ECE 445: Senior Design Laboratory

Fall 2021

Advanced Interface Box for Solar Panels

Final Report

By:
Maram Safi (msafi2)
Nikhil Mathew Sebastian (nikhils4)
Sydney Li (sydneyl3)

TA: Evan Widloski

December 2021

Abstract

The “Advanced Interface Box for Solar Panels” is a monitoring system that has been designed for the 60 research solar panels on the Electrical and Computer Engineering Building (ECEB) roof. The primary goal of this design is to provide researchers and ECEB personnel remote access to real-time solar panel data, as well as options to configure panel monitoring sections, to allow for research studies and safe monitoring. This design also keeps in mind the need to ensure safety by checking thresholds indicating overvoltage, overcurrent, or overheating in order to prevent disastrous failures or collateral damage to the building and/or solar panels. This solution contains a remote user interface (“Research Hub”) to allow for ECEB personnel to access the solar panel data and also edit system configuration and monitoring settings of solar panels, all of which are implemented wirelessly to allow for scalability and practicality. The device was successfully designed, tested, and implemented, but the complete integration was not successful in a stand-alone setting.

Contents

1 Introduction

This report details the design, implementation and verification behind the “Advanced Interface Box for Solar Panels” project. This interface box is meant to observe solar panel data from the roof of the ECEB and wirelessly update the same on an external server and portal (“Research Hub”), while also being able to receive configuration settings for monitored panels. In the end, the device was successfully designed, tested, and implemented, but the complete integration was not fully successful in a stand-alone setting.

1.1 Problem Statement

There are 60 solar panels on top of the ECEB building, currently used for research, which are not producing any power as of now and can potentially be integrated into the power grid. Additionally, they are not adequately monitored at the moment and this poses a large hazard, especially considering there are no protection interfaces between the panels and their connections to the distribution box.

In the Fall 2019 semester, a team of students was able to create an interface which was able to display a single panel's voltage and current, but the solution could not be scaled up to interface with multiple panels as is required [1]. Each solar panel was directly connected to an ethernet cable that would only allow for the display of that particular solar panel onto a remote display. Not only was this a physical impracticality due to the ethernet cables, there was also no way to collectively display the data of multiple solar panel’s parameters onto a singular remote display. The solar panels also have designated box mounts (Attabox) which also gives us a constrained size (8 x 6 x 4 inches) in which our solution must fit in..

1.2 Solution Overview

Our solution is to design a smart interface box for these panels to allow for large-scale system behavior and output monitoring, as well as to support panel up-keep, to prevent any potential disasters like fires from occuring. The goal of our project is to monitor and maintain the research solar panels. We also plan to interface with multiple solar panels to produce a single wireless gateway of information that feeds into a visually attractive remote access portal for observation and access to research panel data. In addition, it will also provide a means to control the solar panel’s configuration.

The system will have a 12-volt isolated power supply in order to provide power to the microcontroller and other respective components. The power generated by each monitored solar panel will run through our smart interface box, giving us the ability to detect overvoltage and overcurrent conditions and disconnect the panels if necessary to prevent hazardous situations. Other features of the box will include reconfigurable tapping to allow users to determine which solar cells are being observed. The various configurations are determined through a relay subsystem where we are able to output a set number of solar cells (32-cell output, 64-cell output, and 128-cell output) for researchers conducting experiments on solar panels. For example, choosing a 128-cell output will produce more voltage than the 32-cell output configuration [2]. This is due to the fact that the surface area of the solar panels used to capture the sunlight is larger therefore, producing more power. This configuration of solar panels is controlled through a wireless interface, allowing users to configure and monitor the solar panel remotely through an external server/portal.

1.3 Visual Aid

Figure 1 below is a visualization of how our solution would fit into the existing layout for solar panels:

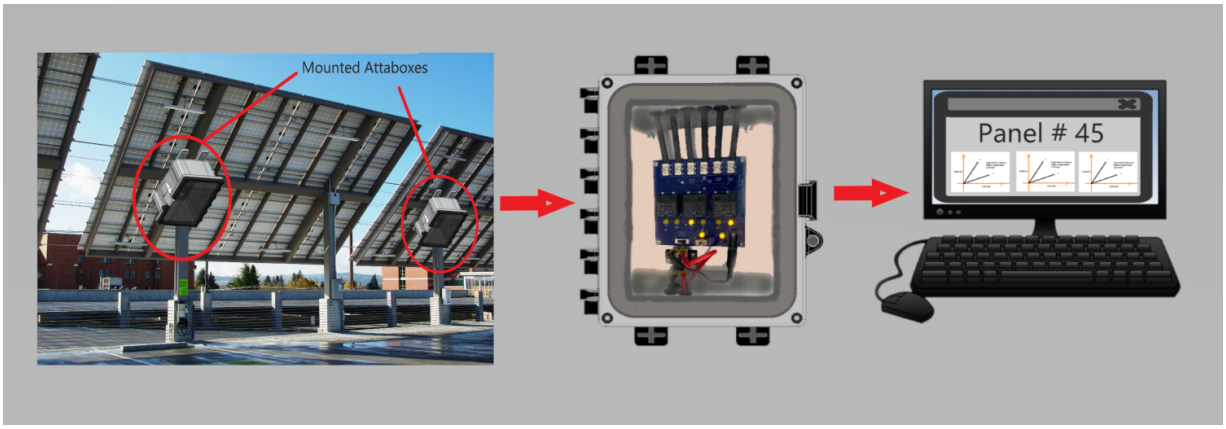

Figure 1: Diagram of the connections between different components of our solution

1.4 High-Level Requirements

1. Record 3 key solar panel parameters at the interface box level: (Variations in reading these values is due to the nonlinear nature of the microcontroller's ADC pins)
    a. Voltage – Accurately monitor the voltage of the relay output within an error range of ± 300 mV [3]
    b. Current – Accurately monitor the current of the relay output within an error range of 1.5% [4]
    c. Temperature – In ranges of -10°C to +85°C, expect ± 0.5°C accuracy [5]
2. Wireless communication capability with interface box through a remote external access portal that shows researchers intuitive visualizations of obtained solar panel data for observation as well as configuring which section of the solar panel is being monitored.
3. Scalability – capability of interface box retrieving data and communicating with more than one solar panel. Starting with one solar panel and interface box, we want to be able to show scalability to a set of two solar panels and their corresponding interface box.

The above high-level requirements directly fall in line with our project's overall purpose. Measuring the key solar panel parameters is critical to support research endeavors with ECEB's solar panels, as well as monitor the panels for safety reasons. The wireless communication requirements ensure that these observed values can be shared and observed while also providing configuration options remotely. Finally, scalability is a key requirement as our final solution should be able to be applied for all 60 of the solar panels available, something that was not fully met by the previous attempt at this project.

1.5 Subsystem Overview

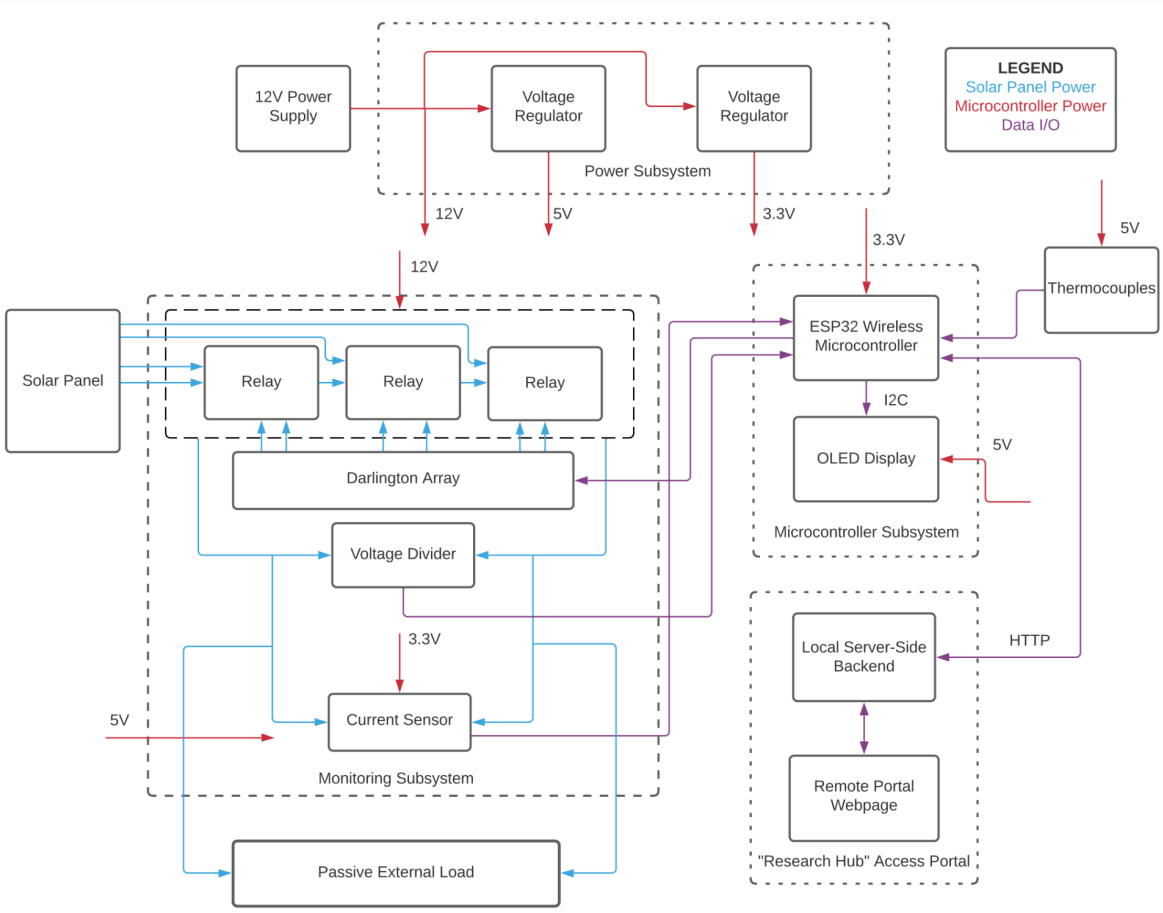

Figure 2: Labeled block diagram for an entire Advanced Interface Box system

Our top-level block diagram provides a high-level overview of our four monitoring systems. The power subsystem's primary function is to convert a 12-volt external supply and provide 3.3-volt and 5-volt supply lines to power up various components.

Our monitoring subsystem is responsible for relaying information about the voltage and current measurements to the ESP32 microcontroller. Within the monitoring subsystem, the ESP32 also dictates which of the relays are closed to determine which configuration of solar cells the user would like to measure and output to the load.

The microcontroller subsystem consists of our ESP32 microcontroller as well as our OLED display. The voltage, current, and temperature readings of the solar panels will be wirelessly communicated to our external "Research Hub" access portal through our microcontroller. These same readings will also be shown on the OLED display mounted inside the Attabox for convenience.

The Research Hub subsystem encompasses the remote access server and online portal to which data will be sent from monitored solar panels, and also from which configuration settings can be input by the user. The Research Hub subsystem is implemented wirelessly, has security access protocols in place, and is designed for scalability.

2 Design

This section of the report details the planning and design of our project including the four main subsystems, all additional components, as well as the adjustments we made in the final implementation.

2.1 Subsystem Descriptions

We begin this section by describing the details behind our design process for our four main subsystems.

2.1.1 Power Subsystem

Our power subsystem is driven by an external 12-volt supply that we are able to step-down to power the individual components and chips that create our smart interface box. This 12-volt supply voltage is stepped down through two linear regulators to provide 3.3 V and 5 V supplies [6]. The thermocouples and current sensor require a 3.3 V supply, whereas the OLED, ESP32 microcontroller and ADC converter are powered with a 5 V supply. Lastly, the darlington array takes in a 12 V input straight from the source [7]. In addition, LEDs are also available on the PCB to indicate whether the 3.3 V and 5 V supplies are available as shown in Figure 1 in Appendix B.

2.1.2 Monitoring Subsystem

Relays are utilized to configure the number of solar cells that are being monitored by the interface box. The different configurations of solar cells include 32, 64, and 128-solar cells (refer Figure 8 in Appendix B). Our relay configuration is composed of 2 main components: the darlington array and three ultra-small high-voltage DC FTR-J2 series relays [3]. Control signals are outputted from our microcontroller into our darlington array, which acts as a relay driver between the ESP32 and our relays as shown in Figure 2 in Appendix B. The outputs of our darlington connect to the electromagnetic side of our relay, which allows for physical switching to occur to our desired configuration.

Once the desired configuration is determined by the user, the solar output feeds directly into our voltage divider to step-down the voltage between 0 V and 3.3 V, the acceptable range for our microcontroller's ADC pins. In our initial design shown in Figure 4 in Appendix B, the output of the voltage divider is connected to a MCP3428 16-bit ADC in order to provide galvanic isolation between the voltage of the microcontroller and the voltage output of the solar panels [8]. This is a safety measure implemented so the user is not exposed to high voltage in the event that they come in contact with the microcontroller. The current measurements are obtained through an ACS723 current sensor. This sensor detects the amount of current that flows through it and outputs a corresponding voltage to our microcontroller. Our current sensor is also protected with galvanic isolation [9].

On the firmware side, the voltage measurement and the current sensor are routed to the ESP32 through its ADC pins as shown in Figure 3 in Appendix B. These pins read a "raw value" from which a raw voltage is calculated using power supply and reference voltage. For the voltage measurement, we simply need to reverse engineer the input voltage via a voltage division, while calibrating the initially measured and reverse engineered voltage values (shown in Figure 1 in Appendix C). For the current sensor, we take the raw voltage and then use the sensor sensitivity to calculate a raw current, which is then also calibrated (shown in Figure 2 in Appendix C).

2.1.3 Microcontroller Subsystem

The ESP32 microcontroller we are using is responsible for communicating with our monitoring system to gather measurements for our smart interface box. Our initial design as shown in Figure 5 in Appendix B, had the ESP32 taking in voltage readings via the ADC through the I2C bus, current readings directly into an on-board ADC pin, and temperature readings as a digital signal direct to the ESP32.

The OLED is what displays the latest set of measurements gathered from the panel. It is programmed by the ESP32 via its I2C bus to display real time solar panel parameters (refer Figure 6 in Appendix B) [10]. The decision to implement an OLED visual display is solely for convenience to observe the solar parameters on site at each individual smart interface box.

2.1.4 Research Hub Subsystem

An external web-based server system – Research Hub – is set up with two-way wireless communication with the interface box for easy management of the panels. Only authorized ECEB personnel will have access to this portal and it is intended for internal use, so the interface will be secured as such. The focus is on being able to receive and monitor key solar panel data points, as well as porting the in-box switching capabilities to a remote setting.

The Django framework has been used to manage the front-end webpage and the back-end database system for managing data from the interface box. The front-end will be scripted in HTML for the visual structure, while using Python for managing Django applications for different website functionalities. It is through this Django project that the security/user authentication and navigable routing within the webpage will also be handled. The wireless capabilities of the Django project have been implemented using the Django-REST framework which facilitates the two-way communication requirement from the portal's end. At the microcontroller (ESP32) level, it has been programmed to act as both a client and an asynchronous server through the Arduino IDE in order to facilitate the same two-way communication channel. All of this communication will be done through back-and-forth HTTP protocol (POST) requests. Appendix D contains flowcharts depicting these parallel functionalities of the software subsystem.

The primary goal of the external portal is **data observation**. This ensures that the data collected and reported for monitoring purposes will be stored and presented on the access portal. A web-based access of the data could also be adapted and provided for public display as well. Data visualization has been achieved with the Python-based *plotly* library along with HTML to build it up on a webpage. This is built on top of the aforementioned Django project, with a dedicated model for each monitored solar panel. This data is updated in real time from the microcontroller every one minute, allowing for a constant stream of data. The external portal itself is built with three key functionalities in mind (implemented through the Django project/applications) – managing multiple solar panels while providing each panel dedicated pages, handling user authentication, and plotting obtained data for observation and research. At the microcontroller level we have set fixed thresholds of 40 V, 2 A, and 40ºC past which if measured the microcontroller forces physical connection to the solar panel (relay configuration) to be closed.

The second goal of this external portal is **wireless panel configuration**. The objective is to port the physical switch capability and the microcontroller-based panel section adjustments so as to allow for remote access. Since these are controlled through the switching subsystem and relays, we process on the

front-end the three specific configurations to be chosen from and sent through the microcontroller to the relay system. This will use the same structure as mentioned above, utilizing the Django-REST framework on the back-end for communication to the microcontroller through HTTP requests, while using Django forms and views to process user input through GUI.

2.1.5 External Components

We have four additional components that do not fall exactly into any one subsystem but are still critical to our project. Of these components, the thermocouples are verified as part of the monitoring subsystem.

2.1.5.1 Solar Panels

The solar panels we are using are SPR-425E-WHT-D panels which have a peak power of 425 W ± 5%. The open-circuit voltage and the short-circuit current are 85.6 V and 6.18 A respectively, and if maximum power point tracking were to be implemented into our project in the future, the voltage at the maximum power extraction point (V_{mp}) would be 72.9 V and the current at the maximum power extraction point (I_{mp}) is 5.83 A. All these following ratings as specified in the datasheet are under standard test conditions of 1000 W/m^2, AM 1.5, and 25 ºC [11].

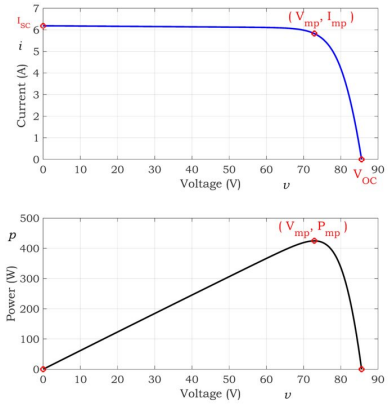

Figure 3: Current-voltage and power-voltage characteristics of the solar panel under standard test conditions

2.1.5.2 12-Volt Power Supply

An on-site 12-volt power supply line runs through all 60 solar panels which powers up the components of our smart interface box. This supply line includes a connection to ground.

2.1.5.3 Thermocouples

A pair of thermocouples will be attached to each solar panel, one at the center of the panel and another at the edge to capture different temperature readings. The thermocouples utilized are the waterproof DS18B20 digital temperature sensors. The DS18B20 has a precise 1-wire digital temperature sensor that connects to the digital pins of our ESP32 microcontroller and gives up to 12-bits of precision with an onboard digital-to-analog converter as shown in Figure 7 in Appendix B. Each thermocouple has a unique 64-bit ID on the chip allowing us to distinguish between different temperature readings on the software side even when reading multiple sensors through the same ESP32 pin (as has been implemented) [5].

2.1.5.4 Passive External Load

The output load of our solar panel is configured by the output of our relays. By driving control signals from our microcontroller through our darlington array, we decide which cell configuration will feed into our external load. Our load will consist of entirely passive components and mostly resistive just to dissipate the power. Once the design can prove that it works as expected, then additional features such as energy storage, satellite imaging, and the possible integration of power into the grid are all possible upgrades.

2.2 Supporting Material

Since our final implementation needs to be mountable on the ECEB roof's solar panels, we have also planned out the physical details of our enclosure and attaching our box to specific panels.

2.2.1 Mechanical Design and Mounting Diagrams

Figure 4 below provides a visual representation of the box enclosure that we will use on each solar panel:

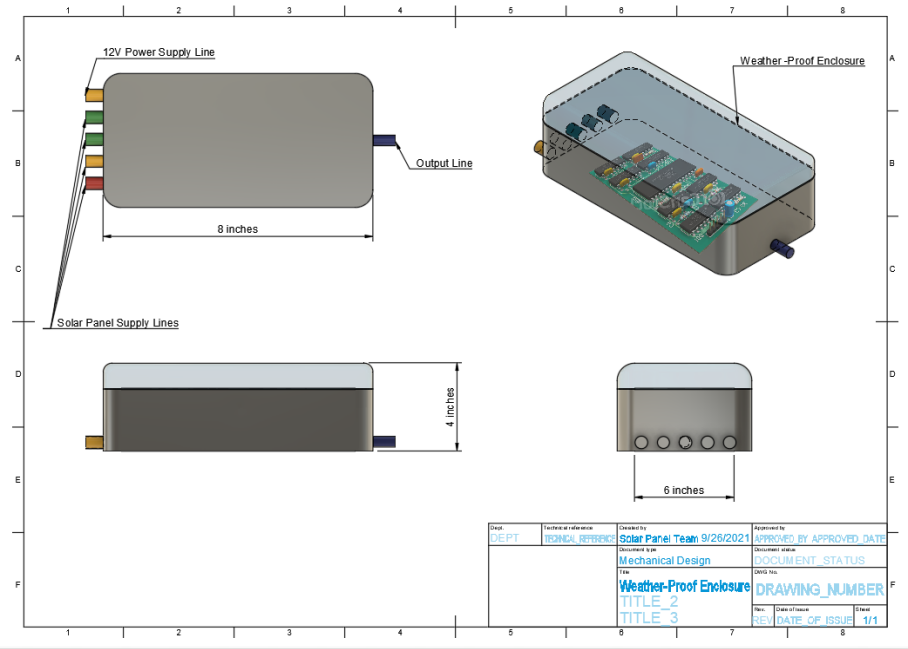

Figure 4: 3-D visual aid for the box enclosure

2.2.2 Mechanical Design and Mounting Explanation

The mechanical design of our project consists of a weatherproof enclosure. This was one of our design constraints because these enclosures were purchased for all 60 panels before our team began to work on this project. Our enclosure constraints are 8 x 6 x 4 inches as referenced from our enclosures datasheet [12]. This box is to be mounted on the backs of the solar panels on the roof of ECEB as shown in Figure 14. We know this is feasible as this design and mounting process have been implemented on eight of the 60 solar panels that are currently being housed on the roof.

Figure 5: Mounted Attaboxes on the solar panels on the roof of the ECEB

2.3 Design Alternatives/Adjustments

Throughout the modular testing of our design, there were adjustments to be made in order for our PCB to function. Starting with our power subsystem, we had to refine our design to make it suitable for fixed linear regulators instead of adjustable linear regulators due to the components being out of stock. We made this accommodation by shorting across the resistors that were meant to adjust the output of the intended linear regulators.

Moving onto our microcontroller subsystem, one of our initial changes to the design was modifying our enable pin so that the pin was left floating and not tied down to ground as in the initial design. We then had to move 4 out of 6 darlington control signals due to the fact that they were connected to input-only pins on the ESP32. When conducting modular testing on our thermocouples, we recognized that we would only need one GPIO pin to read measurements from both of our thermocouples on the microcontroller instead of the two pins that were originally planned. Another adjustment that had to be made to our microcontroller subsystem is the addition of pull up resistors to the SCL and SDA lines of our OLED – using these resistors allows us to properly communicate with the display.

During the testing of the monitoring subsystem, the communication between the ADC converter did not function as expected. The ADC was implemented in our design to provide isolation in our circuit separating the high voltage side of the solar panels from the low voltage side of the microcontroller. When sending out the opening address byte to the ADC from our microcontroller, as seen in Figure 6 below, the ADC did not return an acknowledgement bit, therefore it rendered our attempt at communication ineffective. Because of this, the ADC was omitted from the final design board. In order to measure the voltage without the utilization of the ADC, the voltage divider output was wired to manually connect to the input of the ESP32 microcontroller. Separate ground connections then had to be tied together to receive correct measurements for our voltage and current.

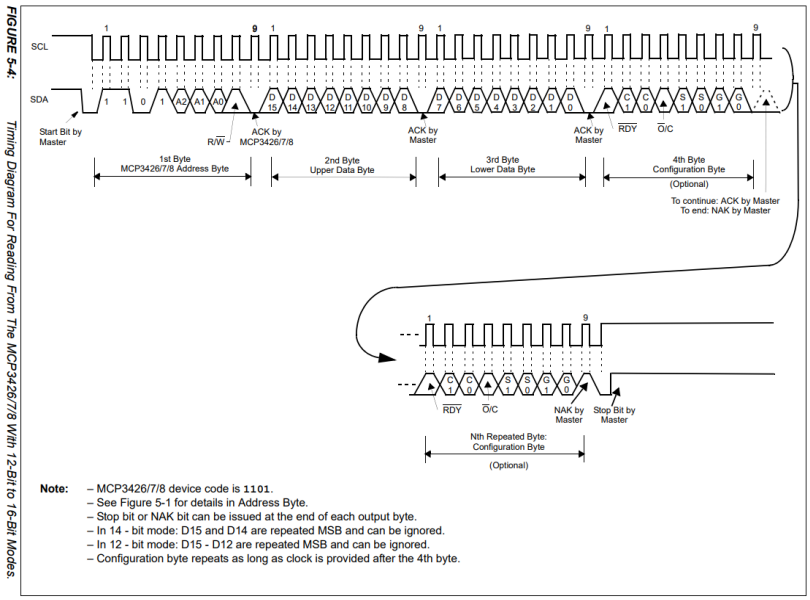

Figure 6: SDA and SCL waveforms as expected when utilizing the MCP3428 ADC converter

For our relay configuration, while our control signal pins on our microcontroller needed to be adjusted, our relays and darlington array inputs and outputs did not change during the testing of our PCB. However to achieve access to all configurations outputting from a solar panel, there needed to be an adjustment made to our relay arrangement. In our original design, we had control signal 2 as an input into pin 5B of our darlington array. However it was evident that the outputs of our darlington chip had no connection between 5C and our relays. Therefore, we changed our configuration in order for control signal 2 to route to pin 7B, which can then be correctly connected to our relays and achieve switching in Panel_C.

Finally for our external components, a main adjustment in our design was the addition of pull-up resistors to our thermocouples. Just as in our OLED, these resistors are what allowed our thermocouples to be registered by our microcontroller. We would also advise as an improvement to use a screw terminal instead of a pin screw for the isolated 12 V supply in order to accomodate for the 12 V supply line that can be already found on the roof of the ECEB.

3 Design Verification

This section of the report details our functional tests and verifications of requirements for each subsystem.

3.1 Power Subsystem

The power subsystem was a critical component of our design because it powered the entire PCB. It required some initial debugging, but eventually we were able to output 3.3-volt and 5-volt lines accurately as specified in our requirements (refer Table 1 in Appendix A). The 12-volt output was also within our tolerance range accounting for the resistivity and losses between copper traces. Figures 7 and 8 below show the successful results on probing the outputs of this subsystem:

1. 12-volt power rail provides a 12-volt DC input within the range of ± 1% [13]

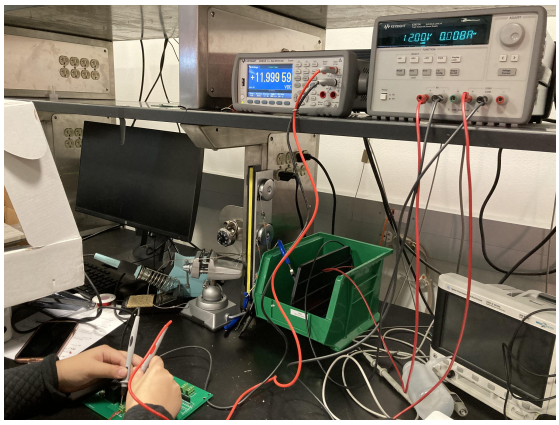

Figure 7: Providing 12-volt input supply to our PCB and measuring a 11.99-volt output voltage on the multimeter

2. Converts a 12-volt input into a 5-volt DC output within the range of 4.90 V ≤ Vout ≤ 5.10 V
3. Converts a 12-volt input into a 3.3-volt DC output within the range of 3.235 V ≤ Vout ≤ 3.365 V

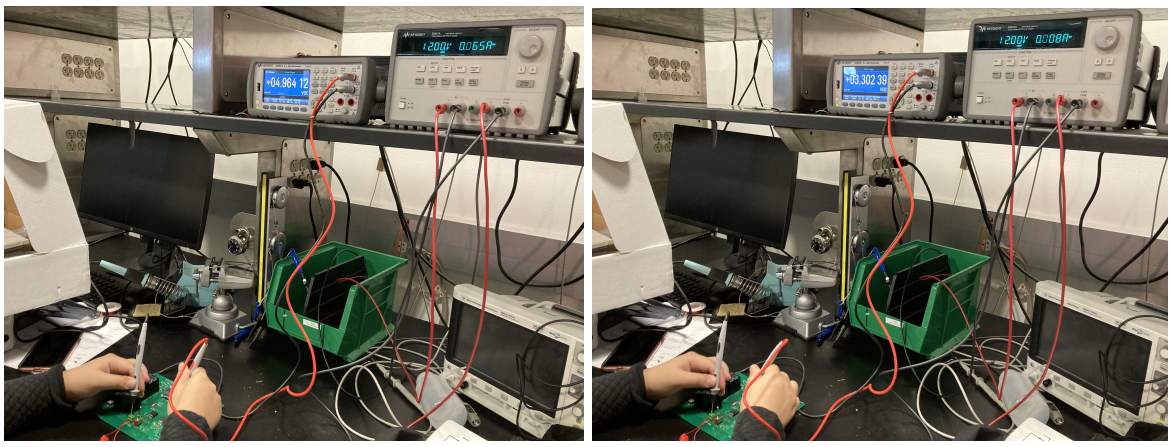

Figure 8: Providing 12-volt input supply to our PCB and measuring a 4.964-volt and a 3.302-volt output voltage on the multimeter from the respective regulators

3.2 Monitoring Subsystem

The first requirement of the monitoring subsystem is management of configuration between monitoring

32, 64, or 128- solar cells by the interface box. The relay subsystem must be able to configure and choose between the different solar cell configurations: 32-cells (CD), 64-cells (BC), and 128-cells (AD). We were able to successfully switch between all but one configuration due to the partial functionality of the darlington array which sustained some damages due to it being initially improperly soldered on our PCB. Once we realized the mistake and changed our approach, each darlington pair was driving their respective relays, except for the damaged one (Panel_C) which also affected our ability to measure voltage and current together because of the inability for that singular solar panel input to be connected to the load. Figure 9 below shows our successful continuity test to prove our working relay outputs were as expected:

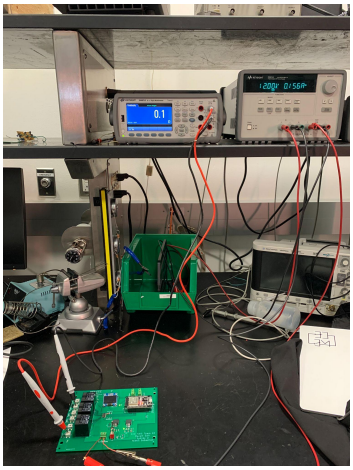

Figure 9: Utilizing a multimeter for a continuity test to ensure that the right panels are connected to the load at specific configurations

The monitoring subsystem is also responsible for displaying the key measurements of our solar panels, i.e., voltage, current, and temperature. Starting with voltage, the voltage divider could step down voltages as high as 85.6 V down to values under 3 volts, which is the maximum voltage input for the ADC pins on our microcontroller. At the beginning of our PCB testing, we attempted to test our ADC on an isolated SOIC breakout board using the connection seen in Figure 10 below. As aforementioned (Section 2.3), we were not able to receive an acknowledgement back from the ADC, meaning that we were not able to receive an output from the converter, which eventually led to us bypassing the chip.

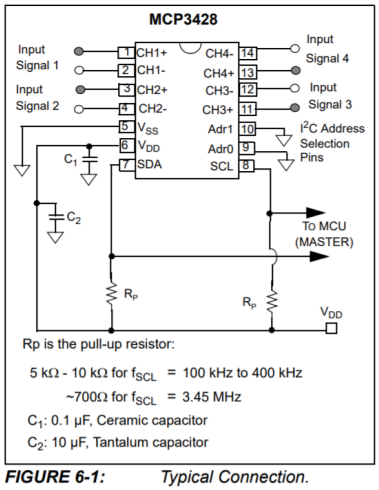

Figure 10: ADC converter typical connection per datasheet (Rp = 10kΩ)

But, our voltage divider itself met the requirement that it must be able to step down input voltages within the ranges of 0 V – 85.6 V and output corresponding voltages within the range of 0 V – 3.3 V [14]. Figures 11 and 12 below show our PCB testing of the same, with the OLED displaying “Meas. Vol” (calibrated output of the voltage divider), “Real Vol.” (voltage that is scaled up but not calibrated), and “Adj, Voltage” (calibrated voltage at the load):

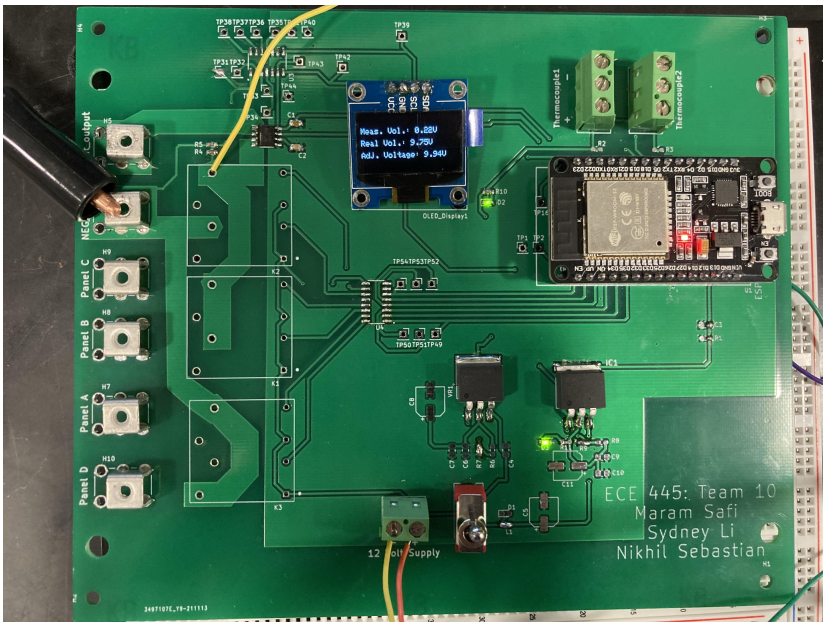

Figure 11: Probing a 10-volt supply to the inputs of our voltage divider to measure the voltage

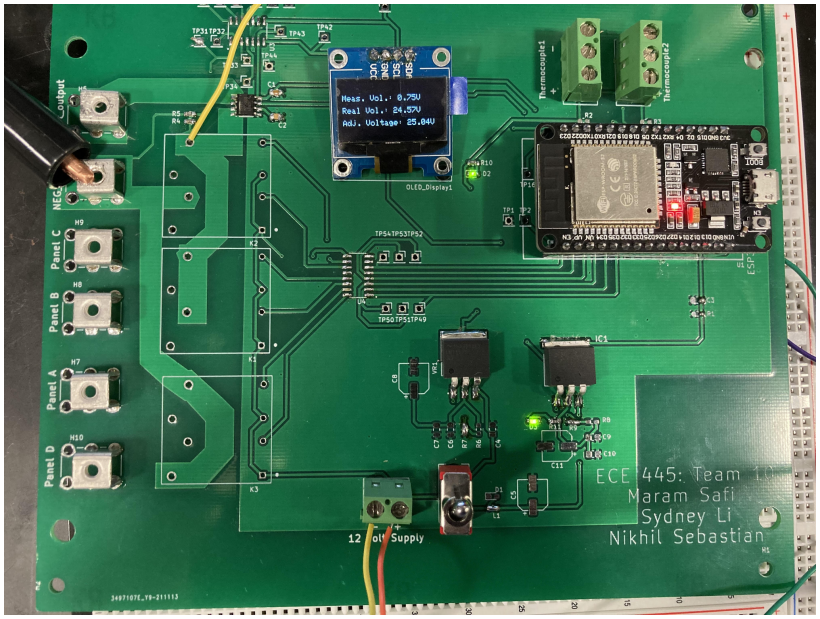

Figure 12: Probing a 25-volt supply to the inputs of our voltage divider to measure the voltage

Moving onto the current sensor, it initially provided amperage with an error of around ± 300 mA but after calibrating that, we were able to have an error within our tolerance range of 1.5%, hence fully meeting our measurement and accuracy requirements. Figure 13 below shows the serial monitor output of the ESP32

when taking current measurements, with print outs of “Raw Value” (analog output), “mV” (corresponding calculated voltage), and “Amps Adj.” (calibrated current at the load):

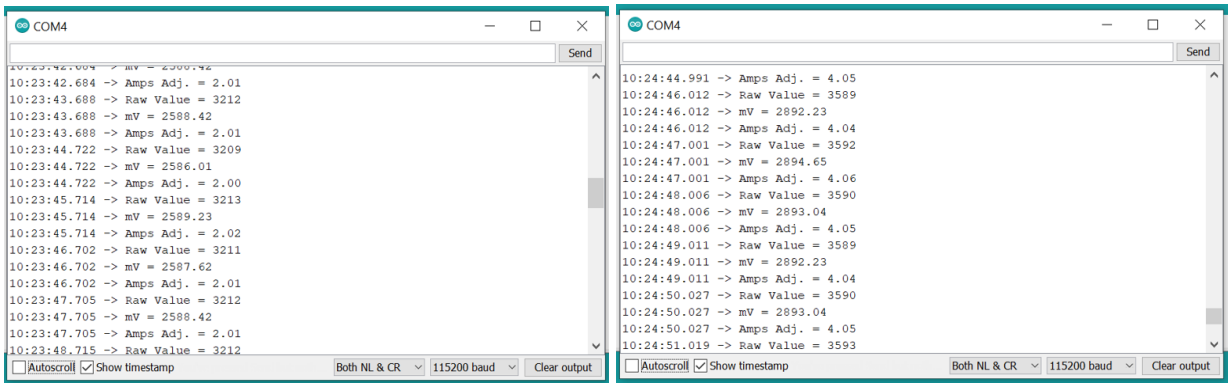

Figure 13: Outputs at the microcontroller when probing a 2-Amp supply (left) and a 4-amp supply (right) to the inputs of our current sensor and measuring them

Finally, the thermocouples were also able to measure a digital temperature and responded accordingly when introduced to a manual heat signature. We were able to verify that the thermocouples fulfilled their requirements of measuring values between the range -10°C and 85°C within an accuracy of ± 0.5°C [5]. Figure 14 below shows the serial monitor output of the ESP32 when taking temperature measurements, with print outs of the actual measurements in degrees Celsius:

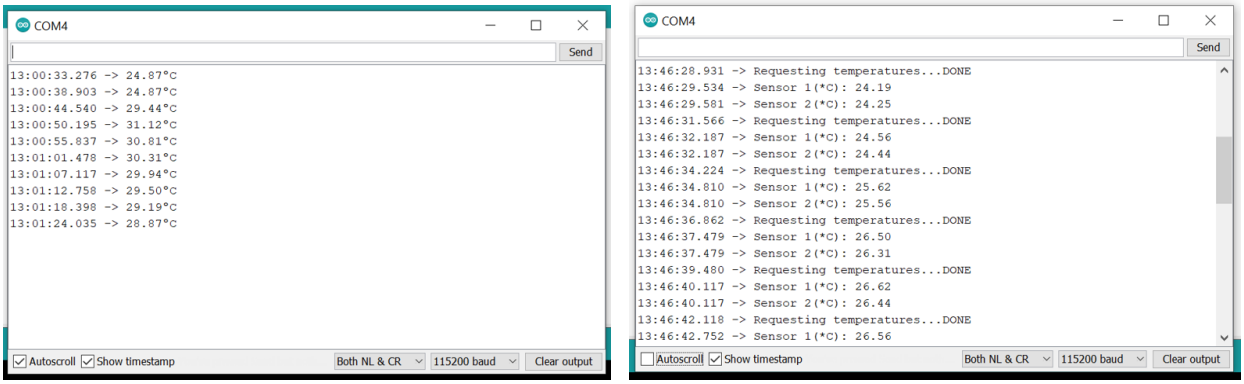

Figure 14: Outputs at the microcontroller when receiving thermocouple(s) measurements of ambient room temperature and the response to introduction of a heat signature; single thermocouple connection on the left and two thermocouples connected simultaneously on the right

3.3 Microcontroller Subsystem

The first of the microcontroller’s two requirements is emitting high signals to our darlington array, which acts as a relay driver. Specifically, depending on the configuration we are operating in, the ESP32 microcontroller’s corresponding IO pins must be able to provide a DC output within the range of 2.7 V ≤ Vout ≤ 3.3 V to each Darlington pair [15]. Figure 15 below shows our successful testing of this capability by wiring the darlington array chip and the relays to LED outputs for verification:

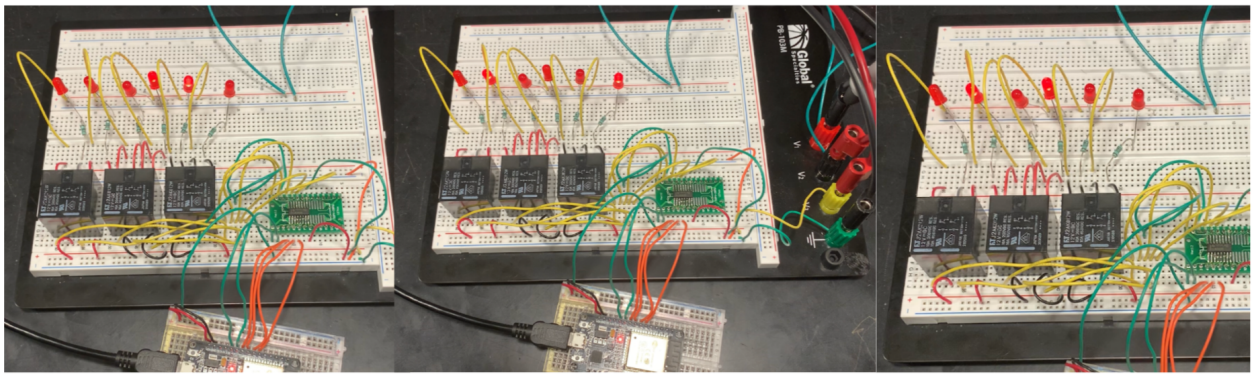

Figure 15: Proof of ability to switch between different solar cell configurations when the ESP32 outputs high control signals to 2 darlington pairs

The second microcontroller subsystem requirement is simultaneously displaying the solar parameters that are received from the thermocouples, current sensor, and voltage divider on the on-board OLED. As mentioned, full functionality of the relays was unavailable. As the project revolves around measuring the voltage and current of the distribution box input, the measurements depend on which solar inputs were connected to the load, and without full switching functionality all measurements couldn't be taken simultaneously. To mediate this issue during testing, the voltage and current inputs were manually probed at the load so the measurements of these values could be observed. Due to the shared inputs of the voltage divider and the current sensor, this measurement test was broken down in two instances because of the limited number of outputs from the Keysight DC power supply in the laboratory [16]. Figures 16 and 17 below show the successful display of the OLED on the PCB consistently displaying temperature measurements while also displaying current and voltage measurement (respectively) depending on where the DC supply was attached:

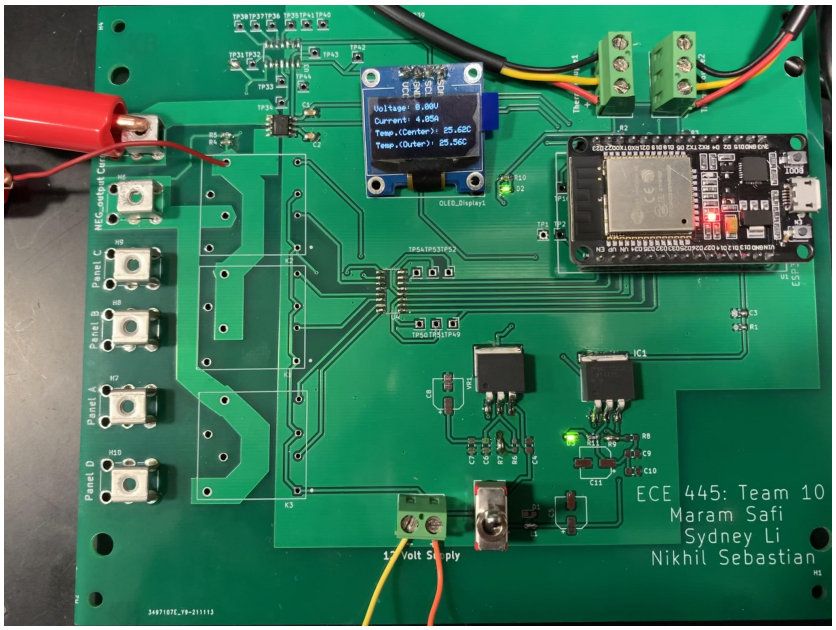

Figure 16: Current and temperature measurements displayed on the on-board OLED; the probes on the left are inputting into our current sensor

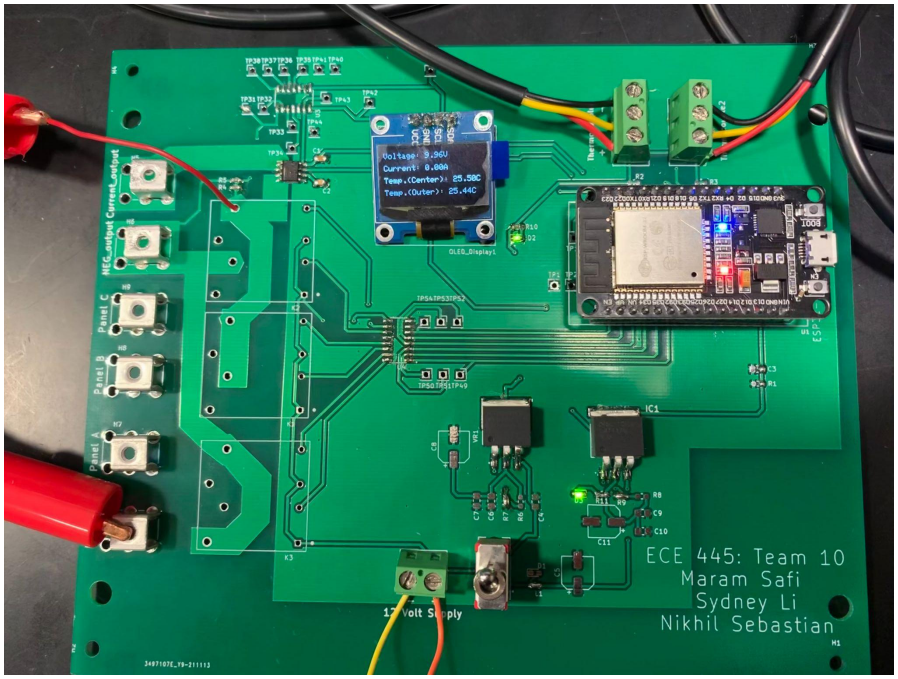

Figure 17: Voltage and temperature measurements displayed on the on-board OLED; the probes on the left are inputting into our voltage divider

3.4 Research Hub Subsystem

Looking at Table 4 in Appendix A, we can group the six key requirements for the Research Hub subsystem into 3 pairs, with the first being the requirements that ensure successful interactive user design. This encompasses both the front-end website being successfully routed between pages and displaying solar panel data based on security and observation use cases, as well as having working user authentication on the remote portal that restricts Research Hub access to required ECEB personnel only. To begin, Figures 18, 19 and 20 below show the functioning login mechanism for the website that the user faces on accessing the portal:

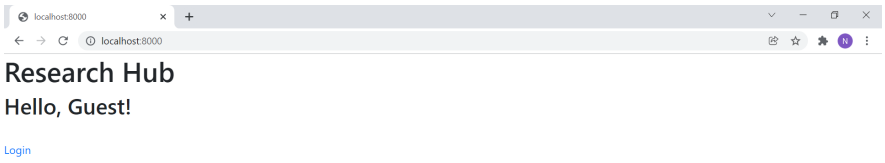

Figure 18: Landing welcome page of the portal

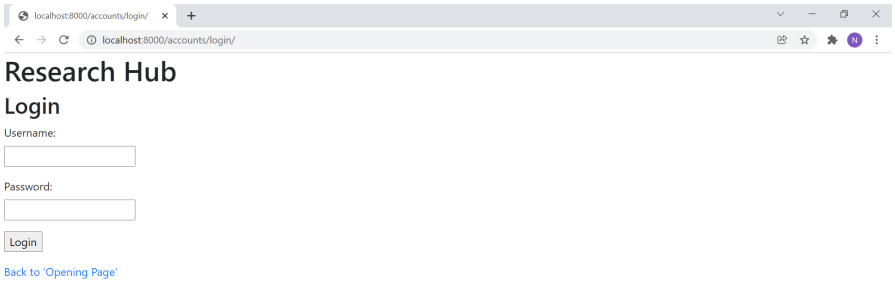

Figure 19: Login form and screen

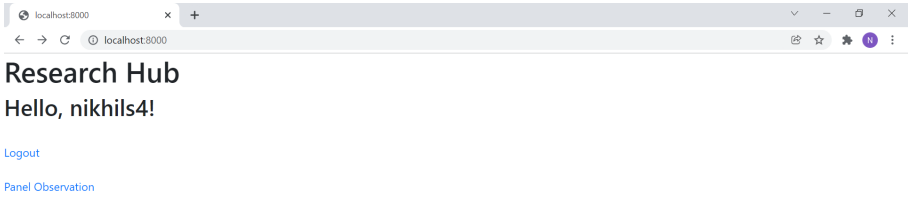

Figure 20: User dashboard with acknowledgement on successful login

Figure 21: Redirected Login page when attempting to access URLs without authentication (compare to Figure 19)

Furthermore, as seen in figure 21 above, our security measures extend past just the login as it also successfully restricts unauthorized access to panel data and more. Moving onto the navigation of the portal itself, Figures 22, 23 and 24 below show the index of monitored panels that serves as the “home” for observation, along with the dedicated pages for each monitored solar panel:

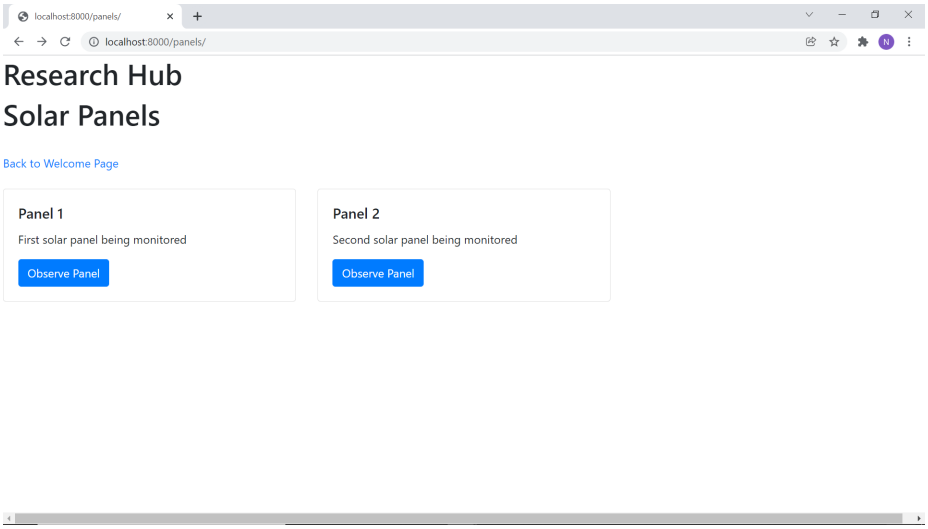

Figure 22: Panel observation home showing index of monitored panels

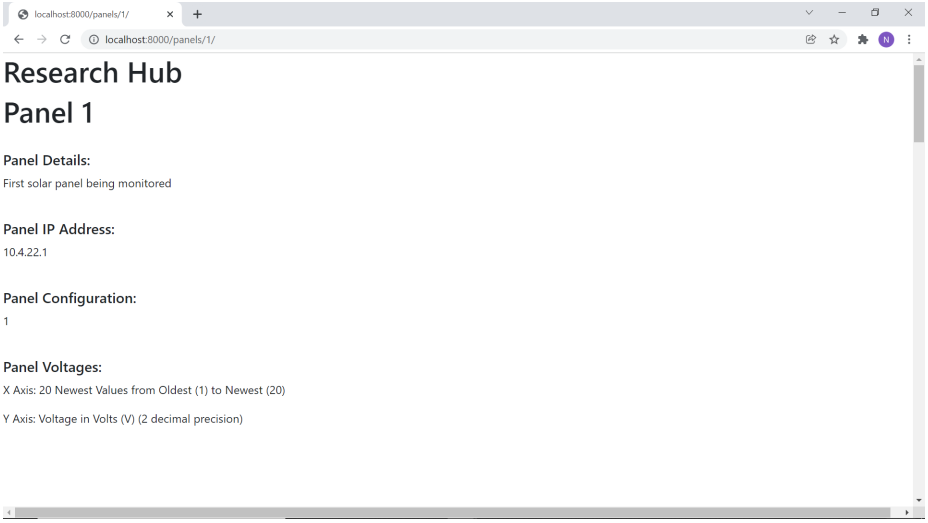

Figure 23: Panel-specific page for the first of our 2 monitored panels

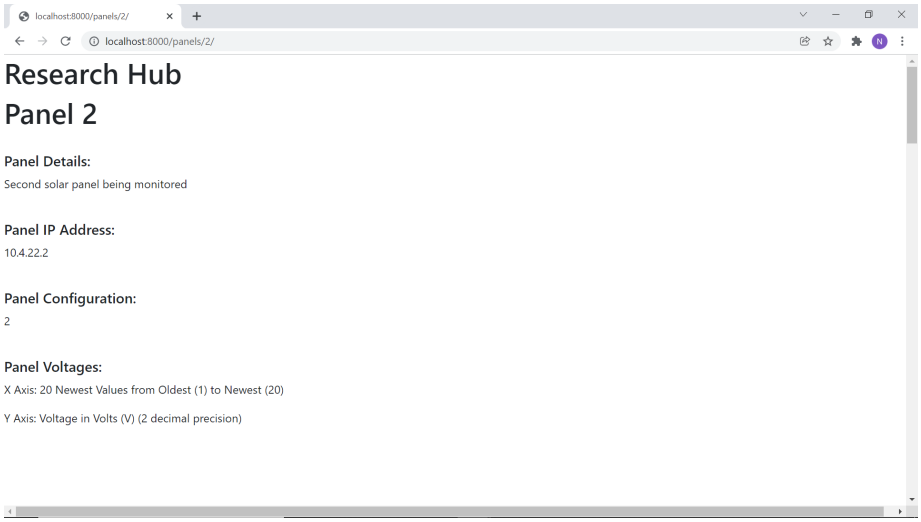

Figure 24: Panel-specific page for the second of our 2 monitored panels

Moving onto our next pair of requirements, we focus on the wireless communication capabilities of both the Django framework as well as the ESP32 microcontroller, which is central to our project as a whole. Broadly, the 2 requirements for this are that both the Django project and the ESP32 should be able to send and receive HTTP POST requests successfully. For our functional testing, we broke this up into modular testing of both frameworks' ability to send and receive POST requests independently, before integrating the forward and backward flows of software and testing the same. Figure 25 and 26 below together show the functioning ESP32 capability to send post request:

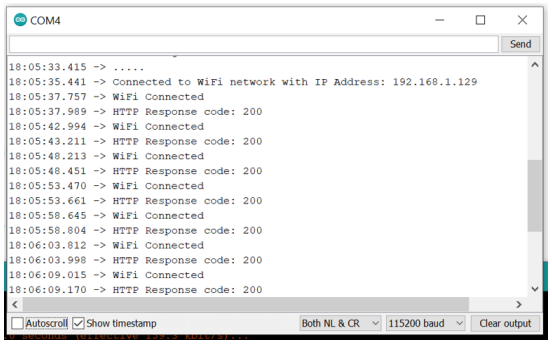

Figure 25: ESP32 and its serial monitor output showing successful sending of POST requests

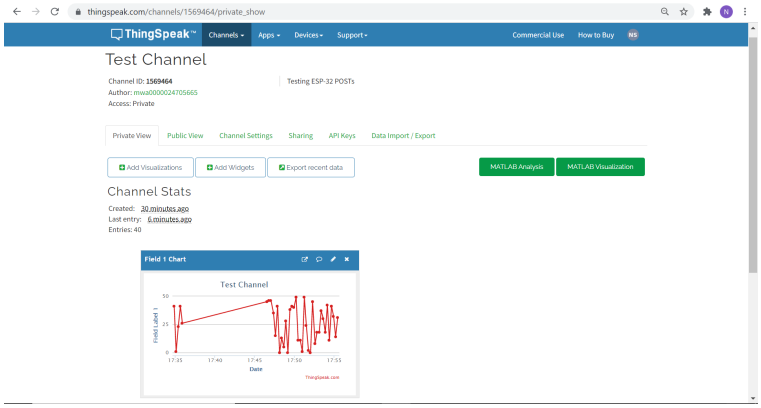

Figure 26: The ThingSpeak API visually showing reception of ESP32 POST requests through the plot

We were then able to successfully test the independent capability to receive POST requests for both the Django project and the ESP32 microcontroller. Figures 27 and 28 below respectively show the successful posting and reception of these requests from the Postman API (used for testing) to the Django back-end server and the ESP32 microcontroller:

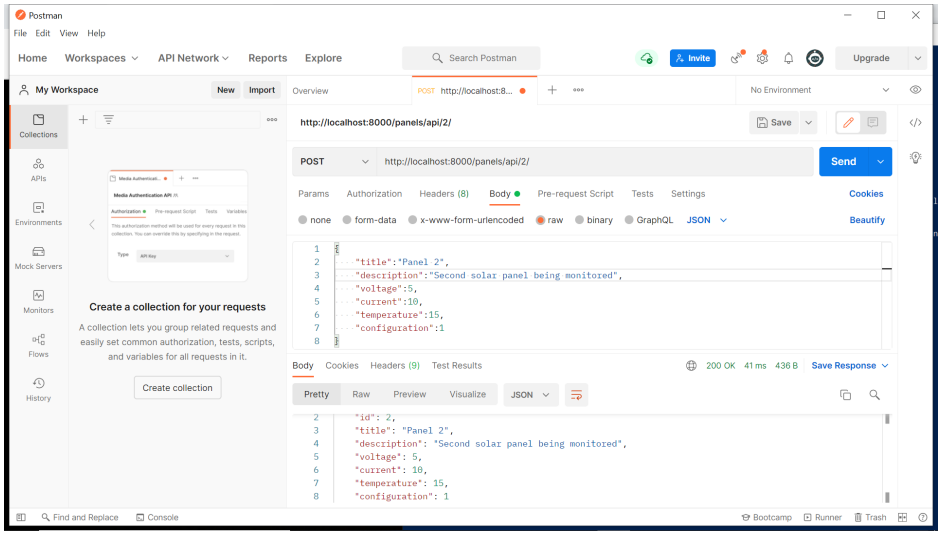

Figure 27: Postman API showing the "200 OK" success response from the Django back-end after sending a POST request

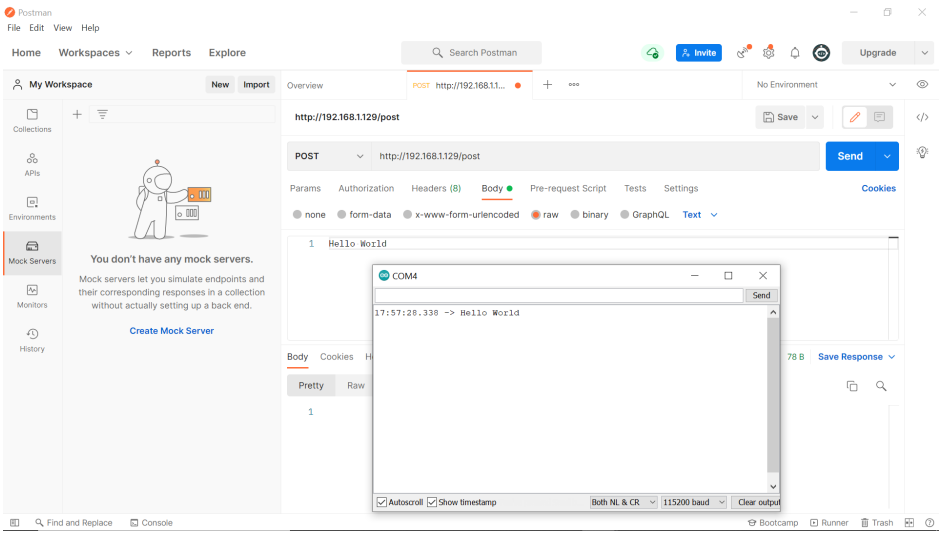

Figure 28: Serial monitor of the ESP32 displaying the received POST request from the Postman API successfully

With modular testing successful, we then integrated the functionalities to form our forward (observation data) and backward (configuration input) flows of software. The forward flow meant the ESP32 would post measurement updates to the Django back-end, which needed to successfully receive and store the same, while the backward flow meant the Django server would post a raw configuration setting that needs to be successfully received and stored at the ESP32 level. Figure 29 below shows our successful integration testing for the forward flow, while Figure 30 shows the successful integration testing for the backward flow:

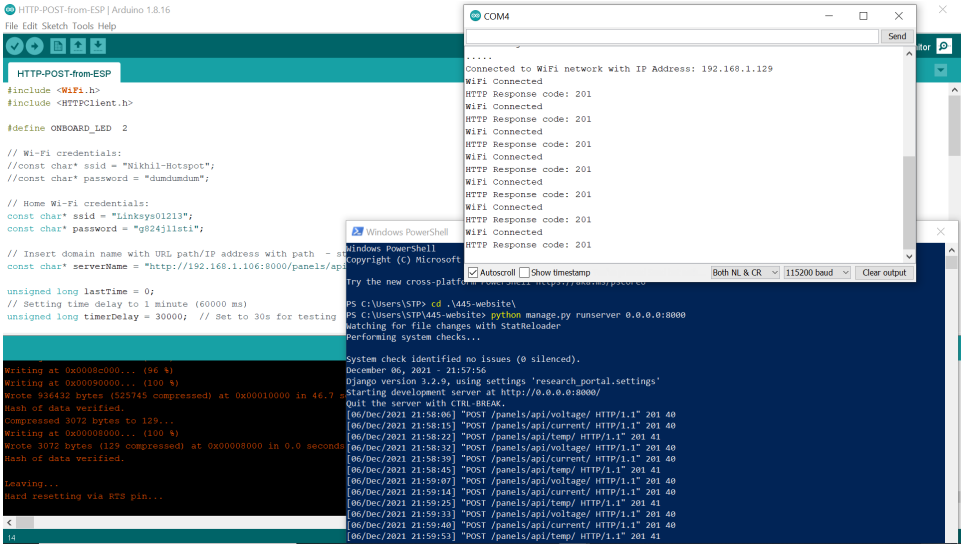

Figure 29: ESP32 serial monitor showing successful periodic POST requests being sent (for measurement updates), while the Powershell terminal shows the successful reception and handling of these requests at the Django back-end

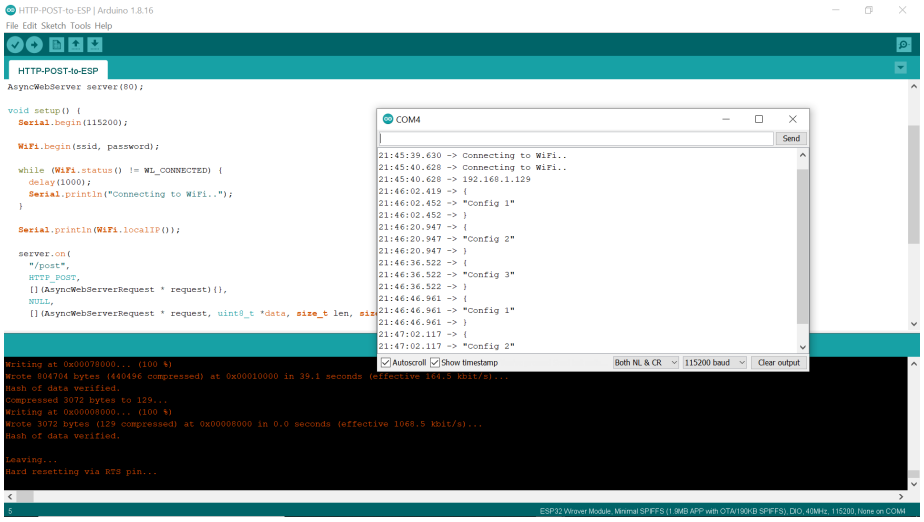

Figure 30: ESP32 serial monitor showing successful reception and storing of new configuration settings through POST requests from the Django back-end (cycled through all options for testing)

Coming to our last pair of requirements, these relate to portal updates, including those based on user input. The first of these is of course that our panel-specific pages need to update in real time with new data and reflect these changes on the panel-specific plots for the 3 solar panel parameters. Figures 31, 32 and 33 below show the voltage, current and temperature plots (and their updates) respectively for the different panels on the access portal:

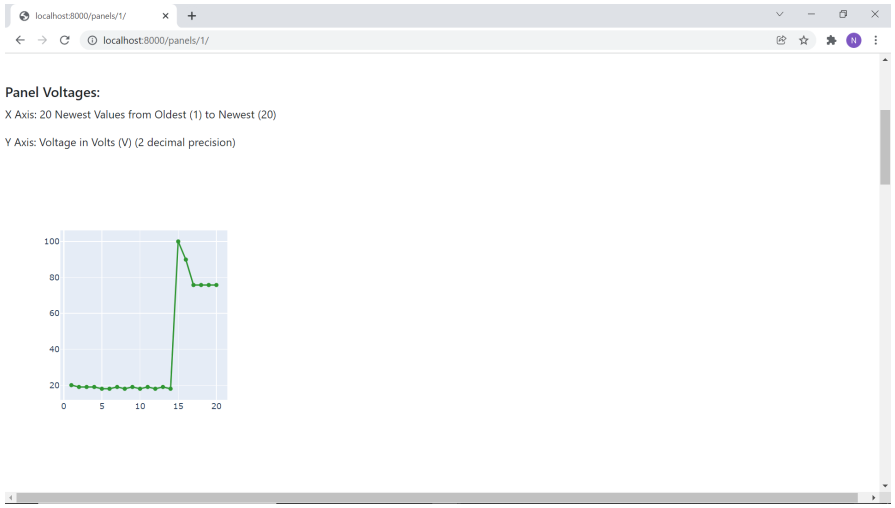

Figure 31: Data visualization of the last 20 voltage measurement updates on a panel-specific page

Figure 32: Data visualization of the last 20 current measurement updates on a panel-specific page

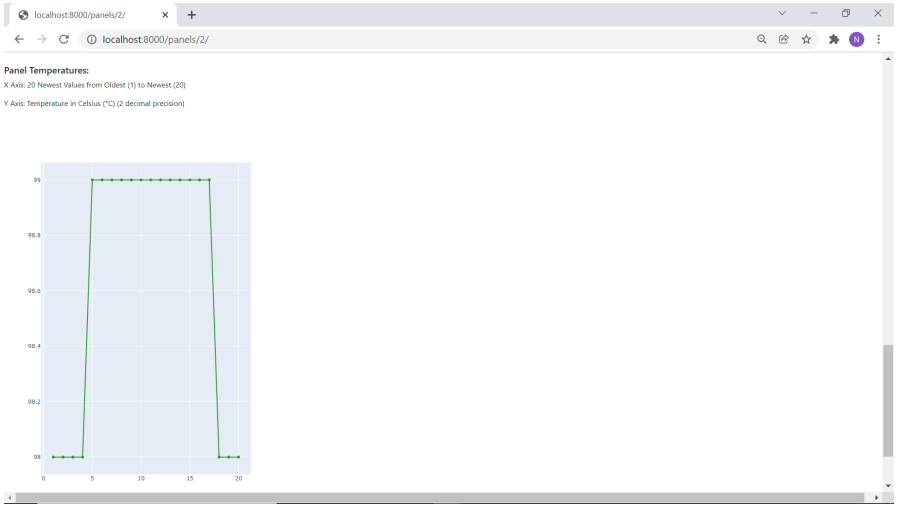

Figure 33: Data visualization of the last 20 temperature measurement updates on a panel-specific page

Finally, the second requirement in the last pair is that the user should be able to input a new configuration (relating to 32, 64 or 128 solar cells being monitored) and our Django framework should be able to handle and store this input for then posting to the ESP32. Figures 34 and 35 below respectively show the GUI for user-input configurations changes and the Django-level acknowledgment that this input was successfully handled in the back-end:

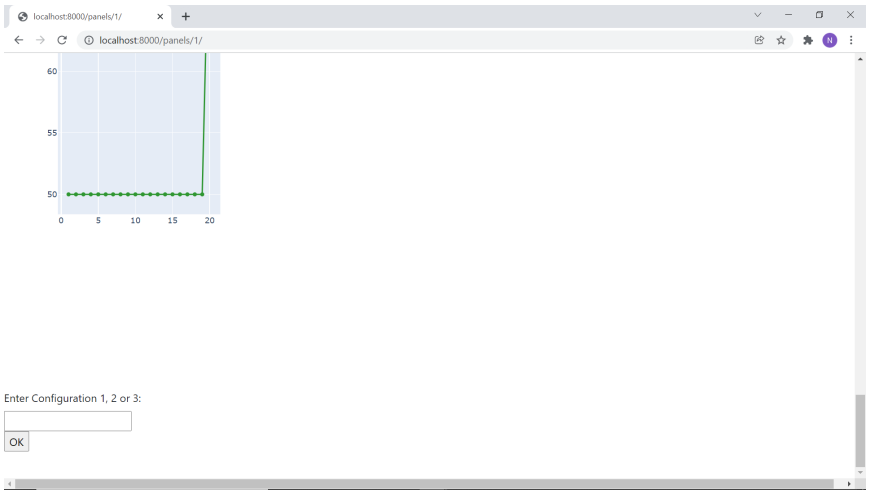

Figure 34: GUI prompting user to enter new configuration settings at the bottom of a panel-specific page

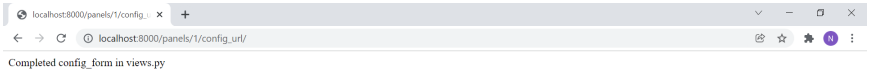

Figure 35: Confirmation displayed (forced for testing) which shows the user input was received, handled, and transitioned to the correct 'view' to trigger the subsequent POST request to the ESP32

4 Cost and Schedule

As seen in the itemized Table 1 in Appendix E, we needed a total of **$219.93** to obtain all the necessary parts for our project, i.e., to obtain the parts needed to construct two separate solar panel interface boxes.

Furthermore, this is a four credit hour class, which implies a minimum of eight hours of work a week toward our project and its goals. In alignment with our schedule planned for the semester, we had 11 weeks or 88 hours of work individually. Now, as per industry standard, UIUC Electrical Engineering undergraduates on average are paid $80,000 per annum, while UIUC Computer Engineering undergraduates are paid $100,000 per annum [17]. Allowing for some variance, we budget that an employee team member would be paid about $40 per hour as employees on this project.

So, as per the assigned class budgeting formula, our project cost would be: ($219.93) + [($40/hour) x 2.5 x 88 hours] = $9019.93 ≅ **$9100**

A final “schedule of work” explaining the week-to-week activities completed and goals achieved over the course of this semester as we worked on our project can be found in Appendix F.

5 Conclusion

Following the review of our design plans, project building, and functionality verification, this final section of the report will review what we were able to achieve, the issues we considered during the project, and what remains to be improved in the future.

5.1 Accomplishments

Overall, we believe the project was a success! We were able to get all of our subsystems functioning independently. We were able to measure everything we targeted, and read and monitor the key solar panel parameters. We could also fully integrate the functioning power, monitoring and microcontroller subsystems on the PCB, along with displaying our measurements onto the on-board OLED. The Research Hub and software functionality also worked as expected and the server and webpage run in a clean and organized manner. Additionally, we were able to maintain good documentation of all of our work through GitHub – which was a specification emphasized by Professor Banerjee from the previous attempt at this project – so we are confident that our work can be continued as well.

5.2 Uncertainties

Due to some of the challenges faced we did end up with some uncertainties in our final solution. The first of these is the question of isolation due to our removal of the ADC in the final voltage measurement layout. While our voltage measurement still works, its direct connection to the ESP32 poses a risk to the microcontroller in case of power surges. But, since we have already set up the hardware and software layouts for this, the ADC can easily be added to our project in the future once it can be shown to successfully communicate with the ESP32 in a stand-alone manner.

Additionally, while logically functional (as proven by testing), we could not provide an ideal final implementation of our darlington array chip and the relay switching circuit at the hardware level due to the fact that mistaken wiring and soldering used up a lot of our time and fried through components. But, once again, since we have been able to complete the software side of this and have already got a plan that would work on hardware implementation, we are sure this is something that can be easily fixed in the future.

Finally, although working when tethered with a micro-USB cable, we were not able to get our ESP32 functioning with a WiFi network when powered independently, hence preventing the total integration of our project. Some reasons for this could be cheaply-made/faulty WiFi modules on our ESP32 that don't have the capacity to handle heavy wireless capabilities when independently powered, or possible mistakes on our PCB wiring that may be setting important board pins to wrong values. A better examination of this problem, and trying a similar approach with different boards, would help solve this issue as well.

5.3 Ethical Considerations

- Project Issues:
    - Solar energy should not contribute to net greenhouse gas emissions and exacerbate global climate change, and our peripheral device additions should not affect this either

- General Solar Panel Issues:
    - Costs and benefits of solar energy should be distributed in an equitable way by regulatory agencies
    - Solar Energy should be environmentally sustainable – the recycling of solar panels at the moment is a huge impediment to their overall sustainability [18]
    - The manufacture of solar panels has also seen negative outcry due to problems like forced labor camps for polysilicon [19]

5.4 Safety and Regulatory Standards

Given below are standards and procedures we implemented when designing and building our project. We adhered to these guidelines carefully in order to keep ourselves and others safe, while maintaining the integrity of our work:

1. IEEE Ethics Code #1: Hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment' [20]
2. IEEE Ethics Code #7: To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others' [20]
3. IEEE 1547-2018: IEEE Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power Systems Interfaces [21]
4. NREL/TP-550-38603 October 2005: Procedure for Measuring and Reporting the Performance of Photovoltaic Systems in Buildings [22]

5.5 Future Work

A possible future improvement would be to update our PCB with correct internal connections, in which we would properly map the control signals to our darlington array and eliminate the need of soldering wires across different test points of the board. Another useful update would be to achieve isolation utilizing an ADC that could be recognized and acknowledged with our ESP32 microcontroller. From a software point of view, establishing a reliable remote ESP32 WiFi connection so that wireless communication is functional even on an independent power supply would be incredibly beneficial. On a similar note, visually enhancing the Research Hub interface to make it more appealing would be a simple yet valuable improvement.

References

[1] D. Vadgama, D. J. Lee, and S. Reddy, “Smart Interface Box for Solar Panels Final Report,” *ECE 445* , 12-Dec-2019. [Online]. Available: https://courses.engr.illinois.edu/ece445/getfile.asp?id=16307.

[2] SunPower, “What is Solar Energy and How Do Solar Panels Work?,” *SunPower*, 24-Mar-2021. [Online]. Available: https://us.sunpower.com/solar-array-definition. [Accessed: 08-Dec-2021].

[3] “FTR-J2AK012W Fujitsu: Mouser,” *Mouser Electronics*. [Online]. Available: https://www.mouser.com/ProductDetail/Fujitsu/FTR-J2AK012W?qs=EvTR%252B%2FkpER9cjG7hKs6Mbg%3D%3D. [Accessed Sept 25. 2021]

[4] “ACS714 Hall Effect Current Sensor,” *DigiKey*. [Online]. Available: https://www.digikey.com/en/product-highlight/a/allegro-microsystems/acs714-automotive-grade-hall-effect-current-sensor. [Accessed Sept 26. 2021]

[5] “DFR0198 datasheet by DFRobot,” *Digi*. [Online]. Available: https://www.digikey.com/htmldatasheets/production/2423516/0/0/1/dfr0198.html?utm_adgroup=xGeneral&utm_source=google&utm_medium=cpc&utm_campaign=Dynamic+Search_EN_Product&utm_term=&utm_content=xGeneral&gclid=CjwKCAjwz5iMBhAEEiwAMEAwGBQIl0BBa8TLN6clv0pc3m26AJrk8NiFfkHFsCgklkJpG2OzAhdIURoCcZ8QAvD_BwE. [Accessed Sept 21. 2021]

[6] “LM1117 800-mA, Low-Dropout Linear Regulator,” *TI*. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm1117.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1634572215589. [Accessed Sept 25. 2021]

[7] “ULN2003B High-Voltage, High-Current Darlington Transistor Array.” [Online]. Available: https://www.ti.com/lit/ds/symlink/am6442.pdf?ts=1623989396311&ref_url=https%253A%252F%252Fwww.google.com%252F. [Accessed: 08-Dec-2021].

[8] “16-Bit, Multi-Channel Analog-to-Digital Converter with I2C Interface and On-Board Reference,” *MCP3426/7/8*. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/22226a.pdf. [Accessed: 08-Dec-2021].

[9] “Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor,” *ACS712*. [Online]. Available: https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf. [Accessed: 08-Dec-2021].

[10] “128 x 64 Graphic OLED.” [Online]. Available: https://www.vishay.com/docs/37902/oled128o064dbpp3n00000.pdf. [Accessed: 08-Dec-2021].

[11] S. M. Derek Mitchell, Sunpower SPR-425E-WHT-D (425W) Solar Panel. [Online]. Available: http://www.solardesigntool.com/components/module-panel-solar/Sunpower/3875/SPR-425E-WHT-D/specification-data-sheet.html. [Accessed: 08-Dec-2021].

[12] “Heartland 8 x 6 x 4 inches AH864C enclosure - Attabox: Non-metallic ENCLOSURES - polycarbonate and fiberglass,” *Attabox*, 15-Jun-2021. [Online]. Available: https://attabox.com/product/heartland-8-6-4-ah864c-enclosure/. [Accessed: 27-Sep-2021].

[13] A. Mittal, “Losses in PCB Transmission Lines,” *Sierra Circuits*, 15-Jun-2021. [Online]. Available: https://www.protoexpress.com/blog/losses-in-pcb-transmission-lines/. [Accessed: 08-Dec-2021].

[14] “Analog Read Accuracy - ESP32 Forum,” *Espressif ESP32*. [Online]. Available: https://esp32.com/viewtopic.php?t=13089. [Accessed Sept 25. 2021]

[15] Espressif, “ADC - Issue #1804 - Espressif Arduino ESP32,” *GitHub*. [Online]. Available: https://github.com/espressif/arduino-esp32/issues/1804. [Accessed Sept 21. 2021]]

[16] “Keysight E363xA Series Programmable DC Power Supplies.” [Online]. Available: https://www.keysight.com/us/en/assets/7018-06785/data-sheets/5968-9726.pdf. [Accessed: 08-Dec-2021].

[17] Grainger Engineering Office of Marketing and Communications, “Salary averages,” *Electrical & Computer Engineering | UIUC*. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages. [Accessed: 25-Sep-2021].

[18] V. Taylor, “The Ethics of Solar Power Are More Complicated Than You Think,” *Mic*, 07-May-2021. [Online]. Available: https://www.mic.com/p/the-ethics-of-solar-power-are-more-complicated-than-you-think-75484220. [Accessed: 15-Sep-2021].

[19] J. Gudmundsen, 2020 Jason Hayes | March 25, and 2020 Jack McHugh | December 18, “There Are Ethical Concerns Surrounding Solar Energy,” *Mackinac Center for Public Policy*, 12-Jul-2021. [Online]. Available: https://www.mackinac.org/the-ethical-concerns-surrounding-solar-energy. [Accessed: 15-Sep-2021].

[20] “IEEE Code of Ethics.” Institute of Electrical and Electronics Engineers. 2016. https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed Sept 19. 2021]

[21] “IEEE 1547-2018 - IEEE Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power SYSTEMS INTERFACES.” *IEEE SA - The IEEE Standards Association - Home*, standards.ieee.org/standard/1547-2018.html.

[22] Pless, S., et al. “Procedure for Measuring and Reporting the Performance of Photovoltaic Systems in Buildings.” *UNT Digital Library*, National Renewable Energy Laboratory (U.S.), 3 Dec. 2015, digital.library.unt.edu/ark:/67531/metadc785614/.

Appendix A: Requirements and Verifications Tables

Given below are the four sets of requirements and verifications for the four key subsystems of the project, which is the basis of Section 3 (Design Verification):

Table 1: R&V Table for Power Subsystem

Requirements	Verifications
The 12-volt power rail provides a 12-volt DC input within the range of ± 1% [13]	1. Utilize a DC-power supply and input 12-volts into our screw terminal via jumper wires
Converts a 12-volt input into a 3.3-volt DC output within the range of 3.235V ≤ Vout ≤ 3.365V [6]	1. Probe the output of the linear regulator LM1117-3.3 with a multimeter and see if a stable 3.3-volt output is available
Converts a 12-volt input into a 5-volt DC output within the range of 4.90V ≤ Vout ≤ 5.10V [6]	1. Probe the output of the linear regulator LM1117-5.0 with a multimeter and see if a stable 5.0-volt output is available

Table 2: R&V Table for Monitoring Subsystem

Requirements	Verifications
The relay subsystem must be able to configure and choose between the different solar cell configurations: 32-cells (CD), 64-cells (BC), and 128-cells (AD). The relay output must be the same as the solar panel output minus the contact voltage and current drop [3] - Initial Voltage Contact Drop: ± 0.1V - Initial Current Contact Drop: ± 100mA	1. Connect a 15-volt DC power supply between CD screw terminals 2. Between the Current_Output and NEG_Output screw terminals, measure the voltage with a multimeter and see whether we observe the 15-volt input when configured to CD 3. Repeat A1 and A2 with screw terminals BC and AD
The voltage divider must be able to step down input voltages within the ranges of 0V - 85.6V and corresponding output voltages within the range of 0V - 3.3V [15]	1. Apply a DC power supply to the Current_Output and NEG_Output screw terminal* 2. Utilize a multimeter to probe VOLT_Output to observe whether there is a voltage within a 0V - 3.3V range [8] 3. Vary the range of the DC power supply to ensure it works for the 0V - 85.6V range
The current sensor must accurately read the current of the relay output within an error range of 1.5% [4]	1. Connect a DC power supply between CD screw terminals and make the input current 3A

	2. Measure the current between Current_Output and NEG_Output 3. Measure the output of the ACS714 current sensor and observe whether the output is within 1.5% range of the current measured in step 2
Thermocouples should measure values between the range -10°C and 85°C within an accuracy of ± 0.5°C [5]	1. Utilize a thermal gun to measure the ambient room temperature 2. Observe whether the thermocouples are within our tolerance range of that room temperature 3. Add a heat signature and see if the thermocouple response and increases in temperature
The ADC Converter should output a 16 bit digital code to our ESP32 that corresponds to the analog input into the IC once decoded [8].	1. Connect to a DC voltage supply of 5V to an input pin and ground its corresponding pin. 2. Capture 16 bit digital code sent by ADC to ESP32 using serial monitor. 3. Decode what was written to the ESP32 by multiplying the output code with the LSB and dividing by the PGA setting. 4. Verify that it is the respective voltage that was inputted into the ADC.

* Note for Table 3: Keithley DC Power Supplies provided in the lab have a max voltage range of 25V [16]. For safety, the voltage divider is simple enough to verify the output voltage range in simulation but for safety, we will not input a max 85.6-volt supply for safety reasons and because there is no voltage supply available for 85.6-volts

Table 3: R&V Table for Microcontroller Subsystem

Requirements	Verifications
Depending on which configuration we are operating the solar panels on, the ESP32 microcontroller's corresponding IO pins must be able to provide a DC output within the range of 2.7V ≤ Vout ≤ 3.3V to each Darlington pair [15] The following IO pins correspond to different configurations:	1. Choose CD configuration through the user interface 2. Use a multimeter to measure the output of the ADC2 pins corresponding to PANEL_C with Current_Output and PANEL_D with NEG_Output and observe whether it is 3.3V. The rest of the ADC2 pins should be 0V 3. Change the configurations to BC and AD and

	repeat A1 and A2 with the according changes (i.e.: BC configuration measures output pins corresponding to PANEL_B with Current_Output and PANEL_C with NEG_Output)
The ESP32 can communicate the current solar panel parameters to the OLED display 17]	1. Connect a Yokogawa power meter across the Current_Output and the NEG_Output probes and measure their DC voltage 2. Connect the Yokogawa power meter probes across POS_Output and Current_Output to measure the current 3. Use a temperature gun to measure the solar panel temperature at the locations where the thermocouples are attached (if digital temperature can be retrieved and displayed from the web interface, then the option to compare the OLED temperature readings and the temperature displayed on the web interface is viable) 4. Observe the OLED display and see whether the voltage is displayed ± 300mV [9], current within a 1.5% error range inline with the readings we retrieve from the power meter. 5. Observe the OLED display and see whether the temperature is within the range of ± 0.5% accuracy we retrieve from the thermal imaging camera [11] * The Yokogawa power meter has the option of displaying the DC voltage and DC current

Table 4: R&V Table for Research Hub Subsystem

Requirements	**Verifications**
Front-end website should successfully route between pages and display solar panel data based on security and observation use cases	1. Landing page on accessing website with base URL (http://localhost:8000/) should be the login page 2. Successful login should be the only way to access panel data - direct URL modifications should reroute to login page 3. Solar panel data is accessed through an index (should be easy to add additional panels for scalability) 4. Website routing should go from login page to authentication to solar panel index which also has options for logout and team/project info
Working user authentication on the remote portal that restricts Research Hub access to	1. Add a user other than the Django superuser to the “users” Django application (through Django administration)

required ECEB personnel only *(Note: new users will only be added through administrative editing to allow for further security)*	2. Access the website and try to login through the welcome/initial landing page when live 3. Website should welcome the user explicitly by username and route them to solar panel data if successful
The back-end of the Django framework should have HTTP communication ability (2-way POST requests)	1. Base test: a. Send a POST request to the Postman API (hardcoded) from the Django back-end and it should be received and readable on the API b. Send a POST request from the Postman API to the Django back-end IP and it should be received and readable on the web server 2. Integration test: a. Send a POST request to the ESP-32 server IP (hardcoded) from the Django back-end and it should be received and readable at the board level b. Send a POST request from the ESP-32 server (hardcoded) to the Django back-end IP and it should be received and readable on the web server
The ESP-32 Microcontroller should have HTTP communication ability (2-way POST requests)	1. Base test: a. Send a POST request to the Postman API (hardcoded) from the ESP-32 server and it should be received and readable b. Send a POST request from the Postman API to the ESP-32 server IP and it should be received and readable at the board level 2. Integration test: a. Send a POST request from the ESP-32 server (hardcoded) to the Django back-end IP and it should be received and readable on the web server b. Send a POST request to the ESP-32 server IP (hardcoded) from the Django back-end and it should be received and readable at the board level
	1. Upload stream of arbitrary data points to the Django back-end for a specific panel by its

Real-time observed data that is received should be processed correctly to allow for plot/visualization updates for the specific panel	model instance (can be randomly generated) 2. Dedicated panel webpage should show update in data visualization/plotted charts in a time series manner 3. Integration: follow same approach but by triggering the data update through a Postman API POST request to the Django framework with information
User-input configuration settings should be panel-specific and submission should trigger back-end communication	1. User-input through GUI on panel-specific webpage for configuration settings should update Django back-end database for the specific panel a. It should also update visual cues on the webpage for that specific panel 2. Any back-end database change for a panel should trigger an outgoing POST request to the corresponding microcontroller

Appendix B: Circuit Schematics

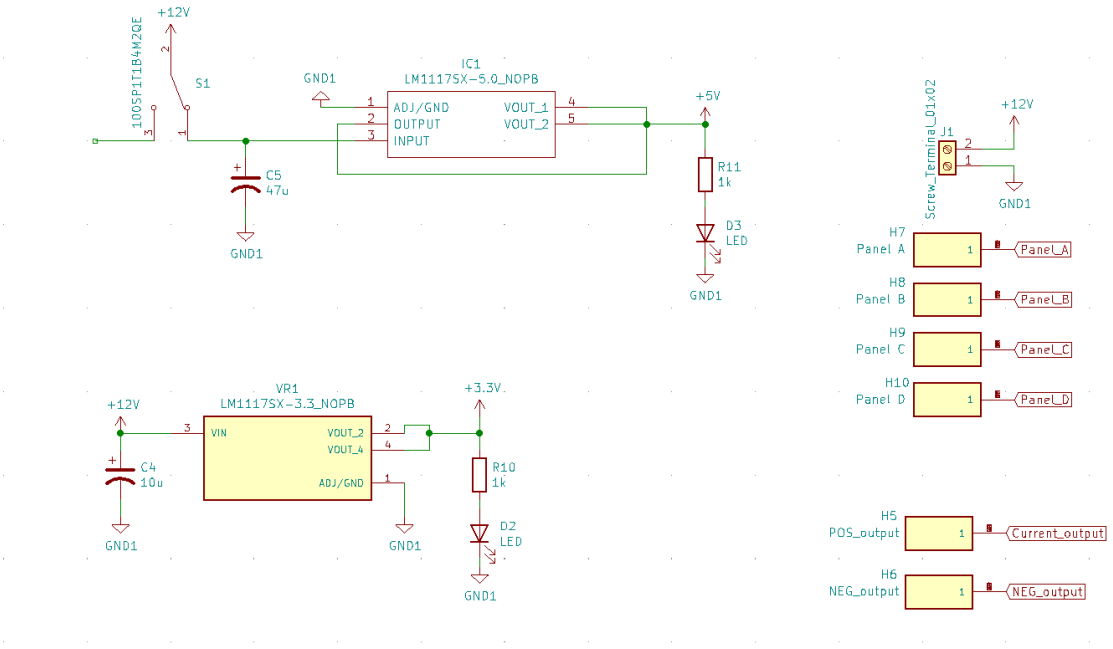

Figure 1: Linear Regulator utilized to step-down the 12-Volt to separate 3.3-Volt and 5-Volt lines

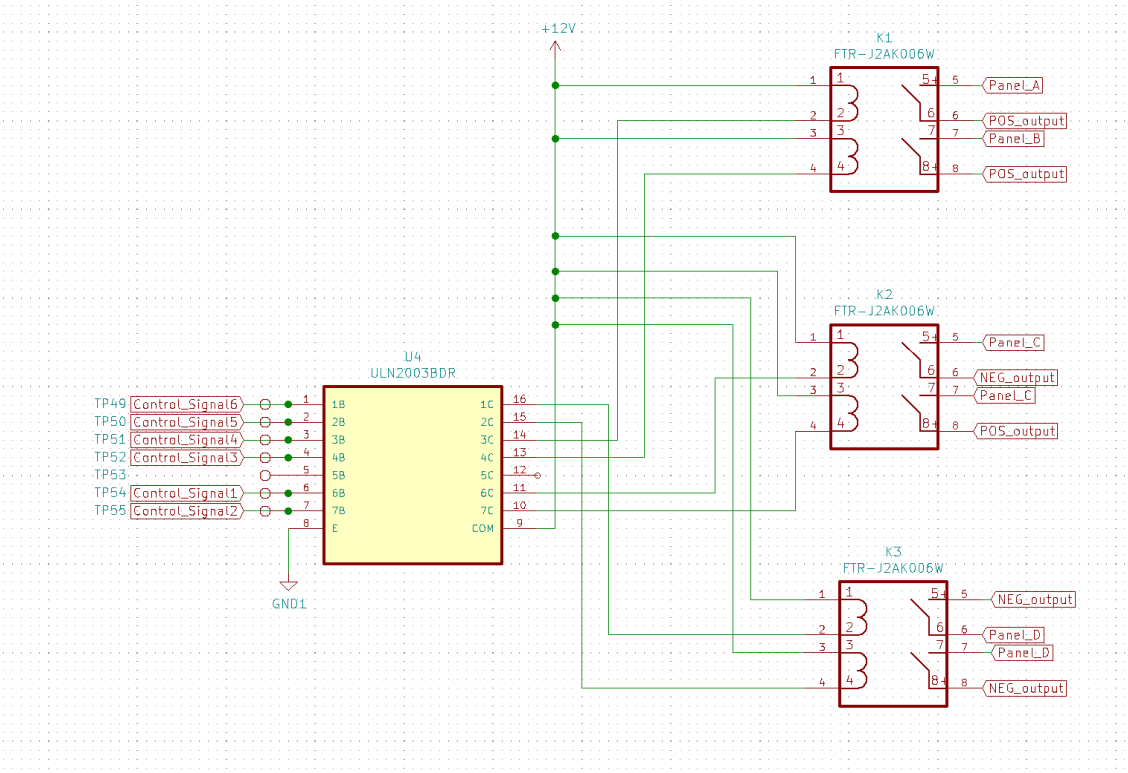

Figure 2: Relay system to configure which solar cells are being monitored

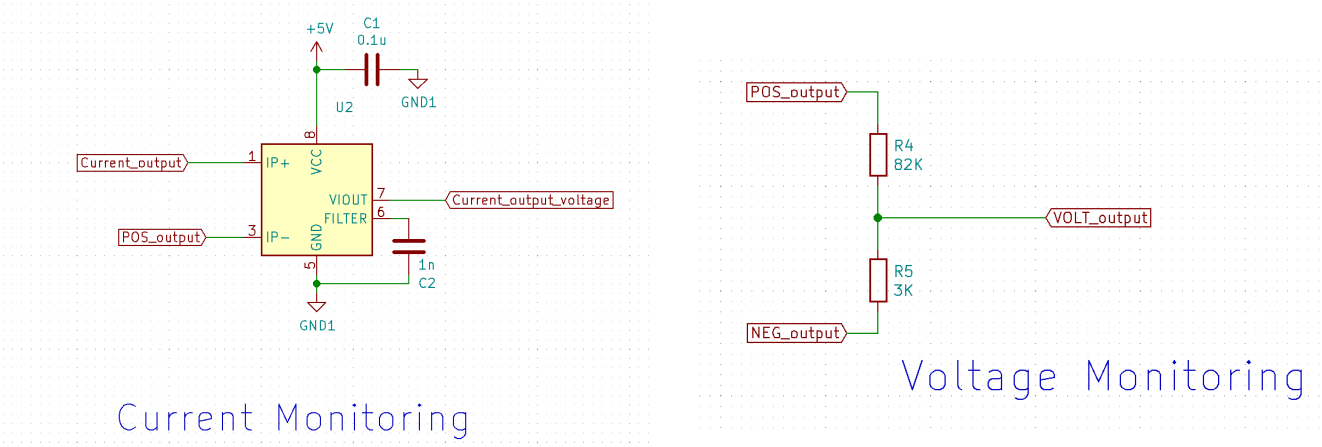

Figure 3: Voltage Divider stepping-down the voltage for voltage monitoring, and an ACS714 current sensor for current monitoring

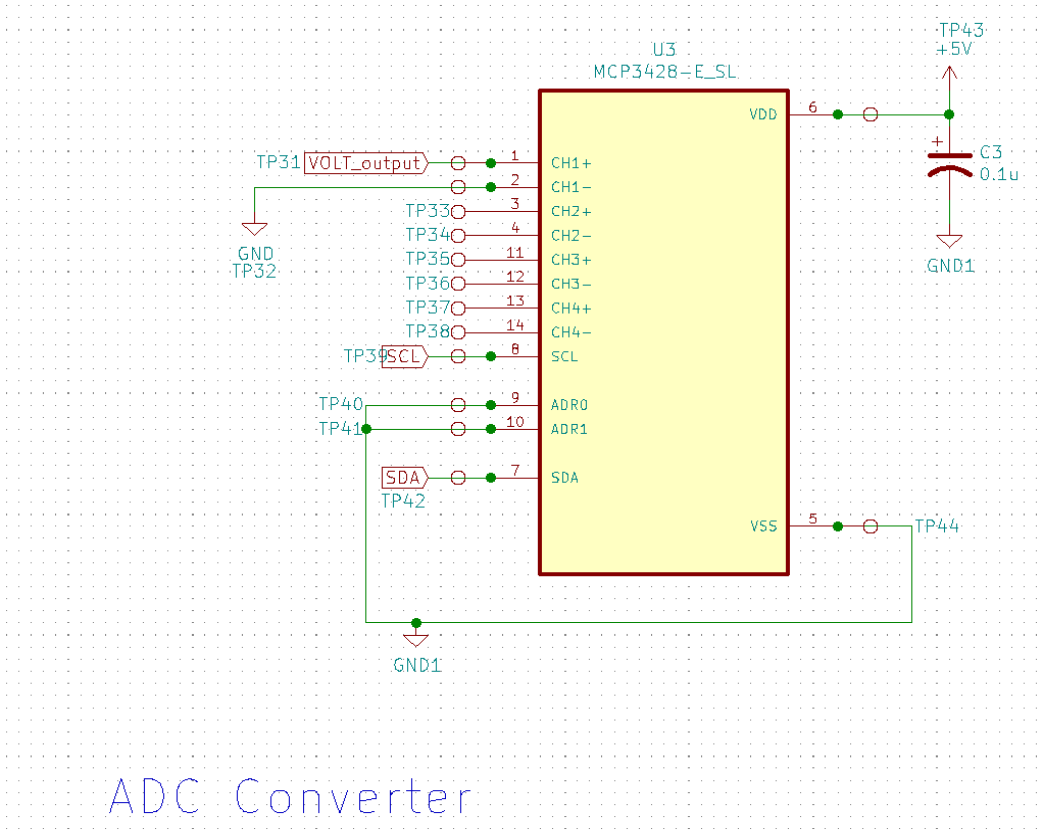

Figure 4: ADC Converter utilized to provide galvanic isolation between the solar panels and the microcontroller

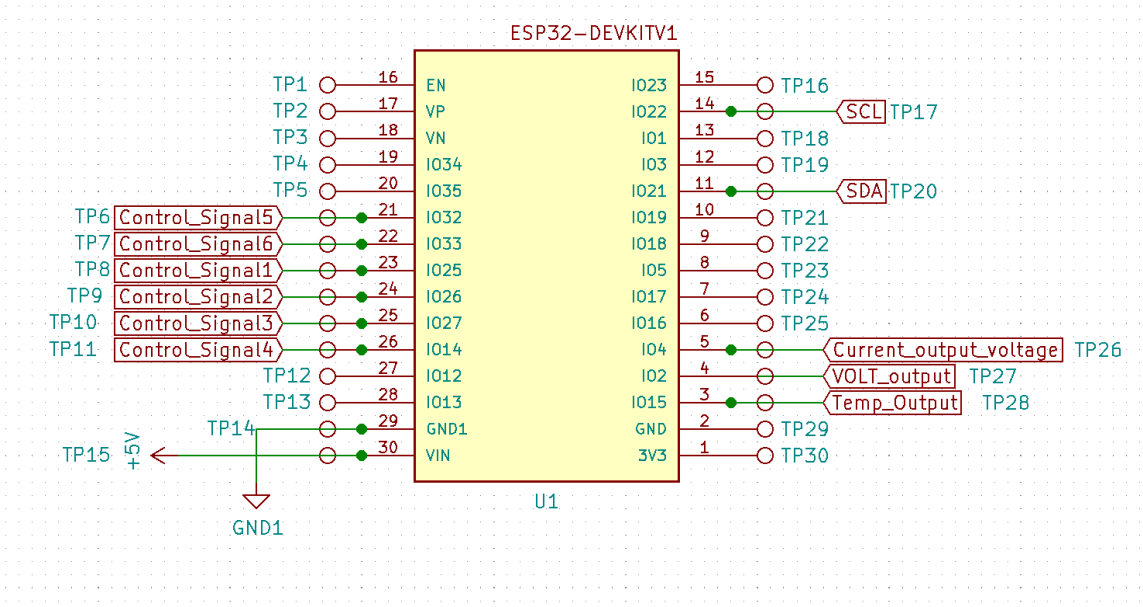

Figure 5: Pin configuration of the ESP32 Microcontroller that communicates user inputs and dictates solar panel measurements

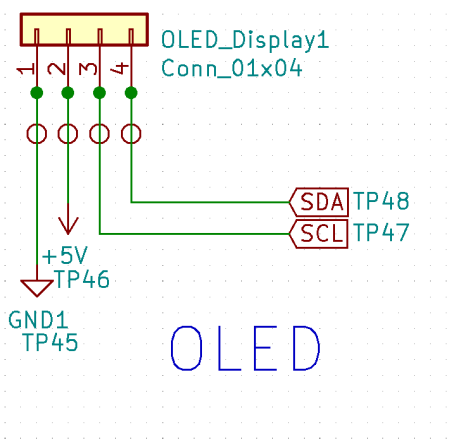

Figure 6: OLED Display to see voltage, current, and temperature readings of the solar panels

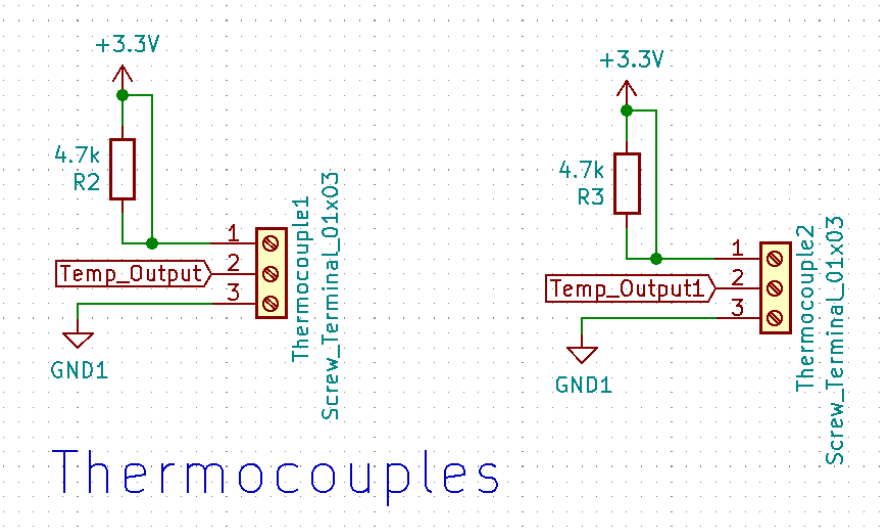

Figure 7: Thermocouples utilized to sense solar panel temperature

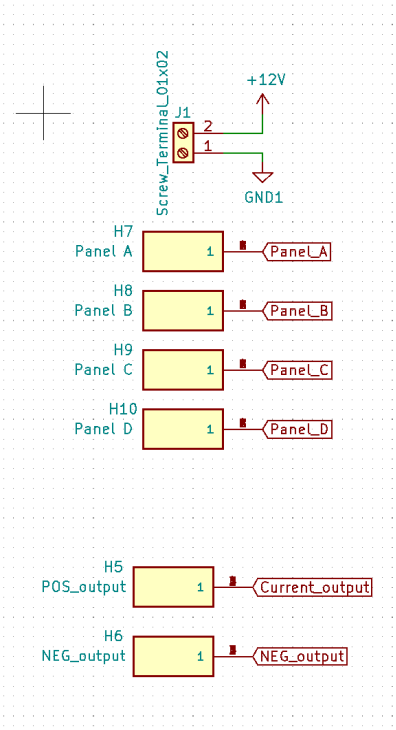

Figure 8: 12-Volt supply, External connections to solar panels; Output of our PCB

Appendix C: Calibration Results

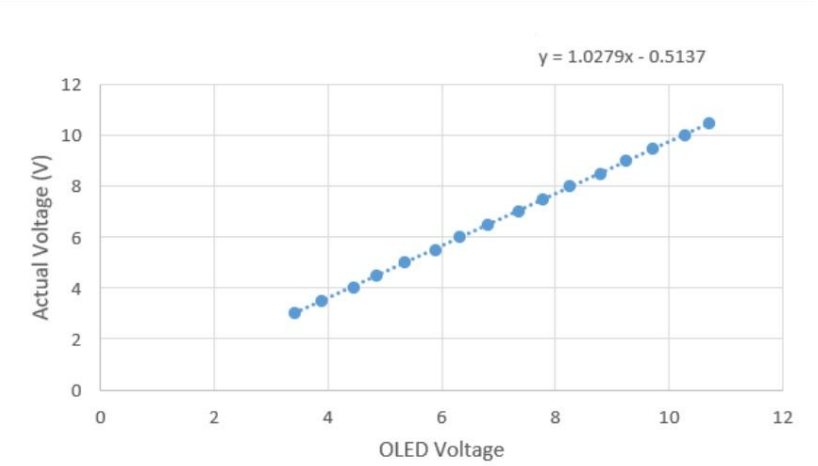

Figure 1: Plot showing the calibration of reverse-engineered voltage values from voltage divider circuit along with achieved equation

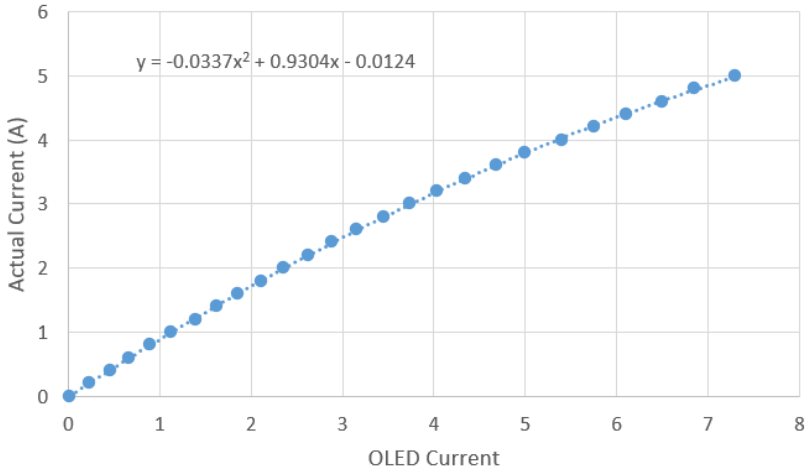

Figure 2: Plot showing the calibration of the ACS712 Current Sensor values along with the achieved equation

Appendix D: Software Flowcharts

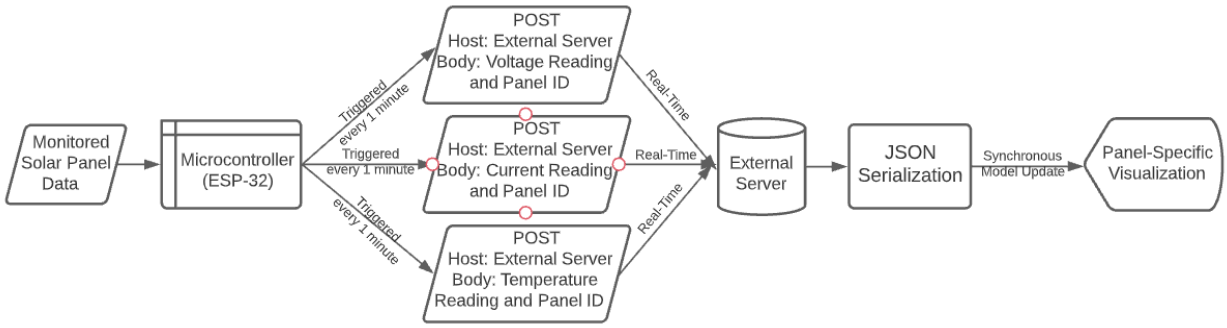

Figure 1: Forward flow of software – from observations at microcontroller to periodic updates on the portal

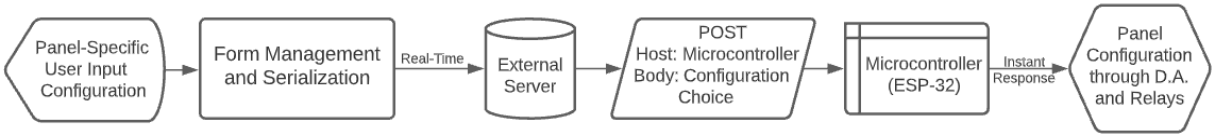

Figure 2: Backward flow of software - from user input configuration change to physical update at the interface box

Appendix E: Parts Costs

Table 1: Parts to be bought for use (0 build hours)

Part Name	Vendor	Vendor Part #	Manufacturer Part #	Quantity	Cost x # Units = Total Cost
EVAL BOARD FOR ESP-WROOM-32	Amazon	1965-1000-ND	ESP32-DEVKITC-32D	2	$7.30 x 2 = $14.60
SENSOR CURRENT HALL 30A AC/DC	Newark	SEN-13679 ROHS	ACS714ELCTR-30A-T	2	$7.08 x 2 = $14.16
Dorhea 6PCS 0.96" OLED Display Module 12864 128x64 Pixel	Amazon	15758-1	7.02E+11	2	$3.33 x 2 = $6.66
WATERPROOF DS18B20 DIGITAL TEMPE	DigiKey	1738-1311-ND	DFR0198	4	$6.90 x 4 = $27.60
DIODE SCHOTTKY 40V 3A SMA	DigiKey	B340AE-13DITR-ND	B340AE-13	6	$0.41 x 6 = $2.46
DIODE RECT 100V 20A DPAK	DigiKey	497-16946-2-ND	FERD20H100SB-TR	15	$0.97 x 15 = $14.55
IC INVERTER 6CH 6-INP 14TSSOP	DigiKey	296-26491-5-ND	SN74HCT04PW	3	$0.49 x 3 = $1.47
IC REG LIN 3.3V 800MA DDPAK	DigiKey	LM1117SX-3.3/NOPB TR-ND	LM1117SX-3.3/NOPB	4	$1.93 x 4 = $7.72
IC REG LIN 5V 800MA DDPAK/TO263	DigiKey	LM1117SX-5.0/NOPB TR-ND	LM1117SX-5.0	4	$1.41 x 4 = $5.64
CAP ALUM 47UF 20% 25V SMD	DigiKey	493-3935-2-ND	UCL1E470MCL1GS	6	$0.57 x 6 = $3.42
CAP CER 10UF 25V X5R 0805	DigiKey	445-5985-2-ND	C2012X5R1E106M125AB	6	$0.35 x 6 = $2.10
CAP CER 0.1UF 50V X7R 0805	DigiKey	399-17644-2-ND	C0805J104K5RACAUTO	5+5	$0.53 x 10 = $5.30
CAP CER 1000PF 25V X7R 0805	DigiKey	399-15433-2-ND	C0805C102K3RAC7800	2	$0.19 x 2 = $0.38
RES SMD 240 OHM 1% 1/10W 0603	DigiKey	YAG3582CT-ND	AC0603FR-07240RL	4	$0.10 x 4 = $0.40
RES SMD 1K OHM 1% 1/10W 0603	DigiKey	13-AC0603FR-131KLTR-ND	AC0603FR-131KL	6+4+4	$0.10 x 14 = $1.40
RES SMD 390 OHM 1% 1/10W 0603	DigiKey	311-390HRCT-ND	RC0603FR-07390RL	2	$0.10 x 2 = $0.20
RES SMD 4.7K OHM 1% 1/10W 0603	DigiKey	YAG3613CT-ND	AC0603FR-074K7L	2	$0.10 x 2 = $0.20
RES SMD 560 OHM 1% 1/10W 0603	DigiKey	13-AF0603FR-07560RLTR-ND	AF0603FR-07560RL	6	$0.10 x 6 = $0.60
RES SMD 10K OHM 1% 1/10W 0603	DigiKey	13-RT0603FRD0710KLTR-ND	RT0603FRD0710KL	6+6+8	$0.15 x 20 = $3.00
RES 3K OHM 1% 1/10W 0602	DigiKey	RMCF0603FT3K00TR-ND	RMCF0603FT3K00	2	$0.10 x 2 = $0.20
RES 240 OHM 1% 1/10W 0603	DigiKey	311-240HRTR-ND	RC0603FR-07240RL	2	$0.10 x 2 = $0.20
IC PWR RELAY 7NPN 1:1 16SOIC	DigiKey	296-41065-2-ND	ULN2003BDR	2	$0.71 x 2 = $1.42
RES 715 OHM 1% 1/10W 0603	DigiKey	311-715HRTR-ND	RC0603FR-07715RL	2	$0.10 x 2 = $0.20
RES 82K OHM 1% 1/10W 0603	DigiKey	RMCF0603FT82K0TR-ND	RMCF0603FT82K0	2	$0.10 x 2 = $0.20
SWITCH TOGGLE SPDT 5A 120V	DigiKey	EG2355-ND	100SP1T1B4M2QE	2	$2.59 x 2 = $5.18
LED GREEN CLEAR 0603 SMD	DigiKey	732-4980-1-ND	150060VS75000	8	$0.15 x 8 = $1.20
TERM SCREW 6-32 4 PIN PCB	DigiKey	36-8191-ND	8191	12	$0.47 x 12 = $3.76
IC GATE DRVR LOW-SIDE 8SOIC	DigiKey	MCP14A0452-E/SN-ND	MCP14A0452-E/SN	12	$1.29 x 12 = $15.48
IC DECODER/DEMUX 1X2:4 16SOIC	DigiKey	296-8229-5-ND	SN74HC139D	2	$0.47 x 2 = $0.96
DEPEPE 30 Pcs 40 Pin 2.54mm Male and Female Pin Headers	Amazon	DE37566	7.10E+11	1	$5.39
Primary PCB	PCBWay	-	-	2	$4.90 x 2 = $9.80
Weather Proof Enclosure	Amazon	-	-	2	$30.99 x 2 = $61.98
Cable Glands	Amazon	-	-	16	$0.40 x 16 = $6.40
MC4 Connectors	Amazon	-	-	12	$9.99/6 x 2 = $19.98
FTR-J2 Series Relay	Provided	N.A.	FTR-J2AK012W	4	$0
Total	-	-	-	-	= $219.93

Appendix F: Schedule of Work

Table 1: Schedule of Work (per teammate)

Week	Sydney	Maram	Nikhil
09/27/21	Perform additional research regarding the hardware aspect whilst working on our PCB and review unique safety concerns	Sent first draft of PCB schematic, Talked with Prof. Banerjee and Kevin about relay configuration	Completed researching and planning wireless communication framework between external server and microcontroller
10/04/21	First schematic draft completed and completion of the final version of our design document, Performed additional research regarding ADC isolation	Continued to discuss our relay configuration with 445 TAs, Performed various voltage and current monitoring updates and research	Completed researching and planning communication paths for remote configuration (panel->relay->microcontroller->server)
10/11/21	Met with professors to discuss further improvements on our schematic design and visited the rooftop to observe the panels and connections	Worked further on our schematic, Discussed our design further with Prof. Banerjee, his team, and our TA	Planned integration paths of work done so far, testing approaches, and integration of hardware and software with regards to ordered parts
10/18/21	Performed simulations for our power subsystem to ensure that the proper voltage will power the rest of the components onto our PCB	Researched further in isolation in order to implement an ADC converter, Continued sending updated schematics to TA in order to begin working on PCB.	Compiled work for entire Django project for solution (front-end and back-end); Almost completed front-end work including wireless capabilities
10/25/21	Working on implementing the relay subsystem onto our board utilizing the relays that have been provided to us from our meeting with Kevin, Also utilize as an ESP32 microcontroller as a control instead of a switch	Sent PCB design files to TA, Continued to update PCB based on feedback from TA, Conducted research on Darlington array to exchange out for our relay drivers	Completed preliminary front-end Django work, Completed coding out skeleton for Django back-end (server), Completed skeleton for ESP32 software programming
11/01/21	Implemented TA suggestion of replacing many components from our relay subsystem with a darlington array as a relay driver to directly switch the configurations	New PCB configuration sent, Updated darlington array, current output, and switch for our linear regulator, Continued to send PCB updated to TA for approval	Completed researching and planning Django real-time updates for front-end, Began implementation of ESP32 stand-alone wireless capabilities (ran into some issues)
11/08/21	Researching the hardware communication aspect of our project with the ESP32	Sent the final version of the board to TA and received	Successfully completed ESP32 stand-alone wireless capabilities and integrated it with Django server wireless

	microcontroller.	approval, Ordered PCB	capabilities, Updated Django front-end to meet independent requirements
11/15/21	Part testing prior to the PCB arrival, We tested whatever we could on breadboards and breakout boards so that we could ensure that some parts worked in advance	ADC arrived and began to test parts prior to PCB arrival, Conducted OLED and thermocouple isolated testing, Implemented pull up resistors, Began working on communicating with ADC	Researched and planned out code updates as per mock demo feedback (Django model updates, reverse HTTP pathway and GUI)
11/22/21	Soldered and debugged most of our subsystems, Worked on isolating components before soldering and implementing into our project.	ADC component testing continued, Soldered and troubleshooted components on PCB, Integrated our thermocouple readings onto OLED, Successfully tested darlington array on breadboard, Talked to the ECE machine shop to re-drill holes in the PCB	Began hardware programming for ESP32 and PCB and completed majority of it, Completed individual integration testing of parts and began PCB assembly
11/29/21	Went up to the rooftop to access the solar panels and observe solar panel parameters onto our PCB, Further calibration had to be done with the readings, Completed Project Demo	Current sensor and ADC testing. Soldered pull up resistors on PCB, Troubleshooting on PCB with various components, Calibrations for monitoring systems, Completed Project Demo	Completed all adjustments to ESP32 programming and for hardware integration, Implemented planned Django changes and integrated it, Completed Project Demo
12/06/21	Finalized presentation and presented; Prepared and submitted final paper	Finalized presentation and presented; Prepared and submitted final paper	Finalized presentation and presented; Prepared and submitted final paper