EcoSense

ECE 44x
Group 27
IoT Group 2
By Alex Feng, Carson Ehlers, Peter Thompson, Aiden Olsen

Table of Contents

Section 1: Overview

1.1 Executive Summary:

Our node/hub-based system is capable of intelligent environmental and home sensing. As it stands right now, our system consists of a central hub connected to our Garden Node. This node is self-powered by solar energy and can measure air temperature, humidity, soil moisture, and light intensity. The central hub node pulls data from each node, does any needed processing, and uploads the data to the cloud. The communication protocol we have chosen to use between the nodes and central hub is Matter, a new IoT framework. An Android app will be able to communicate with the hub for device configuration, initial setup to onboard devices onto the Matter network, and configuration of automations. These automations will be able to process data from garden node sensors to provide actionable insights and control of other devices, such as water valves for proper plant watering.

1.2 Team Contacts and Protocols:

Name	email	Roles
Peter Thompson	thomppet@oregonstate.edu	Hub power supply and Enclosures
Aiden Olsen	olsenai@oregonstate.edu	Schematic Capture, PCB Design, Component Selection
Carson Ehlers	ehlersca@oregonstate.edu	Hub Electronics, Hub GUI
Alex Feng	fenga@oregonstate.edu	Matter Integration, Cloud computing, Android app

1. Show up to meetings on time and communicate absences when needed.
2. Keep up to date on where we are at on the project over Zoom/Discord
3. Meet once a week on Wednesdays at 4:15 PM. Additional meetings can be scheduled after being agreed upon by the whole team.
4. Meeting locations and format will be determined before each meeting by the team. (Zoom or in person)
5. Maintain professional communication between team members throughout the project.

1.3 Gap Analysis:

- The reason your project exists

    - Soil moisture sensors aren't currently available in the market for home systems. There are plenty of soil sensors that are made specifically for the agricultural systems, but they aren't necessarily user friendly or easily integrated into existing home automation systems. The closest thing we have is a water leak sensor.
- Assumptions you are making about the need(s) the project will fulfill
    - Thread hub will be built on top of an existing piece of Linux hardware
        - Customers who don't want to use our hub may be able to integrate our sensors into their own hubs such as a Google Nest Hub or Apple Homepod Mini
    - Customers will have a phone
    - Customers have a lawn (so not necessarily applicable to people who live in apartments)
- Any relevant results of informational interviews with project partners, potential customers, or online research that impact your understanding of how the project will fulfill a need.
    - The new Matter protocol is supposedly going to be the future of smart home ecosystems. It standardizes the communication protocols between smart home devices which include WiFi, Bluetooth, and Thread technologies. We intend to implement our communication protocol using Thread, which is currently popular in Google and Apple smart home ecosystems.
    - Soil moisture sensors for home environmental sensing aren't currently available on the market. Soil moisture sensors are only made for agricultural systems, therefore it isn't easily integrated into your existing home.
- A description of your end user(s) and any other key end-product stakeholders based on the above information
    - People that own a home with a lawn
    - Could also potentially be used for general industrial and agricultural ecosystems that require environmental sensing of the soil (e.g. for smart plant/lawn watering, HVAC heating/cooling)

1.4 Proposed Timeline:

Phase	Task	Start date	End date	Status
Planning	Research communication protocol	10/14/22	10/21/22	Completed(10/20/22)
	System block diagrams	10/14/22	10/21/22	Completed(10/21/22)
	Component Selection	10/14/22	10/21/22	Completed(11/3/22)
	Concept sketches	10/14/22	10/21/22	Completed(10/20/22)
Prototyping	PCB design	10/24/22	11/03/22	Completed(11/3/22)
	Initial Cloud software	11/28/22	2/10/23	Completed(11/5/22)
	Prototype Android App	11/28/22	2/10/23	Completed(11/5/22)
	Embedded programming	10/17/22	1/5/23	Completed(11/7/22)
	Design enclosure	1/23/23	1/26/23	Completed(11/3/22)
	System Verification	3/3/23	3/11/23	Completed(3/11/23)
Final product	Build Enclosure	3/25/23	4/14/23	Completed(1/8/23)
	Finalize app	2/10/23	4/25/23	Completed(2/10/23)
	Finalize Cloud software	2/10/23	4/25/23	Completed(2/13/23)
	Assemble the device	5/10/23	5/24/23	Completed(2/20/23)
	Final verification	4/3/23	6/3/23	Completed(5/11/23)
Presentation and closure	Creation of presentation	4/3/23	6/3/23	Completed(5/09/23)
	Documentation	10/14	6/10/23	Completed(3/12/23)

1.5 References and File Links:

[1] “CC2652R7 SimpleLink™ Multiprotocol 2.4 GHz Wireless MCU Datasheet”, Rev A, Texas Instruments, November 2021, CC2652R7 SimpleLink™ Multiprotocol 2.4 GHz Wireless MCU datasheet (Rev. A) - cc2652r7.pdf.
[2] UG103.11: Thread Fundamentals, Rev 1.4, Silicon Labs, https://www.silabs.com/documents/public/user-guides/ug103-11-fundamentals-thread.pdf

1.6 Revision Table

10/14/22	Aiden Oslen, Alex Feng, Carson Ehlers, Peter Thompson: Created Section 1
11/01/22	Peter Thompson: Changed the executive summary to fit our goals
11/02/22	Peter Thompson: Updated the Timeline
11/16/22	Alex Feng, Peter Thompson: Updated the timeline. Updated Executive Summary
3/08/23	Aiden Olsen: Updated teammates roles, updated project description to match new Matter use
3/10/23	Aiden Olsen, Alex Feng: Updated revision table w/ in progress and completed sections

Section 2: Impact and Risks

2.1 Design Impact Statement

Due to the accelerated innovation that the world is experiencing in this century, it is easy to gloss over the negative impacts that technology can bring into the fold. With this in mind, our aim for EcoSense is to improve the productivity and efficiency of a consumer’s home while limiting the harmful effects that are created as a result.

Public Safety and Welfare Impacts

With the expected close proximity to growing food, the materials used will have a direct impact on the health of the consumers. This means that if the wrong materials or

integrated circuits are used it can be expected that many chemicals will be leached into the ground, absorbed into the plants, and consumed by people[1]. To minimize the risks to health that placing plastic and circuitry in a garden can lead to. It is best to use food-safe grade HDPE or ASA for their capability for outdoor usage, the circuit should be free of lead.

Cultural and Social Impacts

The Matter protocol used in the IoT project is subject to cybersecurity risks, including energy depletion attacks and online password guessing attacks, which could lead to sensitive information being stolen or devices being controlled by an attacker. To mitigate the cybersecurity risks associated with the Matter protocol, the project team should ensure that devices ignore insecure messages and do a validation check to authenticate received messages before processing them. Data privacy is also an important concern for IoT devices, and it is up to the manufacturer to uphold the integrity of their devices. Companies should take measures to protect data and ensure equal treatment regardless of gender or race [2]. To protect data privacy, the team should reduce the amount of data sent to the cloud, encrypt all sensitive data, and keep as much data as possible only on the local network.

Environmental Impacts

While most lithium-ion batteries are considered safe, there is always the potential for fires and explosions that can lead to more serious environmental issues. Improper disposal of lipo batteries can also be harmful to the environment. When not disposed of properly, lithium-ion batteries can release harmful chemicals and metals into the ground and water [3]. We plan to mitigate these common issues by providing proper lithium charging and disposal instructions in our product.

Economic Impacts

With EcoSense planning to take over the task of gathering environmental data, a glaring economic impact is of course, job displacement. EcoSense will be just one of the many already created agricultural devices that are designed to make agriculture more efficient and to require less labor hours. These reduced labor hours cause people in the workforce to lose their jobs due to them not being needed anymore. At the beginning of the 20th century, it is estimated that up to a third of the American workforce was employed in agriculture. Since then, this number has dropped to around only 2% while the nation’s agricultural production has increased dramatically. This is a result of better technologies and devices [4]. Our goal in creating this environmental sensor is not to replace the jobs of recording environmental statistics but rather to help increase their productivity. With the recordings that will be able to be achieved by our sensor, these jobs will be able to focus more on what to do with the information rather than the time it takes to gather the information. While this negative impact resulting from innovation

cannot be eliminated, it can be avoided where possible and that is our aim for this project

2.2 Risks

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Performance Indicator	Action Plan
R1	Shipping delay	Timeline	M	L	Part does not arrive on time	Wait, Contact supplier / shipping company
R2	Too Expensive	Cost	L	M	We go over budget	Reduce Costs
R3	Out of Time	Timeline	L	H	Incomplete project at the end of spring term	[5] Extra meetings until the project is done
R4	Part out of stock	Timeline	M	L	The needed part is out of stock	Order backups
R5	Task is too challenging / difficult	Technical	L	M	Failure to complete ones tasks	Reorganize the workload
R6	Power supply failure	Technical	M	H	Doesn’t turn on	[6] Redesign and test
R7	High voltage electricity	Health	L	M	Someone injured due to device	[7] Cool with water
R8	Teamwork dissolves as time nears	Social	L	H	Team fallout and disorganization	Have extra team meetings and ask Rachel for help
R9	Part burns out	Technical	L	M	Magic smoke	Go over design and simulations to fix the error or check for a

						short

2.3 References and File Links

[1] Smiths, “Polyethylene (HD-PE & UHMW-PE) Technical Datasheet,” Accessed: Nov. 2 2022, https://www.smithmetal.com/pdf/plastics/polyethylene.pdf
[2] Lippett, Mark. “Privacy, Intelligence, Agency: Security In The Smart Home.” Forbes, May 5 2022, https://www.forbes.com/sites/forbestechcouncil/2022/05/05/privacy-intelligence-agency-security-in-the-smart-home/?sh=4ec7143d4aac
[3] Used Lithium-Ion Batteries, United States Environmental Protection agency, May 24, 2022, https://www.epa.gov/recycle/used-lithium-ion-batteries
[4] How Does Technology Affect Economics , *Bizfluent.com*, November 8th, 2018. https://bizfluent.com/info-8443960-effects-globalization-technology-business.html
[5] “How to create a project timeline: The ultimate guide”, Teamwork.com, February 22nd, 2021, https://www.teamwork.com/blog/project-timeline.
[6] ”LIPO Safety & Warnings”, Tenergy Power, https://power.tenergy.com/lipo-safety-warnings/
[7] “Electrical burns: First aid”, Mayo Clinic Staff, August 05 2022, https://www.mayoclinic.org/first-aid/first-aid-electrical-burns/basics/art-20056687.

2.4 Revision Table

11/02/22	Aiden Oslen, Alex Feng, Carson Ehlers, Peter Thompson: Created Section 2
11/16/22	Alex Feng, Peter Thompson: Added extra risk + modified some action plan descriptions
4/25/23	Added section 2.1

Section 3: Top Level Architecture

3.1 Block Diagrams

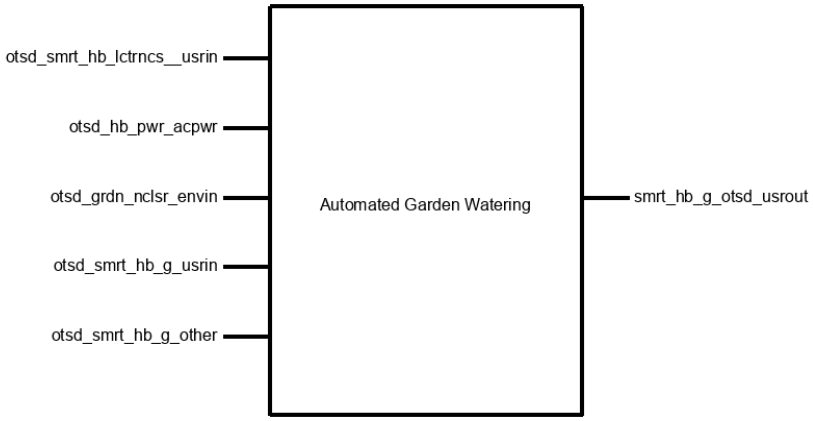


Fig. 3.1.1: System black box diagram

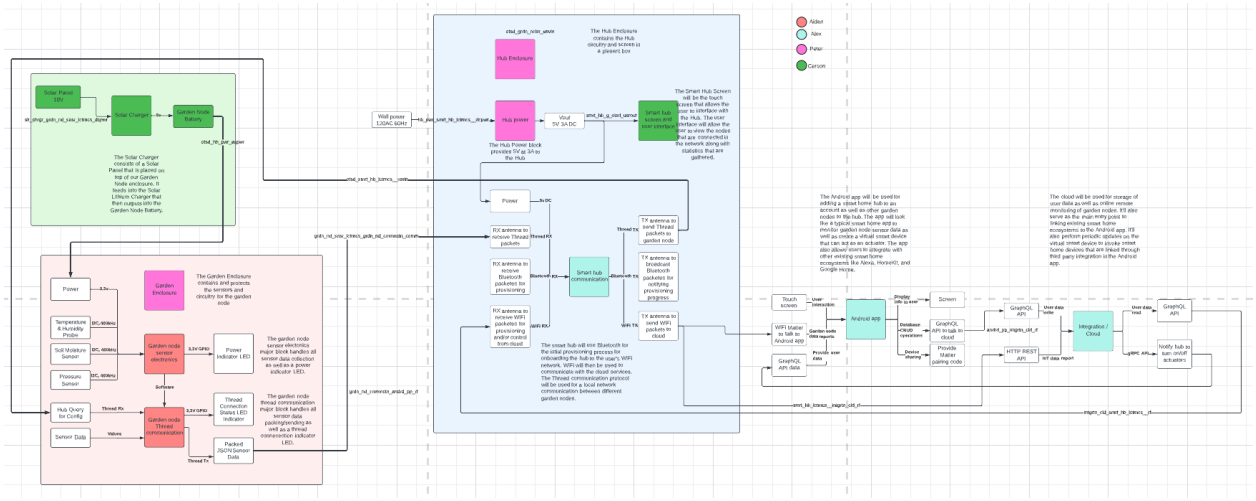


Fig. 3.1.2: System block diagram

3.2 Block Descriptions

[1] Smart Hub Electronics (Champion:Carson Ehlers):
Inputs: *otsd_smrt_hb_lctrncs_usrin, hb_pwr_smrt_hb_lctrncs_dcpwr*
Output: *None*
The Smart Hub electronics consist of a plug and play touch screen that allows the user to interface with the Hub and a raspberry pi. Through the use of the two devices, the user will be able to view the nodes that are connected in the network along with statistics that are gathered. The screen is a 5 inch capacitive touch screen that will be wired into the raspberry pi in order to view and interact with our software. The raspberry pi will be the main brain of the Hub which will eventually handle the communication to all of the nodes on the network. The four blocks that the

Smart Hub Electronics will interact with are the hub power block, the user interface block, the integration / cloud block and the Hub enclosure block. All of these blocks working together will form our final design for the Central Hub in our entire smart network.

[2] Smart Hub GUI (Champion: Carson Ehlers):
Input: *otsd_smrt_hb_g_usrin*
Output: *smrt_hb_g_otsd_usrout*
Input/Output: *otsd_smrt_hb_g_other*
The Smart Hub GUI is the graphical user interface that will be loaded onto the Hub's Raspberry pi and displayed on the 5 inch capacitive touch screen. The UI is a code block that is written using the python language. The code utilizes a python library named Dear PyGUI which is a powerful tool used for making interactive user interfaces. Through the use of the UI, the user will be able to add external devices or "nodes" to the homepage and from there, be able to monitor the statistics that the devices gather. These statistics include temperature, humidity, soil moisture, air pressure, and light level. These statistics are input into the UI through the use of a text file that is sent from the external nodes.

[3] Android App (Champion: Alex Feng):
Input: *grdn_nd_cmmnctn_andrd_pp_rf*
Output: *andrd_pp_intgrtn_cld_rf*
The Android app will primarily be used to monitor garden nodes' sensor data and set up integrations. The app will also serve as the entry point for adding new garden nodes directly through the app or via device sharing from another commissioner. Our app looks like a typical smart home app that can monitor sensor data with the ability to set up our proprietary integration software. The app communicates with devices via the Matter smart home protocol, which allows users to control and monitor devices over the local network and integrate with other existing smart home ecosystems like Alexa, HomeKit, and Google Home. The app aims to be functional and should be easy to use by at least nine out of ten users, as part of our engineering requirements.

[4] Integration / Cloud (Champion: Alex Feng):
Inputs: *andrd_pp_intgrtn_cld_rf, smrt_hb_lctrncs__intgrtn_cld_rf*
Output: *intgrtn_cld_smrt_hb_lctrncs__rf*
The system described is designed to remotely monitor and control garden nodes through a centralized cloud-based infrastructure. The smart hub collects sensor data from the garden nodes and sends it to the cloud. The cloud then checks this data for any measurements that exceed a threshold before notifying a user if an anomaly is detected. The data is then passed onto a processing pipeline that determines which actuators should be turned on or off. After processing, requests are sent back to the smart hub to turn on/off the relevant actuators based on the information in the message. The integrations reference the data set by the user from the Android app, which contains device-specific information and integration data in a cloud database. The integration data is used to determine which garden nodes will trigger which

actuators. Overall, this system provides a comprehensive solution for monitoring and automating gardening systems, allowing for an improved home lifestyle.

[5] Garden Node Communication (Champion: Aiden Olsen):
Input: *Slr_chrgr_grdn_nd_snsr_lctrncs_dcpwr*
Output: *Grdn_nd_cmmnctn_smrt_hb_rf*
The Garden Node Communication block plays a critical role in transmitting the air temperature and humidity, soil moisture, light intensity, and pressure information from the garden to the central hub. In order to achieve this, an ESP32 in order to achieve an internet connection, then, each node can be commissioned to a network using the app. Once commissioned, the data measured by the garden node will be available to the user in a well structured manner.

[6] Garden Node Sensor Electronics (Champion: Aiden Olsen):
Input: *Slr_chrgr_grdn_nd_snsr_lctrncs_dcpwr*
Output: *Grdn_nd_snsr_lctrncs_grdn_nd_cmmnctn_comm*
The Garden Node Sensor Electronics are implemented through the use of 4 total sensors, all using the I2C protocol. The first sensor, an SHT35, provides temperature and relative humidity data abiding by our engineering requirements. The next sensor, a VEML7700 provides light intensity value in the units of lux. In order to measure soil conductivity, a weatherproof soil capacitance probe has been selected to provide the user with soil information. The last of the sensors, an LPS25, supplies the garden node with information about the surrounding air pressure. Without this block, users wouldn't be able to collect data necessary for a healthy lawn, garden, etc.

[7] Solar Charger (Champion: Aiden Olsen):
Output: *Slr_chrgr_grdn_nd_snsr_lctrncs_dcpwr*
Since the garden node electronics will be battery powered, it is important that a solar panel is used. The garden node will make use of deep sleep in order to draw little current when data recording and transmission isn’t taking place. During this time, the sun will recharge the 10050mAh battery in order to ensure the device can last for extended periods of time. The solar charger block is essential for the garden node to be able to take the sensor measurements and transmit them to the hub.
[8] Hub Power (Champion: Peter Thompson):
Output: hb_pwr_smrt_hb_lctrncs__dcpwr
The Hub Power block sub-system is the sole power source for the raspberry Pi 4B and associated touch screen. The touch screen works directly with the Raspberry Pi and draws its power from it. This requires that the power supply can provide 5V at 3A, as per the Raspberry Pi’s maximum input, and provide a connection to a USB-C power input. To provide the necessary power the power supply will draw its power from a wall outlet and transform it to the required voltage.
The system will connect the wall power input to the Hub power system using a IEC C13 wire that will connect to a fused junction that will pass the power into the enclosure. The power supply converts from the 120V Nominal power from the wall input to 12V using a transformer. Then using a full wave bridge rectifier the AC input is transformed into a DC current. The 12V

input is then stepped down to 5V using a switching rectifier and is put through a low pass filter which was based off of the recommendation of the AP63356QZV7. The output of the system is a USB-C Female connector and will use a USB-C to USB-C to power the Raspberry Pi.

[9] Garden Enclosure (Champion: Peter Thompson):
Output: None
The Garden Enclosure Serves two purposes. The first is to provide a space protected from wind and rain that gives enough space to mount the PCB holding the circuitry used read, and send the environmental inputs, the battery and solar panel used to keep the system powered, and the lux sensor. Secondly the enclosure must provide two outlets, one for the soil moisture sensor, and another for the temperature and humidity sensor, because these sensors cannot function properly inside an enclosure.
The outside of the enclosure is a large waterproof enclosure bought from adafruit, it provides protection for the elements and provides five square inches of floor space beyond what is needed to give plenty of space for wired connections. To mount the PCB and batter the enclosure will contain a 3d printed mounting block that will provide the necessary connections. And to mount the lux sensor and the solar panel they will be epoxied to the clear roof of the enclosure to maximize sunlight. For the outlets the enclosure provides two cable glands that create a waterproof outlet for wires between 0.12 and 0.25 inches in diameter.

[10] Hub Enclosure (Champion: Peter Thompson):
Output: None
The Hub enclosure is a 163 X 125 X 110 mm box with rounded corners. The Base consists of the front which provides a stable mounting structure for the 5” LCD touch screen, the bottom that provides four standoffs for the Raspberry pi 4 and the left and right wall. The second piece consists of the top and it has two holes for 3m screws that hold the system together, and the back piece which has a 7mm square hole to allow the power cable through to the Raspberry Pi.

3.3 Interface Definitions

Name	Properties
otsd_smrt_hb_lctrncs__usrin	• **Timing:** At any random time for the length of a press • **Type:** Touch Screen • **Usability:** Needs to be usable by at least 4 out of 5 users.
otsd_hb_pwr_acpwr	• **Inominal:** .15A • **Ipeak:** 0.17A

	• **Vnominal:** 120Vac
otsd_grdn_nclsr_envin	• **Water:** a gallon per minute from all sides
otsd_smrt_hb_g_usrin	• **Other:** Ability to add devices to home screen • **Other:** On screen buttons that controls the interface • **Usability:** Usable by at least 4 out of 5 users
otsd_smrt_hb_g_other	• **Other:** Read values in from text file • **Other:** Save information to text file
grdn_nd_snsr_lctrncs_grdn_nd_cmmnctn_comm: this interface exists between the ESP32 microcontroller and the four I2C sensors. Standard speed I2C is used in our case.	• **Datarate:** 100kHz • **Protocol:** I2C • **Vnominal:** 3.3V
grdn_nd_snsr_lctrncs_grdn_nclsr_mech	• **Fasteners:** 4 3m screws • **Other:** 3.5 mm diameter exit cable gland • **Other:** 3.25mm diameter exit cable gland • **Other:** 5mm diameter exit Cable gland
slr_chrgr_grdn_nd_snsr_lctrncs_dcpwr: This interface's role is important for supplying power to the entire garden node from a LiPo battery charged up by a solar panel.	• **Inominal:** 5mA • **Ipeak:** 50mA • **Vmax:** 3.4V • **Vmin:** 3.2V

slr_chrgr_grdn_nclsr_mech	• **Fasteners:** 2 zip ties to hold down the 56.5x69x19 mm battery
smrt_hb_lctrncs__intgrtn_cld_rf: Smart hub sends device data to cloud	• **Messages:** Device id • **Messages:** Temperature, moisture, air pressure • **Protocol:** HTTP REST API
andrd_pp_intgrtn_cld_rf: Communication between Android app and integration cloud	• **Messages:** Device id and name • **Messages:** Integration configuration • **Protocol:** GraphQL API
grdn_nd_cmmnctn_andrd_pp_rf: The garden node to hub communication uses Matter in conjunction with a 2.4GHz WiFi connection to transmit sensor readings in a JSON data structure.	• **Messages:** Temperature, moisture, air pressure • **Other:** Matter device sharing • **Protocol:** Matter
intgrtn_cld_smrt_hb_lctrncs__rf: Integration notifies hub which actuators needs to be turned on/off	• **Messages:** Device id • **Messages:** Device state • **Protocol:** gRPC REST API
hb_pwr_smrt_hb_lctrncs__dcpwr	• **Inominal:** 3A • **Ipeak:** 3.2A • **Vmax:** 5.25V • **Vmin:** 4.75V • **Vnominal:** 5V
smrt_hb_g_otsd_usrout	• **Other:** Displays statistics gathered from connected nodes • **Type:** Display Output through a screen • **Usability:** Understandable by at least 4 out of 5 users

3.4 References and File Links

[1] I2C Bus, “Specification - I2C Bus”, 4th of April 2014 [Online]. Available: https://www.i2c-bus.org/specification/ (Accessed: January 20 2023).
[2] Texas Instruments, “BQ2407x Standalone 1-Cell 1.5-A Linear Battery Chargers with Power Path” October, 2021 [Online]. Available: https://www.ti.com/lit/ds/symlink/bq24074.pdf?ts=1674278918600&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ24074 (Accessed: January 20 2023).
[3] Tindie, “I2C Soil moisture sensor - Capacitive soil moisture sensor interfaced via I2C. Additionally provides ambient light and temperature readings. Open Source Hardware”, [Online]. Available: https://www.tindie.com/products/miceuz/i2c-soil-moisture-sensor/ (Accessed: February 7 2023).
[4] Vishay Semiconductors, “VEML7700 - High Accuracy Ambient Light Sensor With I2C Interface”, Rev. 1.6, 28-Apr-2022 [Online]. Available: https://www.vishay.com/docs/84286/veml7700.pdf (Accessed: January 28 2023).
[5] ST life.augmented, “MEMS pressure sensor: 260-1260 hPa absolute digital output barometer”, August 2016 Rev. 4 [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/datasheet/9a/4c/aa/72/1f/45/4e/24/DM00141379.pdf/files/DM00141379.pdf/jcr:content/translations/en.DM00141379.pdf (Accessed: January 26 2023).
[6] Sensiron - The Sensor Company, “Datasheet SHT3x-DIS Humidity and Temperature Sensor”, December 2022 - Version 7 [Online]. Available: https://sensirion.com/media/documents/213E6A3B/63A5A569/Datasheet_SHT3x_DIS.pdf (Accessed: February 1 2023).

3.5 Revision Table

3/09/23	Aiden Olsen: Added block information for the garden node sensor electronics, garden node communication, and solar charger

Section 4: Block Validations

Terminology used in section 4

- Garden node

    - The sensor that will be placed in lawns outside to measure moisture, temperature, and air pressure. It communicates with the app and the smart hub via the Matter protocol.
- Smart hub
    - It is a device that receives data from garden nodes over the local network via the Matter protocol. It then forwards all garden node data to the cloud for automation. Without this hub, the user can still monitor garden node data over the local network via the Matter protocol.
- Integrations / Cloud
    - Integrations (sometimes called the cloud) respond to garden node sensor data and automatically turn on/off actuators. It also keeps a record of all user data.
- Actuators
    - Actuators are devices that support the on/off capability. This means that these types of devices can be turned on or off. Devices include, but are not limited to, sprinklers, valves, and hoses.
- Matter
    - This is a new smart home protocol used for local network communication between the Android app, the smart hub, and the garden nodes. Matter-capable devices like the garden node can integrate with existing systems that support Matter, such as Amazon Alexa, Google Home, Apple HomeKit, and Samsung Smartthings. All devices supporting the Matter protocol will communicate over the same local WiFi network for our system.
- Commissioner
    - This is a term used to describe devices/apps capable of adding a new Matter device into its own network. For example, the Google Home app, our own app, and the smart hub act as commissioners because they can add Matter devices into their own networks.

4.1 Garden Node Communication

4.1.1 Description

The Garden Node Communication block plays a critical role in transmitting the air temperature and humidity, soil moisture, light intensity, and pressure information from the garden to the central hub. In order to achieve this, an ESP32 in order to achieve an internet connection, then, each node can be commissioned to a network using the app. Once commissioned, the data measured by the garden node will be available to the user in a well structured manner.

4.1.2 Design

In order to achieve this, we have decided to incorporate an ESP32-WROOM-D DEVKIT V4 microcontroller board into the block. The board provides the functionality required for a 2.4GHz WiFi connection as well as 4MB of flash in order to contain the large program sizes necessary for Matter connected devices. By using Espressif's SDK for Matter in conjunction with ESP-IDF, the board can be programmed to perform the necessary task of gathering sensor data and sending it when a Matter device requests its information.

4.1.3 General Validation

The inclusion of the ESP32-WROOM-D DEVKIT V4 microcontroller into this block is a significant component of the design since it provides the necessary functionality for a 2.4GHz WiFi connection and has 4MB of flash memory for the large Matter connected Devices. In order to ensure that the board is functioning in a correct manner, a strict validation procedure is in place to ensure the board is able to perform the tasks at hand.

The validation procedure for the board will include the ability to provide dummy data to an existing Matter connected device when it is requested. This is achieved by setting up a test environment where the phone app pulls data from a given populated Matter cluster. Additionally, the procedure includes the board's ability to be registered to a 2.4GHz WiFi network through the use of a Matter commissioning device of any kind. As an added measure of precaution, the dummy data stored on the ESP will be compared against the data that was received to ensure accuracy of the measurements.

4.1.4 Interface Validation

slr_chrgr_nd_snsr_lctrncs_dcpwr

Interface Property	Why is the interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Inominal: 5mA	The four I2C sensors draw on average 50mA when taking and transmitting measurements.	I have reviewed the data sheet for each sensor used and summed up each sensor's Nominal power draw.
Ipeak: 50mA	The bq24074 IC supports a	I followed the

	maximum load current of 1.5A, however, our max power draw will only ever total 100mA.	recommended layout and schematic in the datasheet which has the ability to achieve an output current of 1.5A, way over what is needed.
Vmax: 3.4V	There might be slight voltage regulation differences due to part tolerance.	I am using a fixed 3.3V regulator and supplying it with its supplementary manufacturer's recommendation layout and capacitor values.
Vmin: 3.2V	There might be slight voltage regulation differences in voltage due to part tolerance.	I am using a fixed 3.3V regulator and supplying it with its supplementary manufacturer's recommendation layout and capacitor values.

grdn_nd_cmmnctn_smrt_hb_rf

Interface Property	**Why is the interface this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Messages: JSON Data	JSON is a standard when it comes to sending data in a structured manner over an internet connection. Using JSON it is easy to parse data on the receiving end.	When a Matter device requests information from the garden node, a JSON packet is constructed and populated with the cluster ID's values.
Other: 2.4GHz Antenna	2.4GHz wifi uses longer transmission waves making it more suitable for traveling through walls. This is desirable since the garden node will most likely be placed outside.	According the the ESP32-WROOM-D module on the board's datasheet, the ESP32 is capable of of both 2.4 and 5GHz WiFi communication on the on-module antenna.

Protocol: Matter	Matter is an up and coming IoT protocol aimed at connecting all IoT devices making it compatible with all IoT ecosystems.	Espressif, the company responsible for the development of ESP chips has provided a SDK for Matter and the ESP-WROOM-D module is on the list of approved chips.

4.1.5 Verification Process

In order to verify the functionality of the JSON data, 2.4GHz antenna interfaces, and Matter protocol, the following can be followed:

1. Download Espressif's Matter Commissioning application and attempt to register the device to a network by using the device's unique QR code. The Espressif's Matter Commissioning application will show when a successful commissioning has been completed leading to a successful connection.
2. Set dummy data to the sensor reading's variables and query the data by using a chip-tool. When the data is queried, the cluster ids should be updated and sent in JSON form to the chip-tool command line.
3. As an added step, it is important to verify the strength and quality of the 2.4GHz signal by conducting a simple distance test.

4.1.6 References and File Links

[6] Espressif, "Espressif's SDK for Matter", 2022 - Master Branch [Online]. Available: https://docs.espressif.com/projects/esp-matter/en/main/esp32/index.html (Accessed: February 2 2023).
[6] Espressif, "ESP32-WROOM-32D Datasheet", 2023 - Version 2.4 [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf (Accessed: February 3 2023).

4.1.7 Revision Table

3/09/23	Aiden Olsen: Completed the entirety of section 4.1 from a

	previous block checkoff
3/09/23	Aiden Olsen: Added reference links

4.2 Garden Node Sensor Electronics

4.2.1 Description

The Garden Node Sensor Electronics block is a core component of the entire system. This block is necessary for collecting soil moisture, temperature, humidity, light intensity, and pressure data in a garden, lawn, or outdoor application. In order to achieve this, a total of four sensors are being used. First of all, in order to gather temperature and humidity data, an SHT35 humidity and temperature sensor from Sensirion is used. The SHT35 is capable of recording relative humidity within $\pm$1.5% and temperature within $\pm$ 0.1°C which meets our design requirements [6]. Pressure is recorded using an LPS25 pressure sensor from STMicroelectronics. This particular sensor has an absolute pressure range of 260 to 1260 hPa and has and error of $\pm$0.01hPa RMS [5]. The sensor responsible for measuring light is the VEML7700. Using the ALS module within the device, ambient, white, and lux values are able to be computed [4]. The last measurement needed is a soil capacitance sensor which is handled by the *I2C Soil moisture sensor* created by catnip electronics and can be found on Tindie.com [3].

4.2.2 Design

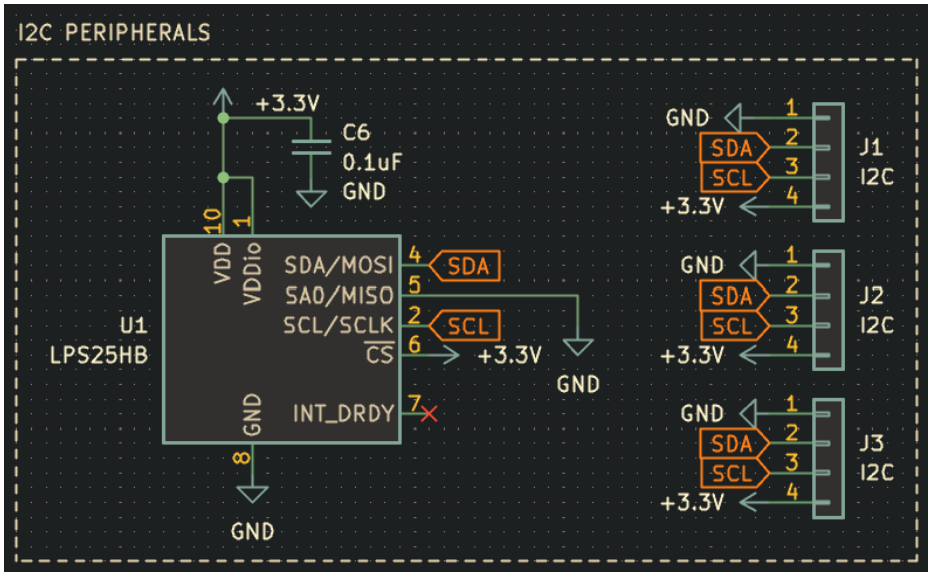


Fig. 4.2.2.1: Sensor Electronics Schematic*

*Figure 2 Notes:
+3.3V is the input to this block, **slr_chrgr_nd_snsr_lctrncs_dcpwr**
ISCL/SDA is the output to this block, **grdn_nd_snsr_lctrncs_grdn_wifi_comm**
IO21/IO22 lead to the SDA and SCL pins on our microcontroller (ESP32-WROOM-32D), respectively

4.2.3 General Validation

This block takes in power supplied by our solar charging block and uses the I2C communication protocol at 400kHz to supply sensor data to our microcontroller. All three sensors use the same I2C protocol which needs to have an individual address assigned to each sensor in order to read and write to a specific sensor. In order to validate this, extensive reading through the datasheets was required to ensure none of the devices I2C addresses conflict.

Our light sensor, temperature, humidity, and soil conductivity sensor are all external sensors connected using a JST PH connector, while our pressure sensor is built into our pcb layout. In order to achieve the I2C protocol at 3.3V, 4.7k pull-up resistors have been included on the SDA and SCL lines.

To test all these things I have added three test points to the PCB I have designed which is further detailed in section 4. However, a test point is included on the 3.3V rail, lipo input, and solar panel input terminal.

4.2.4 Interface Validation

slr_chrgr_nd_snsr_lctrncs_dcpwr

Interface Property	Why is the interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Inominal: 5mA	The four I2C sensors draw on average 50mA when taking and transmitting measurements.	I have reviewed the data sheet for each sensor used and summed up each sensor's Nominal power draw.
Ipeak: 50mA	The bq24074 IC supports a maximum load current of 1.5A, however, our max power draw will only ever total 100mA.	I followed the recommended layout and schematic in the datasheet which has the ability to achieve an output current of 1.5A, way over what is needed.
Vnominal: 3.3V	Using a fixed 3.3V LDO regulator on the output of the bq24074 we are able to achieve our garden-node sub-system voltage.	I am using a fixed 3.3V regulator and supplying it with its supplementary manufacturer’s recommendation layout and capacitor values.
Vmax: 3.4V	There might be slight voltage regulation differences due to part tolerance.	I am using a fixed 3.3V regulator and supplying it with its supplementary manufacturer’s recommendation layout and capacitor values.
Vmin: 3.2V	There might be slight voltage regulation differences in voltage due to part tolerance.	I am using a fixed 3.3V regulator and supplying it with its supplementary manufacturer’s recommendation layout

		and capacitor values.

grdn_nd_snsr_lctrncs_grdn_wifi_comm

Interface Property	Why is the interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Protocol: I2C	I2C is a serial communication allowing multiple devices to send data to a master device using addresses to distinguish themselves.	I have provided the necessary pull-up resistors on the SDA and SCL lines since they are required for the protocol to work. I also have verified that our microcontroller and external sensors use I2C.
Data Rate: 400kHz	This is considered as the standard I2C transmission rate and is plenty fast for what we are trying to achieve.	During PCB design I took into account bus capacitance as it can lead to issues as frequency increases.
Vnominal: 3.3V	The microcontroller and sensors are all compliant w/ 3.3V as the I2C line voltage.	All the data sheets allow for 3.3V as the I/O voltage during I2C transmission.

4.2.5 Verification Process

For my first interface, **slr_chrgr_nd_snsr_lctrncs_dcpwr**, I plan to test each component of the table separately. During the PCB design process I have provided test pads to make sure that the 3.3V rail is getting the proper voltage and supplying the necessary current. The bullet points below describe how I will carry out each test, working vertically down the related interface table.

- Inominal: 50mA
    - I have added a jumper from the 3.3V source to the sensors allowing me to

break the line and place a multimeter on the two pins. Then, I will power up the microcontroller and sensors making sure the current sits at around 50mA for 30 seconds using a lab power supply providing 3.3V and limited to 50mA.

- Ipeak 100mA
    - For this, I will have the sensors all take measurements using arduino libraries and record them to the microcontroller as fast as possible and watch on the multimeter to make sure the current does not surpass 100mA.
- Vnominal: 3.3V
    - To verify this interface property, I will set a multimeter to measure voltage and probe my 3.3V rail on the pcb to make sure that it remains at 3.3V while idle and taking measurements.

For the output interface, **grdn_nd_snsr_lctrncs_grdn_wifi_comm,** I plan to use an oscilloscope and multimeter to verify the necessary requirements. Similar to the verification plan for the solar chargers and sensor electronics dc power interface, I have provided test pads to ensure that the voltage, speed, and proper protocol is taking place.

- Protocol: I2C
    - Acknowledgements are sent whenever a byte of data is sent using I2C. Using an oscilloscope, I plan to decipher a transmitted message and decode an acknowledgement back from a I2C sensor. The acknowledgements should be as follows:
        - The master device generates a clock pulse on the SCL line, while holding the SDA line high.
        - The slave device receives the data on the SDA line and sends an ACK signal by pulling the SDA line low during the next clock pulse.
        - The master device detects the low level on the SDA line during the clock pulse and recognizes this as an ACK signal.
        - The master device generates another clock pulse to signal the end of the ACK.
        - The slave device releases the SDA line, allowing it to return to its high state.
- Data Rate: 400kHz
    - Using the same oscilloscope, I will ensure that the rise and fall times at 400kHz I2C will ensure proper data transmission from and to the microcontroller. In order to check this I will probe the SCL line and make sure that the inverse of the time it takes from a rising edge k to the next, k+1 is close to the designated data rate.
- Vnominal: 3.3V
    - Using an oscilloscope and multimeter I will make sure that during an I2C transmission the voltage swings close to the nominal voltage of 3.3V. Also, I will ensure that each pin coming from the JST connectors on the PCB that is responsible for VCC is close to the Vnominal voltage of 3.3V.
- All of the steps above involve using two oscilloscope probes, one for the SDA line, and another for the SCL line.

4.2.6 References and File Links

[1] I2C Bus, “Specification - I2C Bus”, 4th of April 2014 [Online]. Available: https://www.i2c-bus.org/specification/ (Accessed: January 20 2023).
[2] Texas Instruments, “BQ2407x Standalone 1-Cell 1.5-A Linear Battery Chargers with Power Path” October, 2021 [Online]. Available: https://www.ti.com/lit/ds/symlink/bq24074.pdf?ts=1674278918600&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ24074 (Accessed: January 20 2023).
[3] Tindie, “I2C Soil moisture sensor - Capacitive soil moisture sensor interfaced via I2C. Additionally provides ambient light and temperature readings. Open Source Hardware”, [Online]. Available: https://www.tindie.com/products/miceuz/i2c-soil-moisture-sensor/ (Accessed: February 7 2023).
[4] Vishay Semiconductors, “VEML7700 - High Accuracy Ambient Light Sensor With I2C
[5] ST life.augmented, “MEMS pressure sensor: 260-1260 hPa absolute digital output barometer”, August 2016 Rev. 4 [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/datasheet/9a/4c/aa/72/1f/45/4e/24/DM00141379.pdf/files/DM00141379.pdf/jcr:content/translations/en.DM00141379.pdf (Accessed: January 26 2023).
[6] Sensiron - The Sensor Company, “Datasheet SHT3x-DIS Humidity and Temperature Sensor”, December 2022 - Version 7 [Online]. Available: https://sensirion.com/media/documents/213E6A3B/63A5A569/Datasheet_SHT3x_DIS.pdf (Accessed: February 1 2023).

4.2.7 Revision Table

3/09/23	Aiden Olsen: Completed section 4.2 with previous validation paper

4.3 Solar Charger

4.3.1 Description

The solar charger block, as simple as it is, is an essential block to the garden node’s entire system. Without the solar panel, the longevity of the garden node sub-system would be left up to the capacity of the battery included. By adding a solar panel, the system will be able to recharge the battery using a sustainable form of energy, further prolonging the battery life of the device.

4.3.2 Design

In order to charge the battery on the garden node, A solar panel has been selected that can output a nominal 6V at 330 mA peak current via DC barrel jack connector. The panel is constructed using ETFE (Ethylene Tetrafluoroethylene) technologies, making it extremely durable and resistant to environmental elements that the garden node may face. These are a superior upgrade to PET or laminate solar panel construction. ETFE panels can easily stand up to typical outdoor use, including being dropped and leaned on. The panel is over-spec'd in order to ensure more than enough energy is being allow for prolonged life more than the battery could provide.

4.3.3 General Validation

In order to validate the solar panel block will be able to supply the energy required to the garden node, the panel must be tested for durability and power supply. In order to tell if the panel will live up to the expectation, two separate tests will be conducted. Using a variable load, the solar panels power generation can be measured. By putting the solar panel in the window, both the voltage and current produced will be measured. In addition to the power tests, the solar panel must also withstand a wet outdoor environment. To test this, the panel will be fully submerged in water for 30 seconds and it must continue to work afterwards.

4.3.4 Interface Validation

slr_chrgr_grdn_nd_snsr_lctrncs_dcpwr : Output

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Inominal: 5mA	The garden node consumes a peak of 5mA when in sleep mode, so the solar charging circuitry must be able to supply a minimum of 5mA to ensure either a net positive or net zero gain in battery charge.	The garden node electronics draw a nominal of 5mA when in sleep mode. This has been measured using a lab power supply.
Ipeak: 50mA	When the ESP32 is connected to the internet and recording sensor measurements, the power draw can	The garden node draws up to 50mA peak when transmitting over wifi. The BQ24074 IC is more than

	reach up to 50mA so the solar circuitry has to be able to supply this.	capable of supplying that current [1].
Vmax: 3.4V	The system runs on 3.3V and a ripple of + or - 0.1V is ideal to ensure communications are successful between the wifi and I2C transmissions.	An AZ1117H fixed 3.3V linear voltage regulator is used [2]. The output voltage table shows a max of 3.365V which is under Vmax.
Vmin: 3.2V	The system runs on 3.3V and a ripple of + or - 0.1V is ideal to ensure communications are successful between the wifi and I2C transmissions.	An AZ1117H fixed 3.3V linear voltage regulator is used [2]. The output voltage table shows a min of 3.235V which is above Vmin.

4.3.5 Verification Process

By following the step provided below, the solar charging block can be ensured to provide the power necessary for the garden node:

1. Verify that the garden node consumes a peak of 5mA when in sleep mode, as stated in the documentation by using a lab bench power supply.
2. Confirm that the solar charging circuitry is able to supply a minimum of 5mA to ensure a net positive or net zero gain in battery charge with an adjustable load.
3. Verify that the garden node electronics draw a nominal of 5mA when in sleep mode, as measured using a lab power supply.
4. Confirm that the power draw can reach up to 50mA when the ESP32 is connected to the internet and recording sensor measurements, as stated in the documentation using a lab power supply.
5. Verify that the solar circuitry is capable of supplying up to 50mA peak when the garden node is transmitting over wifi, as stated in the documentation by using a variable load.
6. Confirm that the system runs on 3.3V and that a ripple of + or - 0.1V is ideal to ensure successful communications between the wifi and I2C transmissions.
7. Confirm that the output voltage from the voltage regulator shows a maximum of 3.365V, which is below the Vmax interface definition using a multimeter.
8. Confirm that the output voltage from the voltage regulator shows a minimum of 3.235V, which is above the Vmin interface definition using a multimeter.

4.3.6 References

[1] “BQ24074 Datasheet” Texas Instruments, Sept 2008. [Online]. Available: https://www.ti.com/product/BQ24074?utm_source=google&utm_medium=cpc&utm_campaign=app-null-null-GPN_EN-cpc-pf-google-wwe&utm_content=BQ24074&ds_k=BQ24074&DCM=yes&gclid=Cj0KCQiAjbagBhD3ARIsANRrqEt87vJwmOLsPy0aATcPMTuF__U8ZeynNWgozbwiNHgIFfcrL9eXDacaAjVvEALw_wcB&gclsrc=aw.ds (Accessed: February 21, 2023)
[2] “AZ1117 Datasheet” Diodes Incorporated, Jan 2019. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Diodes%20PDFs/AZ1117_Rev5.3_Jan2019_DS.pdf (Accessed: January 12, 2023)

4.3.7 Revision Table

3/09/23	Aiden Olsen: Wrote validation for solar charger block
3/11/23	Aiden Olsen: fixed grammatical errors

4.4 Android app

4.4.1. Description

The Android app will primarily be used to monitor garden nodes’ sensor data and set up integrations. The app will also serve as the entry point for adding new garden nodes directly through the app or via device sharing from another commissioner. Our app looks like a typical smart home app that can monitor sensor data with the ability to set up our proprietary integration software. The app communicates with devices via the Matter smart home protocol, which allows users to control and monitor devices over the local network and integrate with other existing smart home ecosystems like Alexa, HomeKit, and Google Home. The app aims to be functional and should be easy to use by at least nine out of ten users, as part of our engineering requirements.

4.4.2. Design

The user will be able to monitor the sensor data of their garden nodes inside our app and use these nodes to control other devices, such as smart irrigation controllers, using integrations. The app will allow users to add new Matter capable devices directly through our app or by sharing devices via the Google Home app. Users can then view the sensor data of their garden nodes within our app. They can also create an integration that

automatically turns on/off actuators based on data received from the garden nodes. Suppose a garden node's temperature, moisture, or air pressure values exceed a threshold (e.g., too much moisture). In that case, the integration will automatically turn on/off the actuators configured by the user.

Black box diagram

This diagram shows a high-level overview of the interfaces that interact with the Android app.

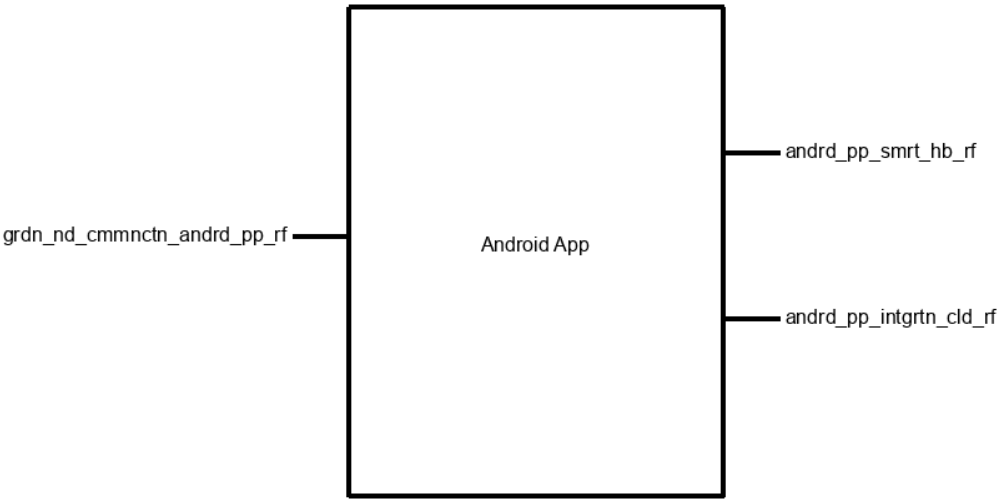


Fig. 4.4.2.1: Black box diagram

- grdn_nd_cmmnctn_andrd_pp_rf
  - Bidirectional communication
  - Communication between the garden node and the Android app
- andrd_pp_smrt_hb_rf
  - Android app provides pairing code that smart hub can use
  - Communication between Android app and smart hub
- andrd_pp_intgrtn_cld_rf
  - Bidirectional communication
  - Communication between Android app and integration cloud

Block diagram

This diagram below describes in more detail the interfaces that interact with the Android app.

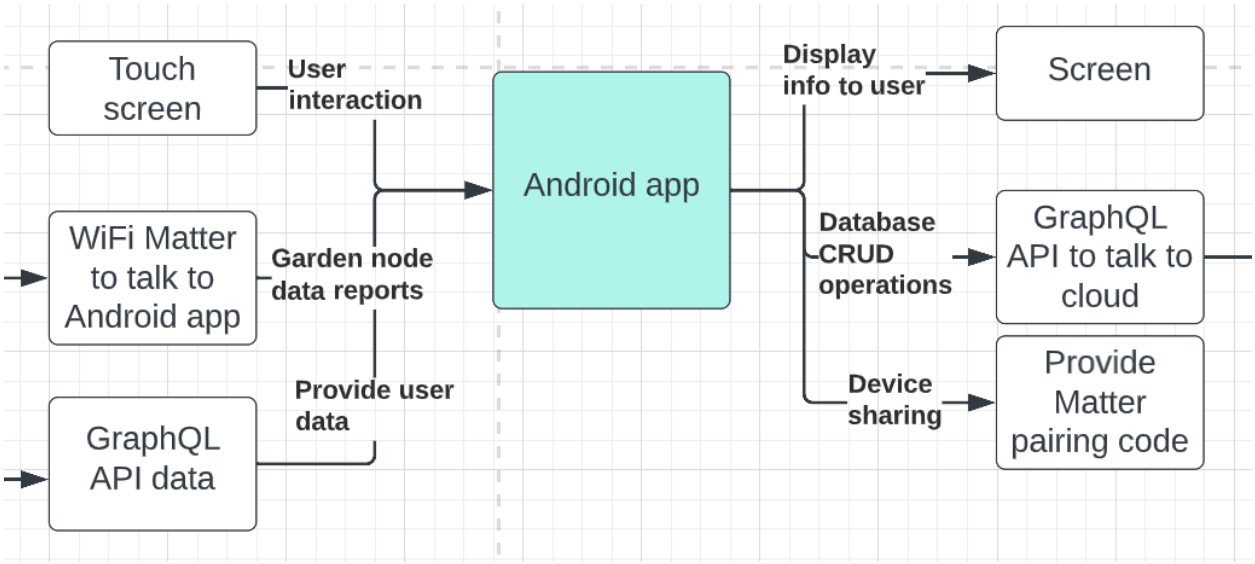


Fig. 4.4.2.2: Block diagram

Phone touchscreen

- The user will be able to configure settings in the app, such as adding a new device, viewing device data, and setting up integrations from their Android phone touchscreen. Data received from garden nodes will be presented back to the user on the screen in text format.

Garden node to Android app (grdn_nd_cmmnctn_andrd_pp_rf)

- Garden nodes will send data to the Android app directly (if in range) over the Matter protocol in a local WiFi network. The app can request and send data (e.g., name) to the garden nodes. The app will also be responsible for setting up the network credentials and onboarding new devices onto its Matter network.

Smart hub to Android app (andrd_pp_smrt_hb_rf)

- There is only one instance in which the Android app will communicate with the smart hub, which is to share Matter devices to the hub. The user can select to share a particular device with the hub, prompting the user to enter the pairing code on the hub to complete the setup. At the same time, the app will also tell the device to reopen its commissioning window so the smart hub can communicate with it for initial setup.

Cloud to Android app (andrd_pp_intgrtn_cld_rf)

- The app will be able to store device-specific information and integration data in a database in the cloud. This data will tell the integrations which garden nodes will trigger which actuators.

<u>Commissioning flow</u>

The flowcharts below describe how a user can add a new Matter device to the Android app's Matter network.

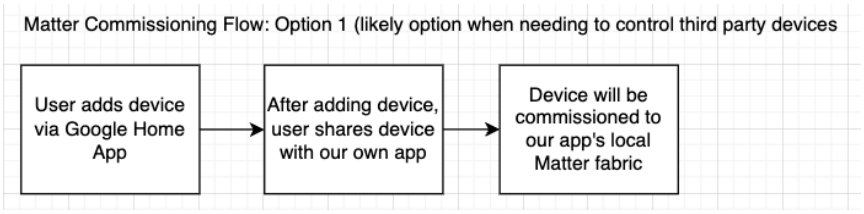


Fig. 4.4.2.3: Commissioning flow option 1

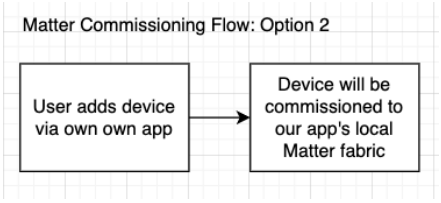


Fig. 4.4.2.4: Commissioning flow option 2

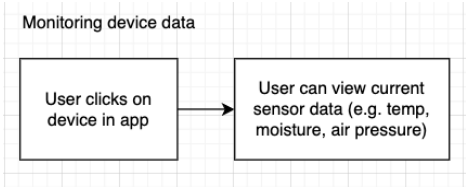


Fig. 4.4.2.5: Commissioning flow option 3

Integration setup flow

This diagram shows the process of adding a new integration. Once added, the cloud will take care of automating it. A separate block handles the automation part.

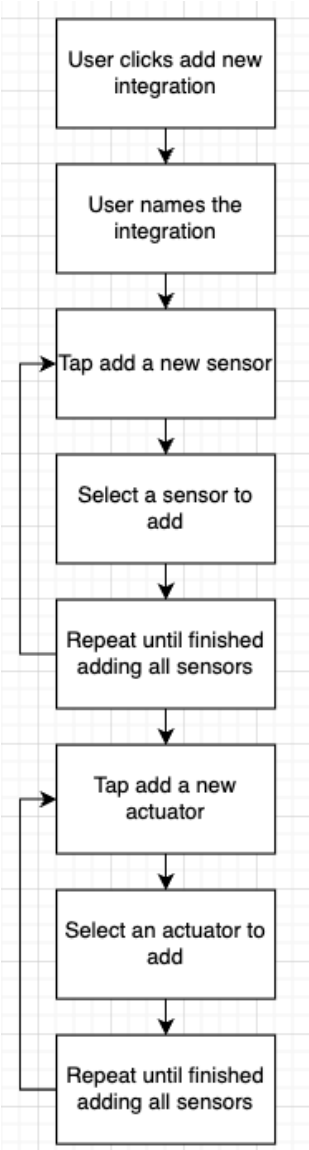


Fig. 4.4.2.6: Integration flow

App design

The image below shows the layout of all the pages I will create. The top row shows all the pages users will see to add, view, edit, and delete a device. The bottom row shows all the pages users will interact with to add, edit, view, and delete an integration. In an integration, a user can select which garden nodes will act as the input and which devices will act as the actuators. The cloud will handle the automation of the actuators.

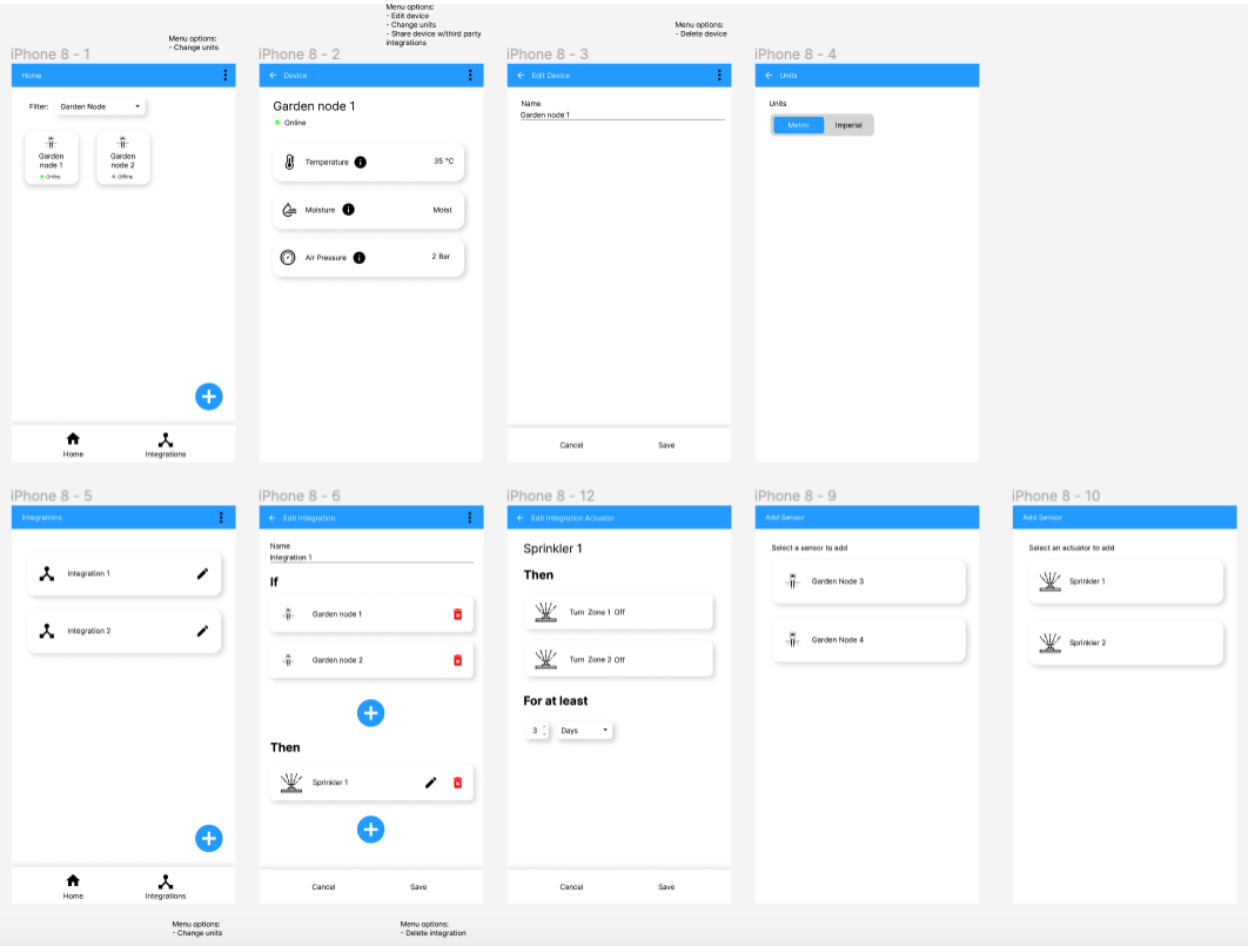


Fig. 4.4.2.7: App design

4.4.3. General Validation

The Android app is a critical part of our design because it is the central point for users to interact with our system. Users can add, edit, control, and delete devices through our app. It also extends upon the Google Home app by allowing for the creation of smart integrations.

The app will be developed using Android Studio, which is free and utilizes the Google Home Matter SDK [1], which is also free to use. Changes in the app should not require significant modification to the garden node program or integration cloud code. It will be designed agnostic and should generally work with other parts of our system without requiring changes to other systems. It communicates with the garden nodes via the Matter smart home protocol. It will communicate to the integration cloud via GraphQL API requests.

I have experience with Android development in Kotlin and have already started creating the basic app that can add, edit, control, and delete a Matter device. It also supports device sharing between our app and the Google Home app with an ESP32

WiFi board (the microcontroller used for our garden node). I have also tested that the app can share a Matter device with a smart hub via a pairing code. The ESP32 can then pair with the smart hub over its local Matter network, and the hub can query sensor data from the device. The app can also save user data onto a database in the cloud.

The app should be finished by the end of this term, with the exceptions of bugs and other minor changes we may make should we change ideas with the functionality of our app.

4.4.4. Interface Validation

andrd_pp_smrt_hb_rf

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Other: Matter device sharing	The Android app needs to provide a pairing code when the user selects to share a Matter device. The user can then enter this code into the hub to add the same Matter device to its network.	The hub will be able to connect to a Matter device using a pairing code using the following command that was provided by the Matter documentation on GitHub. It takes in a pairing code as the payload to add the device. ./chip-tool pairing code <node_id> <payload> [2]
Other: Setup in under 5 minutes	Pairing devices to new hubs is supposed to be fast, and shouldn't take a long timer. Otherwise setup of the device would be very frustrating and time consuming.	The Matter protocol is expected to be easy for the user to set up as part of the requirement. The goal behind Matter should be easy frustration free setup. This means pairing devices to new hubs should be quick.

Protocol: Matter	Matter is the smart home protocol used for communicating with the garden nodes.	The smart home hub, garden nodes, and Android app will communicate via the Matter protocol to be compatible with the Android app.

Table 1: andrd_pp_smrt_hb_rf interface descriptions

grdn_nd_cmmnctn_andrd_pp_rf

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Messages: Temperature, moisture, air pressure.	Three core pieces of sensor data need to be transmitted from the garden node to the Android app. The data must include temperature, moisture, and air pressure.	Although the Matter protocol does not support the transmission of moisture or air pressure directly, I can send the data using any data type (i.e., cluster [3]) that I choose. It so happens that there are clusters called temperature, humidity, and pressure that will suffice. The Google Home SDK supports the reception of data on these clusters.
Other: Matter device sharing	The garden node needs to make itself available for commissioning for other commissioners to add the device to a new Matter network.	Matter device sharing is also known as multi-admin [4], a central concept in the Matter protocol. It specifies that Matter devices can be linked and controlled by multiple commissioners. The Google Home SDK supports the ability for multi-admin

		commissioning.
Protocol: Matter	Matter is the smart home protocol for communicating with the garden nodes.	The smart home hub, garden nodes, and Android app will communicate via the Matter protocol to be compatible with the Android app.

Table 2: grdn_nd_cmmnctn_andrd_pp_rf interface descriptions

andrd_pp_intgrtn_cld_rf

Interface Property	**Why is this interface property this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Messages: Integration configuration	The user needs to be able to configure integrations. This means they can add a new integration, select garden nodes as inputs, and select actuators as outputs.	The GraphQL interface to the cloud will allow for CRUD (create, read, update, delete) operations for the integrations. The Android app design also showcases the layout of the pages the user will see in order to configure an integration.
Messages: Device id and name	The user needs to store the device id and name in the cloud database so that the integrations can reference them. It also serves as a record of the user's device information.	When the user adds a new device, it will also add it as a new record in the database via the GraphQL API protocol. The Android app will support the CRUD operation of these devices via the pages shown in the design.
Protocol: GraphQL API	This protocol is necessary for performing database operations and keeping track of user	The Android Retrofit library will be used to send GraphQL API requests to the cloud, and this library

	transactions.	ensures that all standard procedures are already in place, such as error handling and back-off retries.

Table 3: andrd_pp_intgrtn_cld_rf interface descriptions

4.4.5. Verification Plan

Regarding the andrd_pp_smrt_hb_rf interface, I will utilize the Google Home Matter SDK, which already supports the Matter protocol and provides sample code to enable device sharing. This means it can provide a Matter-compatible pairing code that the smart hub can use for commissioning new Matter devices.

Interfaces proven by verification

- Protocol: Matter
- Other: Matter device sharing

1. The user will open the Android app
2. The user will select a device already commissioned to the Android app
3. The user will tap on the Share Device menu option
4. The Android app will prompt the user with the method of sharing. The user will select the pairing code. A pairing code will be displayed on the screen.
5. The user can enter this code into the hub to commission the Matter device onto its network.

Regarding the grdn_nd_cmmnctn_andrd_pp_rf interface, the Android app will use the Google Home Matter SDK to communicate with the garden nodes. It handles the commissioning, device sharing, and messaging capabilities while conforming to the Matter specification.

Interfaces proven by verification

- Protocol: Matter
- Messages: Temperature, moisture, air pressure

1. The user will open the Android app
2. The user will select a device already commissioned to the Android app
3. The user will see temperature, moisture, and air pressure data displayed on the screen
4. I will open up the Android Studio logs to show that the app is polling the garden node for data and that the device responds with data.

Interfaces proven by verification

- Protocol: Matter

- Other: Matter device sharing

1. The user will open the Android app
2. The user will select a device already commissioned to the Android app
3. The user will tap on the share device menu option
4. I will open up the device logs and show that it received the notification to open its commissioning window.

Regarding the andrd_pp_intgrtn_cld_rf interface, I will send GraphQL API requests to the cloud for user database interactions. All messages sent or received will conform to the JSON message format.

Interfaces proven by verification
- Protocol: GraphQL API
- Messages: Integration configuration

1. The user opens app
2. The user selects integration tab
3. The user selects an existing integration or adds a new integration
4. The user modifies the name, garden nodes selected as inputs, and actuators selected as outputs
5. I will show that the database operations are saved in the database in the cloud
6. I will show that the Android code utilizes the GraphQL protocol to communicate with the cloud.

Interfaces proven by verification
- Protocol: GraphQL API
- Messages: Device id and name

1. The user opens app
2. The user selects a device from the list
3. The user selects the edit menu option
4. The user changes the name and saves
5. I'll show that the database operation was saved in the database in the cloud
6. I'll show that the Android code utilizes the GraphQL protocol to communicate with the cloud.

4.4.6. References and File Links

[1] "Google Home Sample App for Matter." Google, Oct 10, 2022. [Online]. Available: https://developers.home.google.com/samples/matter-app (Accessed: Jan 18, 2023).
[2] "connectedhomeip." Github. [Online]. Available: https://github.com/project-chip/connectedhomeip/tree/master/examples/chip-tool (Accessed: Feb 1, 2023).

[3] Sovani, Kedar, “Matter: Clusters, Attributes, Commands.” Medium, Dec 1, 2021. [Online]. Available: https://blog.espressif.com/matter-clusters-attributes-commands-82b8ec1640a0 (Accessed: Feb 1, 2023).
[4] Sovani, Kedar, “Matter: Multi-Admin, Identifiers, and Fabrics.” Medium, Jan 19, 2022. [Online]. Available: https://blog.espressif.com/matter-multi-admin-identifiers-and-fabrics-a291371af365 (Accessed: Feb 1, 2023).

4.4.7. Revision Table

1/19/23	Alex Feng: Section created
2/1/23	Alex Feng: Added more details, added more diagrams, updated interfaces and properties, added more references, and updated verification plan
2/19/23	Alex Feng: Added third property to andrd_pp_smrt_hb_rf interface

4.5 Integration / Cloud

4.5.1. Description

The system described is designed to remotely monitor and control garden nodes through a centralized cloud-based infrastructure. The smart hub collects sensor data from the garden nodes and sends it to the cloud. The cloud then checks this data for any measurements that exceed a threshold before notifying a user if an anomaly is detected. The data is then passed onto a processing pipeline that determines which actuators should be turned on or off. After processing, requests are sent back to the smart hub to turn on/off the relevant actuators based on the information in the message. The integrations reference the data set by the user from the Android app, which contains device-specific information and integration data in a cloud database. The integration data is used to determine which garden nodes will trigger which actuators. Overall, this system provides a comprehensive solution for monitoring and automating gardening systems, allowing for an improved home lifestyle.

4.5.2. Design

The integration cloud is used to automatically turn actuators such as sprinklers, hoses, and valves on or off depending on the measurements received by the garden nodes. The architecture of this is described in the following images.

Black box diagram

This diagram shows a high-level overview of the interfaces that interact with the integration / cloud.

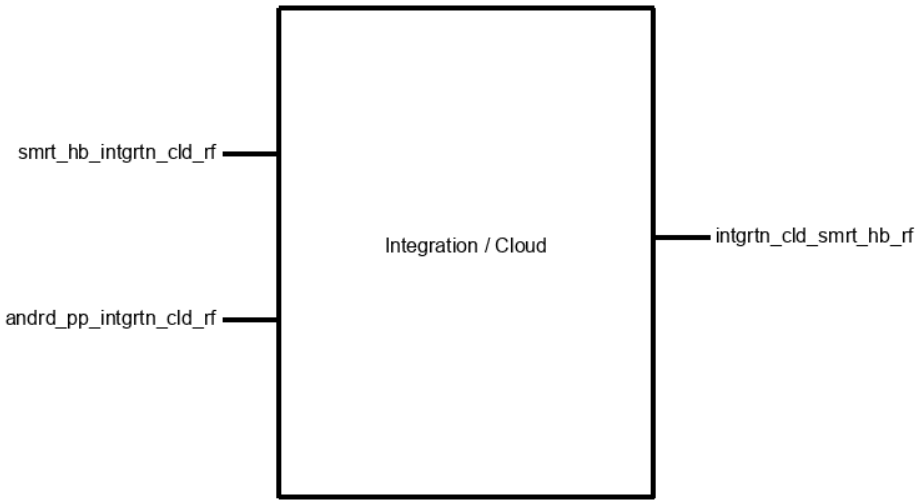


Fig. 4.5.2.1: Black box diagram

- smrt_hb_lctrncs__intgrtn_cld_rf
  - Input
  - Smart hub sends device data to cloud
- intgrtn_cld_smrt_hb_lctrncs__rf
  - Output
  - Integration notifies hub which actuators needs to be turned on/off
- andrd_pp_intgrtn_cld_rf
  - Bidirectional communication
  - Communication between Android app and integration cloud

Block diagram

This diagram below describes in more detail the interfaces that interact with the cloud.

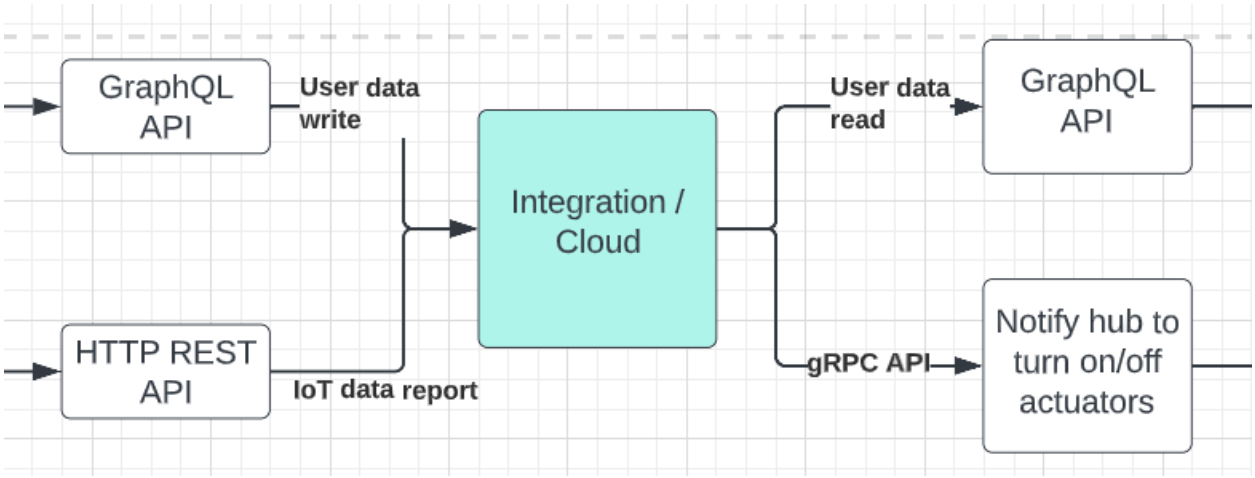


Fig. 4.5.2.2: Block diagram

Smart hub to Cloud (smrt_hb_lctrncs__intgrtn_cld_rf)

- The smart hub will periodically poll garden nodes for their sensor data and then forward this data to the cloud via a HTTP REST API interface. The cloud will then run a check on this data to see if any of the values exceed a threshold value.

Integration to smart hub (intgrtn_cld_smrt_hb_lctrncs__rf)

- If an anomaly is detected from the sensor data, the integration will send out a notification on its gRPC endpoint. The hub is always listening to this gRPC endpoint and will receive the alert. The message contained in the alert informs the hub which actuators need to be turned on/off.

Cloud to Android app (andrd_pp_intgrtn_cld_rf)

- The app will be able to store device-specific information and integration data in a database in the cloud. This data will tell the integrations which garden nodes will trigger which actuators.

Integration / Cloud backend setup

Below is the database structure of the Android app and virtual device cloud (i.e. integrations). It was built using AWS Amplify, AppSync GraphQL, and DynamoDB. The Android app and virtual device cloud will both be able to interact with the database via a GraphQL endpoint.

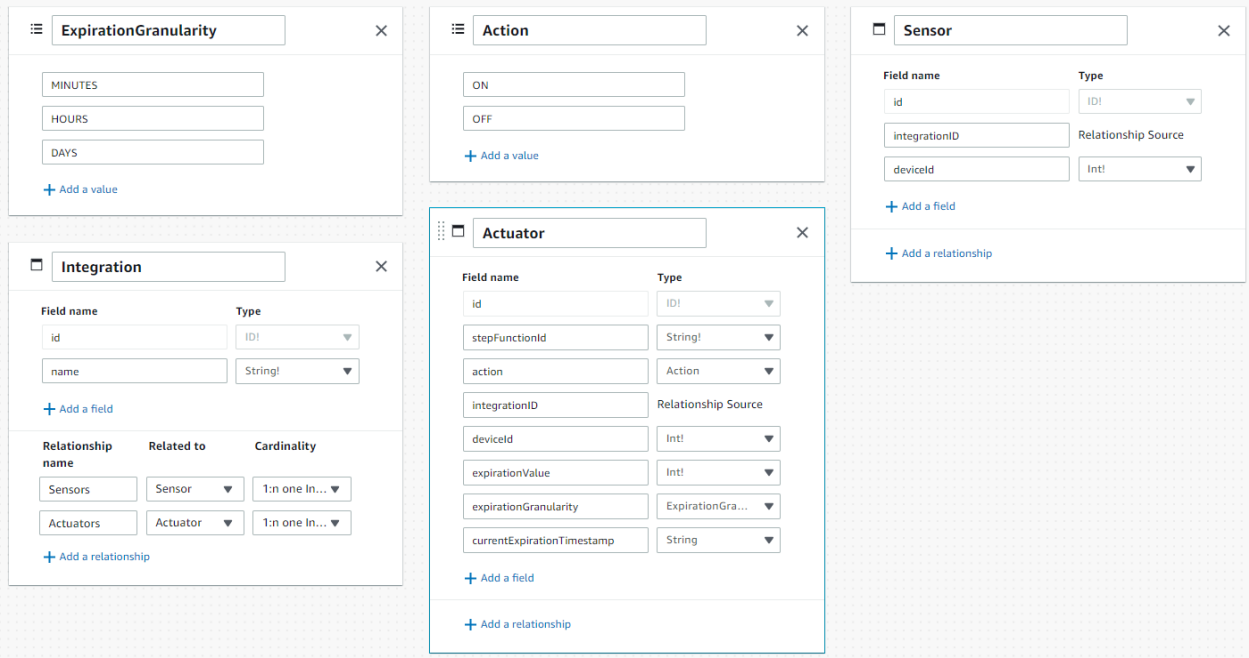


Fig. 4.5.2.3: Database structure

Next, the smart hub will query measurements from the garden nodes periodically via the Matter protocol. It'll then forward all data to a REST API endpoint. The API is handled by an ASP.NET application that stores the data in the appropriate "device shadow" (mini IoT database for each device). Anytime the device shadow is updated, a couple of IoT rules evaluate where to forward the data to, described in the next steps.

Note: The reason for using an HTTP REST endpoint instead of having devices directly interact with their own IoT device shadow over MQTT was to simplify the process of data reporting on the smart hub side (i.e. less coding).

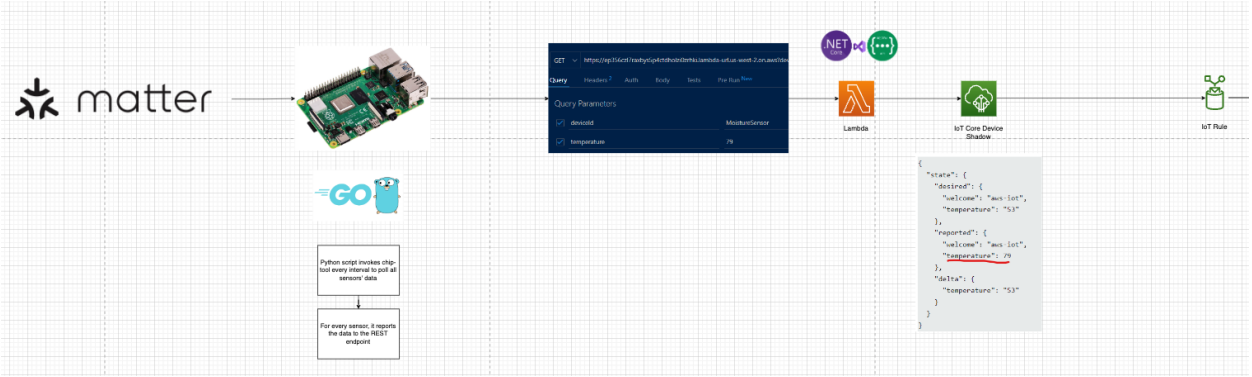


Fig. 4.5.2.4: Data reporting

The first IoT rule is used only for monitoring. It forwards the data to a Timestream database to keep a historical record of all data in a format that can be graphed. This data is then fed into Grafana to be graphed out in real time.

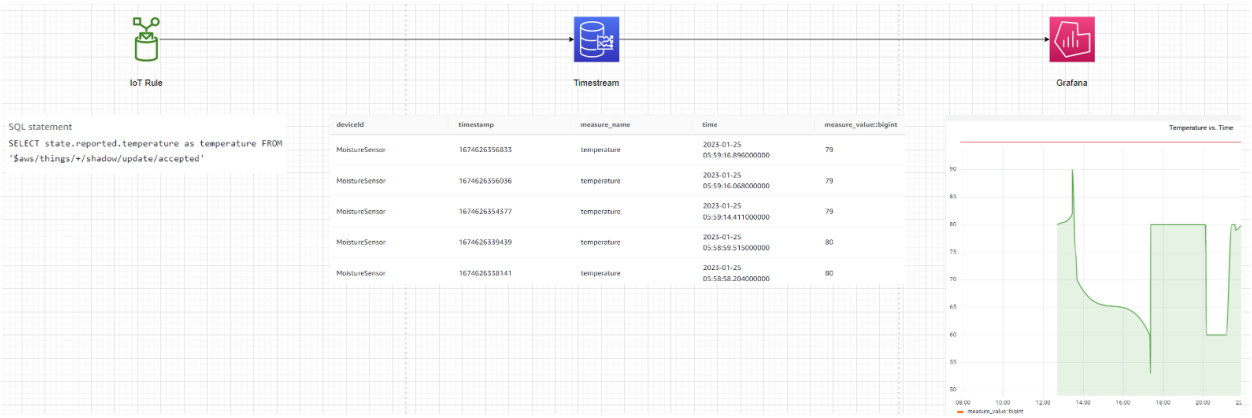


Fig. 4.5.2.5: Real time monitoring

The second IoT rule takes in the data and passes it to IoT Events. This is the bulk of the integration's core functionality. It looks pretty complicated, but that's mainly because all the code was spread out into a microservices architecture. Here are the detailed steps of how it works:

1. Each garden node's measurements are evaluated by an IoT event detector model. Any time the device state changes (e.g. Normal to Hot or Hot to normal), it triggers step 2.
2. This Lambda function is the IoT event handler and responds to any state changes. If the state returns to Normal and the actuator expiration timer has expired, then it'll send a request to turn the actuator back on at step 5. If the state is not Normal, then it'll submit a scheduler job request in step 3 and a request to turn the actuator off. In either case, it'll also send a push notification to the user's phone.
3. This Docker container is in charge of starting timers for each actuator that has been turned off by the integration. The timer will run until the user's desired minimum off time (e.g. 1 day). When the timer expires, it'll submit a request to reset the actuator in step 4.
4. This Lambda function will first check if the garden node is back in its Normal state. If not, it will make a request to the scheduler in step 3 again with the user's desired minimum off time again. If the garden node is back in its Normal state, then it will make a request to turn the actuator back on at step 5.
5. This Docker container runs a gRPC server to communicate with all smart hubs connected to it. Every time a request to turn the actuator on or off is received, it forwards all the messages out to all gRPC clients in fan-out style.

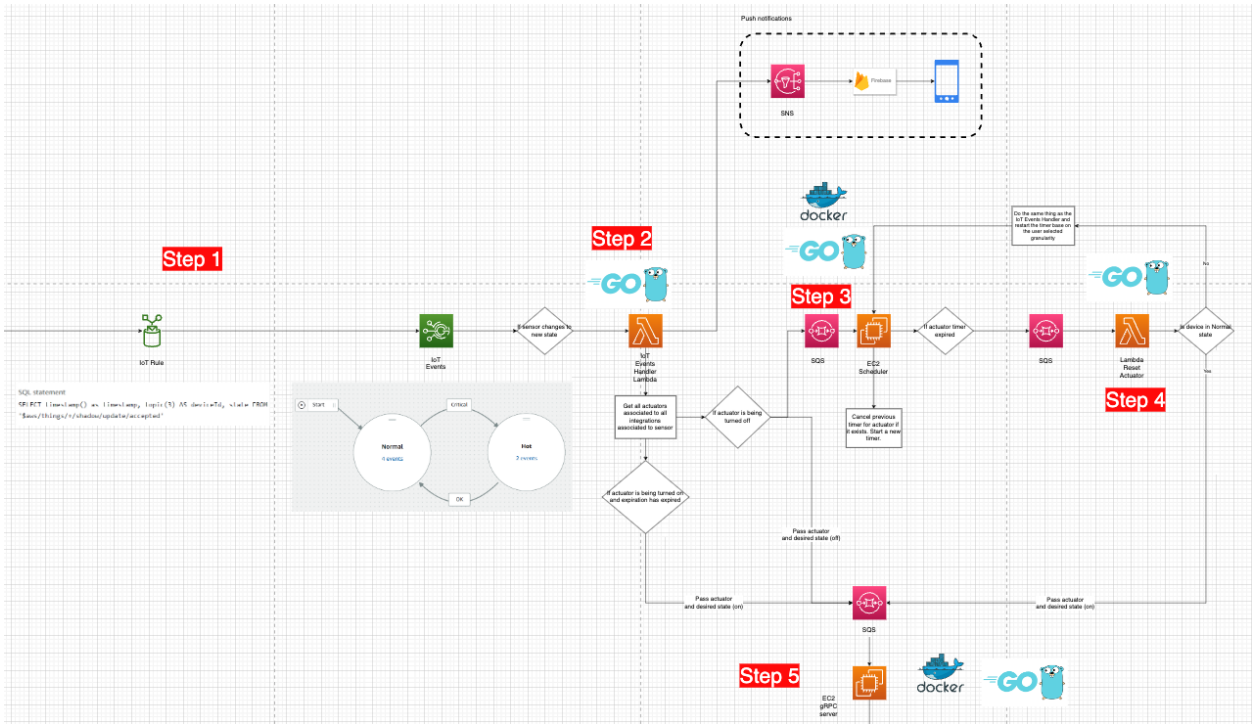


Fig. 4.5.2.6: Integration automation

The final step of the architecture is where the smart hub runs a gRPC client that listens to the gRPC server. Every time a new message is received, it runs a bash script that executes the Matter chip-tool to update an actuator's on/off state.

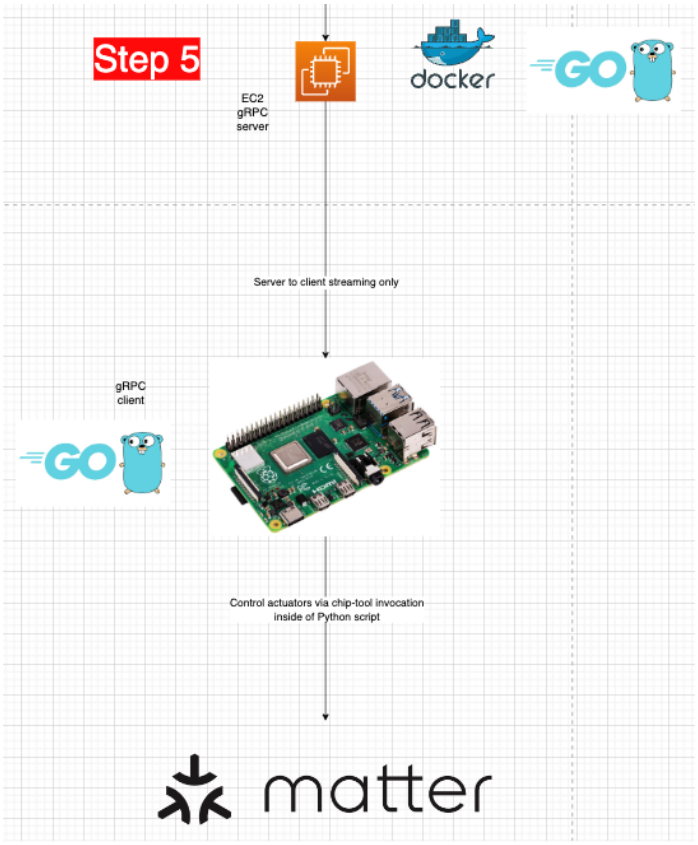


Fig. 4.5.2.7: Hub actuator notifications

4.5.3. General Validation

This block has two interfaces for communicating with the system wirelessly to the hub. The first interface is when it receives HTTP REST API data reports from the hub. The second interface is when it publishes gRPC API [1] messages to the hub about which actuators to turn on or off. This meets our system requirement of all processes communicating wirelessly. The Android app also communicates wirelessly to the cloud via a GraphQL API interface to store data. All APIs are proven methods of wireless communication transfer protocols and are reliable. Though there have been many comparisons between them, regarding which one is faster and which API to use in each scenario, the way I've structured my block, all APIs can be used in the architecture. The REST API is best suited as a simple interface that the client can publish individual messages to. The GraphQL API is best suited when data needs to be retrieved from multiple sources in a single fast query. The gRPC API is best suited for enforcing type conformity between two separate processes while allowing for real time message streaming [2].

To implement the system, I've chosen to use Amazon Web Services to host everything in the cloud so that the system can work outside of the local network. Each pragmatic microservices will be deployed directly to either a Lambda function or a

Docker container running inside of an EC2 instance. I've worked with all services listed in the design section and have already begun implementing them. I'm confident in my ability to complete the block by our project's required deadline.

Every microservice in the architecture is connected to each other whether they be events, queues, or API calls. This ensures that there is a continuous flow of messages to reach the destination. The cloud runs independently of the hub and is fault tolerant, meaning that it can still operate even if the hub is powered off or loses a network connection. For example, if the hub is unable to reach the internet, the gRPC server will wait to send out a message until a client is able to consume that message. I've chosen to deploy SQS buffers between different programs written in GoLang so as to enable backoff retries should any errors occur, and to serve as a temporary storage mechanism if there are too many messages begging publishes that the consumer is unable to process. It can also aggregate incoming messages for consumers to process in batches [3].

4.5.4. Interface Validation

smrt_hb_lctrncs__intgrtn_cld_rf

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
Messages: Device id	The hub needs to tell the cloud which device the data being reported belongs to so that it can be saved to the correct device IoT database.	I am using AWS Lambda to run custom code to store the device information into AWS IoT. AWS IoT contains a catalog of all devices with a thingName corresponding to the device id. Provided a valid device id, the reported data will be stored in the appropriate place.
Messages: Temperature, moisture, air pressure	The hub needs to report to the cloud the sensor data that had just been polled from a garden node. These values should	After AWS Lambda stores the sensor data into the respective device IoT database, it'll automatically trigger a function that

	correspond to the device id that is also being passed along.	evaluates the data to determine whether or not to turn on/off any actuators for any integrations that associated to a particular garden node. This is done with a service called IoT Events.
Protocol: HTTP REST API	The smart hub will be posting all data to an HTTP REST API endpoint.	I am using AWS Lambda to send data from its query parameters to AWS IoT. The Lambda function provides a function url which serves as the HTTP REST API endpoint.

Table 1: smrt_hb_lctrncs__intgrtn_cld_rf interface descriptions

intgrtn_cld_smrt_hb_lctrncs__rf

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
Messages: Device id	The integration needs to tell the smart hub which device that needs to be turned on or off. This device id corresponds to the Matter device thingName attribute.	After the AWS IoT Event has triggered an alert, it'll also trigger a Lambda function that checks the user database to see which integrations and actuators are associated to the garden node that has triggered an anomaly. The database will return the actuator device ids which can be sent to the hub.
Messages: Device state	The device state will determine whether or not to turn an actuator on or off.	When the AWS IoT Event processes new garden node sensor data, it can either set

		an actuator on or off depending on whether or not a threshold has been exceeded.
Protocol: gRPC REST API	The gRPC protocol is used to notify the smart hub of new messages any moment in time.	AWS provides a service called API Gateway which allows for the creation of a managed gRPC service. An AWS Lambda function will be used to respond to gRPC events. The smart hub will listen to this endpoint.

Table 2: intgrtn_cld_smrt_hb_lctrncs__rf interface descriptions

andrd_pp_intgrtn_cld_rf

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
Messages: Integration configuration	The user needs to be able to configure integrations. This means they can add a new integration, select garden nodes as inputs, and select actuators as outputs.	The GraphQL interface to the cloud will allow for CRUD (create, read, update, delete) operations for the integrations. The Android app design also showcases the layout of the pages the user will see in order to configure an integration.
Messages: Device id and name	The user needs to store the device id and name in the cloud database so that the integrations can reference them. It also serves as a record of the user's device information.	When the user adds a new device, it will also add it as a new record in the database via the GraphQL API protocol. The Android app will support the CRUD operation of these devices via the pages shown in the design.

Protocol: GraphQL API	This protocol is necessary for performing database operations and keeping track of user transactions.	AWS Amplify provides the GraphQL endpoint and manages the connections between the database and the GraphQL resolvers and schema. This service can also handle authentication. It is a serverless application.

Table 3: andrd_pp_intgrtn_cld_rf interface descriptions

4.5.5. Verification Plan

Regarding the smrt_hb_lctrncs__intgrtn_cld_rf interface, I will be demonstrating how the smart hub will be sending out data reports to the cloud via a graphical interface for sending out HTTP REST API requests.

Interfaces proven by verification

- Messages: Device id
- Messages: Temperature, moisture, air pressure
- Protocol: HTTP REST API

1. Go the public URL endpoint that the hub will be publishing data reports to
2. Go the Swagger endpoint of the url (a documented version of the API)
3. Populate the “Try It Out” section with a device id, temperature, moisture, and air pressure.
4. The API should respond with a success message.
5. The AWS IoT shadow should show that the correct device’s data was updated.

Regarding the intgrtn_cld_smrt_hb_lctrncs__rf interface, I will be showing that all clients connected to the gRPC server can receive data sent out by the server.

Interfaces proven by verification

- Messages: Device id
- Messages: Device state
- Protocol: gRPC REST API

1. After a device enters the Hot or Normal state, I will show how messages are passed down through the architecture.
2. I will have several terminals open show the logs from the gRPC server and the timer microservices. I will also have two gRPC clients to show that messages are

received, one on Postman and one as a Go application that will actually run on the hub.
3. When a request to turn an actuator on or off is received by the gRPC server, you'll see logs printed out.
4. At the same time the gRPC receives a request, all gRPC clients should receive the data instantly as well.

Regarding the andrd_pp_intgrtn_cld_rf interface, I will open the Android app and show that data is being written and read from the cloud via a GraphQL interface.

Interfaces proven by verification

- Protocol: GraphQL API
- Messages: Integration configuration

1. The user opens app
2. The user selects integration tab
3. The user selects an existing integration or adds a new integration
4. The user modifies the name, garden nodes selected as inputs, and actuators selected as outputs
5. I will show that the database operations are saved in the database in the cloud
6. I will show that the Android code utilizes the GraphQL protocol to communicate with the cloud.

Interfaces proven by verification

- Protocol: GraphQL API
- Messages: Device id and name

1. The user opens app
2. The user selects a device from the list
3. The user selects the edit menu option
4. The user changes the name and saves
5. I'll show that the database operation was saved in the database in the cloud
6. I'll show that the Android code utilizes the GraphQL protocol to communicate with the cloud.

4.5.6. References and File Links

[1] Valappil, Ruby, "What is gRPC? Is it Better Than REST API?" Medium, Mar 31, 2022. [Online]. Available: https://medium.com/javarevisited/what-is-grpc-is-it-better-than-rest-api-58a3b7aff13a#:~:text=gRPC%20is%20a%20modern%2C%20open,easier%20to%20build%20connected%20systems (Accessed Feb 19, 2023).
[2] McLarty, Matt, "API Showdown: REST vs. GraphQL vs. gRPC – Which Should You Use?" InfoQ, Jan 17, 2022. [Online]. Available:

https://www.infoq.com/podcasts/api-showdown-rest-graphql-grpc (Accessed Feb 19, 2023).
[3] Schwartz, Yedidya, “Using a Buffer Microservice and Amazon SQS to Reduce the Load on a DB.” Medium, Jun 29, 2020. [Online]. Available: https://medium.com/walkme-engineering/using-a-buffer-microservice-and-amazon-sqs-to-reduce-the-load-on-a-db-2f18831ba3fb (Accessed Feb 19, 2023).

4.5.7. Revision Table

1/18/23	Alex Feng: Document Created
2/18/23	Alex Feng: Added Section 1
2/19/23	Alex Feng: Added section 3 and 6 and finished section 5

4.6 Smart Hub Electronics

4.6.1 Description

The Smart Hub electronics consist of a plug and play touch screen that allows the user to interface with the Hub and a raspberry pi. Through the use of the two devices, the user will be able to view the nodes that are connected in the network along with statistics that are gathered. The screen is a 5 inch capacitive touch screen that will be wired into the raspberry pi in order to view and interact with our software. The raspberry pi will be the main brain of the Hub which will eventually handle the communication to all of the nodes on the network. The four blocks that the Smart Hub Electronics will interact with are the hub power block, the user interface block, the integration / cloud block and the Hub enclosure block. All of these blocks working together will form our final design for the Central Hub in our entire smart network.

4.6.2 Design

The owner of the system will be able to use the Hub as a central device to connect and monitor all of the external devices or “Nodes” connected to the smart network. The electronics of the hub consist of the raspberry pi and the capacitive touch screen. The brain of the hub is a raspberry pi 4 Model B that will eventually be running our custom

user interface in order to view the details of the system. User input is handled by a 5 inch LCD capacitive touchscreen that is connected to the raspberry pi using micro usb to usb for power and mini hdmi to hdmi for the display. The Smart Hub receives power directly into the raspberry pi from our custom made power supply block. Lastly, the raspberry pi and screen will be fully integrated into an enclosure to make a presentable product that can be sold to consumers.

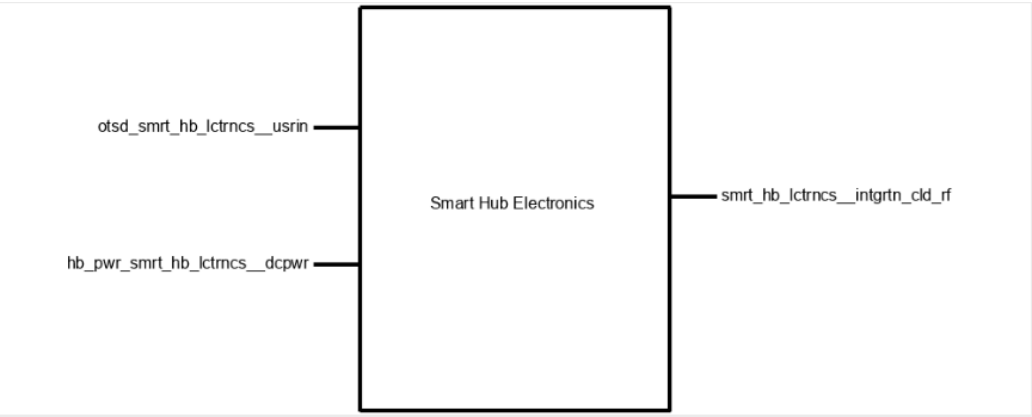


Fig. 4.6.2.1: Black Box Diagram of Block

- Otsd_smrt_hb_lctrncs_nd_sr_ntrfc_usrin: A capacitive touch screen that allows user input .
- Hb_pwr_smrt_hb_lctrncs_nd_sr_ntrfc_dcpwr: Power input into Raspberry Pi.
- Smrt_hb_lctrncs_intgrtn_cld_rf: Communication between the smart hub and the cloud network.

Fig. 4.6.2.2: Capacitive Touch Screen Connections to Raspberry Pi

The touch screen is a 5 inch capacitive touch screen display made by the company Elecrow. It is designed to be plug and play with any model of the raspberry pi and displays a resolution up to a maximum of 800 x 480 pixels. The screen itself is not built into its own enclosure which makes it perfect to be integrated into a design of our own

with the 4 available mounting holes that are included in the design. The available connections on the screen consist of a micro usb port that carries power and touch control to the screen and an hdmi port that will carry the display.

The raspberry pi that we are using is a Raspberry Pi 4 Model B. This model can be powered using usb c to provide its required 5 volts at 3 amps input. Our design for the hub will not use any of the on board pins of the raspberry pi but rather the ports that it offers. As the only devices that we are connecting to the raspberry pi are the touch screen and power brick, we will only need one of the usb ports, the mini hdmi port and the usb c port for power. The board also features an ethernet port which we might make available to the user to make it easier to connect to a home internet network.

4.6.3 General Validation

The Smart Hub will be the central device that will allow our users to interact with and manage their network. With this in mind, we made our design decisions based on two things, the ease of use and the look of the product.

We decided to go with a capacitive touchscreen for our user input in order to streamline the process of managing the network. We decided that a touchscreen on the hub would be better than having either buttons for control or external devices such as a keyboard and mouse. With just the need to have the touchscreen implemented into the Hub enclosure, this will ensure that our device has a sleek design that can comfortably be placed in any room without it looking out of place. The touchscreen will also improve the devices overall ease of use by not confusing the user with random onboard buttons. The user will be able to use the touchscreen on the hub just like any other touchscreen device such as a phone or tablet. Having a screen to display our user interface will also allow us to display important details and statistics that are gathered by the different nodes in the system. This way, the information that the user wants will always be displayed and can be casually viewed when walking past the device or having it in the corner of the room.

We made the decision to go with a raspberry pi 4 model B due to its market availability and the features that it offers. This specific model of the raspberry pi offers a mini hdmi port as well as a usb port which will allow us to connect the screen to the raspberry pi without any issues or workarounds. Its usb c port is perfect for allowing us to design our own power supply and not have to worry about its connection into the board. The model B version of the raspberry pi also offers an ethernet port which we may decide to use in order for our users to connect to their home network a little easier.

For the communication between the Smart Hub and the external nodes in the system, we decided on using the Matter protocol due to its compatibility with our raspberry pi and its easy to use programming environment. Using the Matter protocol allows our device to not only communicate with other nodes in the system but also with our android Smart Home app that will be able to be downloaded onto a phone.

Lastly, we decided to design our own power supply so we can better regulate how

much power is being supplied to the Hub. This will allow us to save on power consumption while also improving the longevity of our device.

4.6.4 Interface Validation

otsd_smrt_hb_lctrncs_nd_sr_ntrfc_usrin : Input

Interface Property	**Why is this interface, this value?**	**Why do you know that your design details for this block above meet or exceed each property?**
Timing: At any random time for the length of a press	Users will be able to input onto the touch screen at any moment just like a phone or tablet.	The touch screen is wired into the Raspberry pi and will constantly be checking for user input.
Type: Touch Screen	A touch screen will be used to handle user input into the Smart Hub	A touch screen is the easiest and cleanest way to handle user input into our system. [1]
Usability: Needs to be able to be used by at least 9 out of 10 users.	Our touch screen needs to be easy to use by any of our users.	The touch screen is plug and play with the raspberry pi and only relies on the user being able to touch the screen with their fingers.

Smrt_hb_lctrncs_intgrtn_cld_rf : Output

Interface Property	Why is this interface, this value?	Why do you know that your design details for this block above meet or exceed each property?
Data Rate: Max size of 1mB	Only a small amount of data needs to be transferred per message.	Less than 10 custom properties are required per message and a data rate of 1mB will cover that.
Protocol: Matter WLAN	Matter is the smart home protocol used for communicating between the Hub and the Garden Nodes.	Matter is compatible with the raspberry pi 4 model B and allows it to communicate with other devices on the network.

hb_pwr_smrt_hb_lctrncs_nd_sr_ntrfc_dcpwr : Input

Interface Property	Why is this interface, this value?	Why do you know that your design details for this block above meet or exceed each property?
Inominal: 3A	This current was chosen based on the expected current needs of the smart hub.	Both the raspberry pi and the touch screen are well within the 3A range. [2]
Ipeak: 3.2A	We expect the smart hub to never spike above this current.	From what we expect from the smart hub, it will never reach 3.2A. [2]
Vmax: 5.4V	This value was chosen based on our design of the power supply block.	The raspberry pi is run on 5V so this gives us room to work with. [2]
Vmin: 4.6V	This value was chosen based on our design of the power supply block.	The raspberry pi is run on 5V so this gives us room to work with. [2]
Nominal: 5V	This value was chosen based on our design of the power supply block.	The raspberry pi is run on 5V. [2]

4.6.5 Verification Plan

In order to verify that this block meets all the requirements and needs of our system, we will undergo the following steps:

1. Connect the power supply to the usb c port on the raspberry pi and verify that it turns on.
2. Attach the touch screen to the raspberry pi using hdmi to mini hdmi and micro usb to usb and verify that the screen is supplied with power and displays an image.
3. Interact with the operating system of the raspberry pi by using the touch screen controls.
4. Keep the touch screen turned on and verify that the raspberry pi does not lose power over the course of a day.

4.6.6 References and File Links

[1] Milton Kazmeyer, “Benefits of Touch Screen Technology”, [Online] Available: https://smallbusiness.chron.com/benefits-touch-screen-technology-54942.html (Accessed: Jan 19th, 2023).

[2] Hammad Zahid, “How Much Power Does Raspberry Pi Consume While Operating”, April, 2022. [Online] Available: https://linuxhint.com/power-consumption-raspberry-pi/ (Accessed: Jan 19th, 2023).

4.6.7 Revision Table

1/19/23	Carson Ehlers: Section Created
2/8/23	Carson Ehlers: Document revised based on feedback

4.7 Smart Hub GUI

4.7.1 Description

The Smart Hub GUI is the graphical user interface that will be loaded onto the Hub's Raspberry pi and displayed on the 5 inch capacitive touch screen. The UI is a code block that is written using the python language. The code utilizes a python library named Dear PyGUI [1] which is a powerful tool used for making interactive user interfaces. Through the use of the UI, the user will be able to add external devices or "nodes" to the homepage and from there, be able to monitor the statistics that the devices gather. These statistics include temperature, humidity, soil moisture, air pressure, and light level. These statistics are input into the UI through the use of a text file that is sent from the external nodes.

4.7.2 Design

The Smart Hub GUI is designed to be loaded onto the Smart Hub’s raspberry pi and displayed on the 5-inch touchscreen at a resolution of 800x450. On system startup, the GUI will automatically start and will display the user’s home screen with all of their added devices.

The GUI is written using a python library named Dear PyGUI [1]. This library is an easy-to-use, dynamic, graphical user interface toolkit that features traditional GUI elements such as buttons, menus and various methods to create a functional layout. The

library is available on Windows 10, Linux, and more importantly for us, the Raspberry Pi 4 OS.

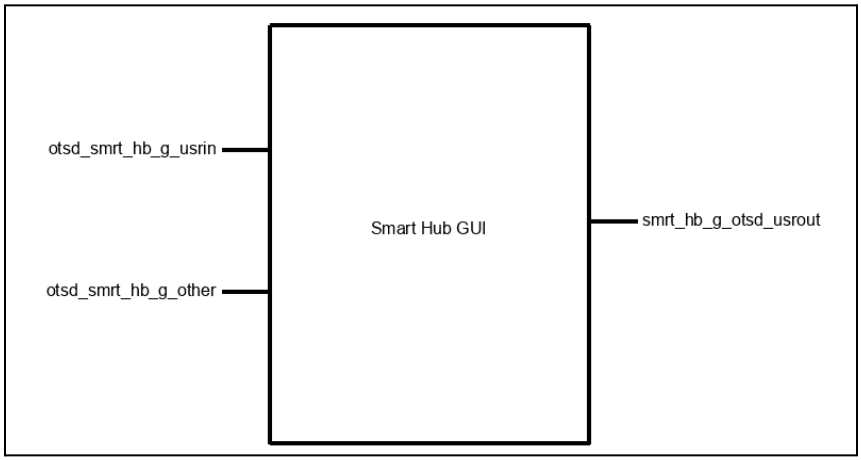


Fig. 4.7.2.1: Black Box Diagram of Block

The home screen of the GUI is the primary viewport that device will be displayed. On this screen, the user is able to see all of their connected devices as well as the statistical readings that are gathered for said device. The home screen also includes a button on the top right that allows the user to add a new device to their home screen.

Upon pressing the add new device button, the user will be brought to the new device screen where they will be prompted to enter a device ID code for the device that they want to add to the home screen. When a valid device ID code is entered, the user will click the “Add” button and then be brought back to the home screen. Once a new device has been added, the home screen will display the device along with the statistics that are being recorded by the external device.

Lastly, the GUI will constantly be saving the devices and their values to a text file in order to not lose this information when the device is shut down. Every time the device is powered on, the interface will read in values from the text files and the GUI will pick up right where it left off.

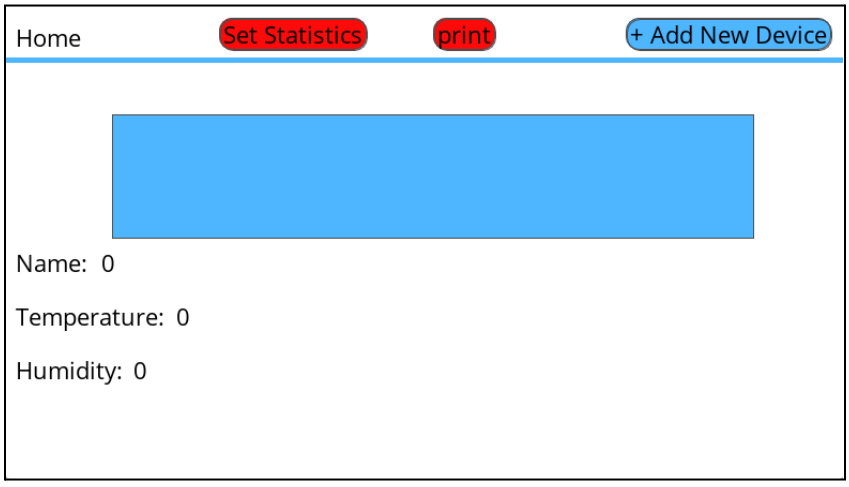


Fig. 4.7.2.2: Home Screen

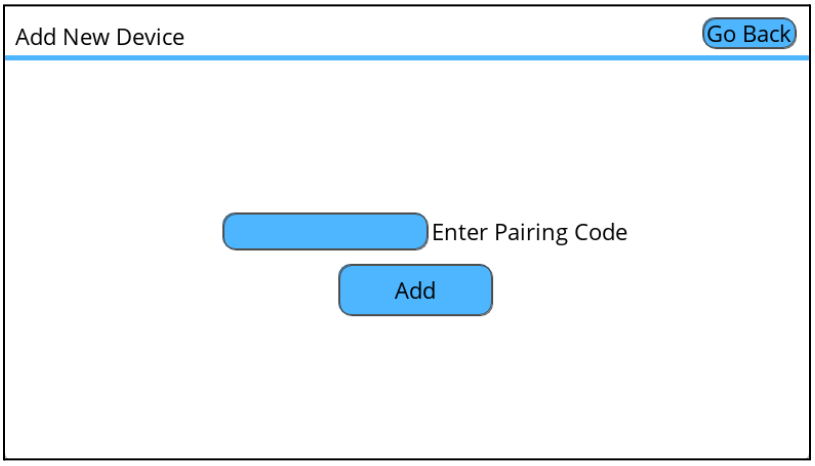


Fig. 4.7.2.3: New Device Screen

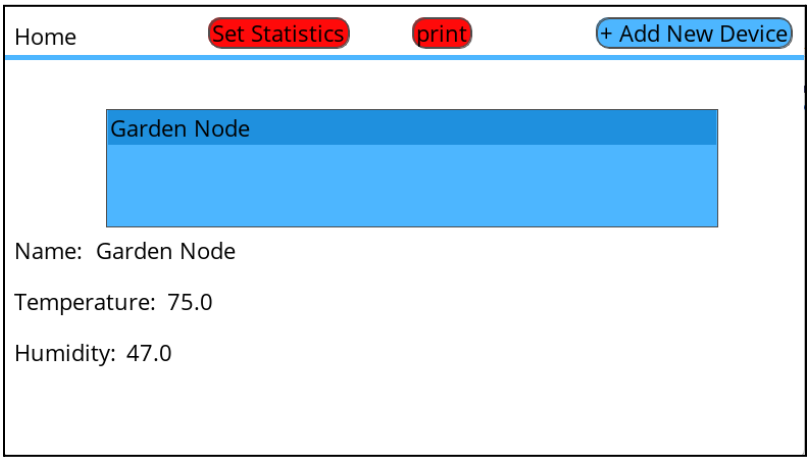


Fig. 4.7.2.4: Home Screen with Device Added

4.7.3 General Validation

For the Smart Hub GUI, we need to accomplish two things. The first is providing our users with an interface that allows them to control their system on the Hub, and the second is being able to display the statistics that are gathered from the external nodes in our system.

In order to create an interface that accomplishes what we need, our team looked through many different tools but ultimately decided on using the Dear PyGui library. This library

gives us everything that we need in order to create an interface that is easy to use and understand while also giving our users complete control over their system.

For the home screen of the GUI, we decided to show all of the connected devices in the form of a drop down list and upon clicking one of these devices, its statistics will be displayed on the lower half of the screen. We went with this format because we discovered that it was the cleanest way to show our user all of their devices and information on one page. The list of devices is a scrollable list box that will ensure that the user can see all of their devices regardless of how many they add.

One problem that we encountered while making the GUI is that every time the device is powered off and reset, all of the saved devices and information would be lost. We eventually solved this problem by allowing the GUI to save information to a text file before shutting down to ensure that everything is saved. We want our users to be able to turn off the hub whenever they want so this is the best way that we found to ensure that they will not lose information and have to redo adding everyone of their devices every single time the Hub is powered on.

4.7.4 Interface Validation

otsd_smrt_hb_g_usrin : Input

Interface Property	Why is this interface, this value?	Why do you know that your design details for this block above meet or exceed each property?
Other: Ability to add devices to home screen	The user needs to be able to add our external devices to the home screen GUI in order to view statistics.	The GUI includes a button that allows the user to add devices to the home screen.
Other: On screen buttons that controls the interface	The user needs to be able to control the Hub using on screen buttons because it is a touch screen.	The GUI is created with on screen buttons that can control every aspect of the interface. No other input is needed.
Usability: Usable by at least 4 out of 5 users	The GUI needs to be understandable by a majority of users.	The GUI is simple and every section/button is properly labeled with what is displayed.

otsd_smrt_hb_g_other : Input

Interface Property	Why is this interface, this	Why do you know that your

	value?	design details for this block above meet or exceed each property?
Other: Read values in from text file	The GUI needs to be able to read values from a text file in order to get statistics that are produced by other devices.	The GUI reads strings of input from text files on startup and whenever the action is requested through the interface.
Other: Save information to a text file	The GUI needs to save information to a text file so the information can be saved when the device is powered off.	The GUI saves strings of text to a list of text files when it is powered down and whenever the action is requested through the interface.

Smrt_hb_g_otsd_usrout : Output

Interface Property	Why is this interface, this value?	Why do you know that your design details for this block above meet or exceed each property?
Other: Displays statistics gathered from connected nodes	The user needs to be able to view the statistics gathered by other devices.	The home screen of the GUI has a section where all the information is displayed under their proper heading.
Type: Displays output through a screen.	The Hub uses a screen to display the GUI.	The GUI is loaded onto the Hub's touch screen as is able to be viewed.
Usability: Usable by at least 4 out of 5 users	The output through the screen needs to be understandable by a majority of users.	The Hub consists of a simple touch screen that will turn on when the device is supplied power.

4.7.5 Verification Plan

In order to verify that this block meets all the requirements and needs of our system, we will undergo the following steps:

1. On startup, ensure that the GUI displays the home screen.
2. Click on the + Add New Device button on the top right and ensure that the GUI is now displaying the add new device screen.
3. Type in any random name into the pairing code text box and click the Add button.

4. Ensure that the GUI is now back to the home screen and that the device that was added is now displayed on the home screen.
5. Use the “Set Statistics” testing button to give the node that was added temperature and Humidity values. After clicking the button, enter in the name of the node and the values to be set. Then click save.
6. Use the go back button to go back to the home screen.
7. Click on the added node and ensure that the values that were set are now being displayed under their correct labels.

4.7.6 References and File Links

[1] Dear PyGui, “Dear PyGui’s Documentation”, [Online] Available: https://dearpygui.readthedocs.io/en/latest/ (Accessed: March 8th, 2023)

4.7.7 Revision Table

3/08/23	Carson Ehlers: Section Created

4.8 Hub Power

4.8.1 Description

The Hub Power block sub-system is the sole power source for the raspberry Pi 4B and associated touch screen. The touch screen works directly with the Raspberry Pi and draws its power from it. This requires that the power supply can provide 5V at 3A, as per the Raspberry Pi’s maximum input, and provide a connection to a USB-C power input. To provide the necessary power the power supply will draw its power from a wall outlet and transform it to the required voltage.

The system will connect the wall power input to the Hub power system using a IEC C13 wire that will connect to a fused junction that will pass the power into the enclosure. The power supply converts from the 120V Nominal power from the wall input to 12V using a transformer. Then using a full wave bridge rectifier the AC input is transformed into a DC current. The 12V input is then stepped down to

5V using a switching rectifier and is put through a low pass filter which was based off of the recommendation of the AP63356QZV7. The output of the system is a USB-C Female connector and will use a USB-C to USB-C to power the Raspberry Pi.

4.8.2 Design

The Hub Power block is designed to convert the input of 120Vac 60Hz wall power to the output of 5V 3A nominal to power the Smart Hub Electronics and User Interface sub system. The sub-system is the combined Raspberry Pi 4B and a 5 inch touch screen display that is designed to work with a raspberry pi. The touch screen relies on the Raspberry pi for power meaning that the power supply needs to supply enough power to meet the Raspberry Pi 4B's maximum power input which is 5V 3A.

The power supply draws its power from a wall outlet with a IEC C13 wire and transitions into the enclosure through a PF0001 power connector that provides a 0.5 A fuse and connects to a free standing transformer which lowers the voltage to 12V at 1.63A, providing 20W. The transformer connects to the PCB through a 282834-2, a screw clamp connector. The power is passed through a KMB26S bridge rectifier and filtered with a 1000 µF capacitor. To step down from 12V to 5V the switching DC-DC buck converter AP63356QZV-7 is used and the recommended components from its data sheet have been placed to match the recommendation as closely as possible. The PCB will be able to output 5V at 3A through a USB-C power only female output, and the PCB will $2\frac{1}{2}$ x 1 square inches.

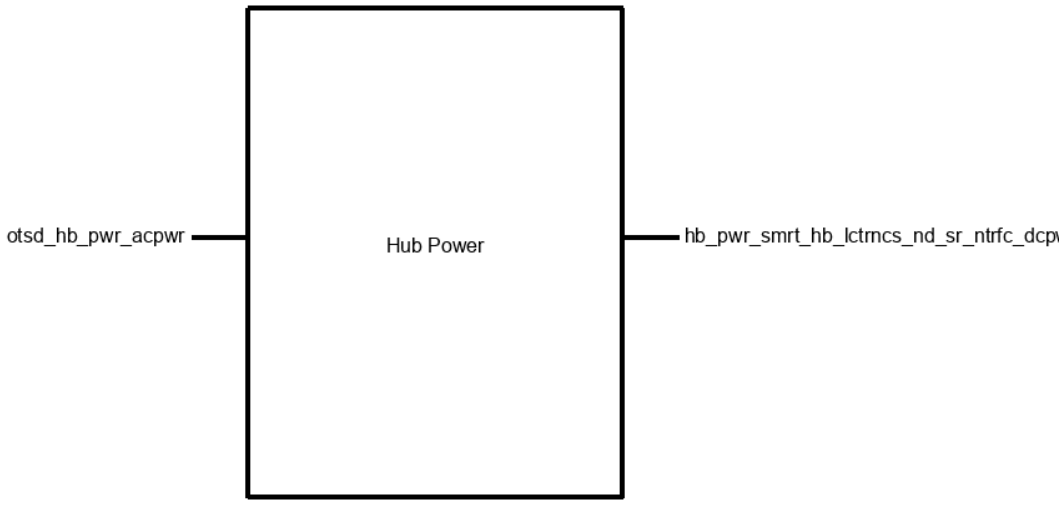


Fig. 4.8.2.1: Block interface connection.

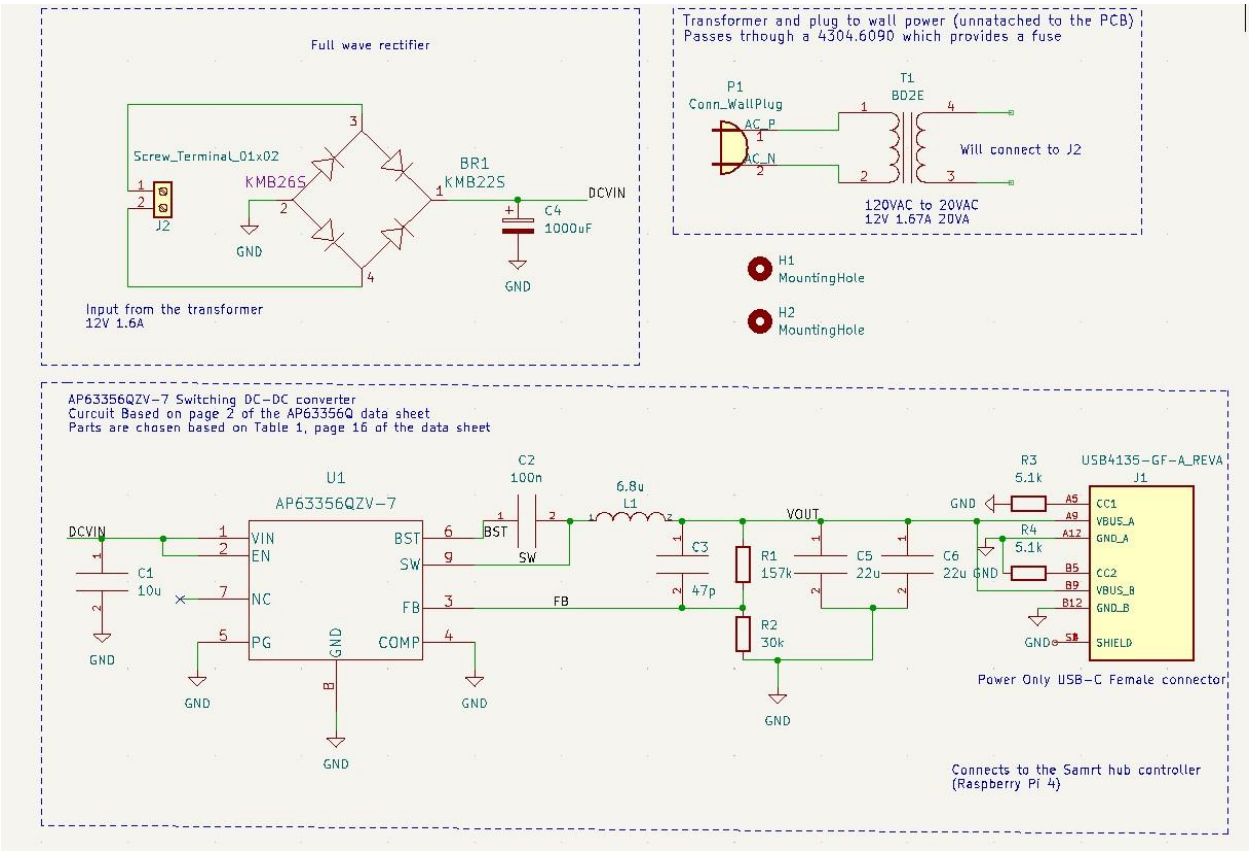


Fig. 4.8.2.2: Schematic of the power supply.

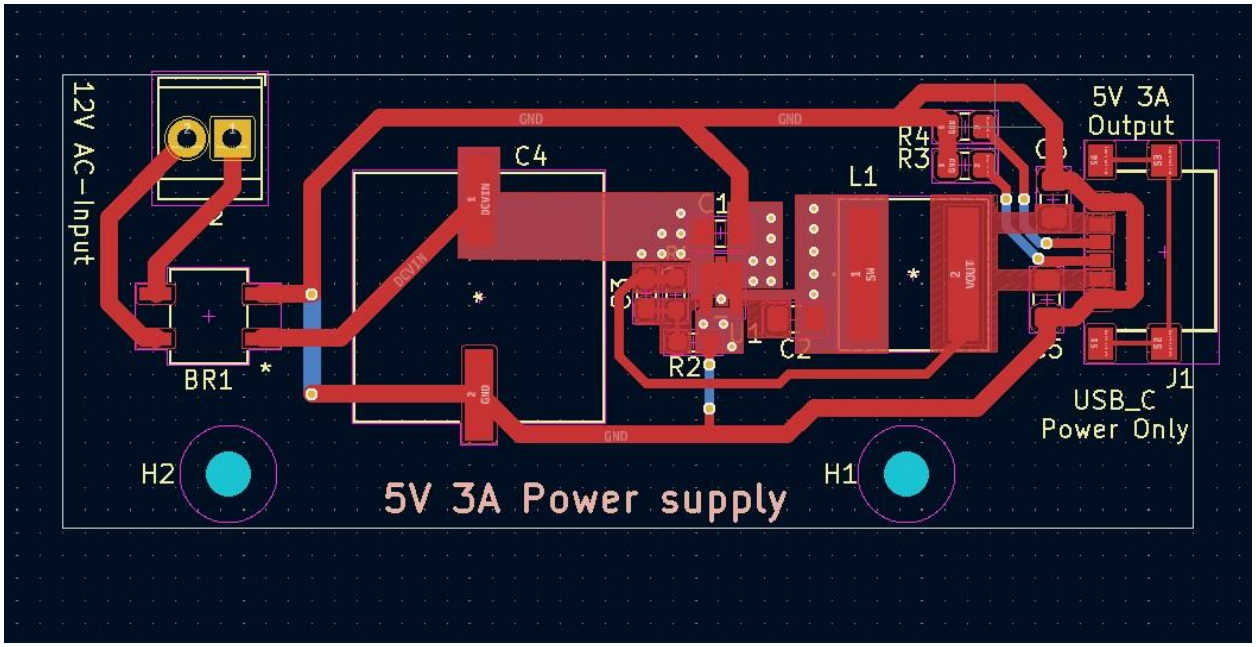


Fig. 4.8.2.3: PCB design of the power supply.

4.8.3 General validation

There are three major requirements that the power supply has to meet: it has to run off of 120VAC 60Hz power, provide 5V at 3A nominal for the Raspberry Pi 4,[1] and it must be at least 65% efficient. There are other considerations that are being taken into account that fall under two sections, the part selection considerations are aimed at keeping the cost to a minimum, having protections from current fluctuations and thermal overload, limiting noise, and selecting terminals that can interface with the input and output. Then there are a separate set of considerations going into the PCB design. The first is that the traces are capable of carrying the required current, second that the components are big enough that I can use them while still maximizing their value, and third the system has to be able to dissipate the heat created by the power converter. To meet the goals a simple power supply model consisting of a transformer, a full wave rectifier, and a dc-dc converter will be implemented.

The design for the power supply is based around the DC-DC converter. The converter had to be a switching converter to be able to handle the high current while keeping efficiency above 65%. To have a slight current buffer from the required maximum current I looked through converters that could handle at least 3.2A. The only stocked chips under five dollars were a family of 4 AP6335 switching converters. I chose the QZV section of the family because they could operate at higher temperatures than the DV counterparts and with a constant 3A load the larger range of operation will reduce overheating risks. Between the 6 and 7 of the remaining options the 7 had better efficiency at lower currents but had larger voltage drop off at high currents than the 6. Because the power supply is expected to provide currents outside the range of benefit of the AP63357QZV, I decided to use the AP63356QZV[2].

The AP63356QZV has additional benefits beyond the major requirements that make it reasonable to use it over a more expensive chip. It can provide the necessary current while remaining above 90% efficient. It also has protections shutting itself down if it overheats or if there is a current overdraw. The AP63356QZV is also able to provide the necessary power to the Raspberry pi while remaining well within its maximum power range which should increase its lifetime and avoid triggering the current overdraw protections.[2]

The power supply outputs 15 watts nominal, and the DC-DC converter is 90% efficient with a 12V supply and including the voltage drop across the bridge rectifier the transformer needs to be able to supply 12V at 1.5A. Of course

transformers are not made for 18VA and there should be a buffer so I decided to use a 20VA, 120V to 12V transformer. There are some alternatives. I found a 20VA 8V output but chose not to use it because a 8V decreased the efficiency of the DC-DC converter. Another alternative was a 30VA transformer with a 10V output however it costs twice as much and while it offers a slight improvement to I peak it does not change the functionality of the device enough to be worth the cost.[3]

The other issues that required my consideration were how to connect the power supply to an input and output. The power supply PCB will reside within the Hub enclosure requiring that something acts as a transition for the input. The best transition I found between the outside and the inside of the enclosure was a PF0001 power connector that provides a fuse slot at a reasonable size and price. To connect it to an outlet it needs an IEC C13 wire which is a cheap and common power cord. The inside of the PF0001 also provides easy access to the pins which will work well with the transformer.[4]

For the output I decided to use a power only USB-C female connector because the raspberry pi uses a USB-C connector for its power input, a USB-C to USB-C wire is a fitting connection, and by providing a wire connection instead of a direct connection between the power supply and the raspberry pi, it will allow for more flexibility within the enclosure.

To design the PCB the packages for each component and trace sizes between them had to be considered. The circuit design is based off of the AP63356QZV data sheet for the resistors and capacitors are the recommended ones from the data sheet for a 5V output. There is a benefit for both accuracy and a smoother signal the closer they are to the DC-DC converter. To do this the smaller the chip the closer it can be to the inputs. Having to place the components on the board myself limits how small they can be, and because of this the smallest package I am willing to use is a 0603, and wherever the size of the component is less impactful I will use an 0805 package.

The trace sizes of the PCB were also an issue because some traces had to carry up to 3.2 amps. To provide enough copper a trace would need to be about 60 mils wide at 1oz/ft^2. Taking into account that the output had two separate ground pads I decided to use two 40 mil traces to act as ground because that would be able to carry the I peak current with a reasonable excess. For the output I found that the copper fill zones could easily connect the components while providing enough copper for 3.2 amp current to flow and were used in place of traces. The

copper fill zones can also be used to dissipate heat and are being used for just that. While a heatsink can be added later if there is not enough heat dissipation, I believe by adding a lot of vias as recommended by the data sheet around the power converter that a heatsink can be avoided.

4.8.4 Interface validation

This section Shows the interface properties of the input and output for the hub power supply block. For each table the first column shows the interface value, the second column explains why each value was chosen, and the third column explains why the block will meet that requirement.

Input: otsd_hb_pwr_acpwr

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
I peak 0.17A	The peak current would be the maximum power draw of the transformer[1].	● The transformer is capable of supplying 20VA. To supply 20VA at 120V the input would require 0.17A.
V nominal 120VAC	We are using wall power and 120V AC at 60Hz is standard.	● 120V AC 60Hz is the standard in the US for wall power and this is intended to be plugged into a wall.
I nominal 0.15 A	Most outlets can handle up to 15A so there should be no issue reaching 0.15A.	● The nominal output of the power supply is 15W, to match this the system will have a nominal input of 17.5W, for 120V to provide 17.5W it needs a current of 0.15A.

Output: hb_pwr_smrt_hb_nd_sr_ntrfc_dcpwr

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each

		property?
I nominal 3.0A	The maximum power draw of the raspberry pi 4B is 3A as indicated by its data sheet.	• The DC-DC converter is rated to supply up to 3.5A. • The inductor is rated for 3.5 A. • The transformer supplies 20VA which after the bridge rectifier is 18 watts. Giving the converter enough power to supply 3.2A at 5 volts.
I peak 3.2A	The peak current is based on the maximum power the transformer can supply assuming the DC-DC converter is supplying 5V.	• The transformer supplies 20VA which after the bridge rectifier is 18 watts. Giving the converter enough power to supply 3.2A at 5V.
V max 5.4V	The max voltage is chosen based on the worst case scenario of the resistors tolerances.	• Using 5% tolerance on resistors the maximum error when targeting 5V is a 5.4V output.
V min 4.6V	The max voltage is chosen based on the worst case scenario of the resistors tolerances.	• Using 5% tolerance on resistors the maximum error for 5V can produce a 4.6V output.
V nominal 5V	The nominal voltage was decided since the DC-DC converter circuit is designed to deliver 5V.	• Using the resistor values recommended by the DC-DC converters data sheet to achieve 5 volts.

4.8.5 Verification Plan

The verification plan is a step by step description of how the input and output will be tested to see if they meet their interface requirements.

1. To verify that the wall power supplies 120V with a nominal current of 0.15 Amps the system will be plugged in and hooked up to a variable load, and a clamp meter will be attached to the wire connected to the wall outlet.

2. The variable load will be set to 3.0A and the voltage across the load will be measured and compared to the nominal voltage. Once the nominal voltage and

current have been checked the clamp meter will be compared to the nominal current input.

3. The variable load will be set to 3.2A to verify the peak output current. The voltage across the load will be measured and compared to the maximum and minimum voltage. The clamp meter will then be compared to the 0.17A peak current input.

4.8.6 References and file links

[1] Raspberry Pi, “Raspberry Pi 4 Model B,” datasheet, June. 2019. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
[2] Diodes Incorporated, “AP63356Q/AP63357Q,” datasheet, September, 2020. https://www.diodes.com/assets/Datasheets/AP63356Q_AP63357Q.pdf
[3] Hammond Manufacturing, “Class 2 Energy Limiting Small Box Mount”, datasheet, 2023. https://www.hammfg.com/electronics/transformers/class-2/ba-c-d-e.pdf
[4] Bulgin, “IEC Connectors”, datasheet. https://www.bulgin.com/products/pub/media/import/attachments/IEC_connectors.pdf

4.8.7 Revision Table

Date	Revision
1/20/23	Peter: created document
1/25/23	Peter: Updated the description section to properly describe the project
2/11/23	Peter: Updated General validation

4.9 Garden Enclosure

4.9.1 Description

The Garden Enclosure Serves two purposes. The first is to provide a space protected from wind and rain that gives enough space to mount the PCB holding the circuitry used read, and send the environmental inputs, the battery and solar panel used to keep the system powered, and the lux sensor. Secondly the enclosure must provide two outlets, one for the soil moisture sensor, and another for the temperature and humidity sensor, because these sensors cannot function properly inside an enclosure.

The outside of the enclosure is a large waterproof enclosure bought from adafruit, it provides protection for the elements and provides five square inches of floor space beyond what is needed to give plenty of space for wired connections. To mount the PCB and batter the enclosure will contain a 3d printed mounting block that will provide the necessary connections. And to mount the lux sensor and the solar panel they will be epoxied to the clear roof of the enclosure to maximize sunlight. For the outlets the enclosure provides two cable glands that create a waterproof outlet for wires between 0.12 and 0.25 inches in diameter.

4.9.2 Design

The Garden enclosure is being designed to keep its contents dry in a rainstorm, allow light into the enclosure, and provide enough surfaces to mount the solar panel, the battery, the lux sensor, and the PCB. It also must provide two outside connections for the cables of the soil moisture sensor and the temperature/humidity sensor. The enclosure has a 3d printed standoff for components to mount to, it has a clear top to allow sunlight through the enclosure, and it uses cable glands to provide outside connections while preserving the enclosure's integrity against water.

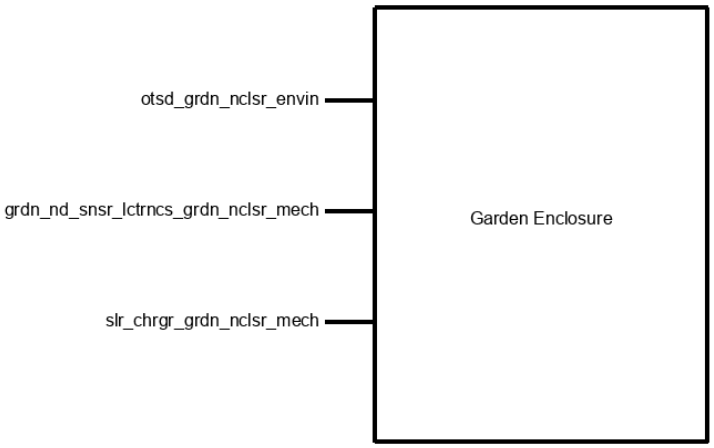


Fig. 4.9.2.1: Garden enclosure interface connections

otsd_grdn_nclser_envin covers the waterproof interaction between the enclosure and the environment

grdn_nd_snsr_lctrncs_grdn_nclsr_mech is the mechanical connections for the enclosure and the sensor electronics.

slr_chrgr_grdn_nclsr_mech is the mechanical connection between the enclosure and solar charging system.

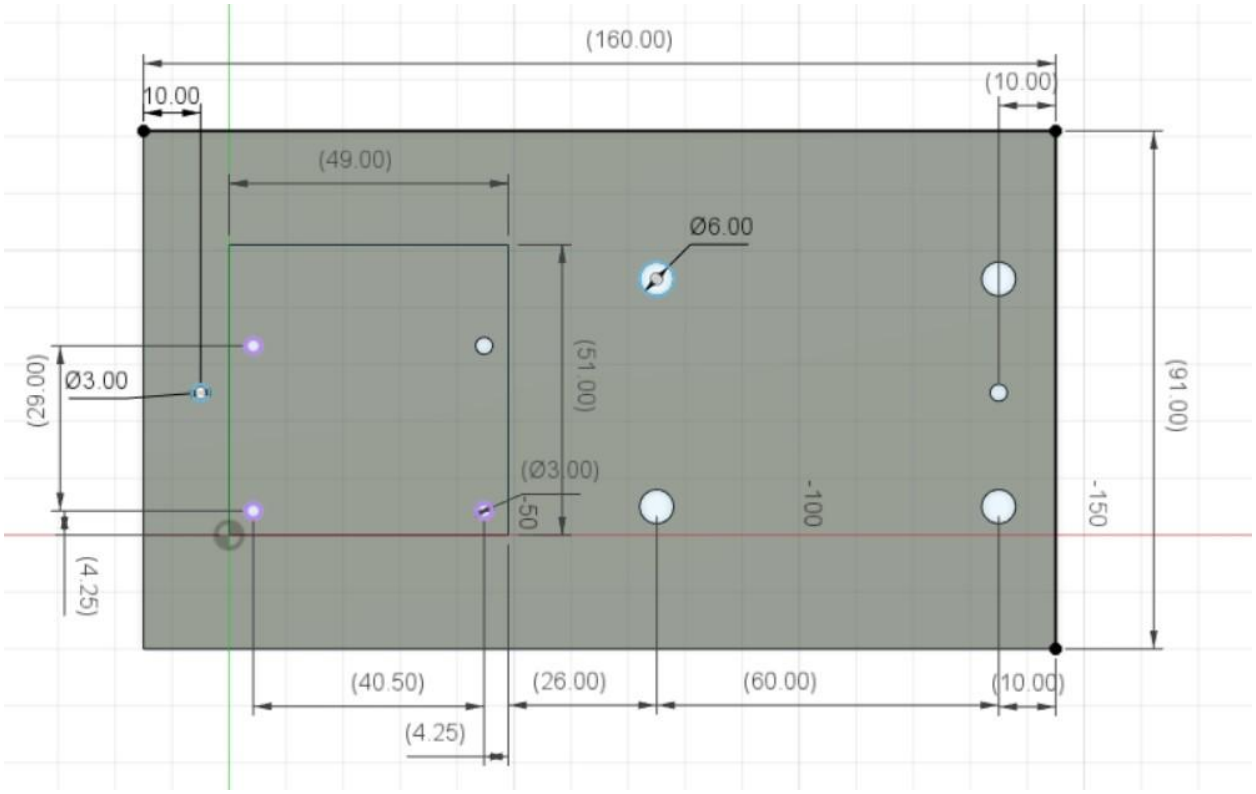


Fig. 4.9.2.2: mounting block for the PCB and battery

4.9.3 General validation

Excluding the cable glands the enclosure is 180 X 130 X 80 mm, and the enclosure is weatherproof rated for IP66, meaning it should be able to withstand high pressure water sprays from all directions, and while this is vague high pressure implies that it can easily withstand the heavy rain it is being designed for[1]. The outside is a high density polyethylene which is weather resistant and will not break down under ultra violet rays.

For the inside of the enclosure the standoff is made out of PLA plastic because it is a common plastic and I had it on hand. The standoff must be able to hold a 67 X 56 X 20 mm lithium battery [3], and a 41 by 51 mm PCB with 3m mounting holes in a rectangle 41 mm across widthwise and 29 mm lengthwise. The battery will be held using two zip ties across its width as shown on the right side of image 2, and the pcb will fit on the left side of image two, and the holes are oriented to allow the wires ample room to reach their destinations.

The cable glands are on the edge of the box near the respective PCB interface to simplify the connections and they will provide a waterproof hole in the container. There

are two separate types of cable glands, the first one fits wires of 0.158” to 0.252”, the second is made to fit a smaller wire of 0.118” to 0.169” diameter[3].

4.9.4 Interface Validation

Otsd_grdn_nclsr_envin

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
Water Proof: A gallon per minute from all sides	It needs to meet the global requirements.	The outside of the enclosure is pre bought and rated for IP66, and for the wires leaving the box they are protected by waterproof cable glands.

grdn_nd_snsr_lctrncs_grdn_nclsr_mech

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
Fasteners: 4 3m screws	The PCB that the enclosure holds has 4 m3 mounting holes in a 29 by 41 mm rectangle.	The standoff provides 3m holes 41 by 21 mm apart oriented to simplify the connection wires.
Other: 3.25mm diameter exit hole	The soil moisture sensor wire is 3.25mm diameter.	There is a 4mm hole in the side protected by a cable gland that is compatible with 3.25mm wires.
Other: 3.5 mm diameter exit cable gland	The temperature and humidity sensor connection wire is 3.5mm in diameter.	There is a 4mm hole in the side protected by a cable gland that is compatible with 3.5mm wires.

slr_chrgr_grdn_nclsr_mech

Interface Property	Why is this interface property	Why do you know that your

	this value?	design details for this block above meet or exceed each property?
Mechanical connection: 2 zip ties to hold down the 56.5x69x19 mm battery	The battery is 56.5x69x19 mm and has no way to mount it.	Zip ties can provide enough static friction to hold the battery in its place

4.9.5 Verification Plan

To test that the system is waterproof:

1. A gallon jug of water will be filled halfway.
2. Over a 30 second period the half gallon of water will be poured onto one side of the container.
3. Steps one and two will be repeated for the remaining sides.

To test the connections for the sensor electronics:

1. Observe that the wires fit through the cable glands.
2. Observe that the pcb is properly mounted.

To test the connection for the Battery:

1. Observe that the battery is properly mounted.

4.9.6 References and File Links

[1] TOYOGIKEN, “TOBOX,” datasheet.https://cdn-shop.adafruit.com/datasheets/TIBOX.pdf
[2] PKCELL, “ICR18650 10050mAh 3.7V,” datasheet, May, 2021.https://cdn-shop.adafruit.com/product-files/5035/5035_10050mAh_3.7V_A1____20210511.pdf
[3] Cable glands direct, “NPT-12,” datasheet.https://cableglandsdirect.com/wp-content/uploads/2020/07/NPTspecsheet.pdf

4.9.7 Revision Table

3/08/23	Peter Thompson: Section Created

4.10 Hub Enclosure

4.10.1 Description

The Hub enclosure is a 163 X 125 X 110 mm box with rounded corners. The Base consists of the front which provides a stable mounting structure for the 5” LCD touch screen, the bottom that provides four standoffs for the Raspberry pi 4 and the left and right wall. The second piece consists of the top and it has two holes for 3m screws that hold the system together, and the back piece which has a 7mm square hole to allow the power cable through to the Raspberry Pi.

4.10.2 Design

The Hub enclosure is designed to give a good external mounting surface to an ELECROW Raspberry Pi Touchscreen Monitor 5 inch Screen Display, as well as provide a way to mount a Raspberry Pi out of sight of the users. The overall dimensions of the enclosure were chosen to fit the 5” screen and give it enough space to plug in the needed wires with about a centimeter extra room length and height wise for ease of access, the depth was chosen to be an arbitrary size larger than the height of the enclosure.

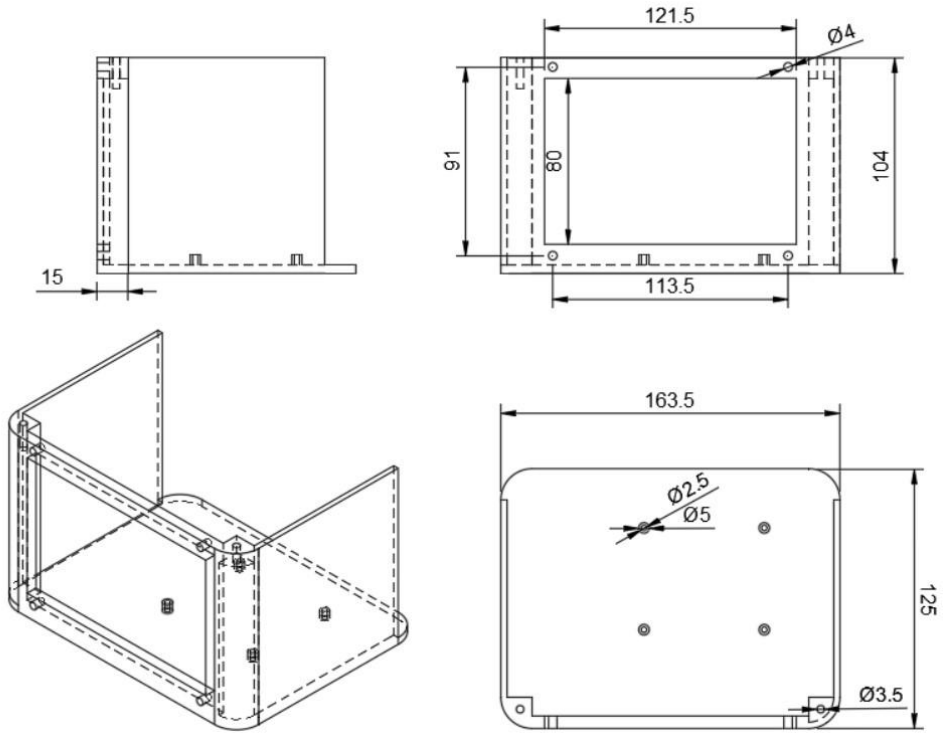


Fig. 4.10.2.1: Hub enclosure base drawing

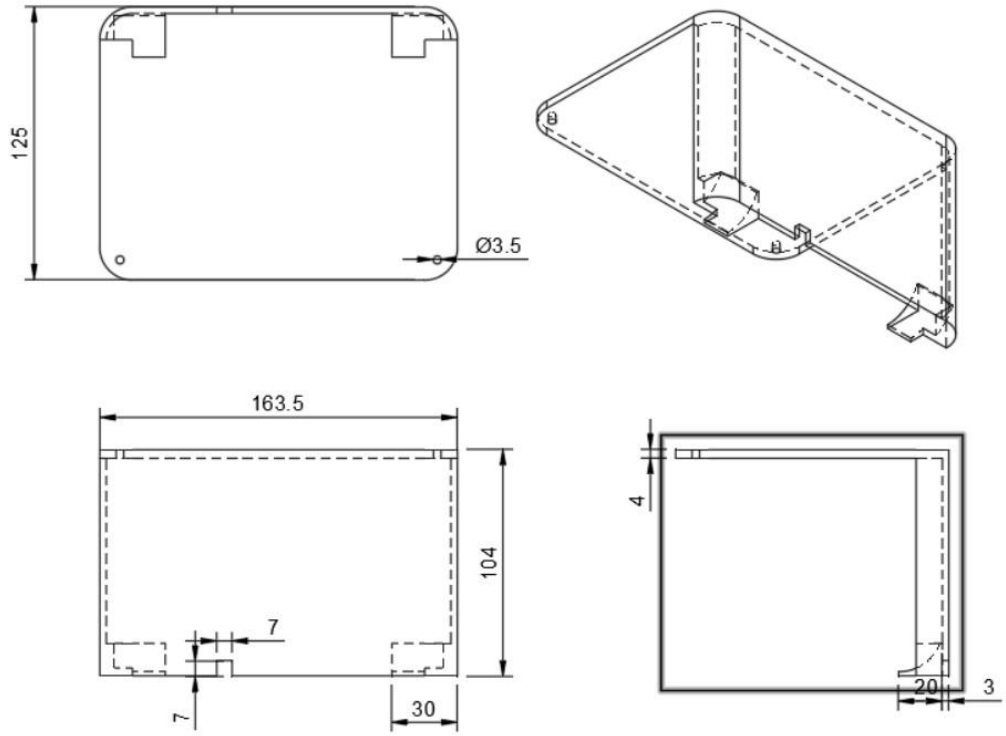


Fig. 4.10.2.2: Hub enclosure top drawing

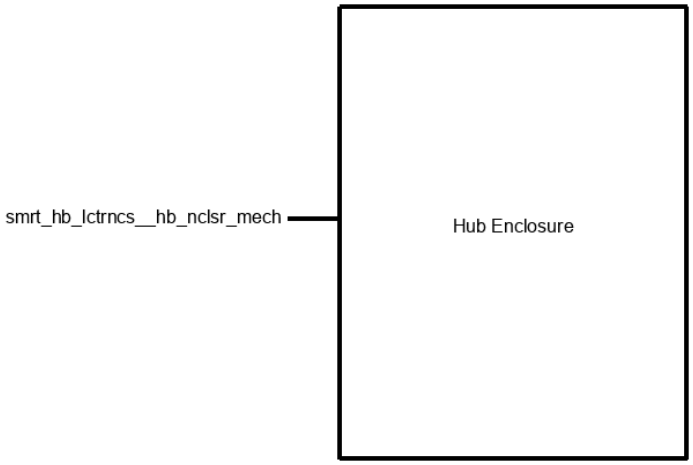


Fig. 4.10.2.3: Hub enclosure interface connections

4.10.3 General validation

The Hub enclosure is designed to fit on a table indoors. It is built using PLA plastic because the system is protected from the weather and sunlight, and PLA is easily accessible. It is also designed to discreetly fit in a house, keeping a low profile while still allowing the user easy access to the information presented on its screen. It provides

screw holes for mounting the screen as well as standoffs for the Raspberry Pi inside the enclosure[1]. The standoffs are placed to give ample room for the wires and power supply to connect to the raspberry pi, and the front was widened to give space for the wiring to the screen. To mount the screen 3.5mm holes have been placed in a rectangle with length 113.5mm and a height of 91mm to give the screen a way to mount onto the 79 by 121 mm opening in the front[2].

4.10.4 Interface Validation

Smrt_hb_lctrncs_hb_nclsr_mech

Interface Property	Why is this interface property this value?	Why do you know that your design details for this block above meet or exceed each property?
Fasteners: 4 3m screws	The 5” touch screen has four mounting flanges made for 3m screws	There are four holes larger than 3 mm positioned for the screen.
Other: Must have an 79 X 121 mm opening	The screen needs to be in the enclosure for wiring needs while the user needs to be able to interact with the screen.	The enclosure has a 80 X 121.5 mm hole in the front to fit the screen into.

4.10.5 Verification Plan

1. Place the screen in its given hole and screw it into place.
2. Observe that it fits.

4.10.6 References and File Links

[1] Raspberry Pi, “Raspberry Pi 4 Model B,” datasheet, June. 2019.
https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-mechanical-drawing.pdf
[2] QIAO DIAN XIAN SHI Co, “QD050001C0-40,” datasheet.
https://www.elecrow.com/download/QD050001C0-40%20HDMI.pdf

4.10.7 Revision Table

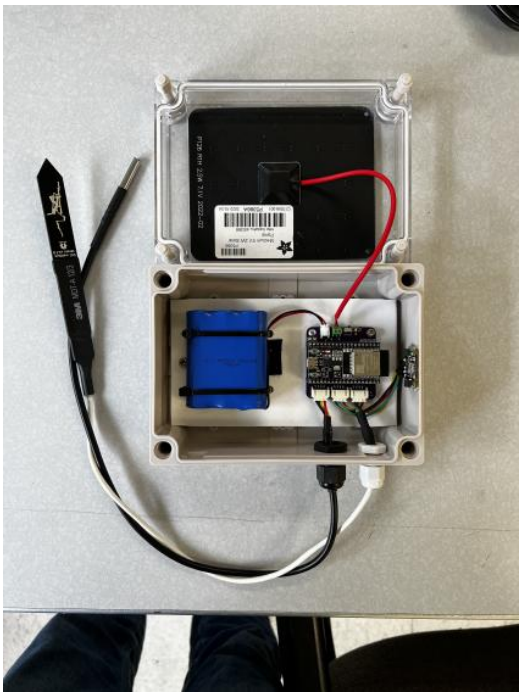
3/08/23	Peter Thompson: Section Created

Section 5: System Verification Evidence

5.1 Universal Constraints

5.1.1. The system may not include a breadboard.

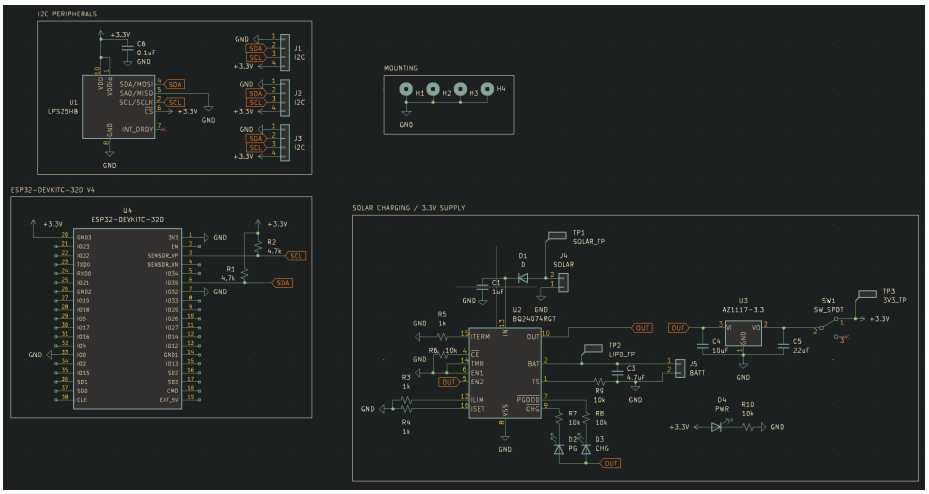
- As it stands, a breadboard is not a component of both the hub and garden node blocks. Both circuits are put together on PCB's.



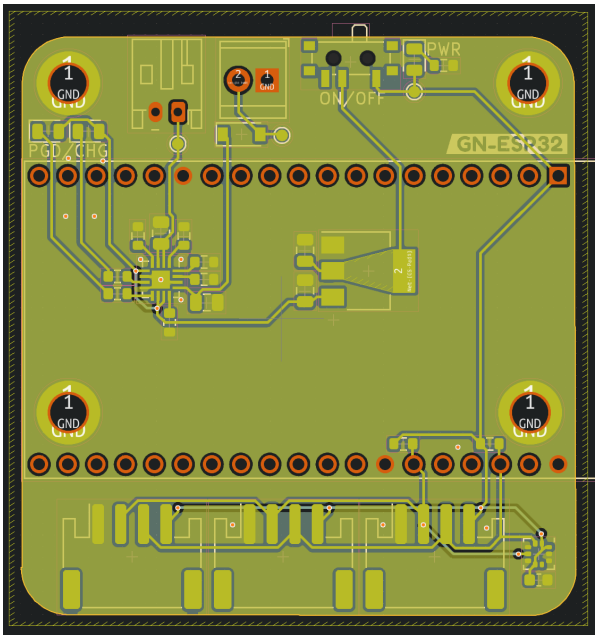
- Figure 5.1.1.a - System image with no breadboard
- Verified on 5/5/23

5.1.2. The final system must contain a student designed PCB.

- The PCB for the Garden Node was designed to connect multiple sensors to a microcontroller.



- Figure 5.1.2.a - PCB schematic
- Verified on 5/7/23
- Figure 5.1.2.b - PCB schematic



- Figure 5.1.2.b - PCB with at least 30 SMD pads (not connector pads)
- Verified on 5/2/23

- Figure 5.1.2.c - PCB assembled with connectors plugged in
- Verified on 5/2/23

5.1.3. All connections to PCBs must use connectors.

- The solar panel connects to the garden node with screw terminals, the lipo with a 2-pin JST-PH connector, and then the three sensors have corresponding 4-Pin JST-PH connectors in order to swap out fault sensors easily.

- Figure 5.1.3.a - PCB assembled with connectors plugged in
- Verified on 5/2/23

5.1.4. All power supplies in the system must be at least 65% efficient.

- The power supply we bought is rated efficiency level VI. Meaning it is at least 81.4% efficient (Web Page: https://www.amazon.com/dp/B08HT1HKJ3?psc=1&ref=ppx_yo2ov_dt_b_product_details)
- The power supply for the garden node is built up from a solar panel, and lipo charging circuit.

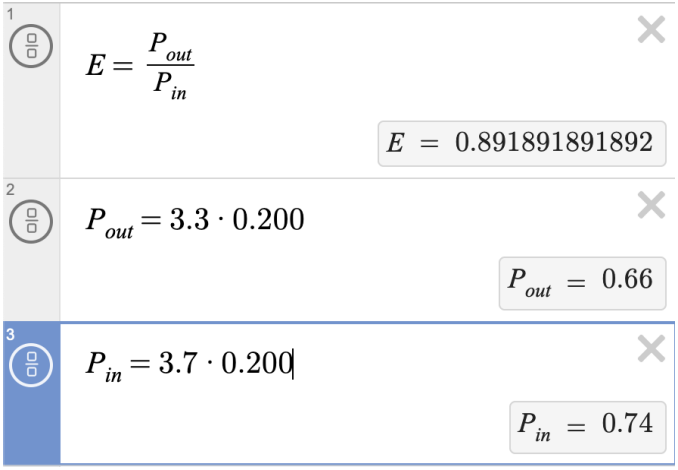


Figure 5.1.4.a - Solar charger efficiency

- Power in: The voltage of the battery and how much it supplies to the system
- Power out: The voltage supplied to the system and the current consumed by the esp32 and sensors
- Efficiency: ~90%
- https://www.adafruit.com/product/5366
- https://media.digikey.com/pdf/Data%20Sheets/Diodes%20PDFs/AZ1117_Rev5.3_Jan2019_DS.pdf

- https://www.digikey.com/en/products/detail/texas-instruments/BQ24074RGTR/2047269

5.1.5. The system may be no more than 50% built from purchased 'modules.'

- When you sum all the components purchased, more than 50% of components are not modules. The microcontroller and sensors are the only modules that have been included on the garden node. All of our code was custom built.
- Total percent of built modules = Built modules count / Total modules count = 20 / (20 + 8) = 71.4%
- Built Modules

5.1.i. Code blocks

5.1.1. MoistureSensorMatterApp GitHub

5.1.a. https://github.com/tobyloki/MoistureSensorMatterApp

5.1.2. MoistureSensorFirmware GitHub

5.1.a. sensor

5.1.i. https://github.com/tobyloki/MoistureSensorFirmware

5.1.b. actuator

5.1.i. https://github.com/tobyloki/MoistureSensorFirmware/tree/actuator

5.1.3. MoistureSensorBackend GitHub

5.1.a. api

5.1.i. https://github.com/tobyloki/MoistureSensorBackend

5.1.b. grpc-client-server

5.1.i. https://github.com/tobyloki/MoistureSensorBackend/tree/grpc-client-server

5.1.c. hub-data-reporter

5.1.i. https://github.com/tobyloki/MoistureSensorBackend/tree/hub-data-reporter

5.1.d. iot-event-handler

5.1.i. https://github.com/tobyloki/MoistureSensorBackend/tree/iot-event-handler

5.1.e. reset-actuator

5.1.i. https://github.com/tobyloki/MoistureSensorBackend/tree/reset-actuator

5.1.f. scheduler

5.1.i. https://github.com/tobyloki/MoistureSensorBackend/tree/scheduler

5.1.4. MoistureSensorHub GitHub

5.1.a. https://github.com/tobyloki/MoistureSensorHub

5.1.5. Hub GUI GitHub

5.1.a. https://github.com/CarsonEhlers/HubGUI

5.1.ii. Garden Node PCB

5.1.1. https://github.com/AineeJames/EcoSense-PCB

5.1.iii. Garden Standoff 3D model

5.1.1. https://drive.google.com/drive/folders/1R2QWAnI-stDA-ehWAaS2HWgdznXrhgBt?usp=sharing

5.1.iv. Hub enclosure 3D models

5.1.1. Base

5.1.a. https://drive.google.com/drive/folders/1R2QWAnI-stDA-ehWAaS2HWgdznXrhgBt?usp=sharing

5.1.2. Top

5.1.a. https://drive.google.com/drive/folders/1R2QWAnI-stDA-ehWAaS2HWgdznXrhgBt?usp=sharing

5.1.v. Air pressure sensing

5.1.1. https://www.st.com/en/mems-and-sensors/lps25h.html

5.1.vi. Temperature / Humidity

5.1.1. https://www.amazon.com/Temperature-Waterproof-Agricultural-Greenhouse-Monitoring/dp/B07X4668VC/ref=sr_1_3?crid=2ZDV8NPS9TUJY&keywords=sht35&qid=1683518121&s=industrial&sprefix=sht35%252Cindustrial%252C202&sr=1-3

5.1.vii. Solar charge circuit

5.1.1.

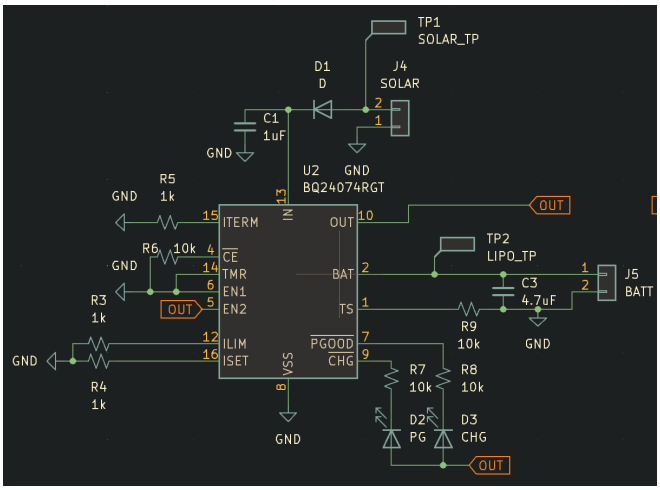


Figure: 5.1.7 Solar Charge Circuit

5.1.viii. Battery regulation circuit

5.1.1.

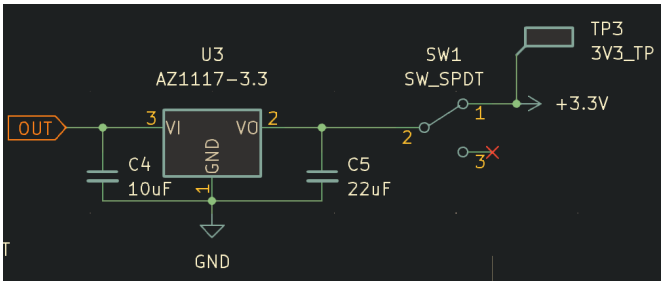


Figure: 5.1.8 Battery Regulation Circuit

- Purchased Modules

5.1.i. ESP32
    5.1.1. https://www.amazon.com/ESP-WROOM-32-Development-Microcontroller-Integrated-Compatible/dp/B08D5ZD528/ref=sr_1_3?keywords=espressif+esp32&qid=1683516485&sr=8-3

5.1.ii. LiPo battery
    5.1.1. https://www.adafruit.com/product/5035

5.1.iii. Light sensor
    5.1.1. https://www.adafruit.com/product/4162

5.1.iv. Soil moisture sensor
    5.1.1. https://www.tindie.com/products/miceuz/i2c-soil-moisture-sensor/

5.1.v. Power Supply
    5.1.1. https://www.amazon.com/dp/B08HT1HKJ3?psc=1&ref=ppx_yo2ov_dt_b_product_details

5.1.vi. Waterproof enclosure
    5.1.1. https://www.adafruit.com/product/905

5.1.vii. Raspberry Pi Model 4 B+
    5.1.1 https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

5.1.viii. 5-inch Capacitive Touch Screen
    5.1.1
https://www.amazon.com/gp/product/B07FDYXPT7/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

5.2 Requirements

5.2.1. Air pressure sensor

5.2.1.1. Project Partner Requirement: The garden node will measure air pressure

5.2.1.2. Engineering Requirement: The system will report pressure to within 25 inHg to 30 inHg

5.2.1.3. Testing Method: Inspection

5.2.1.4. Verification Process:
1. Place garden node pcb in a vacuum sealed Tupperware container

2. Record air pressure and ensure it is around 30inHg

3. Begin pumping out the air in the Tupperware container until the app displays an air pressure of 25inHg
Pass Condition: Sensor is able to read a range of 25-30inHg

5.2.1.5. Testing Evidence:

https://youtu.be/xR65z-S_jkM

- This video shows that the device is able to report the upper and lower bounds of the pressure to the app.

Figure 5.2.1.5: Pressure sensor verification

Date verified: 5/4/2023

5.2.2. Garden Node power

5.2.2.1. Project Partner Requirement: the outdoor nodes should be solar powered

5.2.2.2. Engineering Requirement: The garden node sub-system will operate without being connected to external electricity for 6 months.

5.2.2.3. Testing Method: Analysis

5.2.2.4. Verification Process:
1. Measure the current consumed by the garden node using a multimeter

2. Measure the voltage of the garden nodes power rails using a multimeter

3. Measure the current provided by the solar panel using a multimeter

4. Measure the voltage across the red and black wire of the solar panel using a multimeter
Pass Condition: The power used by the garden node is 3x less than the power supplied from the solar panel (assuming 8 hours of sun each day).

5.2.2.5. Testing Evidence:
https://docs.google.com/document/d/1pkiZyLkx2U-oeAyuiy_DgS2vXXDG_vsKhGRDoLHDCwI/edit?usp=sharing

- This document outlines the measurements and calculations used to determine that the system would be able to operate within the required battery timespan.

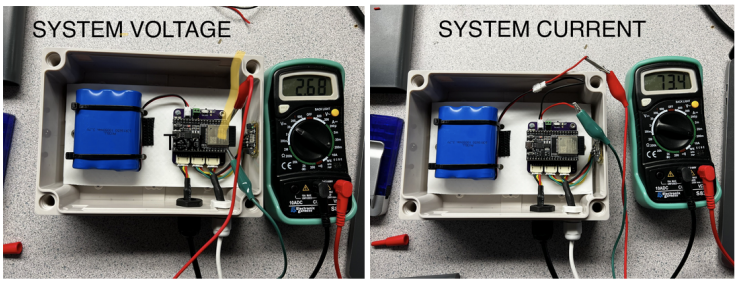


Figure 5.2.2.5: Power Usage verification

Date verified: 5/9/2023

5.2.3. Humidity sensor

5.2.3.1. Project Partner Requirement: The garden node will measure humidity

5.2.3.2. Engineering Requirement: The system will report relative humidity from 20-80% w/ +-5% accuracy

5.2.3.3. Testing Method: Test

5.2.3.4. Verification Process:
1. Place water into a pot

2. Add both the test and SHT35 sensor into the water

3. Ensure that the humidity levels reach 80% or higher and that the SHT35 is within +-5% of the test probe

4. Remove both sensors from the water and dry with a hand towel

5. Place the SHT35 sensor into a bag filled with silica packets and seal it with tape

6. Leave the sensor sealed in the bag overnight (10 hours)

7. Take the SHT35's measurement and ensure that the relative humidity (RH) reaches 20% or lower

Pass Condition: Our system is able to demonstrate a range in recording relative humidity within 20%-80% and be within a +-5% tolerance from the test probe.

5.2.3.5. Testing Evidence:

https://youtu.be/tkSJEPa9Sgo

- This video demonstrates that the relative humidity sensor is able to measure the lower and upper bounds of the air's humidity with the required accuracy.

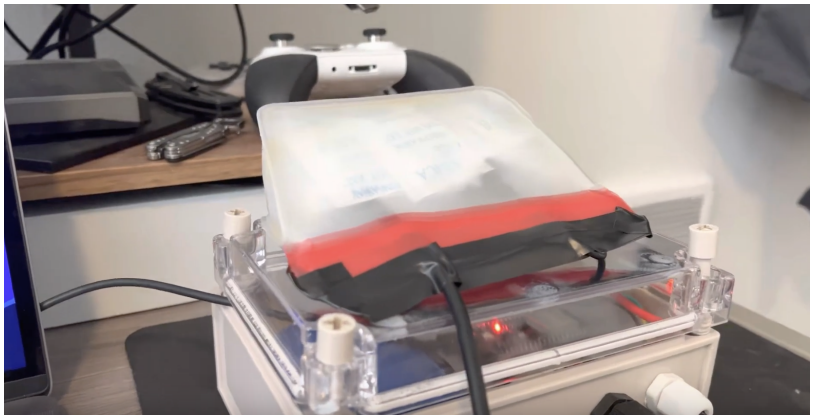


Figure 5.2.3.5: Relative humidity sensor verification

Date verified: 5/3/2023

5.2.4. Soil Moisture sensor

5.2.4.1. Project Partner Requirement: The garden node will measure soil moisture
5.2.4.2. Engineering Requirement: The garden node will measure the soil moisture saturation with a 10% accuracy from fully saturated to unsaturated

5.2.4.3. Testing Method: Inspection

5.2.4.4. Verification Process:
1.) Two soil samples will be provided in order to define fully saturated and dry soil. For each sample, a cup of dry soil will be used. 2.) In the fully saturated sample, water will be added until water begins to puddle on top of the soil. 3.) In order to ensure soil is dry for the second step, a cup of soil will be dried in an oven until the soil temperature reaches 180 degrees F. 4.) Once the two soil samples have been prepared, the soil moisture probe will be placed in the saturated soil first. 5.) The soil capacitance measurement will be recorded, the probe will be removed and dried, and then placed into the dry sample. 6.) Once recording the dry soil measurement, there must be at least a difference of 100 to

determine a change in soil capacitance.
Pass Condition: A difference in 100 between dry and fully saturated soil samples

5.2.4.5. Testing Evidence:
https://www.youtube.com/watch?v=HFhcn9SkiGA&feature=youtu.be

- This video showcases that the soil moisture sensor is able to measure between saturated and unsaturated and within the required accuracy.

Figure 5.2.4.5: Soil moisture verification

Date verified: 3/10/23

5.2.5. Temperature sensors

5.2.5.1. Project Partner Requirement: The garden node will measure temperature

5.2.5.2. Engineering Requirement: The system garden node subsystem will measure temperature to within 2 (F) from 32 (F) to 120 (F)

5.2.5.3. Testing Method: Test

5.2.5.4. Verification Process:
1. Place water, the test temperature probe, and the SHT35 sensor into a pot filled with water

2. Put the pot on the stove and kick on the heat

3. Ensure that the temperature reading for the SHT35 reaches 120degF and is within +-2degF of the test temperature probe

4. Turn off the heat and remove both probes from the water

5. Place both probes into a refrigerator freeze

6. Ensure that the temperature reading from the SHT35 falls at or below the 32degF threshold
Pass Condition: The SHT35 is able to measure a range of 32-120degF and report within +-2degF of the test probe's measurement.

5.2.5.5. Testing Evidence:

https://youtu.be/uGLhJHy7ziA

- This video shows that the temperature sensor is able to measured within the required lower and upper ranges of temperature and measured within the required accuracy specification.

Figure 5.2.5.5: Temperature sensor verification

Date verified: 5/3/23

5.2.6. User Experience

5.2.6.1. Project Partner Requirement: The data should be accessible by phone

5.2.6.2. Engineering Requirement: The system will report information that is indicated as easy to understand by 9 out of 10 users.

5.2.6.3. Testing Method: Inspection

5.2.6.4. Verification Process:
1. I will walk through and give a tour of the app and show how to use/navigate it.

2. I will also show that ten people signed a paper, and at least nine said the app was easy to understand.

Pass Condition: People can see user data stored on the phone and at least 9 out of 10 people said the app was easy to understand.

5.2.6.5. Testing Evidence:

Video Overview:
https://drive.google.com/file/d/1Y2TtWdbfGSEFUGnZbwQ9l2inKi5aRsAZ/view?usp=sharing

- This video covers how user will be able to use the app, view data, connect devices to actuators via integrations, and how to navigate it.

Signatures:
https://drive.google.com/file/d/1iQbdNLaFSOJSi_S-2P9VDHvwCfn4LJhW/view?usp=sharing

- This is a list of signatures gathered form 10 people who all said the app was easy to understand. The process was as follows:
    - I informed them about the app and gave them details as to what it would be used for.
    - I told them to review and play with the app.
    - I then asked them to give a review of the app in terms of easiness to understand. A score of 5 is the best, and a score of 1 is the worst.

Live signature recording:
https://drive.google.com/file/d/1_LTt1PXZHpyCG8zAO5aau6Qx2VJTWL5b/view?usp=sharing

- This is a live recording of the one of the reviewers who explains that the app was easy to understand.

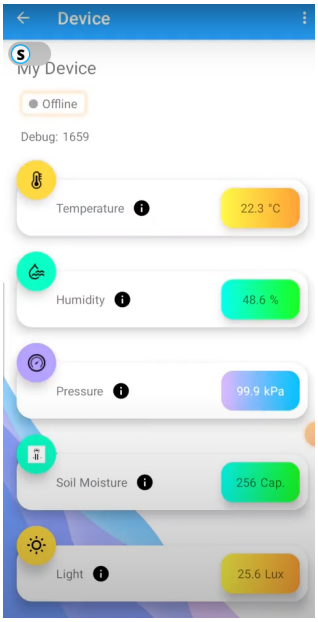


Figure 5.2.6.5: User experience verification

Date verified: 3/11/23

5.2.7. Weather Resistant

5.2.7.1. Project Partner Requirement: The garden node will need to water resistant capabilities

5.2.7.2. Engineering Requirement: The garden node will continue to function after being sprayed with water of at least 1 gallon per minute from all directions for 30 seconds (IPX5).

5.2.7.3. Testing Method: Demonstration

5.2.7.4. Verification Process:
To test that the system is waterproof: 1. A gallon jug of water will be filled halfway. 2. Over a 30 second period the half gallon of water will be poured onto one side of the container. 3. repeat steps one and two for the remaining sides.
Pass Condition: the inside remains dry

5.2.7.5. Testing Evidence:

https://youtu.be/uojGeOc5xzM

- This video shows that after pouring lots of water on the system, the system is still able to function properly.

Figure 5.2.7.5: Weather Resistance Test

Date verified: 5/5/23

5.2.8. Wireless communication

5.2.8.1. Project Partner Requirement: The system must communicate wirelessly

5.2.8.2. Engineering Requirement: Our system will communicate using the Matter protocol

5.2.8.3. Testing Method: Demonstration

5.2.8.4. Verification Process:
1. Open the Android app and navigate to the sensor page

2. Change the value of any of the sensors on the garden node and watch as the value changes on the app

3. Show that the hub can pull data from the garden node via Matter. We can check this is working once the hub uploads the data to the internet.

Pass Condition: Android app can request for sensor data from Matter device and can be viewed on app. Smart hub is able to pull data from garden node via Matter protocol and can be seen from the internet.

5.2.8.5. Testing Evidence:

Overview of everything working together:
https://drive.google.com/file/d/1AEMASsf8jqzZMxpht7SBo3bggmclAm6N/view?usp=sharing

- This is a system overview that explains how data goes from one system to another, basically showcasing the flow of Matter data.

Matter integration proof with Google Home and HomeKit: https://drive.google.com/file/d/1EwiXUOuWAmr1l6lkUdk1Hcu2moFhep0C/view?usp=sharing

- This is proof that our system is indeed using Matter by showing that it can be added to the Google Home and Apple HomeKit system via Matter, as promoted by the Matter alliance.

Figure 5.2.8.5: Wireless communication verification

Date verified: 4/25/23

5.3 References and File Links

- No external reference links applicable

5.4 Revision Table

Date	Revision
3/10/23	Alex, Peter, Aiden: Section 5 created
3/12/23	Alex, Aiden, Carson: Finished section 5.2
5/6/23	Alex: Updated sections 5.1 and 5.2

5/7/23	Alex, Peter, Aiden, Carson: Updated sections 5.1 and 5.2
5/10/23	Alex, Peter, Aiden, Carson: Updated sections 5.1 and 5.2
5/12/23	Alex: Update section 5.2

Section 6: Project Closing

6.1 Future Recommendations

6.1.1 Technical recommendations

- Rebuild the enclosures to take up less space
    - When we started the design for the Enclosures the internal components were not fully known and to provide the space needed for wires we added plenty of space for the unknown. As it is currently, the enclosures have space to be shrunken, and if modifications to the internal components are made the enclosure should be able to reduce the size of the hub by a few square inches [1].
- Replace the Raspberry Pi 4 with a lower power model
    - Swapping a Raspberry Pi 4B hub for a smaller, cheaper, and lower power alternative can provide significant benefits in a smart home ecosystem. A smaller hub can free up space and reduce clutter, while a lower power option can help conserve energy and reduce electricity bills. Additionally, newer smart home hubs are emerging that support Thread and Matter communication, which can provide a standardized platform for interoperability among various smart devices. Some examples of these alternatives include the Raspberry Pi Pico [2], which is a microcontroller-based board that supports both Thread and Matter, as well as the Particle Argon, which is a small, low-cost Wi-Fi and mesh-enabled board that supports Thread and Matter. These alternatives offer the same capabilities as the Raspberry Pi 4B hub but at a fraction of the cost and with a smaller form factor, making them an ideal option for anyone looking to create an energy-efficient and cost-effective smart home ecosystem.
- Build a second sensor node and connect it to the system
    - One of the main highlights of this project is its ability to connect a network of sensor nodes to the hub. With the budget and time constraints for this project we were only able to design and build one sensor node. While the one sensor was enough to show that the system works, it does not show the full capabilities of what our project could do. There are two ways to go about implementing a sensor. The simplest way to show the network of sensors working would be to duplicate the original garden sensor as we

have documented. The other method would be to design a new sensor system and integrate it into the original system, like making a node for indoor use with a separate set of sensors [3].

- Upgrade the system to use Thread with Matter
    - Using Thread with Matter can help conserve power and improve the performance of smart home and IoT devices. Thread is a low-power wireless protocol that uses significantly less energy than Wi-Fi, while Matter provides a standardized platform for interoperability among various smart devices. By using Thread with Matter, devices can communicate directly with each other without relying on a central hub, resulting in less energy consumption and greater efficiency. In addition, using a mesh network topology with Thread and Matter can extend the range of the network and provide a self-healing, reliable system for smart homes and IoT environments. One way to get started with Thread over Matter is to use the Silicon Labs Thread Matter development kit [4]. Once the dev kit has been obtained, open Simplicity Studio and create a new Matter project to get started.
- IEEE Links

[1] A. Industries, “Flanged weatherproof enclosure with PG-7 cable glands,” adafruit industries blog RSS, https://www.adafruit.com/product/3931 (accessed May 12, 2023).
[2] Raspberry Pi, “Buy A raspberry pi pico,” Raspberry Pi, https://www.raspberrypi.com/products/raspberry-pi-pico/ (accessed May 12, 2023).
[3] P. Thompson, A. Olsen, A. Feng, and C. Ehlers, “Project Document Group #27,” Google Docs, https://docs.google.com/document/d/1MwY7L9va99qT6WhpNn0tTr2XKgXAYbFziT8sJkG1rzc/edit# (accessed May 12, 2023).
[4] “EFR32XG24 Explorer Kit - Silicon Labs,” EFR32xG24 Explorer Kit - Silicon Labs, https://www.silabs.com/development-tools/wireless/efr32xg24-explorer-kit?tab=overview (accessed May 12, 2023).

6.1.2 Global impact recommendations

- Out of all the components that have been used in our project, the Lithium Ion battery has the potential to cause the most harm. To potentially move away from using a lithium ion battery, the benefits of different types of batteries can be used such as Nickel Metal Hydride (NiMH), Nickel Cadmium (NiCad), and Lithium Iron Phosphate (LiFePO4, or LFP) batteries [1].
- Instead of sending the sensor measurement devices data to the cloud, it could instead be stored on the central hub itself. Not only would this provide better data security, but it would also improve the speed at which the data is loaded by the hub since it wouldn't need to pull the data from the cloud. This could be achieved

by storing the data into a database that is “self-hosted” by the central hub. An alternative could be a simple csv file that is stored on the sd card of the central hub’s raspberry pi [2].

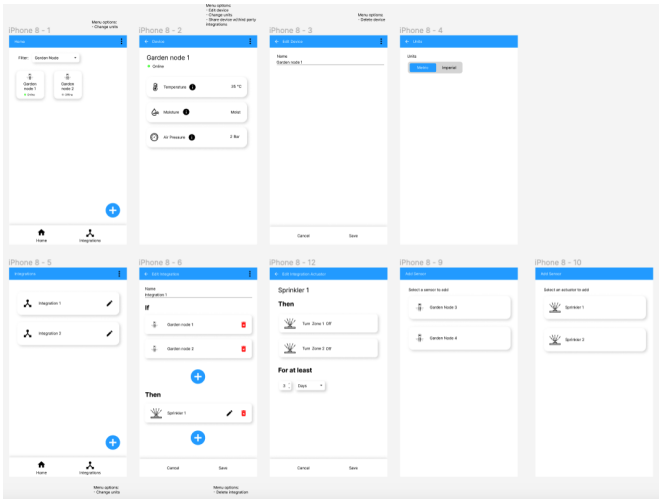
- IEEE Links:
    - [1] H. V. Venkatasetty and Y. U. Jeong, "Recent advances in lithium-ion and lithium-polymer batteries," *Seventeenth Annual Battery Conference on Applications and Advances. Proceedings of Conference (Cat. No.02TH8576)*, Long Beach, CA, USA, 2002, pp. 173-178, doi: 10.1109/BCAA.2002.986391.
    - [2] Sharma, Arnov. “Raspberry Pi - Data Logging : 5 Steps (with Pictures).” *Instructables*, https://www.instructables.com/Raspberry-Pi-Data-Logging/. Accessed 12 May 2023.

6.1.3 Teamwork recommendations

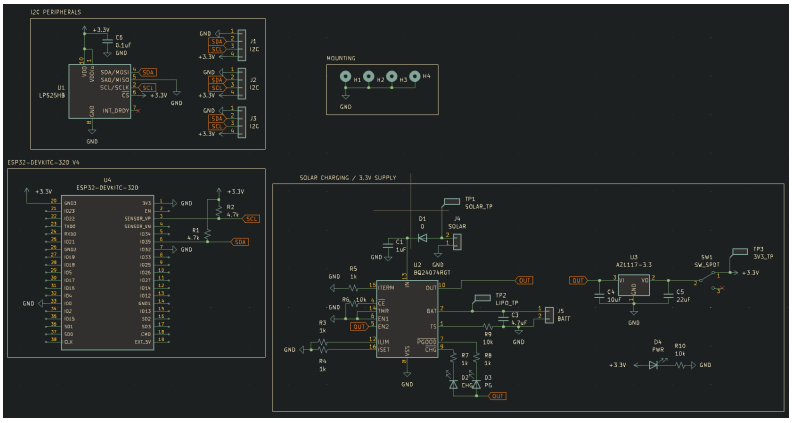
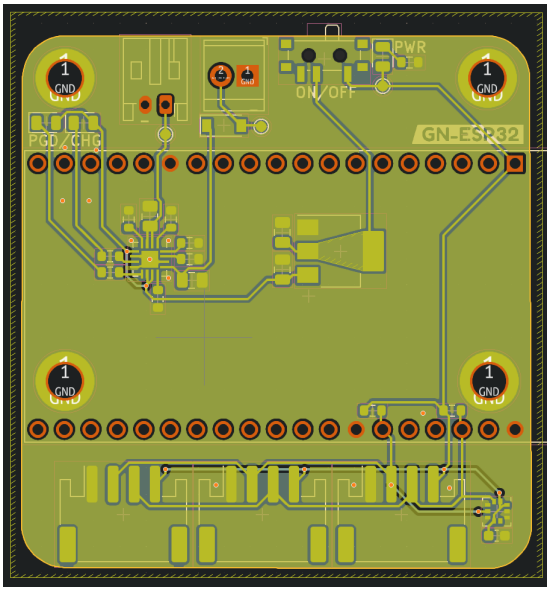
- Our team is promoting two teamwork recommendations based on our experience working on the project. The first recommendation emphasizes the importance of effective project time management in order to avoid last-minute scrambling to meet deadlines. To achieve this, we suggest setting up a designated project space where all team members can report their progress and see upcoming tasks that are required to be completed. This will help to ensure that all team members are aware of their responsibilities, deadlines, and the progress of the project as a whole. Platforms such as Slack or Microsoft Teams can be used to create a shared space where team members can communicate, collaborate, and stay updated on the project's progress. You can easily set up a platform like Microsoft Teams by looking into resources on YouTube [1] or the Microsoft docs. This recommendation is supported by a study on the benefits of using project management software for team collaboration [2].
- The second recommendation emphasizes the need for sufficient time to improve the quality and depth of designs. To achieve this, we suggest that future teams should meet twice a week outside of designated class time to keep up with future deadlines. This approach will give team members more time to work on the project, allowing them to refine their designs and produce higher quality work. By meeting outside of class time, team members can also focus exclusively on the project without being distracted by other academic commitments. This recommendation is supported by an article on the importance of taking the time to refine designs [3].
- IEEE Links:
    - [1] “🏫 Microsoft teams tutorial in 10 min,” YouTube, https://www.youtube.com/watch?v=VDDPoYOQYfM&ab_channel=KevinStratvert (accessed May 12, 2023).
    - [2] Foley Marketing Advisors, “Major Advantages and disadvantages of using project management software,” Foley Marketing Advisors,

https://foleymarketingadvisors.com/2021/11/04/major-advantages-and-disadvantages-of-using-project-management-software (accessed May 12, 2023).
○ [1] S. Mabe, “Hands-on with design thinking: Refine phase,” Ronsen Consulting, LLC, https://www.ronsenconsulting.com/blog/hands-on-with-design-thinking-refine-phase (accessed May 12, 2023).

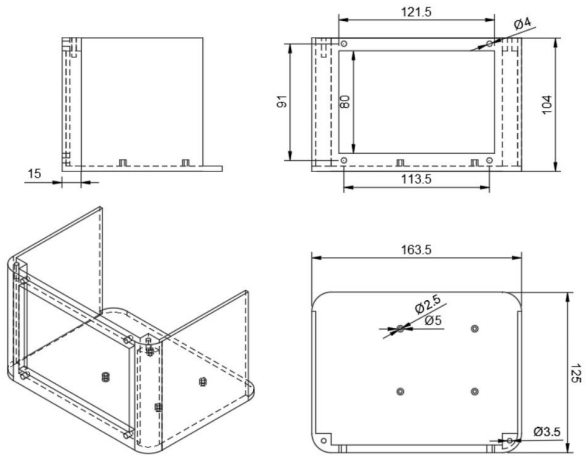
6.2 Project Artifact Summary with Links

- Code
  - MoistureSensorMatterApp GitHub (code)
    - https://github.com/tobyloki/MoistureSensorMatterApp
    - 
      - Description: Android app that can add and manage Matter devices. It is the main entrypoint for users to view sensor data on our EcoSense product.
  - MoistureSensorFirmware GitHub (code)
    - sensor
      - https://github.com/tobyloki/MoistureSensorFirmware
        - Description: This is the firmware for the EcoSense that reads all the sensor measurements and reports them to any Matter controller such as an Android app or the Hub.
    - actuator
      - https://github.com/tobyloki/MoistureSensorFirmware/tree/actuator
        - Description: This is the firmware for actuators such as water valves, hoses, and sprinklers that can be turned on and off via the Matter protocol.
  - MoistureSensorBackend GitHub (code)
    - api
      - https://github.com/tobyloki/MoistureSensorBackend

            - Description: This is the endpoint that the Hub will report all connected sensor data to.
        - grpc-client-server
            - https://github.com/tobyloki/MoistureSensorBackend/tree/grpc-client-server
                - Description: This is the code that notifies the Hub which actuators should be turned on or off via the gRPC API protocol.
        - hub-data-reporter
            - https://github.com/tobyloki/MoistureSensorBackend/tree/hub-data-reporter
                - Description: This is the code that runs on the Hub that periodically polls data from all sensors and reports them to the api endpoint..
        - iot-event-handler
            - https://github.com/tobyloki/MoistureSensorBackend/tree/iot-event-handler
                - Description: This is an IoT event handler that responds to events that have been processed by the IoT events service. If the event is not “Normal”, then it will request to the Hub that all actuators associated with the sensor be turned off, send a notification to the user’s phone, and send out a timer request to check if the actuators should resume normal operations.
        - reset-actuator
            - https://github.com/tobyloki/MoistureSensorBackend/tree/reset-actuator
                - Description: This code is executed once the timer expires. It then makes a request to the Hub to turn off all actuators associated with the sensor.
        - scheduler
            - https://github.com/tobyloki/MoistureSensorBackend/tree/scheduler
                - Description: This just runs a timer that waits an user specified time before making a request to reset-actuator to check whether the sensor has returned back to “Normal” state.
    - MoistureSensorHub GitHub (code)
        - https://github.com/tobyloki/MoistureSensorHub
            - Description: This code runs on the Hub and is responsible for all of the Matter related service calls that run behind the GUI.
- Garden Node PCB
    - https://github.com/AineeJames/EcoSense-PCB

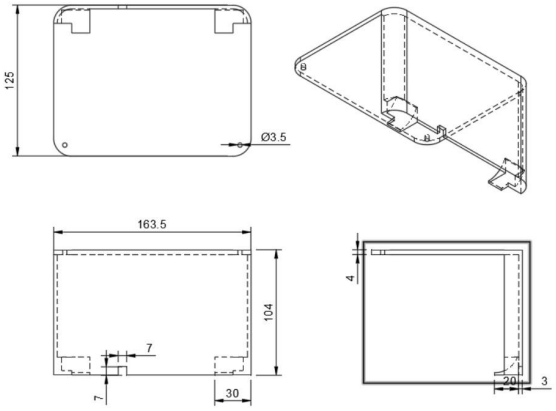
- 
    - Description: Schematic Capture for Garden Node
- 
    - Description: PCB Layout for Garden Node

- 3D CAD
    - Garden Standoff 3D model
        - https://drive.google.com/drive/folders/1R2QWAnI-stDA-ehWAaS2HWgdznXrhgBt?usp=sharing
            - Description: This is the garden node standoff made to help mount and hold the electronics off of the base to protect them from water.
    - Hub enclosure 3D models
        - Base
            - https://drive.google.com/drive/folders/1R2QWAnI-stDA-ehWAaS2HWgdznXrhgBt?usp=sharing



- Description: This holds the touch screen and provides mounting for the Raspberry pi with ample space to connect the wires

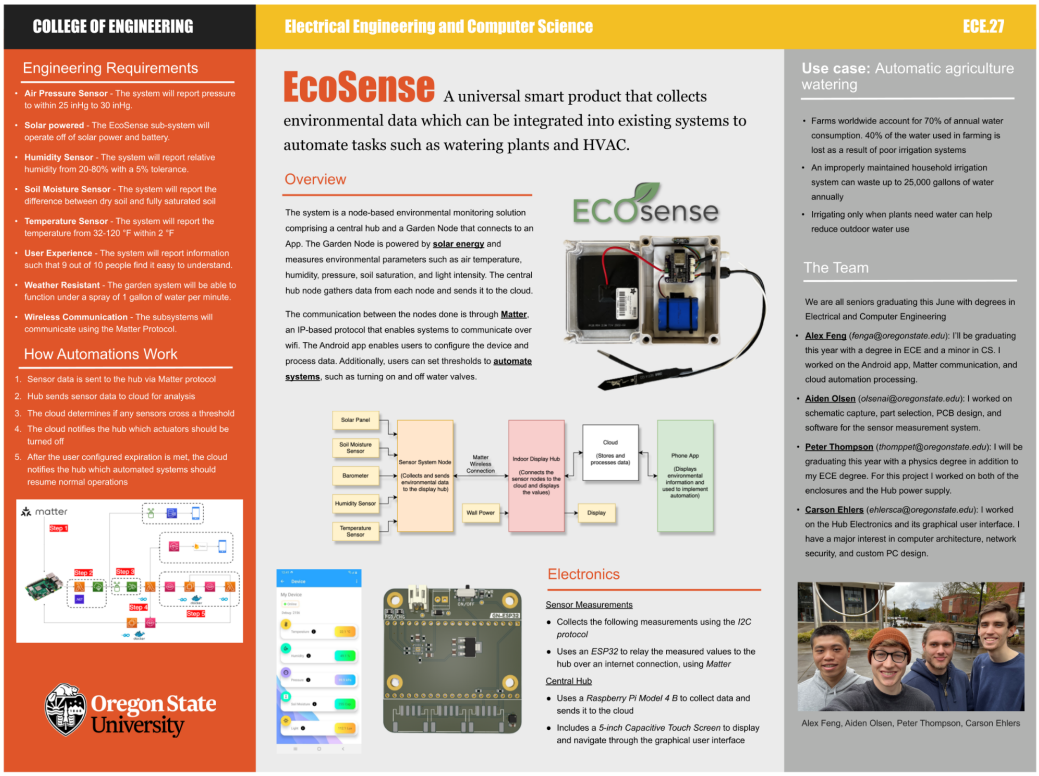
- Top
  - https://drive.google.com/drive/folders/1R2QWAnI-stDA-ehWAaS2HWgdznXrhgBt?usp=sharing



- Description: This is the top of the hub

6.3 Presentation Materials

- System Overview
  - https://drive.google.com/file/d/1AEMASsf8jqzZMxpht7SBo3bggmclAm6N/view?usp=share_link
    - Description: This video explains how the whole system works.
- Poster



- Project Showcase link
  - https://eecs.engineering.oregonstate.edu/project-showcase/projects/?id=HwFIcR4aahcm5Yrt