



**ASHESI UNIVERSITY**

**AN ACCESS CONTROL FACIAL RECOGNITION  
SYSTEM USING TINYML: HARDWARE COMPONENT**

**CAPSTONE PROJECT**

B.Sc. Computer Engineering

**Romel Nerkwei Nettey**

**2022**

**ASHESI UNIVERSITY**

**DESIGNING FACIAL RECOGNITION SYSTEM WITH TINYML**

**CAPSTONE PROJECT**

Capstone Project submitted to the Department of Engineering, Ashesi University  
in partial fulfilment of the requirements for the award of Bachelor of  
Science degree in Computer Engineering.

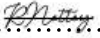
**Romel Nerkwei Nettey**

**2022**

## Declaration

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....

Candidate's Name: Romel Nerwkei Nettey

Date: Monday, April 27th, 2022

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

.....

Supervisor's Name:

.....

Date:

.....

## Acknowledgements

I would like to thank God for the grace and strength to be able to come this far. He is my everything. I would also like to thank my supervisor, Mr. Francis Gatsi, for guiding me and giving his support and advice. My family deserves a special mention as they supported me throughout and encouraged me. To my friends, who always checked up on me and provided support when they could, I appreciate you all.

## **Abstract**

Technological advancement brought about improvements in the way IoT systems are implanted. The introduction of Tiny Machine Learning (TinyML) and edge computing has made it possible to process data at the edge and carry out machine learning processes at the edge without relying solely on cloud platforms for such purposes. Due to the advancement in technology, security can be improved and achieved at a cheaper cost. This project exploits TinyML technology to develop an access control system that uses facial recognition to control access to a room. The proposed system sends data that contains the name of the person who accessed the room and the time the room was accessed. This allows the user to monitor when and who accessed a room. The project focuses on the hardware aspect of the system. The system was able to grant access to authorized persons and can also be controlled from the web application.

## Table of Contents

<b>DECLARATION.....</b>	<b>I</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>I</b>
<b>ABSTRACT .....</b>	<b>III</b>
<b>TABLE OF CONTENTS.....</b>	<b>IV</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 BACKGROUND .....	1
1.2 PROBLEM STATEMENT.....	3
1.3 AIMS AND OBJECTIVES .....	3
1.4 SCOPE OF WORK.....	3
<b>CHAPTER 2: LITERATURE REVIEW .....</b>	<b>4</b>
<b>CHAPTER 3: DESIGN .....</b>	<b>7</b>
3.1 DECISION .....	7
3.2 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS .....	8
3.3 SYSTEM CONSTRAINTS .....	9
3.4 DESIGN PROCESS.....	9
3.4 DESIGN PROCESS.....	10
3.5 SYSTEM COMPONENTS .....	11
3.6 CIRCUIT SCHEMATIC .....	13
3.7 SYSTEM FLOWCHARTS.....	15
<b>CHAPTER 4: METHODOLOGY.....</b>	<b>18</b>

4.1 SETTING UP THE CAMERA .....	18
4.2 FACIAL RECOGNITION MODEL .....	19
4.2.1 <i>Face detection</i> .....	20
4.2.2 <i>Face recognition</i> .....	22
4.2.3 <i>Local webservice</i> .....	23
4.3 WAKE WORD SYSTEM (ALTERNATIVE DESIGN) .....	24
4.4 DATABASE AND APPLICATION .....	29
5.5 CIRCUIT .....	29
<b>CHAPTER 5: RESULTS &amp; CONCLUSION .....</b>	<b>30</b>
5.1 RESULTS .....	30
5.1.1 <i>Results for wake word system</i> .....	30
5.1.2 <i>Results from main system</i> .....	31
5.2 CONCLUSION .....	32
5.3 LIMITATIONS .....	33
5.4 FUTURE WORK .....	33
<b>REFERENCES.....</b>	<b>34</b>





# Chapter 1: Introduction

## 1.1 Background

The advancement in technology has brought a lot of improvement to IoT technologies and how it usually functions. IoT is a network of general objects which are embedded with technologies that help to communicate and interact within themselves and the external environment [1]. A basic IoT architecture is the three-layer architecture consisting of the perception layer, also known as the physical layer, the network layer, and the application layer. With this basic architecture, the physical layer is responsible for gathering necessary data or sensing physical parameters within a particular environment. The network layer connects smart objects, servers and transmits and aids in the processing of gathered data. The application layer provides some form of service to a user. Usually, the devices at the physical layer level just transfer the data they gather from the edge to the cloud so some form of processing can be done. Then, the processed data is sent to the application layer for some form of service or display of the processed data. As the number of connected Internet of Things (IoT) devices increases, the amount of data generated by these devices puts pressure on the traditional centralized cloud computing infrastructure [2]. Aside from that, the traditional centralized cloud computing infrastructure comes with some form of disadvantages or challenges. This involves latency issues, reliability, bandwidth, and security. Since a centralized cloud computing infrastructure is solely responsible for processing data, high latency issues may come up, causing delays in delivering the service or action at the application layer. Some of these challenges have been addressed by the introduction of Edge computing. Edge computing is a distributed information technology (IT) architecture in which client data is processed at the periphery of the network, as close to the originating source as possible [3]. Edge computing moves the computing resources and storage from the

central data centre to the ‘edge’ where data is produced and accumulated. Edge computing provides us with a lot of opportunities that can be explored, including bring machine learning to the edge.

Machine learning (ML) is a dynamic and powerful field of computer science that has permeated nearly every digital thing that we interact with, be it social media, our cell phones, our cars, or even household appliances [4]. With the traditional centralized cloud computing infrastructure, the data is sent to the cloud for ML to take place, but with edge computing and Tiny Machine learning (TinyML), processing and ML can be done at the edge without the need for a central data centre. The rapid growth in miniaturization of low-power embedded devices and advancement in the optimization of machine learning (ML) algorithms have opened up a new prospect of the Internet of Things (IoT), tiny machine learning (TinyML), which calls for implementing the ML algorithm within the IoT device [5]. TinyML simply involves the use of relatively cheap devices such as microcontrollers and embedded devices to run machine learning algorithms to achieve machine learning at the edge and obtain real-time responsiveness. TinyML framework in IoT is aimed to provide low latency, effective bandwidth utilization, strengthen data safety, enhance privacy, and reduce cost [5]. TinyML provides cost-effective solutions as it overcomes most of the challenges facing the traditional centralized infrastructure and also allows devices to perform ML without constantly accessing the cloud for data processing.

## **1.2 Problem Statement**

The traditional architecture used in IoT comes with some challenges, such as high latency, bandwidth limitations, questionable data safety, privacy issues and relatively high cost. Edge computing and TinyML have opened up new prospects of IoT, which manages to address most of these challenges. This advancement in technology has made it possible to improve security systems making them more robust. Security systems that leverage smart technology such as facial recognition are usually expensive because of the components used in implementing such systems. This project focuses on developing a facial recognition security system using cheaper components such as low-power embedded devices or microcontrollers with TinyML.

## **1.3 Aims and Objectives**

This project involves designing and building a system that gives or denies access to an enclosed space by using facial images with TinyML. The facial images are captured by a camera connected to an edge device that can perform machine learning to authenticate the image to give or deny access. A notification is sent to an appropriate authority, while an alarm is triggered when there is a security breach.

## **1.4 Scope of work**

This project involves building a system that implements facial recognition technology to improve security in situations where only a select number of people should have access, such as a Data Center and the vault room of a bank. A TinyML device will host the software, and a camera will be used to capture the images. Edge computing is involved in this project to process facial images and an IoT implementation for automating the process of granting access and security alerts.

## Chapter 2: Literature Review

The facial recognition security system designed rely on a leading technology — TinyML (Tiny Machine Learning). Other relevant work, books, and articles that inspired the design and idea of the system are summarized and evaluated in this part section.

Facial recognition systems mostly require significant computing resources and adequate or high computing power to perform at an optimum level. In most cases, Machine learning systems rely on cloud services to effectively run and produce accurate results. Either cloud services are used in making facial recognition security systems or computers with high specifications or the necessary hardware resources are used in building such systems. Authors such as S Jahnvi and C. Nandhini built a smart anti-theft locking system [6] that uses the facial recognition and detection models, a Raspberry Pi, an IR sensor and a stepper motor. The facial detection algorithm used in this system is the Viola Jones algorithm, and the facial recognition algorithm used is the Local Binary Pattern algorithm (LBP). With this system, the Raspberry Pi camera is in a state of idleness until the IR sensor detects an obstacle. When detection occurs, the camera is triggered, and it captures the detected object. The Viola Jones algorithm then processes the image for face detection. Once the processing is done and a face is detected, the facial recognition process begins. The algorithms used to provide high accuracy the in detection and recognition of faces. The LBP algorithm highlights the facial characteristics of the original image by creating an intermediate image [6]. Another algorithm, the Principal Component Analysis (PCA), is used to extra the local features of the image and then the face image is compared with the images trained in the data set. Access is then granted if there is a match and if not, an alarm is set off, and access is denied. This system relies on the use of the Raspberry Pi as the central processing unit so such complex and resource demanding algorithms can be used. The

single-board computer is much more expensive than some low-powered embedded devices and microcontrollers. Though the system has very effective algorithms, the Viola-Jones algorithm which is used for face detection trains the model too slow.

An IoT based automated door accessing system for visually impaired people was proposed by Farhat Lamia Barsha et.al [7]. This system includes a camera that is attached to a door to detect and recognize people that try to access the door. The main features of the system are visitor authentication using face recognition, voice command, suspicious activity detection using audio alert and recognizing harmful objects visitors may carry using object and metal detection. The components that were required for the hardware implementation are Raspberry pi, Arduino, ultrasonic sensor, sonar sensor, webcam, and a speaker. The Raspberry Pi 3 was used as the main processing unit. When a person is detected by the ultrasonic sensor, the camera is turned on it captures an image of the person. The portion of the image containing the face is cropped and facial recognition begins to determine if the face matches any of the images stored in the database. The system also generates a sound indicating if the person has access to the door or not when facial recognition has occurred. This system has a lot of good features such as metal detection and the inclusion of voice alert. Since a Raspberry Pi is being used as the central processing unit, the necessary resources are provided for the facial detection and recognition to run effectively. Aside from the use of the Raspberry Pi, the Arduino is also used for the metal detection part of the system. The system involves a lot of components and some complexities. A Raspberry Pi and an Arduino are both use to develop this system making it relatively expensive to develop.

In [8], a facial recognition security system was also designed using a single board, the Raspberry Pi. With this system, a mobile application was developed to interface with the security system. When a face is detected and recognized a notification is sent to alert

the owner. The owner then must grant access or not to the person trying to gain access. This system requires the owner to give authorization even though recognition has already been made. The second authorization seems redundant, but it becomes quite understandable if it was designed with the idea of two-step verification in mind. In the case where the owner does not see the alert, the person trying to get access does not acquire access to the door.

In [9] the authors proposed a wearable device and a smart doorbell home automation system. It uses biometrics to provide authentication. The system captures information such as Iris image, face image and voice of the visitor. When a visitor comes to the entrance of the door, a PIR sensor detects the visitor's presence and sends a signal to a control system [9]. An alert is sent to the user by a control system and the system activates a voice system to guide the visitor through a series of steps to take data from the visitor such as Facial data, Iris, and voice data. All the data is stored in a database which the user can see using a wearable device [9]. The data is transferred via Wi-Fi to the database. The system also provides live feed from the camera using Wi-Fi. The system is secure as it uses different form of authentication and not just facial recognition. It also gives a lot of control to the user through the wearable device. In situations where the Wi-Fi is unstable the user loses control over the system. Unreliable Wi-Fi may also cause latency and delays when storing the captured data from the visitor.

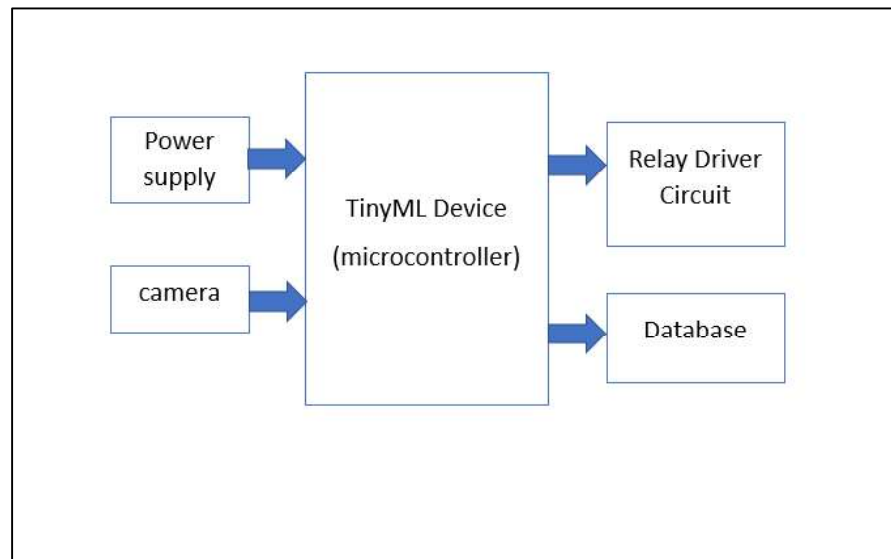
The research papers based on facial recognition normally develop the system using a single-board computer. These computers are more powerful than other constrained devices and are more expensive. Developing a final product with such components may lead to an expensive design.

## Chapter 3: Design

The system in this project comprises two subsystems: the hardware subsystem and the software. The hardware basically consists of the central point where processing would occur and other components that respond to the processed data results. The software aspect provides some form of control to the system.

### 3.1 Decision

The system requires the right microcontroller to be selected in order to develop the system. Not all microcontrollers have enough resources to be able to run Machine learning algorithms and carry out IoT processes, so a decision had to be made on the suitable microcontroller to be used for the central processor of the system. The microcontroller interfaces with the application, the solenoid lock, and LEDs for indication purposes. A general block diagram was designed to show how the components relate to each other; a general overview of the system.



*Figure 3.1 A general block diagram of facial recognition access control system with TinyML*

### **3.2 Functional and Non-Functional Requirements**

- The system should have low energy consumption.
- The system should be able to work both using an external source of power and also an internal source(batteries)
- The system should be able to last for more than 8 hours when dependent on internal source of power before needing to be charged.
- The system should be able to detect any tampering and go into a lockdown state to ensure security.
- The system should be low cost and affordable
- The system should be able to be controlled from the web application
- The system should be able to keep logs on when and who accessed a system; that is, including the name of the person, the time of access and the room accessed
- The system should be easy to use without the need for prior training
- The web application should be user-friendly
- The system should provide real-time information
- The system should be power efficient
- The system should be robust and durable



### **3.3 System Constraints**

A significant constrain of the system is cost, power, and size. One of the projects main aims is to implement Tiny Machine learning and apply in a use case such as security. Since the focus of the project is centered around edge computing and TinyML the central devices that can be used in this system are microcontrollers instead of the use of higher-class devices (in terms of specifications) such as single-board computers. Single board computers have a high storage, higher RAM, peripherals, possible GPU, and other higher specifications that are not available on lower-class devices. Different microcontrollers had to be carefully looked at to select the best microcontroller that could efficiently run Machine Learning algorithms.

### **3.4 Design process**

Using the above requirements, components were selected for the system. The first decision was for the suitable microcontroller to be used for running the Machine learning algorithm. There are two primary specifications to look at when picking a microcontroller for this system. The first one is whether it has the capability of running a machine learning algorithm or library, and the other is if the microcontroller can support some form of wireless communication as the central system would be communicating with a database and a web application. Another specification that must be considered also is if the microcontroller is compatible with a camera module.

Through various research made, there are two ways that can be used to run machine learning on the microcontroller. One way that can be used to achieve this is by using a cloud platform such as TensorFlow Lite and edge impulse to train and generate a model. Another way is by using a library and adding the facial image data that would be processed and recognized.

### 3.4 Design process

The following Pugh chart was used in selecting the microcontroller for the system. The matrix used includes three scores that is given to each component depending on the criteria. The scores are ‘0’ which is the baseline, ‘+’ which has a weight greater than that of the baseline and ‘-’ which is lower than the baseline and has a negative weight. The baseline in this case is a single-board computer, particularly the raspberry-pi 2. Table below illustrates the selection process.

*Table 3.1 Pugh chart for the brain of the system*

criteria	weight	Raspberry-pi 2 (baseline)	Raspberry pi pico	Arduino-nano ble sense	Esp32-CAM
Efficiency (energy)	3	0	+	+	+
Cost	2	0	+	-	+
Flash memory size	3	0	2MB+)	1MB(-)	4MB(+)
Maximum CPU frequency	1	0	133MHz (+)	64MHz(-)	240MHz (+)
compatibility(camera)	2	0	+	+	+
wireless communication(Wi-Fi)	2	0	-	+	+
Total	--	0	11	4	13

### 3.5 System Components

#### *Ideal System Design*

An ideal system for the project would be constructed using the components mentioned

below. The ideal design takes into consideration the requirements mentioned earlier.

- a) The Esp-32 microcontroller was selected because of its low cost, relatively high memory, and adequate processing power.



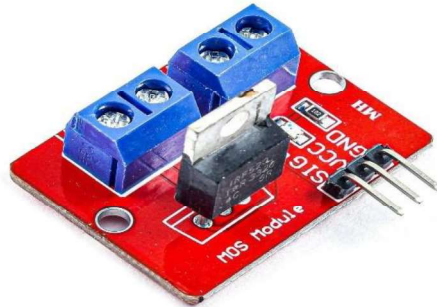
*Figure 3.2: An image of the esp32-cam*

- b) The ov2640 camera module was selected because it is compatible with the esp32 and it provides high resolution and has an on-chip JPEG encoder, which offloads processing power from low-end MCU.



*Figure 3.3: An image of the ov2640*

c) An IRF520 MOSFET Driver Module is selected to drive high voltage to the solenoid lock when detection occurs. The module is designed to switch heavy DC loads from a single digital pin of the microcontroller



© Photo by ElectroPeak

*Figure 3.4: An image of the ov2640*

d) A solenoid lock is selected to lock or unlock when access granted or not by the microcontroller. The lock has an unlocking power mode and locking power-on mode. It can stay in any of the states.



*Figure 3.5: An image of a solenoid lock*

### 3.6 Circuit Schematic

The circuit diagram was created using the EasyEda online platform. The power supply used to power the Esp32-CAM microcontroller is the 5v power supply. The microcontroller requires 5 volts to keep running. If the power being supplied is not enough, the board would not be able to run efficiently. Also, any amount of power supply greater than 5v would damage the board.

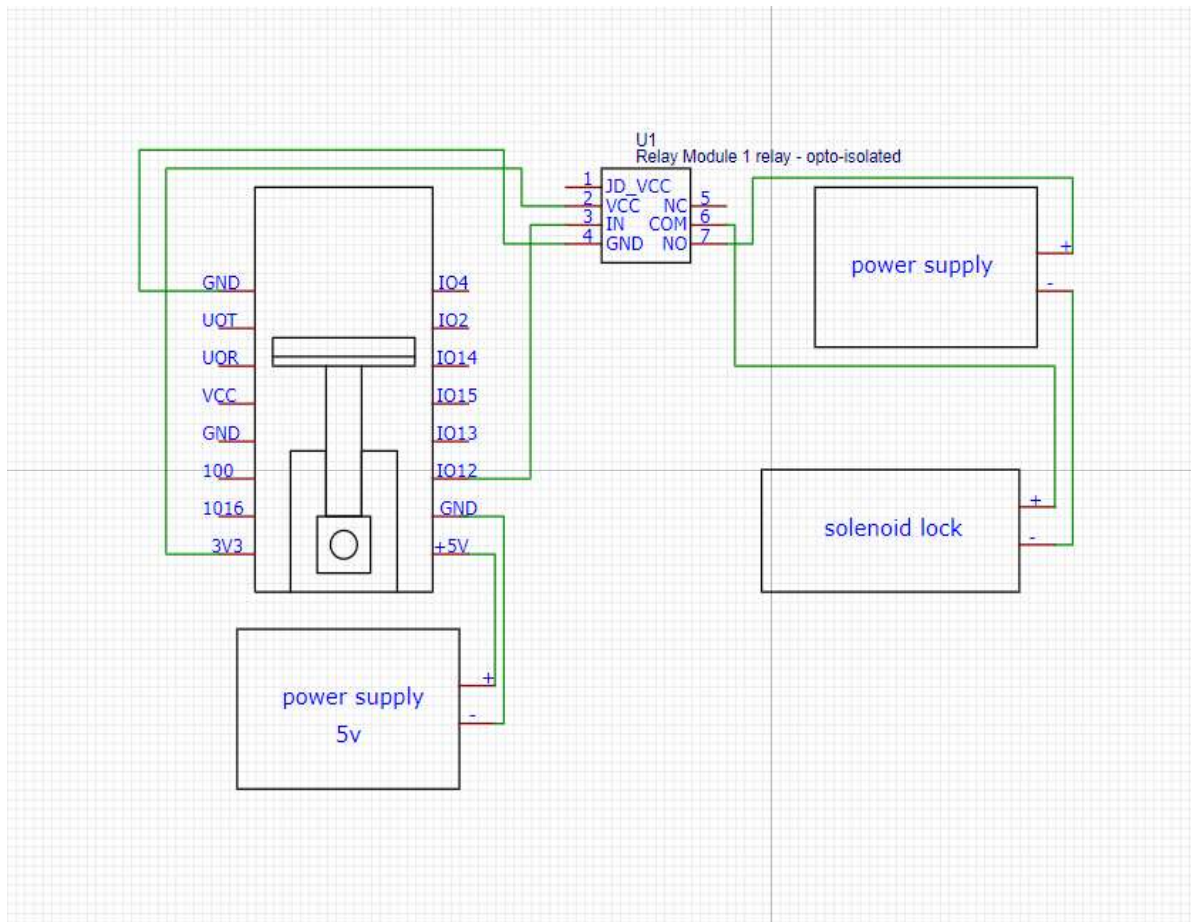


Figure 3.6: Circuit schematic of the system design

Figure 3.6 consist of the esp32-cam, which is the 'brain' of the system, power supply, and a relay module. The esp32-cam is connected to the 5v power supply using the 5v pin and the ground pin on the board. The solenoid lock requires 12 volts in order to operate. There are two power modes for the system, which are unlocking power-mode and the locking power-on mode. The microcontroller controls the solenoid lock based on the response from the facial recognition process. The Esp32-CAM cannot provide the 12v necessary to lock and unlock the solenoid. A relay is therefore, used to provide the 12v needed for by the solenoid lock. The second GPIO pin is connected to the 'IN' port of the relay module. When the recognition takes place and the face detected is authorized, the pin sets to high, triggering the relay to supply power to the solenoid lock. The common of the relay is connected to the positive wire of the solenoid lock. The ov2640 camera module can be easily connected as the esp32-cam has an open slot for the camera module, where the camera can be placed in and locked in place by a casing on the board.

### 3.7 System Flowcharts

The following flowchart depicts the system's behaviour.

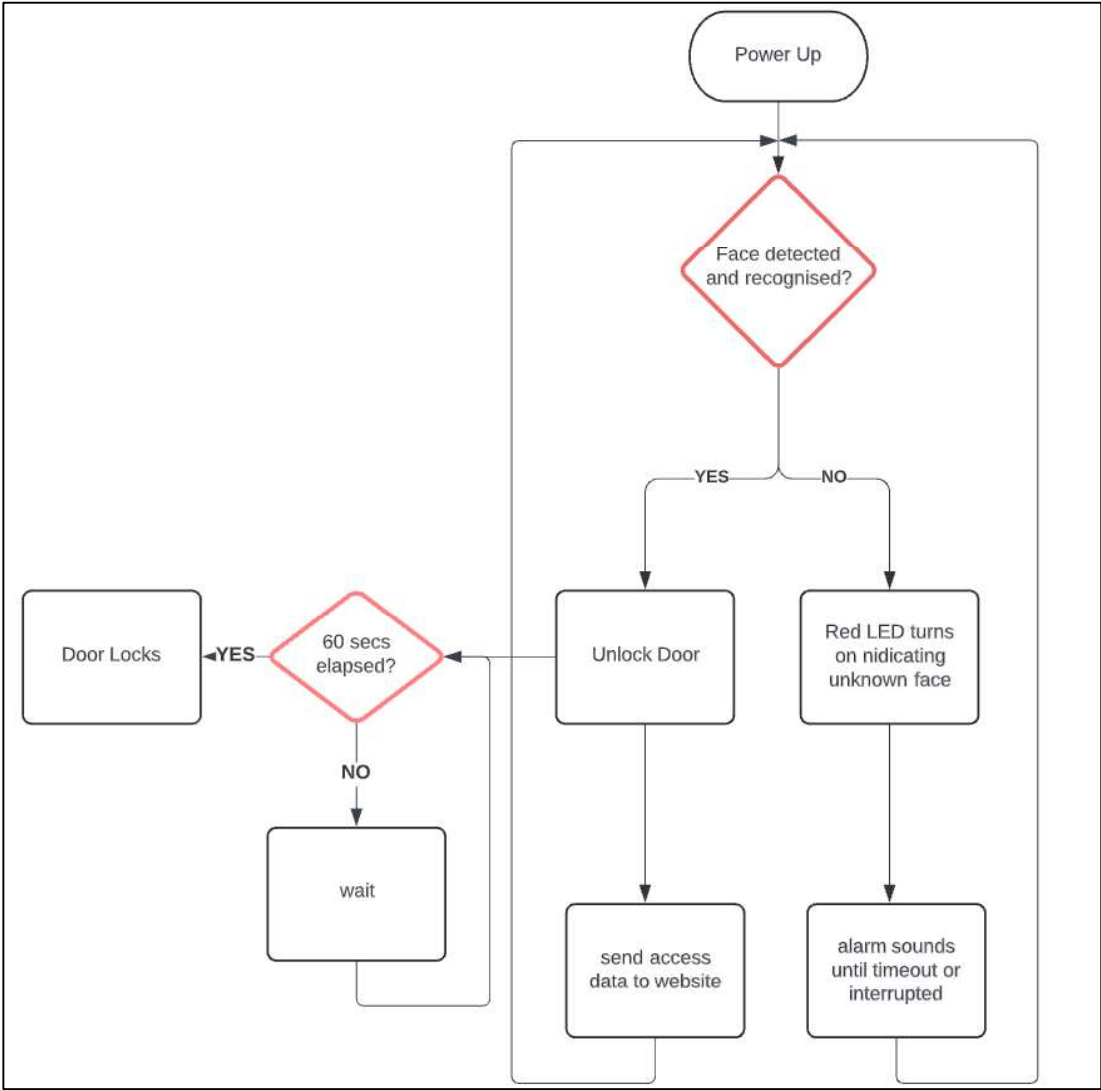


Figure 3.7: basic flowchart design

### Alternate flowchart design

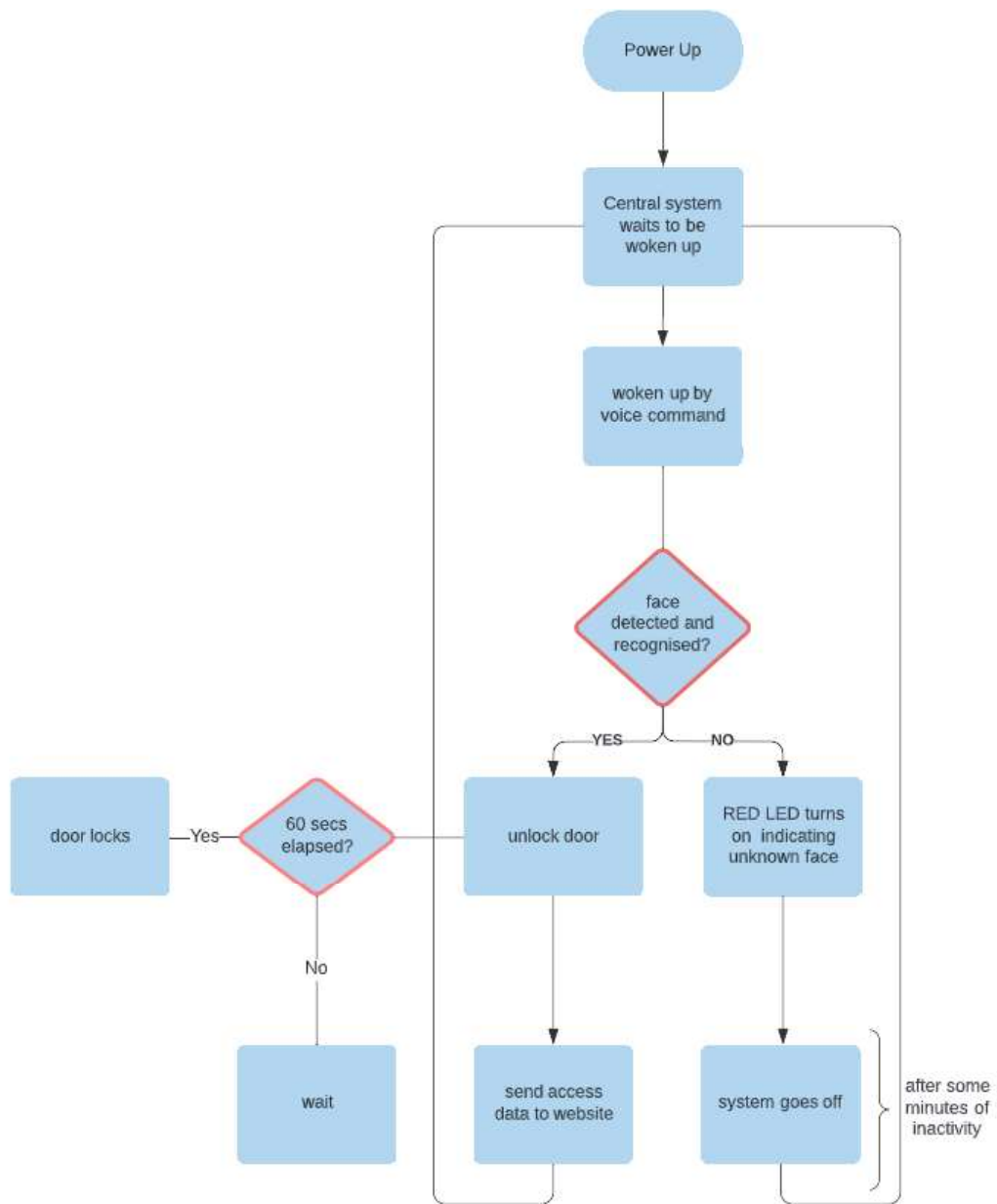


Figure 3.8: alternate flowchart design



Figure 3.8 is an alternative design is developed for the microcontroller running the facial recognition model. The model running on the board requires a lot of processing power and thus results in the board heating up sometimes when the model is running. The alternate design is just added to protect the board, so it is not running continuously. TinyML is implemented at the start of the flowchart. A microcontroller also running a machine learning algorithm waits for a wake word to be uttered and once the microcontroller picks it up, the Esp32-CAM is turned on and the facial recognition process begins. If the system is inactive for some minutes, the Esp32-CAM is turned of until the wake word is spoken again.

## Chapter 4: Methodology

This Chapter explains the steps taken to build, test and implement the project, as described earlier. Concerning the hardware aspect of the system, a breadboard was used to connect the components. For the software section, code was typed, and a facial recognition library was used and uploaded to the ESP32-CAM using the Arduino IDE and an Arduino uno.

### 4.1 Setting up the camera

The camera, being an essential part of the project must be set up.

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
```

*Figure 4.1: snippet of configuration code for ov2640*

The code in the figure above configures each of the pins of the ov2640 camera for the ESP32-CAM microcontroller so the microcontroller can access the camera and camera can communicate with the board effectively.

After configuring the camera, we then initialize the camera, so it begins to communicate with the board.

```

esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
}

```

Figure 4.2: initialization code for ov2640

The `esp_camera_init` function carries out the initializing process. If the initialization process fails, the camera would not be accessible.

## 4.2 Facial recognition model

The most important aspect of the project is the facial recognition model that would be used to detect and recognized specific faces that are trained in the dataset by the model.

The ESP\_WHO platform provides libraries for Espressif chips such as the ESP32, that can be used to carry out face detection and recognition. ESP-WHO is basically an image processing development platform. The ESP32-CAM consists of the ESP-WHO module that supports facial detection and recognition. Firstly, a face must be detected and this is achieved by using MTCNN. MTCNN stands for multi-task cascaded convolutional networks [10]. It is a framework that enables the possibility of face detection and face alignment. After a face is detected, another framework called MoblieFace is used to carry out the recognition. The ESP\_WHO platform consists of various libraries that can be used to develop Machine learning applications. Some examples on the platform are human face detection, motion detection and a deep learning library. The facial recognition library would be used to build the system in this case.

The code uses a webserver library to enable the board to stream the camera via an IP address. From the webserver the view of the camera can be accessed a long with all the settings and configurations for the camera. From the webserver, the settings of the camera

can be change. The ESP-WHO framework requires a resolution of QVGA (320x240) or below. Since the microcontroller has limited resources, the facial recognition model cannot run with high resolutions. The resolution was set to QVGA. The code setting the resolution to QVGA is shown below.

```
sensor_t * Fs = esp_camera_sensor_get();  
Fs->set_framesize(Fs, FRAMESIZE_QVGA);
```

*figure 4.2: setting camera resolution to QVGA(320x240)*

To run the facial recognition/detection model, three files from the library must be included. ESP-WHO library contains the fd\_forward.h file, fr\_forward.h and the fr\_flash.h file. The fd\_forward.h file is what would be used for the facial detection. A set of global variables and objects are then then defined and initialize for the implementation of the detection/recognition process.

```
camera_fb_t *fb;  
box_array_t *detected_face;  
dl_matrix3du_t *image_matrix;  
dl_matrix3du_t *aligned_face;  
dl_matrix3d_t *face_id;  
face_id_node *face_recognized;  
mtmn_config_t mtmn_config;
```

*figure 4.3: image showing code of global variables and objects defined*

#### 4.2.1 Face detection

'\*fb' is the frame buffer pointer which will hold the pointer to the pictures captured by the ov2640 camera. The pointer **\*detected\_face** points to a struct holding information from a face that has been detected by the camera. The **\*aligned\_face** pointer points to a struct holding information based on whether a face is lined correctly or not. The information held

by this pointer is used in the face recognition process. The **\*face\_id** points to a struct that contains all the results of the face recognition processes and stores the faces that are enrolled so a comparison can be done later when a face is detected and the recognition process begins again. All the configuration parameters for the face detection process is contained in the struct **mtmn\_config**.

After all the preparation of the global variables and objects is done, the face detection process can start. Before the facial recognition process can begin, a face must be detected first.

```
fb = esp_camera_fb_get();  
fmt2rgb888(fb->buf, fb->len, fb->format, image_matrix->item);  
detected_face = face_detect(image_matrix, &mtmn_config);
```

*figure 4.4: image showing a code snippet of the face detection*

The first function in the code snippet above, **esp\_camera\_fb\_get()**, takes the picture from the ov264 camera. To carry out the detection process the image received must be in a plain bitmap format so it must be converted. The second line of code carries out the conversion on the image. The last parameter, which is the **image\_matrix**, contains the converted bit map picture. After the conversion is done, the **face\_detect()** function is called, using the **image\_matrix** and the **mtmn\_config** as the input parameters. The function contains all the information from the detection process. The value of the detected face struct becomes true when a face is detected.

#### 4.2.2 Face recognition

Before the facial recognition process begins, the *detected\_face* variable is checked first to see if a face has been detected by the face detection process. When a face is detected, the *align\_face* function is called. This function takes the *detected\_face* struct and bitmap information and crosschecks to determine if the face detected is properly aligned. The facial recognition process would not run properly if the face is not well aligned. Once the face is aligned, the information of the aligned face is obtained by calling the *get\_face\_id()* function. Recognition can only take place by comparing the *face\_id* of stored faces to the *face\_id* of the detected faces. The characteristic information of aligned faces is contained in the *face\_id*. The *recognize\_face\_with\_name()* compares the face id with the stored id and if a face is recognized, *face\_recognized* variable would be true.

```
if (detected_face) {  
  
    if (align_face(detected_face, image_matrix, aligned_face) == ESP_OK) {  
  
        face_id = get_face_id(aligned_face);  
  
        if (st_face_list.count > 0) {  
  
            face_recognized = recognize_face_with_name(&st_face_list, face_id);  
  
            if (face_recognized) {  
                face_detected(face_recognized->id_name);  
            }  
        }  
        dl_matrix3d_free(face_id);  
  
    }  
}
```

Figure 4.5: image showing a code snippet of the face recognition

### 4.2.3 Local webservice

The Esp32-CAM contains a WebSocket library makes it possible to stream the camera view over a local webservice. The ESP\_WHO facial recognition/detection library interfaces with the website, where faces can be viewed, detected, and enrolled. The webservice is used to detect and enroll faces to the flash memory of the Esp32-CAM board so the recognition can take place by comparing the stored face id with the face being detected by the camera. The library provides a default look to the website, but it would be altered for this project.

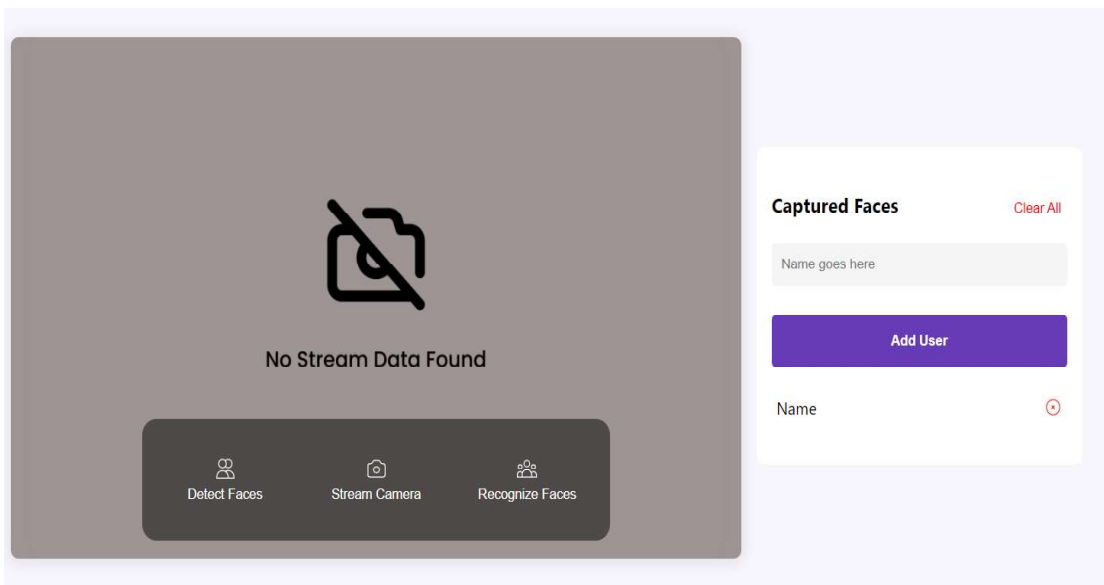


Figure 4.6: an image of the local webservice stream and controls

From the webservice, the stream is started so the view of the camera is visible. The detect faces button triggers the facial detection code explained in *chapter 4.1.1* and when it is triggered the detection process begins. After the detection process begins a user is added by first typing in the name in the text box and triggering the enrolling process by clicking on the 'Add user' button. The ESP-WHO facial recognition algorithm is designed to use five samples images of the face being recognized. After the sample is taken, the user's face can then be recognized whenever the camera is accessed.

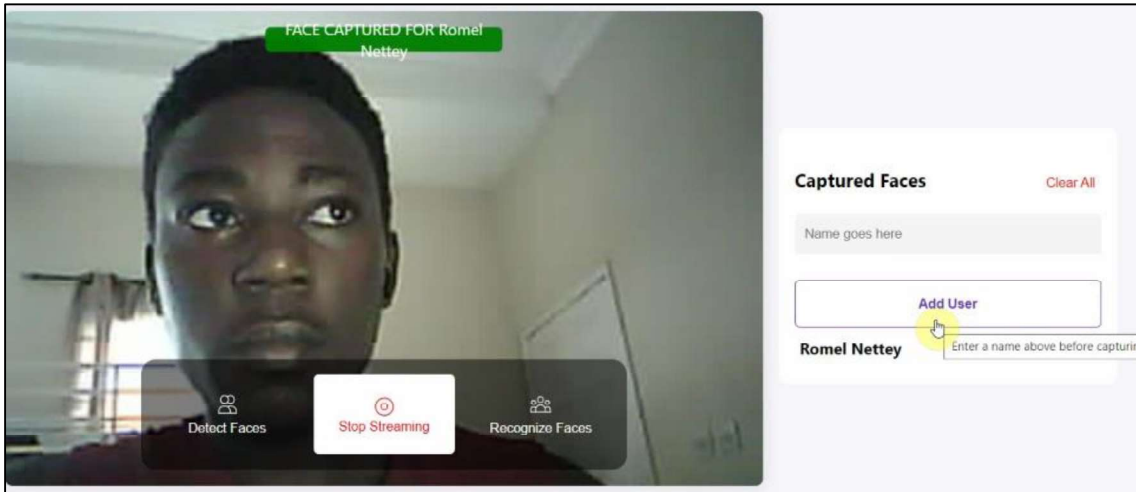


Figure 4.7: an image of the local webserver indicating a user has been enrolled

The user face Id is stored in the memory of the microcontroller and would be accessed whenever the recognition process begins. After enrolling the faces needed, the webserver is ended as it is no longer needed.

#### 4.3 Wake word system (alternative design)

This part of the system is an extra feature added to the system, so the microcontroller board is not consistently running since the facial recognition algorithm is quite tasking. For this part of the system, an Arduino nano BLE 33 sense would be used to run our machine learning model to detect speech. The edge impulse platform would be used to train the model needed for the speech recognition and generate a library. The library is added to Arduino and the code is modified to wake up the board when the wake word “start recognition” is uttered and picked up by the board. The Arduino nano contains an in-built speaker, so an external speaker is not needed.

On the edge impulse platform, the first step taken is to collect the data needed. The microphone on the laptop was used to collect the data. The keyword “start recognition” was uttered continuously through the microphone for ten seconds with short pauses after every



utterance. The audio data is then split into parts. Only the parts of the data containing the uttered keyword are kept. The platform makes it easy to split the audio by just clicking on the split button.

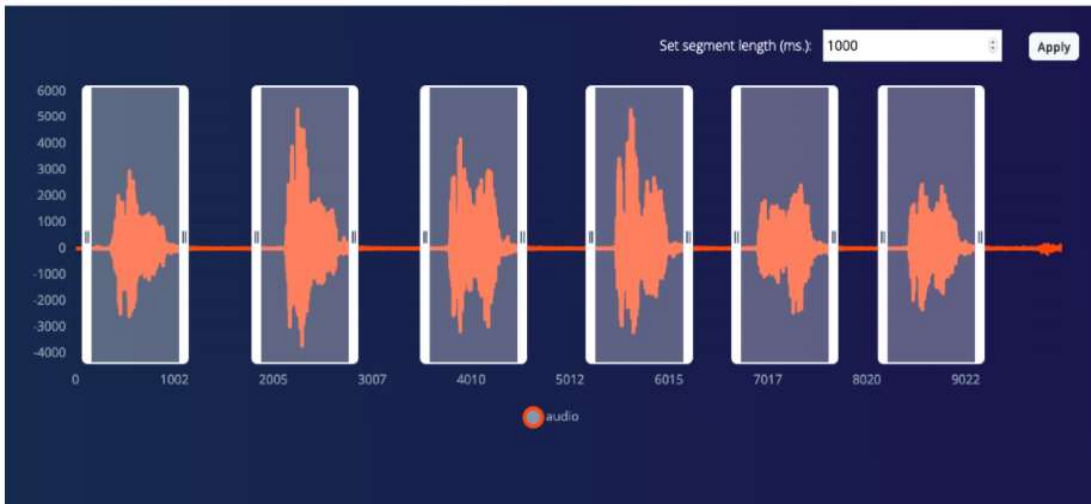


Figure 4.8: image showing split sample data

### Upload existing data

You can upload existing data to your project in the [Data Acquisition Format](#) (CBOR, JSON, CSV), or as WAV, JPG or PNG files.

Select files

104 files

Upload into category

Automatically split between training and testing [?](#)

Training

Testing

Label

Infer from filename [?](#)

Enter label:

Figure 4.9: image showing upload process

The next step was to collect other data samples that would be used to train the model. Three categories of data were used. The first category is the keyword dataset collected in the previous step, the noise dataset, and the unknown dataset. For the unknown set, 1 minute sample data was recorded and then split on the edge impulse platform. This was done 10 times to get a large data collection. For the noise dataset, the edge impulse noise sample data was used. The noise dataset was downloaded from the edge impulse website and uploaded to the platform.

After gathering all the necessary data, all the keyword data would be in training dataset. Some amount of the keyword data must be separated from the training data to validate the machine learning model. Rebalancing is performed on the data which automatically separates the data between training and testing data.

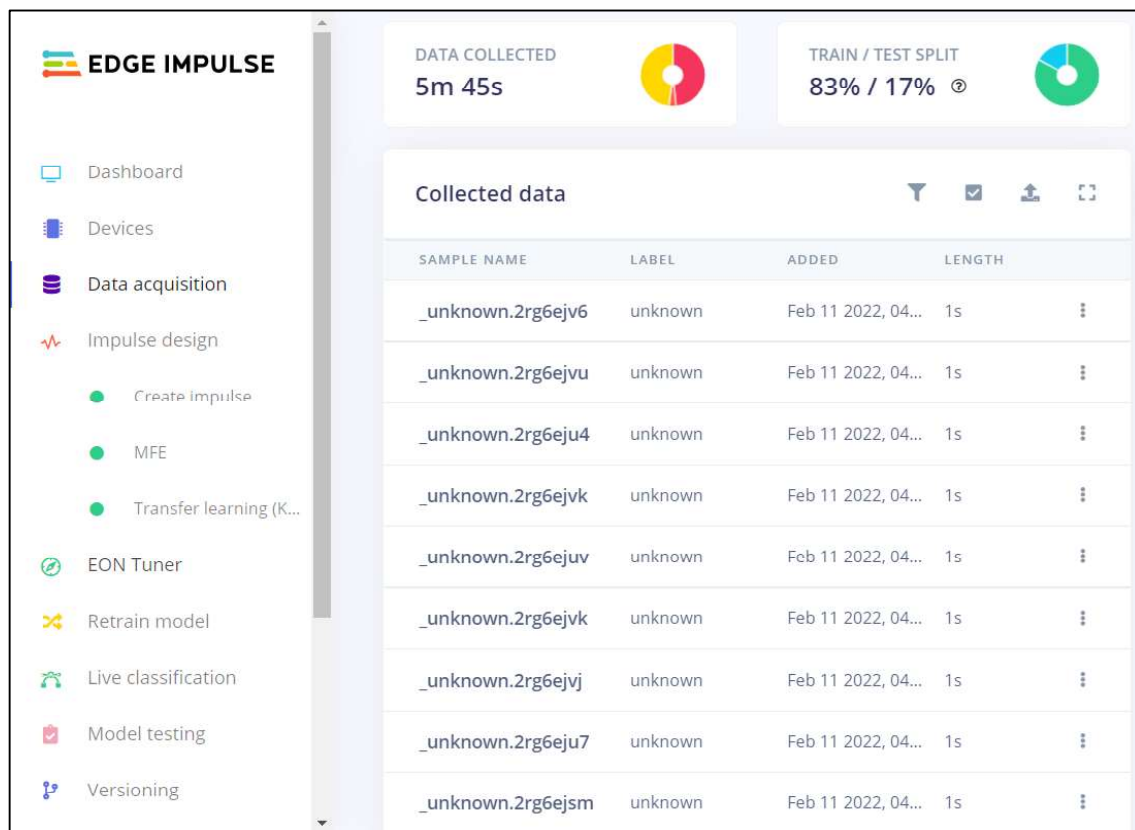


Figure 4.10: image showing the rebalanced dataset

The next step was to process the gathered data. The signal processing block was used to extract features from the dataset, and the learning block was then used to classify new data. The “MFCC” signal processing block was used for the feature extraction. MFCC stands for Mel Frequency Cepstral Coefficients. This block turns raw audio into a simplified version by removing redundant information. The simplified audio is then passed to the neural network block, which would carry out the learning process.

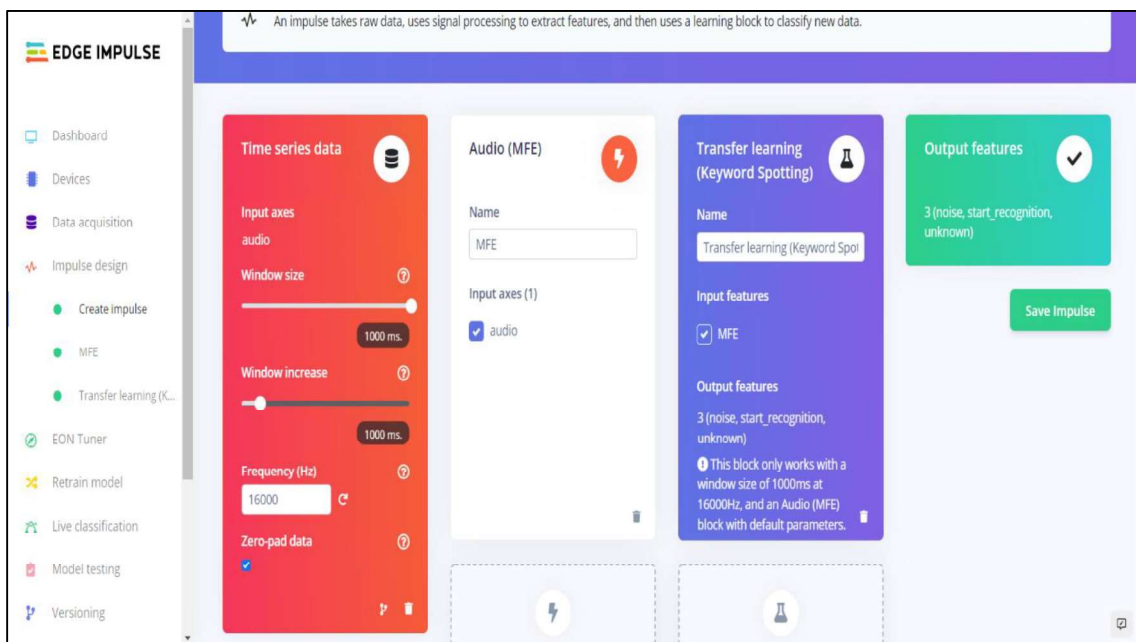


Figure 4.11: showing the training blocks

The parameters for the MFCC block were then set to obtain a spectrogram. The spectrogram would be used by the neural network for the identification of each word. Based on the parameters selected, results are shown, depicting how the program would perform on the board. The performance details can be seen in the image below.

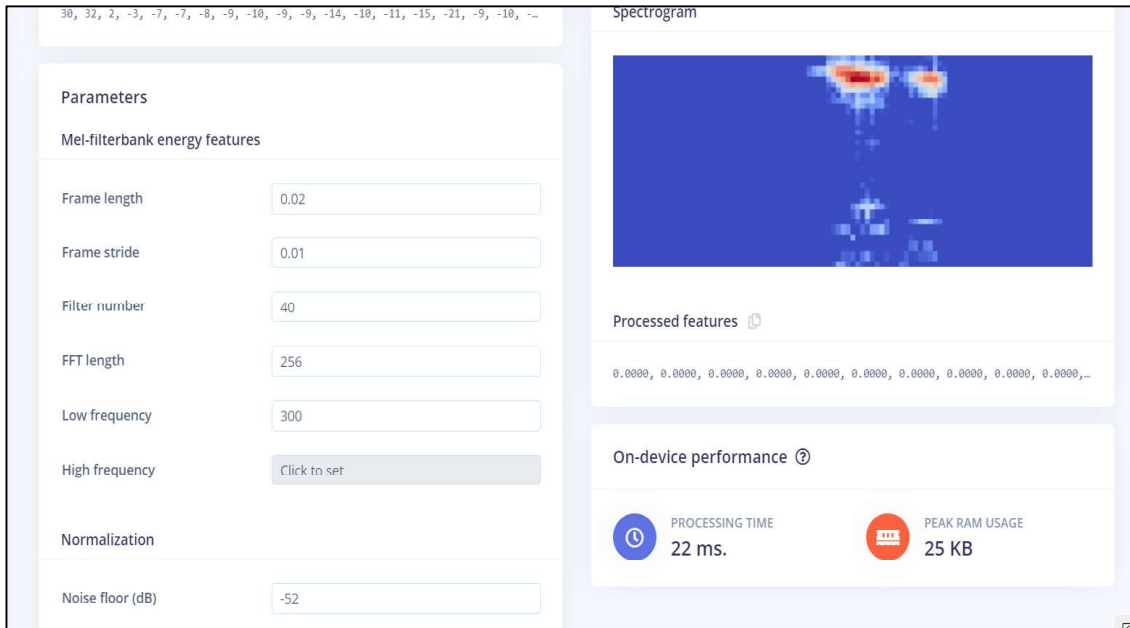


Figure 4.12: Image showing spectrogram and MFCC parameter

On the Neural network page, the parameters are set, and then the model is trained, and a performance panel shows the results after training.

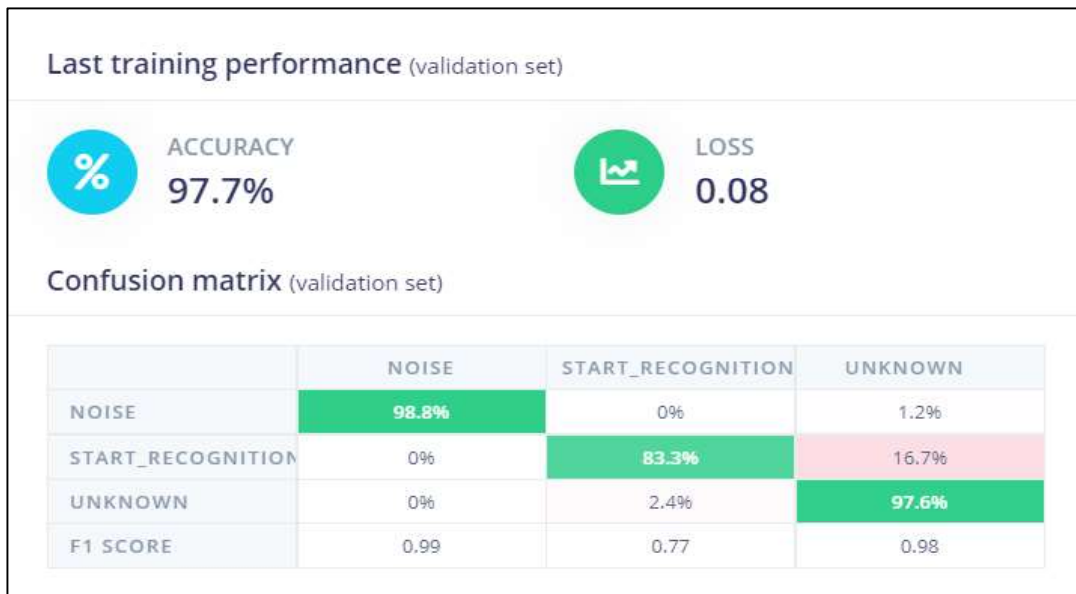


Figure 4.13: Image showing spectrogram and MFCC parameter

The confusion matrix shows the balance between correctly classified data and incorrectly classified data. An accuracy of 97.7% was obtained after training the model. For the noise data, 98% of the data was classified correctly. For the keyword, 83.3% were classified

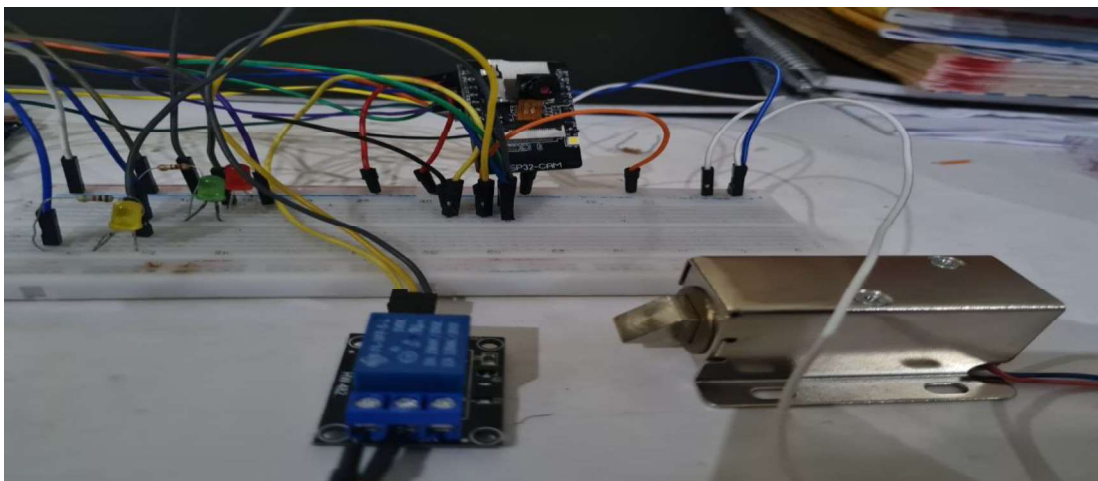
correctly, with 16.7% being wrongly classified, and for the unknown words, 97.6% were classified correctly. After training of the model was done, an Arduino library was generated using the deployment page on the edge impulse platform.

#### 4.4 Database and Application

The database used in this system is a non-relational database. A non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems [11]. Non-relational database makes it easy to send the data from the microcontroller to the database without any standard formatting or structure. These databases are claimed to perform better than SQL databases [12]. The Firebase platform was used to create a real time data base that would store the data. The rules of the database were set to True for both read and write to enable the microcontroller to send data to the database and read from it. An application was developed to interface with the system and keep the logs of when the system is accessed, who accessed it and if there was an intruder.

#### 5.5 Circuit

The circuit was built based on the design made in the previous chapter. The figure below shows the circuit



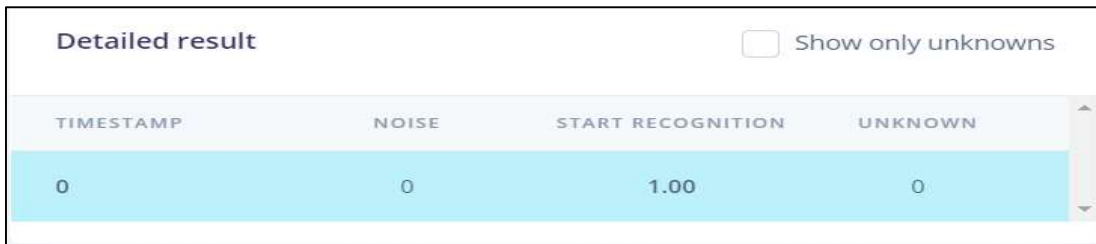
*Figure 4.14: Image showing the built circuit*

## Chapter 5: Results & Conclusion

### 5.1 Results

#### 5.1.1 Results for wake word system

The test was done on the live classification section of the edge impulse platform. Since the samples were split into test and training samples, the test samples that were not used in the training process is used to test the trained model. This data is unfiltered and unprocessed test audio of the keyword phrase. The results were as follows in *figure 5.1* and *figure 5.2* below.



Detailed result  Show only unknowns

TIMESTAMP	NOISE	START RECOGNITION	UNKNOWN
0	0	1.00	0

Figure 5.1: Image showing classification results

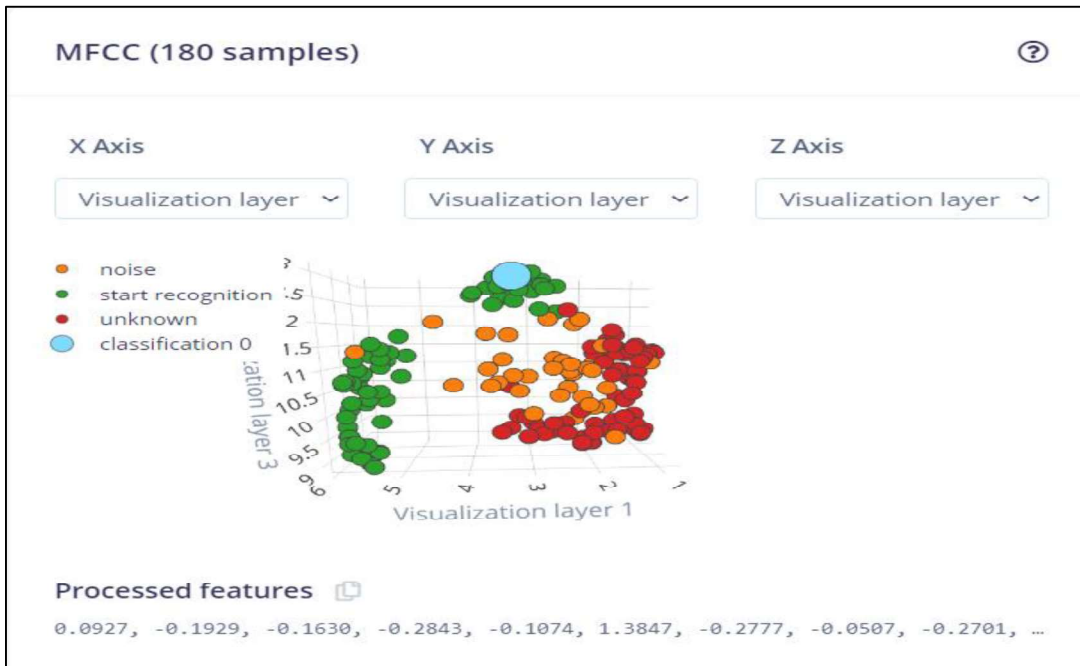



Figure 5.2: Image showing a 3d graph of the classified data

The values used to depict the accuracy of the classification are number between and including 0.00 and 1.00. A high accuracy of 1.00 was obtained from the sample test. A 3d graph shows a visual representation of all the data that has been classified and grouped based on the similarities of the features extracted. From the graph, the test sample can be seen classified amongst the keyword data samples, showing accurate data classification.

### 5.1.2 Results from main system

To determine if the system runs without any problems a series of test were carried out.

The system was first tested in a room that was properly lit to see if it functions without any issues. In testing the system, it was observed that there was a bit of a delay in detecting a face before the recognition process begins. In *figure 5.3*, even though a face was in front of the camera, it does not detect any face for about 4 seconds. Though the slight delay occurs, it finally detects and carries out the recognition process and unlocks the door since the face that was in front of the camera was recognized. The system was also unable to recognize a face sometimes even though the face was enrolled.



```
COM7
16:00:55.394 -> DETECTING
16:00:55.678 -> NO FACE DETECTED
16:00:55.912 -> NO FACE DETECTED
16:00:56.103 -> NO FACE DETECTED
16:00:56.326 -> NO FACE DETECTED
16:00:56.556 -> NO FACE DETECTED
16:00:56.792 -> NO FACE DETECTED
16:00:57.022 -> NO FACE DETECTED
16:00:57.255 -> NO FACE DETECTED
16:00:57.441 -> NO FACE DETECTED
16:00:57.671 -> NO FACE DETECTED
16:00:57.863 -> NO FACE DETECTED
16:00:58.600 -> Door Unlocked
16:01:01.993 -> Set string... ok
16:01:01.993 -> Time variables
```

The screenshot shows a serial terminal window titled 'COM7'. The main area contains a log of system messages with timestamps. The messages indicate a 4-second delay in face detection before the door is unlocked. The bottom of the window has control buttons for 'Autoscroll', 'Show timestamp', 'Newline', '115200 baud', and 'Clear output'.

Figure 5.3: Image showing classification results

After the recognition occurred and the door unlocked, the system successfully sent the data to the database.



Figure 5.1: Image showing classification results

## 5.2 Conclusion

This project aimed to design and build an access control system that gives or denies access to an enclosed space by using facial images with TinyML. This project was achieved to some extent. The system was able to detect and recognize faces that were enrolled on it and grant access to a room by unlocking the door. The use of TinyML made it possible to grant access using facial recognition without having to involve the use of cloud services. Issues such as latency, bandwidth limitations, network connection lags and poor connections that come up when using cloud services for the machine learning process was avoided. The system was able to perform its functions well. However, there were some limitations that could be addressed in the future.



### **5.3 Limitations**

Some of the limitations faced in this project are as follows:

- Acquiring the necessary materials and resources was difficult as some of the components needed to build some parts of the project were not available.
- The resources provided by machine learning platforms were limited for microcontrollers making it difficult to train a facial recognition model from scratch and deploy it on a microcontroller.
- The hardware resources that are contained in microcontrollers are not enough to run a robust machine learning algorithm.

### **5.4 Future work**

Building on the wake word system, the Arduino nano 33 sense can be acquired and add to the system to run the speech recognition model generated. The system can act as a two-step verification method, meaning that only those who know the key word can access the main system for the recognition process to begin.

With the possible availability of more resources for facial recognition on machine learning platforms and framework, a more robust machine learning model can be developed for the system.

## References

- [1] S. Naveen, "Study of IoT: Understanding IoT Architecture, Applications, Issues and Challenges", Researchgate, 2016. [Online]. Available: [https://www.researchgate.net/publication/330501274\\_Study\\_of\\_IoT\\_Understanding\\_IoT\\_Architecture\\_Applications\\_Issues\\_and\\_Challenges](https://www.researchgate.net/publication/330501274_Study_of_IoT_Understanding_IoT_Architecture_Applications_Issues_and_Challenges). [Accessed: 25- Apr- 2022].
- [2] N. K. Giang, R. Lea, M. Blackstock and V. C. M. Leung, "Fog at the Edge: Experiences Building an Edge Computing Platform," 2018 IEEE International Conference on Edge Computing (EDGE), 2018, pp. 9-16, doi: 10.1109/EDGE.2018.00009.
- [3] S. J. Bigelow, "What Is Edge Computing? Everything You Need to Know", SearchDataCenter, 2021. [Online]. Available: <https://www.techtarget.com/searchdatacenter/definition/edge-computing>. [Accessed: 25- Apr- 2022].
- [4] J. Hertz, "What Is TinyML?", allaboutcircuits, 2022. [Online]. Available: <https://www.allaboutcircuits.com/technical-articles/what-is-tinyml>. [Accessed: 27- Apr- 2022].
- [5] D. Dutta and S. Bharali, "TinyML Meets IoT: A Comprehensive Survey", Internet of Things, vol. 16, p. 100461, 2021. Available: 10.1016/j.iot.2021.100461.
- [6] S. Jahnavi and C. Nandini, "Smart Anti-Theft Door locking System," 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), 2019, pp. 205-208, doi: 10.1109/ICATIECE45860.2019.9063836.

- [7] Farhat Lamia Barsha, SanzidaMoj ib ZarinTasneem and Masuda Afrin, "An IoTbased Automated Door Accessing System for Visually Impaired People", *5th IEEE international Women in Engineering (WIE) Conference on Electrical and Computer Engineering (IEEE WIECON - ECE)*, 2019.
- [8] Guha, A. Chakrabarti, S. Biswas, and S. Banerjee, "Implementation of face recognition algorithm on a mobile single board computer for IOT Applications," 2020 IEEE 17th India Council International Conference (INDICON), 2020.
- [9] A. Basit, K. Saxena and A. Rana, "A Wearable Device used for Smart Doorbell in Home Automation System," 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE), 2020, pp. 90-93, doi: 10.1109/WIECON-ECE52138.2020.9398001.
- [10] J. Xiang and G. Zhu, "Joint Face Detection and Facial Expression Recognition with MTCNN," 2017 4th International Conference on Information Science and Control Engineering (ICISCE), 2017, pp. 424-427, doi: 10.1109/ICISCE.2017.95.
- [11] "Non-relational data and NoSQL - Azure Architecture Center", *Docs.microsoft.com*, 2022. [Online]. Available: <https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/non-relational-data>. [Accessed: 27- Apr- 2022].
- [12] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), 2013, pp. 15-19, doi: 10.1109/PACRIM.2013.6625441.
- [13] C. -L. Zhong, Z. Zhu and R. -G. Huang, "Study on the IOT Architecture and Gateway Technology," 2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), 2015, pp. 196-199, doi: 10.1109/DCABES.2015.56.