The Household Power Management System: “Smart Interior Monitoring Plug” (S.I.M.P.)

# Jose-Valentin Sera-Josef, Anthony Perez, Edin Durgutovic, Austin Raynor

# University of Central Florida School of Electrical Engineering and Computer Science, Orlando, FL

## *Abstract —* Our electrical and computer engineering project, the Smart Interior Monitoring Plug (SIMP) is a personal assistant that monitors power usage of electronic devices in ones home and discourages sustained use of said devices to reduce the user’s electricity bill, while encouraging the user to “disconnect” from technology in today’s world of tech-dependency. It does so by reading current going to any device that uses wall outlets for power, and shutting off power to said device or giving notifications to the user after it notices a device has been pulling current for an amount of time set by the user.

I. INTRODUCTION

At first glance a consumer may think that any additional features to be added to an outlet would be solutionism at its worst, because they perceive their outlet to serve a solitary purpose -that is, providing power to a connected device. But when we come to understand the role of an outlet we can begin to devise ways of incorporating Smart technologies in an effort to make them as intelligent and efficient as possible. What most consumers recognize by now is that advancements in technology means additional devices in their home; additional devices in a consumer's home means additional power necessary to power each individual device; additional power drawn from a consumer's home electricity will eventually inflate their electric bill - at what rate ultimately depends on the consumer. It's this simple problem, the issue of home power consumption in a world moving in the direction of automation and technology, that we aim to tackle and mitigate with our version of a Smart monitoring plug.

The Smart Interior Monitoring Plug (SIMP) will be able to provide both digital and physical support in the home to help monitor the power usage of the different devices plugged into it within a home or office. The monitored home layout will physically consist of one or several SIMP units plugged into the walls that are connected over Wi-Fi to a web server - each unit being no larger than a NEMA 5 plug/receptacle.. All the SIMPs will be monitoring the power usage of the devices plugged into them and sending the information to an online server where it will be saved. The data can then be accessed through a webpage to see the power usage consumed and costs spent over time. Some form of a smart digital assistant, “SIMPY” will aim to provide analytics to the user in the form of cost analysis, change in power usage for the month, as well as other basic functions to help the user learn more about their households power usage and offer solutions to reduce power usage while maintaining their basic quality of life.

II. SYSTEM COMPONENTS

The system, as a whole, is easier to understand when explained as a sum of its parts. This section splits the SIMP into its various electrical components and explains why we needed them as well as what each one does.

*A. Microcontroller*

The brains of this project is an Arduino Nano 33 IoT [1]. This specific microcontroller was chosen for its internet and Wi-Fi capabilities, coding capabilities and its incredibly small overall surface area that would helped keep our final footprint under the specifications we wanted.

*B. AC-To-DC Converter*

The power supply to the various electrical components in the SIMP is a MEAN WELL IRM-20-5 [2]. It draws from the 120VAC source of the wall outlet and safely and effectively converts it to 5VDC that powers our Arduino Nano 33 Iot, our current sensors and our relays with safety features included to safely connect and disconnect said electrical components.

*C. Relays*

We’ve incorporated two SONGLE SRD relays [3] in this project, with nominal coil voltage of 5VDC each. They are responsible for controlling the power to their corresponding outlets. They act as switches, cutting the power to the outlet off or connecting the outlet to power, using signals produced by the Arduino.

*D. Ammeters/Current Sensors*

Connected, in series, to the power line that feeds the outlet plugs are two current sensor modules that are able to convert the current passing through them into DC signals that can be read by our Arduino Nano 33 Iot. These current sensor modules each contain an ACS712 Linear Current Sensor [4] manufactured by Allegro MicroSystems, a pair of surface mounted capacitors for proper application, a surface mounted LED and resistor to indicate operating voltage, and a 2-port terminal block. We went with a 5A model for increased accuracy, and chose to use the module in our final design because of the added benefit of using the 2-port terminal block to connect the module in series with a more accurate ammeter so we could dial-in the correct currents we should be reading. The other added benefit of using a finished module is the headache we saved ourselves by having to solder an extremely small surface-mount style current sensor to our PCB.

*E. Master Switch*

The master switch we implemented into the design of the SIMP is a RB1 rocker switch manufactured by E-Switch [5]. This switch is the only physically operable component of the SIMP, and acts as a master ON/OFF button should the SIMP malfunction or need to be reset. It reliably holds the power being taken from the wall outlet and can be switched on or off to stop or allow that power from powering the SIMP. It also does the work that the relays do, only it will only be able to operate both outlet plugs at the same time.

*F. LEDs*

The two 5mm LEDs on the SIMP are the primary forms of physical communication to the user for more simple messages. These messages can be signaled through the flashing of one or multiple LEDs or a specific LED staying on or staying off.

III. SYSTEM CONCEPT

We aimed at making the main user interface as simple as possible. With simplicity in mind, the main hub or home page of the SIMP website will consist of two main selections. After logging in to the website with a valid email and password combination, the user will have the option of two selections. The first selection,“Electricity Monitoring”, will display the current kilowatts being used by either device plugged into the SIMP. Further selections in the “Electricity Monitoring” page will include total power used in the passed amount of time the user can select, like past day, past week, or past month. The second selection in the main home page will be “Parental/Master Controls”. This page will give the user power over the electricity being connected to either device plugged into the SIMP, as well as the ability for the user to input a maximum allotted time for the device connected to the SIMP to be considered ON. Once selecting this maximum allotted time option for either one or both plugs of the SIMP, the user will be prompted with the option of setting a specific time slot for when the selected device(s) is allowed to be powered or setting an amount of hours the device is allowed to be on every day. This will allow the user to set their own schedules or stop themselves from overusing a device they wish to “disconnect” from.

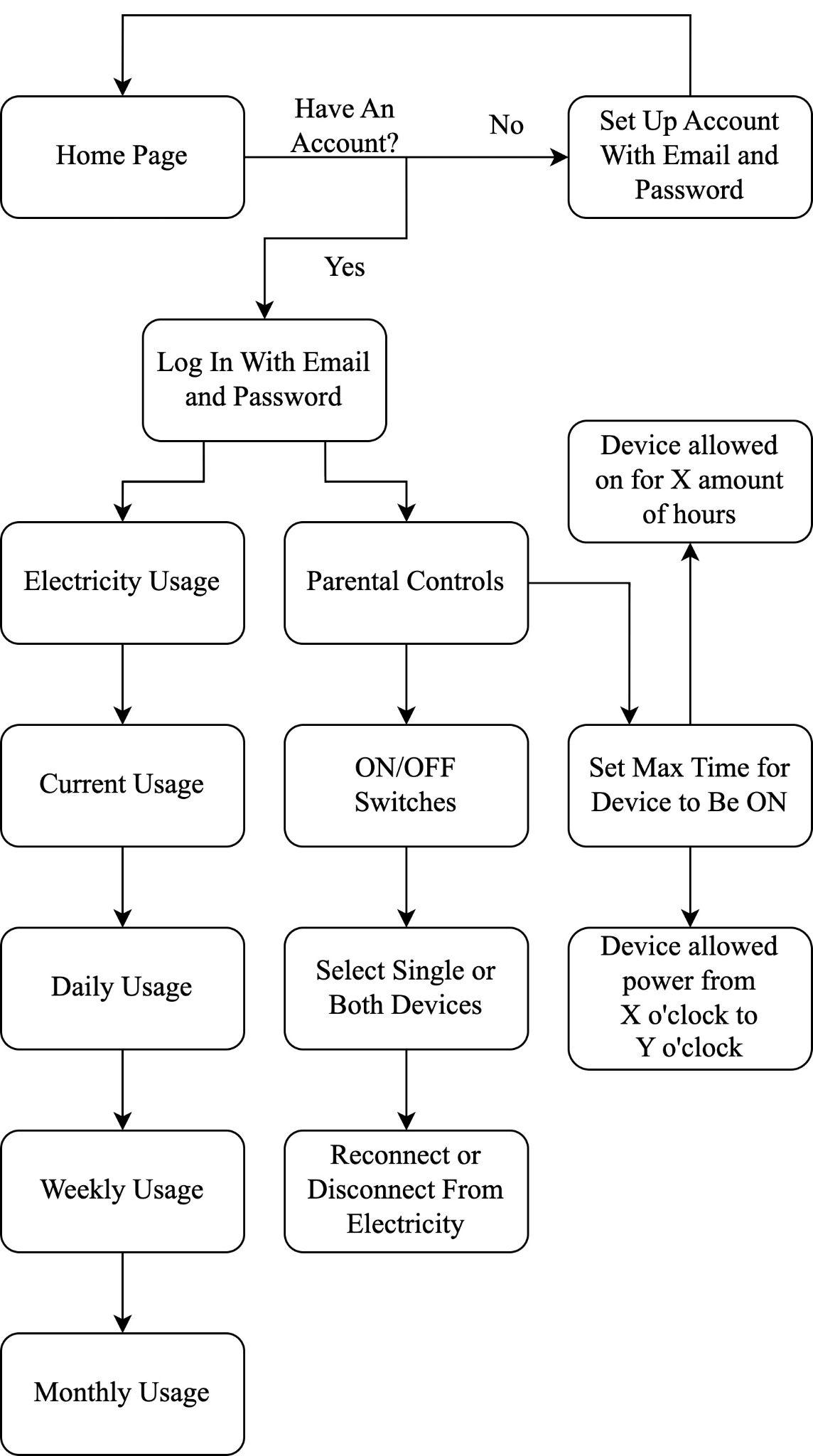


Fig. 1. Flowchart of the user interface/website used to control the SIMP device(s) and keep track of monitored electricity usage

IV. HARDWARE DETAIL

This section will dive into the intricacies of the different segments of our final schematic and PCB. Some require more in-depth explanations while others are more straight-forward.

*A. Relays*

In our search for help in designing our schematic for our final PCB, we found a few articles and posts about different experiments using the Keyes\_SR1y relay module we used during breadboard testing. One such article, posted on techdiy.org [6], provided insight into the schematic of the module, as seen in Figure 1, which made replicating to our PCB schematic very easy. Changes were made to the schematic, though, in terms of the NPN transistor, as it was a surface-mounted style component, which while great for mass production and pre-assembly, is nearly impossible with very small components such as the transistor that came on the module. Finding a through-hole style transistor with equal specifications was an almost-instant process, and the relay segment of our board was complete.

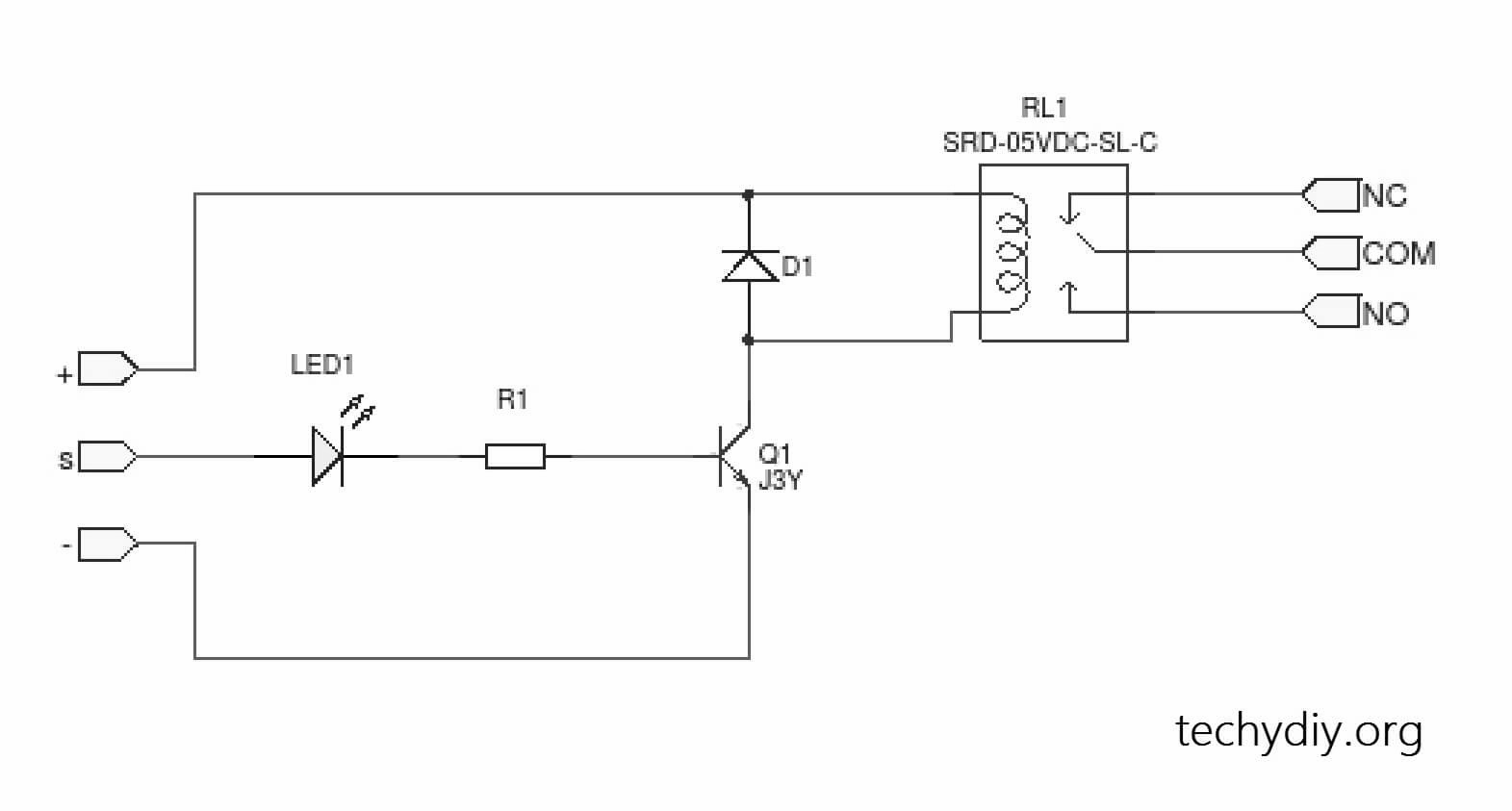


Fig. 2. Circuit schematic of relay module used in breadboard testing.

*B. Current Sensor*

When determining how to record the current drawn by whatever appliance is plugged into the SIMP unit, we found that there had to be a balance between the accuracy of the sensor and the maximum range of current the sensor could read. We tested three types of current sensors to find the best fit for our system: a noninvasive AC current sensor using a split core transformer and a hall effect current sensor.

The split core current transformer works by having the primary coil be the wire the AC current is traveling through and the second coil be a coil of wire that is connected to the device reading the measurements. The current being measured creates a magnetic field around the wire that induces a current in the secondary coil that is a known ratio smaller than the original current. In our case, we placed a burden resistor across the ends of the secondary coil to create a voltage output we could measure using the Arduino’s onboard Analog-to-Digital Converter. The problem our team ran into was that the readily available transformers were rated for such a high current range that the appliances we were testing could not draw enough current to induce a secondary current that was reliably measurable by the Arduino.

The other current sensing components we tested were hall effect current sensors in the form of integrated circuit components. These components work by being placed in series with the load whose current draw is being measured. The input current is then fed through a wire that again induces a magnetic current which is then used to create a voltage output that is a known ratio in magnitude smaller than the input current. We found the readings we got with these sensors were much more accurate and measurable at the current ranges we were testing the system with.

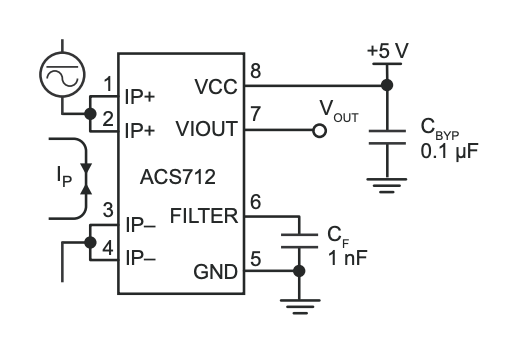


Fig. 3. Circuit schematic of a typical application of the hall effect current sensor used in our final schematic

We found that after several rounds of testing, the hall effect current sensor ICs were a better, more reliable way of measuring the load’s current draw at the current range we were expecting to use with the system.

*C. Microcontroller*

When designing our system, we knew the microcontroller we planned on using needed Wi-Fi capabilities to communicate to the web server. After looking at several options of boards, we decided to use the Arduino Nano 33 IoT. We chose this specific board because it came with an onboard Wi-Fi module and allowed us to use the Arduino IDE, which comes with plenty of documentation. This would help us streamline the development process and let us focus more on the current measuring as well as the control components of the hardware portion of the project.

We also looked at other microcontrollers, such as the MSP430, that we have worked with before in previous classes like Junior Design and Embedded Systems. We found that while using it would save us money, we would be trading a lower cost for time spent implementing the features which were already present in the Arduino Nano 33 IoT.

When comparing the Arduino Nano to other Wi-Fi-enabled Arduino boards we found that the Nano was the best fit for the project due to its incredibly small footprint and features and capabilities we needed from our microcontroller selection. For example, we looked at the Arduino Portenta and the Arduino MKR Wi-Fi 1010. The Arduino Portenta has exponentially more RAM and processing power than the Nano, and though these specifications would be more than sufficient to have in for the project, it cost more than half of the SIMP’s allocated resources and did not have a sufficient number of Input/Output pins for the project’s purposes. The Arduino MKR Wi-Fi 1010 was the group’s second choice of microcontroller since it had similar specifications to the Nano but it had a strict input voltage limit of 5V where as the Nano had a much more flexible input voltage range of 4.5V to 21V when powered by it’s voltage input pin. Due to the high voltage nature of the project, the Nano was the preferred choice since the likelihood of the board being damaged by a sudden spike in voltage was much lower with its wider operational range.

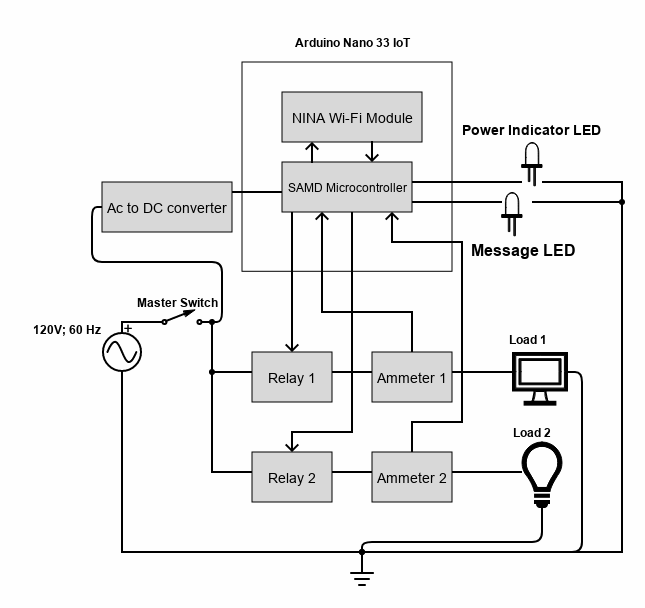


Fig. 4. Hardware system overall block diagram

The idea to buy the bare minimum parts the project needed from the Arduino Nano and assemble them on the printed circuit board was discussed but to due to part shortages brought on by the COVID-19 pandemic mission critical parts were either unavailable or would not be delivered in the time required for the project to be completed.

V. SOFTWARE DETAIL

The SIMP enables an end user the enhanced capability to alter its power use state and measure current, voltage, and wattage accordingly - depending on whichever device may be connected. So that we may alter and read said recordings, a platform must be developed to not only store measurements and plug ID information, but also keep record of current states of activity to be reported to the microcontroller so the device may act as intended by the end user.

The components necessary to ensure a stable platform for altering and recording states and measurements for the SIMP device include the creation of a: Database, Web Server, front-end application, and a general code-base required to interpret information on the side of the microcontroller used within the SIMP. Lacking in any part of these fundamental components would either lead to an insufficient means of storing electrical measurements, or simply a general inoperability of our manufactured device.

*A. Database*

The Database that we had our minds set on from the beginning was Mongodb a document style database solution for detailed and fast queries. We sort our data by newest sent document to oldest sent document, this allows us to query our most recent data as fast as possible for quick power analysis.

*B. Web Server*

When deciding on a webserver to host our website and APIs we had some difficulties choosing between platforms for their features, price, and performance. We went with Heroku for its relatively simple web hosting, github collaboration support, and its previous use from the engineers to help with setting up early on.

*C. Microcontroller*

The microcontroller software follows a simple loop that has it read the voltage coming from the sensor and calculates the true value using arithmetic. Once this value is acquired it then will send it to the web server using APIs in a JSON payload. Whenever the web server makes a request to toggle the relay, the Arduino receives the request through an API and then flips the signal to the relay which will then turn off power at the outlet. This was achieved using a Database entry that changes when the web server makes a request to change the power state of the outlets. The Arduino will check this database entry every cycle if it notices a change it will set the relay accordingly. It will also change the LED to light up when the relay is also turned off.

*D. Application*

When determining the front-end framework we would intend to use to interact with the SIMP device, several characteristics needed to be verified in order to ensure appropriate interoperability with the rest of our system components and web hosting services. Having already established using MongoDB for our Databasing, and Heroku for our Web Hosting, we realized that the front-end framework we build our app on would need to be easily integrated with both services and be fast enough to handle multiple asynchronous requests necessary to populate information on a user dashboard. With these prerequisites in mind, along with using Express.js as our back-end framework, we decided that we would move forward with using React.js to build our SIMP’s front-end user interface.

Building a React app, though more versatile and faster to use than many other UI frameworks, comes with challenges that must be dealt with by those developing the app. To ensure that the SIMP is able to be interacted with on a stable and functional platform, we would need to establish some form of Security - protecting clients privileged information such as passwords or plug info - established Routes - enabling a user cross site accessibility according to pre-established routes made on the back-end client - established API’s (Application Programming Interface) - enabling the web application the ability to get, post, or update values to any connected components - and establishing pages

VI. DEVELOPMENT PROCESS

Once the initial design of the SIMP system was agreed upon by the group, the team began testing each individual component of both the hardware and software sides of the system. For the hardware side, we started by using the currents sensors to find the current produced by creating a voltage difference across a resistor. This proved to be pointless as we could not safely create a high enough current to be reliably measured by our current sensor with only a function generator and a resistor. We found that the only way to produce a current high enough to be properly read by our current sensors was to tap into the electricity going into the hot side of the terminal by putting our current sensor in series with it. This allowed us to read the voltage of everyday electrical appliances like laptop chargers or lamps or televisions.

Once we were familiar enough with the current sensors we moved onto testing the Wi-Fi capabilities of the Arduino Nano by having it detect a local Wi-Fi network then connecting to it, as well as having it create its own Wi-Fi network that could be connected to via a Wi-Fi enabled device. We further tested the Wi-Fi connectivity by recording how long it took for a group member to tap a selection on their device that would send a signal to turn a relay module on or off. The switch from tapping a selection was almost instantaneous, so we could confirm our Wi-Fi section needed no further development.

Testing of the relays was challenging, as we could not seem to get the relay to change states when creating the control signal using a normal DC power supply, even when creating an input signal fifty percent larger than the components listed control signal strength. We believe that the reason for this is because the DC power supplies being used to generate the control signals did not rise to the appropriate voltage level fast enough for the relays to change states. This was solved by using the GPIO pins on the Arduino Nano which create digital signals when written to that had sharp enough pulses to trigger the state change in the relays. Following this discovery when testing the relays, code was written to the Arduino Nano to change the relays when given an input over Wi-Fi.

This lead to us also further testing the Arduino Nano’s capabilities of communication over Wi-Fi, connecting a local PC to the Arduino’s Wi-Fi network and having the PC simulate being the web server that would communicate to the Arduino it’s control signals to turn on or off certain relays and to flash the LED’s the different messages the would have to display.

With the Arduino being able to communicate with a device on the same wireless network, the logical next step was to have the Arduino be able to make and receive requests from the web server we would be hosting the SIMP software on. For this we would use REST api requests to and from the arduino. Our most important API is the POST API which sends all the metadata, sensor data, and time to the web server. This goes on loop for about 4-5 seconds while it tries to collect accurate data, in the meantime the program also is running other requests from the server including GET requests from the APIs to know the wanted power state of the outlet for it to turn on or off accordingly.

The structure of the project is situated such that all components relating to the backend are stored in the root of the project folder and all components containing the React app, pages, styling sheets and application routes are stored in a react app folder.

The login page employs the use of react, react-bootstrap, axios, universal-cookies, and practical css (derived from login.css). On startup, the page loads all form components relating to user input [Username, Password] and grants the user the ability to Register if applicable. Upon inputting user credentials, an axios call is parsed to the “Login” api with a client token holding the users password in a bcrypt hash to mongodb requesting a matching authority to the corresponding username. If the bcrypt token provided does not match the username in the database, the application will prompt an error and will not enable the user access to the users specific dashboard.

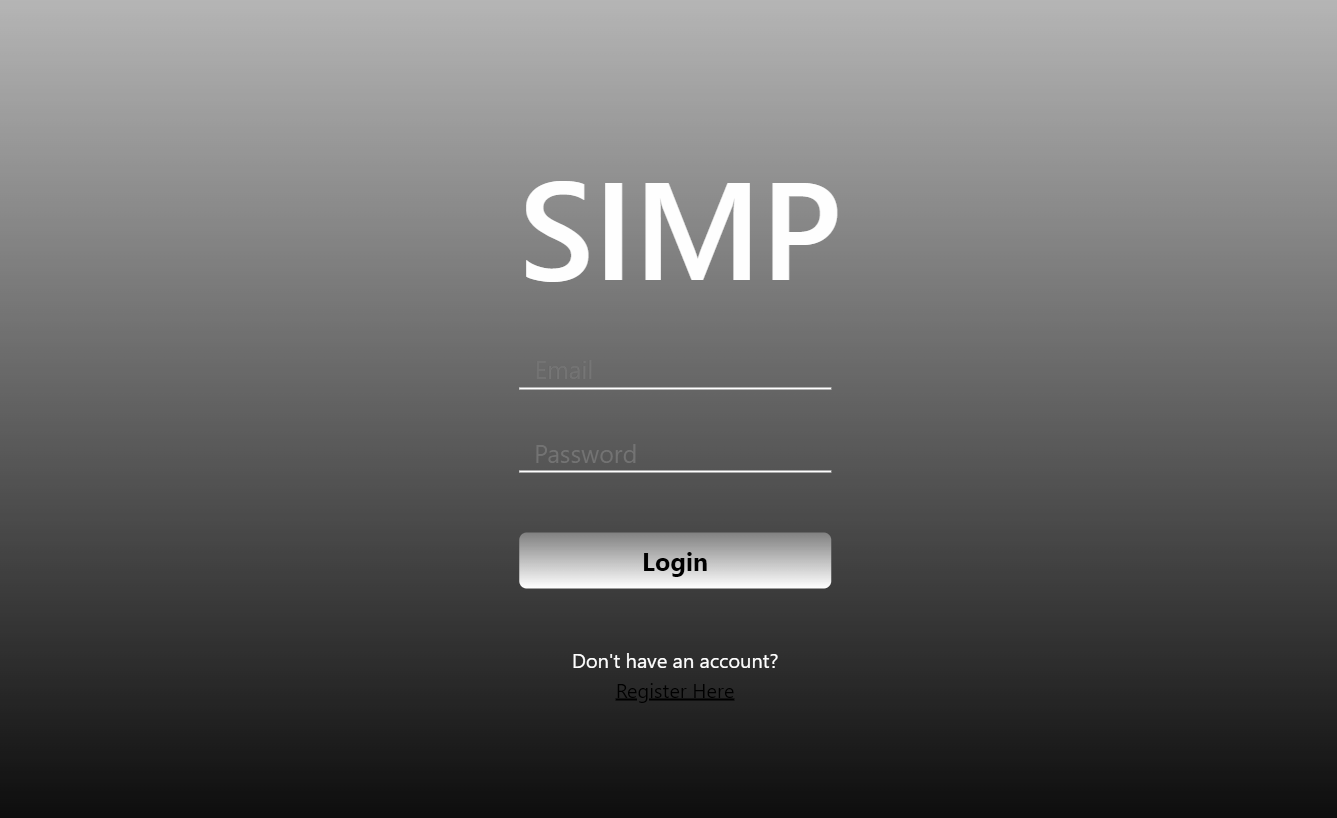


Fig 5. SIMP Login Page

The registration page, similar to the login page, prompts input forms that, upon submission, performs an axios call to the “Register” api and populates an entry into the database according to the pre-created user schema. On login, the user is redirected to a dashboard page that populates elements and charts according to the juxtaposed user id assigned to each document in the database.

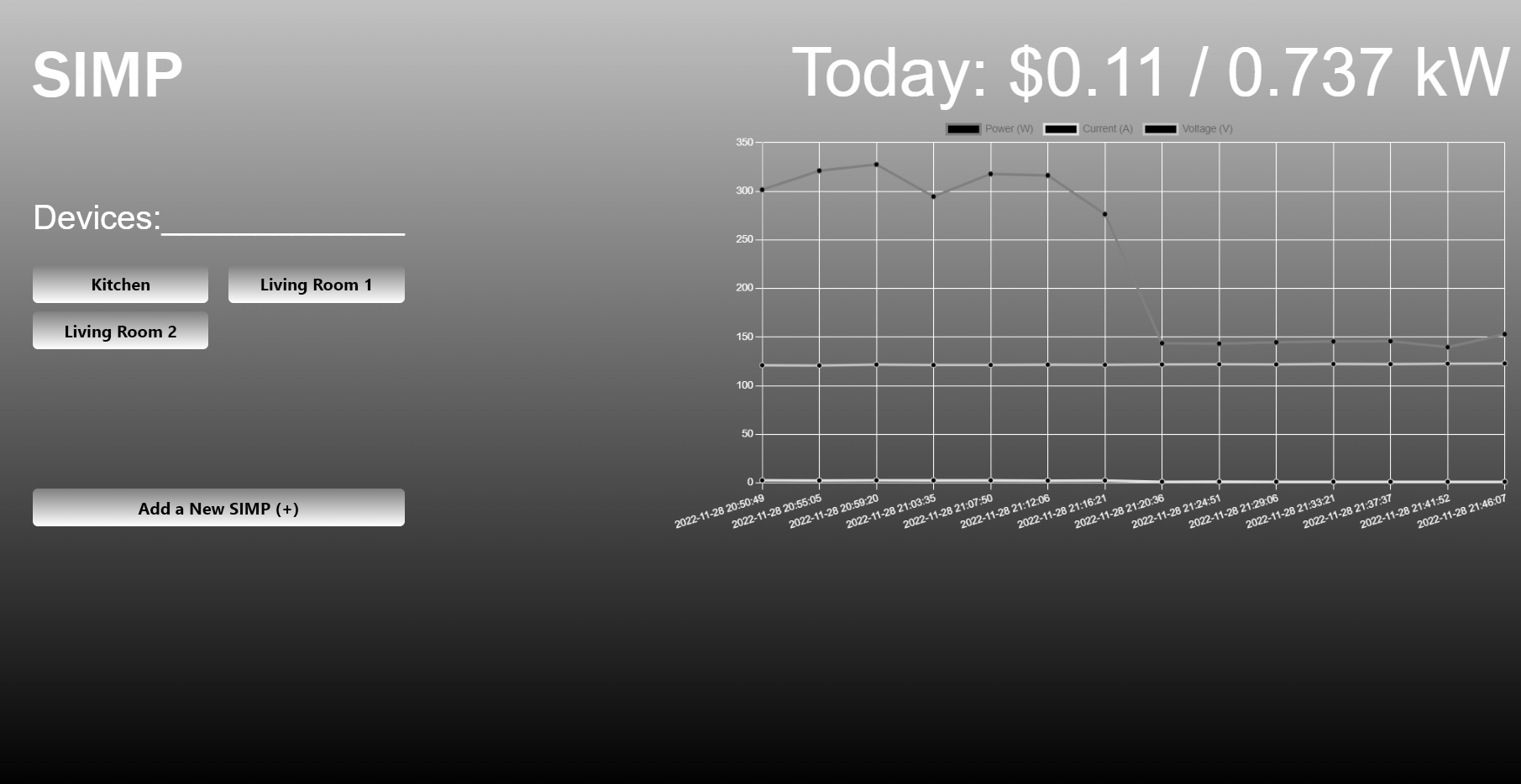


Fig 6. SIMP Dashboard Page

The dashboard is a collective access page composed of buttons (used to redirect to a devices corresponding control and read out page contingent on the device id), a line chart (pulled and populated from Chart.js library), and value elements that are derived from the measurement values stored in MongoDB. The value populated for cost is calculated by pulling the last 24-hours of documents that have been recorded to the database (17280 documents, where each document recorded at a 5-second interval between each measurement) into an array, iterating over each document, and performing arithmetic on each value to be recorded into a useState array within the app as follows:

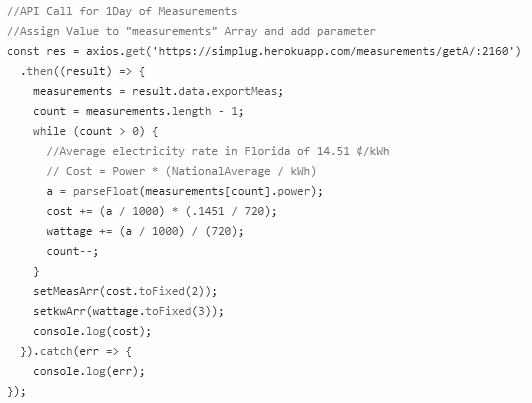


Fig 7. API Request to Fetch Data and Calculate Cost/Wattage

After adding a SIMP device to a user's account, corresponding to a user's ID, a user will have the ability to start monitoring device specific measurements, along with access control directly to the device. Interfacing to the device is established via the toggle switches on the device page. Upon clicking the toggle option, an axios call is created to update the document value of “State” corresponding to the device id either to 0 (OFF) or 1 (ON). The “State” value is eventually pulled by the SIMP’s microcontroller and updates the onboard relays accordingly to ensure appropriate power state of any connected device.

VII. PRINTED CIRCUIT BOARD DESIGN AND ENCLOSURE

*A. Printed Circuit Board (PCB)*

With the final schematic tested and drawn-up, all that was left was to arrange the components on a PCB. Doing so was easier said than done, as the group had to arrange them in a way so that they wouldn’t interfere with one-another in terms of footprint overlapping or wire trace overlapping or size.

The main problem we found when trying to arrange the components was that the components that had the 120V AC passing through them also needed to be connected to the Arduino Nano 33 IoT for an input or output signal and the 5V DC from the AC/DC converter for their operating voltage. This meant that any given relay or analog current sensor had three or more wire traces connecting to them that were not allowed to intersect or overlap. To remedy this, we places the 10-POS terminal block, the master switch and the AC pins of the AC/DC converter on the left-most end of the PCB while placing the Arduino Nano 33 IoT and DC pins of the AC/DC converter on the other end. This meant that we could focus first on routing the thick wire traces that would carry the 120VAC in the middle of the board and have the thin 5VDC routed around them anyway we could find.

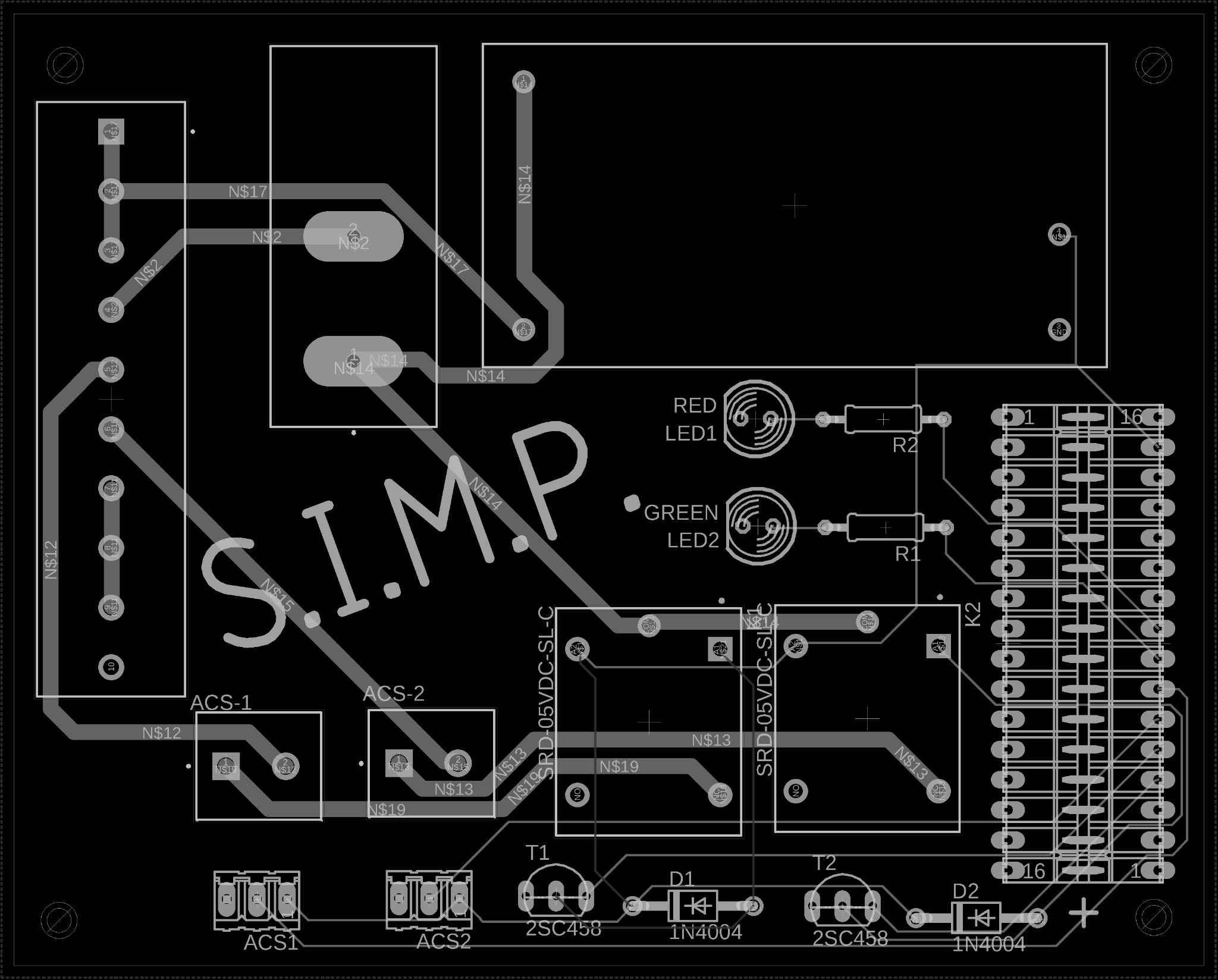


Fig 8. Final PCB Component and Wire Trace Layout

We also ran into the problem of a faulty footprint when using a generic through-hole style NPN transistor from EAGLE. The EAGLE footprint had the base and collector pins switched when comparing it to the outline of a proper through-hole style NPN transistor. This problem was one of the many reasons we went with a through-hole style for all our components, where if we were to have used surface-mount style components, we would have had to reroute, reorder and wait on delivery of a brand new PCB. We were thankful to our intuition and to be working with through-hole style transistors, which allowed us to fix the problem by bending the legs of the base and collector pins to their rightful ports in the PCB and soldering them accordingly.

*B. PCB Enclosure*

With the final PCB designed, shipped, soldered and tested, we began the process to make a box that would signify our final footprint specification of less than 3x5x5 inches cubed. To get an idea of where the female outlet plug would fit best in the box, we took a box used to ship the PCBs and cut sections out from it until the female plug was no longer rubbing against any PCB components.

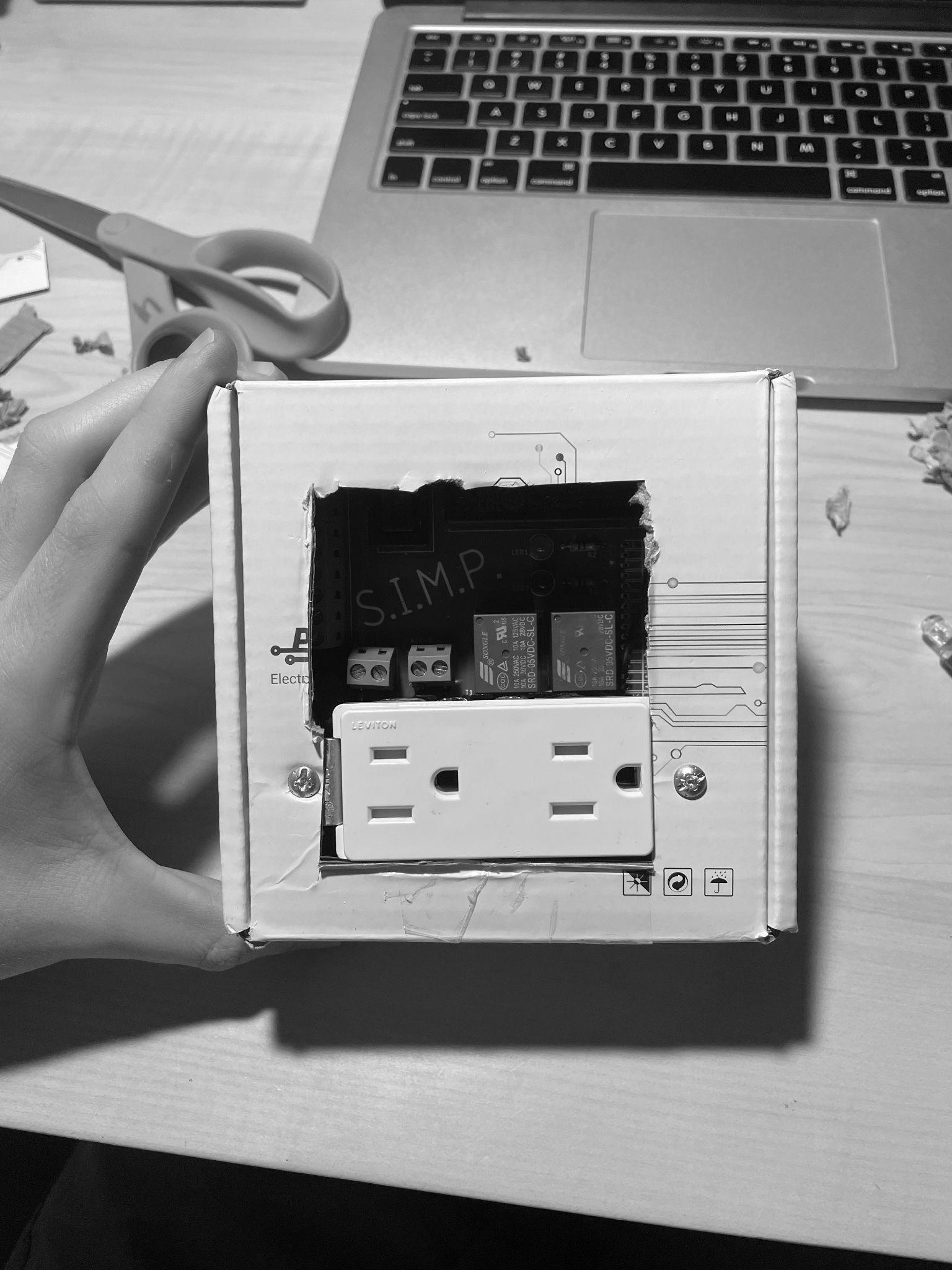
**

Fig 9. Cardboard PCB Enclosure

With the final PCB designed and completed, a 3-D printed shell was created to be the enclosure for the SIMP hardware. The gerber file of the PCB was placed into a 3D printing software and from there we were able to model a box that would fit within our deliverable measurement parameters while providing holes needed to access the master switch, female outlet ports, and LEDs. A figure of the box in the 3D printing software is shown below.

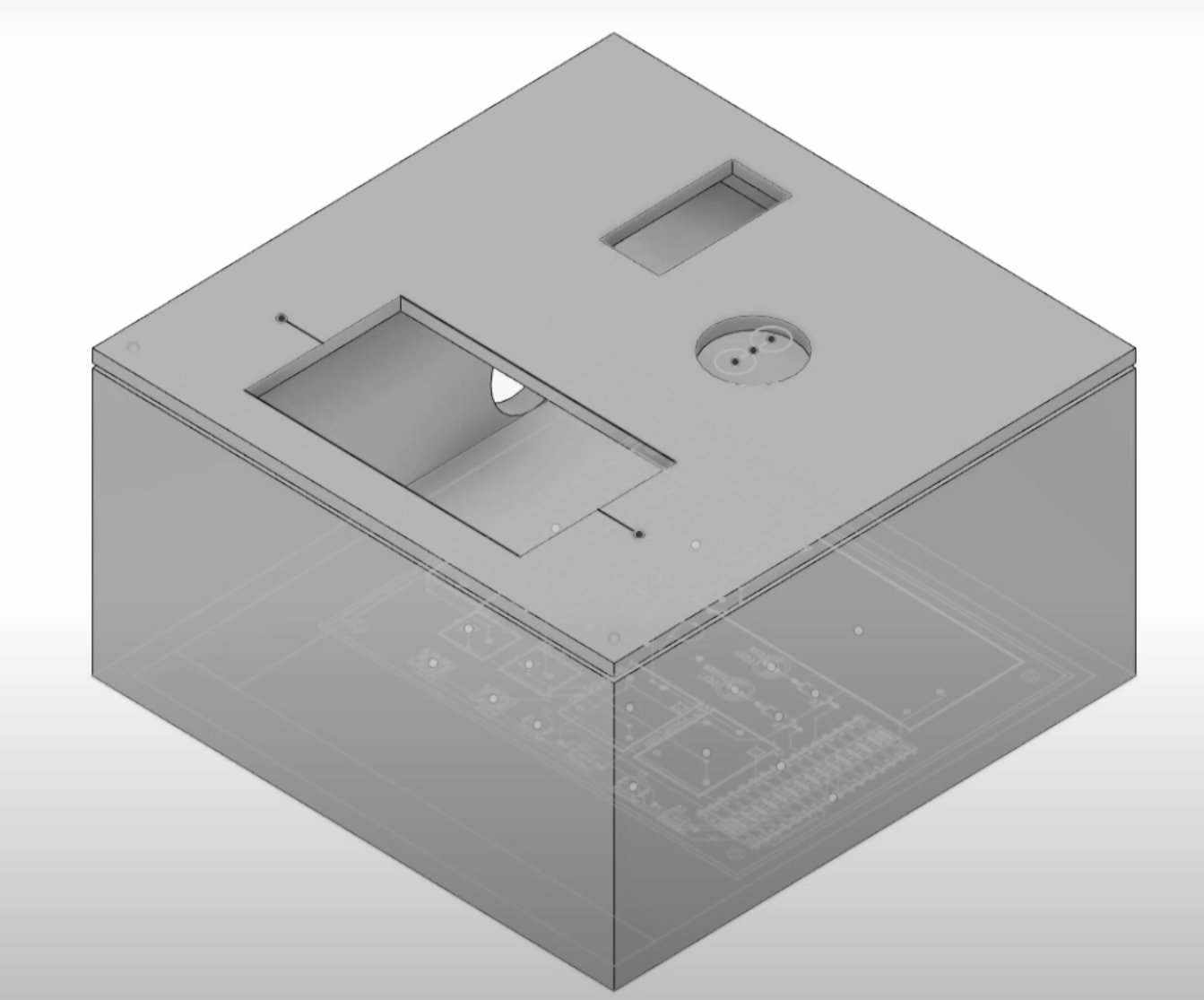


Fig 10. 3-D render of the SIMP enclosure created in Blender360

THE ENGINEERS

|  | **Austin Raynor** is a 24 year-old graduating Computer Engineering student. He aims to pursue a career in the home automation / artificial intelligence field with his understanding in electronics and programming. |
| --- | --- |
|  | **Edin Durgutovic** is a 24 year-old graduating Electrical Engineering student. He hopes to pursue a career in the electric automotive industry with a passion for performance racing keeping him interested in the future of technology. |
|  | **Anthony Perez** is a 24-year student graduating in Computer Engineering. With an adept understanding in both programming and embedded-systems, he aims to pursue a career in R&D or Software Engineering. |
|  | **Jose-Valentin Sera-Josef** is a 24 year old graduating Electrical Engineering student. He hopes to pursue a Masters with Thesis degree in Electrical Engineering while doing research into Virtual Reality and Human-Computer Interaction. |

ACKNOWLEDGEMENT

The authors would like to acknowledge Dr. Samuel Richie, Dr. Lei Wei, Dr.Xun Gong, Dr. Joseph LaViola Jr and Dr. Suboh Suboh; University of Central Florida.

REFERENCES

[1] Arduino Nano 33 IoT. Arduino Docs | Arduino Documentation. (n.d.). Retrieved November 1, 2022, from https://docs.arduino.cc/

[2] 20W AC-DC PCB-mount green power module IRM-20 - Mean Well. www.meanwellusa.com. (n.d.). Retrieved November 1, 2022, from https://www.meanwell.com/Upload/PDF/IRM-20/IRM-20-SPEC.PDF

[3] Songle SRD Relay. Circuit Basics. (n.d.). Retrieved November 1, 2022, from https://www.circuitbasics.com/wp-content/uploads/2015/11/SRD-05VDC-SL-C-Datasheet.pdf

[4] Allegro MicroSystems Inc. (n.d.). ACS712: Fully integrated, hall-effect-based linear current sensor IC with 2.1 kvrms voltage isolation and a low-resistance current conductor. ACS712: Hall-Effect-Based Linear Current Sensor IC. Retrieved November 1, 2022, from https://www.allegromicro.com/en/products/sense/current-sensor-ics/zero-to-fifty-amp-integrated-conductor-sensor-ics/acs712

[5] RB1 Series Rocker Switch DataSheet. E-Switch. (n.d.). Retrieved November 1, 2022, from https://sten-eswitch-13110800-production.s3.amazonaws.com/system/asset/product\_line/data\_sheet/85/RB1.pdf

[6] Keyes SR1Y Relay Module Circuit. Techydiy. (2017, February 28). Retrieved November 1, 2022, from http://www.techydiy.org/keyes-sr1y-relay-module/