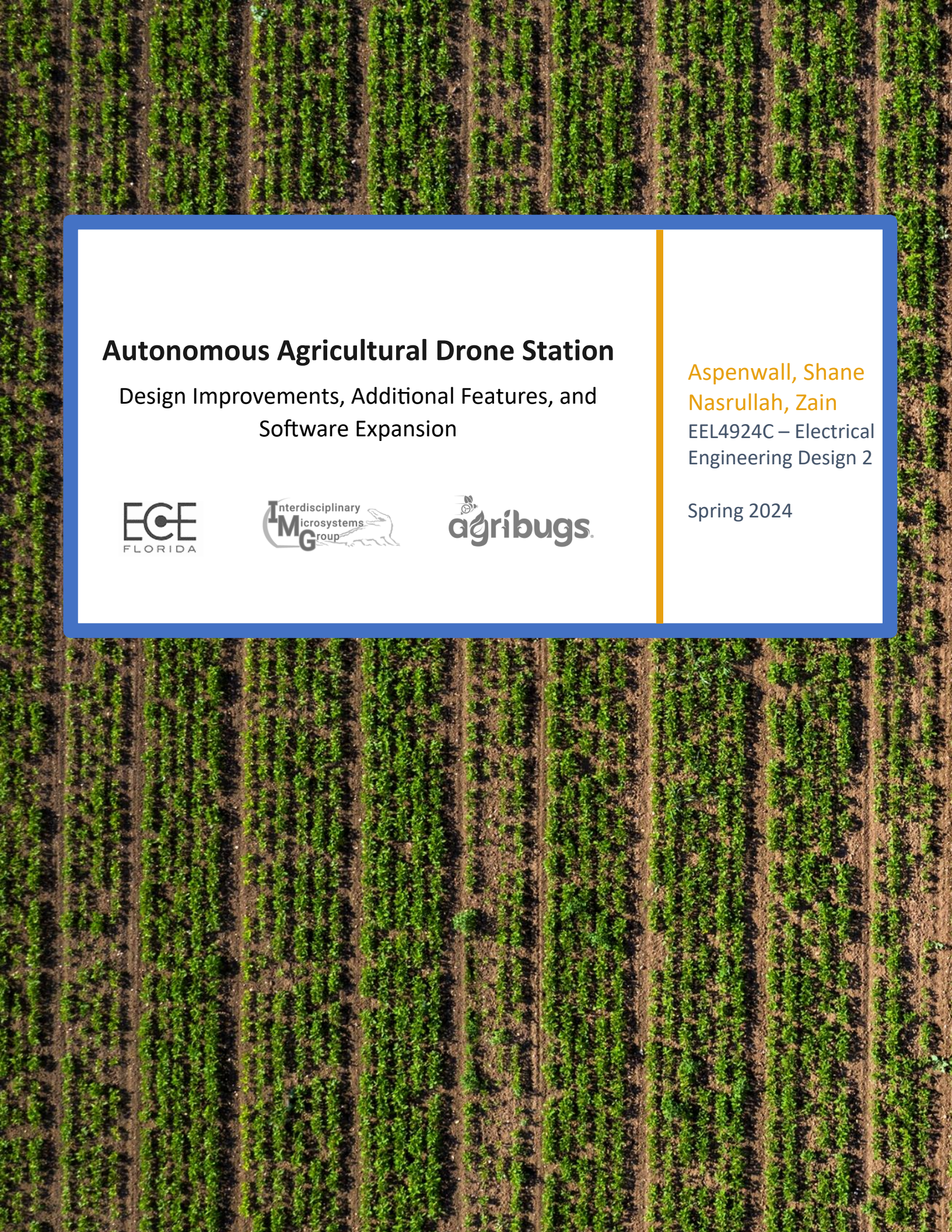


Autonomous Agricultural Drone Station

Design Improvements, Additional Features, and Software Expansion

Aspenwall, Shane
Nasrullah, Zain

EEL4924C – Electrical Engineering Design 2

Spring 2024

EE Design 2 Final Design Report

Autonomous Agricultural Drone Station

Team AgriGators

Contact Information

Shane Aspenwall
shane.aspenwall@ufl.edu
561-379-8234

Zain Nasrullah
z.nasrullah@ufl.edu
904-864-0570

Faculty Information

David Arnold
darnold@ufl.edu
352-392-4931

Dieter Steinhauser
dsteinhauser@ufl.edu
954-774-9034

3rd Party Sponsors & Affiliations

Agribugs LLC
Tallahassee, FL

Acknowledgements

We would like to extend our gratitude to the faculty, staff, and colleagues in the UF ECE department. A personal thank you to both Dieter Steinhauser for his analog circuits design knowledge and advice, as well as Dr. David Arnold for this research opportunity and financial support. Furthermore, the Lab Support offered by Eric Liebner and Michael Stapleton was paramount for ensuring the success of this project, especially with the SMD assembly of the new power board. Among others in the department who aren't directly mentioned, our appreciation cannot be overstated.

Contents

Abstract

One challenge plagued by humanity is the ability to provide sustenance to an ever-growing population. Operating costs on farms, including pest control, recovery from natural disasters, and use of fertilizers determine the gross output of a single farm, and trickle down to the market value of crops for entire regions. Decreasing operating costs, reducing resource consumption, and increasing crop output are all key to combatting food insecurity and improving the quality of living of people around the world. The goal of IoT4Ag is to facilitate the advancement and improvement of agriculture using Internet of Things infrastructure.

In partnership with Agribugs LLC, we are working to develop a complete agriculture management and monitoring ecosystem. This is a long-term research project aimed to design a dock station, autonomous aerial vehicle, and management computer that can effectively survey, monitor, and provide data to farmers to combat overfertilization, pests, and other adverse growing conditions that decrease the efficiency of a farm's crop output, as well as decrease negative effects to the environment. Actively, there is both a drone station and basic readout computer that reports data relevant to charging, system status, and weather conditions. The primary goal is to make the system easy to maintain, have maximum reliability in adverse outdoor environments, be intuitive to operate, and be as cheap as reasonable.

Introduction

The entire system must be capable of autonomous self-sufficiency, including protecting the drone from outside elements when docked. The main purpose of the autonomous drone station is to take in weather data, log time and date data, conserve power when idle, and be able to discern appropriate times to deploy automatically to take pictures of crop fields. Once docked, the drone will offload taken pictures to an on-site computer running a full OS that can use machine learning, neural networks, and AI to decide if any pictures show sub-optimal crop activity such as blight, parasites, malnutrition, physical damage, or crowding. Pictures will ideally by geolocated to create an interactive heatmap that can be used to identify trouble areas in a field that may require human intervention.

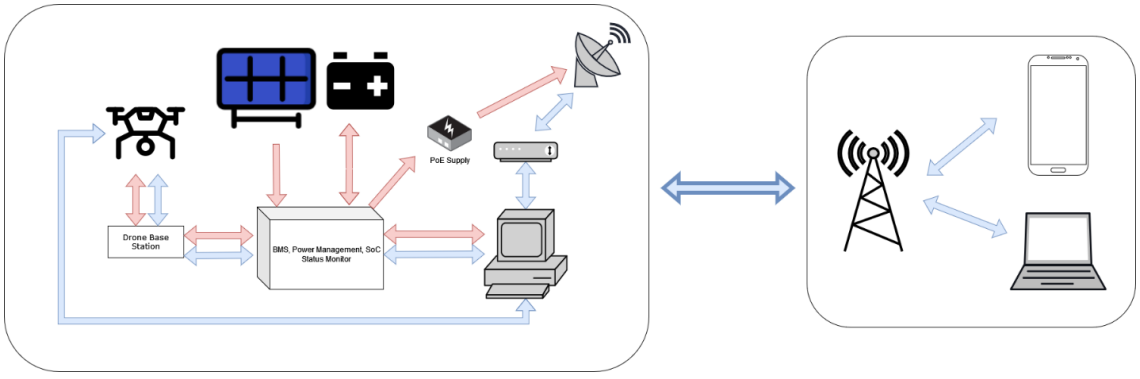


Figure 1: Primary drone system diagram showing how dry-docking can be achieved with on-site charging and satellite internet connection. 5G and 5GE (4G LTE) reception is not expected in a real-world environment.

Preliminary Methods

I. Design Goals

Listed are the primary goals for the final deployment. These are targets that may or may not be addressed in the allotted time for the semester:

- Drone Station
    - Fully autonomous clamshell protection area for primary drone charging pad.
    - Fully integrated BMS board and PC
    - Compatible with large-scale drones
    - Extreme weather resistance and durability in adverse outdoor conditions
- Drone
    - Fully autonomous operation, including image capture, “Safe to deploy” decision making, GPS/RTK data logging, ICE image stitching, flight path logging.
    - Remote-control override by operator
    - Full custom flight controller (no proprietary systems)
    - Lightweight
    - High endurance
    - High quality camera and gimbal setup for mid-flight macro recording
- Monitor Station
    - Full diagnostic mode for technician servicing
    - Intuitive and foolproof status screen with live monitoring
    - Weather pattern and environmental data recording
    - Satellite data transmission over Starlink

II. Features

The features proposed for the final design have not been finalized yet. However, many of the proposed device goals are set to be confirmed features.

III. Expectations

With the time set out to complete this, the primary goals are as follows:

- Create a new charger board revision, moving off the TI EVM modules, fix design errors, and migrate to smaller and more robust SMD components.
- Adapt Starlink router PoE injector to accept 48V supply from main BMS board.
- Continuing for Starlink, a 12VDC/48VDC 2 Amp Boost converter for Starlink support will need to be added. The rail should be able to supply these 2 amps.
- Revise desktop software for improved readability and features.
- Design new drone charging pad capable of detecting drone presence and opening/closing protective cover.
- Create and improve charging circuit that drone will use when landing on the station's landing pad. This circuitry onboard the drone will accept the voltage/current from the landing pad and safely supply it to the battery, keeping in mind short circuit, overvoltage, and overcurrent protection.

IV. Timeline

Many of the main tasks do rely heavily on parts sourcing and overseas manufacturing, dramatically affecting deadlines and timetables. The primary focus is to prioritize long-waiting tasks for now, then change to other less time-intensive tasks while primary tasks are in the post. Turnaround for PCB and parts orders will be 1.5 weeks. The goal is to order the MK1.1 board, PoE injector, ordering supporting parts, then moving to GUI/software work, as well as the 3D prototype drone door.

V. Cost

Cost is a major concern for the final product. As design progresses, shifting away from bulkier THT components to the more efficient and cheaper SMD parts, overall cost of the mainboard will dramatically decrease.

Below is an itemized BOM from the IoT4Ag autonomous drone project from a prior semester, with Dieter Steinhauser and Ryan Laur leading the project.

Name	Description	Quantity	Price	Component Total	Link
Beelink Series 5		1	$ 350.00	$ 350.00	https://a.co/d/2AIAyE9
SSD 1 TB	Solid State Storage Drive	1	$ 100.00	$ 100.00	https://a.co/d/5wTkXSJ
Solar Panel	50W Solar Panel	3	$ 50.00	$ 150.00	https://a.co/d/hoDinJv
BBQ24650 EVM-639	MPPT Charge Controller	1	$ 100.00	$ 100.00	https://www.ti.com/tool/BQ24650EVM-639
Router	Local Network Router	1	$ 40.00	$ 40.00	https://a.co/d/3fTL7Gd
Raspberry Pico	Raspberry Pico uP	1	$ 4.00	$ 4.00	https://a.co/d/1YXqlHi
Battery	Lithium Iron Phosphate Battery 50Ah+	1	$ 200.00	$ 200.00	https://a.co/d/ftXwLlj
Drone	3DR IRIS Drone	1	$ 649.00	$ 649.00	
Starlink	Satellite router	1	$ 600.00	$ 600.00	
PixHawk	PixHawk flight controller	1	$ 300.00	$ 300.00	
RFD900x	900 MHz Radio Frequency Modem	1	$ 200.00	$ 200.00	https://irlock.com/products/rfd900-telemetry-bundle
4G Modem	NETGEAR 4G LTE Broadband Modem	1	$ 140.00	$ 140.00	https://a.co/d/25gcTCD
RTK GPS	RTK GPS Receiver	1	$ 290.00	$ 290.00	https://www.sparkfun.com/products/22660
TPS552892 EVM	36V, 8A Buck Boost Eval module	3	100	$ 300.00	TPS552892EVM Evaluation board | TI.com
TPS51387A EVM	10A - Buck regulator Eval module	1	$ 50.00	$ 50.00	https://www.ti.com/tool/TPS51397AEVM
GPS Antenna		1	$ 140.00	$ 140.00	https://www.sparkfun.com/products/17751

Figure 2: BOM for autonomous drone allocated in Fall 2023

Source: IoT4Ag (Dieter Steinhauser and Ryan Laur)

VI. Methodology

Long-term reliability and a focus on using non-proprietary technologies as the main form of system control are of utmost importance. It would not be unheard of to think that an autonomous farm monitor system could be in use for a long time, completely or seldom maintained. More stable battery chemistries such as LiFePo4 have been seen to sustain much more usable life for longer as compared to Li-Ion in deep discharge scenarios.

Cost is another key factor to consider here. Agriculture is a highly cost-competitive market, so any product meant to sell in this sector must offer significant and obvious value to be successful. We would also place care to make sure the GUI is as simple as possible and abstracts away as many of the workings of our system as possible, and just give the users the bare minimum information they need to give a simple yet elegant product.

VII. End User

Two clients are to be considered when deciding features: farmers in the private sector, as well as researchers in the public sector.

Farmers and other crop cultivators such as in nurseries will want a reliable, cheap, and simple to operate ecosystem with major focus on convenience.

Researchers will be more focused on high detail and volume of data. An ability to mount specialized test equipment would be of high priority as well.

VIII. Gantt Chart

The following is a Gantt Chart highlighting the key deadline markers for the primary problems we will be tackling:

Figure 3: Gantt Chart

Goals

This report will heavily rely on the reader's background knowledge of Dieter Steinhauser's and Ryan Laur's Fall 2023 Senior Design Report. Design decisions, improvements, errors, and improvements are enumerated in this document:

"EE Design 2 Design Report - Title: Autonomous Agricultural Drone Station - Drone Maestros"

I. Hardware

In the initial design, proof of concept and speed of testing were considered high priority. The original V1.0 board (retroactively codenamed 'Apis'), was to show proof-of-concept, as well as allow for full modular customization. Many of the features of the next phase were yet to be realized until a preliminary board was tested. Basic functionality needed to be proven before more design choices were considered. Dieter Steinhauser, the designer of the V1.0 board, impressively built much of the primary functionality from scratch.

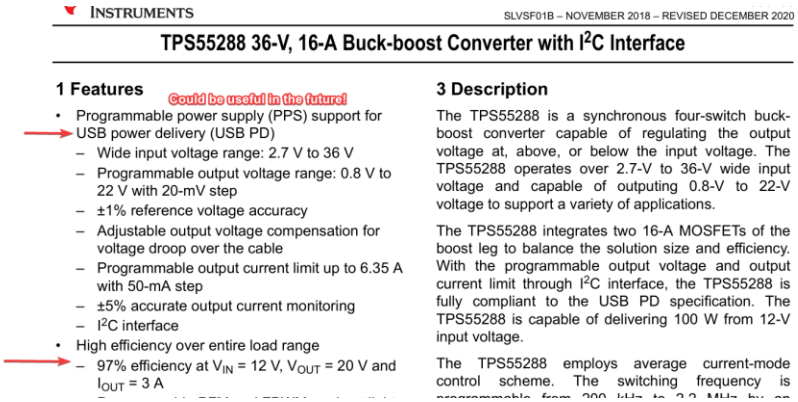
a. Power Supplies

The specific choices of boost/buck modules were made to offer maximum flexibility. The V1.0 could power the 19V DC supply of the BeeLink Small-Form-Factor PC and other support devices. On the V1.1, I made no attempt to replace the modules, as they proved to offer outstanding performance at a low cost and size. The parts selected were excellent for the application.

Power Rail - Application	Primary IC	Voltage Supplied	Cost / Module
20V – BeeLink PC	TPS55288	19V	$6.06
12V - TBD	TPS55288	12V	$6.06
ADJ – Dome Mech/ Drone Charging	TPS55288	10.7V-15V	$6.06
5V – Onboard MCU	TPS51397	5V	$1.30
ADJ – Solar Charge In	BQ24650	Variable	$5.93

Figure 4: Power Supply Sheet, describing the service and cost of each rail.

One thing worth mentioning with the I2C supplies is the compatibility with USB-PD, a power supply standard introduced with USB-C. If we decide to control the drone with an ARM SBC, such as a Raspberry Pi 5, or Jetson Orin, this would introduce hardware AI and Machine Learning capabilities as well as decreased power consumption in comparison to x86-based machines such as BeeLink.


INSTRUMENTS
SLVSF01B – NOVEMBER 2018 – REVISED DECEMBER 2020

TPS55288 36-V, 16-A Buck-boost Converter with I²C Interface

1 Features

Could be useful in the future!

- Programmable power supply (PPS) support for USB power delivery (USB PD)
  - Wide input voltage range: 2.7 V to 36 V
  - Programmable output voltage range: 0.8 V to 22 V with 20-mV step
  - ±1% reference voltage accuracy
  - Adjustable output voltage compensation for voltage droop over the cable
  - Programmable output current limit up to 6.35 A with 50-mA step
  - ±5% accurate output current monitoring
  - I²C interface
- High efficiency over entire load range
  - 97% efficiency at V_{IN} = 12 V, V_{OUT} = 20 V and I_{OUT} = 3 A

3 Description

The TPS55288 is a synchronous four-switch buck-boost converter capable of regulating the output voltage at, above, or below the input voltage. The TPS55288 operates over 2.7-V to 36-V wide input voltage and capable of outputing 0.8-V to 22-V voltage to support a variety of applications.

The TPS55288 integrates two 16-A MOSFETs of the boost leg to balance the solution size and efficiency. With the programmable output voltage and output current limit through I²C interface, the TPS55288 is fully compliant to the USB PD specification. The TPS55288 is capable of delivering 100 W from 12-V input voltage.

The TPS55288 employs average current-mode control scheme. The switching frequency is

Figure 5: The TPS55288 power converters feature excellent efficiency and 3A current capability.

One of the main goals of the V1.1 board is to remove reliance on TI EVM modules for both cost and size reasons. The board can also be rearranged to decrease wire lengths, improve organization, and shrink area. Decreasing board size matters for adding features, and reducing stress on the PCB.

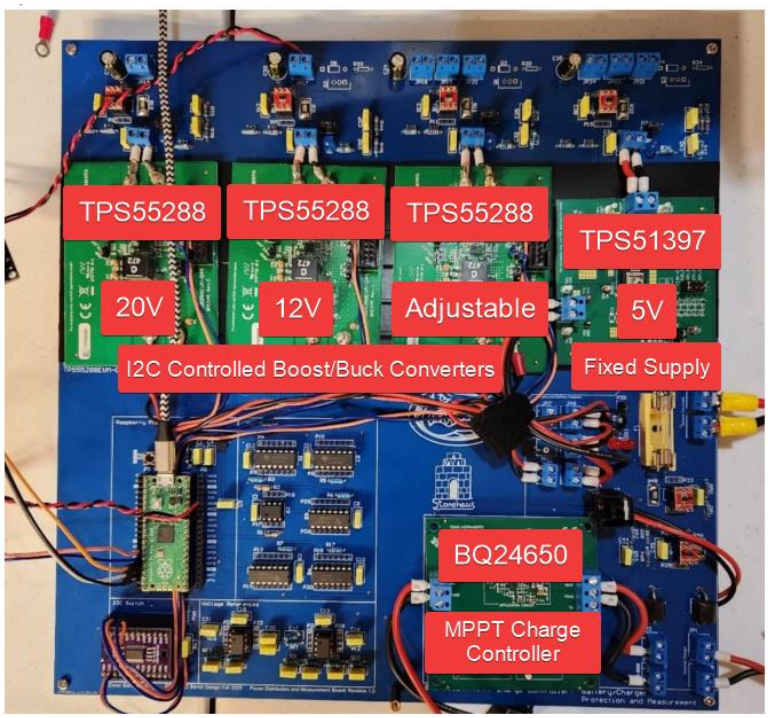


Figure 6: Drone station BMS board V1.0. Labelled are the main supply ICs selected for the design.

b. Supporting ICs

Many of the thru-hole ICs, such as the analog muxing network, reference voltage supplies, and IC-mounted current shunts, can be migrated over to SMD parts for the V1.1 board. The benefit here is 3-fold: smaller footprint, lower parasitic R,C,L values, and lower price.

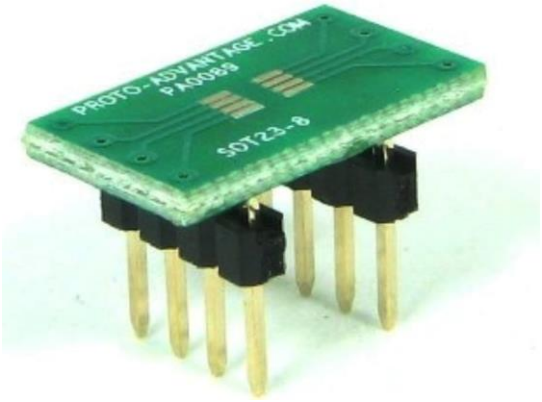


Figure 7: SOT23 to 8-DIP Adapter for example of size comparison. Currently, the INA181 current sense amplifiers are mounted in these. Dramatic size reductions can occur when we remove THT.

c. ADC Mux

One issue that was brought up by the V1.0 board was the introduction of bizarre behavior at the ADC inputs on the Pi Pico. Primary issues with the Pico design still exist, such as the SMPS power supply being tied to the ADC VREF introducing noise to the analog read circuit.

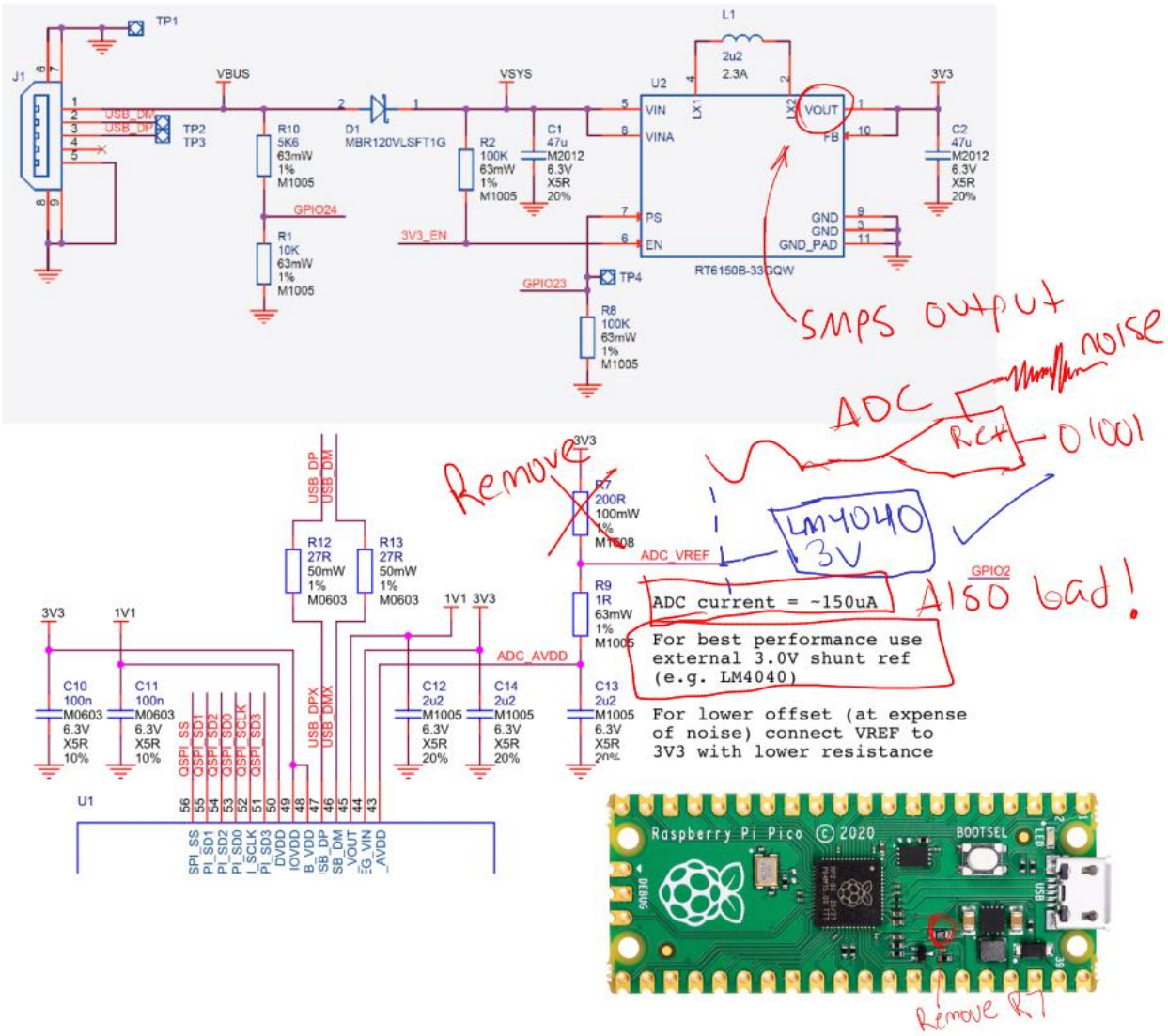


Figure 8: Hardware modifications to Pi Pico due to powerchain design. Using a LM4040 removes the GND offset and SMPS noise from analog inputs.

This issue is circumvented with the LM4040 shunt reference being placed at the ADC VREF pin after removing R7 from the Pico. However, issue number 2 is this ADC current metric of ~150uA. Depending on the source of a voltage at the ADC input, the input current can cause the voltage to drop below expected. Small signals are not expected to supply current, and as such, high impedance outputs can and WILL be affected.

This issue in fact came into play for another class where Pi Picos are used. In this situation, the output of a high-R divider was causing the voltage to sag at different voltages.

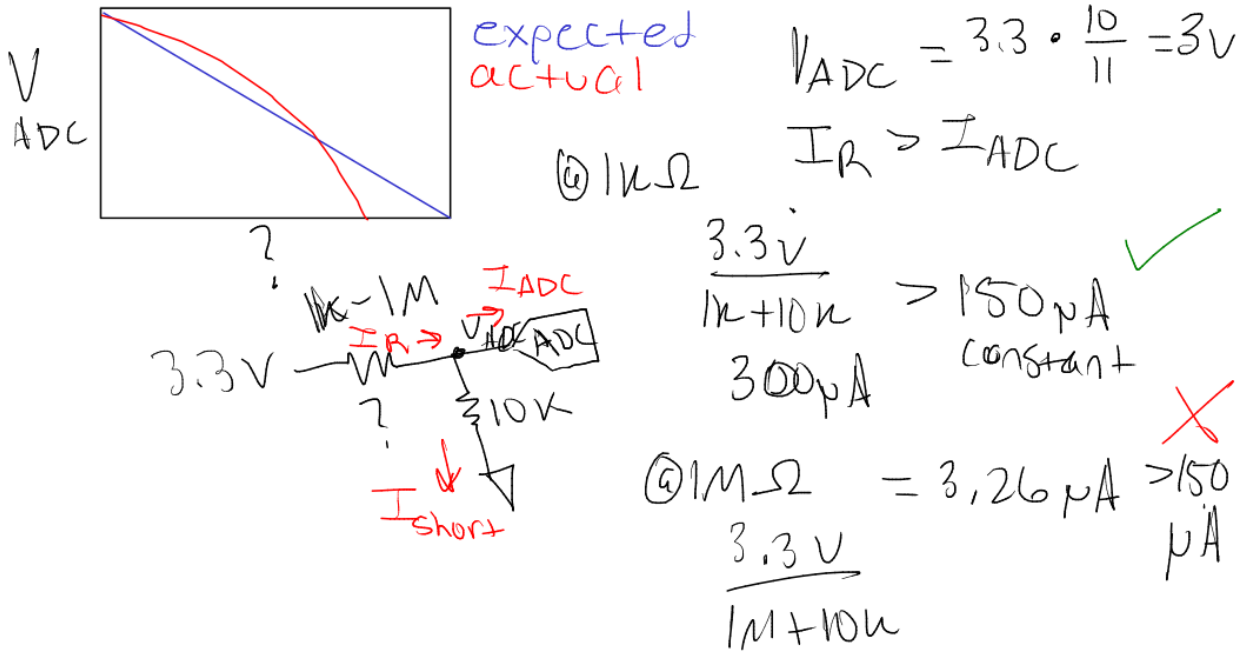


Figure 9: Graphed ADC loading on analog input in basic resistor divider example. If unable to supply 150uA to the input of the ADC, the ADC will create a further voltage division. This logic may be flawed, but generally gets the idea across.

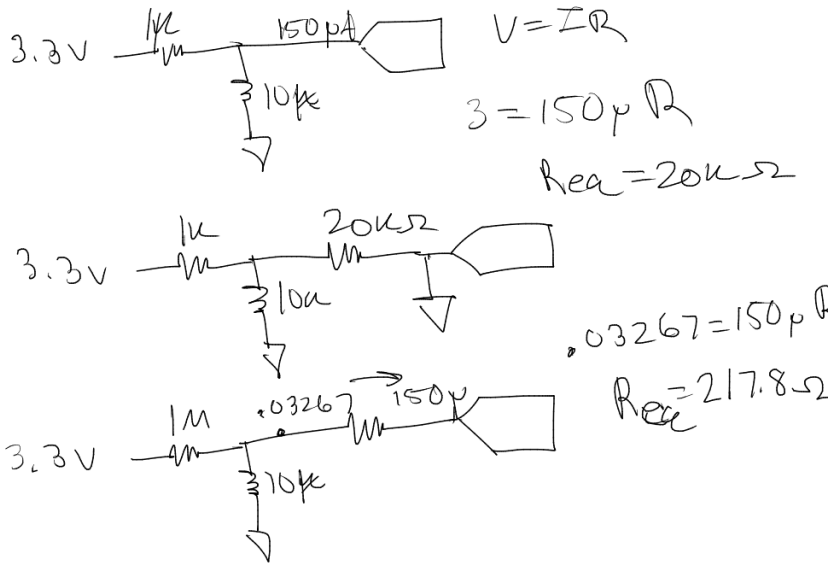
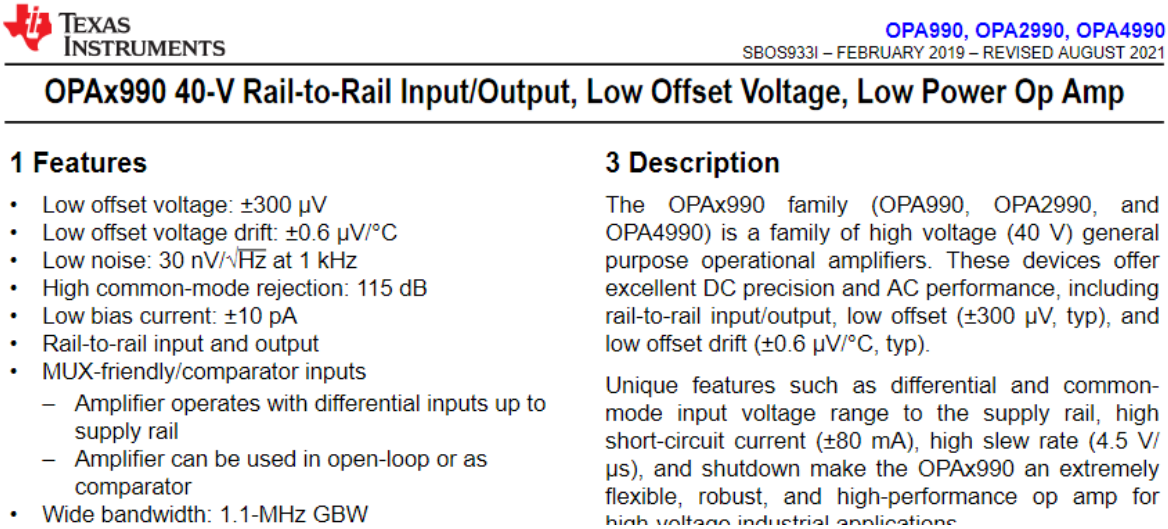


Figure 10: Further elaboration: the Req of the ADC changes when input current to ADC goes below needed 150uA. Req has the other side grounded, so 20k won't affect the "seen" voltage as much as a 200ohm in the same situation.

The solution is to supply a voltage follower from the output of the ADC mux network to the ADC inputs on the PICO. The idea is to have a low-impedance output piped directly into the current-consuming input. This way, the voltage drop is negligible. In fact, TI sells a voltage follower IC optimized to be used in Analog systems that utilize analog muxes.



TEXAS INSTRUMENTS

OPA990, OPA2990, OPA4990
SBOS933I – FEBRUARY 2019 – REVISED AUGUST 2021

OPAx990 40-V Rail-to-Rail Input/Output, Low Offset Voltage, Low Power Op Amp

1 Features

- Low offset voltage: ±300 µV
- Low offset voltage drift: ±0.6 µV/°C
- Low noise: 30 nV/√Hz at 1 kHz
- High common-mode rejection: 115 dB
- Low bias current: ±10 pA
- Rail-to-rail input and output
- MUX-friendly/comparator inputs
  - Amplifier operates with differential inputs up to supply rail
  - Amplifier can be used in open-loop or as comparator
- Wide bandwidth: 1.1-MHz GBW

3 Description

The OPAx990 family (OPA990, OPA2990, and OPA4990) is a family of high voltage (40 V) general purpose operational amplifiers. These devices offer excellent DC precision and AC performance, including rail-to-rail input/output, low offset (±300 µV, typ), and low offset drift (±0.6 µV/°C, typ).

Unique features such as differential and common-mode input voltage range to the supply rail, high short-circuit current (±80 mA), high slew rate (4.5 V/µs), and shutdown make the OPAx990 an extremely flexible, robust, and high-performance op amp for high voltage industrial applications

Figure 11: OPA4990 Analog Mux-Optimized Voltage Buffer

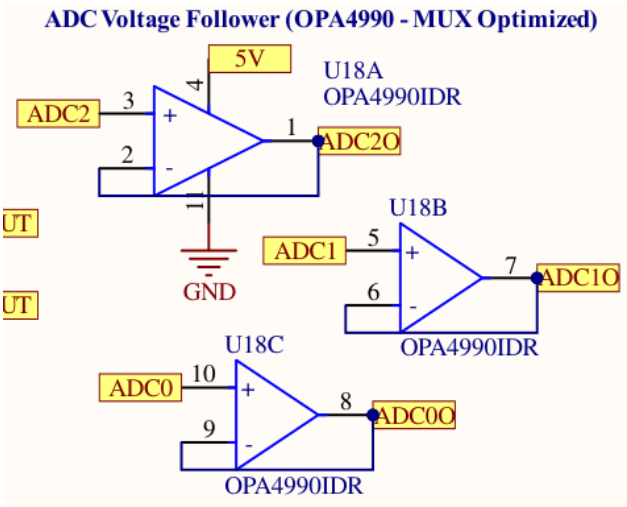


Figure 12: OPA4990 Used as non-inverting voltage follower. It would have been wise to tie the V+ rail to the 3V reference to prevent damage to the ADC but should be fine.

d. Stepper Motor Drone Dome Driver

For the drone design, a protection dome would be idea in order to protect the drone when charging from adverse weather, vandalism, and other physical intrusions.

Figure 13: Prototype drone station dome in open position. Drone is ready to launch.

Stepper motors were chosen for a few reasons:

- Low cost
- High Torque
- Precise positioning
- Easy to drive.

Figure 14: Mineabea NEMA 17 Stepper Motor with a 10.4V operating voltage. This exact motor is currently on the new drone station board. Source: https://www.mpja.com/NEMA-17-Stepper-Motor-104V-with-Square-Shaft/productinfo/33133+MS/

When compared to DC motors, it's much easier to guess the status of the dome and make precise controls and motions. Moreover, stepper motors offer freewheeling and locking capabilities depending on how they're driven. 1 full rotation of a stepper motor is 200 steps.

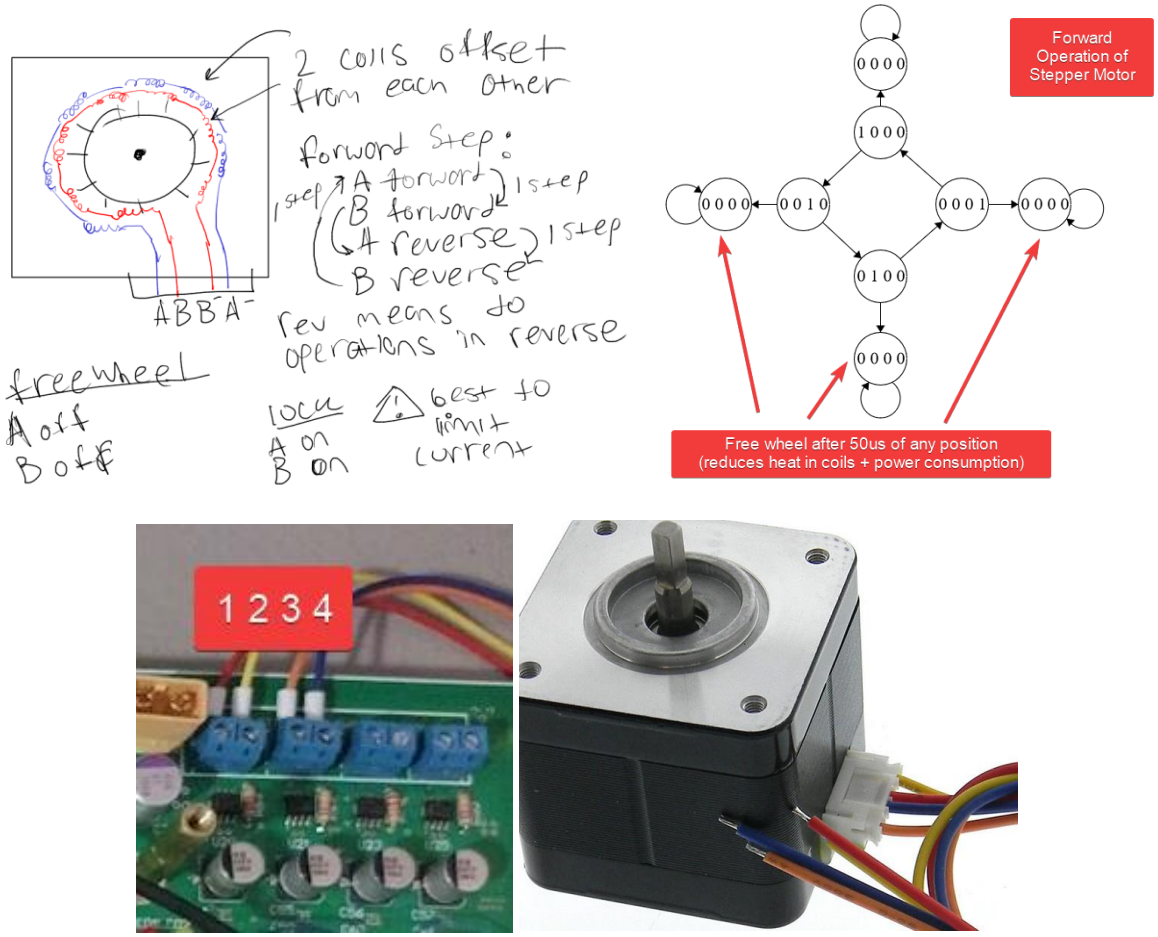
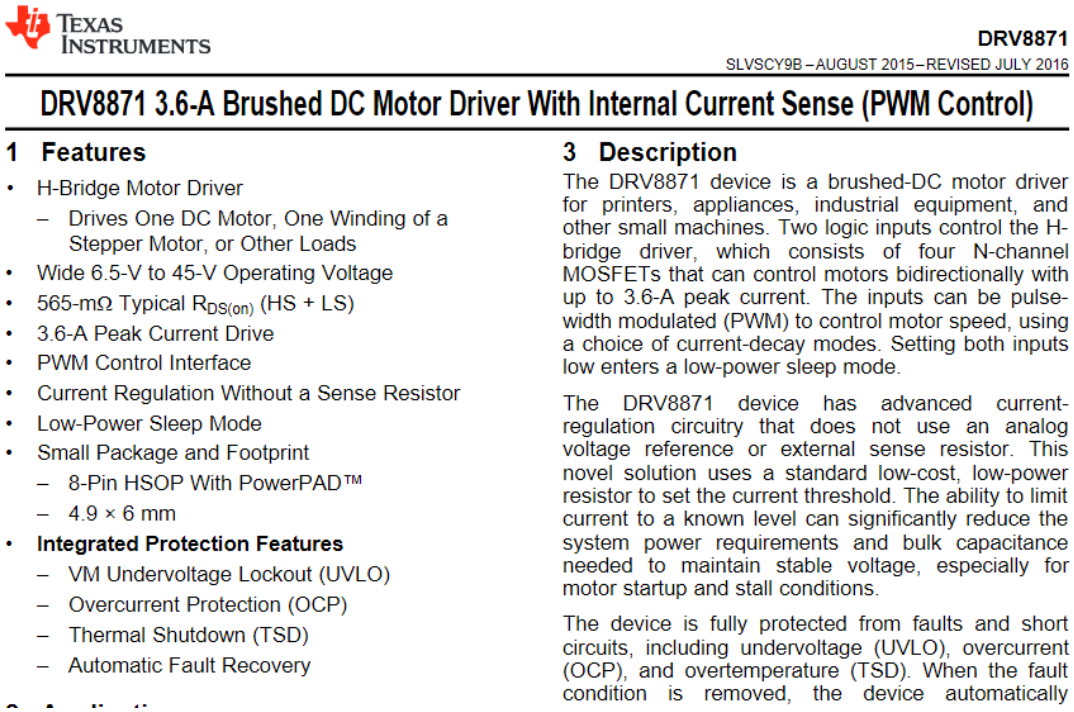


Figure 15: Basic diagram of a stepper motor's internals and operation. Keep in mind that the coils are slightly offset from each other so that each step smoothly connects to the next one. In fact, at just the right speed, the transition can be made perfectly smooth, when the motor freewheels in between steps. The coils should only be energized one at a time.

Red/Yellow = A/A- || Orange/Blue = B/B-

To drive the steppers, we use DRV8871 stepper motor drivers. 4 of them are used and are all tied to VADJ to have polarity and voltage control for not just the motors, but the charging pad and anything that may need separate control.



TEXAS INSTRUMENTS

DRV8871

SLVSCY9B – AUGUST 2015 – REVISED JULY 2016

DRV8871 3.6-A Brushed DC Motor Driver With Internal Current Sense (PWM Control)

1 Features

- H-Bridge Motor Driver
  - Drives One DC Motor, One Winding of a Stepper Motor, or Other Loads
- Wide 6.5-V to 45-V Operating Voltage
- 565-mΩ Typical $R_{DS(on)}$ (HS + LS)
- 3.6-A Peak Current Drive
- PWM Control Interface
- Current Regulation Without a Sense Resistor
- Low-Power Sleep Mode
- Small Package and Footprint
  - 8-Pin HSOP With PowerPAD™
  - 4.9 × 6 mm
- **Integrated Protection Features**
  - VM Undervoltage Lockout (UVLO)
  - Overcurrent Protection (OCP)
  - Thermal Shutdown (TSD)
  - Automatic Fault Recovery

3 Description

The DRV8871 device is a brushed-DC motor driver for printers, appliances, industrial equipment, and other small machines. Two logic inputs control the H-bridge driver, which consists of four N-channel MOSFETs that can control motors bidirectionally with up to 3.6-A peak current. The inputs can be pulse-width modulated (PWM) to control motor speed, using a choice of current-decay modes. Setting both inputs low enters a low-power sleep mode.

The DRV8871 device has advanced current-regulation circuitry that does not use an analog voltage reference or external sense resistor. This novel solution uses a standard low-cost, low-power resistor to set the current threshold. The ability to limit current to a known level can significantly reduce the system power requirements and bulk capacitance needed to maintain stable voltage, especially for motor startup and stall conditions.

The device is fully protected from faults and short circuits, including undervoltage (UVLO), overcurrent (OCP), and overtemperature (TSD). When the fault condition is removed, the device automatically

Figure 16: DRV8871 Datasheet. It's important to use a well-designed H-bridge driver to prevent flyback voltage spikes returning to the microcontroller. Removing an energized motor from a connection can cause serious voltage spikes.

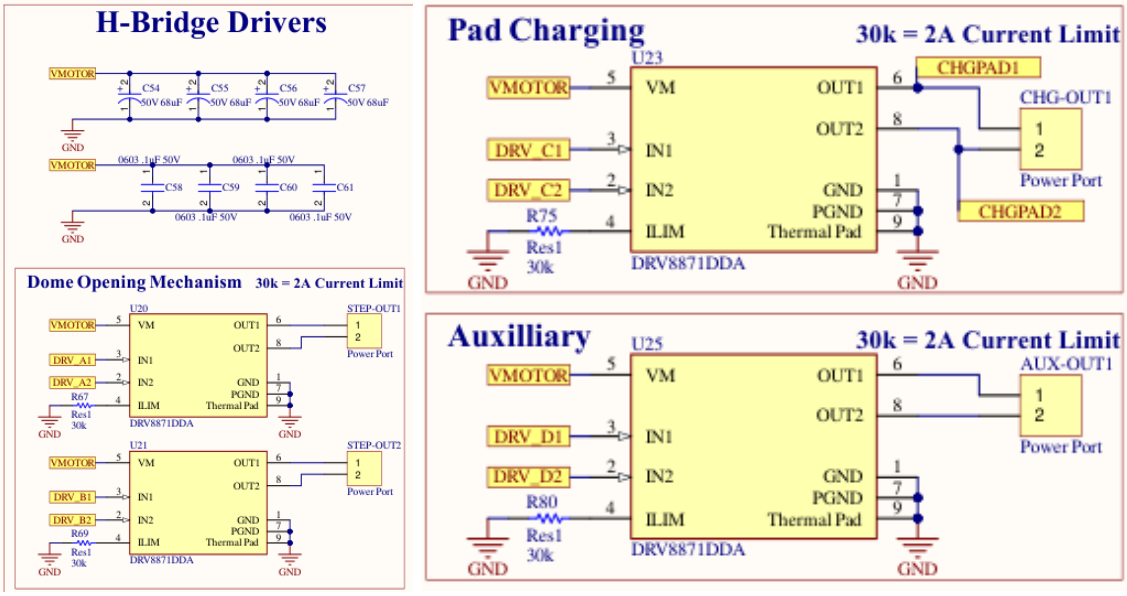


Figure 17: DRV8871 Schematic. 4 Are used, all tied to VADJ, while the Charging H-Bridge is also connected to the polarity detection circuit.

e. Charge Port Polarity Detection

Keeping true to Dieter's polarity detection design, this has been migrated off the prototype and directly onto the new board. SMD parts were implemented. Unfortunately, this was implemented incorrectly on the board and has been remedied after the fact.

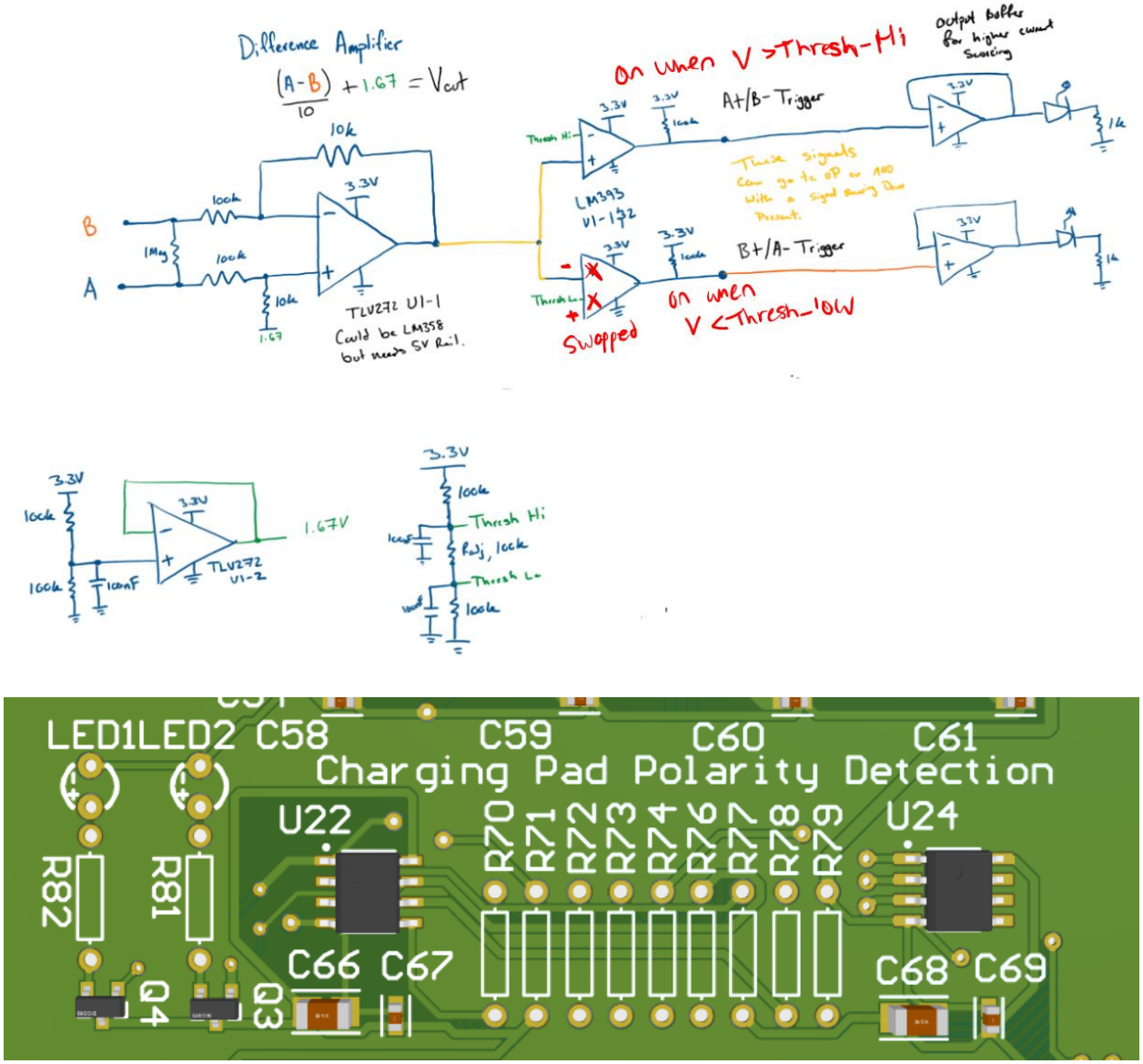


Figure 18-19: Polarity Detection Circuit. Sends 1 of 2 inputs high when a battery is connected. Both off in a disconnect scenario. This H-bridge should always be off until after polarity is detected. At such time, the H-bridge is set to parallel with the voltage detected.

f. User Controls

On-board controls were added to test functionality when the BeeLink is not in use. We went with a 4x20 LCD, Rotary Encoder, Push Buttons, and a Piezo buzzer for status feedback.

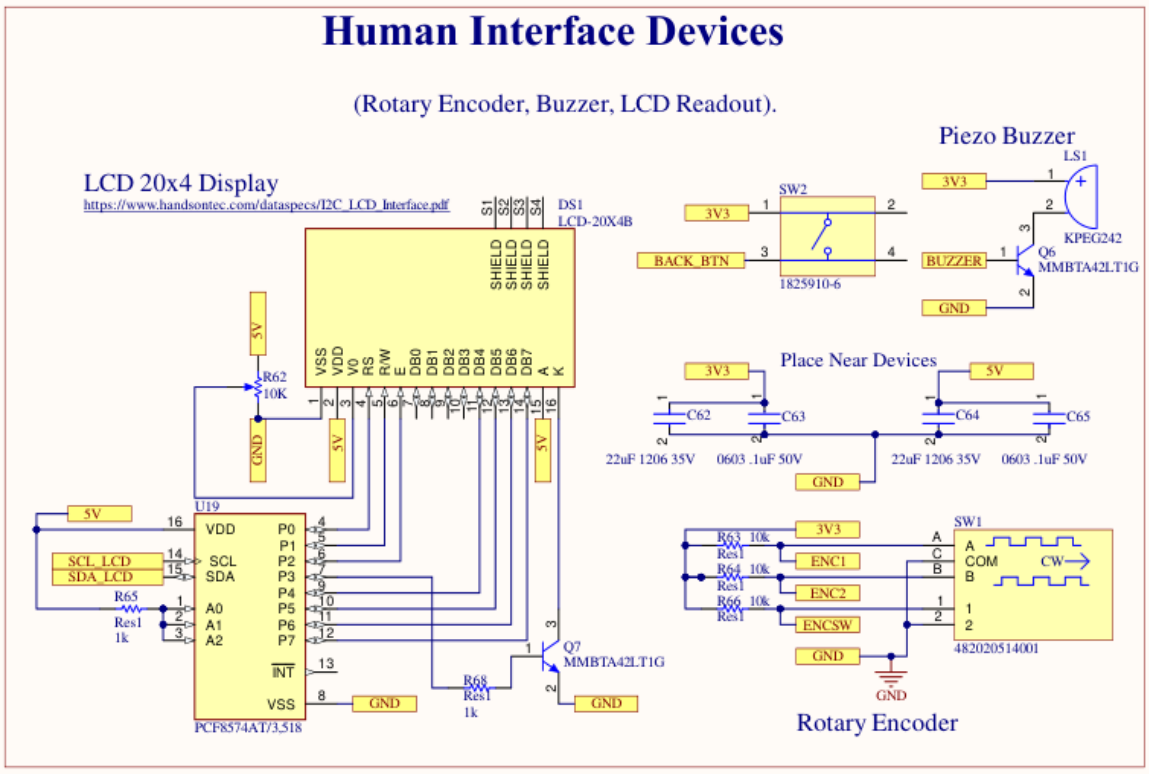


Figure 20: Wiring Diagram for Human Interface Devices added to the board.

Figure 21-24: Interface Devices

Utilizing a rotary encoder is a solid choice for many reasons. However, the primary feature involves the integration of 3 separate control devices into a single input. As stated in figure 19, the rotary encoder is a digital input device with a grounded common port and pull-up inputs at the A, B, and 1 connection, corresponding to left, right, and enter.

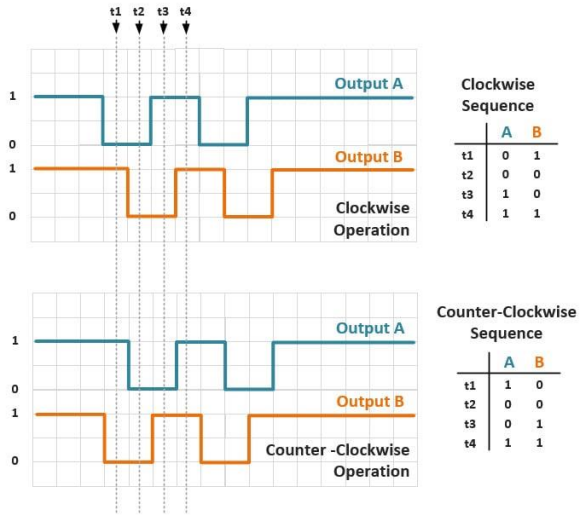


Figure 25: Rotary encoder timing diagram

A simple solution for integrating rotary encoders on microcontrollers is quite simple:

1. Attach both A and B pins to input pullup pins.
2. Attach both A and B to a falling edge interrupt.
3. Upon handling interrupts, check if the value of the opposite signal, if '1' (active low), accept input as valid. Add a time delay requirement before accepting the next interrupt for either A/B
4. The push button input can be treated identical to how you would handle a debounced button.

The 20X4 LCD utilizes a PCF8794 I2C I/O expander commonly used in I2CLCD combo modules. This device utilizes sequential bit banging to free up GPIO for other controls on the Pi Pico. A major downside is decreased display update rates, as 100kBaud is the maximum official transfer rate.

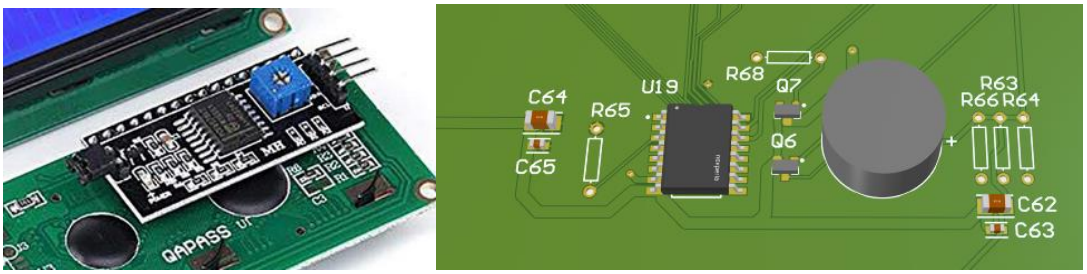


Figure 26-27: I2C interpreter module commonly implemented to control LCD I/O (left). The module components are implemented on the top layer of the final PCB (right).

g. I2C Switch

The TCA9548A I2C switch was previously utilized. A separate channel was configured for use with the 20X4 LCD and its accompanying IO expander.

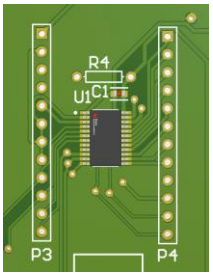


Figure 28: TCA9548A I2C Switcher IC

h. Final Board Changes

Many proposed changes and additions of the original design are laid out below:

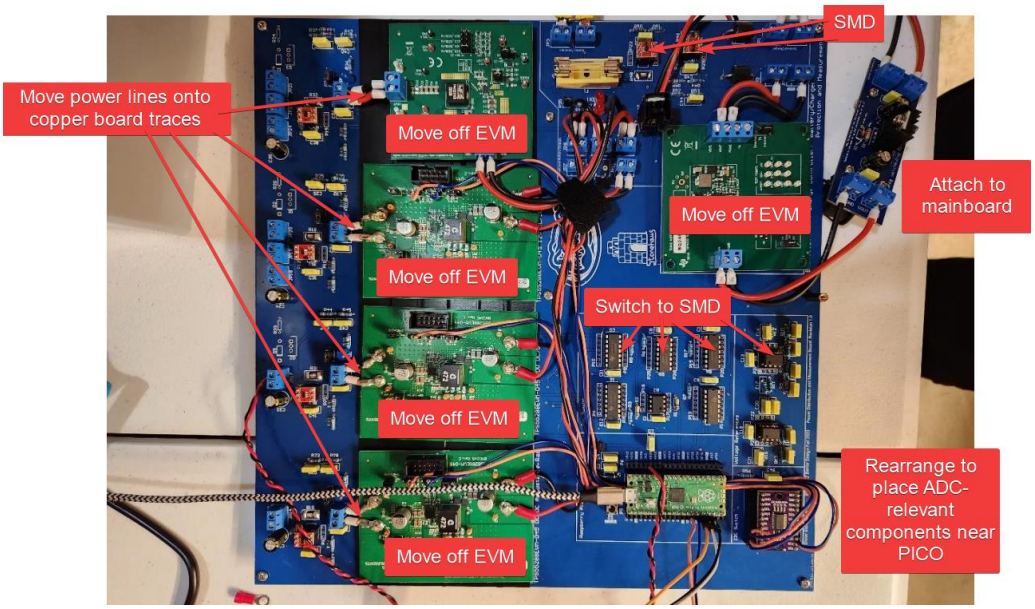


Figure 29: Drone station BMS board V1.0 major design change points. It is also important to mention that the reverse polarity modules should ideally be mounted onboard as well. The Raspberry Pi Pico will be left socketed in the event of a failure.

Other changes involved include, but were not limited to:

- Reconfiguring the MPPT Charge Controller set point by adjusting resistors.
- Replacing major power connections with XT-60 connectors
- Fix diode arrangement on power input.
- Fix capacitor on 3V shunt supply as board capacitance caused voltage floating.

II. Software

Primary goals for the JUCE GUI on the BeeLink module include:

- Display data sent via Pi Pico, including all Voltages, Currents, and Power Calculations.
- Plot information from power readings to real-time graphs.
- Add multiple pages for data separation of graphs, real-time data, and other relevant information.
- Prepare core to accept network information directly from Starlink, the satellite-based internet service used for remote devices in low-service areas. Starlink will prove excellent for autonomous farm drone research in areas that 5G or 4GLTE are incapable of reaching.

Methods

I. Hardware

a. Board Implementation Guidelines

When designing the new board using SMD tools, special considerations needed to be made:

- EVM schematic designs MUST adhere to proper placement to ensure reliable operation.
- Keep passive components as parts-bin as possible. Ensure when similar capacitances and resistors are used, it’s better to reuse parts from other designs whenever possible.
- Avoid NRFND parts (not recommended for new designs)
- Spec parts close, but not too close to maximum ratings.
- Keep part designators organized and easy to locate.
- Keep placements logical and close to active components.

b. Altium Design Duplication

One main tool I wanted to use heavily in this design was the reuse block/snippet tools. Unfortunately, this feature is locked behind organizational function. Any devices that are reused should be copied and pasted across both schematics and PCB placement. It's important to annotate alternative suffixes for reused schematics to improve placement efficiency across the board, especially since the parts used are identical.

The schematics can simply duplicate, but when it comes to PCB placement, the steps are a bit more complicated:

- Create separate projects for each reused schematic.
- Annotate with suffix 'A' for first of each copy, then ascend alphabetically.
- Do 1st PCB layout just like normal.
- Select all, copy to B, C, D… PCB docs.
- Reannotate copied schematic docs, change suffix. Doing this keeps numbering the same.
- Update copied PCB docs to reflect reannotations in schematics for each mini-project.
- In the actual final project, copy each PCB doc individually to the main file before calling update. This will set Altium straight when it realizes the parts already exist in the PCB. These components won't be overwritten.

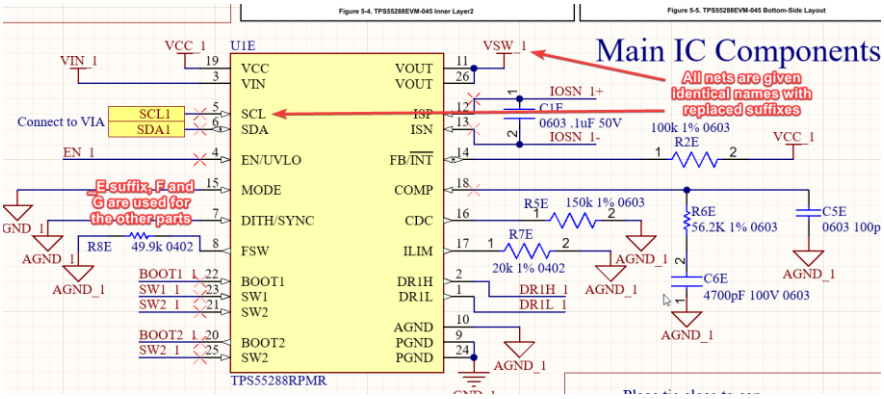


Figure 30: Schematic annotation example

c. Altium Schematic Design Templates

Although not entirely necessary, a uniform design template improves professional appearance, as well as allowing for more defaulted area for schematic sheets. For this project, a size of 9.5"x15" was used. This size will comfortably fit on ledger 11"x17" paper. An IMG schematic sheet template has been created and is included in the supplied files.

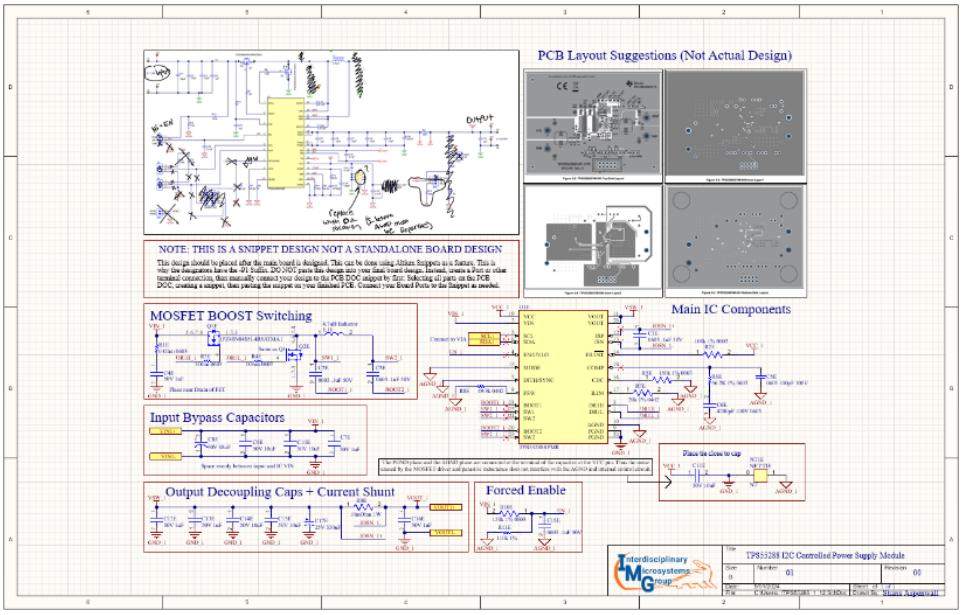


Figure 31: IMG Schematic Template

d. EVM Translation

Many of the replacement designs from the EVMs follow extremely close to the EVM designs, but with reduced part counts for components used for modular configuration.

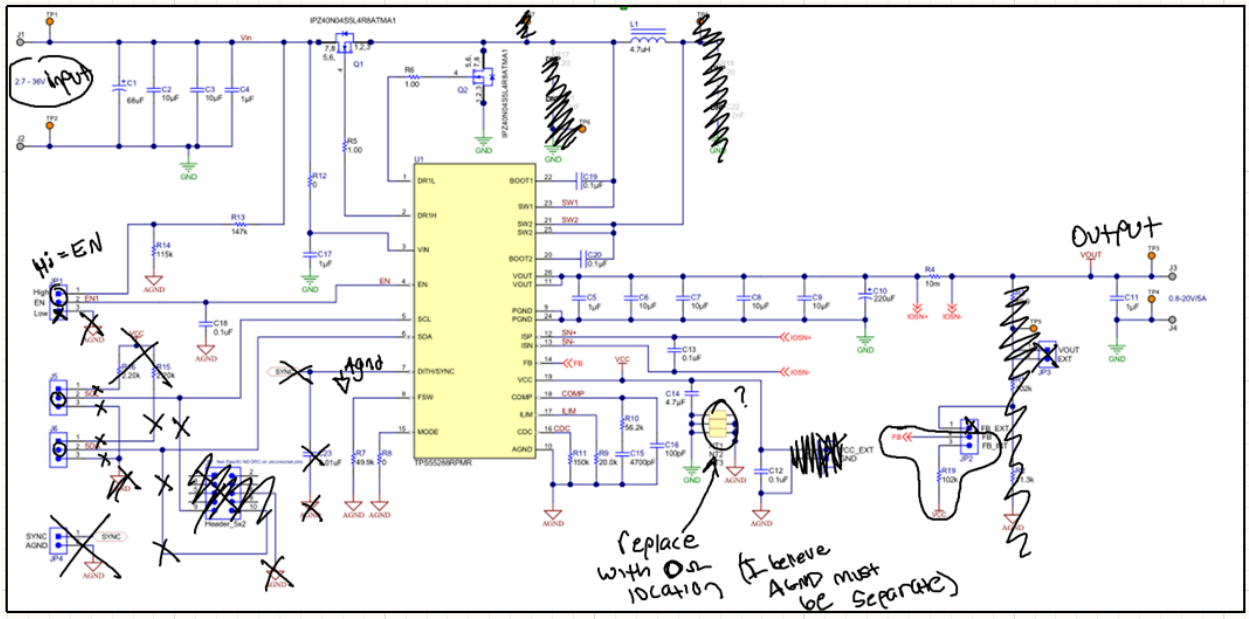


Figure 32: EVM Schematic for the TPS55288 I2C. Take note that many components can be ignored, as they are for configuration.

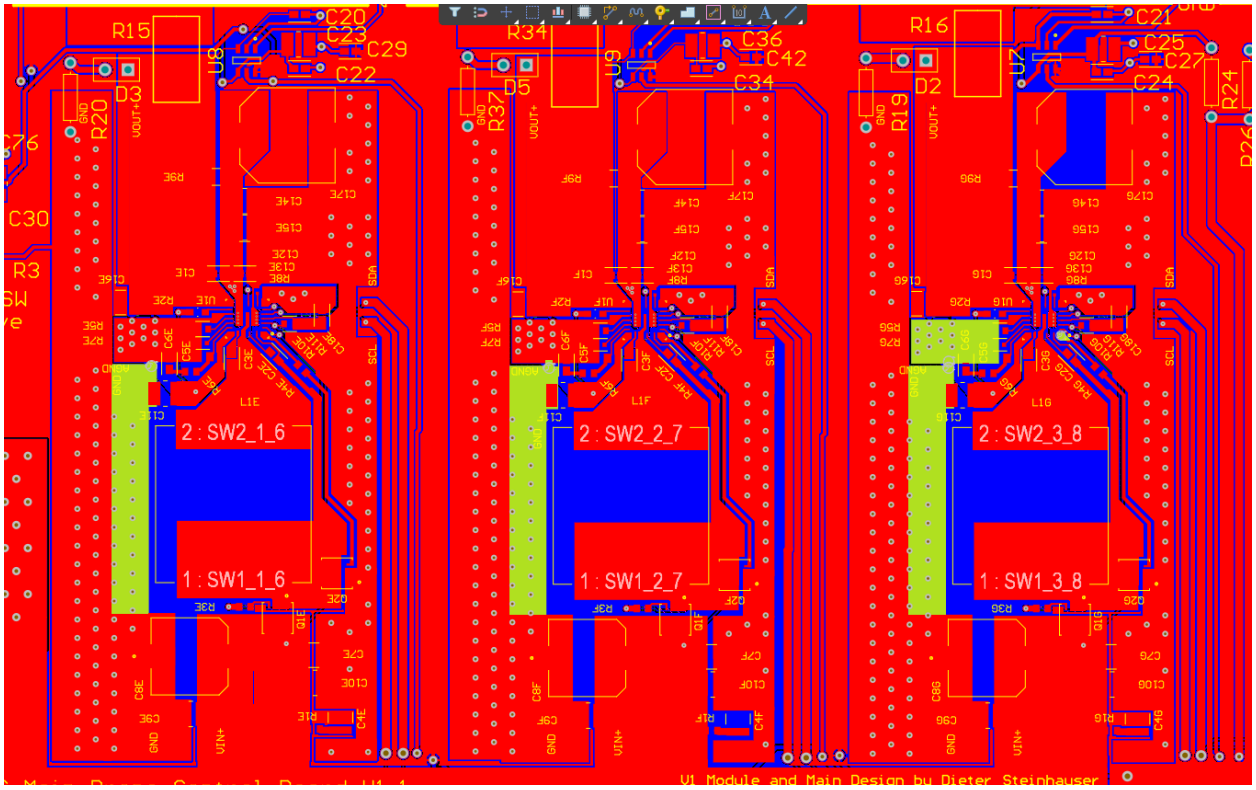


Figure 33: TPS55288 copy/pasted design on the board. Notice the replaced suffixes for replicated components.

e. LM5156 PoE Power Supply

The following section refers to the design and implementation of a 48V PoE power supply for Starlink, a satellite-based internet service provided by SpaceX. Our desired specifications were a minimum of 3A of continuous supply.

Using TI's WEBENCH tool, you can put in the specs of a DC/DC power design you want. We want to power Starlink from a 12 V battery, and Starlink requires about 48 W and 100-150 W of power (meaning 48 V and 4 A of current MAX), we put this as our specs in WEBENCH.

We chose to go with following circuit among the many that WEBENCH gave us:

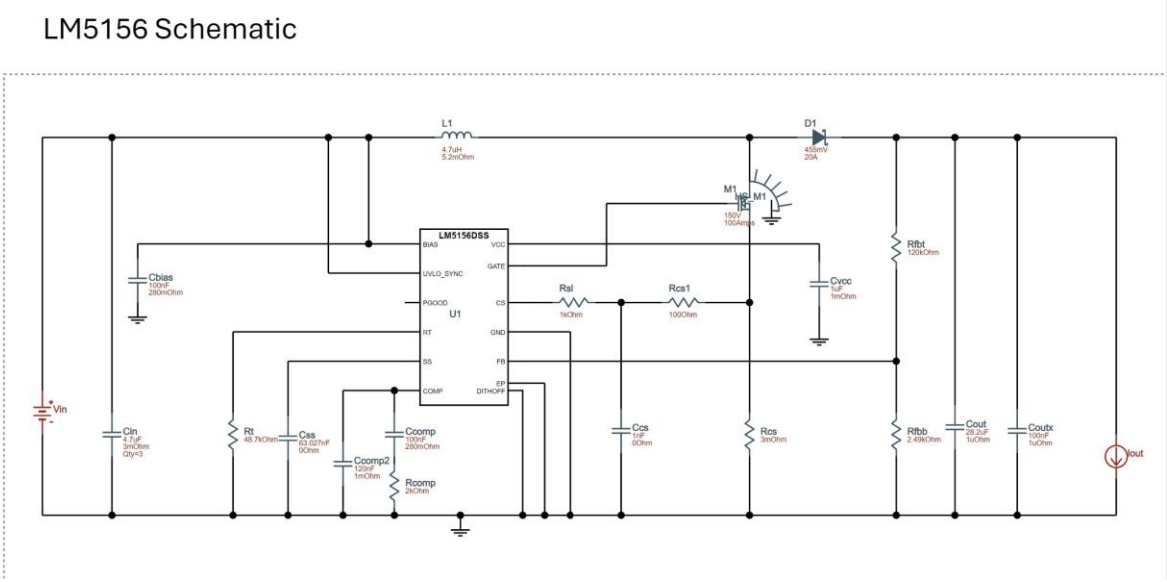


Figure 34: LM5156 Design from WebBench

Why this one? Because of the many boost converters with different IC's (which are Boost Converter controllers, sending the PWM signals to turn on and off the MOSFET), the LM5156 is a newer chip than most of them, so it's the most improved / resistant to noise.

For the LM5156, at first, in Altium Designer there was NO schematic NOR footprint. So you have to search this component up in DigiKey and download it, and import it into Altium to make an Integrated Library.

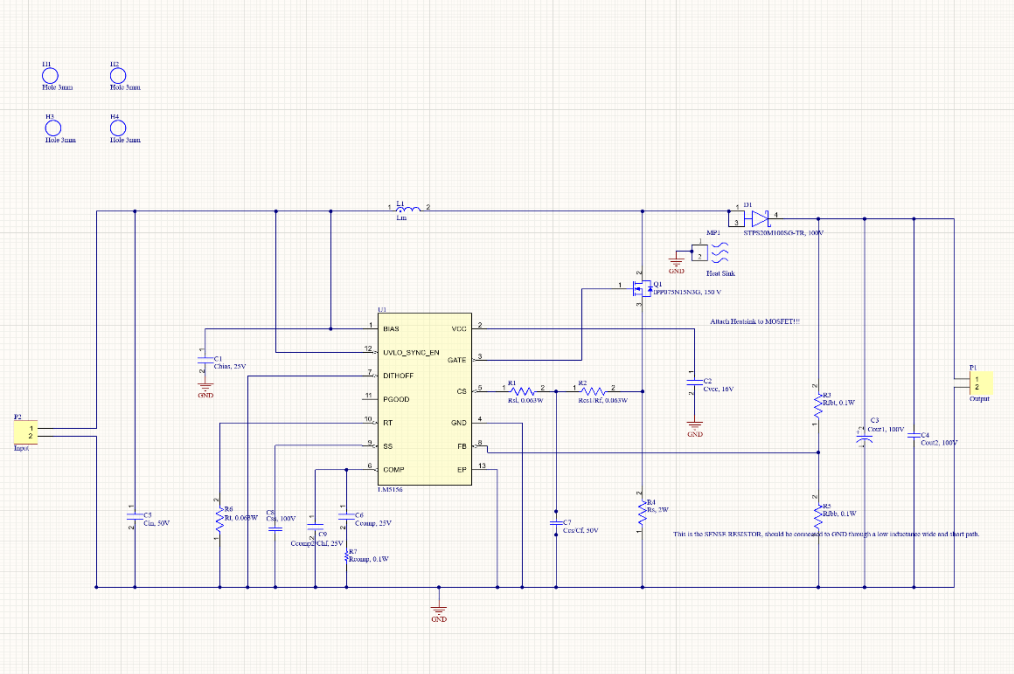


Figure 34: LM5156 Design from WebBench

(It's good practice to comment on the voltage ratings (for capacitors), and power ratings for resistors, to be able to see them quickly. Then we can quickly eyeball and see if everything can handle its max voltage or max power.)

Design Choices:

We made C3 larger than the initial value given by WEBENCH, and this drastically decreased the output voltage ripple in the transient simulation on the WEBENCH site.

There were 4 components we had to choose ourselves instead of using what the WEBBENCH decided:

1. Rs- this was 3 mOhms, so we just found a similar resistor that could handle the power requirements

And 3 custom capacitors:

2. Cout1: Went from 28uF to 47 uF. **When re-simulating,** we found the simulation on WEB-BENCH with this Cout1 value made the output vary LESS, so we improved the original design.

3. Cout2: Kept at 100 nF

4. Css: Initially chosen by WEB-BENCH to be 63 nF, we bumped up to 100 nF.

- Css is the "Soft-Start" capacitor. Immediately when we turn the boost converter on, this controls how fast the voltage jumps from 12 V to 48 V for us. The higher the capacitance, the slower the jump is.

$$R_{UVLOT} = \frac{0.967 \times V_{SUPPLY(ON)} - V_{SUPPLY(OFF)}}{5\ \mu A} = \frac{0.967 \times 2.6\ V - 2.2\ V}{5\ \mu A} = 62.8\ k\Omega \quad (19)$$

A standard value of 60.4 kΩ is selected for R_{UVLOT}. Using Equation 20 the top UVLO resistor (R_{UVLOB}) is calculated

$$R_{UVLOB} = \frac{1.5\ V \times R_{UVLOT}}{V_{SUPPLY\_ON} - 1.5\ V} = \frac{1.5\ V \times 60.4\ k\Omega}{2.6\ V - 1.5\ V} = 82.36\ k\Omega \quad (20)$$

A standard value of 80.6 kΩ is selected for R_{UVLOB}

3.10 Soft-Start Capacitor Selection.

The soft-start capacitor is used to minimize and overshoot on the load voltage during the start-up of the regulator. Equation 21 is used to calculate the minimum recommended soft-start capacitor value.

$$C_{SS} > \frac{10\ \mu A \times V_{LOAD} \times C_{LOAD}}{I_{LOAD} \times V_{REF}} = \frac{10\ \mu A \times 12\ V\ 200 \mu F}{3\ A \times 1\ V} = 8\ nF \quad (21)$$

where

- V_{REF} is the feedback voltage reference of the LM5156.

For this design a CSS value of 220 nF is selected to minimize any overshoot on the load voltage during start-up.

3.11 Feedback Resistor Selection

The feedback resistors (R_{FBT}, R_{FBB}) set the regulated load voltage by comparing the scaled voltage to the internal voltage reference. To help limit the bias current of the feedback resistor divider, R_{FBT} is selected to be 49.9 kΩ. Equation 22 is used to calculate the value of R_{FBB}.

$$R_{FBB} = \frac{R_{FBT}}{\frac{V_{LOAD}}{V_{REF}} - 1} = \frac{49.9\ k\Omega}{\frac{12\ V}{1\ V} - 1} = 4.53\ k\Omega \quad (22)$$

R_{FBB} is selected to be 4.53 kΩ.

3.12 Control Loop Compensation

Figure 35: Soft start capacitance calculations

PCB Layout

If we look at the datasheet for the LM5156, they give Layout guidelines for the PCB:

12 Layout

12.1 Layout Guidelines

The performance of switching converters heavily depends on the quality of the PCB layout. The following guidelines will help users design a PCB with the best power conversion performance, thermal performance, and minimize generation of unwanted EMI.

- Put the Q1, D1, and R_S components on the board first.
- Use a small size ceramic capacitor for C_{OUT}.
- Make the switching loop (C_{OUT} to D1 to Q1 to R_S to C_{OUT}) as small as possible.
- Leave a copper area near the D1 diode for thermal dissipation.
- Put the device near the R_S resistor.
- Put the C_{VCC} capacitor as near the device as possible between the VCC and GND pins.
- Use a wide and short trace to connect the GND pin directly to the center of the sense resistor.
- Connect the CS pin to the center of the sense resistor. If necessary, use vias.
- Connect a filter capacitor between CS pin and power ground trace.
- Connect the COMP pin to the compensation components (R_{COMP} and C_{COMP}).
- Connect the C_{COMP} capacitor to the power ground trace.
- Connect the GND pin directly to the analog ground plane. Connect the GND pin to the R_{UVLOB}, R_T, C_{SS}, and R_{FBB} components.
- Connect the exposed pad to the GND pin under the device.
- Connect the GATE pin to the gate of the Q1 FET. If necessary, use vias.
- Make the switching signal loop (GATE to Q1 to R_S to GND to GATE) as small as possible.
- Add several vias under the exposed pad to help conduct heat away from the device. Connect the vias to a large ground plane on the bottom layer.

Figure 36: Layout Guidelines

They also give sample layouts. I decided to follow the 1st one:

12.2 Layout Examples

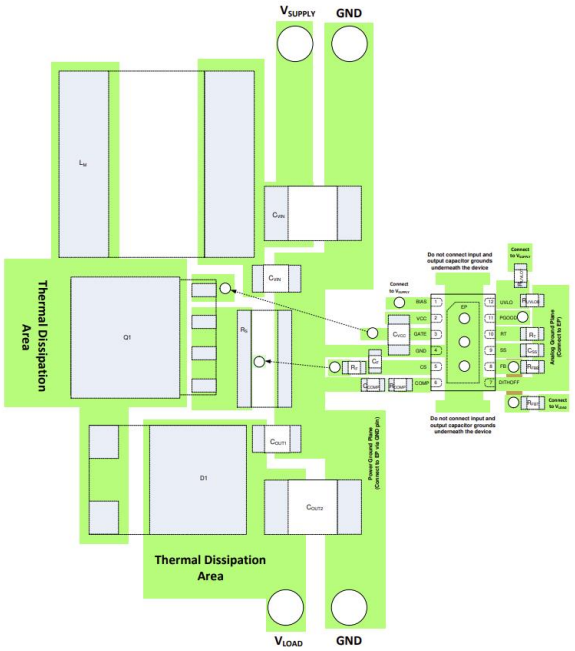


Figure 12-1. PCB Layout Example 1

Figure 37: Layout Examples

10.2 Typical Application

Figure 10-1 shows all optional components to design a boost converter.

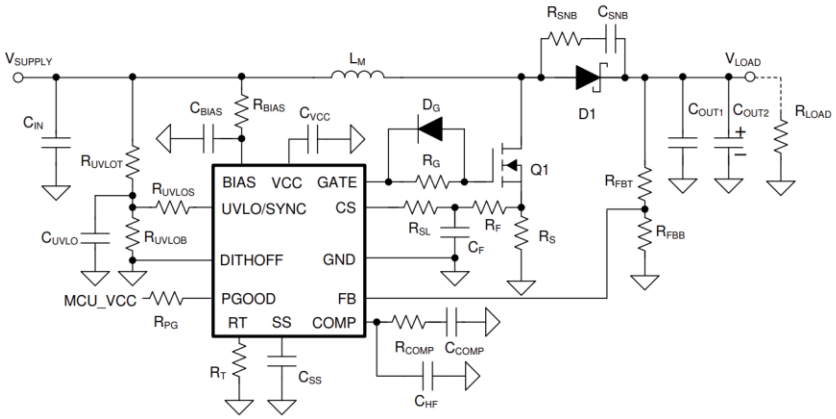


Figure 10-1. Typical Boost Converter Circuit With Optional Components

Figure 38: Boost converter application

Altium's AutoRouting was NOT good at all to keep with the Layout guidelines and for these surface mount parts. So I had to auto route. The final Layout turned out to be:

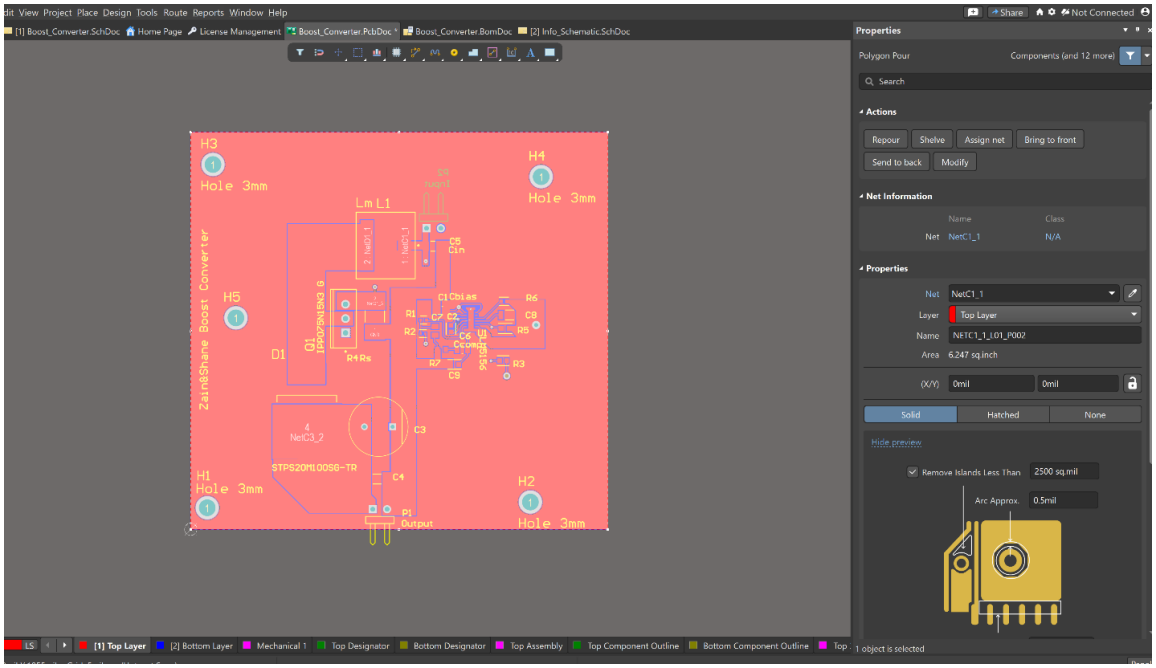


Figure 39: PCB layout of LM5156

The entire space around and including H5 (to the left of the board), is for us to solder Q1 (the MOSFET), and then fold it and put it flat on that area. By contact, it'll help dissipate its heat throughout the PCB, because this MOSFET will hold a lot of the power that is passing through the boost converter.

One key thing I want to point out in the below picture…

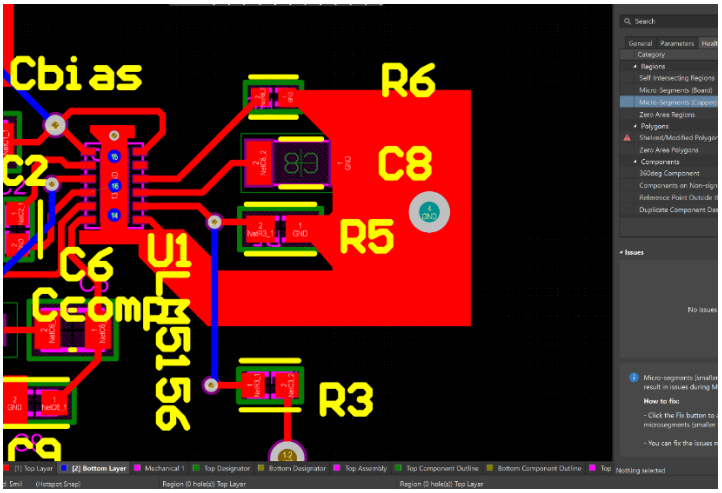


Figure 40: Layout considerations regarding quirks of the switching supply controller

1. Why did we design it like this, with that pad connected to GND? Because if we look at the LM5156 Example layout:

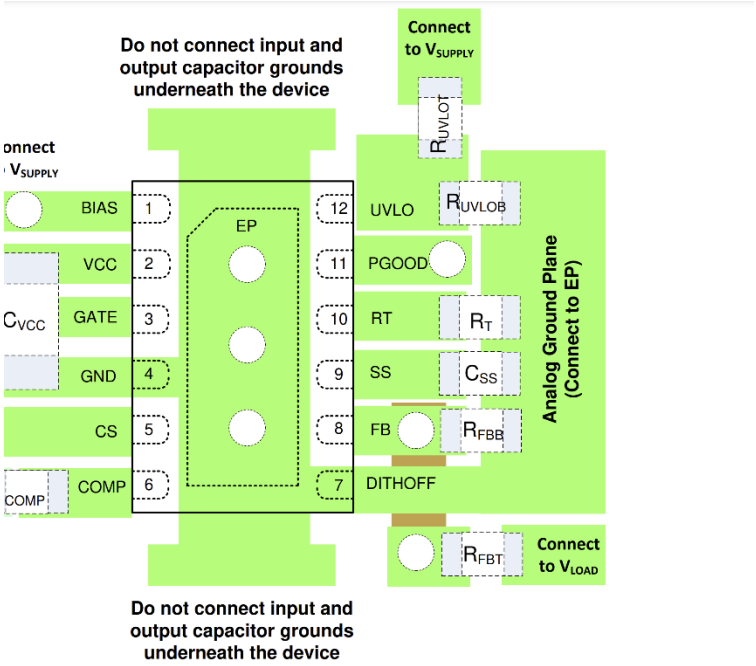


Figure 41: Example layout

We see it connects Pin 7 and all the component's at pins 8, 9, and 10 (the entire right side of the LM5156 chip) are ALL connected to the **same** "ANALOG" GND

Side Note: And also, to connect Pin 8 to Rfbt (or R3 in the picture), the example layout recommends to use a VIA, which I followed.

Why? **This PCB has TWO GND's – a Power GND vs. an Analog GND (encouraged by datasheet).**

To reduce the noise that reaches the LM5156, which is sensitive to it.

Consider the following Analogy: Down in SoFlo, we have the Atlantic Ocean (this is Power GND in the analogy)

Then we have the Intercoastal (short strip of water between mainland Florida and Barrier islands (this is Analog GND in the analogy)

In a storm, or even on a normal day, the Intercoastal is WAY less wavy than the ocean

So, for the IC to have a more STABLE GND reference to operate, it has its own separate GND plane

But why for the Boost Converter?

Because we're charging and discharging the inductor, which produces HUGE spikes of electric field and huge spikes of Voltage

This actually messes with the GND plane- normally its at 0 V, but it could be at -5 V (meaning now, 5V is the new 0V), or it could be 3 V

If this were to happen with the LM5156, an IC, this would destroy it

So the LM5156 has its own GND, or Analog GND which is more stable

Power GND IS connected to Analog GND through a SINGLE EP (Exposed Pad), seen here:

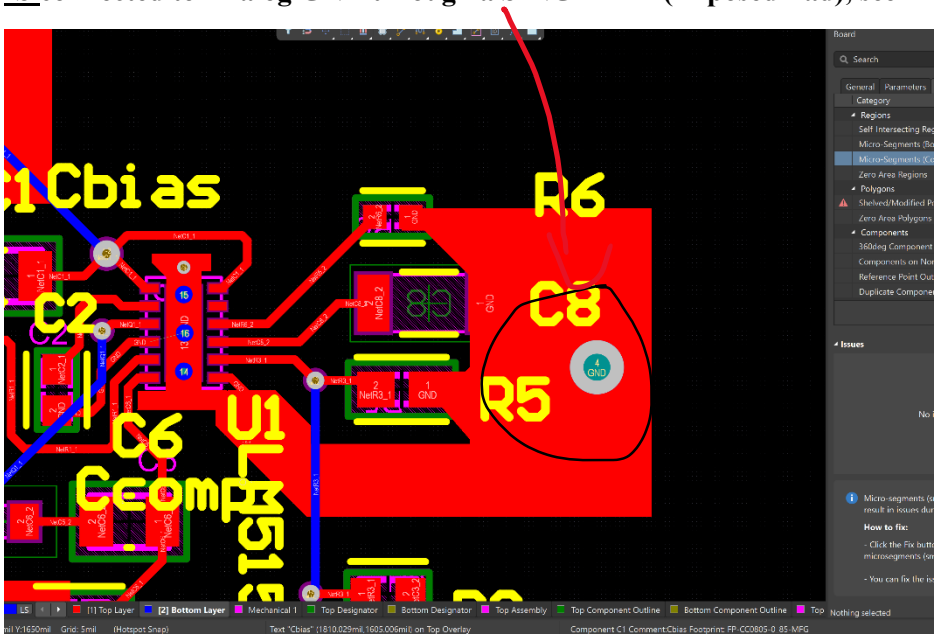


Figure 42: Net Tie between PGND and AGND

We only want ONE connection from Analog GND to Power GND (which is the pad shown above). Why?

Because we DO want the 2 GND's to be connected

But at the crossing point between 2 GND's, it should be narrow, so there'll be a resistance.

So, it'll discourage noise, in the form of current (which causes a VOLTAGE DROP), to come to the Analog GND and make it change from 0V

This causes Analog GND to be WAY more stable than Power GND, letting the LM5156 function smoothly.

Another notable picture is:

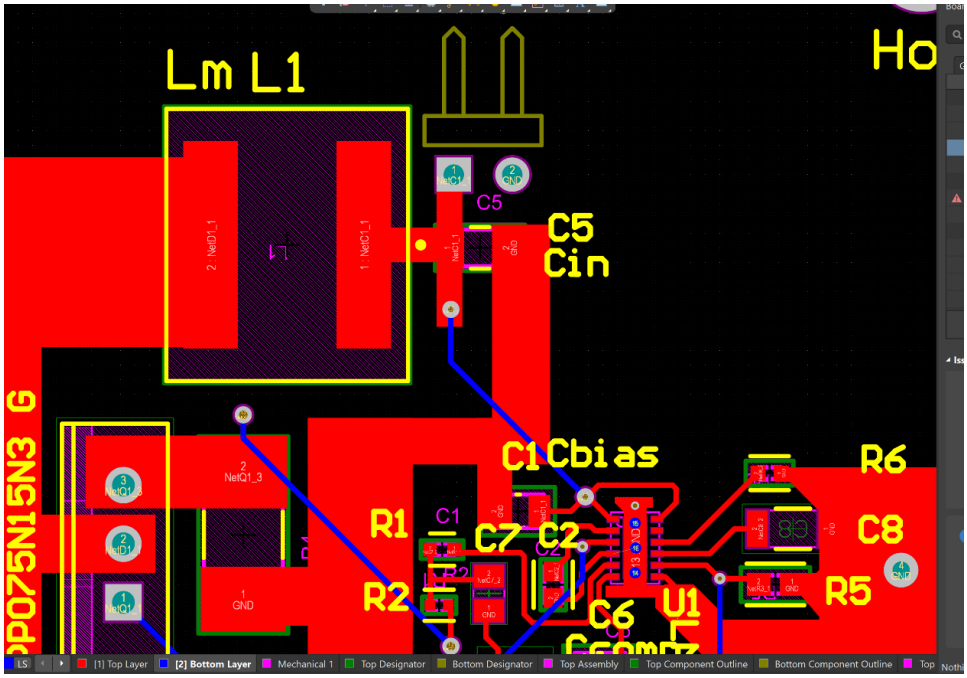


Figure 43: Part placement considerations

Also, we want U1 (LM5156 IC) and L1 (the inductor) to be as close as possible. Why?

These are the MAIN two components on our boost converter

Their distance should be as close as possible to each other on the PCB, to minimize resistance and parasitic inductance / capacitance

This minimizes any interference to this “switching loop”

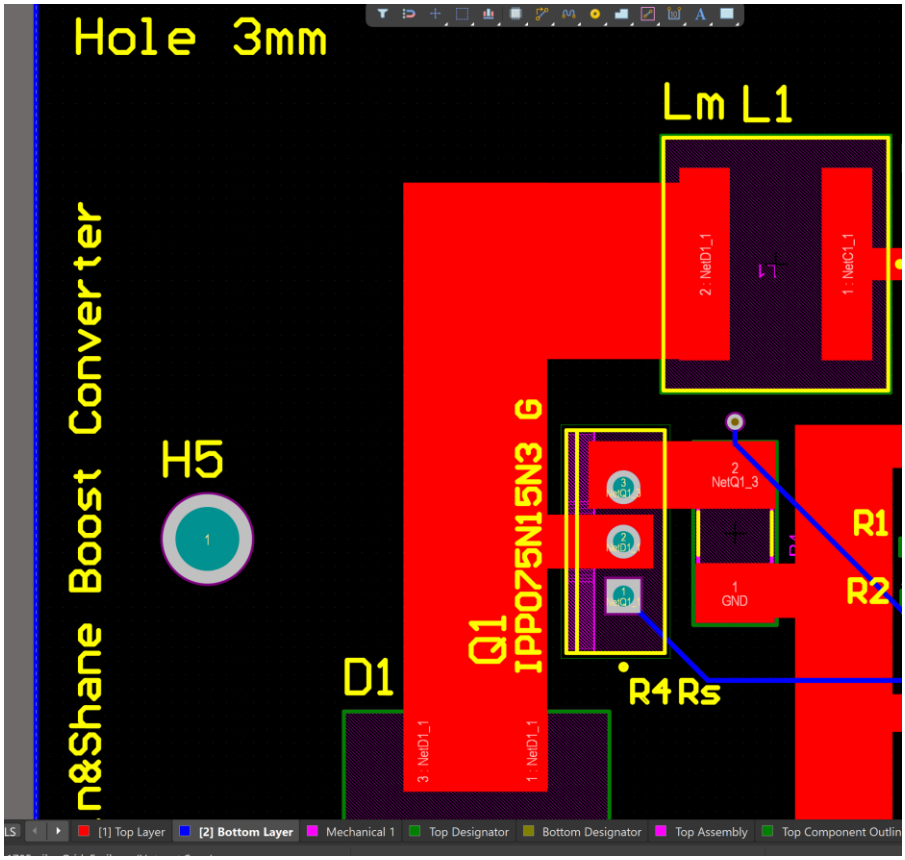


Figure 44: Power traces

Key points in the above picture:

1. Because this trace connected the output of the inductor to the diode, and MOSFET, it's carrying those 100-150 W of power

Thus, the copper should be as WIDE as possible, because a lot of current is going through this trace, and we want there to be as little resistance as possible, to MINIMIZE heat dissipation (so the current won't heat up the trace and we'll lose our efficiency)

These nice, rectangular regions are made using the "Fill" feature (in the same place the "Solid Region" in Altium

2. We have Hole H5, because we'll bend the Transistor Q1 when we solder it onto the PCB, to the left, and the hole on the transistor will match up with this hole

So we intend all the heat in the transistor to be dissipated by the PCB. This is even recommended in the LM5156 datasheet:

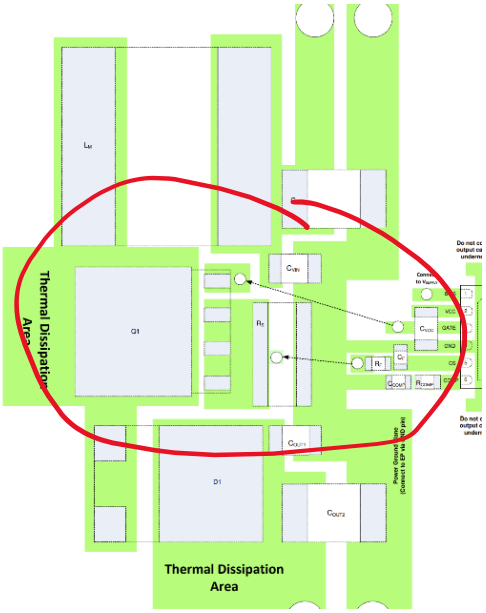


Figure 45: Polygon pour connection.

Also, since the board takes in about 12V at 10A, and outputs 48W at about 2-2.5A, there's a LOT of input current. To spread this current and heat out onto the board, we made the top layer's polygon pour connected to the input pin

And, like normal, the bottom layer's polygon pour be the GND (which represents Power GND).

These are just some of the design decisions that made our boost converter this successful and be able to handle the near 100W of power running through it.

f. Finished Design and Assembly

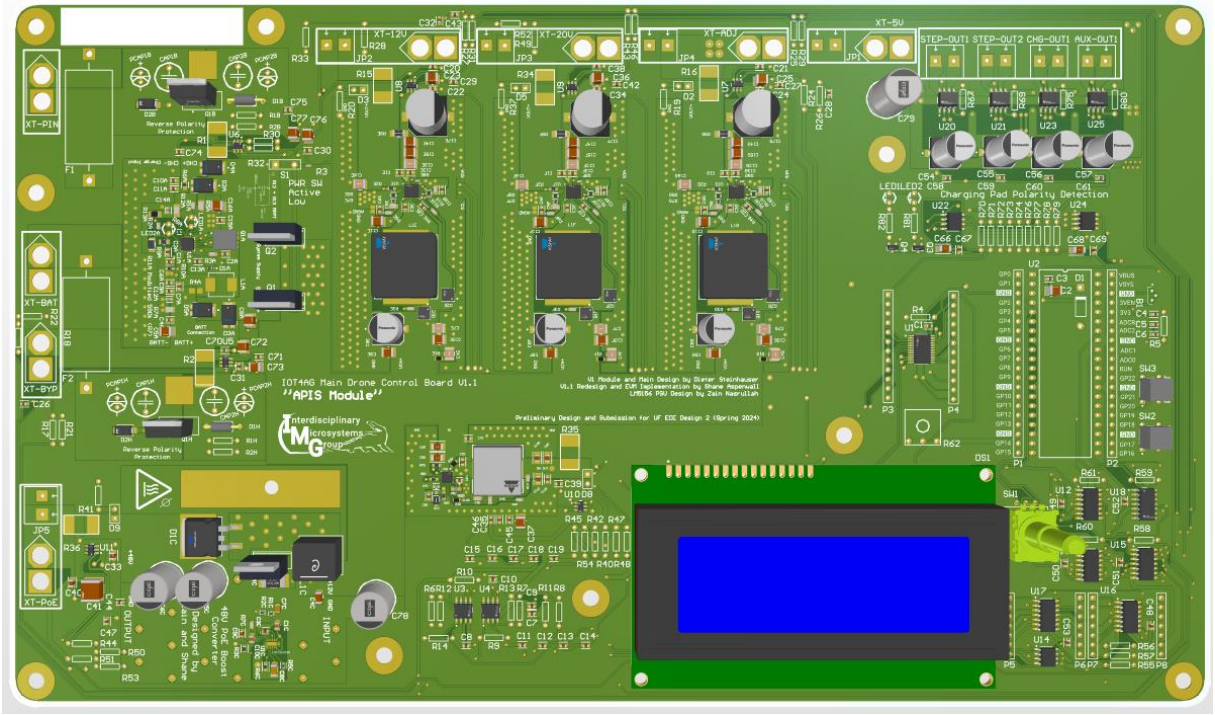


Figure 46: Final V1.1 Board Design and Mock-Up

Before finalized ordering and assembly, parts needed to be cross-referenced with the parts suppliers to ensure compatibility with footprints, ensure plentiful stock, and guarantee parts information with the rated requirements of each section.

This board was designed on a 2-Layer process. The only benefit gained by this was increased lateral strength, simpler and easier to fix routing, and especially lower cost. However, if given the ability to redo, I would recommend a 4-Layer process to add dedicated power and ground planes that could also separate high-power and small-signal connections on the top and bottom layers.

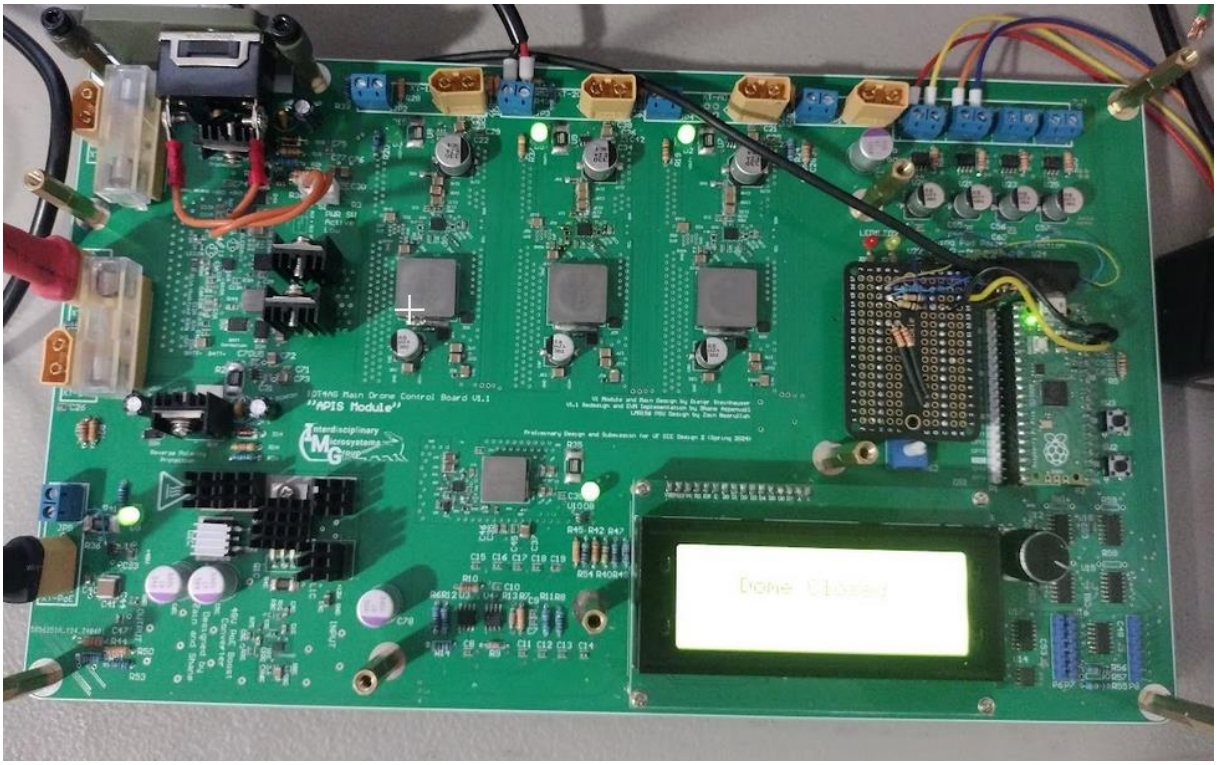


Figure 47: Final V1.1 Assembly

g. Starlink

First, we used the AC powered router they gave us, plugging it into a wall outlet. We passed this through a wattmeter to see how much power it consumed and thus which specs to design the BOOST CONVERTER with.

Then, we tried hooking up a DC Power Supply from our Lab with the following setup:

DC Boost Converter goes into PoE Injector (this temporarily takes the place of the Boost Converter which takes it's input from 12 V on board our final PCB).

The travel router is plugged in via Ethernet to the PoE Injector.

The PoE injector takes in both, and outputs an Ethernet cable going to an adaptor, which converts from RJ45 (the Ethernet) to a Dishy V2, which plugs into the satellite.

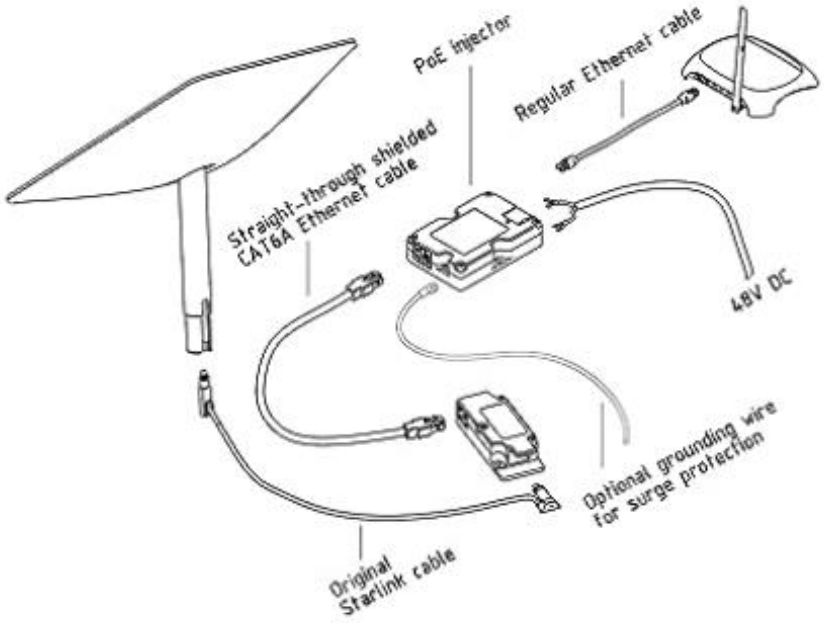


Figure 48: Starlink PoE bypass setup for use with a personal router setup.

Running the test, we might've gotten something messed up, because this broke our Starlink dish. SpaceX support said our Dish wouldn't work anymore and gave us a new one for free.

It was also worth noting that the pins of the Ethernet must be matched up correctly with the Ethernet port that is on the PoE injector (some of those pins are power and some are GND, and if you mismatch them, this will cause a short in the worst case).

We improved the connections from the power supply to the PoE injector and re-ran the process with the new satellite they gave us.

We first tried it on AC power, and it worked. Then we tried it by powering it using the lab DC power supply, and that worked.

The next logical step is to get the Boost Converter designed, put it on the overall PCB, and see if it can input a 12 V battery, pass a stepped-up voltage to the PoE injector, and power the Starlink satellite. See the next steps for this.

Starlink DC-DC Conversion

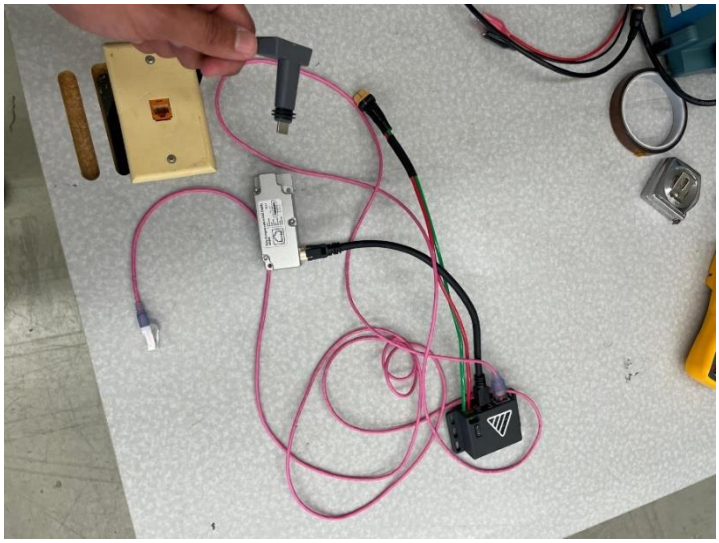


Figure 49: Starlink Dishy Wiring Connectors

What I'm holding in my hand (near the top of the picture) is the Dish V2 connector that comes straight from the Starlink dish. This plugs into the metal rectangle right below it in the picture (which is the Dishy V2 to Ethernet adaptor), and this goes into the PoE injector (seen in the bottom right of the photo). The green and red wires plug straight into the output of the Boost Converter on our PCB, seen below:

Figure 50: PoE injector receptacle on-board

This gets power to the PoE injector. Finally, the pink Ethernet cable plugs into whatever router you're using, so that the signal it broadcasts receives Internet.

A clearer of the PoE Injector is seen:

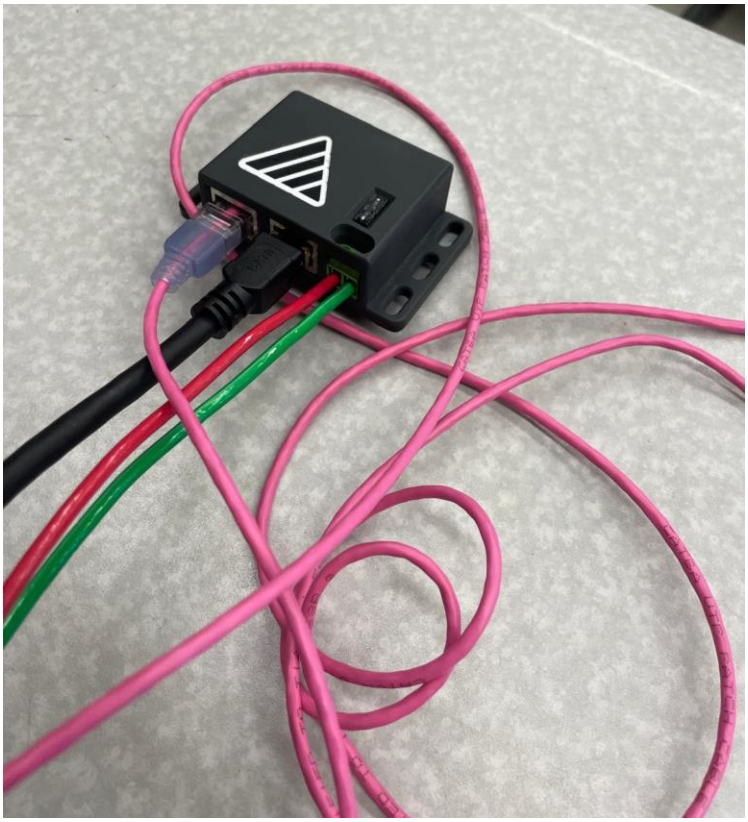


Figure 51: PoE Injector Box, with 48V Fused Input

II. Software

a. PC GUI Setup, Pico Setup, and getting Pico Data Transmission Working

First things first, you open the ProJucer by going to:

"Drone-Staton/control_station_gui/control_station, and opening "control_station.jucer" with Projucer (this should be an option after you install JUCE.

This gives you something like:

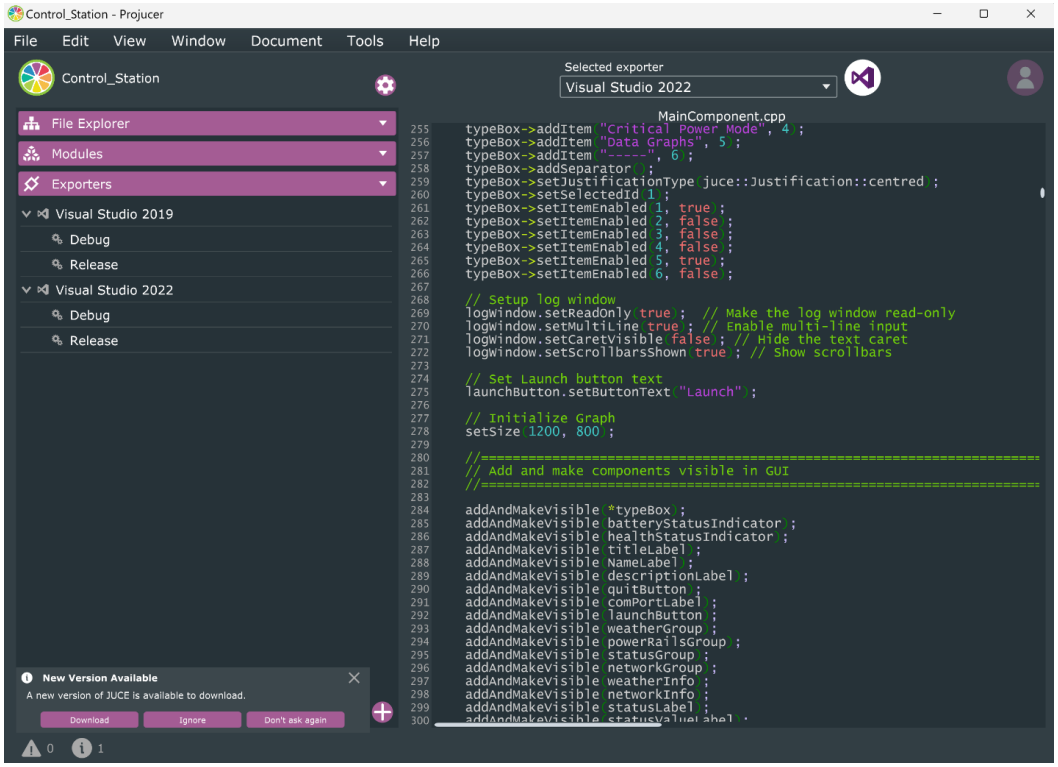


Figure 52: JUCE IDE in ProJucer

Once you press that white button (in the top right of the picture, near “Visual Studio 2022”, this exports to Visual Studio, opening it up, and you see the following:

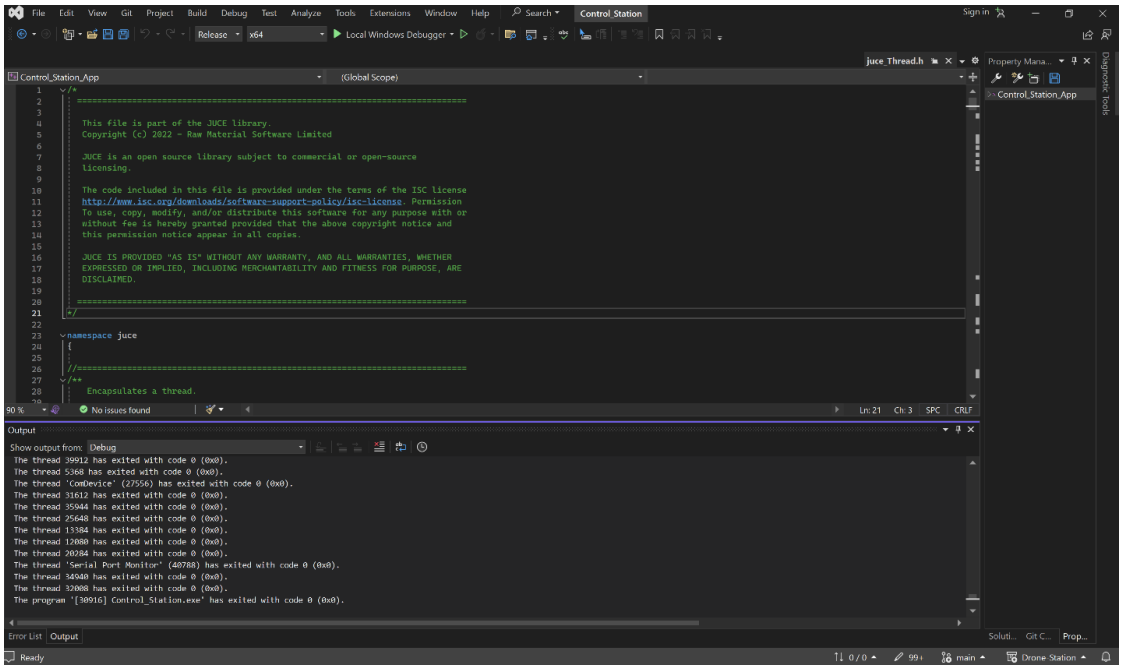


Figure 53: Visual Studio

To run the project and open the GUI, you click the green right arrow to the left (the one on the right is to run without Debugging, which doesn’t make sense to do. Running with debugging is still pretty fast).

To make the Data Transmission work from Pico to USB:

Connect the Pico to your computer via USB.

In Visual Studio Code, install the “MicroPico” extension. In the command palate, when you do CTRL > shift > p, and type in “micro”, it gives you all the commands you run.

Navigate to the “Drone_Station > pico_power_management folder, open it up in VsCode.“

Run the “Configure Project” command in the command palate. Then, “Upload project to Pico”. Now, the Pico will start running your code, and you have to make sure to close Visual Studio Code so the COM port of the Pico connected to your computer is free, **so the GUI can use it**

******ANY TIME you make a change in a file that’s NOT “main.py”, you have to select the command “Delete all files from board” and then re-upload all the files to the Pico.**

Now, CLOSE the VSCode window that has the Pico code open. This is because this window will be asking for the COM port, but we want the VS Code window that has the GUI code to have it. Only ONE Window can be open in order for serial transmission from the Pico to the GUI’s COM port to work.

Open the GUI from the running Visual Studio window. **To make the Pico transmission work**, **FIRST have the GUI running, THEN, with your code uploaded to the Pico, unplug and replug the Pico via USB to your computer. The GUI should show NEW data. If this doesn't work, while the GUI is up and running, press the "reset" button on the board to reset the Pico. This should work.**

b. PC GUI Network Information Panel

The first feature I added is a Network Information Panel. This is on the main, "Default" window of the GUI. The steps to do this are:

In MainComponent.h, in the MainComponent class, I added a private variable, juce::GroupComponent networkGroup. This makes the square that holds all the information about networks, and you can title that square.

Made a function, getNetworkInformation() in the MainComponent class, that uses **Windows APIs** to read the network information from your Windows computer and display it.

Thus, at the top of the "MainComponent.h" file, you see the logic: IF you're on a windows computer, then add the Windows specific libraries. If on Linux, then add those libraries specific to Linux. Thus, in the actual implementation of "getNetworkInformation()", you can have logic that says: if on Windows, run this code, and if on Linux, run this code

I made a timer, "eTimerUpdateNetworkInfoTimer", which runs regularly, and every time it runs, it calls the "getNetWorkInformation" function (this is in the MainComponent::timerCallback(int timerId)) function in "MainComponent.cpp"

Source > MainComponent.cpp > getNetworkInformation()

```
1641  void MainComponent::getNetworkInformation() {
1971
1972      }
1973      if (pConnectInfo != NULL) {
1974          WlanFreeMemory(pConnectInfo);
1975          pConnectInfo = NULL;
1976      }
1977
1978      if (pIfList != NULL) {
1979          WlanFreeMemory(pIfList);
1980          pIfList = NULL;
1981      }
1982
1983      juce::String display = "- Hostname: " + hostname_str + "\n"
1984                             "  - Resolved to " + ip_addr_str + "\n" +
1985                             "- System Connection status: " + connection_str + "\n" +
1986                             "WLAN_CONNECTION_ATTRIBUTES for this interface...\n" +
1987                             "  - Connection status: " + interface_connection_str + "\n" +
1988                             "  - Profile Name: " + profile_name_str + "\n" +
1989                             "  - SSID: " + ssid_str + "\n" +
1990                             "  - BSS Network Type: " + bss_network_str + "\n" +
1991                             "  - MAC Address: " + mac_address_str + "\n" +
1992                             "  - PHY Index: " + phy_index_str + "\n" +
1993                             "  - Signal Quality: " + signal_quality_str + "\n" +
1994                             "  - Receiving Rate: " + receiving_rate_str + "\n" +
1995                             "  - Transmission Rate: " + transmission_rate_str + "\n";
1996
1997      // Update the TextEditor component with the provided text
1998      networkInfo.setText(display, juce::NotificationType::dontSendNotification);
1999      networkInfo.setJustificationType(juce::Justification::topLeft);
2000
2001  }
```

Figure 54: Network Information

As seen, this function does all the logic to get “hostname_str”, “ip_addr_str”, etc, and formats them all into one big gigantic string that use as the text that displays in the “networkInfo” group.

This machine can run EITHER Windows OR Linux.

If it runs Windows, then currently I have the Windows API’s fetching the Network information displaying them, using the following libraries that I’ve imported, in the “MainComponent.cpp” file:

```
// Header files Specific to Windows or Linux
// WINDOWS (used in the getNetworkInformation() function)
#ifdef _WIN32
    #include <Winsock2.h>
    #include <ws2tcpip.h>
    #include <Windows.h>
    #include <wlanapi.h>
    #include <Windot11.h>           // for DOT11_SSID struct
    #include <objbase.h>
    #include <wtypes.h>

    #include <stdio.h>
    #include <stdlib.h>
    // to make the above headers work, you have to link with the below libraries
    #pragma comment(lib, "Ws2_32.lib")
    #pragma comment(lib, "User32.lib")
    #pragma comment(lib, "wlanapi.lib")
    #pragma comment(lib, "ole32.lib")

#endif
// LINUX
#ifdef linux
    #include <unistd.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <string.h>
#endif
```

Figure 55: Lib Importing

Thus, if you’re on a Linux machine, none of the Windows libraries will throw errors when importing, and vice versa.

It’s a future work of this project, to display network information IF you’re on a Linux system. In the “getNetworkInformation()” function, you’d structure it the same way, where all the code I have right now would be encapsulated in “ifdef _WIN32”, and then code for Linux would be added, that would be encapsulated in the “ifdef linux” statement (again, within the actual getNetworkInformation() function).

c. PC GUI Data Graphs

General Structure of the “MainComponent.cpp” file

The “MainComponent.cpp” file has ALL the GUI logic.
It is instantiated by the “Main.cpp”, and it uses the “ComDevice.cpp” (instantiating the ComDevice) class. I organized all the code written into the following blocks (the following pictures are obtained when you condense down the functions):

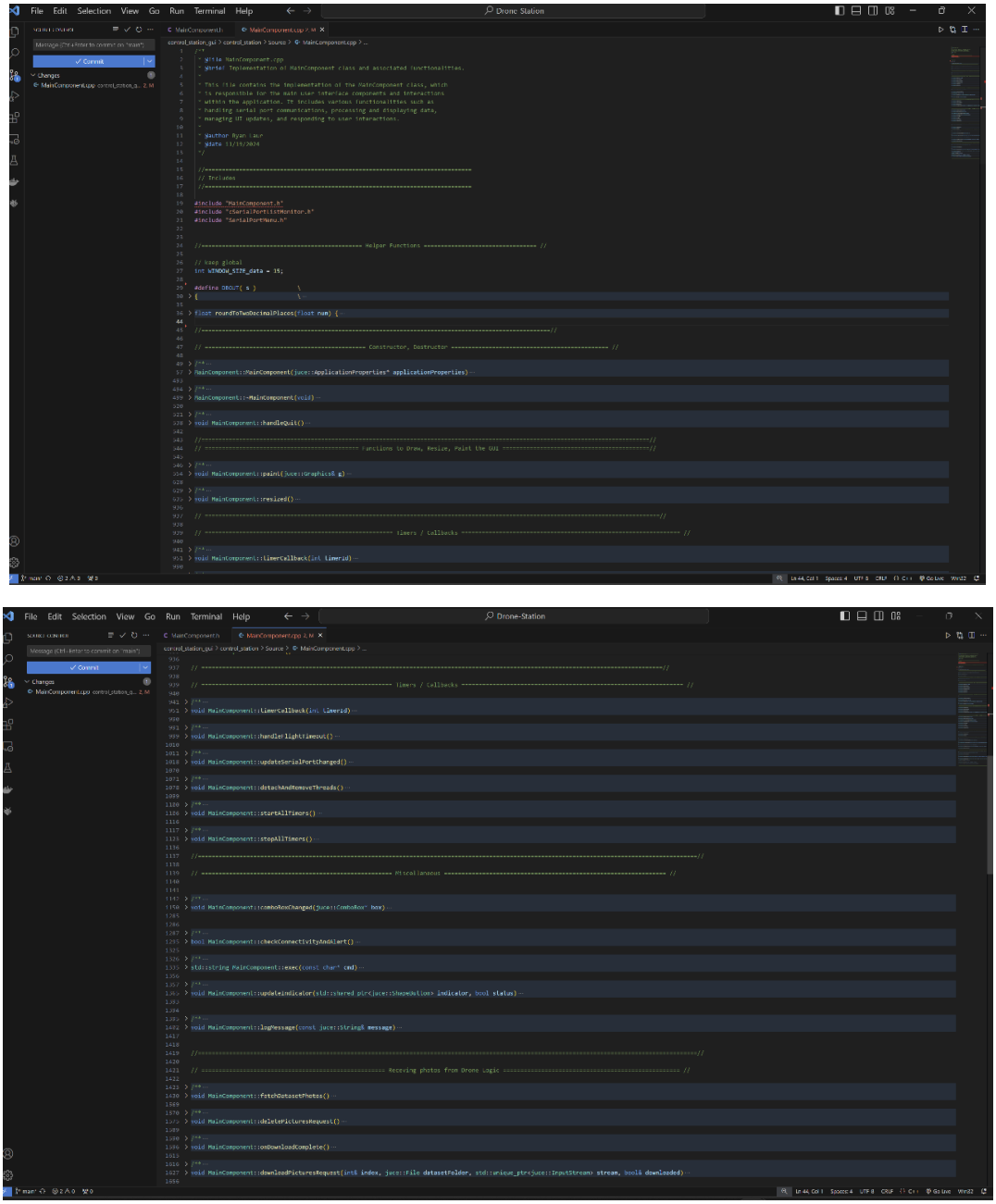


Figure 56-57: GUI Data Graphs

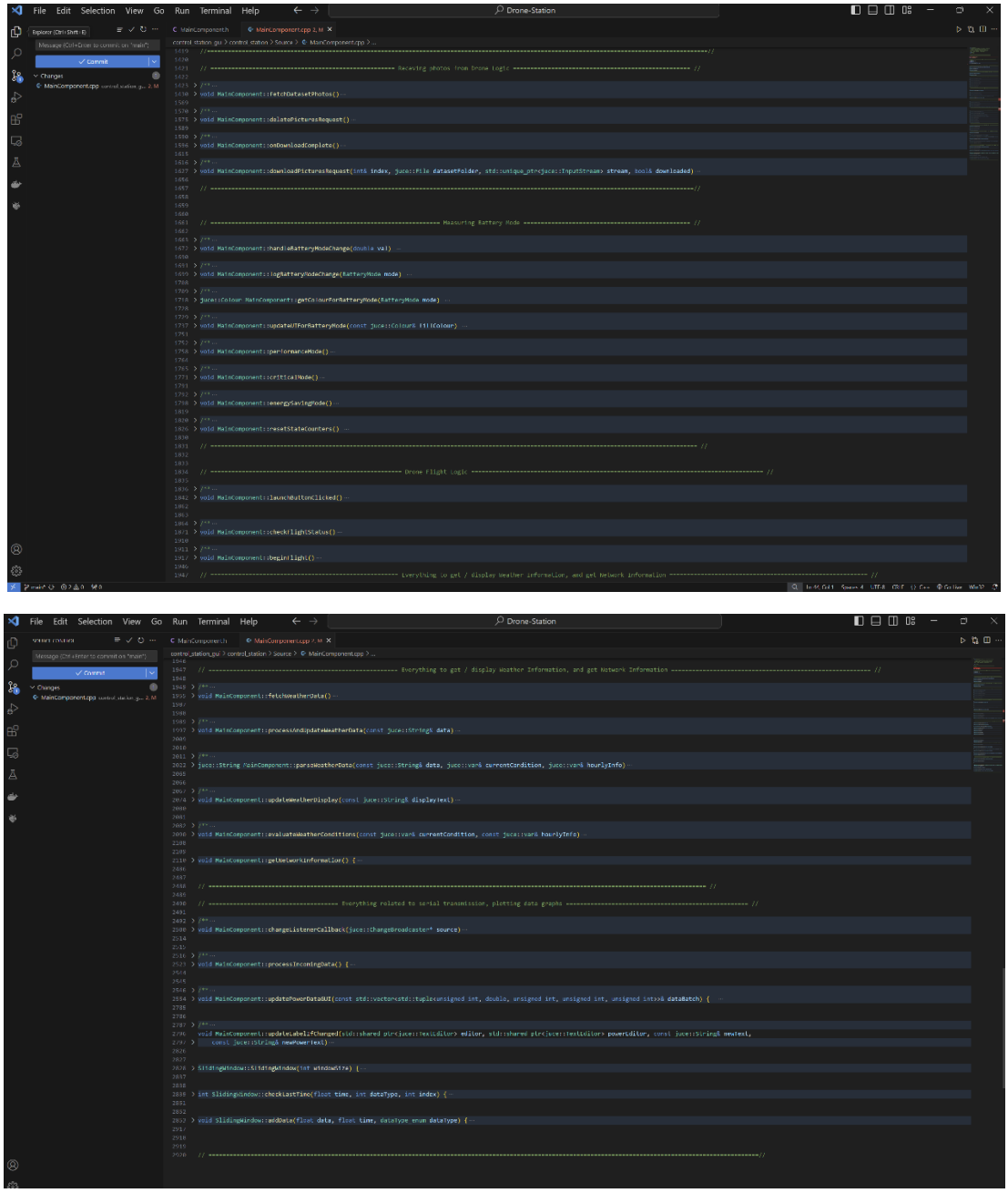


Figure 58-59: GUI Data Graphs

Setting up the External Library to get graphing working:

To have the data graphs running, I used an external library made to display graphs on JUCE projects, from the following repo: https://github.com/franshej/CustomMatPlot . This took me a LONG time to get working, as he doesn't give you good instructions on installation.

On his README, you kind of have to deduce that to use the library, you have to use his method of getting the examples working:

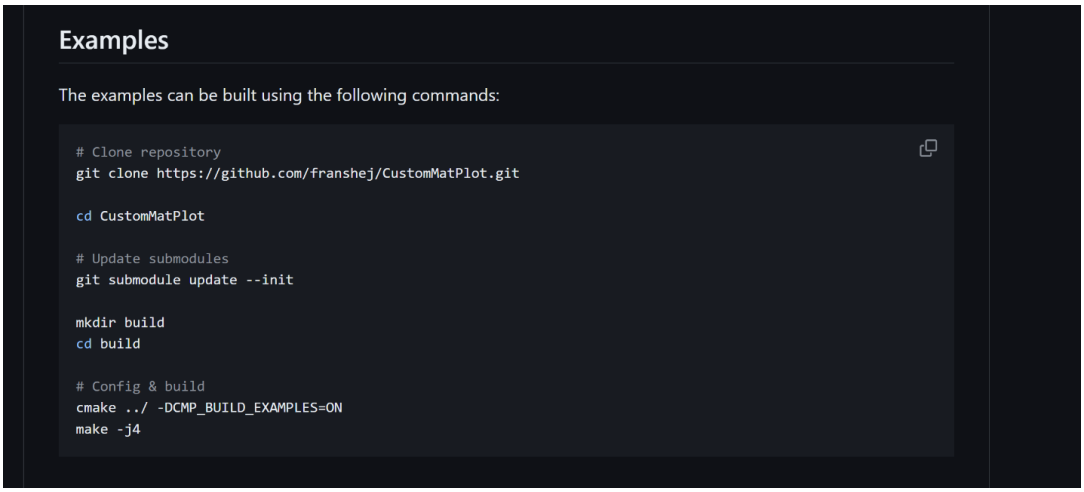


Figure 60: Github Examples for CustomMatPlot

To get this "CustomMatPlot" library working with JUCE, do the following:

- Clone the repo to a folder. I did it in C:/modules.
- Modify 2 lines of code in the source code, otherwise it won't work. And they are:

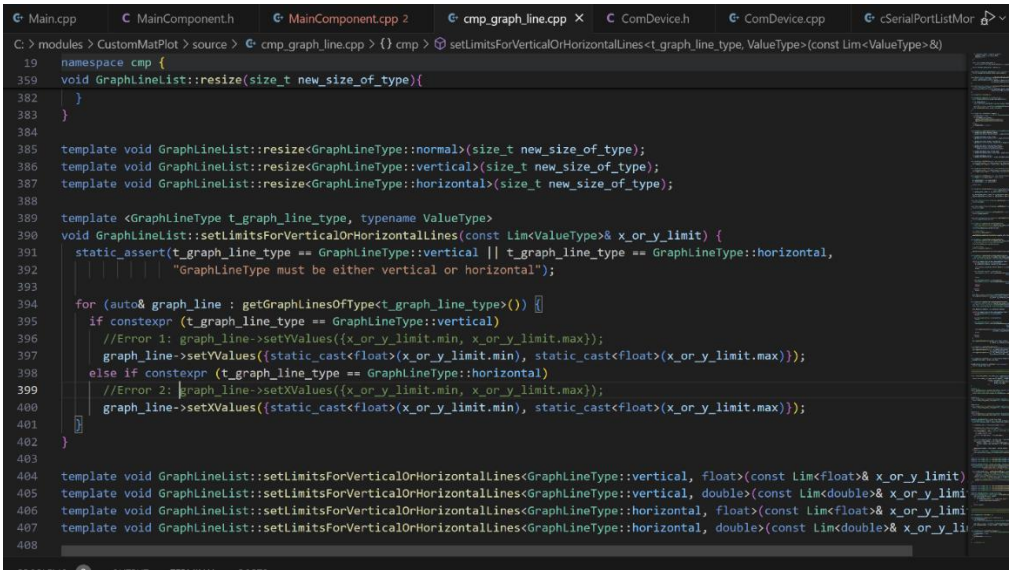


Figure 61: Code Changes

- In the CustomMatPlot/source/cmp_graph_line.cpp file, lines 396 and 399 in the picture above, make the 2 changes seen above (my changes are the code below the 2 commented lines).
- Now, compile this library. First, you run the “git submodule update –init command”.
- Then you make a “build” folder, and cd into it
- Do “cmake ..”.
- This runs the overall “CMakeLists.txt” file in the directory before
- Then you’d usually do “make” like the above picture tells you….But Windows doesn’t have “make”, so you have to do “cmake –build .” (where the “.” IS a part of the command, indicating to put the outputs of this command in the current directory).
- This generates the “build/Debug” folder, in which “cmp_plot.lib” file is located. **I copied this to the C:/modules”** folder, **which is where Visual Studio expects it to be, because this was exported from the PROJUCER, seen below:**

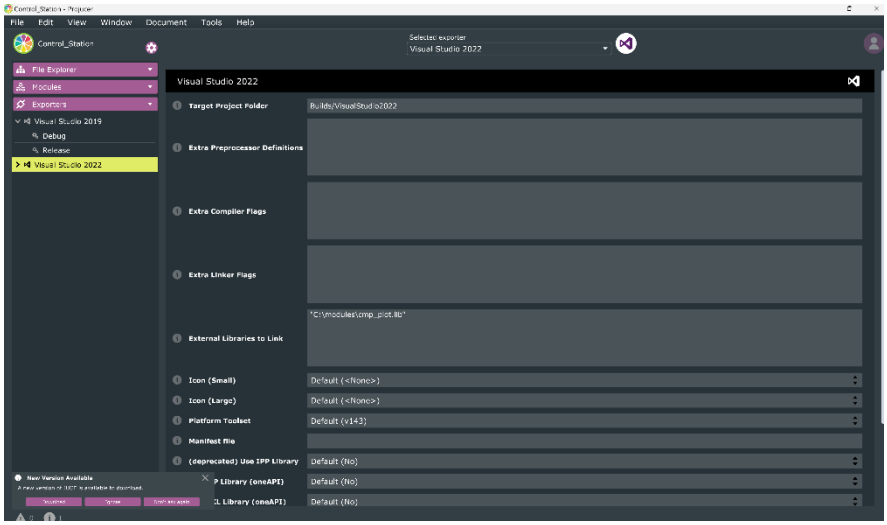


Figure 62: ProJucer Output

Notice how I make the directory of the .lib file to be in C:/modules. Thus, this is where it'll look for the .lib file to be able to compile and run all the CustomMatPlot functions that the GUI code uses.

Thus, my C:/modules folder looks like:

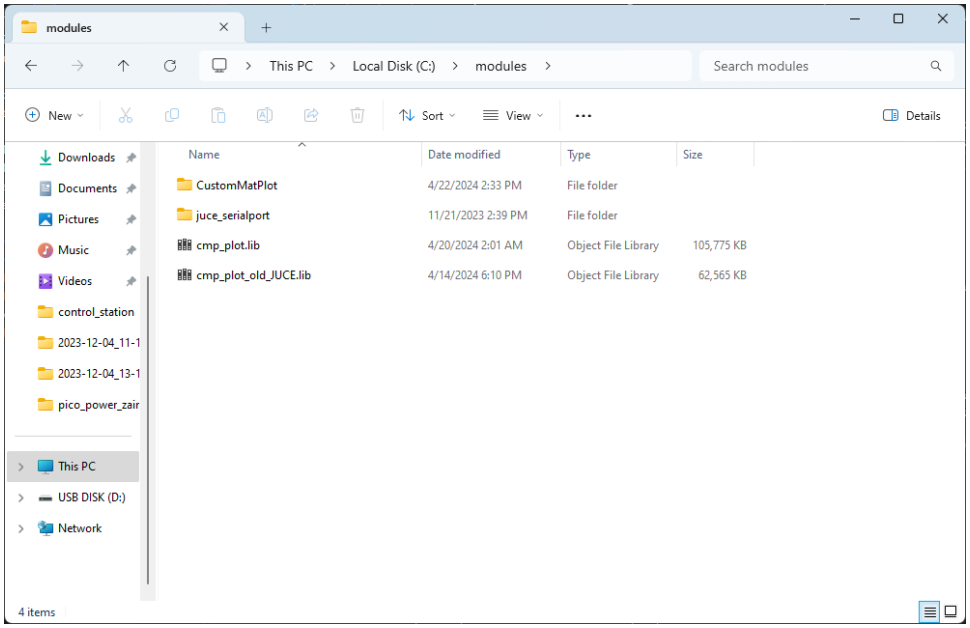


Figure 63: C:/modules folder

Ignore "cmp_plot_old_JUCE.lib", you don't need that.

This ALSO generates a "build/include/include/cmp_version.h" file. **Copy this file, ALONG with all the header files in "C:/modules/CustomMatPlot/include/include** into your Visual Studio's project directory, putting them into a folder called "CustomMatPlot" in the project directory. So if you look in the "control_station_gui/control_station/Source" folder, these headers will be there.

So now in MainComponent.h, I can use the header files like so:

#include "CustomMatPlot/cmp_plot.h"

You also have to do: #include <juce_gui_extra/juce_gui_extra.h>

Finally, set up all the paths in Visual Studio:

In Visual Studio, which opens after you export from Projucer, in Visual Studio, do Project > Properties > Linker > Input, and you'll see the .lib file in "Additional Dependencies, **with QUOTATION MARKS** around it, like- and this was setup in the Projucer so that when you export using it (see above), it'll already configure Visual Studio to run the GUI with all the necessary dependencies.

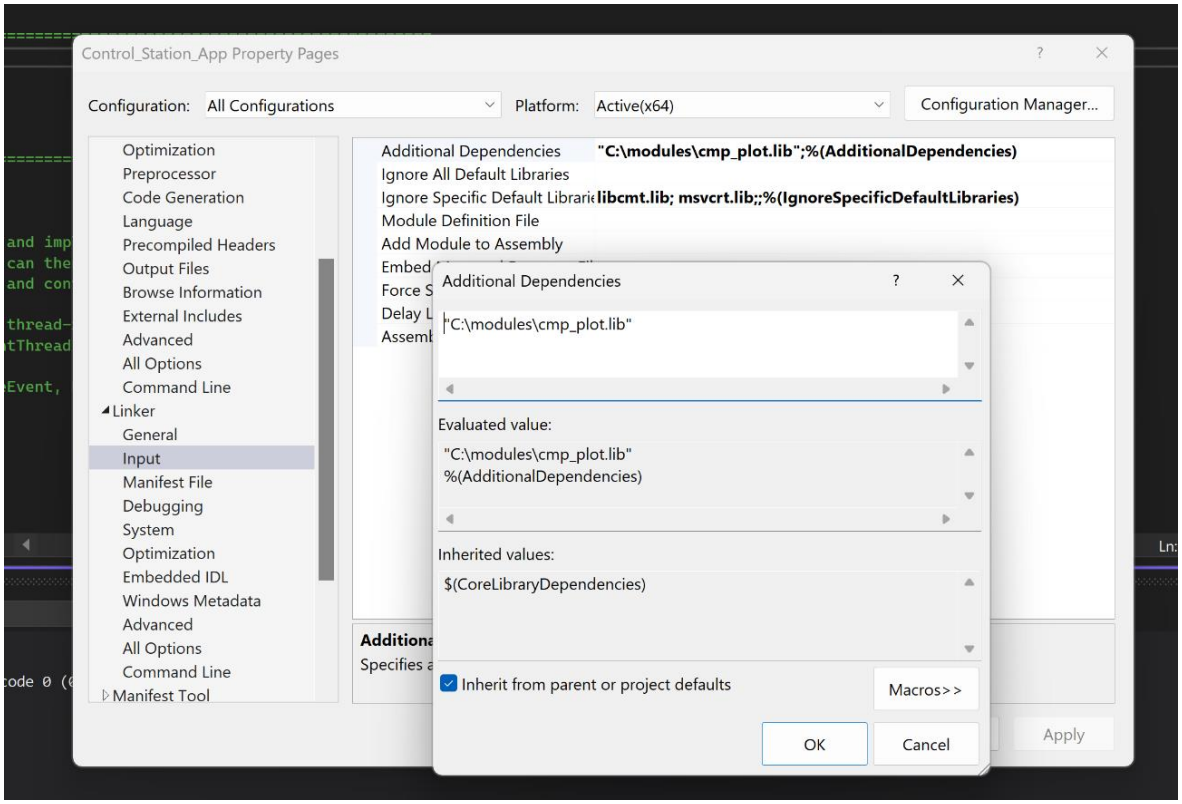


Figure 64: GUI Dependencies

Finally, in Visual Studio > Project > Properties > C/C++ > General, in “Additional Include Directories”, notice how it is:

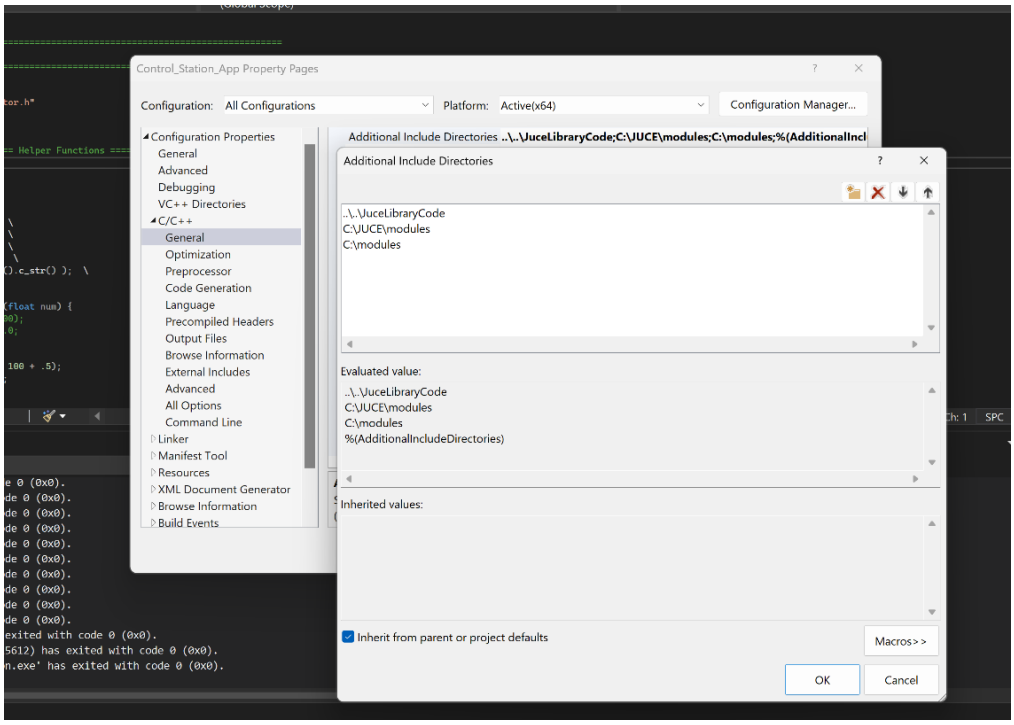


Figure 65: Additional Directories

This is everything you need to get the GUI up and running. It’ll use that JUCE folder you put in the C: directory, and look in the “modules” folder within it.

CustomMatPlot, in its “externals” folder, usually has its own JUCE repository that it uses, but again, I didn’t use that, and just used the one in “C:/JUCE”. But, CustomMatPlot still needs that JUCE folder within “externals/JUCE” so you can generate the .lib file, but again, I’ve already done this, so no need.

You should be able to run this project in Visual Studio now.

Explanation of the code that gets the Graphs working
For each of the rails (ADJ, 5V, 12V, 20V, Battery, Charger, Starlink), there are 3 graphs (Voltage, Current, and Power), so 21 graphs in total which are initialized, as seen below in the “MainComponent.h”:

```
619
620
621     // Declare the actual data we want to graph
622     // pointers because we don't want to instantiate them yet
623     SlidingWindow* data_adj;
624     SlidingWindow* data_20;
625     SlidingWindow* data_12;
626     SlidingWindow* data_5;
627     SlidingWindow* data_bat;
628     SlidingWindow* data_charg;
629     SlidingWindow* data_star;
630
631     // Declare plot objects.
632     // ADJ rail
633         cmp::Plot m_plot_adj_vol, m_plot_adj_curr, m_plot_adj_pow;
634     // 20V rail
635         cmp::Plot m_plot_20_vol, m_plot_20_curr, m_plot_20_pow;
636     // 12V rail
637         cmp::Plot m_plot_12_vol, m_plot_12_curr, m_plot_12_pow;
638     // 5v rail
639         cmp::Plot m_plot_5_vol, m_plot_5_curr, m_plot_5_pow;
640     // Battery
641         cmp::Plot m_plot_bat_vol, m_plot_bat_curr, m_plot_bat_pow;
642     // Charger
643         cmp::Plot m_plot_charg_vol, m_plot_charg_curr, m_plot_charg_pow;
644     // Starlink (PoE Injector)
645         cmp::Plot m_plot_star_vol, m_plot_star_curr, m_plot_star_pow;
646
647     // The data graphs are all sliding windows (because you can't just keep pushing data into them infinitely)
648     // since cmp::Plot plots data from element 0 onwards
649     // To create a sliding window of size n
650     // The class below implements the functionality
651
```

Figure 66: MainComponent.h

But these graphs need data to be plotted. This is done with all the “SlidingWindow” objects, and we can see the class below:

```
437
438 enum dataType_enum { voltage, current, power}; // voltage = 0, current = 1, power = 2
439
440 class SlidingWindow {
441     // You can do a vector of vectors and pass this into a graph to have MULTIPLE PLOTS
442     int index_vol, index_curr, index_pow;
443
444 public:
445     vector<vector<float>> data;
446     vector<vector<float>> time;
447
448     SlidingWindow(int windowSize);
449     void addData(float data, float time, dataType_enum dataType);
450     int checkLastTime(float time, int dataType, int index);
451
452 };
453
454
```

Figure 67: Sliding Window Implementation

So, every instantiation of this class holds the data to be plotted, and the time that corresponds to each data (so we can plot the time the data was measured).

The different data graph windows are setup here (in the “MainComponent.cpp” file):

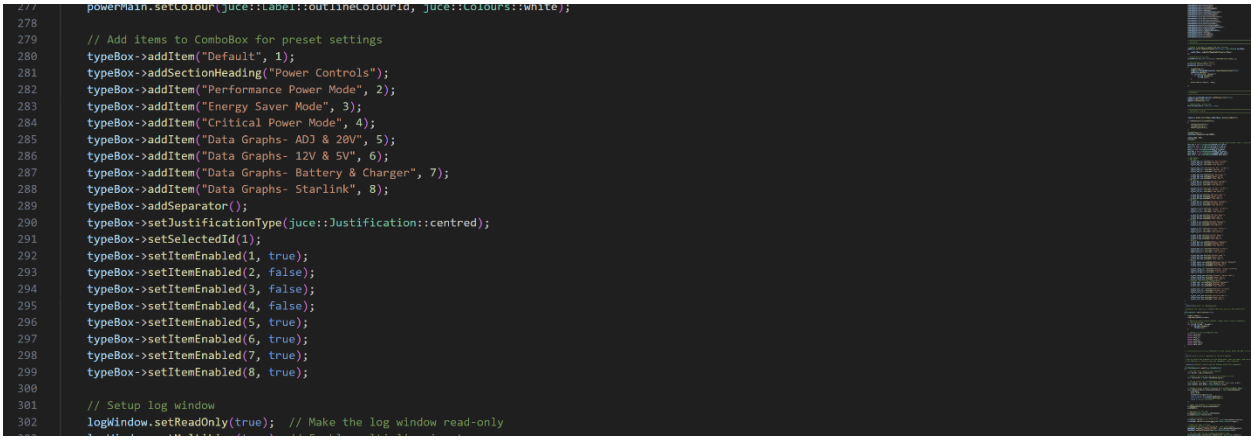


Figure 68: Differentiated Graph Window configuration

The “addItem” adds another window, that when you select the drop down, you can click on these and it’ll switch to a new window.

What holds these windows is the “typebox” object, which is of type “juce::ComboBox”. This is an object that whenever you select the drop down window, it’ll switch the windows. Whenever a mouse event is detected in JUCE, or a mouse click, it’ll call the "MainComponent::comboBoxChanged(juce::ComboBox* box)" function as a callback, and this is where the “switching window” functionality happens:

MainComponent.cpp > comboBoxChanged(juce::ComboBox *)

```
1608  void MainComponent::comboBoxChanged(juce::ComboBox* box)
1642              addAndMakeVisible(batteryStatusLabel);
1643              addAndMakeVisible(*esp32StatusIndicator);
1644              addAndMakeVisible(esp32StatusLabel);
1645              addAndMakeVisible(internetStatusLabel);
1646              addAndMakeVisible(*internetStatusIndicator);
1647              addAndMakeVisible(comStatusLabel);
1648              addAndMakeVisible(*comStatusIndicator);
1649              addAndMakeVisible(railMain);
1650              addAndMakeVisible(voltageMain);
1651              addAndMakeVisible(currentMain);
1652              addAndMakeVisible(powerMain);
1653              for (int i = 0; i < 7; i++) {
1654                  addAndMakeVisible(voltageLabels[i]);
1655                  addAndMakeVisible(*voltageValues[i]);
1656                  addAndMakeVisible(*currentValues[i]);
1657                  addAndMakeVisible(*powerValues[i]);
1658              }
1659              break;
1660          case 2: performanceMode();
1661              break;
1662          case 3: energySavingMode();
1663              break;
1664          case 4: criticalMode();
1665          case 5: // ADJ and 20V
1666              // First remove all components
1667              removeAllChildren();
1668              // First, re-add the necessary stuff
1669              addAndMakeVisible(*typeBox);
1670              addAndMakeVisible(titleLabel);
1671              addAndMakeVisible(NameLabel);
1672              addAndMakeVisible(descriptionLabel);
1673              addAndMakeVisible(quitButton);
1674              addAndMakeVisible(comPortLabel);
1675              // Make graph visible
1676              // ADJ Rail
1677                  addAndMakeVisible(m_plot_adj_vol);
1678                  addAndMakeVisible(m_plot_adj_curr);
1679                  addAndMakeVisible(m_plot_adj_pow);
1680              // 20V Rail
1681                  addAndMakeVisible(m_plot_20_vol);
1682                  addAndMakeVisible(m_plot_20_curr);
1683                  addAndMakeVisible(m_plot_20_pow);
1684              break;
1685          case 6: // 12V and 5V
1686              removeAllChildren();
1687              // Necessary Stuff
1688              addAndMakeVisible(*typeBox);
1689              addAndMakeVisible(titleLabel);
1690              addAndMakeVisible(NameLabel);
1691              addAndMakeVisible(descriptionLabel);
```

Figure 69: Fixes to operation

This is seen above- where if, lets say the program enters “case 5”, then in that case, first all the components are removed (the children of Main.cpp, or everything in MainComponent). Then, everything that’s necessary is added back (like the Agribugs logo, to the typebox being there so you can still switch the window, etc). Finally, the necessary graphs for this window are made visible.

If we look in the following picture (from Section A):

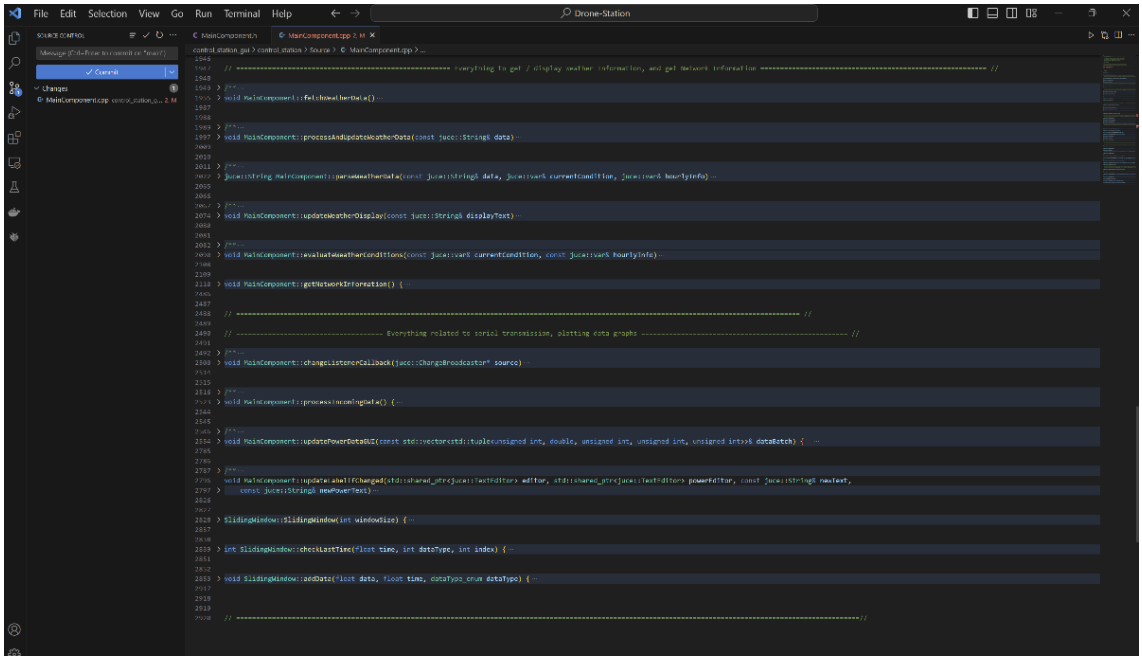


Figure 70: Serial Transmission

In the “Everything related to serial transmission, plotting data graphs”, you’ll see all the functions that deal, start to end, with receiving the data + the time from the Pico, and plotting it in the data graphs among the different windows. If you want to learn more about these functions, read the documentation that I’ve written at the top of them.

C. Modifying the Raspberry PI firmware to send RTC (real time clock data) as well as extra data for the PoE injector ADC measurements.

With Dieter and Ryan’s work from the year before, they only sent the 1) ID, and 2) value over serial transmission, where the ID of each value is where it comes from in the following data structure in the “ADC.py” file:

```
def poll_adc_channels():
    """
    Perform a conversion on each ADC channel for each GPIO Switch channel.

    The function iterates through the switch pins (GPIO18-21) that controls the attached Analog multiplexer IC's.
    For each active switch, an ADC can take a reading and store the data. We perform this to fill 12 readings.

    Every time the method is called, GPIO22 is toggled so that we swap between mux3 and mux4 of the Analog multiplexer IC's.
    This alternating means that one of the four returned data structures will be empty as a result.

    :return: Returns four Dictionaries, one for each group of multiplexer readings.
    :rtype: tuple(dict)
    """
    global mux_toggle
    global ADC_OFFSET
    global CURR_OFFSET
    global voltage_scalars
    global current_scalars
    global mux3_scalars
    global mux3_offsets
    global mux4_scalars
    global mux4_offsets

    # Set the timeout function to be called after the specified duration
    #timeout_timer.init(mode=Timer.ONE_SHOT, period=5000, callback=timeout_handler)

    '''
    voltage_dict = {}
    current_dict = {}
    mux3_dict = {}
    mux4_dict = {}
    '''

    voltage_dict = OrderedDict()
    current_dict = OrderedDict()
    mux3_dict = OrderedDict()
    mux4_dict = OrderedDict()


    mux_1_order = ['ADJ', '20V', '12V', '5V']
    mux_2_order = ['ADJ', '20V', '12V', '5V']
    mux_3_order = ["VBAT", "IBAT", "ICHARGER", "MISC"]
    mux_4_order = ["VPOE", "IPOE", "M3", "M4"]



    # Iterate through the Mux Switches
    for index in range(4):

        # Set the desired GPIO high while others are low. Read each ADC channel.

PROBLEMS  OUTPUT  TERMINAL  PORTS
TERMINAL
```

Figure 71: ADC.py

We see “mux_1_order” up to “mux_4_order”, where each value corresponding to this in the dictionary will have its own ID, which gets sent over serial transmission.

What we added for our project, is we replaced “M1” and “M2” in mux_4_order with “VPOE” and “IPOE”, which gets sent over serial transmission to the GUI for plotting the Starlink’s PoE injector’s voltage and current.

Finally, with each data point, the Raspberry Pi’s Real Time Clock (RTC) is utilized- whenever the data is sent, the RTC is called, getting the time at that moment, and sending this over serial along with the ID and value. The RTC sends the hour, minute, and second. The code for this serial transmission is seen below:

```
def send_serial_data(sys, id_value, value, hour, minute, second):
    """
    Sends data over UART. ID first, followed by value.

    :param uart: UART communication object
    :type uart: uart
    :param id_value: ID for the type of data.
    :type id_value: int
    :param value: The data value.
    :type value: float | int
    """

    value = int(value * 1000)
    #print(value)

    combined_bytes = id_value.to_bytes(1, 'big') + value.to_bytes(2, 'big') + hour.to_bytes(1, 'big')  + minute.to_bytes(1, 'big')  + second.to_bytes(1, 'big')

    # Convert ID and value to bytes and send
    sys.stdout.buffer.write(combined_bytes)
    time.sleep_ms(20)
```

Figure 72: RTC Data Structures

The order of data being sent is: First, ID, then value, then hour, minute, and second. Hour, minute, and second only require 1 byte, since their maximum value's are 23, 59, and 59, which is less than 256.

Within the GUI code, this is read in the following function in "ComDevice.cpp":

```
524  /**
525   * @brief Adds the read data to internal buffers.
526   *
527   * This method processes the data read from the serial port and stores it
528   * in internal buffers for later retrieval and use.
529   */
530  void ComDevice::addDataToBuffers() {
531      unsigned int id = getReadid();
532      double value = getReadVal();
533      // Time
534      unsigned int hour = getReadHour();
535      unsigned int minute = getReadMinute();
536      unsigned int second = getReadSecond();
537
538      if (isBufferOverrun())
539      {
540          // Handle buffer overrun
541          DBG("Buffer overrun detected in ComDevice");
542          handleBufferOverrun();
543      }
544      else
545      {
546          idBuffer.push(id);
547          valueBuffer.push(value);
548          // Time
549          hourBuffer.push(hour);
550          minuteBuffer.push(minute);
551          secondBuffer.push(second);
552      }
553  }
```

Figure 73: ComDevice.cpp

This function is called in ComDevice::run(), which is automatically is the default callback what's constantly looped through over and over when the thread runs.

Results

I. Power Board V1.1

a. Successes

Some of the things we got right with the SMD board were size reduction and board neatness. The EVMs ended up transitioning surprisingly well with moderate intervention to the reference designs. We got a reduction in board size from 12”x12” to 12”x8” (~33% area decrease). Moreover, all external reverse polarity protection and polarity detection boards have been mounted, along with display, controls, and 4X H-bridge driver array for dome control, charging, and other functions.

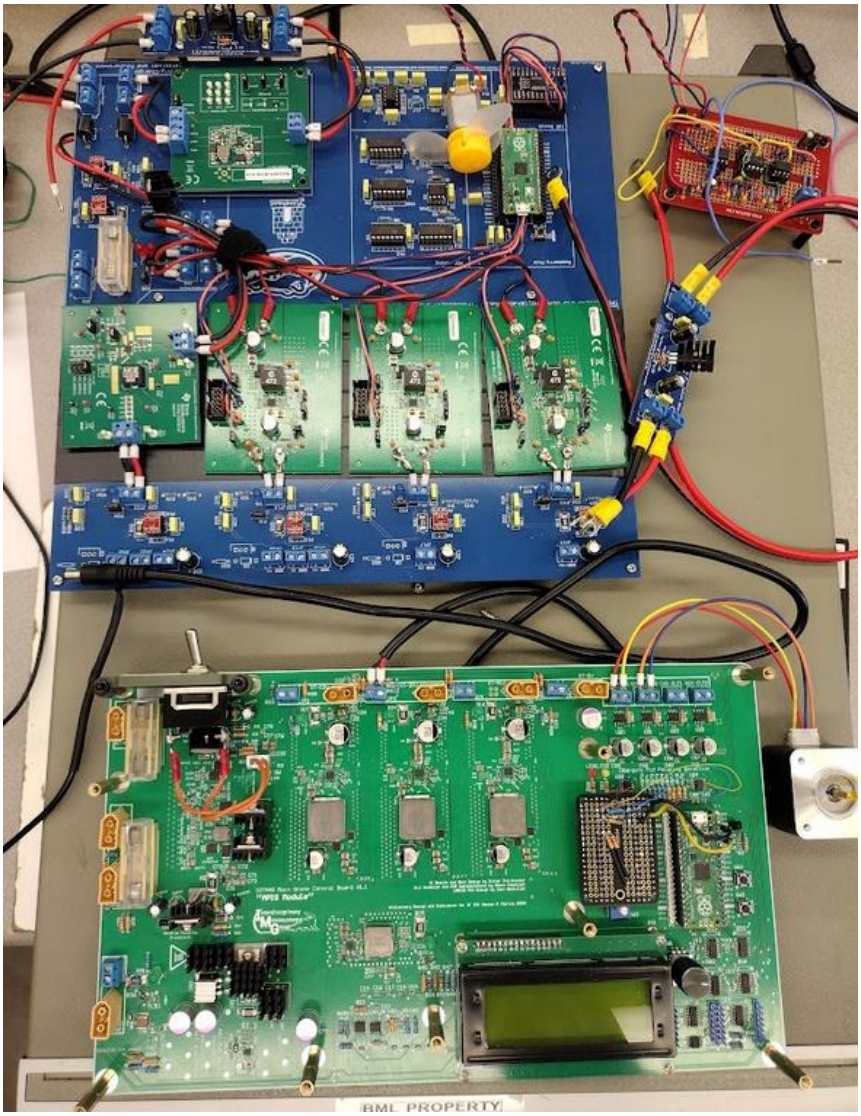


Figure 74: V1.0 (above) and V1.1 (below)

The addition of an onboard 3D-printed mount for the power up switch makes the board so much more mobile, with way less in the way of physical devices that hang off.

Separately, the PoE supply, as well as 2 of the 3 I2C boost/buck modules are functioning (covered later in failures). The H-bridge drivers, display, controls, and other modules are functioning. The PoE supply has both voltage monitoring, as well as current monitoring with a drop-in replacement to the INA181 current sense amplifiers used elsewhere in the design. The reason is for the higher common-mode voltage tolerance, as well as the overall higher voltage rating required for this supply.

The original ADC non-ideality issue has been permanently resolved by placing the analog-mux driven voltage follower in the analog measurement circuitry.

C19, originally replaced for a load resistor due to board capacitances, has been completely fixed with the new component placement. Parasitic capacitances are dramatically reduced with smaller SMD ICs and shorter wiring runs.

Power Consumption
At full power, with starlink active, and no PC attached, we measured a peak current draw of 12A at 12V, an astounding 144W of usage. With the efficiencies of the boost/buck converters at 85%, a 0.3V voltage drop across the PFETs and Schottky diodes at the input of the power rails, and waste heat at the inductors, we can assume a nominal efficiency of around ~70%. For a general proof-of-concept, I consider this rating to be decent. However, at idle, power consumption is around 150mA at 12V, which is substantial. Keeping in mind that this is without the BeeLink connected to the system.

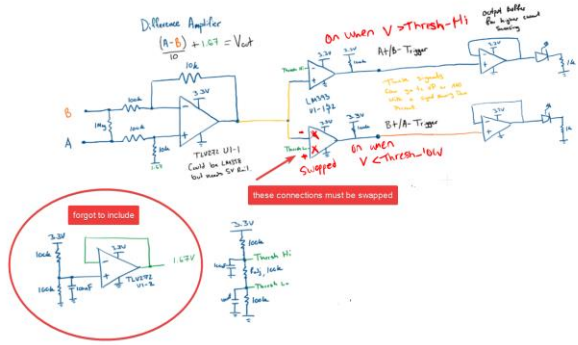
The consensus here is that one of 3 options be exercised in the future:

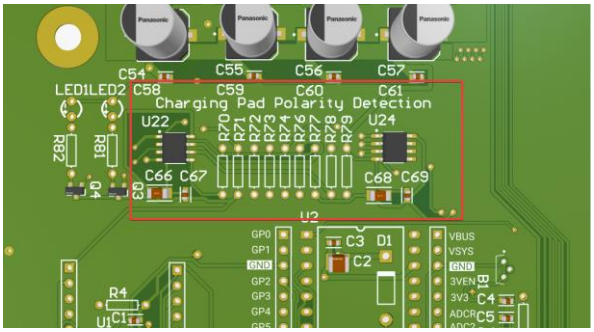
- Switch the system out for a 24V battery supply to decrease current, therefore losses in conductors to heat.
- Decrease load, by either having the starlink be switchable in the future, or removing it.
- Increase supply, by adding more solar panels, auxiliary charging, or larger battery reserve.

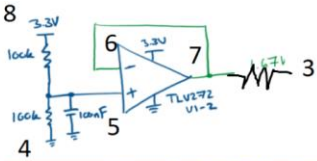
b. Failures and Fixes

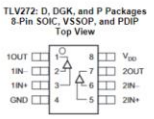
This section will be enumerated in a table to improve readability:

1. The TLV272 and LM393 ICs in the polarity circuit are both missing the 1.67V reference, as well as the pins on the thresh_low side being swapped. This was also the case in the previous report and has been remedied on the board as a bodge.



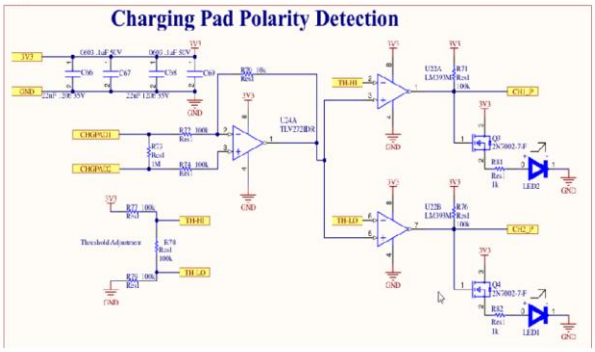


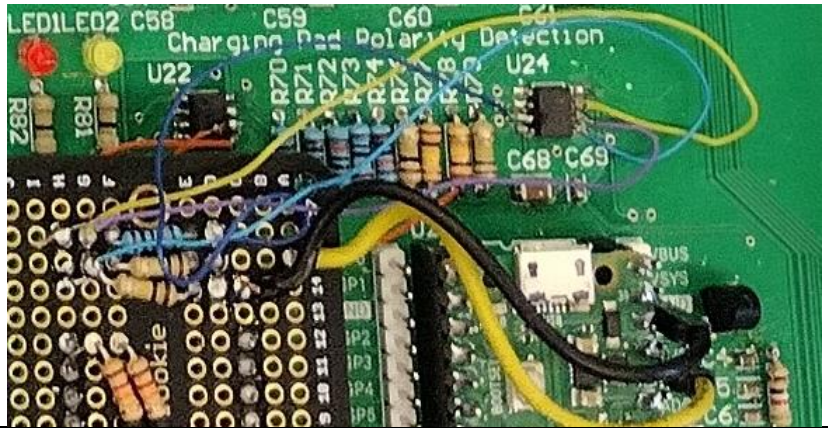




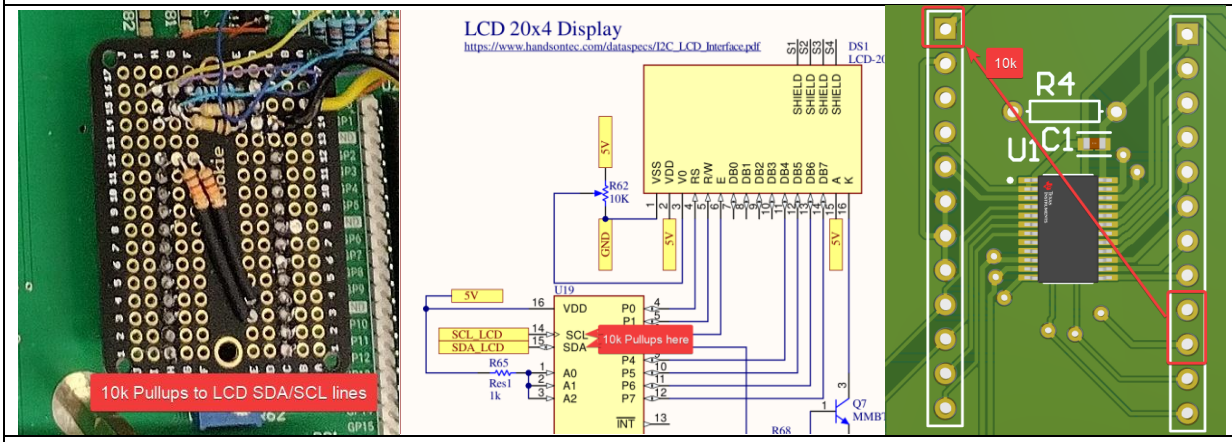
Pin Functions

PIN NAME	PIN TLV272 SOIC VSSOP PDIP	I/O	DESCRIPTION
GND	4	—	Negative (lowest) supply or ground (for single-supply operation)
1IN−	2	I	Inverting input, channel 1
1IN+	3	I	Noninverting input, channel 1
2IN−	6	I	Inverting input, channel 2
2IN+	5	I	Noninverting input, channel 2
1OUT	1	O	Output, channel 1
2OUT	7	O	Output, channel 2
V_{DD}	8	—	Positive (highest) supply

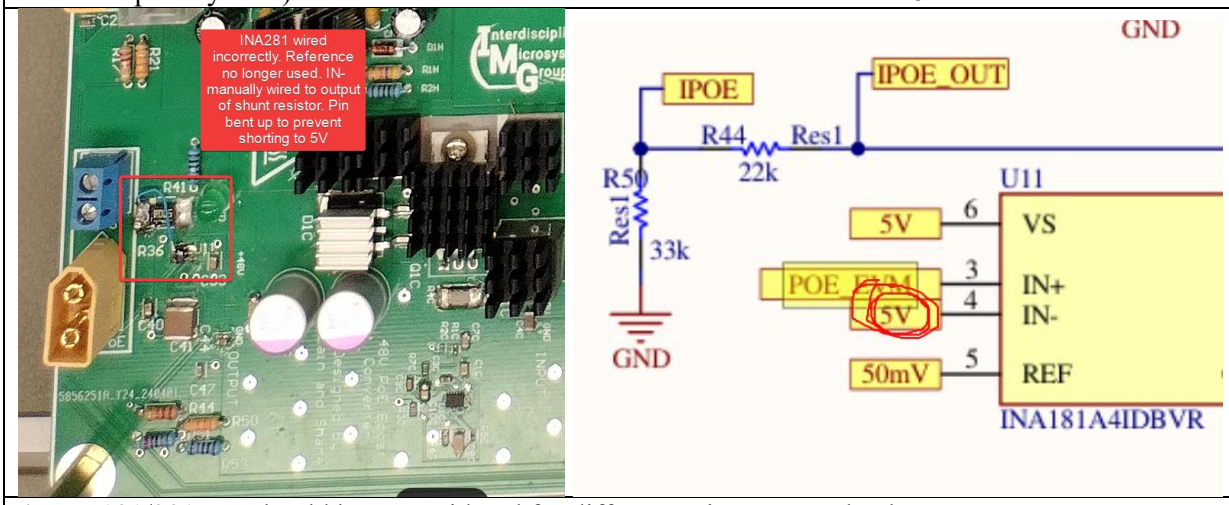




2. The I2C I/O expander for the LCD addition REQUIRES 2 10k Pullup resistors on the SDA/SCL lines. This was not an issue for the boost/buck ICs because their modern design accounts for some weak pullup internal to the silicon.



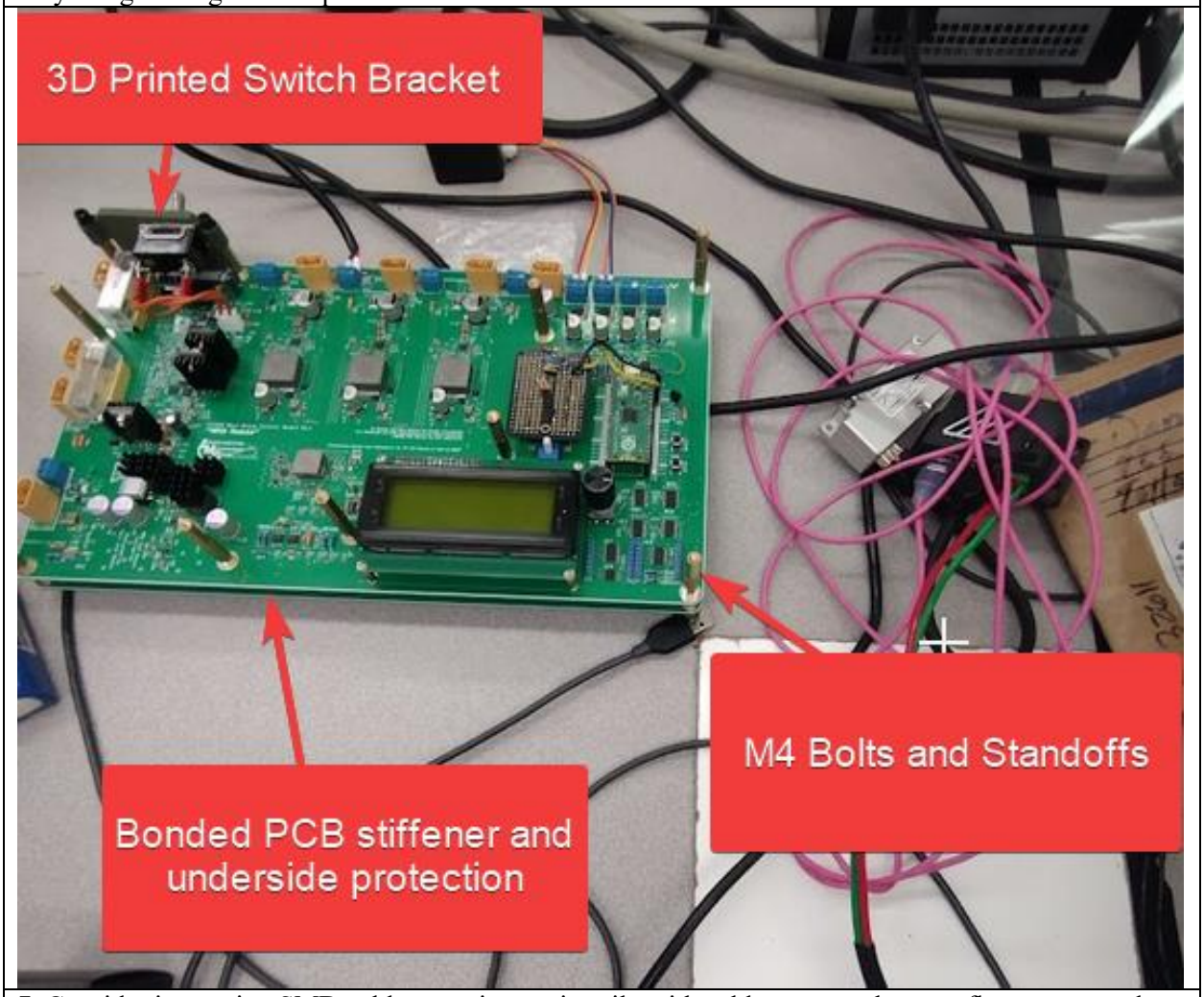
3. The wrong IC (INA181) was used for 48V I sense. Replace with INA281 (pin compatible, without reference pin anymore). Schematic needs current sense node fixed to not be 5V.



4. INA181/281 ICs should be reconsidered for different gains. Currently chosen current sense amplifiers labelled in BOM/Assembly sheet at end of the report

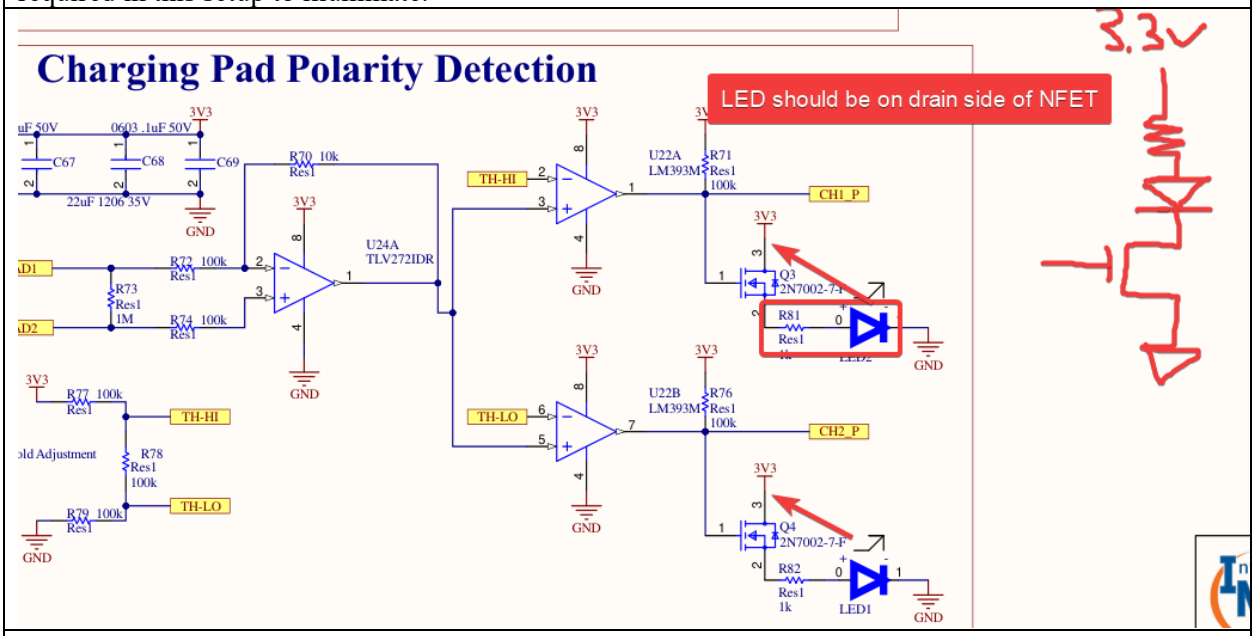
5. Not enough time to check for this semester, but the battery and charge sense amplifiers should be tested for accuracy and range.

6. Due to the overall size of the board and difficulty of placement, we lost one of our I2C buck/boost supply modules. This was primarily due to flexion in the board during handling and assembly, as well as possible excess solder underneath the packages. The current solution was to bolt a spare PCB to the bottom with M4 nuts and bolts, as well as some 3D-printed parts to add rigidity and protect the underside from shorts. In the future, absolutely consider switching to 4 layer and shrinking size to ideally 8”x8” or less, or place small power modules onto PCB cards that can be slotted in the top for easy design changes and replacement.

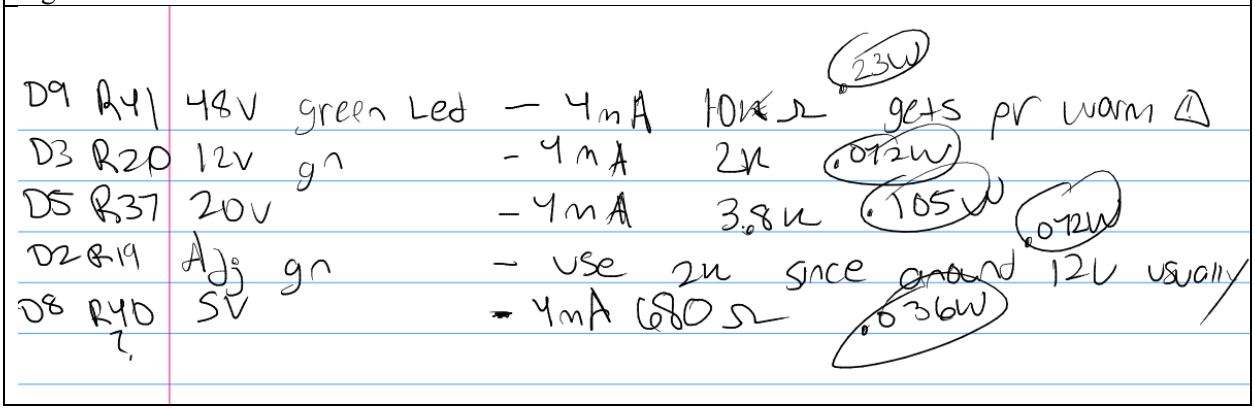


7. Consider improving SMD solder practices, primarily with solder paste volume, reflow curve, and reflow device. The reflow oven used for this project barely fit the board, so it was possible that uneven heating was a genuine issue that greatly affected the reliability of QFN packages like the boost/buck ICs.

8. The NMOS FETs on the Ch1 and Ch2 polarity pins have the LED on the source side rather than drain. Although not a detrimental issue, does increase the Vth and effective Ron. Low Vf LEDs are required in this setup to illuminate.



9. Indicator LEDs for voltage rails waste heat for higher voltages, since only a resistor is used to regulate current thru each LED.



II. Starlink

a. System Setup

The starlink router is powered by the board and can search for constellations. Once connected, the power consumption does drop significantly. When setting up, it's critical to enable bypass mode in either the Starlink app or the Starlink web UI. Once bypass mode is enabled, any router can be used to manage your network.

b. Web UI

Connecting to the Satellite to control the stow function or connection information is driven thru dishy.starlink.com in your web browser. The DNS is resolved automatically, redirecting you to the proper IP address on entry.

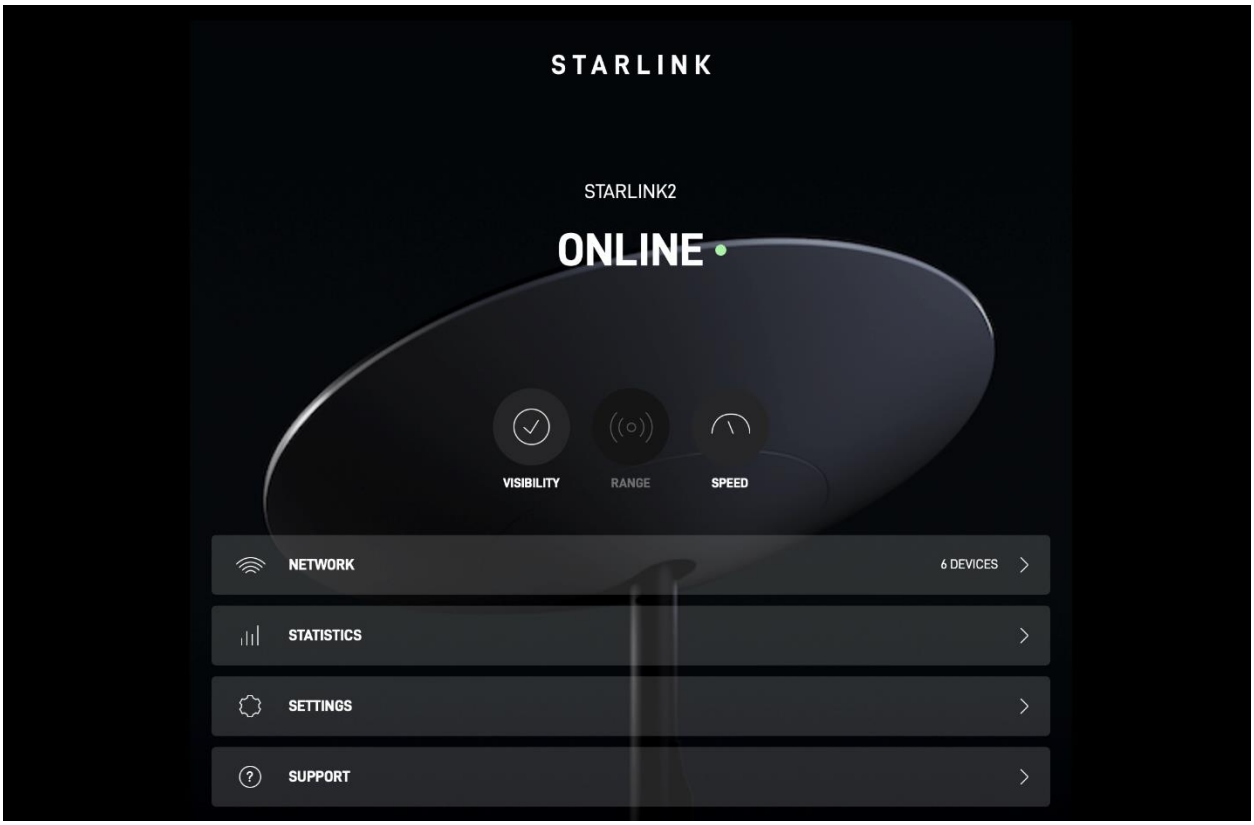


Figure 74: Starlink WebUI

c. Notes

Proper line-of-sight to the Starlink constellations is extremely imperative for reliable operation. Management thru the web UI 'dishy.starlink.com' is a better experience, overall.

III. GUI

The GUI turned out perfectly, being able to receive data over serial transmission from the Pi Pico and display it on the main, "default" page of the GUI, seen below:

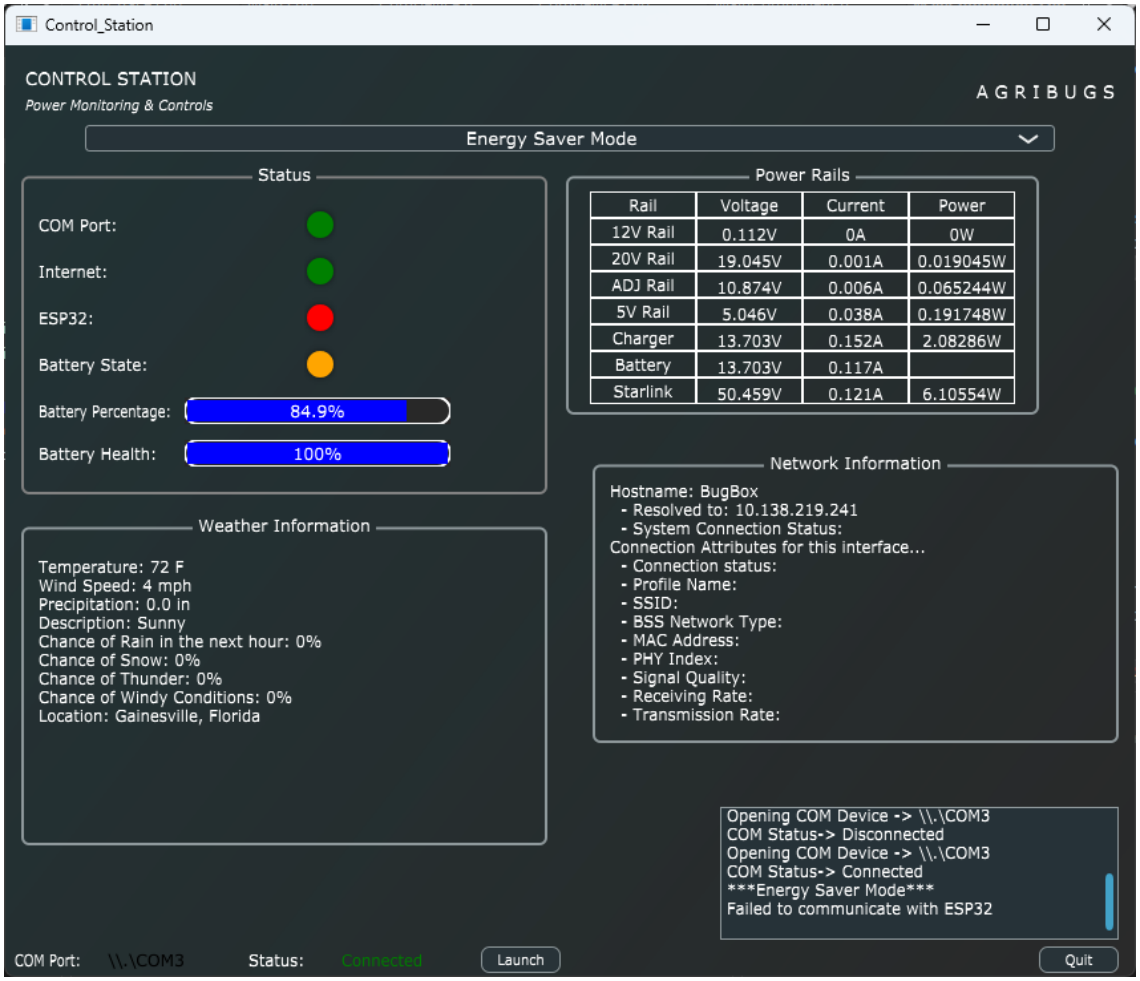


Figure 75: New Control GUI

This shows the fully functioning front page of the GUI. The table at the top right shows the different values received over serial transmission, updating real-time.

This also shows the "Network Information" panel, which isn't updating anything because the timer that updates it in software hasn't been reached yet, so currently nothing is being displayed. But this DOES work.

When you open the combo box, it reveals a drop-down where you can select the different windows, seen here:

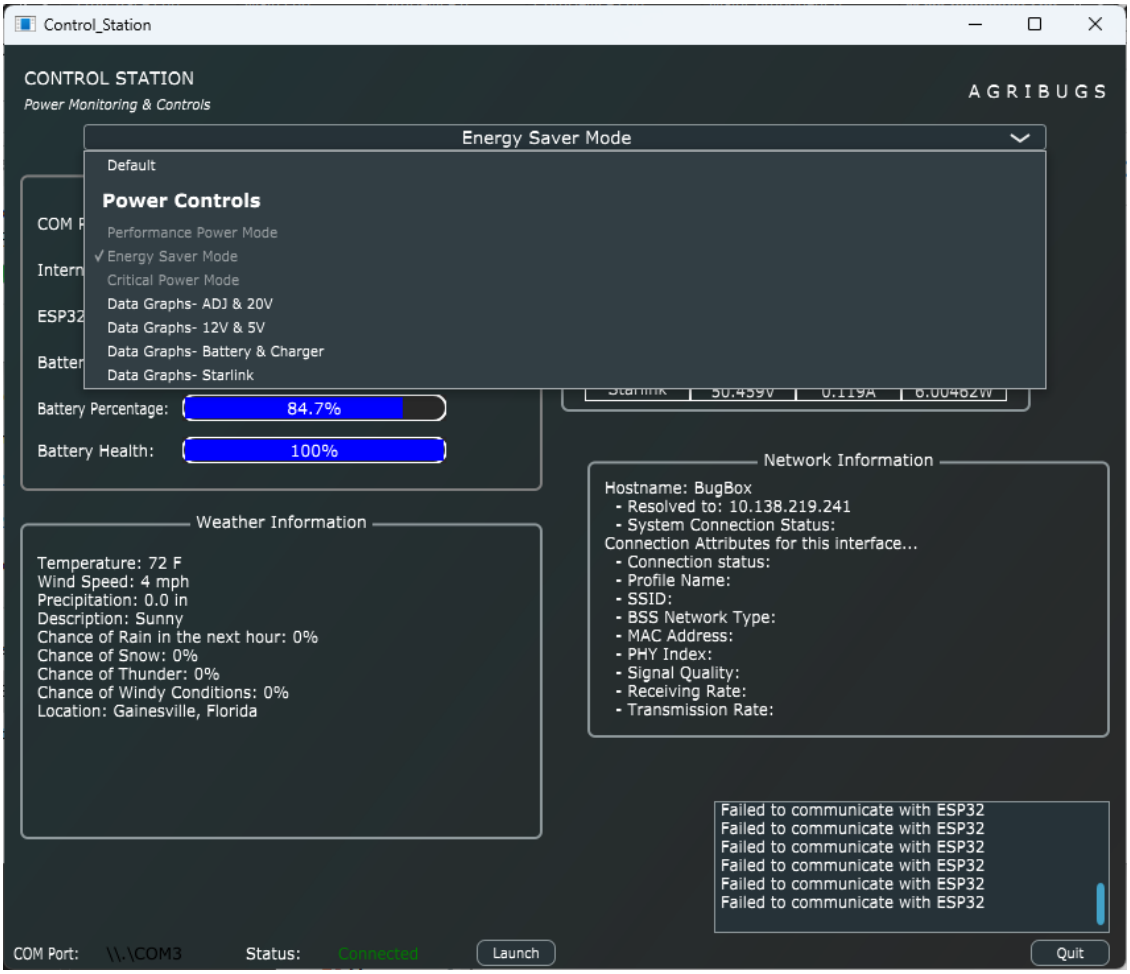


Figure 76: Dropdown Menus

The different windows look like:

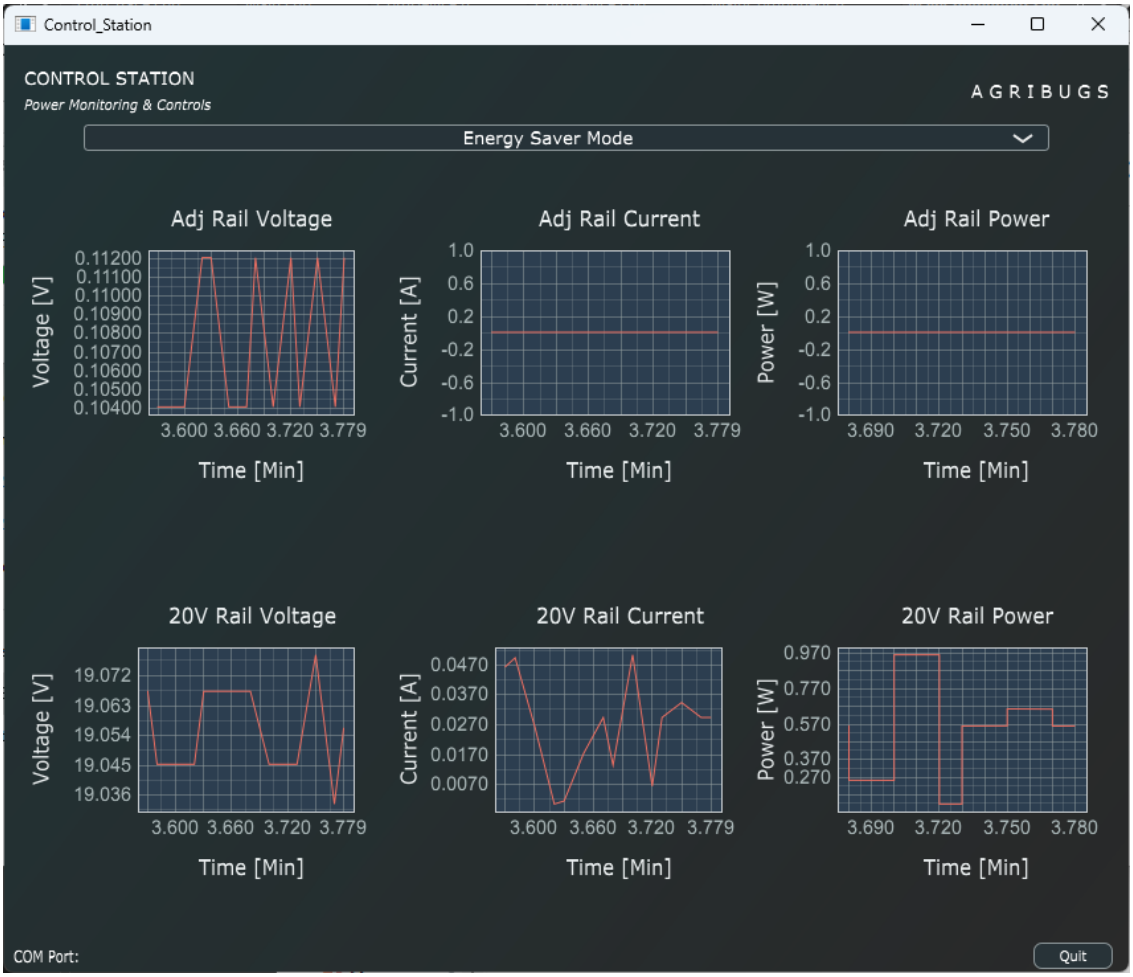


Figure 77: Graphs (2)

For some reason, it says this window is “Energy Saver Mode”, but this window was displayed after I clicked the drop down option called “Data Graphs- AJD & 20V”.

The x-axis is calculated using the following formula in code: (hour * 60) + minute + (seconds / 60). Thus, the x-axis displays the total minutes that have gone past, every since the Raspberry Pi was powered up. This formula is in the “updatePowerDataGUI()” function.

Likewise, the rest of the windows are:

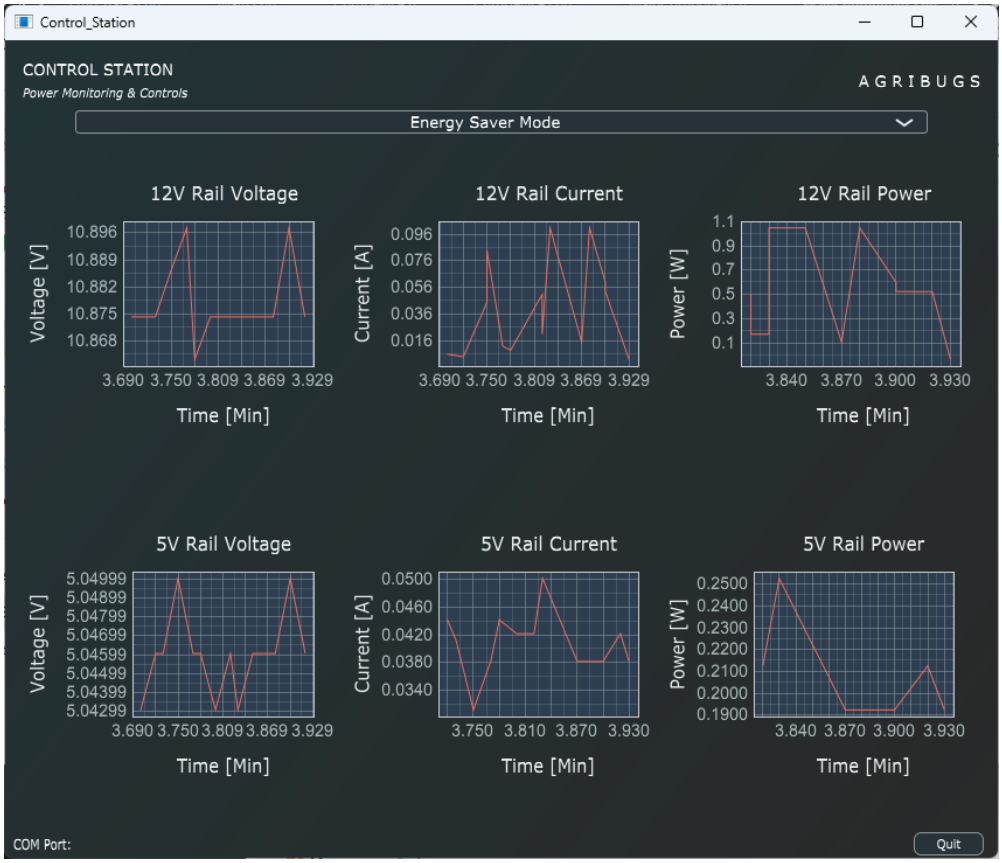


Figure 78: Graphs (2)

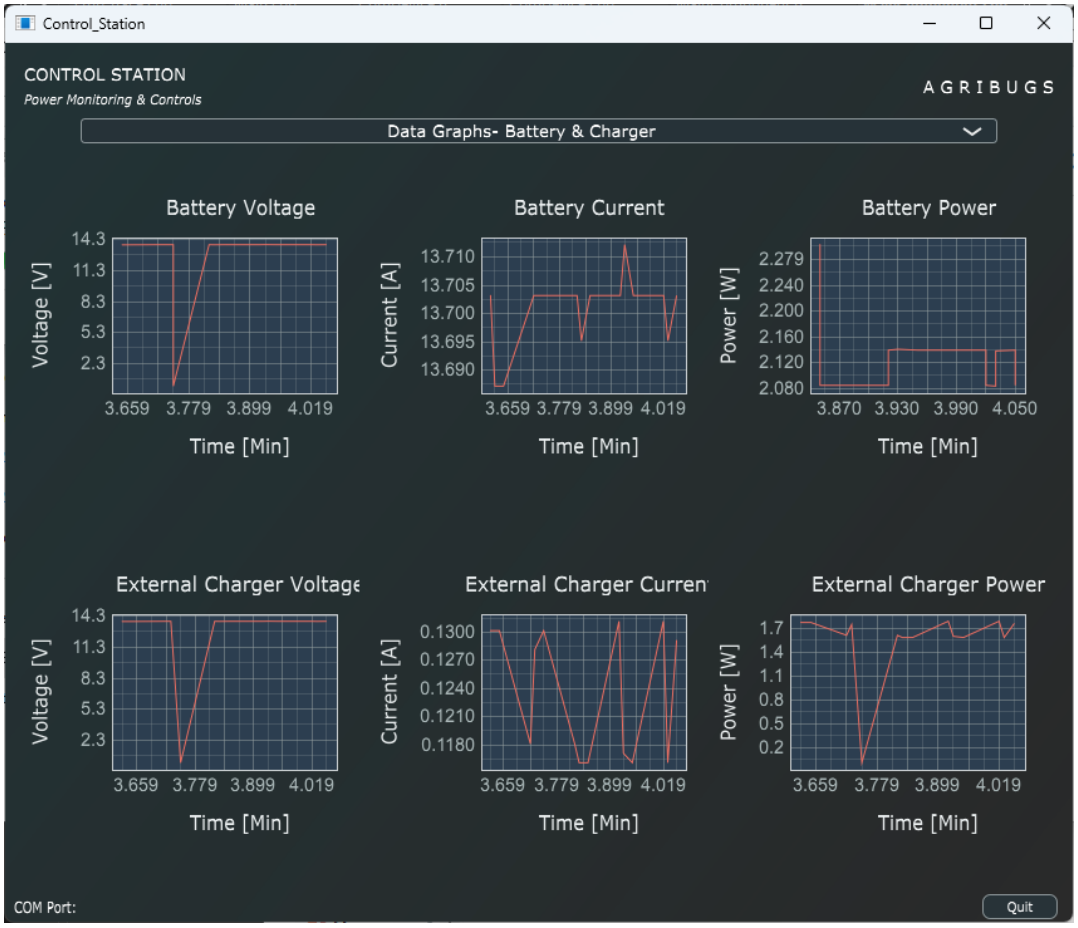


Figure 79: Graphs (3)

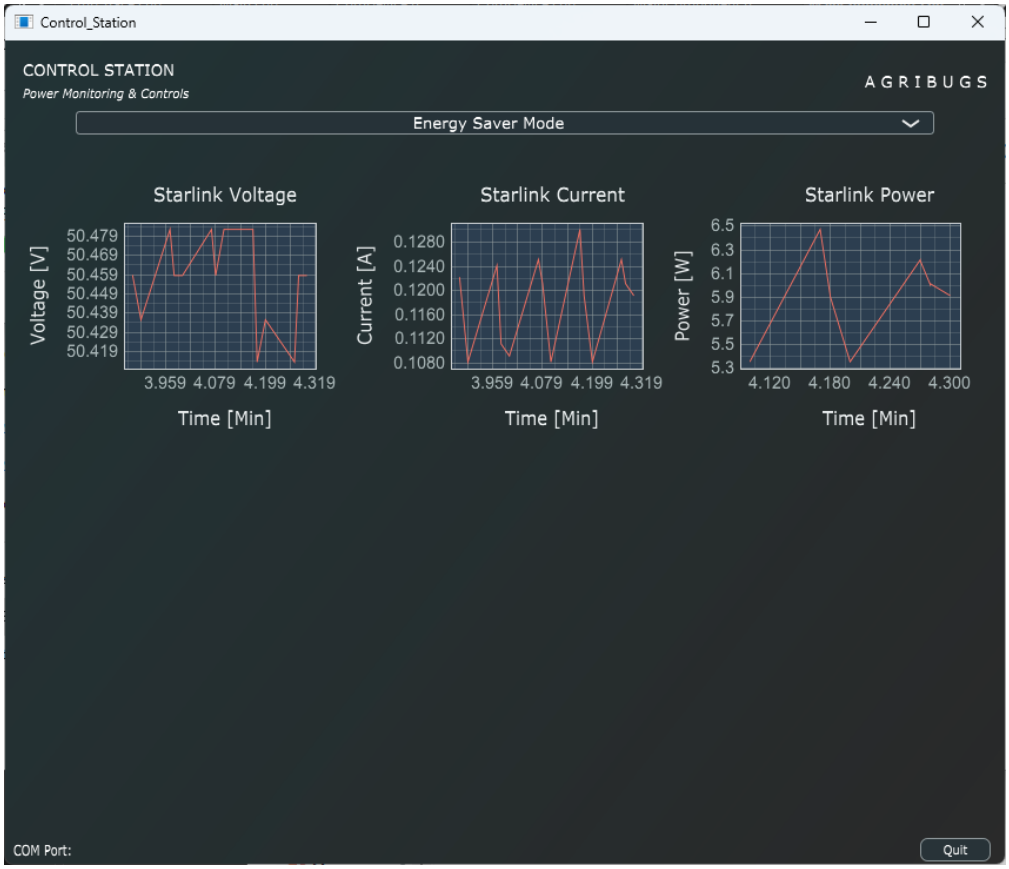


Figure 80: Graphs (4)

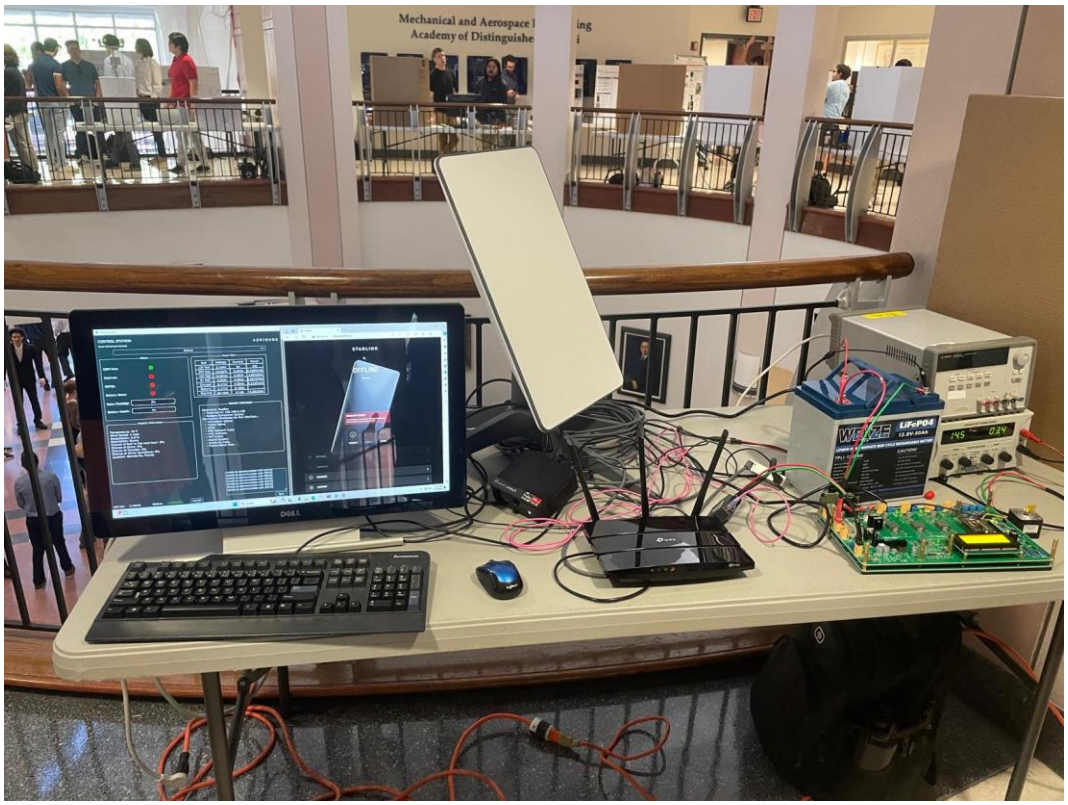
Even though the graphs themselves do look choppy, notice that the y-axis is very close together, so overall, these lines are smooth when zoomed out.

Over the course of the semester, valuable additions to the GUI were added in. Now, it can tell anyone monitoring the system Network, Wifi, and various connectivity information. It can now monitor the battery and voltage of Starlink and it's PoE injector that is supplying it with power.

The different windows of the GUI, with the data graphs displaying the voltage, current, and power of the various systems on board the PCB work perfectly, updating in real-time as new data comes in from the Raspberry Pi Pico onboard the PCB.

IV. Final Setup

The final setup can be seen below at the Senior Design Rotunda Fair.



On the monitor, which is powered externally (AC wall outlet), the left half of the screen is our JUCE GUI, while the right half is the Starlink Web GUI that we used to control the Dish.

V. Additions and Suggestions

a. Network Panel- Adding Linux Support

Currently, the code for the Network Panel runs only on the Windows OS, leveraging native Window APIs to pull network information from the computer and display this on the GUI. It would be a good idea to be able to add Linux support for this, implementing this in Linux, so if the user is using Linux this code is still robust enough to display Linux information on the Network Panel.

b. Starlink Web GUI integration with our GUI

Currently, we can pull up the Starlink Web GUI separately on the computer, and manually control it from there. But, on our GUI, we could add various buttons, that when pressed, would send commands to the Web GUI through socket programming or Network Interfaces to be able to control Starlink from our GUI.

d. GUI control over Drone Station PCB

Right now, the GUI only takes in and displays information from the Raspberry Pi Pico. I've set up some code so that when the "Launch" button is pressed on the main page of the GUI, it'll send byte commands over serial transmission to the Pi Pico. It is a future implementation to write code on the Pico that'll receive these commands, and perhaps open and close the stepper motor (thus opening and closing the drone station's "dome doors"), when the drone needs to come in for landing / recharging or go out for a test flight.

e. GPS / RTKS System / Drone Flight Control

All these areas require further research and development.

References

Texas Instruments, "LM5156 Datasheet," Texas Instruments, [Online]. Available: https://www.ti.com/lit/ds/symlink/lm5156.pdf?ts=1714530580796&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM515. [Accessed: April 30, 2024]

Frans Hegendorp, "CustomMatPlot," GitHub. [Online]. Available: https://github.com/franshej/CustomMatPlot. [Accessed: April 30, 2024].

JUCE. "Documentation," JUCE. [Online]. Available: https://juce.com/learn/documentation/. [Accessed: April 30, 2024]

Webench.ti.com, "Power Designer - Switching Regulator," TI Webench Power Designer. [Online]. Available: https://webench.ti.com/power-designer/switching-regulator?powerSupply=0. [Accessed: May 1, 2024]

Adafruit Industries, "TC1602A-01T Datasheet," Adafruit Industries Datasheets. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/TC1602A-01T.pdf. [Accessed: May 1, 2024]

SparkFun Electronics, "DS-15141 Rotary Encoder - Illuminated (RGB)," SparkFun Electronics Datasheets. [Online]. Available: https://cdn.sparkfun.com/assets/4/e/e/b/f/DS-15141-Rotary_Encoder_-_Illuminated__RGB_.pdf. [Accessed: May 1, 2024]

C. Young, "How to Use a Rotary Encoder in a MCU-Based Project," All About Circuits, Oct. 22, 2020. [Online]. Available: https://www.allaboutcircuits.com/projects/how-to-use-a-rotary-encoder-in-a-mcu-based-project/. [Accessed: May 1, 2024]

F. Shejbal, "CustomMatPlot," GitHub. [Online]. Available: https://github.com/franshej/CustomMatPlot. [Accessed: May 1, 2024]

Texas Instruments, "TPS55288," Texas Instruments, Dallas, TX, USA, Datasheet, 2019. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps55288.pdf?ts=1714572857386&ref_url=https%253A%252F%252Fwww.google.com%252F. [Accessed on: May 1, 2024]

Texas Instruments, "BQ24650," Texas Instruments, Dallas, TX, USA, Datasheet, 2018. [Online]. Available: https://www.ti.com/lit/ds/symlink/bq24650.pdf?ts=1714562778196&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fde-de%252FBQ24650. [Accessed on: May 1, 2024]

Texas Instruments, "TPS51397A," Texas Instruments, Dallas, TX, USA, Datasheet, 2017. [Online]. Available: https://www.ti.com/lit/ds/symlink/tps51397a.pdf?ts=1714516265663. [Accessed on: May 1, 2024]

Raspberry Pi Foundation, "Raspberry Pi Pico Datasheet," Raspberry Pi Foundation, Cambridge, UK, 2021. [Online]. Available: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf. [Accessed on: May 1, 2024]

Texas Instruments, "LM5156," Texas Instruments, Dallas, TX, USA, Datasheet, 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm5156.pdf?ts=1714548786089&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM5156. [Accessed on: May 1, 2024]

MPJA, "33133ms," MPJA, Orlando, FL, USA, Datasheet, 2017. [Online]. Available: https://www.mpja.com/download/33133ms.pdf. [Accessed on: May 1, 2024]

Texas Instruments, "PCF8574," Texas Instruments, Dallas, TX, USA, Datasheet, 2017. [Online]. Available: https://www.ti.com/lit/ds/symlink/pcf8574.pdf?ts=1714556338830. [Accessed on: May 1, 2024]

Texas Instruments, "DRV8871," Texas Instruments, Dallas, TX, USA, Datasheet, 2018. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8871.pdf?ts=1714573533222&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FDRV8871. [Accessed on: May 1, 2024]

Texas Instruments, "OPA4990," Texas Instruments, Dallas, TX, USA, Datasheet, 2018. [Online]. Available: https://www.ti.com/lit/ds/symlink/opa4990.pdf?ts=1714573550984&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FOPA4990. [Accessed on: May 1, 2024]

Boat Hackers, "Starlink," Boat Hackers, San Francisco, CA, USA. [Online]. Available: https://boathackers.com/starlink/. [Accessed on: May 1, 2024]

Appendix

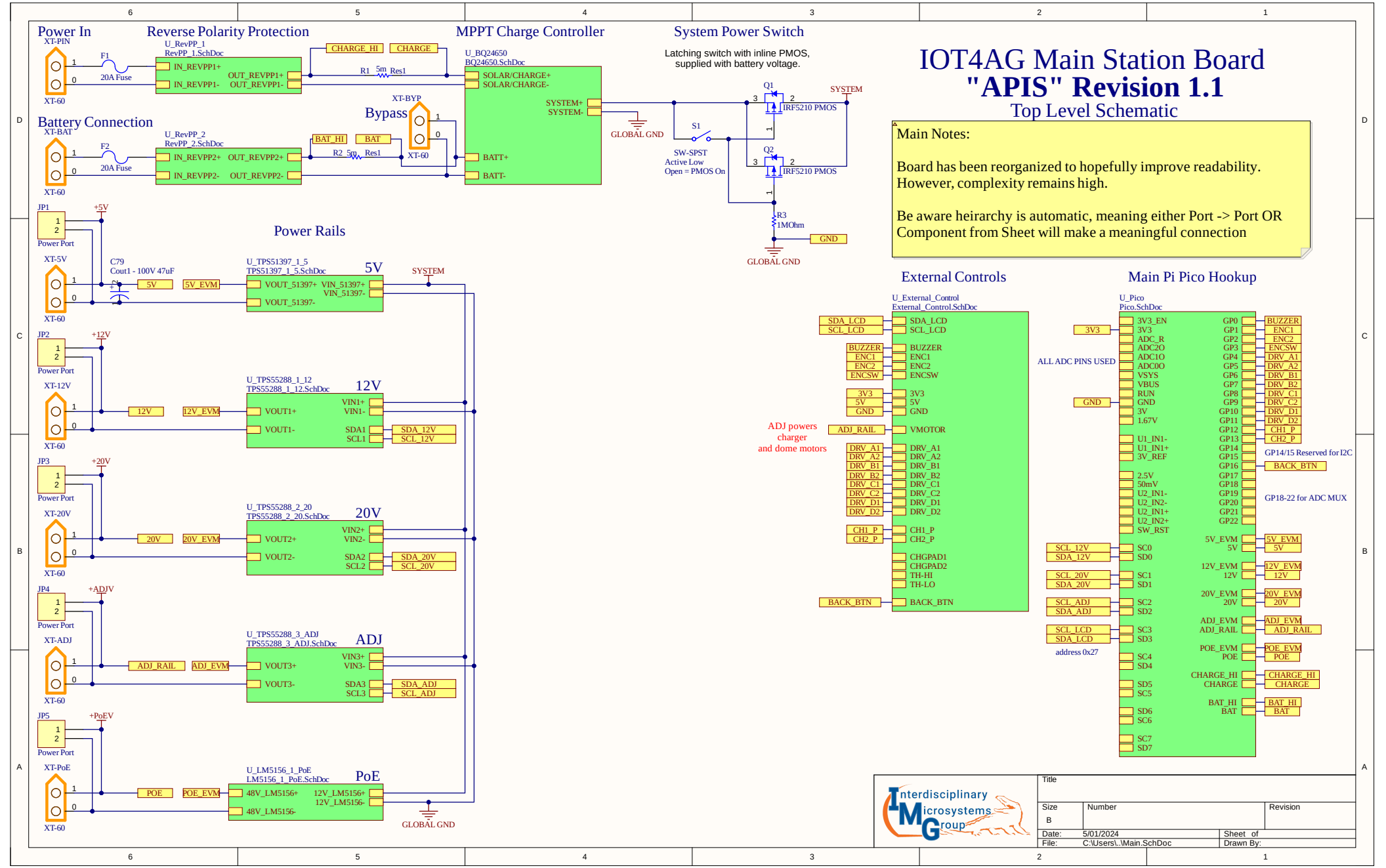
Bill of Materials / Assembly Sheet

Name	Designator	QTY
TPS55288RPMR	U1E, U1F, U1G	3
476AVG100MGBJ	C5C, C6C, C78, C79	4
IHLP6767GZER4R7M01	L1E, L1F, L1G	3
EEHZA1H680P	C8E, C8F, C8G, C54, C55, C56, C57	7
XAL1010-562MED	L1C	1
482020514001	SW1	1
BQ24650RVAT	U1A	1
IRF5210PBF PMOS	Q1, Q2, Q1B, Q1H	4
EEH-ZA1E331P	C17E, C17F, C17G	3
DRV8871DDA	U20, U21, U23, U25	4
XT-60	XT-5V, XT-12V, XT-20V, XT-ADJ, XT-BAT, XT-BYP, XT-PIN, XT-PoE	8
IPP075N15N3 G	Q1C	1
Raspberry Pi Pico	U2	1
IPZ40N04S5L4R8ATMA1	Q1E, Q1F, Q1G, Q2E, Q2F, Q2G	6
CL32B106KBJNNNE	C2, C5A, C8A, C9E, C9F, C9G, C10E, C10F, C10G, C11E, C11F, C11G, C14E, C14F, C14G, C15E, C15F, C15G, C16A, C23, C25, C36, C37, C72, C73, C76, C77	27
KPEG242	LS1	1
C3216X5R1V226M160AC	C5D, C6D, C62, C64, C66, C68	6
22201C106MAT2A	C41	1
PDS1040L-13	D2A, D3A, D4A, D5A	4
OPA4990IDR	U18	1
INA281A4	U11 – Replaced from original INA181	1
INA181A2	U5, U6 – Replaced from original INA181	2
INA181A4IDBVR	--U5--, --U6--, U7, U8, U9, U10, --U11--	7
LM5156	U1C	1
SI7288DP-T1-GE3	Q1A	1
IHLP5050FDER1R8M01	L1D	1
LM393M	U14, U22	2
TCA9548APWR	U1	1
C1210C105K5RACTU	C4E, C4F, C4G, C7E, C7F, C7G, C12E, C12F, C12G, C16E, C16F, C16G	12
TPS51397ARJER	U1D	1
RN73C2A5K23BTDF	R9A	1
GRM21BR61A476ME15L	C9D, C10D, C11D	3
TLV272IDR	U24	1

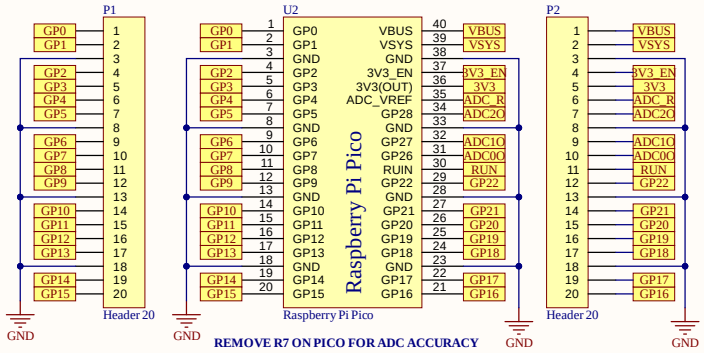
CD4066BM96	U12, U13, U15, U16, U17	5
IHLP2525CZER100M01	L1A	1
STPS20M100SG-TR	D1C	1
PCF8574AT/3,518	U19	1
LM4040C30ILPR	B1	1
C1608X7R1H104K080AA	C1, C1E, C1F, C1G, C2A, C2D, C2E, C2F, C2G, C3, C3E, C3F, C3G, C4, C5, C6, C6A, C7, C7A, C7D, C8, C8D, C9, C9A, C10, C11, C12, C12D, C13, C14, C15, C16, C17, C18, C18E, C18F, C18G, C19, C20, C21, C22, C24, C26, C27, C28, C29, C30, C31, C32, C34, C35, C38, C39, C42, C43, C44, C45, C46, C47, C48, C49, C50, C51, C52, C53, C58, C59, C60, C61, C63, C65, C67, C69, C70, C71, C74, C75	77
CSNL2512FT3L00	R4C	1
ERJ-6LWFR008V	R4A	1
LM358BIDR	U3, U4	2
C1608X5R1H105K080AB	C1A, C3A, C4A, C10A, C11A, C13A, C13E, C13F, C13G, C15A	10
885382208006	C2C, C8C	2
GRM31CR71H475KA12L	C4C	1
ZLLS350TA	D1A	1
06031U101FAT2A	C3D, C5E, C5F, C5G	4
SMAJ40CA	D2B, D2H	2
Schottky 1N5819	D1	1
C0603C183K1RACTU	C4D	1
1N4745A	D1B, D1H	2
MMBTA42LT1G	Q6, Q7	2
RG1608P-2491-B-T5	R6C	1
CRF1206-FZ-R010ELF	R9E, R9F, R9G	3
08051C104KAT2A	C33, C40	2
C1608X5R1E105K080AC	C1D	1
GCM188R71E124KA37D	C9C	1
ERA6AEB4993V	R12A	1
MSASE168SB7105KTNA01 [Replaces: EMK107B7105KA-T]	C3C	1
MMSZ5231B-7-F	D1D	1
RT0603BRD072KL	R7C	1
CRCW080536K0FKEA	R13A	1
C1608X7R2A472K080AA	C6E, C6F, C6G	3
1825910-6	SW2, SW3	2
RC0603FR-071RL (Replaces: CRCW06031R00FKEA)	R3E, R3F, R3G, R4E, R4F, R4G	6

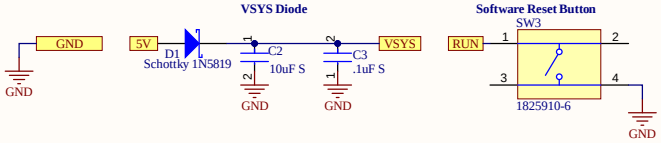
CRCW0603150KFKEA	R5E, R5F, R5G, R10E, R10F, R10G	6
CR1206-JW-752ELF	R11A	1
RG1608P-124-B-T5	R3C	1
CRCW06032R00FKEA	R6A	1
2N7002-7-F	Q3, Q4	2
RC0603FR-07100KL	R2E, R2F, R2G, R5D, R10A	5
CRCW06030000Z0EA	R1E, R1F, R1G, R2D, R3D	5
RC0603FR-0715KL	R1A, R2A, R6D, R10D	4
08053C104KAT4A [Replaces CC0805KRX7R8BB104]	C1C, C10C	2
C0603C220J5GACTU	C12A, C14A	2
CRCW04021K00FKED	R1C	1
RC0603FR-0747KL	R7D, R9D	2
CRCW060356K2FKEA	R6E, R6F, R6G	3
ERJ-3EKF1103V	R4D, R11E, R11F, R11G	4
CL21B103KBANNNC	C7C	1
CRCW06032R20FKEA	R3A	1
CRCW080510R0FKEA	R5A	1
CRCW040220K0FKED	R7E, R7F, R7G	3
CRCW040249K9FKED	R8E, R8F, R8G	3
CRCW060351R1FKEA	R1D	1
CRCW040248K7FKED	R5C	1
CRCW0805590KFKEA	R7A	1
RC0402FR-07100RL	R2C, R8A	2
RC0402FR-071ML	R3	1
330K 1% 0603(1608)	R8D	1
Screw Terminal	AUX-OUT1, CHG-OUT1, JP1, JP2, JP3, JP4, JP5, STEP-OUT1, STEP-OUT2	9
0.1uF	CAP1B, CAP1H, CAP2B, CAP2H	4
LED1	D2, D3, D5, D8, D9	5
LCD-20X4B	DS1	1
20A Fuse	F1, F2	2
Light Emitting Diode	LED1, LED1A, LED2, LED2A	4
529802B02500G	* FOR TO-220 POWER FET Ics	4
NT	NT1A, NT1E, NT1F, NT1G	4
Header 20	P1, P2	2
Header 12	P3, P4	2
Header 7	P5, P6, P7, P8	4
100uF	PCAP1B, PCAP1H, PCAP2B, PCAP2H	4

THT RES - 5m Ohm	R1, R2, R15, R16, R34, R35, R36	7
THT RES - 47k	R1B, R1H	2
THT RES - 10k	R2B, R2H	2
THT RES - Values Vary	R4 (1k), R5 (1k), R65 (1k), R68 (1k), R81 (0 ohm), R82 (0 ohm)	6
THT RES - 100k, (DNP R57, R58, R59, R60, R61)	R6, R12, R55, R56, R57, R58, R59, R60, R61, R71, R72, R74, R76, R77, R78, R79	16
THT RES - 330k	R7, R18, R28	3
THT RES - 10k (DNP R63, R64, R66)	R8, R53, R63, R64, R66, R70	6
THT RES - 5.6k	R9, R13	2
THT RES - 8.1k	R10	1
THT RES - 50k	R11	1
THT RES - 47k	R14, R22, R33, R54	4
THT RES - 22k	R17, R25, R26, R27, R30, R42, R43, R44, R52	9
THT RES - Values Vary	R19 (2k), R20 (2k), R37 (3.8k), R40 (680), R41 (10k)	5
THT RES - 33k	R21, R29, R31, R32, R45, R46, R50	7
THT RES - 220k	R24, R47, R48, R49, R51	5
Potentiometer 10k	R62	1
THT RES - 30k	R67, R69, R75, R80	4
THT RES - 1MEG	R73	1
SW-SPST	S1	1


Power In
XT-PIN
F1
20A Fuse
XT-60
Reverse Polarity Protection
U_RevPP_1
RevPP_1.SchDoc
IN_REVPP1+
IN_REVPP1-
OUT_REVPP1+
OUT_REVPP1-
CHARGE_HI
CHARGE
R1 5m Res1
MPPT Charge Controller
U_BQ24650
BQ24650.SchDoc
SOLAR/CHARGE+
SOLAR/CHARGE-
SYSTEM+
SYSTEM-
BATT+
BATT-
System Power Switch
Latching switch with inline PMOS,
supplied with battery voltage.
Q1
Q2
IRF5210 PMOS
SYSTEM
S1
SW-SPST
Active Low
Open = PMOS On
R3
1MOhm
GND
GLOBAL GND
Bypass
XT-BYP
XT-60
Battery Connection
XT-BAT
F2
20A Fuse
U_RevPP_2
RevPP_2.SchDoc
IN_REVPP2+
IN_REVPP2-
OUT_REVPP2+
OUT_REVPP2-
BAT_HI
BAT
R2 5m Res1
IOT4AG Main Station Board
"APIS" Revision 1.1
Top Level Schematic
Main Notes:
Board has been reorganized to hopefully improve readability. However, complexity remains high.
Be aware heirarchy is automatic, meaning either Port -> Port OR Component from Sheet will make a meaningful connection
Power Rails
JP1
+5V
Power Port
XT-5V
C79
Cout1 - 100V 47uF
5V
5V_EVM
U_TPS51397_1_5
TPS51397_1_5.SchDoc
VOUT_51397+
VOUT_51397-
VIN_51397+
VIN_51397-
SYSTEM
JP2
+12V
XT-12V
12V
12V_EVM
U_TPS55288_1_12
TPS55288_1_12.SchDoc
VOUT1+
VOUT1-
VIN1+
VIN1-
SDA1
SCL1
SDA_12V
SCL_12V
JP3
+20V
XT-20V
20V
20V_EVM
U_TPS55288_2_20
TPS55288_2_20.SchDoc
VOUT2+
VOUT2-
VIN2+
VIN2-
SDA2
SCL2
SDA_20V
SCL_20V
JP4
+ADJV
XT-ADJ
ADJ
ADJ_RAIL
ADJ_EVM
U_TPS55288_3_ADJ
TPS55288_3_ADJ.SchDoc
VOUT3+
VOUT3-
VIN3+
VIN3-
SDA3
SCL3
SDA_ADJ
SCL_ADJ
JP5
+PoEV
XT-PoE
PoE
POE
POE_EVM
U_LM5156_1_PoE
LM5156_1_PoE.SchDoc
48V_LM5156+
48V_LM5156-
12V_LM5156+
12V_LM5156-
GLOBAL GND
External Controls
U_External_Control
External_Control.SchDoc
SDA_LCD
SCL_LCD
BUZZER
ENC1
ENC2
ENCSW
3V3
5V
GND
ADJ powers
charger
and dome motors
ADJ_RAIL
VMOTOR
DRV_A1
DRV_A2
DRV_B1
DRV_B2
DRV_C1
DRV_C2
DRV_D1
DRV_D2
CH1_P
CH2_P
CHGPAD1
CHGPAD2
TH-HI
TH-LO
BACK_BTN
Main Pi Pico Hookup
U_Pico
Pico.SchDoc
3V3_EN
ADC_R
ADC2O
ADC1O
ADC0O
VSYS
VBUS
RUN
3V
1.67V
U1_IN1-
U1_IN1+
3V_REF
2.5V
50mV
U2_IN1-
U2_IN2-
U2_IN1+
U2_IN2+
SW_RST
ALL ADC PINS USED
GP0
GP1
GP2
GP3
GP4
GP5
GP6
GP7
GP8
GP9
GP10
GP11
GP12
GP13
GP14
GP15
GP16
GP17
GP18
GP19
GP20
GP21
GP22
GP14/15 Reserved for I2C
GP18-22 for ADC MUX
SC0
SD0
SC1
SD1
SC2
SD2
SC3
SD3
SC4
SD4
SC5
SD5
SC6
SD6
SC7
SD7
address 0x27
5V_EVM
12V_EVM
20V_EVM
ADJ_EVM
POE_EVM
CHARGE_HI
CHARGE
BAT_HI
BAT
Interdisciplinary Microsystems Group
Title
Size
B
Number
Revision
Date: 5/01/2024
Sheet of
File: C:\Users\..\Main.SchDoc
Drawn By:

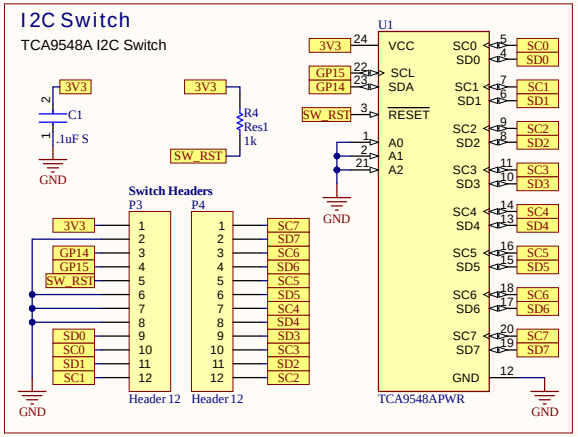
Raspberry Pico Microcontroller





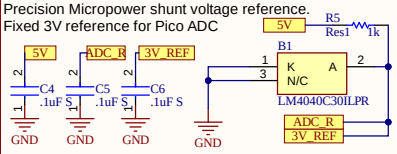
I2C Switch

TCA9548A I2C Switch



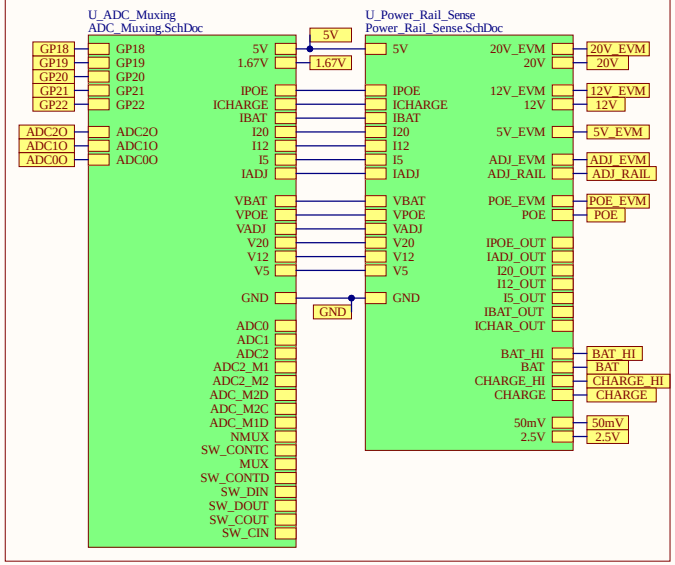
LM4040 Voltage Reference

Precision Micropower shunt voltage reference.
Fixed 3V reference for Pico ADC



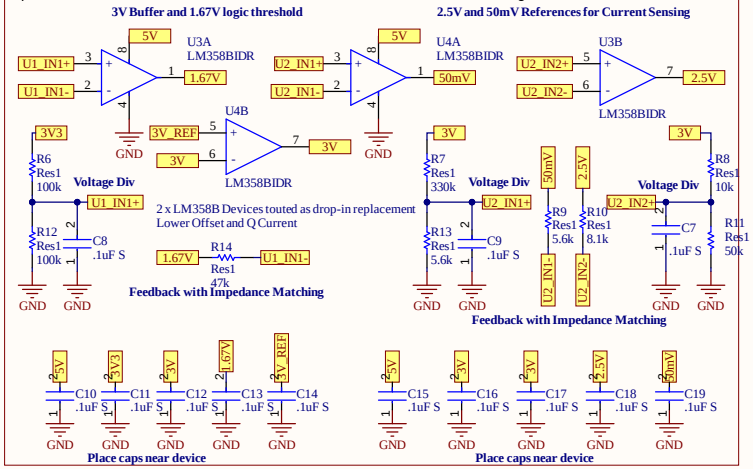
Place one 3V cap near Pico and the rest near the device

Voltage / Current Monitoring to The ADC + Controls



Voltage References

1.67V voltage reference for comparator threshold, splitting the 3.3V logic levels in half for best level shifting performance. 3V Buffer to feed 2.5V and 50mV reference for current sensing.



Place caps near device

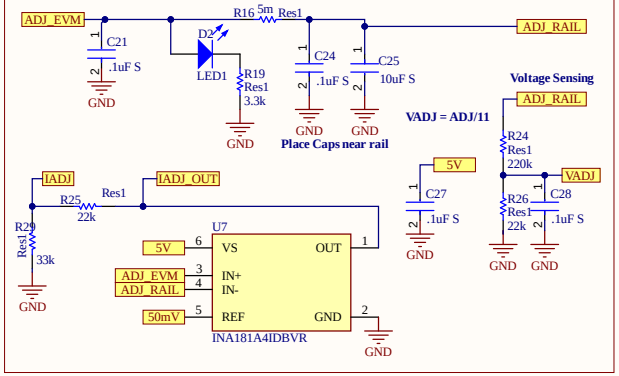
Place caps near device

Title	Raspberry Pi Pico Wiring		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of: 1 of 1	
File: C:\Users\..\Pico.SchDoc		Drawn By: Shane Aspenwall	

Power Rail Measurement and Sensing

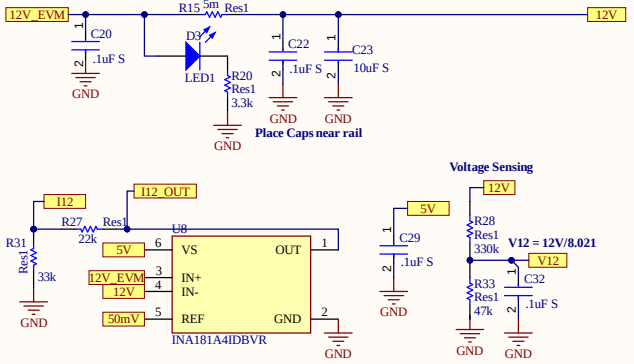
ADJ Rail Measurement

Voltage and Current Measurement and LED rail status.

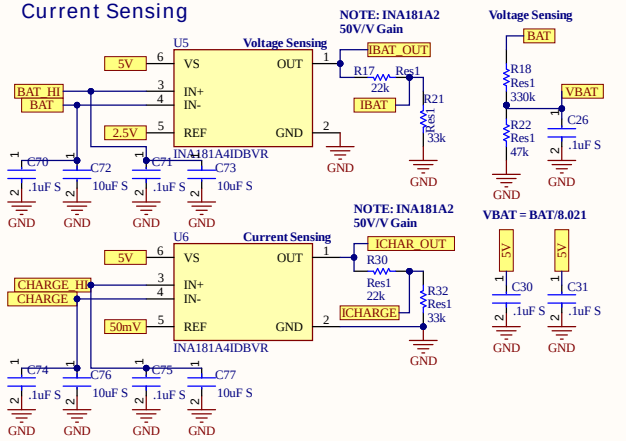


Variable 12V/5A Rail Measurement

Voltage and Current Measurement, overvoltage protection, and LED rail status.

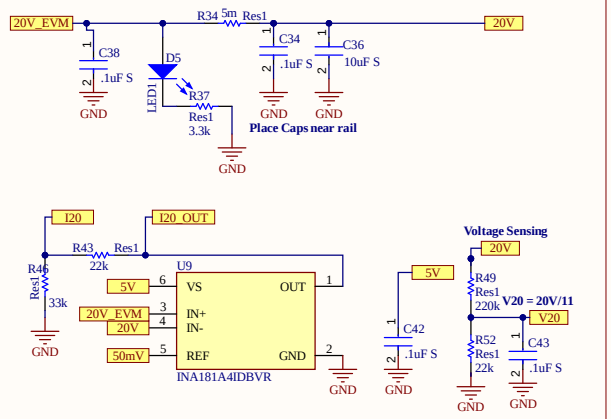


Current Sensing



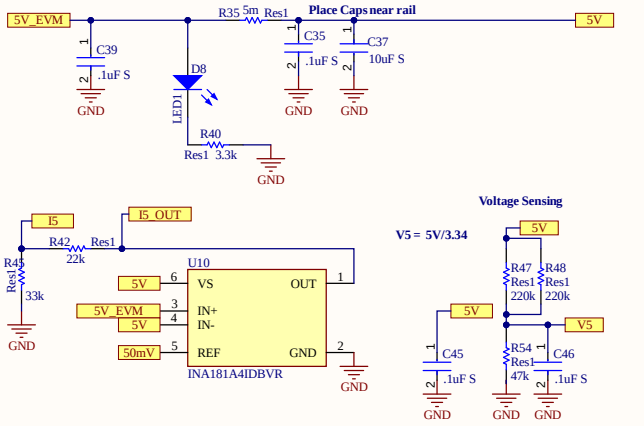
20V/5A Rail Measurement

Voltage and Current Measurement, overvoltage protection, and LED rail status.



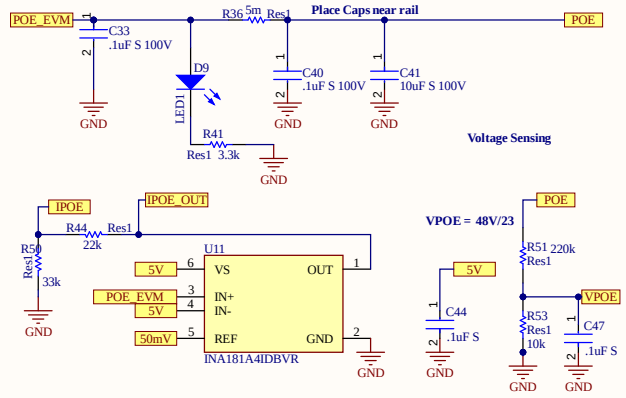
5V/10A Rail Measurement

Voltage and Current Measurement, overvoltage protection, and LED rail status.

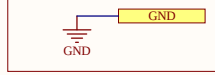


48V/3A PoE Rail Measurement

Voltage and Current Measurement, and LED rail status.



GND Tie

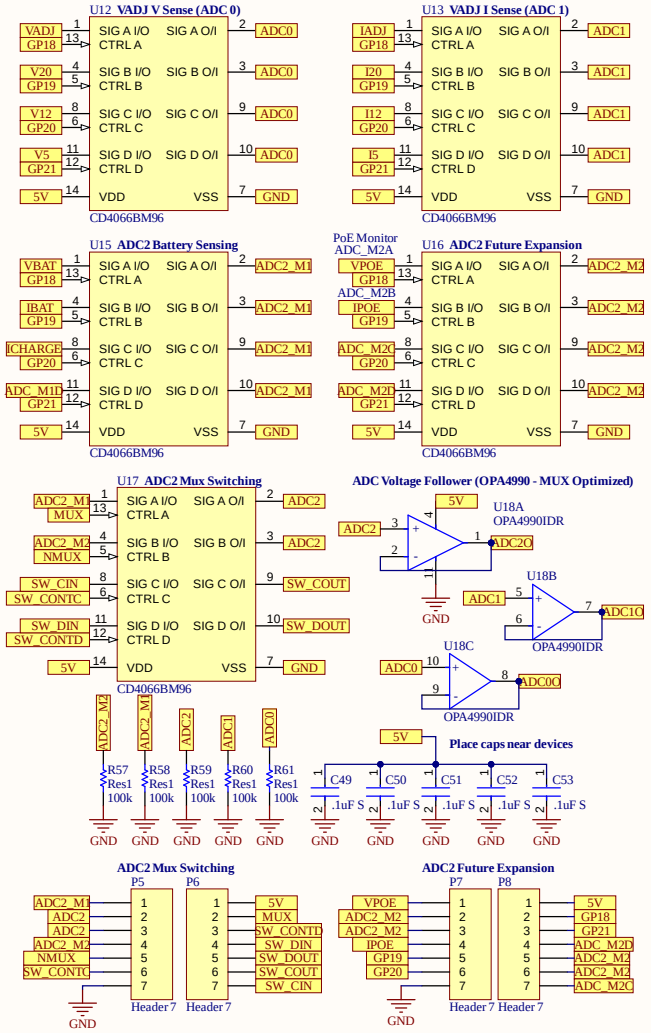


Interdisciplinary Microsystems Group

Title	Power Measurement Systems		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of: 1 of 1	
File: C:\Users\..\Power_Rail_Sense.SchDoc		Drawn By: Shane Aspenwall	

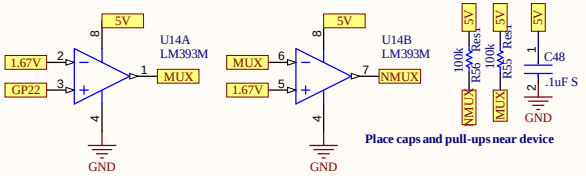
ADC Analog Switching

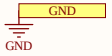
CD4066 IC's extend the 3 ADC channels to a potential 16 ADC readings. Weak Pull-Downs discharge ADC lines.



Level Shifter / Mux Alternator

LM393 Comparator for level shifting of 3.3V to 5V logic levels. Creates 5V and inverted 5V signal to select between certain Mux IC's



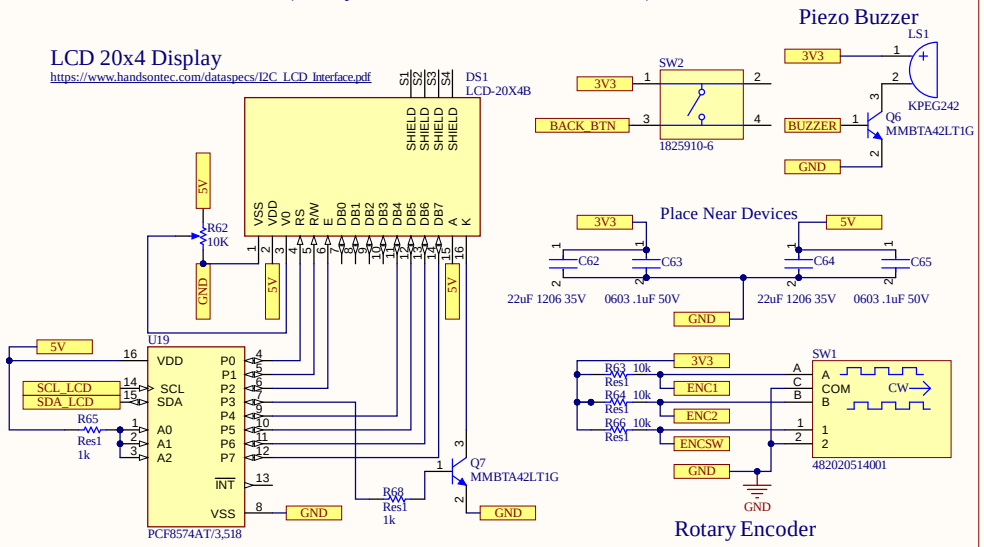


Title	ADC Muxing		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of	1 of 1
File: C:\Users\..\ADC_Muxing.SchDoc		Drawn By:	Shane Aspenwall

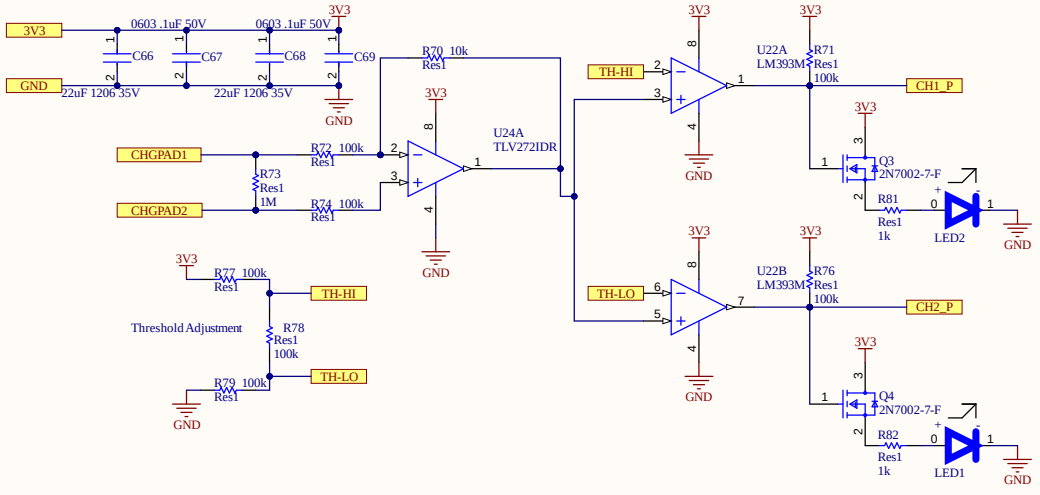
Interdisciplinary Microsystems Group

Human Interface Devices

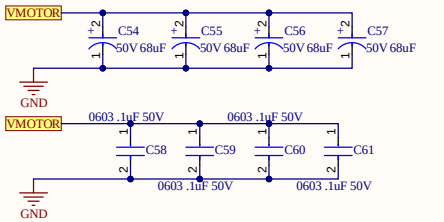
(Rotary Encoder, Buzzer, LCD Readout).



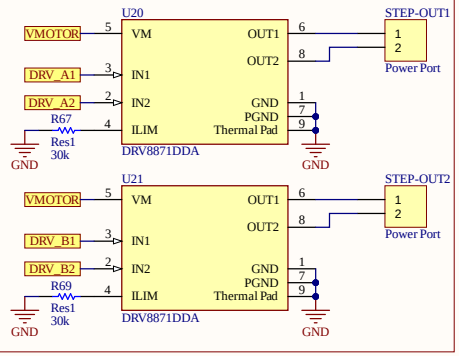
Charging Pad Polarity Detection



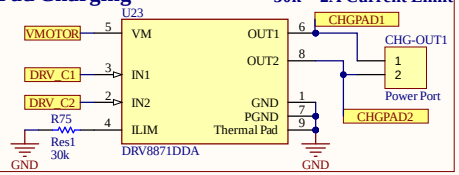
H-Bridge Drivers



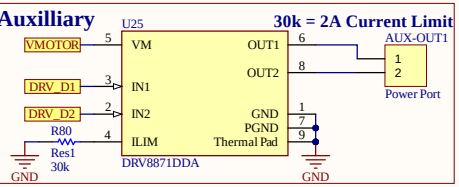
Dome Opening Mechanism 30k = 2A Current Limit



Pad Charging 30k = 2A Current Limit



Auxilliary 30k = 2A Current Limit

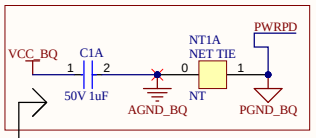


Interdisciplinary Microsystems Group

Title	External Control		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of: 1 of 1	
File: C:\Users\..\External_Control.SchDoc		Drawn By: Shane Aspenwall	

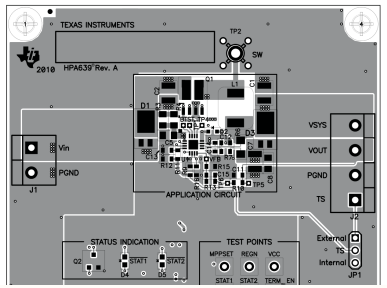
NOTE: THIS IS A SNIPPET DESIGN NOT A STANDALONE BOARD DESIGN

This design should be placed after the main board is designed. This can be done using Altium Snippets as a feature. This is why the designators have the -P1 Suffix. DO NOT paste this design into your final board design. Instead, create a Port or other terminal connection, then manually connect your design to the PCB DOC snippet by first: Selecting all parts on the PCB DOC, creating a snippet, then pasting the snippet on your finished PCB. Connect your Board Ports to the Snippet as needed.

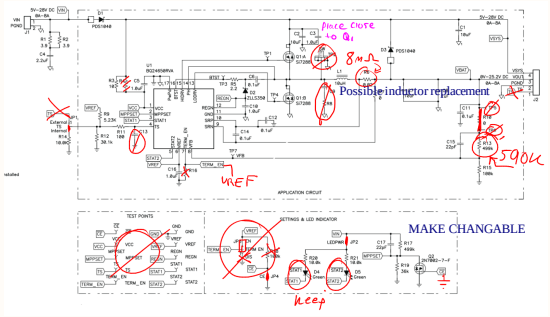


The PGND plane and the AGND plane are connected at the terminal of the capacitor at the VCC pin. Thus the noise caused by the MOSFET driver and parasitic inductance does not interfere with the AGND and internal control circuit.

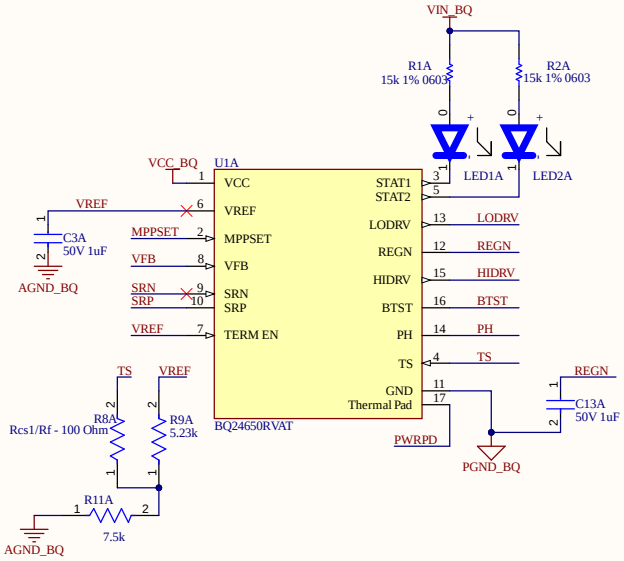
PCB Layout Suggestions (Not Actual Design)



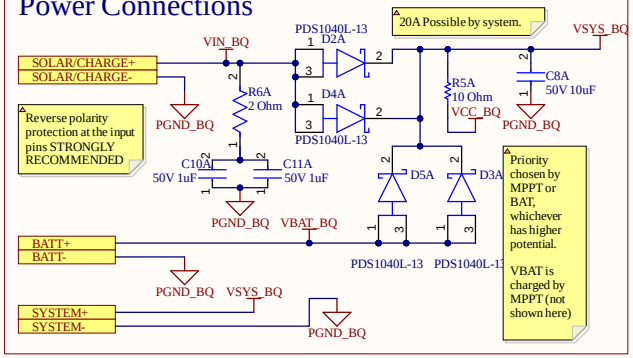
Board Modifications



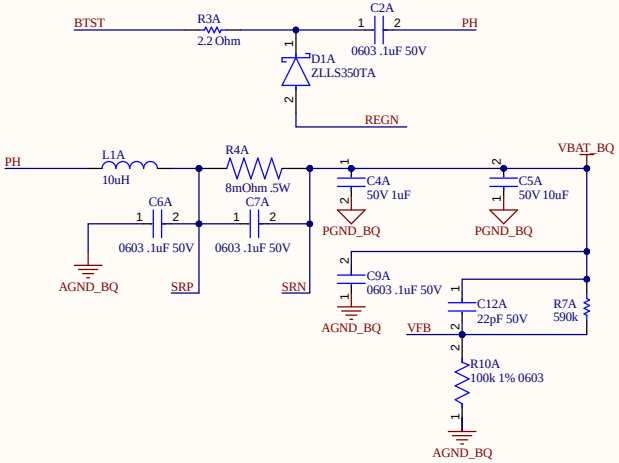
MOSFET BOOST Switching



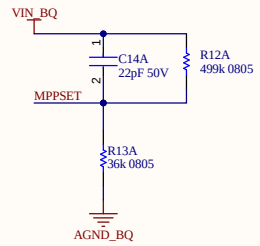
Power Connections



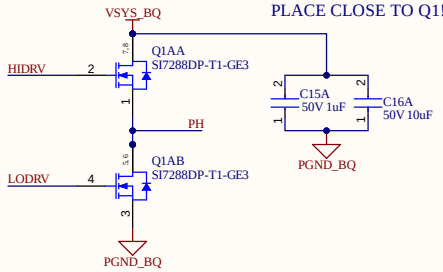
MPPT Charge Section



MPPT Set



PowerFET switching



Interdisciplinary Microsystems Group

Title: BQ24650 MPPT Charge Controller		
Size: B	Number: 01	Revision: 00
Date: 5/01/2024	Sheet of: 1 of 1	
File: C:\Users\..\BQ24650.SchDoc	Drawn By: Shane Aspenwall	

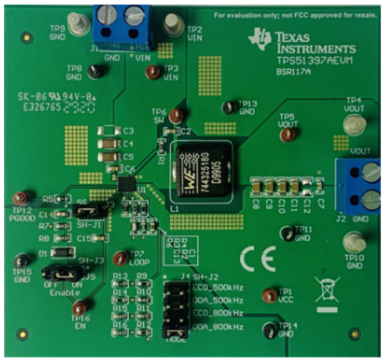


Figure 5-6. Board Top View

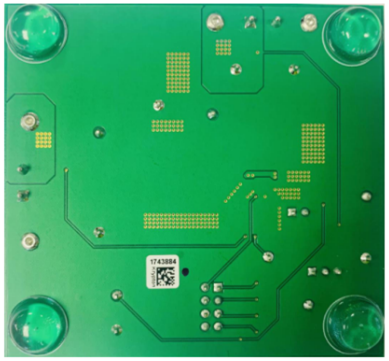


Figure 5-7. Board Bottom View

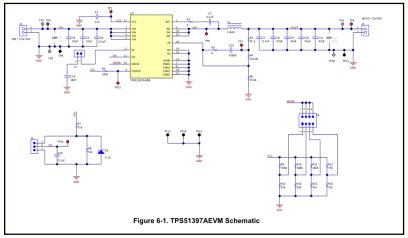
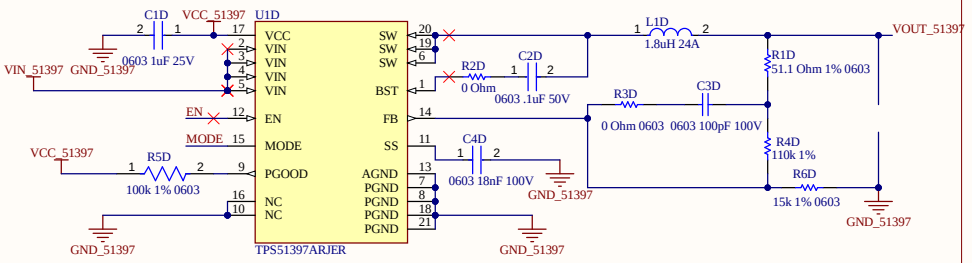


Figure 6-1. TPS51397AEVM Schematic

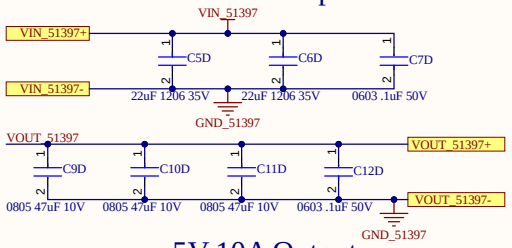
EVM Reference Schematic and Board Layout

Main Chip Connections and Supporting Components

Make reference to the EVM schematic to note that VCC and VIN are separate voltages. Be aware that VCC is ONLY USED for internal control of the digital modules. This is basically a low-current linear regulator supply internal to the chip that sets interna l config settings and is decoupled from the VIN input

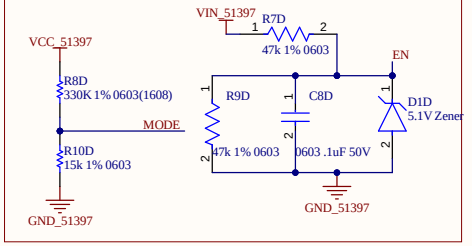


5.5 - 24V Input



5V 10A Output

Mode / Enable Hardware Selects

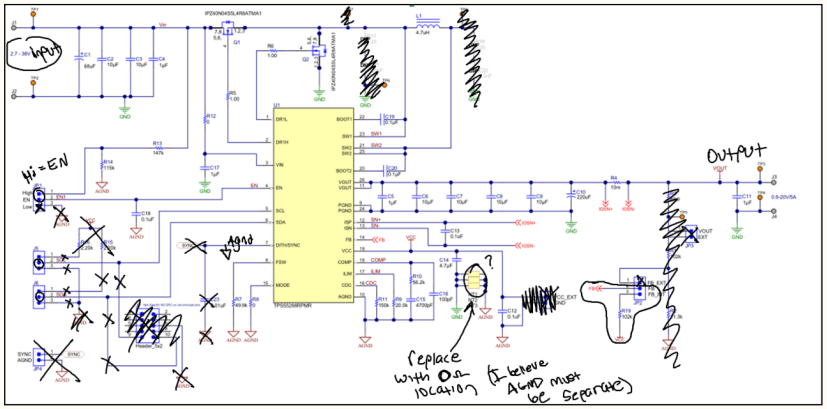


NOTE: THIS IS A SNIPPET DESIGN NOT A STANDALONE BOARD DESIGN

This design should be placed after the main board is designed. This can be done using Altium Snippets as a feature. This is why the designators have the -P1 Suffix. DO NOT paste this design into your final board design. Instead, create a Port or other terminal connection, then manually connect your design to the PCB DOC snippet by first: Selecting all parts on the PCB DOC, creating a snippet, then pasting the snippet on your finished PCB. Connect your Board Ports to the Snippet as needed.

Interdisciplinary Microsystems Group

Title	TPS51397A 5V Supply for Drone Station		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of: 1 of 1	
File: C:\Users\..\TPS51397_1_5.SchDoc		Drawn By: Shane Aspenwall	



PCB Layout Suggestions (Not Actual Design)

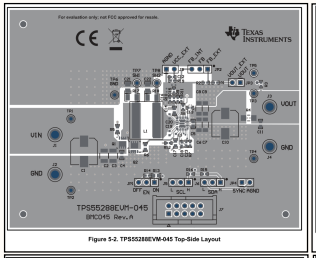


Figure 5-2. TPS55288EVM-045 Top-Side Layout

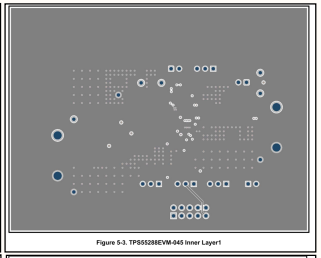


Figure 5-3. TPS55288EVM-045 Inner Layer1

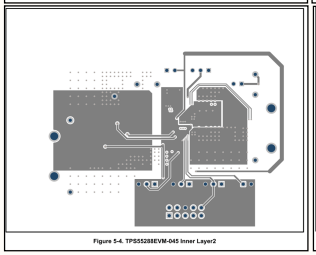


Figure 5-4. TPS55288EVM-045 Inner Layer2

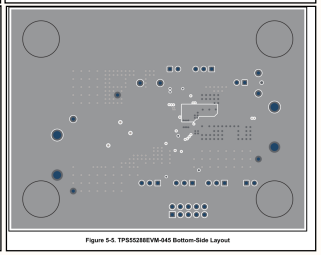
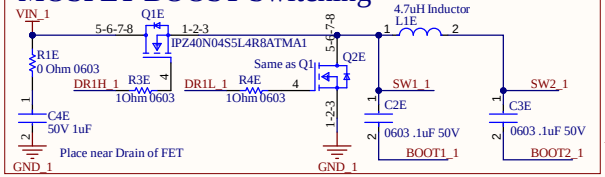


Figure 5-5. TPS55288EVM-045 Bottom-Side Layout

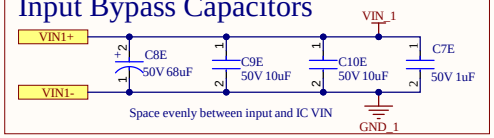
NOTE: THIS IS A SNIPPET DESIGN NOT A STANDALONE BOARD DESIGN

This design should be placed after the main board is designed. This can be done using Altium Snippets as a feature. This is why the designators have the -P1 Suffix. DO NOT paste this design into your final board design. Instead, create a Port or other terminal connection, then manually connect your design to the PCB DOC snippet by first: Selecting all parts on the PCB DOC, creating a snippet, then pasting the snippet on your finished PCB. Connect your Board Ports to the Snippet as needed.

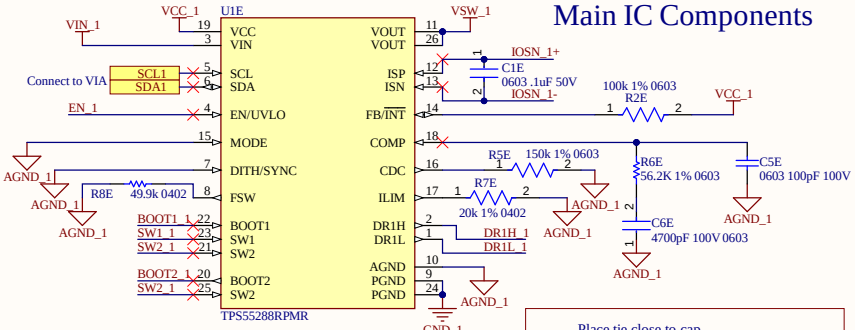
MOSFET BOOST Switching



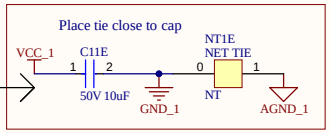
Input Bypass Capacitors



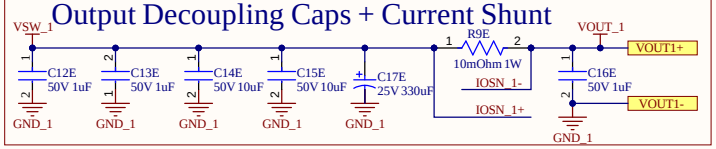
Main IC Components



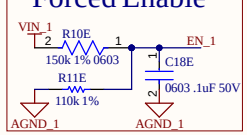
The PGND plane and the AGND plane are connected at the terminal of the capacitor at the VCC pin. Thus the noise caused by the MOSFET driver and parasitic inductance does not interfere with the AGND and internal control circuit.



Output Decoupling Caps + Current Shunt

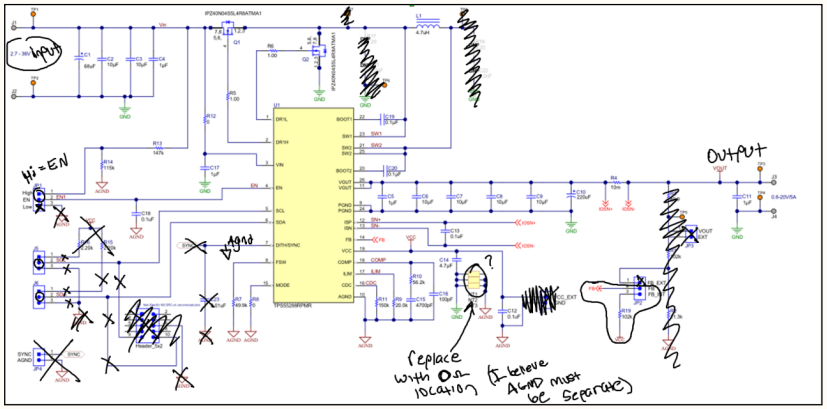


Forced Enable



Interdisciplinary Microsystems Group

Title	TPS55288 I2C Controlled Power Supply Module		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of 1 of 1	
File: C:\Users\..\TPS55288_1_12.SchDoc		Drawn By: Shane Aspenwall	



PCB Layout Suggestions (Not Actual Design)

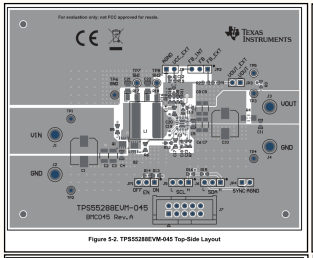


Figure 5-2. TPS55288EVM-045 Top-Side Layout

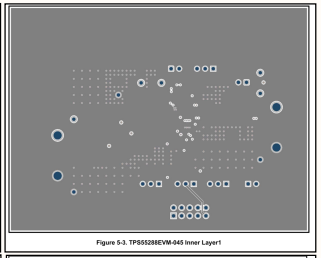


Figure 5-3. TPS55288EVM-045 Inner Layer1

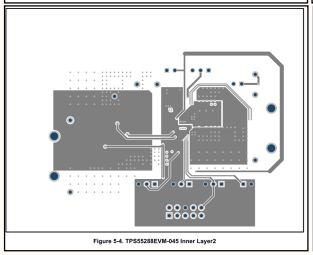


Figure 5-4. TPS55288EVM-045 Inner Layer2

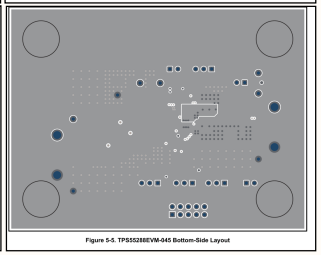
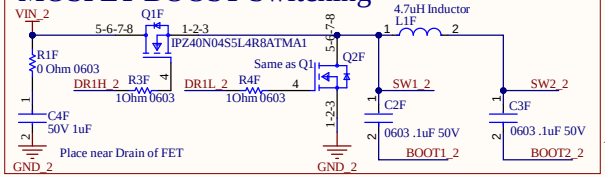


Figure 5-5. TPS55288EVM-045 Bottom-Side Layout

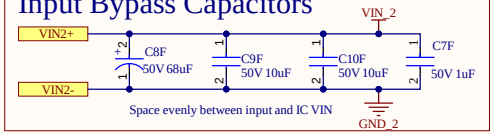
NOTE: THIS IS A SNIPPET DESIGN NOT A STANDALONE BOARD DESIGN

This design should be placed after the main board is designed. This can be done using Altium Snippets as a feature. This is why the designators have the -P1 Suffix. DO NOT paste this design into your final board design. Instead, create a Port or other terminal connection, then manually connect your design to the PCB DOC snippet by first: Selecting all parts on the PCB DOC, creating a snippet, then pasting the snippet on your finished PCB. Connect your Board Ports to the Snippet as needed.

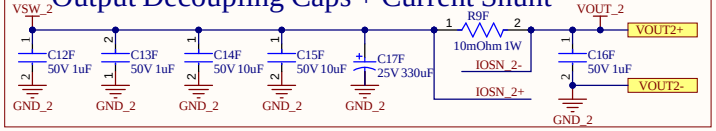
MOSFET BOOST Switching



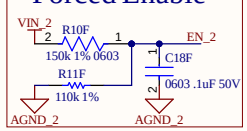
Input Bypass Capacitors



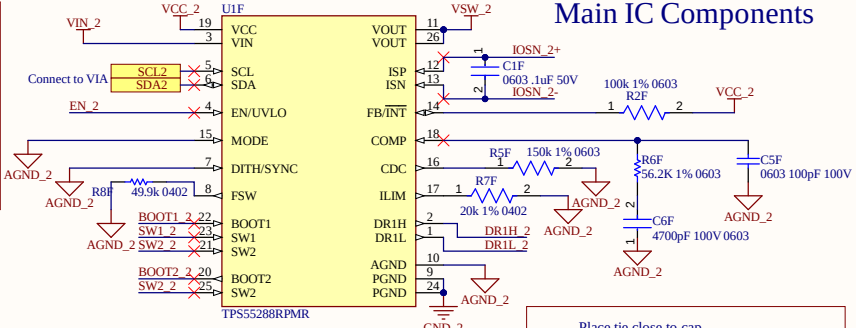
Output Decoupling Caps + Current Shunt



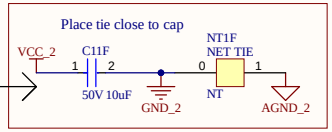
Forced Enable



Main IC Components

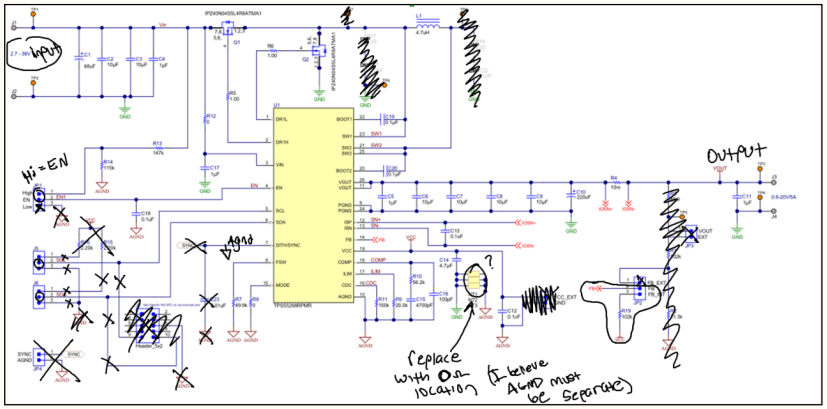


The PGND plane and the AGND plane are connected at the terminal of the capacitor at the VCC pin. Thus the noise caused by the MOSFET driver and parasitic inductance does not interfere with the AGND and internal control circuit.



Title: TPS55288 I2C Controlled Power Supply Module			
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of 1 of 1	
File: C:\Users\..\TPS55288_2_20.SchDoc		Drawn By: Shane Aspenwall	

Interdisciplinary Microsystems Group



PCB Layout Suggestions (Not Actual Design)

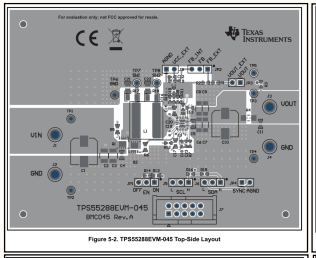


Figure 5-2. TPS55288EVM-045 Top-Side Layout

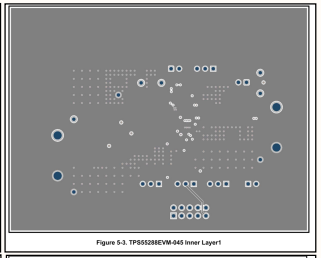


Figure 5-3. TPS55288EVM-045 Inner Layer1

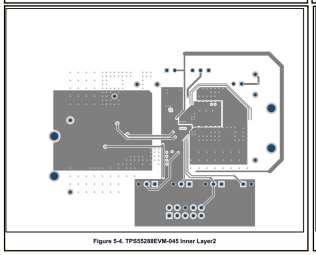


Figure 5-4. TPS55288EVM-045 Inner Layer2

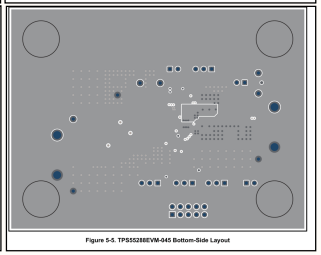
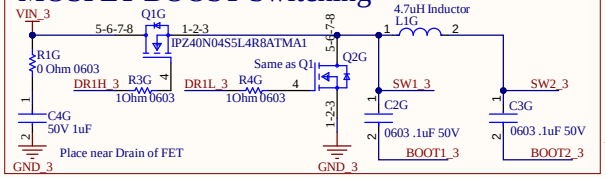


Figure 5-5. TPS55288EVM-045 Bottom-Side Layout

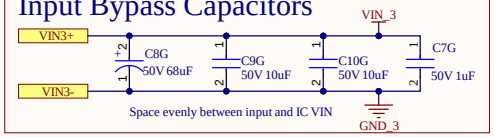
NOTE: THIS IS A SNIPPET DESIGN NOT A STANDALONE BOARD DESIGN

This design should be placed after the main board is designed. This can be done using Altium Snippets as a feature. This is why the designators have the -P1 Suffix. DO NOT paste this design into your final board design. Instead, create a Port or other terminal connection, then manually connect your design to the PCB DOC snippet by first: Selecting all parts on the PCB DOC, creating a snippet, then pasting the snippet on your finished PCB. Connect your Board Ports to the Snippet as needed.

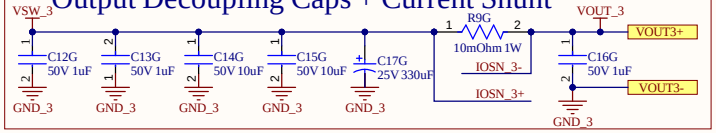
MOSFET BOOST Switching



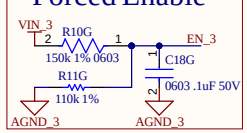
Input Bypass Capacitors



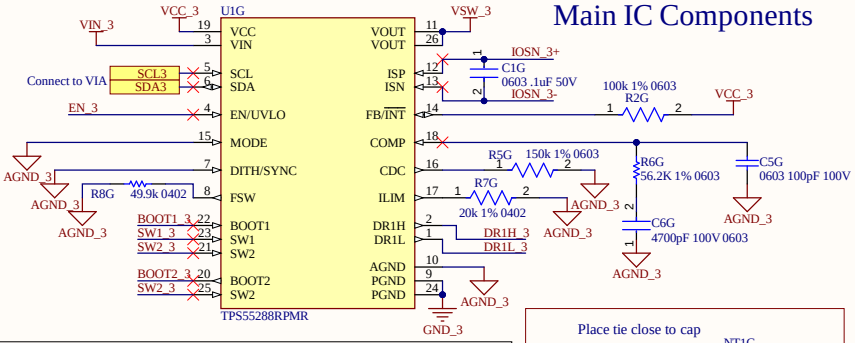
Output Decoupling Caps + Current Shunt



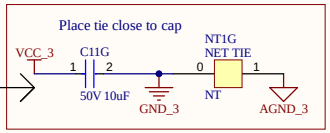
Forced Enable



Main IC Components



The PGND plane and the AGND plane are connected at the terminal of the capacitor at the VCC pin. Thus the noise caused by the MOSFET driver and parasitic inductance does not interfere with the AGND and internal control circuit.



Title	TPS55288 I2C Controlled Power Supply Module		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of 1 of 1	
File: C:\Users\..\TPS55288_3_ADJ.SchDoc		Drawn By: Shane Aspenwall	

Interdisciplinary Microsystems Group

10.2 Typical Application

Figure 10-1 shows all optional components to design a boost converter.

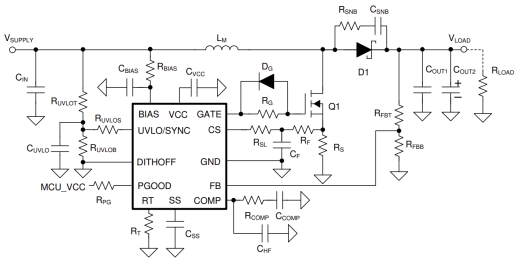
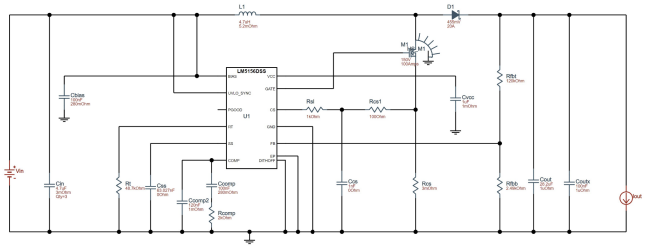


Figure 10-1. Typical Boost Converter Circuit With Optional Components

LM5156 Schematic



Part	Manufacturer	Part Number	Quantity	Price ($)	Footprint (mm²)	Description
HS_M1	Aavid	529802B02500G	1	6.16	1203.13	ThetaSA: 5.05 C/W Height: 1.5 mm HeatSink_Power: 20 W
Rsl	Vishay-Dale	CRCW04021K00FKED	1	0.01	3	Resistance: 1 kΩ Tolerance: 1.0% Power: 63 mW
U1	Texas Instruments	LM5156DSSR	1	0.73	12	
Ccomp	AVX	08053C104KAT2A	1	0.01	6.75	Cap: 100 nF Total Derated Cap: 100 nF VDC: 25 V ESR: 280 mΩ Package: 0805
Cin	Murata	GRM31CR71H475KA12L	3	0.1	10.92	Cap: 4.7 µF Total Derated Cap: 11 µF VDC: 50 V ESR: 3 mΩ Package: 2220
Rcs1	Vishay-Dale	CRCW0402100RFKED	1	0.01	3	Resistance: 100 Ω Tolerance: 1.0% Power: 63 mW
Rcs	Stackpole Electronics Inc	CSNL1206FTL00	1	0.19	10.92	Resistance: 3 mΩ Tolerance: 1.0% Power: 1 W
Rt	Vishay-Dale	CRCW040248K7FKED	1	0.01	3	Resistance: 48.7 kΩ Tolerance: 1.0% Power: 63 mW
M1	Infineon Technologies	IPP075N15N3GXKSA1	1	3.72	127.32	VdsMax: 150 V IdsMax: 100 Amps
Ccomp2	Murata	GRM188R71E124KA01D	1	0.02	4.68	Cap: 120 nF Total Derated Cap: 120 nF VDC: 25 V ESR: 1 mΩ Package: 0603
Css	CUSTOM	CUSTOM	1	NA	NA	Cap: 63.09 nF Total Derated Cap: 63 nF VDC: 0 V ESR: 0 Ω Package: null
Rfbb	Susumu Co Ltd	RG1608P-2491-B-T5	1	0.06	4.68	Resistance: 2.49 kΩ Tolerance: 0.1% Power: 100 mW
Rcomp	Yageo	RT06038RD072KL	1	0.04	4.68	Resistance: 2 kΩ Tolerance: 0.1% Power: 100 mW
Cout	CUSTOM	CUSTOM	1	NA	NA	Cap: 28.2 µF Total Derated Cap: 28 µF VDC: 72 V ESR: 1 µΩ Package: null
Coutx	CUSTOM	CUSTOM	1	NA	NA	Cap: 100 nF Total Derated Cap: 100 nF VDC: 96 V ESR: 1 µΩ Package: null
Rfbt	Susumu Co Ltd	RG1608P-124-B-T5	1	0.06	4.68	Resistance: 120 kΩ Tolerance: 0.1% Power: 100 mW
Cbias	AVX	08053C104KAT2A	1	0.01	6.75	Cap: 100 nF Total Derated Cap: 100 nF VDC: 25 V ESR: 280 mΩ Package: 0805
D1	STMicroelectronics	STPS20M100SG-TR	1	1.94	209.7	Type: Schottky VRRM: 100 V Io: 20 A
Cvcc	Taiyo Yuden	EMK107B7105KA-T	1	0.01	4.68	Cap: 1 µF Total Derated Cap: 1 µF VDC: 16 V ESR: 1 mΩ Package: 0603
L1	Coilcraft	XAL1010-472MEB	1	1.71	158.8	L: 4.7 µH DCR: 5.2 mΩ IDC: 24 A
Ccs	Samsung Electro-Mechanics	CL21C102JBCNNNC	1	0.01	6.75	Cap: 1 nF Total Derated Cap: 1 nF VDC: 50 V ESR: 0 Ω Package: 0805

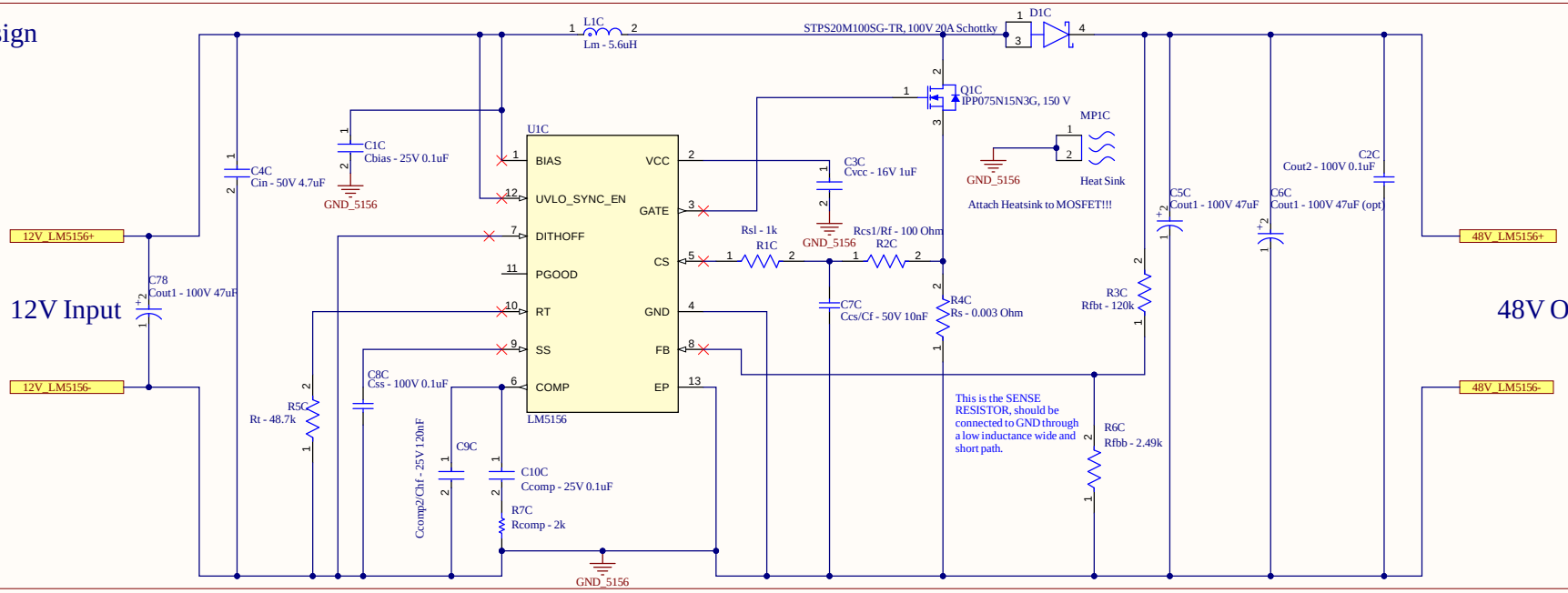
Notes

This IC is capable of boosting a 12V input to 48V at a maximum of 3 amps. This inherently requires 12+ amps of input current in order to supply this output. Keeping trace length short is critical for proper performance. C10 is an optional inclusion in order to improve ripple, if needed.

NOTE: THIS IS A SNIPPET DESIGN NOT A STANDALONE BOARD DESIGN

This design should be placed after the main board is designed. This can be done using Altium Snippets as a feature. This is why the designators have the -P1 Suffix. DO NOT paste this design into your final board design. Instead, create a Port or other terminal connection, then manually connect your design to the PCB DOC snippet by first: Selecting all parts on the PCB DOC, creating a snippet, then pasting the snippet on your finished PCB. Connect your Board Ports to the Snippet as needed.

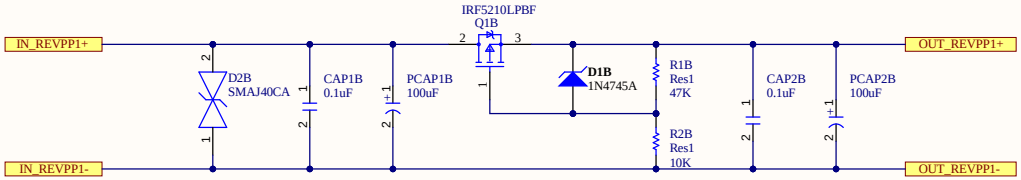
Full Design



Title	LM5156 48V PoE Ethernet Power Supply		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of 1 of 1	
File: C:\Users\..\LM5156_1_PoE.SchDoc		Drawn By: Zain Nasrullah	

Reverse Polarity Protection

Voltage limited by Vgs of PMOS, while the current is limited by the physical junctions



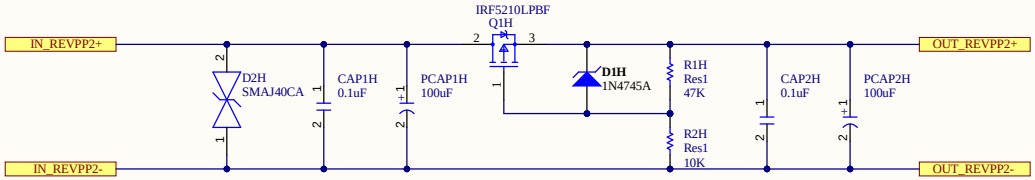
If D1 is not populated and R1 is shorted to GND, the protection will still work, however Vin will be limited by Vgs.

D1 and R1 regulate Vgs within a safe region to increase maximum Vin value.

Interdisciplinary Microsystems Group	Title: Reverse Polarity Protection - Remake		
	Size: B	Number: 01	Revision: 0
	Date: 5/01/2024	Sheet of: 1 of 1	
	File: C:\Users\..\RevPP_1.SchDoc	Drawn By: Shane Aspenwall	

Reverse Polarity Protection

Voltage limited by Vgs of PMOS, while the current is limited by the physical junctions

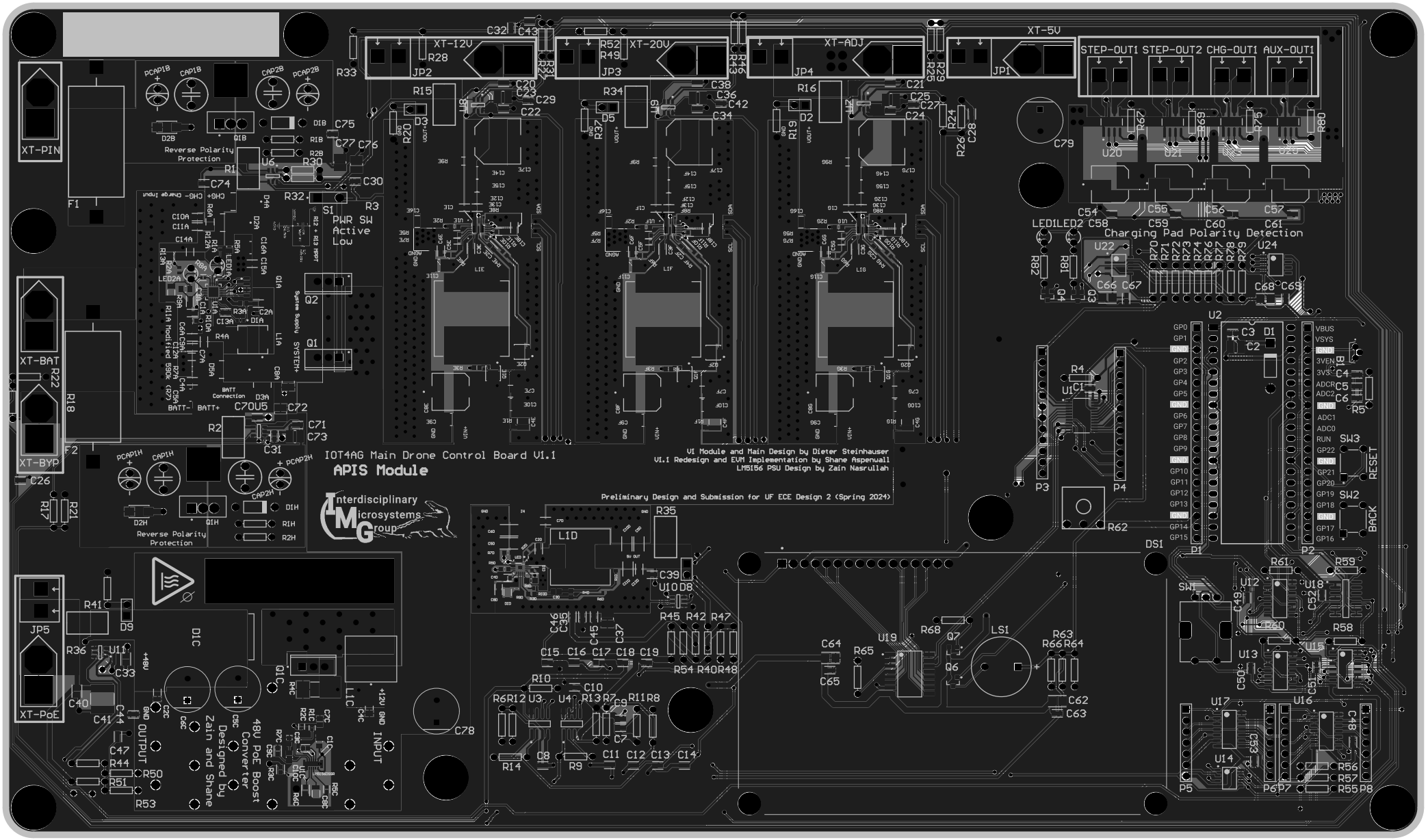


If D1 is not populated and R1 is shorted to GND, the protection will still work, however Vin will be limited by Vgs.

D1 and R1 regulate Vgs within a safe region to increase maximum Vin value.

Interdisciplinary Microsystems Group

Title	Reverse Polarity Protection - Remake		
Size: B	Number: 01		Revision: 00
Date: 5/01/2024		Sheet of: 1 of 1	
File: C:\Users\..\RevPP_2.SchDoc		Drawn By: Shane Aspenwall	


XT-PIN
XT-BAT
XT-BYP
XT-PoE
JP5
F1
F2
XT-12V
XT-20V
XT-ADJ
XT-5V
JP2
JP3
JP4
JP1
STEP-OUT1 STEP-OUT2 CHG-OUT1 AUX-OUT1
Reverse Polarity Protection
PWR SW Active Low
Charging Pad Polarity Detection
IOT4AG Main Drone Control Board V1.1
APIS Module
V1 Module and Main Design by Dieter Steinhauser
V1.1 Redesign and EVM Implementation by Shane Aspenwall
LM5156 PSU Design by Zain Nasrullah
Preliminary Design and Submission for UF ECE Design 2 (Spring 2024)
Interdisciplinary Microsystems Group
48V PoE Boost Converter
Designed by Zain and Shane
INPUT
OUTPUT
SW1
SW2
SW3
RESET
BACK