

# PORTFOLIO - Muhammad Refansa Annaafi

Email: [muhammad.refansa01@gmail.com](mailto:muhammad.refansa01@gmail.com)

Website: <https://muhammadrefansaa.vercel.app>

LinkedIn: <https://www.linkedin.com/in/muhammadrefansa>

GitHub: <https://github.com/Rennaafi>

## 1. Edge AI Motion-State Classifier: ESP32 Sensing, Inference in the FPGA Fabric

*Individual project · 2026.09*

This project classifies each motion window as stationary, tilt, freefall, impact, or shake, with the neural network running as hardware in the Zynq-7010's FPGA fabric instead of on a PC. The data path runs from an MPU6050 (I2C) to ESP32 feature extraction, UART frames, a Zynq ARM core running bare-metal C, an AXI4-Lite interface, and an hls4ml MLP IP in the FPGA fabric. The resulting class is then sent back to the ESP32 web dashboard.

I built a browser-based data logger inside the ESP32 dashboard (264 labeled windows: 53 stationary, 54 tilt, 69 freefall, 55 impact, 33 shake), a 581-parameter Keras MLP, an HLS IP with an io\_parallel core and an AXI4-Lite register interface, and a bare-metal C driver that receives UART frames, writes the IP's registers, and returns the class.

The streaming ports came out 480-bit and 80-bit, which no standard DMA accepts, so I switched to io\_parallel with AXI4-Lite. ap\_fixed<16,6> kept full accuracy, whereas ap\_fixed<8,3> lost about 21 points. A ReuseFactor of 16 with the Resource strategy cut DSP use from 577% to 52% (LUT 42%, FF 34%, BRAM 18%).

For verification, I compared the fixed-point accuracy against the Keras model in Python, synthesized and implemented the IP in Vivado, and ran the complete chain on the board (2026.09.20). The result is 77.4% accuracy over 5 classes, about 150 ns inference latency in the fabric, and live results on the dashboard.

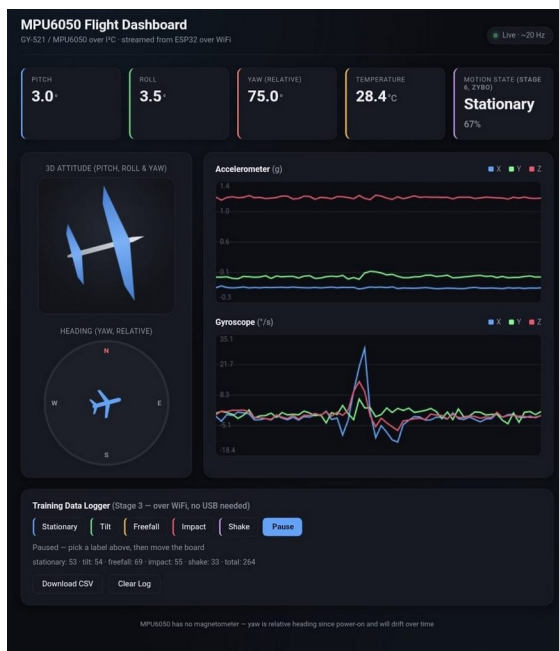


Fig. 1(a). Live class from the Zybo (Stationary) and the 264-window training logger

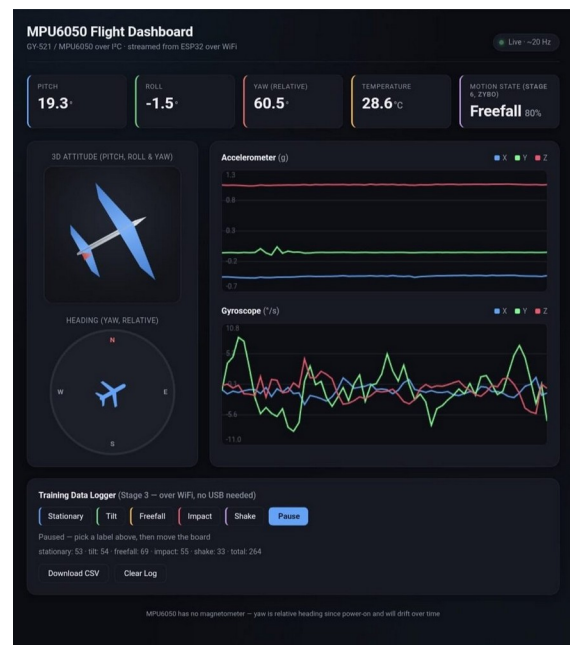


Fig. 1(b). Live class from the Zybo: Freefall

**Tools:** Python, TensorFlow/Keras, hls4ml, Vitis HLS, Vivado, Vitis, bare-metal C, Arduino C++, ESP32, Zybo Z7-10 (Zynq-7010), MPU6050, UART, AXI4-Lite

**Repository:** [github.com/Rennaafi/mpu6050-zybo-classifier](https://github.com/Rennaafi/mpu6050-zybo-classifier)

## 2. RTL-to-GDSII Physical Design and Timing Closure with OpenROAD

*Individual study · 2026.08 ~ 2026.09*

The goal of this study was to reproduce a complete backend flow with open-source tools and to practice acting on real signoff data, on GCD (Nangate45 and SkyWater Sky130) and on `ibex_core`, a real RISC-V CPU of about 31,761 instances. The flow runs from RTL through Yosys synthesis, floorplan, placement, CTS, and routing to OpenSTA signoff and GDS (KLayout).

I wrote the flow configurations and SDC constraints for each design, ran a 5-point clock-period sweep (0.25–0.60 ns) and a cross-process run on Sky130, and carried out ECO experiments with OpenDB (`dbInst::swapMaster`) and `repair_timing`.

GCD at 4 GHz failed with 51 setup violations (WNS –0.16 ns) through 14 logic levels. `report_clock_min_period` showed a ceiling of about 2.44 GHz, so I fixed the constraint, not the logic: 2 GHz gave a clean signoff and 65% less power (6.38 → 2.22 mW). On `ibex_core` I found an undersized MUX2\_X1 driving about 32 fF where similar cells drove about 2 fF, upsized it by hand, and then let `repair_timing` close the rest.

I ran OpenSTA (`report_checks`, `report_clock_min_period`) after every change and inspected the layouts in KLayout. A later full manual reroute failed with 3 disconnected nets, so the `ibex` result is recorded as STA-verified, not signed off. For GCD, WNS reached 0.00 ns with 0 violations at 2 GHz. For `ibex_core`, all 240 setup violations were closed (WNS –15 ps → +5 ps, 8 cells upsized, 2 pin swaps). Sky130 ran about 6x slower per path than Nangate45 (`fmax` about 390 MHz vs about 2.44 GHz).

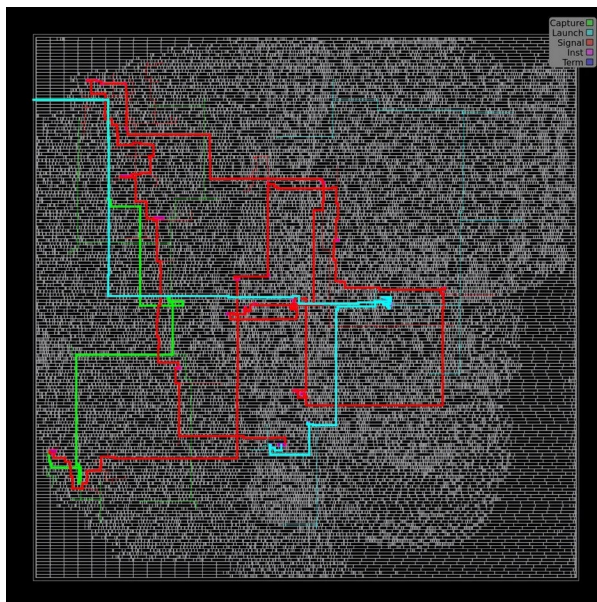


Fig. 2(a). `ibex_core`: worst path on the routed layout

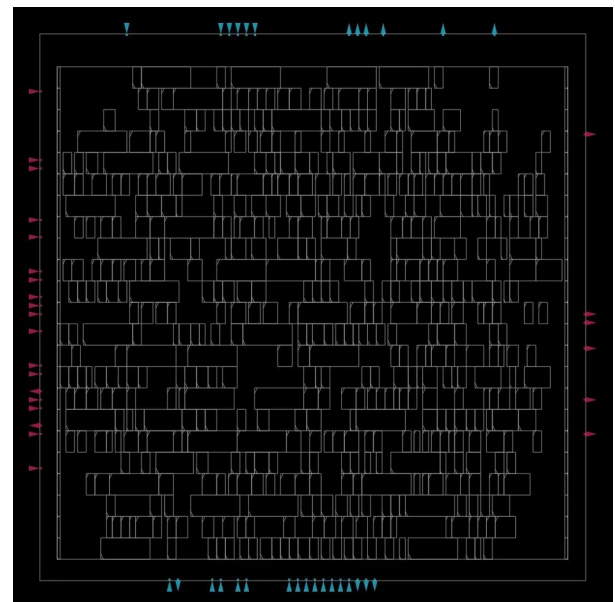


Fig. 2(b). GCD: placement on Nangate45

**Tools:** OpenROAD-flow-scripts, OpenROAD, OpenSTA, Yosys, KLayout, Nangate45, SkyWater Sky130, SDC, Tcl, Linux (WSL)

**Repository:** [github.com/Rennaafi/ibex-core-timing-eco](https://github.com/Rennaafi/ibex-core-timing-eco)

### 3. PenPlasma: Portable Microplasma Excitation Source (Undergraduate Thesis, Manuscript under Review)

*Undergraduate thesis, individual (supervisor Dr. Tomy Abuzairi) · 2025.08 ~ 2026.06 · second author at HardwareX*

Elemental-analysis instruments such as AAS and ICP-OES cost US\$50,000–200,000. PenPlasma is an open-source, battery-powered point-discharge microplasma source built for US\$6.46, intended as the excitation source for future optical-emission elemental detection. It is my thesis and is now a manuscript under review at HardwareX. The data path runs from an 18650 Li-ion cell (3.7 V, 2000 mAh) to an HV-1 flyback blocking oscillator (D880 BJT), which produces 4.7 kV peak-to-peak at about 61.7 kHz. This drives Kenko KE-100 pen tips (hollow + point) in a 3D-printed holder with a gap of 0.5–2.5 mm, creating a point discharge and optical emission.

I built the prototype and its 3D-printed holder with an adjustable gap, an LTspice model of the generator, and the measurement workflow: a high-voltage probe (1000:1) and oscilloscope, emission intensity in ImageJ, and smartphone spectral analysis. The design files are released as open hardware (CERN-OHL-S-2.0).

I measured the generator against the LTspice model: 3.04 kV positive peak (sim about 3.5 kV), –1.63 kV undershoot (sim about –0.5 kV), and about 61.7 kHz (sim about 50 kHz). I traced the more than 3x larger undershoot to transformer leakage inductance above the assumed  $k = 0.99$  in the encapsulated module. Because the gap trades duration against stability and intensity, I swept 5 gaps with 5 replicates each, using the coefficient of variation (CV) for stability.

Verification consisted of 5 independent waveform measurements, a comparison with LTspice, 5 replicates per gap, and a spectrum check (peaks at about 450 and 626 nm, nitrogen emission). A gap of 1.5 mm gave the longest discharge (163.4 s), the best stability (CV 11.3%), and adequate intensity (1.76 a.u., CV 8.7%). Intensity peaked at 0.5 mm (2.06 a.u.), while gaps of 2.0 mm and above became unstable (CV up to 50.6%). Among the comparable systems reviewed, it is the only one combining a verified cost under US\$10, 3.7 V battery operation, full output characterization, and a systematic gap study.



Fig. 3(a). Assembled PenPlasma prototype

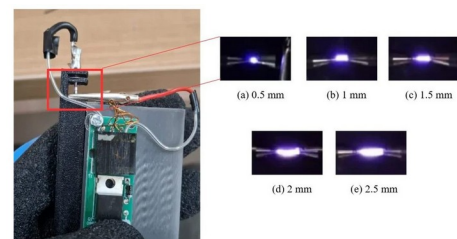


Fig. 3(b). Plasma at gaps of 0.5 to 2.5 mm

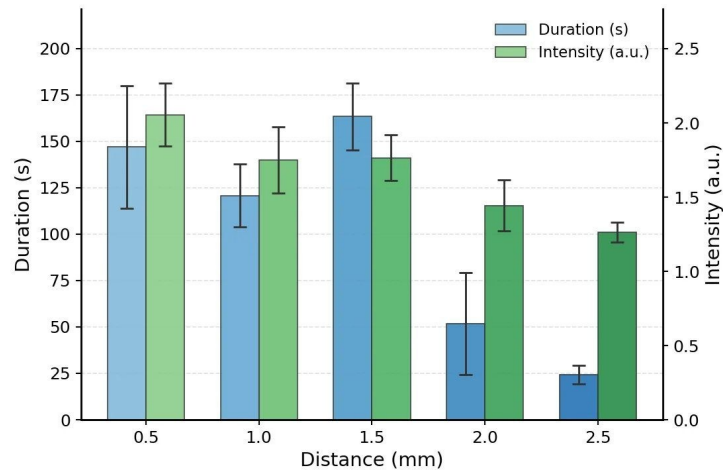


Fig. 3(c). Duration and intensity per gap (n = 5)

**Tools:** HV-1 flyback blocking oscillator, LTspice, high-voltage probe, oscilloscope, ImageJ, smartphone spectral analysis, 3D printing, 18650 Li-ion

## 4. Audio Codec Bring-Up on Zynq: AXI IIC and Bare-Metal C

*Individual project · 2026.09*

This project configures the Zybo Z7-10's onboard SSM2603 audio codec, which is wired to the FPGA side of the Zynq, so the Arm core needs an I2C master in the fabric. The data path runs from the Cortex-A9 (bare-metal C) over AXI and SmartConnect to an AXI IIC block in the FPGA fabric, then over I2C (pins N18/N17) to the SSM2603 (address 0x1A).

I built a Vivado block design (Zynq PS + AXI IIC + SmartConnect), pin constraints, and a bitstream, plus a Vitis platform and a bare-metal C driver. A helper packs the codec's 7-bit register address and 9-bit data into its 2-byte I2C write, and a 7-register sequence runs reset, power-up, analog path, digital path, I2S 16-bit format, 48 kHz sample rate, and activate. The new Vitis flow (SDT) takes base addresses instead of device IDs, which I confirmed in the driver header rather than following older tutorials.

I checked the UART log at 115200 baud for each write. A first run showed the factory demo and a failed JTAG connection; the cause was the boot-mode jumper set to SD instead of JTAG. In the end, all register writes were acknowledged by the codec on real hardware.

```
Opened with baud rate: 115200
AXI IIC initialized successfully
SSM2603 reset sent
SSM2603 power-up sent
SSM2603 analog audio path configured
SSM2603 digital audio path configured
SSM2603 digital interface format configured
SSM2603 sample rate configured
SSM2603 activated
□
```

Fig. 4. UART log: every register write acknowledged

**Tools:** Vivado, Vitis (SDT flow), bare-metal C, Zynq-7010, AXI IIC, AXI SmartConnect, I2C, UART, SSM2603

**Repository:** [github.com/Rennaafi/zybo-i2c-ssm2603-bringup](https://github.com/Rennaafi/zybo-i2c-ssm2603-bringup)

## 5. From Logic Gates to a RISC-V SoC: Twelve Verilog Modules on the Zybo Z7-10

*KCCI FPGA training · 2026*

The aim of this training work was to build digital design up from first principles on real hardware, one module at a time. The progression runs from gates to counters and tick generators, debouncers, FSMs, and UART, then a single-cycle RISC-V core and a small SoC around the core.

I built twelve parameterized Verilog modules, each with a testbench and board constraints (125 MHz board clock, buttons, switches, LEDs). Verification used self-checking testbenches with PASS/FAIL output in simulation, followed by implementation in Vivado and testing on the Zybo Z7-10. The result is a working path from basic logic to a small RISC-V SoC on the board.



Fig. 5. Digilent Zybo Z7-10

**Tools:** Verilog, Vivado, Zybo Z7-10 (Zynq-7010), XDC constraints, simulation testbenches

**Repository:** [github.com/Rennaafi/zybo-fpga-coursework](https://github.com/Rennaafi/zybo-fpga-coursework)

## 6. HC-SR04 Distance Meter over UART in Pure Verilog

*Individual project · 2026*

This project measures distance with an ultrasonic sensor using only FPGA logic, with no processor involved. An FSM drives the trigger pulse and times the echo from the HC-SR04, an echo pulse-width counter and distance conversion produce the reading, and a UART transmitter streams each reading to a serial terminal on the PC. The sensor is wired through Pmod JD and JE, and live distance readings are shown in the terminal.

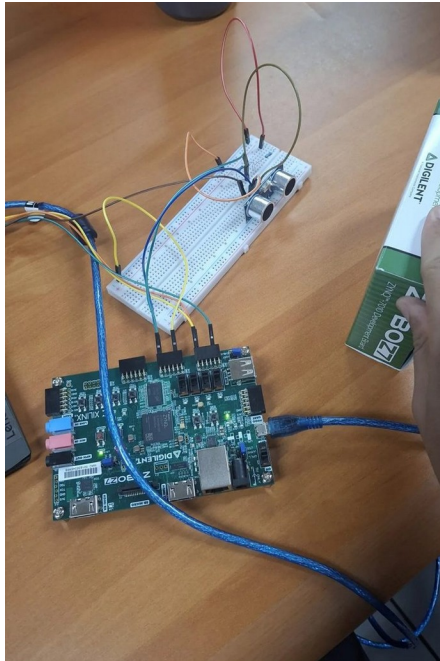


Fig. 6(a). HC-SR04 wired to Pmod JD and JE

```

Listening on COM3 @ 115200 baud, Hit Ctrl-C to quit.
[09:33:26] distance = 10 cm
[09:33:27] distance = 10 cm
[09:33:28] distance = 42 cm
[09:33:29] distance = 0 cm
[09:33:30] distance = 41 cm
[09:33:31] distance = 0 cm
[09:33:32] distance = 40 cm
[09:33:33] distance = 9 cm
[09:33:34] distance = 6 cm
[09:33:35] distance = 7 cm
[09:33:36] distance = 6 cm
[09:33:37] distance = 0 cm
[09:33:38] distance = 3 cm
[09:33:39] distance = 0 cm
[09:33:40] distance = 10 cm
[09:33:41] distance = 3 cm
[09:33:42] distance = 9 cm
[09:33:43] distance = 0 cm
[09:33:44] distance = 12 cm
[09:33:45] distance = 21 cm
[09:33:46] distance = 10 cm
[09:33:47] distance = 0 cm
[09:33:48] distance = 7 cm
[09:33:49] distance = 7 cm
[09:33:50] distance = 4 cm
[09:33:51] distance = 5 cm
[09:33:52] distance = 5 cm

```

Fig. 6(b). Live distance readings over UART

**Tools:** Verilog, Vivado, Zybo Z7-10, HC-SR04, UART, FSM

**Repository:** [github.com/Rennaafi/zybo-z7-hcsr04-distance-uart](https://github.com/Rennaafi/zybo-z7-hcsr04-distance-uart)

## 7. STM32 Development Boards in KiCad 9: 2-Layer STM32F103 and 4-Layer STM32G474RE

*Individual projects · STM32G474RE: KCCI PCB course, 2026.09 ~*

These two projects design compact MCU boards from schematic to fabrication files: a small STM32F103 board, then a Nucleo-64-style STM32G474RE board without the on-board ST-LINK. The F103 power path runs from USB through a 3.3 V LDO to the STM32F103 with a 16 MHz crystal. The G474 path runs from USB-C (5.1 kOhm CC) through a 500 mA polyfuse, a Schottky diode, and an AP2112K-3.3 LDO to the MCU (100 nF decoupling), with a ferrite bead feeding the analog supply.

The F103 is a 2-layer board of 33.5 x 24.2 mm with boot and reset switches and SWD, SPI, and I2C on headers. For the G474, I drew a schematic with every MCU pin on morpho-layout female headers, a 1x6 SWD header, 16 MHz and 32.768 kHz crystals, an LED, a button, and solder jumpers, referenced to ST's Nucleo MB1367 design. The G474 constraints from the brief are 4 layers, at most 100 x 100 mm, 0.3/0.45 mm tented vias, and no ST-LINK.

The F103 has fab-ready Gerbers. For the G474, the schematic was started on 2026.09.26 and the fabrication files are due 2026.10.04.

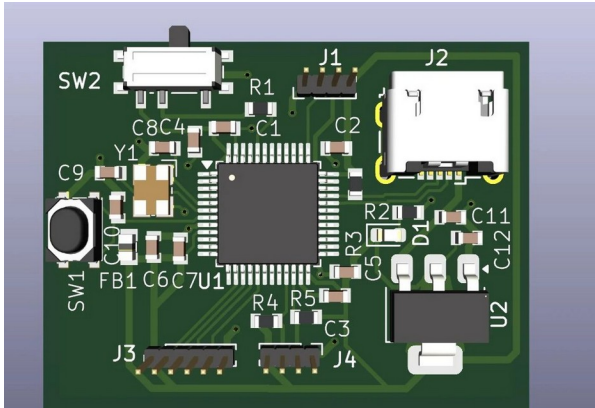


Fig. 7(a). STM32F103 board, KiCad 3D render

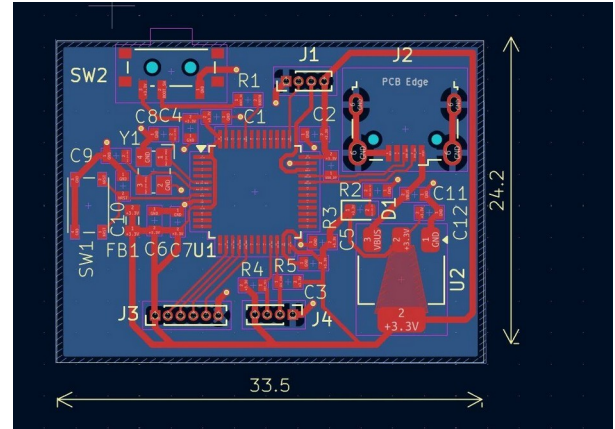


Fig. 7(b). 2-layer layout, 33.5 x 24.2 mm

**Tools:** KiCad 9, STM32F103, STM32G474RE, AP2112K-3.3, USB-C, SWD

**Repository:** [github.com/Rennaafi/stm32f103-dev-board](https://github.com/Rennaafi/stm32f103-dev-board)

## 8. ESP32 IMU Live Dashboard

*Individual project · 2026.09*

This project shows an IMU's attitude live on any phone or laptop, with no cloud, no app, and no router. An MPU6050 (I2C) feeds an ESP32 complementary filter (pitch, roll, heading), which exposes a JSON endpoint, and the dashboard is served from flash over the ESP32's own Wi-Fi access point.

I implemented the sensor fusion with a complementary filter, a static dashboard with an attitude indicator, a compass, and live accelerometer and gyroscope charts polling at about 20 Hz, plus automatic I2C bus recovery. Wi-Fi transmit current spikes could wedge the sensor's I2C bus, so the firmware detects this and recovers the bus by itself. The result is a self-contained live dashboard that later became the front end and data logger of the motion classifier.



Fig. 8. Dashboard served from the ESP32's flash

**Tools:** ESP32, Arduino C++, MPU6050, I2C, Wi-Fi access point, HTML/CSS/JavaScript

**Repository:** [github.com/Rennaafi/esp32-mpu6050-dashboard](https://github.com/Rennaafi/esp32-mpu6050-dashboard)

## 9. PrioWatt: Smart kWh Meter with Priority Load Shedding, and the Despro App

*Capstone, team of 4 · 2025.08 ~ 2025.12*

PrioWatt is an energy monitor that protects a circuit by switching off the least important loads first, and warns the user about unusual consumption. Sensors feed an ESP32 (6 parameters, every 3 s), which runs the priority load-shedding logic for 4 loads and sends data to Firebase (every 5 s) for the Despro Flutter app.

As a team, we built ESP32 measurement of voltage, current, real power, energy, power factor, and frequency; tiered shedding across 4 loads with remote override; and per-socket control and analytics in Despro. My part was the LSTM model behind the app's AI tips, which predicts cost and detects anomalies, plus work on firmware, logging, and app integration.

Verification consisted of bench tests of the complete system covering sampling and logging intervals, shedding behavior, and Wi-Fi stability. The 3 s sampling and 5 s logging held with no Wi-Fi reconnection failures, and the system was delivered in 14 weeks for the design exhibition.



Fig. 9(a). Team and hardware at the design exhibition



Fig. 9(b). Despro app: live power and AI forecast

**Tools:** ESP32, C/C++, Firebase, Flutter/Dart, Python, LSTM

**Repository:** [github.com/Rennaafi/despro-smart-socket-monitor](https://github.com/Rennaafi/despro-smart-socket-monitor)

## 10. FOTRIX: Solar-Powered Seawater Hydrogen and Microplastic Degradation Prototype

*Team of 5 · 2025.01 ~ 2025.11 · PIMNAS 2025 national finalist (PKM-KC)*

FOTRIX is one portable, solar-powered unit that produces hydrogen from seawater and breaks down microplastics at the same time. Sensors for hydrogen production, degradation metrics, and PV power feed microcontroller telemetry to Firebase, and the FOTRIX Flutter app shows live and historical hydrogen readings and controls the actuators and solar tracker.

My part was the sensor and telemetry architecture, the Flutter app with its Firebase backend, and hardware-software integration and system-level debugging. Verification went from lab integration to a 2-week field-testing cycle before the finals, recording behavior, anomalies, and corrective actions, and then a live national evaluation. The prototype ran through the live national demonstration, and the team finished as a PIMNAS 2025 national finalist.

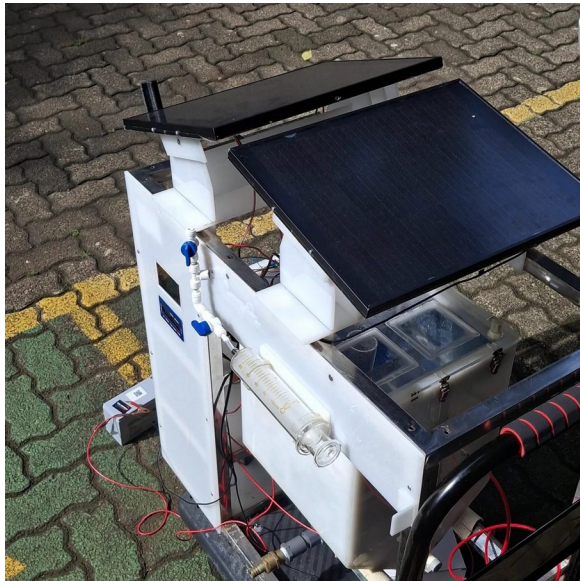


Fig. 10(a). Reactor: solar panel, control box, dosing syringe

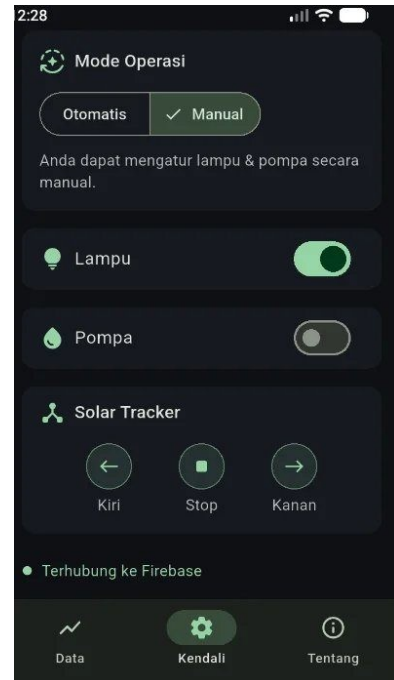


Fig. 10(b). FOTRIX app: actuator and solar-tracker control

**Tools:** Sensors, microcontroller telemetry, PV monitoring, Flutter/Dart, Firebase

**Repository:** [github.com/Rennaafi/fotrix-hydrogen-monitor](https://github.com/Rennaafi/fotrix-hydrogen-monitor)

## 11. Pt100 RTD Temperature Measurement Chain

*Individual project · 2025.08 ~ 2025.12*

This project aims to measure temperature more stably than an MCU's built-in ADC allows. The chain runs from a Pt100 RTD through a Wheatstone bridge to an ADS1115 16-bit ADC (I2C), then to an Arduino, where a Callendar–Van Dusen conversion produces the temperature. I built the bridge conditioning, the ADC interface, the resistance-to-temperature conversion, and a calibration workflow documenting accuracy and repeatability. The result is lower voltage fluctuation than direct MCU ADC sampling (qualitative comparison).

**Tools:** Pt100, Wheatstone bridge, ADS1115, I2C, Arduino, calibration

## 12. Scanning Thermal Array for Presence Detection

*Course project, group of 5 · 2025.08 ~ 2025.12*

This project detects a person's presence without a camera, using a low-resolution thermal sensor that sweeps the room. An AMG8833 8x8 thermal array (I2C) is moved by a servo scan (0°, 90°, 180°, about 500 ms cycle, PWM), and an ESP32 performs threshold detection and sends results over Wi-Fi.

I wrote the ESP32 firmware for I2C acquisition, PWM servo scanning, Wi-Fi transmission, and absolute and differential threshold detection. A continuous-operation test of the complete system ran for 120 minutes of stable operation, and I co-authored the project write-up.

**Tools: ESP32, AMG8833, I2C, PWM servo, Wi-Fi**

## 13. Voice Command Lamp

*Team of 3*

This project switches a lamp on and off by voice through a phone app. A voice command goes to a companion app, then over BLE to an ESP32, which drives a relay and the lamp. The result is a working ESP32-based voice-controlled lighting prototype connected to its companion app over BLE.



Fig. 11. Team, prototype, and companion app

**Tools: ESP32, BLE, relay, mobile app**

## 14. RPLIDAR A1 to Zybo Z7-10

*Individual project · 2026.09 ~*

This project reads a 360° lidar directly on the Zybo, as groundwork for mapping on the robot project. The RPLIDAR A1 (UART) connects to an AXI UART Lite in the FPGA fabric, a bare-metal C parser on the Zynq ARM core extracts points, and a motor PWM block is driven through AXI GPIO.

I built Python host tools (test, viewer, fake lidar, self-test); a Vivado design with AXI UART Lite, AXI GPIO, and my motor PWM block; and a bare-metal C parser with sync detection, a non-blocking console ring buffer, and a watchdog.

The motor PWM xsim testbench passes, and implementation closed with WNS +2.394 ns. The lidar was verified over USB (about 290 points at 6.5 Hz) and the Zybo UART by loopback. I also fixed a timer bug where the global timer only started inside the first sleep call. The lidar and Zybo are each verified on their own; the next step is the direct lidar-to-Zybo link, after confirming the connector pin mapping.

**Tools:** Python, Vivado, Vitis, bare-metal C, AXI UART Lite, AXI GPIO, Verilog, RPLIDAR A1

## 15. Recall: Spaced-Repetition Web App with AI Card Generation

*Individual project · 2026.09 · deployed and in daily use*

Recall is a personal flashcard app for studying FPGA, STM32, and physical design, with cards generated from long transcripts. A transcript is cleaned on the client and split into chunks of about 1,400 words, Workers AI (Llama 3.3 70B, JSON-schema output) generates cards, duplicates are removed, and the cards are reviewed and edited before being saved to a D1 database.

I built a React/TypeScript front end on Cloudflare Workers with a D1 database behind Cloudflare Access, and the AI generation feature with parallel requests and a usage estimate. All bulk writes use one statement over a JSON array, because the free D1 plan limits queries per request, a limit local development does not enforce. The app is deployed and used daily; a transcript of about 2,000 words takes about 13 s and about 240 of the 10,000 free daily AI units.

**Tools:** React, TypeScript, Cloudflare Workers, D1, Workers AI, Cloudflare Access

## 16. OpenUSD with Python: Authoring Scenes and Assemblies from Scripts

*Self-study project · 2026.10 ~*

Digital-twin and simulation tools such as NVIDIA Omniverse are built on OpenUSD. I started with the format itself, authoring scenes from Python with `usd-core` and checking each file in Blender, so that I understand units, references, and materials before using a larger platform. It runs on an ordinary laptop, with no special GPU.

I wrote six scripts: a stage with a cube and a sphere (prims, attributes, transforms); a mesh from raw point, face-count, and index arrays; an STL-to-USD converter; a `UsdPreviewSurface` material bound to the converted part; an assembly of three references to one part, one with a scale override; and a bounce-and-sway animation with time samples.

An STL file carries no units: a 115 mm part imported as 115 m until I scaled the points by 0.001, so I now check Blender's Dimensions panel for every import. I set the stage up-axis and meters-per-unit as file metadata, so Blender imports without a rotation or scale fix. In the assembly, references are links rather than copies: the assembly file holds no mesh data, and editing the part file changes every instance.

Every generated `.usda` file was opened in Blender (Material Preview) to check scale, orientation, color, and animation. Prims, meshes, STL conversion, references, materials, and animation all work and are checked in Blender. Next: `UsdPhysics` joints for a simple robot, then URDF-to-USD and simulation in Isaac Sim.

**Tools:** Python, `usd-core` (OpenUSD), trimesh, Blender

## Link to the Roles I Apply For

The projects above follow one path through the hardware stack: a circuit and a PCB (STM32 boards, the PenPlasma generator), firmware that brings a chip up (ESP32, bare-metal Zynq), logic in the FPGA fabric (Verilog modules up to a RISC-V SoC, an `hls4ml` accelerator), and the physical layout of a real CPU (OpenROAD, `ibex`). Around that sits measurement: sweeps, stability metrics, and signoff reports read before decisions are made.

For embedded and MCU roles, the codec bring-up, the IMU dashboard, and PrioWatt show firmware verified on hardware. For FPGA and SoC roles, the motion classifier and the Verilog coursework show RTL and bus integration. For PCB and hardware roles, the STM32 boards and PenPlasma show circuit design and validation. For test and quality roles, PenPlasma and the RTD chain show structured measurement and documentation.