Employee Access Control System

Group 4

111598095 陳雅音
112598017 楊淨雯
112054340 羅際東
112598005 徐楊越

2023/12/31

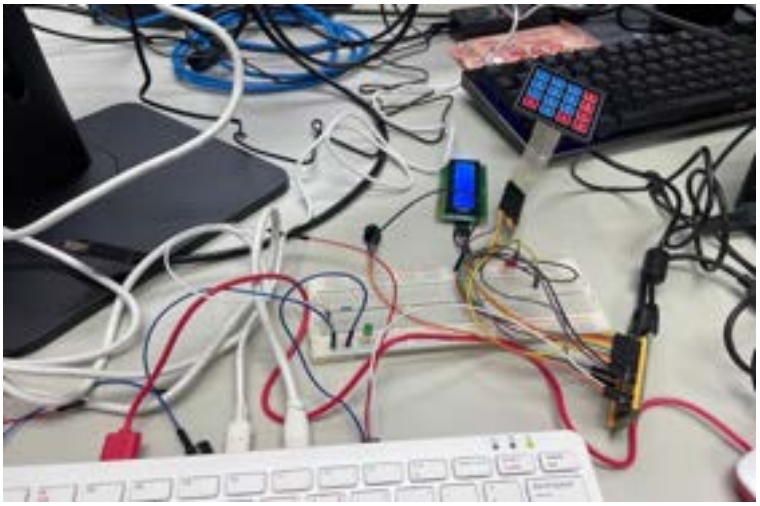
Introduction

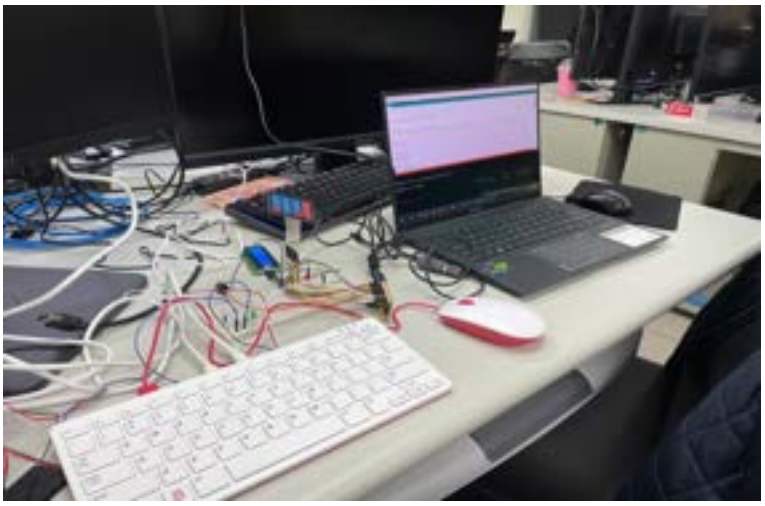
Our objective is to control and manage the access of individuals to specific areas, ensuring the security of the premises and protecting assets.

User Side:
Employees use the keypad to input passwords, and the entered password is displayed on the LCD screen. If the password is incorrect, the buzzer will sound for 0.5 seconds as a reminder, and "Incorrect" will be displayed on the LCD screen. If the password is correct, "Correct" will be displayed.

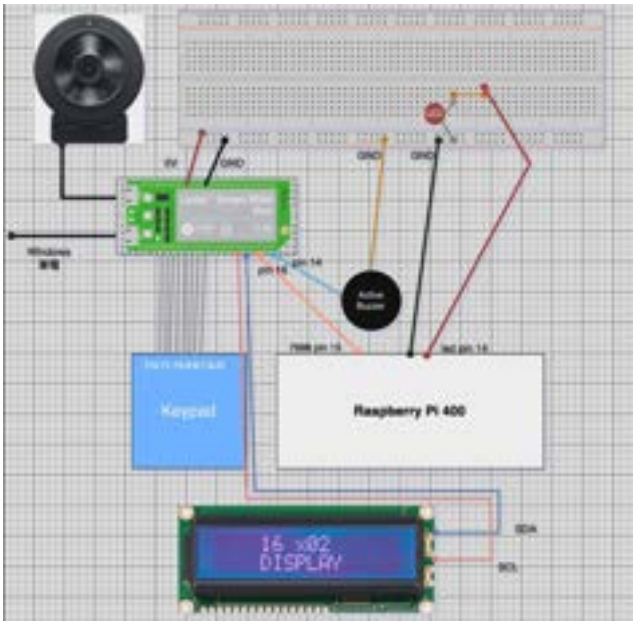
Management Side:
Managers can use the web interface to browse real-time video images through streaming services to monitor individuals entering passwords. They can also use the on/off input button on the web interface to activate the buzzer and drive away unauthorized individuals.

Physical Diagram:





Schematic Diagram of Component Connections:



Raspberry Pi 400

Keypad: 4*4

Active Buzzer

LCD1602 IIC

LED

Webcam: RAZER KIYO X

Challenge

1.The built-in Python version is 2.7, and both pip and setup package versions are very low, making it impossible to install other software packages or update Python, pip, and other packages. Many packages currently only support Python 3.0 and above. Therefore, we can only use the built-in mraa for control.

2.When using CGI Python, redirecting to another webpage is more complicated, requiring the direct printing of HTML structure. Therefore, later on, we moved the webpage to Raspberry Pi 400. The Linkit 7688 Duo is responsible only for the terminal's receiving and transmitting of data.

3.Regarding Arduino, if the Arduino IDE is installed with the latest version, it will not be compatible. MediaTek's support cannot be found in the Board Manager. Later on, we found pre-arranged JSON and ZIP files on GitHub for installation in Arduino IDE version 1.6.5, which works for program compilation.

4.The MQTT WiFi package, due to version issues, cannot support connecting to a mobile hotspot. Therefore, we later modified it to directly connect the Linkit Smart 7688 Duo to the Pi 400 for communication, achieving the goal of signal transmission.

Environment

Hardware Refence:

Hardware
Raspberry Pi 400
Linkit Smart 7688 Duo

Software Version Reference:

Software	Version	Github	Website
Arduino IDE	1.6.5		https://www.arduino.cc/en/software
Adafruit_Keypad	1.3.2	https://github.com/adafruit/Adafruit_Keypad	https://blog.jmaker.com.tw/arduino-keypad-4x4/
Wire	1.0.0		https://www.arduino.cc/reference/en/language/functions/communication/wire/
LiquidCrystal_I2C	1.1.2	https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library	https://reference.arduino.cc/reference/en/libraries/liquidcrystal-i2c/
Python	2.7		https://www.python.org/

Process

Getting Started:
Since the preliminary environment setup is the same for both the 7688 and 7688 Duo, we will use the readily available 7688 Duo as the operational example.

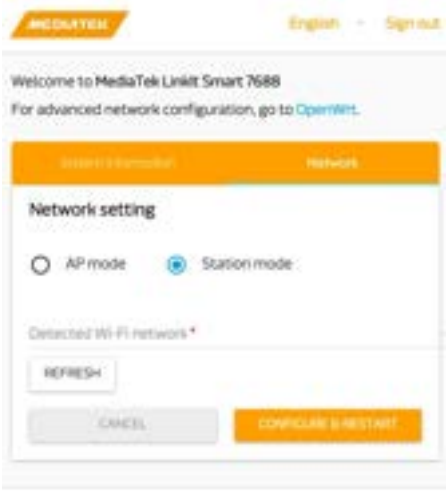
Step 1: Set up the 7688 to connect to your designated Wi-Fi network.
Follow the steps below to import the example:

1. Connect the 7688 to your PC using the Micro USB cable. At this point, the 7688 is in AP mode, allowing your computer to connect.

2. After powering on the 7688, if it is the first time using it, the WiFi is in AP mode. You can connect from your computer to the shared SSID, which will be similar to Linkit_Smart_7688.

3. Open a web browser and enter the URL http://mylinkit.local to access the login page. For the first entry, you will need to set up a login password.

4. Change the 7688 to Station mode, connect to your specified WiFi network, and restart.



Step 2: Set Up the 7688 SSH

Connect your computer to the same WiFi network as the 7688. Open SSH connection program. If you are a MAC user, you can directly open the Terminal application. If you are a Windows user, you can download the Putty. Then, use the following command:

```
sudo ssh root@mylinkit.local
```

After successful login, the following screen will appear.

Step 3: Set up SFTP

The vanilla OpenWrt out of the box has a small Dropbear SSH server. But it doesn't support the SFTP and you need to install openssh-sftp-server package. The package comes from another OpenSSH server which is bigger but has more features and default on desktop systems like Ubuntu. Many routers with OpenWrt as a stock firmware use it out of the box and the openssh-sftp-server is installed too. Thus, for this routers you really don't need anything to do and just start using it. But if not then install the required package:

```
opkg update

opkg install openssh-sftp-server
```

Step 4: Set up mjpg_streamer

Connect the Webcam: RAZER KIYO X to the Linkit smart 7688 Duo and type the command below.

```
mjpg_streamer -i "input_uvc.so -f 20 -d /dev/video0" -o "output_http.so"
```

Option Description:

- i: input_xxxx.so executes input plugins
- f: fps, how many pictures to run per second, set the frame rate under FLASH, it is recommended to set 20 for more stability
- d: /dev/video0 is the device file of the USB Webcam. It is automatically generated after connecting the USB Webcam to the Arduino Yun.
- o: output_xxxx.so is the plugins that perform output
- r: Resolution, you can use the following string to directly enter QSIF QCIF CGA QVGA CIF VGA SVGA XGA SXGA, or you can also enter it manually, for example: 1024*768

Finally, open the browser and enter http://mylinkit.local:8080?action=stream to easily

perform live video streaming

Step 5: Connect 7688 Duo with Keypad
Connect the 4*4 keypad to the 7688 Duo, which occupies a total of 8 pins from pin 5 to pin 12, and download the Adafruit_Keypad kit from the Arduino library. Open a keypad example code from the library and set up the rows, columns, pins, and the name of each key on the keypad.

Step 6: Connect 7688 Duo with LCD
Connect the LCD1602IIC to the 7688 Duo, which occupies pin SDA and SCL, and download the LiquidCrystal_I2C kit from the Arduino library. Open an LCD example code from the library and set up init, backlight, and cursor to (0,0).

Step 7: Set up active buzzer
Connect the active buzzer to 7688 Duo on pin 14. In the setup, use pinMode to set pin 14 to output and digitalWrite to set buzzer low. When the user's password is entered incorrectly, use digitalWrite to make pin 14 output HIGH, and then use delay(500) to make the buzzer sound for 0.5 seconds.

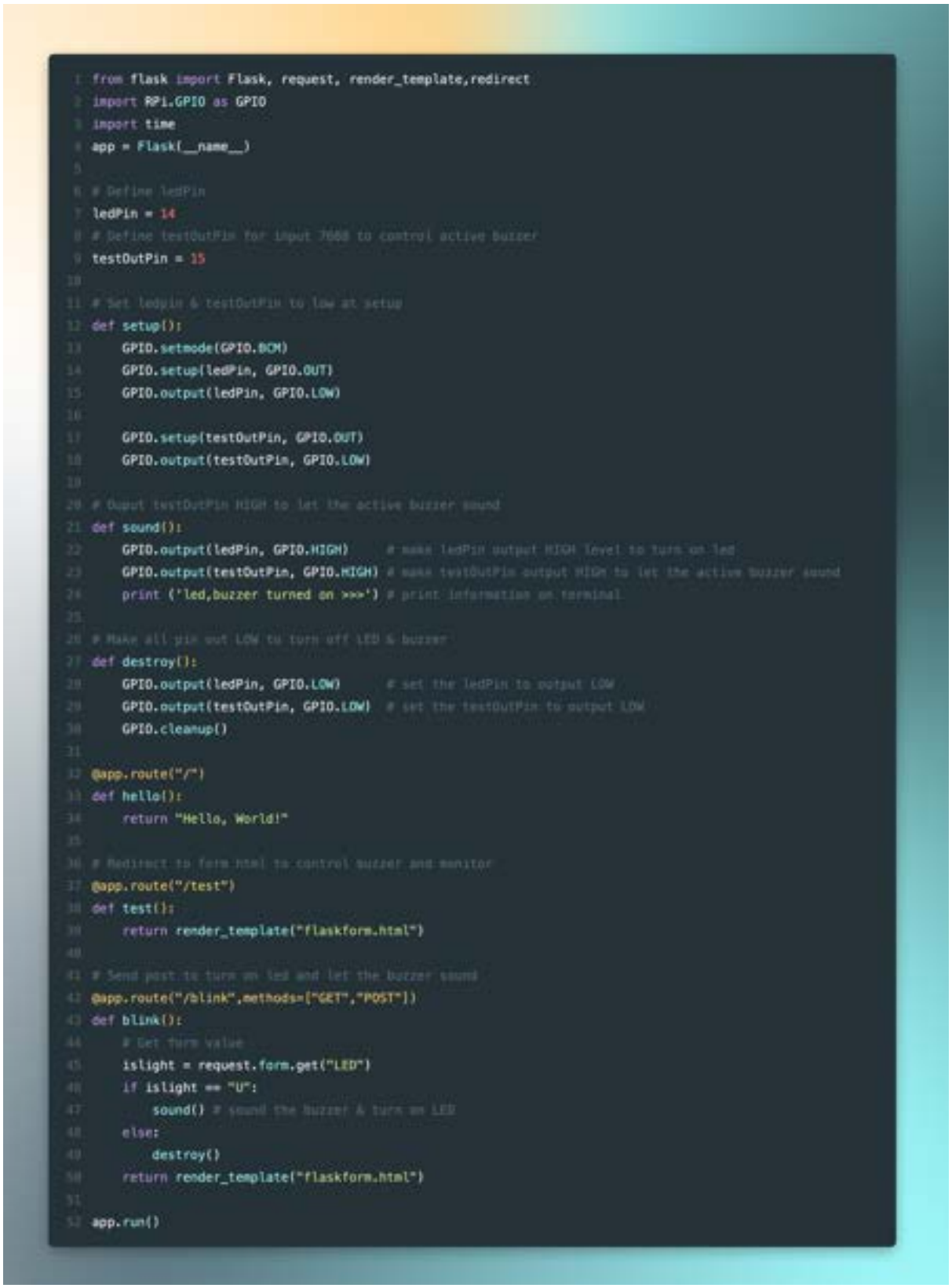
Step 8: Connect Linkit smart 7688 Duo with Raspberry Pi 400
Connect pin 16 of 7688 Duo to pin 5 of Pi 400 as signal transmission. When the administrator chooses to turn on the light, the HIGH signal is output from pin16 to pi 400. After pi 400 receives 1/0, it will perform the corresponding program.

Step 9: Set up LED
Connect pin 14 of Pi 400 as LED indicator for successful transmission of the on/off command.

Code

Running on Raspberry Pi 400:

1. Python:



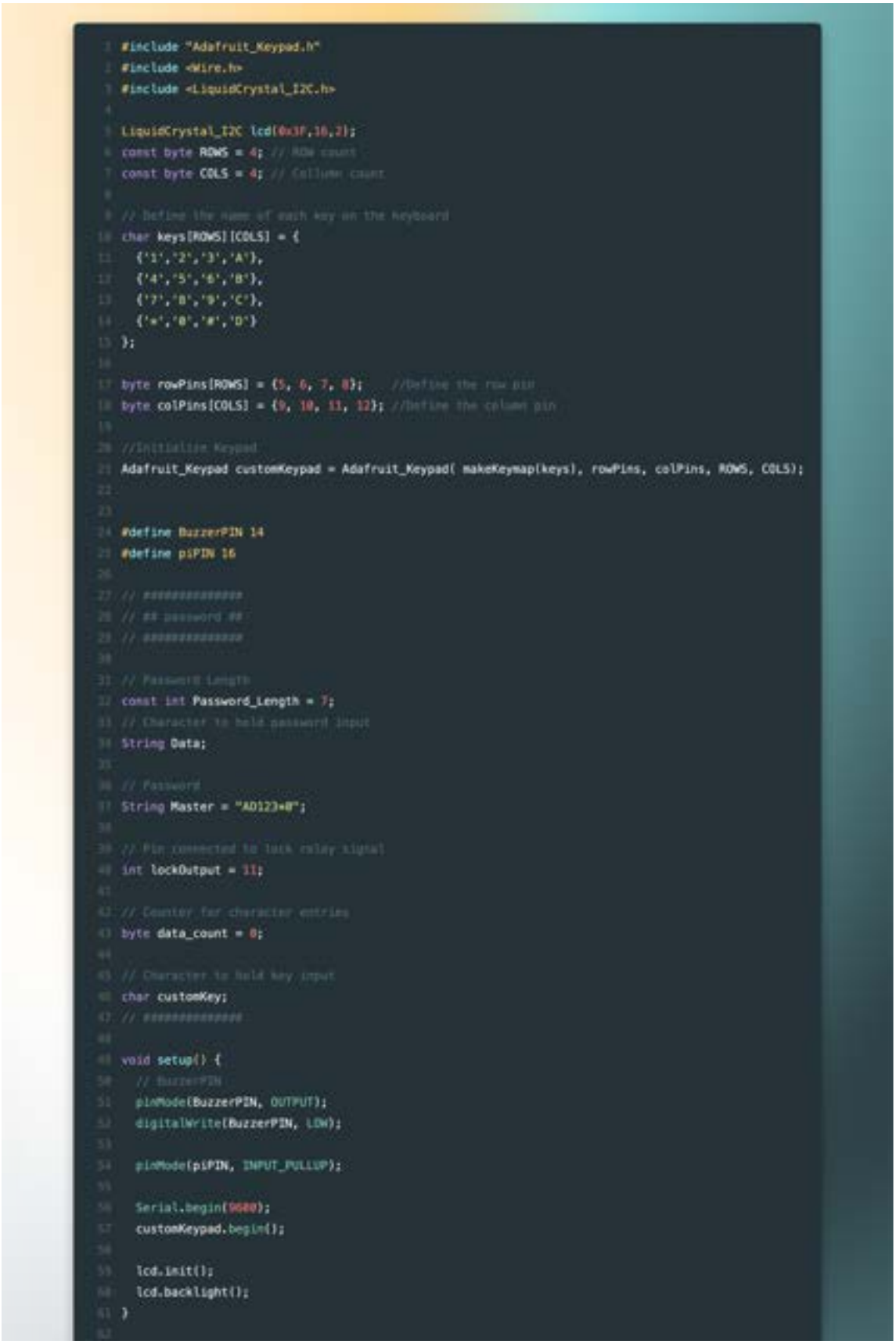
```python
from flask import Flask, request, render_template,redirect
import RPi.GPIO as GPIO
import time
app = Flask(__name__)

# Define ledPin
ledPin = 14
# Define testOutPin for input 7688 to control active buzzer
testOutPin = 15

# Set ledpin & testOutPin to low at setup
def setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(ledPin, GPIO.OUT)
    GPIO.output(ledPin, GPIO.LOW)

    GPIO.setup(testOutPin, GPIO.OUT)
    GPIO.output(testOutPin, GPIO.LOW)

# Ouput testOutPin HIGH to let the active buzzer sound
def sound():
    GPIO.output(ledPin, GPIO.HIGH)     # make ledPin output HIGH level to turn on led
    GPIO.output(testOutPin, GPIO.HIGH) # make testOutPin output HIGH to let the active buzzer sound
    print ('led,buzzer turned on >>>') # print information on terminal

# Make all pin out LOW to turn off LED & buzzer
def destroy():
    GPIO.output(ledPin, GPIO.LOW)      # set the ledPin to output LOW
    GPIO.output(testOutPin, GPIO.LOW)  # set the testOutPin to output LOW
    GPIO.cleanup()

@app.route("/")
def hello():
    return "Hello, World!"

# Redirect to form html to control buzzer and monitor
@app.route("/test")
def test():
    return render_template("flaskform.html")

# Send post to turn on led and let the buzzer sound
@app.route("/blink",methods=["GET","POST"])
def blink():
    # Get form value
    islight = request.form.get("LED")
    if islight == "U":
        sound() # sound the buzzer & turn on LED
    else:
        destroy()
    return render_template("flaskform.html")

app.run()
```

2. HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Any kinds of test</title>
</head>
<body>
    <form method='post'action="{{ url_for("blink")}}">
        開起led燈泡和蜂鳴器 :<br>
        <input type="radio" name="LED" value="U">ON<br>
        <input type="radio" name="LED" value="OFF">OFF<br>
        <input type="submit" value="Submit" />
    </form>
    <img src="http://192.168.43.205:8080/?action=stream">
</body>
</html>
```

Running on Linkit smart 7688 Duo:

1. Arduino



```
#include "Adafruit_Keypad.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2);
const byte ROWS = 4; // Row count
const byte COLS = 4; // Column count

// Define the name of each key on the keyboard
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

byte rowPins[ROWS] = {5, 6, 7, 8};    //Define the row pin
byte colPins[COLS] = {9, 10, 11, 12}; //Define the column pin

//Initialize Keypad
Adafruit_Keypad customKeypad = Adafruit_Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS);


#define BuzzerPIN 14
#define piPIN 16

// ##############
// ## password ##
// ##############

// Password Length
const int Password_Length = 7;
// Character to hold password input
String Data;

// Password
String Master = "AD123#B";

// Pin connected to lock relay signal
int lockOutput = 11;

// Counter for character entries
byte data_count = 0;

// Character to hold key input
char customKey;
// ##############

void setup() {
  // BuzzerPIN
  pinMode(BuzzerPIN, OUTPUT);
  digitalWrite(BuzzerPIN, LOW);

  pinMode(piPIN, INPUT_PULLUP);

  Serial.begin(9600);
  customKeypad.begin();

  lcd.init();
  lcd.backlight();
}
```

```
void loop() {

  // Initialize LCD and print
  lcd.setCursor(0, 0);
  lcd.print("Enter Password:");

  customKeypad.tick();
  while (customKeypad.available())
  {
    // Enter keypress into array and increment counter
    keypadEvent e = customKeypad.read();
    customKey = (char)e.bit.KEY;
    if (e.bit.EVENT == KEY_JUST_RELEASED) {
      Data += customKey;
      lcd.setCursor(data_count, 1);
      lcd.print(Data[data_count]);
      data_count++;
      break;
    }
  }


  // See if we have reached the password length
  if (data_count == Password_Length) {
    lcd.clear();
    Serial.println("Password: ");
    Serial.print(Data);

    if (Data == Master) {
      // Correct Password
      lcd.print("Correct");
      // Turn on relay for 5 seconds
      // digitalWrite(BuzzerPIN, HIGH);
      delay(500);
      // digitalWrite(BuzzerPIN, LOW);
    }
    else {
      // Incorrect Password
      lcd.print("Incorrect");
      digitalWrite(BuzzerPIN, HIGH);

      delay(500);
      digitalWrite(BuzzerPIN, LOW);

    }

    // Clear Data and LCD display
    lcd.clear();
    clearData();
  }


  int piState = digitalRead(piPIN);
  Serial.println(piState);
  if (piState == HIGH){
     digitalWrite(BuzzerPIN, HIGH);
     delay(100);
     digitalWrite(BuzzerPIN, LOW);
    }
   else{
     digitalWrite(BuzzerPIN, LOW);
    }
}


void clearData() {
  //Reset data_count
  data_count = 0;
  //Reset Data
  Data ="";
}
```

Evaluation Result

In terms of achievements, we have accomplished more than 90%. However, it is undeniable that there are still some minor bugs present. For example, after shooting a recent video, certain functionalities that were working before now encounter unexpected issues upon reconnection. Issues such as the buzzer not turning off or the inability to receive signals that were previously working.

Demo video - 1	https://youtube.com/shorts/u9iraENu_ps
Demo video - 2	https://youtu.be/uD_uT3pB8uw

Conclusion

We can extend our project to include features such as recording video when someone enters the wrong password at the door, or setting up a sentry mode like Tesla, where the system activates video recording when someone enters the LAB. It should save power when there's no one around, and the buzzer should directly alert campus security.