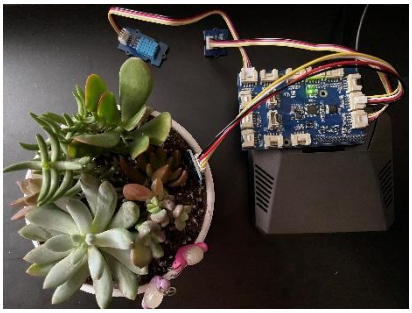
EE 250 Final Project: Plant Monitor

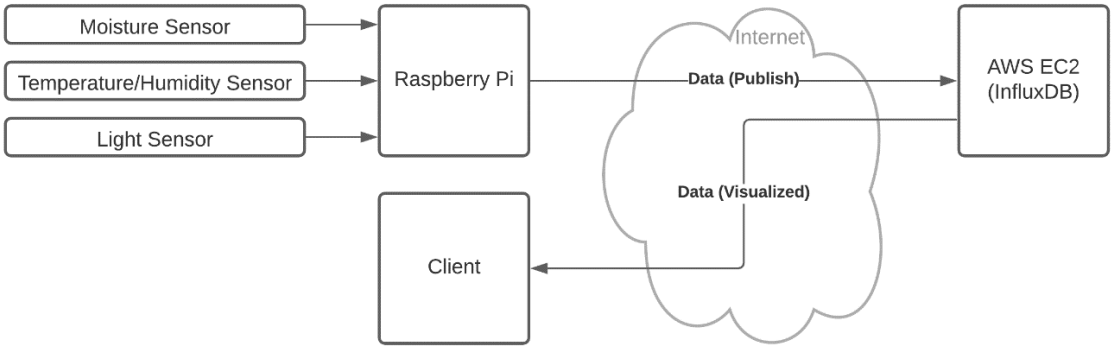
Team Members: Joshua Williams
Github Repository: https://github.com/jwills15/PlantMonitor
Video Demonstration Link: drive.google.com/file/d/1NpRbgDctqdNc5Er7V7WB9srjZDacSF5W/view

Description:

The Plant Monitor system is designed to monitor the health of a plant. The system uses a moisture sensor, a light sensor, a temperature sensor, and a humidity sensor to measure the plant's environment. The data is sent over the internet, stored on a cloud database, and visualized through Grafana. The data visualizations can be accessed from any device with an internet connection. The system also performs data processing and event detection to give clear insights about the health of the plant. The program's settings can be set to a plant's needs, which are fed into an algorithm to determine if the plant's environment is healthy or unhealthy. The system provides real-time insights as well as logged data to improve plant care.



Block Diagram:



Components:

- GrovePi+: system designed for the Raspberry Pi that allows for easy sensor interfacing to collect data from the environment.
- Grove Moisture Sensor: sensor that measures the moisture of the soil that it is in.
- Grove Temperature and Humidity Sensor: sensor that measures the temperature and humidity of its surrounding environment.
- Grove Light Sensor: sensor that measures the light level of its surrounding environment.

Platforms and Nodes:

- Raspberry Pi: A Linux machine that can utilize the GrovePi+ system to interface with sensors. The Raspberry Pi was chosen for implementation because it can interface with sensors and have an internet connection for sending data to a cloud database.
- AWS EC2 Instance: A Linux machine hosted on an AWS EC2 cloud computing instance. AWS EC2 was chosen because it is free, easy to implement, and can provide nearly 100% uptime so that data can always be stored or visualized.
- Linux Virtual Machine (VM): A Linux virtual machine was utilized to develop the code and SSH into the other nodes for implementation. The VM was chosen because it provides a user-friendly interface for developing while allowing SSH connections to operate the other nodes. The VM can also display Grafana on a web browser.

Protocols:

- HTTP: The HTTP protocol is utilized for communications with InfluxDB. The Python script on the Raspberry Pi uses an HTTP POST request to send data to InfluxDB on the AWS EC2 instance. Grafana uses an HTTP GET request to retrieve data from InfluxDB on the AWS EC2 instance. HTTP was chosen for implementation because InfluxDB is set up to be easily configurable with HTTP and HTTP satisfied the network limitations of the system (remaining in the free tier of AWS EC2).
- SSH: The SSH protocol was utilized for connection and command-line operation between nodes. SSH with basic authentication was used to establish an SSH connection from the VM client to the Raspberry Pi. SSH with asymmetric key authentication was used to establish an SSH connection from the VM client to the AWS EC2 instance. SSH was chosen for implementation because it provides simple, secure access to other devices without needing additional hardware.

Data Processing Techniques:

- Time-Domain Low Pass Filter implemented by a Moving Average Filter: A moving average filter averages each sensor reading with the adjacent readings. This process acts as a low-pass filter, smoothing out the variance between different readings. This filter was chosen because testing found that the light and moisture sensors produced sequential readings that differed by several percentage points, so a simple average of the values was not sufficient for producing an accurate result. Instead, a low pass filter is needed to smooth out the values. Then the average can be taken to produce an accurate reading.
- Real-time detection of environment status: For any given plant, it has a desired temperature and humidity range for optimal health. In this system, the healthiness of the plant's environment was determined by a real-time data processing algorithm to determine whether the plant's current environment is within the appropriate bounds set by the plant's settings. The plant's temperature and humidity are graphed and the program checks that the plot is within the bounds of an ellipse. An ellipse was chosen because a plant becomes stressed when it is at one of the extremes of either temperature or humidity, so having both at their respective extreme (like a rectangle plot) would be unhealthy for the plant. The detection output is sent to InfluxDB and Grafana, which indicates "unhealthy" when the plot is outside the ellipse and "healthy" when it is inside the ellipse.

Visualization:

- Grafana: real-time visualization tool for displaying data. Grafana was chosen for implementation because it can easily connect to InfluxDB for data, it has a simple user interface for designing data visualization dashboards, and it runs on an HTTP port that can be accessed by other devices.

Reflection:

Designing and building this system helped me to realize how much I have learned in this class. This system brought together components from Lab 1, 2, 3, and 9, while the design process involved considerations of all other labs. I was amazed at how many technologies I now understood and knew how to use. This project helped me to synthesize all that I have learned and produce a system that combines them.

The current iteration of the plant monitoring system has limitations that can be improved upon. Most importantly, the health detection algorithm can be improved to incorporate the amount of light that the plant is getting and notify if it is too much or too little. Incorporating this would require a knowledge of the plant's requirements that I do not yet have. Furthermore, HTTP has higher overhead that could be a limitation if the network requirements were stricter. Configuring the system to communicate with COAP would reduce this potential limitation.