ECEB Submetering

By

Mike Lee

Aleksai Herrera

Jonathan Izurieta

Senior Design Laboratory(ECE 445) Final Report

TA: Sanjana Pingali

01 May 2024

Project #75

Abstract

As the power grid grows, the need to measure the quality of power delivered also increases in order to ensure that clean and proper energy is being delivered to loads. This report serves to document the process that we undertook this semester in order to design, test, and implement a power metering system designed to measure the single phase power and voltage of a load representing the power consumed by the ECEB. This power meter would then display these values in order to see how power was being delivered. This report will discuss the design process and considerations that went into designing the overall system as well as its subsequent subsystems, and it will conclude with the results that we achieved as well as improvements to be made in the near future.

Contents

1 Introduction

1.1 Problem

Businesses, hospitals, data centers, schools, and so many other infrastructures are continually growing and expanding year after year. All of these facilities, and homes alike, require reliable power and energy to run. These facilities all get their power from the grid, where power is delivered in a 3 phase format. These facilities are seen as a resistive, capacitive, or inductive load by the power grid. Capacitors and inductors in theory are lossless, but in reality they are not. This means that some power may be lost within inductive and capacitive loads, which would mean that the power being delivered by the grid is not efficient. The power being delivered is not physically evident, but it is only apparent that the correct amount of power is being delivered if the facility receiving it turns on and runs. This could mean that a facility needing 100kW of power is being sent 500kW by the grid, which would be an efficiency of 20%, a horrible loss of both power and resources. There is a need to monitor this power in order to avoid huge power losses which could cost the business more than it needs. These power losses also lead to a loss of the resources used in generating the power.

1.2 Solution

Our goal is to create a 3 phase power meter to be installed at facilities in order to calculate the true power that is being delivered to the facility. To help in achieving this goal, a 1 phase power meter will be created because if 1 phase can be measured accurately, the same technology can be expanded to measure 3 phases. This power meter will calculate phase angle, power factor, and rms voltage and current values. It will then display these values for viewing, allowing grid operators to manage power delivery from the grid, which in turn will allow them to find areas of major power loss and be able to fix them for more efficient power delivery.

1.3 Requirements

1. Our power meter must be able to measure the correct RMS values of current and voltage of a 3 phase 208V 60Hz input signal to within ±5% of the correct value. (This was later modified to be able to measure just 1 phase to within the same margin of error. This input signal would come from a signal generator, 10 Vpp, which is what the transformer that was purchased would output if connected to mains power.)
2. Our power meter must be able to accurately calculate the phase angle and power factor of the input signal to within ±10%. (Since RMS voltage and current values are used to calculate these the margin of error may be greater.)
3. The data we collect should be displayed to an LCD screen for power monitoring, as well as uploaded to an external database which can be used to display these same power statistics to TVs in the ECEB.

2 Design

2.1 Overarching Design

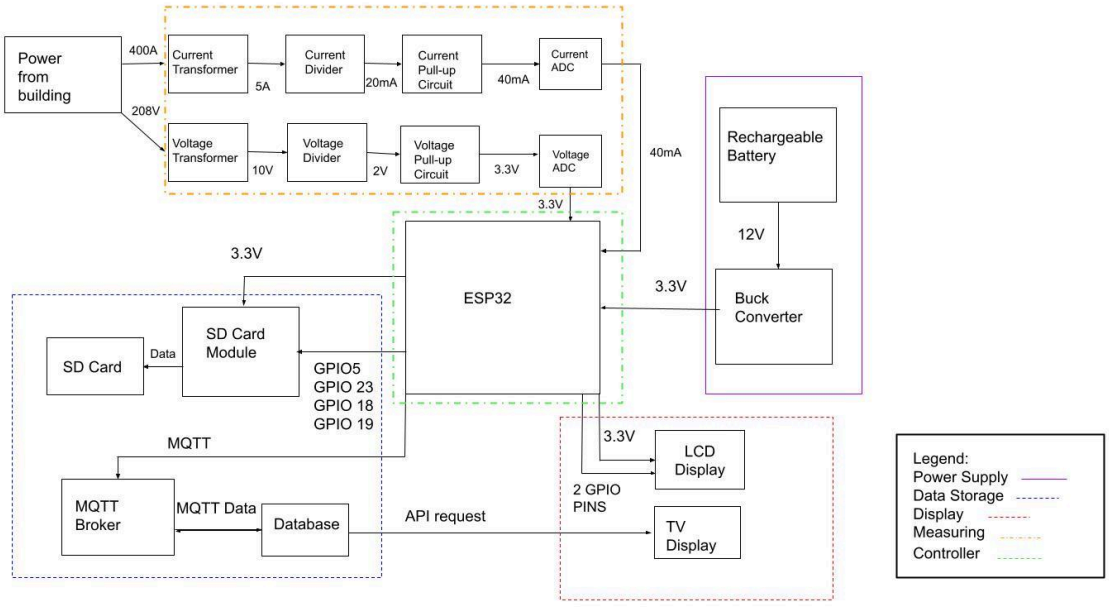

Figure 1: Block Diagram of Power Meter

Figure 1 presents our subsystems at the highest level. Our design contains a measuring subsystem which is responsible for getting the current and voltage signals to our controller subsystem for processing. This controller subsystem then communicates with our display and data storage subsystems in order to both save and display the data it received. Lastly, we have a power supply system which is responsible for making sure that everything on the PCB is powered. The rest of this design section aims to dive deeper into each of the subsystems seen in Figure 1 and how these subsystems play a pivotal role in the overall function of our power meter.

2.2 Measuring Subsystem

The measuring subsystem consists of the current and voltage sensing circuits. These two circuits are identical except that the current sensing circuit contains a shunt resistor. Our measuring circuits connect to the leads of the transformer in order to get the waveforms to measure. We utilize a 1.65V pseudo ground (created using a simple voltage divider from our 3.3V supply) connected to the GND terminal of the transformer in order to shift up the incoming voltage signal. This is necessary because the microcontroller that is being used for calculations (ESP32) does not handle negative voltages, and they can potentially damage the microcontroller. After this waveform is shifted up, it is brought down to around 2.2Vpp via voltage divider since

our op-amps are being powered with 3.3V and the signal would otherwise be cut off at the top end.

The op amps we chose to use for our circuits were the OPA2340. These op amps can function in single supply mode and are rail-to-rail amplifiers which is very important since we want the signal as close to 0-3.3V as possible. This signal is then sent through a non-inverting operational amplifier that amplifies the signal via Equation(1).

$$V_{out} = V_{in}\left(1+\frac{R_1}{R_2}\right) - 1.65\left(\frac{R_1}{R_2}\right) \tag{1}$$

The signal is then filtered through a low pass RC filter with a cutoff frequency of 59Hz in order to recover the original waveform as well as cut out any possible higher order harmonics and noise which may be present outside the 60Hz. Finally, there is an additional op amp which acts as a buffer before sending the signal to the ADC on the ESP32. This buffer was included due to the initial current draw of the ESP32 ADC pin which has an internal capacitor.

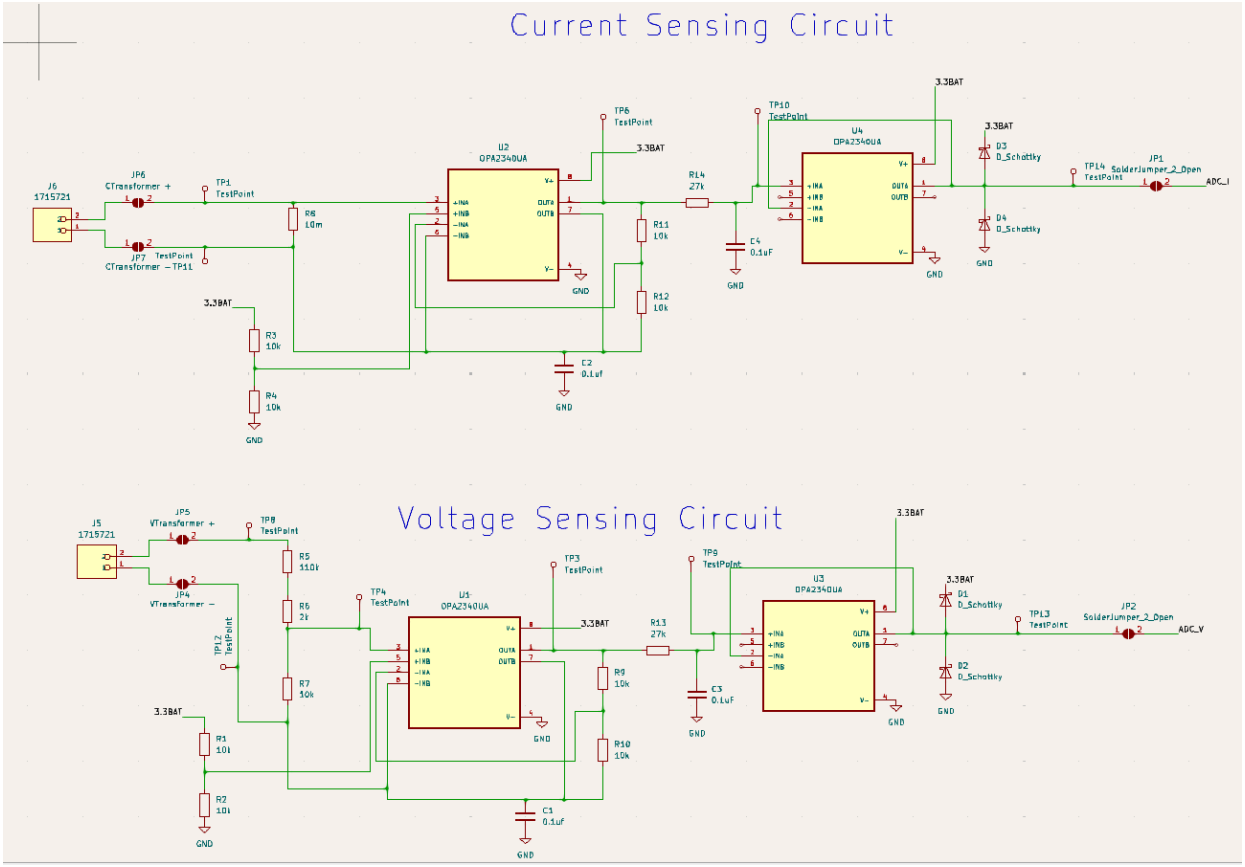

Figure 2: Schematic of Current and Voltage Sensing Circuits

2.3 Power Subsystem

Our board needed a 3.3V supply in order to power the op amps, the microcontroller, for creating the pseudo ground, and for the LCD screen. We wanted a long local battery life for our meter, at least 10 hours, so we had to calculate the current draw in the worst case scenario. Since the ESP32's wifi module was going to be utilized in order to offload data to the cloud, the current draw for this process was used since the current draw while using the wifi module is the greatest, around 240mA [3]. The current draw of our op amps was about 1ma [4], and our lcd screen was around 60mA [5]. This brings our total current draw to roughly 300mA, and since we wanted our battery to last for at least 10 hours, this would require a battery with a rating of about 3000mAh. Most batteries with this mAh rating are car batteries or have very high voltage which is impractical for a power meter. We went with a 12V 2400mAh battery which would give us 8 hours of battery life.

Since a 12V battery was chosen, it had to be stepped down to 3.3V. We had to decide between utilizing a linear regulator or a buck converter in order to step down 3.3V. Ultimately, a buck converter was used because the linear regulator would have dissipated too much heat because of our high power battery. The circuit chosen in order to implement our buck converter came from the manufacturer datasheet [2], and it stepped down our input voltage to 3.3V.

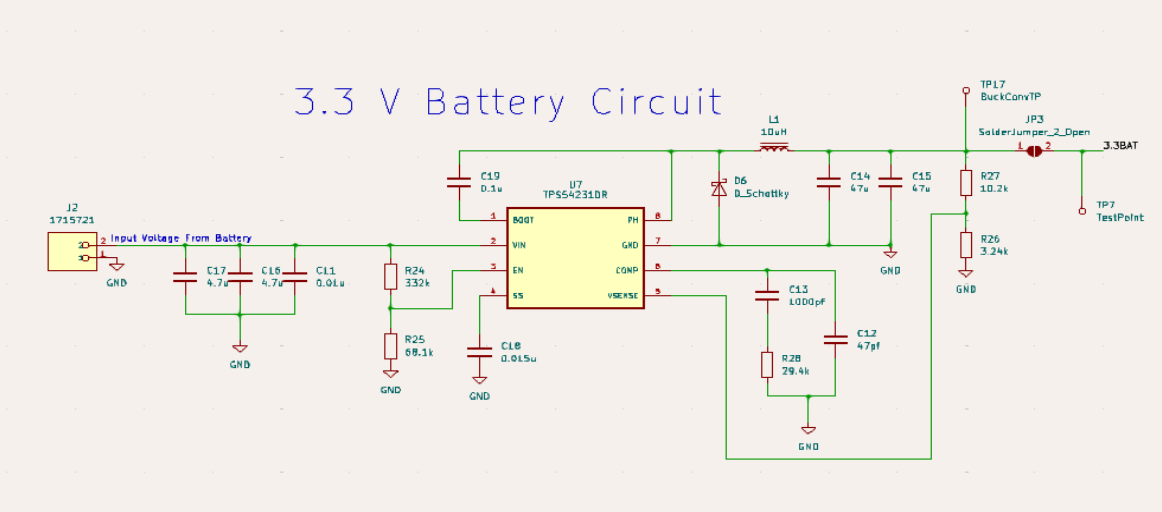

Figure 3: Schematic for Step Down Buck Converter to 3.3V

2.4 Controller Subsystem

The controller subsystem consists of the microcontroller, the ESP32, and how it interfaces with the other subsystems, as well as the code implemented and programmed onto the device. More specifically, how it interfaces between the measuring subsystem, display subsystem, and data storage subsystem, and is powered by the power supply system. Figure 4 is

a schematic showing which pins interface with which subsystems, with the unlabeled CHIP_PU, TX, RX, and GPIO0_STRAP pins being used to program the ESP32 while it is on the PCB.

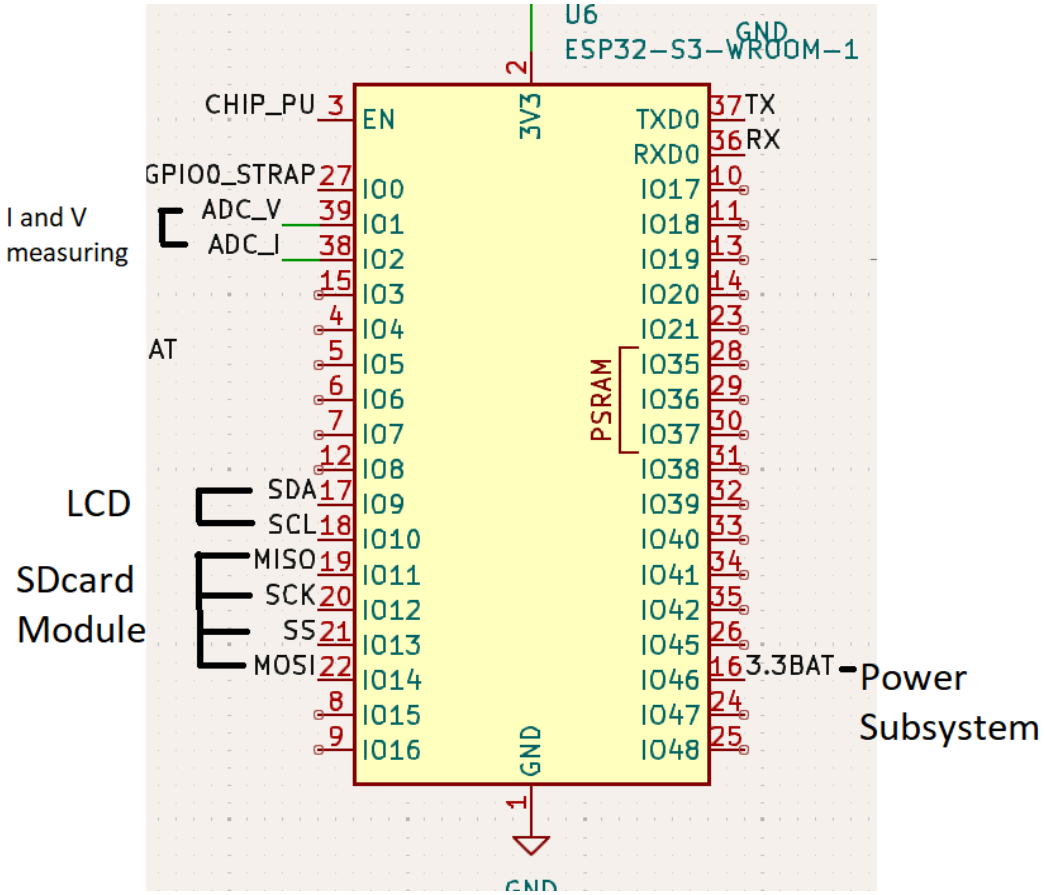

Figure 4: Schematic pin out of ESP32

2.4.1 Programming the ESP32

Initially, for early debugging purposes, the ESP32S3 development board was used to upload the programs via a USB to microUSB cable. For the ESP32 module used on the PCB, a programmer circuit was designed based on the ESP32 example on the wiki website [9], shown below in Figure 5, along with a USB to UART bridge module. The ESP32 is programmed by the circuit bringing the GPIO0 pin to a low logic level, allowing the ESP32 to go into Boot mode and code to be written onto the flash memory through the RX and TX pins.

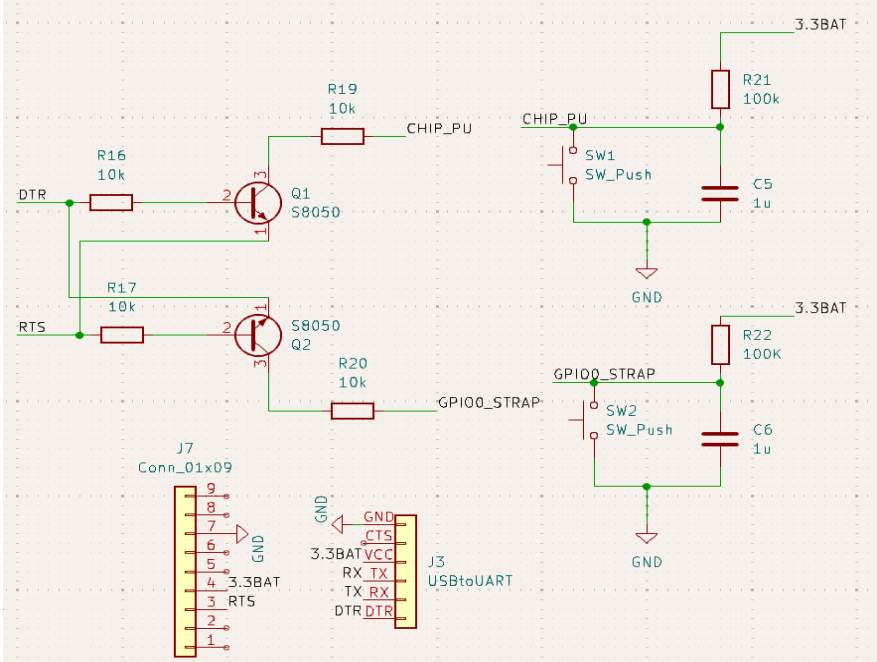

Figure 5: Programming Circuit Schematic

2.4.2 Analog Measurements and Calculations

The ESP32 uses type ADC_1 pins, ADC_I and ADC_V, to receive the analog measurements for current and voltage. These pins contain an internal analog to digital transformer that allows a discrete value to be read and recorded in the code using the analogRead() function. In addition, EmonLib is used and initialized with the aforementioned ADC pins and is used to calculate Irms, Vrms, power factor, and phase angle. It calculates the input signals real power by taking the average of instantaneous power over several cycles. Apparent power is then calculated by using Vrms * Irms, subsequently calculating power factor is then achieved by dividing real power by apparent power. Finally, the phase angle phi is calculated by taking the inverse cosine of real power over apparent power. The calculations are summarized in Figure 6, with phi representing the phase angle.

$$P_{apparent} = V_{rms} \cdot I_{rms}$$

$$P_{factor} = \frac{P_{real}}{P_{apparent}}$$

$$\phi = \arccos\left(\frac{P_{real}}{P_{apparent}}\right)$$

Figure 6: Equations used in Calculations

2.5 Data Storage Subsystem

The data storage subsystem consists of storing data locally, by recording a .csv file onto the microSD card that was connected via the microSD card module. We initially decided to store the data in a text file, but it would make organizing data and reading the data harder to do when looking at raw data. We designed the file system so that we could easily purge data over time. Each day a new file is created where the ESP32 would then start writing data to. Then the MQTT client we coded would read the data off of the SD card and publish the data written in the last 15 minutes. AWS IoT Core ingests the MQTT messages and writes to a Time Series Database(TSDB). After a month, data will be moved to long term storage.

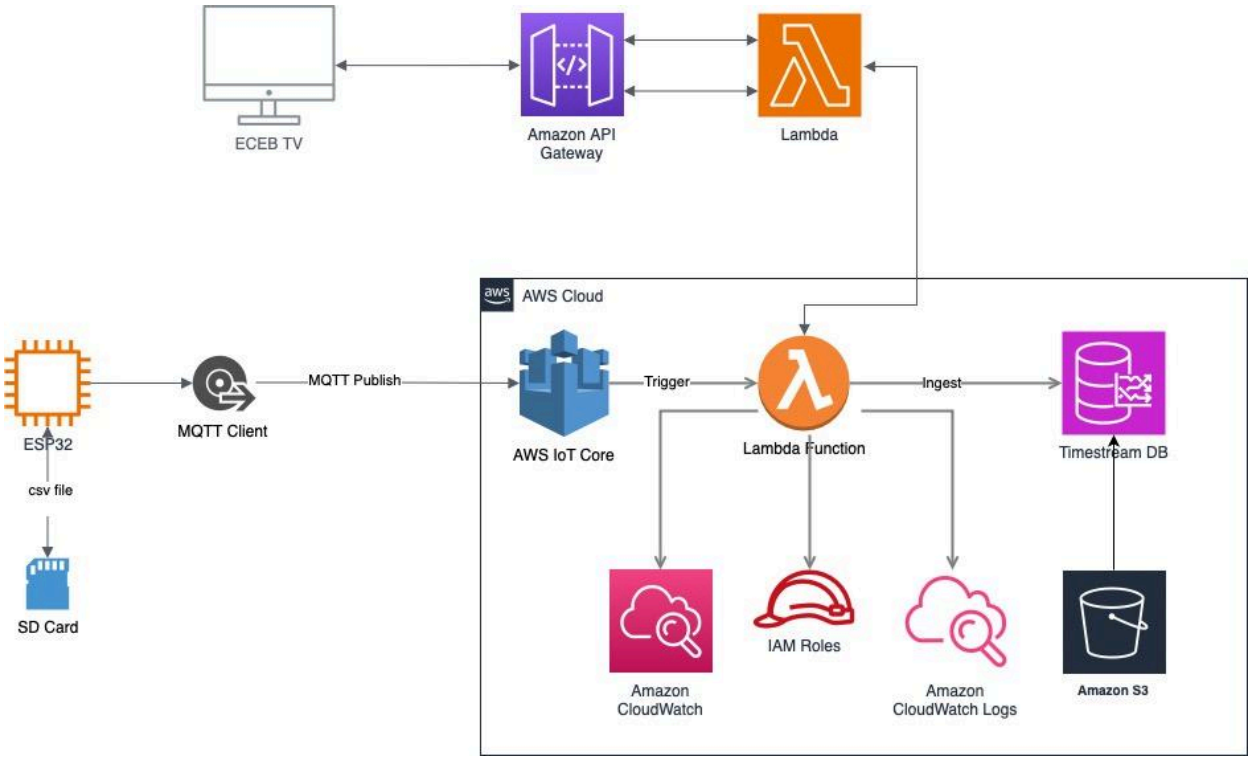

Figure 7: Cloud Infrastructure Diagram

2.6 Display Subsystem

The display subsystem consists of two major components, an LCD display and a routine to display onto the ECEB lobby television. The LCD is connected to the ESP32 and displays the power factor and phase angle data. The ECEB lobby television would display .csv file tables that have been uploaded to the website from the database on the online server. Under ideal conditions, recorded data would be displayed on graphs that also illustrate trends and perform additional data analysis. The ECEB TV's would access the data by displaying a website hosted on AWS that displays the data as seen in the block diagram in Figure 7.

3. Verification

This section dives into the requirements that were set for each subsystem in order to determine that they were functioning adequately, and the tests that were performed in order to verify these requirements. These will be discussed at a high level except the measuring subsystem requirements, data is expressed in their verification since this is the most crucial component of the project. Appendix B details the full in depth requirements and verifications for each subsystem.

3.1 Measuring Subsystem Requirements & Verification

3.1.1 Acceptable ADC inputs

One of the requirements we established was that the sensing circuit should bring the voltage and current into an acceptable range for the internal ESP32 ADC input such that it does not damage the device. The set standard was the current sensing circuit to bring the current into a range of less than 40mA, and the voltage sensing circuit to bring the voltage to 3.3V ± 0.3V. Verification would be performed by connecting the load, or in our case the signal generator simulating the load's current and voltage signals, to the sensing circuit and then using a multimeter and oscilloscope to measure the output, and observing if it is in an acceptable range.

This verification for current was easily met since the current transformer that we chose had an output current of 5mA which is way below our target of around 40mA. To bring our voltage into the range for the ESP32 a simple voltage divider was created in order to bring the voltage to around 2.2Vpp before passing it through the non inverting operational amplifier. Some safety was also in place just in case there were some voltage spikes that occurred. Since our op amps are being powered by a 3.3V power supply, any voltage higher than that would be cut off by the op amps in the circuit, and there was also a final safety net of a diode clamp circuit before the ADC pins in order to ensure nothing above 3.3V or below 0V would get to the ADC pin. This was tested with an oscilloscope and the voltage always stayed in range when varying the input.

3.1.2 Minimal Modification/Change of Signals

Another requirement was ensuring our sensing circuit should not significantly modify or change the actual input values of the voltage and current. This behavior is validated by recovering the original waveform and calculating the RMS values of current and voltage. Verification was achieved by shifting the incoming waveform down by 1.65V and then multiplying by 6.06 to bring the 0-3.3V waveform back to its original 10Vpp. We found the V_{RMS} after shifting and scaling to be 7.0V compared to the expected 7.07V giving us an error of 1% which demonstrates that the original signals were modified minimally.

3.2 Power Supply Subsystem Requirements & Verification

3.2.1 Acceptable ESP32 Power

The power supply subsystem must bring down the battery voltage, 12V, down to an acceptable level of 3.3V so that it does not damage the ESP32 and is at the specified acceptable logic level high. Verification is performed by connecting the buck converter circuit to the battery, however, the ESP32 is not connected until we have measured an acceptable voltage value that will not damage the component. A multimeter on the oscilloscope measures the output voltage and confirms it is in the specified acceptable range.

3.2.2 Secondary Backup Power Source

Initially, we had planned on including a backup battery circuit that would use the battery as a backup power supply in case the primary power supply (the wall outlet) faced an outage. This source would similarly provide an acceptable range of 3.3V± 0.3V of voltage to power the ESP32. Verification of this circuit would be performed by connecting both the primary and secondary source, or signals/waveforms simulating these sources, to the backup battery circuit. The primary source would then be decreased to 0V and the output of the backup battery circuit would be measured using a multimeter.

3.3 Controller Subsystem Requirements & Verification

3.3.1 Analog Measurements are Read

The microcontroller requirements are simply that the ESP32 code should be able to access the current and voltage input signals and use them to perform calculations. Verification was performed by recording measured current and voltage values received from the analogRead() function onto a .csv file and viewing if it matches the input current and voltages. Furthermore, printing measured current and voltage onto the serial monitor was also used to more quickly test and validate the requirement, however, this method requires the microcontroller to be connected to the Arduino IDE environment to view the output.

3.4 Data Storage Requirements & Verification

3.4.1 Data Uploaded every 15 minutes

It was established that our local data would be offloaded onto the online database once every 15 minutes in order to monitor if we ever lose connection. Verification would again be done utilizing the time stamps we stored with the data, and monitoring the database to view the timestamp between uploads. We were not able to upload the data to the cloud as planned due to the MQTT client not functioning properly on the ESP32. Our cloud infrastructure was capable of ingesting data and when unit testing, it would ingest data properly within 30 sec of the scheduled data ingestion.

3.4.2 Data Measured every 0.1 seconds

A requirement for our project was that the current and voltage measurements be taken at a rate of 0.1 seconds. We were able to verify the rate at which measurements were taken by including a timestamp along with each measurement. We looked at the data stored on the SD card and found that the .csv file stored data every 1-2 seconds. We were not able to meet the requirements due to inefficiencies in our code. The code responsible for modifying the LCD and float calculations are resource intensive which results in the data being stored at a slower rate.

3.4.3 Online Storage for 5 years, Local Storage of 96 hours

Local storage of 96 hours is required in case of a WiFi or power outage, allowing the users to have data for 4 days with the hope of the user being able to detect the issue and still have data available. We are not able to comprehensively test if data integrity is maintained over 5 years, but this can be provisioned in AWS.

3.5 Display Requirements & Verification

3.5.1 Data is displayed on the LCD

Displaying data, initially current, voltage, and power but later transitioning to just power factor and phase angle, was the main requirement the display subsystem needed to fulfill in order to be of any use. Verification was performed by connecting the LCD to the GPIO pins, allowing the device to collect its first set of data, and then confirming it is visible on the LCD screen.

3.5.2 Data is displayed onto TV

Displaying the recorded data onto the ECEB lobby TV was the secondary requirement of the display requirements. This would be done by uploading our data from our server/online database onto the website that is displayed on the ECEB lobby TV. Verification would be performed by seeing if we are able to upload our data from our online database onto the ECEB website. Ultimately, we were not able to complete this requirement due to the bottleneck of the MQTT client not functioning properly. Had the MQTT client uploaded data to the cloud, we would have been able to display the data on the ECEB TV.

4 Costs & Schedule

4.1 Parts

Most of the parts that were used in the construction of the power meter were ordered off digikey, but some parts were ordered through the ECE smd component inventory which came at no cost to us. A price for these components that were ordered for free are included in the full breakdown of parts ordered in Appendix A. The total for all our parts was $96.17.

4.2 Labor

The labor involved in our project included the planning, designing, and assembling of the project. The main time cost with regards to the assembly of our project was soldering the components onto the board and testing them to see if they functioned properly. Additionally, our project had a 3D printed container used to encapsulate the PCB and hold the LCD display.

4.3 Total Calculation

Our costs include the labor and the sum of our parts, as we do not plan on using any external services such as the machine shop. Assuming an hourly rate of $40/hour, and given 11 weeks, estimating the time spent being on average 5 to 6 hours a week, we use the provided formula to get $40/hour * 2.5 * 55 hours = $6600 per individual, meaning the entire labor cost would be $19800. Adding the sum of our parts gets us to a total cost of $19800+$101.17 = $19901.17.

5. Conclusion

5.1 Results

The voltage and current sensing circuits were tested against a purely capacitive load and an RC load. Inductors were unfortunately not available in the lab and so a purely inductive load and an RLC load were not able to be tested. Table 1 & 2 shows the results of expected vs observed power factor and phase angle, as well as the percent error associated with each for both the capacitive load and the RC load.

Table 1: Power Factor for Capacitive and RC Circuits

	RC Circuit	**Capacitive Circuit**
Expected	0.88	0.00
Observed	0.80	0.12
% Error	9%	12%

Table 2: Phase Angle for Capacitive and RC Circuits

	RC Circuit	**Capacitive Circuit**
Expected	28°	90°
Observed	42°	80°
% Error	50%	11%

5.2 Future Work and Possible Improvements

The microSD card module required 5V which meant we had to use an additional source to power it as it was not properly functioning when it was being powered by the 3.3V primary source from the power supply subsystem. This issue could be resolved using a microSD card module with a built in level shifter from 3.3V to 5V allowing it to be connected to the ESP32 power supply.

Another issue present in our design was that the sampling rate was varying from 1-3 seconds as opposed to 0.1 seconds due to the amount of time it takes to run the code. Logic could be added to selectively run certain functions, such as LCD and float computations, once every N loops, reducing the time it takes for the code to run. A possible alternative solution would be

utilizing parallel loops, one for the analog measurements and the other for display and data storage code. This is possible as the ESP32 is a multicore microcontroller, however it is recommended not to use the third low power core so we would be restricted to two parallel loops [1].

Finally, our circuit was directly powered by the battery meaning it would need to be replaced every so often which would be inconvenient for users. Initially, we had simulated and designed a backup battery circuit to solve the issue but did not end up implementing it due to the need of additional components to interface the outlet with the circuit. The schematic design is implemented in Figure 8, primarily relying on a PMOS gate to open and allow access to the secondary source when the primary source reaches a low enough voltage.

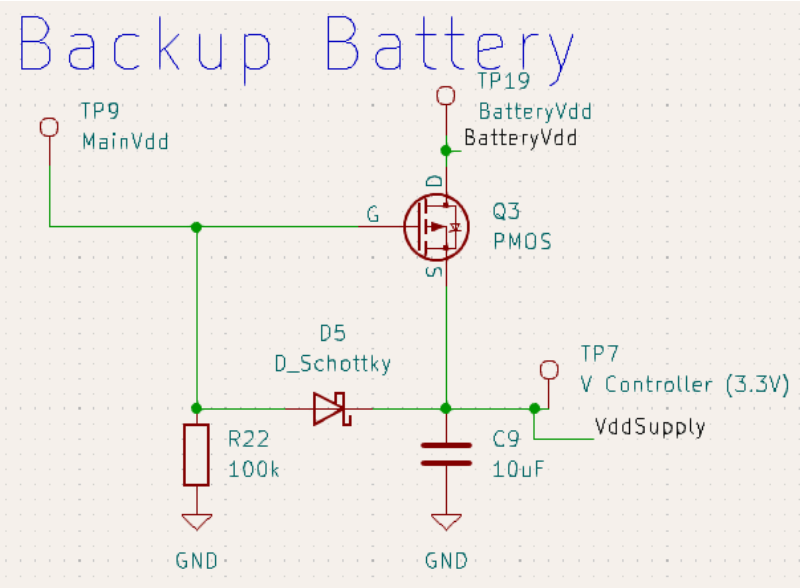

Figure 8: The backup battery design we planned on using

5.3 Ethical and Safety Considerations

With regards to safety we are going to need to take into consideration the CAT Rating. This relates to safety equipment ratings needed when handling different types of electrical devices or systems, which can pose a danger in the event of arc flashes[6]. Our device is a CAT II rating as it will be attached to a single-phase AC load. Ideally we would need to take into account arc flashes and sudden harmonics in the power grid which could cause irregularities when measuring voltage and current. Ultimately, we ended up simulating the load using signal generators in the lab during the demo, so these safety considerations were not as much of an issue as if we were to have actually connected to the load.

We will make sure that the data presented is allowed to be public, and that displayed values are not manipulated. This falls under maintaining integrity aspect of the IEEE code of ethics by not lying about our results to improve perceived accuracy [7]. When handling

potentially private data we must take the proper precautionary methods to make sure that this data is not shared, which is why we will only display public data on to the TV's [8].

References

[1] F. K. my Site!, "ESP32 With Arduino IDE - Multi-Core Programming," *Instructables*. https://www.instructables.com/ESP32-With-Arduino-IDE-Multi-Core-Programming/ (accessed Apr. 30, 2024).

[2] *IG REG BUCK ADJ 2A 8SOIC,* datasheet, Texas Instruments, October, 2016. Available at: https://www.digikey.com/en/products/detail/texas-instruments/TPS54231DR/1960936

[3] *ESP32*, datasheet, Espressif Systems, 2024. Available at: esp32_datasheet_en.pdf (espressif.com)

[4] *Dual single-supply, rail-to-rail, low power operational amplifier*, datasheet, Texas Instruments, August, 2016. Available at: OPAx340 Single-Supply, Rail-to-Rail Operational Amplifiers MicroAmplifier Series datasheet (Rev. C)

[5] *Character Display Module LCD - Color LED I2C, Serial*, datasheet, Sunfounder. Available at: CN0295D DATASHEET.pdf (digikey.com)

[6] N. Fire and N. Fire, *NFPA 70E*. National Fire Protection Association (NFPA), 2004.

[7] IEEE Code of Ethics. (n.d.). https://www.ieee.org/about/corporate/governance/p7-8.html

[8] "IEEE SA - IEEE 7002-2022," IEEE Standards Association. https://standards.ieee.org/ieee/7002/6898/

[9] ECE 445, ESP32-S3-WROOM EXAMPLE BOARD: MOTOR CONTROLLER, web page. Available at: :: ECE 445 - Senior Design Laboratory (illinois.edu)

Appendix A: Cost Table

Table 3: Parts and Cost

Name	Quantity	Manufacturer	Link	Cost
SD Card Module	1	DFRobot	SD Card Module	$5.20
Voltage Transformer	1	Triad Magnetics	Voltage Transformer	$12.02
Buck Converter	1	Texas Instruments	Buck Converter	$1.20
Current Transformer	1	AcuAmp	Current Transformer	$26.00
Backup battery circuit	1	Texas Instruments	Backup battery circuit	$4.20
Linear Regulator	1	Texas Instruments	Linear Regulator	$1.51
ESP32 (Microcontroller)	1	Espressif Systems	ESP32	$2.68
LCD Display	1	UNIVERSAL-SOLDER Electronics	LCD	$3.66
Micro SD Card	1	SP Silicon Power	MicroSD Card	$9.98
12V Rechargeable Battery	1	Mighty Max Battery	Battery	$19.99
Op Amp OPA2340UA	4	Texas Instruments	Op Amp	$1.819 (each)
12V 2400mAh Battery	1	ECE Supply Center	ECE Supply Center	$0.17
10 μH Inductor	1	CoilCraft	DigiKey	$0.9375
0.1 μF Capacitor	6	Murata Electronics	DigiKey	$0.10
1.0 μF Capacitor	2	YAGEO	DigiKey	$0.10
10 μF Capacitor	1	Murata Electronics	DigiKey	$0.10
100 nF Capacitor	1	Samsung Electro-Mechanics	DigiKey	$0.10

0.01 μF Capacitor	1	KEMET	DigiKey	$0.10
47 pf Capacitor	2	KEMET	DigiKey	$0.57
1000 pF Capacitor	1	ECE	ECE Supply Center	$0.10
47 μF Capacitor	2	Samsung Electro-Mechanics	DigiKey	$0.26
4.7 μF Capacitor	2	Sasmsung Electro-Mechanics	DigiKey	$0.10
0.015 μF Capacitor	1	KEMET	DigiKey	$0.38
Schottky Diodes	5	Diodes Incorporated	DigiKey	$0.41
Phoenix Contacts	4	ECE	ECE Supply Center	$0.94
1x04 Pin Header	1	Wurth Elektronik	DigiKey	$0.19
1x06 Pin Header	1	Adam Tech	DigiKey	$0.20
BJTs	2	ECE	ECE Supply Center	$0.42
10mΩ Resistor	1	YAGEO	DigiKey	$0.10
2kΩ Resistor	1	YAGEO	DigiKey	$0.10
3.24kΩ Resistor	1	YAGEO	DigiKey	$0.10
10kΩ Resistor	14	YAGEO	DigiKey	$0.087
27kΩ Resistor	2	YAGEO	DigiKey	$0.10
30kΩ Resistor	1	Panasonic Electronic Components	DigiKey	$0.37
68.1kΩ Resistor	1	TE Connectivity Passive Product	DigiKey	$0.75
100kΩ Resistor	2	YAGEO	DigiKey	$0.10
110kΩ Resistor	1	YAGEO	DigiKey	$0.10
332kΩ Resistor	1	Panasonic Electronic Components	DigiKey	$0.37

Button Switch	2	C & K	DigiKey	$0.18
TestPoint	12	Keystone Electronics	DigiKey	$0.38
PCB	10	PCBWay	PCBWay	$5.00
			Total Cost:	**$101.17**

Appendix B: Requirements and Verifications Tables

Table 4: Power Subsystem Requirements and Verifications

Requirement	Verification
1. The system must convert the battery voltage to 3.3V ± 0.3V in order to power the ESP32.	1a. Connect our device to a power source. As a preventive measure it would be best not to connect the ESP32 until we have measured an acceptable voltage value that will not damage the component. 1b. Using the multimeter on the oscilloscope, measure the output and it should yield a reading of 3.3V ± 0.3V.
2. The system must be able to supply power from a battery of 3.3V ± 0.3V in the case of a power outage.	2a. Begin operation of our system and once it is measuring (on) make sure the battery is connected. 2b. Remove the primary power source and observe if our submeter is still being powered by using the voltmeter on the oscilloscope to measure 3.3V ± 0.3V.

Table 5: Measuring Subsystem Requirements and Verifications

Requirement	Verification
1. Voltage and current circuits bring voltage and current into an acceptable range for the internal ESP32 ADC input such that it does not damage the device.	1a. Once our measuring circuits are connected to the load, and the components are connected to the power source, we will use the oscilloscope multimeter to measure the current, which should be 40mA ± 15mA, and the voltage, which should be 3.3V ± 0.3V.
2. Our measuring circuits should not significantly modify or change the values of voltage and current when bringing these values to a safe range for our ESP32.	2a. After recording data while operating our device we will compare the measured power and power factor to the actual values, and the percent error should be within 15%

Table 6: Controller Subsystem Requirements and Verifications

Requirement	Verification
1. Current and voltage inputs are received and the power and power factor are calculated and recorded.	1a. This will be verified through viewing the data sent to the server and seeing if any values are there, it should not be N/A or zeros.

Table 7: Data Storage Subsystem Requirements and Verifications

Requirement	Verification
1. Data will be uploaded onto a time series database(TSDB) through MQTT protocol at least 4 times an hour.	1a. After 15 minutes of operation we will access the server to see if our first batch of data has been stored.
2. Voltage, current, and power will be recorded 10 times a second.	2a. Using a timestamp on each data measurement, we will observe if the interval is 0.1 seconds between samples by checking the server.
3. Data will be stored and maintained for 5 years. The local memory storage will need to hold 96 hours of data.	3a. After 96 hours, we will verify that the oldest data points are 96 hours old. This will be done by manually going through the local storage and viewing the timestamps of the data sent. Data stored on the cloud is managed by cloud providers so we can only apply settings to retain data for 5 years.

Table 8: Display Subsystem Requirements and Verifications

Requirement	Verification
1. Display current, voltage, and power values on LCD.	1a. The first set of collected data should be displayed on the LCD screen and correspond to the data. This will be verified by connecting the LCD to the GPIO pins and observing if the values are displayed.
2. Display current, voltage, and power values onto a TV.	2a. Once our device has collected its first set of data it should be visible on the TV and correspond to the data. This will be done by connecting to a TV in the ECEB by sending the data from the database to the TV display via an API request.

Appendix C: Schedule

Week	Tasks	Person
2/29	Finish Design Document	Everyone
	Project Proposal regrade	Everyone
2/26	Simulate measuring subsystem	Aleksai, Jonathan
	Begin to order parts	Everyone
	Finish initial PCB layout for PCB review	Everyone
	Research cloud solutions for data storage/website hosting/MQTT broker	Mike
3/4	First round Order of PCB	Everyone
	Simulate MQTT publish and subscribe with example data and ingest into a Time series DB	Mike
	Write code to interface with LCD display	Jonathan
	Work on code used to calculate power measurements on ESP32	Aleksai
3/11	Spring Break	Spring Break
3/18	Second Round Order of PCB	Everyone
	Assemble current and voltage measuring circuits	Jonathan, Aleksai
	Build backend of website ● API calls/structure ● SQL queries	Mike

	Set up web server for website hosting	Mike
3/25	Third Round order of PCB	Everyone
	Test and verify if data is being stored onto SD card	Aleksai
	Test current and voltage measuring circuits	Jonathan
	Connect TSDB, backend, and sample MQTT and get it to display sample data on a website (very rough data)	Mike
4/1	Fourth Round order of PCB	Everyone
	Debug any areas/subsystems which are not meeting required verification	Aleksai, Jonathan
	Upload data from the SD card to the cloud through MQTT	Mike
	Ingest legitimate data from the meter and display onto the website	Mike
4/8	Fifth Round order of PCB	Everyone
	Add data visualizations on the website	Aleksai, Mike
	Display the website onto TV of the ECEB	Jonathan, Mike
4/15	Prepare for Mock Demo	Everyone
4/22	Final Demo	Everyone
	Prepare for Mock Presentation	Everyone
4/19	Final Presentation	Everyone
	Complete Final Paper	Everyone

	Turn in lab notebook	Everyone