Power Quality Monitor and Sub Meter System

ECE 445 Design Document - Spring 2024

Team #30
Nicole Viz
(nviz2@illinois.edu)
Soham Manjrekar
(sohammm2@illinois.edu)
Roshan Mahesh
(roshanm2@illinois.edu)

TA: Surya Vasanth
Professor: Jonathon Schuh

May 1, 2024

Abstract

This document provides details, diagrams, and schematics for the design and implementation of our project, which is a power quality monitor and sub meter system. Beyond our design and implementation, we provide information on the parts we used, the total cost, the requirements for each subsystem along with how we verified it, a weekly schedule of when we completed each part of the project, and an analysis of the safety and ethics of our project.

Table of Contents

1 Introduction

In this section, we explain the problem that we are attempting to solve, and we highlight its relevance in the field of power electronics today. We also illustrate our solution to the problem, our visual aid, and our high-level requirements along with its relation to the project so that others can understand our planning and design process. Finally, we introduce an overview of our subsystems along with a high-level block diagram.

1.1 Problem

In the rapidly evolving field of power electronics and energy technologies, maintaining consistent and high-quality power distribution and energy usage is critical for residential and commercial buildings. Using submeters can create energy savings, lower operating costs, increase building efficiency and reliability, and improve occupant comfort. However, devices today can be cost-inefficient, complex to operate and to read, and they may lack real-time insights. These shortcomings lead to difficulty in meeting recent sustainability efforts, and as such, an innovative solution is needed.

For example, a common residential electrical submetering product is from Byram Labs. It's a single phase submeter that costs $420, doesn't provide prior electrical usage data, doesn't provide a way to perform data analysis, and doesn't monitor power quality or notify owners of poor power quality for the following metrics: large voltage changes/irregularities, power outages, harmonic disturbances. This can cause many problems, especially for large commercial applications such as hospitals. In Japan, there was a study done on 13 pieces of medical equipment and found that voltage dips caused 7 devices to stop for about 0.5 seconds, which caused some of the 7 devices to start an automatic reboot. Without having the electrical submeter also measure for power quality, medical equipment could get damaged or patients could suffer from ineffective medical care.

1.2 Solution

For our project, we designed and constructed an improved device that monitors power quality and acts as a submeter to its loads – a device that is cost-effective, and has high-fidelity and real-time data acquisition. Our project solves the problems listed above by combining a power quality monitor along with a submeter in a cost-effective manner that stores real-time data and loads the data to a database that can be accessed through a website. The website also allows users to be alerted of power quality issues, provide access to prior data, and showcase voltage and power waveforms.

Over the course of the build process, we produced a prototype that analyzes voltage, current, and power from an outlet and examines the power usage of a connected load. To do this, the device

connects into an outlet, then steps down the voltage and current to be sensed proportionally with the ICs, and sends measurements through the microcontroller to the database. The control unit of the device is powered by a converter circuit with a linear regulator, and in backup cases, with a battery subsystem. We encased all circuitry and hardware within a box to showcase a formal final product. A visual representation along with our finalized product can be found in Appendix A, Figure 1.1 and 1.2.

1.3 High-Level Requirements

For our project to be considered successful, there are various high-level criteria that must be met. Our device should be able to perform the following tasks:

- **I.** Sample a single-phase input for its voltage and current (within 10% accuracy).
- **II.** Use a microcontroller to process the voltage and current samples and calculate apparent and real power. Store data points to a cloud database (once every 10 seconds). Instantaneous waveforms will also be displayed on a screen.
- **III.** Notify the user (within 5 seconds) of any disturbances in measurements outside of a set tolerance (5%) and of any failures.

The high-level project functionalities are intricately interconnected with the overarching purpose of providing comprehensive monitoring, analysis, and notification capabilities for electrical systems. By accurately sampling voltage and current from a single-phase input within a 10% accuracy threshold, the device can spot problems like overloading. We must sample a single-phase input for its voltage and current because it forms the foundation for understanding and analyzing the electrical system's behavior. The microcontroller is also integral to our project as it enables us to process data from the ICs, perform calculations, and send the data to a database. Without a microcontroller, interpreting the data would be impossible.

Additionally, storing data points in a cloud database every 10 seconds ensures its availability and integrity for later review, enabling users to track trends and identify potential issues. Displaying waveforms on-screen helps users visualize and interpret the data in real-time, enhancing their understanding of the electrical system's performance. Furthermore, a notification system for failures is essential, as it alerts users to power outlet issues promptly. This is crucial for preventing potential damage to equipment that relies on wall power, ensuring uninterrupted operation and minimizing downtime. Successfully implementing these three high-level requirements will yield a rapid, functional, and advanced solution for power quality monitoring and submetering, providing users with actionable insights and prompt alerts to maintain system reliability and efficiency.

1.4 Subsystem Overview

As shown in our top-level diagram in Appendix A, Figure 2, we have four subsystems: step down, control unit, power supply, and database. We start off with getting the single phase mains power from the wall outlet. We take this 120V AC to both the power supply and step down unit. In the Power Supply, this AC waveform is fed to a transformer that converts the voltage to a lower one, subsequently, full wave diode rectifier followed by a large capacitor to convert the voltage to a near DC waveform. After the capacitor a 5V linear regulator keeps the voltage a constant 5V. From there the 5V is used to power the battery charging circuit, or is further converted down by yet another linear regulator to 3.3V which is sent to the Control Unit to power the measurement ICs and the microcontroller.

In the Step Down Subsystem, a transformer is in parallel with the wall to down convert the voltage, following the voltage transformer, a voltage divider made of two resistors is present to further take the voltage down to meet our requirements. Another transformer in the Step Down Subsystem shorts across the hot terminal, this is the current transformer, it steps current down from a maximum of 15A to a safe value (defined in our RV table) for measurement . The output of the Step Down Subsystem goes to our power quality and power metering IC in the Control Unit. The power quality IC is an ADE9430, which measures dips and swells, interruptions, magnitude, harmonics, and much more. The power metering IC is an ADE7953, which measures active power, reactive power, apparent power, sampled current, and RMS voltage. The signals that are outputted from these 2 ICs are sent to the ESP32 via Serial Peripheral Interface (SPI).

SPI is a communication protocol that the ESP32 uses to take in information for processing. We then connect the ESP32 to Wi-Fi, calculate apparent and real power, then send the data to our database subsystem via HTTP requests. The database subsystem consists of a MySQL database that is hosted by Google Cloud Platform. The data that is held in the database is then graphed and put onto a website. The data is sent from the database to the backend web server via Query requests, and the data is then sent to the frontend via websockets. Once the data is at the frontend, we use the Chart.js library, which is a JavaScript library, to graph the data in real-time.

2 Design

2.2 Physical Design

2.2.1 Design Procedure

To house the hardware of our device, we chose a simple aluminum 200 mm by 300 mm project box, shown in Appendix A, Figure 1.2. This was chosen for its lightweight nature and durability, and because it was open on one side, allowing us to easily connect and disconnect components when needed. To make sure everything fit snugly, we 3D printed rails for our two PCBs to snap into, then used command strips to adhere them to the box, along with a battery pack and one

transformer that we were unable to fit onto a PCB. For safety, we also made sure to connect the earth ground of the wall outlet to the earth ground pin of our panel-mounted power inlet port, and then to multiple contact points on our enclosure.

2.2.2 Design Details and Decisions

a) Interfaces
    i) Wall Outlet
        1) For input, we panel-mounted a three-pronged female power inlet port on the side of our box and used a male-to-female power cable to connect from the wall outlet to this port. For output, we panel-mounted a typical wall outlet configuration, which could receive any basic power cable.

b) Design Decisions
    i) All decisions regarding our enclosure were made with the utmost concern for safety. This included frequently checking continuity between all earth ground contacts, as well covering all points that may carry wall voltage with electrical tape or shrink wrap.

2.3 Power Supply Subsystem

The power supply unit lays the groundwork for the functionality of the rest of our device; without power, none of the other subsystems would work. Its main purpose is to take in wall voltage and current and output 3.3 V to the necessary components of the control unit, and to run on battery power in the event of wall power failure. To do this, it consists of a 5 V DC converter, a battery pack, a battery charging circuit, a diode-OR circuit, and a 3.3 V linear regulator.

2.3.1 Design Procedure

Since wall power was being directly fed to this unit, all components were chosen with specific attention to maximum voltage and current ratings. To begin, the ST-4-12 power transformer was chosen due simply to its ability to handle wall voltage 120 V AC at its input and output at a level around 12 V. The remaining components in the 5 V AC-DC converter were chosen based on our existing familiarity with their operation from other courses (namely ECE 469) and their immediate availability through self-service. All together, this converter block consisted of the aforementioned transformer, a full wave rectifier with power diodes, a bulky output capacitor, and an LM309H 5 V regulator. We also included banana jack test points at the secondary of the transformer, at the output of the rectifier, and at the output of the regulator to ensure easy debugging.

After a discussion with TAs, we chose to use two 18650 3.2 V lithium iron phosphate batteries in series, which were desirable for their safety compared to typical lithium ion batteries. We considered two different battery charging ICs, and we ended up ordering both of them at the beginning of the semester. Eventually, we chose the MCP73831T chip because we found its

datasheet to be more helpful, straightforward, and easy to understand. We then used a diode-OR circuit consisting of two Schottky diodes (for their low forward voltage drop) to choose between wall power and battery power. For this step, we had seen that a group from a previous semester had used a more complicated selection circuit consisting of many more switches and logic, but after some initial testing, we found that our two diodes worked fine as long as battery voltage remained equal to or less than 5 V. For our purposes, this was suitable, and it allowed us to maintain our desired level of simplicity. Finally, we selected a LM3940IT 5 V to 3.3 V regulator to provide our final output voltage because it was readily available in the lab. In hindsight, we realized this regulator got hot after prolonged use of our device, and likely could have performed better if we had additionally added a heat sink. No safety issues were encountered, though, and it performed sufficiently well to achieve our requirements.

2.3.2 Design Details and Decisions

a) Interfaces
   i) Wall Outlet
      1) The power supply unit connects to the wall outlet through a female 3-prong power inlet port mounted on the side of the enclosure. It steps down this voltage as described in the previous section using a transformer, a voltage converter, a rectifier, and various regulators.
   ii) Control Unit
      1) The ESP32, Power Quality IC, and Power Metering IC all require 3.3 V to be powered and functional.

b) Detailed circuit diagrams for each section of the power supply unit can be found in Appendix A in Figures 3.1 and 3.2. The resistor and capacitor values were mainly chosen from datasheet recommendations and requirements.

2.4 Step Down Subsystem

The Step Down Subsystem plays a critical role in the overall functionality of the project by converting voltage and current from the wall outlet to levels compatible with the metering and monitoring ICs, namely the ADE7953 and ADE9430. It comprises two main components: a Voltage Divider circuit, which steps down the voltage to <3.4V, and a Current Transformer (B82801B0925A200), which reduces the current to <75mA. These components are designed to accommodate potential current limits, although the ICs' only limitation is on voltage. Despite this, the Current Transformer still serves to provide an isolated current input to the measurement ICs, allowing for accurate current measurements using the onboard PGA.

2.4.1 Design Procedure

Originally, the design of this subsystem involved the utilization of resistors, a current divider, and a voltage divider to transmit stepped-down waveforms to the Control Unit. However, this approach encountered significant challenges, particularly in managing power distribution. Under

high-load conditions, where substantial current flowed across the resistor, power consumption soared, reaching nearly 1.8 kW. Implementing such a design necessitated the acquisition of bulky and costly resistors, rendering it impractical. In response to these challenges, we opted for an isolated approach, leveraging current and voltage transformers. This transition to an isolated approach also addressed concerns regarding the direct connection to the wall outlet, enhancing overall safety and reducing potential risks of electrical hazards. By utilizing current and voltage transformers, we were able to effectively isolate the electrical signals, ensuring that the measurement and monitoring ICs received accurate and reliable input data without exposing the control unit to high-voltage environments.

Furthermore, the adoption of transformers enabled us to achieve a more compact and cost-effective design compared to the initial resistor-based approach. By leveraging the inherent characteristics of transformers, we could efficiently step down voltage and current levels while maintaining signal integrity and minimizing power dissipation. This not only optimized the performance of the Step Down Subsystem but also contributed to the overall efficiency and reliability of the project.

2.4.2 Design Details and Decisions

a) Interfaces
    i) Wall Outlet
        1) Direct input 120V AC from the outlet to the subsystem to be stepped down
    ii) Control Unit Subsystem
        1) Directly to VAN-VAP and IAN-IAP
        2) Should deliver less than 3.4V.
b) Design Decisions
    i) In summary, the design decisions for the Step Down Subsystem resulted in the use of an isolated configuration using current and voltage transformers. This decision not only addressed power-related challenges but also enhanced safety and contributed to the compactness and cost-effectiveness of the subsystem design. Detailed circuit diagrams are included in Appendix A, Figure 4.

2.5 Control Unit Subsystem

The Control Unit subsystem is constructed of the two measurement ICs (ADE7953 and ADE9430) and the ESP32 Microcontroller (ESP32-S3-MINI-1-N8). The ICs are where the voltage and current signals are converted into digital signals. Stepped down voltages are delivered to both measurement ICs which then pass through a PGA and an ADC followed by a digital block where all pertinent DSP calculations are made. Data then will be passed through to the Microcontroller via SPI protocols to be outputted to a database. We will leverage the SPI interface on both the measurement ICs using the MISO pins as an output to stream data out from

the ICs, with SCLK and MOSI and CS (Chip Select) being inputs. On the ESP32 these are all GPIO pins specifically pins 10-13 and pins 23-27 , we will use these to generate a clock signal (SCLK), stream data from the ESP32 to the ICs (MOSI), stream data from the ICs (MISO), and enable/disable communication between the two sections of our subsystem. Once on the ESP32, the data will be sent via Wi-Fi using HTTP requests to the Database subsystem for processing and delivery to a web app. The ESP32 itself will be powered from the Power Supply subsystem at 3.3V.

2.5.1 Design Procedure

In our pursuit of deeper insights into power quality and metering, we intentionally veered away from conventional methods involving hall effect sensors and traditional voltage sensing circuits. Instead, we opted for cutting-edge ICs, namely the ADE7953 and ADE9430. These ICs are specifically designed for submetering and power quality analysis, catering to professional applications. Despite their advanced capabilities, their interface via SPI—a widely documented and standard form of communication between ICs and controllers—provided us with confidence in navigating the unknown. We believed that the potential rewards of the data they could offer outweighed the risks associated with exploring unfamiliar territory. Detailed block diagrams of these ICs are included in Appendix A, Figures 5.2 and 5.3, along with the complete design of the control unit in Figure 5.1 for reference.

The design procedure for the Control Unit subsystem was seemingly straightforward, devoid of intricate power management complexities aside from the input voltage from the power supply and the stepped-down voltage and current from the Step Down Unit. Initially, the process involved meticulous examination of the datasheets for the ICs and microcontroller, ensuring precise alignment of PCB traces with the correct pinouts, and appropriate strapping or decoupling of pins. However, a significant oversight in not submitting the PCB within the first or second wave of orders contributed to the failure of the Control Unit Subsystem. Despite its seemingly simple design compared to other subsystems, it proved to be the most problematic.

Efforts to debug the board included redesigning the board for the fourth wave of PCB orders wherein five of the PCBs were sacrificed in an attempt to connect to the microcontroller. Additionally, attaching wires directly to the pads of the microcontroller surface mount and running them to the development board on a breadboard were employed, but these efforts also proved unsuccessful.

Multiple factors could have contributed to the board's failure. It is plausible that excessive voltage and current from the wall supply led to the microcontroller burning out. Alternatively, inadequate strapping or decoupling capacitors for stable operation may have been a factor. However, attempts to replicate the development board microcontroller's setup on the production board yielded similar challenges. Reflecting on the experience, if given another opportunity, our

initial step with this subsystem would involve constructing a digital model to simulate data signals. The absence of such a model left us without a clear starting point for the debugging process, making it challenging to pinpoint whether issues lay in the code, the PCB, the hardware, or intermediary components like resistors and capacitors.

2.5.2 Design Details and Decisions

a) Interfaces
   i) Step Down Subsystem
      1) Voltage and Current Signals are taken in from the Step Down subsystem
      2) SPI Data is passed through via SPI from the ICs to the ESP32

   ii) Power Supply Subsystem
      1) The 3.3V that powers the ESP 32 and measurement ICs comes from the Power Supply subsystem

   iii) Database Subsystem
      1) Data is passed to the Database subsystem over Wi-Fi and HTTP connection

b) Design Decisions
   i) In selecting our current and voltage measurement system, we deliberately opted for a more costly and complex solution. Our decision was driven by the desire to explore the advanced capabilities offered by newer IC technology. We anticipated that these measurement ICs would provide us with a wealth of data that traditional sensors or circuits may not capture. Importantly, our project scope did not impose any budget constraints. Consequently, our goal was to develop a device capable of meeting industry-level submetering and monitoring requirements, and we believed that these ICs would enable us to achieve precisely that objective. Moreover, we aimed to package our final device in a sophisticated manner, aligning with our commitment to delivering a high-quality, professional-grade solution.

2.6 Database Subsystem

The database subsystem is a full-stack web application that displays graph data, specifically the voltage and current of the power outlet and the load. The web app also notifies the user when there are disturbances or failures with the power outlet voltage in real-time so that a building manager or electrician can fix the problem as early as possible. The backend is written in JavaScript using Node.js and Express.js libraries to create a server and send Query requests to the database. The frontend is written in JavaScript as well by using the React.js library to create an interactive web application. To create real-time graphs, we used websockets to get data from the backend to the frontend instantaneously and the Chart.js library, which allows us to read and

plot data. Lastly, we used a MySQL database hosted on Google Cloud to store the data from the ESP32. An image of how the data is stored in our database can be found in Appendix A, Figure 6.1. Additionally, a flowchart of how the data is transmitted throughout the system can be found in Figure 6.2.

2.6.1 Design Procedure

During the course of this project, we immediately knew that we wanted to use JavaScript libraries and Google Cloud to create the web application and database; however, we weren't sure of how to create graphs that would refresh with new data every minute. With prior experience using Matplotlib, which is a Python library, we considered using it to create the graphs on our website. Once we finished creating the database table and inserted simulated data, we sent the data to our Python file to graph it. However, we were unable to get the graph to appear on the website. After many attempts at debugging and reading documentation online, we found out that applying a Python library to a JavaScript application can create compatibility issues. From here, we pivoted to JavaScript libraries, specifically Chart.js, since compatibility wouldn't be an issue. After spending some time debugging our new code and understanding how the library works, we were able to get the graphs for voltage, current, apparent power, and real power to appear on the website. Another benefit that we didn't know about this library is that it's also an interactive graph, meaning that a user can use their mouse to hover over data points to read the value.

After this, we looked to implement the notifications feature. This feature provides a notification to the user when there's a deviation from the standard 120V AC wall voltage by 5% or greater. Since we wanted this notification to appear within 5 seconds, and to persistently stay on the website so that the notifications can form a log of prior issues, we decided to have the backend server query the database for data once every second. Additionally, we decided to use the local storage that every web browser has built-in to keep the notifications in the notification page forever. However, we found many issues with these design decisions. First, by querying the database once every second, we were missing out on data points that should've been shown on the notification page. Second, we were unable to reliably get the notification to persist in the logs since a user could switch browsers. For example, if the user uses Internet Explorer to go to the website, but then they switch to Google Chrome, the notifications wouldn't show up in Google Chrome since local storage is browser specific. To fix this, we switched from using local storage to searching the entire database for all faulty voltages. Since this must be done a lot quicker than once per second, we switched to using websockets which has the ability to get data from the database at a rate of 5 requests per second. By making these design changes, we were able to get the notifications page to update instantaneously and have older notifications persist in the form of a log.

2.6.2 Design Details and Decisions

a) Interfaces
   i) Control Unit
      1) HTTP connection from ESP32 to Google Cloud Platform MySQL database to receive data
      2) SQL Query to get data from the MySQL database to the backend server. The data arrives to the
      3) Websocket to get data from the backend server to the frontend so that the data can get graphed or put in notifications page

A clear flowchart in Appendix A, Figure 6.2 shows the movement of data and how that was accomplished. A few other notes about all the interconnections is that the data from the ESP32 gets correctly put into MySQL columns. These columns are data id, outlet name, load name, measurement time, voltage, current, apparent power, and real power. The data is put into the correct column due to the SQL statement that is used to send the data from the ESP32: INSERT INTO PowerData (data_id, outlet_name, load_name, measurement_time, voltage, current, apparent_power, real_power) VALUES (%d, %s, %s, %s, %d, %d, %d, %d). In this statement, we state the column names and insert the data in the right order. Additionally, PowerData is the name of the MySQL database table. Another item to note is that when the backend server gets the data from the database, it's in the form of an array. The backend server makes the following query: SELECT 'voltage' FROM PowerData. The server then gets all the voltage data from the database, which looks like this: voltage: [120.45, 117.89, 122.21]. Since we get the data as an array, all we have to do is iterate through it to get all the data points.

b) Design Decisions
   i) When looking at alternative approaches to the design, there are many different ways to create a full-stack web application and there are many different database providers. For example, web applications can also be made using Python libraries and platforms such as Django, using PHP, or using C#. These are all options that are commonly used in industry practices as well. For database providers, there's many different options available as well such as AWS, Oracle, and MongoDB. For graphing the data, there's also plenty of libraries available. For example, Python has the Matplotlib library for easy graphing using minimal code, and it has plenty of documentation online.
   ii) Despite all these options, we chose to use JavaScript for the website, specifically using Node.js, Express.js, and React.js as the libraries, MySQL through Google Cloud Platform for the database, and Chart.js library to graph the data. We made these design choices since one of our group members has experience using JavaScript and Google Cloud to create full-stack web applications through personal projects, internships, and CS411. None of us had any experience with

Chart.js; however, we chose to use it since integrating another JavaScript library with a JavaScript project would be a lot easier. Additionally, there's plenty of documentation online on how to use the library. By using technology that our group was already familiar with, we knew that we would have the website and database working by the demo deadline. Due to this time constraint, these technologies became desirable to use.

2.7 Verification

2.7.1 Power Supply

Table 1.1 in Appendix B displays the necessary requirements and verification steps associated with the power supply unit. All of these requirements were met and shown during our final demo (three of them physically in real time, and one of them discussed mathematically). Most importantly, Figure 7.1 in Appendix A shows the achievement of our first requirement: an output of 3.3 V DC (green waveform). This was verified using multimeter probes at the output of our power board. Our second requirement was verified again simply by unplugging our device from the wall and ensuring that 3.3 V DC was still available at the output.

The third requirement, which involved ensuring the device could run on battery power for at least 10 hours, was not verified physically due to safety concerns. In order to test this, we would have had to let our device run for 10 hours while someone constantly monitored it, which was just not realistically feasible. Instead, we demonstrated mathematically that it should be possible. Our battery pack has a capacity of 1500 mAh, so we simply needed to show that the battery discharge current was less than or equal to 150 mA.

$$\frac{1500\ mAh}{150\ mA} = 10\ hours$$

Using a current probe, we found that the discharge current was typically on the order of 70 mA, ensuring that if need be, our device could run for at least ten hours off of battery power. It is important to note here, however, that this is something that should only occur when absolutely necessary, as such deep discharges of LiFePo batteries are not recommended to be frequently performed

Our final requirement involved ensuring that our charging circuit would not charge our battery pack past 80% state-of-charge, again for safety reasons. We verified this by connecting the pack to the charging circuit for at least an hour, and probing the battery terminals every fifteen minutes to ensure that the voltage was within the accepted range:

$$0.8 \times (2 \times 3.2\,V) = 5.12\,V$$

Appendix A, Figure 7.2 shows the battery voltage at several points in time during this verification procedure, proving that it was maintained at about 4.6 V to 4.7 V (well within our requirement).

2.7.2 Step Down

The verification process of our Step Down Subsystem is relatively straightforward. Under the condition of no additional load connected to the box besides powering itself, we observe specific waveforms as shown in the accompanying figure. Among these waveforms, the first three demonstrate the fulfillment of our requirements as outlined in the verification tables shown in Appendix B, Table 1.2. The yellow waveform represents the input voltage, measured at 120V RMS, while the purple waveform depicts the output of the step-down voltage transformer, measured at 1.6V. The blue waveform illustrates the signal inputted to our current transformer.

The first requirement, pertaining to voltage step-down, is evidently met, as the purple waveform clearly falls below the 3.4V threshold specified in our requirements. However, the second requirement warrants further explanation. In our setup, the current transformer is shorted across the input, and the observed blue waveform represents the input current. Due to the absence of proper probe points, a limitation discussed in our design procedure, we are unable to directly measure the current output. Nevertheless, leveraging the known 1:200 turns ratio of the current transformer, we can infer that the current on the transformer secondary is approximately 250uA, given the observed input current of ~50mA in the absence of extra load. This deduction is based on the principle that current on the transformer secondary is divided by 200 from the primary current due to the transformer's turns ratio.

2.7.3 Control Unit

We were unable to meet this requirement. As outlined in our design procedure we faced challenges and potential design limitations that would prevent us from meeting this requirement. Despite the Step Down Subsystem functioning as intended and current flowing through the device, we faced significant issues with the microcontroller. Attempts to program the microcontroller using the D+ and D- pins connected to probe points proved unsuccessful. Surprisingly, the microcontroller was not even recognized as a device by our computers, thwarting our efforts to program the chip. This technical hurdle significantly hindered our ability to accurately transmit voltage and current data from the measurement ICs to the ESP32, ultimately leading to the inability to meet the specified requirement, shown in Appendix B, Table 1.3.

2.7.4 Database

For the database subsystem, we had 3 requirements: the database must store new data once every 10 seconds from the ESP32, the website must show a notification within 5 seconds when there's any wall voltage disturbances greater than 5% of 120 V AC, and the website must show graphs of voltage and current data that is refreshed once every minute. Based on these requirements, we had a set of verification procedures that we ran to ensure that the requirements were met. These requirements and verification procedures can be found in a table at Appendix B, Table 1.4.

For the first requirement, we had to improvise the verification procedure since we couldn't get the ESP32 to work. So, we inserted data straight into the database from the Google Cloud Shell using the following SQL statement: INSERT INTO PowerData (data_id, outlet_name, load_name, measurement_time, voltage, current, apparent_power, real_power) VALUES (%d, %s, %s, %s, %d, %d, %d, %d). We then started a clock to see how quickly the data would show up in the database. The data was inserted into the database in 0.08 seconds and all data points that we inserted showed up in the database correctly. This time is significantly faster than the 10 seconds that we were aiming to get. While this isn't coming from the ESP32, we can also theoretically guarantee that the data will reach the database much faster than 10 seconds. By setting the baud rate to 9600, sending the data within a loop, and accounting for a slight delay of messages over HTTP connection, we know that we can store at least 5 data points into the database from the ESP32 every second, which still satisfies the requirement.

For the second requirement, we set up a similar test to the first requirement, but we were instead checking to see how fast the notification appears on the website. When we input a voltage of 140.00 V into the database, we were able to get the notification to pop up in 0.14 seconds. Essentially, we're able to get the notification on the website to appear instantaneously, which is again much faster than the 5 seconds in our requirement.

For our last requirement, we wanted to have the graphs refresh once every minute. Since we were able to use the same websocket technology that we used for the notification, we were also able to get the graph to refresh instantaneously. By setting up a similar test to the notifications, except this time we input 10 sets of data, all the voltage, current, and power graphs updated within 0.23 seconds. While we were hoping to get the graphs to refresh once every minute since we wanted to account for messaging delay, we were able to use industry standard technology to get the data sent and visualized instantly.

3 Cost & Schedule

3.1 Cost Analysis

3.1.1 Parts/Materials

Component	Price (per Unit)	Units
ESP32 Microcontroller	$3.10	1
Step Down XFMR	$11.67	1
Current Sense XFMR	$2.94	1
Battery Charging IC	$0.76	2
Battery Charging IC	$0.23	5
Voltage Regulator 3.3V	$0.00	1
Voltage Regulator 5V	$1.28	1
Power Quality IC	$22.40	1
Submetering IC	$5.19	1
Li-Polymer Batteries 18650	$5.32	2
Fuel Gauge IC	$3.37	1
Voltage Transformer	$10.59	1
BOOST IC	$2.33	1
	Total	$90.18

3.1.2 Estimated Hours of Development

An average UIUC EE and CE graduate has a starting salary of $87,769 and $109,176 respectively. For the sake of simplicity, and since we have two electrical engineers and one computer engineer in our group, we will take the mean annual salary of our group to be:

$$(2/3)(\$87,769) + (1/3)(\$109,176) = \$94,904.67$$

Considering a 40-hour work week for 48 weeks of the year, this is equivalent to a wage of $49.43/hour. We estimate that we have each worked on this project for roughly 10 hours every week, for a total of 10 weeks in the semester. Therefore we can estimate the total cost of labor for this project as follows:

$$\$49.43/hour \times 10\,hours/week \times 10\,weeks \times 3\,members \times 2.5\,overhead\,factor = \$37,072.50$$

3.1.3 External Materials and Resources

Machine Shop:
In addition to the parts listed in Figure 9, we assembled our finished project in an enclosure with the necessary input and output connections. We purchased an existing enclosure (waterproof and suitably electrically rated) and simply had the machine shop put everything together. After perusing various offerings from Amazon, Home Depot, and other retailers, we feel a reasonable estimate for the enclosure is roughly $50 and $30, respectively. After speaking to Machine Shop personnel, we were informed that our project should be doable within a couple business days, so to be safe, we will estimate on the high side and calculate Machine Shop hours needed to be 4 hours a day for a whole week, or 20 hours total. Considering an hourly rate for the shop to be $60 (assuming comparability to the Physics Machine Shop since we are unable to find figures for the ECE shop), this would come out to:

$$\$50 + \$30 + (\$60/hour)(20\ hours) = \$1,280$$

Senior Design Lab Resources:
In addition to the parts and resources we have listed above, we used the parts and equipment available to us at no cost in the senior design lab. This includes (but is not limited to) soldering equipment, oscilloscopes, multimeters, and commonplace resistors and capacitors as needed.

3.1.4 Total Estimated Cost

All things considered, the total estimated cost for this project is shown below.

	Cost
Parts	$59.25
Labor	$37,072.50
Machine Shop	$1,280.00
TOTAL	$38,411.75

3.2 Schedule

WEEK OF	TASK	TEAM MEMBER(S)
2/26/2024	Prepare for design review	Everyone
	Create firmware to connect ESP32 to Wi-Fi and set up Google Cloud database	Roshan
	Start PCB design + parts should begin arriving	Everyone
3/4/2024	Simulate hardware subsystems and finalize PCB design	Soham, Nicole
	Determine layout, dimensions, and all other necessary specifications for Machine Shop order	Soham, Nicole
	Create firmware to connect ESP32 to Google Cloud using HTTP connection	Roshan
	Design 2 subsystems (step down and control unit), PCB 1st wave will be missed	Everyone
3/11/2024	Spring Break	
3/18/2024	Create the basic frontend design without the graphs for the web app	Roshan
	Breadboard power supply and control unit	Soham, Nicole
	Finish Designing PCB for Step Down and Control Unit, PCB 2nd wave will be missed	Soham, Nicole
3/25/2024	Get notifications working on web app	Roshan
	Finish Designing Power Supply, Redesign Control Unit Board	Soham, Nicole
	Submit third wave PCB order	Everyone
4/1/2024	Get graphs working for power outlet input	Roshan
	Submit improved fourth wave PCB order	Everyone
4/8/2024	Get graphs working for load	Roshan
	Soldering and testing available PCBs	Everyone
	Debug Control Unit	Soham, Nicole
4/15/2024	Mock demo	Everyone
	Resolder Control Unit, systematically step through the design, going over each potential failure point and fixing as best as possible	Everyone

4/22/2024	Final demo	Everyone
	Mock presentation	Everyone
4/29/2024	Final presentation	Everyone
	Final report (due Wednesday)	Everyone

4 Conclusion

4.1 Accomplishments

By the end of the project, we successfully got ¾ of the subsystems to work as intended when looking at the requirements and verification table, and we got 2.5/3 of the high-level requirements to work as well. Specifically, the step down, power supply, and database subsystems all work. The only high-level requirement and subsystem that doesn't work is the control unit, specifically the ESP32. We will dive deeper into these issues in the Uncertainties section next. With the step down subsystem, we were able to step the wall voltage down from 120V AC to 3.3V, and we were able to step the current down from 15A to roughly 75mA. With the power supply subsystem, we were able to switch between wall powered and battery powered modes, we were able to charge the battery to 80% to accomplish charging circuit protections, and mathematically we are able to run the system for 10 hours on just battery power. Lastly, for the database subsystem, we have the capability to store data into the database at least once every 10 seconds, we are able to display notifications when there's disturbances within 5 seconds, and we have graphs on the frontend that display voltage and current data. Overall, we believe that the project was relatively successful, and we are very proud that the portions of the project that we have a background in, which is power and software, worked.

4.2 Uncertainties

There's 2 main uncertainties we had with our project: the control unit didn't work and we weren't able to demo the website in person due to the Wi-Fi blocking Google Cloud connections. Regarding the latter, we realized the Wi-Fi issue about a week prior to the final demo when we tried to run the website at the ECE building. The backend server gave a timeout error when it tried to get data from the database, which didn't happen when we worked on a different Wi-Fi network. To look further into this, we contacted professors in the CS department such as Professor Alawini to get an understanding of what could cause the issue, which is when we were told that IllinoisNet Wi-Fi blocks the connection to Google Cloud. To work around this issue, we took a video of our website working to showcase the functionality during the demo.

For the control unit, when we powered our PCB, housing both the Step Down Subsystem and Control Unit, using an external lab power supply, we hit a snag. Trying to load code to the

ESP32 by connecting the D+ and D- connections of a USB from our laptops failed. Our device drivers didn't recognize any connected device. Realizing this issue just one day before the mock demo (a week before the final demo), we tried reseating the microcontroller on its surface mount contact pads. We thought maybe incorrect soldering, possibly due to our first-time use of a stencil and oven, was the problem. But even after doing that, the behavior of our control unit stayed the same. We then turned to the development board, hoping to establish a connection via D+ and D-. While we knew this wouldn't give us full PCB functionality, we hoped it would let us complete the project. Unfortunately, our efforts were in vain. We couldn't load code to the development ESP32 using only the D+ and D- pins. We tried other troubleshooting steps like adding an RC filter and using different cables and development boards, but nothing worked. In the end, we could only load code and send data to the database when connecting via the micro-USB connection on the development board. Looking back, we think having a micro-USB connection point on our PCB explicitly designed to power and program the microcontroller would have helped us understand the issue better.

4.3 Future Work

Now that we've finished our final project, there's plenty of future improvements that we identified that we could've made with more time or other groups in the future can make on our work. When looking at the website, we know that it's not visually appealing enough to release it to the public. For the project, we focused on functionality, so the appearance of the website was sacrificed. We know that this is important for users since people are more likely to use something that catches their eye. Additionally, we would like to implement security features for the website before releasing it to the public since there's a lot of data that is measured in this project that should be encrypted.

For the control board, we would need to completely redesign it. Specifically, we would add probe points for better and easier testing, we could potentially look into using a different microcontroller for easier use, and we would utilize through hole mounts for SMT ICs so that it's easier to solder and debug. The pads on our PCBs were extremely small since the components weren't through hole mounted, which could've definitely resulted in our control unit PCB not working properly.

Lastly, we would look to replace the regulator that we used or add a heat dissipation system. In our design, the regulator heated up very quickly since we didn't use a bigger regulator or add in heat dissipation, which could very easily lead to issues in our PCB if we left it on for too long. Since we caught this issue earlier, we were able to work around it for now, but it's something that should be dealt with in the future.

4.4 Ethical and Safety Considerations

In terms of ethics, we foresaw a few ethical issues that could arise during the development of our project as well as the misuse of our prototype. For example, during the development of our project, we could have poor teamwork, alienate group members, or stop participating in the project. Additionally, we could copy the designs of other submeters or power quality monitors on the market. In terms of the misuse of our prototype, the data that is collected is private data for the users and must be protected. There are people that could try to access other people's data, which is a violation of laws and regulations.

Our group committed to maintaining the highest ethical standards throughout the course of this project by following the **IEEE Code of Ethics**, such as upholding the highest standards of integrity, treating everyone equally and with respect, and making sure that the Code of Ethics is followed by the entire group. We avoided these ethical breaches and followed the aforementioned Code of Ethics through the following:

1. We set up a group chat through iMessage where we consistently communicated our schedules, when we could work, and what we planned to work on. This allowed us to split the workload well and finish our work on time while ensuring high quality. We also set up a Google Drive folder for all our ECE 445 documents so that we could stay organized and stay on the same page.
2. We held each other accountable to make sure all of the work we do is of high quality and is original. If questions or uncertainties arose, we made sure to contact and consult with experts in the field such as Surya (our TA), Jason (another TA), Professors, Machine Shop Technicians, or our mentor Mr. Jack Blevins.
3. Since this project isn't planned to be released to the public, we didn't address data privacy since having secure encryption could take months to create. However, if we were to sell the product, we would have all data be encrypted and password protected so that there is data privacy.

In regards to safety, we identified lithium-ion batteries and working with wall voltage to be the 2 main safety concerns. We worked around this in many ways. First, we used LiFePo batteries since the chemistry in the battery is less likely to cause a fire or other safety hazards. Second, we marked the device as a potential safety hazard due to high voltage. Lastly, we used adequate lab safety equipment while working in the lab such as safety goggles to protect ourselves.

References

"1910.137 Electrical Protective Equipment Standards." *Occupational Safety and Health Administration*, 11 Apr. 2014, https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.137.

"18650 Batteries - High Quality Rechargeable Lithium-Ion Batteries." *18650BatteryStore.Com*, https://www.18650batterystore.com/collections/18650-batteries?pf_rs2_discharge_current_amps=4.90%3A4.90. Accessed 8 Feb. 2024.

"1MHz Step-Up PWM Converter with 2A Switch Current." *Perfect Microelectronics Co., Ltd. FP6291*, https://file2.dzsc.com/product/17/08/05/1125865_100842357.pdf. Accessed 8 Feb. 2024.

Communications, Grainger Engineering Office of Marketing and. "Machine Shop." Physics.illinois.edu, physics.illinois.edu/research/facilities/machine-shop#:~:text=Prices%20Professional%20products%20%2460.00%2Fhour%20plus%20cost%20of%20materials. Accessed 22 Feb. 2024.

Grainger Engineering Office of Marketing and Communications. "Salary Averages." Ece.illinois.edu, ece.illinois.edu/admissions/why-ece/salary-averages. Accessed 22 Feb. 2024.

"High Efficiency 1.2MHz 2A Step Up Converter." *AEROSEMI*, https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf. Accessed 8 Feb. 2024.

"How to Build a 18650 Lithium Battery Charger and Booster Module." *Circuit Digest - Electronics Engineering News, Latest Products, Articles and Projects*, https://circuitdigest.com/comment/31902#comment-31902. Accessed 8 Feb. 2024.

JKElectronics. "Esp32 Max Current Output - General Electronics - Arduino Forum." *Arduino Forum*, Arduino Forum, 6 Nov. 2022, https://forum.arduino.cc/t/esp32-max-current-output/1050205.

Staff, Editorial. "Insight Into ESP32 Sleep Modes & Their Power Consumption." *Last Minute Engineers*, Last Minute Engineers, 23 Dec. 2018, https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/#:~:text=ESP32%20Active%20Mode,-Normal%20mode%20is&text=Since%20everything%20is%20always%20active,and%20Bluetooth%20are%20used%20simultaneously.

Appendices

Appendix A - Figures

Figure 1.1: Visual Aid

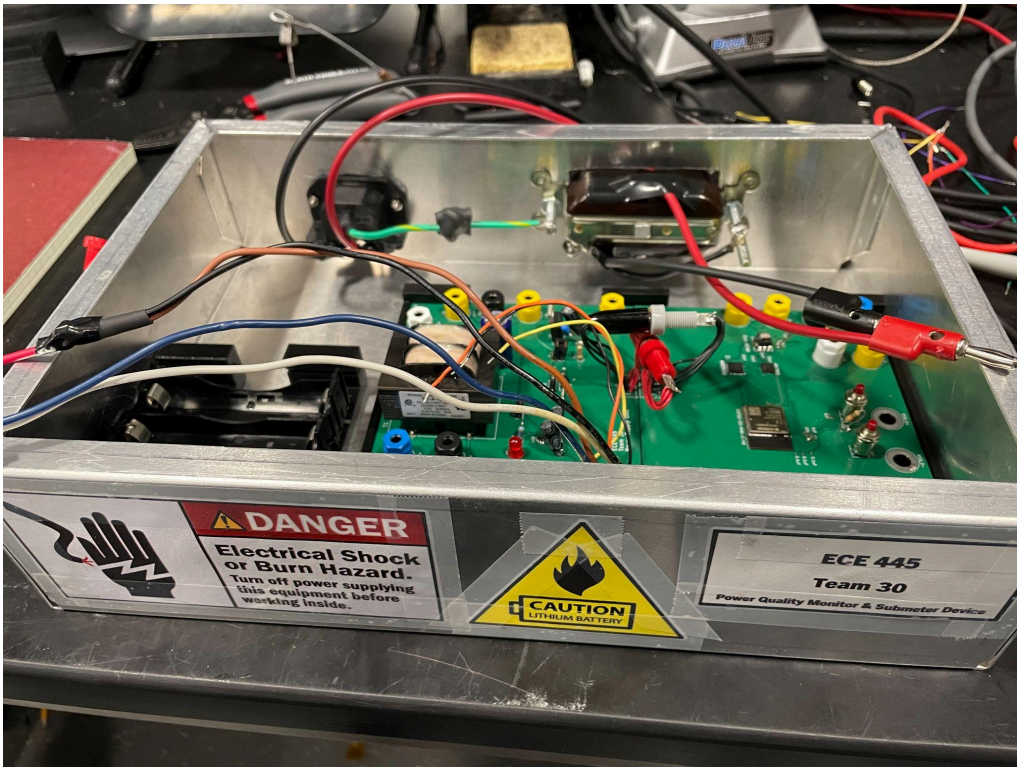


Figure 1.2: Project Enclosure

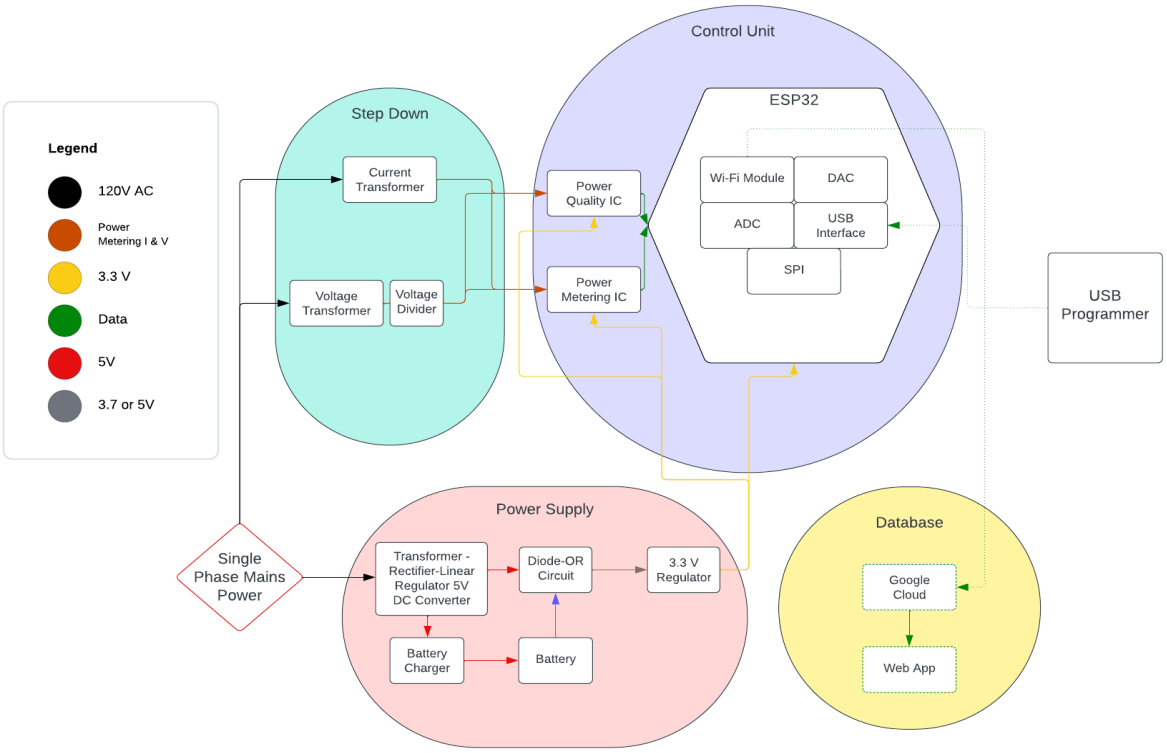


Figure 2: High Level Block Diagram

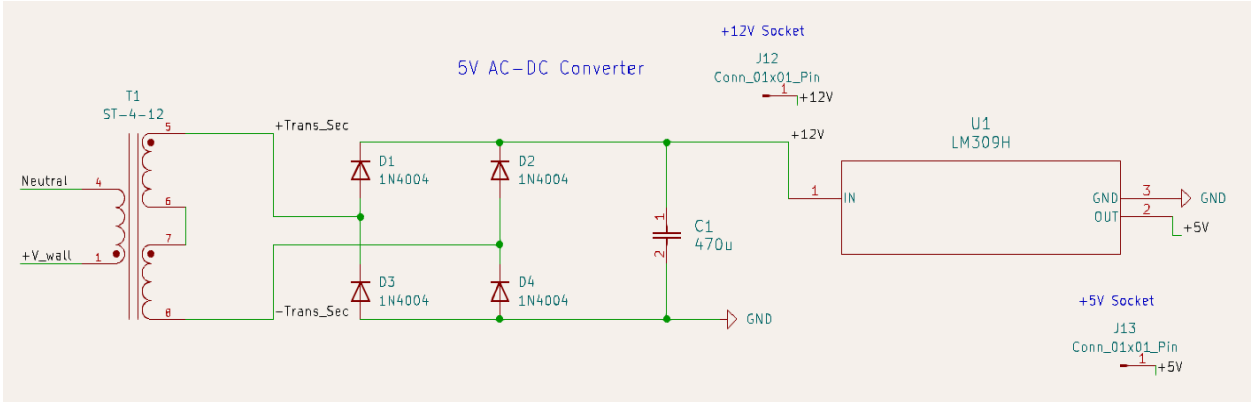


Figure 3.1: Power Supply 5V Converter Circuit Diagram

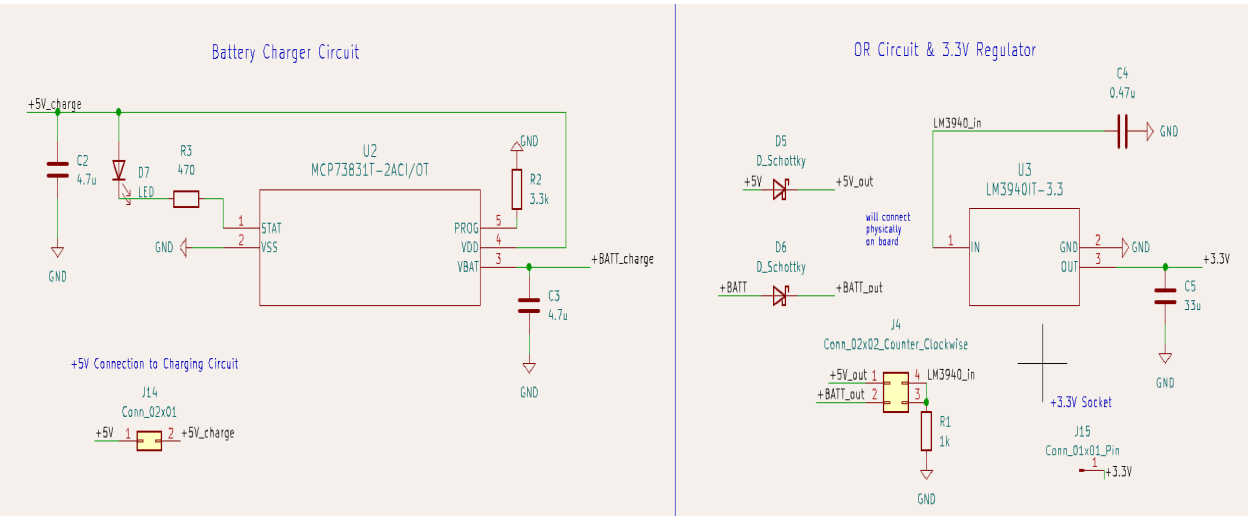


Figure 3.2: Power Supply Battery Charger and 3.3V Regulator Circuit Diagram

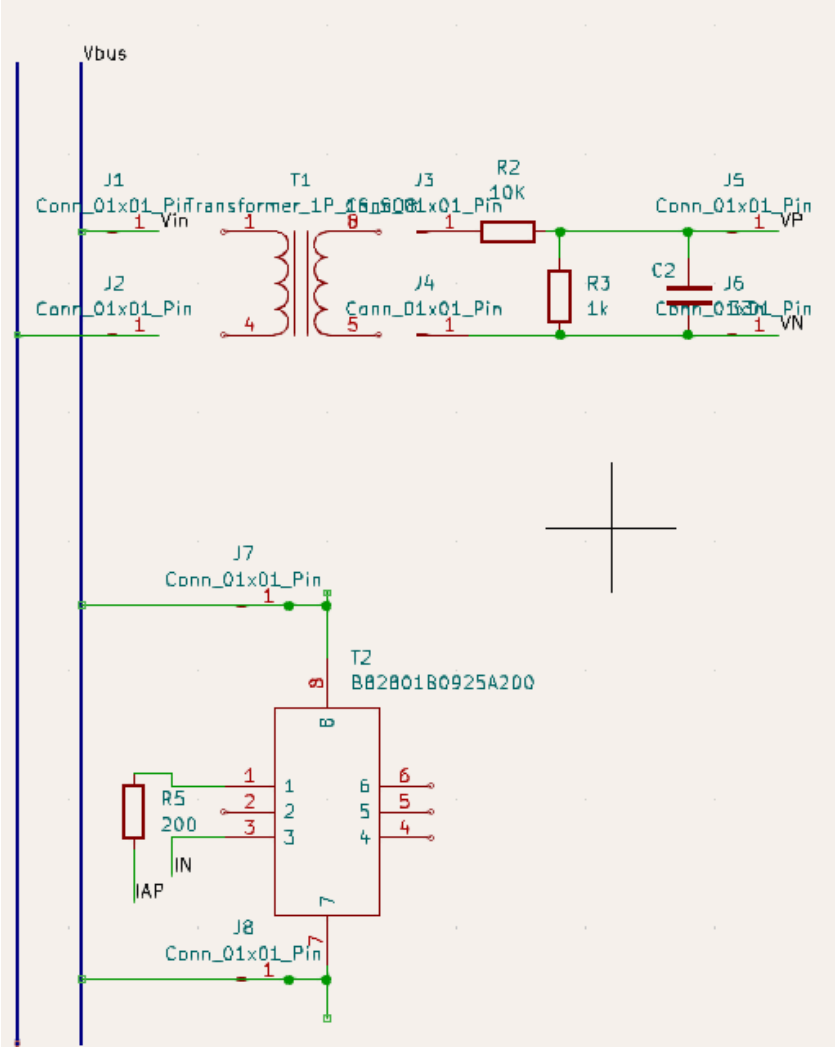


Figure 4: Step Down Subsystem Circuit Diagram

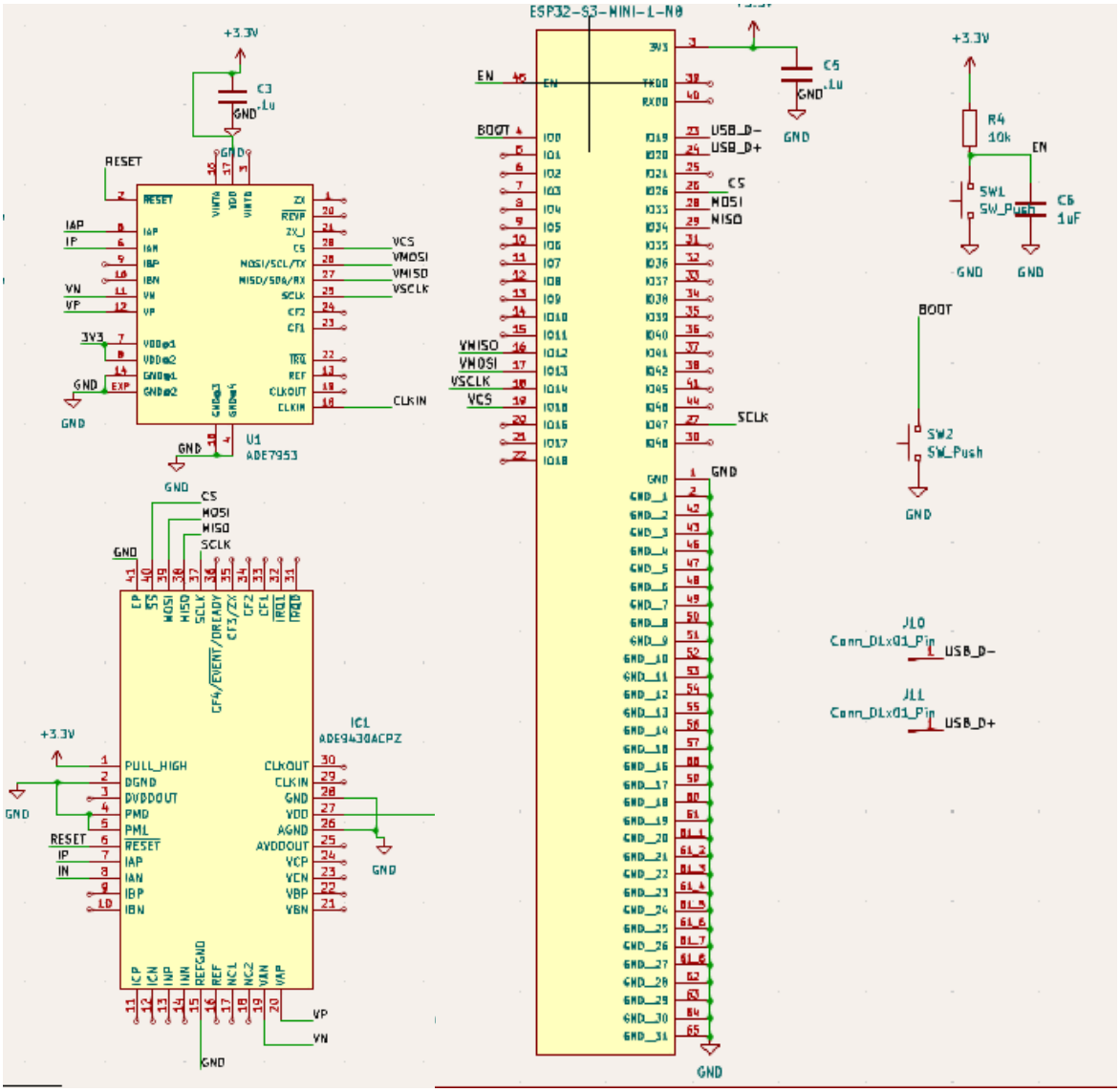


Figure 5.1: Control Unit Subsystem Circuit Diagram

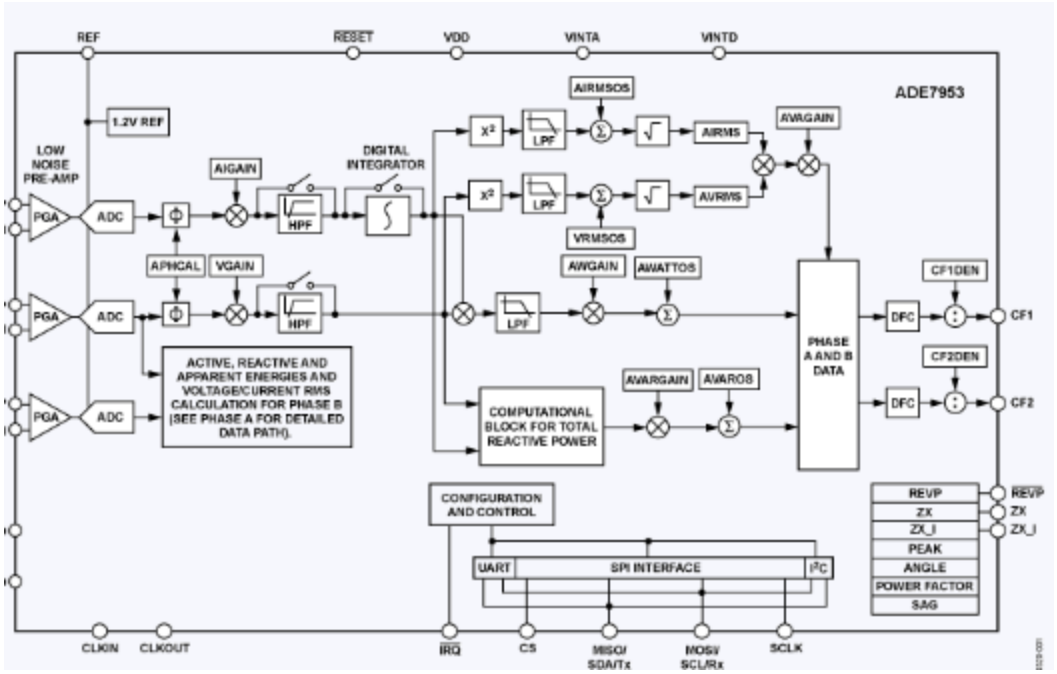


Figure 5.2: ADE 7953 Block Diagram

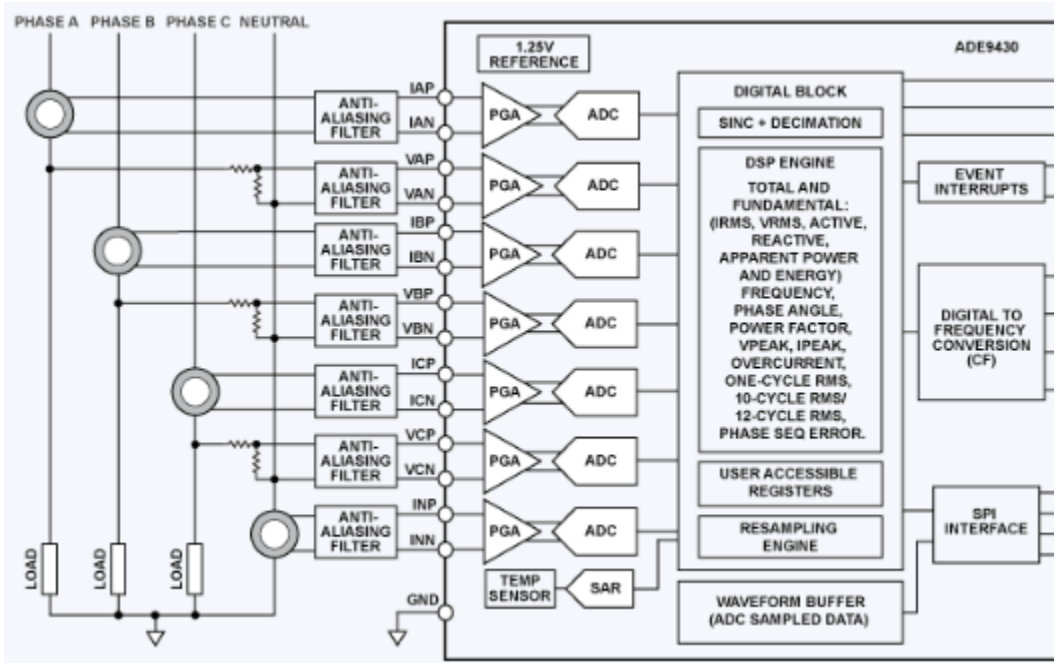


Figure 5.3: ADE 9430 Block Diagram

```
mysql> SELECT * FROM PowerData;
+---------+-------------+-----------+---------------------+---------+---------+----------------+------------+
| data_id | outlet_name | load_name | measurement_time    | voltage | current | apparent_power | real_power |
+---------+-------------+-----------+---------------------+---------+---------+----------------+------------+
|       0 | ECEB        | CHARGER   | 2024-04-24 07:50:22 |  122.31 |    3.11 |         380.38 |     380.38 |
|       1 | ECEB        | CHARGER   | 2024-04-24 07:50:27 |  121.44 |    3.06 |         371.61 |     371.61 |
|       2 | ECEB        | CHARGER   | 2024-04-24 07:50:37 |  119.23 |    3.09 |         368.42 |     368.42 |
|       3 | ECEB        | CHARGER   | 2024-04-24 07:50:43 |  117.68 |    3.02 |         355.39 |     355.39 |
|       4 | ECEB        | CHARGER   | 2024-04-24 07:50:56 |  123.11 |    3.11 |         382.87 |     382.87 |
|       5 | ECEB        | CHARGER   | 2024-04-24 07:51:08 |  121.22 |    3.27 |         396.39 |     396.39 |
|       6 | ECEB        | CHARGER   | 2024-04-24 07:51:17 |  118.33 |    3.06 |         362.09 |     362.09 |
|       7 | ECEB        | CHARGER   | 2024-04-24 07:51:29 |  117.98 |    3.12 |         368.10 |     368.10 |
|       8 | ECEB        | CHARGER   | 2024-04-24 07:51:36 |  120.43 |    2.95 |         355.27 |     355.27 |
|       9 | ECEB        | CHARGER   | 2024-04-24 07:51:44 |  122.44 |    3.17 |         388.13 |     388.13 |
|      10 | ECEB        | CHARGER   | 2024-04-24 07:51:52 |  126.31 |    3.11 |         392.82 |     392.83 |
|      11 | ECEB        | CHARGER   | 2024-04-24 07:52:01 |  135.44 |    3.06 |         414.45 |     414.45 |
|      12 | ECEB        | CHARGER   | 2024-04-24 07:52:09 |  113.23 |    3.09 |         349.88 |     349.88 |
|      13 | ECEB        | CHARGER   | 2024-04-24 07:52:16 |  109.68 |    3.02 |         331.23 |     331.23 |
|      14 | ECEB        | CHARGER   | 2024-04-24 07:52:24 |  109.68 |    3.02 |         331.23 |     331.23 |
|      15 | ECEB        | CHARGER   | 2024-04-26 06:49:00 |  140.00 |    3.00 |         420.00 |     420.00 |
+---------+-------------+-----------+---------------------+---------+---------+----------------+------------+
16 rows in set (0.00 sec)
```

Figure 6.1: Database Stored Data Format

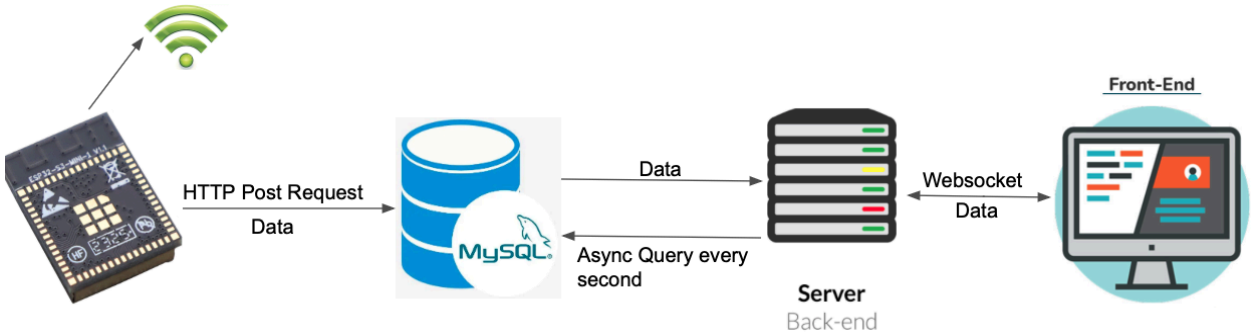


Figure 6.2: Database Flowchart

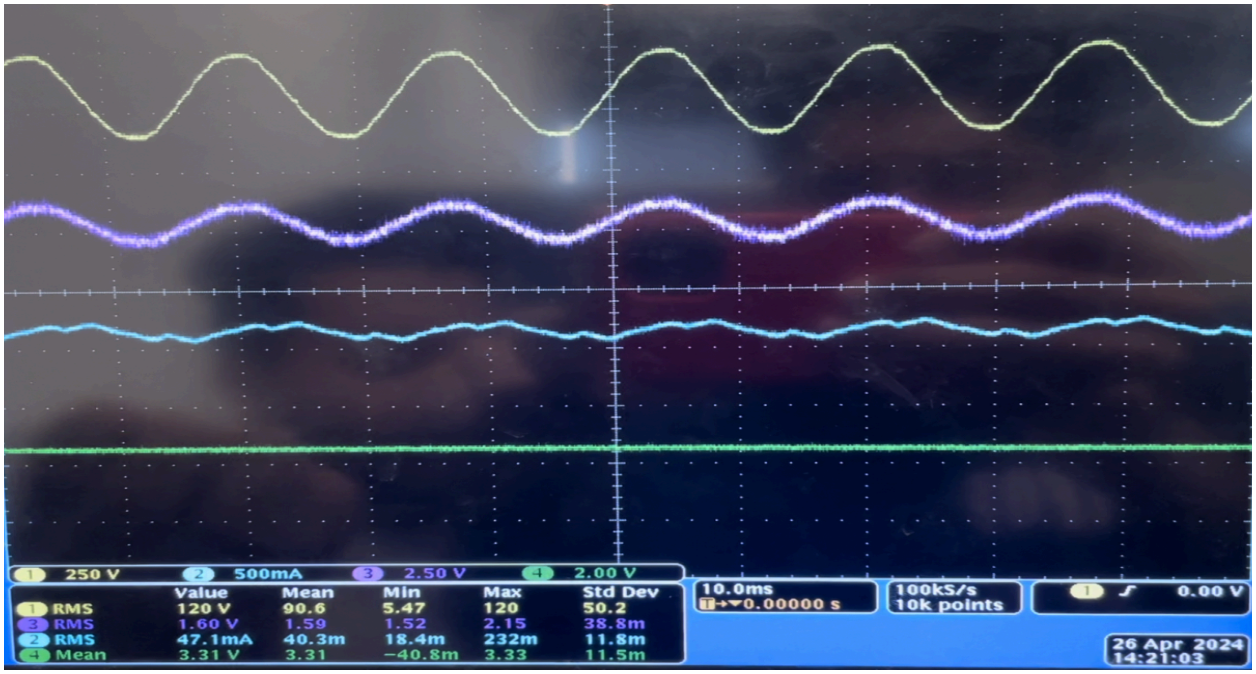
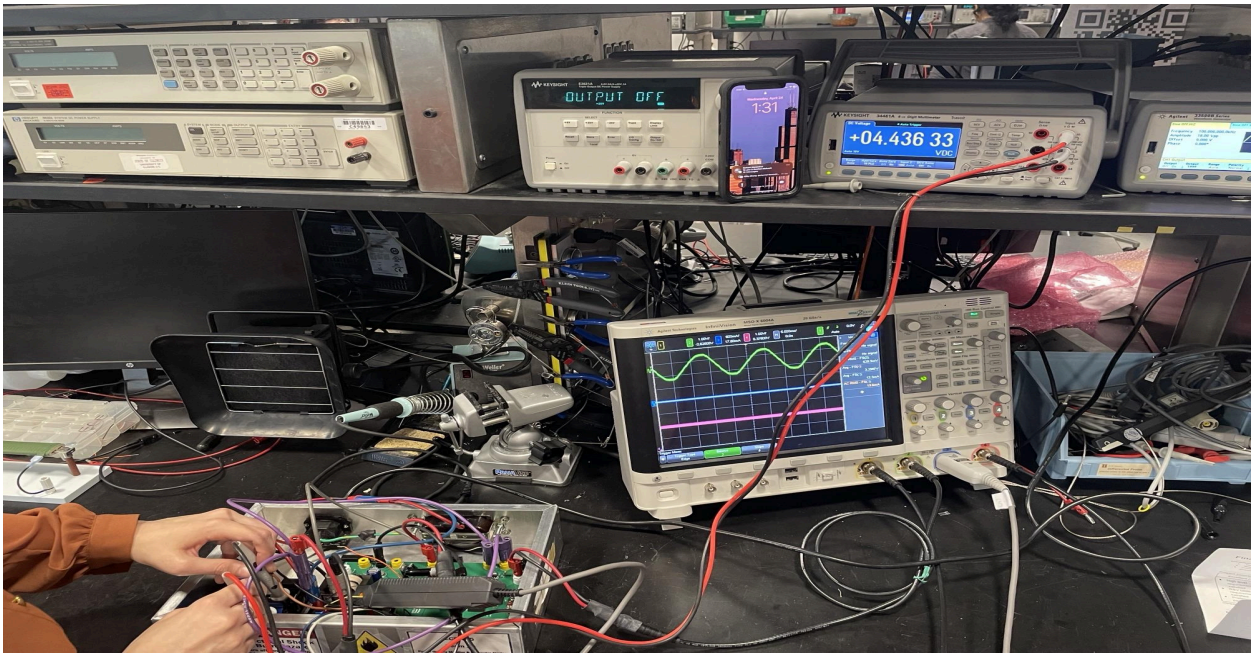


Figure 7.1: 4 Major Waveforms: Wall voltage (yellow), Step Down Unit voltage (purple), Wall current under no load (blue), Power Supply DC output voltage (green)



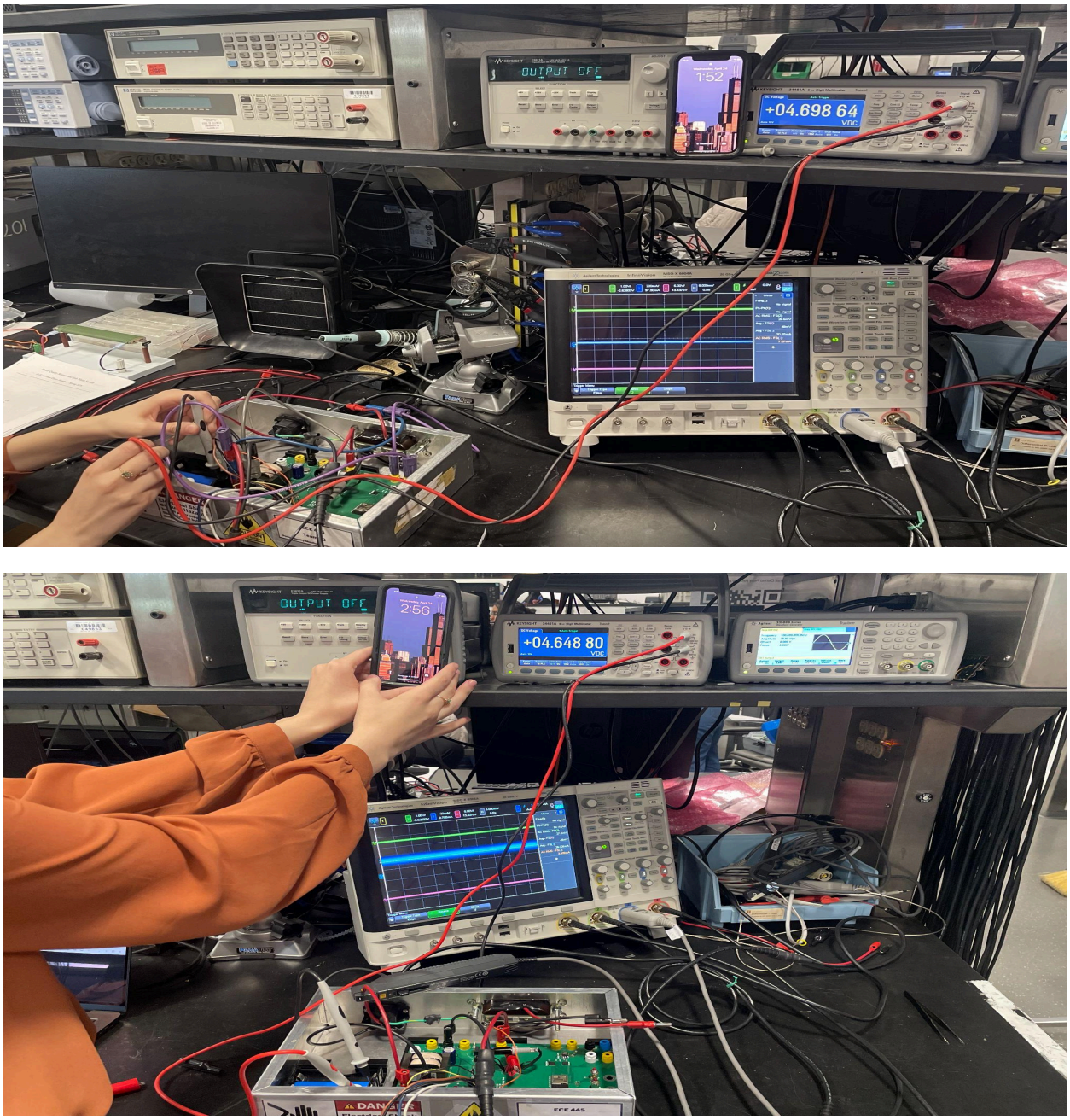


Figure 7.2: Battery charging limited to 80% state-of-charge

Appendix B - Tables

Requirements	Verification
Convert 120 V AC to 3.3 V DC and deliver to Control Unit	● Use a multimeter to probe test points on Power Supply, supplying intermediate voltages (testing 120 V AC last) to ensure safety.
Switch between wall powered and self-powered modes in the event of a blackout without loss of operation	● Simulate a blackout by removing wall power and validate that 3.3 V is still being delivered to some load at the output of the supply
In self-powered mode, system should run for >10 hours	● Fully charge battery and run system powering an analog bit adder counter circuit. Verify that the battery operates for more than 10 hours.
Charging circuit protections: charging battery to 80%, at rated current maximum	● Probe Battery voltage and current every 15 minutes when charging from low voltage (<4 V). Verify voltage does not exceed 80% of rated value. ● Optionally integrate fuel gauge IC onto PCB to display battery percentage.

Table 1.1: Power Supply RV Table

Requirements	Verification
Step Down Voltage from 120 V RMS to <3.4 V since the ICs cannot handle a voltage higher than that	● Send an AC waveform smaller than the wall voltage through the divider with some load across the resistor and observe that it steps down to 3.4 V ± 2%
Step Down Current from <15 A to ~<75 mA	● Send an AC waveform smaller than the wall voltage through the transformer with max load of 15 A across the secondary and observe that the current steps down to 75 mA ±5%

Table 1.2: Step Down Subsystem RV Table

Requirements	Verification
Original Voltages and Currents (pre Step Down) should be sent to the ESP32 from the measurement ICs and be accurate to ± 5%	● Send a small AC waveform directly to the ICs and record the data directly to On-Chip Memory as a calibration test. ● Give 3.3 V to VDD and Vin for the ICs and ESP32 respectively and use the wall as a source testing the Step down and Control unit subsystem at the same time, under no load and low load condition

Table 1.3: Control Unit Subsystem RV Table

Requirements	Verification
The database must store new data once every 10 seconds from the ESP32.	● Set up ESP32 Wi-Fi connection and establish HTTP connection with Google Cloud database. Set the ESP32 at least 50 meters away from the laptop running the database. ● Write firmware code in ESP32 to send fake time (Unix epoch time), voltage, and current data once every 10 seconds for 5 minutes. This comes out to a total of 30 sets of data being sent. We would send fake data at first just for testing purposes to verify that we can satisfy this requirement. ● Use “SELECT *” statement in Google Cloud terminal to see all the data. For success, we would need at least 29/30 data sets to be correctly present in the database.
The system must show a notification on the frontend when there’s any disturbances, such as voltage fluctuations outside of a set tolerance by 5%, or power failures. This notification must appear within 5 seconds.	● Write firmware code in ESP32 to send fake time (Unix epoch time), voltage, and current data with the voltage set at 5% above a set tolerance, then the voltage set at 5% below a set tolerance, then repeated for the current. We are using fake data to simulate data for verification purposes. ● We will then repeat the prior step but

	set the voltage to 0 V to simulate a power failure. ● After sending the data from the ESP32, we will then start a stopwatch to count the time it takes for the notification to show up. ● For each of the 5 instances, we must see a notification pop up on the frontend of the web application within 5 seconds of sending the data from the ESP32.
The frontend must have a graph that shows voltage and current data that is refreshed at least once per minute.	● Write firmware code in ESP32 to send fake time (Unix epoch time), voltage, and current data that is the exact same for 55 seconds. We are using fake data to simulate data for verification purposes. ● Then, we will change the data that is sent from the ESP32 for the next 55 seconds. ● For every 1 minute, we must see the graph changing to reflect the correct data.

Table 1.4: Database Subsystem RV Table