# Assignment 02 – 102533320

## Video Demonstration –

https://www.youtube.com/watch?v=-LVt01sA7Fw

## Summary –

As society continues to transition to an age of interconnected devices and home security becomes more of a necessity, the introduction of smart security devices is crucial. Through multiple layers of intuitive secure systems, security devices allow users to protect themselves and any possessions from any malicious individuals. As a result, the IoT device I have created revolves around this concept, comprising of two smart security layers that are passcode protected. This device is a foundation for more comprehensive systems, ensuring users can withhold their valuables with a reliable and interchangeable passcode secured layer of systems.

### IoT Node –

Consequently, the IoT node that I have assumed is comprised of a series of actuators and sensors that intuitively collaborate to present users with two passcode protected barriers that must be correctly surpassed. The first barrier consists of a potentiometer that uses segmented values that are accessed through turning the dial within the corresponding region. These segments are separated into four quarters, defined by conditions based off the range values that are collected. As a result, each segment contains a unique letter (A, E, F, S) allowing users to input a series of combinations that cater to a four-letter word. Displayed by the 7-segment LED display, with each letter briefly appearing as the user selects their respective segments. Additionally, this selection process is dictated through the implementation of an IR sensor and IR remote, confirming each input through the press of the power button on the IR remote. Consequently, if the combination of letters results in the incorrect word, the user is presented with a red LED flash. Furthermore, if the letter combination results in the word ‘SAFE’, the user is then instead presented with a green LED flash and the sound of a buzzer, allowing users to progress to the next security layer. This next challenge is similar in scope to the last section but instead utilizes inputs dictated by the user pressing digits on the IR remote. Once a series of 4-digit codes result in the passcode ‘9276’, the user is then presented with the green and red LED continuously flashing simultaneously while the buzzer is activated, allowing the user to access any theoretical valuables behind these security systems.

### Edge Device

Consequently, the Arduino board is also connected to an edge device that is emulated within a Virtual Machine using Linux Debian as its OS, capable of receiving data projected to the serial monitor and writing data to the Arduino board to dictate if conditional rules are met. As a result, the edge device reads the output values that the serial monitor is presented with, where values consist of each password attempt the user makes. Each entered passcode is then checked against the correct passcode, causing the edge server to write a value to the Arduino board dependent on the outcome, dictating how the actuators will react.

## Implementation –

IR Sensor (Digital Sensor) –

- The IR sensor allows users to both confirm entered potentiometer values and input digits with the press of specific buttons on the IR remote. Each button press on the IR remote corresponds to a unique hexadecimal code that the IR sensor collects. These hexadecimal codes then dictate which actions will occur.

Potentiometer (Analog Sensor) –

- The Potentiometer allows users to turn the dial to the corresponding segment that has been defined by its range values, allowing users to enter a letter based off these values. The value of the dial is measured constantly to keep track of how far the dial has been turned. This allows our Arduino program to define conditions that separate each segment into four unique quarters, ensuring collected potentiometer values can be tested against these conditions and result in different letters being output.

Green LED and Red LED (Actuator) –

- Both LEDs indicate if the user has inputted the password correctly. This action occurs if the data collected from serial monitor from the edge server presents a string that satisfies a condition, the edge server would then send a letter code to the Arduino board and dictate which LED would flash / if the user inputted the correct passcode.

Buzzer (Actuator) –

- The buzzer functions the exact same as the green LED, activating if the green LED conditions are met. This indicates that the user inputted the correct passcode.

8-Segment LED (Actuator) –

- The 8-segment LED presents the user with the values they inputted. These values appear in both security layers, as both the letters and digits are capable of being presented to the user. This allows the user to gain visual of what exactly they are inputting into the system.

IR Remote (Actuator) –

- The IR remote allows the user to input and send values to the IR sensor to successfully collect the corresponding hexadecimal codes.

Arduino Libraries –

- IRremote
- SevSeg

Python Libraries –

- Serial
- Time
- Pymysql

**Resources –**


https://www.youtube.com/watch?v=Akr-aNIEb7U

https://www.circuitbasics.com/arduino-7-segment-display-tutorial/

https://www.youtube.com/watch?v=FEq-w6FDRVM

https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink

https://roboticsbackend.com/arduino-potentiometer-complete-tutorial/

https://en.wikichip.org/wiki/seven-segment_display/representing_letters

https://realpython.com/arduino-python/

https://problemsolvingwithpython.com/11-Python-and-External-Hardware/11.03-Controlling-an-LED/

https://www.arduino.cc/reference/en/language/structure/control-structure/dowhile/

https://www.tutorialspoint.com/python_data_access/python_mysql_insert_data.htm

https://pynative.com/python-cursor-fetchall-fetchmany-fetchone-to-read-rows-from-table/

## Appendices –

Official_Program.ino :

```
#include <IRremote.h>
#include <SevSeg.h>

int RECV_PIN = 10;
int redPin = 11;
int greenPin = 12;
int buzzer = 13;
int analogValue = A0;
unsigned long key_value = 0;
String input_passcode1;
String input_passcode2;
String value;
bool j = false;
int incomingByte;
IRrecv irrecv(RECV_PIN);
decode_results results;
SevSeg sevseg;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
  byte numDigits = 1;
  byte digitPins[] = {};
  byte segmentPins[] = {6, 5, 2, 3, 4, 7, 8, 9};
  bool resistorsOnSegments = true;
  byte hardwareConfig = COMMON_ANODE;
  sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegments);
  sevseg.setBrightness(90);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(buzzer, OUTPUT);
}

void loop() {
  // Edge Server Check
  edgeServerCheck();

  // Barrier 01
  int sensorValue = analogRead(analogValue);
  if (irrecv.decode(&results)) {
    key_value = results.value;
    if (results.value == 0xFD00FF) {
      if (sensorValue >= 0 && sensorValue <= 20) {
        input_passcode1 += "A";
        sevseg.setChars("A");
      } else if (sensorValue >= 250 && sensorValue <= 400) {
        input_passcode1 += "E";
        sevseg.setChars("E");
      } else if (sensorValue >= 500  && sensorValue <= 650) {
        input_passcode1 += "S";
        sevseg.setChars("S");
      } else if (sensorValue >= 800 && sensorValue <= 950) {
        input_passcode1 += "F";
        sevseg.setChars("F");
```

```
    }
    sevseg.refreshDisplay();
    delay(300);
    sevseg.blank();
  }
  irrecv.resume();
}
if (input_passcode1.length() == 4) {
  Serial.println(input_passcode1);
  input_passcode1 = "";
}

// Barrier 02
while (j == true) {
  edgeServerCheck();
  if (irrecv.decode(&results)) {
    switch (results.value) {
      case 0xFD30CF:
        sevseg.setNumber(0);
        value = "0";
        break;
      case 0xFD08F7:
        sevseg.setNumber(1);
        value = "1";
        break;
      case 0xFD8877:
        sevseg.setNumber(2);
        value = "2";
        break;
      case 0xFD48B7:
        sevseg.setNumber(3);
        value = "3";
        break;
      case 0xFD28D7:
        sevseg.setNumber(4);
        value = "4";
        break;
      case 0xFDA857:
        sevseg.setNumber(5);
        value = "5";
        break;
      case 0xFD6897:
        sevseg.setNumber(6);
        value = "6";
        break;
      case 0xFD18E7:
        sevseg.setNumber(7);
        value = "7";
        break;
      case 0xFD9867:
        sevseg.setNumber(8);
        value = "8";
        break;
      case 0xFD58A7:
```

```
          sevseg.setNumber(9);
          value = "9";
          break;
        default:
          break;
      }
      key_value = results.value;
      input_passcode2 += value;
      irrecv.resume();
      sevseg.refreshDisplay();
      delay(300);
      sevseg.blank();
    }
    if (input_passcode2.length() == 4) {
      Serial.println(input_passcode2);
      input_passcode2 = "";
    }
  }
}

void edgeServerCheck() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    if (incomingByte == 'A') {
      j = true;
      digitalWrite(greenPin, HIGH);
      digitalWrite(buzzer, HIGH);
      delay(500);
      digitalWrite(greenPin, LOW);
      digitalWrite(buzzer, LOW);
    } else if (incomingByte == 'B') {
      while (j == true) {
        digitalWrite(buzzer, HIGH);
        digitalWrite(greenPin, HIGH);
        delay(100);
        digitalWrite(greenPin, LOW);
        digitalWrite(redPin, HIGH);
        delay(100);
        digitalWrite(redPin, LOW);
      }
    } else {
      digitalWrite(redPin, HIGH);
      delay(500);
      digitalWrite(redPin, LOW);
    }
  }
}
```

program.py :

```
import serial
import time
import pymysql

device = '/dev/ttyS0'
arduino = serial.Serial(device, 9600)

dbConn = pymysql.connect('localhost','pi','','assignment_db') or die('could not connect t
cursor = dbConn.cursor()

sqlInsert = """INSERT INTO attempts (attempt_input, attempt_passed) VALUES (%s, %s)"""

cursor.execute("SELECT * FROM passcodes")
passcodes = cursor.fetchmany(2)
for row in passcodes:
    pass01 = row[0]
    pass02 = row[1]

while True:
    data = arduino.readline()
    print(data)
    if data.strip() == pass01:
        cursor.execute(sqlInsert,(data.strip(),1))
        dbConn.commit()
        arduino.write(b'A')
    elif data.strip() == pass02:
        cursor.execute(sqlInsert,(data.strip(),1))
        dbConn.commit()
        arduino.write(b'B')
    else:
        cursor.execute(sqlInsert,(data.strip(),0))
        dbConn.commit()
        arduino.write(b'C')
```