ZLVZJPXNBBFG5JLM

Arduino security system

UNIVERSITY OF
LINCOLN

Jamie Godwin
GOD18675053
18675053@students.lincoln.ac.uk
School of Computer Science
College of Science
University of Lincoln

Submitted in partial fulfilment of the requirements for the Degree of BSc(Hons) Computer Science

Supervisor: Dr. Mubeen Ghafoor

May 2021

Abstract

This project was the creation of a prototype security system using step motors to control door locks and an Arduino to control the motors. The security system has two modes of facial recognition, which requires a user dataset to be added and given a corresponding number. These numbers work as roles and will be the output of the facial recogniser program once a user's face has been detected via a video stream; this output will be sent to the Arduino. If the user has the correct output it will unlock a door, this works as a facial ID for a system. The other section works via a user entering a passcode which is meant to be used as a supporting system if the main facial recognizer stops working. When the passcode is entered correctly a serial code will be sent to the Arduino which will cause it to unlock. This system can be duplicated to work on more doors.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main aim of this project is to produce a fully functional security system which facilitates the use of an independent locking mechanism. This will be done with a facial ID or a passcode entered by a user. For this to work a dataset is required and trained to differentiate between both users and nonusers of the system. Not all users should have access to certain areas making it important that different users can be distinguished so different doors can have different security standards working as a form of role implementation. It is also important that once the door is unlocked, it will lock after some time automatically.

For this project to work, these seven goals must be completed in order. The assembly of the hardware into the correct locations so that they are functional. After, the hardware is required to be programmed to act like a door lock while receiving serial commands. Then a video stream will be required for the ability to see faces to unlock doors. Next is the ability to find a face inside of the video stream which will be followed by data collection of the system's user faces. Then this data will be trained and finally used for facial recognition.

Chapter 2

Literature Review

2.1 Background

Over the past few years, huge leaps have been made in the area of computer vision and machine learning. Many people say we are on the cusp of a new technological era with many new homes and businesses having integrated computer systems in the realm of the IoT (internet of things) from smart fridges to fully automatic security systems. It's integral to the project that basic security is considered during the planning phase.

This new age of development has computer vision and machine learning leading the charge into the unknown as they are both leading areas of research and development. This project will require the use of both and the integration of hardware in the form of an Arduino. Arduinos have a large link to the IoT with many projects using them, including this one, not all IoT devices are required to connect via the internet however they do require interconnectivity between devices such as in this project.

2.2 Related Literature

Computer vision is growing at an incredible pace with new advancements made every year. An integral part of this has been the over 2500 optimized algorithms used for state-of-the-art computer vision and machine learning. This project focuses on the systems used for facial recognition. A conference about the use of face detection based on OpenCV (Fan, Zhang, Wang and Lu, 2012) discusses a large number of applications that this sort of software will have along with the large market value. This demonstrated the great demand for such software with integrated systems, which this project will be an example of. They also discuss the improvements that can be made to the base OpenCV library by using a modified algorithm as well as the use of dual-thread which makes detection more precise. A later project prototype might implement these changes to increase accuracy.

OpenCV is one of the best open-source libraries for facial recognition available. Shervin Emami and Valentin Suciu (2021) show how they use OpenCV to recognize individual faces using computer vision and image analysis inside of the OpenCV library, this will be similar to how this project hopes to introduce facial recognition. Shervin Emami and

Valentin Suciu (2021) Discusses how the algorithms work by measuring positions on multiple different individual faces and recording the results of facial features such as the position of eyes and size of eyes. This data is all compiled together to give a numerical description of user's faces from the data allowing for comparison between them.

Many people have concerns about the use of facial recognition, especially in certain use cases such as described by Andrejevic and Selwyn (2019), who go into detail about the use case of software like this inside compulsory education, for many of the same reasons this project is being developed. This would be implemented in this setting primarily for security concerns and monitoring user attendance. Comparisons have been made to "authoritarian" ways of security as many of the places in which this type of system has been tested and implemented by "extensive cultures of monitoring and surveillance". This makes a very valid point about the use of this project and where it is appropriate to use, such as in a home security system or that of a corporation. This can be a grey area as these cameras may be facing into a school or other private spaces. As stated Andrejevic and Selwyn (2019), people's behaviour changes when they know they are being watched and this could have a large effect on society as a whole. This must be taken into account when developing the system which is why only user facial data will be stored and no facial data will be recorded by the project to help lower potential concerns.

The increase in security that facial recognition offers is very hard to ignore. Karovaliya, Karedia, Oza and Kalbande (2015) show the amount of crime that can be prevented by the addition of facial recognition on ATMs. The main purpose is to eliminate fraud due to stolen cards, due to the potential identification of users' faces. This also negates user error from forgetting passcodes for the system as the primary use of the system is facial recognition, meaning passcodes are not as necessary. They also draw a comparison between the cost of other security systems such as biometric iris scanners, this helps to clarify the effectiveness of facial ID and the cost-effectiveness of it. This also helps to point out some of the potential flaws with face ID systems, such as users with glasses and beards lowering the accuracy of such systems.

During data collection for machine learning, it is important to consider the quality of data that is being collected. Wilman Zou and Pong Yuen (2012) discuss the problems of low-resolution surveillance cameras that have facial recognition running on them. Any camera with low resolution will not give a satisfactory performance when running facial recognition software, meaning the higher the quality of the camera the better the results will be. This also includes the amount of data collected, meaning the more data, the more accurate results. From this dataset, collection can also be affected if low-resolution data is used as this will affect the program's ability to recognize a user face from a nonuser or someone who isn't meant to have access to a set area. To avoid these pitfalls, all data that will be uploaded to my project and the camera used will be high resolution.

Not only is the quality of data important but the quantity of data is also critical. Tran, Mayhew, Kim, Karande and Kaplan (2018) are using a massive but out of context dataset of human faces to recognise human emotions. This data is used alongside a much smaller and more relevant dataset that is used to distinguish emotions in the larger set, thus making the project more accurate as the larger datasets improve the trained model. This can be adapted to show that larger datasets, in conjunction with high-quality data, will improve the accuracy of any machine learning.

Smart door locks described by Jeong (2016) have the ability to lock/unlock a door without the use of a traditional method. Typically a remote method, such as a mobile device, is used to toggle the door. The idea is to have a more protected system than a simple key, which can be lost or the lock itself can be picked. Smart locks have fewer points of failure from users though require more software defences, such as encryption. For larger corporations, smart locks will often save money due to not needing replacement keys and the resulting higher security leads to fewer breaches of security. However smart locks allow for one point of failure which might also result in larger damage such as locks being hacked causing users to be locked into rooms. Even though these are valid, the market share of IoT security systems is increasing every year, suggesting this is a valuable and growing market.

Precise movements are a requirement of step motors to lock and unlock doors without causing any damage to the door frame or lock itself. Le, Cho, Jeon, (2006) discuss the high precision movements a step motor is capable of. The ability to control the velocity of a motor at any time during its movements allows for precision programming to fit many cases requirements. Le, Cho, Jeon (2006) explain how the motors are capable of providing a smooth motion to continue operations. Due to these motors only running in closed-loop systems, motors can lose synchronisation with one another when working in parallel, this problem can be avoided with the use of separated systems such as in this use case as each motor will be receiving specific data for its use case. This also allows for calculations to be made for the precise amount of steps required by the step motor to lock and unlock a door, also setting the speed of rotation in rotation per minute to make sure enough time will pass before the door begins to lock.

For this project, it is vital we understand the basic Arduino board and its ability to send and receive serial communications as well as knowing how to safely configure the circuit. Leo Louis (2016) provides information about each Arduino board, including clock speeds and the microcontroller accompanying the Arduino board. Leo Louis (2016) also offers many of the use cases for each board, discussing which board is better suited for different scenarios. The basic model Arduino is described with all of the parts and what each is used for allowing for a greater understanding of how the boards work. This is accompanied by some basic code that helps to get the hardware running for proof of concept and some simple project designs to help teach the integral parts of Arduino programming. Leo Louis (2016) makes compelling points about the core reasons for use of Arduino boards, the most relevant being that they are multi-platform allowing use on multiple devices such as Windows and MacOS, have an active user community and are generally inexpensive due to low hardware cost and lots of open source libraries.

Chapter 3

Methodology

3.1 Project Management

Before the start of project management, a choice of which methodology to use must be made. The most suited for this project being Agile management and Waterfall management so both will be compared.

Management Traits	**Agile**	**Waterfall**	**Choice for this Project**
Project scope	Changes can be made through the process of developed cycles as the project is broken down into single development cycles which are referred to as sprites .	Each implementation is developed linearly allowing for the project to be split into larger development stages.	Waterfall
Stages of development	Each stage is done iteratively allowing for moving forward and backwards between each stage when needed.	Is linear with each stage requiring the previous stage to be completed.	Waterfall

Schedule	Updates to Schedule based on feedback from implementation.	Set time based on a defined scope of project.	Waterfall
Progression measurements	Very little documentation, progress is measured against the MVP (minimum viable product).	Each stage will be measured against the time given in the plan. Giving clear milestones in the documentation.	Waterfall
Primary goal	Creation of a MVP that is then iterated for improvements to the product.	Completion of the product.	Waterfall
Working product	Available in early stages of the project.	Only available at the end of the project cycle.	Agile

Table 1:Methodology choice table

The Waterfall methodology was chosen as it follows a strict linear method that would allow me to implement the seven stages of this project. It is also one of the easier methods to follow as each stage must be completed before the next task, meaning the next step will always be clear. This will prevent most cases of needing to go back on previous work to implement a feature as most stages won't work without the previous stages being fully completed, allowing for a clear set of goals that must be achieved alongside with a set of requirements for the project end point.

As the project has a very clearly defined goal it allows for the best case usage of the Waterfall method. This also allows for analysis of each step before development to catch any design or order of implementation errors. Waterfall also has a clear set of milestones which allow for progress to be charted and followed very easily allowing for a timeline to be followed. One of the problems with this methodology is that if a requirement is missed in an earlier stage it can be very hard to find the correct implementation of a new feature, however this can be mitigated via a well structured plan, such as the seven stages as it would be noticeable if anything was missing. Waterfall also often has problems with deadlines as certain aspects of implementation might take longer than expected. The chances of this happening have been lowered with the use of a Gantt chart from the project proposal which allowed for setbacks on each stage of development.

3.2 Software Development

As discussed above, the chosen software development cycle used for this project was the Waterfall model due to this project having very easily definable goals that needed to be accomplished before the next stage of development, each stage is required to be fully functional before the addition of the next step.

The only exception to the Waterfall model was the analysis of the project requirements from the beginning as each stage of development was very clear and required very little additional thinking. This is mainly due to the requirements of the system being completed right at the beginning of the project. This was because of all the hardware requirements, meaning that the project had to be fully planned out before the hardware arrived. Due to this, the requirements of the project were very clear and broken down into seven distinct steps that then each followed the Waterfall cycle without the full use of the requirement step. These were previously stated in the introduction.

Implementation followed the seven steps that were laid out, with considerations for system design at each step such as the component layout and user interface. Once each stage was implemented, it was required to be tested to make sure that the implementation was functional. If this step was not completed, the project could not progress as each previous step is vital for the next stage of development. Once testing was complete, the project was deployed and any changes to previous steps that were required by future stages were made during the next cycle of the Waterfall model.

3.3 Toolsets and Machine Environments

Arduino Integrated Development Environment (IDE) was used as it is cross platform, allowing it to be developed and used on any common machine type. This ensures a wider compatibility. The application itself is written in a programming language with a large similarity to C and C++ meaning lots of supporting documentation for any problems that may be run into. The IDE also comes with lots of supporting code libraries filled with project ideas and common code implementation, as well as a built in uploader with settings designed specifically for the uploading of code onto an Arduino. This can be done using other methods such as the use of Python or PlatformIO however they are generally used for much wider scope projects that require the use of web controls or mobile adaptations. They also have less specifically designed interfaces for the validation and uploading of code onto an Arduino. Therefore, the IDE was chosen for all code running on the Arduino.

OpenCV was the library chosen for both facial detection and dataset training. The main reason for this choice was its large library and focus on computer vision which is specifically designed for human facial detection, as well as for machine learning to identify faces using collected data. OpenCV also has a large community with lots of explanations of how certain aspects of the algorithms work, allowing for ease of use and debugging. The image processing can also be done in real time, allowing for it to be connected to a video stream. Meaning that the system does not have to take a photo and then process that, which makes it much faster than other alternatives. The main alternative being Tensorflow which has a differentiation package that can be implemented on machine learning to distinguish between faces. This sort of machine learning is Deep Learning which is not required for this project.

Selection of a programming language for the development of the project was heavily decided by the use of OpenCV and which language worked best with it.

Features of language	**C++**	**C#**	**Python**
OpenCV support	Yes	Yes	Yes
Library support	Medium	Medium	High
Open Source	No	Some implementation	Fully
Compatibility	Write once, compile anywhere	Write once, compile anywhere	Write once, run anywhere

Table 2: Language comparison between C++, C# and Python

As evidenced via Table 2 the best option for the programming language is Python. This was justified as more Library support will allow for easier additions of features as well as being open source. The final reason was the ability to write once and run anywhere meaning less compiling is required, making it faster to run on different systems.

Chapter 4

Design, Development and Evaluation

4. Software Development Projects

1. Requirements elicitation, gathering, collection and analysis

The first set of requirements for this project includes all the hardware used for each project aspect, as well as the main library in use with the data required to train the dataset.

An ELEGOO UNO R3 board was used as the basis. This board was connected to a PC using a USB to B connector that allows for code to be uploaded to the board. This board was chosen as it has more than enough storage capacity (32KB) for the program that it will be running, allowing for addition to the program in the future. The clock speed of the board (16Mhz) is much more processing power than required for the program allowing for the expansion or the addition of computations. For instance, the control of multiple motors operating different door locks. For the prototype, the chip's voltage will also be sufficient with 7-12V of input voltage only requiring 5V per step motor in the circuit. This can be improved with the addition of power relays to separate the circuit, allowing for each motor to be controlled separately via the same power source. For the prototype, the amount of pins on the board itself is enough as only 6 pins are required. However, this can be increased with the use of a breadboard for more expansion of the system; the board itself has 14 digital I/O (Input/output) pins and 6 PWM (Pulse Width Modulation) outputs.

The step motor 28BYJ-48 is used to demonstrate how a door would be locked/unlocked using serial inputs from a Python program, which are then read by the programming running on the UNO R3. This motor has five pins, four used for control of the coils and one for voltage(red wire).

For this motor to function it requires a ULN2003 stepper motor driver. This is used as the interface between the UNO R3 and the stepper motor. This interface is also used to confirm the motor is functional via an LED display.

Dericam Webcam is required for facial recognition which can be done with any HD webcam that can run with at least 30 FPS (frames per second). This is necessary to make the recognition software more accurate and distinguish between users and non-users more swiftly. The webcam used does not require colour as the frames will be converted to grayscale via the Python program.

Power must also be provided, each Digital I/O pin has a maximum of 40 mA(milliamps) 4 of these pins are in use as well as 2 power pins which are 50mA, each pin is * by the voltage giving 1.3W(watts) required per step motor . For the basic prototype only 5V is required. Power can be accessed by: using a PC connected to the UNO R3; connecting a battery pack to the Arduino and linking the webcam via Bluetooth; or cabling onto a breadboard that is connected to the Arduino, instead of the webcam being linked via a PC.

OpenCV is an open source computer vision library. This is required for the image processing side of the project as it will be used to recognise certain faces. This is done using the significant geometric features of a certain face such as eye, nose, and mouth location. The file in use will be haarcascade_frontalface_default.xml (OpenCV 2013).

Other more common libraries were also used, these being: Pickle to convert a Python object into a byte stream which will be used by the trainer; Serial is used to send serial commands to the Arduino which allow for motor function; time is used for the ability to add simple delays; Readchar will be used for the ability to read user inputs; Datetime will be used for the collection of data of the date and time of when a user interacts with the system; OS is used for the ability to read and write files; and finally NumPy will be used for array processing.

Each user will be required to upload at least 20 photos of their face from different angles and different facial expressions on a blank background. This will be used in conjunction with haarcascade_frontalface_default.xml to train the system to recognise individual users that have uploaded data, Making this a required to be done before the program is running. At least one user must upload data to allow for any non-users to not be recognised by the system.

Wires are required to connect each component. For the basic prototype, six male to female wires are used to connect the motor to the UNO R3. A USB 2 to B wire is needed to connect the UNO R3 to a PC for code uploading as well as a webcam connection.

1. **Design**

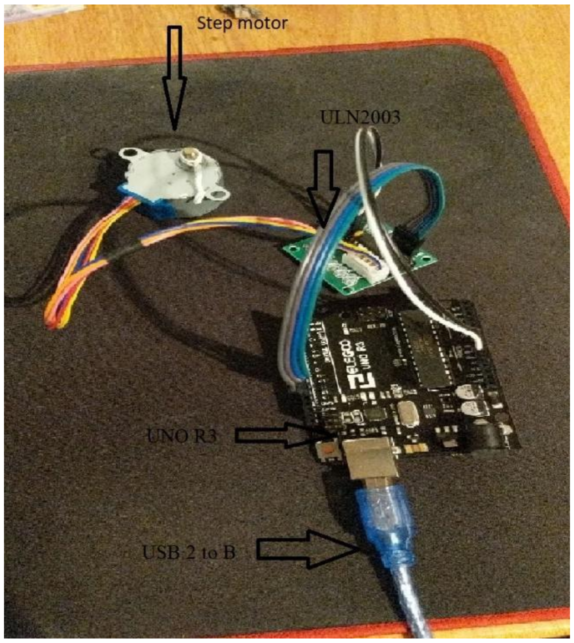


Figure 1: Prototype layout

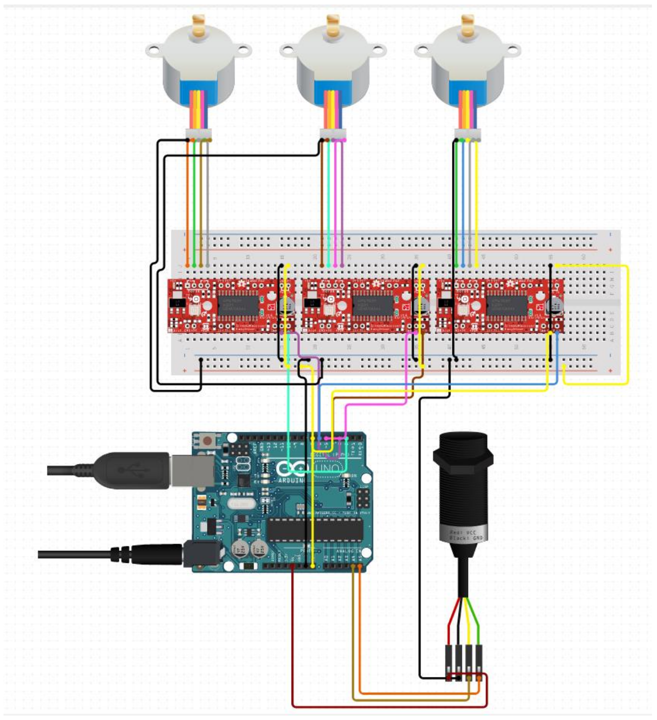


Figure 2: Prototype expansion made using(Circuito.io. 2021)

Figure 1 shows the prototype layout of the system with the webcam being attached via USB 2 to B. this design can be easily expanded with the use of a breadboard allowing for more step motors functioning as door locks shown in Figure 2. Figure 2 also shows how the circuit can have the camera integrated with the rest of the board. In a real situation, wire length would be increased as well as a functional lock bar would be added which would be moved by the step motors.

The software design will be an interface asking a user which login method they would like to use, either password or facial ID. After the choice is made the users will enter a passcode to unlock a door or facial ID will be used.

In this system, the passcode is entered on the PC that is running the Python code. However, this can be easily adapted to a Numpad that could be connected to an Arduino allowing for both to run simultaneously. This means the failure of face ID would not be a large problem if the password is known.

In the prototype, both face ID and passcode are options after starting up allowing for user choice. After a password has been entered wrong five times in a row the program will end working as a safeguard. Facial recognition will continue until the rest key is pressed which is labelled “R”. This was done so that a password would be used as a backup rather than the main method.

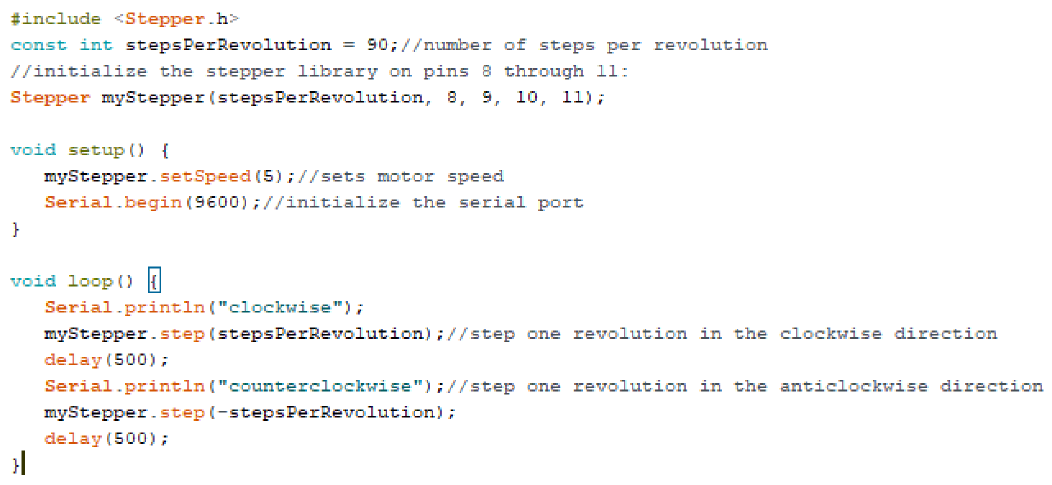
Once a face or password has been correctly identified, the motor will do a full turn simulating a door becoming unlocked. Six seconds will pass, giving users more than enough time to enter a door and close it before the door will lock again to secure the room.

When facial recognition is running, anytime a face is recognised, user or non-user, a 10-second delay was added to prevent overloading the Arduino with commands as they will begin to stack up if a face is recognised. The implementation of a photo each time a face was detected was not used as warned by Andrejevic and Selwyn (2019) as the locations in which this project may be used are unknown making it a safety precaution not to add this.

Users can be added by uploading at least 20 photos of their face which is used to train the program. With the addition of new users, different numbers must be assigned which will work as roles, stopping certain people from having access to different rooms. The name of the user is recognised and the current date and time is recorded which will allow admins to see who went into each room at what time. This will also allow them to check when a lock was attempted to be accessed by anyone. This was added so that any time equipment is lost or damaged admins will be able to see who was in the room and what time they entered, facilitating easier investigation.

As this project will mostly be used by admins, the user interface was not a top priority making it a simple text based interface with multiple options that will contain exception handling to help prevent user error on the system. This exception handling will not allow the use of any inputs that are not recognised and will inform the user of any mistakes made.

2. Building and programming

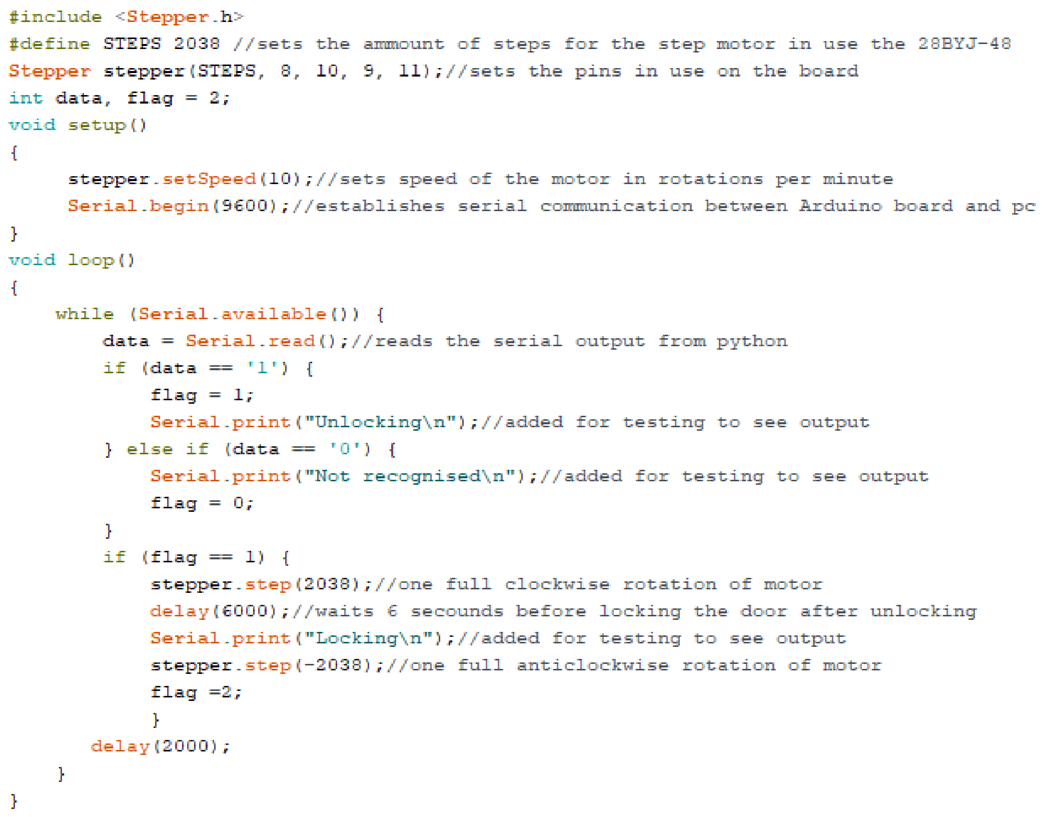

```
#include <Stepper.h>
const int stepsPerRevolution = 90;//number of steps per revolution
//initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
   myStepper.setSpeed(5);//sets motor speed
   Serial.begin(9600);//initialize the serial port
}

void loop() {
   Serial.println("clockwise");
   myStepper.step(stepsPerRevolution);//step one revolution in the clockwise direction
   delay(500);
   Serial.println("counterclockwise");//step one revolution in the anticlockwise direction
   myStepper.step(-stepsPerRevolution);
   delay(500);
}
```

Figure 3: Step motor test code

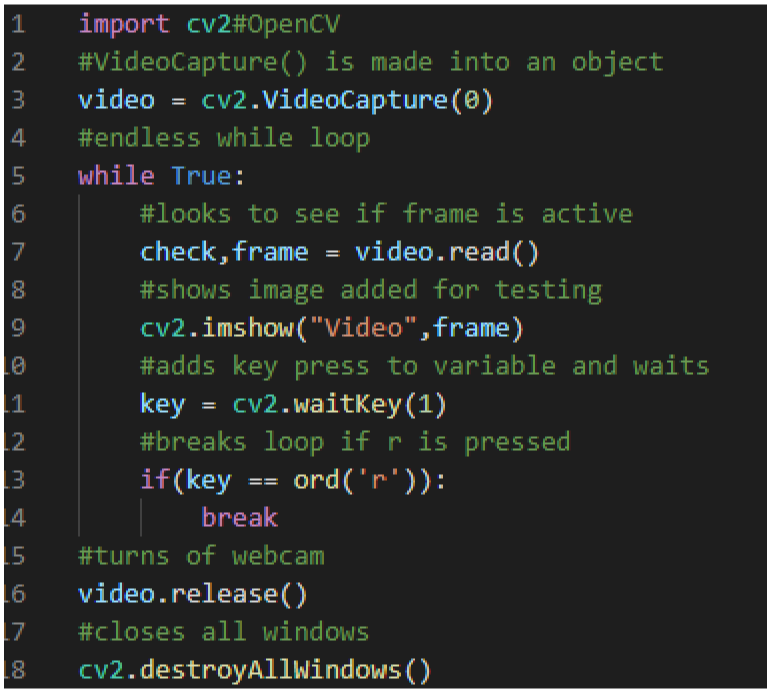
The first building stage was wiring up the UNO R3 and using a basic program to confirm the step motor was functional. This was done using C++ in the Arduino editor as shown in Figure 3. This proved that all hardware required for the locking mechanism was fully functional, excluding a webcam.



```
#include <Stepper.h>
#define STEPS 2038 //sets the ammount of steps for the step motor in use the 28BYJ-48
Stepper stepper(STEPS, 8, 10, 9, 11);//sets the pins in use on the board
int data, flag = 2;
void setup()
{
     stepper.setSpeed(10);//sets speed of the motor in rotations per minute
     Serial.begin(9600);//establishes serial communication between Arduino board and pc
}
void loop()
{
    while (Serial.available()) {
        data = Serial.read();//reads the serial output from python
        if (data == '1') {
            flag = 1;
            Serial.print("Unlocking\n");//added for testing to see output
        } else if (data == '0') {
            Serial.print("Not recognised\n");//added for testing to see output
            flag = 0;
        }
        if (flag == 1) {
            stepper.step(2038);//one full clockwise rotation of motor
            delay(6000);//waits 6 secounds before locking the door after unlocking
            Serial.print("Locking\n");//added for testing to see output
            stepper.step(-2038);//one full anticlockwise rotation of motor
            flag =2;
            }
       delay(2000);
    }
}
```

Figure 4: Full Arduino code

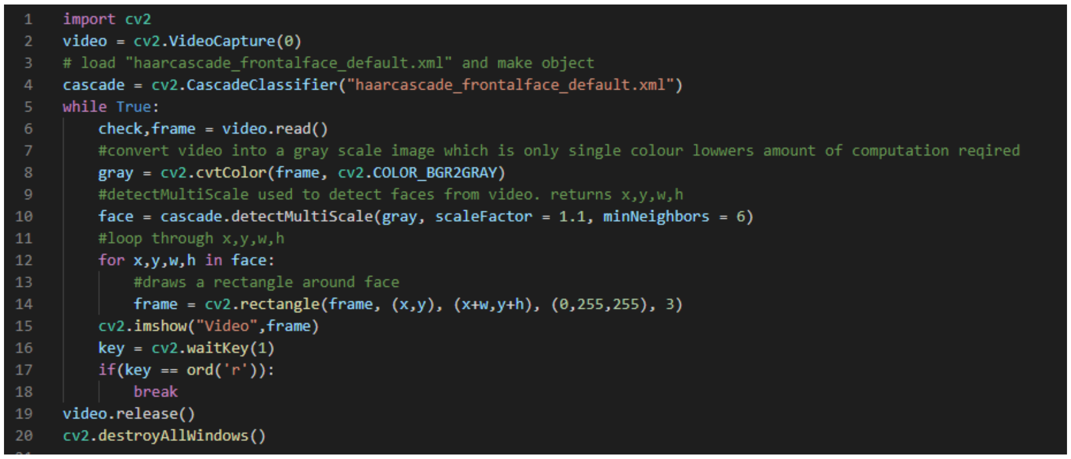
Next was the development of the locking mechanism using C++ in the Arduino editor shown in Figure 4. Using the Arduino editor allows for the use of serial manger, a great testing tool. This section of code works by receiving a serial input of a 1 or 0 to move the step motor. This allows for any other code to output serial outputs to influence the hardware for a desired result e.g. locking and unlocking a door. This code is uploaded to the UNO R3 and can be run without the use of a PC allowing for a wireless connection. Using serial manager lets me send a signal as if a password had been entered correctly as it will have a direct effect on the board or by having the output displayed in the serial manger window in the case of "Not recognised". In the final product the step motor's movement will have to be precise like described in Le, Cho, Jeon (2006) and a mechanism for the motor lock will have to be created. meaning that calculations of how many steps the lock must do to actually lock will be required. This might also require the speed setting to be changed for a smooth locking motion.



```python
import cv2#OpenCV
#VideoCapture() is made into an object
video = cv2.VideoCapture(0)
#endless while loop
while True:
    #looks to see if frame is active
    check,frame = video.read()
    #shows image added for testing
    cv2.imshow("Video",frame)
    #adds key press to variable and waits
    key = cv2.waitKey(1)
    #breaks loop if r is pressed
    if(key == ord('r')):
        break
#turns of webcam
video.release()
#closes all windows
cv2.destroyAllWindows()
```

Figure 15: Gaining access to webcam

The first step in facial recognition part of the project was accessing the webcam. This was done by using Python and the OpenCV library. OpenCV was created to provide a universal infrastructure to be used in computer vision applications. Figure 15 shows the Python code used for this stage. This works as the webcam is connected to a PC instead of an Arduino.This code will be the backbone of facial recognition as all the next steps require the usage of the video stream to be analysed.This program will run continuously until the "R" key is pressed which works as a system reset key, this allows for the security system to running constantly.



```python
import cv2
video = cv2.VideoCapture(0)
# load "haarcascade_frontalface_default.xml" and make object
cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
while True:
    check,frame = video.read()
    #convert video into a gray scale image which is only single colour lowwers amount of computation reqired
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #detectMultiScale used to detect faces from video. returns x,y,w,h
    face = cascade.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 6)
    #loop through x,y,w,h
    for x,y,w,h in face:
        #draws a rectangle around face
        frame = cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,255), 3)
    cv2.imshow("Video",frame)
    key = cv2.waitKey(1)
    if(key == ord('r')):
        break
video.release()
cv2.destroyAllWindows()
```

Figure 5: Identifying faces

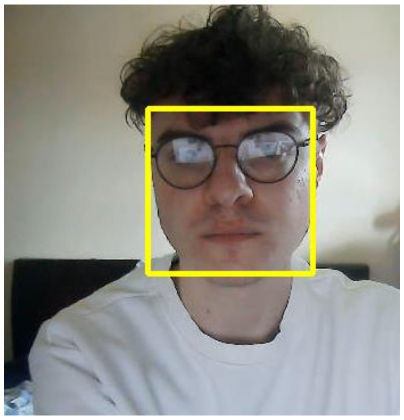


Figure 6: Face detection

Next step in facial recognition is the ability to find a face in the video stream from the webcam shown in Figure 5. This section requires the use of a cascade classifier (OpenCV 2013) which contains pre-trained models for face detection and will be used throughout this project. The video stream from the webcam is converted into greyscale to lower the computing power required. Once a face is detected, the program will draw a rectangle around the detected area to show what is being recognised and what is not - a valuable tool for testing. Once this code is run, shown in Figure 6, it can find faces from the camera stream and create a rectangle around the recognised area.

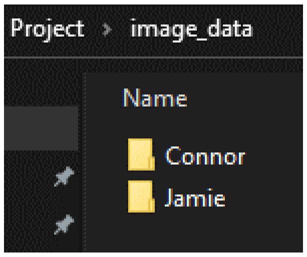
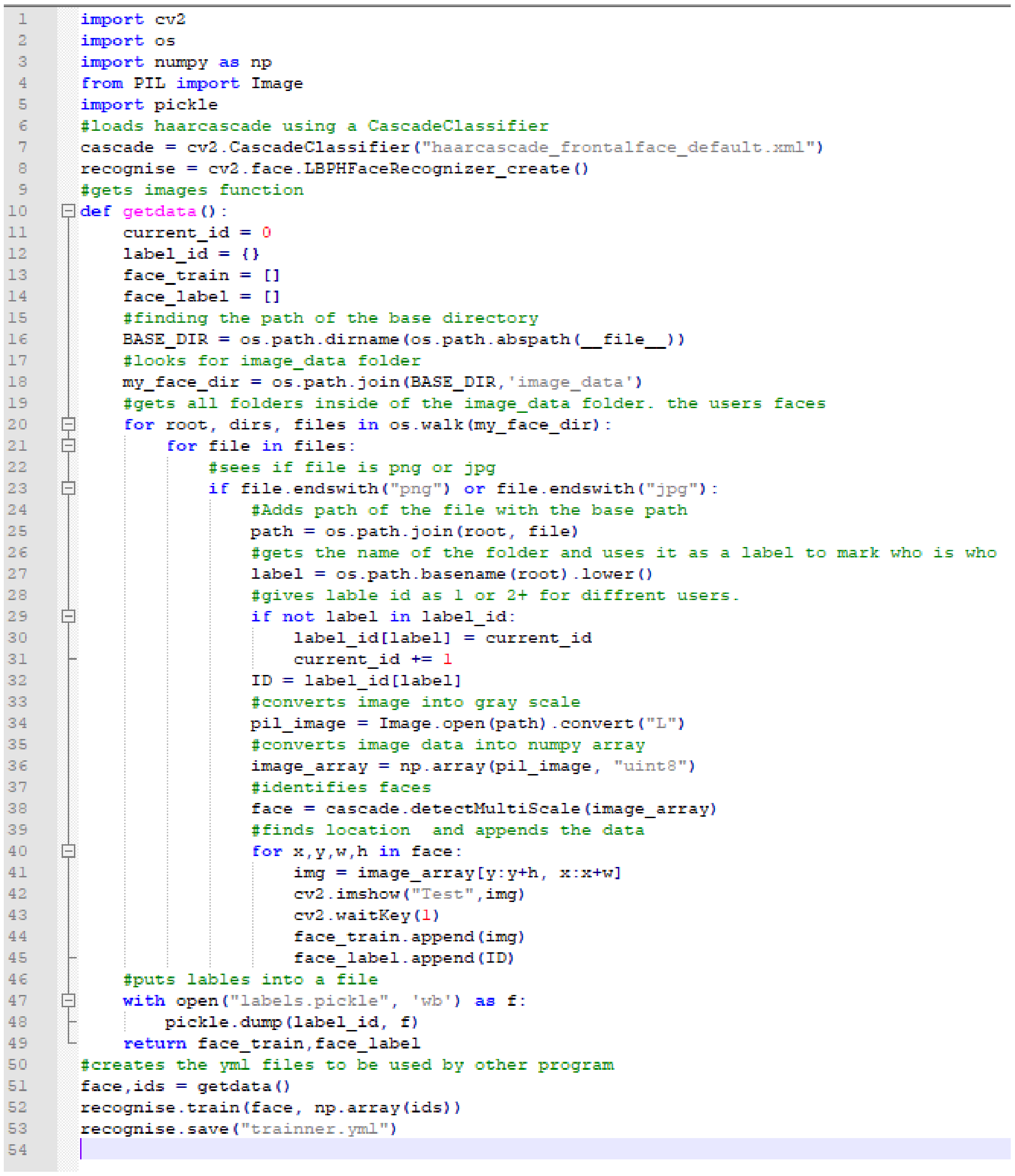


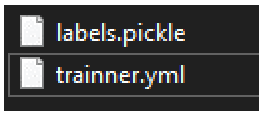
Figure 7: Folder setup and datasets

Following face recognition, was the collection of a dataset to begin the training process to take the face recognition software to a facial recogniser. This section will allow for any setup user to be recognised by the program and will require each user to upload at least 20 photos of their face. This number was used because of (Wilman Zou and Pong Yuen, 2012). The more data collected, the higher the accuracy of the trained data. All the user's data must be put in an image_data folder. Inside of this folder, users will name a new folder their name and have the data stored as shown in Figure 7 .



```python
import cv2
import os
import numpy as np
from PIL import Image
import pickle
#loads haarcascade using a CascadeClassifier
cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
recognise = cv2.face.LBPHFaceRecognizer_create()
#gets images function
def getdata():
    current_id = 0
    label_id = {}
    face_train = []
    face_label = []
    #finding the path of the base directory
    BASE_DIR = os.path.dirname(os.path.abspath(__file__))
    #looks for image_data folder
    my_face_dir = os.path.join(BASE_DIR,'image_data')
    #gets all folders inside of the image_data folder. the users faces
    for root, dirs, files in os.walk(my_face_dir):
        for file in files:
            #sees if file is png or jpg
            if file.endswith("png") or file.endswith("jpg"):
                #Adds path of the file with the base path
                path = os.path.join(root, file)
                #gets the name of the folder and uses it as a label to mark who is who
                label = os.path.basename(root).lower()
                #gives lable id as 1 or 2+ for diffrent users.
                if not label in label_id:
                    label_id[label] = current_id
                    current_id += 1
                ID = label_id[label]
                #converts image into gray scale
                pil_image = Image.open(path).convert("L")
                #converts image data into numpy array
                image_array = np.array(pil_image, "uint8")
                #identifies faces
                face = cascade.detectMultiScale(image_array)
                #finds location  and appends the data
                for x,y,w,h in face:
                    img = image_array[y:y+h, x:x+w]
                    cv2.imshow("Test",img)
                    cv2.waitKey(1)
                    face_train.append(img)
                    face_label.append(ID)
    #puts lables into a file
    with open("labels.pickle", 'wb') as f:
        pickle.dump(label_id, f)
    return face_train,face_label
#creates the yml files to be used by other program
face,ids = getdata()
recognise.train(face, np.array(ids))
recognise.save("trainner.yml")
```

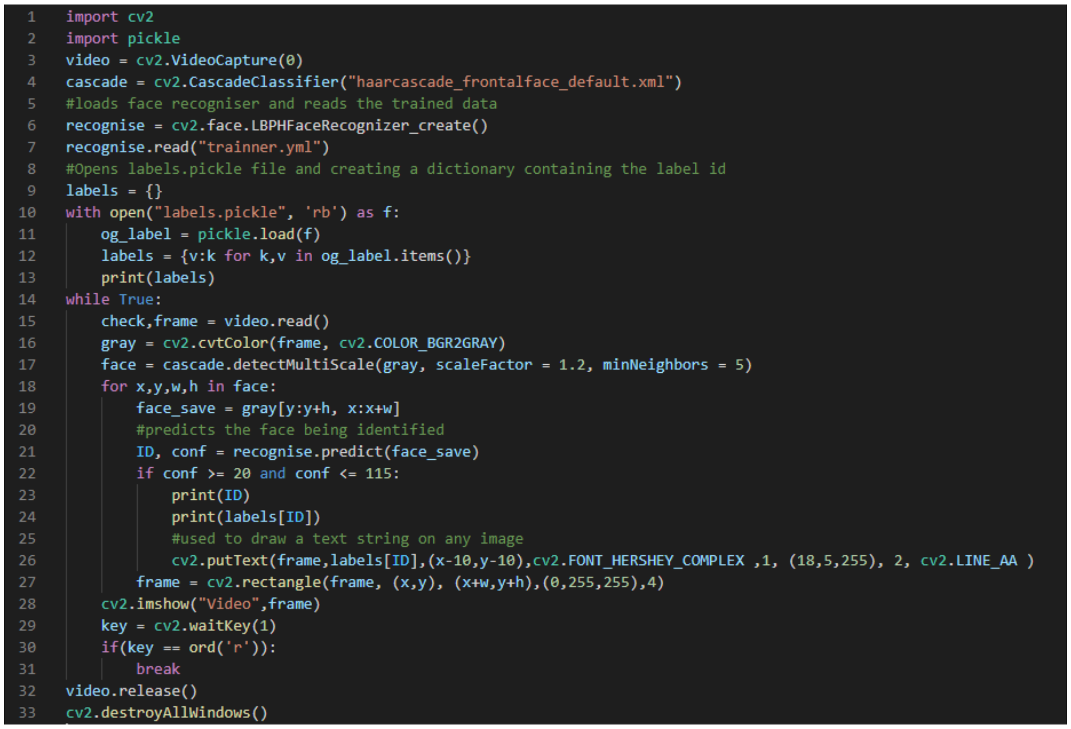
Figure 8: Trainer



labels.pickle
trainner.yml

Figure 9: Trainer files

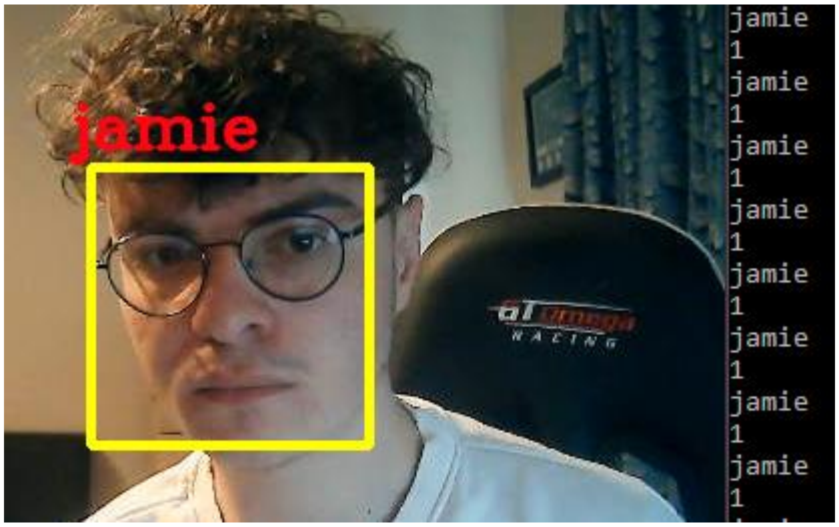
Figure 8 program must be run before any facial identification can be done. This works by collecting the data inside of the image_data folder. This data is then trained by using (OpenCV2013) library called “Haarcascade” which is a real time face detector. This library uses machine learning object detection that can identify faces in a video stream. Each individual folder inside of image_data will be given a number, this number can be used later to allow certain people access and others not. It is also used for the serial data that is required by the UNO R3 to turn the motor. Each image is then converted into grayscale to lower compute time and then into a NumPy array to be identified by “detectMultiScale”. After, it is run through a face finder similar to Figure 5 which then appends the data, creating a trained model to find the users faces. This is then saved in two files: one for user’s names called “labels.pickle” and the face trained data in “trainer.yml”. The yml loads the image with a detected face from the data and will create regions of interest which contains data like distance between eyes for every image uploaded and is what is used for identification. Once run the program creates the files shown in Figure 9.



```python
import cv2
import pickle
video = cv2.VideoCapture(0)
cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
#loads face recogniser and reads the trained data
recognise = cv2.face.LBPHFaceRecognizer_create()
recognise.read("trainner.yml")
#Opens labels.pickle file and creating a dictionary containing the label id
labels = {}
with open("labels.pickle", 'rb') as f:
    og_label = pickle.load(f)
    labels = {v:k for k,v in og_label.items()}
    print(labels)
while True:
    check,frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    face = cascade.detectMultiScale(gray, scaleFactor = 1.2, minNeighbors = 5)
    for x,y,w,h in face:
        face_save = gray[y:y+h, x:x+w]
        #predicts the face being identified
        ID, conf = recognise.predict(face_save)
        if conf >= 20 and conf <= 115:
            print(ID)
            print(labels[ID])
            #used to draw a text string on any image
            cv2.putText(frame,labels[ID],(x-10,y-10),cv2.FONT_HERSHEY_COMPLEX ,1, (18,5,255), 2, cv2.LINE_AA )
        frame = cv2.rectangle(frame, (x,y), (x+w,y+h),(0,255,255),4)
    cv2.imshow("Video",frame)
    key = cv2.waitKey(1)
    if(key == ord('r')):
        break
video.release()
cv2.destroyAllWindows()
```

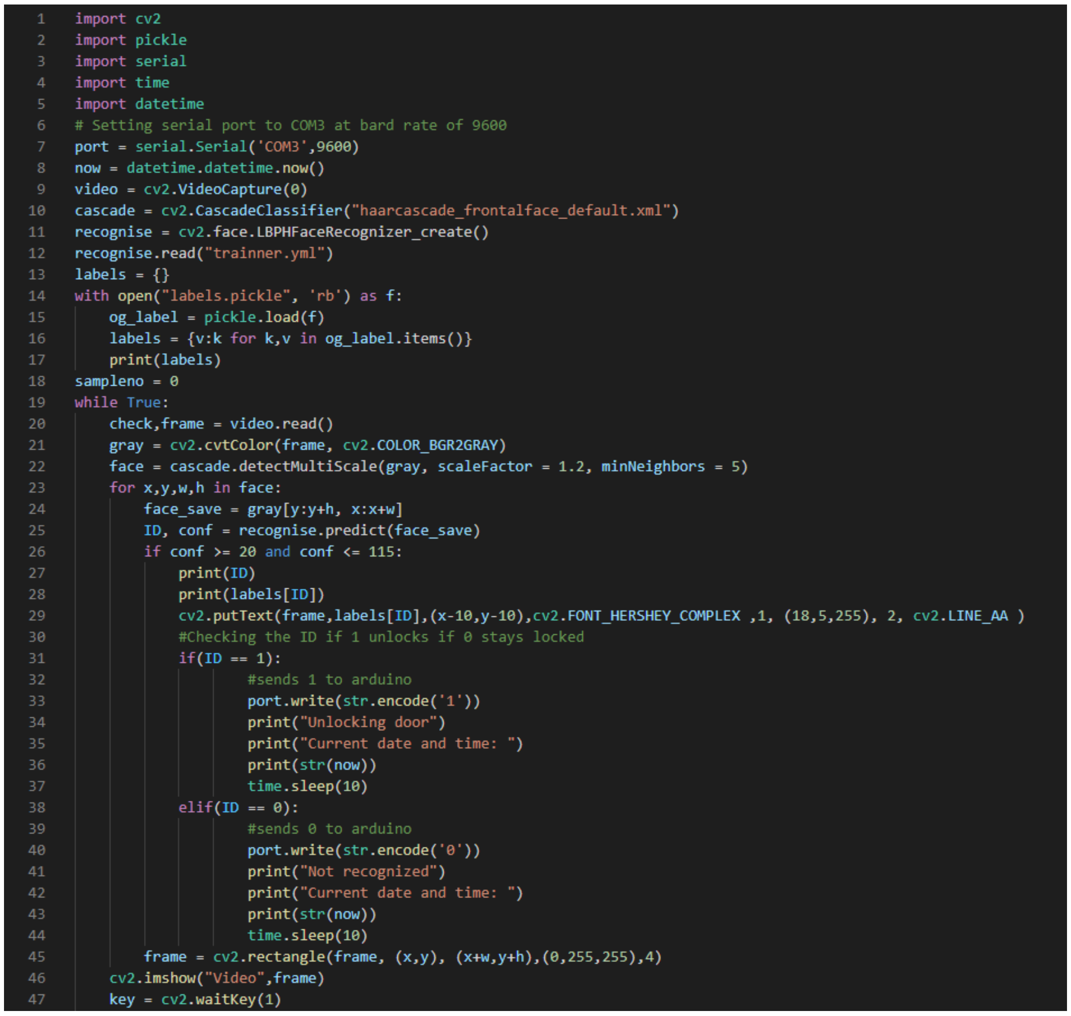
Figure 10: Facial recognition

Figure 10 will load in the trained model of all accepted users faces as well as the labels and is used in parallel with the webcam to begin to identify user's faces. In the webcam's video stream, once a face is found in the video it will be compared to the trained data and, if a match is found, it will output the user folder name and the number corresponding to it. This will be more accurate with larger datasets as shown in Figure 7. The program can be seen working in Figure 11 where it correctly identified a user and outputs the corresponding number. User faces are distinguished between one and another with the use of OpenCV library working as described by Shervin Emami and Valentin Suciu (2021).



```
jamie
1
jamie
1
jamie
1
jamie
1
jamie
1
jamie
1
jamie
1
jamie
1
```

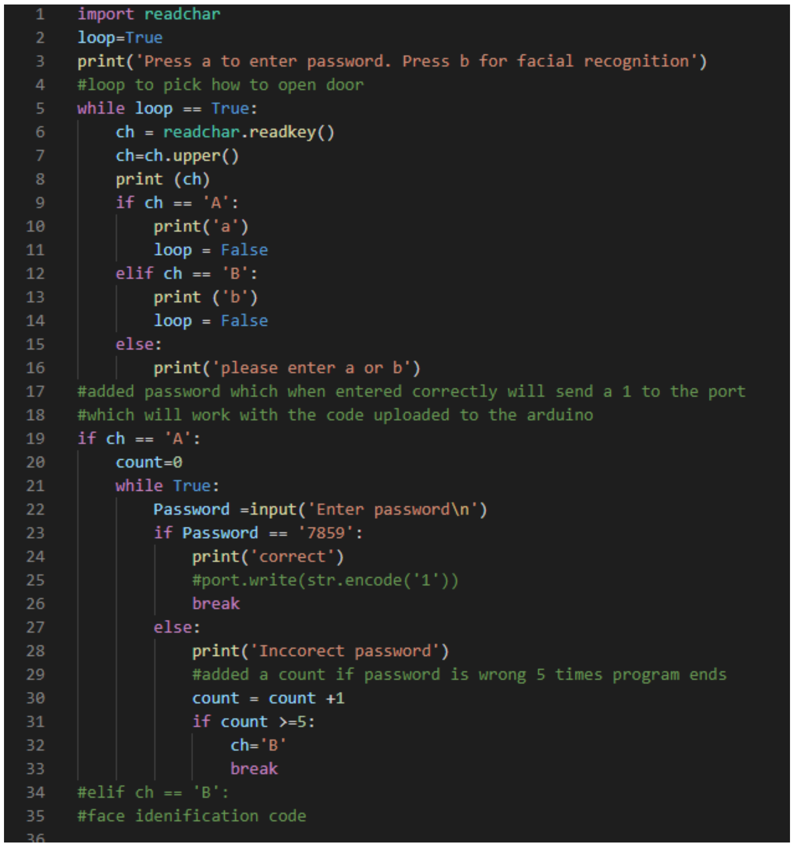
Figure 11: User facial detection



```python
import cv2
import pickle
import serial
import time
import datetime
# Setting serial port to COM3 at bard rate of 9600
port = serial.Serial('COM3',9600)
now = datetime.datetime.now()
video = cv2.VideoCapture(0)
cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
recognise = cv2.face.LBPHFaceRecognizer_create()
recognise.read("trainner.yml")
labels = {}
with open("labels.pickle", 'rb') as f:
    og_label = pickle.load(f)
    labels = {v:k for k,v in og_label.items()}
    print(labels)
sampleno = 0
while True:
    check,frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    face = cascade.detectMultiScale(gray, scaleFactor = 1.2, minNeighbors = 5)
    for x,y,w,h in face:
        face_save = gray[y:y+h, x:x+w]
        ID, conf = recognise.predict(face_save)
        if conf >= 20 and conf <= 115:
            print(ID)
            print(labels[ID])
            cv2.putText(frame,labels[ID],(x-10,y-10),cv2.FONT_HERSHEY_COMPLEX ,1, (18,5,255), 2, cv2.LINE_AA )
            #Checking the ID if 1 unlocks if 0 stays locked
            if(ID == 1):
                    #sends 1 to arduino
                    port.write(str.encode('1'))
                    print("Unlocking door")
                    print("Current date and time: ")
                    print(str(now))
                    time.sleep(10)
            elif(ID == 0):
                    #sends 0 to arduino
                    port.write(str.encode('0'))
                    print("Not recognized")
                    print("Current date and time: ")
                    print(str(now))
                    time.sleep(10)
        frame = cv2.rectangle(frame, (x,y), (x+w,y+h),(0,255,255),4)
    cv2.imshow("Video",frame)
    key = cv2.waitKey(1)
```

Figure 12: Serial outputs

Following this was the addition of serial interaction. This will control the UNO R3 using the Python program if it is connected to the same PC the program is running on. Setting the COM3 and the band rate to “9600“allows for the connection to be made. Depending on the Arduino in use, the port might differ; this can be found on the Arduino editor under tools and subsection port. Figure 4 shows the outcomes when each of the numbers are sent to the Arduino in addition, each time any request is made, the date and time of the event is recorded for added security this will also record any user names that corresponded with the request made.



```python
import readchar
loop=True
print('Press a to enter password. Press b for facial recognition')
#loop to pick how to open door
while loop == True:
    ch = readchar.readkey()
    ch=ch.upper()
    print (ch)
    if ch == 'A':
        print('a')
        loop = False
    elif ch == 'B':
        print ('b')
        loop = False
    else:
        print('please enter a or b')
#added password which when entered correctly will send a 1 to the port
#which will work with the code uploaded to the arduino
if ch == 'A':
    count=0
    while True:
        Password =input('Enter password\n')
        if Password == '7859':
            print('correct')
            #port.write(str.encode('1'))
            break
        else:
            print('Inccorect password')
            #added a count if password is wrong 5 times program ends
            count = count +1
            if count >=5:
                ch='B'
                break
#elif ch == 'B':
#face idenification code
```

Figure 13: Unlock door with password or face

Finally, a choice would be made of how a user would like to open a door. Constantly running option B is the best option as this allows for a user to reset the system with “R” if their face is not recognised. This will then allow them to enter option A which will allow them to enter a password. In this use case, the password can only be entered on a PC; however the simple addition of a keypad by the door would replace this, allowing for both the password and facial ID to be used at the same time . If the user enters the password wrong five times, the program is reset to help avoid attacks.

3. **Testing**

Test	Expected Outcome	Actual Outcome	Solution
Is UNO R3 receiving power?	When plugged in power light flashes on.	Power light flashes on.	Not required.
UNO R3 is wired with the layout shown in Figure 1 and the code in Figure 3 is used to see if the hardware is functional.	The motor will turn clockwise then anticlockwise continuously with a LED display on the motor interface.	Motor rotates clockwise then anticlockwise with a LED display to show it is functional.	Not required.
Are serial inputs of 1 and 0 to the UNO R3 interpreted correctly and are all the components fully wired?	When receiving serial input of a 1 the step motor should do one full rotation, wait 6 seconds and then go back on itself with text outputs saying when it is locking and unlocking. When receiving 0 the output should read 'not recognised'.	When 1 is received unlocking is outputted followed by 1 full clockwise rotation 6 seconds pass and locking is outputted then a full anticlockwise turn is made. When 0 is received, not recognised is outputted.	Not required.
Are we able to see the webcam stream?	Once the program is running, a window displaying the webcam's stream should pop-up.	The video stream window pops up.	Not required.

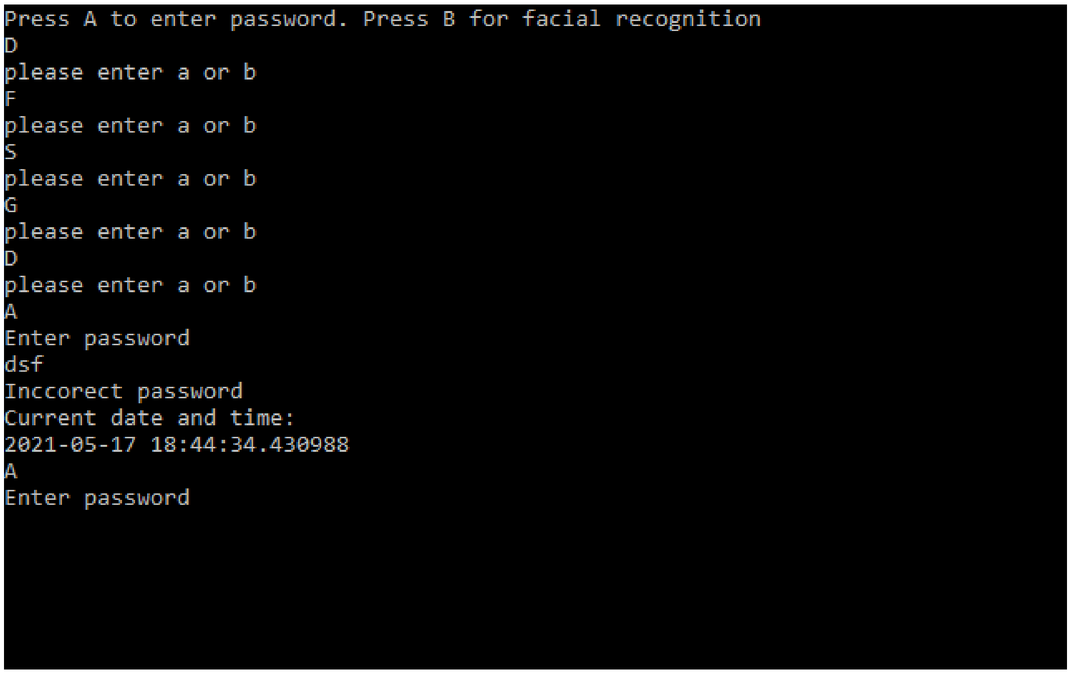
Is any face identified with a box?	Any face that enters the video stream will have a box drawn around it.	Lots of different results depending mostly on light levels and obstructions on faces. Sometimes identifies objects as faces.	As faces will eventually be found the password backup will work to allow users access. Additionally, once a user's face data is uploaded it will be more accurate.
Can datasets be used to train a model?	User data will be trained; this is done via each image in the dataset being processed with outputs into two separate files, the first being user labels and the second being the trained dataset values.	Both files are outputted, however when running not all the faces are recognised in the dataset. Some backgrounds are recognised as faces. Both files are outputted correctly.	The error rate can be improved both by taking higher quality photos on blank backgrounds and utilising larger datasets leading to a more accurate model.
Facial recognition on a user	A user label will be outputted as well as a box around their face with the label on the dataset.	The user is recognised with a box and label around the face and name and number outputted in the console. It can take some time to find the face.	More data would improve the speed of recognition.

Multiple user datasets	Each user dataset will be trained with only two file outputs all contained in the same files. As well as a number given to each user.	All user data is trained and assigned a number.	Not required.
Facial recognition on multiple users	Both users will be recognised, and the corresponding name will appear on the box around their face.	Sometimes wrong names appear on the user. In most cases correct names appear with the correct number outputted.	More data would increase the accuracy of the results.
Use of the user's photo and not their face.	The face will not be recognized.	The users face will be found in the photo with a box around with the corresponding label and number output.	More data should help the problem somewhat but this is a problem with OpenCV.
Non-user facial recognition	The face will be found however no name or number will be outputted.	Occasionally a non-user face is seen to be a user's face outputting a wrong name and number.	More data would increase the accuracy and a non-user default case loop would allow for non-users to be found but no action taken.

Facial recognition serial output.	Once a face is recognised a serial output is sent to the UNO R3.	Face is recognised and the correct number is sent as a serial output to Arduino causing the code shown in Figure 4 to run.	Not required.
Ability to pick how to control the motor.	Has a choice of password or facial recognition to open the lock.	Choice is given.	Not required.
When picking how to control the motor enter the wrong key.	The user will be asked to re-enter the key .	The user is told to re-enter the key.	Not required.
Enter the password for serial output.	If the password is entered correctly serial 1 is sent to UNO R3.	A correctly entered password results in serial 1 is sent to UNO R3.	Not required.
Password spam stopper.	If the wrong password is entered five or more times the program closes.	The program closes if an incorrect password is entered five or more times.	A delay could be added instead of stopping the program.

Table 3: Full project testing table

4. **Operation**



```
Press A to enter password. Press B for facial recognition
D
please enter a or b
F
please enter a or b
S
please enter a or b
G
please enter a or b
D
please enter a or b
A
Enter password
dsf
Inccorect password
Current date and time:
2021-05-17 18:44:34.430988
A
Enter password
```

Figure 14: Initial start up

When the program is first run, the initial user will be greeted with Figure 14. This gives the first user a choice between entering a password or using facial recognition. The use of exception handling prevents another choice from being performed. This procedure also isn't case sensitive. It also shows what happens when "A" is selected, and a password is entered incorrectly. This information will be very helpful for admins as it will make checking for any damages easier as well as being used for a clock in timer for all users. When the correct password is entered the step motor will rotate fully clockwise, wait six seconds and lock again. This allows enough time for users to input the password and enter the door before it starts to lock again. This prevents users from forgetting to lock doors.

However, the password is used as more of a back-up, the main system being facial recognition which is option "B". When option "B" is selected, two windows will open. One of these windows will be for the webcam and one for the serial outputs, labels and current data and time of request. This will also allow admins to watch the live feed of any camera and see the time that a face was recognised and who it is. Once a face has been recognised the same process will be followed as for if a password was used. The door will be unlocked if the corresponding face has access to that room. Unlike the password, different doors will not require a different passcode, only the correct serial output from the identified face which is assigned with datasets. This is the main reason it is the primary use of this program as users will not need to remember passwords.

If the program does not recognise a face that should have been granted access, the “R” key will reset the program, allowing a user to enter a passcode instead and working as a failsafe. The implementation of this feature allows the facial recognition software to run constantly, only being turned off when updates are required, or it stops working. This also allows for admins to update user datasets. Any time datasets are added to or changed; the trainer must be run again to include the new data. This is also true when expanding the system to more doors.

Some general improvements that could be made are the addition of a log in which all user data is logged when entering a room such as storing the data and time along with the person who entered at that set time. This could be further improved by the addition of a photo being taken anytime a serial signal is set so that, no matter the action that has taken place, a log will have been made. However, this could cause problems depending on the location of the camera as stated by Andrejevic and Selwyn (2019). A general improvement that could be made is the addition of a greater number of higher quality images inside of the datasets. This would decrease the likelihood of a false reading and improve the speeds at which faces are found to unlock doors Wilman Zou and Pong Yuen (2012) and Tran, Mayhew, Kim, Karande and Kaplan (2018) respectively showed that this can be done through both increasing the quality and size of the training datasets.

Maintenance will be required to keep the system fully operational. This will include updating the datasets to keep UpToDate images of each user while also keeping the backlog of images to improve the accuracy. All hardware will be required to be replaced if damaged or broken. The cameras will be required to be cleaned on a regular basis to prevent dust build up and smudges.

Improvements for accessibility for disabled users would be a great addition to the project in the future. Some improvements would be audio clues for when a door is locking and unlocking, allowing users with poor sight to be able to use the system much more easily. As well as the addition of braille on a keypad. Another improvement that would be extremely helpful for disabled users is the addition of automatic door opening.

Chapter 5

Conclusions

The results from this project indicate that a functional facial recogniser, when trained with enough data, has the ability to distinguish between multiple users, who will each have separate outputs. This will then allow different users access to different locations. When certain outputs are sent to the UNO R3 from the Python program, it will perform a set action of rotation if the command received was “1”. This will be the locking and unlocking mechanism. After the output for the python code is received and the door unlocked, six seconds will pass allowing the user to enter and shut the door before the lock resets into the locked position. This is provided that the “B” option was selected upon start-up. “B” Should be the default running configuration allowing the program to run continuously. This was done with the use of a dataset per person of twenty photos each taken on a blank background.

When option “A” is selected upon start-up or resetting of the program, users will be asked to enter a passcode. If incorrectly entered five times, the program will stop to prevent brute force attacks. If entered correctly, a serial code will be sent to the Arduino which will begin to unlock before locking again after six seconds. This method requires users to remember passcodes which is why it is recommended that option “B” is continuously run until a problem occurs.

A few problems that have been noticed are light can have a negative effect on the recognition of any faces even without the datasets being used to recognise different users. This should be fixed with the addition of more data however it might be a problem with the OpenCV library.

(Zuo, Zeng and Tu, 2010) discussed potential improvements to the algorithms with the use of Adaboost classifier algorithm, which should help improve the accuracy of the original algorithm and detection of faces. This should also help with the second problem of no users sometimes being recognised as a user which will unlock a door causing a security risk. The main solution to this would be the implementation of a much larger data set which will help fix the problem. A change that should be implemented is for the password and facial recogniser to be running in parallel so that both can be used as backup systems if the other fails instead of requiring a reset.

For this project to be truly used in a real life scenario it would require expansion that was not available at the time of developing this project. This expansion would increase the amount of motors and therefore require the use of resistors and power relays for the ability to make a parallel circuit. This would then demonstrate how different users have access to different locations. This also would have allowed for further development into the role system of the project. Users grouped into roles would have access to certain doors and passcodes, as well as the ability to set admins who would have full control over the system. This is also the reason GPS was not implemented due to the lack of connectivity via the development PC and a mobile device which was not available.

Finally, with the addition of a larger and higher quality dataset, this project could be implemented onto a door providing a locking mechanism was attached to the step motor alongside a keypad. This makes the project viable for use on a single door in a real world scenario. However, due to the lack of in-person access to hardware and university resources throughout the project, the scalability of the project is untested. In theory, the code should work the same for different use cases such as different door locks, this project could definitely be adapted to fit in a multi-door security system.

Chapter 6

Reflective Analysis

At the beginning of the project, the planning phase went very well with the creation of the seven-step process in which each step had to be completed before the last. This plan worked incredibly well and was followed with a few changes due to unforeseen circumstances, such as not having multiple doors as well as the implementation of GPS. Everything else that was planned was added with various levels of success. This shows that the Waterfall method was the correct choice for this project due to its strict nature and clear list of goals. The literature review section helped with the development of many aspects of the program, such as features which should not be implemented, as well as how to work libraries such as OpenCV and motor controls. The review greatly helped with the implantation of datasets as understanding that higher resolution images would result in a much higher accuracy, the amount of data collected being a larger problem than quality.

The most successful part of the project was the hardware aspect with the serial inputs from the Python code. The UNO R3 and all hardware required for the project worked as intended and without problem. This is also true for the implementation of a password for door unlocking and the exception handling that accompanied it along with the user interface exception handling making the program very user friendly once data had been uploaded.

Some slight problems with face detection were encountered. The main issue was with light which can cause small problems when the program is trying to find a face in the video stream in a suitable time frame. However, for the purpose needed, facial detention works and is implemented well with only a few cases of problems, even in these cases the face is eventually found after some time. Another minor problem was in the datasets. When being trained, some faces that would be recognised would not be faces at all, most commonly random places on the background or the user's hair would be marked as a face. This was only a problem on some images, however it would cause problems in facial identification later on.

The largest problem is occasionally non users are seen to be a user as well as some users are identified with the wrong label, both of these problems have the same effect. The wrong person has access to something they should not. This issue is mostly caused by the training data problems which can be improved with the addition of more data into the datasets. This could also be improved with the use of a different library such as Tensorflow which would use deep learning for the datasets, this could increase accuracy.

The largest problem that affected the project as a whole was the lack of access to university facilities during the time of this project. This was not a possibility due to Covid-19 complications and affected the final outcome of this project in many ways. The main one being a lack of Bluetooth on the development PC which prevented the wireless connection of the Arduino and the addition of a mobile phone. This also had an effect on the hardware as it required demonstration on only one motor making it impossible to have a multi-lock system to show the use as a real security system.

Overall the project was successful at accomplishing the main goals set with minor problems such as the dataset size being too small which can be easily fixed with additional time. This meant the prototype worked as intended and can be demonstrated with one motor however the addition of more would only require small changes to the code and updating the circuitry.

References

OpenCV(2013)haarcascade_frontalface_default.xml. Available at: https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml. [Accessed 14/04/2021]

Xianghua Fan, Fuyou Zhang, Haixia Wang and Xiao Lu, "The system of face detection based on OpenCV," 2012 24th Chinese Control and Decision Conference (CCDC),23-25 May 2012 , Taiyuan China. Available from https://ieeexplore.ieee.org/abstract/document/6242980 [Accessed 14/02/2021]

Andrejevic, M. and Selwyn, N., 2019. Facial recognition technology in schools: critical questions and concerns. Learning, Media and Technology, Available at: https://www.tandfonline.com/doi/full/10.1080/17439884.2020.1686014 [Accessed 15/02/2021].

Karovaliya, M., Karedia, S., Oza, S. and Kalbande, D., 2015. Enhanced Security for ATM Machine with OTP and Facial Recognition Features. Procedia Computer Science, Available at: https://www.sciencedirect.com/science/article/pii/S1877050915004093[Accessed 16/02/2021].

Jeong, J., 2016. A Study on the IoT Based Smart Door Lock System. *Lecture Notes in Electrical Engineering*,Available at: https://link.springer.com/chapter/10.1007/978-981-10-0557-2_123 [Accessed 18/02/2021].

Ngoc Quy Le, Jung Uk Cho, Jae Wook Jeon , *2006* "Application of Velocity Profile Generation and Closed-Loop Control in Step Motor Control System," Available at:https://ieeexplore.ieee.org/abstract/document/4108386[Accessed 23/02/2021].

Leo Louis, 2016. Working Principle of Arduino and Using it as a Tool for Study and Research. *International Journal of Control, Automation,* *Communication and Systems*, Available at: https://www.academia.edu/download/54016484/6ijcacs03.pdf [Accessed 25/02/2021].

Zong, W. and Huang, G., 2011. Face recognition based on extreme learning machine. *Neurocomputing*, Available at: https://www.sciencedirect.com/science/article/abs/pii/S0925231211002578.[Accessed 27/02/2021].

Wilman W W Zou and Pong C Yuen, 2012"Very Low Resolution Face Recognition Problem," in *IEEE Transactions on Image Processing*,Available at:https://ieeexplore.ieee.org/abstract/document/5957296[Accessed 02/03/2021].

E. Tran, M. B. Mayhew, H. Kim, P. Karande and A. D. Kaplan,2018 "Facial Expression Recognition Using a Large Out-of-Context Dataset," *2018 IEEE Winter Applications of Computer Vision Workshops (WACVW)* Available at: https://ieeexplore.ieee.org/abstract/document/8347112 [Accessed 04/03/2021]

Shervin Emami and Valentin Suciu, 2021. *Facial Recognition using OpenCV*.Available at: http://jmeds.eu/index.php/jmeds/article/view/Facial_Recognition_using_OpenCV[Accessed 04/03/2021].

Circuito.io. 2021. *How to wire to Arduino Uno*.Available at: https://www.circuito.io/app?components=512,11021 [Accessed 13/4/2021].

Min Zuo, Guangping Zeng and Xuyan Tu, "Research and improvement of face detection algorithm based on the OpenCV," *The 2nd International Conference on Information Science and Engineering*, 2010,Hangzhou, China [Accessed 13/05/2021].