aPLANT: A Low-Cost Greenhouse Manager System

Dylan McKeever
Ben Neuendorf
Devon Simmons

Advised by: Aleksander Malinowski

Abstract

Changing global climate, increasing food scarcity, and the need for water conservation lends to the value of systems that grow plants quickly and optimally. This project addresses the fiscal aspects of that paradigm. While other systems exist and do as advertised, the cost barrier to entry is too high, sometimes in the range of thousands of dollars. We believe that the hardware used in this project is a key to alleviate the steep, initial, fiscal problem that exists in current monitoring platforms because the ESP8266 is a low power, feature-rich, and an inexpensive chip that can perform similar tasks. ESP8266, which is the computing platform of choice, is inexpensive coming in at about 5$ a module and is very robust as for this price point. The project uses a module and a sensor array to provide temperature, humidity, soil moisture, and control nutrition delivery to plants.

Table of Contents

Table of Figures

I. Statement of the Problem

Most small to medium sized greenhouses have a distinct disadvantage when it comes to the price of plant management devices. While a relatively new concept, plant management on an individual basis has a very valid and measurable impact on the production of crops. Indoor greenhouses are the future of increased yield in limited space. Current devices on the market, require the user to have a sensor per pot to get individual control over a variable in the "growing" equation such as moisture control, ph control, and temperature control. Some variables are easily managed with other techniques ex. temperature control. The immobile elements of growing, such as planting and moisture control, are vastly benefitted by precise control. The modules made by other manufacturers for this purpose are usually separate, and typically don't expand functionality easily to more than one growing surface.

The need for these systems is steadily growing. Changing global climate, growing food scarcity, and needs for water conservation lend to the value of systems that guard against potential problems such as these. The problem chosen to tackle with the project is the fiscal aspect of these systems. While other genuine systems exist and do as advertised, the barrier to entry is too high sometimes in the range of thousands of dollars. The hardware being used is key to solving the fiscal problem because there exists a wide variety of low power, feature-rich, and inexpensive chips that do exactly what is needed. The ESP8266 fits for 2 of those main reasons and a few tertiary ones. The ESP8266 is inexpensive coming in at about 5$ a module and is very robust for this price point with multiple modes and fantastic connectivity. The tertiary reasons are highlighted by an embedded C capability.

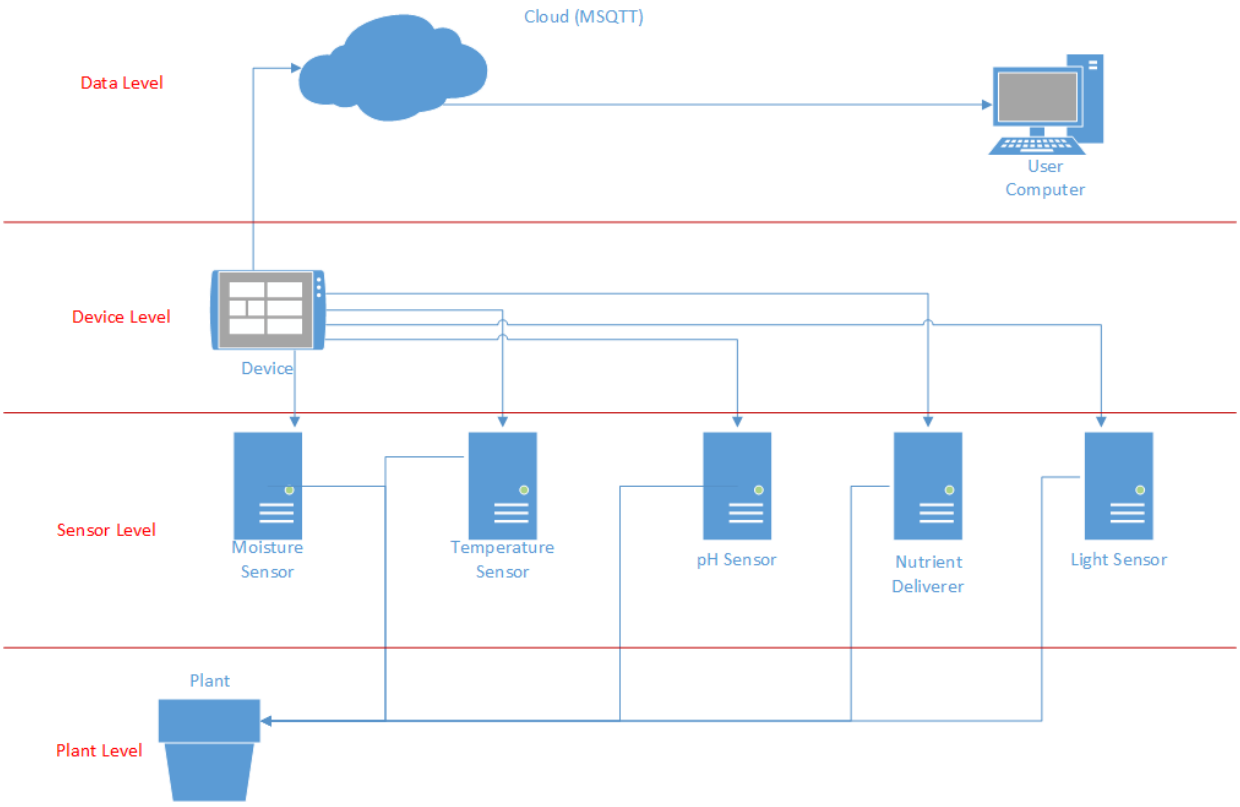

Figure 1: Flow Diagram

II. Functionality

Figure 1: Flow Diagram is not meant to show the physical configurations of the devices most of those will be done at the hardware level using the c environment that is provided by the manufacturer of the ESP8266.

Data Level: The Data Level contains the cloud MQTT (Message Queuing Telemetry Transport) database that has the sensor data arranged into files per sensor type and further divided based on the specific pot from which the data was taken. It will be responsible for outputting the data to the user end to make evaluations from and set up the sensors.

Device Level: The Device Level contains the individual pots arranged by an esp8266 chip designated as the “Controller.” The controller serves as the vehicle for the modularity and maintains connections to the data going to the server and the configuration requirements that the user wants to set. The plan is to have a controller implemented using esp8266 device set solely for this purpose in this functional level.

Sensor Level: The Sensor Level contains all of the sensors and devices to be connected to the plant locally restricted to one esp8266 chip. The different sensors, from moisture to light, will be

localized on one esp8266, but for the interest of showing how they will be initialized and implemented, it is useful to show them as separate entities, as shown in the flow diagram in Figure 1 on page 3.

Plant Level: The Plant Level will only contain the plant itself, the ends of the sensors, and the connection to the esp8266 module that is interpreting the data and sending it upward to the Sensor Level.

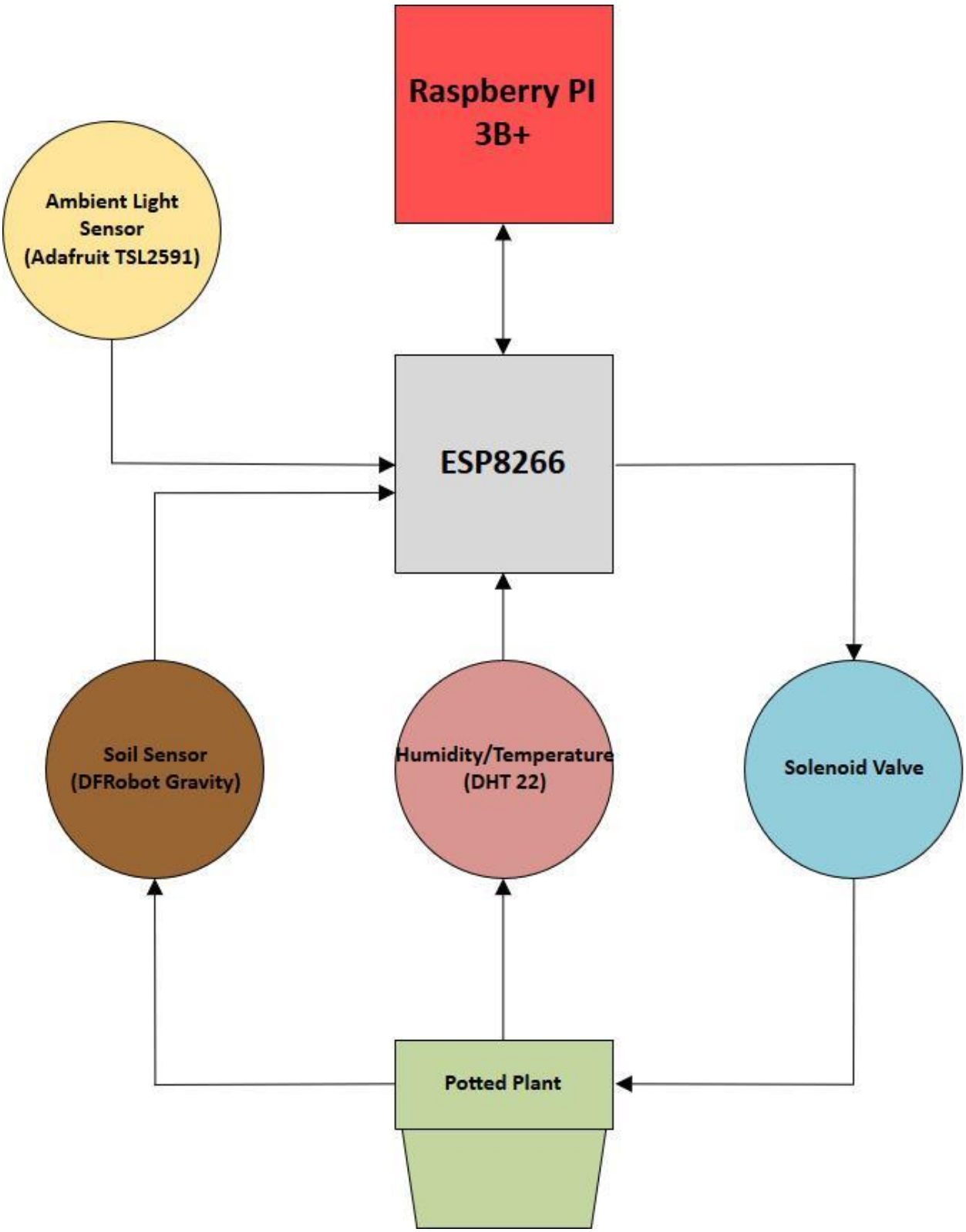

Figure 2: Hardware Schematic

This depiction is of the final system configuration with relation to each other in a clean diagram of core components in the physical space.

A. Hardware

Controllers:

ESP8266: In terms of functionality, this device is the main sensor processor. It is responsible for the direct communication and interpretation of the sensor output. All developments were developed in the Atom IDE using Platform IO. The ESP reports data asynchronously after a short period of the startup where it enables the connections through the MQTT broker to subscribe and post messages on the preconfigured Broker.

Raspberry Pi3b+: The Pi serves as the broker, access point, and the database. It is configured to create the topics and then subscribe to them to facilitate putting the information in the database.

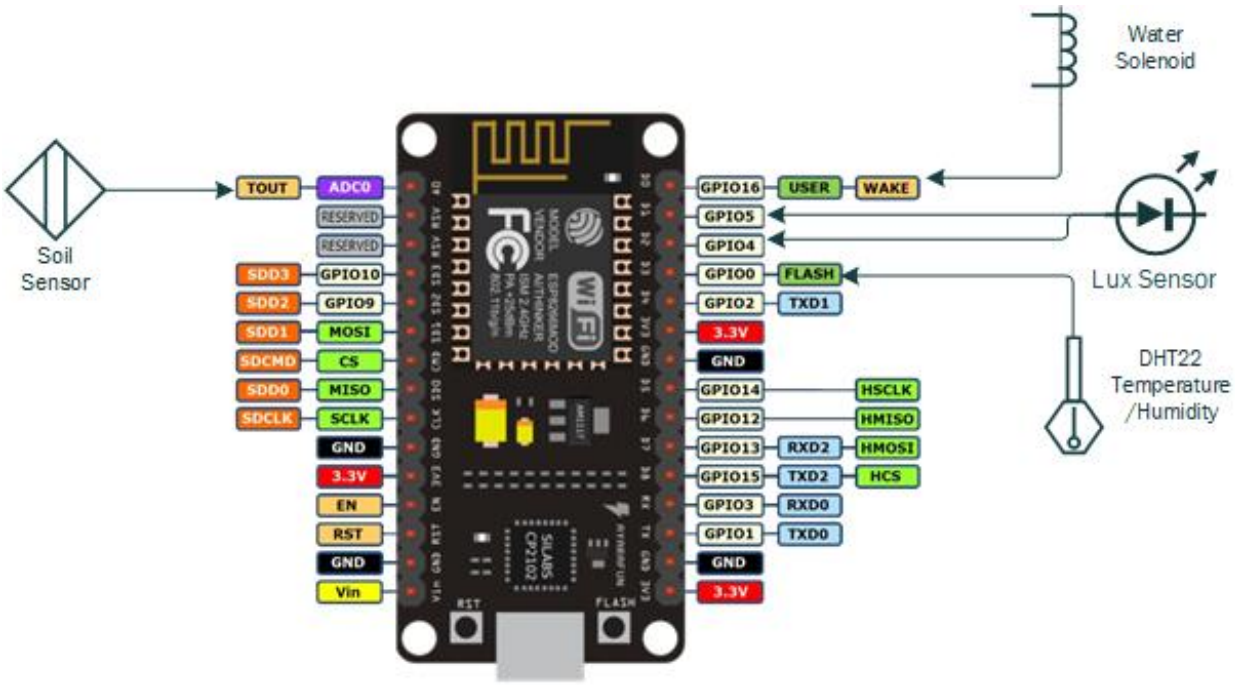

Figure 3: ESP8266 Topology

The ESP8266 has an abundance of IO to take advantage of but in some cases limited use for the pins. The internal watchdog and system commands can borrow certain functionalities for interrupts that will take over certain pins. The soil sensor is connected to the only analog to a digital IO pin. In the program, the analog voltage is taken in via the comparator, and the voltage levels decide how moist or arid the soil is. The water solenoid is connected via the optocoupler to the ESP8266's D0 pin. The digital lux sensor is connected to D1 and D2 both serving to relay information. The DHT22 is connected via its one control line.

Sensors:

The DHT22 is the Humidity and Temperature sensor because of the price point and functionality. It has a substantial body of support in the hobbyist world.

The DFrobot Gravity is a unique capacitive soil sensor in that it was one of the few serial soil sensors that were coated to be resistant to corrosion. The use case for the soil sensor requires that the sensor that can last multiple months in a single deployment

The TSL2591 is one of the few digital lux sensors on the market that has a complete range of values that the typical human eye would be able to process and interpret. It can display this information over an i2c bus.

The water solenoid is the main nutrient delivery system. It operates normally close at 12 VDC. The primary problem using this solenoid is the generation of a proper operating voltage for the solenoid. The power circuit is isolated via an optocoupler.

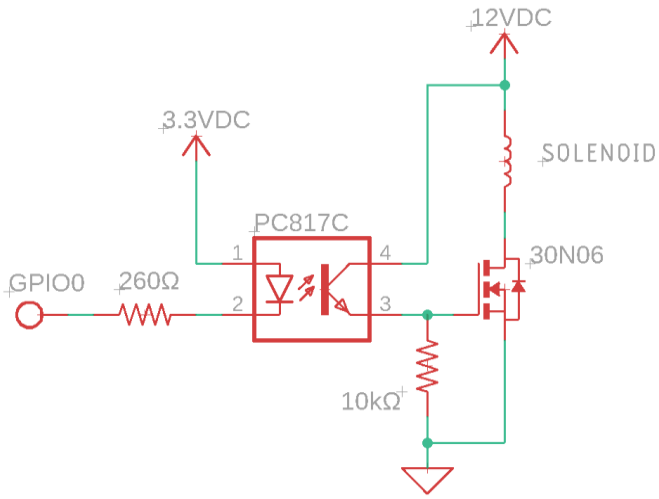

Figure 4: Solenoid Circuit

The solenoid circuit contains the MOSFET that is used to provide the proper on voltage and the optocoupler that isolates the higher voltage solenoid from the 3.3v ESP.

B. Software

Raspberry Pi 3b+:

OS: Raspbian Stretch Lite was chosen for its simple nature and ample support as Raspberry Pi's official supported the operating system. A command line interface is utilized to save space and limit resource usage when operating the pi as the ap. DNSMASQ, HOSTAPD, and IPtables are the backbone of why the Raspberry Pi can serve in a split role as the ap, and the data collection

manager. DNSMASQ is the data transfer manager. It does DNS services as well as DHCP and network control for small scale networks.

Data Acquisition on the Pi level is handled by using Mosquitto as the hub for messages to be sent from the various ESP8266 and the selected sensors that have been connected to each. The esp8266 can self-check the sensors it has and generates the topics over Mosquitto with that information. Once topics are created, a way of disseminating the information to the SQL database was needed.

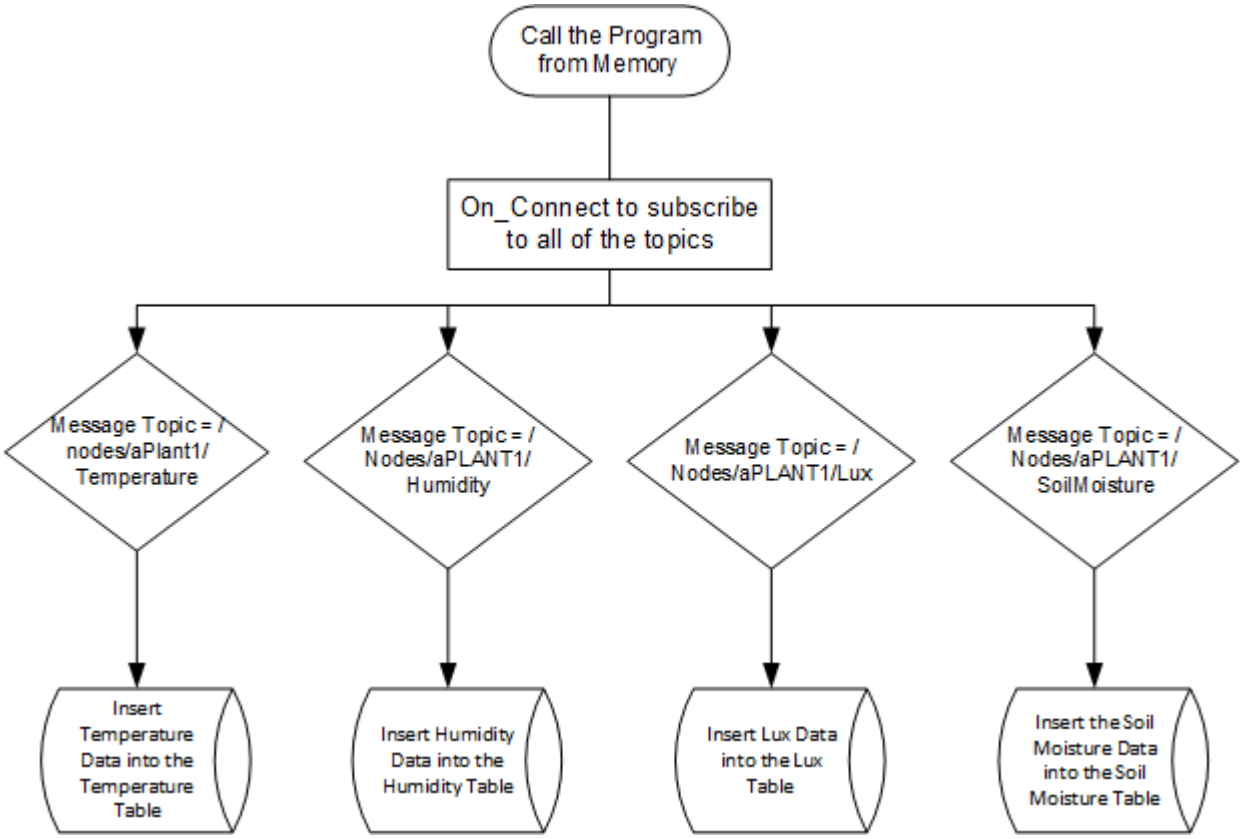

Figure 5: MQTT to SQL Software Flowchart

To connect the MQTT and SQL database, a Python script was created to handle the passing of the MQTT messages to the database. The script subscribes to each of the broker's topics and reads the messages as they are published in real time. These messages have three main parts, including the message topic, which is the MQTT topic, the message payload, which is the data to be passed to the database, and the Quality of Service, which in the case is not relevant since it is 0. The time of reading for each of these messages is also recorded. Both of these recordings are placed in a data tuple. The script then creates a cursor to interact with the database. This cursor is used to pass SQL INSERT commands to the database in order to place the data in appropriate tables.

The SQL data tables are simply designed. There is a table present for each sensor, each one named LUX, MOISTURE, HUMIDITY, and TEMPERATURE. Each table contains two columns. The first column in each table is the date and time of data acquisition. The second column contains the data acquired from the appropriate sensor.

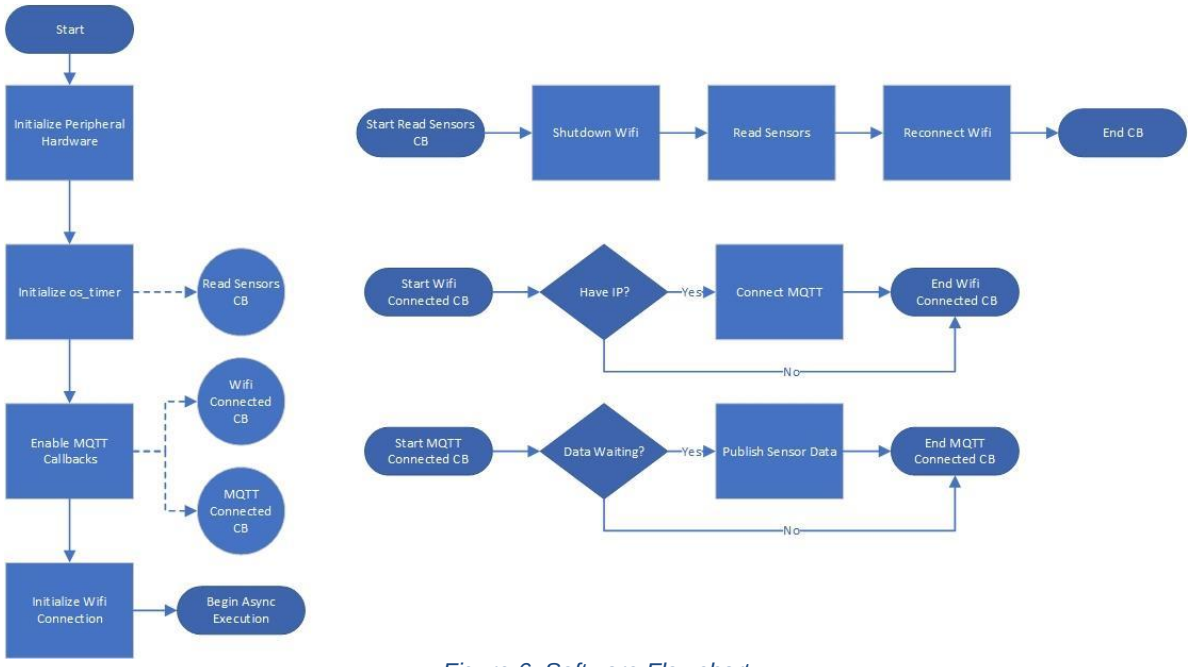

Figure 6: Software Flowchart

As the proper interfacing device, the ESP's operation modes are mostly based on the configuration of the user. Setting intervals for measurement and deciding when to water the plant are pretty good targets for user configuration based on the use case. The user reserves most rights to control the feature set. The other core challenge was making sure that the measurements were accurate and wholesome. Design of the ESP8266 as the node had to incorporate the limitations of the board as well.

The ESP begins its operational time setting the hardware, determining what sensors are connected, initializing the timers built in, and enabling MQTT callbacks so the reporting can begin. The three tasks left are Reading Sensors, connect to Wi-Fi, and re-initialize the connection to MQTT and publish to the topics. All of those events happen on the ESP level and after the initial block, asynchronously.

III. Final Build of Materials

Description	Quantity	Cost
NodeMCU ESP8266 Board	6	$32.91
Adafruit TSL2591 High Dynamic Range Digital Light Sensor	2	$18.88
DFROBOT Gravity: Analog Capacitive Soil Moisture Sensor- Corrosion Resistant	3	$25.80
Plastic Water Solenoid Valve - 12V - 1/2" Nominal	2	$19.98
DHT22	4	$9.90
Raspberry Pi 3b+	1	$37.88
Voltage Regulator	1	$0.00
Optocoupler (PC817)	1	$0.34
End Channel Mosfet (30N06)	1	$1.00
Total		$146.69

Figure 7: Final Prototype BOM

With the functionality intact, the final build of materials ends up being $146.69 for prototyping a design, but even with that, the final product can operate even further under that price point for a completed design.

The costs were greatly inflated by the broker solution. The Raspberry Pi 3b, while a lovely device, is expensive at $37.88 the overhead is for a device that can effectively serve so many roles. In future work, the management of the other ESP8266 can be handled by another ESP.

IV. Division of Labor

Task:	Worked on by:	Status
Research	**All**	Completed

Soil Sensor	**Dylan, Devon**	Completed
Humidity/Temperature	**Ben,** Devon	Completed
Solenoid	**Devon**	Completed
TSL2591	**Devon,** Ben	Completed
Integration on ESP8266	**Devon**, Ben, Dylan	Testing
MQTT	**Dylan**, Devon, Ben	Completed
Raspberry Pi Implementation	Devon**,** Ben**,** Dylan	Testing

Figure 8: Division of Labor

Labor was divided into 2 groupings: Sensor development, and Integration. The majority of each task's efforts are extremely interwoven because troubleshooting was an extensive part of the work, due to the ESP8266's non-OS SDK being rather difficult to code and implement correctly.

References

Simon, M. (2017, November 20). The Hydroponic, Robotic Future of Farming in Greenhouses. Retrieved from https://www.wired.com/story/the-hydroponic-robotic-future-of-farming-in-greenhouses-at-iron-ox/

Thomas, P. A., Westerfield, R., & Pennisi, S. V. (2006, June 01). Growing Ferns. Retrieved from http://extension.uga.edu/publications/detail.html?number=B1318&title=Growing Indoor Plants with Success

CUBASCH, U. and WUEBBLES, D. (2018). *Fifth Assessment Report - Climate Change 2013*. [online] Ipcc.ch. Available at: https://www.ipcc.ch/report/ar5/wg1/.

Appendix

This is the link to the entire project's source code along with some documentation and implementation notes. http://ee.bradley.edu/projects/proj2019/aplant/second.html