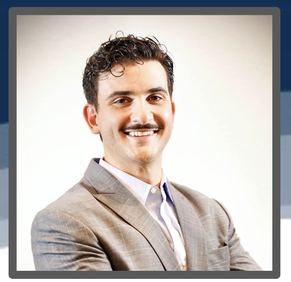
OpenSense v3

Simplifying Sensors for Educational Experimentation



James Eyler
System/Power Design
Project Management
Computer Engineering
BSCpE

Brandon Collins
Software Design
Computer Engineering
BSCpE

Noah Wilfond
Photonics Sensor Design
Photonics Science and Engineering
BSPSE
Electrical Engineering
BSEE

Daniel Gonzalez
Sensor Design
Computer Engineering
BSCpE

Motivation and Background

Problem: Many sensor platforms are expensive, closed-source, and difficult to customize.
This makes hands-on learning less accessible.

Educational Goal:
OpenSense is designed to lower barriers to circuit design and embedded systems.

It helps make sensor development more approachable for students and educators.

Foundation: DRACO Labs developed OpenSense as a low-cost, open-source sensor platform.
It provides a scalable base for educational use and experimentation.

Flexibility Demonstration:
The custom optical spectrometer demonstrates that the platform supports advanced sensing applications.
It highlights the system's expandability and customization potential.

Our Contribution: Our team improved modularity, usability, and sensor support.
These additions expand the platform while preserving its educational focus.

Table of Engineering Specifications

System Specification Table		
Specifications	**Description**	**Target Values**
Unit Cost	The platform should have a low cost per unit for volume	$40- $60
System Lifespan	The device should be guaranteed to support data collection over a set period of time	2-4 hours
Sampling Rate	The platform should be able to record multiple points of data over time	10-20 Hz per Sensor
Data Storage	The device should have enough storage to gather significant data for analysis	1-4 hours storage
Sensor Accuracy	The sensor should collect data accurately	85-95% Accuracy
PCB Dimensions	The PCB should be made to fit in a compact/portable housing	100-200cm2
Sensor Support	The platform should be able to support 5 or more sensors	5 Sensors
Spectrometer wavelength range	The spectrometer should be able to plot a spectrum consisting of at least the visible wavelength as well as some UV wavelengths.	400-700 nm
Spectral Resolution	To be able to distinctly measure different wavelengths, a spectral resolution of at least 5 nm must be met.	1-5 nm

Basic Goals and Objectives

Goals	Objectives
Design a data logger capable of acquiring and recording outputs from multiple sensors simultaneously.	• Implement a data acquisition system capable of sampling each connected sensor at a minimum rate of 10 Hz. • Support simultaneous support between a central MCU and at least 5 sensors. • Store sensor data to an SD card in CSV format, with a header row listing channel names and units. • Ensure that sensor readings achieve a minimum accuracy of 85% in relaying correct values.
Maintain a minimum of 1 hour of onboard storage and 2 hours of battery life for autonomous field operation.	• Integrate a battery module and automatic USB-to-battery power source switchover • Implement microSD logging to store at least 1 hour of 5-channel sensor data at 10 Hz.
Develop a user-configurable interface to present sensor data in user-readable units.	• Implement a web-based configuration interface where users can input calibration values (offset, scale, units) per sensor channel.
Design an optical spectrometer module to establish proof-of-concept integration with the sensor platform, focusing on accessibility and educational use.	• Implement a diffraction-grating-based spectrometer with a single photodiode detector to verify integration of custom sensors with the platform. • Control grating rotation via a servo motor to scan across the target wavelength range of the visible spectrum, from 400 to 700 nm. • Use OpenSense configuration interface to map each angular position to a wavelength and display a spectrum.

Advanced Goals and Objectives

Goals	Objectives
Develop a software framework that supports user-defined sensor models, increasing customization and expanding supported sensor options.	• Develop a parameterized generic driver covering device address, register, byte count, endianness, scaling, offset, and units for any I^2C or analog sensor. • Extend configuration to be able to adapt more complex calibration functions than just linear sensors
Incorporate a GUI to support real-time data visualization, enabling intuitive interpretation for educational purposes.	• Host a browser-accessible dashboard over Wi-Fi SoftAP delivering live sensor telemetry at ≥ 2 Hz using WebSocket push. • Provide on-demand CSV download of logged data directly through the web interface without requiring SD card removal.
Create a housing that enables system portability by minimizing weight, reducing volume, and ensuring robustness during transport and deployment.	• Design and fabricate an enclosure for the main board and battery module that protects all components from handling stress during transport and field deployment.

Stretch Goals and Objectives

Goals	Objectives
Incorporate modular sensor ports to support plug-and-play integration and replacement of multiple sensor types.	• Design sensor port connectors using a standardized pinout so that any compliant sensor module can be physically swapped without tools or soldering. • Implement hardware-level protections against incorrect sensor connections, overvoltage, or reversed polarity.
Enable wireless data transmission for automated upload to external devices or storage systems	• Implement a cross-platform interface for data visualization to ensure broad accessibility and usability beyond the onboard dashboard. • Support automated wireless upload of logged CSV files to an external device or cloud storage upon reconnection to a known network.
Extend the spectrometer's wavelength range into the near-UV or near-IR to broaden educational and research applications	• Modify optical components to extend wavelength range from visible (400 to 700 nm) to 300 to 800 nm.

Choosing an MCU

ESP32-WROOM	STM32	MSP430

• Core/Speed
  ○ Dual Core
  ○ Up to 240MHz
• Memory
  ○ 384kB ROM
  ○ 512kB SRAM
• Connectivity
  ○ Wifi / Bluetooth
• Languages
  ○ C
  ○ C++
	• Core/Speed
	  ○ Single Core
	  ○ 56MHz
	• Memory
	  ○ 256kB Flash Memory
	  ○ 512kB SRAM
	• Connectivity
	  ○ Nothing Listed
	• Languages
	  ○ C
	  ○ C++
		• Core/Speed
		  ○ Single Core → 16-Bit
		  ○ 16MHz
		• Memory
		  ○ 15.5kB FRAM
		• Connectivity
		  ○ Nothing Listed
		• Languages
		  ○ C

- We choose the ESP32 because it allows for the implementation of FreeRTOS tasks making the use of concurrency easier to implement and allows priority integration.

Choosing an I²C Multiplexer

TCA9548APWR	PCA9547PW	LTC4305IGN

• Number of Channels
  ○ 8
• Clock Frequency
  ○ 0 to 400kHz
• Voltage Logic
  ○ 1.8V, 2.5V, 3.3V, 5.0V
	• Number of Channels
	  ○ 8
	• Clock Frequency
	  ○ 0 to 400kHz
	• Voltage Logic
	  ○ 1.8V, 2.5V, 3.3V, 5.0V
		• Number of Channels
		  ○ 2
		• Clock Frequency
		  ○ 0 to 400kHz
		• Voltage Logic
		  ○ 1.8V, 2.5V, 3.3V, 5.0V

- TCA9548APWR is a more popular IC, because of this there is more available documentation on circuit orientation for the device
  - There are development breakout boards (with schematics) for this IC specifically

Choosing a Regulators (3.3V)

LM2596SX-3.3/NOPB	MP2315	LM2623

• Switching Frequency
  ○ 150kHz
• Vin_{MIN}
  ○ 4.5V
	• Switching Frequency
	  ○ 500kHz
	• Vin_{MIN}
	  ○ 4.5V
		• Switching Frequency
		  ○ 300kHz - 2MHz
		• Vin_{MIN}
		  ○ 0.8V

- Switching Frequency is very important
  - Regulators with lower switching frequency are less sensitive to board design than those with higher
  - <= 1MHz is recommended (Dr. Weeks)

Choosing a Regulators (5.0V)

LM3481MMX/NOPB	TPS563201	LM27402

• Switching Frequency
  ○ 100kHz ~ 1MHz
• Vin_{MIN}
  ○ 2.97V ~ 48V
	• Switching Frequency
	  ○ 580kHz
	• Vin_{MIN}
	  ○ 4.5V
		• Switching Frequency
		  ○ 200kHz - 1.2MHz
		• Vin_{MIN}
		  ○ 3V

- Switching Frequency is very important
  - Regulators with lower switching frequency are less sensitive to board design than those with higher
  - <= 1MHz is recommended (Dr. Weeks)

Choosing a Humidity Sensor

HIH-6130	SHTC3	HDC1080

• Accuracy (RH)
  ○ ±4%RH
• Supply
  ○ 2.32V - 5.5V
• Package
  ○ 4.9mm x 6.0 mm
	• Accuracy (RH)
	  ○ ±2%RH
	• Supply
	  ○ 1.62 - 3.6V
	• Package
	  ○ 2.0 x 2.0 mm
		• Accuracy (RH)
		  ○ ±2%RH
		• Supply
		  ○ 2.7V - 5.5V
		• Package
		  ○ 4.0 x 4.0 mm

- Although the %RH for the SHTC3 and the HDC1080 are better than the HIH-6130 which is for more general purpose applications, we still chose the HIH-6130 over them as the footprints for those sensors were too small to reliably solder and the pins were on the bottom.
- On the other hand the HIH-6130 has a footprint that is much easier to work with for this project while still being fairly accurate for our purposes.

Choosing a Temperature Sensor

LMT87DCK	TMP36	MCP9700A

• Accuracy (25°C)
  ○ ±0.4°C
• Supply
  ○ 2.7V - 5.5V
• Quiescent Current
  ○ ~5–10 µA
	• Accuracy (25°C)
	  ○ ±1°C
	• Supply
	  ○ 2.7V - 5.5V
	• Quiescent Current
	  ○ ~50 µA
		• Accuracy (25°C)
		  ○ ±1°C
		• Supply
		  ○ 2.3V - 5.5V
		• Quiescent Current
		  ○ ~6 µA

- We chose to go with the LMT87DCK because it had the greatest accuracy of of ±0.4°C while also having a low quiescent current as a lower quiescent current is always better.

Choosing a Pressure Sensor

34-00015 FSR(Force Sensitive Resistor)	7BB-27-4CL0 (Piezo Disk)

- Best for
  - Measuring static pressure such as steady presses
- Power
  - Divider current draws current whenever powered

- Best for
  - Measuring dynamic pressure such as taps or knocks
- Power
  - Requires essentially zero senor power other than the op-amp

- We chose to go with a FSR for our sensor as we found that it works more consistently and reliably than a piezo disk for detecting changes in pressure and it was easier to work with static pressure than dynamic pressure for showing off the capabilities of the sensor.

Choosing a Brightness Sensor

VEML7700	OPT3001	VEML6030

• Range
  ○ 0 - 140,000 lux
• Supply
  ○ 2.5 V - 3.6 V
• Package
  ○ 6.8 x 2.35 x 3.0 mm
	• Range
	  ○ 0.01 - 83,000 lux
	• Supply
	  ○ 1.6 V - 3.6 V
	• Package
	  ○ 2.0 x 2.0 x 0.65 mm
		• Range
		  ○ 0 -140,000 lux
		• Supply
		  ○ 2.5 V - 3.6 V
		• Package
		  ○ 2.0 x 2.0 x 0.87 mm

- We chose to go with the VEML7700 due to it being very easy to use and reliable as it is used on many different dev boards.
- It also supports a wide range of lux values which makes it more versatile compared to the OPT3001.
- One of the major factors was that it also has a package size that is much easier to work with compared to the other options.

Choosing a Photodiode

Vishay TEMD5080X01	OSRAM SFH 2716 A01	Thorlabs FD11A

• Spectral Range
  ○ 350 - 100 nm
• Active Area
  ○ 7.7 mm
• Cost
  ○ $1.8
	• Spectral Range
	  ○ 350 - 1000 nm
	• Active Area
	  ○ 0.35 mm
	• Cost
	  ○ $0.86
		• Spectral Range
		  ○ 320-1100 nm
		• Active Area
		  ○ 1.21 mm
		• Cost
		  ○ $1.20

- We chose to go with the TEMD5080 photodiode due to it having the necessary spectral range to meet our project requirements while also having a good cost and the largest active area making it the easiest to work with.

Choosing a Motor

DFRobot SER0046	Adafruit 1404	Adafruit 154

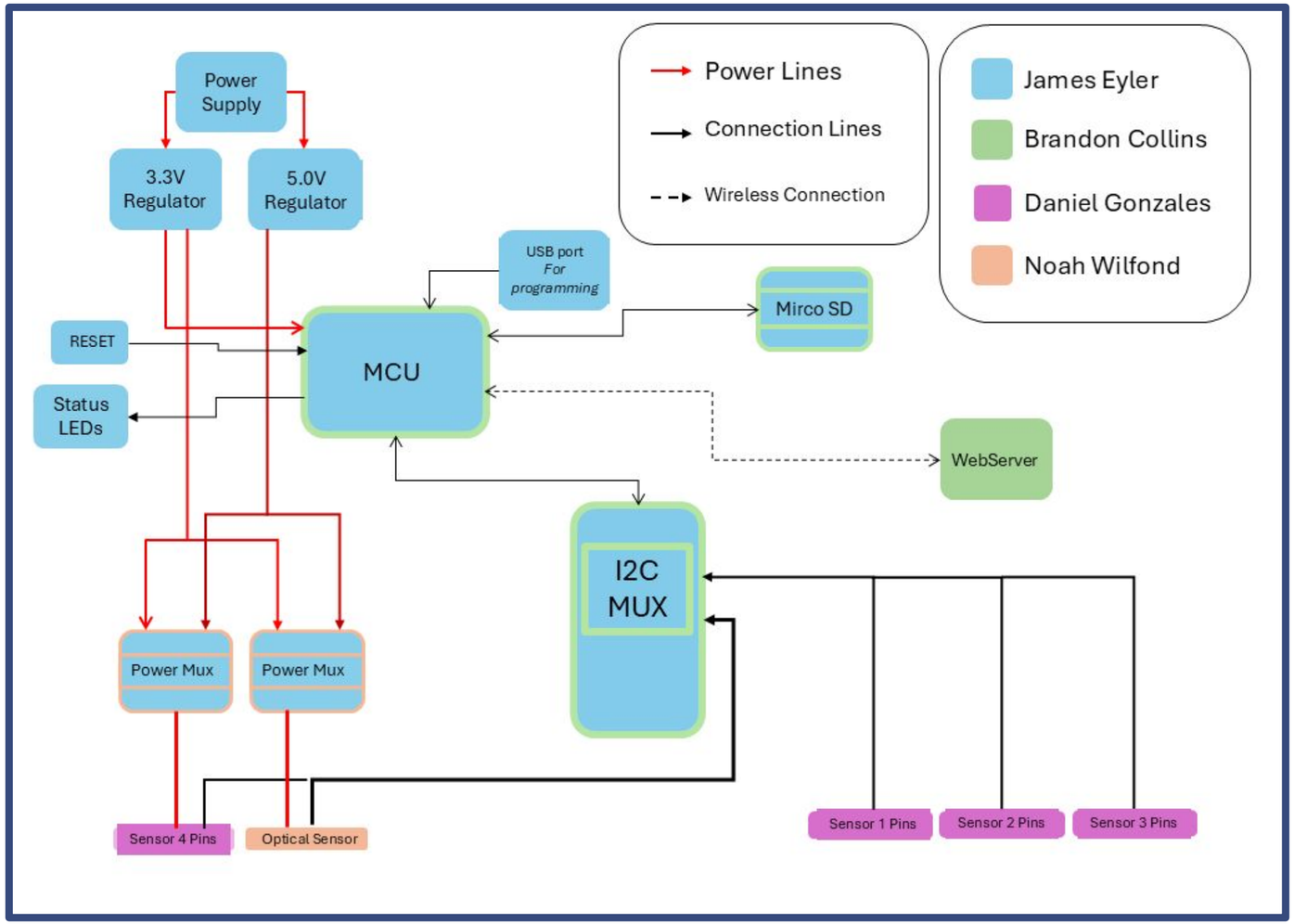
Micro-Servo:
- Operating Voltage
  - 4.8V - 6V
- Max Current Draw
  - 800 mA
- Position Feedback
  - Yes
- Cost
  - $6.90

Servo Motor:
  - 4.8 - 6 V
- Max Current Draw
  - 1.3 - 1.5A
- Position Feedback
  - Yes
- Cost
  - $14.95

Servo Motor:
- Operating Voltage
  - 5 - 6 V
- Max Current Draw
  - 0.9-1.1 A
- Position Feedback
  - No
- Cost
  - $11.95

- We chose to go with the SER0046 since it operates at the 5V required voltage, has a position feedback line, and has the lowest current requirements of all the options we looked at. Additionally, as a micro-servo, the motor has the smallest size making it easier to fit in the physical housing.

Hardware Diagram



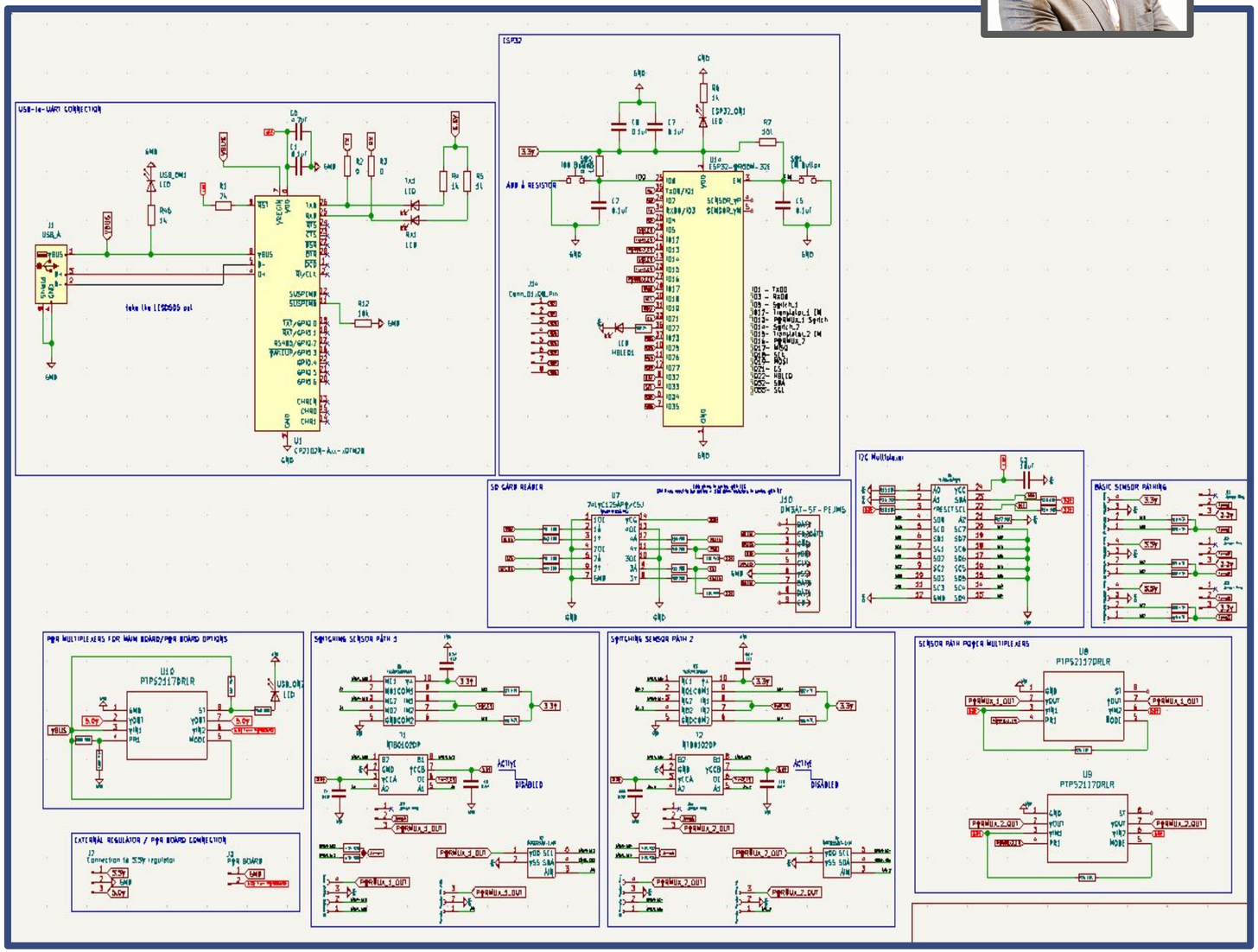
Hardware Design Diagram

- **Sensor Integration**
  All sensors connect directly to the MCU for data acquisition
- **Data Flow**
  Sensors → MCU → SD Card for storage and logging
- **Onboard Indicators**
  LEDs provide system status and heartbeat indication
- **User Controls**
  Reset and Boot buttons allow for system control and programming

Main PCB Schematic

Key Points of Interest in the Main Board:

- 5 individual I2C sensor ports
- A USB-A designed for both power supply and programming/serial capabilities
- Modular design connections to easily add/remove
    - 3.3V Regulator board
    - 5.0V Battery Power board
        - *Battery board is only required for portable device use*
- Jumper pins with easily removable jump-connectors allow users to add/remove on board pull-up resistors for each sensor port individually



Main Board Full Schematic

Power Calculations

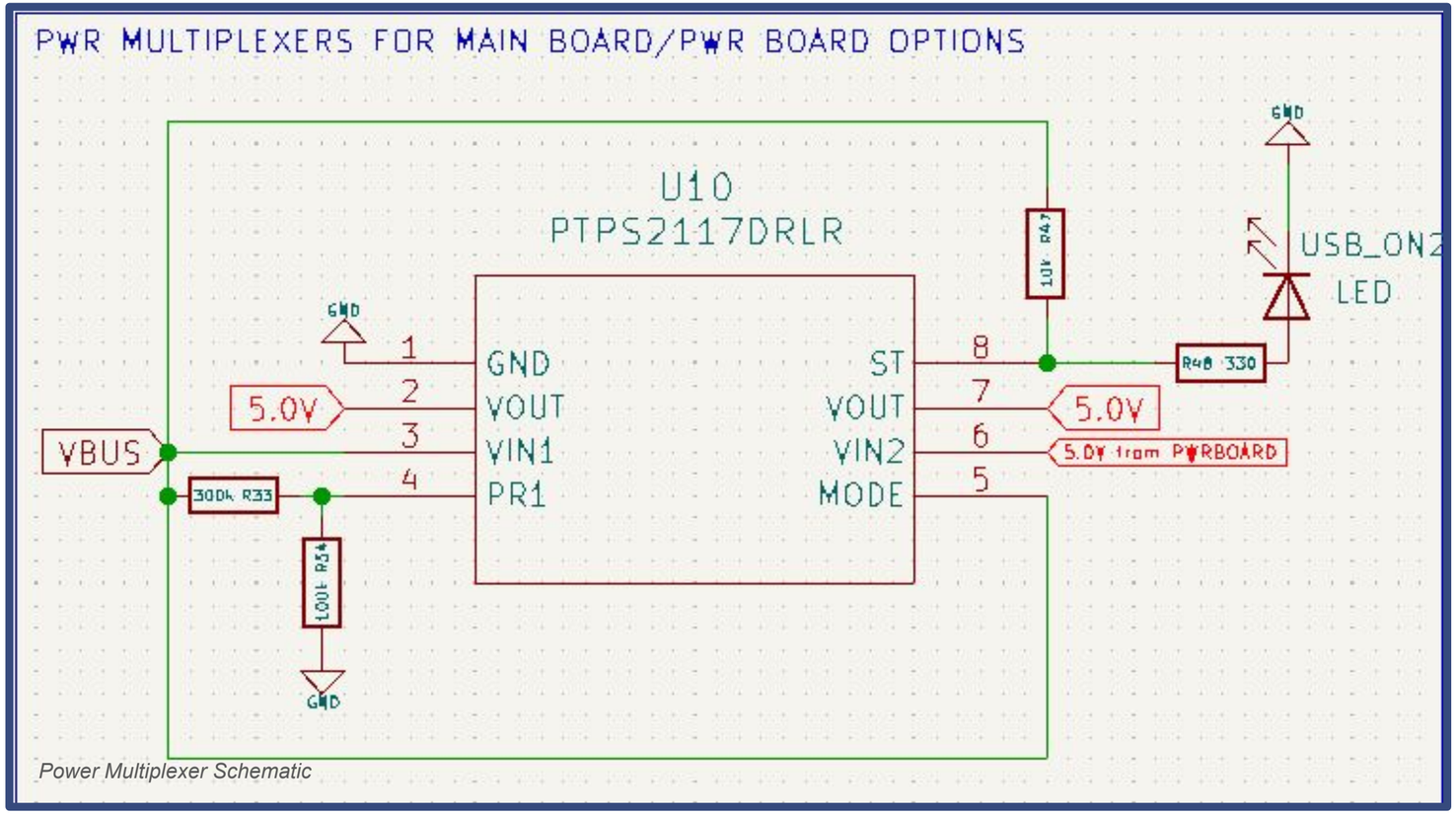
Part #	Component	Voltage	Typ. Current (A)	Power (W)
Main Board				
TPS2117	Power Multiplexer	1.6 - 5.5	4	8.00E-02
MCP3221	12-bit ADC	3.3	1.75E-04	5.78E-04
TS3A24157DGSR	SPDT switch		1.50E-08	
CP2102N-A02-GQFN28	USB TO UART BRIDGE		9.50E-03	
TCA9548APWR	8 channel I2C switch	5.5	5.00E-05	2.75E-04
74LVC125APW/C5J	Non-inverting buffer	3.3	1.00E-07	3.30E-07
ESP32-WROOM	MCU	3.3	5.00E-01	1.65E+00
NTS0102DP-Q100H	Voltage Level Translator		5.10E-06	1.43E-05
Sensors				
LMT87	Temperature Sensor	3.3	5.40E-06	1.78E-05
VEML7700	Light Sensor	3.3	8.00E-06	2.64E-05
HIH6130	Humidity Sensor	3.3	6.50E-04	2.15E-03
FSR402	Pressure Sensor	3.3	3.30E-07	1.09E-06
Monochromator				
ATtiny84	MCU	5	2.40E-03	1.20E-02
SER0046	Micro-Servo Motor	5	2.00E-01	1.00E+00

Power Calculations Table

Main PCB (Schematic) Layout

Why is this hardware important?

- This allows the board to automatically switch between the on board USB and battery power board based on which is active
- On board USB will take priority, even if battery board is active

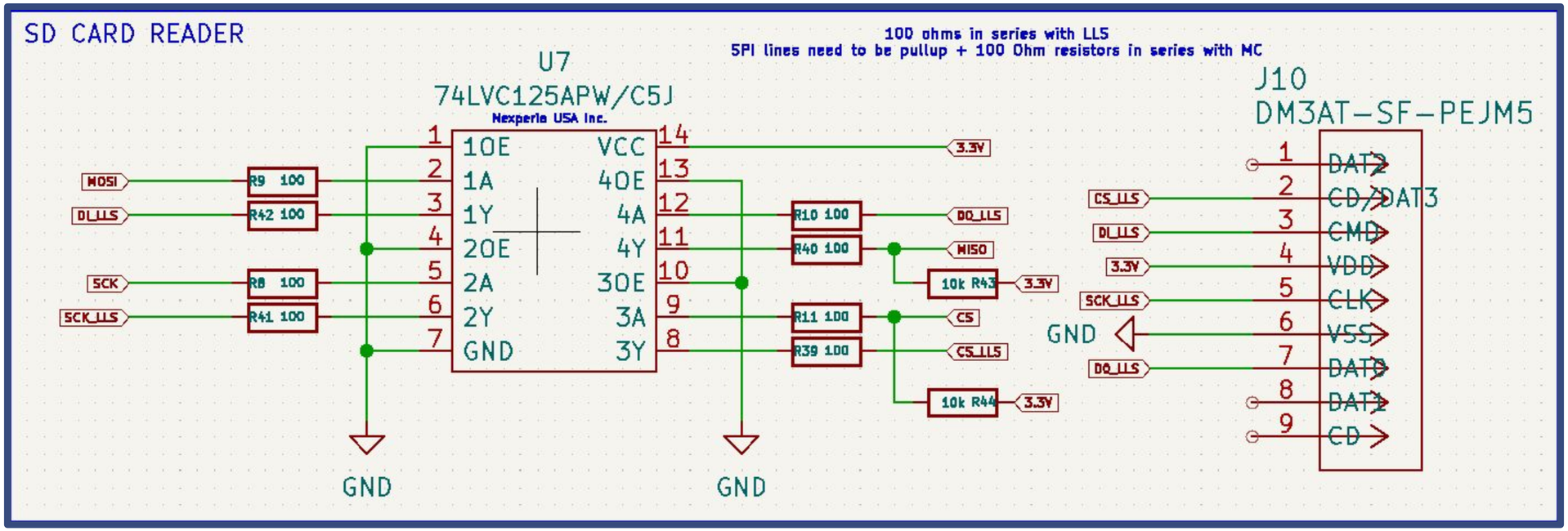


Power Multiplexer Schematic

Main PCB Schematic (SD Card Storage)

Why is this hardware important?

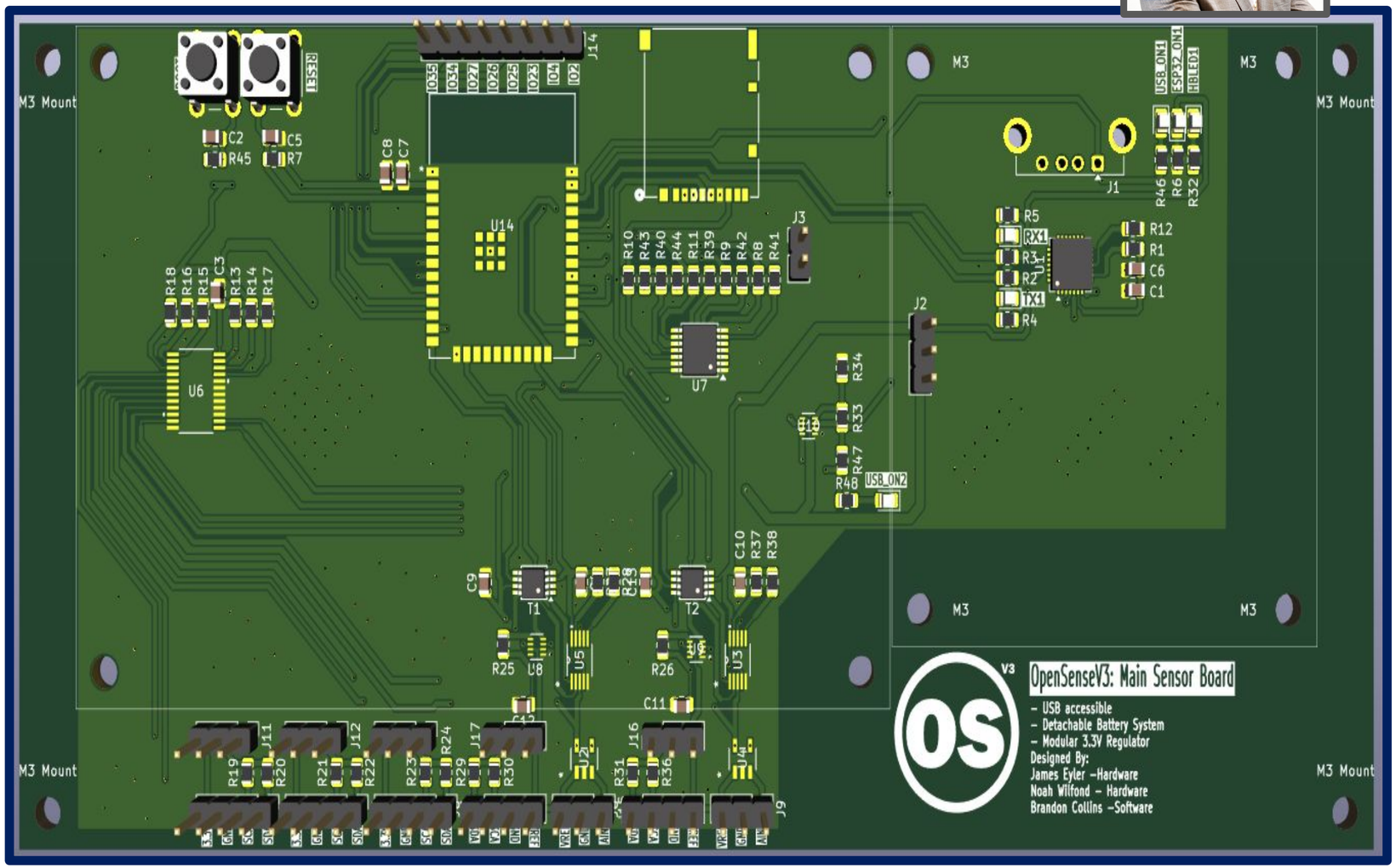
- Allows data collected to be stored and recalled by the user, either by downloading the .csv via the web server or removing the SD card from the slot on the board



SD Card Slot Schematic

Main PCB (MCU PCB) Layout

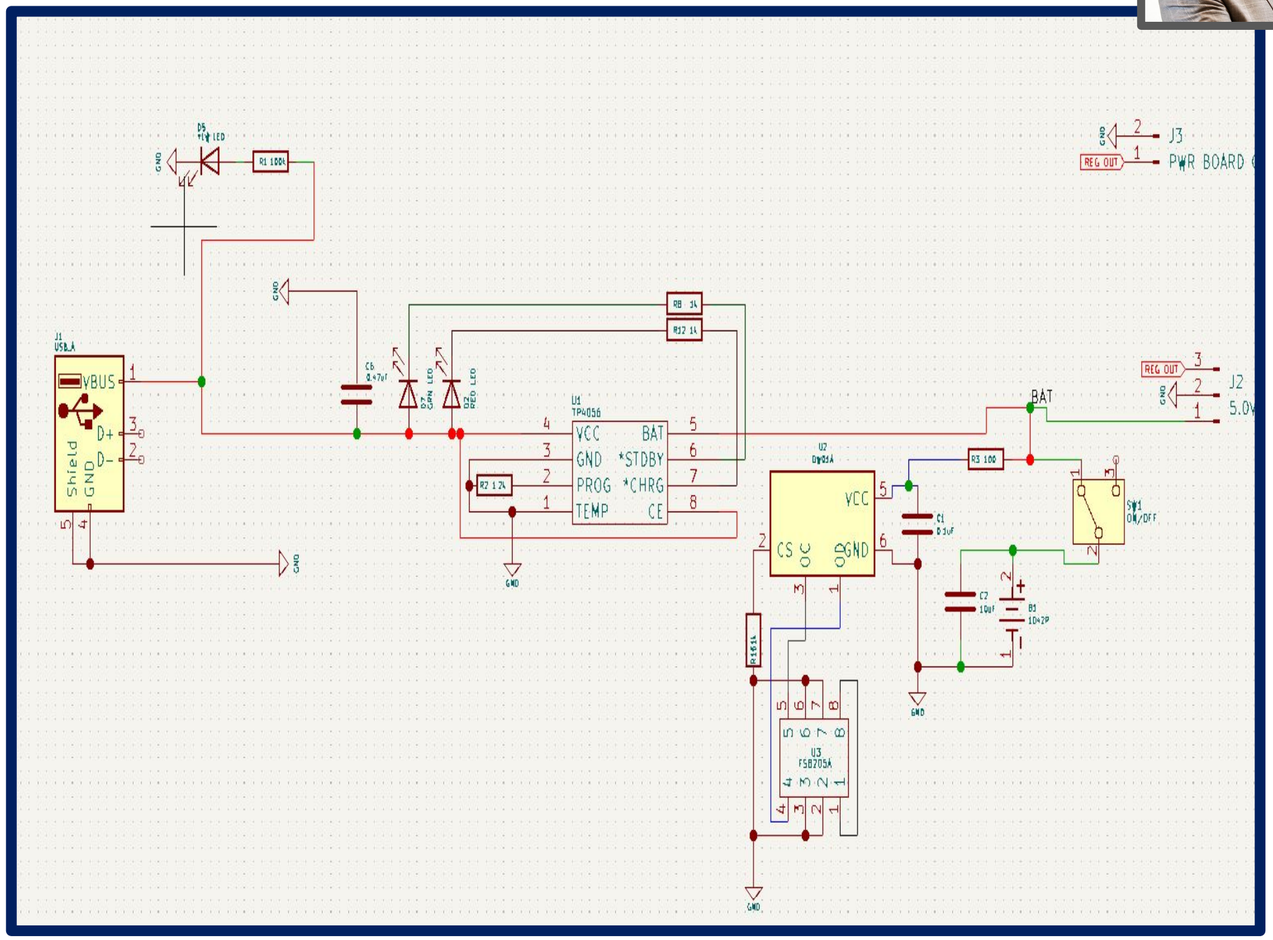
- **Compact Footprint**
  Total board area reduced to 133 cm² for a more efficient design
- **Accessible Sensor Layout**
  Sensor ports are positioned for easy access through the enclosure
- **Web Server Integration**
  Users can control sensor configurations remotely through the web interface



Main Board PCB Layout

Battery Charging Module Design

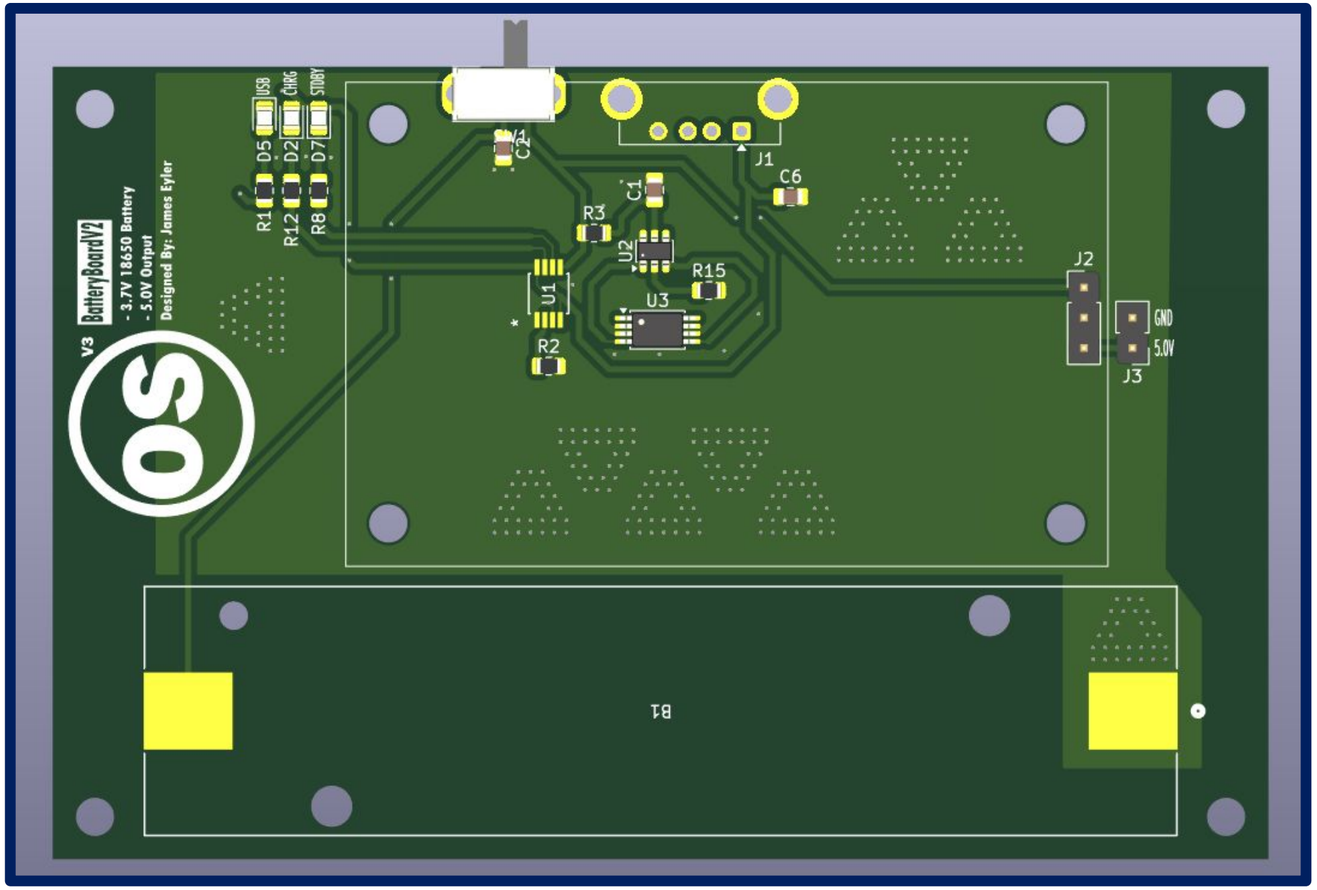
- **5.0V Boost Regulation**
  - Attachable boost regulator ensures a stable 5.0V output from the battery
- **Battery Management System**
  - Integrated charging and protection via TP4056 module
- **Overcharge & Thermal Protection**
  - Built-in IC safeguards against overcharging and excessive temperature
- **Power Control Switch**
  - On/Off switch fully disconnects the battery from the circuit
- **Safe and Reliable Operation**
  - Designed to protect both the system and the user during operation
- **Reference Design**
  - Based on the TP4056 battery charging module



Battery Board Schematic

Battery Charging Module Design

- **Portable Operation**
  - Enables the OpenSense platform to function without a wired power connection
- **Battery-Powered Flexibility**
  - Supports operation using a rechargeable 18650 lithium-ion battery
- **Onboard Charging Capability**
  - Allows users to charge the battery while the system is in use
- **Modular Design**
  - Battery board can be attached or removed depending on user needs
- **Improved Usability**
  - Expands the range of applications beyond stationary setups
- **Cost Flexibility**
  - System can operate without the battery board for lower-cost configurations



Battery Board PCB Layout

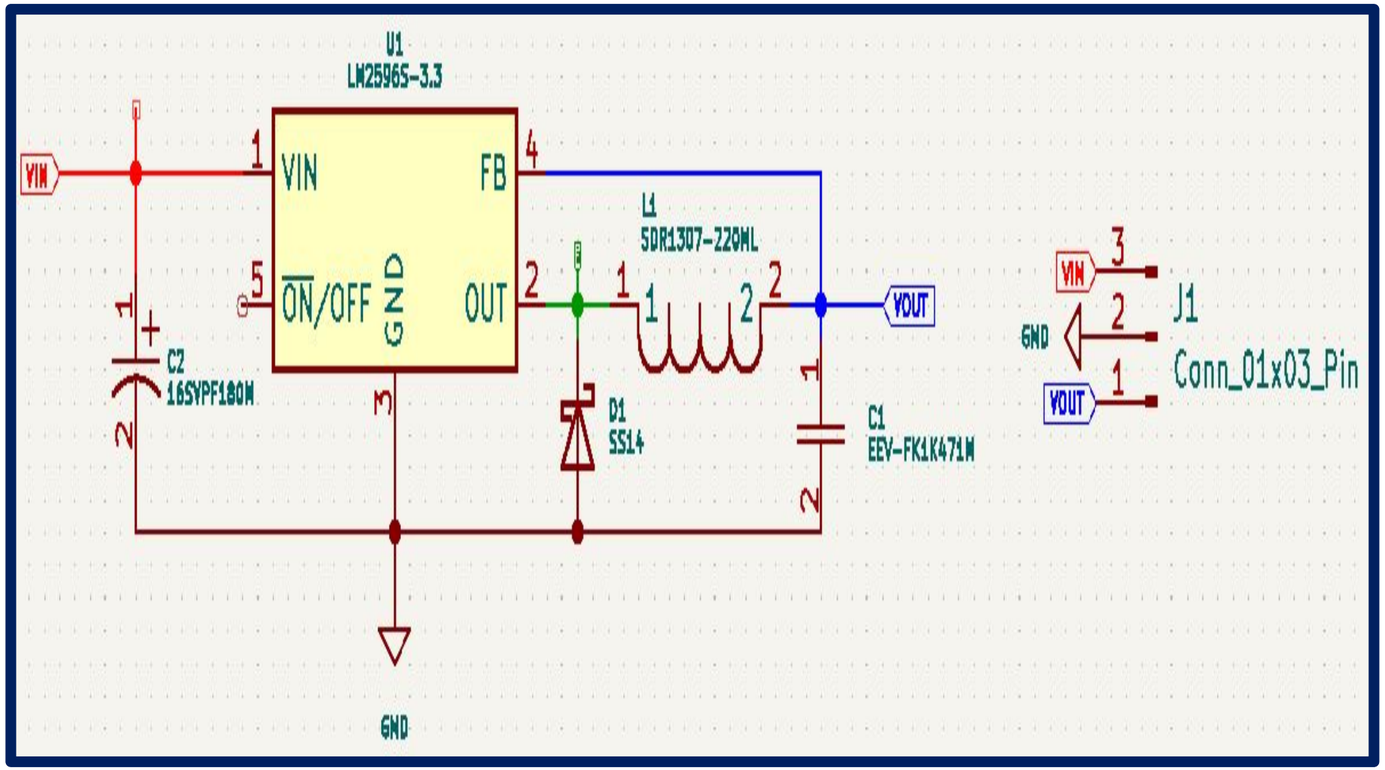
3.3V Regulator Design

Main Functions of the Regulator

- Provides proper voltage for logic-level components like the ESP32
- Enables full operation of the main board when powered via USB
- Ensures consistent performance across all low-voltage component

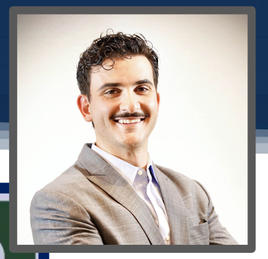
Design Considerations

- Power Planes Required
- Vin and Vout should use dedicated power planes for stability and current handling
- Follow Datasheet Layout
- Noise and Heat Management
- Proper layout helps reduce noise and improves thermal performance



3.3V Regulator Schematic

3.3V Regulator Design

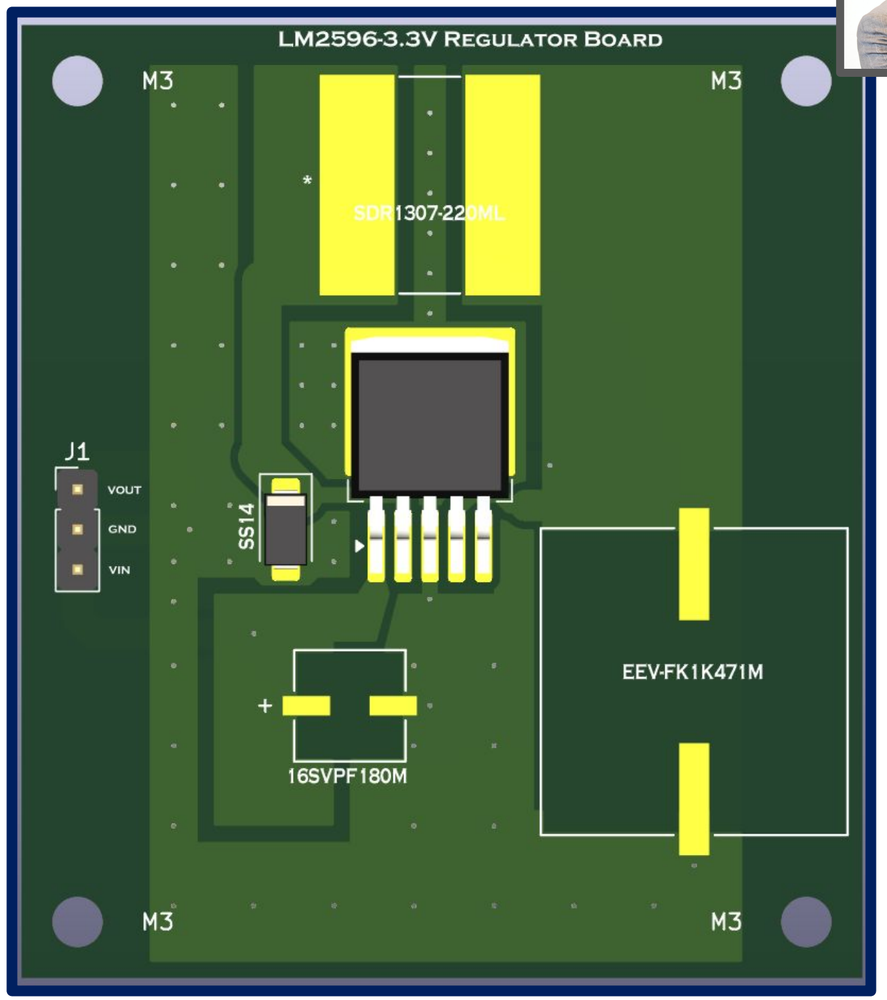


Main Functions of the Regulator

- Provides proper voltage for logic-level components like the ESP32
- Enables full operation of the main board when powered via USB
- Ensures consistent performance across all low-voltage component

Design Considerations

- Power Planes Required
- Vin and Vout should use dedicated power planes for stability and current handling
- Follow Datasheet Layout
- Noise and Heat Management
- Proper layout helps reduce noise and improves thermal performance



3.3V Regulator PCB Layout

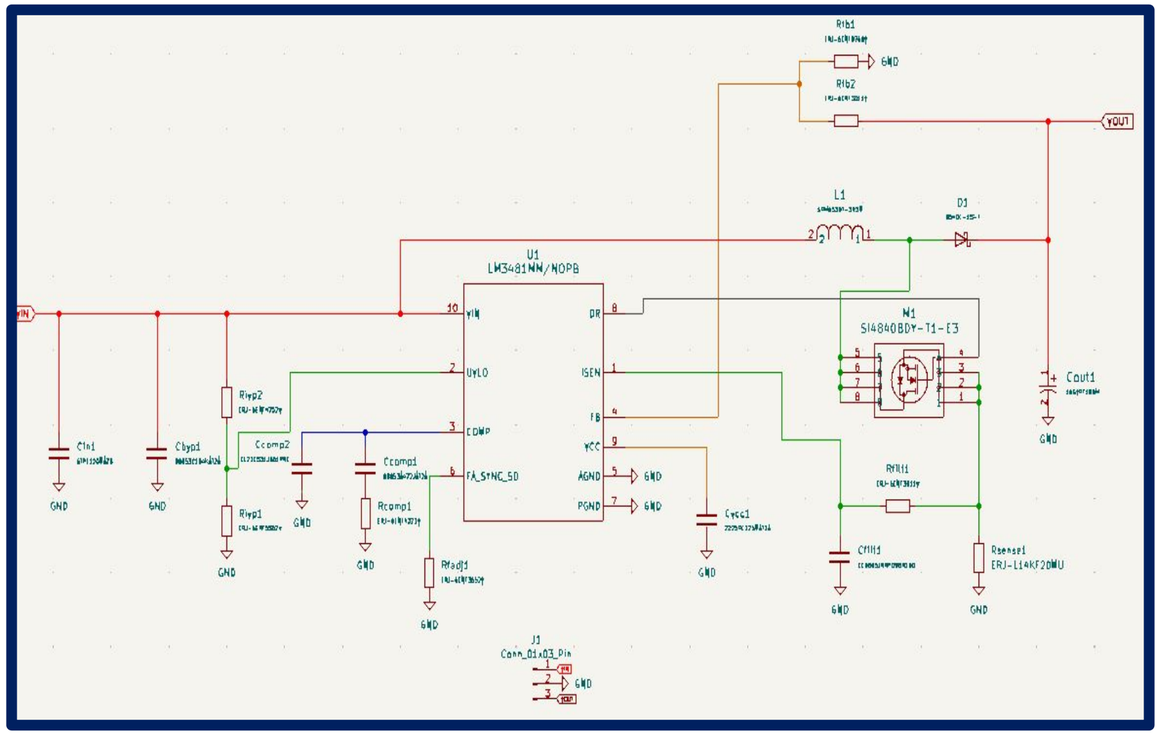
5.0V Regulator Design

Main Functions of the Regulator

- Maintains a constant 5.0V supply for the system
- Accepts fluctuating voltage from the 18650 Li-ion battery and USB-A input
- Ensures reliable operation regardless of input source variations

Design Considerations

- Vin and Vout should use proper power planes for stability and current handling
- PCB layout should match manufacturer recommendations
- Proper design helps minimize losses and heat generation



5.0V Regulator Schematic

Caption Placeholder

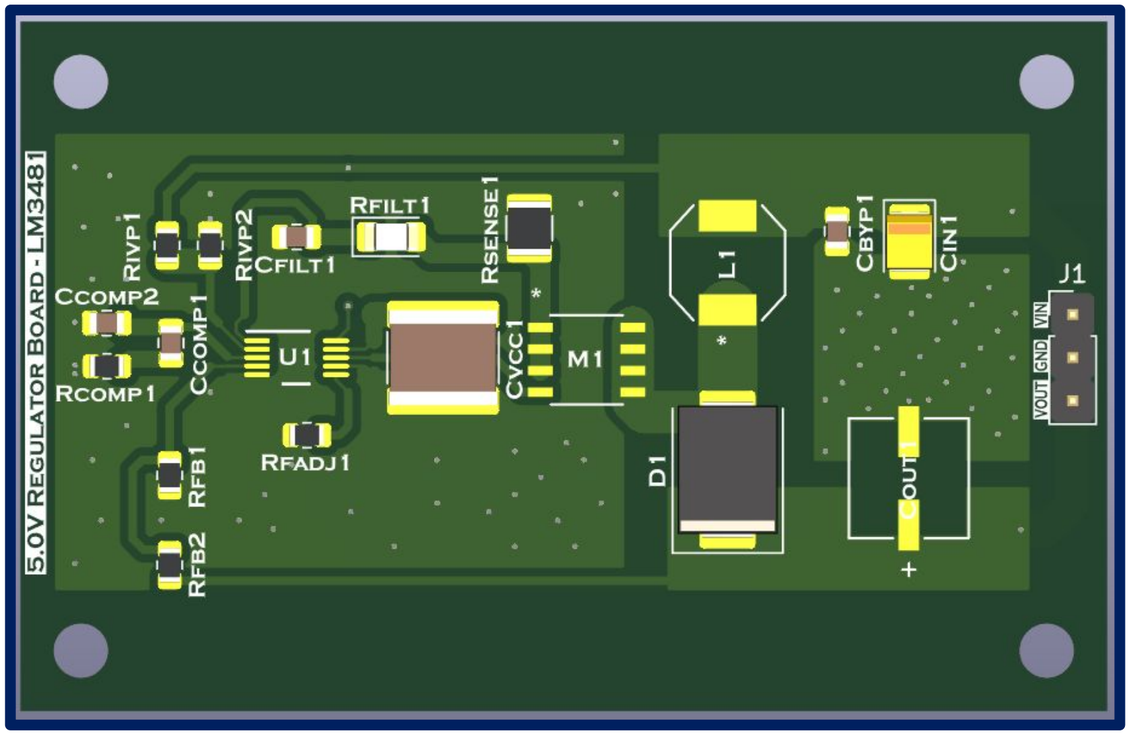
5.0V Regulator Design

Main Functions of the Regulator

- Maintains a constant 5.0V supply for the system
- Accepts fluctuating voltage from the 18650 Li-ion battery and USB-A input
- Ensures reliable operation regardless of input source variations

Design Considerations

- Vin and Vout should use proper power planes for stability and current handling
- PCB layout should match manufacturer recommendations
- Proper design helps minimize losses and heat generation

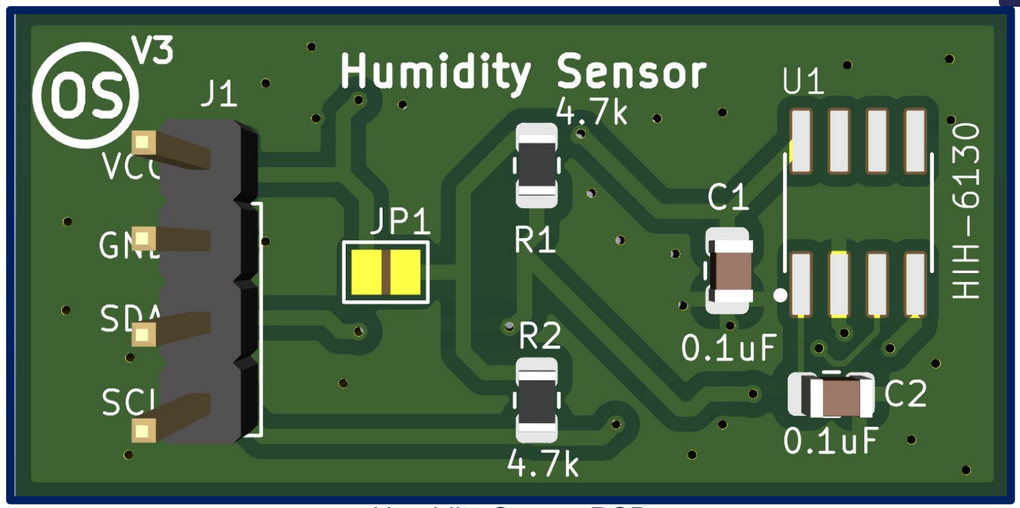


5.0V Regulator PCB Layout

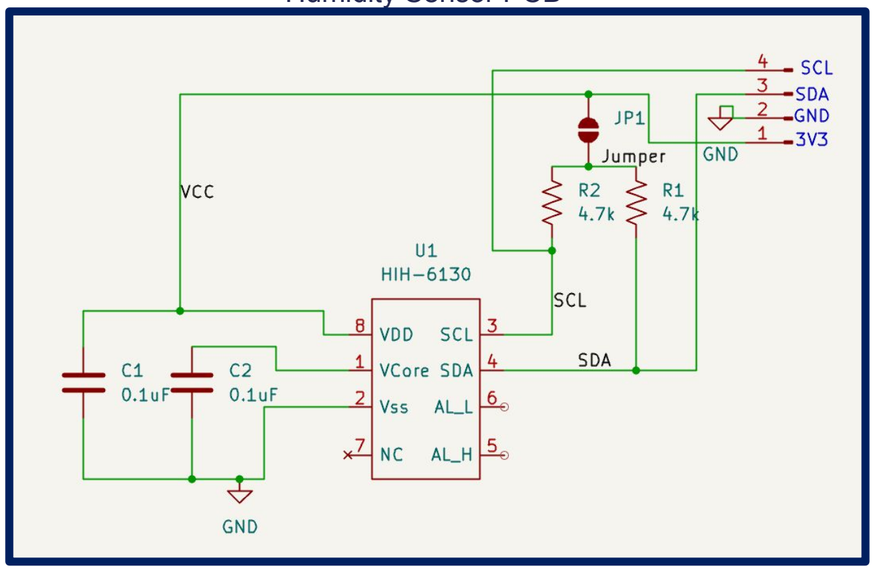
Caption Placeholder

Humidity Sensor Design

- **HIH-6130 Humidity Sensor**
  Uses the Honeywell HIH-6130 to provide digital humidity sensing through an I2C-based daughterboard design
- **I2C System Integration**
  Connects to the main board through SDA and SCL, enabling efficient communication with minimal wiring
- **Noise Reduction Components**
  Pull-up resistors and decoupling capacitors were included to improve communication stability and sensor reliability
- **PCB Layout Considerations**
  Routing, spacing, and component placement were selected to support clean signal paths and a compact layout
- **Ground and Layer Optimization**
  Ground nodes were placed carefully, and the second PCB layer was used to create smoother and less congested routing
- **Jumper-Based Bus Flexibility**
  An I2C jumper was added to support easier integration with systems that may already include pull-up resistors on the main board

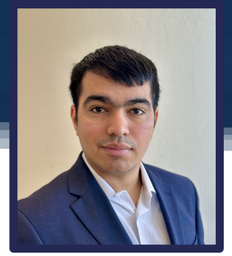


Humidity Sensor PCB

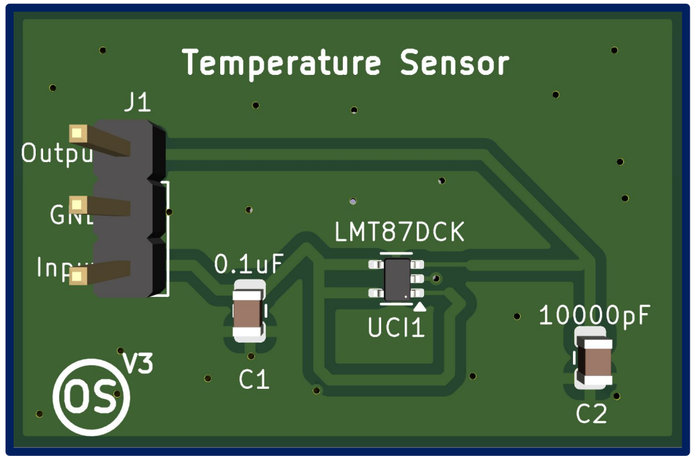


Humidity Sensor schematic

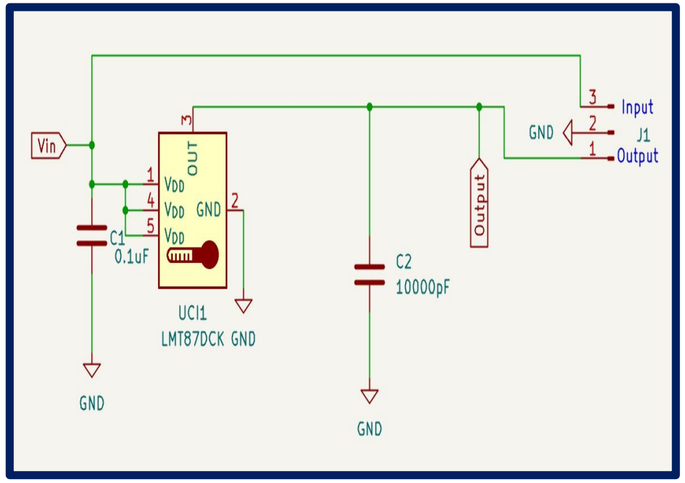
Temperature Sensor Design



- **LMT87 Temperature Sensor**
  Utilizes the LMT87DCK for accurate analog temperature measurement
- **Signal Stability Components**
  Includes decoupling and output capacitors for noise reduction and stable readings
- **Analog Interface**
  Outputs temperature data through an analog connection to the main board
- **Proper PCB Routing**
  Trace widths selected to ensure reliable signal integrity
- **Component Spacing**
  Layout designed to minimize interference and improve measurement accuracy



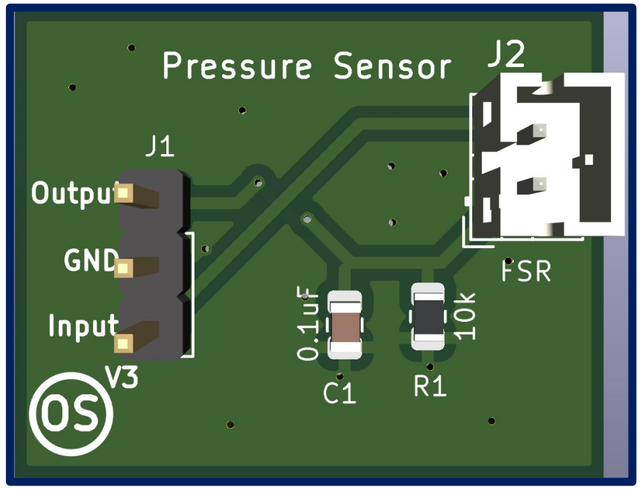
Temperature Sensor PCB



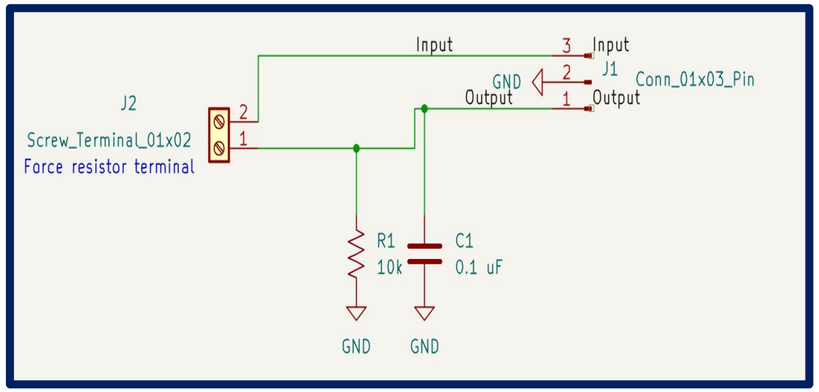
Temperature Sensor Schematic

Pressure Sensor Design

- **Force Sensitive Resistor (FSR)**
  Uses the 34-00015 FSR / FSR 402 to detect applied pressure through resistance changes
- **Voltage Divider Configuration**
  Paired with a 10kΩ resistor to convert resistance changes into a readable voltage,with an additional 1.2kΩ variant evaluated during design exploration.
- **Signal Stability Components**
  Includes a decoupling capacitor for smoother and more reliable output
- **Modular Connector Design**
  Uses a pin connector interface instead of direct PCB mounting for flexibility and stability
- **Optimized PCB Layout**
  Proper trace width and spacing used to ensure reliable performance
- **3D Printed Housing**
  Custom enclosure improves consistency and accuracy of pressure readings



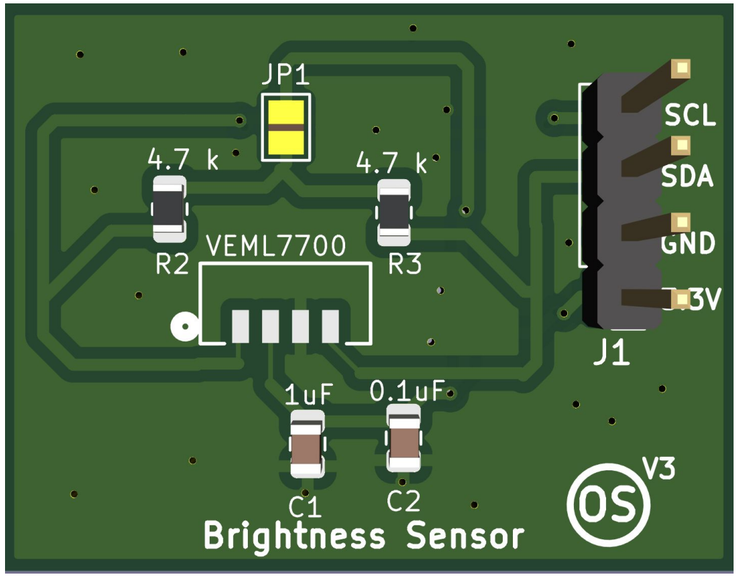
Pressure Sensor PCB



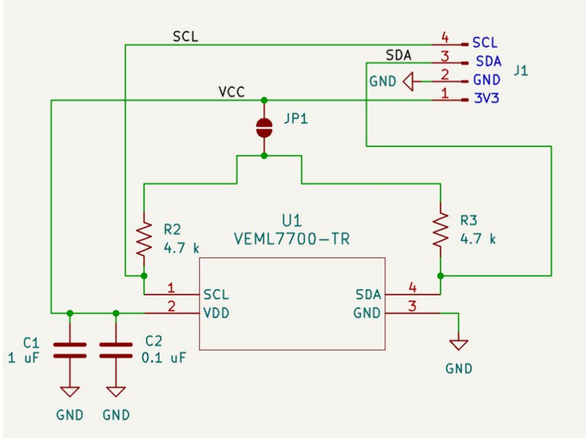
Pressure Sensor Schematic

Brightness Sensor Design

- **VEML7700 Brightness Sensor**
  Uses the VEML7700 ambient light sensor to provide digital brightness measurements for the OpenSense platform
- **I2C System Integration**
  Connects to the main board through SDA and SCL for efficient and reliable digital communication
- **Noise Reduction Components**
  Pull-up resistors and local decoupling capacitors were included to improve signal stability and reduce electrical noise
- **Jumper-Based Bus Flexibility**
  An I2C jumper was added to support easier integration with systems that may already include pull-up resistors on the main board
- **PCB Layout Considerations**
  Routing, spacing, and placement were selected to support clean signal paths and a compact board design
- **Ground and Layer Optimization**
  Grounding was placed carefully, and the second PCB layer was used to create smoother routing and a cleaner overall layout



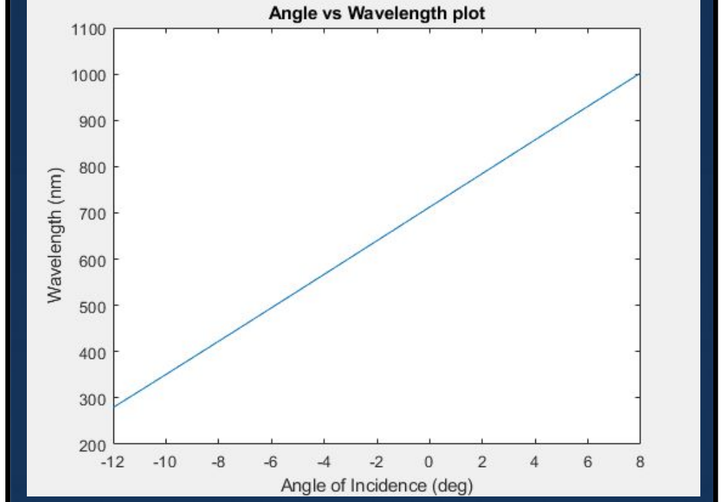
Brightness Sensor PCB

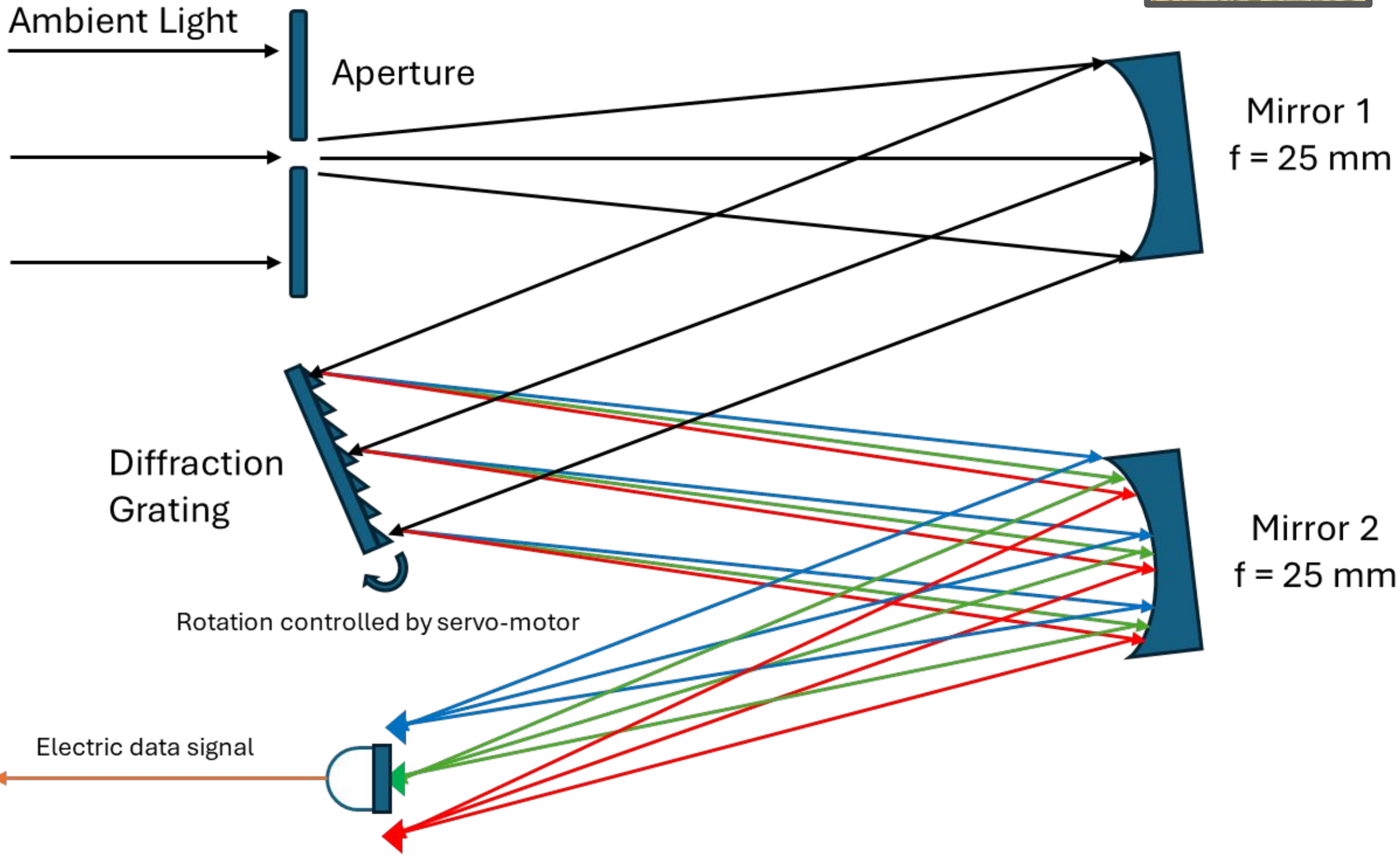


Brightness Sensor Schematic

Monochromator - optical design

MATLAB Calculations



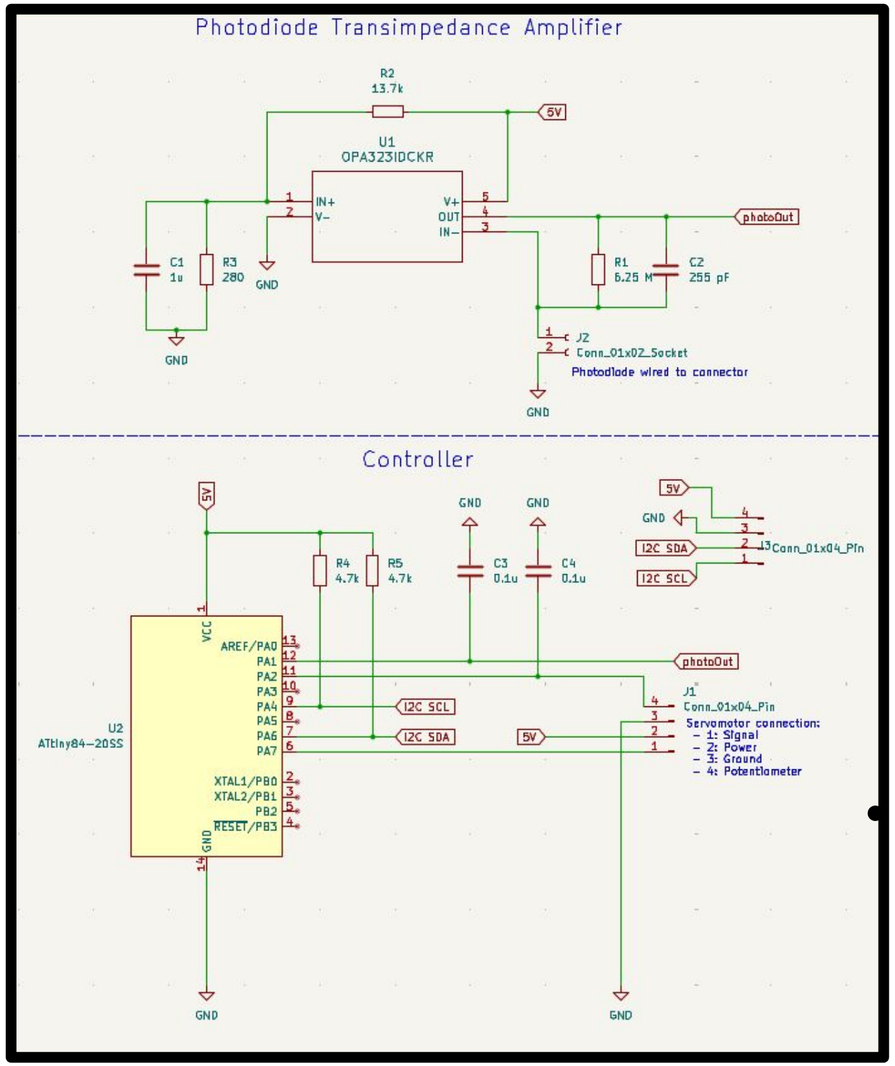


User readable data

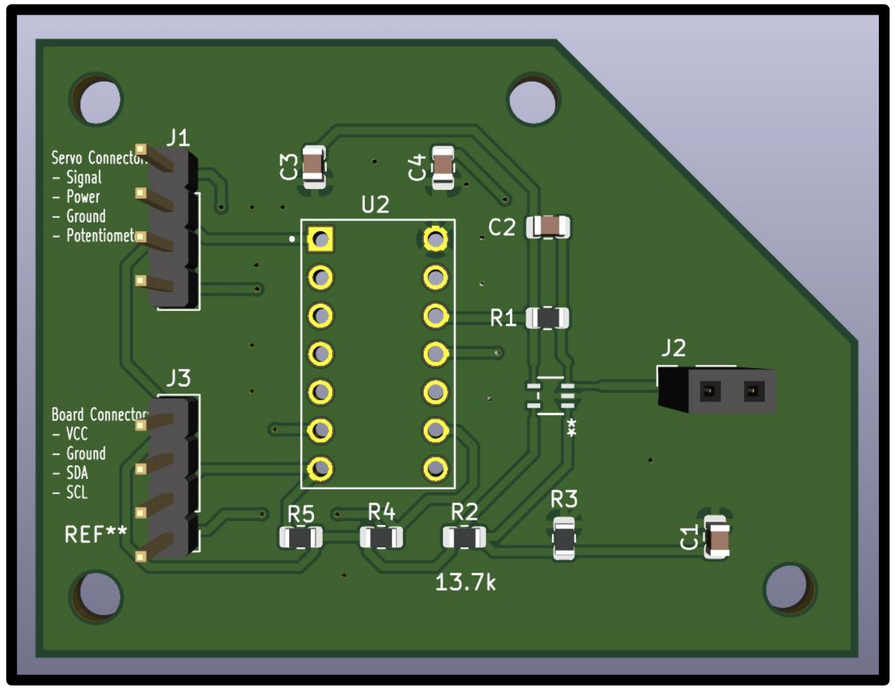
OpenSense Platform

Electric data signal

Monochromator - electrical design



Schematic for the monochromator PCB. Top half shows the transimpedance amplifier, and bottom half the motor controller.



PCB Layout for the monochromator.

Three main functions: Transimpedance amplifier, servo controller, and I2C slave

- **Transimpedance amplifier** built around OPA323 Op-Amp converts photocurrent from the TEMD5080x01 photodiode into a measurable voltage.
- **Controller** uses ATtiny84 to manage servo motor rotation with pulse width modulation and records outputs from both the transimpedance amplifier and potentiometer on 10-bit ADC.
  - Transmits optical and rotation data to the sensor platform via I2C to OpenSense v3 platform for processing into spectrum

Choosing a Programming Environment

ESP-IDF
Native Framework

- C/C++ implementation
- Full system control
- Built-in HTTP, FAT, NVS
- Production-ready stability
- FreeRTOS foundation

ESP-IDF

Arduino Core
Rapid Development

- C/C++ with abstraction
- Extensive libraries
- Fast prototyping
- Community support
- IDF compatibility

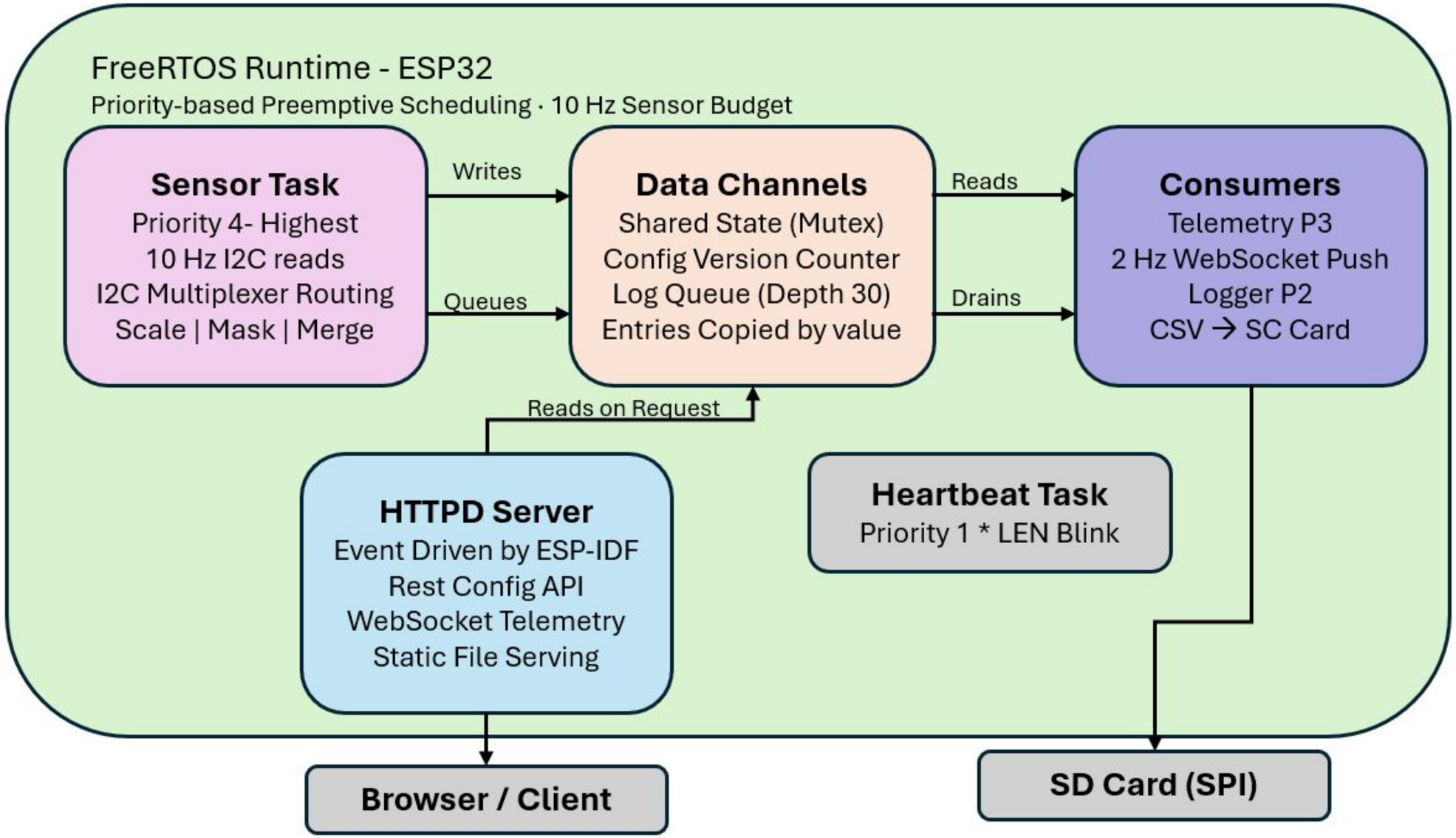
ARDUINO™

MicroPython
Scripting Option

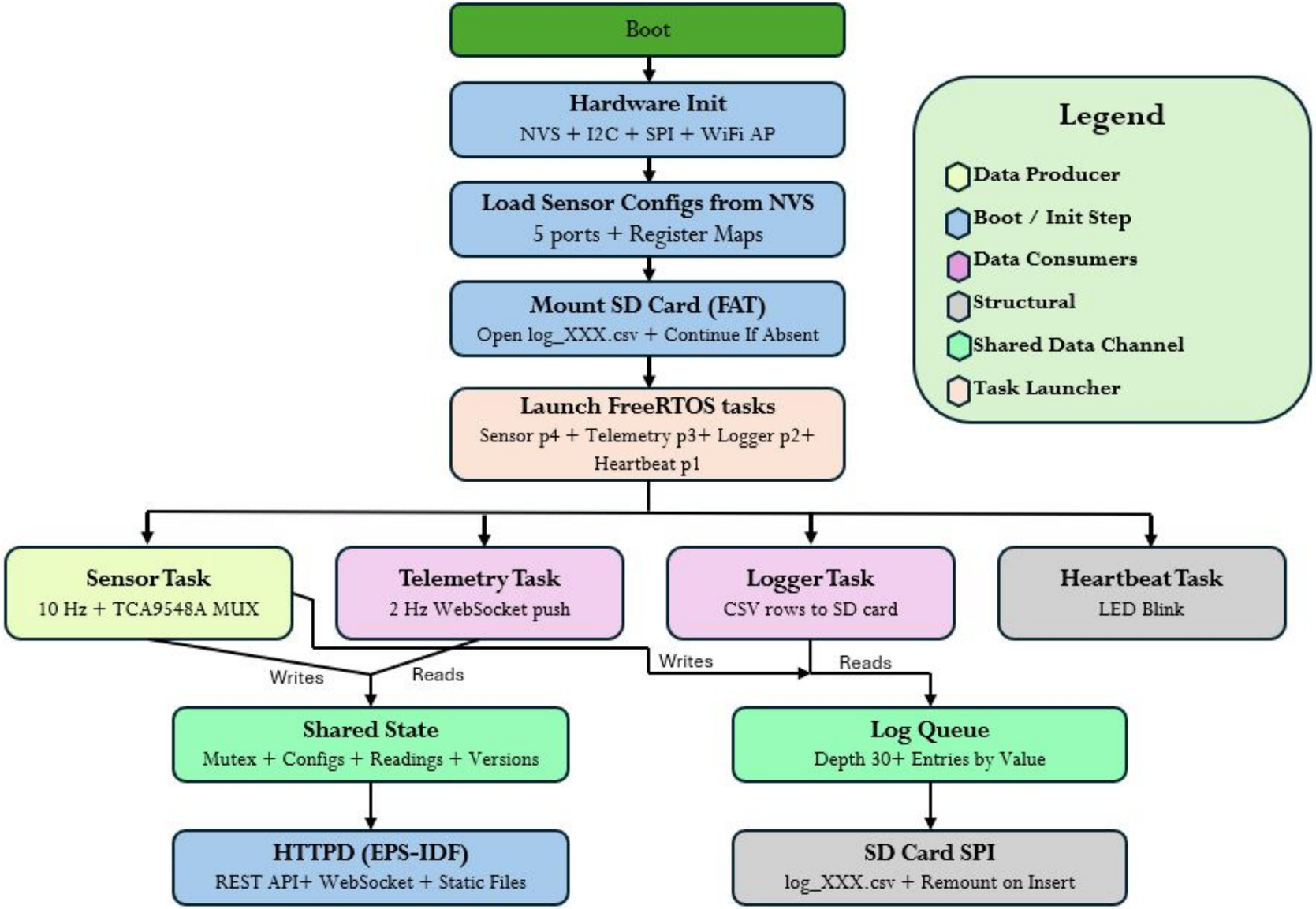
- Python implementation
- Fastest iteration
- Readable code
- Less deterministic
- Lower throughput

MicroPython

Software Architecture



Main Program Flow



Use Case Diagram

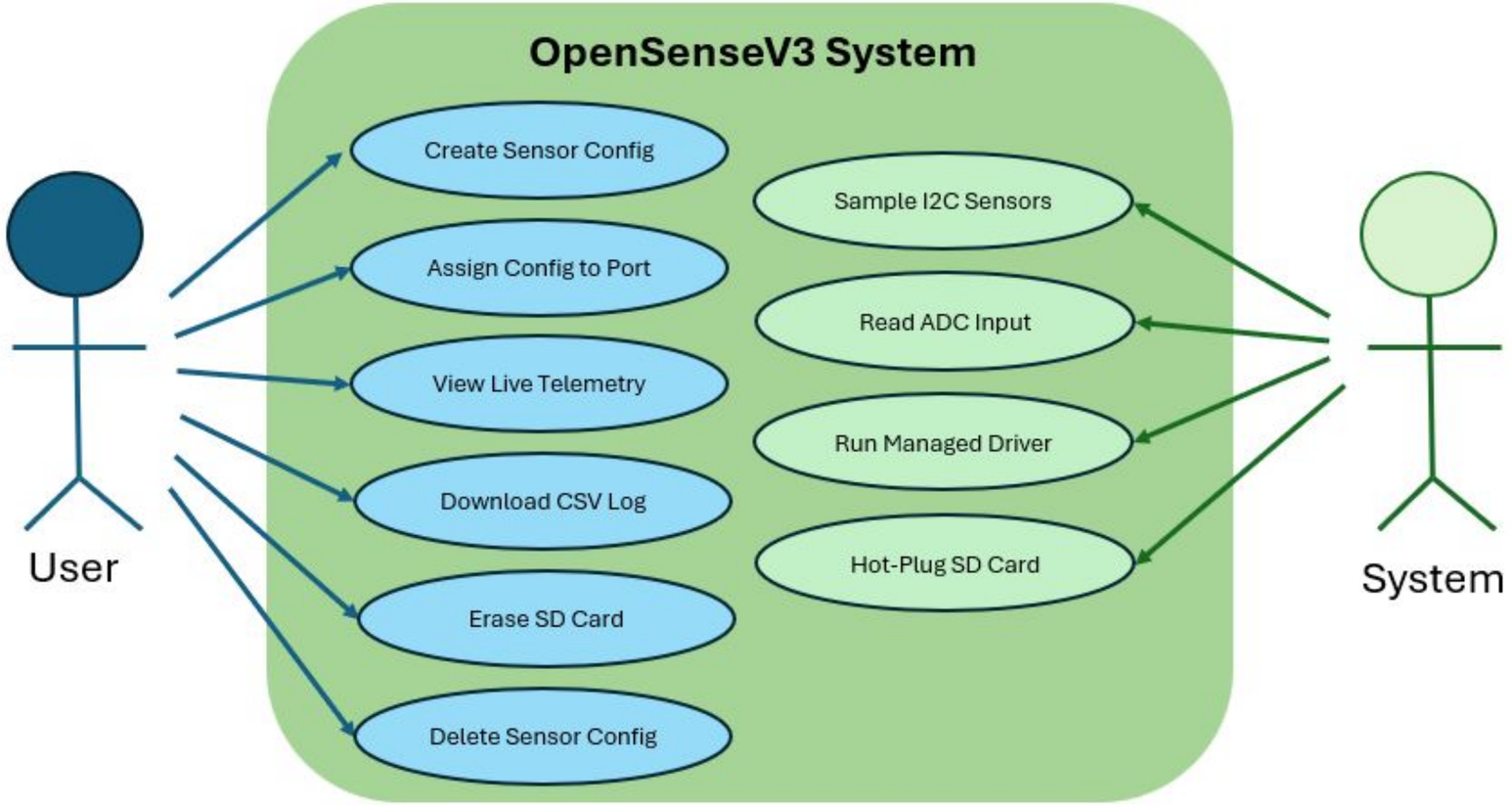


Illustration shows the two actors that interact with the OpenSense system and what each can do. The user (via browser) initiates configuration, monitoring, and data management tasks, while the sensor hardware drives the sampling, and logging that happen autonomously at runtime.

Hybrid Driver Approach

Generic I2C Driver

- User-configurable through the web UI
- Handles any standard register-read sensor
- Supports scaling, masking, shifting, and register merging
- Expressions allow custom calibration formulas
- Works for I2C and ADC senos without firmware changes

Preset Drivers

- Trigger-before-read sequences handled automatically
- Built in CRC validation and factory calibration
- Address dropdowns replace free text fields in UI
- Currently Supports SHT30, BME280, and HIH6130

Both driver types share the same sensor task, MUX routing, data channels, and CSV logging pipeline.

Limitations of the Generic Driver

No multi-step transactions

The generic driver performs one write-then-read per register. Sensors requiring a trigger write followed by a delay and then a separate read transaction — such as the HIH6130 — cannot be expressed in the register config and require a dedicated preset driver.

No CRC or checksum validation

Raw bytes are accepted as-is. Sensors like the SHT30 that append a CRC byte after each measurement value need a preset driver to validate data integrity before publishing readings.

Write buffer limited to 4 bytes

The write-before-read buffer is capped at four bytes. Sensors requiring longer initialisation sequences, such as multi-register configuration writes, cannot be fully initialised through the generic driver alone.

When a sensor exceeds these constraints a dedicated preset driver is the correct solution.

User Interface - Dashboard Tab

OpenSense V3 | Dashboard | Ports | Configs | SD: Logging | Live

Live Telemetry

10 Hz sample rate

144.9 s uptime

SD Logging | Logging Active | 30436 MB | log_0025.csv | 1,419 rows | ~83.1 KB

Refresh | Download CSV | Erase & Reset

Rows this session 1,419 rows

Unconfigured
Disabled · CH1
Port disabled

Unconfigured
Disabled · CH2
Port disabled

Unconfigured
Disabled · CH3
Port disabled

Unconfigured
Disabled · CH4
Port disabled

Unconfigured
Disabled · CH5
Port disabled

User Interface - Port Tab

OpenSense V3 Dashboard Ports Configs SD: Logging Live

Port Assignment

Assign a saved config to each hardware port

Port 1
I2C · 3.3V · CH1
ASSIGNED CONFIGURATION
No config assigned
Assign Config

Port 2
I2C · 3.3V · CH2
ASSIGNED CONFIGURATION
No config assigned
Assign Config

Port 3
I2C · 3.3V · CH3
ASSIGNED CONFIGURATION
No config assigned
Assign Config

Port 4
I2C or ADC · CH4
ASSIGNED CONFIGURATION
No config assigned
Assign Config

Port 5
I2C or ADC · CH5
ASSIGNED CONFIGURATION
No config assigned
Assign Config

User Interface - Port Tab

OpenSense V3 Dashboard Ports Configs SD: Logging Live

Saved Configurations

Build named sensor templates, then assign to ports

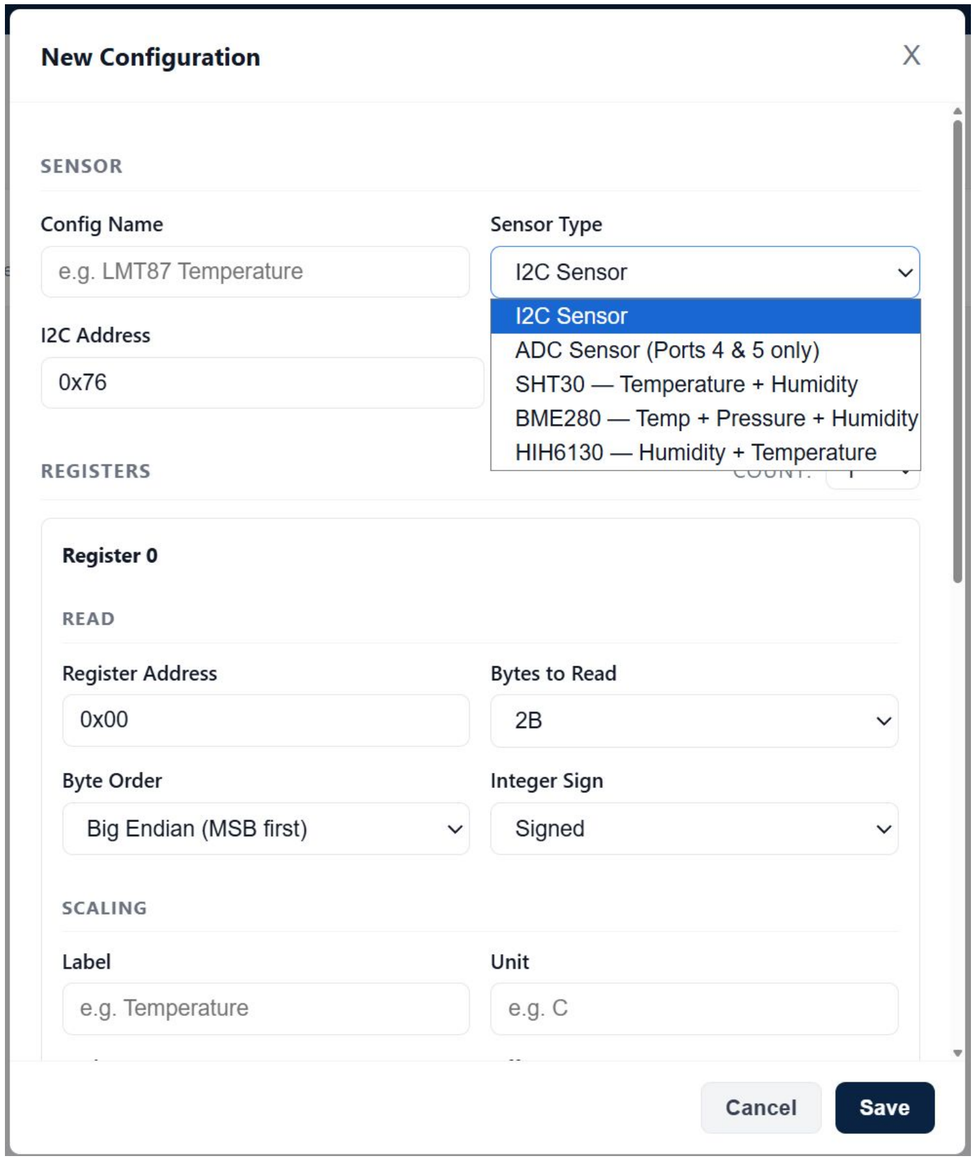
+ New Config

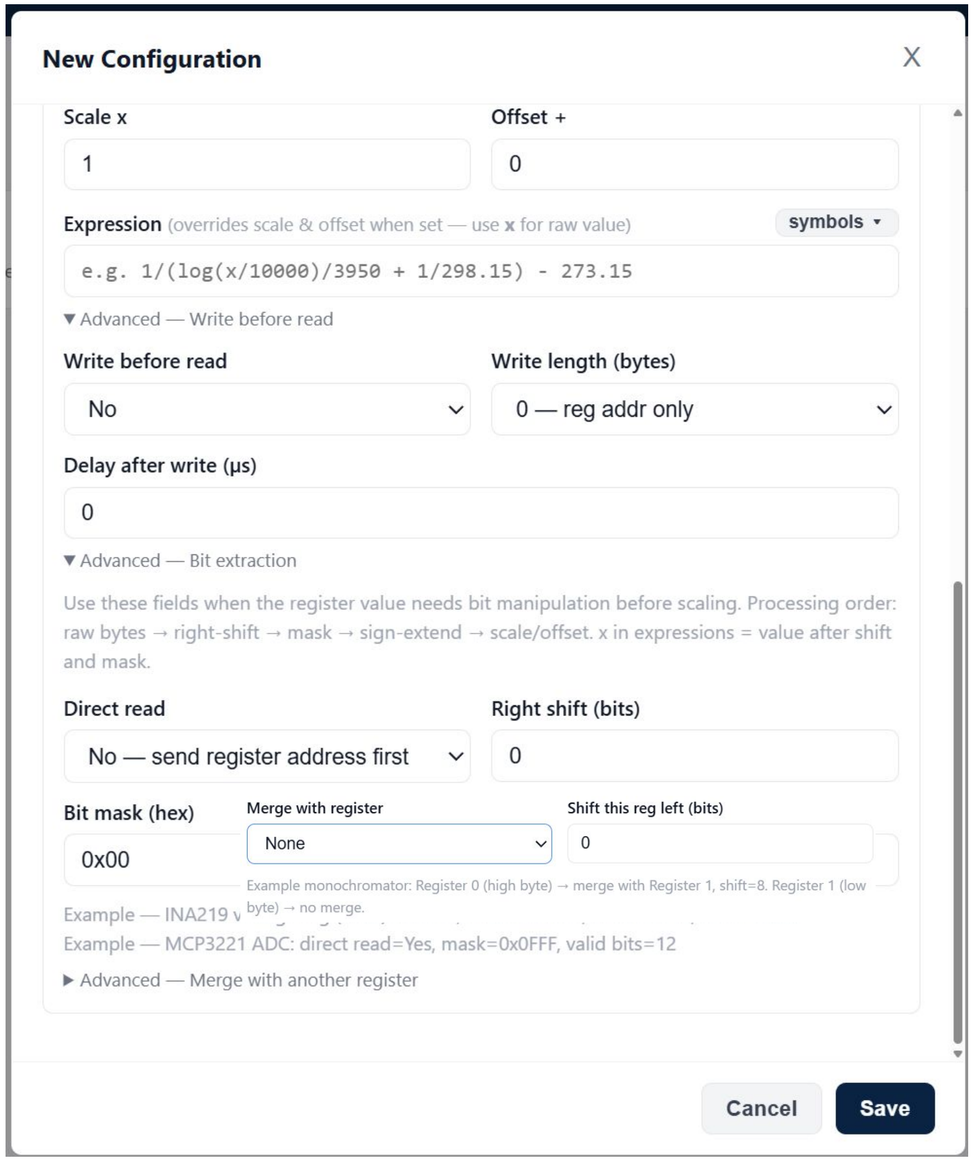
Monochromator

I2C 0x08 · 4 regs · Photodiode, Reg, Wavelength, Reg

Edit Delete

Create Configurations





There is also the final Advanced drop down that allows the user to merge register bytes for sensors that send values in two separate registers

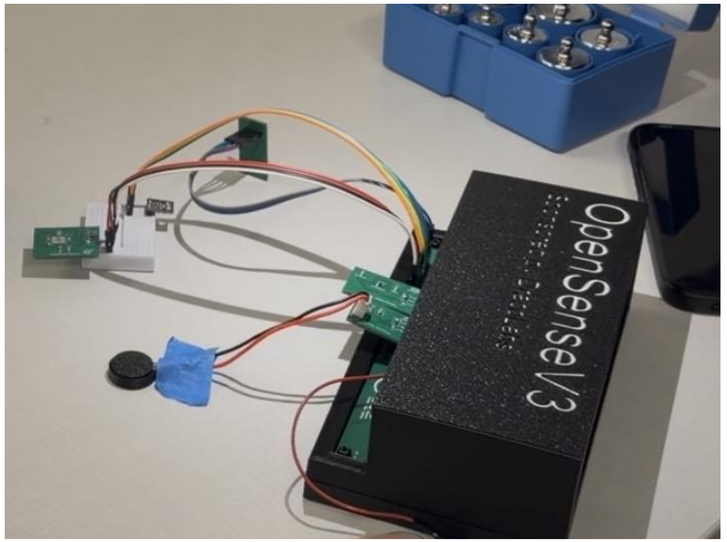
Finished Board Features:I2C pull ups and jumpers

- **Integrated I2C Pull-Ups**
  Main board includes pull-up resistors to support communication with I2C sensor modules
- **Configurable Jumper Control**
  Jumpers were added so the onboard pull-ups can be enabled or disabled as needed
- **Default Bus Support**
  Pull-ups are enabled by default to simplify initial sensor integration
- **Sensor Compatibility Flexibility**
  Jumpers allow pull-ups to be removed when a connected sensor already includes its own resistors
- **Bus Conflict Prevention**
  This design helps avoid excessive pull-up loading on the I2C lines when multiple devices are connected
- **Humidity Sensor Example**
  The humidity sensor was used with its pull-ups enabled to demonstrate this feature in the finished system

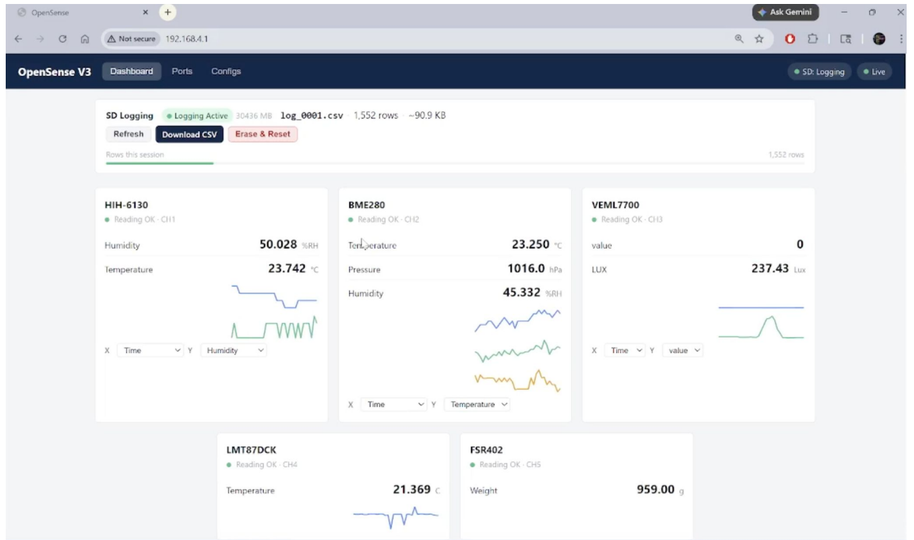
I2C jumper pins with their caps on.

Sensor Showcase

- **Five-Sensor Operation**
  Demonstrates that the platform can support all five sensors at the same time
- **Real-Time Dashboard Display**
  Sensor readings are shown simultaneously on the website interface
- **Integrated System Functionality**
  Confirms successful communication between the main board and connected sensor modules
- **Individual Sensor Visibility**
  Each sensor's output can be monitored separately through the UI
- **Platform Validation**
  Demonstrates full multi-sensor operation in the finished system

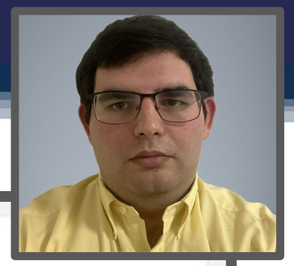


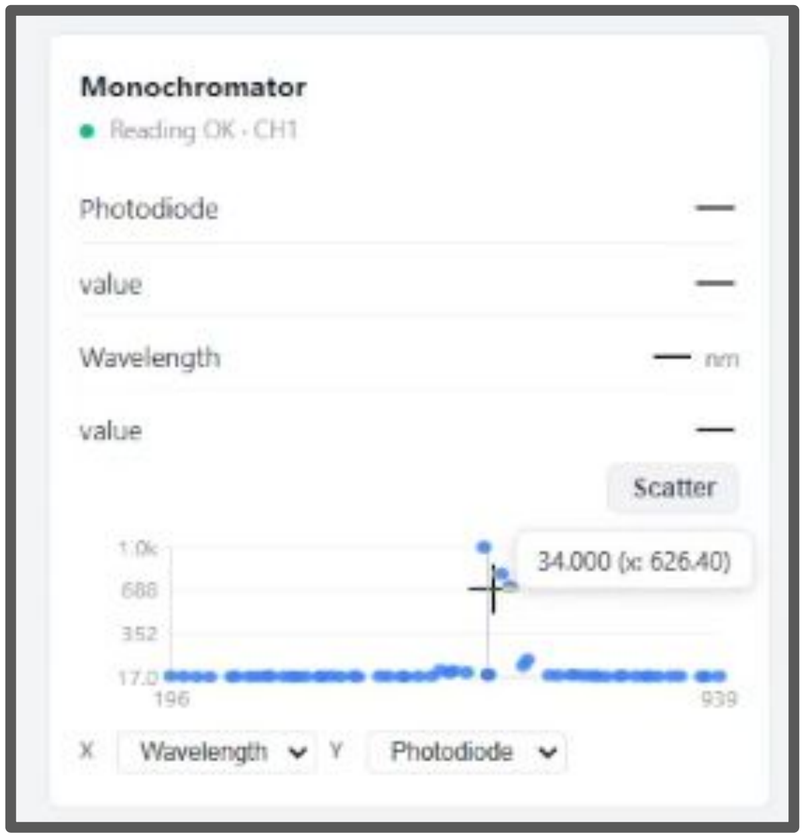
Here we have 5 sensors plugged into the mainboard



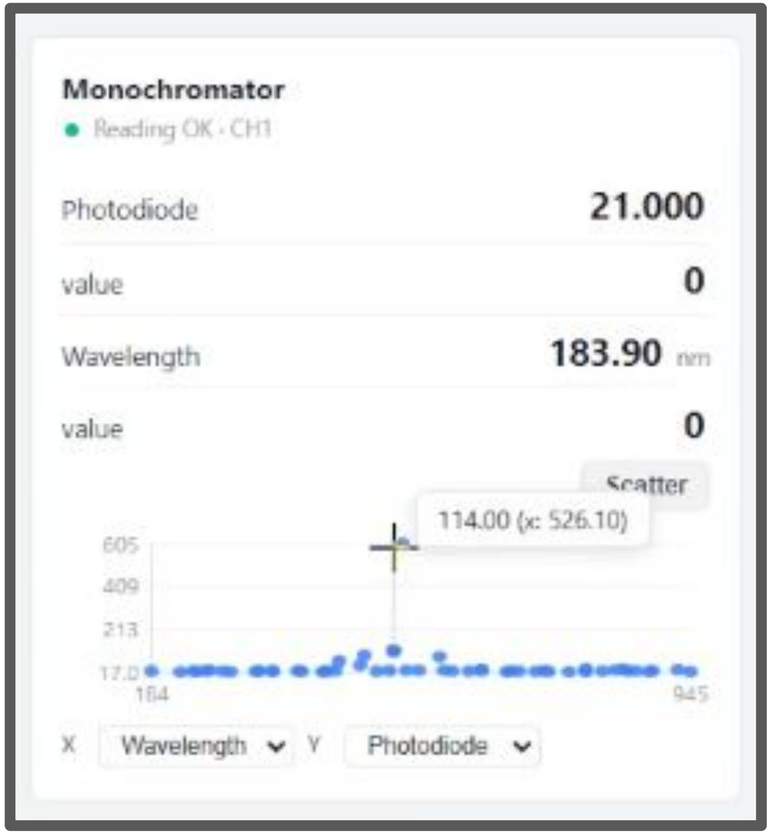
And this is how the 5 sensors are displayed on our website

Monochromator Showcase

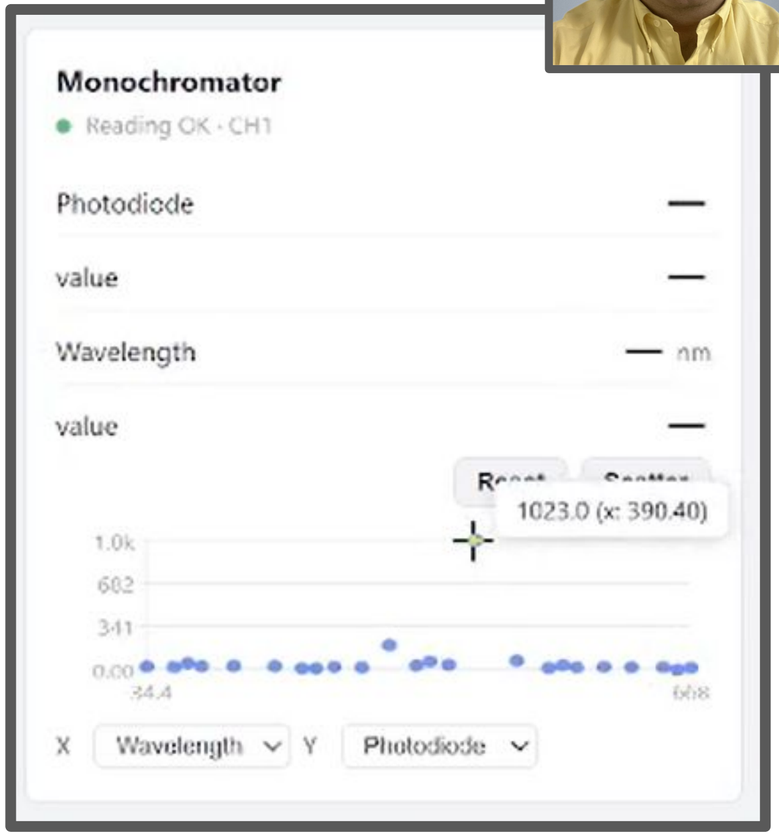




Red Spectrum displayed by OpenSense v3 platform.



Green Spectrum displayed by OpenSense v3 platform.



Blue Spectrum displayed by OpenSense v3 platform.

To test the monochromator, I used red (650 nm), green (525 nm), and blue (405 nm) lasers since their spectra are well-known and together cover the full spectral range of the device

Administrative Content

Name	MCU Board	Power system	Sensors	Software
Brandon				X
James (PM)	X	X		
Daniel			X	
Noah		X	X	

Administrative Content

Budget

Item	Cost
Main platform	~$300
Power System	~$60
Sensors	~$200
Monochromator	~$230
PCBs	~$800
Total	$1590