ECE44X Senior Design Team 26

Hobby Hub

Project Documentation

5/14/2022

Jia Wei Cheng chengjia@oregonstate.edu
Matthew Gragg graggm@oregonstate.edu
Rene Aimba aimbar@oregonstate.edu
Kristina Mason masonkr@oregonstate.edu

1. Overview

1.1 Executive Summary

The goal of this project is to design a “smart home”, that entails a smart hub and a smart plug and can control IOT (Internet of Things) devices. These devices can be togglable plugs for lamps, appliances, etc. controllable lights, or other devices used to control or manage other home devices. Initially, the project currently entails a wall plug/socket that can be controlled remotely through WIFI via Smart hub that uses a Raspberry Pi Microcontroller.

1.2 Team Contacts and Summary

Team Member	Email	Primary Project Role
Jia Wei Cheng	chengjia@oregonstate.edu	Smart Plug Software
Matthew Gragg	graggm@oregonstate.edu	Smart Hub Software
Renee Aimba	aimbar@oregonstate.edu	Smart Hub Hardware
Kristina Mason	masonkr@oregonstate.edu	Smart Plug Hardware

Table 1:Team Member Information

Don’t work more than 2 hours at a time without 20-30 minute breaks.
Meet-up weekly (in-person or virtually) to relay any gained progress/knowledge regarding the project.
Everyone should have 1-2 pieces of feedback on any drafted technical document (if possible).
Hold meetings whenever there’s technical decisions/designs that need to be finalized.

Table 2:Team Protocol

1.3 Gap Analysis

This smart-home hub allows for both hobbyists and professionals alike to create an interconnected web of devices in order to make one's life more convenient. The product allows one to customize their home connections to fit with their specific needs. It will also be open source, as fits our assumption that the user may want to add their own custom built devices to their network.

The smart-home hub can control the lights from anywhere via a web-device which can be beneficial in terms of energy efficiency and/or automation. Users should be able to control the devices easily from an application, and after verification, the system provides the user a web application that 9 out of 10 users say is easy to use. The end user will be anyone who is into employing an energy efficient system, to wirelessly control certain devices.

1.4 Timeline

	Terms	Fall Term									
WBS NUMBER	Week	1	2	3	4	5	6	7	8	9	10
	TASK TITLE										
1	Project Conception and Initiation										
1.1	Project Research										
1.2	Project Design										
1.3	Prototype Ideas										
1.3.1	Prototype Design										
1.3.2	Prototype Build										
1.4	Design Revision										

Figure 1: Timeline Gantt Chart

1.5 Reference and File Links

[1] Taifur and Instructables, "Smart plug," *Instructables*, 12-Oct-2017. [Online]. Available: https://www.instructables.com/Smart-Plug/. [Accessed: 14-Oct-2022].

1.6 Revision Table

Revision	Description
3/8/2023	Kristina: Added new sections/content based off of Project Document Content Guide and recent accomplishments
3/8/2023	Renee: made changes to the previous content we had included in the document at the beginning of fall term
1	Initial Write-up
2	Changed timeline from a picture to a google sheet gantt chart.

2. Impact and Risk

2.1 Design Impact Statement

1. Introduction

The point of this document is to assess and analyze potential risks and impacts of Smart Home System designed by Project Group #26. This document will analyze risk and potential harms/risk for public health, possible cultural and societal implications, environmental impacts, and economic factors and costs. This document will contain reference and example to justify the analysis presented. Possible solutions and risk mitigation techniques will also be proposed.

2. Public Health, Safety, and Welfare Impacts

The obvious safety factor when considering a home automation system such as this is system and software security. With possible personal devices being controlled via the internet, this creates opportunities for exploitation of system oversights and vulnerabilities, which can lead to a user's personal data being visible to hackers. Although these risks are small, these are risks that still exist and should be taken into account.

3. Cultural and Social Impacts

There could also be potential societal impacts with owning a smart home system. In general, such a system would provide one with better convenience and further

accessibility with controlling home appliances. Hence, giving families, who are capable of owning such a system, more capability within their homes.

4. Environmental Impacts

A home automation system could potentially be eco-friendly if configured a certain way; such as timer based solutions which turn off appliances based on time passed. However, there could be environmental impacts if the solution is not fully well rounded. In general, the hub of a smart home system has to always be continuously running as each manages the statuses of all the other home appliances; hence, in a flawed solution, the hub could be potentially wasting electricity.

5. Economic Factors

When it comes to the economic impacts of smart home devices some points can be derived immediately. One, smart home devices have the potential to save energy costs, especially for homeowners, smart home devices establish a new consumer market, and three smart home and more specially internet of things devices have much greater impact in places where readily available internet access is normal. These claims are mostly referring to smart home systems such as Google Nest, Hub, etc. that are mass market solutions with a large product ecosystem, as tangible economic and cost benefits for automation services scale with home many facets of the home can be automated. From our product angle, we are shooting for a DIY/prototyping angle. We want to deliver an ecosystem of products that other engineers and tinkers can use to test out their ideas. This means that evaluating these economic impacts get more difficult as the general use case for our project is very general.

Energy Usage

A core idea of our project is a smart plug, that can turn on and off home appliances, lights, etc from an app, and track energy consumption. In the ScienceDirect paper, “Environmental Impacts and Benefits of Smart Home Automation,” Nicloas, ANtonio, Kuako, and Eva complete a full carbon footprint and energy consumption evaluation on Finnish households who have installed internet-controlled energy systems. They found that those systems alone proved a 12% decrease in energy consumption annually [x]. Even more energy saving could be achieved, they highlight, as these energy management systems could also be applied to the aging grids that the homes were running off of, meaning integration at the city level could further reduce energy cost and consumption. Logically these benefits could not be fully extended to just internet controlled wall plugs. The plugs alone then could contribute to a consumer trend towards being mindful about energy usage.

Costs to Maintain an Initial Costs:

According to a home advisor, the average cost of a full home automation system is around 820-1440$[3]. This varies wildly based on how many automated devices are used. Our requirements put us at a much lower price point at around 300$. As for

maintenance costs, this is also hard to judge. Most home automation systems need little maintenance, and this project scope is not intended to be a large scale product designed for mass consumption. The average user of a linked arduino system is not going to be someone looking for this product in an attempt to save money.

6. Conclusions

In conclusion, the impacts of our design can be summarized mostly in the relation to the homeowner, as mass production is not planned. The security of the software is determined as the most important risk that must be minded.

2.2 Risks

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Performance Indicator	Action Plan
R1	Parts Delay	Timeline	M	M	Parts get delayed.	Order early
R2	Design Flaw	Technical	M	H	Design requirements are still not met after design discussion.	Perform early prototypes and revisions
R2	Software usability is not sufficient.	Technical	M	H	User surveys do not indicate ease of use.	User guided design decisions.

Table 3: Risks and their descriptions

2.3 References and File Links

[1] “How to stop your smart plug from giving hackers access to devices on your Wi-Fi Network,” *HT Tech*, 23-May-2021. [Online]. Available: https://tech.hindustantimes.com/home-appliances/news/how-to-stop-your-smart-plug-from-giving-hackers-access-to-devices-on-your-wi-fi-network-71621775987120.html. [Accessed: 04-Nov-2022].

[2] R. J. Pierce, “Smart plug users? be careful if you have a cheap one, because it's vulnerable to hacking,” *Tech Times*, 23-May-2021. [Online]. Available: https://www.techtimes.com/articles/260574/20210523/cheap-smart-plug-vulnerable-hacking.htm. [Accessed: 04-Nov-2022].

[3] J. Chung, “The role of culture in adopting Smart Home Technologies,” *SpringerLink*, 01-Jan-1970. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-3-319-01583-5_58. [Accessed: 04-Nov-2022].

[4] D. D. F. D. Rio, B. K. Sovacool, and S. Griffiths, “Culture, energy and climate sustainability, and Smart Home Technologies: A mixed methods comparison of four countries,” *Energy and Climate Change*, 23-Apr-2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266627872100012X. [Accessed: 04-Nov-2022].

[5] J.-N. Louis, A. Calo, K. Leiviskä, and E. Pongrácz, “Environmental impacts and benefits of Smart Home Automation: Life Cycle Assessment of Home Energy Management System,” *IFAC-PapersOnLine*, 17-Jun-2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315001597. [Accessed: 04-Nov-2022].

[6] D. Yang Meier, P. Barthelmess, W. Sun, and F. Liberatore, “Wearable Technology Acceptance in health care based on national culture differences: Cross-country analysis between Chinese and Swiss consumers,” *Journal of medical Internet research*, 22-Oct-2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7644382/. [Accessed: 04-Nov-2022].

[7] M. Ghazal, M. Akmal, S. Iyanna, and K. Ghoudi, “Smart plugs: Perceived usefulness and satisfaction: Evidence from United Arab Emirates,” *Renewable and Sustainable Energy Reviews*, 28-Aug-2015. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S1364032115007431. [Accessed: 04-Nov-2022].

[8] Mariya-Greeley, “Living like the jetsons: The benefits (and risks) of Smart HomeTechnology,” *Sponsored*, 07-Nov-2018. [Online]. Available: https://sponsored.bostonglobe.com/future-forward/smart-home-technology/. [Accessed: 04-Nov-2022].

[9] Stanislav, Mark and Beardsley, Tod, “Hacking iot: A case study on baby monitor exposures and vulnerabilities,” *Rapid7 Report*, 2015. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/63/2015/11/21031739/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf. [Accessed: 18-Nov-2022].

2.4 Revision Table

Revision	Description
1	Initial Write-up of Impact statement and risks.
2	Added to Public Health, Safety, and Welfare Impacts
3	Edited formatting

3. Top-Level Architecture

3.1 Block Diagram

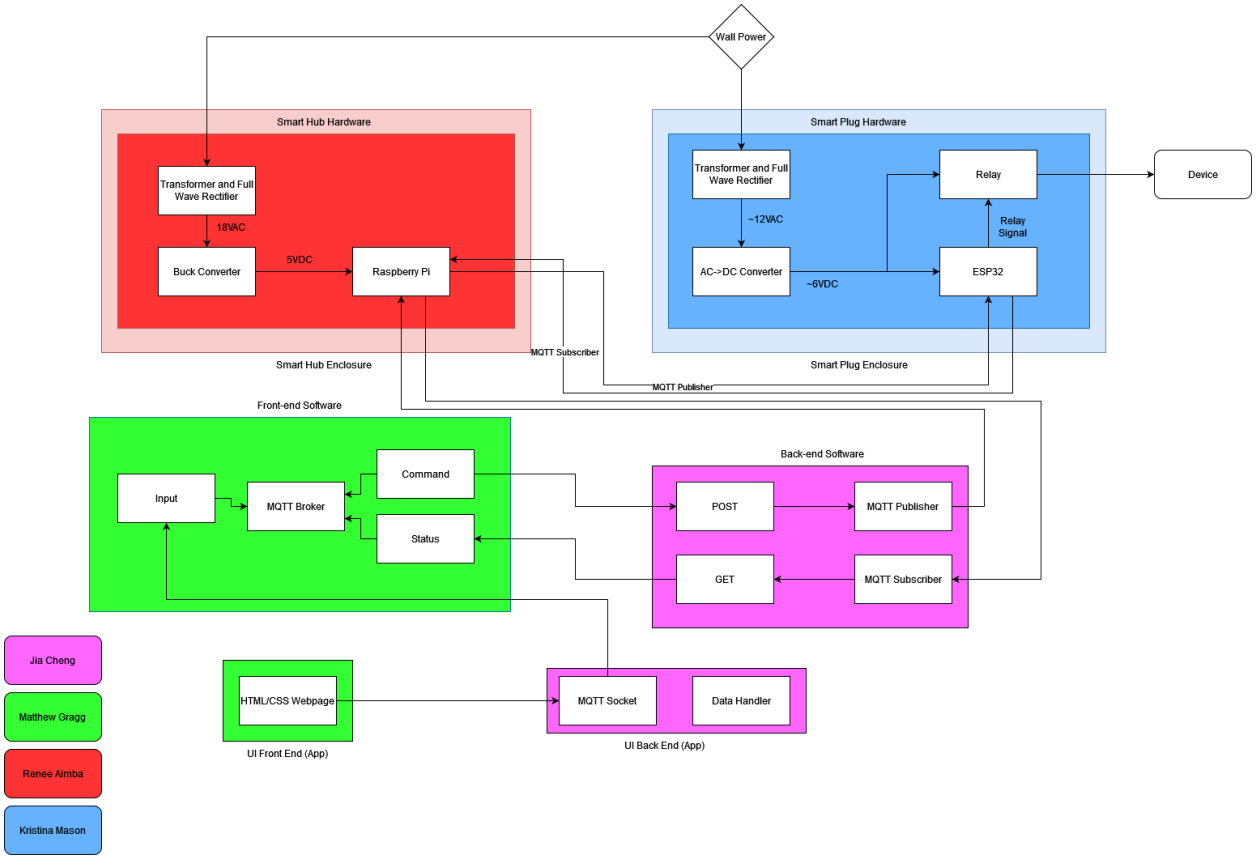

Figure 2: System Block Diagram

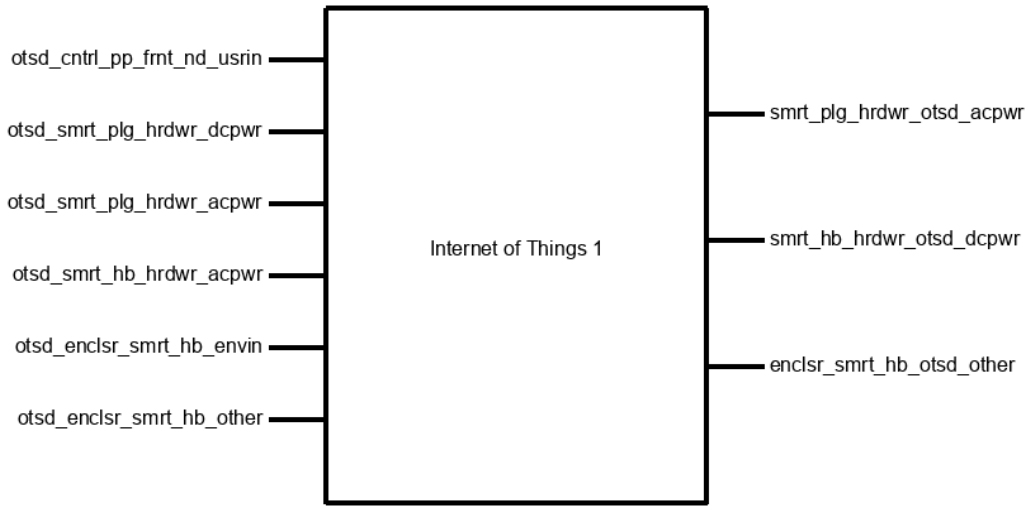

Figure 3: Black Box Diagram

3.2 Block Descriptions

Name	Description
Smart Hub Software Champion: Matthew Gragg	The Smart hub will be a HTTPS socket layer/server, also using MQTT to schedule updates to ESP32's running MQTT scheduling software.
Control App Front End Champion: Matthew Gragg	This block is the front-end/design of our companion web application. This application will be used to send commands to the main hub, and will communicate directly to the backend of our web application. From there, commands will be sent to the hub. This application will allow the user to the smart plug/switch, allowing for remote switching of outlet power. This application will also allow the user to send generic commands to other connected devices. This application will also show the status of current connected devices on the hub network. As stated earlier, all direct connections and communication will be done with the web applications backend, which will handle sending the proper commands and organization of incoming status messages from connected devices on the network.
Control App Back End Champion: Jia Wei Cheng	A HTTPs socket-based backend for the User control Application. This will send the status of user input and commands to the hub, along with the desired device. The hub will also send status of connected devices to this block.
Smart Plug Software Champion: Jia Wei Cheng	Microcontroller code that receives signals from the Hub via wifi and acts accordingly to turn on/off the connected appliances.
Smart Plug Hardware Champion: Kristina Mason	The hardware and circuitry required to create a smart plug in which it will act as a wirelessly activated relay activating an incandescent or LED light bulb usually placed into lamps. Inside of the Smart Plug will be an ESP32 microcontroller along with a transformer that will turn the voltage output from the microcontroller into a voltage high enough to power lightbulbs and other AC powered devices.

Smart Hub Hardware Champion: Renee Aimba	This is a smart home hub implemented to power a Raspberry Pi module which will be used as a microcontroller. The system will receive AC power via a wall outlet, which will be passed through a regulator circuit to change it to DC power, and a filter and converter system to step it down.
Enclosure Smart Plug Champion: Kristina Mason	An enclosure for the smart plug, where the circuit and microcontroller for the Smart Plug are in it. This enclosure will be able to withstand the shock of being dropped from chest height. The enclosure will also not conduct electricity to the outer world, ensuring the safety of those who will use this device.
Enclosure Smart Hub Champion: Renee Aimba	An enclosure for the smart hub that holds all the components used in the hub circuit. The enclosure is fitted with holes for heat dissipation, for when the components heat up as they are being used.

Table 3: Project Block Descriptions

3.3 Interface Definitions

Name	Properties
otsd_cntrl_pp_frnt_nd_usrin	● **Other:** Touch screen buttons. ● **Timing:** User interacts with interface using HID or touch (force update), otherwise 1hz status update when buttons are not pressed.. ● **Type:** Touch inputs, keyboard inputs (alphanumerical)
otsd_smrt_plg_hrdwr_dcpwr	● **Inominal:** .5ADC ● **Ipeak:** 1ADC ● **Vmax:** 12VDC ● **Vmin:** 5VDC
otsd_smrt_plg_hrdwr_acpwr	● **Inominal:** 5A ● **Ipeak:** 10A ● **Vnominal:** 120VAC

otsd_smrt_hb_hrdwr_acpwr	• **Inominal:** 0.0296 A • **Vmax:** 124.7 VAC • **Vnominal:** 120 VAC
otsd_enclsr_smrt_hb_other	• **Other:** Input connector 1:120 VAC connector • **Other:** Input connector 2: Power switch • **Other:** Output connector: female jumper wires and wired connection
otsd_enclsr_smrt_hb_envin	• **Other:** Height: 3 inches • **Other:** Width: 4.5 inches • **Other:** Length: 6 inches
smrt_hb_sftwr_cntrl_pp_bck_nd_data	• **Data Rate:** 1Hz update rate • **Messages:** MQTT subscriber messages (device ID, and command) • **Protocol:** MQTT
smrt_hb_sftwr_smrt_plg_sftwr_data	• **Data Rate:** 1 Hz Update Rate • **Messages:** MQTT published commands (Device Commands and device name (strings).) • **Protocol:** MQTT
cntrl_pp_frnt_nd_cntrl_pp_bck_nd_data	• **Data Rate:** 1Hz Update rate to backend. • **Messages:** Status of button presses and user input messages. Such as turning on the smart plug, changing the name of plug, etc. • **Protocol:** HTTP POST
cntrl_pp_bck_nd_smrt_hb_sftwr_data	• **Data Rate:** 1Hz Update rat. • **Messages:** Command to Turn on and off plug. • **Protocol:** MQTT

cntrl_pp_bck_nd_cntrl_pp_frnt_nd_data	• **Data Rate:** 1Hz update rate. • **Messages:** Status of devices from hub. Connected devices at hub. • **Protocol:** HTTP GET
smrt_plg_sftwr_smrt_hb_sftwr_data	• **Data Rate:** 1 Hz Update Rate • **Messages:** Connected status, current command in queue, fault codes • **Protocol:** MQTT
smrt_plg_hrdwr_otsd_acpwr	• **Inominal:** 0.08 Amps AC (With Nominal power being 10W at 120VAC) • **Ipeak:** 0.17 Amps AC (Max power being 20W at 120VAC) • **Vnominal:** 120VAC to power lamps that run on AC power
smrt_hb_hrdwr_otsd_dcpwr	• **Inominal:** 0.03 V • **Ipeak:** 0.06 A • **Vmax:** 18.8 V • **Vmin:** 17.0 V
enclsr_smrt_hb_otsd_other	• **Other:** Output connector: female jumper wires and wired connection • **Other:** Input connector 1: 120VAC connector • **Other:** Input connector 2: Power switch

Table 4: Project Interface definitions

3.4 References and File Links

3.5 Revision Table

Revision	Description
1	Initial Revision

4. Block Validation

4.1 Smart Hub Enclosure

4.1.1 Description

This is an enclosure for the smart hub system. It has plenty of ventilation for the transformer, which should only get warm and not hot. It also has holes for the Raspberry Pi ports and the Power switch module too.

4.1.2 Design

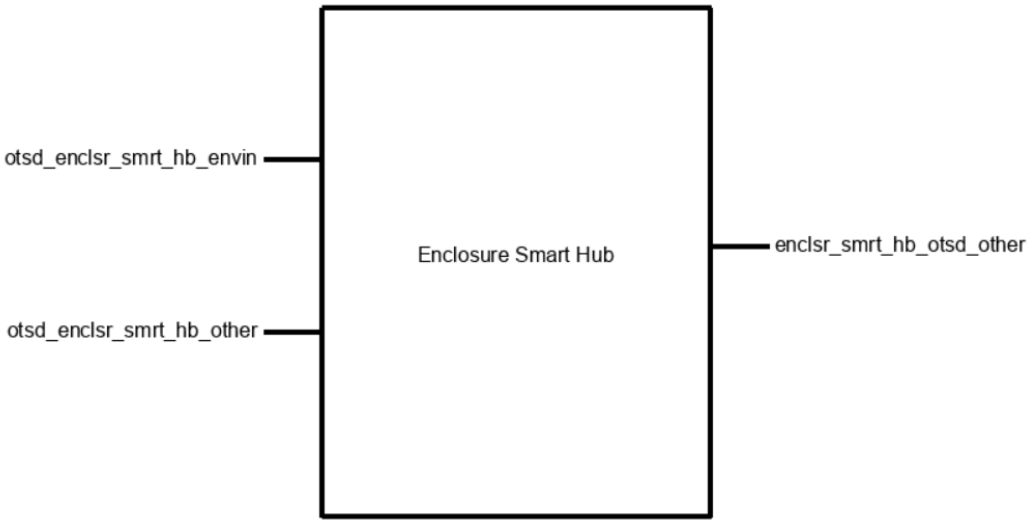

Figure 4: Smart Hub Enclosure Block Diagram

The enclosure has a height, length and width of 3”, 6”, 4.5” respectively. It also has three connectors going into and out of it. The first is the power cable that connects to the wallpower and to the exterior of the enclosure. This is enabled by the second connector which is a power switch which has a port and a switch. The final connector will be used to power the Raspberry Pi.

4.1.3 General Validation

This block entails a surface of numerous holes that enables heat dissipation for the transformer. It is also the right dimensions to encompass the whole system. It also has port connectors that enable seamless powering of the system inside it.

4.1.4 Interface Validation

otsd_enclsr_smrt_hb_envin : Input

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Other: Length: 6 inches	The total sum length of all the components added up to approximately 5.5 Inches. A little room was given to ensure distance between components.	This interface was tested and verified manually, and demonstrated to the TA's
Other: Height: 3 inches	The height of the tallest component was approximately less than 3 inches.	This interface was tested and verified manually, and demonstrated to the TA's.
.Other: Width: 4.5 inches	The total sum length of all the components added up to approximately 5.5 Inches. A little room was given to ensure distance between components.	This interface was tested and verified manually, and demonstrated to the TA's.

Table 5: Smart Hub Interface definitions

otsd_enclsr_smrt_hb_other : Input

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Other: Output connector: female jumper wires and wired connection	This is the connector between the buck converter and the Raspberry Pi.	The buck converter, and raspberry Pi module, have male header pins thus connecting them via female jumper wires is sufficient.
Other: Input connector 1:120 VAC connector	The C13 power cable has a capability to handle wall power. Our input voltage is within the voltage range of its compatibility.	According to page 2 of Power cable datasheet , the voltage Compatibility of this cable ranges from 100-250V.
Other: Input connector 2: Power switch	The C14 power switch is used alongside the C13 power cable to connect to the wall outlet.	This connector has a mount socket to connect to the C13 cable. According to page 1 of the C14 datasheet , the rated maximum voltage that it can handle is 120-250VAC.

Table 6: Smart Hub Interface definitions

enclsr_smrt_hb_otsd_other : Output

Interface Property	**Why is this interface this value?**	**Why do you know that your design details <u>for this block</u> above meet or exceed each property?**
Other: Output connector: female jumper wires and wired connection	This is the connector between the buck converter and the Raspberry Pi.	The buck converter, and raspberry Pi module, have male header pins thus connecting them via female jumper wires is sufficient.
Other: Input connector 1:120 VAC connector	The C13 power cable has a capability to handle wall power. Our input voltage is within the voltage range of its compatibility.	According to page 2 of Power cable datasheet , the voltage Compatibility of this cable ranges from 100-250V.
Other: Input connector 2: Power switch	The C14 power switch is used alongside the C13 power cable to connect to the wall outlet.	This connector has a mount socket to connect to the C13 cable. According to page 1 of the C14 datasheet , the rated maximum voltage that it can handle is 120-250VAC.

Table 7: Smart Hub Interface definitions

4.1.5 Verification Process

a. User testing:

- Using a measurement tool, measure the length.
- Using a measurement tool, measure the width.
- Using a measurement tool, measure the height.

b. Functional testing

- Connect the C13 power cable to the wall outlet
- Connect the other end of the C13 power cable to the C14 power switch (ensure the switch is off).
- Connect the female jumper wires to the raspberry Pi and turn the switch on.

4.1.6 References and File Links

a. References(IEEE)

[1]Taifur, T., & Instructables. (2017, October 12). *Smart plug*. Instructables. Retrieved January 19, 2023, from https://www.instructables.com/Smart-Plug/

a. Files

[2] C13 power cable datasheet - https://www.digikey.bg/htmldatasheets/production/1605133/0/0/1/P004-004-13A-Datasheet.pdf

[3] C14 Power switch datasheet - https://eecs.engineering.oregonstate.edu/education/inventory_datasheets/P1488483833.pdf

4.1.7 Revision Table

03/11/2023	Renee: Created the section and populated it with information about Block 2

4.2 Smart Hub Hardware

4.1.1 Description

This is a smart home hub implemented to power a Raspberry Pi module which will be used as a microcontroller. The system will receive AC power via a wall outlet, which will be passed through a regulator circuit to change its DC power, and a filter and converter system to step it down.

4.1.2 Design

Within the black box diagram of my block is a transformer, a rectifier circuit and a buck converter. The design entails power supply block, which is connected to the mains voltage and receives an input power of 120VAC. The wall power is connected to the circuit through a C13 power cable and C14 power socket with a switch. The design will power a Raspberry pi module for wireless communication.

I chose this over the ESP32 because it is a better option to load a server onto. The ESP32 is a dual-core 160 Mhz to 240 MHZ CPU,whereas the Raspberry Pi has a 1.4GHz clock speed. It was easy to install an Operating System on the Raspberry Pi as a beginner, and one can write a python script to execute tasks if they are not comfortable with Linux. It also has good GPIO functionality, and can handle tasks as well as a PC. The smart plug will include an ESP32 module which will communicate with the Raspberry Pi in the smart hub. The smart hub will transmit a signal to the smart plug, which thereafter turns on a light. This is done by an application which is available on one's phone and via a website if you would like to use a computer/laptop. This block is powered by otsd_smrt_hb_hrdwr_acpwr which is 120VAC mains voltage, and its output will be smrt_hb_hrwdwr_otsd_dcpwr to power the raspberry pi as shown by the black box representation below.

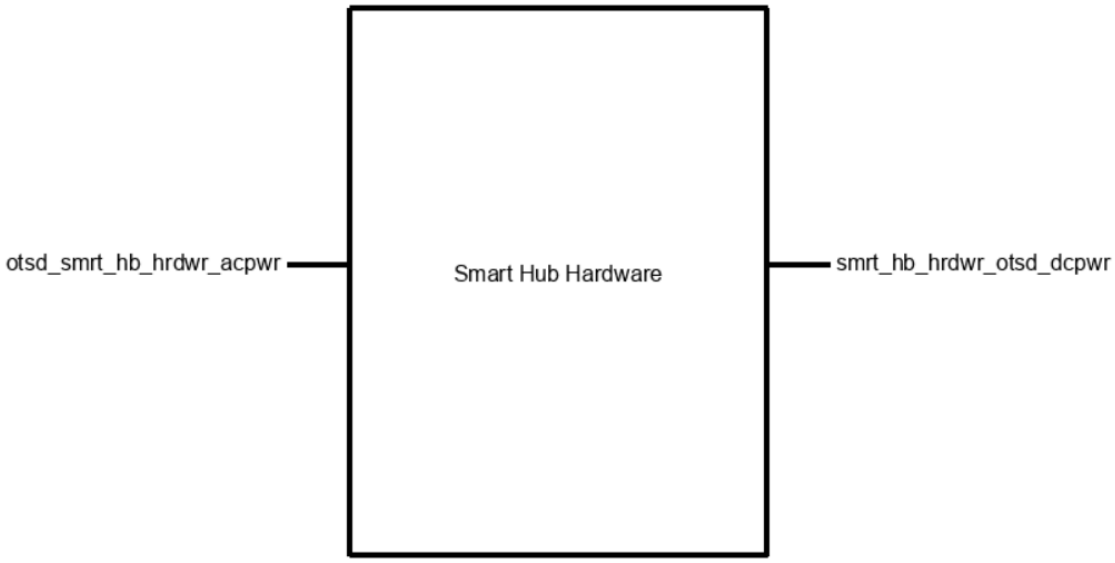

Figure 5: Smart Hub Block Diagram

4.1.3 General Validation

This block entails a lot of parts which are readily available at the TekBots store. This course of action was chosen because it is more cost effective for students, and easily accessible as opposed to waiting and paying for shipping costs after ordering parts online.The system is powered by 120VAC which is limited by a switch. When the switch is turned on that AC power is converted to DC as it passes through the rectifier, and stepped down via the buck converter. A voltage of %V is used to turn on the Raspberry Pi which turns on when connected to the converter.

4.1.4 Interface Validation

Otsd_smrt_hb_hrdwr_acpwr :

Vmax: 124VAC	This was measured with an AC ammeter	• Vin can range from 114VAC to 126VAC.
Vnominal: 120VAC	Wall power is 120VAC and that is stepped down to 5V to power the Raspberry Pi.	• To power the Raspberry Pi a nominal Voltage of 5V is required. Raspberry Pi datasheet. Pg 2
Imax: 2.5A	This is the maximum amount of current required to power Raspberry pi with USB devices connected.	• There is an expected peak output current of 2.5A @120VAC

Inominal: 1.3A	This is the current draw for a Raspberry Pi 3B+ provided it does not have any accessories connected to it.	• Predicting the actual nominal current is challenging, because the Raspberry Pi has 4 USB ports and the current draw varies with each device connected. 1.3A is used to power the Raspberry 3B and 1.2A to power the USB accessories.

Table 8: Smart Hub Interface definitions

Otsd_smrt_hb_hrdwr_dcpwr :

Vmax: 5.25V	This is the maximum amount of power that the raspberry Pi can handle.	• The TPS333D chip on the buck converter allows for a +/- 300mV ripple according to the Tech Demo specifications followed while building the buck converter.
Vnominal: 5V	Wall power is 120VAC and that is stepped down to 5V to power the Raspberry Pi.	• To power the Raspberry Pi a nominal Voltage of 5V is required. Raspberry Pi datasheet. Pg 2
Vmin: 4.7V	This is the minimum amount of power that the Raspberry Pi can handle with no accessories connected to it. However, the zero will run and there will be a lot of under voltage warnings.	• The TPS333D chip on the buck converter allows for a +/- 300mV ripple according to the Tech Demo specifications followed while building the buck converter.
Imax: 2A	This is the maximum amount of current required to power Raspberry pi with USB devices connected.	• The chip that is used for the buck converter has a maximum Io of 2A, according to

		TPS54233 Datasheet pg 12. This in turn influences the maximum DC current.
Inominal: 1.5A	This is the nominal current required to power on the Raspberry Pi.	• The buck converter allows for a 1.5 Inominal current according to the Tech Demo specifications followed while building the buck converter.
Imin: 1.3A	This is the current draw for a Raspberry Pi 3B+ provided it does not have any accessories connected to it.	• Predicting the actual nominal current is challenging, because the Raspberry Pi has 4 USB ports and the current draw varies with each device connected. 1.3A is used to power the Raspberry 3B and 1.2A to power the USB accessories.

Table 8: Smart Hub Interface definitions

4.1.5 Verification Process

a. User testing

This test will verify whether the user interface is compatible with the system.

1. Connect the system to mains voltage.
2. Switch on the power switch. The light will turn red on the switch if it is on
3. Connect the raspberry pi via a HDMI cord to a computer to see if it is powered on.

4.1.6 References and File Links

b. References(IEEE)

[1]Hugo, eM., Diego, & Pablo. (2022, February 7). *Adding battery charger to ESP8266 and ESP32 (well done)*. eMariete. Retrieved January 19, 2023, from https://emariete.com/en/co2-meter-with-battery-well-done/

[2]Taifur, T., & Instructables. (2017, October 12). *Smart plug*. Instructables. Retrieved January 19, 2023, from https://www.instructables.com/Smart-Plug/

[3]Zthipps. (2017, May 14). *DIY smart switch - part 1 how to use a Relay*. YouTube. Retrieved January 20, 2023, from https://www.youtube.com/watch?v=5NxVmg8ZFEc

c. Files

[4]Raspberry Pi 3B+ datasheet - https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[5]Wall plug schematic- https://drive.google.com/file/d/1YKgXc-X47uoHegjJA8BEpZfZEayqXGba/view?usp=share_link

[6]Hardware High Level design - https://drive.google.com/file/d/1ZUUw2BXm3I8gXhwGgrsCy52DM2sdnKof/view?usp=share_link

[7]Software High Level Design - https://drive.google.com/file/d/1ZUUw2BXm3I8gXhwGgrsCy52DM2sdnKof/view?usp=share_link

[8] TPS54233 Datasheet - https://www.ti.com/lit/ds/symlink/tps54233.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1678644409099&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftps54233

[9] Tech Demo Specifications - https://docs.google.com/document/d/e/2PACX-1vSTGN-MsnjkEHcNywmy30H1PFOFKHb0VqDk4epHT8vFfdkYmJU_G73ZmjhTbj4UtabMG7anF15-OSYx/pub

4.1.7 Revision Table

03/11/2023	Renee: Populated the section on the Smart hub hardware with content from block validation documents and made edits.
02/10/2023	Renee: Revised block validation document to included suggestions from Don, Rachel and Peer reviews
01/18/2023	Renee: Created and populated the block validation document

4.3 Back End Control App

4.3.1 Description

The Code block serves as a backend web application for the user control application. The program is an API written in Python with the Flask library; the program accepts HTTP GET and POST requests and MQTT Publish and Subscribe requests from client applications [1]. The user control frontend application will communicate with the program through HTTP requests, and the smart hub devices will be able to communicate with the program through MQTT requests.

4.3.2 Design

The Internet of Things smart home project provides user mobile control over home electronic appliances when attached to the project's smart plug and connected to the project's smart hub. To accomplish the home appliance control and management system, the code will retrieve HTTP requests from the frontend application client and perform any MQTT requests as necessary to the smart hub.

The HTTP requests are used for communication between the frontend application client and the backend server application. A HTTP GET request from the client will trigger the backend API to send the current statuses of all the connected devices (smart plugs). A HTTP POST request allows the frontend application client to perform commands onto a specified smart plug if possible.

The MQTT requests are used for communication between the backend server application and the smart hub. A MQTT Subscribe request allows the backend server application to retrieve the current device status of a specified and valid smart plug. A MQTT Publish request will be used for the backend server application to communicate any commands to a specified target smart plug if possible.

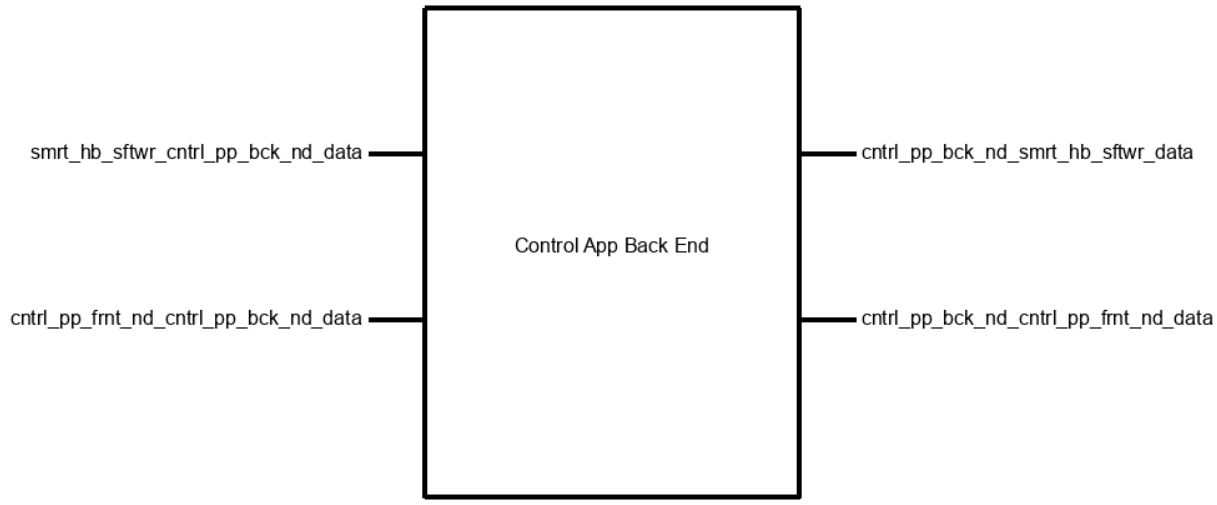

Figure 6: Back End Block Diagram

4.3.3 General Validation

For validating the Code block, functional validation will be used. The function of the code has the most important impact on the final system as a whole, hence function validation is crucial.

Functional Validation:

1. The system needs to be able to send data to the frontend application client based on the request call to the program API. The code design shown includes steps for handling HTTP GET requests. This step is in a loop, repeating each time a client performs a HTTP GET request to the backend API. If this process fails, a timeout error will be triggered and an error message will be sent back to the client.
2. The system needs to be able to retrieve data to the frontend application client based on the application request to the API. The code design shown includes steps for handling HTTP POST requests. This step is in a loop, repeating each time a client performs a HTTP POST request to the backend API. If this process fails, a timeout error will be triggered and an error message will be sent back to the client.
3. The system needs to be able to send data to smart hub clients based on frontend application requests to the API. The code design shown includes steps for handling HTTP POST requests and performing MQTT requests to the device specified in the request. This step is in a loop, repeating each time a client performs a HTTP GET request to the backend API.
4. The system needs to be able to connect and retrieve data from the smart hub clients routinely. The code design shown includes steps routinely checking the statuses of each smart plug device. This step is in a loop, repeating routinely and sending MQTT subscribe requests to all the known and valid smart plug devices

4.3.4 Interface Validation

The tables below include the interface property validation for this block. All interface properties have been addressed and the design meets or exceeds the properties.

Interface (cntrl_pp_frnt_nd_cntrl_pp_bck_nd_data) Property Validation for the Backend Code Block

Property	**Validation**
Protocol: HTTP	Python Flask API should be communicated through HTTP requests
Other: GET request	Program contains code that handles GET HTTP requests from clients.
Other: Code Size less than 4KB	Program size could be measured and seen.
Other: Time delay less than 1 second	Time out added for requests to ensure response is within 1 second

Table 9: Back end property validation table

Interface (cntrl_pp_bck_nd_cntrl_pp_frnt_nd_data) Property Validation for the Backend Code Block

Property	Validation
Protocol: HTTP	Python Flask API should be communicated through HTTP requests
Other: POST request	Program contains code that sends POST HTTP requests to clients.
Other: Code Size less than 4KB	Program size could be measured and seen.
Other: Time delay less than 1 second	Time out added for requests to ensure response is within 1 second

Table 10: Back end property validation table

Interface (cntrl_pp_bck_nd_smrt_hb_sftwr_data) Property Validation for the Backend Code Block

Property	Validation
Protocol: MQTT	Python Flask MQTT API should be able communicated through MQTT requests [1]
Other: Publish request	Program contains code that sends MQTT publish requests to clients.
Other: Code Size less than 4KB	Program size could be measured and seen.

Table 11: Back end property validation table

Interface (smrt_hb_sftwr_cntrl_pp_bck_nd_data) Property Validation for the Backend Code Block

Property	Validation
Protocol: HTTP	Python Flask MQTT API should be able communicated through MQTT requests [1]
Other: Subscribe request	Program contains code that handles MQTT subscribe requests from clients.
Other: Code Size less than 4KB	Program size could be measured and seen.

Table 12: Back end property validation table 4.3.5 Verification Process

For this code block, test code to resemble the frontend application software and a test MQTT client is required. The block could be verified through examining the data collected by the frontend application test software, test MQTT client, and the readings from the backend server application. Verification could be performed as followed:

Backend HTTP API verification:

- Tests interfaces cntrl_pp_frnt_nd_cntrl_pp_bck_nd_data HTTP GET and cntrl_pp_bck_nd_cntrl_pp_frnt_nd_data HTTP POST requests.

1. Start up the backend server application on localhost and port 5000.
2. Setup test scripts that perform requests calls to the backend server application to mimic frontend request calls.
    a. Perform a HTTP GET to the path http://localhost:5000/device/info
    b. Perform a HTTP POST to the path http://localhost:5000/device/command
3. Verify from the test scripts that both GET and POST requests are successful with status 200.
4. Verify from the backend server application that requests are retrieved from the test script.

- PASS: To pass this test:
    - The program must be able to send a data packet to the test frontend application client timely with no data loss.
        - Examine the data received from the test frontend application client matches the data sent from the backend server application.
        - Use time out for the API request to ensure data is received in under 1 second.
    - The program must be able to receive a data packet to the test frontend application client timely with no data loss.
        - Examine the data received from the backend server application matches the data sent from the test frontend application client.
        - Use time out for the API request to ensure data is received in under 1 seconds.
    - Program size less than 4KB.

Backend MQTT calls verification:

- Tests interfaces cntrl_pp_bck_nd_smrt_hb_sftwr_data MQTT publish and smrt_hb_sftwr_cntrl_pp_bck_nd_data MQTT subscribe requests.

1. Start up the backend server application on localhost and port 5000.
2. Setup test MQTT client scripts that perform requests calls to the backend server application to mimic smart hub client requests.
    a. Perform a MQTT Publish to the topic mqtt_pub
    b. Perform a MQTT Subscribe to the topic mqtt_sub

3. Verify from the test client that connection was successful and requests are sent and retrieved.
4. Verify from the backend server application that connection was successful and requests are sent and retrieved.

- PASS: To pass this test:
    - The program must be able to send a data packet to the test MQTT client with no data loss.
        - Examine the data received from the test MQTT client matches the data sent from the backend server application.
        - The program must be able to receive a data packet to the test MQTT client with no data loss.
        - Examine the data received from the backend server application matches the data sent from the test MQTT client.

4.3.6 References and File Links

[1] “Welcome to Flask — Flask Documentation (2.2.x),” flask.pallets projects.com. https://flask.palletsprojects.com/en/2.2.x/

[2] S. Lehmann, “Flask-MQTT: Flask extension for the MQTT protocol,” PyPI. https://pypi.org/project/Flask-MQTT/ (accessed Feb. 8, 2023).

4.3.7 Revision Table

03/12/2023	Jia Wei: Edited section to have latest information.
03/10/2023	Jia Wei: Populated the section on the Backend app software with content from the block validation document.

4.4 Smart Plug Software

4.4.1 Description

The Code block serves as a software application for the smart plug. The program is written in Arduino C to work within an esp microcontroller; the program uses MQTT subscribe to receive commands from the smart hub, and uses MQTT publish to send out current status.

4.4.2 Design

The Internet of Things smart home project provides user mobile control over home electronic appliances when attached to the project's smart plug and connected to the project’s smart hub. To accomplish the home appliance control and management system, the code will retrieve

commands from MQTT topics published by the smart hub, perform the command, and update its status in a MQTT topic.

MQTT is used for communication between the smart hub and the smart plug. The smart plug will constantly poll commands from the topic published by the smart hub, and perform the command it received. The smart plug will also constantly update its latest device by publishing that information to a specific MQTT topic.

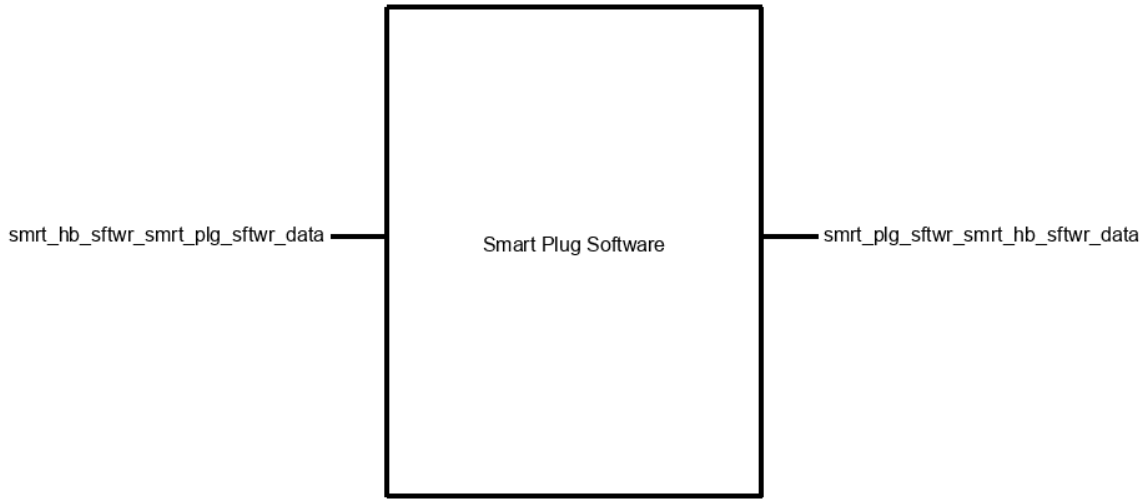

Figure 7: Smart Plug Software Block Diagram

4.4.3 General Validation

For validating the Code block, functional validation will be used. The function of the code has the most important impact on the final system as a whole, hence function validation is crucial.

Functional Validation:

1. The system needs to be able to send data to the smart hub client by publishing current status to a MQTT topic. This step is in a loop, repeating and updating current statuses at a 1Hz rate. If this process fails, the MQTT topic will stop receiving any messages from this system.
2. The system needs to be able to retrieve data from the smart hub client through subscribing to the MQTT topic posted by the smart hub. This step is in a loop, repeating and constantly polling for new commands by the client at a 1Hz rate. If this process fails, an error message will be sent back to the client.

4.4.4 Interface Validation

The tables below include the interface property validation for this block. All interface properties have been addressed and the design meets or exceeds the properties.

Interface (smrt_hb_sftwr_smrt_plg_sftwr_data Property Validation for the Smart Plug Code Block

Property	Validation
Protocol: MQTT	Arduino C PubSubClient library should communicate through MQTT [1].
Other: Subscribe	Program contains code that handles MQTT topic subscription.
Other: Time delay less than 1 second	Time out added for requests to ensure response is within 1 second

Table 13: Smart PLug Software property validation table

Interface (smrt_plg_sftwr_smrt_hb_sftwr_data) Property Validation for the Backend Code Block

Property	Validation
Protocol: MQTT	Arduino C PubSubClient library should communicate through MQTT [1].
Other: Publish	Program contains code that publishes to MQTT topics.
Other: Time delay less than 1 second	Time out added for requests to ensure response is within 1 second

4.4.5 Verification Process

For this code block, test code to resemble the smart hub software as a MQTT client is required. The block could be verified through examining the data collected by the smart hub test software, the readings from the smart plug application. Verification could be performed as followed:

Hub to Plug communication verification:

- Tests interface smrt_hb_sftwr_smrt_plg_sftwr_data

1. Start up the smart plug application.
2. Setup test scripts that perform MQTT publish to the device/mqtt_sub topic, to mimic smart hub command communication.
3. Verify from the smart plug that commands are retrieved from the test script.

- PASS: To pass this test:
    - The program must be able to receive a data packet to the test smart hub application client timely with no data loss.
        - Examine the data sent from the test application client matches the data received by the smart hub application.
        - Verify data retrieval rate is under 1 second.
    - Program size less than 4KB.

Plug to Hub communication verification:

- Tests interfaces smrt_plg_sftwr_smrt_hb_sftwr_data

1. Start up the smart plug application.
2. Setup test MQTT client scripts that subscribes to the MQTT topic device/mqtt_pub, to mimic smart hub plug-status retrieval.
3. Verify from the test client that connection was successful and messages could be retrieved from the MQTT topic.
5. Verify from the smart plug application that connection was successful and messages could be sent to the MQTT topic.

- PASS: To pass this test:
    - The program must be able to send a data packet to the test smart hub application client timely with no data loss.
        - Examine the data sent from the smart plug matches the data received by the test smart hub application.
        - Verify data retrieval rate is under 1 second.
    - Program size less than 4KB.

4.4.6 References and File Links

[1] “PubSubClient - Arduino Reference,” reference.arduino.cc. https://reference.arduino.cc/reference/en/libraries/pubsubclient/ (accessed Mar. 13, 2023).

4.4.7 Revision Table

03/12/2023	Jia Wei: Edited section to have latest information.
03/10/2023	Jia Wei: Populated the section on the smart plug software.

4.5 Front End Control App

4.1.1 Description

This block is the front-end/design of our companion web application. This application will be used to send commands to the main hub, and will communicate directly to the backend of our web application. From there, commands will be sent to the hub. This application will allow the user to the smart plug/switch, allowing for remote switching of outlet power. This application will also allow the user to send generic commands to other connected devices. This application will also show the status of current connected devices on the hub network. As stated earlier, all direct connections and communication will be done with the web applications backend, which will handle sending the proper commands and organization of incoming status messages from connected devices on the network.

4.1.2 Design

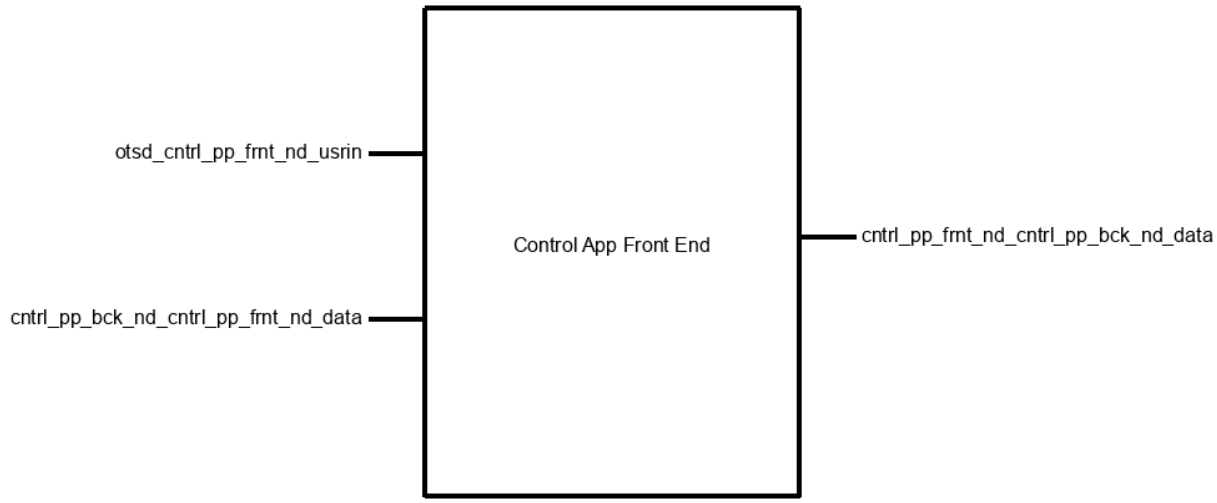

Figure 8: Front End Software Block Diagram

4.1.3 General Validation

The intent of the front end software is mainly to provide an user input interface and status message delivery system for the project as a whole. The general program flow was already outlined in the design overview. As mentioned before, the requirement that this front end application has to be full is that the system will provide the user a web application that 9 out of 10 users say is easy to use " while also meeting the interface definitions that will be expanded upon below. To do this, there are three large steps that need to be taken. One, a script
will be made that will exhaustively test all the input combinations for the user interface. Two, two separate devices will be brought to validation, one touch screen mobile device and the other is a laptop that interfaces through HID hardware such as a mouse. Three, set up a test terminal and edit the front end software to provide explicit status messages for every button combination so that validation can be done.
Later, a user survey can be conducted that evaluates the successfulness of this UI in relation to the system requirements. According to Browserstack, blind user testing with a post-test survey is
a commonly used way of testing the UI [2]. Specifically, a sample of 10 college students/peers can be selected to participate in a survey that has them try to use all the features of the device by
executing a series of commands to set-up a device, change its name, and then use all the features.
The instructions will not say how to do these things. Each survey can be timed to make and then
A small customer survey is given at the end [3].
To test the interfaces, we can utilize a simple .io script that executes html commands to test our web-app. This script is also a javascript file that raises expectations from the input button on the UI. In other words, it tests each button by raising an expected response and pressing a button

corresponding to a UI ID code. Finally, we can also just look at the output of the command struct, because that is the only thing that the back-end software looks at to distribute commands to the rest of the system.

4.1.4 Interface Validation

Interface Property **Interface Value** **Property Justification**

Otsd_cntrl_pp_frnt_nd_usrin

Other: Users must change the name of connected devices on the network via text boxes attributed to each device tile. At a bare minimum at least 4 buttons will be present in the UI. (device on on/off), device name change, save, device information.)	User input must be handled in this way from either a touch screen interacting with html buttons or via a HID device such as a mouse.	The web application must allow for users to change the name of their smart plugs so that it is easy to track in situations where there are multiple devices connected to the network.
Other: Front end will provide buttons to manipulate connected devices, such as switch plugs on and off and sending code to arduino units.	User input must be handled in this way from either a touch screen interacting with html buttons or via a HID device such as a mouse.	The web app must be able to manipulate devices and buttons provides an easy way to toggle device functions. This so that the basic use of the project is handled in the UI (which is a basic expectation).
Compatibility: HID / touch-screen	Since the web-app needs to be as portable as possible, allowing for the most common input type for web apps makes sense.	Update rates are set by devices, but allowing for these input devices seems like the bare minimum for a web app.

Cntrl_pp_frnt_nd_cntrl_pp_bck_nd_data

Other: *struct command* contains status of all buttons and the current array of saved devices names and associations. This struct populates an array of button status as integers, array of device names, and for arduinos, the text code input.	This struct will be a javascripts struct that can be sent or post requests to node.js servers and also can be read by the flask python backend since it's compatible with post commands as well.	Since our backend and front end are devicded, abstracting the communication between the two will make development and testing easier. This also allows for easier hosting on devices

		that are not the raspberry pi.
Other: *update_in_status():* This function will be called on button status changes to read all button ideas and fill the user input fields of the c*ommand struct.*	This routine will be a javascript function that the front end will run every 5 seconds or will run with changes to button status. This populates the struct.	Abstraction and portability are increased by this decision. Front end software has the aim of being multiplatform and therefore this method of abstracting all updates into parseable struct in and http.post request increases portability.
Update Rate: 0.2 Hz	The status function should update by itself so that sanity checks with the back end and connection hardware can be done.	According to [4] and looking at the study's optimal update rate 0.2 Hz should be a good balance for lower power device control modules that are not expected to be changed very often.

Cntrl_pp_bck_nd_cntrl_pp_frnt_nd_data

Other: *struct status* will contain updated devices array and update devices_names array and another array that will contain the status of each device (for plugs it will be on/off) and for arduinos it will be confirmation of code execution).	This struct will contain string data-type for device ids, integer data types for connection status, and the rest will be integers. This is because using these primitive limits our bandwidth usage.	Again, using this struct based approach to the front end allows for flexibility for the backend and allows the front end web server to run on any devices on the network as long as it can connect the hub.
Other: update_out_status(): This will post after the backend sends a message that the packet was delivered to the hub, then the backend will update the status struct to the latest value observed from the hub so that the front end and user can be informed. This will populate the html with correct values.	This layout makes the backend and front-end more spreadable and portable because it makes the interface between them as abstracted as possible.	Using the status update approach allows the backend to tell the front end when to update, which also reduces the bandwidth overhead and improves efficiency.

Update Rate: 0.2 Hz	The status function should update by itself so that sanity checks with the back end and connection hardware can be done.	According to [4] and looking at the study's optimal update rate 0.2 Hz should be a good balance for lower power device control modules that are not expected to be changed very often.

4.1.5 Verification Process

Verification Plan:

The verification plan goes as follows:

- First, I run the front end web server on a raspberry pi connected to an external monitor.
- Next, change the name of devices, observe terminal as update_status will print the command struct to the screen.
- Next, add a device and observe that the device array has increased in size with the default name added as the last element.
- Next, change the power state of smart plugs in UI and observe command structure.
- Next, Type code into the arduino sample box and observe code being passed to approve the device in the command structure.
- Next, select save and run code commands and observe command structure being filled.

This will test the command struct exhaust. Now to test, update _status function this is where the test script comes in.

- Load test script into web server as client and have it send status changes to front-end to emulate the back end.
- Observed the test script is able to change all changeable parameters and observed the changes are visible on the front-end.

If the UI updates, this implicitly proves that the status struct is also functioning as planned.

Finally, to test the user input and HID and touch screen compliance, the first part of the verification can be done using both a laptop with multiple keyboards and mice, and two mobile devices, an ipad and an android phone. This will test almost all types of devices expected to load and interact with the front end and display the web application.

4.1.6 References and File Links

References and Links:

[1] Node.js, “About,” Node.js. [Online]. Available: https://nodejs.org/en/about/. [Accessed: 20-Jan-2023].

[2] S. Bose, “Ui Testing: A detailed guide,” *BrowserStack*, 27-Dec-2022. [Online]. Available: https://www.browserstack.com/guide/ui-testing-guide. [Accessed: 11-Feb-2023].

[3] T. Hawkins, “How to unit test HTML and Vanilla JavaScript without a UI framework,” *DEV Community*, 23-Apr-2020. [Online]. Available: https://dev.to/thawkin3/how-to-unit-test-html-and-vanilla-javascript-without-a-ui-framework-4io. [Accessed: 11-Feb-2023].

[4] L. Hu, Z. Chen, Y. Dong, Y. Jia, L. Liang and M. Wang, "Status Update in IoT Networks: Age-of-Information Violation Probability and Optimal Update Rate," in *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11329-11344, 15 July 15, 2021, doi: 10.1109/JIOT.2021.3051722.

4.1.7 Revision Table

Revision Table:

1/28/2023	Initial draft created and finalized and added sections 1-7. -Matthew Gragg
2/10/23	Implemented draft feedback and created the final draft. -Matthew Gragg

4.6 Smart Hub Software

4.1.1 Description

The Smart hub will be a HTTPS socket layer/server, also using MQTT to schedule updates to ESP32's running MQTT scheduling software. The main point is to provide MQTT brokering to all connected devices.

4.1.2 Design

The control hub software's main duty is to provide MQTT brokering and surveillance. This can be done via a terminal.

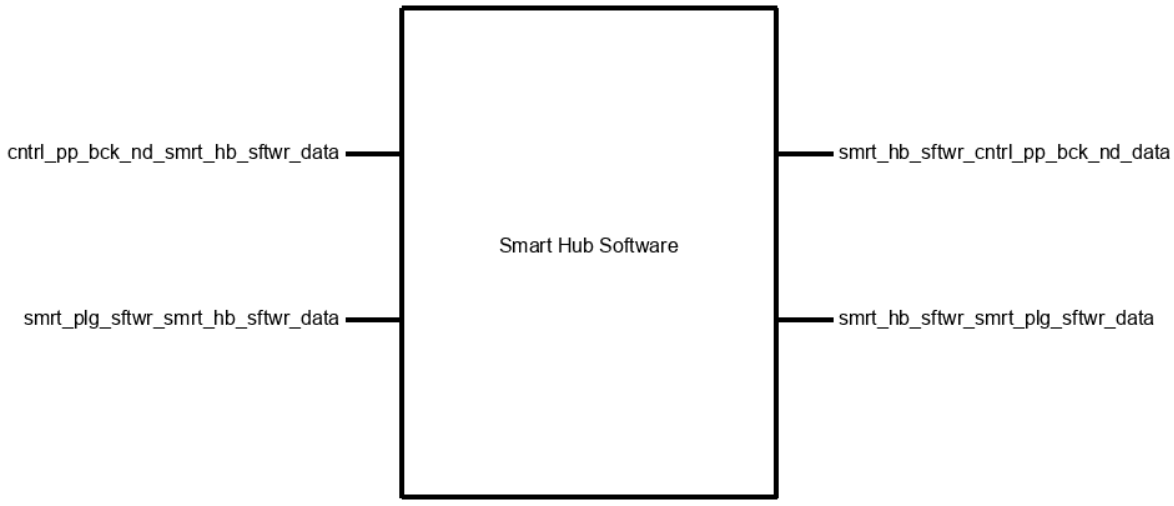

Figure 9: Smart Hub Software Block Diagram

4.1.3 General Validation

Validation of this block can be simply done via viewing the terminal both publisher and subscriber flows and seeing that MQTT messages and topics are updated.

4.1.4 Interface Validation

Interface Property	**Interface Value**	**Property Justification**
Otsd_cntrl_pp_frnt_nd_usrin		
Other: Users must change the name of connected devices on the network via text boxes attributed to each device tile. At a bare minimum at least 4 buttons will be present in the UI. (device on on/off), device name change, save, device information.)	User input must be handled in this way from either a touch screen interacting with html buttons or via a HID device such as a mouse.	The web application must allow for users to change the name of their smart plugs so that it is easy to track in situations where there are multiple devices connected to the network.

Other: Front end will provide buttons to manipulate connected devices, such as switch plugs on and off and sending code to arduino units.	User input must be handled in this way from either a touch screen interacting with html buttons or via a HID device such as a mouse.	The web app must be able to manipulate devices and buttons provides an easy way to toggle device functions. This so that the basic use of the project is handled in the UI (which is a basic expectation).
Compatibility: HID / touch-screen	Since the web-app needs to be as portable as possible, allowing for the most common input type for web apps makes sense.	Update rates are set by devices, but allowing for these input devices seems like the bare minimum for a web app.

Cntrl_pp_frnt_nd_cntrl_pp_bck_nd_data

Other: *struct command* contains status of all buttons and the current array of saved devices names and associations. This struct populates an array of button status as integers, array of device names, and for arduinos, the text code input.	This struct will be a javascripts struct that can be sent or post requests to node.js servers and also can be read by the flask python backend since it's compatible with post commands as well.	Since our backend and front end are devicded, abstracting the communication between the two will make development and testing easier. This also allows for easier hosting on devices that are not the raspberry pi.
Other: *update_in_status():* This function will be called on button status changes to read all button ideas and fill the user input fields of the c*ommand struct.*	This routine will be a javascript function that the front end will run every 5 seconds or will run with changes to button status. This populates the struct.	Abstraction and portability are increased by this decision. Front end software has the aim of being multiplatform and therefore this method of abstracting all updates into parseable struct in and http.post request increases portability.
Update Rate: 0.2 Hz	The status function should update by itself so that sanity checks with the back end and connection hardware can be done.	According to [4] and looking at the study's optimal update rate 0.2 Hz should be a good balance for lower power device control modules that are

		not expected to be changed very often.

Cntrl_pp_bck_nd_cntrl_pp_frnt_nd_data

Other: *struct status* will contain updated devices array and update devices_names array and another array that will contain the status of each device (for plugs it will be on/off) and for arduinos it will be confirmation of code execution).	This struct will contain string data-type for device ids, integer data types for connection status, and the rest will be integers. This is because using these primitive limits our bandwidth usage.	Again, using this struct based approach to the front end allows for flexibility for the backend and allows the front end web server to run on any devices on the network as long as it can connect the hub.
Other: update_out_status(): This will post after the backend sends a message that the packet was delivered to the hub, then the backend will update the status struct to the latest value observed from the hub so that the front end and user can be informed. This will populate the html with correct values.	This layout makes the backend and front-end more spreadable and portable because it makes the interface between them as abstracted as possible.	Using the status update approach allows the backend to tell the front end when to update, which also reduces the bandwidth overhead and improves efficiency.
Update Rate: 0.2 Hz	The status function should update by itself so that sanity checks with the back end and connection hardware can be done.	According to [4] and looking at the study's optimal update rate 0.2 Hz should be a good balance for lower power device control modules that are not expected to be changed very often.

4.1.5 Verification Process

- Observe Publisher and subscriber terminals.
- Check to see if all devices are posting.

4.1.6 References and File Links

[1] Node.js, “About,” Node.js. [Online]. Available: https://nodejs.org/en/about/. [Accessed: 20-Jan-2023].

[2] S. Bose, “Ui Testing: A detailed guide,” *BrowserStack*, 27-Dec-2022. [Online]. Available: https://www.browserstack.com/guide/ui-testing-guide. [Accessed: 11-Feb-2023].

[3] T. Hawkins, “How to unit test HTML and Vanilla JavaScript without a UI framework,” *DEV Community*, 23-Apr-2020. [Online]. Available: https://dev.to/thawkin3/how-to-unit-test-html-and-vanilla-javascript-without-a-ui-framework-4io. [Accessed: 11-Feb-2023].

[4] L. Hu, Z. Chen, Y. Dong, Y. Jia, L. Liang and M. Wang, "Status Update in IoT Networks: Age-of-Information Violation Probability and Optimal Update Rate," in *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11329-11344, 15 July 15, 2021, doi: 10.1109/JIOT.2021.3051722.

4.1.7 Revision Table

Revision	Description
1	Initial Revision
2	Final Revision

4.7 Smart Plug Enclosure

Block Champion: Kristina Mason
Date: Mar-08-2023

4.1.1 Description

An enclosure for the smart plug, where the circuit and microcontroller for the Smart Plug are in it. This enclosure will be able to withstand the shock of being dropped from chest height. The enclosure will also not conduct electricity to the outer world, ensuring the safety of those who will use this device.

4.1.2 Design

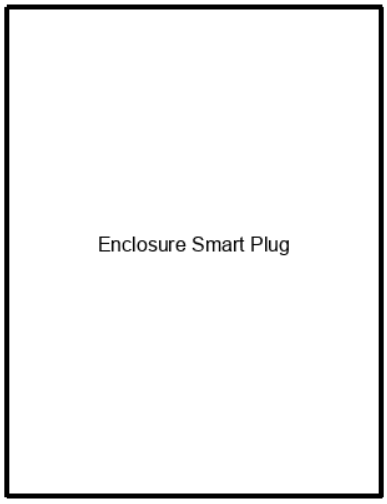

Figure 10: Smart Plug Enclosure Block Diagram

4.1.3 General Validation

This system will protect both the users and the internal hardware of the system from outside forces, most commonly seen in households.

4.1.4 Interface Validation

This block has no interfaces, since it doesn't interact with other parts of the whole system, but instead protects a single block from the outside world.

4.1.5 Verification Process

To verify that the enclosure does its job, the enclosure will need to be examined to make sure there are no exposed wires. Kicking and hitting the enclosure will also test to see if it can protect the circuit from bumps and keep the circuit functioning despite the disturbances. Touching the enclosure should not shock the user or harm them in any way.

4.1.6 References and File Links

4.1.7 Revision Table

Revision	Description
1	Initial Revision

4.8 Smart Plug Hardware

Block Champion: Kristina Mason
Date: Mar-08-2023

1. Description

The hardware and circuitry required to create a smart plug in which it will act as a wirelessly activated relay activating an incandescent or LED light bulb usually placed into lamps. Inside of the Smart Plug will be an ESP32 microcontroller along with a transformer that will turn the voltage output from the microcontroller into a voltage high enough to power lightbulbs and other AC powered devices.

2. Design

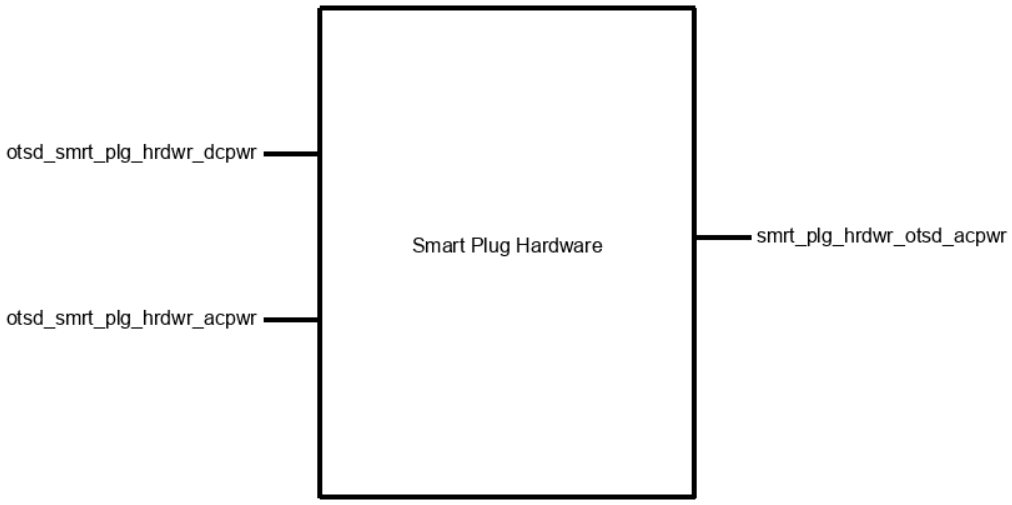

Figure 11: Smart Plug Hardware Block Diagram

Schematic:

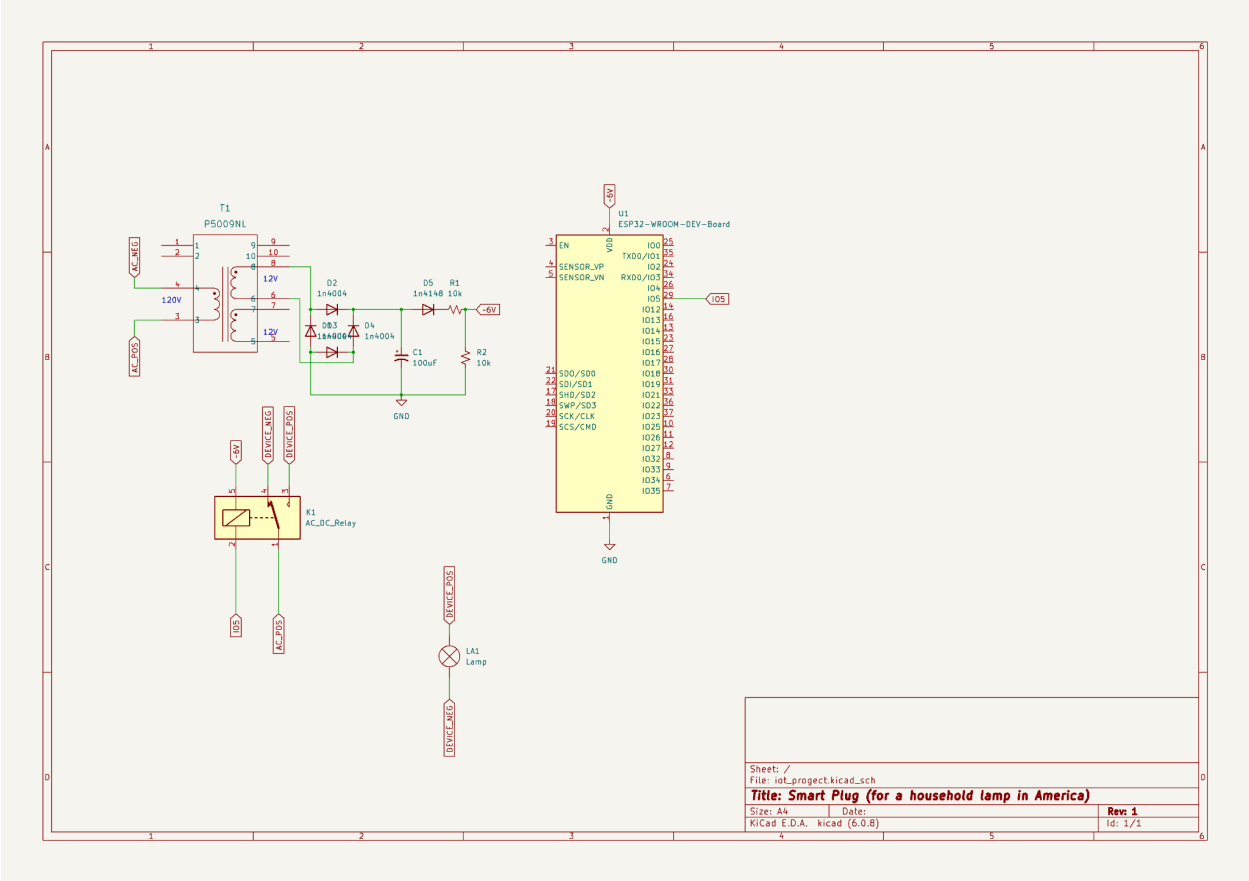

Figure 12: Smart Plug Hardware schematic

3. General Validation

The sub-system design meets the requirements of this block because it will be able to be powered by mains voltage and direct that power to an external device. This sub-system will also be able to power any DC powered components within this block, like the ESP32, by converting the AC mains into ~5VDC.

4. Interface Validation

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

otsd_smrt_plg_hrdwr_dcpwr : Input

Inominal: .5ADC	ESP32 needs at least .5A in order to function.	The output current of the transformer is 2A, well over the

		bare minimum to run the ESP32
Ipeak: 1ADC	It is unnecessary that any part of the design would need more than 1A of current.	The output current of the transformer is 2A.
Vmax: 12VDC	The voltage regulator can handle a max of 15V, but going to the absolute max voltage rating would break it, so the more common 12V will be used.	The transformer is able to output 12VAC.
Vmin: 5VDC	The minimum voltage the relay can use is 5V, any lower and the relay wouldn't be able to function.	The voltage divider in the circuit reduces the 12V to approx. 6V, so this will work.

otsd_smrt_plg_hrdwr_acpwr : Input

Inominal: 5A	The transformer doesn't need a minimum current to function and the lightbulbs only need a little current in order to turn on without burning out.	
Ipeak: 10A	The relay's contact rating was 10A/250VAC, so the max current the relay can take and give is 10A.	
Vnominal: 120VAC	The voltage given from mains voltage in most American homes is 120VAC.	

smrt_plg_hrdwr_otsd_acpwr : Output

Inominal: 0.08 Amps AC (With Nominal power being 10W at 120VAC)	The lightbulbs don't need that much current to function and the power ratings of most LED light bulbs is 6-10W.	

Ipeak: 0.17 Amps AC (Max power being 20W at 120VAC)	The current taken by incandescent bulbs is around 0.17 Amps AC, using 20W at most.	
Vnominal: 120VAC to power lamps that run on AC power	The voltage taken from mains will be passed along to any device connected to the smart plug, 120VAC.	

5. Verification Plan

In order to verify:

1. Send signal to microcontroller from computer
2. Check if lightbulb/device turns on and relay activates

6. References and File Links

7. Revision Table

Revision	Description
1	Initial Revision

5. System Verification Evidence

5.1 Universal Constraints

5.1.1. The system may not include a breadboard

All the parts of this system are built on a protoboard or a PCB. A breadboard was used for prototyping and the final system was made later. Shown below are the Smart Hub and smart plug circuits that do not include a breadboard

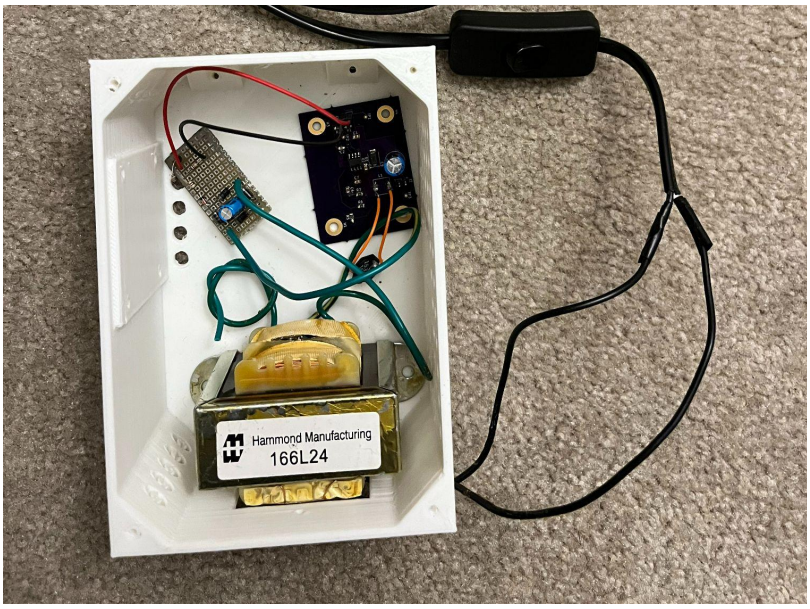

Figure 13: Smart Hub Circuit

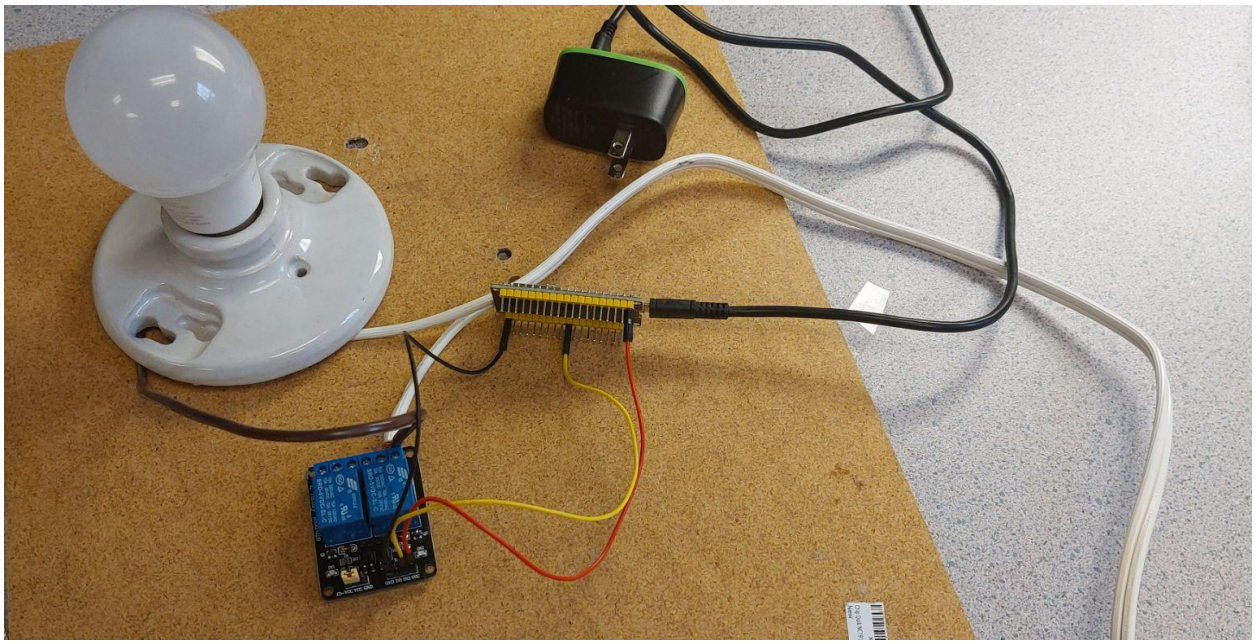

Figure 14: Smart Plug Circuit

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

The PCB that the group owns has at least 30 non-connector surface mount pads that are used.

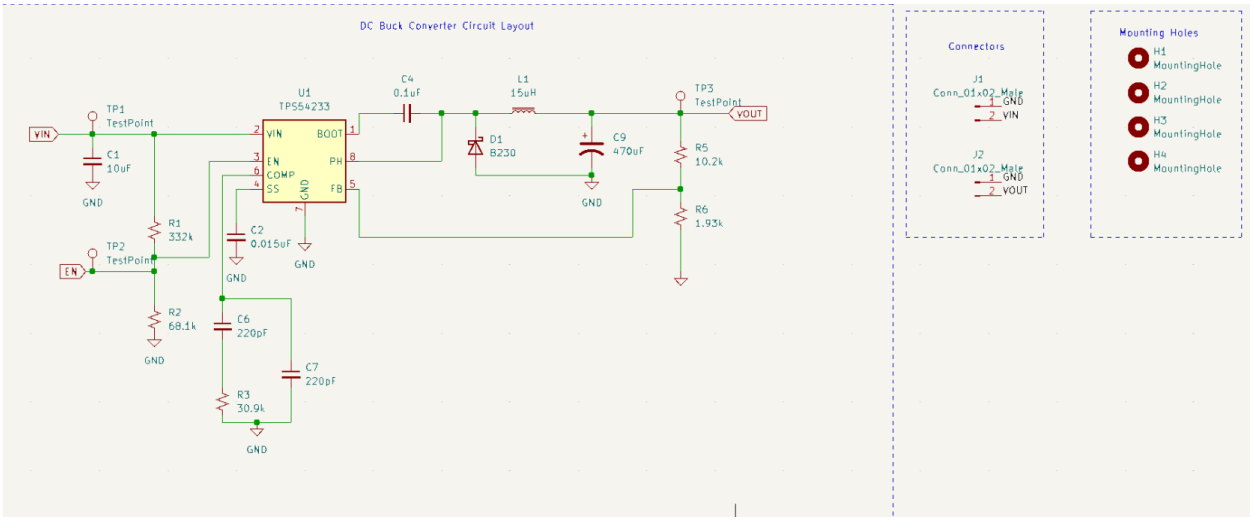

Figure 15: Buck Converter Schematic.

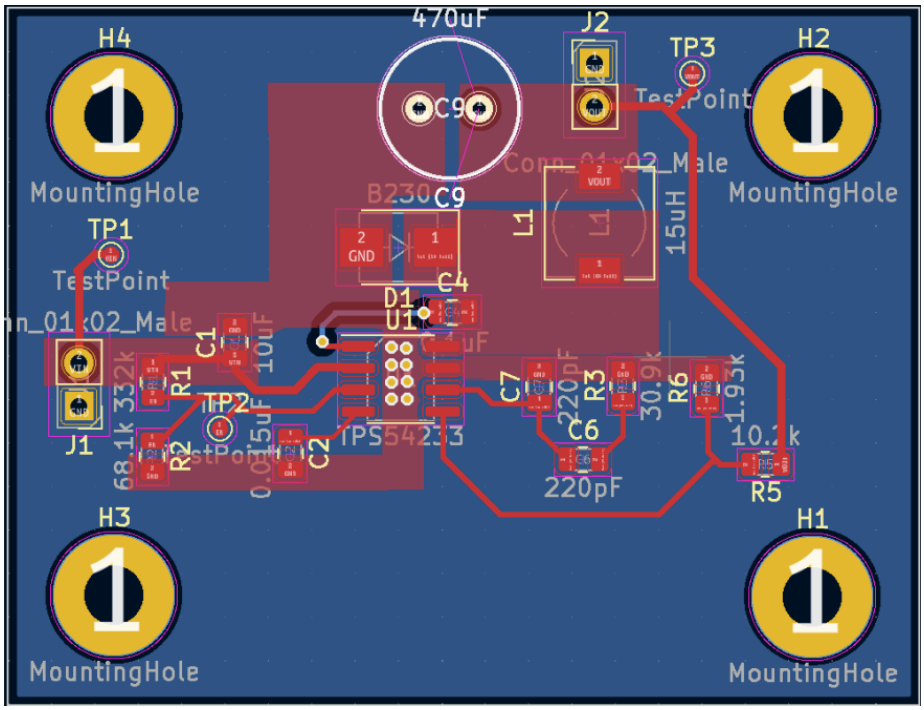

Figure 16: Buck Converter PCB

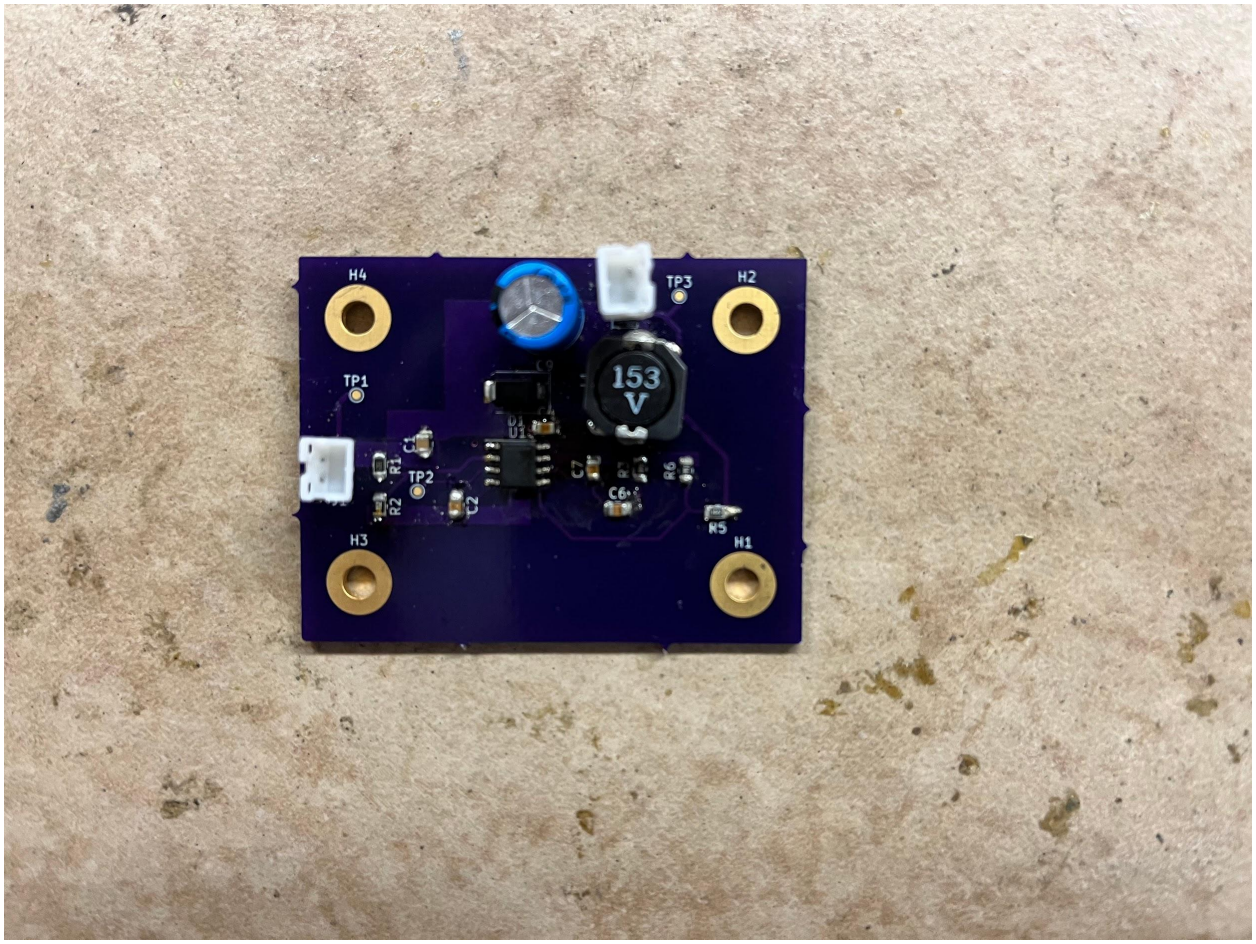

Figure 17: Fully assembled Buck Converter

5.1.3. All connections to PCBs must use connectors.

The Buck Converter has a wire connection to the inductor because the footprints chosen were too small for the part that was bought.

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

Due to inefficiencies in the equipment used, the wall power recorded is not able to be used in the efficiency calculations. However, upon calculation on the primary side of the transformer and after the rectifier circuit, the efficiency meets this requirement.

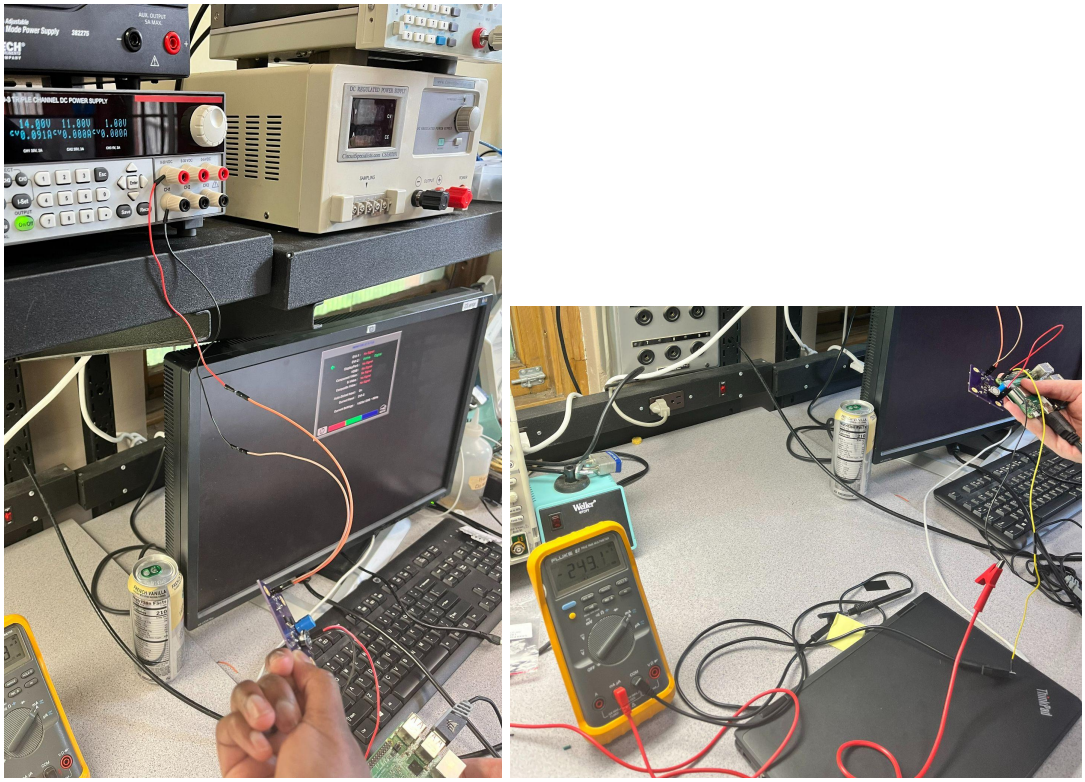

Figure 18: Power Verification Measurements

This measurement shows an input voltage supply of 14V at 0.09A, Pin equals 1.26 There is an output current of 0.243A, and according to the Raspberry Pi datasheet, a voltage draw of 5V. The efficiency is therefore 96%

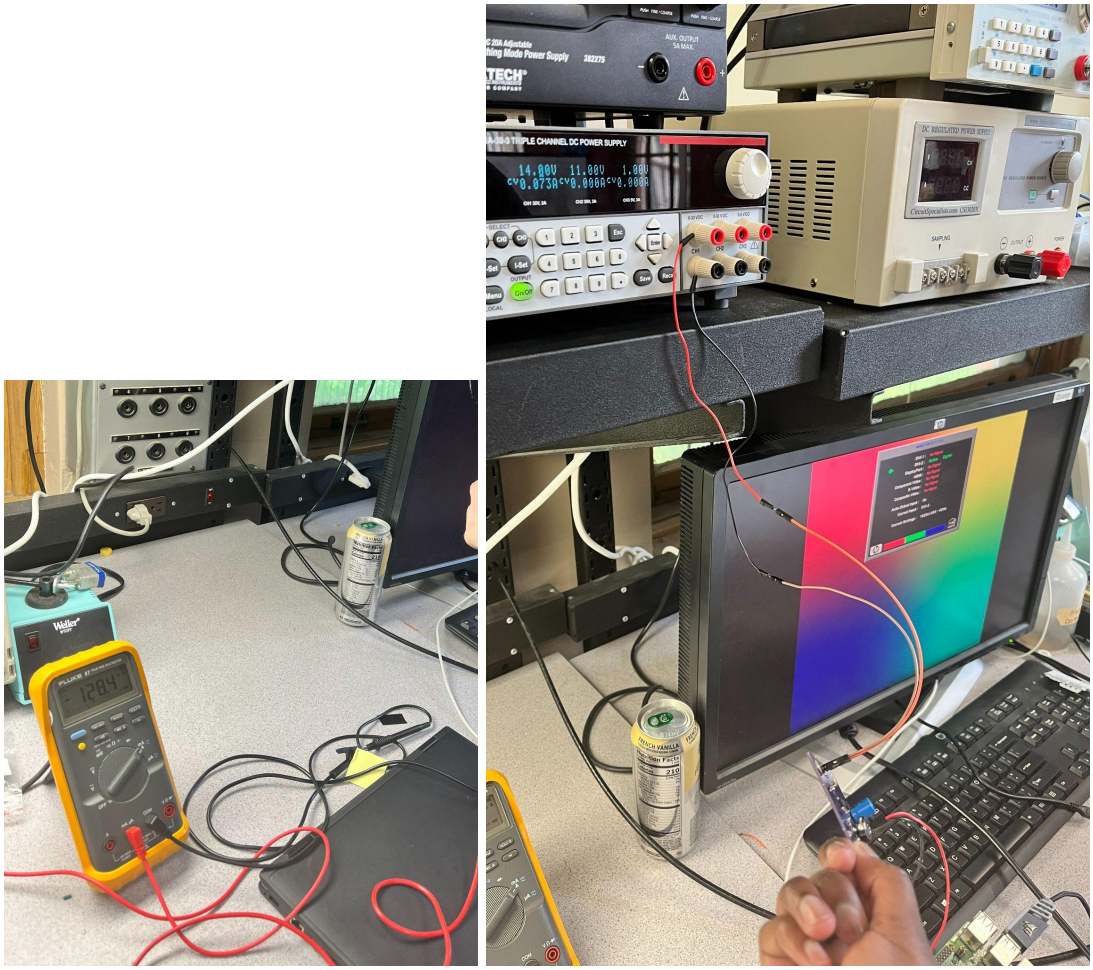

Figure 19: Power Verification Measurements

The calculation above shows an input of 14V at 0.07A, 0.98A. And an output of 0.128A and according to the Raspberry Pi datasheet, the voltage draw is 5V. Therefore the Pout is 0.64W. The efficiency is therefore 65%

These two calculations were taken at different stages of testing, and upon averaging the results, the efficiency is 80.5%

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

At least half of the parts in our current system are built modules.

Proof: Smart Hub and Smart Plug hardware.

Link to spreadsheet:

https://docs.google.com/spreadsheets/d/e/2PACX-1vTha0chZ_y6ila4ewjeU-JxDi3eggg3RboFxRcfD6X3P5fijIPPiDtw1FS05m-Ce7u8DhlV6aoTS_mD/pubhtml

A	B	C	D	E
Smart Hub				
Component	**Built/Bought**			
Transformer	Bought	-1		
Rectifier Circuit	Built	1		
Voltage regulator	Built	1		
Buck Converter	Built	1		
Raspberry Pi	Bought	-1		
Smart Plug				
Light bulb	Bought	-1		
Rectifier circuit	Built	1		
Voltage regulator	Built	1		
ESP32	Bought	-1		
	Total:	1	Positive sum = a larger % of built modules	

5.2 Requirements

5.2.1. Requirement Short Name: Communication between 3 nodes(Verified on 5/8/2023)

5.2.1.1. Project Partner Requirement: Communication between 3 devices must be present

5.2.1.2. Engineering Requirement: The system will have at least 3 devices that will communicate with each other in order to cause an event to occur.

5.2.1.3. Verification Process:

1. Enter a device number and the word “ON” after it via the user interface
2. Watch as the bulb goes on (due to connection to the ESP32)
3. The computer screen connected to the Raspberry Pi would show a corresponding “ON” and “OFF” message corresponding to the message sent to the ESP32.
4. This shows connection between the laptop, the ESP32 and the Raspberry Pi

5.2.1.4. Testing Evidence:

Link to video:
https://drive.google.com/file/d/1DwL_NnmIjqLLSl2eUJljlYrAkOSsZXSF/view?usp=share_link

5.2.2:Requirement Short Name: Expandable IOT Device Documentation(Verified on 5/8/2023)

5.2.2.1. Project Partner Requirement: The user will be provided with complete online documentation

5.2.2.2. Engineering Requirement: The system will have online documentation that 9 out of 10 university students after reviewing say is "complete and easy to understand".

5.2.2.3. Verification Process:

1. Take a look at the survey results
2. Note that 9 out of 10 people approved the documentation

5.2.2.4. Testing Evidence:

Link to documentation:
https://docs.google.com/document/d/1ZECVT21oMQCCYkDWFhx_fBS1GbHDvlJmFqapXlvT2Hc/edit

Figure 20:Survey Results

Survey Results:

Name	Email	Is the provided documentation complete and easy to understand?
Wei Yu Tang	tangwe@oregonstate.edu	Yes
Adam Farhat	farhata@oregonstate.edu	Yes

Michael Chen	chenmich@oregonstate.edu	Yes
Patrick Liang	liangpa@oregonstate.edu	Yes
Adam Grzelewski	agrzelewa@oregonstate.edu	Yes
Benjamin	gunadib@oregonstate.edu	Yes
Jessica Lee	jessicalee0985@gmail.com	Yes
Chelsey Chiu	gi25ni@gmail.com	Yes
Paul	pauldania11@gmail.com	Yes
Ting-hsuan Chen	chenting@oregonstate.edu	Yes
Preston Hang	hangp@oregonstate.edu	Yes
Madalyn Gragg	graggma@oregonstate.edu	Yes
Parn	tiangdab@oregonstate.edu	Yes
Nicole Aimba	aimban@oregonstate.edu	Yes
Lyon Kee	keel@oregonstate.edu	Yes
Yu Wei Koh	kohyu@oregonstate.edu	Yes

5.2.3: Requirement Short Name:Power Input(Verified on 3/14/2023)

5.2.3.1.Project Partner Requirement: Product should be able to be powered from an external source

5.2.3.2.Engineering Requirement: The system hub sub-system will be powered by 120VAC.

5.2.3.3.Verification Process:

1. Connect a DMM to the output of the Smart hub/Smart Plug circuit
2. Set the DMM to read DC voltage
3. Plug in the smart hub/smart plug circuit into the wall outlet
4. Turn on the switch if relevant
5. Read the voltage on the DMM, demonstrating that the circuit is being powered by the wall.

5.2.3.4.Testing Evidence: System verification demonstration
Link to video:
https://drive.google.com/file/d/1hwDvYTrY3sru6wUC9r6RUEdJhv5NEDZN/view?usp=share_link

5.2.4: Requirement Short Name: Project Budget(Verified on 3/14/2023)
5.2.4.1.Project Partner Requirement: Building this project must not exceed $300
5.2.4.2.Engineering Requirement: Find components and materials that, when the costs are totaled, don't exceed $300 in order to make this IOT device
5.2.4.3.Verification Process: (Verified on 3/15/2023)
1. Reference the Combined IOT BOM which entails the Smart Plug BOM and the Smart Hub BOM.

5.2.4.4.Testing Evidence: Bill of Materials presented to TA's during system verification
Link to BOM: Combined IOT BOM

5.2.5: Requirement Short Name: Transmission Distance
5.2.5.1. Project Partner Requirement: The system should provide a signal strong enough to be detected by conventional devices.
5.2.5.2. Engineering Requirement: The system will be able to provide -67 dB of signal strength at 200 centimeters.
5.2.5.3. Verification Process:
1. Place the smart hub 200 cm away from the smart plug.
2. Place a tape measure in between the two smart devices.
3. Send a message between the two devices to verify whether transmission works.

5.2.5.4. Testing Evidence:
Link to video:
https://drive.google.com/file/d/1-O3d5DSO6-p5O8bCaZkMKeFu61gY5cGK/view?usp=share_link

5.2.6: Requirement Short Name: User Interface(Verified on 5/8/2023)
5.2.6.1. Project Partner Requirement: Users should be able to control the devices easily from an application
5.2.6.2. Engineering Requirement: The system will provide the user a web application that 9 out of 10 users say is easy to use.
5.2.6.3. Verification Process:
1. Enter a device number and the word "ON" after it via the user interface
2. Watch as the bulb goes on (due to connection via the laptop i.e. user interface, and the ESP32)
3. The computer screen connected to the Raspberry Pi would show a corresponding "ON" and "OFF" message corresponding to the message sent to the ESP32.
4. One can also turn off a device via the user interface using the power button.
5. Users who used our software application would then fill in our survey.

5.2.6.4. Testing Evidence: Image of UI below

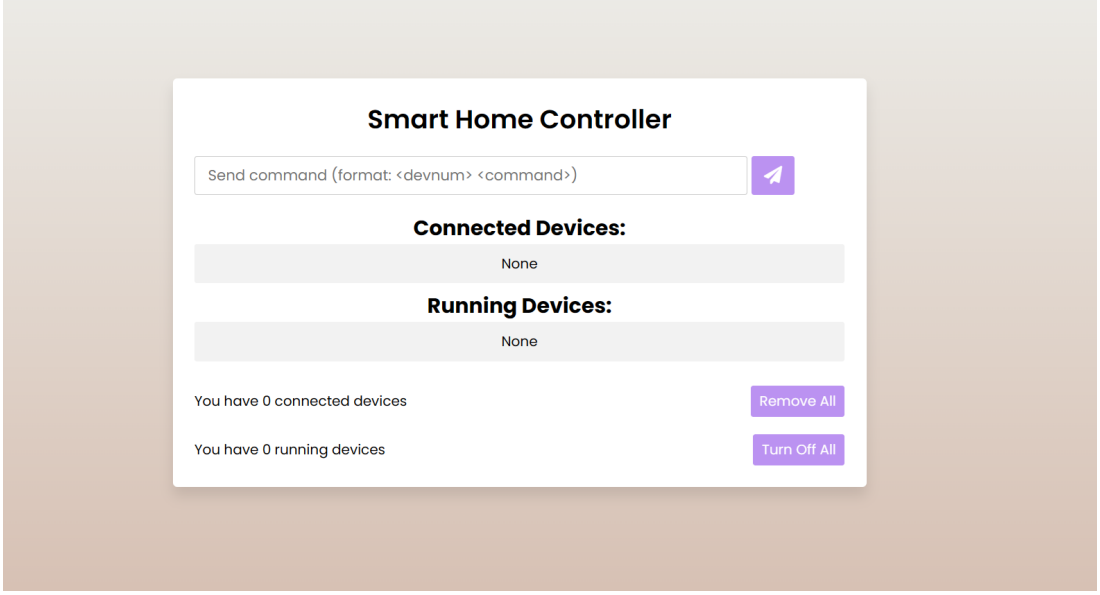

Figure 21:User Interface

Survey Results:

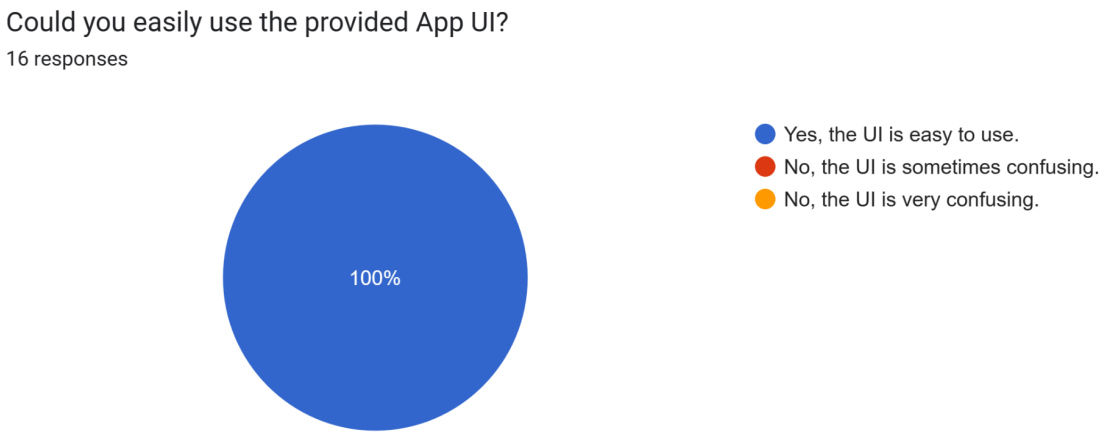

Figure 22: Survey Results

Name	Email	Is the App UI clean and easy to understand?	Could you easily use the provided App UI?	Is the provided documentation complete and easy to understand?
Wei Yu Tang	tangwe@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes

Adam Farhat	farhata@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Michael Chen	chenmich@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Patrick Liang	liangpa@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Adam Grzelewski	agrzelewa@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Benjamin	gunadib@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Jessica Lee	jessicalee0985@gmail.com	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Chelsey Chiu	gi25ni@gmail.com	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Paul	pauldania11@gmail.com	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Ting-hsuan Chen	chenting@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Preston Hang	hangp@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Madalyn Gragg	graggma@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Parn	tiangdab@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Nicole Aimba	aimban@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Lyon Kee	keel@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes
Yu Wei Koh	kohyu@oregonstate.edu	Yes, it's clean and easy to understand.	Yes, the UI is easy to use.	Yes

5.2.7: Requirement Short Name: Wireless Communication Connection(Verified on 5/8/2023)

5.2.7.1. Project Partner Requirement: The device must be able to communicate with other devices via a network

5.2.7.2. Engineering Requirement: The subsystems will communicate with each other over a wireless network

5.2.7.3. Verification Process:

1. Enter a device number and the word “ON” after it via the user interface
2. Watch as the bulb goes on (due to connection to the ESP32)

3. The computer screen connected to the Raspberry Pi would show a corresponding “ON” and “OFF” message corresponding to the message sent to the ESP32.
4. This shows connection between the laptop, the ESP32 and the Raspberry Pi

5.2.7.4. Testing Evidence:
Link to video:
https://drive.google.com/file/d/1DwL_NnmljqLLSl2eUJljlYrAkOSsZXSF/view?usp=share_link

5.2.8: Requirement Short Name:Wireless communication speed

5.2.8.1.Project Partner Requirement: Wireless Communication between devices should be fast

5.2.8.2. Engineering Requirement: The system will perform communication wirelessly under 1 second

5.2.8.3. Verification Process:

1. Enter a device number and the word “ON” after it via the user interface
2. Watch as the bulb goes on (due to connection to the ESP32)
3. The computer screen connected to the Raspberry Pi would show a corresponding “ON” and “OFF” message corresponding to the message sent to the ESP32.
4. This shows connection between the laptop, the ESP32 and the Raspberry Pi

5.2.8.4. Testing Evidence:
Link to video:
https://drive.google.com/file/d/1DwL_NnmljqLLSl2eUJljlYrAkOSsZXSF/view?usp=share_link

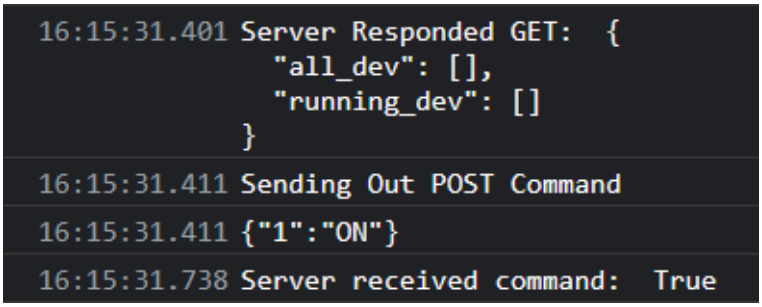

Figure 23: Client Response Timing Screenshot

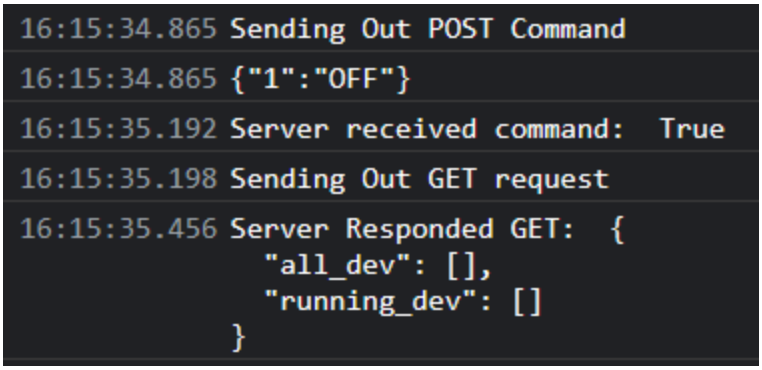

Figure 24: Client Response Timing Screenshot 2

5.3 References and file links

a. References

[1] Project Document guide - https://docs.google.com/document/d/e/2PACX-1vSTGN-MsnjkEHcNywmy30H1PFOFKHb0VqDk4epHT8vFfdkYmJU_G73ZmjhTbj4UtabMG7anF15-OSYx/pub

b. File links

[2] Transformer datasheet - https://eecs.engineering.oregonstate.edu/education/inventory_datasheets/P463-1419958697.pdf

[3] ESP32 datasheet - https://cdn-shop.adafruit.com/product-files/3269/esp32_datasheet_en_0.pdf

[4] Smart Hub BOM - https://docs.google.com/spreadsheets/d/e/2PACX-1vRmMGISnWZnRh2jvBIEkxuExQeuG2Lvhg9NUhSbvfvBiEXSQXLMVN1AIk6uAPoc8znSqotDR2Q_Tfml/pubhtml

[5]Smart Plug BOM - https://docs.google.com/spreadsheets/d/e/2PACX-1vSTrKNzgEyj0m-YLY5LMmFDNNR6L58hdYarM14oBZX3jF3bd2psrr6zgOo5oze3LHTLGqSYXlqeZIF6/pubhtml

[6] Combined IOT BOM - https://docs.google.com/spreadsheets/d/e/2PACX-1vRTwxklAcsG0oeJi37BrhX_6dcS2TFGkNxOdjKbL1WlMvmAzynEdbZSIDwiz31ztSmFrqtBzUC18eXN/pubhtml

5.4 Revision Table

Revision	Description
3/13/2023	Team: Entered the requirements and wrote about some verification processes.

5/5/2023	Team: Updated Engineering requirements

6. Project Closing

6.1 Future Recommendations

6.1.1 Technical Recommendations

- If possible, get a smaller transformer to shrink the size of the project. We are currently using the “Hammond Manufacturing” transformer which is 93mm by 62mm, and is quite heavy. However, we have explored other transformer options such as the “P5009NL Pulse Transformer” [1], but they proved faulty.
- Once one block is somewhat completed, try to integrate/test it out as soon as possible with other blocks to make changes in the process as early as possible. This is because it prevents the team from progressing too much with a faulty prototype/block
- Make sure everything is wired correctly when dealing with mains power, because mains power is rated at 50/60Hz at 120V, can kill, and has a high current/voltage which can damage one's circuit.
- Be aware of explosions, fire, electrocutions and other calamities. Have an emergency plan in place when testing. This is because testing with mains power can be dangerous and result in fires if done incorrectly/without caution.

Links:

[1] *Electrical Engineering and computer science*. TekBots. (n.d.). Retrieved April 28, 2023, from https://eecs.engineering.oregonstate.edu/education/inventory_datasheets/P463-1419958697.pdf

6.1.2 Global Recommendations

- Lead solder is considered hazardous, make sure to dispose of it accordingly. We would suggest purchasing lead-free solder so that you do not have to dispose of the whole circuit later or inhale hazardous fumes while soldering [1].
- Working with Mains voltage is hazardous, make sure there are no exposed wires coming from Mains power. Make sure to read the datasheets of modules to see if they are rated for mains power, and what voltage and current they can handle. Do not touch any exposed wires while connected to mains power and keep one hand away from the circuit to avoid creating a closed loop and getting electrocuted.

Links:

[1] Environmental Protection Agency. (n.d.). *Questions about the disposal of lead-contaminated items | hazardous waste treatment, storage & disposal*. EPA. Retrieved April 28, 2023, from https://archive.epa.gov/epawaste/hazard/web/html/faq-2.html

6.1.3 Teamwork Recommendations

- Meet as often as you can especially since during Senior year people have busy schedules and it is harder to stay on top of things. Make sure each meeting has an agenda and each member is assigned a role in order to ensure that the meeting is productive. Take notes to have a reference and look back on past designs.
- Distribute the blocks in a way that reduces team dependency. This enables the progress of one member to not be deterred by the progress of another while ensuring combined progress of the team. Make sure to discuss the full system functionality, and ensure that each member understands what is expected of them and the deadlines as well.
- Make sure that the team is in sync with the “technical terms/software” that the team is using.i.e. Versions of software. For example, make sure that two members working on the software side of the project, i.e. frontend and backend, use languages and softwares that are compatible with each other

Links:

[1] ECE Senior Design 2022-2023. (n.d.). Retrieved April 28, 2023, from https://eecs.engineering.oregonstate.edu/capstone/ece/student/index.php

6.2 Project Artifact Summary with Links

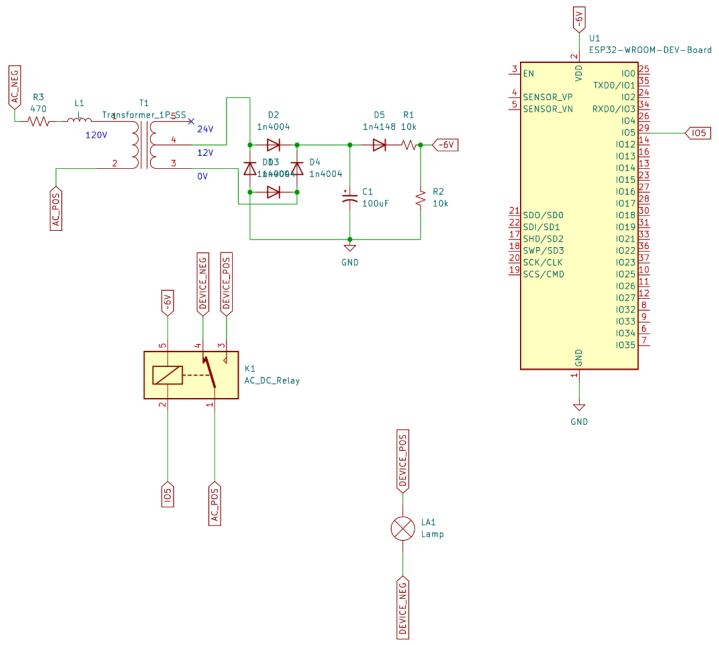

Figure 25: Smart Plug Schematic

Link to Schematic file:
https://drive.google.com/file/d/15A1mdkph68LrdeMObT__GDXUUdBiTJg_/view?usp=share_link

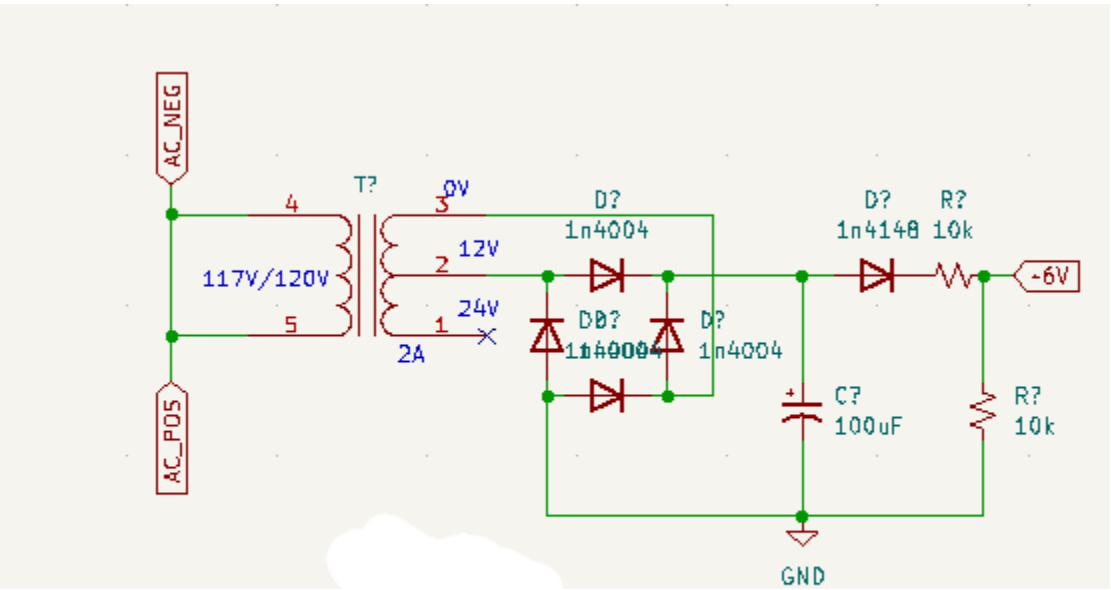

Figure 26: Rectifier Circuit Schematic

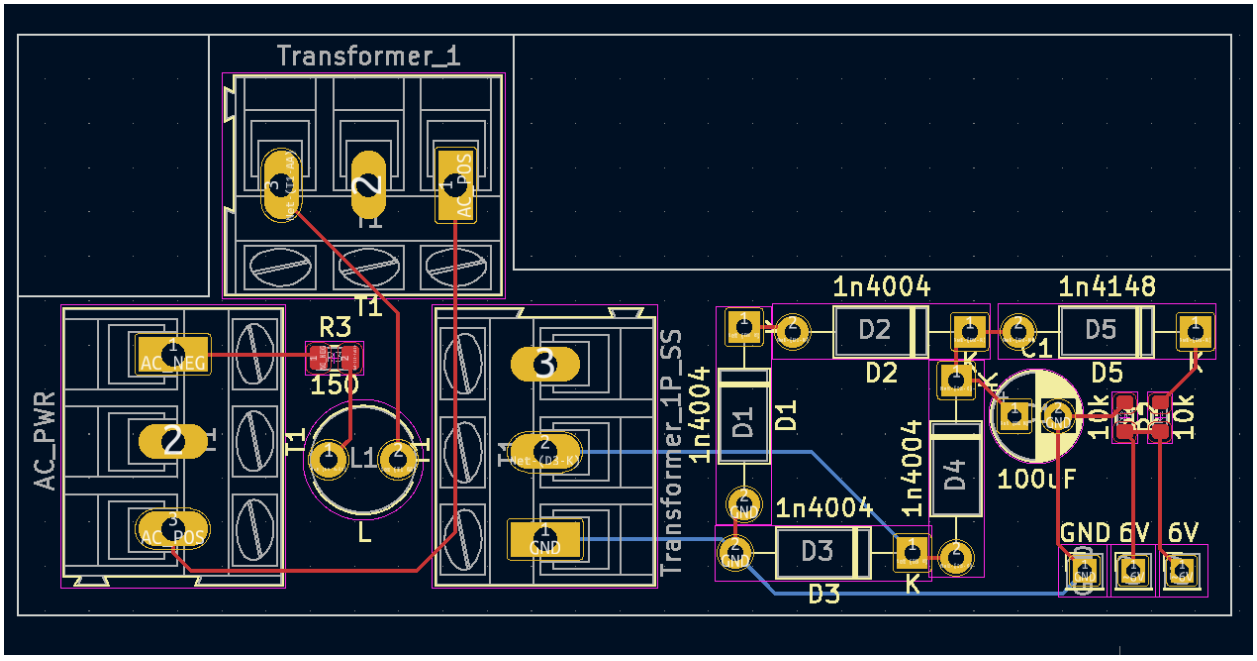

Figure 27: Smart Plug PCB Design

Link to PCB file:
https://drive.google.com/file/d/1TrKqyJlBJ1DuBQQcSjsAqDtCEqKl2uX1/view?usp=share_link

Smart Plug BOM:
https://drive.google.com/file/d/1JrmuxklPdQslJT-PrPba3js4_QwtwNht/view?usp=sharing

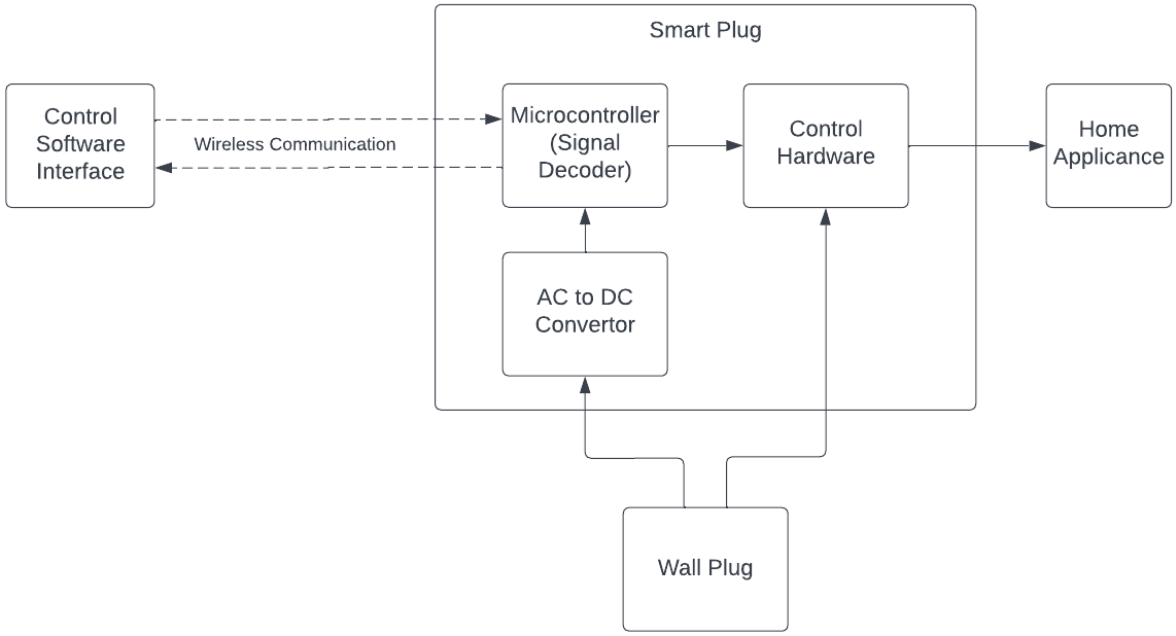

Figure 28: HIgh Level Hardware Diagram

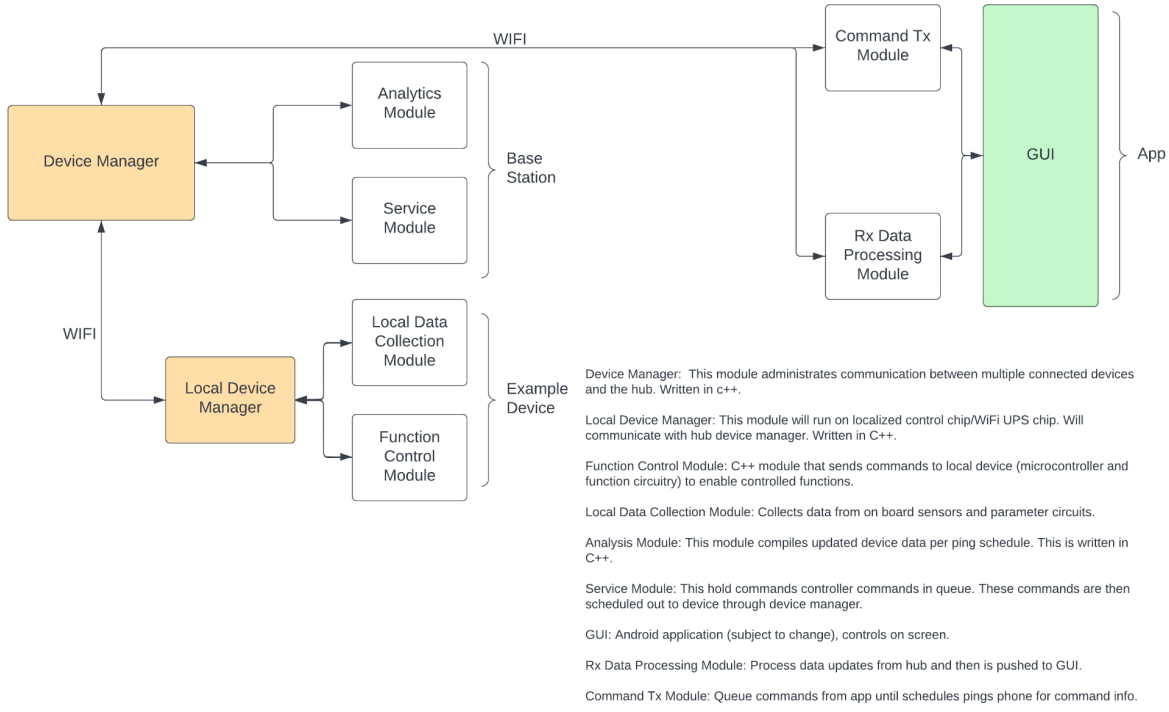

Figure 29: High Level Software

Project Github Link: https://github.com/Mattgrg/Sr.-Project

Code Breakdown:

user_app : Backend and API code.

User_app_frontend: User app html, styling, and webserver.

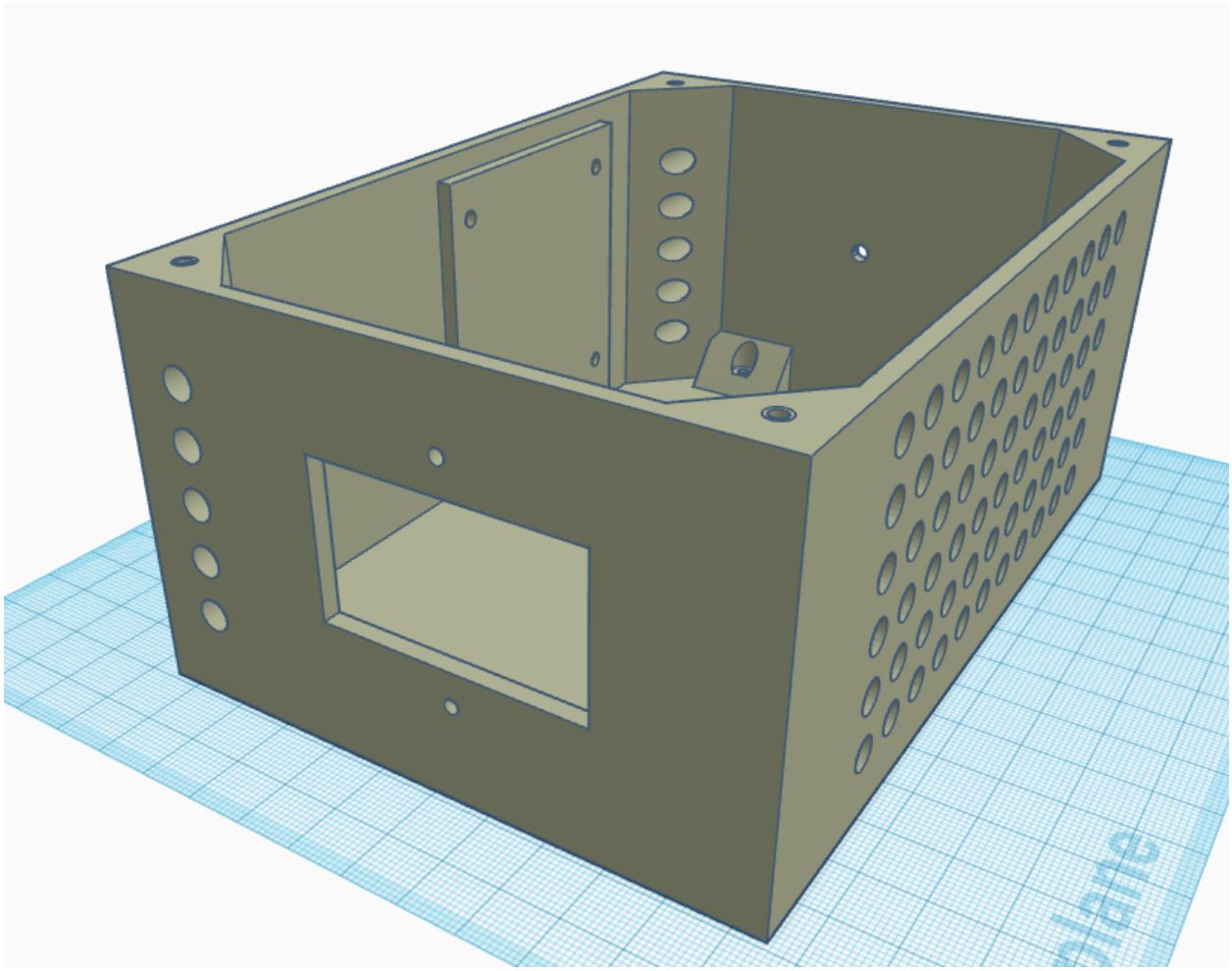

Figure 30: Smart Hub Enclosure
https://drive.google.com/file/d/1d3GbehmJC5RpVB5hYlpk8KMhre2BrHHy/view?usp=share_link

6.3 Presentation Materials

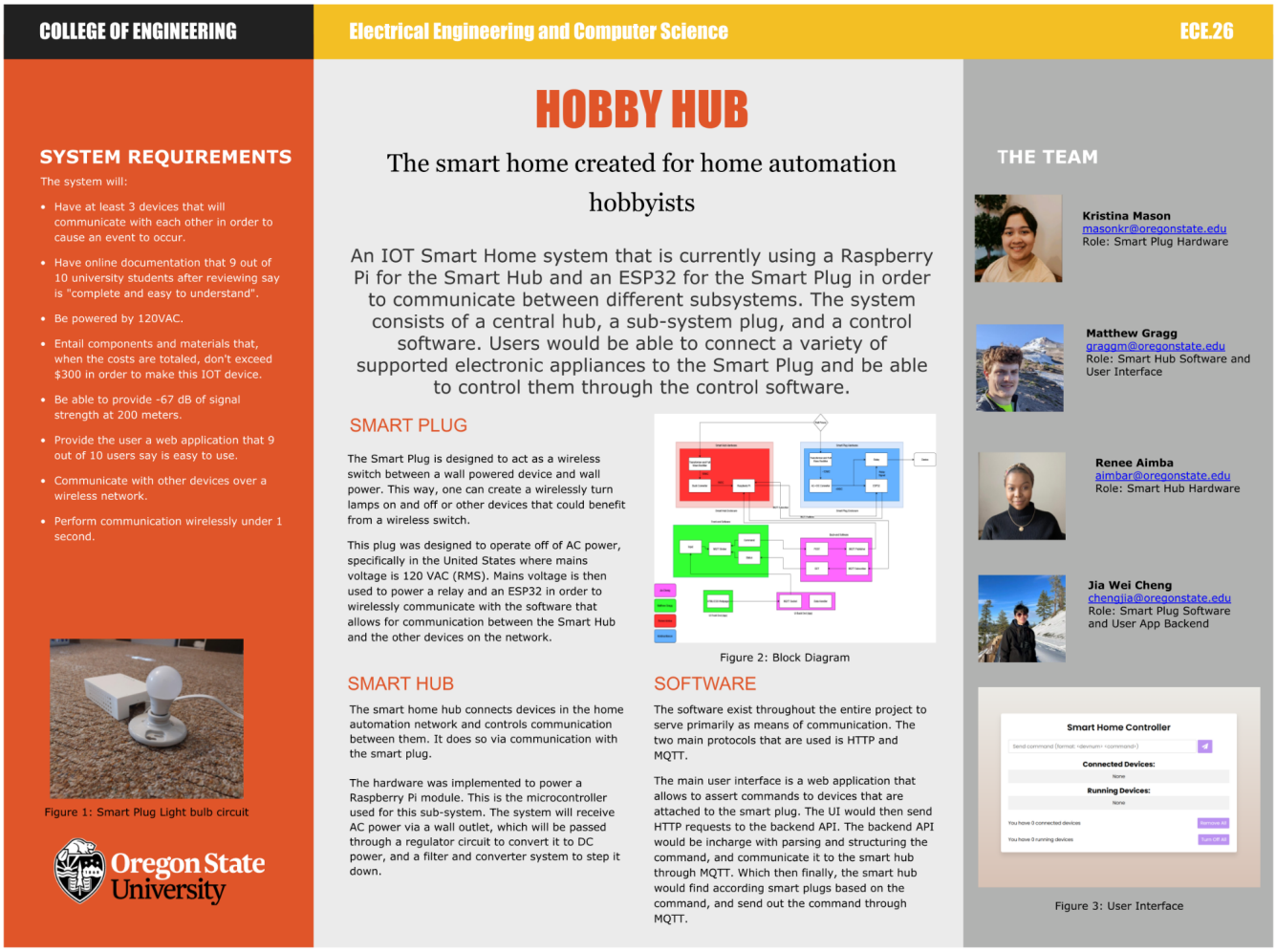

Figure 31: Project Poster

Link to Project showcase page:
https://eecs.engineering.oregonstate.edu/project-showcase/projects/?id=0ohprz42aRrUiyHo