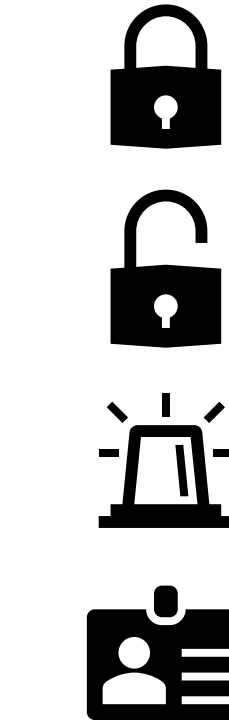
Security System for PRESIDENT'S OFFICE of THE PENTAGON



GP 106 Project – Group 6B

E/19/008 ADHIKARI R.A.J.C.
E/19/009 ADIKARI A.M.K.M.
E/19/018 AMARASEKARA K.G.E.I.
E/19/023 ANUPAMA J.L.

Introduction

The pentagon consists of many important divisions. One of those is the president's office. The president's office has got so many security features. However, our objective is to simplify the most vital features into a scaled down version of this particular president's office. The key features that are to be included in this scaled down version are the fire alarm, a secret knock code for entrance and a panic button for the president for moments of crisis. The following flowchart will clarify how it is being done in this miniature level.

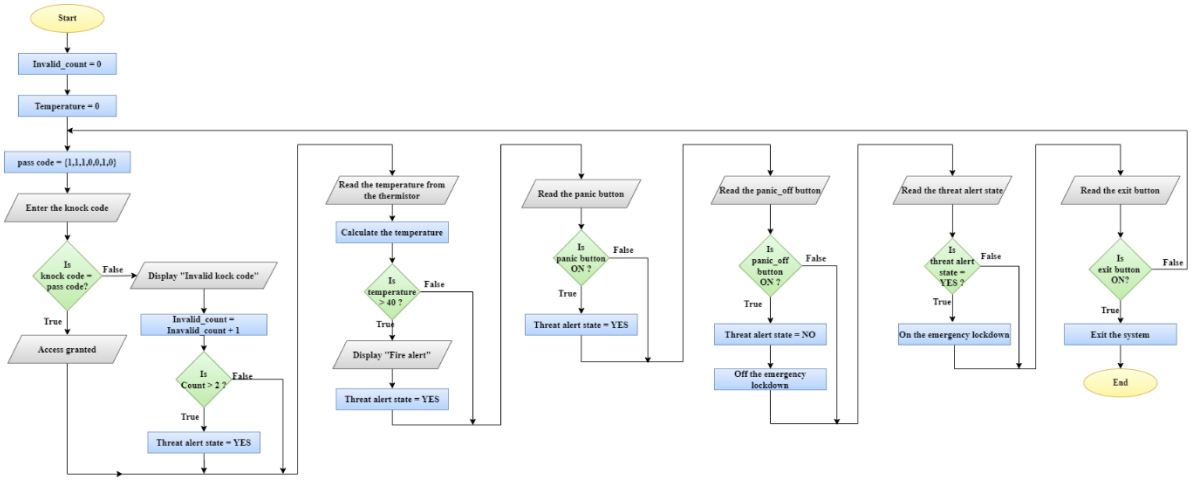


Figure 1 : Flowchart for the general overview of the Algorithm

The Setup

The key components used for this project are an Arduino UNO board, some push buttons, jumper wires, a thermistor, some resistors, LEDs and a piezo buzzer along with a breadboard for connections and a laptop. All the necessary connections have been shown in the circuit diagram below.

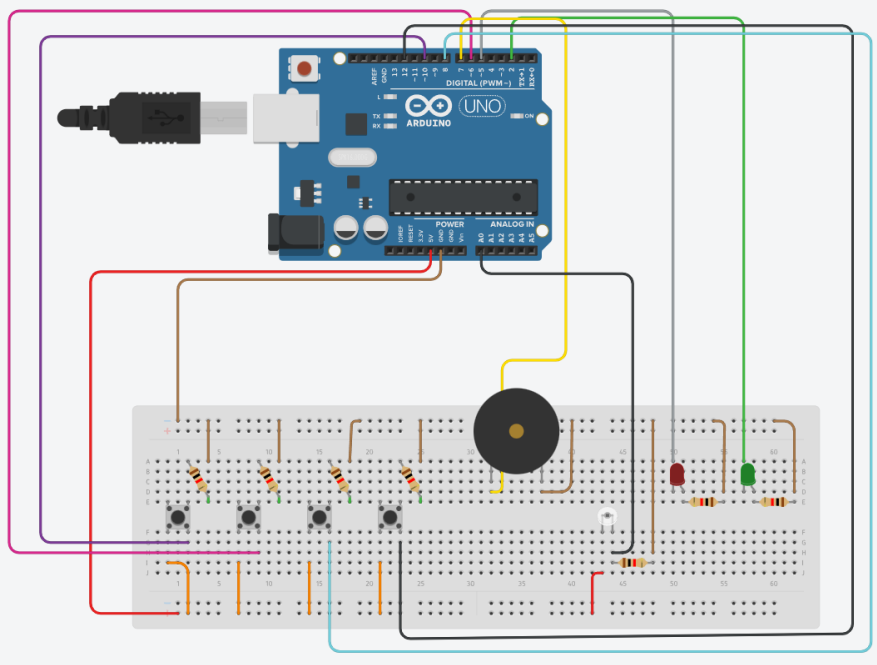


Figure 2 : TinkerCAD model for hardware implementation of the system

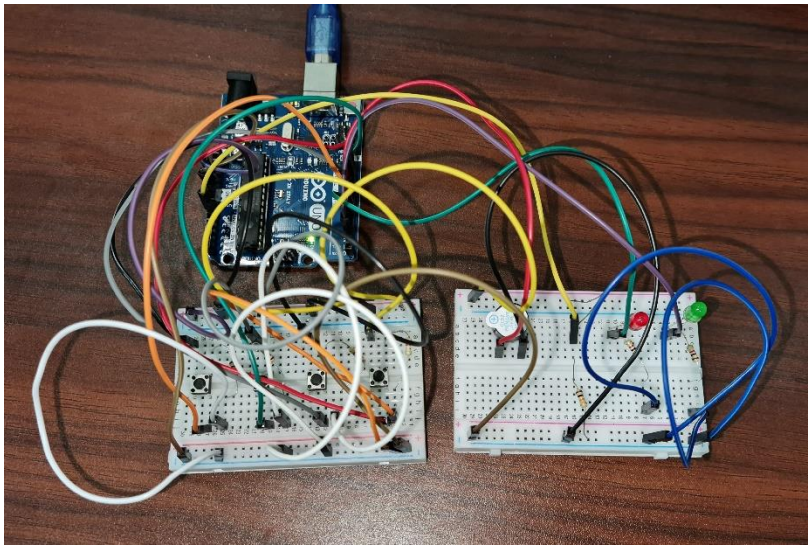


Figure 3 : Physically Created Circuit

Keeping the assembly aside, it is now ideal to start developing the program responsible for performing the intended tasks, The program was written using the python programming language using standard libraries including pyfirmata so that the code can run on the Arduino platform. First, the required libraries for the program are imported which include Arduino from pyfirmata, time, numpy and turtle. Each of these libraries are imported for a specific purpose. Pyfirmata to interact with the Arduino board, time to deal with certain parameters which will be explained further in the latter part, numpy for numerical aspects and turtle to create a display

```
#import libraries
from pyfirmata import Arduino, util, INPUT, OUTPUT
import time
from numpy import log
import turtle
```

Then the board is setup as follows once the libraries are imported.

```
#setup board
board = Arduino('COM7')
iterator = util.Iterator(board)
iterator.start()
```

Once the board is setup, the required pins for the functioning of the system are defined. Each pin is set to read or write a single device for a particular function. Altogether 13 pins have been used one of them (A0) being an analog pin for the thermistor.

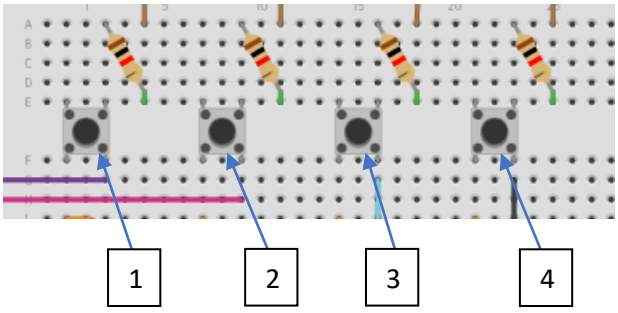
```
#setup pins
thermister = board.analog[0]                #thermister pin
Fire_alert_LED = board.digital[5]           #red LED pin
Fire_alert_buzzer = board.digital[7]        #buzzer pin
Fire_alert_off = board.digital[6]           #push button 1
knocking_button1 = board.digital[12]        #push button 4
knocking_button2 = board.digital[8]         #push button 3
Threat_alert_LED = board.digital[5]         #red LED pin
Giving_Access_LED = board.digital[2]        #green LED pin
Threat_buzzer = board.digital[7]            #buzzer pin
Threat_alert_off = board.digital[6]         #push button 1
Emergency_lockdown = board.digital[3]
panic_button = board.digital[10]            #push button 2
panic_off_button = board.digital[6]         #push button 1
```

Next the pin modes of the pins selected or the interaction they are supposed to make with the devices is defined as follows.

```
#set pin modes
thermister.mode = INPUT
Fire_alert_LED.mode = OUTPUT
Fire_alert_buzzer.mode = OUTPUT
Fire_alert_off.mode = INPUT
knocking_button1.mode = INPUT
knocking_button2.mode = INPUT
Threat_alert_LED.mode = OUTPUT
Threat_buzzer.mode = OUTPUT
Threat_alert_off.mode = INPUT
Emergency_lockdown.mode = OUTPUT
panic_button.mode = INPUT
panic_off_button.mode = INPUT
Giving_Access_LED.mode = OUTPUT
```

The push buttons and the thermistor are used as input devices and the rest are output devices.

In total there are 4 push buttons used



- For knock code,
  Push button 3 & 4
- Panic button – push button 2
- Panic off button – push button 1
- System exit button – push button 3

Apart from the hardware, the system also consists of a display which conveys all the necessary instructions/information regarding access to the president's office and also overall security. For this purpose, a display has been created using turtle graphics. The display was setup as follows.

```
#setup display with turtle graphics
wn = turtle.Screen()
wn.title('President Office')
wn.bgcolor('LightSkyBlue4')
wn.setup(width = 800, height = 800)
wn.tracer(0)
```

To display the necessary information a pen was also created using turtle graphics.

```
#create pen for display with turtle graphics
pen = turtle.Turtle()
pen.speed(0)
pen.shape('square')
pen.color('white')
```

Thereafter quite separately some global variables are defined that would come in handy in the latter part of the code.

```
#set invalid count to zero
invalid_count = 0
#Set temperature value to zero
temp = 0
```

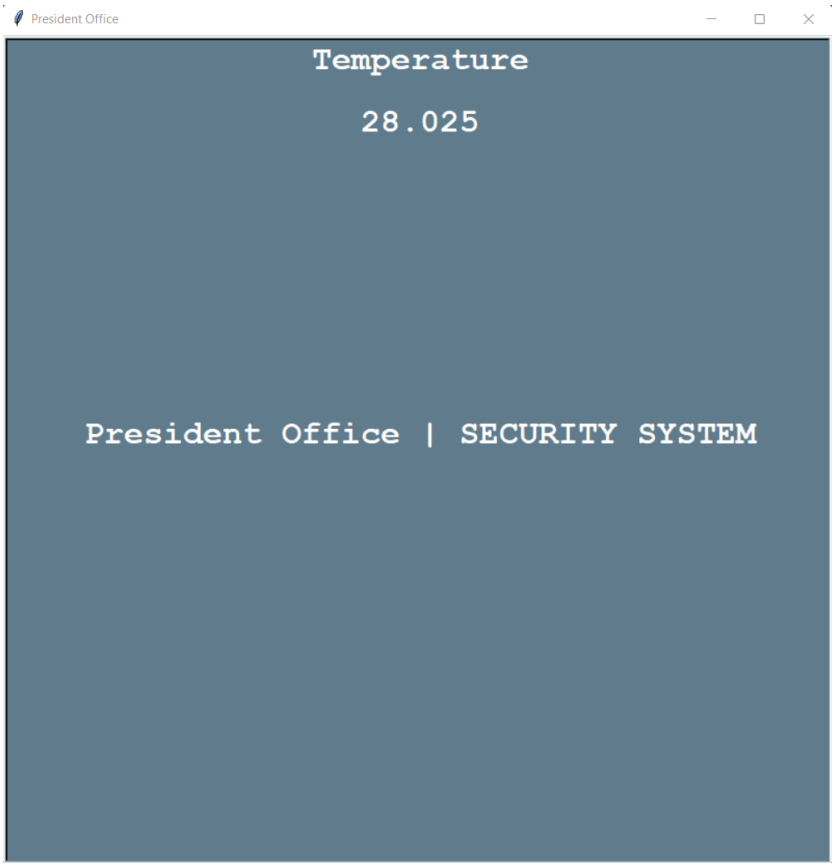
Special Functions

To proceed with the program some functions have been specially defined. In fact all the necessary functionalities are separately coded in the script. A special function for the fire alarm, the secret knock, the panic button and the alert off switch have been separately defined. Then the blocks of code are called in a main loop for the collective functioning of the system.

Display a text

A function is defined to produce text on the created display involving the other major functions of the system. The defined function uses the pen defined earlier along with coordinates to create text on the display. The text is aligned and the text is of a font that is preprogrammed

```
def display_text(pen,x,y,text,align):
    '''A function to display a text with given alignment'''

    pen.penup()
    pen.goto(x,y)
    pen.hideturtle()
    pen.write(text, align = align, font = ('Courier', 24, 'bold'))
```



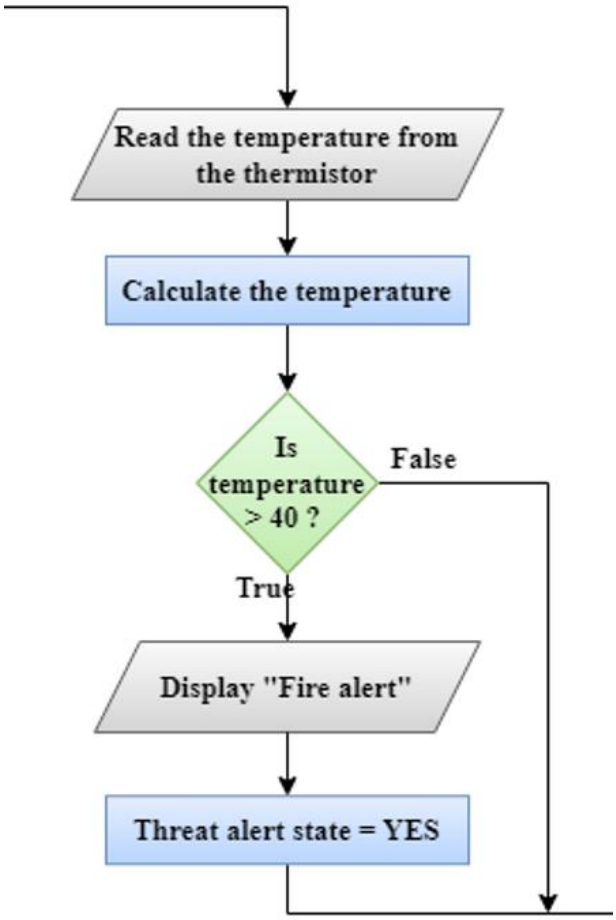
Fire alarm

The fire alarm is made using the thermistor which changes its resistance owing to changes in temperature. This resistance change makes it possible to obtain temperature readings. This is done by obtaining a voltage drop reading due to the change in the resistance. The wiring can be clearly seen where a single wire feeds a current into an analog pin through which the fluctuations are obtained. A function is defined to read these fluctuations and then convert the reading to temperature and return the value which then allows an alarm to be triggered via the piezo buzzer if the temperature is critically high.

The following equation was derived to calculate temperature value,

$$T = \frac{1}{\frac{1}{273} - \frac{1}{1948}\log\left(\frac{V}{0.34}\right)} - 273.15$$

T - the temperature in ˚C
V – The average voltage reading from the thermistor

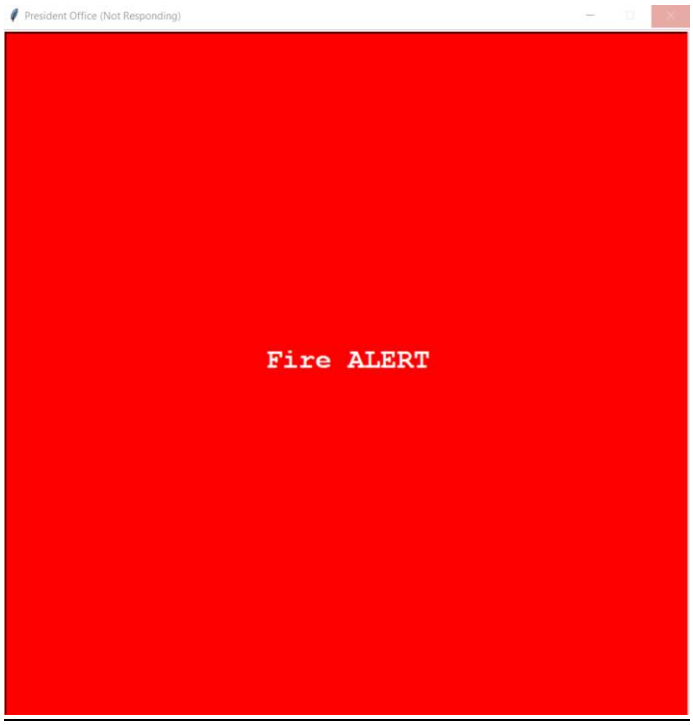


The temperature reading function is defined separately as follows.

```
def read_temperature(thermister):
    '''A function to read thermister value and return temperature'''
    global temp
    sum = 0
    for i in range(10):
        value = thermister.read()
        sum += value

    temp = round(1/((1/273)-(1/1948)*(log((sum/10)/0.34))) - 273.15, ndigits = 3)
    display_text(pen, 0, 360,'Temperature', 'center')
    display_text(pen, 0, 300,temp, 'center')
    time.sleep(1)
    return temp
```

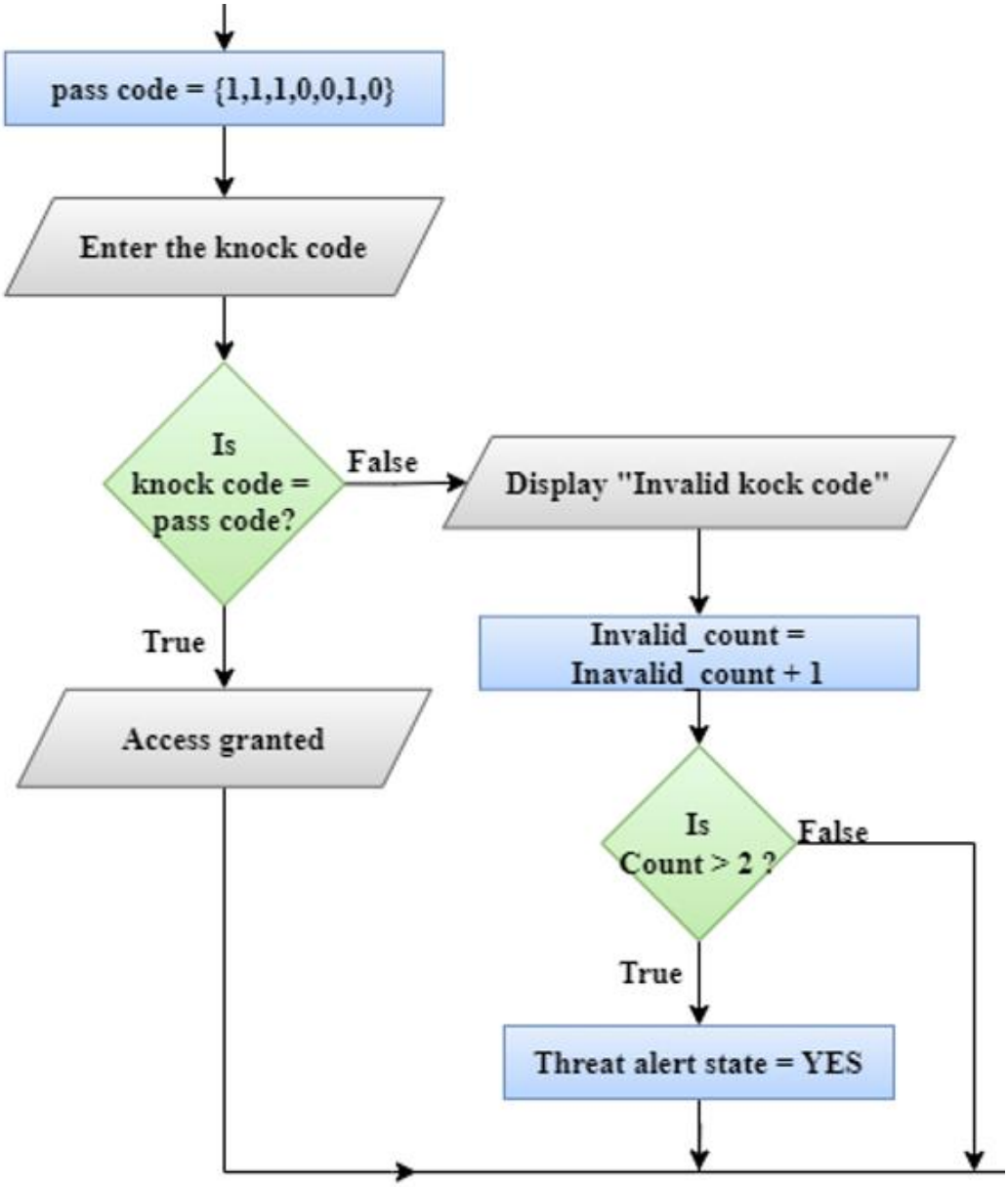
The above defined function only allows the temperature value to be returned. However, it does not trigger any alarms. To make the alarm ring another function comes into play. This function takes the reading given by the **read_temperature()** function and compares it with the critical value and if the temperature is higher than that it immediately triggers the fire alarm.

```
def TemperatureAlert():
    '''A function to check whether temperature is above critical temperature or not'''

    global Threat_alert_State
    temperature = read_temperature(thermister)
    if temperature > 40:
        pen.clear()
        display_text(pen, 0, 0,'Fire ALERT', 'center')
        wn.bgcolor('red')
        print('fire alert')
        Threat_alert_State = 'Yes'
```



Secret knock

It is no secret that accessibility to the president's office is granted to authorized personnel only. Therefore, whoever who knocks the code accurately is given access.

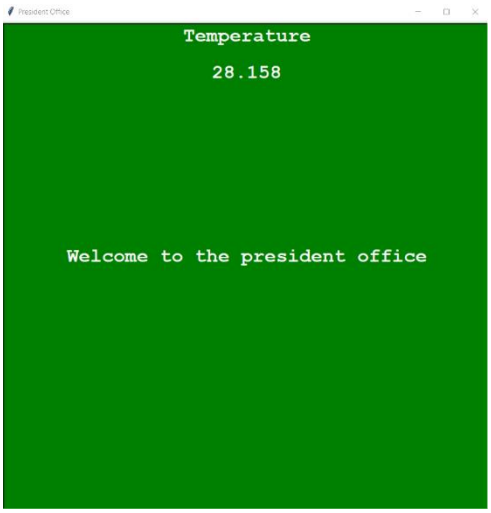


The knock code is entered using a series of push buttons. This particular secret knock is a combination of long presses of the push buttons according to a set pattern. To get access to knock the secret code, first the **push button 4** should be clicked. Then, two of the buttons (**push button 3 & 4**) are used for the knock code. An empty list is created to which, a unique number allocated for each button is appended when pressed such that it creates a list of the same two numbers repeating to a pattern and this pattern is supposed to be the preset knock code which is defined as **pass_code**. Meanwhile the previously created display guides the user through the steps of entering the knock code.

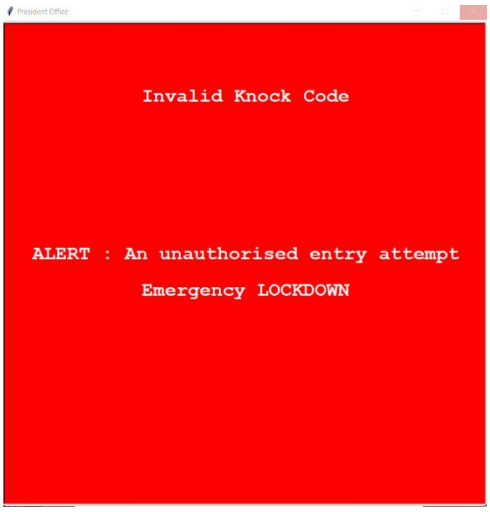
```
def secret_door():
    '''A fuction for allowing access to the President Office'''

    global invalid_count
    global knock_code
    global pass_code
    global Threat_alert_State

    #alocating a space for knock code
    knock_code =[]

    pass_code = [1,1,1,0,0,1,0]

    #gets knock code from user
    print('Enter Knock Knock')
    time.sleep(0.5)
    for i in range(1,8):
        time.sleep(0.5)

        #displays instructions
        pen.clear()
        display_text(pen, 0, 360,'Knock Your Pattern', 'center')
        wn.bgcolor('blue')
        display_text(pen, 0, 250,i, 'center')
        print('knok',i)

        time.sleep(1)
        x = knocking_button1.read()
        y = knocking_button2.read()
        if x == True:
            knock_code.append(1)
        elif y == True:
            knock_code.append(0)

        #displays the entered code
        display_text(pen, 0, 150,knock_code, 'center')
```

If the correct code is entered access is granted and if not displays it is an “Invalid Knock Code” , an alarm is triggered after exceeding two failed attempts and immediately preceding to activate the emergency lockdown.

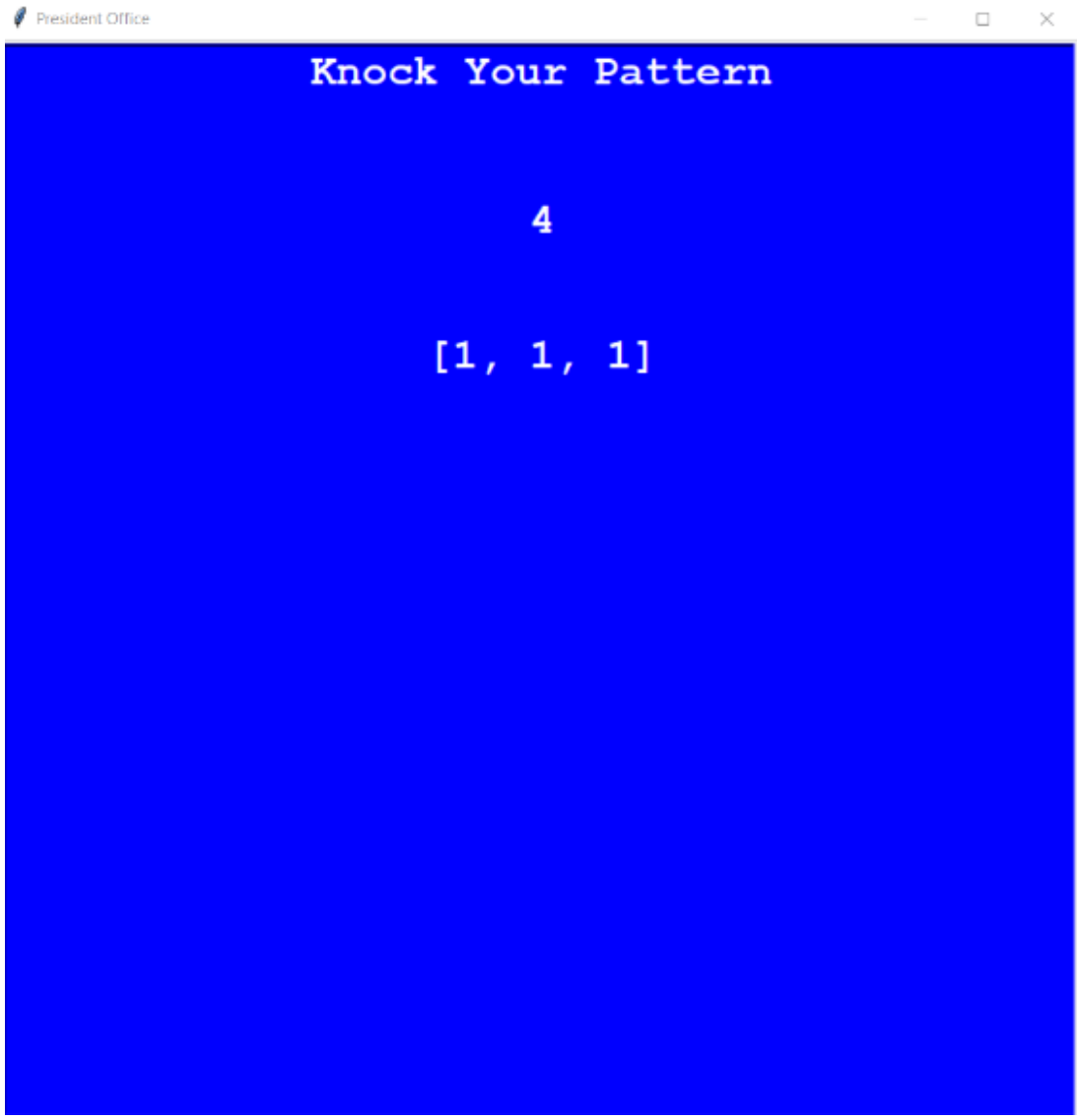
```
#when knock code is correct
if knock_code == pass_code:
    pen.clear()
    display_text(pen, 0, 0,'Welcome to the president office', 'center')
    wn.bgcolor('green')
    print('Access given to the president office')
    Giving_Access_LED.write(1)
    Threat_buzzer.write(1)
    time.sleep(0.2)
    Threat_buzzer.write(0)
    time.sleep(1)
    Giving_Access_LED.write(0)

#when knock code is invalid
else:
   pen.clear()
   display_text(pen, 0, 260,'Invalid Knock Code', 'center')    #displays code is invalid
   wn.bgcolor('red')
   print('invalid knock code')

   #gives alert
   Threat_alert_LED.write(1)
   Threat_buzzer.write(1)
   time.sleep(1)
   Threat_alert_LED.write(0)
   Threat_buzzer.write(0)

   invalid_count += 1
   print(invalid_count)
   if invalid_count > 2:
       #displays the alert
       display_text(pen, 0, 0,'ALERT : An unauthorised entry attempt', 'center')
       display_text(pen, 0, -60,'Emergency LOCKDOWN', 'center')
       print('Threat Alert')
       print('emergency lockdown')

       invalid_count = 2
       Threat_alert_State = 'Yes'
       Threat_alert()
```



When Access Granted



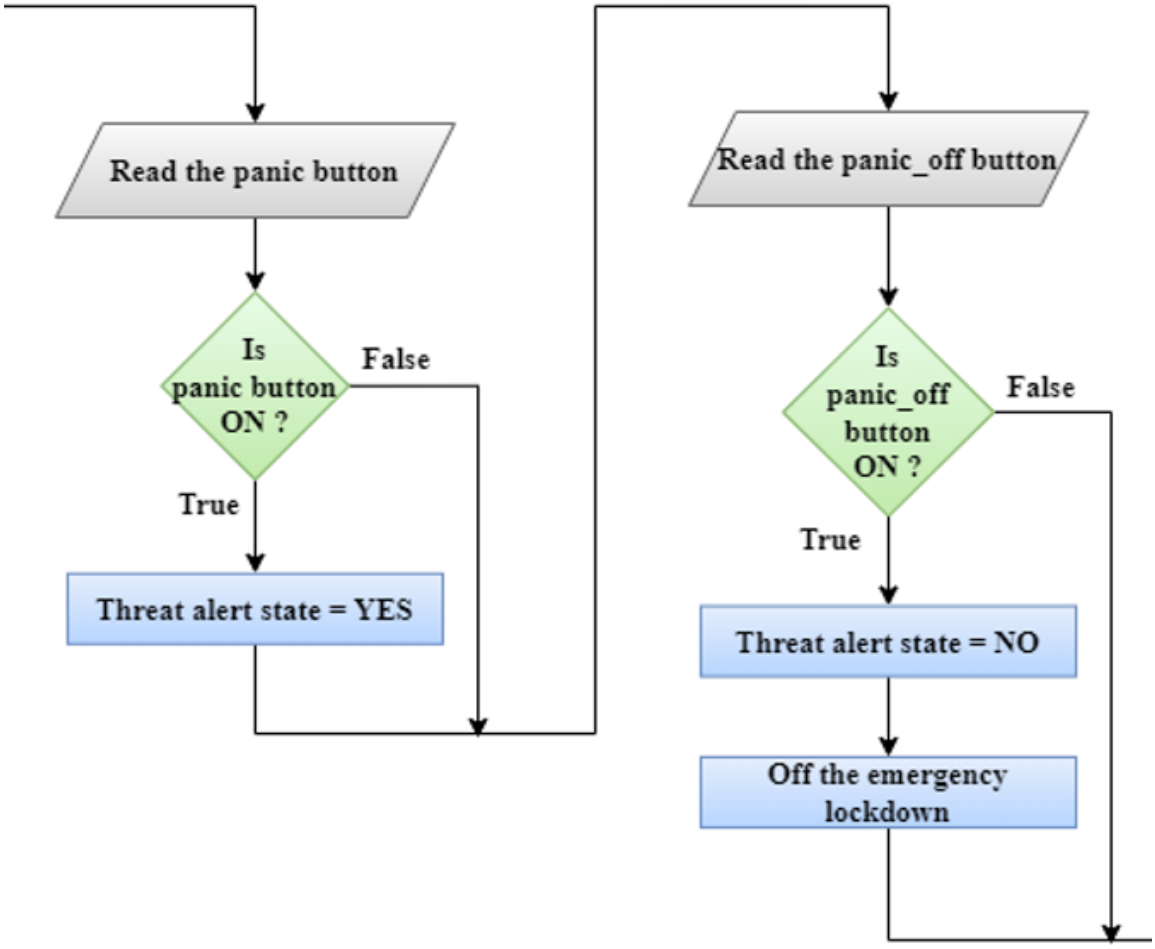
When exceeded two failed attempts



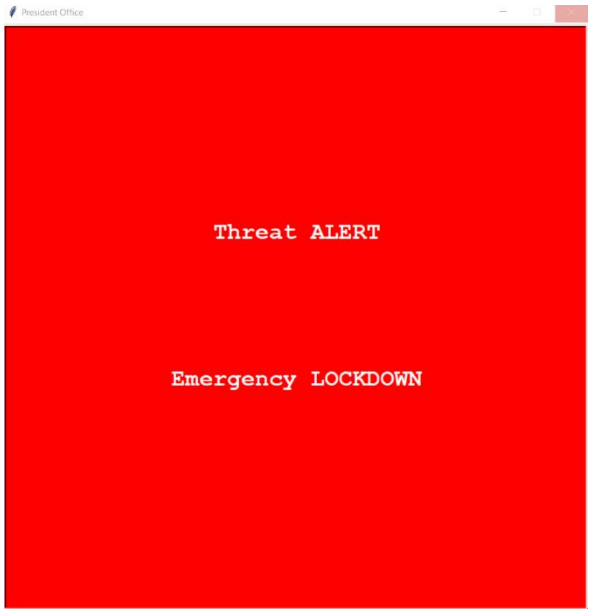
Display Guidance to User

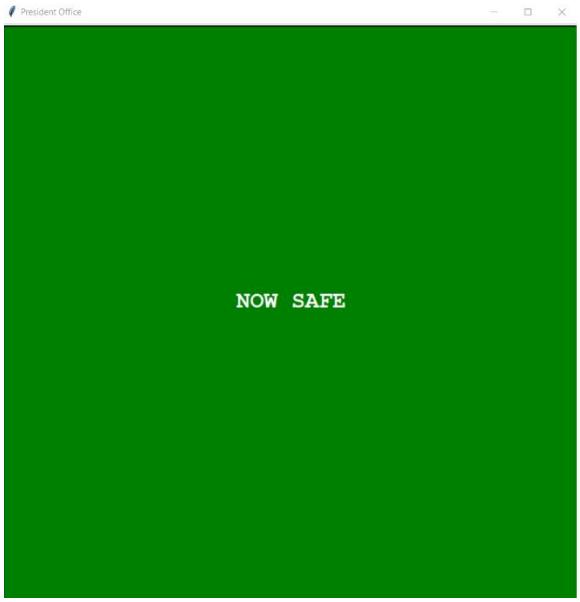
Panic button

The panic button is a button that is reserved for the president to use as a method of alerting the staff in case of an emergency. This button also makes use of the previously introduced threat alert state. Once the panic button is pressed the **Threat_alert_State** is made 'True' by making the panic state 'True' and similarly another button is used to revert the exact procedure by making the **Threat_alert_State** 'False' by making the **panic_off** state 'True'.



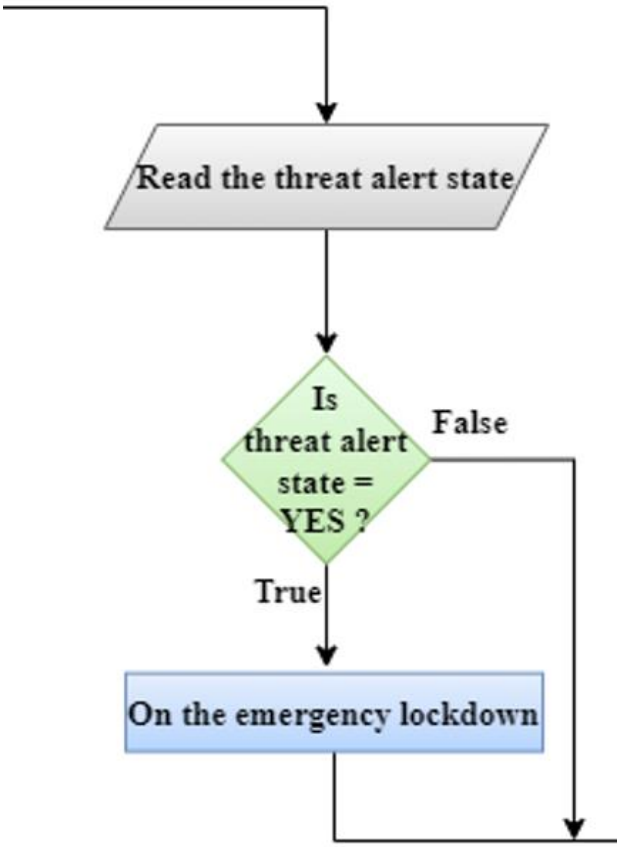
```python
def panic_button_fn():
    '''A function for panic button'''

    global panic_off
    panic = panic_button.read()
    if panic == True:
        global Threat_alert_State
        Threat_alert_State = 'Yes'


        #displays threat alert
        pen.clear()
        display_text(pen, 0, 100,'Threat ALERT', 'center')
        display_text(pen, 0, -100,'Emergency LOCKDOWN', 'center')
        wn.bgcolor('red')
        print('Threat Alert')
        print('emergency lockdown')

    else:
        #when wants to turn off alerts
        panic_off = panic_off_button.read()
        if panic_off == True:
            Threat_alert_State = 'No'
            Emergency_lockdown.write(0)
            pen.clear()
            display_text(pen, 0, 0,'NOW SAFE', 'center')
            wn.bgcolor('green')
            print('Threat Alert and emergency lockdown off')
            time.sleep(1)
```





Threat alert

This is a state which is made ‘true’ or ‘false’ by other functions to execute the block of code under the **Threat_alert()** function. When true this function triggers an alarm which rings until the **panic_off button** is pressed once the danger has ceased to exist



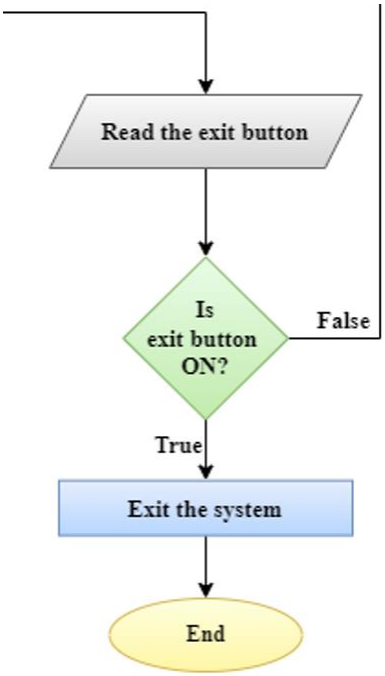

```
def Threat_alert():
    '''A function to difine how threat alert works'''

    while Threat_alert_State == 'Yes':
        Threat_alert_LED.write(1)
        Threat_buzzer.write(1)
        time.sleep(0.3)
        Threat_alert_LED.write(0)
        Threat_buzzer.write(0)
        time.sleep(0.3)
        panic_off = panic_off_button.read()

        if panic_off == True:
            panic_button_fn()
            break
```

The main loop

Once the defined functions are robust and running the main code can utilize them to take necessary readings and output the processed information as physical interactions with the user via the Arduino UNO.

```
#Continues_main_loop
while True:


    #introducing the program to user
    pen.clear()
    display_text(pen, 0, 0,'President Office | SECURITY SYSTEM', 'center')
    wn.bgcolor('LightSkyBlue4')

    S_d = knocking_button1.read()
    if S_d == True:
        secret_door()

    Threat_alert_State = 'No'

    TemperatureAlert()
    panic_button_fn()
    Threat_alert()


    # to exit from the system
    exit_p = knocking_button2.read()
    if exit_p == True:
        exit()
```

At the end of the main loop their introduced an exit button to exit from the system.



The complete code

```
#import libraries
from pyfirmata import Arduino, util, INPUT, OUTPUT
import time
from numpy import log
import turtle

#setup board
board = Arduino('COM7')
iterator = util.Iterator(board)
iterator.start()

#setup pins
thermister = board.analog[0]                #thermister pin
Fire_alert_LED = board.digital[5]           #red LED pin
Fire_alert_buzzer = board.digital[7]        #buzzer pin
Fire_alert_off = board.digital[6]           #push button 1
knocking_button1 = board.digital[12]        #push button 4
knocking_button2 = board.digital[8]         #push button 3
Threat_alert_LED = board.digital[5]         #red LED pin
Giving_Access_LED = board.digital[2]        #green LED pin
Threat_buzzer = board.digital[7]            #buzzer pin
Threat_alert_off = board.digital[6]         #push button 1
Emergency_lockdown = board.digital[3]
panic_button = board.digital[10]            #push button 2
panic_off_button = board.digital[6]         #push button 1

#set pin modes
thermister.mode = INPUT
Fire_alert_LED.mode = OUTPUT
Fire_alert_buzzer.mode = OUTPUT
Fire_alert_off.mode = INPUT
knocking_button1.mode = INPUT
knocking_button2.mode = INPUT
Threat_alert_LED.mode = OUTPUT
Threat_buzzer.mode = OUTPUT
Threat_alert_off.mode = INPUT
Emergency_lockdown.mode = OUTPUT
panic_button.mode = INPUT
panic_off_button.mode = INPUT
Giving_Access_LED.mode = OUTPUT


#setup display with turtle graphics
wn = turtle.Screen()
wn.title('President Office')
wn.bgcolor('LightSkyBlue4')
wn.setup(width = 800, height = 800)
wn.tracer(0)
```

```
50  #create pen for display with turtle graphics
51  pen = turtle.Turtle()
52  pen.speed(0)
53  pen.shape('square')
54  pen.color('white')
55
56  #set invalid count to zero
57  invalid_count = 0
58  #Set temperature value to zero
59  temp = 0
60
61  def display_text(pen,x,y,text,align):
62      '''A function to display a text with given alignment'''
63
64      pen.penup()
65      pen.goto(x,y)
66      pen.hideturtle()
67      pen.write(text, align = align, font = ('Courier', 24, 'bold'))
68
69
70
71  def read_temperature(thermister):
72      '''A function to read thermister value and return temperature'''
73      global temp
74      sum = 0
75      for i in range(10):
76          value = thermister.read()
77          sum += value
78
79      temp = round(1/((1/273)-(1/1948)*(log((sum/10)/0.34))) - 273.15, ndigits = 3)
80      display_text(pen, 0, 360,'Temperature', 'center')
81      display_text(pen, 0, 300,temp, 'center')
82      time.sleep(1)
83      return temp
```

```
86  def secret_door():
87      '''A fuction for allowing access to the President Office'''
88
89      global invalid_count
90      global knock_code
91      global pass_code
92      global Threat_alert_State
93
94      #alocating a space for knock code
95      knock_code =[]
96
97      pass_code = [1,1,1,0,0,1,0]
98
99      #gets knock code from user
100     print('Enter Knock Knock')
101     time.sleep(0.5)
102     for i in range(1,8):
103         time.sleep(0.5)
104
```

```
105         #displays instructions
106         pen.clear()
107         display_text(pen, 0, 360,'Knock Your Pattern', 'center')
108         wn.bgcolor('blue')
109         display_text(pen, 0, 250,i, 'center')
110         print('knok',i)
111
112         time.sleep(1)
113         x = knocking_button1.read()
114         y = knocking_button2.read()
115         if x == True:
116             knock_code.append(1)
117         elif y == True:
118             knock_code.append(0)
119
120         #displays the entered code
121         display_text(pen, 0, 150,knock_code, 'center')
122
123     #when knock code is correct
124     if knock_code == pass_code:
125         pen.clear()
126         display_text(pen, 0, 0,'Welcome to the president office', 'center')
127         wn.bgcolor('green')
128         print('Access given to the president office')
129         Giving_Access_LED.write(1)
130         Threat_buzzer.write(1)
131         time.sleep(0.2)
132         Threat_buzzer.write(0)
133         time.sleep(1)
134         Giving_Access_LED.write(0)
135
136     #when knock code is invalid
137     else:
138        pen.clear()
139        display_text(pen, 0, 260,'Invalid Knock Code', 'center')  #displays code is invalid
140        wn.bgcolor('red')
141        print('invalid knock code')
142
143        #gives alert
144        Threat_alert_LED.write(1)
145        Threat_buzzer.write(1)
146        time.sleep(1)
147        Threat_alert_LED.write(0)
148        Threat_buzzer.write(0)
```

```
149
150        invalid_count += 1
151        print(invalid_count)
152        if invalid_count > 2:
153            #displays the alert
154            display_text(pen, 0, 0,'ALERT : An unauthorised entry attempt', 'center')
155            display_text(pen, 0, -60,'Emergency LOCKDOWN', 'center')
156            print('Threat Alert')
157            print('emergency lockdown')
158
159            invalid_count = 2
160            Threat_alert_State = 'Yes'
161            Threat_alert()
162
```

```
163
164 def panic_button_fn():
165     '''A function for panic button'''
166
167     global panic_off
168     panic = panic_button.read()
169     if panic == True:
170         global Threat_alert_State
171         Threat_alert_State = 'Yes'
172
173
174         #displays threat alert
175         pen.clear()
176         display_text(pen, 0, 100,'Threat ALERT', 'center')
177         display_text(pen, 0, -100,'Emergency LOCKDOWN', 'center')
178         wn.bgcolor('red')
179         print('Threat Alert')
180         print('emergency lockdown')
181
182     else:
183         #when wants to turn off alerts
184         panic_off = panic_off_button.read()
185         if panic_off == True:
186             Threat_alert_State = 'No'
187             pen.clear()
188             display_text(pen, 0, 0,'NOW SAFE', 'center')
189             wn.bgcolor('green')
190             print('Threat Alert and emergency lockdown off')
191             time.sleep(1)
192
193
194 def Threat_alert():
195     '''A function to difine how threat alert works'''
196
197     while Threat_alert_State == 'Yes':
198         Threat_alert_LED.write(1)
199         Threat_buzzer.write(1)
200         time.sleep(0.3)
201         Threat_alert_LED.write(0)
202         Threat_buzzer.write(0)
203         time.sleep(0.3)
204         panic_off = panic_off_button.read()
205
206         if panic_off == True:
207             panic_button_fn()
208             break
209
```

```
209
210
211 def TemperatureAlert():
212     '''A function to check whether temperature is above critical temperature or not'''
213
214     global Threat_alert_State
215     temperature = read_temperature(thermister)
216     if temperature > 40:
217         pen.clear()
218         display_text(pen, 0, 0,'Fire ALERT', 'center')
219         wn.bgcolor('red')
220         print('fire alert')
221         Threat_alert_State = 'Yes'
222
223
224 #Continues_main_loop
225 while True:
226
227
228     #introducing the program to user
229     pen.clear()
230     display_text(pen, 0, 0,'President Office | SECURITY SYSTEM', 'center')
231     wn.bgcolor('LightSkyBlue4')
232
233     S_d = knocking_button1.read()
234     if S_d == True:
235         secret_door()
236
237     Threat_alert_State = 'No'
238
239     TemperatureAlert()
240     panic_button_fn()
241     Threat_alert()
242
243
244     # to exit from the system
245     exit_p = knocking_button2.read()
246     if exit_p == True:
247         exit()
248
249
```