Engineering Portfolio

Haya Dakhil, BASc Nanotechnology Engineering

Concentray PCB Development (V1 → V2) - Solo Electronics Engineering

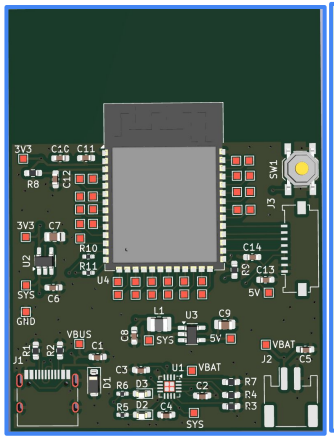
Context Solo hardware engineering co-op at Concentray, a startup building a desk-mounted focus device – think a physical commitment device that sits on your desk, forces you to block out time, and holds you accountable to it. I owned the electronics end-to-end, from blank schematic to assembled validated board, with no senior hardware engineer above me. As someone who struggles with focus myself, I genuinely believed in what we were building.

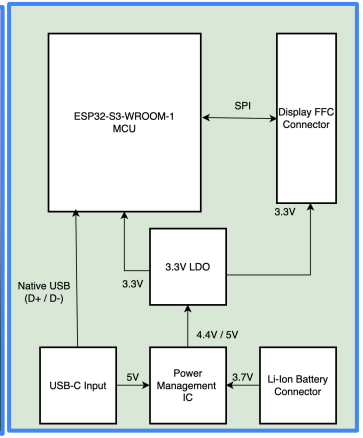
V1 → V2 The V1 PCB got the concept working but had an over-complicated power architecture, higher BOM cost, and routing issues. I noted down all of the issues I had and delivered V2 in the same month: a ground-up redesign with real constraints: 4-layer ESP32-S3 board, simplified USB-C + Li-Ion power path, 3.3V LDO regulation, SPI display interface, and transient protection. I was able to reduce the board size by 20% and cut ~$50 from the Bill of Materials.

Bring-up I designed with the intent of a smooth bring-up, including test points and hand-solderable components. I assembled using SMT and stencil soldering with a LPKF reflow oven, and debugged 5+ boards myself with the help of common EE tools such as a DMM and oscilloscope – checking power rails, signal integrity on the SPI lines, and USB enumeration.

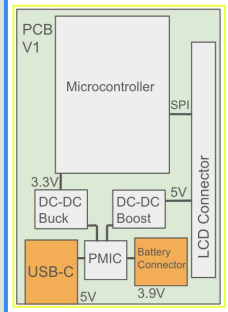
KiCad • Oscilloscope • Multimeter • Bench Power Supply • LPKF Reflow Oven • SMT/Stencil Soldering

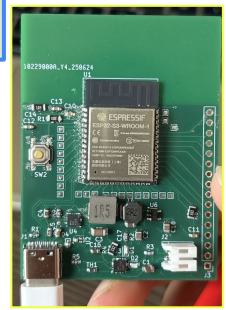
V2





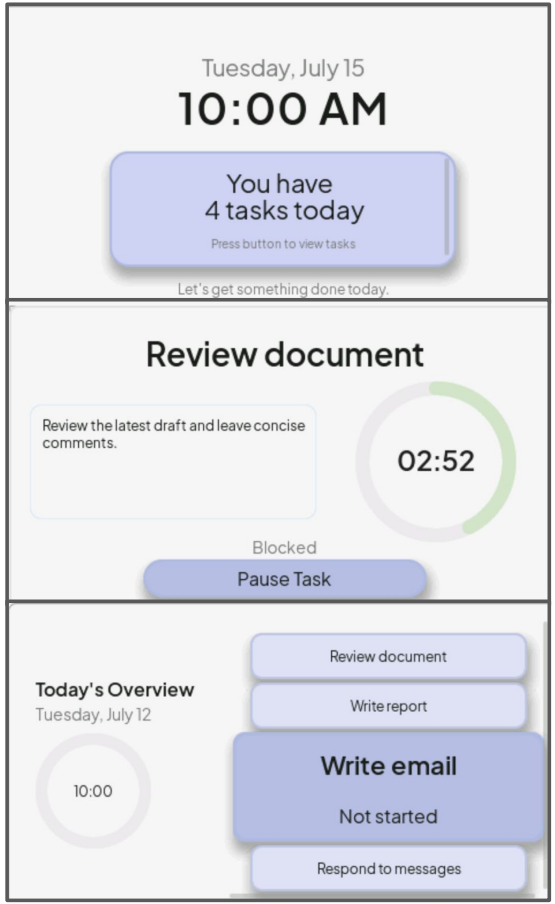
(V1 Reference)





CONCENTRAY

Concentray HMI Interface - Solo Electronics Engineering



Context The PCB is just the foundation — what the user actually sees and touches is the HMI. I wrote the entire UI firmware for the touchscreen: the thing that greets you in the morning, shows your tasks for the day, and counts down your focused work blocks. As someone who actually uses focus timers daily, I had strong opinions about how it should feel.

The simulator problem Embedded UI development is painful without tooling - you can't just hit run and see your interface, you'd have to flash to hardware every single iteration. So I built a hardware-independent simulator using SDL on desktop, letting me develop and iterate on the full UI on my laptop before the physical board even existed. Cut iteration time dramatically.

How it works Built on LVGL, a graphics library designed for embedded displays. Navigation runs on a finite state machine – each screen is a state, button pressed trigger transitions. The countdown timer, task list, and status widgets all render dynamically from live state rather than hardcoded layouts, so the UI always reflects exactly what's happening on the device in real time.

C++ • LVGL Graphics Library • Visual Studio Code • SDL • PlatformIO

RepSense - Hardware (Solo project)

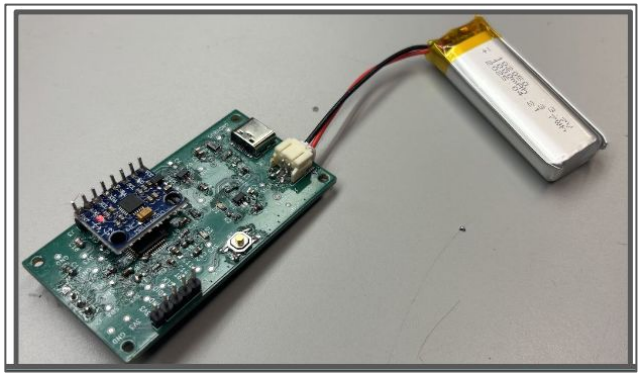
Context I've been weightlifting for 4 years and powerlifting for 1. My biggest training annoyance is losing count mid-set. My Apple Watch has an app that auto-detects exercises, but I wanted to build my own version from scratch: custom hardware, custom firmware, no app dependency, and overall fixing a problem from my favourite hobby: lifting weights!

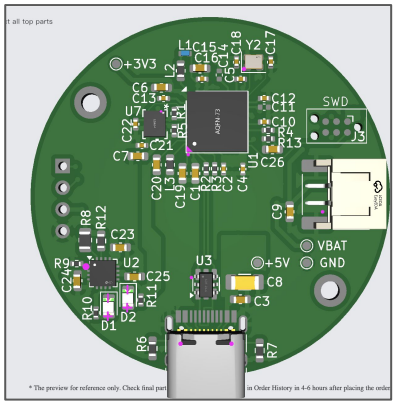
V1 — Does the sensing work? Prototyped on a dev board I made first in August 2025 with an STM32. Raw motion data is noisy - a bicep curl looks similar to other movements unless you mathematically separate orientation from acceleration. I implemented a sensor fusion algorithm that cleans the signal; reps appear as clear, countable waves. Validated on real gym data before designing any hardware. V1 had no size or power constraints – purely proving the concept.

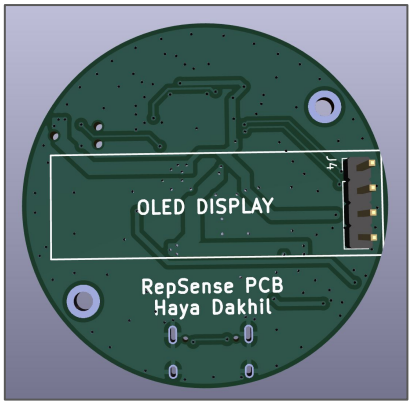
V2 — Making it wearable Redesigned around real constraints: 42mm circular PCB (size constraint to be wearable), battery-powered (power constraint), with M2-screw-size mounting holes for an enclosure. It contains a small rectangular OLED display to showcase the real-time rep count data.

I switched to a Nordic NRF52840 microcontroller because it has a built-in power converter and ultra-low sleep mode which is exactly what I need for a low-power and size constrained PCB. The device only wakes when it detects motion, so between sets it's effectively off. I calculated ~18 full gym sessions per charge on a 40mm-sized battery. All component values were calculated from first principles, and all to adhere with my design goals of this project.

V1 PCB (Dev board)





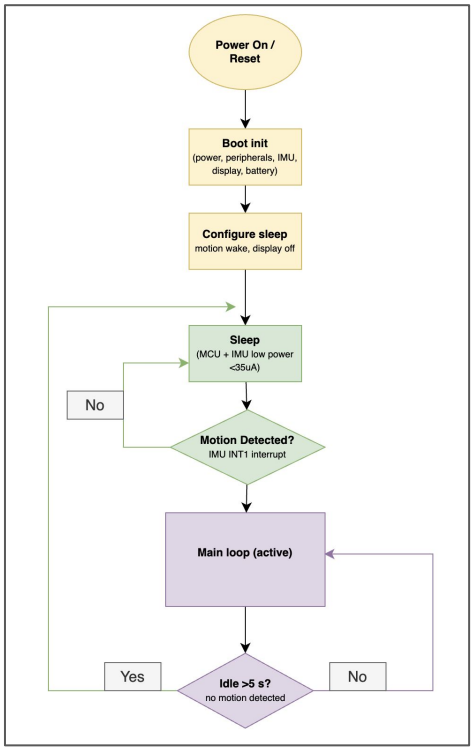


V2 PCB (Real wearable)

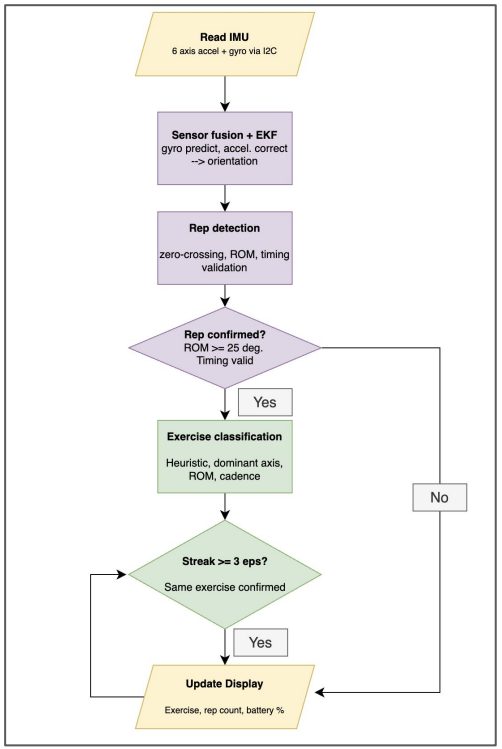
KiCad • JLC PCB • Stencil Soldering

RepSense - Firmware (Solo project)

State machine



Main loop



C++ • PlatformIO • Arduino framework

The firmware problem Having hardware that knows you're moving isn't enough — it needs to know what you're doing and count it correctly. A bicep curl and a lateral raise both involve rhythmic wrist motion; the difference is in the axis of rotation, the range of motion, and the timing which all relies on robust firmware.

How it works Raw IMU data feeds into an Extended Kalman Filter which produces a clean orientation estimate. From there, a rep detection algorithm watches for zero-crossings in the motion signal with a minimum range of motion threshold (25°) to filter out incidental movement. Once a rep is confirmed, an exercise classifier reads the dominant axis and motion signature to determine what exercise you're doing. The display updates in real time.

Why the sleep architecture matters The device spends most of its life doing nothing. The state machine keeps the system in a low-power sleep (~35μA total) and only wakes when the IMU detects motion above a threshold. No polling, no timers - the sensor itself triggers the wake. This is what makes the battery life practical.

V2 Roadmap The current classifier is heuristic – hand-tuned rules per exercise. V2 replaces this with a TinyML model trained on real rep data from my gym sessions for better accuracy across different users and movement styles.

Audio Spectrum Visualiser - Hardware (Solo project)

Context I'm a music lover – mostly rock and indie. There's something about putting on Arctic Monkeys and actually *seeing* the bass hit in real time that I'd always wanted to build. Fully standalone: no PC, no phone, no app. Three physical buttons control display mode, brightness, and color theme.

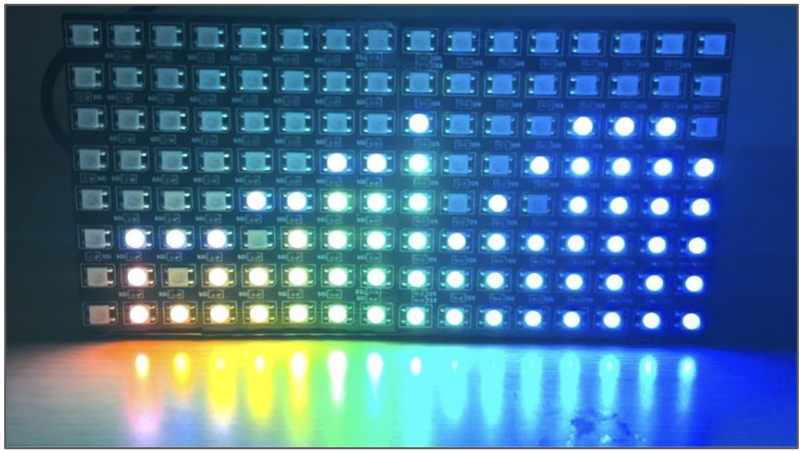
Software first Before designing any PCB, I validated the entire pipeline on a Raspberry Pi with a USB mic and an off-the-shelf LED matrix. A microphone outputs a raw pressure wave — it has no concept of bass or treble. The FFT (Fast fourier transform) is a mathematical transform to decompose that signal into its individual frequency components, telling you exactly how much energy exists at 60Hz, 1kHz, 8kHz and everywhere in between. I proved this pipeline in a day — FFT, frequency band mapping, and LED rendering – worked end-to-end before committing anything to silicon.

Key hardware decisions I designed two custom PCBs with the constraint of everything being hand-solderable and cost-effective. The ESP32-S3 module was chosen for its dual-core 240MHz processor which is essential for FFT math and easier hand-solderablility. A buck converter replaced a simpler voltage regulator after thermal analysis showed the regulator would dissipate 595mW at peak load, too much heat for a board with no heatsinking. The microphone supply gets its own RC filter, calculated to provide −50dB attenuation at the converter's 500kHz switching frequency – isolating the sensitive audio circuitry from power supply noise. The LED matrix is 4-layer so a dedicated ground plane handles the fast switching currents from 128 LEDs without EMI issues a 2-layer board would have.

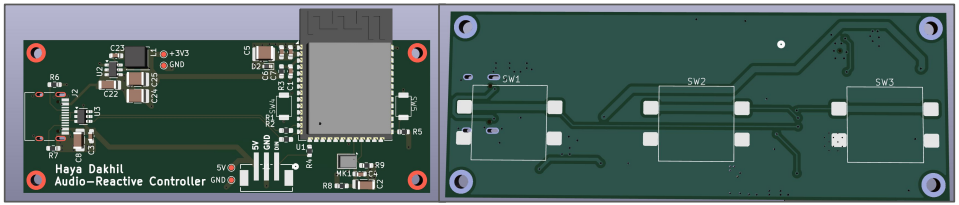
Results Fully working prototype. The linked video shows its ability to map audio!

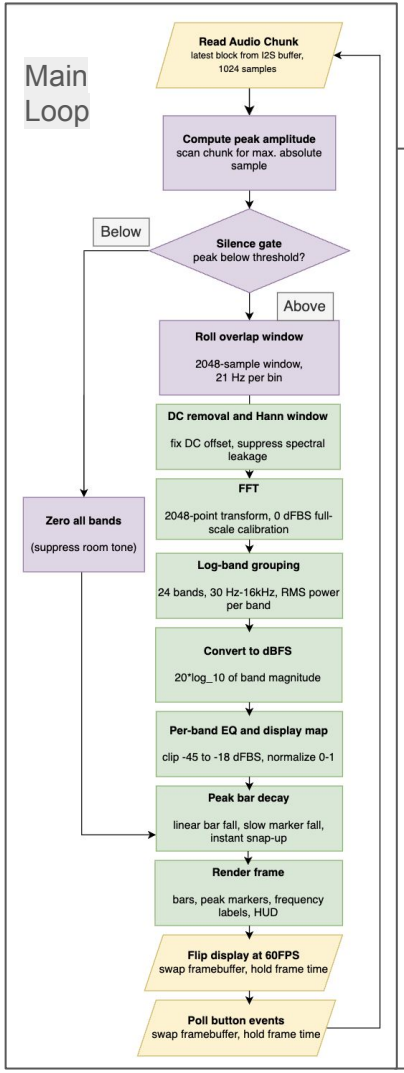
Audio Engineering • KiCad • Raspberry Pi • JLC PCB • PCBA

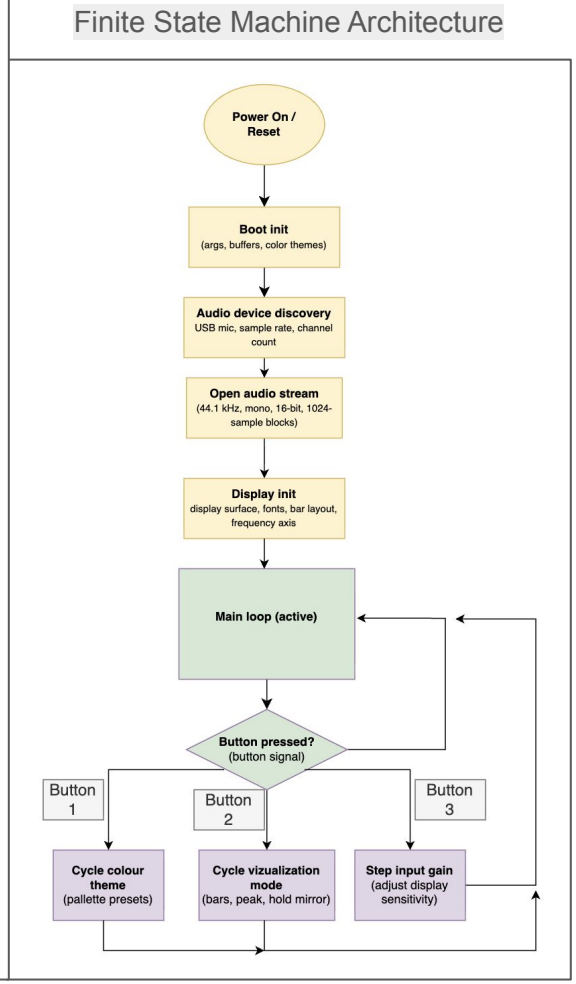
[Demo Video Here]



Main controller PCB







Audio Visualiser: Firmware (Solo project)

What is FFT and why does it matter? A microphone just sees pressure changing over time – no concept of bass or treble. The FFT mathematically decomposes that raw signal into individual frequency components, telling you exactly how much energy exists at every frequency. It's the core tool behind every spectrum analyzer and equalizer ever built.

The signal chain Mic captures raw audio → silence gate skips processing when nothing is playing → FFT converts time-domain signal into frequency data → 24 log-spaced bands map that into what your ear actually perceives → each band converts to dBFS so the display behaves consistently regardless of volume → peak-decay animation smooths the output so it's readable without losing responsiveness.

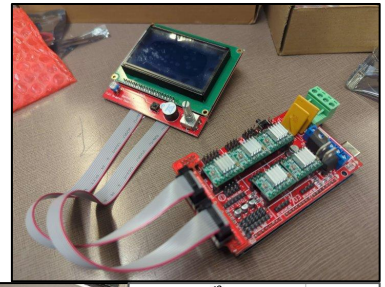
Why two cores? Audio capture and the FFT/render pipeline run on separate processor cores. If they shared one, the tight timing requirements of driving the LED matrix could cause the audio buffer to drop frames. Splitting them via FreeRTOS keeps both hard real-time.

The flickering problem Early versions flickered badly — raw FFT output changes faster than the eye can track. Fixed with a peak-hold animation: bars snap up instantly but fall gradually, with a separate slower-decaying marker above each bar. Decay rates tuned empirically on the Raspberry Pi first, then carried over directly to the ESP32 firmware.

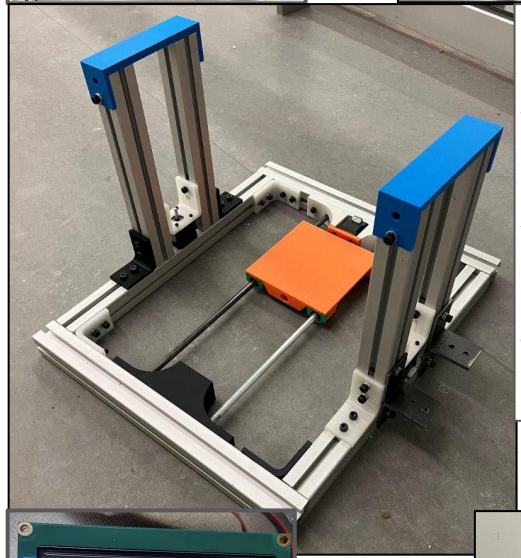
C++ • PlatformIO • ESP-DSP • FastLED • FreeRTOS • I2S

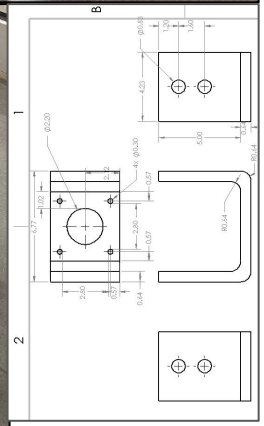
UW NanoRobotics Group - Hardware Lead

UNIVERSITY OF WATERLOO
NANOROBOTICS GROUP



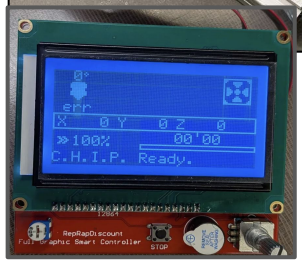
Context UW NanoRobotics Group is a student design team building CHIP — a conductive hydrodynamic ink printer that deposits silver nanoparticle ink onto flexible substrates to print functional electronics. The goal is making printed electronics research accessible to labs and startups who can't afford industrial fabrication equipment. I was a core member for 8 months then hardware lead for another 8.

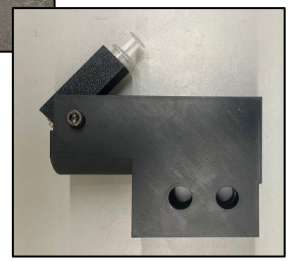




My role – Core member Hands-on building. I machined 5 custom aluminum U-bracket mounts in the UW machine shop, proposed and sketched the architecture for the ink deposition mechanism, assembled and soldered the electronics onto the controller board, and validated the stepper motors that drive the 3-axis motion system.

My role – Hardware lead Shifted toward system-level ownership. I ran weekly team meetings, guided first-years through SolidWorks, consulted professors when the team hit chemistry roadblocks, and helped acquire funding. Defined team priorities and kept the mechanical and electronics workstreams moving in parallel.





The machine CHIP uses a RAMPS 1.4 controller with an ATmega-based board driving 5 stepper motors - three for XYZ positioning and additional axes for the syringe-based ink dispensing mechanism. 1/16 microstepping gives 3200 steps per rotation, translating to ~0.66µm of theoretical linear movement per step — the precision needed for fine trace deposition.

Results Printer assembled and in final calibration. Presented at the 2026 ADPECS conference in Montréal on printed electronics and additive manufacturing.

Materials Science • Robotics • Additive Electronics

TENG Class Competition - 1st Place!

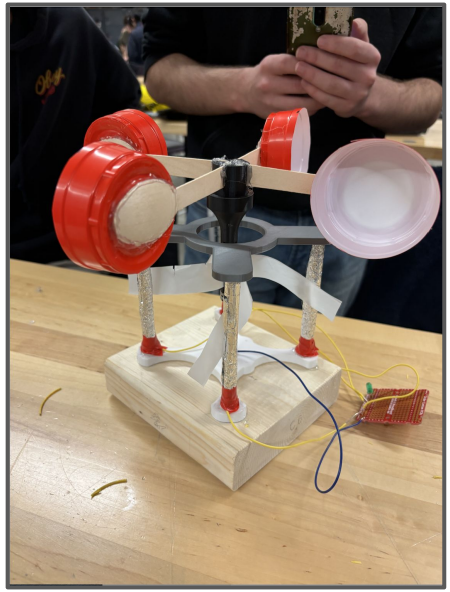
Context First-year nanotechnology course competition - placed in randomized groups of 4 and tasked with building a wind-powered device that converts mechanical energy into electricity using the triboelectric effect, a materials science concept we have learned in our courses. Two sessions total: one to plan and source materials, one 3-hour prototyping window.

My Role I proposed using aluminum and teflon as our specific material pairing because they sit at opposite ends of the triboelectric series, meaning they generate the highest charge transfer when rubbed together while being light enough for a spin-driven design. I handled the Solidworks CAD of a supporting structure, and during prototyping I managed the cable harnesses and soldered them to a perfboard containing a green LED which would light up if sufficient voltage were provided.

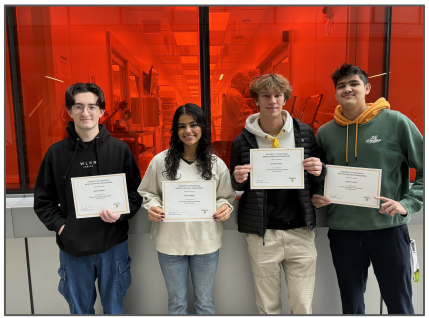
The Build Constraints were tight – limited materials with no budget for anything fancy. We raided our dining hall for wooden spoons as structural supports and used a ball bearing from our teammate's fidget spinner for the rotating mechanism. Judges tested output live with an air compressor and oscilloscope.

Results As shown in the linked video, our test received 6.6V and a 45Hz output – the highest in the class. We were awarded 1st place out of 28 teams, earning a perfect score across collaboration, design, efficiency and optimization. The project got selected for showcase at UW Open House, and I've been happy to present it at every open house since.

Our prototype



Our award



[Video Link Here]

HowToElectronics: Technical Content Contributor

Context I've been into hobby electronics for a while and worked with the person running HowToElectronics from my most recent internship – a 94k subscriber YouTube channel covering ESP32, IoT, and PCB design. I offered to help contribute and ended up writing and recording technical voiceovers for their tutorial videos.

My Role Taking complex embedded systems topics and explaining them clearly to a general audience — writing scripts that are technically accurate but accessible to someone just getting into electronics. Topics covered include ESP32, IMUs, PCB design, battery management, and motion sensing.

Why it matters Being able to communicate engineering concepts to a non-expert audience is a skill I genuinely enjoy. It forces you to actually understand something deeply enough to explain it simply. The last two videos I contributed to are at 20k+ views — and if even a fraction of those people caught the same bug I did for building things from scratch, that's enough for me for me to feel accomplished.

[Channel Link Here]