Classified Document Room

GP 106 - COMPUTING
GROUP PROJECT

GROUP 21-A

GROUP MEMBERS

1. E/19/001 ABAYAKOON A.M.M.
2. E/19/002 ABEYNAYAKE A.G.R.T.
3. E/19/006 ABREW S.T.D.
4. E/19/028 ATUGEDARA A.N.I.

Content

1. Project Outline

The Classified Document Room (CDR) is one of the most secure sections of the pentagon. Here, we looked into the cases which the CDR requires the system to handle. The cases are as follows.

- Temperature is monitored for fire detection
- Light intensity is monitored for unusual activities
- Floor Pressure sensors are implemented for security breaches
- Secret entry sequence by 2 guards simultaneously.
- Emergency Lockdown Indicator

2. Description about Flowchart

Since there are three security clearance categories (confidential, secret, and top secret) and only two guards can enter at a time, we considered the two guards who enter for a time are in the same category. And there are six secret codes, one for each of them. First, the first guard must press the first button and enter the first code. If the first code is correct, the second guard is required to enter the second code and press the second button. The second button should be pressed within 30 seconds after pressing the first button.

If the pressure button is pressed inside the CDR when the first button is not pressed, the alarm fires and the system goes to an emergency lockdown. The pressure button is neglected only when the guards enter the CDR using the secret code correctly within 30 seconds. And, if the two codes are incorrect or the second button is not pressed after the second code is entered or the time difference between pressing the first button and the second is larger than 30 seconds, the alarm fires and the system goes to an emergency lockdown.

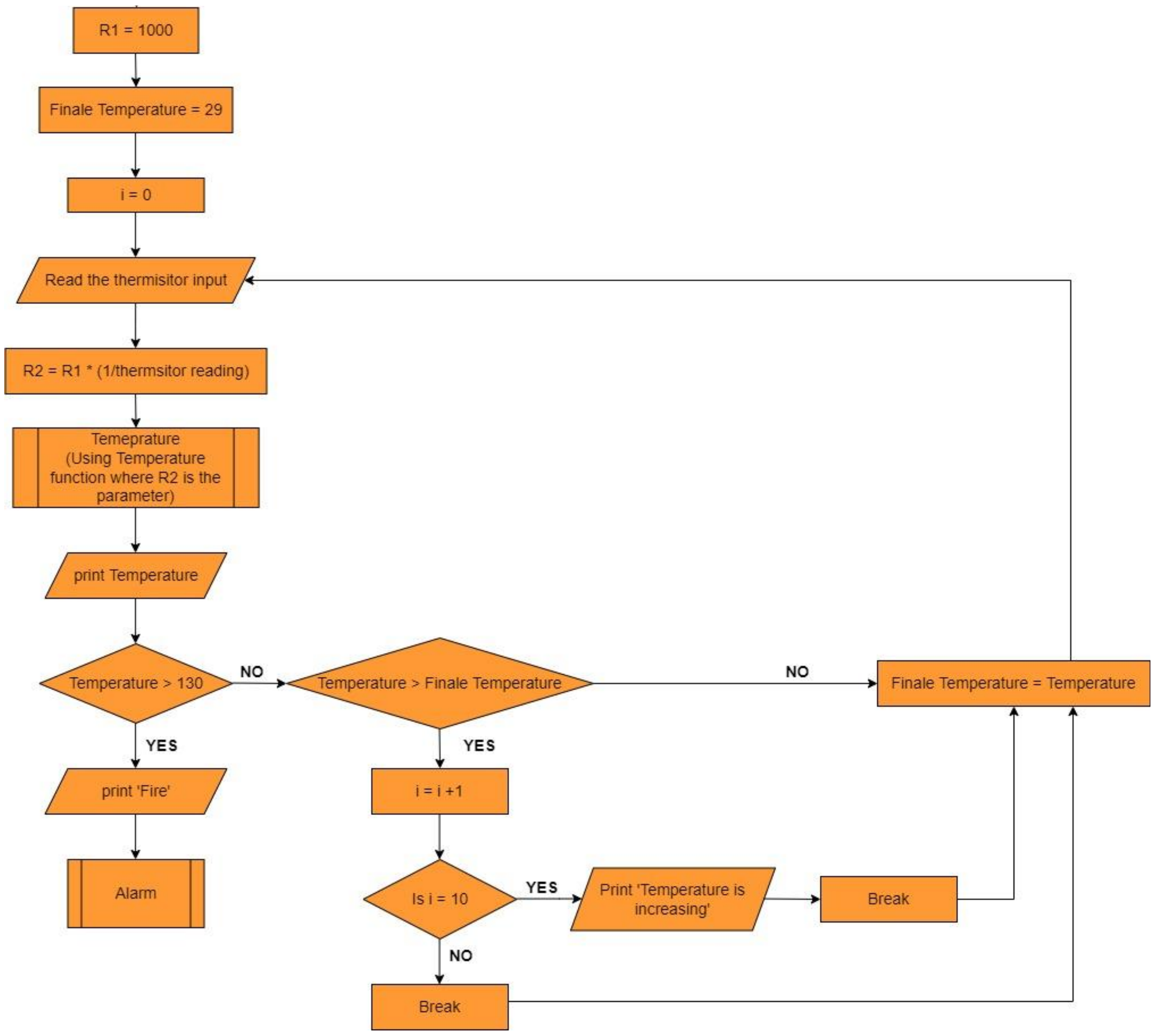
And inside the CDR, the guards can take documents two times, for 8 seconds each. Which means the light intensity sensors near the documentations can be darkened only two times for 8 seconds each. If the ldr darkens for 3 times or it is darkened for more than 8 seconds, then the alarm fires and the system goes to an emergency lockdown. And, when guards are done with their work inside CDR, they should press the off button when they leave, letting the loop begin again.

Furthermore, the temperature is measured throughout the system. If the temperature is more than 130ºC or the temperature is raised constantly, the alarm fires and the system goes to an emergency lockdown.

When the alarm is fired and the system is in emergency lockdown, the off button is pressed if the guards come to look into the problem, and then the alarm stops, and the loop begins again letting guards go inside the CDR entering secret codes.

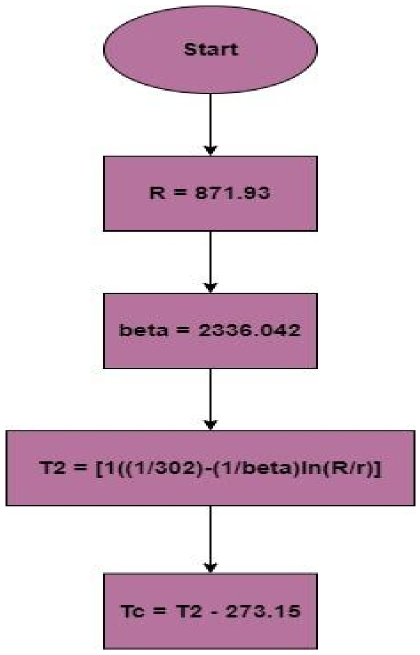
3. Flow Chart

3.1 Fire alarm function

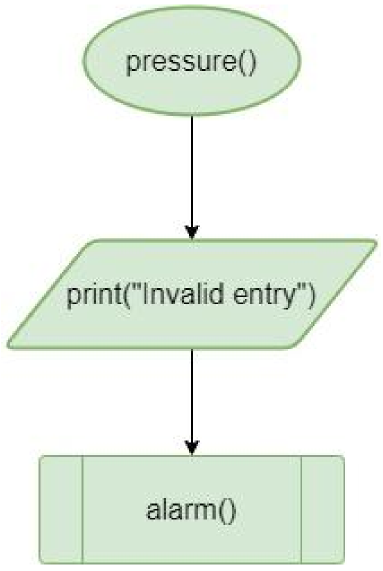


R1 = 1000
Finale Temperature = 29
i = 0
Read the thermistor input
R2 = R1 * (1/thermsitor reading)
Temeprature (Using Temperature function where R2 is the parameter)
print Temperature
Temperature > 130
NO
YES
print 'Fire'
Alarm
Temperature > Finale Temperature
NO
YES
Finale Temperature = Temperature
i = i +1
Is i = 10
YES
NO
Print 'Temperature is increasing'
Break
Break

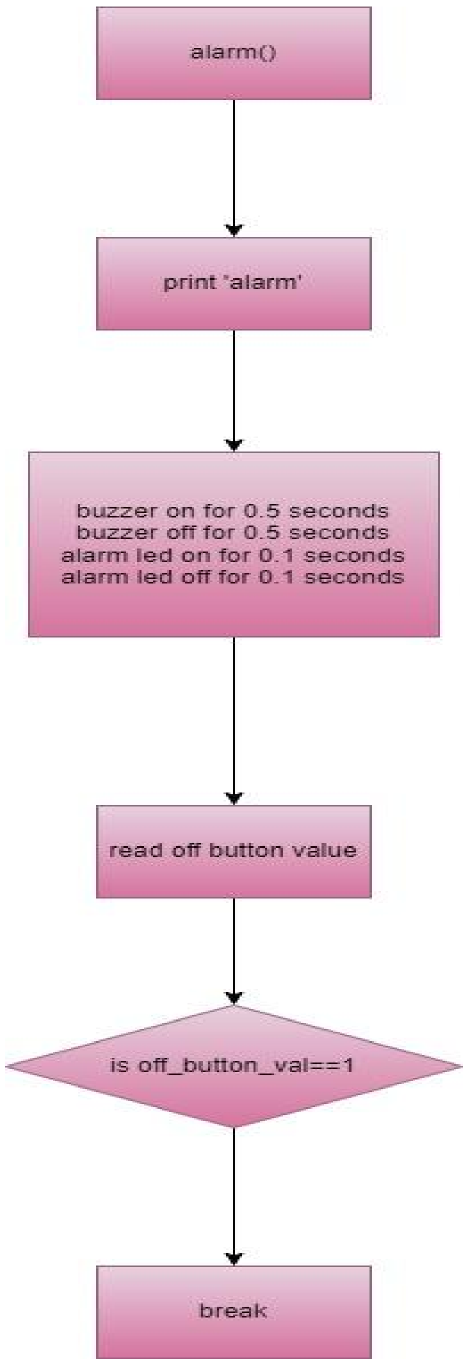
3.2 Temperature function



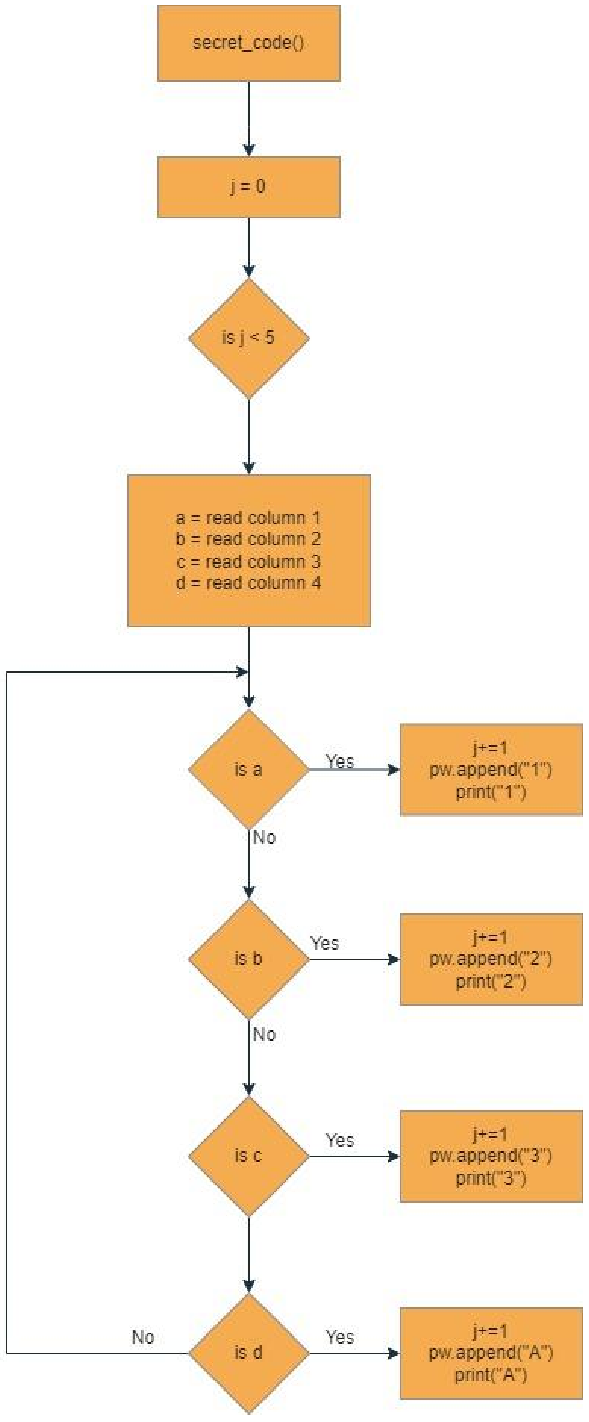
3.3 Pressure function

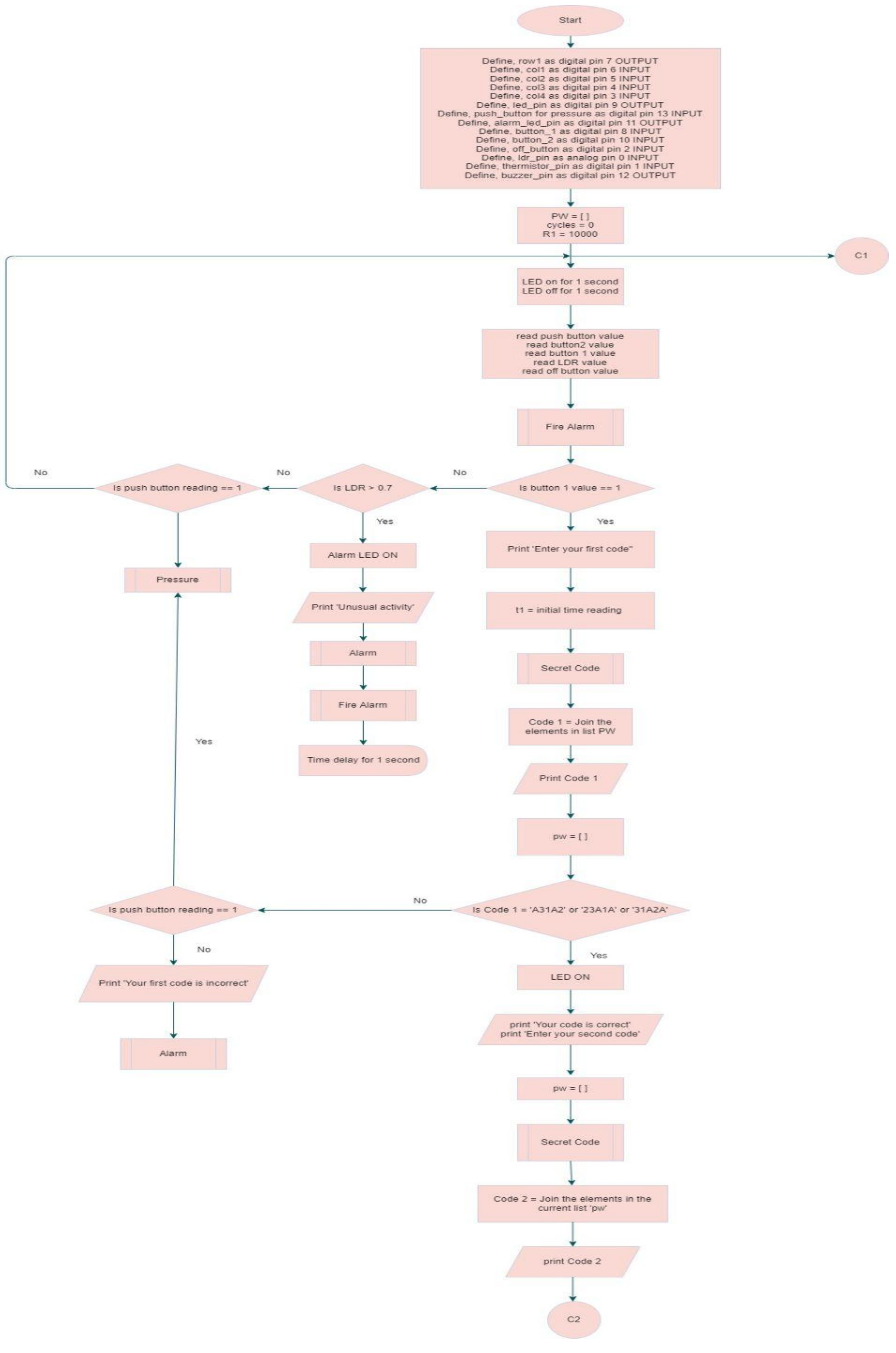


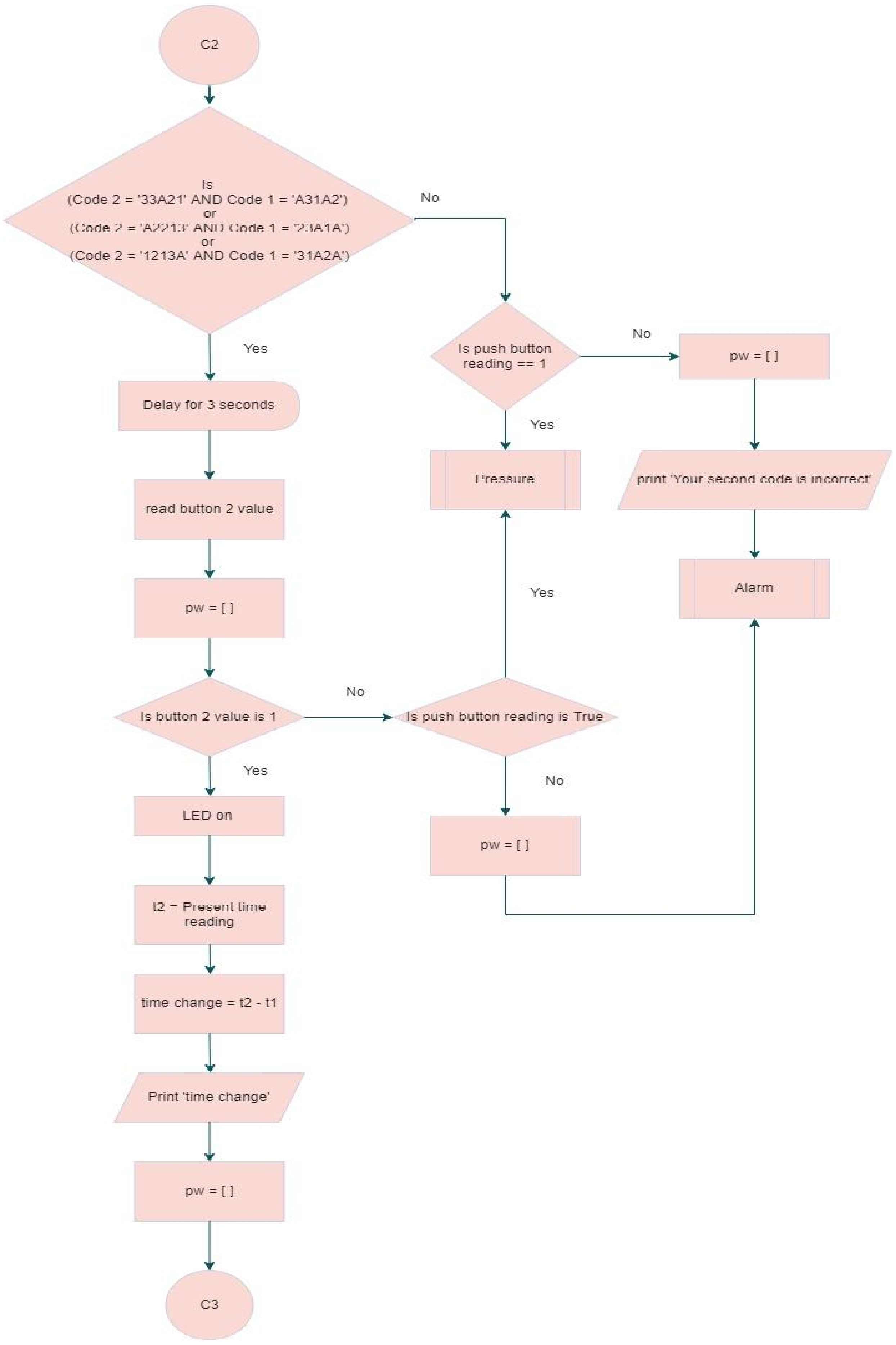
3.4 Alarm function

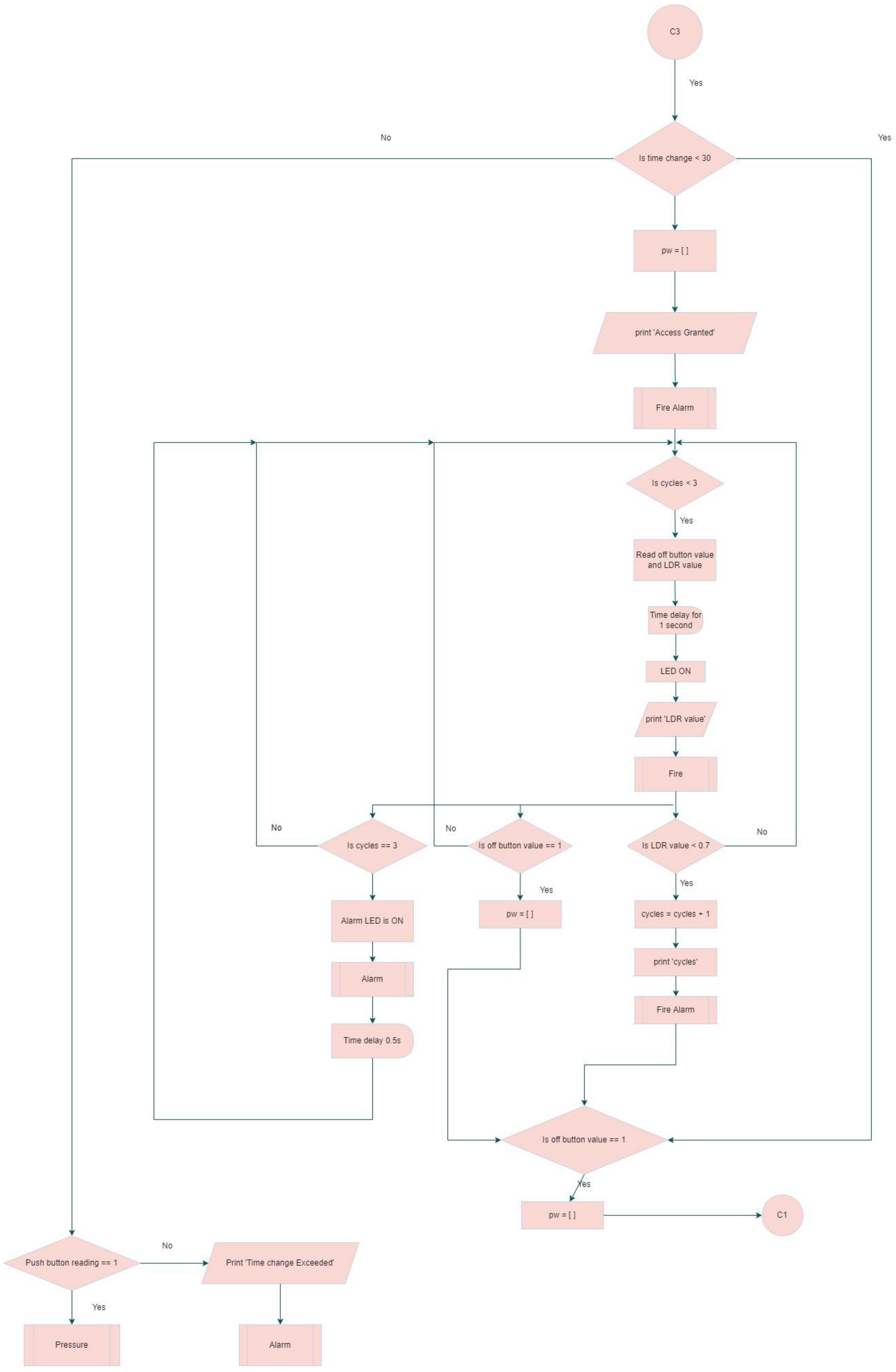

alarm()
print 'alarm'
buzzer on for 0.5 seconds
buzzer off for 0.5 seconds
alarm led on for 0.1 seconds
alarm led off for 0.1 seconds
read off button value
is off_button_val==1
break

3.5 Secret code function




Start
Define, row1 as digital pin 7 OUTPUT
Define, col1 as digital pin 6 INPUT
Define, col2 as digital pin 5 INPUT
Define, col3 as digital pin 4 INPUT
Define, col4 as digital pin 3 INPUT
Define, led_pin as digital pin 9 OUTPUT
Define, push_button for pressure as digital pin 13 INPUT
Define, alarm_led_pin as digital pin 11 OUTPUT
Define, button_1 as digital pin 8 INPUT
Define, button_2 as digital pin 10 INPUT
Define, off_button as digital pin 2 INPUT
Define, ldr_pin as analog pin 0 INPUT
Define, thermistor_pin as digital pin 1 INPUT
Define, buzzer_pin as digital pin 12 OUTPUT
PW = [ ]
cycles = 0
R1 = 10000
C1
LED on for 1 second
LED off for 1 second
read push button value
read button2 value
read button 1 value
read LDR value
read off button value
Fire Alarm
No
Is push button reading == 1
No
Is LDR > 0.7
No
Is button 1 value == 1
Yes
Yes
Print 'Enter your first code"
Alarm LED ON
Pressure
Print 'Unusual activity'
t1 = initial time reading
Alarm
Secret Code
Fire Alarm
Code 1 = Join the elements in list PW
Yes
Time delay for 1 second
Print Code 1
pw = [ ]
No
Is push button reading == 1
Is Code 1 = 'A31A2' or '23A1A' or '31A2A'
No
Yes
LED ON
Print 'Your first code is incorrect'
print 'Your code is correct'
print 'Enter your second code'
Alarm
pw = [ ]
Secret Code
Code 2 = Join the elements in the current list 'pw'
print Code 2
C2


C2
Is
(Code 2 = '33A21' AND Code 1 = 'A31A2')
or
(Code 2 = 'A2213' AND Code 1 = '23A1A')
or
(Code 2 = '1213A' AND Code 1 = '31A2A')
No
Yes
Delay for 3 seconds
read button 2 value
pw = [ ]
Is button 2 value is 1
Yes
LED on
t2 = Present time reading
time change = t2 - t1
Print 'time change'
pw = [ ]
C3
No
Is push button reading is True
Yes
No
pw = [ ]
Is push button reading == 1
Yes
Pressure
No
pw = [ ]
print 'Your second code is incorrect'
Alarm


C3
Yes
No
Yes
Is time change < 30
pw = [ ]
print 'Access Granted'
Fire Alarm
Is cycles < 3
Yes
Read off button value and LDR value
Time delay for 1 second
LED ON
print 'LDR value'
Fire
No
Is cycles == 3
No
Is off button value == 1
No
Is LDR value < 0.7
Yes
Yes
Alarm LED is ON
pw = [ ]
cycles = cycles + 1
print 'cycles'
Alarm
Fire Alarm
Time delay 0.5s
Is off button value == 1
Yes
pw = [ ]
C1
No
Push button reading == 1
Print 'Time change Exceeded'
Yes
Pressure
Alarm

4. code:

```
#Group 21-A
#02/03/2022

#Import time and math modules
import time
import math
from pyfirmata import Arduino, util

board = Arduino('COM3')

##defining pins
row1=board.get_pin('d:7:o') #define first row in keypad as Digital output pin 7
col1=board.get_pin('d:6:i') #define columns in keypad as Digital input pins 6,5,4 and 3
col2=board.get_pin('d:5:i')
col3=board.get_pin('d:4:i')
col4=board.get_pin('d:3:i')
button1=board.get_pin('d:8:i') #define first push button as Digital input pin 8
button2 = board.get_pin('d:10:i') #define second push button as Digital input pin 10
off_button=board.get_pin('d:2:i') #define alarm_off push button as Digital input pin 2
p_button = board.get_pin('d:13:i') #define push button for pressure as Digital input pin 13
led_pin = board.get_pin('d:9:o') #define led as Digital output pin 9
alarm_led_pin = board.get_pin('d:11:o') #define alarm led as Digital output pin 11
buzzer_pin = board.get_pin('d:12:o') #define buzzer as Digital output pin 12
ldr_pin = board.get_pin('a:0:i') #define ldr as Analog input pin 0
Thermister_Pin = board.get_pin('a:1:i') #define thermistor as Analog input pin 1

iterator = util.Iterator(board)
iterator.start()

R1 = 10000 #value of the resistor which is connected to the thermistor
```

```
cycles = 0 #getting number of times when ldr darkens
pw=[] #getting a list to collect code

##defining the code function to enter
def secret_code():
    j=0
    while j<5: #taking the code of five digits
        row1.write(1)
        time.sleep(0.1)
        a=col1.read()
        b=col2.read()
        c=col3.read()
        d=col4.read()

        if a==True: #printing '1' if col1 is pressed in row1
            j+=1
            pw.append("1") #appending '1' to the list 'pw'
            print("1")
        elif b==True: #printing '2' if col2 is pressed in row1
            j+=1
            pw.append("2") #appending '2' to the list 'pw'
            print("2")
        elif c==True: #printing '3' if col3 is pressed in row1
            j+=1
            pw.append("3") #appending '3' to the list 'pw'
            print("3")
        elif d==True: #printing 'A' if col4 is pressed in row1
            j+=1
            pw.append("A") #appending 'A' to the list 'pw'
            print("A")

##defining the alarm and the alarm led function
```

```
def alarm():
    print('alarm')
    #execute the loop for buzzer
    while True:
        buzzer_pin.write(1) #turn buzzer on
        time.sleep(0.5)
        buzzer_pin.write(0) #turn buzzer off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn alarm led on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn alarm led off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading off button value
        if (off_button_val): #if off_button is pressed(indicating guards came and looked into the
alarm), break the buzzer loop
            break

##defining pressure alarm function
def pressure():
    print("Invalid entry")
    alarm()

##defining temperature function to convert the resistor value of the thermistor to temperature
def temperature(r):
    R = 871.93
    beta = 2336.042
    T2 = 1/((1/302)-(1/beta)*math.log(R/r))
    Tc = T2 - 273.15
    return Tc

##defining fire alarm and temperature increasing alert
def fire_alarm():
    i = 0 #defining number of times the temperature is increasing
    T_finale  = 28 #defining the variable T_finale as 28 celsius(room temperature)
    thermistor_read = Thermister_Pin.read() #reading thermistor value
    R2 = R1 *(1 / float(thermistor_read) - 1.0) #calculating the resistance of the thermistor
    T_val = temperature(R2) #calling the temperature function to calculate temperature
    print('Temperature value is ',T_val)
    print()

    if T_val > 130: #detecting a fire using the temperature reading when temperature reading is
more than 130
        print('Fire')
        alarm() #calling alarm function

    elif T_val > T_finale: #detecting fire increment for 10 temperature readings
        i += 1
        while True:
            if i == 10: #giving temperature increasing alert
                print('Temperature is Increasing')
                T_finale = T_val
                break
            else:
                break

#executing the while True loop
while True:
    led_pin.write(1) #turn led on
    time.sleep(0.1)
    led_pin.write(0) #turn led off
    time.sleep(0.1)

    #reading push button value, button2 value, button1 value, ldr value and off button value
    sw = p_button.read()
```

```
  button2_val = button2.read()
  button1_val = button1.read()
  ldr_val = ldr_pin.read()
  off_button_val=off_button.read()

  fire_alarm() #calling fire_alarm

  if button1_val ==1:
      print("Enter your first code : ") #asking for the first code
      start=time.monotonic()
      secret_code() #calling secret_code
      code_1="".join(pw) #joining the elements in the list
      print(code_1)
      pw=[] #emptying the list 'pw'

      #Confidential category secret sequence  = A31A2
      #Secret category secret sequence = 23A1A
      #Top Secret sequence = 31A2A

      ##checking whether the first code is correct according to 3 security clearance categories:
confidential, secret, and top secret
      if code_1=='A31A2' or code_1=='23A1A' or code_1 == '31A2A':
          led_pin.write(1) #turning led on
          print("Your first code is correct.")
          print("Enter your second code : ") #asking for the first code
          pw = [] #emptying the list 'pw'
          secret_code() #calling secret_code
          code_2="".join(pw)
          print(code_2)

          #Confidential category secret sequence  = 33A21
          #Secret category secret sequence = A2213
```

```
    #Top Secret sequence = 1213A

    ##checking whether both codes are correct according to 3 security clearance categories
    if (code_2=='33A21' and code_1=='A31A2') or (code_2=='A2213' and code_1=='23A1A') or
(code_2== '1213A' and code_1 == '31A2A') :
        time.sleep(3)
        button2_val = button2.read() #reading button2 value
        pw=[] #emptying the list 'pw'
        if button2_val==True: #checking whether button2 is pressed
            end=time.monotonic()
            led_pin.write(1)
            time_change=end-start
            print(time_change)
            pw=[]
            if time_change<30: #granting access if time difference between pressing button1
and button2 is smaller than 30 seconds
                pw=[]
                print("Access granted")
                fire_alarm()

                while cycles < 3: #getting ldr values until ldr darkens for two times
                    off_button_val=off_button.read()
                    ldr_val = ldr_pin.read()
                    time.sleep(1)
                    led_pin.write(1)
                    print("LDR Val %s" % ldr_val)
                    fire_alarm()

                    if ldr_val > 0.7: #counting cycles when ldr darkens
                        cycles+=1
                        print(cycles)
                        fire_alarm()
                        led_pin.write(0)
                        time.sleep(8) #letting ldr to be dark for 8 seconds

                    if cycles==3: #firing alarm if ldr darkens for three times
                        alarm_led_pin.write(1)
                        alarm()
                        time.sleep(0.5)
                        cycles=0 #taking number of cycles as zero for the next loop
                        print('alarmed')
                        fire_alarm()
                        break

                    if off_button_val==1: #checking whether the off_button is pressed to check
whether the guards have left
                        pw=[] #emptying the list 'pw'
                       cycles=0 #taking number of cycles as zero for the next loop
                        break #break the loop, if off_button is pressed

                    else:
                        continue

                if off_button_val==1: #continue the loop from the beginning, if off_button is
pressed when guards are leaving
                    pw=[]
                    continue

            elif sw == 1: #calling pressure function if pressure button is pressed when the time
difference between pressing button1 and button2 is larger than 30 seconds
                pressure()
            else: #calling alarm function if pressure button is pressed when the time difference
between pressing button1 and button2 is larger than 30 seconds
                print("Time change exceeded.")
```

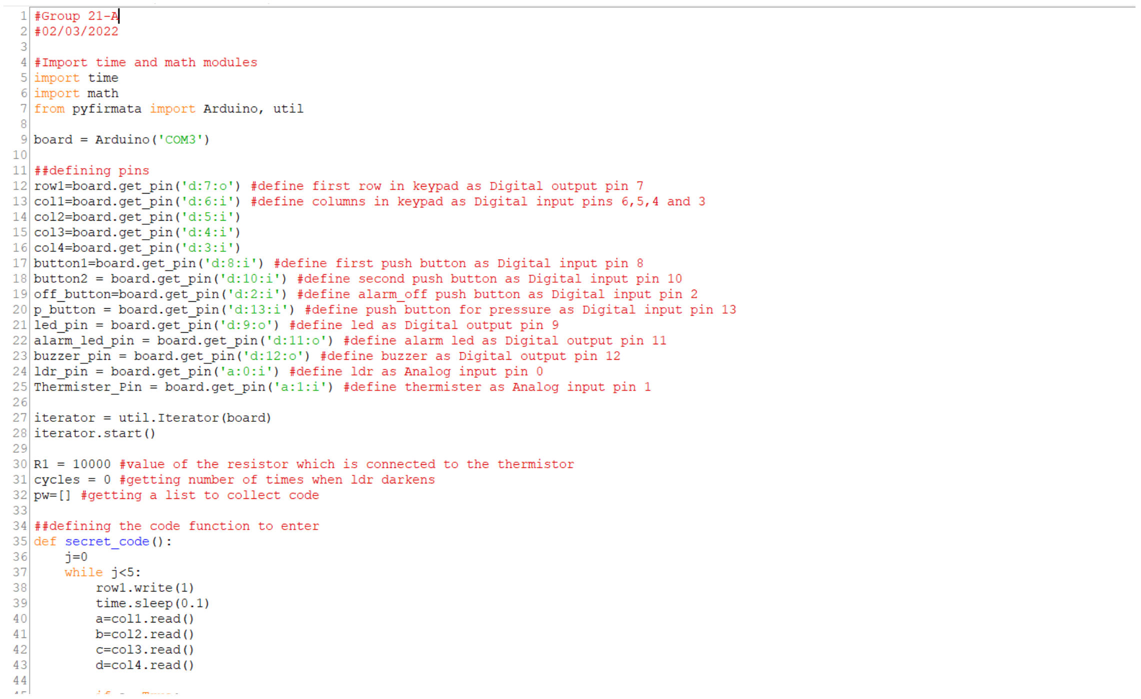
```
                        alarm()

                elif sw == 1: #calling pressure function if pressure button is pressed when button2 is
not pressed
                    pressure()
                else: #calling alarm function if pressure button is pressed when button2 is not
pressed
                    end=time.monotonic()
                    print(end-start)
                    pw=[]
                    alarm()

            elif sw == 1: #calling pressure function if pressure button is pressed when the second
code is incorrect
                pressure()
            else: #calling alarm function if pressure button is pressed when the second code is
incorrect
                pw=[]
                print("Your second code is incorrect.")
                alarm()

        elif sw == 1: #calling pressure function if pressure button is pressed when the first code is
incorrect
            pressure()
        else: #calling alarm function if pressure button is pressed when the first code is incorrect
            print("Your first code is incorrect.")
            alarm()

    elif ldr_val > 0.7: #firing alarm if ldr darkens when button1 is not pressed
        alarm_led_pin.write(1)
        fire_alarm()
        print('Unusual activity')
        alarm()
        time.sleep(1)

    elif sw == 1: #calling pressure function if pressure button is pressed when button1 is not
pressed
        pressure()

    else: #continue the loop
        continue

    time.sleep(0.1)
```

5. Observing functionality of the code

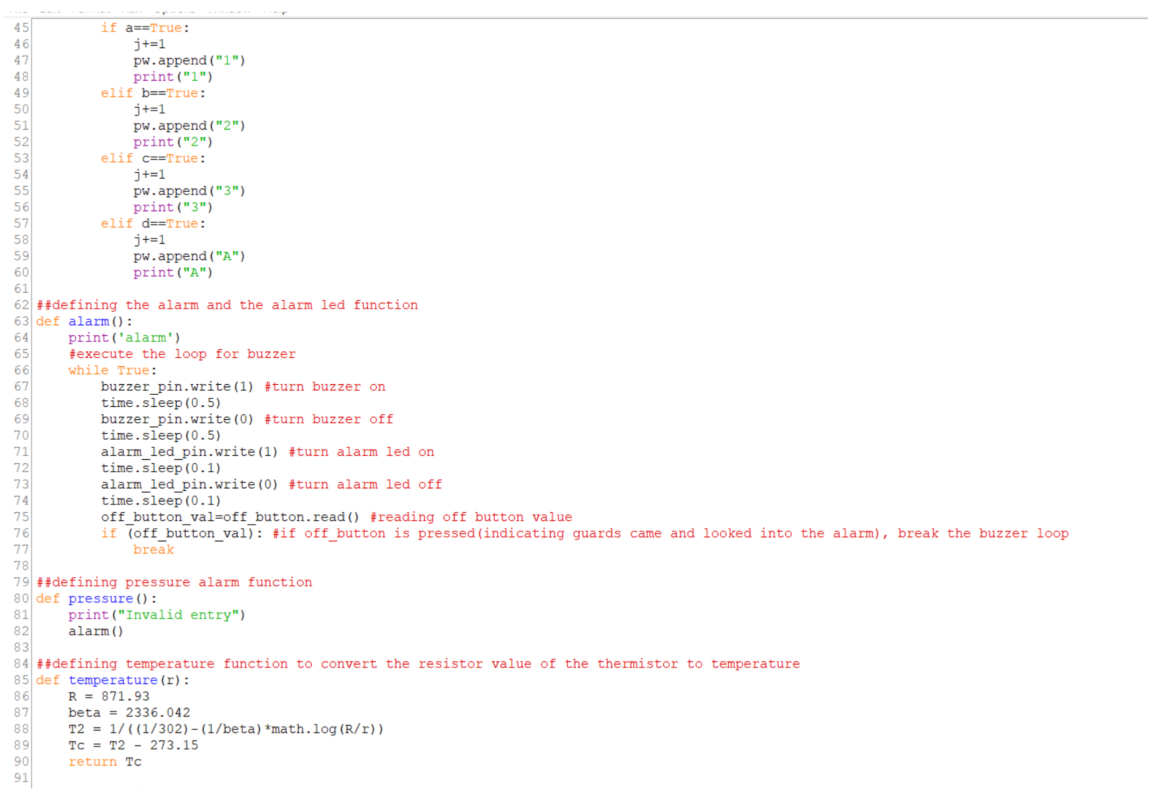
Drive link :
https://drive.google.com/drive/folders/1UGxn4z4Fclc13r9Dta4Q9CXvSOqHH55q?usp=sharing

Through these videos, the functionality of the code is shown in various situations.
In the first video,at the beginning, the functionality of pressure and light intensity when the secret code is not entered was observed respectively (alarm fires). After that, the wrong first code and wrong second code was entered and the alarm was fired. Then, the codes were entered correctly and pressure,temperature and light intensity for two cycles functions were observed and went out of the code pressing the off button.
In the second video, codes were entered correctly and the ldr was darkened for three times and the alarm was fired.
In the third video, the fire alarm was observed when the code was not entered and when the codes were entered.
In the fourth video, the functionality of code when the time difference between the first and second code is more than 30 seconds was observed and the alarm was fired.

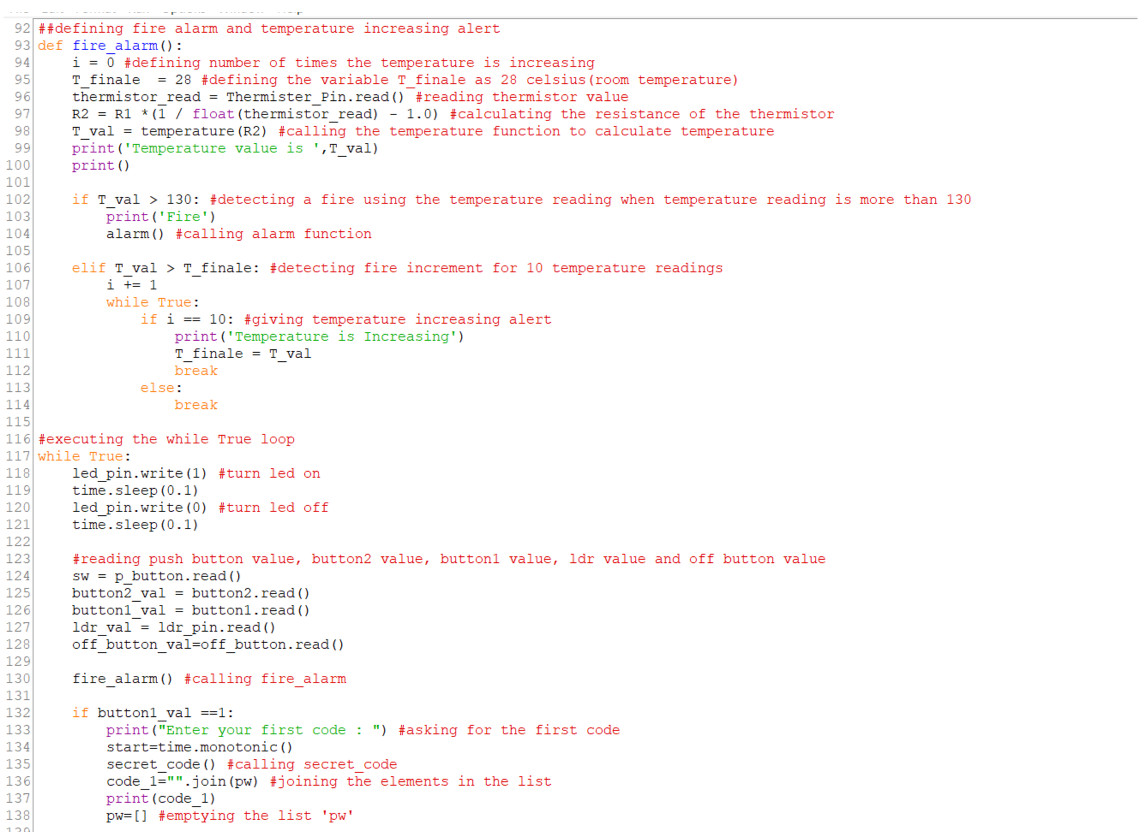
6. Code snippets:



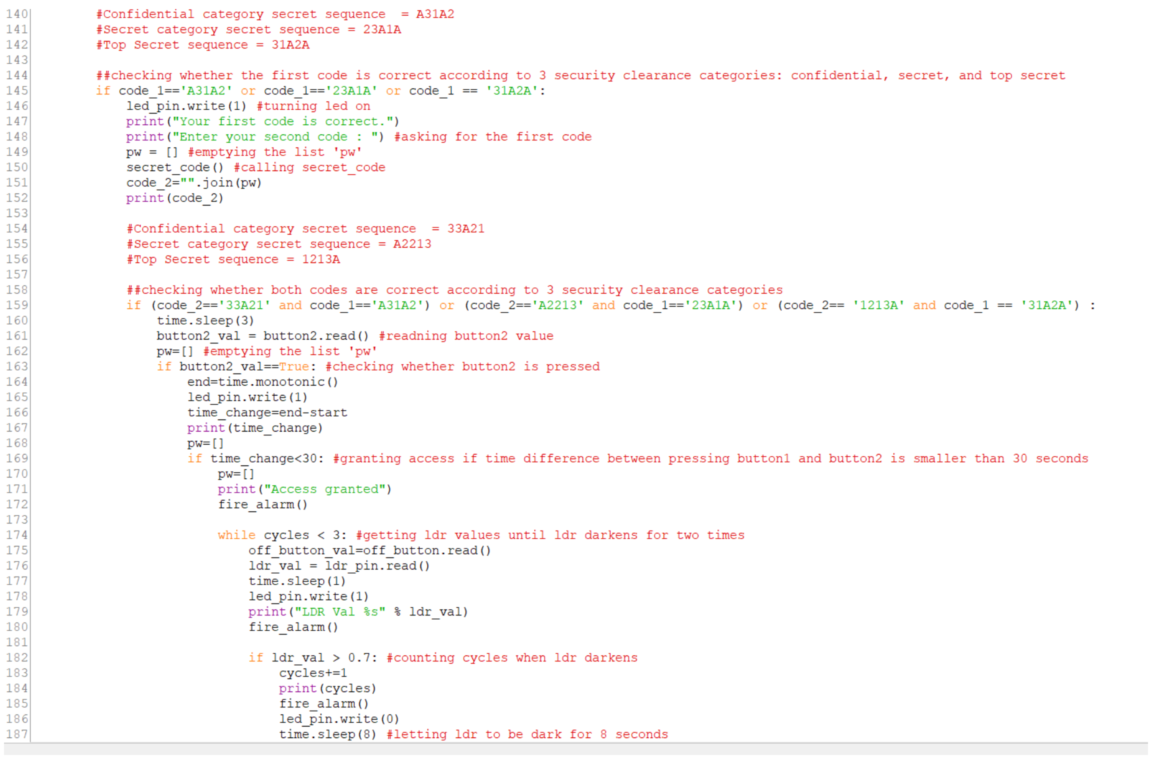
```python
#Group 21-A
#02/03/2022

#Import time and math modules
import time
import math
from pyfirmata import Arduino, util

board = Arduino('COM3')

##defining pins
row1=board.get_pin('d:7:o') #define first row in keypad as Digital output pin 7
col1=board.get_pin('d:6:i') #define columns in keypad as Digital input pins 6,5,4 and 3
col2=board.get_pin('d:5:i')
col3=board.get_pin('d:4:i')
col4=board.get_pin('d:3:i')
button1=board.get_pin('d:8:i') #define first push button as Digital input pin 8
button2 = board.get_pin('d:10:i') #define second push button as Digital input pin 10
off_button=board.get_pin('d:2:i') #define alarm_off push button as Digital input pin 2
p_button = board.get_pin('d:13:i') #define push button for pressure as Digital input pin 13
led_pin = board.get_pin('d:9:o') #define led as Digital output pin 9
alarm_led_pin = board.get_pin('d:11:o') #define alarm led as Digital output pin 11
buzzer_pin = board.get_pin('d:12:o') #define buzzer as Digital output pin 12
ldr_pin = board.get_pin('a:0:i') #define ldr as Analog input pin 0
Thermister_Pin = board.get_pin('a:1:i') #define thermister as Analog input pin 1

iterator = util.Iterator(board)
iterator.start()

R1 = 10000 #value of the resistor which is connected to the thermistor
cycles = 0 #getting number of times when ldr darkens
pw=[] #getting a list to collect code

##defining the code function to enter
def secret_code():
    j=0
    while j<5:
        row1.write(1)
        time.sleep(0.1)
        a=col1.read()
        b=col2.read()
        c=col3.read()
        d=col4.read()

```



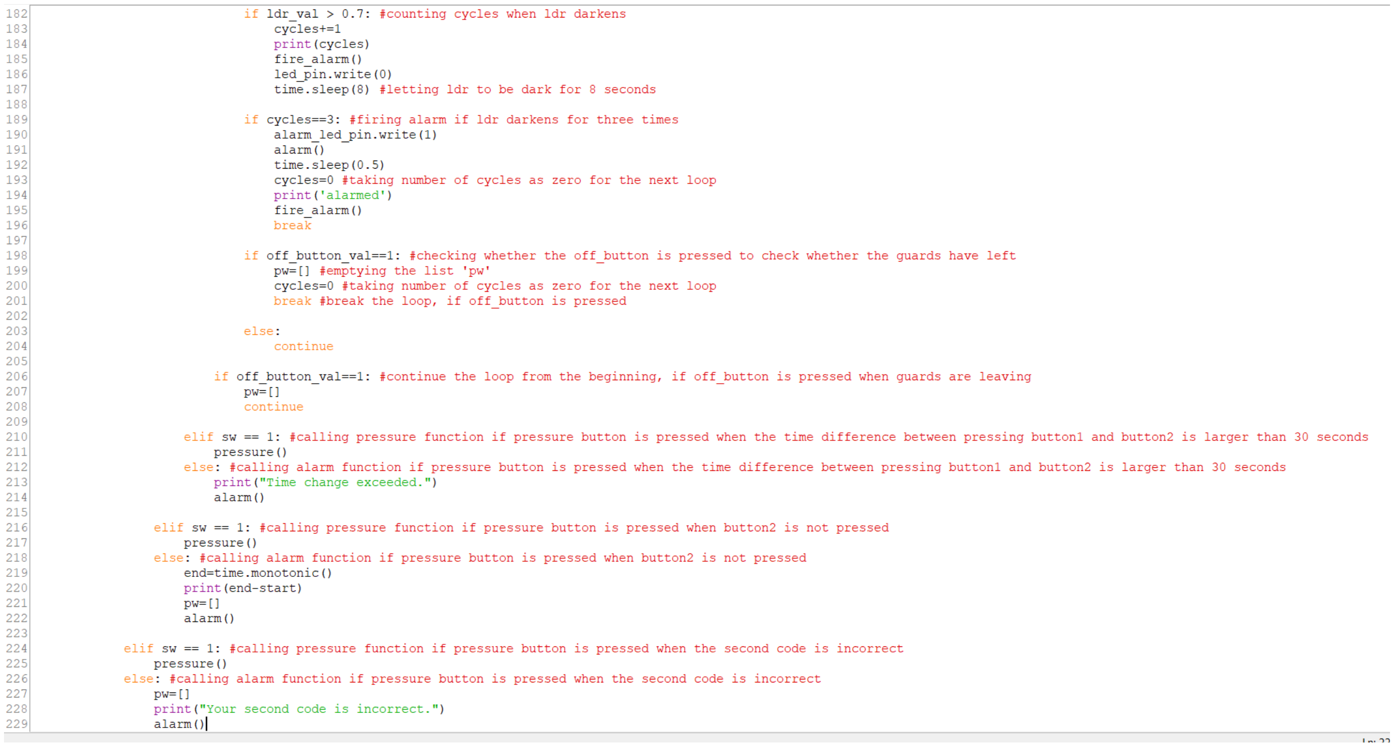
```python
        if a==True:
            j+=1
            pw.append("1")
            print("1")
        elif b==True:
            j+=1
            pw.append("2")
            print("2")
        elif c==True:
            j+=1
            pw.append("3")
            print("3")
        elif d==True:
            j+=1
            pw.append("A")
            print("A")

##defining the alarm and the alarm led function
def alarm():
    print('alarm')
    #execute the loop for buzzer
    while True:
        buzzer_pin.write(1) #turn buzzer on
        time.sleep(0.5)
        buzzer_pin.write(0) #turn buzzer off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn alarm led on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn alarm led off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading off button value
        if (off_button_val): #if off_button is pressed(indicating guards came and looked into the alarm), break the buzzer loop
            break

##defining pressure alarm function
def pressure():
    print("Invalid entry")
    alarm()

##defining temperature function to convert the resistor value of the thermistor to temperature
def temperature(r):
    R = 871.93
    beta = 2336.042
    T2 = 1/((1/302)-(1/beta)*math.log(R/r))
    Tc = T2 - 273.15
    return Tc

```



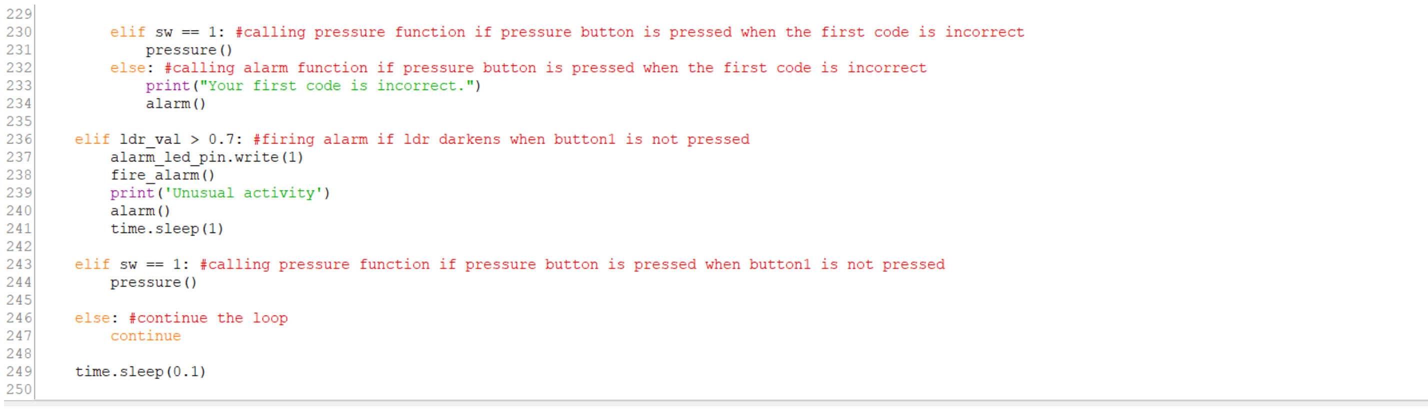
```python
##defining fire alarm and temperature increasing alert
def fire_alarm():
    i = 0 #defining number of times the temperature is increasing
    T_finale  = 28 #defining the variable T_finale as 28 celsius(room temperature)
    thermistor_read = Thermister_Pin.read() #reading thermistor value
    R2 = R1 *(1 / float(thermistor_read) - 1.0) #calculating the resistance of the thermistor
    T_val = temperature(R2) #calling the temperature function to calculate temperature
    print('Temperature value is ',T_val)
    print()

    if T_val > 130: #detecting a fire using the temperature reading when temperature reading is more than 130
        print('Fire')
        alarm() #calling alarm function

    elif T_val > T_finale: #detecting fire increment for 10 temperature readings
        i += 1
        while True:
            if i == 10: #giving temperature increasing alert
                print('Temperature is Increasing')
                T_finale = T_val
                break
            else:
                break

#executing the while True loop
while True:
    led_pin.write(1) #turn led on
    time.sleep(0.1)
    led_pin.write(0) #turn led off
    time.sleep(0.1)

    #reading push button value, button2 value, button1 value, ldr value and off button value
    sw = p_button.read()
    button2_val = button2.read()
    button1_val = button1.read()
    ldr_val = ldr_pin.read()
    off_button_val=off_button.read()

    fire_alarm() #calling fire_alarm

    if button1_val ==1:
        print("Enter your first code : ") #asking for the first code
        start=time.monotonic()
        secret_code() #calling secret_code
        code_1="".join(pw) #joining the elements in the list
        print(code_1)
        pw=[] #emptying the list 'pw'
```



```python
        #Confidential category secret sequence  = A31A2
        #Secret category secret sequence = 23A1A
        #Top Secret sequence = 31A2A

        ##checking whether the first code is correct according to 3 security clearance categories: confidential, secret, and top secret
        if code_1=='A31A2' or code_1=='23A1A' or code_1 == '31A2A':
            led_pin.write(1) #turning led on
            print("Your first code is correct.")
            print("Enter your second code : ") #asking for the first code
            pw = [] #emptying the list 'pw'
            secret_code() #calling secret_code
            code_2="".join(pw)
            print(code_2)

            #Confidential category secret sequence  = 33A21
            #Secret category secret sequence = A2213
            #Top Secret sequence = 1213A

            ##checking whether both codes are correct according to 3 security clearance categories
            if (code_2=='33A21' and code_1=='A31A2') or (code_2=='A2213' and code_1=='23A1A') or (code_2== '1213A' and code_1 == '31A2A') :
                time.sleep(3)
                button2_val = button2.read() #readning button2 value
                pw=[] #emptying the list 'pw'
                if button2_val==True: #checking whether button2 is pressed
                    end=time.monotonic()
                    led_pin.write(1)
                    time_change=end-start
                    print(time_change)
                    pw=[]
                    if time_change<30: #granting access if time difference between pressing button1 and button2 is smaller than 30 seconds
                        pw=[]
                        print("Access granted")
                        fire_alarm()

                        while cycles < 3: #getting ldr values until ldr darkens for two times
                            off_button_val=off_button.read()
                            ldr_val = ldr_pin.read()
                            time.sleep(1)
                            led_pin.write(1)
                            print("LDR Val %s" % ldr_val)
                            fire_alarm()

                            if ldr_val > 0.7: #counting cycles when ldr darkens
                                cycles+=1
                                print(cycles)
                                fire_alarm()
                                led_pin.write(0)
                                time.sleep(8) #letting ldr to be dark for 8 seconds
```



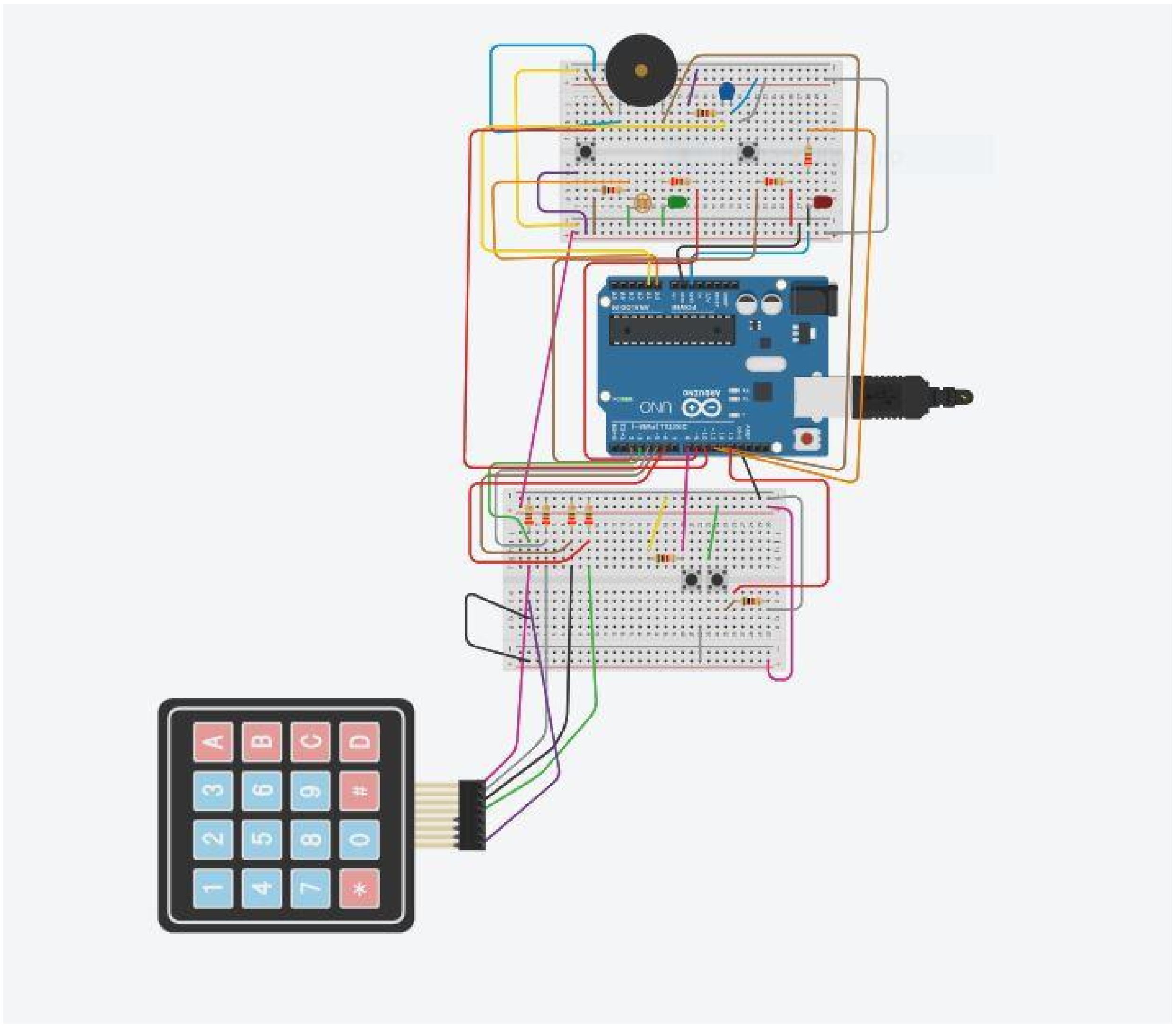
```
                        if ldr_val > 0.7: #counting cycles when ldr darkens
                            cycles+=1
                            print(cycles)
                            fire_alarm()
                            led_pin.write(0)
                            time.sleep(8) #letting ldr to be dark for 8 seconds

                        if cycles==3: #firing alarm if ldr darkens for three times
                            alarm_led_pin.write(1)
                            alarm()
                            time.sleep(0.5)
                            cycles=0 #taking number of cycles as zero for the next loop
                            print('alarmed')
                            fire_alarm()
                            break

                        if off_button_val==1: #checking whether the off_button is pressed to check whether the guards have left
                            pw=[] #emptying the list 'pw'
                            cycles=0 #taking number of cycles as zero for the next loop
                            break #break the loop, if off_button is pressed

                        else:
                            continue

                    if off_button_val==1: #continue the loop from the beginning, if off_button is pressed when guards are leaving
                        pw=[]
                        continue

                elif sw == 1: #calling pressure function if pressure button is pressed when the time difference between pressing button1 and button2 is larger than 30 seconds
                    pressure()
                else: #calling alarm function if pressure button is pressed when the time difference between pressing button1 and button2 is larger than 30 seconds
                    print("Time change exceeded.")
                    alarm()

            elif sw == 1: #calling pressure function if pressure button is pressed when button2 is not pressed
                pressure()
            else: #calling alarm function if pressure button is pressed when button2 is not pressed
                end=time.monotonic()
                print(end-start)
                pw=[]
                alarm()

        elif sw == 1: #calling pressure function if pressure button is pressed when the second code is incorrect
            pressure()
        else: #calling alarm function if pressure button is pressed when the second code is incorrect
            pw=[]
            print("Your second code is incorrect.")
            alarm()
```



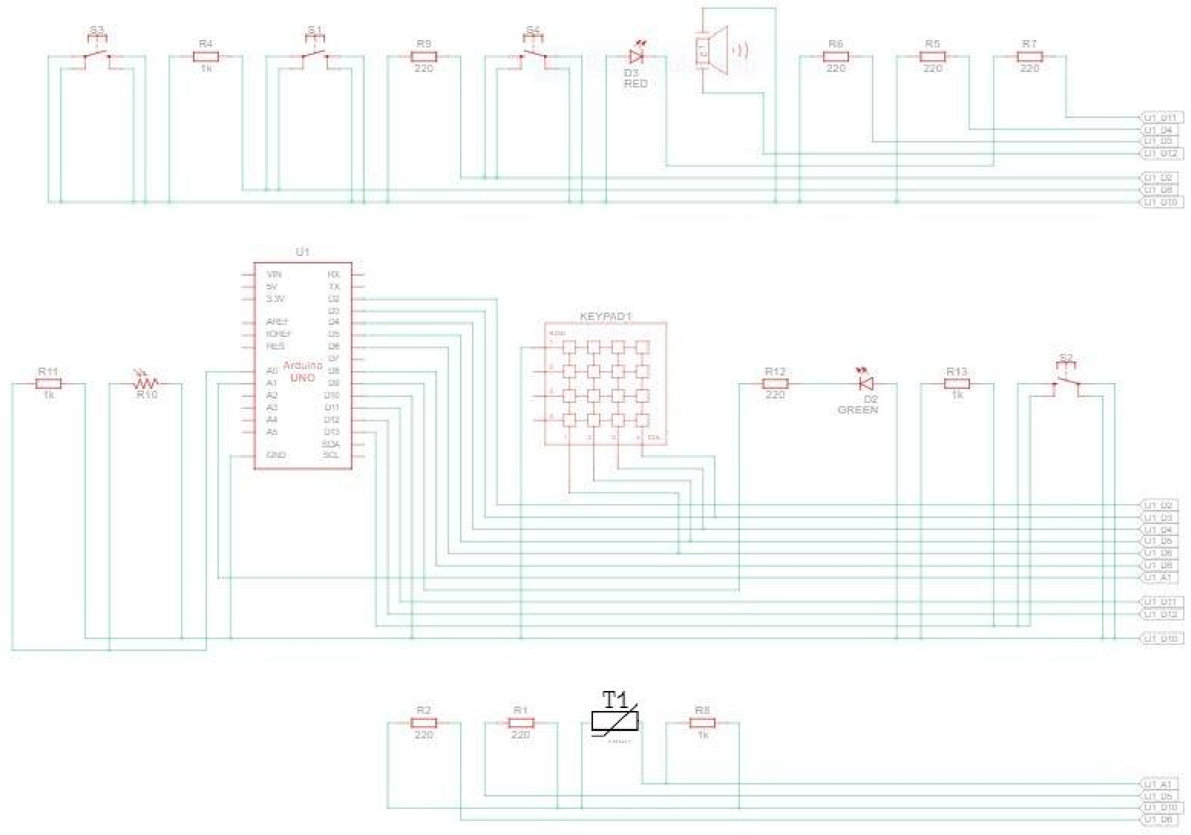
```

        elif sw == 1: #calling pressure function if pressure button is pressed when the first code is incorrect
            pressure()
        else: #calling alarm function if pressure button is pressed when the first code is incorrect
            print("Your first code is incorrect.")
            alarm()

    elif ldr_val > 0.7: #firing alarm if ldr darkens when button1 is not pressed
        alarm_led_pin.write(1)
        fire_alarm()
        print('Unusual activity')
        alarm()
        time.sleep(1)

    elif sw == 1: #calling pressure function if pressure button is pressed when button1 is not pressed
        pressure()

    else: #continue the loop
        continue

    time.sleep(0.1)
```

7. Circuit diagram

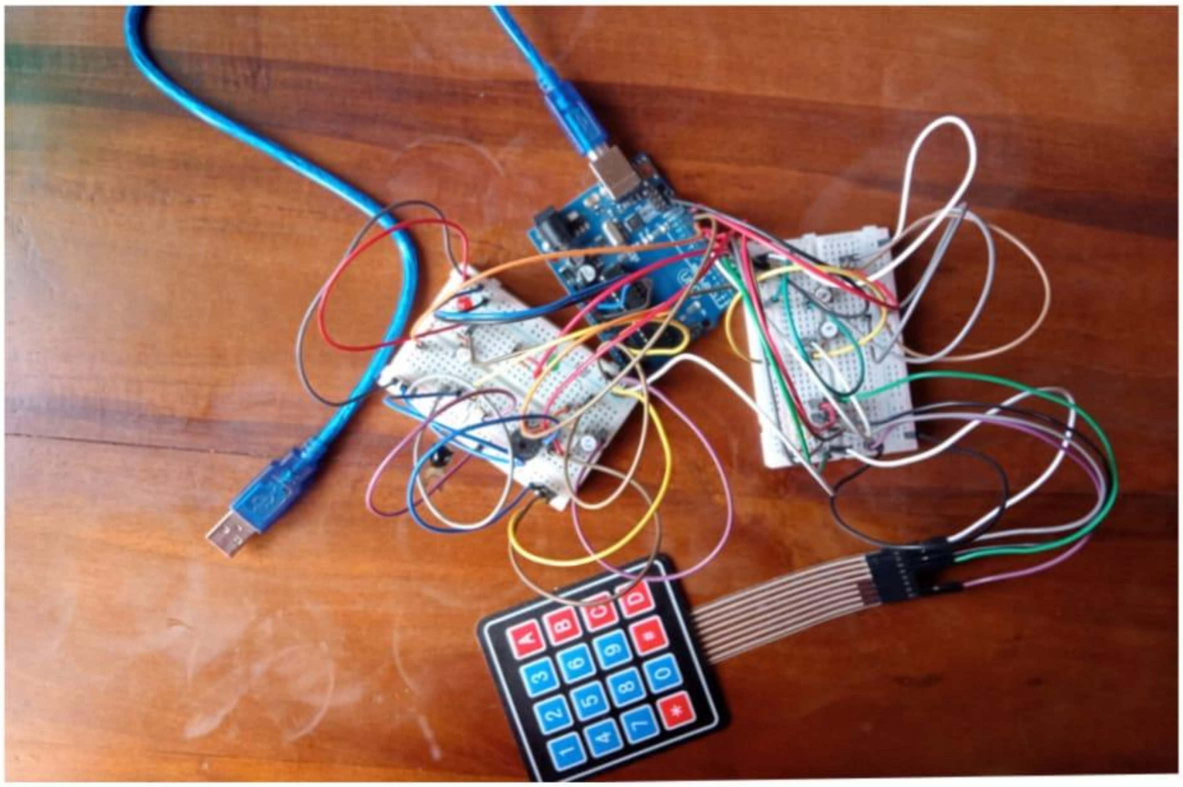
7.1 Circuit view

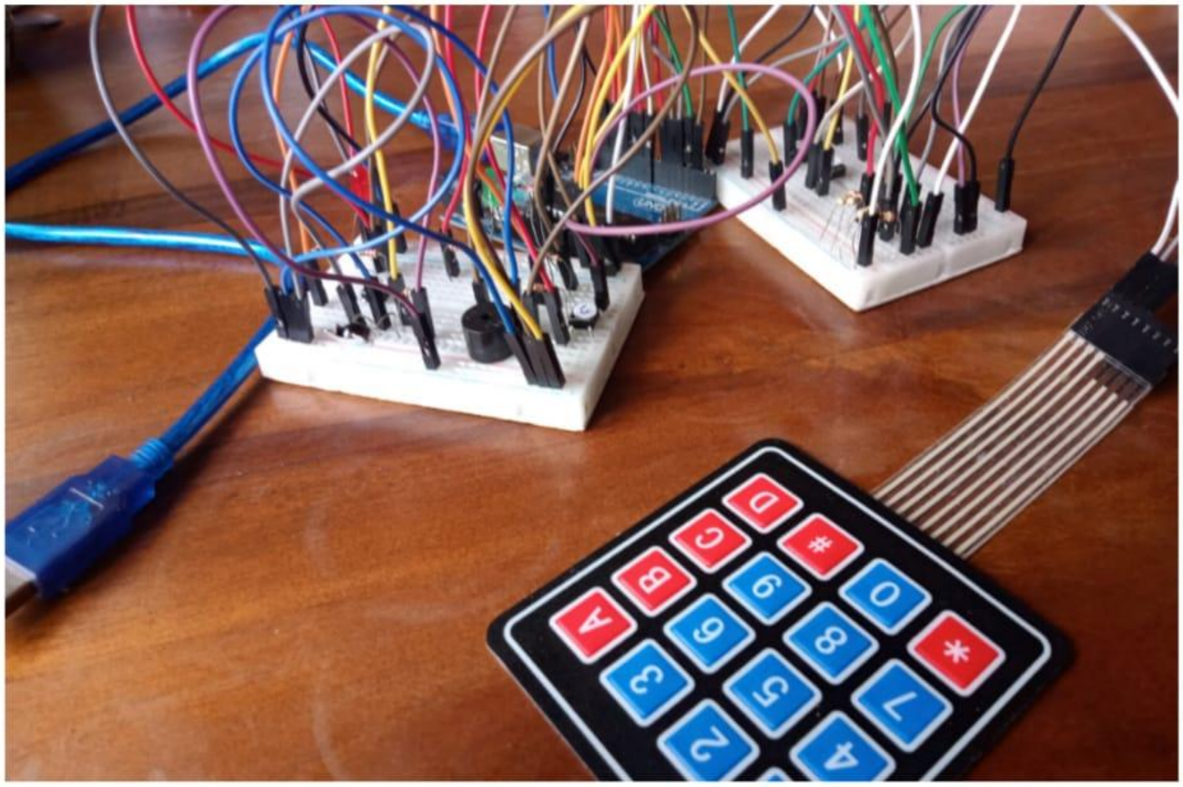


7.2 Schematic view



7.3 Actual circuit



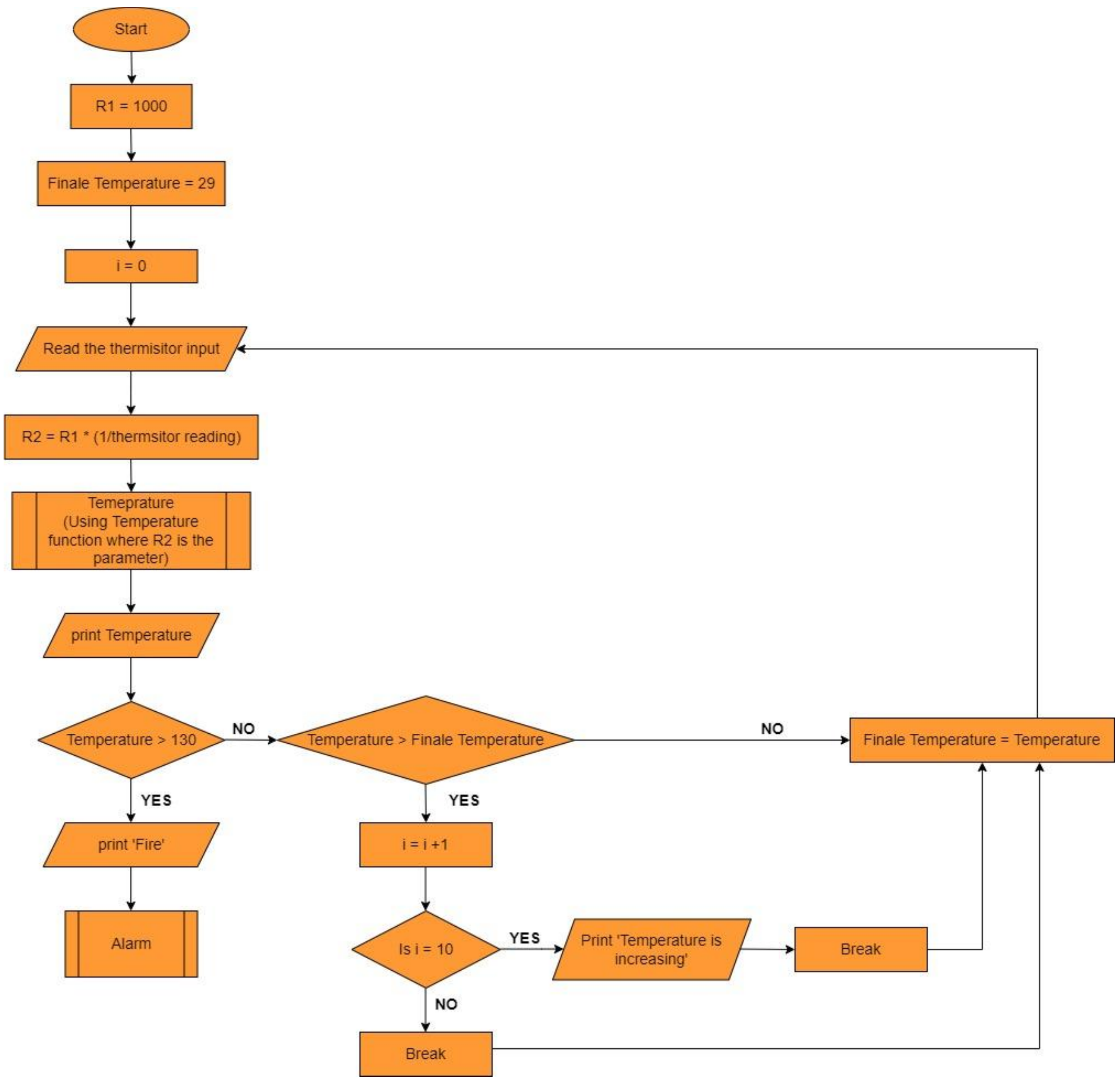


8. Temperature

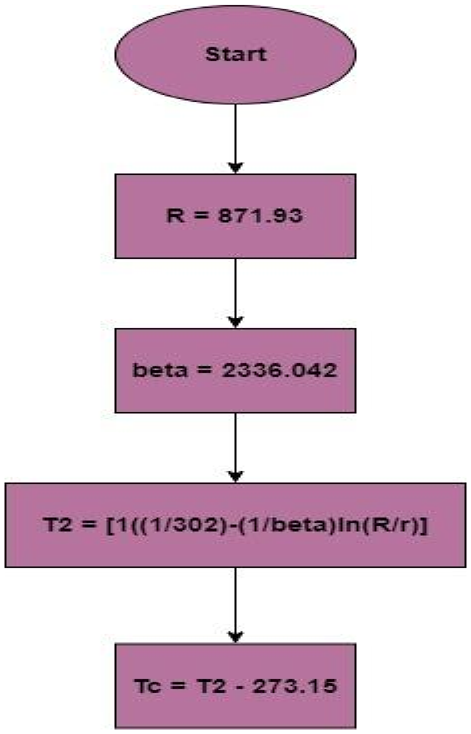
Temperature is monitored for fire detection.

8.1 Flowchart:

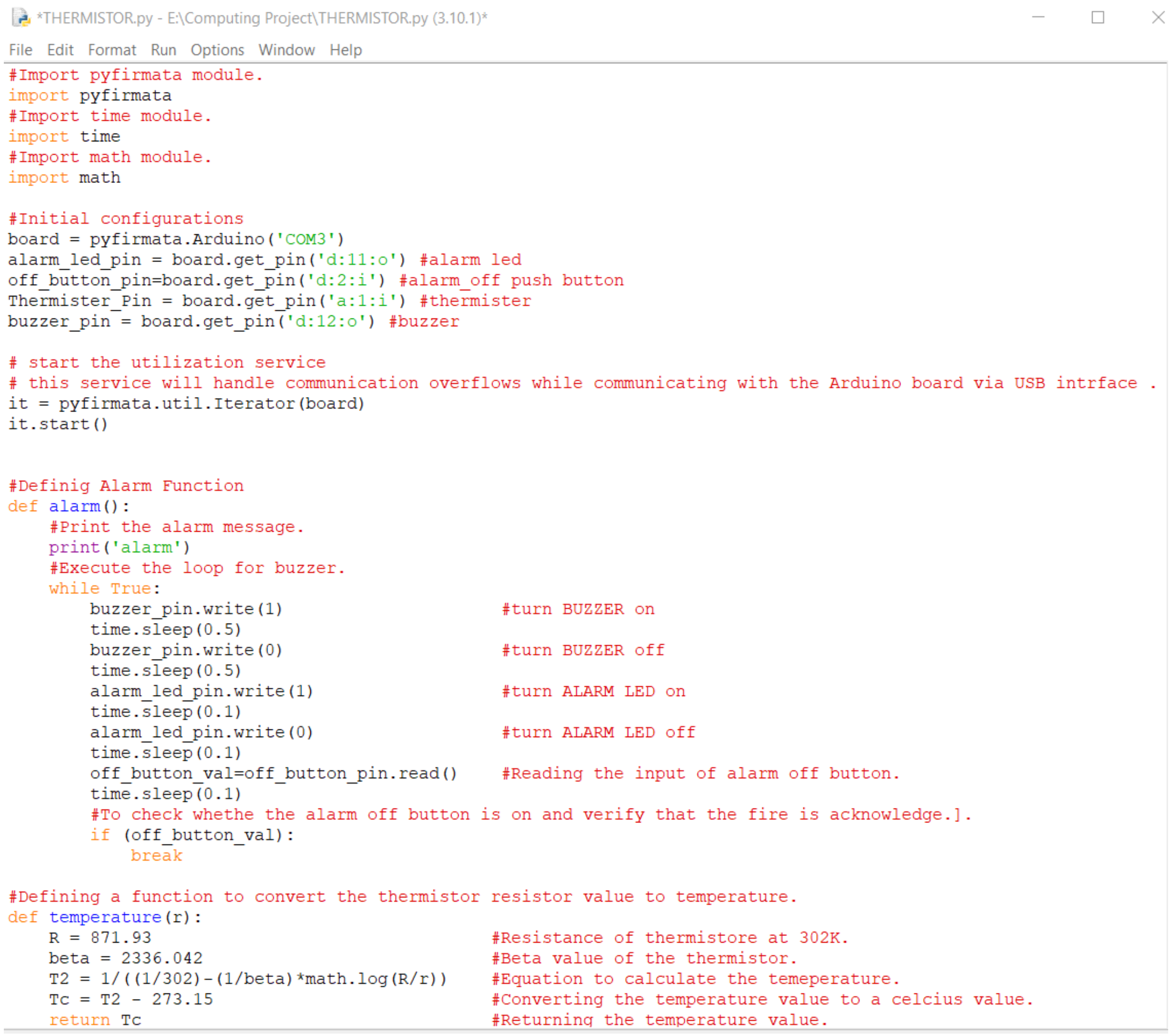
8.1.1 For the fire alarm


Start
R1 = 1000
Finale Temperature = 29
i = 0
Read the thermistor input
R2 = R1 * (1/thermsitor reading)
Temeprature (Using Temperature function where R2 is the parameter)
print Temperature
Temperature > 130
NO
YES
print 'Fire'
Alarm
Temperature > Finale Temperature
NO
YES
Finale Temperature = Temperature
i = i +1
Is i = 10
YES
NO
Print 'Temperature is increasing'
Break
Break

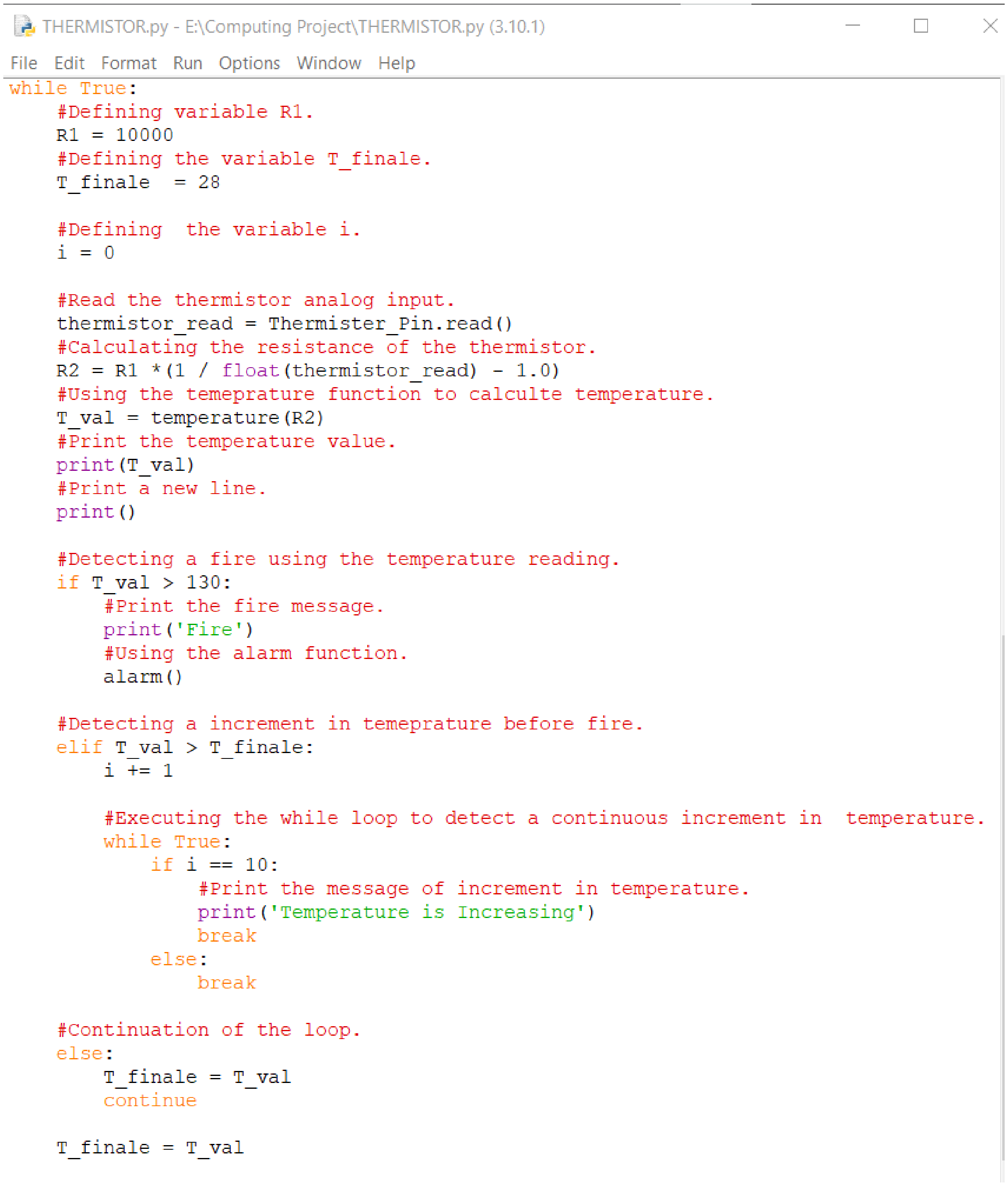
8.1.2 For the temperature function



8.2 Code Snippets for the temperature function


THERMISTOR.py - E:\Computing Project\THERMISTOR.py (3.10.1)
File Edit Format Run Options Window Help

```
#Import pyfirmata module.
import pyfirmata
#Import time module.
import time
#Import math module.
import math

#Initial configurations
board = pyfirmata.Arduino('COM3')
alarm_led_pin = board.get_pin('d:11:o') #alarm led
off_button_pin=board.get_pin('d:2:i') #alarm_off push button
Thermister_Pin = board.get_pin('a:1:i') #thermister
buzzer_pin = board.get_pin('d:12:o') #buzzer

# start the utilization service
# this service will handle communication overflows while communicating with the Arduino board via USB intrface .
it = pyfirmata.util.Iterator(board)
it.start()


#Definig Alarm Function
def alarm():
    #Print the alarm message.
    print('alarm')
    #Execute the loop for buzzer.
    while True:
        buzzer_pin.write(1)                     #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0)                     #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1)                  #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0)                  #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button_pin.read()    #Reading the input of alarm off button.
        time.sleep(0.1)
        #To check whethe the alarm off button is on and verify that the fire is acknowledge.].
        if (off_button_val):
            break

#Defining a function to convert the thermistor resistor value to temperature.
def temperature(r):
    R = 871.93                              #Resistance of thermistore at 302K.
    beta = 2336.042                         #Beta value of the thermistor.
    T2 = 1/((1/302)-(1/beta)*math.log(R/r)) #Equation to calculate the temeperature.
    Tc = T2 - 273.15                        #Converting the temperature value to a celcius value.
    return Tc                               #Returning the temperature value.
```


THERMISTOR.py - E:\Computing Project\THERMISTOR.py (3.10.1)
File Edit Format Run Options Window Help

```
while True:
    #Defining variable R1.
    R1 = 10000
    #Defining the variable T_finale.
    T_finale  = 28

    #Defining  the variable i.
    i = 0

    #Read the thermistor analog input.
    thermistor_read = Thermister_Pin.read()
    #Calculating the resistance of the thermistor.
    R2 = R1 *(1 / float(thermistor_read) - 1.0)
    #Using the temeprature function to calculte temperature.
    T_val = temperature(R2)
    #Print the temperature value.
    print(T_val)
    #Print a new line.
    print()

    #Detecting a fire using the temperature reading.
    if T_val > 130:
        #Print the fire message.
        print('Fire')
        #Using the alarm function.
        alarm()

    #Detecting a increment in temeprature before fire.
    elif T_val > T_finale:
        i += 1

        #Executing the while loop to detect a continuous increment in  temperature.
        while True:
            if i == 10:
                #Print the message of increment in temperature.
                print('Temperature is Increasing')
                break
            else:
                break

    #Continuation of the loop.
    else:
        T_finale = T_val
        continue

    T_finale = T_val
```

8.3 Code:

```
#Import pyfirmata module.
import pyfirmata
#Import time module.
import time
#Import math module.
import math

#Initial configurations
board = pyfirmata.Arduino('COM3')
alarm_led_pin = board.get_pin('d:11:o') #alarm led
off_button_pin=board.get_pin('d:2:i') #alarm_off push button
Thermister_Pin = board.get_pin('a:1:i') #thermister
buzzer_pin = board.get_pin('d:12:o') #buzzer

# start the utilization service
# this service will handle communication overflows while communicating with the Arduino
board via USB intrface .
it = pyfirmata.util.Iterator(board)
it.start()

#Definig Alarm Function
def alarm():
    #Print the alarm message.
    print('alarm')
    #Execute the loop for buzzer.
    while True:
        buzzer_pin.write(1)          #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0)          #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1)              #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0)              #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button_pin.read()    #Reading the input of alarm off button.
        time.sleep(0.1)
        #To check whether the alarm off button is on and verify that the fire is acknowledge.].
        if (off_button_val):
            break

#Defining a function to convert the thermistor resistor value to temperature.
def temperature(r):
    R = 871.93                              #Resistance of thermistore at 302K.
    beta = 2336.042                         #Beta value of the thermistor.
    T2 = 1/((1/302)-(1/beta)*math.log(R/r))    #Equation to calculate the temeperature.
    Tc = T2 - 273.15                        #Converting the temperature value to a celsius value.
    return Tc                               #Returning the temperature value.

#Executing the while true loop.
while True:
    #Defining variable R1.
    R1 = 10000
    #Defining the variable T_finale.
    T_finale  = 28
    #Defining  the variable i.
    i = 0

    #Read the thermistor analog input.
    thermistor_read = Thermister_Pin.read()
    #Calculating the resistance of the thermistor.
    R2 = R1 *(1 / float(thermistor_read) - 1.0)
    #Using the temperature function to calculate temperature.
```

```
T_val = temperature(R2)
#Print the temperature value.
print(T_val)
#Print a new line.
print()

#Detecting a fire using the temperature reading.
if T_val > 130:
    #Print the fire message.
    print('Fire')
    #Using the alarm function.
    alarm()

#Detecting an increment in temperature before fire.
elif T_val > T_finale:
    i += 1

    #Executing the while loop to detect a continuous increment in  temperature.
    while True:
        if i == 10:
            #Print the message of increment in temperature.
            print('Temperature is Increasing')
            break
        else:
            break

#Continuation of the loop.
else:
    T_finale = T_val
    continue

T_finale = T_val
```

8.4 Fire Alarm Function

Temperature is monitored for fire detection in the fire alarm function. Other two functions, the alarm function and the temperature function, are utilized within this function.

Thermistor is the sensor used here to read the temperature.

8.5 Functionality of the thermistor

Thermistors are temperature-dependent resistors, changing resistance with changes in temperature. They are very sensitive and react to very small changes in temperature. They are best used when a specific temperature needs to be maintained.

The Beta(β) value of the thermistor is used here to calculate the temperature. The β value is not a true material constant and is temperature dependent. The Beta(β) value of the given thermistor was calculated by reading the thermistor resistance at room temperature (302K) and at boiling water temperature (373K). The following equation was used for the above calculation.

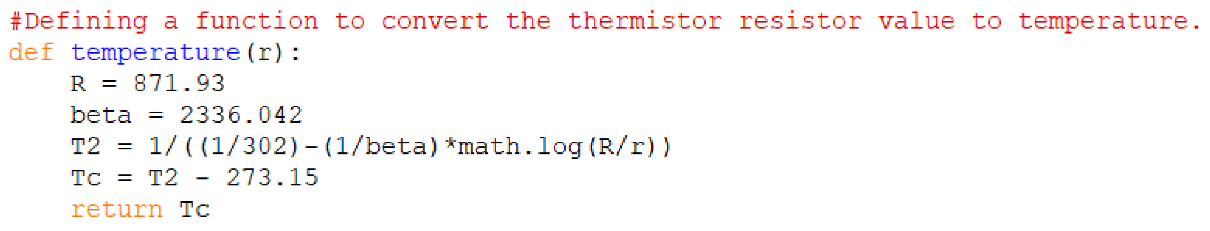
$\beta = \ln(R1/R2) / (1/T1 - 1/T2)$

The β value obtained for the given thermistor is 2336.042. Then by using a different configuration of the above equation temperature can be calculated.

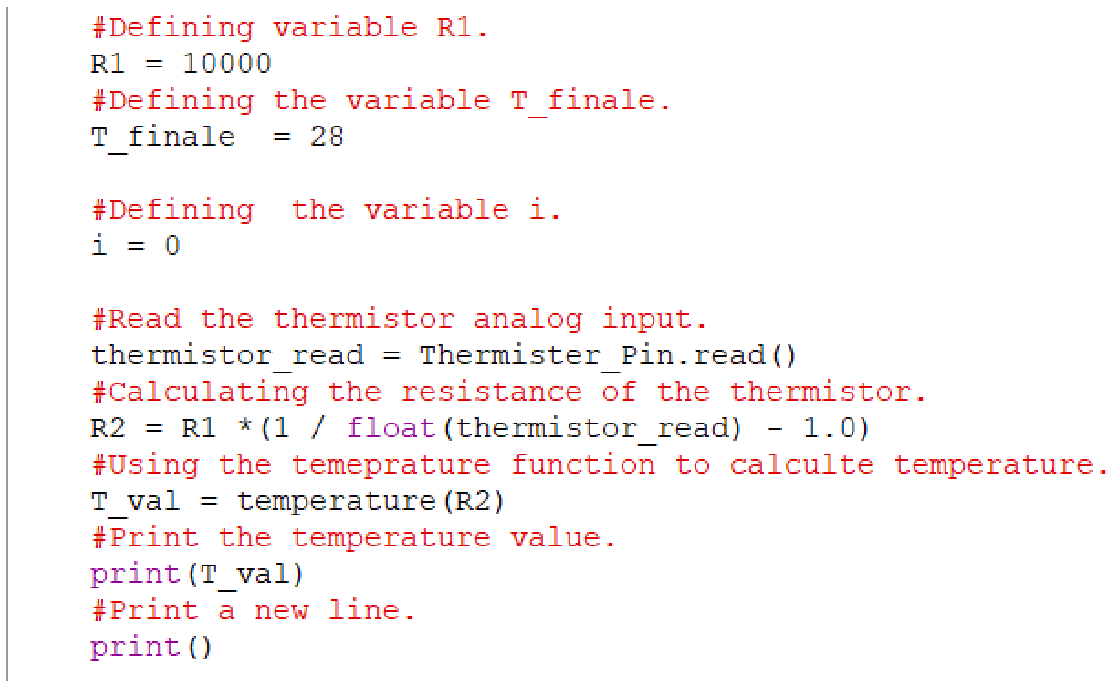
$T2 = 1 / (1/T1 - \ln(R1/R2) * 1/\beta)$

The resistance of the thermistor was determined by reading the analog input voltage of the thermistor. The temperature is then calculated using the temperature function. The acquired resistance value was used as the temperature function's parameter.

8.6 Temperature function



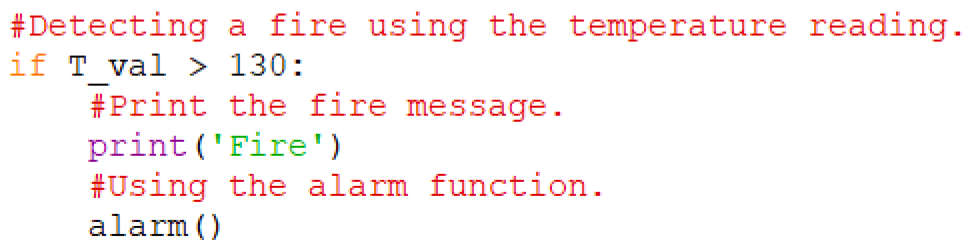
```
#Defining a function to convert the thermistor resistor value to temperature.
def temperature(r):
    R = 871.93
    beta = 2336.042
    T2 = 1/((1/302)-(1/beta)*math.log(R/r))
    Tc = T2 - 273.15
    return Tc
```



```
#Defining variable R1.
R1 = 10000
#Defining the variable T_finale.
T_finale  = 28

#Defining  the variable i.
i = 0

#Read the thermistor analog input.
thermistor_read = Thermister_Pin.read()
#Calculating the resistance of the thermistor.
R2 = R1 *(1 / float(thermistor_read) - 1.0)
#Using the temeprature function to calculte temperature.
T_val = temperature(R2)
#Print the temperature value.
print(T_val)
#Print a new line.
print()
```

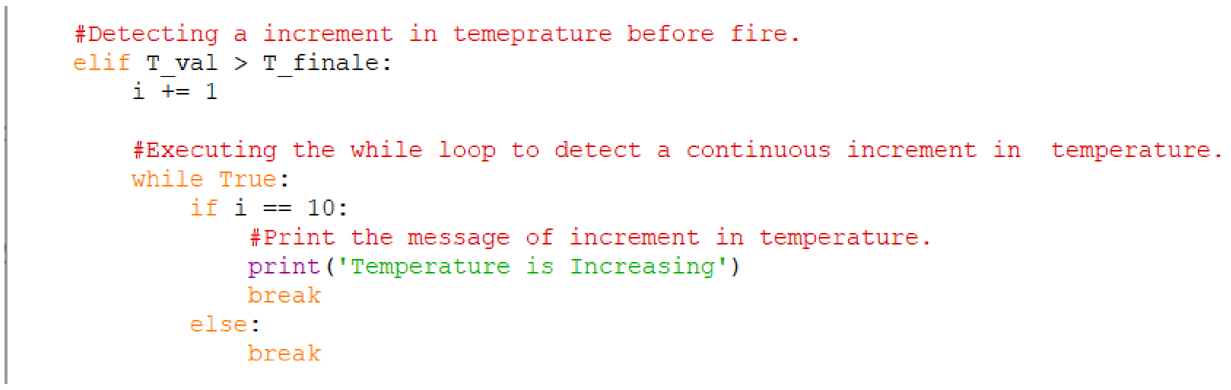
Then the temperature value is printed.

The temperature values are utilized to analyze the present state of the room once the temperature is executed. The control structure is utilized in this case, and it falls under the category of selection. If the temperature in the room rises above 130ºC, it is considered a fire. As a result, the message 'Fire' is printed, and the alarm function is activated.



```
#Detecting a fire using the temperature reading.
if T_val > 130:
    #Print the fire message.
    print('Fire')
    #Using the alarm function.
    alarm()
```

Aside from that, if the room's temperature is consistently raised, it is noticed as a temperature increase in the room. Temperature variation is checked five times in this room. When the temperature is increased ten times in a row, it is recognized as an increase in temperature and the message 'Temperature is Increasing' is printed.



```
#Detecting a increment in temeprature before fire.
elif T_val > T_finale:
    i += 1

    #Executing the while loop to detect a continuous increment in  temperature.
    while True:
        if i == 10:
            #Print the message of increment in temperature.
            print('Temperature is Increasing')
            break
        else:
            break
```

8.7 Alarm function

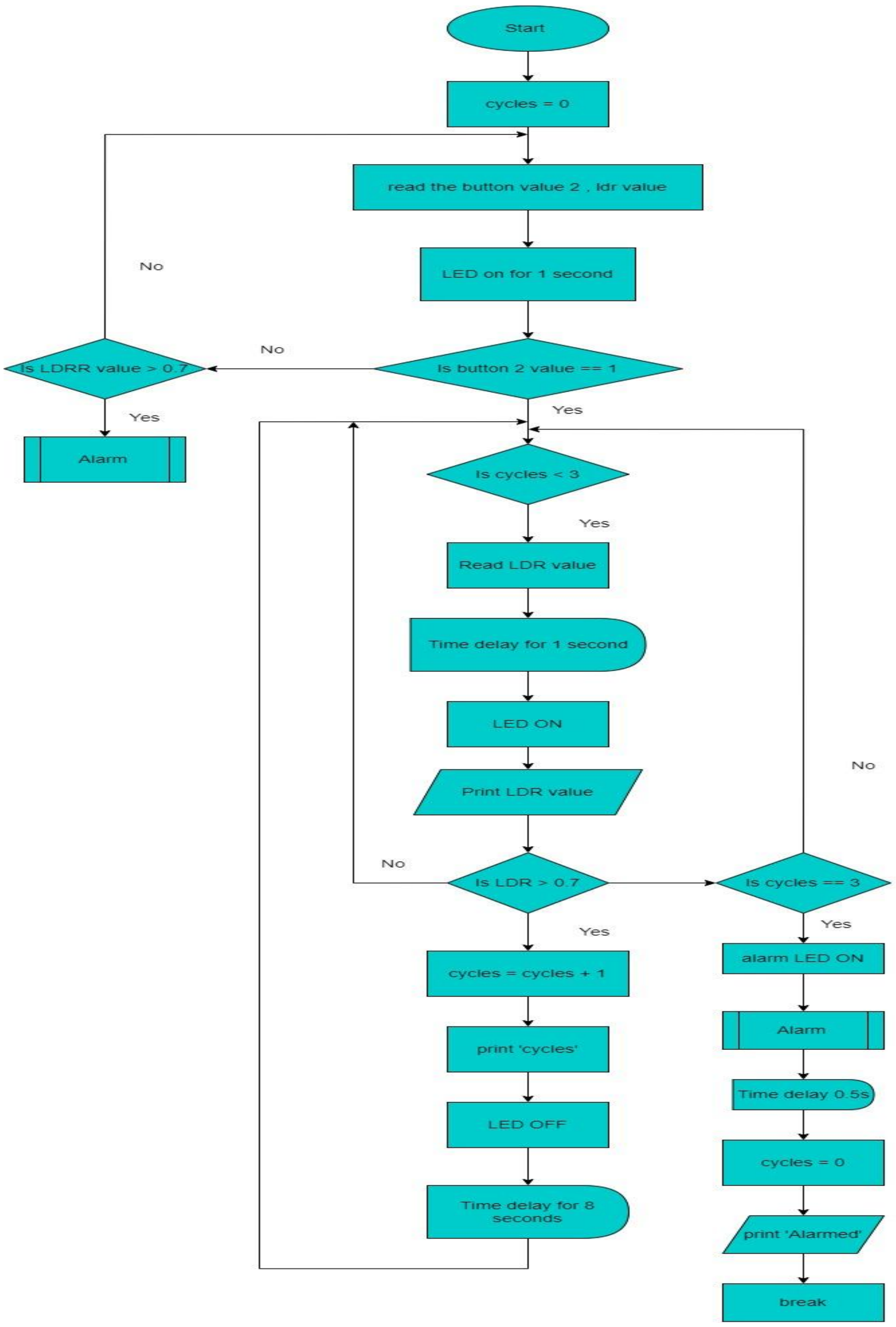
When a fire is detected, the alarm function is activated. A while loop is employed here as a control structure that falls under the category of repetition. When a fire is detected, the Piezo buzzer and Alarm LED are activated inside the while loop. The loop can be broken by turning on the alarm off push button once the fire detection message has been received and the necessary measures to put out the fire have been followed.

```
##defining the alarm and the alarm led
def alarm():
    #print the alarm message
    print('alarm')
    #execute the loopfor buzzer
    while True:
        buzzer_pin.write(1)     #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0)     #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading alarm off button
        if (off_button_val):
            break
```

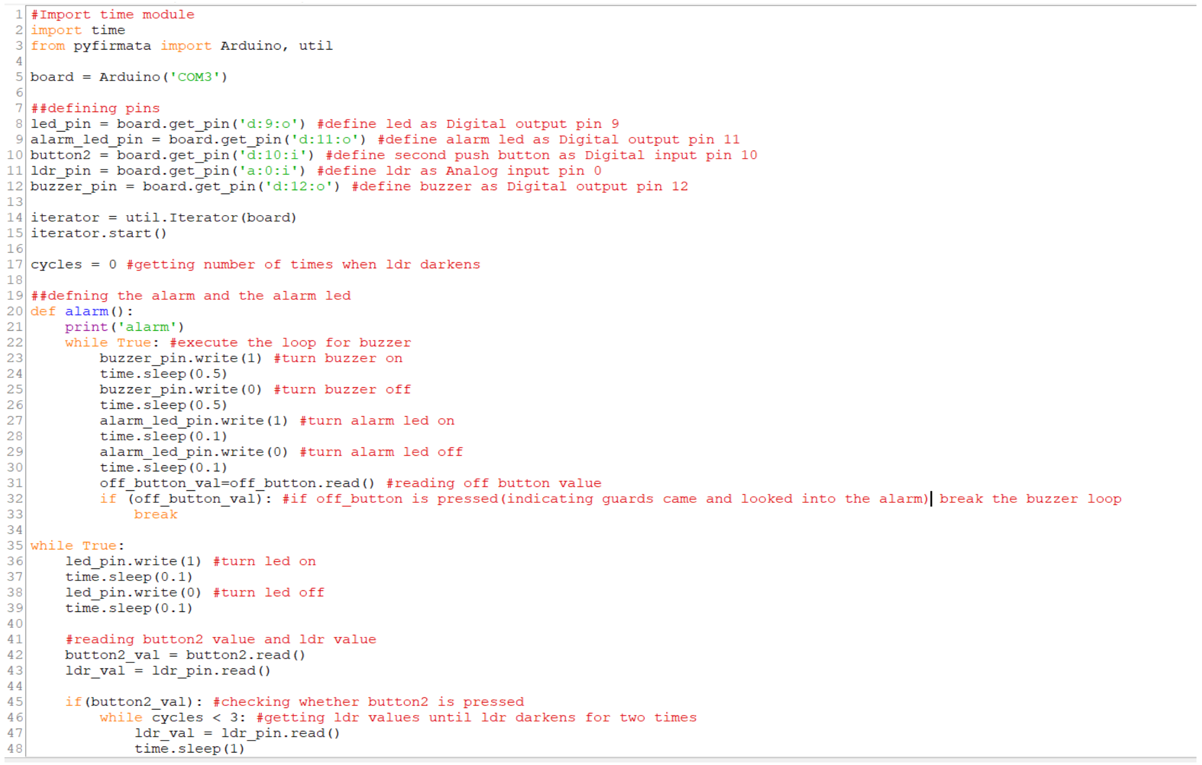
9. Light intensity

9.1 Flowchart

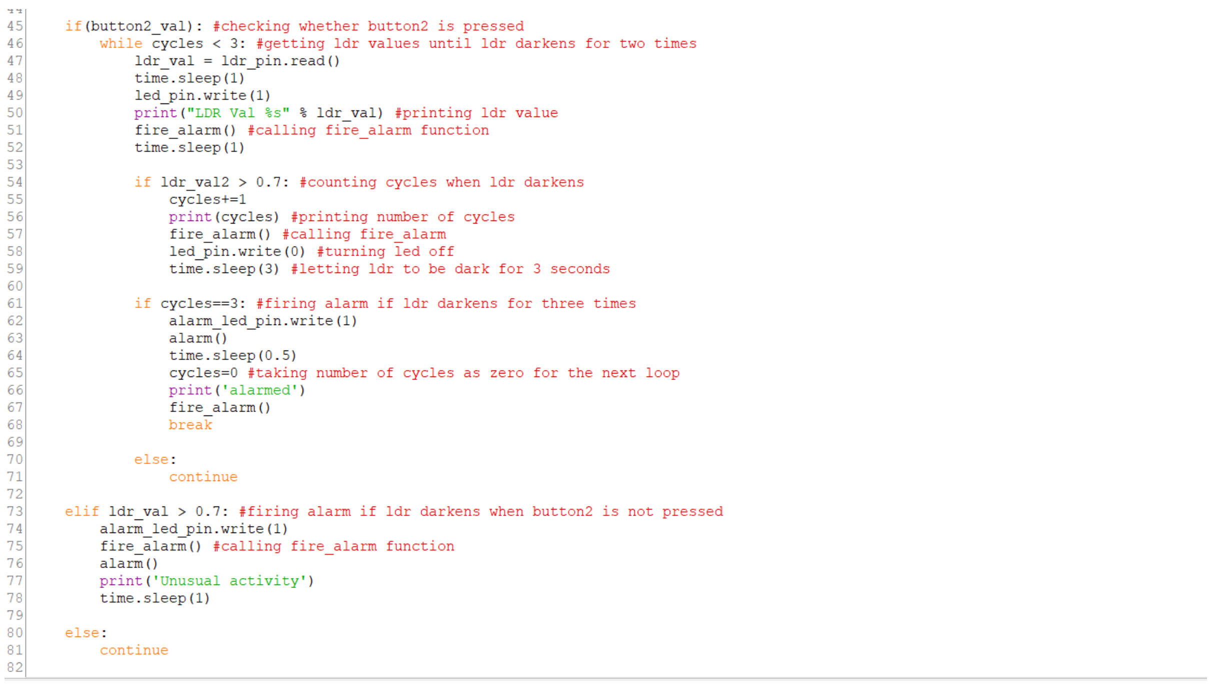
This flowchart is to show monitoring unusual activities using light intensity. Here, it is coded to let the ldr darken for two times, for three seconds each, after the second button is pressed. Which means, when guards enter, they are allowed to take documents (it is then, the ldr darkens.) two times for the maximum. If ldr darkens for three times, it means outsiders have come and then the alarm fires until the guards come and press the off button. If ldr darkens when the second button is not pressed, it also means unusual activity, and then the alarm fires. (Here, the second button is used for the ease of the code instead of guards entering using secret codes within 30 seconds.)


Start
cycles = 0
read the button value 2 , ldr value
LED on for 1 second
Is button 2 value == 1
No
Is LDRR value > 0.7
No
Yes
Alarm
Yes
Is cycles < 3
Yes
Read LDR value
Time delay for 1 second
LED ON
Print LDR value
Is LDR > 0.7
No
Is cycles == 3
No
Yes
Yes
cycles = cycles + 1
print 'cycles'
LED OFF
Time delay for 8 seconds
alarm LED ON
Alarm
Time delay 0.5s
cycles = 0
print 'Alarmed'
break

9.2 Code Snippets for Light Intensity



```python
#Import time module
import time
from pyfirmata import Arduino, util

board = Arduino('COM3')

##defining pins
led_pin = board.get_pin('d:9:o') #define led as Digital output pin 9
alarm_led_pin = board.get_pin('d:11:o') #define alarm led as Digital output pin 11
button2 = board.get_pin('d:10:i') #define second push button as Digital input pin 10
ldr_pin = board.get_pin('a:0:i') #define ldr as Analog input pin 0
buzzer_pin = board.get_pin('d:12:o') #define buzzer as Digital output pin 12

iterator = util.Iterator(board)
iterator.start()

cycles = 0 #getting number of times when ldr darkens

##defning the alarm and the alarm led
def alarm():
    print('alarm')
    while True: #execute the loop for buzzer
        buzzer_pin.write(1) #turn buzzer on
        time.sleep(0.5)
        buzzer_pin.write(0) #turn buzzer off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn alarm led on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn alarm led off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading off button value
        if (off_button_val): #if off_button is pressed(indicating guards came and looked into the alarm) break the buzzer loop
            break

while True:
    led_pin.write(1) #turn led on
    time.sleep(0.1)
    led_pin.write(0) #turn led off
    time.sleep(0.1)

    #reading button2 value and ldr value
    button2_val = button2.read()
    ldr_val = ldr_pin.read()

    if(button2_val): #checking whether button2 is pressed
        while cycles < 3: #getting ldr values until ldr darkens for two times
            ldr_val = ldr_pin.read()
            time.sleep(1)
```



```python
    if(button2_val): #checking whether button2 is pressed
        while cycles < 3: #getting ldr values until ldr darkens for two times
            ldr_val = ldr_pin.read()
            time.sleep(1)
            led_pin.write(1)
            print("LDR Val %s" % ldr_val) #printing ldr value
            fire_alarm() #calling fire_alarm function
            time.sleep(1)

            if ldr_val2 > 0.7: #counting cycles when ldr darkens
                cycles+=1
                print(cycles) #printing number of cycles
                fire_alarm() #calling fire_alarm
                led_pin.write(0) #turning led off
                time.sleep(3) #letting ldr to be dark for 3 seconds

            if cycles==3: #firing alarm if ldr darkens for three times
                alarm_led_pin.write(1)
                alarm()
                time.sleep(0.5)
                cycles=0 #taking number of cycles as zero for the next loop
                print('alarmed')
                fire_alarm()
                break

            else:
                continue

    elif ldr_val > 0.7: #firing alarm if ldr darkens when button2 is not pressed
        alarm_led_pin.write(1)
        fire_alarm() #calling fire_alarm function
        alarm()
        print('Unusual activity')
        time.sleep(1)

    else:
        continue
```

9.3 Code:

```
#Import time module
import time
from pyfirmata import Arduino, util

board = Arduino('COM3')

##defining pins
led_pin = board.get_pin('d:9:o') #define led as Digital output pin 9
alarm_led_pin = board.get_pin('d:11:o') #define alarm led as Digital output pin 11
button2 = board.get_pin('d:10:i') #define second push button as Digital input pin 10
ldr_pin = board.get_pin('a:0:i') #define ldr as Analog input pin 0
buzzer_pin = board.get_pin('d:12:o') #define buzzer as Digital output pin 12

iterator = util.Iterator(board)
iterator.start()

cycles = 0 #getting number of times when ldr darkens

##defining the alarm and the alarm led
def alarm():
    print('alarm')
    while True: #execute the loop for buzzer
        buzzer_pin.write(1) #turn buzzer on
        time.sleep(0.5)
        buzzer_pin.write(0) #turn buzzer off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn alarm led on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn alarm led off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading off button value
        if (off_button_val): #if off_button is pressed(indicating guards came and looked into the
alarm) break the buzzer loop
            break

while True:
    led_pin.write(1) #turn led on
    time.sleep(0.1)
    led_pin.write(0) #turn led off
    time.sleep(0.1)

    #reading button2 value and ldr value
    button2_val = button2.read()
    ldr_val = ldr_pin.read()

    if(button2_val): #checking whether button2 is pressed
        while cycles < 3: #getting ldr values until ldr darkens for two times
            ldr_val = ldr_pin.read()
            time.sleep(1)
            led_pin.write(1)
            print("LDR Val %s" % ldr_val) #printing ldr value
            fire_alarm() #calling fire_alarm function
            time.sleep(1)

            if ldr_val2 > 0.7: #counting cycles when ldr darkens
                cycles+=1
                print(cycles) #printing number of cycles
                fire_alarm() #calling fire_alarm
                led_pin.write(0) #turning led off
                time.sleep(3) #letting ldr to be dark for 3 seconds
            if cycles==3: #firing alarm if ldr darkens for three times
                alarm_led_pin.write(1)
                alarm()
                time.sleep(0.5)
                cycles=0 #taking number of cycles as zero for the next loop
                print('alarmed')
                fire_alarm()
                break

            else:
                continue

    elif ldr_val > 0.7: #firing alarm if ldr darkens when button2 is not pressed
        alarm_led_pin.write(1)
        fire_alarm() #calling fire_alarm function
        alarm()
        print('Unusual activity')
        time.sleep(1)

    else:
        continue
```

9.4 Functionality of LDR

A light-dependent resistor, or LDR, is an electrical component that responds to light. When light rays strike it, the resistance changes very instantly. An LDR's resistance levels can vary by several orders of magnitude. When the light level rises, the resistance value decreases.

10. Secret entry sequence

Access to the room using a secret entry sequence using two push buttons and a keypad for the 3 security clearance categories.

10.1 Flowchart:

There are 3 security clearance categories namely, confidential, secret and top secret. For these categories there are unique and different codes to enter within a time limit. The time limit to enter the both two codes is 30 seconds. When entering the room only the same security category people can enter the code.

Confidential category

- Code 1 = A31A2
- Code 2 = 33A21

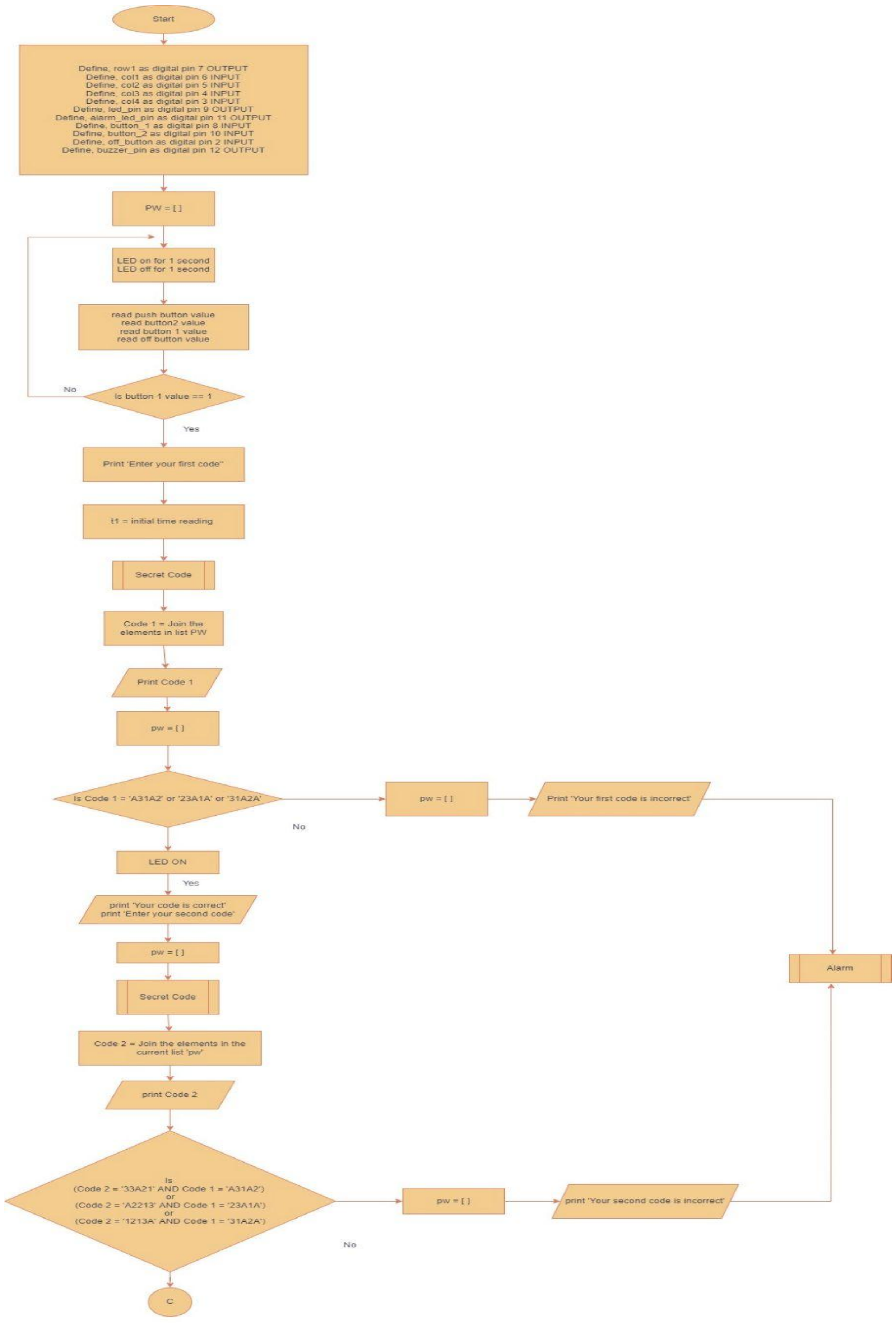
Secret category

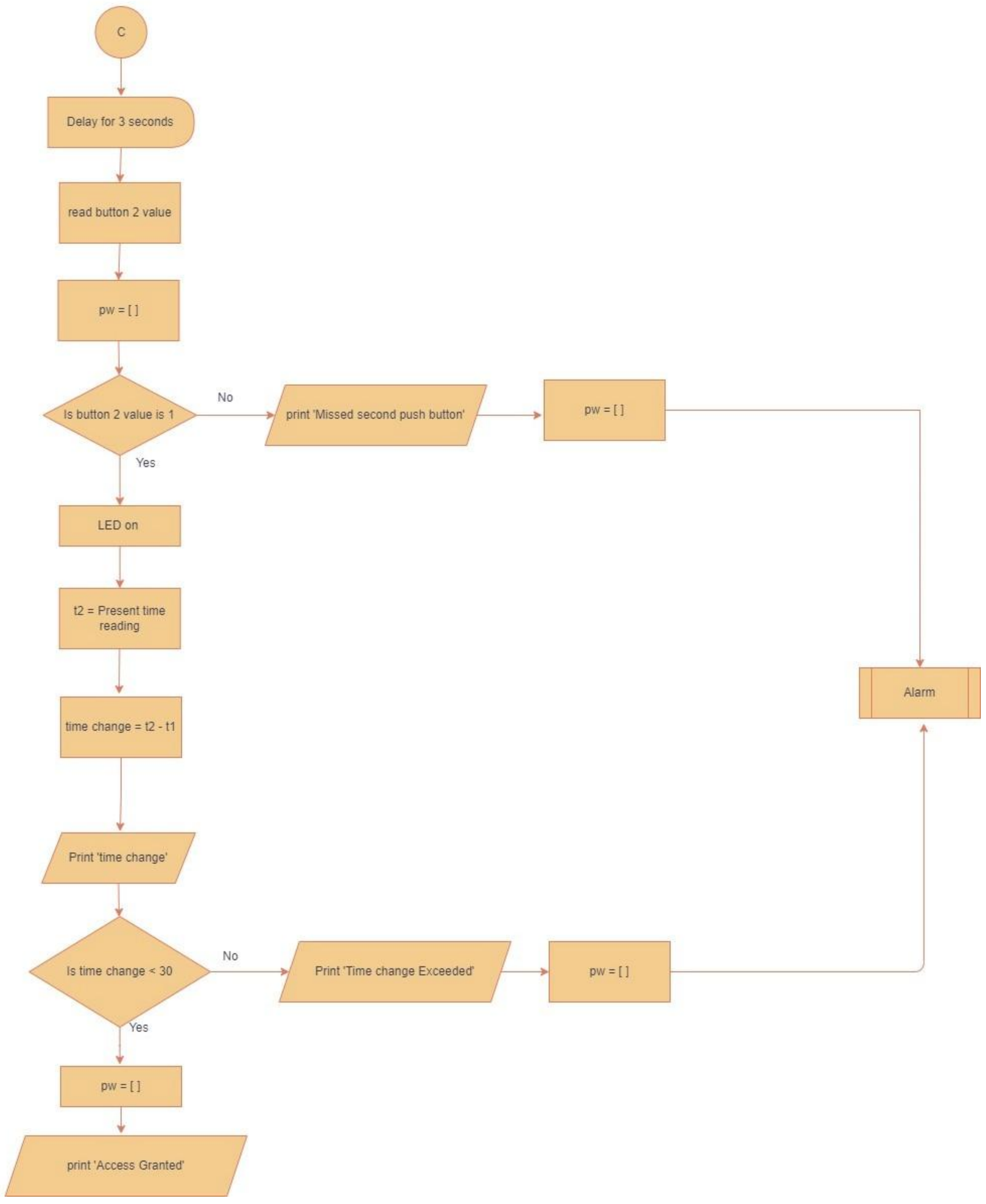
- Code 1 = 23A1A
- Code 2 = A2213

Top Secret category

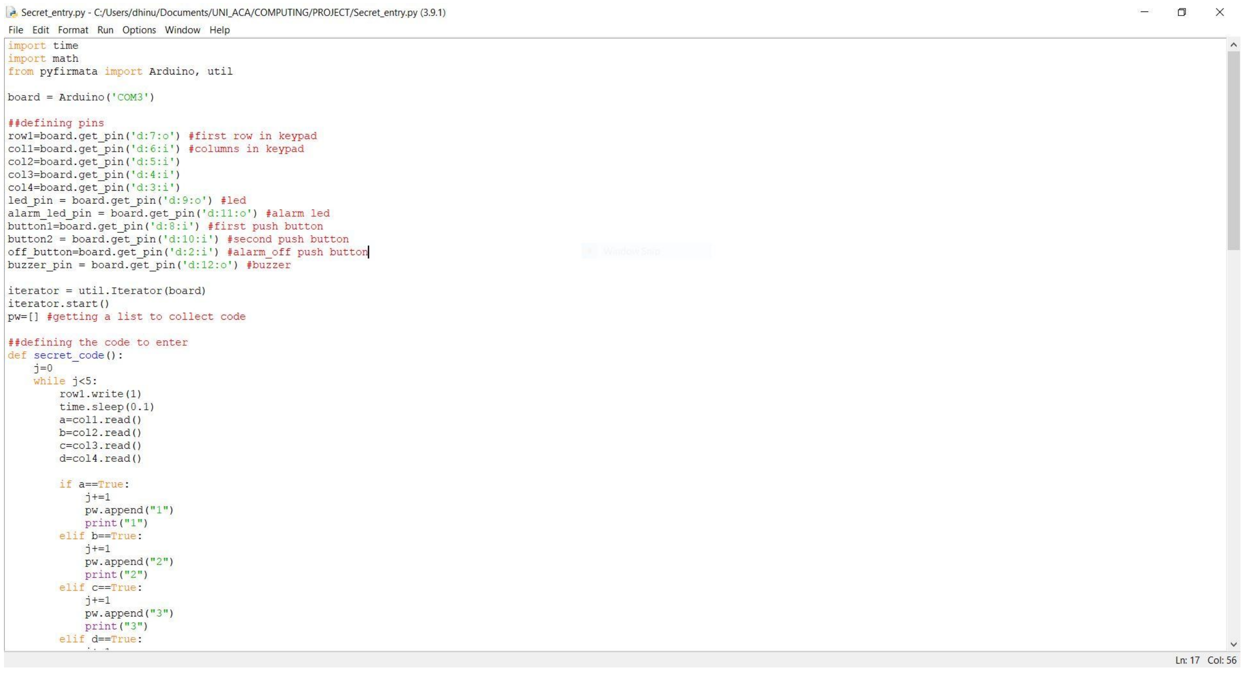
- Code 1 = 31A2A
- Code 2 = 1213A

To get the secret code, a keypad is used and to this in a time limit two push buttons are used.

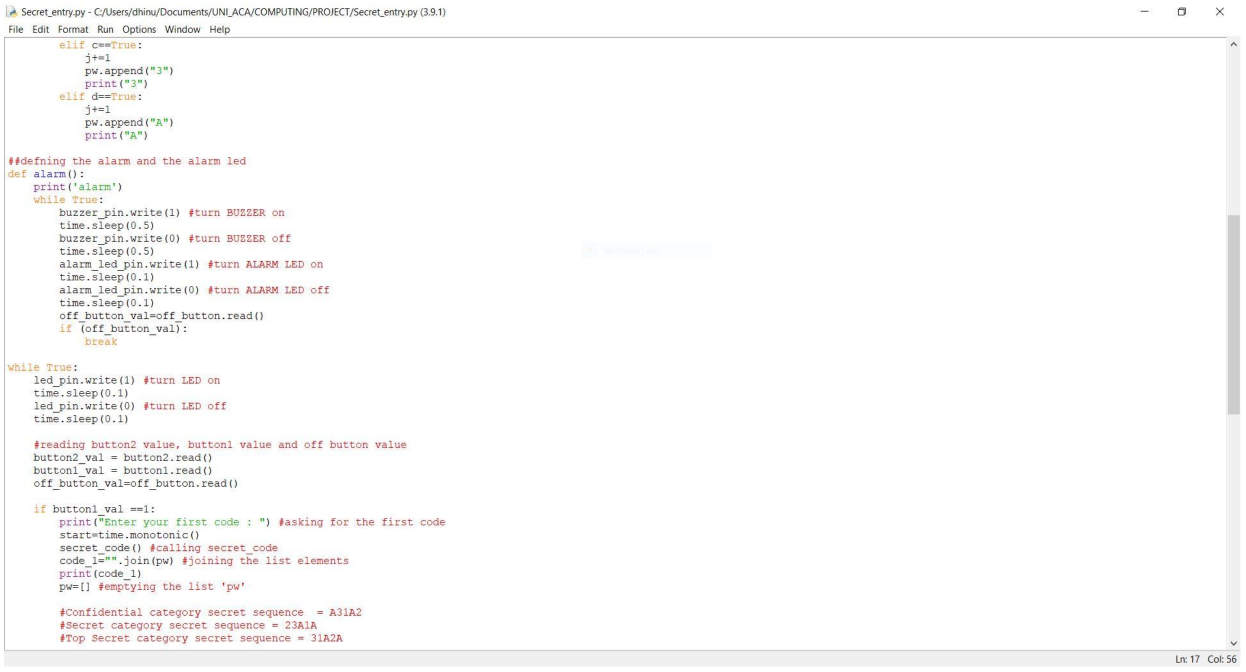

Start
Define, row1 as digital pin 7 OUTPUT
Define, col1 as digital pin 6 INPUT
Define, col2 as digital pin 5 INPUT
Define, col3 as digital pin 4 INPUT
Define, col4 as digital pin 3 INPUT
Define, led_pin as digital pin 9 OUTPUT
Define, alarm_led_pin as digital pin 11 OUTPUT
Define, button_1 as digital pin 8 INPUT
Define, button_2 as digital pin 10 INPUT
Define, off_button as digital pin 2 INPUT
Define, buzzer_pin as digital pin 12 OUTPUT
PW = [ ]
LED on for 1 second
LED off for 1 second
read push button value
read button2 value
read button 1 value
read off button value
No
Is button 1 value == 1
Yes
Print 'Enter your first code"
t1 = initial time reading
Secret Code
Code 1 = Join the elements in list PW
Print Code 1
pw = [ ]
Is Code 1 = 'A31A2' or '23A1A' or '31A2A'
pw = [ ]
Print 'Your first code is incorrect'
No
LED ON
Yes
print 'Your code is correct'
print 'Enter your second code'
pw = [ ]
Alarm
Secret Code
Code 2 = Join the elements in the current list 'pw'
print Code 2
Is
(Code 2 = '33A21' AND Code 1 = 'A31A2')
or
(Code 2 = 'A2213' AND Code 1 = '23A1A')
or
(Code 2 = '1213A' AND Code 1 = '31A2A')
pw = [ ]
print 'Your second code is incorrect'
No
C


C
Delay for 3 seconds
read button 2 value
pw = [ ]
Is button 2 value is 1
No
print 'Missed second push button'
pw = [ ]
Yes
LED on
t2 = Present time reading
time change = t2 - t1
Print 'time change'
Is time change < 30
No
Print 'Time change Exceeded'
pw = [ ]
Alarm
Yes
pw = [ ]
print 'Access Granted'

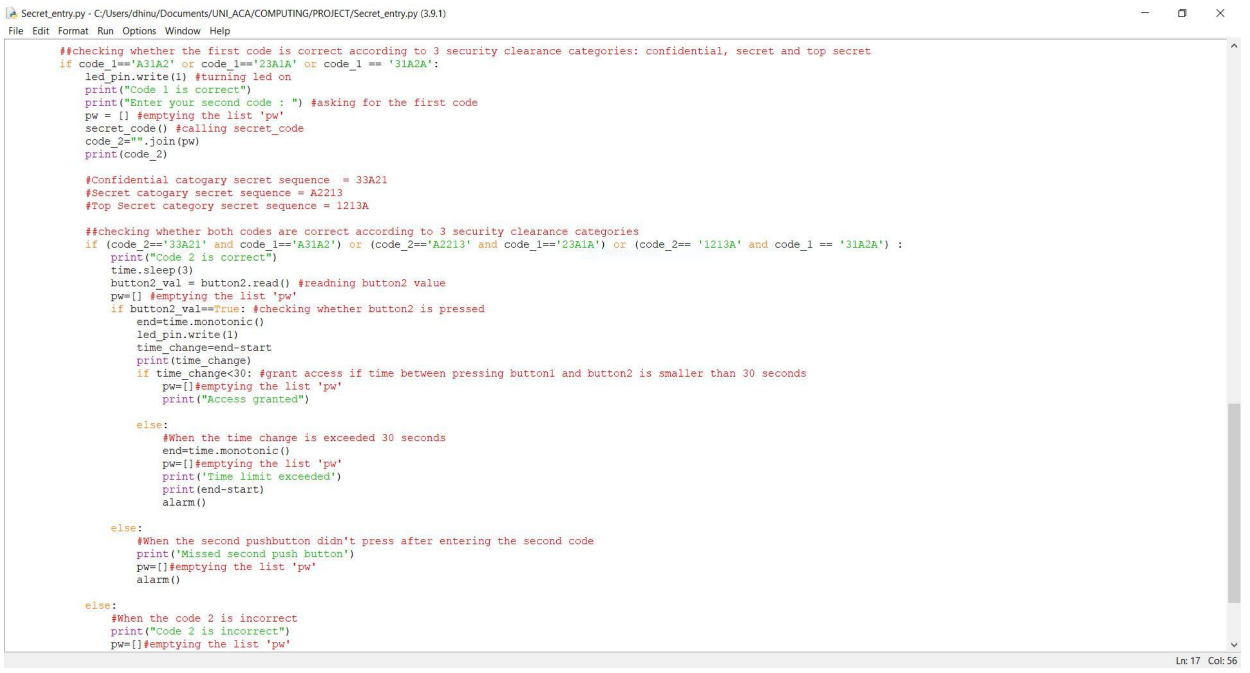
10.2 Code snippets for secret sequence entry


Secret_entry.py - C:/Users/dhinu/Documents/UNI_ACA/COMPUTING/PROJECT/Secret_entry.py (3.9.1)
File Edit Format Run Options Window Help
Ln: 17 Col: 56

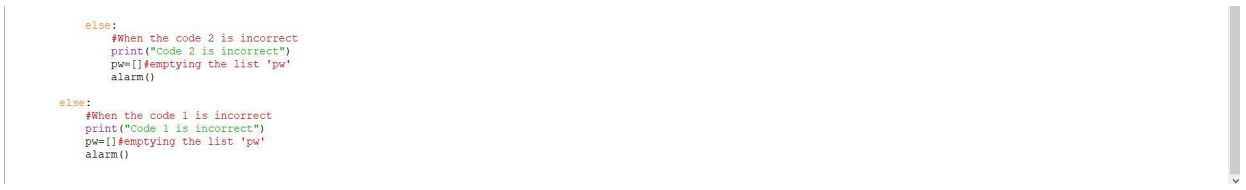
```python
import time
import math
from pyfirmata import Arduino, util

board = Arduino('COM3')

##defining pins
row1=board.get_pin('d:7:o') #first row in keypad
col1=board.get_pin('d:6:i') #columns in keypad
col2=board.get_pin('d:5:i')
col3=board.get_pin('d:4:i')
col4=board.get_pin('d:3:i')
led_pin = board.get_pin('d:9:o') #led
alarm_led_pin = board.get_pin('d:11:o') #alarm led
button1=board.get_pin('d:8:i') #first push button
button2 = board.get_pin('d:10:i') #second push button
off_button=board.get_pin('d:2:i') #alarm_off push button
buzzer_pin = board.get_pin('d:12:o') #buzzer

iterator = util.Iterator(board)
iterator.start()
pw=[] #getting a list to collect code

##defining the code to enter
def secret_code():
    j=0
    while j<5:
        row1.write(1)
        time.sleep(0.1)
        a=col1.read()
        b=col2.read()
        c=col3.read()
        d=col4.read()

        if a==True:
            j+=1
            pw.append("1")
            print("1")
        elif b==True:
            j+=1
            pw.append("2")
            print("2")
        elif c==True:
            j+=1
            pw.append("3")
            print("3")
        elif d==True:
```


Secret_entry.py - C:/Users/dhinu/Documents/UNI_ACA/COMPUTING/PROJECT/Secret_entry.py (3.9.1)
File Edit Format Run Options Window Help
Ln: 17 Col: 56

```python
        elif c==True:
            j+=1
            pw.append("3")
            print("3")
        elif d==True:
            j+=1
            pw.append("A")
            print("A")

##defning the alarm and the alarm led
def alarm():
    print('alarm')
    while True:
        buzzer_pin.write(1) #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0) #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button.read()
        if (off_button_val):
            break

while True:
    led_pin.write(1) #turn LED on
    time.sleep(0.1)
    led_pin.write(0) #turn LED off
    time.sleep(0.1)

    #reading button2 value, button1 value and off button value
    button2_val = button2.read()
    button1_val = button1.read()
    off_button_val=off_button.read()

    if button1_val ==1:
        print("Enter your first code : ") #asking for the first code
        start=time.monotonic()
        secret_code() #calling secret_code
        code_1="".join(pw) #joining the list elements
        print(code_1)
        pw=[] #emptying the list 'pw'

        #Confidential category secret sequence  = A31A2
        #Secret category secret sequence = 23A1A
        #Top Secret category secret sequence = 31A2A
```


Secret_entry.py - C:/Users/dhinu/Documents/UNI_ACA/COMPUTING/PROJECT/Secret_entry.py (3.9.1)
File Edit Format Run Options Window Help
Ln: 17 Col: 56

```
        ##checking whether the first code is correct according to 3 security clearance categories: confidential, secret and top secret
        if code_1=='A31A2' or code_1=='23A1A' or code_1 == '31A2A':
            led_pin.write(1) #turning led on
            print("Code 1 is correct")
            print("Enter your second code : ") #asking for the first code
            pw = [] #emptying the list 'pw'
            secret_code() #calling secret_code
            code_2="".join(pw)
            print(code_2)

            #Confidential catogary secret sequence  = 33A21
            #Secret catogary secret sequence = A2213
            #Top Secret category secret sequence = 1213A

            ##checking whether both codes are correct according to 3 security clearance categories
            if (code_2=='33A21' and code_1=='A31A2') or (code_2=='A2213' and code_1=='23A1A') or (code_2== '1213A' and code_1 == '31A2A') :
                print("Code 2 is correct")
                time.sleep(3)
                button2_val = button2.read() #readning button2 value
                pw=[] #emptying the list 'pw'
                if button2_val==True: #checking whether button2 is pressed
                    end=time.monotonic()
                    led_pin.write(1)
                    time_change=end-start
                    print(time_change)
                    if time_change<30: #grant access if time between pressing button1 and button2 is smaller than 30 seconds
                        pw=[]#emptying the list 'pw'
                        print("Access granted")

                    else:
                        #When the time change is exceeded 30 seconds
                        end=time.monotonic()
                        pw=[]#emptying the list 'pw'
                        print('Time limit exceeded')
                        print(end-start)
                        alarm()

                else:
                    #When the second pushbutton didn't press after entering the second code
                    print('Missed second push button')
                    pw=[]#emptying the list 'pw'
                    alarm()

            else:
                #When the code 2 is incorrect
                print("Code 2 is incorrect")
                pw=[]#emptying the list 'pw'
```



```
            else:
                #When the code 2 is incorrect
                print("Code 2 is incorrect")
                pw=[]#emptying the list 'pw'
                alarm()

        else:
            #When the code 1 is incorrect
            print("Code 1 is incorrect")
            pw=[]#emptying the list 'pw'
            alarm()
```

10.3 Code:

```
import time
import math
from pyfirmata import Arduino, util

board = Arduino('COM3')

##defining pins
row1=board.get_pin('d:7:o') #first row in keypad
col1=board.get_pin('d:6:i') #columns in keypad
col2=board.get_pin('d:5:i')
col3=board.get_pin('d:4:i')
col4=board.get_pin('d:3:i')
led_pin = board.get_pin('d:9:o') #led
alarm_led_pin = board.get_pin('d:11:o') #alarm led
button1=board.get_pin('d:8:i') #first push button
button2 = board.get_pin('d:10:i') #second push button
off_button=board.get_pin('d:2:i') #alarm_off push button
buzzer_pin = board.get_pin('d:12:o') #buzzer

iterator = util.Iterator(board)
iterator.start()
pw=[] #getting a list to collect code

##defining the code to enter
def secret_code():
    j=0
    while j<5:
        row1.write(1)
        time.sleep(0.1)
        a=col1.read()
        b=col2.read()
        c=col3.read()
        d=col4.read()

        if a==True:
            j+=1
            pw.append("1")
            print("1")
        elif b==True:
            j+=1
            pw.append("2")
            print("2")
        elif c==True:
            j+=1
            pw.append("3")
            print("3")
        elif d==True:
            j+=1
            pw.append("A")
            print("A")

##defining the alarm and the alarm led
def alarm():
    print('alarm')
    while True:
        buzzer_pin.write(1) #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0) #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button.read()
        if (off_button_val):
            break

while True:
    led_pin.write(1) #turn LED on
    time.sleep(0.1)
    led_pin.write(0) #turn LED off
    time.sleep(0.1)

    #reading button2 value, button1 value and off button value
    button2_val = button2.read()
    button1_val = button1.read()
    off_button_val=off_button.read()

    if button1_val ==1:
        print("Enter your first code : ") #asking for the first code
        start=time.monotonic()
        secret_code() #calling secret_code
        code_1="".join(pw) #joining the list elements
        print(code_1)
        pw=[] #emptying the list 'pw'

        #Confidential category secret sequence  = A31A2
        #Secret category secret sequence = 23A1A
        #Top Secret category secret sequence = 31A2A

        ##checking whether the first code is correct according to 3 security clearance categories:
confidential, secret and top secret
        if code_1=='A31A2' or code_1=='23A1A' or code_1 == '31A2A':
            led_pin.write(1) #turning led on
            print("Code 1 is correct")
            print("Enter your second code : ") #asking for the first code
            pw = [] #emptying the list 'pw'
            secret_code() #calling secret_code
            code_2="".join(pw)
            print(code_2)

            #Confidential category secret sequence  = 33A21
            #Secret category secret sequence = A2213
            #Top Secret category secret sequence = 1213A

            ##checking whether both codes are correct according to 3 security clearance categories
            if (code_2=='33A21' and code_1=='A31A2') or (code_2=='A2213' and code_1=='23A1A') or
(code_2== '1213A' and code_1 == '31A2A') :
            print("Code 2 is correct")
            time.sleep(3)
            button2_val = button2.read() #reading button2 value
            pw=[] #emptying the list 'pw'
            if button2_val==True: #checking whether button2 is pressed
                end=time.monotonic()
                led_pin.write(1)
                time_change=end-start
                print(time_change)
                if time_change<30: #grant access if time between pressing button1 and button2 is
smaller than 30 seconds
                    pw=[]#emptying the list 'pw'
                    print("Access granted")

                else:
                    #When the time change is exceeded 30 seconds
                    end=time.monotonic()
                    pw=[]#emptying the list 'pw'
                    print('Time limit exceeded')
                    print(end-start)
                    alarm()

            else:
                #When the second pushbutton didn't press after entering the second code
                print('Missed second push button')
                pw=[]#emptying the list 'pw'
                alarm()

            else:
            #When the code 2 is incorrect
            print("Code 2 is incorrect")
            pw=[]#emptying the list 'pw'
            alarm()

    else:
            #When the code 1 is incorrect
            print("Code 1 is incorrect")
            pw=[]#emptying the list 'pw'
            alarm()
```

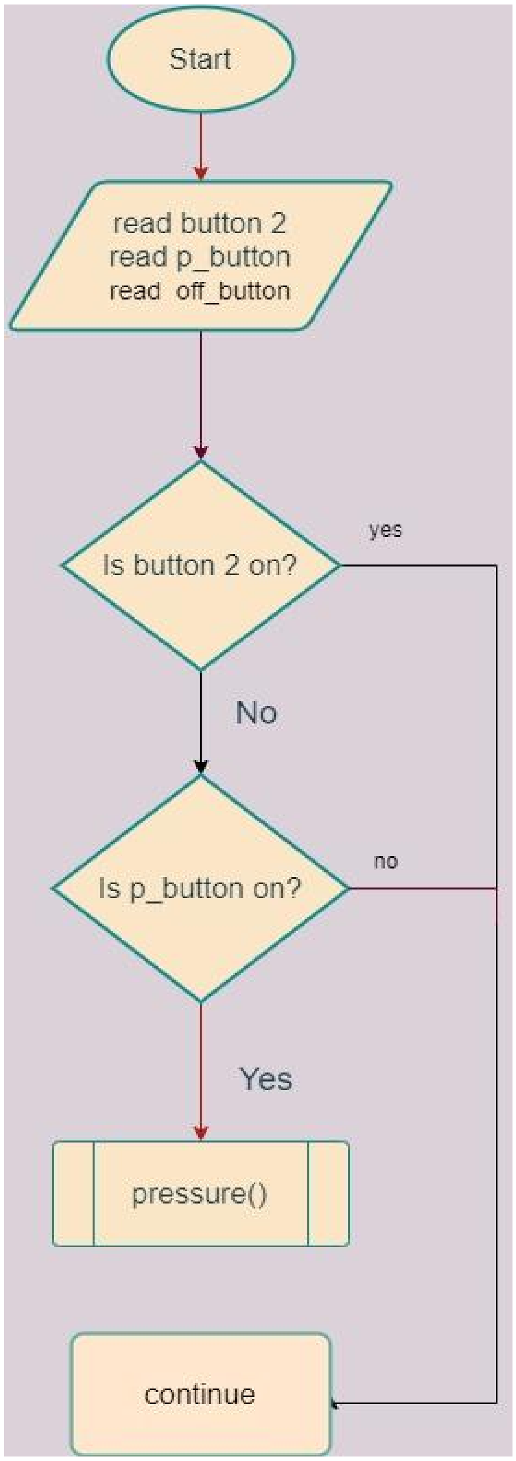
10.4 Explanation - code of secret sequence entry

To get the access to enter the room two people from the same category must come together. First , the first push button must be pressed at that moment time starts to count. After entering the fist code correctly by the first person and then the second code must be entered by the second person. After entering both codes correctly the second push button should be pressed at that moment how much time has passed to press the second push button after the first push button is counted (time_change). The two codes must be entered in between 30 seconds. If the time limit exceeds the alarm function works. When all the codes are correctly entered according to the time limit two persons are allowed to enter the room.

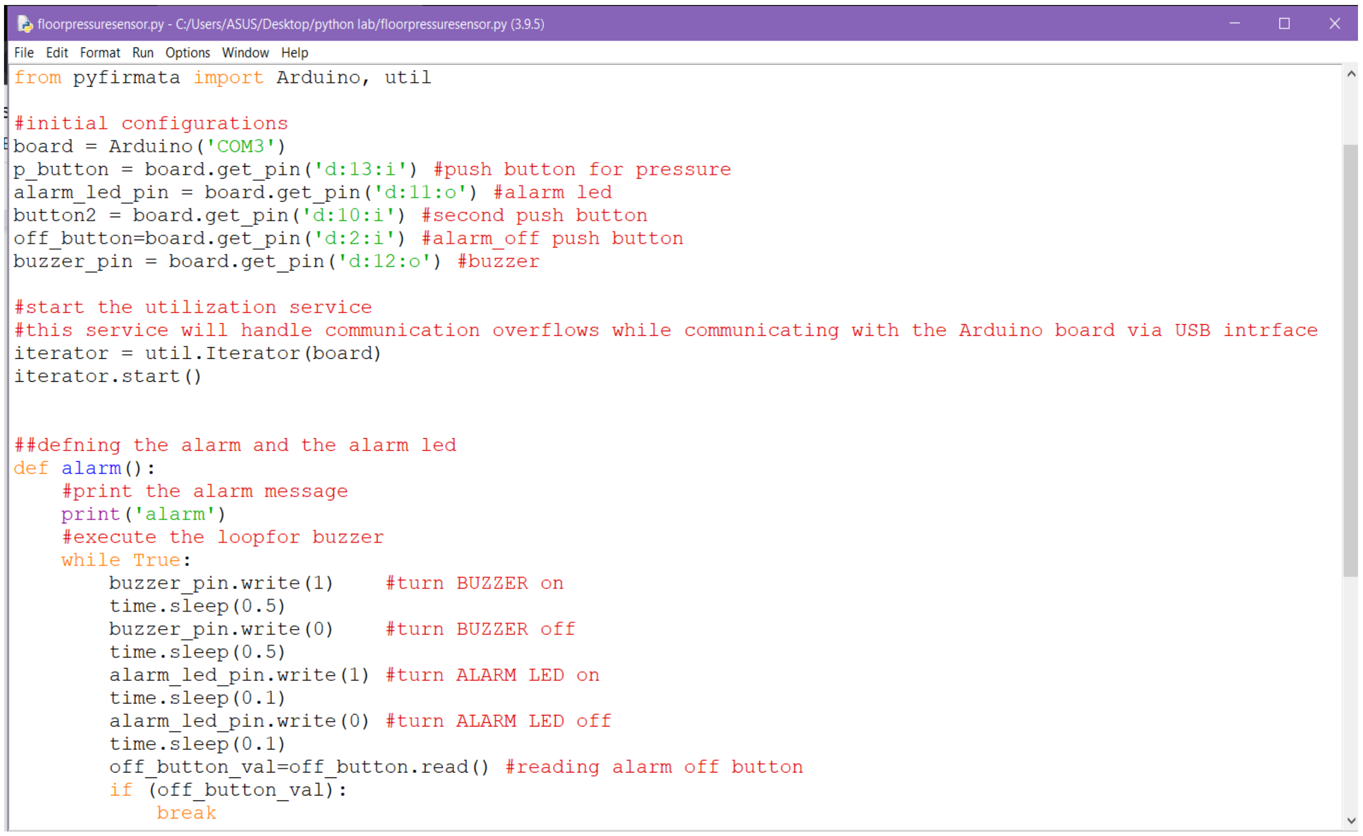
11. Floor Pressure

Floor pressure sensors are installed inorder to detect if there is any unusual behavior. Those sensors are only activated when the guards are not inside. In other terms, it is only activated all the time except when an authorized person enters. It works when the second push button is off, when the second push button is activated it means an authorized person is inside the room. Here a push button acts as floor pressure sensor and when it is turned on alarm led and the buzzer turn on and the system goes directly to a lockdown. When guards are not in the room the second push button is off , so the pressure sensors are activated and if an unauthorized person comes inside now the push button turns on and both led , buzzer goes on.

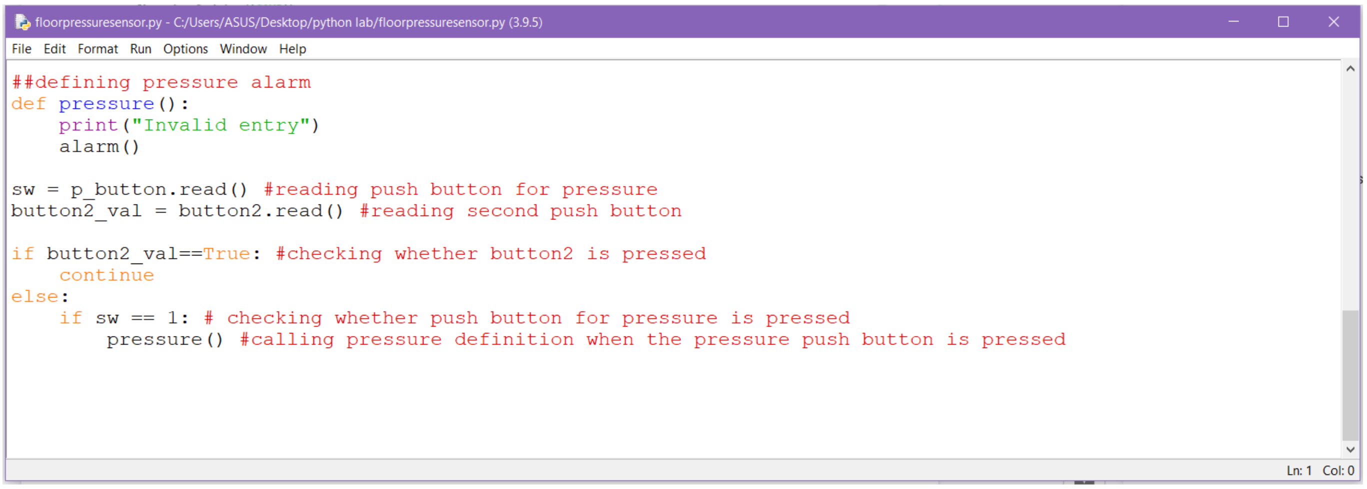
11.1 Flowchart for floor pressure sensor



11.2 Code snippets for floor pressure sensor


floorpressuresensor.py - C:/Users/ASUS/Desktop/python lab/floorpressuresensor.py (3.9.5)
File Edit Format Run Options Window Help

```
from pyfirmata import Arduino, util

#initial configurations
board = Arduino('COM3')
p_button = board.get_pin('d:13:i') #push button for pressure
alarm_led_pin = board.get_pin('d:11:o') #alarm led
button2 = board.get_pin('d:10:i') #second push button
off_button=board.get_pin('d:2:i') #alarm_off push button
buzzer_pin = board.get_pin('d:12:o') #buzzer

#start the utilization service
#this service will handle communication overflows while communicating with the Arduino board via USB intrface
iterator = util.Iterator(board)
iterator.start()


##defning the alarm and the alarm led
def alarm():
    #print the alarm message
    print('alarm')
    #execute the loopfor buzzer
    while True:
        buzzer_pin.write(1)     #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0)     #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading alarm off button
        if (off_button_val):
            break
```


floorpressuresensor.py - C:/Users/ASUS/Desktop/python lab/floorpressuresensor.py (3.9.5)
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
##defining pressure alarm
def pressure():
    print("Invalid entry")
    alarm()

sw = p_button.read() #reading push button for pressure
button2_val = button2.read() #reading second push button

if button2_val==True: #checking whether button2 is pressed
    continue
else:
    if sw == 1: # checking whether push button for pressure is pressed
        pressure() #calling pressure definition when the pressure push button is pressed
```

11.3 Code:

```
#import time module
import time
#import math module
import math
#import pyfirmata module
from pyfirmata import Arduino, util

#initial configurations
board = Arduino('COM3')
p_button = board.get_pin('d:13:i') #push button for pressure
alarm_led_pin = board.get_pin('d:11:o') #alarm led
button2 = board.get_pin('d:10:i') #second push button
off_button=board.get_pin('d:2:i') #alarm_off push button
buzzer_pin = board.get_pin('d:12:o') #buzzer

#start the utilization service
#this service will handle communication overflows while communicating with the Arduino
board via USB interface
iterator = util.Iterator(board)
iterator.start()

##defining the alarm and the alarm led
def alarm():
    #print the alarm message
    print('alarm')
    #execute the loopfor buzzer
    while True:
        buzzer_pin.write(1)    #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0)    #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1) #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0) #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button.read() #reading alarm off button
        if (off_button_val):
            break

##defining pressure alarm
def pressure():
    print("Invalid entry")
    alarm()

sw = p_button.read() #reading push button for pressure
button2_val = button2.read() #reading second push button

if button2_val==True: #checking whether button2 is pressed
    continue
else:
    if sw == 1: # checking whether push button for pressure is pressed
        pressure() #calling pressure definition when the pressure push button is pressed
```

11.4 Alarm function and pressure function

As we already know that the alarm led and the buzzer is activated when the second push button is on and the push button for pressure is on. For this process we have used the alarm() and pressure() functions.

11.4.1 pressure() function

```
##defining pressure alarm
def pressure():
    print("Invalid entry")
    alarm()
```

11.4.2 alarm() function

```
##defining the alarm and the alarm led
def alarm():
    #print the alarm message
    print('alarm')
    #execute the loopfor buzzer
    while True:
        buzzer_pin.write(1)     #turn BUZZER on
        time.sleep(0.5)
        buzzer_pin.write(0)     #turn BUZZER off
        time.sleep(0.5)
        alarm_led_pin.write(1)  #turn ALARM LED on
        time.sleep(0.1)
        alarm_led_pin.write(0)  #turn ALARM LED off
        time.sleep(0.1)
        off_button_val=off_button.read()  #reading alarm off button
        if (off_button_val):
            break
```

When the conditions are true(both second push button and the push button for pressure is on) it goes through pressure()
function and then through alarm() function. When an unusual behavior is detected, the Piezo buzzer and Alarm LED are activated inside the while loop.

12. References

1. Python, R. (n.d.). *Arduino With Python: How to Get Started – Real Python*. [online] realpython.com. Available at: https://realpython.com/arduino-python/.

2. www.youtube.com. (n.d.). *Control Arduino with Python using Firmata / PyFirmata*. [online] Available at: https://www.youtube.com/watch?v=KPfBOGjJdqE&t=277s&ab_channel=KevinMcAleer

3. Arduino | 23, K.P. | (2017). *How to Set Up a Keypad on an Arduino*. [online] Circuit Basics. Available at: https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/.

4. www.itopen.it. (2012). *Arduino pyfirmata LDR semaphore | Open Web Solutions, GIS & Python Development*. [online] Available at: https://www.itopen.it/arduino-pyfirmata-ldr-semaphore/