COEN/ELEC 390 Report

Team 7 – SmartPlug App

Alexandre Page-Slowik	40193184
Anthony Louis D'Amico	40180639
Lotfi Boubchir	40202760
Mamadou Diao Kaba	27070179
Toby Smith	40165892

Abstract– This document outlines the details of the project developed by Team 7 for the COEN/ELEC 390 course. The details will communicate the purpose of the product and how it functions. It will also contain testing details and additional details like ethical considerations.

Table of Contents

1. Mission Statement

The Smart Plug is an AC mains feed-through (similar to a power bar) that monitors power consumption and gives user access to turn it on/off remotely.

The Smart Plug enables users to gather data surrounding their power use. With this information, they are able to make decisions as to connect and disconnect their devices remotely in order to be as energy conscious as they desire. Users gain the ability to turn on their outlets when they desire as well. For example, if you're running home late you can turn a light on for your pet to make sure they aren't left in the dark.

The estimated cost per unit during development is estimated to be $49.00 (see materials breakdown). Development costs are assumed to be roughly $15,000. With the product sold at $75, the development costs are recovered after the sale of 600 products. For higher volume production, the per unit cost would be greatly reduced and development costs greatly increased as a factor of purchasing in volume but requiring a method of high volume production (i.e. contract manufacturer).

The users will also be the purchasers of this product. It is aimed towards environmentally conscious consumers. Specifically, targeting North American, Android phone users. The product allows its users to track power usage of individual electronics in their home and make decisions on how to cut down on power usage.

An assumption of this product is that it is marketable. With many "smart plugs" in the market, we are assuming that our product is desirable since it will provide the user with single plug power monitoring. Another assumption is that the product is very feasible. With less experienced engineers, this is a risky assumption but is one that gives reasonable margins of financial success. A final assumption, is assuming that the capital to develop at a minimum of 600 units is available to the development team in order to make the product financially viable.

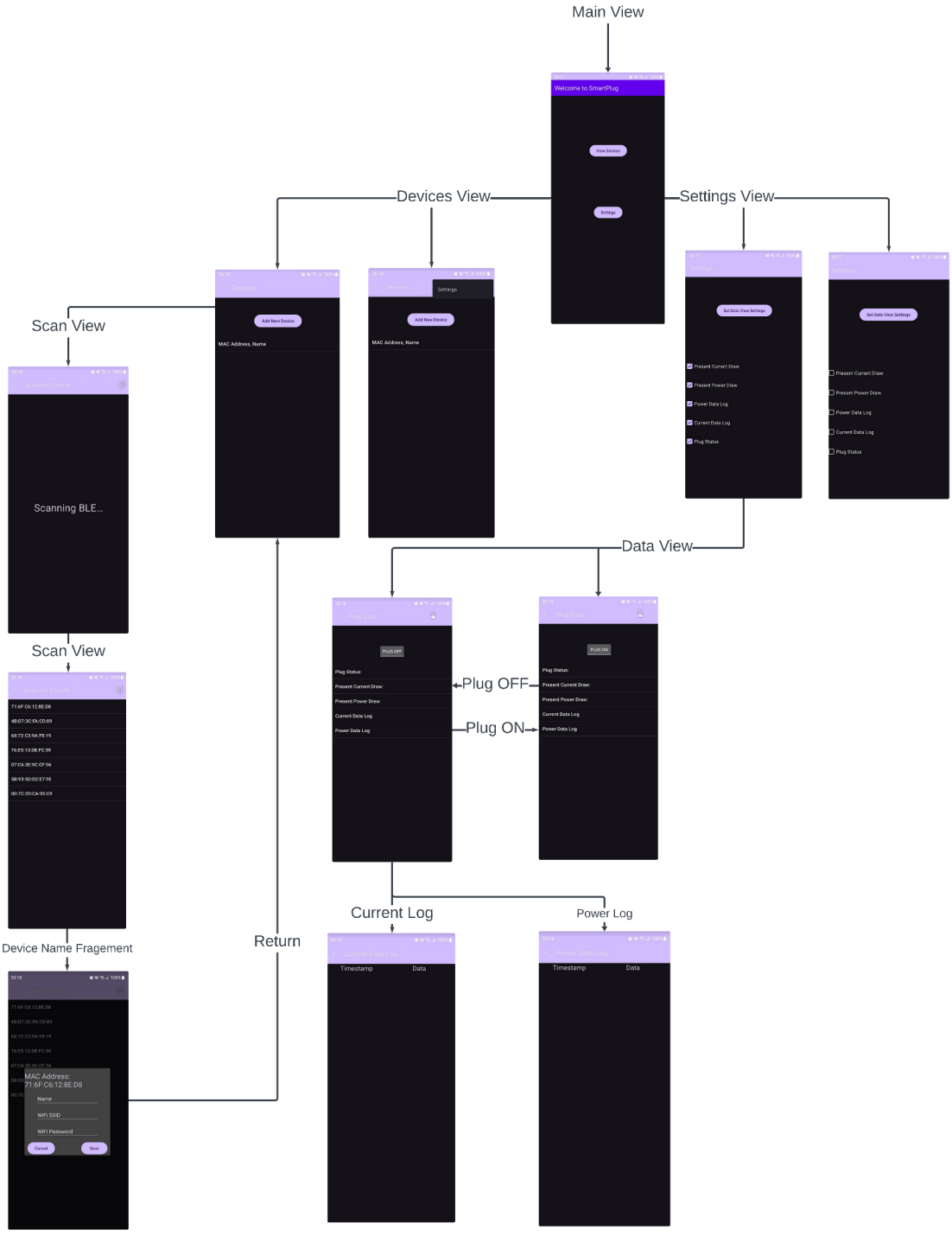
The initial assumption of marketability has evolved. We understand that how the data is displayed will be key to the marketability of the device. The assumption of feasibility has been proven by the ability to create the product. However, not all features that were expected to be added by the final demo were fulfilled. Finally, the assumption to produce 600 units is as unreasonable now as it was at the project's inception.

A considerable constraint of this project is the timeline. The strict deadline of a demo date requires consistent and diligent sprint planning and execution.

The stakeholders for this project are energy conscious homeowners and renters alike. The ability to monitor and control the power usage in their residence enables them to make decisions about their power usage. The user-friendly representation of the data is more important than the accuracy of the displayed data according to interviews conducted with stakeholders.

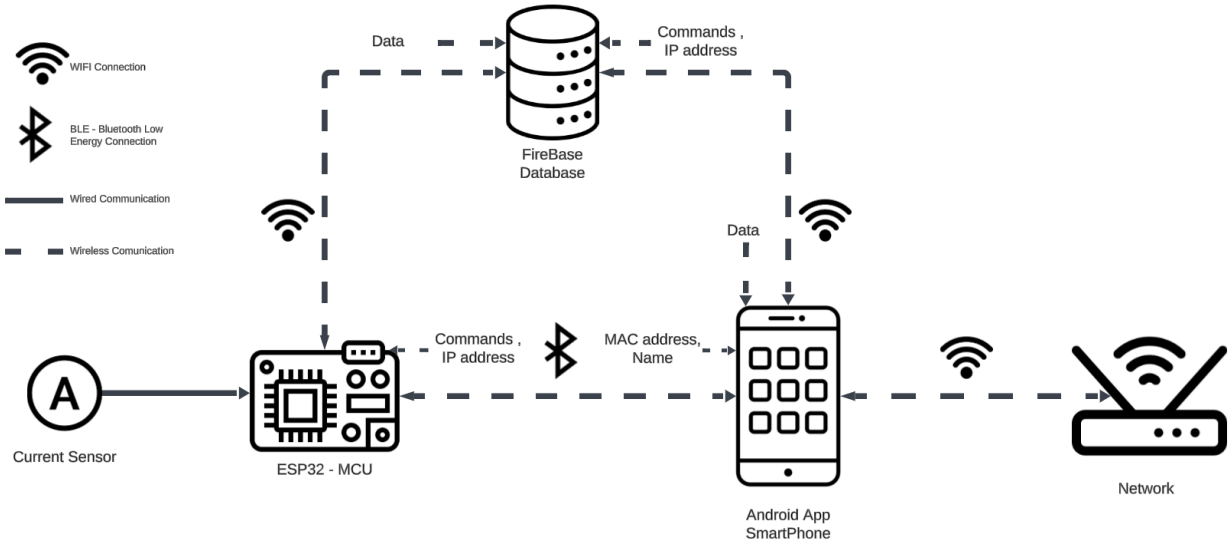
2. Design

Figure 4.0.1 – Design Sketch



2.1 Connection Architecture

Figure 4.1.1 – Architecture diagram for a Current sensing & Current & Power measuring device



We obtain data from a current sensor which is continuously being monitored by an ESP32 - micro-controller.

BLE Connection:

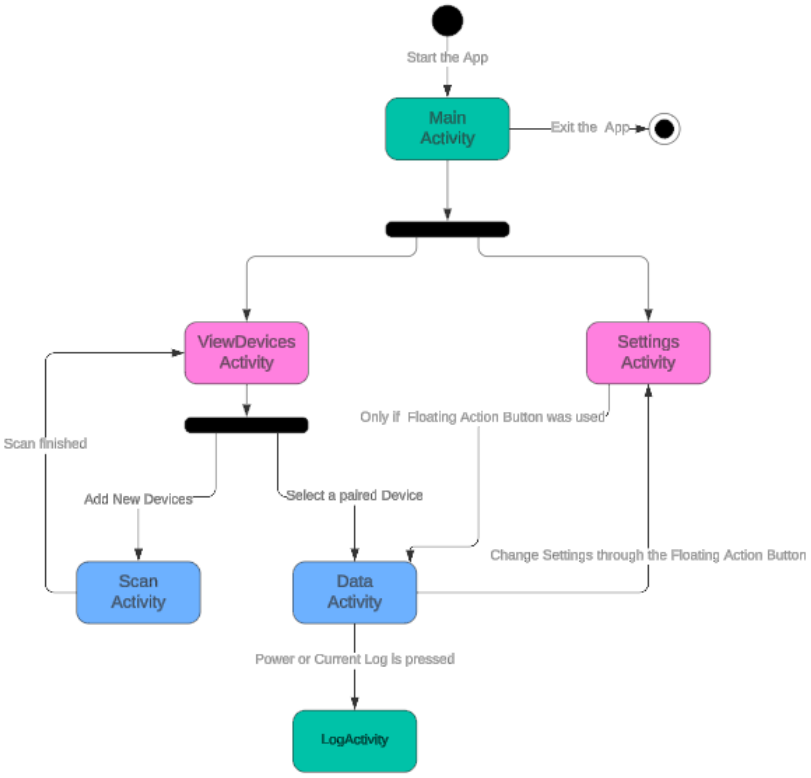
The ESP32 continuously advertises if no application has paired. Once a BLE connection has been established, the application sends the ESP32 the WiFi credentials for the network the ESP32 will connect to. The BLE connection then drops and is no longer available unless the device is deleted from the application.

Wifi Connection:

Once WiFi connection is established, communication between the ESP32 and application is managed through the Firebase database. This gives a centralized location for communication, simplifying the wireless link.

2.2 Android Application Activity Structure

Figure 4.2.1 – Android Application Activity Structure Diagram



The application starts in the Main activity, where the user can either access the Device activity or the settings activity.

- The settings are accessible from the main view through a button or through the menu inflater that will provide a “Settings” option. When either the button is pressed or the menu option pressed, the user will be directed to the Settings Activity.

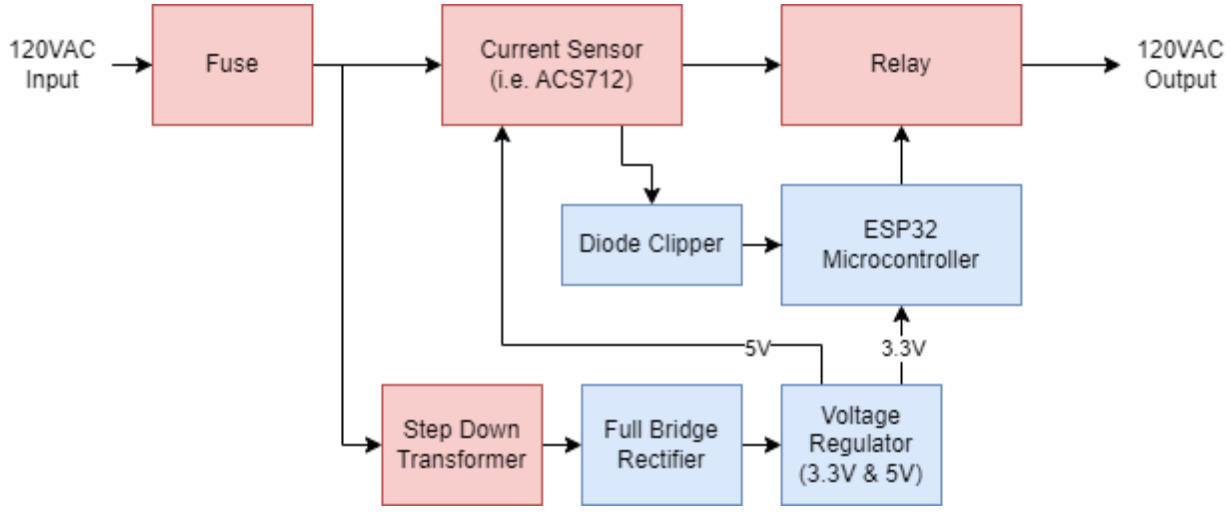
The user can navigate to the device Activity, where “known” devices can be paired to again or the user can decide to add a device which is done in the Scan activity. This activity will allow the smartphone to listen for any advertising packets in the vicinity and connect to them while exchanging MAC address, device name and the IP address with the SSID and Password.

The app then fetches data from the FireBase database whenever device information is required, this occurs in the device view and data logging views mainly.

2.3 Hardware Design

2.3.1 High Level Architecture

Figure 4.3.1.1 – Hardware Architecture Block Diagram



The application uses a ACS712 which is a current sensor which varies linearly with the input signal.

The ESP32 will be powered through a Step Down Transformer which will take in 120VAC and output 12VAC to then be rectified by a full bridge rectifier. The newly made DC signal will be used to power a Voltage regulator which will output 3.3 VDC and 5 VDC to power the ESP32 micro-controller and the ACS712 Current sensor respectively.

The system is broken down into two parts, the ESP32 Support (pictured in blue) and the Power Portion (pictured in red).

2.3.2 ESP32 Support

The purpose of this board is to house the ESP32 microcontroller, the Full bridge rectifier, the voltage regulators, and the diode clipper. This module manages everything that does not need to be connected to the 120VAC lines. Additionally, the separation between the ESP32 Support allows for the ESP32 mounting position to be unique compared to other modules. This may prove to be important because of the concern of BLE and WiFi 2.4GHz signals propagating through a grounded metal box.

The ESP32 microcontroller was selected for its ease of integration and WiFi and BLE support. The ESP32 is used in a variety of projects and has support on the Arduino platform. This reduces integration complexity since our team is able to work with a System on Chip (SoC) and Operating System (OS) that is proven and that we have experience with. The availability of libraries for the Arduino platform greatly reduces the knowledge debt.

The input power (nets TRANSFO+ & TRANSFO-) comes from the output of the step down transformer. This sinusoid is converted into a DC signal using a full bridge rectifier. The ~15VDC power line is fed into 2 linear regulators to provide 5V and 3.3V rails. The 5V rail provides power for the current sensor. The 3.3V rail provides power for the ESP32. The power required from the 5V rail is at most 15mA & the power required from the 3.3V rail is on average 500mA. For this reason, a relatively low power and small profile 5V regulator can be used. The 3.3V regulator must be able to supply at least 500mA and dissipate 5.85W ((15V - 3.3V)*500mA). For this reason, a suggestion for the next revision would be to use a switching regulator for the 3.3V rail in order to not get the same power loss, and therefore power dissipation requirement, as a linear regulator would.

The headers (J3, J4, & J5) present on the hardware are not populated in the final revision, these are provisioned to allow for ease of testing. Removing them will reduce the overall size and profile of the ESP32 Support Board without affecting functionality. The ESP32 can still be programmed through the USB to UART board since the male headers on the bridge can be fed through the through hole pads and torqued to make electrical contact. The UART lines from the bridge are used to program the ESP32 and provide a communication line to the PC. The ESP32 uses a UART bootloader to be programmed by the PC without requiring the purchase of an expensive JTAG debugger. In order to have the ESP32 reference its UART bootloader, the GPIO9 must be pulled LOW. For this reason, the output power from the USB to UART bridge is fed into a Common source NMOS whose Drain is connected to GPIO9.

Figure 4.3.1.1 - ESP32 Support Schematic

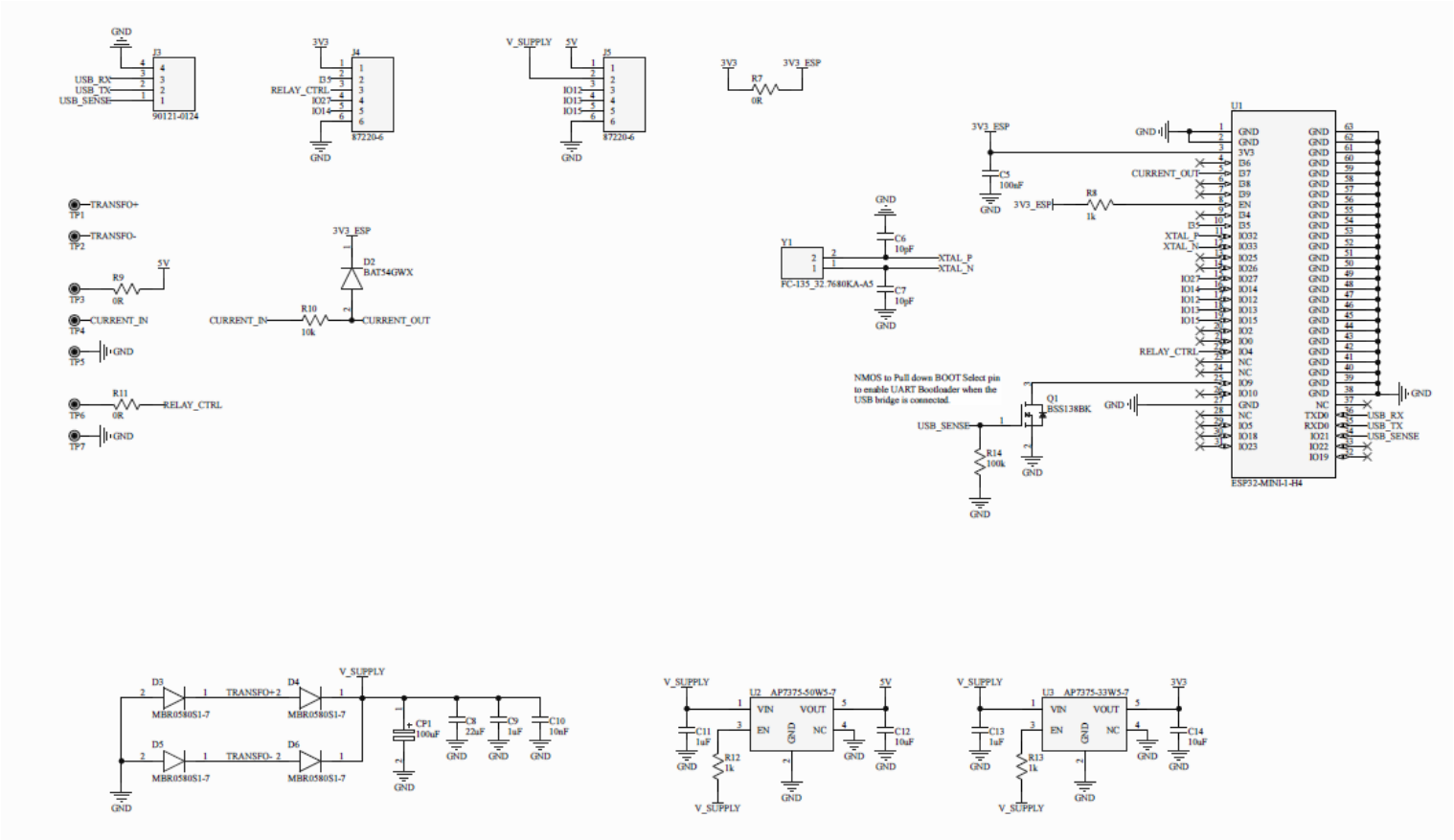


Figure 4.3.1.2 - ESP32 Support PCB Layout

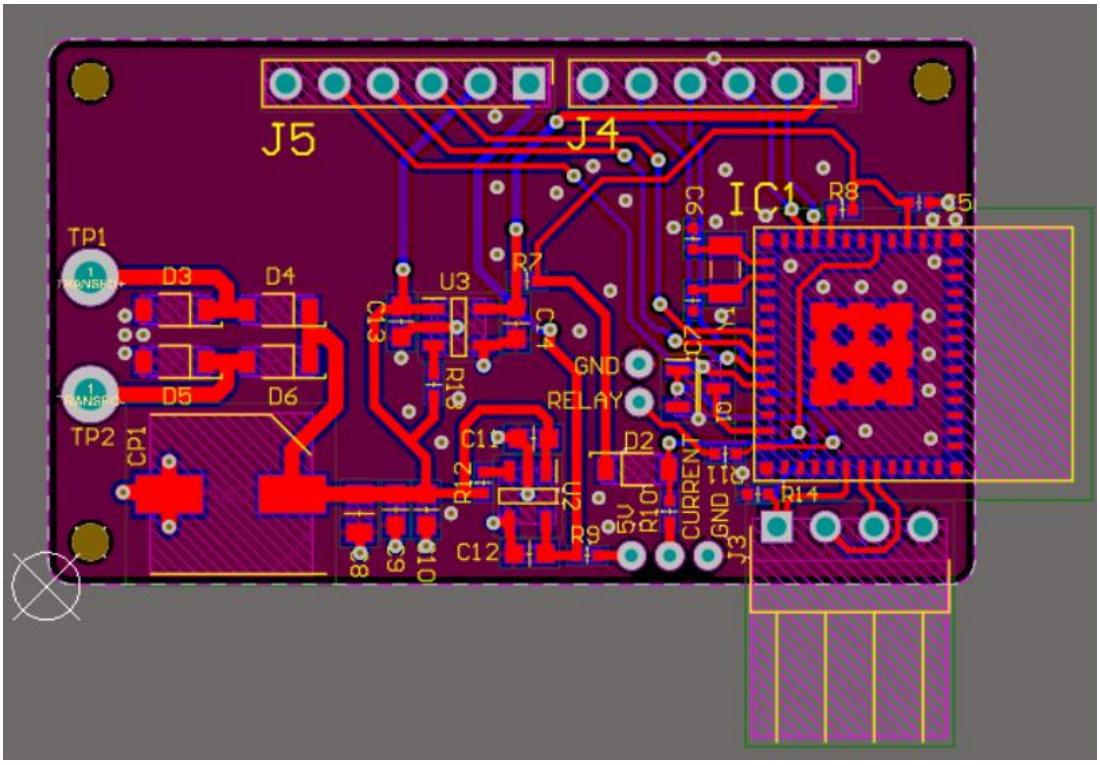
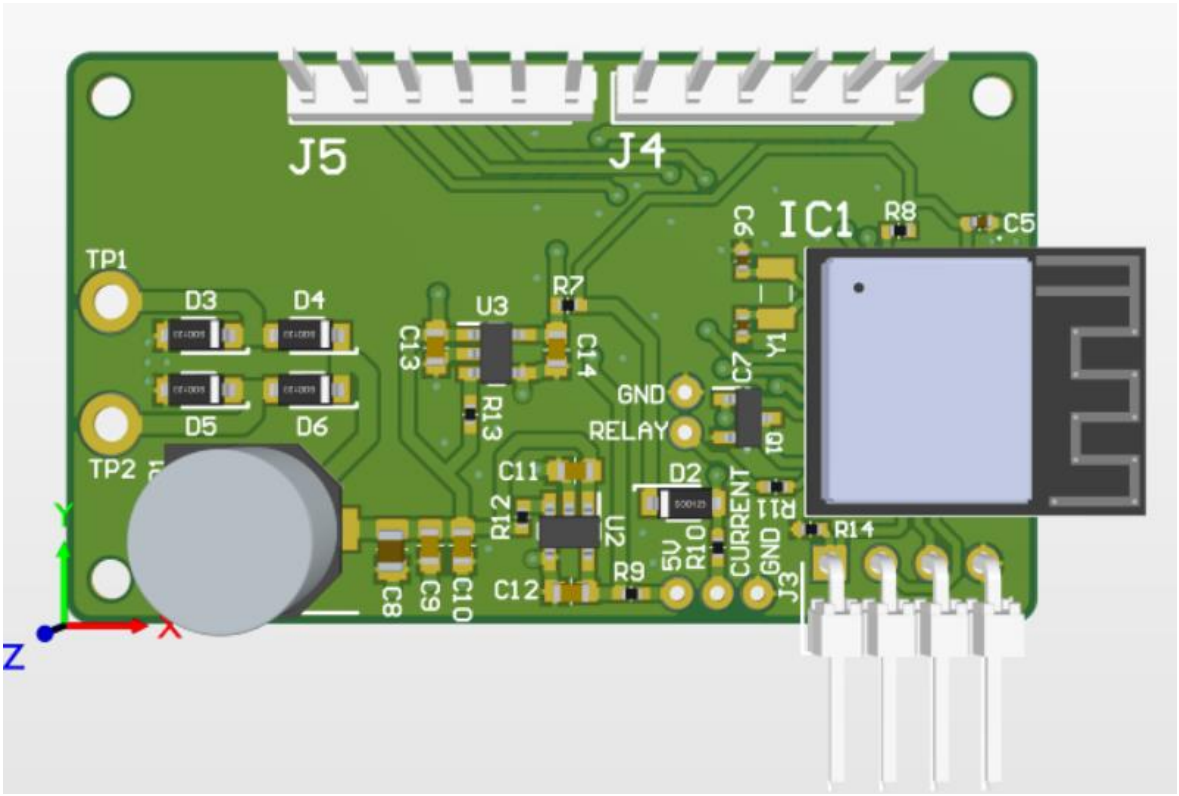


Figure 4.3.1.3 - ESP32 Support PCB 3D Model



2.3.3 Power Portion

The power portion manages all of the connections and components that interface directly with the 120VAC. This includes the fuse, step down transformer, current sensor, and the relay. It also provides contacts necessary to solder an outlet output and plug input to the device. The PCB was designed to isolate up to 240Vrms. This is done by having appropriate gaps between the power traces. At higher voltages, the chance of arcing becomes a concern, therefore this air gap must be greater than the rated recommendation.

Figure 4.3.2.1 - Power Portion Schematic

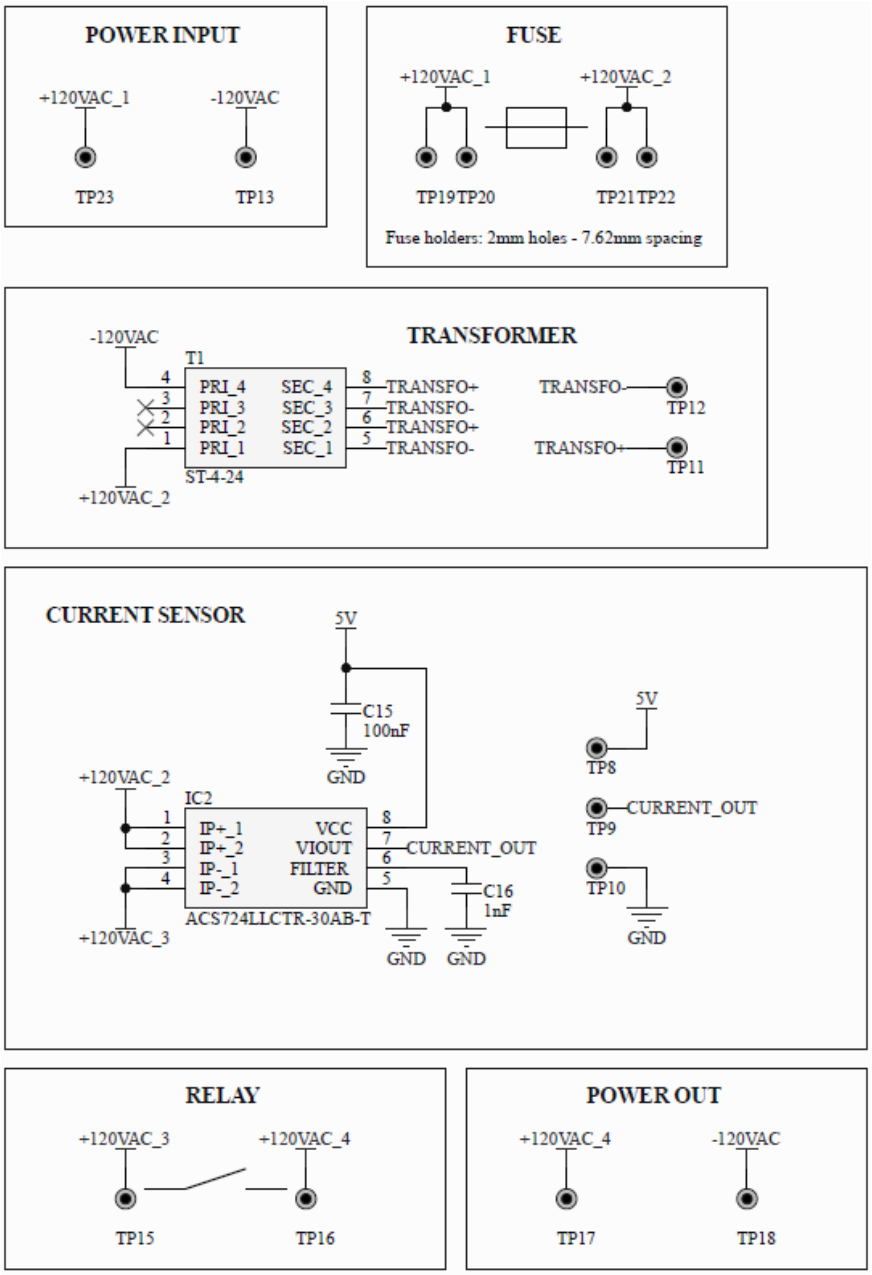


Figure 4.3.2.2 - Power Portion PCB Layout

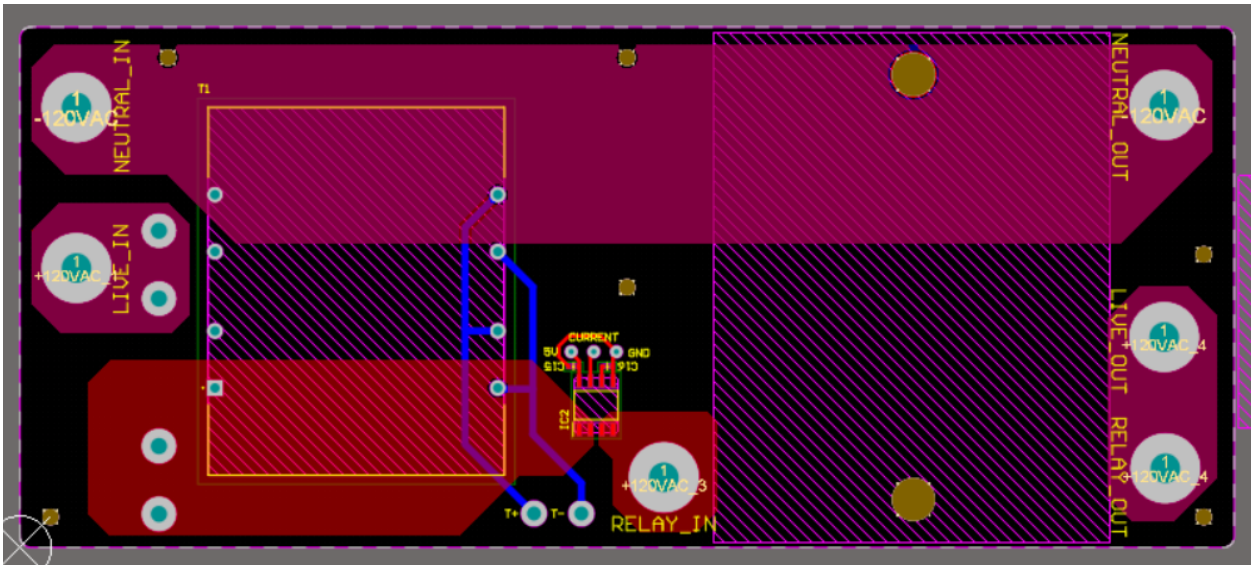
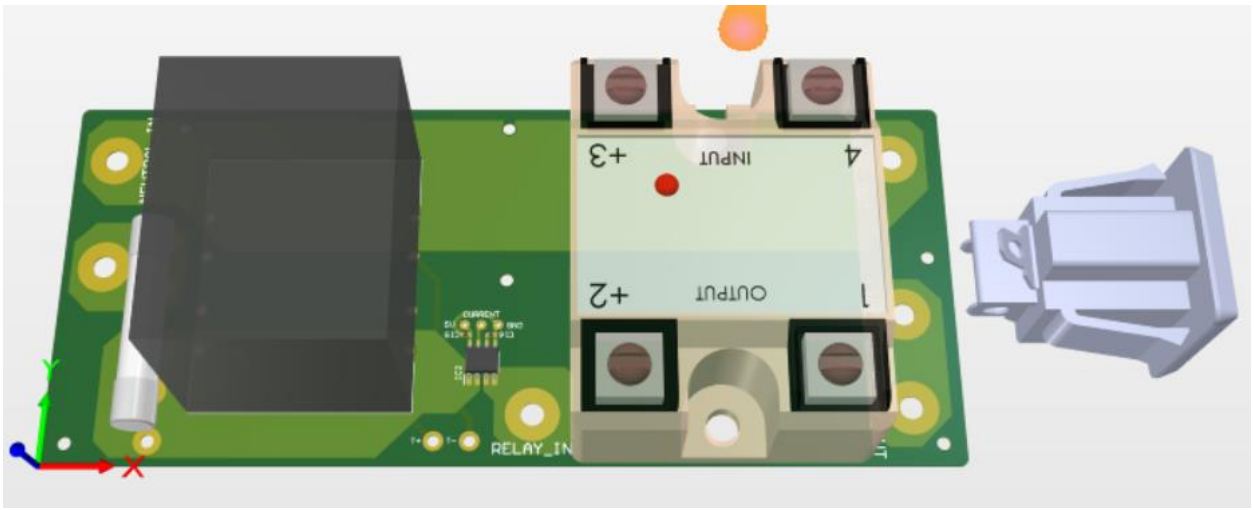


Figure 4.3.2.2 - Power Portion PCB 3D Model



2.3.4 USB to UART Bridge

The purpose of the USB to UART bridge is to provide a communication link to and from the PC to the ESP32. This is done so that the ESP32 Support board does not require additional components, increasing its overall size and the system bill of materials. It takes a micro-USB connector and converts to UART protocol through the CP2102 IC. The output from the USB to UART bridge are the Tx and Rx outputs from the converter. Additionally, the CP2102 has an internal 3.3V regulator to convert the 5V power from the USB line to 3.3V. This is used for the ESP32 to determine when the bridge is connected and powered.

Figure 4.3.3.1 - USB to UART Bridge Schematic

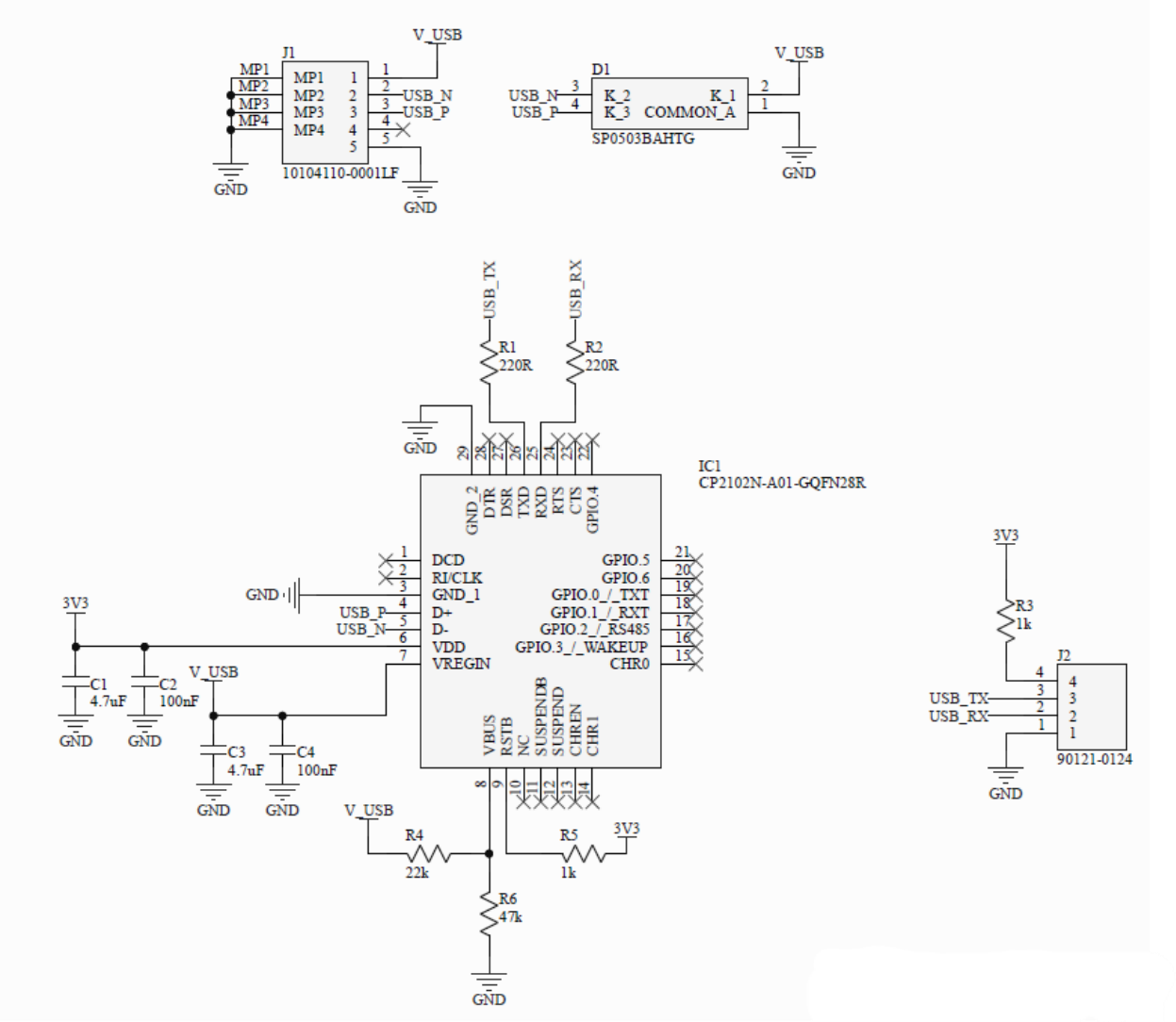


Figure 4.3.3.2 - USB to UART Bridge PCB Layout

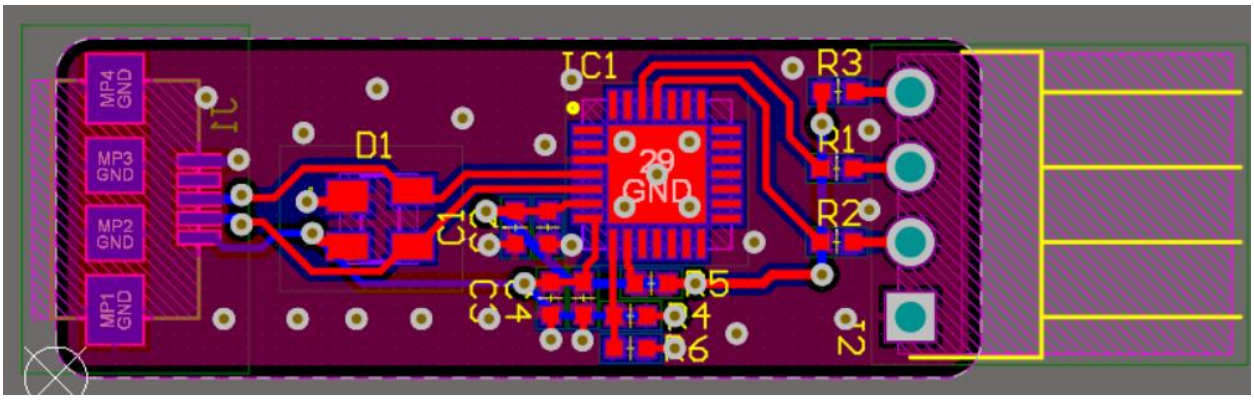
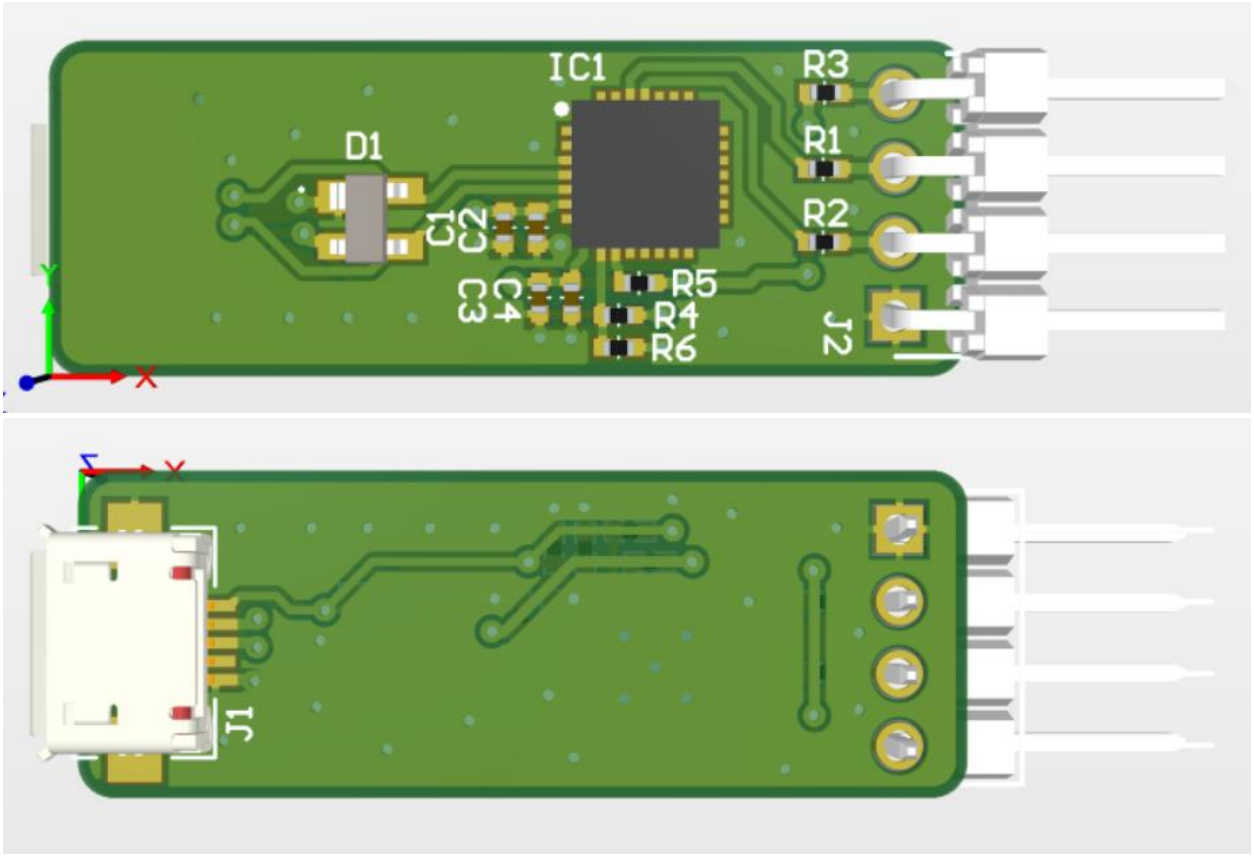
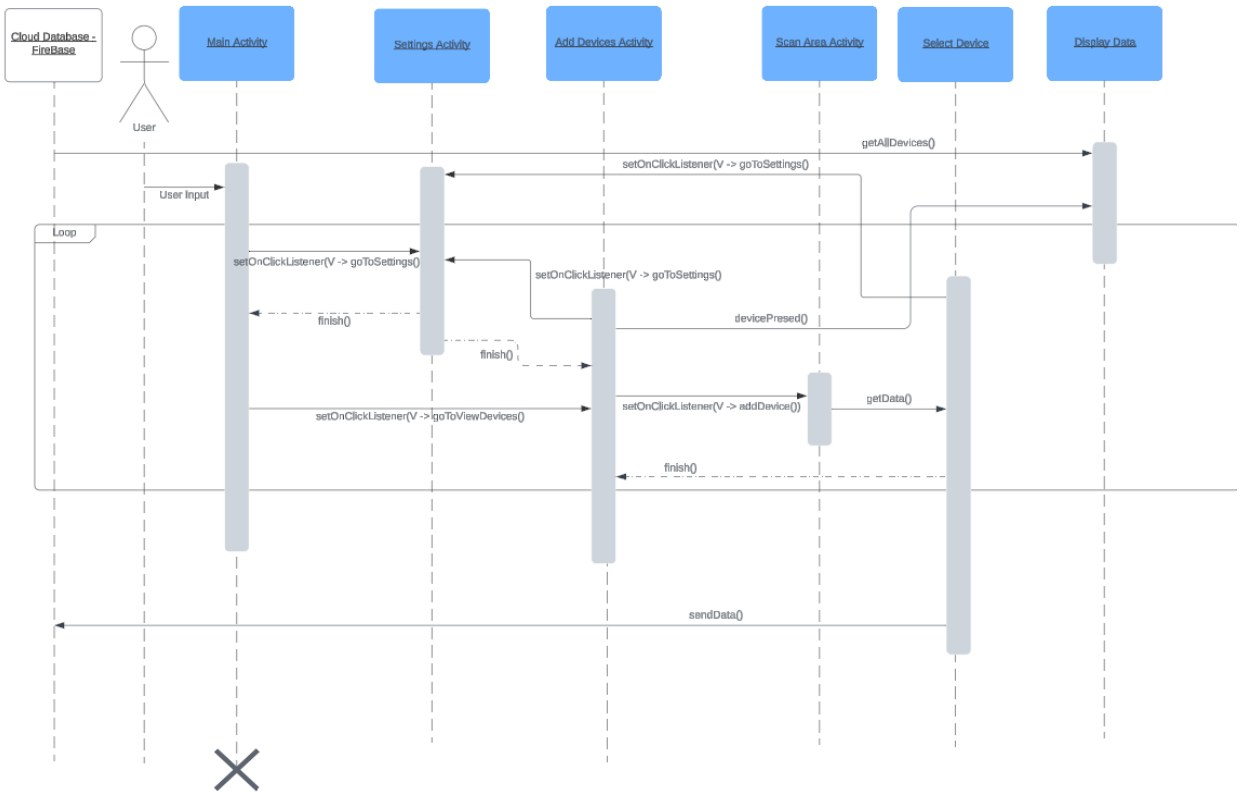


Figure 4.3.3.3 - USB to UART Bridge PCB 3D Model



2.4 Software Architecture

Figure 4.4.1 – Software Architecture Block Diagram



The software architecture will be explained by going over the main activities and any related class used through the application.

For the app to work, we use a base class of Device which will be our main variable when dealing with data transfers. This means that we will mainly be sending and receiving packets containing either an array of Devices or simply a singular device which contains all the information needed for the program.

When running the application, the user is greeted in the Main Activity where the user can either access the Device or Settings activity.

The Device Activity allows for users to add new devices which prompts the use of the Scan activity. This activity sets the android phone to start scanning for advertising packets from nearby devices such as the ESP32 and displays their MAC addresses. The user then picks the appropriate MAC address and is then prompted with a fragment to send the Wifi SSID, password and name back to the ESP32. The user is then sent back to the Device Activity. Furthermore, the

paired device can be accessed to display the information that was selected from the Settings Activity.

The data displayed and stored whenever the user is asked for input or the application receives packets over wifi or BLE is handled by the DatabaseHelper which communicates with a cloud database, FireBase, to either fetch or store data.

2.5 BLE Communication

The Bluetooth Low Energy (BLE) communication system is the first point of contact from the application to the hardware. The BLE system sends Wi-Fi SSID and password information to the smart plug (ESP32). The plug then automatically connects to the internet, and establishes a connection with the database. In order to establish the BLE connection both devices go through different modes in order to ensure proper bonding or pairing.

Bonding: In order to proper bond a device to another, the following sequence has to be met;

- The Central & Peripheral are in the Standby state (Devices do no transmits nor receive packets)
- Peripheral transmits advertising packets to nearby devices
- Central is in scanning state, where it will “listen” for any incoming advertising packets
- Once the central receives those packets it moves to the Initiating state where it responds to the particular peripheral. The information broadcasted by the peripheral is only its name & the services it offers.
- Then finally, once the connection requests have been handled, both devices can now move down to the Connected state. Only now can the ESP32 exchange MAC address and request for the IP address through a DHCP request.

Any connection after the bonding process is done once, we then call it Pairing.

Once the Wi-Fi connection has been established, the BLE system shuts off to reduce power consumption as all future communication is handled by the database.

2.6 IP Communication

Once having requested an IP address through a DHCP request, our ESP32 can now freely access that network. This feature allows us to utilize an online database, firebase offered by google services to host our database where the ESP32 will update data that we are interested in. From here, the smartphone will fetch the plug status, current draw, and a timestamp of the current draw measurement to be used and displayed.

Wifi communication is preferred due to its higher reliability but mainly for its extended range compared to BLE allowing the user to be further from the ESP32 and still have access to any data it desires.

Due to the DHCP assignment of the IP address of the ESP32, the Firebase database will be the form of communication used for IP communication. It will handle the communication according to Section 2.7.

2.7 Firebase Database Schema

2.8 Application SQLite Database Schema

The application will locally store the Devices that have been added to the application. Additionally, for each device added, a table will be added that will be used to store the history of current readings in order to output a log of stored historic current readings associated with the device.

"Devices" - Table		
ID (Integer) *Primary Key Auto-incremented	Name (Text)	Mac Address (Text)
1	LivingRoomLamp	0C:0D:xx:xx:xx:xx
2	WorkshopOutlet	1D:A4:xx:xx:xx:xx

"Timestamp_Table_" + Device Name - Table		
ID (Integer) *Primary Key Auto-incremented	Current (Double)	Timestamp (Text)
1	1.23	2024:03:27 @ 18:45:12
2	1.678	2024:03:27 @ 18:45:45

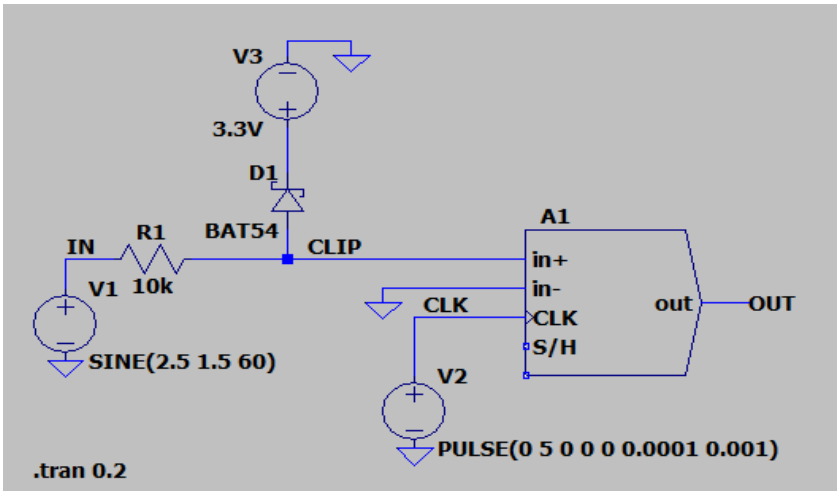
The tables above are filled with example data that could be filled in through use. The timestamp table is created and dropped when the device is added or deleted from the devices table.

2.9 Cost Breakdown

Item	Purpose	Cost
ESP32	Embedded microcontroller that can connect via wifi.	$5.00
AC Mains Outlet	Provides power to users plugged in device	$2.00
AC Mains Plug	Provides power to system & to outlet	$2.00
Metal Enclosure	To enclose the product and provide grounding protection.	$15.00
Relay	To allow current to pass from plug to outlet.	$10.00
Hall effect current sensor	To read current flowing through the “hot” wire from plug to outlet.	$2.00
Step down transformer	A 10:1 10VA transformer to provide our system with a useable AC power input	$3.00
Full bridge rectifier	To provide DC power from step down transformer output	$1.00
Regulator	To provide power to our microcontroller and sensor.	$1.00
User Button	To allow for a pairing process between phone application and device.	$1.00
Passive components	Decoupling capacitors, bulk capacitance, rectifier, etc.	$1.00
Fasteners	Screws to hold enclosure, plastic standoffs to isolate PCB from metal enclosure	$1.00
PCB	PCB to provide electrical connection between all components and reduce overall product size.	$5.00
	Total	$49.00

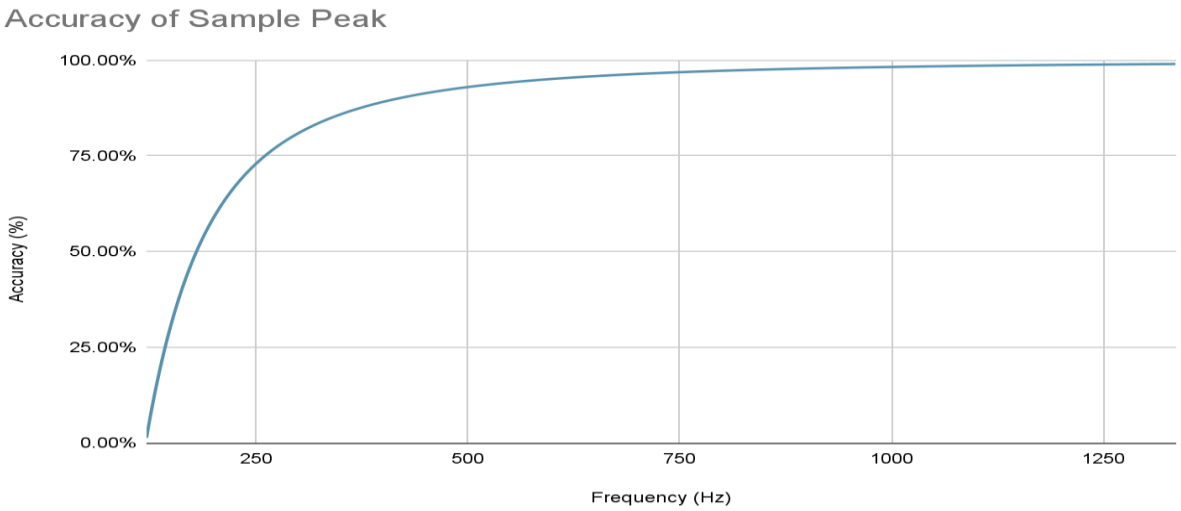
3. Computer Simulation Results

Figure 3.1 - Computer Simulation Setup



The computer simulation was set up on the LTSpice platform. Its purpose was to gather data on how sample rate will affect data collection. The initial setup of the simulation allowed our team to determine that simply following the Nyquist minimum sample rate would require recreating the signal in order to have an accurate reading. From this conclusion, we gathered that the sample rate would have to be high enough so that the sampled valley of the sinusoid would be used to determine the amplitude of the signal. The lowest accuracy of a single cycle is when each sample occurs at the same distance from the peak. It is determined by the formula $a_{min} = sin(\pi/2 + \pi(60Hz/f_s))$ where a_{min} is the minimum accuracy due to sample rate error and f_s is the sampling frequency. As the sample rate increases, a_{min} approaches 1. In order to achieve an accuracy of at least 90%, a sample rate of 425 Hz was selected.

Figure 3.3.1 – Minimum Accuracy as a Function of Sample Rate



4. Testing

Test Case: 1.1		
Pre-Condition: *Application launched. Devices Activity open. HW Advertising.*		
Steps:	**Expected Results**	**Actual Results**
1. Start Scanning for a device	Scanning screen is given. After Scan is done, a list of devices advertising is displayed.	As expected
2. Select device	When a device is selected, a fragment appears to send WiFi credentials to HW.	As expected
Result: Test Case Passed		

Test Case: 1.2		
Pre-Condition: *Application launched. Devices Activity open. HW Advertising.*		
Steps:	**Expected Results**	**Actual Results**
1. Settings button is pressed	Open settings activity	As expected
2. Plug status, Present Current Data, & Current data log are selected.	Check marks are selected when selecting options.	As expected
3. Set data view settings button is pressed.	Return to the home screen.	As expected
4. View devices button is pressed	Open devices activity. If not the first time entering the page, Toast error is thrown due to dummy data entry. Name, MacAddress is displayed	As expected
5. Press on dummy device	Data activity is open. The three data views shown are the Plug status, Present Current Data, and Current Data Log.	As expected
6. Press on current data log.	Current data log activity is opened.	As expected

Result: Test Case Passed.

Test Case: 1.3		
Pre-Condition: *Application launched. Scanning complete. HW connected to Arduino IDE. Serial Monitor Open.*		
Steps:	**Expected Results**	**Actual Results**
1. Data transfer from central to HW	Once BLE connects, on connect data will transfer to peripheral giving IP and password	As expected
2. Data error check	ESP32 runs error correction to compare if data sent and received are equal	As expected
3. Data receive confirmation	ESP32 sends confirmation once error correction is done	As expected
Result: Test Case Passed.		

Test Case: 1.4		
Pre-Condition: *Application launched. Scanning complete. HW connected to Arduino IDE. Serial Monitor Open. IP, SSID, password given to the ESP32 for Wifi communication.*		
Steps:	**Expected Results**	**Actual Results**
1. Data transfer from peripheral to FireBase database	Once Wifi is connected,the peripheral will transfer random value to the FireBase database.	As expected
2. Data error check	ESP32 fetches data from the database, compares it using the CRC in the packet sent.	As expected
3. Data receive confirmation	ESP32 sends confirmation once error correction is done	As expected

Result: Test Case Passed.

Test Case: 1.5		
Pre-Condition: *Application launched. Scanning complete. HW connected to Arduino IDE. Serial Monitor Open.*		
Steps:	**Expected Results**	**Actual Results**
1. Hardware Connects	All hardware interacting and can power plug-in device	As expected
2. ESP32 Reads ADC values	ESP32 Reads ADC values based on 12 bit connection	As expected
3. ESP32 Reads RMS current values	ESP32 Reads RMS current values based on connected HW	As expected
Result: Test Case Passed.		

Test Case: 1.6		
Pre-Condition: *Application launched. Home screen opened. View Devices Pressed.*		
Steps:	**Expected Results**	**Actual Results**
1. Press on dummy device	Data activity is opened. The four data views shown are the Plug status, Present Current Data, and Current Data Log and power Data Log.	As expected
2. Display Current draw.	Current in the format “ double + ‘A’ ” displayed next to "Present Current Draw: "	As expected
Result: test case passed.		

Test Case: 1.7		
Pre-Condition: *Application launched. BLE & Wifi implemented,merged & functional*		
Steps:	**Expected Results**	**Actual Results**
1. ESP32 advertising	Device is advertised using its Mac-Address. Connection over BLE & Wifi login information successfully transmitted.	As expected
2. Connect to Wifi	Device turns off BLE for power saving & connects to Firebase Database through Wifi successfully.	As expected
3. Send & receive data or commands through the Firebase Database.	Dummy values sent and received at the cloud database. Commands can be sent from peripheral firmware and the application from the central.	As expected
4. Fetch data from the cloud database to display in the data Activity.	Once a device is added, the device can be clicked on from the device activity. Where the user will be able to view real time data in the data Activity.	Ad expected
Result: test case passed.		

Test Case: 1.8		
Pre-Condition: *Application launched. BLE & Wifi implemented,merged & functional - (Full loop test - dummy values), current reading implemented*		
Steps:	**Expected Results**	**Actual Results**
1. ESP32 advertising	Device is advertised using its Mac-Address. Connection over BLE & Wifi login information successfully transmitted.	As expected
2. Connect to Wifi	Device turns off BLE for power saving & connects to Firebase Database through Wifi successfully.	As expected
3. Send & receive data or commands through the Firebase Database.	Real time current reading sent and received at the cloud database. Commands can be sent from peripheral firmware and the application from the central.	As expected
4. Fetch data from the cloud database to display in the data Activity.	Once a device is added, the device can be clicked on from the device activity. Where the user will be able to view real time data in the data Activity.	Ad expected
Result: test case passed.		

Test Case: 1.9		
Pre-Condition: *Application launched. Home screen opened. View Devices Pressed.*		
Steps:	**Expected Results**	**Actual Results**
Press on dummy device	Data activity is opened. The four data views shown are the Plug status, Present Current Data, and Current Data Log and power Data Log.	As expected
Display Power draw.	Power in the format “double + ‘W’ ” displayed next to "Present Power Draw: "	As expected

Result: test case passed.

Test Case: 2.0		
Pre-Condition: *Application launched. Home screen opened. View Devices Pressed. Dummy device Pressed.*		
Steps:	**Expected Results**	**Actual Results**
Press on Current Data Log	Current data log is opened. The two columns shown are the timestamps and the current data.	As expected
Display timestamp and current.	Timestamp in the format “day, month, YYYY + HH:MM:SS” and current in the format “ double + ‘A’ ” are displayed in their respective column	As expected
Result: test case passed.		

Test Case: 2.1		
Pre-Condition: *Application launched. Home screen opened. View Devices Pressed. Dummy device Pressed.*		
Steps:	**Expected Results**	**Actual Results**
Press on Power Data Log	Power data log is opened. The two columns shown are the timestamps and the power data.	As expected
Display timestamp and power.	Timestamp in the format “day, month, YYYY + HH:MM:SS” and power in the format “ double + ‘W’ ” are displayed in their respective column	As expected
Result: test case passed.		

Test Case: 2.2		
Pre-Condition: *Application launched. Home screen opened. View Devices Pressed. Dummy device Pressed. Current Data Log Pressed*		
Steps:	**Expected Results**	**Actual Results**
Press on the eye button next to Current Data Log	Graph view of the Current Data Log is opened.	As expected
Display the graph representation of the Current Data Log.	A Current vs Timestamp graph is displayed with the option to see the value of the current when touched on the graph.	As expected
Result: test case passed.		

Test Case: 2.3		
Pre-Condition: *Application launched. Home screen opened. View Devices Pressed. Dummy device Pressed. Power Data Log Pressed*		
Steps:	**Expected Results**	**Actual Results**
Press on the eye button next to Power Data Log	Graph view of the Power Data Log is opened.	As expected
Display the graph representation of the Power Data Log.	A Power vs Timestamp graph is displayed with the option to see the value of the Power when touched on the graph.	As expected
Result: test case passed.		

5. Ethical Dimensions

The Smart Plug, a device designed to monitor power consumption and provide remote control capabilities, introduces various ethical challenges that necessitate thorough examination. This innovative device enhances convenience and energy efficiency but also raises substantial concerns regarding potential misuse and privacy violations. As such, it is crucial to critically assess and proactively address these ethical considerations to ensure the responsible deployment of the Smart Plug.

Privacy and Data Security Concerns

The capability of the Smart Plug to remotely control and monitor electrical devices inherently introduces risks related to cybersecurity and privacy. The primary ethical concern is the potential for unauthorized access, which could lead to safety hazards or breaches of privacy. For instance, malicious actors might exploit vulnerabilities to control household appliances or gain sensitive information about users' habits and lifestyles from their power usage patterns. This information could be used for unwanted targeted advertising, profiling, or even surveillance, thus compromising user autonomy and privacy.

The collection, analysis, and storage of power consumption data further compound these privacy issues. Users may inadvertently reveal personal details through their energy consumption patterns, such as their presence at home, daily routines, or specific device usage times. To mitigate these risks, the Smart Plug utilizes a local SQLite database for data storage within the app, which is designed to reset after storing a certain amount of data. This approach not only limits the volume of data retained but also reduces the exposure period of any sensitive information, thereby enhancing privacy.

The security of the Smart Plug is further reinforced by the implementation of Bluetooth Low Energy (BLE) communication protocols, particularly through secure BLE handshake mechanisms. The BLE handshake process involves robust authentication and encryption steps that ensure only authorized devices can connect and communicate with each other. This secure connection prevents eavesdropping and tampering with the data transmitted between the Smart Plug and the user’s app, significantly reducing the risk of unauthorized access.

Addressing Stakeholder Conflicts

An additional ethical consideration is the potential for conflicts between different users or stakeholders. For instance, employers could use the Smart Plug to monitor the power usage of office appliances, indirectly tracking employees' work habits. Similarly, parents might employ the device to oversee their children's internet usage or television time, raising questions about the appropriate level of surveillance and privacy within the household. These scenarios highlight the delicate balance between using the Smart Plug for its intended benefits and respecting individual privacy rights.

To address these stakeholder conflicts, it is vital to implement user controls that enable customized privacy settings, allowing individuals to decide how their data is used and shared. This approach empowers users to maintain control over their information and sets boundaries for monitoring, thereby aligning with ethical usage practices.

Legal and Regulatory Compliance

In addition to addressing ethical concerns, it is crucial for the Smart Plug to comply with regional legal standards, such as those in Quebec. The device must meet the requirements set by the Quebec Act Respecting the Protection of Personal Information in the Private Sector (PPIPS), which governs the collection, use, and disclosure of personal data. Compliance involves obtaining informed consent for data collection, implementing adequate security measures, and ensuring users have access to their information. Furthermore, the Smart Plug must adhere to electrical safety standards enforced by organizations like the Bureau de normalisation du Québec (BNQ) and the Canadian Standards Association (CSA), which help mitigate the risk of electrical hazards and confirm the product's suitability for local use.

Conclusion

While the Smart Plug offers significant benefits in terms of energy management and convenience, its deployment brings forward substantial ethical challenges that must be proactively managed. By prioritizing user privacy, enhancing security measures, and ensuring regulatory compliance, the Smart Plug can effectively navigate these ethical complexities. Implementing a human-centered design and adopting principles of ethical innovation will maximize the device's advantages while safeguarding against potential risks and ensuring that it serves the best interests of all stakeholders involved.

6. References

[1] Santos, Rui. “ESP32 Analog Input with Arduino IDE | Random Nerd Tutorials,” May 31, 2019. https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/

[2] Espressif. “ESP32-WROOM-32 Datasheet,” ESP32-WROOM-32 Datasheet, Aug. 2016 [Feb. 2023]

[3] Allegro MicroSystems, Inc. “Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor,” ACS712 Datasheet, Nov. 2012 [Feb. 2023]

[4] “Advertisement packet – Nordic Developer Academy,” *academy.nordicsemi.com*. https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/lessons/lesson-2-bluetooth-le-advertising/topic/advertisement-packet/ (accessed Mar. 11, 2024).

[5] “BLE Advertising Primer,” *Argenox*. https://www.argenox.com/library/bluetooth-low-energy/ble-advertising-primer/

[6] “Understanding How UUIDs Are Generated,” *Aryaman Sharda*, Sep. 30, 2020. https://digitalbunker.dev/understanding-how-uuids-are-generated/ (accessed Mar. 11, 2024).

[7] “What Is an IP Address?,” *Fortinet*. https://www.fortinet.com/resources/cyberglossary/what-is-ip-address

[8] J. Bogna, “What Is a MAC Address, and How Does It Work?,” *How-To Geek*, Nov. 16, 2021. https://www.howtogeek.com/764868/what-is-a-mac-address-and-how-does-it-work/

[9] “Act respecting the protection of personal information in the private sector”, *LegisQuébec*, Feb. 2024. https://www.legisquebec.gouv.qc.ca/en/document/cs/p-39.1

[10] “Exigences de certification basées sur les normes ISO 17088 et ISO 18606”, *Bureau de normalisation du Québec*, March 2024. https://www.bnq.qc.ca/fr/nouvelles-et-evenements/detailevenement/708/40,178,39,140,55,170,171/consultation-publique-exigences-de-certification-basees-sur-les-normes-iso-17088-et-iso-18606.html