Hardware in the Loop Testing

https://www.octoprobe.org/octoprobe/big_picture.html 2025-11-11_Octoprobe.pdf

Octo probe

- Motivation Hans
- Dive into specific topics:
  - Micropython, demo on real hardware
  - USB pyudev
  - USB 2/3 dual bus
  - USB hub with power control
- Octoprobe: testbed_micropython
- Octoprobe: tentacle
- Octoprobe: How to use in your project

2025-11-11, Hans Märki

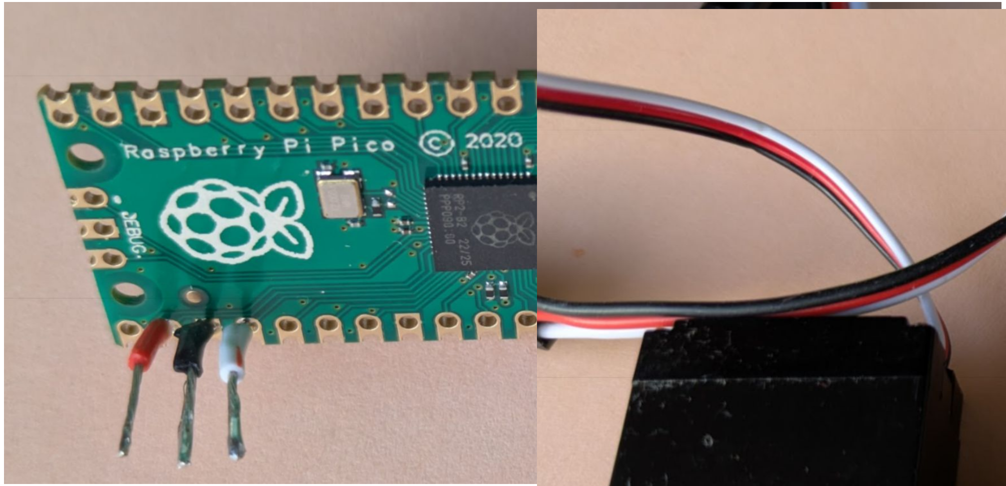
Motivation

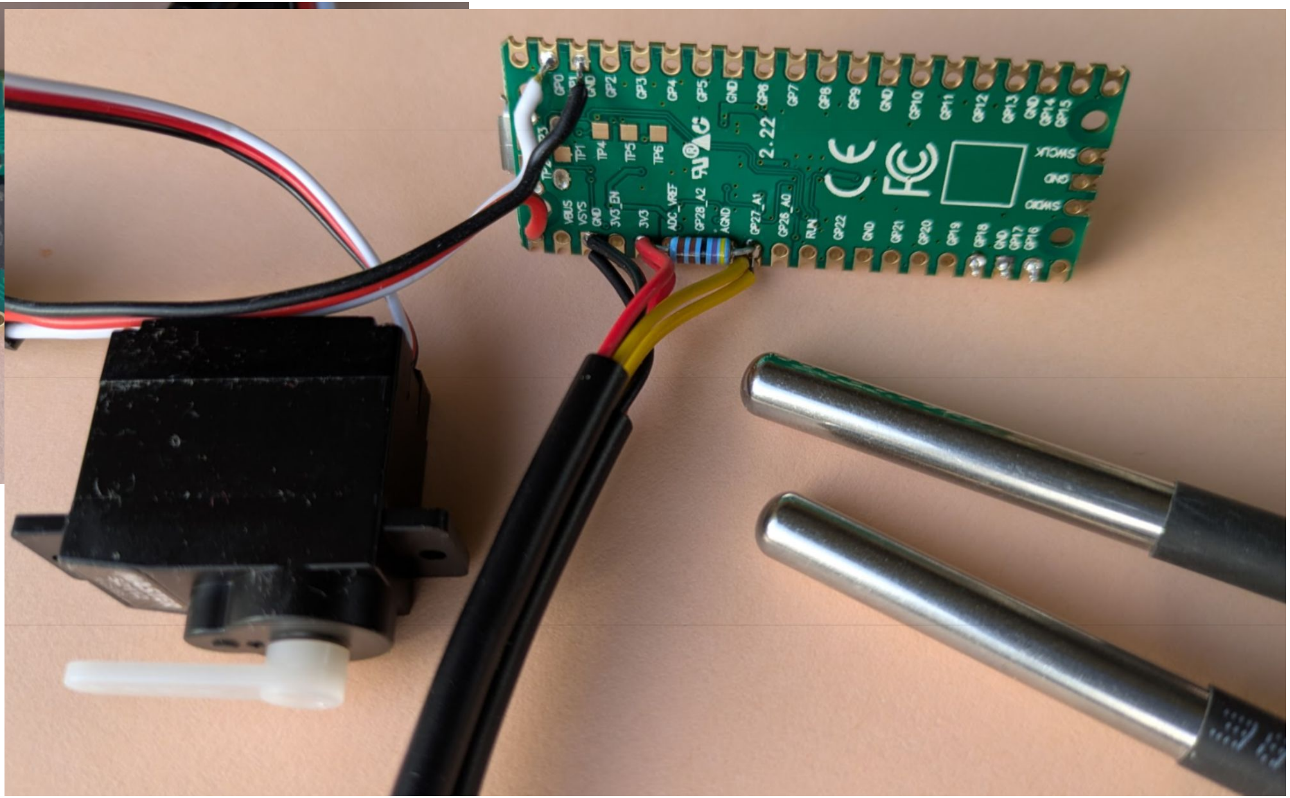
Octo probe

- Best practices konsolidieren
- Octoprobe, Tentacle
- Open Source / Hardware
- MIT Lizenz

Micropython - Demo Board

https://github.com/hmaerki/demo_micropython





USB: polling vs pyudev

Octo probe

Situation to solve

- SW: power on DUT (Device under test)
- DUT takes 0.001s to 6s to appear
- SW: When expected USB device appears: start test

Polling

- Easy to implement

Polling

- System load
- Tuning of intervals, timeout
- Race conditions

pyudev

- asynchronous
- no delay

pyudev

- a bit tricky to program

USB: pyudev

Octo probe

Situation to solve

- SW: power on DUT (Device under test)
- DUT takes 0.001s to 6s to ap
- SW: When expected USB de
  appears: start test

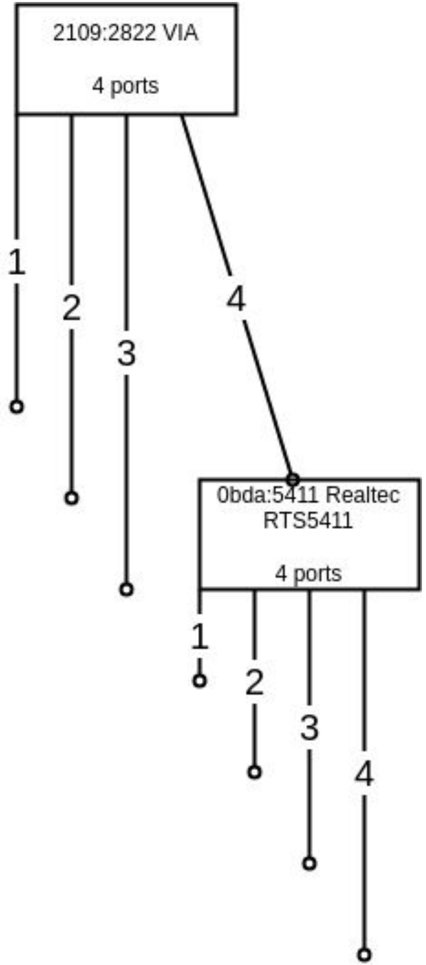
guard before power!
Prevent race condition

```
def power_up():
    with udev.guard as guard:
        tentacle.power.dut = True

        udev_filter = UdevFilter(
            usb_location="3-7.4.4",
            subsystem="tty",
            actions=["add"],
        )

        event = guard.expect_event(
            udev_filter=udev_filter,
            text_expect="Expect ESP32 to become visible",
            timeout_s=6.0,
        )

    return event.tty  # Example: /dev/ttyACM0
```

throws exception if > 6s:

```
Tentacle 2d2a: Expect ESP32 to
become visible within 6s
```

USB 2/3: HUB



Demo: USB 2/3 Hub

2025-11-11, Hans Märki

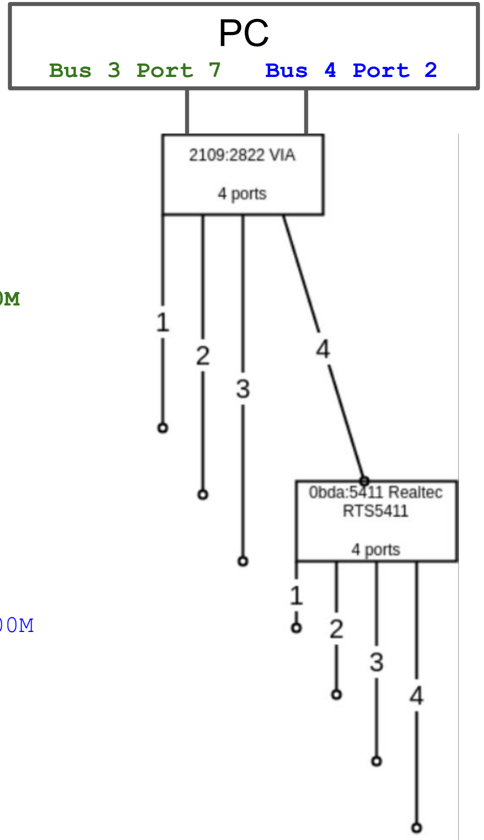
Demo: USB 2/3 Hub

USB2 (HighSpeed)

```
dmesg --follow
usb 3-7.4.4: new high-speed USB device number 37 using xhci_hcd

lsusb -t
/:  Bus 003.Port 001: Dev 001, Class=root_hub, Driver=xhci_hcd/12p, 480M
    |__ Port 007: Dev 030, If 0, Class=Hub, Driver=hub/4p, 480M
        |__ Port 004: Dev 031, If 0, Class=Hub, Driver=hub/4p, 480M
            |__ Port 004: Dev 037, If 0, Class=Mass Storage, Driver=usb-storage, 480M
```

USB3 (SuperSpeed)

```
dmesg --follow
usb 4-2.4.4: new SuperSpeed USB device number 13 using xhci_hcd

lsusb -t
/:  Bus 004.Port 001: Dev 001, Class=root_hub, Driver=xhci_hcd/4p, 10000M
    |__ Port 002: Dev 002, If 0, Class=Hub, Driver=hub/4p, 5000M
        |__ Port 004: Dev 003, If 0, Class=Hub, Driver=hub/4p, 5000M
            |__ Port 004: Dev 013, If 0, Class=Mass Storage, Driver=usb-storage, 5000M

lsusb
Bus 003 Device 031: ID 2109:2822 VIA Labs, Inc. USB2.0 Hub
Bus 004 Device 003: ID 2109:0822 VIA Labs, Inc. USB3.1 Hub
Bus 003 Device 037: ID 0bda:0411 Realtek Semiconductor Corp. Hub
Bus 004 Device 013: ID 0bda:5411 Realtek Semiconductor Corp. RTS5411 Hub
```



USB3: USB2 compatibility

Octo probe

USB 3.2 Specification
https://www.usb.org/document-library/usb-32-reision-11-june-2022

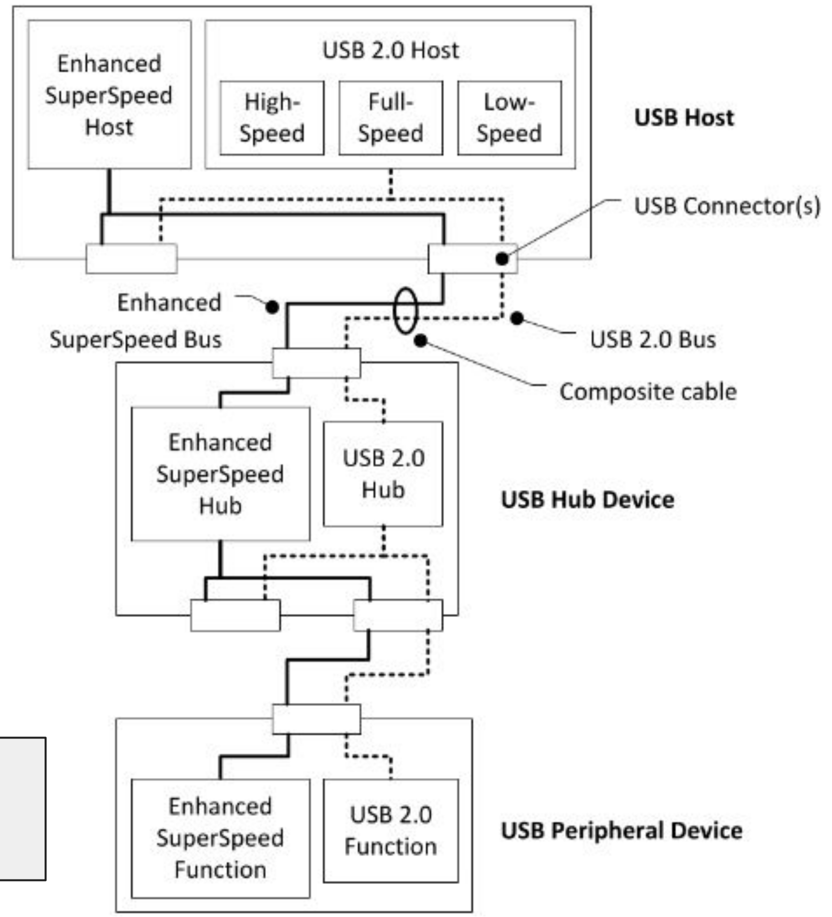
3 Architectural Overview

This chapter presents an overview of Universal Serial Bus 3.2 architecture and key concepts. USB 3.2 is similar to earlier versions of USB in that it is a cable bus supporting data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share bandwidth through a host-scheduled protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

USB 3.2 is a dual-bus architecture that provides backward compatibility with USB 2.0. One bus is a USB 2.0 bus (see *Universal Serial Bus Specification, Revision 2.0*) and the other is an Enhanced SuperSpeed bus (see Section 3.1). USB 3.2 specifically adds dual-lane support.

USB3: Dual Bus

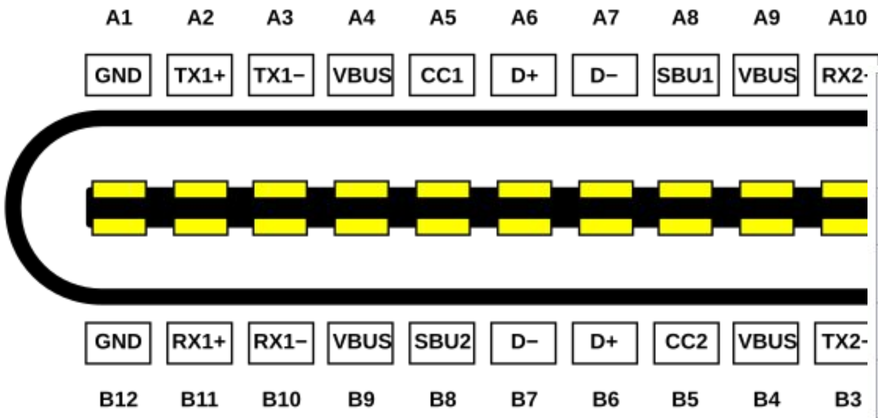
Figure 3-1. USB 3.2 Dual Bus System Architecture



USB 3.2 Specification
https://www.usb.org/document-library/usb-32-reision-11-june-2022

https://en.wikipedia.org/wiki/USB-C

USB C Connector



Pin	Name	Description
A1	GND	Ground return
A2	SSTXp1 ("TX1+")	SuperSpeed differential pair #1, transmit, positive
A3	SSTXn1 ("TX1−")	SuperSpeed differential pair #1, transmit, negative
A4	V_{BUS}	Bus power
A5	CC1	Configuration channel
A6	D+	USB 2.0 differential pair, position 1, positive
A7	D−	USB 2.0 differential pair, position 1, negative
A8	SBU1	Sideband use (SBU)
A9	V_{BUS}	Bus power
A10	SSRXn2 ("RX2−")	SuperSpeed differential pair #4, receive, negative
A11	SSRXp2 ("RX2+")	SuperSpeed differential pair #4, receive, positive
A12	GND	Ground return

Dual bus !

USB 2 Hub: Power Control

USB 2 Specification
https://www.usb.org/document-library/usb-20-specification

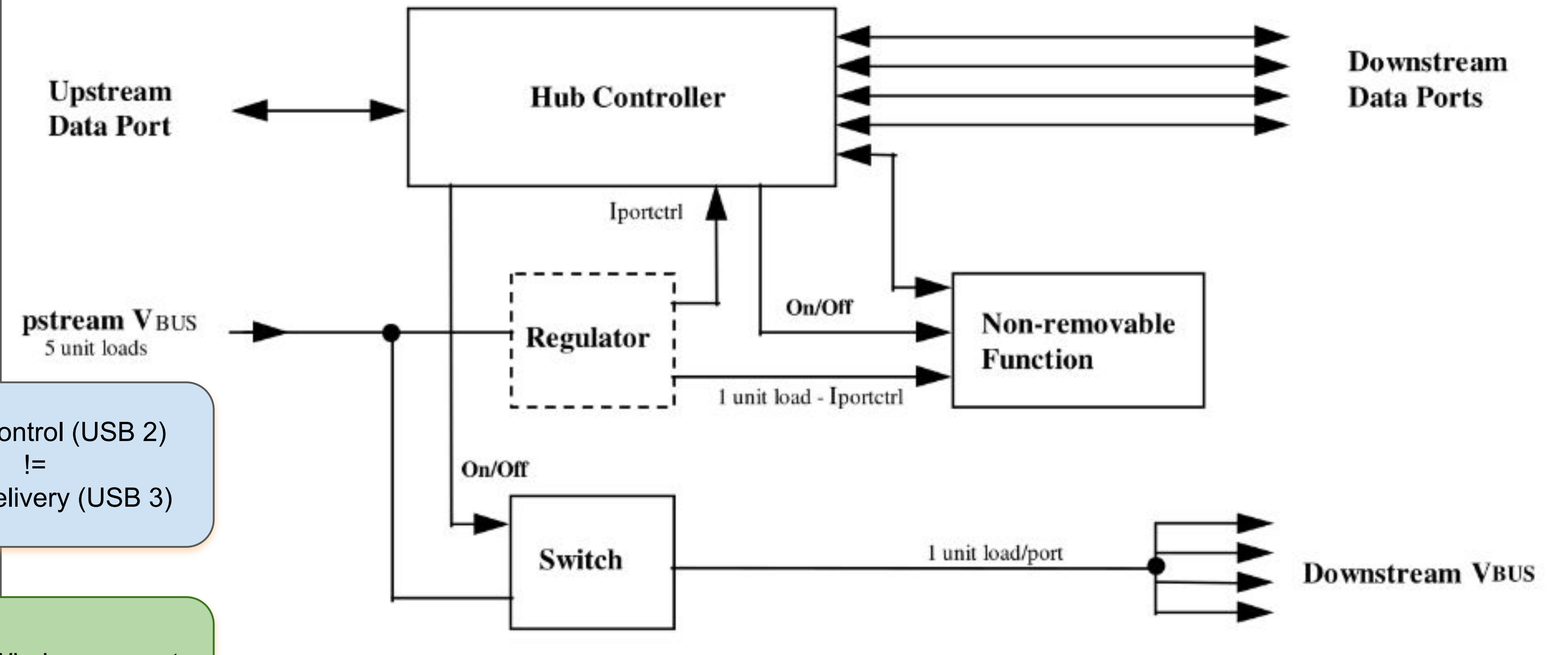


Figure 7-42. Compound Bus-powered Hub

power control (USB 2)
!=
power delivery (USB 3)

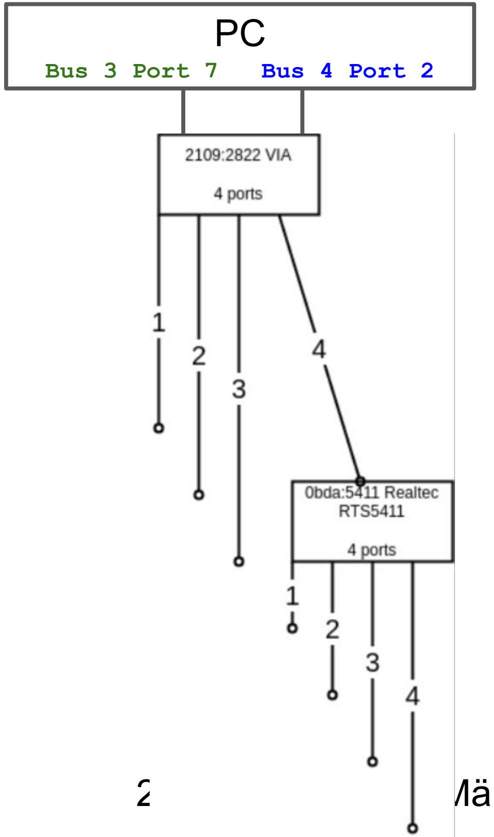
Linux & Window support power control!

USB2: Demo Linux/uhubctl

```
sudo uhubctl --level 4
Current status for hub 4-2.4 [0bda:0411 Generic USB3.2 Hub, USB 3.20, 4 ports, ppps]
  Port 1: 02a0 power 5gbps Rx.Detect
  Port 2: 02a0 power 5gbps Rx.Detect
  Port 3: 02a0 power 5gbps Rx.Detect
  Port 4: 02a0 power 5gbps Rx.Detect
Current status for hub 3-5.3.4 [0bda:5411 Generic USB2.1 Hub, USB 2.10, 4 ports, ppps]
  Port 1: 0100 power
  Port 2: 0100 power
  Port 3: 0100 power
  Port 4: 0100 power
```

Power on/off on 3-3.7.4 port 4

```
sudo uhubctl --action=on --location=3-7.4 --port=4
```



Hub Power Control: Assessment

Octo probe

Hubs with power control

- very useful feature for automation
- hubs are cheap
- hubs are widely available

Downsides

- USB location (3-7.4.4) is not intuitive
- Plugs are NOT numbered
- LEDS are not clearly visible
- Manual on/off overrides USB on/off
- Manual on/off is nice for office use but definitely unwanted in production environments

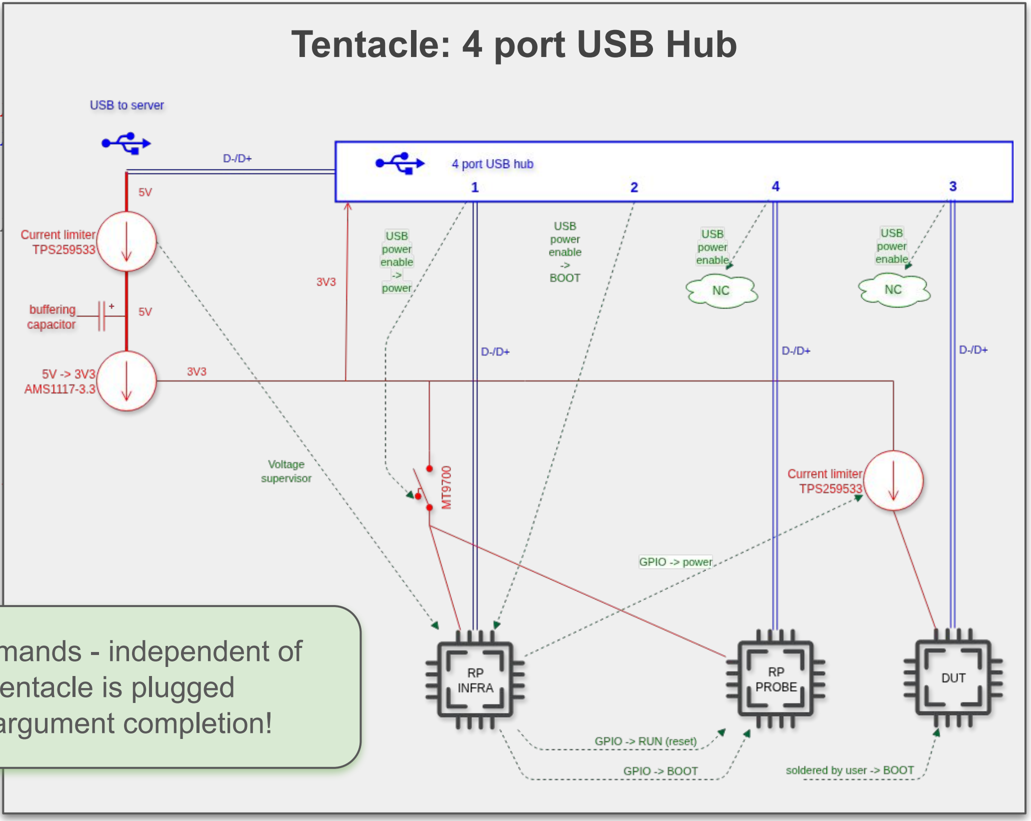
Conclusion

- Commercial hubs are useless IMHO
- But we can do better!

Octoprobe: Tentacle

Two tentacles are connected

```
op query
Tentacle e46340474b55-1722 v0.3 on USB 3-7.4.4 /dev/tty
Tentacle e46414481310-2d2a v0.3 on USB 3-7.4.3 /dev/tty

op power --on dut --serials=2d2a
Tentacle e46414481310-2d2a v0.3 on USB 3-7.4.3 /dev/tty
  +dut


mptest list-tentacles

Connected
  Tentacle 1722-RPI_PICO
      infra: 3-7.4.4.1 /dev/ttyACM0
      dut:   3-7.4.4.3 /dev/ttyACM2 - Board in FS mode
    variants=RPI_PICO
    futs=FUT_MCU_ONLY,FUT_EXTMOD_HARDWARE
  Tentacle 2d2a-ESP32_DEVKIT
      infra: 3-7.4.3.1 /dev/ttyACM1
      dut:   3-7.4.3.3 /dev/ttyUSB0 - CP2
    variants=ESP32_GENERIC
    futs=FUT_MCU_ONLY,FUT_EXTMOD_HARDWARE
```



- Same commands - independent of where the tentacle is plugged
- command argument completion!

Hub on tentacle - Conclusion

Octo probe

- Very convenient for developer!
- Reduces overall complexity
- Robust
- Plug&play on tentacle level

Downsides

- Maximum hub limit
- USB 2 only
- Required HW ~ USD3

Octoprobe

2025-11-11, Hans Märki

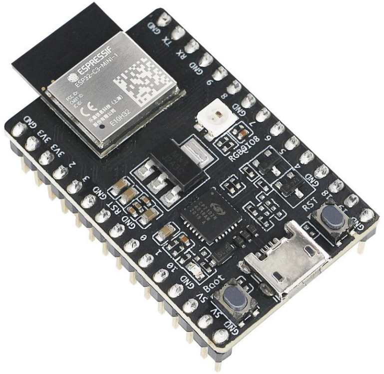
Octo probe

Micropython

Micropython
192 supported boards
36 cpu

Micropython git repo
Python implementation
Ports
Tests

Maintainers role
PR 17946
reports.octoprobe.org



2025-11-11, Hans Märki

Automated HIL testing

Octo probe

When do I need fully automated HIL testing?

- SW runs on multiple HW
- Tests may not be abstracted from HW
- Developers do not have full access/overview to HW
- Project duration > 10 years

- Fast feedback loop
- Keep code quality
- Make sure existing code will not break

Downsides

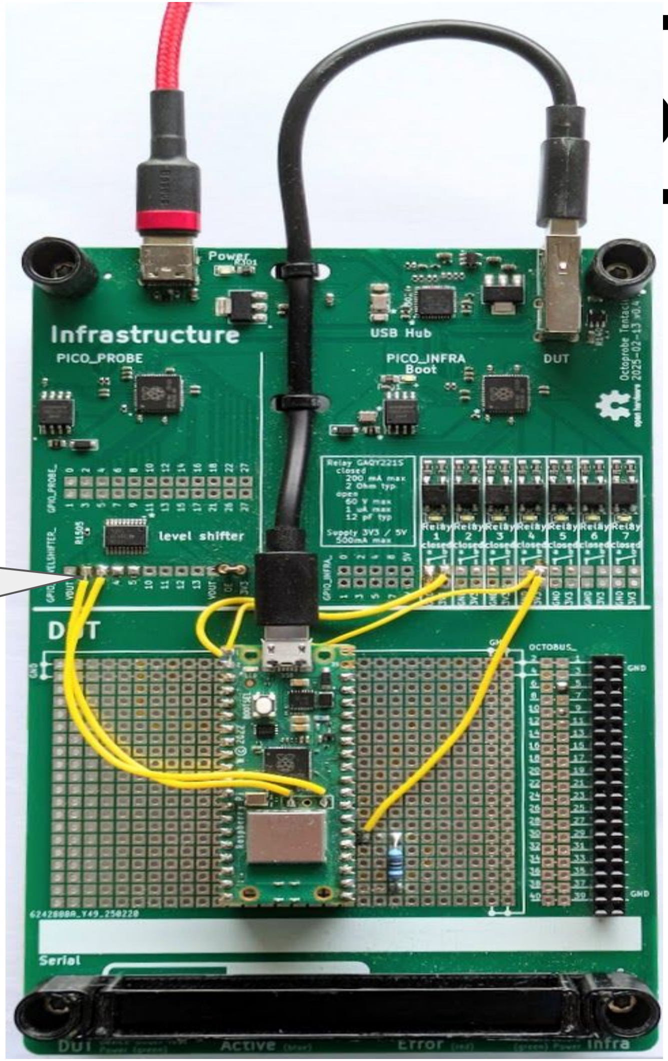
- Implement infra
- Effort to write/maintain tests

Octoprobe Tentacle

Features:

Powercycle
Flashing (boot button)
USB 2.0 FS
7 opto relays

UART
SWD
DAQ



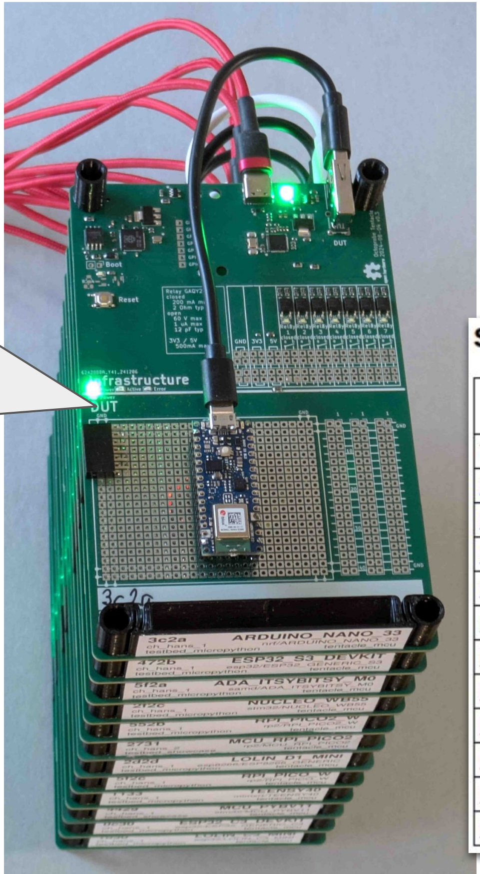
Fully automated
unattended

2025-11-11, Hans Märki

testbed_micropython

Octo probe

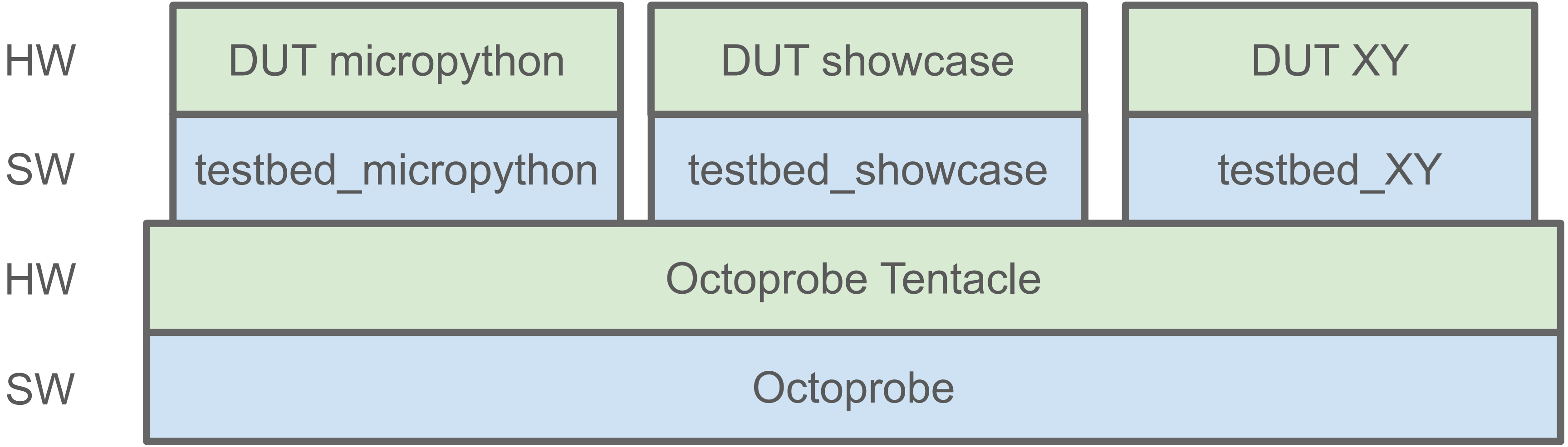
Action (30min):
- git checkout
- Build required firmware / variants
- Flash tentacles
- Run tests
- collect testresults
- Summary report



Summary

Test	Groups run	Groups skipped	Groups error	Tests passed	Tests skipped	Tests failed
Total	74	9	**29**	18269	1956	**595**
RUN-MULTITESTS_MULTIBLUETOOTH	6					
RUN-MULTITESTS_MULTINET		2	**8**			
RUN-NATMODTESTS	10		**2**			
RUN-PERFBENCH	10		**2**			
RUN-TESTS_EXTMOD_HARDWARE_NATIVE	8	1	**3**	10	7	**15**
RUN-TESTS_EXTMOD_HARDWARE	8	1	**3**	15	12	**5**
RUN-TESTS_NET_HOSTED	5	1		45	4	**1**
RUN-TESTS_NET_INET	5	1		57	1	**7**
RUN-TESTS_STANDARD_NATIVE	5	1	**6**	3787	526	**362**
RUN-TESTS_STANDARD_VIA_MPY	8	1	**3**	6664	652	**185**
RUN-TESTS_STANDARD	9	1	**2**	7691	754	**20**

Layering



2025-11-11, Hans Märki

Layer: testbed_micropython

```
mptest list-tentacles
  Tentacle 552b-RPI_PICO2
      infra: 1-7.4.3.1 /dev/ttyACM3
      dut:   1-7.4.3.3 -
    variants=RPI_PICO2,RPI_PICO2_W-RISCV
    futs=FUT_MCU_ONLY,FUT_EXTMOD_HARDWARE
  Tentacle 0c30-ESP32_C3_DEVKIT
      infra: 1-7.3.2.1 /dev/ttyACM2
      dut:   1-7.3.2.3 /dev/ttyUSB1 - CP2102N USB to UART Bridge Controller
    variants=ESP32_GENERIC_C3
    futs=FUT_MCU_ONLY,FUT_EXTMOD_HARDWARE,FUT_WLAN,FUT_BLE
```

FUT - Feature under Test

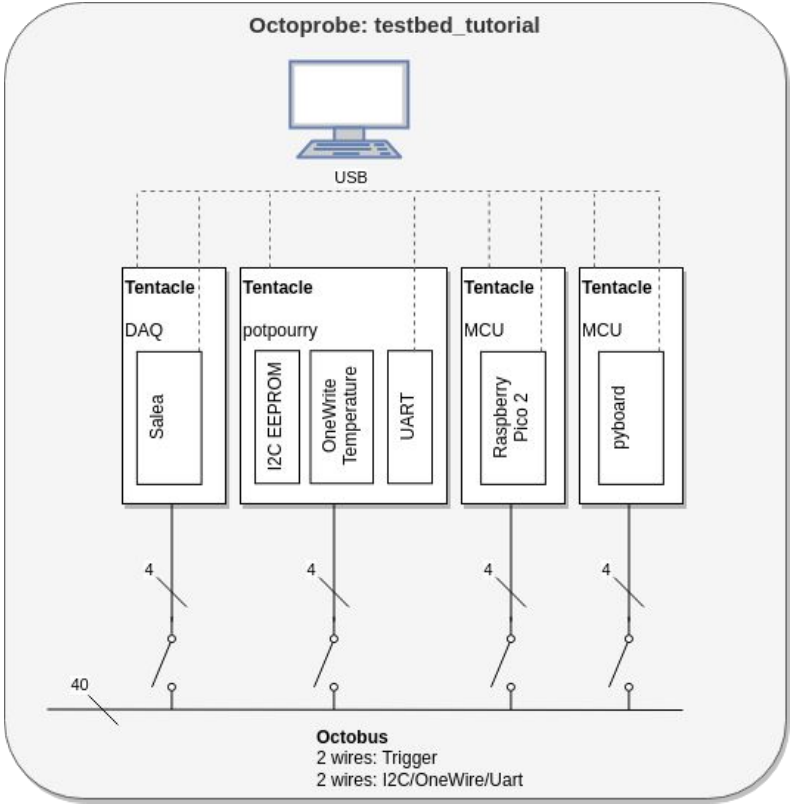
- FUT_MCU_ONLY
- FUT_EXTMOD_HARDWARE: UART loopback, PWM loopback
- FUT_WLAN
- FUT_BLE

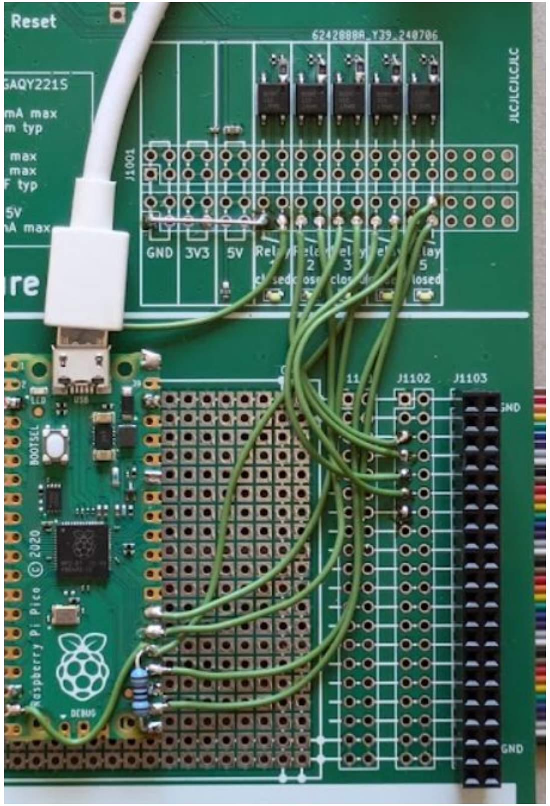
```
mptest test --only-board ESP32_GENERIC_S3 --only-test RUN-PERFBENCH
```

Fragen - Ideen - Feedback

Octo probe

testbed_showcase - octobus

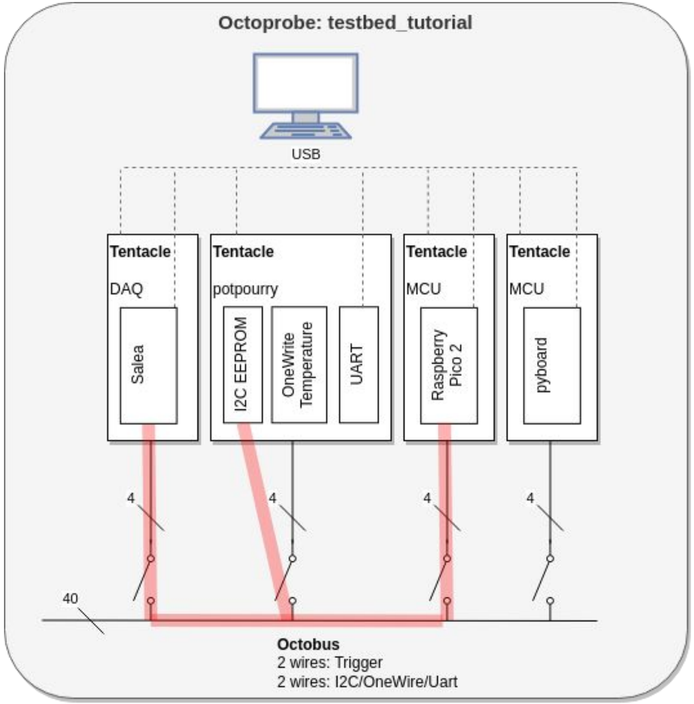


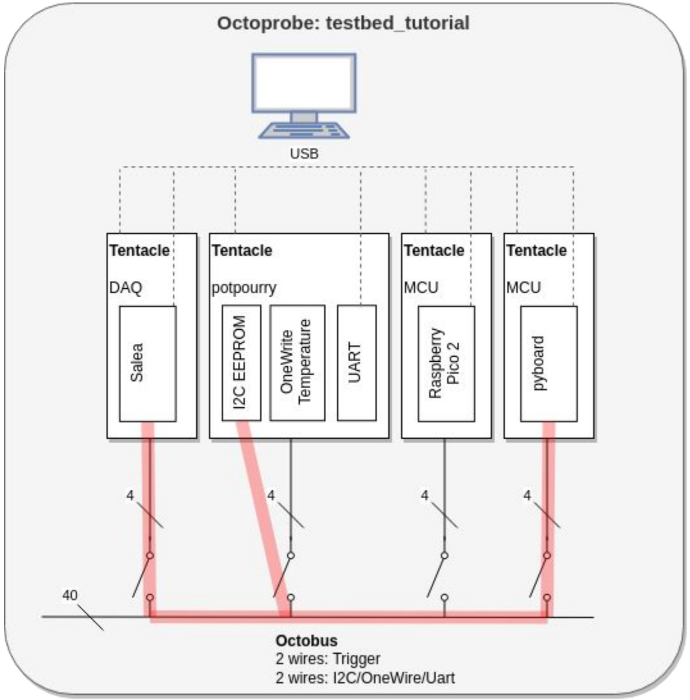


Feature under test

FUT_I2C
FUT_UART
FUT_ONEWIRE

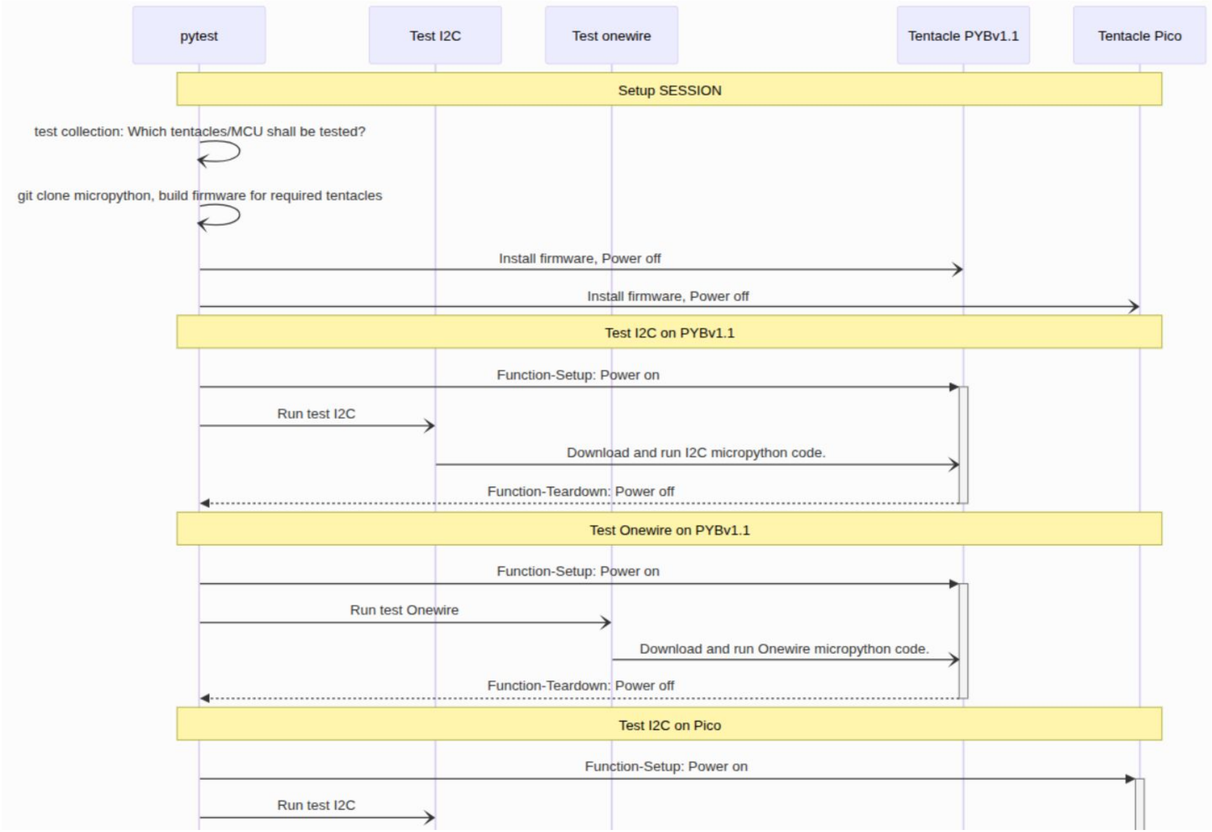
Testing FUT_I2C





2025-11-11, Hans Märki

testbed_showcase - testflow



2025-11-11, Hans Märki

testbed_showcase - pytest

$ pytest --collect-only -q
⇒ Discover tentacles!

$ pytest
tests/test_mip.py ...
tests/test_simple.py ....

Pytest
- another complex layer

Pytest
- Flexible and powerful
- discovery, fixtures, …
- well know

pytest - demo

Onewire on 5425 fail - lets debug

Take over serial line:
mcu.infra.mp_remote_close()

mcu.dut.mp_remote.read_str('ds.scan()')

Control relays:
mcu.infra.mcu_infra.relays([6, 7],[1,2])

How to write your HIL testbed?

Define test cases. Group them by FUT (Feature under Test)
Draw schematics, solder Tentacles
Write `testbed_XY` (flash, run test, collect results…)
Write tests

Octoprobe

- Many details are solved by octoprobe
- Use testframework as pytest, ceedling, …
- Open Source/Open Hardware
- Adapt tentacle (KiCad, JLCPCB)

Octoprobe

- Linux only

2025-11-11, Hans Märki

2025-11-11, Hans Märki

Future

Octoprobe

Use octoprobe regression testing for

- Higher level tests: I2C, SPI, driver testing, library testing
- regression testing: circuit python
- regression testing: tinyusb

Interested to contribute?