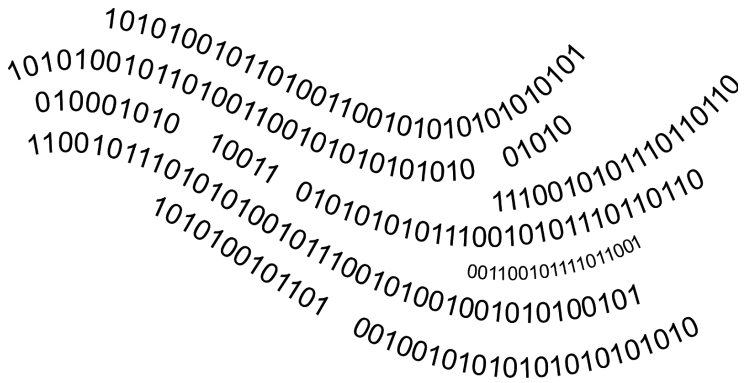
EECS 113 Final Project

CIMIS Weather Monitoring System



Arshdeep Dhillon :: 47797712

Aakif Hussaini :: 63828630

Prannay Kapur :: 45621364

15 June 2019

I. Source Code Description

A. Main Program

The main code, defined by the python file main.py, is the driver file for the weather monitoring system. This file stitches the python modules that define the peripherals in such a way that any weather based information will be processed and displayed on the LCD, and, depending on the time of day, will activate a sprinkler system to irrigate a plot of lawn.

The processes defined in the run function are being constantly looped to continuously run the weather monitoring system. The loop has 3 states defined by two flags:

- constantly processing and displaying data when the irrigation system is off
- displaying existing data while the irrigation is on, and listening for any movement
- when movement is detected, updating the display to reflect the detection and pausing irrigation until some period of time has passed

B. CIMIS Data Extraction

The CIMIS Data Extraction, defined by the python file cimis.py, sends a request to the CIMIS database to parse the data that will be used in the irrigation system and the LCD display. It does this by sending an HTTP GET request to the CIMIS server for three parameters: ‘HlyEto’, ‘HlyAirTmp’, and ‘HlyRelHum’ (ETo,

temperature, and humidity). If these parameters exist, they are sent back to the calling function; otherwise, some default "wrong" values are returned.

C. LCD Display

The LCD Display, defined by the python file lcd.py, only has the setup module imported to the main. In the setup, the lcd object is instantiated, and the port extension PCF8574 is defined to an I2C address. The LCD call occurs in the main any time information needs to be displayed.

i. Adafruit LCD and PCF8574

To have the LCD run only two pins using the serial ports on the Raspberry Pi,we utilize the PCF8574 extension. The Adafruit LCD defines specific functions to make writing to the LCD easier.

D. DHT11 Weather Monitoring

The DHT11 Weather Monitoring is defined by the python file DTH11.py . It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). We are using it to get the temperature It calculates relative humidity by measuring the electrical resistance between two electrodes. In our project, we are sensing temperature and humidity every minute and generate hourly averages of both quantities and display it on the LCD.

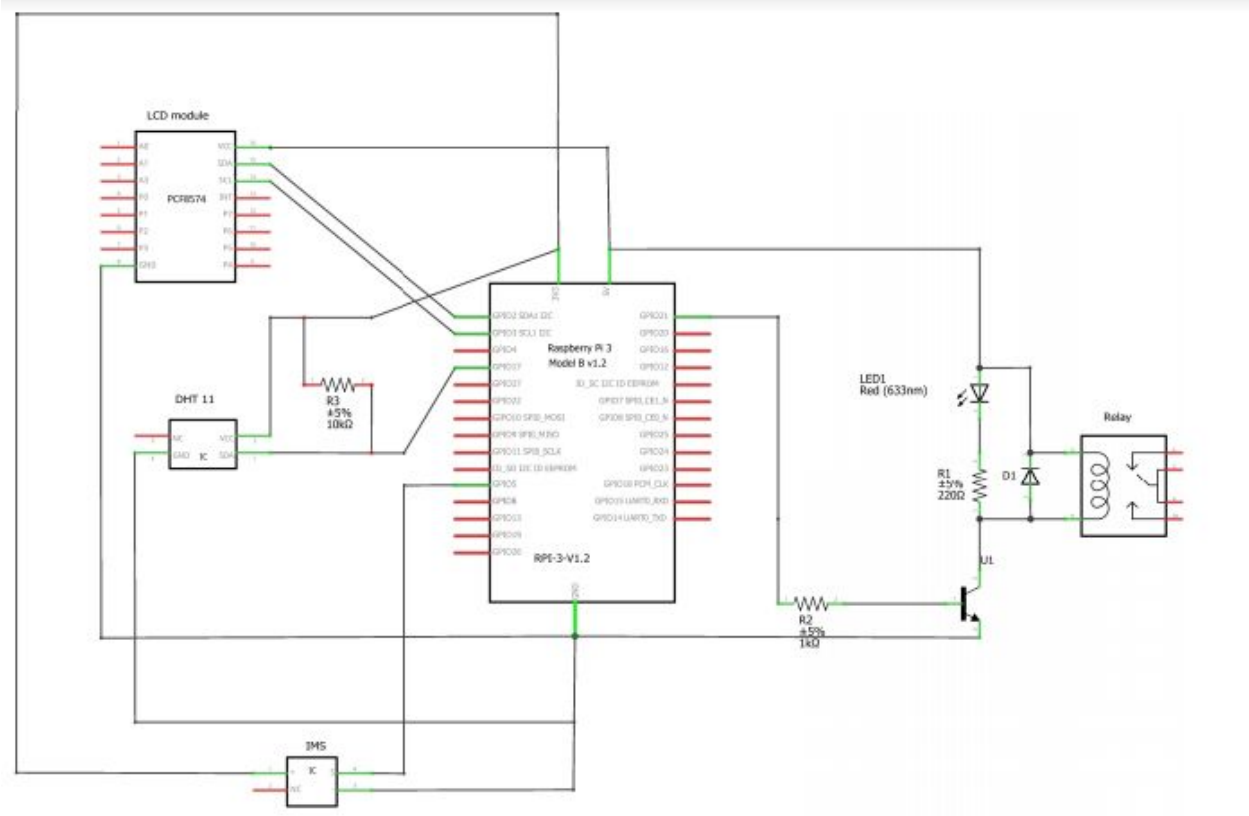
i. Freenove_DHT

This file defines specific functions to make reading from the DHT11 sensor possible.

E. PIR Motion Sensor

The PIR Motion sensor is defined by the python file PIR.py . PIR stands for passive infrared. This motion sensor consists of a fresnel lens, an infrared detector, and supporting detection circuitry. The lens on the sensor focuses any infrared radiation present around it toward the infrared detector. Our bodies generate infrared heat, and as a result, this heat is picked up by the motion sensor. In our project, we are using PIR motion sensors to stop the sprinklers once someone passes by to avoid wetting him.

II. Circuit Schematic and Description



A. LCD Peripheral

The LCD (LCD1602) is connected to the Raspberry Pi via serial to parallel chip PCF8574 (PCF8574A), this enables us to drive the LCD using only two GPIO pins as can be seen in the schematic (SDA1, SCL1).

B. Relay Peripheral

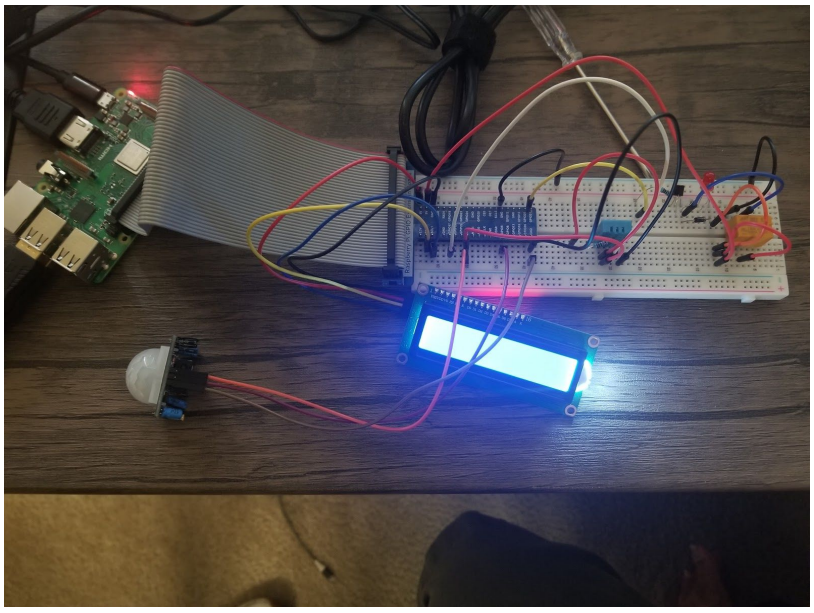
The relay is used to drive external circuits, here LED connected to it shows whether it is active or inactive, the relay is driven by GPIO 21.

C. DHT11 Peripheral

The DHT11 measures the temperature and humidity and it is read using GPIO 17.

D. Infrared Motion Sensor Peripheral

The IMS detects movement in its vicinity, its input is read using GPIO 5. When it detects that there is movement it sets the pin to high and reads low otherwise.



III. Potential Complications

Regarding threading of processes in our main function, even though we had multiple peripherals from which we were listening for data, we were unable to utilize threading on the circuit peripherals. Our Raspberry Pi seemed unable to handle the processing, as data collection from the peripherals slowed down significantly, so we polled data from all the modules synchronously, utilizing threading only for the CIMIS data extraction.

Regarding the threading for CIMIS data extraction, there were multiple instances when we were unable to receive any response from the CIMIS server for reasons including the website being down, the JSON object not decoding properly, etc. In these cases, we would request data from the CIMIS server for multiple times, only quitting after a certain amount of time had elapsed and going on to process other information.

Regarding the calculation of hourly ETo, temperature, and humidity values from the CIMIS server, many times the server would not update for the hour which we were requesting the aforementioned data, so to resolve these issues, we instead would return an average of the data from the past 3 or 4 hours (depending on the frequency of data updating on CIMIS).

Regarding the duration of the irrigation system, we were given conflicting data about when to turn on the irrigation system. In a *Discussions* thread, one TA said to poll the irrigation system every hour. In an *Announcements* thread, a pdf was provided which implied to poll the irrigation system every three hours. We decided to listen to the TA and poll the irrigation system every hour, and the length of the sprinkler time was determined by the following math:

- assuming the maximum range of a sprinkler = 6 ft
- so 1 sprinkler can cover $2\pi(6\text{ ft}) \approx 38$ ft
- given that we have 1500 square feet to cover, we need $1500/38 \approx 40$ sprinklers
- note that we will calculate the amount of water needed to be sprinkled in the code. call this value TOTAL_WATER
- assuming sprinkler rate of spray = 2.5 gpm (gallons per minute)
- so water_time_duration = (TOTAL_WATER/ 2.5)/40
- note that the assumptions were taken from internet resources