OPEN HARDWARE · OPEN FIRMWARE · OPEN SOFTWARE

Build the hardware. Write the firmware. Ship the app.

RC02 Open-Scope: a real, working board for the next generation of makers

PIC32 · USB oscilloscope · signal generator · I²C lab · Android + Desktop apps

WHY THIS BOARD EXISTS

Learn embedded without starting from zero

- Most kits teach blinking an LED. Real products need hardware, firmware, protocols and apps working together.
- This board is a complete, working system you can read end to end.
- Everything is open: schematic, Gerbers, firmware, library, desktop and mobile apps.
- Break it, fix it, remix it, then design your own board and reuse the same software.

For young engineers

A platform to try things, with working pieces to learn from.

For tinkerers

Plug in an I²C sensor and write your own driver.

For future designers

Spin off your own board; same apps still work.

WHAT IT IS

One board, a whole lab

2 ch · 12-bit scope

MCP3202 SPI ADC, 0 to 5 V, 200 Hz to 500 kHz

2 ch · 12-bit signal gen

MCP4922 SPI DAC, 8 waveforms, timer + DMA

2 × I²C master

Scan, read, write; drivers for 10 sensor families

GPIO · capture · UART

Port E, input capture, PWM pins, Wiegand out

USB serial

Plug and play at 115200, same protocol everywhere

PIC32MX @ 80 MHz

FreeRTOS, USB bootloader, 512 KB flash

END TO END

Follow one signal through the whole stack



A 64-byte framed packet (SOH · escaped body · CRC-16 · EOT) carries every command over USB.

Every layer is open. You can enter at any layer and ignore the rest.

EVERYTHING IS OPEN

Four things you can open, read and change

Hardware

Altium schematic + PCB, Gerbers, annotated pinouts

Firmware

PIC32 + FreeRTOS + USB CDC + AN1388 bootloader

Library

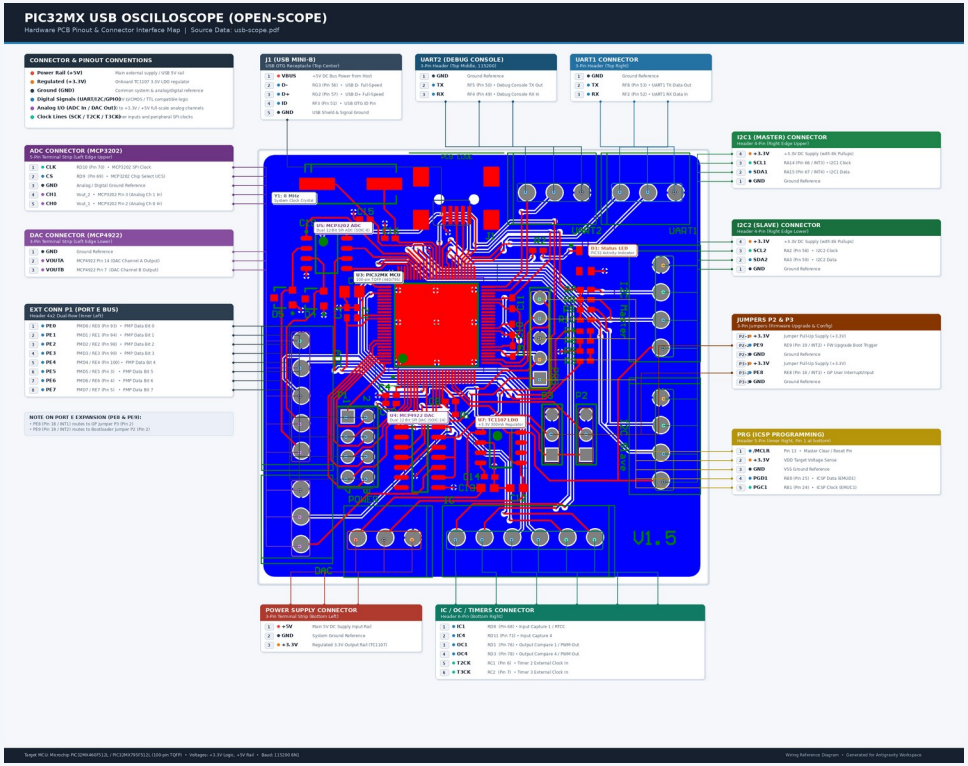
openlib: C++17 and C API, 10 I²C device drivers

Apps

Qt5/Qwt desktop · Flutter Android · Windows/Linux/Android

Cross-compiles across Windows, Linux and Android.

Meet the board



- Every connector documented: pins, ports, MCU pin numbers
- 3.3 V logic; 0 to 5 V analog in and out
- ICSP header + USB HID bootloader
- Daughter-board connectors: P1, P2, P3

Full-resolution annotated pinout: docs/assets/

TRACKS

You don't have to learn everything. Pick a path.

Hardware

Read the schematic, probe the board, design your own.

`docs/02-hardware`

Firmware

Embedded C: SPI, I²C, DMA, timers, FreeRTOS, USB.

`docs/03-firmware`

openlib (C++/C)

Write your own sensor library on a clean API.

`docs/04-openlib`

Desktop (Qt)

Plotting, GUIs, instrument software.

`docs/05-desktop-app`

Android (Flutter)

Real-time plots on a phone over USB-OTG.

`docs/06-android-app`

Labs

Guided experiments from first light to your own driver.

LEARN IT BY READING IT

Track: Hardware

- Trace a signal from the ADC header to the MCU's SPI pins
- Understand power: 5 V in, 3.3 V LDO, decoupling
- Bring-up checklist: shorts, rails, crystal, USB
- Remix: daughter boards, new front-ends, new layouts

```
Key parts
  MCU   PIC32MX 100-pin
  ADC   MCP3202 (SPI1)
  DAC   MCP4922 (SPI2)
  LDO   TC1107 3.3 V
  XTAL  8 MHz -> 80 MHz
```

Open design: Altium + Gerbers

EMBEDDED C

Track: Firmware inside the PIC32

What you will touch

SPI1 + Timer5 ISR scope sampling

SPI2 + Timer4 + DMA1 waveform output

I²C master, input capture, UART, USB CDC

Real-time OS

FreeRTOS on MIPS M4K at 80 MHz

USB HID bootloader (AN1388) for easy updates

```
typedef struct MED_pkt {
  uint8_t cmd;
  uint8_t subcmd;
  uint8_t incnum;
  uint8_t used_size;
  union { uint8_t uc_p[60];
          uint16_t us_p[30];
          float fl_p[15]; };
} MED_pkt;   // 64 bytes
```

Track: openlib. Your gateway to the board

```
openlib::ProtocolClient c("/dev/ttyACM0");
if (!c.connect()) c.setSimulationMode(true);

openlib::I2CBus i2c(c, 1);   // I2C2
i2c.open();
for (auto a : i2c.scanBus(0x08, 0x77))
    printf("found 0x%02X\n", a);

openlib::SignalGenerator g(c);
g.generateSine(1000, 0.5f, 4.5f);
```

- Scope, signal gen, capture
- I²C, GPIO, daughter board
- 10 ready sensor drivers
- --sim mode: no board needed

C++ GUI

Track: Desktop studio (Qt5 + Qwt)

Oscilloscope + generator

Dual channel, triggers, CSV, Vpp/Vrms/freq/duty

Logic analyzer

Pulse timing on IC and Port E headers

I²C sensor studio

Scan, dump registers, ready drivers

GPIO, CLI, bootloader

Console, Wiegand, firmware update

Learn: Qt widgets, Qwt plotting, wrapping a library in a GUI.

MOBILE

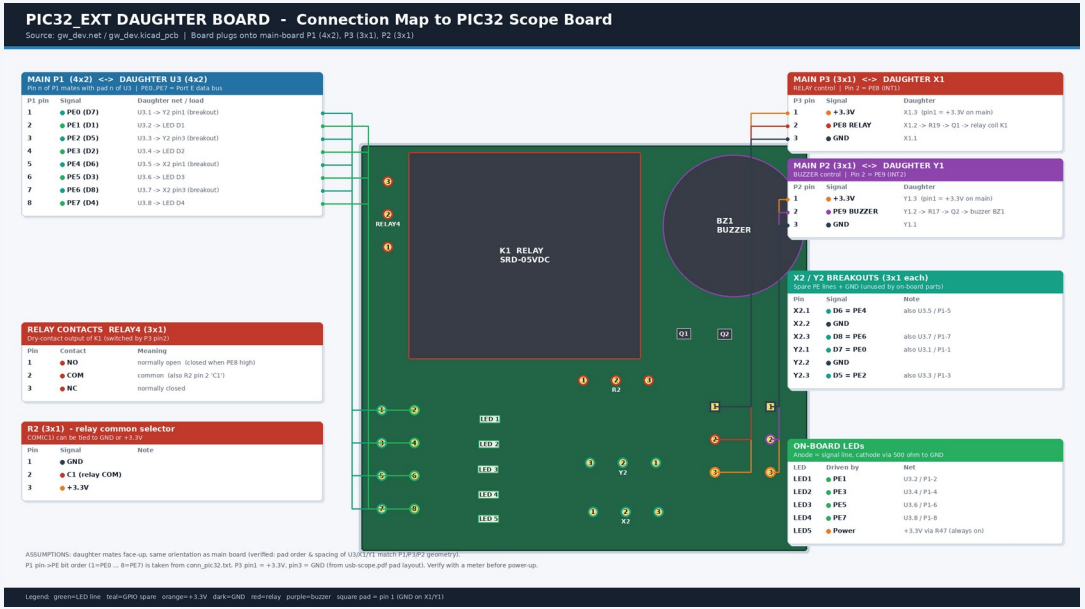
Track: Android oscilloscope (Flutter)

- Real-time dual-channel plot on your phone
- USB-OTG CDC-ACM, serial or simulation
- Same 64-byte protocol as desktop
- CustomPainter drawing, Kotlin USB bridge

```
lib/
  main.dart
  protocol/
    pic32_protocol.dart
  transport/
    serial_transport.dart
  ui/
    oscilloscope_painter.dart
```

No hardware knowledge needed to start.

First outputs: LEDs, a relay and a buzzer



- LEDs D1 to D4 on PE1, PE3, PE5, PE7
- Relay K1 on PE8, buzzer BZ1 on PE9
- MPU-9050 IMU on the I2C master header (I2C2)
- One example tests it all: test_imu_daughter

```
./bin/test_imu_daughter
./bin/test_imu_daughter --sim
```

THE MAIN GOAL

Your turn: write a sensor library

1. **Wire it** — 3.3 V, GND, SCL, SDA
2. **Scan** — find its address
3. **Read ID** — WHO_AM_I = 0x68
4. **Configure** — wake, set ranges
5. **Burst read** — 14 bytes at 0x3B
6. **Convert** — counts to g, dps, °C

```
bus.writeReg8(addr, 0x6B, 0x00);            // wake the sensor
bus.readRegBlock(addr, 0x3B, buf, 14);      // accel, temp, gyro
float g = (int16_t)((buf[0] << 8) | buf[1]) / 16384.0f;   // +/-2 g range
```

Learn by doing: four labs

Lab 1

First light

Build, simulate, connect

Lab 2

Scope meets generator

DAC to ADC loopback, aliasing

Lab 3

IMU + daughter board

Real sensor, LEDs, relay, buzzer

Lab 4

Your own library

Datasheet to driver

Every lab has expected output, questions and a challenge.

DESIGN YOUR OWN BOARD

Then make it yours

1. **Daughter boards**
   Sensors, motors, displays on P1/P2/P3 or I²C
2. **New layout**
   Same MCU; firmware and apps unchanged
3. **New analog front-end**
   Attenuators, buffers, wider ranges
4. **New MCU**
   Port the firmware; same protocol, same apps

Bring-up checklist

No shorts on rails
3.3 V LDO output
Bootloader via ICSP
USB enumerates
Load app firmware
scan_i2c, test_scope

Your first board, your first driver, your first app.

Get started in 15 minutes

```
cd openlib && make -j4
./bin/test_imu_daughter --sim
# docs/README.md  ->  pick a track
```

Open hardware. Open firmware.
Open tools.
Open AI tools can help fill the gaps.