

Engineering Portfolio

Caden Polk

Electrical/Mechanical Engineering

polkcade27@sehs.net github.com/notrelaxedd Rocky River, Ohio

DESIGN PHILOSOPHY I believe that the exponential growth of technology combined with the slow growth toward the understanding of it amongst the general population creates many imbalances, causing many consumers to overpay or overvalue products/services that can be done cheaper, better, or more efficiently. Many of my projects aim to “level the playing field” by providing alternatives to the current solutions that hold a monopoly over the market.

AT A GLANCE

Focus	Electrical/Mechanical Engineering
Building since	2024
Fabrication	3D printing: 8 years CAD: 3 years Soldering: 3 years PCB Design: 1 year
Firmware / languages	Arduino, C/C++
Software	Onshape for 3D Design Arduino IDE/Claude Code for code Next.js and Tailwind CSS for software design

Project index

NO.	PROJECT	WHAT IT DOES	YEARS
01	HapBe	Assistive Haptic Device for Deaf People	2024-present
02	RowTech	First-of-its-kind force measuring system for crew athletes	2026-present
03	BikeComputer	Workout tracking device for personal use	2026
04	RC Jetboat	Jet-drive RC boat with a phone-controlled electronics system	2024-present
05	Automated Trading Platform	Rack of 10–15 recycled computers running automated trading as one cluster	2026-present
06	Lecturly	AI teaching assistant for K–12 teachers and homeschool parents	2025

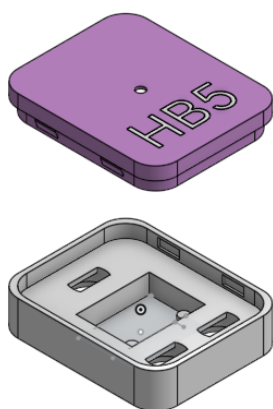
01

HapBe (Haptic Belt)

hapbe.net

ASSISTIVE TECHNOLOGY 2024-PRESENT INVENTOR AND LEAD ENGINEER

An alternative solution to give individuals with complete hearing loss a sense of direction through haptic (vibration) feedback.



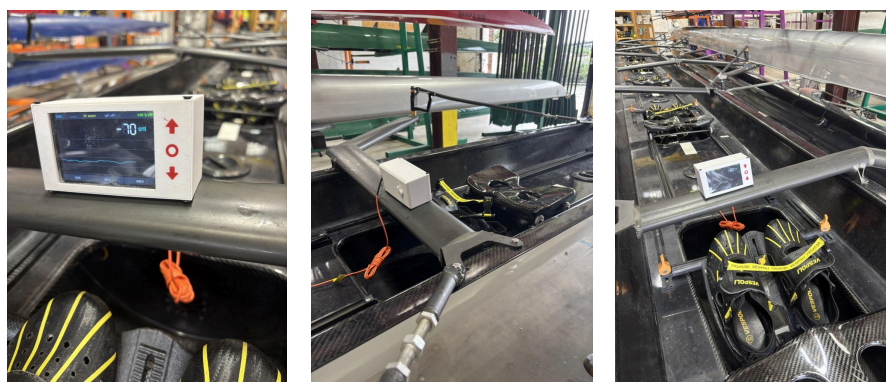
SPECIFICATIONS

Role	Inventor and Lead Engineer
Sensing / input	5 MEMS microphones located around the user's waist
Compute	ESP32-S3
Interfaces	I2S
Output / display	No display, only a small battery indicator located on the top of the device facing the user. Web app accessible by logging into the device via the user's phone
Power	2000mah Li-ion battery. Charges via a magnetic connector similar to older Macbook generations.
Fabrication	~200 prototypes printed at home with my Bambu Lab P1S
Software	Arduino
Constraints	The main constraint of the project was the cost to research, create, and test prototypes. While I did secure a micro-grant from John Carroll University, it was not enough to develop the project a substantial amount.
Status	Prototype 7 is currently paused while I wait for more capital to come in. I put away a portion of my lunch money and gas money every week to afford the parts to build and test prototypes.

THE PROBLEM	I came up with the idea in 2024 after an interaction with a deaf person inside of a Walmart near my house and realized that current solutions to total hearing loss – such as Cochlear implants – were extremely expensive, laborious, and not applicable to all individuals.
HOW IT WORKS	5 MEMS microphones located around the user's waist listen to the noise levels around the user at all times. When a microphone reads a sudden rise in decibel level – such as a call for help, a baby crying, or another situation where the user needs to understand where the sound is coming from – a vibration motor positioned behind the microphone alert the user to the direction of the sound, allowing them to turn and respond quickly.
DESIGN DECISIONS	The biggest decisions I had to make were definitely the form factor and the type of microphone. I went through a couple different form factors, such as a necklace and a hat, but I ended up choosing a belt because of its versatility, size, and position on the user. The microphone choice, however, was significantly harder. I went through 3 different types of microphones before I ended up choosing digital MEMS microphones because of the digital output that was easy to read, their small size, and the durability of the module.
WHAT WENT WRONG	My biggest failure was definitely HapBe 3, where I spent about 200 dollars and ~20 hours on a prototype that would not turn on. That failure caused me to almost give up on the project because of how much I lost, but I kept pushing through with the goal of giving under-privileged individuals the solution they deserve.
WHAT I LEARNED	Being my first real electrical engineering project, I learned quickly that the skills required to make a custom product were not in my wheelhouse. Despite this, the project taught me how to research the market and competition, overcome financial struggles and doubt from others, and pivot when a design is not the best that it can be.
WHAT'S NEXT	The current plan for this product is to continue with it once I get into college. I have halted almost all work on it to focus on school, crew, and other less intensive projects that are easier to balance with my schedule.

DATA ANALYTICS FOR CREW TEAMS/ATHLETES 2026-PRESENT INVENTOR AND LEAD ENGINEER

A first of its kind force-measuring system for under-privileged crew teams that gives data currently accessible to only the wealthiest of teams.



SPECIFICATIONS

Role	Inventor and Lead Engineer
Sensing / input	A load cell sits in the rigger of the boat and measures the force from the oar
Compute	Adafruit Feather ESP32-S3
Interfaces	SPI (ESP32 to the ILI9488 display), I2C (MPU-6050 IMUs), the HX711's two-wire clock-and-data protocol, and ESP-NOW between the seat nodes and the hub.
Output / display	3.5inch TFT display; users can also wirelessly access their data and upload it to the cloud
Power	3000mah rechargeable Li-ion battery
Fabrication	~10 prototypes 3D printed on my Bambu Lab P1S
Software	Arduino/C
Constraints	Budget and testing
Status	Prototype is fully built, waiting on a day to test on the water

THE PROBLEM

Current data analysis systems for crew teams cost upwards of \$11,000 USD for one eight-man boat. My team, Saint Edward High School, has very limited resources and cannot afford the systems. Frustrated with the cost of such systems, I researched the companies and found their patents and designs. What I found shocked me: an eight man boat costs at most \$3,000 USD for the companies to build, meaning the companies have a profit margin of ~72%. This caused me to begin building my own solution that is a tenth of the price of current solutions.

HOW IT WORKS

A load cell inside the rigger measures the force coming from the oar. This reading feeds into an amplifier which gives the microcontroller a readable output. This force reading can be used to measure:

Force	Drive/Recovery ratio
Force Graphs	Catch/Finish timing
Difference in force across all rowers	Acceleration through the drive
Drive time	Impulse in the drive
Recovery time	

All of these data points can be used to significantly help a crew increase their efficiency on the water, which "levels the playing field" against other crews.

DESIGN DECISIONS

The biggest design decision was the location of the force sensor. Originally, the idea was to put the force sensor on the pin that connects the oarlock to the boat. I quickly found that a force sensor inside a bolt is the most expensive design I could have come up with. After lots of force diagrams, I settled on putting the force sensor inside of the adjustable bolt located on the backstay of the rigger (see photo above). This proved to work significantly better and cut costs down to just over one hundred dollars a seat.

WHAT WENT WRONG

The biggest failure I faced was in the design of the load cell. I originally wanted to put the load cell inside of the oarlock, which would give the most direct force reading. When I was researching competitors, however, I found a patent for that exact design. Knowing the complications of violating patent law, I had to pivot.

WHAT I LEARNED

The biggest thing I learned was how to create custom PCBs in CAD. This was crucial for my future projects, as it allows me to create exactly what I need for each individual project instead of relying on pre-made parts.

WHAT'S NEXT

I currently have 4 pre-orders for the equipment, with much more to come. We are currently developing the product into its version one form, and will start the seed funding round shortly after.

03 BikeComputer

ATHLETIC TECHNOLOGY 2026 CREATOR

Created a low-cost bike computer myself instead of paying ~300+ dollars for an off-the-shelf model



SPECIFICATIONS

Role	Planned, designed, and created the product.
Sensing / input	GPS
Compute	Raspberry Pi Zero 2W
Interfaces	SPI
Output / display	3.5in TFT Color Display
Power	Portable battery attached to the back of the bike
Fabrication	3D modeled in Onshape, coded in Linux
Software	Raspberry Pi OS Lite
Constraints	Money, time, and resources
Status	Project was completed and is now a permanent addition to my bike

THE PROBLEM

I like to bike, but I don't have the money to afford a bike computer. I wanted to track my speed, distance, heart rate, and most importantly, be able to leave my phone back at my house and still have music. This is only possible with an expensive bike computer like a Garmin, so one weekend, I set out to build my own

HOW IT WORKS

The onboard GPS connects to ~6-8 satellites overhead. Once a connection is established, the satellites transmit information such as location and speed. The onboard computer reads this data and creates a bitmap, which is a digital map of the surrounding area. It then places the user on the bitmap according to the satellites information and and orient itself according to the direction in which the user is traveling. This process happens approximately ten times per second, giving the user instant feedback on their bike ride. All of this data is shown to the user on the 3.5in TFT display.

WHAT'S NEXT

N/A: project is finished.

04 RC Jetboat

MARINE / EMBEDDED SYSTEMS 2026-PRESENT DESIGNER AND BUILDER

A radio-controlled jet-drive boat whose lights, pumps, and fans are all controlled and monitored from a phone.



SPECIFICATIONS

Role	Designing and building the boat and its onboard electronics
Sensing / input	GPS (speed), 2 temperature sensors (ESC and battery), battery voltage
Compute	Adafruit Feather ESP32-S2
Interfaces	Wi-Fi (web dashboard)
Output / display	8 RGB underwater LEDs, 6 fans and 2 water pumps (each individually controllable), 10 high-powered lights on two channels (8 + 2). Web dashboard at 192.168.4.1 for control and for GPS speed, battery voltage, and temperatures.
Power	Two separate batteries: one for the motor and ESC, one for the electronics
Software	Arduino/C++
Constraints	Keeping the cost of the electronics low
Status	In progress. Started September 2026.

THE PROBLEM

A normal RC boat tells the driver nothing once it leaves the shore. I wanted to know how fast the boat was going, how much battery was left, and how hot the ESC and battery were getting, and I wanted to control every pump, fan, and light on board individually. I also wanted to do all of it from my phone while keeping the cost of the electronics low.

HOW IT WORKS

A GPS module measures the boat's speed, and two temperature sensors watch the ESC and the battery. The Feather ESP32-S2 reads those sensors along with the battery voltage. It serves a web dashboard at 192.168.4.1, so a phone connected to the boat can see speed, voltage, and temperatures live. From the same page I can switch each of the 6 fans and 2 water pumps individually, set the 8 RGB underwater LEDs, and turn the two channels of high-powered lights on and off. The drive motor and ESC run from their own battery, separate from the electronics.

DESIGN DECISIONS

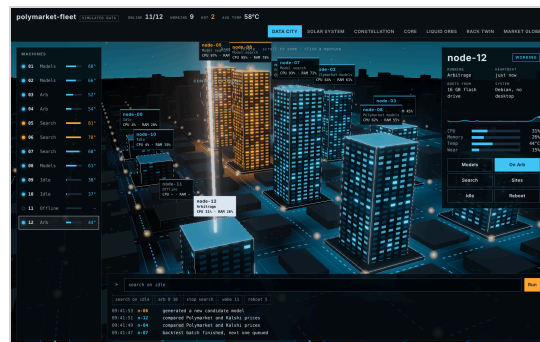
For speed measurement I went with GPS, which needs no sensor in the water. I run the electronics on a separate battery from the motor and ESC instead of sharing one pack. This keeps the microcontroller's supply steady when the motor draws heavy current, at the cost of carrying a second battery. I originally planned an onboard camera and dropped it, which fit the goal of keeping the electronics cheap.

05

Automated Trading Platform

DISTRIBUTED COMPUTING 2026-PRESENT BUILDER AND OPERATOR

A rack of 10–15 old office computers that I turned into one cluster for running automated trading models.



SPECIFICATIONS

Role	Set up the rack and the system that runs on it
Sensing / input	Live market data from the APIs of two event-contract exchanges
Compute	10–15 old workstation computers (DDR3 memory, older Intel processors)
Output / display	No monitors; every machine runs without a desktop
Software	Debian without a desktop on every machine. Some workers boot from 16 GB flash drives.
Constraints	Old hardware, and small storage on the flash-drive workers
Status	Running. It currently handles automated trading and cross-exchange arbitrage only. I want to build on it so other workloads can be deployed to the same machines.

THE PROBLEM

I had 10–15 old workstation computers with DDR3 memory and older Intel processors. None of them is powerful on its own, but together they add up to a useful amount of computing. I wanted to put them to work as one system instead of letting them sit.

HOW IT WORKS

Each computer runs Debian with no desktop, so all of its memory and processor go to the work. Some of the machines boot from a 16 GB flash drive. The machines pull live prices from two exchanges through their APIs, run trading models against them, and look for cases where the same contract is priced differently on the two exchanges.

DESIGN DECISIONS

I run Debian without a desktop on every machine. On hardware this old, a graphical desktop would use memory and processor time that the work needs, and the tradeoff is that everything has to be managed from the command line. Some workers run from 16 GB flash drives instead of internal drives, which gets a machine running with almost no cost but leaves very little storage on it.

WHAT'S NEXT

The system only runs trading right now. The next step is making it general enough that I can deploy other projects onto the rack.

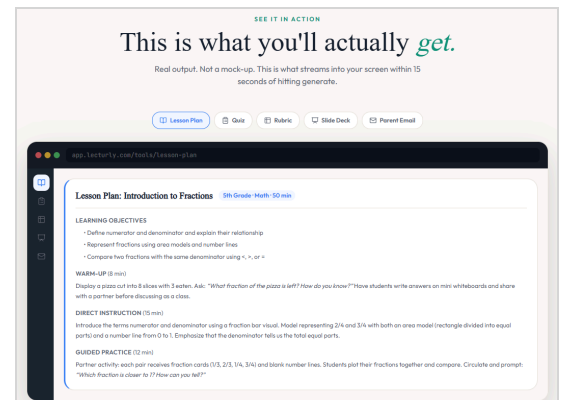
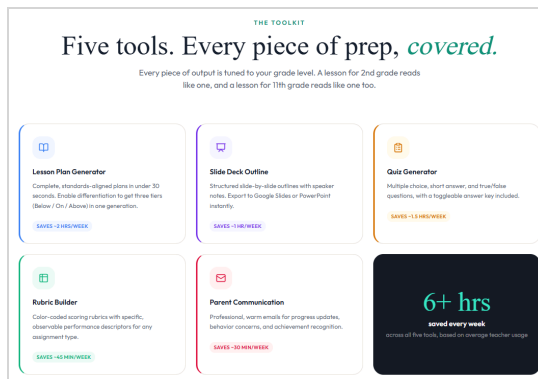
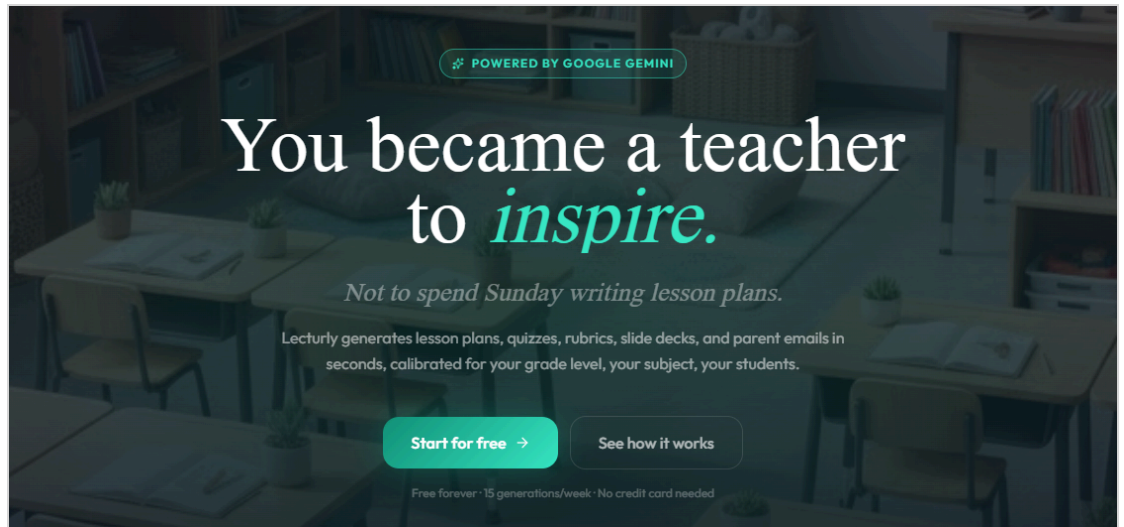
06

Lecturly

lecturly.app

EDUCATION SOFTWARE FOUNDER AND SOLO DEVELOPER

An AI teaching assistant for K–12 teachers and homeschool parents.



SPECIFICATIONS

Role	Founder and solo developer: product, code, pricing, and marketing
Sensing / input	A teacher's or parent's request in the web app
Compute	Hosted on Vercel; Google Gemini for generation
Interfaces	Supabase (database and accounts), Stripe (payments), MailerLite (email automations)
Output / display	Generated teaching material, shown in the web app
Software	Next.js 14, Supabase, Google Gemini, Tailwind CSS, shadcn/ui
Constraints	One developer, competing with established products like MagicSchool AI and Brisk Teaching
Status	Live at lecturly.app with a Free tier (15 generations per week), an unlimited Pro plan at \$9.99/month, and a quote-based Enterprise tier

THE PROBLEM	Preparing lessons takes teachers and homeschool parents hours outside of the time they spend teaching. Lecturly is my attempt to give both groups an affordable AI assistant for that work, with a free tier and a low-cost unlimited plan.
HOW IT WORKS	A teacher or homeschool parent signs in to the web app, which is built in Next.js and hosted on Vercel. Their request is sent to Google Gemini, which generates the material. Accounts are stored in Supabase, and paid plans run through Stripe. Email sequences in MailerLite welcome new signups, follow up with users who have been inactive for 14 days, and prompt users who are near the free limit to upgrade.
DESIGN DECISIONS	The biggest decision was pricing. I restructured the plans into a Free tier capped at 15 generations per week, an unlimited Pro plan at \$9.99/month, and a quote-based Enterprise tier that replaced the old Max tier. Before making the change, I benchmarked Lecturly against MagicSchool AI and Brisk Teaching.
WHAT'S NEXT	Feature work on school single sign-on, district pricing, an AI model selector, differentiation tools, and a homeschool mode. For outreach, I built a directory of K–8 school staff in Northeast Ohio.
