

포트폴리오 - Muhammad Refansa Annaafi

이메일: muhammad.refansa01@gmail.com

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

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

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

1. Edge AI 모션 상태 분류기: ESP32 센싱, FPGA Fabric 내 추론

개인 프로젝트 · 2026.09

각 모션 원도우를 정지(stationary), 기울임(tilt), 자유낙하(freefall), 충격(impact), 흔들림(shake)의 다섯 가지 상태로 분류하되, 신경망을 PC 가 아닌 Zynq-7010 의 FPGA fabric 에서 하드웨어로 구동하는 프로젝트입니다. 데이터 경로는 MPU6050(I2C) → ESP32 특징 추출 → UART 프레임 → Zynq ARM(bare-metal C) → AXI4-Lite → FPGA fabric 의 hls4ml MLP IP 로 구성되며, 분류 결과는 ESP32 웹 대시보드로 다시 전달됩니다.

ESP32 대시보드 내에 브라우저 기반 데이터 로거(라벨링된 264 개 윈도우: 정지 53, 기울임 54, 자유낙하 69, 충격 55, 흔들림 33), 581 개 파라미터의 Keras MLP, io_parallel 코어와 AXI4-Lite 레지스터 인터페이스를 갖춘 HLS IP, 그리고 UART 프레임의 수신하여 IP 레지스터에 기록하고 분류 결과를 반환하는 bare-metal C 드라이버를 구현했습니다.

스트리밍 포트가 480-bit 와 80-bit 로 생성되어 표준 DMA 가 지원하지 않았기 때문에, io_parallel 과 AXI4-Lite 방식으로 전환했습니다. ap_fixed<16,6>은 정확도를 그대로 유지한 반면 ap_fixed<8,3>은 정확도가 약 21%p 하락했습니다. 또한 Resource 전략에 ReuseFactor 16 을 적용하여 DSP 사용률을 577%에서 52%로 낮췄습니다(LUT 42%, FF 34%, BRAM 18%).

검증은 Python 에서 고정소수점 정확도를 Keras 모델과 비교하고, Vivado 에서 IP 를 합성 및 구현한 뒤, 보드에서 전체 체인을 실행하는 순서로 진행했습니다(2026.09.20). 그 결과 5 개 클래스에서 77.4%의 정확도와 fabric 내 약 150 ns 의 추론 지연시간을 달성했으며, 결과는 대시보드에 실시간으로 표시됩니다.

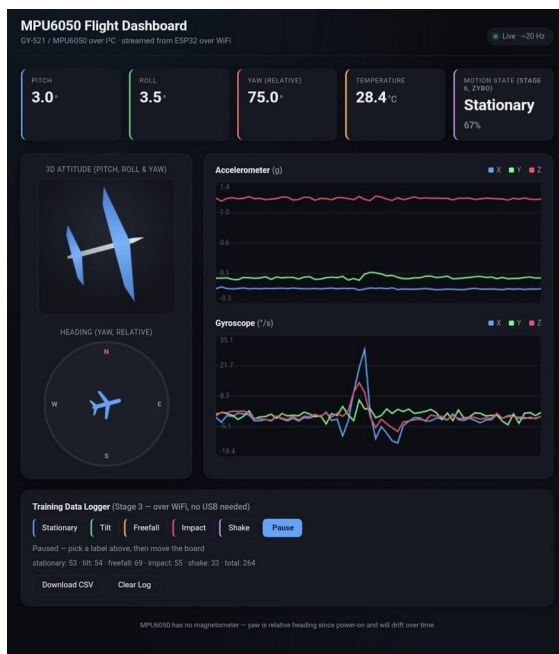


그림 1(a). Zybo 의 실시간 분류 결과(Stationary)와 264 개 윈도우 학습 로거

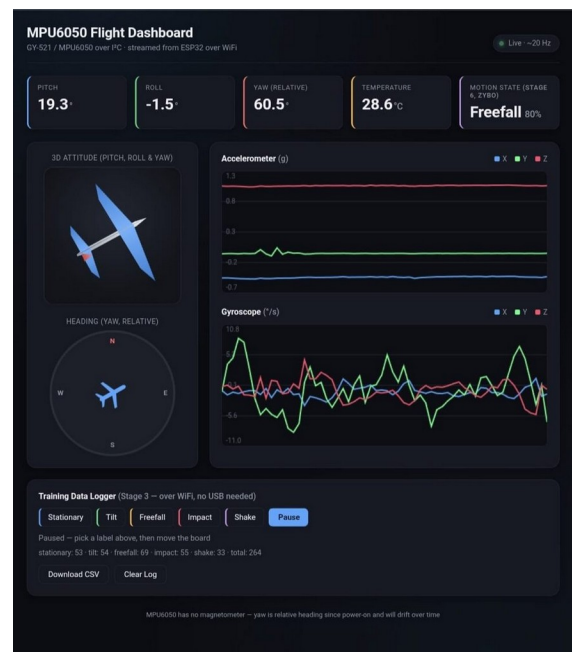


그림 1(b). Zybo 의 실시간 분류 결과: Freefall

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

저장소: github.com/Rennaafi/mpu6050-zybo-classifier

2. OpenROAD 를 이용한 RTL-to-GDSII 물리 설계 및 타이밍 클로저

개인 학습 · 2026.08 ~ 2026.09

오픈소스 도구로 전체 백엔드 플로우를 재현하고 실제 signoff 데이터를 분석하여 대응하는 경험을 쌓는 것을 목표로, GCD(Nangate45 및 SkyWater Sky130)와 실제 RISC-V CPU 인 ibex_core(약 31,761 개 instance)에 적용했습니다. 플로우는 RTL → Yosys 합성 → floorplan → placement → CTS → routing → OpenSTA signoff → GDS(KLayout) 순서로 구성됩니다.

각 설계에 대한 플로우 설정과 SDC 제약 조건을 작성하고, 5 개 지점의 클럭 주기 sweep(0.25~0.60 ns), Sky130 에서의 cross-process 실행, 그리고 OpenDB(dbInst::swapMaster)와 repair_timing 을 이용한 ECO 실험을 수행했습니다.

GCD 를 4 GHz 로 구동했을 때 14 단 로직을 거치며 51 건의 setup violation(WNS -0.16 ns)이 발생했습니다.

report_clock_min_period 결과 한계가 약 2.44 GHz 임을 확인하여 로직이 아닌 제약 조건을 수정했고, 2 GHz 에서 깨끗한 signoff 와 65% 낮은 전력(6.38 → 2.22 mW)을 얻었습니다. ibex_core 에서는 유사한 셀이 약 2 ff 를 구동하는 곳에서 크기가 작은 MUX2_X1 이 약 32 ff 를 구동하고 있음을 발견하여 수동으로 upsizing 한 뒤, 나머지는 repair_timing 으로 해결했습니다.

모든 변경 후 OpenSTA(report_checks, report_clock_min_period)로 검증하고 KLayout 에서 레이아웃을 확인했습니다. 이후 수행한 전체 수동 reroute 는 3 개의 net 이 끊어져 실패했으므로, ibex 결과는 signoff 완료가 아닌 STA 검증 완료로 기록했습니다. GCD 는 2 GHz 에서 WNS 0.00 ns, violation 0 건을 달성했고, ibex_core 는 240 건의 setup violation 을 모두 해결했습니다 (WNS -15 ps → +5 ps, 셀 8 개 upsizing, pin swap 2 건). Sky130 은 Nangate45 대비 경로당 약 6 배 느렸습니다(fmax 약 390 MHz 대 약 2.44 GHz).

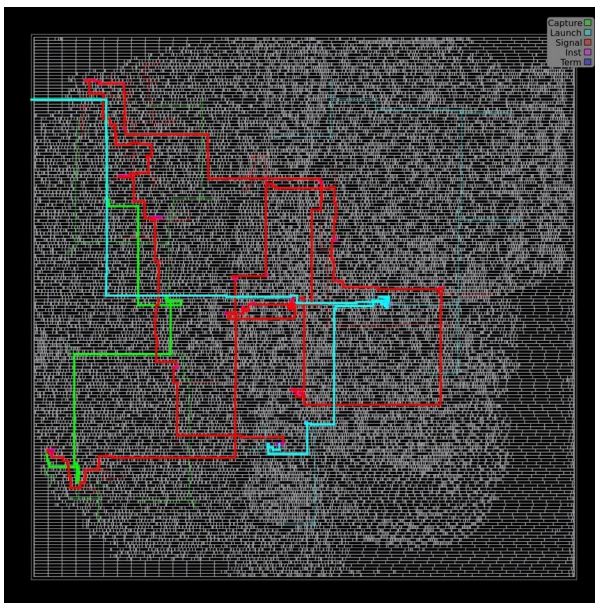


그림 2(a). ibex_core: 라우팅된 레이아웃의 worst path

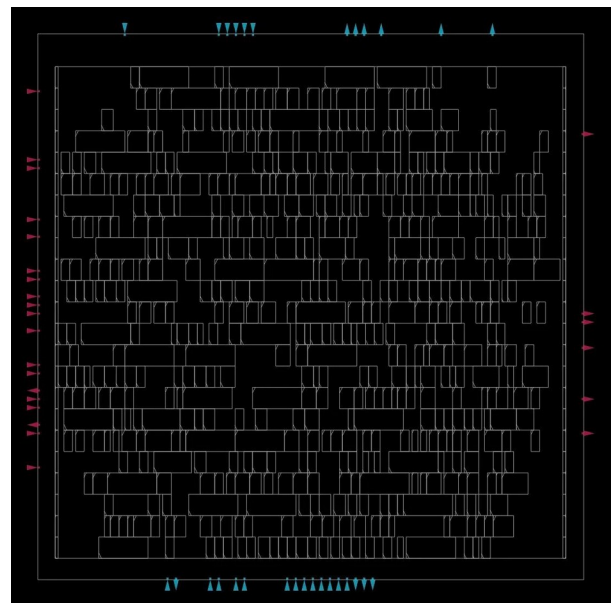


그림 2(b). GCD: Nangate45 placement

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

저장소: github.com/Rennaafi/ibex-core-timing-eco

3. PenPlasma: 휴대용 마이크로플라즈마 여기 소스 (학사 졸업 논문, 심사 중인 원고)

학사 졸업 논문, 개인 (지도교수 Dr. Tomy Abuzairi) · 2025.08 ~ 2026.06 · HardwareX 제 2 저자

AAS, ICP-OES 와 같은 원소 분석 장비의 가격은 US\$50,000–200,000 에 달합니다. PenPlasma 는 향후 광학 발광 기반 원소 검출의 여기 소스로 활용할 수 있도록 US\$6.46 에 제작한 오픈소스 배터리 구동 point-discharge 마이크로플라즈마 소스이며, 저의 졸업 논문이자 현재 HardwareX 에서 심사 중인 원고입니다. 18650 Li-ion 셀(3.7 V, 2000 mAh)이 HV-1 flyback blocking oscillator(D880 BJT)를 구동하여 약 61.7 kHz 에서 4.7 kV peak-to-peak 를 생성하고, 이 전압이 3D 프린팅 홀더에 고정된 Kenko KE-100 펜 팁(중공형 + 점형, 간격 0.5–2.5 mm)에 인가되어 point discharge 와 광학 발광을 일으킵니다.

간격 조절이 가능한 3D 프린팅 홀더가 포함된 프로토타입, 발생기의 LTspice 모델, 그리고 측정 워크플로우(고전압 프로브 1000:1 과 오실로스코프, ImageJ 를 이용한 발광 강도 분석, 스마트폰 스펙트럼 분석)를 구축했습니다. 설계 파일은 오픈 하드웨어(CERN-OHL-S-2.0)로 공개했습니다.

발생기를 LTspice 모델과 비교하여 측정한 결과, 양의 피크는 3.04 kV(시뮬레이션 약 3.5 kV), 언더슈트는 -1.63 kV(시뮬레이션 약 -0.5 kV), 주파수는 약 61.7 kHz(시뮬레이션 약 50 kHz)였습니다. 3 배 이상 큰 언더슈트의 원인을, 봉지(encapsulated) 모듈에서 가정한 $k = 0.99$ 보다 큰 변압기 누설 인덕턴스로 추적했습니다. 또한 간격이 방전 지속 시간, 안정성, 강도 사이의 trade-off 를 만들기 때문에, 5 개 간격에 대해 각 5 회 반복 측정하고 변동계수(CV)로 안정성을 평가했습니다.

검증은 5 회의 독립적인 파형 측정, LTspice 와의 비교, 간격별 5 회 반복 측정, 스펙트럼 확인(약 450 nm 및 626 nm 피크, 질소 발광)으로 구성했습니다. 1.5 mm 간격에서 가장 긴 방전 시간(163.4 s), 가장 높은 안정성(CV 11.3%), 적절한 강도(1.76 a.u., CV 8.7%)를 얻었습니다. 강도는 0.5 mm 에서 최대(2.06 a.u.)였으며, 2.0 mm 이상의 간격에서는 불안정해졌습니다(CV 최대 50.6%). 검토한 비교 대상 시스템 중에서, US\$10 미만의 검증된 비용, 3.7 V 배터리 구동, 전체 출력 특성 평가, 체계적인 간격 연구를 모두 갖춘 시스템은 이것이 유일했습니다.



그림 3(a). 조립된 PenPlasma 프로토타입

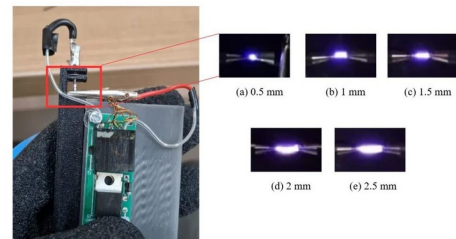


그림 3(b). 0.5–2.5 mm 간격별 플라즈마

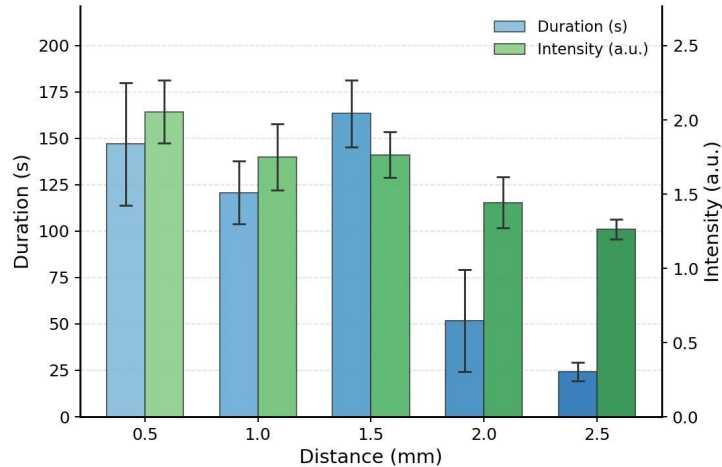


그림 3(c). 간격별 방전 지속 시간과 강도 (n = 5)

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

4. Zynq 오디오 코덱 Bring-Up: AXI IIC 와 Bare-Metal C

개인 프로젝트 · 2026.09

Zybo Z7-10 에 탑재된 SSM2603 오디오 코덱을 설정하는 프로젝트입니다. 코덱이 Zynq 의 FPGA 측에 배선되어 있어 Arm 코어가 사용할 I2C 마스터를 fabric 안에 구성해야 했습니다. 데이터 경로는 Cortex-A9(bare-metal C) → AXI → SmartConnect → FPGA fabric 의 AXI IIC → I2C(핀 N18/N17) → SSM2603(주소 0x1A)입니다.

Vivado 블록 디자인(Zynq PS + AXI IIC + SmartConnect), 핀 제약 조건, 비트스트림과 함께 Vitis 플랫폼 및 bare-metal C 드라이버를 구현했습니다. 코덱의 7-bit 레지스터 주소와 9-bit 데이터를 2 바이트 I2C 쓰기로 패키징하는 helper 를 작성하고, reset, power-up, analog path, digital path, I2S 16-bit 포맷, 48 kHz 샘플링 레이트, activate 의 7 단계 레지스터 시퀀스를 실행했습니다. 새로운 Vitis 플로우(SDT)는 device ID 대신 base address 를 사용하는데, 이전 튜토리얼을 따르지 않고 드라이버 헤더에서 직접 확인했습니다.

각 쓰기 동작마다 115200 baud 의 UART 로그를 확인했습니다. 첫 실행에서 공장 데모가 실행되고 JTAG 연결이 실패했으며, 원인은 부트 모드 점퍼가 JTAG 가 아닌 SD 로 설정되어 있었기 때문입니다. 최종적으로 실제 하드웨어에서 모든 레지스터 쓰기가 코덱으로부터 acknowledge 되었습니다.

```
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
□
```

그림 4. UART 로그: 모든 레지스터 쓰기 acknowledge 확인

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

저장소: github.com/Rennaafi/zybo-i2c-ssm2603-bringup

5. 논리 게이트에서 RISC-V SoC 까지: Zybo Z7-10 의 12 개 Verilog 모듈

KCCI FPGA 교육 · 2026

실제 하드웨어에서 모듈을 하나씩 쌓아 올리며 디지털 설계를 기초 원리부터 구축하는 것이 목표였습니다. 게이트에서 시작하여 카운터와 tick generator, debouncer, FSM, UART 를 거쳐 single-cycle RISC-V 코어와 이를 중심으로 한 소규모 SoC 까지 단계적으로 진행했습니다.

각각 testbench 와 보드 제약 조건(125 MHz 보드 클럭, 버튼, 스위치, LED)을 갖춘 12 개의 parameterized Verilog 모듈을 구현했습니다. 검증은 시뮬레이션에서 PASS/FAIL 을 출력하는 self-checking testbench, Vivado 에서의 implementation, Zybo Z7-10 에서의 테스트 순서로 진행했습니다. 그 결과 기본 논리 회로에서 보드 위에서 동작하는 소규모 RISC-V SoC 까지 이어지는 경로를 완성했습니다.

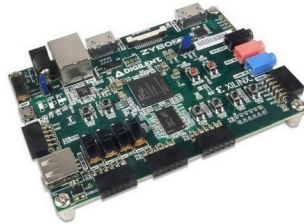


그림 5. Digilent Zybo Z7-10

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

저장소: github.com/Rennaafi/zybo-fpga-coursework

6. 순수 Verilog 로 구현한 UART 기반 HC-SR04 거리 측정기

개인 프로젝트 · 2026

프로세서 없이 FPGA 로직만으로 초음파 센서를 이용해 거리를 측정하는 프로젝트입니다. FSM 이 HC-SR04 의 trigger 펄스를 구동하고 echo 시간을 측정하며, echo 펄스 폭 카운터와 거리 변환 로직이 측정값을 산출하고, UART 송신기가 각 측정값을 PC 의 시리얼 터미널로 전송합니다. 센서는 Pmod JD 와 JE 를 통해 연결했으며, 터미널에서 실시간 거리 측정값을 확인했습니다.



그림 6(a). Pmod JD 및 JE 에 연결한 HC-SR04

```
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 = 13 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
```

그림 6(b). UART 를 통한 실시간 거리 측정값

사용 기술: Verilog, Vivado, Zybo Z7-10, HC-SR04, UART, FSM

저장소: github.com/Rennaafi/zybo-z7-hcsr04-distance-uart

7. KiCad 9 기반 STM32 개발 보드: 2-Layer STM32F103 및 4-Layer STM32G474RE

개인 프로젝트 · STM32G474RE: KCCI PCB 과정, 2026.09 ~

회로도부터 제조 파일까지 소형 MCU 보드를 설계한 두 개의 프로젝트로, 소형 STM32F103 보드와 온보드 ST-LINK 가 없는 Nucleo-64 스타일의 STM32G474RE 보드로 구성됩니다. F103 의 전원 경로는 USB → 3.3 V LDO → STM32F103 이며 16 MHz 크리스탈을 사용합니다. G474 의 경로는 USB-C(5.1 kOhm CC) → 500 mA 폴리퓨즈 → Schottky 다이오드 → AP2112K-3.3 LDO → MCU(100 nF 디커플링)이며, 페라이트 비드를 통해 아날로그 전원을 공급합니다.

F103 은 33.5 x 24.2 mm 의 2-layer 보드로, boot 및 reset 스위치와 헤더로 인출된 SWD, SPI, I2C 를 갖추고 있습니다. G474 는 모든 MCU 핀을 morpho 배열의 female 헤더로 인출하고, 1x6 SWD 헤더, 16 MHz 및 32.768 kHz 크리스탈, LED, 버튼, solder jumper 를 포함하는 회로도를 ST 의 Nucleo MB1367 설계를 참고하여 작성했습니다. G474 의 과제 요구 조건은 4 개 layer, 최대 100 x 100 mm, 0.3/0.45 mm tented via, ST-LINK 미탑재입니다.

F103 은 제작 가능한 Gerber 파일을 완성했습니다. G474 는 2026.09.26 에 회로도 작업을 시작했으며, 제조 파일 제출 기한은 2026.10.04 입니다.

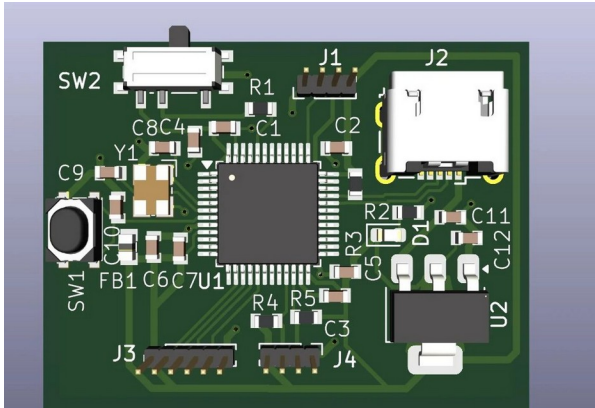


그림 7(a). STM32F103 보드, KiCad 3D 렌더링

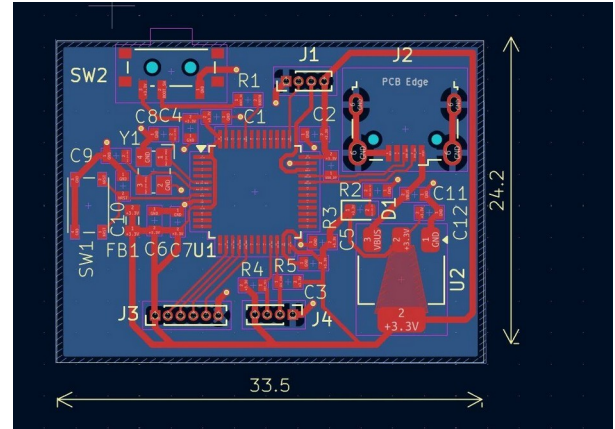


그림 7(b). 2-layer 레이아웃, 33.5 x 24.2 mm

사용 기술: KiCad 9, STM32F103, STM32G474RE, AP2112K-3.3, USB-C, SWD

저장소: github.com/Rennaafi/stm32f103-dev-board

8. ESP32 IMU 실시간 대시보드

개인 프로젝트 · 2026.09

클라우드, 앱, 공유기 없이 어떤 스마트폰이나 노트북에서도 IMU의 자세를 실시간으로 확인하는 프로젝트입니다. MPU6050(I2C)의 데이터를 ESP32의 complementary filter(pitch, roll, heading)로 처리하여 JSON 엔드포인트로 제공하며, 대시보드는 ESP32 자체 Wi-Fi 액세스 포인트를 통해 flash에서 직접 서비스됩니다.

complementary filter 기반 센서 퓨전, 자세 표시기와 나침반, 약 20 Hz로 폴링하는 실시간 가속도계 및 자이로스코프 차트를 갖춘 정적 대시보드, 그리고 I2C 버스 자동 복구 기능을 구현했습니다. Wi-Fi 송신 시 발생하는 전류 스파이크가 센서의 I2C 버스를 멈추게 할 수 있어, 펌웨어가 이를 감지하고 스스로 버스를 복구하도록 했습니다. 그 결과 독립적으로 동작하는 실시간 대시보드를 완성했으며, 이는 이후 모션 분류기의 프론트엔드이자 데이터 로거로 활용되었습니다.



그림 8. ESP32 flash에서 서비스되는 대시보드

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

저장소: github.com/Rennaafi/esp32-mpu6050-dashboard

9. PrioWatt: 우선순위 부하 차단 기능을 갖춘 스마트 kWh 미터 및 Despro 앱

캡스톤, 4인 팀 · 2025.08 ~ 2025.12

PrioWatt는 중요도가 가장 낮은 부하부터 차단하여 회로를 보호하고, 비정상적인 전력 소비가 발생하면 사용자에게 경고하는 에너지 모니터입니다. 센서 데이터를 ESP32가 수집(6개 파라미터, 3초마다)하고, 우선순위 부하 차단 로직으로 4개의 부하를 제어하며, Firebase(5초마다)를 거쳐 Despro Flutter 앱으로 전달합니다.

팀 차원에서 ESP32를 이용한 전압, 전류, 유효 전력, 에너지, 역률, 주파수 측정, 원격 override가 가능한 4개 부하 대상 단계별 차단, Despro의 소켓별 제어 및 분석 기능을 구현했습니다. 저는 앱의 AI 팁을 뒷받침하는 LSTM 모델(비용 예측 및 이상 탐지)을 담당했고, 펌웨어, 로깅, 앱 연동 작업에도 참여했습니다.

검증은 샘플링 및 로깅 주기, 부하 차단 동작, Wi-Fi 안정성을 포함하여 전체 시스템에 대한 bench test로 진행했습니다. 3초 샘플링과 5초 로깅이 Wi-Fi 재연결 실패 없이 유지되었으며, 14주 만에 설계 전시회에 맞춰 완성했습니다.



그림 9(a). 설계 전시회에서의 팀과 하드웨어

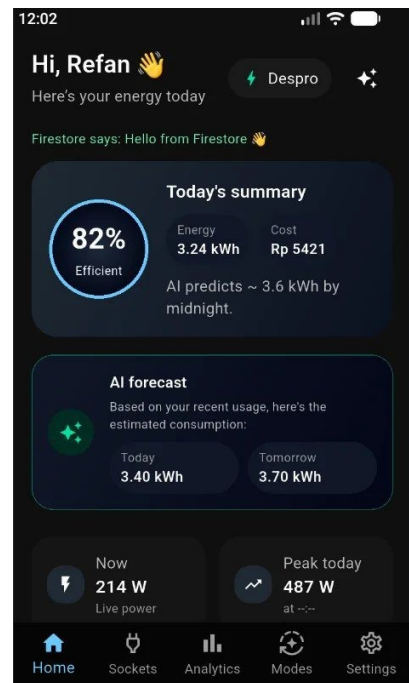


그림 9(b). Despro 앱: 실시간 전력 및 AI 예측

사용 기술: ESP32, C/C++, Firebase, Flutter/Dart, Python, LSTM

저장소: github.com/Rennaafi/despro-smart-socket-monitor

10. FOTRIX: 태양광 기반 해수 수소 생산 및 미세플라스틱 분해 프로토타입

5인 팀 · 2025.01 ~ 2025.11 · PIMNAS 2025 전국 결선 진출 (PKM-KC)

FOTRIX는 해수에서 수소를 생산하는 동시에 미세플라스틱을 분해하는 휴대용 태양광 구동 장치입니다. 수소 생산량, 분해 지표, PV 전력을 측정하는 센서의 데이터를 마이크로컨트롤러가 Firebase로 전송하고, FOTRIX Flutter 앱에서 실시간 및 과거 수소 데이터를 확인하며 액추에이터와 태양광 트래커를 제어합니다.

저는 센서 및 텔레메트리 아키텍처, Firebase 백엔드를 갖춘 Flutter 앱, 하드웨어-소프트웨어 통합과 시스템 수준 디버깅을 담당했습니다. 검증은 실험실 통합 → 결선 전 2주간의 현장 테스트(동작, 이상 현상, 개선 조치 기록) → 전국 현장 평가 순으로 진행했습니다. 프로토타입은 전국 대회의 현장 시연을 끝까지 수행했으며, 팀은 PIMNAS 2025 전국 결선 진출팀이 되었습니다.

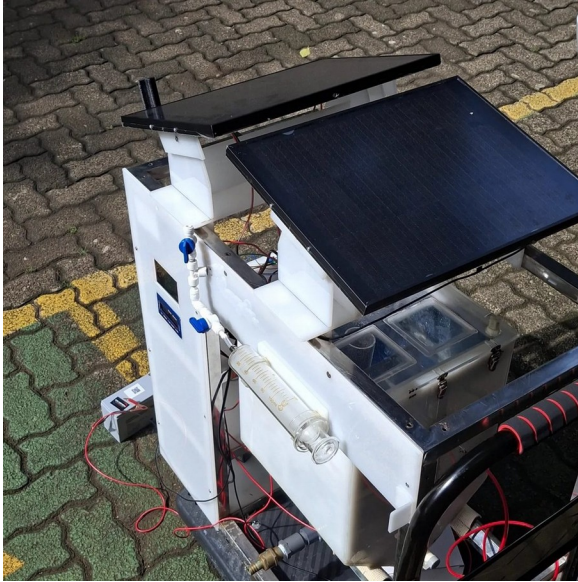


그림 10(a). 반응기: 태양광 패널, 제어 박스, 주입용 시린지

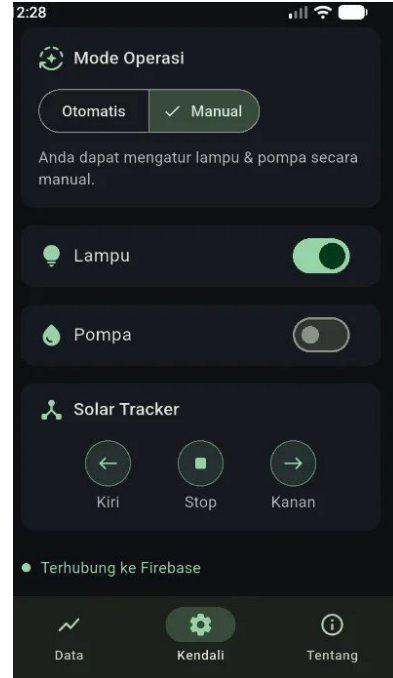


그림 10(b). FOTRIX 앱: 액추에이터 및 태양광 트래커 제어

사용 기술: Sensors, microcontroller telemetry, PV monitoring, Flutter/Dart, Firebase

저장소: github.com/Rennaafi/fotrix-hydrogen-monitor

11. Pt100 RTD 온도 측정 체인

개인 프로젝트 · 2025.08 ~ 2025.12

MCU 내장 ADC 보다 안정적으로 온도를 측정하는 것을 목표로 한 프로젝트입니다. 체인은 Pt100 RTD → Wheatstone 브리지 → ADS1115 16-bit ADC(I2C) → Arduino 순서로 구성되며, Arduino 에서 Callendar–Van Dusen 변환을 통해 온도를 산출합니다. 브리지 신호 조절 회로, ADC 인터페이스, 저항-온도 변환, 그리고 정확도와 반복성을 문서화하는 보정(calibration) 워크플로우를 구현했습니다. 그 결과 MCU ADC 를 직접 사용하는 방식보다 전압 변동이 낮았습니다(정성적 비교).

사용 기술: Pt100, Wheatstone bridge, ADS1115, I2C, Arduino, calibration

12. 서보 스캔 방식 열화상 어레이 기반 재실 감지

교과목 프로젝트, 5 인 그룹 · 2025.08 ~ 2025.12

카메라 없이 저해상도 열화상 센서로 공간을 스캔하여 사람의 존재를 감지하는 프로젝트입니다. AMG8833 8x8 열화상 어레이(I2C)를 서보 스캔(0°, 90°, 180°, 약 500 ms 주기, PWM)으로 움직이고, ESP32 가 threshold 기반 감지를 수행하여 Wi-Fi 로 결과를 전송합니다.

I2C 데이터 수집, PWM 서보 스캔, Wi-Fi 전송, 절대/차분 threshold 감지를 위한 ESP32 펌웨어를 작성했습니다. 전체 시스템에 대한 연속 동작 테스트에서 120 분간 안정적으로 동작했으며, 프로젝트 보고서 작성에 공동 저자로 참여했습니다.

사용 기술: ESP32, AMG8833, I2C, PWM servo, Wi-Fi

13. 음성 명령 램프

3인 팀

스마트폰 앱의 음성 명령으로 램프를 켜고 끄는 프로젝트입니다. 음성 명령이 컴패니언 앱으로 전달되고, BLE를 통해 ESP32에 전송되며, ESP32가 릴레이와 램프를 제어합니다. 그 결과 BLE로 컴패니언 앱과 연결된 ESP32 기반 음성 제어 조명 프로토타입을 완성했습니다.



그림 11. 팀, 프로토타입 및 컴패니언 앱

사용 기술: ESP32, BLE, relay, mobile app

14. RPLIDAR A1과 Zybo Z7-10 연동

개인 프로젝트 · 2026.09 ~

로봇 프로젝트의 매핑을 위한 기반 작업으로, 360° 라이다를 Zybo에서 직접 읽는 프로젝트입니다. RPLIDAR A1(UART)을 FPGA fabric의 AXI UART Lite에 연결하고, Zynq ARM 코어의 bare-metal C 파서가 포인트를 추출하며, 모터 PWM 블록은 AXI GPIO를 통해 구동합니다.

Python 호스트 도구(테스트, 뷰어, 가상 라이다, 셀프 테스트), AXI UART Lite, AXI GPIO 및 직접 설계한 모터 PWM 블록을 포함한 Vivado 디자인, 그리고 sync 검출, 비차단(non-blocking) 콘솔 ring buffer, watchdog을 갖춘 bare-metal C 파서를 구현했습니다.

모터 PWM xsim testbench를 통과했고, implementation은 WNS +2.394 ns로 timing을 만족했습니다. 라이다는 USB로 검증(6.5 Hz에서 약 290 포인트)했고, Zybo UART는 loopback으로 검증했습니다. 또한 global timer가 첫 번째 sleep 호출 내부에서만 시작되던 timer 버그를 수정했습니다. 라이다와 Zybo는 각각 단독으로 검증을 마쳤으며, 다음 단계는 커넥터 핀 매핑을 확인한 후 라이다와 Zybo를 직접 연결하는 것입니다.

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

15. Recall: AI 카드 생성 기능을 갖춘 간격 반복 학습 웹 앱

개인 프로젝트 · 2026.09 · 배포 완료 및 매일 사용 중

FPGA, STM32, 물리 설계 학습을 위해 만든 개인용 플래시카드 앱으로, 긴 transcript 에서 카드를 자동 생성합니다. transcript 는 클라이언트에서 정제하여 약 1,400 단어 단위로 분할하고, Workers AI(Llama 3.3 70B, JSON-schema 출력)가 카드를 생성한 뒤, 중복을 제거하고 검토 및 편집을 거쳐 D1 데이터베이스에 저장합니다.

Cloudflare Access 뒤에 위치한 D1 데이터베이스와 Cloudflare Workers 기반의 React/TypeScript 프론트엔드, 그리고 병렬 요청과 사용량 추정 기능을 갖춘 AI 생성 기능을 구현했습니다. 무료 D1 요금제는 요청당 쿼리 수를 제한하지만 로컬 개발 환경에서는 이 제한이 적용되지 않기 때문에, 모든 bulk write 는 JSON 배열을 입력으로 하는 하나의 구문으로 처리합니다. 앱은 배포되어 매일 사용 중이며, 약 2,000 단어의 transcript 는 약 13 초가 걸리고 하루 무료 AI 유닛 10,000 개 중 약 240 개를 사용합니다.

사용 기술: React, TypeScript, Cloudflare Workers, D1, Workers AI, Cloudflare Access

16. Python 기반 OpenUSD: 스크립트로 장면과 어셈블리 작성

자기 주도 학습 프로젝트 · 2026.10 ~

NVIDIA Omniverse 와 같은 디지털 트윈 및 시뮬레이션 도구는 OpenUSD 를 기반으로 합니다. 더 큰 플랫폼을 사용하기 전에 단위, reference, 머티리얼을 이해하기 위해 포맷 자체부터 시작하여, Python 의 usd-core 로 장면을 작성하고 각 파일을 Blender 에서 확인했습니다. 별도의 GPU 없이 일반 노트북에서 실행됩니다.

총 여섯 개의 스크립트를 작성했습니다. 큐브와 구가 있는 stage(prim, attribute, transform), 원시 point, face-count, index 배열로 만든 mesh, STL-to-USD 변환기, 변환된 파트에 바인딩한 UsdPreviewSurface 머티리얼, 하나의 파트를 세 번 reference 하고 그중 하나에 scale override 를 적용한 어셈블리, 그리고 time sample 을 이용한 bounce-and-sway 애니메이션입니다.

STL 파일에는 단위 정보가 없어, 115 mm 파트가 115 m 로 import 되었고 포인트를 0.001 배로 스케일링하여 해결했습니다. 이후 모든 import 에서 Blender 의 Dimensions 패널을 확인합니다. stage 의 up-axis 와 meters-per-unit 을 파일 메타데이터로 설정하여 Blender 에서 회전이나 스케일 보정 없이 import 됩니다. 어셈블리에서 reference 는 복사본이 아닌 링크이므로, 어셈블리 파일에는 mesh 데이터가 없고 파트 파일을 수정하면 모든 인스턴스에 반영됩니다.

생성한 모든 .usda 파일을 Blender(Material Preview)에서 열어 스케일, 방향, 색상, 애니메이션을 확인했습니다. prim, mesh, STL 변환, reference, 머티리얼, 애니메이션 모두 정상 동작하며 Blender 에서 검증했습니다. 다음 단계는 간단한 로봇을 위한 UsdPhysics joint, 그리고 URDF-to-USD 변환과 Isaac Sim 에서의 시뮬레이션입니다.

사용 기술: Python, usd-core (OpenUSD), trimesh, Blender

지원 직무와의 연계

위 프로젝트들은 하드웨어 스택을 관통하는 하나의 흐름을 따릅니다. 회로와 PCB(STM32 보드, PenPlasma 발생기), 칩을 구동시키는 펌웨어(ESP32, bare-metal Zynq), FPGA fabric 의 로직(RISC-V SoC 에 이르는 Verilog 모듈, hls4ml 가속기), 그리고 실제 CPU 의 물리적 레이아웃(OpenROAD, ibex)입니다. 이 모든 과정의 중심에는 측정이 있으며, sweep, 안정성 지표, signoff 리포트를 분석한 뒤에 의사결정을 내렸습니다.

임베디드 및 MCU 직무에서는 코덱 bring-up, IMU 대시보드, PrioWatt 가 하드웨어에서 검증된 펌웨어 역량을 보여줍니다. FPGA 및 SoC 직무에서는 모션 분류기와 Verilog 실습 과제가 RTL 및 버스 통합 역량을 보여줍니다. PCB 및 하드웨어 직무에서는 STM32 보드와 PenPlasma 가 회로 설계와 검증 역량을 보여줍니다. 테스트 및 품질 직무에서는 PenPlasma 와 RTD 체인이 체계적인 측정과 문서화 역량을 보여줍니다.