ECE HONORS · UT AUSTIN

ADITYA PULIPAKA

I thrive at the intersection of **hardware and software**, with experience in embedded systems, robotics, controls, and computer vision. I'm actively seeking interesting, multifaceted roles in these areas.

- Currently: Driver Dev Intern @ Qualcomm · Lev Subteam Lead @ Guadaloop

adipu@utexas.edu
aditya.pulipaka.com
linkedin.com/in/aditya-pulipaka

SELECTED ENGINEERING PORTFOLIO · JUNE 2026

ABOUT

I'm an **ECE Honors undergraduate at UT Austin** who likes problems that don't fit neatly into "hardware" or "software." I've written OS-level drivers at **Qualcomm**, shipped flight-control, maintenance, and testing software at **Bell Flight**, built a LiDAR SLAM localization stack with the **UT Dallas NOVA team**, and architected prototyping, testing, and control solutions at **Texas Guadaloop**.

What drives me is seeing my work come to life in realistic scenarios. I've done this through building **closed-loop control systems** on real hardware, bringing up **computer vision systems** that function in practical spaces, and working on multiple **sensor-fusion projects** that solve real problems. I care about systems that are fast, robust, and honest about their limits.

BASED IN	FOCUS	EDUCATION	LANGUAGES
Austin, Texas	Embedded · Robotics · Controls	BS ECE Honors, UT Austin '28	English · Telugu · Spanish

02 — EXPERIENCE

WHERE I'VE WORKED

Embedded software, robotics research, and controls — across industry, university labs, and student engineering teams.

MAY 2026 — PRESENT · SAN DIEGO, CA

Driver Development SWE Intern · Qualcomm

Windows on Snapdragon. Extending device drivers to handle multiple clients and building a testing framework for the sensor suite. Driver/OS-level work.

JAN 2026 — PRESENT · AUSTIN, TX

Computer Vision Researcher · UT Austin AMRL

Built ROS2 system using the Spinnaker SDK to pull frames over limited bandwidth from FLIR Blackfly S GigEVision cameras, correlate keypoints from multiple views, and output global pose estimates for people and obstacles in a space. Now working toward a markerless multi-camera calibration pipeline using humans as calibration targets.

DEC 2025 — PRESENT · AUSTIN, TX

EM Levitation Subteam Lead · Texas Guadaloop

Leading development of our control-algorithm and a repeatable formula for the HEMS levitation success.

JUN 2025 — AUG 2025 · ARLINGTON, TX

Embedded SWE Intern, 525 Flight Control Systems · Bell Flight

Resolved 5+ software problem reports across FCS code and low-level tests; built a Python tool that migrated ~1000 high-level tests and comms databases between environments, saving an estimated 6–12 months of manual work. Resolved 13 key issues with .NET windows app, gaining comfort with .NET C# code using WPF.

MAY 2025 — AUG 2025 · DALLAS, TX

Autonomous Driving SWE · UT Dallas NOVA

Built the LiDAR SLAM + map-based localization stack (Open3D / KISS-ICP), reaching sub-5 cm localization accuracy via combined ICP and feature matching.

SEP 2024 — MAY 2025 · AUSTIN, TX

Software Team · Stampede — UT Austin RoboMaster

Monocular localization pipeline for the Sentry robot using OpenCV SolvePnP on standardized markers; tested odometry methods for tighter positioning including dead-wheel and encoder.

2023 — EARLIER

Earlier: Articulate Labs · UT Southwestern (Takahashi Lab) · Motorheads Robotics

Product-dev intern on the KneeStim rehab device; circadian-rhythm biochemistry research; and co-captain of a state-championship-debut FTC robotics team.

> *"Aditya is a pleasure to work with and has excellent engineering skills... he developed full software-lifecycle solutions for our flight control computer software, exhibited agile thinking, and took initiative to suggest and implement additional improvements."*
>
> — Steve Feenstra, Flight Controls Software Lead / Staff Software Engineer, Bell Flight

03 — SELECTED WORK

THINGS I'VE BUILT

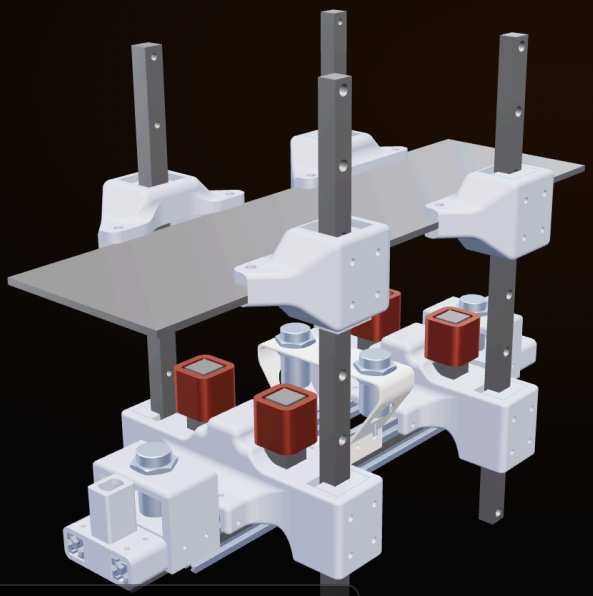
Research, internships, and hackathon wins. The four lead projects have live, scroll-driven 3D walkthroughs on the site; their key views are captured here.

01	**Guadaloop Levitation Rig**	Maglev test rig · sim env · subteam lead
02	**RescueVision**	mmWave + UWB through-wall localization
03	**SmartPT**	CV + IMU sensor-fusion recovery tracking
04	**BlindMaster**	Full-stack IoT smart blinds
05	**TweinStein**	Robotics · Embedded · 🏆 1st · ECE445M Final Race
06	**LiDAR SLAM & Localization**	Robotics · Autonomy · Localization
07	**Harmonium**	Software · Web · Computer Vision · ★ Most Risky @ TVG

01 MAGLEV TEST RIG · SIM ENV · SUBTEAM LEAD

GUADALOOP LEVITATION RIG

LevSim is the simulation environment I built so levitation controllers can be designed and tuned against the rig's *real* magnetic behaviour — not a hand-derived model that falls apart on contact with hardware.



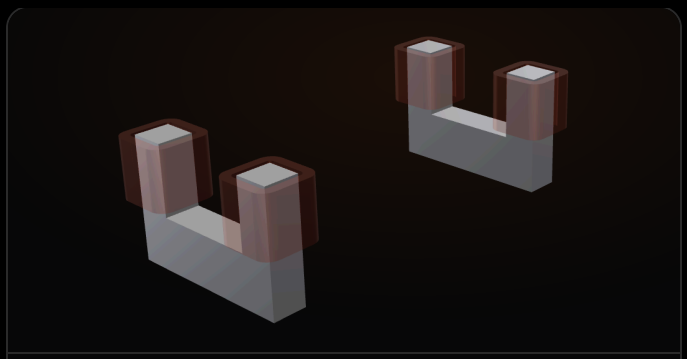
INTERACTIVE 3D WALKTHROUGH · LIVE ON THE SITE

Texas Guadaloop · Levitation

Ansys FEA | SciKit-Learn | PyBullet | Gymnasium | PID · LQR · RL

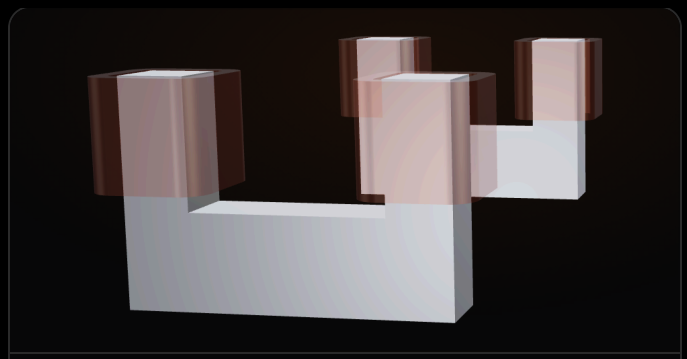
GitHub ↗

ROLE	TIMELINE	STACK	STATUS
CAD · All Software Architecture + Implementation · Ansys work guided by Aditya Matam & Neha Ramachandran	Nov – Dec 2025	Ansys Electronics Desktop · SciKit-Learn · PyBullet · Gymnasium	Functional prototype



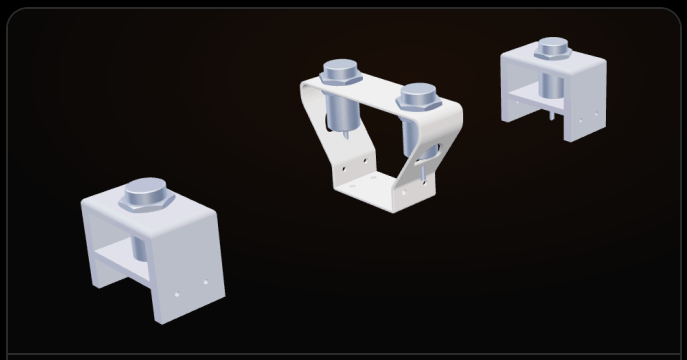
Isometric View

Two U-shaped 1020 carbon steel cores wrapped with the copper coils at either end to modulate attractive force.



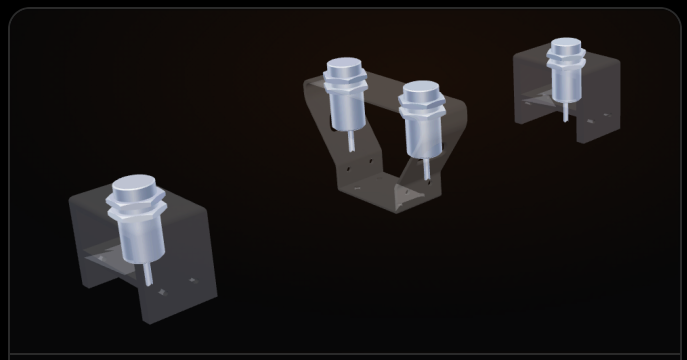
Front View

These yokes have the same width and height as the full-size yokes on our large pod. In fact, those are just 9 of these stacked together face-to-face!



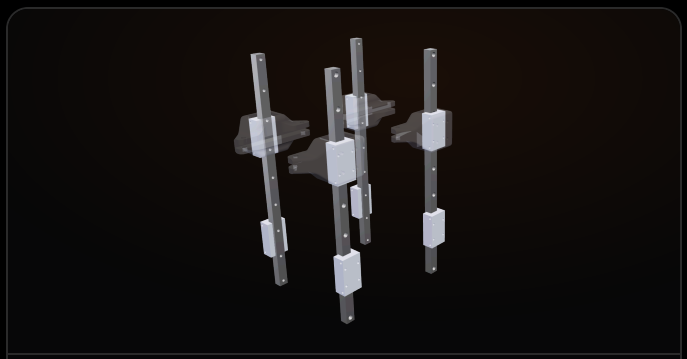
Isometric View

Our team opted for inductive sensors due to their price point and performance at close proximity. We're guaranteed to have a steel track, so why not make the most of it?



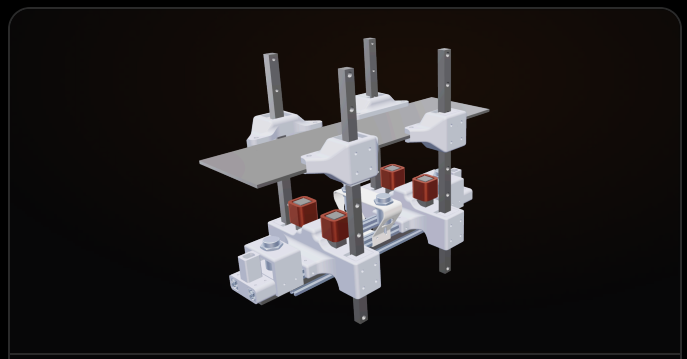
Isometric X-Ray

The sensors are held in place by robust plastic structures to encourage clean data.



Stage 1 · rails & carriages

A vertical rail system was used to stabilize along attitude axes for preliminary testing of z-only control.



Stage 3 · assembled

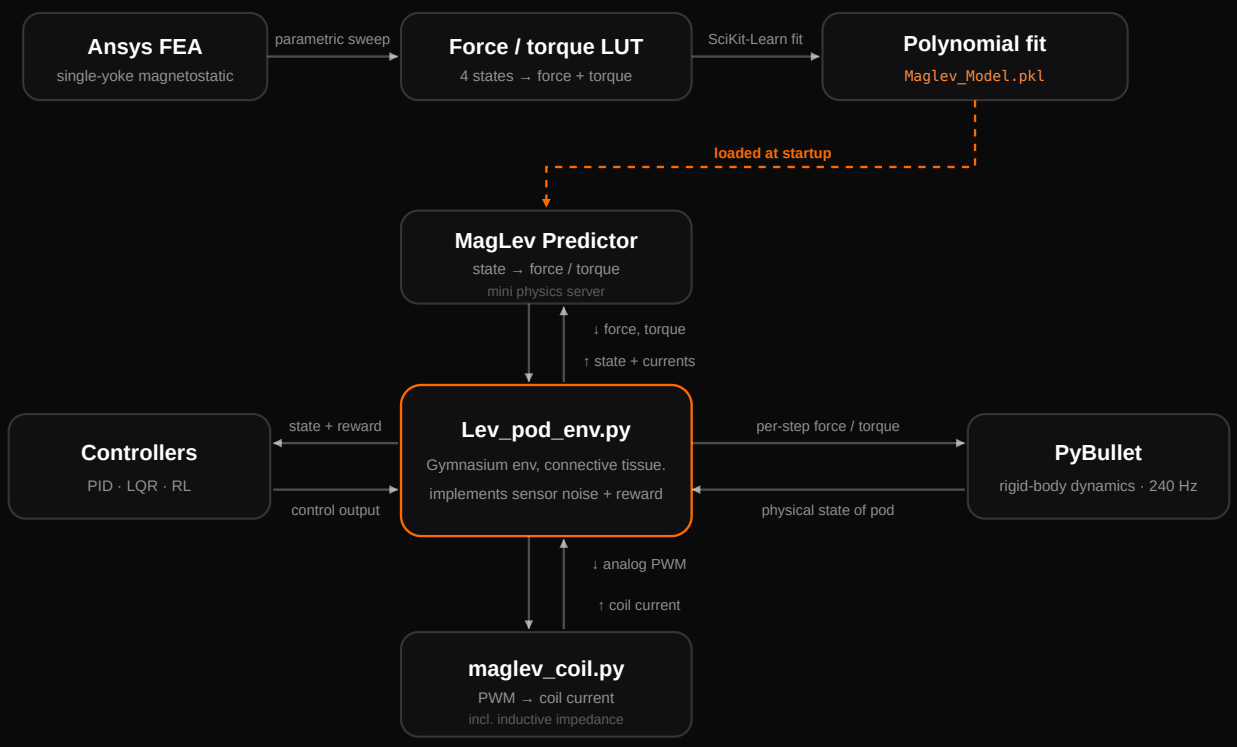
Together, the track, rails, and chassis allow us to rapidly prototype our control algorithms in a safe way.

— THE PROBLEM

Hybrid electromagnetic suspension (HEMS) is a proven low-power way to levitate, but it's brutally nonlinear — and for a student team there's no clean way to simulate one without re-deriving the magnetic flux through the whole structure (and the air around it) for every design change. At best that's tedious; at worst it's a wall that keeps teams without graduate-level magnetostatics from ever touching the control problem.

After two years on the lev sub-team I'd watched the same gap open again and again: controllers that held rock-steady in our MATLAB sim under any disturbance fell apart on the real rig under near-perfect conditions. We were missing a fundamental piece of the analytical model, and MATLAB wasn't scaling past PID anyway. The fix was to stop hand-calculating forces and root the simulation in empirical data instead — **Ansys FEA (verified against load-cell measurements) → a polynomial force model → a PyBullet rigid-body sim**.

SYSTEM OVERVIEW



Two-phase architecture: an Ansys FEA sweep of a single yoke is distilled offline into a polynomial force model, which a 240 Hz runtime loop (`Lev_pod_env.py` ⇄ PyBullet) queries every step while a controller drives the pod. Redrawn from the project's dataflow diagram.

The pipeline splits in two. **Offline**, an Ansys magnetostatic model of a single yoke is swept across the coil currents, gap heights and roll angles the rig actually sees; the resulting force/torque table is fit to a polynomial (`Maglev_Model.pkl`) so it can be evaluated in microseconds. **At run time**, `Lev_pod_env.py` is the orchestrator: each step it pulls the pod's state from PyBullet, converts the controller's analog output into real coil currents with `maglev_coil.py`, asks the MagLev Predictor for the per-yoke force and torque, and feeds those point loads back into PyBullet — all wrapped in a Gymnasium interface so any controller can drive it.

DECISIONS & DEEP DIVES

Why FEA, not an analytical model?

I first proved FEA was trustworthy: a load-cell setup measured the force on a single small yoke, and Ansys magnetostatic results at the same gaps agreed within ~10%. That made an FEA-derived lookup a reliable replacement for the broken analytical chain.

Why a polynomial on top of FEA?

Running Ansys inside the sim loop is infeasible. A parametric sweep across the rig's working range of heights, roll angles and coil currents gives a discrete table; fitting it in SciKit-Learn (`lev_sim/Function Fitting.ipynb`) turns each step into a cheap polynomial evaluation — and makes the pipeline plug-and-play for new yoke designs.

Why PyBullet + Gymnasium?

PyBullet handles the rigid-body dynamics; a custom `maglev_coil` component models coil current (including inductive impedance) so the force model gets *real* currents. Our LUT-fit polynomial model ensures we have FEA-backed force and torque driving our simulation, and Gymnasium standardizes the interface, so the same environment runs under PID (`lev_PID`) or RL (`lev_PPO`).

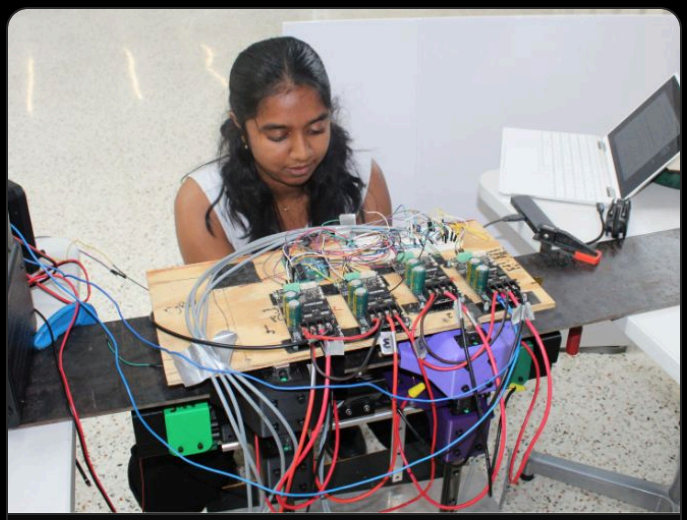
How do you reuse it across controllers?

`lev_pod_env.py` exposes exactly the I/O a controller sees on the real pod — distance-sensor readings in, motor-controller PWM out — so anything written against the sim can move straight onto the rig with no interface changes.

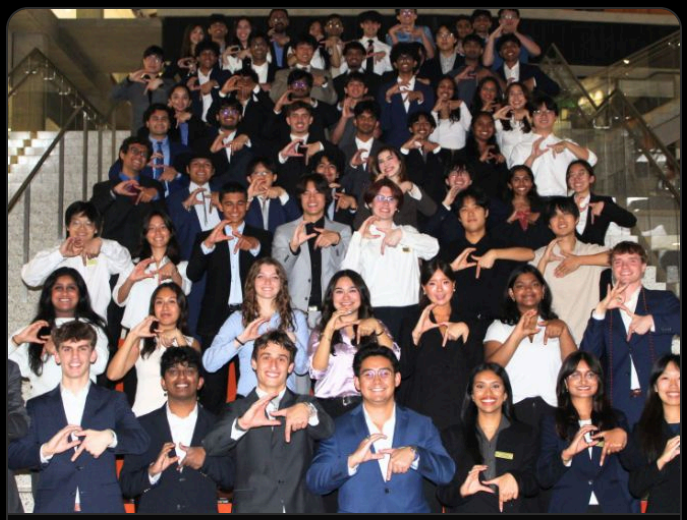
RESULTS & HONEST LIMITS

- A repeatable simulation process that works for complex systems like maglev, grounded in FEA data and tested with both a hand-tuned **PID** loop and a trained **PPO** policy.
- The force model is only as good as the swept range — extrapolate far outside it and the polynomial is unreliable.
- It's a functional prototype: the coil model is a simplification, and fully closing the sim-to-real gap still requires tuning runs on the physical rig, though our starting point improves considerably.

THE TEAM



Teammate Neha Ramachandran wiring up the levitation test rig.



The Texas Guadaloop team.

VIDEOS

▶	Full walkthrough · 8:22	youtu.be/McGaIZm4ikw ↗
▶	Working demo · 1:43	youtu.be/SVYZi0yS-i0 ↗
▶	Safe completion demo · 0:58	youtu.be/uVn-wENo2yE ↗
▶	Clean levitation run · 0:08	youtu.be/b1LHZhLbZhc ↗
▶	Early instability · 0:42	youtu.be/4QhSVZc7bHg ↗

02 MMWAVE + UWB THROUGH-WALL LOCALIZATION

RESCUEVISION

A wall-mounted hub that sweeps a 60 GHz radar to find people through drywall, smoke and fire — then drops each victim into the responder's own AR view, even with no line of sight.

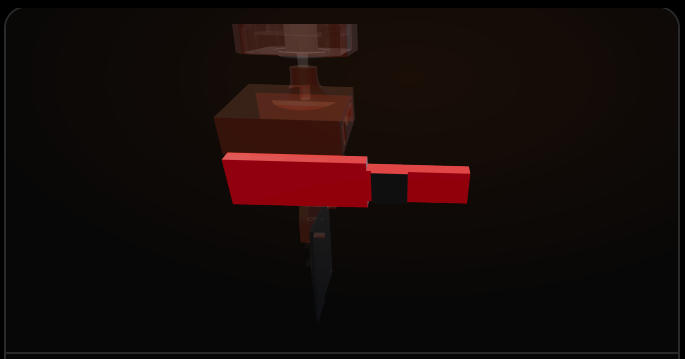
INTERACTIVE 3D WALKTHROUGH · LIVE ON THE SITE

1st Place · Analog Devices Sensor-Fusion · StarkHacks @ Purdue

TI IWR6843 mmWave | ESP32 + ESP-NN | Qorvo UWB | iOS · ARKit VIO | BLE

GitHub ↗ **DevPost ↗** **LinkedIn Post ↗**

ROLE	TEAM	WHEN	STATUS
CAD · Embedded firmware · USB Host · frame-of-reference math · hub mechanics	4 · Shoban Ananth, Su Park, Betool Mohsen	Apr–May 2026 · hackathon build + post-finals polish	Shipped — functional prototype



The eye

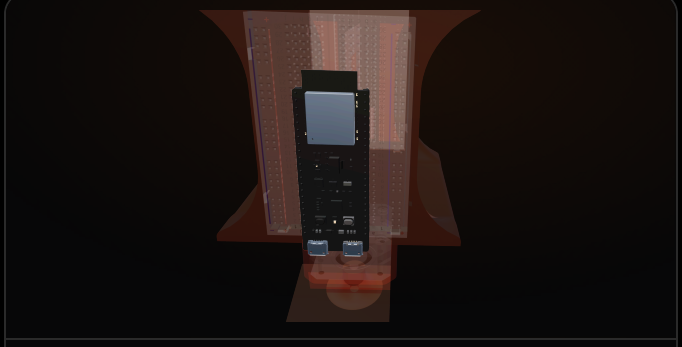
TI IWR6843AOPEVM — a 60 GHz mmWave radar with an on-board DSP. Running TI's Vital-Signs & People-Tracking binary, it finds people by their micro-motion and vitals — through drywall and smoke.

120° FOV

A 120° field of view per look. The DSP reports bounding boxes for each detected person straight off the chip — no host-side radar processing.

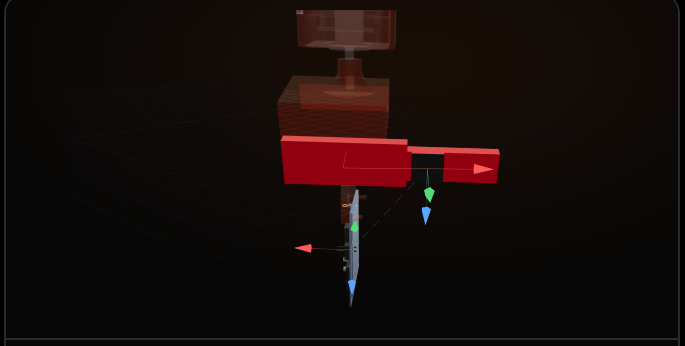
Sweep

A NEMA-17 stepper sweeps the radar in 90° steps, 0 → 360 → 0. With a 120° FOV that overlap gives full 360° coverage of the room between looks.



Brain

An ESP32-S3 is the hub brain: USB-host configures the radar, ESP-NN classifies each track as active / unconscious / phantom, and it runs the sweep + compass calibration.



Radar → UWB · second view

A −10° tilt about X separates the two frames. Detections are rotated and offset (−43.18, +13.97, +92.43 mm) into the UWB frame before BLE.

UWB × VIO

The hub streams victim positions over BLE. The phone fuses UWB ranging with ARKit visual-inertial odometry (Gauss-Jordan) to place the hub — and the people it sees — in the responder's frame, even with no line of sight.

THE PROBLEM

First responders entering an unfamiliar building have no good way to know where the people inside are — especially if those people are stationary or unconscious and can't call out. A persistently-installed sensor hub plus a responder phone app can give that information cheaply, even through walls.

HOW IT WORKS

Sensing

A TI IWR6843AOPEVM runs TI's Vital-Signs & People-Tracking binary: a 120° field-of-view, 60 GHz radar scan whose on-board DSP identifies people by their detected vitals and movement — work that survives drywall and smoke.

Hub control loop

- An ESP32-S3 uses USB-host to configure and initialize the radar, then ingests its bounding-box output.
- Compass calibration: the installer sweeps the hub through 360°; the stepper is locked at 0° until that's done.
- The stepper then sweeps in 90° steps (0→360→0). With a 120° FOV that overlap covers the full room.
- ESP-NN classifies each track as active, unconscious/stationary, or phantom; if 2 s pass with no real detection, it steps again.

Frame-of-reference math

Detections start in the radar's frame. Using SolidWorks-derived offsets plus the live stepper angle, each position is transformed into the Qorvo DWM3001CDK (UWB) frame, bundled with the compass heading, and streamed over BLE to the responder app.

Responder localization

The app pairs UWB ranging with ARKit's visual-inertial odometry and solves (Gauss-Jordan) for the hub's pose relative to the phone — then re-expresses every detection in the responder's own frame and renders an on-screen indicator, even far away with an occluded view.

RESULTS & HONEST LIMITS

Through-wall victim localization, rendered live in the responder's AR frame — enough to take 1st place in Analog Devices' Sensor-Fusion track at StarkHacks. It's a prototype: ranging accuracy degrades with multipath, the sweep cadence trades latency for coverage, and a v2 would add multiple hubs for triangulation.

VIDEOS

- Hackathon demo · 1:39 — youtu.be/SlMJJwz8QeA ↗
- Short cut · 0:56 — youtu.be/WuoMCzevC1U ↗
- IWR sensor bring-up · 1:02 — youtu.be/JVRwZt_0suA ↗

03 CV + IMU SENSOR-FUSION RECOVERY TRACKING

SMARTPT

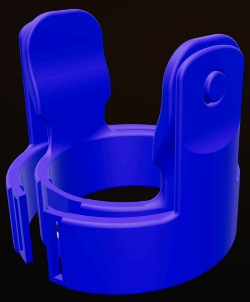
A physical-therapy app that turns a phone — plus an optional IMU brace — into a closed-loop joint-angle tracker, giving clinicians richer movement data than a self-reported scale.

1st Place · Healthcare Track · Nexhacks @ CMU

Flutter | ESP32 IMU · BLE 100 Hz | RTMPose3D | Extended Kalman Filter | Overshoot API

GitHub ↗ **DevPost ↗** **LinkedIn Post ↗**

ROLE	TEAM	WHEN	STATUS
Sensor fusion · ESP32 firmware · BLE	3 · with Mateo Taylor & Dustin Nguyen	Jan 2026 → ongoing	Live on TestFlight (Feb 2026)

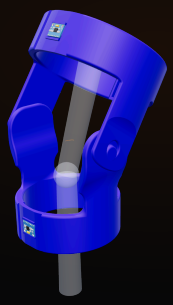


Brace + 2 IMUs

A hinged knee cuff. Two shells clamp the leg; each carries an MPU-6050 IMU wired to a XIAO ESP32-C6 that streams orientation over BLE in packets of 3 whenever 3 measurements are obtained at 100 Hz, effectively making transmission rate ~33Hz.

One IMU per shell

The thigh (upper) shell swings open on the hinge — each shell keeps its own IMU fixed in place: one on the upper side, one on the lower (shin) side.



IMU Δ = joint angle

As the knee flexes, the upper shell rotates on the hinge relative to the lower. The angle between the two IMUs is exactly the joint angle — fused with the camera by an EKF.

— OVERVIEW

Doctors configure a patient's exercises; patients record themselves and get long-term recovery insights. 3D keypoints from RTMPose3D fuse with an optional ESP32 IMU brace over BLE through an Extended Kalman Filter that weights each camera update by detection confidence. Won the Healthcare track at Nexhacks @ CMU (1500+ participants) and went live on TestFlight in Feb 2026.

THE BRACE

The hinged knee cuff splits into two shells that clamp the leg. Each shell carries its own MPU-6050 IMU; a XIAO ESP32-C6 reads both over I²C at 100 Hz and streams packetized orientation to the app over BLE every 3 measurements. This doesn't cause an issue for our kalman filter implementation since it's not in real-time, but rather gets processed after-the-fact. As the knee flexes, the hinge rotates the shells relative to each other — and the angle between the two IMUs is exactly the joint angle the EKF needs.

HOW IT WORKS

Camera + IMU time-sync over BLE

100 Hz IMU reads on the XIAO batch into 33.3 Hz BLE notifications (transmission every 3 measurements); a round-trip-time calibration (the app sends an ack, RTT/2 is added to each timestamp) aligns the IMU stream to the recorded video.

EKF for joint-angle fusion

State = [joint angle, IMU bias]. The IMU drives the angle at high rate; each CV estimate is a corrective update weighted by detection confidence — high-confidence frames pin the bias (CV angle − IMU angle).

Dual video pipeline

Frames stream to the Overshoot API for live activity classification while simultaneously recording locally; on session end the full clip runs through RTMPose3D for 3D keypoints used by the EKF.

Hardware-optional by design

The IMU brace sharpens accuracy but isn't required — the same EKF falls back to a constant-velocity model so the app still works phone-only.

RESULTS

- 🏆 **1st place** — Healthcare track, Nexhacks @ CMU (1500+ participants).
- In-session activity classification plus per-session and cross-session joint-angle progress.
- Currently requires hefty cloud compute for skeleton-tracking.

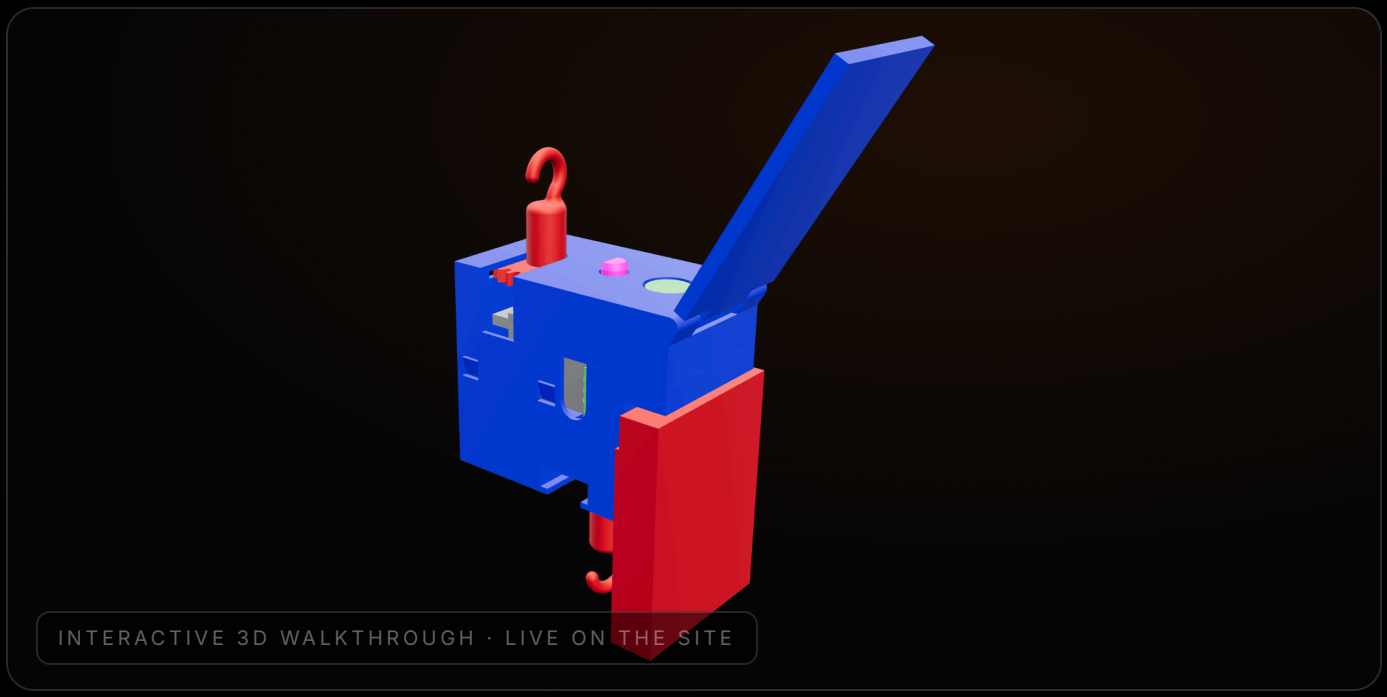
VIDEOS

- ▶ Hackathon demo — youtu.be/TumBOi9pYTY ↗
- ▶ Full walkthrough — youtu.be/243DQZQlmLI ↗

04 FULL-STACK IOT SMART BLINDS

BLINDMASTER

Smart blinds you control, schedule, and calibrate end-to-end — a Flutter app talks to an Express server over Socket.IO, which relays to ESP32-C6 hubs, with BLE-only provisioning for the first handshake.

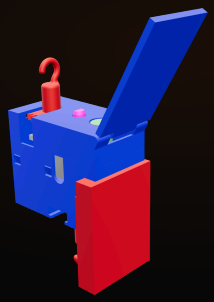


Full-stack IoT · Solo · 3 codebases

ESP32-C6 · ESP-IDF | FreeRTOS · NimBLE | Express · Socket.IO | Flutter | PostgreSQL · MongoDB

GitHub ↗

ROLE	STACK	WHEN	STATUS
Solo · CAD + MFG + firmware + backend + app	ESP-IDF · Express/Socket.IO · Flutter	Ongoing · pre-release	Functional prototype

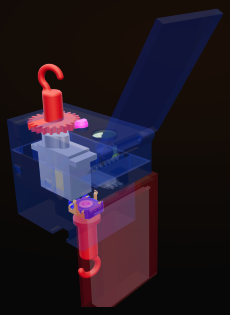


~90 × 132 × 40 mm

The whole hub is about 90 × 132 × 40 mm. Everything (servo, drivetrain, battery and the ESP32-C6) lives in one palm-sized enclosure.

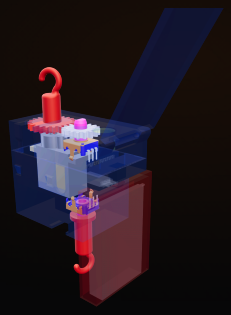
Clips on

The back plate attaches the hub to the blind’s headrail, and the entire top segment clips into the bottom for easy assembly.



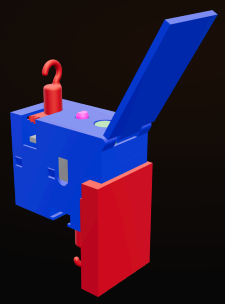
Tilt & track

A continuous-rotation servo turns the hook-gear; a meshed encoder gear feeds the rotary encoder that counts ticks — not quite 1:1, but it doesn't matter since the user calibrates during setup anyway. The bottom hook is a separate manual wand: turn the blind by hand and the servo follows.



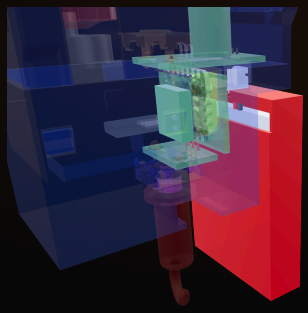
0 → 5 → 10

One 0–10 model (5 = open): the encoder counts 24 ticks per revolution, measuring the blind’s real travel compared to the calibrated values, then shares the classified 0-10 bin verbatim across the firmware, server and app.



XIAO ESP32-C6

A Seeed XIAO ESP32-C6 runs the show — ESP-IDF + FreeRTOS, NimBLE for setup, Socket.IO for control — seated in the heart of the device, with the USB-C port out the side for flashing and, in the current version, charging.



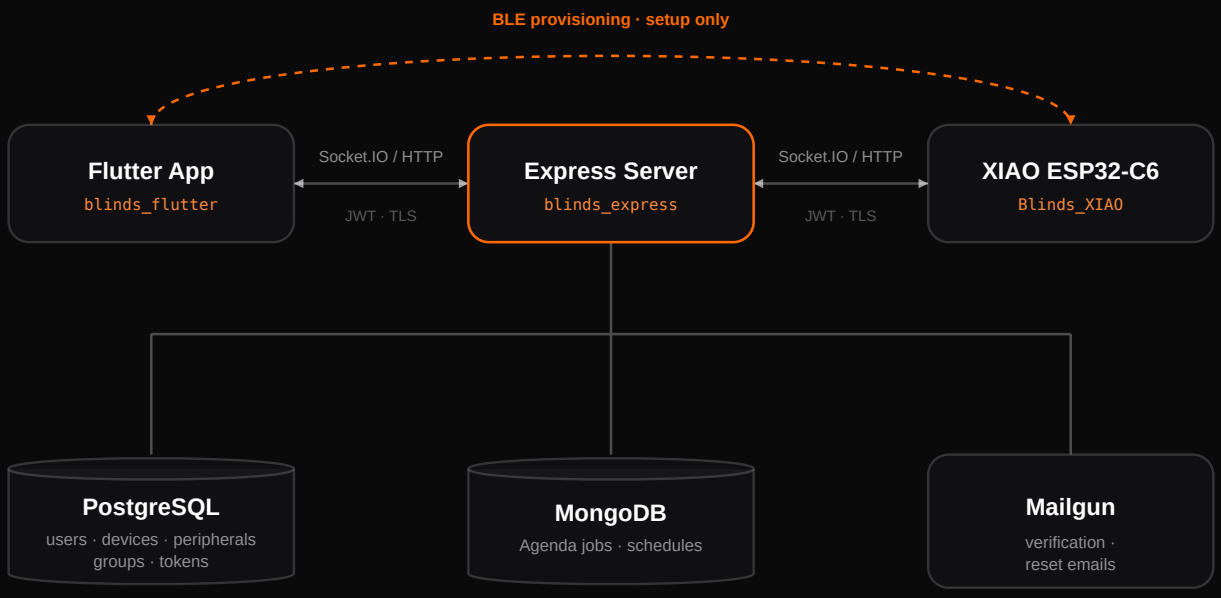
App ↔ Server ↔ Hub

One node in a full stack: the Flutter app and ESP32-C6 hubs talk only through the Express server (Socket.IO/HTTP over TLS); BLE links the phone to the device for one-time provisioning.

— OVERVIEW

A Flutter app talks to an Express server over Socket.IO/HTTPS, which relays commands to ESP32-C6 hubs — also over Socket.IO — with BLE used only for the initial Wi-Fi + auth handshake. Blind position is unified across all three codebases as an 11-step model (0–10, where 5 = fully open).

— SYSTEM ARCHITECTURE



Server-as-relay topology: the Flutter app and ESP32-C6 hubs talk only through the Express server (Socket.IO/HTTP over TLS); BLE links the phone to the device for one-time provisioning.

- **Server as relay** — all app ↔ device traffic routes through the server after provisioning.
- **Dual-path updates** — Socket.IO for real-time, HTTP polling as a fallback.
- **BLE-only provisioning** — the phone delivers Wi-Fi creds + initial JWT straight to the ESP32, never via the server.
- **One 11-position model** shared across HTTP, Socket.IO, and BLE keeps the firmware tiny and the UI legible.

HIGHLIGHTS

One drivetrain, closed-loop both ways

A continuous-rotation servo turns a gear that tilts the blind; a meshed encoder gear feeds a rotary encoder that counts ticks for absolute position. A second encoder is a manual wand — turn the blind by hand and the servo follows it.

Calibration as a Socket.IO handshake

A multi-stage exchange (`calib_start → stage1 → stage2 → calib_done`): the device records the "up" then "down" tick extremes as the user moves the blind, and the tick→position mapping is saved to NVS — the server is a pure relay.

BLE for provisioning only

Wi-Fi creds + JWT go over a custom NimBLE service; once provisioned, BLE goes dormant and all real-time traffic switches to Socket.IO. Trade-off: LE Secure Connections without bonding, assuming physical proximity at setup.

Power management on the ESP32-C6

Dynamic frequency scaling (80–160 MHz), light sleep when idle, GPIO-gated servo power, and a MAX17048 fuel gauge for state-of-charge telemetry and a low-battery warning under 20%.

Hardened backend

Layered rate limits (HTTP, auth, WS connects/messages, email), JWT + Argon2 auth, and boot-time cleanup of stale connections and expired tokens so restarts don't leave zombie devices.

STATUS

- So close to having a demo! Functional prototype — provisioning, real-time control, scheduling, multi-device groups, and the full user lifecycle all work.
- A `taskdrivenpowersave` branch (deeper power management) will replace main; then it may be released, with the app on TestFlight and the hardware open-sourced.

05 ROBOTICS · EMBEDDED · 🏆 1ST · ECE445M FINAL RACE

TWEINSTEIN

An autonomous racer on a **custom RTOS platform** — it wall-follows with a PD controller, classifies oncoming objects as walls or cars, and **overtakes** through a dedicated state machine, with a residual-RL model riding alongside. It **won the ECE445M final race.**

Custom RTOS (C) · LD19 LiDAR · IMU · IR · CAN · DAC · MotorBoard v7 · Residual RL

Code ↗ · Course Info ↗

ROLE	TIMELINE	CONTEXT	STACK
RTOS components · classical control algorithm + overtaking and collision avoidance	Spring 2026	ECE445M lab final · team	RTOS · LiDAR · RL residual

— OVERVIEW

A small autonomous racer that follows walls with a PD controller (modulating its distance setpoint), classifies oncoming objects as static walls or moving cars via an ego-motion-compensated range-rate residual, and executes overtakes through an `S_FOLLOW → S_DETECT → S_COMMIT → S_PASS → S_REJOIN` state machine. A residual-RL model runs alongside, gated to only apply while wall-following.

— THE PROBLEM

ECE445M's lab final is a head-to-head autonomous race: stay on the track, detect and pass slower cars, and don't crash. On top of that, I built the underlying platform — a custom RTOS with a FAT filesystem, heap management, and dynamic process loading from disk — so the racing logic had a real operating system underneath it.

HIGHLIGHTS

Classical control for the bulk path

The PD wall-follower modulates `dist_ref_cur` to shift laterally — the *same* low-level controller does both following and overtaking, just with a different setpoint. The RL residual is strictly gated to `S_FOLLOW` so it never fights the passing logic.

Range-rate object classifier

An ego-motion-compensated residual `r = front_filt - front_pred` accumulates to distinguish static walls (approaching at ego speed) from moving cars; tuning chases false positives on apexes.

Overtaking state machine

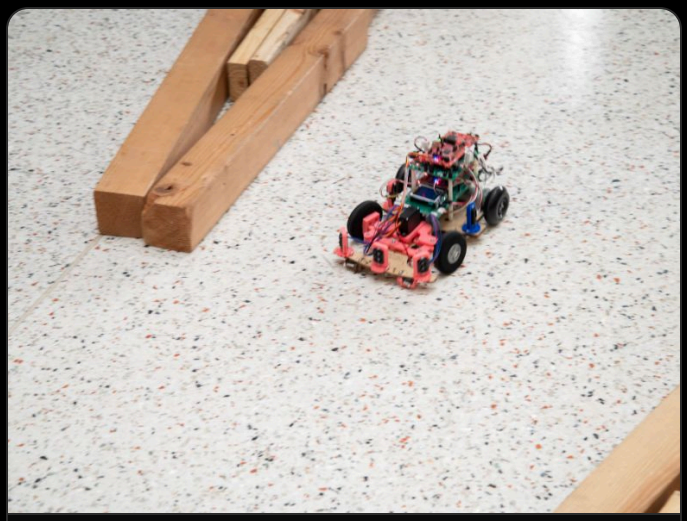
`S_FOLLOW → S_DETECT → S_COMMIT → S_PASS → S_REJOIN`, with timing guards (`T_PASS_MIN`, `T_CLEAR_MS`) against premature rejoins.

Velocity calibration as a routine

A first-class `RobotCalib()` drives perfectly straight and computes top speed from the LiDAR range derivative, printing the result to the ST7735 — turning a magic constant into a measured value.

GALLERY

The team with TweinStein after our race day victory.



TweinStein navigating the track between barriers.

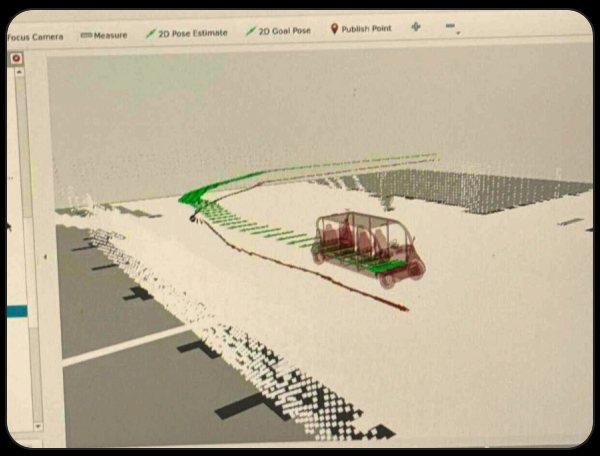
RESULTS

- 🏆 **1st place** in the ECE445M final race (F-0001 format) at UT Austin.
- Classical-control overtaking integrated with an RL residual, on a custom RTOS.
- LiDAR/IMU telemetry logged per state-machine branch for post-run failure analysis.
- Honest limitation: the overtaking feature still needs empirical tuning — and the robot has, regrettably, once driven off a table.

06 ROBOTICS · AUTONOMY · LOCALIZATION

LIDAR SLAM &LOCALIZATION

A ROS2 package that does offline SLAM, real-time **map-based localization**, and incremental map addition from one stack — validated to **~5 cm** against GPS ground truth.



ROS2 | C++ / PyBind | KISS-ICP | KISS-SLAM | Open3D | OpenCV ORB

GitHub (ROS2 nodes) ↗

GitHub (KISS_ICP adaptation) ↗

GitHub (KISS_SLAM adaptation) ↗

ROLE	TIMELINE	CONTEXT	ACCURACY
Solo developer for the LiDAR SLAM and map-based localization stack	May – Aug 2025	UT Dallas NOVA Team	Sub-5 cm vs GPS truth

— OVERVIEW

I forked and extended KISS-ICP, KISS-SLAM, and Map-Closures so the same stack could do offline SLAM, real-time localization against a pre-built static map, and additive mapping — all from a single ROS2 package with three nodes (mapping, localization, map_addition).

— THE PROBLEM

The NOVA team needed localization against a *pre-built static world map*, not just the dynamic local maps the off-the-shelf KISS-* libraries produce. They also needed to start SLAM at a non-origin keypose (so maps stitch cleanly to an existing global map) and close the loop against that global map — not only against recent local scans.

HIGHLIGHTS

Loop closure against a global map

Stock Map-Closures only matches recent local maps. I added a cv2.ORB detector whose params scale with point-cloud max dimension, injected the global map into the HBST with a *negative* ID, and branched KISS-SLAM to add a pose-graph edge straight to origin on a negative-ID match.

Non-origin SLAM start

Overloaded KISS-SLAM init to accept an initial KeyPose from config, so maps centered elsewhere in the world stitch cleanly onto the pre-existing global map.

Static-map localization

Overloaded KISS-ICP init and odometry to localize against a static world map rather than rebuilding a dynamic local map every frame.

ICP + features together

Combining ICP with feature matching gave robust convergence and the sub-5 cm accuracy the team needed on bag-replayed data.

DEMO

▶ LiDAR SLAM demo `youtu.be/qvsudqgiFFI` ↗

RESULTS

- Sub-5 cm localization accuracy vs simulated GPS ground truth on ROS2 bag tests.
- Mapping from scratch (KISS_SLAM), localization from pre-existing map (KISS_ICP modified), adding to an existing map (KISS_SLAM modified) all shipped.

07 SOFTWARE · WEB · COMPUTER VISION · ★ MOST RISKY @ TVG

HARMONIUM

A browser app where you **play piano or guitar with your hands** via webcam tracking, work through curated lessons, or jam with a friend over WebRTC — built in under 24 hours.

React · Vercel | Tone.js | Mediapipe | WebRTC

GitHub ↗

ROLE	TIMELINE	CONTEXT	STATUS
Piano tone mapping · Guitar rendering and tone mapping · MediaPipe filtering	Oct 2025 · <24 hrs	TVG Vibeathon · Most Risky Hack	Shipped & live

— OVERVIEW

Buying a physical instrument just to find out whether it suits you is expensive — and there's no good free way to noodle around with one first. Harmonium is a friction-free way to try (and to goof off with friends): free-play piano and guitar, curated piano lessons, and multiplayer jamming over WebRTC, all in the browser.

HIGHLIGHTS

Frequency interpolation for missing samples

Only some note samples were available, so I used Tone.js frequency interpolation to synthesize the in-between tones — a direct application of attack-release envelopes from my Linear Systems & Signals course.

EMA filter on hand tracking

Raw Mediapipe keypoints jittered and made the on-screen instrument feel unstable; an exponential moving average on the landmarks smoothed control — another signal-processing payoff from coursework.

Multiplayer over WebRTC

Peer-to-peer audio/data so two people can jam together in real time, no server in the hot path.

Built in under a day

Scoped aggressively to ship a genuinely playable instrument in <24 h — the ambition is what earned the "Most Risky Hack" prize.

DEMO

▶ Harmonium demo — youtu.be/mPMlWaZJMw0 ↗

RESULTS

- "Most Risky Hack" winner at the TVG Vibeathon.
- Free-play piano + guitar, curated piano lessons, and WebRTC multiplayer.
- Honest limitation: lessons are only implemented for piano so far.

TOOLKIT

EMBEDDED & LOW-LEVEL

- Embedded C / C++
- ARM Assembly
- Device Drivers
- GPIO · UART · USB
- BLE
- Memory & Power Mgmt
- Arduino

ROBOTICS, CONTROLS & AUTONOMY

- ROS
- Computer Vision
- Control Systems
- Machine Learning
- Signal Processing
- Motor Control
- LiDAR · Radar
- Sensor Fusion
- Kalman Filtering
- Visual Odometry

SOFTWARE & LANGUAGES

- C/C++
- Python
- C#
- Java
- Verilog
- MATLAB / Simulink
- Docker
- FastAPI

HARDWARE, VISION & SIM

- PCB Design
- SolidWorks · Fusion 360
- Ansys / FEA
- 3D Printing
- Machine Learning
- Signal Processing

HONORS

🏆 1st

Autonomous Car Race

ECE445M F-0001 race @ UT Austin · Apr 2026 →

🏆 1st

Analog Devices Sensor Fusion Track

StarkHacks @ Purdue · Apr 2026 →

🏆 1st

Healthcare Track

Nexhacks @ CMU · Jan 2026 →

4.00

GPA at UT Austin

ECE Honors · view coursework →

★

Most Risky Hack

TVG Vibeathon · Oct 2025 →

🎓

Engineering Honors Scholarship

Virginia & Ernest Cockrell Jr. · UT Austin

STANDARDIZED TESTS **1590**/1600 · SAT **1520**/1520 · PSAT

WHAT I'VE STUDIED

ECE Honors coursework at UT Austin — embedded systems, computing, signals & circuits, and the underlying math.

EMBEDDED + REAL-TIME SYSTEMS

ECE445M	Embedded & Real-Time Systems Laboratory (RTOS)	Spr '26
ECE319H	Intro to Embedded Systems (Honors)	Spr '25

ELECTRICAL ENGINEERING & CIRCUITS

ECE411H	Circuit Theory (Honors)	Fall '25
ECE302	Intro to Electrical Engineering	Fall '24

SOFTWARE & COMPUTING

COE379L	Software Design for Responsible Intelligent Systems	Fall '25
CS309	Quantum Computing I (FRI)	Spr '25
ECE312H	Software Design & Implementation I (Honors)	Spr '25
ECE306	Introduction to Computing	Fall '24

COMMUNICATION & INNOVATION

ECE333T	Engineering Communication	Spr '26
MAN327	Innovation & Entrepreneurship	Spr '26

COMPUTER ARCHITECTURE & DIGITAL LOGIC

ECE460N	Computer Architecture	Fall '25
ECE316	Digital Logic Design	Spr '25

SIGNALS & APPLICATIONS THEREOF

ECE385J	Neural Engineering	Spr '26
ECE313H	Linear Systems & Signals (Honors)	Fall '25

MATHEMATICS

M340L	Matrices & Matrix Calculations	Fall '25
M325K	Discrete Mathematics	Sum '25
M427J	Differential Equations w/ Linear Algebra	Spr '25
M427L	Multivariable Calculus (Honors)	Fall '24

Foundational calculus, engineering physics, and university core requirements were cleared through AP and transfer credit (26 hours).

LET'S BUILD SOMETHING

adipu@utexas.edu

aditya.pulipaka.com ↗ | LinkedIn ↗ | GitHub ↗

ADITYA PULIPAKA · AUSTIN, TX · JUNE 2026