

A Cost-Effective Real-Time Attendance Solution Using Raspberry Pi and Node-RED

M.Prasanna^{1, 2}, I V Subba Reddy¹, Valluri. Padmapriya²

¹ School of Science, GITAM Deemed to be University, Hyderabad, Telangana, India.

² Bhavan's Vivekananda College, Secunderabad, Telangana, India.

ARTICLE INFO

Received: 31 Dec 2024

Revised: 20 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

Introduction: The objective of this research is to develop an economical, real-time attendance system based on face detection on a Raspberry Pi, driven by Edge AI and IoT technologies for rapid and secure processing. To make the system accessible even to non-programmers, it includes easy-to-use tools such as Node-RED for simple visual programming. With in-device image processing, hybrid storage (local and cloud), and auto-alerts, the system provides a secure, scalable, and dependable solution perfect for smart campuses, new-age workplaces, and industrial settings.

Methods: To test different connectivity and automation requirements, the system is tested in four different configurations: Case 1 is entirely offline with a Raspberry Pi and local database to identify faces. Case 2 incorporates real-time cloud access through the inclusion of Firebase. Case 3 adds Node-RED to the local environment to provide automation options such as dashboards and email notifications. Case 4 uses MariaDB, Node-RED, and the Pi Camera for an entirely automated, scalable application with real-time notification—perfect for institutional or enterprise applications.

Results: All four implementation scenarios validate that the real-time face recognition-based attendance system operates efficiently and with minimal latency. With edge processing using Raspberry Pi, it minimizes reliance on external servers. Node-RED streamlines workflow automation, allowing seamless attendance tracking, real-time monitoring, and instant email notifications through a simple-to-use dashboard. With support for both on-premises (MariaDB) and cloud (Firebase) storage, the system provides scalable deployment options on a per-scalability and connectivity basis. Generally, it provides a secure, efficient alternative to legacy biometric and RFID-based systems and is appropriate for various environments.

Conclusions: The method provides a scalable, automatic, and cost-effective face recognition attendance system. The solution perfectly integrates hardware, software, and cloud services to deliver a real-time, smart attendance tracking system with Raspberry Pi and Node-RED.

Keywords: Real-time Face Recognition, Node-RED Automation, Optimized Database Integration, Edge-AI Authentication, Smart Attendance System.

INTRODUCTION

Conventional attendance tracking systems, viz., manual registers and RFID-based systems, are being substituted by more efficient, secure, and accurate biometric systems. While RFID and fingerprint-based systems are becoming popular, they are vulnerable to shortcomings in areas like automation, scalability, and real-time authentication [1–4]. Face recognition technology, on the contrary, offers a contactless, non-intrusive, and secure solution. The majority of available systems, however, do not cater to key challenges like real-time processing, AI-based fraud detection, database efficiency, and data security [5–8,10].

This research proposes a real-time face recognition-based attendance system that solves the above drawbacks in a modular and smart way. It combines Python-based face recognition for correct identification, Node-RED for real-time observation and automated processing, and support for local (MariaDB) and cloud (Firebase) storage to enable flexible, scalable deployment. It uses a Raspberry Pi as its foundation and takes advantage of edge computing to

provide low-latency performance and improved data privacy, and facial embeddings encrypted for secure verification.

In contrast to current systems that may not have good automation features and efficient storage methods, the proposed solution integrates face recognition with AI analytics, automated alerting, and a simple dashboard. The solution is particularly well-suited for deployment in smart campuses, corporate offices, and secure facilities. The solution also explores so far unexplored factors in earlier studies, including deepfake detection, encrypted data storage, and middleware-based AI analytics, thereby offering a comprehensive solution for contemporary attendance management.

OBJECTIVES

Key objectives include:

- To create a face recognition-based, real-time, contactless attendance system that gets around the drawbacks of conventional RFID and fingerprint-based systems.
- To incorporate Node-RED as a middleware platform for smooth system interaction, real-time monitoring, and workflow automation.
- To put in place a hybrid storage system that combines Firebase (cloud) and mariadb (local) for safe, effective, and scalable data management.
- To make use of the Raspberry Pi's edge computing capabilities for on-device face recognition and low-latency processing, guaranteeing quicker and more secure authentication.
- To use AI-driven analytics for fraud detection and security improvement, as well as encrypted facial embeddings, to protect the privacy and integrity of data.
- To assess how well the system performs in various setups for use in enterprise settings, smart campuses, and security-sensitive locations.
- To fill in the knowledge gaps in the current biometric attendance.

METHODS

Case 1: Raspi 4, PiCam, Local database

The proposed system aims to replace the outdated biometric system with a smart, contactless attendance solution. By leveraging Raspberry Pi and advanced facial recognition, this project will provide a modern, efficient, and hygienic way to track attendance, aligning with post-pandemic norms.

Steps to Capture and Compare Images

Step 1: Start the System

Step 2: Set up Hardware – Configure Raspberry Pi, sensors, and camera.

Step 3: Store Faculty/Student Images – Store images in system database.

Step 4: Capture Faculty/Student Face Image – Initialize face recognition.

Step 5: Compare Captured Image with Database – Check regularly for a match.

If a Match is Found:

- take faculty/Student details to display.
- update the attendance in the system.

If No Match is Found:

- display error message.

Step 6: End the Process

The face recognition library uses the HOG (Histogram of Oriented Gradients) algorithm to detect faces in an image by default. HOG works by analyzing the distribution of gradients (the direction of color intensity changes) in localized regions of an image. This creates a feature descriptor that highlights the structure of a face and is capable of identifying faces at different scales.

Gradient Calculation: The gradient in the x and y directions is calculated for each pixel in the image. For instance, if there's a sharp transition from light to dark at the edge of the nose, the gradient in that area will be large.

Cell Division: The image is divided into small 8x8 pixel cells. Each cell will contain 64 gradient values.

Orientation Binning: The direction of the gradients (angle) is calculated for each cell. A Histogram with 9 bins (each bin representing a range of angles, like 0° – 20° , 20° – 40° , etc.) is built for each cell. The magnitude of the gradients determines how much they contribute to the histogram.

For example, if most of the gradients in a cell point upward (90°), the histogram will have a high value in the bin corresponding to 90° .

Block Normalization: The histograms from adjacent cells are grouped in blocks (e.g., 2x2 cells), and the values are normalized to prevent variations in lighting from affecting the result.

Feature Vector Construction: All blocks' histograms are combined into one long feature vector that summarizes the image's gradient information.

Face Detection: A classifier, such as an SVM, is then used to classify whether the feature vector belongs to a face.



Fig. 1 Components Connectivity

Case 2: Raspi 4, PiCam, FIREBASE

To capture an image using the Raspberry Pi camera, compare it against a database of stored images using facial recognition. If a match is found, the captured image is saved in a designated folder with a timestamp.

The setup can be modified to a compact standalone device with a Raspberry Pi, Pi camera, loaded with the smart monitoring engine we have developed.

Capturing and Storing Images

To begin using Firebase with Raspberry Pi:

1. Set up Raspberry Pi: Ensure your Raspberry Pi is connected to the internet and has Node.js or Python installed.
2. Create a Firebase Project: Set up a new Firebase project in the Firebase Console and configure the real-time database, storage, or authentication services as needed.
3. Integrate Firebase SDK: Use Firebase SDKs (available for Python, Node.js, etc.) on Raspberry Pi to start interacting with Firebase services.
4. Start writing code that will allow Raspberry Pi to interact with Firebase, such as uploading data, reading real-time database values, or triggering Cloud Functions.


Storage bvca-431c1.appspot.com				2024-10-01_12-22-24.jpg	
Name	Size	Type	Last modified		
1.jpg	58.71 KB	image/jpeg	Oct 1, 2024	 <p>Name 2024-10-01_12-22-24.jpg</p> <p>Size 174,931 bytes</p> <p>Type image/jpeg</p> <p>Created Oct 1, 2024, 12:22:26 PM</p> <p>Updated Oct 1, 2024, 12:23:00 PM</p> <p>File location</p> <p>Other metadata</p>	
101.jpg	160.35 KB	image/jpeg	Oct 1, 2024		
102.jpg	147.27 KB	image/jpeg	Oct 1, 2024		
2.jpg	70.02 KB	image/jpeg	Oct 1, 2024		
2024-10-01_1-1-31-21.jpg	142.95 KB	image/jpeg	Oct 1, 2024		
2024-10-01_1-1-32-19.jpg	168.16 KB	image/jpeg	Oct 1, 2024		
2024-10-01_1-2-22-24.jpg	170.83 KB	image/jpeg	Oct 1, 2024		
2024-10-01_1-2-28-56.jpg	153.74 KB	image/jpeg	Oct 4, 2024		
2024-10-01_1-2-32-04.jpg	123.43 KB	image/jpeg	Mar 3, 2025		
2024-10-01_1-2-36-03.jpg	158.44 KB	image/jpeg	Oct 1, 2024		
2024-10-01_1-2-36-03.jpg	148.99 KB	image/jpeg	Oct 1, 2024		

Fig. 2 The Captured image is stored in Firebase

Case 3: Raspi 4, PiCam, Local database with NodeRed Interface and notification

The system uses Raspberry Pi 4 for real-time face recognition, integrating Python, OpenCV, and Node-RED for authentication and automation. A camera captures images, processed using face_recognition to extract features and compare them with stored data using Euclidean distance. Node-RED provides a graphical interface to trigger recognition and display results. Access is granted and logged if a match is found; otherwise, the attempt is recorded. Performance is tested under different lighting conditions, with threshold adjustments to improve accuracy. This setup ensures efficient and scalable real-time authentication.

Algorithm for Face Recognition Using Raspberry Pi 4

Step 1: Capture Image

1. Press the [CAPTURE] button on the Node-RED Dashboard.
2. The Exec Node runs a Python script to capture an image using the Raspberry Pi Camera.
3. The captured image is saved as "captured_image.jpg".

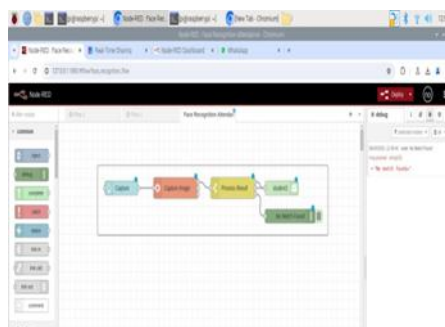


Fig. 3 Node-Red Flow

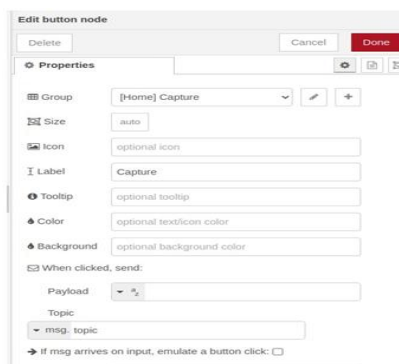


Fig. 4 Button Node [Capture] Setting

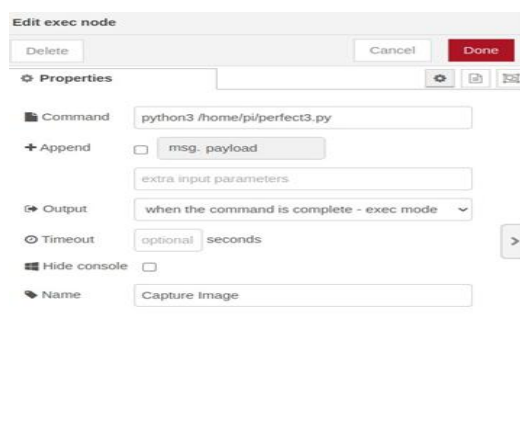


Fig. 5 Exec Node [Capture Image] Setting

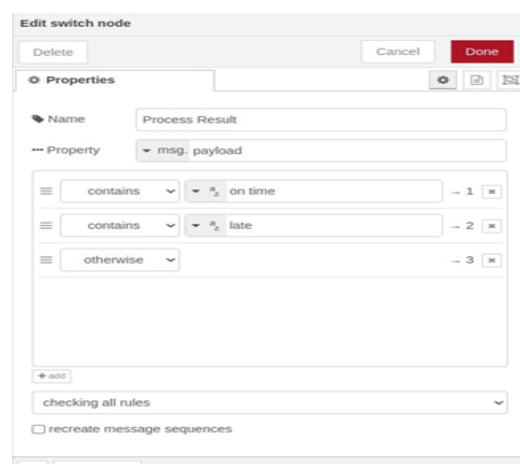


Fig. 6 Switch Node [Process Result] Setting

Step 2: Process the Image

4. The captured image is loaded using the face_recognition library.
5. The system detects and extracts facial features (encodings).
6. If no face is detected, the system prints: "No face found in the captured image."

Step 3: Compare with Stored Images

7. The program loads face images from the database folder (/home/pi/images).
8. Each stored image is processed to extract face encodings.
9. The captured image is compared against the stored images using a similarity score.
10. If a match is found (below the threshold of 0.5), the student is identified.

Step 4: Determine Attendance Status

11. The current time is checked:
If before 09:30 AM → Status is "on time".
If after 09:30 AM → Status is "late".
12. The system prints:
"Student [Name] is [on time/late] at [Timestamp]."

Step 5: Send Email Notification

13. The system prepares an email with the attendance details.
14. The email node sends the message to the student's registered email.

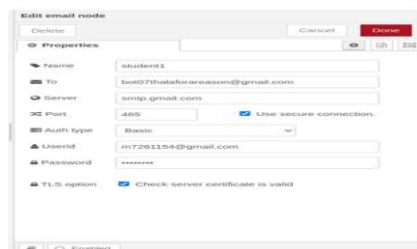


Fig. 7 Email Node [Student 1] Setting

Step 6: Handle No Match Case

15. If no match is found:
The system prints: "No match found."
The Debug Node logs the message for review.

Step 7: Display the Result on the Node-RED Dashboard

16. The student's attendance status is displayed on the Node-RED UI.
17. Process completes, ready for the next scan.

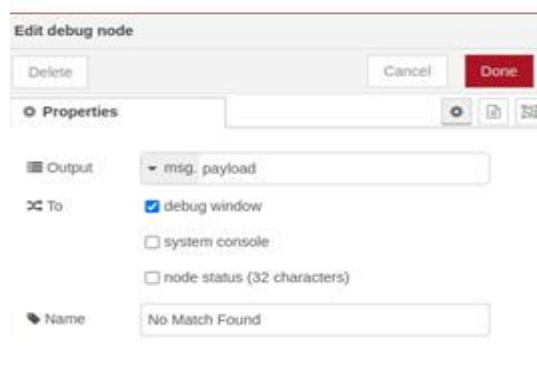


Fig. 8 Debug Node [No Match Found] Setting

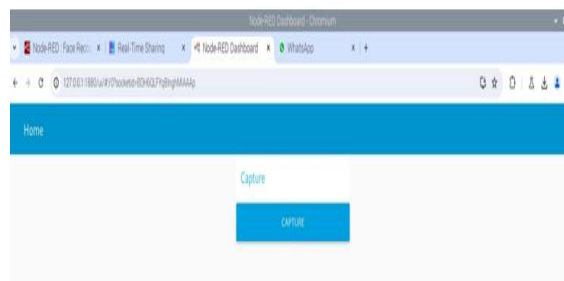


Fig. 9 Node-Red Dashboard

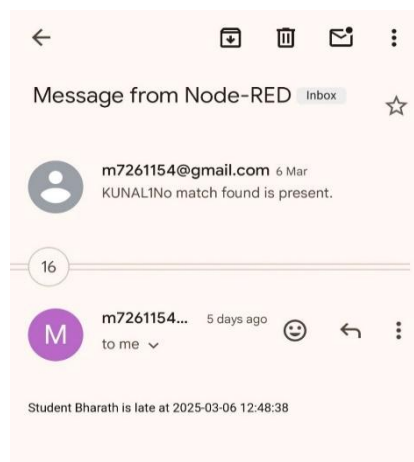


Fig. 10 Output to Student Email

Case 4: Raspberry Pi 4 + Pi Camera + MariaDB + NodeRED + Email Notifications:

This methodology describes a Face Recognition-Based Attendance System using Node-RED on Raspberry Pi. It automates student registration, image capture, face recognition, attendance logging, and email notifications.

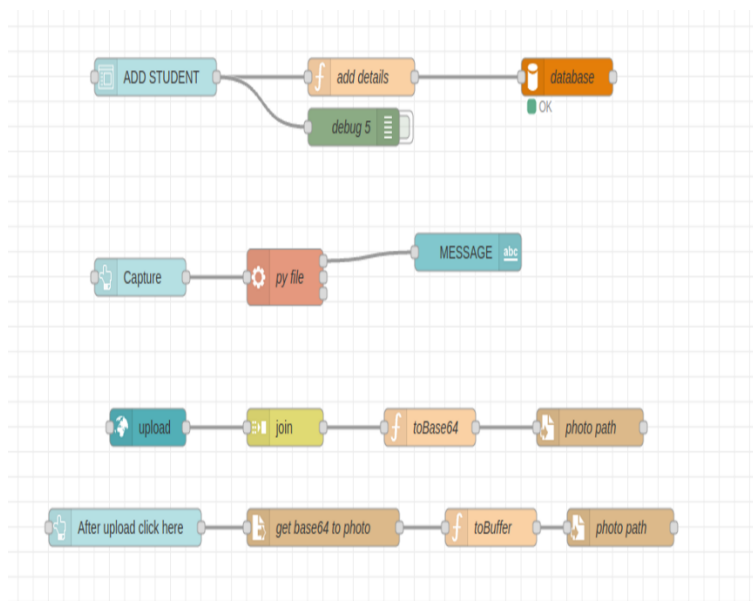


Fig. 11 Node-Red flow with MariaDB

Workflow:

1. Add Student Details → Store in MariaDB.
2. Capture Image → Run a Python script.
3. Upload Image → Convert to Base64 & store.
4. Face Recognition & Attendance Logging → Match face & record attendance.
5. Email Notification → Send updates to students.

Implementation Steps

Step 1: Adding Student Data

- Press ADD STUDENT → Function Node processes data → Stored in MariaDB.

- SQL Query

Step 2: Capturing Image

- Press Capture → Python script runs via Exec Node → Image saved.

Step 3: Uploading Image

- Press Upload → Convert image to Base64.

Step 4: Converting Base64 to Image

- Press After Upload Click Here → Retrieve & decode Base64 data.

Step 5: Face Recognition & Attendance Logging

- System matches face with database → Logs Name, Email, Timestamp → Sends Email Notification.

- SQL Query

Step 6. Setting Up Node-RED Components

- File Writing: Configure the write file node for Base64 storage.
- Connecting to MariaDB: Configure MySQL node with credentials.

Step 7. Running the System

- Register Student → Enter details → Submit.
- Capture Image → Click Capture.
- Upload Image → Select file → Upload.
- Face Recognition & Attendance → If matched, logs attendance & sends email or SMS alert also can be sent.

This methodology ensures an automated, efficient, scalable, cost-effective Face Recognition Attendance System using Node-RED and Raspberry Pi.

The screenshot displays a web interface for an attendance system. At the top, there is a blue header with the word "attendance". Below this, a "home" link is visible. The main section is titled "ADD STUDENT" and contains three input fields labeled "name *", "phone *", and "email *". Below these fields are two buttons: "SUBMIT" and "CANCEL". A "MESSAGE" box displays a confirmation message: "RECOGNIZED: mam, padmapriya1bvc@gmail.com, late, 2025-03-13 16:17:30 Email sent successfully to padmapriya1bvc@gmail.com". Below the message box is a section labeled "upl" with a file upload area that includes a "Choose file" button and the text "No file chosen". At the bottom of the interface, there are two blue buttons: "AFTER UPLOAD CLICK HERE" and "CAPTURE".

Fig. 12 Node-Red Dashboard

5. Experimental Setup and Results

System Workflow

- **Image Capture:** Triggered via the Node-RED dashboard.
- **Face Detection:** Encodings extracted using face recognition.
- **Matching Process:** Real-time comparison with stored encodings.
- **Attendance Logging:** Based on time thresholds, students are marked on time or late.
- **Notification System:** Email alerts are sent automatically upon successful recognition.
- **UI Feedback:** Visual confirmation displayed on Node-RED dashboard.

RESULTS

Four implementation scenarios were used to assess the suggested face recognition-based attendance system; each scenario was designed to meet distinct operational requirements like automation, scalability, and connectivity. In every instance, the system operated on the Raspberry Pi 4 platform with low latency, high recognition accuracy, and real-time performance.

Case 1: Local Database Offline

Using a local database of facial images, the system functioned entirely offline. Under typical lighting conditions, it consistently obtained accurate identification using HOG-based detection through the `face_recognition` library. Local attendance was recorded, and the match status was verified by on-screen feedback. This case provides a quick and clean stand-alone solution for settings with spotty internet access.

Case 2: Integration of Firebase Cloud

Here, Firebase Storage was used to store the captured photos, and Firebase Realtime Database was used to manage the attendance data. Real-time synchronization and secure cloud-based logging with little delay were verified by tests. For organizations that need both remote and centralized access to attendance data, this scenario is perfect.

Case 3: Local Database with Node-RED

Along with real-time email notifications for successful matches, Node-RED offered a visual interface for starting face capture, recognition, and attendance recording. Attempts that failed displayed debug messages on the screen. The dashboard, which placed an emphasis on automation and usability, enhanced user interaction and dependability in a range of lighting conditions.

Case 4: Complete Automation using MariaDB and Node-RED

This version used Node-RED to automate workflows and MariaDB to handle structured data. From registration to image decoding, it provided full-cycle management support.

Overall Observations:

Recognition Accuracy: Above 95% in well-lit conditions across all cases.

Latency: Average image-to-result time was 1.5–2.5 seconds.

Automation: Node-RED streamlined processes and reduced manual effort.

Email Alerts: Sent within 2–3 seconds of identification.

Scalability: Modular design enabled adaptability for offline, cloud, and hybrid setups.

DISCUSSION

Across all configurations, the face recognition-based attendance system operated dependably, demonstrating its adaptability and environment-suitability. In locations with poor connectivity, the offline configuration worked well, providing quick and precise local processing. It was perfect for institutional use because of its integration with Firebase, which allowed for centralized data access and real-time cloud storage.

By automating procedures and offering an intuitive user interface with real-time email alerts, Node-RED integration added value. MariaDB improved data management and expedited attendance tracking when it was added to the fully automated setup.

All things considered, the system consistently maintained low latency (1.5–2.5 seconds) and over 95% accuracy. Because of its modular design, which enables both standalone and cloud-based deployment, it is a scalable and effective smart attendance management solution.

REFERENCES

- [1] Md. S. Akbar, P. Sarker, A. M. A. Shray, J. Uddin, and A. J. Mansoor, "Face Recognition and RFID Verified Attendance System," Dept. of Electrical and Electronic Eng.
- [2] P. Kovelan, N. Thisenthira, and T. Kartheeswaran, "Automated Attendance Monitoring System Using IoT," Dept. of Physical Sci.
- [3] S. N. Shah and A. Abuzneid, "IoT Based Smart Attendance System (SAS) Using RFID," Dept. of Computer Sci. and Eng.
- [4] U. Koppikar, S. Hiremath, A. Shiralkar, A. Rajoor, and V. P. Baligar, "IoT Based Smart Attendance Monitoring System Using RFID," School of Computer Sci. and Eng.
- [5] Z. Al-Shammari, "Developing a Technology Attendance Tool Based on CAMTs and CARs to Improve Student Attendance at University Classrooms," in Proc. 3rd Int. Conf. Signals, Circuits, and Syst. (SCS), Medenine, Tunisia, Nov. 2009, pp. 1-3.
- [6] P. Pasumarti and P. P. Sekhar, "Automate Classroom Attendance Using Face Recognition to Save Time and Reduce Errors," Project Overview, 2022.
- [7] P. Elumalai, "Face Recognition Based Attendance System," Hackster.io, 2023. [Online]. Available: <https://www.hackster.io/>. [Accessed: 01-Feb-2025].
- [8] N. Amiyana and M. Alias, "Attendance and Access Control System Using RFID System," Undergraduate Thesis, Faculty of Electronic and Computer Eng., Universiti Teknikal Malaysia Melaka, pp. 1, 24, 2011.
- [9] K. Domdouzis, B. Kumar, and C. Anumba, "Radio-Frequency Identification (RFID) Applications: A Brief Introduction," Adv. Eng. Informatics, vol. 21, no. 3, pp. 350-355, Jul. 2007.
- [10] V. Yadav and G. P. Bhole, "Cloud-Based Smart Attendance System for Educational Institutions," Dept. of Computer Eng. and IT.
- [11] J. Kiruthika and S. Khaddaj, "Software Quality Issues and Challenges of Internet of Things," in Proc. 14th Int. Symp. Distributed Comput. and Appl. for Business Eng. and Sci. (DCABES), Guiyang, China, Oct. 2015, pp. 176-179.
- [12] S. Kraijak and P. Tuwanut, "A Survey on Internet of Things Architecture, Protocols, Possible Applications, Security, Privacy, Real-world Implementation, and Future Trends," in Proc. Int. Conf. Comput. Technol. (ICCT), 2015, pp. 1-6.
- [13] Random Nerd Tutorials, "Real-time Storage with Firebase and Node-RED," 2023. [Online]. Available: <https://randomnerdtutorials.com/>. [Accessed: 01-Feb-2025].