Muğla Sıtkı Koçman University

EEE-3004 – Microprocessors

GROUP PROJECT

Ahmad Zameer Nazarı – *220702706*

Ahmed Mahmoud Elsayed Hussein – *220702705*

Fevzi Keshta – *210702725*

Mohamed Shawki Eid Elsayed Nejm – *210702723*

16th May 2025

Gesture Detection-Based Security System

With Arduino Uno, OpenCV & Mediapipe

Contents

Overview

This project implements a gesture-based security system using computer vision and Arduino hardware. By combining face detection and hand gesture recognition, the system identifies a user's presence, then authenticates them through a sequence of finger gestures as a password.

A python script processes video in real-time using the OpenCV library and handles face and gesture recognition using Google's open-source machine learning framework, Mediapipe, while communication with an Arduino microcontroller controls LEDs and an LCD display to guide the user through each stage of interaction.

The project developed is a rudimentary implementation of a security system using object detection and recognition technology backed by machine learning, exploring novel ways for non-traditional security systems.

System

Components

The hardware components used in the project:

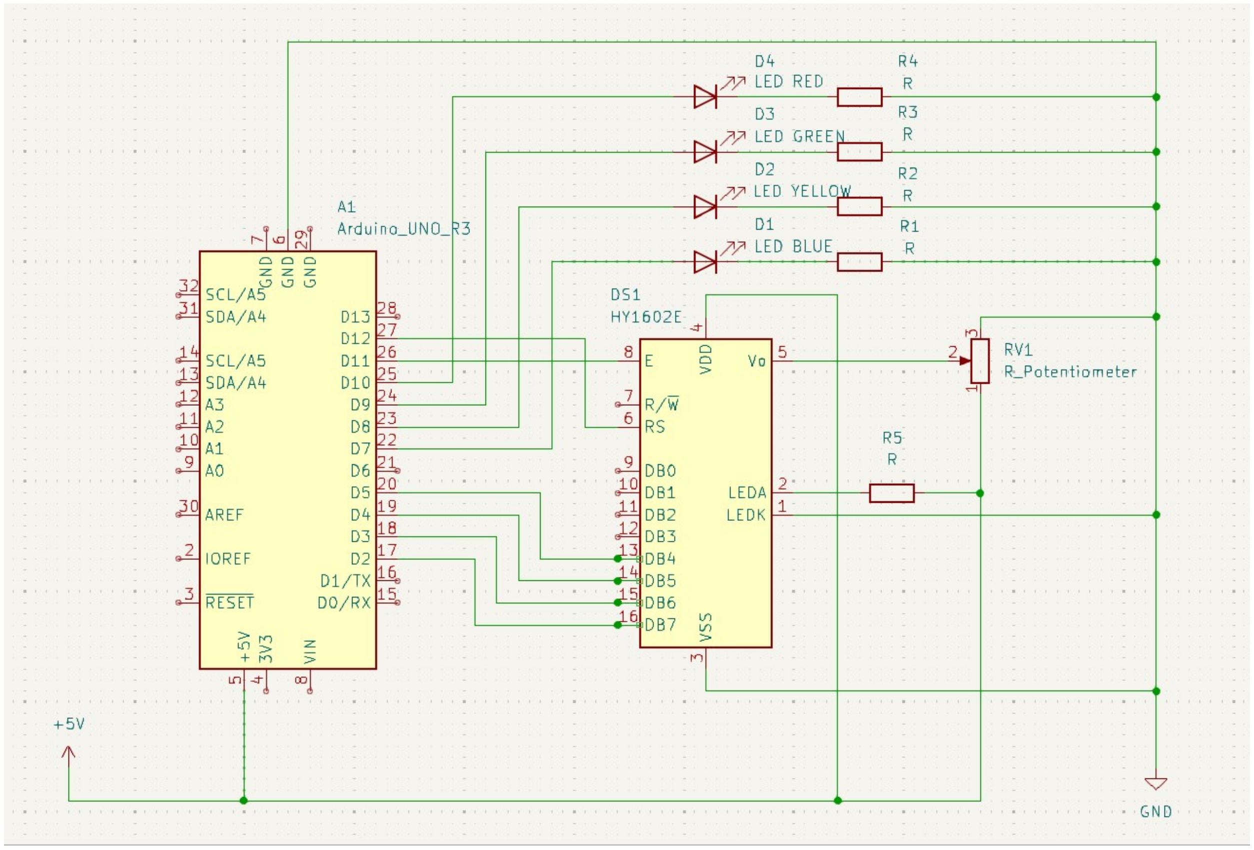
- Arduino Uno R3 for the microcontroller
- 16x2 Liquid Crystal Display (LCD) to display state messages
- 4x LEDs in colors blue, yellow, green and red to indicate different stages
- 5x 220Ω resistors to properly ground some components
- 10kΩ potentiometer to adjust LCD contrast
- Breadboard to assemble all the components
- Some jumper wires for connections, and a USB A-B cable to facilitate connection between a computer and the microcontroller
- A computer with a camera.

Programs and libraries used are:

- The Arduino IDE, using C++ to program the sketch to be uploaded to the Arduino microcontroller.
- An IDE (Visual Studio Code was used) to write the python script that
- The OpenCV python library that processes video fed by the computer's camera
- Google's Mediapipe python library that detects gesture recognition by detecting face and hands
- The serial module allowing python to communicate with the Arduino via USB serial port.

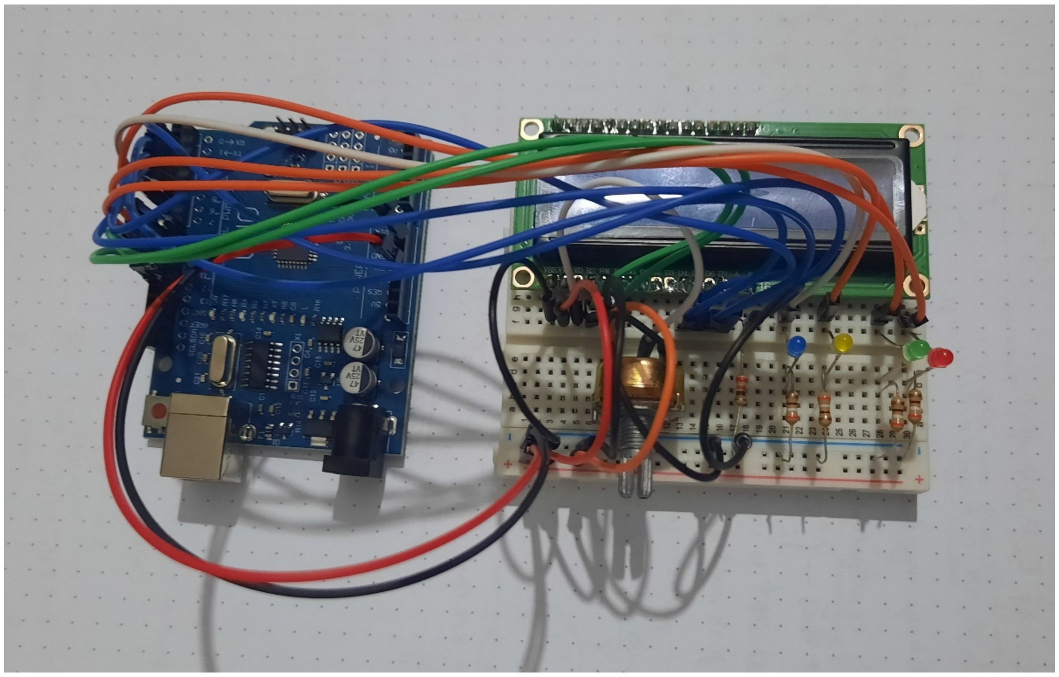
Schematic

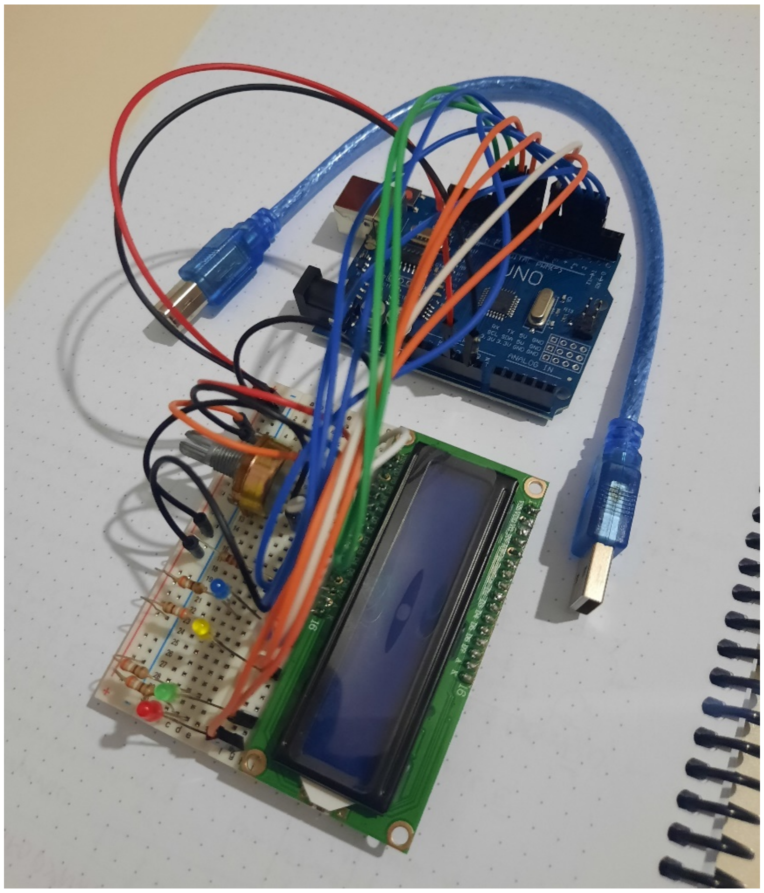
Here is the circuit schematic of the system:



Assembly

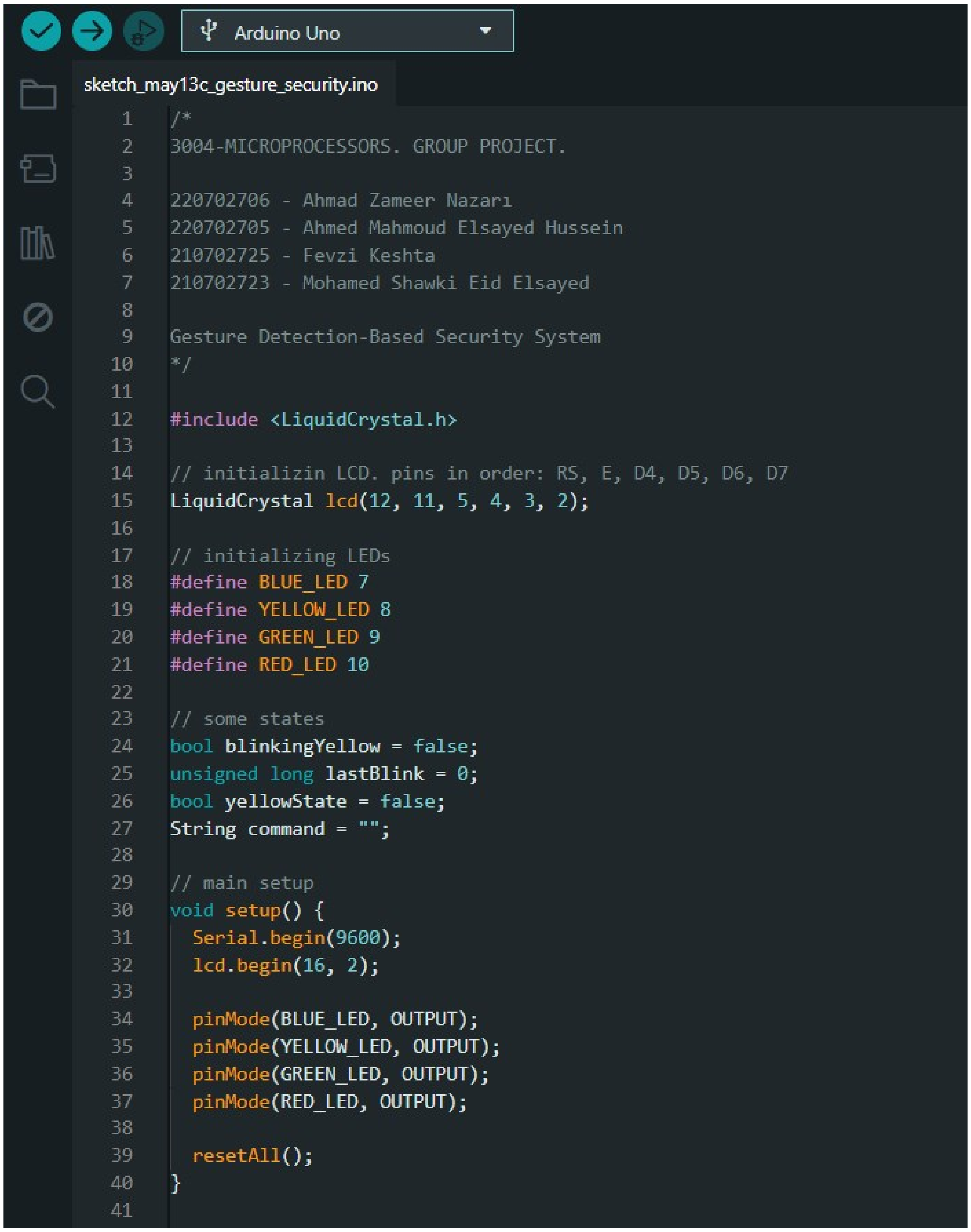
Here is the assembled microcontroller system:



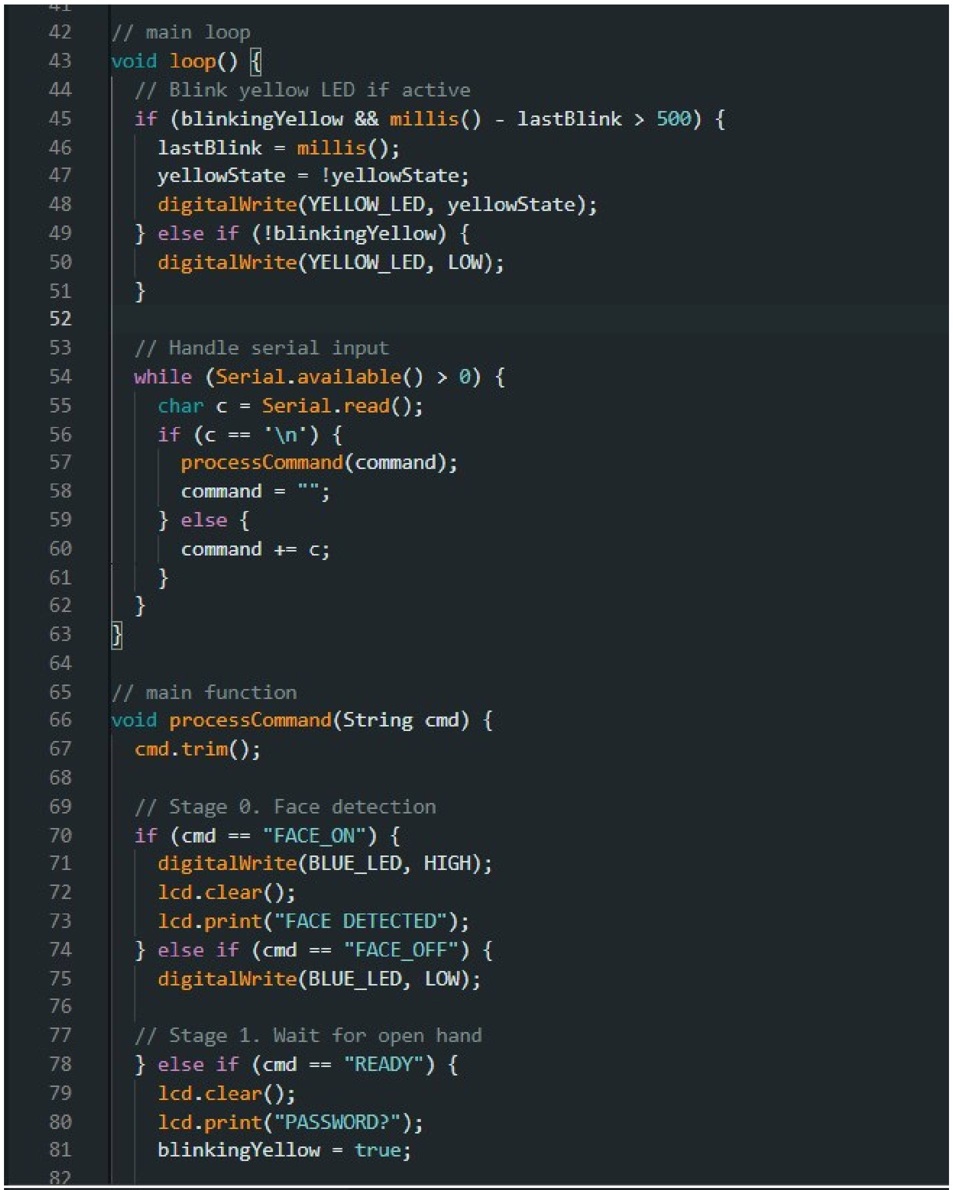


Programming

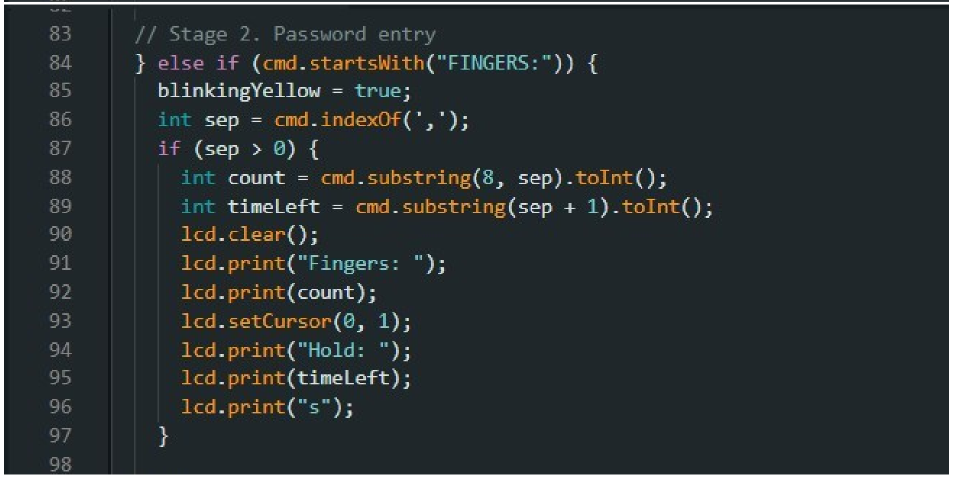
The C++ code that was programmed to the Arduino microcontroller in the Arduino IDE is as follows:



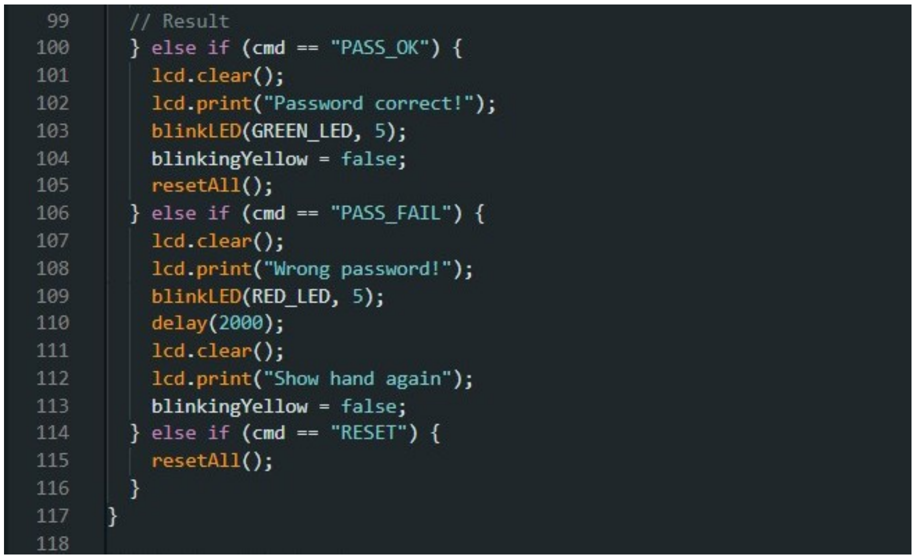
```cpp
/*
3004-MICROPROCESSORS. GROUP PROJECT.

220702706 - Ahmad Zameer Nazarı
220702705 - Ahmed Mahmoud Elsayed Hussein
210702725 - Fevzi Keshta
210702723 - Mohamed Shawki Eid Elsayed

Gesture Detection-Based Security System
*/

#include <LiquidCrystal.h>

// initializin LCD. pins in order: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// initializing LEDs
#define BLUE_LED 7
#define YELLOW_LED 8
#define GREEN_LED 9
#define RED_LED 10

// some states
bool blinkingYellow = false;
unsigned long lastBlink = 0;
bool yellowState = false;
String command = "";

// main setup
void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);

  pinMode(BLUE_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);

  resetAll();
}

```



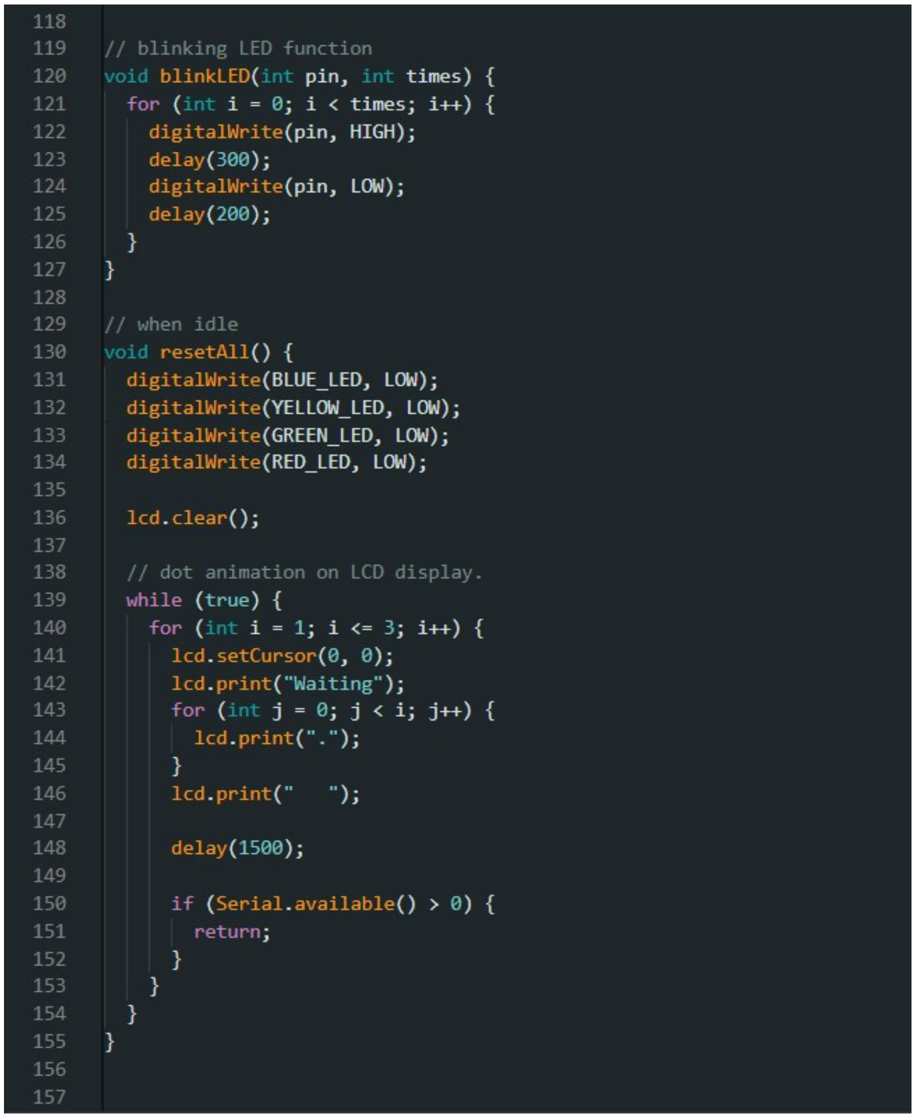
```
// main loop
void loop() {
  // Blink yellow LED if active
  if (blinkingYellow && millis() - lastBlink > 500) {
    lastBlink = millis();
    yellowState = !yellowState;
    digitalWrite(YELLOW_LED, yellowState);
  } else if (!blinkingYellow) {
    digitalWrite(YELLOW_LED, LOW);
  }

  // Handle serial input
  while (Serial.available() > 0) {
    char c = Serial.read();
    if (c == '\n') {
      processCommand(command);
      command = "";
    } else {
      command += c;
    }
  }
}

// main function
void processCommand(String cmd) {
  cmd.trim();

  // Stage 0. Face detection
  if (cmd == "FACE_ON") {
    digitalWrite(BLUE_LED, HIGH);
    lcd.clear();
    lcd.print("FACE DETECTED");
  } else if (cmd == "FACE_OFF") {
    digitalWrite(BLUE_LED, LOW);

  // Stage 1. Wait for open hand
  } else if (cmd == "READY") {
    lcd.clear();
    lcd.print("PASSWORD?");
    blinkingYellow = true;

```



```
  // Stage 2. Password entry
  } else if (cmd.startsWith("FINGERS:")) {
    blinkingYellow = true;
    int sep = cmd.indexOf(',');
    if (sep > 0) {
      int count = cmd.substring(8, sep).toInt();
      int timeLeft = cmd.substring(sep + 1).toInt();
      lcd.clear();
      lcd.print("Fingers: ");
      lcd.print(count);
      lcd.setCursor(0, 1);
      lcd.print("Hold: ");
      lcd.print(timeLeft);
      lcd.print("s");
    }

```

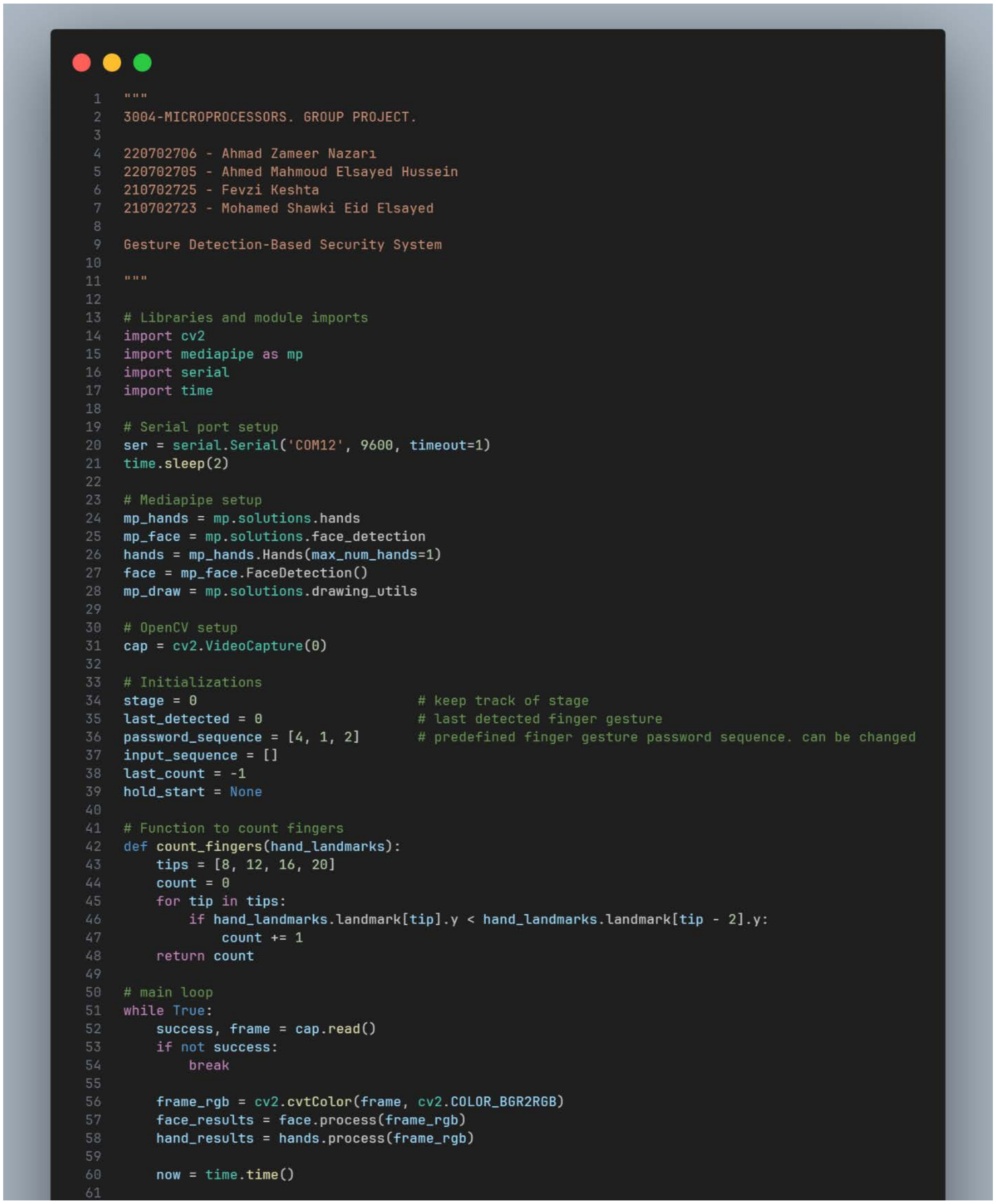


```
    // Result
  } else if (cmd == "PASS_OK") {
    lcd.clear();
    lcd.print("Password correct!");
    blinkLED(GREEN_LED, 5);
    blinkingYellow = false;
    resetAll();
  } else if (cmd == "PASS_FAIL") {
    lcd.clear();
    lcd.print("Wrong password!");
    blinkLED(RED_LED, 5);
    delay(2000);
    lcd.clear();
    lcd.print("Show hand again");
    blinkingYellow = false;
  } else if (cmd == "RESET") {
    resetAll();
  }
}

```

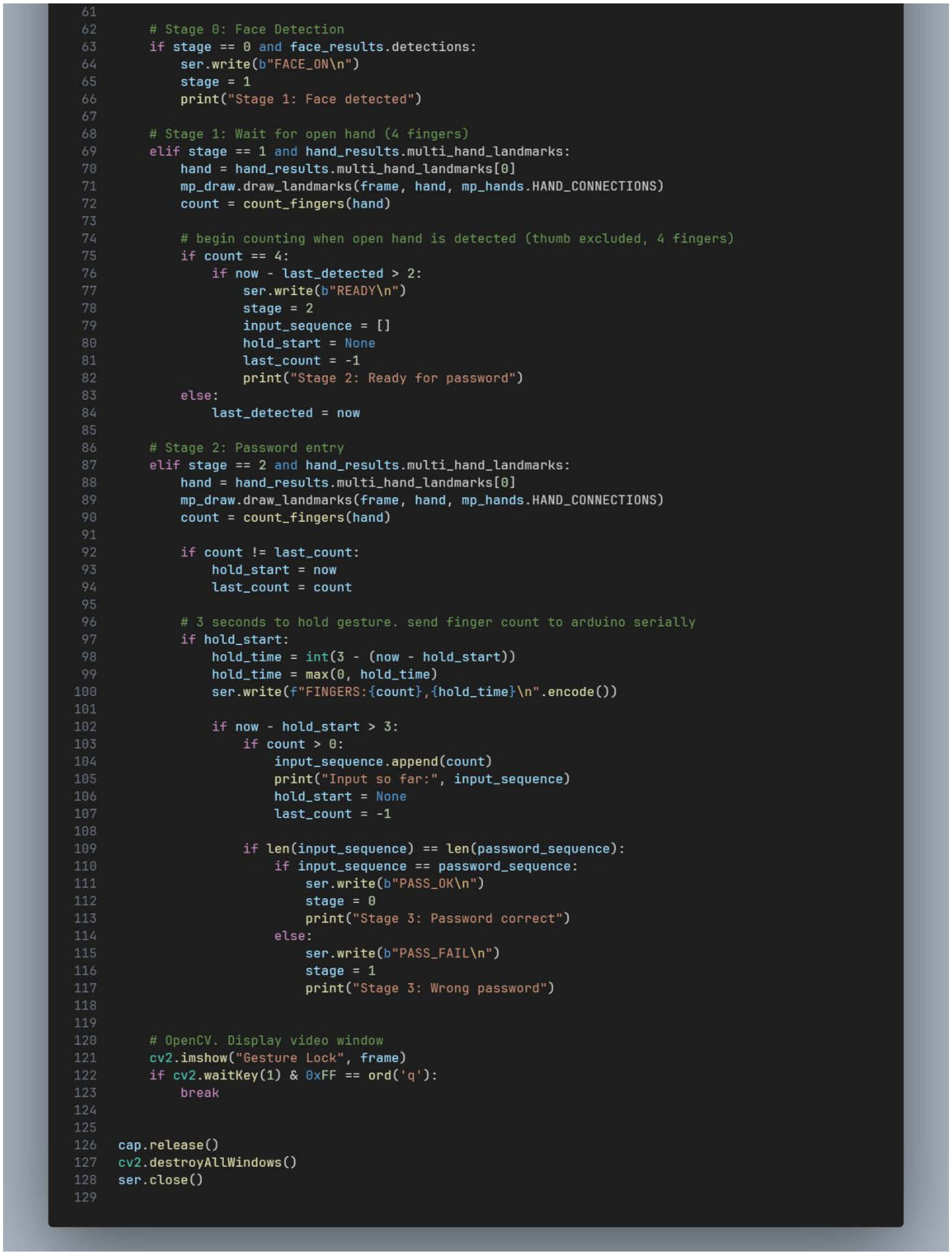


```

// blinking LED function
void blinkLED(int pin, int times) {
  for (int i = 0; i < times; i++) {
    digitalWrite(pin, HIGH);
    delay(300);
    digitalWrite(pin, LOW);
    delay(200);
  }
}

// when idle
void resetAll() {
  digitalWrite(BLUE_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, LOW);
  digitalWrite(RED_LED, LOW);

  lcd.clear();

  // dot animation on LCD display.
  while (true) {
    for (int i = 1; i <= 3; i++) {
      lcd.setCursor(0, 0);
      lcd.print("Waiting");
      for (int j = 0; j < i; j++) {
        lcd.print(".");
      }
      lcd.print("   ");

      delay(1500);

      if (Serial.available() > 0) {
        return;
      }
    }
  }
}


```

The Python script handling facial-detection and gesture-recognition is provided below:



```python
"""
3004-MICROPROCESSORS. GROUP PROJECT.

220702706 - Ahmad Zameer Nazarı
220702705 - Ahmed Mahmoud Elsayed Hussein
210702725 - Fevzi Keshta
210702723 - Mohamed Shawki Eid Elsayed

Gesture Detection-Based Security System

"""

# Libraries and module imports
import cv2
import mediapipe as mp
import serial
import time

# Serial port setup
ser = serial.Serial('COM12', 9600, timeout=1)
time.sleep(2)

# Mediapipe setup
mp_hands = mp.solutions.hands
mp_face = mp.solutions.face_detection
hands = mp_hands.Hands(max_num_hands=1)
face = mp_face.FaceDetection()
mp_draw = mp.solutions.drawing_utils

# OpenCV setup
cap = cv2.VideoCapture(0)

# Initializations
stage = 0                           # keep track of stage
last_detected = 0                   # last detected finger gesture
password_sequence = [4, 1, 2]       # predefined finger gesture password sequence. can be changed
input_sequence = []
last_count = -1
hold_start = None

# Function to count fingers
def count_fingers(hand_landmarks):
    tips = [8, 12, 16, 20]
    count = 0
    for tip in tips:
        if hand_landmarks.landmark[tip].y < hand_landmarks.landmark[tip - 2].y:
            count += 1
    return count

# main loop
while True:
    success, frame = cap.read()
    if not success:
        break

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    face_results = face.process(frame_rgb)
    hand_results = hands.process(frame_rgb)

    now = time.time()

```



```python

    # Stage 0: Face Detection
    if stage == 0 and face_results.detections:
        ser.write(b"FACE_ON\n")
        stage = 1
        print("Stage 1: Face detected")

    # Stage 1: Wait for open hand (4 fingers)
    elif stage == 1 and hand_results.multi_hand_landmarks:
        hand = hand_results.multi_hand_landmarks[0]
        mp_draw.draw_landmarks(frame, hand, mp_hands.HAND_CONNECTIONS)
        count = count_fingers(hand)

        # begin counting when open hand is detected (thumb excluded, 4 fingers)
        if count == 4:
            if now - last_detected > 2:
                ser.write(b"READY\n")
                stage = 2
                input_sequence = []
                hold_start = None
                last_count = -1
                print("Stage 2: Ready for password")
        else:
            last_detected = now

    # Stage 2: Password entry
    elif stage == 2 and hand_results.multi_hand_landmarks:
        hand = hand_results.multi_hand_landmarks[0]
        mp_draw.draw_landmarks(frame, hand, mp_hands.HAND_CONNECTIONS)
        count = count_fingers(hand)

        if count != last_count:
            hold_start = now
            last_count = count

        # 3 seconds to hold gesture. send finger count to arduino serially
        if hold_start:
            hold_time = int(3 - (now - hold_start))
            hold_time = max(0, hold_time)
            ser.write(f"FINGERS:{count},{hold_time}\n".encode())

            if now - hold_start > 3:
                if count > 0:
                    input_sequence.append(count)
                    print("Input so far:", input_sequence)
                    hold_start = None
                    last_count = -1

                if len(input_sequence) == len(password_sequence):
                    if input_sequence == password_sequence:
                        ser.write(b"PASS_OK\n")
                        stage = 0
                        print("Stage 3: Password correct")
                    else:
                        ser.write(b"PASS_FAIL\n")
                        stage = 1
                        print("Stage 3: Wrong password")


    # OpenCV. Display video window
    cv2.imshow("Gesture Lock", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
ser.close()

```

Operation

The entire system operation is divided into three principal stages.

Stage 0:

The system is first in the pending stage, waiting for a person to come into the camera view. The LCD displays "`Waiting…`"

Stage 1:

When a person comes into view, a face is detected, and the system moves to Stage 1. The blue LED turns on, and the LCD displays the text, "`Face Detected`" The user is prompted to show an open palm with all five fingers visible. Holding this gesture for a few seconds, the system transitions to password mode.

Stage 2:

At this stage, a yellow LED starts blinking and the LCD prompts "`Password?`"

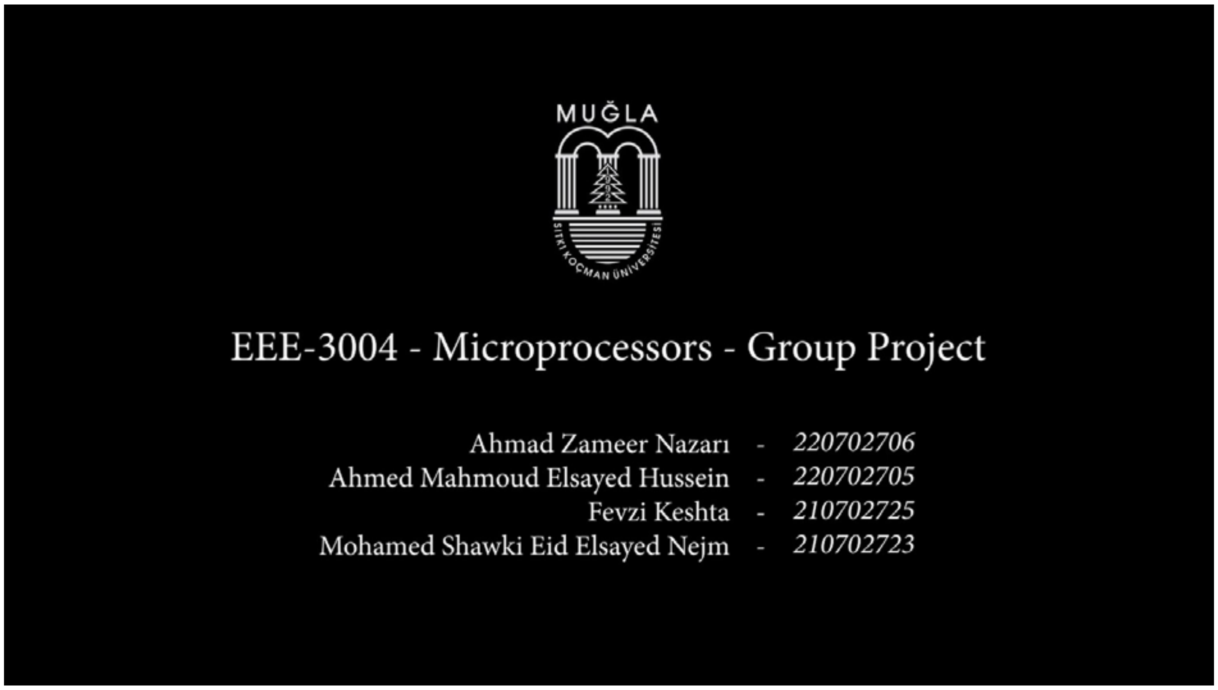
The user is requested to input a predefined sequence of finger counts (e.g. 4, 1, 2), each held for a duration of a few seconds.

If the sequence is correct, a green LED blinks and the LCD displays "`Password Correct!`". If incorrect, a red LED blinks, the LCD displays "`Wrong Password!`" and the system resets, waiting for the open-palm gesture again.

Demonstration

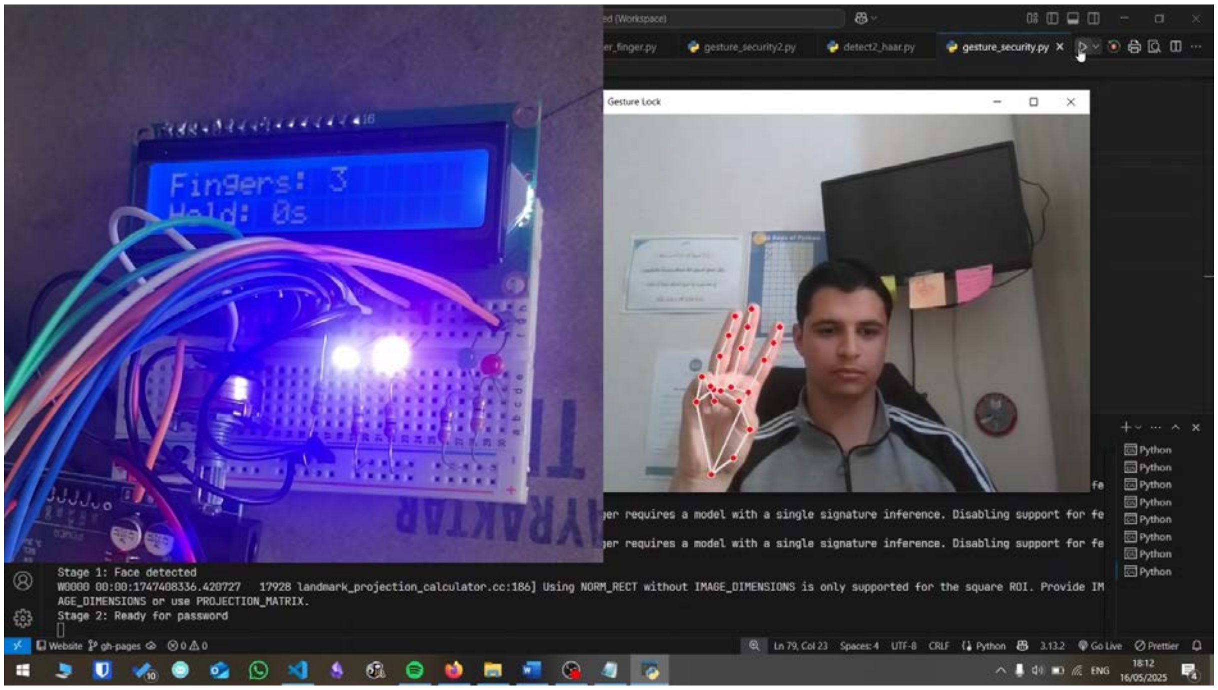
A video demonstrating the system has been uploaded to YouTube with the following URL:

https://youtu.be/pU1nFCVVoqo



EEE-3004 - Microprocessors - Group Project

Ahmad Zameer Nazarı	-	*220702706*
Ahmed Mahmoud Elsayed Hussein	-	*220702705*
Fevzi Keshta	-	*210702725*
Mohamed Shawki Eid Elsayed Nejm	-	*210702723*



```
ger requires a model with a single signature inference. Disabling support for fe
ger requires a model with a single signature inference. Disabling support for fe
Stage 1: Face detected
W0000 00:00:1747408336.420727   17928 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only supported for the square ROI. Provide IM
AGE_DIMENSIONS or use PROJECTION_MATRIX.
Stage 2: Ready for password
```

All files available at the repository:

https://github.com/az-yugen/EEE-3004-Microprocessors