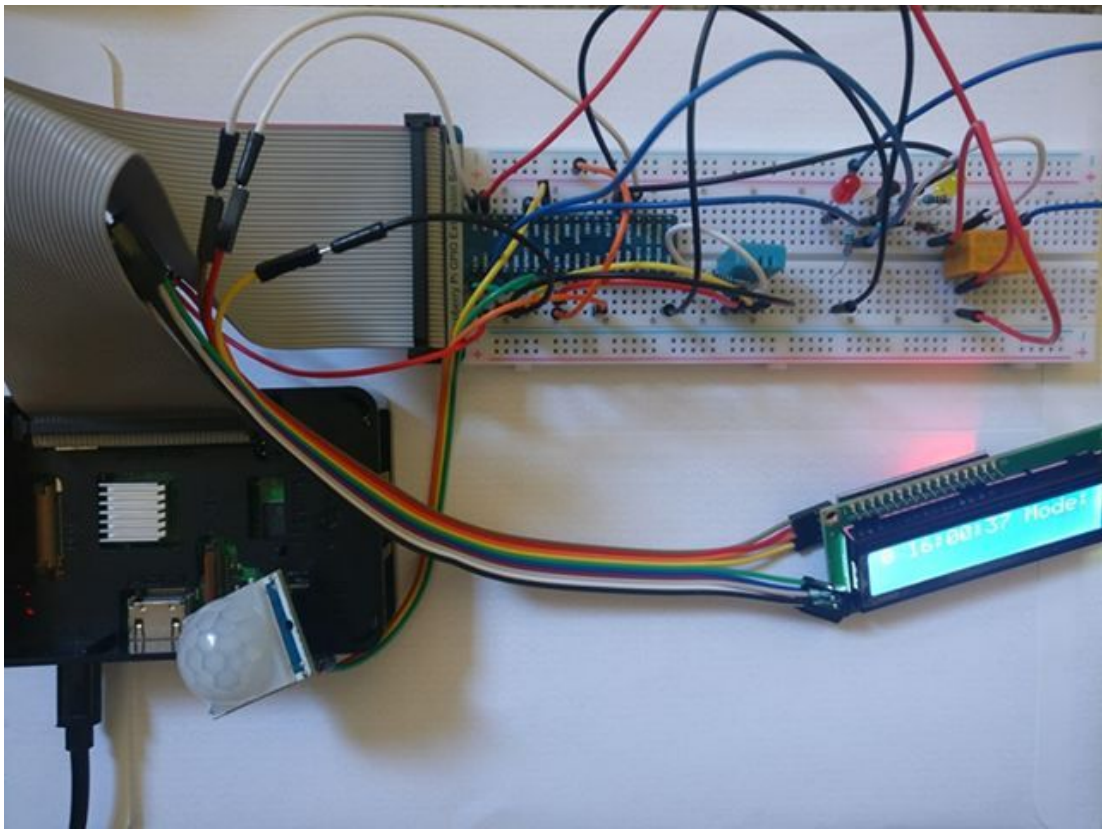
Final Project
Atmosphere Monitoring System

Sienna Ballot: 16881771
Isaac Anasatcio Bates: 81096903
Tong Ray Huang: 52347257
Hayden Z. Yu: 66185399

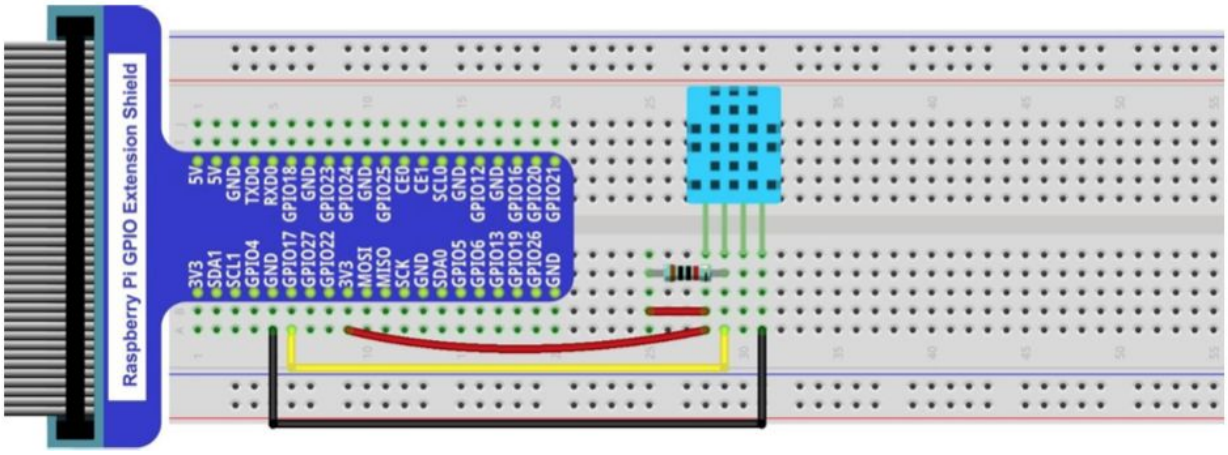
Video Link
https://drive.google.com/open?id=1N5S12UUpWSZw3OSZd-kSMV2Lt55a4iFr

Code Table of Contents

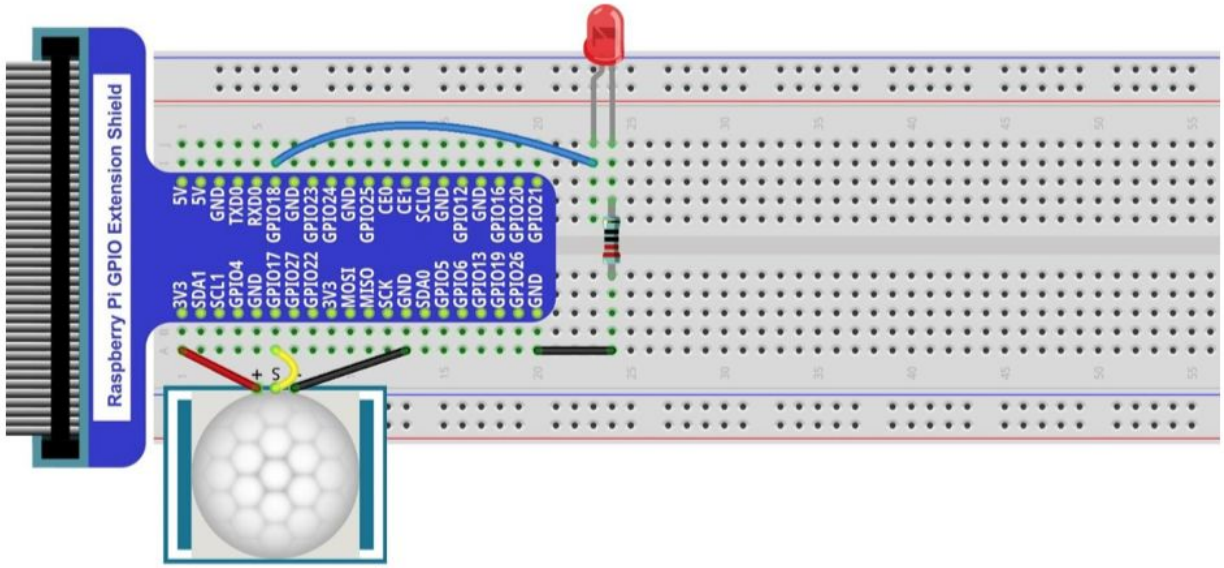
Breadboard Set-Up Picture



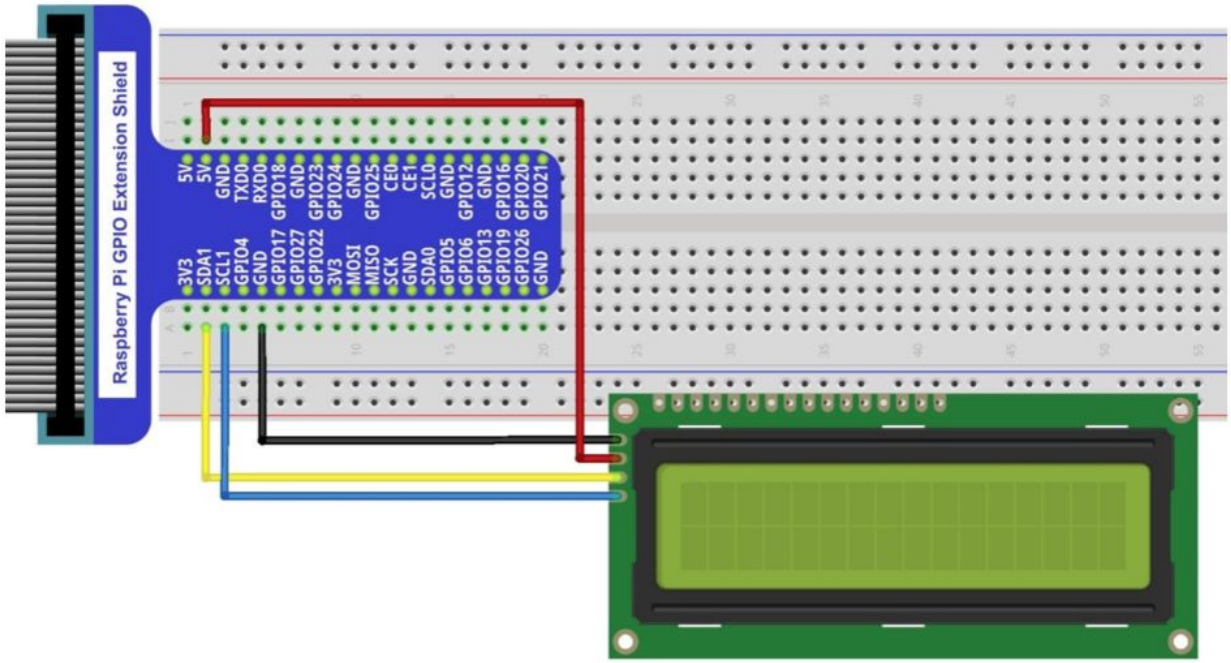
DHT



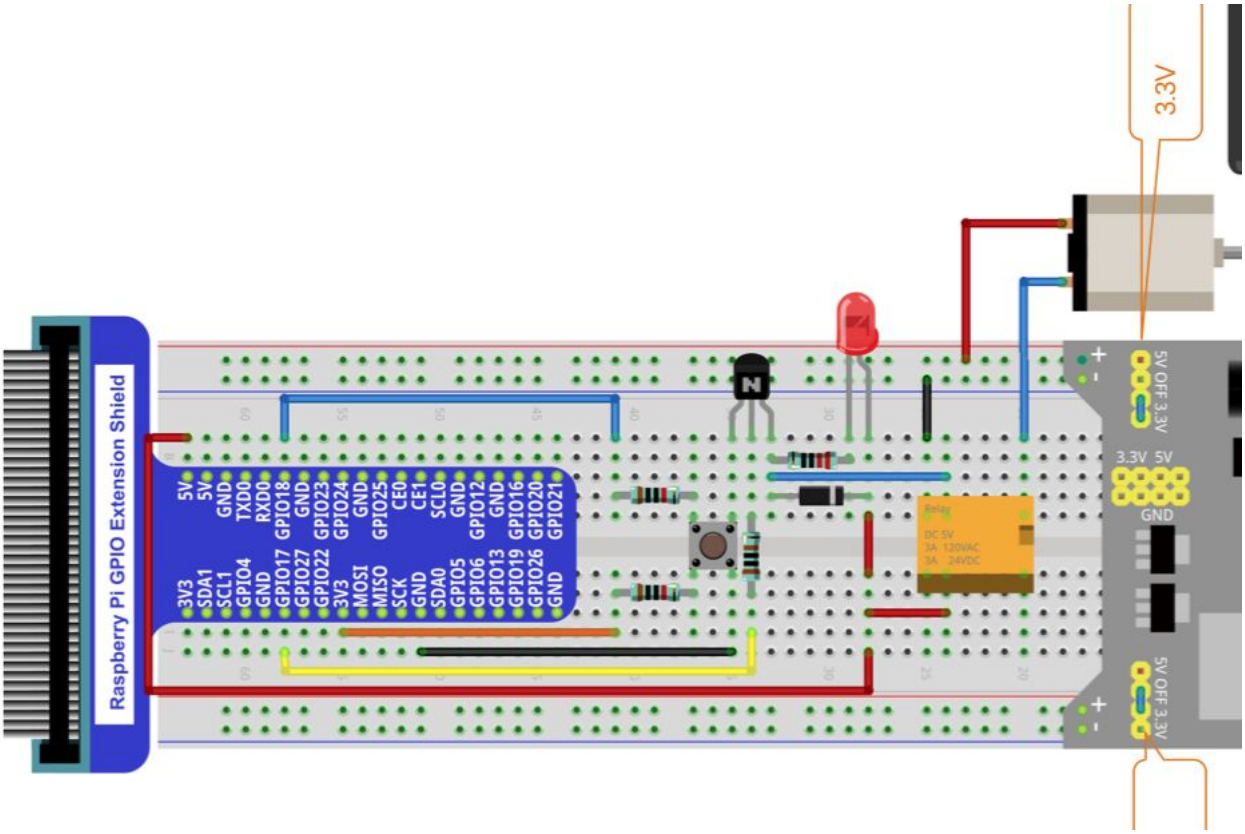
PIR Sensor



LCD



Relay



(Did not use the board at the end or button)

Explanation

Our board setup has a total of 16 parts which is the count not including all of the wires that were required to connect everything. All of the tutorials provided by freenove were followed accurately except the relay tutorial. For the DHT the 1st leg is connected to the 3.3v source on the GPIO board. From there one side of the resistor is connected to the 3.3v source and the other side is connected to the second leg of the DHT. The second leg of the DHT is also connected to the

GPIO port. The next component connected is the PIR sensor and its' LED for indication of movement. The PIR sensor has three legs: the first one is connected to the 3.3V source, second is connected to the GPIO port, and the last is connected to the ground. As for the LED the positive leg is connected to the GPIO port and the positive side is connected in series with a 220-ohm resistor and the ground. The next component connected is the LCD this has a series of wires that connect it to the board. It has a total of 4: one is connected to the ground, number two is connected to the 5v source, number 3 is connected to the SDA1 connection on the GPIO board, and the last one is connected to the SCL1 on the GPIO board. The last component connected is the relay. This component has a total of 6 legs to connect. The first leg is not connected to anything. The second leg is connected to the ground. The third leg is connected to the 5v source, the LED in series with a 220-ohm resistor, and diode. The fourth leg is connected to the NPN transistor which is connected to the ground and 1 k-ohm resistor. The fifth leg is not connected to anything. The sixth leg is connected to one side of the motor and the other side of the motor is connected to the 3.3v source.

Design Details

The project begins within the FinalProj.py file. This code file is the heart of the project because it is in charge of setting up the GPIO pins as well as starting the threads for the LCD, DHT, and Relay. DHT then is the file that is in charge of retrieving the humidity and temperature information both locally and from the CIMIS database. The CIMIS database information is retrieved by the CIMIS.py code file. After retrieving the humidity and temperature and after a series of calculations it gives the time the sprinkler should run for. This information is then sent to the LCD. The LCD then displays the hourly averages for the humidity and temperature, comparison between local/CIMIS values, values for irrigation time calculated for both CIMIS/local values and potential water savings. All of the information displayed on the LCD is handled by the LCD.py code file. After calculating the values for irrigation time relay.py handles turn the sprinkler system on for that amount of time. Irrigation may be interrupted when the motion sensor picks up movement after the movement is no longer present the irrigation will resume for the remaining irrigation time. Motion sensing is handled by SenseLED.py code file which sends a value to Relay when motion is sensed so that the irrigation can be paused.

FinalProj.py (Team Collaboration)

As stated before this code file is the heart of the entire project because it sets all of the operations in motion. In the beginning, the code the modules that are needed throughout this file are imported. After this is completed the next step that takes place is the GPIO pins are set for each individual component connected to the GPIO.

```
import threading
import RPi.GPIO as GPIO
import time
from datetime import datetime
import DHT
import LCD
import SenseLED
```

```
thermoPin = 11  # pin for the thermo sensor
ledPin = 12     # pin for motion LED
sensorPin = 16  # pin for the motion sensor
relayPin = 7 # pin for the relay
```

Once these steps are completed then the functions “setup” and “loop”. “Setup” sets the input and output pins for the components sensorPin and relayPin. The function “Loop” creates all of the threads needed to start the program which listed respectfully are: DHT and LCD.

```
def setup():
    # setup the board input and output pins
    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)
    GPIO.setup(sensorPin, GPIO.IN)
    GPIO.setup(relayPin, GPIO.OUT)
    GPIO.setup(ledPin, GPIO.OUT)
    GPIO.output(relayPin, True)
```

```
def loop():
    #Start threads
    t_DHT = None
    print("starting DHT thread")
    t_DHT = threading.Thread(target=DHT.loop)
    t_DHT.daemon = True
    t_DHT.start()
    t_LCD = None
    print("starting LCD Thread")
    t_LCD = threading.Thread(target=LCD.display_cimis)
    t_LCD.daemon = True
    t_LCD.start()
    while(True):#function loops forever
        time.sleep(0.1)
```

CIMIS.py (Tong)

CIMIS.py is the code file in charge of communicating with the CIMIS website so that it can get the variable values needed to calculate the necessary irrigation time. The code first begins in the function “getcimisdata()” this is where the information from the website is retrieved. It starts off by setting the appkey which is used to access the CIMIS API. Next is the information we send the website so that we can receive the information what we want. “site = [75]” is used to get the information for Irvine specifically. The is set the interval in which we pull information which is set to hourly. An array of all the parameters that we would like to retrieve is made. The URL of the site is then set, and it contains the appkey and the items which we would like to receive. The link to the URL is then pulled using the request library in python and it is then converted into the JSON format. Once the information is pulled then it needs to be sorted into the information wed like. To do this a for loop iterates through the information and stores it into the hour_entries global array. The information is then set for the DHT file so that it may access it. After that, if the information has not already been updated the data is pulled from our own stored information using the function “get Hour Data”.

CODE PICTURES BELOW

```
def getHourData(hour, date):
    hour_str = None
    hour += 1
    if hour < 10:
        hour_str = '0' + str(hour)
    else:
        hour_str = str(hour)
    hour_str = hour_str + '00'
    print("CIMIS hour: ", hour_str)
    #pprint.pprint("Hour String")
    #pprint.pprint(hour_str)
    for i in hour_entries:
        if hour_entries[i]['Hour'] == hour_str and hour_entries[i]['Date'] == date:
            if hour_entries[hour]['HlyAirTmp']['Value'] != None:
                DHT.cimisTemp = float(hour_entries[hour]['HlyAirTmp']['Value'])
                DHT.cimisET = float(hour_entries[hour]['HlyEto']['Value'])
                DHT.cimisHumidity = float(hour_entries[hour]['HlyRelHum']['Value'])
                print("Value Found")
                return
            else:
                DHT.cimisTemp = None
                DHT.cimisET = None
                DHT.cimisHumidity = None
                print("Value not updated")
        #else:
            #print("At " + hour_entries[i]['Hour'])
```

```
def getcimisdata(hour, date):
    #appKey = 'a28ddf14-568e-45b8-8050-6925a8ff77e1'  # cimis appKey
    #appKey = '3cae5dfd-ef01-49e4-b6f4-0441a144c5e5'
    #appKey = '952d594c-ff2e-4011-b1d9-8d62e6300ec8'
    #appKey = 'fe36cc18-4506-4cca-8ba9-c903131fde2f'
    appKey = 'fe36cc18-4506-4cca-8ba9-c903131fde2f'
    # list of CIMIS station ID's from which to query data
    sites = [75]  # uncomment to query single site
    sites = [str(i) for i in sites]  # convert list of ints to strings
    ItemInterval = 'hourly'
    # start date fomat in YYYY-MM-DD
    start = date
    # end date fomat in YYYY-MM-DD
    # e.g. pull all data from start until today
    end = datetime.datetime.now().strftime("%Y-%m-%d")

    station = sites[0]
    dataItems_list = ['hly-air-tmp',
                      'hly-eto',
                      'hly-asce-eto',
                      'hly-asce-etr',
                      'hly-precip',
                      'hly-rel-hum',
                      'hly-res-wind']
    dataItems = ','.join(dataItems_list)
    url = ('http://et.water.ca.gov/api/data?appKey=' + appKey + '&targets='
        + str(station) + '&startDate=' + start + '&endDate=' + end +
        '&dataItems=' + dataItems +'&unitOfMeasure=E')

    #test = requests.head(url)
    #if test.status_code == 302:
    #    print("error 302")
    #else:
    #    print("200")

    print(url)
    r = requests.get(url).json()
    #print(type(r))
    #pprint.pprint(r)

    data = r['Data']
    #print(type(data))
    #pprint.pprint(data)

    providers = data['Providers']
    #now a list and access using providers[int]
    #print(type(providers))
    #pprint.pprint(providers)

    access_list = providers[0]
    #print(type(access_list))
    #pprint.pprint(access_list)

    records_list = access_list['Records']
    #print(type(records_list))
    #pprint.pprint(records_list)

    #moved hour_entries dictionary

    for i,val in enumerate(records_list):
        hour_entries[i] = val

    #print(type(hour_entries))
    pprint.pprint(hour_entries)

    #targ_tmp = hour_entries[0]['HlyAirTmp']['Value']
    #targ_eto = hour_entries[0]['HlyEto']['Value']
    #targ_hum = hour_entries[0]['HlyRelHum']['Value']
    #print(targ_tmp)
    #print(targ_eto)
    #print(targ_hum)
    if (not hour_entries[hour]['HlyEto']['Value']):
        DHT.cimisTemp = hour_entries[hour]['HlyAirTmp']['Value']
        DHT.cimisET = hour_entries[hour]['HlyEto']['Value']
        DHT.cimisHumidity = hour_entries[hour]['HlyRelHum']['Value']
        return
```

DHT.py (Sienna)

This file is in charge of taking both the local and the CIMIS data received and calculating the time the irrigation should run for. It starts in the function, called “loop()”. In this function it begins by initializing an output data file with all the headers for the data that will be stored within the file. It then enters a loop where the humidity and temperature are endlessly checked. The local information for humidity and data is pulled using the DHT sensor and the Freenove library function. The data is then checked and if it is useful to use then it is then averaged with the data recorded in the past hour and then increments the data count by one. Once the current hour has ended which is determined by either the data count or by the time kept then it gets all of the averages and appends them to the global list of hourly data. The code then enters the second large function in the file which is the “getIrrigatrionTime()”. In this function it starts by accessing the CIMIS to check if the data in the oldest hour in the local data list has been entered, If this is not the case then all the CIMIS data is set to none and the hourly information is written to the file. If the site has been updated, then the function iterated through the local hour data list and used the derating calculations to calculate the total ETO for all the hours whose data has been updated on the CIMIS site. The information is then removed from the list. It also calculates the ETO for only the current hour to find the additional water that will be used for the current hour. This total ETO is then used in the equation provided to us to calculate the water needed per day and divide it by 24 to find the water needed per hour (in gallons). This amount is then divided by water flow in the

system to get the time the irrigation system should be ran. A thread for the relay.py file so that the relay can start the irrigation for the time calculated.

```
def getIrrigationTime():
    global irrigationTime
    global ET0
    global cimisET
    global cimisHumidity
    global cimisTemp
    global localHumidity
    global localTemp
    global displaycimis

    # get current date and time info
    result = time.localtime(time.time())

    # get ET, humidity, and temp from CIMIS
    # check if CIMIS site has been update for first hour in list
    CIMIS.getcimisdata(localHourly[0][0], localHourly[0][1])#result.tm_hour, date)

    # if CIMIS has not been update for first hour in list
    # skip irrigation and wait for data updates
    if (not cimisET):
        cimisET = None
        cimisHumidity = None
        cimisTemp = None
        gallons = None
        irrigationTime = None
        additionalWater = 0
        waterSaved = 1020
    # otherwise, get data from CIMIS for all hours in list that have been updated
    # and get ET0 and irrigation time to turn on motor
    else:
        displaycimis = True
        while True:
            # get cimis data for the next hour in the list
            CIMIS.getHourData(localHourly[0][0], localHourly[0][1])

            # if the cimis has not been updated for that hour then break
            if (cimisET == None or len(localHourly) == 0):
                break

            # get derating factors for the hour in the list to derate ET0
            humidityDerate = cimisHumidity / localHourly[0][2]
            tempDerate = localHourly[0][3] / cimisTemp
            currET = cimisET * (tempDerate * humidityDerate)        # get the ET0 for the current time to calculate additional water used f
            ET0 = ET0 + (cimisET * (tempDerate * humidityDerate))   # add derated ET0 to find total ET0 for all hours whose data has been u
            localHourly.pop(0)                                      # remove hour from list if data has been used

        print("ET0: ", ET0)

        # get total gallons of water needed per hour for total ET0 (using gallons needed per day formula divided by 24)
        gallons = ((ET0 * pf * sqft * conversion) / IE) / 24
        print("Gallons Total Needed: ", gallons)

        # get gallons of water needed only for the ET0 of the current hour
        galHour = ((currET * pf * sqft * conversion) / IE) / 24
        print("Gallons for only this hour: ", galHour)

        # additional water is the total water needed minus the required water for that hour
        additionalWater = gallons - galHour
        print("Additional Water: ", additionalWater)

        # water saved is the rate of water per hour minus the total water used
        waterSaved = 1020 - gallons
        print("Water Saved: ", waterSaved)

        # get time to run irrigation in minutes
        # gallons needed / (gallons per min) = minutes needed to run
        irrigationTime = gallons / systemRate
        print("Irrigation Time (min): ", irrigationTime)

        # signal relay to turn on and start relay thread
        Relay.systemState = True
        t = None
        print("starting Relay/Motor thread")
        t = threading.Thread(target=Relay.loop)
        t.daemon = True
        t.start()

    # open output file to store information for the hour
    date = str(result.tm_mon)+'/'+str(result.tm_mday)+'/'+str(result.tm_year)
    row = [date, str(result.tm_hour), str(ET0), str(localHumidity), str(localTemp), str(cimisET), str(cimisHumidity), str(cimisTemp), str(g

    with open('output.csv', mode='a') as outputFile:
        outputWriter = csv.writer(outputFile)
        outputWriter.writerow(row)
```

```
def loop():
    global localHumidity
    global localTemp
    global display

    # intialize output file and write headers to the file
    row = ['Date (MM/DD/YYYY)','Hour','Local ET0', 'Local Humidity', 'Local Temp(F)', 'CIMIS ET0', 'CIMIS Humidity', 'CIMIS Temp (F)', 'Gal

    with open('output.csv', mode='a') as outputFile:
        outputWriter = csv.writer(outputFile)
        outputWriter.writerow(row)

    outputFile.close()

    dht = DHT.DHT(thermoPin)        # creates DHT class object
    count = 0                       # initialize minute count for an hour

    while(True):
        chk = dht.readDHT11()
        print("Check DHT: ", chk)

        # CONVERT CELSIUS TO FAHRENHEIT
        if (chk is dht.DHTLIB_OK):
            # if the start of an hour, do not need to average 2 values
            if (localHumidity == 0 and localTemp == 0):
                localHumidity = dht.humidity
                localTemp = 32 + (1.8*dht.temperature)
            # otherwise avergae the new data with the past averages of the hour
            else:
                localHumidity = (localHumidity + dht.humidity)/2
                localTemp = (localTemp + (32+(1.8*dht.temperature)))/2

        count += 1
        print("Local Humidity: ", localHumidity)
        print("Local Temperature: ", localTemp)

        # check CIMIS for new data
        # if there is new data for the hour
        result = time.localtime(time.time())
        if (count >= 60 or result.tm_min == 59):
            # format month for date string
            if (result.tm_mon/10 == 0):
                month = '0'+str(result.tm_mon)
            else:
                month = str(result.tm_mon)
            # format day for date string
            if (result.tm_mday/10 == 0):
                day = '0'+str(result.tm_mday)
            else:
                day = str(result.tm_mday)
            # formulate date string and send as argument to CIMIS function
            date = str(result.tm_year)+'-'+month+'-'+day

            # make a list containing the date and local data for the current hour
            # and append this data to the localHourly list
            data = [result.tm_hour, date, localHumidity, localTemp]
            localHourly.append(data)

            getIrrigationTime()

            # reset variables to calculate new data for the new hour
            localHumidity = 0
            localTemp = 0
            count = 0

        # sleep for 1 minute
        display = True #enable LCD to display
        time.sleep(60)
        display = False #disable LCD to display
        time.sleep(0.5)
```

Realy.py (Isaac & Sienna)

This file holds the code that is in charge of communicating with DHT and SenseLED to turn on the motor for irrigation. Based on the calculated irrigation time received from DHT the motor (which would be the water system in an actual irrigation system) will turn on and run for the given time. The motor operation can be interrupted if it receives a signal from SenseLED which sends a signal to stop the irrigation if it senses movement. Once the sensor no longer senses movement then the irrigation will resume, but once the pause is over a minute the irrigation will resume.

Below are the screenshots of the code that deal with setting up connections between code files as well as setting the pins so that raspberry pi can communicate with components.

```
def setup():
    # setup the board input and output pins
    GPIO.setmode(GPIO.BOARD)
    GPIO.setwarnings(False)
    GPIO.setup(sensorPin, GPIO.IN)
    GPIO.setup(relayPin, GPIO.OUT)
    GPIO.setup(ledPin, GPIO.OUT)

def destroy():
    GPIO.output(relayPin, True)
    GPIO.output(ledPin, GPIO.LOW)
    GPIO.cleanup()

# main function to start program
if __name__ == '__main__':
    global systemState
    print("Program starting...")
    setup()
    try:
        systemState = True
        loop()
    except KeyboardInterrupt:
        destroy()
        exit()
```

```
import threading
import time
import RPi.GPIO as GPIO
import DHT
import SenseLED as PIR

thermoPin = 11
ledPin = 12
sensorPin = 16
relayPin = 7
output = True          # system initialized to off
```

Below is the code which starts the thread for the sensor so that the irrigation can be stopped if needed. So, while irrigation is running the sensor is always on alert waiting for any movement to arise.

```
# start a new thread for the PIR sensor
t = None
print("starting PIR thread")
t = threading.Thread(target=PIR.loop)
t.daemon = True
t.start()
```

The while loop is where the irrigation is run. The first if statement is for checking if the sensor has been tripped. If no motion is sensed and the irrigation time runs out then the irrigation thread is terminated and irrigation is stopped.

```
while (True):
    # if the motion sensor triggered, pause system
    if (PIR.senvar == 1):
        print("Pause irrigation")
        output = True
        GPIO.output(relayPin, True) # send signal to relay to turn off the motor

        # loop to wait 1 min or until motion no longer detected
        pauseStart = time.time()
        while(True):
            # break loop and resume irrigation if 1 min exceeded
            if ((time.time()-pauseStart) > 60):
                output = False
                GPIO.output(relayPin, False)
                break
            # resume irrigation if motion no longer detected
            if (PIR.senvar == 0):
                output = False
                GPIO.output(relayPin, False)
                break

            runTime += 0.5      # add the extra paused time to the run time
            time.sleep(0.5)     # quick sleep delay
        print("Resume Irrigation")

    # if irrigation pause over a minute, resume irrigating
    if ((time.time()-start) > runTime):
        output = True
        GPIO.output(relayPin, True)
        break

    # 0.5 second sleep delay for loop/sensor
    time.sleep(0.5)

# when finished irrigation time, kill PIR thread
PIR.start = False
print("Stop irrigation")
```

SenseLED.py (Tong)

This code file is in charge of storing the code that deals with the motion sensor. First, the modules needed are imported and then the GPIO pins are set. After inputs and outputs are set up within the setup () function. After the loop function is where the movement is handled. A global variable senvar is used to that the irrigation can be stopped in Relay.py. If the motion is sensed, then the LED is turned on and senvar is set to 1. Once the motion is no longer sensed then senvar is reset to zero that irrigation can be resumed.

```
import RPi.GPIO as GPIO
import time

ledPin = 12
sensorPin = 16
senvar = 0          # variable to signal to relay thread to pause irrigation
start = True

def setup():
    print('IR Program is starting...')
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ledPin, GPIO.OUT)
    GPIO.setup(sensorPin, GPIO.IN)

def loop():
    global senvar

    # use start variable from relay as interrupt for loop
    while start:
        # read the signal from the motion sensor pin
        i = GPIO.input(sensorPin)
        # if motion sensed by sensor, turn on LED
        if i == GPIO.HIGH:
            GPIO.output(ledPin, GPIO.HIGH)
            senvar = 1          # pause irrigation

        # if motion not sensed by sensor, turn off LED
        elif i == GPIO.LOW:
            GPIO.output(ledPin, GPIO.LOW)
            senvar = 0          # resume irrigation

        # small delay
        time.sleep(0.1)

    # when relay loop is finished, set all outputs to low
    GPIO.output(ledPin, GPIO.LOW)
    senvar = 0
    return

def destroy():
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    try:
        loop()
    except KeyboardInterrupt:
        destroy()
```

LCD.py (Hayden)

This code file deals with displaying the rolling message which contains CPU Temp, Current Time, local temperature, local humidity, CIMIS humidity, CIMIS temperature, CIMIS ET, local ET, water saving and added water. The function “get_cpu_temp()” retrieves the CPU temp by communicating with the pi through a file path. The function loop () then prints the CPU temp to the LCD. The “display_cimis” function is used to display all the information retrieved from locally and the CIMIS website on the LCD. It starts by turning on the LCD and then sets the lines and columns for the information that is going to be displayed. It then enters a while loop that then creates all of the strings by concatenating them into the top line and bottom line. It then sends the strings to the LCD to be displayed.

```
def get_cpu_temp():     # get CPU temperature and store it into file "/sys/class/thermal/thermal_zone0/temp"
    tmp = open('/sys/class/thermal/thermal_zone0/temp')
    cpu = tmp.read()
    tmp.close()
    return '{:.2f}'.format( float(cpu)/1000 ) + ' C'

def get_time_now():     # get system time
    return datetime.now().strftime('%H:%M:%S ')

def loop():
    #print("hi")
    mcp.output(3,1)     # turn on LCD backlight
    lcd.begin(16,2)     # set number of LCD lines and columns
    while(True):
        lcd.clear()
        lcd.setCursor(0,0)  # set cursor position
        lcd.message( 'CPU: ' + get_cpu_temp()+'\n' )# display CPU temperature
        lcd.message( get_time_now() )   # display the time
        sleep(1)
```

```
def display_cimis():#local_temp, local_hum, c_temp, c_hum, local_ET, cimis_ET, water_saving, addi_water):
    mcp.output(3,1)     # turn on LCD backlight
    lcd.begin(16,2)     # set number of LCD lines and columns
    mode = None #mode for when the water is on or off
    sleep(1) #wait for DHT thread to start
    while True:
        lcd.clear()
        if(Relay.output==False): #water mode
            mode = 'On'
        else:
            mode = 'Off'
#Create strings for the variables
        relay_str = 'Mode: ' + mode + ' '
        local_temp_str = 'Local Temp:' + str(DHT.localTemp) + ' '
        local_hum_str = 'Local Humidity:' + str(DHT.localHumidity) + ' '
        c_temp_str = 'CIMIS Temp:' + str(DHT.cimisTemp) + ' '
        c_hum_str = 'CIMIS Humidity:' + str(DHT.cimisHumidity) + ' '
        local_ET_str = 'Local ET:' + str(DHT.ET0) + ' '
        cimis_ET_str = 'CIMIS ET:' + str(DHT.cimisET) + ' '
        water_saving_str = 'Water Saved:' + str(DHT.waterSaved) + ' '
        addi_water_str = 'Additional Water Used:' + str(DHT.additionalWater) + ' '
#end of create strings
        top_line = get_time_now() + relay_str + local_temp_str + local_hum_str #concatenate strings for top line on LCD
        #print(DHT.displaycimis)
        if(DHT.displaycimis):
            start = time.time()
            bot_line = c_temp_str + c_hum_str + local_ET_str + cimis_ET_str + water_saving_str + addi_water_str #concatenate strings for bo
            while True:
                if (time.time() - start) > 60: #don't display cismis data after a minute
                    DHT.displaycimis = False
                    break
                lcd.setCursor(0,0) # cursor top line
                lcd.message(top_line[:16]) #display top line
                lcd.setCursor(0,1) # cursor bottom line
                lcd.message(bot_line[:16])# display bottom line
                top_line = top_line[1:]+top_line[0]# send first char to last
                bot_line = bot_line[1:]+bot_line[0]# send first char to last
                sleep(0.1)
        else:
            while DHT.display:
                lcd.setCursor(0,0) # cursor top line
                lcd.message(top_line[:16]) #display message onto LCD
                top_line = top_line[1:]+top_line[0]# send first char to last
                sleep(0.1)

        sleep(0.5) #DHT time buffer
```

Adafruit_LCD1602.py, PCF8574.py, Freenove_DHT.py (Freenove)

These are all 3 code files given to us so and each one has a specific purpose.

Adafruit_LCD1602.py was used so that the rolling text could be displayed on the screen.

PCF8574.py is the backpack for the LCD. Freenove_DHT.py is the library for the DHT file.

Our Data

Date (MM/DD/YYYY)	Hour	Local ET0	Local Humidity	Local Temp(F)	CIMIS ET0	CIMIS Humidity	CIMIS Temp (F)	Gallons Needed (gal/hr)	Time Needed (min)	Additional Water (per hour)	Water Saved (per hour)
6/11/2019	17	0	50.63378906	78.61402344	None	None	None	None	None	0	1020
6/11/2019	18	0	50.41634154	77.90227558	None	None	None	None	None	0	1020
6/11/2019	19	0.02542388	51.87691512	77.89999964	0.01	44	79.6	0.175142262	0.010302486	0.117961215	1019.824858
6/11/2019	20	0	57.92644756	76.34794779	None	None	None	None	None	0	1020
6/11/2019	21	0	61.1308246	75.53787657	None	None	None	None	None	0	1020
6/11/2019	22	0	59.0441018	76.10306042	0	75	70.8	0	0	0	1020
6/11/2019	23	0	62.56255317	74.60223086	None	None	None	None	None	0	1020
6/12/2019	0	0	63.50024223	73.68115491	None	None	None	None	None	0	1020
6/12/2019	1	0	62.99999953	73.55750154	0	88	66.1	0	0	0	1020
6/12/2019	2	0	62.06542967	73.51758404	None	None	None	None	None	0	1020
6/12/2019	3	0	60.00001526	74.48140204	None	None	None	None	None	0	1020
6/12/2019	4	0	59.99999619	74.47042541	0	92	61.9	0	0	0	1020

Figure 1: Ran while CIMIS was up

Date (MM/DD/YYYY)	Hour	Local ET0	Local Humidity	Local Temp(F)	CIMIS ET0	CIMIS Humidity	CIMIS Temp (F)	Gallons Needed (gal/hr)	Time Needed (min)	Additional Water (per hour)	Water Saved (per hour)
6/13/2019	16	0	58	73.94000017	None	None	None	None	None	0	1020
6/13/2019	17	0	58	74.10875	None	None	None	None	None	0	1020
6/13/2019	18	0	56.65139586	74.29686957	None	None	None	None	None	0	1020
6/13/2019	19	0	104.499941	74.72753262	None	None	None	None	None	0	1020
6/13/2019	20	0	55	74.05232834	None	None	None	None	None	0	1020
6/13/2019	21	0	54.12496948	74.09750051	None	None	None	None	None	0	1020
6/13/2019	22	0	54.00011825	74.11997803	None	None	None	None	None	0	1020
6/13/2019	23	0	54.25000764	73.94000824	None	None	None	None	None	0	1020
6/14/2019	0	0	53.99999945	73.58210938	None	None	None	None	None	0	1020
6/14/2019	1	0	55.87499964	74.2099972	None	None	None	None	None	0	1020
6/14/2019	2	0	55	73.94035431	None	None	None	None	None	0	1020
6/14/2019	3	0	55.99999905	73.58000017	None	None	None	None	None	0	1020
6/14/2019	4	0	56	73.4	None	None	None	None	None	0	1020
6/14/2019	5	0	56	73.04067566	None	None	None	None	None	0	1020
6/14/2019	6	0	56.99914551	72.8600769	None	None	None	None	None	0	1020
6/14/2019	7	0	57	72.86	None	None	None	None	None	0	1020
6/14/2019	8	0	57	73.03947922	None	None	None	None	None	0	1020
6/14/2019	9	0	57	73.03995605	None	None	None	None	None	0	1020
6/14/2019	10	0	106.4676208	62.80531113	None	None	None	None	None	0	1020

Figure 2: Ran while CIMIS Site was down

The above two pictures were running at two separate times. The first was ran when the site was up and running. All the information was accurate and irrigation times were calculated accordingly. The second picture is for when we ran the program and the site was down. When the program picks up that the site is down (which is indicated by the request timing out if it has not received information within ten seconds) it sets all the values to none and no irrigation time is calculated.

DHT DATA

```
Check DHT:  0
Local Humidity:  57.74212455743691
Local Temperature:  73.38980194091798
Check DHT:  0
Local Humidity:  57.371062278718455
Local Temperature:  73.39490097045899
Check DHT:  -2
Local Humidity:  57.371062278718455
Local Temperature:  73.39490097045899
Check DHT:  0
Local Humidity:  57.68553113935923
Local Temperature:  73.3974504852295
Check DHT:  -1
Local Humidity:  57.68553113935923
Local Temperature:  73.3974504852295
Check DHT:  0
Local Humidity:  57.842765569679614
Local Temperature:  73.39872524261474
Check DHT:  -2
Local Humidity:  57.842765569679614
Local Temperature:  73.39872524261474
Check DHT:  0
Local Humidity:  57.92138278483981
Local Temperature:  73.39936262130738
Check DHT:  0
Local Humidity:  57.9606913924199
Local Temperature:  73.39968131065369
Check DHT:  0
Local Humidity:  57.98034569620995
Local Temperature:  73.39984065532684
Check DHT:  0
Local Humidity:  58.990172848104976
Local Temperature:  73.48992032766343
Check DHT:  0
Local Humidity:  57.49508642405249
Local Temperature:  73.53496016383173
Check DHT:  0
Local Humidity:  57.247543212026244
Local Temperature:  73.46748008191587
Check DHT:  -2
Local Humidity:  57.247543212026244
Local Temperature:  73.46748008191587
```

The above data is the local humidity and temperature which is printed to the terminal whenever the DHT sensor records it.