RAMS: Room Attendance Monitoring System

Syed Tauseeq Hussain
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago,USA
shussain10@hawk.iit.edu

Mario Antonio Nogueira Mansur Carvalho
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago,USA
mnogueiramansurcarva@hawk.iit.edu

Vikrant Kishor Rathod
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago,USA
vrathod@hawk.iit.edu

Srinivasa Mani Sankar Reddy Karri
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago,USA
skarri5@hawk.iit.edu

***Abstract*— This paper presents a Room Attendance Monitoring System (R.A.M.S) that utilizes Internet of Things (IoT) and Artificial Intelligence (AI) technologies to automate the class attendance process in a reliable and efficient manner. 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, R.A.M.S combines object detection 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, Artificial Intelligence, Attendance, Bluetooth, AWS, MQTT, JSON, RFID, Arduino, Raspberry Pi

I. INTRODUCTION

Class attendance is a crucial aspect of the educational system and holds significant importance for teachers and professors alike. The traditional process of attendance-taking has been time-consuming involving methods such as calling out names or passing around attendance sheets. With the rapid development of the Internet of Things and Artificial Intelligence technologies, many repetitive processes have been automated in various industries. The R.A.M.S system generates student-by-student attendance data, which can be used to monitor overall attendance statistics. This can help Universities to allocate classrooms to specific classes more efficiently, and faculty can use the data to better understand student attendance patterns. To ensure the reliability of attendance data, R.A.M.S will use facial recognition to ensure students are actually present in class.

The R.A.M.S system consists of three sub-systems, as shown in **Figure 1**. The ID reader sub-system allows students to tap their campus ID cards containing a radio frequency tag to a radio frequency reader. The reader is connected to an Arduino which wirelessly sends attendance data to the cloud monitoring sub-system through HC-06 Bluetooth Module via Raspberry Pi. The facial recognition system sub-system utilizes a camera connected to a Raspberry Pi and an AI algorithm that recognizes the people present in the room. The cloud monitoring sub-system uploads all attendance data generated by the first two sub-systems to a cloud platform. The cloud platform, in this case, is Amazon Web Services (AWS),which has resources to generate attendance statistics, that can be used by faculty and university departments.

The R.A.M.S' system logic is summarized in **Figure 2**, which shows how attendance data gets verified and stored in the cloud.

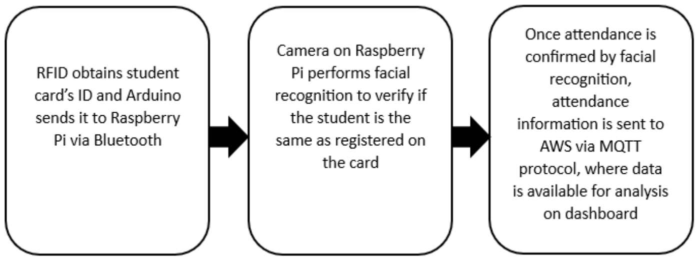

RFID obtains student card's ID and Arduino sends it to Raspberry Pi via Bluetooth
Camera on Raspberry Pi performs facial recognition to verify if the student is the same as registered on the card
Once attendance is confirmed by facial recognition, attendance information is sent to AWS via MQTT protocol, where data is available for analysis on dashboard

Figure 1 – Overview of RAMS Architecture or three main parts of RAMS interact.

II. System Design

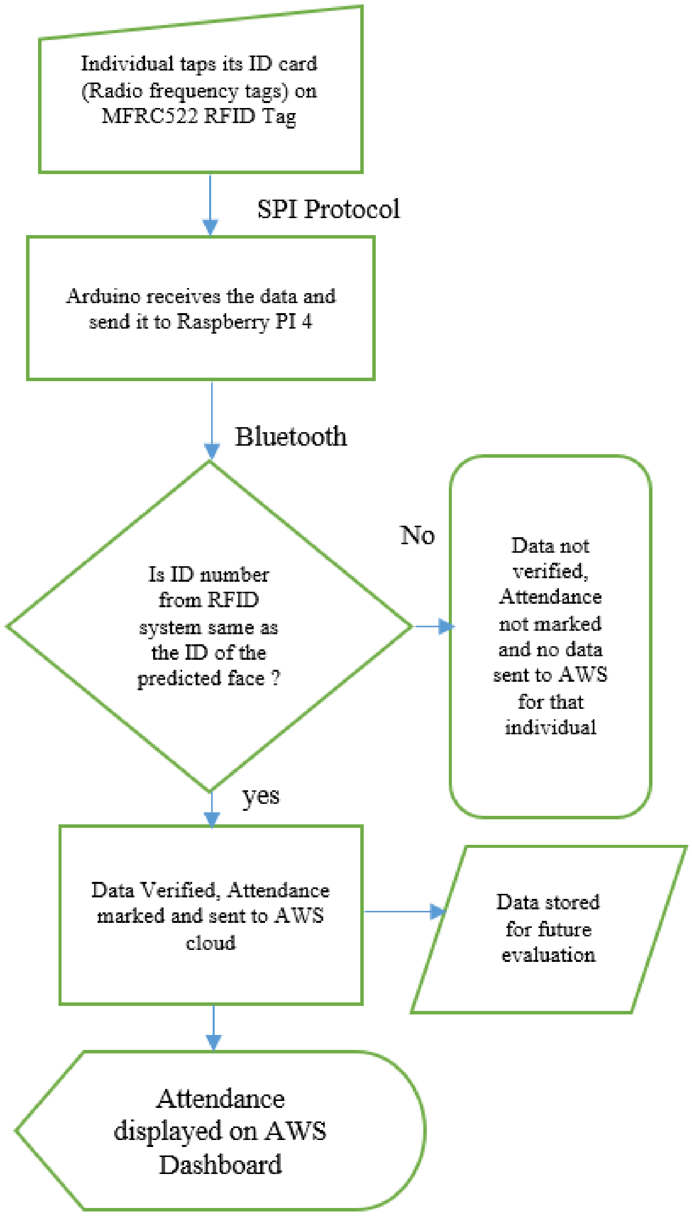

Individual taps its ID card (Radio frequency tags) on MFRC522 RFID Tag
SPI Protocol
Arduino receives the data and send it to Raspberry PI 4
Bluetooth
Is ID number from RFID system same as the ID of the predicted face ?
No
Data not verified, Attendance not marked and no data sent to AWS for that individual
yes
Data Verified, Attendance marked and sent to AWS cloud
Data stored for future evaluation
Attendance displayed on AWS Dashboard

Figure 2 – System flowchart for R.A.M.S

A. *RFID System*

This sub-system is composed of an Arduino connected to an RFID reader/writer and a Bluetooth module. These components are connected between themselves according to the wiring diagram on **Figure 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, 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 Arduino platform has its own software and supports various platforms like Windows, Mac OS, and Linux. In this project, Arduino is used to transmit the data from MFRC522 RFID reader to the Raspberry Pi using 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 Project, 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 which 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. It consumes less power and works at 3.3V. It has a maximum range of 50 mm. In this project, 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: Bluetooth is a wireless communication technology that allows electronic devices to talk to each other when they are in close range. 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 need to act as a Master device to begin the communication. It has 4 pins, VCC, GND to power the device and TXD, RXD to communicate. In this Project, HC-06 is connected to an Arduino and it transmits the Student ID number that the Arduino receives from MFRC522 RFID reader to a Raspberry Pi, which acts as a Master device.

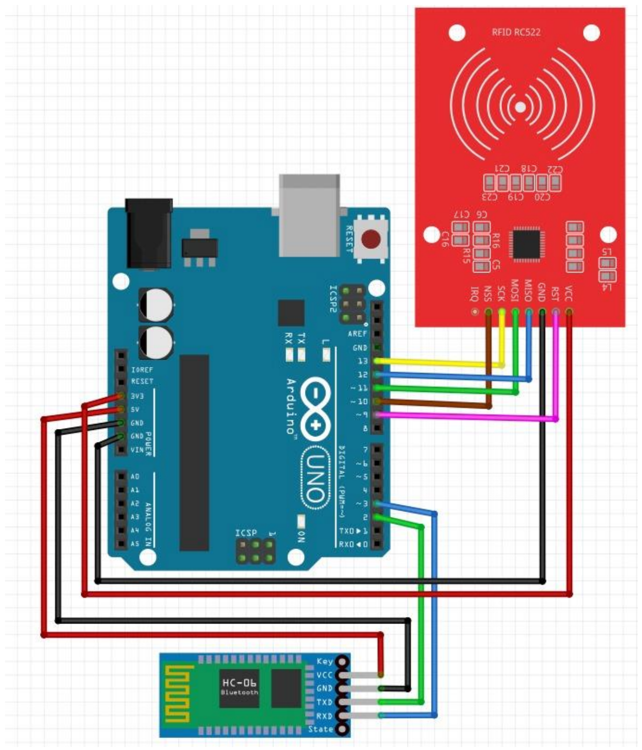


Figure 3 - RFID & Bluetooth connected to Arduino UNO

B. *Artificial Intelligence System*

Facial recognition 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 unless they try to tap the card and get verified again. The hardware for the facial recognition functions is the following:

- Raspberry Pi 4: The Facial Recognition capabilites required a computer with reasonably good memory to work. Because of this, a Rasoberry Pi 4 with 4GB of RAM was utilized. The Rapsberry Pi runs a Raspbian 64-bit OS and has built-in Bluetooth, Serial communication, and Wi-fi capabilities, so it was appropriate for R.A.M.S.

- Logitech C525 Webcam: This webcam was selected due to its autofocus capabilities, the 720 pixel resolution, and relatively low cost (less than $40 in some online stores), not to mention the USB capability. The images were utilized by a Python code in the Rasoberry Pito perform the facial recognition of the students.

There are three parts to the facial recognition system :

Face Detection: [2] The system has to initially detect a face in the frame. For that, Haar Cascade classifier 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 project.

For training 40 images are considered that contain the faces of each individual. There are a few points that need consideration when creating a training database for face detection and recognition. This includes:

- Images should be high in resolution with good lighting 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 R.A.M.S the frontal face detection part of the Haar cascade algorithm is used, which just need a small amount of variation in angles, not 360 degrees [3].

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 with 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 on a complete image. Once the face is detected , it's considered 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.

Face Recognition: Once the faces are detected, facial recognition does its part by using an LBPH (Local Binary Patterns Histograms) face recognizer object which is an algorithm used for facial recognition. It trains using the NumPy Arrays and their corresponding labels. After the data is trained, it is stored in a .yml file.

A .yml file is a human-readable data serialization language [4]. 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 in a consistent way.

On the testing part, the trained model loads the label mappings, and it uses Open CV to capture frames from the camera. For each frame, it detects face first and then applies the trained face recognizer to predict the identity of the face.

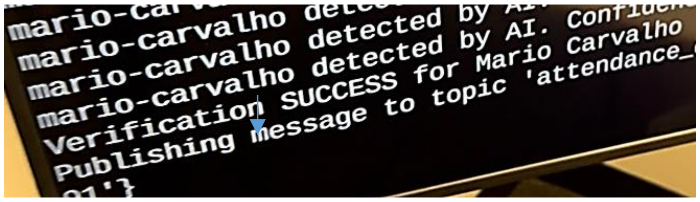
For the facial recognition part there are a few set of lines in code that are important to discuss:

- **cv2.face.LBPHFaceRecognizer_create()** 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 and does the classification of these features as belonging to respective individuals.
- **recognizer.predict(roi_gray)** this predict function will take the roi_gray which is the gray scale ROI ,the face, as input and returns two values: id and the confidence value of this prediction.

Verification: Finally, 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, its attendance is generated (**Figure 4**). 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.

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 (**Figure 5**).



```
mario-car
mario-carvalho dete
mario-carvalho detected by AI
mario-carvalho detected by AI. Confiden
mario-carvalho SUCCESS for Mario Carvalho
Verification message to topic 'attendance_
Publishing
01'}
```

Figure 4 -Verification successful on the terminal. This is the back end of the project and will not appear in practical implementation.

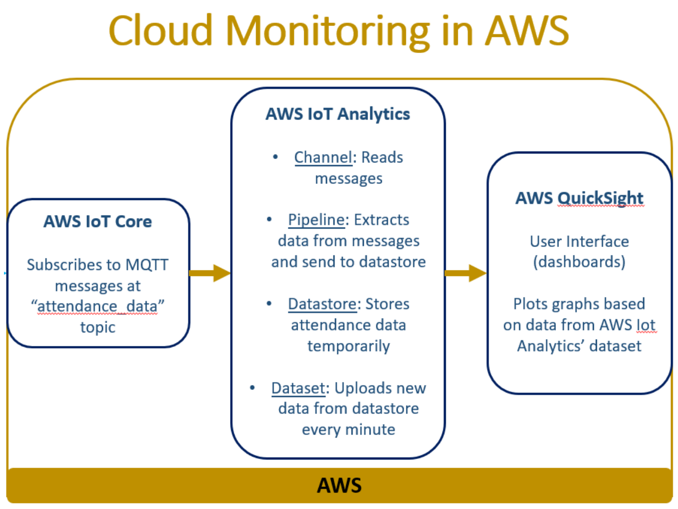
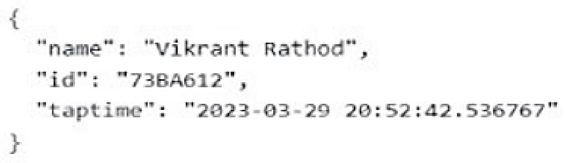

Cloud Monitoring in AWS
AWS IoT Core
Subscribes to MQTT messages at "attendance_data" topic
AWS IoT Analytics
Channel: Reads messages
Pipeline: Extracts data from messages and send to datastore
Datastore: Stores attendance data temporarily
Dataset: Uploads new data from datastore every minute
AWS QuickSight
User Interface (dashboards)
Plots graphs based on data from AWS Iot Analytics' dataset
AWS

Figure 5 – Summary of the functions of each AWS service on R.A.M.S

On AWS IoT Core, the R.A.M.S. Raspberry Pi was configured as a "Thing resource" according to AWS instructions [5], 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 to 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 (**Figure 6**) 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"
}
```

Figure 6 – JSON message format utilized to transmit attendance data via MQTT protocol.

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. These steps were done according to official AWS instructions [6].
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. Such

dashboard is possible through AWS QuickSight, a service that allows for analysis and graphs to be generated based on datasets from other AWS services [7], which includes IoT Analytics datasets.

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. 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 R.A.M.S. 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.

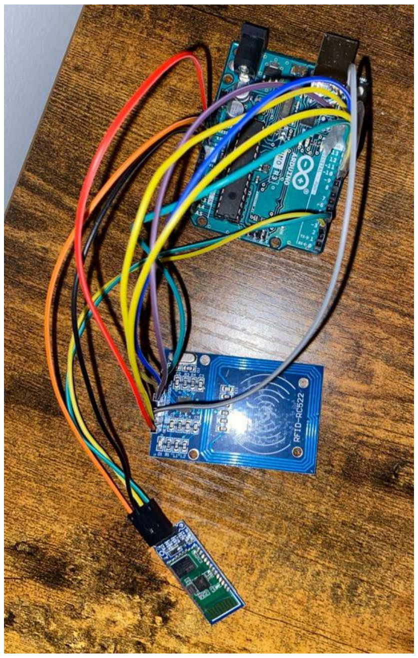
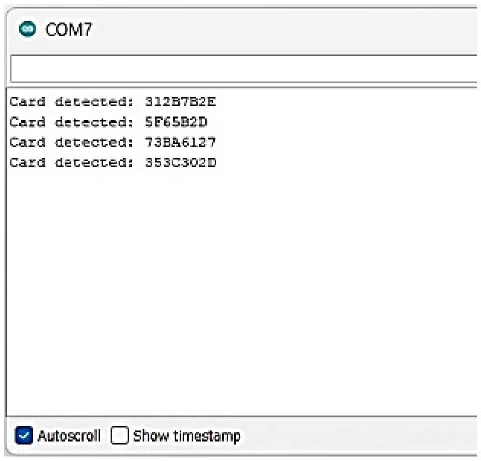


Figure 8 - RFID system setup with Arduino, RFID reader, and Bluetooth module.

III. Results

A. *RFID System*

Once the cards were detected, the serial monitor of the Arduino printed the card IDs (back end process, not visible to students), as shown in **Figure 7**. The hardware setup used for the student ID reader is shown in **Figure 8**.

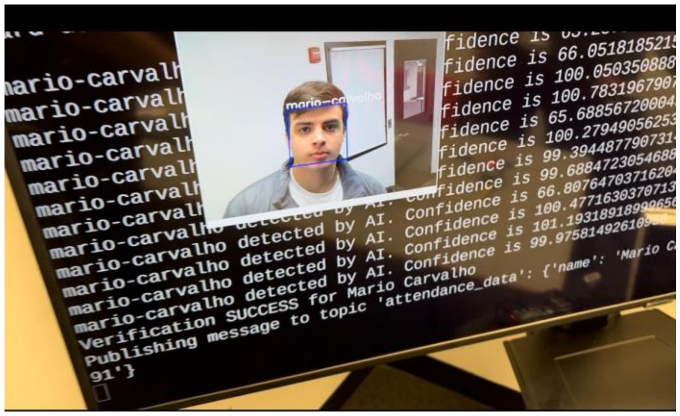


```
Card detected: 312B7B2E
Card detected: 5F65B2D
Card detected: 73BA6127
Card detected: 353C302D
```

Figure 7 - IDs detected at the Arduino UNO terminal after RFID tag is detected via RFID reader.

B. *Artificial Intelligence System*

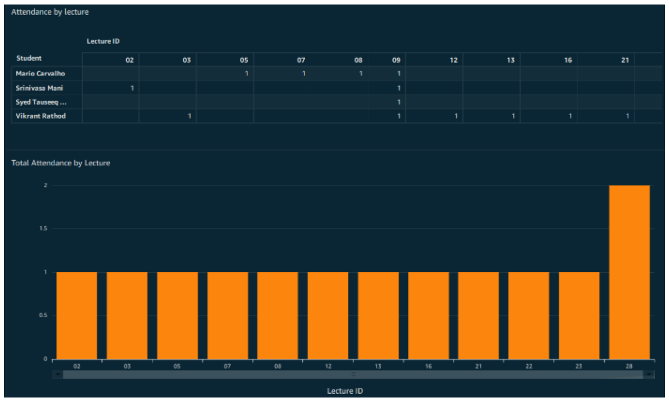
As seen in the display on **Figure 9**, the facial recognition system displays that Mario Carvalho (a student) is recognized. The output of the model is a bounding box around the face of the person along with the name of the person Verification successfully means that the student was detected by RFID and verified by facial recognition and the attendance is marked and sent to AWS cloud.



```
ario-carvalh                 fidence is 66.0518185215
mario-carvalh                fidence is 100.050350888
mario-carvalh                fidence is 100.7831967907
mario-carvalh                fidence is 65.68856720004
mario-carvalh                fidence is 100.2794905625
mario-carvalh                fidence is 99.3944877907314
mario-carvalh                fidence is 99.6884723054688
mario-carvalh                fidence is 66.80764703716204
mario-carvalh         by AI. Confidence is 100.4771630370713
mario-carvalho detected by AI. Confidence is 101.1931891899656
mario-carvalho detected by AI. Confidence is 99.975814926109
mario-carvalho detected by AI. Confidence
mario-carvalho detected by AI. Confidence
Verification SUCCESS for Mario Carvalho
Publishing message to topic 'attendance_data': {'name': 'Mario C
91'}
```

Figure 9 – Output of facial recognition on Raspberry Pi's desktop.

C. *AWS cloud Monitoring Implemenetation*


Attendance by lecture

Student	02	03	05	07	08	09	12	13	16	21
Mario Carvalho			1	1	1	1				
Srinivasa Mani	1					1				
Syed Tauseeq ...						1				
Vikrant Rathod		1				1	1	1	1	1

Figure 10 – Attendance data per student and per lecture for a sample class of 4 registered students.

The dashboard created with AWS QuickSight was a friendly online interface. **Figure 10** shows the main view of the attendance statistics that Professors and other University members can see online for the example class of 4 students created (ECE 500).

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 applications of R.A.M.S., 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.

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. Future work

A. *Arduino encryption*

To address cybersecurity concerns, AES encryption can be added to the card ID data that is transmitted via Bluetooth between the Arduino and the Raspberry Pi.

B. *Hardware user interface*

For testing R.A.M.S, a monitor was used to show the command line terminal of the Raspberry Pi, which showed the students whether their attendance was confirmed or not. For future work, our team could develop a more friendly webpage using HTML to show the outputs of the Raspbian terminal window.

C. *Improving the accuracy of the Facial Recognition System*

We are going to work on improving the accuracy of facial recognition. This includes exploring more algorithms and techniques. Our team is going to use our computer vision knowledge and understanding into creating our own feature extraction techniques. This can be possible if we first implement various past algorithms and compare their technique and the difference in the result, accuracy, efficiency, and memory consumption. With the learning process, we would be able to work on making our own feature extraction algorithm

D. *AWS QuickSight dashboard*

Only one table and one graph is displayed on the AWS QuickSight dashboard currently. For the future development of R.A.M.S, more graphs can be added to provide a deeper analysis of attendance statistics, e.g. the percentage of students that attended lectures on Tuesday vs. percentage on Thursday.

V. 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 R.A.M.S ready for deployment in a big scale, 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.

VI. References

[1] What is RFID? How It Works? Interface RC522 RFID Module with Arduino. Available at: https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/

[2] OpenCV Python TUTORIAL #4 for Face Recognition and Identification https://www.youtube.com/watch?v=PmZ29Vta7Vc

[3] The Computer Vision Pipeline, Part 3: image preprocessing. Available at: https://freecontent.manning.com/the-computer-vision-pipeline-part-3-image-preprocessing/

[4] What is YAML? The YML File Format. Available at: https://www.freecodecamp.org/news/what-is-yaml-the-yml-file-format/

[5] Connect a Raspberry Pi or other device https://docs.aws.amazon.com/iot/latest/developerguide/connecting-to-existing-device.html

[6] Getting started with AWS IoT Analytics (console) https://docs.aws.amazon.com/iotanalytics/latest/userguide/quickstart.html

[7] What is Amazon QuickSight? https://docs.aws.amazon.com/quicksight/latest/user/welcome.html