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

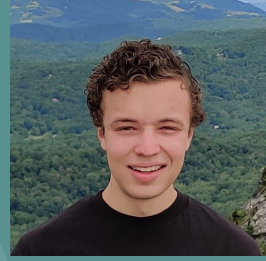
Group 24

Hunter Volonino
Computer Engineering

Sebastian Sotomayor
Computer Engineering

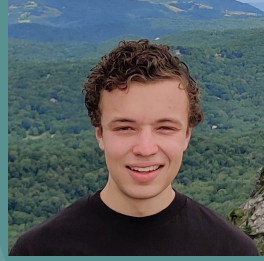
Jonathan (John) Luna
Electrical Engineering

Siya Sharma
Electrical Engineering



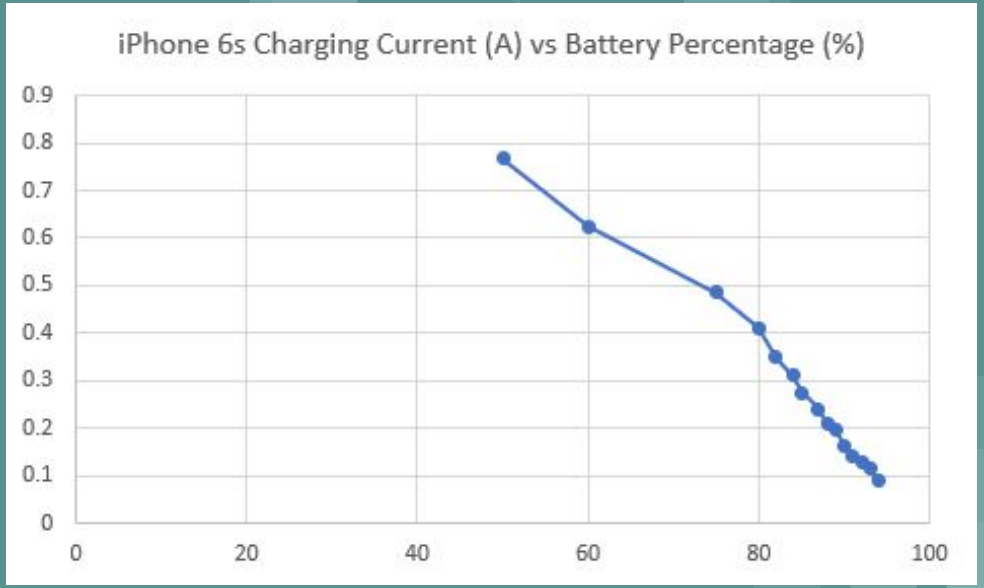
Motivation and Background

- 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!

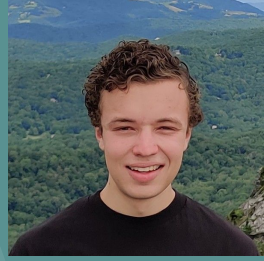


Our Solution: Preface

- The graph on the right shows a test run with a 5V 1A USB charger and a USB tester to gather data.
- You can see that as the indicated device battery percentage (horizontal axis) increases past a certain point, the current draw from the charger decreases.
- This can vary by amount and linearity by device, but nearly all devices with a lithium-ion battery follow this same general trend.
- If we set a minimum current threshold, then turn off the port below the threshold, we can control charging for nearly any device!

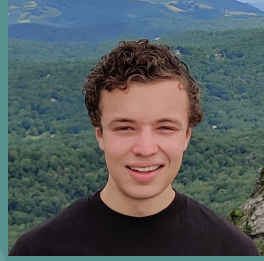


Our Solution: Goals & Objectives



- Goals
  - Create a four-port device charger that distributes charging current over USB to charge a wide variety of devices
  - Allow each charging port to be adjusted individually, and turned-off once the connected device has charged a specific amount
  - Inform user of charging statistics
  - Start charging quickly when plugged-in
  - Build a custom enclosure for the charger
- Objectives
  - Integrate the device with a DC power supply and four switchable ports
  - Allow current, voltage, and power to be read in real-time
  - Display the voltage, current, and power in real time on both a local user and web user interface
  - Once the charging current drops below a set value, shut off current to that port
  - Be able to detect if a device is plugged-in or not.

Engineering Specifications



- Acceptable voltage and current output to all devices to ensure compatibility
- The current threshold should be able to be set to reasonable values.
- Power should be distributed to a plugged-in device quickly
- When charging current threshold is reached, the port should turn off quickly

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

Specification List

Hardware Housing Design

Incorporating a 3D Enclosure

- **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 will be placed followed by the 3.3V regulators, 5.25V Regulators and current switches, and the power distribution board from top to bottom of the enclosure.
  - Clean wire routing to limit any loose wiring connections, easy access for debugging, and keeping the wires in one place. Windows have been designed throughout the enclosures for visibility and connection purposes.

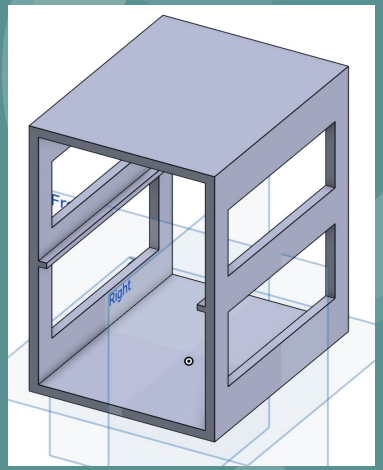
Hardware Prototype

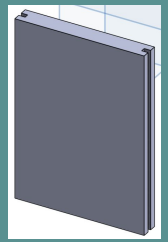
Control Module:

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.

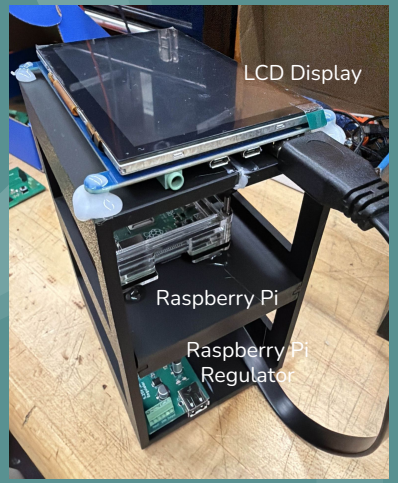
Properties:

- **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.





Control Module CAD half (top) with shelf (left)



Control Module with PCBs, SBC, and LCD display installed

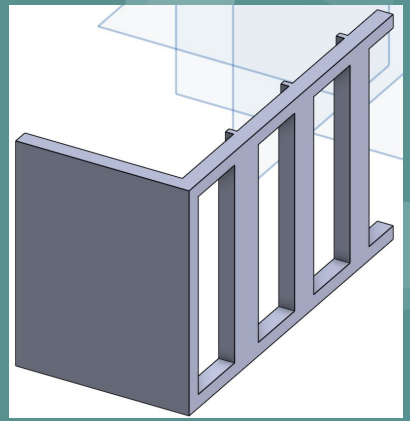
Hardware Prototype

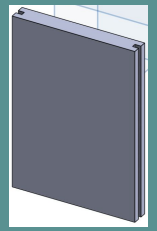
Charger Module:

Purpose: The charger module allows for the user to connect their devices for charging.

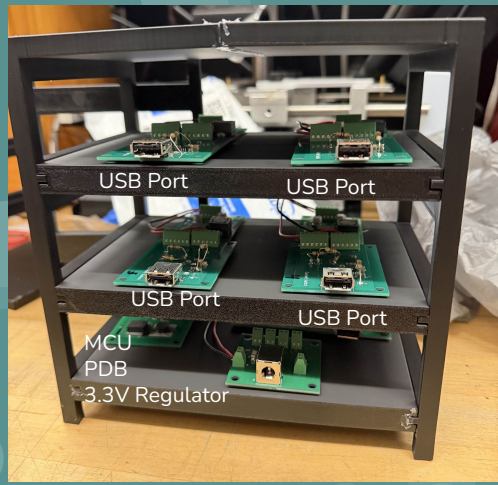
Properties:

- **USB-A:** There are four USB-A inputs for the user to connect their devices for charging.
- **Hardware Components:** Several hardware components allow for the charger module to work. We have 3.3V regulators, 5.25V regulators (behind each USB port) , current switches, the MCU, and the Power Distribution Board.
    - These components allow for smooth connection between the control and the charger modules.
    - The user is also able to monitor charging throughout the entirety of the charging





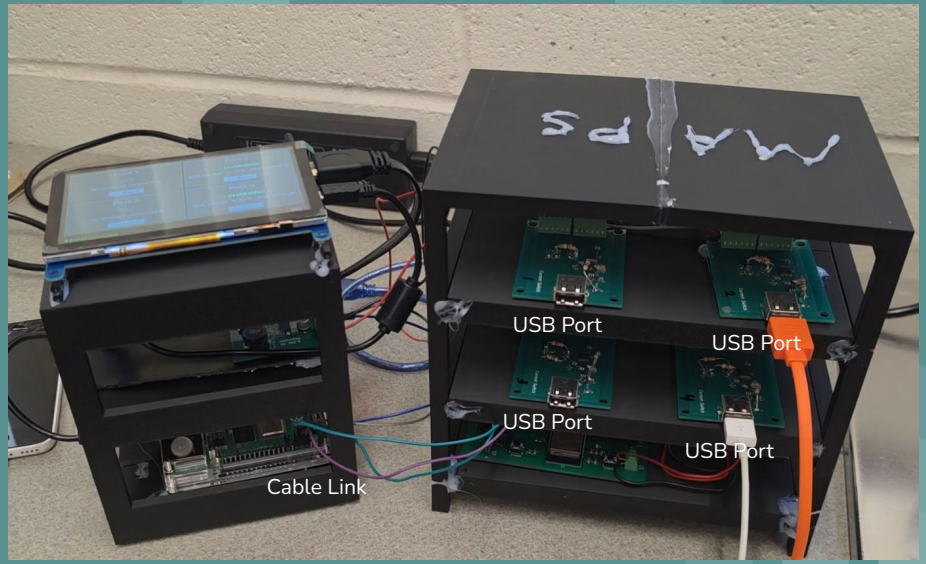
Charger Module CAD half (top) with shelf (left)



Charger Module with PCBs installed

Hardware Prototype

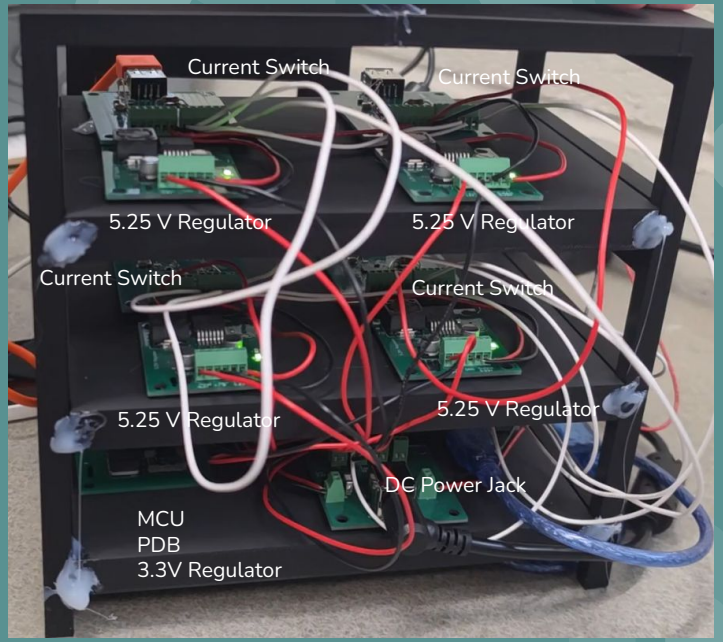
- **4 USB-A Ports:** There will be 4 ports on the charging module to allow for each of the devices connected to charge concurrently. There is also per-port control so this means, each of the ports can be handled independently as well.
- **Cable Link to Control Module:** Allows for communication with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



Isometric View of Charging Module (Front)

Hardware Prototype

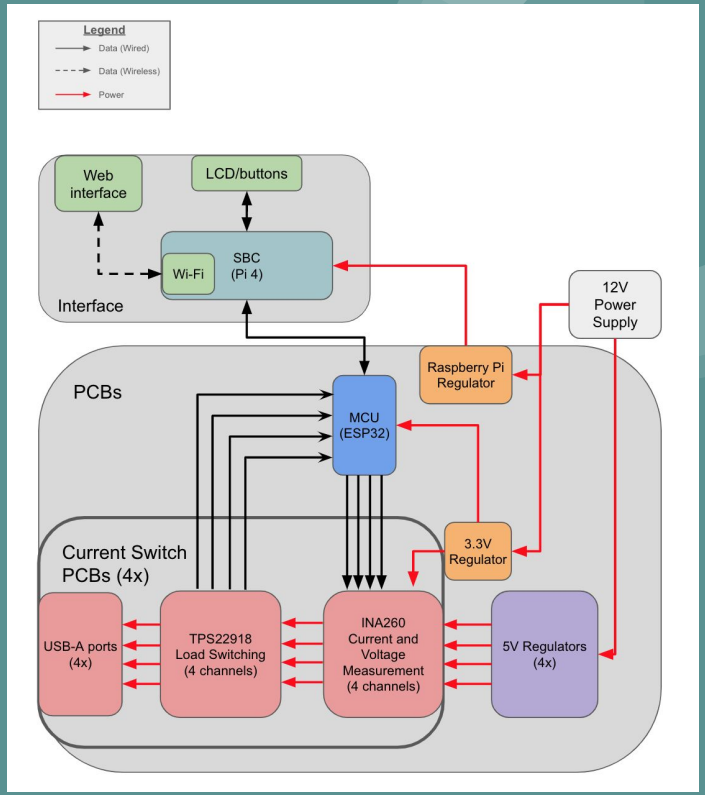
- **DC Power Jack**: For all ports to be supported, feeds power to support each of the ports via their regulators as well as the MCU.
- Current Switches (x4): Controls power to each individual USB port. Also allows for individual port switching and fault isolation.
- 5.25 V Regulators: Provides a stable 5.25V supply for the USB output ports.
- PDB: Distributes power to various parts within our system.
- MCU: Controls system's hardware as well as communication
- 3.3V Regulators: Provides a stable 3.3V supply for the lower voltage components



View of Charging Module (Back)

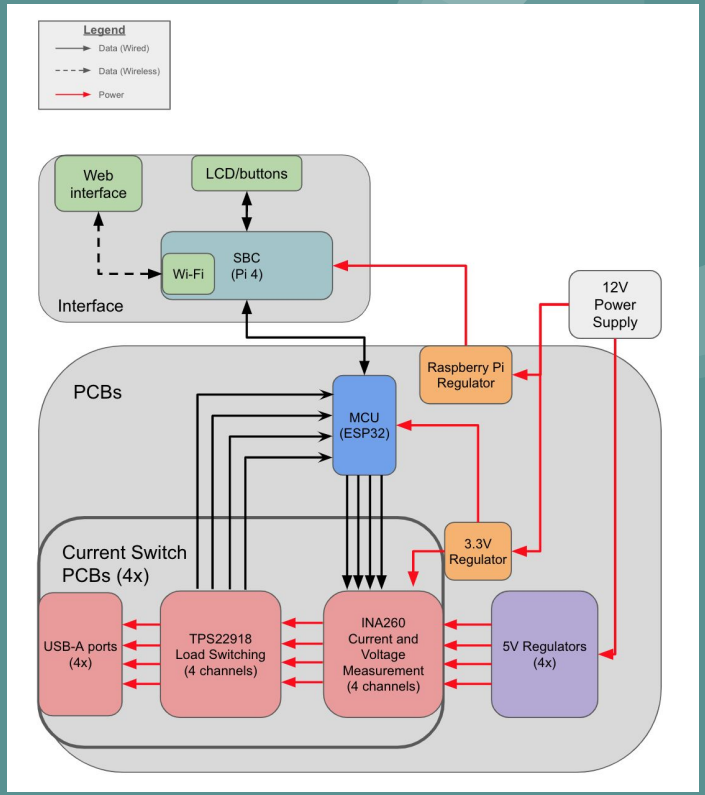
Hardware Design

- **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



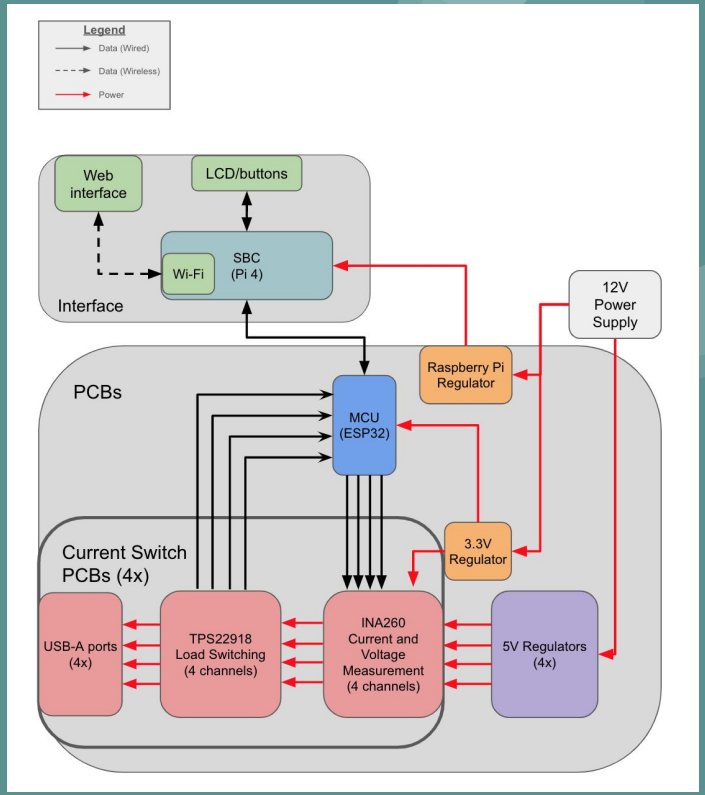
Hardware Design

- **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.
- **INA260 Current/Voltage Measurements:** Each port where a device is connected monitors and reports the current and voltage statuses.

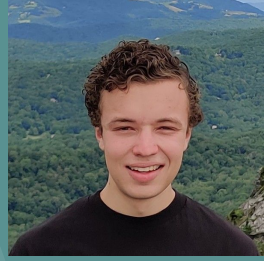


Hardware Design

- **Load 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.
- **Raspberry Pi Regulator** - Supplies power to the SBC



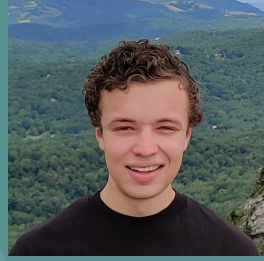
Comparison and Selection of Hardware: MCU



- 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 (26) 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 RAM) for our single-board computer.
- The Pi 4 is well documented, and it is powerful enough to host our local website and interface with significant overhead. Its power draw is still acceptable.
- 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/1 6GB
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

- LM 2670 chosen as 3.3V Regulator
  - Adjustable version offering fine tuning for output voltage
- Frequency < 1MHz for our configuration

	MP1584	**LM2596**	**AMSR-783.3-NZ**	**LM2670**
Brand	Monolithic Power Systems	Texas Instruments	Recom Power	Texas Instruments
Input Voltage Range	4.5 V - 28 V	4.5 V - 40 V	4.75V-28V	3.5 V - 36 V
Switching Frequency	100kHz-1.5MHz	150-kHz	330 kHZ	260kHz
Efficiency	~90%	94.6%	91%	85%

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
- Can be directly controlled by the MCU

	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

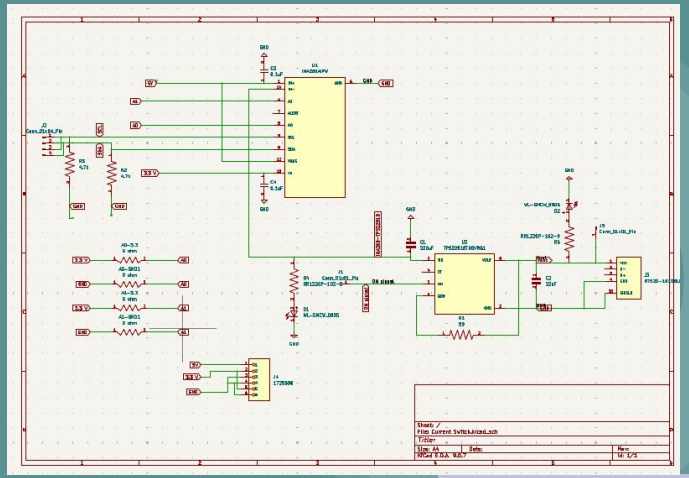
Comparison and Selection of Software

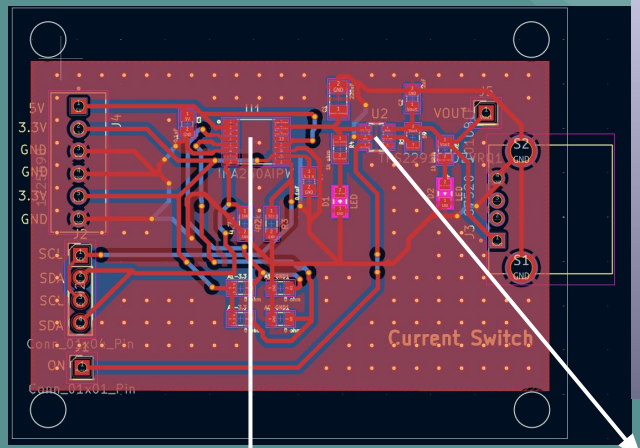
- The User Interface uses a modified traditional MERN stack.
- React enables efficient real time UI updates to reflect MCU controlled charging data
- Node.js with Express handles UART ingestion, WebSocket broadcasting, and REST API requests.
- SQLite on Raspberry Pi OS provides a lightweight, embedded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

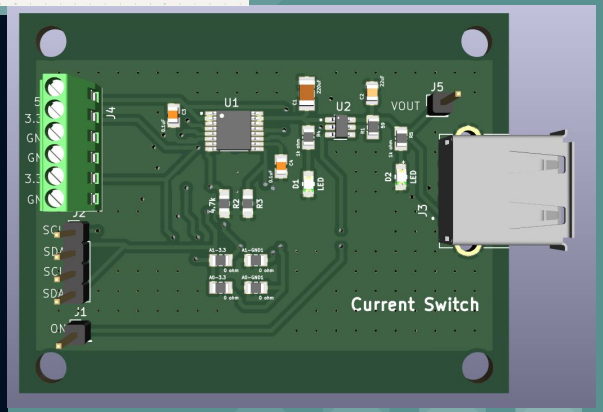
Web Stack	**Frontend**	**Backend**	**Database**	**Operating System**
LAMP	HTML/CSS/ JavaScript	PHP	MySQL	Linux
Modified MERN	React	Node/ Express	SQLite	Raspberry Pi OS

Significant PCB Design Current Switch

- INA260
  - Measure Current
  - Measure Voltage
  - Communicates to MCU with I2C
  - 0 ohm jumpers to determine I2C address
- TPS22918
  - Is either on or off through MCU connection
  - Capacitors values chosen to prevent input voltage from dipping
  - Added a bypass resistor post assembly





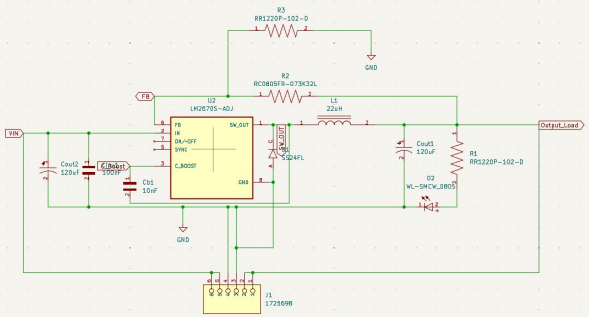


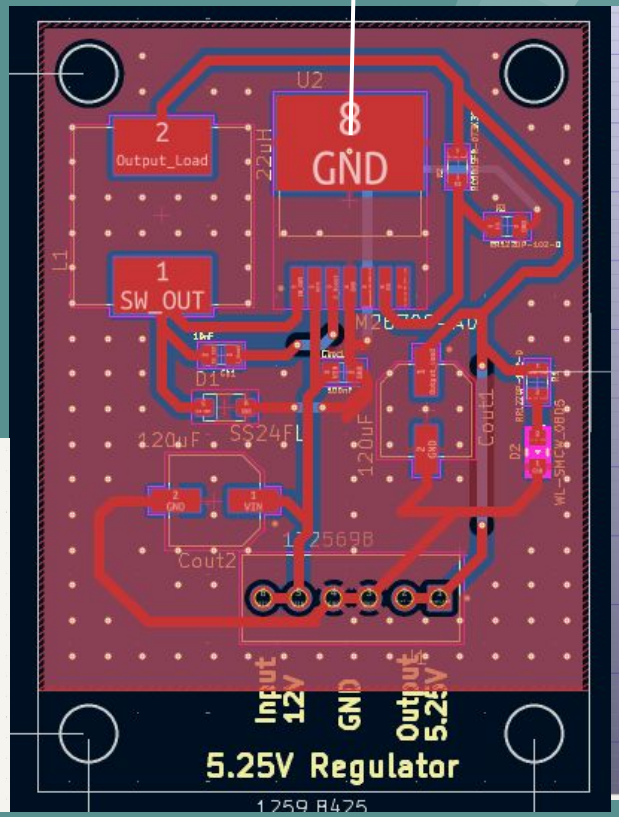
INA260

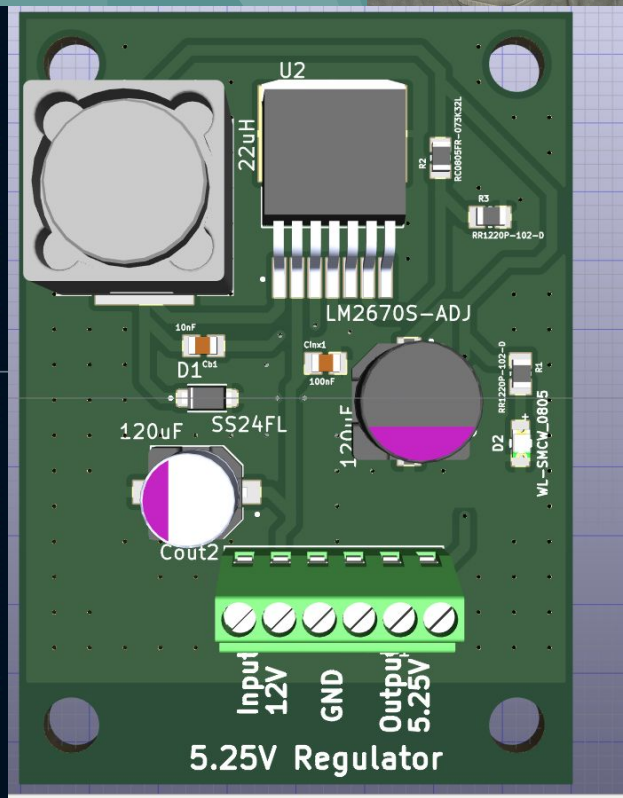
TPS22918

Significant PCB Design
5V Regulator

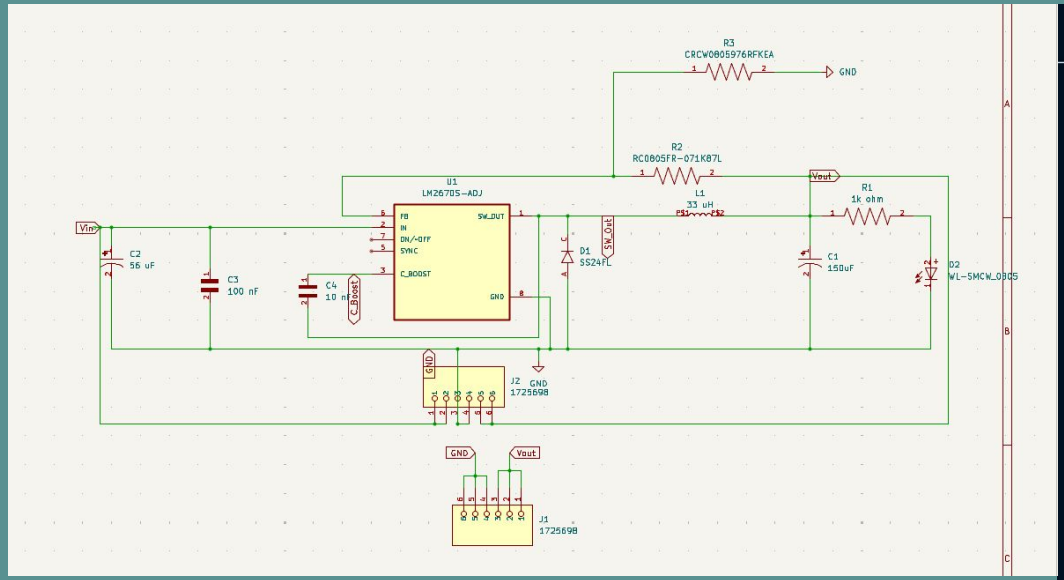
- 5V regulator with 1oz copper
- Trace widths 32 mil



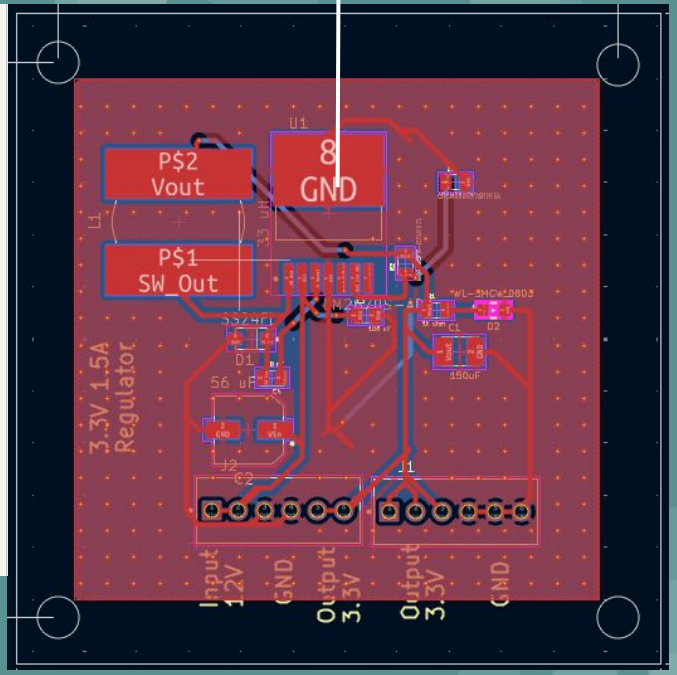




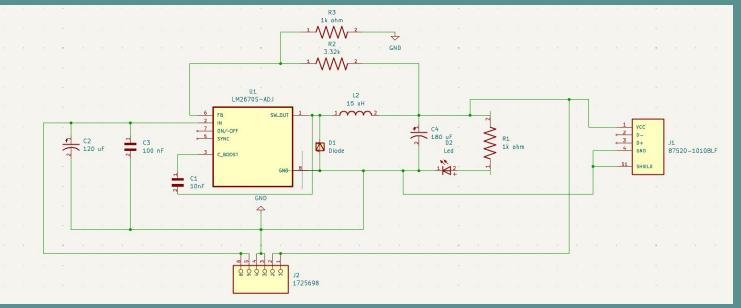
Significant PCB Design
3.3V Regulator

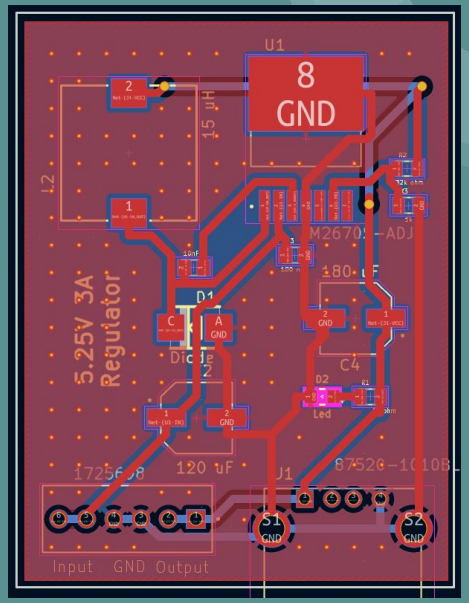


LM2670

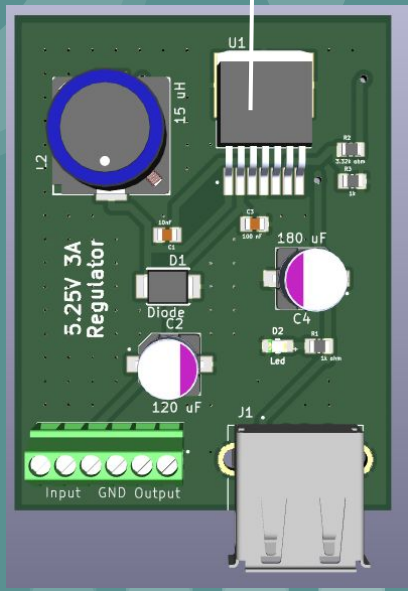


Significant PCB Design
Raspberry Pi Regulator

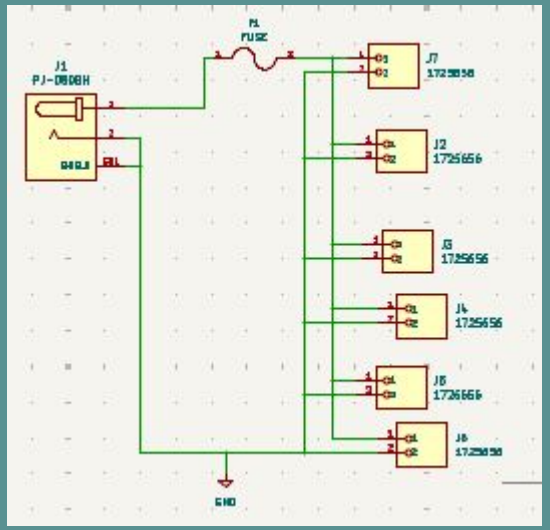


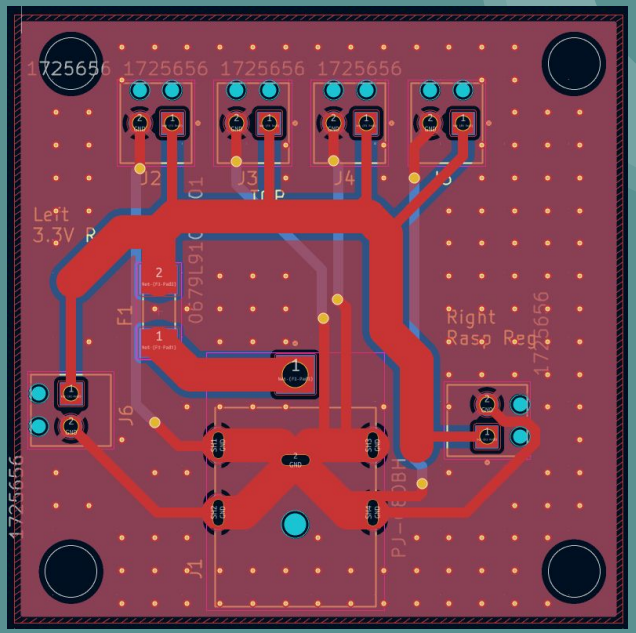


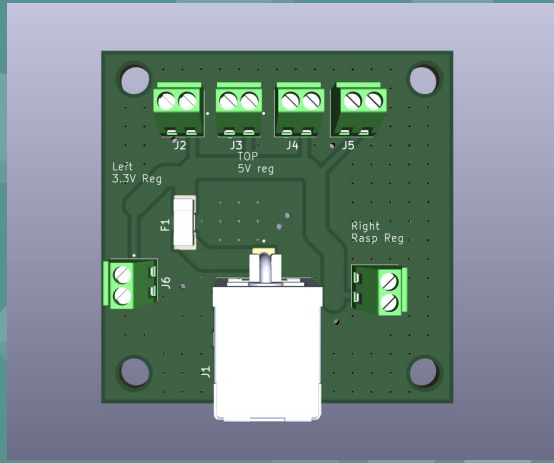
LM2670



Significant PCB Design
Power Distribution Board (PDB)







Power Distribution Table (Available)

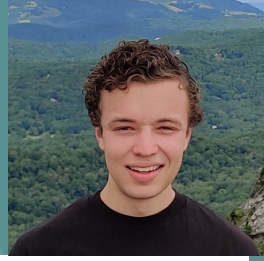
	Voltage	Max Current	Total Power
5V Regulators x 4	5.25 Volts per Reg	2 Amps per Reg	42 W
3.3V Regulators	3.5 Volts	1.5 Amps	5.25 W
Raspberry Pi Regulator	5.25 Volts	3 Amps	15.75 W
Total Power			63 W

Power Distribution Table (Consumed)

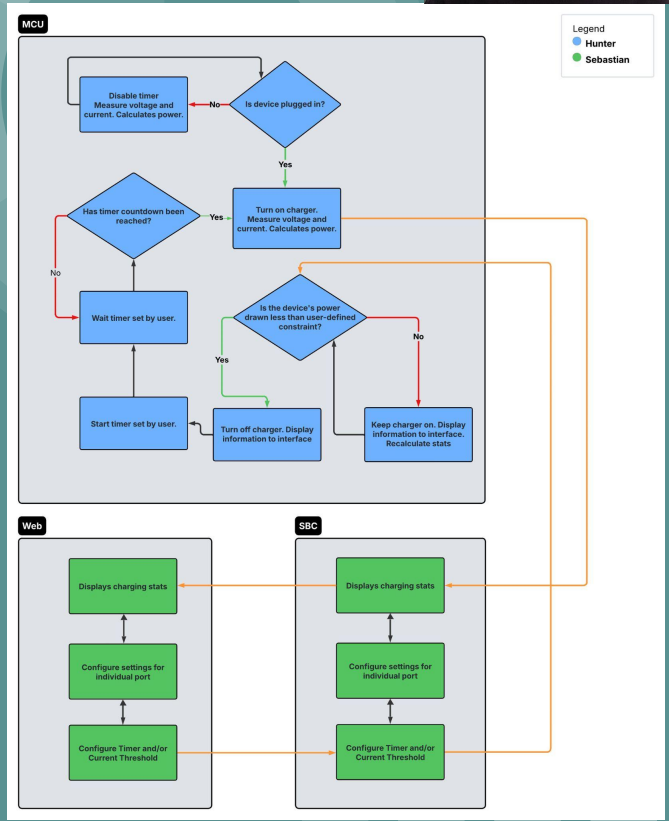
	Voltage	Max Current	Total Power
ESP32	3.3 Volts	260 mA	858 mW
INA 260 X 4	3.5 Volts	0.42 mA	5.376 mW
TPS22918 X 4	3.5 Volts	8.3 uA	116.2 uW
Raspberry Pi	5.25 Volts	3 A	15.75 W
Load Power X 4	5.25 Volts	2 A	42 W
Total Power			~59 W

- All Power calculation are assumed to be at Maximum delivery for Available Power and Maximum Load for Consumed Power
- We estimate to use 4W less than what our system could use
- Power delivery is sustainable during normal operation

Software Design: MCU

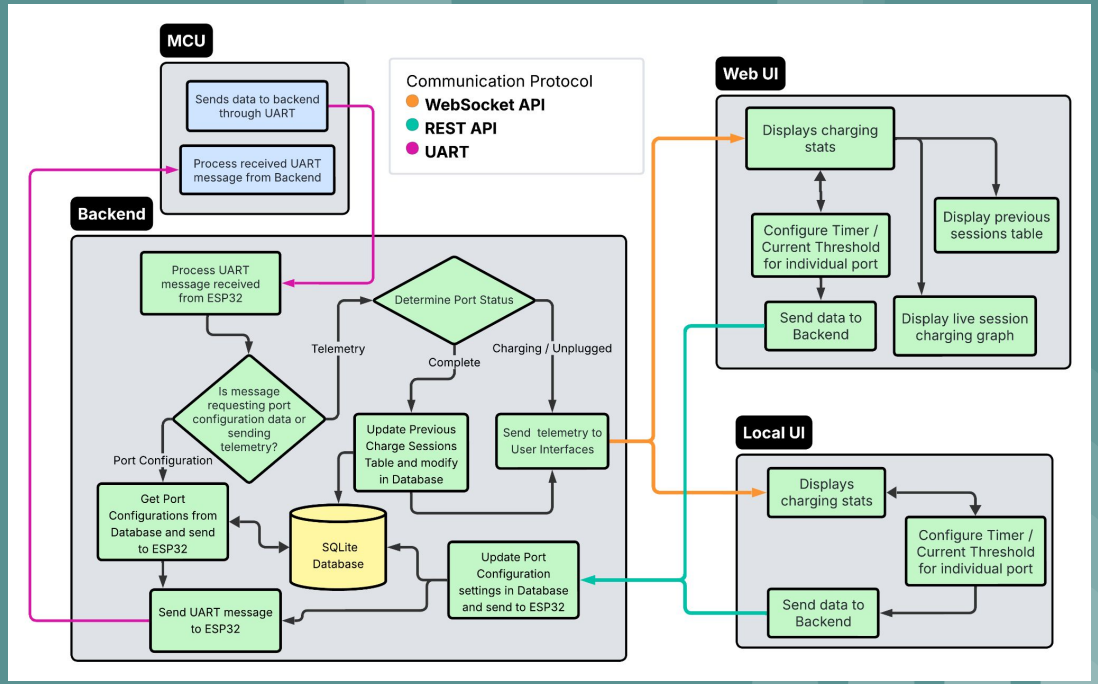


- We will be using FreeRTOS tasks and taking advantage of the ESP32's two cores.
- We will be programming the ESP32 in C++, but the code is also Arduino IDE compatible.
- Several periodic tasks exist in order to check charging stats, make decisions based on charging stats, switch ports on and off, detect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

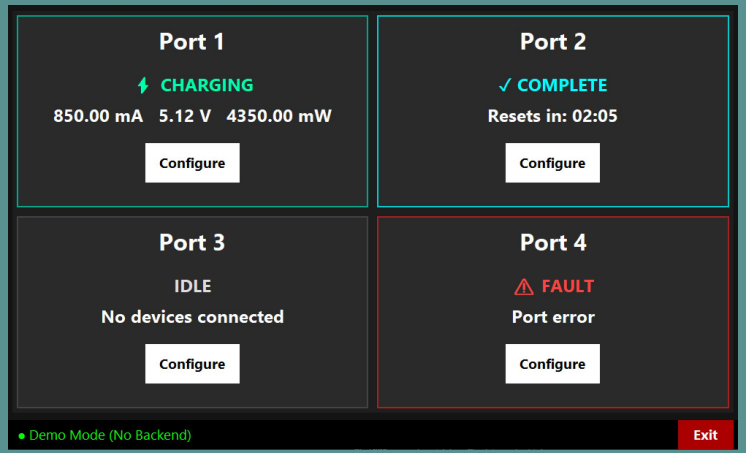
Home Screen

- Displays real time data from ESP32 to the user

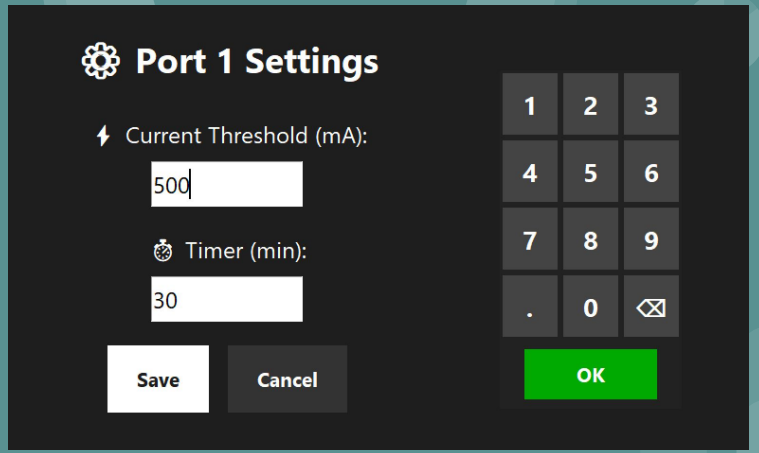
Configuration Menu

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

Local User Interface



Home Screen



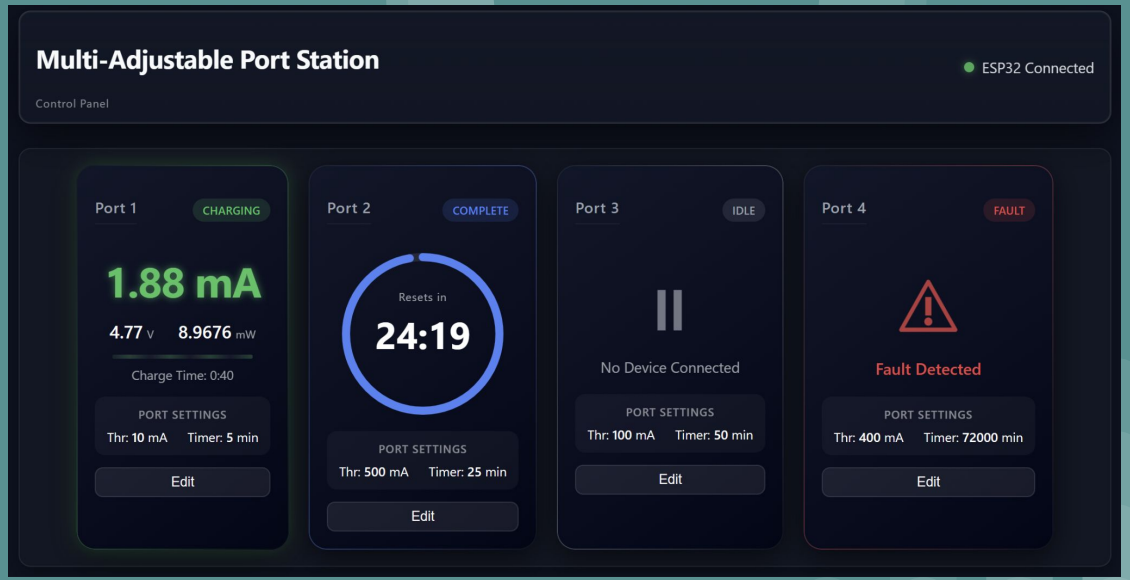
Configuration Menu

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



(MAPS Dashboard)

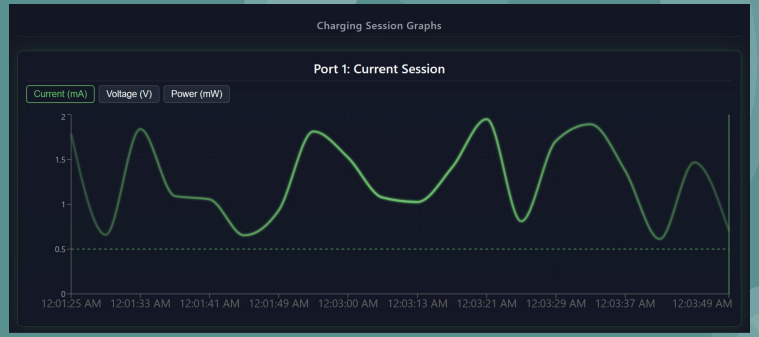
Website User Interface

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



(Live Charging Graph)

Port 1

START TIME	END TIME	CHARGE DURATION	AVG CURRENT	AVG VOLTAGE	AVG POWER	ENERGY DELIVERED	END REASON
4/3/2026, 12:54:11 PM	4/3/2026, 12:54:31 PM	0:20 min	952.06 mA	4.86 V	4728.42 mW	23.89 mWh	Reached threshold
4/2/2026, 7:45:45 PM	4/2/2026, 7:47:08 PM	1:23 min	782.48 mA	4.85 V	3828.66 mW	86.76 mWh	Reached threshold
4/2/2026, 7:42:29 PM	4/2/2026, 7:45:31 PM	3:01 min	4.33 mA	4.99 V	19.93 mW	0.79 mWh	User unplugged
4/2/2026, 6:57:06 PM	4/2/2026, 7:03:38 PM	6:31 min	172.05 mA	4.90 V	847.41 mW	91.90 mWh	Reached threshold
3/7/2026, 9:43:11 PM	3/7/2026, 9:44:51 PM	1:39 min	400.00 mA	0.00 V	2.00 mW	0.06 mWh	Reached threshold

(Session History Table)

Application Programming Interface (API)

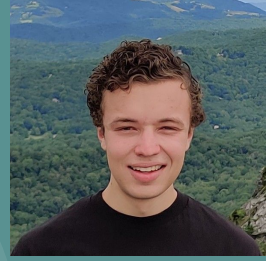
- The SBC hosts both REST and WebSocket APIs.
- REST API sends configuration data to the ESP32
- WebSocket API streams live telemetry from the ESP32 to the client.

	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

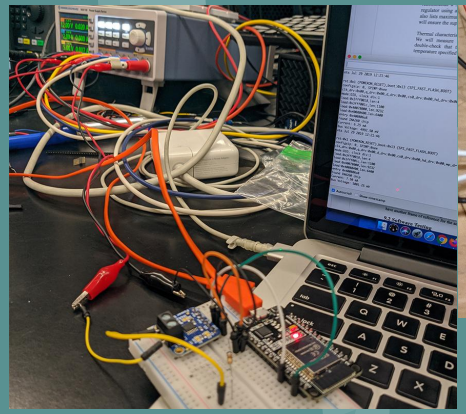
Prototyping and Testing (Hardware Testing)

- **Goal**: Confirm power regulation, USB control ports, voltage and current accuracy, and false detections .
- **Power rails**: Confirms 12V input, the regulated 5.25V 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.
- **Measurement accuracy:** Accuracy of the INA260 voltage/current/power readings using I2C. See how this compares against using a USB multimeter and DMM.
- **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.

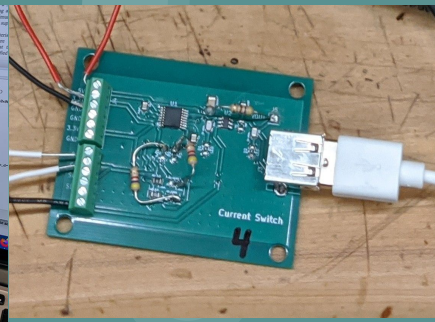
Hardware Testing (Current Measurement)



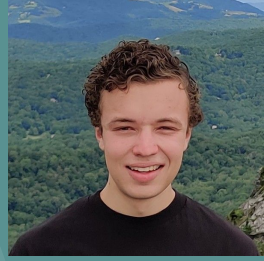
- Pre-made breakout boards utilizing the INA260 current measurement IC exist
- We can connect this breakout board to an ESP32 development board over I2C
- There, we can test our INA260 with code written for the ESP32, and a breadboard with a load resistor.
- Measuring the current and voltage with a DMM,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



Testing Current Measurement on Breadboard shown

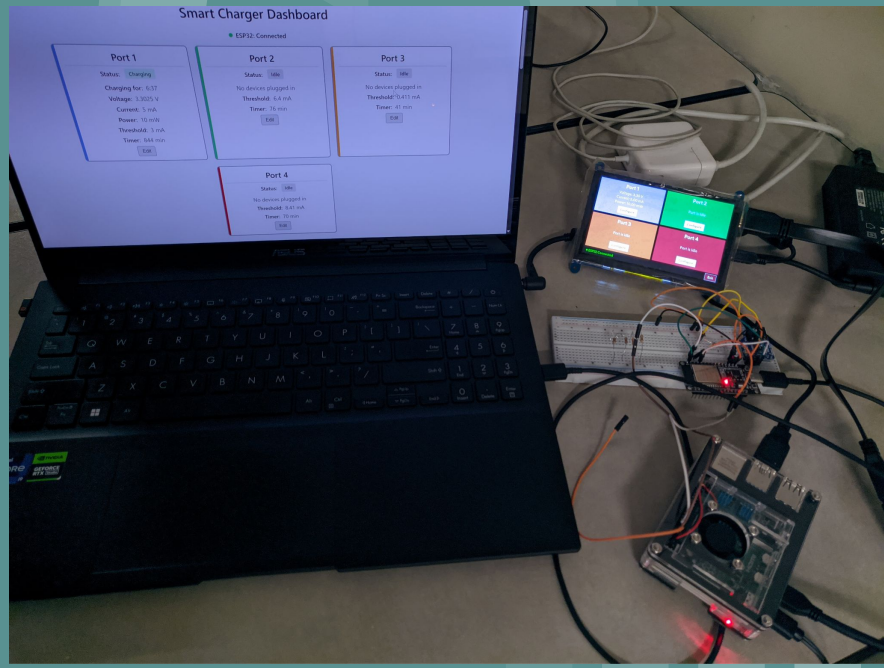


Testing Current Measurement on PCB shown

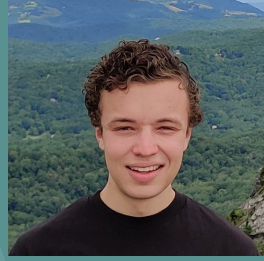


Prototyping and Testing (Software Testing)

- 2 parts of software testing: UI/SBC testing, and microcontroller testing
  - First Phase of UI/SBC: The UI and SBC interface were tested with an ESP32 generating simulated values.
  - First Phase of Microcontroller Testing: Test current measurement and simulated tasks
- After PCBs arrived and confirmed working, we continued software testing.
  - Second Phase of UI/SBC: Testing with synchronization between local UI/web UI. Revised to show charging states, limit edge values, and show values more clearly.
  - Second Phase of Microcontroller Testing: Device plugged in detection, Edge Case Testing with low (<10mA)/high (>1A) current and state transitions (charging, complete, unplugged). Test and revise charge timer functionality.



First Phase of UI/SBC and Microcontroller Testing shown



Project Challenges

- Technical Challenge: Device Plugged-In Detection
  - A significant hardware/software challenge, was detecting whether a device was plugged in while it is not charging.
  - Our eventual solution was to put a resistor between the input and output pins of the load switch, creating a small detectable current while the load switch was off.
  - This current isn't enough to charge a device, but is enough to detect.
- Scheduling Challenges
  - Since all of us have different schedules, we had to coordinate meeting times to work on the project.
  - We shared our schedules in a Discord channel and kept each other up-to-date when we needed to meet and when we were/weren't available.
  - We also tried to communicate key tasks effectively between all group members, and keep each other updated about progress.

Project Budget

Part	Description or P/N	Cost (incl tax/shipping)
PCBs (First Revision)	Ordered from JLCPCB	$129.35
PCBs (Second Revision)	Ordered from JLCPCB	$223.21
Components (First Revision)	Ordered from DigiKey	$469.60
Components (Second Revision)	Ordered from Mouser	$679.50
Raspberry Pi 4B & Accessories	Raspberry Pi 4 Model B (incl replacements)	$150.00
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
3D Printer Filament	SUNLU PLA 3D Filament 1.75mm	$30.08
Total spent:		$1,765.05

Work Distribution

Hunter Volonino	Project Lead	**Sebastian Sotomayor**	Local User Interface
	Microcontroller & Hardware Interface		Web User Interface
	PCB Assembly/Testing		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

Thank you!