

University of Central Florida
Department of Electrical & Computer Engineering
Senior Design 1 – Summer 2019

Final Paper

SmartLeaf
Indoor Greenhouse System



Group 12

Jonathan Gilbert – CpE
Alex Lam – EE/CpE
Dany Mazboudi – EE
Megan Morris - EE

Contents

Contents	ii
1.0 Executive Summary -AL	1
2.0 Motivation - MM.....	2
3.0 Market Motivations - MM.....	2
3.1.1 Market Goal - MM.....	3
3.1.2 Market Appeal - MM.....	3
3.1.3 Market Specifications - MM.....	3
3.1.4 Market and Engineering Requirements - MM.....	4
3.2 House of Quality - MM.....	5
3.3 Health & Wellness Motivations - MM.....	6
4.0 Constraints - MM.....	7
4.1 Size and Weight Constraints - MM.....	7
4.2 Power Constraints - MM.....	8
4.3 Realistic Design Constraints - DM.....	9
4.3.1 Social Constraints - DM.....	9
4.3.2 Sustainability - DM.....	10
4.3.3 Manufacturability - DM.....	10
5.0 Research - MM	11
5.1 Plant Needs - MM	11
5.1.1 Light - MM	12
5.1.1.1 Light Intensity - JG.....	13
5.1.2 Water & Drainage - MM.....	14
5.1.3 Nutrition - MM	15
5.1.4 Temperature - MM.....	16
5.1.5 Humidity - MM.....	17
5.1.6 Airflow - MM	17
5.1.7 Overall Environmental Requirements - MM	18
5.2 Types of Plants - JG	19
5.3 Sensors - DM.....	20
5.3.1 DHT22 Temperature and Humidity Sensor - DM	20
5.3.2 High Sensitivity Water Sensor – Red Version - DM.....	21
5.3.3 LM393 Soil Moisture Sensor - DM	21
5.3.4 pH Sensors – JG/DM	22

5.4	Microcontrollers -AL	23
5.4.1	Main MCU – AL/JG	23
5.4.2	WiFi Controller – AL/JG	23
5.4.3	Bluetooth Controller – AL/JG	24
5.4.4	Auxiliary Microcontrollers (Sensor Nodes) - JG.....	24
5.4.5	Auxiliary Bluetooth Modules - JG.....	25
5.5	Communication - JG	26
5.5.1	UART - JG.....	26
5.5.2	I2C - JG.....	27
5.5.3	SPI - JG.....	27
5.5.4	Wireless connectivity (WiFi) - JG	27
5.5.5	HTTP - JG.....	27
5.5.6	Bluetooth – AL/JG.....	28
5.5.7	Zigbee - JG.....	28
5.5.8	Sub-1GHz Transceivers - JG	28
5.5.9	IoT - JG.....	29
5.5.10	Publishing Messages - JG	30
5.5.11	PubNub - JG.....	31
5.6	Communication Design - JG	33
5.6.1	Developing in CCS - JG	33
5.6.2	Developing in Energia - JG.....	33
5.6.3	AWS Cloud Solution - JG.....	33
5.6.4	Google Cloud Solution - JG.....	34
5.6.5	Communication challenges - JG	35
5.7	Programming Languages - JG.....	35
5.7.1	C/C++ - JG.....	35
5.7.2	Python - JG	36
5.7.3	Assembly - JG.....	36
5.7.4	JavaScript - JG	36
5.7.5	AT Commands -AL	36
5.8	Software – JG.....	36
5.8.1	Code Composer Studio - JG	36
5.8.2	Energia - JG	38
5.8.3	Uniflash - JG.....	38

5.9	Operating systems - JG	38
5.9.1	Why Use an Operating System? - JG	39
5.9.2	TI-RTOS - JG	39
5.9.3	AWS FreeRTOS - JG.....	39
5.10	Cloud – JG.....	40
5.10.1	CCS Cloud - JG	40
5.10.2	Security - JG	41
5.10.3	Google Cloud Platform - JG	41
5.10.4	Amazon IoT Services - JG.....	42
5.11	GUI -AL.....	42
5.12	Bluetooth Modules -AL	43
5.12.1	DSD Tech HC-05 - AL.....	44
5.12.2	HiLetgo HC-05 – AL.....	44
5.12.3	HiLetgo CC3541 HM-10 – AL.....	44
5.13	LEDs -AL.....	45
5.13.1	Adafruit RGBW LEDs -AL.....	45
5.13.2	CHINLY LEDs -AL	45
5.14	LCD Screen -AL	45
5.14.1	Adafruit LCD -AL	45
5.14.2	Booster pack LCD for MSP432 -AL	46
5.15	Fans – DM/MM.....	46
5.15.1	RDK Brushless Cooling Fan - MM	47
5.15.2	CUI DC Axial Fan – 5V - MM.....	47
5.15.3	CUI DC Axial Fan – 12V - MM.....	47
5.16	Pumps – DM/MM	47
5.16.1	Ledgle Submersible Mini Pump - MM.....	48
5.16.2	Daniu Submersible Brushless Water Pump - MM.....	48
5.16.3	Peristaltic Liquid Pump - MM	48
5.17	Humidifier - MM.....	49
5.17.1	APGtek Aluminum Mini Mist Maker - MM	49
5.17.2	IC Station Ultrasonic Mist Maker - MM	50
5.18	Power Supply - MM.....	50
5.18.1	Power Calculation - MM	52
5.18.2	Power Supply Selection - MM.....	53

5.19	Voltage Regulators - MM	54
5.19.1	Linear Regulators - MM	54
5.19.2	Switching Regulators - MM.....	54
5.20	Batteries - MM	56
5.21	Control Systems - MM.....	58
5.21.1	Power MOSFET - MM	58
5.21.2	Motor Controllers - MM.....	58
5.21.3	MSP432 Control -AL.....	60
5.22	PCB Design Software - DM.....	60
5.22.1	Kicad and Eagle - DM	60
5.22.2	EasyEDA - DM.....	61
5.22.3	PADS - DM.....	61
5.22.4	OrCAD - DM	62
5.22.5	NI Circuit Design Suite - DM.....	63
5.23	Board Environment - DM	64
5.23.1	JLCPCB - DM.....	64
5.23.2	4PCB - DM	65
5.23.3	Sunstone - DM	65
6.0	Vendors - MM	65
6.1	Texas Instruments – DM/MM.....	65
6.2	Adafruit - MM.....	66
6.3	Smart Prototyping - MM	66
6.4	Mouser - JG.....	66
6.5	Amazon - MM.....	67
7.0	Standards - MM.....	67
7.1	Greenhouse Environment Standards - DM	67
7.2	Software Standards - JG.....	67
7.3	Communication Standards - JG.....	68
7.3.1	WIFI Standards - JG	68
7.3.2	Bluetooth Standards - JG	68
7.3.3	MQTT Standards - JG.....	69
7.4	Power Standards - MM	70
7.4.1	NFPA 70 National Electrical Code - MM	70
7.4.2	ASHRAE Standard 90.1 - MM.....	71

7.5	PCB Standards – DM	71
7.5.1	Component Placement - DM	71
7.5.2	Traces - DM	72
7.5.3	Separation of Components and Mitigating Heat Issues - DM	72
7.6	Design Standards -AL	73
8.0	Safety Standards - MM.....	75
8.1.1	FCC Requirements - MM	75
8.1.2	Miscellaneous Safety Testing - MM.....	75
8.1.3	NRTL Program - MM.....	75
8.1.4	IP Code - MM.....	75
8.1.5	UL 8750 LED standard.....	77
8.2	Sensor Standards - MM.....	77
8.2.1	IEEE Standard for Sensor Performance - MM	77
9.0	Project Design -AL.....	78
9.1	Design Goals -AL.....	78
9.1.1	Hardware Goals - AL.....	78
9.1.2	Software Goals -AL	79
9.2	Power Supply & Rail Design - MM.....	79
9.2.1	TPS56637 Synchronous Buck Converter - MM.....	81
9.2.2	LMR62014X Simple Switcher - MM.....	81
9.2.3	TPS55330 Boost/SEPIC/Flyback DC-DC Regulator - MM.....	82
9.2.4	TPS560430Y Synchronous Step-Down Converter - MM	82
9.2.5	TPS62175 Step-Down Converter with Sleep Mode - MM.....	83
9.2.6	TPS62177 Step Down Converter with Sleep Mode - MM.....	83
9.2.7	Efficiency and Overall Power Demands - MM	84
9.3	Embedded Software Design - JG	85
9.3.1	Communication Design - JG.....	86
9.4	GUI Design - AL.....	87
9.5	PCB Design - DM	87
9.5.1	PCB Planning - DM	87
9.5.2	Schematic Creation – DM/AL	87
9.5.2.1	Hierarchal Sheet #1 for Fans - DM	89
9.5.2.2	Hierarchal Sheet #2 for LEDs – DM.....	90
9.5.2.3	Hierarchal Sheet #3 for Water Pump - DM.....	92

9.5.2.4	Hierarchal Sheet #4 for Humidifier - DM	94
9.5.2.5	Hierarchal Sheet #5 for the LCD Screen - DM	95
9.5.2.6	Hierarchal Sheet #6 for the MSP432 – DM/AL	97
9.5.3	Assigning Footprints to Schematic Symbols - DM	100
9.5.4	PCB Layouts - DM	101
9.5.5	Traces and Planes - DM	104
10.0	Component Integration/ Testing – MM/AL	105
10.1	Hardware Testing – MM/AL	105
10.1.1	Continuity Testing -AL	105
10.1.2	Power Supply Functionality Test - MM	106
10.1.3	Voltage Regulator Functionality Test - MM	106
10.1.4	Microcontroller Functionality Testing -AL	107
10.1.5	Temperature/Humidity Sensor Testing -AL	107
10.1.6	Soil Moisture Sensor Testing -AL	107
10.1.7	pH Sensor Testing - MM	108
10.1.8	Fan Functionality Testing -AL	108
10.1.9	Pump Functionality Testing - AL	108
10.1.10	LEDs Testing -AL	109
10.1.11	LCD Screen - AL	109
10.2	Communication Testing - JG	110
10.3	Software Testing - JG	110
10.3.1	WiFi Testing - JG	111
10.3.2	Bluetooth Testing - JG	112
10.3.3	MQTT Testing - JG	112
10.3.4	PubNub Testing -JG	112
10.3.5	Bluetooth Testing - AL	114
10.4	Prototype - MM	115
10.5	Prototype Planned Pin Layout -AL	116
10.5.1	Parent Microcontroller -AL	116
10.5.2	Child Microcontroller -AL	116
11.0	Greenhouse Construction - MM	117
11.1	Placement and Mechanics of Environmental Controls - MM	118
11.1.1	LED Placement - MM	118
11.1.2	Watering Mechanism - MM	119

11.1.3	Fan Placement - MM	119
11.1.4	Humidifier Placement - MM.....	120
11.2	Final Planned Pin Layout -AL	120
11.2.1	Parent Microcontroller -AL	121
11.2.2	Children Microcontrollers -AL.....	121
11.3	GUI User Guide -AL.....	121
11.3.1	Home Page -AL	123
11.3.2	Sensor Pages -AL.....	124
11.3.2.1	Connected Plant Page -AL	125
11.3.2.2	Pairing New Plant Page -AL	126
11.3.3	LED Page -AL	127
11.3.4	Environment Changer Page -AL.....	128
11.3.5	Cloud Page -AL	129
11.4	Project Operation & User’s Guide - MM.....	130
12.0	Project Administration - MM	130
12.1	Budget - MM.....	130
12.2	Milestones	132
12.2.1	Senior Design 1 Milestones -AL.....	132
12.2.2	Senior Design 2 Milestones -AL.....	132
13.0	Final Comments - MM	133
13.1.1	Future Goals - MM	133
13.1.2	Potential Areas of Expansion - MM	133
13.1.3	Conclusion - MM.....	133
14.0	Appendices	134
14.1	Permissions.....	134
14.1.1	Niwa One Request for Permission.....	134
14.1.2	Science ABC Request for Permission.....	134
14.1.3	IC Station Request for Permission.....	135
14.1.4	NEMA Enclosures Request for Permission.....	135
14.1.5	Renesas Electronics Request for Permission	135
14.1.6	NEMA Request for Permission	136
14.2	User Pin Layout.....	136
14.2.1	Parent controller - AL	136
14.2.2	Child Controller - AL.....	139

15.0 Works Cited..... 140

Figure 1- Examples of indoor greenhouse/gardening systems available on the market today. Permissions requested from Niwa One..... 2

Figure 2 - House of Quality 5

Figure 3- Overall equation for the type of photosynthesis that occurs in plants 12

Figure 4- Proper soil composition for healthy roots. Permissions requested from ABC Science 15

Figure 5- IoT Ecosystem..... 30

Figure 6- How the pub/sub MQTT mechanism works 30

Figure 7- PubNub channel topologies..... 32

Figure 8: Communication Model..... 34

Figure 9- MQTT Client-Server Demo [26] 38

Figure 10: AWS FreeRTOS architecture 40

Figure 11: Example Google IoT Device Framework 41

Figure 12: TI GUI example..... 43

Figure 13- Receptacle and plug type used by the SmartLeaf system. Permissions requested from NEMAenclosures.com 51

Figure 14- Compatible plugs for the NEMA 5-15R; NEMA 1-15 (left), NEMA 5-15 (right). Permissions requested from NEMA 51

Figure 15- Power savings between low-dropout linear regulators and switching regulators. Permissions requested from Renesas Electronics 55

Figure 16- One-line diagram for the SmartLeaf System 57

Figure 17- Control Diagram..... 59

Figure 18: Block Diagram of Smart Greenhouse 78

Figure 19- General power rail design 80

Figure 20- LED power rail circuitry 81

Figure 21- Fan power rail circuitry 81

Figure 22- Pump power rail circuitry..... 82

Figure 23- Humidifier power rail circuitry 83

Figure 24- LCD display power rail circuitry 83

Figure 25- MSP430 power rail circuitry 84

Figure 26: Plant sensor node architecture 85

Figure 27: Communication Block Diagram..... 86

Figure 28: Schematic for Parent MCU 88

Figure 29: Fan Power Circuit with Connections via Global Labels in Kicad 89

Figure 30: Assigned Footprint for LMR62014XMF Simple Switcher..... 90

Figure 31: LED Power Circuit with Connections via Global Labels in Kicad..... 91

Figure 32: Assigned Footprint for the TPS56637RPA Synchronous Buck Converter..... 92

Figure 33: Water Pump Power Circuit with Connections via Global Labels in Kicad 93

Figure 34: Assigned Footprint for the TPS55330RTER DC-DC Regulator 94

Figure 35: Humidifier Power Circuit with Connections via Global Labels in Kicad..... 94

Figure 36: Assigned Footprint for the TPS560430Y Synchronous Step-Down Converter..... 95

Figure 37: LCD Screen Power Circuit with Connections via Global Labels in Kicad..... 96

Figure 38: Assigned Footprint for the TPS62175DQC Step-Down Converter 96

Figure 39: MSP432 Power Circuit with Connections via Global Labels in Kicad 97

Figure 40: Assigned Footprint for the TPS62177DQCR Step-Down Converter 98

Figure 41: Schematic for MCU Child #1.....	99
Figure 42: Schematic for MCU Child #2.....	100
Figure 43: Footprint assignment to symbols in parent PCB.....	101
Figure 44: PCBNew Main PCB outline.....	102
Figure 45: PCBNew Child #1 PCB Outline.....	103
Figure 46: 3D Viewer Mounting Hole clearance on MSP430 on child PCB #1.....	103
Figure 47: PCBNew Child #2 PCB Outline.....	104
Figure 48: Example of a Ground Plane in PCBNew.....	105
Figure 49: WiFi software block diagram.....	111
Figure 50: PubNub communication block diagram.....	113
Figure 51: Creating a widget on Freeboard. JSON format makes accessing our sensor data easy.	114
Figure 52: Freeboard Dashboard. This is where we can visualize our data sent from the microcontroller to PubNub using MQTT.....	114
Figure 53: Example Bluetooth Test Sequence.....	115
Figure 54- Front and back view.....	117
Figure 55- Side views.....	118
Figure 56- Watering mechanism demonstration.....	119
Figure 57- Basis of design for the humidifier. Permissions requested from IC Station.....	120
Figure 58: GUI Block Diagram.....	122
Figure 59: Example Home Page.....	123
Figure 60: Example Sensor Home Page.....	124
Figure 61: Example Connected Plant Page.....	125
Figure 62: Example Plant Insertion Page.....	126
Figure 63: Example LED Show Page.....	127
Figure 64: Example Environment Changers Page.....	128
Figure 65: Example Cloud Page.....	129

1.0 Executive Summary - AL

This document serves as the final report documentation for the *Smart Tabletop Greenhouse*. This project is to be designed and developed in accordance with the University of Central Florida's Computer/Electrical Engineering Senior Design course and the Accreditation Board for Engineering and Technology's (ABET) requirements for accredited engineering programs. The final goal of this project is to showcase our groups skills in brainstorming, researching, designing and fabricating a working product that is within the constraints of our choosing.

There rarely are opportunities to work on such a project that has these many places where an individual's ingenuity skillset can be showcased to such a degree. Since there have been similar previous designs as the one we have envisioned, we were put up with an additional task to come up with a way to stand out as a team all the while of producing a product that is realistic and doable in the time that we have to work on it. Our group did a fantastic job of assigning tasks to each individual, set up a realistic timeline for everyone to be accountable for, be understanding when issues that were out of the hand of the individual occurred, and communicating to the rest of the team of our progress and if any of us needed any assistance in the tasks that we were assigned with.

The first half of this document covers the overall motivation, how we originally planned on proceeding with our initial ideas, research and what we found to be feasible in our timeframe, and the initial design and requirement specifications of our projects. The second half this document covers more on how we interpreted our requirement specifications, how we implemented said specifications, how we went about testing our product to see that it met all of our requirements and were within our constraints, a general user guide for the user to figure out how to operate our product, and the overall administrative planning that went into creating our product. Initially our group was overambitious with what we thought we could do in our timeframe, but fortunately we were able to scale it back to present a finalized product by the end of the two semesters allotted for the time to finish this project.

Smart greenhouses and hydroponics are new horticulture techniques when compared to the methodologies that mankind has had since its beginning of cultivating crops and plants for their own needs. Only in the past century or so when new technologies emerged to observe, regulate, and optimize a plants growth are when these methods were able to shine. Using these methods were found to cultivate nearly all crops faster, and usually assisted them in creating a higher yield in the process.

Further research and ingenuity will continue to help these new horticulture techniques grow into a reliable and consistent method of raising crops and hopefully increase the growth in the community that has the potential to gather many enthusiasts, create a better and more optimal environment for all, generate a high revenues for the agricultural community, and ultimately change the current way that we currently cultivate our food.

2.0 Motivation - MM

This section serves to outline the motivation and intent behind the project and provide a means of meeting the market and user wants through engineering specifications. Comparable indoor units will be noted, and the basis of design for this project will be identified.

3.0 Market Motivations - MM

Small, indoor greenhouses/gardening systems already exist on the market. However, those tend to fall into one of four categories (listed from least to most expensive):

- Simple windowed enclosures with no control capabilities- these are typically used as an interior design element, but they also prove a simple solution to holding in humidity and providing protection from pests. Terrariums also fall into this category.
- Agricultural ‘sheds’ with pliable enclosures and shelving- these tent-like structures are typically an indoor/outdoor unit, but their affordability and diverse size range makes them good candidates for users who want to efficiently generate produce within their space. The open floor and flexible enclosure allow for the inclusion of separate watering and light systems.
- Small, open-topped tubs with overhead light- these are most notably the hosts of countertop herb gardens. Some models that are self-watering, most are limited to grow only herbs due to their small root systems and lenient humidity requirements.
- Futuristic-looking ‘smart’ tank enclosures- smart gardens with app-control. In general, they water the plants and provide them with light and can be controlled via an iOS/Android app. More expensive varieties provide humidity control. Design and size options are limited.



Figure 1- Examples of indoor greenhouse/gardening systems available on the market today. Permissions requested from Niwa One.

3.1.1 Market Goal - MM

This project's goal is to combine the desired elements from the above categories (as shown in Figure 1) to create a device that can more closely target the end user's desires. More specifically, the aesthetic appeal of the simple windowed enclosure, the maneuverability and functionality of the agricultural shed, the conservative size and accessibility of the open-top tub, and the user-friendly controls and automation of the smart tank enclosures. With this goal outlined, market specifications can be explored.

3.1.2 Market Appeal - MM

A modern design is necessary to appeal to consumers, but the hyper-futuristic designs present in comparable units should be avoided. This will provide distinction on the market and have a greater emphasis on aesthetic beauty to further differentiate the enclosure. The unit will be a cleanly integrated, complimentary feature to a living space, while simultaneously providing a nurturing and accessible environment for plant growth. A vertical orientation will conserve floor space while maximizing the area for new growth; this format also allows for the plants to be the greatest emphasis, softening the overall look of the unit. This will be further emphasized with wooden and textured finishings to nicely frame the foliage and cleverly hide electronic components.

The user will be able to comfortably control, monitor, and alter the greenhouse conditions via web app from a computer, tablet, or mobile device. Given time and budget constraints, this method is more feasible than creating an iOS or Android application while ultimately working for devices of either party. Specific user controls will be described later in the document, but the aim is to expand the user's ability to modify greenhouse conditions while not overwhelming them with free range. This will be a significant feature as the inability/limitations in making alterations to pre-existing care routines is often a complaint of similar devices.

In terms of installation, any device with plants involved generally constitutes a greater setup time than other household hobbyist-units. This should be streamlined and made as simple as possible for the end user, specifically in regard to sensors that need to be placed into soil. This will be thoroughly detailed later in the Greenhouse Operation section. Installation for professionally manufactured units have the upper hand, as most have a single planter with integrated sensors to determined conditions; regardless, this project will allow the user to monitor multiple planters. This grants the user more flexibility in plant placement while simultaneously reducing design cost, making the unit more affordable and competitive on the market.

3.1.3 Market Specifications - MM

With all this in mind, the market specifications for this project are as follows:

1. Greenhouse should have a modern appeal with an organic and simple design
2. The greenhouse controls system should be automated and controlled with a web app
3. The app should be intuitive and easy to navigate
4. The installation process should be safe and easy
5. The unit should be affordable in comparison to similar models on the market

3.1.4 Market and Engineering Requirements - MM

To achieve the market requirements, engineering requirements are defined to quantitatively set parameters and provide structure for how to move forward with the design. outlines the engineering requirements, specifies which market requirements they target, and provides justification for their inclusion in the design.

Table 1 - Market and Engineering Requirements

Market Requirements	Engineering Requirements	Justification
2	Should have a web app that supports WIFI or Bluetooth 4.0 connectivity to a microcontroller	'smart' devices typically allow for mobile control and monitoring of a system over Bluetooth 4.0 or an internet connection
2	The device must communicate with the cloud to store data	We won't have much space on the microcontroller, so storing the data online is the best solution.
1	Power supply shall not obscure the plants from receiving sunlight or detract from the overall appearance of the system.	Nothing added to the system is to negatively impact the quality of the plants.
2	Should have various sensors to measure the environmental factors that the plant life is experiencing.	In order to provide a nurturing environment for plant growth, factors such as humidity and hours of sunlight are to be monitored and used for controls
2	Should include a watering mechanism for all plant life within the system.	Having a self-watering greenhouse will give the system a user-friendly and foolproof nature
2,3	Should collect and display environmental data (from the sensors)	Users will want to be informed of the greenhouse conditions
4	Should be within 5 cubic feet with a vertical orientation.	Due to this being an indoor device, it should take up as little space as possible
2	Should have a user-friendly GUI	Have an easy-to-access interface is imperative for data collecting on this scale
2	Should be able to control environment changers in the system	We need to be able to change the environment to optimize the growth of the different plants

3.2 House of Quality - MM

The House of Quality is a tool used to describe and define the various relationships between marketing and engineering requirements for a product. It also provides an indication for how changes made in one portion of the product can impact other areas. The base of the chart (or ‘house’) defines the relationships between the engineering and market items as having a strong relationship, moderate relationship, weak relationship, or no relationship. The ‘roof’ of the table shows the connection between engineering requirements, from having a strong positive correlation, positive correlation, negative correlation, and strong negative correlation. The roof is particularly important in showing potential conflicts that may arise during the engineering design process. The row at the base of the roof indicates if that individual engineering requirement is to be minimized, maximized, or targeted. A house of quality for product design is a necessity as it provides clear documentation of the roles and relationships at play. Ideally, it will speed up design and prevent serious miscommunication on the engineering team. The House of Quality for the greenhouse system is shown below in Figure 2.

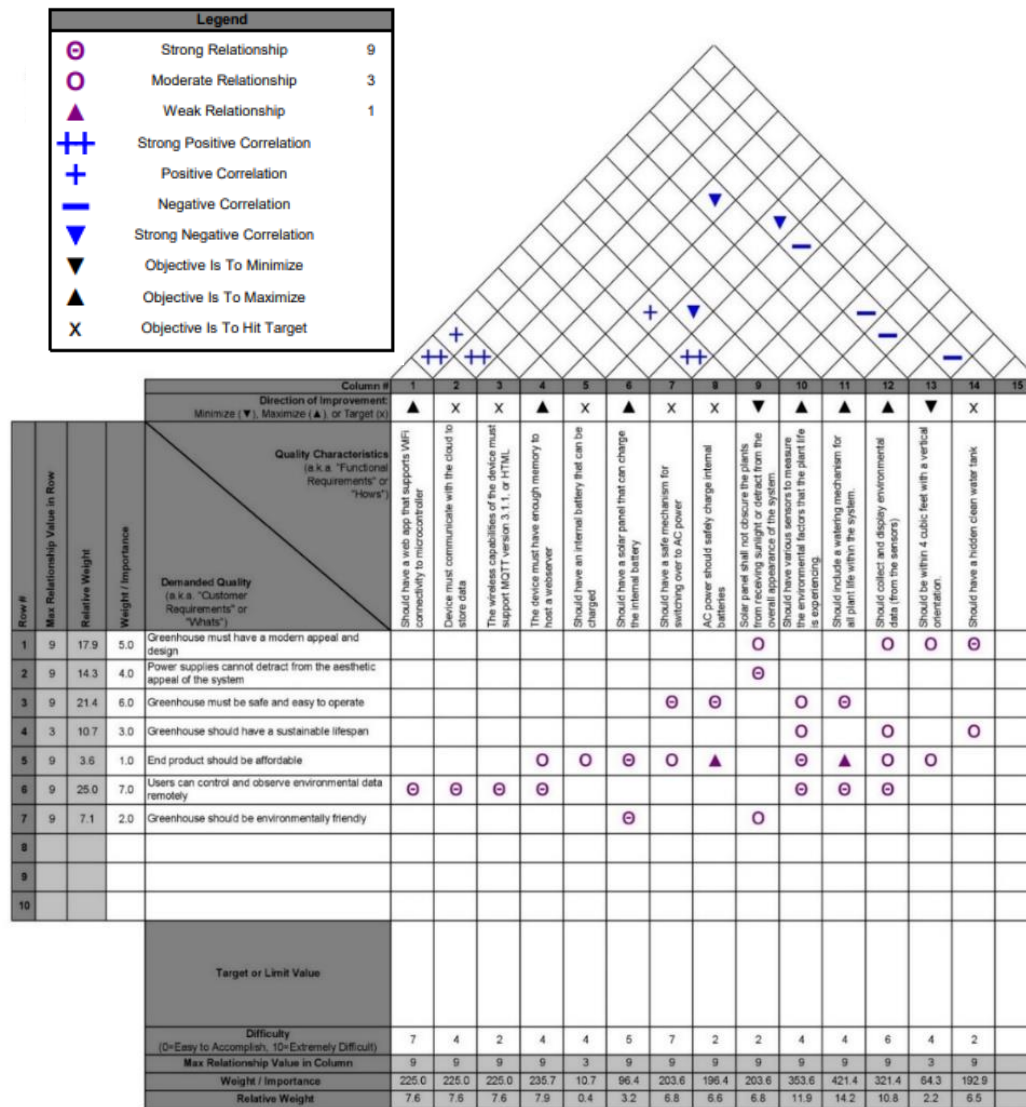


Figure 2 - House of Quality

3.3 Health & Wellness Motivations - MM

Most people can appreciate the aesthetic and air-purifying qualities that house plants have to offer – not to mention the potential savings and health benefits of growing fruits and vegetables – but lack the ‘green thumb’ needed to keep them alive and happy. With greater monitoring and control of the plant’s environment, indoor gardening could become more feasible for the typical household.

The appeal of indoor gardening extends past the convenience of producing vegetation for food, though that ability should not be taken lightly. ‘Food deserts’ – urban areas where it is difficult to obtain affordable or good-quality fresh food – are a rising issue in the United States. These regions tend to coincide with higher rates of poverty, and also bring in a slew of health issues such as obesity and heart disease. People living in these conditions often have to choose between buying expensive produce in inconvenient quantities to then go home and prepare meals for their household, or buying cheap, highly caloric fast food that is filling, albeit non-nutritious. If an indoor garden system could be made available to these homes at a reduced rate, it could at least save them time and money in procuring fresh vegetation and give more meal options.

In addition to providing greater access to fresh produce, indoor plants have been known increase the air-quality within a confined space and, quite simply put, can make people happier. Through photosynthesis, plants convert the carbon dioxide humans exhale into fresh oxygen. During this process, as found in a 1989 experiment conducted by NASA, plants have the ability to filter the air of volatile-organic compounds such as formaldehyde and benzene [1]. Within a space with regularly circulating air, such as a typical home, this effect is reduced. However, with a greater volume of plants in a controlled environment, it is possible reduce the level of pollutants that would be present in an urban environment. Another health-based benefit of houseplants are its effects on mood. A 2015 study conducted by the National Center for Biotechnology Information demonstrated how interaction with indoor plants can reduce psychological and physiological stress by calming the autonomic nervous system activity in young adults [2].

Even with all the benefits associated with indoor gardening, there are likely many people who wouldn’t consider incorporating it into their daily lives due to the difficulty of providing and maintaining a nurturing environment for plant growth. Some main issues that can impact a plants ability to thrive indoors are over/under watering, lack of drainage, too much or too little sunlight, dry air, and depleted nutrients in the soil. The good news is that most of these factors can be monitored and controlled in a smart greenhouse setting – soil moisture sensors can be used to indicate when to water, sunlight sensors can determine if the garden is receiving adequate light, hygrometers can be incorporated to monitor ambient humidity, and soil pH sensors are able to serve as an indication for nutrient deficiencies. Having access to these factors alone could be sufficient to caring for this vegetation, but these sensors could also be used communicate with other devices to automatically maintain a nurturing environment; i.e., humidifiers, a watering mechanism, grow lights for supplemented sunlight, and even a separate reminder system for fertilization.

4.0 Constraints - MM

This section serves to detail the project constraints for the indoor greenhouse system. Some constraints are imposed by factors and limitations outside of our control and need to be accounted for during the engineering process. Other constraints have been elected to provide a basis of design and help meet the market requirements.

4.1 Size and Weight Constraints - MM

The greenhouse system is intended for indoor use in houses, apartments, offices, and retail environments as a consumer product. These environments are expected to have limited space, and it is important to consider the actual size of the finished product prior to implementation of the design. The device needs to be large enough to operate as a functioning greenhouse, but small enough to exist comfortably and not hinder the functionality of a space. A large indoor greenhouse design could be considered as an architectural detail for visual interest, but the average consumer would likely not have the funds, area, or ability to install the unit. For this reason, we have decided to make the greenhouse vertically oriented and containable within a 2' x 2' square area. This conserve floor space and feasibly entry into doorways while maximizing planting ability with vertical gardening practices.

A tall build is ideal as it will capture the visual interest of passersby by pure merit of existing within the plane of the average adult's eyelevel. However, height is not an unlimited resource, and the size limitations of ceilings – and especially doorways – needs to be considered. The typical dimension of a residential interior door is 6'8" x 2'8", while the typical ceiling height in the same environment is 8' to 9'. Factors like tight corners and awkward inclines in residential spaces can prevent or hinder horizontal movement of the unit to fit through doorways into larger open areas safely. For this reason alone, the size is to be constrained to fit vertically through a doorframe with a few inches of clearance: 6' x 2' x 2'.

The weight of the finished unit needs to be considered primarily for safety concerns during the installation process and prevent the unit from tipping if a minor collision occurs. OSHA notes that individuals lifting loads that exceed 50 pounds increases the risk of injury [3]. While it would be ideal to limit the weight of this design to 50 pounds, this could potentially limit greenhouse functionality and create nuisances for the user by limiting the amount of water the system can hold at a time, requiring water to be filled and emptied more frequently.

Another concern for the system revolves around weight distribution. The greenhouse is to be vertically oriented as to preserve floor space and create visual interest, but this design choice could increase the risk of the unit tipping if the top is significantly heavier than the base. To mitigate this risk, the finished unit will be designed to have a low center of gravity to prevent this from occurring. All water tanks and applicable electrical components are to be contained to the base of the unit to aid in tipping prevention, and additional weight can be incorporated to the structure if the components and furnishings do not provide sufficient stability. In order to provide a low enough weight for a feasible installation while still allowing enough freedom to potentially add weight for stability, the unit weight will be constrained to 90 pounds. This will allow for two people to safely move the device into place while mitigating the risk of tipping.

The final size and weight constraints applied to this project can be seen in Table 2.

Table 2- Project size and weight constraints

Variable	Maximum Dimension
Length	2 ft
Width	2 ft
Height	6 ft
Weight	90 lbs

4.2 Power Constraints - MM

As it is expected for the indoor greenhouse system to reside within a residential home or commercial space, the most straightforward and user-friendly way for the device to receive power is from mains via standard wall-mounted receptacles. This is more feasible than a solar or battery powered design that would require additional maintenance; it may, however, restrict placement for the unit, as it needs to be within the length of its power cord to an outlet. Receiving power directly from mains limits the degree of energy-efficiency that can be achieved, as the power will be stepped down significantly, and some energy will be lost to heat during the conversion.

While this could result in a marginally higher electric bill, any similar system with solar power would be a significant additional expense and have more potential points of failure. Even without the consideration of expense and engineering complication, the incorporation of a solar-powered system for a greenhouse introduces a significant contradiction: it is assumed that the indoor system will not receive enough sunlight from the ambient surroundings to encourage plant growth without the addition of grow-lights. If we assume poor lighting conditions exist that our device needs to correct, we cannot also assume there will be enough light to power our device. External workarounds could be applied, but not without additional complications and detracting from the overall aesthetic design.

Typical receptacles are rated for 15A, and at maximum receive 120VAC. Therefore, the absolute maximum power we have available to our system is 1800W; however, it is unlikely that any single general-purpose receptacle (GPR) would be drawing 15A as it is common practice to have multiple GPR's on a single circuit. Luckily, most components with applicable use to a greenhouse of this size require significantly lower input voltage for standard operation, and the use of a step-down transformer can effectively lower the voltage delivered to certain components within our device while providing enough current to power several small components.

Per the reasons outlined in the paragraphs above, power delivered to our device will be contingent on the use of a step-down transformer that receives 120VAC, 15A at maximum from a GPR on the primary line. Further details on this transformation can be found in the *Power Supply* section.

4.3 Realistic Design Constraints - DM

As engineers, we need to consider a multitude of things before we begin constructing a project. We need to look at safety being on the top of our priorities list before we even begin building our smart greenhouse project. We need to answer the question of how we're going to tackle this in the most efficient and safest way possible. For this we're taking careful consideration of how we're going to go about powering our project to satisfy the issue of safety. Though, safety is not the only thing that we need to look at regarding these things.

ABET needs to see the impact with regards to economic, social, political, environmental, safety, sustainability as well as manufacturability. When it comes to our greenhouse project, there will be little to no economic impact since we're doing this project as a university level project and not as a project that we're going to manufacture and distribute to consumers on a widespread scale. When it comes to environmental impacts, the greenhouse we're creating will be something that people will have within their homes. I know that when talking about greenhouse projects, there is a concern when it comes to greenhouse gasses but there will be no issue with regards to our project since it's much smaller than the typical large-scale greenhouse that one would see a large company creating.

4.3.1 Social Constraints - DM

Now when we talk about social constraints, things could be a bit extensive given the size of a group that's working on the project and their schedules surrounding it. We need to talk about a few things that fall under social constraints such as trust, time, tools, materials, resource availability and cost. When it comes to trust, we need to make sure that each of our group members have a clear idea of what needs to be done and we all need to agree upon an idea and how we're going to be implementing it. We cannot say we're going to do something then suddenly one of our team members decides we need to do something else. There's only one exception to this and that is if we all agree to the change that the group member is suggesting. This will likely happen if it's a very important change that can affect the efficiency as well as the safety of the project we're going to be building.

Also falling under trust, we need to also all agree to a planned-out schedule as to when we all need to meet to work on this project and get it done. If anything, this is the most important factor when it comes to trust since this is a team effort, we all need to get together to get this done. Next, under social constraints will be the materials. When we design our product and need to decide on what we need to construct it, we need to look at all aspects of the materials and what can go wrong. We need to look at factors such as the weight of the material, strength of the material as well as the probability of the material we're going to be using being corrosive. If we use material that will corrode with our greenhouse project, then we'll be facing major setbacks since we need to find a replacement for those corrosive materials.

Next for social constraints we need to look at the tools we're going to be using as well. We need to ensure that we have everything we need to solder components onto the PCB and to get everything tested. We're going to need to have a solid functioning soldering iron as well as a laptop/pc to test certain components of the project as well as an oscilloscope and the computer software required to run these tests and for the design of the PCB. Another social constraint we ought to look at is the resource availability. If we wanted to order a certain sensor, we need to make sure that there's plenty in stock for us to order them. We also need to make sure that when

we place these orders, we're also getting them from vendors that we can deem as reputable with some solid credentials to make sure that the parts we order work. If we order from a well-known vendor, we can be more confident in our purchase and can use our parts with confidence.

Also, back to the discussion earlier about having more than enough stock on standby for us to utilize, if we order our parts and we need to purchase more for excess, we need to make sure that they are still available for purchase and that they are not sold out. That would really inconvenience the group as well as the project entirely. Finally, for social constraints we need to talk about the cost of our project.

Before we even purchase these parts, we as a team need to decide on a rough estimate of how much this project is going to cost us and base our budget around that rough estimate. We also need to make sure that the estimate that we make for this project is reasonable and realistic. We don't want to underestimate how much this greenhouse project is going to cost us. Now this is all just regarding the social constraints, now when it comes to the political constraints, there really isn't much to speak of given the scope of our project and what our project is about. It being a greenhouse project on a university level, there is much doubt that there will be any political impact whatsoever, so there's no need to have any concern over this.

4.3.2 Sustainability - DM

Up next for discussion would be the constraints regarding sustainability. Our project will be designed with sustainability in mind. How sustainable our project will be will honestly be dependent on the project itself. Our project being a university level project will not have the idea of being sustainable for an extensive period such as a time span of a few years. We are not designing our project. We are designing this project with the idea of showcasing it to professors and not for something that is going to be laying around for a very long and extensive period. Next and finally we need to talk about manufacturability.

4.3.3 Manufacturability - DM

When it comes to our greenhouse project, the constraint of manufacturability is something we need to look at in terms of building it. We need to look at exactly how feasible it is to put this project together is a question that must be asked when looking at any sort of team-based project. With a project of our scope, the constraint regarding manufacturability is nothing that is too detrimental to us. We already have a plan of materials we're going to be using and we already laid out how we're going to construct the project, so now all we need to do is put it all together. Justifications for the realistic design constraints noted in this document can be found Table 3 below.

Table 3- Realistic Design Constraints

Realistic Design Constraint	Justification for Constraint
Economic	Not sold to consumers

Realistic Design Constraint	Justification for Constraint
Environmental	Too small of a scale for greenhouse gas pollution
Social	Careful discussion with team members
Political	Our project will not have any political constraints due to the scope and the idea of our project.
Ethical	Regarding ethics, we'll be sure to grow our plants in an environment in which they'll be likely to thrive and survive in
Health and Safety	We'll make sure our project will be up to standards with health and safety when creating our design. It'll be of utmost priority
Manufacturability	Our project will be able to be manufactured in with no issues at all due to the materials and parts we choose
Sustainability	Our project will be plenty sustainable and will be sure of that through rigorous testing.

Table 4 (Cont.)- Realistic Design Constraints

5.0 Research - MM

This section serves to present research and justification for the components and environmental conditions the indoor greenhouse system will need for optimal consumer operation. Biological plant needs will be discussed in detail in order to form electrical and computer engineering solutions for an ideal system. Plants that will thrive within the size constraints and engineering limitations will be discussed. Finally, the framework for the devices needed to correct the environmental conditions will be presented.

5.1 Plant Needs - MM

In order to target the environmental conditions needed for our greenhouse, we first need to understand the biological mechanisms needed for plant growth. The factors to be considered in this section are sunlight, hours of sunlight, watering and drainage cycles, fertilization requirements and indicators, etc. Identifying these needs will allow for the main control mechanisms to be

As seen in Table 5 below, vegetables are typically grouped into three categories: low, medium, and high light. As a general rule, the larger and more complex the fruit, the more sunlight needed to produce it. Foods in the lower light category largely include leafy greens as the edible portion, as less glucose is being produced. Plants that produce vegetation below the soil and near the root system (mostly tubers) require more energy than the leafy vegetation, but are comprised of simple starches, and therefore tend to need less sun exposure than vegetables that fruit above the soil and have more complex sugars.

Table 5- Light Requirements of Common Garden Vegetables

Category	Low Sunlight	Medium Sunlight	High Sunlight
Time Exposed	3-4 hours per day	4-6 hours per day	6-8 hours per day
Fruit/Vegetable	Swiss Chard	Beets	Peppers
	Cos Lettuce	Carrots	Tomatoes
	Lettuce	Potatoes	Watermelon
	Parsley	Broccoli	Okra
	Arugula	Radishes	Eggplant
	Asian Greens	Turnips	Strawberries

5.1.1.1 Light Intensity - JG

The range of UV light intensity varies for each type of plant. To provide optimal lighting for the plants in our device, we will need to measure the intensity of the UV light directly affecting the plants ability to grow indoors where natural sunlight might not be possible. UV light is measured in Lux, and can be used to determine how bright our grow lights should get when they're on. For plants that do well in the shade, 5-15 kLux will be fine. Plants that require direct sunlight will be comfortable with 15-40 kLux of intensity [4]. However, the required intensity for plants with high light requirements can reach up to 100 kLux. Consequently, the lights our system supports will directly impact the types of plants we can grow.

For WS2812B LEDs, the luminous intensity is given in the datasheet, and can be used to figure out the illuminance of each LED. To convert from candela to lux, we need the distance in meters to complete the following formula:

$$E_{v(lux)} = \frac{I_{v(cd)}}{d_{(m)}^2}$$

Table 6: WS2812B Light characteristics [5]

Emitting Color	Wavelength (nm)	Luminous intensity (mcd)	Illuminance (lux), d=10mm	Voltage (V)
Red	620-625	390-420	39-42	2.0-2.2
Green	522-525	660-720	66-72	3.0-3.4
Blue	465-467	465-467	46.5-46.7	3.0-3.4

5.1.2 Water & Drainage - MM

As noted in the previous sections, water is an integral component of the photosynthesis cycle. Water will travel up a plant’s stem to the leaves where it will then be exposed to sunlight, evaporate through small openings in the foliage - called stomata – and exchanged for carbon dioxide, which is then used to produce glucose (sugar) and oxygen. Water is also important to vegetation because it provides turgor, which is the water pressure inside the cells that make up a plant’s skeleton. Without enough turgor, a plant will not be able to support its structure, causing the leaves to wilt and droop, limiting the leaf area exposed to sunlight. Too much water can also be the detriment of vegetation, as plants require oxygen to escort nutrients and water up the root system and to the leaves. If this movement, called evapotranspiration, does not occur, water cannot evaporate from the stomata to trigger photosynthesis.

Another effect is that the root hairs, which are each a single modified plant cell, are unable to metabolize without oxygen and will eventually die. Root hairs are quite small in comparison to the mass of the root system and vitally important for the uptake of nutrients and water; unfortunately, their meager size makes them extremely sensitive and susceptible to rot. Although there may be enough water in and around the root environment, the root hairs have now died through suffocation and the plant will enter a period of stress due to internal drought.

The plant will attempt to reduce water loss through leaf curling and drooping, as the turgor is effectively lowered. Photosynthesis stops, the roots begin to rot, and the organism will slowly die unless the rot is removed, and proper drainage is introduced. Gardeners typically introduce rigid particulates such as sand or perlite to prevent the soil from becoming too compacted and encouraging aeration and drainage away from the roots. Overwatering is a main concern of potted plants as water tends to pool and stagnate at the bottom and cannot run off into earth below. Therefore, a proper run-off system for excess water is absolutely vital to any indoor garden.

Drainage and runoff solutions will need to be cleverly engineered as to protect the plant roots effectively while conserving precious resources such as space and available power. The drainage system employed in this project will make use of inclines and gravity to lead excess water to a separate holding tank for removal. A diagram of ideal soil conditions can be seen in Figure 4 below.

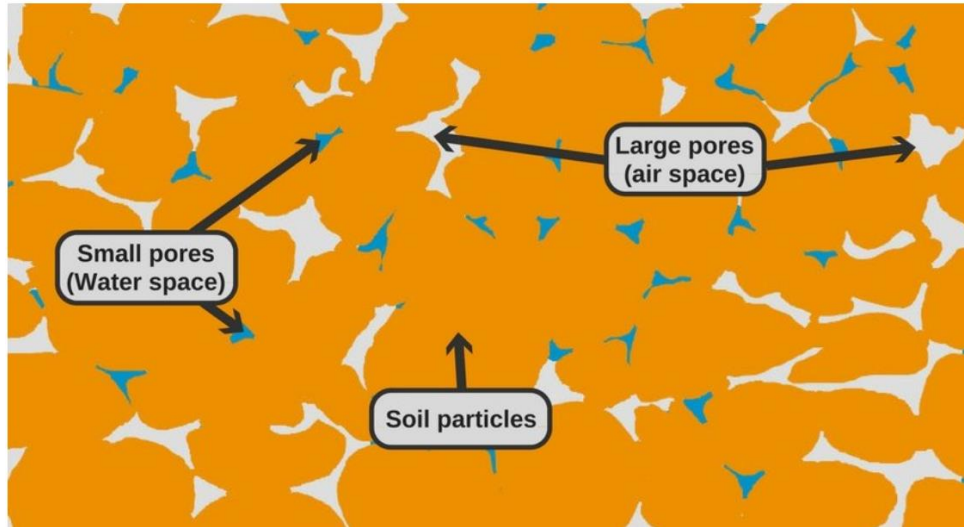


Figure 4- Proper soil composition for healthy roots. Permissions requested from ABC Science

5.1.3 Nutrition - MM

Most vegetation can survive early on with simply water and sunlight. However, if a plant does not have access to nutrients during the major growth stages, it will likely become a weak plant or simply die off early [6]. A lack of nutrients will also limit the organism's ability to breed and grow fruit. So, if a gardener's desire is to produce a bounty of edible vegetation, nutrients need to be introduced to the routine.

The primary nutrients are Nitrogen (N), Phosphorus (P), and Potassium (K); these are commonly referred to as the NPK nutrients. Nitrogen atoms are needed to produce amino acids, proteins, and enzymes. This encourages green, leafy growth and the formation of stems. Phosphorus is especially important during the process of seed germination, as it encourages the formation of root systems. Certain Phosphorus compounds can optimize the energy converted during photosynthesis and act as an immediate source of energy in all cells. Potassium promotes fruiting, flowering, and can increase disease resistance as well as general plant hardiness. It is necessary for water regulation, photosynthesis, respiration, and the efficient use of nitrogen. Potassium is also key for ample glucose and starch production.

Secondary nutrients and micronutrients are supplemental to the NPK nutrients. There are three secondary nutrients: Calcium, Magnesium, and Sulfur. Calcium is involved in some enzyme processes and helps to regulate the transport of other nutrients around the plant. Calcium pectate is also used to glue together cell walls. Magnesium is needed for healthy leaves and is needed to produce chlorophyll; its pectate also assists in gluing cell walls together. Sulfur atoms are used to produce select amino acids and vitamins. Micronutrients such as Boron, Copper, Iron, Manganese, Molybdenum, and Zinc help maintain healthy cell formation, assist in photosynthesis, and activate certain enzymes.

Luckily, plant nutrients are widely available on the market and come in a variety of forms. The difficulty arises in identifying deficiencies and administering supplements in a manner that does not overwhelm the plant. Water soluble NPK fertilizers can be heavily diluted and added to the watering routine- this method effectively delivers the nutrients directly and evenly dispersed to the roots while limiting the concentration exposed to the root hairs, protecting them from chemical

burn. In terms of identifying nutrient deficiencies, we must look to the impact of soil pH on a plant. In general, you can provide a plant with all the key nutrients, but the alkaline or acidic nature of the soil might be negating the effect or prevent them from being absorbed. A neutral pH range of 6.5-7.5 is generally accepted as the ideal condition for plant growth.

5.1.4 Temperature - MM

Temperature plays an important role in promoting seed germination and photosynthesis. In general, 77°F is considered the optimum temperature for photosynthesis. For greenhouse environments, the optimal range is extended to be a few degrees warmer, up to around 80°F. This is a few degrees above standard room temperature and would require supplemental control and power to provide a warmer environment.

Conditions for seed germination largely depend on the plant being grown, though ideal conditions typically surpass 80°F; however, germination is possible at lower temperatures- albeit with lower rates of success. In Table 7, a variety of vegetables are charted according to their practical and optimum temperatures for planting; the middle column indicates a point at which a 70% success rate was achieved, and the far-right column indicates the temperature at which the success rate approaches 100% [7].

Table 7- Practical and Optimal Temperatures for Germination

Vegetable	Practical Temperature for Planting	Optimal Temperature for Seed Germination
Beets	45°F	85°F
Carrots	45°F	80°F
Lettuce	45°F	75°F
Parsley	45°F	75°F
Radishes	45°F	85°F
Spinach	45°F	70°F
Turnip	50°F	85°F
Cabbage	53°F	85°F
Swiss Chard	54°F	85°F
Corn	55°F	96°F
Tomatoes	55°F	85°F
Cucumbers	65°F	96°F

Vegetable	Practical Temperature for Planting	Optimal Temperature for Seed Germination
Peppers	65°F	85°F
Cantaloupe	69°F	90°F
Squash	70°F	95°F
Beans	71°F	81°F
Watermelon	72°F	95°F
Okra	74°F	95°F
Eggplants	75°F	82°F
Pumpkin	75°F	96°F

As seen in Table 7 above, plant life can easily be supported at a realistic range of temperatures as long as the correct plant is chosen for a particular environment. Conveniently, this practical range is easily achieved in a room-temperature environment; ideal temperatures for an indoor greenhouse enclosure.

5.1.5 Humidity - MM

Plants need moisture in the air in order to thrive, as without it, water in the leaf's stomata will be evaporated before it can be exchanged for carbon dioxide during the process of photosynthesis. So, humidity is necessary to prevent the plant from losing this water, rather than actually supplying any to the plant. Humidity and temperature share a unique relationship and one should not be considered without accounting for the other, as the former is typically considered in terms of its relative humidity. Relative humidity is the ratio of the partial pressure of water vapor to the equilibrium vapor pressure of water at a given temperature [8].

This calculation provides context to the limits of water vapor in the air at various temperatures, as hot air can physically hold more moisture content than cooler air. This means that cooler air at the same level of absolute humidity will have a higher relative humidity than that of hot air. Relative humidity is widely used in the context of gardening because lower temperatures typically correspond with less water being able to evaporate out of the stomata. Ideally, the relative humidity of a greenhouse should be above 50%.

5.1.6 Airflow - MM

Plants need a balance between humidity, heating, and ventilation to prevent the development of molds, fungi, and viruses. Proper air circulation should allow for water vapor to be present within the environment without letting it condensate and stagnantly collect for long periods of time. This is especially important in a greenhouse environment, where the conditions for sustain plant life in

a closed environment can also grow harmful cultures that could bring disease to the entire enclosure. Proper ventilation will also provide an outside source of carbon dioxide for the plants to use during photosynthesis.

5.1.7 Overall Environmental Requirements - MM

By gaining a fundamental understanding of what plant life needs to thrive, it is now possible to identify the factors and environmental conditions we need to provide in the greenhouse via sensors, devices, and electronic controls. The greenhouse needs to provide the conditions shown below in Table 8. Nearly all mechanisms will rely on a microcontroller receiving sensor inputs and initiating the appropriate controls response. Timers will also be employed to trigger controls activity for conditions that are difficult or redundant to monitor, but nevertheless required for ideal operation of the indoor greenhouse system. Routines, methods of control, and additional device data will be presented within the contents of this document.

Table 8- Overall Environmental Requirements

Plant Need	Provision	Engineering System
Sunlight	Radiant energy light source with either a full spectrum of light or red and blue wavelengths	LED strip lights of blue and red wavelengths on a timed schedule, 6-8 hours per day
Water	A watering system that fully waters plants only when the soil is dry and has a means of collecting and removing excess water from the soil to prevent root rot	Pump and tubing that delivers water to plant beds when moisture sensor indicates dry soil; runoff system that collects water prior to drainage
Nutrients	A system that allows for the delivery liquid NPK nutrients on a regular schedule	Accessible water tank for adding liquid nutrients to standard watering system
Balanced pH	Monitoring of soil pH to indicate nutrient absorption	pH sensor that communicates levels to be displayed to the user
Humidity	Relative humidity monitoring and maintenance of at least 50%	Small diffuser that activates upon low hygrometer measurements
Airflow	A ventilation system for providing carbon dioxide and preventing disease	Fans that operate on a regular basis and upon high hygrometer measurements

5.2 Types of Plants - JG

To make our device more appealing to the general public, we would like to be able to grow a wide variety of plants that aren't too meticulous to keep track of. After all, we're engineers, not gardeners. However, we would like our device to have capabilities that can extend to experienced, capable gardeners that would like to use our product to grow more intensive kinds of plants.

Basil is a great choice for first time gardeners. A well-drained plant can produce a cup of basil every two weeks. It only requires 4-6 hours of sunlight, and minimal space (2 per square foot). It is weak to the cold less than 50°F, but that won't be a big problem since our smart garden is intended to stay indoors [9]. Basil is sturdy enough to be transplanted to another pot and do fine. It also has a wide pH range, between 5.1 and 8.5 [10].

More ambitious gardeners might be looking to grow something a little more complex than just herbs. With our smart garden you will also be able to grow certain fruits, for example strawberries. Strawberries need 6 hours of sunlight a day and have a shallow root system, about 4-5 inches of length, which makes planting strawberries in our smart garden something even a novice gardener can accomplish. Strawberries require a soil pH level of 5.6-6.3 and once flowering they should be fertilized every ten days, until your finished harvesting.

Another plant to grow in our smart garden is parsley. Seeds are planted 1 to 2 inches apart and can grow to be 1 to 2 feet tall. To mimic 1 to 2 inches of rain per week, parsley might need to be watered more intensively than other plants, thus our device needs to handle the different rates of watering for each plant. Soil moisture sensors will make sure the soil never runs dry by alerting the watering system if the moisture ever drops below a threshold.

By knowing certain conditions that each plant grows most efficiently in, we can monitor for those certain conditions with sensors for data such as humidity, temperature, pH, and light exposure. The more sensors we have, the better we make predictions on how to help the plant grow. However, more sensors mean a bigger financial cost, which is a constraint we must worry about. Table 9 shows the optimal readings these sensors would be looking for in the plants mentioned above. It will be ideal to pick plants that tolerate the same pH levels so that the planting bed has an even spread of nutrients [11]. To keep the watering system design simple, it will be best to pick a variety of plants that share the same soil moisture levels, since then watering can be done for the whole plant bed at once, rather than at different times for each plant. Having the ability to water each plant individually would be a great advancement to make but given the project scope for the timeline we have, it seems out of reach to have in our final device.

Table 9: Plant specifications [12]

Plant	Soil Moisture	Soil pH
Basil	40-80%	6.1-7.0
Strawberries	20-60%	6.1-7.0
Parsley	40-80%	6.1-7.0

5.3 Sensors - DM

For this project, we're going to be using multiple sensors that will be used to monitor everything that we're going to be needing to keep a watchful eye on to make sure that everything is being done properly. We have considered the specifications of these sensors and how useful they will be for us regarding this specific project that we're working on which is the indoor greenhouse project. Stability is an important factor as well when looking at these sensors. We need to make sure that our system remains stable when everything is connected and running. We will attempt to verify stability and functionality of these sensors via a prototype before we attempt to make the final build of the project. When constructing the prototype, we'll make sure to verify the specifications and test whether the functionality of the sensors that we order are functioning properly. These sensors will be monitoring a variety of things such as temperature, humidity, water level, and soil moisture. Making sure that all these things are being monitored and recorded for data is imperative for our project to ensure that the plants that we're going to be growing will be thriving in the most optimal way. Since we're dealing with living plants, careful monitoring is necessary with these sensors.

Given our project being an indoor greenhouse, sensors such as these are important to have. Selection of our sensors was also something that was carefully done to make sure that we try to make our project as optimal as possible. We factored in many things for our selection, primary concerns being cost as well as accuracy. There is an interesting reason as to why the DHT22 that we have selected is needed to measure both temperature and humidity. Air that is at a cooler temperature holds less water than air that is warm even though they may be reading at very similar humidity levels. Even though humidity is the same, if the temperature is different, the amount of moisture that can be picked up can also differ if the temperature is warm or cold. We're going to remedy the temperature issue of going cold by attempting to keep the temperature as consistent as possible.

Due to the nature of the project being a tabletop greenhouse, temperature fluctuations should not be too common unless under extreme circumstances which may cause the tabletop to not function altogether. One such extreme circumstance would be a power outage. If a power outage were to occur that would naturally cause everything to stop functioning within the project such as the DHT22 humidity sensor, high sensitivity water sensor, and our LM393 soil moisture sensor. No power would be sent to the project since it would be connected to the power outlet. Then, there would be rising temperatures within the tabletop greenhouse due to the lack of air conditioning within the environment outside of the greenhouse. Though this may not be an issue for a greenhouse, but for our purposes it is an issue since our smart tabletop greenhouse will not be functioning regardless.

5.3.1 DHT22 Temperature and Humidity Sensor - DM

This sensor measures temperature as well as humidity. It will be utilizing a thermistor as well as a capacitive sensor for humidity by reading the data observed by the air around it [13]. After getting this data, the data pin will be receiving this information for the user to observe. This sensor was selected for multiple reasons. One of the reasons is that the DHT22 is not that expensive and can be considered fairly cost efficient. Data on this sensor is refreshed every 2 seconds which is fine for our current project, we do not need data updates to occur too often regarding temperature and humidity, we are just looking for accuracy in that regard. This sensor will be connected to a power of 3 to 5 volts which is the power that we'll be considering for this project regarding the sensors

and microcontroller. When it comes to readings regarding humidity, there are accuracy variations of 2-5%, whereas the temperature readings vary in accuracy from around $\pm 0.5^{\circ}\text{C}$. For our project purposes the variations ranging from these percentages could be negligible. Being off by a couple of percent will not be detrimental to our project in anyway. The only way something like this can affect how our project is going to operate is if the percentages vary well over a mark of over 10-15% which we're way far off from.

Sampling rate is more than enough with this part whereas we are updated with new data every 2 seconds with sampling rate of 0.5 Hz. There was a previous version of this part that we were considering which was the DHT11 but considering the difference in costs and the benefits we get out of the cost difference on the DHT22 we had to go with the one that was more accurate and precise, being the DHT22 Temperature and Humidity Sensor. There are 4 pins on the DHT22 for VDD, signal, null, and ground. When it comes to communicating with the microcontroller that we'll be using, there will approximately be about a 5ms time delay per communication with the microcontroller. If we take any sort of lag into consideration, there wouldn't be too much to speak about when it comes to the DHT22 since a 5ms time delay is extremely negligible. Plus, we wouldn't need too many updates on the temperature and humidity regardless. Though it is nice for us to receive constant updates on the Temperature and Humidity of our greenhouse.

5.3.2 High Sensitivity Water Sensor – Red Version - DM

We will be using the High Sensitivity Water Sensor for our specific project. When deciding on this sensor, we considered the cost as well as durability. We need something that will last for a project of our scope, so we considered other sensors and their specifications but for our needs and purposes the High Sensitivity Water Sensor will be enough in terms of cost and durability. We will be using this sensor to monitor water levels within our indoor miniature greenhouse. Basic sensor which will be using parallel wires to get a reading on the water level of our miniature greenhouse [14].

This water level sensor will mainly be used to keep a measure on the water tank that we will be utilizing in this project. We do not want the water to go beyond a certain level, which is why we'll be using this sensor to keep a watchful eye on it. This sensor is enough for our purposes as well given its specifications of having a working operation of 3.3-5 volts as well as working in temperatures between 10°C - 30°C [14]. Given the operating voltage as well as operating temperature, we should be able to use this specific water sensor for our project. There are 3 pins on the sensor for the signal, ground and VCC respectively. Since this water sensor is capacitive in nature it will be placed around the water tank to make sure we always have a reading on our water levels.

5.3.3 LM393 Soil Moisture Sensor - DM

For us to monitor when we will need to give our plants more water or to see if we are giving our plants more water than they need, we will need to incorporate a soil moisture sensor in our project. It would be easy to make a simple error regarding the watering of the plants that we're going to be growing so we're going to need to be extra cautious by adding the soil moisture sensor so that we can avoid this issue entirely.

We will be utilizing the LM393 soil moisture sensor to do this. This sensor is capacitive in nature, so this sensor will have two probes that will be inserted into the soil in our miniature greenhouse to keep track of the moisture in the soil. An example of how this will work can be seen.

Of course, given that we are building a miniature greenhouse, we'll be using multiple soil moisture sensors to monitor different varieties of plants that we'll be growing. Specifications seem reasonable and within the scope of our project, having an operation of 0.3-5 volts. Operating temperature for the LM393 is between 0-70°C which is well within the range of the scope of our project [15]. Our project will be an indoor greenhouse so there's no way the temperature will dip below 0°C and there's no way it'll rise above 70°C. There are exceptions to when temperature could rise in the tabletop greenhouse and that's if there was poor temperature regulation in the room surrounding the tabletop greenhouse. This would be due to air conditioning failure or if the fans installed for cooling and airflow regulation stopped working. Though we're going to prevent this from occurring to the smart tabletop greenhouse by plugging it into a powerful surge protector with a large backup battery that can continuously support our project and prevent it from shutting down. This way we'll be able to at least keep everything functioning with regards to the greenhouse at least until the surrounding temperature regulates itself again.

5.3.4 pH Sensors – JG/DM

Having a pH sensor will help us monitor data about the nutrient levels of the soil. Plants that require more organic material will grow better in a lower pH setting. Generally, most plants do well in the 6-7 pH level.

The Ezo-pH embedded pH circuit is an affordable sensor that can send pH reading levels over UART and I2C in ASCII format. The circuit comes with a probe that measures the hydrogen ion activity in a substrate. A glass membrane at the tip of the probe permits the hydrogen ions from the surrounding soil being measured to diffuse into the outer layer of the glass. Larger ions remain in the soil, creating a potential difference across the probe, resulting in a small current proportional to the concentration of hydrogen ions in the soil. The Ezo-pH sensor comes with three different pH level solutions to use for calibrating the tool. There's also example code online for the device provided by the manufacturer to help us get a demo set up quickly [16]. When it comes to the usage of the Ezo-PH sensor, there are also very well created documentation to support this product. All the information that we need to make sure that our project will be able to support this peripheral is there for us when we need to reference it.

Another pH sensor is the Go Direct Tris-Compatible pH sensor. This sensor is marketed towards consumers aiming to measure the pH of semi-solid material such as soil or food. This is important to note because there are other pH sensors that exist just for measuring the pH of liquids and would give inaccuracies while trying to get any sort of readings from a semi-solid. There are pH ranges that go from 0-14. Also, the specifications for this sensor range from 0-100 degrees Celsius which is well within our range for our specific project. Response times for readings from this pH sensor are a full response every 30 seconds at a temperature reading of 25 degrees Celsius. Also, at those readings we have a large margin regarding temperature readings. The impedance of this pH sensor is approximated to be about 20kΩ also being read at the temperature reading of 25 degrees Celsius. Regarding the electrode type of the pH sensor, it is a double-junction and has a polycarbonate body. Also, the style of the membrane of the pH sensor is completely made from flat glass. Also, for this pH sensor, there are also errors that may be picked up due to sodium causing the range of

the pH go over 12. Even though the range says 0-14 for pH, more than likely there are inconsistencies with the sodium causing the reading to be read over a pH level of 12.

5.4 Microcontrollers -AL

For this particular project, an Arduino or a Raspberry PI would have easily been a surefire way to have a microprocessor to handle all the functions that we needed, but we decided to try to appeal more to the skills that corporate companies would desire rather than hobbies skills. Thus, we decided to search for microcontrollers instead that would be able to micromanage the functions that we need on a smaller scale. On the market currently, there are were so many MCUs that we could choose from, but out of all of them there were not many affordable ones that could handle all the function that we were planning to implement. For our project, it was planned to use too many pins for a single microcontroller, thus we came with the idea to have a central parent microcontroller that would handle the main processing, receiving and delivering of data along with controlling the user interface while a having smaller, less intensive microcontrollers to handle all the peripheral controls i.e. water pump, temperature sensor, pH sensors, etc.

5.4.1 Main MCU – AL/JG

For the main parent microcontroller, we came to the consensus that we should use an ARM processor. This is because it would be easy to purchase and test these since Texas Instruments has a plethora of launchpads with an ARM processor already embedded within the development boards along with different kinds of expansion backpacks to add functionality to them. The MCU we chose for this project is the MSP432P401R. Its core contains an ARM 32-bit CPU that operates up to 48 MHz. It has up to 24 analog input channels and up to 48 I/O pins with interrupt capability, so we'll have plenty of pins to use to control the system. We can use up to four 16-bit timers, each with five options for capture, comparison, and PWM capability. These will be used top drive the fans, LEDs and provide clocks for the WiFi and Bluetooth modules. Two 32-bit timers are provided for interrupts, which will be helpful for programming when to gather data from sensors and for enabling low power mode. The ultra-low power modes for the MSP432 chip are very efficient. In active mode, the chip draws about 80uA/MHz. in low power modes, the chip will only draw 660 nA, but can shut down even more to draw only 25 nA. the chip operates at a wide supply voltage range, but we'll be using the standard 3.3 V. The chip stores up to 256 KB of flsh memory, capable of simultaneous reading and execution. There's also up to 64KB of SRAM, and 32KB of ROM for peripheral driver libraries [17]. To program the cchip, there are the 4-pin JTAG and 2-pin serial wire debug (SWD) interfaces. Serial communication is supported with four eUSCI_A modules for UART and SPI (up to 16 Mbps) and four eUSCI_B modules with SPI and I2C.

5.4.2 WiFi Controller – AL/JG

To do some initial testing on, we purchased the MSP430F5529 Launchpad from TI along with the CC3100 WiFi Booster Pack to experiment with the idea of a functionality of having our *Smart Greenhouse* connect to the internet. The CC31XXEMUBOOST emulator board was needed to flash settings to the WiFi booster pack. We choose to experiment with this device since there are many resources for us to find if we ever need assistance in implementing our ideas. The CC3100

network processor chip comes with a dedicated ARM MCU that splits WiFi and internet protocols from peripheral microcontroller operations. It comes pre-equipped with WiFi drivers, but to install updated ones, the emulator is needed. The chip will interface with the MSP432 over SPI or UART. It has a low memory footprint, requiring less than 7KB of code and only 700 B or RAM to run the TCP client application. The chip can operate at 3.3 V, and has a variety of low power modes. When connected and idle, the chip draws about 690 uA, but can be put into deep sleep for a draw of 115 uA, or hibernate down to 4 uA. When receiving data, the device will draw around 53 mA. Likewise, the device will require more current for transmission at around 223 mA. The clock source can come from the master MCU or a dedicated crystal at 40 MHz or 32.768 kHz.

5.4.3 Bluetooth Controller – AL/JG

For other testing, we also decided on purchasing a MSP432P401R Launchpad along with a CC2564MABOOST module backpack to experiment with the functionality of having a star model to connect multiple Bluetooth modules to a single, parent module which will oversee all the operations of the peripheral sensors. As to why we choose these evaluation boards, there are many applications of them being used in similar way all over the engineering community. To find and optimize other engineers’ solution to similar problems that arise in the future would be easy enough. The CC2564MABOOST also has dual-mode Bluetooth and Bluetooth low energy to switch back and forth as to optimize power usage, a certified and royalty-free TI Bluetooth stack, demos, 4-wire or 3-wire UART and PCM/I2S Interfaces so we can send different kinds of data packets as we see fit. The CC2564B chip controlling the Bluetooth module will operate at 3.3 V, and also has low power optimizations for current consumption. In active mode, the chip draws about 1 mA, and spikes to around 110 mA during transmission. In sleep mode, current draw drops to 105 uA, and 7 uA in shutdown mode. We will need a copper antenna on the PCB we build for the chip to use for radio communication.

5.4.4 Auxiliary Microcontrollers (Sensor Nodes) - JG

The child nodes that will be collecting sensor data from the plants will not need to be as complex as the master ARM microcontroller. For the sensor nodes, the MSP430G2553 will be enough to incorporate up to 3 sensors and Bluetooth capability. The MSP430G2553 makes up for its lack of processing power with its low power mode capabilities. 95% of the time, the MCU will be shut off, only to wake up to take periodic sensor readings. This will allow the Bluetooth node devices to have long lasting battery life. As shown in the following table, the low power modes (LPM) of the MSP430 drastically reduce the amount of current draw in the device.

Table 10: Summary of MSP430 Low power modes

Mode	DCO	MCLK	SMCLK	ACLK	Current (Amps)
Active 3.3V	Up to 16 MHz	ON	ON	32 kHz crystal	4 mA

Mode	DCO	MCLK	SMCLK	ACLK	Current (Amps)
Active 1.8V	Up to 1 MHz	ON	ON	32 kHz crystal	200 uA
LPM0	Up to 1 MHz	OFF	ON	32 kHz crystal	80 uA
LPM3	OFF	OFF	OFF	32 kHz Crystal/VLO	1 uA/0.5 uA
LPM4	OFF	OFF	OFF	OFF	0.1 uA

By using LPM3, the MSP430 can use ACLK to time interrupts of when to make sensor readings. Keeping our microcontroller in low power mode for most of its operation will take advantage of the low power efficiency of the sensor nodes. Fortunately, Bluetooth technology has become optimized for low power consumption as well.

5.4.5 Auxiliary Bluetooth Modules - JG

To connect the master MCU with the child modules, we'll use the CC2541 BLE chip equipped with serial communication over I2C. The schematics for the Bluetooth module will mimic the HM-10 wireless Bluetooth board that is a popular module for use with low power microcontrollers. The Bluetooth module will operate at 3.3V and will run at 32MHz. To communicate with the MSP430 controlling the sensor child node, there will need to be two pins for UART transmit and receive. A state pin showing the connection status could be useful for lighting an LED on the device to show that it is connected [18]. There are many fakes of the HM-10 for sale, so we'll need to be careful about purchasing from a trusted vendor. Though we'll be integrating the CC2541 with the PCB for our sensor modules, it will be worth getting a premade HM-10 board to test out to understand how it works first.

To illuminate the low energy capabilities of BLE compared to classic Bluetooth, we can measure the current draw of the device. When the HM-10 is paired with a device and operating, it draws around 9.1 mA. When the it is disconnected but still active, it draws around 9 mA. Compared to older models such as the HC-05 module that consumes around 20 mA while operating, BLE cuts current draw in half. The biggest advantage of BLE comes when using the sleep mode, where the HM-10 will only consume around 120 uA, opposed to the HC-05 low power state that draws about 2 mA. A coin cell battery rated for 240 mAh would then be capable of powering the Bluetooth module for 2000 hours, or around 83 days [19]. The HM-10 can be woken up by sending it a string or at least 80 characters or more, then put to sleep by issuing an sleep command. In BLE, the connection link is periodically acknowledged to respond to wake up events. This is handled by the CC2541 internal stack and already optimized for the lowest possible power consumption [20].

5.5 Communication - JG

Since this is a 2019 project, we want to implement the newest wireless communication features into this project. Skills using WiFi and Bluetooth technology are in high demand in the tech industry. Wireless communication also eliminates constraints wired devices have and opens up new possible solutions using the wide variety of applications available on the internet. While communication may be done wirelessly, many of the familiar protocols such as UART, I2C, and SPI are used by wireless technologies and should be understood by our group if we want to venture deeper into the ocean of wireless connectivity.

5.5.1 UART - JG

Universal Asynchronous Receiver Transmitter, or UART, is a type of serial I/O protocol that is supported by most microcontrollers. UART is used to communicate with peripheral devices as well as computers. UART is asynchronous since the transmitter and receiver each have its own clock signal. With two wires, UART can be implemented as bidirectional simultaneous transmission, known as a full duplex. The transmission line begins idle at high, then drops low for a start bit. After that, data is usually transmitted starting with the least significant bit (LSB) followed by a high stop bit at the end of transmission. The baud rate we set defines the bit duration, which is the clock rate the transmitter is operating on. Popular baud rates include 9600 for 16-bit controllers, or 115200 for 32-bit. Parity bits are useful for detecting errors in transmissions, typically reserved for wireless communication since data loss is more common. Another popular protocol in wireless communication is flow control, which manages the throughput of data. For a crowded network with devices all transmitting at the same time, it's good to have flow control to ease the pace of data flow.

Table 11- UART Configuration

Parameter	Meaning	Popular Configuration
Baud Rate	Transmission speed	9600
Data Size	Number of bits transmitted	8-bit
First bit	Most or least significant first	LSB
Parity	A bit to detect errors	None
Stop bit	End signal	1-bit
Flow Control	Prevents congestion of data	None

5.5.2 I2C - JG

Inter-Integrated Circuit (I2C) communication is based on a bus topology. Two wires for Serial Data (SDA) and Serial Clock (SCL) are used to provide communication between the master MCU, various sensors, memory, and other modules we add to the circuit. The master MCU controls reading and writing to I2C connected components and also drives the clock. The two wires of I2C are pulled-up to high by default. Devices pull wires low to transmit a 0.

5.5.3 SPI - JG

Serial Peripheral interface (SPI) is another common communication protocol where a master MCU communicates with one or more devices. Two data wires between the master and device make a full duplex. The protocol is synchronous in that the master generates a specific clock signal on a third wire that must match both devices. A fourth wire is used as the chip select signal, which is activated by the master to allow a device to communicate. SPI is not officially considered a standard, therefore SPI technology may differ between common devices. An LCD display can support most SPI protocol to display data.

5.5.4 Wireless connectivity (WiFi) - JG

WiFi networks are made up of clients, such as laptops, phones, or internet connected microcontrollers. Access Points (AP) connect to the wired network, and all communication done wirelessly between clients goes through the access point. Transmission on 802.11 wireless LAN networks gets complicated due to the physical conditions of the environment. Radio signals on these frequencies can bounce off solid objects and create echoes that can cancel or reinforce each other. This causes the received signal to be slightly different than the signal transmitted. Since wireless signals get broadcasted openly, there is a concern for proper security measures to be implemented. WiFi can usually be accessed up to 100 meters and have a throughput of up to 54Mbps.

5.5.5 HTTP - JG

Hypertext Transfer Protocol (HTTP) is an application-level protocol used for distributing data across the internet. In HTTP, a client sends a request to a server in the form of a request method, Uniform resource Identifier (URI), and protocol version followed by a message containing data. The http scheme is used to locate network resources with HTTP protocol. The http syntax identifies a host, port, and path. For IoT applications, HTTP lacks scalability. MQTT is preferred for being able to listen to messages and distributing small packets of information in large volumes. These aren't issues that we might run into with our smart garden, but it might be best to model our communication based on IoT preferred methods.

5.5.6 Bluetooth – AL/JG

Bluetooth is a wireless personal area network (WPAN) protocol designed to eliminate the need for wires in the physical layer of the communication stack. Bluetooth operates at the 2.4GHz and uses adaptive frequency hopping to avoid interference with other non-hopping communication network devices such as WiFi or ZigBee. Bluetooth works in a star topology, with one master node in the center of the network and one or more slave nodes connected to the master. The master node provides packet exchange, reference clock time, and the frequency hopping spread spectrum pattern. Slave nodes are connected to the master and synchronized with its clock and frequency hopping scheme. The slave nodes must go through the master to communicate. One master node can interconnect with up to 7 slave nodes.

Operations include Classic Bluetooth and Low Energy. Classic Bluetooth operates with standards set before Bluetooth 4.0. Low Energy (LE) Bluetooth protocol was newly introduced in 4.0 standards. Bluetooth classic supports a total of 79 channels, each given 1 MHz of bandwidth. The total bandwidth of Bluetooth devices is between 2.4-2.4835 GHz on the ISM band. It also uses a time division duplex scheme. Modulation modes include gaussian frequency shift keying (GFSK) or differential phase shift keying. Bluetooth LE systems give channels 2 MHz of bandwidth for a total of 40 channels. BLE operations are based on TDMA and FDMA multiple access schemes. Bluetooth 5 supports error correction coding, resulting in a lower data rate but a more secure and reliable connection. BLE devices transmit data on a specific time duration called an event.

5.5.7 Zigbee - JG

Zigbee 802.15 defines a protocol for mesh communication between devices using low power. Zigbee allows up to 250kbps data transfer rate and is operable up to 1KM. Instead of connecting to ethernet using an access point like WiFi, it uses a gateway which provides connectivity from the distributed nodes back to the ethernet source. Each node is equipped with a wireless antenna and communicate with the gateway over Zigbee. The nodes can also serve as routers to extend the range of connectivity, also known as mesh networking. The mesh connects all the nodes to the network and provides multiple pathways to the gateway. Even if one node in the network were to go offline, another route around the mesh allows the devices to stay active. Since our product is meant for a small indoor garden, we don't necessarily need a mesh network to extend connectivity range. Zigbee operates at the 2.4 GHz ISM band, so it would be a good solution to modifying this tabletop garden IoT system into a large-scale smart farm with more plants than Bluetooth could handle.

5.5.8 Sub-1GHz Transceivers - JG

For home connectivity, there are some benefits to using sub-1GHz signals for transmitting data. Sub-1GHz technologies offer advantages in range, power consumption, and reliability. For a transmitter T and receiver R a distance d away from one another transmitting at a frequency f , the received power P_R of a signal is proportional to the transmission power P_T divided by $d^2 * f^2$. For the signal to be received properly, the power received must be greater than a defined sensitivity limit.

$$P_R \propto \frac{P_T}{d^2 f^2} > \text{sensitivity limit}$$

Thus, we can achieve longer range with the same transmission power if we reduce the frequency. Because of their longer wavelengths, lower frequencies are better at traveling through obstacles and around walls in a home. We can also achieve a larger distance by designing a more sensitive sensor. There's less bandwidth to cover with lower frequencies, and less noise at those frequencies since 2.4GHz is often crowded with other nearby WiFi and Bluetooth devices. So we can get more precise with the filtered analog signal we receive, lowering the sensitivity limit. To achieve the same distance with a 2.4GHz signal, another node can be added to forward the signal, but this increases the cost and complexity of the system. If our smart garden project needed to be suited to connecting to an assortment of smart home devices, Sub-1GHz would be a better option. However, since the plant sensors will be in close proximity to the garden, they won't need the extended range benefits of Sub-1GHz.

5.5.9 IoT - JG

The Internet of Things (IoT) is a highly growing topic in industry, as well as academia. IoT refers to the rapidly growing network of devices that communicate over the internet to utilize the advantages of being connected online. This emerging field of electronics allows embedded devices to gather data from their environment using sensors and publish that data to an online database where further analytics can be done. From there, the data can be presented back to the user in an intelligent fashion that allows them to monitor any of their device remotely. With the rise of internet-connected devices estimated to approach 24 billion by 2020, IoT technologies are soon to become the standard for creating smart systems across the globe.

IoT has protocols at different layers of the network stack to make its magic happen. The ecosystem in IoT is commonly presented in a 7-layer model, as in *Internet of Things Protocols and Standards* by Tara Salman and Raj Jain. At the bottom, we have our market domain. This may be a smart home which has its devices connected. These devices would make up the second layer and might include a door camera, temperature sensors, or light control sensors. The third layer consists of the various ways the data from the devices can be communicated across the internet.

WiFi can be used to stream the Door Camera using UDP, Bluetooth can connect the temperature sensors to the thermostat and fans in the house, and NFC can be used to control the lighting with the touch of a phone. Next in the fourth layer we have our database, where we integrate the data gathered from sensors with previous sensor data, weather data, and cost tracking that we gather with other resources. The data gets passed onto the fifth layer, where further processing can be done. Computer vision analysis on the door camera data can detect and identify who is at the door. Predictive analytics that combine temperature data with recent weather reports and monthly electricity spending bills can determine the optimal cost-efficient way to regulate a home's internal temperature. The sixth layer contains the software that allows the smart IoT system to function through software like an App or Android Things. The smart home can also utilize Amazon Web Services (AWS) or Google's IoT Core to access the cloud, which can support all of layers below it. At the top of the stack, the seventh layer defines the service that enables the smart home, in this case energy management. Alongside the system are security and management applications which come with accessing the cloud.



Figure 5- IoT Ecosystem

5.5.10 Publishing Messages - JG

With the growing demand for “smart” devices, we want our tabletop garden to connect to resources and tools online in the cloud. Our smart garden will be fitted with a WIFI module that can communicate through HTTP or MQTT. This will allow us to work with the newest cloud functionalities available since they’re highly efficient protocols for IoT communication. Our first goal will be getting our microcontroller to host a server in order to send data to and from the microcontroller. MQTT is the most desirable protocol since it operates on a publish/subscription framework. Unlike a client-server architecture, in which the devices communicate directly, the publish/subscription method allows sensors to publish messages for an MQTT broker to send to devices subscribed to the incoming messages. This will make it easy to publish sensor data from our garden to the cloud, where a web server subscribed to our sensors can pick up on the data. Likewise, we can control our device from a web server by publishing tasks that our device will be subscribed to, such as turning on a water pump to water the plants even when on a computer away from home.

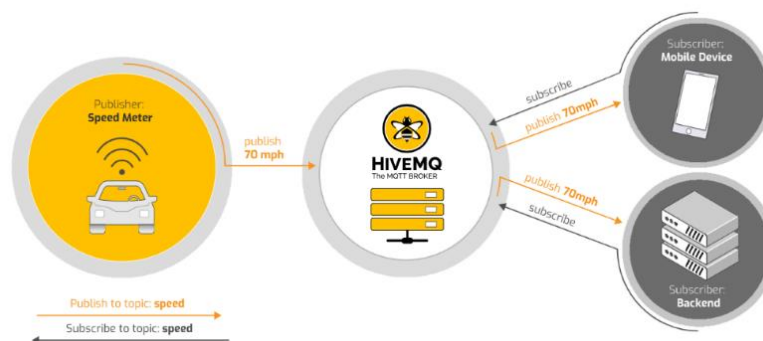


Figure 6- How the pub/sub MQTT mechanism works

5.5.11 PubNub - JG

One way that we can communicate over a publish/subscribe based network is with PubNub. PubNub delivers the key building blocks for needed for real time, secure connectivity. Data gets published by our sensors, and the devices subscribed to that data will receive that information in real time [21].

The packets of data are sent in messages that get published to channels in PubNub. They can contain any serializable data including objects, arrays, numbers, and strings. Strings may contain any UTF-8 character, single-byte or multibyte and single message sizes can scale up to 32KB. It is recommended that the message gets formatted in JavaScript Object Notation (JSON), a simple data-interchange format that is easy for humans and computers to understand. PubNub provides SDKs that will turn outgoing JSON objects into strings for serial communication over the PubNub network.

To organize incoming data, messages that are sent into PubNub are placed on a channel. Devices subscribed to the channel quickly receive messages published to it. Unlike older messaging system designs, PubNub allows our device to subscribe to an unlimited number of channels through a single connection, simultaneously. So, each sensor we add can have its own channel, our master MCU can be subscribed to the group of sensor channels, and our global web application can communicate over all the channels. Each channel is identified by its unique key set for publishing and subscribing. PubNub limits channel and channel group names to UTF-8 compatible characters and limit their size to 92 characters in length.

While our data is in midstream, we can deploy serverless PubNub functions. Common features with pre-built blocks of code include filtering, language translation, third-party API integrations, and other analysis to enhance our projects cloud connectivity features. PubNub provides a GUI for monitoring and creating these functions, with documentation for how to set them up.

Channels in PubNub are divided into four general topologies: Unicast, Broadcast, Multicast, or consolidation. Each topology has multiple ways of being implemented. For unicast, or one-to-one connections, there are three ways of implementation. A simple solution is to have device A and device B publish and subscribe to a common channel, like “devA_devB-ch”. This works but might not be secure enough for the user. Another way is to have each device subscribe to their own private channels but publish to each other’s channel. For example, device A will subscribe to “devA-ch” and publish to “devB-ch”, and likewise device B will subscribe to “devB-ch” and publish to “devA-ch”. A third solution can implement both ways of connectivity, where both devices publish to each other’s channel, as well as a global channel for easy history retrieval. This third method also makes it easier to divide up authorization roles.

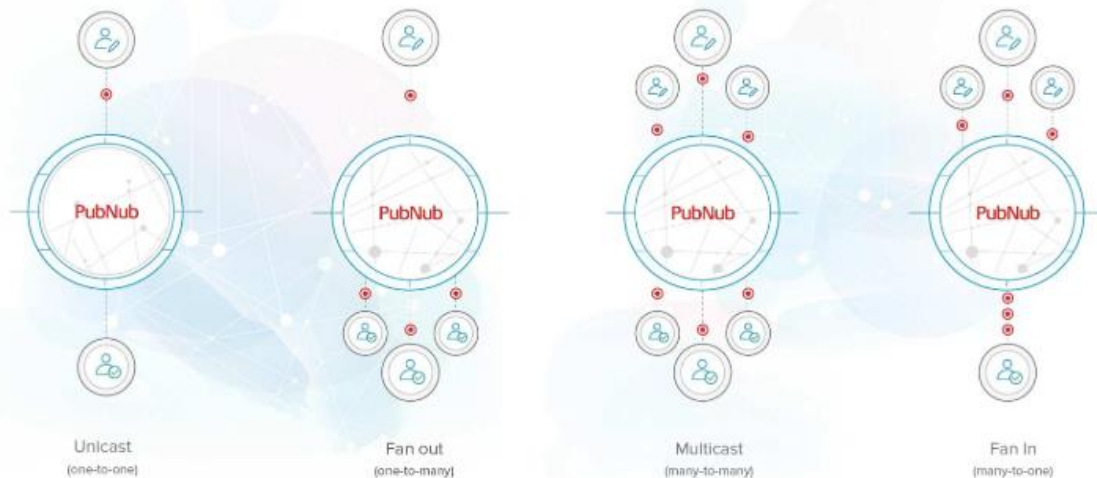


Figure 7- PubNub channel topologies

Channel groups in PubNub allow for a user to subscribe to and receive data from numerous channels very easily. Depending on the implementation for our smart garden, having channel groups might be useful. For instance, if each plant in are garden are equipped with a group of monitoring sensors, it might be useful to assign a channel to each plant and form a channel group with them. This way our sensors can publish data to the respective channel for the plant they are monitoring, and our web server can receive the information from all our plants by being subscribed to the channel group.

PubNub also enables us to store the last known state of sensor readings through presence. A “presence” channel is made for each channel we create and will return specific presence events to applications subscribed to them. Presence events can return state change messages that can be used to alert a user about a sensor going off. Webhooks are supported for presence events, so when a state change is detected, the webhook can be configured to get the new values of our sensor or alert the user in some type of way.

Security is an important aspect of embedded systems. PubNub has security measures that will keep data safe. At the legislative level, PubNub is HIPAA compliant, meaning our connection is secure enough to even send medical data. It is also EU-US Privacy Shield compliant, meaning our messages aren’t getting stored somewhere unsafe. PubNub helps eliminate attacks with data center routing and intelligent Transport Layer Security (TLS) encryption. Point-to-point encryption is provided by the PubNub SDK libraries. To turn on these extra security measures, we enable a TLS parameter passed into the instance of a PubNub object when we first initialize it.

To visualize data gathered and transmitted with PubNub, we can use Freeboard.io. Freeboard provides ridiculously simple dashboard GUIs for our data. We can add different types of widgets such as line graphs, gauge dials, text, and indicator lights to represent our data. The widgets are set up to parse incoming JSON files from the channel it is subscribed to. To link channels as a data source in Freeboard, we simply provide the channel name, PubNub subscription key, and select PubNub as the type.

5.6 Communication Design - JG

There are several ways for us to design the communication system for our project, but we must do the research to figure out which one will be the most fruitful based on the timeline we have and the abilities we have as engineers.

5.6.1 Developing in CCS - JG

Using CCS, we will need to set up the proper development environment. The CCS App center and resource explorer allows us to download the correct tools and drivers needed for the project. Communicating over MQTT will be the best method for transporting data from the microcontroller to the cloud, so we will first need to set up MQTT on the device. To support multiple thread functionality, an operating system must be used. The TI-RTOS is available for download through the resource explorer. After downloading and installing the TI-RTOS SDKs, or other real-time operating system drivers, we can download the MQTT demo to get started. There will most likely be bugs to resolve when dealing with CCS, but with the help of various resources online, we can see how other developers resolved these issues. The MQTT client code will need to be flashed to the device using Uniflash, a TI program for storing applications in the internal flash memory of our device. Because CCS mostly uses C, we will rely heavily on pre-built libraries and header files that define helpful functions for communication. Unlike energia, which is written in a more object-oriented sense, CCS will require a more adept sense of programming to get the project on its feet. However, CCS comes with more freedom in that the code doesn't have to work on a loop-based system. This will give us the ability to incorporate interrupts to send data when a specific condition is met.

5.6.2 Developing in Energia - JG

Since we want to use a Texas instruments microcontroller, using tools TI provides examples and tutorials for is a reasonable option. Venturing into IoT will be a difficult task on its own, since the methodologies used haven't necessarily been covered directly by our undergraduate curriculum. However, as engineers we must learn to apply what we know to solve novel challenges ahead of us. While it might be too much to design an entire IoT software from scratch, there are resources available that can do most of the heavy lifting. TI has paired with Temboo to provide machine-generated code for the CC3220 to interact with a handful of RESTful APIs [22]. To use Temboo, we first choose which MCU we are using, and how its connected. Depending on which API call we try to use, other information regarding authentication, source/destinations, and URLs to websites may be needed. Temboo then generates code based on the information we provide that can be copy and pasted into Energia, where we can upload it to our board. Using Temboo, our MCU can interface with APIs such as Google sheets to store data, Twilio to send SMS text alerts and calls, or other APIs we can write ourselves if needed.

5.6.3 AWS Cloud Solution - JG

To set up an AWS solution to cloud communication, we must download and install the right tools to set up our project development environment. Either CCS or IAR Embedded Workbench IDEs can be used. Using CCS, SimpleLink WIFI CC32xx Wireless MCUs must be installed. We will also need to install the SimpleLink CC3220 SDK, which contains essential drivers for the

CC3220SF MCU and the latest service pack. CCS Uniflash will be used to upload the service pack. The next step would be to configure the WIFI settings for the chip, either with the Amazon FreeRTOS demo, which has its own documentation on the AWS website, or using the SmartConfig app from Texas instruments [23]. Once the demo code gets built, compiled, and debugged for errors, we should be able to monitor MQTT messages on the cloud in the AWS IoT console. To subscribe to the MQTT topic, we will need to make a AWS account, sign into the AWS IoT console, and subscribe to the demo’s topic. After getting this framework set up, we can then alter the code to communicate with our sensors, have the sensors publish MQTT messages to be viewed on the AWS console, and have our console set different states on the device. Eventually, we expand our infrastructure to create a webserver to serve as a user-friendly GUI for data visualization and device control.

Using the TI CC3220SF microchip, we can learn from various online tutorials and solutions. Company’s like Google and Amazon both recommend the CC3220SF-LAUNCHXL Development Kit to get started on a cloud connectivity. The embedded software should allow the device to gather sensor data from the plants in our smart garden and post it to a web server online.

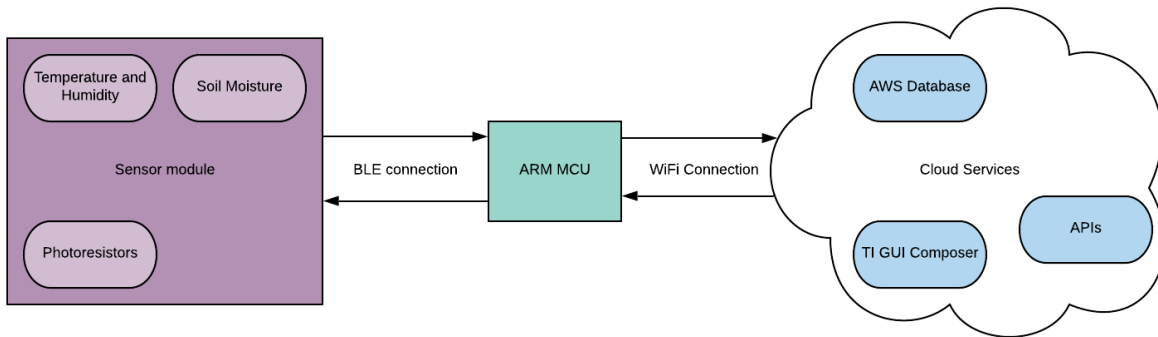


Figure 8: Communication Model

5.6.4 Google Cloud Solution - JG

Like Amazon, Google also recommends using the CC3220SF Launchpad as a starter development kit. The online tutorial provides an easy way to begin developing an IoT device using the CC3220, Mongoose OS, and Google IoT Core. Prior to writing code for the device, we install google cloud SDK for its command line functionalities, and Mongoose OS to program the microcontroller. Using the gcloud command line feature we can create the project on IoT cloud, set up authentications, make Pub/Subtopics and subscriptions, and construct a registry to store the device’s data on the cloud. Then, using Mongoose OS we can flash JavaScript code to the MCU along with WiFi information. The code can be altered in the Mongoose OS script editor as needed. Following the tutorial Google has for the CC3220, we can first begin implementing IoT Core by learning how to send device state updates to the cloud with an internet button, as well as changing the LED state of the board remotely through the cloud. Once these features are implemented, we can begin to shape the device into a smart garden. For example, by replacing the button with a sensor, we can trigger updates to the cloud when a threshold is reached. The information sent over MQTT is represented in a json structure, making it easily transferable over Google’s resources.

5.6.5 Communication challenges - JG

With the features we want to implement in our smart garden, there will be a few challenges to consider with communication. With WiFi, many internet service providers (ISPs) will block unprotected and unauthorized devices from accessing the network. For testing purposes, we can use a phone hotspot, but to make this a commercially available product, we will need to figure out how to connect the device to a user's home internet without being blocked by the ISP. For Bluetooth, there might be challenges connecting multiple devices to Bluetooth at once. We want Bluetooth sensor nodes to monitor the state of each plant we have in the garden, or at least each soil bed we have. We expect to have at least two sensor nodes communicating data over Bluetooth to our main MCU. We'll need to make sure the master MCU can connect to all the sensor nodes we have over Bluetooth, either one by one, or all at the same time to retrieve sensor data. Also, we want these Bluetooth sensor nodes to operate in low power. Putting our sensor nodes into low power mode will restrict the operational functionalities. Using the MSP430G2553 as the main processor for the sensor nodes, we can put the device into a low power mode to wait for the clock to wake it up. However, if the user wants to manually retrieve data at any point in time, we should accommodate Bluetooth low power mode (LPM) wake up so that we can wake up the device wirelessly. To have our sensor node be able to react quickly out of LPM for a Bluetooth request will be difficult, and we might have to sacrifice keeping the sensor nodes in ultra low power mode to let the user request data at all times, or sacrifice having real-time user readings in order to save battery life on our sensor nodes.

5.7 Programming Languages - JG

The various programming languages utilized in the indoor greenhouse system will be discussed and justified in this section. This primary languages used in this project are C/C++, Python, Assembly, and JavaScript.

5.7.1 C/C++ - JG

One of the most widely used programming languages for embedded applications is C/C++. Designed to be portable, C code can be adapted and arranged for many MCUs. IDEs such as Code Composer Studio, can be used to program TI microcontrollers in C. C/C++ code can also be uploaded to the device using the command line, with the right configuration set up. New Online IDEs such as CCS cloud can be used to flash C code to a device from online. Because C/C++ has been an industry standard for many embedded applications, there is a copious number of projects available for us to use as a reference. However, C code is often written in a procedural manner, and doesn't always work correctly from one device to another. For embedded developers, Embedded C was created to interface with different kinds of devices, supporting features such as fixed-point arithmetic, the ability to differentiate between memory banks, and basic I/O operations [24]. The use of C for embedded programming often includes a variety of libraries, or SDKs to help make the code neater and reusable.

5.7.2 Python - JG

As devices are becoming more capable of handling higher-level language code, python has risen to the forefront of microprocessor development languages. Python outranked C++ as the most popular programming language in a report by IEEE [25]. MicroPython is a software implementation written in C that can translate Python code to instructions that can be run on a microcontroller. There are examples of using MicroPython online that we can use for getting started, but it depends on what processor we pick. Raspberry Pi and Arduino devices are often compatible with python code, and many hobbyists use it to build rapid prototypes. However, since python is interpreted, it might not be as fast or effective as programming in C.

5.7.3 Assembly - JG

Programming in assembly would give us more control over the MCU, but would hinder the portability of our code. Assembly interacts directly with the registers contained in the processor, there's a steep learning curve to understand assembly that our group doesn't have experience in. The code is hard to read for unfamiliar developers, and functions turn out to be long and tedious. We would prefer to use a higher-level language to take advantage of code that is already written to interface with our MCU. Assembly code might run more efficiently on our device, but the level of optimization that languages such as C or python have achieved make the benefits of using Assembly negligible.

5.7.4 JavaScript - JG

We can program in JavaScript to set up an MQTT Client and communicate with APIs across the internet. A JavaScript MQTT client can also pass along messages to applications running Node.js.

5.7.5 AT Commands -AL

We will be using AT commands to control the actions of the BLE module, such as putting the module to sleep, seeing the address of the module within the memory, changing the mode from master to slave, etc. All of the commands in this language starts with "AT" or "at, thus the name AT commands. As a quick background for these commands, many of the commands that are used in controlling dial-modems.

5.8 Software – JG

In this section, the various software platforms for programming and design will be discussed to outline the framework and basis of design for this project.

5.8.1 Code Composer Studio - JG

Code Composer Studio (CCS) is an integrated development environment (IDE) used to program various Texas Instrument (TI) microcontrollers and embedded processors. The software provides tools used to develop embedded applications such as compilers and libraries for specific hardware,

as well as a project build environment, source code editor, debugger, console output window, and other amazing features that make setting up working software easy for developers. The interface is designed to be user intuitive so more effort can be put into writing the code, rather than spent setting up a project environment over the command line. This software is available for download on TI's website and is now available on the cloud.

To set up MQTT using CCS, we first need to have the right compilers up to date. The CCS App center and resource explorer will have most of what we will need. First, the right Ti-RTOS drivers will need to be installed. These will need to be stored in the same folder as the SDKs for CCS. Next are Free RTOS drivers, AWS FreeRTOS is also compatible. After setting up our drivers, we can also download an mqtt client demo from the resource explorer. We'll first need to set the local access point parameters, the WiFi and password of a local network, and define them in a header file. Subscription topics can be defined, and the port number needs to be set. In order for MQTT to operate, there needs to be a defined MQTT broker.

TI Recommends using eclipse as their open source client code will work on port 1883. After editing our code, we can compile and run the code on our microcontroller. We can publish topics to the device by using defined web link extension such as /cc3200/ToggleLEDCmdL<x> to toggle an LED. The LED indicated by x should turn on if set up properly. ICPDAS MQTT for iOS can be downloaded to test out our device on a mobile phone subscribed to the topic. By subscribing to the WaterLevel topic, we can enable the MCU to send a notification to the app when a water level threshold has been reached, notifying the user that their plants need more water. The CC3220 SimpleLink WiFi chip can even run a local MQTT server and connect to the internet as an edge device. Since we would like to implement a star network, this processor can easily implement software that will let it extend MQTT communication out to other sensors we hook up [26].

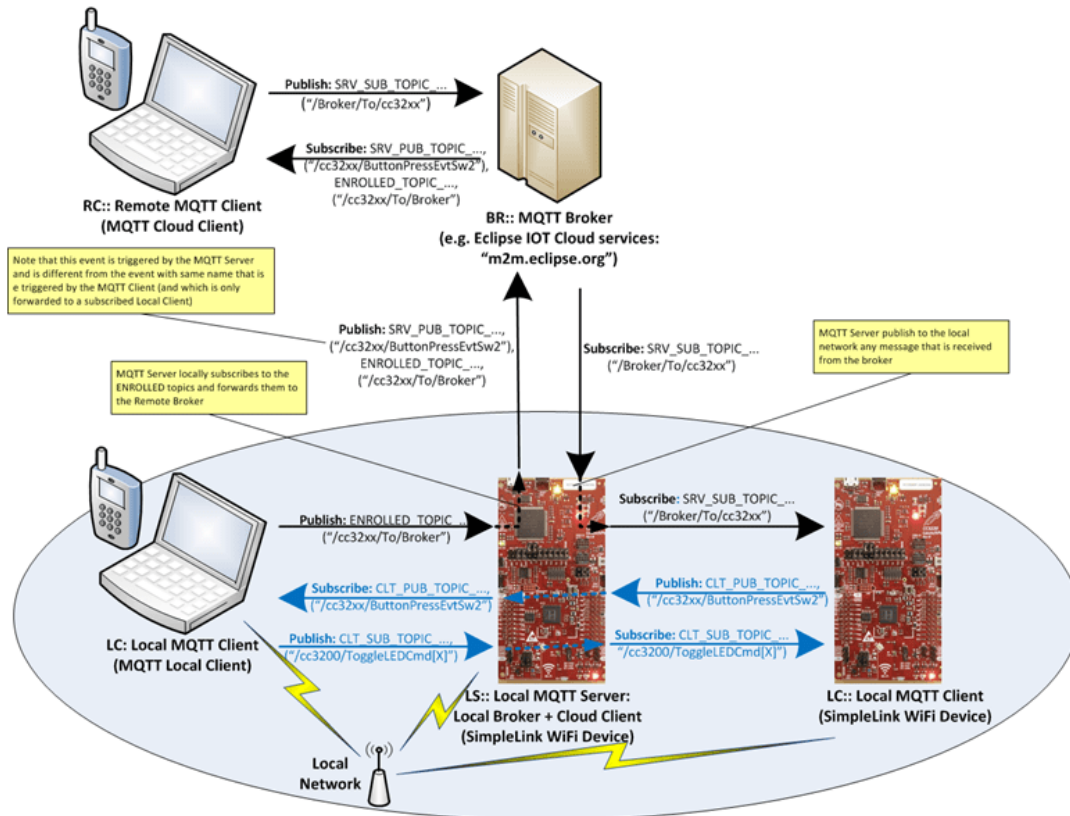


Figure 9- MQTT Client-Server Demo [26]

5.8.2 Energia - JG

Like the Arduino IDE, Energia attempts to make programming TI MCUs easier with a high-level, object-oriented framework of programming. Energia has example projects we can use to experiment with as we develop solutions to make our device communications work. Energia is available to download for free and can easily be installed on a Windows, Linux, or Mac computer. The software supports compilers for a variety of TI MCUs and uses the mspgcc compiler.

5.8.3 Uniflash - JG

CCS Uniflash is a tool used to program on-chip flash memory on TI MCUs. It has a GUI, command terminal, and scripting interface to create and load new files onto a chip's memory. Some TI WIFI boards need an additional emulator board in order to be programmed properly. For example, the CC31XXEMUBOOST is needed to update the firmware stored in the CC3100's serial flash. Uniflash makes updating the firmware easy for embedded developers, and will likely be needed for our project.

5.9 Operating systems - JG

Most embedded commercial applications use an embedded operating system to schedule tasks. Embedded operating systems are often designed with resource-efficiency and reliability in mind. Because embedded devices are constrained by memory, efficiency comes at the cost of losing some functionality. The type of OS that is frequently used is called a real-time operating system

(RTOS). Operating systems can aid our project by managing multiple threads at once. For each sensor we connect to the microcontroller, we can add a thread to poll sensor values. The operating system can be programmed to start and stop threads over WiFi.

5.9.1 Why Use an Operating System? - JG

To gain better control of an embedded chip's processing power, software developers may write the application code in assembly. In assembly, the programmer can specify exactly how memory gets utilized on the device. However, assembly code isn't always portable, and gives developers a hard time when trying to integrate the device with others using higher-level languages. I couldn't imagine sending JSON payload through WiFi to another device using assembly. For this reason, we can design an operating system to manage the nitty gritty memory optimization for us, while we comfortably develop code in C. However, unlike desktop operating systems, embedded operating systems do not load and execute applications, restricting the device to running only a single application.

5.9.2 TI-RTOS - JG

TI-RTOS is a scalable embedded tools platform for TI devices. It supports a real-time multitasking kernel (SYS/BIOS) and supports additional middleware components and device drivers to make up a complete RTOS [27]. TI-RTOS doesn't come pre-installed on TI devices, nor do all TI devices support an RTOS. If we want to incorporate TI-RTOS into our project, we must pick an MCU with enough processing power to handle an operating system, such as the MSP432 which comes with a 32-bit ARM processor. TI-RTOS can be installed through the Code Composer Studio app center. The TI-RTOS components include source files and SDKs that come with pre-compiled libraries and examples.

5.9.3 AWS FreeRTOS - JG

First released in November 2017, Amazon FreeRTOS is an MCU operating system design for connecting lightweight embedded devices to AWS IoT core. This means we can have sensor devices connect directly to the cloud, without the need for an intermediate gateway [28]. Amazon FreeRTOS augments the existing FreeRTOS kernel with libraries for connectivity, security, and over the air updates. The operating system is typically flashed to devices as a compiled image of all the components required for the device's applications. This image packs the combination of the FreeRTOS kernel needed to manage the device's memory, software libraries provided by Amazon to connect with AWS IoT Core or other AWS cloud applications, drivers and board support packages for the hardware, and the applications written for the device by a developer. A visual of the AWS FreeRTOS architecture is shown in this figure.

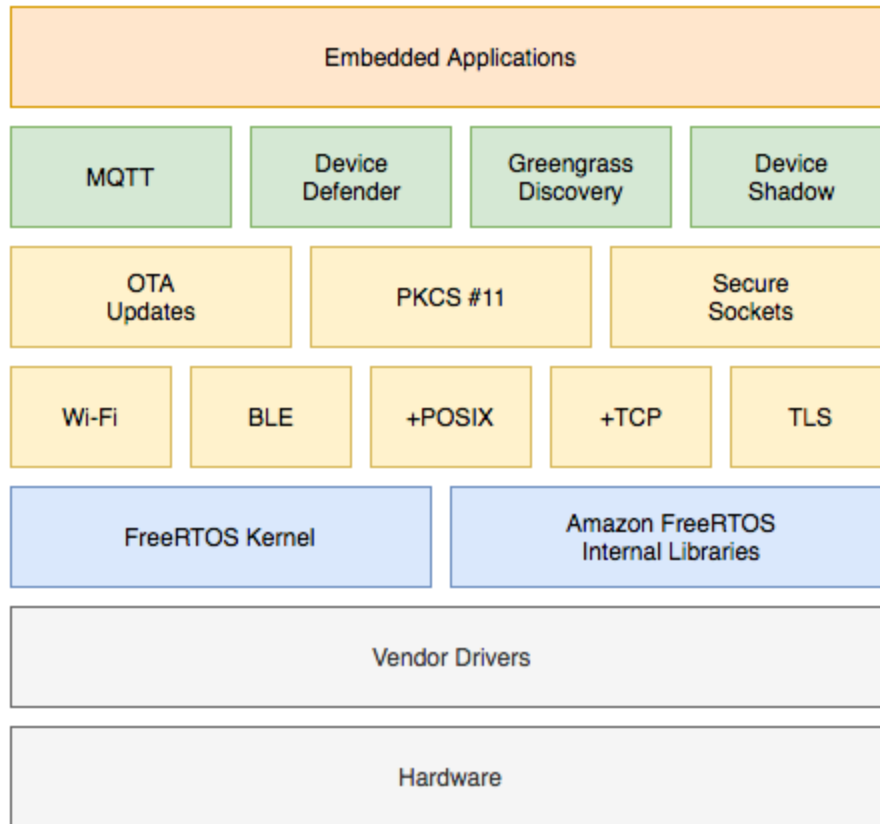


Figure 10: AWS FreeRTOS architecture

5.10 Cloud – JG

Embedded applications are often constrained by memory space and processing power. By connecting our device online, we can offload all the data we gather into the cloud, so we don't have to waste power or constrain our memory storing it on the device. We also get access to external APIs to help run analytics on our data if we wish, resources otherwise not supported by our microcontroller alone. The “cloud” is an umbrella term that refers to the vast amount of online resources we can use for the project.

5.10.1 CCS Cloud - JG

Though there are many microcontrollers to choose from, TI offers hardware familiar to our group while offering a robust set of functions to implement an IoT system. While building our software, we can use CCS cloud to compile and flash code from any device if the hardware has a WIFI connection. The code gets stored in the cloud as well, so we won't have to worry about keeping our offline copies consistent with each other. It also comes with user friendly tools to get a simple web-based GUI set up, and combined with websites like Temboo, can open a gateway to other external APIs we can use for data analytics and storage.

5.10.2 Security - JG

There's no "S" in IoT, but Security is one of the biggest concerns with IoT devices. In 2016, a massive hack to the internet took advantage of unsecure devices and took down large portions of the internet in what has been called the Mirai Botnet attack. The hackers scanned the internet and found millions of unsecure IoT devices to take advantage of. A massive attack was launched against DYN, the domain services for the entire internet and down went the internet for a few hours. At the micro-level, there was an attack on a casino in 2017 where a hacker found an unsecure fish tank temperature sensor, got into the internal network, and pulled out 10 Gigabytes of high roller data from the database. They lost millions of dollars and credibility just because one device wasn't properly secured. Since we want our device to connect to a cloud database, the connection needs to be secure, so our project doesn't contribute to macro-level internet attacks or expose a user's home internet to malicious software.

5.10.3 Google Cloud Platform - JG

Using Google IoT core, data can be securely sent into the cloud and passed on to other applications for further processing. Once in IoT core, the data is stored online, eliminating the risk of losing data if a transfer to a database gets corrupted midway. Google Cloud pub/sub implements the MQTT protocol that will control and listen for information from a device. Pipelines like Cloud Functions are a quick way of passing information on to other applications needed to do analytics. Cloud Firestore can handle syncing data across multiple connected devices and allow for flexible scalability. Firebase can be used to securely host a web application for user control and observation of an IoT system. The web application can be integrated with an Angular Frontend to interface with Google Sheets, where data can be represented for the user. Our project might only take bits and pieces of what google services has to offer, but the project will remain flexible for future development and scaling.

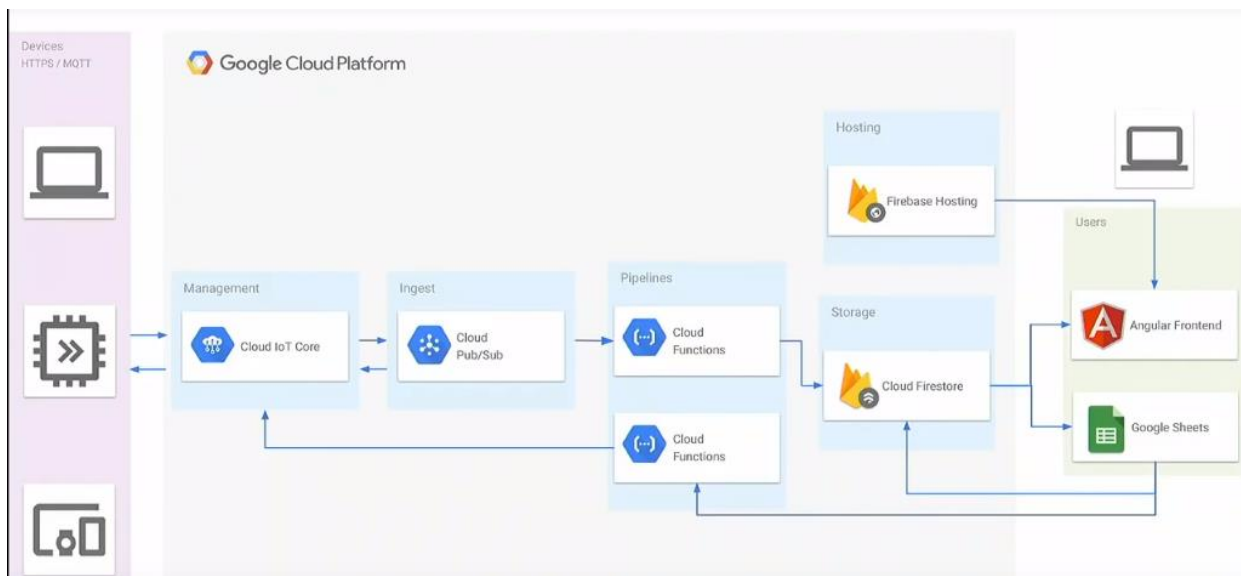


Figure 11: Example Google IoT Device Framework

5.10.4 Amazon IoT Services - JG

Another cloud service is Amazon Web Services (AWS) which provides IoT tools to monitor the states of things about a device so further reasoning on the data can be done to solve novel problems [29]. Starting with our devices, where most of our data will be generated from, Amazon FreeRTOS provides an IoT connected operating system for microcontrollers that allows easy connectivity from low power devices on a local network to more powerful devices that connect to the internet over Amazon IoT Core. Since microcontrollers usually have limited computing power and often perform simple, functional tasks it is especially helpful to offload the data gathered to the cloud. The operating systems that run on microcontrollers often do not have the functionalities to connect to local networks or the cloud, which makes designing IoT systems difficult. However, AWS FreeRTOS helps solve the problem by providing an operating system and other software libraries to securely connect to the cloud or other edge devices. This way we can collect data from them for IoT applications and eventually further process the data on a more powerful system. If we want our device to act locally, AWS IoT Greengrass can make that happen. In the case a device gets disconnected from the network, Greengrass ensures the data stays securely in sync even when not connected to the internet.

To gain access to the cloud, our device will need to access AWS IoT core, the backbone for enhancing the microcontroller's capabilities with online functionalities. AWS IoT core is scalable to accommodate for millions of connected devices, which allows our smart garden to easily integrate itself into a smart home with an existing framework. It supports HTTP, WebSockets, and the most popular communication protocol for IoT devices, MQTT. Even if other connected devices are using different protocols, AWS IoT core manages that too. Security features include authentication and end-to-end encryption throughout the transfer of data, nothing gets exchanged between devices and AWS IoT core without proven identity. The latest state of connected devices gets stored in AWS IoT core, making it appear to be online even if it is disconnected or in a low power state. This allows the user to set a device state to be implemented when the device reconnects, without having to provide overhead for making sure it happens real-time.

Once uploaded into AWS IoT Core, other services can act upon the information. For instance, AWS IoT Events can be utilized to respond to sensor data triggers. By defining the conditional logic that describes an event to be listened for, such as a water level sensor threshold when the garden needs to be refilled, AWS IoT events can detect and trigger the appropriate response to send a text message to the user, or turn on a notification light on the system.

5.11 GUI - AL

To have a way to communicate with the controller and all its peripheral devices we will plan to have a GUI in the front of the Smart Tabletop Greenhouse. This will serve as a local interface for a user to be able to access and control all the environment changes in our greenhouse, look at data provided from sensors, and provide a troubleshooting system that will notify the owner if anything is out of the ordinary for the greenhouse.

To do this we plan on using an application on TI cloud, as we plan to have our controller be from Texas Instruments. In addition to the GUI Composer application being the main compiler for TI components, there are also features in this application to be able to easily access and show data

through the chip via graphs along with numbers and words. There are even objects within the application that simulate thermometers to make ease of custom event handling as our data is processed. For our error messages, the composer application has features where pop-up windows can be created just as if on a regular PC to notify its users.

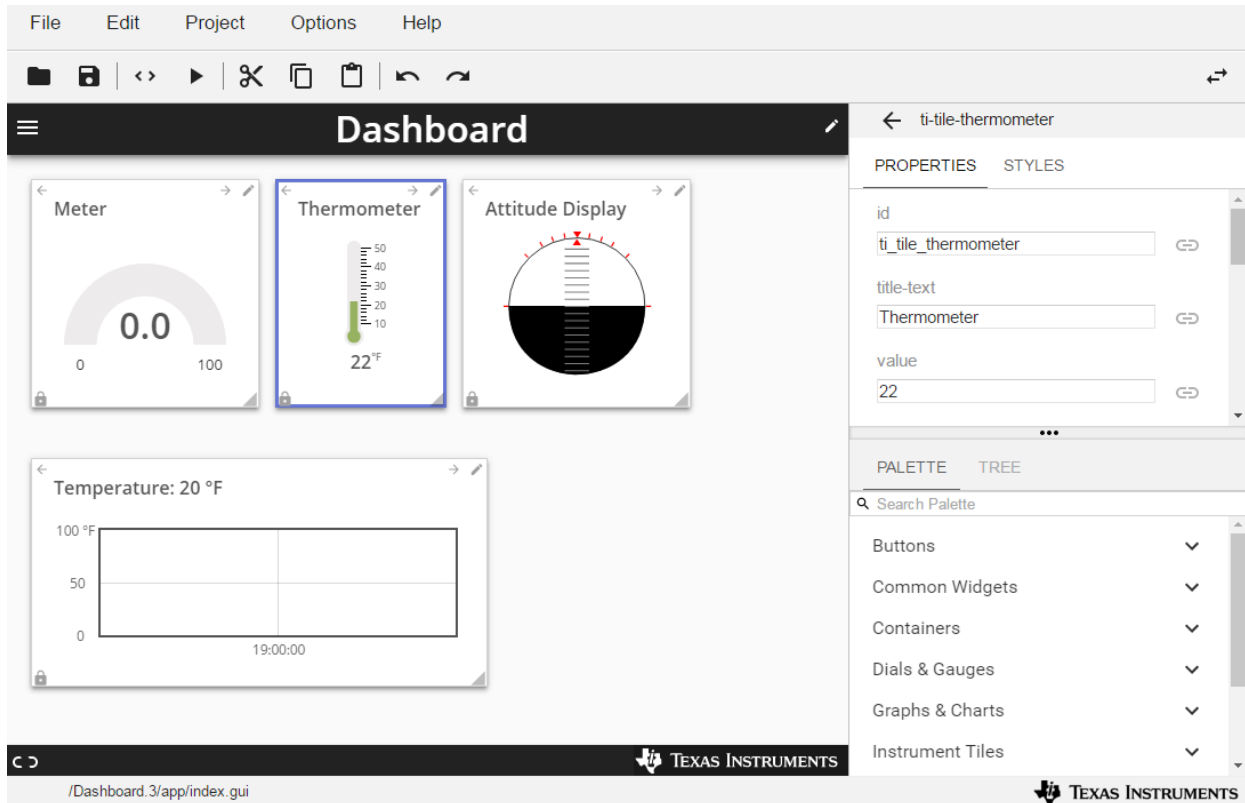


Figure 12: TI GUI example

Other features that we plan to have for this GUI is have an option to send the data from the connected sensors in its system to either be stored locally or to be sent out to the cloud to be offloaded and be observed from a separate device. This functionality was added so we could easily store the information observed from the peripheral devices just in case that the ARM processor did not have enough memory to hold all the data. As one our group members wanted to have more experience with Cloud data storage and communication, this seemed like a perfect task to add to our project.

5.12 Bluetooth Modules - AL

Our choice of communication between our parent and children microcontrollers are via Bluetooth so there were many considerations in which modules to use to incorporate our idea. We had a slight debate on whether we were to use regular Bluetooth or BLE so we did research on a few modules that had some options for both types of communication. Many of these modules were

simple enough where if we were to have 5-6 pins (Vcc, GND, TXD, RXD, enable, and state) we would be able to fully integrate the separate module into our microcontroller configuration via jumpers.

5.12.1 DSD Tech HC-05 - AL

We first looked at this module as it was able to be a master and a slave, two in one module. It uses the CSR BC417 as its Bluetooth chip and follows the V2.0 SPP protocol standard. Though AT commands we would be able to switch easily between master and slave mode. DSD Tech also had a nice software to easily modify the default baud rate of 9600, Bluetooth name, pairing password and role. The only drawback was the working voltage was between 3.6V and 6V which be inconvenient to adjust all of our power supplies incorporate the module.

5.12.2 HiLetgo HC-05 – AL

While this module's factory setting is in slave mode, one could set it to master mode as to be able to connect other Bluetooth 2.0 devices to it. The transceiver master slave integrated module has an effective transmission distance of 10 meters in an open area, which is more than enough for our enclosure's constraints. The module can be added our design easy enough, but does not work with iOS, and although this might not seem like a problem at the moment, in future plans, we might want to consider creating an app in the Apple Store to be able to interface with *SmartLeaf*.

5.12.3 HiLetgo CC3541 HM-10 – AL

This Bluetooth module adopts the CC2541 chip of American TI Company, has a 256Kb of space and follows the V4.0 BLE Bluetooth specification. The input voltage needed for this device can be either 3.3V- 5VDC which is perfect for an idea of a child microcontroller that is to be powered by coin celled batteries. The module supports SPP Bluetooth serial port protocol and UART interface. The BLE technology that is incorporated with this board is ideal for our uses for *SmartLeaf* as we plan to use the module that draws the least power and is able to connect quickly to our devices.

Through tutorials and videos online, we were able to find that if we were able to implement that AT commands correct and interface with the microcontroller correctly, the module would provide us with a better service that a regular V2.0 Bluetooth module. When the device is not being paired with other devices and is sending and receiving data, it will draw around 10mA. When we put the into sleep mode, which is not possible with V2.0 devices we would find that the module draws around 0.12mA. The device also communicates quicker than the device shown in DSD Tech HC-05 - AL and HiLetgo HC-05 – AL and works perfectly as a device that only intends to send data occasionally. This device can be found on Amazon and even comes as a two piece set for less than \$10, and is within the budget and gives us the best value per module.

5.13 LEDs -AL

We decided to go try to go with an aesthetically pleasing look while also being a reliable source of light for the plants inside our structure. The LEDs we were looking for were to also try to have a low power consumption as to preserve the longevity of our project. To do this we had several choices of LED light strips to choose from.

5.13.1 Adafruit RGBW LEDs -AL

One of the first LED strips that we looked upon was Adafruit's analog RGBW LED strip. This is a 60 LED-per-meter strip that even had a cool white and was even weatherproof as it had a plastic covering over the LED reel. The way that there are wired though, we would need a 9-12 VDC power supply, a ground, and a RGPW pins to turn on any of the colors.

5.13.2 CHINLY LEDs -AL

Next, we considered CHINLY's LED reels. These reels had nearly all the same functionalities as Adafruit's; RGBW potential LEDs with a 24-bit color display, with a 3-pin connector for the power, ground and signal ports, a waterproof but had a drastic differences in pricing per LED and also used 5VDC instead. Other than those two companies, the other options on the market were more or less the same of what was described above but were either more expensive, used more energy per LED, or was manufactured in a different country which would drastically change the timeline of when our components would arrive.

So it was a natural choice to choose CHINLY as our main supplier of LEDs, as they had a package for a 5 meter, 300 RGBW LED reel on amazon, which would give us plenty of spare LEDs to do testing on, an easy to use software development code that would easily make the LEDs do what we want to do, consume less power as a whole, and save us money per LED. Many hobbyist have also done projects with the same LEDs and have many examples to be able to reference if there is any trouble in the future in implementing them.

5.14 LCD Screen -AL

Our group also decided to look for a single LCD screen that was to be connected to the main ARM processor to be able to provide the user a way to interface with the GUI we were to develop. The optimal LCD screen that we had in mind was a 3.5 inch to 7 inch screen that had touchscreen capabilities, was able to be controlled from our ARM processor, had either SPI or I2C interface capabilities, and have an open source libraries to be able to configure, upload and adjust code from the ARM processor. We looked in many sources on the internet and came to find that many of the LCD screen were more on the expensive side. Some of the screens we looked into were TFT (Thin film transistor), had the capability to have more pixels on a smaller surface.

5.14.1 Adafruit LCD - AL

With Adafruit's 3.5" TFT 320x480 LCD screen, it had a 6 white-LED backlight with individual RGB pixel control, had a build in controller to help with RAM buffering, SPI and 8-bit interfacing

and a full open source graphics library that can be easily used to implement our ideas for a GUI. Power consumption is also a plus with this device, 3.3VDC and 5VDC compatible with a 150mA LDO (low-dropout regulator), so it doesn't use as much power as the bigger TFT screens and will have less of a burden on the ARM processor if chosen. Another reason to use this particular component is because Adafruit does a really good job on making its product user friendly and provides many tutorials and examples on how to program and use their products.

5.14.2 Booster pack LCD for MSP432 - AL

There was also an option to purchase the BOOSTXL-K350QVG-S1 Booster pack for the MSP432. This is a Kentec QVGA Display Boosterpack that is an easy-to-use plug-in module that adds a touchscreen color display to the MSP430. It too is a 3.5-inch screen but has a 320x240 resolution instead. It also has a SPI interface and a LED backlight driver circuit. The only advantage in getting this backpack is that it would be easy to implement on top of our already existing MSP430 and would be a tad bit cheaper than the Adafruit TFT. The problem being that the booster pack has a horrible resolution rate compared to that of the Adafruit TFT.

5.15 Fans – DM/MM

Main things to consider when looking at our project, we need to ensure that the plants that we're going to be growing will survive. Being a tabletop greenhouse, the interior will warm up and to equalize this, we'll be adding fans to the tabletop greenhouse for circulation and cooling purposes. It is expected that two fans will be featured in the final design, on either side of the unit in order to encourage air circulation quickly and effectively. As this is an indoor system, the fans need to be quiet enough as to not cause a disruption/distraction in the surrounding environment, yet powerful enough to make an impact as an environmental control.

(MM) Any fans considered for this project should have an operating voltage in the range of 5V-12V to be easily compatible with the power supply. Luckily, various manufacturers sell fans with these operating conditions, largely thanks to hobbyist computer builds. Nevertheless, Table 12 shows the viable options for use in the *SmartLeaf* system.

Table 12- Fan Comparison

Fan Model #	Manufacturer	Distributor	Power Requirements	Noise	Airflow	Cost
RA04010HD1	RDK	Amazon	12V, 90mA	28 dBA	6.08 CFM	\$13.99 (set of two)
490-CFM-5010-03-22	CUI	Mouser	5V, 280mA	36 dBA	16.07 CFM	\$7.90
CFM-6015-13-22	CUI	Mouser	12V, 220mA	39 dBA	27.50 CFM	\$8.74

5.15.1 RDK Brushless Cooling Fan - MM

This option comes in as the most affordable at about \$7 per fan. The RDK fan also has low noise and load current demands. It is also available on Amazon Prime with free 2-day shipping, amounting to greater savings. Although convenient, Amazon typically does not provide datasheets for additional useful information. The major concern, however, is that this device may be insufficient in providing enough airflow for the greenhouse, providing only 6.08 cubic feet per minute (CFM). This is likely insufficient to act as a means of control for temperature and humidity conditions within the environment. For this reason alone, it is not a viable option for the *SmartLeaf* system.

5.15.2 CUI DC Axial Fan – 5V - MM

This option from Mouser is able to deliver more airflow than the previous option, measuring in at 16.07 CFM. While the greater noise is a concern, the power requirements make a valid argument for this device. The distributor, Mouser, also provides datasheets for their products, which would be helpful in the design and construction of this unit. Available in a 5V input option, there would be more room for growth in other areas of the project. While a viable option, there is still the possibility of there not being enough ability to quickly control the airflow and correct environmental conditions. In this case, it may be more feasible to select a more powerful fan as a basis design and adjust the revolutions per minute through pulse width modulation (PWM).

5.15.3 CUI DC Axial Fan – 12V - MM

This fan comes from the same manufacturer and distributor as the device above, but with greater ability to adjust airflow, measuring in at 27.5 CFM. While requiring more power and adding more noise, this model has lines for control signals as a selectable option. Specifically, the design team would have the ability to include a PWM line, which could eliminate the need for an additional external controller. Upon this fan's arrival, the team can determine the appropriate speed and noise allowed for the system. For this reason, the CUI DC Axial 12V fan is selected for use in the *SmartLeaf* system.

5.16 Pumps – DM/MM

Since, this is a “smart” tabletop greenhouse we’re going to attempt to make the regulation of the water autonomous via a water pump of sorts. We’re planning on setting up our soil moisture sensors such that when the data reads below a certain threshold that we manually set for the moisture, the water pumps would be signaled to activate and water the plants near where we placed our soil moisture sensors. This pump that we’re going to use will more than likely be installed with a tubing that will be hidden from plain view. The tubing will be attached to the top our tabletop greenhouse that’ll be just lined up with where we’ll be placing our plants. The plants themselves will be positioned near the soil moisture sensors and more than likely each plant will have their own individual sensor to avoid inconsistencies in the soil itself when it comes to the moisture content. If we ended up facing inconsistencies such as this, we could potentially set the sensors to a threshold that would be considered inaccurate which could trigger the pump to give more water than the plant needs or vice versa.

(MM) Various manufacturers make pumps with operating conditions at and below 12V that could fulfil the watering needs for this project. Pumps of this nature are typically used for aquarium filtration/circulation and fountain features; largely compatible with microcontrollers thanks to hobbyists in these areas of interest. Table 13 compares some accessible options for pumps of this type.

Table 13- Pump Comparison

Pump	Manufacturer	Distributor	Power Requirements	Price
Submersible Mini Pump	Ledgle	Amazon	12V, 300mA	\$8.99
Submersible Brushless Water Pump	Daniu	Banggood	12V, 400mA	\$7.88
Peristaltic Liquid Pump	Adafruit	Adafruit	5V, 500mA	\$24.95

5.16.1 Ledgle Submersible Mini Pump - MM

This 12V pump has relatively low demand current and is conveniently available for free 2-day shipping via Amazon Prime. Amazon has additional warranty provisions for this product, making it a viable option for use in the *SmartLeaf* system. The downside, however, is the comparatively higher voltage requirement. A 5V pump would be ideal to conserve power, but luckily the power supply used in this project was selected for its scalability, and a 12V pump that works well with the other systems may be justification for using this excess power. The Ledgle pump has been used in similarly-sized greenhouse projects as a means for plant watering, which has added appeal.

5.16.2 Daniu Submersible Brushless Water Pump - MM

The Daniu pump is extremely similar to the Ledgle pump; having the same voltage requirements, this pump has a comparable – albeit higher – power demand. The load current for this device is 100 mA greater than the previous, which is a drawback. Additionally, while this pump is slightly cheaper than the Ledgle option, shipping costs will exceed the price of the latter. More time for shipping would have to be accounted for, and an unfamiliar distributor may not offer replacements or refunds in the case of defective or broken equipment upon arrival. For these reasons, the Ledgle pump is the preferred option.

5.16.3 Peristaltic Liquid Pump - MM

Unlike the two previous pumps discussed above, this device from Adafruit is a peristaltic pump, which works by compressing the attached tubing containing the liquid being moved rather than impelling it directly. This ensures that the liquid does not make contact with the pump, maintaining a sterile system ideal for applications in the food and agriculture industry. The lower voltage requirements make this device appealing, but the price is a drawback. Overall, the unique

functionality of the peristaltic pump does not offer enough value to the project to justify the excessive cost. Pumps that come in contact with the water being transported are not considered unsanitary by any means, and any produce grown or purchased is expected to be washed prior to consumption, making the added benefit of a sterile system redundant. For these reasons, this device was not selected as the basis of design for this project.

Out of the two remaining options, the Ledgle submersible pump was selected for use in the *SmartLeaf* indoor greenhouse system on the basis of cost and convenience. Amazon Prime’s free 2-day shipping allows the design team to get access to the pump much quicker than purchasing from other distributors, and there is enough documentation and tutorials/examples including this type of pump on the internet to make up for a lack of engineering datasheets.

5.17 Humidifier - MM

A humidifier is needed in the *SmartLeaf* system in order to raise humidity and cool the internal environment of the greenhouse when necessary. The humidifiers considered for use in this project are typically used for aquariums, terrariums, or garden fountain features. There are specific models used for the average greenhouse, but those are likely too powerful for use in the indoor system. Although the humidifier will be controlled based on sensor readings, if it increases the air moisture too rapidly, the fans would be required to run more often to correct the condition, leading to more power consumption and audible noise than necessary. Given the small and fully enclosed volume needed to control, as well as the relatively low humidity level needed to maintain – around 50%-60% relative humidity, a low-power humidifier should be acceptable.

While several manufacturers offer similar units that could potentially be used in the *SmartLeaf* system, there is a notable lack of options and flexibility with those offered. The same type of 5V diffuser disk with attached driver is sold by many distributors, but there are no notable differences amongst this type. For this reason, fewer options are compared to eliminate redundancies. The following sections serve to compare and select from these options. Table 14 displays data about the two humidifier types.

Table 14- Humidifier Comparison

Model #	Manufacturer	Distributor	Power Requirements	Price
B00PAK245E	AGPtek	Amazon	24V, 1A	\$9.99
11434	IC Station	Amazon	5V, 400mA	\$5.99

5.17.1 APGtek Aluminum Mini Mist Maker - MM

This humidifier type is of interest due to the ease of use; it is able to be submerged to produce humidity, and for basic operation it does not require a driver. However, this device – and the vast majority of humidifiers – does not come with exposed wires or additional information that would allow for flexible design. Cutting the device from its pre-connected AC adaptor could be done, but it would be unideal and potentially create a dangerous situation, which is why the manufacturer strongly discourages the practice. Another drawback of this humidifier is the high power needed for operation. For these reasons, the APGtek humidifier is not considered for use in this project.

5.17.2 IC Station Ultrasonic Mist Maker - MM

This humidifier is initially appealing due to the small size and low power consumption. Available on Amazon with standard shipping and handling, this device is easily purchasable and accessible. This unit comes with its own USB powered driver which does limit some flexibility of design. Additional testing will need to be conducted in order to modify this unit to be powered directly from the power rails. If anything, the actual ceramic fog-inducing disks are available in bulk on Amazon Prime (pack of 5 for \$8.98), which could come in handy for additional testing. Further research into alternative drivers for this device should be conducted to ensure controls compatibility with the MSP432. Although the choice is less straightforward than the other devices specified throughout this project, a general lack of low-power humidifier options points the IC Station Ultrasonic Mist Maker as the basis of design for the *SmartLeaf* system.

5.18 Power Supply - MM

The greenhouse system is to be powered by standard general-purpose receptacles. In North America, it is common that multiple general-purpose receptacles share a 120V, 20A, single-phase branch circuit from a 120/208V panelboard. The most common receptacle type (and the one this power supply will be plugged into) is the NEMA 5-15R, which is rated for 15A at 125V. Since multiple general-purpose receptacles are typically on the same circuit, each individual receptacle on a 20A circuit is limited to 15A (or rather 80% of that amperage, as noted below in *Section 5.4 Power Standards*) to protect equipment and wiring. Compatible plugs for the NEMA 5-15R are the NEMA 1-15 and 5-15 power cords, as shown in Figure 13 and Figure 14 below.



Figure 13- Receptacle and plug type used by the SmartLeaf system. Permissions requested from NEMAenclosures.com



Figure 14- Compatible plugs for the NEMA 5-15R; NEMA 1-15 (left), NEMA 5-15 (right). Permissions requested from NEMA

5.18.1 Power Calculation - MM

With a basic understanding of the mains power constraints, the power supply can now be considered and accounted for. Power supply circuitry generally consists of a step-down transformer that lowers the mains input voltage to the needed amount. A full-wave rectifier will then convert the alternating current (AC) to direct current (DC), and a smoothing capacitor filters any remaining AC ripple voltage. With the AC to DC conversion fully completed, the correct output voltage can be provided to any equipment through a voltage regulator. To understand the input current and voltage the system needs, we need to consider requirements for each of the components; particularly, the highest voltage needed in the circuitry and the sum of all load currents. For the initial sizing of this power supply, only the high-power components are considered, as the combination of low power devices being considered are unlikely to exceed an amp, and the final calculations are to be upsized for more flexibility in the design.

The calculation shown below in Table 15 provides a rough estimate of the total current draw of the system. All ratings above indicate the maximum current draw possible for each individual device; it is unlikely that all systems will require full power for operation. However, room for growth and additional features need to be considered. To account for all systems within the greenhouse safely, the power supply will be upsized to 120W at 12V, 10A. Due to the high amperage and desire to alleviate any risk of leakage current, a NEMA 5-15 plug will be required for its additional grounding pin. Both external and internal power supplies that meet the above power demands are widely available on the market, but an internal power supply will better maintain an orderly appearance for the unit. Integrating the supply into the base of the greenhouse will not be met without additional challenges, such as a heavier unit and increased heating concerns. Small tanks for watering and runoff will also be located in the base, and it is crucial to provide a means of separation and ground fault protection for safety, whether through a specialized waterproof enclosure or a completely waterproof device. It should be noted that internally integrating the power supply does alleviate tripping hazards in regard to the cord, which would be more exposed in an external unit. This will lessen the chance of fall hazards and the potential toppling of the unit, which could cause extreme injury.

Table 15- Total Power Estimation

Device	Input Voltage (VDC)	Input Current (A)	Quantity	Total Current (A)	Total Power (W)
Adafruit 3.5" TFT 320x480 LCD Screen	5	0.15	1	0.15	0.75
Chinly RGBW LEDs SR5338	5	0.06	120	7.20	36.0
Micro Quiet Brushless Motor Submersible Water Pump	12	1.00	2	2.00	24.0

Device	Input Voltage (VDC)	Input Current (A)	Quantity	Total Current (A)	Total Power (W)
IC Station Ultrasonic Mist Maker Fogger 20mm	5	0.50	1	0.50	2.50
Pi-Fan Brushless DC Fan	12	0.25	2	0.50	2.50
MSP-EXP432P401R	3	0.10	1	0.10	0.30
			Total:	9.25 A	52.05 W

5.18.2 Power Supply Selection - MM

As stated, there are several 120W power supplies available off-the-shelf. Market-ready devices may bring along undesired limitations when compared to custom designed units, but are extremely reliable, often available under warranty, and include various safety features that would otherwise be costly to integrate. Additionally, these units can be shipped, delivered, and tested in a short time period. Of these options, only waterproof units should be considered. Unfortunately, this effectively doubles the price in respect to non-waterproof devices of the same power ratings, but safety is priority. In Table 16 below, a handful of power supplies that meet the 12V/10A/120W specification and necessary waterproof ratings are compared.

Table 16- Comparison of Waterproof 120W power supplies

Power Supply	Warranty	Dimensions	Cost
Ideally 120W DC 12V Ip67 Waterproof LED Power Supply	2 year	6.61"x 2.67"x 0.78"	\$31.99
Yaetek AC 90-250V to DC12V 120W Transformer IP67 Waterproof Electronic LED Driver Power Supply	N/A	1.6" x 2.8" x 7.5"	\$16.99
YGS-Tech 12V LED Power Supply 120W IP67	1 year	6" x 2.71" x 0.6"	\$39.99

While the second option is most appealing due to the price, it does not have the NEMA 5-15 plug attached; purchasing and safely soldering the power cord would be additional cost and labor, and therefore is not ideal. Of the two remaining options, the first has a longer period under warranty

and is overall cheaper. The size advantage of the last option is not considerable enough to justify additional cost; therefore, the Ideally power supply will be used for this project.

5.19 Voltage Regulators - MM

The power supply will effectively transform and step-down the 120VAC to 12VDC to feed the entire greenhouse system, but voltage regulators will be necessary to provide the appropriate power rails and added protection needed for individual devices and sensors. In the text below, different types of voltage regulators will be compared and discussed for their potential use in this project, primarily linear regulators and switching regulators. Factors such as efficiency, design flexibility, noise, cost, and complexity for the two regulator types will be discussed and compared in this section, and the device type used as the basis of design for the power rails in the *SmartLeaf* system will be identified.

5.19.1 Linear Regulators - MM

Linear regulators are extremely cheap and have useful application in devices with very low power requirements. They effectively maintain a constant DC output voltage regardless of any fluctuations in the input conditions. While they can provide the correct input conditions and protection to the circuitry downstream, linear regulators are extremely inefficient due to the specific nature of their operation. Any difference in input and output voltage is wasted in heat, and they all require an input voltage higher than the desired output. Larger variances of voltage result in more heat being exhausted. Typical efficiencies expected of linear regulators are around 40%, but they can be as inefficient as 14% [30]; so while some money can be saved choosing this type of voltage regulator, additional expenses of heatsinks and system cooling may need to be considered. In low-power applications, where the heat loss would not introduce significant complication, cost savings as well as low complexity and small noise production would likely justify the use of linear regulators. However, a 120W power supply is hardly a low power application and would likely result in significant heat loss that would need further mitigation.

5.19.2 Switching Regulators - MM

A switching regulator is a voltage regulator that employs a switching element in order to transform the input voltage into a ‘pulsed’ output voltage, which is then smoothed using passive components such as capacitors and inductors. By rapidly switching a series element on and off, the input energy is temporarily stored and released as the desired output voltage through the utilization of synchronous or non-synchronous switches (FETs). Switching regulators are extremely efficient; no significant power is lost to heat, and a wide range of input voltages can be taken and converted to the desired output – including voltages lower than the desired output voltage – which allows for various regulatory configurations. Switching regulators do come with some drawbacks, however. Inclusion of capacitors and inductors within the circuitry can mean additional bulk from external components and added electromagnetic interference. In applications where complex circuitry and additional noise would add unjustifiable complications and cost, linear regulators may be a better choice. In our case, efficiency, design flexibility, and input range easily justify these negative traits, as outlined in Figure 15 and Table 17 below.

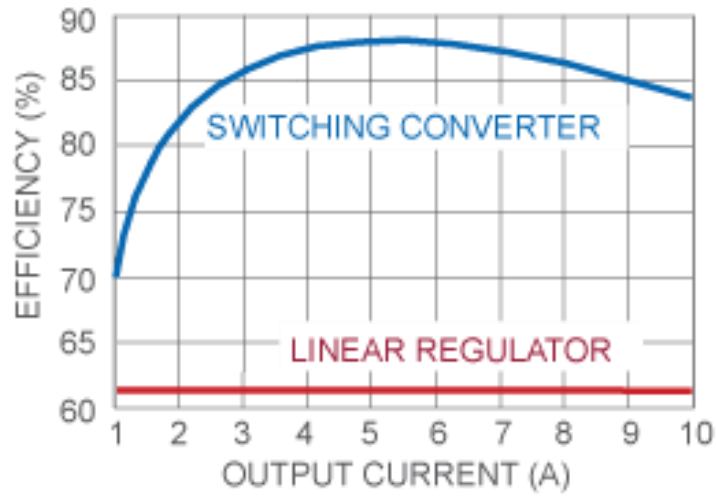


Figure 15- Power savings between low-dropout linear regulators and switching regulators.
 Permissions requested from Renesas Electronics

Table 17- Comparing Linear and Switching Regulators

Factor	Linear Regulator	Switching Regulator
Design Flexibility	Buck	Buck, Boost, Buck-Boost
Efficiency	Low to medium-high for low $V_{in} - V_{out}$ differential	High
Complexity	Low	Medium/high
Size	Small/medium, larger at high power	Smaller at high power (dependent on switching frequency)
Cost	Low	Medium/high (external components)
Noise/Electromagnetic Interference	Low	Medium/high
V_{in} Range	Small (dependent on power dissipation)	Large

5.20 Batteries - MM

The *SmartLeaf* system will employ three separate microcontrollers to communicate via Bluetooth. This will allow for two child units in the upper greenhouse enclosure, and a parent controller in the base of the unit with the rest of the circuitry. Given that the two child MSP430 controllers will only serve to receive and transmit sensor data, they can be operated in a low power mode the majority of the time. To streamline design, the two child units will be battery powered. This design decision may present some challenges in that the user may have to replace batteries over the years, but for now, this allows us to limit the amount of wires being fed through the unit and aids in maintaining a neat, organic appearance. The MSP430 requires a supply of 1.8V-3.6V, but likely will be in LPM3 or 4 for the majority of use. A 3V battery will be supplied for each child MCU, which will be enough to support them in active mode, and will have a considerable shelf life in its low power states.

The flowchart showed in Figure 16 below shows the overall scheme for delivering power to each device within the project. The fans, pumps, LEDs, LCD display, humidifier, and parent MCU will all be fed from the main voltage rails via switching regulators. The water level sensors, Bluetooth and WIFI modules will be fed directly from the parent MSP432. The child MCUs, fed from 3V batteries, will provide power to the various sensors and Bluetooth modules to support parent-child communication.

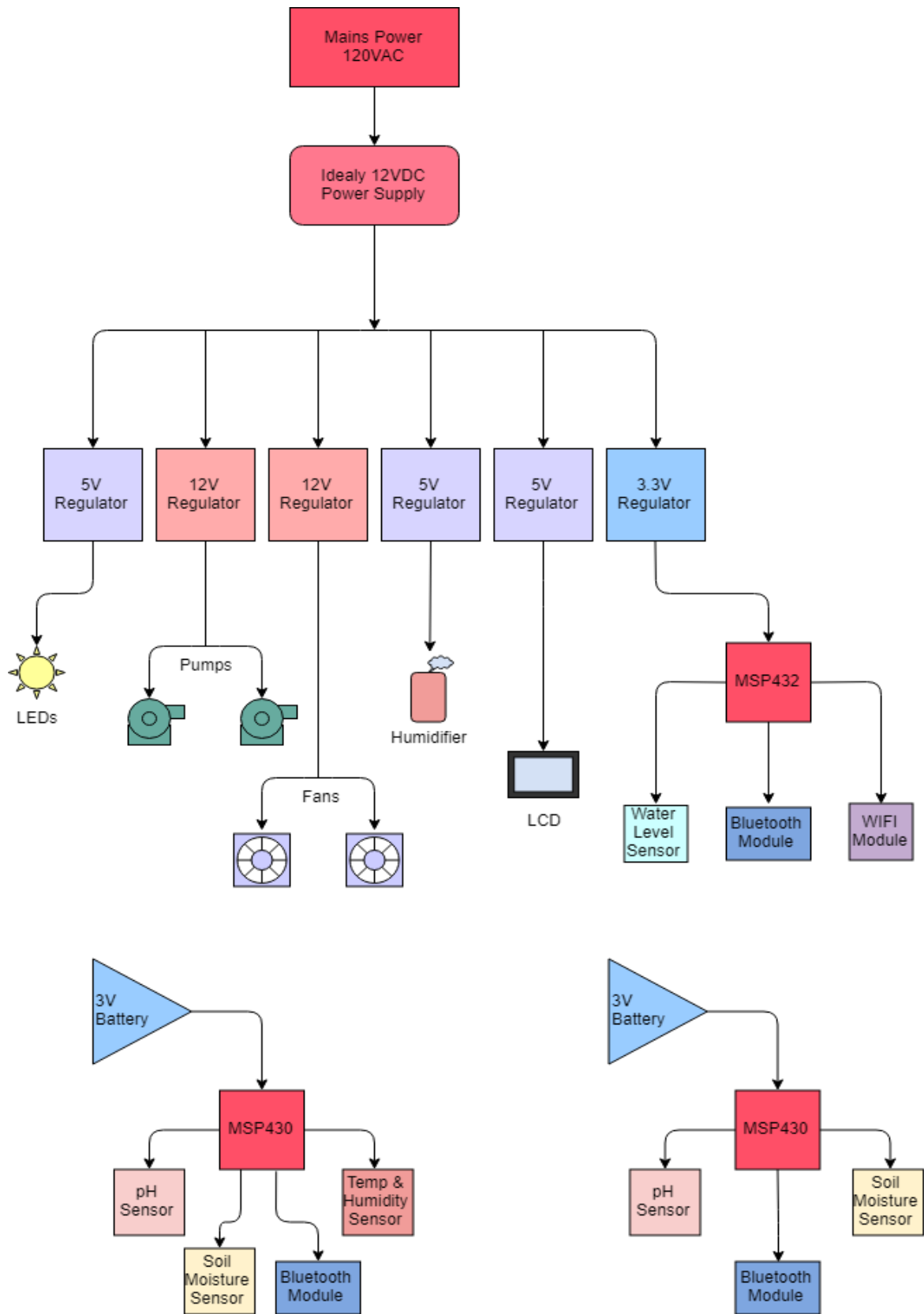


Figure 16- One-line diagram for the SmartLeaf System

5.21 Control Systems - MM

The *SmartLeaf* indoor greenhouse will largely rely on control systems to maintain a nurturing environment for plant growth. Four main environmental control systems for supplying water, airflow, humidity and light will rely on switches, drivers, or controllers in order to turn on and off and vary the output characteristics of the devices when triggered by the sensors. In this section, we will be discussing relevant devices that can be used to control the LEDs, pumps, fans, humidifier, and LCD screen.

5.21.1 Power MOSFET - MM

Using a power MOSFET is the most straightforward approach to implementing a switch for our external devices, as the gate can be directly connected to PWM driving pin on the microcontroller. Typical gate to source voltages (V_{GS}) are around 0.7V, which is small enough to be driven directly from the MCU. For an N-Channel MOSFET, the positive terminal of the power supply would be connected to the positive terminal of the device (fan, pump, etc.), the negative terminal of the power supply would be connected to the source of the MOSFET, and the drain would be connected to the negative terminal of the device. The microcontroller can be programmed to pull the port feeding the gate to high to turn the fan on, or low to turn the fan off. Using an infinite loop and delays within the code, the port can be pulled high and low indefinitely, driving a PWM that can vary the speed or mechanical work done by the device. Using a Power MOSFET for this design is not only intuitive, but also convenient as many of the switching regulators considered for use in this project have an integrated MOS device for this exact purpose. This switch is most likely to be used in the *SmartLeaf* system to control fan speed and turn the humidifier on/off. Further research will be discussed in the sections below to determine the best means of control for the pump circuit.

5.21.2 Motor Controllers - MM

These controllers are able to drive multiple motors, capable turning them on and off, varying the speed at which they operate, and even changing the direction of rotation. This could enable the fans to have more flexibility in how the air is circulated throughout the system, and could potentially allow the pump to work in either direction. The widely-available L293D Dual H-Bridge Motor Driver can drive two DC motors bi-directionally, and has built-in heat sinks for thermal regulation. However, this driver adds little valuable functionality over a standard MOSFET switch for this project; having the ability to change motor direction is not necessary for basic greenhouse operation. Additionally, more control lines from the MCU would be needed to operate this driver, putting further strain on the demand for GPIO pins. For these reasons, a motor controller would not be ideal for use in the *SmartLeaf* system.

The following figure shows a diagrammatic plan for how the fans, pumps, and humidifier are to be controlled using N-Channel MOSFETs and the MSP430. The LEDs used for this project have an integrated driver that can be directly controlled by the MCU, and the LCD display can also be directly controlled without more external devices. Additional components such as ripple capacitors and pull down resistors are not shown in this diagram for simplicity, but these components are expected to be included within the final design according to the manufacturer's recommended design.

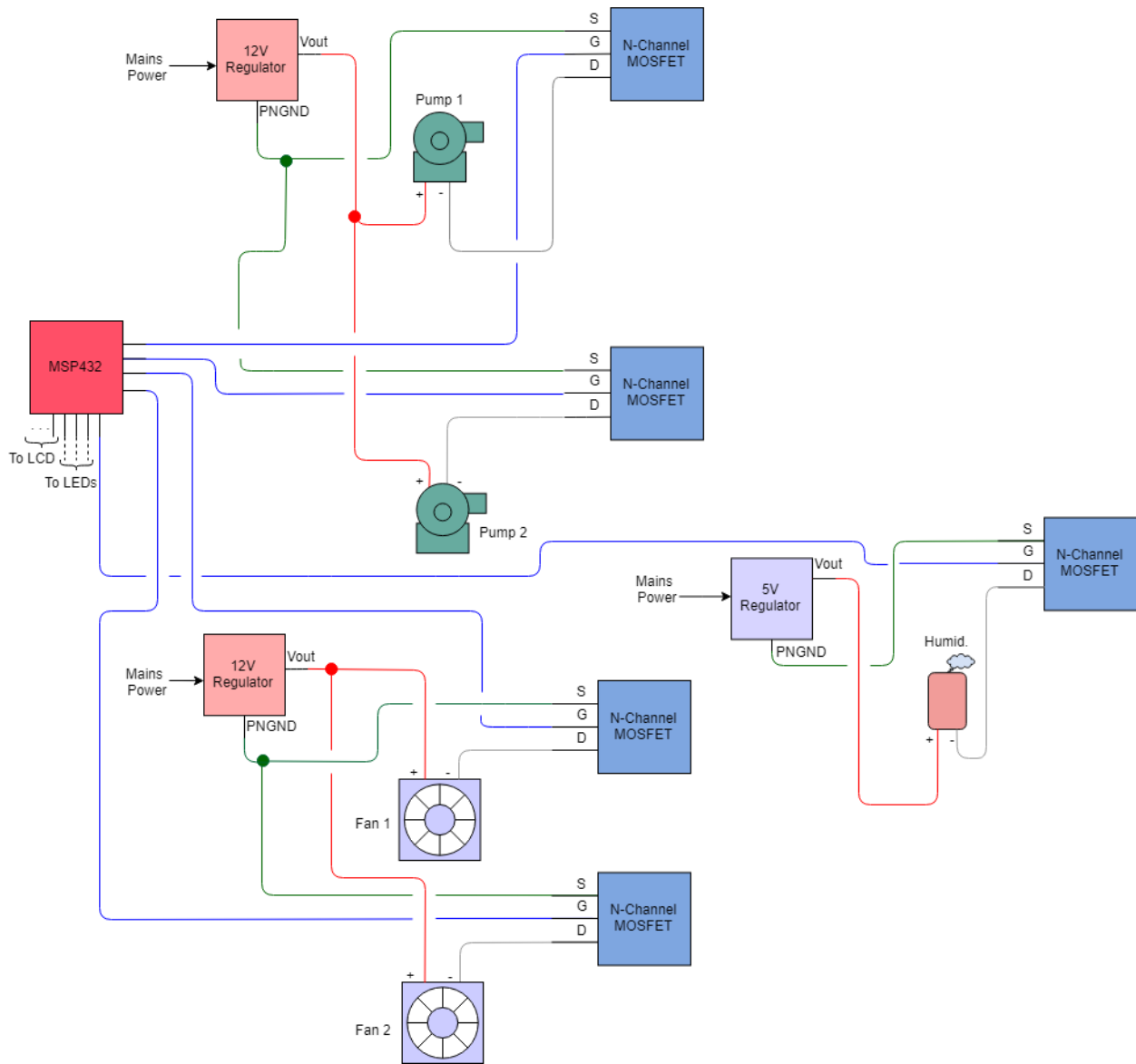


Figure 17- Control Diagram

Given the demands of this project, the MOSFET chosen will need to have a drain to source voltage (V_{DS}) of at least 12V in order to ensure the pumps and fans receive enough voltage for operation, and a low enough threshold voltage to be driven by the MSP432. The Texas Instruments 40V N-Channel NexFET Power MOSFET will be used as the basis of design for this project, as it can be used for each device in our 5V-12V input range.

5.21.3 MSP432 Control - AL

A few components within the *SmartLeaf* system do not need an additional device for its control and can receive data lines directly from the parent microcontroller. With having the MSP432 as the central “brain” of the system, we can easily send signals to these N-Channel MOSFETs to select whether or not power the specific peripheral devices. We plan to set the gate of each gate channel in the MOSFET to a single GPIO/ PWM pin on the MSP432 and to set the signal high when we deem that it is time to power each peripheral system. The great thing about this control system is that we can also modulate the speed that each motor comes up in with the peripherals connected through PWM pins so that the circuit has a chance to be able to drive those lines without fear of having too much of an inrush current when a system is suddenly turned high.

For the MOSFETs that would go to devices that would use GPIO pins on the MSP432, we would the devices that have integrated regulators included them such as the Wi-Fi module and Bluetooth module as they would provide a cushion for the excessive inrush current. These MOSFETs will serve more so as a switch to select which device we want to use for the occasion.

5.22 PCB Design Software - DM

When it comes to the design of the PCB for this project, we need to consider many important things regarding the design. Even though we may have everything carefully planned out on how we’re going to be putting the project together regarding our selection of sensors, microcontrollers and Bluetooth or wireless functionality, it would mean nothing at all if we have a poorly designed PCB layout. It’s one of the most important parts of the project and shouldn’t be taken lightly, the PCB is what keeps everything together and working as a functioning unit. We need to consider a multitude of things such as the placement of our components, where to place the traces for power and ground, component separation, and lastly, we need to know how to deal with the issues of potential heat damage that can occur on the PCB.

Here are some things that could be considered standard when considering the construction of our PCB. We need to also make sure that we double check everything regarding the design of this PCB because it would be a problem if we found a problem much later when we’re building the project itself and we must send a PCB redesign, and have it manufactured and sent to us a second time. We would much rather avoid that issue by checking our work thoroughly to avoid this.

There are many CAD software’s that exist for PCB design such as Kicad, Eagle, EasyEDA, PADS, OrCAD, NI Circuit Design Suite, PCBWeb Designer, ZenitPCB, TinyCAD, Osmond PCB, BSch3V, ExpressPCB, gEDA, Fritzing and DesignSpark PCB. Our discussions have been gearing us towards using the Kicad software but our decision on deciding this is determined mainly by ease of use.

5.22.1 Kicad and Eagle - DM

We need to accurately compare all this CAD software and decide which one is optimal for us to be using. We should start by comparing both Kicad and Eagle. Both programs are used for the same purpose but the main differences between the two are how user friendly they are. Both Kicad as well as eagle have ample resources on the internet with tutorials on how to use and navigate their software. There is one major difference between both Eagle and Kicad and it’s that in Kicad you’re able to view your PCB design in a 3D format. You’ll be able to see an accurate

representation of how your PCB layout is going to look before it'll be sent out for manufacturing. This is only one of the advantages that Kicad has over Eagle.

Another advantage that our group will be considering before we attempt the design of our PCB is the cost of the software. What gives Kicad an edge over Eagle in this regard is that Kicad is a software that anyone can download and install free of charge. Eagle on the other hand has two versions, a free version as well as a paid version with subscription. The paid version simply has more features than the free version but nonetheless still lacks a 3D viewer for us to utilize. Cost of the Eagle subscription is about \$15 a month which is costly given the comparison of features between the free version of Kicad and the paid version of Eagle. This is just comparing the two programs being just Kicad and Eagle.

5.22.2 EasyEDA - DM

We need to start discussing the other software's as well. Ideally, we should speak about EasyEDA next. This program from what we've seen is a very professional program for PCB design with many free libraries for us to use. Though EasyEDA may have everything that we may require there is also a paid version that isn't too expensive for us to use. EasyEDA comes in multiple types of subscriptions, there's a free version, a \$5 subscription version and a \$10 subscription version, each with their own benefits obviously the highest costing having the same benefits as the previous tiers. Though there is a downside to using EasyEDA and that is that this PCB design software is web based, and not something that is downloadable to your computer. The fact that this is sort of thing exists whereas it is a web-based software shows that it is not very secure. Regardless of what is said on the website about security there's always a chance that the information that we have set for our PCB design could be leaked and we want to avoid that as much as possible, though the odds of something like this happening is extremely low.

Also, being web based, there's also the chance of the website being shut down for maintenance purposes and if we're in a situation where we need to make certain revisions to our design and we see that the website is shut down temporarily we be in a bad spot and the project itself will be at a standstill. Though they do have some other upsides that they mention on their features section such as them allowing full team collaboration on the web-based PCB design software as well as support via email, however the email support that they provide is divided into the tiers. The free version of EasyEDA only offers their email support for only 12 hours whereas the paid versions of their software offer support for 24 hours. This makes a huge difference because if we need to make an emergency contact to their support and they're unavailable it would not be good for us or our project. Other features that they showcase are mundane things such as lack of advertisements, a customizable logo, and local/cloud hosting. Those are the features for EasyEDA but there are other PCB design software's to speak about and one of those is known is PADS.

5.22.3 PADS - DM

PADS is a very useful PCB design software with their own benefits as well as their setbacks. PADS has a ludicrous amount of features but their business model and how they go about distributing the software holds it back. PADS have an insane price point of around nine thousand dollars and that's what they consider to be their special pricing for it, so based off this, we can infer that this is for a limited time sale. Though that price point is for their professional version of

their software which has nearly everything a person wanting to do PCB design would need. They do offer a standard version called PADS standard which includes features such as schematic design, constraint management, PCB layout, options for creating your own libraries, 3D viewing of the PCB layout models, archive managements as well as quick-start guides, design references and already available libraries for beginners.

Comparing this software to EasyEDA from earlier, this one is a downloadable software that would be installed to the user's computer and can be accessed offline without the need for internet access. They do a one-time payment model so once you purchase the software, it's usable for as long as you need it. What differentiates the PADS professional from the PADS standard version are the features included. PADS professional offers everything that the standard edition offers plus a few more extra features that a PCB designer would find useful for their projects. Features that are offered with the professional version include design entries, quick-start guides with lessons on how to tackle and approach the use of this software, signal integrity for post-layout and pre-layout, analysis of thermals, simulations for analog, 3D PCB layout as well as a native 2D option if the user prefers, and sketch routing.

As we can see from the features of the professional version, the main difference between the standard and professional would be the addition of included simulations and sketch routing as well. Sketch routing is a very useful tool to have with this PCB design software's since this option allows the designer to auto-route traces on the PCB design except for focusing on optimizing the routing to make the best possible connections. This allows for minimal errors with routing and saves the designer a lot of time and effort trying to figure them out for themselves. The addition of the feature of simulations also allows the designer to pre-test what they're about to place in their design before even making the addition to the PCB. These are only the features that the PADS software provides, they have a lot of positives to their software, but their price point holds them back from our decision to use their software.

5.22.4 OrCAD - DM

Another PCB design software for discussion is OrCAD. This software has many features that someone looking into PCB design would want. Features that are included in OrCAD are schematic designs, simulations of parts to see whether they would be useful in your PCB design, options for simulations with MATLAB Simulink, PCB design software, the ability to perform an analysis on signal integrity and finally the ability to perform ERC checks. ERC checks or Electrical Rule Checks check for any irregularities in the PCB design which can be viewed as very critical and drastic errors that need to be taken care of when found. If there are errors such as these our design could simply malfunction and not work properly due to these very critical errors. This is a very nice feature to have from OrCAD. OrCAD offers a free trial to anyone wanting to use their software. Though this trial needs to be made upon request which is available on their website. Other features that OrCAD has are an interactive 3D viewer for the PCB design to check for any irregularities as well as a DFM that can be used as well. DFM is a term for Design for Manufacturability. This will allow us to check in the final steps of our PCB design for any errors that may be critical when it comes to final assembly of the PCB. This is a very versatile and useful feature that we as a group need to take into consideration when making our decision on what software we're going to be using.

Unfortunately, there's one major downside to this software and that is that we do not know how much this software costs. The main website for OrCAD requires the consumer to contact a sales representative for a quote on licensing and how to purchase the product. This really inconveniences us in terms of getting our hands on this product because we simply do not know how much this product costs. They claim that there's a student version as well but that also requires us to contact one of OrCAD's representatives for a quote on pricing. Another negative downside to this is that they mention that the student version of OrCAD is limited in terms of features. The features that are limited are unfortunately not mentioned on their website. Though seeing what OrCAD offers in terms of features and functionality makes it a nice candidate for selection for us on deciding which PCB design software we're ultimately going to select and use.

5.22.5 NI Circuit Design Suite - DM

NI Circuit Design Suite is another prime candidate for a PCB design software that we may be using as well. This software for us engineering students is somewhat familiar to us because in our courses we used some of National Instruments' software that being their Multisim software. Us as students in a laboratory environment are well versed in using this software in multiple courses in engineering. They offer an upgraded version of their Multisim software that can be used for PCB design as well. It's called Multisim for Designers which offer many features such as simulation in SPICE, multiple tools for analysis and many PCB design tools that would be more than enough for us to use for our greenhouse project PCB design. Though there is a setback to using this software as well, since if we were to get this Multisim software, we'll also need to get National Instruments other software called Ultiboard which allows us to port the schematic onto the PCB design software.

There are PCB design tools available on the Multisim for Designers software but to make our layout we'll need two software's in total not just one. Pricing for both software's vary but they're not on the cheap end whatsoever. For the National Instruments Ultiboard software as well as the Multisim for Designers operate on a service plan going by yearly subscriptions. These software's require recurring payments after a certain amount of time to continue their use which is not ideal for our purposes. They offer for both software three different editions. They offer Educational, Professional and Full. On their website they're very unclear on what features are missing and what features are included in each edition. For our purposes on selecting our PCB design software, the pricing of the software for National Instruments as well as the fact that we need to make two separate purchases will more than likely deter us from using their PCB design software.

Table 18- Comparing PCB design software

PCB Software	Supported Operating System	Pricing	Features
KiCAD	Windows/Mac/Ubuntu/Debian/Linux	Freeware	Schematic creator, PCB designer, 3D PCB viewer
Eagle	Windows/Mac/Linux	Subscription	Schematic Creator, PCB designer

PCB Software	Supported Operating System	Pricing	Features
EasyEDA	Windows/Mac/Linux	Free/Subscription	Schematic creator, PCB designer
PADS	Windows	Subscription	Schematic creator, PCB designer, 3D viewer
OrCAD	Windows	One-time payment	Schematic creator, PCB designer, 3D viewer, Error Testing
NI Design Suite	Windows	One-time payment	Schematic creator, PCB designer

5.23 Board Environment - DM

After we have created our PCBs using our chosen software, we'll need to have it manufactured and sent to us in order to use it. There are multiple manufacturing companies that can take care of the manufacturing process of our PCBs and send it to us in a timely manner. Also, a couple companies that not just manufacture the PCB but also can have it shipped to you with all the components attached to it as well, so we don't need to waste time with soldering. Though these are simply the positives. Negatives that some board manufacturers might have would be an extremely slow shipping time. There are some board manufacturers that can take up to a month maybe even longer before you'll be able to receive your circuit board. All components are going to already be attached to the PCB and all that'll needs to be done is attaching our PCB to the peripherals that we have such as the fan, humidifier, water pump, LEDs, water sensors etc. And mount it to our enclosure and start running our completed project. Now, each board environment has different positives and negatives to them but the main ones we're going to be discussing are JLCPCB, 4PCB and Sunstone.

5.23.1 JLCPCB - DM

This company for custom made circuit boards is well known and one of the more popular manufacturing companies for custom made PCBs. This is largely since the time for manufacturing doesn't take an extensive period. On the main website for JLCPCB they claim to manufacture, and ship custom made PCBs to the consumer in a short time span of a maximum of two weeks. Comparing to some of the other manufacturers that offer manufacturing for custom made PCBs, the time it takes them is very commendable and good for people that are under a very short time constraint and really need to get their hands on their circuit board as soon as possible. Another downside to choosing JLCPCB is that their prices are expensive when comparing to their competitors. These prices are considered expensive because if you want your board to be prioritized and shipped as quickly as possible, you need to pay a premium price for that to happen, but they offer their quick shipping as a selling point to entice the consumer to purchase from them.

5.23.2 4PCB - DM

This board is going to more than likely be our main choice since their prices can be considered fair and they run multiple quality checks on the PCBs before being sent out to the consumer. Their main site mentions that they have the best PCB shipping record within the PCB manufacturing industry. They also offer technical support 24/7 whenever we need to get in touch with a customer service representative. There is no specific requirement to order a minimum number of PCBs from them in order to make a purchase. If anyone also wanted to check their own PCB design, they have their own special software that we can use to check whether our PCB will function as we intend it to based off our design.

5.23.3 Sunstone - DM

Sunstone is a board manufacturer that, like 4PCB claims to be the leader in board manufacturing. Though the first negative I saw when navigating their website is that you cannot order a single PCB for them to manufacture. You need to purchase a minimum amount and that would be in pairs of two. Though this can be seen as both a positive and a negative simply due to the fact that if something went wrong with the board on the first attempt of our assembly of the components such as the accidental burning of a solder pad on the circuit board essentially ruining the circuit board itself, we would have an extra board laying around to try our hand at assembling it again. This would be very useful for us, but the fact is that Sunstone simply does not have the option to purchase a single board from them.

Board Environment	Affordability	Minimum Purchase Requirement	Shipping Time
JLCPCB	Premium	No	2 weeks
4PCB	Fair	No	3 weeks
SunStone	Premium	Yes	3 weeks

6.0 Vendors - MM

In this section, the manufacturers and distributors of components purchased for the *SmartLeaf* system will be listed and discussed. Factors such as convenience, price, shipping speed, design support, and device documentation will be considered for each vendor as a means of justifying their inclusion. Additional considerations for customer service, warranties, and design tools will also be noted.

6.1 Texas Instruments – DM/MM

Many of the development boards for testing and initial experimenting on were from TI. Since they have so many tutorials, such a wide range of backpacks to have added on to the development boards for our purposes, and not to mention all the software user guides to go with it, we decided

that TI would be the best companies to have a centralized hardware and software standpoint more the microcontrollers.

(MM)Aside from microcontrollers, Texas Instruments has a wide selection of switching regulators, power MOSFETs, and miscellaneous passive components with detailed, in-depth documentation. Their engineer-to-engineer (E2E) program gives us access to assistance from applications engineers with device-specific expertise. Tools created for design assistance are also available through TI, including the Webench Power Architect, a powerful program that outlines devices capable of meeting the power demands in a certain project. While TI typically requires large, bulk purchases of circuit components, it is possible to request device samples to be sent, which will not only relieve strain on the project's budget, but also allow for device testing prior to the ordering of the printed circuit boards. Overall, Texas Instruments is not only a reliable supplier of quality semiconductors, but also a great resource for information and project support.

6.2 Adafruit - MM

Adafruit Industries is an opensource hardware company that sells a wide range of sensors, microprocessors, drivers, and several other devices applicable for the *SmartLeaf* system. In addition to their store, they provide a multitude of tutorials and present technical information in a clear, concise manner that can simplify and streamline the design process. Adafruit also publishes a significant amount of documentation for their products, which further simplifies project planning and reduces the amount of time needed to get a controlled-component functioning. Reviews for customer support are widely positive, shipping costs are reasonable, and overall, Adafruit is a great resource for technical information.

6.3 Smart Prototyping - MM

This website is a distributor based in China that sells various sorts of electronics for multiple purposes. The *SmartLeaf* greenhouse system will be using their Soil Moisture sensor as well as utilizing their datasheet for reference. While shipping from China comes with additional wait time and cost for transportation, a limited number of distributors provide the exact sensor type compatible for this project; although not the most ideal, limited options and lack of device documentation from other distributors make Smart Prototyping the best choice for sourcing this device.

6.4 Mouser - JG

A variety of electronic components are available on Mouser.com. From resistors and capacitors up to ARM MCUs, Mouser has it all. Shipping costs vary, but usually 2-day shipping UPS costs the same as 5-day shipping FedEx. There are also a bunch of coupons available to make purchasing parts cheaper. When it comes to designing our PCB, part lists can be exported into Mouser and with the help of their new BOM tool, we can find all the components we need without having to do tedious searches. Additionally, datasheets for nearly every component are easily available off the search engine, and large spreadsheets of device options can be imported directly into Microsoft Excel for further evaluation and comparisons.

6.5 Amazon - MM

Perhaps the most recognizable household name of the vendors listed, Amazon is a great source of cheap, fast-shipping equipment. While the general lack of technical device documentation makes this distributor less-than-ideal for the more complex components needed in this project, the vast selection and range of equipment applicable to the project cannot go unnoted. With Amazon Prime available to the team, free 2-day shipping can be taken advantage of for the simpler components needed in the unit. Additionally, their customer support team is fairly responsive in addressing broken or defective equipment upon delivery. The website even offers additional warranties directly through Amazon or the third-party retailers. Overall, what this distributor lacks in documentation and useful information is made up for by cheap prices, fast shipping, and adequate customer support.

7.0 Standards - MM

This section outlines the various standards and regulatory authorities that govern design principals relevant to this project. Standards for the environment, communication, software, power, printed circuited boards, and safety will be discussed and detailed for their relevancy in this project.

7.1 Greenhouse Environment Standards - DM

For us to build a project such as a smart tabletop greenhouse, we would need to discuss the standards that come with it as well. These are general standards that allows for the regulation of chemicals which are a cause of greenhouse environments. These standards are mainly for companies that need to accurately report all emissions that occur due to their business [31]. This is all a standard that should be followed to monitor greenhouse gas emissions carefully. These greenhouse standards are in place such that also if a company wants to construct a greenhouse, they would need to go through the effort of getting paperwork filed to get a permit to do so. There are also careful measures that must be taken in the construction of these large greenhouses, so it makes sense that they need to be regulated in this manner. This also is applicable to government agencies as well as universities. Seeing as how this project that we're working on is a university level project, it's under the assumption that this project also applies to us as well even though we're on a much smaller and negligible scale compared to other companies and government agencies.

7.2 Software Standards - JG

IEEE 12207 defines the lifecycle standard for software systems. The first process is defined as the primary group, which includes acquisition, supply, development, operation, and maintenance. These are the basic needs of software development. The next group includes the supporting processes of documentation, configuration management, quality assurance, verification, validation, joint review, audit, and problem resolution. Organizational aspects include management, infrastructure, improvement, and training.

7.3 Communication Standards - JG

Communication standards are essential to make sure the systems in our device can communicate properly. Standards for communication include WiFi and Bluetooth, and MQTT

7.3.1 WIFI Standards - JG

The IEEE 802.11 standards for WIFI (Wireless Fidelity) describe the different flavors of wireless communication based on a set of criteria. Each network type has a specific policy determined by the speed of the network measured in bits per second (bps) and the frequency the network is carried on in Gigahertz.

Table 19- Common WIFI Standards

Network	WIFI Standard	Speed	Frequency	Note
802.11a	WIFI 2	Max 54Mbps, commonly 6 to 24 Mbps	5 GHz	Not compatible with b or g networks. Old standard that is still in use by many devices.
802.11b	WIFI 1	11 Mbps	2.4 GHz	Made backwards compatible by g networks
802.11g	WIFI 3	54 Mbps	2.4 GHz	Most popular and easily backwards compatible with existing systems
802.11n	WIFI 4	100 Mbps, up to 600 Mbps under perfect conditions	2.4 and 2.5 GHz	Fastest network. Uses multiple frequencies at once to create a very fast network

Combinations of network numbers exist such as 802.11abg, meaning that each of those types are supported by the router. Modern day routers support almost every type of network to ensure they're up to date with the latest standards but also backwards compatible.

The newest standard, WIFI 6 (802.11ax) releasing soon, promises to offer frequencies in the 2.4 and 5 GHz range, as well as offering backwards compatibility. Future improvements are to include support for 1 GHz and 7 GHz frequencies. This network also supports improved security measures with advanced encryption and authorization systems, keeping a user's data safe while maintaining an easy way to connect to the network.

7.3.2 Bluetooth Standards - JG

Bluetooth is a wireless communication protocol standardized by IEEE 802.15. There have been many iterations of Bluetooth versions over the years, but the newest technologies use Bluetooth 4.2 or have recently adopted Bluetooth 5.0.

Table 20- Modern Bluetooth Standards

Version	Speed	Bandwidth	Frequency	Notes
Bluetooth 4.1	24 MBps	100 meters / 300 ft	2.4 to 2.485 GHz	Backwards compatible
Bluetooth 4.2	24 MBps	100 meters / 300 ft	2.4 to 2.485 GHz	Low power and security upgrades
Bluetooth 5.0	48MBps	300 meters / 985 feet	2.4 to 2.485 GHz	No backwards compatibility, requires new hardware.

Experimenting with Bluetooth 5.0 would be a nice addition to our project. This newly released version packs an advanced privacy and security protocol, consumes less power, and has faster and farther connection capabilities.

Utilizing the existing Bluetooth 4.2 wouldn't be a bad idea either. There are currently more devices that support Bluetooth 4.2 than 5.0, and 4.2 is backward compatible with previous versions. For our benefit as a group designing our system utilizing Bluetooth, more projects have been documented for us to research and learn from with Bluetooth 4.2.

7.3.3 MQTT Standards - JG

MQTT version 5 standards were set by Oasis and updated in March 2019. The MQTT protocol operates by exchanging a series of control packets. These control packets are split up into three parts, in the order of a fixed header, variable header, and a payload. Control packets may contain as little as two bytes of data, or as much as 256 MB. MQTT also relies on a TCP protocol for data transmission. The MQTT packet header is shown in the following table.

Table 21: MQTT packet header

Bit	7	6	5	4	3	2	1	0
Byte 1	MQTT control packet type				Flags specific to each MQTT control packet			
Byte 2	Remaining length							

Bits 4-7 of byte one in the MQTT are represented as a hexadecimal integer. The remaining bits 0-3 of byte 1 in the Fixed Header contain flags specific to each MQTT Control Packet type. The control packet types will be important to note for debugging MQTT issues. The following table describes the standard of control packet types for MQTT.

Table 22- MQTT Control Packet types

Name	Value	Direction of Flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server (C2S)	Connection request
CONNACK	2	Server to Client (S2C)	Connect acknowledgement
PUBLISH	3	C2S or S2C	Publish message
PUBACK	4	C2S or S2C	Publish acknowledgement
PUBREC	5	C2S or S2C	Publish received
PUBREL	6	C2S or S2C	Publish release
PUBCOMP	7	C2S or S2C	Publish Complete
SUBSCRIBE	8	C2S	Subscribe request
SUBACK	9	S2C	Subscribe acknowledgement
UNSUBSCRIBE	10	C2S	Unsubscribe request
UNSUBACK	11	S2C	Unsubscribe acknowledgement
PINGREQ	12	C2S	PING request
PINGRESP	13	S2C	PING response
DISCONNECT	14	C2S or S2C	Disconnect notification
AUTH	15	C2S or S2C	Authentication exchange

7.4 Power Standards - MM

The greenhouse unit will be reliant on standard 120V/60Hz general-purpose receptacles in residential and/or commercial settings. The power standards for general-purpose receptacles in North America are set by the National Fire Protection Association (NFPA), which publishes the National Electrical Code (NEC). For larger buildings and structures, the American Society of Heating, Refrigerating and Air-Conditioning (ASHRAE) provides additional Energy Standards for efficient design practices.

7.4.1 NFPA 70 National Electrical Code - MM

Article 210.9 of the NEC specifies the minimum ampacity and size for conductors, which will provide a guideline for the maximum allowable power/load for the greenhouse system. Avoiding excessive current draw will prevent excessive voltage drop in the branch circuits, which causes inefficient operation of electrical equipment and potentially reduce the reliability of motors, heaters, lighting, etc. For non-continuous loads, branch circuit conductors should be designed to have an ampacity not less than the maximum load served. In layman's terms, a device on a circuit cannot pull more current than its breaker rating. If the branch circuit typically supplies power to a

device for more than 3 hours, it is considered a continuous load. For continuous loads or any combination of continuous and non-continuous loads, the minimum branch-circuit rating is to have an allowable ampacity not less than the non-continuous load plus 125% of the continuous load. For example, if a continuous load device is plugged into a dedicated receptacle on a 120V, 20A, single phase circuit, it should not draw more than 80% of the maximum 20A. The maximum allowable load on this circuit is 16A or 1.92 kW.

7.4.2 ASHRAE Standard 90.1 - MM

This standard provides minimum requirements for energy efficient designs for commercial buildings or high-rise residential spaces. This code would be especially relevant if the greenhouse were to be installed in an office or school environment, as there are more stringent energy requirements for these types of spaces. Section 8.4.2 describes automatic receptacle control, which is a process of switching off certain receptacles in areas that are not currently occupied; specifically, this standard requires that 50% of all 15 and 20A receptacles in private offices, conference rooms, printing/copy rooms, break rooms, classrooms and workstations are to be automatically controlled via occupancy sensors or timers. In the case that certain equipment connected to a receptacle in one of these areas requires 24-hour power, the circuit should not be controlled, and it is acceptable to exclude this receptacle from the 50% requirement calculations. Section 8.4.2 is noted as a standard in this paper because electrical designers for these spaces need to know what type of equipment is to be installed to provide the correct control requirements; it should be noted that the greenhouse system requires 24-hour power and should not be placed on a controlled circuit.

7.5 PCB Standards – DM

7.5.1 Component Placement - DM

We need to consider multiple things when placing our components on the PCB, such as the orientation, placement and the organization of our components that we'll be placing on the PCB. We also really need to consider how we're going to approach soldering our components onto the PCB. We need to make sure the components that we will be placing are in the optimal orientation for us to not have any issues regarding soldering. Orientation is not the only thing we need to consider when it comes to soldering, if components are placed too close together, there can be issues when it comes time to place components on the PCB which can make the act of soldering unnecessarily difficult.

To remedy this, we need to make sure that when we place our components, everything will be properly spaced out so that we can avoid all of this. When placement is concerned, we need to attempt to have no issues with components that could be considered through-hole, so we must have the components be placed in an area on the side of the board that we will not have any soldering for those specific components. Organization of our components is also something that is paramount for us to have a simpler time with the actual assembly of the PCB itself. We need to have all our surface mount components be arranged on the same side of our PCB to simplify our assembly and to not cause any major confusion or issues. Below are examples of poor designs as

well as examples of good designs for the PCB layouts considering all these discussed topics of orientation, placement and organization.

7.5.2 Traces - DM

We need to carefully consider how we're also going to be placing our traces for our components in general. We need to carefully figure out how the traces for our power, signal and ground are going to be laid out on the PCB design. We need to investigate the rails for the supply and consider the width that we'll have for our traces themselves. With regards to the power and ground, we need to make sure those centered in our board as well as symmetrical. This is necessary for us to do because we need to give all our components enough room on the PCB itself. Signal traces are important to PCB design, and we need to make sure that the traces that we have on our schematic is accurate to the traces and connections that we're going to have on the PCB design itself or we'll be running into issues with the PCB not functioning properly. Finally, we need to account for the widths of our traces which would be holding the currents that will be flowing to all the components in our PCB design.

We need to account for the widths of these traces because of trace resistance. Trace resistance is very important to consider because if there is an issue that can occur with our PCB it may not have anything to do with the component placement or orientation at all. Current flow will be limited between components due to the traces themselves even if the schematic is perfectly laid out on a PCB board, and this can be due to traces not being the correct width. For us to know what width we need to use in our traces we need to calculate trace resistance. There is a common formula that we can use to calculate this and that is $R = \rho \times \frac{L}{A}$ [32]. R standing for resistance, ρ standing for resistivity of any material, and A standing for the area [32].

7.5.3 Separation of Components and Mitigating Heat Issues - DM

Another thing we need to take a close look at would be the attempt to mitigate issues regarding how close we'll be having the power and ground in our PCB design. We need to make sure we have adequate separation of these two to avoid any major conflicting issues such as the PCB not functioning properly [33]. All of this is just for mitigating issues that may arise with spikes in current as well as voltage. When it comes to taking control of issues regarding heat with our PCB, we need to look at things outside of the PCB design first to minimize the potential of this problem occurring in the first place. We need to make sure all the components that we select have had their datasheets carefully analyzed to make sure that their temperatures are within the specification of our design. Also, as an added measure of caution it would be wise also for us to also add an addition to the PCB board itself to minimize the heat and temperature spikes that may occur. This would be a "thermal relief", which would reduce the risk of thermal issues occurring in our design. Below is an example of a pattern of a thermal relief that could be expected in our PCB design. Though having a thermal relief can be an optional step because if our design was made properly and we have taken all the precaution measures to ensure that we will not be overheating do to an overwhelming increase in current flow, we should be in a good space when it comes to heat management.

Even though thermal relief is an option for us to manage heat, there are other avenues for regulating heat in our PCB. Heat regulation can come in the form of just simply purchasing a thicker board. Though purchasing a thicker board isn't the only thing that should be done to regulate the heat

flowing through the traces in the PCB. Another thing that needs to be done on top of this is also placing the traces in the center of the PCB [34]. Placing the traces in the center would allow for the heat flowing to dissipate in a much wider area instead of being constrained to a smaller portion of the board potentially burning the traces or other components on the PCB itself. Also, in addition to doing all of this to manage heat, we'll also incorporate thermal interface materials or TIM to manage it.

Adding thermal interface materials along with having a thicker PCB with properly placed traces will be very beneficial for us to avoid the potential worry of accidentally burning out the traces on the PCB. Another thing that would need to be monitored in terms of our PCB would be the surrounding heat as well as the heat being transferred into the PCB via current flow because if we simply consider the heat being flown into the PCB and neglect other sources of heat that could be detrimental to the project, then the PCB might burn out in that manner. We'll also be considering the extremes when it comes to surrounding temperature so that shouldn't be too much of a cause for concern.

7.6 Design Standards - AL

Making sure that your design abides to existing standards is imperative to designing and creating a new product. Implementing these standards correctly ensures that the new product will be compatible to existing technologies and protocols that are currently in use, thus ensuring that the functionalities of the product will work, especially if the product is to interact with external technologies. These standards require us as engineers to take the proper steps to ensure that the product that we are making has a positive impact on the user.

The main standards for designing a project takes place in many forms and has many constraints but can be broken down to nine main branches; economics, environmental, sustainability, manufacturability, ethical, health and safety, social, and political. To expand more on these subjects, we will briefly provide the importance of abiding to each of these branches while in the process of designing a project.

1. **Economics:** Abiding to the budget should be one of the main constraints of a project. Having the team overspend and not distribute funds correctly destroys the cost effectiveness of the design. In order to reduce the budget, the team's spending mainly consisted of the mindset of out-of-the-pocket expenses as we couldn't find a sponsor for our project. Our team's mentality was to provide the highest of quality of product we could create while minimizing the effect of the project on our wallets! To do this, there were many improvements and adjustments of the components that we had in store for the final project to reduce the cost of parts when finding of an equivalent component, choosing test equipment that could double up and have multiple uses, i.e. the MSP432 Launchpad Evaluation kit that could double as testing our the capabilities of an ARM processor that could handle more data and as well as being able to test out the capabilities of the LCD screen, Bluetooth module, and internet capabilities.
2. **Environmental:** With environmental constraints, we were positive that we wouldn't break any current laws or standards that have to do with polluting or corrupting the environment. If anything, our product would optimally reduce waste, growing plants to their maximum

potential, and consume minimal power to reduce the negative footprint this leaves on the environment.

3. **Sustainability:** With the Smart Tabletop Greenhouse, we had the project designed with sustainability in mind. We limited the complexity of our assembly and minimized the chances of having a discontinued component by choosing more popular parts. To come with the Smart Tabletop Greenhouse we created an instruction manual for the system so that plants within the environment could be switched out once they were grown to their highest yield or if the user wants to insert another plant within the environment. If the need to replace a component of the Smart Tabletop Greenhouse were to ever occur, we also were sure to create an easy to understand user manual to instruct anyone on how exactly each component of our assembly fits in to the system.
4. **Manufacturability:** For the easy of manufacturability, most of the system was made in a lab setting. The degree of complexity of our system will be explained within the User Guide and have an easy-to-understand step-by-step instructions on how to assembly our design so that others could easily replicate and optimize our system currently.
5. **Ethical:** The ethical values we were to use for our design were more or less the ones that are used at the University of Central Florida and other nationally recognized educational institution during classwork. We were sure to not have any portion of our design plagiarized from an outside source and made sure to cite the sources that we used to optimize our design. The hopes we had for our project was for the plants growing within the system to be legal, and not to have any illicit and illegal vegetation to be used for the system for any unethical purposed. This design will be handed off to individuals who are trustworthy and are in good standing with the government as to prevent any possibility for this to happen.
6. **Health and Safety:** The properties that are to be used in this portion of ethicality are to be referred to the health and safety of the individuals creating and using this environment. Since we are dealing with electrical devices and liquids within close proximity of each other in our system, we had to ensure that none of the sensitive equipment was exposed to any of the water to provide nourishments for the plants and ensured that the water running off of the plants after they have been nourished are well away from the electrical components. When constructing the Smart Tabletop Greenhouse, one should refer to the User Manual as to make sure that each component is properly installed. If any component within the system is installed incorrectly, there is a chance for the plants within the system to not be optimally grown or in the worst case, the system would not operate at all.
7. **Social:** Social ethical values encompass creating a design that was feasible enough to provide us, the creators, a challenging project, but not so much that we would have to spend more than a few hours per day to work on the Smart Tabletop Greenhouse. As students we have to consider scheduling our other education responsibilities outside of this project. Many of us have at least one other class to work on besides senior design so we have to time manage our given tasks in the group to be able to accommodate for these responsibilities. On top of our education responsibilities, many of us have jobs and internships to take into account of, so we had the task to optimally design our project all the while of juggling our schedules.

8. Political: For political obscurities that could occur in the talk of ethics, the only concerns that we have is if the user were to grow illegal vegetation within the system. This system is intended to be strictly a tool that abides that laws that we have in the United States. Other than that, should have no problems in being used as a usually educational tool as well as a design that lawful hobbyist can try in the future.

8.0 Safety Standards - MM

There are several consumer electronics certifications and standards that need to be met in order to sell a product on the US and North American market. Additional standards are to be considered for workplace safety and ground fault protection in the presence of water systems.

8.1.1 FCC Requirements - MM

The Federal Communications Commission (FCC) primarily enforces two major sets of requirements in regard to consumer electronics: Electromagnetic Compatibility (EMC) testing and Radio Frequency (RF) testing [35]. The former involves the product undergoing a series of tests to ensure that the electromagnetic fields emitted by the device do not exceed limits that could endanger life or introduce significant electrical interference to an environment. RF testing is similar in that it involves measuring the impact of electromagnetic fields emitted, but it takes a deeper look into the fields within the RF spectrum. Fields within this spectrum are typically emitted by Bluetooth and WiFi modules within a device. The FCC will determine if the transmitted frequencies are appropriate for a device's use on the market.

8.1.2 Miscellaneous Safety Testing - MM

There are often instances where safety concerns are present for certain aspects or intended uses of a product, but no widely available standardized test exists to quantify the specific risk involved. Products that fall into this category should be sent to specialized safety testing labs, such as UL, Intertek, SGS, or Met Labs to get expert guidance and gain clarity on legal issues [35].

8.1.3 NRTL Program - MM


The Nationally Recognized Test Lab (NRTL) program was implemented by the Occupational Health and Safety Administration (OSHA) as an authority over the environments in which government and private employers conduct work. This program would be fundamental if a product is to be marketed to office environments; however, it does not regulate devices for consumer use.

8.1.4 IP Code - MM

The International Protection (IP) Marking code is defined in the International Electrotechnical Commission (IEC) standard 60529. This code classifies and compares the amount of protection provided by mechanical casings and electrical enclosures against intrusion, dust, accidental contact, and water [36]. This marking is often associated as a waterproof identifier for a consumer electronics (typically mobile devices), however users should be aware that not all IP ratings

describe resistance to water submersion. Following the ‘IP’ marking are two mandatory digits; the first of which indicates solid particle protection, the other displaying the mechanical impact resistance. Table 23 displays the Ingress Protection ratings of electronic enclosures.

Table 23- Ingress Protection Ratings Guide

Solids		Water	
1	Protection against a solid object greater than 50 mm.	1	Protected against vertically falling drops of water. Limited ingress permitted.
2	Protection against a solid object greater than 12.5 mm.	2	Protected against vertically falling drops of water with enclosure tilted up to 15 degrees from vertical. Limited ingress permitted for 3 minutes.
3	Protection against a solid object greater than 2.5 mm.	3	Protected against sprays of water up to 60 degrees from the vertical. Limited ingress permitted for three minutes.
4	Protection against a solid object greater than 1 mm.	4	Protected against water splashed from all directions. Limited ingress permitted.
5	Dust protected. Limited ingress of dust permitted. Will not interfere with operation of the equipment, 2-8 hours.	5	Protected against jets of water. Limited ingress permitted.
6	Dust Tight. No ingress of dust, 2-8 hours.	6	Water from heavy seas or water projected in powerful jets shall not enter the enclosure in harmful quantities.
Rating Example: 		7	Protected against the effects of immersion in water between 15 cm and 1 m for 30 minutes.
		8	Protected against the effects of immersion in water under pressure for long periods.

8.1.5 UL 8750 LED standard

The UL 8750 standard requirements are important for engineering safe luminaire or other kinds of lighting that operate between 400 to 700 nanometers. This scope encompasses the broad spectrum of visible light, thus LEDs fall under these standards as well. Our smart garden may include LEDs for growing purposes, or just for visual appeal. Either way, these are important standards that apply to our project.

8.2 Sensor Standards - MM

The Institute for Electrical and Electronics Engineers (IEEE) has published a set of standards for sensor performance in IEEE Std 2700. This document provides a basis for sensor terminology and units, as well as limits and conditions of operation for common devices such as the accelerometer, magnetometer, gyrometer/gyroscope, barometer/pressure sensors, hygrometer/humidity sensors, temperature sensors, ambient light sensors, and proximity sensors [37]. For the purposes and scope of this project, standards concerning temperature and humidity sensors will be discussed in this section.

8.2.1 IEEE Standard for Sensor Performance - MM

Specifications for digital humidity sensor standards are outlined in this document, detailing the various testing that should be conducted prior to market release in order to determine the effectiveness of the meter in question. For a digital humidity sensor to be considered sound, a full-scale range, or a peak-to-peak measurement range of the sensor, needs to be defined. Testing to determine the measurable range is in-depth, encompassing tests for each selectable mode, after mechanical shock, and throughout the expected lifespan of the device. Digital humidity sensors are to measure air moisture content in units of %RH (percent relative humidity). Relative humidity measurement accuracy over the entire measurement range is to be plotted and compared to the expected plot for its ideal values. Additional testing and data is to be presented by the sensor manufacturer for its digital bit depth, sensitivity, noise, current consumption, integral non-linearity, response time, and stability.

This standard also outlines the testing and data needed to validate the functionality of temperature sensors. A key concept noted throughout this paper is ambient temperature versus the device temperature, and the importance device efficiency and insulation to prevent heat of the circuitry impacting the sensor reading. All testing discussed above for the humidity sensor is conducted for the temperature sensor. IEEE notes that the standard unit for temperature is the metric unit in Celsius, which allows for widespread and international applications.

9.0 Project Design - AL

This section will serve as a general outline in how we decided to implement all of our gathered research into a coherent design to present as our product.

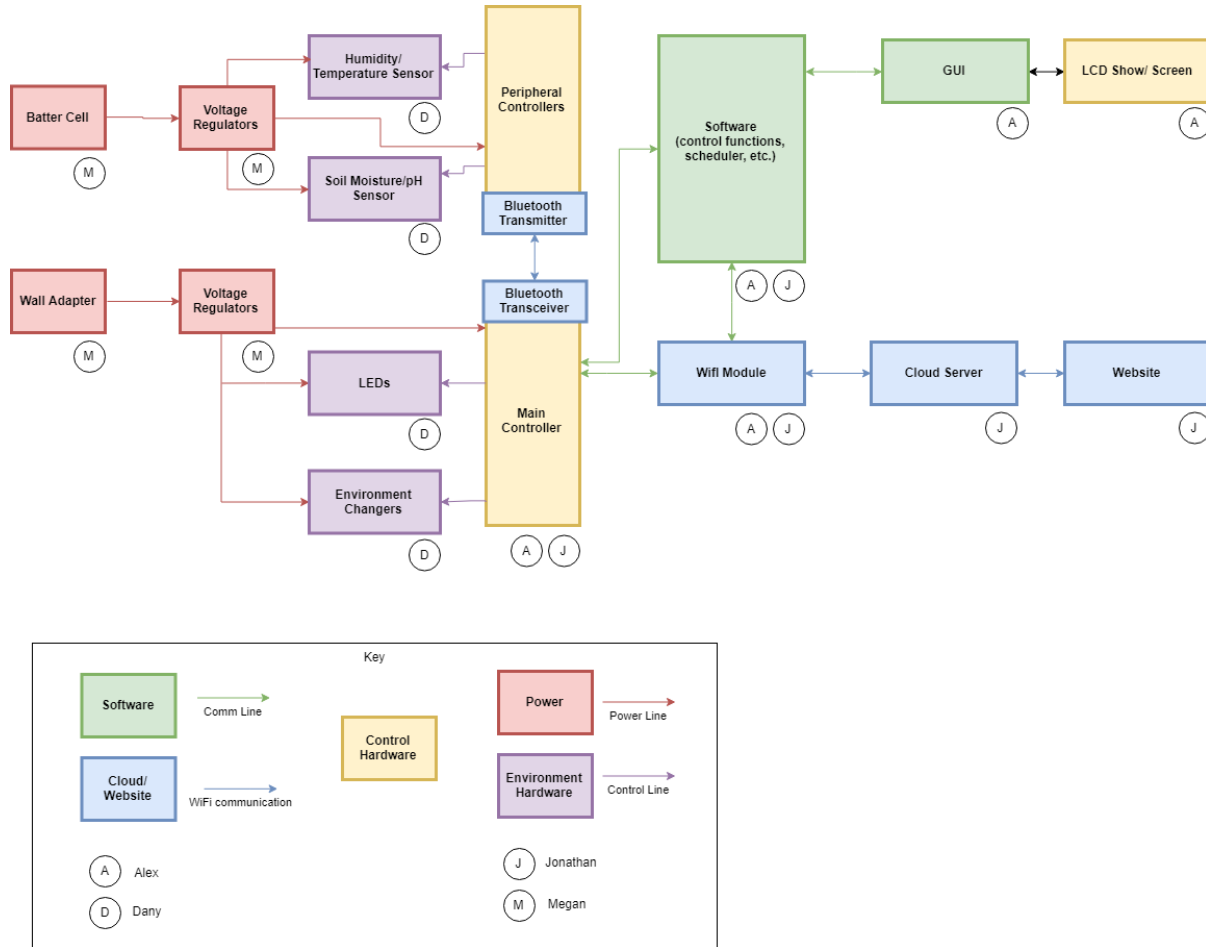


Figure 18: Block Diagram of Smart Greenhouse

9.1 Design Goals - AL

Above is the what should be the general relationship of all the components to each other. One can see the left half of the block diagram shows the hardware connections while the right half shows all the software connections. The general functionality of our product is as follows:

9.1.1 Hardware Goals - AL

Power starts from the either the battery pack we have planned or a wall outlet depending on if the battery has enough charge to drive the system. The power will then go into regulators that will convert the power into its required specification to drive each individual component of our system. Once all the sensors have their required power, they will begin sending off their data and

measurements to its microcontroller. After every few minutes or when the user requests for it, the main controller will request the observations of a certain plant within the system via Bluetooth, the child microcontroller will then transmit its sensor information via Bluetooth for the parent microcontroller to process the information and send it off to the Cloud when the user permits it. The parent microcontroller will have control of the environment changers such as the grow lights, ventilation fans, and water pump to be able to optimize the ecosystem's growth patterns and be the main interface for the user with its GUI. We also plan to have an idle state LED show, that will add an aesthetically pleasing feature to the whole project.

We also plan to implement two children microcontrollers that will be able to send information back to the parent controller by having two separate PCB boards that will be able to stick in to the side of a soil base. These children PCBs will be planned to be powered via coin cell batteries that and will be well within the max range of 10 meters of the Bluetooth modules to be able to communicate back to the parent microcontroller. We also plan to have this PCB inside of a plastic container as to protect it in the case that water were to accidentally be sprayed by the water pumps

9.1.2 Software Goals -AL

For the planned software that we want to implement; we want to implement a sort of optimized code for the children microcontrollers that will be able to handle collecting data from its connected sensors, acknowledge if any of the sensor aren't working and sending off data once every few minutes via Bluetooth. With this in mind, our parent microcontroller would have to be able receive the data from the children microcontrollers at all times and send the information to the cloud, be able to get information from the cloud and be able to adjust the environment modifiers based on the feedback it gets from the cloud. We would also need to make a friendly GUI interface that would be able to access any of the environment modifiers and be able to change them on demand if the user wishes, be able to access any of the children microcontrollers' data as necessary and provide options to adjust the LED light show on the exterior of our system.

Other things to keep in mind are the fact the Bluetooth modules have their own way of communicating to each other. With the BLE modules that we chose out to use our design, we have to be aware of all the AT commands that we set up our software with as to prevent the children microcontroller and parent microcontroller to miscommunicate with each other. This could be from either having the child BLE module awake while the parent Bluetooth module is asleep and visa versa, having the incorrect baud rates to be able to communicate with the right timing, etc.

9.2 Power Supply & Rail Design - MM

This project will be utilizing a 12V, 10A, 120W DC power supply to step-down mains power, but switching regulators will be needed to provide the appropriate voltage and current to each device. To get a general idea of how the rails will be configured, the Webench Power Architect tool from Texas Instruments was used. The main sources of current draw for the system – fans, humidifier, pump, LEDs, and LCD display – were entered, and the image shown below in Figure 19 was produced.

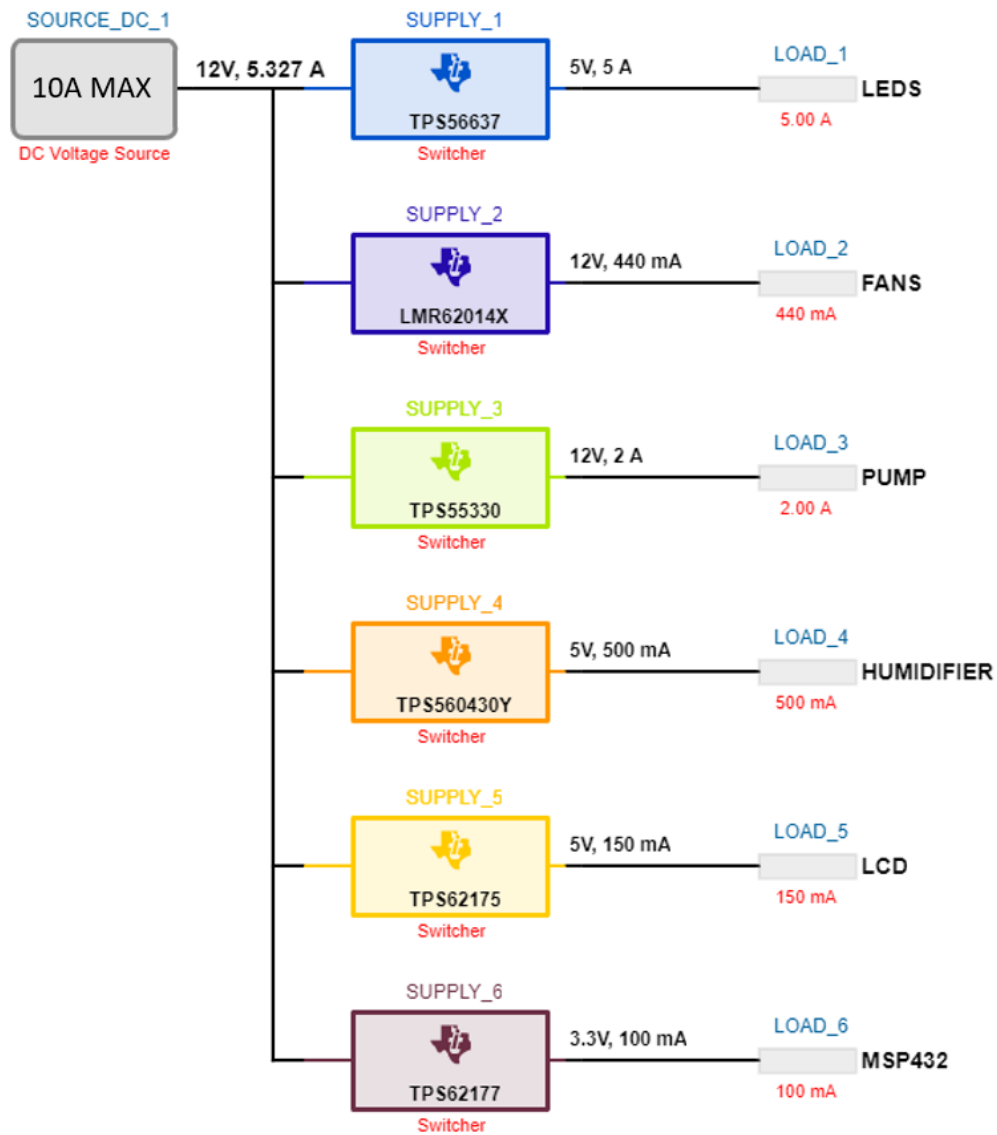


Figure 19- General power rail design

The Webench Power Architect generated a 5-rail output design with three switcher types: the TPS560430Y Synchronous Step-Down Converter for low power loads like the fans, LCD, and humidifier; the TPS562231 Synchronous Step-Down Buck Converter for the mid-tier power needs of the pump; and for the largest load of the LEDs, the TPS40305 Synchronous Buck Controller. Having five separate power rails and switching regulators may limit some efficiency in terms of space on the PCB and power consumption, but this configuration ultimately provides a clear method of control and protection for each mechanism.

9.2.1 TPS56637 Synchronous Buck Converter - MM

The TPS56637 is a cost-optimized buck controller used in high performance DC-DC converters. The device has a wide input range of 3V-20V [38] and the ability to support high load-current applications; these characteristics make it ideal for power regulation to the LEDs.

While this device has higher output capabilities, more external components are needed to support the design. The additional circuitry is to be optimized to limit space occupied on the printed circuit board and minimize potential interference. Regardless, the design flexibility, power efficiency, and safety characteristics make the TPS40305 Synchronous Buck Controller ideal for the high-power device requirements present in this project. The schematic shown in Figure 20 displays the circuitry that will be used for this power rail, delivering 5V and 5A to the LEDs.

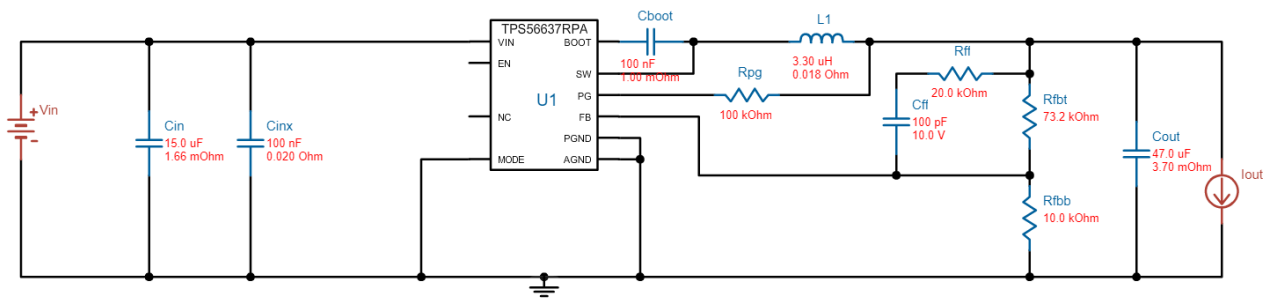


Figure 20- LED power rail circuitry

9.2.2 LMR62014X Simple Switcher - MM

The LMR62014 is a simple switching step-up regulator with an input voltage range of 2.7V-14V, and output characteristics up to 20V and 1.4A [39]. Highly efficient with a 90% duty cycle and additional current protection to the device it supplies, this switcher is an ideal fit for supplying the fans. Additional benefits include a compact form and a design fully enabled for Webench Power Architect support from Texas Instruments. The schematic in Figure 21 shows the circuitry that will be used for this power rail, delivering 12V and 0.44mA to the fans.

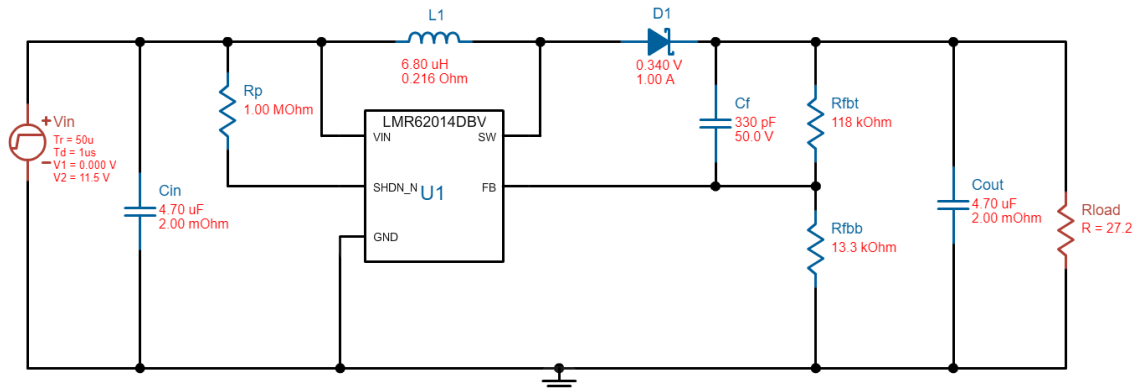


Figure 21- Fan power rail circuitry

9.2.3 TPS55330 Boost/SEPIC/Flyback DC-DC Regulator - MM

The TPS55330 is a switching regulator capable of operating in several common switching-topologies. This device has an input range of 2.9V-16V and can support an output up to 22V [40]. Capable of driving 5A of load current, this device is also able to regulate output voltage with current mode pulse width modulation (PWM) control, the switching frequency of which can be set by an external resistor/external clock signal. These characteristics make it ideal for regulation of motor or pump systems, as fewer external controls will be needed to send PWM's to vary the mechanical output of the pump to an ideal operating condition.

While the main power supply operates at 12V, the voltage will not need to be stepped up/down; however, this regulator offers various overcurrent protections such as a programmable soft-start function to limit inrush current during start up, cycle-by-cycle overcurrent limit, and thermal shutdown. These added protections, as well as PWM capabilities for control, make this device ideal for supplying the pumps. Figure 22 shows the circuitry to be used alongside the TPS55330 to deliver 12V/2A for the pumps.

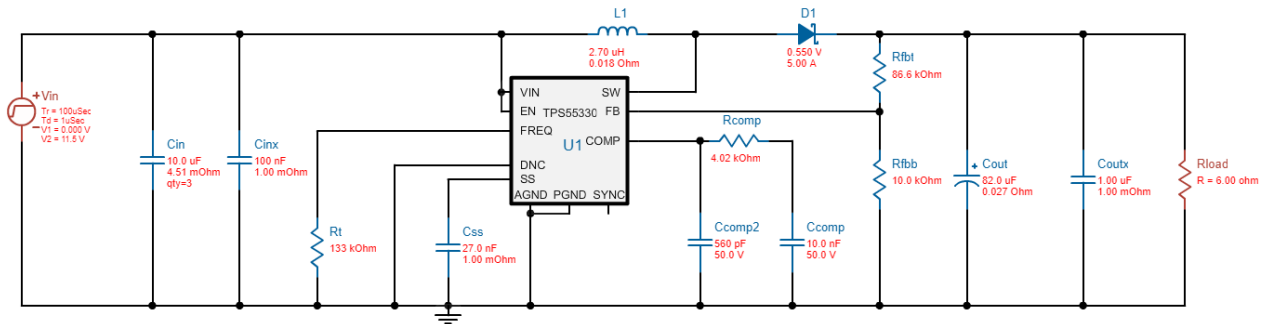


Figure 22- Pump power rail circuitry

9.2.4 TPS560430Y Synchronous Step-Down Converter - MM

The TPS560430Y is a simple DC-DC step-down converter capable of driving 600mA of load current at a constant 5V output [41]. This makes it a good fit for the fans and humidifier as they have lower power requirements. The wide input voltage range and dropout characteristics are ideal for device protection in the case that voltage from the power supply fluctuates. This device can maintain a 5V output up to a maximum input of 36V.

This device has a peak efficiency when operating at an input voltage of 8V rather than the 12V used in this project, but the efficiency differential greatly decreases when the device is operating near the maximum load current, which will be the case for the humidifier circuit. Figure 23 below shows the circuitry to be used in conjunction with the TPS560430Y in order to deliver the necessary 5V/500mA to the humidifier power rail.

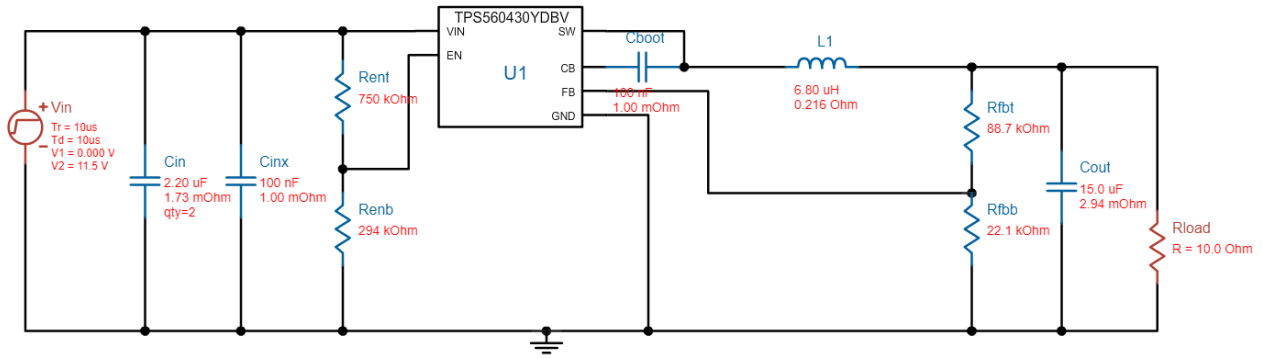


Figure 23- Humidifier power rail circuitry

9.2.5 TPS62175 Step-Down Converter with Sleep Mode - MM

The TPS62175 is a highly efficient synchronous step-down DC/DC converter with a wide input range of 4.75-28V [42]. This device is ideal for stepping down 12V supply rails and can provide up to 500 mA output current. The characteristics of this device make it a great contender for voltage regulation of the low-power devices in this project, such as needed for the LCD display screen. The circuit shown in Figure 24 below displays the various components and component values used alongside the TPS62175 Step-Down Converter to deliver 5V/150mA to the LCD power rail.

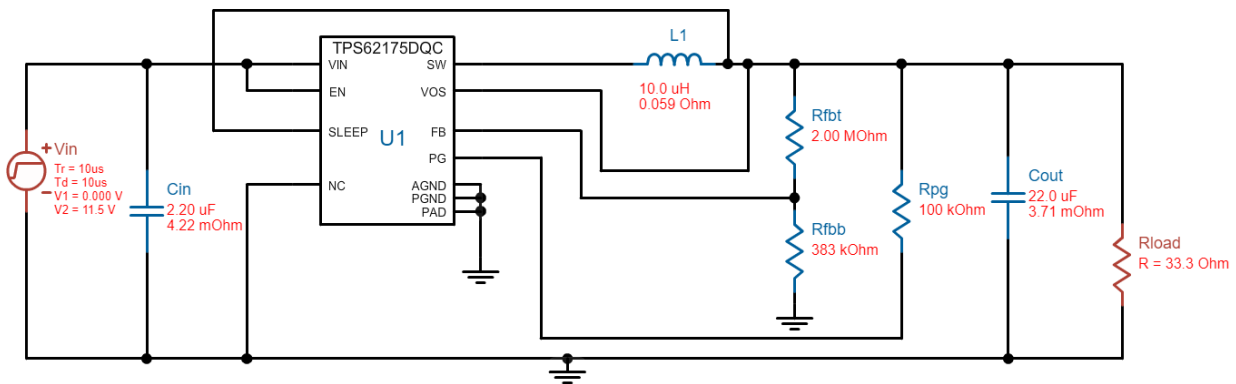


Figure 24- LCD display power rail circuitry

9.2.6 TPS62177 Step Down Converter with Sleep Mode - MM

This device comes from the same family as the TPS62175 described above, but with a fixed output of 3.3V [42]. The TPS62177 was designed with low-power microcontroller needs in mind. The MSP432 microcontroller has a recommended V_{CC} voltage of 3.3V, making this step-down converter the ideal choice for regulating its power supply. The circuit shown in Figure 25 below displays the components used alongside the TPS62177 to deliver 3.3V/100mA to the MSP430 microprocessor.

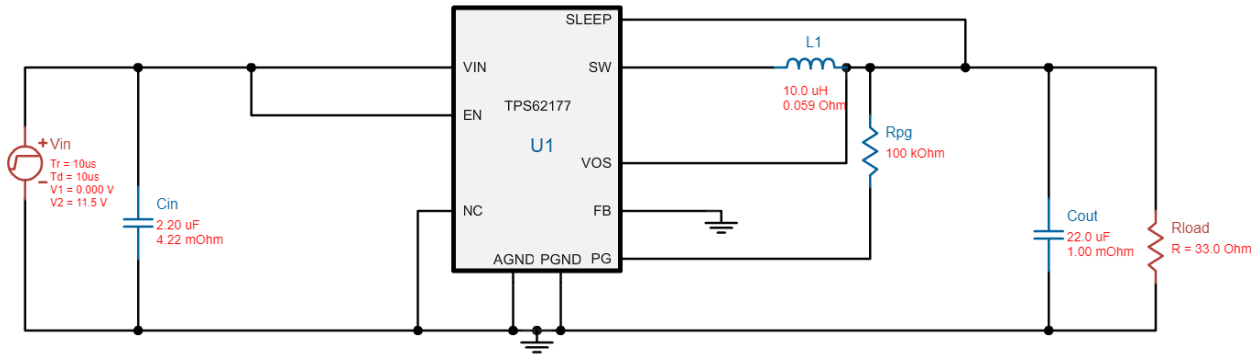


Figure 25- MSP430 power rail circuitry

9.2.7 Efficiency and Overall Power Demands - MM

It is important to consider the ramifications of the power rail designs. In particular, efficiency needs to be evaluated, as power dissipated by the circuitry will be transformed into heat. Excessive heat lost by the system could potentially damage the equipment. In addition to power efficiency, efficient use of space on the PCB should be considered prior to design. Table 24 below evaluates these characteristics. The power supply circuitry has a combined power dissipation of 3.28W, total efficiency of 94.63%, and a total footprint of 811mm².

Table 24- Power Rail Efficiency Data

Model #	V _{in}	I _{in}	V _{out}	I _{out}	Power Diss.	Efficiency	Footprint
TPS56637	12V	2.21A	5V	5.00A	1.51W	94.3%	170 mm ²
LMR62014X	12V	0.50A	12V	0.44A	0.26W	95.3%	73 mm ²
TPS55330	12V	2.28A	12V	2.00A	1.04W	95.9%	287 mm ²
TPS560430Y	12V	0.24A	5V	0.50A	0.34W	88.0%	95 mm ²
TPS62175	12V	0.07A	5V	0.15A	0.08W	90.9%	96 mm ²
TPS62177	12V	0.03A	3.3V	0.10A	0.06W	85.3%	90 mm ²
				Totals:	3.28W	94.63%	811 mm ²

9.3 Embedded Software Design - JG

To monitor the plants, we want to develop sensor nodes that comprise of our various sensors such as temperature, humidity, and soil moisture controlled by a low-level MCU like the MSP430G2553 that can communicate with a more powerful master ARM MCU. The star architecture is ideal for this project since the sensor nodes for each plant will all connect to the main MCU. From the main MCU, data from the sensors will be pushed to the cloud to be stored in our database and monitored on our GUI. Each sensor node will consist of a battery to keep it running wirelessly, a Bluetooth module to connect to the master, the sensor equipment we use to monitor the plants, and a USB port for configuration and debugging. The sensor node is shown in the figure below.

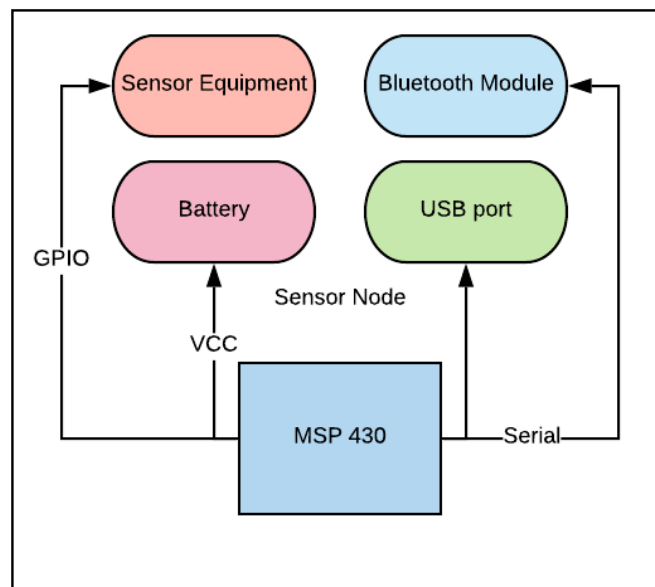


Figure 26: Plant sensor node architecture

Ideally, our sensor node will operate in low power mode for most of its operation. Interrupts will wake the sensor up when it is time to take a sensor reading. This conserves the most power as the MSP430G2553 is optimized for low power consumption. When it is time to take a sensor reading, an interrupt will occur to trigger a sensor read function and then create the JSON message to be published. The Bluetooth connection will need to be established before data gets sent. Enabling Bluetooth connection to stay on during times of the day will allow the user to make manual requests for the data by publishing messages to the sensor node from our webserver or device GUI to retrieve the sensor data. Sensor modules will be battery powered to stay wireless. Since the device will run in low power mode most of the time, battery power should be good enough to last the user for at least a year of operation.

9.3.1 Communication Design - JG

Communication is centered around a master ARM microcontroller edge device that is suited with an integrated WiFi module and Bluetooth transceiver. Bluetooth will allow the main MCU to gather data from sensor nodes, forming a star network. Instead of wirelessly connecting all of our sensor nodes, we'll push and receive all messages from the main MCU. Once our data is collected, we use PubNub as an MQTT broker to publish the data to a channel that our web application is subscribed to. From there, a JSON string containing our data can be unbundled and displayed for the remote user. Data will be stored in a database for historical reference. To take advantage of external online APIs, our device will trigger text alerts to the user when it needs attention, such as water levels being low. The following figure displays how the communication works between the master MCU, sensor nodes, and online resources.

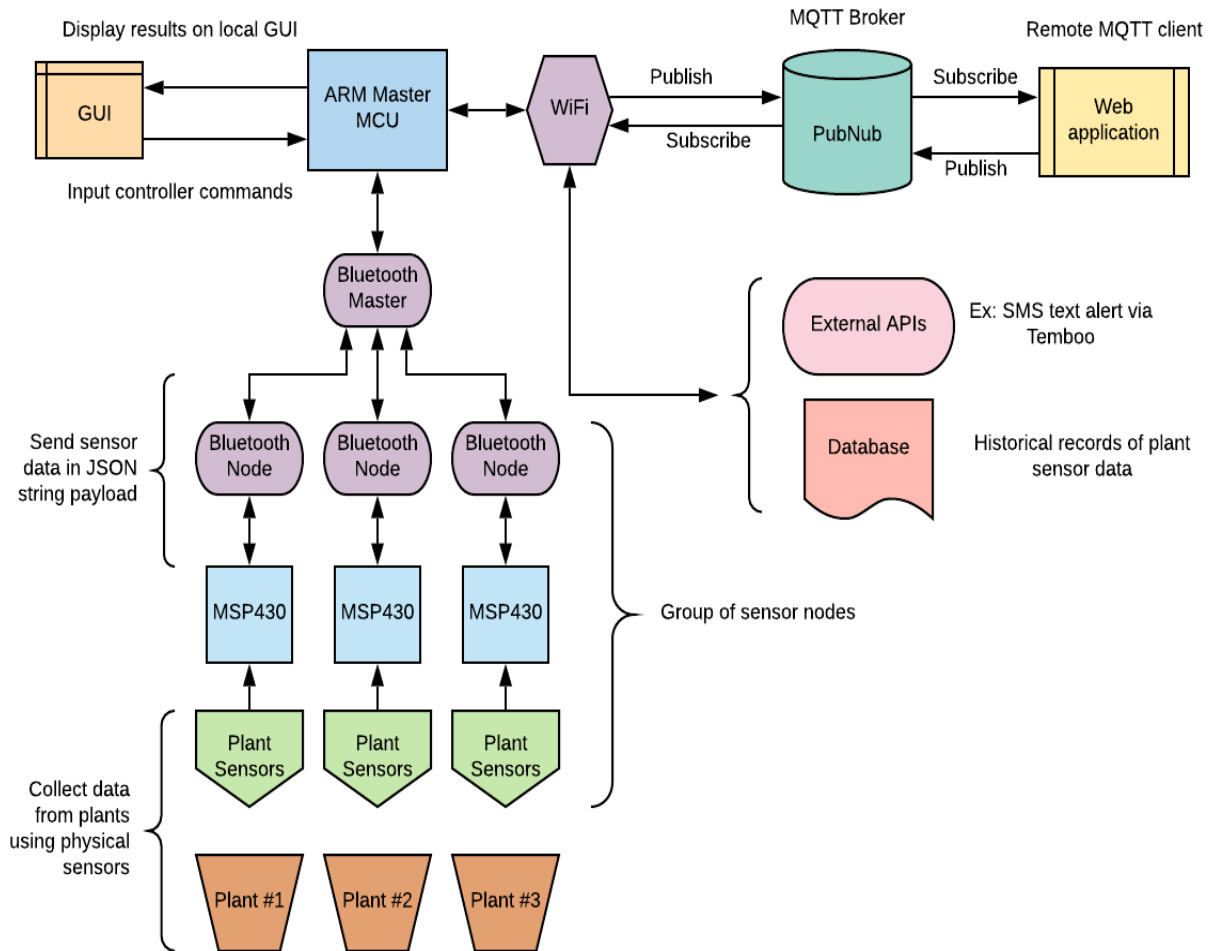


Figure 27: Communication Block Diagram

9.4 GUI Design - AL

The GUI will provide the user with a touch screen interface for monitoring sensor values from the plants and issuing commands to the device. The optimal design for the GUI is to make it as user-friendly as possible, having as many unambiguous instructions for each button, have a help window on each of the GUI page to describe what each function on the page does, having descriptive error messages when the software gets unexpected data from sensor/ environment changers, and having a descriptive user guide to explain and help troubleshoot things that the GUI is intended to do.

9.5 PCB Design - DM

Prebuilt PCBs like the Texas instrument Launchpads are great for fast prototyping. However, these boards are often over-equipped for specific designs since they're made to support a variety of purposes. Since we want our circuitry to be as compact as possible, breadboards are out of the question. PCBs also make circuitry easier to wire than solderable breadboards, since we won't need to make solder traces to connect components. Our PCB will be designed using software, which we can then give the finished design to a manufacturer make for us. Designing a PCB is also a senior design requirement, which means it will be an essential part of our project.

9.5.1 PCB Planning - DM

The first step in designing our PCB is planning out our circuitry on paper, and then porting that design over to a PCB design software. We decided on using Kicad as the software for our *SmartLeaf* project. For our design we have two child MCUs and one parent MCU, so this requires three different schematics and PCB layouts. The main challenge with designing these PCBs stems mainly from the design of the parent microcontroller and the associated peripherals. This is since the parent MCU the MSP432P401R along with multiple peripherals have their own associated power circuit. Before power reaches any one of the peripherals as well as our MSP432 in our PCB, they need to pass through their own individual regulator first.

9.5.2 Schematic Creation – DM/AL

Since we are using Kicad, we need to utilize the Eeschema schematic software in order to create this. In the schematic for our parent microcontroller, there was simply not enough room to fit everything on one schematic sheet, so in order to save space, we utilized the built-in tool to create hierarchal sheets within the schematic as well as using the function for global labeling. Global labeling allows us to connect pins on multiple components with each other without the need for an extensive amount of wires creating a clunky mess on the schematic itself. First schematic we're going to talk about is the one for the parent containing the six individual regulators within it. Utilizing the information from the datasheets for all the components, we can figure out where to allocate each pin, of course since this is the parent microcontroller the MSP432 it contains many GPIO pins which we utilize for all our peripherals. Due to the size of the symbol on the schematic, drawing individual wires and connections would not be very efficient and cause more confusion than anything, so as you can see global symbols are placed on every single pin to signal a connection to another component. On the left, there are six hierarchal sheets, each labeled with a name to their corresponding peripheral component to connect to. At first while using this function, there was a little issue regarding the connection of pins between sheets, but this is due to two

different functions in the Eeschema program with near similar functions. There are global labels which connect multiple components with each other in the schematic, but if a hierarchal sheet is created, we need to utilize the tool known as a hierarchal label. Using these labels within the sheets allow us to connect the regulators in those sheets to the components within the main sheet of the schematic. Another issue regarding the regulators are the footprint and symbol availability. Some of the regulators for some of the circuits did not have an available symbol or footprint in any of the libraries to add to the schematic. Those were minor inconveniences which are mitigated by creating them from scratch in Eeschema. Kicad attempts to make creating these symbols and footprints easy with their program. Of course, the datasheets needed to be looked at in order to create these symbols and get the symbol size and pinouts correct. In the center of the main sheet for the schematic is our parent microcontroller which is the MSP432 that we're using, and to the right of the microcontroller are all the peripherals and necessary things we'll need to attach to that MCU. On the right we have connections for the LCD, power rail, debugger, LEDs, fans, water sensor, water pump, and the parent Bluetooth module, the CC2564MODA as well as a crystal to take care of the clocks on the microcontroller. The debugger is needed to be manually added to the schematic because the microcontroller does not have a built-in debugger and the addition of a schematic like this is necessary for debugging purposes of the MSP432 microcontroller. Everything on the connectors are connected to their corresponding pins via the global label in Eeschema. Creating these schematics required us to be carefully analyzing every single pin on the microcontrollers as well as the peripherals and making sure that all our components are being connected in the proper manner.

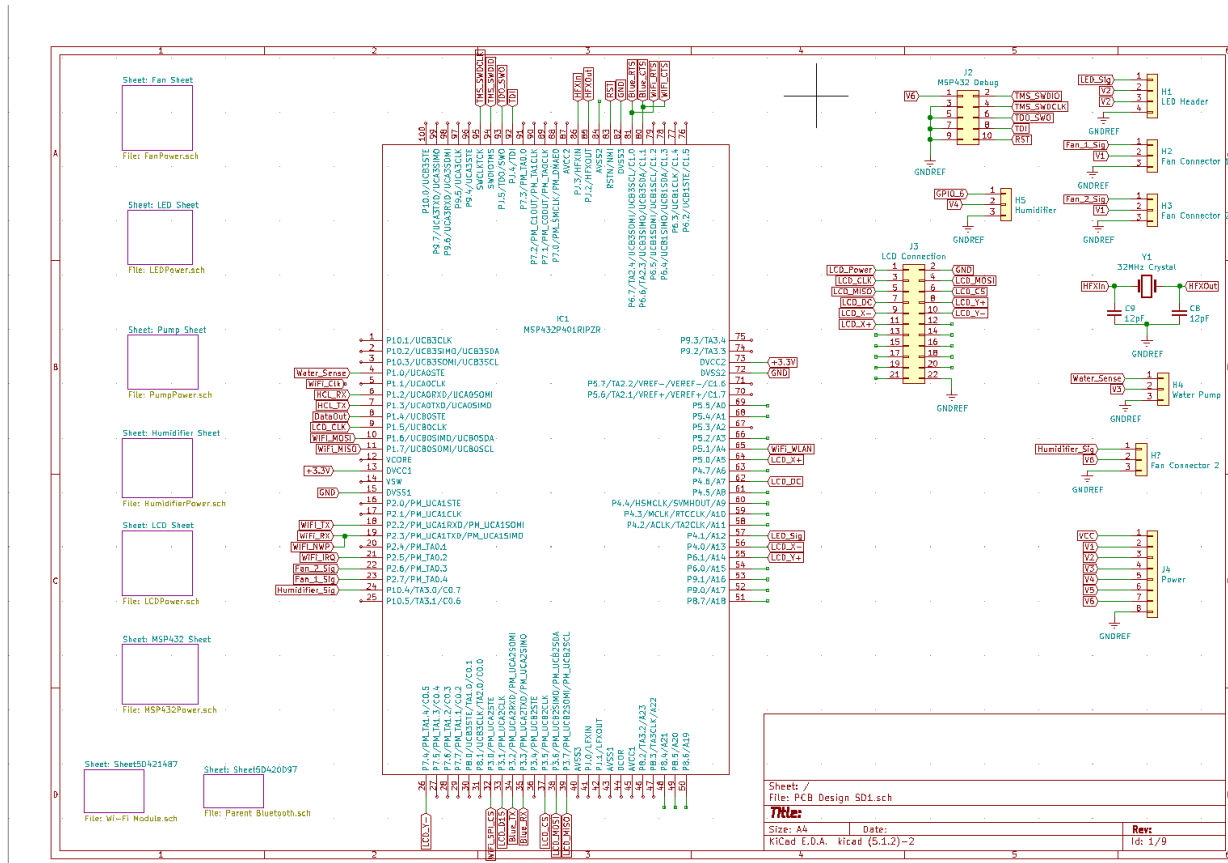


Figure 28: Schematic for Parent MCU

9.5.2.1 Hierarchal Sheet #1 for Fans - DM

For the LEDs, when it comes to power, we needed a simple switcher for our purposes. This regulator is the LMR62014XMF. Now when using Kicad we had issues finding certain components within the built-in libraries in the software. Usage of websites such as “SnapEDA” as well as “Componentsearchengine” were necessary in order to find symbols for a good majority of the regulators as well as their associated footprints for that specific symbol. Though in this case, Kicads libraries contained what we needed with regards to the symbol for the LMR62014XMF and the associated footprint for it. So, we didn’t blindly choose the footprint simply based off the fact Kicad had it available within their libraries, we double checked the information that was already available on the footprint with the information that was available on the datasheet for the switcher.

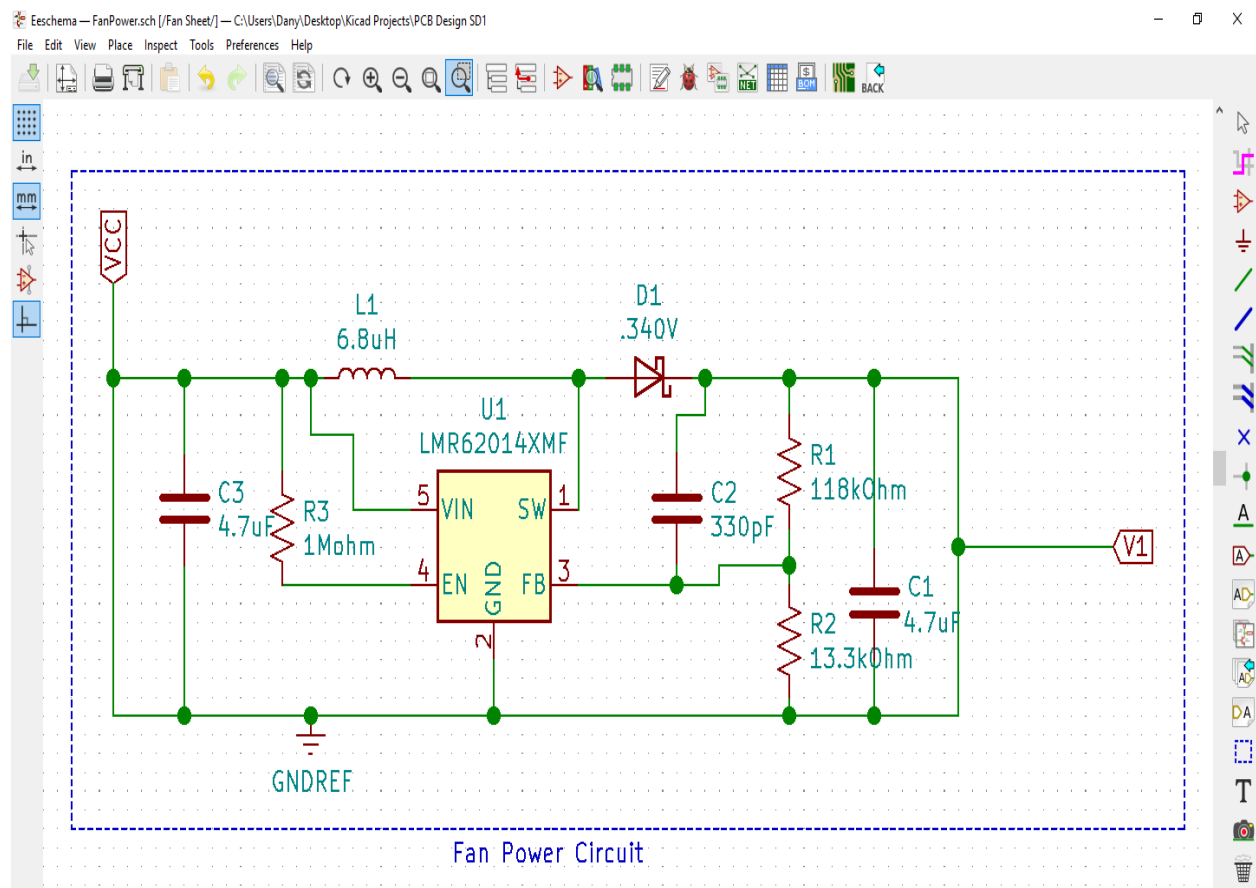


Figure 29: Fan Power Circuit with Connections via Global Labels in Kicad

For the footprint, we already decided we’re going to be using surface mount for nearly all our components on all of our PCBs for simplicity purposes, decided footprint is the Package_TO_SMD:SOT-23-5 within Kicad. It has all the pads we need to match our pins and porting it to PCBNew via importing of the netlist showed no issues with the ratsnest giving us a clear solid outline for our traces to each individual pad on the components. There were other options regarding footprints for this specific component such as creating it using the footprint

editor just in case there was a sort of mistake with the attained footprint. Though, this was not the case for this footprint because using the footprint editor and cross-checking with the datasheet for the LMR62014XMF, we can see that the dimensions match up with little to no issues which means that it's safe for us to use our chosen footprint for this component. Further verifications that we performed were the electrical rules checker within Eeschema as well as careful analyzation of the surface mount pads on the footprint itself. If there were any inconsistencies that we noticed with the labeling of the nets on the pads, we'd have to find out the error that is being caused with that specific connection in the schematic. Even if there's a created netlist for the schematic that can be created even if there are no errors with your connections just so if we properly created our reference fields. What we needed to verify within our schematics were the wirings as well as the attached global labels pairing each power circuit with their designated peripheral or microcontroller. So initially our ratsnest needed to be verified with careful analyzation.

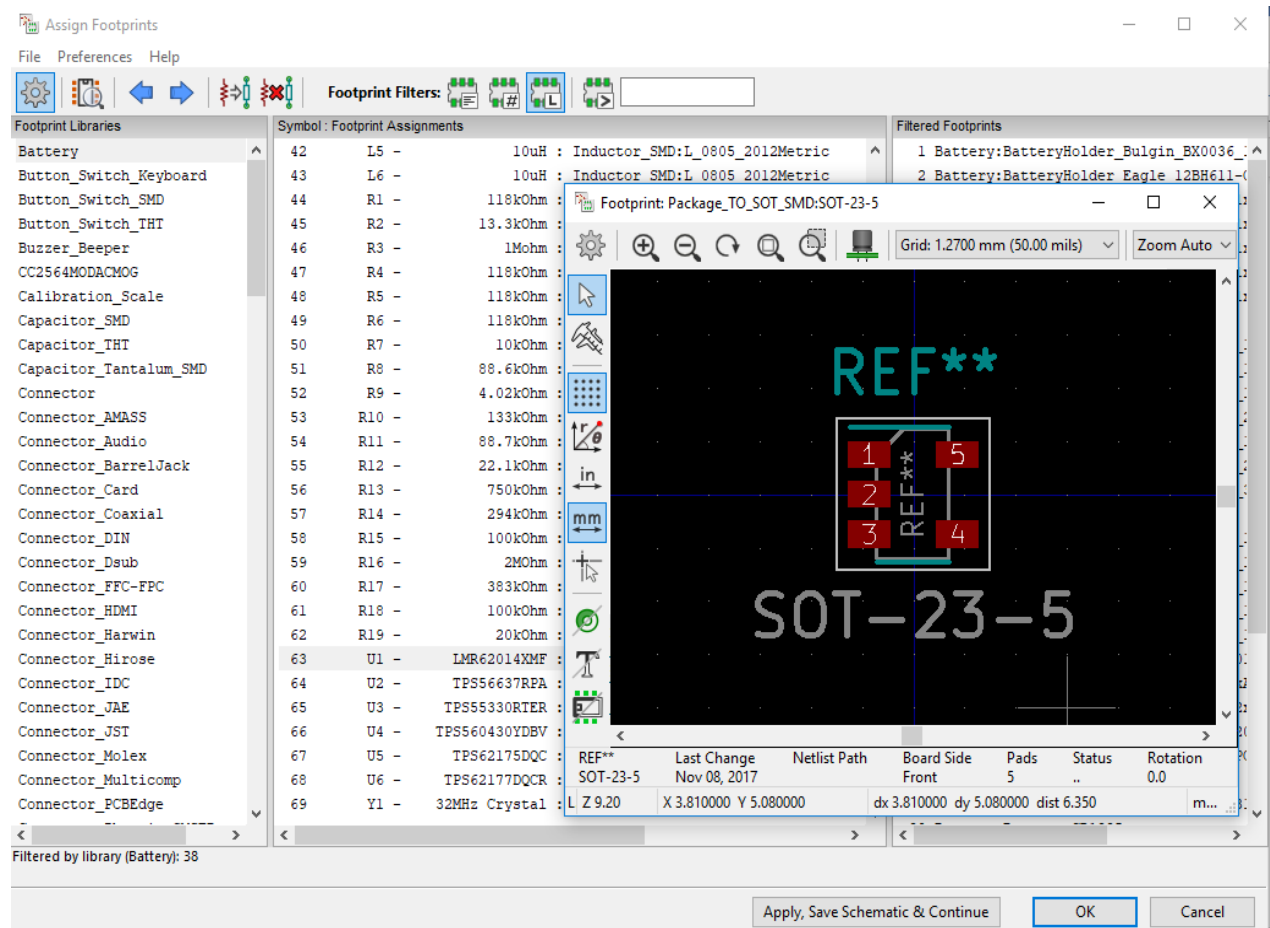


Figure 30: Assigned Footprint for LMR62014XMF Simple Switcher

9.5.2.2 Hierarchal Sheet #2 for LEDs – DM

For the LED power circuit, it was not as easy to deal with in comparison to the circuit for the fans. Even whilst utilizing all available sources to easily locate the symbol for the TPS56637RPA as well as the footprint is was not available anywhere. I had to manually create the symbol for the synchronous buck-converter in Eeschema as well as manually create the footprint using the

footprint editor tool in Kicad. After creating the footprint manually, we went about carefully verifying everything that we have done in terms of our connections to the peripheral for this power circuit, being the LEDs as well as verifying these connections with the electrical rules checker within the Eeschema software to make sure everything is up to standard with our connections within the schematic.

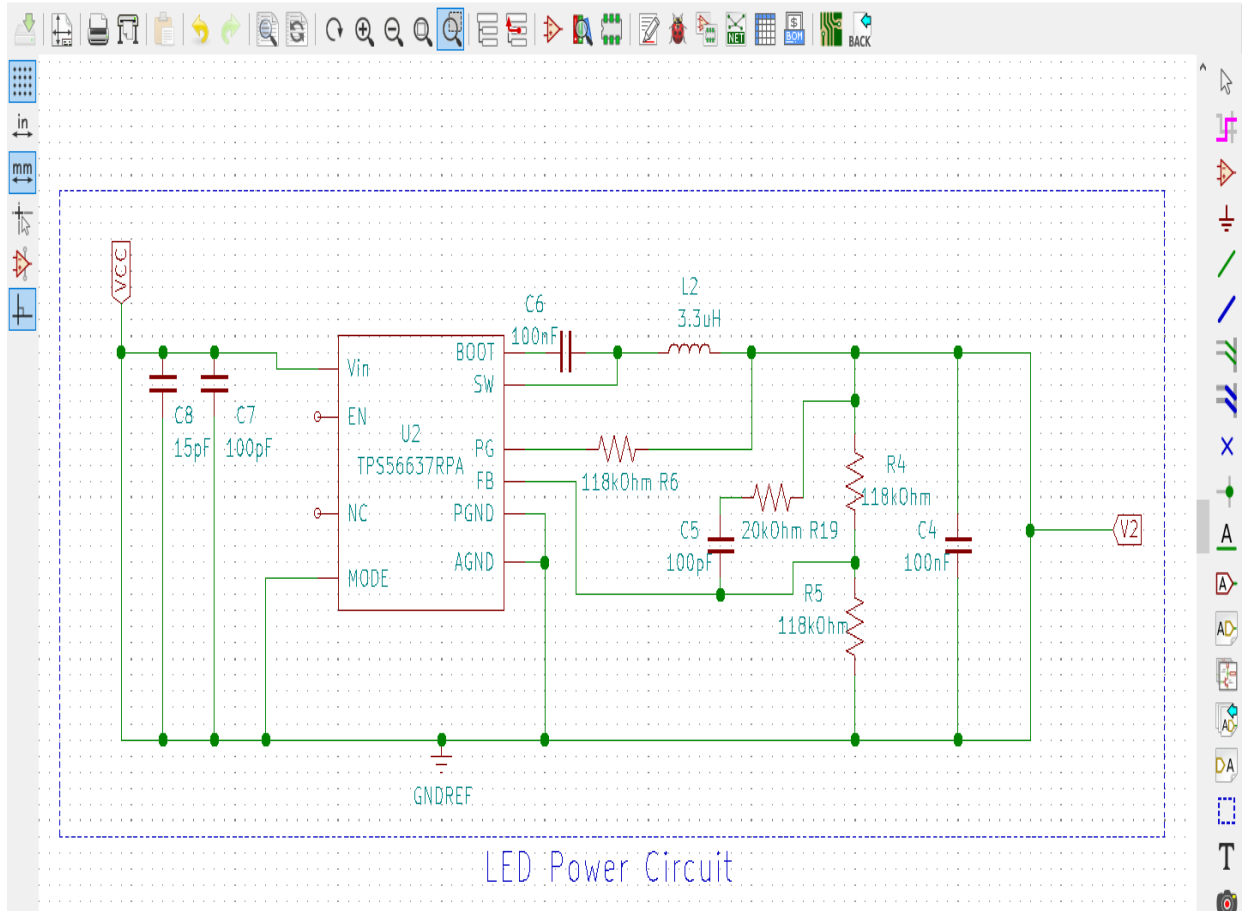


Figure 31: LED Power Circuit with Connections via Global Labels in Kicad

Using the footprint editor in Kicad, we had to look at the datasheet for the TPS56637 and scroll down to the section regarding the dimensions of the part and had to create our pads based off the package of the synchronous buck converter. This requires careful precision to do, so double checking the distances and dimensions are important so that the part itself does not have any issues with fitting on the pad as well as the PCB properly. The use of silkscreen and to create the outline is extremely important for the creation of this footprint. Since this is the first time we created a footprint, there would be initial difficulties with creating the library for it. Being able for us to use the footprint required us to create a separate library that is saved outside of Kicad's folders locally as a .pretty file is required in order to even save the footprint that we created. Then we assigned the footprint to that separate library and paired it with our symbol within the schematic and then generated the netlist for it. Our footprint creation was also a little worrisome based off the fact that we cannot be wrong whatsoever with our assigned dimensions. So, on paper we had to make sure that our mathematics was on point for the dimensions of the designated footprint during creation.

Also, the layers needed to be created to the proper specifications such as making sure we had surface mount and silkscreen.

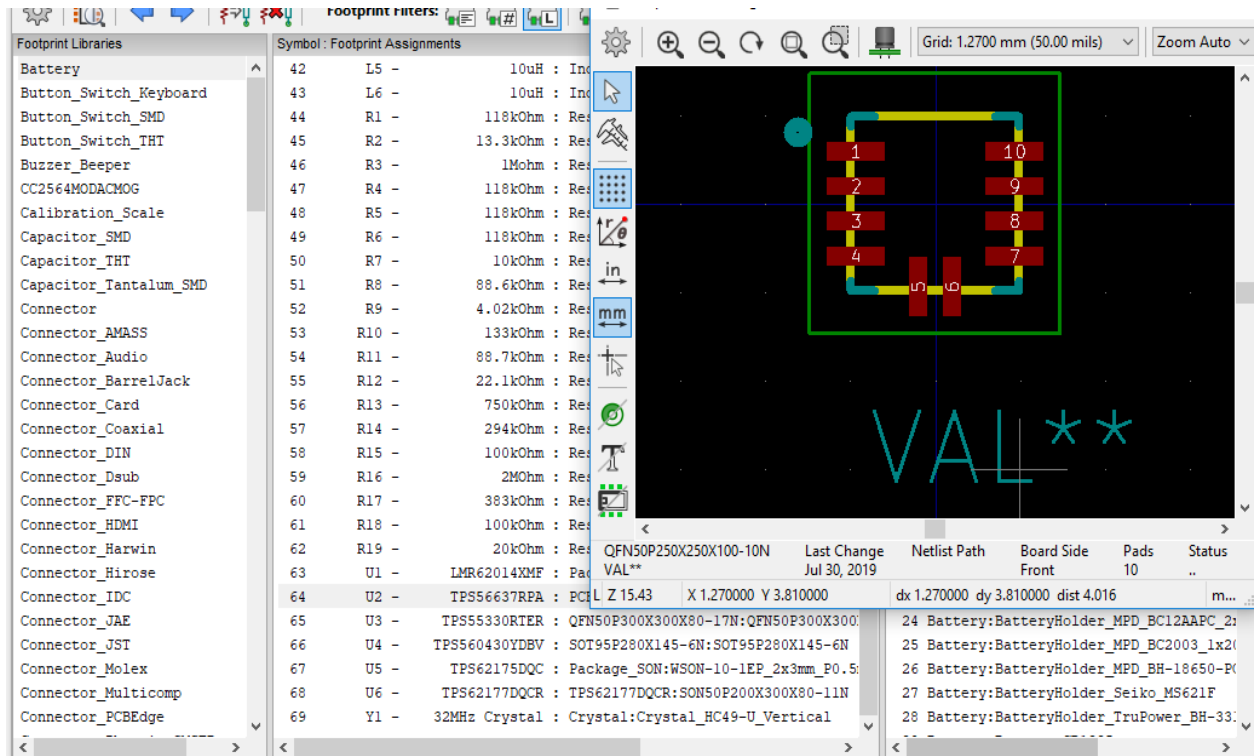


Figure 32: Assigned Footprint for the TPS56637RPA Synchronous Buck Converter

9.5.2.3 Hierarchal Sheet #3 for Water Pump - DM

For the water pump, the voltage regulator the TPS55330RTER had no readily available symbol in Kicads main libraries or anywhere readily accessible, though there was availability for a downloadable footprint for the regulator itself which is fine since the symbol is not as difficult to create in comparison to the footprint of the component. Since the footprint is already readily available to tag onto the symbol, generating the netlist and porting into the design software will go without any issues. We needed to make sure that the circuit was properly labeled with each component in order to generate a proper netlist. There can be no inconsistencies with the reference fields which we accounted for. Each individual component within the schematic for the water pump has its own unique identifier for this known as the reference field. If the reference field for the schematic is solid, then the netlist will be in order and we will not arrive with any errors. The good thing about Eeschema and the generation of the netlist is that it warns us if there are any minor components that contain the same reference field which allows us to mitigate all these problems that we were potentially facing. This is also further paired with the electrical rules checker to make sure connections were properly made as well. Though we ran into a critical situation where due to one of the global labels that we made, it was not being properly connected to its designated peripheral. This is due to human error on our part via inconsistent labeling and

was easily fixed by altering the name of the global labels that was pairing the regulator to the incorrect peripheral.

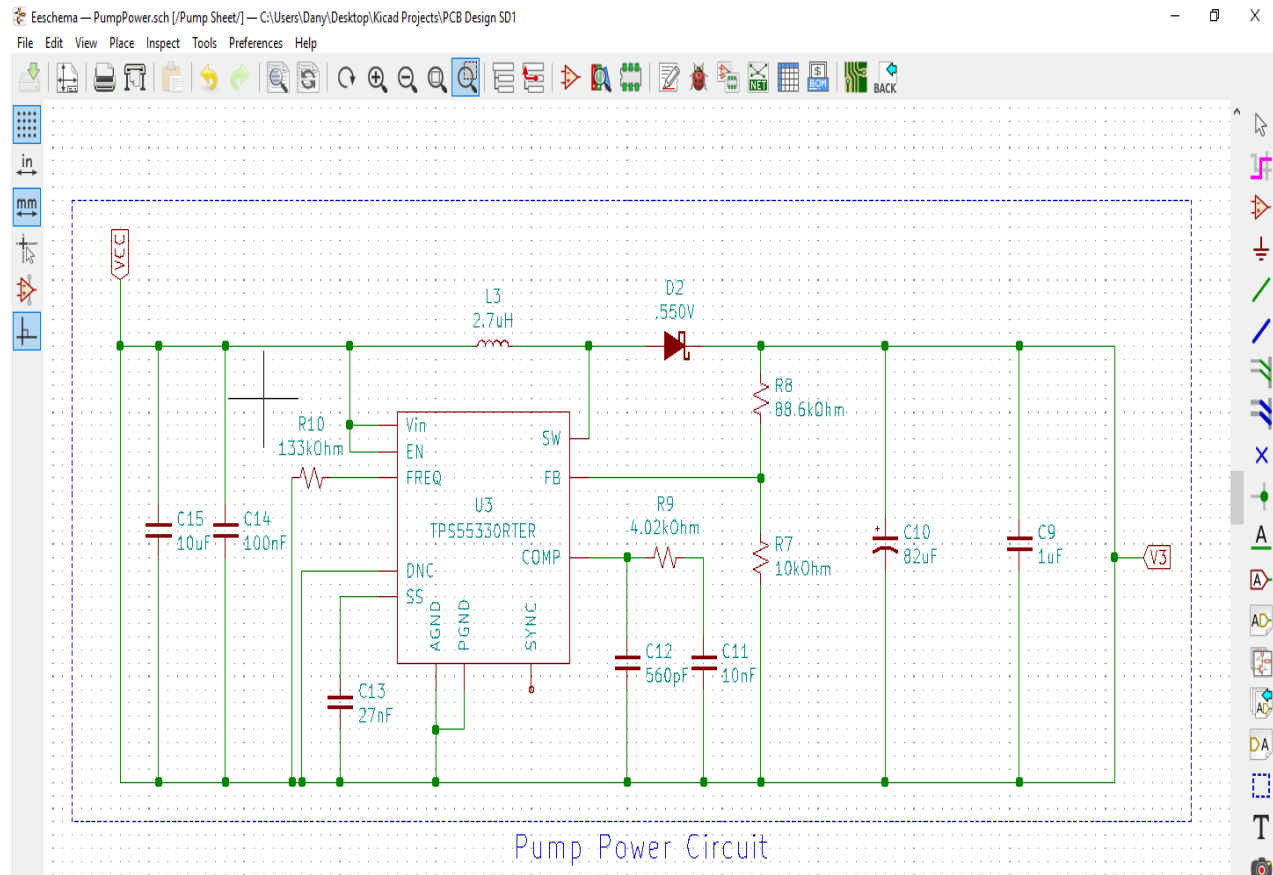


Figure 33: Water Pump Power Circuit with Connections via Global Labels in Kicad

For the TPS55330RTER DC-DC regulator the footprint was readily available within Kicad's libraries. Of course, since we're dealing with surface mount components, it narrows the list down to what we needed to choose in terms of footprints. Plus checking the regulator on Componentchecker or SnapEDA both give us the same exact footprint that we should be using for this DC-DC regulator. Due to this fact, we decided to trust in these validations of these two websites, but we also verified the dimension of the specific footprint with the information given to us about the component within the datasheet as well just to be on the safe side. Also, to further verify that the footprint was correct for the DC-DC regulator, we used the electrical rules checker within the Eeschema software in Kicad to make sure all connections were made and then we even further verified everything by checking each individual pad on the surface mount footprint that we chose with the ratsnest located within PCBNew. Verifying the pads and the connections that the ratsnest is showing us put our concerns at ease when deciding on the footprint that we're using. Though it is easy for us to be missing any errors even whilst having properly made our footprint. We needed to see where the ratsnest was connecting the DC-DC regulator to make sure that the pin that it's being connected to is also properly labeled. We do not want to be in a situation where we have our power circuits be properly created but the nets within the peripherals ended up causing us any major issues.

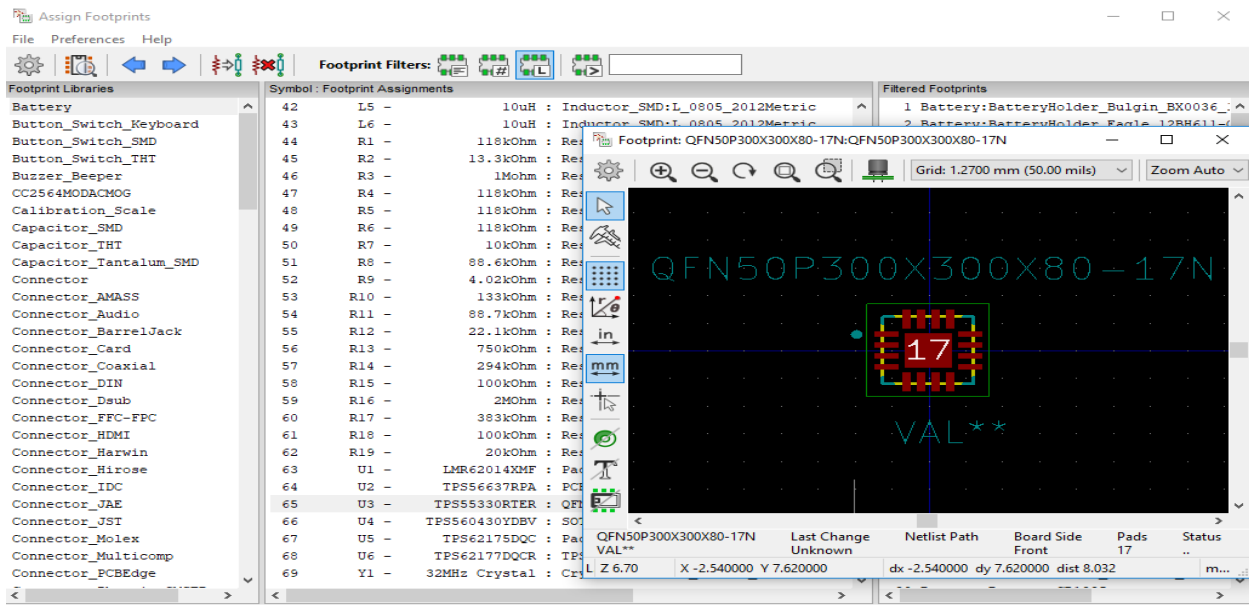


Figure 34: Assigned Footprint for the TPS55330RTER DC-DC Regulator

9.5.2.4 Hierarchal Sheet #4 for Humidifier - DM

For the power circuit for our humidifier that we're going to be using in our project, we're using the TPS560430Y step-down converter. Now, for this component the component once again like most of the other symbols for these power rail circuits needed to be created from scratch utilizing the symbol editor and creator in Eeschema. All that needed to be done is assign a reference field for the step-down converter as well as assign it the proper name. After these designations, we simply update our netlist once again and then port it to PCBNew after giving it the proper designated footprint for it.

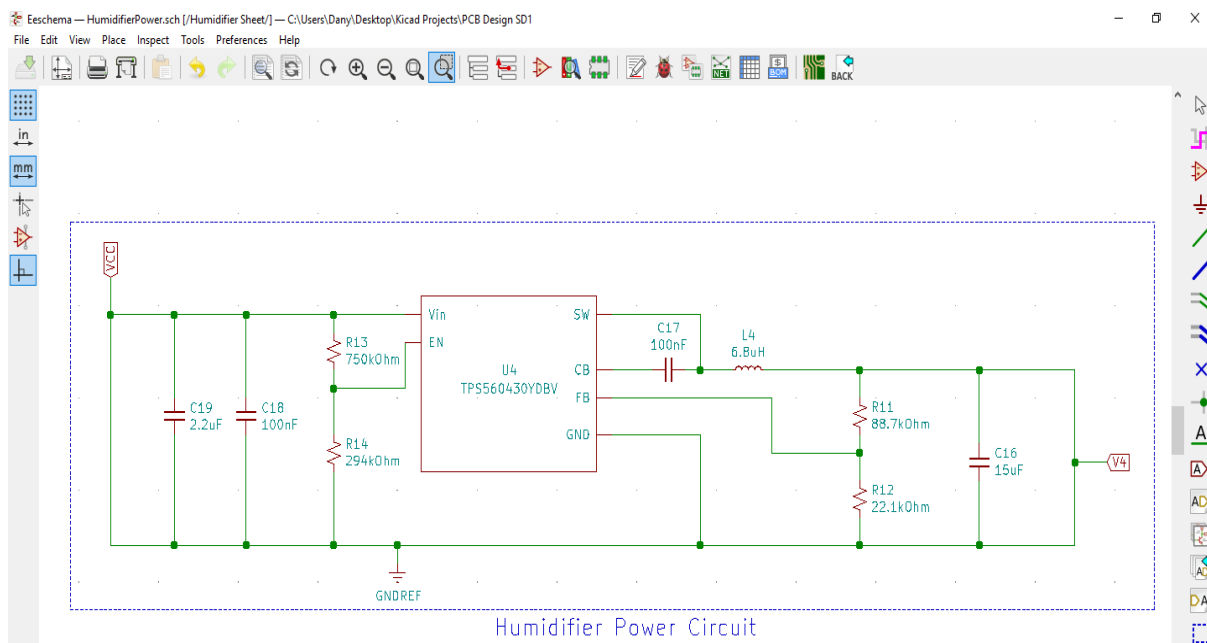


Figure 35: Humidifier Power Circuit with Connections via Global Labels in Kicad

For this footprint, it was standard to designate to this component since this one was also one that didn't need to be manually created as well. Being a surface mount component there was only one type of package for this TPS560430Y so options for the footprint are limited. There were six connections on the symbol within the schematic with their own designations to the pads and the footprint we decided on which is the SOT95P-280X145-6N contains the proper dimensions for our component which is optimal for us and saves us a lot of time.

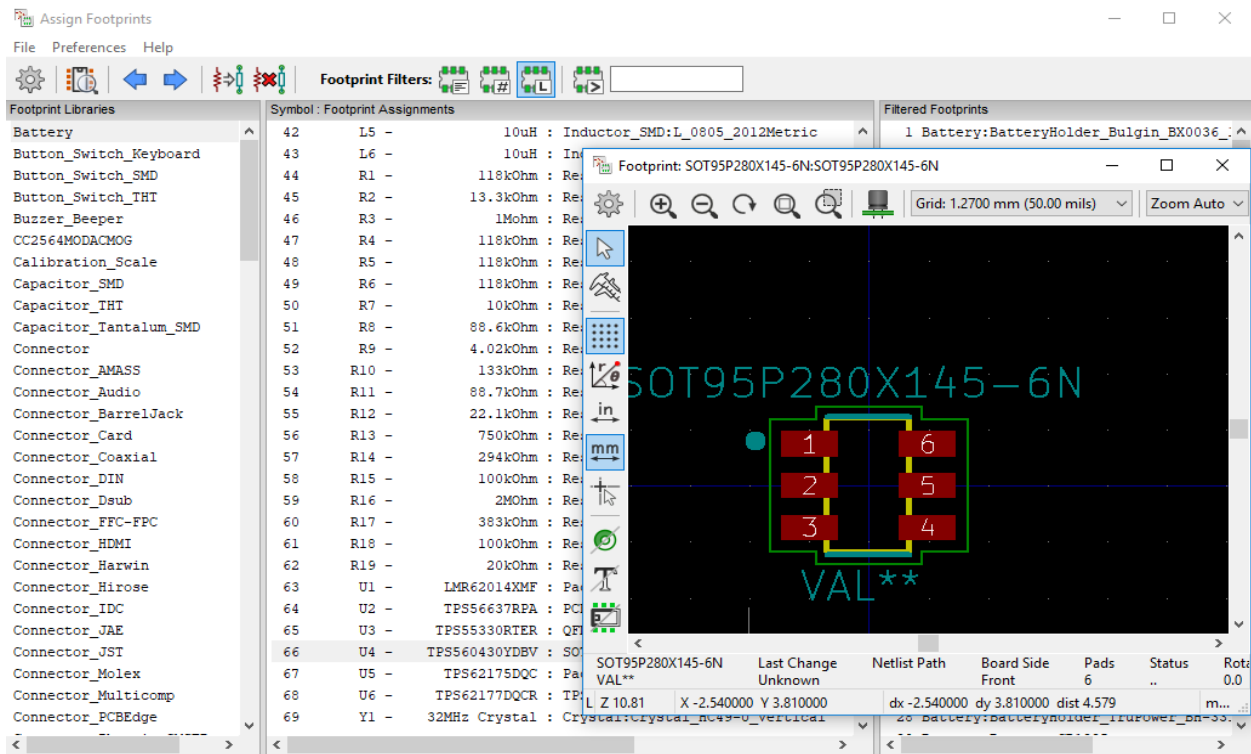


Figure 36: Assigned Footprint for the TPS560430Y Synchronous Step-Down Converter

9.5.2.5 Hierarchal Sheet #5 for the LCD Screen - DM

For the LCD screen the step-down converter that we used being the TPS62175DQC fortunately had a readily available symbol within Kicad's default libraries. Something such as this is very useful for us because the symbol being there more than likely means that the part itself is widely used and there are lots of support for the component as well. Mitigating the need to create the symbol saves time on the creation of the schematic. There is no need to place pins and manually assign the nets for the creation either as everything is already created and finished in the symbol. Having this level of availability for us allows us to save time on the creation of the symbol and instead utilize the extra time we have by double checking the pinouts in the hierarchal sheet and checking for other errors. Eeschema allowed us to be able to verify any critical errors with their electrical rules checker program within the software. This was able to pinpoint any inconsistencies with our labeling on each pin on every component in our schematic. It'll also verify this check for the circuits that we have for the power rail within all six of our hierarchal sheets making sure that the global pins that we placed for the connection of the peripherals on the main sheet are being connected. Utilizing this software, we were able to verify this check for the circuit within the LCD screen hierarchal sheet assuring that everything was connected.

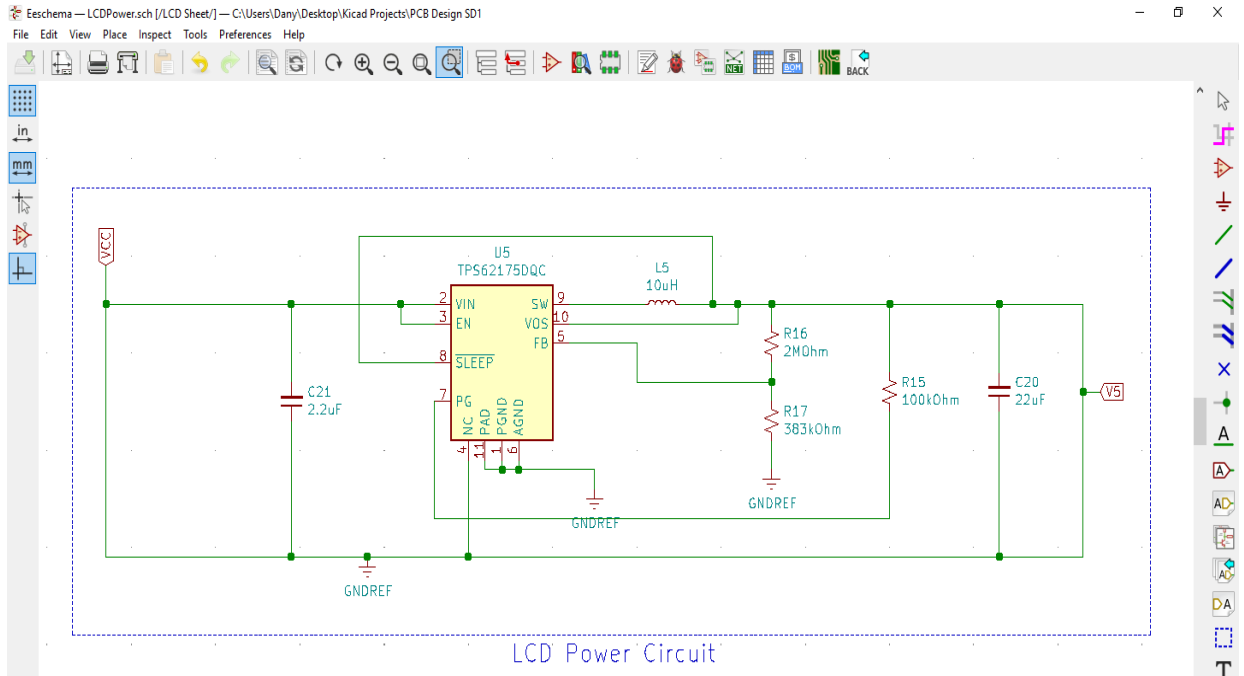


Figure 37: LCD Screen Power Circuit with Connections via Global Labels in Kicad

Regarding the footprint for our TPS62175DQC step-down converter for the LCD screen, our selection was a difficult choice because there were multiple recommendations for footprints for the specific symbol for that step-down converter. Though navigating around that wasn't difficult because all we needed to do was look at the pinouts in the datasheet for the converter as well as careful observation of the datasheet as well. Observing the datasheet for the step-down converter allowed us to verify the pins and the dimensions for it so that we would be able to choose the correct footprint.

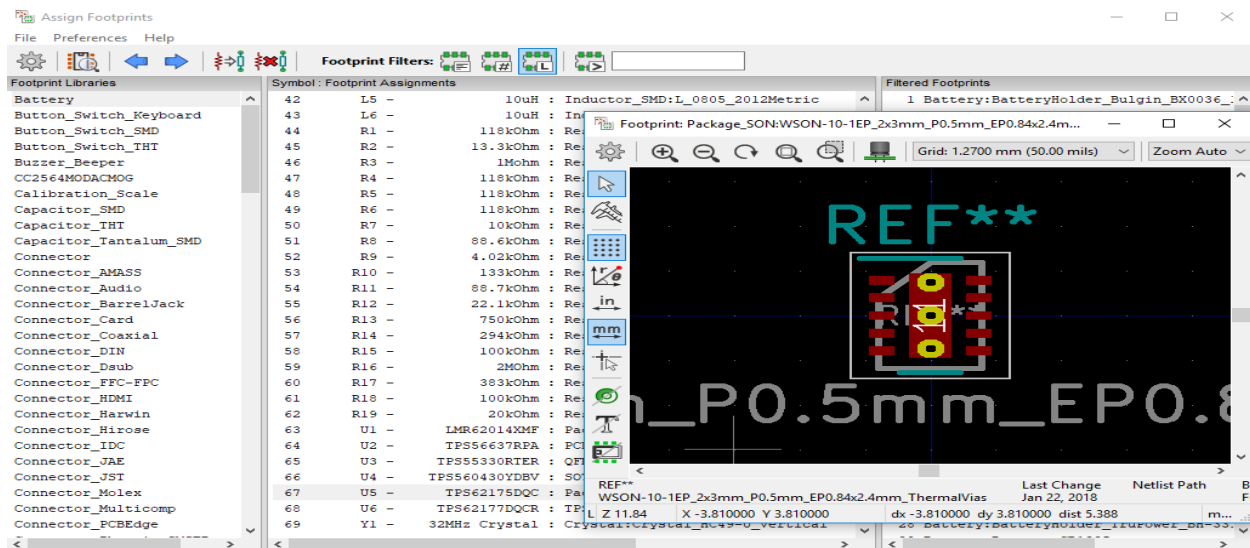


Figure 38: Assigned Footprint for the TPS62175DQC Step-Down Converter

9.5.2.6 Hierarchal Sheet #6 for the MSP432 – DM/AL

Now, for the main microcontroller on the parent PCB, there's a step-down converter being the TPS62177DQCR. This symbol in the power circuit was unfortunately not found within the main libraries within Kicad, nor was the symbol found anywhere else that is readily available. The way we went about creating this step-down converter is by taking the same approach we made for the previous hierarchal sheets in which we needed to use the symbol editor which is located within the Eeschema software. Verifying all of our needed pins for the step-down converter was something that we went about doing

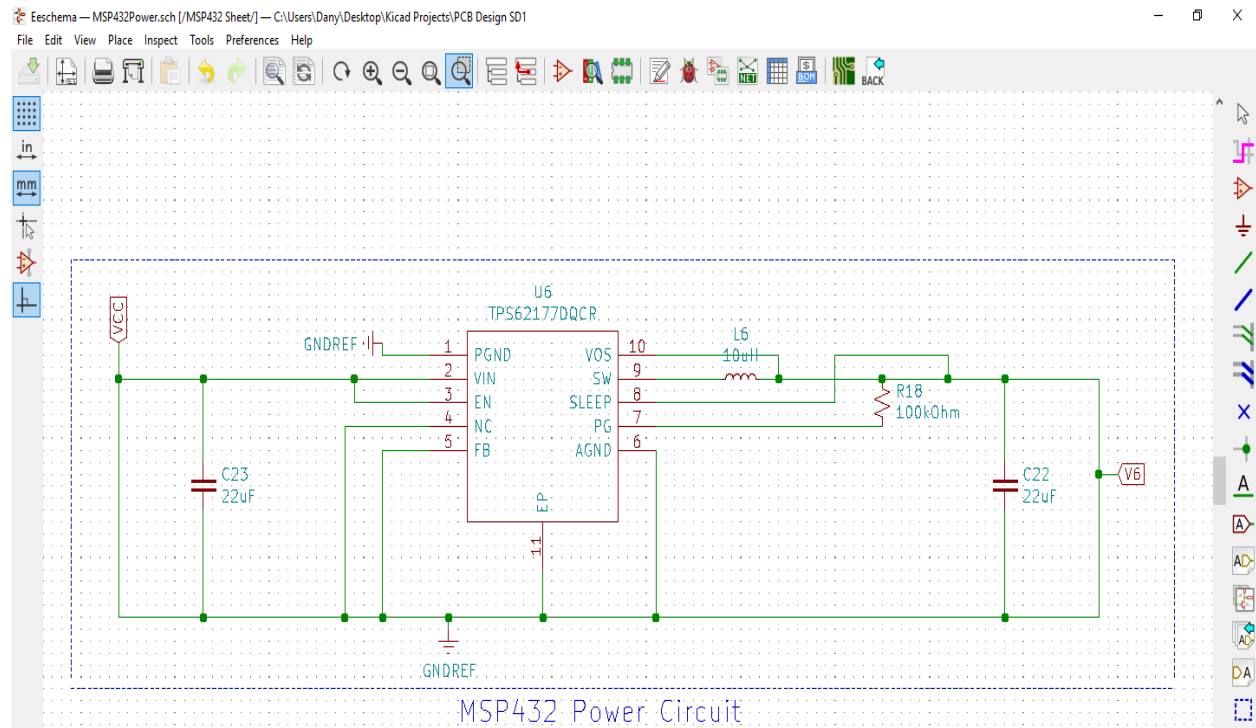


Figure 39: MSP432 Power Circuit with Connections via Global Labels in Kicad

When it comes down to the designation of the footprint for the MSP432, since the microcontroller itself is used in so many projects, hobbyists or not, the footprint is readily available nearly anywhere someone can look, though it's easy to make a simple mistake regarding the specific footprint since there are many versions of the MSP432 all with varying numbers of pins so we had to triple check that the footprint that we got our hands on for the designation to the symbol was as accurate as can be. To also once again verify that our footprint was a solid choice, we verified the netlist according to this footprint using the PCBNew viewer to analyze our nets on that component for the step-down converter. We simply looked at the ratsnest and verified that all the connections are being made to the proper components such as all of the capacitors, resistors, grounds, inductors as well as the connections to the main power supply and the main microcontroller the MSP432. Verifying all these connecting via the electrical rules check and the ratsnest in the PCBNew software was paramount to us not making any errors.

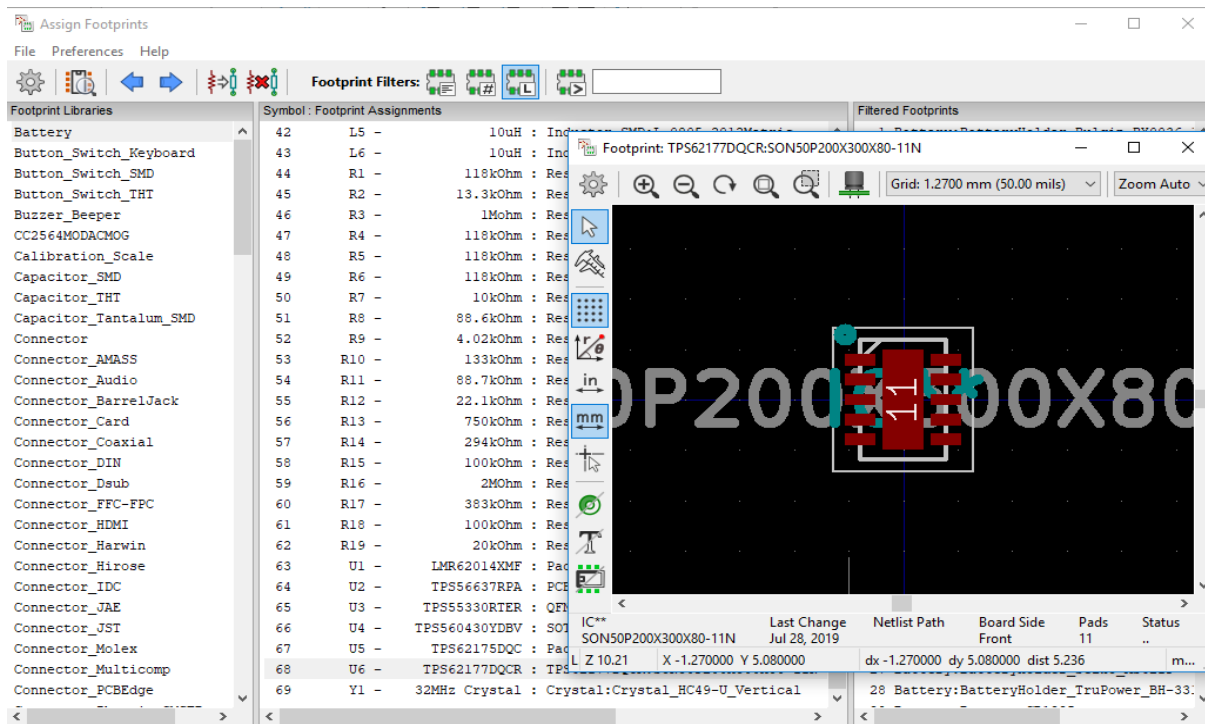


Figure 40: Assigned Footprint for the TPS62177DQCR Step-Down Converter

Next, we have the child microcontrollers as well as their schematics. These were much easier to create due to the lack of attached regulators and peripherals. They are on a much smaller scale than the schematic for the parent MSP432 microcontroller. How these two child MCU schematics were created was by opening two more different projects in Kicad after working on the schematic for the parent microcontroller. For the first schematic, what we added first was the MSP430 microcontroller and the few additions it required. We added connections for the soil moisture sensor, PH sensor, temperature and humidity sensor as well as a Bluetooth module. When it comes to providing power to this PCB, we're going to be connecting them to batteries for power. For both schematics, there was no need to utilize the hierarchal sheets feature because we could fit everything on the main schematic sheet. Organization is also made easy for these two because of another useful tool in Eeschema which allows us to draw graphic lines around our components as well as a tool to place text wherever we wish within the schematic. In our experience the best way of utilizing these tools for the schematics for the child PCBs were to individually separate our components using the tool for graphic lines and then label each one with the text creator. This made pinpointing each component within these schematics simple and straightforward as well. Though this is all just for the child microcontroller with the three attached peripherals. The same scenario does not occur with the parent schematics since there is simply just not enough room to create a neat and organized template using graphic lines. The best that we could do is to simply organize the components in such a manner that they're easily identifiable via the connections as well as utilizing the hierarchal sheet with the text editor to know which circuit is connected to what component. This makes navigation around our schematics rudimentary without too much being cluttered on the screen. Also pairing this feature along with the global labels allow us to not have wires overlapping multiple components within our schematics for these child PCBs.

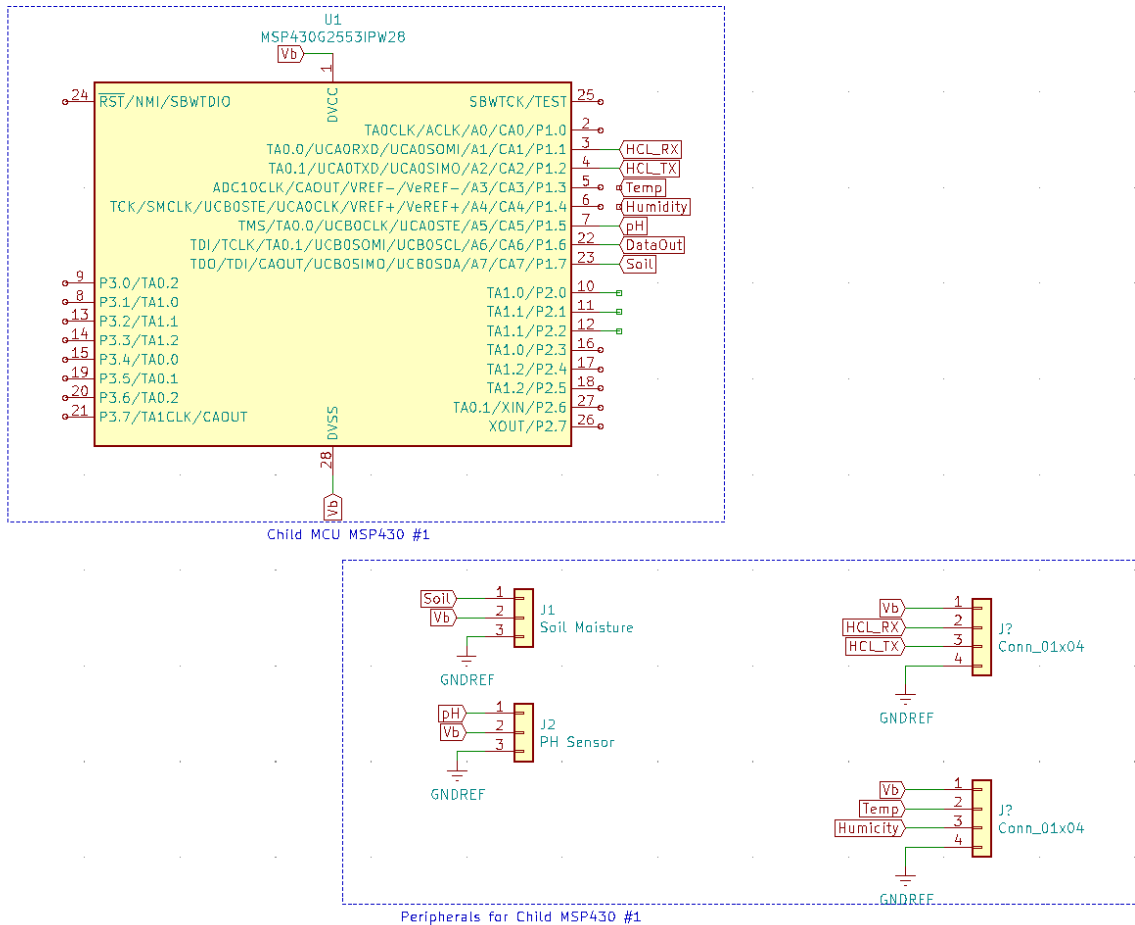


Figure 41: Schematic for MCU Child #1

The other schematic we created was for the old MSP430 PCB we're creating. This other schematic is very similar to the other child PCB is terms of connections and peripherals, but the main difference is that one of these schematics contain a temperature and humidity sensor while the other one does not. This second schematic does not contain the temperature and humidity sensor and only contains connectors for soil moisture as well as a PH sensor. Standard symbols for 01x03 connectors are mainly used for the peripherals for this child microcontroller as well as the other one. The Bluetooth module symbol that was added was not originally available in Kicad libraries but there were available footprints and libraries for our specific module available on Github for us to access. Also, there is also are no need for the usage of a regulator in these child PCBs as they are only going to be battery powered, unlike the parent which is going to be powered via an outlet. The connectors that we're going to be using are the same ones as the parent schematic as well. There was no need to change them since the connections to the MSP430 operate in the same manner as with the connections from the sensors on the parent schematic being attached to the MSP432 microcontroller. Each one is attached to a ground, a voltage source which in this case is the battery, and a GPIO pin.

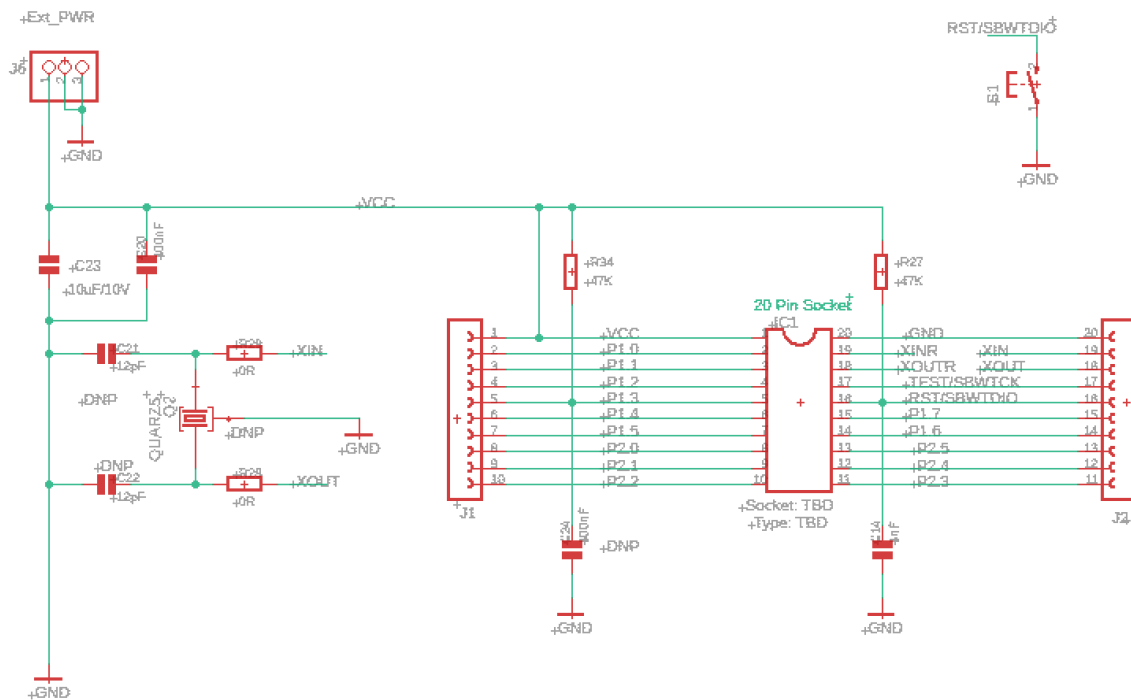


Figure 42: Schematic for MCU Child #2

9.5.3 Assigning Footprints to Schematic Symbols - DM

Next, we needed to figure out what exact footprints we needed for our PCB components. We already decided on the footprints that we're going to be using for our regulators for the peripherals, now it's just deciding on the footprints for the other components. Mainly, we wanted to have footprints for our components so that they are mainly surface mount. Surface mount components that are a decent enough size allow us to be able to hand-solder our components to the PCB without the need or use of a reflow oven. Needed to be able to use a reflow oven inconveniences us in terms of time due to the need to careful application of solder paste to the board. If we have too little solder paste or if we have too much, there will not be a solid connection from the components to the PCB, so it's very important that we decide on package sizes that are feasible for us to be able to solder to the board. When it comes to choosing packages for resistors and capacitors, we decided on 0805 package sizes which allow us to be able to solder by hand. If any difficulties arise from soldering by hand with these package sizes, we can use flux which allows us to easily solder these components to the PCB since solder will flow towards the flux if applied to the component. Some components are not too difficult to assign footprints to their symbols such as the MSP432 or the MSP430 microcontrollers because they already have their own library with the available footprint. We need to decide on packages for the other components such as our connectors for our peripheral components. For our diodes, we decided to go with a surface mnt 0603 which is a small package size, but still a good enough size for hand soldering to the PCB. We need to make sure that our footprints are also selected in a way to save space on the board to give us enough clearance to be able to solder properly without accidentally damaging other components with a soldering iron. For this same exact reason, when deciding on the package to use for our inductors we also decided to go with a surface mount with a manageable package size of 0805, which is the same size of our resistors and capacitors. Remaining consistent with package sizes also reduces

clutter and makes the PCB seem much more organized. Other components that we needed to decide footprints for were the LEDs, fans, sensors, pump and humidifier. An optimal selection for these components is female connectors that require male pins for connection. These allow for easy connection to the PCB from the peripheral components via male jumper wires. For the debugger for the MSP432, we decided to have a through holes connectors since it is simply for testing and debugging, not there in order to occupy those areas on the PCB permanently. When it comes to the footprint for the crystal symbol for the schematic, Kicad's libraries contains a plentiful amount of them, so we decided to choose a symbol that was a decent size for the PCB and won't take up a lot of space. Of course, the need to avoid using a reflow oven for our team is paramount because we do not want to be running the risk of making an extremely critical mistake with setting up the reflow and ultimately end up bricking the microcontroller and ruining connections on the PCB making the circuit board that we ordered ultimately a waste of time. On the opposite end of the spectrum, soldering irons are not too expensive and gives us more control on how we're going to be applying our components to the circuit board. We can also greatly mitigate the risk of burning any of our components by using a soldering iron with a specific temperature adjuster. These are just temporary solutions to using extremely small package sizes for our surface mount components on our parent and child circuit board designs. While also deciding which packages, we're going to choose for our minor components we also made sure to remain as consistent as possible with them as well in order to avoid any conflicts.

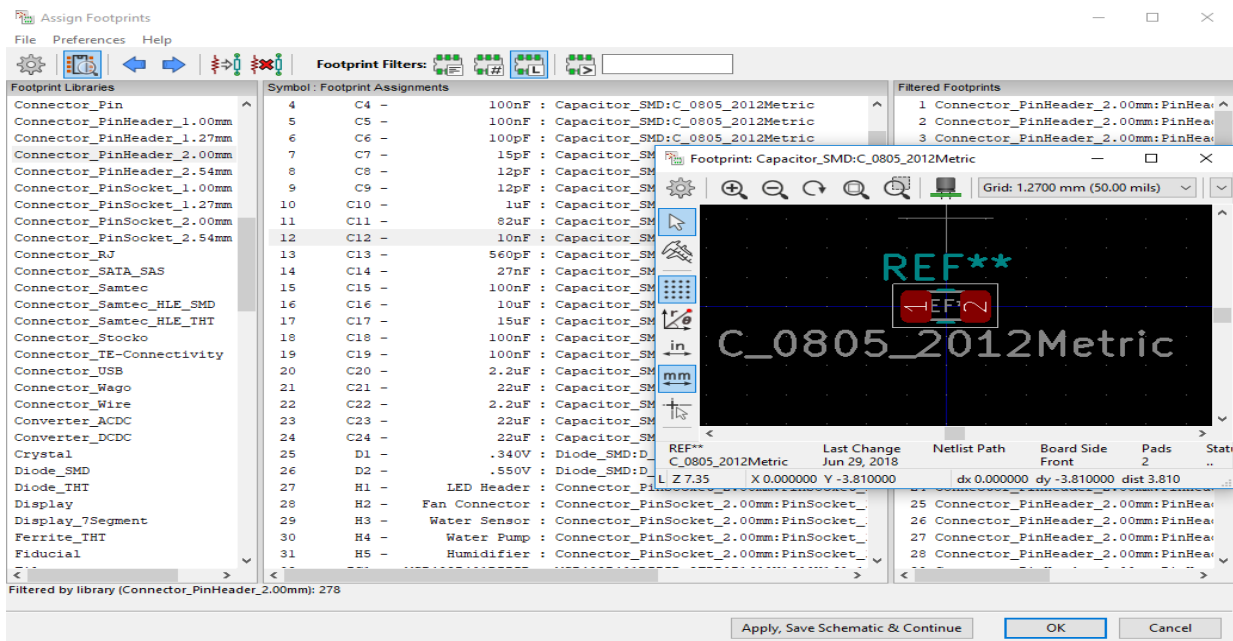


Figure 43: Footprint assignment to symbols in parent PCB

9.5.4 PCB Layouts - DM

Now after we assigned footprints to all our symbols in the schematic, we needed to save our schematic utilize a tool and generate a netlist within Eeschema which will allow us to transfer our symbols into the PCB software within Kicad known as PCBNew. Making sure the netlist is perfect is paramount because if it isn't PCBNew will give us a ratsnest with pads not giving a sign to a connection. Having a proper netlist will allow PCBNew to properly organize and label each one

of the pads on all the components along with their corresponding connection. The ratsnest that is generated allows us to figure out where the traces will go. If a proper netlist was not created, there will be conflicts within PCBNew with warnings and errors prompting the user to go back into the schematic and fix their connections. It's better to catch the issues of a flawed netlist early to avoid losing many hours of hard work. Of course, a little confusion arises at first since generating the netlist and then opening the PCB software does nothing. We also need to load the netlist within the design software as well even though we just created it within Eeschema. After doing this, we arrive to a cluttered mess with our components within the design software. Kicad requires us to utilize keyboard hotkeys in order to be able to do anything within PCBNew such as unlocking components and allowing us to organize and move them around. After learning all the hotkeys for PCBNew, we begin by organizing all our components so that they're easily identifiable on the board. Next, we need to utilize the layers section on the right-hand side of the PCB software and navigate to our edge cuts. These layers are used to create the borders for the board itself which is great in order to save space or to add more space. We want our PCB to not have any areas in which there is nothing being used. Of course, we're going to leave more than enough room for our traces to be able to be a little separate from each other. Plus, we want to be able to have our PCB to be able to be mounted, so we added mounting holes to each corner of the board not too small and not too large. The size of the holes should be large enough to be able to get a standard screw size and manually attach and mount our PCB. Though if necessary, we could create our own footprint for these mounting screws as well.

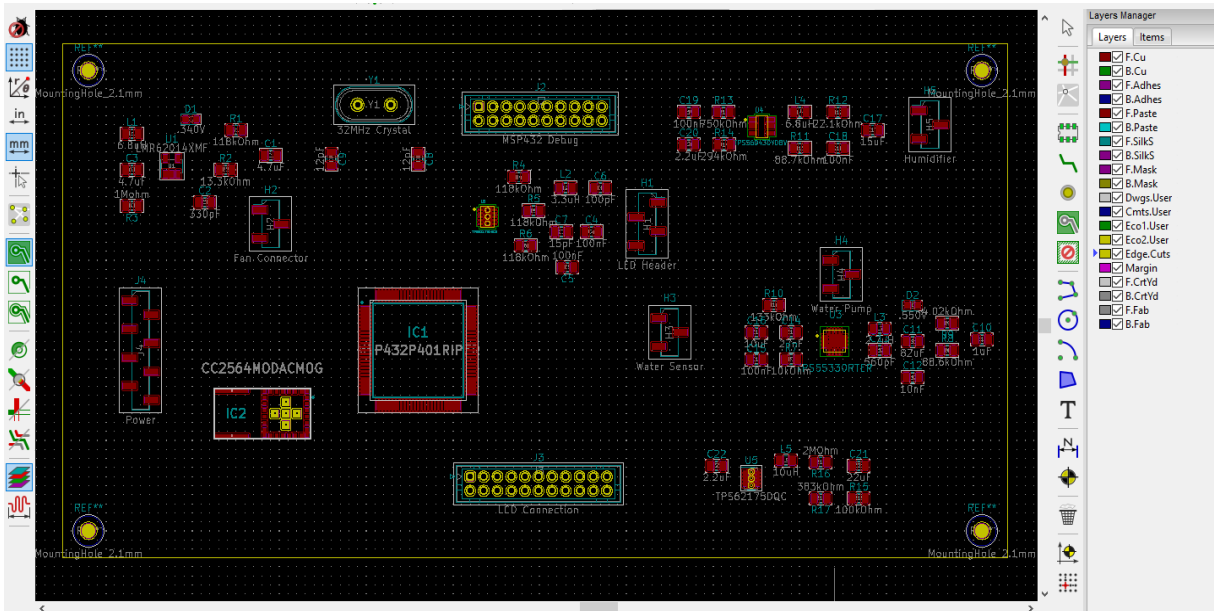


Figure 44: PCBNew Main PCB outline

For our other PCBs that have the MSP430, the boards for those will be much smaller simply because there are way less components populating the board. Populating the board containing the MSP430 are a few sensors, Bluetooth and of course the main microcontroller. The board will be a little larger due to us needing to add mounting holes to the corner of the boards themselves. Depending on whether we're going to have multiple layers for these minimal number of components depends on how big the board itself will end up being. Even though these PCBs are

small it's also easy to make simple errors with anything regarding the schematic, footprints and symbols.

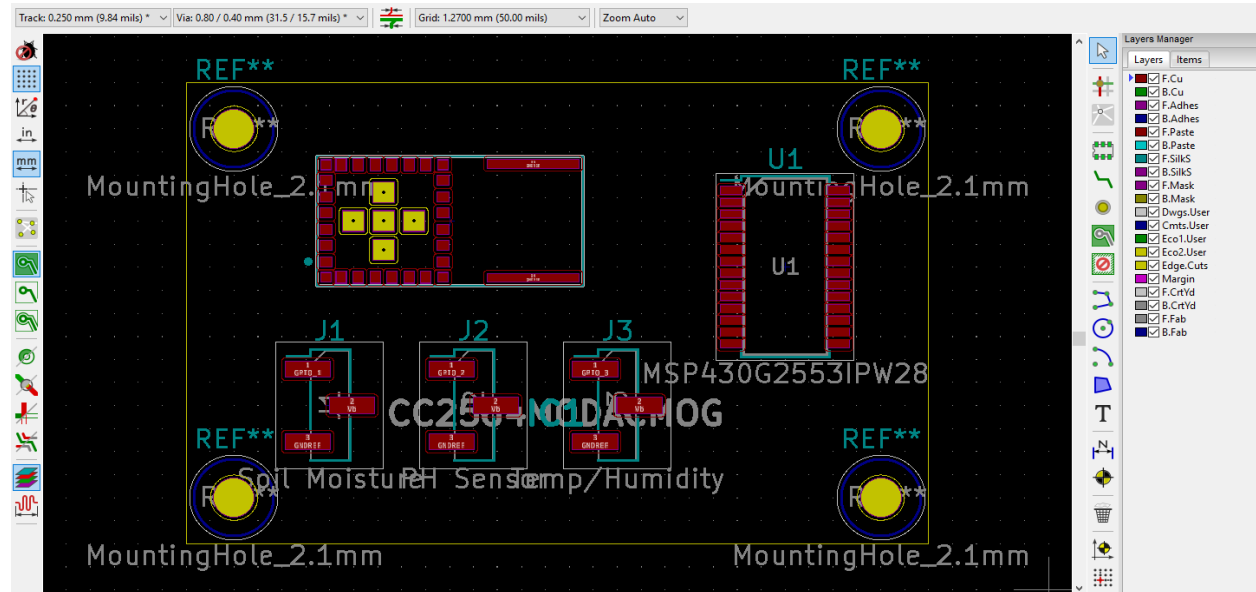


Figure 45: PCBNew Child #1 PCB Outline

As we can see in this image of the first child PCB, the board itself is much smaller than the parent, containing only a few components whereas in comparison to the parent the board is nearly twice the size of this one. Also, from the 2D view in PCB new we can see that the 2.1mm mounting holes for the board show very little clearance between the edge cuts as well as one of the sensors and the MSP430, but we can see how the board itself looks by utilizing the built-in 3D viewer in Kicad. From 3D view we can see that the mounting holes for the board gives good clearance.

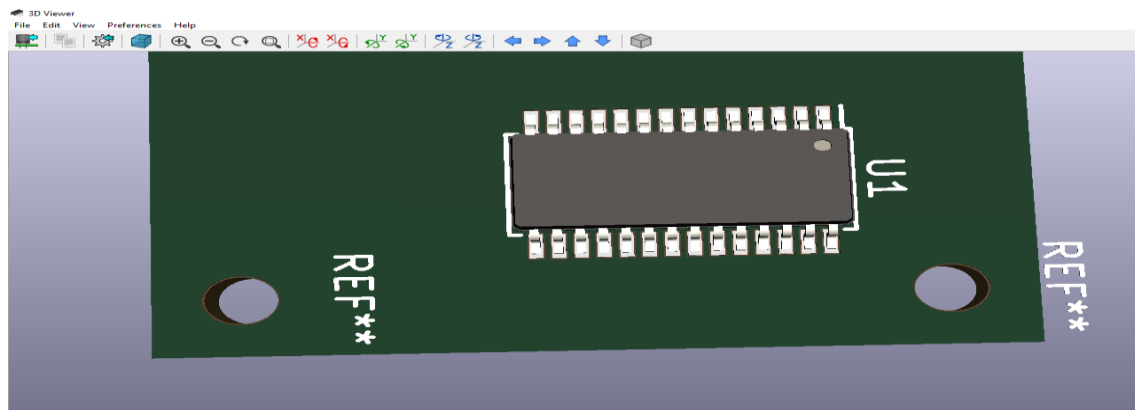


Figure 46: 3D Viewer Mounting Hole clearance on MSP430 on child PCB #1

For the other MSP430 PCB, the only difference is that it does not have a sensor for temperature and humidity, other than that the PCB should look extremely similar to the other child PCB with all three sensors. Of course, the size of the PCB may increase depending on how we're going to approach the traces between components. Below we can see how the PCB looks for the second

child PCB. Though with this PCB, due to the lack of one extra sensor, we may be able to create the board such that it is much smaller before we ship it out for board development.

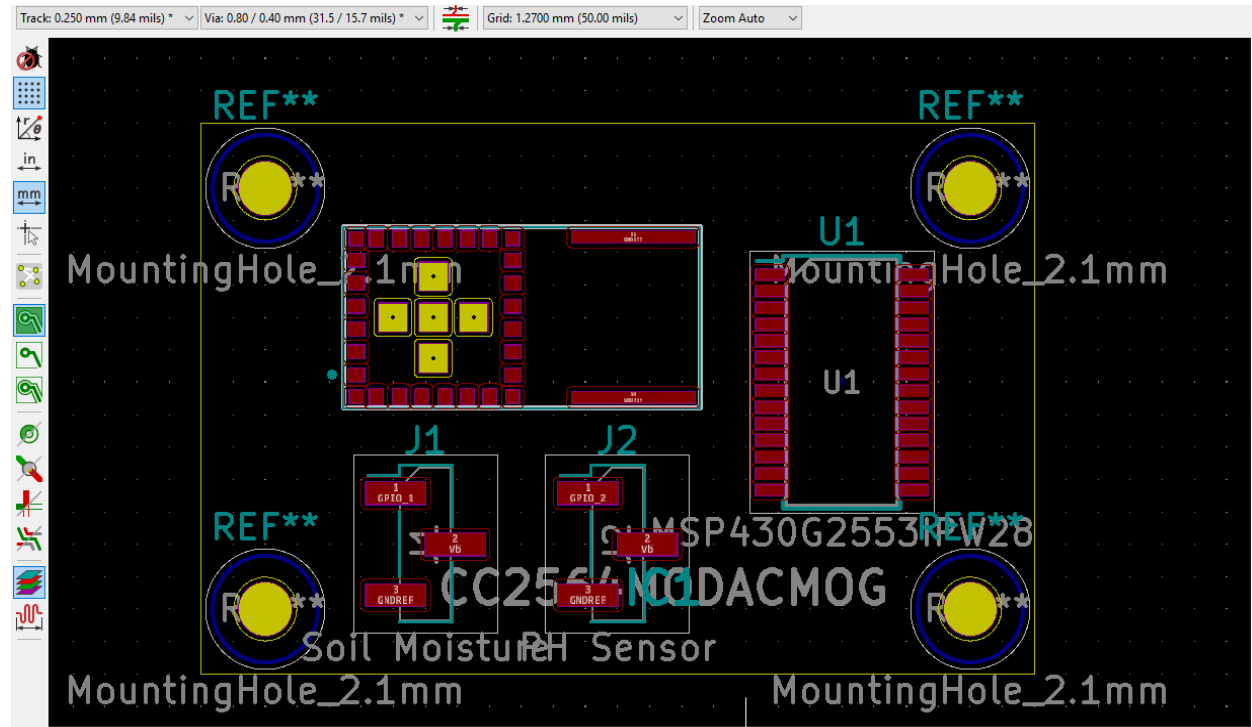


Figure 47: PCBNew Child #2 PCB Outline

9.5.5 Traces and Planes - DM

For our parent PCB, it's necessary that we have multiple layers for us to create all the necessary traces on the board. With the number of components that we have on the main PCB, a single layer is impossible to do since overlapping traces is inevitable. We need to create multiple layers in order to have a plane for our ground as well. Having a ground plane is essential for our design, since most of our components are also being tied to ground. Also, having many components on the board even while spacing everything out, we'll be held up due to the numerous amounts of traces on the top layer of the board. Even if we were to reduce the widths of our traces, there will be inevitable overlapping which we need to avoid at all costs. Of course, this is mainly an issue for the main PCB due to the numerous components, for the child PCBs with only a couple components, we can simply reduce the widths of our traces to make all our connections on a single layer. Also, another issue we needed to figure out and we did was how to associate all our grounds on the PCB. PCBNew has a very useful feature that allows us to add zones on our design and associate those zones with a specific net that we defined on our schematics. For our project purposes and to make things easy on us when it comes to grounding certain components, we're going to use Kicad's feature of creating filled zones and fill those zones with straight copper, which will allow us to ground our components. Since the PCB will be multiple layers, we'll have the plane for our ground be on a layer that isn't near the top of the PCB. Instead we'll have the layer more than likely rest underneath the top layer which is hidden and lead our traces for ground to the copper ground. Of course, since a large percentage of our components have a tie to ground, we'll more than likely surround a good majority of the PCB design with the ground plane for the ease of grounding those components.

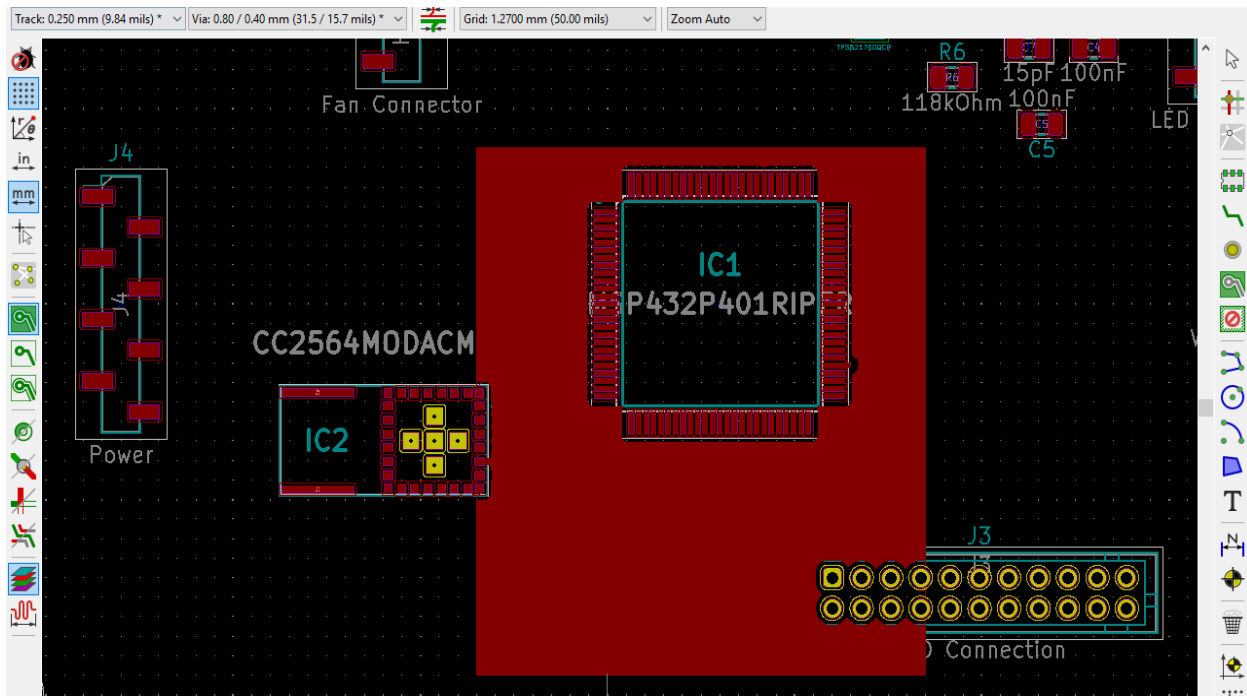


Figure 48: Example of a Ground Plane in PCBNew

10.0 Component Integration/ Testing – MM/AL

This section serves to test the condition and functionality of major components to be utilized in this project. Testing will confirm a device’s utilization for the overall design and identify major issues prior to installation. Through continuity testing and functionality testing one is usually able to troubleshoot most problems on the board level.

10.1 Hardware Testing – MM/AL

This section serves to discuss the experiments that will be conducted to validate the safety and functionality of power supplies and regulators used in this project. This testing is of great importance as it should ensure no faulty power modules be installed, which would potentially break and damage equipment downstream. Equipment broken during runtime will contribute additional cost and time for shipping and repairs and should be avoided whenever possible.

10.1.1 Continuity Testing -AL

On the actual design of our indoor greenhouse system, we will have to do a series of continuity testing to ensure that all of the connections that we made on our PCBs for both the main microcontrollers and the peripheral microcontrollers are all soldered correctly and all of the parts are correctly matching the schematics that we created from our PCB creator. To do this we will have individuals looking at all the schematics that we created as a reference and have them probe out each of the components with a handheld multimeter. The task itself might be considered tedious but is in fact one of the most important hardware testing that we will have to undergo in this project. This is because if we find that if any of the connections in the hard is compromised for whatever

reason it goes to say that it might cause a trickle effect throughout the whole system, causing incorrect signals to be sent between devices or power lines being cut off from where current needs to be delivered.

If we were to find discontinuities within a section of our system whether it be within the PCBs that we created and ordered from our vendors or from the peripheral devices such as the temperature sensor, pH sensor, water pump, etc., we will undergo the steps to mend the connections by either soldering the broken connection or by attaching a jumper wire from the two connections in question.

10.1.2 Power Supply Functionality Test - MM

The Ideally 120W power supply will undergo three tests in order to verify proper functionality for use in the indoor greenhouse system. The primary test will serve to measure the power supply unit (PSU)'s voltage regulation capacity. To do this, the PSU will be plugged into a standard receptacle and the output voltage (with no load connected) will be measured using a digital multimeter. The expected output is a steady 12V. Small variances to this amount can be negated as long as the measurement maintains consistency. Large variances, however, are indicative of power supply failure. In this case, a new unit will have to be ordered and the model reconsidered for viability within this project.

If the test above is passed, the second test will begin. This will be conducted in the same manner as the previous test, but at full-load conditions. The Ideally PSU has a maximum load capacity of 10A. An electronic load will be used to simulate the full-load condition of the supply.

The third test serves to check the ripple voltage characteristics of the transformed output. Unfortunately, Ideally does not provide an allowable range for ripple voltage; comparable PSU's can serve as a basis for evaluation. In any case, it should be as small as possible to indicate optimum performance. While at a full-load of 10A (as simulated by the electronic load), a digital oscilloscope will be connected in parallel to the load to provide an indicator of ripple voltage.

10.1.3 Voltage Regulator Functionality Test - MM

The voltage regulators will be used to produce the power rails needed for the various devices and control mechanisms. To test their functionality for use in our project, a handful of experiments will be conducted to ensure the proper power requirements are being provided to each rail prior to connection to devices downstream. While it is difficult to purchase the specified regulators in reasonable quantities, Texas Instruments allows users to request samples of the individual devices through the Webench Power Architect tool. This will give the team the chance to test each regulator prior to finalizing and ordering the printed circuit board.

Once the devices have arrived (along with their associated circuitry), tests similar to those conducted for the power supply will be carried out for the regulators. An electronic load will be used to simulate the expected load of the control equipment, and the output voltage and current characteristics will be measured and recorded using a digital multimeter. Oscilloscope measurements should be taken to determine if the ripple voltage is in line with the expected values provided in the data sheets. Two iterations of these tests should be conducted, one with the input voltage source being a lab-standard DC power supply, and the second wave of tests being with the

Idealy power supply. This will be helpful in identifying any compatibility issues or unwanted interference amongst the regulators and the Idealy power supply.

10.1.4 Microcontroller Functionality Testing - AL

To test if the ARM processor and on the main controller PCB and the MSP430s are working on the peripheral controllers. We will run a simple sample code on each of the devices to make sure that each of the pins that are used are intact are able to receive and transmit signals. This program will be similar to a “Hello World” project where we will have a single test LED on all of the boards we create and have it on a two second flashing interval so that we will know that it is operational.

10.1.5 Temperature/Humidity Sensor Testing - AL

For the four sensors used in this project, testing needs to be conducted to validate their functionality and approve their inclusion in the *SmartLeaf* system. This testing will also provide an opportunity to interface directly with the microcontrollers and identify any unforeseen compatibility issues prior to installation within the unit.

To verify that the temperature/ humidity sensor is functional, we will have two separate environments, one that is hotter and more humid (such as outside during the Florida summer) and another environment that is colder and less humid (indoors with air conditioning). An external humidity and temperature monitoring device will be needed to serve as a control for this experiment. We will then power up the temperature/ humidity sensor via microcontroller or power supply and then place the sensor in one of the environments and test if the readings are near correct within the boundaries that we expect. Next, we will place the sensor in the other environment almost right away as to have the sensor have the most contrast of reading in a period of time and check again if the readings it is emitting is within the boundaries of what we expect.

If the readings between both environments what we expect, all is well. If not though, we will go through a series of testing to see why the sensor failed to observe the readings we expected. We will first check if there are any continuity connection issues within the sensor as to see if one or more of the components weren't getting the power it needed to or created short/ open circuits. Next, we would check the code within the microcontroller to see if the controller was not reading the bits that have the information correctly, if the microcontroller was connected to the sensor the wrong way, if the sensor was sending information to the wrong microcontroller pin, etc.

Once figuring out its respective problem, we can then fix the solution in a lab setting or by hand. If not, then we would then have to consider either replacing the whole sensor completely or having another team member look at the sensor and check if they can see the issue from another angle.

10.1.6 Soil Moisture Sensor Testing - AL

To verify if the soil moisture sensor is functional, it will be similar than with the temperature/ humidity sensor. An analog soil moisture probe will be used as a control and provide context for the data being collected by the sensor. We will have two separate environments except now one of them will be in a dried soil environment and the other in a totally saturated in water environment. We will then power up the soil moisture sensor via microcontroller or power supply and then place the sensor in one of the environments and test if the reading are near correct within the boundaries

that we expect. Next, we will place the sensor in the other environment almost right away as to have the sensor have the most contrast of reading in a period of time and check again if the readings it is emitting is within the boundaries of what we expect.

If the readings between both environments what we expect, all is well. If not though, we will go through a series of testing to see why the sensor failed to observe the readings we expected. We will first check if there are any continuity connection issues within the sensor as to see if one or more of the components weren't getting the power it needed to or created short/ open circuits. Next, we would check the code within the microcontroller to see if the controller was not reading the bits that have the information correctly, if the microcontroller was connected to the sensor the wrong way, if the sensor was sending information to the wrong microcontroller pin, etc.

Once figuring out its respective problem, we can then fix the solution in a lab setting or by hand. If not, then we would then have to consider either replacing the whole sensor completely or having another team member look at the sensor and check if they can see the issue from another angle.

10.1.7 pH Sensor Testing - MM

To test the functionality of the pH sensor selected for this project, a few experiments will be conducted using a few soil samples of different conditions. Standard pH strips are cheap and can serve as a control measurement for the data received from the sensor. The strips can also provide context for said data, as there is a possibility that the data received will not be in recognizable units for pH and could need a multiplier to be properly displayed. If the sensor readings are comparable to those of the pH strips, the pH sensor will be approved for use in the *SmartLeaf* system.

10.1.8 Fan Functionality Testing - AL

To confirm that the fans that we purchased are working correctly, we will pass the operating voltage and current to the fan via an outside power source and see if the fan will operate through its power connection. If the fan does not work when plugged into the power source, we can assume that there is there is a problem with the circuitry of the fan, thus we would have to either buy another on search for an alternative component that would equvalate to removing heat from the greenhouse system.

When we receive an operating fan, we would then move on to seeing if it is compatible with the microcontroller that we have. Once verifying that the code we created to control the fan is correct, we would connect the fan to the microcontroller, make sure we have all the data signals matched up to the correct pins from the code itself and let the code run. If the fan is not operational at this step, then we can assume that there is still something wrong with the code or the fact that the command signals to the fan are being compromised. If this is to occur, the way to go about fixing this would be for us to use an oscilloscope to sense if there are any compromises of data error being sent from the microcontroller to the fan.

10.1.9 Pump Functionality Testing - AL

To verify that the water we purchased are working correctly, it will undergo nearly the same testing that the fan went though. We will pass the operating voltage and current to the water pump via an outside power source and see if the water pump will operate through its usual power connection.

If the water pump does not work when plugged into the power source, we can assume that there is a problem with the circuitry of the water pump, thus we would have to either buy another or search for an alternative component.

When we receive an operating water pump, we would then move on to seeing if it is compatible with the microcontroller that we have. Once verifying that the code we created to control the water pump is correct, we would connect the water pump to the microcontroller, make sure we have all the data signals matched up to the correct pins from the code itself and let the code run. If the water pump is not operational at this step, then we can assume that there is still something wrong with the code or the fact that the command signals to the water pump are being compromised. If this is to occur, the way to go about fixing this would be for us to use an oscilloscope to sense if there are any compromises of data error being sent from the microcontroller to the water pump.

10.1.10 LEDs Testing -AL

To verify if the LEDs throughout the greenhouse system are operating correctly, we will go through a series of steps to ensure that the connections between each LED are stable and acceptable. Luckily with the Chinly LED strip, all the connections between each LED are secure and waterproof. The only way they could be compromised is if we were to fold the actual strip to a point where no current could flow through the strip or if we were to make an incorrect cut between the strip when creating smaller portions from the five-meter strip.

Once we have confirmed that there are no continuity compromises with the given LED strip, we can start the process of confirming if the code to control the LEDs are correct. Luckily there are many LED controller libraries on the internet that are more than capable to be used as references to create our own unique led patterns. Although there would be many led strips to take into account for at different sizes, it is not impossible to control them using a single microcontroller.

To test to see if each individual LED is operational we will have drive out a code that will run through and turn on each LED in the strip one color at a rate of 5 LEDs per second. Once the sequence gets to the end of the strip it will restart in the order of red, green, blue, and finally white; then at a sequence of 10% brightness, 50% brightness, and 75% brightness to test out if the individual LED is able to handle a sudden change of brightness.

10.1.11 LCD Screen - AL

To verify the power up sequence of the LCD screen, we will undergo a series of tests by sending known images to the LCD screen to test out if all of the data pins are reading correctly and that none of the chips on the LCD board are experiencing any difficulties. If the expected image does not show up on the screen when it is being sent, the following procedures should go as follows. Acquire an oscilloscope to test out if the bits of a estimated broken pin is getting and see if the bitstream going toward the pin is what is expected. If that is not the case, run a continuity testing to see if there is an open or short circuit anywhere on the LCD board and rerun the testing with the oscilloscope. If even after that the problem is not solved, look at the IC components on the and test those pins to see if those are correct.

After that we will undergo another series of test to ensure that the resistive touch on the LCD screen works and that it is calibrated correctly. To start off with the power up sequence, we will

first provide the correct voltage and current to the correct pins and see if the resistive touch of the LCD screen there is a tool created by Texas Instruments to calibrate any chosen screen. To use it, one must download and run the open source code provide by Texas Instruments.

10.2 Communication Testing - JG

To test Texas Instrument's WiFi Hardware capabilities, we began by purchasing an MSP430F5529 Launchpad board and a CC3100 WiFi booster pack shield. We chose this hardware to start because TI provides various tutorials online as part of its SimpleLink series. The hardware didn't set up as simple as the videos online made it seem. We found out that the CC31xxEMUBOOST board was required in order to flash the SimpleLink SDK to the CC3100. The CC32xx series of boards would have had the WiFi and emulator built in, but for our testing purposes, buying the emulator was cheaper than getting the CC3220.

The MSP430F5529 features a variety of GPIO pins. With the WiFi booster pack shield attached, we can still use most of the pins for our device inputs. To test analog input, we connected a HB-SR04 Ultrasonic Sensor to pins 21-24. The HB-SR04 works by transmitting a signal and listening to receive the signal when it bounces off an object. The time it takes to receive the signal determines how far away the object is. After uploading code to configure the sensor, we were able to output distance data to the COM terminal. The values looked accurate, so our board is reading the analog input correctly. The Ultrasonic sensor works like the humidity and moisture sensors that we want to use for our garden, so it serves as a good example even though we may not use it in our final project.

Communication testing checklist:

- WiFi
 - The device connects to a wireless access point.
 - The device obtains an IP address from the access point.
 - A webpage is created using the IP address given to the device.
- MQTT
 - The device can send an MQTT message to the webpage.
 - The device can receive an MQTT message posted by the webpage.
- Bluetooth
 - The device can broadcast a Bluetooth signal to connect to.
 - The device can transmit data over a Bluetooth connection.
 - The device can receive information over a Bluetooth connection.
 -

10.3 Software Testing - JG

To understand how the project will work before we build it, we purchased development boards to start implementing some of the features we have been researching. Building working software takes persistence and practice, testing the methods we have researched about before building our final product will make sure we are prepared for the project. Testing on development boards we are familiar with makes comprehension of the new programs we are writing come easier. Each step in the testing process will bring us closer to achieving a working prototype of our device.

10.3.1 WiFi Testing - JG

Since our device will need to connect to the internet, we made it a priority to get software running that enables us to communicate with the embedded controller online over local internet connectivity. In Energia, we followed the Simple Wifi Server demo, found in the example files that comes installed with the IDE. The only edit needed to get the code working is to change the network SSID and password to be congruent with the network credentials we want to connect our microcontroller with. After trial and error trying to connect to a home network, we realized that security measures imposed by the internet service provider make this difficult. By connecting a computer and embedded controller to a phone's hotspot wireless network, we were able to get the demo program working to where the launchpad's LEDs could be controlled by going to a webpage hosted by the MCU.

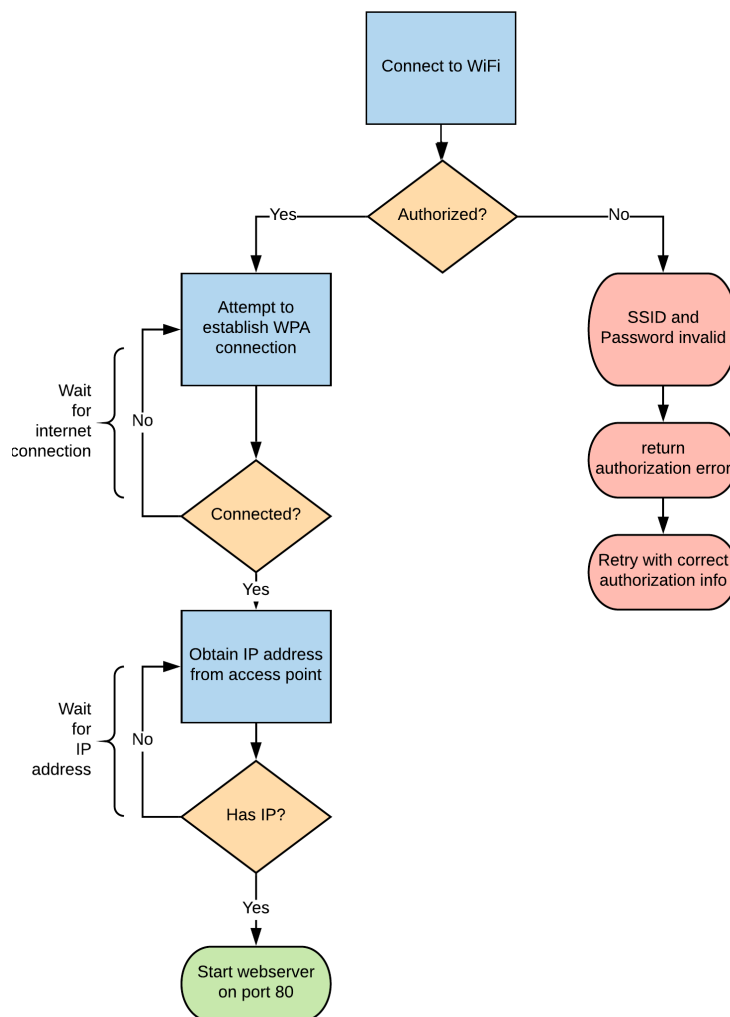


Figure 49: WiFi software block diagram

10.3.2 Bluetooth Testing - JG

Helpful header files will ease the process of programming Bluetooth code. In Energia, we set up communication with *ble.begin()*. The data comes through serial reads. The master Bluetooth node will be set up as the access point. Serial communication is done over UART.

10.3.3 MQTT Testing - JG

To send data over WiFi, we will want to use MQTT. This protocol requires a broker, which is a third party server that routes published MQTT packets to their appropriate topics. Eclipse is an open source IoT provider that we can set up an MQTT server on. To program our device to publish MQTT messages, we can use the Pub/Sub client header in Energia. The Pub/Sub libraries allow us to create a PubSubClient object and call a connect function to the channel we set up in Eclipse. Once the connection is made, we call the subscribe function in the client object to subscribe to a certain topic in the channel we are connected to. To send data, we simply call the client's publish function and define the topic we want to push data to, along with the message we would like to send. The example demo on Energia sends a "hello, world" message, but an MQTT message can also contain sensor data either as a character string or as a JSON formatted string. Sending our information in JSON format will easily allow our MQTT message to be passed along to various APIs.

10.3.4 PubNub Testing -JG

We can also use PubNub as our MQTT broker and subscribe to the channel with widgets in freeboard.io to monitor sensor readings. To test this out, we hooked up an MSP430F5529 with a CC3100 Booster pack and attached an HCRS04 Ultrasonic sensor to give an analog reading. Our smart garden probably won't need a distance meter, but the analog reading we get from the sensor mimics the readings we might get with the other sensors we will be using in our project. Setting up a PubNub account is required to use their broker resource. After creating an account, we created a new project and made a new channel and pub/sub keys. The channel name and keys will be needed by our publisher, in this case our MSP430, so we define them as constants in the code. To set up our MQTT test, we based our code off of the PubNubJsonWiFi example provided in Energia. Library files for PubNub are included in the latest version of Energia and provide the functions needed to begin a connection. To pass messages along, we create a JSON object since its structure will be easy to parse by our dashboard in freeboard.io, and most other online applications. The JSON message consists of our device name and the data we store from reading the sensor.

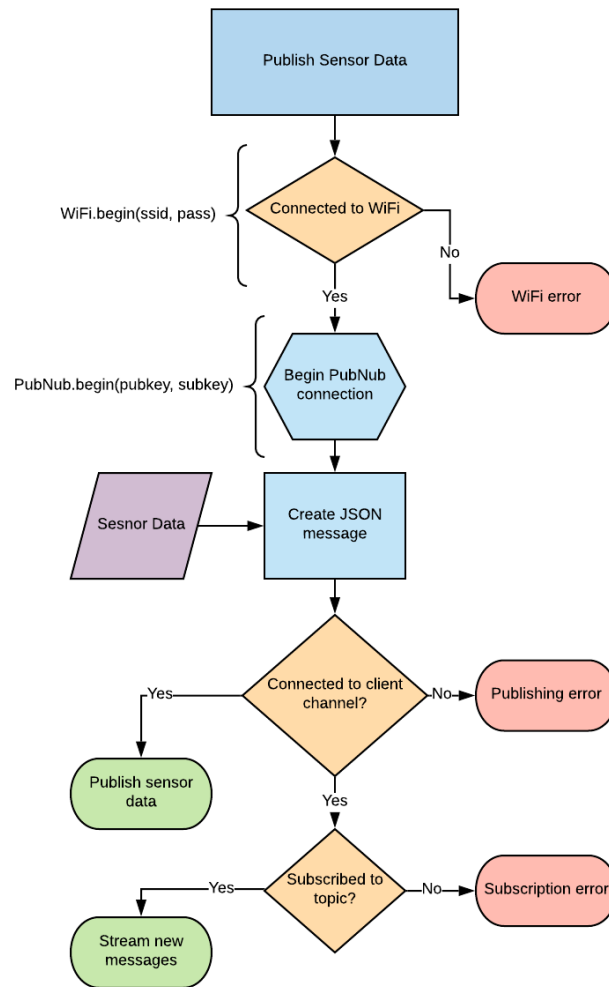


Figure 50: PubNub communication block diagram

After the appropriate set up is done, the main loop in the code establishes a WiFi connection, creates a JSON message with the sensor data, converts that JSON to a string, then publishes the stringified message to the PubNub channel we specified. The data can be viewed in the debug terminal on PubNub to confirm it is sending. For other debug outputs, a serial output can be referenced in the com terminal of Energia. WiFi will need to be either open or on a mobile hot spot to connect, home internet routers will have security measures to block unsecure devices. Once the data is on PubNub, we use freeboard.io to visualize the data. Freeboard allows us to create widgets that subscribe to the messages we publish to the PubNub channel. For this example, we made a distance sparkline that parsed our JSON message for the ultrasonic sensor analog reading shown in this figure.

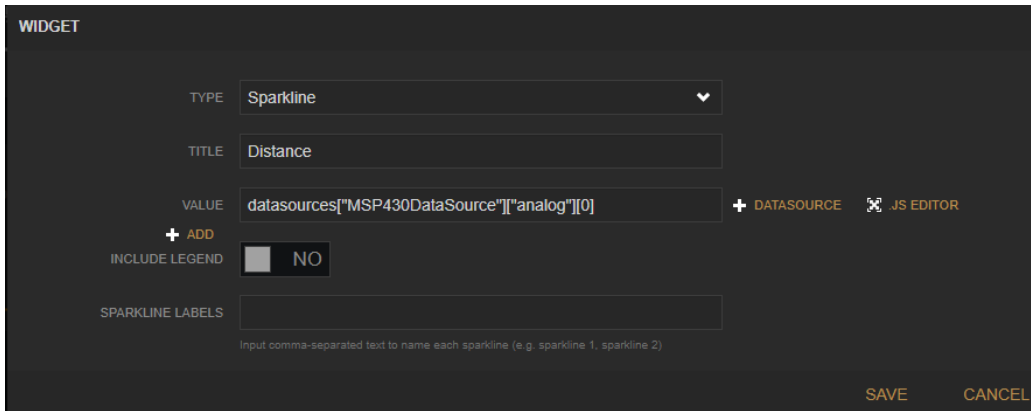


Figure 51: Creating a widget on Freeboard. JSON format makes accessing our sensor data easy.

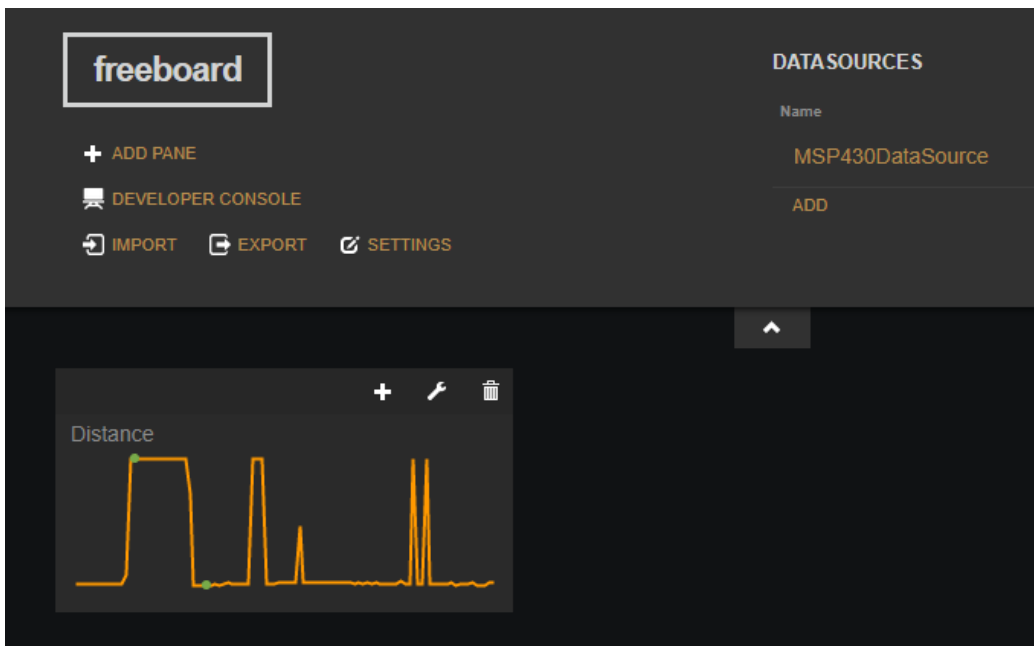


Figure 52: Freeboard Dashboard. This is where we can visualize our data sent from the microcontroller to PubNub using MQTT

10.3.5 Bluetooth Testing - AL

In order to test out if our Bluetooth module is working correctly, we will use AT commands to attempt to control it. From our Code Composer, we would be attempt to power the Bluetooth module on the all of the modules one at time and see if at least a regular laptop/PC can pick the device up. Some steps that we might take would be to connect a breakout board directly from the module in testing to be connected to our PC via FTDI cable. We would could then download any sort of application on our PC to be able to receive/send basic data to our HM-10 device, such as BLE Scanner, LightBlue, nRF Toolbox App, etc. Once everything is hooked-up correctly we should attempt to connect to the Bluetooth module on the application, the default name of this device is “HM-10”. After that we would open either command line or any type of GUI that would

be able to send commands to the Bluetooth module and attempt to strings from our application to our module and visa versa and see if we received them.

Once we get the hang of sending and receiving strings from our terminal, we would practice sending AT commands to see the incoming response. Commands such as “AT+ADDR?” and expect to see the native address, “AT+SLEEP” to set the device in sleep mode and to wake the device up, send a large string, etc.

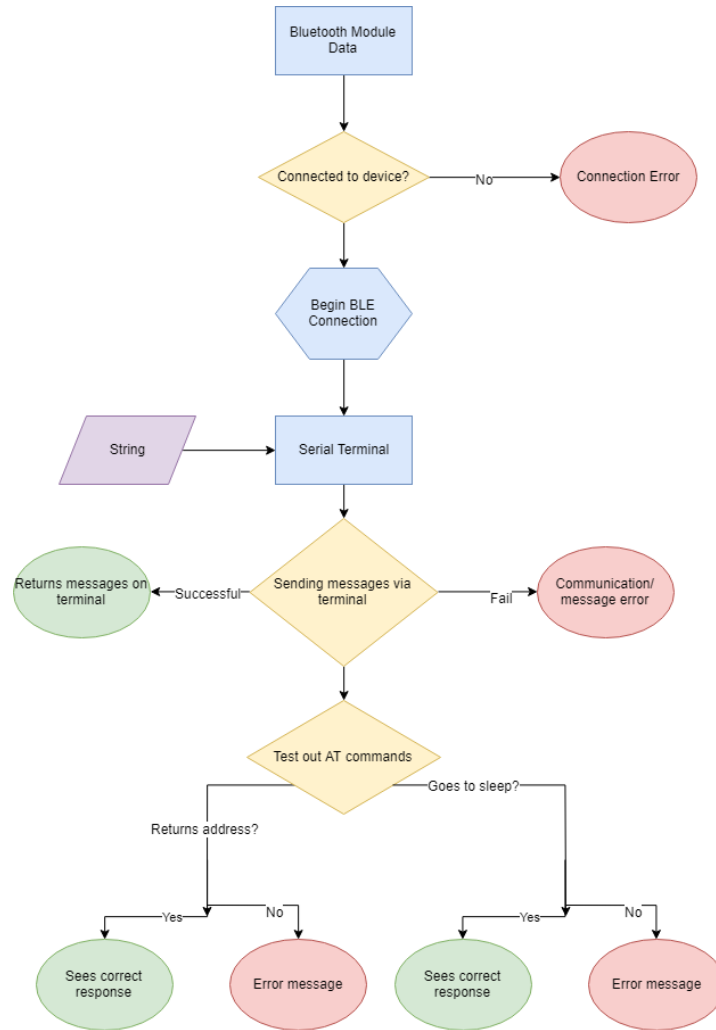


Figure 53: Example Bluetooth Test Sequence

10.4 Prototype - MM

In order to verify that the design of the *SmartLeaf* system is sound prior to installation within the greenhouse enclosure, a prototype of the entire system is to be built and tested. This will provide an opportunity to test the power and controls circuitry for the system prior to finalizing the design of the printed circuit board. Additionally, the compatibility of the various environmental control systems can be modified to create the ideal maintenance routines. The construction and testing of the prototype will need to be conducted in a lab setting with the proper equipment needed to

troubleshoot and verify that the voltage and current characteristics are in the range of their simulated values. Equipment available in the Senior Design and TI Innovation labs should be sufficient for testing. The following data shown in Table 25 below outlines the equipment needed for testing and provides a justification for their inclusion.

Table 25- Prototype Testing Equipment

Equipment	Justification
Digital Multimeter	Used to verify voltage and current characteristics; troubleshooting
Oscilloscope	Used to measure ripple voltage and transient responses; identify interference issues; troubleshooting
DC Power Supply	Used for device isolation; troubleshooting
Variable Potentiometer	Used for device isolation; troubleshooting

10.5 Prototype Planned Pin Layout -AL

In the case of repairing any of the peripheral devices connected to the microcontrollers, attempting to add additional sensors, or troubleshooting any problems that were to arise in the future, this section serves as a guideline to how we initially wired up everything to the microcontrollers and the pins that they used.

10.5.1 Parent Microcontroller - AL

As we are using a MSP432 as the chip to control our main systems, there were a variety of types of data to send. The whole MSP432 device itself has four 16-bit timers with PWM, two 32-bit times a RTC, up to eight serial communication channels (I²C, SPI, UART, and IrDA) so we had to be careful to not hook up the wrong port to a device that needs another type of signal. The 40-pin layout found in its datasheet shows what signals can be configured on each pin on the MSP432, while Table 29: MSP432 Breakout Pin Selection shows the signals we needed from the device and how we utilized the MSP432 to satisfy them.

There were many pins on the actual MSP432 that were not used though within the 40-pin breakout provided from the breakout board, the shared signals can be shown on Table 29: MSP432 Breakout Pin Selection.

10.5.2 Child Microcontroller - AL

As the same as section Parent Microcontroller - AL, this section is to describe the how we are utilizing a MSP432 as the chip to control our main systems, there were a variety of types of data

to send. The whole MSP430 device itself has two general-purpose digital I/O pins connected to LEDs, two push button, a high-quality 20-pin DIP socket for an easy plug-in to the target device. The different signals the specific MSP430G2553 has a 17-bit MSP430 microcontroller with an 8-channel 10-bit ADC, on-chip comparator, 16kB flash memory and 512 bytes of RAM we had to be careful to not hook up the wrong port to a device that needs another type of signal. The 20-pin layout shows how the 20-pin breakout is configured on the shows the signals we needed from the device and how we utilized the MSP430 to satisfy them.

11.0 Greenhouse Construction - MM

This section serves to detail the physical design and construction of the indoor greenhouse system. Additionally, the orientation and mechanics of the sensors and various control systems will be outlined. The concept and basis of design can be seen in Figure 54 and Figure 55 below.

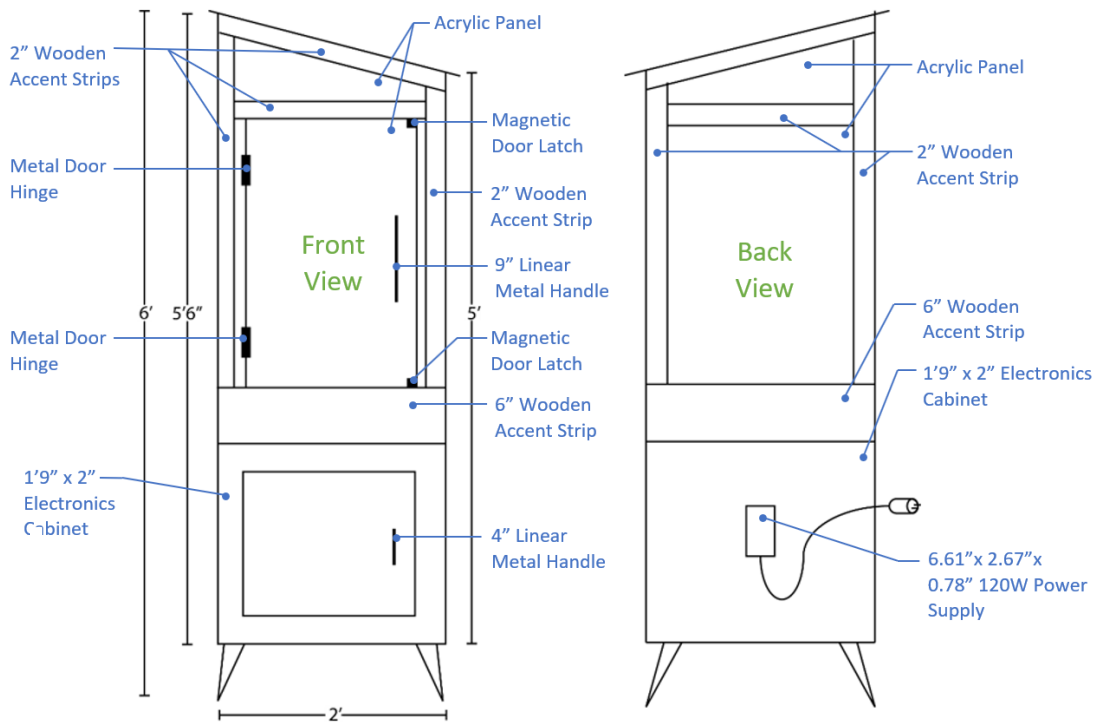


Figure 54- Front and back view

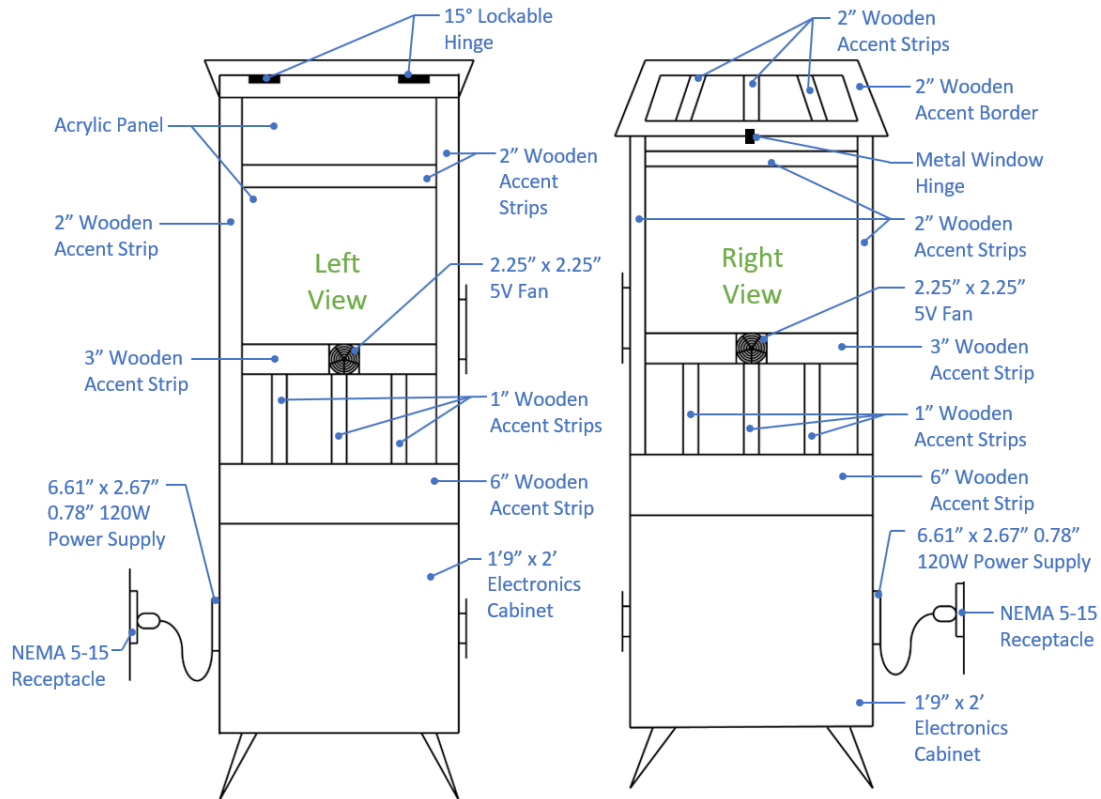


Figure 55- Side views

11.1 Placement and Mechanics of Environmental Controls - MM

Four main controls systems will be used to regulate the environmental conditions within the greenhouse. These systems provide light, water, humidity, and airflow to the plants on a cyclical schedule or interrupt-procedure triggered by sensor readings. All hardware within the visibly exposed upper portion of the unit will have to be cleverly disguised as to not detract from the interior vegetation.

11.1.1 LED Placement - MM

The LED strip lights will be affixed to the roof of the unit in three strips, concealed from the clear acrylic paneling with 2" ornamental strips. Due to the density on LEDs on each strip, this will provide more than enough light coverage and energy provided to the plants when operating at maximum power supplied from the respective switching regulator. For added convenience, the roof will have a latch that the user can prop up perpendicular to the floor, which will provide additional access and air circulation to the plants below. The design decision adds some complication to the LED placement, as the wires supplying the strips are subject to wear from making the nearly 15° movement. Enough slack in the wire and a potential shifting mechanism is to be utilized to protect the circuitry.

11.1.2 Watering Mechanism - MM

The system will employ a 12V brushless submersible pump to deliver water from the cabinet enclosure in the base to the two main plant beds based on soil moisture readings from the sensors. Vinyl tubing connected to the pump will be wrapped around the periphery of the plant beds with small slits in this area to allow water to escape into the soil. Testing done during the prototyping stage will determine the correct power level to deliver water to plants in a laminar manner and without excessive vibration and noise. This will be accomplished via pulse width modulation. In Figure 56 below, the general mechanism for delivering the water to the various plant beds is shown.



Figure 56- Watering mechanism demonstration

11.1.3 Fan Placement - MM

The two fans are to be positioned in a way that encourages the circulation of outside air into the upper greenhouse enclosure. Located at a central position on the sides, the fans will be oriented identically – not mirrored – as to pull in and push out the air. This design will ideally provide the plants with enough carbon dioxide for photosynthesis, effectively lower the temperature and/or humidity when needed, and maintain an equilibrium pressure within the unit. The top enclosure of the unit is to be as watertight as the various connections from the bottom will allow to protect the parent printed circuit board and other external components located in the cabinet from water damage. Powerful fans could potentially create some non-ideal pressure conditions, but that likely will not be the case in our unit; regardless, pressure-related issues should at least be considered as a possibility. This orientation should discourage the issue. Care should be taken into the programming for fan speed as to not damage any plant foliage or fruit.

11.1.4 Humidifier Placement - MM

The humidifier is to be placed within the upper greenhouse enclosure to provide air moisture and cooling to the plants within the unit. Due to a general lack of options for devices of this kind, the humidifier selected requires some further assembly to get it in a practical working condition. The water-atomizing disk cannot be placed below water level, but the device itself is only a few millimeters tall. Operating the device in this manner would require constant replenishing of its water source, and overall, would be highly impractical. In order to deliver water to the device without exceeding past its sides, the device can be placed on sponge sitting in a pool of water. While this would lessen the frequency of its water being refilled, it would still need to be replenished more than in desired, not to mention the possibility for bacterial growth that could occur within the sponge. Another option would be to have a cup-like basin for the humidifier water and a small bit of flexible tubing placed inside the cup, like a straw. The top of the straw would be tightly sealed to the bottom of the condenser disk. The disk provides a minor pressure differential that, within the small volume of the straw, will be enough to pull the water up to the top, to then be diffused into the air. This method is the soundest for the *SmartLeaf* system, as it requires infrequent water fillings, and is overall more sanitary. The image shown in Figure 57 shows this mechanism in action. To maintain a clean look, the ‘cup’ will be placed in the corner of the unit, and outfitted in a color scheme that will allow the mechanism to be easily concealed by the foliage.



Figure 57- Basis of design for the humidifier. Permissions requested from IC Station

11.2 Final Planned Pin Layout - AL

In the case of repairing any of the peripheral devices connected to the microcontrollers, attempting to add additional sensors, or troubleshooting any problems that were to arise in the future, this section serves as a guideline to how we initially wired up everything to the microcontrollers and the pins that they used.

11.2.1 Parent Microcontroller - AL

This section is to serve as an easy pin map to show which pins we used on the MSP432 on our parent microcontroller. As we are using a MSP432 as the chip to control our main systems, there are many pins that aren't used in the MSP-EXP432P401R LaunchPad development kit, this goes to inform that some of the connections TX-RX lines in the Bluetooth module and Wi-Fi modules are being used toward the the same pins because the MSP432 only has two sets of TX-RX lines that one of the sets are used for the LCD screen. The final planned pinouts for the parent microcontroller will be explained in Table 29: MSP432 Breakout Pin Selection.

11.2.2 Children Microcontrollers - AL

This section is to serve as an easy pin map to show which pins we used on the MSP430 on our child microcontroller. The peripheral devices connected to this were the CC2541 Bluetooth Module, the LM393 Soil Moisture and the Ezo-pH Sensor for measuring out the status of the plants they were to oversee and send the data back through the Bluetooth module to the parent microcontroller. The only addition we decided to incorporate was a temperature/humidity sensor to one of the child microcontroller as to be able to take the measurement of the whole enclosure. The final planned pinouts for the children microcontroller should be the same as explained in Table 30: MSP430 Breakout Pin Selection.

11.3 GUI User Guide - AL

To have a local interface with the controller and all its peripheral devices we will plan to have a GUI in the front of the Smart Tabletop Greenhouse. This will serve as a local interface for a user to be able to access and control all the environment changes in our greenhouse, look at data provided from sensors, and provide a troubleshooting system that will notify the owner if anything is out of the ordinary for the greenhouse.

To do this we plan on using an application on Code Composer Studio, as we plan to have our controller be from Texas Instruments. In addition to the GUI Composer application being the main compiler for TI components, there are also features in this application to be able to easily access and show data through the chip via graphs along with numbers and words. There are even objects within the application that simulate thermometers to make ease of custom event handling as our data is pre-processed. For our error messages, the composer application has features where pop-up windows can be created just as if on a regular PC to notify its users.

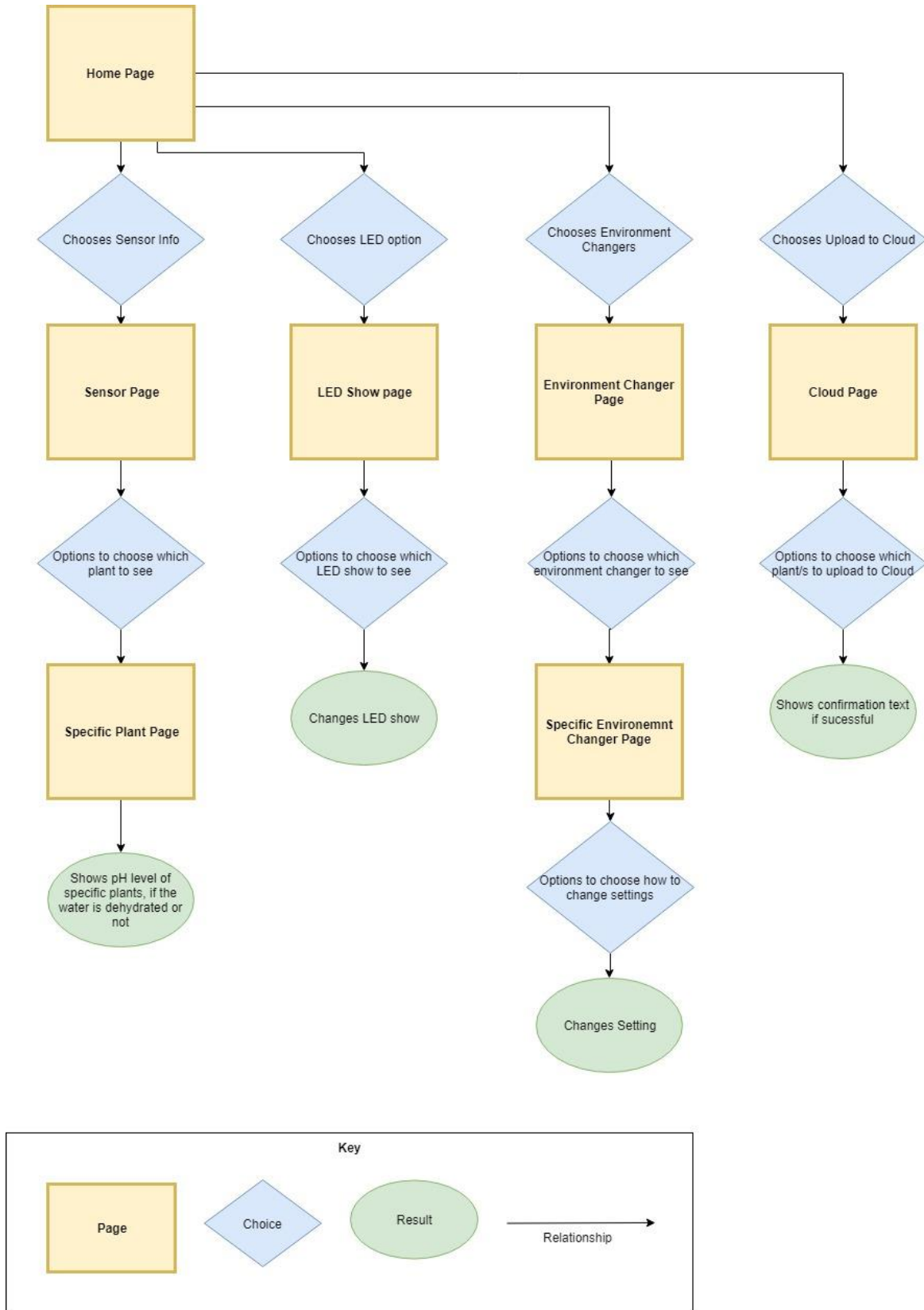


Figure 58: GUI Block Diagram

To describe the series of events happening in the block diagram above, the main sequences of the GUI will work as follows:

11.3.1 Home Page -AL

The main page will have a fun, engaging picture video with all of the creators' names on it on the left half of the screen. The other half of the screen will have four individual clickable box options which will take you either to the sensor page, the LED show page, the environment changer page, or the Cloud page. Once clicked the LCD board will take no longer than 0.5 seconds to load the page from the local memory. With the TI GUI composer, these functions should be easy to implement.

The only thing we have to be prepared for when implementing a GUI from TI GUI Composer is the fact that we have to send off the correct information through the Wi-Fi module to be able to be viewed onto the LCD as well. We will ensure that the GUI is working correctly on the Cloud first to eliminate that the miscommunication errors shown are caused by the GUI were to come locally from the MSP432 to the LCD screen. If errors were to occur on the LCD screen after the verification that the GUI works without the screen we would assume there would be an error of data transfer between the MSP432 and LCD screen.

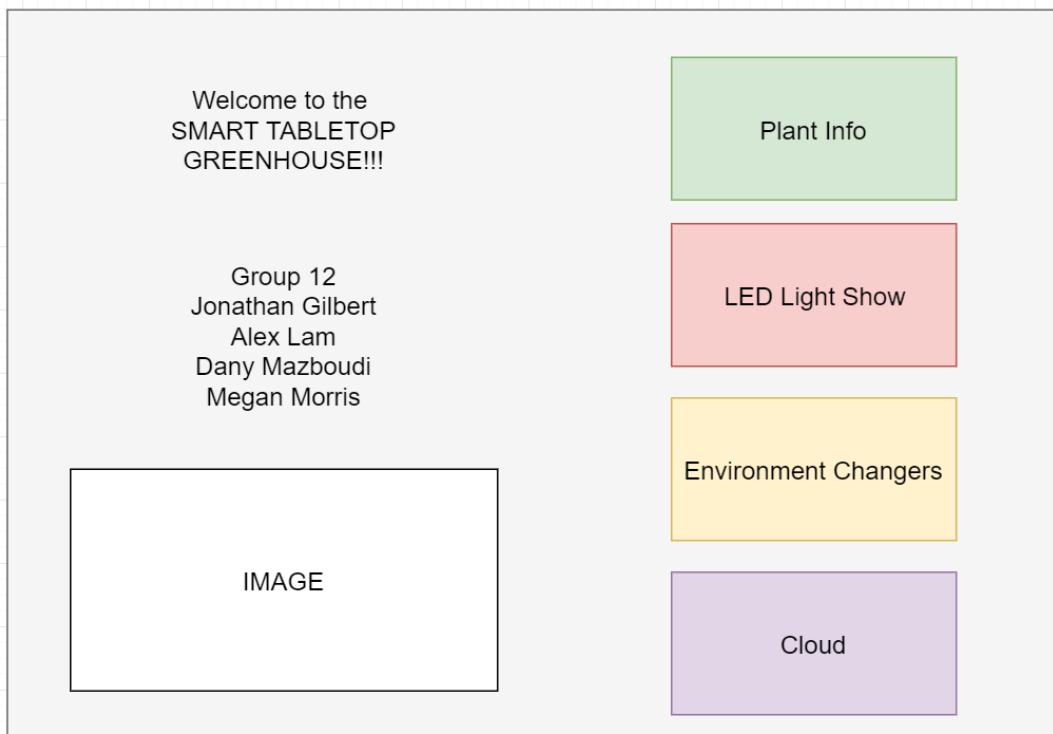


Figure 59: Example Home Page

11.3.2 Sensor Pages -AL

Starting with the sensor section of the flowchart, if clicked from the home page, will take you to a preemptive page with all the currently paired devices to the main ARM processor. The page will be consisted of eight buttons, all of which will contain text that will show the name of the plant a peripheral device will monitor. Each currently paired device will have its own green button and there will a maximum of eight separate devices to represent each plant in the system. If there are less than eight devices paired to the main ARM processor, the extra buttons will be colored red and have text that says “PAIR ME!”. There will be a return home button in the upper right-hand corner of the page also that will take the user to the home page if clicked.

Cases that one would have to be careful on this page would be if you were to click on a green box leading to the sensor of a paired plant yet coming up to a blank page, if you were to click on a red box and to be directed to a known plant page, or if you clicked on a unknown page and were not sent to the sensor connector page. If this is to occur, look back within the GUI code itself and look at the functions controlling these boxes and check back if of the syntax/pinouts are not matching those on the schematic.

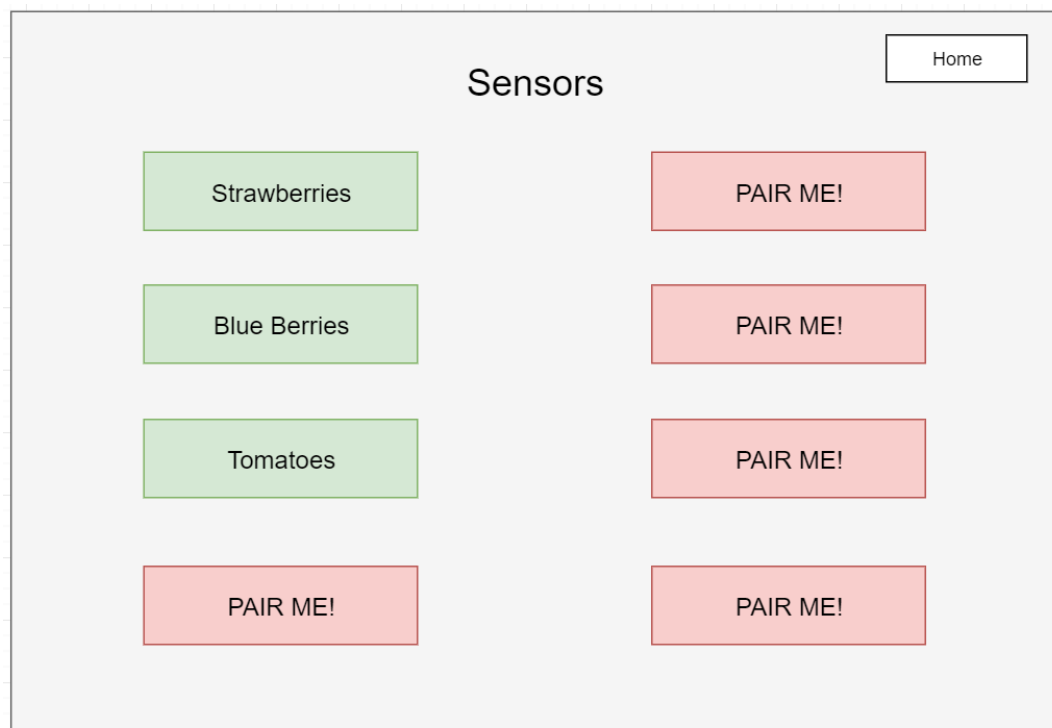


Figure 60: Example Sensor Home Page

11.3.2.1 Connected Plant Page -AL

If the user decides to click on a currently connected plant, it will take the user to another page which will show all the sensors that are connected to that microcontroller, the data that those sensors are gathering and if any of the sensors connected are functioning incorrectly. Some boxes that will be on the page would be for, pH level vs what it should be, last time the plant was watered vs how often it should be watered, how much longer until the plant should be at the ripest yield and if the temperature and humidity is what it should be for the specific plant to grow optimally. If any of the observations from the sensors does not line up to the inputted information, the box fill be colored in yellow, if the sensor itself is not working the box will be filled red, and if everything is up to the optimal region the box will be colored green. Up on the upper right-hand corner will be options to head back to the main sensor page or the main home page.

Cases that we have to be careful on this page would be if there the boxes in green are not matching within the error range we have for the specific measurement; i.e optimal pH level: 6.00, current pH level: 10.11; if the boxes in yellow are within the error range but are not green; ideal humidity:55%, current humidity: 55%; or if the box is red but are getting reading from the actual sensor. If this is to occur, look back within the GUI code itself and look at the functions controlling these boxes and check back if of the syntax/pinouts are not matching those on the schematic.

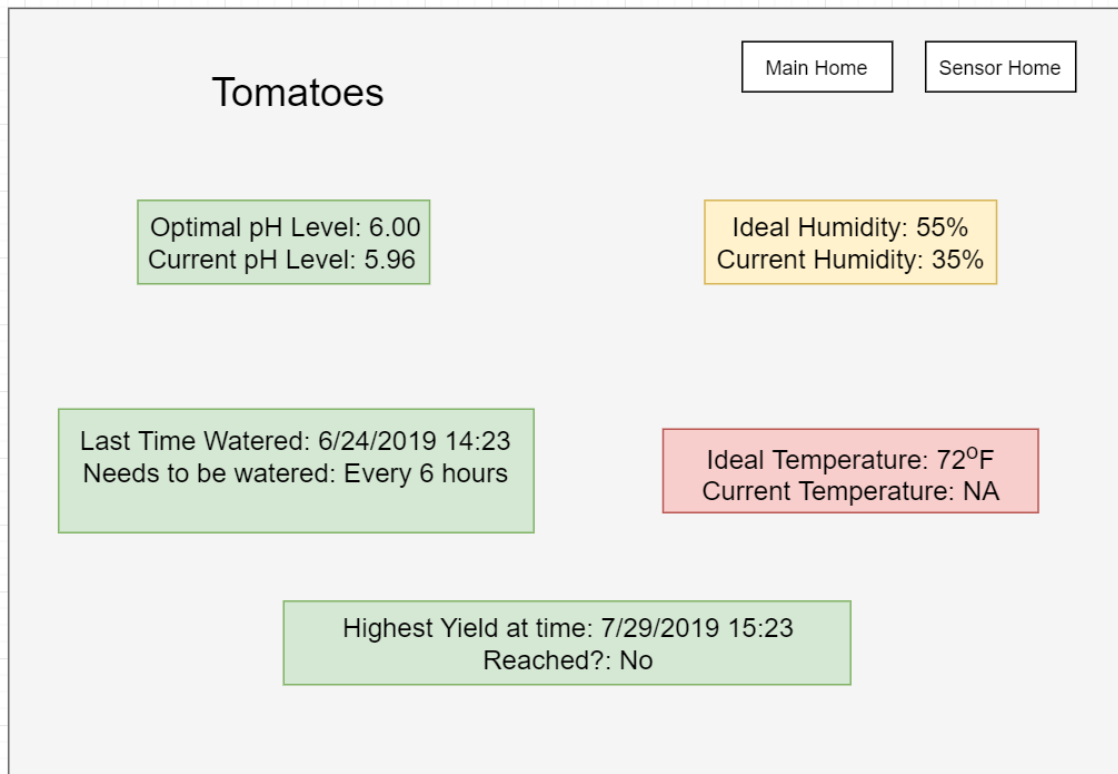


Figure 61: Example Connected Plant Page

11.3.2.2 Pairing New Plant Page -AL

Going back to the sensor home page, if the user instead pushes the “PAIR ME!” option, the user will be directed to a blank sensor page in which the user can input the name of the plant to be inserted in the system, what it’s optimal pH level should be, how often the plant would need to be watered, about how long the plant will take until it should be fully grown, the ideal humidity and ideal temperature that the plant needs to be grown optimally.

The required variable types for each field will be as follows; name of plant will be of type string, optimal pH level will be of type double, ideal temperature will be of type double, ideal humidity will be of type double, and the optimal water schedule will initially be of type char to be casted to type int later. If any of the fields inputted by the user are of the incorrect data type, the GUI will spit out an error text box the says “***** IS FILLED INCORRECTLY!!!”. Will also include two buttons to return with the sensor page and the main home page in the upper right-hand corner.

Cases that one would have to be careful on this page would be if you were to enter the correct information in all of the fields but still getting the error message, entering all correct information and not having the planted added on the Plant Sensor Page, and entering an incorrect data type in the field but still being able to create a new plant. If this is to occur, look back within the GUI code itself and look at the functions controlling these boxes and check back if of the syntax/pinouts are not matching those on the schematic.

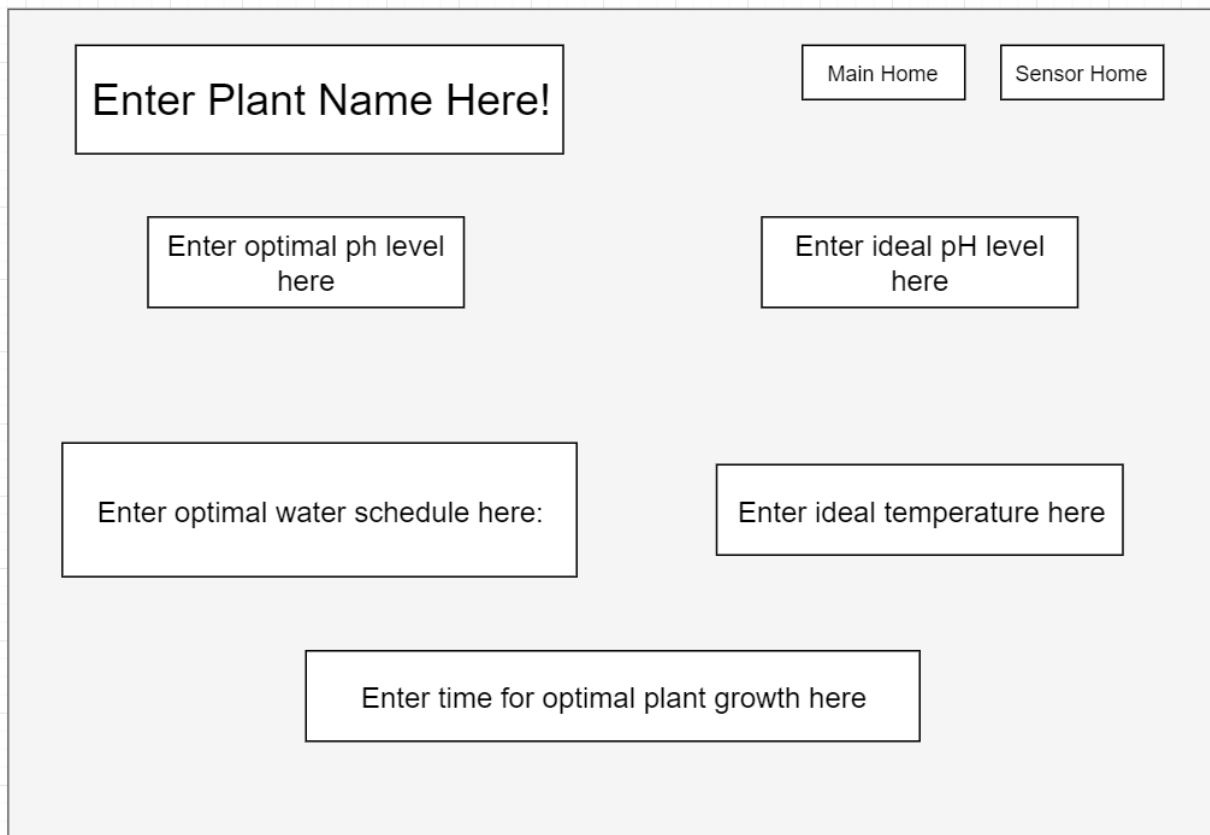


Figure 62: Example Plant Insertion Page

11.3.3 LED Page -AL

With the LED light show section of the flowchart, the user will have the option to change the light settings of the LEDs in and around the enclosure. This page will be split into two sides, the left half of the page giving options on what LED patterns that are programmed to the outside of the enclosure; either having a flowing rainbow pattern, random rainbow patterns, flashing rainbow patterns, and much more. On the right half of the page, eight spots will be given to represent the possible plants that could be put in the enclosure to provide for the plants, these can mimic the amount of sunlight that the specific plant will need for optimal growth in each day. The user can manually manipulate the level of light that is emitted for its specified LED section, and can be adjusted to change the amount of power heading to the section in percentage. Will also include a button to return to the main home page in the upper right-hand corner.

Cases that one would have to be careful on this page would be if you were to change any of the LED options and having no difference in the LEDs after one complete cycle of the previous sequence or nothing were to occur at a max of 10 seconds. If this is to occur, look back within the GUI code itself and look at the functions controlling these boxes and check back if of the syntax/pinouts are not matching those on the schematic.

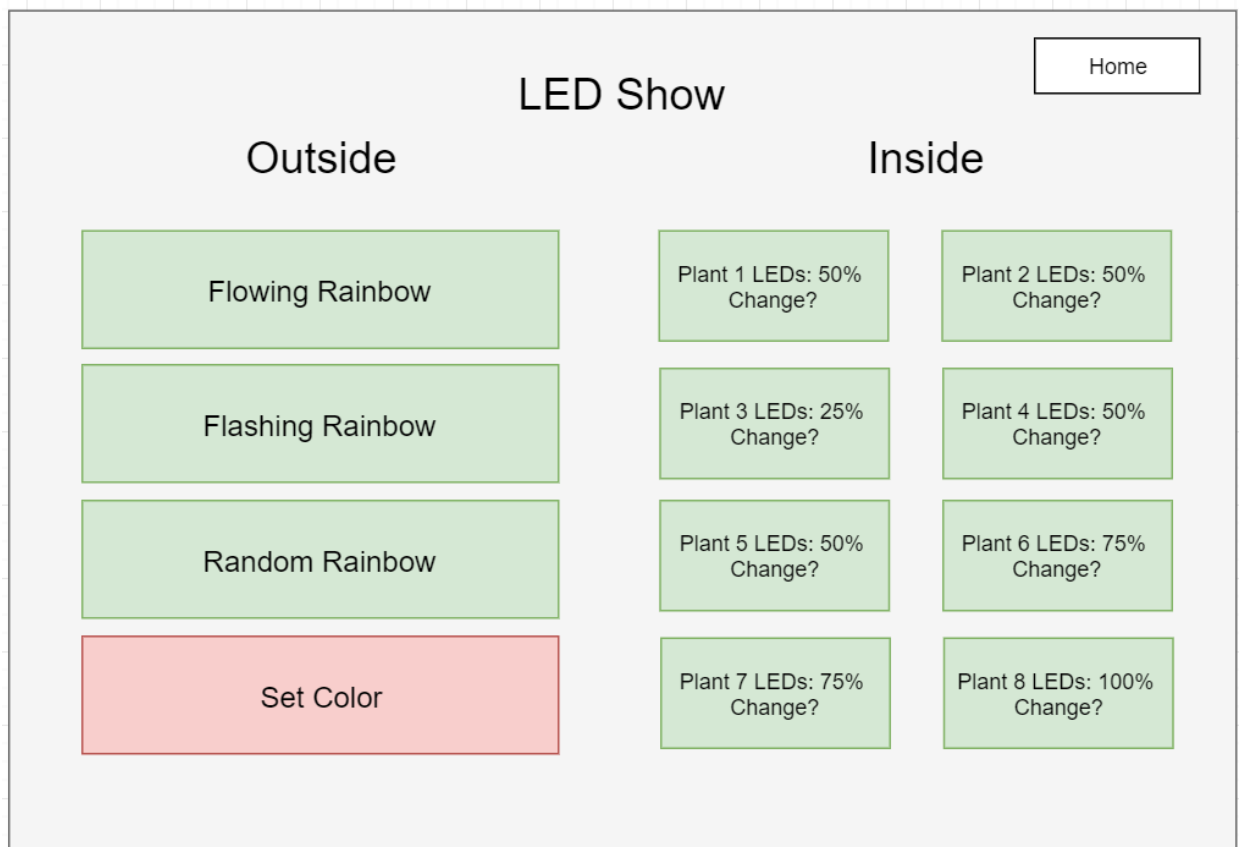


Figure 63: Example LED Show Page

11.3.4 Environment Changer Page -AL

With the Environment Changer section of the flowchart, the user will be directed to a page that has multiple boxes in where they can change the elements that would change the living conditions of the plants inside of the enclosure. There will be a box to adjust the temperature, the humidity, eight boxes to manually water the plants if the user feels like the schedule initially entered for the plant is not enough. Same with the individual sensor errors, if any of the environmental changers are working not to what the user set it to within a hysteresis the box will become yellow, if the ARM processor is not getting a signal at all from the environment changer in question the box will be filled in red. Will also include a button to return to the main home page in the upper right-hand corner of the page.

Cases that one would have to be careful on this page would be if you were to change any of the environment changes and having no difference in the environment changer at max of 10 second, if there is a green box where there is no plant attached in that spot, if the one of the environment boxes is yellow yet none of the environment changers are adjusting the enclosure up to a max of 10 seconds. If this is to occur, look back within the GUI code itself and look at the functions controlling these boxes and check back if of the syntax/pinouts are not matching those on the schematic.

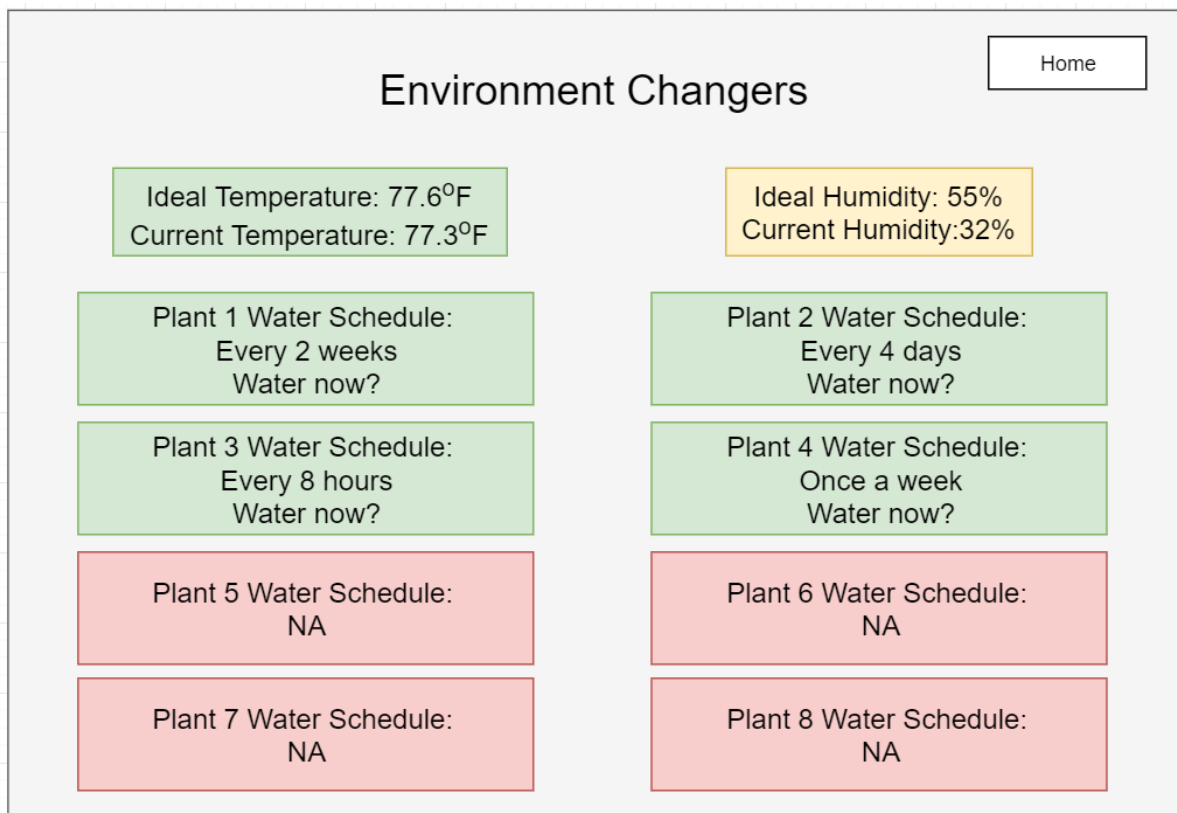


Figure 64: Example Environment Changers Page

11.3.5 Cloud Page -AL

With the Cloud section of the flowchart, the user will have an option to manually upload or schedule a time to upload information up and download from the cloud. This page will feature a scheduler showing when the last time information has been uploaded to the cloud and when the next schedule time will be, another scheduler showing when the last time information was download from the cloud and when the next scheduled time will be, and a two separate buttons for the user to manually upload information from the cloud and download information from the cloud. If any of these process are to go in error, the respective box will be filled in red and a notification will show up inside the box.

With the buttons provided, the code to be able to actually send the data to the cloud might be more difficult than initially expected. This is because if we aren't able to correctly implement the GUI to communicate to the website in the way that we need it to, either the data being sent to the Cloud will be incorrect and stores broken data, or visa versa. One thing that we will have to be cautious of will be getting the correct time stamp while transferring data back and forth. This will ensure that the data we are attempting to send off goes through the correct handshaking procedure.

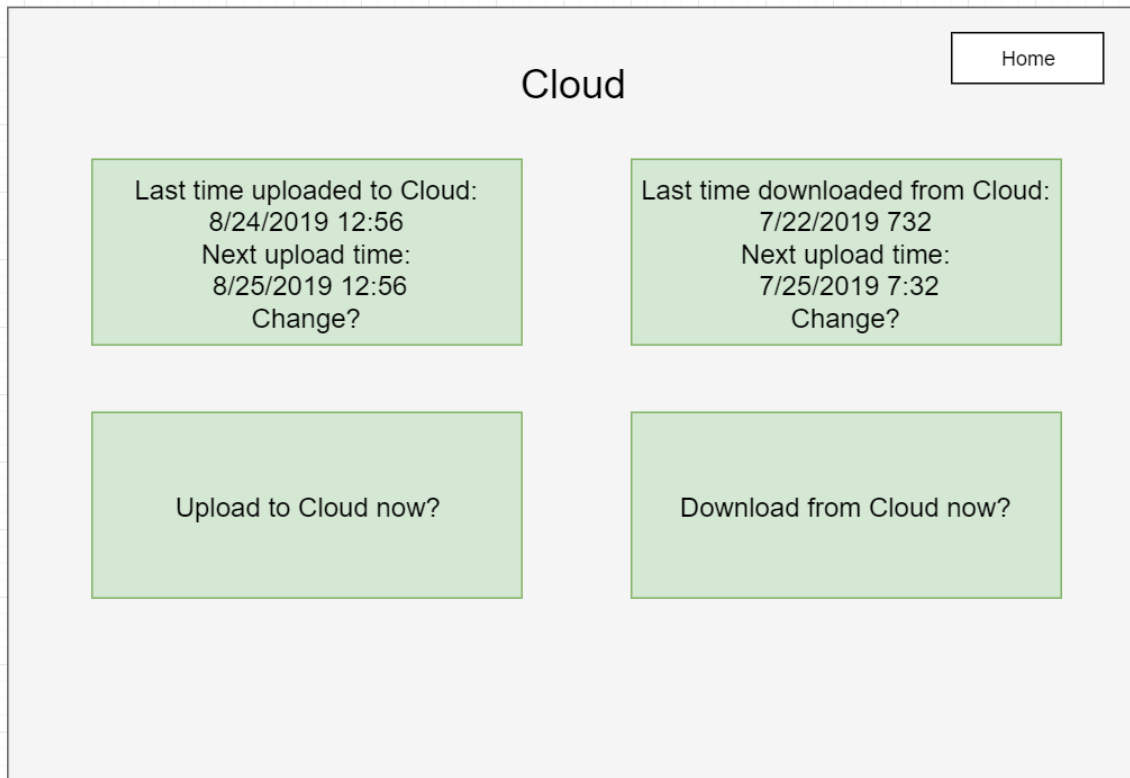


Figure 65: Example Cloud Page

11.4 Project Operation & User's Guide - MM

The *SmartLeaf* indoor greenhouse system – initially without soil and plants – is to be placed in front of a 120V general-purpose receptacle to discretely plug in the device. Before this occurs, the user is to insert soil into the main and peripheral plant beds. This greenhouse system is able to support both plants that have already taken root and seeds for germination. Plants with root systems exceeding 10” are not ideal for this system. Root systems of 6” or more in length should be placed into the main plant bed. Once the plants are arranged to the user’s liking, the soil moisture and pH sensor probes should be placed at least 1” into the soil for both the main plant bed and the vertically oriented peripheral bed, as close to being centered as possible, and adjusting the plants and foliage to conceal the sensor to the user’s best ability. With time, more foliage should grow in and conceal unsightly wires.

After the soil, plants, and sensors have been placed, water can be accounted for. Within the bottom cabinet, three water tanks can be seen- two for fresh water to be delivered to the plants, and one to collect the drainage. To fill the fresh-water tanks, unscrew the top cap, and add distilled water with a watering pot or pitcher. The water level should not exceed the maximum fill line for either tank. When powered on, the system will alert the user when the fresh-water tanks are low, and when the drainage tank is approaching maximum capacity. To empty the drainage tank, simply detach the top tubing, remove the tank from the cabinet, and empty into the nearest sink. Reusing the drainage water within the *SmartLeaf* system is not encouraged, as any dirt or particles from the soil could potentially damage the pumps. The final water source that needs to be addressed is for the humidifier, located in the upper enclosure. Simply fill water to the maximum capacity indicated and ensure the tubing and diffuser disk is returned to its original position within the cup.

At this point, the system is ready to be plugged in. The user should see the LCD display boot and display the welcome screen. Once this occurs, the user will be able to select or modify a routine for the environmental controls of the greenhouse, add a cellphone number for text notifications, and gain instructions for accessing the web app. After this is setup, the user can select back to the home screen, and the environmental conditions within the unit will be displayed.

12.0 Project Administration - MM

Administrative aspects of this project include finances, team designations, and planning for future milestones and deadlines. The following sections were utilized to keep the project on budget and delivered on schedule. It is important for these plans to be discussed with the design team fully and deeply prior to the commencement of planning and purchases. The following budget and milestones presented in this section have been understood and agreed upon at the start of this project.

12.1 Budget - MM

The following chart outlines the component cost for the entire greenhouse system. It should be noted that this cost analysis is an estimation as not all components have been purchased and prices may fluctuate at a later date. Additional costs due to unforeseen circumstances could possibly arise and put more financial strain on the project; design prototypes are to be built and device testing is to be conducted to mitigate and prevent financial loss whenever possible.

While the electronic components have been specified and discussed at length in this document, materials such as building and gardening materials cannot be fully specified at this time and a rough estimate is presented in excess of the expected cost for these items to err on the side of caution and give a more realistic budget for this project.

Table 26- Budget

Item	Quantity	Price	Total Price
Acrylic panels	6	\$7	\$42
Miscellaneous hardware & materials	-	\$100	\$100
Water tank	2	\$5	\$10
Tubing	1	\$13	\$13
Water pump	2	\$8	\$16
Plants & soil	-	\$20	\$20
LED strip lights	1	\$14	\$14
Power supply & voltage regulators	-	\$40	\$40
PCB	1	\$60	\$60
WIFI modules	3	\$10	\$30
Microcontroller	1	\$13	\$13
Humidity/temperature sensor	1	\$9	\$9
Soil moisture sensor	2	\$13	\$26
Soil pH sensor	1	\$30	\$30
Water level sensor	2	\$4	\$8
Humidifier	1	\$7	\$7
LCD display for GUI	1	\$30	\$30
Ventilation fan	2	\$4	\$8
		Total:	\$476

12.2 Milestones

12.2.1 Senior Design 1 Milestones -AL

The following chart is used as a means of staying on target to the deliverable dates. The final outcome of this course is to turn in final project documentation, start general prototype testing, and place initial orders for major components of the system.

Table 27- Senior Design 1 Milestones

Week	Date	Event
3	May 28	Divide and Conquer 1
4	Jun 4	Research
5	Jun 13	Begin writing document
8	Jul 7	At least half of writing is complete
11	Jul 21	75% of the writing is complete
12	Jul 30	Format and print final paper
13	Aug 2	Turn in completed document

12.2.2 Senior Design 2 Milestones -AL

The chart below serves as an expected timeline for the following course in Fall 2019. During this semester, the greenhouse will be assembled and demonstrated to peers, professors, and industry professionals.

Table 28- Senior Design 2 Milestones

Week	Date	Event
1	Aug 26	Order parts
4	Sep 17	Assemble PCB
6	Oct 2	Have completed code
7	Oct 9	Build project
13	Nov 20	Test/finish project
15	Dec 4	Prepare presentation
16	Dec 11	Final presentation

13.0 Final Comments - MM

This section serves to provide final comments after the design stage of this project has been completed, keeping goals for the following semester in mind, and noting areas for possible growth and expansion prior to the delivery of the finished product.

13.1.1 Future Goals - MM

With the commencement of the following semester rapidly approaching, the *SmartLeaf* team will be transitioning from the design phase to the prototype and construction phase. With the full bill of materials documented, the next step is to start purchasing components, building the prototype circuitry, and developing programs to integrate the design vision as a fully realized system. Additional time between the summer and fall semester will be devoted to finalizing a design for the printed circuit board. Areas that are relevant to the structural integrity and aesthetic design of the unit need to be further researched as it lies outside the team's realm of expertise. Advice and guidance from friends, family, and coworkers will likely be needed. With access to power tools at home and at the TI Innovation lab, progress on the frame has already begun.

13.1.2 Potential Areas of Expansion - MM

In the future, granted extra time and a surplus in the budget were found, the current plans for the *SmartLeaf* system could be expanded upon. A fully functional iOS/Android app could be implemented in addition to the standard webpage and text notifications already accounted for. This could potentially spark consumer interest and fall more in-line with comparable units available on the market today.

A more practical area of potential expansion for the *SmartLeaf* system would be to make the unit more appealing for viewers during the daytime. It is already within our means to schedule the red-blue wavelength light periods during the night as to not detract from the organic appearance of the unit and plant life within. In addition to this, a daytime routine of tunable white light could be implemented, spectrally removing the blue wavelengths as the day fades to the afternoon and night. The LEDs are not spectrally balanced enough when producing white light for this to affect the photosynthesis cycle of the plants, but the human eye would be benefited. Tunable color temperature and circadian lighting is becoming more commonplace in the engineering world, and adapting this cycle within our unit would provide additional visual interest.

13.1.3 Conclusion - MM

Throughout this paper, research, constraints, applicable standards, component specification, and overall design were detailed and documented for the *SmartLeaf* indoor greenhouse system. With the initial design completed, the team can now plan for the following term. Next steps include placing orders for components, building prototype circuitry and systems, developing programs and scripts, and commencing the physical construction of the unit.

14.0 Appendices


14.1 Permissions

14.1.1 Niwa One Request for Permission

To customer_support@niwa.com B

Cc

Image Permissions Request



Hello,

I am a senior engineering student at the University of Central Florida. I am writing a report that describes modern-day solutions to indoor gardening and harvesting. I would like to include this image of the [Niwa One](#) unit if you would grant me the permission.

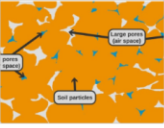
Thank you,
[Megan Morris](#)

14.1.2 Science ABC Request for Permission

To help@science_abc.com

Cc

Image Permissions Request





Hello,

I am a senior electrical engineering student at the University of Central Florida. My senior design project revolves around autonomous greenhouse, and I would like to use this image, if granted the permission, in my discussion about soil requirements. Let me know if you have any questions.

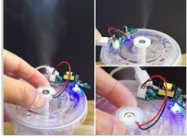
Thank you,
[Megan Morris](#)

14.1.3 IC Station Request for Permission

To  orders@icstation.com  icstation.com@outlook.com

Cc

Image Permissions Request




Hello,

I am writing a paper detailing a greenhouse project for my university, and I am using the IC Station 20mm 5V Humidifier with Driver. I would like to include the attached image, if permissions would be granted from IC Station.

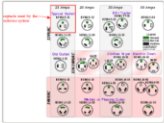
Thank you,
[Megan Morris](#)

14.1.4 NEMA Enclosures Request for Permission

To  customer_service@nema_enc.com

Cc

Image Permissions Request



Hello,

I am a senior electrical engineering student at the University of Central Florida. I would like to include the attached image in my senior design report, in the section where I discuss general purpose receptacles. Let me know if you have any questions.


Thank you,
[Megan Morris](#)

14.1.5 Renesas Electronics Request for Permission

To support@renesas.com

Cc

Image Permissions Request



Hello,

I am a senior electrical engineering student at the University of Central Florida. I would like to include the attached image in my senior design report if granted the permission from Renesas. Let me know if you have any questions.

Thank you,
[Megan Morris](#)

14.1.6 NEMA Request for Permission

Contact Us

NEMA > About Us > Contact Us

Have a question, want to request information, or give us feedback?

* indicates 'Required Field'

Comment Type*

My Name*

My Email*

My Mailing Address

My Organization*

Comments*

14.2 User Pin Layout

14.2.1 Parent controller - AL

Table 29: MSP432 Breakout Pin Selection

Device	Signal Type Needed	MSP432 Pin	MSP432 Pin	Peripheral Device Pin
TFT LCD screen	5V Vcc	J3.1	73	3-5V
TFT LCD screen	GND	J3.2	72	GND
TFT LCD screen	SPI_CLK	J1.7 (P1.5)	11	CLK
TFT LCD screen	MOSI	J2.10 (P3.6)	38	MOSI
TFT LCD screen	MISO	J4.10 (P3.7)	39	MISO
TFT LCD screen	GPIO	J4.9 (P3.5)	37	CS
TFT LCD screen	GPIO	J1.8 (P4.6)	62	D/C
TFT LCD screen	Analog	J3.3 (P6.1)	55	-Y
TFT LCD screen	Analog	J3.4 (P4.0)	56	+X

Device	Signal Type Needed	MSP432 Pin	MSP432 Pin	Peripheral Device Pin
TFT LCD screen	Digital	J2.4 (P5.7)	26	+Y
TFT LCD screen	Digital	J2.8 (P5.0)	64	-X
Fan 1	PWM	J4.1 (P2.7)	23	Data
Fan 2	PWM	J4.2 (P2.6)	22	Data
Humidifier	PWM	J4.3 (P2.4)	24	Data
LEDs	GPIO	J1.5 (P4.1)	57	Data
Water Sensor	GPIO	J2.10 (P3.6)	4	Signal
Bluetooth Module	+3.3V	J1.1	13	+3.3V
Bluetooth Module	UART_TX	J1.3 (P3.2)	34	UART_TX_1
Bluetooth Module	UART_RX	J1.4 (P3.3)	35	UART_RX_1
Bluetooth Module	+5V	J3.1	NA	+5V
Bluetooth Module	GND	J3.2	72	GND
Bluetooth Module	Analog In	J3.7 (P4.5)	61	AUD_FSYNC
Bluetooth Module	Analog In	J3.8 (P4.7)	63	AUD_CLK
Bluetooth Module	Analog In	J3.9 (P5.4)	68	AUD_DOUTIN
Bluetooth Module	Analog In	J3.10 (P5.5)	69	AUD_DINOUT
Bluetooth Module	GND	J2.1	15	GND
Bluetooth Module	RST	J2.5	83	RST

Device	Signal Type Needed	MSP432 Pin	MSP432 Pin	Peripheral Device Pin
Bluetooth Module	UART_CTS_1	J4.4 (P6.6)	80	UART_CTS_1
Bluetooth Module	UART_RTS_1	J4.5 (P6.7)	81	UART_RTS_1
Wi-Fi Module	+3.3V	J1.1	13	Vcc (3.3V)
Wi-Fi Module	UART1_TX	J1.3 (P3.2)	18	UART1_TX
Wi-Fi Module	UART1_RX	J1.4 (P3.3)	19	UART1_RX
Wi-Fi Module	SPI_CLK	J1.7 (P1.5)	9	SPI_CLK
Wi-Fi Module	GND	J2.1	72	GND
Wi-Fi Module	PWM	J2.2 (P2.5)	21	IRQ
Wi-Fi Module	SPI_CS	J2.3 (P3.0)	32	SPI_CS
Wi-Fi Module	SPI_MOSI	J2.6 (P1.6)	10	SPI_MOSI
Wi-Fi Module	SPI_MISO	J2.7 (P1.7)	11	SPI_MOSI
Wi-Fi Module	+5V	J3.1	NA	+5V
Wi-Fi Module	GND	J3.2	82	GND
Wi-Fi Module	UART1_CTS	J4.4 (P5.6)	70	UART1_CTS
Wi-Fi Module	UART1_RTS	J4.5 (P6.6)	80	UART1_RTS
Wi-Fi Module	UCA1TXD	J4.7 (P2.3)	19	NWP_LOG_TX
Wi-Fi Module	Analog In	J4.8 (P5.1)	65	WLAN_LOG_TX

14.2.2 Child Controller - AL

Table 30: MSP430 Breakout Pin Selection

Device	Signal Type Needed	MSP430 Pin	MSP430 Pin	Peripheral Device Pin
Bluetooth Module	+3.3V	J1.1	1	Vcc
Bluetooth Module	GND	J2.1	14	GND
Bluetooth Module	UART	J1.3 (P1.1)	3	TXD
Bluetooth Module	UART	J1.4 (P1.2)	4	RXD
Temperature/Humidity Sensor	+3.3V	J1.1	1	Vcc
Temperature/Humidity Sensor	GPIO	J1.5 (P1.3)	8	Data
Temperature/Humidity Sensor	GND	J1.6 (P1.4)	20	GND
pH Sensor	+3.3V	J1.1	1	+3.3V
pH Sensor	GPIO	J1.7 (P1.5)	9	Data
pH Sensor	GND	J2.1	20	GND
Soil Moisture Sensor	+3.3V	J1.1	1	+3.3V
Soil Moisture Sensor	GPIO	J2.6 (P1.7)	10	Data
Soil Moisture Sensor	GND	J2.1	20	GND

15.0 Works Cited

- [1] B. & J. A. & B. K. C. Wolverton, "Interior Landscape Plants for Indoor Air Pollution Abatement," 1989.
- [2] M.-S. Lee, "Interaction with indoor plants may reduce psychological and physiological stress by suppressing autonomic nervous system activity in young adults: a randomized crossover study.," *Journal of Physiological Anthropology*, vol. 34, 2015.
- [3] OSHA, "Materials Handling: Heavy Lifting," Occupational Safety & Health Administration, Washington D.C., 2019.
- [4] AcuRite, "Acurite," 6 April 2018. [Online]. Available: <https://www.acurite.com/blog/planting-uv-recommendations-and-tips.html>. [Accessed 29 June 2019].
- [5] Worldsemi, "Adafruit," [Online]. Available: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>. [Accessed 30 July 2019].
- [6] R. Benckiser, "Why Plants Need Nutrients," The Royal Society of Chemistry, London, 2019.
- [7] K. LaLiberte, "When is it Warm Enough to Plant?," Gardener's Supply, 2019. [Online]. Available: <https://www.gardeners.com/how-to/when-is-it-warm-enough-to-plant/9029.html>. [Accessed 06 July 2019].
- [8] Wikipedia, "Relative humidity," Wikipedia, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Relative_humidity. [Accessed 06 July 2019].
- [9] J. HENDERSON, "thekitchn," 11 July 2015. [Online]. Available: <https://www.thekitchn.com/everything-you-need-to-know-about-growing-basil-221272>.
- [10] J. F. K. I. V. K. D. Erickson, "Plant Automated Sustainable System (PASS)," UCF Senior Design, Orlando, 2013.
- [11] AcuRite, "AcuRite," 2018. [Online]. Available: <https://www.acurite.com/blog/soil-ph-recommendations-and-tips-for-fruits-vegetables-trees-shrubs.html>. [Accessed 2019].
- [12] AcuRite, 2018. [Online]. Available: <https://www.acurite.com/blog/soil-moisture-guide-for-plants-and-vegetables.html>. [Accessed 2019].
- [13] Adafruit, "DHT22 temperature-humidity sensor + extras," Adafruit, 2019. [Online]. Available: <https://www.adafruit.com/product/385>. [Accessed 4 July 2019].
- [14] Emartee, "High Sensitivity Water Sensor -Red Version," Emartee, 2019. [Online]. Available: <https://www.emartee.com/product/42285/High%20Sensitivit%E2%80%8BBy%20Water%20Sensor%20%20Red%20Version>. [Accessed 3 July 2019].
- [15] Smart-Prototyping, "SOIL MOISTURE SENSOR," Smart Prototyping, 2019. [Online]. Available: <https://www.smart-prototyping.com/Soil-Hygrometer-Detection-Module-Soil-Moisture-Sensor-For-Arduino.html>. [Accessed 1 July 2019].

- [16] A. Scientific, "Ezo-pH Datasheet V5.1," 2019. [Online]. Available: https://cdn.sparkfun.com/assets/2/2/5/8/6/pH_EZO_Datasheet_v51.pdf. [Accessed 2019].
- [17] T. Instruments, "MSP432P401x SimpleLink™ Mixed-Signal Microcontrollers Datasheet," June 2019. [Online]. Available: <http://www.ti.com/lit/ds/symlink/msp432p401r.pdf>. [Accessed July 2019].
- [18] M. Currey, "HM-10 Bluetooth 4 BLE Modules," martyncurrey.com, 5 January 2017. [Online]. Available: <http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/>. [Accessed 31 July 2019].
- [19] GreatScott!, "Bluetooth 2.0 VS Bluetooth 4.0 (BLE) || Is an Upgrade worth it?," 25 March 2018. [Online]. Available: <https://www.youtube.com/watch?v=1i-6cz4KHXE>. [Accessed 31 July 2019].
- [20] J. Lindh, "E2E support forums: Sleep & Wake-up on CC2541," Texas Instruments, 9 December 2012. [Online]. Available: <http://e2e.ti.com/support/wireless-connectivity/bluetooth/f/538/t/232324?Sleep-Wake-up-on-CC2541>. [Accessed 31 July 2019].
- [21] PubNub, "Publish/Subscribe Getting Started Guide," PubNub, [Online]. Available: <https://www.pubnub.com/docs/getting-started-guides/pubnub-publish-subscribe>. [Accessed July 2019].
- [22] AdrianFerTi, "AdrianFerTi," [Online]. Available: <https://www.youtube.com/user/AdrianFerTI/videos>. [Accessed July 2019].
- [23] Amazon, "Getting Started with the Texas Instruments CC3220SF-LAUNCHXL," Amazon, [Online]. Available: https://docs.aws.amazon.com/freertos/latest/userguide/getting_started_ti.html. [Accessed July 2019].
- [24] J. Skansholm, "Vägen till C [The road to C]," Studentlitteratur, 1987.
- [25] S. Cass, "IEEE Spectrum," 2018. [Online]. Available: <https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>.
- [26] T. Instruments, "Simplelink WiFi-MQTT," Texas Instruments, [Online]. Available: http://dev.ti.com/tirex/content/simplelink_academy_cc32xxsdk_1_14_02_04/modules/wifi_mqtt/wifi_mqtt.html. [Accessed July 2019].
- [27] T. Instruments, "TI-RTOS: Real-Time Operating System (RTOS) for Microcontrollers (MCU)," Texas Instruments, [Online]. Available: <http://www.ti.com/tool/TI-RTOS-MCU>. [Accessed June 2019].
- [28] Amazon, "Amazon FreeRTOS," Amazon, [Online]. Available: <https://aws.amazon.com/freertos/>. [Accessed June 2019].
- [29] Amazon, "AWS IoT," Amazon, [Online]. Available: <https://aws.amazon.com/iot/>. [Accessed June 2019].

- [30] Dimension Engineering, "A Beginner's Guide to Switching Regulators," Dimension Engineering, 2017. [Online]. Available: <https://www.dimensionengineering.com/info/switching-regulators>. [Accessed 16 July 2019].
- [31] C. Humphrey, "Building codes and greenhouses," Greenhouse Management, 26 July 2010. [Online]. Available: <https://www.greenhousemag.com/article/gmpro-0710-building-codes-greenhouses-state-of-industry/>. [Accessed 27 June 2019].
- [32] John, "PCB Trace-The Importance of PCB Traces In the PCBs," OurPCB, 15 November 2018. [Online]. Available: <https://www.ourpcb.com/pcb-trace.html>. [Accessed 2 July 2019].
- [33] D. Marrakchi, "Top 5 PCB Design Guidelines Every PCB Designer Needs to Know," 30 November 2016. [Online]. Available: <https://resources.altium.com/pcb-design-blog/top-pcb-design-guidelines-every-pcb-designer-needs-to-know>.
- [34] Eessentra, "How to prevent your PCB from overheating," Eessentra, 2018. [Online]. Available: <https://www.eessentracomponents.com/en-gb/news/guides/how-to-prevent-your-pcb-from-overheating>. [Accessed 7 July 2019].
- [35] A. Sherman, "Crash Course in Consumer Electronics Certifications: FCC Regulations, EMC Testing, and More," Fictiv, 2019. [Online]. Available: <https://www.fictiv.com/hwg/plan/crash-course-in-consumer-electronics-certifications-fcc-regulations-emc-testing-and-more>. [Accessed 07 July 2019].
- [36] Wikipedia, "IP Code," Wikipedia, 2019. [Online]. Available: https://en.wikipedia.org/wiki/IP_Code#Ingress_Protection_for_consumer_electronics. [Accessed 07 July 2019].
- [37] IEEE Electron Devices Society, "IEEE Standard for Sensor Performance Parameter Definitions," IEEE Standards Association, New York City, 2014.
- [38] Texas Instruments Inc., "TPS55330 Integrated 5-A, 24-V Boost/SEPIC/Flyback DC-DC Regulator," Texas Instruments Inc., Dallas, TX, 2019.
- [39] Texas Instruments Inc., "LMR62014 SIMPLE SWITCHER® 20V_{out}, 1.4A Step-Up Voltage Regulator in SOT-23," Texas Instruments Inc., Dallas, TX, 2013.
- [40] Texas Instruments Inc., "TPS56637 4.5-V to 28-V Input, 6-A Synchronous Buck Converter," Texas Instruments Inc., Dallas, TX, 2018.
- [41] Texas Instruments Inc., "TPS560430 SIMPLE SWITCHER® 4-V to 36-V, 600-mA Synchronous Step-Down Converter," Texas Instruments Inc., Dallas, TX, 2018.
- [42] Texas Instruments Inc., "TPS6217x 28-V, 0.5-A Step-Down Converter With Sleep Mode," Texas Instruments Inc., Dallas, TX, 2015.
- [43] En.wikipedia.org, "Photosynthesis," Wikipedia, [Online]. Available: <https://en.wikipedia.org/wiki/Photosynthesis>. [Accessed 7 July 2019].

- [44] C. Speake, "Shade Tolerant Vegetables vs Sun Friendly Veggies," The Gardening Cook, 2019. [Online]. Available: <https://thegardeningcook.com/sun-or-shade/>. [Accessed 6 July 2019].
- [45] Ashish, "Why Does Over-Watering Kill Plants?," Science ABC, 2019. [Online]. Available: <https://www.scienceabc.com/nature/why-does-over-watering-kill-plants.html>. [Accessed 06 July 2019].
- [46] Wikipedia, "NEMA connector," Wikipedia, 2019. [Online]. Available: https://en.wikipedia.org/wiki/NEMA_connector. [Accessed 07 July 2019].
- [47] J. Errington, "Linear DC Power Supply design," Skillbank, 2019. [Online]. Available: <http://www.skillbank.co.uk/psu/>. [Accessed 07 July 2019].
- [48] R. Bohn, "IP Ratings Explained - What Are IP Ratings? | NEMA Enclosures," NEMA, 2019. [Online]. Available: <https://www.nemaenclosures.com/blog/ingress-protection-ratings/>. [Accessed 07 July 2019].
- [49] D. Marrakchi, "Top 5 PCB Design Guidelines Every PCB Designer Needs to Know," Altium, 2016 November 2016. [Online]. Available: <https://resources.altium.com/pcb-design-blog/top-pcb-design-guidelines-every-pcb-designer-needs-to-know>. [Accessed 4 July 2019].
- [50] Adafruit, "Adafruit STEMMA Soil Sensor - I2C Capacitive Moisture Sensor," Adafruit, 2019. [Online]. Available: <https://www.adafruit.com/product/4026>. [Accessed 2 July 2019].
- [51] B. Bergquist, "Standards defining Temp Sensor Notes," Burns Engineering, 2019. [Online]. Available: <http://www.burnsengineering.com/local/uploads/content/files/Standards%20Defining%20Temp%20Sensors%20Notes.pdf>. [Accessed 6 July 2019].
- [52] Adrianf. [Online].