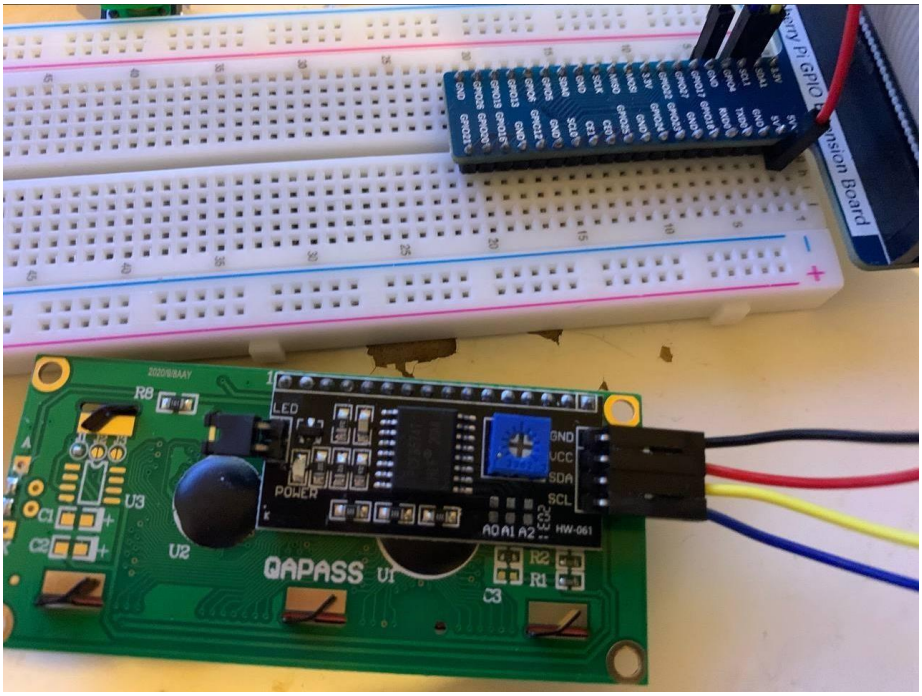
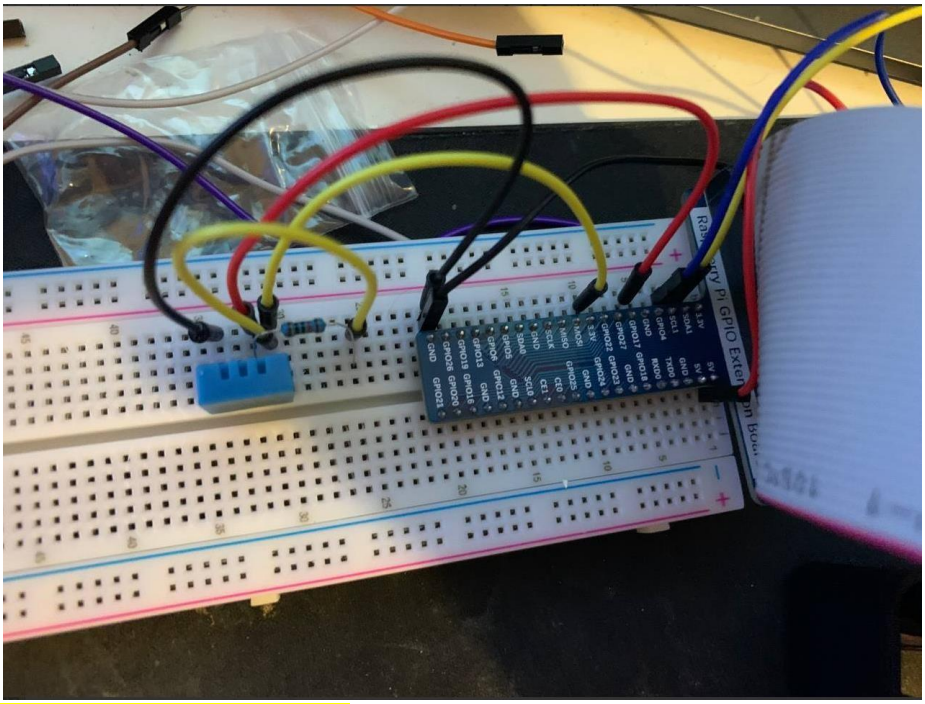
REPORT

1, Installation

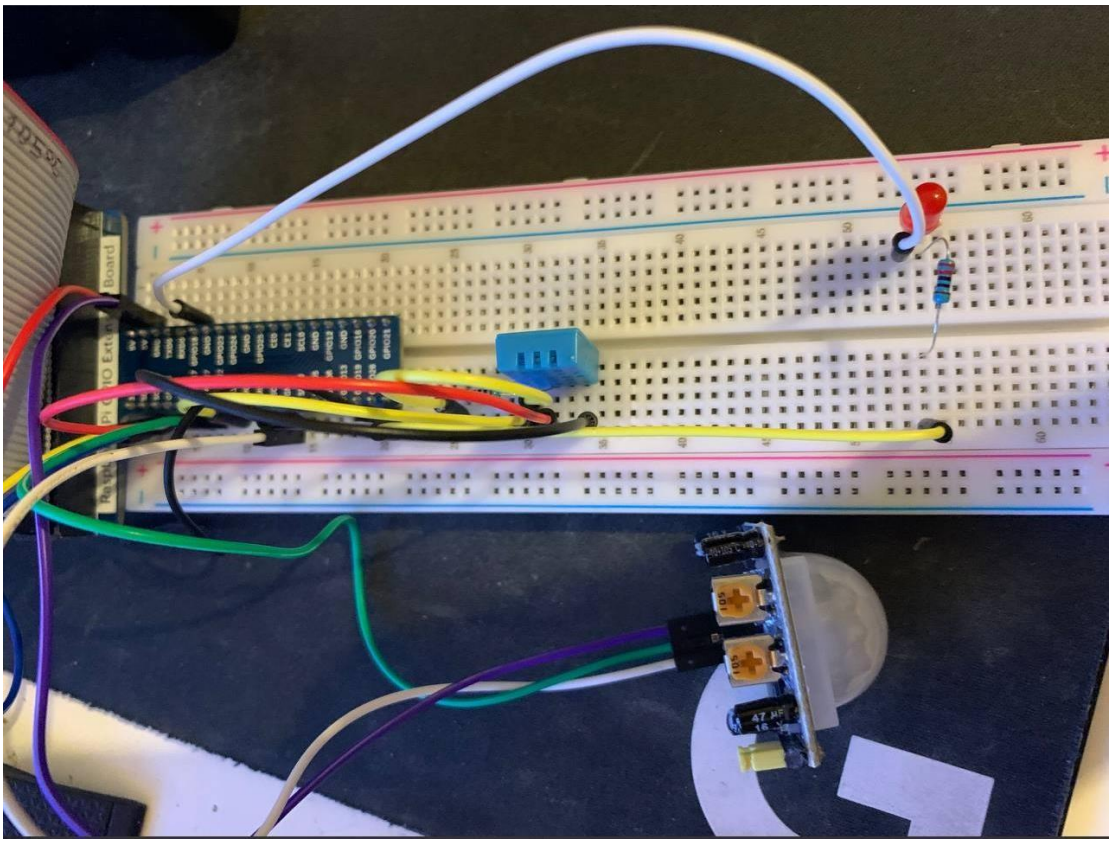
1, Connect the lcd



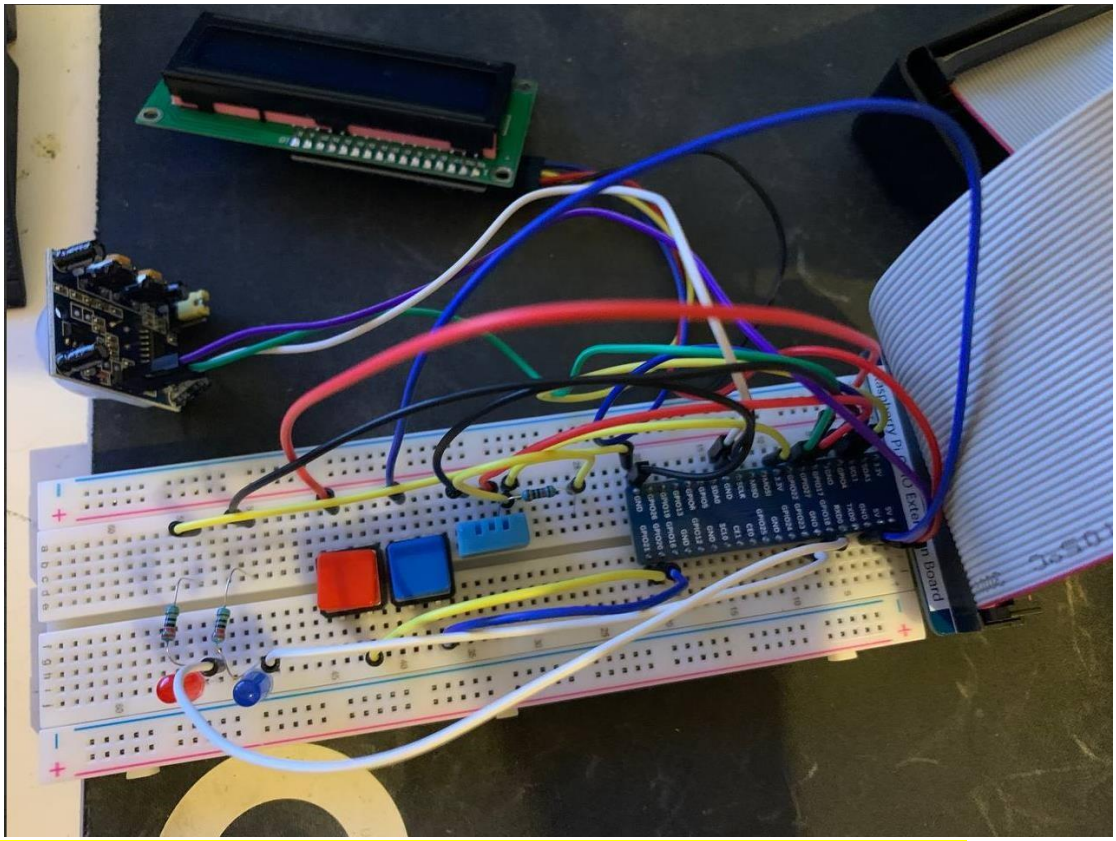
2, Connect the temp sensor DHT11



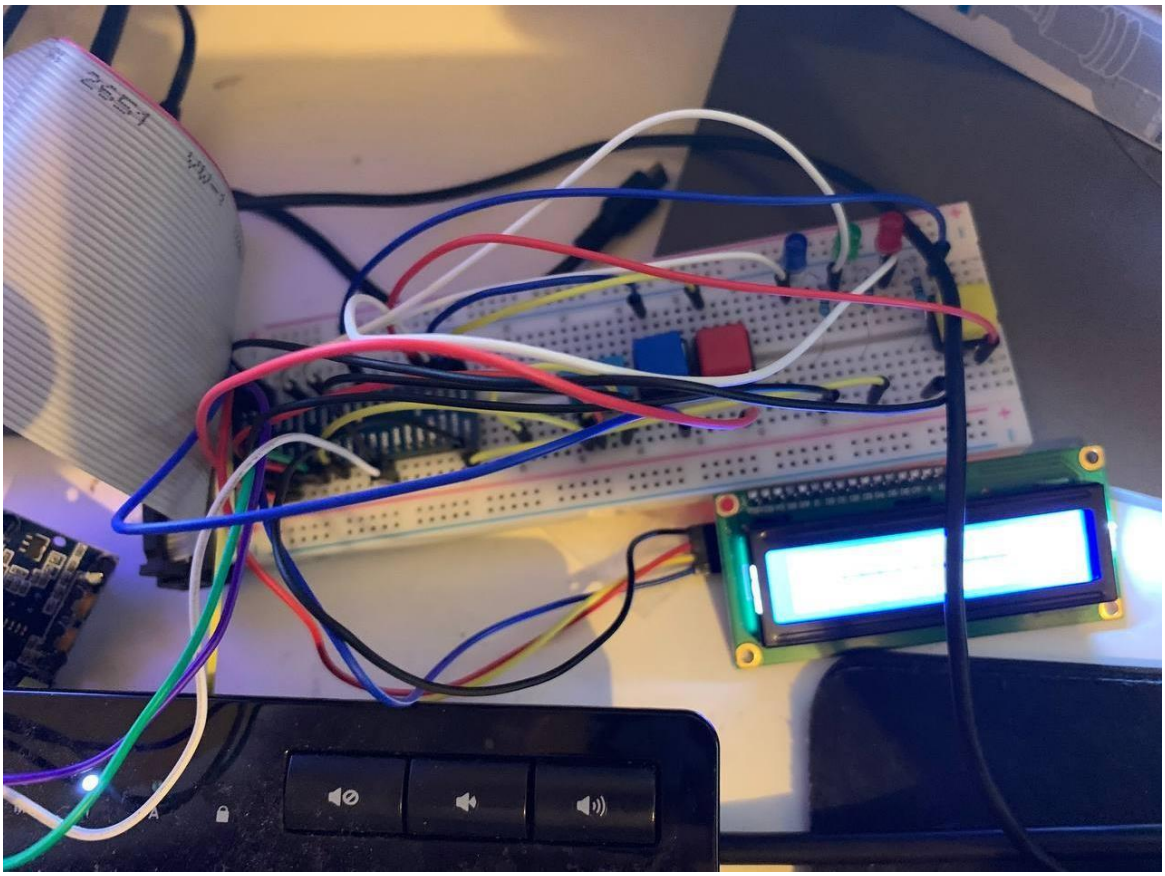
3, Connect the motion sensor



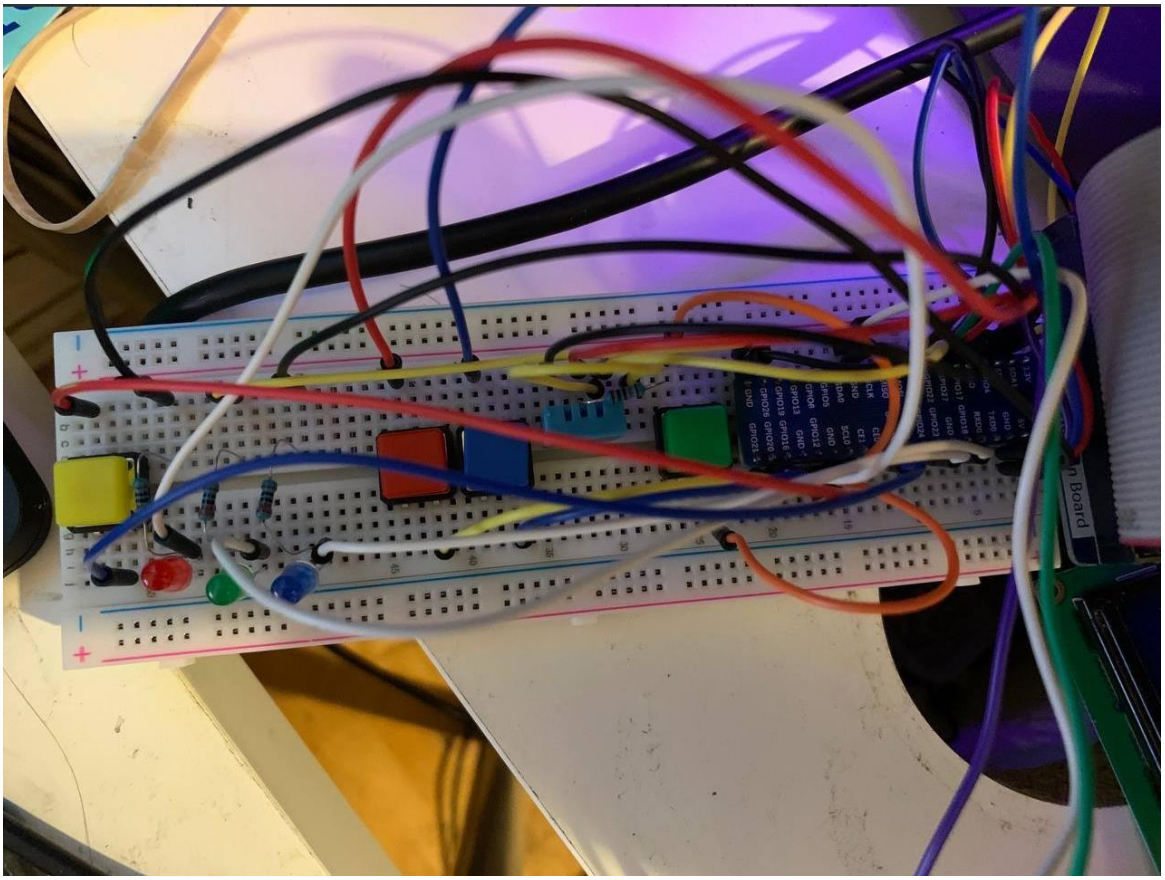
4, Connect the buttons and lights (Used two 220 ohm resistors)



5, Connect the push button for the door and the green led for the motion sensor

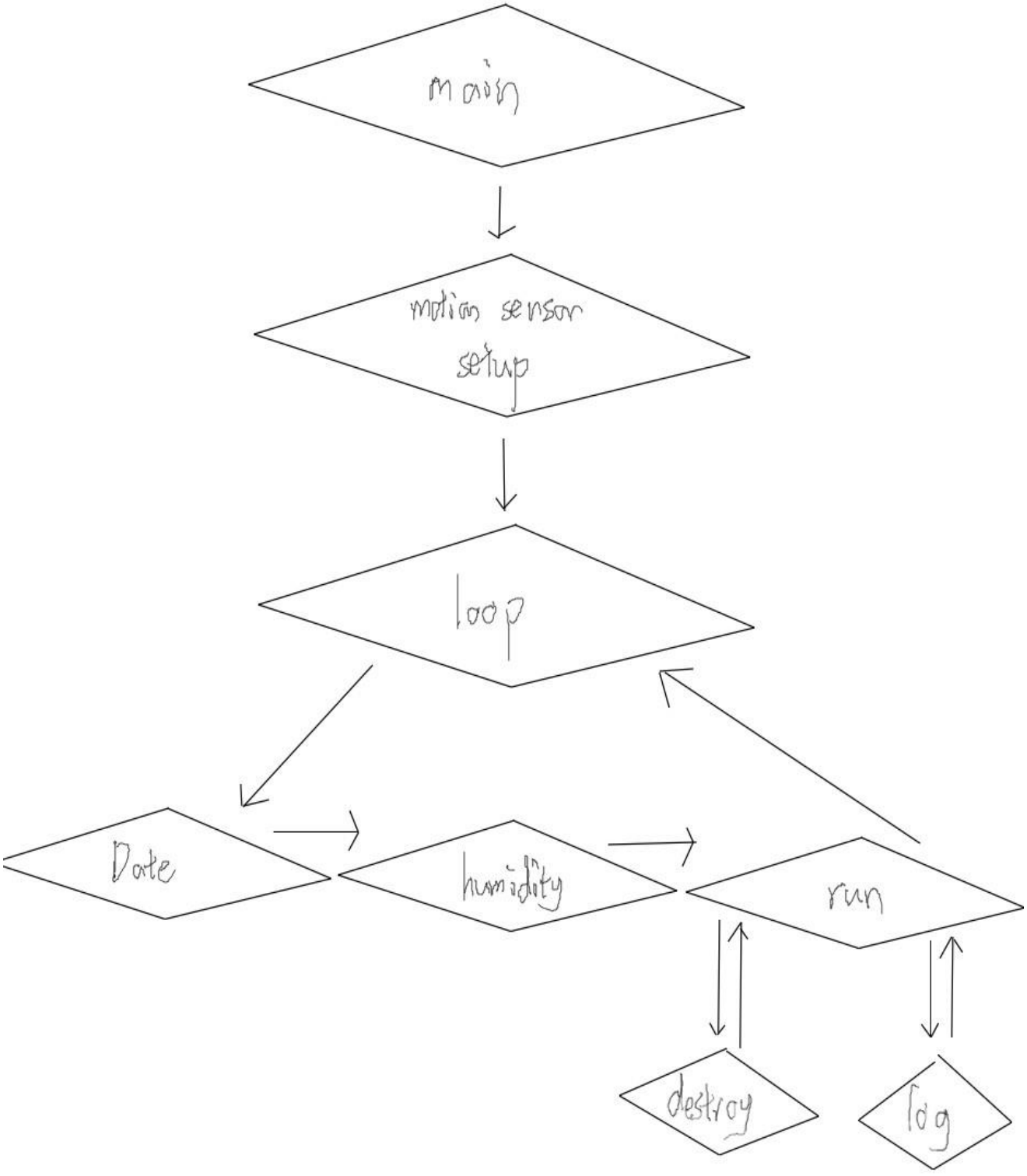


6, add a new button (green button) for displaying date and time(additional feature)



2, description

Basic structure of my code:



ALL THE CODE IS COMMENTED (did not comment some repeated part though)

0, Before running any functions

The program imports all the libraries and initialize all the ports for leds as well as buttons. Also, their initial value(state), and behavior are also initialized.

```
import RPi.GPIO as GPIO
from PCF8574 import PCF8574_GPIO
from Adafruit_LCD1602 import Adafruit_CharLCD
import time
import Freenove_DHT as DHT
from time import sleep
from datetime import datetime
import urllib.request
import json


ledPin = 12      # define ledPin
sensorPin = 13    # define sensorPin
DHTPin = 11     #define the pin of DHT11
GPIO.setwarnings(False)        #set the warming to be false at the beginning
GPIO.setmode(GPIO.BOARD)       #set the mode to BOARD

PCF8574_address = 0x27  # I2C address of the PCF8574 chip.
mcp = PCF8574_GPIO(PCF8574_address)
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)

#setup the ports. When the button is preesed the value is 1
GPIO.setup(38, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(40, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(32, GPIO.IN, pull_up_down=GPIO.PUD_UP)
#initialize the port(led). at the beginning led is off
GPIO.setup(16, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(36, GPIO.OUT, initial=GPIO.LOW)
#add event detect to the according ports that are installed with buttons, RISING state will be true
GPIO.add_event_detect(38, GPIO.RISING)
GPIO.add_event_detect(40, GPIO.RISING)
GPIO.add_event_detect(22, GPIO.RISING)
GPIO.add_event_detect(32, GPIO.RISING)

DW = 75              #set the default desired weather index to be 75
b = 0                #set b to be 0 for later use
```

1, Main:

```
if __name__ == '__main__':
    motionSetup()                                   #initialize the motioncensor
    try:
        loop()                                      #call the loop funciton
    except KeyboardInterrupt:                       #if we ctrl + c the program will end
        GPIO.cleanup()
        destroy()
        exit()
```

The main function is the top layer of my program, it will firstly setup the motion sensor, and then call the loop function. If there is any keyboard interruption, the program will end.

2, MotionSetup:

The motionSetup() is the function that initialize the motion sensor.

```
def motionSetup():
    GPIO.setmode(GPIO.BOARD)        # use PHYSICAL GPIO Numbering
    GPIO.setup(ledPin, GPIO.OUT)    # set ledPin to OUTPUT mode
    GPIO.setup(sensorPin, GPIO.IN)  # set sensorPin to INPUT mode
```

3, Loop

The next layer aims to read the first two temperature. It also initializes variables like the array to store the temperature, also including the variables that are used to check change of states (Because we need to show the change of state to the lcd).

```
def loop():
    dht = DHT.DHT(DHTPin)   #create a DHT class object
    mcp.output(3,1)     # turn on LCD backlight
    lcd.begin(16,2)     # set number of LCD lines and columns

    H = ['0','0','0']         #create a array to store the temp, initial temps are 0.
    chk = dht.readDHT11()     #read DHT11 and get a return value. Then determine whether data read is normal according to the return value.
    H[0] = dht.temperature    #get the temp and store it to H[0]
    sleep(1)                  #delay for 1s
    chk = dht.readDHT11()     #read DHT11 and get a return value. Then determine whether data read is normal according to the return value.
    H[1] = dht.temperature    #get the temp and store it to H[1]
    sleep(1)                  #delay for 1s
    global TS                 #timer TS
    global TE                 #timer TE
    global halt               #"halt" used for the discrepancy for the state of the door
    global times              #'timer' used for indicating the numbers that the door(bottun) is pushed
    global T                  #HAVC'state won't show up at the first time
    global humid              #for humidity
    global TIMER1             #timer for date, becasue we need the humidity data 1 time per hour
    global TIMER2             #timer for date, becasue we need the humidity data 1 time per hour

    #initialize all the value
    TE = 0
    TS = 11
    halt = 0
    times = 0
    T = 1
    TIMER1 = time.time()      #record the time outside of the while loop
    while 1:
        TIMER2 = time.time()  #record the time inside the while loop
        currentT()            #the function records the date
        Humidity()            #the function takes the humidity value from the web
        run(dht, H)           #run the main program
```

4, CurrentT

The function aims to get the date, because we need the date to get the humidity from the web. There are 2 timers here, timer 2 is outside of this function (in loop), the purpose of those timers is to make sure we only check for date update every 21600 seconds (one day) for once.

```
def currentT():
    global TIMER1
    global TIMER2
    global Date
    if((TIMER2 - TIMER1) % 216000 < 10):                    #record the date once per day
        now = str(datetime.now())                           #get the time
        Date = now[0:10]                                    #get the date
        #print(Date)
```

5, Humidity

As what its name means, this function gets the humidity from the web by using Json, a lightweight data-interchange format. There are also timers in this function because we only need to check the humidity once per hour. Also, I used a loop to make sure we get the latest valid humidity because sometimes the humidity of the most recent hour is missing.

```
def Humidity():
    global humidZ
    global TIMER1
    global TIMER2
    global Url
    global Date
    URL='http://et.water.ca.gov/api/data?appKey=c391cd61-6c13-46a9-a7f2-2bc53431e042&targets=75&startDate='+ Date + '&endDate=' + Date + '&dataItems=hly-rel-hum'
    if((TIMER2 - TIMER1) % 3600 < 10):                      #record the time one time person hour
        #create Json
        Url = urllib.request.urlopen(URL)
        data = Url.read()
        Encoding = Url.info().get_content_charset('utf-8')
        Json = json.loads(data.decode(Encoding))
        #print(Json)
        for i in Json['Data']['Providers'][0]['Records']:   #loop the directionary
            if(i['HlyRelHum']['Qc'] != 'M'):
                humid = int(i['HlyRelHum']['Value'])
                #print("%d",humid)
                #print(json)
```

6, Run

The core function of this project. In this function, all buttons' detection, sensor detection, lcd display, and led start working. At first, I am stuck at the part that we need to show the change of state on the lcd. For instance, if now the AC is on, but previously it was off, I need to show the change "Ac is on" on the lcd. Finally, I solved this by adding some variables to check the discrepancy of the state.

```
def run(dht, H):
    global Weather
    global DW
    global halt
    global times
    global TE
    global a
    global T
    global humid
    global event
    global once
    global b

    lcd.setCursor(0,0)                                      #set the starting position of the lcd
    #when the green button is pressed, time with no date will show up, refreshing one time per sec
    if(GPIO.event_detected(32)):
        destroy()                                           #clean up the lcd
        current = datetime.now()                            #get the time
        lcd.message(current.strftime("%Y-%m-%d \n       %H:%M:%S"))# display
        sleep(1)                                            #stall for 1s
        current = datetime.now()                            #get the time
        lcd.message(current.strftime("%Y-%m-%d \n       %H:%M:%S"))# display
        sleep(1)                                            #stall for 1s
        current = datetime.now()                            #get the time
        lcd.message(current.strftime("%Y-%m-%d \n       %H:%M:%S"))# display
        sleep(1)                                            #stall for 1s
        destroy()                                           #clean up the lcd

    #when the button for the door and the window is pressed
    if(GPIO.event_detected(22)):
        if(times == 0):...
        elif(times == 1):
            times = 0
            destroy()                                       #set the next condition
            lcd.message('   DOOR/WINDOW\nCLOSED HVAC OPEN')  #display
            halt = 0                                        #save the current state of the HVAC
            sleep(3)                                        #stall for 3s
            destroy()                                       #clean up the lcd
            event = 'DOOR SAFE'                             #event is 'door safe'
            log()                                           #write to the log.txt

    chk = dht.readDHT11()                                   #read DHT11 and get a return value.
    if(0<=dht.temperature<=50):                             #determine temp read is normal according to the return value.
        H[2] = dht.temperature                              #if it's a vaild value it will be stored to H[2]
        sleep(1)                                            #stall for 1s

    AH = (H[0]+H[1]+H[2])/3                                 #calculate the temp, average the last 3 values
    Weather = AH * 1.8 + 32 + 0.05 * humid                  #calculate the Weather

    if(0<=dht.temperature<=50):                             #if the temp read is normal update the old value
        H[0] = H[1]
        H[1] = H[2]

    if(GPIO.event_detected(38)):                            #if blue button is pressed, DW will be deducted by 2
        if(65 < DW <= 85):                                  #set the range
            DW -= 2
```

```
if(GPIO.event_detected(40)):                    #if blue button is pressed, DW will be increased by 2
    if(65 <= DW < 85):                          #set the range
        DW += 2

if((GPIO.input(sensorPin)==GPIO.HIGH) and b != 1):      #if b != 1, it indicates that the state of the light has changed, so we need to write it into log file
    event = 'LIGHTS ON'                         #event = 'LIGHT ON'
    log()                                       #write it to the log file

lcd.setCursor(11,1)                             #set cursor position
if(GPIO.input(sensorPin)==GPIO.HIGH):           #if the motion sensor detects the motion
    TE = time.time()                            #record time here
    GPIO.output(ledPin,GPIO.HIGH)               #turn on led
    lcd.message(' L:ON')                        #show on the lcd
    b = 1                                       #record the current state

if(time.time() - TE > 10):                      #if the difference between 2 timer is greater than 10, we need to turn the light off
    GPIO.output(ledPin,GPIO.LOW)                #turn off led
    lcd.message('L:OFF')                        #show on the lcd
    b = 2                                       #record the current state


if(T != 1):                                     #We don't need the state show in the first time
    lcd.setCursor(0,0)                          #set the starting position of the lcd
    #check if the state is different, if the state is different, we need to show in the lcd
    if(Weather - DW > 3 and a != 0 and halt != 1):       #check if Weather - DW >3 and it's not at this state before
        destroy()                               #clean up the lcd
        lcd.message('AC is on')                 #show on the lcd
        event = "HVAC AC"                       #event = 'HVAC_AC'
        log()                                   #write to the log.txt
        sleep(3)                                #stall for 3s
        destroy()                               #clean up th the whole screem
    if(DW - Weather > 3 and a != 1 and halt != 1):       #check if DQ - Weather >3 and it's not at this state before
        destroy()                               #clean up the lcd
        lcd.message('HEAT is on')               #show on the lcd
        event = "HVAC HEAT"                     #event = 'HVAC_AC'
        log()                                   #write to the log.txt
        sleep(3)                                #stall for 3s
        destroy()                               #clean up th the whole screem
    if((DW - Weather)**2 <9 and a != 2):        #check if (DW - Weather)**2 <9 and it's not at this state before
        destroy()                               #clean up the lcd
        lcd.message('Both AC and HEAT\n are off')       #show on the lcd
        event = "HVAC OFF"                      #event = 'HVAC_OFF'
        log()                                   #write to the log.txt
        sleep(3)                                #stall for 3s
        destroy()                               #clean up th the whole screem
    if(halt == 1 and a != 2):                   #check if halt == 1 and it's not at this state before
        destroy()                               #clean up the lcd
        lcd.message('Both AC and HEAT\n are off')       #show on the lcd
        event = "HVAC OFF"                      #event = 'HVAC_OFF'
        log()                                   #write to the log.txt
        sleep(3)                                #stall for 3s
        destroy()                               #clean up th the whole screem
```

```
if(Weather - DW > 3 and halt == 0):             #make sure that Weather - DW > 3 and the door is safe
    GPIO.output(16, GPIO.HIGH)                  #the blue led is on if ac is on
    lcd.message('\nH:AC')                       #show on the lcd
    a = 0                                       #record the current state

if(DW - Weather > 3 and halt == 0):             #make sure that DW - Weather > 3 and the door is safe
    GPIO.output(36, GPIO.HIGH)                  #the red led is on if ac is on
    lcd.message('\nH:HEAT  ')                   #show on the lcd
    a = 1                                       #record the current state

if(halt == 1 or (DW - Weather)**2< 9):          #if (DW - Weather)**2< 9 or the door is open, HAVC is off
    GPIO.output(16, GPIO.LOW)                   #turn off the blue led
    GPIO.output(36, GPIO.LOW)                   #turn off the red led
    lcd.message('\nH:OFF   ')                   #show on the lcd
    a = 2                                       #record the current state

T = 0
lcd.setCursor(0,0)                              #set starting position of lcd
lcd.message('' + '{:.0f}'.format(float(Weather)) +'/' + '{:.0f}'.format(float(DW)))#display WEATHER


lcd.setCursor(10,0)                             #set starting position of lc
if(times %2 == 0):
    lcd.message('D:SAFE')                       #display DOOR
else:
    lcd.message('D:OPEN')                       #display DOOR
```

7, Destroy

This function cleans up the lcd.

```
def destroy():                                  #the destroy function aims to clear up the lcd
    lcd.clear()
```

8, Log

This function writes the “event” to the log.txt file. Event is global variable that contains the string of event from every part of the “Run” function. It firstly gets the current time with no

```
def log():
    global event                                #event contains the info that is gonna be stored in the log.txt file
    now = str(datetime.now())                   #get the time
    noDate = now[11:19]                         #get the time with no date
    with open('log.txt','a') as f:              #open the file to write, mode append
        f.write(noDate)                         #write the date first
        f.write(' ')                            #then write a ' '
        f.write(event)                          #write the event
        f.write('\n')                           #next line
```

date, and then writes then into the text file.