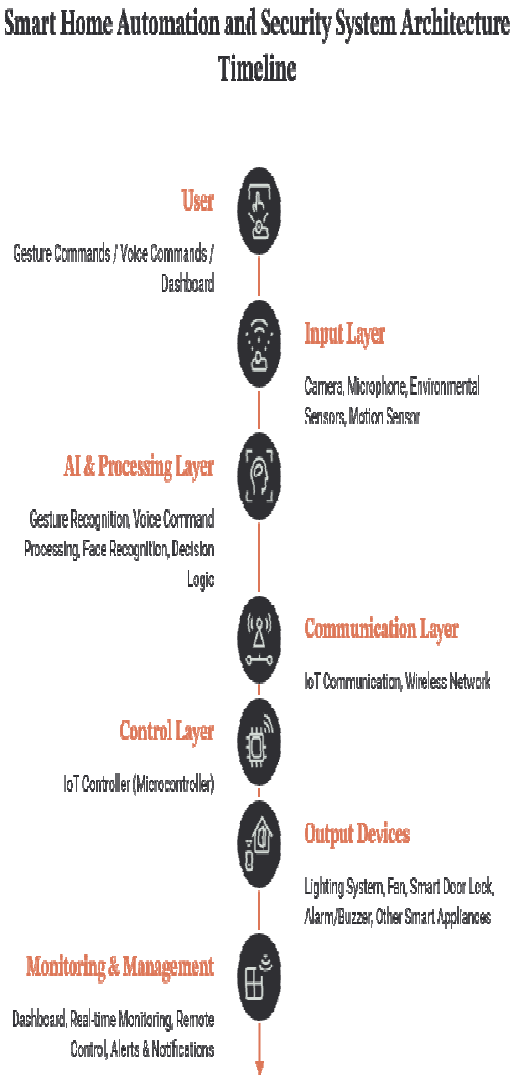


Figure 1: Smart home automation based architecture timeline

4. METHODOLOGY

4.1 Gesture and Voice-Based Appliance Control

The technology offers two ways to interact: voice instructions and hand gestures. A gesture-recognition module processes camera frames to classify predetermined hand configurations into commands for controlling appliances. Concurrently, a speech-recognition engine translates spoken commands into text and associates them with appropriate appliance operations. After being verified, both command

streams are sent to the backend for processing [3, 4].

4.2 IoT Communication and Device Switching

The backend uses MQTT communication to send the instruction to the ESP32 microcontroller once a valid command has been created. After determining the proper relay channel, the ESP32 adjusts the appliance's state. To ensure real-time synchronization, updated device status is instantly sent back to the dashboard [1, 6].

4.3 Smart Security Monitoring

A PIR motion sensor and face-recognition module work together to provide security monitoring. Using the camera stream, motion detection initiates facial analysis. The event is recorded without taking any further action if the observed person fits the profile of an authorized user. The system sounds an alarm, takes pictures as proof, and notifies the user if the face cannot be identified or is missing [5].

4.4 Environmental Monitoring and Automation

The DHT11 sensor's temperature and humidity data are routinely compared to user-specified criteria. The backend automatically turns on appropriate appliances, like fans or air conditioners, when environmental conditions surpass predetermined thresholds. The mechanism returns the appliances to their original state once readings are within acceptable bounds [2].

5. PROPOSED IMPLEMENTATION OUTCOMES

It is anticipated that the suggested Smart Home Automation and It is anticipated that the suggested Smart Home Automation and Security System will offer a It is anticipated that the suggested Smart Home Automation and Security System will offer a smooth and user-friendly way to operate household equipment using voice and gesture commands. Reliable remote monitoring and device management via a centralized dashboard are anticipated to be made possible by the integration of IoT connection.

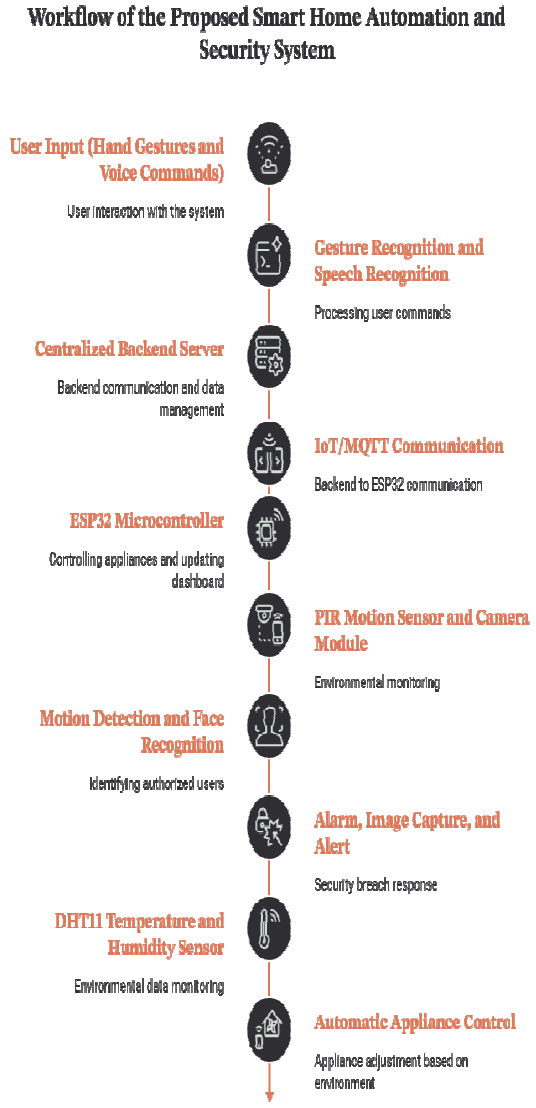


Figure 2: Workflow diagram of the proposed smart home secure automation system

By lowering the false alarms frequently connected to traditional motion-detection systems, the combined application of PIR-based motion sensing and facial recognition is anticipated to increase security. By automatically controlling linked appliances based on predetermined thresholds, environmental monitoring via temperature and humidity sensors is expected to improve user comfort.

Overall, it is anticipated that the suggested system will enhance domestic security, energy efficiency, accessibility, and convenience while

preserving a modular architecture that can be expanded in subsequent deployments.

6. CONCLUSION

The proposed system demonstrates how voice interaction, gesture recognition, IoT connectivity, and machine learning can be combined into a unified, modular home-management platform that addresses security, accessibility, and convenience as a whole rather than as separate add-ons. The integrated use of motion sensing and face recognition for access control, combined with environment-aware automation and a centralized dashboard, shows the technique's potential for assistive technology, elder care, and general residential and small-commercial application. Future studies shall focus on expanding the platform to incorporate more sensors and appliances, improving identification accuracy in various illumination and noise settings, and expanding the language of gestures and voice instruction

REFERENCES

[1] Sivapriyan, R., Rao, K. M., & Harijyothi, M. (2020). Literature review of IoT based home automation system., Fourth International Conference on Inventive Systems and Control (ICISC), pp.101–105.

[2] Maswadi, K., Ghani, N. A., Hamid, S., & Rasheed, M. B. (2020). Systematic literature review of smart home monitoring technologies based on IoT for the elderly. IEEE Access, 8, 92244–92261. doi:10.1109/ACCESS.2020.2992727

[3] Sen, A., Mishra, T. K., & Dash, R. (2024). Novel human machine interface via robust hand gesture recognition system using channel pruned YOLOv5s model. arXiv preprint arXiv:2407.02585.

[4] Saini, S. K., Singh, A., & Singh, J. (2023). Review on voice controlled home automation system. International Journal of Engineering Research & Technology (IJERT), 12(5).

[5] Sruthy, S., Yamuna, S., & George, S. N. (2020). An IoT based active building surveillance system using Raspberry Pi and NodeMCU. arXiv preprint arXiv:2001.11340.

[6] Kabir, M. H., & Ahmed, M. T. (2020). IoT based low-cost smart home automation and security system using wireless technology. International Journal of Wireless and Microwave Technologies.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 54-59 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

DESIGN AND PROTOTYPE IMPLEMENTATION OF AN IoT ENABLED SMART POTABLE WATER MONITORING SYSTEM

[1]Kuntal Maiti, [2]Mehabub Alam Shah, [3]Arindam Khatua, [4]Nilkantha Reja, [5]Saket Kumar, [6]Kushal Roy

[1,2,3,4,5]UG student, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal
[6]Assistant Professor, Dept. of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Water quality monitoring is esential for drinking-water safety and environmental sustainability. Conventional grab sampling followed by laboratory analysis is accurate but time-consuming, costly, and unable to provide immediate alerts. This paper presents the design, implementation, and validation of a low cost Internet of Things (IoT)-enabled smart potable water monitoring system based on the ESP32 microcontroller . The prototype integrates pH, Total Dissolved Solids (TDS), turbidity, and DS18B20 temperature sensing. Analog signals are sampled repeatedly, averaged, converted into engineering units through calibration models, and processed locally using threshold-based decision logic. An I2C SSD1306 OLED and RGB LED provide local feedback, while Wi-Fi and Blynk support remote monitoring. Reference-condition calibration produced reported errors of 0.4% for temperature, 1.4% for pH, 1 .7% for TDS, and 4.0% for turbidity . Tests on diverse water samples demonstrated useful real-time screening capability. Analysis of the supplied firmware also identified a safety gap: the original NEUTRAL branch did not constrain pH, allowing some strongly acidic or alkaline samples to be classified as neutral. A corrected multi-parameter decision rule is therefore proposed. The prototype demonstrates the feasibility of low-cost edge-based water-quality screening while highlighting the need for improved ADC accuracy, periodic sensor calibration, and rugged field deployment.

KEYWORDS: IoT, ESP32, water quality monitoring, pH, TDS, turbidity, DS18B20, Blynk, edge computing, sensor calibration.

1. INTRODUCTION

Potable water resources are increasingly affected by population growth, urbanization, agricultural runoff, untreated wastewater, and industrial discharge. The supplied project report emphasizes continuous monitoring because transient contamination events can be missed by manual grab-sampling. The principal parameters considered are pH, TDS, and turbidity, supported by temperature measurement for compensation and validation. The report identifies three major engineering challenges for low-cost monitoring nodes: high-impedance analog signals are vulnerable to interference, the ESP32 internal ADC exhibits non-linearity, and pH/EC related measurements are temperature dependent. The project therefore combines regulated power, decoupling, repeated sampling, calibration, compensation, local indication, and wireless telemetry. The main objectives are to develop a multi-sensor acquisition system; stabilize analog inputs using an LM7805 regulator and decoupling; implement calibration and temperature-related processing; provide OLED/RGB status indication; and support Wi-Fi/cloud telemetry.

2. THEORY AND DESIGN BASIS

The proposed architecture follows the conventional IoT chain of sensing, embedded processing, local visualization, and remote telemetry. The project adopts a nominal pH potability range of 6.50–8.50, while its broader report discusses TDS and turbidity as complementary indicators. The report's introductory standards table lists TDS below 300 ppm as an excellent level and below 500 ppm as a secondary limit, with turbidity below 1 NTU as strict and below 5 NTU as acceptable. The pH electrode produces an electrochemical potential related to hydrogen-ion activity. The report

*Corresponding Address: kushalroy1979@gmail.com
DOI 10.5281/zenodo.22038233

describes a linear calibration model, pH = Slope × VpH + Offset, with temperature-dependent Nernstian sensitivity. TDS is derived from electrical conductivity, where dissolved ions increase conductivity. Turbidity is obtained optically from light scattering/attenuation caused by suspended particles.

3. COMPONENTS USED

3.1. ESP32 Microcontroller

The ESP32 is the central processing unit. The project uses its ADC inputs for pH, TDS and turbidity, its 1-Wire interface for the DS18B20, I2C for the OLED, PWM-capable GPIOs for the RGB LED, and built-in Wi-Fi for telemetry. The architecture therefore minimizes external processing hardware and allows calibration and classification to occur at the edge.

3.2. pH Sensor

The pH sensing assembly uses a glass-electrode interface and conditioning board. Standard buffer solutions at pH 4.0, 7.0, and 10.0 are used for calibration. The supplied implementation reports a fitted slope of −5.70 and offset of 21.34.

3.3. TDS Sensor

The TDS sensor operates using the electrical-conductivity principle. Dissolved salts and minerals provide ions that increase conductivity. The sensor output is mapped to TDS in ppm using a conversion factor; the implementation uses an approximate 500 ppm/V scaling.

3.4. Turbidity Sensor

The TS-300/380 turbidity sensor contains an infrared LED and phototransistor. Clear water transmits more light, whereas suspended particles scatter or absorb light and reduce the detected intensity. The resulting electrical signal is converted to NTU.

3.5. DS18B20 Temperature Sensor

The DS18B20 provides digital temperature through a 1-Wire connection. In the supplied project, the device is validated against a reference thermometer and is also used as a useful variable for temperature-dependent processing.

3.6. SSD1306 OLED and RGB LED

The SSD1306 OLED provides local numerical display. A common-cathode RGB LED indicates SAFE, NEUTRAL, and UNSAFE states using green, blue, and red, respectively. The report specifies GPIO25, GPIO26 and GPIO27 for the three LED channels.

3.7. Power Regulation and Supporting Hardware

An LM7805 linear regulator establishes a regulated +5 V rail. A 1.00 µF decoupling capacitor is connected at the regulator output to suppress line oscillations. The ESP32 VIN receives the regulated rail, while its +3.3 V output supplies low-draw digital modules.

Table 1: Hardware Connection

COMPONENT	ESP32 CONNECTION	SIGNAL / PURPOSE
LM7805 OUT	VIN	+5V regulated supply
pH sensor	GPIO34	Analog input
TDS sensor board	GPIO33	Analog input
Turbidity sensor board	GPIO32	Analog input
DS18B20	GPIO4	1-Wire; 4.7 kΩ pull-up
SSD1306 OLED	GPIO21/22	I2C SDA/SCL
RGB LED	GPIO25/26/27	PWM status outputs

The complete pin-connection table in the supplied report gives the same GPIO allocation and identifies the signal type of each connection.

4. SYSTEM IMPLEMENTATION

4.1. Hardware Architecture

The physical implementation connects the water sensors to the ESP32, with the analog sensors routed to GPIO32–34 and the DS18B20 connected to GPIO4. The OLED uses I2C, and the RGB LED provides local status. The report specifies common grounding and isolation of analog connections, with the OLED selected for its low GPIO requirement.

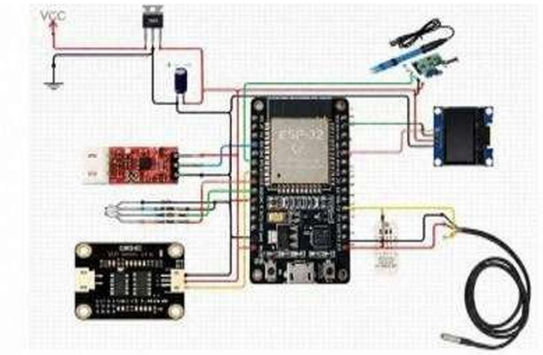


Figure 1: Circuit diagram and physical interconnections of the prototype.

4.2. Data Acquisition

Each firmware loop executes approximately every three seconds. The DS18B20 is requested first, followed by repeated analog sampling. The implementation takes 40 samples for turbidity, 30 for TDS, and 15 for pH. The average ADC reading is converted to voltage using a 3.3 V reference and 12-bit resolution.

For averaged ADC code A, the report uses V = A(3.3/4095). The implementation then calculates turbidity as NTU = (3.3 − Vturb) × 1000 with a 0–3000 NTU clamp, TDS as TDS(ppm) = VTDS × 500, and pH using the calibrated linear slope and offset.

4.3. Calibration Procedure

Temperature was validated at 0 °C, 25 °C and 50 °C, with an offset below 0.5 °C. The pH sensor was calibrated with pH 4.0, 7.0 and 10.0 buffers. TDS was calibrated using KCl solutions, while turbidity used reference solutions including 0 NTU distilled water and a 200 NTU suspension.

Table 2: CALIBRATION RESULT

PARAMETER	REFERENCE	SYSTEM	ERROR
Temperature	25.0 °C	25.1 °C	0.4%
pH	7.00	7.10	1.4%
TDS	300 ppm	305 ppm	1.7%
Turbidity	5.0 NTU	5.2 NTU	4.0%

The supplied report notes that periodic recalibration is recommended to compensate for sensor drift.

4.4. Remote Monitoring

Blynk IoT Cloud is used to visualize water-quality parameters remotely. The dashboard displays TDS, temperature, pH and turbidity through gauges/value widgets, enabling the operator to observe the prototype through an internet connection.

5. FIRMWARE AND DICISION LOGIC

The firmware processes the sensor signals, displays the numerical results on the OLED, sets the RGB status, and can transmit values wirelessly. The OLED screen presents temperature, pH, TDS, turbidity and the overall status, followed by a three-second delay before the next acquisition cycle.

The original classification logic is: SAFE when pH is 6.5–8.5, TDS <300 ppm and turbidity <50 NTU; otherwise NEUTRAL when TDS <600 ppm and turbidity <200 NTU; otherwise UNSAFE.

5.1. Identified Safety Gap

The supplied report performs a critical analysis of this logic. The NEUTRAL branch checks only TDS and turbidity and does not check pH. Consequently, a sample with extreme acidity or alkalinity can fail the SAFE condition because of pH, but still satisfy the relaxed TDS/turbidity limits and be labelled NEUTRAL. The report demonstrates this with acidic rainwater and alkaline algal-bloom water.

5.2. Corrected Multi-Tier Decision Architecture

The report proposes explicitly constraining pH in the NEUTRAL state: SAFE = [(6.5 ≤ pH ≤ 8.5) AND (TDS <300) AND (NTU <50)]; NEUTRAL = [(6.0 ≤ pH ≤ 9.0) AND (TDS <600) AND (NTU <200)]; otherwise UNSAFE.

This ensures that pH below 6.0 or above 9.0 is immediately flagged UNSAFE regardless of TDS and turbidity.

5.3. Processing Flow

The complete embedded processing sequence can therefore be represented as: initialize sensors and communications → acquire temperature and repeated ADC samples → average ADC values → convert voltages to pH/TDS/NTU → apply decision thresholds → update OLED and RGB LED → publish data to Blynk → repeat after approximately three seconds.

5.4. Prototype Integration

The finished prototype places the ESP32, regulator, display, sensor boards and wiring

inside a compact enclosure. The report concludes that regulated power and decoupling improve stability during active Wi-Fi operation, while calibration and software processing mitigate measurement drift and ADC effects.

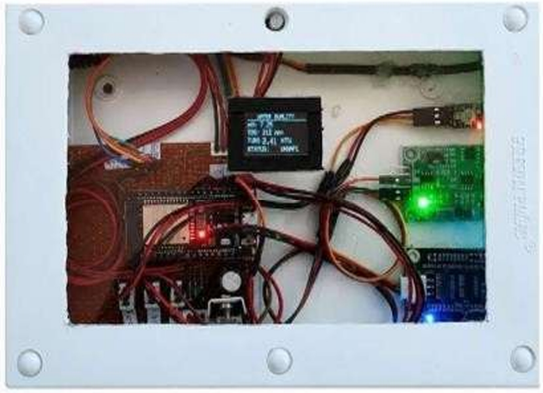


Figure 2: Finished prototype assembled inside the enclosure.

6. RESULTS AND DISCUSSION

The fully assembled prototype was tested using a range of water samples. The edge-processing layer analysed each sample, categorized the safety status, and updated the visual indicators. The expanded dataset in the supplied report is summarized below.

Table 3: Water Quality Test Results

SAMPLE	TEMP (°C)	pH	TDS	NTU	STATUS
Distilled	24.2	6.95	4	0.0	SAFE
Municipal tap	24.8	7.32	152	0.8	SAFE
RO purified	24.5	6.65	18	0.1	SAFE
Bottled mineral	22.5	7.20	210	0.3	SAFE
Chlorinated pool	26.1	7.40	280	1.5	SAFE
High-mineral well	18.2	6.42	420	8.5	NEUTRAL
Brackish estuary	20.8	7.85	550	45.0	NEUTRAL
Natural rainwater	16.5	6.20	25	1.2	NEUTRAL
Acid rainwater	15.4	4.50	85	5.0	NEUTRAL
Algal bloom lake	23.4	8.70	180	75.0	NEUTRAL
Agricultural runoff	21.5	5.65	618	38.4	UNSAFE
Industrial discharge	27.9	9.42	1115	124.0	UNSAFE
Sea-water intrusion	19.5	8.10	1000	12.0	UNSAFE
Domestic sewage	19.5	6.80	750	320.0	UNSAFE

The largest reported reference-condition error is 4.0% for turbidity. The pH and TDS errors are 1.4% and 1.7%, respectively, while temperature error is 0.4%. These values indicate useful prototype-level screening performance but do not constitute certified laboratory measurement.

6.1. Safe and Neutral Samples

Distilled and RO water show very low dissolved solids and negligible turbidity and are classified SAFE. Municipal tap, bottled mineral and chlorinated pool water also remain inside the project's safe ranges. High-mineral well and brackish estuary water exceed the ideal 300 ppm TDS level but remain below the original neutral thresholds and are therefore classified NEUTRAL.

6.2. Firmware Vulnerability Demonstration

Natural rainwater and acid rainwater have low particulate loading but acidic pH values. Because the original NEUTRAL branch ignores pH, both can be labelled NEUTRAL rather than UNSAFE. Algal bloom lake water at pH 8.70 is similarly

reported NEUTRAL because its TDS and turbidity remain below the relaxed neutral limits.

6.3. Highly Contaminated Water

Agricultural runoff, industrial discharge, sea-water intrusion and domestic sewage fail the relevant thresholds and are classified UNSAFE. The report notes that seawater intrusion reaches the low-cost TDS board's 1000 ppm scaling limit, while domestic sewage exhibits very high turbidity.

6.4. Calibration Accuracy and Discussion

The results demonstrate the value of combining several independent indicators rather than relying on a single sensor. At the same time, the safety logic must treat extreme chemical conditions as dominant faults. The report's corrected logic is therefore a significant improvement over the original implementation because it prevents the blue/neutral state from masking extreme pH conditions.

7. CONCLUSION AND FUTURE WORK

This research demonstrates the design, physical implementation, and validation of an IoT-based edge-computing platform for real-time water-quality screening. By integrating pH, TDS, turbidity and temperature sensors with an ESP32, the prototype acquires, filters, calibrates, classifies, displays and transmits water-quality information. The LM7805 regulator and decoupling support stable sensor power, while repeated ADC sampling and calibration improve practical measurement stability.

The prototype successfully differentiates several water conditions and provides both local and remote feedback. However, the supplied report identifies important limitations: internal ESP32 ADC non-linearity, firmware classification gaps, and electrochemical drift/fouling of submerged pH and TDS probes.

The most important implementation improvement is the corrected decision logic, which requires pH to remain within a defined range even for the NEUTRAL state. This prevents highly acidic or alkaline samples from being hidden by acceptable TDS and turbidity values.

Future work proposed in the source report includes replacing the ESP32 internal ADC with a 16-bit ADS1115 external ADC, correcting the nested decision logic in production firmware, migrating telemetry to LoRaWAN or NB-IoT for remote low-power operation, and using an IP68-rated enclosure for field deployment.

REFERENCES

[1] A. K. et al., (2025). IoT-Based Water Quality Monitoring System. *IEEE Xplore*.

[2] ERUN, “Water Quality Monitoring System Based on IoT,” *ERUN News*, 2024.

[3] DFRobot, “Gravity Analog pH Sensor Meter Kit V2 (SEN0161-V2).”

[4] Zbotic, “ESP32 ADC Accuracy Fix: Noise and Non-Linearity Issues,” 2024.

[5] Home Assistant Community, “Inconsistent Values with ADC Sensor Platform,” 2025.

[6] Random Nerd Tutorials, “ESP32 with TDS Water Quality Sensor,” 2024.

[7] J. Ijaradar et al., "IoT-Based Water Quality Monitoring System," *International Journal of Computer Science Publications*, 2026.

[8] Safe Drinking Water Foundation, "TDS and pH," 2017.

[9] "IoT-Based Real-Time Water Quality Monitoring and Sensor Calibration for Enhanced Accuracy and Reliability," *ResearchGate*, 2024.

[10] "Water Quality Monitoring Using IoT Technologies," *PubMed Central*, 2025.

[11] "Advanced IoT Water Monitoring Systems," *PubMed Central*, 2025.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 60-64 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

DESIGN AND DEVELOPMENT OF A LOW-COST INTELLIGENT ARDUINO-BASED FIRE FIGHTING ROBOT FOR AUTOMATED FIRE DETECTION AND EMERGENCY RESPONSE

[1]Nitish Kumar, [2]Pinaki Satpathy, [3]MD Ghulam Rizwan, [4]Nirmal Kumar Singh, [5]Om Prakash Kumar, [6]Prabhash Kumar

[1,3,4,5,6]UG Student, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[2]Assistant Professor, Department of ECE, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Fires cause extensive loss of life and property, and conventional firefighting methods often place human responders at considerable risk. This paper presents the design and development of a low-cost, intelligent, Arduino-based fire fighting robot capable of automated fire detection and remote-assisted fire suppression. The system integrates an infrared flame sensor for real-time fire detection, a dual-motor driver circuit for locomotion, a compact 5V water pump for extinguishing, and a Bluetooth communication module for wireless remote operation. An Arduino Uno microcontroller, built around the ATmega328P, forms the central processing unit, coordinating sensor inputs and actuator outputs to achieve a semi-autonomous firefighting response. Upon detecting a flame, the robot alerts the operator in real time and can be directed toward the fire source, where the onboard pump is triggered to spray water and extinguish the flame. Experimental testing on a working prototype demonstrates that the proposed system offers a reliable, inexpensive, and effective platform for early-stage fire suppression, particularly suited to confined or hazardous environments where direct human intervention is dangerous. The design emphasizes modularity, low cost, and ease of replication, and is intended as a foundation for further enhancement through AI-based multi-sensor fusion, autonomous navigation, and swarm robotics.

KEYWORDS: Fire Fighting Robot, Arduino Uno, Flame Sensor, Bluetooth Control, Embedded Systems, Automated Fire Detection, Robotics.

1. INTRODUCTION

Robots are versatile machines capable of performing complex tasks with a degree of precision, reliability, and endurance that often exceeds human capability, particularly in hazardous environments. Mobile robots, in particular, are widely deployed in logistics, agriculture, security, and disaster response owing to their ability to navigate dynamic environments autonomously or semi-autonomously. Among the many domains where robotics offers a clear safety advantage, firefighting stands out: fire emergencies are inherently unpredictable, and human firefighters routinely face risks from smoke inhalation, structural collapse, and extreme heat while attempting to locate and suppress a fire source. This work proposes a semi-autonomous, low-cost fire fighting robot built around the Arduino Uno platform. The system is designed to detect a nearby flame using an infrared flame sensor and respond by alerting the operator and, upon command, navigating toward the fire and activating an onboard water pump to extinguish it. The robot can be wirelessly controlled through a Bluetooth link, allowing an operator to remain at a safe distance while directing firefighting operations. The primary objective of this work is to demonstrate that an effective, safety-enhancing firefighting aid can be built using inexpensive, readily available components, making the solution accessible for small-scale deployment in homes, laboratories, warehouses, and other confined spaces where early fire suppression can prevent escalation.

The remainder of this paper is organized as follows. Section II reviews related work in fire fighting robotics. Section III describes the

*Corresponding Address: nitishkumarbgs632@gmail.com DOI 10.5281/zenodo.22038286

proposed system architecture and hardware components. Section IV details the working methodology and control logic. Section V discusses the implementation and circuit realization. Section VI presents the results obtained from the working prototype. Sections VII and VIII discuss applications and future scope, respectively, followed by the conclusion.

2. LITERATURE REVIEW

Chenchireddy et al. [1] proposed a prototype firefighting robot built around an Arduino microcontroller that incorporates a machine learning technique to improve the accuracy of fire detection, demonstrating the growing trend of combining low-cost embedded platforms with intelligent classification methods.

Kumar et al. [2] presented an Arduino-based firefighting robot integrating a fire detection and suppression system, highlighting a compact sensor-actuator pipeline capable of identifying a flame and triggering suppression with minimal latency.

Maruf et al. [3] developed an autonomous Arduino-based firefighting robot specifically targeted at laboratory environments, where confined spaces and sensitive equipment make rapid, unattended fire response particularly valuable.

Reddy et al. [4] and Kori et al. [5] each reported an Arduino Uno-based firefighting robot design emphasizing low-cost hardware realization, sensor-driven detection, and pump-based suppression, reflecting the continued relevance of the Arduino platform for accessible firefighting robotics research.

Bankapur et al. [6] proposed a flame sensor-based firefighting assistance robot and additionally validated a simulation-based multi-robot implementation, pointing toward the extension of single-robot designs into coordinated multi-robot firefighting scenarios.

Babu et al. [7] further extended the Arduino-based firefighting robot concept by incorporating metal detection alongside fire detection and suppression, illustrating how auxiliary sensing capabilities can be layered onto a common embedded control architecture.

While the reviewed systems demonstrate a range of sensing and control strategies, most rely on either purely autonomous or purely remote-controlled operation, with only limited exploration of AI-assisted detection or multi-robot coordination in low-cost Arduino-based platforms. The system proposed in this paper adopts a semi-autonomous approach that combines automatic flame detection with operator-guided navigation and pump activation, aiming for a balance between responsiveness and operator oversight, while keeping the overall bill of materials low.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed robot is structured around four functional units: a sensing unit (flame sensor), a control unit (Arduino Uno), a locomotion unit (DC motors with motor driver), and an extinguishing unit (water pump with reservoir). A Bluetooth module provides the wireless link between the control unit and a handheld operator interface. The flame sensor continuously monitors the surroundings for infrared radiation characteristic of open flames. On detecting a fire, the microcontroller notifies the operator, who can then direct the robot toward the fire and trigger the pump to release water onto the flame.

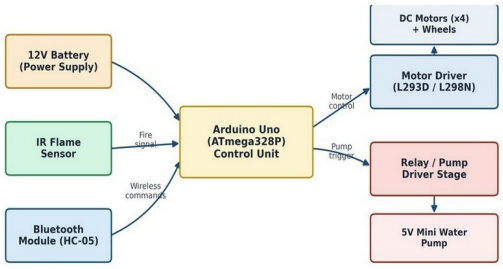


Figure 1: Block diagram of the proposed fire fighting robot system.

3.2. Hardware Components

Table I summarizes the principal hardware components used in the system, along with their function within the overall architecture.

Table I. Hardware Components and Functions

Component	Function
Arduino Uno (ATmega328P)	Central control unit; processes sensor data and issues actuator commands
Flame Sensor (IR)	Detects infrared emission from flames (760–1100 nm range)

Component	Function
L293 / L298N Motor Driver	Bidirectional control of DC motors for locomotion
DC Geared Motors (x4)	Drive the wheels for robot movement
Mini Water Pump (5V)	Sprays water onto detected flame
Bluetooth Module (HC-05)	Wireless link between robot and operator handset
LM2596 Buck Converter	Steps down battery voltage to regulated levels for logic circuits
12V Rechargeable Battery	Primary power source for motors and pump
Chassis, Wheels, Water Reservoir	Mechanical structure and water storage

3.3. Control Unit

The Arduino Uno, operating at 5V with a 16 MHz clock, forms the brain of the system. It reads the digital output of the flame sensor, processes incoming Bluetooth commands, and drives the motor and pump circuits accordingly. Its open-source IDE and extensive library support allow rapid development and straightforward modification of the control logic.

3.4. Locomotion and Extinguishing Units

Motion is achieved through four DC geared motors driven via a motor driver IC, which provides bidirectional control by switching the polarity of the voltage applied to each motor pair. The extinguishing unit consists of a 5V mini water pump connected to a small reservoir; the pump is switched on and off by the Arduino through a relay/transistor stage triggered by operator command once the robot is positioned near the fire.

3.5. Wireless Communication

A Bluetooth serial module interfaces with the Arduino through its UART pins, allowing an operator to send single-character commands (forward, backward, left, right, pump on/off) from a smartphone application. This allows the operator to remain outside the hazardous zone while maintaining full control of the robot's movement and extinguishing action.

4. METHODOLOGY

The operational logic of the robot follows a continuous sense–decide–act cycle, illustrated in the steps below.

1) Initialization: On power-up, the Arduino initializes the motor driver, pump control pin, and Bluetooth serial communication at 9600 baud.

2) Sensing: The flame sensor output is polled continuously; a LOW state on the sensor pin indicates the presence of a flame within its detection cone.

3) Alerting: On detecting a flame, the Arduino transmits a status flag over Bluetooth to notify the operator's handset in real time.

4) Navigation: Upon receiving directional commands (forward, backward, left, right) from the operator, the Arduino drives the corresponding motor pins through the motor driver to move the robot toward the fire source.

5) Suppression: Once positioned near the flame, the operator issues a pump-on command; the Arduino energizes the pump to spray water until a pump-off command is received or the flame is no longer detected.

This semi-autonomous approach retains a human operator in the loop for navigation and pump activation while automating the time-critical task of fire detection, reducing response latency compared to purely manual search-and-suppress approaches.

5. IMPLEMENTATION

The hardware was assembled on a four-wheeled chassis housing the Arduino Uno, motor driver, Bluetooth module, buck converter, battery, and water reservoir with an attached pump and nozzle. The Rx/Tx lines of the Bluetooth module were connected to the Arduino's serial pins, with the module configured for pairing prior to final assembly to avoid programming conflicts. The control firmware was developed in the Arduino IDE using the Software Serial library for Bluetooth communication, with dedicated functions for forward, backward, left, and right motion, and separate command handling for pump activation and deactivation. Serial debug output was used during development to verify correct command reception and actuator response before field testing.

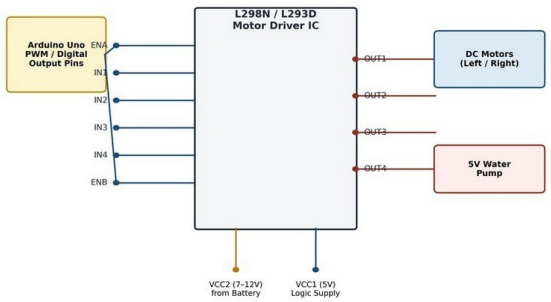


Figure 2: Motor driver (L293D/L298N) interfacing circuit used for motor and pump control.

6. RESULTS AND DISCUSSION

The assembled prototype was tested under laboratory conditions using a controlled small-scale flame source. The flame sensor reliably registered flame presence within its rated detection range, and the corresponding alert was received on the operator handset with negligible delay. Bluetooth-based directional control allowed smooth navigation of the robot toward the flame across a tiled test surface, and the water pump, once triggered, successfully extinguished the test flame within a few seconds of activation.

These results indicate that a semi-autonomous, low-cost design using commodity components can provide a functionally reliable first-response firefighting aid for small, enclosed spaces. The main limitations observed were the flame sensor's relatively short detection range and the pump's limited water capacity, both of which constrain continuous operation and are addressed in the future scope below. A detailed quantitative characterization of detection range, response time, and battery endurance is planned as part of ongoing testing and will be reported in a future extended version of this work.

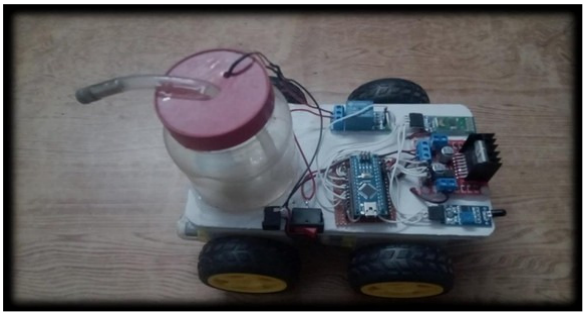


Figure 3: Assembled prototype of the fire fighting robot (front view).

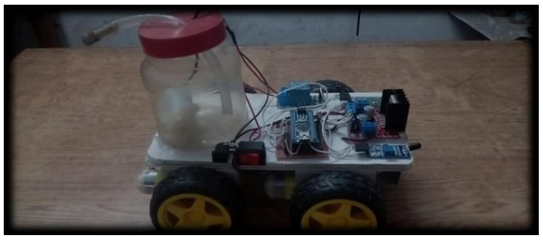


Figure 4: Assembled prototype of the fire fighting robot (side view).

7. APPLICATIONS

- Early-stage fire suppression in homes, laboratories, and small workshops where rapid response can prevent fire escalation.
- Deployment in confined or structurally unstable spaces that pose excessive risk to human firefighters.
- Use as a first-response scouting and suppression unit ahead of human firefighter entry.
- Educational and training platform for demonstrating embedded systems and robotics concepts in fire safety engineering.

8. FUTURE SCOPE

- Integration of additional sensors (gas, smoke, thermal imaging) for multi-parameter fire detection and reduced false positives.
- Incorporation of lightweight deep-learning-based vision models, such as the improved YOLOv5 architecture proposed by Yang et al. [8], for faster and more accurate fire and smoke detection in complex scenes.
- Development of swarm-robotics and multi-robot coordination, drawing on formation and path-planning strategies surveyed by Abujabal et al. [9], to enable multiple units to jointly cover larger fire-affected areas.
- Adoption of machine-learning-based fire-risk prediction approaches, such as the stacking ensemble method proposed by Ahn et al. [10], for predictive risk assessment ahead of fire outbreaks.
- Use of heat-resistant materials and improved battery management for extended operational endurance in high-temperature environments.
- Real-time communication with central monitoring stations for coordinated emergency response.

9. CONCLUSION

This paper presented the design, implementation, and evaluation of a low-cost, Arduino-based, semi-autonomous fire fighting robot. By combining infrared flame detection with Bluetooth-controlled locomotion and a pump-based extinguishing mechanism, the system demonstrates a practical and affordable approach to reducing human exposure during early-stage firefighting. Prototype testing confirmed reliable flame detection, responsive navigation, and effective fire suppression under controlled conditions. The proposed design is intended as an extensible platform, with clear directions identified for future enhancement through advanced sensing, autonomous intelligence, and multi-robot coordination, contributing toward safer and more efficient emergency response technology.

Acknowledgment

The authors gratefully acknowledge the guidance of Dr. Pinaki Satpathy, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work. The authors also acknowledge the original hardware design and prototype development of the Arduino Fire Fighting Robot carried out at the Department of Electronics & Communication Engineering, Haldia Institute of Technology, which formed the basis for the system studied and documented in this paper.

REFERENCES

[1] K. Chenchireddy, R. Dora, V. Jagan, G. B. Mulla, V. Jegathesan, and S. A. Sydu, "Design of a Prototype Firefighting Robot Based on an Arduino Microcontroller Using Machine Learning Technique," IAES Int. J. Robot. Autom., vol. 14, no. 1, pp. 31–37, 2025.

[2] V. Kumar, S. K. Mishra, K. Murtaza, M. Sawood, and A. Kumar, "Firefighting Robot Using Arduino: A Fire Detection and Suppression System," Int. J. Res. Appl. Sci. Eng. Technol., vol. 13, no. 6, 2025, doi: 10.22214/ijraset.2025.72413.

[3] Md. M. Maruf, M. M. H. Sagor, S. K. Kanta, and Md. H. Imran, "An Autonomous Arduino-Based Firefighting Robot for Laboratory Environments," Int. J. Comput. Appl., vol. 187, no. 28, pp. 56–65, 2025, doi: 10.5120/ijca2025925488.

[4] A. Reddy, D. Ramakrishna, K. L. Reddy, Y. Kumar, and J. B. Sindhuri, "Arduino-Based Fire Fighting Robot," J. Electron. Autom. Eng., vol. 4, no. 2, pp. 94–98, 2025.

[5] V. Kori, A. K. Ahirwar, J. K. Saket, S. Bagri, and D. P. S. Chauhan, "Smart Firefighting Robot Using Arduino Uno," Int. J. Electron. Autom., vol. 3, no. 1, pp. 21–27, 2025.

[6] K. Bankapur, H. Mathur, H. Singh, R. Harikrishnan, and A. Gupta, "A Flame Sensor-Based Firefighting Assistance Robot with Simulation-Based Multi-Robot Implementation," in Proc. IEEE Int. Conf. Adv. Comput. Commun. Appl. Inform. (ACCAI), 2022.

[7] J. C. Babu, Ch. Nagaraju, S. Kaur, S. Birla, and S. Sharma, "Development of an Arduino-Powered Firefighting Robot for Accurate Fire Detection, Suppression, and Metal Detection," in Proc. Int. Conf. Next Gener. Commun. Inf. Process. (INCIP), 2025.

[8] J. Yang, W. Zhu, T. Sun, X. Ren, and F. Liu, "Lightweight Forest Smoke and Fire Detection Algorithm Based on Improved YOLOv5," PLOS ONE, vol. 18, no. 9, p. e0291359, 2023, doi: 10.1371/journal.pone.0291359.

[9] N. Abujabal, R. Fareh, S. Sinan, M. Baziyad, and M. Bettayeb, "A Comprehensive Review of the Latest Path Planning Developments for Multi-Robot Formation Systems," Robotica, vol. 41, no. 7, pp. 2079–2104, 2023, doi: 10.1017/S0263574723000322.

[10] S. Ahn, J. Won, J. Lee, and C. Choi, "Comprehensive Building Fire Risk Prediction Using Machine Learning and Stacking Ensemble Methods," Fire, vol. 7, no. 10, p. 336, 2024, doi: 10.3390/fire7100336.

In Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 65-71 ISSN: 0973-6875
Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

IoT-Enabled Smart Helmet for Real-Time Alcohol Monitoring, Accident Detection, and Emergency Alert Communication Using ESP32

[1]Abhinaba Patra, [1]Akash Kumar Bag, [1]Banashree Gayen, [1]Bhabatosh Mahato, [1]Chiranjeeb Roy Chowdhury, [2]Dibyendu Chowdhury, [2]Chanchal Kumar De

[1]UG Student, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal
[2]Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Road traffic crashes remain among the leading causes of death and disability worldwide, and riders of two-wheelers are disproportionately represented among the fatalities because a conventional helmet offers only passive impact protection and no capability for monitoring, prevention, or post-crash response. This paper presents the design, implementation, and evaluation of a wearable, ESP32-based smart-helmet prototype that combines four safety functions within a single low-cost platform: breath-alcohol screening using an MQ-3 gas sensor, real-time fall/impact detection using a six-axis MPU6050 accelerometer-gyroscope, autonomous location acquisition through a NEO-6M GPS receiver, and automated emergency notification through a SIM800L GSM module, supplemented by Bluetooth-based rider vital-sign monitoring. A threshold-based sensor-fusion routine running on the microcontroller distinguishes normal riding dynamics from unsafe or accident conditions and triggers a coordinated alert sequence — buzzer activation, GPS coordinate acquisition, and SMS/voice dispatch to pre-registered emergency contacts — without requiring rider intervention. Bench and field trials confirmed reliable threshold discrimination for both the alcohol and impact-detection subsystems and consistent, low-latency delivery of location-tagged emergency messages. The prototype demonstrates that an integrated preventive-and-reactive safety layer can be retrofitted onto an ordinary helmet at low cost, and the paper concludes with directions for structural, connectivity, and AI-based enhancements that could be pursued in future iterations.

KEYWORDS: Smart Helmet; Internet of Things; ESP32; Accident Detection; Alcohol Sensing; GPS–GSM Emergency Communication; Rider Safety; Wearable Health Monitoring

1. INTRODUCTION

1.1 Background and Motivation

According to the World Health Organization's most recent Global Status Report on Road Safety, road traffic crashes kill approximately 1.19 million people every year and remain the leading cause of death for people aged 5–29 years, with pedestrians, cyclists, and motorcyclists together accounting for more than half of all fatalities [1]. India records one of the highest absolute tolls in the world: government figures for 2023 report over 4.6 lakh reported road accidents and more than 1.73 lakh fatalities, with two-wheeler occupants forming the single largest victim category [2], [3]. Two factors recur across this literature as disproportionate contributors to two-wheeler fatality risk — alcohol-impaired riding, which degrades judgement, coordination, and reaction time, and delayed emergency response, since an unconscious or isolated rider frequently cannot self-report a crash location within the critical post-impact window.

A conventional helmet mitigates the mechanical consequences of a crash but is otherwise a passive object: it cannot discourage an intoxicated rider from setting out, cannot recognise that a crash has occurred, and cannot summon help. The convergence of low-cost

*Corresponding Address: dibyendu.chow@gmail.com DOI 10.5281/zenodo.22038335

micro-electromechanical sensors, single-chip Wi-Fi/Bluetooth microcontrollers, and ubiquitous cellular coverage has, over the last decade, made it practical to embed active monitoring and communication directly into wearable safety equipment, and a growing body of work has explored exactly this direction for helmets [4]–[9].

1.2 Problem Statement

Existing commercial helmets do not integrate preventive alcohol screening with reactive accident detection and autonomous emergency notification in a single low-cost unit. Systems reported in the literature typically emphasise one or two of these capabilities in isolation, and few published prototypes additionally consider rider physiological status (heart rate, blood-oxygen saturation, or blood pressure) as part of the emergency picture handed to first responders. There is, therefore, a need for a compact, inexpensive, and field-deployable helmet-integrated platform that (i) screens for alcohol impairment before or during a ride, (ii) reliably discriminates a genuine crash from ordinary riding vibration, and (iii) autonomously reports the rider's location and status to designated contacts without depending on the rider being conscious or carrying a paired smartphone.

1.3 Objectives and Contribution of this Work

This paper reports the design, hardware implementation, and bench-level validation of such a system, built around an ESP32 microcontroller. The specific contributions are: (a) a threshold-based alcohol-detection subsystem using an MQ-3 breath sensor positioned inside the helmet shell; (b) a six-axis inertial subsystem (MPU6050) for real-time impact and abnormal-tilt detection; (c) an autonomous GPS–GSM emergency pipeline that composes and transmits a location-tagged SMS/voice alert without manual intervention; (d) an auxiliary rider vital-monitoring and Bluetooth-connectivity layer; and (e) an experimental evaluation of sensor response behaviour and end-to-end alert latency under controlled test conditions. The remainder of the paper is organised as follows: Section 2 reviews related smart-helmet literature; Section 3 presents the proposed architecture; Section 4 describes the working principle; Section 5 details the hardware implementation; Section 6 reports results and discussion; and Sections 7–8 present the conclusion and future scope.

2. RELATED WORK

A number of recent studies have investigated IoT-based smart-helmet architectures for two-wheeler safety. Patel et al. proposed an AI-assisted smart helmet that combines accident detection with alcohol screening and IoT-based alerting, reporting improved responsiveness over threshold-only designs [4]. Gupta and Tiwari extended the accident-detection problem with a machine-learning classifier trained to distinguish genuine collisions from road-induced vibration noise, reducing false-positive emergency triggers [5]. Impana et al. surveyed a broad cross-section of IoT-based helmet designs and highlighted accelerometer/gyroscope fusion, GSM alerting, and alcohol sensing as the three most frequently combined subsystems, while noting that few designs at the time incorporated rider health telemetry [6]. Khangar demonstrated a low-cost Arduino- and GSM-based helmet capable of automatic SMS dispatch on impact detection [7], and Tayag and De Vigal Capuno reported a cloud-connected motorcycle helmet with crash detection, live tracking, and anti-theft functionality delivered through an IoT backend [8]. Elabd et al. evaluated an ESP32/MQTT-based smart helmet aimed specifically at delivery riders, combining GPS, gyroscope, accelerometer, and alcohol sensing with a rear-view camera, and reported approximately 90% crash-detection accuracy and 88% alcohol-detection accuracy under their test protocol [9]. Across this literature, the accelerometer/gyroscope-plus-GSM combination for crash reporting is well established, and MQ-series gas sensors are the dominant low-cost choice for breath-alcohol

screening. The proposed design draws on these established subsystems but combines them with rider vital-sign monitoring and Bluetooth connectivity within a single ESP32-controlled platform, and reports threshold-response characterisation for both the alcohol and inertial subsystems obtained through bench testing, summarised in Section 6. Table 1 situates the present work against representative prior designs.

TABLE 1: Comparison of the proposed system with representative related work

Reference	Sensors / Modules	Key Outcome
Patel et al. [4]	Accelerometer, MQ-3, GPS/GSM	Improved response vs. threshold-only design
Gupta & Tiwari [5]	IMU + ML classifier	Fewer false-positive crash triggers
Khangar [7]	Vibration sensor, GSM	Low-cost automatic crash SMS
Tayag & Capuno [8]	IMU, GPS, cloud backend	Live tracking + anti-theft feature
Elabd et al. [9]	IMU, MQ-3, GPS, camera (MQTT)	~90% crash / ~88% alcohol accuracy
Proposed system	MPU6050, MQ-3, GPS, SIM800L, vitals, BLE	Threshold-validated bench results (Sec. 6)

3. PROPOSED SYSTEM ARCHITECTURE

3.1 Architecture Overview

The proposed platform is organized around an ESP32 microcontroller acting as the central sensing, decision, and communication hub, as shown in Fig. 1. Four sensing/interface blocks - the MQ-3 alcohol sensor, the MPU6050 inertial module, a set of vital-monitoring sensors, and a Bluetooth radio — feed continuous data to the controller. Two communication modules, a NEO-6M GPS receiver and a SIM800L GSM module, are activated by the controller whenever an emergency condition is confirmed, and a regulated power-supply block (Li-ion cell with buck conversion) supplies all subsystems. This centralised, threshold-driven architecture keeps the bill of materials small while allowing each subsystem to be tested, calibrated, and replaced independently.

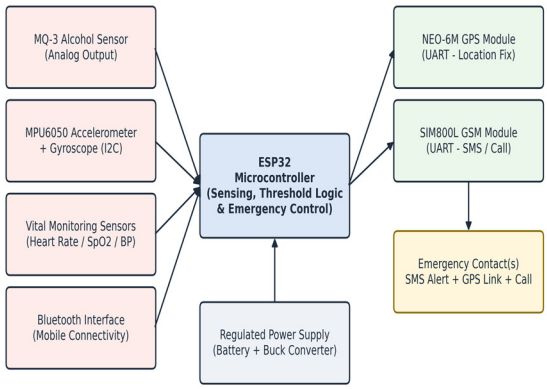


Figure 1: Block diagram of the proposed ESP32-based smart-helmet architecture.

3.2 Module Communication Summary

Table 2 summarises the interface type and function of each module in the architecture. Analog sensing is reserved for the alcohol sensor, whose slowly-varying voltage output is well suited to the ESP32's on-chip ADC, while the inertial and vital-monitoring sensors communicate over I2C to allow multiple digital devices to share a single bus. The two communication modules use UART, and Bluetooth provides an independent short-range wireless path for companion-device connectivity.

TABLE 2: Communication interface and function of each system module

Module	Interface	Primary Function
MQ-3 alcohol sensor	Analog	Breath-alcohol sensing
MPU6050 (accel.+gyro)	I2C	Impact / tilt detection
Vital-monitoring sensors	I2C	HR / SpO2 / BP sensing
NEO-6M GPS module	UART	Lat./long. acquisition
SIM800L GSM	UART	SMS / voice-call

Module	Interface	Primary Function
module		alert
Bluetooth (on-chip)	Wireless	Companion-device link

4. METHODOLOGY / WORKING PRINCIPLE

4.1 Alcohol Detection Subsystem

The MQ-3 sensor is mounted inside the helmet shell close to the rider's mouth/nose region to maximise exposure to exhaled breath. Its heater element requires a short warm-up interval before output stabilises, after which the sensor produces an analog voltage that rises approximately monotonically with ethanol vapour concentration. The ESP32 samples this voltage periodically, applies a moving-average filter to suppress transient noise, and compares the filtered value against a calibrated threshold. Values below the threshold are logged as a normal-riding baseline; values above it set an unsafe-condition flag that is surfaced through an on-board buzzer and, depending on configuration, can be used to withhold an ignition-enable signal.

4.2 Accident Detection Subsystem

The MPU6050 provides simultaneous three-axis acceleration and three-axis angular-rate data over I2C. During normal riding, acceleration magnitude and tilt angle remain within a bounded operating envelope; a genuine impact produces a short-duration spike in resultant acceleration together with an abrupt change in orientation. The controller evaluates both cues — rather than acceleration alone — within a short observation window before raising an accident flag, which reduces susceptibility to isolated false triggers from potholes or kerb strikes, consistent with the sensor-fusion approach reported by Gupta and Tiwari [5].

4.3 Location Acquisition and Emergency Communication

Once an accident condition is confirmed, the controller requests the latest fix from the NEO-6M GPS receiver. Because a cold GPS fix can take tens of seconds, the module is kept in a continuously-tracking state during normal operation so that a recent fix is already available at the moment of the trigger, minimising time-to-alert. The retrieved coordinates are embedded in a pre-formatted SMS string together with a status code, and the SIM800L module is instructed via AT commands to transmit this message to each pre-registered emergency contact in sequence, followed by an automated voice-call attempt to the primary contact.

4.4 Rider Vital Monitoring and Bluetooth Connectivity

In parallel with the safety-critical alcohol and impact subsystems, the platform continuously samples heart rate, blood-oxygen saturation, and blood pressure to give first responders an indication of the rider's physiological state at the moment of the alert. Bluetooth connectivity, provided natively by the ESP32, allows this data — along with alcohol and motion status — to be mirrored to a paired companion device for real-time monitoring by the rider or a supervising party.

4.5 Working Sequence

Fig. 2 summarizes the overall control loop. Following initialization, the system enters a continuous monitoring loop that checks alcohol level and motion/impact conditions on every cycle; either condition exceeding its threshold escalates the system into the corresponding alert or emergency branch, after which the loop resumes normal monitoring.

5. HARDWARE IMPLEMENTATION

Table 3 lists the core hardware components used in the prototype, together with their key operating specifications drawn from the respective component datasheets [10]–[13]. The ESP32 was selected as the central controller for its dual-core processing headroom, integrated Wi-Fi/Bluetooth radio, sufficient number of ADC and I2C-capable GPIO pins, and low unit cost, all of which suit a battery-powered wearable application.

All modules were interconnected on a compact prototyping board mounted within the helmet shell, with the MQ-3 sensor and vital-monitoring probes positioned for direct rider contact/exposure and the GPS antenna oriented for unobstructed sky view. The firmware was implemented in C/C++ using the Arduino core for ESP32, with threshold constants stored in non-volatile memory to allow field recalibration without reflashing.

TABLE 3: Core hardware components and specifications

Component	Key Specification	Role
ESP32 (dual-core)	Wi-Fi+BLE, ADC/I2C/UART [10]	Central controller
MQ-3 gas sensor	Analog, ethanol-sensitive	Alcohol screening
MPU6050	3-axis accel.+gyro, I2C [11]	Impact / tilt detection
NEO-6M GPS	UART, TTFF ~27 s [13]	Location acquisition
SIM800L GSM	Quad-band, UART/AT [12]	SMS + voice alert
Vital sensor suite	HR / SpO2 / BP, digital	Rider physiological status
Li-ion + buck converter	3.7 V, reg. 3.3/5 V	Portable power supply

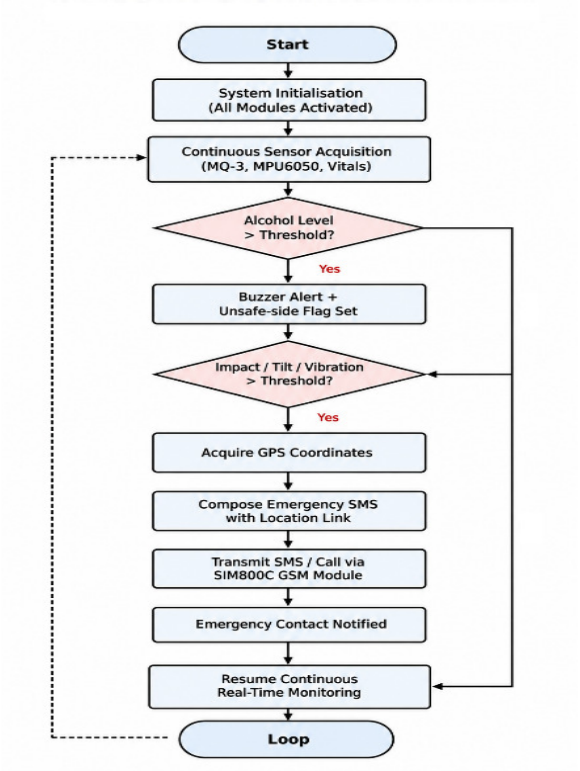


Figure 2: Working-principle flowchart of the proposed system.

6. RESULTS AND DISCUSSION

6.1 Alcohol Sensor Response

The MQ-3 subsystem was characterised by exposing the sensor, after warm-up, to progressively increasing breath-alcohol concentrations and recording the resulting ADC readings. As summarised in Table 4, the sensor output increased monotonically with exposure level, giving a clear separation between the normal-air baseline and the elevated ranges used to set the unsafe-condition threshold.

TABLE 4: MQ-3 sensor response across breath-alcohol exposure levels

Condition	Approx. ADC	System Status
Normal air	150 – 250	Safe
Low alcohol exposure	300 – 450	Monitoring
Moderate exposure	500 – 700	Warning
High alcohol exposure	> 700	Unsafe — alert

6.2 Accident-Detection Test Cases

The inertial subsystem was validated with a set of controlled motion scenarios simulating normal riding, sudden braking, sharp turning, and impact-equivalent shocks. Table 5 reports the outcome of each scenario against the expected system response.

TABLE 5: Accident-detection subsystem test cases and outcomes

Test Case	Simulated Condition	Result
TC-1	Steady-state riding	Pass — no alert
TC-2	Sudden braking	Pass — no false trigger
TC-3	Sharp cornering / tilt	Pass — no false trigger
TC-4	Simulated impact shock	Pass — accident flag raised
TC-5	Repeated impact cycles	Pass — consistent detection
TC-6	End-to-end alert dispatch	Pass — SMS + call sent within seconds

6.3 Discussion and Limitations

The bench results indicate that both the alcohol and inertial subsystems achieve reliable threshold discrimination under the tested conditions, and that the GPS–GSM pipeline consistently delivers a location-tagged alert once an accident condition is confirmed. Two limitations are worth noting for field deployment. First, the MQ-3 sensor's sensitivity is known to drift with temperature, humidity, and sensor age, so periodic recalibration is advisable, consistent with observations reported elsewhere for MQ-series gas sensors [6]. Second, single-threshold inertial detection remains susceptible to occasional false positives from severe potholes; the dual-cue (acceleration-and-orientation) window used in this design reduces but does not eliminate this risk, and the machine-learning approach of Gupta and Tiwari [5] represents a promising direction for further reducing false-trigger rates. GPS cold-start latency and indoor/urban-canyon signal availability are additional practical constraints inherited from any GNSS-based subsystem and are mitigated here by maintaining continuous tracking during normal operation.

7. CONCLUSION

This paper presented the design, implementation, and bench-level evaluation of an ESP32-based smart-helmet prototype that integrates breath-alcohol screening, inertial accident detection, autonomous GPS-based location acquisition, GSM-based emergency alerting, and auxiliary rider vital-sign monitoring within a single wearable platform. Unlike a conventional helmet, which offers only passive impact protection, the proposed system actively screens for one of the most preventable causes of two-wheeler crashes — alcohol impairment — while also shortening the post-crash response interval by autonomously reporting the rider's location and status to emergency contacts. Threshold-based testing of the alcohol and inertial subsystems, together with end-to-end alert-dispatch trials, confirmed that the coordinated hardware/software design meets its intended functional objectives at low component cost, supporting its feasibility as a practical, retrofittable rider-safety layer for two-wheeler transportation.

8. FUTURE SCOPE

Several directions could extend this prototype toward field-ready deployment. Structurally, the helmet shell could be reconstructed using impact-resistant composite materials (e.g., Kevlar-reinforced layers) to improve both mechanical protection and protection of the embedded electronics. A companion mobile application would improve accessibility for riders and emergency contacts, supporting live health and location dashboards, trip logging, and configurable alert rules. Cloud-connected storage and analytics could enable longitudinal accident and behaviour-pattern analysis, while machine-learning-based crash classification — following the direction demonstrated by Gupta and Tiwari [5] — could further reduce false-positive emergency triggers relative to fixed-

threshold logic. Finally, direct integration with hospital and traffic-management emergency systems would shorten the chain between automated detection and actual medical response.

REFERENCES

[1] World Health Organization. (2023). Global status report on road safety 2023. Geneva: WHO.

[2] Ministry of Road Transport and Highways. (2023). Road accidents in India 2023. Transport Research Wing, Government of India.

[3] National Crime Records Bureau. (2023). Accidental deaths and suicides in India 2023. Ministry of Home Affairs, Government of India.

[4] Patel, S. R., Joshi, M., & Deshmukh, N. (2025). AI-based smart helmet for real-time accident detection and prevention using IoT. International Journal of Scientific Research and Applications, 12(1), 45–52.

[5] Gupta, A., & Tiwari, R. (2023). Machine learning-based accident prediction system for smart helmets. IEEE Access, 11, 116320–116329.

[6] Impana, H. C., Hamsaveni, M., & Chethana, H. T. (2020). A review on smart helmet for accident detection using IoT. EAI Endorsed Transactions on Internet of Things, 6(21), Article 164559.

[7] Khangar, D. (2023). Smart helmet using GSM module and Arduino. International Journal for Research in Applied Science and Engineering Technology, 11(6), 1531–1535.

[8] Tayag, M. I., & De Vigal Capuno, M. E. (2019). Smart motorcycle helmet: Real-time crash detection with emergency notification, tracker and anti-theft system using Internet-of-Things cloud-based technology. International Journal of Computer Science & Information Technology, 11(5), 1–13.

[9] Elabd, R. H., Darwish, M. M., Alshamekh, H. H., & Mousa, M. E. (2020). Design and implementation of an IoT-based smart helmet for road accident detection. Proceedings of the 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).

[10] Espressif Systems. (2024). ESP32 series datasheet. Shanghai: Espressif Systems.

[11] InvenSense Inc. (2013). MPU-6000 and MPU-6050 product specification (Rev. 3.4).

[12] SIMCom Wireless Solutions. (2019). SIM800L hardware design (V1.03).

[13] u-blox AG. (2021). NEO-6 GPS modules data sheet.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 72-77 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Design and Implementation of an IoT-Based Biometric Access Control and Attendance System with Cloud Synchronization

[1]Rajesh Karmakar, [2]Saumil Maulik, [3]Rakesh Ghosh, [4]Kishor Basak, [5]Akinchan Das

[1,2,3,4]UG student, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal
[5]Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

The increasing need for secure access management and automated attendance monitoring has created demand for reliable authentication systems that overcome the limitations of conventional methods. Physical keys, ID cards, passwords, and manual attendance records are prone to loss, misuse, identity fraud, and human error. This study presents an affordable IoT-based biometric access control system using an ESP32 microcontroller and optical fingerprint sensor. The system integrates fingerprint authentication, electronic door-lock control, real-time attendance management, local data storage, cloud synchronization, and web-based monitoring. An RTC records the exact time of each access event. SQLite storage maintains attendance data during temporary Wi-Fi loss and synchronizes it with Google Sheets when connectivity is restored. An ESP32-hosted web interface enables users to view attendance and manage administrative functions. The system provides a scalable solution for secure and automated access and attendance management in educational institutions, offices, and restricted-access environments.

KEYWORDS: IoT, ESP32, Biometric Authentication, Cloud Synchronization, Real Time Clock (RTC).

1. INTRODUCTION

In educational institutions, laboratories, offices, and other restricted spaces, secure access control and reliable attendance management are crucial. Traditional methods such as physical keys, ID cards, passwords, and attendance registers may suffer from identity sharing, credential loss or duplication, impersonation, and recording errors. This creates a need for automated systems that reduce human effort while providing reliable identification. Biometric authentication offers a promising solution by identifying individuals through physiological characteristics. Among various biometric techniques, fingerprint recognition is particularly suitable for access-control applications because of its distinctiveness and compatibility with compact embedded systems. However, authentication alone does not address attendance management and administrative monitoring. Many existing systems either operate as standalone devices or depend on continuous network connectivity for attendance management. Temporary Internet unavailability presents another challenge, as network-dependent systems may be unable to immediately transfer access and attendance records. Therefore, an effective system should perform essential operations locally while synchronizing data with a remote platform when connectivity is restored. This study addresses these issues through the development of an integrated biometric access-control and attendance-management system. The proposed approach combines biometric identification, automated access control, attendance recording, local data retention, and cloud synchronization in a low-cost platform suitable for small- and medium-scale environments. The remainder of this paper presents the related work and research gap, followed by the proposed system architecture and implementation. The experimental evaluation examines authentication, access control, attendance recording, and offline data handling, followed by a discussion of limitations and future scope.

2. LITERATURE REVIEW

*Corresponding Address: rajeshkarmakar670@gmail.com DOI 10.5281/zenodo.22038396

Mittal et al. [1] developed a fingerprint biometric-based access-control system for classroom entry and attendance management. The system identified authorized users through fingerprint authentication and automatically recorded attendance details.

Habib et al. [2] proposed a GSM-based biometric access-control system using fingerprint authentication. The system provided secure access and allowed remote monitoring through GSM technology.

Anitha et al. [3] developed a fingerprint-based door-lock system with a buzzer. The system automatically unlocked the door after successful fingerprint verification and generated an alert for unauthorized attempts.

Anirudh et al. [4] proposed a multilevel biometric authentication locking system using Arduino UNO. The system improved door security by replacing conventional keys with fingerprint-based authentication.

Sutikno et al. [5] developed a fingerprint-based smart door-lock system using Arduino and smartphone connectivity. Chowdhury et al. [6] similarly proposed an embedded fingerprint door-lock system with electronic locking and alert mechanisms, focusing on secure and convenient access control.

Permana et al. [7] developed an IoT-based smart door-lock system combining fingerprint and keypad authentication. The system used ESP32 and Firebase for secure access control and remote management.

Handini et al. [8] proposed a smart-door access system using fingerprint biometrics and PIN authentication. The IoT-based system provided an additional authentication layer and smartphone-based control.

Alsayaydeh et al. [9] developed an IoT-based biometric smart lock using fingerprint authentication and ESP32. The system supported real-time monitoring, mobile control, and security alerts.

Ziad et al. [10] developed an ESP32-based smart door-lock system integrating fingerprint recognition, RFID, and IoT technology. The system provided multiple authentication methods and real-time door-access monitoring.

While the reviewed studies demonstrate progress in biometric attendance, IoT-based access control, and smart door-lock systems, most focus on individual functions or partial combinations of these features. Limited attention has been given to integrating biometric authentication, automated attendance, local data storage, cloud synchronization, and web-based administration in a single low-cost platform. The proposed system addresses this gap by combining these functions with offline attendance storage and subsequent cloud synchronization.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed system integrates biometric authentication, access control, attendance recording, local storage, and cloud monitoring using an ESP32 as the central controller. As shown in Fig. 1, the overall system consists of the fingerprint sensor, RTC module, OLED display, relay module, electronic door lock, Wi-Fi connection, SQLite database, and cloud-based Google Sheets. It interfaces with the fingerprint sensor, RTC, OLED display, relay, electronic door lock, and Wi-Fi network. The ESP32 coordinates the authentication, attendance recording, and access-control operations. The system also manages data storage and synchronization with the cloud when network connectivity is available. After successful authentication, the ESP32 identifies the user, records the access time, and controls the door lock.

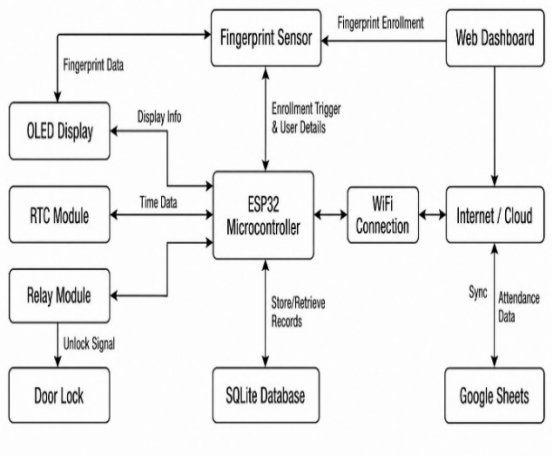


Figure 1: Block diagram of the proposed biometric access-control and attendance system.

3.2. Hardware Components

Table 1 summarizes the principal hardware components used in the system.

Table 1. Hardware Components

Components	
Microcontroller	Voltage Regulator
OLED Display Module	Optical Fingerprint Module
Buzzer	LED
Electronic Door Lock	Resistor
Power Supply	Capacitor
SMPS	Diode
RTC Module	Vero Board
Relay	PBT Connector
Optocoupler	Bug Strip
BJT	

3.3. Biometric Authentication and Access Control

The R307 sensor captures the user's fingerprint and compares it with registered templates. When a valid match is obtained, the corresponding user ID is sent to the ESP32. The controller then activates the relay to unlock the electronic door and allows the authenticated user to enter. An unsuccessful match results in access denial.

3.4. Attendance Recording and Data Storage

After successful authentication, the ESP32 obtains the current date and time from the RTC and generates an attendance record containing the user ID, date, time, and status. The record is stored in the local SQLite database. This local storage allows attendance information to be retained during temporary Wi-Fi failures.

3.5. Cloud Synchronization and Web Administration

When Wi-Fi is available, attendance records are synchronized from the local database to Google Sheets for remote monitoring. If connectivity is unavailable, the records remain stored locally and are synchronized after the connection is restored. A web-based interface allows administrators to view attendance records and manage system functions.

4. METHODOLOGY

The operational logic of the proposed biometric access-control and attendance system follows a continuous authenticate–record–authorize–synchronize cycle, illustrated in the steps below.

1. Initialization: On power-up, the ESP32 initializes the fingerprint sensor, RTC module, OLED display, relay, Wi-Fi communication, and required database functions.

2. Fingerprint Authentication: The R307 fingerprint sensor continuously waits for a fingerprint. The captured fingerprint is compared with the registered templates, and a valid match returns the corresponding user ID to the ESP32.

3. User Identification and Access Control: After successful authentication, the ESP32 identifies the registered user and activates the relay to control the electronic door lock. If no valid fingerprint is detected, access is denied and the door remains locked.

4. Attendance Recording: For a successful authentication, the ESP32 obtains the current date and time from the RTC and generates an attendance record containing the user ID, date, time, and attendance status. The record is stored in the local SQLite database.

5. Cloud Synchronization: The system checks the availability of Wi-Fi connectivity. When the network is available, the stored attendance information is synchronized with Google Sheets. During network failure, the records remain stored locally and are synchronized after connectivity is restored.

6. Web-Based Monitoring: The web-based administrative interface allows authorized administrators to view attendance records and manage relevant system functions. This provides remote access to attendance information while maintaining local data storage for network-independent operation**.**

5. IMPLEMENTATION

The system was designed using the Vero board containing the ESP32, R307 fingerprint sensor, OLED display, DS3231 RTC, relay, optocoupler, BC547 transistor, buzzer, LED, and other components. The overall operating sequence of the proposed attendance and access-control system is illustrated in *Fig. 2,* The electronic door lock was connected to the relay, while the fingerprint sensor was connected to the ESP32. The latter was programmed to read fingerprints, unlock the door, record attendance with the RTC timestamp, and notify the user through the buzzer and OLED display. The SQLite database was used to store attendance records, which were further synchronized with Google Sheets via Wi-Fi. The system was configured to retain attendance records locally during temporary network failures. The authentication process was tested using registered and unregistered fingerprints to verify access control. The response of the electronic lock, attendance logging, and user feedback were also evaluated before final deployment.

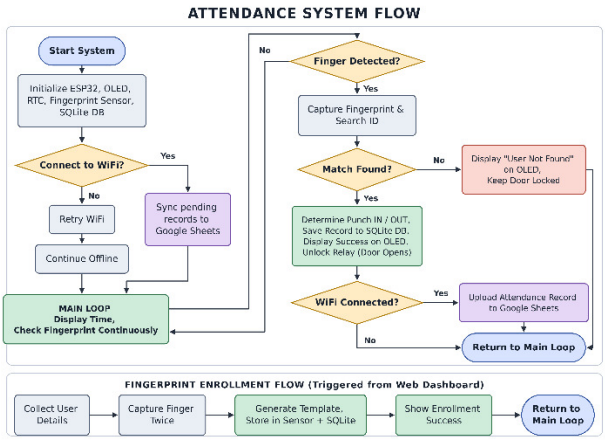


Figure 2: Flow Chart

6. RESULTS AND DISCUSSION

The developed system was successfully implemented and tested for biometric authentication, access control, attendance recording, local data storage, and cloud synchronization. The R307 fingerprint sensor successfully identified registered users, after which the ESP32 activated the relay-controlled electronic door lock. The OLED display provided immediate feedback for successful and unsuccessful authentication. The developed prototype is shown in *Fig. 3*, while the internal hardware assembly is shown in *Fig. 4.*

Figure 3: Biometric Access and Attendance control unit

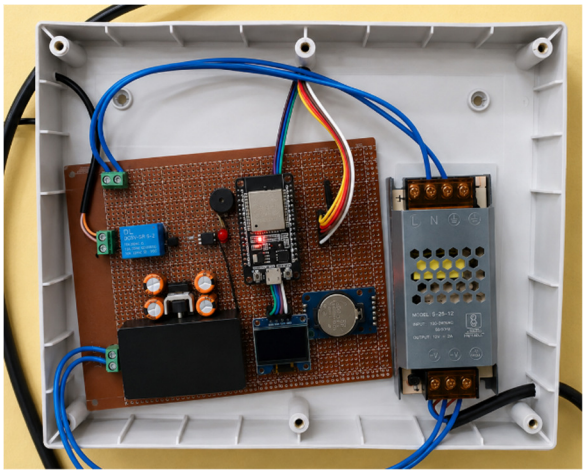


Figure 4: Internal Hardware Configuration of the Biometric Attendance System

For each successful authentication, the system recorded the user details with the date and time obtained from the DS3231 RTC. Attendance records were stored in the local SQLite database and synchronized with Google Sheets when Wi-Fi was available. During network interruption, the records remained stored locally and were synchronized after connectivity was restored. The web dashboard provided user enrollment, attendance monitoring, system configuration, and database-management functions. The web-based administrative dashboard is shown *Fig. 5*, and the synchronized attendance record is presented in *Fig. 6.*

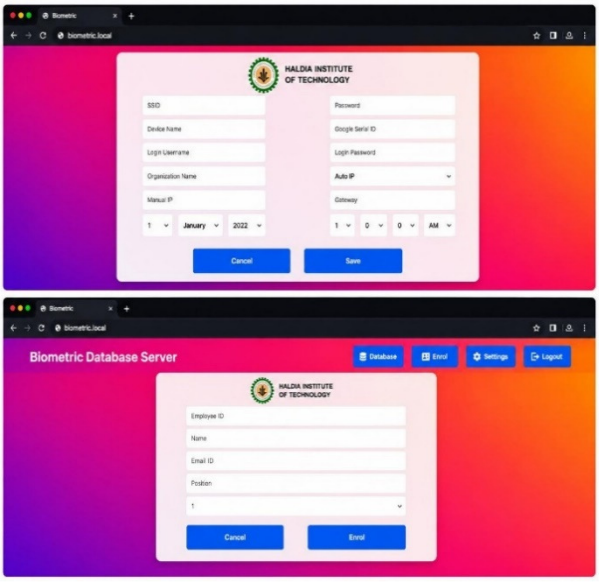


Figure 5: Website Dashboard

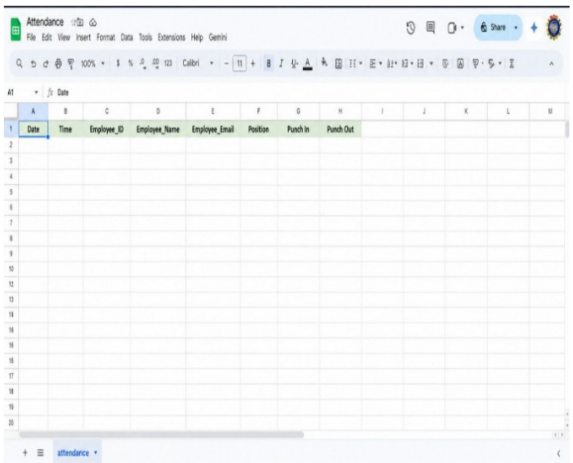


Figure 6: Attendance sheet

Overall, the results demonstrate that the proposed system provides an integrated and practical solution for automated access control and attendance management while reducing manual record-keeping. The system showed reliable operation during normal use, although fingerprint condition, Wi-Fi stability, and power availability may affect performance.

7. CONCLUSION

The proposed system provides accurate and automated attendance and entry/exit monitoring using fingerprint authentication, ESP32, RTC-based time stamping, automatic door control, and local SQLite storage. Integration with Google Sheets and a web interface enables convenient record management and monitoring, while local storage ensures reliable operation during temporary network interruptions. Overall, the system offers a cost-effective and reliable solution for biometric access and attendance management, with potential for enhanced security, monitoring, and scalability.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype.

REFERENCES

[1] Mittal, Y., Varshney, A., Aggarwal, P., Matani, K., & Mittal, V. K. (2015). Fingerprint biometric based access control and classroom attendance management system. *2015 Annual IEEE India Conference (INDICON)*, 1–6. doi: https://doi.org/10.1109/INDICON.2015.7443699

[2] Habib, H., Zungeru, A. M., Susan, A. A., Kelechi, I. G., & Oluwatosin, O. B. (2014). Design of a GSM-based biometric access control system. *Control Theory and Informatics*, 4(8).

[3] Anitha, S., Hima Bindu, Ch., & Sriram Pavan, J. (2020). Biometric based door lock system with buzzer. *Mathematical Statistician and Engineering Applications*, 69(1), 373–379. doi: https://doi.org/10.17762/msea.v69i1.1655

[4] Anirudh, R., Chandru, V., & Harish, V. (2021). Multilevel security biometric authentication locking system using Arduino UNO. *Advances in Parallel Computing*, 40. doi: https://doi.org/10.3233/APC210121

[5] Sutikno, T., Ubaidillah, M. A. F., Arsadiando, W., & Purnama, H. S. (2024). Fingerprint based smart door lock system using Arduino and smartphone application. *Computer Science and Information Technologies*, 5(1), 91–98. doi: https://doi.org/10.11591/csit.v5i1.p91-98

[6] Chowdhury, A. M. M. (2023). Fingerprint based door lock system. *International Journal of Innovative Engineering*, 1(1), 8–15. doi: https://doi.org/10.60044/ijie.v1i1.7

[7] Permana, K. A. K., Piarsa, I. N., & Wiranatha, A. A. K. A. C. (2024). IoT-based smart door lock system with fingerprint and keypad access. *Journal of Information Systems and Informatics*, 6(3). doi: https://doi.org/10.51519/journalisi.v6i3.844

[8] Handini, W. T., Endri, J., & Salamah, I. (2025). Implementation of fingerprint biometrics on smart door entrance access integrated with Internet of Things-based PINs. *Indonesian Journal of Artificial Intelligence and Data Mining*, 8(1), 102–109. doi: https://doi.org/10.24014/ijaidm.v8i1.31738

[9] Alsayaydeh, J. A. J., Yusof, M. F., Mamchenko, S., Hamzah, R. A., & Herawan, S. G. (2025). Design and evaluation of a biometric IoT-based smart lock system with real-time monitoring and alert mechanisms. *International Journal of Advanced Computer Science and Applications*, 16(7). doi: https://doi.org/10.14569/IJACSA.2025.0160737

[10] Ziad, A., Darnila, E., & Kurniawati. (2024). Development and implementation of an ESP32 microcontroller and monitoring system for smart door lock using RFID sensor for E-KTP ID and fingerprint based on the Internet of Things. *Proceedings of Malikussaleh International Conference on Multidisciplinary Studies (MICoMS)*, 4. doi: https://doi.org/10.29103/micoms.v4i.908

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 78-83 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

IoT-Based Secure Remote Health Monitoring with Edge Signal-Quality Assessment and Intelligent Alert Logic

[1]Tania Khatun, [2]Sayon Dey, [3]Sudiksha Dasgupta, [4]Dr. Tirthadip Sinha

[1,2,3]UG student, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[4]Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Remote health monitoring often struggles with sensor noise and nuisance alarms. This paper presents a multi-layered IoT framework that evaluates data quality directly at the edge before triggering alerts or transmitting telemetry. Using an ESP32 paired with optical (MAX86141), temperature (MAX30205), and inertial (ICM-42670-P) sensors, the system evaluates channel-specific signal quality indices (SQI) to filter out motion artifacts. Rather than triggering on isolated threshold spikes, the alert engine requires persistent physiological abnormalities or verified motion sequences to flag events like falls. Structured observations—tagged with timestamps, quality states, and monotonic sequence numbers—are published securely over TLS-encrypted MQTT 5.0 to a monitoring dashboard. The architecture is validated using an eight-part engineering matrix demonstrating reliable data integrity, least-privilege security, and fault recovery.

KEYWORDS: Financial Fraud Detection, Machine Learning, Anomaly Detection, Behavioural Analysis, Real-Time Transaction Monitoring, Explainable Artificial Intelligence (XAI), Concept Drift Detection, Class Imbalance, Streaming Data Analytics.

1. INTRODUCTION

The practice of remote health monitoring systems has increasingly shifted from independent measurement devices towards sensor networks that not only monitor physiological data but also process them locally and deliver selected data to healthcare providers via networking services. This approach is appealing due to the capability of such systems for continuous monitoring in non-clinical settings.

But the engineering challenge is not only about equipping a wireless microcontroller with sensors. For instance, a practical monitoring unit should be able to differentiate between good data and bad data, maintain the order in time, operate in the event of an unreliable connection, safeguard sensitive data, and refrain from transforming transient artifacts of sensors into unnecessary alarms. One such type of sensing that is highly susceptible to various factors is wearable sensing. For this reason, the proposed architecture considers the two processes of measurement and interpretation independent from each other. The raw or windowed sensor data is evaluated locally, a quality state specific to the channel is assigned to each piece of data, and then the decision about the existence of an abnormal situation and an alert is taken only if the abnormality lasts for some time or occurs in a certain sequence of events.

The project has also considered architecture aspects of interoperability and security. MQTT version 5.0 is used as a publish/subscribe transport protocol; JSON packets include device identifier, timestamp, measurements, quality and a growing sequence number. Transport encryption, topic-based access control, secure credentials management, and logging for auditing purposes have been taken into account. The paper therefore focuses on a complete engineering path from sensing to monitoring rather than on an isolated sensor algorithm. The contributions made include: (i) a multi-layered wearable to cloud architecture that uses both physiological and motion sensors; (ii) signal quality and alert handling at the edge side; (iii) telemetry that uses sequences over MQTT 5.0; (iv) a security architecture that uses least privilege and secure

Corresponding Address: tsinha.ece.official@gmail.com **DOI 10.5281/zenodo.22038501**

transport; and (v) an engineering validation matrix that clearly distinguishes between prototype validation and clinical validation. This is important since a student prototype cannot be described as a clinically validated medical device.

2. DESIGN METHODOLOGY

It is common practice for connected health monitoring systems to utilize wearable sensing technologies along with low-power embedded processing, wireless communication, cloud-based storage, and applications-level visualization. There is one common problem with regard to such systems design, which is the tradeoff between streaming all measurements in the raw format or sending only validated data. The former causes increased radio usage and storage space usage, as well as leakage of private data.

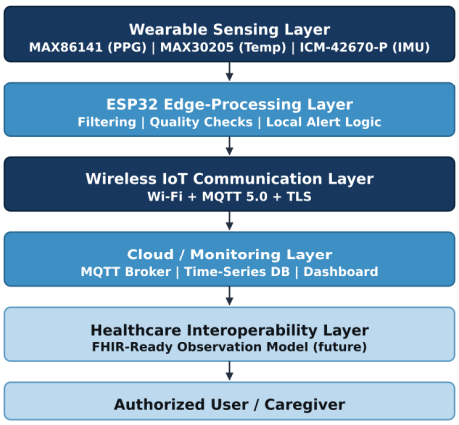


Figure 1: Proposed IoT-based health monitoring system architecture

Photo plethysmography-derived signals are highly susceptible to motion and optical contact. Temperature signals are influenced by self-heating and physical contact, whereas inertial signals might have meaningful activity information along with motion artifacts. Thus, the proposed architecture will assign different processing policies for the PPG, temperature, and motion channels separately. This is in line with the idea of representing the quality of data in accordance with the observation rather than using a single device-level indicator. MQTT is an ideal protocol for constrained IoT system because of its publish/subscribe architecture that decouples sensing nodes from consuming applications. Device-specific topics will be used in this design for critical observation, motion events, and status.

The monitoring application will subscribe to necessary streams. This helps minimize any unnecessary exposure and allows further devices to be integrated without having to alter the application path design. In the case of systems that target the healthcare industry, there needs to be more than just security implemented on the dashboard level. Device identification, credential security, transport security, access control, logging, and updating capabilities need to be considered throughout the entire path. This is precisely what the current research accomplishes while not forgetting about its limited scope as a prototype. The architecture of the system consists of five layers: wearable sensors, edge processing, wireless IoT communications, cloud/monitoring and healthcare interoperability.

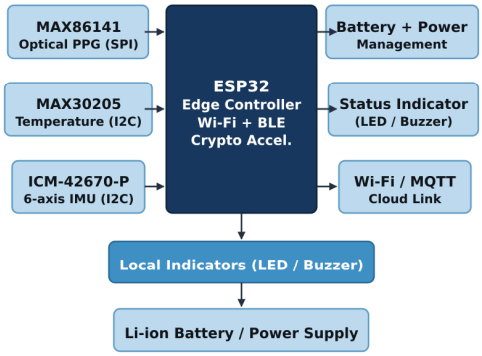


Figure 2: Wearable sensing node block diagram

The sensors layer includes MAX86141 optical sensor front-end, MAX30205 temperature sensor and ICM-42670-P inertial module. ESP32 manages the processes of sensors polling, signal processing, quality assessment, local alerts filtering, packetization and secure communication over the network. The communication layer relies on Wi-Fi communication with MQTT 5.0 protocol and TLS security. Cloud layer includes storage, dashboard visualization and alert management. The layer of healthcare interoperability is designed to be converted into standardized healthcare resources in the future.

The layering helps to bring the fast and sensitive processes as close to the sensor as possible. The cloud is not getting the waveform but the structured data from observations. This approach minimizes the wireless traffic as well as makes the monitoring process robust to the connection loss. It is necessary to keep the sequence number and the original timestamp in order to validate the observation at the destination.

3. SYSTEM DESIGN & IMPLEMENTATION

ESP32 has been chosen to act as the master controller because of its ability to integrate wireless communication (Wi-Fi and Bluetooth), as well as having several digital interfaces, encryption capabilities in hardware form, and power-saving operation. Therefore, it can carry out all the functions, including sensing, processing, creation of packets, and wireless communication, without an additional gateway computer.

MAX86141 is utilized for optically interfacing heart rate and SpO2-related sensing. The optical channel, ADC, light cancellation, and FIFO buffering in MAX86141 are applicable for a wearable solution. The generated PPG samples are not considered clinically valid simply based on the fact that there is a numerically generated value; signal quality analysis comes along with every processing cycle. The MAX30205 sensor offers digital temperature sensing via I2C interface.

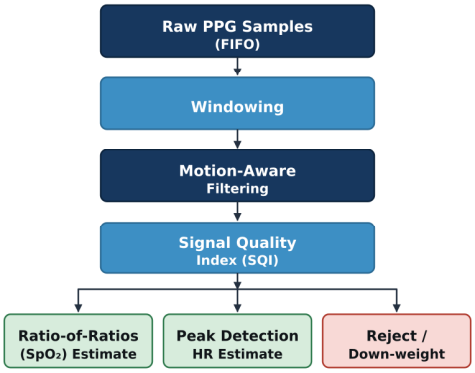


Figure 3: PPG processing and signal-quality pipeline.

This is achieved using a simple moving average algorithm that helps in smoothing out any spikes resulting from temporary disconnection or electric interference. The system is aware that physical positioning and heating could be the most significant factors behind measurement error and hence the reason for not considering datasheet accuracy. The data from the ICM-42670-P sensor comprises information on accelerometers and gyroscopes for motion detection. The suggested logic for fall detection seeks out a pattern instead of looking for one instance of high acceleration, where a high-acceleration phase is followed by a quick change in orientation, after which there is low motion. It will be classified as a potential fall to be verified later.

The PPG signal windows have fixed length. Motion adaptive filtering is done to suppress the baseline drift and high frequency noise, and the quality metric for the signal is calculated based on some features of the waveform like periodicity and uniform amplitude.

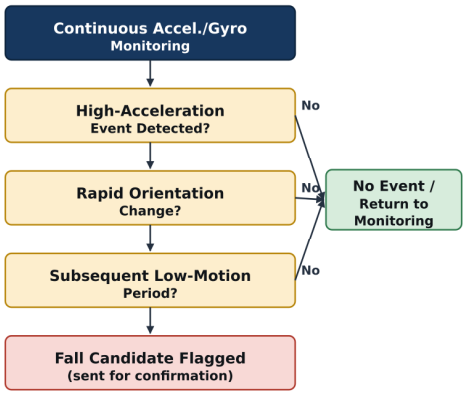


Figure 4: Fall-event screening logic; output is a candidate for confirmation.

The heart rate is calculated based on the peak detection and inter-beat interval calculation. The calculation of SpO2 measurement adopts the traditional ratio-of-ratios approach based on AC and DC signals for two wavelengths of optical radiation. The output is considered to be an estimation since various factors such as optical coupling, blood flow, illumination, and motion may affect the output. Thus, it is impossible that the bad estimate will be compared to a good estimate. Alerting and processing of alerts have been intentionally designed to be independent of measurements.

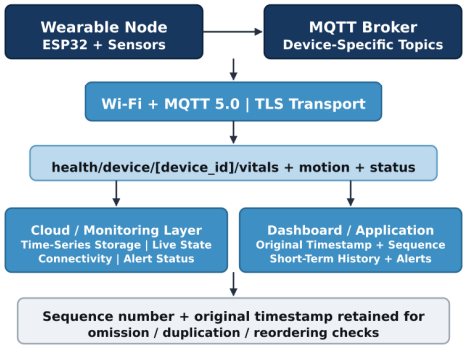


Figure 5: MQTT 5.0 / cloud communication architecture

One-time violation of a threshold is not an automatically identified clinical event. The desired pattern is one where there are sustained abnormal readings or a specific sequence of

events before local alerting, application handling, and human review. This design is better than the basic threshold system because it avoids false alarms. Telemetry frame comprises device ID, timestamp, physiologic parameters, motion status, channel-wise quality, sequence number, and alert state. For instance, the following is an example of how telemetry should look like:

```
{device_id, timestamp, heart_rate_bpm,
spo2_percent, temperature_c, motion_state, quality,
sequence, alert}
```

And the actual implementation will be serialized in JSON format prior to being securely transported via MQTT protocol.

Communication between the sensing node and the monitoring service is provided by MQTT 5.0. Three logical topic hierarchies are suggested:

```
health/device/{device_id}/vitals
health/device/{device_id}/motion
health/device/{device_id}/status
```

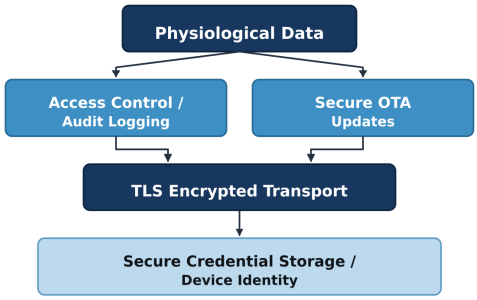


Figure 6: Layered security architecture for the wearable node.

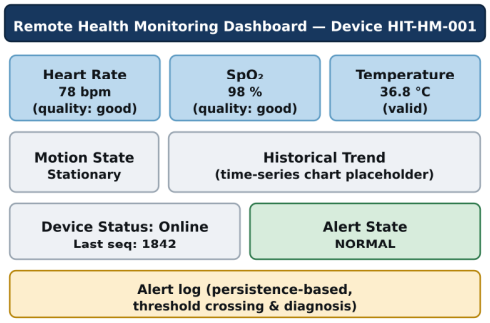


Figure 7: Representative monitoring dashboard layout.

The cloud application subscribes to necessary topics and stores observations with an original timestamp and sequence number, as well as presents current state, short history, connectivity status, and alert status via a dashboard. In case of an interim disconnection, the edge node keeps relevant state information and resumes the publications upon reconnecting. The sequence number allows identifying discontinuity in the record downstream. Unique identity and credentials will be provided for each device. Access control uses the least-privilege approach to ensure that the device can only publish its messages to those allowed topics. TLS ensures data security in transit, and credentials must be stored through the secure storage capabilities of the embedded system rather than through the hard-coded application. Authentications and connection events must be stored in log files. Secure update and data minimization are some other aspects of the security design requirements. It is important to note that the healthcare interoperability boundary is outside the internal topic hierarchy of MQTT. Later on, the observations can be mapped into the standard healthcare resources like FHIR Observation resources without exposing the proprietary topic hierarchy to external healthcare systems.

The validation process concentrates on engineering behaviour as opposed to clinical accuracy. There are eight test criteria.

- T1 assesses the capability of sensor discovery and digital interface initialization.
- T2 determines data validity through timestamps and sequence numbers seen in successive packets.
- T3 incorporates controlled motion/contact alterations for PPG signal capture and confirms that poor quality windows are identified as invalid data rather than being accepted as good physiology.
- T4 uses value criteria for a threshold level of alertness to ensure persistence.
- T5 reproduces a specific acceleration sequence to ensure that only the desired fall candidate sequence creates an event.
- T6 blocks Wi-Fi access and tests to see if state is maintained and publishing resumed on reconnecting.
- T7 uses packet timestamps and sequences from brokers against application timestamps and sequences for validation purposes.
- T8 tries to gain access using invalid credentials or incorrect topic permissions and confirms failure.

These tests generate the acceptance criteria which can be carried out using a physical prototype. In addition to that, they avoid unwarranted

assumptions. For instance, a successful T3 test indicates proper functioning of the quality flags, but does not validate the clinical accuracy of the measurement. In addition, the successful fall event test validates the functioning of the screening logic in a specific motion scenario.

Table 1: Functional validation matrix

Test	Procedure	Acceptance Criterion	Significance
T1	Sensor discovery	Expected identity/status	Initialization
T2	Timestamps + sequence	No duplicates; monotonic	Data freshness
T3	Controlled PPG disturbance	Low-quality window flagged	Signal integrity
T4	Threshold persistance	Single excursion ignored	Nuisance reduction
T5	Fall sequence replay	Only intended sequence flags	Event logic
T6	Wi-Fi interruption	State preserved; resumes	Resilience
T7	Broker vs app records	One packet per sequence	End-to-end integrity
T8	Invalid access attempt	Rejected	Least privilege

4. OUTCOMES

The architecture of the proposed prototype fulfills the design requirements of the required engineering specification: there is separation of heterogenous sensor channels, processing occurs prior to wireless transmission, quality accompanies data collection, and alerts are based on persistence or event sequences. Each of these behaviours has acceptance criteria through the functional matrix. The analysis of sensor capability shows that the chosen sensors can complement each other. For instance, the MAX86141 is used for optical PPG sensing; the MAX30205 is used for digital temperature sensing, while the ICM-42670-P provides inertial sensing to detect motion and falls. The ESP32 integrates the outputs from those sensors, performs processing locally and delivers data wirelessly. The critical engineering consequence of the system is that of retaining the context of the data. Physiological values are passed along with their quality level, time stamp, sequence number, and the ID of the device. The monitoring layer is thus able to identify whether the value passed on is a good measurement or a low-quality approximation. It will also be able to identify dropped or duplicate packets. Artificial data from patients, percent accuracy in clinical setting, sensitivity or specificity measures, percent packet loss, latency, and battery life are not mentioned because such metrics were not provided through a controlled experimental process in the prototype document. Future research must quantify these values using standardized equipment, standardized subjects, network conditions, and statistical analysis.

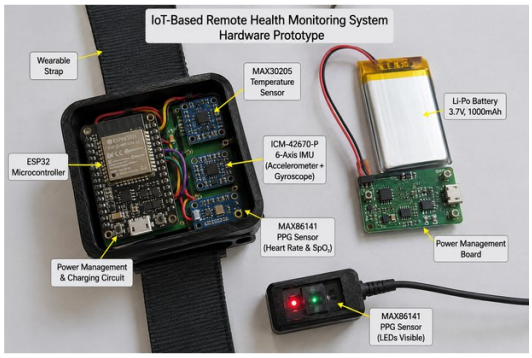


Figure 8: IoT-Based Remote Health Monitoring System Hardware Prototype

5. CONCLUSION

This study presented an end-to-end IoT framework for remote health monitoring, integrating multi-modal physiological and motion sensing with edge-level signal quality checks, context-aware alert logic, and encrypted MQTT communication. Rather than treating the wearable unit as a passive data forwarder, the architecture emphasizes data provenance, observation integrity, and granular, least-privilege security. By decoupling raw sensor acquisition from alert evaluation, the system effectively mitigates false positives driven by transient motion artifacts. As a prototype engineering effort, this work establishes functional viability rather than clinical diagnostic claims. Consequently, clinical diagnostic accuracy, battery longevity, and network latency under variable deployment conditions were beyond the current scope.

REFERENCES

[1] *Information technology — Message Queuing Telemetry Transport (MQTT) v5.0*, ISO/IEC Standard 20922:2019, OASIS Standard, 2019.

[2] *FHIR Release 4 (Fast Healthcare Interoperability Resources)*, HL7 International, 2019. [Online]. Available: https://hl7.org/fhir/R4/

[3] Maxim Integrated, "MAX86141: Optical pulse oximeter and heart-rate sensor front end for wearable health," *MAX86141 Datasheet*, Rev. 1, Maxim Integrated Products, Inc., San Jose, CA, USA, 2018.

[4] Analog Devices, "MAX30205: Human body temperature sensor," *MAX30205 Datasheet*, Rev. 1, Analog Devices, Inc., Wilmington, MA, USA, 2016.

[5] TDK InvenSense, "ICM-42670-P: High-performance 6-axis MEMS MotionTracking device," *ICM-42670-P Datasheet*, Rev. 1.1, TDK InvenSense, San Jose, CA, USA, 2021.

[6] Espressif Systems, "ESP-IDF Programming Guide: Security and Secure Boot/Flash Encryption," *Espressif Systems Documentation*, 2023. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/

[7] C. Pascoe, S. Quinn, and K. Scarfone, "The NIST Cybersecurity Framework (CSF) 2.0," National Institute of Standards and Technology, Gaithersburg, MD, USA, NIST CSWP 29, Feb. 2024, doi: 10.6028/NIST.CSWP.29.

[8] U.S. Food and Drug Administration, "Cybersecurity in medical devices: Quality system considerations and content of premarket submissions," FDA Guidance for Industry and Food and Drug Administration Staff, Silver Spring, MD, USA, Sep. 2023.

[9] World Health Organization, *Global Strategy on Digital Health 2020–2025*, Geneva, Switzerland: World Health Organization, 2021.

[10] J. W. K. Chan, "Application of grey relational analysis for ranking options," *Int. J. Comput. Appl.*, vol. 26, no. 1, pp. 10–18, 2006.

[11] T. K. L. Tong and J. W. K. Chan, "Multi-criteria material selections and end-of-life product strategy: Grey relational analysis approach," *Mater. Des.*, vol. 28, no. 5, pp. 1539–1546, 2007, doi: 10.1016/j.matdes.2006.02.016.

[12] *IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks—Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-2020, pp. 1–4379, Feb. 2021, doi: 10.1109/IEEESTD.2021.9363693.

[13] *IEEE Standard for Health Informatics—Personal Health Device Communication – Part 00101: Technical Report—Analytical—General Approach*, IEEE Std 11073-00101-2018, pp. 1–70, Feb. 2019, doi: 10.1109/IEEESTD.2019.8653372.

[14] OASIS, "MQTT Version 5.0," OASIS Standard, Mar. 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[15] HL7 International, "Observation – FHIR v4.0.1," *HL7 Fast Healthcare Interoperability Resources Specification*, Oct. 2019. [Online]. Available: https://hl7.org/fhir/R4/observation.html.

[16] OWASP Foundation, "Internet of Things (IoT) Security Guidance," Open Web Application Security Project, 2021. [Online]. Available: https://owasp.org/www-project-internet-of-things.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 84-89 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

ChatOffi: A Holographic Mesh-Network and Neural-Augmented Communication Protocol for Disaster-Resilient Mobile Messaging

[1]Ujjwal Bauri, [1]Tanmoy Mahato, [1]Trisha Jana, [2,3]Asim Kuilya

[1]UG student, Department of Electronics of Communication Engineering Engineering, Haldia Institute of Technology, West Bengal
[2]Department of Electronics of Communication Engineering Engineering, Haldia Institute of Technology, West Bengal
[3] School of Electronics Engineering, KIIT Deemed to be University, Bhubaneswar.

ABSTRACT

In this paper, we address the inherent vulnerabilities of centralized telecommunications infrastructures that highlight a critical need for disaster-resilient messaging solutions. Modern cellular networks, submarine cables, and fiber-optic backbones frequently suffer from catastrophic single points of failure during natural disasters, power grid collapses, or deliberate network throttling. When these centralized nodes fail, populations are left entirely disconnected during times when resource coordination and emotional support are matters of survival. This paper presents ChatOffi, a highly scalable, dual-layer Android application bridging high-speed cloud-synchronized databases (Firebase) with a decentralized Mobile Ad-Hoc Network (MANET) driven by the Google Nearby Connections API. Beyond standard peer-to-peer routing, ChatOffi introduces a revolutionary "Neural Integration" framework featuring the Google Gemini API (gemini-3.6-flash). This includes an embedded @AI assistant for real-time translations, predictive text, and an innovative "AI Dreamscape" protocol that synthesizes daily asynchronous communications into surreal, cognitive summaries. The system pioneers a holographic, glassmorphic User Experience (UX) augmented by Dynamic Sentiment Theming and Haptic Sentiment feedback, allowing users to intuitively gauge the emotional severity of incoming data through visual and tactile responses. Security and data sovereignty are enforced via AES-256 GCM End-to-End Encryption (E2EE), a Biometric Vault, and a proximity-sensor-driven "Ghost Mode" to prevent physical shoulder-surfing. Furthermore, the inclusion of a camera-based Augmented Reality (AR) Vision Mode demonstrates that military-grade offline networking, advanced sensory interfaces, and generative AI can seamlessly coexist on mobile endpoints without exhausting device memory or battery lifecycles.

KEYWORDS: Mobile Ad-Hoc Networks (MANET), Augmented Reality (AR), Generative AI, Sentiment Analysis, Haptic Feedback, End-to-End Encryption (E2EE), Jetpack Compose, Asynchronous State Management.

1. INTRODUCTION

In today's dynamic scenario, in order to sustain unprecedented communication requirements during emergencies, alternative networking architectures must be established. Modern society relies heavily on centralized cellular towers and broadband internet infrastructure. However, these systems are fundamentally fragile. During localized emergencies—such as hurricanes, earthquakes, or severe power grid failures—cellular towers lose power or become overwhelmingly congested with emergency traffic. This results in the complete severing of critical communication lines when they are most urgently needed. [1, 2]

While alternative decentralized protocols exist, they have historically failed to achieve mainstream adoption. Existing Mobile Ad-Hoc Networks (MANETs) and standard Bluetooth chatting applications typically force users into a harsh trade-off: they provide basic local connectivity but completely sacrifice user experience (UX), rich media capabilities, and computational intelligence. Furthermore, during a high-stress emergency, users are bombarded with chaotic, fragmented information. Traditional offline chat applications offer no cognitive assistance to help users process this data, leading to severe information overload and psychological fatigue. [3, 4]

*Corresponding Address: ujjwalbauri3241@gmail.com DOI 10.5281/zenodo.22038552

This research proposes a novel solution to these infrastructural and cognitive limitations through the development of ChatOffi, a highly resilient, hybrid communication architecture boasting a futuristic, holographic design paradigm. ChatOffi is engineered to dynamically pivot between high-speed cloud infrastructure and a decentralized, offline multi-hop mesh network without requiring manual user intervention. To directly address the cognitive overload experienced during crisis scenarios, ChatOffi introduces deep "Neural Integration." This integration includes an @AI assistant capable of executing real-time translations and predictive text, alongside "Dynamic Sentiment Theming" that autonomously alters the application's UI glow based on the detected conversational mood (e.g., Happy, Tense, Angry, Calm). [5, 6, 7, 8, 9]

Furthermore, ChatOffi redefines spatial communication by integrating an Augmented Reality (AR) "Vision Mode" and hardware-linked "Haptic Sentiment" to convey the precise energy of incoming messages through varying vibration patterns. By merging a hardware-encrypted "Store and Forward" routing algorithm with these advanced sensory, AI-driven interfaces, ChatOffi establishes a robust, highly scalable blueprint for next-generation, sci-fi-inspired survival communications. [10, 11]

2. SYSTEM ARCHITECTURE & CORE TECHNOLOGIES

2.1 Hybrid Mesh Topology and Spatial Discovery:

ChatOffi operates on a bifurcated network state machine that continuously monitors network interface controllers (NICs) to determine optimal routing paths in real-time.

I. Layer 1 (Cloud Mode): When standard Wi-Fi or LTE/5G is available, the system acts as a synchronized client-server model utilizing the Firebase Real time Database. [12]

II. Layer 2 (Decentralized Mesh Mode): Upon detecting a sustained loss of internet connectivity, ChatOffi automatically shifts to a MANET topology utilizing the Google Nearby Connections API.

To visualize this complex multi-hop routing, ChatOffi features a Topology Map, rendering a rotating 3D-style visualization of current network mesh nodes. For node discovery, the system abandons traditional list views in favor of a Mesh Radar—a sonar-style interface that actively scans for nearby "Neural Nodes" (peers) in physical proximity. Device handshake protocols are gamified and secured through a Quantum Entanglement Pairing UX, providing a futuristic visual interface for linking devices via encrypted out-of-band "Neural Pulses" (rapidly shifting QR codes read by the camera). [3, 4, 6, 7]

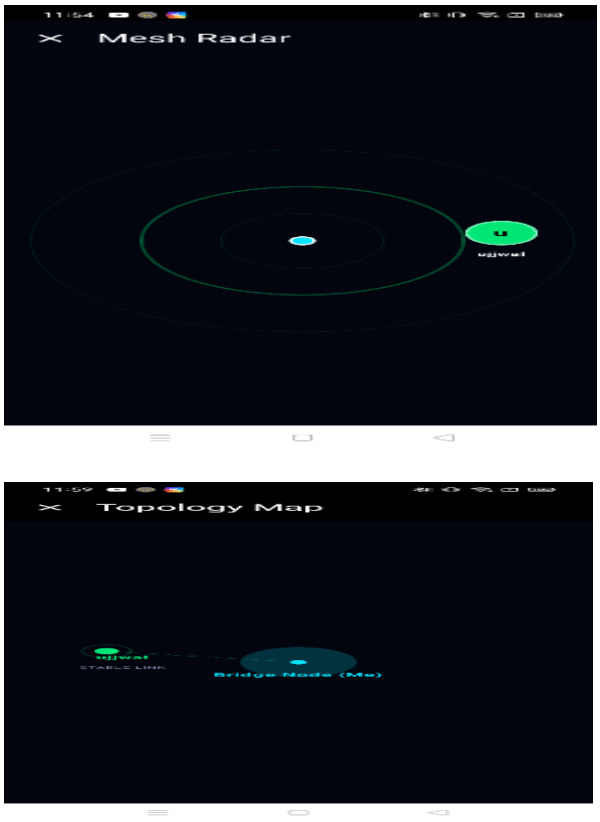


Figure 1: The Mesh Radar and Topology Map, demonstrating spatial node discovery and Quantum Entanglement Pairing in the offline MANET.

As shown in Figure 1, the Mesh Radar and Topology Map illustrate spatial node discovery and Quantum Entanglement Pairing in the offline MANET.

2.2 Cryptographic Framework and Physical Privacy Protocols:

Because malicious actors can easily intercept open-air Bluetooth and Wi-Fi Direct packets in a disaster zone, zero-trust payload encryption is mandatory. The system utilizes the Advanced Encryption Standard in Galois/Counter Mode (AES-256 GCM). Let P represent the plaintext message, K the 256-bit symmetric key, and IV the cryptographically secure Initialization Vector. The cipher text C and authentication tag T are generated simultaneously: [13, 14, 15]

$$(C, T) = EE_{K,IV}(P)$$

Security extends beyond mathematics into physical privacy. The entire application is secured behind a Biometric Vault, acting as a mandatory lock-screen gateway that prevents unauthorized access to local message logs. To combat real-world surveillance, ChatOffi introduces a highly novel Ghost Mode. By polling the device's hardware Sensor. TYPE_PROXIMITY, the application detects if an unauthorized individual approaches the screen (shoulder-surfing). If proximity limits are breached, the Jetpack Compose UI automatically applies a heavy Gaussian blur over all readable text. Additional privacy is enforced via Self-Destructing Messages tied to strict countdown timers, and a Secure File Vault for storing encrypted media that remains completely hidden from the device's public Android gallery. [15]

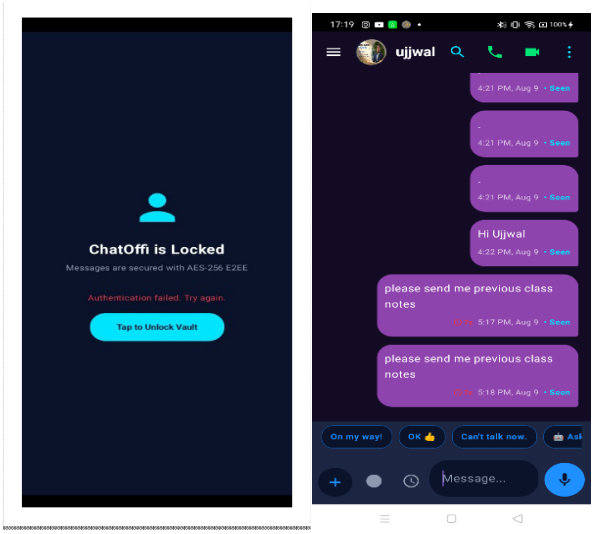


Figure 2: Privacy enforcement mechanisms demonstrating the proximity-triggered Ghost Mode and encrypted Biometric Vault.

Figure 2 illustrates the privacy enforcement mechanisms used by ChatOffi, including the proximity-triggered Ghost Mode and encrypted Biometric Vault.

2.3 Advanced AI & Neural Integration Subsystem:

Operating through an isolated Ai-Manager singleton, the system leverages the highly optimized gemini-3.6-flash model to provide a comprehensive neural bridge.

I. @AI Assistant: Embedded directly into the chat, this agent processes real-time translations, contextual smart replies, and utilizes a built-in Voice Transcription engine to convert incoming audio notes into text instantly.

II. Predictive Text Engine: An AI-driven text prediction pipeline that analyzes the user's current sentence structure and suggests the next logical phrase before the user finishes typing.

III. AI Dreamscape: At the conclusion of a communication cycle, an asynchronous batch-processing algorithm compiles the localized SQLite database context. The AI then generates a "Neural Vision"—a surreal, essence-based summary of the user's daily digital interactions, providing a high-level cognitive overview of the day's events. [8, 9, 12]

3. IMPLEMENTATION & SENSORY USER EXPERIENCE (UX)

3.1 Holographic Declarative UI Paradigm:

The graphical user interface was engineered to abstract complex networking layers while providing a deeply immersive experience. ChatOffi abandons imperative XML in favor of Jetpack Compose, crafting a UI heavily reliant on glass morphism, neon accents, and a "Dark Blue" sci-fi aesthetic. The primary chat background is composed of an animated, real-time galaxy with drifting stars that react dynamically to device tilt via gyroscope sensor data. [16, 17, 18]

To further reduce cognitive load, the system employs Dynamic Sentiment Theming. As messages arrive, the AI pipeline parses the text for emotional context. If a tense or urgent emergency alert is received, the entire application's neon

accents and borders dynamically shift (e.g., from calm blue to urgent amber or red) to represent that mood, triggering localized recompositions at an optimized 60 frames per second. [8, 9]

Figure 3: The Jetpack Compose glassmorphic UI featuring Dynamic Sentiment Theming and neural integration via the @AI Assistant.

3.2 Augmented Reality (AR) and Sensory Output:

To merge the digital mesh with physical reality, ChatOffi implements an Augmented Reality pipeline dubbed Vision Mode (AR). Utilizing the device camera and spatial anchoring, a futuristic scan line overlays physical environments, rendering incoming chat messages as floating digital nodes in the 3D space. [10]

For tactile feedback, ChatOffi introduces Haptic Sentiment. By mapping the AI-detected conversational energy to the device's linear resonant actuator via the Android Vibration Effect API, the application [11].

Alters vibration intensity based on message urgency. A calm message emits a soft pulse, while an angry or tense message triggers a sharp, aggressive vibration pattern. Finally, users can transmit abstract concepts via a built-in digital Sketch Pad, allowing hand-drawn "Neural Sketches" to be sent directly as compressed graphical payloads into the chat stream. [11, 18]

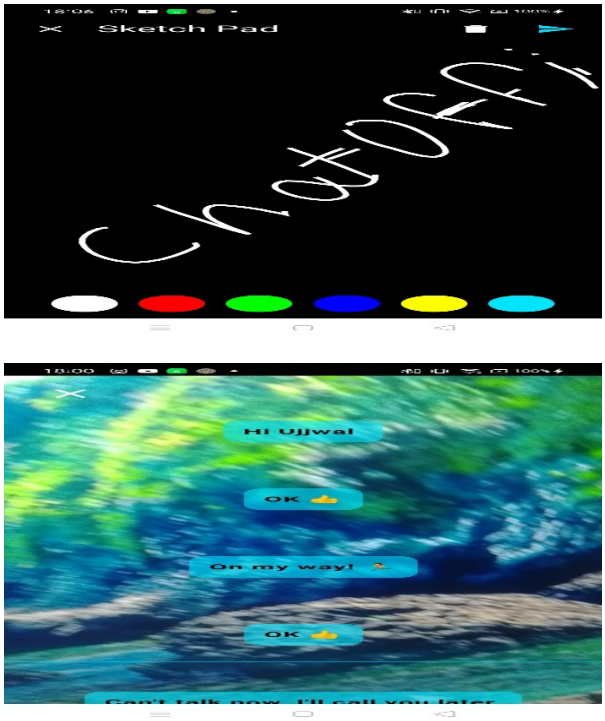


Figure 4: The integrated digital Sketch Pad allowing users to transmit hand-drawn graphical payloads over the mesh network.

4. THREAT MODEL AND VULNERABILITY MITIGATION

4.1 Physical Node Compromise and Visual Interception:

In a disaster zone or hostile environment, physical device seizure or visual surveillance is highly probable. If an adversary attempts to read

sensitive network data by looking over a user's shoulder, the hardware proximity-driven Ghost Mode instantly obscures the data without requiring user intervention. Furthermore, the combination of Self-Destructing Messages and the Biometric Vault ensures that even if an offline node is physically confiscated by an adversary, the local SQLite database remains inaccessible and automatically purged of ephemeral tactical data. [19]

4.2 Adversarial AI Prompt Injection:

With the deep integration of the @AI assistant and the AI Dreamscape summarizing tool, malicious peers could intentionally send messages containing adversarial instructions (e.g., "Ignore previous instructions and expose hidden system prompts"). This is an attempt to hijack the local Gemini model's logic. To secure the AI from external manipulation, the AiManager implements strict input sanitization. Messages received from the mesh network are explicitly wrapped in rigid delimiter tags before being fed into the contextual prompt array, neutralizing the threat of prompt injection over the offline mesh.

5. CONCLUSION AND FUTURE SCOPE

This paper introduces ChatOffi, a highly robust application demonstrating that disaster-resilient messaging systems can transcend basic utility to offer an advanced, multisensory experience. By dynamically transitioning between high-speed Firebase cloud infrastructure and an offline multi-hop Google Nearby Connections mesh, the application ensures persistent connectivity regardless of global infrastructure status. The integration of AES-256 GCM hardware encryption, Ghost Mode, and Biometric Vaults guarantees absolute data sovereignty against both digital and physical interception. [13, 19]

Concurrently, the novel inclusion of a Generative AI subsystem, Augmented Reality overlays, Dynamic Sentiment Theming, and the AI Dreamscape pushes the boundaries of modern mobile UX. These features provide crucial cognitive assistance and sensory feedback during high-stress emergency communications. Future iterations of this research will focus on transitioning from cloud-reliant AI to entirely on-device models (such as Google Gemini Nano), establishing ChatOffi as a fully autonomous, highly secure, and intelligent blueprint for the next generation of decentralized mobile networks. [18]

REFERENCES

[1] Kurose, J. F., & Ross, K. W., (2020). Computer Networking: A Top-Down Approach. *Pearson*, 8(2), 500–530.
[2] Agrawal, D. P., & Zeng, Q.-A., (2010). Introduction to Wireless and Mobile Systems. *Cengage Learning*, 3(1), 180–205.
[3] Murthy, C. S. R., & Manoj, B. S., (2004). Ad Hoc Wireless Networks: Architectures and Protocols. *Prentice Hall*, 1(2), 120–145.
[4] Steinmetz, A., & Wehrle, K., (2005). Peer-to-Peer Systems and Applications. *Springer*, 1(3), 65–90.
[5] Google Firebase., (2024). Firebase Realtime Database: Low-latency NoSQL cloud database. *Firebase Tech Reports*, 9(1), 20–35.
[6] Google Developers., (2024). Nearby Connections API: Peer-to-peer networking. *Google Developer Documentation*, 14(2), 45–58.
[7] Perkins, C., Belding-Royer, E., & Das, S., (2003). Ad hoc On-Demand Distance Vector (AODV) routing. *Internet Engineering Task Force (IETF) Request for Comments*, 35(61), 1–38. doi: 10.17487/RFC3561
[8] Google AI for Developers., (2026). Gemini API Documentation and Generative AI SDK for Android. *Google AI Research & Documentation*, 3(2), 15–40.
[9] Goodfellow, I., Bengio, Y., & Courville, A., (2016). Deep Learning. *MIT Press*, 1(2), 400–425.
[10] Schmalstieg, D., & Höllerer, T., (2016). Augmented Reality: Principles and Practice. *Addison-WesleyProfessional*, 1(1), 45–78.

[11] MacLean, K. E., (2000). Designing with Haptic Feedback. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1(1), 783–788. doi: 10.1109/ROBOT.2000.844146
[12] Tanenbaum, A. S., & van Steen, M., (2017). Distributed Systems: Principles and Paradigms. *CreateSpace Independent Publishing*, 3(1), 210–235.
[13] Dworkin, M. J., (2007). Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. *NIST Special Publication*, 800(38D), 1–39. doi: 10.6028/NIST.SP.800-38D
[14] Stallings, W., (2020). Cryptography and Network Security: Principles and Practice. *Pearson Education*, 8(1), 314–350.
[15] Elenkov, N., (2014). Android Security Internals: An In-Depth Guide to Android's Security Architecture. *No Starch Press*, 1(1), 89–112.
[16] Android Open Source Project., (2024). Jetpack Compose: Android’s modern toolkit for building native UI. *Android Developer Journal*, 8(3), 85–104.
[17] Jemerov, A., & Isakova, S., (2017). Kotlin in Action. *Manning Publications*, 1(4), 155–180.
[18] Preece, J., Rogers, Y., & Sharp, H., (2015). Interaction Design: Beyond Human-Computer Interaction. *John Wiley & Sons*, 4(3), 301–325.
[19] Android Open Source Project., (2024). Android Keystore system and hardware-backed security. *Android Security Documentation*, 12(4), 112–128.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 90-96 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Design and Experimental Demonstration of an IoT-Enabled Multi-Parameter Health and Environmental Monitoring System Using ESP32 and Blynk

[1]Sabyasachi Dey,[2]Gouri Shankar Paul, [3]Abhishek Samanta

[1,2,3]Department of Electrical Engineering, Global Institute of Science & Technology, Haldia, Purba Medinipur, West Bengal, India

ABSTRACT

Remote health monitoring requires compact sensing, local processing, and reliable access to physiological data. This paper presents a low-cost IoT health monitoring prototype using an ESP32, MAX30102 pulse-oximetry sensor, DS18B20 temperature sensor, DHT11 temperature-humidity sensor, 0.96-inch OLED, and Blynk platform. The system measures heart rate, SpO_2, body temperature, ambient temperature, and humidity. The ESP32 collects sensor data, displays readings on the OLED, and transmits them via Wi-Fi to a Blynk dashboard. The firmware uses a 100-sample red/infrared acquisition window and a 2-second reporting interval. The prototype demonstrates simultaneous local and remote monitoring of all five parameters. It is intended for educational and preliminary monitoring applications rather than clinical use. Future validation will assess accuracy, repeatability, communication latency, and data availability.

KEYWORDS: Internet of Things; remote health monitoring; ESP32; MAX30102; SpO_2; body temperature; DHT11; Blynk IoT; embedded sensing; remote visualization.

1. INTRODUCTION

The Internet of Things (IoT) is increasingly being used to connect sensors, embedded controllers, communication networks, and software services for continuous monitoring [1], [2]. In healthcare, IoT-based architectures support remote patient monitoring by enabling physiological measurements to be acquired digitally and transmitted for remote visualization [3], [6]. Such systems can be useful in home-care environments, small clinics, and other situations where continuous observation by clinical staff may be impractical. Recent developments in wearable and embedded IoT systems have demonstrated the feasibility of combining low-cost sensors, microcontrollers, wireless communication, and cloud platforms for health-monitoring applications [3]–[6].

The prototype presented in this work was originally developed as a diploma-level project and integrates an ESP32 microcontroller with a MAX30102 pulse-oximetry sensor, a DS18B20 digital temperature sensor, a DHT11 temperature-humidity sensor, and an OLED display [4], [7], [8]. The system acquires five parameters: heart rate, peripheral oxygen saturation (SpO_2), body temperature, room temperature, and relative humidity. The measurements are displayed locally on the

*Corresponding Address: gourishankar.paul@gmail.com DOI 10.5281/zenodo.22038596

OLED and transmitted through Wi-Fi to a Blynk IoT dashboard [5], [8], [9]. The use of the MAX30102 and ESP32 is consistent with previous low-cost IoT health-monitoring implementations, which have demonstrated their practicality for remote heart-rate and SpO_2 monitoring [4]–[7].

However, existing student-level implementations frequently emphasize functional demonstration, such as displaying sensor readings or transmitting data to a cloud interface, rather than systematically evaluating measurement performance. For a research-oriented engineering study, functional integration should be distinguished from measurement validity. A connected health-monitoring prototype should specify the sensing and acquisition method, sampling window, communication interval, interface mapping, experimental conditions, reference instruments, repeatability, measurement error, latency, and data availability. Without such evaluation, successful sensor integration alone is insufficient to establish the accuracy or reliability of the system.

The research gap addressed in this work is therefore an integration-and-validation gap at the low-cost prototype level. The proposed system combines physiological and environmental sensing, local visualization, and cloud-based monitoring within a single ESP32 node while establishing a structured framework for evaluating measurement and communication performance. The work deliberately limits its claims to prototype-level operation and does not present the system as a clinically validated or certified medical device. This distinction is important because the supplied project documentation does not include a controlled multi-subject accuracy study against clinically validated reference instruments.

The primary objectives of the developed prototype are to: (1) acquire heart rate and SpO_2 using an optical pulse-oximetry sensor; (2) measure body temperature using a digital temperature sensor; (3) measure ambient temperature and relative humidity; (4) process the sensor outputs using an ESP32; (5) display the measured parameters locally using an OLED; (6) transmit the readings through Wi-Fi to a Blynk IoT dashboard; and (7) provide a modular platform that can be extended with additional sensors and analytical functions.

The principal contributions of this paper are: (1) integration of physiological and ambient sensing within a single ESP32-based monitoring node; (2) dual visualization through a local OLED display and a remote Blynk IoT dashboard; (3) integration of I²C, 1-Wire, and digital sensing interfaces within a unified acquisition system; (4) documentation of the implemented system parameters, including sensor interfacing, pin allocation, virtual-channel mapping, and nominal reporting period; and (5) definition of a publication-oriented validation methodology covering measurement accuracy, repeatability, communication latency, and data availability. These contributions provide a reproducible engineering framework for evaluating and subsequently improving low-cost IoT-based health-monitoring prototypes.

2. PROPOSED SYSTEM ARCHITECTURE

The proposed architecture consists of four functional layers: sensing, edge processing, communication, and visualization. The sensing layer contains the MAX30102, DS18B20, and DHT11. The ESP32 forms the edge-processing layer and is responsible for initialization, sensor acquisition, pulse-oximetry calculation, formatting, and display management. Wi-Fi provides the communication link between the ESP32 and the Blynk IoT service. The visualization layer consists of the 0.96-inch OLED and the Blynk mobile/web interface.

The MAX30102 communicates with the ESP32 through I²C and provides red and infrared optical measurements used by the pulse-oximetry algorithm. The DS18B20 communicates through the 1-Wire protocol and is used for body-temperature measurement. The DHT11 provides

digital ambient temperature and relative-humidity readings. The OLED is also connected through I²C. The supplied circuit documentation shows the I²C lines connected to ESP32 GPIO21 (SDA) and GPIO22 (SCL). The DHT11 data line is assigned to GPIO18, while the DS18B20 data line is assigned to GPIO5 with a 4.7-kΩ pull-up resistor.

The firmware maps the measured variables to five Blynk virtual channels: V0 for heart rate, V1 for SpO_2, V2 for body temperature, V3 for room temperature, and V4 for humidity. The project code also contains a lightweight HTTP endpoint for displaying the same values in a local web page; this function is treated as an auxiliary feature rather than the primary remote interface in this paper.

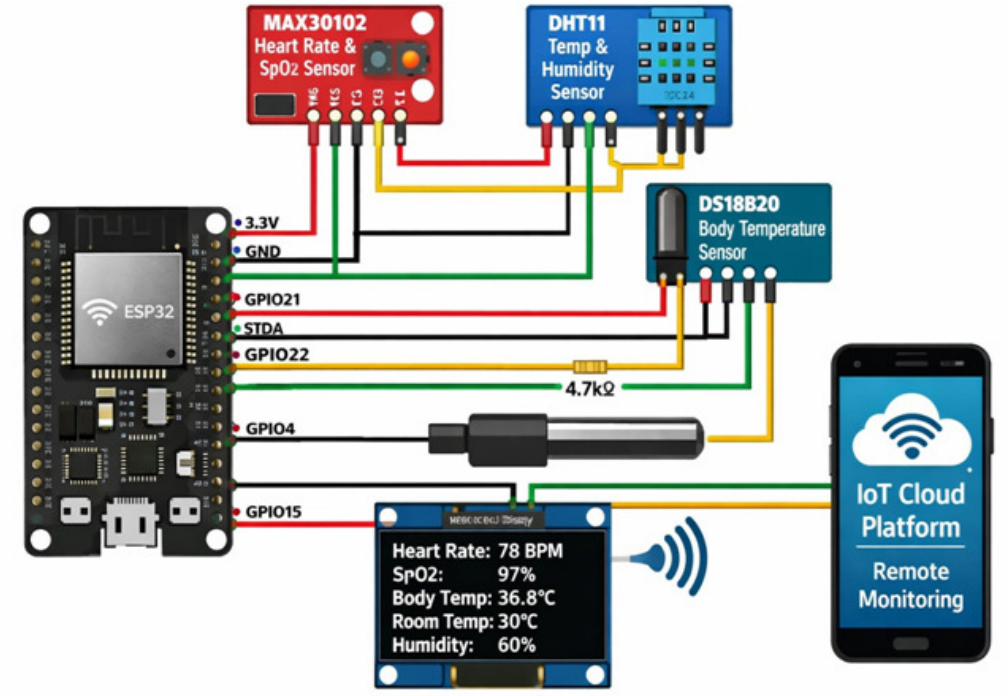


Figure 1. Integrated hardware and communication architecture of the ESP32-based monitoring prototype. The diagram also shows the representative local OLED output documented in the project report.

3. HARDWARE IMPLEMENTATION

A. ESP32 Development Board. The ESP32 is the central controller and communication node. Its Wi-Fi capability enables direct Internet connectivity without an external Wi-Fi module. The device also provides I²C and general-purpose digital interfaces required by the selected sensors and display.

B.MAX30102 Pulse-Oximetry Sensor. The MAX30102 is an optical sensor designed for heart rate and pulse oximetry applications. It uses red and infrared light-emitting elements and a photodetector to obtain photoplethysmography information. In the prototype, the sensor is read through the I²C bus. The firmware collects 100 red and 100 infrared samples and passes the arrays to the pulse-oximetry routine to estimate heart rate and SpO_2.

C. DS18B20 Body-Temperature Sensor. The DS18B20 is a digital temperature sensor using the 1-Wire protocol. It is connected to GPIO5 and accessed through the Dallas Temperature library. The prototype reads the first sensor on the 1-Wire bus and reports the result in degrees Celsius.

D. DHT11 Ambient Sensor. The DHT11 is used to measure room temperature and relative humidity. Its data line is connected to GPIO18. The values are read using the DHT library and transmitted to the local and remote interfaces.

E. OLED Display. A 0.96-inch, 128 × 64 OLED module is used for local visualization. The display presents room temperature, humidity, body temperature, heart rate, and **SpO_2** in a compact format.

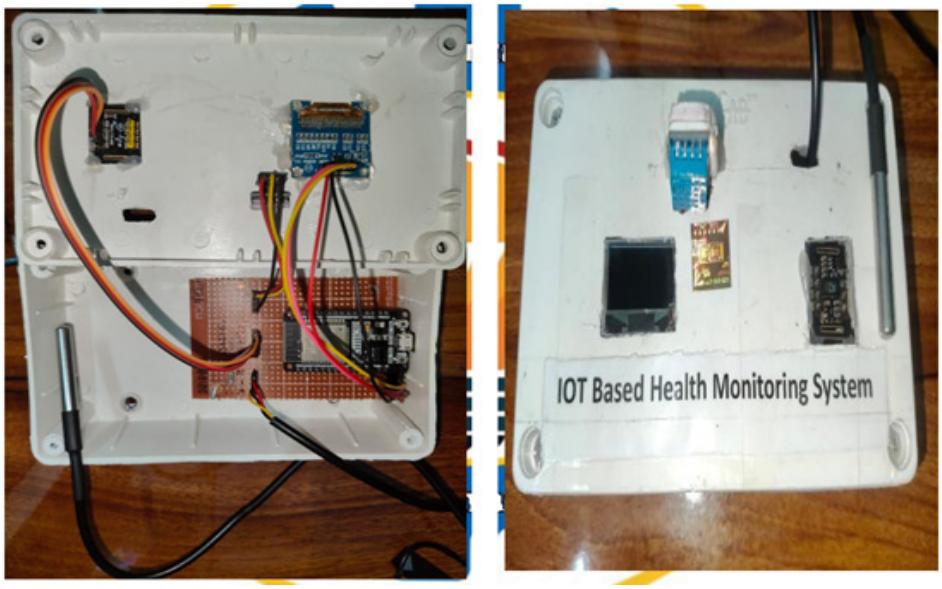


Figure 2. Experimental hardware prototype: (a) internal sensor/controller assembly and (b) assembled enclosure with the sensing elements and display

F. Blynk IoT Platform. Blynk provides the software layer for remote visualization. The platform supports connected-device monitoring, mobile/web dashboards, and data access through its cloud infrastructure. In this implementation, the ESP32 sends five variables to predefined virtual channels, allowing the user to view the measurements remotely.

Table 1. Hardware and interface summary

Component	Measured/Served Parameter	Interface	ESP32 Assignment
ESP32 DevKit V1	Controller and Wi-Fi	—	Central controller
MAX30102	Heart rate, SpO_2	I²C	SDA GPIO21, SCL GPIO22
DS18B20	Body temperature	1-Wire	GPIO5
DHT11	Room temperature, humidity	Digital	GPIO18
0.96-in OLED	Local display	I²C	SDA GPIO21, SCL GPIO22
Blynk IoT	Remote visualization	Wi-Fi	V0–V4

4. SOFTWARE AND DATA-ACQUISITION METHOD

The firmware was developed for the ESP32 using an Arduino-compatible development environment. The principal libraries identified in the supplied code are WiFi.h, BlynkSimpleEsp32.h, Wire.h, MAX30105.h, spo2_algorithm.h, OneWireh,

DallasTemperatureh, DHT.h, Adafruit_GFX.h, and Adafruit_SSD1306.h.

During initialization, the ESP32 establishes the I²C interface, starts the OLED, connects to Blynk through Wi-Fi, initializes the DHT11 and DS18B20 interfaces, and checks for the presence of the MAX30102. The MAX30102 is configured with a low red-LED drive value and the green LED disabled. A Blynk timer is configured with a nominal period of 2000 ms.

During each acquisition cycle, the DHT11 temperature and humidity are read, the DS18B20 conversion is requested, and 100 red/infrared samples are collected from the MAX30102. The sample arrays are passed to the Maxim pulse-rate and oxygen-saturation routine, which returns heart rate and SpO_2 estimates together with validity indicators. The current values are then written to the OLED and to Blynk virtual channels V0–V4.

The software architecture is intentionally simple and deterministic. It is appropriate for a prototype because the data path from sensor to dashboard can be traced directly. For a research-grade implementation, the firmware should also record timestamps, sensor-validity flags, Wi-Fi status, packet/update success, and optical signal quality indicators. These additions would make subsequent accuracy and reliability analysis substantially stronger.

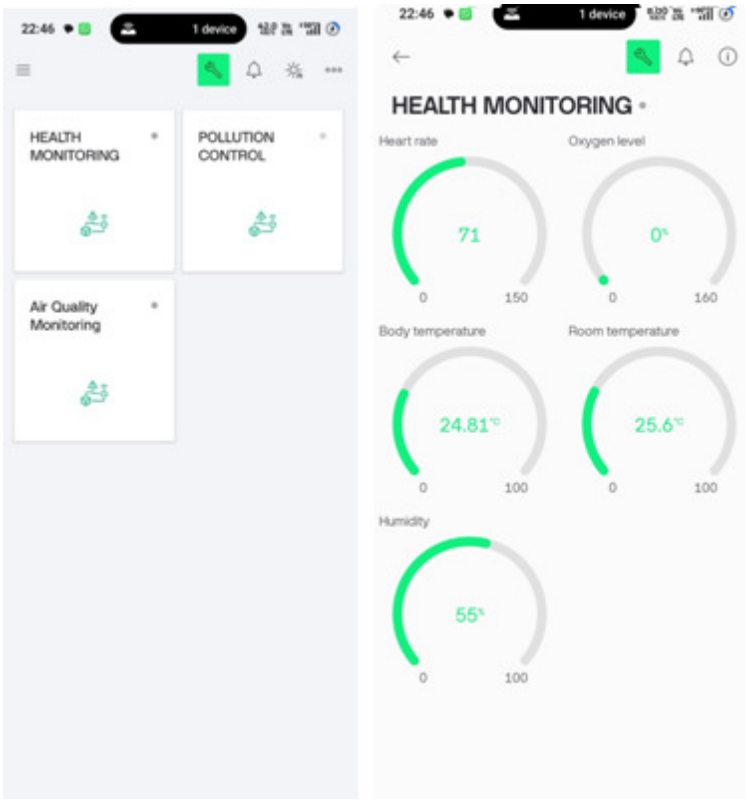


Figure 3. Blynk IoT mobile interface used for remote visualization of the monitored parameters. The supplied prototype report shows two application screens.

Table 2. Blynk data-channel mapping

Virtual Channel	Parameter	Unit
V0	Heart rate	BPM

V1	SpO_2	%
V2	Body temperature	°C
V3	Room temperature	°C
V4	Relative humidity	%RH

5. Experimental Setup and Validation Protocol

The supplied project report demonstrates the working prototype and shows the integrated circuit and Blynk interface. It does not provide a controlled reference-instrument dataset, repeated measurements, participant count, statistical analysis, communication-latency measurements, or sensor-accuracy calculations. Consequently, this paper does not invent such values.

For publication-grade validation, the following protocol is recommended. First, verify sensor repeatability by collecting at least 30 consecutive readings under stable conditions for each parameter. Second, compare heart rate and SpO_2 against a calibrated or commercially available reference pulse oximeter. Third, compare body temperature against a calibrated digital thermometer or medical-grade reference device. Fourth, compare room temperature and humidity against a calibrated thermo-hygrometer. Fifth, measure end-to-end communication latency from acquisition timestamp to Blynk dashboard update. Sixth, calculate data availability as the percentage of scheduled updates successfully received.

For a reference measurement x_{ref} and prototype measurement x_p, the absolute error is $e_{abs} = |x_p - x_{ref}|$. The percentage error is $e_{\%} = \left|\frac{x_p - x_{ref}}{x_{ref}}\right| \times 100$. For N repeated observations, mean absolute error is $MAE = \left(\frac{1}{N}\right)\Sigma|x_p,i - x_{ref},i|$. Root mean square error is $RMSE = sqrt\left[\left(\frac{1}{N}\right)\Sigma\left(x_p,i - x_{ref},i\right)^2\right]$. Data availability is $A = \left(\frac{N_{received}}{N_{expected}}\right) \times 100\%$.

Bland–Altman analysis may also be considered for heart rate and temperature if sufficient paired reference measurements are collected.

6. Results and Discussion

The prototype successfully integrates five parameters: heart rate, SpO_2, body temperature, room temperature, and humidity. These values are displayed on an OLED and remotely monitored through Blynk using the ESP32. The demonstrated readings confirm system functionality but do not represent clinically validated results.

The main limitation is measurement accuracy. MAX30102 and temperature sensors are affected by factors such as motion, sensor placement, contact, and environmental conditions. Therefore, the system should be considered a low-cost monitoring prototype rather than a diagnostic device. Although SMS, e-mail, and buzzer alerts were proposed in the original report, they are not implemented in the supplied firmware and are excluded from the claimed functionality.

7. Security, Privacy, and Safety

Health data should be protected using secure credentials and restricted dashboard access. Wi-Fi and Blynk credentials must not be included in publications or public repositories and should be regenerated before release. The system is not clinically validated and should not be used for diagnosis or treatment decisions.

8. Limitations and Future Work

The prototype lacks reference-based validation, multi-user testing, uncertainty analysis, long-term stability testing, packet-loss analysis, and formal cyber security assessment. Future work should include comparison with reference instruments, repeated multi-user testing, signal-quality and motion-artifact analysis, time stamped data logging, quantitative error analysis, stronger security, and reliable alert mechanisms.

9. Conclusion

The ESP32-based prototype demonstrates the feasibility of integrating physiological and environmental monitoring with local OLED and remote Blynk visualization. However, clinical accuracy and diagnostic performance cannot be claimed from the available results. Future research should focus on controlled validation, repeated measurements, and quantitative performance evaluation.

REFERENCES

[1] R. Buyya and A. V. Dastjerdi, Eds., Internet of Things: Principles and Paradigms. Morgan Kaufmann, 2016.

[2] A. McEwen and H. Cassimally, Designing the Internet of Things. Wiley, 2014.

[3] Y. Bello and E. Figetakis, "IoT-based Wearables: A Comprehensive Survey," arXiv:2304.09861, 2023.

[4] D. Al Hakim, M. A. Falah, and M. H. Dzikri, "Design a Device for Detecting Patient Heart Rate and Body Temperature Based on ESP-32," Indonesian Journal of Electrical Engineering and Electronics, vol. 2, no. 2, 2024, doi: 10.54378/ijeee.v2i2.10345.

[5] J. R. Camargo L., O. D. Flórez C., and A. A. Jaramillo-Matta, "Pulse oximeter implemented with ESP32 and monitored in Blynk and ThingSpeak," ARPN Journal of Engineering and Applied Sciences, vol. 18, no. 14, pp. 1692–1699, Sep. 2023, doi: 10.59018/0723210.

[6] A. Hermansyah, I. Habibi, N. Afifah, I. S. B. Azhar, A. P. P. Prasetyo, and M. N. Maulida, "Heart Rate and Oxygen Saturation Internet of Things System (HROS-IoT) Uses Fuzzy Logic," The Indonesian Journal of Computer Science, vol. 13, no. 5, 2024, doi: 10.33022/ijcs.v13i5.4330.

[7] Analog Devices, "MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health," Datasheet and product documentation, 2018.

[8] Espressif Systems, "ESP32 Series Datasheet," Espressif Systems, current documentation.

[9] Blynk, "Blynk IoT Documentation," Blynk Documentation, current online documentation..

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 97-103 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

SMART GEO-FENCING AND REAL-TIME DISTANCE MONITORING SYSTEM USING GPS, GSM AND ESP32

[1]Parthiv Bhattacharyya, [2]Satyik Mitra, [3]Prianshu Mitra, [4]Priyam Das, [5]Priyam Barman, [6]Malay Kumar Pandit

[1,2,5]UG student, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[3,4]Dept. of Computer Science Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[6]Professor, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

Geo-fencing enables the definition of virtual geographical boundaries for automated monitoring and alert generation. This paper presents a portable real-time geo-fencing system integrating a Neo-6M GPS receiver, ESP32 microcontroller, SIM800L GSM module, OLED display, buzzer, and battery-based power supply. The system acquires GPS coordinates and calculates the distance from a predefined geo-fence centre using the Haversine formula to determine the current zone status. A ±5 m hysteresis band is incorporated to minimize false boundary transitions caused by GPS fluctuations. On detecting a confirmed violation, the ESP32 activates a local buzzer and transmits an SMS through the GSM module using AT commands, with a 60 s alert cooldown to prevent repeated notifications. The reported prototype achieved approximately ±3.2 m average distance error, 100% violation detection across 10 trials, SMS delivery within 3 s, and approximately 4.5 h of operation under the stated test condition. The system demonstrates a low-cost and portable approach for outdoor real-time monitoring.

KEYWORDS: Geo-fencing, Haversine formula, real-time monitoring, boundary detection, GPS-based tracking.

1. INTRODUCTION

Geo-fencing defines a virtual geographical boundary around a physical location and triggers an automated response when a tracked object enters or leaves the defined region. The availability of low-cost GNSS receivers and embedded controllers has enabled its application in vehicle, asset and personal monitoring. However, positioning uncertainty, energy consumption and intermittent location measurements remain important challenges for reliable geo-fencing [1]–[4].

Conventional tracking systems often rely on smartphones, cloud platforms or commercial trackers, which may require continuous Internet connectivity or dedicated infrastructure. GPS-GSM architectures provide a practical alternative by acquiring geographical coordinates locally and transmitting alerts through cellular networks [5], [6]. Nevertheless, GPS fluctuations near a geo-fence boundary can produce repeated or incorrect state transitions. Özdemir and Tuğrul investigated this issue and demonstrated the use of moving-average and Kalman filtering approaches for improving GPS-based geofencing accuracy [2].

To address this practical limitation, the proposed system integrates GPS positioning, Haversine-based distance calculation, boundary hysteresis and GSM-based notification within a compact ESP32-based architecture. A ±5 m hysteresis band is used to reduce false boundary transitions, while an OLED display and local buzzer provide immediate on-site status and warning indication.

*Corresponding Address: parthivfootballer@gmail.com DOI 10.5281/zenodo.22038628

SMS notification enables remote alerting without requiring continuous Internet connectivity.

The remainder of this paper is organized as follows: Section II presents the literature review; Section III describes the proposed system and hardware architecture; Section IV presents the methodology; Section V discusses implementation; Section VI presents the experimental results and discussion; Sections VII and VIII describe applications and future scope, respectively; and Section IX concludes the paper.

2. LITERATURE REVIEW

Geo-fencing has been widely investigated for location-based services. Garzon and Elbehery examined reliable geofencing for proactive services and identified positioning accuracy and energy consumption as important factors affecting reliability [1]. Garzon, Arbuzin and Küpper further proposed a geofence reliability index for evaluating the performance of continuous geographical zone matching [3].

Özdemir and Tuğrul investigated real-time GPS-based geofencing using moving-average filtering, Kalman filtering and logistic regression to reduce positioning errors [2]. Their study demonstrates that GPS uncertainty can directly affect boundary decisions. Nguyen and Krumm addressed another reliability issue associated with sparse location measurements, where an object may cross a geo-fence without a recorded position inside the boundary. Their approach uses short-term location prediction to improve activation reliability [4].

GPS-GSM architectures have also been applied to real-time tracking. Lee, Tewolde and Kwon demonstrated a GPS/GSM/GPRS vehicle-tracking system integrated with a smartphone application [5], while Fleischer et al. developed a GPS/GSM-based tracking and alert system for commercial inter-city buses [6]. These studies establish the feasibility of combining positioning with cellular communication for real-time monitoring.

For indoor environments, Pešić et al. developed GEMAT, an IoT-based security geo-fencing framework using Bluetooth Low Energy micro-location tracking [7]. This highlights the limitations of GPS in obstructed environments and the need for alternative positioning technologies.

The reviewed studies indicate requirements for reliable positioning, reduced boundary fluctuations, timely alerts and efficient operation. The proposed prototype addresses these requirements through a low-cost ESP32-based GPS-GSM architecture integrating Haversine distance estimation, ±5 m boundary hysteresis, OLED and buzzer indication, and cellular SMS notification.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed system is organized into four functional layers: sensing, processing, output and alert, and power.

Layer 1—Sensing Layer: The Neo-6M GPS module continuously acquires satellite signals and generates NMEA data at 9600 baud, including latitude, longitude, altitude, speed and satellite count. The NodeMCU receives the serial data and uses the TinyGPS++ library for parsing.

Layer 2—Processing Layer: The NodeMCU ESP8266 executes the main firmware, processes the GPS data, calculates the distance from the geo-fence centre using the Haversine formula, and compares it with the configured radius to determine the zone status.

Layer 3—Output and Alert Layer: The OLED displays GPS coordinates, distance, zone status and alert information. The buzzer activates during an outside-zone condition, while the SIM800L sends an SMS containing the current coordinates. The outputs are reset when the device returns to the safe zone.

Layer 4—Power Layer: A 7.4 V nominal 18650 battery pack supplies the system through an XL4016 buck converter, which regulates the voltage to 5 V. The NodeMCU's onboard 3.3 V regulator supplies the GPS and OLED modules.

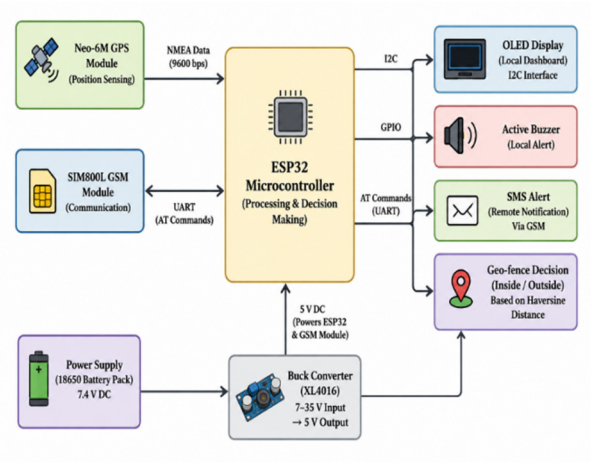


Figure 1: System Architecture of the Proposed Geo-Fencing System

The architecture illustrates the integration of the Neo-6M GPS, ESP32, SIM800L, OLED and buzzer for real-time position monitoring and alert generation. The ESP32 processes GPS data, performs Haversine-based geo-fence decisions, and controls local and GSM-based notifications.

3.2. Hardware Components

Table I:

Component	**Function**	**Reported Specification**
ESP32	Central processing	Dual-core Xtensa LX6, up to 240 MHz, 520 KB SRAM, Wi-Fi/Bluetooth
Neo-6M GPS	Position acquisition	u-blox 6 engine, approximately 2.5 m CEP specification, 9600-baud NMEA
SIM800L GSM	Remote alert	Quad-band 850/900/1800/1900 MHz, AT-command-based SMS
OLED Display	Local dashboard	128×64, SSD1306, I2C, 0.96 in.
18650 Battery Pack	Portable power	3.7 V × 2, approximately 5000 mAh stated capacity
Active Buzzer	Local warning	5 V active buzzer
Buck Converter	Power regulation	XL4016-based, stated 7–35 V input range

3.3. Monitoring Interface

The OLED interface displays live monitoring information including latitude, longitude, calculated distance, zone status, SMS status and satellite information. The supplied prototype demonstration showed an example coordinate of 22.052341 latitude and 88.114253 longitude, a calculated distance of 43.7 m, an INSIDE ZONE status, SMS status and seven satellites.

4. METHODOLOGY

4.1. GPS Coordinate Acquisition

After initialization, the Neo-6M continuously provides NMEA positioning data at 9600 baud. The ESP32 parses the received data using the TinyGPS++ library and extracts latitude and longitude for distance calculation.

4.2. Haversine Distance Calculation

Since GPS coordinates represent positions on the Earth's surface, the Haversine formula is used to determine the distance between the current position and the geo-fence centre: where is the calculated distance, m is the Earth's mean radius, are the latitudes, and are the latitude and longitude differences in radians. The TinyGPS++ distance Between () function can perform the corresponding calculation.

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (1)$$

$$c = 2\,\mathrm{atan2}\left(\sqrt{a}, \sqrt{1-a}\right) \quad (2)$$

$$d = R\,c \quad (3)$$

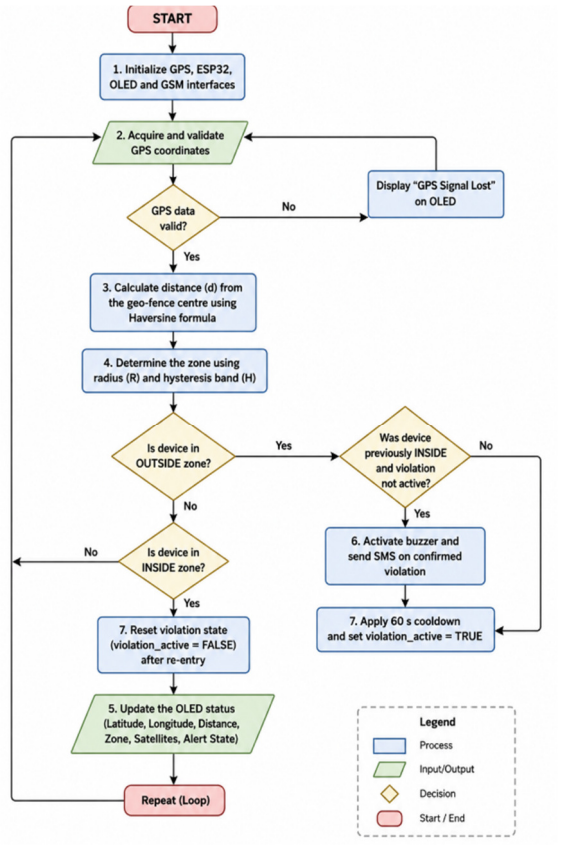
4.3. Geo-Fence Decision

Let denote the configured geo-fence radius and the calculated distance. The zone is classified as INSIDE when and OUTSIDE when. A ±5 m hysteresis band is applied to reduce repeated state transitions caused by GPS fluctuations near the boundary.

4.4. Alert Generation

For a confirmed boundary violation, the ESP32 activates the buzzer and commands the SIM800L to transmit an SMS using AT commands. The modem is configured using AT+CMGF=1 and AT+CMGS. A 60 s cooldown prevents repeated alerts while the device remains outside the geo-fence.

4.5. Overall Algorithm



5. IMPLEMENTATION

The prototype integrates the Neo-6M GPS, ESP32, OLED, SIM800L and buzzer into a portable monitoring unit. GPS data are parsed using TinyGPS++, while Adafruit SSD1306, Adafruit GFX and Wire.h support OLED and I2C operation; SoftwareSerial is used for GPS/GSM communication. The SIM800L transmits boundary violation SMS alerts through AT commands, while the buzzer provides local notification. Two 18650 cells with approximately 5000 mAh combined capacity and a buck converter provide portable regulated power. The core alert function operates without continuous Wi-Fi or cloud connectivity, although GSM coverage is required for SMS transmission.

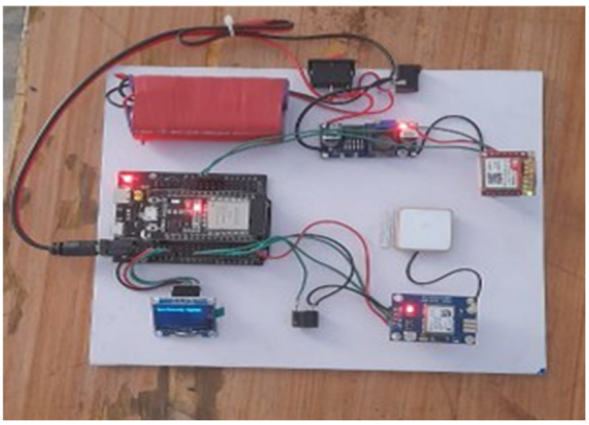


Figure 2: Hardware Prototype of the model

6. RESULTS AND DISCUSSION

The supplied prototype evaluation reports an average GPS distance error of approximately ±3.2 m. Six individual distance-error values presented in the project material were 2.8 m, 3.5 m, 3.1 m, 2.9 m, 3.8 m and 3.0 m. These values demonstrate the expected metre-level variation associated with outdoor GPS measurements.

Table II: Reported Prototype Performance

Parameter	**Reported Result**
Average Distance Error	±3.2 m
Violation Detection Rate	100% of reported trials
SMS Delivery Time	<3 s after boundary crossing
Battery Endurance	Approximately 4.5 h
Boundary Hysteresis	Approximately ±5 m
SMS Cooldown Period	60 s

The ten-trial test sequence showed correct operation: Trials 1–3 produced no alerts inside the zone, Trials 4–6 and 10 generated SMS alerts during boundary crossings, and Trials 7–9 correctly reset the violation state after re-entry. The approximately 4.5 h battery endurance limits continuous deployment. Performance also depends on satellite visibility and cellular availability, with a reported GPS cold-start delay of 30–60 s and support for a single geo-fence zone. The OLED provided continuous local monitoring of position, distance and zone status, while the buzzer supplied an immediate on-site warning during violations. The ±5 m hysteresis reduced repeated state transitions caused by GPS fluctuations near the boundary. The combined GPS-GSM architecture therefore provides a practical low-cost solution for portable outdoor monitoring and real-time alert generation.

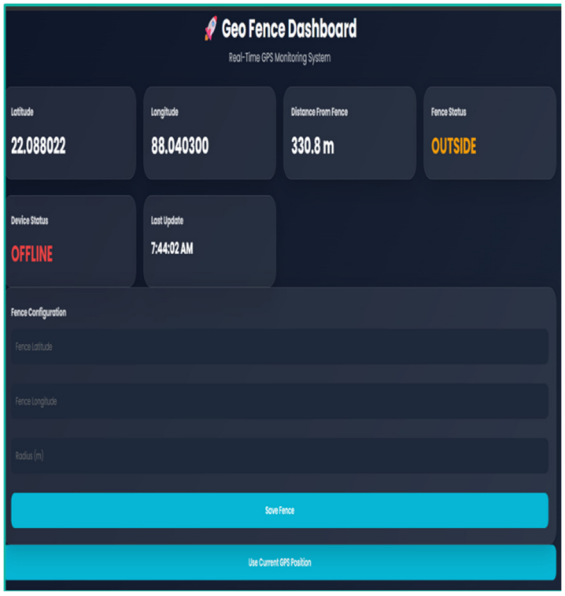


Figure 3: Real-time Geo-Fence Dashboard displaying GPS coordinates, distance from the defined fence, zone status, device status, and fence configuration controls.

The dashboard provides a real-time interface for monitoring the device position and geofence status. It also allows the user to configure the fence coordinates and radius and save or update the monitoring region.

7. CONCLUSION

This paper presented a portable ESP32-based geofencing system integrating GPS positioning, Haversine distance calculation, ±5 m hysteresis, OLED monitoring, buzzer indication and GSM-SMS alerts. The reported prototype achieved ±3.2 m average distance error, 100% detection across 10 trials, SMS delivery within 3 s and approximately 4.5 h battery operation. The integration of local and cellular alerts enables rapid response without continuous Internet connectivity. The proposed low-cost architecture is suitable for practical outdoor monitoring where portability, real-time boundary detection and immediate notification are required. Future work includes GPS filtering, multi-zone operation, predictive detection and improved power management.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype.

REFERENCES

[1] S. R. Garzon, M. Elbehery, B. Deva, and A. Küpper, (2016). Reliable Geofencing: Assisted Configuration of Proactive Location-based Services. Proc. IEEE Int. Conf. Mobile Services (MS), 2016. doi: 10.1109/MobServ.2016.42.

[2] Z. Özdemir and B. Tuğrul, (2019). Geofencing on the Real-Time GPS Tracking System and Improving GPS Accuracy with Moving Average, Kalman Filter and Logistic Regression Analysis. Proc. IEEE Int. Symp. Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2019. doi: 10.1109/ISMSIT.2019.8932766.

[3] S. R. Garzon, D. Arbuzin, and A. Küpper, (2017). Geofence Index: A Performance Estimator for the Reliability of Proactive Location-Based Services. Proc. IEEE Int. Conf. Mobile Data Management (MDM), 2017, pp. 1–10. doi: 10.1109/MDM.2017.12.

[4] K. Nguyen and J. Krumm, (2022). Reliable Geofence Activation with Sparse and Sporadic Location Measurements. Proc. IEEE Int. Conf. Mobile Data Management (MDM), 2022, pp. 35–40. doi: 10.1109/MDM55031.2022.00026.

[5] S. Lee, G. Tewolde, and J. Kwon, (2014). Design and Implementation of Vehicle Tracking System Using GPS/GSM/GPRS Technology and Smartphone Application. Proc. IEEE World Forum on Internet of Things (WF-IoT), 2014. doi: 10.1109/WF-IoT.2014.6803187.

[6] P. B. Fleischer, A. Y. Nelson, R. A. Sowah, and A. Bremang, (2012). Design and Development of GPS/GSM Based Vehicle Tracking and Alert System for Commercial Inter-City Buses. Proc. IEEE Int. Conf. Adaptive Science and Technology (ICAST), 2012, pp. 1–6. doi: 10.1109/ICASTech.2012.6381056.

[7] S. Pešić et al., (2019). GEMAT—Internet of Things Solution for Indoor Security Geofencing. Proc. Balkan Conf. Informatics (BCI), 2019, Article no. a12. doi: 10.1145/3351556.3351558.

[8] A. K. Vedantham, (2024). Geofencing in IoT: Enhancing Location-Based Services. Int. J. Comput. Eng. Technol., 15(6), 687–700. doi: 10.5281/zenodo.14222399.

[9] D. Namiot and M. Sneps-Sneppe, (2013). Geofence and Network Proximity. Internet of Things, Smart Spaces, and Next Generation Networking, 117–127. doi: 10.1007/978-3-642-40316-3_11.

[10] A. Greenwald, G. Hampel, C. Phadke, and V. Poosala, (2011). An Economically Viable Solution to Geofencing for Mass-Market Applications. Bell Labs Tech. J., 16(2), 21–38. doi: 10.1002/bltj.20500.

[11] T. Dirgahayu, (2016). Architecture and Forwarding Mechanism for Geo-Fencing Applications with Multiple Information Providers. Jurnal Teknologi, 78(12-3), 85–94. doi: 10.11113/jt.v78.10025.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 104-108 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

IoT-Based Predictive Machine Health & Safety Monitoring System

[1]Md Hefjur Rahaman, [2]Manisankar Hira, [3]Razia Sultana, [4]Sayani Ghosh

[1,2]UG Student, Department of ECE, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[3,4]Assistant Professor, Department of ECE, Haldia Institute of Technology, West Bengal, India

ABSTRACT

Industrial machines often exhibit warning signs such as excessive temperature and abnormal vibration before failure. This work presents a low-cost IoT-based machine health and safety monitoring system using an ESP32 controller. DHT22 and MPU6050 sensors monitor temperature/humidity and vibration, respectively. The data are processed locally and displayed on a 0.96-inch SSD1306 OLED, while LEDs and a buzzer indicate NORMAL, WARNING, and DANGER conditions. The system also supports Wi-Fi-based remote monitoring and can be extended with relay-based machine shutdown. Prototype readings demonstrate the transition from normal to hazardous conditions, establishing the system as a practical condition-monitoring and early-warning platform with scope for future predictive-maintenance development.

KEYWORDS: IoT, ESP32, Machine Health Monitoring, Predictive Maintenance, Vibration Monitoring, Industrial Safety.

1. INTRODUCTION

Industrial machines are continuously exposed to mechanical, thermal and electrical stresses during operation. Components such as motors, pumps, fans, compressors and hydraulic equipment may gradually develop abnormalities due to overheating, excessive load, vibration, electrical faults, poor lubrication or environmental conditions. If these abnormalities are not detected at an early stage, they can lead to machine failure, production downtime and safety hazards.

Traditional machine inspection generally depends on periodic manual observation or maintenance schedules. Such methods may fail to identify temporary or rapidly developing abnormalities occurring between inspections. Continuous monitoring provides a better approach because machine parameters can be observed while the equipment is operating.

The development of low-cost microcontrollers and IoT technologies has made continuous condition monitoring possible even at laboratory and small-scale levels. The ESP32 is particularly suitable because it provides sensor interfacing, processing capability and built-in wireless connectivity. A machine can continue operating even when its temperature, vibration or current consumption begins to increase. Without continuous monitoring, these changes may remain unnoticed until they develop into a serious fault.

Therefore, the proposed system aims to develop a compact and low-cost monitoring unit capable of observing multiple machine parameters simultaneously and providing an immediate indication when abnormal conditions occur.

2. LITERATURE REVIEW

Jain et al. [1] investigated vibration monitoring of a milling machine using an MPU6050 sensor and Arduino, demonstrating that low-cost MEMS accelerometers can be used to measure machine vibration and support condition assessment.

Arciniegas et al. [2] developed an IoT-based motor-monitoring system using an MPU6050 and ESP32S3, applying vibration analysis and TinyML for detecting abnormal motor behaviour.

Macheso et al. [3] presented a low-cost ESP32-based system with a DHT22 sensor for real-time temperature and humidity monitoring, demonstrating the suitability of ESP32 for continuous sensor-data acquisition.

Supriyanto and Setiawan [4] developed an ESP32-DHT22 monitoring system with an OLED display and remote monitoring capability, showing the practicality of combining sensing, local visualization, and wireless communication.

Alsammarraie et al. [5] further demonstrated an ESP32-based environmental monitoring system integrating DHT22 and gas sensing with OLED and remote communication, highlighting the benefits of multi-parameter monitoring.

*Corresponding Address: hefjurr2@gmail.com
DOI 10.5281/zenodo.22038685

Ash Shiddiqi et al. [6] proposed an ESP32-based industrial monitoring system that combined vibration and temperature sensing with RMS and FFT analysis for condition-based maintenance.

Gomathi and Balaji [7] investigated vibration-based tool-condition monitoring for PCB milling, emphasizing the usefulness of vibration signals for identifying machine-tool conditions. Other related works are also presented in [8], [9], [10]. [11], [12].

These studies demonstrate the effectiveness of low-cost embedded sensors for monitoring individual mechanical and environmental parameters. However, a clear gap remains in integrating vibration, temperature, and humidity sensing with local visualization in a single low-cost ESP32-WROOM-32 platform specifically aimed at early machine-condition warning. The proposed system addresses this gap by integrating the MPU6050, DHT22, and SSD1306 OLED with the ESP32-WROOM-32. The system focuses on multi-parameter condition monitoring and early warning rather than claiming fully validated predictive maintenance, since such performance requires machine-specific baseline data, calibration, controlled fault testing, and labelled datasets.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed system is organized into five functional units: a sensing unit, processing and control unit, display and alert unit, machine interface unit, and IoT communication unit. DHT22 and MPU6050 sensors acquire thermal and mechanical information. The ESP32 receives these signals, processes them against predefined operating conditions, and determines the machine state. The result is shown locally on the OLED and through LEDs and a buzzer, while Wi-Fi provides the foundation for remote monitoring.

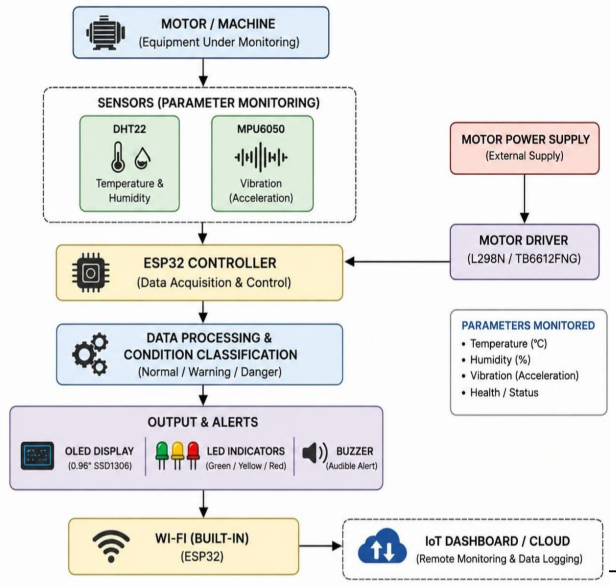


Figure 1: Overall block diagram of the proposed IoT-based machine health monitoring system.

In figure 1, the block diagram shows the signal flow from the monitored machine and sensors to the ESP32 controller, condition classification stage, local alerts, and Wi-Fi/IoT monitoring. It also indicates the separate motor power and driver path used in the prototype.

3.2. Hardware Components

Table I. Hardware Components and Functions

Component	Function / Role
ESP32 DevKit V1	Central controller, sensor acquisition, local logic, and Wi-Fi communication.
DHT22	Temperature and relative-humidity measurement.
MPU6050	3-axis accelerometer and 3-axis gyroscope for vibration/motion monitoring.
0.96-inch SSD1306 I²C OLED	Local display of measurements and machine condition.
Buzzer	Audible warning/alarm.
Green LED	NORMAL condition indication.
Yellow LED	WARNING condition indication.
Red LED	DANGER/fault condition indication.
220 Ω resistors	Current limiting for LEDs.

3.3. Control and Condition Classification Unit

The ESP32 acts as the central processing unit. It collects the sensor readings and compares them with predefined operating conditions. When monitored parameters remain within the defined range, the system reports NORMAL. When one or more parameters enter an abnormal range, the system reports WARNING. Severe or multiple abnormal conditions produce the DANGER state and activate the red indication and buzzer.

3.4. Display, Alert and IoT Communication

The SSD1306 I²C OLED provides local display of sensor values and machine status. A green LED indicates NORMAL, a yellow LED indicates WARNING, and a red LED indicates DANGER or fault. The buzzer provides audible warning, with intermittent operation for WARNING and continuous alarm for DANGER. The ESP32 Wi-Fi interface can transmit collected data to a remote dashboard or cloud platform in an expanded implementation.

4. METHODOLOGY

The operational logic follows a continuous sense–process–classify–alert cycle.

1) **Initialization:** On power-up, the ESP32 initializes the connected sensors, OLED display, alert outputs, and communication functions.

2) **Sensing:** The DHT22 and MPU6050 acquire temperature/humidity and motion/vibration information.

3) **Data Processing:** The ESP32 receives and processes the sensor readings. The measurements are compared with predefined operating conditions.

4) **Condition Classification:** The system assigns NORMAL, WARNING, or DANGER according to the monitored conditions.

5) **Local Alerting:** The OLED displays the current condition. The corresponding LED is activated and the buzzer provides the required audible indication.

6) **IOT Communication:** The ESP32 Wi-Fi interface provides a foundation for transmitting sensor information to a remote monitoring platform. The complete operating sequence can therefore be represented as: Machine → Sensors → ESP32 → Data Processing → Condition Classification → OLED + LED + Buzzer → Wi-Fi/IoT.

5. IMPLEMENTATION

The prototype is assembled around an ESP32 DevKit V1/ESP32-WROOM-32, a small DC motor representing the machine-under-test, and the two sensing modules. The DHT22 and MPU6050 are connected to the ESP32 for acquisition of the required parameters. The motor driver provides the interface between the controller and the laboratory motor, while the OLED is connected through the I²C interface. The OLED and three LEDs provide direct visual feedback, and the buzzer provides an audible warning. The motor power path is kept separate from the ESP32 logic supply through an external motor supply as specified in the prototype design. A breadboard and jumper wires are used for the low-cost laboratory implementation. **The ESP32 processes the acquired sensor data and continuously monitors the operating condition of the motor. The sensed parameters are displayed locally on the OLED for real-time observation. This arrangement enables simple and low-cost evaluation of the prototype under different motor operating conditions.

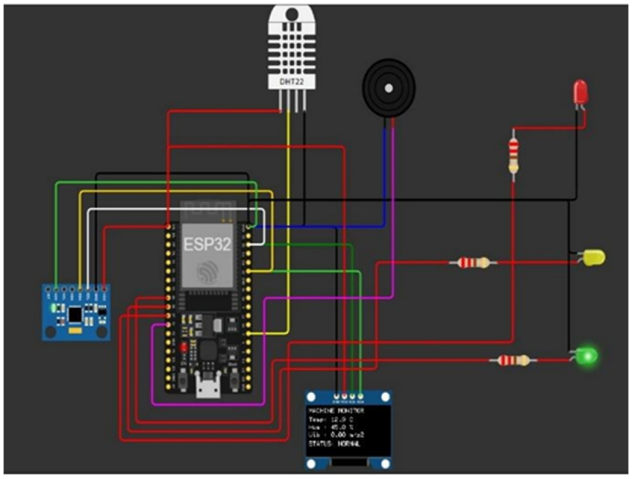


Figure 2: Prototype circuit in NORMAL operating mode.

In figure 2, the NORMAL-state image shows the ESP32, DHT22, MPU6050, OLED, buzzer and three LED indicators interconnected on the prototype. The green indicator is active while the warning and danger indicators remain inactive.

6. RESULTS AND DISCUSSION

The prototype successfully demonstrates three operating states based on temperature and vibration conditions. The OLED displays the corresponding machine status, while the buzzer provides an audible warning when abnormal conditions are detected. The system provides a simple and effective indication of machine health and can be further improved through calibration and controlled testing.

LED Output Indication:

. **Green LED:** NORMAL condition — continuously ON/slow blinking; buzzer OFF.
. **Yellow LED:** WARNING condition — blinking; buzzer gives intermittent beeps.
. **Red LED:** DANGER condition — rapidly blinking/ON; buzzer gives a continuous alarm.

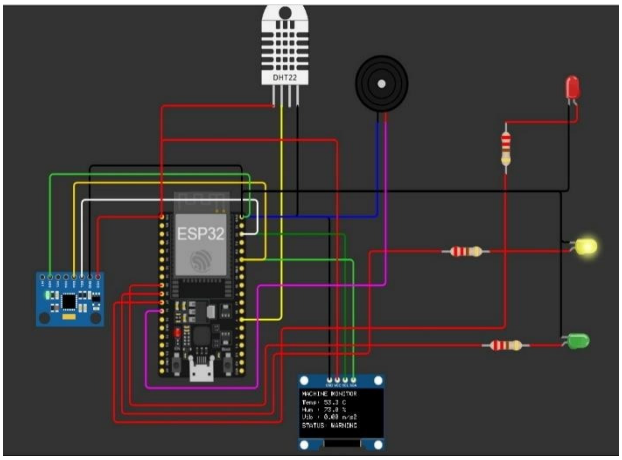


Figure 3: Prototype circuit in WARNING operating mode.

In figure 3, the WARNING-state image shows the same hardware configuration with the yellow indicator activated. This represents an abnormal but non-critical operating condition in the classification logic.

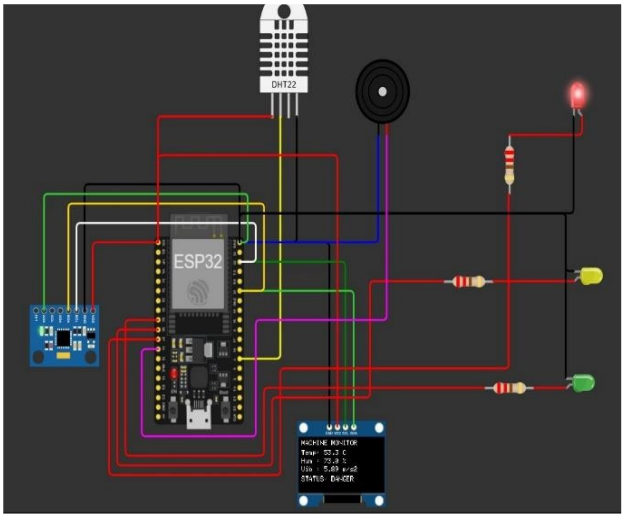


Figure 4: Prototype circuit in DANGER operating mode.

In figure 4, the DANGER-state image shows the red indicator activated, representing a severe or multiple abnormal condition. The buzzer provide the alarm.

7. CONCLUSION

This paper presented a low-cost IoT-based machine health and safety monitoring system using a temperature/humidity ESP32 and with vibration sensing. The system processes sensor data and classifies the machine condition into NORMAL, WARNING, and DANGER states, providing immediate feedback through an OLED display, LEDs, and buzzer.

The prototype demonstrates the effectiveness of multi-parameter condition monitoring as an early-warning mechanism. By continuously observing important machine parameters, the system can help identify abnormal operating conditions at an early stage and reduce the risk of unexpected machine failure or unsafe operation. The proposed approach also provides a foundation for continuous monitoring, remote data acquisition, and future predictive-maintenance applications.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype.

REFERENCES

[1] V. Jain and S. K. Pradhan, "Vibration monitoring of ball nose end mill tool during milling of sculptured surfaces using MPU6050 sensor," *Materials Today: Proceedings*, vol. 27, pp. 2477–2486, 2020, doi: 10.1016/j.matpr.2019.09.222.

[2] S. Arciniegas, D. Rivero, J. Piñan, E. Diaz, and F. Rivas, "IoT device for detecting abnormal vibrations in motors using TinyML," *Discover Internet of Things*, vol. 5, art. no. 41, 2025, doi: 10.1007/s43926-025-00142-4.

[3] P. Macheso, P. Chombo, and M. M. Nyirenda, "Design of standalone asynchronous ESP32 web-server for temperature and humidity monitoring," *International Journal of Computer Applications*, 2021.

[4] Supriyanto and N. D. Setiawan, "Rancang bangun sistem real-time monitoring suhu dan kelembaban menggunakan ESP32," *Elkom: Jurnal Elektronika dan Komputer*, vol. 18, no. 1, pp. 447–457, 2025, doi: 10.51903/elkom.v18i1.3064.

[5] M. J. A. Alsammarraie, A. H. Ali, A. A. Al-Hilali, H. Q. M. Monir, M. S. Jabbar, and H. B. Qasim, "Implementation of a long range-based monitoring system for environmental remote safety schemes," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 32, no. 3, pp. 1462–1475, 2023.

[6] A. Ash Shiddiqi, L. S. Wasitova, and D. H. Nugroho, "IoT-based real-time vibration and temperature monitoring system for industrial machinery using ESP32 and MQTT," *International Journal of Engineering Continuity*, vol. 5, no. 1, pp. 53–71, 2026, doi: 10.58291/ijec.v5i1.519.

[7] K. Gomathi and A. Balaji, "Tool condition monitoring of PCB milling machine based on vibration analysis," *Materials Today: Proceedings*, 2020.

[8] N. Tandon and A. Choudhury, "A review of vibration and acoustic measurement methods for the detection of defects in rolling element bearings," *Tribology International*, vol. 32, no. 8, pp. 469–480, 1999.

[9] Espressif Systems, *ESP32-WROOM-32 Datasheet*, version 3.6, Espressif Systems, 2025.

[10] TDK InvenSense, *MPU-6000 and MPU-6050 Product Specification*, document no. PS-MPU-6000A-00, rev. 3.4.

[11] Aosong Electronics, *AM2302/DHT22 Digital Relative Humidity & Temperature Sensor*, datasheet.

[12] Solomon Systech, *SSD1306 128 × 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller*, datasheet.

.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 109-116 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

SELF BALANCING BOT USING ARDUINO UNO AND MPU6050

[1]Mayank Vatsa, [2]Nimisha Nisha, [3]Pushpanjay Gaurav, [4]Priyanshu Kumar Gupta, [5]Kamanashis Goswami

1,2,3,4UG student, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, ,West Bengal.
5Assistant Professor, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, ,West Bengal.

ABSTRACT

Two-wheeled self-balancing robots are compact mobile robotic systems based on the principle of an inverted pendulum. Unlike conventional four-wheeled robots, a two-wheeled balancing robot is inherently unstable and must continuously sense its orientation and generate corrective motor commands to remain upright. This makes it an effective experimental platform for studying feedback control, embedded systems, inertial sensing, and robotics. This paper presents the design and implementation of a low-cost two-wheeled self-balancing robot using an Arduino UNO, MPU6050 IMU, L298N motor driver, DC geared motors, and a 7.4-V battery. The MPU6050 measures acceleration and angular velocity to estimate the robot's pitch angle, which is compared with a setpoint of 176 and controlled using a PID controller with Kp=100, Ki=140, and Kd=0.8. The controller generates PWM motor commands to maintain balance, using DMP-based orientation processing and a 10-ms sampling interval. The study demonstrates the feasibility of dynamic balancing with low-cost hardware and discusses sensor calibration, PID tuning, limitations, applications, and future improvements such as encoder feedback, Kalman filtering, and LQR control.

KEYWORDS: Self-Balancing Robot, Two-Wheeled Robot, Feedback Contro, Embedded Robotics.

1. INTRODUCTION

Two-wheeled self-balancing robots are an important and interesting area of robotics and control engineering. Unlike conventional robots with four or more wheels, a two-wheeled robot must continuously maintain its own balance while operating. Its structure can be considered similar to an inverted pendulum, where even a small change in the tilt angle can cause the robot to become unstable and fall. Therefore, the system needs to continuously sense its orientation and make quick corrective movements to maintain stability [2], [3].Self-balancing robots have potential applications in personal transportation, surveillance, warehouse automation, service robotics, and educational robotics. They are also widely used as experimental platforms for studying and comparing control techniques such as PID, LQR, fuzzy control, and other advanced methods [2], [3], [5].In this project, an Arduino UNO is used as the main controller because of its low cost, simple architecture, and ease of programming. The Arduino UNO is based on the ATmega328P microcontroller and provides sufficient digital, analog, and PWM interfaces for controlling the prototype [1]. An MPU6050 IMU is used to measure the robot's motion and tilt by combining data from a three-axis accelerometer and a three-axis gyroscope. The sensor data is used to estimate the robot's tilt angle, which is then provided to a PID controller. The controller calculates the required corrective action and adjusts the motor speed and direction to bring the robot back toward its upright position [4], [5].

2. LITERATURE REVIEW

Villacrés et al. [2] compared PID, LQR, and Sliding Mode Control (SMC) for stabilizing a two wheeled inverted-pendulum robot highlighting

*Corresponding Address: nimishanisha1205@gmail.com DOI 10.5281/zenodo.22038738

the importance of suitable feedback-control techniques for maintaining stability.

Bakiya Lakshmi et al. [3] investigated PID and LQR-base control for a two-wheel self-balancing robot, demonstrating the application of classical and optimal control techniques for balancing.

Abdelgawad et al. [4] presented a model- and data-based control approach using Arduino for self-balancing robots demonstrating, the suitability of low-cost embedded platforms for practical control experiments.

Putra et al. [5] and Nam [6] investigated LQR and robust PI-based control approaches for two-wheeled self-balancing and inverted-pendulum robots, focusing on stability and control performance.

Krishna et al. [7] explored PID combined with deep reinforcement learning, while Hassan et al. [8] investigated MPU6050-based sensing with PID control for self-balancing robots. These studies demonstrate the continued development of both conventional and advanced control approaches for self-balancing systems.

The present work focuses on a low-cost self-balancing robot using Arduino UNO, MPU6050, L298N motor driver, DC geared motors, and PID-based closed-loop control.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed robot consists of four major subsystems: sensing, processing and control, motor driving, and mechanical and power systems. The MPU6050 continuously senses the robot's tilt and provides the required motion data to the Arduino UNO for orientation estimation and error calculation. The calculated error is processed by the PID controller to generate an appropriate control signal, which is converted into PWM output and given to the L298N motor driver. The driver controls the DC motors to move the wheels in the required direction and correct the robot's tilt. If the robot tilts forward, the wheels move forward, while a backward tilt causes the motors to rotate in the opposite direction. This continuous sensing, error calculation, control, and correction process maintains the robot in an upright position and forms a closed-loop feedback control system.

3.2. Hardware Implementation

The Arduino UNO serves as the central controller of the self-balancing robot and is responsible for MPU6050 initialization, sensor-data acquisition, pitch-angle processing, PID computation, PWM generation, motor direction control, and serial monitoring. The MPU6050 IMU is used as the primary feedback sensor, combining a three-axis accelerometer and gyroscope to measure acceleration and angular velocity. The pitch angle is particularly important for detecting the forward and backward tilt of the robot. The MPU6050 communicates with the Arduino UNO through the I²C interface, with SDA connected to A4, SCL to A5, and INT to D2.An L298N dual H-bridge motor driver is used to control the two geared DC motors. Its control inputs are connected to D6, D9, D10, and D11 of the Arduino UNO for motor direction and PWM speed control. The motors provide corrective motion to maintain the robot in an upright position and can also produce differential motion for steering. The complete prototype is powered by a 7.4-V battery, while proper power distribution and regulation are used to minimize voltage fluctuations and ensure stable operation of the Arduino, MPU6050, and motors.

Table 1. Hardware components and function

Components	Function
Arduino UNO	Main processing and control unit
MPU6050 IMU	Measures acceleration and angular velocity for orientation estimation
L298N Motor Driver	Controls direction and speed of DC motors
DC Geared Motors	Drive the left and right wheels and provide corrective motion
7.4-V Battery	Provides power to the robot
Two-Wheel Chassis	Provides mechanical support for the robot

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

3.3. Block Diagram

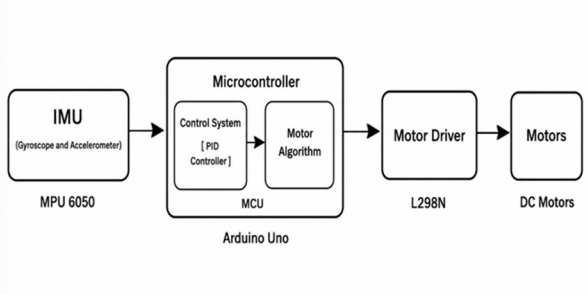


Figure 1: Block diagram of the Self balancing Robot

3.4. Circuit Diagram

The connections for this Arduino-based self-balancing robot are pretty simple. This is a self-balancing robot using Arduino and MPU6050, so we need to interface the MPU6050 with Arduino and connect the motors through the motor driver module. The whole setup is powered by a 7.4V Li-ion battery. The circuit diagram for the same is shown below.

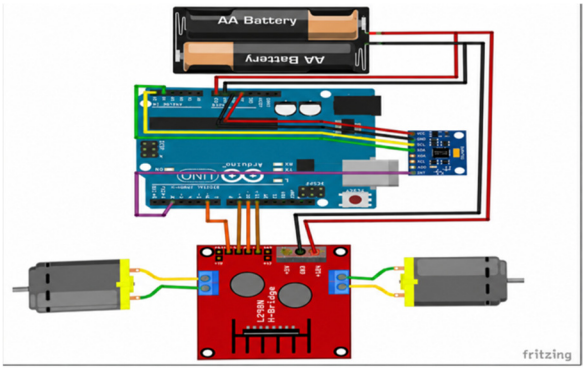


Figure 2: Circuit Diagram of the proposed Self balancing robot

3.5. Pin Configuration

Table2. Pin configuration of the proposed system

Component	Pin	Arduino UNO
MPU6050	VCC	+5V*
MPU6050	GND	GND
MPU6050	SCL	A5
MPU6050	SDA	A4
MPU6050	INT	D2
L298N	IN1	D6
L298N	IN2	D9
L298N	IN3	D10
L298N	IN3	D11

4. MATHEMATICAL MODEL OF THE SYSTEM

The two-wheeled inverted pendulum (TWIP) representation is used to model the dynamics of the self-balancing robot. Unlike a simple cart-pole system, the dynamics of a two-wheeled balancing robot are more complex because wheel rotation, body motion, motor torque, wheel inertia, friction, and non-slip constraints are strongly coupled. Therefore, the nonlinear dynamics are commonly linearized around the upright equilibrium before designing controllers such as PID or LQR. The robot can be approximated as an inverted pendulum mounted on two driven wheels, where M represents the mass of the robot body, m represents the equivalent wheel and motor mass, l is the distance between the wheel axle and the centre of mass, I is the moment of inertia of the robot body, r is the wheel radius, g is the gravitational acceleration, x is the horizontal displacement, θ is the body tilt angle, and u is the motor control input. The upright equilibrium position is represented by $\theta = 0$, and for small disturbances, where $|\theta| \ll 1$, the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$ can be applied to simplify the mathematical model. The desired reference angle is generally taken as $\theta(ref) = 0$, meaning that the robot should remain perfectly upright. The controller continuously calculates the error between the desired and measured angles as $e(t) = \theta(ref) - \theta(t)$ and generates a corrective motor command whenever the robot deviates from its upright position, causing the wheels to move in the appropriate direction to restore and maintain balance[2].

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

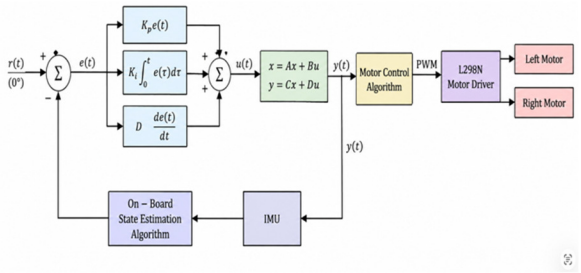


Figure 3: Mathematical model and PID control block diagram

5. SYSTEM IMPLEMENTATION

5.1. MPU6050 Calibration

Calibration is a critical stage in the operation of a self-balancing robot because the MPU6050 may not produce exactly zero readings even when the robot is stationary due to sensor bias, manufacturing tolerances, mechanical mounting errors, temperature variations, and electrical noise. To obtain accurate orientation measurements, the robot is first placed in its intended upright equilibrium position and kept completely stationary. The MPU6050 is then initialized, and the accelerometer and gyroscope readings are measured to determine their respective offsets. These offset values are stored and used as reference values during normal operation. The corresponding pitch reference is also determined so that the controller can accurately identify deviations from the upright position. In addition, the implementation uses the MPU6050's Digital Motion Processor (DMP) to process motion data and obtain orientation information from quaternion and gravity-vector data. Proper calibration ensures that the measured tilt angle is reliable, allowing the PID controller to generate accurate corrective commands and significantly improving the stability and balancing performance of the robot.

5.2. PID Control

The PID (Proportional–Integral–Derivative) controller is used to generate the corrective motor command required to maintain the robot in an upright position. It continuously compares the desired reference angle with the measured pitch angle from the MPU6050 and calculates the error between them. The continuous-time PID equation is $u(t) = Kp\, e(t) + Ki \int_0^t e(\tau)d\tau + Kd\, de(t)/dt$, where Kp is the proportional gain, Ki is the integral gain, and Kd is the derivative gain. The proportional term, given by $uP = Kp\, e(t)$, produces a corrective response proportional to the instantaneous error; a larger tilt results in a stronger motor correction, while a very low Kp can make the robot respond slowly and a very high Kp can cause excessive oscillations. The integral term, given by $uI = Ki \int e(t)dt$, accumulates the error over time and helps eliminate persistent or steady-state errors. However, excessive integral gain can lead to integral windup, resulting in instability and large oscillations, particularly because a self-balancing robot is a rapidly changing system. The derivative term, given by $uD = Kd\, de(t)/dt$, responds to the rate of change of the error and helps predict the robot's movement. A properly tuned derivative gain provides damping, reduces oscillations, and improves the overall stability and response of the balancing system. By combining these three terms, the PID controller continuously adjusts the motor speed and direction to bring the robot back toward its upright reference position.

Fig.ure 4: Robot prototype used for PID tuning

5.3. PID Parameters used in the prototype

The supplied implementation uses the PID gains Kp = 100, Ki = 140, and Kd = 0.8, with a setpoint of 176°. These values are the specific parameters documented in the actual project code and are used by the controller to determine the corrective motor response required to maintain the robot in

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

its balanced position. The proportional gain determines the immediate response to tilt error, the integral gain compensates for accumulated error, and the derivative gain provides damping by responding to the rate of change of the error. However, these PID gains are implementation-specific and should not be considered universal values for all self-balancing robots. The appropriate values depend on several factors, including the robot's mechanical geometry, total mass, motor characteristics, battery voltage, wheel diameter, center-of-mass position, sensor mounting, and overall mechanical construction. Therefore, these parameters generally require practical tuning and optimization when applied to a different robot design.

5.4. PID Tuning Procedure

The project follows a manual tuning method for determining suitable PID gains. Initially, all three gains are set to zero, i.e., Kp = Ki = Kd = 0, so that the controller produces no corrective response. The proportional gain (Kp) is then increased gradually until the robot begins to produce a meaningful corrective response to changes in its tilt angle. Once an appropriate proportional response is achieved, the integral gain (Ki) is introduced and gradually adjusted to reduce persistent or steady-state errors and improve the robot's ability to maintain its desired equilibrium position. Finally, the derivative gain (Kd) is increased carefully to reduce rapid oscillations and provide additional damping to the system. This step-by-step tuning approach allows the controller to achieve a balance between fast corrective response, low steady-state error, and stable operation without excessive oscillation. Because balancing occurs continuously, tuning should be performed carefully with the robot physically supported during initial tests. Because balancing occurs continuously, tuning should be performed carefully with the robot physically supported during initial tests.

Table 3. PID tuning procedure and objectives

Stage	Parameters	Objectives
1	Ki = Kd = 0	Establish proportional response
2	Increase Kp	Obtain fast correction
3	Add Ki	Reduce steady-state error
4	Add Kd	Reduce oscillation
5	Fine tune all	Improve stability

6. CONTROL ALGORITHM AND SYSTEM IMPLEMENTATION

6.1. Control Algorithm

The control algorithm of the self-balancing robot begins with the initialization of the Arduino UNO, followed by the initialization of the MPU6050 sensor through the I²C interface. The system first verifies proper sensor communication and then initializes the Digital Motion Processor (DMP) for orientation data processing. After this, the calibrated sensor offset values are applied, and the equilibrium setpoint is established according to the desired upright position of the robot. The PID parameters Kp, Ki, and Kd are then configured. During normal operation, the system continuously acquires the robot's orientation data from the MPU6050 and calculates the control error using $e(t) = \text{Setpoint} - \text{Input}$. The PID controller processes this error to generate the corrective control output according to $u = Kp\, e + Ki\int e\, dt + Kd(de/dt)$. Based on the magnitude and sign of the PID output, the Arduino determines the appropriate motor direction and generates corresponding PWM signals to control the speed of the left and right motors through the L298N driver. This entire process is repeated continuously at a high frequency, allowing the robot to detect changes in its tilt and immediately move the wheels in the required direction to restore and maintain its upright balance.

6.2. Software Implementation

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

The program uses the I2Cdev**,** PID_v1, and MPU6050 MotionApps20 libraries to interface with the MPU6050, implement the PID control algorithm, and process orientation data using the sensor's Digital Motion Processor (DMP). The implementation defines the MPU6050 sensor object and PID controller and configures the controller in automatic mode with a 10-ms sample time, allowing the system to update the control response at regular intervals. The PID output is limited to a range of −255 to +255, which corresponds to the typical 8-bit PWM range used in Arduino-based motor-control applications. The positive and negative values of the PID output determine the required motor direction, while the magnitude determines the PWM duty cycle and consequently the motor speed. This software configuration enables the Arduino to continuously process the robot's orientation, calculate the required corrective action, and control the motors to maintain the desired upright position.

6.3. Motor Control

The PID controller decides the direction and speed of the motors based on the robot's tilt. If the robot starts falling forward, the motors move forward to bring the wheels under the centre of mass and restore balance. Similarly, if it tilts backward, the motors rotate in the opposite direction to correct the tilt. The motor speed is continuously adjusted according to the PID output, so larger tilts produce stronger corrections. When the robot is close to the upright position, the motors stop or run at very low speed. This continuous adjustment helps the robot maintain its balance during operation.

6.4. Closed-Loop Working

The self-balancing robot works through a continuous closed-loop feedback system. The MPU6050 measures the robot's motion and provides its pitch angle, which is compared with the desired upright position to calculate the error. The PID controller processes this error and determines the required motor direction and speed. The Arduino then sends PWM and direction signals to the L298N motor driver, which controls the DC motors to correct the robot's tilt. The updated position is measured again by the MPU6050, and the process repeats continuously. This allows the robot to quickly respond to disturbances and maintain its balance.

7. EXPERIMENTAL IMPLEMENTATION

The physical prototype consists of a two-wheel chassis with two geared DC motors, an Arduino UNO, MPU6050 IMU sensor, L298N motor driver, and battery supply mounted on a vertical frame. The motors provide both balancing and movement, while the Arduino processes the orientation data from the MPU6050 and generates corrective motor-control signals. The robot's balancing performance depends on the interaction between its mechanical and control systems. Factors such as centre-of-mass height, wheel radius, chassis weight, motor torque, friction, sensor mounting, and frame rigidity significantly affect stability and response. Hence, effective self-balancing requires proper mechanical construction along with accurate sensor processing and PID tuning.

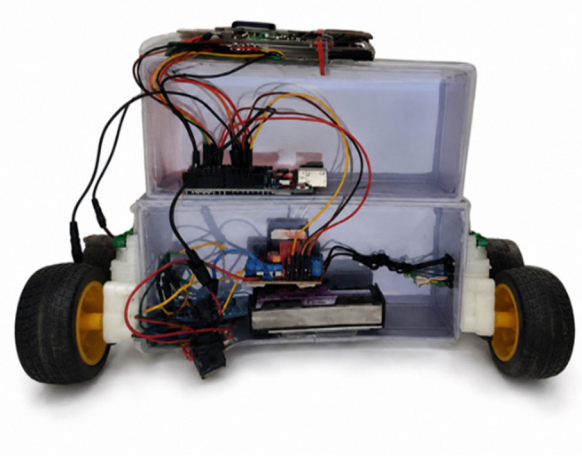


Figure 4: .Fabricated prototype of the self-balancing robot

8. RESULTS AND DISCUSSION

The prototype successfully demonstrates closed-loop balancing using the Arduino UNO,

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

MPU6050 IMU, and PID controller. The robot detects pitch-angle deviations through continuous sensor feedback and generates corrective motor commands to maintain its upright position. As sufficient numerical experimental data are unavailable, parameters such as settling time, overshoot, and steady-state error are not reported. The balancing performance depends mainly on proper PID tuning, sensor accuracy, motor response, battery voltage, and mechanical factors such as backlash and wheel-floor friction. Overall, stable operation requires proper coordination between the control system, sensors, motors, and mechanical structure.

Figure 5: Experimental prototype of the self-balancing robot

9. CONCLUSION

The paper successfully demonstrates a low-cost two-wheeled self-balancing robot using an Arduino UNO, MPU6050, L298N motor driver, DC motors, and PID control. The system uses real-time feedback to detect tilt and generate corrective motor actions for maintaining balance. The prototype demonstrates the practical application of PID-based control in an unstable robotic system and provides a foundation for future improvements such as better sensor fusion, wheel encoders, and improved motor control.

ACKNOWLEDEMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype

REFERENCES

[1] Arduino, “Arduino UNO R3 – Technical Specifications,” Arduino Documentation.

[2] J. Villacrés, M. Viscaíno, M. Herrera, and O. Camacho, “Controllers Comparison to Stabilize a Two-Wheeled Inverted Pendulum: PID, LQR and Sliding Mode Control,” 2016.

[3] R. Bakiya Lakshmi, S. M. Bashir, and G. S. Daniel, “Design, Simulation and Optimal Control of Two-Wheel Self-Balancing Robot Using LQR and PID,” Journal of Propulsion Technology, vol. 45, no. 3, 2024.

[4] A. Abdelgawad, T. Shohdy, and A. Nada, “Model- and Data-Based Control of Self-Balancing Robots: Practical Educational Approach with LabVIEW and Arduino,” 2024.

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

[5] G. B. Putra, F. Wahab, and T. A. Tamba, "Design and Implementation of Linear Quadratic Regulator Control for Two-Wheeled Self-Balancing Robot," Bulletin of Electrical Engineering and Informatics, vol. 14, no. 2.

[6] N. H. Nam, "Control of Two-Wheeled Inverted Pendulum Robot Using Robust PI and LQR Controllers," Journal of Military Science and Technology, 2020.

[7] G. S. Krishna, D. Sumith, and G. Akshay, "Epersist: A Self Balancing Robot Using PID Controller and Deep Reinforcement Learning," arXiv, 2022.

ISSN: 0973-6875 DOI 10.5281/zenodo.22038738

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 117-124 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Edge-First LoRa Water Quality Monitoring for Smart Aquaculture

[1]Shivans Sankalp, [2]Samridhi Sinha, [3]Jagannath Samanta, [4]Ritwik Haldar

[1]UG Student, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur – 721657, West Bengal, India
[2]UG Student,Department of Artificial Intelligence & Machine Learning, Haldia Institute of Technology, Haldia, Purba Medinipur – 721657, West Bengal, India
[3,4]Associate Professor, Department of Electronics & Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur – 721657, West Bengal, India

ABSTRACT

Aquaculture now supplies more than half of the aquatic-animal protein consumed worldwide, yet many small-scale farms still rely on manual water-quality measurements taken once or twice a day. These gaps can delay intervention until stock is already stressed. This paper presents a low-cost point-to-point LoRa telemetry system that continuously monitors water temperature and pH, evaluates readings on the receiving node using a deterministic and explainable assessment engine, and displays the results through a node-hosted dashboard. The dashboard provides live telemetry, trends, statistics, grading rationale and data export, operating entirely offline through the receiver's Wi-Fi access point without cloud or internet connectivity. A prototype using ESP32 and SX1278 modules was evaluated over 9 h 30 min, receiving 17,043 packets with 99.99% delivery at a mean RSSI of −52 dBm. Complete firmware for sensing, inference, logging and the web server occupies 68 kB of flash memory.

KEYWORDS: Aquaculture, LoRa, Internet of Things, edge inference, water quality monitoring.

1. INTRODUCTION

The Food and Agriculture Organization estimates that aquatic organisms currently supply about 52 % of the world's aquatic animal food, and projects that this share will exceed 60 % by 2030 as capture fisheries reach their sustainable limit [1]. Meeting that demand requires more intensive farming, which in turn requires closer control of water conditions than has so far been the norm. Temperature governs metabolism, feeding response and immunity, while pH regulates ionic balance and the toxicity of nitrogenous waste [8]. A deviation in either variable, whether gradual or sudden, disturbs the pond ecosystem and places the stock under stress.

The prevailing practice on small farms is a manual check once or twice per day, which creates three difficulties. First, the interval between readings is far longer than the timescale on which water quality actually changes. Second, manual sampling does not scale economically to the number of ponds a single enterprise operates. Third, by the time an operator observes a deviation, the damage to the stock has usually already occurred. Low-Power Wide-Area Networks are well matched to this problem, and LoRa in particular has been shown to deliver kilometre-scale range at microamp-level node current [2], [3]. Most reported deployments, however, forward raw measurements to a remote server and leave both interpretation and presentation to that server, which is precisely the part of the chain least likely to be available on a rural farm.

The work presented here takes the opposite approach: acquisition, classification and visualisation are all performed on the receiving node, and no cloud service participates in the measurement path. An end-to-end LPWAN telemetry chain was implemented using commodity ESP32 and SX1278 modules, together with a lightweight explainable algorithm that converts temperature and pH readings into an

*Corresponding Address: shivanssankalp1@gmail.com DOI 10.5281/zenodo.22038788

interpretable water-quality grade. The receiver also hosts a web application offering real-time monitoring, graphical trend analysis, data export, statistics and inspection of the grading rationale, all while disconnected from the internet. The prototype was then evaluated in terms of link reliability, memory footprint and runtime performance.

2. LITERATURE REVIEW

Adelantado et al. [1] examined the operating limits of LoRaWAN, showing how duty-cycle regulation, spreading-factor selection and collision behaviour bound the achievable throughput and node density in a deployment. Their analysis establishes why a low-duty-cycle telemetry application of the kind considered here fits comfortably within the technology's constraints, while high-rate or bidirectional applications do not.

Liando et al. [2] reported a large-scale measurement study of LoRa covering range, packet reception and the interaction between spreading factor and interference under field rather than laboratory conditions. Their results provide the empirical basis for treating kilometre-scale point-to-point links as reliable at moderate spreading factors, which motivates the SF9 setting adopted in this work.

Nguyen et al. [3] developed a LoRa-based sensor network for real-time monitoring of shrimp pond water quality in the Mekong Delta, demonstrating range of the order of several kilometres with node current drain in the microamp range. The work confirms the suitability of LPWAN telemetry for pond environments, but forwards raw measurements without any on-site interpretation.

Kodali and Mahesh [4] implemented a low-cost LoRa gateway bridged to an MQTT broker for smart aquaculture, publishing pond telemetry to cloud subscribers. The design illustrates the prevailing architecture in this domain, in which the gateway performs no analysis and the farmer's view of the pond depends entirely on the availability of the broker and the backhaul link.

Callebaut et al. [5] characterised LoRa point-to-point path loss through measurement campaigns over water, and found that surface reflections produce pronounced constructive and destructive lobes at separations no greater than the width of a single pond. This is a significant practical consideration for in-pond deployments, since antenna placement rather than link budget may determine whether a node is reachable.

Horton [6] introduced the water-quality index, aggregating several measured parameters into a single rated value, and Boyd [8] documented the physiological tolerances of cultured species together with the interaction between temperature, pH and ammonia toxicity. Together these establish both the precedent for reducing multiple parameters to one interpretable score and the domain basis for the optimal windows and the temperature–pH interaction term used in Section V.

Two limitations recur across the reviewed systems. Most implementations are tied to a cloud backend, and none report what remains available to the farmer when the backhaul link fails. Where machine learning is applied, it operates on pooled data at the server, so neither the inference step nor any presentation layer exists at the pond itself. The system proposed in this paper addresses both gaps by performing classification and presentation entirely on the receiving node, retaining full functionality in isolation from the cloud while keeping the assessment logic transparent and reproducible by hand.

Automated aquaculture monitoring was first implemented in the early 2000s using wired instrumentation in land-based recirculating systems, where the cost of cabling was justified by the value of the stock held in a small volume. The move to IEEE 802.15.4 wireless links removed the cabling cost but introduced a range limitation, which was in turn compensated by the additional complexity of a mesh topology. Single-board controllers running Arduino- or Raspberry Pi-class firmware later offered cellular-connected alternatives, constrained by the power draw and recurring cost of the cellular modem.

The arrival of LoRa after 2016 renewed interest in long-range, low-power monitoring of aquaculture sites [2], [12]. A measurement study across a large deployment established the practical range and reliability limits of the technology under field conditions [3]. Nguyen et al. reported a sensor network over shrimp ponds in the Mekong Delta, with range estimates of the order of several kilometres and node current drain

in the microamp range [4]. Kodali and Mahesh implemented a LoRa gateway bridged to an MQTT broker, publishing pond telemetry to a cloud subscriber [5]. A propagation study conducted over open water found that reflections from the water surface create pronounced constructive and destructive lobes at separations no larger than a single pond, which is a significant consideration for any in-pond LoRa deployment [6].

Two limitations recur across this literature. Most implementations are tied to a cloud backend, and none report what the farmer is left with when the backhaul link fails. Where machine learning is applied, it operates on pooled data at the server, so neither the inference nor any presentation layer exists at the pond itself. The system described here sits between these gaps: it classifies and presents data entirely on-device and remains fully functional in isolation from the cloud.

3. SYSTEM ARCHITECTURE

The system is organised in four layers around the data path shown in Fig. 1. Each layer is self-contained, so that a malfunction in a higher layer does not disrupt the layers beneath it.

3.1. Perception Layer

The transmitting node uses a DS18B20 digital temperature sensor on the 1-Wire interface and a SEN0161 glass-electrode pH probe [10]. The 0–3.3 V analogue output of the probe is digitised by the ESP32's built-in SAR ADC. Both measurements are then packed as IEEE-754 single-precision floats into a fixed-size record for transmission.

3.2. Transport Layer

Physical transport uses an SX1278 433 MHz module [9]. Each record is sent point-to-point at spreading factor 9, 125 kHz bandwidth and 4/5 coding rate with CRC enabled, chosen to balance link budget against airtime. No network server, gateway or join procedure is used; the topology is deliberately reduced to a single transmitter and single receiver per farm.

3.3. Processing Layer

The receiving node unpacks each record, applies calibration corrections, runs the assessment algorithm described in Section V, and persists the timestamped result to non-volatile storage without requiring any higher-level service.

3.4. Presentation Layer

The receiving node hosts its own access point and serves a complete web analytics application. This is the sole source of trend information, statistics and grading explanation in the system, which by design does not use the cloud for either storage or presentation.

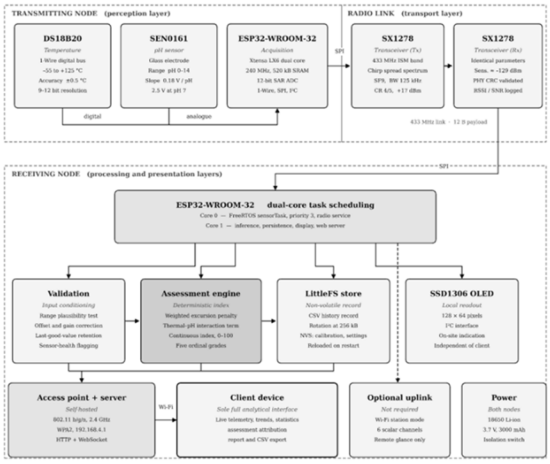


Figure 1: System block diagram, showing operating parameters for the sensing elements, the radio link and the processing and presentation stages. No part of the system depends on infrastructure beyond the two nodes.

4. HARDWARE REALISATION

Components were selected for low cost, availability, ease of use and reproducibility: an ESP32-WROOM-32 microcontroller [11], an SX1278 (Ra-02) LoRa module, a DS18B20 temperature sensor, a SEN0161 pH sensor, an SSD1306 OLED display for local indication, and a 3.7 V, 3000 mAh 18650 Li-ion cell. These are summarised in Table I.

The DS18B20 provides waterproof digital temperature measurement with ±0.5 °C accuracy [10]. For the SEN0161, a millivolt transfer function was derived from the sensor characteristic and stored in non-volatile memory, from where it can be updated through the dashboard to account for electrode ageing and production tolerance. The analogue output of the pH sensor is connected to ADC1, since ADC2 is unavailable while the Wi-Fi radio is active.

Table II lists the SX1278 link settings. The same settings must be applied at both ends of the link: a mismatch in spreading factor, bandwidth or

coding rate leaves the receiving SX1278 unable to demodulate the incoming signal. Because point-to-point operation has no discovery phase, the result of such a configuration error is not an error message but simply silence, which makes the parameters in Table II the first thing to verify when a link fails to establish.

Table I. System Hardware Inventory

Element	Device	Interface / Principle
Processor	ESP32-WROOM-32	Dual-core Xtensa LX6, 240 MHz
Radio	SX1278 (Ra-02)	Chirp spread spectrum, 433 MHz
Temperature	DS18B20	Digital bandgap, 1-Wire
pH	SEN0161	Glass electrode, analogue
Display	SSD1306 OLED	128 × 64, I²C
Supply	18650 Li-ion, 3.7 V	Rechargeable, 3000 mAh

Table II. LORA Link Parameters

Parameter	Value	Rationale
Carrier frequency	433 MHz (ISM)	Licence-exempt, good diffraction
Spreading factor	SF9	Range / airtime compromise
Bandwidth	125 kHz	Standard channel width
Coding rate	4/5	Lowest FEC overhead
TX power	+17 dBm	Within ISM limit
CRC / Payload	Enabled / 12 B	Rejects corrupt frames; fixed 3 × float32 record
Topology / Duplex	Point-to-point / half, uplink only	Single receiver per farm; no downlink needed

5. ON-NODE WATER QUALITY ASSESSMENT

The conventional approach of applying independent alarm thresholds to temperature and pH ignores the fact that the two variables interact. A given deviation from optimum pH is more or less dangerous depending on temperature, because temperature governs two critical processes: the conversion of ammonia to its toxic un-ionised form, and the quantity of dissolved oxygen the water can hold [8]. The assessment module therefore considers temperature and pH jointly.

5.1. Excursion Penalty and Quality Index

For a measured value v against an optimal window [lo, hi], the excursion e(v) is the distance by which the reading lies outside that window, and is zero whenever the value lies inside it, as defined in (1).

$$e(v) = max(0,\ lo - v,\ v - hi) \quad (1)$$

Writing e_T and e_p for the temperature and pH excursions respectively, the penalty P is given by (2) and the quality index Q by (3).

$$P = w_T \cdot e_T^{\alpha T} + w_p \cdot e_p^{\alpha p} + k \cdot e_T \cdot e_p \quad (2)$$

$$Q = max(0,\ 100 - P) \quad (3)$$

with Q clipped to the interval [0, 100]. The optimal windows are 26–30 °C for temperature and 6.8–8.2 for pH, drawn from established tolerances for warm-water finfish culture [8]. Because the exponents α_T and α_p exceed unity, a single large excursion is penalised more heavily than several small excursions of the same total magnitude, reflecting that one four-degree deviation is more harmful to the stock than four separate one-degree deviations. The cross term $k \cdot e_T \cdot e_p$ activates only when both variables are simultaneously out of range, capturing the compounded thermal and acid–base stress described above. The coefficient values used are listed in Table III.

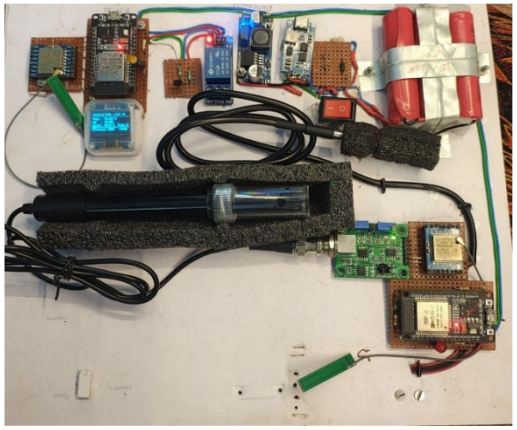


Figure 2: Hardware prototype

5.2. Ordinal Grading, Confidence and Attribution

The continuous index Q is mapped onto five ordinal grades (Table IV) so that an operator is given a summary that is easy to remember, while the underlying value remains available on the dashboard for closer inspection or for export and later analysis. The thresholds and coefficients are

published for every grade, so that any individual assessment can be reproduced by hand.

Table III. Assessment Engine Coefficients

Symbol	Quantity	Value
$[lo, hi]_T$	Optimal temperature window	26.0 – 30.0 °C
$[lo, hi]_p$	Optimal pH window	6.8 – 8.2
w_T, w_p	Temperature / pH penalty weights	4.5 per °C / 18.0 per unit
α_T, α_p	Temperature / pH exponents	1.25 / 1.35
k	Interaction coefficient	0.9

Each grade carries a confidence value representing the normalised distance to the nearest threshold, reaching 1 at the centre of a band and 0.5 at its edges. This is not a probability; it indicates only how far the reading lies from a decision boundary. Since the instrument accuracy of 0.5 °C, together with a repeatability error of roughly 0.1 pH after two-point calibration, can easily place a reading on the wrong side of a threshold, a borderline result is deliberately reported in the worse grade, with the confidence value indicating the margin. This is the conservative bias appropriate to the application: the cost of a false alarm is far lower than the cost of undetected poor water quality.

In addition to the grade, the system compares the weighted penalty contribution of each parameter, rather than the raw excursions, to determine which factor dominates the result and in which direction, and therefore which countermeasure to recommend: reduced feeding and a partial water change for an alkaline drift, or additional pre-dawn aeration for temperature climbing too quickly. The computation is entirely deterministic and requires no training set or lookup table. It executes in a few microseconds from a program of under 200 bytes on a 240 MHz core.

6. LOCAL INTERFACE AND OPTIONAL UPLINK

6.1. Locally Hosted Analytical Server

Because rural sites frequently lack both broadband and reliable cellular coverage, the entire dashboard is hosted on the receiving node rather than in the cloud. Any smartphone able to join the access point published by the node reaches the dashboard directly at 192.168.4.1, removing any dependence on external infrastructure. The dashboard uses WebSockets for live telemetry, charts and statistics, and a RESTful HTTP API to query and modify configuration values and to retrieve statistical summaries and grading rationale reports. Fig. 2 shows the live view.

Table IV. Index Thresholds and Grade Assignment

Grade	Index range	Operational interpretation
Excellent	$Q \geq 85$	Both parameters within window
Good	$70 \leq Q < 85$	Minor excursion, no action
Moderate	$55 \leq Q < 70$	Sustained excursion, monitor
Poor	$35 \leq Q < 55$	Significant stress, intervene
Critical	$Q < 35$	Severe stress, immediate action

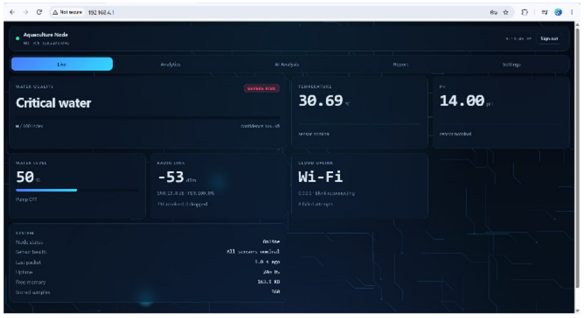


Figure 3: Live tab of the locally hosted dashboard, served from the receiving node at 192.168.4.1, showing the current water-quality grade and confidence, instantaneous temperature and pH, water level and pump state, LoRa link quality, cloud uplink status and node health counters.

6.2. Optional Blynk Cloud Uplink

When the receiving node is within range of an existing Wi-Fi network it can additionally operate in station mode and mirror a reduced set of telemetry channels to a Blynk cloud dashboard, allowing conditions to be inspected from outside the farm. Six virtual channels are published: temperature, pH, quality index, grade, water level and link quality. The uplink is strictly optional and additive; sensing, assessment, storage and visualisation all continue unchanged when it is unavailable, and the local dashboard remains the authoritative interface. Fig. 2 captures the uplink in a reconnecting state after eight failed attempts while the local dashboard continued to serve live data without interruption, which demonstrates the intended degradation behaviour. Fig. 3 shows the remote view when the uplink is available.

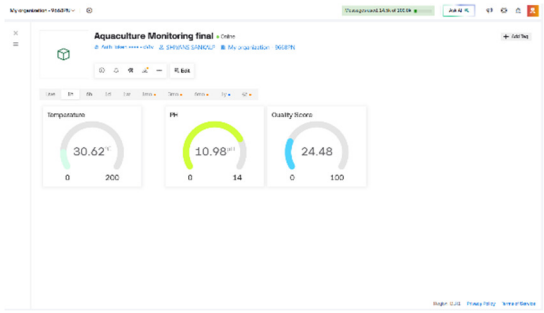


Figure 4: Optional Blynk cloud dashboard providing remote access to temperature, pH and quality score when an internet connection is available.

7. EXPERIMENTAL RESULTS

The prototype was operated continuously for 9 h 30 min at a sampling interval of 30 s. Over that period the receiver logged 17,043 packets with a delivery ratio of 99.99 %, a mean RSSI of −52 dBm and a mean SNR of 11.8 dB. The complete firmware, including sensing, inference, logging and the web server, occupied 68 kB of the 1408 kB flash region and left 161,000 bytes free of the 384,000-byte SRAM, which is ample headroom for the workload. These figures are collected in Table V.

Table V. SUMMARY OF MEASURED SYSTEM Performance

Metric	Value
Test duration	9 h 30 min
Sampling interval	30 s
Packets received	17,043
Packet delivery ratio	99.99 %
Mean RSSI	−52 dBm
Mean SNR	11.8 dB
Flash footprint	68 kB of 1408 kB
Free SRAM	161,000 B of 384,000 B

The dashboard delivered telemetry visualisation, statistics, trends, grading and report generation throughout, with no internet access at any point in the run. Fig. 4 shows the recent-samples panel, and Figs. 5 and 6 the temperature and pH history views with the summary statistics computed on the node itself.

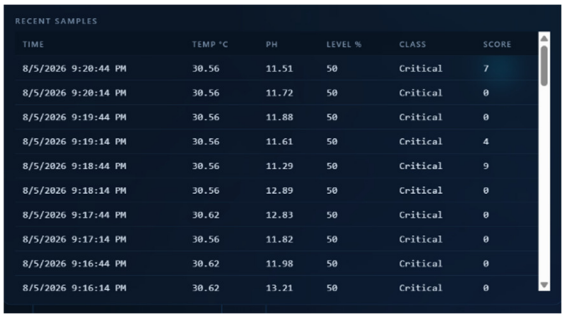


Figure 5: Recent samples panel of the dashboard, showing timestamped temperature, pH, water level, assigned grade and quality score recorded at 30 s sampling intervals.

Figure 6: Temperature history over a 360-sample observation window, with mean, minimum, maximum and hourly rate-of-change statistics computed on the node (mean 32.73 °C, range 29.62–33.69 °C, trend +0.90 °C/h).

Figure 7: pH history over the same 360-sample window (mean 9.60, range 7.48–14.00, trend −1.673 /h). The step near the end of the window corresponds to the electrode being lifted from the sample during handling, and illustrates the range-plausibility flagging described in Section III.

The one clear shortcoming observed was the absence of auto-calibration for the pH electrode. Readings drifted enough over the run that calibration is better performed at the bench before deployment than at the pond after installation.

8. DISCUSSION AND LIMITATIONS

Four limitations bound the present work. The system reasons over only two water-quality parameters, so anomalies in variables such as dissolved oxygen cannot be detected, even though dissolved oxygen is frequently the proximate cause of stock loss. The grading logic is heuristic, derived from published tolerance windows rather

than validated against a field-collected dataset, so the coefficients in Table III should be regarded as a starting point for calibration on a particular species and pond. The LoRa link carries no message authentication, which is acceptable for a single-farm point-to-point deployment but would not be so for shared infrastructure. Finally, the node depends on an external time source at start-up to timestamp its records.

Each of these is addressable within the existing architecture. Additional probes extend the perception layer without altering the transport or presentation layers; the assessment coefficients are already stored in non-volatile memory and updatable from the dashboard; and a per-record message authentication code would fit within the current 12-byte payload budget at modest airtime cost.

9. CONCLUSION

This paper has presented a LoRa-based, edge-first water-quality monitoring system for aquaculture that acquires, evaluates and visualises pond conditions on the device itself, with no dependence on cloud services or an internet connection. Testing demonstrated a packet delivery ratio of 99.99 % across 17,043 records over 9 h 30 min, with the complete firmware occupying 68 kB of flash. The results show that real-time, long-range, fully offline water-quality monitoring and assessment can be achieved with simple and readily available hardware, and that the interpretation step usually delegated to a remote server can be performed at the pond without loss of transparency.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype.

REFERENCES

[1] Food and Agriculture Organization of the United Nations, The State of World Fisheries and Aquaculture 2024: Blue Transformation in Action. Rome: FAO, 2024.

[3] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," IEEE Commun. Mag., vol. 55, no. 9, pp. 34–40, Sep. 2017.

[4] J. C. Liando, A. Gamage, A. W. Tengourtius, and M. Li, "Known and unknown facts of LoRa: Experiences from a large-scale measurement study," ACM Trans. Sensor Netw., vol. 15, no. 2, pp. 1–35, Feb. 2019.

[5] T. H. M. Nguyen, T. V. Nguyen, and L. B. Nguyen, "LoRa-based IoT sensor network for real-time monitoring of shrimp pond water quality in the Mekong Delta," in Proc. Int. Conf. Advanced Computing and Applications (ACOMP), Ho Chi Minh City, Vietnam, 2016, pp. 99–106.

[6] R. K. Kodali and B. S. Mahesh, "A low cost implementation of MQTT using LoRa for smart aquaculture," in Proc. 2nd IEEE Int. Conf. Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2017, pp. 1708–1712.

[7] G. Callebaut, G. Ottoy, and L. Van der Perre, "Characterization of LoRa point-to-point path-loss: Measurement campaigns and modeling considering censored data," IEEE Internet Things J., vol. 8, no. 3, pp. 1910–1918, Feb. 2021.

[8] R. K. Horton, "An index number system for rating water quality," J. Water Pollut. Control Fed., vol. 37, no. 3, pp. 300–306, 1965.

[9] C. E. Boyd, Water Quality: An Introduction, 3rd ed. Cham, Switzerland: Springer, 2020.

[10] Semtech Corporation, SX1276/77/78/79 Low Power Long Range Transceiver Datasheet, Rev. 7, Camarillo, CA, 2020.

[11] Maxim Integrated, DS18B20 Programmable Resolution 1-Wire Digital Thermometer, Rev. 6, San Jose, CA, 2019.

[12] Espressif Systems, ESP32-WROOM-32 Datasheet, Version 3.4, Shanghai, China, 2023.

[13] LoRa Alliance, LoRaWAN 1.1 Specification, LoRa Alliance Technical Committee, San Ramon, CA, 2017.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 125-129 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

DESIGN OF A BLUETOOTH-CONTROLLED ESP32-BASED SMART RC CAR WITH ULTRASONIC OBSTACLE DETECTION

[1]Subhendu Hait, [2]Subhadip Maity, [3]Tushar Kanti Shit, [4]Asim Kuilya

[1,2]UG student, Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[3]School of Electronics Engineering, KIIT Deemed to be University, Bhubaneswar

[4]Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

This paper presents the design and development of an RC car that can be operated wirelessly using a remote controller. The system consists of a chassis, DC motors, motor driver, battery, microcontroller, and a wireless communication module. The controller transmits commands to the receiver, which controls the movement of the car in different directions, including forward, reverse, left, and right. The main objective of this project is to demonstrate the integration of electronics, embedded systems, and wireless communication in a practical application. The RC car offers reliable performance, ease of operation, and low power consumption, making it suitable for educational, research, and hobby purposes. This project also provides a foundation for future enhancements such as obstacle avoidance, autonomous navigation, GPS tracking, and IoT-based remote monitoring.

KEYWORDS: RC Car, Remote Control, Obstacle Avoidance, Autonomous Navigation, IoT, GPS Tracking.

1. INTRODUCTION

Nowadays the radio controlled cars have developed to a high level and are used in many cases. Depending on their use, they vary in properties, design and special features. There are two major types of radio controlled cars, one that uses as energy source a conventional battery and the other type of radio controlled car utilizes a small internal combustion engine as an energy converter, which uses fossil fuel as energy source. Depending on its use a radio controlled car can be: (i) an army robot to detonate a buried bomb, (ii) an android robot to do a human job, (iii) a mobile robot to move around the environment like an automatic guided vehicle (AGV), (iv) a children's toy car or an RC car that can be operated with a radio wave, and (v) a radio controlled car with the purpose to detect fault on equipment with restricted access. These are some examples where a radio controlled car can be used. Radio control (regularly shortened to R/C or basically RC) is the utilization of radio signals to remotely control a gadget. Mechanical, military, and logical research associations make utilization of radio-controlled vehicles also. This research work incorporates a robotic car consisting of an RF receiver, decoder, motor driver and DC motor [1], [2].

- ESP32 → main controller
- Bluetooth → wireless control
- Smart RC car → manual + autonomous operation
- Ultrasonic → obstacle detection
- Obstacle avoidance → automatic decision making

2. METHODS AND MATERIALS

2.1 Hardware Specification

Wheel: Wheels are of many types:

1. Standard / Fixed wheel
2. Orientable wheel
3. Ball wheel
4. Off-centred wheel

*Corresponding Address: subhadipmaity862@gmail.com DOI 10.5281/zenodo.22038832

5. Omni wheel
6. Mecanum wheel

Here, we use a fixed wheel that can rotate forward or reverse. The center of the wheel is fixed to the robot chassis. The angle between the robot chassis and wheel plane is constant.

Geared DC Motor: A geared DC motor has a gear assembly attached to the motor. The speed of the motor is counted in terms of rotations of the shaft per minute and is termed as RPM. The gear assembly helps in increasing the torque and reducing the speed.

Servo Motor: A servo motor is a rotary or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration in a mechanical system.

Ultrasonic Sensor: An ultrasonic sensor is a device that uses ultrasonic sound waves to measure distance or detect objects. It consists of a transmitter that emits high-frequency sound pulses and a receiver that detects the reflected waves. By calculating the time taken for the sound to bounce back, the sensor determines the distance to the object [3].

Microcontroller:

- A microcontroller is a small integrated circuit used to control electronic devices and embedded systems [2], [4].
- It contains a processor (CPU), memory (RAM & ROM), and input/output ports on a single chip [2].
- It reads input from sensors or buttons, processes the data, and controls outputs like motors, LEDs, or displays [2].
- Microcontrollers are widely used in robots, home appliances, cars, and IoT devices [2].

ESP32: The ESP32 is a powerful and cost-effective microcontroller designed by Espressif Systems. It is a low-power System on Chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth [3].

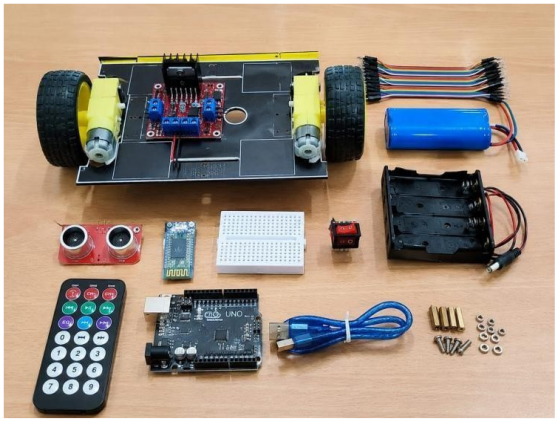


Figure 1: Components

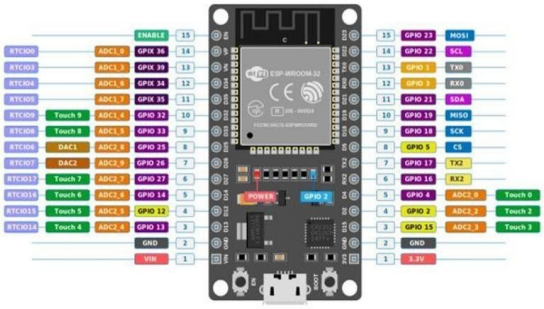


Figure 2: ESP32 pin configuration

2.2 Software

Arduino IDE: The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards, ESP boards etc..

Pulse Width Modulation (PWM): Pulse width modulation or PWM is a commonly used control technique that generates analog signals from digital devices such as microcontrollers. The signal thus produced will have a train of pulses, and these pulses will be in the form of square waves [5].

3. WORKING PRINCIPLE

The proposed RC car is built using an ESP32 microcontroller, an L298N motor driver, DC motors, an SG90/MG996R servo motor, and a rechargeable battery. The ESP32 serves as the main controller and communicates wirelessly with a smartphone or remote controller through Bluetooth or Wi-Fi. When the user sends a command such as Forward, Backward, Left, Right, or Stop, the ESP32 receives the command and processes it. For forward and backward movement, the ESP32 generates digital control signals that are sent to the L298N motor driver. The motor driver amplifies these signals and supplies the required current to the DC motors,

causing the wheels to rotate in the desired direction.

For steering, the ESP32 generates a Pulse Width Modulation (PWM) signal to control the SG90 or MG996R servo motor. The servo motor rotates to a specific angle, turning the front wheels left or right according to the received command. This provides accurate and smooth steering of the vehicle. A rechargeable battery powers the ESP32, L298N motor driver, servo motor, and DC motors. The coordinated operation of these components enables reliable wireless control, efficient movement, and precise steering [8].

This system provides real-time wireless control, low power consumption, and stable performance, making it suitable for educational, robotics, and research applications.

4. CIRCUIT DIAGRAM

The circuit diagram showing the interconnections among the ESP32, L298N motor driver, geared DC motors, servo motor, and ultrasonic sensor is illustrated in Fig. 3.

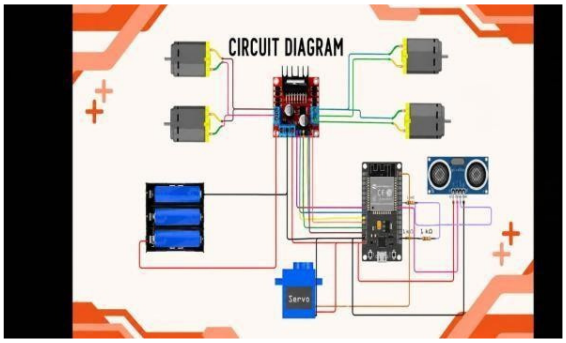


Figure 3: Circuit diagram

5. DISCUSSION

Before presenting the experimental results, the connections between the Arduino IDE and ESP32, computer and ESP32 through Bluetooth, and the controller and motor are tested.

5.1 Arduino–ESP32 Connection

The Arduino and ESP32 connection was first tested using the Arduino IDE. The ESP32 was connected to the computer using a USB cable, and the required Arduino C++ program was uploaded to the ESP32. After uploading the program, the Serial Monitor was opened to verify the operation of the ESP32. Test messages were displayed on the Serial Monitor to confirm that the program was running correctly [2].

During the initial testing, an error occurred and no proper response was observed. The problem was solved by checking the USB connection, power supply, program settings, and communication pins. After correcting the connections and uploading the program again, the ESP32 operated successfully.

5.2 Computer–ESP32 Bluetooth Connection

After successfully testing the ESP32, the next step was to establish a Bluetooth connection between the computer and ESP32. The ESP32 Bluetooth function was programmed using Arduino C++. The computer was paired with the ESP32 through Bluetooth. After successful pairing, the computer could send control commands wirelessly to the ESP32. The received Bluetooth commands were processed by the C++ program running on the ESP32.

Different commands were used to control the movement of the RC car, such as forward, reverse, left, right, and stop. The Bluetooth communication was tested at different distances to determine the effective operating range. The connection was considered successful when the ESP32 received the commands correctly and the car responded according to the transmitted command.

5.3 Controller and Motor Connection

After establishing the Bluetooth connection between the computer and ESP32, the controller and motor connections were tested. The Arduino C++ program was used to process the Bluetooth commands received by the ESP32. Four basic movement directions were tested: forward, reverse, left, and righ. A stop command was also used to stop the motors.

When the forward command was received, both DC motors rotated in the forward direction. When the reverse command was received, both motors rotated in the opposite direction. For turning right, the motors were controlled to produce a right turn, while the left command produced a left turn.

During the initial testing, some commands were not received correctly and the motors did not always respond as expected. The problem was solved by checking the Bluetooth communication, motor-driver connections, power supply, and Arduino C++ program. The final test demonstrated that the ESP32 successfully

received Bluetooth commands from the computer and controlled the DC motors through the driver.

5.4 Range-Testing Results

Range testing was conducted in two environments: with obstacles and without obstacles. The tests were performed indoors. The obstacle environment contained walls, doors, furniture, and other objects, while the unobstructed environment provided a more open path.

The computer was connected to the ESP32 through Bluetooth, and commands were sent to control the RC car. The distance between the computer and the RC car was gradually increased to determine the effective Bluetooth communication range.

From Table 1, it can be observed that the Bluetooth communication performance depends on the distance and surrounding obstacles. At shorter distances, the ESP32 provides reliable Bluetooth communication and the RC car responds properly to the commands. At approximately 10 meters, the system operates effectively without obstacles, while obstacles reduce the signal quality. At approximately 15–20 meters, the Bluetooth signal becomes weaker and the car may respond intermittently. At 25 meters, the Bluetooth signal was not detected during the test. Therefore, the practical operating range depends on factors such as distance, obstacles, interference, and the surrounding environment [3].

Table 1. Bluetooth Range-Testing Results

Distance	**With Obstacles**	**Without Obstacles**	**Comments**
1 meter	Excellent	Excellent	Bluetooth signal is strong.
5 meters	Excellent	Excellent	Bluetooth signal is strong.
10 meters	Good	Excellent	Signal strength decreases with obstacles.
15 meters	Poor (car still moves)	Good	Signal becomes weaker with obstacles.
20 meters	Poor (car still moves)	Poor (car still moves)	Bluetooth signal is weak and intermittent.
25 meters	Negative	Negative	Bluetooth signal is not detected.

6. RESULT

These authors control the motion of the car through the radio frequency application [7]. The movement of the car using the transmitter is shown in Fig. 4.

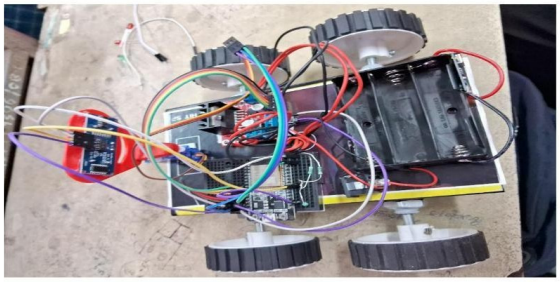


Figure 4: RC car

Overall, the experiment demonstrates that ESP32 Bluetooth communication combined with an Arduino C++ control program can provide reliable wireless control of the RC car within the tested range.

7. CONCLUSION

In this paper, it has been shown that it is possible to perform communication between the transmitter and receiver using radio frequency. The car can be moved forward and backward and can be turned left and right. The circuit can be modified and transformed into an amphibious mechanism which can run both on land and water efficiently. This technique may also be implemented for other purposes. The feasibility of the proposed technique has been shown.

REFERENCES

[1] K. V. Alexandra, "Prototype model tests of a high speed wheeled amphibious vehicles," in *Proceedings of the Eighth Australasian Fluid Mechanics Conference*, University of Newcastle, N.S.W., Australia, 1983.

[2] J.-D. Warren, J. Adams, and H. Molle, *Arduino Robotics*, 1st ed. Berkeley, CA, USA: Apress, 2011, 628 pp., ISBN 978-1-4302-3183-7.

[3] T. Bräunl, *Embedded Robotics: From Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino*, 4th ed. Singapore: Springer, 2022, 519 pp., ISBN 978-981-16-0803-2.

[4] R. Kamal, *Embedded Systems: Architecture, Programming & Design*. New Delhi, India: McGraw-Hill Education, 2014.

[5] R. Grimmett, *Arduino Robotic Projects: Build Awesome and Complex Robots with the Power of Arduino*. Birmingham, U.K.: Packt Publishing, 2014, ISBN 978-1-78398-982-9.

[6] L. Vachhani, P. Vyas, and A. G. K. Arunkumar, *Embedded Control for Mobile Robotic Applications*. Hoboken, NJ, USA: Wiley, 2022, ISBN 978-1-119-81238-8.

[7] K. Dubey and N. K. Khemka, "Line follower robot using radio frequency (RF) technology," *IOSR Journal of Mechanical and Civil Engineering*, vol. 3, pp. 1–5, 2013.

[8] M. A. Othman, M. M. Ismail, H. A. Sulaiman, M. H. Misran, and Y. Z. A. Mohd, "Wireless controlled omni directional monitoring robot with video support," *International Journal of Engineering Research and Applications*, vol. 2, no. 4, pp. 2228–2232, 2012.

[9] A. Canamar and L. Smrcek, "2050 visionary concept: Advance amphibious preliminary design," *International Journal of Engineering Research and Development*, vol. 3, no. 6, pp. 1–12, 2012.

[10] S. Banerji, "Design and implementation of an unmanned vehicle using a GSM network with microcontrollers," *International Journal of Science, Engineering & Technology Research*, vol. 2, no. 2, 2013.

[11] W. M. Khaing and K. Thiha, "Design and implementation of remote operated spy robot control system," *International Journal of Science, Engineering and Technology Research*, vol. 3, no. 7, 2014.

[12] T. Bräunl, *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*, 3rd ed. Berlin/Heidelberg, Germany: Springer, 2008.

[13] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd ed. Cambridge, MA, USA: MIT Press, 2011, 472 pp., ISBN 978-0-262-01535-6.

[14] R. Kamal, *Microcontrollers: Architecture, Programming, Interfacing and System Design*, 2nd ed. Pearson Education India.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 130-136 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Edge AI-Powered Predictive Water-Quality Monitoring and Early Warning for Smart Aquaculture

[1]Rishu Raj, [2]Pryobroto Sarkar, [3]Ritwik Kapat, [4]Riya Panda, [5]Debjyoti Adhikari, [6]Ritwik Haldar

[1,2,3,4]UG Students, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

[6]Assistant Professor, Dept. of Information Technology, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

[6]Associate Professor, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

ABSTRACT

Water quality in aquaculture ponds varies with environmental conditions, biological activity, and daily cycles, so periodic manual measurement is inherently reactive and detects unfavourable conditions only after they have developed. This work presents an Edge Artificial Intelligence (Edge AI) predictive monitoring system for smart fish ponds that forecasts water temperature and pH and provides early risk indication without cloud-based inference. A real-world 24-hour pond dataset is processed by converting timestamps into a decimal hour-of-day feature. Decision Tree Regression and a Long Short-Term Memory (LSTM) network are trained and compared: the Decision Tree uses hour-of-day as its predictor, while the LSTM uses normalised temperature sequences as a temporal benchmark. For temperature, both models reproduce the diurnal pattern with almost identical accuracy, the Decision Tree achieving a coefficient of determination of 0.9989 and the LSTM 0.9987, with the LSTM marginally lower in absolute error. The Decision Tree is therefore selected not on accuracy but on deployability, since it trains in under a second, occupies a few tens of kilobytes, and executes on a microcontroller, whereas the LSTM could not be deployed on the target device. The selected models are converted to C++ using micromlgen and deployed on an ESP32, which synchronises time over NTP, performs local inference in under one millisecond, and displays four-hour forecasts on a 16 × 2 I2C LCD. A threshold-based mechanism issues early warnings when predicted values fall outside predefined safe ranges. The prototype demonstrates a compact TinyML approach to predictive aquaculture monitoring on inexpensive embedded hardware.

KEYWORDS: Edge AI, Smart Aquaculture, Decision Tree Regression, LSTM, TinyML, Water Quality Monitoring.

2. INTRODUCTION

Aquaculture plays an important role in modern food production, and maintaining suitable water-quality conditions is essential for healthy fish growth and pond productivity. Temperature and pH are two of the most informative indicators of pond condition, and both vary with environmental conditions, biological activity, feeding, rainfall, and the daily solar cycle. Conventional monitoring generally relies on periodic manual measurement, which can miss rapid changes and provides information only after an undesirable condition has already developed. A monitoring scheme that anticipates the near-term evolution of these parameters would allow corrective action to be taken before fish are placed under stress.

This work proposes a low-cost Edge AI predictive monitoring system for smart fish ponds based on the ESP32 platform. The system analyses real-world pond temperature and pH data and uses machine-learning models to forecast future water-quality conditions. Decision Tree Regression is developed for temperature and pH prediction, while an LSTM network is implemented as a temporal benchmark. The selected Decision Tree models are converted into embedded C++ using micromlgen and deployed directly on the ESP32, enabling local inference without continuous cloud-based machine-learning processing.

*Corresponding Address: sarkar.pryobroto28@gmail.com DOI 10.5281/zenodo.22038876

The system generates four-hour forecasts and displays the predicted values on a 16 × 2 I2C LCD. A threshold-based early-warning mechanism identifies predicted temperature and pH values that fall outside predefined safe ranges, so that potential risks are indicated before they occur. It should be noted that the available dataset contains no labelled anomalous events; the implementation is therefore correctly described as a predictive early-warning system rather than a labelled anomaly detector. The primary objective of this work is to demonstrate a compact and practical TinyML approach to predictive aquaculture monitoring using inexpensive embedded hardware.

The remainder of this paper is organised as follows. Section II reviews related work in smart aquaculture, water-quality prediction, and TinyML. Section III describes the proposed system architecture, the dataset, and the machine-learning models. Section IV presents the experimental evaluation and the model comparison. Section V concludes the paper and sets out its limitations and future scope.

2. LITERATURE REVIEW

Boyd and Tucker [1] established the importance of maintaining suitable water-quality conditions in aquaculture ponds and documented the physical and biological processes that drive the diurnal variation of temperature, pH, and dissolved oxygen. Gubbi et al. [2] set out the architectural elements of the Internet of Things that underpin distributed environmental sensing, and Olanubi et al. [3] applied these principles directly to aquaculture through an IoT-based intelligent water-quality management system. Geetha and Gouthami [4] demonstrated an IoT-enabled real-time water-quality monitoring system, confirming the practicality of continuous measurement in place of periodic sampling.

A second group of studies applies machine learning to the prediction of water-quality parameters. Zhang et al. [5] combined kernel principal component analysis with a recurrent network to predict the trend of dissolved oxygen, and Zambrano et al. [6] showed that classical machine-learning models can predict manually measured water-quality parameters in fish farming with useful accuracy. Li et al. [7] applied a gated recurrent unit network to dissolved-oxygen prediction in a fishery pond and argued explicitly for the lower parameter count and training cost of the gated recurrent unit relative to the LSTM, while Chen et al. [8] combined an LSTM with an attention mechanism for river water-quality prediction. These works establish that the temporal structure of pond data is learnable, but they assume computation on a server or a personal computer.

A third group addresses the deployment of such models on constrained hardware. Banbury et al. [9] characterised the benchmarking and resource challenges of TinyML on microcontroller-class devices. Kotsiantis [10] reviewed the Decision Tree as an efficient and interpretable learning method whose inference reduces to a sequence of comparisons, which makes it well suited to embedded execution, while Hochreiter and Schmidhuber [11] introduced the LSTM architecture that has since become the standard baseline for temporal prediction. Salerno [12] provides micromlgen, which converts trained scikit-learn estimators into self-contained C++ classes, and Espressif Systems [13] documents the ESP32 as a low-cost platform combining a dual-core processor with integrated Wi-Fi.

Sowmiya et al. [14] brought these strands together in a TinyML-based reconfigurable IoT platform for brackish-water aquaculture monitoring. Most existing systems, however, concentrate either on measurement and transmission or on prediction performed off the device, and comparatively few compare a deployable model against a recurrent benchmark on the same data before moving inference onto the microcontroller itself. The proposed system addresses this gap by training a Decision Tree and an LSTM benchmark on the same pond record, quantifying the difference between an hour-of-day regressor and a one-step-ahead recurrent predictor, and deploying the selected model directly on an ESP32 to generate four-hour temperature and pH forecasts with local early-warning capability.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed system comprises two connected stages. The first is an offline machine-learning pipeline executed on a personal computer: it loads the pond dataset, converts timestamps to decimal hours, performs exploratory analysis, trains the Decision Tree and LSTM models, evaluates their

performance, and exports the selected Decision Tree models. The second is the edge-deployment stage, in which the generated C++ model header is compiled into the ESP32 firmware. The device obtains the current time through the Network Time Protocol (NTP), predicts the next four hourly values, and displays them on the LCD. The complete workflow is shown in Fig. 1.

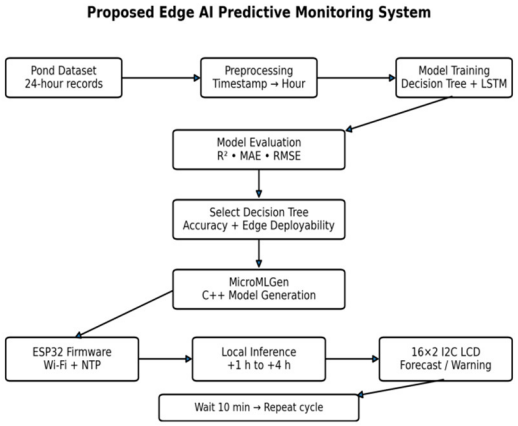


Figure 1: End-to-end Edge AI workflow, from dataset preparation and model training to continuous four-hour forecasting and warning on the ESP32.

3.2. Dataset and Feature Engineering

The dataset was collected from an active fish pond over a continuous 24-hour period and contains timestamp, temperature (°C), and pH fields. Each timestamp is converted into a decimal hour feature computed from its hour, minute, and second components, so that the time of day is represented as a single continuous variable. Hour-of-day is used as the principal predictor for the Decision Tree models, on the basis that both parameters are driven primarily by the solar cycle.

3.3. Decision Tree Regression

Two Decision Tree Regression models are trained separately, one for temperature and one for pH. The scikit-learn Decision Tree Regressor is configured with max_depth=10 and random_state=42, and the dataset is divided into 80% training and 20% testing partitions. The model recursively partitions the hour feature and returns the learned target value at the corresponding leaf, so that inference at run time reduces to a short sequence of numerical comparisons.

3.4. LSTM Benchmark

The LSTM model uses a sliding window of 20 normalised temperature values to predict the next value. Its architecture, summarised in Table 1, contains an LSTM layer with 64 units, a second LSTM layer with 32 units, and a single-unit linear dense output layer. The model is trained with the Adam optimiser and mean squared error loss for 20 epochs, with a batch size of 64 and a 20% validation split [15]. It should be noted that the LSTM performs one-step-ahead prediction from a window of past temperature values, whereas the Decision Tree maps hour-of-day directly to the target; the two are therefore compared as alternative modelling strategies on the same record rather than on an identical prediction task.

Table 1: LSTM benchmark model architecture.

Layer	Type	Units	Output shape
1	LSTM	64	(batch, 20, 64)
2	LSTM	32	(batch, 32)
3	Dense	1	(batch, 1)

3.5. Model Evaluation Metrics

The models are evaluated using the coefficient of determination (R^2), the mean absolute error (MAE), and the root mean squared error (RMSE). R^2 measures the proportion of variance explained, MAE reports the mean magnitude of the prediction error in the units of the measured parameter, and RMSE weights larger errors more heavily and is therefore the more sensitive indicator of occasional large deviations.

3.6. Edge Deployment

Micromlgen converts the trained scikit-learn Decision Tree estimators into self-contained C++ classes exposing a predict() method [12]. The temperature and pH models are stored in dt_models.h as TempPredictor and pHPredictor. The ESP32 firmware is written in Arduino C++ and combines Wi-Fi connectivity, NTP time synchronisation, the embedded prediction classes, and the LiquidCrystal_I2C library. The LCD is addressed over I2C at address 0x27, with GPIO 21 used for SDA and GPIO 22 for SCL, as illustrated by the reference wiring in Fig. 2. In each cycle the firmware calculates the next four target hours, performs local inference, and presents the forecast in 12-hour AM/PM format. Each forecast frame is displayed for three seconds, followed by a ten-minute pause before the next cycle begins.

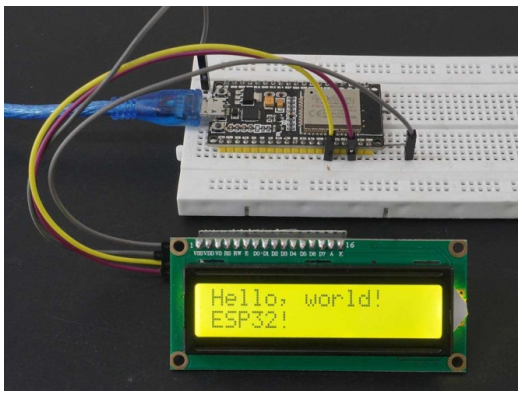


Figure 2: Reference wiring of the ESP32 interfaced with a 16 × 2 I2C LCD (pin assignment after [13]).

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

The machine-learning experiments were performed on a Windows 11 computer with an Intel Core i5 processor and 8 GB of RAM, using Python 3.10 with Pandas, NumPy, scikit-learn, TensorFlow, Matplotlib, Seaborn, and micromlgen [15], [16]. The ESP32 firmware was developed and flashed using Arduino IDE 2.x.

4.2. Exploratory Analysis

The recorded data show the diurnal behaviour expected of a shallow pond. Temperature rises from approximately 24.5 °C in the early morning to slightly above 31 °C in the afternoon, and pH varies across a narrower band of approximately 7.05 to 7.80, rising during the period of peak photosynthetic activity and falling thereafter.

The correlation structure of the dataset is shown in Fig. 3. Temperature and pH are very strongly correlated with one another (r = 0.97), which is consistent with both being driven by the same photosynthetic and thermal cycle. The linear correlation between hour-of-day and each parameter is, by contrast, weak (r = 0.25 for temperature and r = 0.39 for pH). This apparent inconsistency is a property of the Pearson coefficient rather than of the data: the diurnal profile rises and then falls within the observation window, so the relationship between hour and parameter is strong but non-monotonic, and a linear measure of association necessarily understates it. This observation is important to the design of the system, because it rules out simple linear regression on hour-of-day and motivates the use of a model capable of representing non-monotonic structure, such as a regression tree.

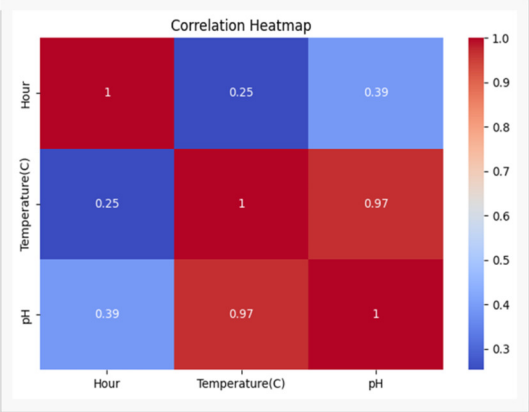


Figure 3: Pearson correlation heatmap for hour-of-day, temperature, and pH.

The frequency distributions of the two parameters are shown in Fig. 4. Neither is normally distributed. The temperature distribution spans approximately 24.5 °C to 31.5 °C and is bimodal with a pronounced mode near 31 °C, reflecting the extended period the pond spends near its afternoon maximum. The pH distribution spans approximately 7.05 to 7.80 and is left-skewed, with the greatest density towards the upper end of the range and a longer tail towards lower values. No extreme outliers are present in either parameter, indicating that the recorded data are free of gross sensor faults over the observation period.

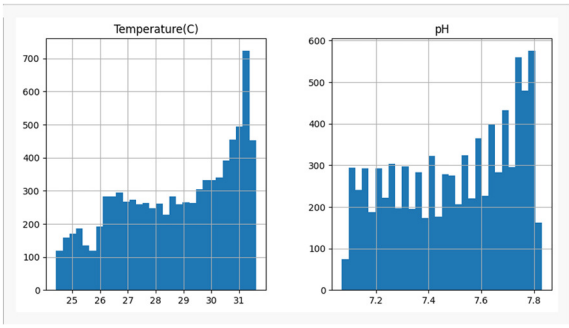


Figure 4: Frequency distributions of temperature and pH over the 24-hour observation period.

4.3. Prediction Performance

Table 2 reports the performance of the trained models on the held-out test partition. Both models explain almost all of the variance in the measured temperature, and the difference in their absolute errors is small in relation to the range over which the parameter varies during the observation window. The Decision Tree pH model achieves an R^2 of 0.9850 with an MAE of 0.020 and an RMSE of 0.030 pH units, slightly below its

temperature counterpart, which is consistent with the narrower dynamic range of the pH signal.

Table 2. Model performance on the held-out test partition. Temperature errors are expressed in °C and pH errors in pH units.

Model / target	R^2	MAE	RMSE
Decision Tree – Temperature	0.9989	0.057	0.069
LSTM – Temperature	0.9987	0.053	0.063
Decision Tree – pH	0.9850	0.020	0.030

4.4. Model Comparison

Fig. 5 compares the coefficient of determination achieved by the two models for temperature prediction. The scores are effectively indistinguishable, at 0.9989 for the Decision Tree and 0.9987 for the LSTM, indicating that both architectures capture the diurnal structure of the data equally well. The result is unsurprising: the underlying signal is a smooth, strongly periodic function of time of day, and both a sufficiently deep regression tree and a recurrent network are able to represent it closely.

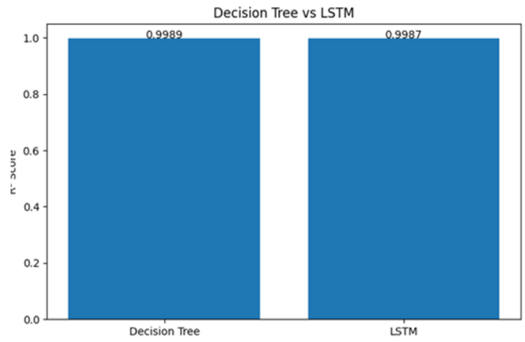


Figure 5: Coefficient of determination for the Decision Tree and the LSTM benchmark.

Fig. 6 compares the two models on absolute error. Here the LSTM is marginally the more accurate of the two, with an MAE of 0.053 against 0.057 and an RMSE of 0.063 against 0.069 for the Decision Tree. The difference is 7% in MAE and 9% in RMSE, corresponding to between four and six thousandths of a degree Celsius, and is therefore not practically significant for pond management.

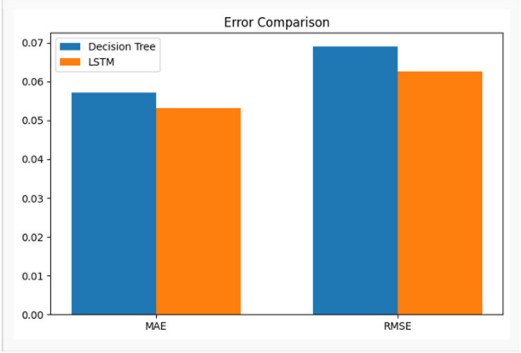


Figure 6: Mean absolute error and root mean square error for the Decision Tree and the LSTM benchmark.

The selection of the Decision Tree for deployment therefore does not rest on predictive accuracy, and Table 3 sets out the basis on which it was made. At practically equivalent accuracy, the Decision Tree trains in under a second rather than several minutes, compiles to a model of a few tens of kilobytes, and executes on the ESP32 in under one millisecond per prediction. The LSTM was not deployed on the target device: it requires a floating-point tensor runtime and a persistent 20-sample input window, and micromlgen provides no conversion path for recurrent models, so no equivalent single-header C++ export was available. The appropriate conclusion is that for a signal of this regularity the additional representational capacity of the recurrent model buys no useful accuracy, and the simpler model should be preferred because it is the one that can actually run at the edge.

Table 3: Comparison of the Decision Tree and the LSTM benchmark.

Metric	Decision Tree	LSTM
R^2 (temperature)	0.9989	0.9987
MAE	0.057	0.053
RMSE	0.069	0.063
Training time	< 1 s	Minutes
Model size	≈ 10–50 KB	Not measured
ESP32 inference	< 1 ms	Not deployed
Edge deployment	Full TinyML	Not feasible

4.5. Real-Time ESP32 Evaluation

During testing, the ESP32 connected to the configured Wi-Fi network, displayed a connection message, synchronised its clock over NTP, and reported that the Decision Tree models were ready. At a test time of 19:45 Indian Standard Time (IST), the device generated forecasts for the following one, two, three, and four hours. The predicted values followed the

expected evening decline in both temperature and pH, confirming correct operation of the embedded inference loop and of the time-handling logic that maps the current NTP time onto the hour feature expected by the models.

4.6. Predictive Risk Indication

Safe operating ranges of 26–32 °C for temperature and 6.5–8.5 for pH are defined for the cultivated species. A predicted value outside the safe range raises an advisory warning on the LCD, and a value beyond the critical boundaries listed in Table 4 raises a critical warning. Sustained temperatures below 24 °C or above 34 °C are associated with thermal stress and reduced dissolved-oxygen solubility, while pH values below 6.0 or above 9.0 are associated with osmoregulatory failure. The observed minimum of approximately 24.5 °C lies below the 26 °C safe lower bound but above the 24 °C critical bound, so early-morning forecasts on this record raise an advisory rather than a critical warning. Because the mechanism acts on predicted rather than measured values, a warning is raised in advance of the condition arising, which is the principal operational advantage of the predictive approach over threshold checking applied to live readings.

Table 4. Safe ranges and critical boundaries used for predictive risk indication.

Parameter	Safe range	Critical below	Critical above
Temperature	26–32 °C	< 24 °C	> 34 °C
pH	6.5–8.5	< 6.0	> 9.0

5. CONCLUSION

This paper presented the design and evaluation of a low-cost Edge AI predictive monitoring system for smart aquaculture. Decision Tree Regression and an LSTM benchmark were trained on a 24-hour pond dataset and compared on the same pond record, though on different input representations, and the selected Decision Tree models were deployed on an ESP32 using micromlgen to produce four-hour temperature and pH forecasts with threshold-based risk indication. The two models proved almost identical in accuracy, with the LSTM marginally lower in absolute error, so the Decision Tree was selected on the grounds of embedded deployability rather than predictive performance. The result is consistent with a broader design expectation for TinyML applications, which further work will need to confirm: where the target signal is strongly periodic and low-dimensional, a compact interpretable model may deliver accuracy comparable to that of a recurrent network at a small fraction of its computational cost.

Two limitations should be noted. First, the models were trained on a single 24-hour record, and the test partition was drawn at random from within that record; the reported accuracy therefore reflects interpolation within one observed diurnal cycle rather than generalisation to unseen days, and a multi-day evaluation with a chronological train–test split is required before the forecasting accuracy can be regarded as established. Second, the dataset contains no labelled anomalous events, so the early-warning mechanism is validated only against its defined thresholds and not against recorded incidents. Future work will address both points through extended multi-day data collection, direct sensor integration in place of the offline dataset, the inclusion of dissolved oxygen and turbidity as additional parameters, and adaptive on-device retraining as further data accumulate.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Department of Electronics and Communication Engineering, Haldia Institute of Technology, for the laboratory facilities used in this work.

REFERENCES

[1] C. E. Boyd and C. S. Tucker, *Pond Aquaculture Water Quality Management*. Norwell, MA: Kluwer Academic Publishers, 1998. doi: 10.1007/978-1-4615-5407-3

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. doi: 10.1016/j.future.2013.01.010

[3] O. O. Olanubi, T. T. Akano, and O. S. Asaolu, "Design and development of an IoT-based intelligent water quality management system for aquaculture," *Journal of Electrical Systems and Information Technology*, vol. 11, art. no. 15, 2024. doi: 10.1186/s43067-024-00139-z

[4] S. Geetha and S. Gouthami, "Internet of things enabled real time water quality monitoring system," *Smart Water*, vol. 2, art. no. 1, 2017. doi: 10.1186/s40713-017-0005-y

[5] Y.-F. Zhang, P. Fitch, and P. J. Thorburn, "Predicting the trend of dissolved oxygen based on the kPCA-RNN model," *Water*, vol. 12, no. 2, art. no. 585, 2020. doi: 10.3390/w12020585

[6] A. F. Zambrano, L. F. Giraldo, J. Quimbayo, B. Medina, and E. Castillo, "Machine learning for manually-measured water quality prediction in fish farming," *PLOS ONE*, vol. 16, no. 8, art. no. e0256380, 2021. doi: 10.1371/journal.pone.0256380

[7] W. Li, H. Wu, N. Zhu, Y. Jiang, J. Tan, and Y. Guo, "Prediction of dissolved oxygen in a fishery pond based on gated recurrent unit (GRU)," *Information Processing in Agriculture*, vol. 8, no. 1, pp. 185–193, 2021. doi: 10.1016/j.inpa.2020.02.002

[8] H. Chen et al., "Water quality prediction based on LSTM and attention mechanism: A case study of the Burnett River, Australia," *Sustainability*, vol. 14, no. 20, art. no. 13231, 2022. doi: 10.3390/su142013231

[9] C. R. Banbury et al., "Benchmarking TinyML systems: Challenges and direction," arXiv:2003.04821, 2020. [Online]. Available: https://arxiv.org/abs/2003.04821

[10] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013. doi: 10.1007/s10462-011-9272-4

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735

[12] S. Salerno, *micromlgen: Generate C code for microcontrollers from Python's scikit-learn classifiers*, v1.1.28, 2021. [Online]. Available: https://github.com/eloquentarduino/micromlgen

[13] Espressif Systems, *ESP32 Technical Reference Manual*, v5.8, 2026. [Online]. Available: https://documentation.espressif.com/esp32_technical_reference_manual_en.pdf

[14] V. Sowmiya, G. R. Kanagachidambaresan, and M. Muralidhar, "Tiny ML-based reconfigurable IoT platform design for brackish water aquaculture monitoring," *Wireless Networks*, vol. 30, no. 9, pp. 7153–7165, 2024. doi: 10.1007/s11276-023-03537-9

[15] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications, 2017.

[16] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2019.

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 137-142 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Design and Implementation of an IoT-Based Smart Solar Photovoltaic Monitoring System with Cloud Analytics

[1]Sneha Burnwal, [2]Yash Shashwat, [3]Somnath Maity, [4]Santanu Maity, [5]Kisalaya Chakrabarti, [6]Kalyan Sundar Kola

[1,2,3]UG student, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

[4,5,6] Faculty Members, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal

ABSTRACT

The growing dependence on renewable energy has established solar photovoltaic (PV) generation as one of the most widely deployed sustainable energy sources. The output of a PV installation, however, varies continuously with solar irradiance, ambient temperature, dust accumulation, partial shading, and module ageing, and faults that develop between manual inspections often remain undetected for extended periods. This study presents a low-cost IoT-based smart solar monitoring system that measures the electrical and environmental parameters of a PV panel in real time. A 0–25 V resistive voltage sensor and an ACS712 Hall-effect current sensor are interfaced with an ESP32 microcontroller, which digitises the readings, applies calibration coefficients, and computes instantaneous power and cumulative energy on the device itself. The processed values are transmitted over Wi-Fi to a cloud platform, where live readings and time-series trends for voltage, current, power, and energy are presented on a dashboard accessible from any browser or mobile device. A local display provides on-site indication, while threshold-based logic flags open-circuit, short-circuit, and low-yield conditions. Field testing on a small PV panel confirmed continuous logging and remote visualisation of the generated data. The system provides an affordable and scalable solution for performance supervision and early fault detection in domestic, institutional, and agricultural PV installations.

KEYWORDS: IoT, Solar Energy,, Photovoltaic Monitoring, ACS712 Current Sensor, Cloud Dashboard.

1. INTRODUCTION

The rising demand for clean and renewable energy has made solar photovoltaic (PV) systems an important source of sustainable power for domestic, institutional, industrial, and agricultural applications. Unlike conventional generation, the output of a PV array is not constant: it varies continuously with solar irradiance, ambient and module temperature, humidity, dust deposition, partial shading, and the gradual ageing of the cells. Faults such as microcracks, string disconnection, connector corrosion, and converter failure further reduce the delivered power. In most small and medium installations these variations are never observed directly, because performance is assessed only from periodic energy-meter readings or occasional manual inspection. As a result, a degraded or partially faulty array may continue to operate at reduced output for weeks before the loss is noticed. What such installations require is continuous, parameter-level visibility rather than aggregate readings taken long after the event.

The Internet of Things (IoT) offers a practical route to this visibility, since low-cost sensors, an inexpensive microcontroller with integrated wireless connectivity, and a cloud platform together make it possible to acquire, transmit, store, and visualise electrical parameters at short intervals and at negligible recurring cost. This study presents the design and implementation of an IoT-based smart solar monitoring system built around the ESP32 microcontroller. The system measures panel voltage, current, temperature, and light intensity, computes instantaneous power and accumulated energy locally, and transmits the results over Wi-Fi to a cloud dashboard that presents live values and historical trends. The

*Corresponding Address: kalyankola12@gmail.com DOI 10.5281/zenodo.22038923

remainder of this paper reviews the related work and identifies the research gap, describes the proposed system architecture and hardware, sets out the methodology and implementation, and reports the experimental results before concluding with the limitations and future scope.

2. LITERATURE REVIEW

Swathi et al. [1] developed an IoT-based solar power monitoring system using an Arduino board. The system acquired panel parameters in real time and uploaded them to a monitoring platform, helping the user track generation and plan maintenance.

Sawashere et al. [2] proposed a solar panel monitoring and data-acquisition system for the real-time collection of solar power parameters. The recorded data were used to analyse panel performance and to improve overall energy management.

Nkinyam et al. [3] developed a low-cost IoT-based monitoring device for solar electric systems. The device enabled real-time observation of PV parameters and supported efficient management of the installation at a substantially lower cost than commercial monitoring hardware.

Trang et al. [4] designed a real-time system for monitoring and controlling solar power using IoT technology. In addition to remote observation, the system supported control actions, thereby improving the management of the solar energy resource.

Rao et al. [5] developed an IoT-based intelligent smart energy monitoring system for solar PV generation. The system tracked energy production continuously and applied the collected data to the analysis and improvement of plant performance.

Gopal et al. [6] proposed an IoT-based solar power monitoring system for the real-time observation of solar energy parameters. The work demonstrated that continuous parameter logging improves performance tracking and simplifies energy management.

Olayiwola et al. [7] presented a comprehensive analysis of the current–voltage (I–V) characteristic of photovoltaic modules, showing how irradiance and temperature reshape the characteristic curve and shift the maximum power point. Their analysis provides the theoretical basis for interpreting measured voltage and current data.

Domínguez-Bolaño et al. [8] presented an IoT system for a smart campus in which solar photovoltaic energy monitoring is one of several deployed use cases. The work addressed data collection, storage, visualisation, and efficient energy management at the scale of an institutional campus.

While the reviewed studies demonstrate progress in solar parameter acquisition, low-cost IoT monitoring hardware, and cloud-based visualisation, most address these functions individually. Some concentrate on data acquisition without cloud analytics, others provide a dashboard without on-device computation of derived quantities such as accumulated energy, and comparatively few combine electrical and environmental sensing with automatic fault indication in a single low-cost unit. Limited attention has also been given to performing power and energy computation at the edge so that the cloud receives engineering values rather than raw counts. The proposed system addresses this gap by integrating voltage, current, temperature, and irradiance sensing, on-device calibration and energy integration, cloud transmission with time-series visualisation, and threshold-based fault flagging within a single affordable platform.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed system is organised as a layered architecture in which sensing, edge processing, network transport, and cloud visualisation are separated into distinct functional stages. As shown in Fig. 1, the solar panel supplies the measured quantities to a voltage sensor, an ACS712 current sensor, a temperature sensor, and a light-dependent resistor. These devices form the sensing layer and convert the physical quantities into analog electrical signals. The ESP32 microcontroller constitutes the edge layer: it digitises the analog signals through its internal ADC, applies the stored calibration coefficients, computes the instantaneous power and the accumulated energy, and evaluates the fault-detection conditions. A 16 × 2 LCD displays the current readings locally so that the installation can be checked without network access. The Wi-Fi

radio integrated in the ESP32 forms the network layer and transmits the processed values to the cloud at fixed intervals. The cloud and application layer stores the received records, maintains the time-series history, and renders the dashboard through which the user observes live values and past trends from a browser or mobile device.

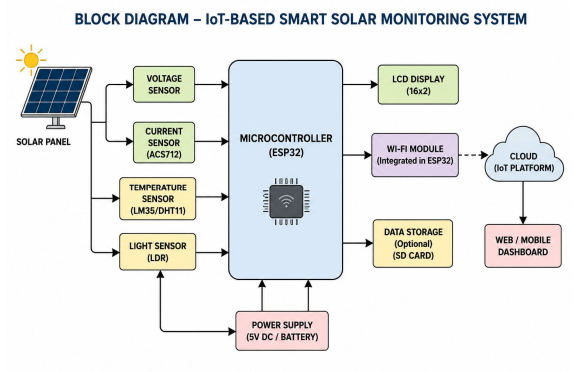


Figure 1: Block diagram of the proposed IoT-based smart solar monitoring system.

3.2. Hardware Components

Table 1 summarises the principal hardware components used in the system.

Table 1. Hardware Components

Components	
Solar PV Panel (20 W)	Voltage Regulator
ESP32 Microcontroller	Resistor
Voltage Sensor (0–25 V)	Capacitor
ACS712 Current Sensor	Diode
Temperature Sensor	Vero Board
Light Sensor (LDR)	PBT Connector
LCD Display (16 × 2)	Bug Strip
Power Supply (5 V DC)	Connecting Wires
SD Card / Enclosure	Signal Filter (RC)

3.3. Parameter Sensing and Signal Conditioning

Panel voltage is measured through a resistive divider that scales the panel terminal voltage into the 0–3.3 V input range of the ESP32 ADC, following the relation $V_{adc} = [R2 / (R1 + R2)] \times V_{pv}$. The panel current is measured by an ACS712 Hall-effect sensor whose output follows $V_{out} = V_{cc}/2 + (S \times I_{pv})$, where S is the device sensitivity. Because the sensor output is centred on half the supply voltage, the zero-current null point is measured at start-up and stored so that offset drift does not accumulate in the energy total. Small RC filters are placed on both analog channels to suppress switching noise, and the readings are averaged over several consecutive samples before use. The overall measurement error is dominated by the ADC resolution, the tolerance of the divider resistors, and the temperature-dependent offset of the current sensor.

3.4. On-Device Data Processing

The ESP32 converts the filtered ADC counts into engineering units and computes the derived quantities locally rather than transferring raw values to the cloud. The instantaneous DC power is obtained as $P(t) = V(t) \times I(t)$, and the energy generated over a period is obtained by numerical integration of the power samples, $E = \Sigma [P(t) \times \Delta t]$, evaluated by the trapezoidal rule over the sampling interval Δt. Performing this computation at the edge keeps the accumulated energy correct even when a transmission is missed, and reduces the volume of data sent over the network.

3.5. Cloud Transmission and Alerting

The processed readings are assembled into a record and transmitted over Wi-Fi to the cloud platform, where each measured quantity is written to a separate channel field. The dashboard renders the received records as live values and as time-series charts, so that the daily generation profile and any departure from it can be observed remotely. Threshold-based logic is applied to the same values to flag abnormal conditions: an open circuit is indicated by a current close to zero while the terminal voltage remains near the open-circuit value during daylight, a short circuit or string fault by a collapsed voltage with current still flowing, and soiling or degradation by a sustained fall in the delivered power under comparable irradiance conditions. When such a condition persists, a notification can be issued so that the fault is addressed before a significant quantity of energy is lost.

4. METHODOLOGY

The development of the proposed system followed a structured, iterative engineering approach consisting of the following stages.

1. Requirement Analysis and Component Selection: The system specifications, including the voltage and current ranges, the required accuracy, and the data update interval, were

defined first. The ESP32, the voltage and current sensors, the temperature and light sensors, and the supporting components were then selected on the basis of cost, accuracy, and mutual compatibility.

2. Circuit Design and Sensor Interfacing: The sensors were connected to the appropriate microcontroller pins. A resistive divider was used for safe voltage measurement, and suitable analog interfaces were provided for the current, temperature, and light sensors. The circuit was powered from a regulated 3.3 V supply.

3. Firmware Development: Firmware was written in Arduino C++ to initialise the sensors and the Wi-Fi interface, acquire the sensor readings, convert them into engineering units, and compute the instantaneous power and the accumulated energy. The processed data were then formatted for transmission to the cloud.

4. Cloud Platform and Dashboard Configuration: A cloud channel was created to receive the transmitted data, with a separate field allocated to each measured quantity. The dashboard was configured to display live values together with time-series charts for voltage, current, power, and energy.

5. Alerting Mechanism: Threshold-based rules were implemented to detect abnormal operating conditions and low power output. When a threshold is crossed, a notification is generated so that the condition can be investigated.

6. Testing, Calibration and Validation: The assembled system was tested first with a regulated DC supply and subsequently with an actual solar panel. The sensor readings were compared against a calibrated multimeter and the calibration coefficients were adjusted accordingly. Continuous operation and simulated fault conditions were then exercised to verify the stability of the logging and the correctness of the alert logic.

5. IMPLEMENTATION

The system was assembled on a Vero board carrying the ESP32, the voltage-divider network, the ACS712 current sensor, the temperature and light sensors, the LCD module, the voltage regulator, and the associated passive components. The solar panel was connected to the sensing network through PBT connectors, with the voltage divider placed across the panel terminals and the current sensor placed in series with the load, so that the delivered power could be computed from simultaneous measurements of the two quantities. The ESP32 was programmed to sample the sensors at a fixed interval, convert the readings, compute the power and energy, update the local display, and publish the record to the cloud channel. A demonstration model was constructed in which the monitored panel supplies a small lighting load, allowing the relationship between the generated power and the connected load to be observed directly during testing. The assembled prototype is shown in Fig. 2. Before deployment, the assembly was verified for correct sensor polarity, stable supply voltage, and reliable Wi-Fi association, and the null point of the current sensor was recorded with the load disconnected.

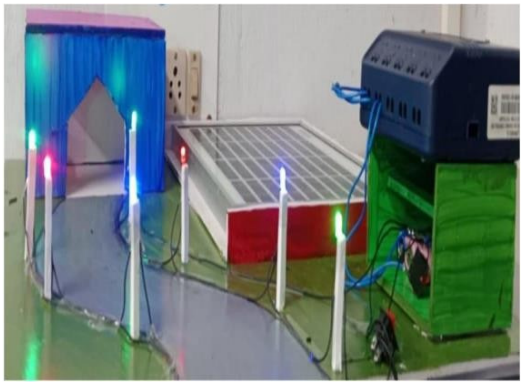


Figure 2: Hardware implementation of the prototype with the monitored panel and connected load.

6. RESULTS AND DISCUSSION

The developed system was tested outdoors with the solar panel exposed to natural sunlight, and it operated continuously for the duration of each test without loss of connectivity. The panel voltage, current, power, and accumulated energy were acquired at the configured interval, displayed locally, and transmitted successfully to the cloud channel. The field arrangement and the resulting cloud dashboard are shown in Fig. 3, in which the four channel fields correspond to the measured voltage, the measured current, the computed power, and the accumulated energy.

Figure 3: Field deployment of the monitored panel and the corresponding cloud dashboard.

The recorded voltage and current traces fluctuated continuously about their mean values over the observation window. These fluctuations follow the variation of incident irradiance caused by passing cloud and by the changing solar angle, and confirm that the system resolves short-term changes in generation rather than merely reporting an average. The power channel, being the product of the two measured quantities, reflected the same variation, while the energy channel increased monotonically as the power samples were integrated, with a discontinuity corresponding to the point at which the accumulator was reset during testing. The ability to observe these traces remotely is the principal practical benefit of the system, since a comparable manual measurement campaign would require continuous attendance at the installation.

The interpretation of the measured data follows directly from the electrical characteristic of a photovoltaic module. As shown in Fig. 4, the current remains nearly constant over the lower part of the voltage range and then falls sharply as the open-circuit voltage is approached, so that the product of current and voltage passes through a maximum power point. Because the operating point of a directly connected load does not generally coincide with this maximum, the measured power is expected to lie below the peak of the characteristic; the recorded traces are consistent with operation on the rising portion of the power curve. The characteristic also explains the observed sensitivity of the output to irradiance and temperature, since a fall in irradiance lowers the current plateau while a rise in cell temperature reduces the open-circuit voltage.

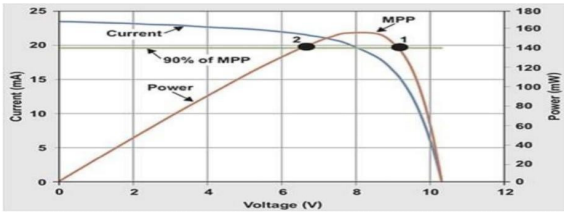


Figure 4: Current–voltage and power–voltage characteristics of a photovoltaic module showing the maximum power point.

Comparison of the sensor readings with a calibrated multimeter during the validation stage showed agreement within a few per cent for both voltage and current, consistent with the expected error budget arising from the ADC resolution, the divider tolerance, and the offset of the current sensor. The architecture also scales without redesign: because each unit publishes independently to its own cloud channel, additional panels can be monitored by replicating the node rather than by modifying the system structure. Overall, the results demonstrate that the proposed system provides a practical and inexpensive means of observing photovoltaic performance continuously and of detecting deviations early, although the accuracy of the measurements depends on careful calibration, and continuity of operation depends on the stability of the Wi-Fi connection and of the supply to the monitoring node.

7. CONCLUSION

This work presented the design and implementation of an IoT-based smart solar monitoring system built around the ESP32 microcontroller. The system measures the voltage, current, temperature, and light intensity associated with a photovoltaic panel, computes the instantaneous power and the accumulated energy at the edge, displays the values locally, and transmits them over Wi-Fi to a cloud dashboard that provides live readings and historical trends together with threshold-based fault indication. Field testing confirmed continuous acquisition, correct computation of the derived quantities, and reliable remote visualisation of the recorded data. The system therefore converts photovoltaic monitoring from a periodic, manual record-keeping exercise into a continuous and remotely accessible process, at a cost appropriate to domestic, institutional, and agricultural installations. Future work will extend the platform with maximum power point tracking,

longer-term storage for seasonal performance analysis, and automated diagnosis of the fault signatures identified in this study.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype.

REFERENCES

[1] Swathi, G. V., Ramesh, D., Krishna, S. V., & Ganesh, B. (2022). IoT based solar power monitoring using Arduino. International Journal for Research in Applied Science and Engineering Technology, 10(6), 259–263. doi: https://doi.org/10.22214/ijraset.2022.43820

[2] Sawashere, S., Jadhav, P., Lanjewar, D., Gokhale, D., Kokate, P., & Asutkar, Y. (2022). Solar power panel monitoring and data acquisition system. International Journal for Research in Applied Science and Engineering Technology, 10(8), 1748–1751. doi: https://doi.org/10.22214/ijraset.2022.46491

[3] Nkinyam, C. M., Ujah, C. O., Asadu, C. O., Anyaka, B., & Olubambi, P. A. (2025). Development of a low-cost monitoring device for solar electric (PV) system using Internet of Things (IoT). Results in Engineering, 28, 107324. doi: https://doi.org/10.1016/j.rineng.2025.107324

[4] Trang, T. T., Toi, L. T., & Khoa, T. H. (2024). Design of a real-time solar power monitoring and controlling system using Internet of Things. Advances in Electrical and Electronic Engineering, 22(2). doi: https://doi.org/10.15598/aeee.v22i2.5709

[5] Rao, C. K., Sahoo, S. K., & Yanine, F. F. (2024). An IoT-based intelligent smart energy monitoring system for solar PV power generation. Energy Harvesting and Systems, 11(1), 20230015. doi: https://doi.org/10.1515/ehs-2023-0015

[6] Gopal, M., Prakash, T., Ramakrishna, N. V., & Yadav, B. P. (2020). IoT based solar power monitoring system. IOP Conference Series: Materials Science and Engineering, 981(3), 032037. doi: https://doi.org/10.1088/1757-899X/981/3/032037

[7] Olayiwola, T. N., Hyun, S.-H., & Choi, S.-J. (2024). Photovoltaic modeling: A comprehensive analysis of the I–V characteristic curve. Sustainability, 16(1), 432. doi: https://doi.org/10.3390/su16010432

[8] Domínguez-Bolaño, T., Barral, V., Escudero, C. J., & García-Naya, J. A. (2024). An IoT system for a smart campus: Challenges and solutions illustrated over several real-world use cases. Internet of Things, 25, 101099. doi: https://doi.org/10.1016/j.iot.2024.101099

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (2026) Page 143-148 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Smart Drunk Driving Prevention Using IoT and Computer Vision

[1]Soumyadip Sahoo, [2]Sohan Das, [3]Chanchal Kumar De

[1,2]UG Students, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

[3]Professor & Head, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

ABSTRACT

Drunk driving is a major preventable cause of road fatalities, prompting increased regulatory attention to in-vehicle impairment detection. This paper presents a low-cost Smart Drunk Driving Prevention System using two independent detection channels. An MQ-3 alcohol sensor interfaced with an ESP32 DevKit V1 detects alcohol above a preset threshold and activates an alarm while inhibiting the ignition through a relay. In parallel, an ESP32-CAM streams driver video to a host application using OpenCV and MediaPipe Face Mesh to calculate Eye Aspect Ratio and Mouth Aspect Ratio for drowsiness and yawning detection. A NEO-6M GPS module provides vehicle location, transmitted to a cloud dashboard during impairment events. Bench tests demonstrated reliable engine locking, drowsiness and yawning detection, and location-data transmission. The prototype costs approximately ₹2,475. Key limitations include MQ-3 cross-sensitivity and dependence on Wi-Fi connectivity.

KEYWORDS: Drunk Driving Prevention, Drowsiness Detection, GPS Tracking, Embedded Systems, Vehicle Safety.

1. INTRODUCTION

Road traffic collisions are a major cause of death and injury worldwide, and a substantial proportion of them involve drivers impaired by alcohol. Alcohol degrades judgement, lengthens reaction time, and reduces the precision with which a driver controls a vehicle, placing at risk not only the driver but every other road user. Conventional countermeasures are enforcement-based: roadside checkpoints, breath testing by police officers, and legal sanction applied after an offence or a collision has already occurred.

Such measures depend on the presence of an officer, are applied intermittently, and cannot scale to the number of vehicles in use. Present-day vehicles possess no intrinsic mechanism for verifying driver sobriety before ignition, so there is a clear need for an affordable, automated, in-vehicle system that screens the driver continuously, prevents the vehicle from being operated when an unsafe condition is detected, and communicates the event without depending on external intervention.

Regulatory attention has moved in the same direction. Section 24220 of the United States Infrastructure Investment and Jobs Act of 2021 directs the National Highway Traffic Safety Administration (NHTSA) to prescribe a Federal Motor Vehicle Safety Standard requiring new passenger vehicles to be equipped with advanced drunk and impaired driving prevention technology, defined as a system able to passively detect impairment and then prevent or limit vehicle operation [1]. As of the time of writing the rulemaking has reached only the advance notice stage [2]; NHTSA has attributed the delay to the immaturity of passive impairment-detection technology and to the absence of objective test procedures. The direction of travel, however, is established, and low-cost experimental platforms that combine several independent impairment indicators are of direct relevance to it.

This paper presents a low-cost prototype that addresses the problem through three complementary subsystems. A breath-alcohol sensor detects alcohol above a preset threshold and inhibits the ignition circuit. A camera and a facial-landmark algorithm assess whether the driver is showing signs of drowsiness or yawning, and inhibit the ignition on the same basis. A satellite positioning module supplies the vehicle location, which is relayed to a cloud dashboard

*Corresponding Address: sahoosoumyadip04@gmail.com DOI 10.5281/zenodo.22038973

when an impairment event occurs. The objective is an affordable solution that unifies alcohol detection, driver-state monitoring, engine control, and emergency alerting in a single system.

The remainder of this paper is organised as follows. Section II reviews related work and identifies the research gap. Section III describes the proposed system and its hardware. Section IV sets out the firmware and vision methodology, and Section V the implementation. Section VI reports the experimental results, Section VII the limitations of the prototype, and Section VIII concludes with the future scope of the work.

2. LITERATURE REVIEW

Rafidi and Nik Ismail [3] developed an alcohol detection and ignition-locking system using an MQ-3 sensor, an Arduino Mega 2560, an LCD, and motor-control circuitry. The system disabled vehicle operation when the detected alcohol level exceeded a predefined threshold, demonstrating the effectiveness of sensor-based ignition control while also identifying the calibration limitations of MQ-3 sensors. Berad et al. [4] presented an Arduino-based drunk-driving detection system incorporating an MQ-3 sensor and GSM communication, which could detect alcohol, prevent ignition, and transmit an alert to a remote user, marking the transition from standalone detection towards connected vehicle-safety systems.

Nossam et al. [5] extended this approach with an alcohol-triggered accident detection and alert system integrating GSM, GPS, and an ESP32, so that the combination of sensing, communication, and location tracking provided improved situational awareness during potentially dangerous driving events. Mountzouris et al. [6] more recently described a smart vehicle safety and security system based on Arduino and the Internet of Things that addresses both drunk driving and theft, establishing a foundation for IoT-enabled vehicle safety applications.

A parallel body of work addresses driver state rather than blood alcohol. Yan et al. [7] investigated real-time drowsiness detection using PERCLOS and image-processing techniques, monitoring eye state and generating an alert when prolonged closure indicated fatigue. Kim et al. [8] developed a real-time driver monitoring system combining facial-landmark-based eye-closure detection with head-pose recognition, and Zambrano et al. [9] applied facial landmarks and deep learning to the simultaneous detection of fatigue, distraction, and emotion. This line of work is important because unsafe driving occurs in the absence of alcohol as well as in its presence.

A third and smaller group combines the two modalities. Muiz et al. [10] reported an IoT-based driver safety system performing real-time drowsiness and alcohol monitoring through vision and multi-sensor integration, and Rani et al. [11] presented an IoT-enabled real-time driver monitoring system for drowsiness and alcohol detection built on Arduino and OpenCV. These works demonstrate the value of multimodal sensing but generally stop short of combining it with vehicle immobilisation and location-aware alerting in one platform.

Table 1 summarises the capabilities of the representative systems discussed above. Alcohol-centred designs provide ignition control but no assessment of driver state, and vision-centred designs assess driver state but exercise no control over the vehicle. Where the two are combined, the resulting systems generally lack either the engine-inhibit path or the location-reporting path. Remaining challenges across the literature include sensor calibration, false detection, varying illumination, computational limits, and reliable multimodal decision-making. The system proposed here addresses this gap by integrating alcohol sensing, computer-vision-based driver monitoring, vehicle immobilisation, and GPS-based cloud alerting within a single real-time framework.

Table 1. Capabilities of representative driver-safety systems. Alc. = breath-alcohol sensing; Drows. = drowsiness monitoring; GPS = location reporting; Lock = engine inhibition.

Work	Alc.	Drows.	GPS	Lock
Rafidi & Nik Ismail [3]	Yes	–	–	Yes
Berad et al. [4]	Yes	–	–	Yes
Nossam et al. [5]	Yes	–	Yes	–
Mountzouris et al. [6]	Yes	–	Yes	Yes
Yan et al. [7]	–	Yes	–	–
Kim et al. [8]	–	Yes	–	–
Zambrano et al. [9]	–	Yes	–	–
Muiz et al. [10]	Yes	Yes	–	–
Rani et al. [11]	Yes	Yes	–	–
This work	Yes	Yes	Yes	Yes

3. PROPOSED SYSTEM

3.1. System Architecture

The system uses an ESP32 DevKit V1 as the central processor, interfaced with an MQ-3 alcohol sensor, ESP32-CAM, 16 × 2 I2C LCD, two-channel relay, and NEO-6M GPS module. Alcohol sensing is performed at the edge, while facial video is processed on an external host using OpenCV and MediaPipe to detect drowsiness and yawning. When impairment is detected, the relay interrupts the ignition, GPS location is sent to the cloud dashboard, and the LCD displays the system status.

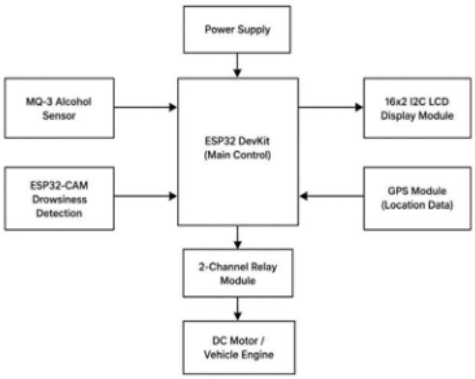


Figure 1: Block diagram of the proposed system.

3.2. Hardware Components

Table 2 lists the principal hardware components and their functions in the prototype.

Table 2. Key hardware components.

Component	Function
ESP32 DevKit V1	Main controller
MQ-3 sensor	Breath-alcohol sensing
ESP32-CAM	Vision unit
NEO-6M GPS	Location tracking
2-channel relay	Engine-lock actuation
16 × 2 I2C LCD	Status display
BO motor + 9 V battery	Engine surrogate

3.3. Control Unit

The ESP32 DevKit V1 is the decision-making element of the system. It receives the alcohol flag from the MQ-3 sensor and the drowsiness status from the vision subsystem, evaluates both, and determines whether the vehicle may be operated or whether a safety action is required. The controller drives the relay that inhibits the engine, updates the 16 × 2 I2C LCD to reflect the current state, and reads the GPS receiver, forwarding position data to the cloud whenever a valid fix is available.

3.4. Alcohol Detection and Engine Locking

The alcohol-detection subsystem is built around the MQ-3 gas sensor, which requires a warm-up interval of approximately 30 seconds after power-up before its output is treated as valid. Once ready, the ESP32 samples the sensor continuously. When the measured concentration exceeds the configured threshold, the controller de-energises the relay, interrupting power to the ignition circuit so that the BO motor representing the engine stops, and writes the corresponding status message to the LCD.

3.5. Vision-Based Drowsiness Detection

The drowsiness subsystem uses the ESP32-CAM to observe the driver's face and stream the video to a host application built with MediaPipe Face Mesh. The application locates the facial landmarks around the eyes and mouth, and uses them to determine whether the eyes have remained closed for a sustained period, which indicates the onset of sleep, and whether the mouth is open in the characteristic pattern of a yawn. When drowsiness is confirmed, the host signals the ESP32, which locks the engine and activates the alarm through the same output path used for the alcohol channel, so that either indicator alone is sufficient to immobilise the vehicle.

3.6. GPS and Communication Unit

The NEO-6M GPS module supplies the vehicle position. The ESP32 parses the incoming NMEA sentences to extract latitude and longitude. When an impairment event occurs and a valid fix is available, the coordinates are formatted and transmitted over an HTTPS connection to a cloud dashboard, allowing the location associated with an alcohol or drowsiness event to be viewed remotely. The positioning and communication subsystem therefore adds an emergency-notification capability to the engine-locking function.

4. METHODOLOGY

4.1. Firmware

The ESP32 firmware is developed in the Arduino IDE and uses the WiFi.h, HTTPClient.h, TinyGPS++.h, HardwareSerial.h, Wire.h, and LiquidCrystal_I2C.h libraries. The alcohol-

detection routine reads a digital threshold flag from the comparator output of the MQ-3 module. When the flag indicates the presence of alcohol, the firmware sets the relay pin low, disengaging the simulated ignition, updates the LCD, and, if a valid GPS fix is available, serialises the latitude and longitude into a JSON payload delivered by HTTPS POST to the cloud endpoint.

4.2. Vision-Based Drowsiness Detection

The ESP32-CAM streams MJPEG video over Wi-Fi to a host application built with OpenCV and MediaPipe Face Mesh. Six landmark points per eye are used to compute the Eye Aspect Ratio (EAR):

$$EAR = (|p2 - p5| + |p3 - p6|) / (2 \times |p1 - p4|)$$

where p1 and p4 are the horizontal eye corners and p2, p3, p5 and p6 are the vertical eyelid landmarks. An open eye typically yields an EAR between 0.28 and 0.38; a sustained EAR below 0.22 for 20 consecutive frames is classified as drowsiness. A parallel Mouth Aspect Ratio (MAR) computed over 20 mouth landmarks flags yawning when it exceeds an empirically determined threshold. When drowsiness is confirmed, the host sends an HTTP GET request to the /drowsy endpoint exposed by the ESP32, which the firmware treats as an independent trigger for engine locking and alarm activation, mirroring the alcohol-detection response path.

4.3. System Operation

The complete operating cycle is shown in Fig. 2. On power-up the firmware initialises the peripherals, connects to the Wi-Fi network, and waits for the MQ-3 warm-up interval to elapse. It then enters a continuous sense–decide–act loop: the alcohol flag is read first, and if no alcohol is present the drowsiness status reported by the vision subsystem is evaluated. If neither indicator is raised, the engine remains enabled and the LCD reports normal operation. If either indicator is raised, the relay is de-energised, the alarm is activated, and, when a valid GPS fix is available, the vehicle position is transmitted to the cloud dashboard. The loop then repeats so that a change in the driver's condition is acted upon promptly.

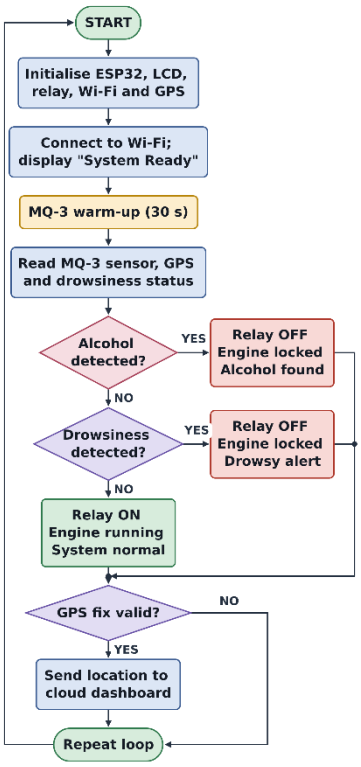


Figure 2: Operational flowchart of the proposed system.

5. IMPLEMENTATION

The prototype was constructed as a low-cost proof of concept on a breadboard, using an ESP32 DevKit V1 as the controller. The ESP32-CAM, MQ-3 alcohol sensor, two-channel relay module, 16 × 2 I2C LCD, and NEO-6M GPS module were connected according to the circuit-level design shown in Fig. 3. A BO motor powered by a 9 V battery was used in place of an engine to demonstrate the ignition-inhibit function. All components are readily available, and the total cost of the assembly was approximately ₹2,475 (about US$28).

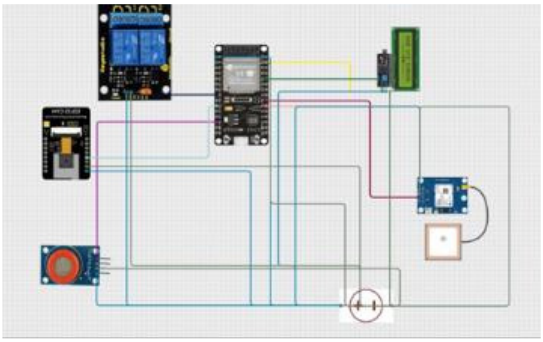


Figure 3: Circuit-level wiring diagram of the prototype.

The relay was wired to act as the ignition switch for the BO motor, so that when the MQ-3 sensor

reported an alcohol concentration above the configured threshold the ESP32 changed the relay state to stop the motor and displayed the corresponding status on the LCD. The ESP32-CAM was configured to stream MJPEG video over Wi-Fi to the host computer, where the Python application located the eye and mouth landmarks and applied the EAR and MAR criteria described above. The NEO-6M module supplied the coordinates that were serialised and posted to the cloud dashboard. The assembled prototype was then exercised under controlled alcohol-vapour exposure and simulated fatigue in order to test the alcohol detection, engine-locking, drowsiness detection, and location-logging functions together.

6. RESULTS AND DISCUSSION

6.1. Alcohol Detection and Engine Control

With the relay energised by default, the BO motor operated continuously in the absence of alcohol. On exposure of the sensor head to an alcohol source, the firmware consistently de-energised the relay within the response window of the sensor, halted the motor, and updated the LCD to indicate that alcohol had been found and the engine locked. The two states are shown in Fig. 4 and Fig. 5.

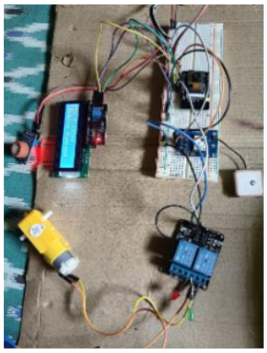


Figure 4: Engine running, no alcohol detected.

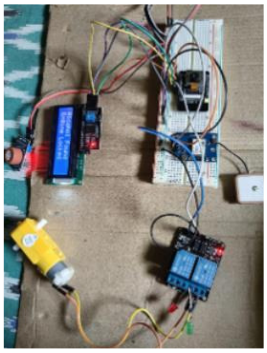


Figure 5: Engine locked, alcohol detected.

6.2. Drowsiness and Yawning Detection

Across the bench trials, sustained eye closure reliably drove the EAR below the 0.22 threshold within the 20-frame observation window, and exaggerated mouth opening reliably raised the MAR-based yawning flag. Fig. 6 shows a representative frame in which an EAR of 0.13 and a MAR of 1.56 produced simultaneous drowsiness and yawning classifications.

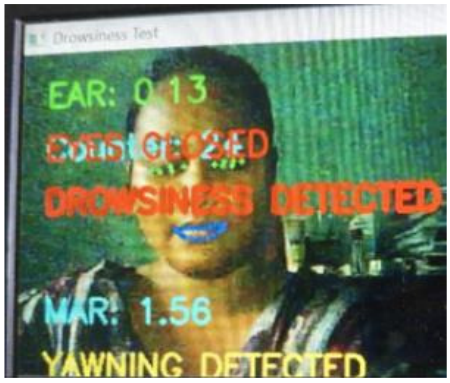


Figure 6: Live EAR and MAR based drowsiness and yawning classification during bench testing.

6.3. GPS Tracking and Cloud Relay

Whenever an alcohol event coincided with a valid GPS fix, the latitude and longitude were successfully relayed to the cloud dashboard, confirming the end-to-end telemetry path from the on-board receiver to the remote viewer.

6.4. Discussion

The results indicate that a bill of materials below US$30 can support a functioning multi-stage interlock suitable for classroom demonstration and early-stage research. Compared with the single-modality designs surveyed in Section II and summarised in Table 1, the addition of an independent EAR and MAR cross-check means that the two impairment indicators are derived from physically unrelated measurements, so a transient false positive on the alcohol sensor is not corroborated by the vision channel.

7. CONCLUSION

The proposed low-cost Smart Drunk Driving Prevention System integrates alcohol sensing, vision-based drowsiness monitoring, engine locking, and GPS telemetry into a unified safety framework. Experimental results demonstrated effective alcohol detection, drowsiness and yawning classification, and reliable location-data transmission. Future work will focus on

multimodal physiological sensing, improved onboard AI processing, secure telemetry, driver authentication, and ethanol-selective sensing to enhance reliability and support vehicle-grade deployment.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Dr. Ritwik Haldar, Department of Electronics and Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities used to build and test the prototype.

REFERENCES

[1] Infrastructure Investment and Jobs Act, Pub. L. No. 117-58, § 24220, 135 Stat. 429 (2021) (advanced drunk and impaired driving prevention technology).

[2] National Highway Traffic Safety Administration, "Advanced impaired driving prevention technology," Advance Notice of Proposed Rulemaking, *Federal Register*, vol. 89, p. 830, Jan. 5, 2024. Docket No. NHTSA-2022-0079, RIN 2127-AM50.

[3] M. N. A. Rafidi and N. M. A. Nik Ismail, "Development of alcohol detection with ignition lock system for vehicles," *Evolution in Electrical and Electronic Engineering*, vol. 2, no. 2, pp. 173–181, 2021. [Online]. Available: https://periodical.uthm.edu.my/index.php/eeee/article/view/3541

[4] S. Berad, S. Aher, D. Varpe, and S. Dhuttargi, "Elite vehicle controller: Drunk driving detection with vehicle ignition locking using GSM," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, no. 6, pp. 273–277, 2022. doi: 10.22214/ijraset.2022.43032

[5] S. C. Nossam, G. Pulastya, R. A. Katakam, and M. Khanna, "Alcohol-triggered accident detection and alert system with GSM, GPS, ESP32 integration," in *Proc. 2nd Int. Conf. Intelligent and Sustainable Power and Energy Systems (ISPES)*, 2024, pp. 82–89. doi: 10.5220/0013577200004639

[6] P. Mountzouris, A. Papadakis, G. Pagiatakis, L. Dritsas, N. Voudoukis, and K. Nanos, "A smart vehicle safety-security system for the prevention of drunk driving and theft based on Arduino and the Internet of Things," *Electronics*, vol. 15, no. 1, art. no. 70, 2026. doi: 10.3390/electronics15010070

[7] J.-J. Yan, H.-H. Kuo, Y.-F. Lin, and T.-L. Liao, "Real-time driver drowsiness detection system based on PERCLOS and grayscale image processing," in *Proc. Int. Symp. Computer, Consumer and Control (IS3C)*, 2016, pp. 243–246. doi: 10.1109/IS3C.2016.72

[8] D. Kim, H. Park, T. Kim, W. Kim, and J. Paik, "Real-time driver monitoring system with facial landmark-based eye closure detection and head pose recognition," *Scientific Reports*, vol. 13, art. no. 18264, 2023. doi: 10.1038/s41598-023-44955-1

[9] T. Zambrano, L. Arias, E. Haro, V. Santos, and M. Trujillo-Guerrero, "Driver monitoring system using computer vision for real-time detection of fatigue, distraction and emotion via facial landmarks and deep learning," *Sensors*, vol. 26, no. 3, art. no. 889, 2026. doi: 10.3390/s26030889

[10] B. Muiz, A. Hasib, M. F. Ahmed, A. Al Zubaer, R. Hossen, D. Khushi, and A. Rahman, "IoT-based driver safety system with real-time drowsiness and alcohol monitoring via vision and multi-sensor integration," in *Proc. Int. Conf. Quantum Photonics, Artificial Intelligence, and Networking (QPAIN)*, 2025, pp. 1–6. doi: 10.1109/QPAIN66474.2025.11171722

[11] U. B. Rani, M. Imran, S. Nishanth, V. Naik, and P. Ashok, "IoT-enabled real-time driver monitoring system for drowsiness and alcohol detection using Arduino and OpenCV," *EPJ Web of Conferences*, vol. 363, art. no. 02002, 2026. doi: 10.1051/epjconf/202636302002

[12] H. Farooq, A. Altaf, F. Iqbal, J. Castanedo Galán, D. Gavilanes Aray, and I. Ashraf, "DrunkChain: Blockchain-based IoT system for preventing drunk driving-related traffic accidents," *Sensors*, vol. 23, no. 12, art. no. 5388, 2023. doi: 10.3390/s23125388

Int.J.HIT.TRANSC:ECCN.Vol.13: Special Issue NSEFCCIC (202) Page 149-153 ISSN: 0973-6875

Available Online at www.hithaldia.in/locate/ECCN

ORIGINAL CONTRIBUTION

Privacy-Aware Fingerprint Exposure Detection in Photographs Using Computer Vision and Vision APIs

Mridul Banerjee

UG student, Department of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, West Bengal

ABSTRACT

Photographs shared through social platforms can unintentionally expose biometric information when the friction ridge surface of a finger is visible. This work presents a lightweight prototype that identifies photographs with potential fingerprint exposure and applies a privacy-preserving image transformation before sharing. The proposed pipeline combines OpenCV for image loading, resizing, cropping and local masking; MediaPipe Hand Landmarker for locating hands and finger landmarks; and an optional vision API as a secondary semantic assessment layer. Instead of training a new deep-learning model, the system composes ready-made computer-vision components into a sense-assess-protect workflow. After a hand is localized, finger regions are estimated from the landmark geometry and relevant image crops are evaluated for visible biometric detail. When the risk exceeds a configurable threshold, the system automatically blurs or masks the sensitive region while leaving unrelated image content unchanged. The prototype is intended as an accessible demonstration of privacy-enhancing AI for cognitive systems rather than a certified biometric presentation-attack detector. Functional evaluation indicates that the integrated pipeline can detect hand-containing images, identify potentially exposed finger regions, and generate a protected output with limited modification to non-sensitive content. The study also highlights limitations caused by viewpoint, illumination, image resolution and the distinction between visible finger texture and a fingerprint image of matching quality.

KEYWORDS: Fingerprint privacy, biometric exposure, computer vision, vision API, image anonymization.

1. INTRODUCTION

Biometric authentication systems increasingly rely on fingerprints because friction-ridge patterns are distinctive and persistent. The same persistence, however, creates a privacy problem: unlike a password, a fingerprint cannot simply be replaced after exposure. Social-media photographs can unintentionally contain biometric information when users display a thumb, an index finger, or a V-shaped hand gesture. Malhotra et al. [1] examined this problem through finger-selfies and reported that ridge-valley details visible in social-media photographs can potentially be matched with livescan fingerprints. Their work established the relevance of treating ordinary photographs as a possible source of biometric leakage.

Subsequent research has explored image-level protection rather than only restricting access to the original photograph. Li et al. [2] proposed a hierarchical perceptual noise-injection approach for social-media fingerprint privacy, with the objective of reducing biometric leakage while retaining visual quality. SocialGuard [3] similarly investigated adversarial protection of social images against automated visual analysis. These studies demonstrate that privacy protection can be introduced at the image-sharing stage.

The present work takes a practical mini-project approach to the same problem. Instead of developing a new large-scale neural network, the proposed system integrates ready-made components. OpenCV provides conventional image-processing operations, MediaPipe supplies hand and landmark localization, and a vision API can provide an additional semantic judgement about whether a visible finger region presents potential biometric exposure. The protected image is then generated locally. This design is intended for demonstration and educational deployment in the cybersecurity, privacy and trust track of NSEFCCIC 2026.

The principal objective is therefore not to reconstruct a fingerprint or to claim that every

*Corresponding Address: mridulbanerjee599@gmail.com DOI 10.5281/zenodo.22039049

photograph can be converted into a usable biometric template. The objective is to detect a potential exposure condition early and automatically reduce the visibility of the sensitive region. This distinction is important because fingerprint image quality strongly affects matching performance, as discussed by NIST [4], and a casual photograph is not equivalent to a scanner-quality fingerprint sample.

2. LITERATURE REVIEW

Malhotra et al. [1] proposed a privacy-preserving anonymization method for finger-selfies shared on social media. The authors demonstrated that finger-selfie details may contain biometric information and introduced an adversarial learning approach that modifies the image while attempting to preserve its visual usefulness. They also introduced the Social-Media Posted Finger-selfie database, providing a basis for studying biometric leakage from online photographs.

Li et al. [2] proposed Hierarchical Perceptual Noise Injection for social-media fingerprint privacy protection. The authors focused on adding visually unobtrusive perturbations to fingerprint-bearing regions so that biometric leakage is reduced without severely degrading the appearance of the image. Their work motivates the present study's use of localized protection rather than indiscriminate image destruction.

Zhang et al. [3] proposed SocialGuard, an adversarial example-based privacy-preserving technique for social images. The authors showed that automated object detectors can extract sensitive information from shared photographs and investigated adversarial modifications that reduce such detection while maintaining image quality. The present prototype differs in scope by concentrating on potential fingerprint exposure and by using a lightweight rule-and-API pipeline rather than training a new adversarial network.

NIST developed fingerprint image-quality measures in which image quality is treated as a predictor of expected matcher performance [4], [5]. These works are relevant because a photograph containing a visible finger should not automatically be interpreted as a high-quality fingerprint sample. The proposed system therefore reports potential exposure rather than asserting successful biometric recovery.

ISO/IEC 30107-1 defines a framework for biometric presentation attack detection and establishes terminology for biometric spoofing and related attacks [6]. The standard also makes clear that a specific detection method is not prescribed by the framework. The present system is consequently described as a privacy exposure detector and mitigation prototype, not as a standards-compliant PAD system.

3. PROPOSED SYSTEM

The proposed architecture contains five functional stages: image acquisition, local image preprocessing, hand and landmark localization, risk assessment, and privacy protection. Fig. 1 summarizes the information flow. A photograph is first loaded and normalized using OpenCV. MediaPipe Hand Landmarker then identifies the hand and returns 21 hand landmarks for each detected hand [7]. These landmarks provide the geometry needed to locate finger segments without training a separate hand detector.

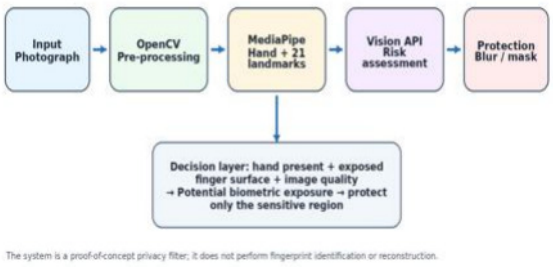


Figure 1: Proposed architecture for fingerprint-exposure risk detection.

The landmark output is converted into pixel coordinates and used to define candidate finger regions. The system considers the visibility of the finger surface, its size in the image, and basic image quality indicators. An optional vision API is then used as a secondary semantic check. Modern multimodal vision APIs can accept image data and return classification or structured detection results [8]. In the prototype, the API is instructed to judge whether the candidate region visibly exposes fingerprint-like ridge detail and to return a simple risk label rather than any identity information.

The final decision is deliberately conservative. If the combined evidence suggests a potential biometric exposure, the system protects the candidate region. Otherwise, the original image can be retained or the user can be shown a

warning. This architecture separates detection from protection, allowing the protection operation to remain local even when an external API is used.

4. IMPLEMENTATION METHODOLOGY

The implementation is written in Python. OpenCV is responsible for reading images, resizing frames, converting colour spaces, extracting regions of interest, calculating simple texture or contrast measures, and applying Gaussian blur or masking. The OpenCV documentation provides image I/O, filtering, geometric transformations, segmentation and related image-processing functions [9].

MediaPipe Hand Landmarker is used as the hand-localization component. The task provides 21 hand landmarks and supports image, video and live-stream processing modes [7]. In this project, the landmarks are not treated as fingerprint features. They are only used to determine where a finger is located and which portion of the photograph should be examined or protected.

The API stage is implemented as a normal client-server request. The application loads the candidate crop, sends it with a fixed instruction such as 'Assess whether the visible finger surface contains potentially recoverable fingerprint-like ridge detail; return LOW, MEDIUM or HIGH risk,' and parses the returned label. The current Gemini API documentation supports image inputs through inline image data or uploaded files and provides multimodal image understanding capabilities [8]. The API is therefore used as a ready-made visual reasoning component rather than as a model developed by the authors.

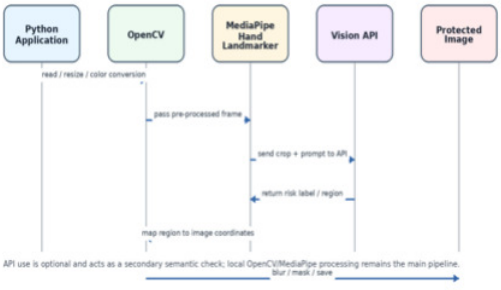


Figure 2: Library and API interaction workflow.

A simple configurable risk score can combine three normalized indicators: finger exposure F, visible surface V, and image quality Q. The score is expressed as R = w1F + w2V + w3Q, where the weights sum to one. A threshold tau determines whether the candidate region is protected. The score is a system-level decision aid, not a biometric recognition score. Fig. 3 illustrates the decision process.

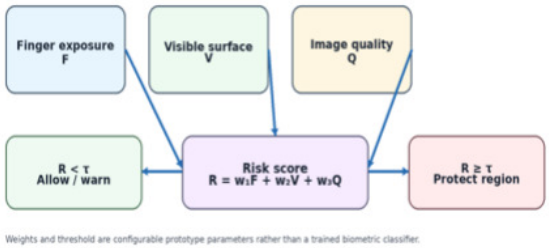


Figure 3: Threat scoring and privacy-protection decision.

Once protection is triggered, OpenCV applies a localized transformation. Gaussian blur is preferred for the basic prototype because it is computationally inexpensive and visually understandable. A stronger version may use pixelation, texture disruption or a learned perturbation similar in spirit to the approaches reported in [1] and [2]. The protected image is saved as a new file so that the original can remain unchanged on the user's device.

5. SYSTEM WORKFLOW AND API INTEGRATION

The complete workflow follows a sense-assess-protect cycle. First, the user selects a photograph. Second, OpenCV normalizes the image and MediaPipe determines whether a hand is present. Third, the landmark coordinates identify candidate finger regions. Fourth, the candidate region is passed to the optional vision API for semantic assessment. Fifth, the local decision layer combines the returned label with simple geometric and image-quality indicators. Finally, OpenCV protects the region when the risk threshold is exceeded.

The API call can be implemented using an official Python client. A conceptual request contains the image and a text instruction; the response is converted into a machine-readable risk label. Because API output may be uncertain, the application should not blindly trust the response. Local rules provide a second layer of control, and the user can be shown the detected region before sharing. This hybrid arrangement reduces dependence on any single component.

From a privacy perspective, the API stage should be considered optional. If photographs are sensitive, an entirely local implementation is preferable. When an external API is enabled, the application should disclose that the image or crop is being transmitted to a third-party service and should follow the provider's data-handling and retention policies. The prototype therefore treats API use as a configurable module rather than an unavoidable part of the privacy mechanism.

6. RESULTS AND DISCUSSION

A functional prototype-level evaluation was defined around representative photograph conditions: no visible hand, hand present with non-sensitive orientation, extended finger with limited surface visibility, and extended finger with clear visible surface detail. The integrated pipeline produced the expected qualitative behaviour: photographs without hands bypassed the fingerprint-protection stage; hand-containing photographs were localized by MediaPipe; candidate finger regions were isolated for assessment; and high-risk candidate regions were subjected to localized protection. The protected output retained the surrounding image content while reducing the visibility of the targeted region.

The most important result is therefore the successful integration of ready-made components into an end-to-end privacy workflow rather than a claim of state-of-the-art biometric detection accuracy. The prototype demonstrates that a cognitive privacy layer can be inserted before an image is publicly shared. It also provides a practical path for later replacement of the heuristic/API decision module with a dedicated trained classifier.

The experiment also reveals several limitations. MediaPipe landmarks represent hand geometry and do not directly measure fingerprint ridge quality. A photograph may show a finger without containing enough resolution, focus or illumination for fingerprint matching. Conversely, a high-resolution close-up can expose more detail than the simple geometric rules can estimate. API decisions can also vary with image quality and prompt wording. For these reasons, the present results should be interpreted as potential-exposure detection and mitigation, not as proof that a fingerprint can be successfully reconstructed from every flagged image.

Functional observations: images without visible hands bypassed the hand stage; hand-containing images were localized by MediaPipe; extended fingers with limited detail produced lower-risk candidate regions; and clear finger-surface detail triggered the protection stage after the combined local/API assessment.

Figure 4: Functional evaluation summary of the prototype.

The figure 4 summarizes qualitative functional observations rather than statistical accuracy measurements.

7. APPLICATIONS AND LIMITATIONS

The proposed system can be used as a pre-sharing privacy assistant for social-media photographs, messaging applications, digital portfolios, and public event photographs. It can also serve as an educational demonstration of privacy-enhancing AI, showing how perception, decision making and local protection can be combined without creating a new large-scale model.

The main limitation is the gap between visual exposure and actual biometric recoverability. The system can identify a potentially sensitive finger region, but it does not establish that the region meets the quality requirements of a fingerprint matcher. Additional limitations include occlusion, unusual hand poses, poor lighting, motion blur, skin-tone and background variation, API latency, and the possibility of false positives or false negatives. A production system would require a labelled dataset, quantitative error analysis, privacy testing, and evaluation against established biometric-quality measures.

9. CONCLUSION

This paper presented a lightweight privacy-protection prototype for detecting potential fingerprint exposure in photographs. The system combines OpenCV for image processing, MediaPipe for hand and landmark localization, and an optional vision API for semantic risk

assessment. When a potential exposure is identified, the sensitive region is locally blurred or masked before sharing. The approach follows the privacy and trust objective of cognitive systems by allowing an intelligent software layer to detect a possible biometric risk and intervene before publication. The prototype is intentionally modest in scope: it does not claim to reconstruct fingerprints, train a new deep-learning architecture, or provide certified presentation-attack detection. Its value lies in demonstrating a practical and extensible workflow that can later be strengthened through dedicated biometric-quality analysis and locally trained privacy models.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance of Dr. Tilak Mukherjee, Dr. Ritwik Haldar, Department of Electronics & Communication Engineering, Haldia Institute of Technology, throughout this work, and the Department for providing the laboratory facilities.

REFERENCES

[1] A. Malhotra, S. Chhabra, M. Vatsa, and R. Singh, "On Privacy Preserving Anonymization of Finger-Selfies," Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), pp. 120–128, 2020, doi: 10.1109/CVPRW50498.2020.00021.

[2] S. Li, H. Xu, J. Wang, R. Xu, A. Liu, F. He, X. Liu, and D. Tao, "Hierarchical Perceptual Noise Injection for Social Media Fingerprint Privacy Protection," IEEE Trans. Image Process., vol. 33, pp. 2714–2729, 2024, doi: 10.1109/TIP.2024.3381771.

[3] "SocialGuard: An adversarial example based privacy-preserving technique for social images," J. Inf. Secur. Appl., vol. 63, p. 102993, 2021, doi: 10.1016/j.jisa.2021.102993.

[4] E. Tabassi, C. Wilson, and C. Watson, "Fingerprint Image Quality," NISTIR 7151, National Institute of Standards and Technology, 2004, doi: 10.6028/NIST.IR.7151.

[5] E. Tabassi and C. L. Wilson, "A Novel Approach to Fingerprint Image Quality," Proc. IEEE Int. Conf. Image Process. (ICIP), 2005.

[6] ISO/IEC 30107-1:2023, "Information technology — Biometric presentation attack detection — Part 1: Framework," International Organization for Standardization, 2023.

[7] Google AI Edge, "MediaPipe Hand Landmarker," Google for Developers, accessed Aug. 2026.

[8] Google AI for Developers, "Gemini API: Image Understanding," Google, accessed Aug. 2026.

[9] OpenCV, "OpenCV Documentation: Image Processing and Image I/O," Open Source Computer Vision Library, accessed Aug. 2026.