Facial Recognition Attendance Tracking: An Intelligent Monitoring Approach

Syed Tauseeq Hussain, Mario Antonio Nogueira Mansur Carvalho, Vikrant Kishor Rathod, Srinivasa Mani Sankar Reddy Karri, Xinrui Yu, Mikhail Gromov, and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*—This paper presents an Intelligent Monitoring System that utilizes the Internet of Things (IoT) and Artificial Intelligence (AI) technologies to automate the class attendance process reliably and efficiently. Conventional approaches for attendance tracking have been laborious and have consumed a significant amount of time, with errors and inconsistencies being a common occurrence. In contrast, the Intelligent Monitoring System combines object detection & recognition AI models, wireless communication, and cloud monitoring to generate reliable attendance data that can be used for various purposes, such as tracking student-by-student attendance data and monitoring overall attendance statistics. The system comprises an ID reader that uses radio frequency tags, a facial recognition system that uses a camera and AI algorithms, and a cloud monitoring system for attendance statistics. The proposed system is designed to overcome the challenges of traditional attendance-taking processes and provide a solution that is accurate, reliable, and efficient.**

Keywords—Internet of Things, deep learning, attendance monitoring, facial recognition

I. Introduction

Class attendance management is a fundamental aspect of educational institutions, playing a pivotal role in assessing student engagement and facilitating effective teaching practices. Traditional methods of attendance-taking have often been labor-intensive and prone to errors, necessitating the exploration of innovative technological solutions. The advent of the Internet of Things (IoT) and Artificial Intelligence (AI) technologies has revolutionized various industries, offering opportunities for automation and optimization. In response to this trend, the Intelligent Monitoring System has been developed to streamline the attendance tracking process in academic settings. This system leverages IoT and AI capabilities to generate comprehensive student attendance data, enabling real-time monitoring and analysis. By replacing manual methods such as verbal roll calls or paper-based attendance sheets, the Intelligent Monitoring System enhances efficiency and accuracy in attendance management. This research aims to explore the functionality and effectiveness of the Intelligent Monitoring System in higher education environments. Specifically, it investigates the integration of RFID (Radio Frequency Identification) technology for ID card authentication and facial recognition algorithms to verify student presence. Additionally, the utilization of cloud-based platforms, such as Amazon Web Services (AWS), for data storage and analysis is examined. Through the implementation of this system, universities can potentially optimize classroom allocation, improve faculty decision-making, and gain insights into student attendance patterns. This paper provides an in-depth examination of the system architecture, comprising three interconnected subsystems, and discusses its implications for educational institutions.

The Intelligent Monitoring system logic is summarized in Fig. 1, and the attendance tracking process is shown in Fig. 2.

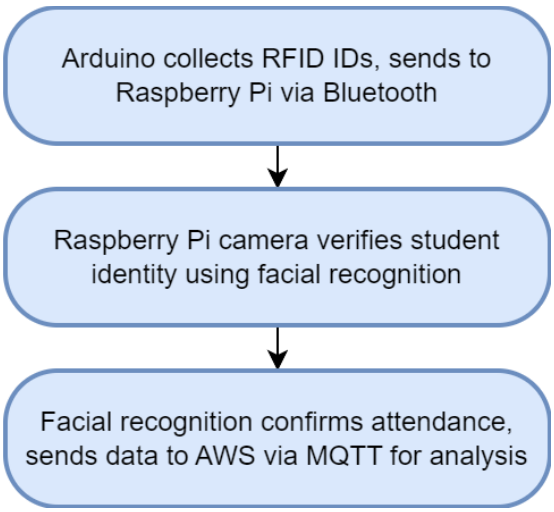

Fig. 1. Overview Architecture of Intelligent Monitoring System

II. System Design

A. Overall

This sub-system is composed of an Arduino connected to an RFID reader/writer and a Bluetooth module. These components are connected between themselves (See Fig. 3). Students tap their cards on the RFID reader, which captures the unique identifier on these cards and immediately transmits them through Bluetooth Serial Communication. The card ID is then captured by the Raspberry Pi. The hardware used by the RFID system is the following:

- Arduino: It is an open-source hardware platform that has many basic components like a processor, GPIO Pins, and Clock and supports various communication protocols such as Serial Peripheral Interface (SPI), Inter-Integrated-Circuit bus (I2C), Universal Asynchronous Receiver Transmitter (UART), and Software Serial. The

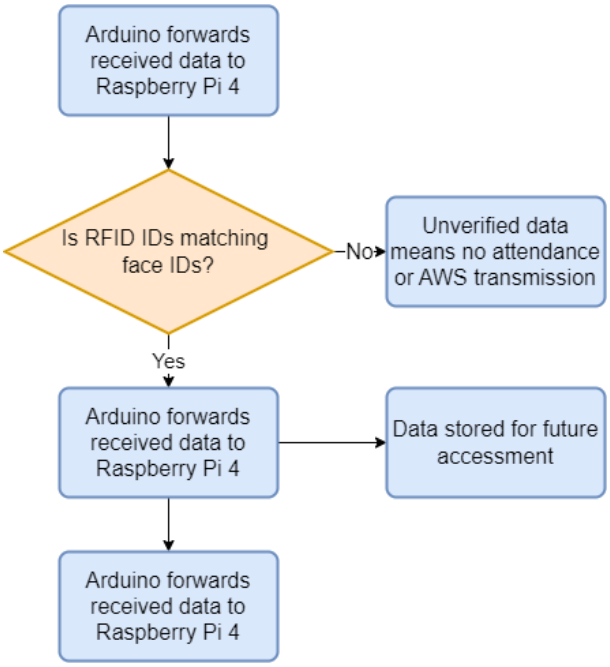

Fig. 2. Workflow Overview of the Smart Multi-Building Energy Monitoring System

Arduino platform has its software and supports various platforms like Windows, Mac OS, and Linux. In this paper, Arduino is used to transmit the data from the MFRC522 RFID reader to the Raspberry Pi using the HC-06 Bluetooth module.

- RFID Tag: RFID stands for Radio Frequency Identification. RFID tags are usually in the form of a tag or a card. They use Radio Frequency Technology and can store small amounts of information [1]. The information can be a short description, serial numbers, or even a few pages of data. The information is stored without using any power and can be modified. In this paper, the RFID tags are used to store the student's college ID numbers.
- MFRC522 RFID reader: The MFRC522 RFID reader is an inexpensive contactless communication IC that uses Radio Frequency technology to read RFID tags. It works at 13.56 MHz and can be used with Arduino or other microcontrollers to read/write data from RFID tags. In this paper, it is mainly used to read and write the Student's ID to the RFID Tags, and it is interfaced with the Arduino using the SPI Protocol.
- HC-06 Bluetooth module: HC-06 is a wireless transceiver that works on Bluetooth technology 2.0. While Bluetooth technology doesn't need a central hub or router to work, the HC-06 module can't work independently as it can only work as a slave device. Therefore, another Bluetooth module or a device like a Raspberry Pi needs to act as a Master device to begin the communication. In this paper, HC-06 is connected to an Arduino and it transmits the Student ID number that the Arduino receives from the MFRC522 RFID reader to a Raspberry Pi, which acts as a Master device [2].

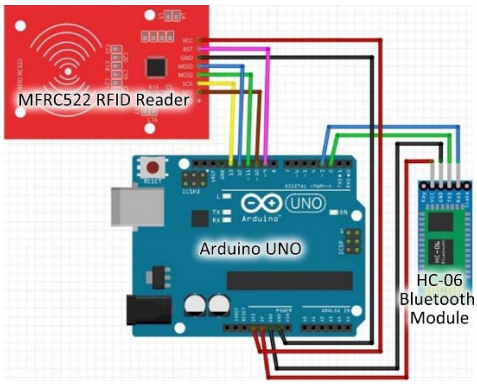

Fig. 2. RFID & Bluetooth connected to Arduino UNO

B. *Artificial Intelligence System*

Facial recognition [3] makes the system smarter and more secure. It delivers a two-way authentication to the attendance monitoring system by verifying the recognized student from the student ID received from the RFID system. If the ID stored in the database for the recognized individual matches the ID of the person received from the RFID system; it's verified, the attendance of that individual is generated and the data is sent to the AWS cloud. However, if the IDs don't match then verification fails, and no data is sent to the AWS cloud hence that individual is considered absent.

There are three parts to the facial recognition system:

1) Face Detection: The system must initially detect a face in the frame. For that Haar Cascade classifier [4] is used. Haar Cascades have a higher accuracy for a wide range of objects, moreover, they are efficient, process images in real-time, and are robust against illumination. Although there are a few other alternatives for deep learning algorithms such as Faster RCNN, SSD, YOLO, Mask R-CNN, etc. However considering the Raspberry PI computational capabilities, memory constraints, and the fact that detection and recognition would work simultaneously, Haar cascade is preferred for the paper. For training 40 images are considered that contain the faces of every individual. After the database is created in the system, images are convolved with a series of filters that are responsible for extracting different features i.e. edges, lines, etc. Then this algorithm classifies whether this image patch contains a face or not. Using the same concept, a sliding window technique is used to apply the algorithm to a complete image. Once the face is detected, it's considered a region of interest (ROI) and finally, it puts the ROI into a list of training data and its corresponding label which is the name of the individual. There are a few points that need consideration when creating a training database for face detection and recognition. This includes:

a) High resolution: Images should be high in resolution with good lighting on them to make sure all the features of the face are properly displayed.

b) Different angles: Images should be high in resolution with good lighting on them to make sure all the features of the face are properly displayed. different angles should be explored

for the face to make sure that well-rounded features are recognized. However, in our case, since we are using the frontal face detection part of the Haar cascade algorithm, we just needed a small amount of variation in angles, not 360 degrees.

2) Face Recognition [5]: Once the faces are detected, facial recognition does its part by using an LBPH (Local Binary Patterns Histograms) [6] face recognizer object which is an algorithm used for facial recognition. It trains using the NumPy Arrays and their corresponding labels.[7] After the data is trained, it is stored in a .yml file. An yml file is a human-readable data serialization language. It is used in machine / deep learning models and Artificial intelligence models to store metadata about the model such as its hyperparameters, architecture, and training configuration. This metadata can be used to reproduce and deploy the model consistently. In the testing part, the trained model loads the label mappings, and it uses OpenCV [8] to capture frames from the camera. For each frame, it detects the face first and then applies the trained face recognizer to predict the identity of the face. For the facial recognition part, there are a few sets of lines in the code that are important to discuss:

a) cv2.face.LBPHFaceRecognizer_create(): It is a function in the OpenCV that creates an instance of a face recognizer object using a Local Binary Patterns Histograms (LBPH) algorithm. This algorithm extracts features from the image and then uses a machine learning algorithm to classify these features as belonging to respective individuals.

b) recognizer.predict(roi_gray): The predicted function will take the roi_gray which is the grayscale ROI [9], the face, as input and returns two values: id and the confidence value of this prediction.

3) Verification: Finally, the Card ID is received by the Raspberry PI and the script on the Raspberry PI matches it with the name of the face detected by matching it with the name from the CSV file containing a list of students' IDs and names. Once the student is verified, their attendance is generated. If the ID doesn't match with the name predicted by the facial recognition system, then verification for that student fails and is considered absent. An example of a verification successful message is shown in Fig. 4.

Fig. 3. Verification on the terminal was successful. This pertains to the backend of the project and won't be visible in practical implementation.

C. AWS Cloud Monitoring Implementation

Cloud monitoring capabilities to make attendance statistics accessible remotely were built using AWS, inside which three main services were utilized to allow for an online user interface: AWS IoT Core, AWS IoT Analytics, and AWS QuickSight [10].

On AWS IoT Core, the system's Raspberry Pi was configured as a "Thing resource", which allowed it to receive the keys and certificates necessary to communicate with AWS via MQTT protocol. Under this protocol, the Raspberry Pi was the publisher of an "attendance_data" topic and AWS was the subscriber.

Using the AWS IoT SDK libraries for Python, the Raspberry Pi published JSON messages that contained the name, ID, and card tapping time (See Fig. 5.) of the students whose attendance was verified by the facial recognition system, and AWS received those messages through MQTT protocol and was able to extract data through AWS IoT Analytics.


```
"name": "Vikrant Rathod",
"id": "73BA612",
"taptime": "2023-03-29 20:52:42.536767"
```


Fig. 4. JSON message format utilized to transmit attendance data via MQTT protocol.

Using the AWS IoT SDK libraries for Python, the Raspberry Pi published JSON messages that contained the name, ID, and card tapping time (See Fig. 5) of the students whose attendance was verified by the facial recognition system, and AWS received those messages through MQTT protocol and was able to extract data through AWS IoT Analytics.

Inside Analytics, a Channel was configured to absorb the messages under the "attendance_data" MQTT topic. Next, a pipeline was created to extract relevant attributes from the JSON messages, which were student name, ID, and tap time, and forward these attributes and their values to a data store that was created specifically for this system. For this pipeline, AWS roles and policies had to be defined to give the necessary reading and writing access so that the data could be transmitted. Finally, an Analytics dataset was created and configured to ingest new data from the data store every one minute.

At this point, the attendance data is already present in AWS, and is organized in SQL format in the IoT Analytics dataset, so the last step to make it accessible through an online human-machine interface is to create a friendly dashboard. This dashboard is possible through AWS QuickSight, a service that allows for analysis and graphs to be generated based on datasets from other AWS services, which includes IoT Analytics datasets.

AWS QuickSight can ingest data from other AWS datasets according to a defined schedule. The most frequent refresh schedule allowed by AWS is once every 24 hours, and this was the choice for our system. That means professors and other University members who have access to attendance data can see an updated dashboard once every day.[11]

On AWS QuickSight, an analysis was created based on the attributes and their values. The analysis contained an attendance table and an attendance totals graph. The attendance table had student names on the rows and one lecture per column, which allowed viewers to see what specific lectures each student attended. In the total attendance graph, the lectures were displayed along the x-axis of a bar chart, in which the bar height represented the total number of students that were present in each class. AWS QuickSight offers filter functions for the data

attributes and values that get displayed, permits the layout of the analysis (colors, shapes, and background) to be altered, and allows the developer to define the size and position of each graph and table on the screen. QuickSight also allows for the creation of calculated fields, which were important for the work done for the Intelligent Monitoring System. Specifically, the lecture number was one of the values displayed on the analysis graphs, however, it was not an attribute obtained by reading the JSON messages received and processed in AWS IoT Analytics. To obtain the lecture number, a calculated field was created on QuickSight that extracted the date and time of the lecture from the attribute "taptime", which was just a timestamp that represented when the student tapped their ID cards at the RFID reader. This timestamp was a String, and the date and time extraction was based on character position.

Once the AWS QuickSight analysis was done, it was published to a dashboard, which could be accessed through a webpage by anyone who received permission to view the data. The main view of the attendance statistics that Professors and other University members can see online (see Fig. 7).

III. Results

A. RFID System

The detected IDs are shown in Fig. 6.

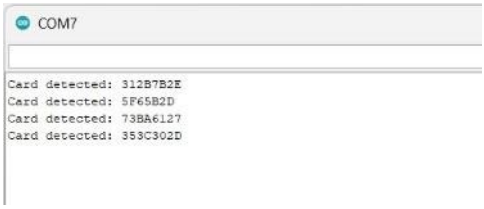

Fig. 5. Display of IDs detected at the Arduino UNO terminal after the RFID tag is detected via RFID reader.

B. Artificial Intelligence System

The facial recognition system indicates that the individual has been recognized. The model outputs a bounding box around the person's face along with their name. Successful verification means that the student was detected via RFID, verified through facial recognition, and their attendance was marked and sent to the AWS cloud.

C. AWS Cloud Monitoring Implementation

The facial recognition system indicates that the individual has been recognized. The model outputs a bounding box around the person's face along with their name. Successful verification means that the student was detected via RFID, verified through facial recognition, and their attendance was marked and sent to the AWS cloud.

The table on the upper part of the dashboard has student names on the rows and lectures on columns and shows if each student was present in each lecture. This table had totals on the right-side end that showed the total number of lectures attended by each student, which could be used by Professors to define attendance-related grades. During the experiments, the lecture numbers were the minute component of the time stamp obtained when the students tapped their cards. In the future real-life application of the Intelligent Monitoring System, this lecture number should be the date of the time stamp, since most classes only happen once per day. The minutes were used instead of dates because the team wanted to see updated results in different columns quickly for testing purposes, so it was not feasible to wait for another day to see if the dashboard was working as desired.

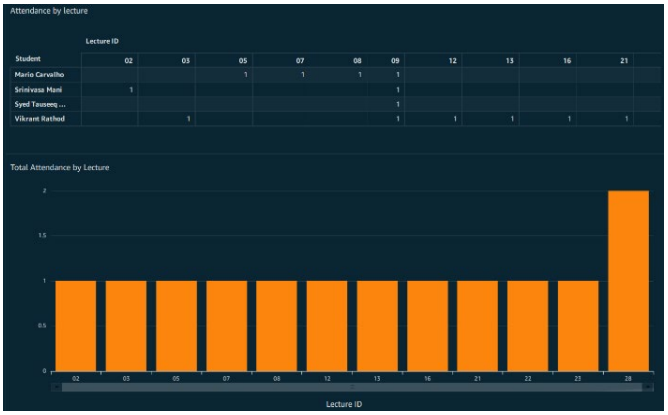

Fig. 7. Attendance data per student and lecture for a sample class of 4 registered students.

The bottom chart of the dashboard is a column chart showing the total number of students present at each lecture. This chart can have many uses, such as showing attendance trends for some specific class that can indicate how engaged students are during the semester. This can be valuable feedback for Professors. Another potential use for this chart is for the educational institutes to re-evaluate what is the appropriate classroom to be assigned for each class, considering the capacity and attendance numbers. Moreover, in case the attendance numbers are extrapolating the room capacity, entities such as campus Public Safety can be alerted and actions can be taken to avoid bigger issues in case of a fire, for example.

IV. Conclusion

The idea of setting up a reliable attendance monitoring system was established successfully, and a centralized system can be set up by storing all the data in a cloud and displaying the attendance on the AWS cloud. We were able to verify two students and their attendance was marked and recorded. We did not face any noticeable delays in communications.

To make this system ready for large-scale deployment, the biggest points to work on are to make the system more robust against wireless connection issues, to make the facial recognition more reliable, and to adapt the dashboard following feedback from Professors and Universities.

Our paper, originally designed for college and university settings, demonstrates versatile applicability across various other domains with minimal adaptation. For instance, the system's robust authentication framework can seamlessly extend to high-traffic environments such as airports or banks, where stringent security measures are imperative. By integrating two-step authentication mechanisms utilizing government-issued IDs like passports or other identification cards, coupled with advanced facial recognition technology, organizations can enhance security protocols while optimizing operational

efficiency. This flexibility underscores the scalability and adaptability of our solution, positioning it as a viable option for diverse applications beyond the educational sector.

REFERENCES

[1] L. E. Staff, “In-depth: What is RFID? how it works? interface RC522 with Arduino,” Last Minute Engineers, https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/ (accessed Feb. 11, 2024).

[2] “HC-06 Bluetooth module,” Components101, https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet (accessed Feb. 11, 2024).

[3] D. A. R. Wati and D. Abadianto, "Design of face detection and recognition system for smart home security application," 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, 2017, pp. 342-347.

[4] Mittal, “Haar Cascades, explained,” Medium, https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d (accessed Feb. 11, 2024).

[5] A. U. Naik and N. Guinde, "LBPH Algorithm for Frontal and Side Profile Face Recognition on GPU," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2020, pp. 776-779.

[6] K. S. do Prado, “Face recognition: Understanding LBPH algorithm,” Medium, https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b (accessed Feb. 11, 2024).

[7] M. G. Sarwar, A. Dey and A. Das, "Developing a LBPH-based Face Recognition System for Visually Impaired People," 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 2021, pp. 286-289.

[8] A. Kumari Sirivarshitha, K. Sravani, K. S. Priya and V. Bhavani, "An approach for Face Detection and Face Recognition using OpenCV and Face Recognition Libraries in Python," 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2023, pp. 1274-1278.

[9] Noel, “Region of interest in Computer Vision,” Scaler Topics, https://www.scaler.com/topics/region-of-interest-opencv/ (accessed Feb. 11, 2024).

[10] Visualizing data in Amazon QuickSight - Amazon quicksight, https://docs.aws.amazon.com/quicksight/latest/user/working-with-visuals.html (accessed Feb. 11, 2024).

[11] P. Pattnaik and K. K. Mohanty, "AI-Based Techniques for Real-Time Face Recognition-based Attendance System- A comparative Study," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1034-1039.