Multi-Adjustable Port Station (MAPS)

Group 24

Members:

Hunter Volonino

Sebastian Sotomayor

Siya Sharma

Jonathan Luna

Reviewers:

Dr. Nazanin Rahnavard

Dr. Azadeh Vosoughi

Dr. Sonali Das

Dr. Avra Kundu

Our Team

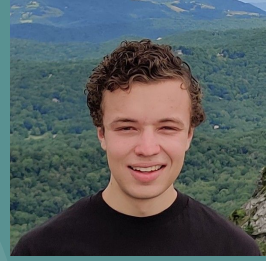
Group 24

Hunter Volonino
Computer Engineering

Sebastian Sotomayor
Computer Engineering

Jonathan (John) Luna
Electrical Engineering

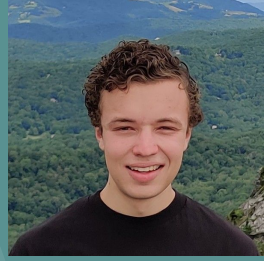
Siya Sharma
Electrical Engineering



Motivation and Background

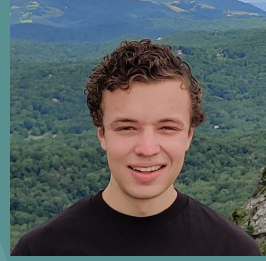
- Lots of battery-powered technology means it must be charged
- Some hobbyists may have many devices used on a semi-frequent basis
  - If a device is left plugged into a normal charger indefinitely, the battery will degrade quickly
  - If a device is left unplugged for a long period of time, the battery will go flat
- We must have some way to control charging!

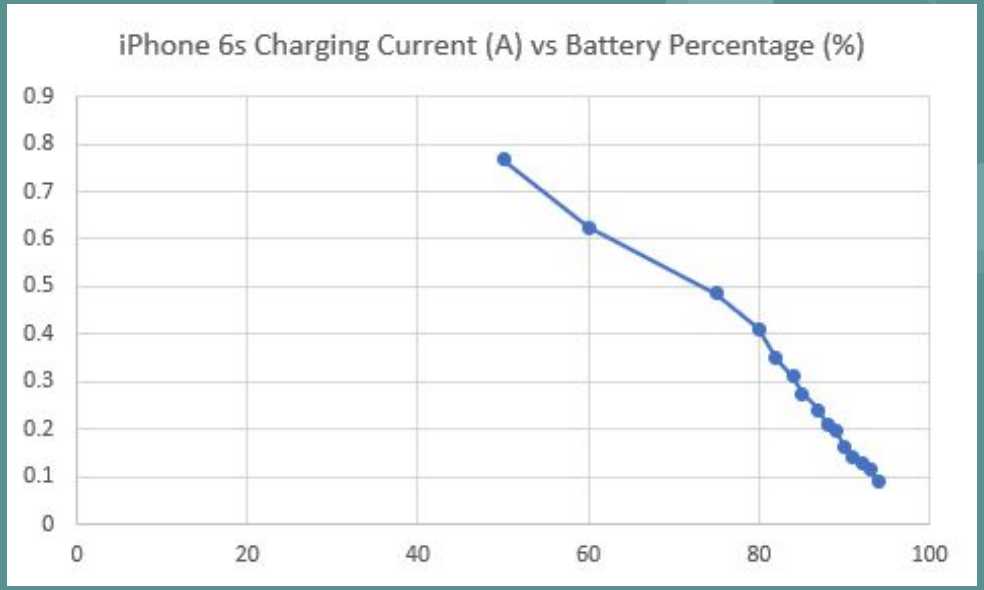
Goals & Objectives



- How can we control the charging of any device that charges over USB?
  - Monitor charging current!
  - As a device charges, the current will continuously drop
  - Allow the user to set a current threshold to stop charging and a time to start charging again!
- In order to determine a current threshold, users must know the charging current of the device in the first place
  - Charging current, voltage, and power should be visible to the user interface and consistently updated in real-time
- The charger should respond quickly
  - Devices should start charging when plugged in
  - When a device consistently drops below the set threshold, the port should be turned off quickly
- The charger should be able to detect whether a device is plugged in or not
- We would also like to build a custom enclosure for the charger

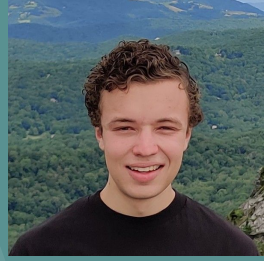
Goals & Objectives: Example





The above graph shows a test run with a 5V 1A USB charger and a USB tester to gather data.

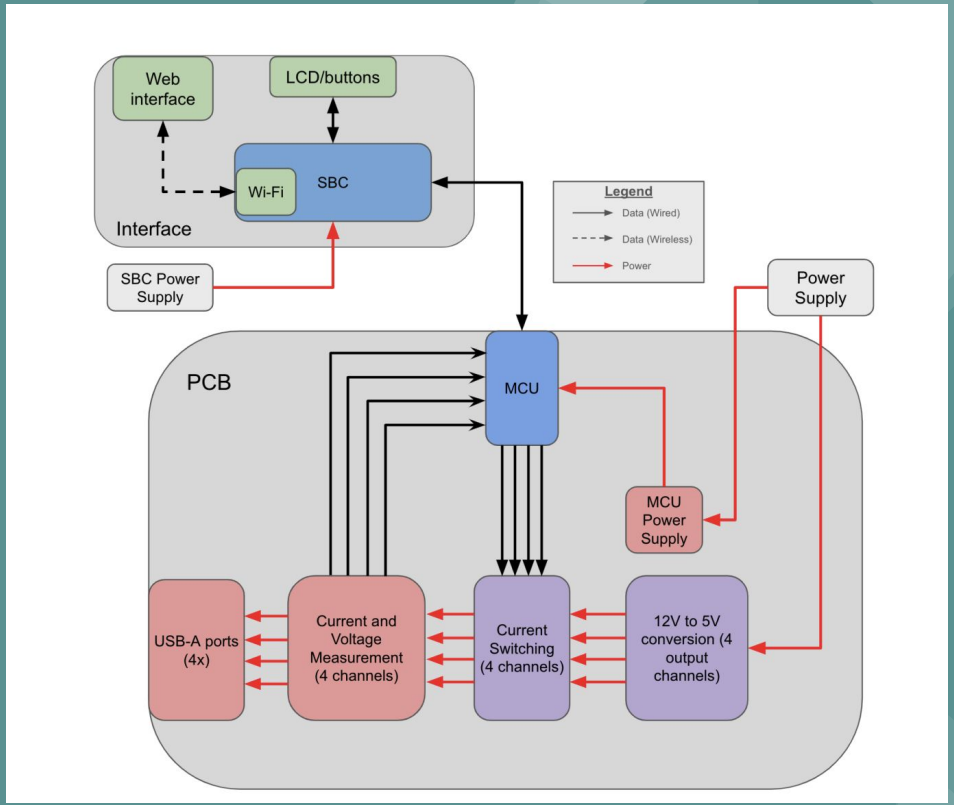
Engineering Specifications



Parameter	Description	Value
Charge Port Voltage	Acceptable voltage of each USB port	5V ± 0.25V
Max Port Output Current	Acceptable output current to charge a wide variety of devices	750mA ± 250mA
User-Set Min Charging Current Range	Range over which the user can set the minimum allowed current	0mA-1000mA
Charging Plug-in Delay	When a device is plugged in, the charger must respond quickly to start charging the device.	< 5 seconds
Charge Cut-off Delay	When a device drops below its set minimum charging current, the charger must respond quickly.	< 1 minute

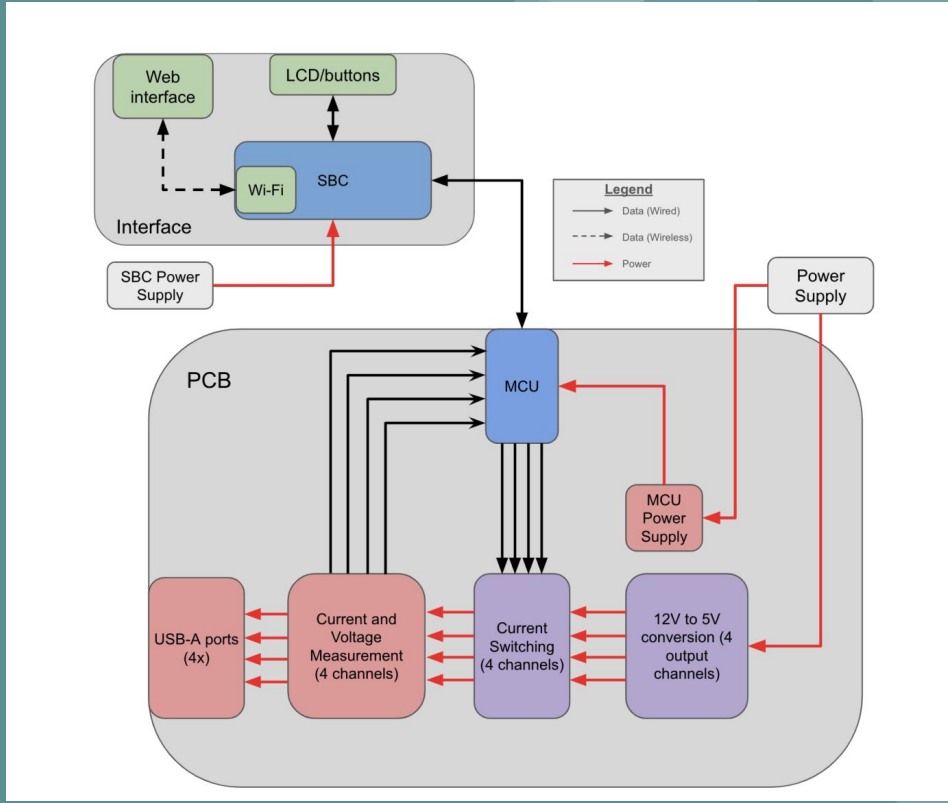
Hardware Design

- **System Overview:** consists of UI and control, 4-port USB charging PCBs
- **UI Layer:** Includes web interface, touchscreen to view charging settings/statuses and updates in real-time
- **SBC:** High-level controller that aids in communicating signals and commands to the charger
- **Wi-Fi Connectivity:** The user can view the statuses at any time through the web interface



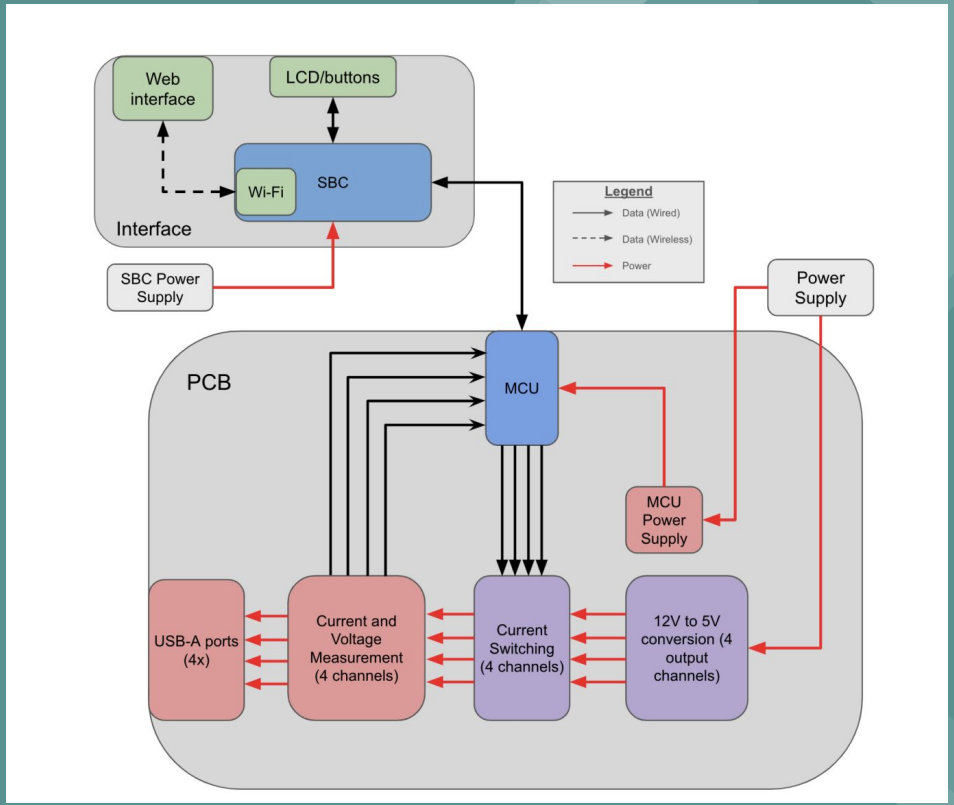
Hardware Design

- **MCU on PCB:** Allows to read the sensor data, reliable for the per-port switching. Essentially, the hardware control is handled in real time.
- **4 USB-A outputs:** Devices connected are able to charge concurrently.
- **Current/Voltage Measurements:** Each port where a device is connected monitors and reports the current and voltage statuses.



Hardware Design

- **Current Switching:** MCU is able to enable/disable each of the ports based on the user's request. This can be done independently.
- **Power path:** Power input where it feeds to the PCB. Note that within the overall device there is the 12V to 5V conversion for each of the USB ports.



Hardware Prototype

Control Module:

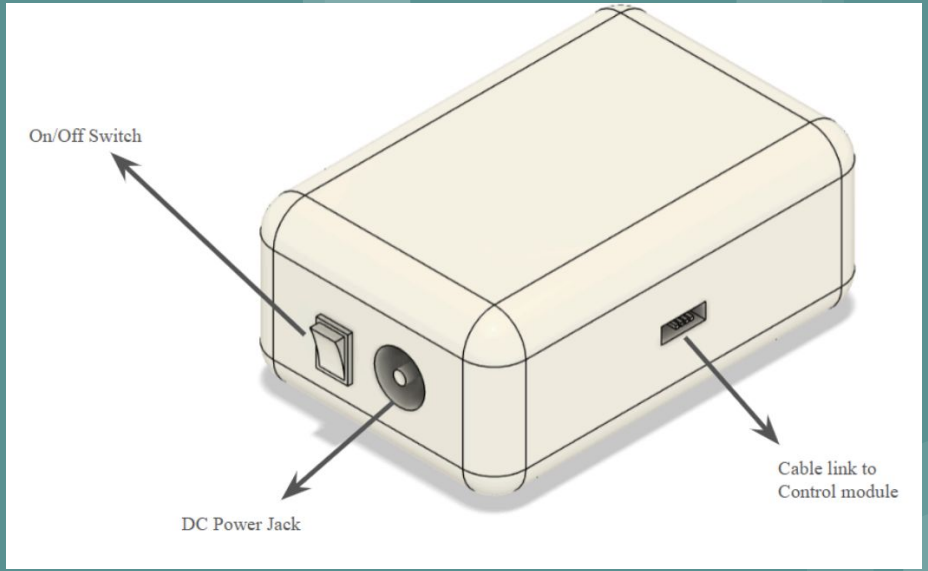
Purpose: The control module allows for the user to monitor the individual USB ports as well as the charging status of each of the devices that are connected.

Properties:

- **LCD screen**: Voltage/current/power is shown or displayed on each of the port statuses.
- **Touchscreen:** Allows for the user to maneuver through the menu and adjust any necessary settings. For example, choosing the specific port or adjusting the minimum current range.
- **Cable to charger:** Bidirectional communication exists between both the SBC in the control module and the MCU in the charger PCB.

Hardware Prototype

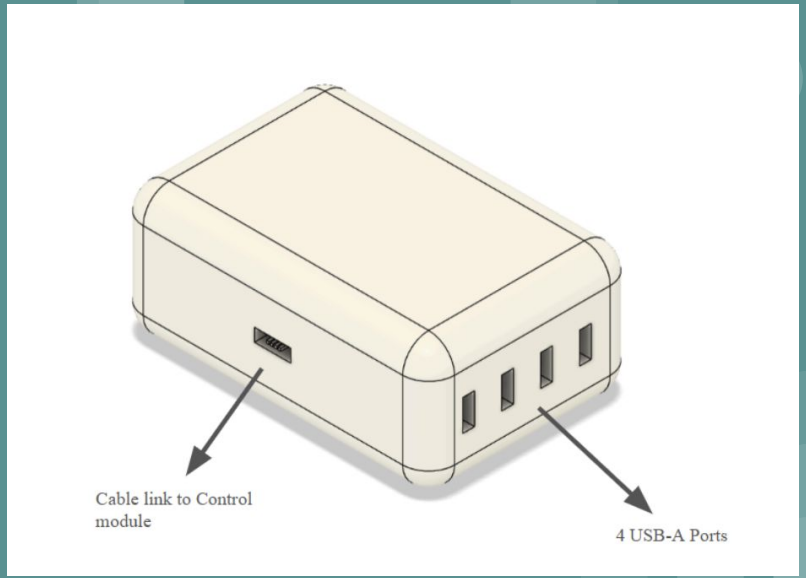
- **ON/OFF Switch**: The user is able to switch on/off to start up or shutdown the charger module.
- **DC Power Jack**: For all ports to be supported, feeds power to support each of the ports via their regulators as well as the MCU.



Isometric View of Charging Module (Front)

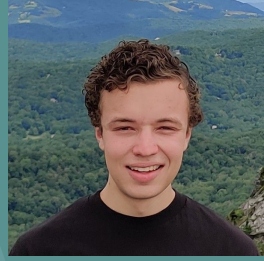
Hardware Prototype

- **4 USB-A Ports:** There will be 4 ports on the charging module to allow for each of the devices connected to charge concurrently. There is also per-port control so this means, each of the ports can be handled independently as well.
- **Cable Link to Control Module:** Allows for communication with the control module so any necessary settings/commands/statuses can be displayed and received in real-time.



Isometric View of Charging Module (Back)

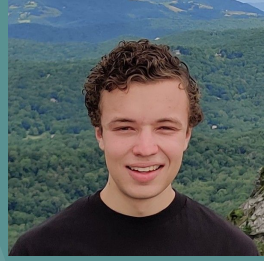
Comparison and Selection of Hardware: MCU



- We have selected an ESP32-WROOM family microcontroller.
- Aside from being one of the fastest options, the ESP32 is also one of the most used.
- This provides a great benefit for ease of development!
- Its main limitation is the low GPIO count, but with 4 charging ports and one MCU interface it is fine for our use.

	ESP32-WROOM-32E-N16	TI MSP430FR6989	Raspberry Pi Pico RP2350B	ATmega 2560	STM32 F722ZE
Processor	240MHz Dual-Core	16MHZ Single Core	150MHz Dual-Core	16MHz Single-Core	216MHz Single-Core
RAM	520KB + 2MB	2KB + 128KB	520KB	8KB	256KB
GPIO Count	26	83	48	86	114
Protocols	UART, I2C, SPI, PWM, Wireless	UART, I2C, SPI, PWM	UART, I2C, SPI, PWM	UART, I2C, SPI, PWM	UART, I2C, SPI, PWM
Cost (chip only)	$5.71	$9.47	$1.00	$17.59	$14.06

Comparison and Selection of Hardware: SBC



- We have chosen a Raspberry Pi 4 Model B 4GB for our single-board computer.
- The Pi 4 is well documented, and its hardware is more than suited for running the user and web interfaces.
- A Pi 5 or NanoPi M5 would be faster, but is more expensive with a higher power draw and little benefit for our use.

	Raspberry Pi 5 B	Raspberry Pi 4 B	Raspberry Pi Zero 2W	Le Potato Pi	NanoPi M5
Processor	2.4GHz 4-Core	1.8GHz 4-Core	1GHz 4-Core	1.416GHz 4-Core	2.2GHz 8-Core
RAM	2GB/4GB/ 8GB/16GB	1GB/2GB/ 4GB/8GB	512MB	2GB	4GB/8GB/16GB
Wi-Fi	802.11ac 2.4+5GHz	802.11ac 2.4+5GHz	802.11n 2.4GHz	No	No
GPIO Count	40	40	40	40	40
Cost	$50-$132	$38.50-$82.50	$19.80	$35	$55 (4GB)

Comparison and Selection of Hardware 5V reg

- The LM2670 was chosen for our 5V regulator
- Low switching frequency
- Lower efficiency requires PCB considerations for other auxiliary boards

	TPS56339	**TPS563200**	**MAX1639**	**LM2670**
Brand	Texas Instruments	Texas Instruments	Analog Devices	Texas Instruments
Input Voltage Range	4.5 V - 24 V	4.5 V - 17 V	8V - 40 V	3.5 V - 36 V
Switching Frequency	500-kHz	650-kHz	220-kHz - 2.2MHz	260kHz
Efficiency	95.2%	94.6%	91%	85%

Comparison and Selection of Hardware 3.3V reg

- MP1584 chosen as 3.3V Regulator
  - External Resistor allows for fine tuning calibration
  - Cheaper price
- Frequency < 1MHz for our configuration

	MP1584	**LM2596**	**AMSR-783.3-NZ**
Brand	Monolithic Power Systems	Texas Instruments	Recom Power
Input Voltage Range	4.5 V - 28 V	4.5 V - 40 V	4.75V-28V
Switching Frequency	100kHz-1.5MHz	150-kHz	330 kHZ
Efficiency	~90	94.6%	91%

Comparison and Selection of Hardware Current Measuring IC

- The INA260 was chosen as our current measuring IC
- Utilized in Battery charger systems + greater documentation for our use case
- Integrated Shunt resistor allowed for easier design and implementation in PCB

	INA260	**INA230**	**MAX34407**	**ACS712**
Brand	Texas Instruments	Texas Instruments	Analog Devices	Allegro Microsystems
Signal Format	I2C	I2C	I2C	Analog
Max Current	15A continuous	Determined by Shunt Resistor	Determined by Shunt Resistor	5A/20A/30A versions
Integrated shunt	2mΩ	No External Shunt	No External Shunt	No Hall Effect
Bus Voltage	0V-36V	0V-28V	2.7V-28V	Depends on Isolation

Comparison and Selection of Hardware IC Load Switches

- TPS22918 chosen for IC load switch
- Quick Output Discharge ensures no floating output

	TPS22918	**LS0504EVT223**	**DML3012LDC**
Brand	Texas Instruments	Littlefuse	Diodes Incorporated
Max Continuous Current	2A	2-4A	~3A
Input Voltage Range	0-5.5V	2,7V-7V	1.2-6Vr
Internal Soft-Start	No	Yes	Yes
Quick Output Discharge (QOD)	Yes	No	Yes

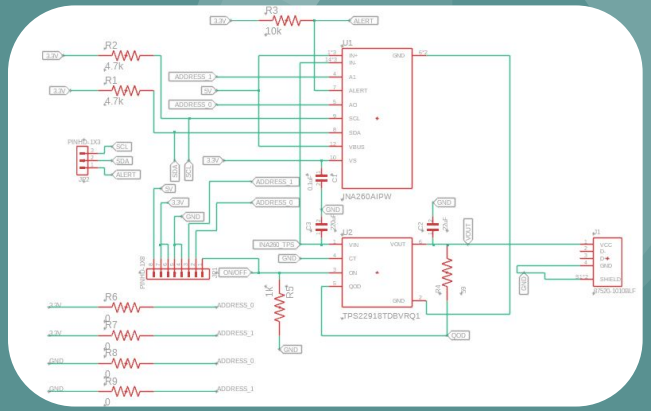
Comparison and Selection of Software

- The User Interface uses a modified traditional MERN stack.
- React enables efficient real time UI updates to reflect MCU controlled charging data
- Node.js with Express handles UART ingestion, WebSocket broadcasting, and REST API requests.
- SQLite on Raspberry Pi OS provides a lightweight, embedded database solution.

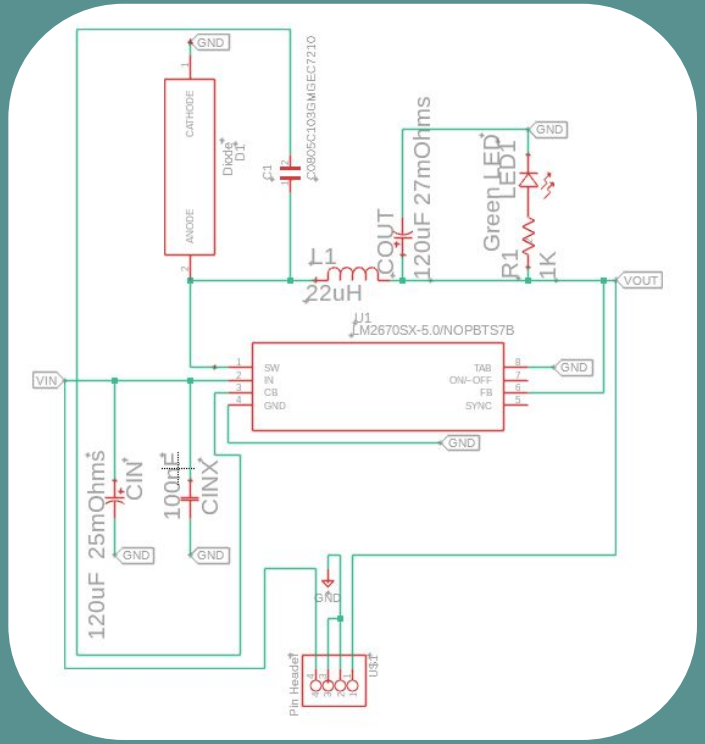
Web Stack	Frontend	Backend	Database	Operating System
LAMP	HTML/CSS /JavaScript	PHP	MySQL	Linux
Modified MERN	React	Node/ Express	SQLite	Raspberry Pi OS

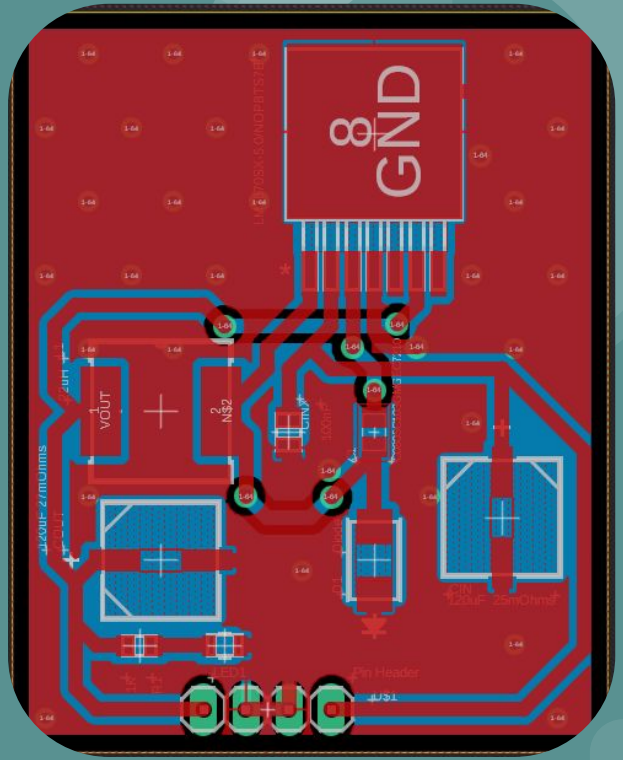
Significant PCB Design Current Switch

- INA 260
  - Measure Current
  - Measure Voltage
  - Communicates to MCU with I2C
  - 0 ohm jumpers to determine I2C address
- TPS22918
  - Quick Output Discharge prevents floating output
  - Is either on or off through MCU connection
  - Capacitors values chosen to prevent input voltage from dipping



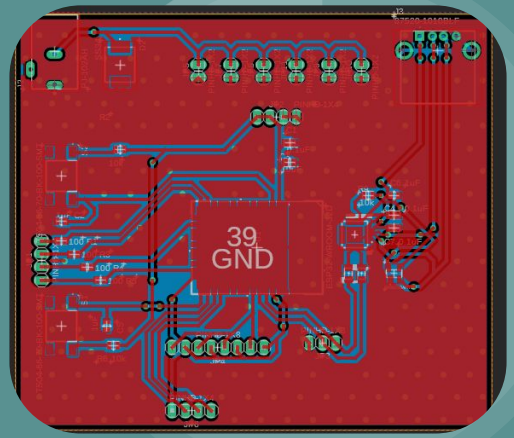
Significant PCB Design
5V Regulator

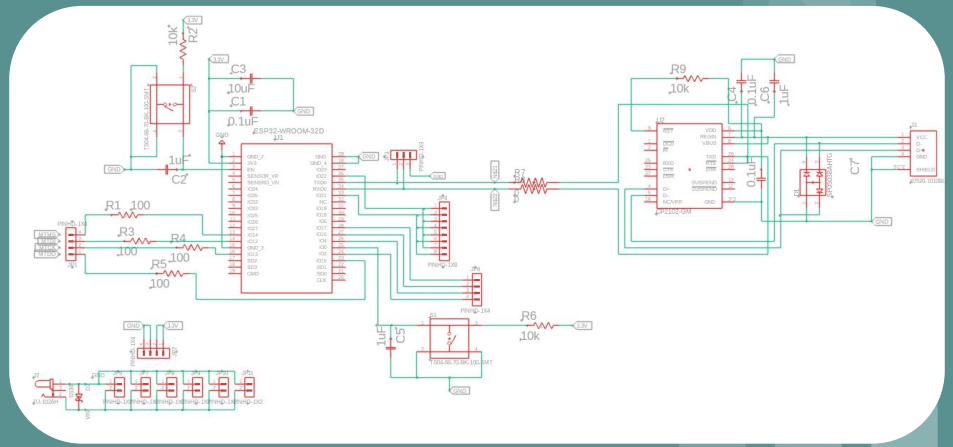




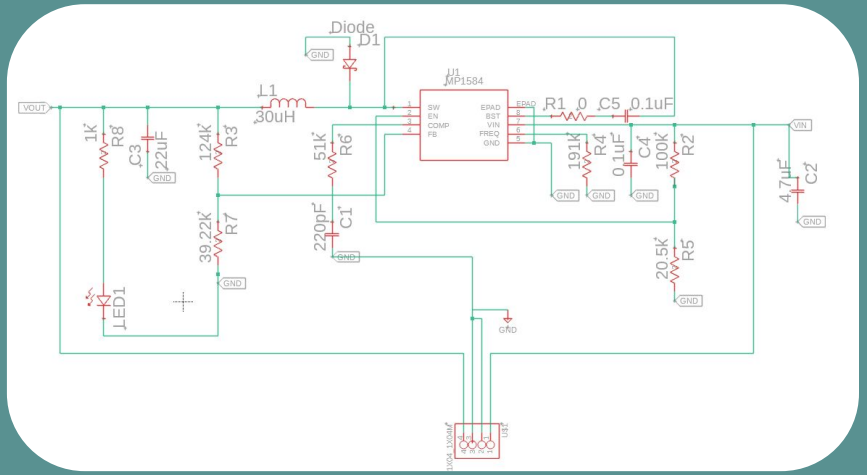
Significant PCB Design MCU

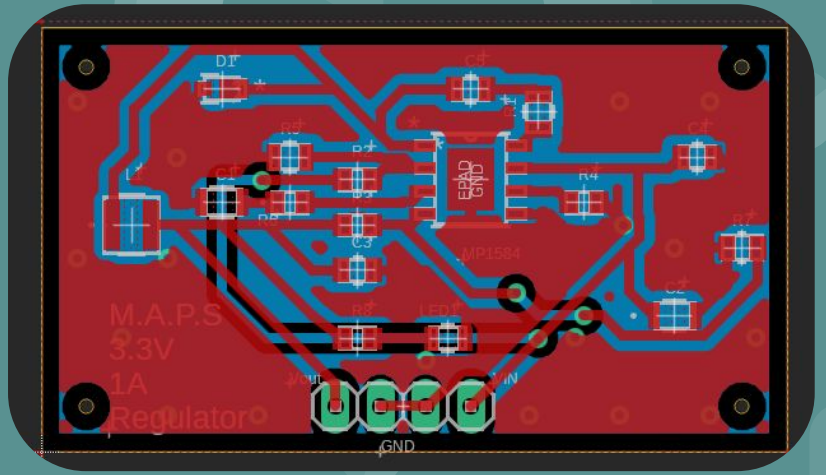
- Microcontroller
  - Communicates with Raspberry Pi to relay INA260 information
  - Will read data from INA260 through I2C GPIO 21 & 22
  - Output GPIO configured to send outputs signal to TPS22918





Significant PCB Design
3.3V Regulator



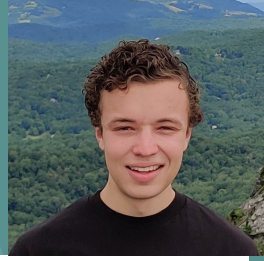


Hardware Housing Design

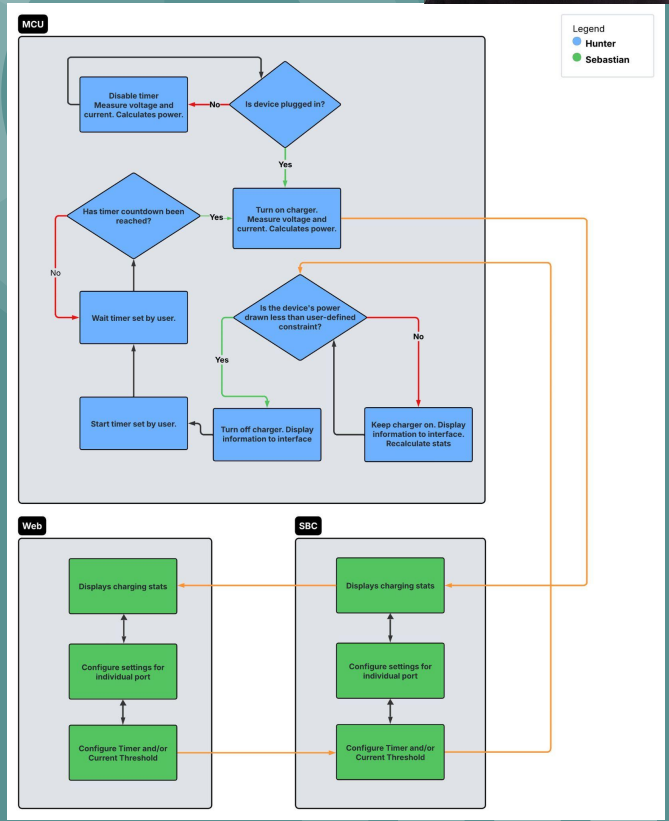
Incorporating a 3D Enclosure/ Plexiglass

- **Objective**: House all parts of our device into a singular enclosure. This includes the front UI, rear ports, and an internal layout.
  - Easy access for the UI (LCD touchscreen display)
  - Connection zone within the enclosure where each of the USB ports as well as the DC input is designed to limit and reduce interaction with any possible internal wiring.
  - Electronics zone within the enclosure where the PCB sits closest to the ports, possibly at the bottom of the enclosure. SBC/Wi-Fi sits above for easy accessibility and to avoid other sections with higher current.
  - Clean wire routing to limit any loose wiring connections, easy access for debugging, and keeping the wires in one place. May design this with a removable top for easy access in any point in time.

Software Design: MCU

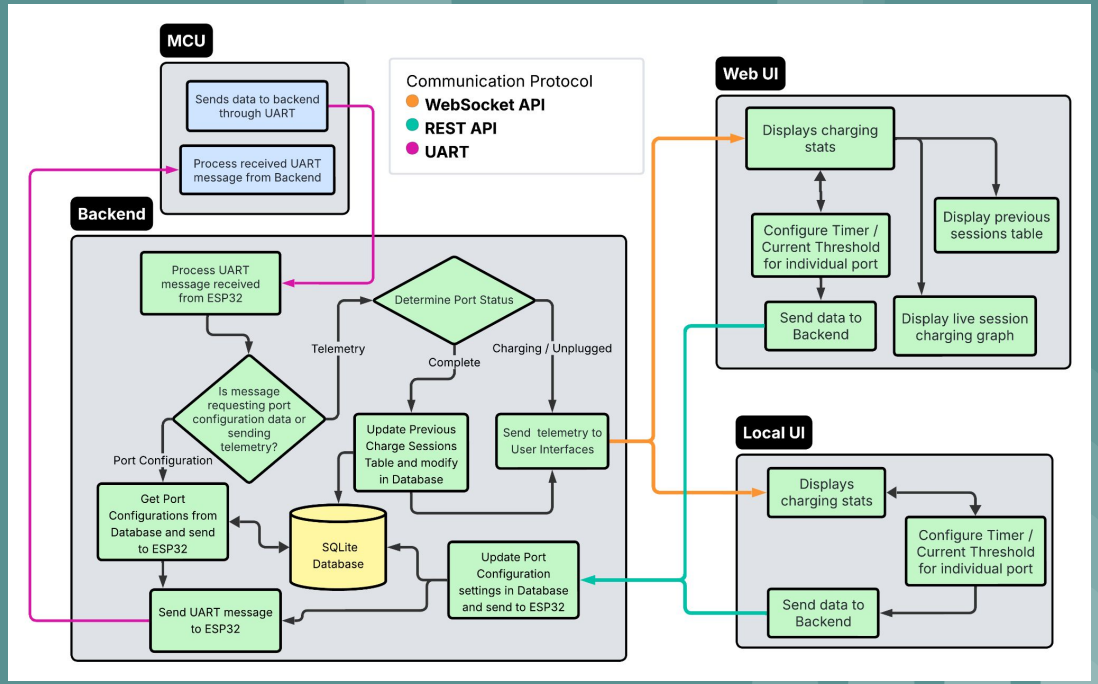


- We will be using FreeRTOS tasks and taking advantage of the ESP32's two cores.
- We will be programming the ESP32 in C++, but the code is also Arduino IDE compatible.
- Several periodic tasks exist in order to check charging stats, make decisions based on charging stats, switch ports on and off, detect if a device is plugged in, and transmit data to/from the SBC's user interface.



Software Design: SBC

- A user can set port configurations through either user interface
- The backend processes and delivers messages between the MCU and both frontends (UI)
- The database stores port configurations set by the user, and data from the previous five charge sessions per port



Local User Interface

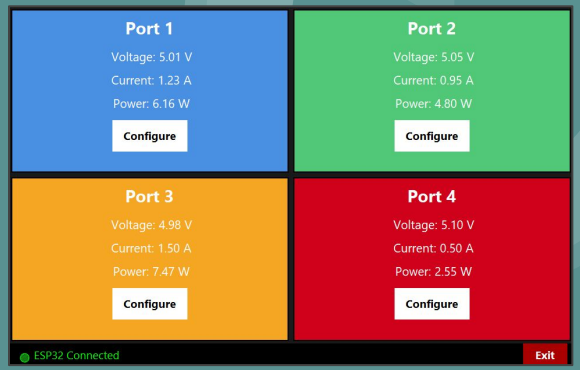
- The local UI is displayed on an LCD touchscreen connected to the SBC
- It is built using the Tkinter Python library and it communicates with the ESP32 through the backend

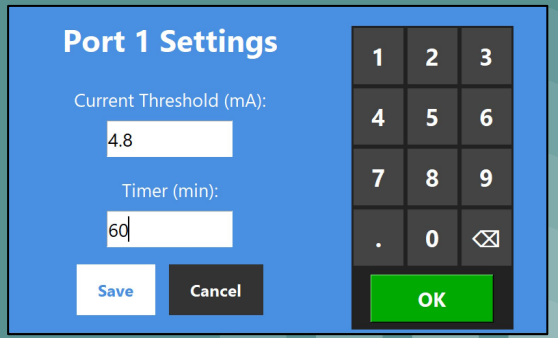
Home Screen

- Displays real time data from ESP32 to the user

Configuration Menu

- Allows adjustment of each port's current threshold and timer settings





Website User Interface

- The website UI is hosted by the Raspberry Pi
- It can be accessed by any device that's on the same Wi-Fi network

Dashboard

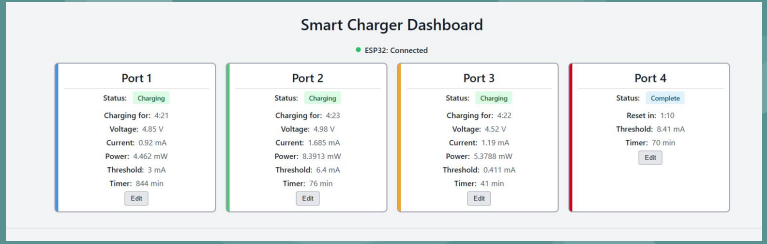
- Displays real time data from ESP32 to the user
- Allows adjustment of each port's current threshold and timer settings

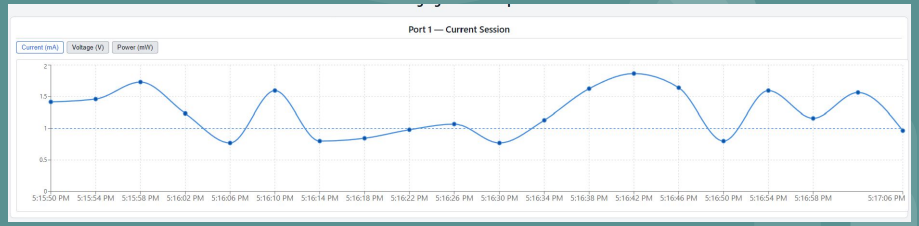
Charge Graph

- Displays real time charge data from ESP32 to the user
- Displays live current/voltage/power over time graph

Previous Charge Session

- Displays stats from previous charge sessions
- Data is stored in SQLite Database





Port 4

Start Time	End Time	Charge Duration	Avg Current	Avg Voltage	Avg Power	Energy Delivered	End Reason
1/25/2026, 5:11:16 PM	1/25/2026, 5:11:17 PM	0:01 min	1.86 mA	4.80 V	8.95 mW	0.00 mWh	Reached threshold
1/23/2026, 8:56:06 PM	1/23/2026, 8:56:22 PM	0:16 min	1.52 mA	4.42 V	6.68 mW	0.02 mWh	Reached threshold
1/22/2026, 4:31:55 PM	1/22/2026, 4:32:03 PM	0:08 min	1.02 mA	4.31 V	4.33 mW	0.01 mWh	Reached threshold
1/22/2026, 4:31:51 PM	1/22/2026, 4:31:52 PM	0:01 min	0.84 mA	4.15 V	3.51 mW	0.00 mWh	Reached threshold
1/21/2026, 11:47:35 PM	1/21/2026, 11:54:32 PM	6:56 min	1.27 mA	4.47 V	5.68 mW	0.65 mWh	Reached threshold

Application Programming Interface (API)

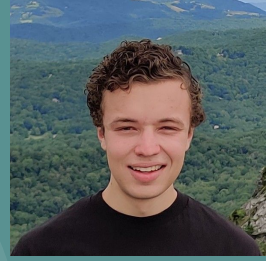
- The SBC hosts both REST and WebSocket APIs.
- REST API sends configuration data to the ESP32
- WebSocket API streams live telemetry from the ESP32 to the client.

	REST API	WebSocket API	GraphQL API
Purpose	Handle on-demand data retrieval	Provides real time data delivery	Flexible querying of structured data
Use Case	Configuration updates and settings retrieval	Live system monitoring and event streaming	Large-scale applications with dynamic data requirements
Resource Usage	Low CPU footprint	Moderate resource usage	High overhead
Complexity	Low complexity, simple to develop and debug	Moderate complexity	High complexity, advanced backend logic

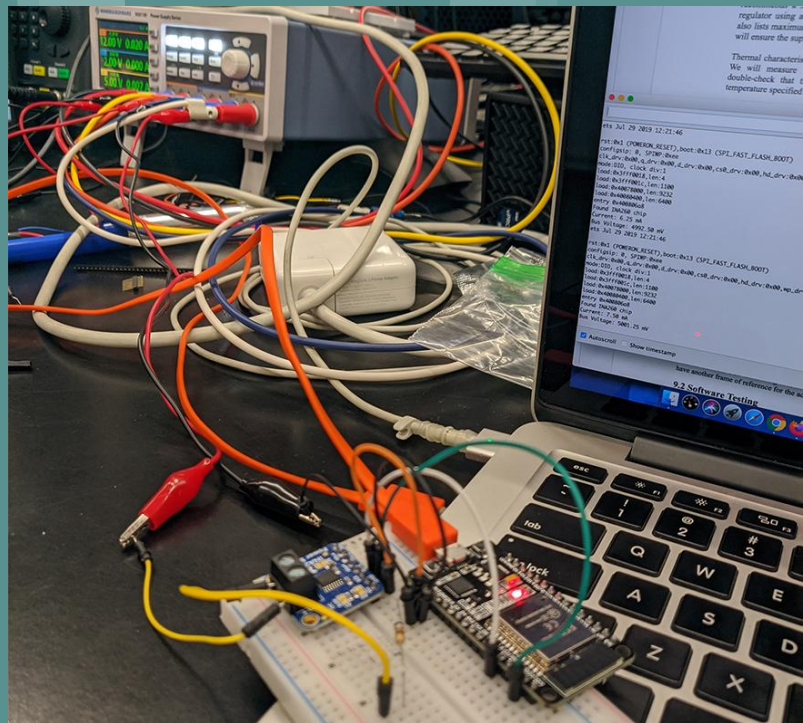
Prototyping and Testing (Hardware Testing)

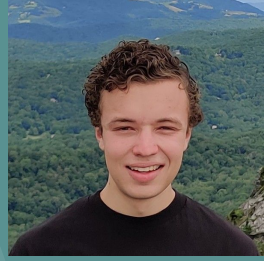
- **Goal**: Confirm power regulation, USB control ports, voltage and current accuracy, and false detections .
- **Power rails**: Confirms 12V input, the regulated 5V and 3.3V. Also considering no-load and under load we want to check for the voltage drops during these specified load changes.
- **Per-port switching:** Verify that the ESP32 can turn each of the USB ports ON/OFF; Confirm Quick Output Discharge (QOD) when a USB port is turned off so that there is essentially no “floating” voltage.
- **Measurement accuracy:** Accuracy of the INA260 voltage/current/power readings using I2C. See how this compares against using a USB multimeter.
- **Load testing:** Test each port at increasing loads, for example: 0.5A, 1A, ~2A, to see if it works and also perform a full system test with all the USB ports to make sure system correctly works when everything is running simultaneously.

Hardware Testing (Current Measurement)



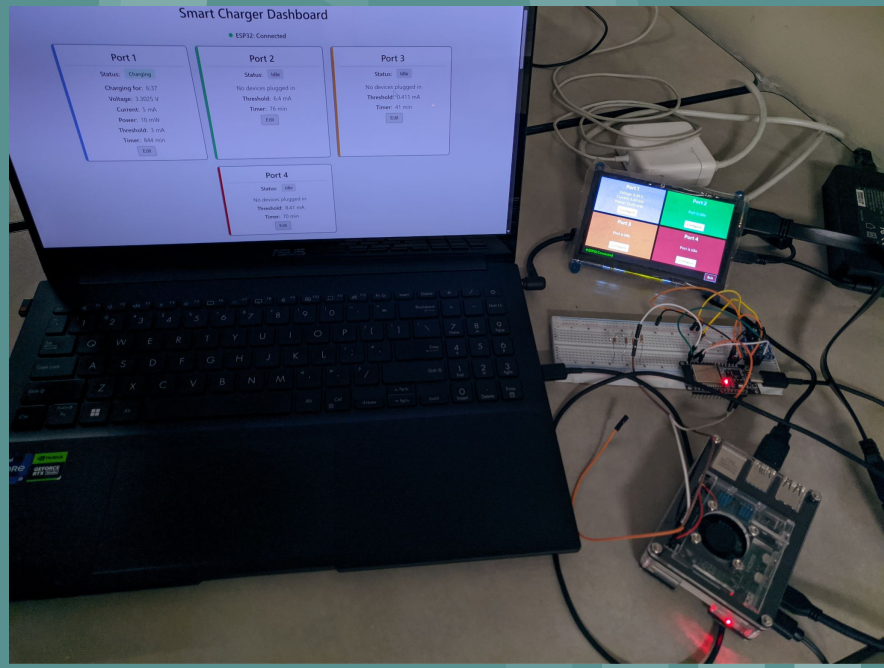
- Pre-made breakout boards utilizing the INA260 current measurement IC exist
- We can connect this breakout board to an ESP32 development board over I2C
- There, we can test our INA260 with code written for the ESP32, and a breadboard with a load resistor.
- Measuring the current and voltage with a DMM, the error is acceptable!





Prototyping and Testing (Software Testing)

- 2 parts of software testing: UI/SBC testing, and microcontroller testing
    - The UI and SBC interface were tested with an ESP32 generating simulated values.
    - Communication over UART works flawlessly in our test environment!
    - INA260 current/voltage/power measurement also works on both ends.
- Some microcontroller code relies on the load switch, which we cannot fully test until the PCB is assembled and otherwise operational.



Project Budget

Part	Description or P/N	Cost (incl tax/shipping)
PCBs (First Revision)	Ordered from JLCPCB	$129.35
Components	Ordered from DigiKey	$469.60
Raspberry Pi 4B & Accessories		$75.00
Display	ELECROW 5” touchscreen LCD	$39.99
INA260 Breakout Board	Used for testing. Adafruit #4226	$27.61
USB Current Measurement Tool	KWS-MX18L	$15.71
Total spent:		$741.55

Work Distribution

Hunter Volonino	Project Lead	**Sebastian Sotomayor**	Local User Interface
	Microcontroller & Hardware Interface		Web User Interface
	PCB Assembly		SBC and MCU Communication
Jonathan Luna	Regulator & Load Switch Selection	**Siya Sharma**	Current Measurement Selection
	Overall PCB Design		Device Enclosure Design
	PCB Testing		PCB Assembly

Progress and Plan for Completion

- Our PCBs have arrived, and our components are on the way. We are aiming for assembly within the week.
- The software is finished in prototype form, but will need to be tested and debugged once the charger is assembled.
- If issues arise, we will need to re-order parts!

Thank you!