EECS 113: Final Project

By,

Raiyan Nasim

ID: 69632419

Date: 06/12/2020

Objective:

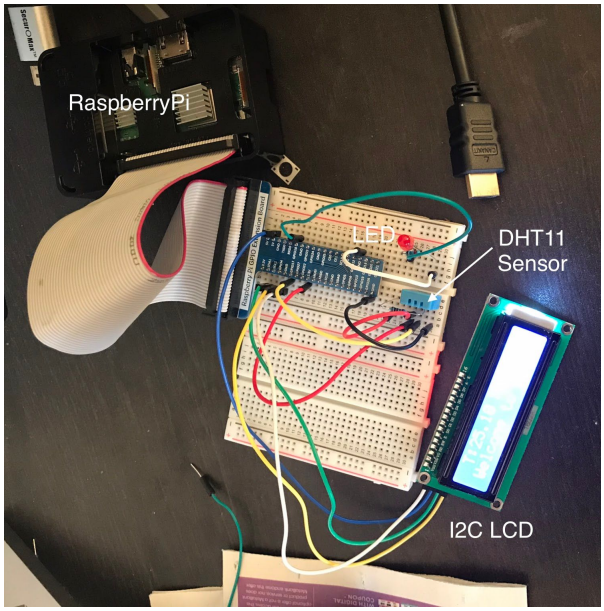
In this project we are going to build and program an atmosphere monitoring system that uses I2C LCD to display local and CIMIS data gathered. It calculates the amount of water and the duration of the water pump to stay on every hour based on an adjusted ET (Evapotranspiration) that compares the local average temperature and humidity recorded every minute and the CIMIS ET for that specific hour. This helps farmers save water when it comes to irrigating the soil.

Hardware Setup:

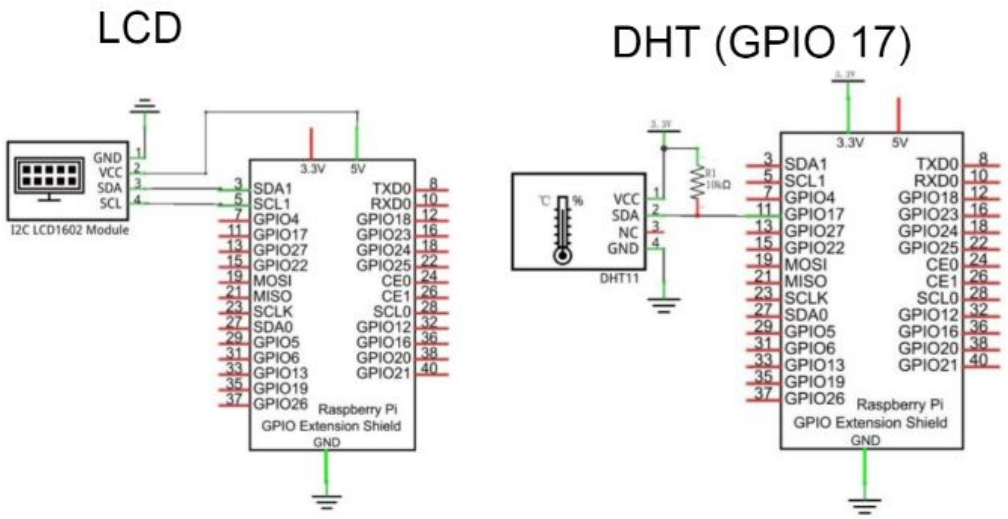
---

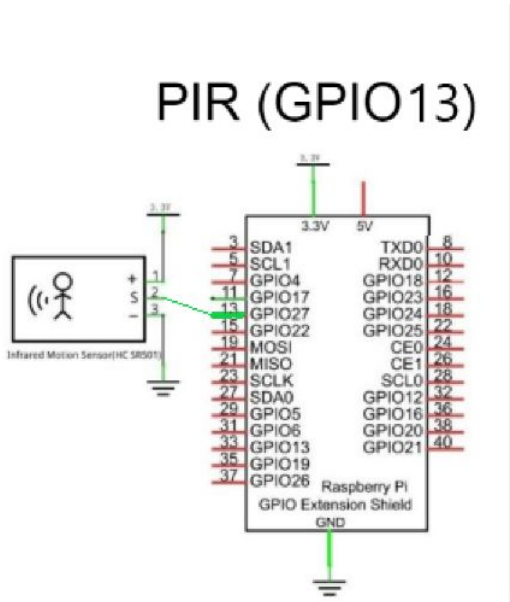
1. Raspberry pi 3
2. Temperature and humidity Sensor
3. PIR Sensor
4. I2C 1602A LCD
5. Raspberry pi GPIO Extension Board
6. LED
7. 10 KΩ and 220Ω resistor
8. Jumper wires
9. Bread board

Circuit Setup:



Pin Connection:





And the LED is in series with a 220Ω resistor connected to GPIO pin 12.

Main Module:(main.py)

The main module that runs all my code is main.py. I run two threads here, one controls the operation my program does every minute called loop() and the other controls the operations for every hour called hour_calculation_thread(). I have multiple global variables that I use to share the data between the threads. I use the Freenove_DHT, PCF8574 and Adafruit_LCD1602 library to control the DHT11 sensor and the LCD

- **CIMIS_TEMPERATURE, CIMIS_HUMIDITY, CIMIS_ETO:** global variables that stores the CIMIS temperature, humidity and ET0 for the hour after I call get_CIMIS_DATA()
- **PUMP_STATUS:** Global variable that keeps track of the pump. if PUMP_STATUS is 0 water pump is OFF and if PUMP_STATUS is 1 water pump is ON
- **LAST_HUMIDITY:** This is the global variable that keeps track of the previous local humidity found by the DHT11. In my test I was reading humidity values above 100 this variable overrides the faulty humidity reading with the last usable humidity recorded. This allows me to get a more accurate average for the local humidity every hour.
- **CURRENT_AVG_TEMP, CURRENT_AVG_HUMIDITY:** These are the global variables that shares the local average temperature and humidity values between loop() and hour_calculation_thread() so I can use them for my hourly calculations
- **CURRENT_HOUR, LAST_HOUR: CURRENT_HOUR:** is keeping track of how many hours has passed and LAST_HOUR is keeping track of what the number of the last hour.
- **PRINTING_HOURLY_STATS:** This helps me prevent my loop() from writing to the LCD when hour_calculation_thread() is writing to the thread also.

I have five functions I use in my implementation. get_CIMIS_DAT(), hour_calculation_thread() and loop() are the main functions that run my code. Below are the description of all the functions I use.

- **get_CIMIS_DATA():**
    - This function gets my CIMIS data every hour from the Irvine station. I call this function every hour to update CIMIS_TEMPERATURE, CIMIS_HUMIDITY and CIMIS_ETO global variables.

```
57  def get_CIMIS_DATA():
58      global CIMIS_ETO
59      global CIMIS_HUMIDITY
60      global CIMIS_TEMPERATURE
61
62      ftp = urllib.request.urlopen("ftp://ftpcimis.water.ca.gov/pub2/hourly/hourly075.csv")
63      csv_file = csv.reader(codecs.iterdecode(ftp,'utf-8'))
64      for line in reversed(list(csv_file)):
65          if (line[4] != "--" and line[14] != "--" and line[22] != "--"):
66              CIMIS_ETO = line[4]
67              CIMIS_HUMIDITY = line[14]
68              CIMIS_TEMPERATURE = line[22]
69              break;
70
```

- **get_local_temp():**
    - This function gets the reading from DHT11 and returns the value of dht.temperature. It also checks if the temp reading is valid or not by running it through the DHTLIB_OK module in the DHT class.
- **get_local_humidity():**
    - This function also does the same sequence of operations as the get_local_temp() but it has an additional guard for negative humidity readings and readings going over 100%. If it gets a bad reading I return the last valid humidity reading I recorded.

```
87  def get_local_humidity():
88      global LAST_HUMIDITY
89      check = None
90      while(check is not dht.DHTLIB_OK):
91          check = dht.readDHT11()
92      humidity = dht.humidity
93      if(humidity > 0 and humidity < 100):
94          return humidity
95      else:
96          print("READ BAD HUMIDITY DATA. USING THE LAST VALID DATA!")
97          return LAST_HUMIDITY
98
```

- **hour_calculation_thread():**

The hour_calculation_thread() gets activated every hour to print the calculated averages of temperature and humidity, CIMIS ET0, temperature and humidity for the hour. It gets activated by an if statement that determines if an hour has passed. After retrieving the CIMIS data it calculates the adjustment ratio and modifies the ET using the following equations:

$$temperature\ adjustment\ ratio\ =\ local\ average\ temperature/CIMIS\ temperature$$

$$humidity\ adjustment\ ratio\ =\ local\ average\ humidity/CIMIS\ humidity$$

And we use the ratio that is higher to calculate the local ET0 or I like to call it adjusted ET0

$$Local\ ET0\ or\ Adjusted\ ET0\ =\ CIMIS\ ET0/(\ adjustment\ ratio)$$

Then it blocks loop() from printing to the LCD by setting the PRINTING_HOURLY_STATS to True. It runs a for loop to scroll the screen from left to right and prints the adjusted ET0, local averages for the temperature and humidity for the hour. Then the thread sleeps for 0.5 sec and starts to print the CIMIS datas using the same method and scrolling the LCD.

```
154            PRINTING_HOURLY_STATS = True
155            lcd.setCursor(0,0)
156            local_text = "adj ET0:%.2f Avg H:%d \n Local avg T:%.2fC"%(adjustedETO, CURRENT_AVG_HUMIDITY, CURRENT_AVG_TEMP)
157            lcd.message(local_text)
158            for x in range(0, len(local_text)):
159                lcd.DisplayLeft()
160                time.sleep(0.5)# duration of scrolling
161            lcd.clear()
162            time.sleep(0.5)
163            lcd.setCursor(0,0)
164            CIMIS_text = "CIMIS ET0:%.2f CIMIS RH:%d \n CIMIS Avg_Temp:%.2fC"%(CIMIS_ETO, CIMIS_HUMIDITY, CIMIS_TEMPERATURE)
165            lcd.message(CIMIS_text)
166            for x in range(0, len(CIMIS_text)):
167                lcd.DisplayLeft()
168                time.sleep(0.5)# duration of scrolling
169            lcd.clear()
170            PRINTING_HOURLY_STATS = False
171
172            gallons_needed_to_irrigate_no_adj = (CIMIS_ETO * PF * SF * 0.62)/IE
173            gallons_needed_to_irrigate_adj = (adjustedETO * PF * SF * 0.62)/IE
```

After it finishes printing the program calculates the amount of water needed to irrigate with the given constants. Using the following formula:

$$Gallons\ needed\ per\ hour\ =\ (Local\ ET0\ *PF*SF*0.62\ )/(IE*24)$$

$$PF\ =\ Plant\ Factor,\ SF\ =\ Area\ to\ be\ irrigated\ in\ square\ feet,\ IE\ =\ irrigation\ Efficiency$$

It calculates the amount with the CIMIS data called gallons_needed_to_irrigate_no_adj and the amount with the modified ET0 called gallons_needed_to_irrigate_adj. With these values we can calculate how much water we are saving or wasting using our system and prints it out to the LCD display using the code below:

```
192             if (gallons_needed_to_irrigate_no_adj-gallons_needed_to_irrigate_adj)>0:
193                 lcd.setCursor(0,1)
194                 lcd.message("H2O saved: %.2f gal"%(gallons_needed_to_irrigate_no_adj-gallons_needed_to_irrigate_adj))
195             else:
196                 lcd.setCursor(0,1)
197                 lcd.message("H2O lost: %.2f gal"%(gallons_needed_to_irrigate_adj-gallons_needed_to_irrigate_no_adj))
198             time.sleep(5)
199             lcd.clear()
200             PRINTING_HOURLY_STATS = False
201             gallons_needed_per_hr = gallons_needed_to_irrigate_adj/float(24)
202             time_needed_to_irrigate = gallons_needed_per_hr/WATER_DEBIT
```

After we print to the LCD we then calculate the time we need to irrigate the soil using our modified ET0. Given by:

$$TIme\ needed\ =\ Gallons\ needed\ per\ hour\ /\ Water\ debit$$

If the time_needed_to_irrigate is 0 that means we do not have to water the soil for that hour but if it is greater than 0 then we need to turn the pump on for that amount of time.I used the following code to turn the pump on indicated by the LED also being on. Here the LED represents the water pump. After we turn the LED/pump on we sleep the thread for the time_needed_to irrigate. Afterwards we turn the LED off and clear the LCD screen indicating the pump has been turned off.

```
204             if time_needed_to_irrigate == 0:
205                 print("No need to irrigate ET0 is zero for hour %d"%(CURRENT_HOUR))
206             elif time_needed_to_irrigate > 0:
207                 print("Turing pump ON for %.2f min"%(time_needed_to_irrigate*60))
208                 #PRINTING_HOURLY_STATS = True
209                 #lcd.clear()
210                 # Turn pump ON
211                 PUMP_STATUS = 1
212                 lcd.setCursor(0,1)
213                 lcd.message("Pump On:%.2f min."%(time_needed_to_irrigate*60))
214                 GPIO.output(LEDpin, GPIO.HIGH)
215                 sleep(time_needed_to_irrigate)
216                 # Turn Pump OFF
217
218
219                 GPIO.output(LEDpin, GPIO.LOW)
220                 print("---------------Turning Pump off ----------------------\n")
221                 PUMP_STATUS = 0
222                 lcd.clear()
223
224             else:
225                 print("There is problem with time_needed_to_irrigate.\n")
226
```

I have a conditional statement that gets activated every 24 hours to write to a text file the amount of water we irrigated with our modified ET0 and the amount of water we would have irrigated if we used the CIMIS data. The following is the code for that:

```
227         if CURRENT_HOUR == 24:
228             #get # of gallons of water irrigated that day
229             # and total adjusted ET for report
230             current_day+=1
231             f = open("24hrLog.txt", "a")
232             print("Printing Daily report...\n")
233             f.write("DAILY REPORT FOR THE DAY : \n")
234             f.write("##############################################\n")
235             f.write("Number of gallons of water irrigated today: \n")
236             total_gal_irrigated_today = 0
237             total_adj_ET_today = 0
238             for i in range(0,len(gallons_of_water_irrigated_today),1):
239                 f.write("Hour: %d     adj_ET0: %.2f     CIMIS: %.2f     LOCAL: %.2f\n"%(i+1, gallons_of_water_irrigated_today[i], adj_ETo_list[i], no_adj_ETo_list[i] ))
240                 total_gal_irrigated_today += no_adj_ETo_list[i]
241                 total_adj_ET_today += adj_ETo_list[i]
242             f.write("Total number of gallons irrigated today (CIMIS): %.2f\n"%(total_gal_irrigated_today))
243             f.write("Total number of gallons irrigated today (LOCAL): %.2f \n"%(total_adj_ET_today))
244             f.close()
245             print("Done printing daily report\n")
246             CURRENT_HOUR = 0
247
```

Every 24 hour I reset the hour values and prepare to start the process again for the next day.

- **loop():**

This function drives the main code and prints the local temperature and humidity to the LCD every minute. It runs a for loop that counts up to 60 representing 60 minutes and uses the get_local_temp() and get_local_humidity() every minute to get the readings from the DHT11 sensor and prints them to the LCD. I also calculate the time it takes to retrieve all the data and subtract it from 60 sec to get the accurate time to put the thread to sleep. I send the thread to sleep mode using an if condition statement that makes sure it goes to sleep for every 60 iteration of the for loop. The thread exits the for loop every hour to calculate the average temperature and humidity values for the hour. Using LAST_HOUR and CURRENT_HOUR values it lets the hour_calculation_thread() know that an hour has passed. Below is my code for these procedures.

```
for i in range (0,60,1):#change to 60
    time_taken_to_retrieve_data = time.time()
    local_temp = get_local_temp()
    #if get_local_humidity() < 100:
    local_humidity = get_local_humidity()
    LAST_HUMIDITY = local_humidity
    avg_local_temperature += local_temp
    avg_local_humidity += local_humidity
    lcd.setCursor(0,0)  # set cursor position

    print("Data #" + str(i) + ": Temp:" + str(local_temp) + " Humid:" + str(local_humidity) + "\n")
    if(PRINTING_HOURLY_STATS == False):
        lcd.message(" T:%.2f H:%.1f\n"%(local_temp, local_humidity))
        #lcd.message('TEMP:' + str(local_temp) + 'HUM:'+ str(local_humidity)+'\n')
        #lcd.message(get_time_now())
    time_taken_to_retrieve_data = time.time() - time_taken_to_retrieve_data
    print( "Time passed : %.2f min\n"%((time.time()-time_started)/60))
    if i != 59:
        #debug
        #time.sleep(10)
        time.sleep(60 - time_taken_to_retrieve_data) #(60s - time it takes to get the data)sleep for a min
```

```
306        avg_local_temperature = avg_local_temperature/60 #change to 60
307        CURRENT_AVG_TEMP = avg_local_temperature
308        avg_local_humidity = avg_local_humidity/60 #change to 60
309        CURRENT_AVG_HUMIDITY = avg_local_humidity
310        LAST_HOUR = CURRENT_HOUR
311        CURRENT_HOUR += 1
312
```

Conclusion:

After I ran the program for 24 hours I compared the amount of water that would have been irrigated if I used the CIMIS data and amount of water we used by using our local values. I found that we saved 0.53 gallons of water every day. And 6.35 gallons of water every year. This means that my system works and it not only saves the farmers water but also protects our environment by not wasting it.