EE 250 Final Project Write Up
Team members:
Michelle Arredondo
Nayeli De Leon

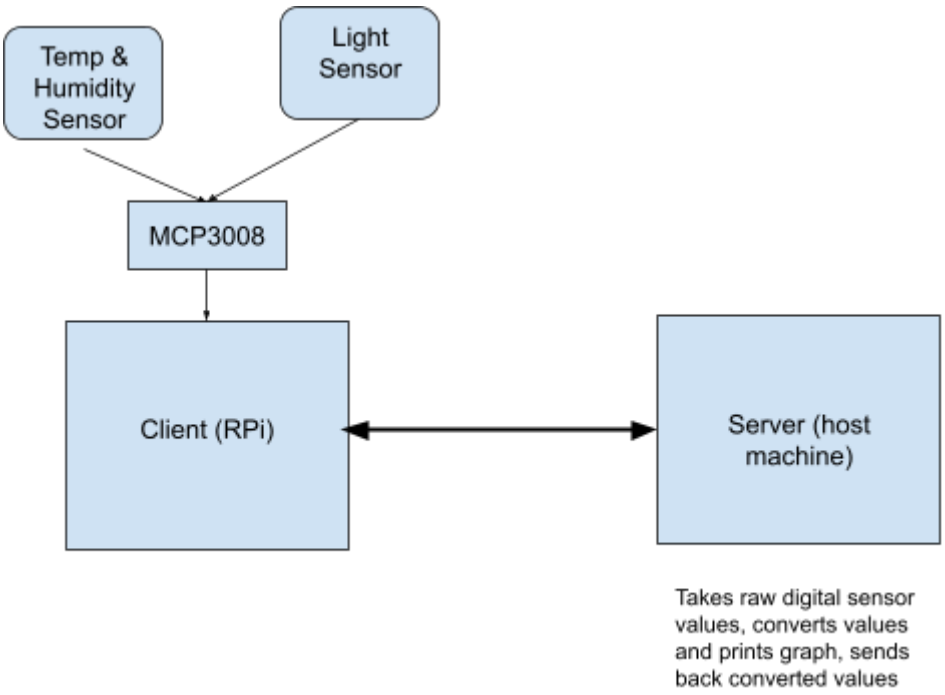
The Plant Monitoring Project:

Our project's intent is to monitor the light, humidity and temperature near a pet plant and keep the owner updated on its habitat status. Sometimes, plants droop and we don't know why: our IoT system can help with that! We created a hardware circuit that connects the Grove temperature/humidity sensor and Grove light sensor to the raspberry pi. The raspberry pi acts as a client and runs the client code in order to read measurements from the sensors. Then, the raspberry pi offloads the raw data measurements to our laptop, which acts as the server, in order to convert the data to appropriate temperature and humidity units. The server returns the converted data. Additionally, the server determines if the environment of the plant is bright or dark based on the raw light measurements. For example, the brightness (light) near the plant changes as the blinds are open or closed: if the light sensor detects it's too dark, it lets the user know the blinds are closed and and the user can know to open the blinds. Not getting enough sunlight can be an indicator of an unhappy plant. The server returns a boolean "True" if bright or "False" if dark. Our client receives the converted data and prints the data in the terminal in dataframe structure. The client also receives the light boolean in order to enable or disable an LED.

Our nodes:

- RPi (client)
- Laptop (server)

Block Diagram of Project:



The libraries used include:

On the client:	On the server:
• Numpy • Pandas • Requests • Time • RPi.GPIO • Adafruit_GPIO.SPI • Adafruit_MCP3008	• Flask • Numpy

Components and Protocols:

We used HTTP protocol to communicate between our raspberry pi and our laptop/computer. In this system, the raspberry pi acts as the client and the laptop acts as the server. Our raspberry pi acts as a power source for our Grove humidity/temperature sensor, Grove light sensor, button, and LED. All sensors connect directly to an MCP3008 analog-to-digital chip, which takes the analog signals from the sensor and converts to a digital signal. The MCP3008 is also connected to our raspberry pi, which receives the digital signal values. We use 'POST' requests in order to send the raw data values from the RPi to the server. The server was created using Flask. The server code contains three app routes, where each app route corresponds to one of the analog sensors. The server code works by receiving the corresponding data and then performing algebraic computations in order to convert the raw data into appropriate units, such as Celsius for temperature and percent-humidity for humidity. For the light measurements, our server code uses an if-statement and threshold value to determine if the outside environment is dark or bright, and returns a boolean to tell the client whether it's dark or bright. The other apps return the converted data to the client code. The client receives all return values from the server, and repeats this process 4 more times to get 5 total samples. The client then prints the mean and standard deviation values in a dataframe structure. The returned boolean determines whether or not an LED receives power (HIGH) or not (LOW). We confirm if the LED is on/off by pressing a button, which is connected to the LED and one of the RPi's output channels.

Reflection and Limitations:

Temperature and humidity data, while processed and converted, are not the most accurate values: the raw values are around ~1023, and once converted to celsius and humidity, the values are ~50 degrees celsius and 100% humidity. Another limitation is that, even though we created a dataframe from our data, we cannot visualize a bar graph because we are running the client code on the RPi. This means we would need to connect the RPi to a monitor if we wanted to directly display the graph. One of our lessons learned is giving ourselves ample time to download certain libraries, and with hotspot (wireless) connection issues, we learned to be flexible and restart the connection when communication lags, but we could alternatively connect the raspberry pi to an ethernet cable. We learned how to incorporate new libraries and synthesize the use of our old labs. We integrated skills we learned from Lab 1 to 10, from flashing to our rpi, push/pulling code from git, to MQTT and reading datasheets for proper RPi/ADC connections.