EECS113 Final Project Report

By Group 10
Team Member:
Yuting Jiang yutinj2@uci.edu
Zehua Wang zehuaw6@uci.edu
Zhifang Zeng zhifanz1@uci.edu
Instructor: Professor Fadi Kurdahi

Responsibilities:

Yuting Jiang yutinj2@uci.edu: Setting up the board and circuit for the project, encapsulate packages for the LCD module and relay to be used by the main function;

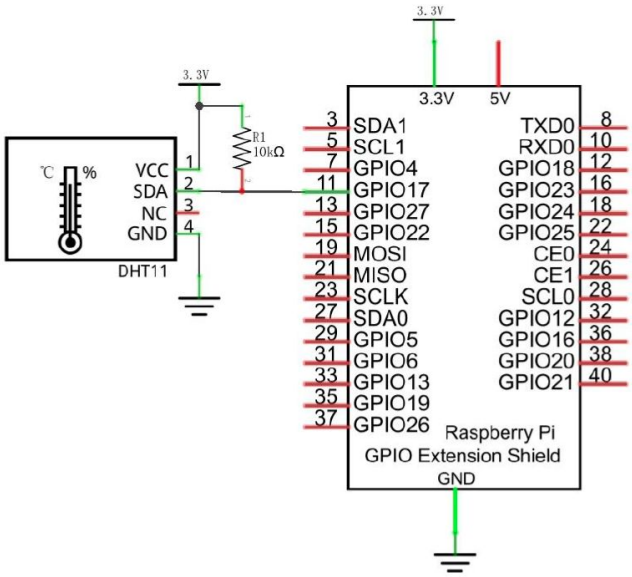
Zehua Wang zehuaw6@uci.edu: Encapsulate the temperature and humidity sensor codes as well as the motion sensor codes, enables them to be used by the main function;

Zhifang Zeng zhifanz1@uci.edu: Implement the code reading data from the CIMIS website, do the calculation, and combine the modules together.

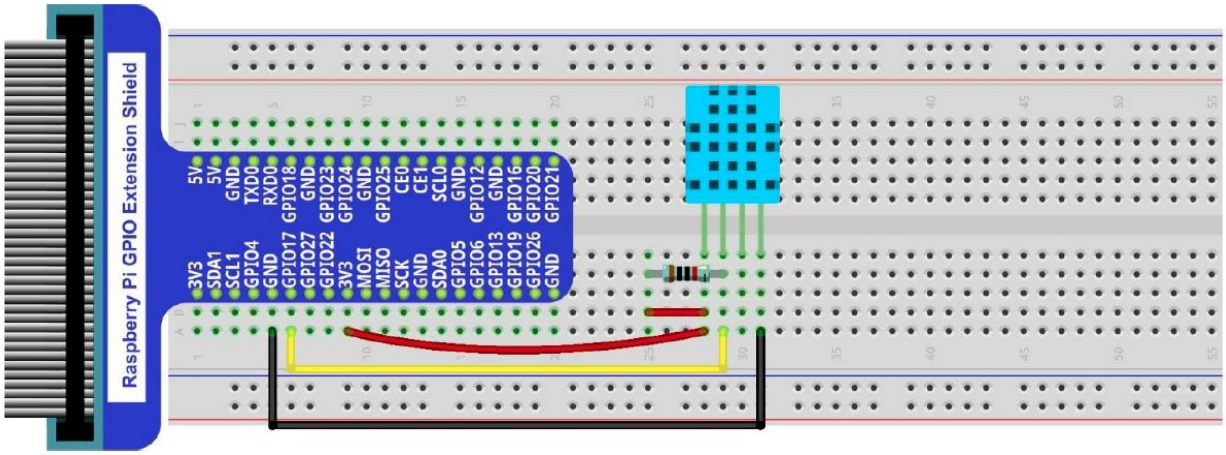
Part 1: Board Setup (by Yuting Jiang)

1. Hygrothermograph DHT11

Scheme:

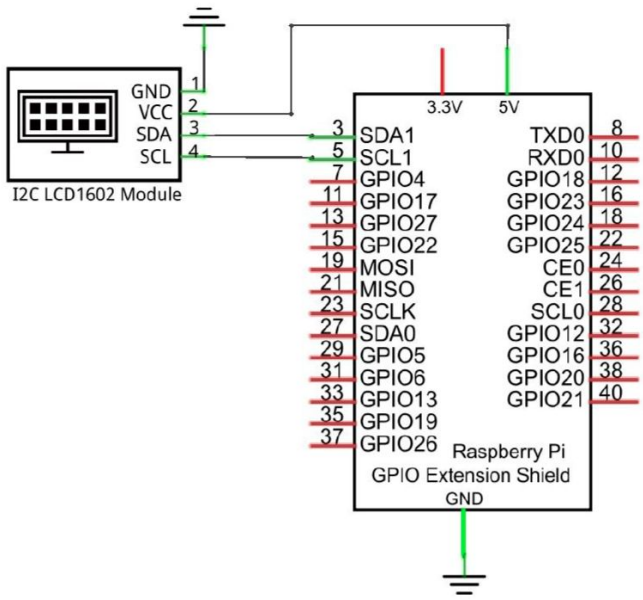


On Board:

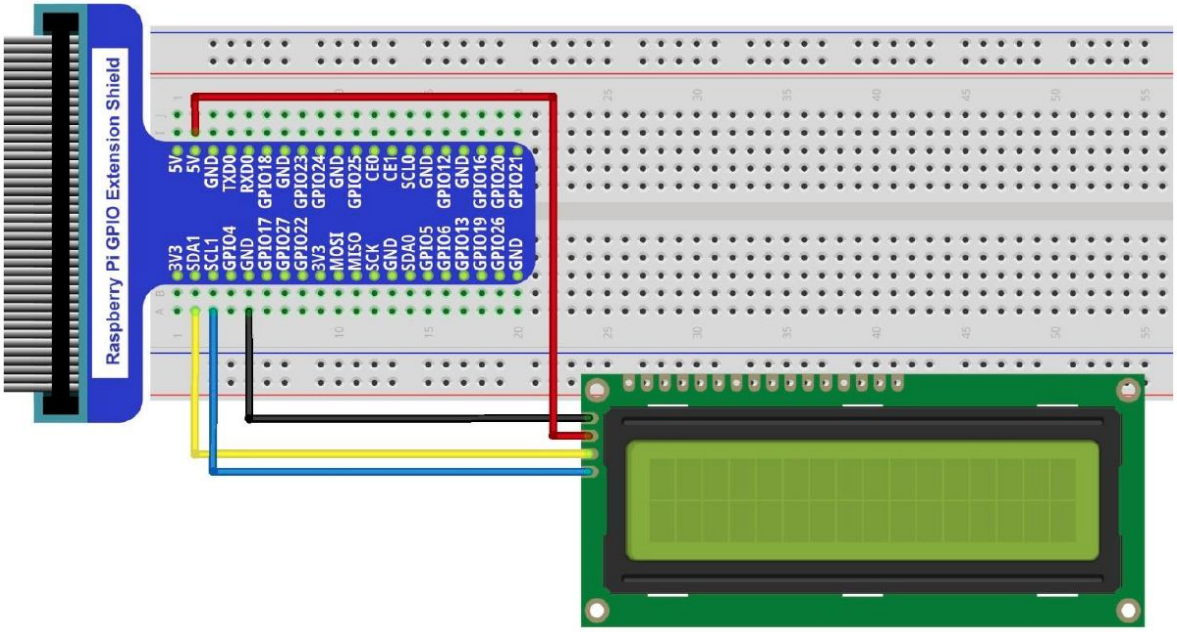


2. LCD1602

Scheme:

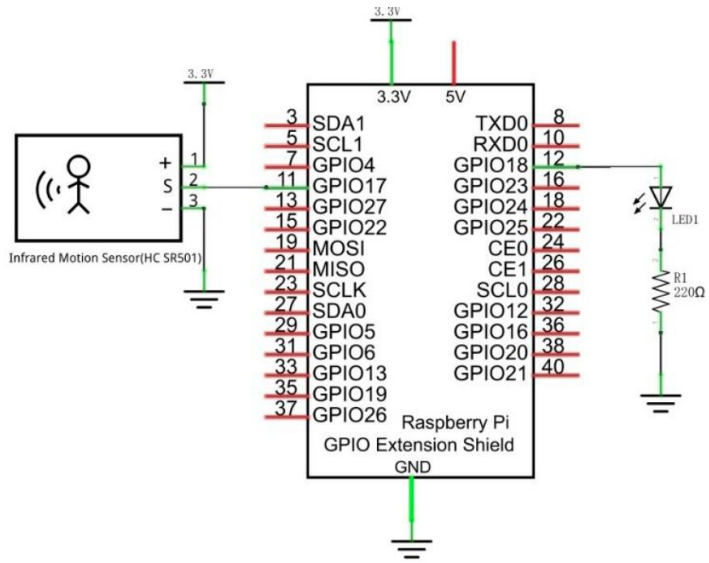


On board:

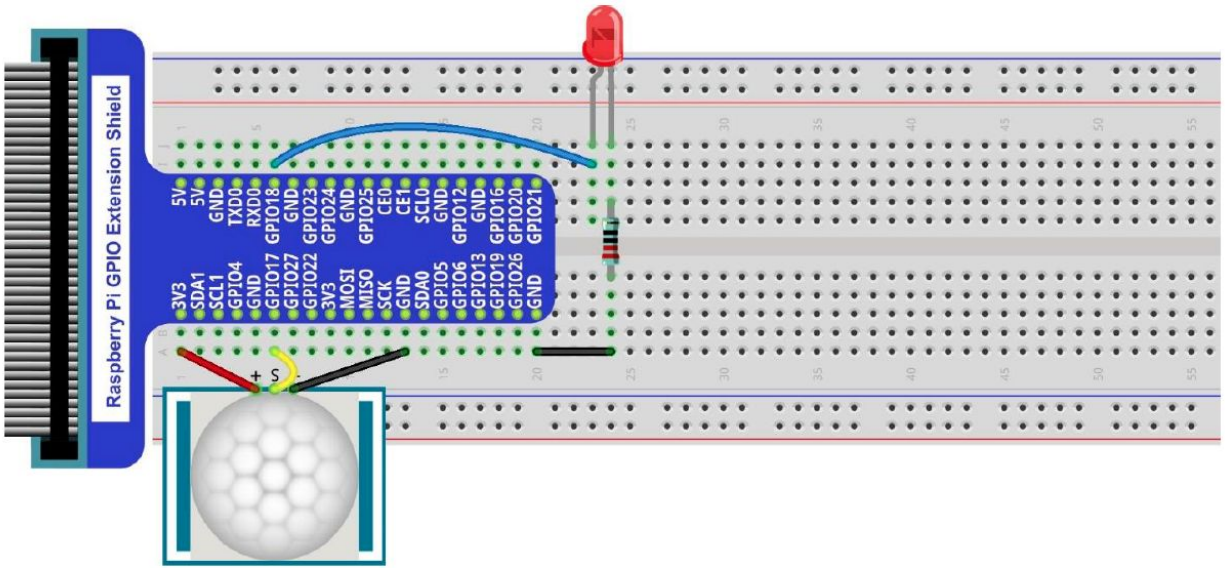


3. Infrared Motion Sensor

Scheme:

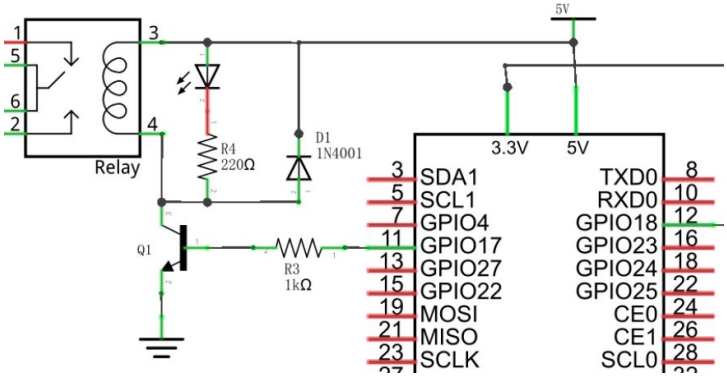


On board:

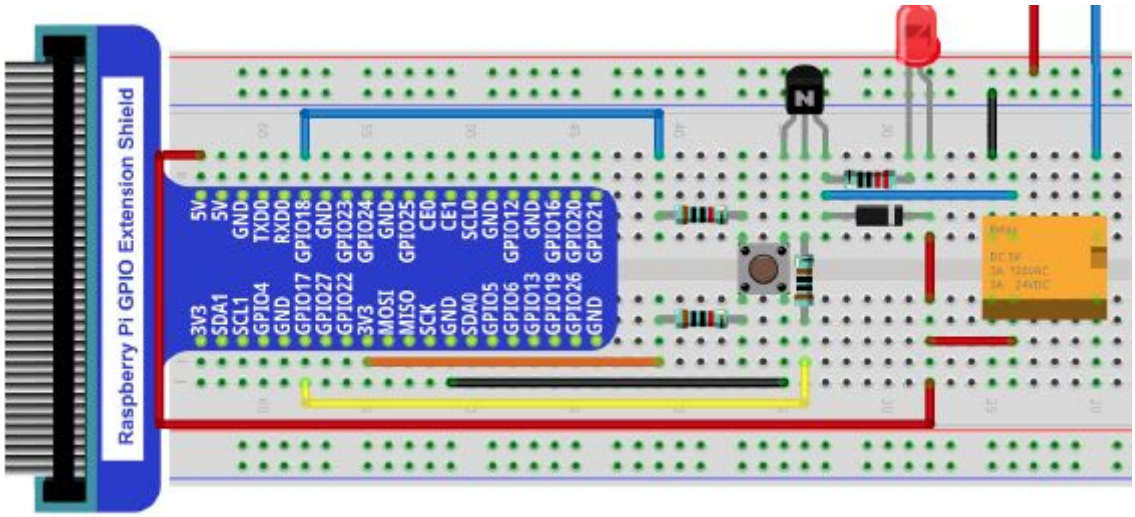


4. Relay

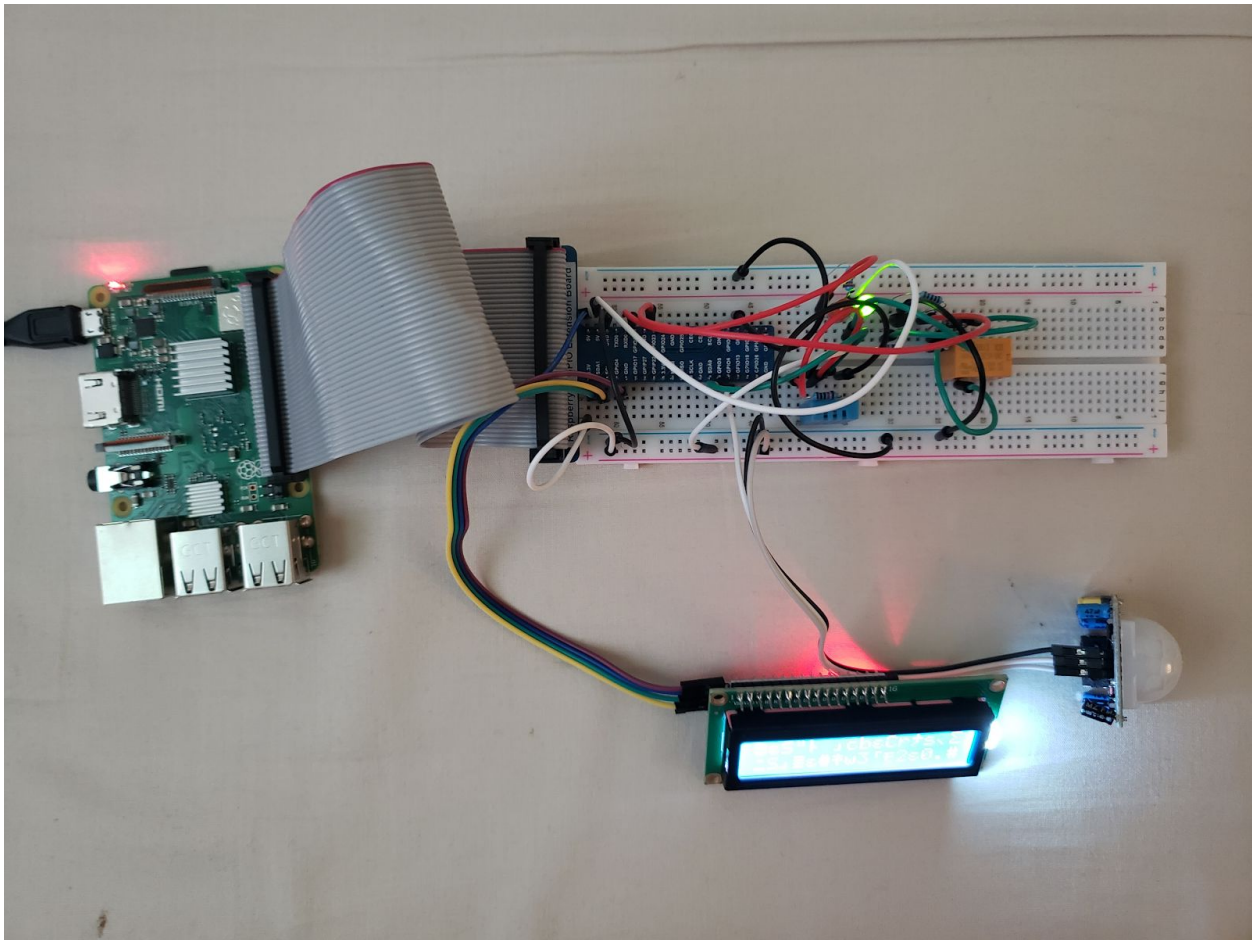
Scheme:



On board:



Final board:



(Final connection of different modules to the pins may vary from what was shown on the scheme)

*All the scheme and on-board pictures come from Freenove's tutorial

Part 2: Codes and Implementations

Overall project hierarchy:

- Adafruit_LCD1602.py
- CIMIS.py
- DHT.py
- final.py
- Freenove_DHT.py
- LCD.py
- MotionSensor.py
- PCF8574.py
- setup.py
- switch.py
- TimerTask.py

1. Hygrothermograph DHT11 (by Zehua Wang)

Code:

DHT.py

```
import RPi.GPIO as GPIO
import time
import datetime
import Freenove_DHT as DHT


DHTPin = 29      #define the pin of DHT11

MAX_READ_RETRIES=20

def get_cur_hour():
    return datetime.datetime.now().hour

dht = DHT.DHT(DHTPin)
def get_dht():
    chk = dht.readDHT11()# try to get tmp and hum
    #try for 10 times else print error
    for i in range(10):
        chk = dht.readDHT11()
        #read DHT11 and get a return value.
        #Then determine whether data read is normal according to the return value.
        if (chk is dht.DHTLIB_OK):

            print("Local: Temperature:%.2f, Humidity:%.2f\n\n"%(dht.temperature*1.8
            return (dht.temperature*1.8+32,dht.humidity)
        time.sleep(1)
    print("Read Local Tmp and Hum Failed!!!!!")
    return None


if __name__ == '__main__':
    while True:
        get_dht()
        time.sleep(1)
```

Freenove_DHT.py

```python
#!/usr/bin/env python3
##############################################################################
# Filename    : Freenove_DHT.py
# Description : DHT Temperature & Humidity Sensor library for Raspberry
# Author      : freenove
# modification: 2018/08/03
######################################################################
import RPi.GPIO as GPIO
import time

class DHT(object):
    DHTLIB_OK = 0
    DHTLIB_ERROR_CHECKSUM = -1
    DHTLIB_ERROR_TIMEOUT = -2
    DHTLIB_INVALID_VALUE = -999

    DHTLIB_DHT11_WAKEUP = 0.020#0.018       #18ms
    DHTLIB_TIMEOUT = 0.0001         #100us

    humidity = 0
    temperature = 0

    def __init__(self,pin):
        self.pin = pin
        self.bits = [0,0,0,0,0]
        GPIO.setmode(GPIO.BOARD)
    #Read DHT sensor, store the original data in bits[]
    def readSensor(self,pin,wakeupDelay):
        mask = 0x80
        idx = 0
        self.bits = [0,0,0,0,0]
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin,GPIO.LOW)
        time.sleep(wakeupDelay)
        GPIO.output(pin,GPIO.HIGH)
        #time.sleep(40*0.000001)
        GPIO.setup(pin,GPIO.IN)

        loopCnt = self.DHTLIB_TIMEOUT
        t = time.time()
        while(GPIO.input(pin) == GPIO.LOW):
            if((time.time() - t) > loopCnt):
                #print ("Echo LOW")
                return self.DHTLIB_ERROR_TIMEOUT
        t = time.time()
        while(GPIO.input(pin) == GPIO.HIGH):
            if((time.time() - t) > loopCnt):
                #print ("Echo HIGH")
                return self.DHTLIB_ERROR_TIMEOUT
        for i in range(0,40,1):
            t = time.time()
```

2. Infrared Motion Sensor (by Zehua Wang)

Code:

MotionSensor.py

```python
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)

_motion_in=31
_motion_out=12

#set in pin and out pin
GPIO.setup(_motion_in,GPIO.IN)
GPIO.setup(_motion_out,GPIO.OUT)

#check if there is a person
def check_people():
    return GPIO.input(_motion_in)==GPIO.HIGH

#turn on the motion sensored light
def light_motion_light():
    GPIO.output(_motion_out,True)

#turn off the motion sensored light
def dark_motion_light():
    GPIO.output(_motion_out,False)

dark_motion_light()

if __name__=='__main__':
    while True:
        if(GPIO.input(_motion_in)==GPIO.HIGH):
            light_motion_light()
        else:
            dark_motion_light()
        time.sleep(1)
```

Presentation:
Youtube link:
https://youtu.be/0teTVGhwSGM

3. LCD1602 (by Yuting Jiang)

Files hierarchy:

Adafruit_LCD1602.py

LCD.py

PCF8574.py

Adafruit_LCD1602.py and PCF8574.py are the files needed to run the LCD module. LCD.py is the code we encapsulate as a package.

Codes in LCD.py:

```python
#!/usr/bin/env python3

from PCF8574 import PCF8574_GPIO
from Adafruit_LCD1602 import Adafruit_CharLCD
from time import sleep
import _thread


PCF8574_address = 0x27  # I2C address of the PCF8574 chip.
PCF8574A_address = 0x3F  # I2C address of the PCF8574A chip.
# Create PCF8574 GPIO adapter.
class LCD(object):
    MAX_FIXED_MSG_LEN=16
    def __init__(self): # initialization of LCDmodule
        try:
            self.mcp = PCF8574_GPIO(PCF8574_address)
        except:
            try:
                self.mcp = PCF8574_GPIO(PCF8574A_address)
            except:
                print ('I2C Address Error!')
                exit(1)
        self.lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2,pins_db=[4,5,6,7], GPIO=self.mcp) #create a object from Adafruit_LCD.py
        self.mcp.output(3,1)
        self.lcd.begin(16,2)
        self.max_msg_len=0
        self.row1=None
        self.row2=None
        _thread.start_new_thread(self._scroll_looper,()) # run on threads to scroll the display

    def destroy(self): # clear the display
        self.lcd.clear()

    def _scroll_looper(self,*argc): #
        while True:
            if(self.max_msg_len>self.MAX_FIXED_MSG_LEN):self.lcd.DisplayLeft()
            sleep(0.5)

    def display_row1(self,msg='Default message'): # display scrolling message on line 1
        # self.lcd.clear()
        self.lcd.setCursor(0,0)
        self.row1=msg
        self.lcd.message(msg)
        self.max_msg_len=max(len(msg),self.max_msg_len)

    def display_row2(self,msg='Default scrolling message'): # display scrolling message on line 2
        # self.lcd.clear()
        self.lcd.setCursor(0,1)
        self.row2=msg
        self.lcd.message(msg)
        self.max_msg_len=max(len(msg),self.max_msg_len)

    def update_row1(self,msg='Default message'): # update the message in line 1 if the data is updated
        if msg!=self.row1:self.display_row1(msg)
    def update_row2(self,msg='Default scrolling message'): # update the message in line 2 if the data is updated
        if msg!=self.row2:self.display_row2(msg)
```

4. Relay (by Yuting Jiang)

Codes in switch.py:

```
import RPi.GPIO as GPIO

_relay_pin=16 # set the relay pin to 16
GPIO.setup(_relay_pin,GPIO.OUT) # set the mode of relay pin to output
switch_on=False # the relay is off by default
GPIO.output(_relay_pin,False) # the pin is set to low by default

def get_switch_status(): # get the current status of the switch (On/Off)
    global switch_on
    return switch_on

def turn_on(): # turn on the switch by setting the pin to high, change the flag to true
    global switch_on
    if switch_on:return
    switch_on=True
    GPIO.output(_relay_pin,True)

def turn_off(): # turn off the switch by setting the pin to low, change the flag to false
    global switch_on
    if not switch_on:return
    switch_on=False
    GPIO.output(_relay_pin,False)
```

5. Online Data Collecting and Main function (by Zhifang Zeng)

Get online data:

To get the data on CIMIS, we:

1. Try to get data of current hour
2. If we fail 1. We try to use the data of the nearest hour
3. If we fail 2. We try to use the data of yesterday at the same hour

```python
def _get_CIMIS_data(date):
    assert(isinstance(date,datetime.date))
    info={'appKey':'d200ee5b-f6b2-47ee-af4b-000c178221e1',
        'startDate':date,'endDate':date}
    # retrive the json string with the nessary info
    r=requests.post(url.format(**info))
    try:
        jsonDict=json.loads(r.text)
    except:
        return None #in case the code fail to attain online data
    data_raw=jsonDict['Data']['Providers'][0]['Records']

    data={}
    for item in data_raw:
        data[int(item['Hour'])//100]={AirTmp:None if item['HlyAirTmp']['Value'] is \
            None else float(item['HlyAirTmp']['Value']),
        RelHum:None if item['HlyRelHum']['Value'] is \
            None else float(item['HlyRelHum']['Value']),
        Eto:None if item['HlyEto']['Value'] is \
            None else float(item['HlyEto']['Value'])}

    return data
```

```
data_yesterday=_get_CIMIS_data(yesterday) #save the data of yesterday
data_today=_get_CIMIS_data(today)

MAX_CIMIS_RETRY_TIMES=5 #Retries of trying to access CIMIS data

def get_data():
    global today, yesterday
    today=datetime.date.today()
    yesterday=datetime.date.today()-datetime.timedelta(1)
    CurHour=datetime.datetime.now().hour

    #try to retrive data from CIMIS with MAX_CIMIS_RETRY_TIMES retries
    data=None
    for i in range(MAX_CIMIS_RETRY_TIMES):
        data=_get_CIMIS_data(today)
        if data is not None:break
    global data_today
    if data is None:
        print('Fail to attain data from CIMIS')
        data=data_today #if fail to get data use the one retrived last time
    else: data_today=data

    #try to get the data of the nearest hour
    #if fail to get it use the data of yesterday of the same hour
    item=None
    for i in range(CurHour-1,0,-1):
        if data[i][AirTmp] is not None and\
        data[i][RelHum] is not None and\
        data[i][Eto] is not None:
            item=data[i]
    if item is None: item=data_yesterday[CurHour] #if fail to get
    return item
```

Main function:

Amount of Water for Current Hour:
First, we need to calculate the amount of water needed for today by the local Et0. Then, we will record the amount of water that has been watered, after that, we set 24 hours as a round, the amount of water needed for this hour will be the calculated water for day of current Et0 subtract the amount of the water that has been water divided by the left hours, where left hours is 24 subtract how many hours have been passed since the program started.

```
#the function to get require water amount of the day by given Et0
def get_require_water(eto):
    return (eto*PF*SF*0.62)/IE

#get the amount of water needed for this hour
def get_water_for_hour(eto,already_water):
    water_for_day=(eto*PF*SF*0.62)/IE-already_water
    water_for_hour=water_for_day/left_hour
    return water_for_hour
```

How to calculate Et0:
The local Et0 is calculated based on this example:

HUMIDITY	TEMPERATURE
70	60
71	61
72	62

Irvine - South Coast Valleys - Station 75

Date	Hour (PST)	ETo (in)	Precip (in)	Sol Rad (Ly/day)	Vap Pres (mBars)	Air Temp (°F)	Rel Hum (%)	Dew Point (°F)	Wind Speed (mph)	Wind Dir (0-360)	Soil Temp (°F)
	0900	0.01	0.00	654	14.2	61.3	76	53.9	3.9	241	68.1
	1000	0.02	0.00	1468	14.3	63.8	71	54.1	3.9	273	68.1
	1100	0.03	0.00	2022	14.6	66.8	65	54.7	4.5	314	68.1

You will calculate three ETo locals based on the temp and hum and the ETo the station has reported:

$ETo_{local1} = 0.01 * 60/61.3 * 76/70$
$ETo_{local2} = 0.02 * 61/63.8 * 71/71$
$ETo_{local3} = 0.03 * 62/66.8 * 65/72$

Where Local_Et0=CIMIS_Et0*(Local_tmp/CIMIS_tmp)*(CIMIS_hum/Local_hum)

```
#function that updates Et0 hourly
def update_eto(*argc, **argv):
    
    global hour_id
    hour_id+=1
    update_local_th()
    CIMIS_data=CIMIS.get_data()#get data from CIMIS
    prev_hour_id=hour_id-1
    local_tmp,local_hum=avg_data[prev_hour_id]
    rate1,rate2=local_tmp/CIMIS_data[CIMIS.AirTmp],CIMIS_data[CIMIS.RelHum]/local_hu
    CIMIS_data[CIMIS.Eto]=0.2#temp
    local_eto=CIMIS_data[CIMIS.Eto]*rate1*rate2#get local Et0
    local_hly_eto[prev_hour_id]=local_eto
    #update msg on LCD
    update_msg2(CIMIS_data[CIMIS.AirTmp],CIMIS_data[CIMIS.RelHum],\
        CIMIS_data[CIMIS.Eto],local_eto,\
        get_require_water(CIMIS_data[CIMIS.Eto])-get_require_water(local_eto))
    print("CIMIS: tmp:%.1f, hum:%.1f, eto:%.1f\nlocal_eto:%.1f water_saving:%.1f"%\
        (CIMIS_data[CIMIS.AirTmp],CIMIS_data[CIMIS.RelHum],CIMIS_data[CIMIS.Eto],
        local_eto,get_require_water(CIMIS_data[CIMIS.Eto])-get_require_water(local_e

    global already_water,left_hour
    left_hour-=1
    water_for_hour=get_water_for_hour(local_eto,already_water)
    already_water+=water_for_hour#update already water amount
    start_watering(get_water_seconds(water_for_hour))#start watering
    if left_hour<=0:
        left_hour=24
        already_water=0

if __name__=='__main__':
    #update tmp and hum and record it every minute
    TimerTask.create_job_per_minute(update_local_data,1)
    #update eto and start watering every hour
    TimerTask.create_job_per_hour(update_eto,1)
    while True:
        #update msg on LCD every 5 seconds
        LCD.display_row1(msg1)
        if msg2 is not None:LCD.display_row2(msg2)
        time.sleep(5)
```