FACE RECOGNITION & DOOR LOCK WITH PIN

DESIGN WITH MICROPROCESSORS PROJECT DOCUMENTATION

Team: Jurcan Diana-Maria, Ilovan Alexandra
Coordinating professor: Danescu Radu
Group: 30432
Year: 2022-2023

Table of contents:

1. Introduction

Face recognition:

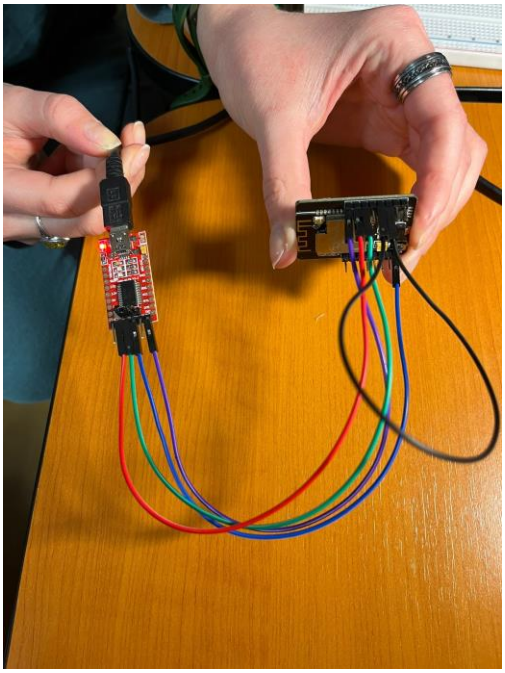
This part of the project requires the usage of an ESP-32 Cam module and an FTDI UART module in order to work. Once the connection between the camera and the server is established, the project will be able to detect the faces of the people facing the camera and, in case the given face matches the pictures stored internally, it will display the name of the person, thus signalling that the face has been recognized.

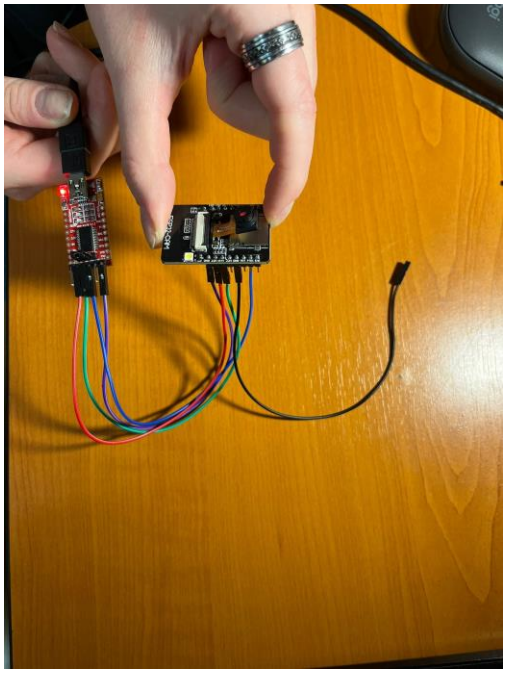
Door lock system:

This part of the project is a simulation of a smart door that opens if given the correct pin via a keypad. The user is first prompted to enter the password and has 3 chances to get it right. If the password is not correct, the red led will turn on and the user will be prompted to re-enter it. If the entered password is correct, the green led will turn on, and the door will be unlocked using a servo-motor system.

2. Analysis and design

Face recognition:

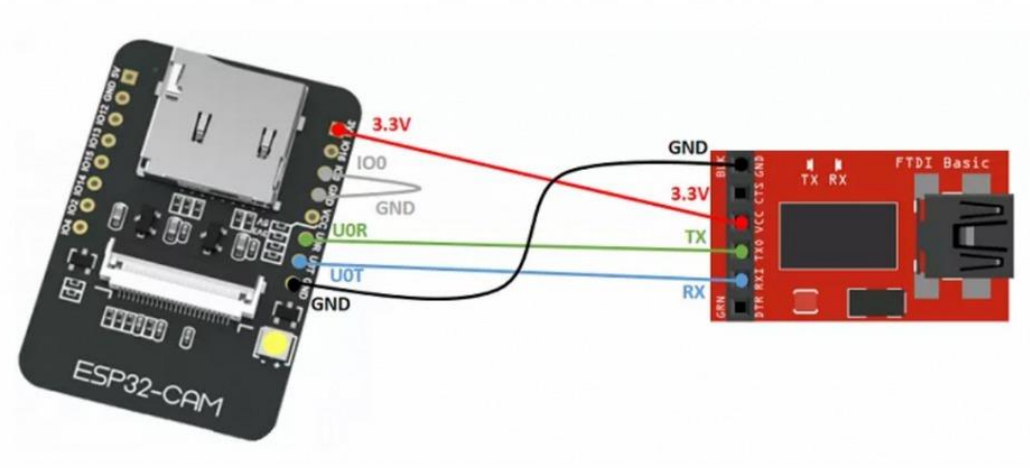




Instructions:

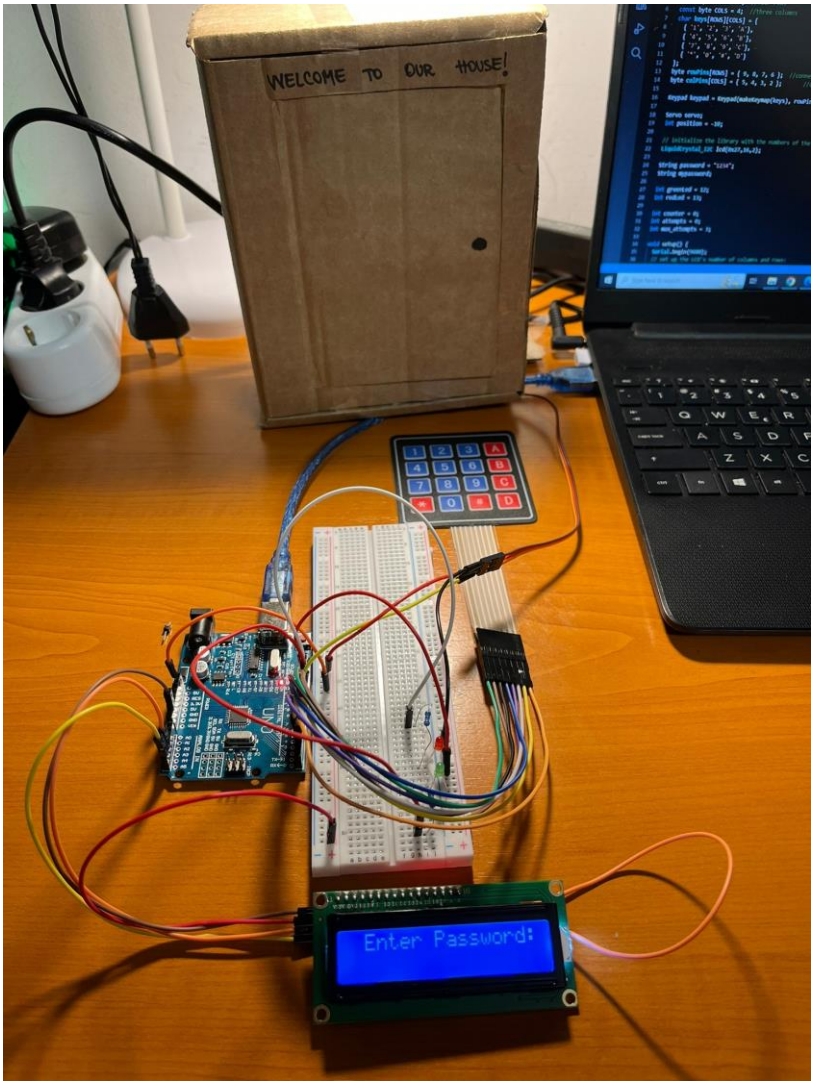
Connect the 5V & GND Pin of ESP32 to 5V & GND of FTDI Module. Similarly, connect the Rx to UOT and Tx to UOR Pin. And the most important thing, you need to short the IO0 and GND Pin together. This is to put the device in programming mode. Once programming is done you can remove it.

This is the connection between the ESP32-CAM and FTDI:



This is the output of the Arduino code, in Serial Monitor. The IP address visible in the serial monitor will then be used to set the URL in Python code in order to establish the connection between our camera and the Python face detection application.

Door lock system:

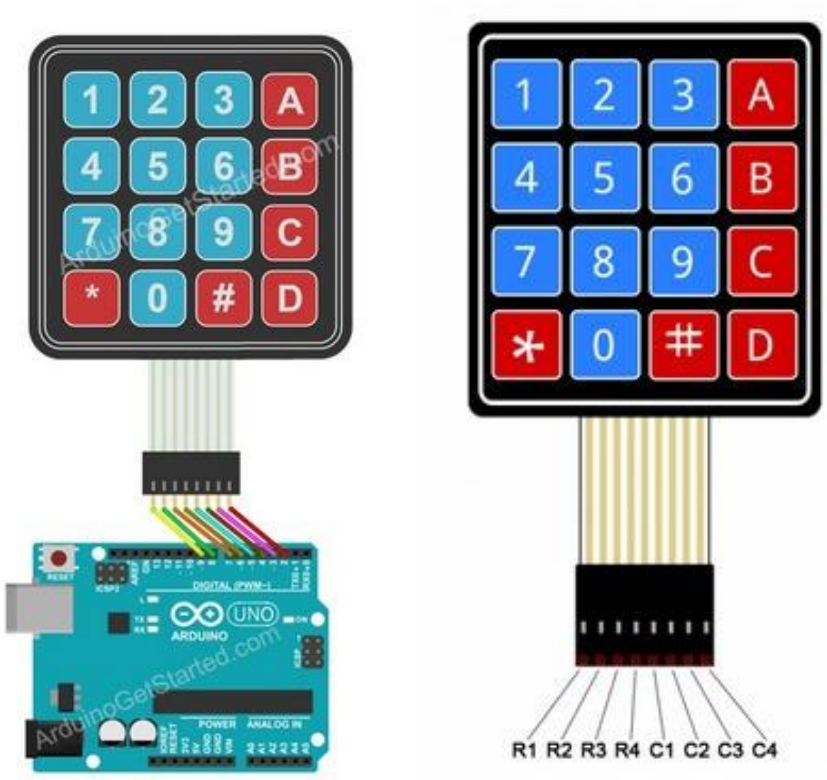


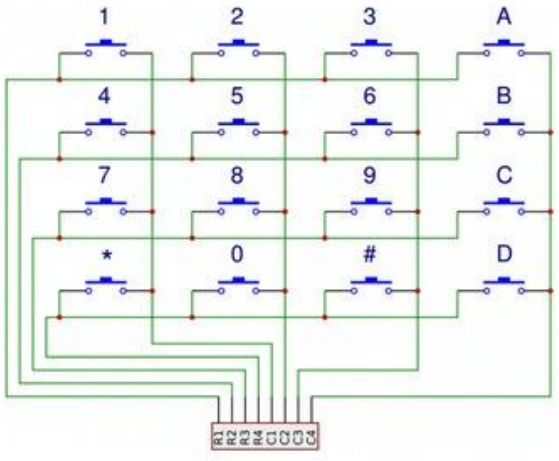
Components used:

- Arduino Uno Board;
- Keypad 4x4;
- LCD I2C;
- Servo motor;
- 2 leds;
- 2 resistors (220 ohms);
- Wires.

Connections:

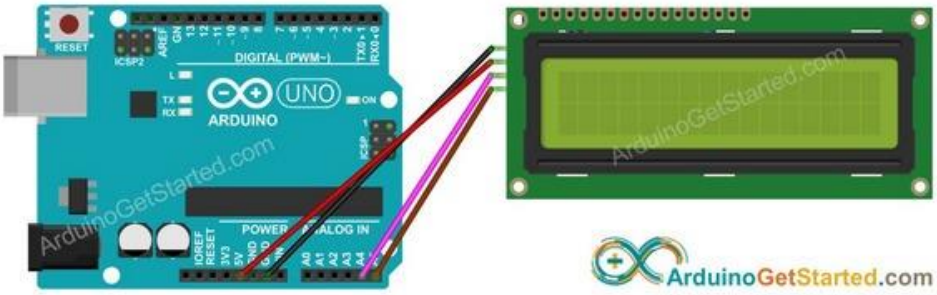
- Keypad & Arduino: R1-Pin 9, R2-Pin 8, R3-Pin 7, R4-Pin 6, C1-Pin 5, C2-Pin 4, C3-Pin 3, C4-Pin 2





- LCD I2C & Arduino Board

Wiring Diagram



This image is created using Fritzing. Click to enlarge image

LCD I2C	Arduino Uno, Nano
Vin	5V
GND	**GND**
SDA	A4
SCL	A5

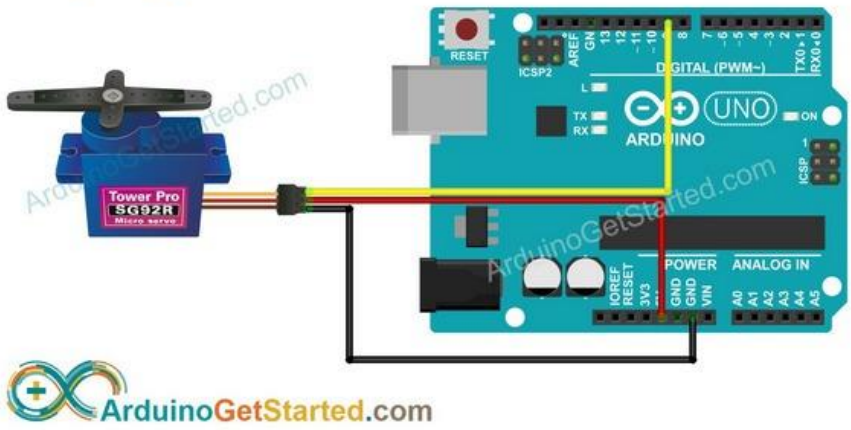
- Servo & Arduino Board

Pinout

The servo motor used in this example includes three pins:

- VCC pin: (typically red) needs to be connected to **VCC** (5V)
- GND pin: (typically black or brown) needs to be connected to **GND** (0V)
- Signal pin: (typically yellow or orange) receives the PWM control signal from an Arduino's pin.

Wiring Diagram



- Leds

Pinout

LED includes two pins:

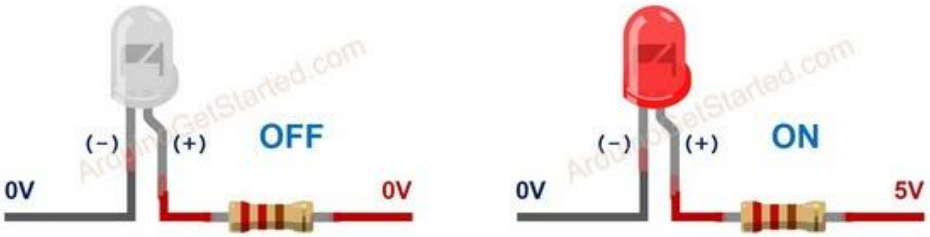
- Cathode(-) pin: needs to be connected to **GND** (0V)
- Anode(+) pin: is used to control LED's state

ArduinoGetStarted.com

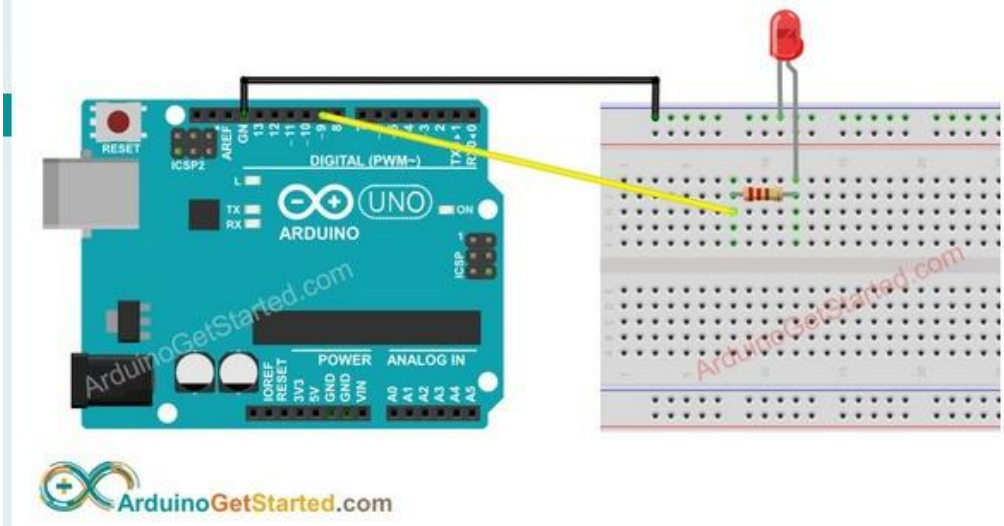
How It Works

After connecting the cathode(-) to **GND**:

- If connecting **GND** to the anode(+), LED is **OFF**.
- If connecting **VCC** to the anode(+), LED is **ON**.



ArduinoGetStarted.com



ArduinoGetStarted.com

3. Implementation

Face recognition:

Arduino Code:

CameraWebServer_copy.ino | app_httpd.cpp | camera_index.h | camera_pins.h | debug_custom.json

```
#include <WebServer.h>
#include <WiFi.h>
#include <esp32cam.h>

const char* WIFI_SSID = "02";
const char* WIFI_PASS = "hateithere";

WebServer server(80);



static auto loRes = esp32cam::Resolution::find(320, 240);
static auto midRes = esp32cam::Resolution::find(350, 530);
static auto hiRes = esp32cam::Resolution::find(800, 600);

void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d %db\n", frame->getWidth(), frame->getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo()
{
  if (!esp32cam::Camera.changeResolution(loRes)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if (!esp32cam::Camera.changeResolution(hiRes)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if (!esp32cam::Camera.changeResolution(midRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}

```

```
void  setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" : "CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");

  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);

  server.begin();
}
```

```
void loop()
{
  server.handleClient();
}
```

The Arduino code uses the ESP32Cam library to interact with the ESP32-CAM module and the WebServer library to set up a web server on the ESP32. The code sets up different routes (handleJpgLo, handleJpgHi, handleJpgMid) to serve different resolutions of the image captured by the ESP32-CAM. These routes are then handled by the server.handleClient() function in the loop() method.

Python Code:

face_detection_attendace.py

```python
import pandas as pd
import cv2
import urllib.request
import numpy as np
import os
from datetime import datetime
import face_recognition

path = r'/home/alexandra/Uni/Year3/Sem1/DMP/FaceRecognitionProject/ATTENDANCE/image_folder/'
url = 'http://192.168.167.206/cam-hi.jpg'
##'''cam.bmp / cam-lo.jpg /cam-hi.jpg / cam.mjpeg '''

if 'Attendance.csv' in os.listdir(os.path.join(os.getcwd(), 'attendance')):
    print("there iss..")
    os.remove("Attendance.csv")
else:
    df = pd.DataFrame(list())
    df.to_csv("Attendance.csv")

images = []
classNames = []

myList = os.listdir(path)
print(myList)
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classNames.append(os.path.splitext(cl)[0])
print(classNames)
```

face_detection_attendace.py

```python

def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList


def markAttendance(name):
    with open("Attendance.csv", 'r+') as f:
        myDataList = f.readlines()
        nameList = []
        for line in myDataList:
            entry = line.split(',')
            nameList.append(entry[0])
            if name not in nameList:
                now = datetime.now()
                dtString = now.strftime('%H:%M:%S')
                f.writelines(f'\n{name},{dtString}')


encodeListKnown = findEncodings(images)
print('Encoding Complete')
```

face_detection_attendace.py

```python
while True:
    # success, img = cap.read()
    img_resp = urllib.request.urlopen(url)
    imgnp = np.array(bytearray(img_resp.read()), dtype=np.uint8)
    img = cv2.imdecode(imgnp, -1)
    # img = captureScreen()
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
        # print(faceDis)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            # print(name)
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            markAttendance(name)

    cv2.imshow('Webcam', img)
    key = cv2.waitKey(5)
    if key == ord('q'):
        break
cv2.destroyAllWindows()
cv2.imread
```

The Python code uses the pandas library to create a CSV file to store the attendance, OpenCV library to capture the frames from the ESP32-CAM and detect the faces, and the face_recognition library to match the detected face with known faces. The code uses the urllib library to download the image from the ESP32-CAM using its IP address and the specified route in the Arduino code. Then it uses the OpenCV library to detect faces in the image and match them with known faces. If a match is found, the code writes the attendance to the CSV file.

It is important to note that for the python script to work, the ESP32 and the device running the Python script must be connected to the same network. Also, the ESP32-CAM IP address should be entered in the Python script.

You should also check that the libraries that are being used are properly installed on your system and that the ESP32-CAM is properly configured and working.

Door lock system:

```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

const byte ROWS = 4;  //four rows
const byte COLS = 4;  //three columns
char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A'},
  { '4', '5', '6', 'B'},
  { '7', '8', '9', 'C'},
  { '*', '0', '#', 'D'}
};
byte rowPins[ROWS] = { 9, 8, 7, 6 };  //connect to the row pinouts of the keypad
byte colPins[COLS] = { 5, 4, 3, 2 };       //connect to the column pinouts of the keypad

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

Servo servo;
int position = 0;

// initialize the library with the numbers of the interface pins
LiquidCrystal_I2C lcd(0x27,16,2);

String password = "1234";
String mypassword;

int greenLed = 12;
int redLed = 13;

int counter = 0;
int attempts = 0;
int max_attempts = 3;
```

```
void setup() {
  Serial.begin(9600);
  // set up the LCD's number of columns and rows:
  lcd.init();
  lcd.clear();
  lcd.backlight();      // Make sure backlight is on

  // Print a message on both lines of the LCD.
  lcd.setCursor(1, 0);   //Set cursor to character 2 on line 0

  pinMode(redLed, OUTPUT);
  digitalWrite(redLed, LOW);

  pinMode(greenLed, OUTPUT);
  digitalWrite(greenLed, LOW);

  servo.attach(10);

  Serial.println("Enter password:");
  lcd.print("Enter Password:");
}

void loop() {

  keypadfunction();
}
```

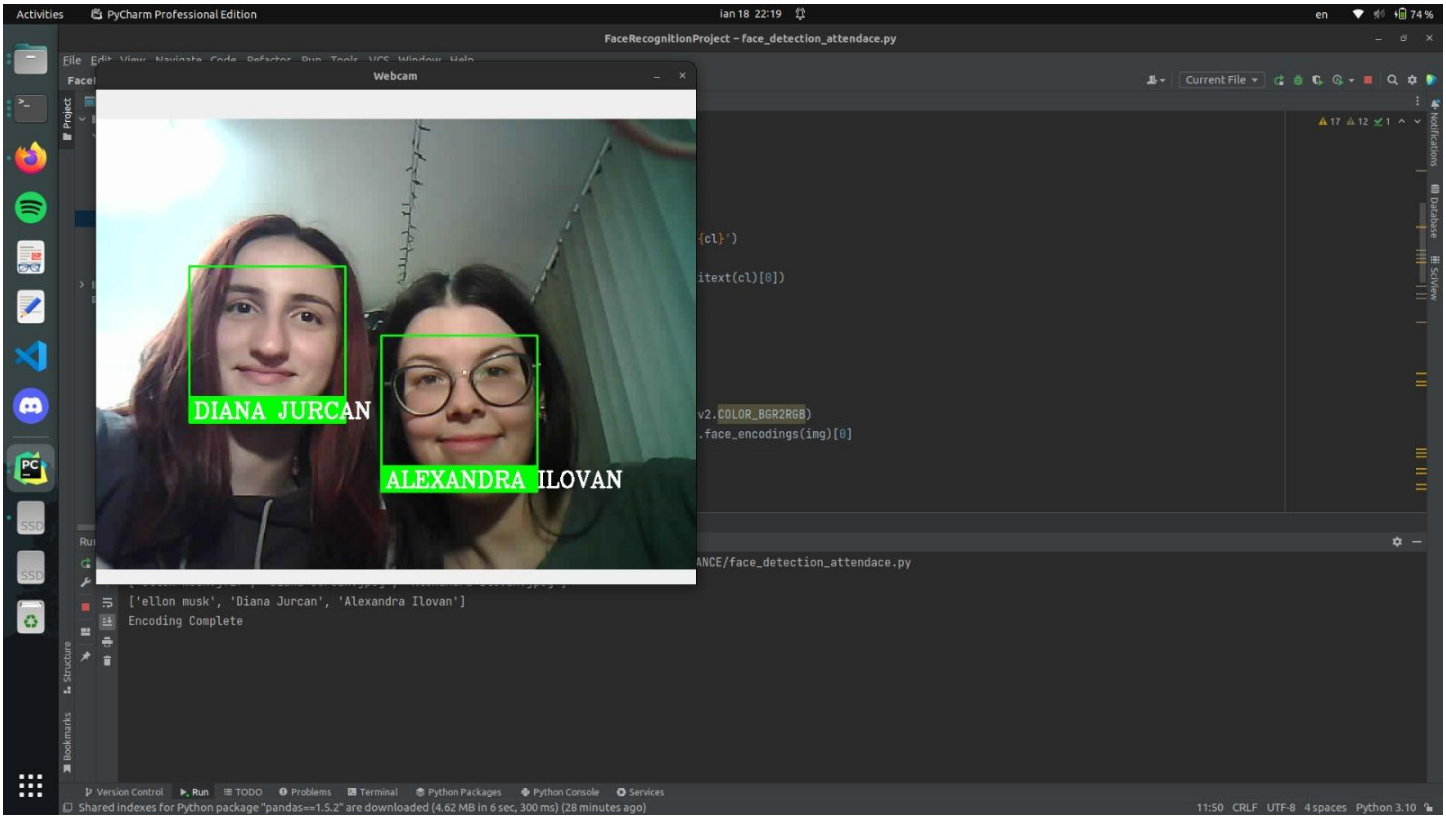
```
void keypadfunction() {
  char key = keypad.getKey();

  if (key) {
    Serial.println(key);
    counter = counter + 1;
    lcd.setCursor(counter, 1);
    lcd.print("*");
  }

  if (key == '1') {
    mypassword = mypassword + 1;
  }

  if (key == '2') {
    mypassword = mypassword + 2;
  }

  if (key == '3') {
    mypassword = mypassword + 3;
  }

  if (key == '4') {
    mypassword = mypassword + 4;
  }

  if (key == '5') {
    mypassword = mypassword + 5;
  }

  if (key == '6') {
    mypassword = mypassword + 6;
  }
```

```
  } else {

    Serial.print("Wrong.");
    digitalWrite(redLed, HIGH);
    digitalWrite(greenLed, LOW);
    attempts = attempts + 1;
    if (attempts >= max_attempts) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Locked out.");

      digitalWrite(redLed, HIGH);
      delay(5000);
      digitalWrite(redLed, LOW);
      attempts = 0;
    }
    mypassword = "";
    counter = 0;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Wrong password.");
    delay(3000);

    lcd.setCursor(0, 1);
    lcd.print("Max attempts: 3");
    delay(3000);

    lcd.clear();
    lcd.print("Enter password:");
    lcd.setCursor(0, 1);
  }
}
```

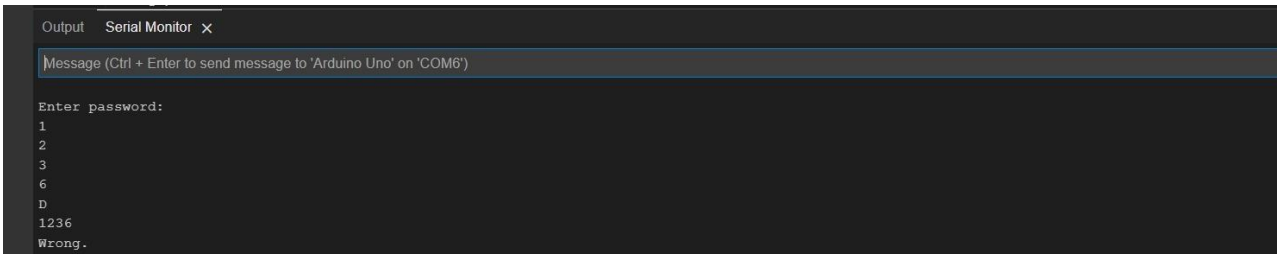
Initially we set up the LCD’s number of columns and rows, implemented the keypad setup and set the pins for the other required components. Once the program is run, the LCD is programmed to display the message "Enter password: ". Moving on, the void loop function has only one function which is the keypadfunction(). Keypadfunction() is a user-defined function which has no return type and does not take any arguments as the input. The getKey() function is used to read the pressed key and is stored in the variable key. The value of each key that is pressed will be added to the password string. If the 'D' key is pressed then it will compare the password entered with the pre-defined password. If the passwords are the same then it will print "Welcome to our house :)" on the LCD, turn on the green led and unlock the door. If a wrong password is entered, it keeps the door locked and increments the number of failed attempts. If the number of wrong attempts is greater than or equal to the value stored in the max_attempts variable (3) then clear the LCD, and announce the user that he is "locked out", while also turning on the red led.

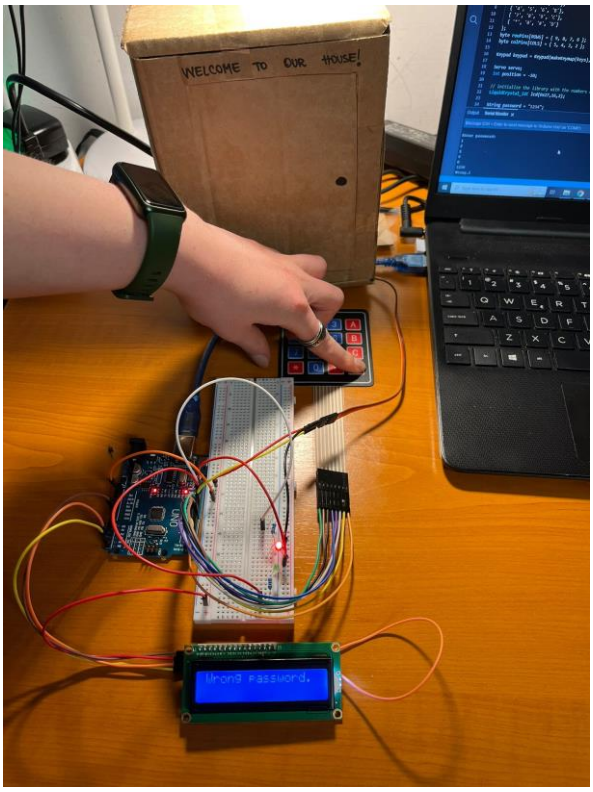
4. Tests and results

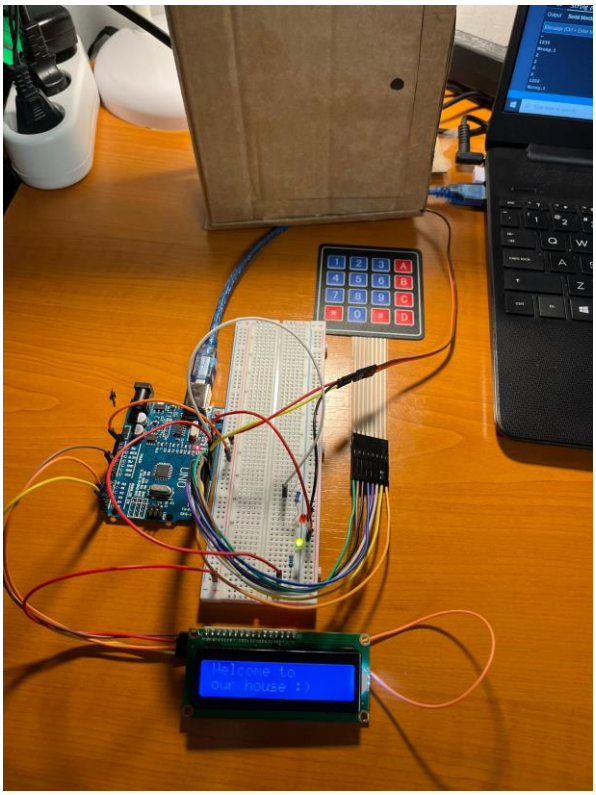
Face recognition:



Door lock system:







5. Conclusion

The initial plan was to combine those two projects and have a smart door that unlocks through a face recognition system. However we encountered a lot of issues with the ESP 32 camera during the process, and being left with no time, decided to showcase as much of what we learned as possible by separating the project.

In conclusion, despite things not going according to plan, we still had a lot to learn from this experience, such as learning to program the ESP32-cam both using an Arduino Uno board and a FTDI module, working with an LCD I2C, programming and using a keypad, and much more.

6. References

1. https://arduinogetstarted.com/tutorials/arduino-lcd-i2c
2. https://arduinogetstarted.com/tutorials/arduino-keypad
3. https://arduinogetstarted.com/tutorials/arduino-servo-motor
4. https://www.instructables.com/Getting-Started-With-ESP32-CAM-Streaming-Video-Usi/
5. https://maker.pro/arduino/projects/how-to-build-an-esp32-based-facial-recognition-system