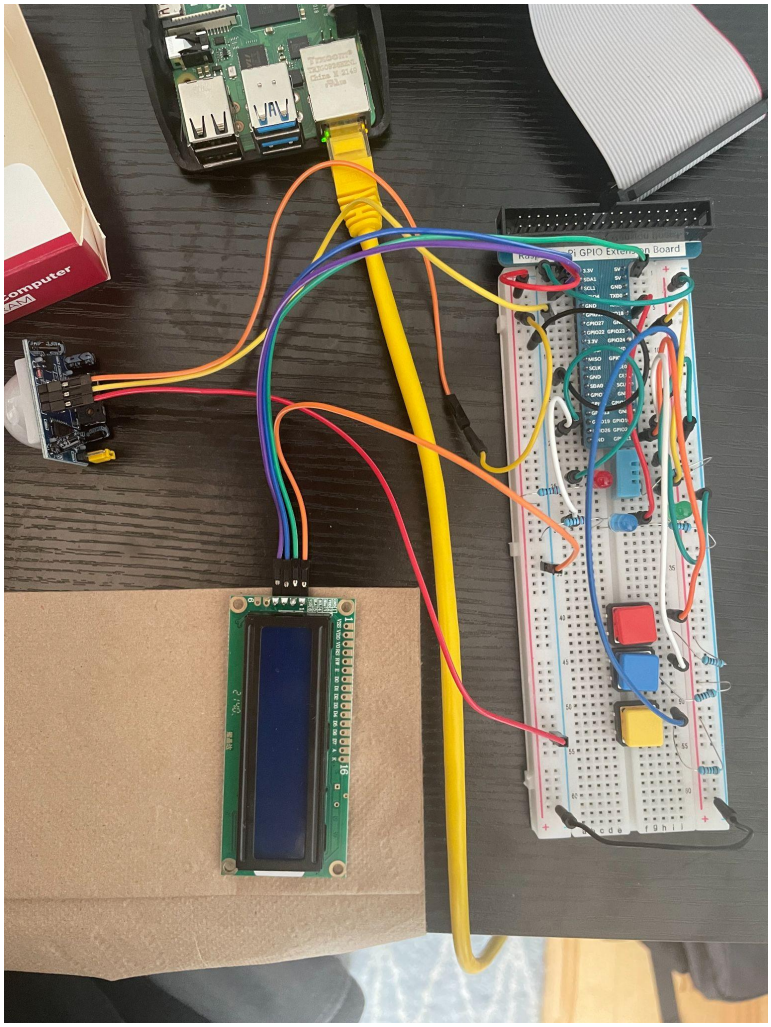
Ayman Taleb
ID: 60011014
EECS 113
June 11th, 2022

Final Project: HVAC System



Imported Libraries:

```
#importing libraries for GPIO, json, time, LCD
import RPi.GPIO as GPIO
import sys
import Adafruit_DHT as DHT
import time
import requests
import json
from datetime import datetime
from pytz import timezone
import pytz
from PCF8574 import PCF8574_GPIO
from Adafruit_LCD1602 import Adafruit_CharLCD
```

Adafruit_DHT is for the DHT 11 for temperature reading
RPI.GPIO is to use the GPIO on the Pi
Json, requests, datetime, pytz are used for the CIMIS humidity retrieval
PCF8574 and Adafruit_LCD are used for the LCD

Ambient light control:
The ambient light control consists of an infrared sensor and an led, when the sensor detects motion it will continue to output a signal powering the LED until there is no more motion. While the LED is powered, its status will be displayed on the LCD. The LED stays powered for 10 seconds, which is counted down and printed in the terminal.

When the sensor detects motion, it interrupts the program with the add_event_detect and calls the SIR function. This function will enable the green LED and count down 10 seconds. If the sensor keeps detecting motion it will keep calling the function.

```
GPIO.add_event_detect(SIRPin, GPIO.RISING, callback = SIR, bouncetime = 500)#interrput for motion sensor
```

```
#when infrared sensor detecs motion this is called, turing on the led
def SIR(channel):
    try:
        timer = 10
        while timer > 0:
            GPIO.output(SensorledPin, GPIO.HIGH)
            print("led on\n", timer)
            time.sleep(1)
            timer = timer -1
        GPIO.output(SensorledPin, GPIO.LOW)

    except KeyboardInterrupt:
        GPIO.cleanup()
```

Initializing the LCD, this code is from the Freenove code examples. It uses the I2C address of the LCD.

```
#initialize LCD
PCF8574_address = 0x27  # I2C address of the PCF8574 chip.
PCF8574A_address = 0x3F  # I2C address of the PCF8574A chip.
# Create PCF8574 GPIO adapter.
try:
    mcp = PCF8574_GPIO(PCF8574_address)
except:
    try:
        mcp = PCF8574_GPIO(PCF8574A_address)
    except:
        print ('I2C Address Error !')
        exit(1)
# Create LCD, passing in MCP GPIO adapter.
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)
```

HVAC System:

DHT Temperature reading:

By using the DHT11 I use this function to read the temperature and humidity of the environment. It takes three readings and averages them. The humidity detected by the DHT 11 is not used however, instead, the function getHumidity is used to get hourly readings from the CIMIS station in Irvine. Then the feels like value is calculated and stored in the weather variable.

```
#reading the data from the DHT 11, getting temp and humidity but only the CIMIS humidity is used for the weather
def DHT_read():
    global ambTemp
    humiDHT, temp = DHT.read(DHT11, DHTPin) #reading temp and humidity
    if humiDHT is not None and temp is not None:
        tempSum = 0
        humidSum = 0
        for i in range(3): #averaging last three temp readings
            tempSum = ((temp * 1.8) + 32) + tempSum
        realHumid = getHumidity() #getting humidity
#         realHumid = 80
        realHumid = float(realHumid)
        temp = tempSum/3
        ambTemp = temp
        global weather
        weather = temp + .05*realHumid #getting the feel like weather
        weather = round(weather)
        print("temp = {0:0.1f}F humid = {1:0.1f}%".format(temp,realHumid))
    else:
        print("DHT error")
```

CIMIS data retrieval:

By signing up to the CIMIS website you can get an API key and use json requests to pull humidity data. This function can sometimes not work because the API requests take too long or the service doesn't have data for that hour.

```
#using the CIMIS API we can use json to pull a data request to get the hourly relative humitity in Irvine
def getHumidity():
    timeNow = datetime.now(tz=pytz.utc) #setting day to current day
    timeNow = timeNow.astimezone(timezone('US/Pacific'))
    date = timeNow.strftime('%Y-%m-%d')
    apiKey = "7abecd04-7a07-4c3a-8016-54558f852560" #my api key
    cimisAPIurl = f"https://et.water.ca.gov/api/data?appkey={apiKey}&targets=75&startDate={date}&endDate={date}&dataItems=hly-rel-hum"

    r = requests.get(cimisAPIurl) #requesting data specified in url, hly-rel-hum, hourly relative humitity
    r_data = json.loads(r.content)
    records = r_data["Data"]["Providers"][0]["Records"] #this function can break the program because the CIMIS data base doesn't update in time
    for item in records: #pulling humidity
        if item["Date"] == date:
            humid = item["HlyRelHum"]["Value"]
            if humid != 'None':
                return humid
            else:
                return 69
```

Temperature changing functions:
When the temperature increase or decrease buttons are pressed these functions are called changing the global desired temperature level.

```
GPIO.add_event_detect(incTempPin, GPIO.RISING, callback = inc_HVACtemp, bouncetime = 300)#interrput for inc temp
GPIO.add_event_detect(decTempPin, GPIO.RISING, callback = dec_HVACtemp, bouncetime = 300)#interrput for dec temp
GPIO.add_event_detect(DoorPin, GPIO.RISING, callback = toggleDoor, bouncetime = 100)#interrput for door open
```

```
#function to inc desired temp
def inc_HVACtemp(channel):
    global HVACtemp
    HVACtemp = HVACtemp + 1
    print("HVACtemp = ", HVACtemp, "F")
#function to dec desired temp
def dec_HVACtemp(channel):
    global HVACtemp
    HVACtemp = HVACtemp - 1
    print("HVACtemp = ", HVACtemp, "F")
```

Main HVAC loop:
A while true loop constantly checks the desired temperature and functions accordingly. The LCD will display the set temperature and current temperature, the HVAC mode, and the status of the motion light. If the current temperature is 3 above the set temperature, it will go into AC mode and display AC on the screen as well as turn on the blue LED. If the current temperature is 3 below the set temperature, it will go into heat mode and display HT on the screen as well as turn on the red LED. If the yellow button is pressed, the “door” button, the loop will set the HVAC to off mode and display OFF, the two temperatures will be displayed as well as light status. The temperature can still be set while in off mode. If the door is opened the LCD will flash “Door Open” for a time and then when the door closes it will flash “Door closed!”. The loop constantly checks both door status and light status to display the appropriate information on the LCD. It also bounds the desired temperature in between 65 and 85.

```
#main loop, try catch for keyboard interrput to clean GPIO and clear LCD
try:
#LCD initialization
  mcp.output(3,1)    # turn on LCD backlight
  lcd.begin(16,2)    # set number of LCD lines and columns
  #main loop, constantly updated LCD and values, detects temp and sets HVAC accordingly
  while True:
    LCDclear()
    lcd.setCursor(0,0)  # set cursor position
    if weather >= HVACtemp + 3:
      HVACmode = 1
    elif weather <= HVACtemp - 3:
      HVACmode = 2

    print(doorOpen)
    print(HVACmode)
    if doorOpen == False and HVACmode == 1: #if HVAC mode is 1, it is set to AC, led is set, and checks if the motion sensor light is on and displays its
    status, displays HVAC temp/weather in F
      energyConsumption = 18
      GPIO.output(ACledPin, GPIO.HIGH)
      GPIO.output(HeatledPin, GPIO.LOW)
      HVACmode = 1
      if GPIO.input(SensorledPin):
        lcd.message('HVAC:AC ' + str(HVACtemp)+'/'+str(weather) + 'F\nL:ON')
      else:
        lcd.message('HVAC:AC ' + str(HVACtemp)+'/'+str(weather) + 'F\nL:OFF')
      print("HVACtemp = ", HVACtemp, "F")
      print("AC ON")
    elif doorOpen == False and HVACmode == 2: #if HVAC mode is 2, it is set to HT for heat, led is set, and checks if the motion sensor light is on and
    displays its status, displays HVAC temp/weather in F
      energyConsumption = 36
      GPIO.output(HeatledPin, GPIO.HIGH)
      GPIO.output(ACledPin, GPIO.LOW)
      HVACmode = 2
      if GPIO.input(SensorledPin):
        lcd.message('HVAC:HT ' + str(HVACtemp)+'/'+str(weather) + 'F\nL:ON')
      else:
        lcd.message('HVAC:HT ' + str(HVACtemp)+'/'+str(weather) + 'F\nL:OFF')
      print("HVACtemp = ", HVACtemp, "F")
      print("Heat ON")
    elif doorOpen == True: #if HVAC mode is 0, it is turned off, led is set, and checks if the motion sensor light is on and displays its status, displays HVAC
```

```
    elif doorOpen == True: #if HVAC mode is 0, it is turned off, led is set, and checks if the motion sensor light is on and displays its status, displays HVAC
    temp/weather in F
        GPIO.output(ACledPin, GPIO.LOW)
        GPIO.output(HeatledPin, GPIO.LOW)
        HVACmode = 0
        if GPIO.input(SensorledPin):
            lcd.message('HVAC:OFF\n' + str(HVACtemp)+'/'+str(weather) + 'F L:ON')
        else:
            lcd.message('HVAC:OFF\n' + str(HVACtemp)+'/'+str(weather) + 'F L:OFF')
        print("Door Open! HVAC OFF")
    #bounds HVAC temp between 65 and 85
    if HVACtemp < 65:
        HVACtemp = 65
    elif HVACtemp > 85:
        HVACtemp = 85
    #reading from DHT
    DHT_read()
    print("feels like: {0:0.1f}F".format(weather))
    time.sleep(1)
except KeyboardInterrupt:
  GPIO.cleanup()
  LCDclear()
```

```
GPIO.add_event_detect(incTempPin, GPIO.RISING, callback = inc_HVACtemp, bouncetime = 300)#interrput for inc temp
GPIO.add_event_detect(decTempPin, GPIO.RISING, callback = dec_HVACtemp, bouncetime = 300)#interrput for dec temp
GPIO.add_event_detect(DoorPin, GPIO.RISING, callback = toggleDoor, bouncetime = 100)#interrput for door open
```

Door functions:
These functions are for handling the LCD response of the door being opened or closed and the logic response. When the door button is pressed toggleDoor is called, the global doorOpen flag is set to either True or False depending on the previous status of the door, this status is then passed to the doorNotif function. This function displays when the door is open or closed on the LCD for a few seconds. The toggleDoor function also displays the energy consumption and cost in the terminal as it wasn't specified in the instructions to have it display on the LCD.

```
doorNotifevent = None #flag to know if the door open/close message was displayed so it wont over take standard lcd output

#making lcd display door open/closed, takes doorOpen flag to check its status and act acorrdingly
def doorNotif(doorOpen):
    try:
        global doorNotifevent
        if doorOpen:
            mcp.output(3,1)     # turn on LCD backlight
            lcd.begin(16,2)     # set number of LCD lines and columns
            LCDclear()
            lcd.setCursor(0,0)  # set cursor position
            lcd.message('Door Open!')
            doorNotifevent = True
        else:
            mcp.output(3,1)     # turn on LCD backlight
            lcd.begin(16,2)     # set number of LCD lines and columns
            LCDclear()
            lcd.setCursor(0,0)  # set cursor position
            lcd.message('Door Closed!')
            doorNotifevent = True
    except KeyboardInterrupt:
        GPIO.cleanup()
        LCDclear()


#sets door status flag and prints energy/cost
def toggleDoor(channel):
    global doorOpen
    if doorOpen == False:
        doorOpen = True
        doorNotif(doorOpen)
        energyCost = energyConsumption * 0.5
        print("Energy: ",energyConsumption,"KWh", "Cost: $",energyCost)
    elif doorOpen == True:
        doorOpen = False
        doorNotif(doorOpen)
        energyCost = energyConsumption * 0.5
        print("Energy: ",energyConsumption,"KWh", "Cost: $",energyCost)
```