CYBER S.H.I.E.L.D.

Smart Home Integrated Environment
for Learning and Defense

120-Page Document
EEL4914 - Senior Design II
Group 12

University of Central Florida
Department of Electrical and Computer Engineering

Nicole Parker, EE
Adam Bouchama, EE
Jordan Threlfall, CpE
Karan Patel, CpE

Sponsored by UCF Digital Grid Laboratory
Reviewers: Dr. Wei Sun, Dr. Qun Zhou Sun, Dr. Mahzar Ali
Mentor: Chung Yong Chan

Table of Contents

1. Executive Summary

As the grid continues to evolve and more renewable energy sources are being integrated, there has been an increase in the vulnerabilities present in our power systems. Now, more than ever, houses are installing renewable energy systems and generation of energy is being integrated at the grid edge. The grid edge is defined by the DOE as “the boundary zone where the utility ends and customer premises equipment (CPE) starts; it begins at the meter interface (the utility demarcation point) and contains all equipment, software solutions, and controls owned by the customer”[1]. Previously, it was common for a centralized power plant to be the sole source of energy generation with the energy generated being sent to loads through transmission and distribution lines. This centralized power plant would have multiple layers of security and appropriate methods in place to prevent cyber attacks. It was clear that it would be crucial to lessen the vulnerabilities present at a location with high levels of energy generation and with impacts to large populations of people. However, as renewable energy sources such as solar became more popular for residential areas, security was not considered at the same level as before. It was seen that a cyber attack on one residential PV panel had the potential to only harm one household. However, as the grid is becoming more connected and smart devices are being incorporated throughout homes, cities, and the grid, there is evidence of the importance of cyber security to protect each individual device. Access to just one smart device can open the door to connected smart devices becoming vulnerable. With the changing energy landscape that the world is now seeing, it is crucial for research to be done on how to smoothly modernize the grid.

Cyber S.H.I.E.L.D. is a project that analyzes these issues and presents opportunities for education on renewable energy integration at the grid edge. This project will create a smart home model that functions in a way that emulates a real smart home. The model will make use of an energy management system (EMS) to select appropriate sources and loads. Selections will be performed by the EMS or manually by the user. A touch screen will be one of the key features of this project as it allows the selection of test cases for the project. It will provide an educational experience for students interested in learning about the evolution of our power systems and the function of a smart home. In order to test the security of the smart home, measurements will be collected at the model and will be sent to the OPAL-RT. The OPAL-RT simulator works to model the grid in real time. This will allow the staging of attacks on the model to observe their implications. Overall, this project is crucial to demonstrate new developments to our power systems and how one can work to detect and prevent cyber attacks at the level of the smart home. This project is an excellent way to provide an educational overview of renewable energy integration at the grid edge and how we can make sure the modernized grid is secure and reliable.

2. Project Description

2.1 Motivation & Background

From its development back in the early 1880s, the U.S. electric grid has been constantly growing and evolving. Now, more than ever, the grid is seeing changes that may have never been imagined. Many of these changes are inspired by the expansion of the electric power system to meet growing demands for energy. Other changes to the grid are motivated by the hope to achieve a net-zero future. In any case, the U.S. electric grid is being modernized to develop a smarter grid that allows for larger amounts of electricity generation and longer transmission distances. Two of the main forces driving this modernization include the expansion of renewables and a shift towards a connected and decentralized grid. The goal is to create an evolved grid that is flexible to integrate distributed energy resources (DER), accommodate the two-way flow of electricity and information for better power management, and provide protection against physical and cyber risks.

Unfortunately, with the changes towards grid modernization comes an increasing risk for cyber attacks. These attacks are able to disrupt the grid, damage costly equipment, and even threaten human life and safety. There are many methods of employing a cyber attack. For example, there are a growing amount of smart microgrids in the U.S. today. These microgrids are small-scale power supply networks that enable local power generation for local loads. They can be created by installing grid-edge technologies such as solar panels, energy storage systems, smart inverters, smart appliances, and electric vehicles charging stations. The increased interconnectedness between critical operational technology and these grid edge devices leads to a higher complexity and risk for cyber attacks unless proper security is built in. A hacker can employ a multitude of attacks in order to violate the confidentiality, integrity, or availability of the network. They can also exploit vulnerabilities of the network to gain access to private information of consumers or illegal access to nodes.

Due to this, it is crucial for engineers to develop a method to effectively detect cyber attacks for the U.S. to continue modernizing its grid. Once these algorithms are developed, they must be tested to ensure that they are able to function in real time scenarios. While one cannot test these algorithms on the physical grid itself, engineers are able to use real time simulators such as the OPAL-RT to model a grid. These simulators allow cyber attack detection and prevention algorithms to be effectively tested. Furthermore, through simulations such as the OPAL-RT, students can learn more about how the grid functions and the necessity of security in our power systems as they continue to be modernized. This project is set up to be a physical testbench of a cyber attack detection and prevention algorithm as well as an educational outlet for the complexity and evolution of the U.S. electric power system.

2.2 Review of Prior Related Work

In preparing for this project, we researched work online to have an understanding of how to complete our objectives. After in-depth research, we came across a great example of a solar metering project. This was found on the website "Open Green Energy" and the project was titled "DIY Solar Panel Monitoring System" [2]. This project detailed why one needs a solar panel monitoring system, the materials one needs to build the project, how it works, the schematic diagram, and setting up the circuit. From here it showed the process of measuring current and voltage from the solar panel. This prior work was extremely helpful in explaining how to create a metering PCB and how it works. We plan to use this open source schematic as a reference for our project. In this, we can successfully design a solar voltage/current metering PCB and a battery voltage/current and state of charge metering PCB.

In addition to this work, we were able to find a video discussing a project called "Smart Hybrid Energy Management System Using Arduino" [3]. In this, the creator of the video discussed a flowchart used to create the energy management system and its implementation into the project. This project made use of three sources: solar, battery, and the grid. It also classified loads into three separate categories: high priority, medium priority, and low priority. These loads would be on or off depending on the state of the battery or if it was during peak/off-peak hours. An arduino was used to implement the energy management system. Relays were controlled by the arduino to connect and disconnect the different sources and loads. The sources and loads could be selected manually or were automatically selected by the energy management system depending on the constraints. High, medium, and low priority loads were represented by three light bulbs. Overall, this project perfectly demonstrated a project that implemented an energy management system to select energy sources and loads. We plan to use ideas from this project to apply on our own.

Lastly, we reviewed a Senior Design project from a few years ago that created a project with a few similar characteristics. This project was titled "Power Systems Knowledge Hub" [4]. For example, this Senior Design project made use of a solar panel to supply energy to a load. Measurements were collected from the panel and loads and sent to be displayed on a touch screen. In addition to this, the measurements were sent to the OPAL-RT simulator in the Siemens Digital Grid Lab. Reviewing this past work allowed us to get a better understanding of what communication protocols could be used to connect the devices together. This project also gave us a baseline on what Raspberry Pi would be good to use and how to connect it to the metering hardware. Overall, this project gave us a good baseline to go off of for our project.

2.3 Overview of Project Goals

As the U.S. electric grid continues to evolve, there are new challenges arising for the protection of its resilience and reliability. Increased use of grid-edge devices, such as

solar panels, inverters, and batteries, has caused the distribution system to become more complex to manage. The UCF Digital Grid Laboratory is currently working on researching this complexity and how to ensure that our power systems are protected from cyber threats. For this reason, this project focuses on the development of a model smart home that functions as a high fidelity testbench for an attack detection and prevention algorithm developed in the UCF Digital Grid Laboratory.

Shown in Table 1 were the three levels of goals for this project. Our team planned to conclude the project by providing a functioning cybersecurity testbed for the UCF Digital Grid Laboratory. This testbench served as an educational experience for students wanting to learn more about the modernization of power and energy as well as the functioning of a smart home with a hybrid-solar system.

Table 1: Project Goals Overview

Goal Type	Description
Basic	Create a model smart home with a functioning Energy Management System. A touch screen used to interact with the model. This serves as an educational experience to demonstrate knowledge on power systems.
Advanced	Model smart home implemented and updated to account for a hybrid solar system. Simulated cyber attacks are employed on the model smart home and attack detection and mitigation algorithms are tested.
Stretch	Model smart home completed with hybrid solar system and educational touch screen. Loads are categorized into three levels of priority: low, medium, high.

2.4 Objectives

In order to more accurately test the algorithm, the model smart home functioned using a hybrid solar system. A solar panel was used to provide energy to the loads within the model smart home. There was a battery used for energy storage. This battery was used in the case where the solar panel was not generating energy. In the model, multiple loads demonstrated that the system was functioning correctly. The more accurately the model smart home resembled an actual smart home, the more accurate the test results. Data collected from meters on the model was sent to the OPAL-RT simulation in the Digital Grid Lab. This data was used to employ multiple forms of cyber attacks. When an attack was detected, the appropriate measures should be taken for the event to be isolated. For example, the power supplying the loads was disconnected to ensure the safety of the smart home.

The model smart home had a functioning energy management system (EMS) in order to appropriately distribute and store energy. The EMS was able to account for instances in which there was not enough power being generated by the solar panels and efficiently switch to an alternative power source, such as a DC battery or AC power source. If there was more power being generated than absorbed by the loads, then the EMS would allow the charging of the DC battery. A digital touch screen was utilized to encourage interaction with the project. Users were able to select areas of the model to see the voltage, current, state of charge (SoC), and power absorption or supply.

2.5 Project Features and Functionalities

This project was made up of several key features. To begin, the project involved having a physical model of a home made out of wood. The front of the model was opened so that the user can see inside to monitor the function of the DC and AC loads. Mounted on the roof of the model were solar panels. The solar panels were connected to a charge controller that was connected to a battery. The DC load port on the charge controller pulled from the battery. In addition to this, the battery was connected to a DC/AC inverter. The DC/AC inverter was connected to an AC load. In a scenario where there was not enough sunlight or battery charge, there was a wall outlet connection to represent a “grid tied” system. At the battery, solar panel, and outlet connection there were meters to collect voltage, current, and state of charge measurements. There was also metering at the loads to collect voltage and current measurements. The model was able to intelligently select what source and load is appropriate for the conditions at the time using a programmed energy management system. To allow the selection of the source and load, relays were utilized. A touch screen was used to show all of the measurements and allow for manual selection of the source and load by the user. Measurements were sent to the OPAL-RT simulator to be used in the RT-Lab model for testing of the UCF Digital Grid Lab’s cyber attack detection algorithm.

The research group working with the UCF Digital Grid Lab listed the required functionalities of the model. As expressed previously, the model allowed for the accurate collection of voltage, current, and state of charge measurements. In addition to this, the measurements collected were communicated with the OPAL-RT simulator to ensure appropriate testing of the cyber attack detection algorithm. Beyond this, we saw there was a need in the market for an educational smart home model. This would allow students to get a better understanding on how smart homes functioned and the cyber risks associated with grid modernization.

2.6 Requirement Specifications

The requirements of this project addressed two main areas of focus:
1) Creation of an educational environment for students to learn more about the technologies used in grid modernization (such as microgrids, smart homes, and security resilience)

2) Building of a cyber-physical system (CPS) testbench in order to test the cyber attack detection algorithm developed in the Siemens Digital Grid Lab at UCF

In order to accomplish the requirements and specifications for these two areas of focus, the team designed a functioning model smart home with a realistic energy management system. This EMS functioned the way that it would in a realistic smart home. The user should also be able to select what source (DC or AC) is supplying the load. In addition to this, the user should be able to select what load is on (DC and/or AC). Each source would be metered. The measurements collected should be stored for use in the EMS and the OPAL-RT Simulation. In the case that a cyber attack was launched, the model should be able to notify the user of a successful or unsuccessful detection of the attack.

The specific engineering requirements and specifications for this project were detailed by the Digital Grid Laboratory at the University of Central Florida. These requirement specifications along with the engineering and marketing requirements can be found in Table 2 on the next page.

Table 2: Requirement Specifications

Touch Screen & Touch Screen Platform			
Parameter	**Specification**	**Engineering Requirement**	**Marketing Requirement**
Dimension	34.06” x 56” x 34.75”	The touch screen platform must be the appropriate size for the weight, length, and width of the screen	11, 12
Weight	At Most 100 lbs	The weight of the touch screen must be suitable for transportation	11, 12
Touch Screen Response Time	At Most 5 Seconds	Once a setting has been selected on the touchscreen, the model must reflect that selection of a source or load within five seconds	7
Number of Touch Points	At Least 2	The touchscreen should be multi touch capable and the total number of touch points should be at least 2	7

Number of Available Measurements	At Least 7	Must display V, I, P measurements of sources on the touch screen	1
Communication with Model	Wired Communication with USB	The model must be able to accurately communicate source measurements to be displayed on the touch screen	5
User Interface Capability	Allows Selection of AC and DC Loads, AC and DC Sources	UI Will Allow Selection of DC and AC loads depending on PV panel and battery SoC; UI Will Allow Selection of Sources (DC or AC)	8, 7
Response to Cyber Attack Detection	Shows Warning on Touch Screen	Following the launch of a cyber attack on the OPAL-RT, the touch screen should send an alert and indicate if attack was detected	2
Accuracy of Metered Data	90% Accurate	Data collected by PCB and AC meter must match the realistic V and I measurements and be displayed on touch screen	5
Smart Home Model			
Parameter	**Specification**	**Engineering Requirement**	**Marketing Require-ment**
Dimension	34” x 34.5” x 83”	The smart home model must be the appropriate size to support the weight, length, and width of the PV Panel, Battery, and Inverter	11, 12
Weight	Up to 300 lbs	The weight of the model must be suitable for transportation	11, 12
Battery Voltage	12V	Battery must meet requirements of PV and supply loads when PV is not producing	9

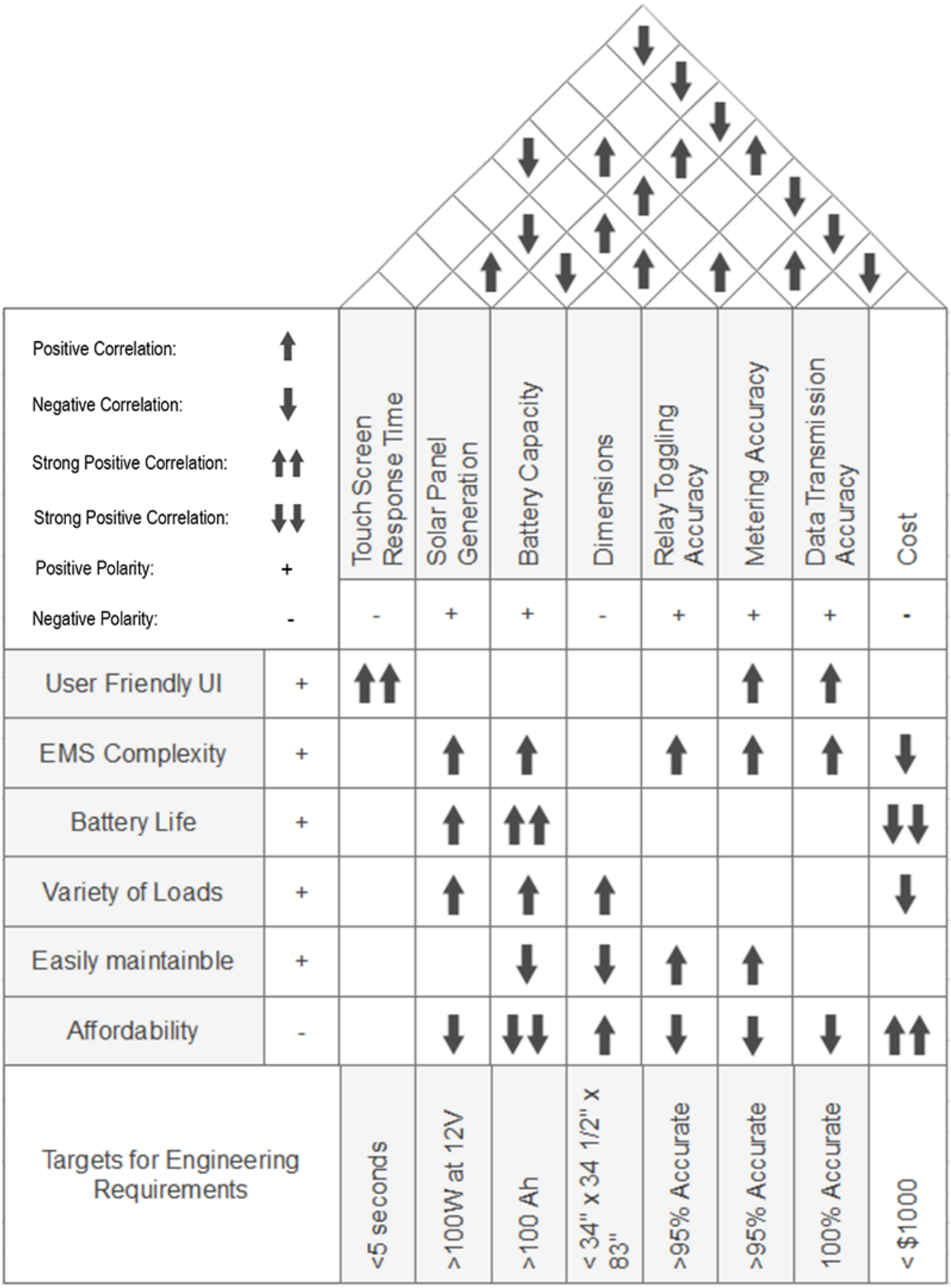
Battery Capacity	100Ah	Battery must meet the current requirements of the PV panel	9
Battery Lifespan	At Least 5 Years	Battery must function successfully for at least five years	9
PV Panel Voltage	12V	PV panel must meet the voltage requirements of the system and provide power to AC/DC loads	9
PV Panel Power	100W	PV panel must meet the power requirements of the system and provide power to AC/DC loads	9
Relay Module Operating Voltage	12V	Relays must meet the voltage requirements of the system and enable connection or disconnection of sources/loads	3
Relay Module Maximum Current	10A	Relays must meet the current requirements of the system and enable connection or disconnection of sources/loads	3
Relay Toggling Accuracy	Equivalent or Above 95% Accurate	Relays must function with over 95% accuracy to switch on and off; Selection of sources and loads	3, 8
Inverter Voltage	12V	Inverter must meet voltage requirements of the system and convert DC/AC	3, 6
Inverter Output Power	At Least 100W	Inverter must meet the power requirements of the system and convert DC/AC	3, 6
Charge Controller Voltage	12V	Charge controller must meet the voltage requirements of the system and allow safe connection from the PV panel to the battery	3, 9

Charge Controller Current	10A	Charge controller must meet the current requirements of the system and allow for connection from the PV panel to the battery	3, 9
Source Types	DC and AC	DC (solar, battery), AC (wall outlet connection, inverter output from battery)	8
Load Types	DC and AC	There must be both DC and AC loads	6, 8
Load Power Consumption	100W	The system must meet load power consumption requirements to function properly	9
MPU	Capable of connecting to Touch Screen and OPAL-RT	MPU Must Receive Data from meters and Communicate with OPAL-RT/Touchscreen	4, 5
Energy Management System	Allow for Appropriate Selection DC, AC Source	EMS must intelligently select the appropriate source and loads for the state of the system	3
Communication with OPAL-RT	Voltage, Current Measurements Sent to OPAL-RT	Measurements collected in the model must be communicated with the OPAL-RT	4, 5
Source Measurements	Voltage, Current, SoC Measurements	There must be a collection of V, I, and P measurements from solar, battery, AC connection; this is sent to MPU	5
Load Measurements	Voltage, Current Measurements	There must be a collection of V and I measurements from DC loads and AC loads and this is send to MPU	5
Metering Accuracy	Equivalent or Above 95% Accurate	Sources and loads measurements must be accurate to reflect real time system conditions	4

Cells Highlighted in Yellow Will Be Shown in the Demo

Marketing Requirements
1. The system should function as an educational tool for students to learn more about the technologies used in grid modernization 2. The system should serve as a testbench to test the cyber attack detection algorithm developed in the Siemens Digital Grid Lab 3. An EMS should be designed to intelligently toggle sources and loads depending on the system state 4. System should communicate metering information to the OPAL-RT simulator 5. Data should be processed and transmitted between devices quickly 6. The model smart home should have a variety of loads to simulate a real smart home. 7. The system's UI should be user friendly 8. Users should be able to choose sources of power and loads 9. The system should be able to meet load requirements using appropriate energy source 10. The system should minimize cost 11. The system should be designed to ensure maintenance is efficient and straightforward 12. The system should be easily transported

2.7 House of Quality



2.8 Prototype Illustrations

In order to envision what the final project will look like, we designed a prototype for the physical model and a prototype for the user interface. To allow for a seamless educational experience, we wanted the physical model to have an open roof in order to see the loads inside. This model would be constructed using wood. It needed to be sturdy enough to hold the PV panel on the roof. The inverter and battery could be seen on the outside of the house with the AC and DC loads inside.

The prototype illustration of the model can be seen below.

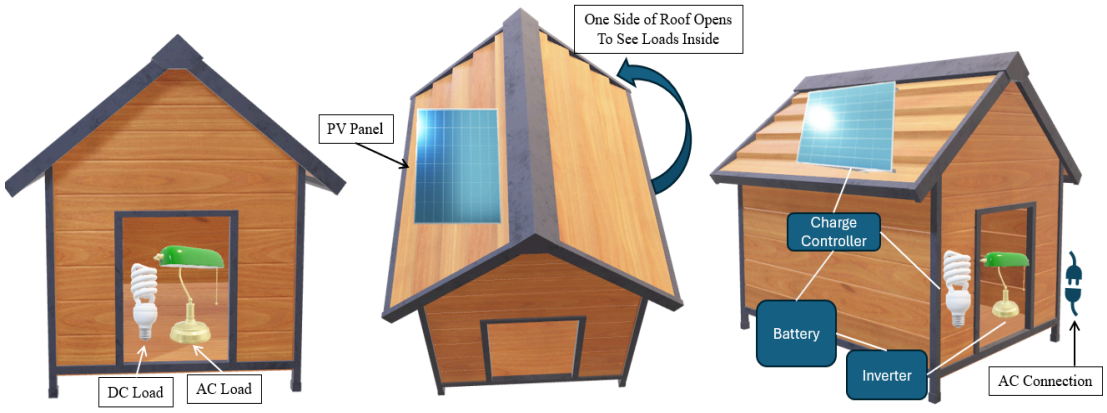


Figure 1: Smart Home Prototype Illustration

Furthermore, the prototype illustration for the user interface can be seen in the figure below. This figure shows off a prototype of the model home we want to integrate in our UI application. In future iterations, we will continue building the house to meet the application's requirements.

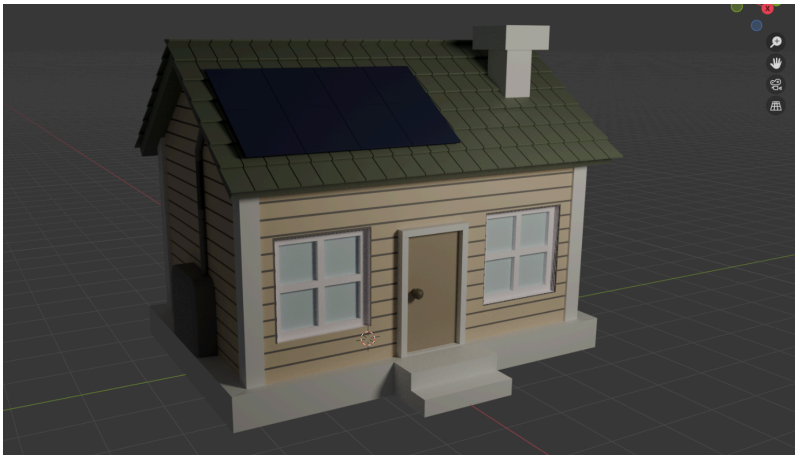


Figure 2: User Interface Prototype Illustration

In Figures 3, 4 and 5 we see a prototype of the various screens when there is a cyber attack and successful/unsuccessful attack detection. One of the major goals of this project was the ability to utilize the model smart home to test various cyber attack detection algorithms. We worked closely with the research lab to ensure that we could integrate their test bed into our system and UI application.

Within the application, we want to be able to display a notification of when a cyber attack was launched. We then wanted to prompt the user and ask if they wanted to test the algorithm. After they deployed the algorithm, we saw if it was successful in the detection of the attack. Figure 4 and Figure 5 show the successful / unsuccessful screens.

On the hardware side, we configured the buttons to send commands to the raspberry pi to interact with OPAL-RT and integrate the algorithms.

Figure 3: User Interface Prototype - Cyber Attack Warning

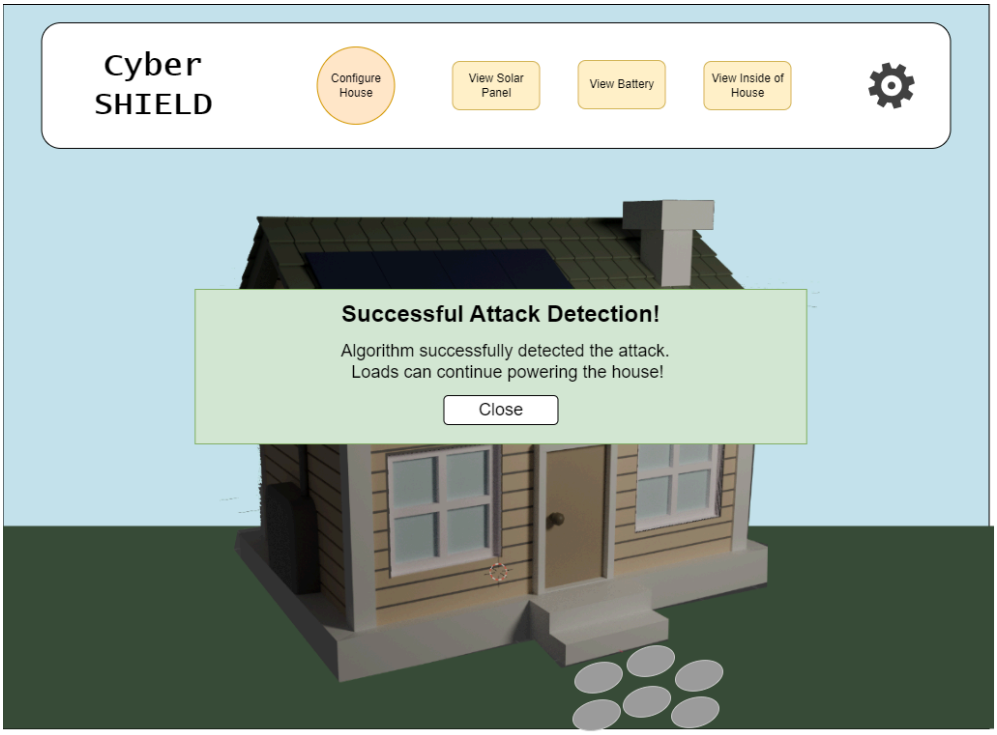


Figure 4: User Interface Prototype - Cyber Attack Successful Detection

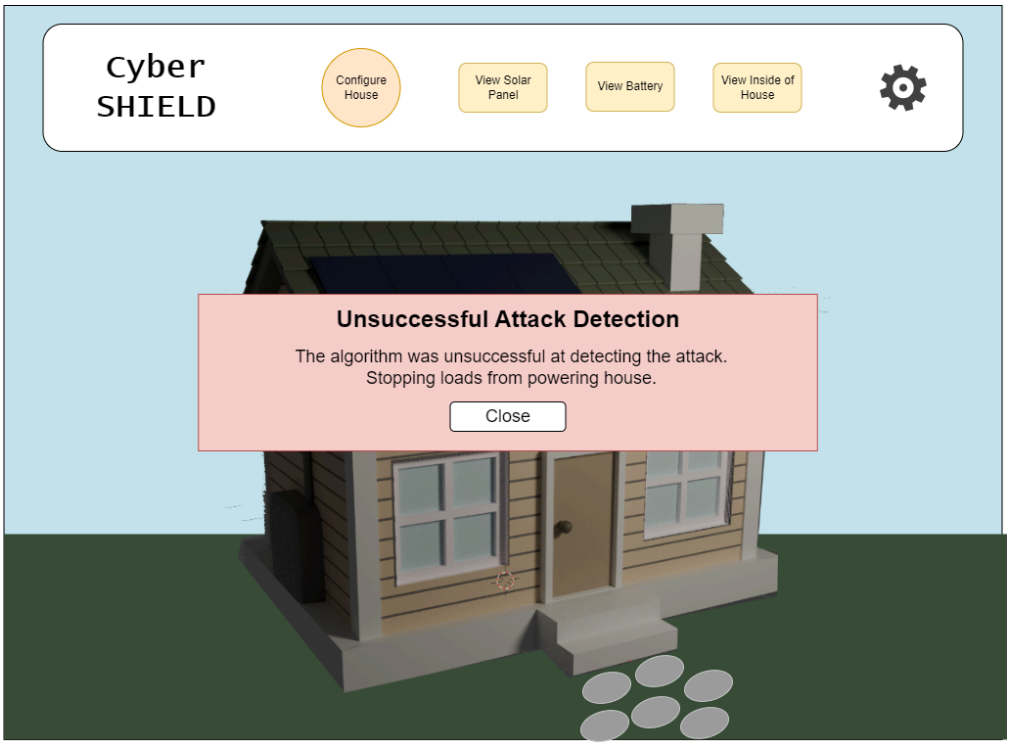


Figure 5: User Interface Prototype - Cyber Attack Unsuccessful Detection

These prototype illustrations were referenced in the construction of the physical model as well as the development of the user interface.

2.9 Project Block Diagram

Shown in the figure below is the hardware block diagram for the project. The sources and the loads were all be metered and the information was used to intelligently select sources and loads to power based on the state of the system. The EMS was able to perform this intelligent selection by switching on and off relays.

As shown, there are three power sources including the PV panel, battery, and AC connection. There are two loads including the DC and AC loads.

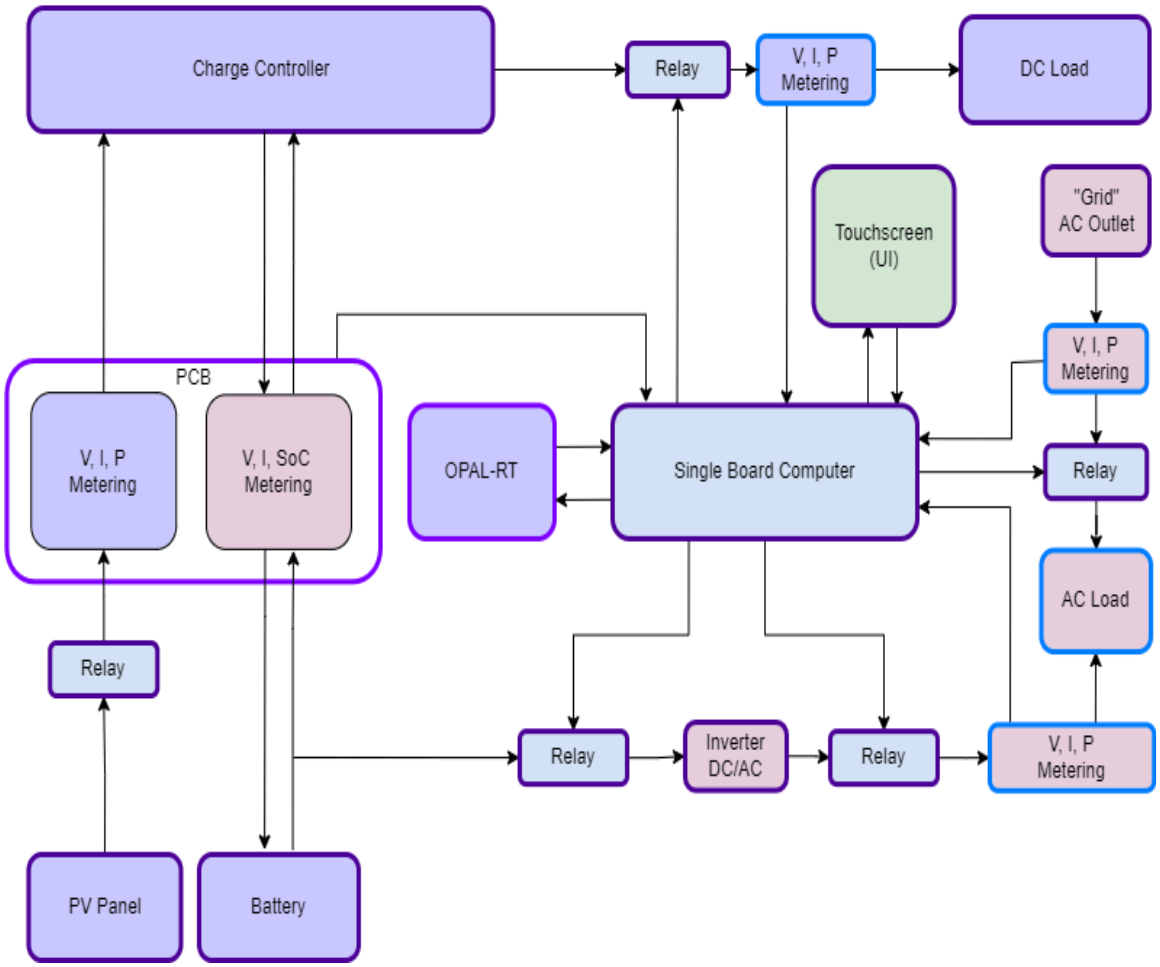


Figure 6: Hardware Block Diagram for Project

3. Research and Investigation

3.1 Part Selection and Relevant Technologies

Selecting the right components for the design was one of the most important parts of the design process. For our smart home, there were a few components that we required. First, we made use of a MPU as well as a form of power supply. This is a typical requirement of most engineering designs. In addition to this, we needed the different energy sources to power the loads in the smart home. This meant that we would need a PV panel, a battery, and an AC power supply.

In order to connect the PV panel to a battery, we needed a solar charge controller. This solar charge controller ensured that the voltage and current from the PV panel to the battery was regulated to prevent an event of overcharging. This prolonged the battery's lifetime. In addition to the sources, we needed to have both AC and DC loads. The loads that we used in our project were typical household items that one would see in a smart home. Since there will be AC loads, we had a DC/AC inverter. Furthermore, to implement an energy management system, we had a Raspberry Pi and relays to connect and disconnect the sources and loads. Lastly, incorporated a touch screen to allow for users to select specific test cases for the model. Along with the components, we also selected the appropriate software to enable the objectives of this project.

There were a handful of parameters that we took into account as we selected our components. Two of the most important parameters included performance and cost. We knew we would need to select components that were able to meet the performance requirements needed from them. If we were able to find two components that could both meet the performance demands then we would select the one that was the least expensive.

Another consideration in the selection of our components were their power consumptions. Since we were making use of one small PV panel, we needed to make sure that the loads would be able to work. In addition to this, we wanted to make sure all the components could be compatible with each other. This meant that they could work together to form the completed project without any issues. Overall, these specifications were considered when determining what components and technologies we would select.

3.2 Hardware Selection

3.2.1 Smart Home Components

The smart home model included all energy source components, loads, and the devices that allow them to be connected. In the following sections, we will discuss the component selection process when choosing parts for the smart home model.

3.2.1.1 PV Panel

One of the energy sources for our smart home model was a PV panel. The PV worked by using photovoltaic cells to convert the energy received from the sunlight into electricity. When choosing a PV panel, there were multiple factors that went into consideration. First, we considered what type of panel we wanted: monocrystalline, polycrystalline, or thin-film. Each type of panel has advantages and disadvantages in terms of cost, efficiency, and aesthetics. For example, polycrystalline is usually less expensive than monocrystalline but tends to be more expensive than thin-film. Another factor considered was the durability of the solar panel. For our project, we used the solar panel in indoor applications to test our model. For this reason, it did not need to be as durable as a traditional solar panel that would be used on the roof of a home. Additionally, when selecting a solar panel, we reviewed the energy output and performance statistics. This allowed us to select appropriate loads for the capabilities of our panel. With these factors in mind, we conducted market research to analyze what PV panel would be used for our project. We decided to focus on panels that were capable of generating a maximum output of 100W with voltage around 12V.

It is important to note the key differences between monocrystalline cells, polycrystalline cells, and thin-film cells. Table 3 summarizes the differences between these solar cells. Mainly, monocrystalline are known for their high efficiency rating. They usually have higher levels of energy production compared with the other two types. Polycrystalline panels are a popular choice for residential installation because they have relatively high efficiency and lower cost compared with monocrystalline panels. Lastly, thin-film solar panels are the least expensive but they are less efficient than monocrystalline and polycrystalline.

Table 3: Comparison of Types of Panels

Parameters	Monocrystalline	Polycrystalline	Thin-Film
Cost	High	Moderate	Low
Efficiency	High	Moderate	Low
Lifespan	Minimum of 25 Years	Up to 25 Years	10-20 Years
Temp. Coefficient	High	Moderate	Low

Out of these three types of solar panels, we decided that we would need to go with the monocrystalline solar panel to meet the requirements of our project. After deciding this, looked into monocrystalline panels that would function well for the model smart home. The first solar panel that we considered for our project was the Renogy 100W 12V Monocrystalline Solar Panel [5]. This panel was shown to be high in power and compact in size. In fact, it was 8-10% smaller compared with other 100W solar panels. It was

rated to have 500Wh of power per day when there was 5-6 hours of direct sunlight. The second solar panel that we considered for the project was the MEGA 100W 12V Monocrystalline Solar Panel [6]. This panel was emphasized to be durable in outdoor conditions. After comparing the specifications for each of these panels, we were able to create a table detailing the similarities and differences.

Table 4: Renogy vs. MEGA

	Specifications	
Parameters	**Renogy**	**MEGA**
Price	$104.99	$109.99
Dimensions	41.8 x 20.9 x 1.38 in	58.7 x 13.8 x 1.2 in
Max Power Output	100W	100W
Optimum Operating Voltage	20.4V	19.5V
Optimum Operating Current	4.91A	5.13A
Open-Circuit Voltage	24.3V	22.8V
Weight	14.1 lbs	12.6 lbs

All of the information above can be found in the specification documents for the solar panels. The Renogy had a lower price than the MEGA. In addition to this, the surface area was smaller on the Renogy. When it comes to the maximum power output, they were both rated at 100W so this showed that they are comparable. The optimum rated voltage and current of the Renogy are comparable to the MEGA. When multiplying the operating voltage with the operating current of the solar panels, we found that the Renogy could produce 100.164W and the MEGA could produce 100.03W. This meant that the Renogy could produce 0.134W more.

After completing this comparison, we went forward with the Renogy solar panel as it was less expensive and had the specifications that we needed to meet for the project. It was found to be the most durable and efficient option.

3.2.1.2 Battery

Selecting the appropriate battery for the model smart home was an important task as we needed it to be affordable and capable of supplying both the DC and AC load. The charge controller that we selected was paired with four different types of batteries. These included the gel, flooded, lithium, and sealed batteries. Each of these batteries presented different advantages and disadvantages. To select the appropriate battery, we completed a table comparing the four available battery types. This table can be seen below.

Table 5: Comparison Between Battery Types

Parameters	Flooded	Gel	Lithium	Sealed
Cost	Low	Moderate	High	Moderate
Lifespan	3-5 Years	5-7 Years	10-15 Years	5-7 Years
Maintenance	High	Low	Low	Low
Efficiency	Moderate	Moderate	High	Moderate
Self-Discharge Rate	Moderate	Low	Low	Low
Depth of Discharge	Moderate	Moderate	High	Moderate
Charge Rate	Moderate	Low	High	Moderate
Weight	Heavy	Moderate	Light	Moderate

Upon reviewing the comparison between the different types of batteries, we were able to select sealed batteries and gel batteries to consider for our project. This was because these were the medium cost out of the different battery types and they ranked well for efficiency, charge rate, and depth of discharge. The lifespan of these batteries fit our needs. Additionally, the weight of the battery was manageable given that the project was placed on wheels for easy transportation.

After selecting that we would look into gel and sealed batteries further, conducted market research on the best selections. In this research, we focused on selecting a battery that was cost effective and efficient. First and foremost, it was important that the battery would be able to meet the requirements of the project. After this was met, we based our decision based on the battery that would be the best for the price.

In order to determine what battery capacity we would need, we looked into the efficiency of the solar panel we selected. Our solar panel that was selected was rated for 100W. This meant that the battery would receive about (100 W * 6 hours) = 600 W. If we have a 12V battery and it is 50% charged, then we would need 600W x 2 = 1200W. By converting watts into amp hours, we divided 1200W / 12V = 100Ah. This meant that a 100Ah battery would work with our solar panel. Anything more than this allowed for excess battery storage for use when the solar panel was no longer producing power. This would happen at night or during times that it was cloudy.

Considering these findings, the batteries that we considered were the Renogy Deep Cycle AGM Battery 12 Volt 100Ah [9], the WEIZE Deep Cycle AGM Battery 12V 100Ah [10], and the Mighty Max Gel Battery 12V 100Ah [11].

To select a battery we should use for the project, we created a table detailing the strengths and weaknesses of each product. This table allowed us to compare the batteries side by side to select the one that met our requirements and was affordable.

Table 6: Comparison of Batteries

	Specifications		
Parameters	**Renogy (Sealed)**	**WEIZE (Sealed)**	**Mighty Max (Gel)**
Cost	$189.99	$219.99	$199.99
Nominal Voltage	12V	12V	12V
Self-Discharge	(77°F/25°C): ＜3% / month	(77°F/20°C): ＜3.3%/ month	(77°F/25°C): ＜3% / month
Rated Capacity	100Ah	100Ah	100Ah
Dimensions	13.1 x 6.9 x 8.6 in	12.99 x 6.73 x 8.43 in	12.10 x 6.65 x 8.47 in
Weight	63.9 lbs	63.0 lbs	59.22 lbs
Max Discharging Current	1100A	1150A	1100A
Standard Operation Temperature	77°F±9°F	77°F	77°F

After analyzing this table, we selected the battery that we would use for our project. We took a few things into account while selecting a battery. The first thing we considered was whether the battery met the requirements of our system. All of the options we selected successfully met the requirements of the system. The next thing that we considered was the price of the battery. As shown in the table above, the Renogy battery was the most cost effective option out of the batteries selected. Due to this, we decided to go with the Renogy Gel Battery as it met all the specifications we were looking for and was listed at a reasonable price. This Renogy battery would also be integrated well into our system because the Renogy charge controller and bluetooth meter worked best with Renogy batteries.

3.2.1.3 Charge Controller

The charge controller was an important component of our project. This component was used for maximizing the efficiency, lifespan, and safety of the solar system. It did this by managing the flow of electricity between the solar panel and the battery. Charge controllers have five main functions: regulating charging voltage, preventing overcharging, temperature compensation, load control, and preventing reverse current flow. Without a charge controller, we could not connect the solar panel to the battery safely. When selecting a charge controller, we took multiple factors into account. We considered the specifications of the charge controller and how well it would be able to integrate into our solar system. In addition to this, we considered its overall cost.

There are two types of charge controllers in the market today: Pulse Width Modulation (PWM) controllers and Maximum Power Point Tracking (MPPT) controllers. PWM controllers are usually smaller and operate at the voltage of the battery. MPPT controllers use new technology and operate at the maximum power voltage. The table compares these two types of controllers.

Table 7: MPPT and PWM Charge Controller Comparison

MPPT Charge Controllers	PWM Charge Controllers
The Excess Vin Is Converted to Amperage	Array V Is Pulled Down to Battery V
Operates at Vmp	Operates Below Vmp
Good for Large Module Configurations with Lower Cost Per Watt	Good for Small Module Configurations
Provides Boost, Especially During Cold Days or When Battery Voltage Is Low	Usually Chosen for Hot Climates and Will Not Have as Much Boost as MPPT

To select a charge controller, we took several factors into account. First, we considered the system voltage. Our system voltage was rated at 12V. Second, we calculated the optimum current necessary in our project. This was found to be around 5A and we decided to select a charge controller with a current rating equal or higher than this value. Third, we decided to limit the battery options for the charge controller to four types of batteries: flooded, gel, lithium, and sealed. Fourth, we considered the level of compensation we needed for temperature variations. Considering these factors, we completed market research on what solar charge controller we should use for our project. We decided to look into both MPPT and PWM charge controllers to compare their cost and capabilities. The first charge controller we considered was the Renogy Wanderer PWM 10A Charge Controller [7]. This charge controller allowed for protection against overcharging, overloading, short-circuits, and reverse polarity. This controller had the ability to be paired with the Renogy BT-1 Bluetooth module to send real-time data of battery and solar monitoring. It is compatible with Lithium Ion, Sealed, Flooded, and Gel.

The second charge controller considered for the project was the Victron Energy BlueSolar MPPT 12/24V 10A Solar Charge Controller [8]. This charge controller allowed for intelligent battery management. It had six ports: two for the positive and negative of the battery, two for the positive and negative of the panel, and two ports for the positive and negative of the DC loads.

After completing market research on the two solar charge controller options (one being a PWM controller and the other being a MPPT controller), we made a table to compare them side-by-side. This aided in the selection of the appropriate controller for our project. The table below compares the specifications of the solar charge controllers.

Table 8: Charge Controller Comparison

Parameters	**Renogy Controller**	**Victron Controller**
Battery Voltage	12/24V	12/24V
Rated Charge Current	10A	10A
Max PV Input	130W / 12V	145W / 12V
Charging Algorithm	Multi-Stage Adaptive	Multi-Stage Adaptive
Self Consumption	Less Than 10mA	Less Than 15mA
Operating Temperature	-31F - 113F	-22F - 140F
Weight	0.27 lbs	1.1 lbs
Dimensions	4.68 x 2.95 x 1.08 inches	3.94 x 4.45 x 1.57 inches

After comparing the two charge controllers, we decided to select the Renogy Wanderer PWM 10A Solar Charge Controller as it was the most affordable and met all of the requirements of our project. This charge controller was easily integrated with our solar panel since they are both Renogy products and designed to be used together. Along with the PWM charge controller, we decided to look into the Renogy BT-1 Bluetooth module to monitor the voltage and current from the solar panel and the voltage, current, and state of charge from the battery. This device allowed us to test the measurements collected by the metering printed circuit board.

3.2.1.4 Inverter

In our project, the inverter was used to convert the DC power from the solar panel and load to AC power for the AC loads. This component was important in order to build a functioning model. In order to select an inverter appropriate for our project, we completed research to find inverters that were compatible with the solar panel and battery that we selected. To determine what capacity of inverter we needed, we calculated our power needs. Given this, we needed an inverter with a capacity that was at least 20% higher than the largest power output. The largest power output of our solar panel was 100W. So we would need an inverter with higher than 120W capabilities. Upon further research, we found that at least a 12V DC to 220V AC 200W inverter is appropriate for appliances using a 100W panel.

In addition to compatibility, the inverter needed to be affordable and meet the specification requirements of the project itself. Depending on what inverter we bought, there were different types of wave outputs produced. The three most common types of inverters are modified sine wave, square wave, and pure sine wave inverters. The table below summarizes the differences between the different types of inverters.

Table 9: Comparison of Inverter Types

Parameters	Pure Sine Wave	Square Wave	Modified Sine Wave
Output Waveform	Smooth Sine Waveform	Square Waveform	Waveform with Steps
Efficiency	High	Low	Moderate
Harmonic Distortion	Low	High	Moderate to High
Appliance Compatibility	High	Low	Moderate to High

Parameters	Pure Sine Wave	Square Wave	Modified Sine Wave
Price	High	Low	Moderate
Noise	Silent	Possible Humming	Possible Humming
Heating	Minimal	Might Cause Heat	Might Cause Heat

Since we would not be interconnecting this inverter to the grid, we decided to go with off-grid inverters. In order to determine the off-grid inverter to choose, we conducted market research. The first inverter that we considered was a pure sine wave inverter. This inverter was the Cotek 200 Watt Off-Grid Solar Inverter 12VDC [12]. This inverter was suitable for use in remote power systems for telecommunications and other small load applications requiring 120VAC output [12]. It was lightweight and user friendly. The specifications for the Cotek can be seen in the table below.

Table 10: Specifications for Cotek Inverter [12]

Parameter	Cotek Inverter
Price	$175.00
Rated Power	200VA
Peak Power	250VA
DC Voltage	12VDC
DC Output V Range	10V - 16VDC
Output Frequency	50/60 Hz +/- 3%
Output Waveform	Pure Sine Wave
Efficiency	89%
Operating Temperature	-4°F - 140°F
No Load Power Consumption (No Load Mode)	< 0.5A
No Load Power Consumption (Save Mode)	<0
Dimensions (Inches)	5.91 x 2.68 x 7.36

The second inverter that we considered was Renogy 700W 12V Pure Sine Wave Inverter [13]. This inverter would integrate well with the rest of the components in our project as they all were Renogy products. The specifications for the Renogy Inverter can be seen in the table below.

Table 11: Specifications for Renogy Inverter [13]

Parameter	Renogy Inverter
Price	$119.99
Surge Output Power	1400W
Continuous Output Power	700W
AC Output V Range	115VAC
Output Frequency	60Hz
Output Waveform	Pure Sine Wave
Efficiency	Exceeding 90%
Operating Temperature	-4°F - 158°F
No Load Current	Less Than 1A
THD	Less Than 3%
Dimensions (Inches)	11.75 x 7.38 x 3.32
Weight	5.3 lbs

The third inverter that we considered was the Schumacher 1000 Watt Inverter [14].

Table 12: Specifications for Schumacher Inverter [14]

Parameter	Schumacher Inverter
Price	$169.99
Surge Output Power	2000W
Continuous Output Power	Max 1000W

Parameter	Schumacher Inverter
Operating Input Voltage	10.5 - 15.5 VDC
Nominal Output Voltage	110 - 125 VAC
Output Frequency (Nominal)	60 Hz
Output Waveform	Modified Sine Wave
Maximum Efficiency	90%
Input Low V Shutdown	0.10 +/- 0.5 VDC

In order to compare the inverters, we compiled all of the information across the three tables detailing specifications and compared them side-by-side. This exercise is shown below in Table 13.

Table 13: Comparison of Inverters

	Specifications		
Parameters	**Cotek**	**Renogy**	**Schumacher**
Price	$175.00	$119.99	$169.99
Surge Output P	250W	1400W	2000W
Continuous Output P	200W	700W	1000W
Output Frequency	50/60 Hz	60 Hz	60 Hz
Output Waveform	Pure Sine Wave	Pure Sine Wave	Modified Sine Wave
Efficiency	89%	90%	90%
Operating Temperature	-4°F - 140°F	-4°F - 158°F	-4°F - 150°F
Operating Vin	12 VDC	12 VDC	10.5 - 15.5 VDC

After careful consideration of the inverter options, we selected the Renogy inverter. We chose this inverter because it met all the specifications we were requiring for this project. It was the most cost effective option and it allowed for the second highest power output for the AC loads. A positive attribute to this was that the Renogy inverter was integrated well into our project as the other components were Renogy parts. Overall, this inverter was chosen because it was cost effective and could perform to the level we needed for the AC loads.

3.2.1.5 Relay Modules

The relay module in our model was a crucial component as it would be used to select the energy sources and the loads that were on according to the user's manual selections or the energy management system. The relay would need to be connected to our microcontroller to ensure that it could turn on and off. In order to select a relay module, we completed market research. We wanted to find a relay module that would function well with the solar system. We also wanted to ensure that the relay could be controlled by either a Raspberry Pi or Arduino.

The first step of the process of choosing a relay was selecting the type of relay we would like to use. The two main types of relays include Electromechanical Relays (EMRs) and Solid State Relays (SSRs). Table 14 compares these two types of relays side by side.

Table 14: Electromechanical vs. Solid State Relays

Feature	EMRs	SSRs
Mechanism for Switching	Mechanical components	Semiconductors
Speed to Switch	Slower (milliseconds)	Faster (microseconds)
Lifespan	Can be limited by mechanical wear; normally millions of operations	Longer than EMRs, but are limited by thermal stress; billions of operations
Size	Larger to hold mechanical components in relay	More compact and smaller
Typical Power Consumption	Higher energy consumption	Lower energy consumption
Noise (Electrical)	Arcing contacts create more noise	Minimal noise
AC or DC	Can be either AC or DC	Normally DC

Feature	EMRs	SSRs
Types of Loads	Both AC and DC	Normally DC
Quality of Electrical Isolation	Good electrical isolation	Better electrical isolation
Dissipation of Heat	Less heat	More heat
Cost	Generally lower cost initially	Higher cost initially
Examples of Applications	Home appliances, industrial controls, starters for motors	Automation in industry, lighting control, electronics that are sensitive

Using the table to compare the different types of relays is important when selecting the relay we will need to use for our project. Next, it is important to know the different types of relay configurations, whether it be an EMR or a SSR. Table 15 shows different types of relay configurations for EMRs and SSRs.

Table 15: Electromechanical Relays

Relay Configurations	Description
Single Pole Single Throw (SPST)	Most simple of the electromechanical relays and acts as a on and off switch
Single Pole Double Throw (SPDT)	Has one input and is able to be switched between two outputs
Double Pole Double Throw (DPDT)	Has two independent circuits that are able to be switched between two outputs

After careful consideration of the different types of relays, we decided to go with the SSR as it offered lower power consumption and faster switching times. These are two aspects that were crucial for the needs of our project as we had strict requirements for these categories. The first relay module that we considered was the HiLetgo 12V 8 Channel Relay Module.

This relay module was able to be integrated with both a Raspberry Pi and an Arduino. The specifications for the relay module can be seen in the table on the following page.

Table 16: Specifications for HiLetgo 12V 8 Channel Relay Module [15]

Specification	Value
Price	$11.29
Number of Relays	8
Maximum Load (AC)	AC 250V/10A
Maximum Load (DC)	DC 30V/10A
Trigger Current	5mA
Module Voltage	12V
Module Size	55.7 * 1.97* 0.73 in

As shown in the table, this relay met the specifications required by our system. It included eight relay channels on one board allowing a clean setup and connections between the various components. In addition to this, the module is competitively priced and affordable.

The second relay module that we considered was called the ANMBEST 12V Relay [16]. Again, this relay module was able to be integrated with both a Raspberry Pi and an Arduino. The specifications for the relay module can be seen in the table below.

Table 17: Specifications for ANMBEST 12V Relay Module [16]

Specification	Value
Price	$14.99
Number of Relays	5
Maximum Load (AC)	AC 250V/10A
Maximum Load (DC)	DC 30V/10A
Trigger Current	2 - 5mA
Module Voltage	12V
Module Size	1.97 * 1.61 * 0.73 in

As shown in the table, this relay also met the specifications required by our system. It was able to meet the action time that our system required at less than 10 milliseconds. This product came with ten relays and were listed at a higher price than that of the HiLetgo 12V 8 Channel Relay Module. Each relay in this product cost about $1.50, while each relay in the other product cost about $1.41. The two components were somewhat comparable in their pricing.

The team sought to find another relay module that would be a feasible option for the needs of our project. The specifications for the relay module can be seen in the table below. This relay board was suitable to be used with our Arduino or Raspberry Pi.

Table 18: Specifications for YQSIYU Solid State Relay [17]

Specification	Value
Price	$9.99
Number of Relays	1
Maximum Load	DC220V & AC250V
Trigger Current	2mA
Module Size	2.55in * 1.88in * 1.29in

After comparing all three options, we decided to go with the YQSIYU Solid State Relay. We chose this option because it was able to meet all the requirements of our project in order to successfully control the sources and the loads. This product was the most affordable option amongst the relay modules that were considered. This relay module can be successfully connected to a Raspberry Pi or Arduino to control the relays.

3.2.1.6 Mains AC Metering Devices

Measuring mains voltages and current can be dangerous due to the high voltage levels typically involved, which are usually in the range of 100V to 127V AC. These voltage levels are significantly higher than those found in low-voltage electronics and can cause severe electric shocks, burns, or even fatal injuries if proper precautions are not taken. Additionally, there is a risk of arcing, which can lead to fires or explosions, especially if the equipment is not rated for such high voltages. Inadequate insulation, improper grounding, and accidental contact with live wires are common hazards that can compromise safety. Knowing this, it was important that we take all safety considerations into account so we do not cause harm to ourselves or others.

For this reason, we have decided to use a noninvasive method to take the appropriate measurements. This method includes using a current transformer, also known as a CT, to measure current traveling through a wire. CTs are devices used to measure alternating current by producing a reduced current proportional to the current in the circuit being measured. They consist of a primary winding, through which the current to be measured flows, and a secondary winding, which produces a smaller, more manageable current. The primary winding is often just a single wire or conductor passing through the CT core, while the secondary winding, with many turns, generates a current that is a fraction of the primary current. This reduced current can be safely measured and analyzed by instruments and monitoring systems. This method does not require any connections directly to the outlet, making it very safe to use.

It is also important to know how much current our loads will be drawing. This value will change with how much our load is. Taking a liberal estimation of a load of 1000W, we can calculate the current by dividing the load by 120V. This will give us a current of around 8.33A. Knowing this ceiling value is important for us to pick an appropriate CT. The table below shows some comparisons of popular CTs.

Table 19: Comparison of CTs

Parameter	ZMCT103C	DL-CT08CL5	SCT010T-D	SCT-013
Maximum Current	5A	20A	20A	100A
Transformer Ratio	1000:1	2000:1	400:1	2000:1
Frequency Range	50Hz/60Hz	50Hz/60Hz	50Hz/60Hz	50Hz/60Hz
Burden Resistance	100Ω	User specified	100Ω	400Ω

The DL-CT08CL5 was chosen because of its high accuracy class, compact size, and reliable performance in measuring AC currents. This current transformer offers a favorable accuracy class of 0.1, ensuring precise current measurements critical for the project’s requirements. Its small form factor makes it easy to integrate into limited spaces without compromising the system's overall design. These features made the DL-CT08CL5 an ideal choice for achieving accurate and reliable current measurements in our project.

3.2.2 Touch Screen

The touchscreen in our project is used to display the user interface and allow for user interaction. Appropriate selection of the touchscreen was vital in order to allow for user-friendliness. The team wanted to have a touchscreen that was large enough to see the 3D model of the smart home in detail. It should look clean and professional while also meeting the requirements of the project itself. The display on the touchscreen needed to support multi-touch with at least two touch points. In addition to this, it needed to use either USB or HDMI connections to connect it to the microcontroller unit. We hoped to have a touchscreen large enough that at least three people could gather around it. Due to this, we found that the touchscreen should be at least 25” in height and 35” in width.

There are different types of touch screens. The two most popular types include resistive and capacitive touch screens. A side by side comparison of the two types of touch screens can be seen in the table below.

Table 20: Comparison of Resistive and Capacitive Touchscreens

Features	Resistive	Capacitive
Material	Two layers coated with a resistive material	Panel of glass with a transparent conductor
Detecting Touch	Measures any changes in resistance when layers make contact	Measures any changes in the electrostatic field
Touch Sensitivity	Lower	Higher
Multi-Touch	Limited	Yes
Clarity	Slightly Lower	Higher
Cost	Lower	Higher
Response Time	Slower	Faster
Accuracy	Less Precise	More Precise
Durability	Prone to Scratches	Scratch Resistant

After comparing the different types of touch screens, we decided to go with the capacitive touch screen as it would meet the requirements of our project more sufficiently. The capacitive touch screen allowed for faster response time and more availability for

multi-touch scenarios when there are many people viewing the project. The touch sensitivity was also higher and the clarity of the screen was better.

With this in mind, the team was able to conduct a comparison of multiple touchscreen options to select the best one. We looked into the specifications for each touchscreen individually and then compared them with each other. When looking into the options for touchscreens, the first one considered was the Samsung PM49H [18].

The specifications for this touchscreen can be seen in the table below.

Table 21: Samsung PM49H Touch Screen Specifications [18]

Price	$2,387.23
Dimensions	43.2 x 24.8 x 1.2 in
Screen	LED, HD, BLU
Available Connections	HDMI 2.0, HDCP 2.2, USB 2.0
Processor	Quad Core Cortex - A12 1.3GHz
Response Time	8ms
Storage	4.25GB Available
Weight	29.04 lbs
Power Consumption	47 W/h

The next touchscreen that the team considered was the Phillips 55BDL4051T [19]. This touchscreen proved to be cost effective and meet the minimum requirements of the project while having a professional display.

The specifications for this touchscreen can be seen in the table on the following page.

Table 22: Phillips 55BDL4051T Touch Screen Specifications [19]

Price	$1,678.00
Dimensions	50.04 x 29.20 x 3.60 in
Available Connections	DP 1.2, USB, HDMI, VGA, DVI-D
Processor	Quad Core Cortex - A9 1.8GHz
Response Time	12ms
Storage	16GB
Weight	39.68 lbs
Power Consumption	76 W/h

The next touchscreen that the team considered was the LG49TA3E- B TA3E [20]. The specifications for this touchscreen can be seen in the table shown below.

Table 23: LG 49TA3E- B TA3E Touch Screen Specifications [20]

Price	$1,910.71
Dimensions	45 x 26.5 x 2.8 in
Available Connections	HDMI, USB, DVI-D
Screen	FHD IPS
Response Time	12ms
Weight	49.2 lbs
Power Consumption	60W/h

After comparing the options selected for the touchscreen, we were able to narrow down which one would be appropriate for the project. First, we needed to stay within the budget that the lab provided for us. This meant that we would need to eliminate the Samsung touchscreen from our options as it was too expensive. In between the Phillips and LG touchscreen, we were able to see that the LG touchscreen had a smaller diagonal size, length, width, and height. Even though these dimensions were smaller, it was still priced higher than the Phillips touchscreen. The Phillips touchscreen met all of the

requirements we were looking for at a lower cost. Due to this, we decided to select this touchscreen for our project.

3.2.2.1 Touch Screen Frame

The touch screen is fixed on a wooden frame. This frame is also parallel to the ground allowing the users to view it from above on all sides. It is important to take note of the Phillips touch screen dimensions in order to ensure that the frame is large enough. The touch screen is 50”W x 29”H x 30”D and has a weight of about 40 lbs. In order to hold the weight of the touch screen and any weight placed by users that lean on the display, we chose wood as the most cost effective material. According to our calculations, we would need a table that supported at most 200 lbs to be secure. We decided that we would like the display to have at least two more inches of wood around the outside of the touch screen. It has legs with wheels on the bottom to raise the display up to around 35 inches tall. The wheels ensure that the touch screen table can be moved around with minimal effort. In order to protect the wooden frame from moisture and to add a more appealing aesthetic, we painted the frame.

3.2.3 Connection to Distribution System in Real-Time

When presented with this project idea by the UCF Digital Grid Lab, we were told that it would be crucial to connect the project to a real-time distribution system. Since we are not able to directly connect the model to the grid, it is important to use a simulated distribution system. As established previously, this project address two main areas of focus: (1) Creating an educational environment for students to learn more about the technologies used in grid modernization (such as microgrids, smart homes, and security resilience), (2) Building a cyber-physical system (CPS) testbench in order to test the cyber attack detection algorithm developed in the Siemens Digital Grid Lab at UCF.

In order to accomplish both these objectives it is vital that connection to a simulated power grid is established. Through this, the users are able to see how distribution systems behave and the UCF Digital Grid Lab is able to test their cyber attack detection algorithm. Currently, the lab has a 13-bus power system being simulated in real-time to perform testing of the algorithm. The testbench model that our team made communicates measurements from the sources to be inputted into the simulation established in the lab.

3.2.3.1 OPAL-RT Simulators

OPAL-RT TECHNOLOGIES is a leader in the development of PC/FPGA based simulators that run in real time, hardware-in-the-loop testing equipment, and rapid control prototyping systems [47]. This company provides engineers with simulation technology that is able to test equipment in power systems and electro-mechanical systems. OPAL-RT TECHNOLOGIES has been able to take part in multiple successful projects. For instance, the US DOE project focused on “High Penetration of Solar

Integration" made use of OPAL-RT's technology [48]. Involved in this project were UCF, the National Renewable Energy Laboratory (NREL), General Electric, Siemens, and Duke.

Due to OPAL-RT's popularity in the industry in simulating power systems in real-time, we decided to look into OPAL-RT technology for the requirements of the project. The OP5600 digital simulator was considered first due to its availability in the UCF Siemens Digital Grid Lab in the L3 Harris Corporation Engineering Center. Our sponsor, Dr. Wei Sun, is the director of the lab and allowed us to have access to the OP5600 real-time digital simulator. The research team in the Siemens Digital Grid Lab is currently making use of this technology for their cyber attack detection algorithm testing.

The important highlights of the OP5600 are listed below [49]. Following this, there is a list of OP5600 specifications in Table 24.

OP5600 Real-Time Digital Simulator Highlights:

- Best for Real-Time Simulation Applications
- Up to 32 Intel Processor Cores (Each at 3.0GHz)
- Linux REDHAT serves as real-time OS
- Up to 8 I/O Boards
- 256 Digital I/O and/or 128 Analog I/O
- 1-4 PCI Slots
- VIRTEX-6 FPGA or Xilinx SPARTAN-3
- DB37 Connectors
- Access to I/O Through Front I/O Monitoring
- Support for Communication Protocols
- Support for Third Party Devices
- IEEE C37.118, DNP3, IEC61850, CAN Bus, ARINC-429

Table 24: OP5600 Real-Time Simulator Specifications

General Specifications	
Price	About $27,000
FPGA	Xilinx Artix-7 FPGA, 200T
CPU	OP5650V3-8, 8 Cores, 3.2 GHz
Software	RT-LAB and HYPERSIM
Communications	4 SFP Sockets 1-5 Gbps Duplex Optical Fiber: 50/125 or 65/125μm) 5 PCI or PCIe Interface Cards Compatible with Xilinx Aurora
Performance	On 3.2 GHz CPU, Up to 120 3ϕ Busses
Dimensions	7” x 18.8” x 19.4”
Simulator Weight	20 lbs to 33 lbs

For our project, it is important that the OPAL-RT supports the use of commonly used power system protocols such as Modbus, DNP3, IEC61850. This is important because we need to communicate the measurements that we receive from the model to the simulated power system in order for the cyber-attack algorithm to be run successfully.

The OP5700 Real-Time Digital Simulator was another option for our project and was considered alongside the OP5600. This simulator can be found in UCF Research 1 Building. The general highlights of this simulator can be seen listed below [50]. Table 25 details the general specifications for the OP5700.

OP5700 Real-Time Simulator Highlights:

- Linux-Based Real-Time OS
- CPU: Intel Xeon E5 with 4, 8, 16, or 32 Processors (Up to 3.2 GHz)
- ATX Motherboard
- Memory: 10MB Per Each 4 Cores
- DRAM: Up to 32GB
- SSD Disk: 512GB
- Six Available PCI Slots for Connecting FPGA and PCIe or Third-Party I/O

Table 25: OP57600 Real-Time Simulator Specifications

General Specifications	
Price	About $70,000.00
FPGA	Xilinx Virtex 7 FPGA
CPU	Intel Xeon E5 4 Cores: 3GHz, 32 Cores: 2.3GHz
Communications	16 SFP Sockets At Most 5GBps
Dimensions	8.75” x 18.8” x 19.4”

After comparing the key features of both simulators, we were able to determine an OPAL-RT Real-Time Simulator selection. Table 26 shows a summary of the main points that were considered when choosing the most appropriate one to utilize for the project. Due to the fact that both simulators were available at UCF, the main factor that went into deciding which one to use was its availability.

Table 26: OPAL-RT Comparison

Parameter	OP5600	OP5700
Availability	More Available for Use	Less Available for Use
Cost	About $27,000	About $70,000
CPU	OP5650V3-8 8 Cores, 3.2 GHz	Intel Xeon E5 4 Cores: 3GHz, 32 Cores: 2.3GHz
FPGA	Xilinx Artix-7 FPGA, 200T	Xilinx Virtex 7 FPGA
Communications	4 SFP Sockets 1-5 Gbps Compatible with Xilinx Aurora	16 SFP Sockets At Most 5Gbps

After comparing the two options for the OPAL-RT simulator we decided to go with the OP5600 as it was the most available and would meet the needs of our project. The OP5700 is currently being used by many researchers at the university and this could

cause it to be less open for use. Due to this, we selected the OP5600 as it had all capabilities we needed and it was more available to be used.

3.2.4 PCB Module

3.2.4.1 Microcontroller Unit (MCU)

The selection of an MCU is an important decision in the design of our PCB. This component is the brains of our PCB and controls all the peripherals we place on it. Our board also must be designed around the MCU, making it a very important component of our project. For our project, the MCU had a few requirements. It must be programmable. It must have an analog to digital converter to convert our analog input voltages to a digital format the MCU can read. It must also be able to take accurate measurements. Finally, it must be low cost. In order to select an appropriate MCU for our application, we compared some of the most popular MCUs on the market today and all data was organized into the table shown at the end of this section.

The first MCU we considered for the metering PCB was the ATmega328. The ATmega328 is a very popular microcontroller most notably known for its use in Arduino dev kits. It comes in a variety of packages and many different versions depending on the user's need. We considered the ATmega328 [21] in the 28-DIP package. The ATmega328 is an 8-bit AVR RISC based microcontroller. This chip has a lot of support because of its popularity, making it very easy to work with and find information about. The chip is also compatible with the Arduino IDE, making it very simple to program and implement. Some features of this MCU include [21]:

- 32KB flash memory
- 20MHz speed
- Throughput of one MIPS per Mhz
- 1 KB EEPROM
- 2 KB SRAM
- 23 general purpose I/O lines
- 32 general purpose working registers
- I2C, SPI, UART/USART connectivity
- Supply voltage of 1.8V ~ 5.5V
- 6-channel 10-bit A/D converter
- Five selectable power saving modes
- Cost of around $2.63 on digikey.com

The MSP430FR6989 is another popular microcontroller most notably known for being used in the embedded systems course at UCF making it very familiar for UCF students. This microcontroller is very powerful and has many applications. There is also plenty of documentation on it found on the Texas Instruments website where it can be purchased. It

comes in 80 and 100 pin configurations. Figure 23 below shows the 80 pin configuration of the MSP430FR6989 chip.

The MSP430FR6989 is a 16-bit RISC based microcontroller. It can be programmed using TI's Code Composer Studio. Students at UCF should have experience using this IDE as it was also used in the same embedded systems course. This MCU has all the requirements to be used in our project. Some other notable features include [22]:

- 128KB nonvolatile memory
- 16MHz speed
- 2KB RAM
- 83 general purpose I/O pins
- I2C, IrDA, SPI, UART/USART connectivity
- Supply voltage of 1.8V ~ 3.6V
- 12-bit SAR ADC
- Five selectable low power modes
- Cost of around $10.77 on digikey.com

The next consideration for the MCU was the ESP32. This is a bit different from the other microcontrollers mentioned in that it is actually a system on a chip (SoC). SoCs are integrated circuits that integrate components of an electronic system onto a single chip. Components can include CPU, RAM, secondary storage, wireless communication modules, and much more. They are usually more powerful than regular MCUs which leads to a drawback of consuming more power.

The ESP32-C3 has a couple SoC packages, 2 including the ESP32-C3-WROOM and the ESP32-C3-MINI shown below in figure 24. Both these models come with WiFi and bluetooth connectivity built-in, simplifying the design process.

These SoCs can have either a ESP32-C3FH4 or ESP32-C3FN4 embedded, 32-bit RISC-V single-core processor. These SoCs have been used by many enthusiasts because of their simplicity and connectivity. Some other features include [23]:

- 4MB flash memory
- 384KB ROM
- 408KB SRAM
- 2.412GHz ~ 2.484GHz speed
- 13 general purpose I/O pins
- I2C, I2S, SPI, UART, USB connectivity
- Supply voltage of 3V ~ 3.6V
- 6 channel 12-bit ADCs
- 802.11b/g/n WiFi
- Bluetooth v5.0
- Two low power modes

Table 27: Comparison of Microcontrollers

	Specifications		
Parameters	**ATmega328**	**MSP430FR6989**	**ESP32**
Cost	~$2.63	~$10.77	~$1.90
Supply Voltage	1.8V ~ 5.5V	1.8V ~ 3.6V	3V ~ 3.6V
GPIO Pins	23	83	13
ADC Resolution	10-bit	12-bit	12-bit
Parameters	**ATmega328**	**MSP430FR6989**	**ESP32-C3**
Clock Speed	20MHz	16MHz	2.412 ~ 2.484GHz
Built-in WiFi/Bluetooth	No	No	Yes
Serial Protocol	UART, I2C, SPI	UART, I2C, SPI	UART, I2C, SPI
Memory	32KB	128KB	4MB
Power Consumption	Low	Low	High

After completing research on these microcontrollers, we decided to choose the ESP32-C3. Specifically the ESP32-C3-WROOM model as it is widely available and has more features while being inexpensive. Because our PCB has some components that operate at 5V, the addition of this microcontroller required an additional power rail to be added for the 3.3V supply voltage required for this chip. This was not a problem, it was just something to consider when designing our board.

Another important thing to note is that this microcontroller is very powerful for our use case. We are not going to be able to utilize all the potential of this chip. But because of its cheap price and accessibility, we have decided to use it. The chip is widely available on many development boards on Amazon. This made the chip easy to replace when we ran into trouble when assembling our PCB because the same chip can be bought from Amazon and shipped in a day or two for a very low price.

3.2.4.2 Current Sensor

Current sensing is a vital part of our project. These measurements are communicated to our energy management system to choose the appropriate loads of our system. Measuring current is a difficult task compared to measuring voltage. There are several solutions to this difficult task but there are three popular technologies that were considered for our project: Shunt resistors, hall effect sensors, and current transformers. The table below summarizes the differences between these three technologies.

Table 28: Comparison of Current Sensing Technologies

Parameters	Shunt Resistors	Hall Effect Sensors	Current Transformer
Cost	Low	Moderate	High
Accuracy	Depends on calibration and the quality of measurement circuitry	High but may be affected by temperature variations	High within dynamic range
Installation and Size	Compact	Required to be near the current carrying wire	Bulky and needs considerations of magnetic coupling
Dynamic Range	Wide	Moderate to Wide	Narrow
Power Dissipation	Does not need power but dissipation is proportional to I^2*R	Requires low power for signal conditioning	Does not need power but may have losses
Frequency Response	Works for DC and low frequency AC	Works for DC and AC	Works for AC only

To select a type of current sensor, we took a few factors into consideration. Firstly, we need it to measure both DC and AC currents. Measurements are taken from the solar panel, battery, inverter, and the “grid” which is a wall outlet connection. Next, we know that the dynamic current range of our system would not be very wide, everything should function between 0 and 10 amps. Lastly we needed these current measurements to be accurate, within 5% of the actual values. From these considerations, we decided that hall effect sensors would be the best for our project. Their relatively small form factor and high accuracy make them a perfect fit for our project. After deciding this we began to research specific models of hall effect sensors.

Next, we need to select an appropriate model of hall effect sensor. There are plenty of options in the market to choose from. One important factor to consider is if they are unidirectional or bidirectional. Unidirectional hall effect current sensors will not be able to measure negative currents, so it is important you wire your system accordingly. Unidirectional current sensors are designed to measure currents ranging from 0 up to their specified maximum value. Common ratings for these sensors include 5A, 10A, and 20A. Bidirectional current sensors are designed to measure currents that are plus or minus their rated value. Common ratings for these sensors also include 5A, 10A, and 20A, making them perfect for our project. Bidirectional current sensors are ideal for our project since they eliminate any of the concerns about polarity during setup. Any negative values can be easily corrected through the PCB's software.

Considering this, we found four main options to choose from. These options include the ACS712, ACS723, MCS1802, and the MCS1806. All these options are a series of families of chips. They each have different packages depending on the needs of our system. For our use, we wanted a chip that can handle a maximum of 10A. Typically the higher the rated current, the lower the sensitivity. In lower current applications, these sensors are typically more precise which is why we want to choose the lowest maximum rated current possible for our system. Given this information, we will aim to select the versions of the chip with a maximum current rating as close to 10A as possible.

Table 29: Comparison of Hall Effect Current Sensors

Parameters	**ACS712ELCTR-20A**	**ACS723LLCTR-10AB**	**MCS1802GS-10**	**MCS1806GS-*-10**	
				***3**	***5**
Supply Voltage	5V	5V	3.3V	3.3V	5V
Sensitivity (mV/A)	100	200	132	132	200

Sensitivity error	±1.5%	±1.5%	±2%	±2%
Current Range	±20A	±10A	±10A	±10A

After completing research on all four hall effect current sensors, the ACS712ELCTR-20A was chosen. This current sensor has good sensitivity and worked with our project's 10A limit. The sensor is bidirectional which will make the installation process simple within our system, not having to worry about polarities.

It is important to note that the ACS712ELCTR-20A is the most available hall effect current sensor on this list. This sensor is available to purchase and ship within a few days on Amazon. Amazon also has plenty of prototyping boards with the many different configurations of this sensor. Even though the ACS723LLCTR-10AB is technically better performing, it unfortunately does not have the luxury of being widely available.

3.2.4.3 Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) is used on the PCB to convert the analog signals read by the current sensor into digital signals that can be processed by the chosen MCU. The current sensor produces signals that are continuous in nature and the MCU must have these values converted into digital form for analysis and monitoring. While there is an ADC built into our chosen MCU, having an external ADC can help enable more accurate and efficient data processing.

There are many different types of ADCs. Two of the most popular ADCs include Flash ADCs (Parallel ADC) and Successive Approximation Register (SAR) ADCs. The table below compares these two types of ADCs.

Table 30: Comparison of ADC Types

	Parallel ADC	**SAR ADC**
Speed	Very fast, high speed	Moderate, sufficient speeds
Operation	Utilizes comparators to compare Vin with reference voltages	Utilizes binary search algorithm to converge on Vin
Resolution	Typically low (6-8 bits)	Moderate to high (8-16 bits)
Applications	Video processing, digital oscilloscopes, radar	MCUs, digital multimeters, industrial control systems

Based on our comparison of different types of ADCs, we decided to go with the SAR ADC because it is commonly used in applications involving MCUs such as the one we would be using on our PCB. Given this, the first ADC that we considered was the ADS1115. This ADC is a low-power and high resolution 16-bit with an I2C interface. It is able to communicate with microcontrollers such as the Arduino and Raspberry Pi. In addition to this, it comes with four single-ended channels that allow for measurement of multiple analog inputs. The ADS1115 supports data rates from 8-860 SPS.

The second ADC that we considered for the project was the MCP3424. This was an ADC that is similar to the ADS1115 and would work for communication of the analog values read from the current sensor to the MCU. MCP3424 is an 18-bit ADC with four differential input channels. It used I2C to interface with the ESP32. It supports data rates between 3.75 SPS to 15 SPS. It is able to provide an integrated reference voltage to read accurate measurements.

The third ADC that we considered for the project was the ADS1015. This ADC is a 12-bit or 16-bit ADC. It has four single-ended input channels and uses I2C protocol for easy communication with MCUs. The ADS1015 supports data rates from 128 SPS to 3.3 kSPS. It is designed to be used for low power consumption and is suitable for battery applications. The table below compares this ADC with the other two ADCs in order to make a proper selection for the project.

Table 31: ADC Selection Comparison

Specifications	**ADS1115**	**MCP3424**	**ADS1015**
Price	$6.35	$5.09	$9.95
Resolution	16-bit	18-bit	12 or 16-bit
Input Channels	4 single-ended	4 single-ended	4 single-ended
Interface	I2C	I2C	I2C
Data Rate	8-860 SPS	3.75-15 SPS	128-3.3 kSPS
Power Consumption	Low	Low	Low
Applications	Sensor data acquisition	Precision Measurements	Sensor data acquisition

As shown in the table above, the ADS1115 was selected to be the ADC for our PCB as it was appropriate for the applications that we needed and was affordable. This ADC easily

interfaced with the ESP32 and accurately translated the current sensor's measurements in analog to digital values for the ESP32 to use.

3.2.5 Single Board Computer

Our project requires a single board computer (SBC) to process data coming in from our metering PCB and transmit it to both the OPAL-RT simulator and the touchscreen. It will also need to process information coming from the OPAL-RT and communicate that to our physical system. The SBC will process and send vast amounts of data. This SBC will need to implement an ethernet port to communicate with the OPAL-RT. There are a few options that would be sufficient for our project: Arduino Uno R3, Arduino Portenta H7, and the Raspberry Pi 4. The table at the end of this section compares these three options.

After careful consideration, we decided on the Raspberry Pi 4: Model B as the base hardware. This decision was made based on its reputation of being easy to implement, as well as the comprehensive operating system that it provides, which makes it easily accessible to all of the members on this project, and also has the capability to handle the power requirements necessary to drive the 5V needed to send the signal to the relays.

The first board we considered was the Arduino Uno R3. This MCU board is based on the ATmega328P mentioned in the previous section making it much less powerful than the other two options. It also does not have ethernet built into the board which will require the Arduino Ethernet Shield Rev2 to be purchased. This is an expansion board that would connect to the Uno to implement an ethernet port. Because of its limited processing power, the Arduino Uno would not be a good fit for our project. The Raspberry Pi 4 and Arduino Portenta H7 are more suitable for our needs.

The second board we considered was the Arduino Portenta H7. This MCU board is a much more powerful companion to the Uno. It is based on the dual core STM32H747 which includes a Cortex® M7 running at 480 MHz and a Cortex® M4 running at 240 MHz. This board comes with built-in WiFi and Bluetooth connectivity, as well as the usual wired interfaces like UART, SPI, and I2C. Like the Arduino Uno, the Portenta H7 will require the additional purchase of an expansion board to implement the ethernet port. The cost of these two components alone combined totals to over $150. For this reason, the Arduino Portenta was not chosen for our project.

The last board we considered was the Raspberry Pi 4. This is a single-board computer (SBC) based on the Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC. This board is significantly more powerful than the two options mentioned earlier. This board has many built-in quality of life features, including 4 USB ports, 2 micro HDMI ports, a USB-C power supply, and lastly a gigabit ethernet port. This board also runs its own operating system called Raspberry Pi OS. With all these advanced features, the Raspberry Pi is significantly less expensive than the Arduino Portenta H7 costing around $35 to $60 depending on the size of RAM the user requires. Because of its powerful

processing power, the built-in quality of life features, and its low cost, the Raspberry Pi 4 would be a good fit for our project's needs.

Table 32: Comparison of Stand-Alone MCU Boards

	Specifications		
Parameters	**Arduino Uno R3**	**Arduino Portenta H7**	**Raspberry Pi 4**
Cost	~$60	~$150	~$40
Processing Power	Very low	Low	High
Parameters	**Arduino Uno R3**	**Arduino Portenta H7**	**Raspberry Pi 4**
Memory	2KB SRAM	8MB SDRAM	4GB LPDDR4
Storage	No external storage available	SD card slot (through the expansion port)	MicroSD card slot
Built-in WiFi/Bluetooth	No	Yes	Yes
Built-in Ethernet Port	No	No	Yes
Operating System	No	No	Raspberry Pi OS

After thorough deliberation, we chose the Raspberry Pi 4: Model B as our base hardware. This decision stems from its stellar reputation for ease of implementation and the robust operating system it offers. The user-friendly nature of the Raspberry Pi 4 ensures that all project members can effortlessly access and utilize it. Additionally, it boasts the necessary capability to meet the power requirements, efficiently driving the 5V needed to send signals to the relays. This combination of accessibility and power made it the ideal choice for our project's foundation.

3.3 Software Selection

3.3.1 Programming Languages

The selection of a programming language is crucial as it directly impacts various aspects of software development and project outcomes. Different languages offer unique sets of features, performance characteristics, and ecosystems of libraries and tools. Choosing the right language can significantly influence development speed, scalability, maintainability, and even the long-term success of a project. Factors such as ease of learning, platform compatibility, performance requirements, and community support all play vital roles in determining which language best suits the project's needs. Therefore, careful consideration of these factors is essential to ensure efficient development, optimal performance, and successful implementation of software solutions. We chose four of the most popular programming languages to compare. These include Python, Java, C#, and C++. Table 33 shows a comparison of all the programming languages considered.

The first language that we considered programming in was Python because of its versatility. This programming language is well-supported on the Raspberry Pi 4 Model B. Its ease of setup is one of its most significant advantages, with simple installation processes for Python itself and its extensive libraries. The performance of Python on the Raspberry Pi is standard, and sufficient for a wide range of applications from automation scripts to data analysis. Python is also highly efficient, balancing computational demands with ease of coding, making it an ideal choice for educational purposes, quick prototyping, and diverse projects. Those traits run parallel with what we are needing for our project. Being versatile and the broad library that it supports make it compatible with a wide range of hardware that the Raspberry Pi can cooperate with, enhancing its appeal for our Smart Home applications and communication with the Relays and Touch Screen.

The second language that we considered was Java. This programming language is an object-oriented language that can also be utilized on the Raspberry Pi 4 Model B. Setting up Java on the Raspberry Pi is relatively complex compared to Python, requiring the installation of the Java Development Kit (JDK) and configuring the environment which can become tedious in nature. Java's performance on the Raspberry Pi is standard, making it suitable for applications that demand a reliable and stable runtime. However, Java's efficiency on the Raspberry Pi is semi-efficient, as it can be more resource-intensive compared to lightweight languages. Java tends to be used in server-side applications and embedded systems. For our purposes, it becomes a middle ground of the few coding languages when it comes to deciding. With adequate support for various libraries and frameworks, Java provides a decent hardware choice.

The third programming language that was considered for the project was C#. This is a powerful programming language developed by Microsoft, suitable for a wide range of applications on the Raspberry Pi 4 Model B. Setting up C# is complex, typically involving the installation of the .NET Core runtime and development environment. The

performance of C# on the Raspberry Pi is effective, particularly for applications that require strong performance and robust functionality. It is used in game development with Unity. In terms of efficiency, C# is semi-efficient due to its relatively higher resource consumption compared to lighter languages like Python. This wouldn't need to be a bigger concern as the Raspberry Pi 4 will be able to provide the resources to keep it functioning as intended, but we needed to be aware of the consumption as we continue to grow our needs. C# provides hardware accessibility, with support for various hardware interfaces and libraries, making it a suitable choice for us in our project.

The fourth programming language considered for the project was C++. Renowned for its high performance and efficiency, C++ is well-suited for a broad range of applications on the Raspberry Pi 4 Model B. Setting up a C++ development environment on the Raspberry Pi is relatively straightforward, involving the installation of the GCC (GNU Compiler Collection) and other necessary tools. C++ excels in performance, making it particularly effective for applications requiring intensive processing and real-time capabilities. It is widely used in systems programming, game development, and applications that demand direct hardware manipulation. In terms of efficiency, C++ is highly efficient due to its close-to-hardware nature and low-level memory management capabilities, allowing for optimized use of system resources. However, this also means that development in C++ can be more complex and time-consuming compared to higher-level languages like Python. The language's manual memory management and intricate syntax can pose challenges, especially for developers who are not as experienced. Despite these complexities, C++ provides excellent hardware accessibility, supporting various hardware interfaces and libraries, making it a powerful choice for our project. The trade-off between complexity and performance is a key consideration, but with the Raspberry Pi 4's robust resources, C++ can deliver the high performance and precise control needed for demanding applications.

Table 33: Comparison of Programing Languages

Feature	**Python**	**Java**	**C#**	**C++**
Ease of Learning	Very easy, beginner friendly	Moderate, more complex syntax	Moderate, similar to Java	Difficult, complex syntax and concepts
Performance	Slower	Moderate	Moderate	High
Syntax	Simple and readable	Verbose but clear	Verbose but clear	Complex
Development Speed	Fast	Moderate	Moderate	Slower

Real-time Applications	Less suitable	Less suitable	Less suitable	Highly suitable
Embedded Systems	Less suitable	Less suitable	Less suitable	Highly suitable
Compatibility with Arduino IDE	No	No	No	Yes

We selected Python as the programming language for our Raspberry Pi due to its simplicity and versatility, making it an ideal choice for developers of all skill levels. Python's clear and readable syntax accelerates the development process, reducing the learning curve and allowing team members to quickly contribute to the project. Its extensive libraries and frameworks provide powerful tools for handling various tasks, from GPIO control to web development, which are essential for our project's diverse requirements. Furthermore, Python's strong community support and abundant online resources ensured that we can easily find solutions to potential challenges. The language's compatibility with the Raspberry Pi's hardware and operating system made Python the perfect fit for our project's needs.

We chose C++ to program our ESP32 microcontroller due to its exceptional performance and efficiency, which are crucial for embedded systems. C++ offers low-level memory management and direct hardware manipulation capabilities, allowing us to optimize the use of the ESP32's resources. This level of control is essential for developing real-time applications and ensuring that the microcontroller operates at peak efficiency. Additionally, C++ is a widely-used language in embedded systems programming, providing extensive support for various hardware interfaces and libraries that are compatible with the ESP32. A significant advantage is the compatibility of C++ with the easy-to-use Arduino IDE, which simplifies the development process with its user-friendly interface and extensive community support. This combination of powerful performance, precise control, and ease of use makes C++ an ideal choice for programming our ESP32 microcontroller, enabling us to achieve the high performance and reliability required for our project.

3.3.2 Communication Protocols

A communication protocol is a way for information and data to flow between different devices. When talking about the different communication protocols we would be using for our project, there were several to choose from. They all have different defined elements for the flow of information between the connected and non-connected devices. Considering the different elements of our smart home system simulation, many of them

will need to be able to successfully make the communication smooth and without flaw. Critical evaluation of these communication protocols was essential.

The information protocols that we have researched and honed in on are GPIO, Wi-Fi, Bluetooth, I2C, UART, and SPI. All have their strengths and weaknesses for our project. These communication protocols will need to be used to communicate the PCB to the Raspberry Pi, and the Raspberry Pi will need to communicate between the Touch Screen. This chain of information needs to also flow backwards as well when new changes are made that are done by the User in the Smart Home Simulation.

To go into more detail about how the information is transferred, we will need to explain how the devices will be connected to a bus. This bus travels via either parallel configuration or serial configuration. This also needs to take into account the change in voltage requirements needing either a Leveler or a voltage regulation system between the communication lines. Communication between these different components is done through that voltage changing. Generally, the bus we described earlier is 0V and 5V, which are used as 0 and 1 respectively. Depending on the device, there is a voltage threshold given to the system to avoid interference. With these devices, they follow this standard as a whole.

One form of configuration we can use includes parallel configuration, which is designed around the concept that data can be received simultaneously as it also sends data. Data flows in parallel has a bit assigned for each frame of data. If you have an 8 bit frame, then you will need 8 channels for the data. Serial configuration instead only needs one channel for the flow of data, but sometimes doesn't have a clock signal given to the data to follow which will become important when it comes to certain protocols we could choose.

To start with, GPIO or General Purpose Input/Output is a communication protocol which takes in an uncommitted digital signal from a signal pin on a PCB or Microcontroller and switches it on and off through software controls to complete tasks without needing to go into more depth. This policy currently is what allows our Raspberry Pi 4 Model B to communicate with the Relays to switch from their Normally Off position to their Normally On position using a 3.3V logic level. Allowing our Smart Home System to decide on which Loads and Sources need to be connected to the Grid. This signal connection is also comparable to PIO, but the key difference is that these Input/Output sources aren't connected to any specific programmable system, meaning that you give it the purpose and allows a lot of freedom for whatever purposes you need to use them for. This is seen more commonly on FPGA systems where PIO commands are more common than the former. GPIO, being simple in nature, allows a level of versatility that makes controlling the relays a streamlined objective than the other protocols could've created.

However, being simple, these communication lines are only input or output. We cannot send over data through these GPIO connections such as variables or character arrays, this protocol is limited to only sending a digital signal across to control other devices. We will dive into more detail how we have our Raspberry Pi 4 communicate with the PCBs.

Next, Wi-Fi is a wireless communication protocol that is known for changing the whole world around us. The protocol provides an immensely high data rate and long distance of connectivity without a direct line of communication. For our purposes, we wanted to use it for remotely controlling the relays without the need of wires. We did learn later that we were able to connect our Raspberry Pi 4 to the internet and were able to communicate with it over PUTTY. Allowing us to access and code it without the need of accessing its OS as often. Beyond this, the protocol's difficulty of implementing its system into our project, we decided to leave its Wireless functionality on hold for now. In a possible future growth for the project, to demonstrate a smart home system in the real world beyond the cyber attack defense, we would want to use this protocol for the signal to send its attack to the grid to inform of the attack. This would grow the main objective of our goal with the project.

Following this is Bluetooth. Bluetooth is a very popular wireless protocol that is mainly seen in smart devices due to its ease and reliability for P2P (peer-to-peer) communication. It has seen a lot of application wireless audio, and location services. For our project, we will have access to Bluetooth 5, which in comparison to Bluetooth 4, has two new physical layers (PHYs) LE 2M and LE coded, along with the existing LE 1M from Bluetooth 4. LE 2M PHY doubles the data rate to 2 Megabits per second, which is crucial for data transfer at a higher rate for systems that will require significant data information to transfer. This PHY is responsible for error detection and correction, which is crucial for maintaining data integrity over longer distances. By employing Cyclic Redundancy Checking (CRC) for error detection and Forward Error Correction (FEC) schemes, Bluetooth 5 ensures a reliable connection even at lower Signal to Noise Ratios (SNR), achieving a Bit Error Rate (BER) of just 0.1%.

For the Raspberry Pi 4 Model B, bluetooth 5 provides an efficient way to communicate between the systems, however, for our project, we will not be able to demonstrate the bluetooth communications, as we don't have any other items that will be able to use bluetooth for the P2P connection/communication. If we had another device to be able to communicate with the Raspberry Pi 4, then we would be considering it as a must have for the project for its ease of use and convenience. Especially for a real smart home system controlling all power sources.

Furthermore, the I2C (Inter-Integrated Circuit) communication protocol is widely used in various applications such as sensors and gyroscope modules due to its simplicity and versatility. The protocol enables multiple masters to communicate with one or more slaves using an address scheme. Key features of I2C include its use of only two wires: Serial Data (SDA) and Serial Clock (SCL), making it efficient in terms of wiring. The I2C protocol operates using serial synchronous technology, where bits are transferred and sampled in sequence based on the clock signal generated by the master device. This allows for precise control over the timing of data transmission.

In an I2C communication setup, the data frame begins with the SDA signal transitioning from high to low, followed by the SCL signal. This sequence is repeated at the end of data transmission but in reverse order. The address frame, which includes the address for the destination of data transfer, is broadcast to all connected slaves. The slave device with a matching address responds with a low ACK signal, while the others remain high, taking no action. The protocol also includes read/write signals to denote the direction of data flow and ACK/NACK signals to validate successful data reception.

In terms of advantages, I2C's two-wire design simplifies circuit layout, supports multiple master/slave configurations, and is easier to implement compared to UART. However, it does have some limitations, including lower data transfer rates than SPI, limited data frames to 8 bits, and a lack of pre-known slave addresses. This communication protocol allows freedom of its use through this policy. One of its big advantages is its efficiency in communicating through the ESP32 to the Raspberry Pi 4.

Table 34: Key Elements of I2C

Feature	Quantity of Feature	Advantages	Disadvantages
Wires	2	Only needing 2 Wires	Less Data Transfer rate than SPI and other Protocols
Speed Modes	Standard 100 kbps Fast 400 kbps High 3.4 Mbps Ultra Fast 5 Mbps	Well Documented and Supported	Only 8 bits for Data
Synchronization	Synchronous	Allows Master/Slave applications and Synch clocks	Harder to implement than SPI and UART

Interface	Serial	Data transfer acknowledgement	Doesn’t know the slave address ahead of time
Masters	Unlimited	Easier to implement than UART	N/A
Slave	1008	N/A	N/A

Following this, UART (Universal Asynchronous Receiver/Transmitter) is a widely used communication protocol that facilitates serial communication between devices. Key characteristics and attributes of UART include its simplicity, use of minimal wiring, and suitability for straightforward master/slave configurations. In a smart home system project implementing the Raspberry Pi 4 Model B, UART plays a crucial role in enabling communication between the Raspberry Pi and various peripheral devices. The PCB that we have created along with the ESP32/Arduino UNO allows a large level of simplicity and ease of use that we had demonstrated simple solutions for system level problems that would’ve cost us weeks of time. The Raspberry Pi 4 Model B is equipped with multiple UART interfaces, making it well-suited for connecting to those devices.

For example, UART can be used to interface with an external microcontroller responsible for managing a specific subsystem within the smart home, such as reading current which is what we have the ESP32 controlling and the PCB sending data to it making it quicker on the data processing center. The asynchronous nature of UART communication simplifies the wiring and connection setup, as it does not require a shared clock signal, reducing the complexity of the system design.

Additionally, the ability to support baud rates up to 115200 allows for relatively fast data transfer, ensuring that commands and sensor data can be communicated efficiently between the Raspberry Pi and connected devices. The inclusion of error-checking mechanisms in UART further enhances the reliability of the communication, enabling a high level of maintaining the stability and responsiveness of the smart home system. Although, despite the limitation of supporting only a single master and slave, UART remains a valuable communication protocol for integrating specific components that require simple, reliable, and low-cost serial communication.

Table 35: Key Elements of UART

Features	Quantity of Features	Advantages	Disadvantages
Wires	2	Only needing 2 Wires	Single Master/Slave
Speed Modes	Up to 115200 baud rate	No Clock needed	10% Baud rate between each device
Synchronization	Asynchronous	Error Checking	Data Frame limit to 9 bits
Interface	Serial	Data transfer acknowledgement	N/A
Masters	1	Well Supported	N/A
Slave	1	N/A	N/A

Another one to consider is SPI. SPI (Serial Peripheral Interface) is a high-speed communication protocol widely used for connecting microcontrollers to various peripherals on top of the allowed accessibility. Its known for simplicity and efficiency in enabling synchronous serial communication between devices. For our purposes, if we wanted to use this, this would streamline the clock signals on all devices.

For our Smart Home System Project, utilizing the Raspberry Pi 4 Model B makes SPI serve as a robust communication protocol for connecting the high-speed peripherals of the sensors along with the other PCBs. The Raspberry Pi 4 Model B supports SPI, which would make it suitable for interfacing with additional devices that require fast and reliable data transfer.

For example, SPI can be used to connect the Raspberry Pi to an external ADC (Analog-to-Digital Converter) for reading analog current sensor data, or to an external display module for real-time data visualization, both of which can be vital to our project. The high-speed data transfer capability of SPI ensures that these peripherals and operations can be done efficiently, allowing us to continue to create an effective benchmark for the smart home system..

The synchronous nature of SPI allows for precise control over the timing of data transmission, ensuring that the master and slave devices remain synchronized. We want to ensure that timing in our project is done properly, as in applications where timing is

critical, our project involves real-time monitoring and control systems within the smart home. Without proper communication synchronicity, we would run into problems of delay or waiting.

Despite the need for four wires, the flexibility of SPI in supporting multiple slave devices makes it usable for our state machine design. Each slave device can be individually selected via the SS line, enabling the Raspberry Pi to manage various sensors, and other peripherals within the smart home system. One of its faults is the absence of built-in error checking and the acknowledgment mechanisms can be mitigated by implementing software-based error detection and correction which would be done through the Raspberry Pi 4, Model B, ensuring reliable communication.

Overall, SPI offered a powerful solution for integrating high-speed peripherals into a smart home system based on the Raspberry Pi 4 Model B, providing the necessary speed and flexibility to handle a wide range of devices and applications.

Table 36: Key elements of SPI

Attributes	Quantity	Advantages	Disadvantages
Wires	4	Constant Data Flow	Requires 4 wires
Speed Modes	Up to 10 Mbps	Addressing isn't needed	No ACK
Synchronization	Synchronous	Allows Master/Slave applications and Synch clocks	Harder to implement than SPI and UART
Interface	Serial	Data transfer acknowledgement	No error checking is done
Masters	1	N/A	N/A
Slave	Unlimited	N/A	N/A

After all the comparisons, we decided to use a mixture of a few different protocols. GPIO, UART, and SPI. Different Aspects of our project will need to be using different protocols. This is due to the different tasks that are needed for this decision. Relays, Accessing information, transferring data between microcontrollers, unity to pi communication are all different aspects of our project that will be using these protocols. For the purposes of explaining further, Relay control was through GPIO, accessing information via serial was through SPI/JSON communication, data between microcontrollers was through UART.

Table 37: Details on the Raspberry Pi 4 Model B Usage and Specifications

Protocol	Pins/Interface	Speed/Range	Use Case
GPIO	40 pins	N/A	Relays
Wi-Fi	Built-in (802.11ac)	433Mbps ~100 meters	Raspberry Pi / UI Application
Bluetooth	Built-in (Bluetooth 5)	Up to 2Mbps ~10 meters	Remote Communication
I2C	SDA.SCL	Up to 400 kbps	Communication between SBC
UART	TX,RX	Up to 115200 bps	Communication between SBC
SPI	MISO, MOSI, SCLK, SS	Up to 10 Mbps	Communication between SBC
OpenDNP3	TCP/IP	Up to 100 Mbps	Raspberry Pi / OPAL-RT

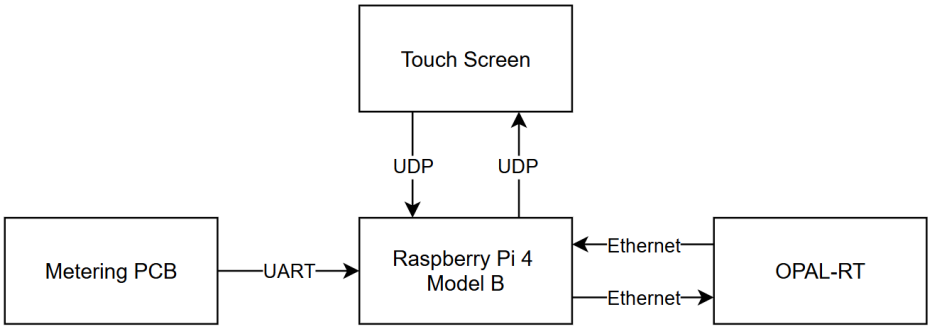


Figure 7: Communication Protocols lines on the Smart Home System

3.3.3 Comparison of Software Technology for Model

One of the core features of this project is the development of a user interface for the smart home model. This user interface serves as a digital twin of the circuit we have developed to replicate that of a smart home. With this application, the user can interact with the smart home and toggle specific settings including: AC/DC, Day/Night, Battery/Grid. As the user adjusts these settings, there is a visual representation of their impact on the smart home. The core feature of this "digital twin" is to act as a testbench to test various cyber attack detection algorithms. When these attacks occur, the user is able to view whether or not the algorithms were successful in the detection of the attacks.

In the project, the software platform serves as the main component that implements the visualizations of the smart home model. In the design of this component, several options are available for this implementation, requiring a good amount of research to reach a final decision. Several technologies are available to visually represent a system. Below are several technologies considered for implementing this model.

Table 38: Technology Considerations

Parameters	Game Engine Executable	Web Application
Components	Executable file	Static Page Generator
Language Type	Object Oriented Language	Markup Language
2D or 3D	Both	2D
Languages	C++ OR C#	HTML/CSS OR Node JS
Application Customization	Tools available to implement	Tools need to be created
Cost	Free	Free
Computing Power	Requires more power	Requires less power

Looking at the table above, there are several benefits to using each of these different technologies. From customization tools to computing power, each of these offer great benefits to specific applications. For the requirements of this project, we determined that utilizing game engine tools would prove to be incredibly useful in the development of this application. Some of the great benefits included available user interface components, simple application start-up using an executable and the object-orientedness of the program. While this solution may require more computation power, the benefits it offers in its user interface implementation justify the choice to use this technology.

3.3.4 Software Selection for User Interface

In this section, we will be comparing various software products for our software technology. The comparisons will be done in a table and the various choices will be thoroughly evaluated. From this, we will be able to identify the best software choice for our application.

3.3.4.1 Application Game Engine

Now that we have chosen to utilize a gaming engine to create the application, it is important to determine which engine works best for this application. In the context of this project, the two engines considered were Unity and Unreal Engine. Both of these are tools greatly used not only in the gaming industry, but also other industries. These companies have grown to reach these other industries and offer tools to implement various solutions. In order to determine the best engine to utilize, we start by identifying the various features that they offer:

Table 39: Software Features

Parameters	Unity	Unreal Engine	Godot
Resources / Support	Engine documentation Large community support	Engine documentation Medium community support	Engine documentation Small community Support
Price	Free	Free	Free, Open source
Typical Applications	Simple 2D/3D projects	Complex high resolution graphics	2D/3D projects
Languages	C#	C++	GDScript, C#, C++
Audience	Small team development	Large team development	N/A
Common Uses	Independent developers, Education, Small to medium sized projects	Triple-A titles, High-quality indie games, VR experiences	2D and simple 3D games, hobbyists and independent developers, education

Both of these options offer great benefits for the creation of our application. Unreal Engine offers great benefits in creating high quality, complex applications. Unity, on the

other hand, offers great options for creating simple applications. In the context of this project, Unity was the best choice, as it offers great tools for creating features and the fact that we are not creating complex applications.

Unity is a great tool that has a ton of resources from the company and its community. When building the interface for the smart home model, Unity offered great premade components for buttons, text, animations, and more. In the design of the application, we created a 3D model of the smart home, where the user can toggle between the various options and the smart home will respond accordingly. Unity is great in the way that the developer can easily implement new features to the application.

3.3.4.2 3D Modeling Software

The next piece of software that required investigation was the software we were going to use for 3D modeling the house and its various components. The UI Application consists of several features: information about the smart home, the ability to view different components, control over the configuration parameters of the simulated smart home. With these features, we want a 3D model of the house in our application. To prepare a customizable house for our application, it is important to select the best software for this task. In Table 23, we see a comparison between various modeling softwares and the features they possess.

Table 40: 3D Modeling Software Comparison

Parameters	**Blender**	**SolidWorks**	**Maya**
Price	Free	Free (student)	Free (student)
Key Features	Keyboard shortcuts, flexible modeling	Industrial design modeling	Layers, flexible modeling
Resources / Community Support	Great	Great	Great
Modeling Flexibility	Great	Good	Great
Ideal Application	Independent projects	Mathematically defined models	Animation and visual effects

Looking at these different comparisons, we see the options available for modeling the house. Between the three options, they are similar in price and resources / community

support. Utilizing student licenses, we can get each of the softwares for free. Additionally, there is great community support for each of the softwares.

Now, looking at the differences, we begin by reviewing the comparison of Blender. Blender is known for its many keyboard shortcuts and flexible modeling. This means that there are several tools available to the user for modeling different things. The ideal use case that Blender is used for is independent projects, which would be useful for this purpose.

Moving over to SolidWorks, this modeling software is used more for industrial design, taking a more mathematical approach. Solidworks is known for being used to develop parts / models that would be printed in the real world. In our case, we are creating models to integrate into the Unity game engine, so this may not be the best option for the application.

Lastly, we evaluated Maya for developing the 3D model of the house and its various components. Maya is known for its use in animation. The ideal application for utilizing Maya is for animation and visual effects. For our application, these features are not necessary, as the house itself will remain static and we are just using it to import to Unity.

After reviewing all these options, we see that the best software is Blender. Blender offers a variety of tools for modeling, giving us the ability to customize the house. In the development pipeline, we used this tool to model the house and apply textures. We then exported the model to Unity and applied overlay effects such as energy movement animations, camera movement in the house and lighting changes. With this, we can prepare the application to show off the smart home and its various features. This will help us communicate our application and its capabilities.

4. Standards and Design Constraints

This section of our report will cover the relevant standards and various design constraints of S.H.I.E.L.D. It is key to take these aspects into account to assure an efficiently designed final product. The goal of this section will be to clearly examine and elaborate on all of the relevant standards and design constraints that were considered throughout this project.

4.1 Relevant Standards

Standards are very important for the development of innovative technology. They can ensure the interoperability of different components and systems by setting clear guidelines to manufacturers. Because S.H.I.E.L.D. is still being developed, some standards may be absent from this section, however, the groundwork of our design and planning have allowed us to determine several standards our group will use.

4.1.1 IEEE Std 1547-2018

IEEE 1547 is a comprehensive standard designed to ensure the safe and reliable interconnection of distributed energy resources (DERs), such as solar photovoltaic systems, with the electric power grid. It provides detailed criteria and requirements for the performance, operation, testing, safety, and maintenance of these interconnections. The standard addresses key aspects such as voltage and frequency regulation, anti-islanding protection, power quality, and communication protocols, promoting grid stability and interoperability between different DER systems and utility control systems. Compliance with IEEE 1547 is essential for achieving seamless integration of DERs into the grid and is often mandated by utilities and regulatory bodies. It is also important to note that it was not realistic for S.H.I.E.L.D. to be able to adhere to everything presented in this standard as it is only a small scale model. This standard is more geared towards larger scale commercial and private uses. It is our goal to create a realistic testbed so we tried as much as possible to follow the guidelines presented in IEEE Std 1547-2018.

4.1.2 IEC 61850

IEC 61850 is a powerful standard for communication and automation in power utility networks, offering robust data modeling, high-performance communication, and comprehensive configuration tools. While its primary application is in substations and utility automation, its principles and components are highly relevant for home energy management systems, particularly in the context of smart grid integration and the increasing adoption of distributed energy resources. Adopting IEC 61850 for a home energy management system can ensure interoperability, enhance real-time performance, and enable seamless integration with the broader power grid.

4.1.3 IEEE 2030

IEEE 2030 is a comprehensive standard that provides guidelines for achieving interoperability, integration, and efficient operation within the smart grid. It addresses the convergence of power systems, communication networks, and information technology to create a cohesive and functional smart grid environment. For home energy management systems, adhering to IEEE 2030 ensures seamless integration with the smart grid, real-time data exchange, efficient energy management, and robust security and privacy measures. This integration ultimately contributes to a more reliable, efficient, and sustainable power system for residential consumers.

4.1.4 IEEE 1379-2000

IEEE 1379-2000 provides comprehensive guidelines for data communication between intelligent electronic devices and remote terminal units within substations. By focusing

on communication protocols, data exchange methods, system architecture, interoperability, and security, the standard ensures reliable and efficient substation automation. Although designed for substations, its principles can be adapted to home energy management systems to enhance their performance, interoperability, and integration with smart grid systems. Adopting these guidelines can lead to the creation of a more efficient and reliable home energy management system.

4.1.5 Universal Serial Bus Standard

Universal Serial Bus also known as USB has been an industry standard for several decades. USB technology was first introduced in 1996 and was used to simplify and standardize the connections of peripherals to personal computers. This technology was developed with the help of multiple companies including: IBM, Intel Corporation, and Microsoft Corporation.

In the present age, there are many different types of USB classified by their shape and data transfer speed. The most popular USB connection types today are USB Type A and USB Type C. USB Type A connections are typically found on laptops and personal computers and are the most common type of USB connection. There are many different versions of USB Type A in use today ranging from low speed USB 2.0 (480 Mb/s) to high speed USB 3.0 (5000 Mb/s) to even higher speed USB 3.1 Gen 2 (10,000 Mb/s). The most recent version of USB in use today is USB4 These versions also support different amounts of power output, ranging from 2.5W for USB 2.0 to 4.5W for USB 3.0 and USB 3.1 Gen 2.

USB Type C connections are often found on many smartphones, tablets, and laptops. It is becoming the new industry standard and is starting to supersede USB Type A. USB Type C supports higher data transfer rates and higher power output over its predecessors. USB Type C supports up to 10,000 Mb/s transfer speeds and 15W of power output. USB Type C can output up to 100W of power depending on the type of cable. On the next page is a table showing versions of USB with their specifications.

Table 41: USB Types and Specifications

USB Version	Supported Form Factor	Maximum Transfer Speed

USB 1.0	Type A, Type B	12 Mb/s
USB 2.0	Type A, Type B, Type C	480 Mb/s
USB 3.0	Type A, Type B, Type C	5000 Mb/s
USB 3.1 Gen 2	Type A, Type C	10,000 Mb/s
USB 3.2 Gen 2x2	Type C	20,000 Mb/s
USB4	Type C	40,000 Mb/s

USB was needed in order to program our EMS onto the Raspberry Pi as well as to program the MCU on the metering PCB. Depending on which Raspberry Pi is chosen to use, it will have to be programmed and powered by using a USB Type A cable. Our metering PCB will also use USB Type A. The metering PCB will also utilize USB for data transmission to the Raspberry PI. USB will also be used to utilize the touchscreen feature of our display. USB makes this process very streamlined and straightforward.

4.1.6 Ethernet Standard

Ethernet was first created in 1973 by Xerox PARC to allow for communication between their personal computers. Standardization for it began in 1980, and by 1983, formal standardization endeavors led to the release of IEEE 802.3. This is a collection of standards that sets the framework of the physical connections as well as the process of actually transmitting data. Ethernet cables are categorized on their data transmission rate. The most popular category of ethernet cables today are CAT 5 (max data rate of 100 Mbps), CAT 5e (max data rate of 1 Gbps), and CAT 6 (max data rate of 10 Gbps at 37 meters but can be used at 1 Gbps for 100 meters).

In our project, Ethernet was used to transmit important data from the Raspberry Pi to the OPAL-RT to simulate our model in the simulator. The status of our loads and power sources, as well as various voltage and current measurements throughout our model, were sent to the OPAL-RT. It was important to use Ethernet for this purpose as it provided a fast and reliable way to transmit this data. Although the Raspberry Pi supported up to 1 Gbps Ethernet, any Ethernet category sufficed for our purpose since we did not send a vast amount of data.

4.1.7 High-Definition Multimedia Interface Standard

High-definition multimedia interface, typically referred to as HDMI, is a form of audio and video interface for transmitting audio and video data from a source device to a compatible display. HDMI was created with the combined efforts of Hitachi, Panasonic,

Philips, Silicon Image, Sony, Thomson, and Toshiba. It was released to consumers in 2003 but became popularized by the mid 2000s.

HDMI uses a 19-pin connector and cable that transmits data through Transition-Minimized Differential Signaling (TMDS). TMDS reduces electromagnetic interference and data loss, ensuring robust signal integrity over the cable. Data is transmitted using four TMDS channels: three for video (Red, Green, Blue) and one for the clock signal. Each TMDS channel consists of a pair of twisted wires, which helps maintain signal quality over distances.

The data transmitted over HDMI is encoded using 8b/10b encoding, where 8 bits of data are converted to 10 bits to maintain DC balance and reduce the likelihood of transmission errors. The clock signal synchronizes the transmission between the source and the display. HDMI supports uncompressed video formats and compressed or uncompressed audio formats. The video signal can handle resolutions from standard definition (SD) to ultra-high definition (UHD), including 4K and 8K resolutions. Audio support includes multi-channel formats like Dolby TrueHD and DTS-HD Master Audio.

HDMI includes several control signals such as the Display Data Channel (DDC) and Consumer Electronics Control (CEC). DDC uses a two-wire I2C bus for communication between the source and the display to exchange information like Extended Display Identification Data (EDID), which tells the source device the display's capabilities, such as supported resolutions and refresh rates. CEC allows control of multiple HDMI devices with a single remote, enabling devices to communicate over a dedicated CEC line. [36]

HDMI has an assortment of different cables for all different use cases classified by how much data can be sent through them. Some even include built in ethernet, however this will not be utilized in our project. The most basic cable is a Standard HDMI 1.4 cable which is capable of displaying 1080i and 720p. Also available is a High Speed HDMI 1.4 cable which is capable of 1080p and even 4K resolution at 30 Hz. Even higher speeds are available with Premium High Speed HDMI 2.0 and Ultra High Speed HDMI 2.1. These cables are capable of displaying 4K resolution at 60 Hz and 10K at 120 Hz respectively.

HDMI was used in our project to display our user interface on the touchscreen. Our touchscreen had a resolution of 1920 x 1080 at 60 Hz. Knowing this specification, we determined that we needed at least a high-speed HDMI 1.4 cable to support it.

4.1.8 Communication Protocol Standards

Communication protocols are very important in the development of any system that requires information to be transmitted between devices. Communication protocols are sets of rules that can be utilized to make sure a message gets sent with no loss of

information. Wired and wireless communication protocols are typically enforced by IEEE, but some have been widely adopted without the backing of IEEE.

4.1.8.1 UART Standard

Universal asynchronous receiver-transmitter, also known as UART, was first developed by Gordon Bell at Digital Equipment Corporation (DEC). Its main innovation was the use of sampling to convert an analog signal to a digital one. In around 1971, Western Digital adopted this design and developed the WD1402A, which became the first single-chip UART readily available to the market. Over the years, UART's transmission rates have steadily increased.

UART is a communication protocol used for asynchronous serial communication between devices, allowing data to be transmitted without a shared clock signal. The communication process begins with the sender transmitting a start bit, typically a low signal, which alerts the receiver that a data frame is starting. Following the start bit, a series of data bits, usually eight, are sent. These bits represent the actual data being transmitted, and both the sender and receiver must agree on the number of data bits beforehand.

An optional parity bit can be included for error checking, helping to verify that the data was transmitted correctly. After the data bits, one or more stop bits, typically a high signal, are sent to signal the end of the data frame. Both the sender and receiver must be configured to operate at the same baud rate, which is the speed of transmission measured in bits per second.

The receiver uses its internal clock to sample the incoming data at the agreed-upon baud rate. It synchronizes to the start bit and samples the data bits at precise intervals to accurately reconstruct the transmitted data. [37]

In summary, UART is an asynchronous serial communication protocol that transmits data without a shared clock, using start and stop bits for synchronization and relying on an agreed-upon baud rate for timing. UART is one of the simplest communication protocols we can implement in our project. This protocol was implemented in our project to transmit metering information from our metering PCB. It may be slower than other communication protocols like SPI and I2C, but because of its simplicity and its widespread support, it was used in our project.

4.1.8.2 SPI Standard

Serial Peripheral Interface (SPI) is unique in the fact that it does not have guidelines from IEEE or any other associations. It is what is known as a *de facto* standard. SPI was first invented by Motorola in the early 1980s and has been widely adopted by consumers. It uses four wires and is capable of full duplex communication. This means data can be

transmitted in both directions simultaneously. SPI also has faster data transfer rates compared to UART and I2C.

SPI is a synchronous serial communication protocol used to transfer data between a master device and one or more slave devices. It employs a master-slave architecture and relies on four main signals for operation. These signals are MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (Serial Clock), and SS (Slave Select).

The MOSI line carries data sent from the master to the slave, while the MISO line carries data sent from the slave to the master. The master generates the clock signal on the SCLK line to synchronize data transmission, with the clock's rising or falling edges used to sample the data bits. The SS line is used by the master to select the active slave device, and it is an active-low signal, meaning the line is pulled low to select a slave.

In SPI communication, the master device initiates the communication by setting the SS line low to select the desired slave. The master then generates clock pulses on the SCLK line, and data is simultaneously transmitted and received over the MOSI and MISO lines. The clock signal ensures that both the master and slave are synchronized during the data exchange, with data typically shifted out on one edge of the clock (rising or falling) and sampled on the opposite edge.

SPI is known for its high-speed data transfer capabilities and simplicity. It is commonly used in applications requiring fast and reliable communication, such as interfacing with sensors, memory devices, and display controllers. However, it requires more pins than other serial protocols, such as I2C, due to the separate lines for data transmission and clock signals. [38]

SPI would have been implemented in the communication between the metering PCB and the Raspberry Pi. One disadvantage of SPI is the fact that it requires four or more pins to work. For our case, this could be a possible downside as we may have a limitation with the number of pins we can utilize on the Raspberry Pi. SPI may also be overly complex for our use. We would not be able to take advantage of full duplex communication as our metering PCB only needs to send information to the Raspberry Pi, not the opposite.

4.1.8.3 I2C Standard

Inter-Integrated Circuit (I2C) was first developed by Philips Semiconductors in 1982. I2C is primarily used on microcontrollers to read hardware monitors, sensors, and to control actuators as well as many other uses. A notable strength of I2C is its ability to communicate between multiple devices with only two wires making it a very simple protocol in terms of wiring complexity. An advantage of this is that it is good to use in space limited scenarios.

I2C (Inter-Integrated Circuit) is a synchronous, multi-master, multi-slave serial communication protocol commonly used for connecting low-speed peripherals to processors and microcontrollers. It operates over two bidirectional lines: SDA (Serial Data) and SCL (Serial Clock). These lines are pulled high with resistors and are shared among all devices on the bus.

The protocol is based on a master-slave architecture, where the master device initiates communication and generates the clock signal on the SCL line. Each device on the I2C bus has a unique address, which the master uses to identify the target slave device. When the master wants to communicate with a slave, it sends a start condition by pulling the SDA line low while SCL is high, followed by the slave's address and a read/write bit. If the addressed slave is present, it acknowledges the request by pulling the SDA line low during the acknowledgment clock pulse.

Data transfer on the I2C bus occurs in bytes, with each byte being followed by an acknowledgment bit from the receiving device. The master controls the timing of the data transfer by generating clock pulses on the SCL line. During data transmission, the SDA line can change state only when the SCL line is low, ensuring data integrity. At the end of the communication, the master sends a stop condition by releasing the SDA line to go high while SCL is high.

I2C is highly efficient for short-distance communication and allows multiple devices to share the same bus without requiring additional select lines, making it ideal for use in embedded systems. However, its speed is limited compared to other protocols like SPI, with standard modes supporting up to 100 kHz and high-speed modes reaching up to 3.4 MHz. Despite these limitations, I2C's simplicity and versatility make it a great candidate for our project. [39]

In our project, I2C was chosen because it requires only two communication lines, SDA and SCL, which helped minimize the number of traces on the PCB and reduced overall complexity. Additionally, its ability to support multiple devices on the same bus allowed for easy integration of the ADS1115 and ESP32 without the need for additional connections. This made I2C an efficient and reliable choice for ensuring accurate and timely data transmission between components on our PCB.

4.1.9 Software Design Standards

Software design was the process of programming our hardware. It was important to follow basic standards of software design to ensure the efficient creation of code in a timely manner. It was also crucial to reduce confusion among team members when working on the code, especially when multiple people were collaborating on the same program.

Software played a pivotal role across all phases of our project, encompassing tasks such as programming our metering PCB, implementing our EMS onto the Raspberry Pi, and developing our user interface to be displayed on the touchscreen. Writing detailed comments throughout our code was a fundamental practice that significantly contributed to maintaining cleanliness and ensuring high-quality code. Comments played a crucial role in aiding comprehension for both present and future developers working with the code. They documented insights into the logic, purpose, and functionality throughout the code, enhancing readability and facilitating better collaboration within the team. Commenting streamlined the processes of reviewing and debugging, making them more efficient and effective.

4.1.10 Wi-Fi Standards

Wi-Fi is a set of wireless network protocols rooted in the IEEE 802.11 standards. It facilitates local area networking and Internet access by allowing digital devices to exchange data via radio waves. Wi-Fi networks are extensively used in homes, small offices, and public spaces like coffee shops, hotels, libraries, and airports. These networks connect devices to wireless routers and access points, providing seamless Internet connectivity globally.

Wi-Fi utilizes various IEEE 802 protocol standards and seamlessly integrates with Ethernet. It allows compatible devices to network through wireless access points, connecting with both wired devices and the Internet. Different versions of Wi-Fi are defined by IEEE 802.11 protocol standards, employing different radio technologies that dictate radio bands, maximum ranges, and achievable speeds. The most common radio bands for Wi-Fi are 2.4 GHz and 5 GHz, with newer generations also utilizing the 6 GHz band. These bands are subdivided into multiple channels, which can be shared among networks. However, only one transmitter can transmit on a channel within range at any given time. [40]

Wi-Fi was a vital part of our project. It was used for communication between the Raspberry Pi and the UI application. Specifically, we used UDP to communicate wirelessly between the two components. For this, we specified the IP addresses and ports of both components to facilitate the data transmission.

4.1.11 Bluetooth Standards

Bluetooth is a widely used short-range wireless technology for connecting mobile and fixed devices over short distances, typically up to 10 meters. It operates on UHF radio waves within the 2.402 GHz to 2.48 GHz ISM bands and is managed by the Bluetooth Special Interest Group (SIG).

Originally standardized as IEEE 802.15.1 by IEEE, Bluetooth is now overseen by the Bluetooth SIG, which develops specifications, manages the qualification program, and

protects trademarks. Manufacturers must meet SIG standards to market devices as Bluetooth-enabled, with technology licensing enforced through a network of patents. [41]

Bluetooth offers a compelling solution for our project by enabling seamless communication between the metering PCB and the Raspberry Pi without the constraints of traditional wired connections like UART, I2C, or SPI. Implementing Bluetooth on our PCB is very simple to integrate, requiring only the addition of a Bluetooth module. This approach also enhances flexibility and scalability within our system. By leveraging Bluetooth technology, we ensure efficient data transfer while maintaining robust connectivity standards, thereby advancing the functionality and adaptability of our project. We opted not to incorporate bluetooth into our project as communication via UART proved to be the simplest method.

4.1.12 PCB Standards

PCB standards play a crucial role in ensuring the quality, reliability, and performance of printed circuit boards. One of the primary organizations responsible for setting these standards is the IPC (Association Connecting Electronics Industries). The IPC is a global trade association that develops and publishes standards for the electronics industry, covering everything from design and manufacturing to assembly and testing.

One of the most widely recognized standards is IPC-2221, which provides generic requirements for the design of PCBs and other forms of component mounting or interconnecting structures. This standard covers a range of topics, including material specifications, mechanical properties, and electrical characteristics, ensuring that PCBs meet a baseline level of quality and performance. [42]

The IPC-J-STD-001 standard, titled "Requirements for Soldered Electrical and Electronic Assemblies," is a globally recognized benchmark for the soldering process in electronics manufacturing. This standard outlines the materials, methods, and verification criteria necessary to produce high-quality soldered connections. It encompasses a wide range of topics, including the proper handling and storage of soldering materials, soldering techniques, and inspection criteria for solder joints. Adhering to IPC-J-STD-001 ensures that soldered assemblies meet stringent reliability and performance criteria, making it a crucial reference for manufacturers aiming to achieve excellence in electronic assembly. [43]

The IPC-6012 standard specifically addresses the performance and qualification requirements for rigid PCBs. It outlines the necessary testing and inspection procedures to ensure that the PCBs can withstand their intended operating conditions, covering aspects such as material integrity, thermal stress, and electrical performance. [44]

Adherence to IPC standards ensured that our PCBs were reliable, safe, and of high quality. It also facilitated better communication between designers, manufacturers, and

customers by providing a common language and set of expectations. For example, when we designed a PCB to handle high currents, such as our 10A application, following IPC standards helped ensure that the trace widths, copper thickness, and other critical parameters were appropriately specified and consistently met.

4.2 Design Constraints

Design constraints are crucial in any project as they establish the boundaries within which we must operate, encompassing technical limitations, budgetary constraints, project timelines, and available resources. By defining these parameters early in the project, constraints ensure efficient use of resources, focused decision-making, and proactive management of project risks. They also stimulate creativity by prompting innovative solutions within defined guidelines, fostering a disciplined approach to problem-solving and project execution. Embracing constraints as integral to project planning encourages high-quality outcomes that meet academic requirements and project objectives effectively.

4.2.1 Load Constraints

Throughout the brainstorming process for S.H.I.E.L.D. we knew we might have some limitations with our loads. The solar panel we are using is a Renogy 100W monocrystalline solar panel. Originally our idea was to have an option where the solar panel would directly power the loads, this means that we would be limited to a load of 100W. However, with more research we determined this would be unrealistic to how a smart home would work in the real world. Instead, our load will now be limited by a battery that is charged via the solar panel.

If we are to use a single 12V 100Ah battery, our maximum power capacity is 1200Wh. Depending on the type of battery chosen, it is not possible to utilize 100% of this power. Typically with lead acid batteries, only 50% depth of discharge can be utilized. This means that if we use one 1200Wh lead acid battery, we will only have approximately 600Wh to work with. This may not be a problem depending on how long we plan to have our model on display for.

It is known that we will be needing light bulbs to represent DC and AC loads. Due to our 600Wh limitation, we will most likely select incandescent bulbs as they only consume around 2 to 15 watts of power. Taking a charitable assumption, we can suspect our bulbs to take around 10 watts each, making up for a total of 30 watts. Keeping in mind that we have a total of 600Wh of head room, we do not want a significant amount of power to be used on just the bulbs. Because we also have AC loads to account for, which will consume more power than light bulbs, we will want some slack in our system. Typical AC loads can consist of fans, television screens, and phone chargers. Desktop fans consume around 35 watts of power to function. Television screens, depending on their

size, consume around 30 watts. Lastly, phone chargers consume anywhere from 5 to 65 watts depending on how fast the charger is rated for.

Looking at the system with our chosen loads:

- 3 LED light bulbs → 3 * 10W = 30W
- Desktop fan → 35W
- Small television screen → 30W
- Phone charger → 20W

Our system, in this configuration, totals to 115W.

If we want to look at our maximum time to run this hypothetical configuration, we can divide 600Wh by 115W which gives approximately 5.2 hours of use.

With this configuration, our model smart home only has four different types of loads with not much head room to expand. If we decide we may need more loads, our options would be to purchase an additional battery or to rather invest in a higher quality lithium battery with a higher depth of discharge.

The configuration we went with included an incandescent bulb, an LED bulb, a DC fan, and a LED light strip.

4.2.2 Charge Controller Constraints

Several technical constraints arose during the planning process of S.H.I.E.L.D. One of these factors involves choosing the power source settings on our charge controller. Initially, as stated in the previous section, we planned on having the option for the solar panel to prioritize the load and have the excess charge the battery. Multiple videos online confirmed that this configuration was possible. Unfortunately this configuration was not confirmed in documentation nor by the team at Renogy. There is no definite answer to this unless we do some testing of our own. Because we are trying to create our model as realistic as possible, we ultimately opted to discard this idea as typical smart homes do not utilize this configuration in the real world. This will simplify our project as well as make it more accurate. It will also eliminate time troubleshooting.

Compared to other brands of charge controllers on the market, our Renogy Wanderer charge controller is very barebones in terms of settings offered. The Renogy Wanderer does not easily allow the user to program their own load output settings. For example, a Renogy competitor known as Victron Energy, has a similarly priced charge controller named the BlueSolar MPPT solar charger. This charge controller allows the user to easily program battery voltage limits to control the load. By setting upper and lower voltage limits, you are able to turn the load off and on depending on your use case. Unfortunately,

the Renogy Wanderer does not offer such settings which thankfully did not lead to any constraints during the prototyping phase of our project.

4.2.3 Raspberry Pi Constraints

Another technical constraint is how our metering PCB and Raspberry Pi are going to be connected and relay information. The Raspberry Pi has four USB ports that can be used for serial communication. It also has a 40-pin general-purpose input/output (GPIO) header. There is a lot of information and control that the Raspberry Pi must handle. Firstly, it has to comprehend and make use of our metering information. This will have to be transferred from our metering PCB using either one of the four USB ports, or through the use of the GPIO pins. Secondly, The Raspberry Pi will need to control the relays throughout the system. This can only be done through the use of the GPIO pins. Depending on the number of relays required in our system and the available GPIO pins on the Raspberry Pi, there may be constraints on how we can send our metering information. The Raspberry Pi has multiple communication protocols it can use through the use of different GPIO pins. We have the option for SPI, I2C, and UART communication. This could also pose a limitation. There may also be constraints with how many relays we can have throughout our system because of this. Overall, we should continue to be mindful about the limit of relays, depending on how many loads we choose to have within our system.

4.2.4 Photovoltaic Panel Constraints

Photovoltaic Panels (PV) have many configurations depending on the amount of panels you have available. Configurations can include series, parallel, and a mix of the two depending on different factors. These factors are typically the voltage level you want the system to operate at and the max current your chosen charge controller can handle. Charge controllers typically become more expensive the higher their current rating is. This is why it is most common for PV panels to be wired in series as the current entering the controller will be the same no matter how many panels you have. Wiring in series will result in the addition of voltages whereas wiring in parallel will result in the addition of currents from each panel. This is known from kirchoff's voltage and current laws.

Figure 8 below shows a basic series configuration. As said before, a series configuration is beneficial to increase voltage in a system. Doing this does not change the amount of current going to the charge controller. Increasing voltage in the system can also make the system more dangerous, this is important to be aware of in order to take all safety precautions. Most charge controllers are capable of running at different voltage levels but will usually have a fixed current limit. The charge controller we have selected, the Renogy Wanderer, is capable of running either a 12 or 24 volt system but has a fixed current limit of 10 amps. By doing a few calculations, we can determine if a series configuration would be possible with our system. Assuming we are using two 100W, 12V

solar panels, we can divide the watts by the voltage to find the current output of the panel. Doing this reveals that a single 100W, 12V solar panel will output 8.33A.

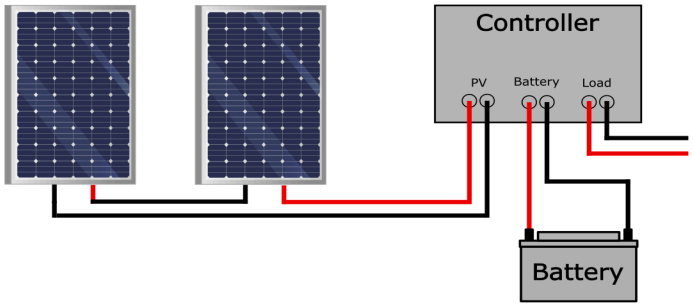


Figure 8: Series Configuration of PV Panels

Knowing these numbers are very important when adding to our system. Using these values we can find that our system, with this series configuration, will operate at 24V and will output 8.33A. This means that our system would be able to handle this configuration, given that we also have a 24V battery. Figure 9 shows a parallel configuration. A parallel configuration is beneficial to keep a low voltage. However, doing this will increase the current output of the panels. It is typically recommended to connect panels in series, though there are cases where connecting in parallel may be needed. Using our same selected charge controller, our limits are 12/24V and 10A. Using the values found earlier, a parallel configuration will operate at 12V and the current output of each panel will be added making 16.66A.

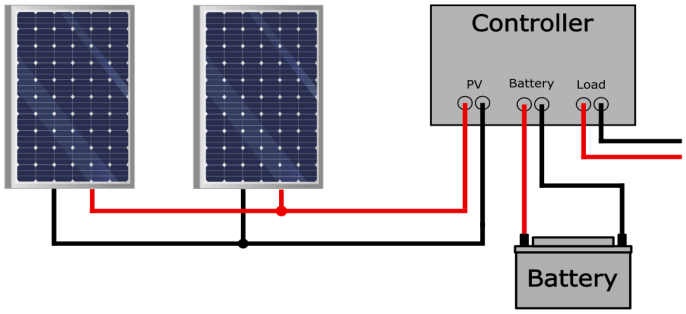


Figure 9: Parallel Configuration of PV Panels

A parallel configuration would not be possible with our system because the current output would exceed the limit of the charge controller. If we were to expand our system, it is important to know that we would have to use a series configuration and would have to step the voltage of the system up to 24V. This would most likely mean we would be required to purchase another 12V battery to wire in series with our existing one. This may be out of the scope of our project, however, it is something to be kept in mind for the future of this project.

Our project utilized one 100W, 12V solar panel. As found earlier, this panel will output around 8.33A making it fully compatible with our chosen charge controller. The only downside of using a single panel would be the charge time of our battery. This can be found via a simple calculation. We know our battery will have around 600Wh that will need to be recharged. By simply dividing this by the rated power of our panel we get 6 hours of charge time required. This may pose some limitations when presenting our project. If we were to have added another panel, it would have cut the charge time in half, taking only 3 hours to charge. This is something we can be mindful of for the future development of our project.

4.2.5 PCB Constraints

When designing a PCB for high current applications, one of the most critical factors to consider is the trace width. The width of the PCB traces must be sufficient to handle the high current without excessive heating. Wider traces reduce electrical resistance, which in turn minimizes the amount of heat generated due to the flow of current. This is essential for maintaining the reliability and longevity of the PCB. The required trace width can be calculated based on the amount of current it needs to carry and the acceptable temperature rise. In addition to width, the thickness of the copper traces, typically measured in ounces per square foot, also plays a crucial role. While standard PCBs use 1 oz/ft² copper, high current applications may require thicker copper, such as 2 oz/ft², 3 oz/ft², or even more, to effectively manage higher currents. This combination of wider and thicker traces ensures that the PCB can handle the desired current levels without excessive heating, thereby maintaining performance and preventing potential damage. Proper calculation and design of trace width and thickness are therefore fundamental steps in the PCB design process for high current applications.

Our PCB is designed to handle a current load of up to 10A. While this is not an exceedingly high amount of current, it is crucial to take this into careful consideration during the design phase to ensure reliability and performance. Trace width calculation becomes a pivotal aspect in this context. Properly designed traces will manage the current flow, minimizing heat generation and ensuring the integrity of the PCB over time.

Fortunately, the process of determining the appropriate trace width for our application is made straightforward by using online calculators, such as those available on Digikey's website. These tools simplify the calculations by allowing us to input our specific requirements, such as the current load, the acceptable temperature rise, and the thickness of the copper. By utilizing these resources, we can accurately determine the optimal trace width, ensuring that our PCB can handle the 10A current without issues. This approach not only enhances the safety and durability of the PCB but also streamlined the design process, allowing us to focus on other critical aspects of the project.

4.2.6 Scope Constraints

For any project with a limited time frame and a small team, there are bound to be constraints within our scope. As a reminder, the underlying goal of our project is to create a model smart home with multiple DC and AC loads to create a realistic representation of a smart home. This model smart home must also have measurements taken throughout the system in order to simulate it accurately on the OPAL-RT. Furthermore, because we are trying to create an educational tool, everything must also be displayed on the touchscreen.

Knowing our scope, it is important we do not expand outside of it to ensure our main goals are achieved in a timely manner. As stated in earlier sections, our project was centered at the edge of the grid. Our main focus is how the smart home affects the grid, not how the grid affects the smart home. This means we will only deal with the responses of the smart home itself rather than looking at what measures could be taken by the grid as a whole.

4.2.7 Economic Constraints

Throughout the planning of our project, recognizing our economic constraints has been crucial. Because of our sponsorship by the UCF Digital Grid Laboratory, we do not want to use excessive amounts of resources from them. It is important for us to realize this constraint and to plan around it in order to ensure the success of our project.

Our touch screen is one of the most expensive items that has been needed for the design of our project. Delightedly, this touch screen is not something that had to be purchased specifically for our team. Because it has been used in a previous senior design project, we were able to repurpose it for our use in the development of S.H.I.E.L.D. This significantly reduced our project's direct upfront costs.

One of the main price limiting factors of S.H.I.E.L.D. is the choice of battery. Batteries can become very costly depending on their size and the type chosen. For this reason, our choice of battery was one of the most economically influenced parts of our project. Knowing this, it is important to try to make the most cost-effective purchase. Given our sponsorship by the UCF Digital Grid Laboratory, we aimed to present the most cost-effective proposal. This is why it is also very important for us to do much research in this area in specific, in order to make an informed decision and to ensure we make the most of what is available for us.

Being sponsored has several advantages, including the coverage of our supply costs; however, it also entails certain limitations. One of which being the delay incurred when ordering supplies. This became disruptive in the prototyping phase of the project. Having to wait for supplies to be approved and then purchased can make progress on the project very slow, not to mention shipping wait times. This is why some items were purchased out of pocket by the team.

5. Comparison of ChatGPT with Similar Platforms

Over the past few years, Artificial Intelligence (AI) and Large Language Models have gained much attention from not only technology professionals, but also the general public. This technology has been greatly impactful on the development of various products, solutions, and tools. While there are several benefits to the utilization of this technology, it is important to acknowledge the several drawbacks or limitations that these pieces of technology offer.

5.1 LLM Platform Comparison

Large Language models are deep learning algorithms that "recognize, summarize, translate, predict, and generate content using very large datasets" [29]. These models are trained off of large datasets to generate content based on parameters given. One of the most well-known Large Language Model AI Tools are AI Chatbots, such as ChatGPT or Google Bard. In recent years, these tools have become more widely available to the public, making it a rapidly growing tool in various industries.

These tools offer various features that can be used in the design and implementation of different projects. One of the most known features is their ability to answer prompted questions with information. If a user is looking to clarify some scientific concepts, they are able to form a "query" and obtain specific information for their question. One aspect of this that must be accounted for is the fact that these tools can provide false information. Therefore, it is important to use these tools with caution. In addition to providing information, these tools also hold the ability to generate creative pieces, such as stories or poems. This allows the tool to be used as a resource for inspiration. As these tools begin their integration into society, it is vital to identify the various platforms available and analyze their strengths and weaknesses. Some tools are more fit for the creation of creative works, while others excel more at generating information.

5.1.1 ChatGPT

One of the most known tools in this space is ChatGPT. Released to the public on November 30, 2022, ChatGPT made a significant impression on the public with regards to AI and its utilization in society [30]. In its primary form, ChatGPT is a chatbot that receives user inputs and utilizes Reinforcement Learning with Human Feedback (RLHF) to reach a response relevant to the input [31]. Holding this capability allows ChatGPT to be a resource for various applications.

ChatGPT offers different versions that impact the response the user may obtain. Two notable versions offered are 3.5 and 4. Both versions allow the user to submit prompts to the model and retrieve the output. The main difference between the two is that 4 is significantly more advanced, providing more accuracy, allowing for visual inputs, etc

[32]. With all these capabilities it is important to understand the advantages and disadvantages of choosing to utilize ChatGPT.

ChatGPT offers great benefits in efficiency, scalability and multilingual support [33]. ChatGPT excels in collecting the prompts from the user and providing a quick turn around with a response. In terms of scalability, it can also handle a large amount of requests at once. This feature maintains the efficiency of the system without sacrificing the quality of the information it is providing. Another significant feature that ChatGPT exhibits is supporting several different languages. This opens up the tool to significantly more people in today's society.

In terms of drawbacks of ChatGPT, it is vital to fully understand the areas that the model is weak in, in order to avoid reliance on a tool that may not be reliable. Some notable disadvantages of ChatGPT include limited control generated content, outputs containing misinformation, and the fact that its knowledge is limited to 2021. Using AI sometimes opens up the risk of generating biased or inaccurate information. ChatGPT may pose a risk if used for specific tasks or solutions. Additionally, ChatGPT-3.5 is only up to date to September 2021. Therefore, it may not provide the most accurate information. ChatGPT can be an excellent resource for gathering information or generating content, but it is important to avoid relying solely on it for specific applications due the potential generation of false information.

5.1.2 Microsoft Copilot

Microsoft Copilot is Microsoft's AI LLM chatbot that is integrated in their Microsoft 365 Apps (Word, Excel, Powerpoint, Outlook, etc). Copilot is known for its use in providing context of the user's work. Microsoft stated that it is designed to assist in the process of creating, understanding and editing documents [34]. This can be helpful when reviewing documents or ensuring that you are conveying the correct message.

As mentioned above, Copilot primarily excels at providing context on the documents that you are working on. This can be incredibly helpful with reviewing your work and understanding the work of others. Additionally, Copilot can be used for content generation via input prompts. Copilot is most effective when integrating its usage with Microsoft products.

Looking at the disadvantages, there are concerns regarding privacy, over-reliance, and misinformation. Allowing co-pilot to analyze your files and documents raises concerns on the privacy of your information and the data it is seeing. Having this in mind, Copilot may not be a good option when working on projects containing sensitive information. Additionally, due to its integration with Microsoft products, there may be an over-reliance on the tools, which may lead to the generation of false-information. If someone is using this tool for a project, they may miss an important piece, causing several issues to arise. As mentioned before, co-pilot is primarily used for Microsoft

products and document analysis. This limits its capability for other tasks, such as generating solutions to large logistical problems.

5.1.3 Google Gemini

Google Gemini is Google's Large Language Model chatbot. Similar to ChatGPT, Gemini receives user inputs and generates content through its model. Notably, Gemini was trained with a dataset of text, code, images, and video [35]. Using this diverse dataset, the chatbot excels in processing information from these various sources. As with ChatGPT, it is important to understand the strengths, weaknesses and risks of using various tools.

When comparing different LLM platforms, there are several features that may make one platform more preferable over another. Google Gemini is known for excelling with writing, brainstorming, learning and more. As mentioned previously, Google Gemini was trained using datasets containing various source types. This opens up the capability for the model to accurately process inputs of these different types. Knowing these strengths, the user may opt to use Gemini when inputting various source types or creating different written pieces.

While Gemini excels in various areas, it also holds the risk of generating bad results. Similar to ChatGPT, Google Gemini is not regulated, so it may generate inaccurate or offensive content. As discussed, it's important to be selective on where to use these tools.

Due to the recent AI explosion, several companies and people are looking into how to best utilize this tool. While it provides great benefits in information gathering and automation, it is vital to understand the risk and weaknesses of the various tools. After analyzing the various capabilities of these platforms, we are now going to shift this discussion to view the impact of these platforms on this senior design project.

5.2 Project Impact

Following our discussion on these AI tools, the various platforms available and their pros and cons, we are going to discuss the impact of these tools on our Senior Design project. Very recently, AI tools have become greatly accessible to our society and have had an impact on learning and various applications.

Holding the ability to utilize AI is powerful when working on a project, but it is important to understand best practices when utilizing it to avoid the collection of misinformation and the potential risk that it brings to a project. As discussed in the section above, there are several AI platforms that excel in certain areas, but fall weak in others. In terms of this project, AI could offer several benefits in the design and implementation. However, there are many determinants that could impact the success and results from the project. Keeping these points in mind, this helped us decide on if or how

we wanted to use AI. In the following sections, we discuss both of these points and identify the impact that AI could have on our project.

5.2.1 Benefits

As mentioned in previous sections, AI has emerged and sparked several discussions on its role in our society. Having this tool readily available is extremely powerful and optimizes and impacts various tasks. In the context of our senior design project AI tools, such as ChatGPT may be useful as a resource for information.

In the context of this project, these tools have been helpful in the research stage, when trying to understand the problem presented and the various components it involves. For example, in the creation of the circuit for this project, the tool may be helpful in explaining the functionality fundamentals of the components, such as the inverter or solar panel. As the user, you are able to construct your input to ask specific questions. In the research stage of this project, this served as a great resource when looking into various approaches for the design of the project.

Another notable benefit of using these types of tools for research is that they can save a great amount of time in the design and troubleshooting of the design. Large language models allow the user to construct their prompts to obtain specific information. Being able to clarify specific pieces of information allows the user to save the time it takes to find the answers. In the context of this project, this was helpful in clarifying concepts.

AI could also be used for generating code when creating the software interface for the Smart Home digital twin. This has assisted with creating the necessary components and features in the application, saving time in the development of the software.

5.2.2 Detriments

While these AI tools have many benefits, there are also many detriments that may impact the reliability of these tools. From spreading misinformation to causing heavy reliance on these tools, there is definitely risk associated with the use of AI. For this project, these risks could greatly impact its success, so it is vital to understand how to use these tools effectively.

One of the greatest drawbacks of using Large Language Model tools such as ChatGPT is the fact that they sometimes generate false information. In specific situations, receiving this false information could greatly impact the effectiveness of the system implementing it. In these scenarios, it is extremely important to avoid heavy reliance on these tools. If there is a system that fully integrates AI, there is risk associated with the results it produces. In terms of this project, any information collected from AI should be fact-checked, as it is important to collect accurate and reliable information. The risk

associated with the use of AI and its tendency to provide false information makes it a tool that may not be suitable for several applications.

Additionally, when obtaining information from AI, it does not provide where it collected the information, so there are no citable sources associated with the information. Therefore, it could be argued that these tools serve as a better resource than source. This contributes to the idea that it is not always good to heavily rely on these tools, as you would not be able to reference where the information is gathered.

AI could also have severe impacts when generating code. When AI generates code, the user is not as involved in the creation and understanding of the system functionality. If the program runs into issues or bugs, the developer may struggle with identifying how to troubleshoot due to lack of knowledge of the system. This introduces great risk to the project. With this in mind, we are opting to not use AI for generating code for the system.

Being able to generate great answers to questions with these tools makes them quite powerful and effective. However, with these tools readily available, dependence on them can start building, which could cause users to avoid fact-checking and collecting false information. For example, as companies begin to look into the integration of these tools into their processes, it is important to understand their risk. For this project, heavy dependence on AI could greatly impact the success of the system we are creating. If AI produces false inputs, this could cause major issues with the success of the project.

5.3 LLM Platform Conclusion

In reference to the discussion above, there are several points made about the benefits and risks associated with the use of different AI platforms. As these tools begin growing within our society, it is important to understand the impact they may have on your projects. Various industries are looking to this type of technology for automation, creation of tools and more.

Overall, our team did not rely heavily on these tools, as the risk associated with them could impact the success of our project. While AI presents several benefits and opens up many doors, there are many risks associated with the information it provides. As mentioned previously, AI is known to sometimes provide misinformation to the user. Seeking to avoid the utilization of false information, our team ensured that the information used in the design and implementation phases of this project were reliable. Additionally, the risks of using AI generated code could introduce large issues in the implementation and utilization of our system. We chose to avoid generating any AI code due to that risk. With all this in mind, it is vital to do thorough research on the tools and information being used for design and implementation. Researching the tools being used for your project can help lead you to success in its implementation.

6. Hardware Design

An overview of the hardware connections is shown in Chapter 2. The block diagram presented shows all inputs and outputs of each block, both internal and external. This aids the team on the general idea of what components will be connected to each other and how the system functions. Beyond this, it is vital to create block diagrams for subsystems within the overall system. This will be discussed in the sections that follow.

6.1 Subsystem Block Diagram

6.1.1 Raspberry Pi Integration with Relays

The first subsystem discussed involves the system of relays controlled by the Raspberry Pi. The relays work to control the sources and loads connected to the system. Each relay is toggled using one of the many GPIO pins on the Raspberry Pi. Our energy management algorithm automatically toggles each relay based on the metering information sent to it. These GPIO pins are manually wired to the relays. To confirm the right relays are configured to the Raspberry Pi, it is important for us to give appropriate labels for the pins controlling the relays. This labeling is done during programming, simplifying the development process.

To ensure the safety of our system, our relays are configured in the “normally open” state. This design choice is crucial because, in the event of a power failure to the relays, they default to an open circuit, effectively preventing any of the loads from being powered. This safety measure is particularly important as it guarantees that we can always disconnect a power source when necessary, thereby avoiding potential hazards or damage. By prioritizing this safety feature, we enhance the reliability and security of our system, ensuring it operates safely under all conditions.

Our relays operate across a wide voltage range, from the battery’s 12V up to mains voltages of 120V. It is crucial that our relays withstand these voltage levels to ensure reliable performance. Fortunately, our relays are rated for up to 250VAC and 30VDC, making them well-suited to handle the required voltage range with a significant margin of safety. We are not utilizing all the relays on our eight-channel relay module. This approach provides the potential for future expansion of our project and, more importantly, offers redundancy in case of relay failure.

Figure 10 below shows an example of how the relays are connected to our system. This example only shows one of the channels being used, but multiple channels are used in our project.

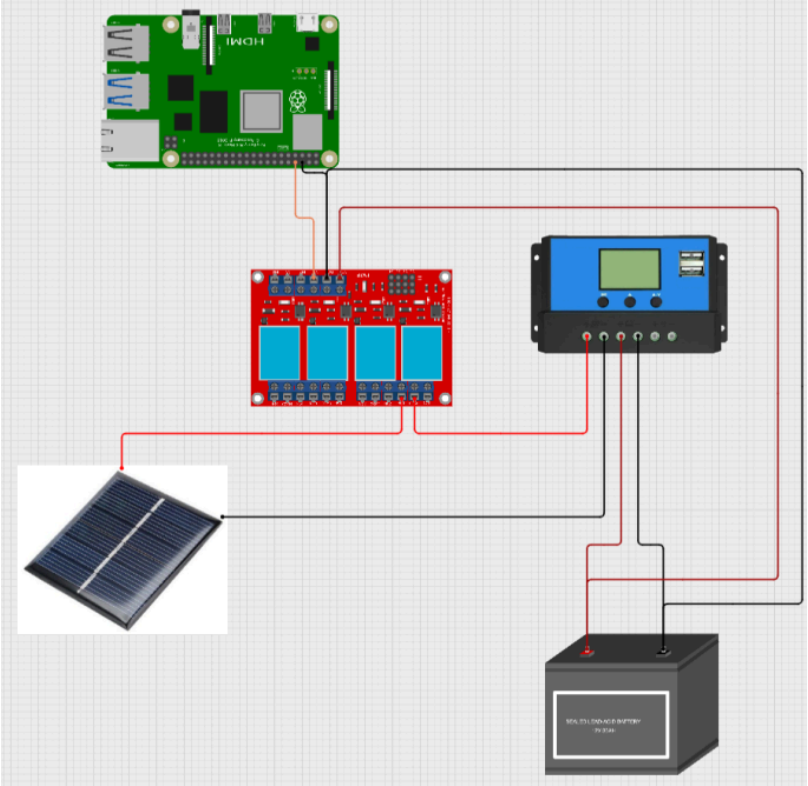


Figure 10: Example Connection of Relays to Solar Panel

6.1.2 Raspberry Pi Integration with Metering Devices

The second subsystem that will be discussed involves the system of metering devices communicating with the Raspberry Pi. In this project, the metering devices serve an important role. They must collect the current, voltage, and power measurements from all the sources. This enables the user to see the measurements on the touch screen and allows the OPAL-RT simulator to receive measurements from the testbench. The DC metering devices for our project were created by the electrical engineers on the team. Two PCBs were designed: one for metering the solar panel and one for metering the battery. For both metering PCBs, a consistent subsystem was followed. The PCBs included a buck converter, a current sensor, voltage sensor, analog-to-digital converter, USB and USB-UART connector, and the ESP32. Figure 11 shows the subsections of the PCB design. These are the key components of the metering PCB.

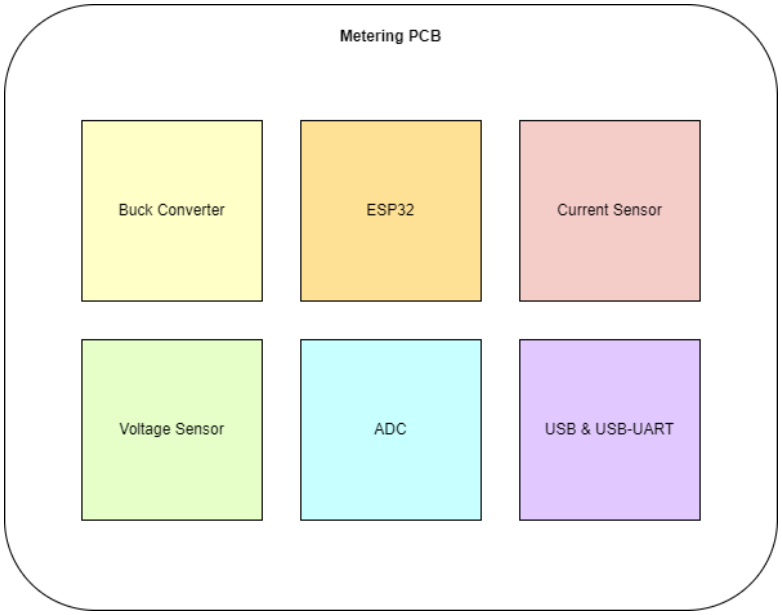


Figure 11: DC Metering PCB Subsections

There were a few important things to note with the DC metering PCB. Firstly, the current sensor's input voltage was 5V while the ESP32's input voltage was 3.3V. Secondly, the design needed to integrate a USB in order to program the ESP32. Thirdly, the ESP32 is able to perform analog-to-digital conversions on certain GPIO pins, but it is more accurate to have an external analog-to-digital converter that feeds into the ESP32. With these things in mind, we began to design the DC metering PCBs. All of the measurements collected by the ESP32 would be sent over UART to the Raspberry Pi.

For the AC load metering, the team decided to utilize current transformers to send the information back to the Raspberry Pi. The AC load metering was set to take place right after the inverter and right after the AC source connection (wall outlet). The current sensors were capable of collecting current and voltage measurements. On our program, we were able to calculate power from these measurements.

6.2 PCB Schematic Diagram

The PCB schematic serves as the blueprint of our design. It provides a detailed diagram of the electronic components and their connections throughout the system. Schematic diagrams helped with the physical placements of components on the board, allowing us to know which components should be placed with each subsystem. PCB layout plays a crucial role in determining the operation and electrical performance of the design. Therefore, adhering to industry-recommended standards such as IPC-2221 is essential. It is highly advantageous to use a PCB software package that supports both schematic and layout design. KiCad was chosen for this project as it is a free and open-source tool,

making it accessible to a wide range of users. Being open-source means that the software is continuously improved by a community of developers, ensuring it stays up-to-date with the latest advancements and user needs.

KiCad integrates a comprehensive schematic and PCB layout capabilities and offers valuable features such as a built-in BOM generator, a robust library management system, and seamless integration with external tools for design verification and manufacturing quotes, enhancing the overall design workflow. Additionally, KiCad supports a variety of plugins that extend its functionality. These plugins provide additional capabilities such as enhanced design rule checks, 3D visualization, automated routing, and more. The plugin ecosystem allows users to customize and optimize their design environment to better suit specific project requirements, making KiCad a versatile tool for PCB design.

6.2.1 Metering PCB

Figure 12 shows the schematic of our microcontroller unit. During the creation of this schematic, each element's datasheet was referenced to ensure the interoperability of these components. For instance, decoupling capacitors were placed near power supply pins of the various integrated circuits within our system. These capacitors stabilize the voltage supply to components and filter out high-frequency noise, ensuring that the components function as intended. We placed an LED on pin IO2 to verify the functionality of our ESP32. We programmed a simple code to make the LED blink. This verified the functionality of the ESP32.

Figure __ shows the two buttons used to program the ESP32. These buttons were needed to put the ESP32 into programming mode. If they were not incorporated, the ESP32 would not know that it needed to read the data being sent to it.

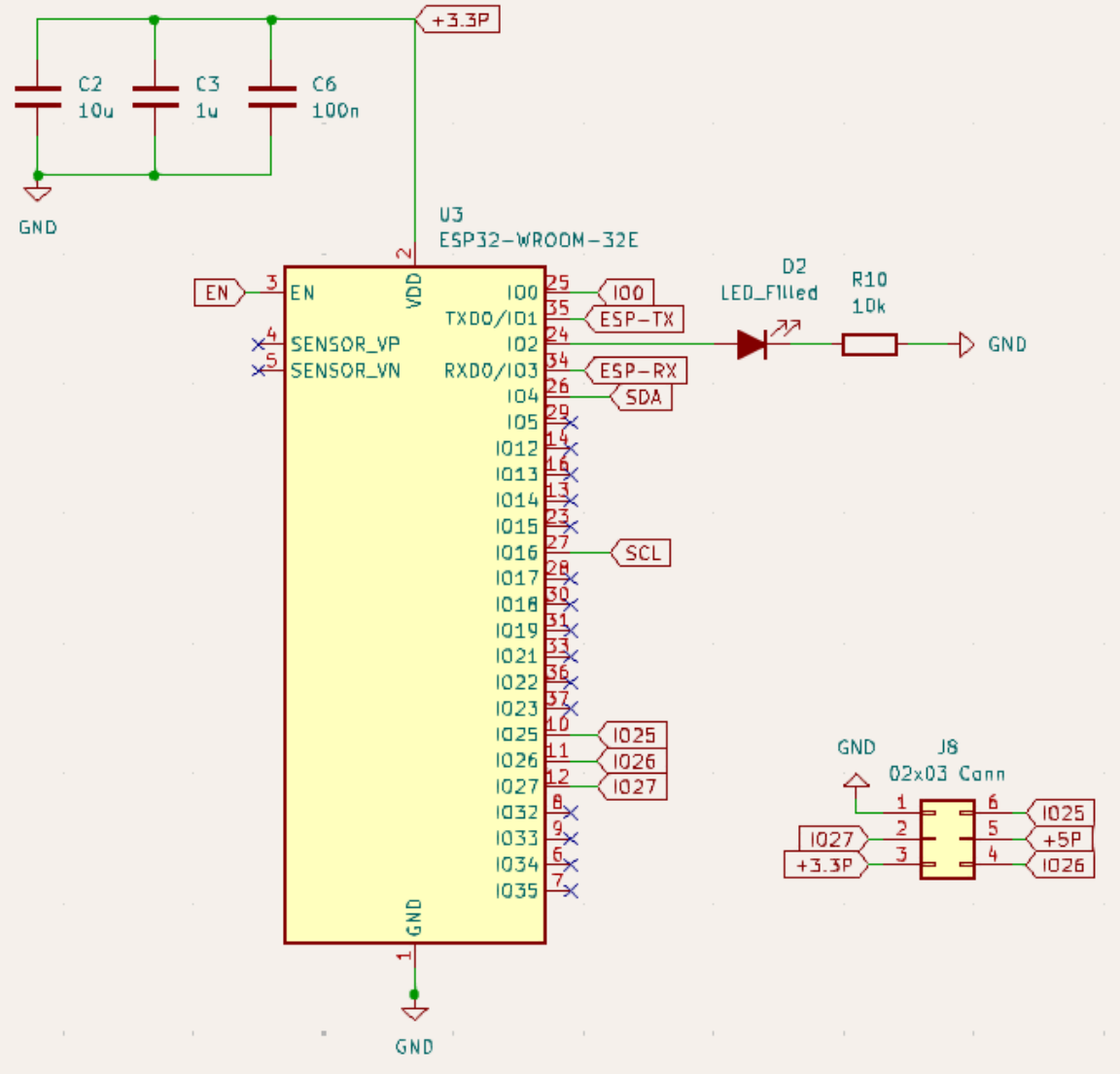


Figure 12: Schematic of Microcontroller Unit with a 3x2 Pin Header for Troubleshooting

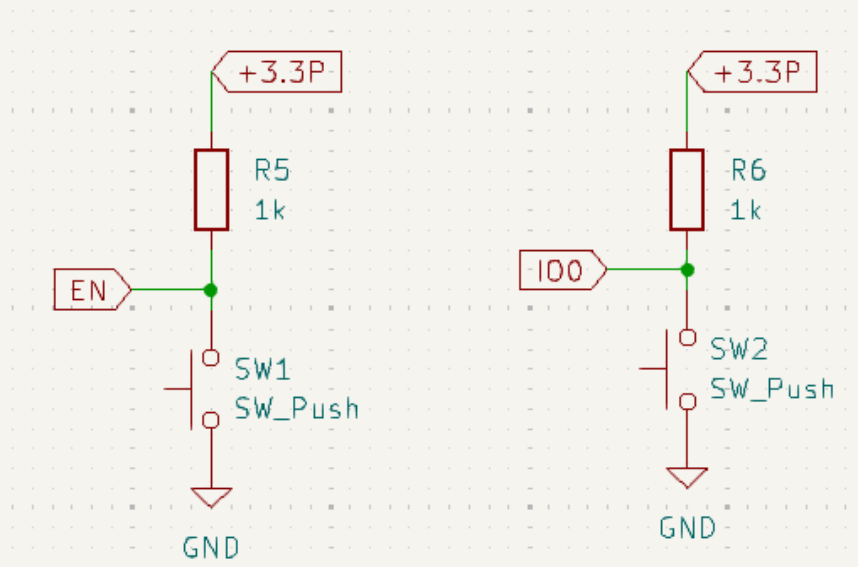


Figure 13: Programming Buttons For The ESP32

Figure __, shown below, shows the current sensors and voltage dividers used to obtain measurements from the battery and PV panel. The nodes we read are ViOUT1, ViOUT2, VOUT1, and VOUT2. These were fed into our 16-bit ADC. The ADC we chose is called the ADS1115, shown in figure __.

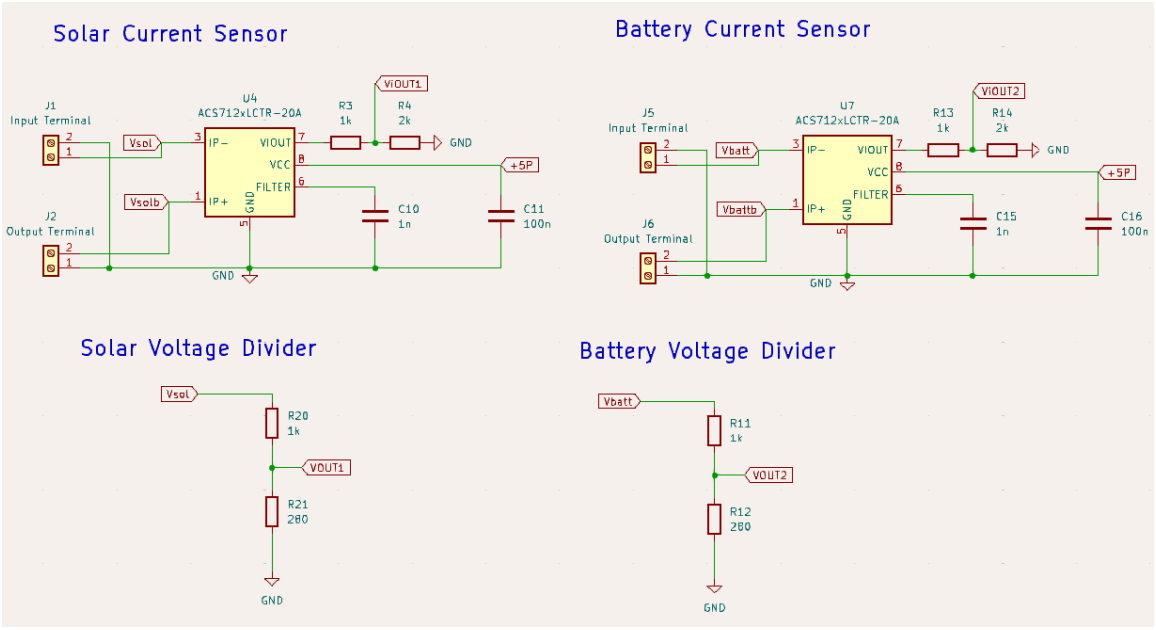


Figure 14: Current and Voltage Sensor Circuits

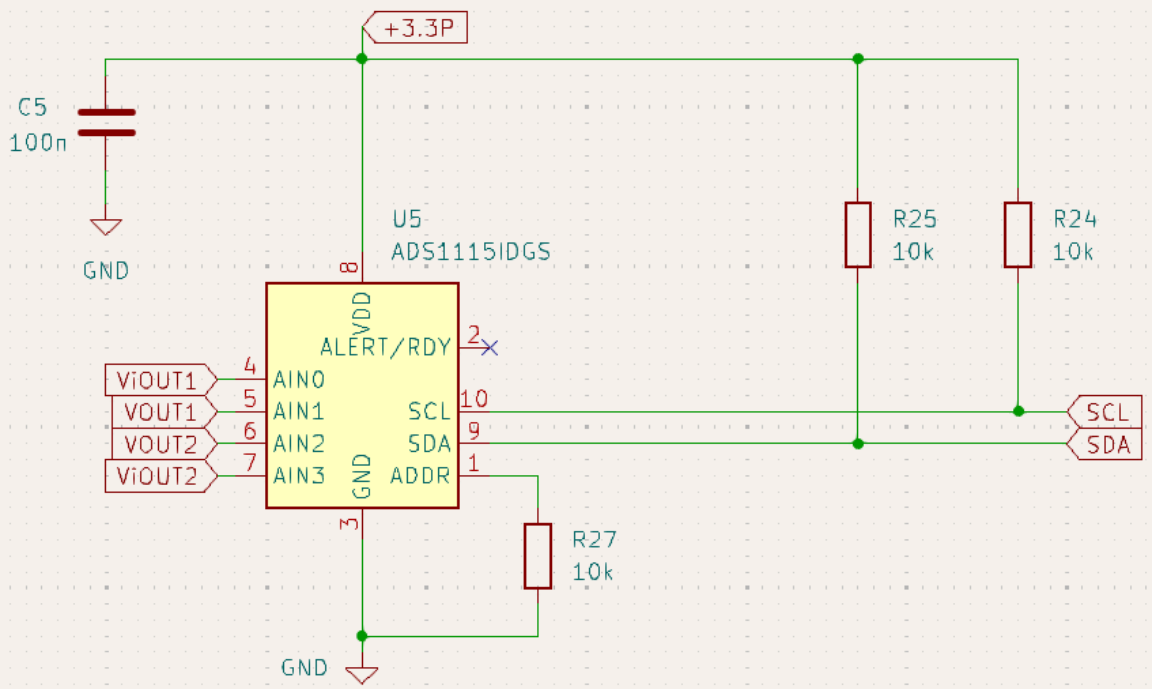


Figure 15: ADS1115 16-Bit ADC Circuit

Shown in figure 14 below is our USB to UART bridge. This circuit allows the ESP32 to send and receive serial data. This same USB connection is used to power the board as well. The USB to UART converter we used is called the CH340C. Three capacitors were used to filter the VBUS line to stabilize the power being supplied. Shown in figure __ is a header we used to verify our TX and RX connections were correct. It is known that each USB to UART chip has a different way to label TX and RX. For this reason, we chose to add this header to eliminate any problems that could have arisen from it.

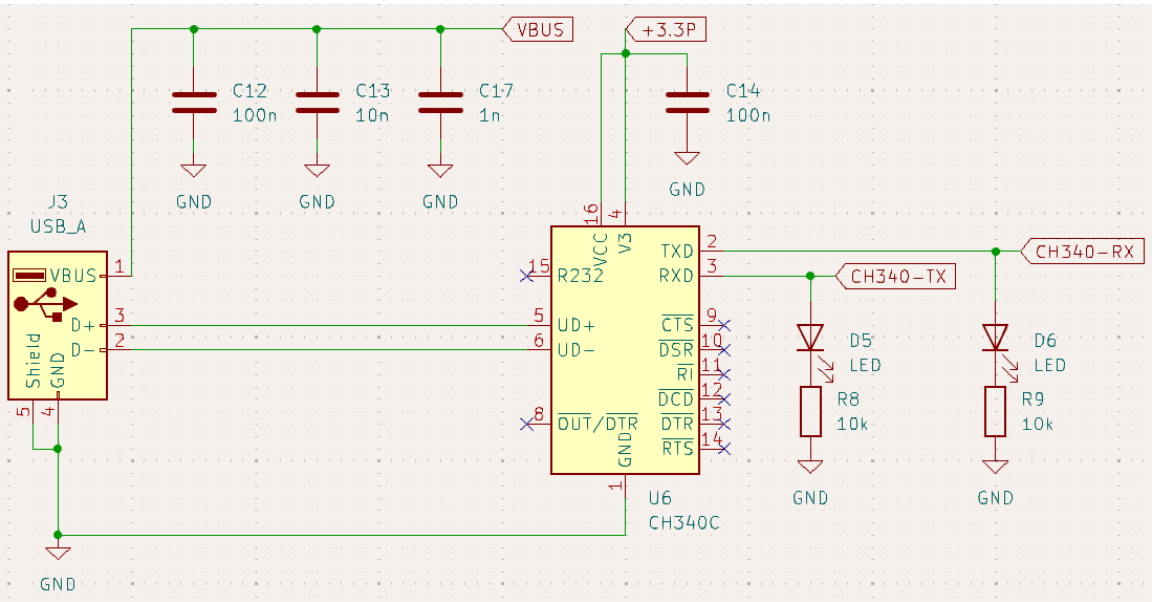


Figure 16: USB to UART Bridge

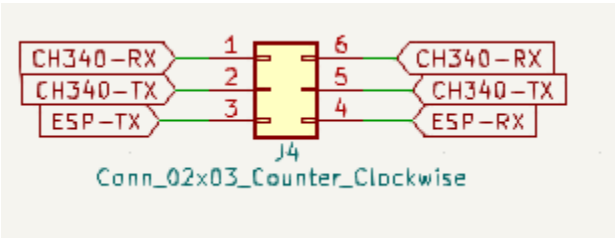


Figure 17: 3x2 Header to Select TX and RX Connections

The last circuits shown in Figure __ and __ are our DC to DC converters. The incorporation of a DC to DC converter is vital for ensuring stable and reliable operation of electronic circuits. These converters allow us to step down or step up the voltage from a power supply to match the specific requirements of different components on the board. This adaptability is crucial for maintaining consistent performance, minimizing voltage-related issues, and optimizing power efficiency. By integrating a DC to DC converter, we can also eliminate the need for multiple power sources and improve overall system reliability, making it an essential component in modern PCB designs. Our system's DC to DC converters were designed using TI's Webench power designer. This website enables users to input a range of input voltages and specify a desired output voltage, and then generates a wide selection of circuit options for them to choose from. Each circuit has an efficiency rating and an estimated price. We chose the circuits with the highest efficiency while maintaining a lower price.

We created two sets of DC to DC converters, one to be used when power is from the USB, and another to be used when we choose to be powered by the battery and PV panel.

We originally had our PCB being powered by only the battery and PV panel, but after receiving the first iteration of the PCB we noticed our current measurements were not as accurate as we were hoping. This is because we were metering the very thing we were taking power from. For this reason we chose to power the PCB with mainly the USB connection. We kept the battery and PV panel connection in the design just for redundancy purposes, in case something went wrong with the USB connection. To switch between the power sources, we had a 3x2 pin header connection with jumpers to choose which power source to use. This can be seen in figure 18 below.

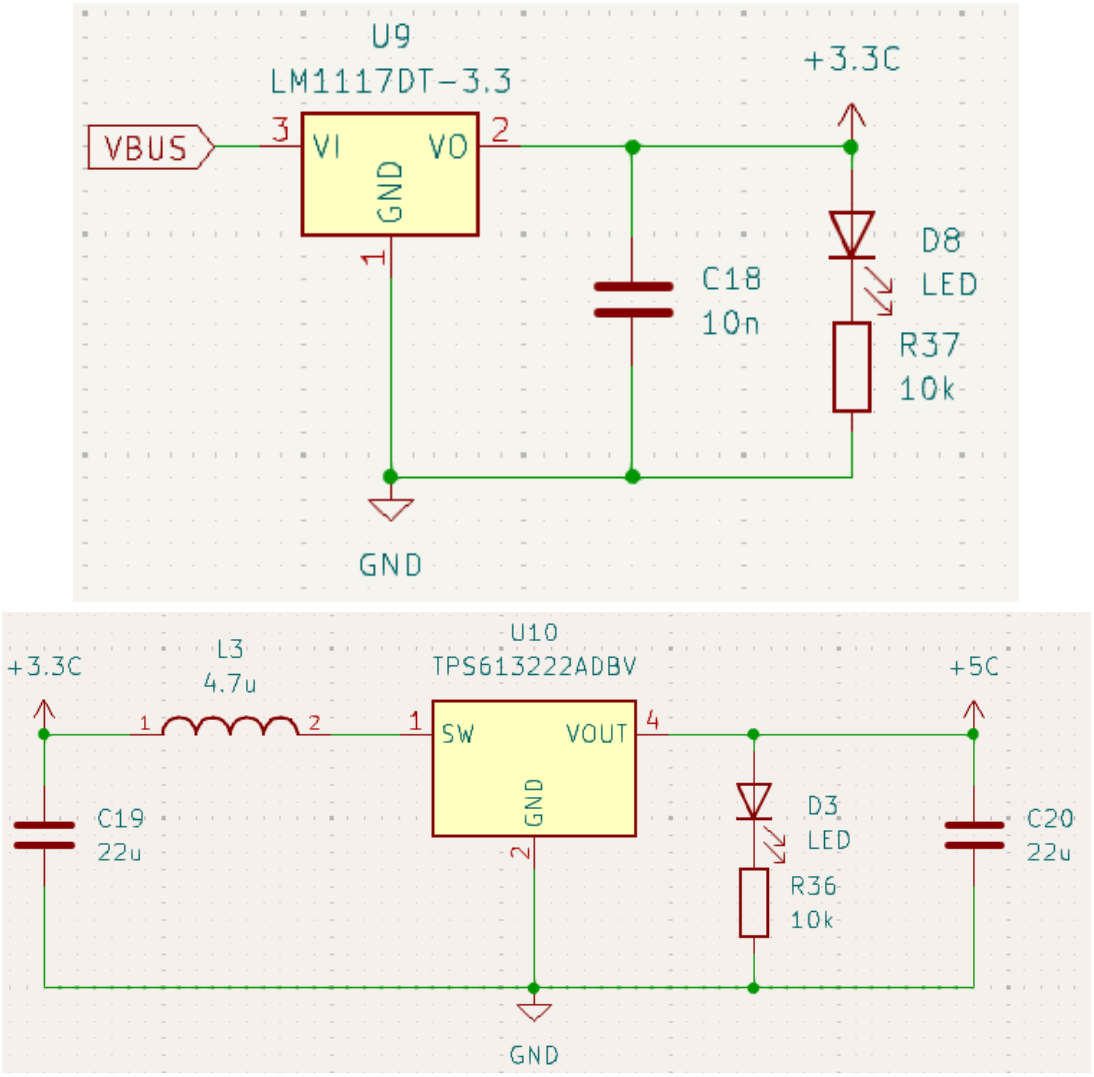


Figure 18: USB Power Supply Connections

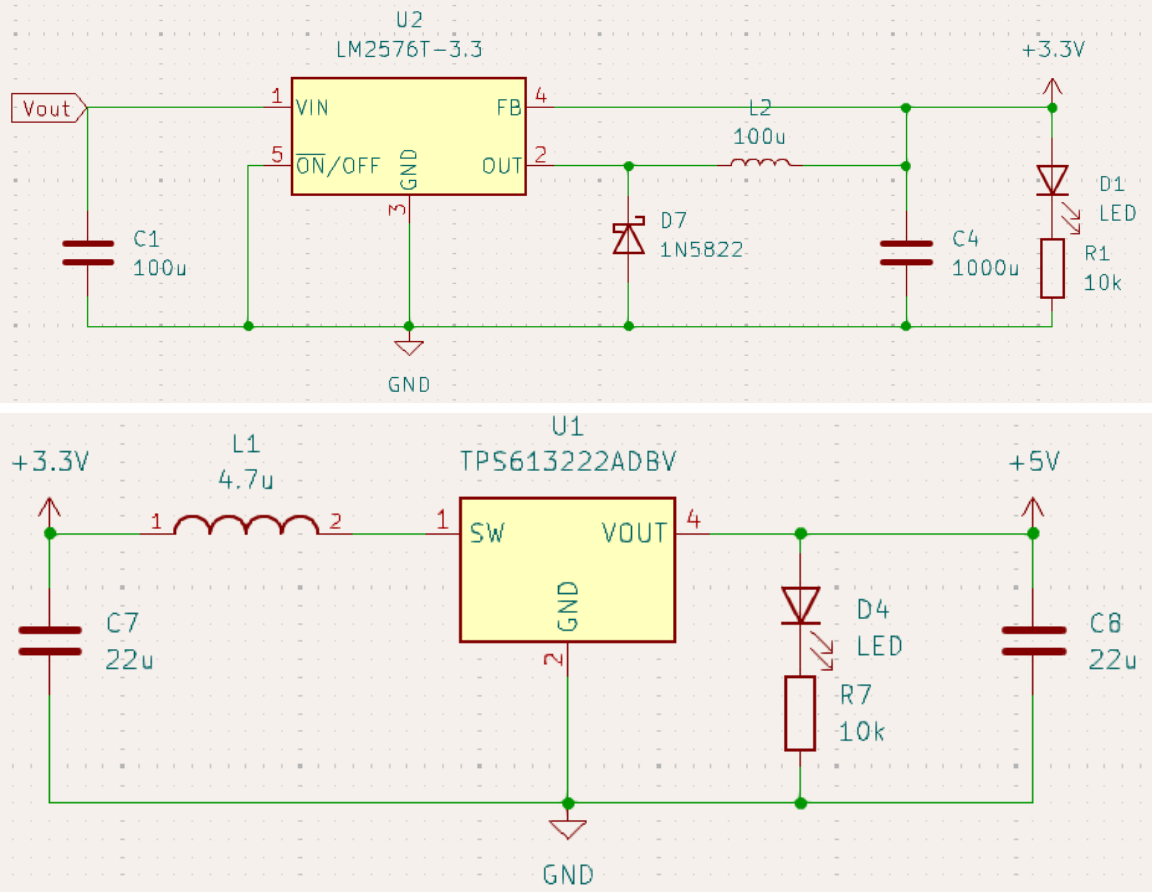


Figure 19: Battery/PV Panel Supply Connections

]

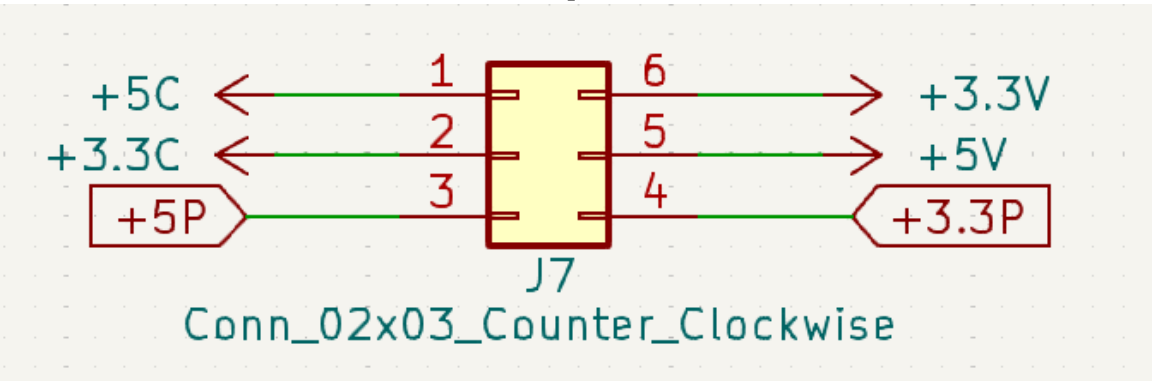


Figure 20: Power Supply Selection Header

The power-switching multiplexer (MUX) on our PCB was designed to select the power source based on the configuration. In our setup, the MUX prioritized the battery as the primary power supply, but if the battery was unable to provide power, it would automatically switch to the PV panel as a backup.

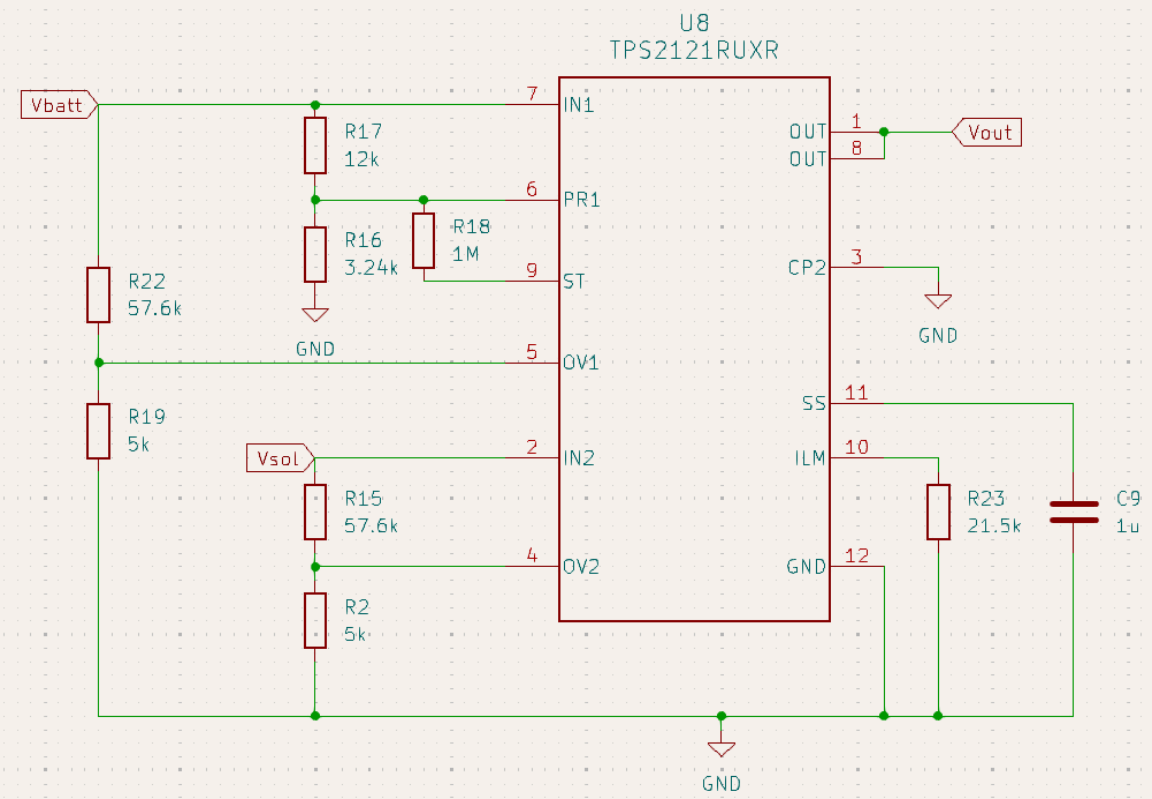


Figure 21: Power Switching MUX

Finally, we utilized many 0 ohm resistors throughout our layout. This was done to ease with any problems that might have risen up during the implementation of our board.

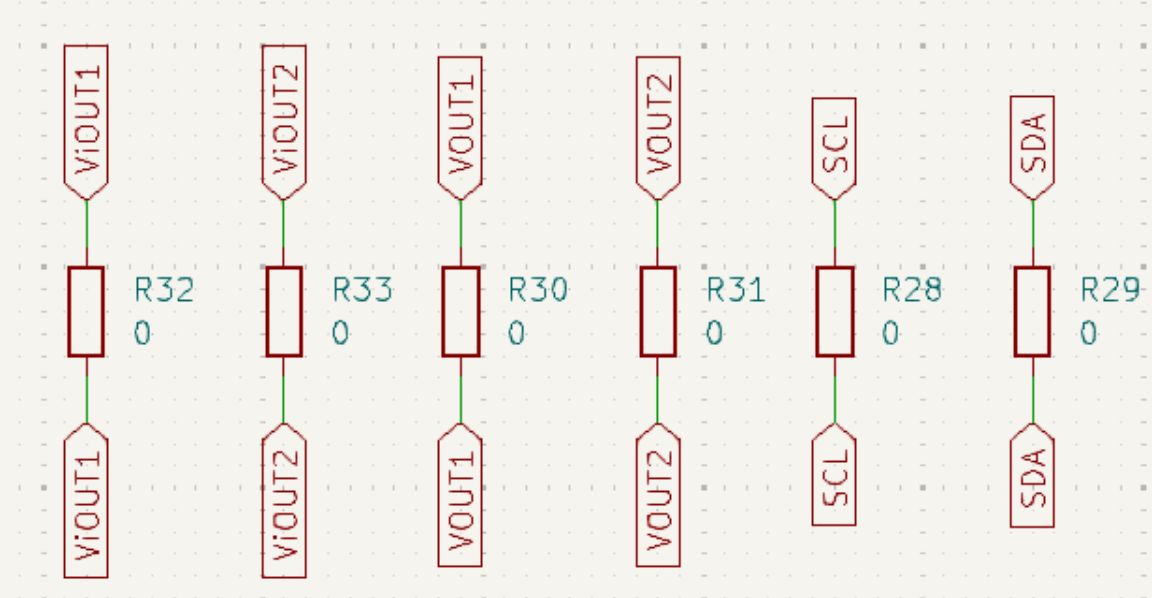


Figure 22: Zero Ohm Resistors

After creating the schematic, we proceeded to determine the layout of our PCB. This meticulous process involved considerable trial and error to optimally place the components. Online tips proved helpful, such as grouping the components of each subsystem together and positioning them close to one another. Additionally, we ensured that all decoupling capacitors were placed near the power supply pins of each IC to guarantee proper functionality. After several attempts, we finalized a layout that worked effectively. We then connected each component with traces, ensuring that power supply traces were wider than standard traces. All applicable traces were designed with a thickness of 30 mils to simplify troubleshooting and ensure ease of maintenance. For the current sensor traces, we used the built-in trace width calculator in KiCad to accommodate the maximum current of 10A. Once all traces were placed, we created a ground fill.

We took care to avoid creating islands, areas on the PCB that are not connected to anything. To address these, we used vias to connect the bottom ground layer with the top ground layer, ensuring proper grounding throughout the board.

Once the board layout was finished, a 3D model of the PCB was generated. This can be found in Chapter 8. The full schematic for the metering PCB can be seen in Figure 16.

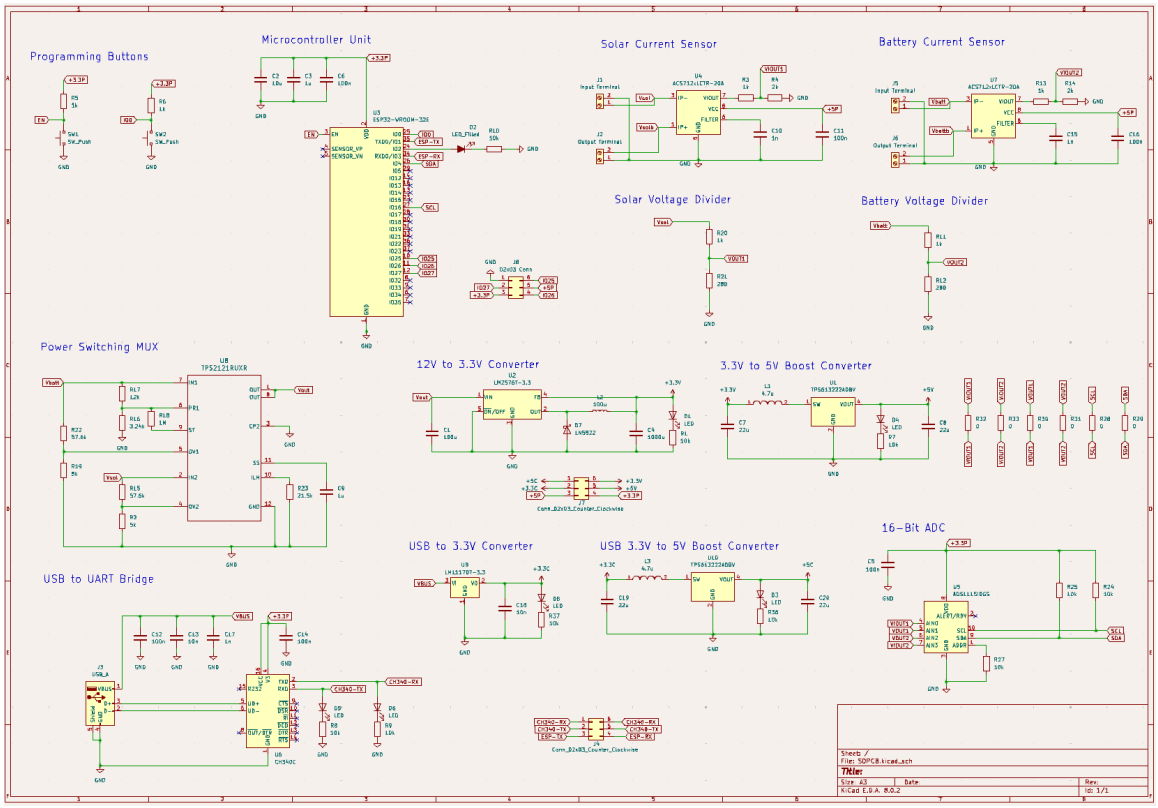


Figure 23: Metering PCB Schematic

6.3 System Architecture

The key aspects of system architecture in hardware design include the subsystems and any components. This includes all physical components such as circuits, MCUs, sensors, actuators, and any communication interfaces. Between these components there should be interconnection with signal paths that carry data and control signals and power distribution paths. Data flow should be shown on the system architecture to display the movement of data through the system. This includes inputs and outputs. From here, communication protocols should be labeled. The standards and protocols that allow for data exchange within the system must be clearly visible.

The diagram detailing the system architecture for the Cyber S.H.I.E.L.D. can be seen in Figure 26. This diagram shows all inputs and outputs for the components as well as communication of data throughout the system.

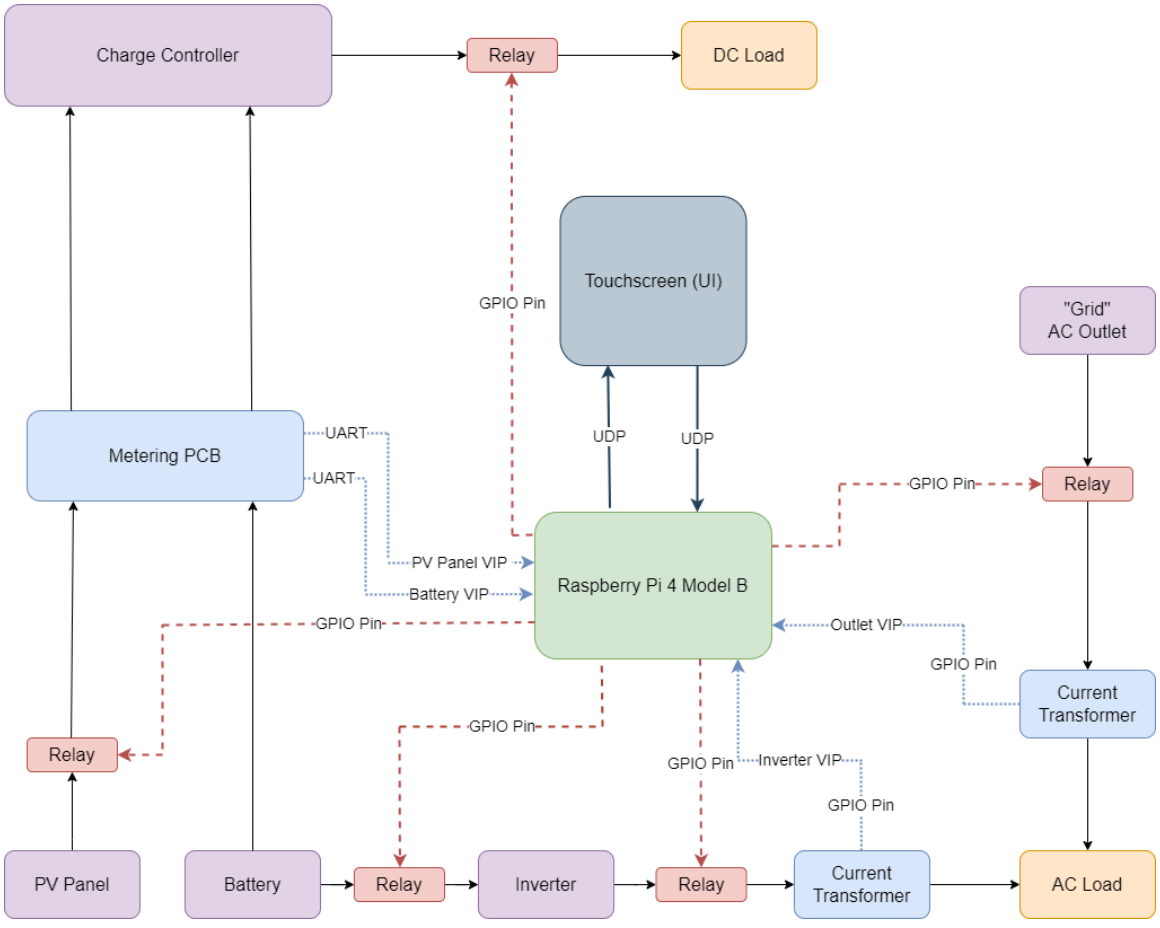


Figure 24: System Architecture

As shown in the system architecture flowchart, the relays are connected with the Raspberry Pi 4 Model B through the GPIO Pin. This allows for the communication of commands to the relays to turn on or off.

Furthermore, the DC metering PCBs are able to communicate using UART to the Raspberry Pi. Data such as the PV panel and battery's voltage, current, and power measurements are communicated. All data is stored within the PCB's ESP32 chip and is communicated to the Raspberry Pi 4 Model B.

In addition to this, the current transformers are able to complete metering of the AC sources. This includes power, voltage, and current measurements for the inverter and the AC outlet. Again, this communication takes place through GPIO Pin connection.

Lastly, the user interface on the touch screen was able to communicate with the Raspberry Pi 4 Model B to send and receive data and commands. These actions would take place by transmitting UDP messages over WiFi.

6.4 Structural Illustration

The structural illustration for the hardware in the project was important in order to begin the process of building the model. We knew that we would need to construct a model home that was large enough to withstand the weight, length and width of the solar panel on the roof. In addition to this, it needed to be portable to allow for ease of moving it to different locations. The 3D model for the model can be seen in the figure below.

Figure 25: Illustration of Model Smart Home

After designing what we wanted the model home to look like, we began to integrate the hardware components that would be needed in the system. This included the solar panel, the charge controller, the battery, the inverter, both AC and DC loads, any relays, AC and DC metering devices, and the AC source connection from a wall outlet. This structural illustration can be seen in the figure shown below.

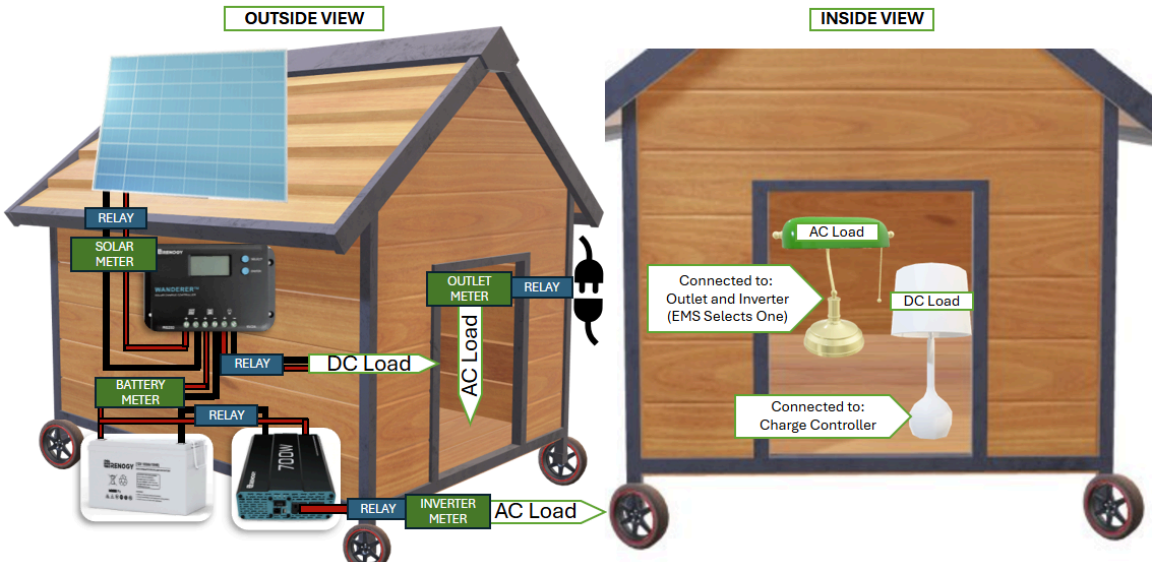


Figure 26: Smart Home Model Structural Illustration

The solar meter and battery meter PCB design was shown at the beginning of this chapter. The inverter and outlet meters were selected to be current transformers that have the ability to communicate with the Raspberry Pi.

7. Software Design

This project involves the creation of two major software components. One is a user application that provides a visual and interactive experience of the project's entirety. The second is the software uploaded to the Raspberry Pi, which controls the various hardware components and provides information back to the user application. These two pieces of software contribute to the success of the project and have their own established goals.

7.1 User Application Design

This project involved the creation of a software element that allows the user to interact and learn from the simulation. The goal of this software are as follows:

1. Create a destination that displays the simulated smart home in real time

2. Allow the user to interact and adjust various parameters in the smart home, including but not limited to: time of day, load type, energy source, etc.
3. Serve as an interactive platform that shares information on the simulator, the various components involved and the impact of adjusting the various parameters.
4. Provides educational information about each of the components in the context of smart homes and the grid.
5. Reacts to cyber attacks and educates user.

These goals will helped guide our team to create an interactive and informative application, fully displaying the capabilities and impact of this project. With these goals in mind, we have designed the software such that it achieves the goals presented and creates an interactive application and impactful learning experience.

The start of the application will display a 3D model of a home. This 3D model will have various locations on the house that the user may click on to zoom into the specific location. These locations include, but are not limited to roof with solar panels, battery behind house, and appliance inside the house.

Once transformed to these locations, the user is shown information on the significance of these components to the smart home and are given an opportunity to learn about the role of this component in the system. The user is also given the option to return back to the home state of the house where they may choose to visit another location.

Another important feature of this house is being able to display its current configuration in real time. The simulated smart home application should be able to display data of the load powering the appliances, the energy source and more. This should reflect the hardware and serve as a visualization of its current state.

With this feature, we are giving the user the ability to interact with the hardware and adjust it in real time. The user will have options such as choosing AC / DC, Day / Night, Battery Power / Grid Power, etc. When changing these various components, the hardware will receive these commands and update in real time. Once switched, the hardware will confirm the success in the switch and the software will adjust to reflect this change, providing updated data on the current state of the system.

Lastly, the application gives the users the ability to monitor the smart home system and be notified when a cyber attack is occuring. Through our partnership with the UCF Siemens labs, we are developing this tool to give them the ability to test their cyber attack detection algorithms. Our application will visualize when a cyber attack occurs and allow the user to remove their home from the grid to protect it.

This application will serve as a way for the user to interact with the hardware and learn about the system and problem this research team is seeking to tackle.

7.1.1 User Application Flowchart

This section consists of flowcharts for the various functionalities of the application. Each flow depicts the response to specific inputs from the user.

In the figure below, we see an application flow chart for turning the application on/off. This shows the process for booting up the application and shutting it down.

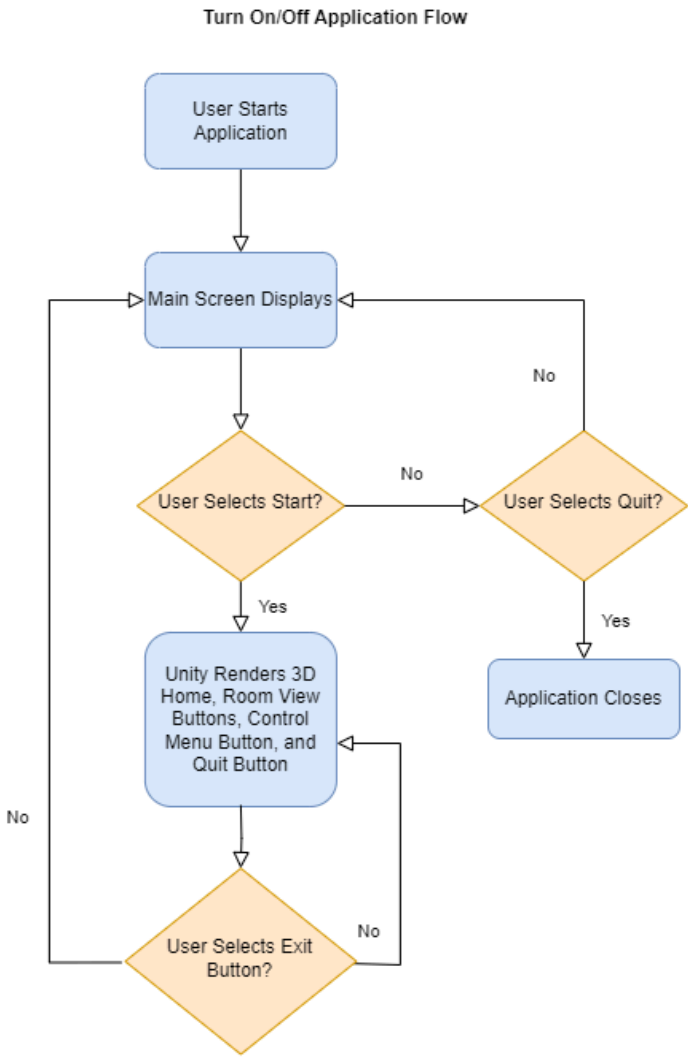


Figure 27: Turn On / Off Application Flowchart

In the figure below, we see a flowchart for selecting different views of the house. When the user selects the specific location, the camera adjusts to the coordinates of that location and we display information about how the system contributes to that part of the house. For example, there will be a view of the solar panel on the roof of the house. When the

user selects that button, the camera moves to it and we see the flow of energy and information about the flow of energy.

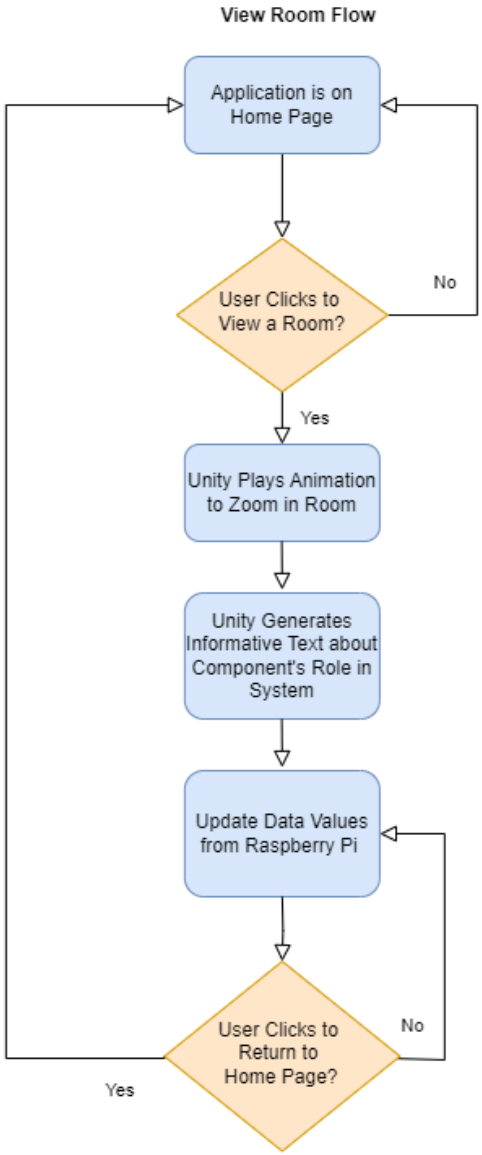


Figure 28: View Room Flowchart

In the figure, we see the flow of the user interacting with the configuration menu, setting the various system parameters. In this system, we are able to control the time of day, load and power source. When the user adjusts the time of day, they can make it either day or night.

In the application the environment adjusts to this configuration and the hardware model turns on / off a sunlight pointed to the solar panel. For the load, the user can choose either an AC or DC load. These options adjust based on the configuration of the other settings, so we will be requesting the settings in order. Lastly, the user can choose their power source. They can either choose to use the PV battery or the grid. This is also related to being able to protect your home from grid cyber attacks by switching your home source to the PV battery charged by the solar panel.

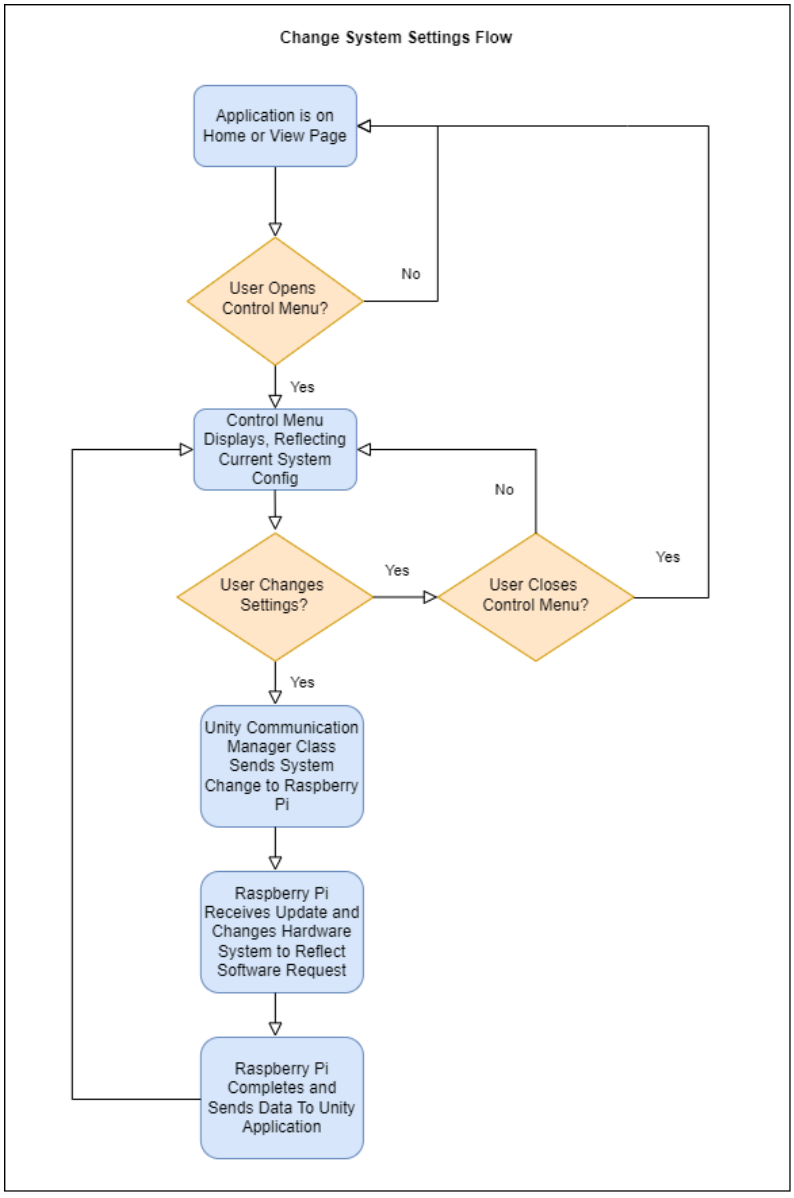


Figure 29: System Settings Flow

The diagram on the following page is important in that it shows off the various possibilities for configuring the system to the user's selections. One important thing to note about the system is that some parameters impact other parameters, so we will need the user to select the parameters in order. Therefore, in the design of the user application, we present each parameter individually so we can display the correct options.

Looking at this diagram, we see that the user starts by powering on the application, opens the model and begins configuring the model. Starting with the time of day, the user can choose to set the time of day to day or night. This impacts whether the PV will be charging the battery from the solar power.

The next layer is the load that the user is selecting. They are given the option to select DC, AC or DC / AC. With these options, the user can choose what type of load they want in their system.

The final layer is selecting the power source. Depending on the time of day and load selection, the options differ slightly. When the time of day is daytime, we are able to choose to utilize the PV panel for powering the system. Otherwise, we are not able to do that and would either opt for the grid or battery.

Looking at the powersource, if we choose a DC load, then we choose the battery as our load. If AC is chosen, we have the choice to utilize the battery / inverter or the AC connection to the grid. Lastly, if we choose AC / DC then we can choose between either of the previously mentioned options with AC or DC.

Ultimately, the order of how the parameters are presented is important because the power source options are dependent on the time of day and the load. If they are chosen out of order, we may not be able to correctly capture the target conditions.

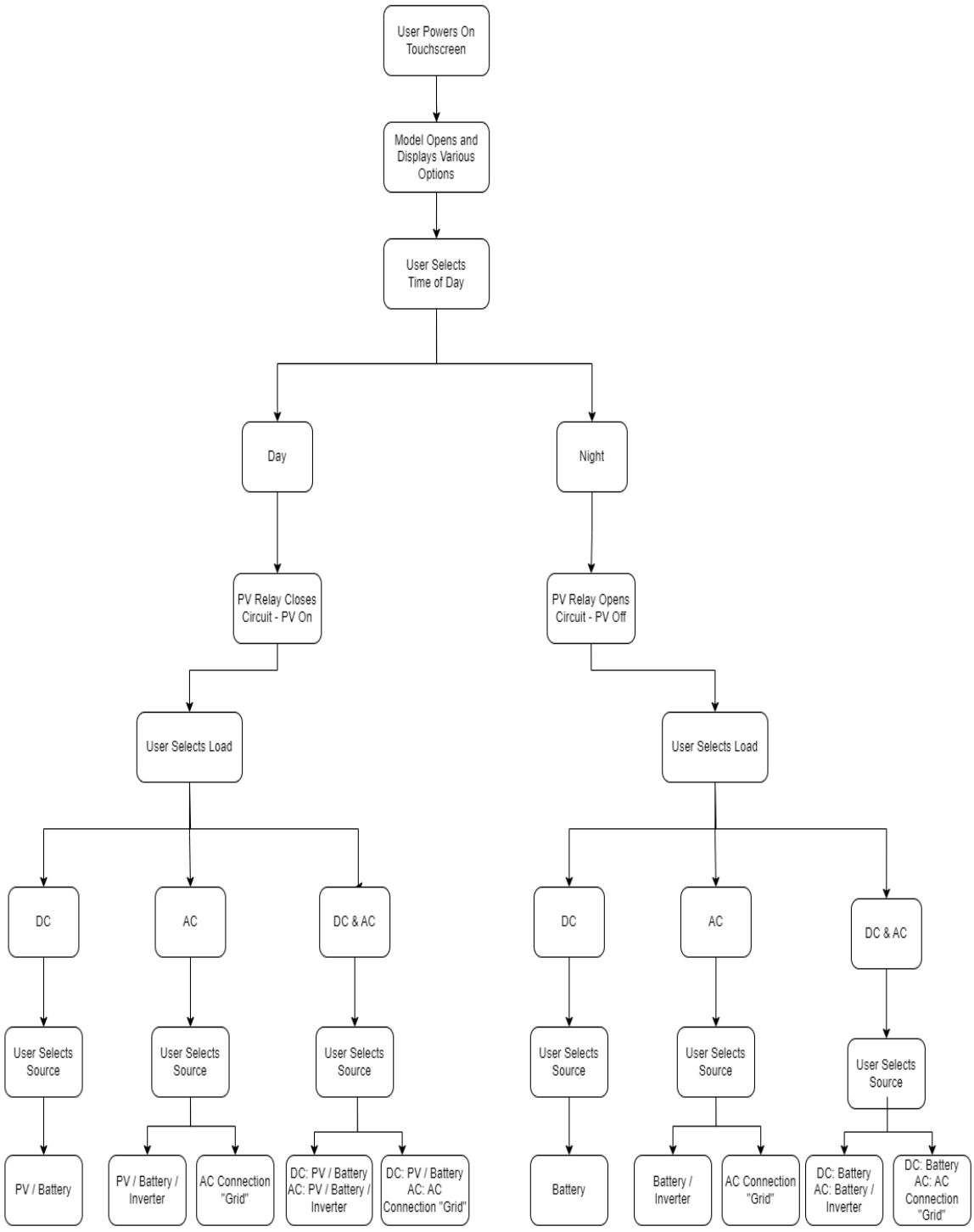


Figure 30: Touch Screen & User Interface

Looking at the next diagram, we see that we can adjust the priority based on the chosen load and the various parameters. Figure 33.2 shows off several different situations we can capture to manage the system's energy.

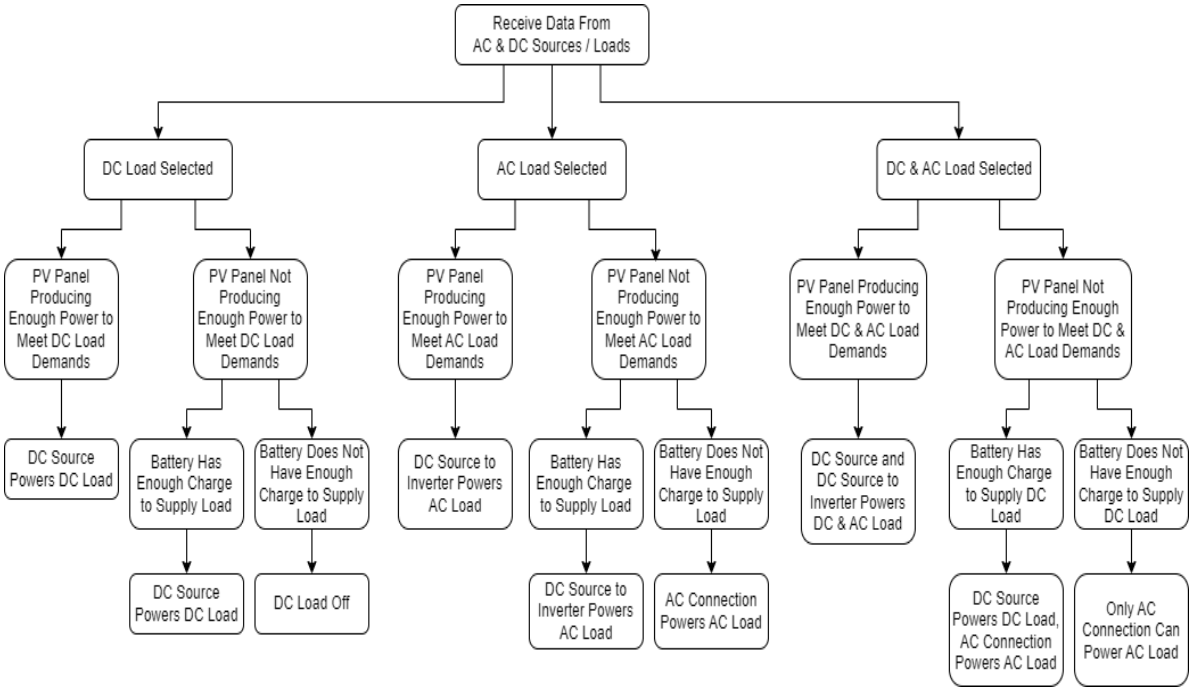


Figure 31: Energy Management System

7.1.2 Integration Flowchart

After the construction of the software, it is vital to correctly integrate the system with the hardware systems. For example, we are interacting with the raspberry pi to handle cyber attack information. With this, we are able to notify the user when an attack is being deployed and give them the option to run the detection algorithm. When the detection algorithm runs, the Raspberry Pi communicates the status of the detection.

This integration piece is important to ensure that our system is working correctly with the other systems. By identifying what each component sends / receives, we can prepare each component for integration.

The diagram on the next page demonstrates the flow of communication between the application on the touch screen and the microprocessor. Starting with the Raspberry Pi 4 Model B, we transmit the data collected from the Smart Home System Metrics as a JSON into the Touch Screen's to be able to be interacted to the user. Afterwards, we give the user the ability to manually select system requirements through the touch screen onto the

Raspberry Pi 4. After the software requests, the Raspberry Pi will change the state of the relays to reflect the changes requested. Then the information changes will occur again and repeat the system.

In addition to this, the flowcharts for the user interface, energy management system, and cyber attack response are shown on the pages that follow.

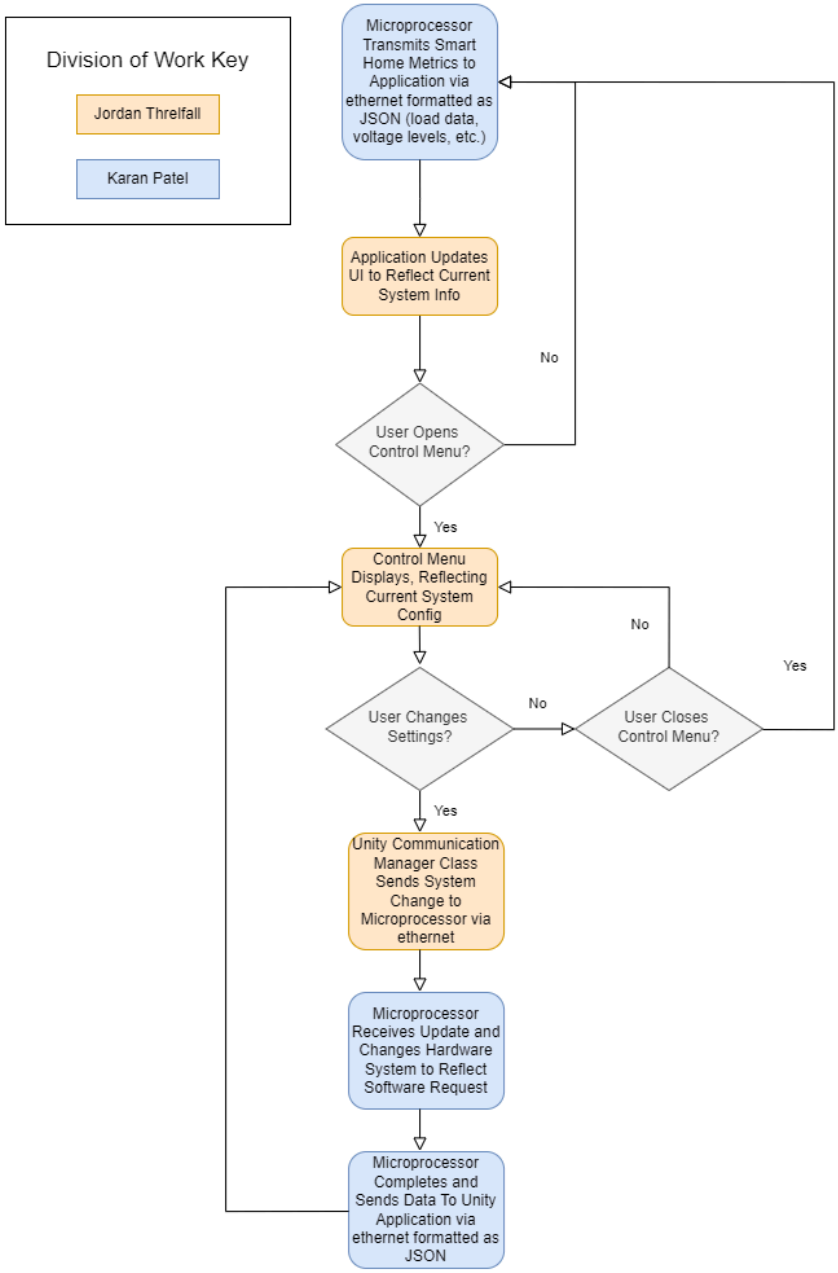


Figure 32: Flow of Communication Between Application and Microprocessor

As indicated earlier, we integrated the raspberry pi to handle the cyber attacks and test various cyber attack detection algorithms. Below is a flow chart indicating the flow of staging a cyber attack, sending the information to the raspberry pi, transmitting that information to the application and then handling the integration of the detection algorithm.

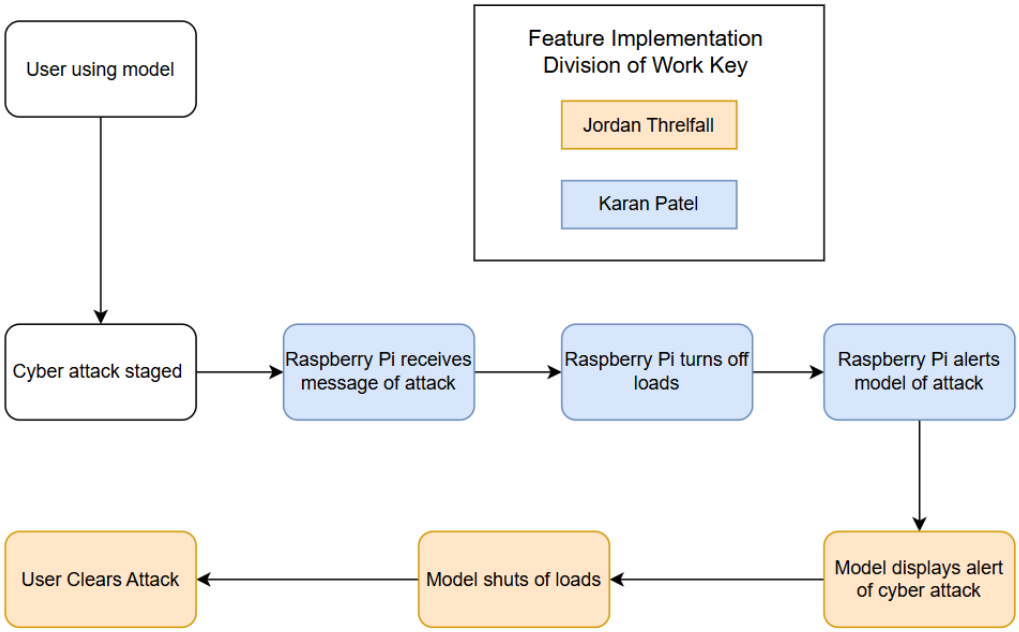


Figure 33: Cyber Attack Response

7.1.3 UI Implementation

This section shows the final implementation for the UI application, displaying the various features and settings that are available to the user.

In the first image below, the UI Application's home page is displayed. This page is showing the various features available to the user and is configured to be daytime. To begin, the left shows several options. The first option is to enable/disable the energy management system. The EMS allows the user to choose the time of day and loads and then automatically decides which power source is most efficient. If we have a fully charged battery, the EMS would choose to power the house with that instead of paying for power from the grid.

The second option is to “Configure House” which allows the user to set parameters for the system and determine how the system should run. When the parameters are set, the entire system adjusts accordingly, matching the user’s inputs.

The magnifying glasses at the bottom allow the user to zoom in on different components of the house. By selecting one of these, the user is able to view the various components of the smart home. When selected, the camera moves to that component, sharing some information about the system and showing real-time data of the components. For example, if the user chose “Solar” the camera would zoom in on the solar panel and display information about that piece and the flow of energy from the panel (if it is daytime).

The last button is the power off button, which shuts off all loads to the house.

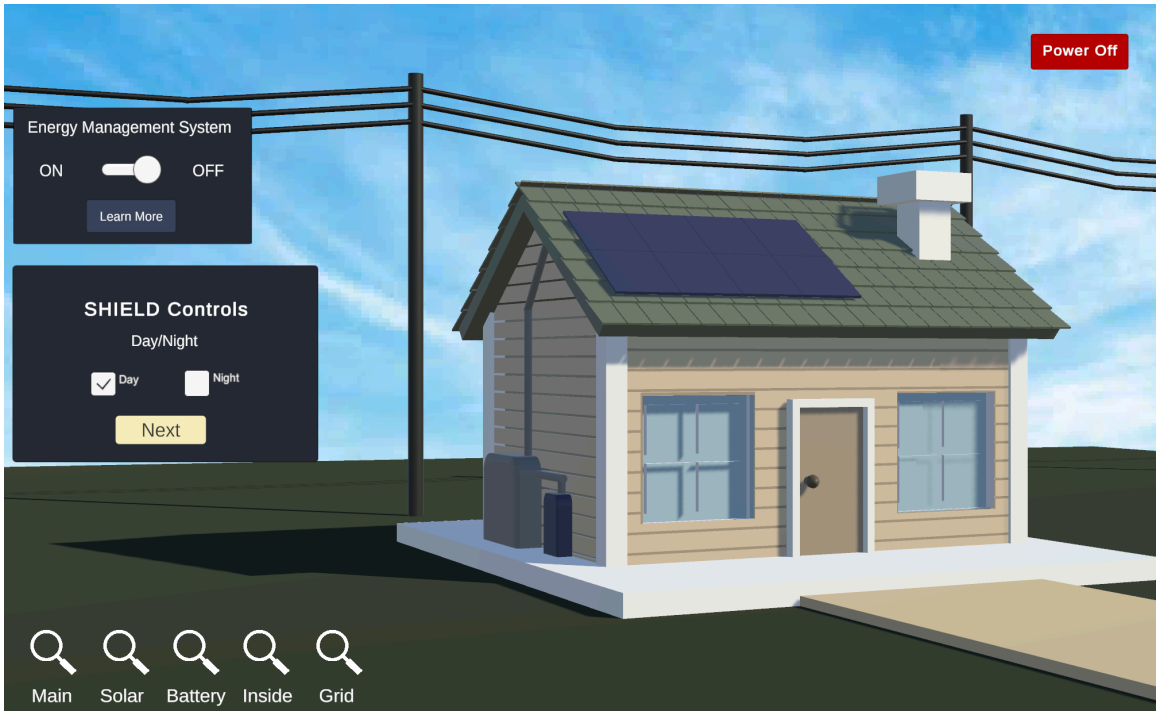


Figure 34: Application UI (Daytime)

For configuring the house, the adjustable parameters and their options include:

- **Time of Day:** Day or Night
- **Load:** AC, DC, or AC / DC
- **Power Source:** Battery, Inverter, or Grid

With this, the user is able to customize how their smart home is being powered, and adjust to specific conditions.

After they select the various configurations, the user can select save and send that over to the raspberry pi. When the Raspberry Pi receives that information, it adjusts the system accordingly by triggering the various relays and controls components.

This UI below is the same as the one above, except it is configured for nighttime. Using the “Configure House” button, the user can change the simulation’s time of day. When daytime is selected, the application adjusts its background to look like it is day. On the hardware side, when the user configures it to be daytime, a lamp powers on, charging the PV Panel. If switched to nighttime mode, the application responds by setting a darker background. The hardware responds by turning off the lamp, no longer charging the battery via the PV panel.

Figure 35: Application UI (Nighttime)

Another component of this is being able to view the different components of the house. Below shows each of the different component views, the metering information available with each and the education panels. Note that the configuration for the home is: Day, AC/DC and Grid. Additionally, the image captures are not connected to the Raspberry Pi nor the hardware system, so there is no metering information present.

Figure 36: UI App Solar View

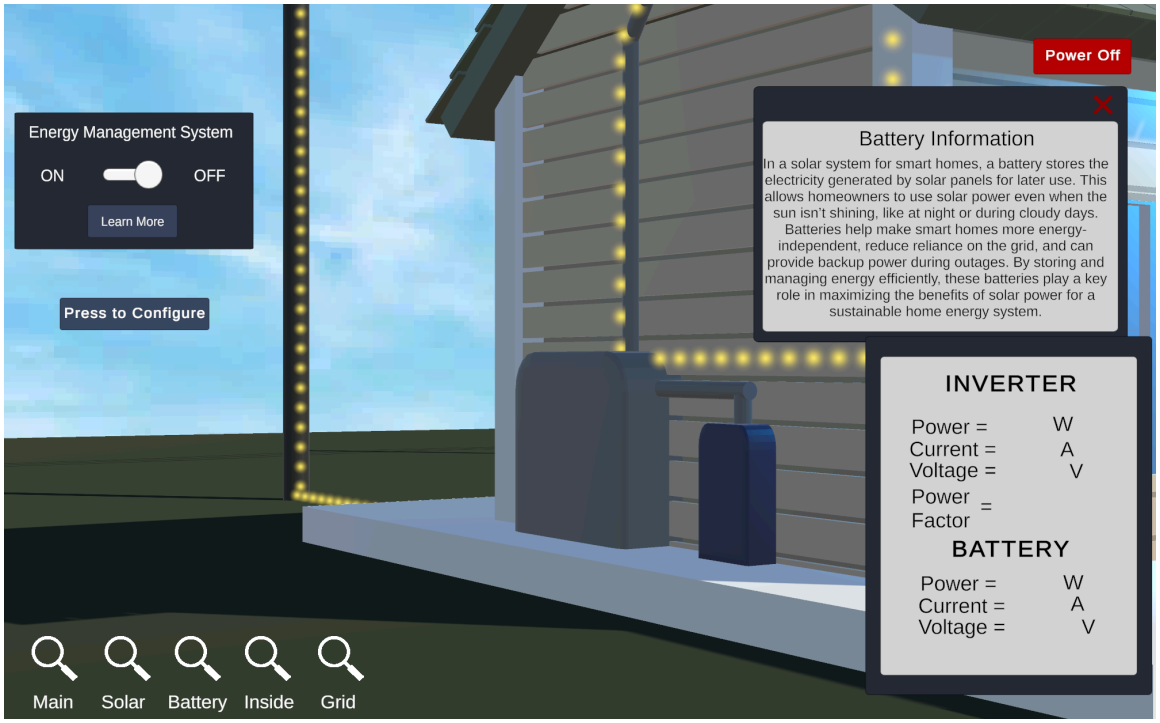


Figure 37: UI App Battery View

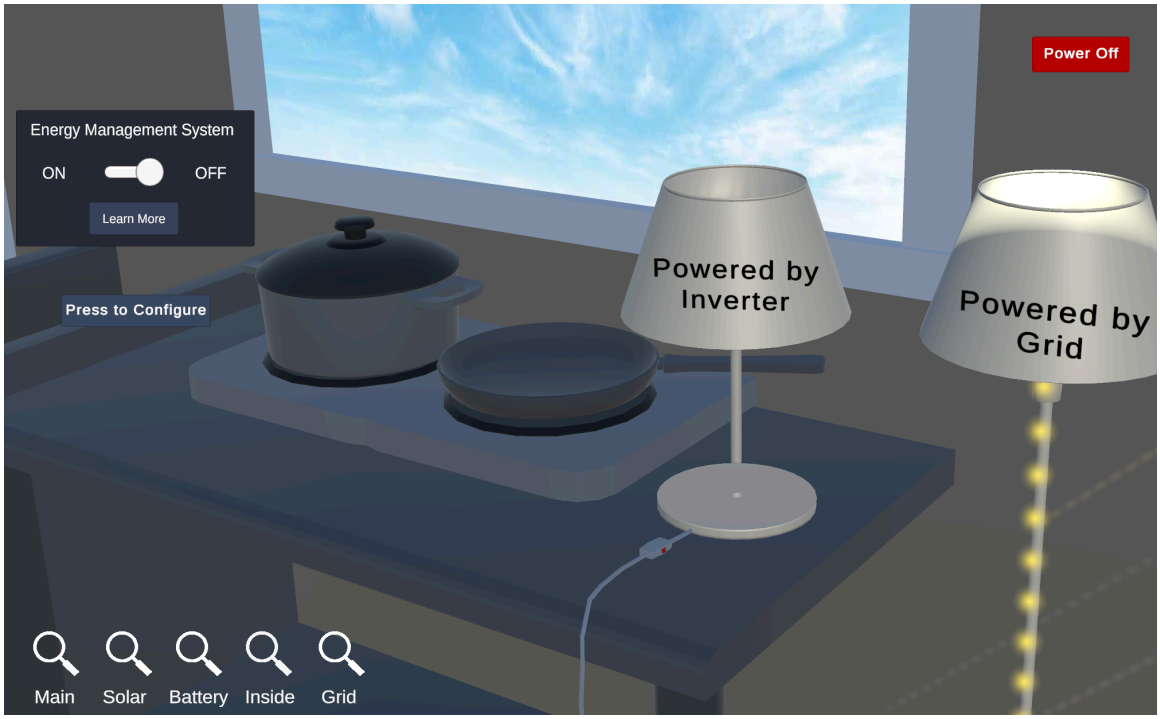


Figure 38: UI App Inside View

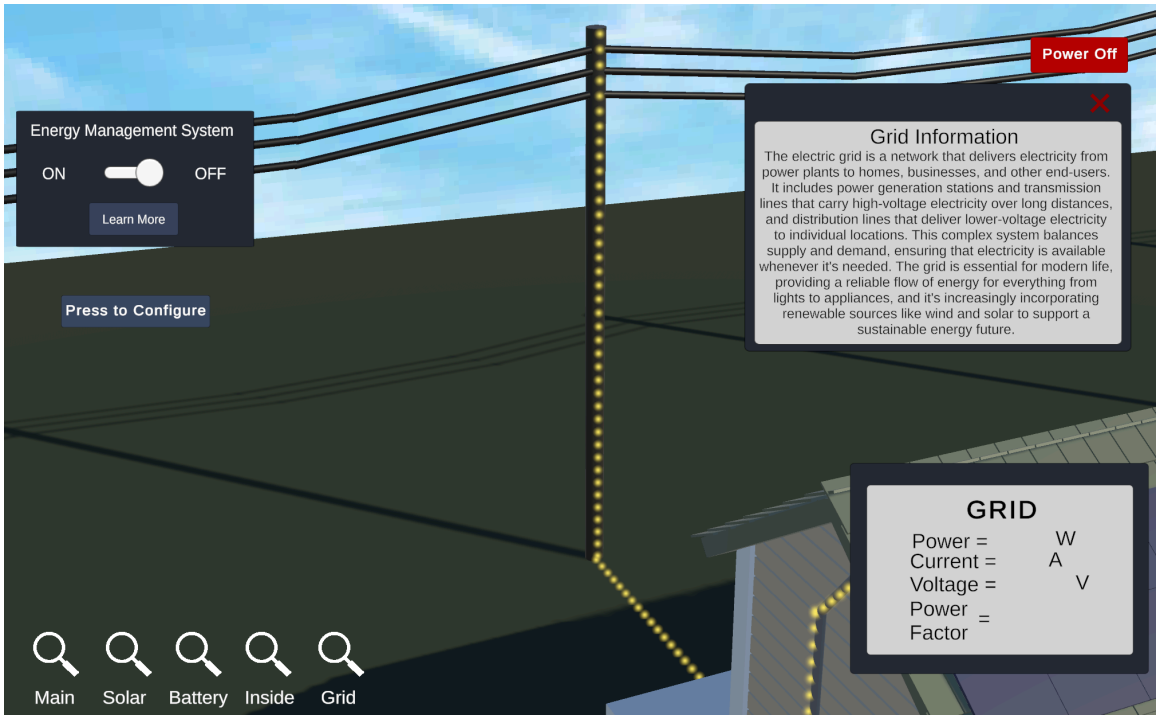


Figure 39: UI App Grid View

All of these views allow the user to interact with the configured smart home, view metering information and learn about the various components.

The last element of the home is the cyber attack handling system. Upon the detection of a cyber attack, the Raspberry Pi notifies the UI Application and the UI Triggers a cyber attack event. Below shows the occurrence of a cyber attack.

Figure 40: UI App Cyber Attack

When this occurs, all loads to the house are shut off. The user is given the option to learn more about different types of cyber attacks or “continue” to normal operation.

7.2 Design Components

The implementation of this design involved planning of the various components being prepared, identification of communication and the schedule for developing the components.

7.2.1 System Components

The following table shows the various software components required for the system and provides a description of how they were implemented

.

Table 42: Software Components

Component	Description
Unity Application Project	The Unity Application Project needs to be set-up to create the development environment for the application. Unity offers several tools to build the application utilizing its UI. With an environment prepared, we can build the application.
3D Model Smart Home	The 3D Model of the Smart Home will be integrated in the UI application to provide the user a visual of the smart home and the ability to interact with it. For its creation, we will be using Blender, which is the selected 3D modeling tool as identified in Chapter 3. After the house has been created, we will be able to export it and import into the unity application.
3D Components	Similar to the smart home, we will be using Blender to create the various 3D component models for the home. This includes the solar panel, battery, lamps, etc.
Application Environment (lighting, objects, UI)	The Unity Application Environment will need to be set-up to prepare the application for use. This includes the implementation of lighting, objects, animations, and UI. Lighting will enhance the environment and allow us to respond to day / night selections. We will also need to import the various 3D models. The UI is an important piece that will allow the user to interact with the application.
Application Camera Views Script	The camera views script will handle the selection of the various camera views and move the camera accordingly. If the user wants a view of the solar panel component, they will select that view and then the camera will move to that specific (x,y,z) location.
Configuration Handler Script	The configuration handler script will handle the various configuration settings available for selection for the user. Additionally, after the settings are determined, the information will be sent to the networking script to prepare and send to the raspberry pi.
Networking Script	The networking script will be used to interface with the raspberry pi. Their communication involves exchanging information on system configuration, cyber attacks, and component information.
Cyber Attack Detection	The cyber attack detection handler will respond to staged

Handler	cyber attacks, allow the user to deploy detection algorithms and view the status of the detections.
UI Handler	The UI handler will be responsible for displaying the UI and its buttons and correctly handling button pushes. This includes the configuration button and the various camera views.
System Information Handler	The system information handler will receive information from the raspberry pi on the different components and display that information on the components of the cyber home. Additionally, when users pull up a specific scene both the component information and the educational information will be available.

All of these components integrated together to create the UI application for our model smart home. This application integrates with the hardware via the Networking Script. As stated in previous sections, the Networking script prepares to send system configuration and cyber detection information for the raspberry pi. On the receiving side, the script handles the component information and cyber attack staging info / status. Each component plays a vital role in the system's operation.

7.2.2 Integration

The final piece to the system was the integration. This part mainly involves two major points: networking integration and UI application integration.

The networking part is ensuring that the raspberry pi and UI application are integrating and sending the correct messages to each other. To ensure that both systems are aligned, an ICD will be created to ensure that both systems know what messages to send / expect.

As specified in Chapter 9, once the integration is complete, there will be thorough testing of the systems, ensuring that the messages are being correctly sent between the two.

The UI application integration involved ensuring that each component of the application interacts with its respective components correctly. For example, we want to ensure that when we select the PV panel camera view button, we want the camera to transition over to the correct position and display the specified view. As Chapter 9 states, there will be thorough testing of this integration in ensuring that all the systems perform correctly.

Ultimately, the fabrication of the UI application involved preparing each of the components and then ensuring that they integrate correctly with one another. Once we enter the testing phase of this project, we will be able to determine if the components are performing correctly and then adjusting them if needed.

8. Chapter 8 System Fabrication and Prototype Construction

8.1 PCB Layout

While reviewing the objectives of this project, we decided that we would like a PCB that was able to accurately meter the solar panel and the 12V battery. This PCB did not have any size requirements, but should not be unnecessarily large. In the following paragraphs, we will go into detail to show the process of constructing the PCB prototype.

To begin, we began by designing the PCB for the metering of the solar panel. The information collected at the solar panel PCB would be communicated to the Raspberry Pi and the Raspberry Pi would communicate the data to the touch screen and the OPAL-RT simulator. In order to design the physical arrangement of the electronic components and their connections on the PCB, we completed several steps to make sure that the board would work correctly. First, we translated the circuit diagram shown in Chapter 6 into a schematic. This schematic showed the components we would need and their connections. We arranged the CPU, ADC, current sensor, resistors, and capacitors on the board so that it would perform well and be an efficient size. From here, we created the traces that connected all of the components together. This process involved finding the ideal path to minimize noise and interference. Following this step, we made sure to define two layers of the PCB. Before continuing on to the next step, we made sure that the design adhered to all constraints and standards. We verified the trace width, the hole sizes, and the spacing. By adding heatsinks, we were able to account for ways to manage heat production within the PCB. From here, we were able to create a file that the PCB manufacturer would use to create the board. This file was called the Gerber File. In Figure 41 the PCB layout is shown. In Figure 42, the 3D Model of the PCB is shown.

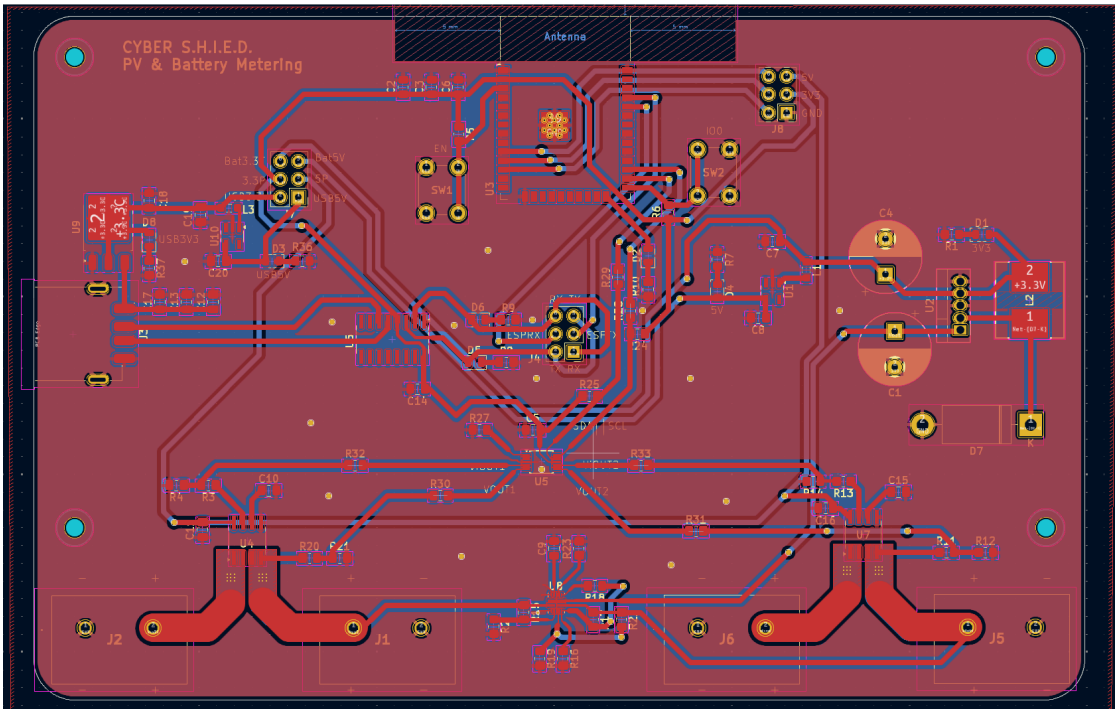


Figure 41: Metering PCB Layout

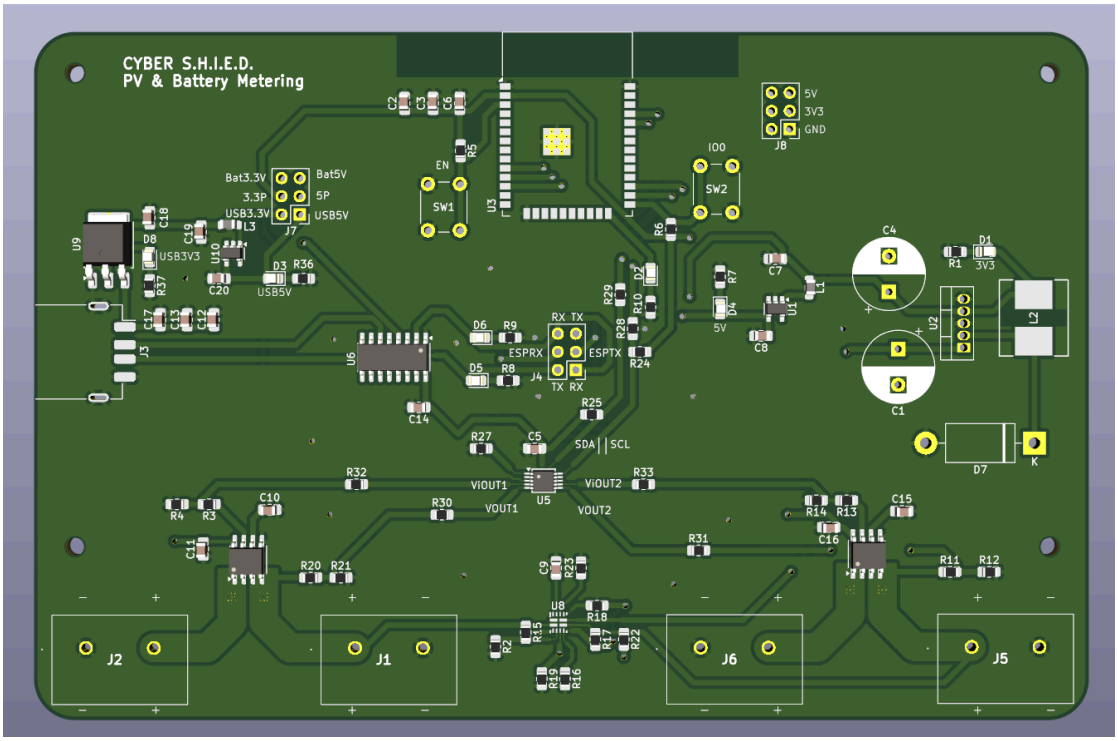


Figure 42: 3D Model of Metering PBC

Following the creation of the initial solar panel metering PCB, we revised our design to combine the functionalities of solar panel metering and battery metering into a single PCB. This combined metering PCB needed to measure both the solar panel output and the battery parameters, including the voltage of the battery and the current being discharged. The MCU would then collect these measurements to calculate critical values such as the battery's state of charge. This data would be communicated to the Raspberry Pi, which would then relay it to the touchscreen and the OPAL-RT.

To create the combined metering PCB, we designed the board according to the schematic outlined in Chapter 6. The schematic integrated the functionalities of both the solar panel and battery metering, as both systems operated at the same voltage and required similar measurement tasks. We carefully arranged the MCU (we selected the ESP32), the ADC, current sensor, capacitors, and resistors on the board to ensure proper performance. Afterward, we traced all the components to optimize the layout and minimize interference.

We thoroughly reviewed the board to ensure it adhered to all design constraints and standards. This involved double-checking trace widths, hole sizes, and spacing. As with the original design, we included heatsinks to manage heat production within the board. Once the layout was finalized, we created a Gerber file for the PCB manufacturer. Figure 43 shows the updated PCB layout, and Figure 44 illustrates a 3D prototype of the combined metering PCB.

After finalizing the PCB layout, we reviewed it to confirm it could be ordered in time for Senior Design 2. Once the board arrived, we planned to solder all components and create a PCB prototype. This prototype would allow us to verify the design's functionality and identify any issues before the final production. Ultimately, this revised design resulted in a single metering PCB that met all electrical requirements, fit within the physical constraints, and could be manufactured cost-effectively.

8.2 Smart Home Prototype Construction

The model smart home was meant to be a physical representation of the functioning of a smart home for educational purposes. By creating a model that resembles that of an actual home, viewers will be able to connect more with the experience. Construction of the model smart home began at the end of Senior Design 1. To begin, we collected the materials that we would need to build the prototype. We wanted to make sure that the model could be easily transported and stored prior to its official demo day at the end of Senior Design 1. The model smart home would be constructed out of wood panels. Wood was determined to be a sturdy enough material to hold the solar panels and any additional devices (such as the inverter or charge controller). One side of the roof has the ability to be lifted in order to see the loads inside. The model has wheels in order to be moved.

Following the creation of the basic smart home model, we included additional devices and components that we would be utilizing in the project. This included the solar panel on one side of the roof, the charge controller, the battery, the inverter, the AC power connection, all metering devices, the DC load, and the AC load. The figure below details the solar panel, charge controller, battery, AC Connection, and inverter connections within the prototype.

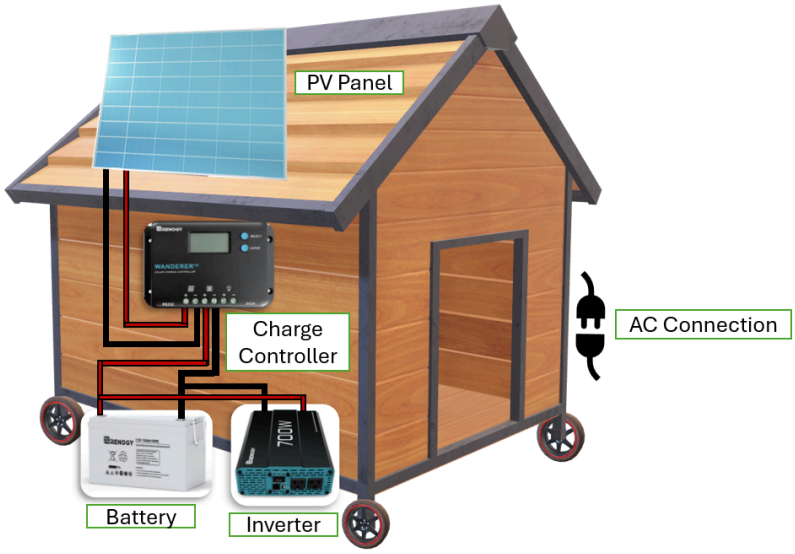


Figure 43: Prototype With Solar, Charge Controller, Battery, AC Connection, Inverter

We then created a prototype to detail where the DC and AC loads would be connected. Arrows leading from the inverter and charge controller show connections to loads.

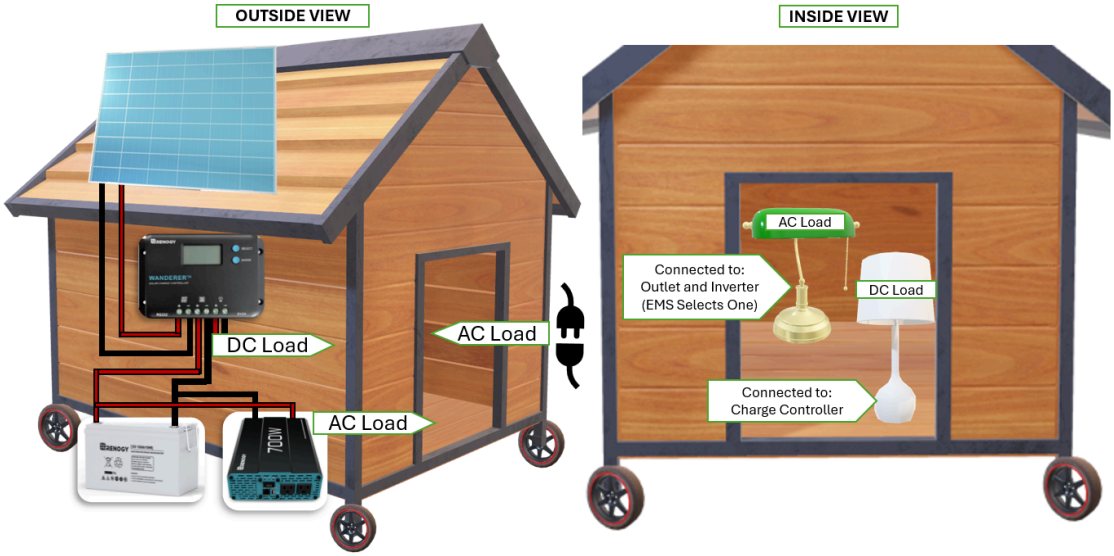


Figure 44: Inside and Outside View of Prototype with Loads Connected

As shown in Section 2.8, metering devices for the model would be placed in various locations along with relays to allow for the proper selection of an energy source and load. The placement of these relays and metering devices for the prototype can be seen below.

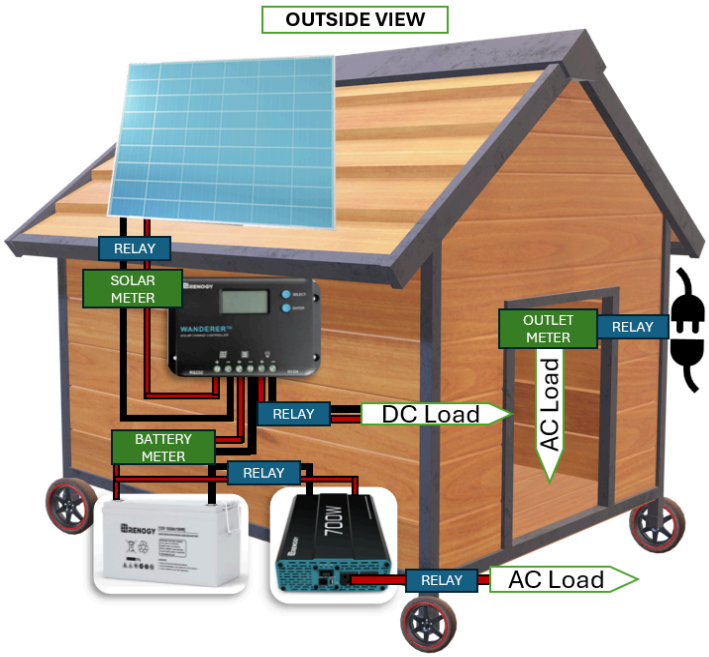


Figure 45: Prototype Including Relays and Metering Devices

8.3 User Interface Fabrication

The UI Application involves the creation of a 3D model of the home and the development of the application and its various features. With this, Chapter 7 identifies the various components involved in the creation of the application. Each component is vital in ensuring that the entire system works as we expect.

8.3.1 System Components Fabrication Plan

In this section, we take the components presented in the previous section and identify the order of execution, to ensure we are prepared for testing, as discussed in Chapter 9. For development, we are tracking progress utilizing Jira. We have created tasks for hardware and software development and divided them into sprints.

8.3.1.1 Unity Application Fabrication

For this project, Unity is the base for the entire application. As discussed in Chapter 3, we evaluated several options for how to develop our UI application and the specific software

to run it. Ultimately, we determined Unity to be the best option due to its 3D capabilities, easy customizability and networking capabilities.

To begin, we will create the Unity project and develop the UI and buttons. After the creation of those UI components, we will create the scripts for handling the various commands. To start, we will create the Configuration Handler Script to handle the various configurations and prepare to send them to the Raspberry Pi. Additionally, we will create the script to handle the various camera views of the model smart home.

The next piece of development involves creating the networking script to integrate with the raspberry pi. This is a vital component to integrate the UI application system with the hardware. Linking these systems gives us control over the hardware via the application and the ability to view the current status of the model smart home.

Finally, we have two systems we want to prepare: cyber attack handler and system status info handler. We want to implement the cyber attack handler to display the correct screens when a cyber attack is staged and the algorithms are deployed. The system status handler will be used to take the system information from the various components and display them on the correct camera views and model components.

With all of this, we want to ensure that all the 3D model components are in place and that everything is presented professionally. After implementation, there will be a fine-tuning phase to ensure that each system is working properly and displaying correctly.

8.3.3.2 Blender Component Fabrication

The Blender phase of the software systems include the development of the 3D model components for the Unity application. For the application, we designed it to be a 3D model where the camera pans and can zoom into specific components, such as the PV panel or battery.

In development of these 3D models, we created the exterior of the house with the PV panel feeding into a battery. We then created the interior showing some of the home's components, such as the two lights that are shown inside. With this in mind, we held an important design in our mind for the house and then built upon that design.

Afterwards, the components will are exported as .obj files that are then imported into Unity. We then integrate all of the scripts and components to ensure that the house and its components are accurately displayed.

9. System Testing

Following the implementation of the system, it is vital to test the various features and verify that everything is working as intended. There are several components of the overall

system, so we investigated each and verified that it was performing / outputting as expected. Looking at the overall system, we first tested both the hardware and software aspects. After that testing, we tested their integration and verified that they were communicating with one another correctly.

After that discussion, we are going to look at our plan for Senior Design 2 in terms of testing the system and adjusting if components are not functioning as expected.

9.1 Hardware Testing

For the Senior Design project, we were involved in maintaining constant control over multiple systems to demonstrate system-level controls and provide a proper understanding of simulating a smart home system. This included components that involved experience in data collection and interpretation of set data. We went about testing our hardware by providing multiple use cases that were demonstrated in our demo. In designing the Smart Home System, we came up with a varied process of showcasing our system as a whole. This comprehensively showed off the system's functionality as demonstrated in the demo.

As we began, we explained the importance of the main control system that interprets the user's information given through the Touch Screen. The touch screen sends information via JSON text to our Raspberry Pi 4 Model B, which uses a script written in Python to open and close different relays that control various power sources for the Smart Home System. When each relay is activated, it displays an LED light and produces a sound as the magnetic field flips the switch from its original position—Normally OFF—to its new position—Normally ON.

This was the main objective of the SBC we chose for our project. With the initial setup of how our project was shown in the demo, we provided a physical demonstration of the system. To ensure the Smart Home System functioned as expected, we designed a testing system involving multiple benchmarks to confirm smooth operation without issues related to application or verification. The Raspberry Pi 4 Model B and the signals from the relays were run through a series of tests to ensure the project's success.

Regarding the Raspberry Pi 4, we ensured the data received from the Touch Screen, inputted by the user, functioned as expected. As mentioned, a script was implemented to process the user's inputs. With the Raspberry Pi acting as the coordinator, it opened/closed specific relays connected to the PV Panel, the battery, and the AC Source. We verified that currents flowed through the LED located on the relays themselves. From there, we observed the values obtained from the specific sources being opened and closed, with the script projecting these values back to the Touch Screen.

Furthermore, with the OPAL-RT, we ensured successful communication between it and the Raspberry Pi 4, which was one of the main stretch goals we completed. We used an Ethernet connection line to transmit data between the Raspberry Pi 4 and the OPAL-RT

system. The system was examined using MobaXterm, utilizing an SSH session for communication. From there, we expanded our Python script to send data from the Raspberry Pi 4 via Ethernet to establish a connection with the simulator. Once completed, communication lines were established between the two systems, allowing us to expand the simulation system for the OPAL-RT and the Digital Grid Lab.

Our goal was to demonstrate the physical state of the smart home system through the integration of software and hardware. Key educational factors included controlling loads, changing the time of day, and showing changes occurring in real time. We emphasized minimizing delays to make the system as close to real time as possible. These changes occurred automatically as we simulated various scenarios, such as thunderstorms, clouds, and power outages. For our purposes, we demonstrated scenarios like cyberattacks, showcasing how the OPAL-RT framework could test these situations for research purposes.

9.2 Software Testing

Our Senior Design project involves creating a simulated smart home with real hardware components and visualizing the home through an educational application. This application allows the user to perform several actions: adjust various parameters of the smart home (load, time of day, etc.), view different rooms/components of the smart home, learn about the system, and view the status of its components. With this list of features, it is important to verify that the software performs as expected.

To begin, we verify that the interactive features of the application function correctly. This testing involves powering up the application and ensuring the smart home is displayed properly.

Additionally, we test the different views of the house and confirm that they are navigable. For example, we have a pre-saved view of the solar panels on the house. Upon clicking the button to access that view, the camera pivots to the predefined location, displaying educational and system component information related to the solar panel. This testing step ensures that each view button is responsive and displays accurate information.

We also verify that the correct animations are shown based on the system's status. For instance, we monitor the current flowing from the solar panel to the battery. This integration test involves verifying two aspects: 1. Receiving accurate information from the Raspberry Pi, and 2. Displaying that information correctly in the UI. Additionally, the UI should respond appropriately to the component statistics. If the battery is being charged by the solar panel, the application should display an animation showing energy flowing into the battery. This behavior is another key aspect to test in our system.

Another part of the system is the ability to adjust the physical state of the smart home hardware. The UI application allows the user to control parameters like the load and time of day. When these settings are changed, the physical components adjust to match the configuration. During software testing, we verify that all expected configuration options are available and functional. After changing the configuration, we also ensure that the simulated smart home updates accurately in the UI application.

Latency is an essential factor in testing a software application. To evaluate the system's latency, we test several aspects. First, we analyze the application's responsiveness, including the time it takes to respond to touchscreen commands. For instance, if a user selects an option to view the home, we measure how quickly the system executes the action. Additionally, we evaluate the latency of implementing configuration settings. This involves measuring the time from selecting a configuration to the Raspberry Pi adjusting the physical components to match. By analyzing these metrics, we can determine whether the system meets our responsiveness requirements.

Finally, we test the UI's response to detecting a cyberattack. When the algorithm identifies a cyberattack, the UI should respond appropriately, showing the user what happens during this event.

After completing all these tests, we can confidently conclude that the system performs as intended. Testing the hardware and software communication and verifying their responses are critical to ensuring the system functions seamlessly.

9.3 Integration

This Senior Design project relies heavily on successful integration between the hardware and software components. The flow of the application begins with the user, who interacts with a UI application to control the smart home and adjust various parameters. When these parameters are configured, the application communicates with the Raspberry Pi via serial and adjusts the hardware accordingly through relays.

In the system, the user can adjust three parameters: load, power source, and time of day. During testing, we verify that when reconfiguration occurs, the system sends messages to the Raspberry Pi, and the Raspberry Pi adjusts the hardware components. For example, if the time of day is set to daytime, the Raspberry Pi should switch on the lamp in our system and begin charging the solar panel. This behavior is verified by observing the system's response.

To perform this testing, we monitor three aspects: the application, the Raspberry Pi's message handling, and the physical components. Starting with the application, we configure the system by setting parameters and sending that information to the Raspberry Pi. At the Raspberry Pi, we ensure it handles the messages correctly and directs the

physical components accordingly. Finally, we observe the relays to confirm they adjust the hardware to match the system configuration. This testing allows us to verify that the communication pipeline is functioning as expected.

Another critical aspect of the system to test is the communication of component statistics. On the hardware side, the system monitors the status of various components (e.g., current, power). The Raspberry Pi sends this data to the UI application, where it is displayed. This test ensures consistency between the data collected by the Raspberry Pi and the values displayed on the application. To verify this, we monitor the component statistics on the Raspberry Pi and validate that these values match those shown on the application.

This project requires seamless integration between two large systems to operate correctly. By performing these tests, we ensure that all components of the system interact properly, confirming its overall functionality.

9.4 Plans during Senior Design 2

During Senior Design 1, our group set a goal to prepare prototypes of our system and finalize the implementation in Senior Design 2. This applies to both our software and hardware components. We then continued to work more into integration and finally we came to the conclusion of winning the Senior Design Showcase for Fall 2024 for Best In Show.

Utilizing Jira to track tasks, we set a goal to have a working prototype of the UI application by the end of the summer. We evaluated the features we needed to add and next steps for refining the system. We tested the integration, latency, user experience and the features of the system.

To begin, the integration of this system involved communication with the Raspberry Pi, receiving component status data and sending system control configurations. To evaluate the component status information, this involved watching the UI application and using electronic tools, such as multimeters to view the current, power, etc. Doing this allowed us comfort in that the correct information is being sent over to the application. From this, we gained confidence that our system works.

Additionally, in senior design 2, we constantly tested the system latency and determined if our hardware meets the performance requirements. We used new techniques in data collection that helped implement the hardware running the application and determine that we only needed to optimize the application and that our hardware was running up to standards we were expected of us.

Regarding the user experience, we invited several people to test the application and provide comments on how to improve their experience. From there, we were able to

improve the system from a user's perspective. This took place several times throughout SD2, which helped improve our application.

The last thing we were determined to accomplish was which features we kept. In Senior Design 1, we implemented several features that our team thought should be in the application. From the user testing we hosted, we brought in several different people and our research lab sponsor to use the system and provided input on what features could improve the application.

Upon the conclusion of Senior Design 2, we kept on improving and adding remaining features to our application and its integration as the deadline approached. By preparing our components in Senior Design 1, we were equipped to successfully implement and prepare for the final delivery and presentation in Senior Design 2. We successfully accomplished the tasks we expected of us.

10. Administrative Content

10.1 Budget and Financing

An itemized list of the materials required for the project, their costs, and the quantity needed is shown in Table 24. The costs of these materials were covered by the UCF Digital Grid Laboratory. Due to this, we had to have strict guidelines on the amount of funds spent on this project. As shown in the table.

Table 43: Second Draft Budget

Item	Price	Qty	Total Cost	Details
Controller for Solar Panel	$24.99	1	$24.99	Controller
Renogy Monocrystalline Solar Panel	$104.99	1	$104.99	Power Source
Renogy Battery (Sealed)	$189.99	1	$189.99	Power Source
HiLetgo 12V 8 Channel Relay Module	$11.29	1	$11.29	EMS Control

Phillips 55BDL4051T Touchscreen	$1,678.00	1	$1,678.00	User Interface
Renogy Inverter	$119.99	1	$119.99	DC/AC Conversion
Meter (V, I, P)	$14.99	2	$29.98	Metering for AC Source/Load
Meter (V, I, P)	$14.99	3	$44.97	Metering for DC Load
DC Fan	$15.99	1	$15.99	DC Load
AC Lamp	$9.99	1	$9.99	AC Load
Raspberry Pi 4 Model B	$79.49	1	$79.49	EMS and Communication with OPAL-RT / Application
Printed Circuit Board (5)	$108.99	1	$108.99	Used for Metering

Cyber S.H.I.E.L.D. is funded through the Siemens Digital Grid Lab at UCF. This is led by the Director of the Siemens Digital Grid Lab, Dr. Wei Sun. Dr. Sun. His research dives into the electrical power and energy systems. His goal with helping fund our project is to give his research a baseline field testbench for his future trials for a cyber attack algorithm. His method of knowledge in interdependent critical infrastructure and cyber-physical security led him to sponsor our project to further his research.

The Siemens Digital Grid Lab is located in the Harris Engineering Center (HEC), Room Number 302. Their mission is to provide new features to the Siemens Commercial Software and beta test different designs that are being researched daily. The lab also conducts research in optimal operation of transmission and distribution systems with high penetration of renewable energies, stochastic modeling of power systems, protection of PV farms, real-time monitoring of transmission and distribution systems, distribution system automation, and power system restoration and resilience analysis [51]. The packages are then also being enhanced to incorporate the new research and functionalities. The Facility comes with state of the art technology, for our project, we primarily will focus on the Spectrum Power Microgrid Management System (MGMS), Spectrum Power Advanced Distribution Management System (ADMS), Power System Simulator for Engineering (PSS/E).

10.2 PCB Bill of Materials (BOM)

Shown in the tables below is the bill of materials for the solar panel metering PCB and the battery metering PCB.

Table 44: BOM for Battery Metering PCB

Battery Metering PCB		
Reference	Value	Qty
C1,C18,C19	10u	3
C2,C4,C9,C11,C12,C21,C22,C23	100n	8
C3,C7,C8	22u	3
C5,C6	10p	2
C10	1n	1
C16,C17,C20	1u	3
D1	LED	1
D2	LED_Filled	1
D5	SP0505BAHT	1
J1,J2	Screw_Terminal_01x02	2
J4	USB_C_Receptacle	1
L1	4.7u	1
L2	VLP8040T-3R3N	1
Q1,Q2	BS108	2
R1	232k	1
R2,R3,R7	30k	3
R4	88.7k	1
R5,R8,R13,R14,R15,R18	10k	6
R6	220k	1
R9	45.3k	1
R10	130	1
R11	6.8k	1
R12	12k	1
R16,R17	5.1k	2

R19	300	1
R20	24k	1
R21	9.1k	1
U1,U2	XTPS564257DRLR	2
U3	ESP32-WROOM-32D	1
U4	ACS712xLCTR-20A	1
U7	CH340C	1

Table 45: BOM for Solar Panel Metering PCB

Solar Panel Metering PCB		
Reference	Value	Qty
C1,C10,C11,C13	0.1uF	4
C2,C4	100nF	2
C3	47uF	1
C5	10uF	1
C6,C8,C12	22uF	3
C7	33nF	1
C9	4.7uF	1
D1	LED	1
D2,D3,D4,D5,D6,D7,D8	D_Zener	7
J1	USB_A	1
L1	SRR1208-100ML	1
Output1,Output2	Screw_Terminal_01x02	2
R1	1k	1
R2,R7,R10	2k	3
R3	100k	1
R4,R6,R11,R12	10k	4
R5	52.3k	1

R8	22.1k	1
R9	47.5k	1
U1	TPS62932DRLR	1
U2	ACS723xLCTR-10AB	1
U3	ADS1115IDGS	1
U4	ESP32-WROOM-32-N4	1
U5	AMS1117-3.3	1
U6	CP2102N-Axx-xQFN28	1

10.3 Initial Project Milestones

In order to stay on track, the team made a table of project milestones with what tasks were accomplished at what time. This table details the time frames for each part of the project and what team member worked on that milestone.

Table 46: Project Milestones

Senior Design I - Summer 2024			
Task	**Time Period**	**Dates**	**Responsibility**
Project Selection	2 Weeks	5/13 - 5/17	All
D&C Documentation	1 Week	5/17 - 5/31	All
D&C Meeting With Advisors	30 Minutes	6/7	All
Research Energy Monitoring	2 Weeks	6/3 - 6/17	Nicole, Adam
Research Designing EMS	2 Weeks	6/3 - 6/17	Nicole, Adam
Research Connection of Raspberry Pi 4 to Relays	2 Weeks	6/3 - 6/17	Karan
Research User Interface	2 Weeks	6/3 - 6/17	Jordan
Research Wired Connection to OPAL-RT to Send Data	2 Weeks	6/3 - 6/17	Karan

Control Smart Relays to Direct Flow of Power	1 Week	6/17 - 6/24	Nicole/Karan
Start Building UI for Touch Screen	4 Weeks	6/24 - 7/22	Jordan
Research Communication Between Relays, Meters, UI	2 Weeks	6/24 - 7/8	All
Start PCB Design	3 Weeks	6/24 - 7/15	Nicole, Adam
Final PCB Design	1 Week	7/15 - 7/22	Nicole, Adam
Order PCB and Other Materials	1 Week	7/22 - 7/29	All
Start Project Prototype	2 Weeks	7/22 - 8/2	All
60 Page Draft	2 Weeks	7/1 - 7/15	All
Final Documentation	6 Weeks	6/21 - 8/2	All

Senior Design II - Fall 2024			
Task	**Number of Weeks**	**Dates**	**Responsibility**
Model Building	4 Weeks	8/19 - 9/9	All
Complete UI	4 Weeks	8/19 - 9/9	Jordan
Connect the MPU and Touch Screen	4 Weeks	8/19 - 9/9	Jordan, Karan
Construct Energy Management System	4 Weeks	8/19 - 9/9	Nicole, Adam
Connect the Energy Management System to the Raspberry Pi 4	4 Weeks	9/9 - 10/7	Nicole, Adam, Karan
Connect Raspberry Pi 4 to the Touch Screen	4 Weeks	9/9 - 10/7	Karan, Jordan

Connect Raspberry Pi 4 to OPAL-RT	2 Weeks	9/23 - 10/7	Nicole, Adam, Karan
Construct The Final Communication Channels	4 Weeks	10/7 - 11/4	All
Connect PCB to the Raspberry Pi 4 and the EMS	4 Weeks	10/7 - 11/4	All
Final Adjustments	2 Weeks	11/4 - 11/18	All
Final Presentation	1 Hour	11/20	All

10.4 Table of Work Distributions

In order to maintain organization throughout the project, we made sure to create a table that distributed the tasks that each project member would complete. This table detailed the primary member that would be responsible for completing the project task and the second member that would assist in that task. This can be seen in Table 46.

Table 47: Work Distributions

Task	Primary	Secondary
Designing Model on RT-Lab	Nicole	Adam
Metering Device PV	Nicole	Adam
Metering Device for Battery	Adam	Nicole
Designing User Interface	Jordan	Karan
Creating Interactivity of UI	Jordan	Karan
Implementing Communication Between Raspberry Pi & UI	Jordan	Karan
Monitoring of DC Load	Adam	Nicole
Connection of DC Load	Adam	Nicole
Monitoring of AC Load	Nicole	Adam

Connection of AC Load to Battery (Inverter)	Nicole	Adam
PCB to Raspberry Pi	Adam	Nicole
Program Raspberry Pi to Control Relays	Nicole	Adam
AC Source Connection	Adam	Nicole
EMS Algorithm Code	Karan	Nicole
OPAL-RT Communication	Karan	Nicole
Receiving and Sending Messages to UI	Jordan	Karan
Communication Management for Messages Received	Jordan	Karan

11. Conclusion

Cyber S.H.I.E.L.D. is a culmination for our 4+ years at UCF. It demonstrates our motivation and implementation of the different periods of our majors. To take a concept that will be founding a future in the cyber protection industry and being able to show the education purposes, to then be able to create a bench plate for additional testing to be done is a mark on what we have done at UCF. To create a proper simulation, the Hardware we have chosen, the communication lines that we have decided upon, and the implementation of the software all need to be in harmony with one another. All of these will in turn create a functioning simulation of a Smart Home System. All the documentation we have provided is to ensure this project's success in the next semester.

We are motivated to learn new techniques and technology to make our mark at UCF. We want to ensure that the Research Project that we brought into the SIEMENS Digital Grid Lab will be successful and will help continue further research. Being able to demonstrate the gateway towards a cyber protected household is what we hope to accomplish.

Our documentation shows the lengths of the time and effort it took to do our due diligence on research and development. We examined multiple different assets we can use for our project, ranging from Battery Sources to the Microcontroller we will be using on our PCB shown in Section 3. We had dived into the Standards and Constraints we will be needing to accommodate for in Section 4, taking into real world scenarios where we would need to show our system acknowledging those problems and solving them.

In section 5, we provide a deep level of comprehension when it comes to the decision behind which programming language the Raspberry Pi 4: Model B will be using to control the Smart Home Management System. The depth of the decisions came to become the backbone for ensuring an easier transition to Senior Design 2. Section 6 goes into detail on how we design our prototyping and communications between those pieces.

To continue from the previous passage, Section 7, 8 and 9 go into details on the User Capabilities, the software/hardware that goes into making it work, and the integration of those pieces into the full picture. This project has been described in detail to the fullest extent possible. Including the background and motivation for its implementation. The final sendoff to the project is the drive and passion of the 4 team members accomplished with this Smart Home.

Appendix

[1] Communications with the grid edge. (n.d.). https://www.energy.gov/sites/default/files/2023-07/Communications with the Grid Edge - Unlocking Options for Power System Coordination and Reliability_0.pdf

[2] Opengreenenergy, & Instructables. (2024, March 20). DIY Solar Panel Monitoring System – v2.0. Instructables. https://www.instructables.com/DIY-Solar-PV-Monitoring-System-V20/

[3] YouTube. (2023, June 10). Smart Hybrid Energy Management System using Arduino Code & Circuit. YouTube. https://www.youtube.com/watch?v=A9fEb-v0cCw

[4] Collazo, Marks, et. al. (May 2020) Power Systems Knowledge Hub.

[5] “100 Watt 12 Volt Monocrystalline Solar Panel.” Renogy. https://www.renogy.com/100-watt-12-volt-monocrystalline-solar-panel-compact-design/

[6] “MEGA 100 Watt Monocrystalline Solar Panel.” Rich Solar. https://richsolar.com/products/100-watt-solar-panel

[7] “Wanderer 10A PWM Charge Controller.” Renogy. https://www.renogy.com/wanderer-10a-pwm-charge-controller/

[8] “BlueSolar MPPT 75/10, 75/15, 100/15 & 100/20.” Victron Energy. https://www.victronenergy.com/solar-charge-controllers/mppt7510

[9] “Deep Cycle AGM Battery 12 Volt 100Ah.” Renogy. https://www.renogy.com/deep-cycle-agm-battery-12-volt-100ah/

[10] “WEIZE 12V 100Ah Deep Cycle AGM.” WEIZE. Batteryhttps://www.weizeus.com/products/12v-100ah-deep-cycle-agm-sla-vrla-battery?variant=44660070645988

[11] “ML100-12GEL – 12 VOLT 100 AH, GEL TYPE.” Mighty Max. https://www.mightymaxbattery.com/shop/12v-sla-batteries/12v-100ah-gel-battery/

[12] “200 watt Off-grid Solar Inverter 12VDC Cotek SE200-112.” SunWatts. https://sunwatts.com/200-watt-off-grid-solar-inverter-12vdc-cotek-se200-112/

[13] “700W 12V Pure Sine Wave Inverter.” Renogy. https://www.renogy.com/700w-12v-pure-sine-wave-inverter/

[14] “SI1000 1000 WATT POWER CONVERTER.” Schumacher. https://www.schumacherelectric.com/products/si1000-1000-watt-power-converter/

[15] “12V 8 Channel Relay Module With Optocoupler High And Low Level Trigger.” HiLetgo. http://www.hiletgo.com/ProductDetail/2157815.html

[16] “ANMBEST Relay Module.” SnapKilk. https://snapklik.com/en-ca/product/anmbest-relay-module-with-optocoupler-high-low-level-trigger-for-arduino-12-12v-relay-16-channel/09J14PS7201A5

[17] “Solid State Relay 25A DC to DC Input 3-32V DC to Output” https://www.amazon.com/dp/B0D5CG2DKB?ref=ppx_pop_mob_ap_share&th=1

[18] “Home.” Samsung PM49H 49 Inch Smart Signage Touch Screen Display. www.touchwindow.com/c/SamsungPM49H.html.

[19] “Philips Signage Solutions Multi-Touch Display 55BDL4051T/00.” Philips. www.usa.philips.com/p-p/55BDL4051T_00/signage-solutions-multi-touch-display.

[20] “LG 49TA3E: 49” Class TA3E Series” LG. www.lg.com/us/business/digital-signage/lg-49TA3E

[21] ATmega328. MicroChip. https://www.microchip.com/en-us/product/atmega328.

[22] MSP430FR6989. Texas Instruments. https://www.ti.com/product/MSP430FR6989.

[23] “ESP32-C3-WROOM-02U-N4. Digikey. https://www.digikey.com/en/products/detail/espressif-systems/ESP32-C3-WROOM-02U-N4/15222540

[24] “Arduino Uno Rev3.” Arduino. https://store-usa.arduino.cc/products/arduino-uno-rev3/?selectedStore=us

[25] “Arduino Ethernet Shield 2.” Arduino. https://store-usa.arduino.cc/products/arduino-ethernet-shield-2?selectedStore=us

[26] “Portenta H7.” Arduino. https://store-usa.arduino.cc/products/portenta-h7?selectedStore=us

[27] “Portenta Vision Shield - Ethernet.” Arduino. https://store-usa.arduino.cc/products/arduino-portenta-vision-shield-ethernet?selected

[28] “Raspberry Pi 4.” Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[29] “Large Language Models Explained.” Nvidia. https://www.nvidia.com/en-us/glossary/large-language-models/

[30] “Introducing ChatGP.” OpenAI. https://openai.com/index/chatgpt/

[31] “What is ChatGPT?” OpenAI.
https://help.openai.com/en/articles/6783457-what-is-chatgpt

[32] “GPT-4.” OpenAI. https://openai.com/index/gpt-4-research/

[33] “The Advantages & Disadvantages of Chat GPT.” Appmatics.
https://www.appmatics.com/en/blog/vorteile-nachteile-chat-gpt

[34] “Introducing Microsoft 365 Copilot.” Microsoft.
https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/

[35] “Introducing Gemini: Our Largest and Most Capable AI Model. Google.
https://blog.google/technology/ai/google-gemini-ai/#performance.

[36] “HDMI”. WikiPedia. https://en.wikipedia.org/wiki/HDMI.

[37] “UART”. WikiPedia.
https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter.

[38] “SPI”. WikiPedia. https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[39] “I2C”. WikiPedia. https://en.wikipedia.org/wiki/I%C2%B2C.

[40] “Wi-Fi” WikiPedia. https://en.wikipedia.org/wiki/Wi-Fi

[41] “Bluetooth” WikiPedia. https://en.wikipedia.org/wiki/Bluetooth

[42] “Applying IPC-2221 Standards in Circuit Board Design.” Sierra Circuits.
https://www.protoexpress.com/blog/ipc-2221-circuit-board-design/

[43] “IPC-J-STD-001: The Standard for Soldering Assemblies.” NextPCB.
https://www.nextpcb.com/blog/ipc-j-std-001

[44] “IPC-6012 or IPC-A-600: Which Standard Should You Use?” Sierra Circuits.
https://www.protoexpress.com/blog/ipc-6012-ipc-600-standard-use/

[45] YouTube. (2022, November 27). How to configure a USB type-C connector for your arduino project. YouTube. https://www.youtube.com/watch?v=c0yRkn_vO3Y

[46] “What is an LDO regulator?” Toshiba.
https://toshiba.semicon-storage.com/us/semiconductor/knowledge/e-learning/basics-of-low-dropout-ldo-regulators/chap1/chap1-3.

[47] “Real Time Simulation Solutions”. OPAL-RT. https://www.opal-rt.com

[48] “Hardware-in-the-Loop Evaluation of an Advanced Distributed Energy Resource Management Algorithm” Jing Wang, Jeff Simpson, Rui Yang, Bryan Palmintier, Soumya Tiwari, and Yingchen Zhang. https://www.nrel.gov/docs/fy21osti/80765.pdf.

[49] “Real Time Digital Simulator: OP5600.” OPAL. www.opal-rt.com/simulator-platform-op5600/.

[50] “OP5707 RCP/HIL FPGA-Based Real-Time Simulator”. OPAL-RT. https://blob.opal-rt.com/medias/L00161_0128.pdf

[51] “Siemens Digital Grid Laboratory” University of Central Florida. https://www.ece.ucf.edu/~qu/labs/siemens-digital-grid-laboratory/