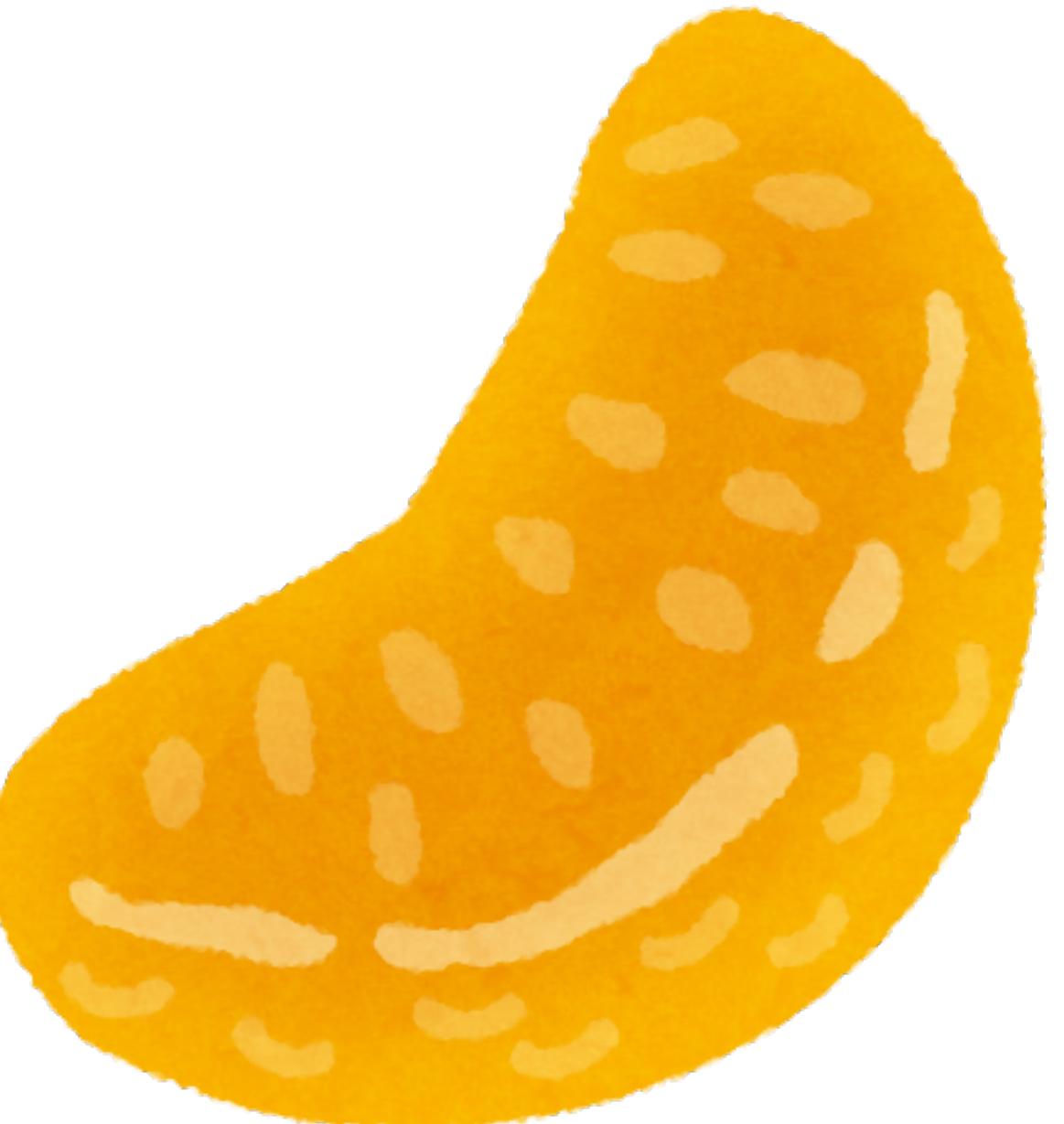


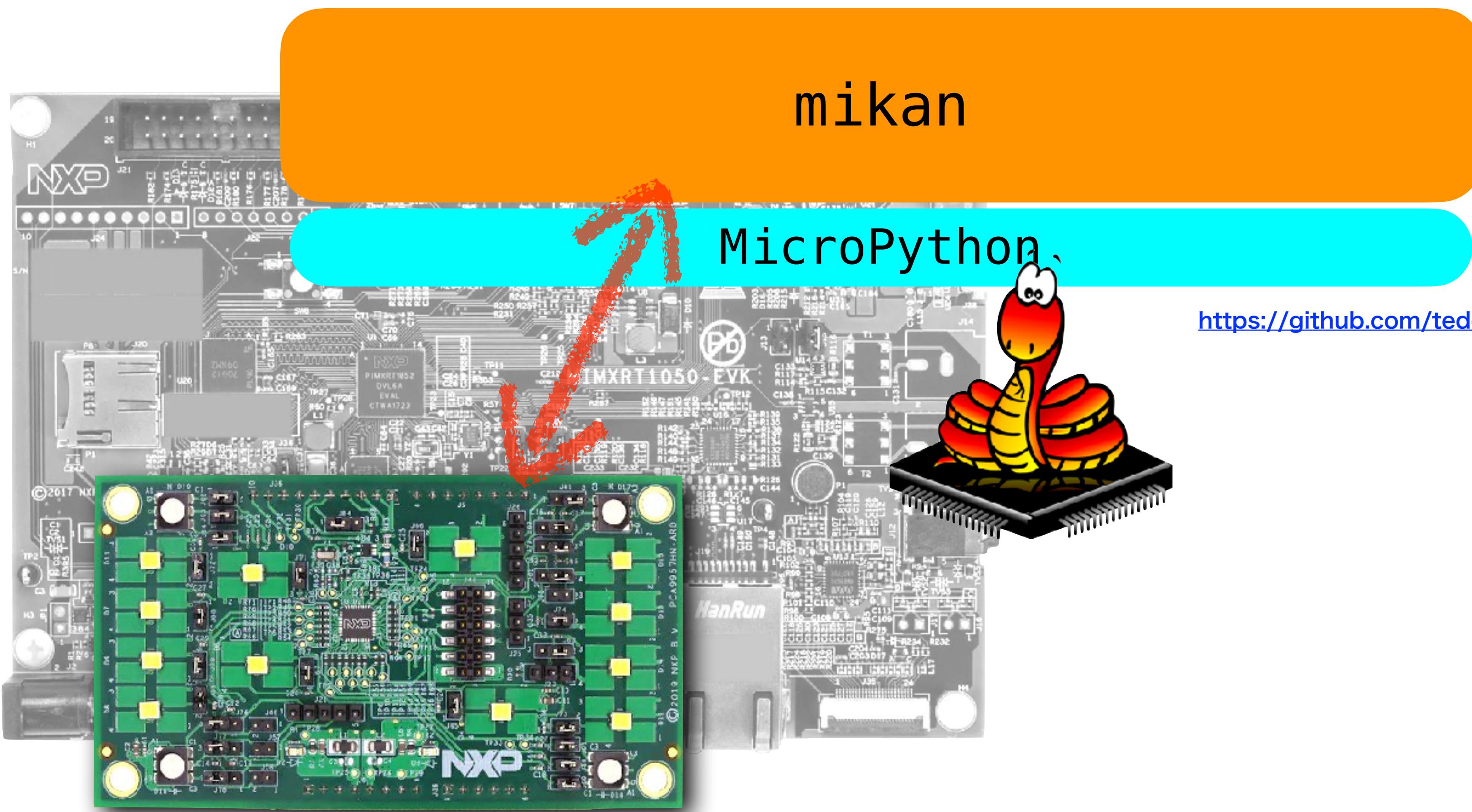
mikan
v1.12

mikan?



mikan?

Peripheral device driver for collection for MicroPython



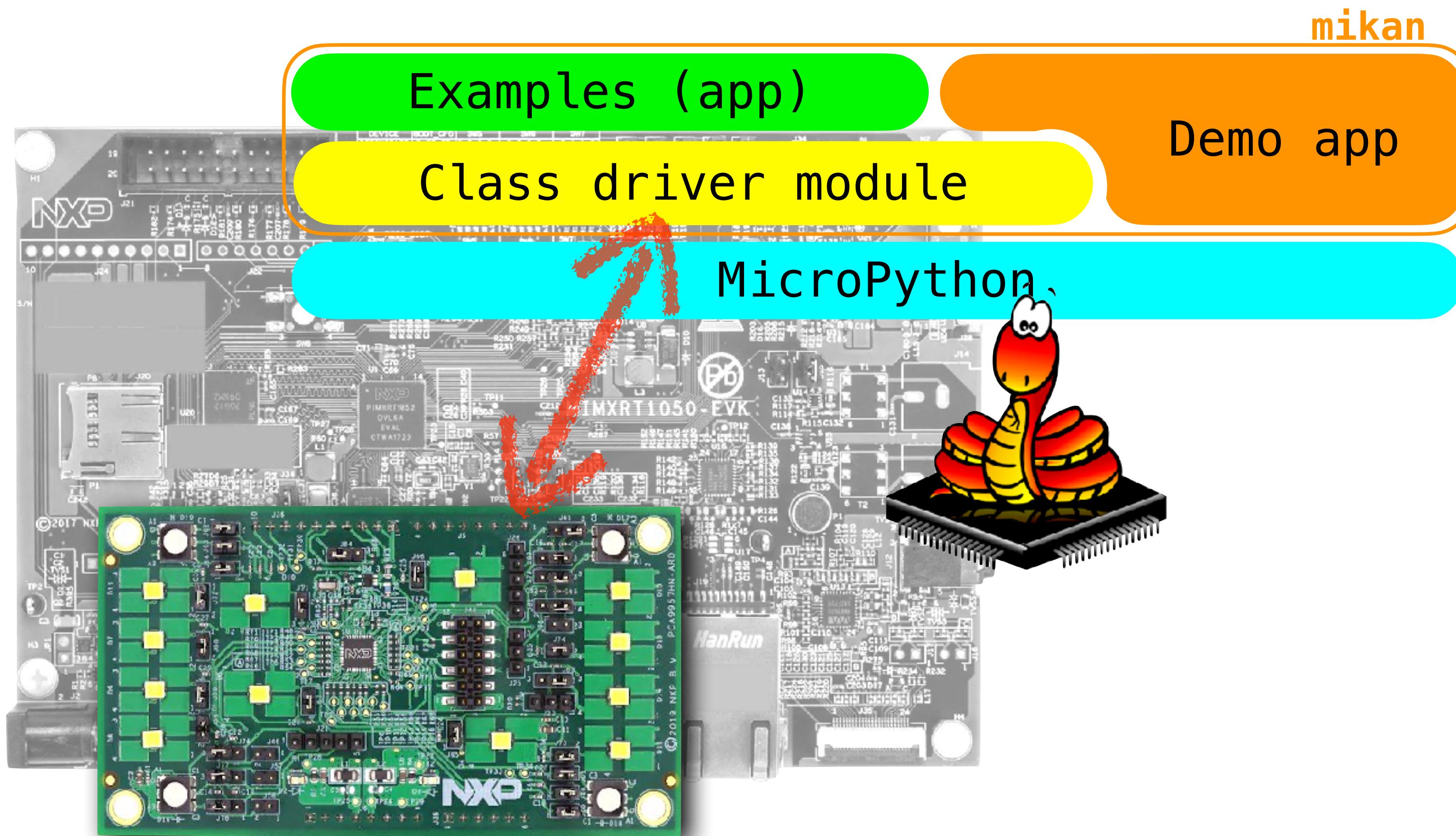
mikan

MicroPython

<https://github.com/teddokano/mikan>

mikan?

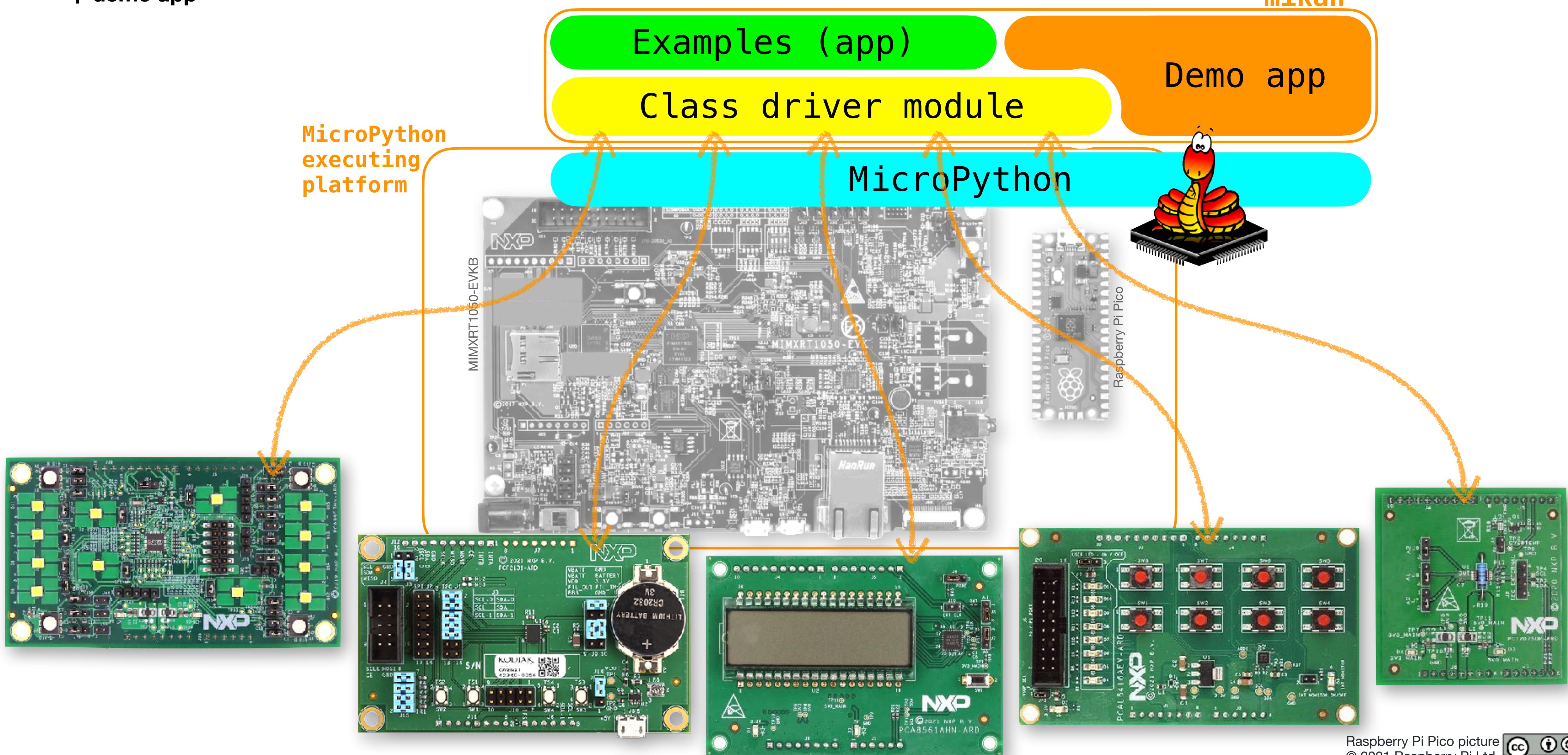
Peripheral device driver for collection for MicroPython



mikan?

Peripheral device driver for collection for MicroPython

- + example code**
- + demo app**

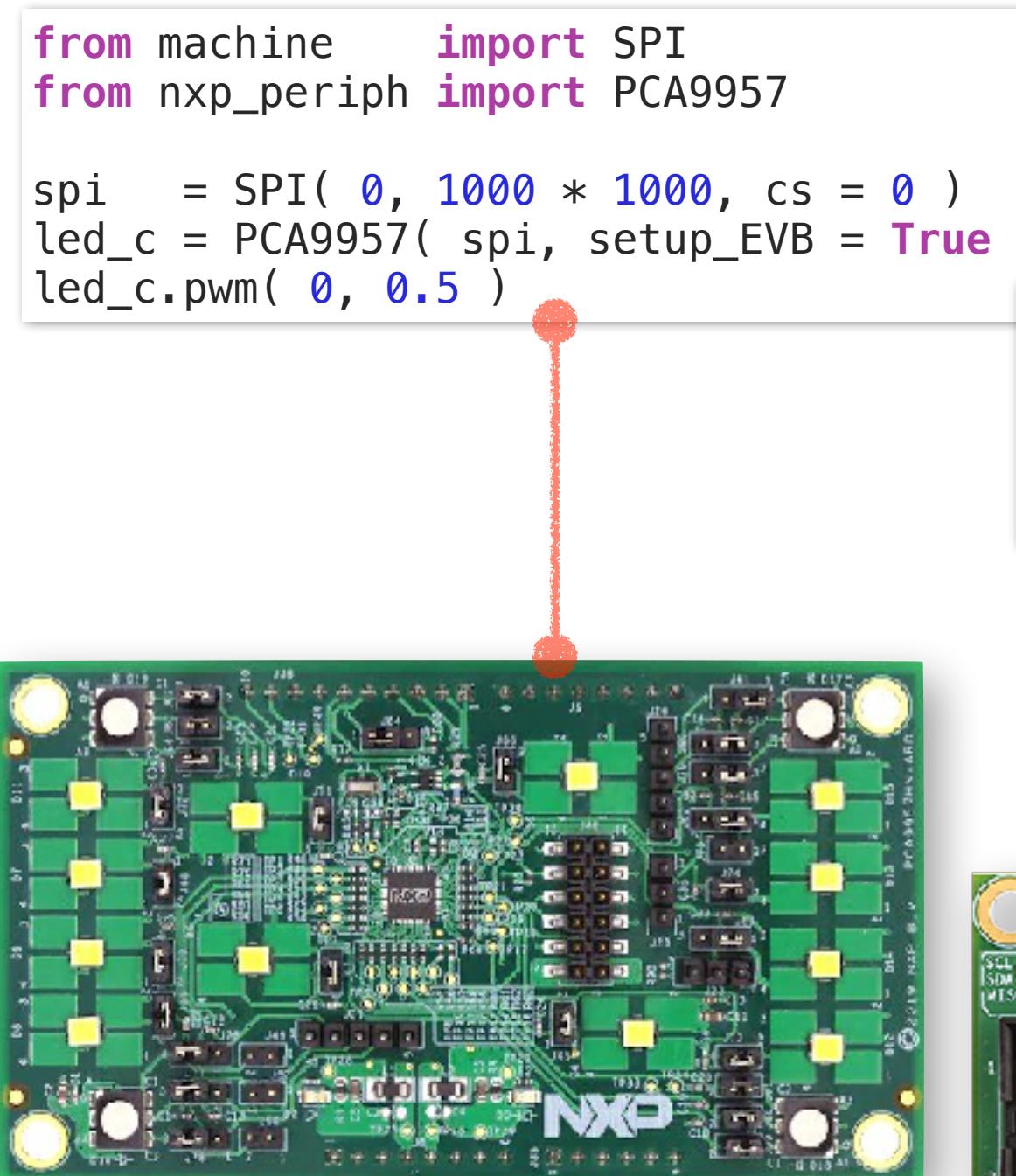


Raspberry Pi Pico picture
© 2021 Raspberry Pi Ltd



mikan?

Enables quick evaluation/test of I²C/SPI devices and rapid prototyping
Can be written by simple Python code



```
from machine import SPI
from nxp_periph import PCA9957

spi = SPI( 0, 1000 * 1000, cs = 0 )
led_c = PCA9957( spi, setup_EVB = True )
led_c.pwm( 0, 0.5 )
```

```
from machine import I2C
from nxp_periph import PCF2131

rtc = PCF2131( I2C( 0 ) )
print( rtc.now() )
```

```
from machine import I2C
from nxp_periph import PCA8561

lcd = PCA8561( I2C( 0 ) )
lcd.write_registers( "Display_ctrl_1", 0x01 )
lcd.puts( "test" )
```

```
from machine import I2C
from nxp_periph import PCAL6416

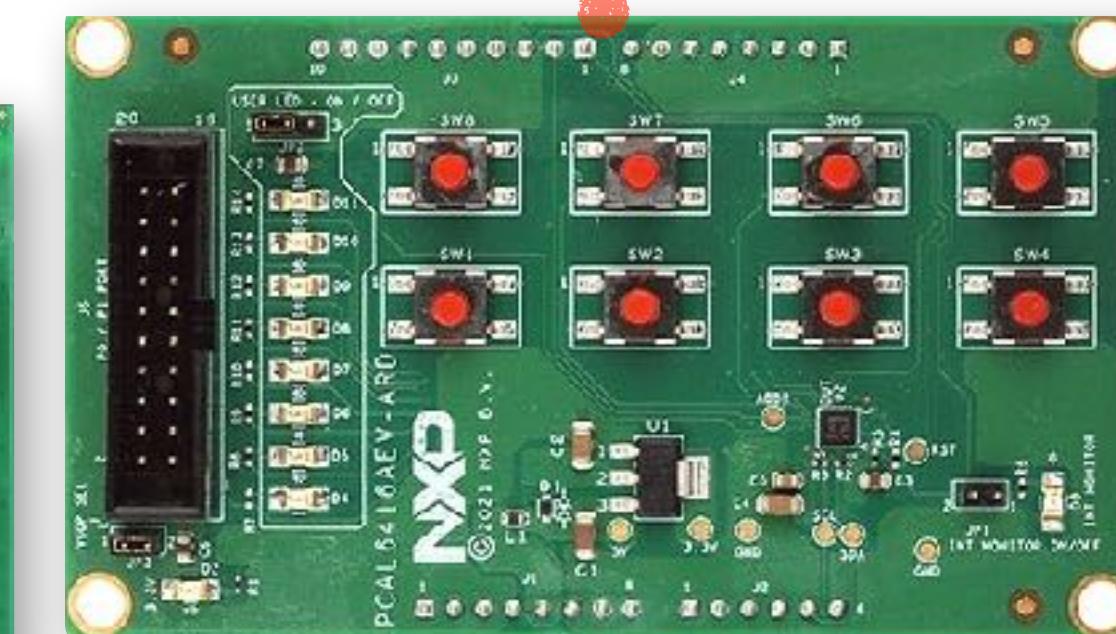
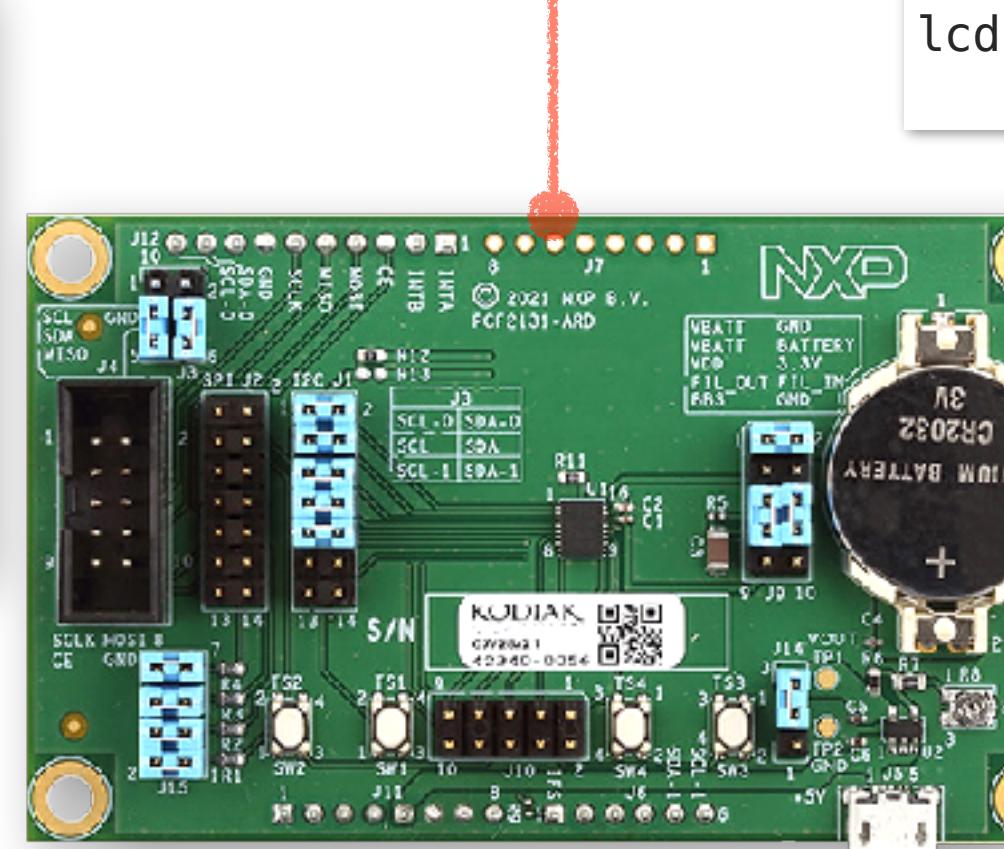
gpio = PCAL6416( I2C( 0 ),
                  setup_EVB = True )
io_config_and_pull_up = [ 0x00, 0xFF ]

gpio.config = io_config_and_pull_up
gpio.pull_up = io_config_and_pull_up
gpio.pull_en = [ 0xFF ] * 2
```

```
gpio.value = 0xA5
print( gpio.value )
```

```
from machine import I2C
from nxp_periph import PCT2075

temp_sensor = PCT2075( I2C( 0 ) )
print( temp_sensor.temp )
```



mikan?

Enables quick evaluation/test of I²C/SPI devices and rapid prototyping
Can be written by simple Python code
For example: PCT2075 = a temp sensor

```
from machine import I2C  
from nxp_periph import PCT2075  
  
temp_sensor= PCT2075( I2C( 0 ) )  
print( temp_sensor.temp )
```



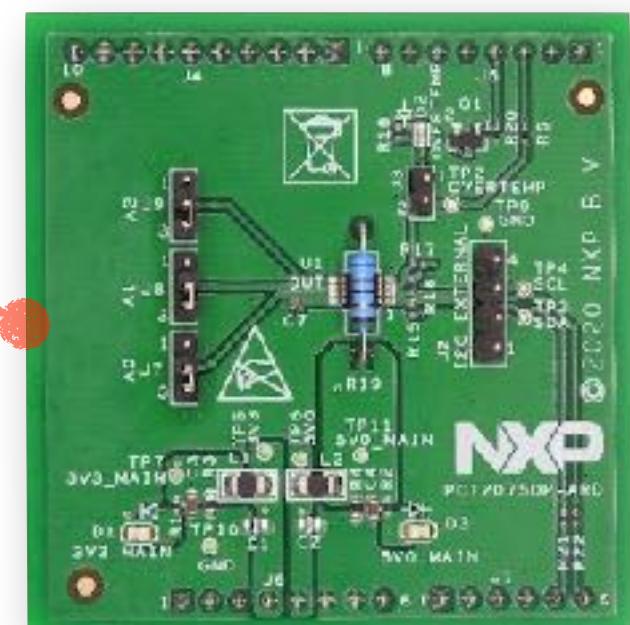
Prepare to use class libraries



Make a class instance



Call an instance method and print



License

main ▾ **mikan / LICENSE** Go to file ⋮

 teddokano/mikan is licensed under the **MIT License**

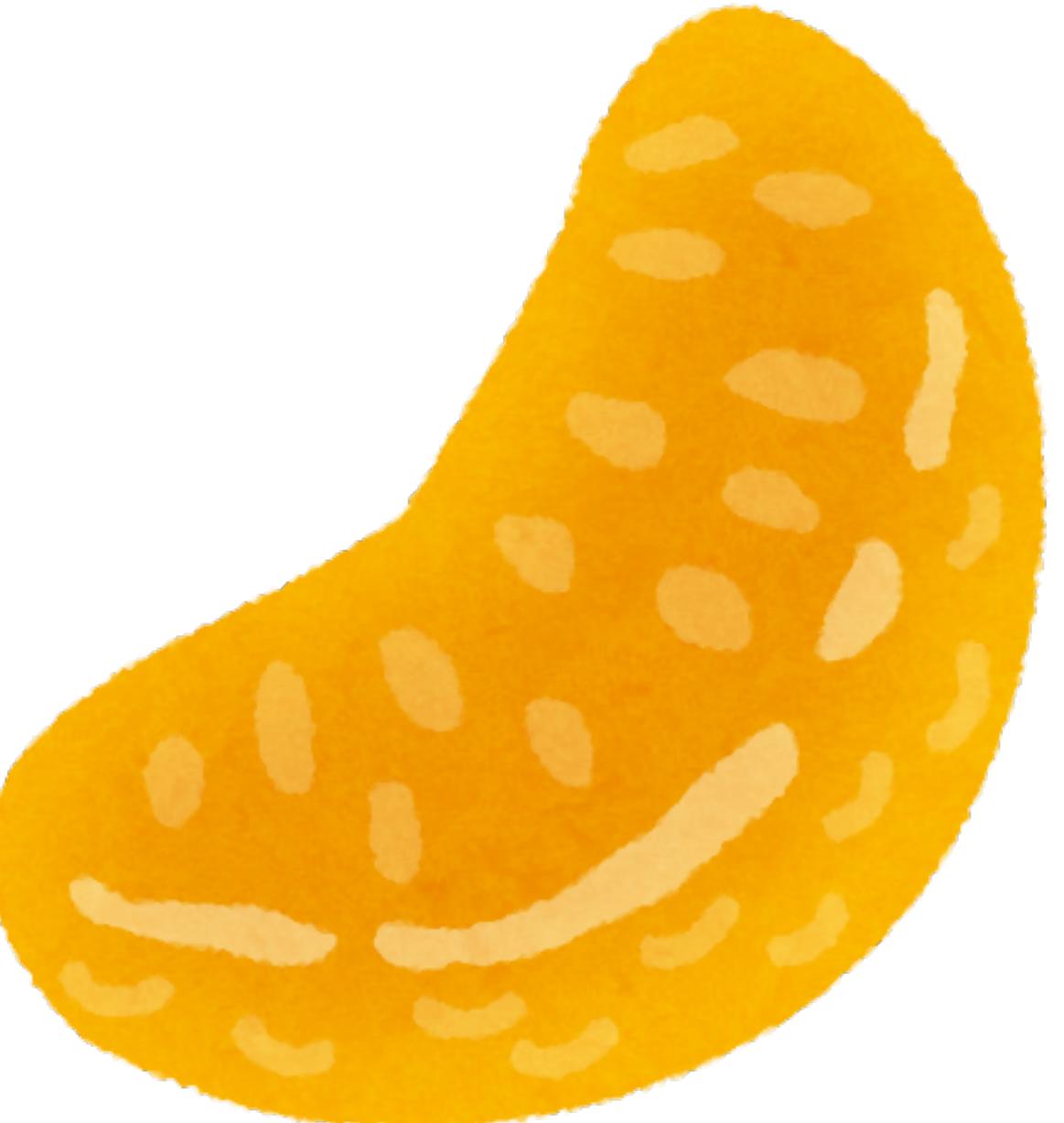
A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions	Limitations	Conditions
✓ Commercial use	✗ Liability	 ⓘ License and copyright notice
✓ Modification	✗ Warranty	
✓ Distribution		
✓ Private use		

This is not legal advice. [Learn more about repository licenses.](#)

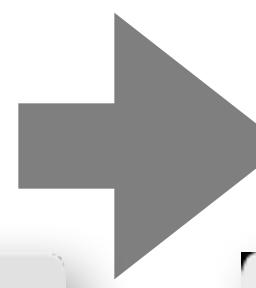
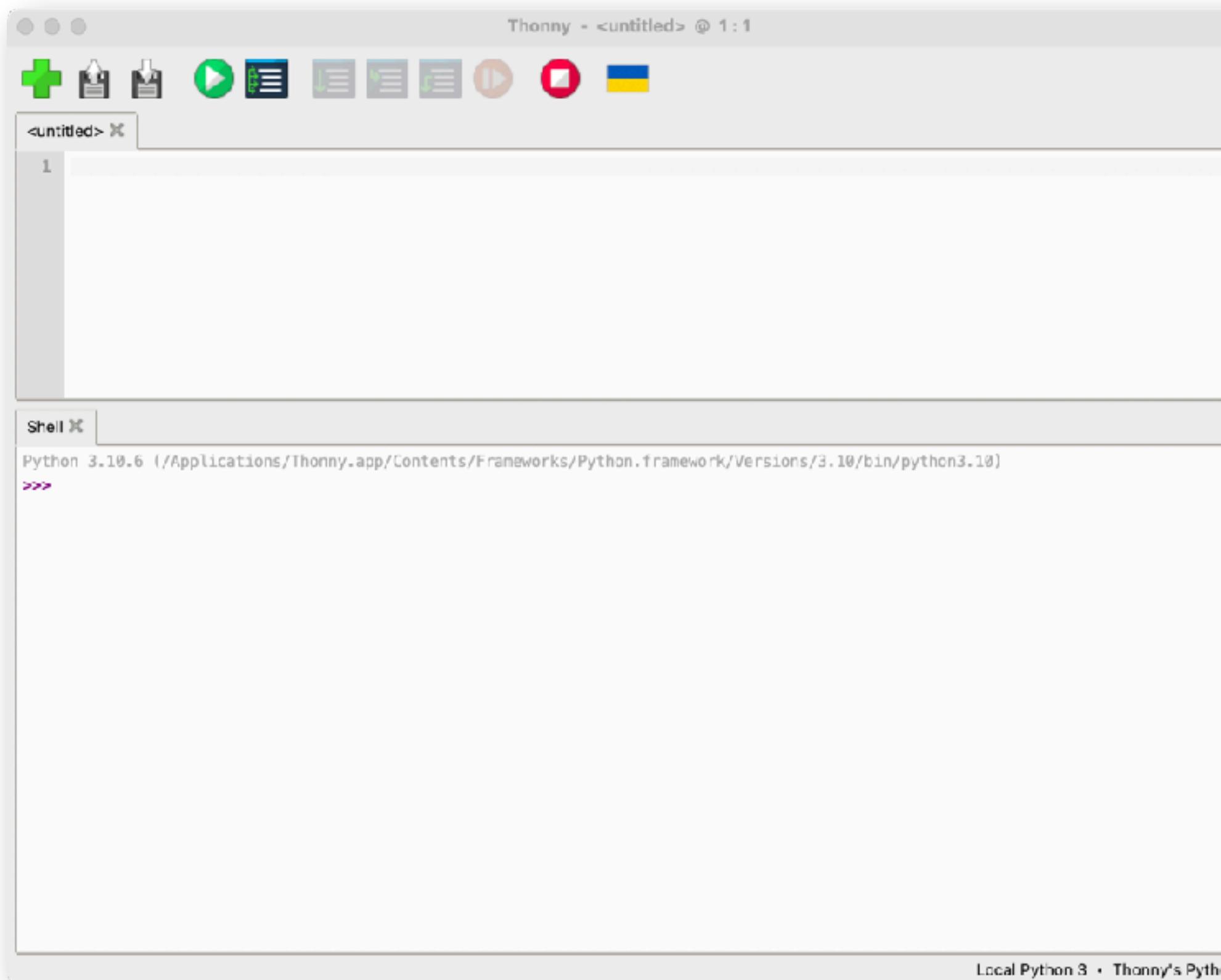
<https://github.com/teddokano/mikan/blob/main/LICENSE>

Get ready
mikan

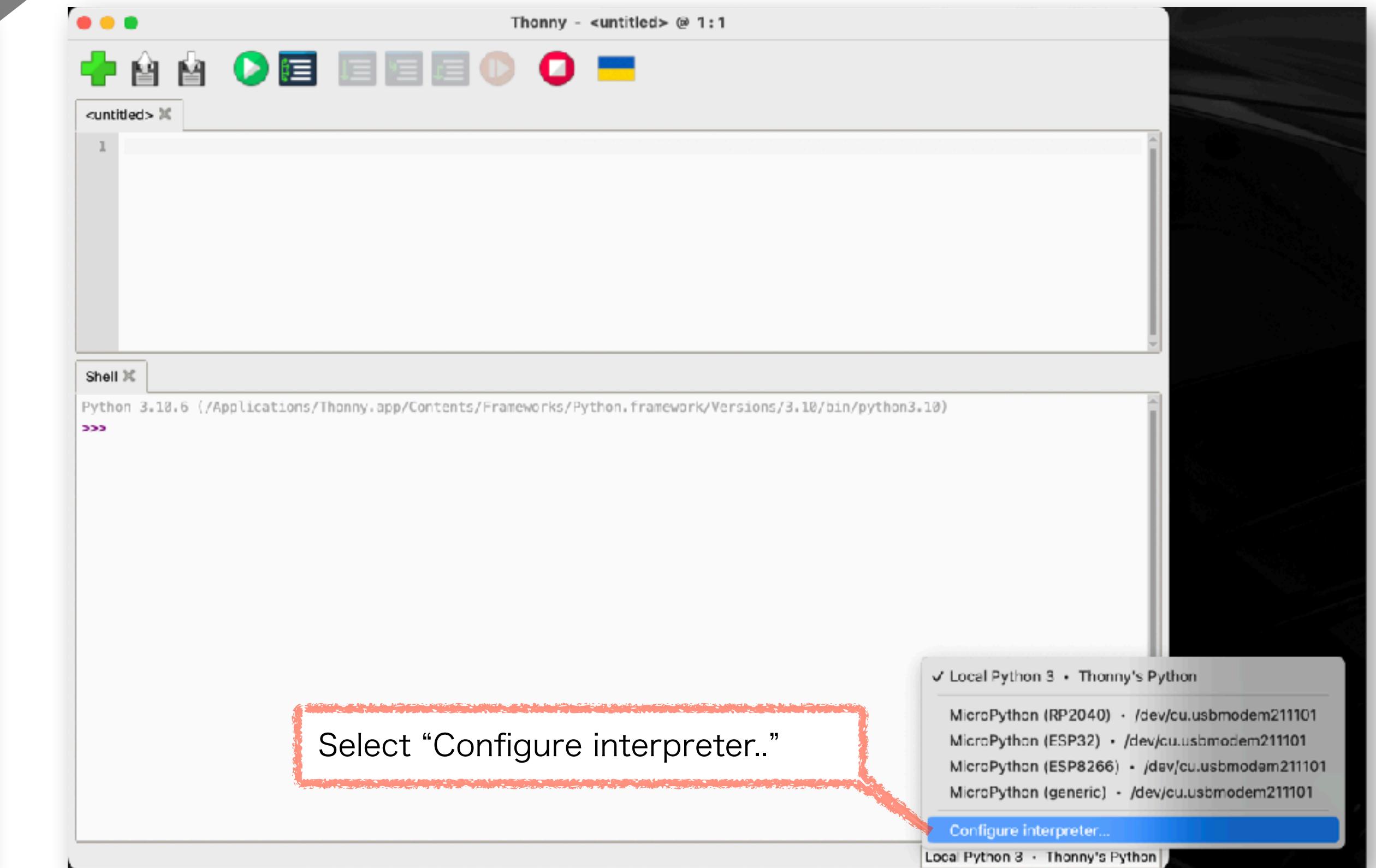


MicroPython executing environment setup

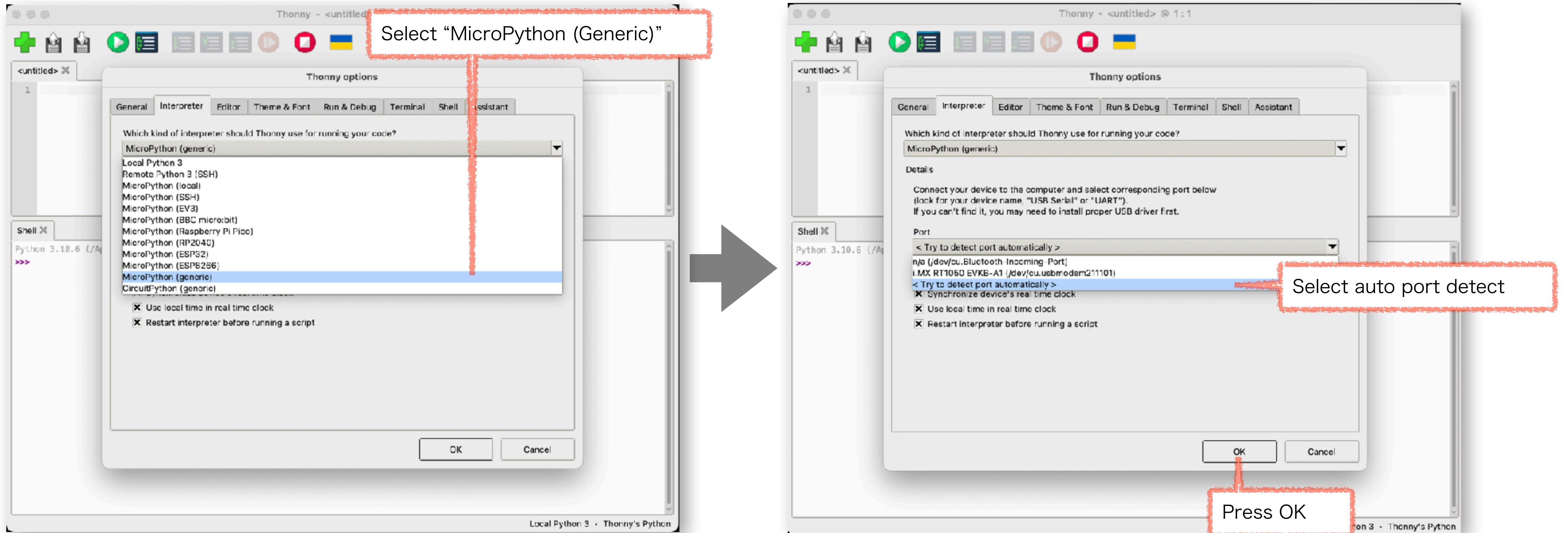
Install [Thonny](#) and launch



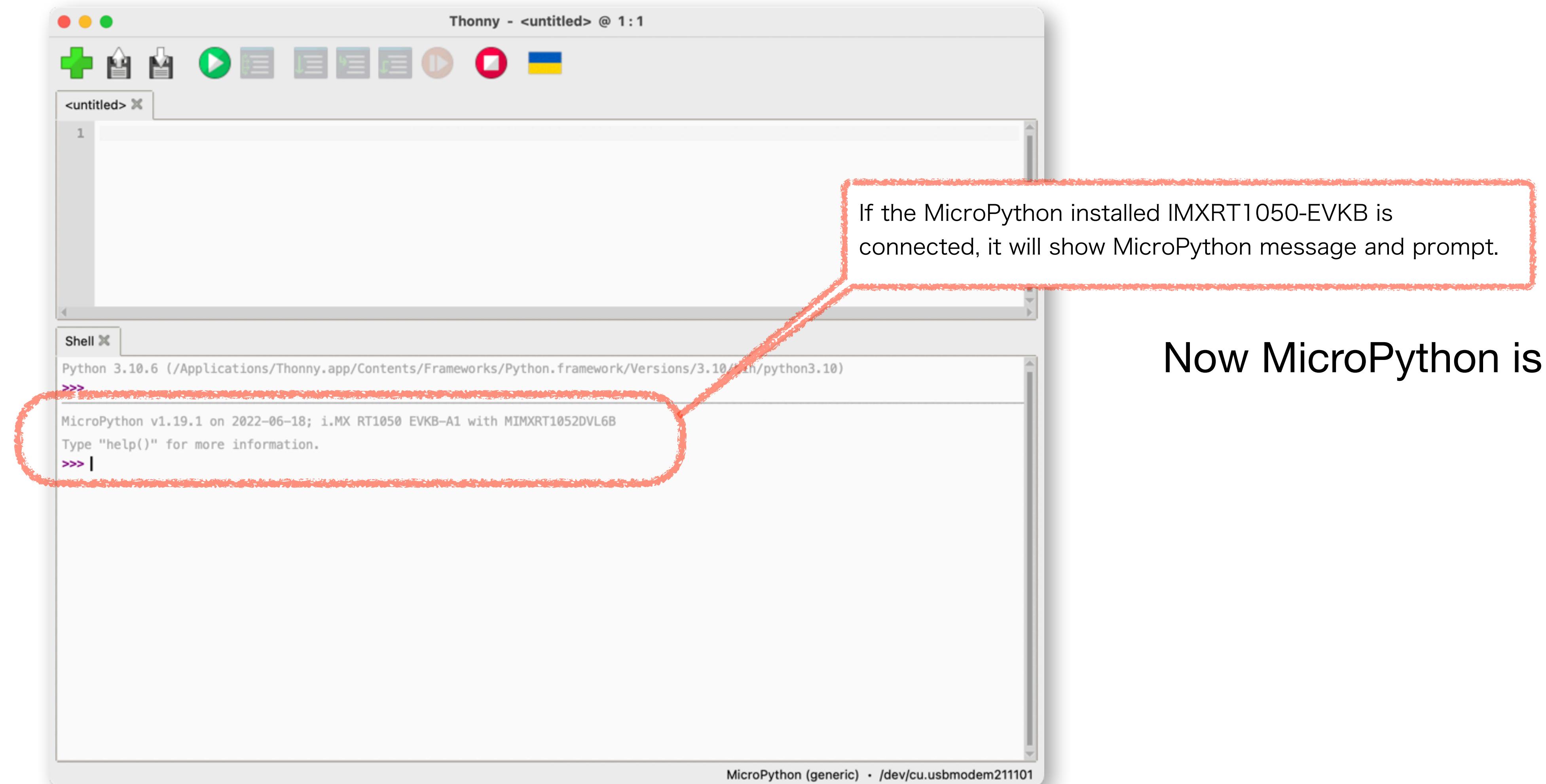
Click right-bottom pop-up menu



MicroPython executing environment setup



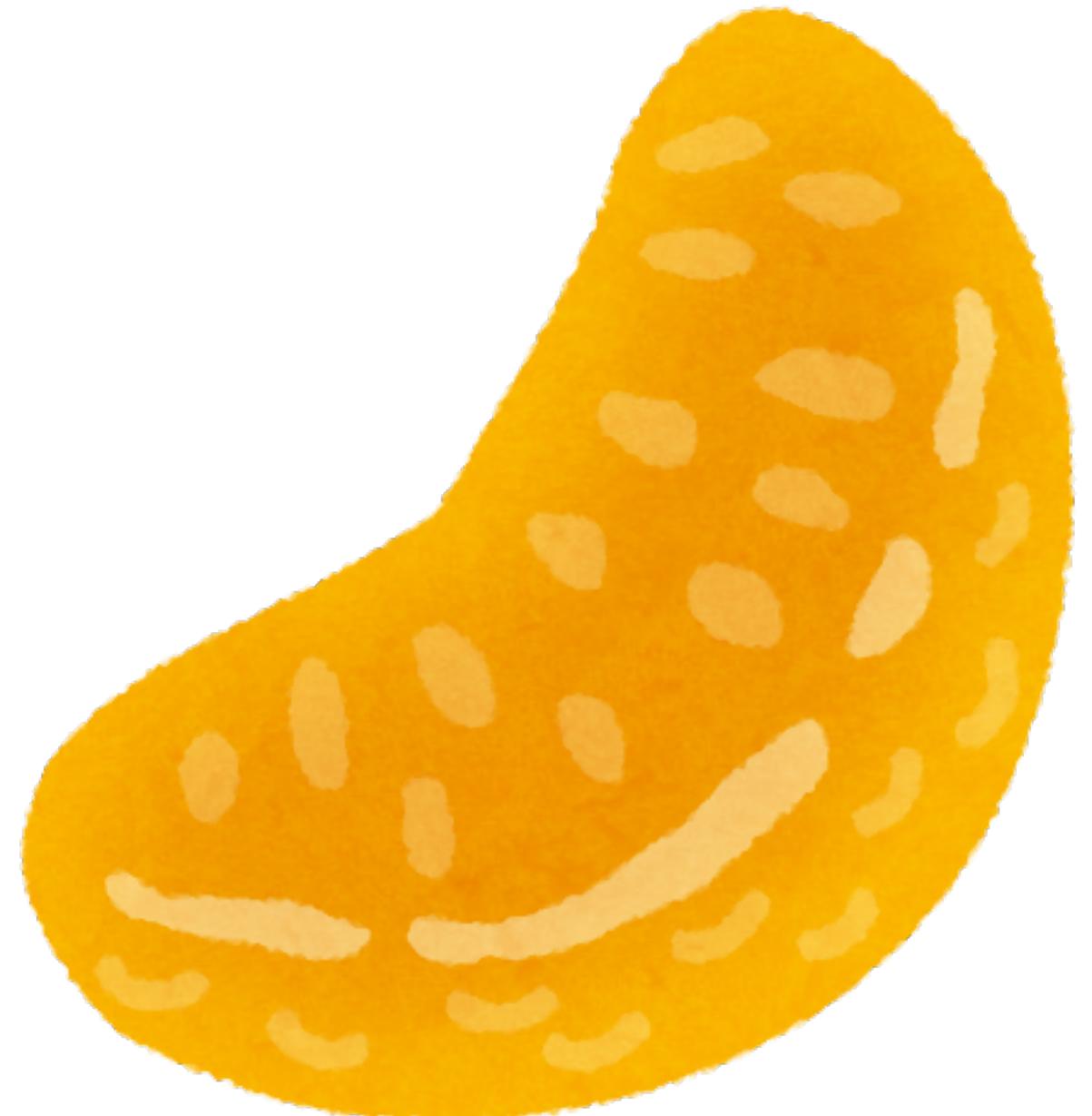
MicroPython executing environment setup



Now MicroPython is ready to run!

Basics of MicroPython

mikan



MicroPython?

<https://micropython.org/>

The screenshot shows a web browser window displaying the MicroPython website at micropython.org. The page has a dark header with white text and links for DOWNLOAD, DOCS, DISCORD, DISCUSSIONS, WIKI, and STORE. The main title "MicroPython" is prominently displayed in large black letters. Below the title is a red callout box containing text about MicroPython's implementation and optimization. Further down, there are sections about the pyboard and a compatibility note.

MicroPython

MicroPython is a lean and efficient implementation of the [Python 3](#) programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.

The MicroPython [pyboard](#) is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to

Why MicroPython?

It's popular!



- Easy to use
- Easy to learn
- Including SDK for hardware
- Runs anywhere on MicroPython ported platform



- Limited platforms
- Big footprint/memory



Use C/C++

Try some samples

The screenshot shows a GitHub repository page for 'teddokano/oop_for_i2c_dev'. The repository has 1 branch and 0 tags. The commit history shows the following activity:

Commit	Message	Time
teddokano ver 00	samples moved into "sample" folder	last week
.gitignore	started to write README.md	last week
LICENSE	Initial commit	last week
README.md	ver 00	last week

The README.md file contains the following Japanese text:

I²Cデバイスをクラス化するまで：ステップ・バイ・ステップ

これはなに？

MicroPythonを使ってマイコンに接続したLM75B互換のI²C温度センサ（LM75B, PCT2075, P3T1085など）を読んでくる例を使って、オブジェクト指向でハードを抽象化する方法を段階的に説明します。

このドキュメント自体がこのリポジトリのメイン部分で、samples フォルダ以下のコードは各ステップで示したコードの例です。

最初にマイコン基板のIMXRT1050-EVKBを用いて、まず基板上のLEDの点滅を確認、その後MicroPythonのざく基本的な動作を確認します。

その後でI²Cで接続されたデバイスのアクセス試してからコードを徐々に変更、クラス化するまでを説明します。

動かしてみる

ステップ0：マイコン基板動作の確認

IMXRT1050-EVKBには"D4"ピンにLEDが接続されています。これを対話式の環境(REPL)で動かしてみます。

The repository has 1 branch and 0 tags. The commit history shows the following activity:

Commit	Message	Time
teddokano ver 00	samples moved into "sample" folder	last week
.gitignore	started to write README.md	last week
LICENSE	Initial commit	last week
README.md	ver 00	last week

The README.md file contains the following Japanese text:

I²Cデバイスをクラス化するまで：ステップ・バイ・ステップ

これはなに？

MicroPythonを使ってマイコンに接続したLM75B互換のI²C温度センサ（LM75B, PCT2075, P3T1085など）を読んでくる例を使って、オブジェクト指向でハードを抽象化する方法を段階的に説明します。

このドキュメント自体がこのリポジトリのメイン部分で、samples フォルダ以下のコードは各ステップで示したコードの例です。

最初にマイコン基板のIMXRT1050-EVKBを用いて、まず基板上のLEDの点滅を確認、その後MicroPythonのざく基本的な動作を確認します。

その後でI²Cで接続されたデバイスのアクセス試してからコードを徐々に変更、クラス化するまでを説明します。

動かしてみる

ステップ0：マイコン基板動作の確認

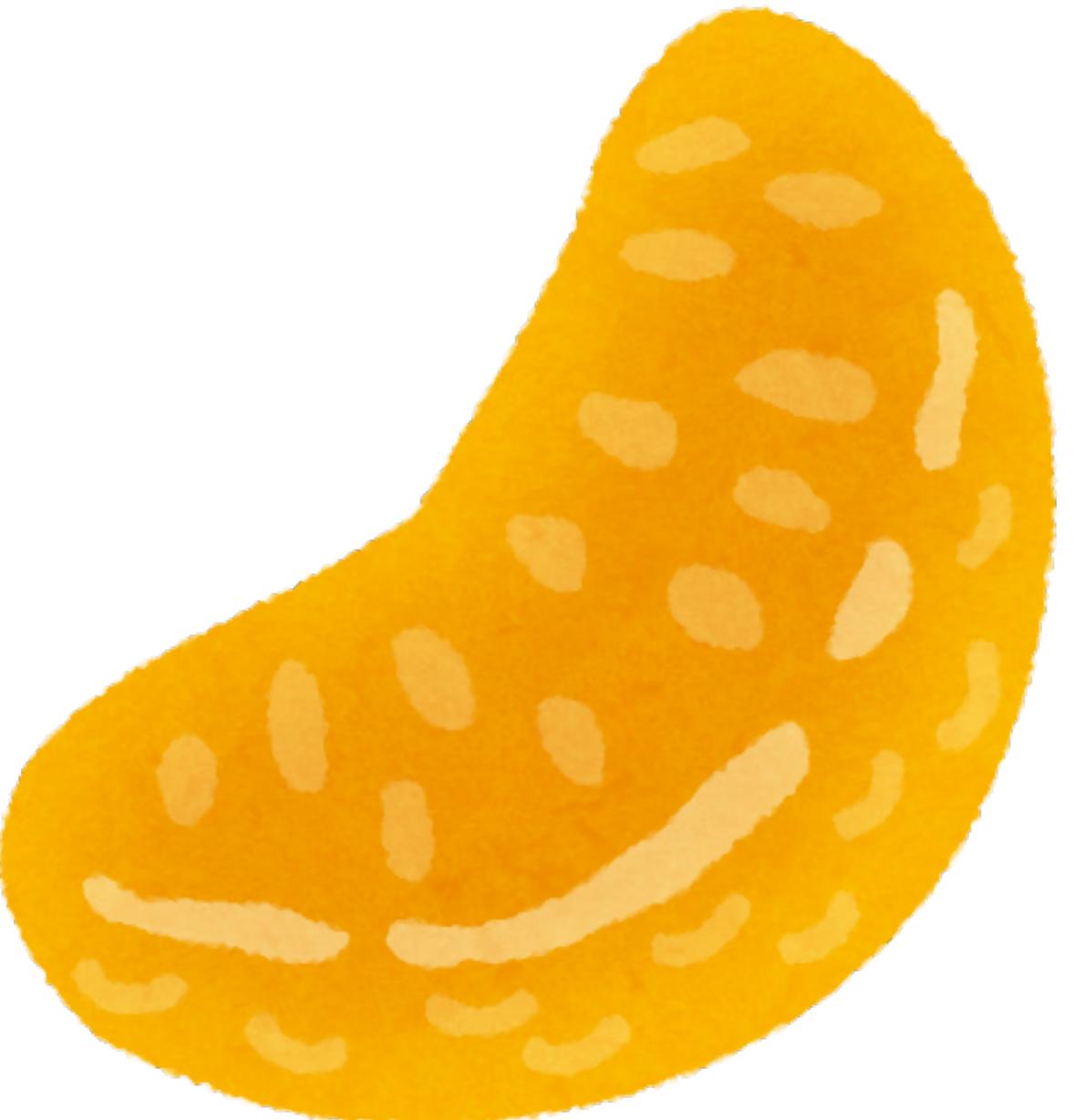
IMXRT1050-EVKBには"D4"ピンにLEDが接続されています。これを対話式の環境(REPL)で動かしてみます。

From step0 to step8 in [this sample code](#) (Japanese) could be good samples to operate MicroPython

This article explains why and how to applying object-oriented-programing style to device communication. However, most of samples are explaining basic operation of MicroPython.

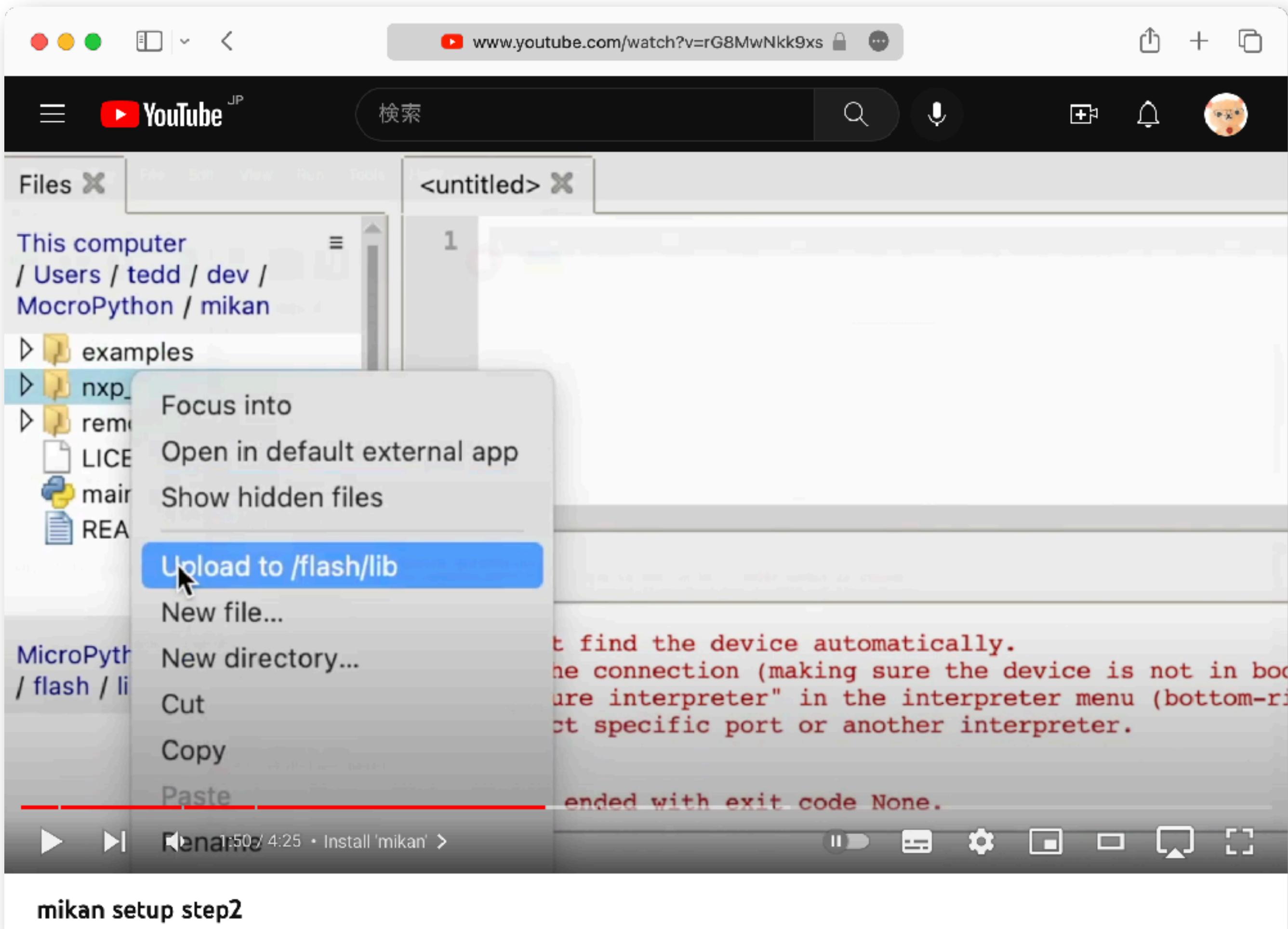
https://github.com/teddokano/oop_for_i2c_dev

class driver
mikan



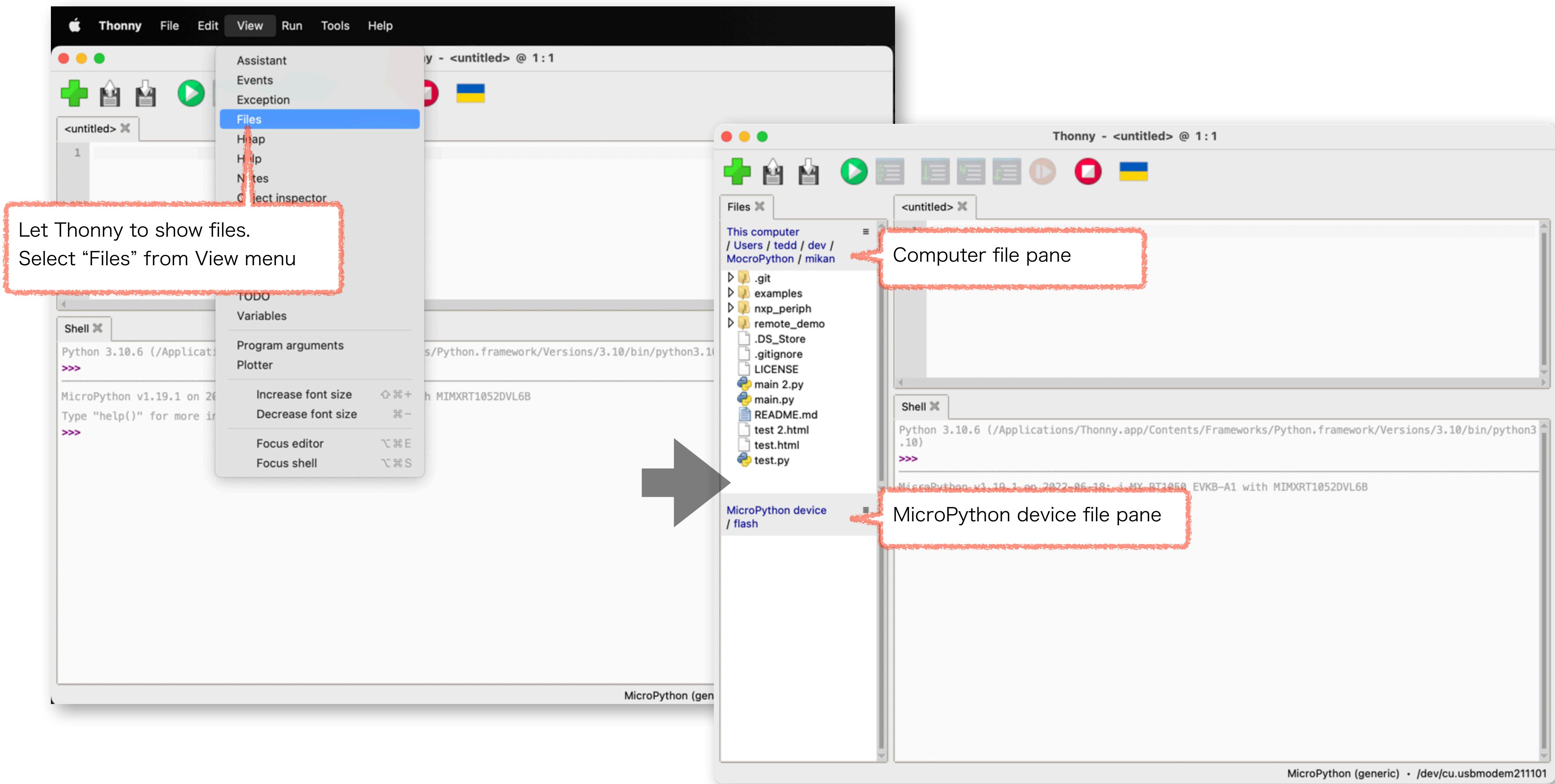
mikan install

Follow this video: <https://youtu.be/rG8MwNkk9xs>

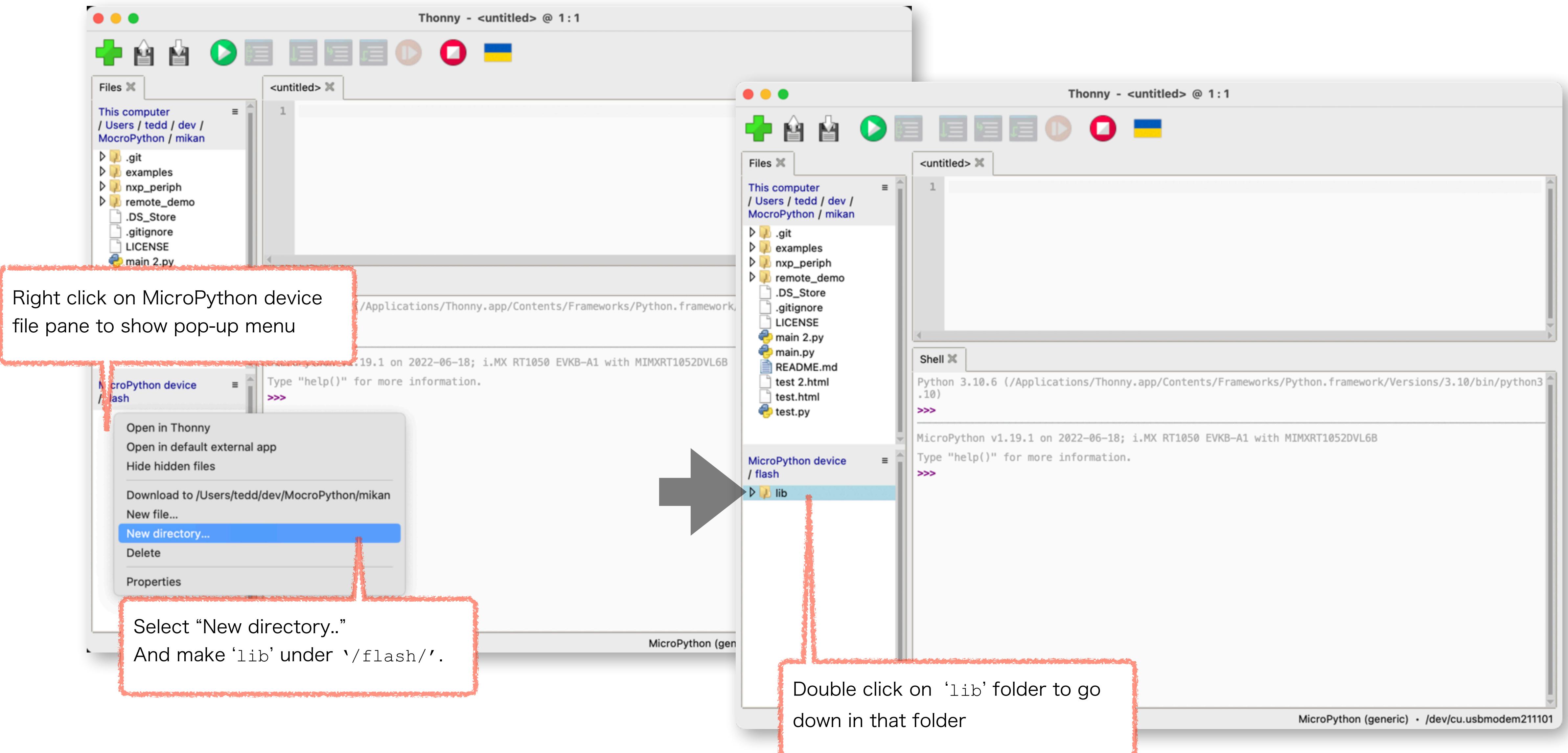


<https://github.com/teddokano/mikan>

Installing mikan class library



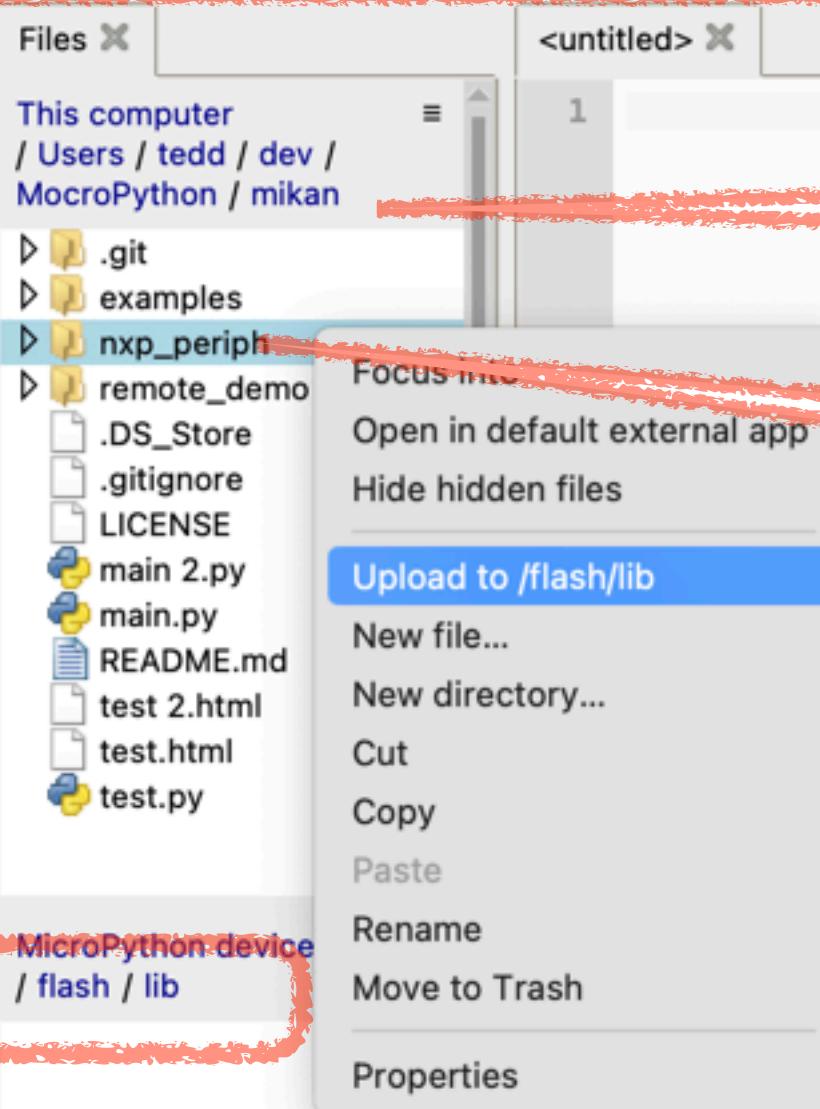
Installing mikan class library



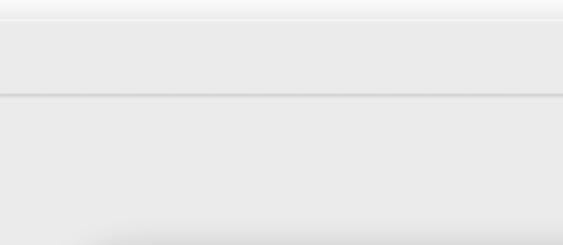
Installing mikan class library

Make sure now the MicroPython device file pane is in '/flash/lib/'

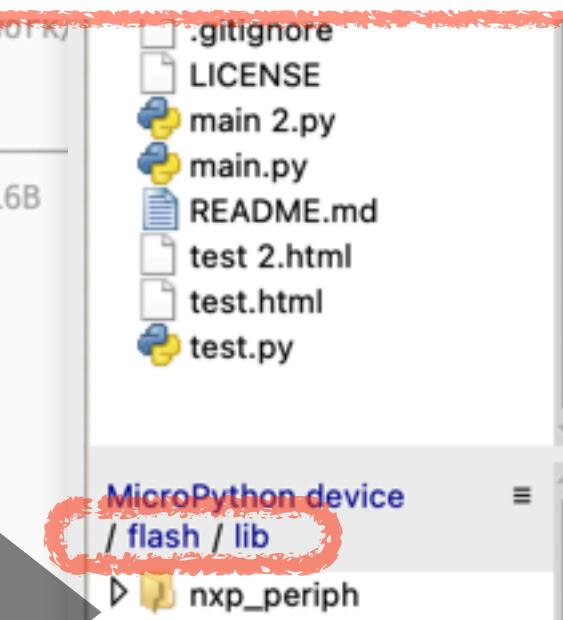
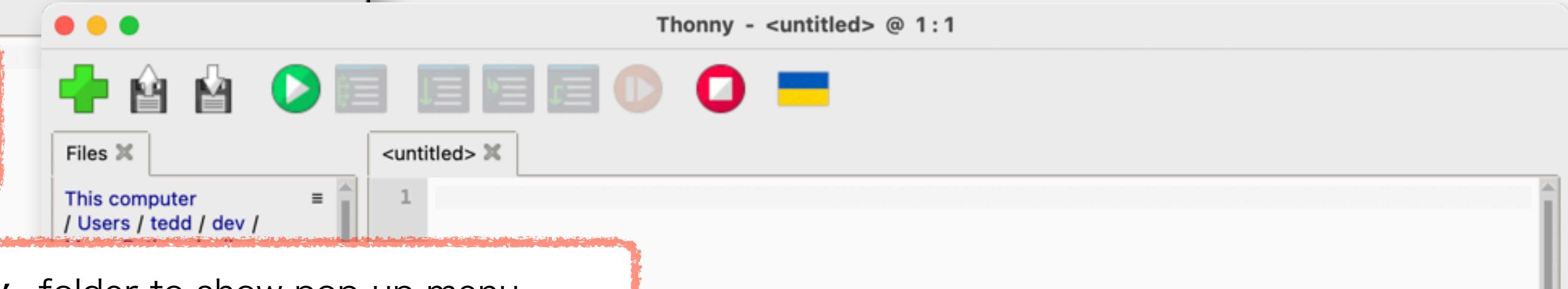
Thonny - <untitled> @ 1:1



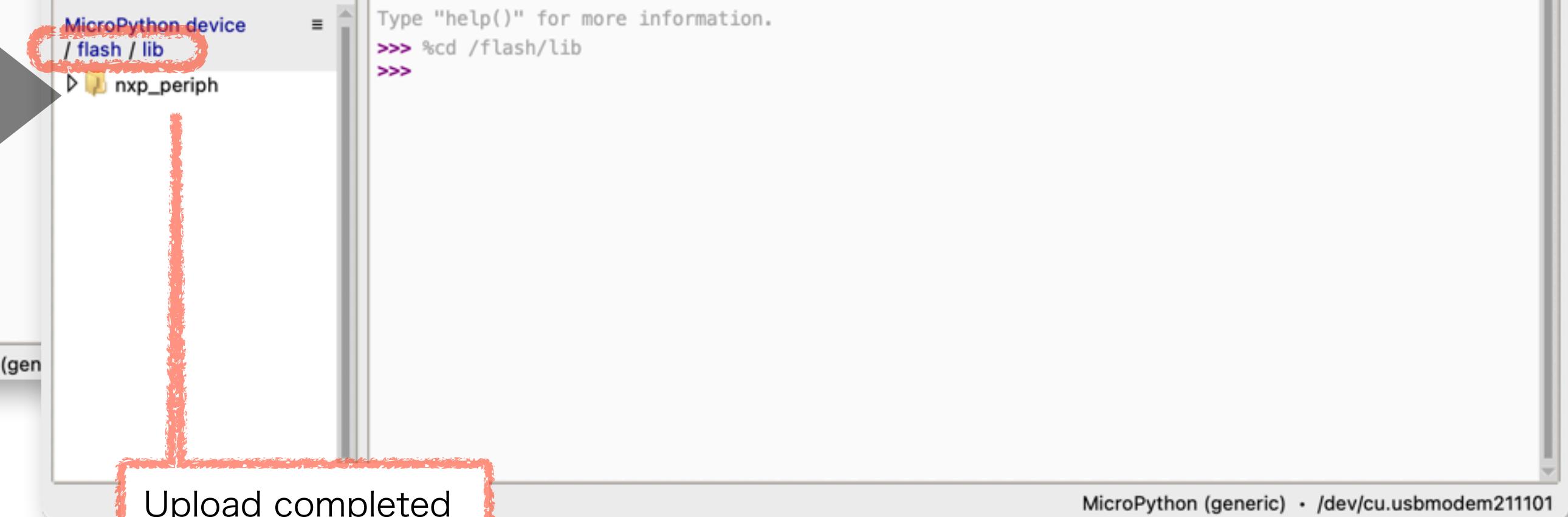
Go down into 'mikan' folder



Right click on 'nxp_periph' folder to show pop-up menu
Select "Upload to /flash/lib"

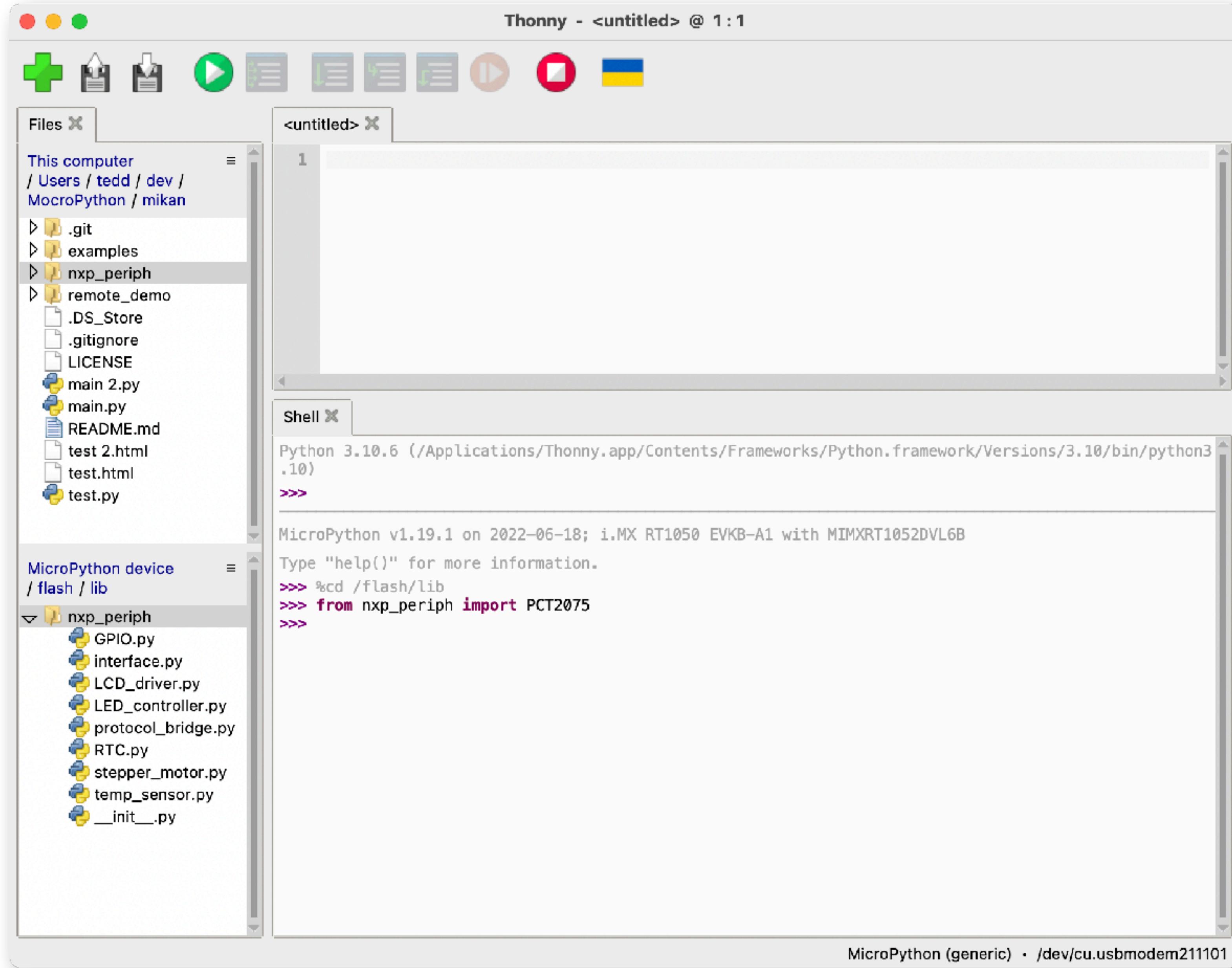


Upload completed



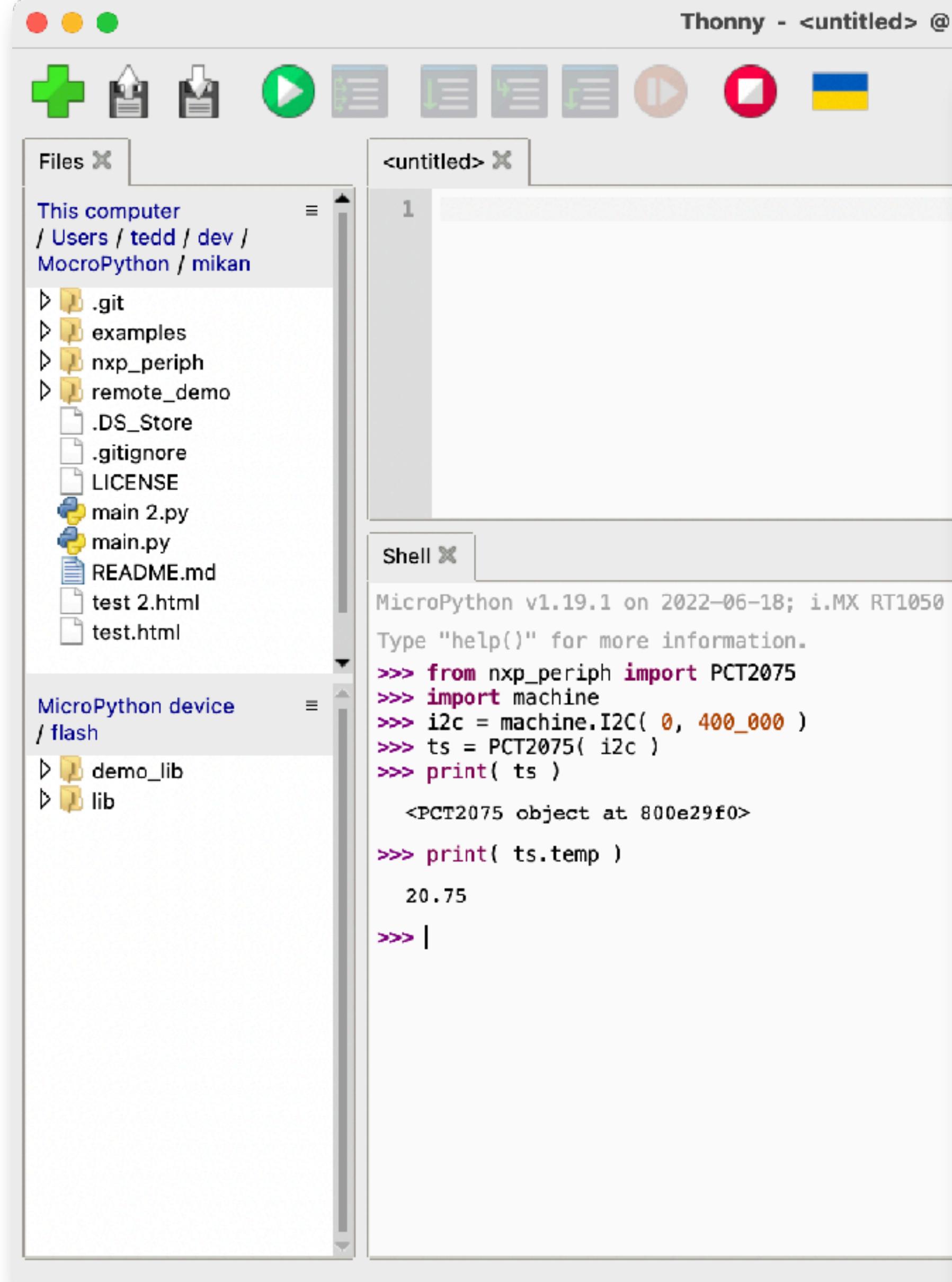
MicroPython (generic) • /dev/cu.usbmodem211101

Installing mikan class library



mikan class library is ready!

Basic mikan operation



Type-in these commands after prompt (">>>") in shell window.

```
>>> from nxp_periph import PCT2075
>>> import machine
>>> i2c = machine.I2C( 0 )
>>> ts = PCT2075( i2c )
>>> print( ts )
```

<PCT2075 object at 800e29f0>

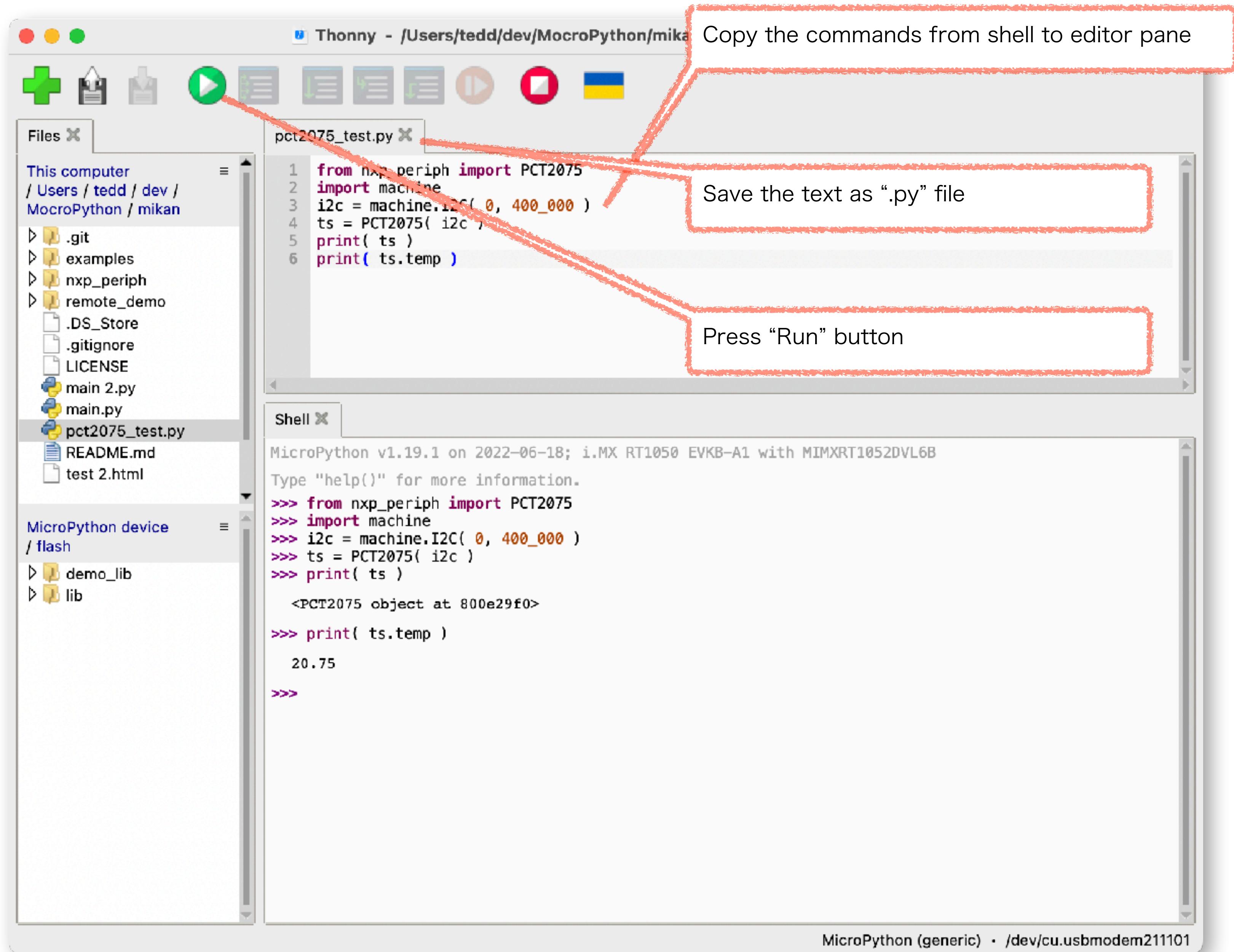
```
>>> print( ts.temp )
```

20.75

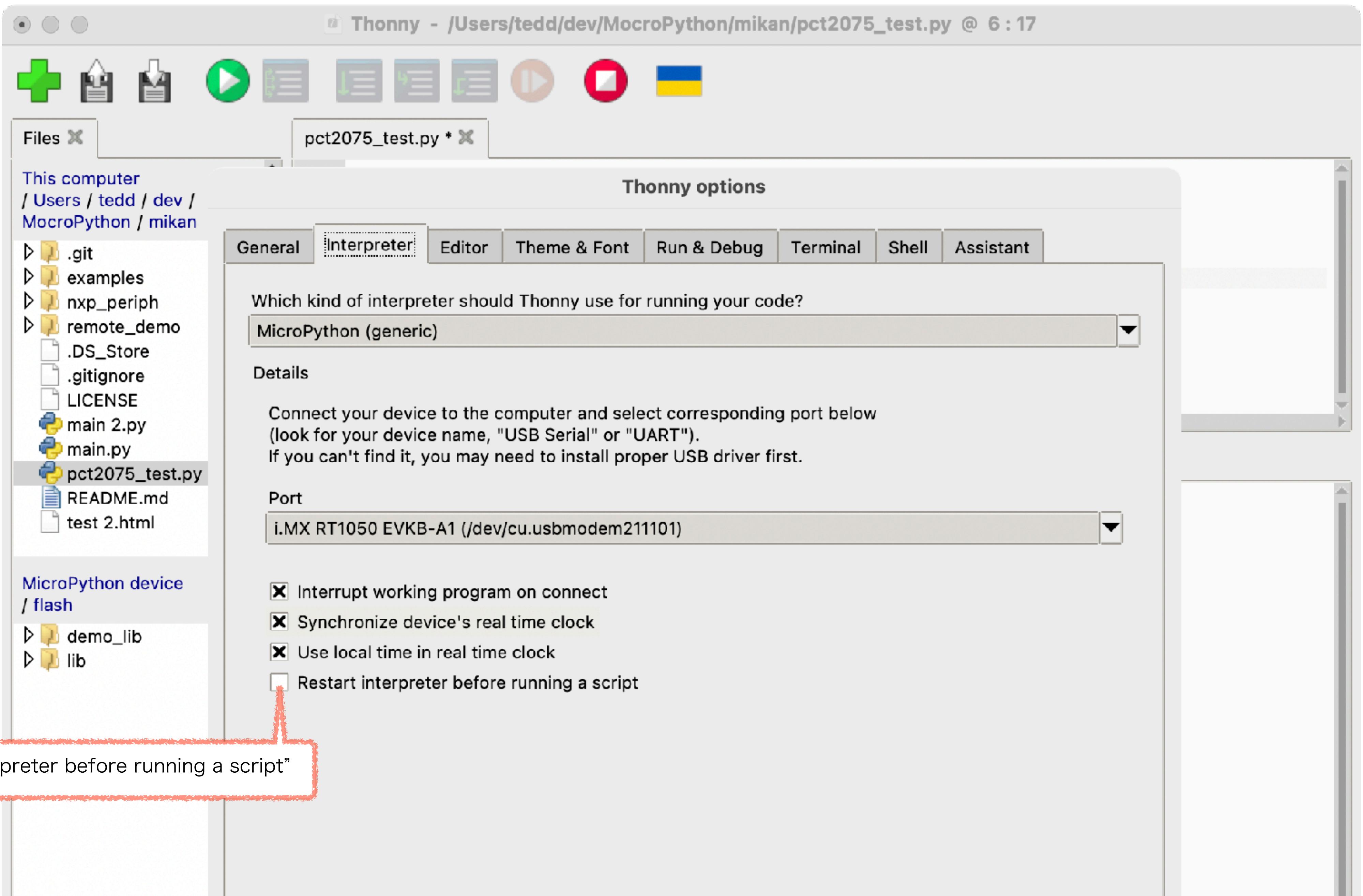
Temperature [°C] will be shown

```
>>>
```

Basic mikan operation



Basic mikan operation



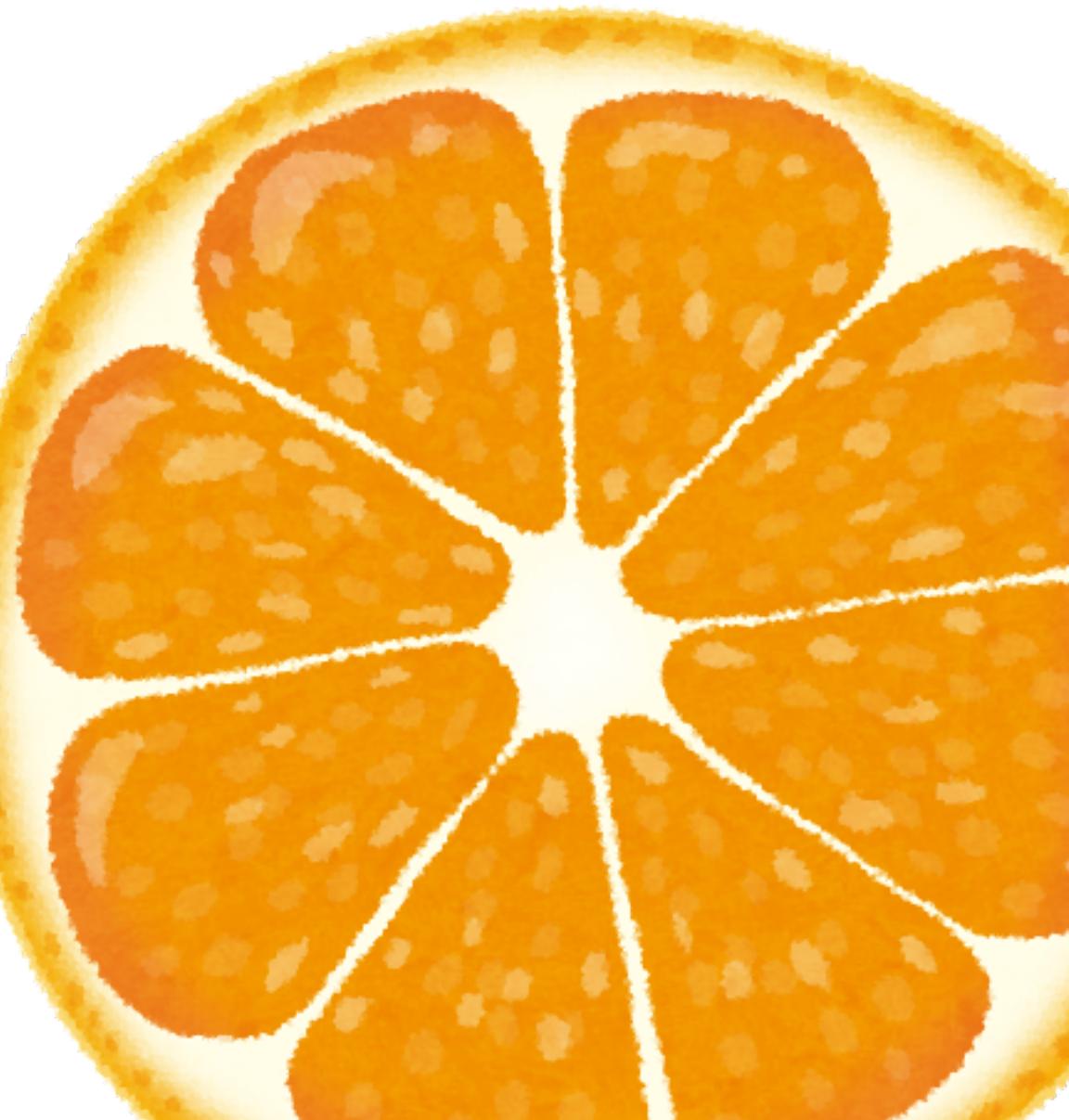
Uncheck: "Restart interpreter before running a script"

```
examples  
▶ nxp_periph  
▶ remote_demo  
.DS_Store  
.gitignore  
.LICENSE
```

Attribute (methods and properties) can be shown by dir()

```
main.py  
pct2075_test.py  
README.md  
test 2.html  
  
MicroPython device      ≡  
/ flash  
▶ demo_lib  
▶ lib  
  
=> run -c $EDITOR_CONTENT  
  
=> >>> print( ts.temp )  
  
<PCT2075 object at 800eb4f0>  
23.0  
  
>>> dir( ts )  
  
['__class__', '__init__', '__module__', '__qualname__', 'read', 'send', '__dict__', 'dump', 'info', 'temp', 'REG_NAME', 'write_registers', 'read_registers', 'bit_operation', 'DEFAULT_ADDR', 'show_reg', 'dump_reg', 'dev_access', '__adr', 'ignore_fail', '__if', '__ai', 'live', 'ping', 'receive', 'heater', '__read', 'reg_access', 'REG_ACC', 'temp_setting', '__value_setting', 'REG_LEN']  
  
>>> ts.info()  
  
'an instance of PCT2075, target address 0x48 (0x90)'  
  
>>> ts.dump()  
  
[5888, 0, 19200, 20480, 1]  
  
>>> ts.REG_NAME  
  
('Temp', 'Conf', 'Thyst', 'Tos', 'Tidle')  
  
>>> ts.read()  
  
23.0  
  
>>> ts.temp  
  
23.0  
  
>>> |
```

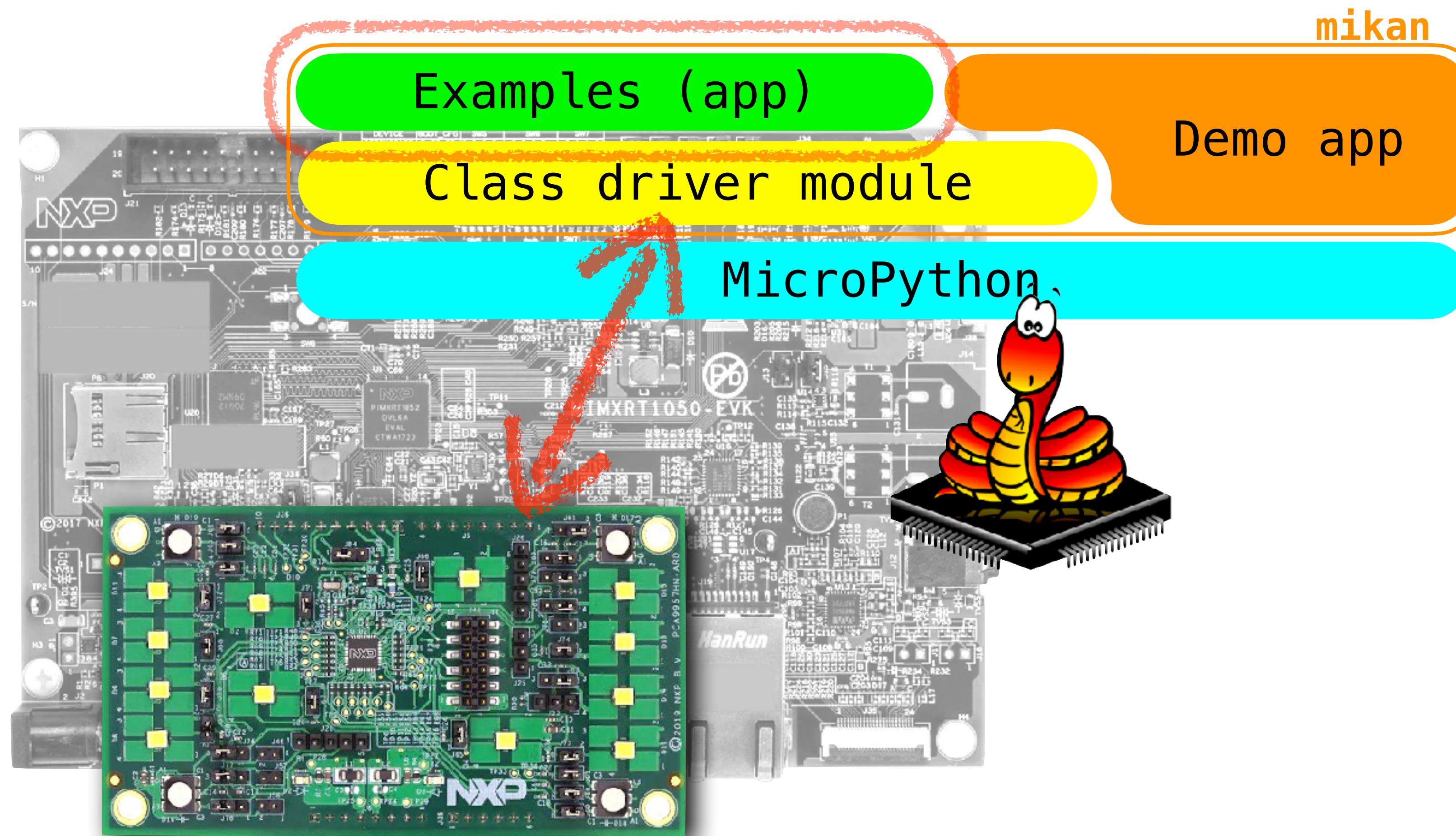
examples/
mikan



examples?

Sample code of mikan class drivers.

It can operate single/multiple target device(s) from MicroPython shell



examples?

Sample code and its features

Category	File name	Feature	Target
LED	LED_controller.py	Simple sample: making an LED_controller instance and how PWM can be controlled	PCA9955B, PCA9956B, PCA9957, PCA9632
LED	LED_gradation_ctrl.py	Gradation control (hardware) feature demo	PCA9955B, PCA9957
LED	LED_instance.py	Using another class to abstract LED controllers	PCA9955B, PCA9956B, PCA9957
LED	LED_demo.py	Showing idea to use 'LED class' to manage LED and white LED individually	PCA9955B, PCA9956B, PCA9957, PCA9632
LED	LED_demo_dual_om13321.py	Showing idea to use 'LED class' to manage multiple LED controller devices	PCA9956B
RTC	RTC_demo_PCF2131_ARD.py	Operate a PCF2131 through MicroPython's machine.RTC equivalent APIs. Using 2 interrupt lines	PCF2131
RTC	RTC_demo_PCF85063AT_ARD.py	Operate a PCF85063 through MicroPython's machine.RTC equivalent APIs.	PCF85063
Temp sensor	temp_sensor_simple.py	Very simple sample to operate a temp sensor	LM75B, PCT2075
Temp sensor	temp_sensor_demo_PCT2075DP_ARB.py	Operate with interrupt and heater-resister on ARD board	PCT2075
Temp sensor	temp_sensor_P3T1085.py	Similar to "temp_sensor_simple.py" but different I2C pin assign.	P3T1085
Temp sensor	temp_sensor_demo_P3T1085UK_ARB.py	Similar to "temp_sensor_demo_PCT2075DP_ARB.py" but no heater operation	P3T1085
GPIO expander	GPIO_demo.py	Operation sample of a PCA9555 API	PCA9555
GPIO expander	GPIO_demo_PCAL6xxxA-ARD.py	Operation sample of a PCAL6xxx ARD board. Using interrupt	PCAL6408, PCAL6416, PCAL6524, PCAL6534
LCD	LCD_demo_PCA8561AHN-ARD.py	Shows direct ON/OFF of segments and using putc(), puts() methods	PCA8561
Protocol bridge	protocol_bridge_SC16IS7xx.py	Operate an I ² C/SPI to UART protocol bridge through MicroPython's machine.UART equivalent APIs. SC16IS7xx	
Protocol bridge	protocol_bridge_SC18IS606_with_AT25010.py	Operate an I ² C to SPI protocol bridge through MicroPython's machine.SPI equivalent APIs. AT25010 as an SPI target	SC18IS606
Stepper motor	stepper_motor_simple.py	Operating stepping motor with simple API	PCA9629A
Stepper motor	stepper_motor_5_motors.py	Operating 5 instances of PCA9629A class	PCA9629A
Accelerometer	accelerometer.py	Get X, Y and Z gravitational acceleration [G] values	FXOS8700, FXLS8974
Magnetometer		Get X, Y and Z geomagnetism [nT] values	FXOS8700

examples?

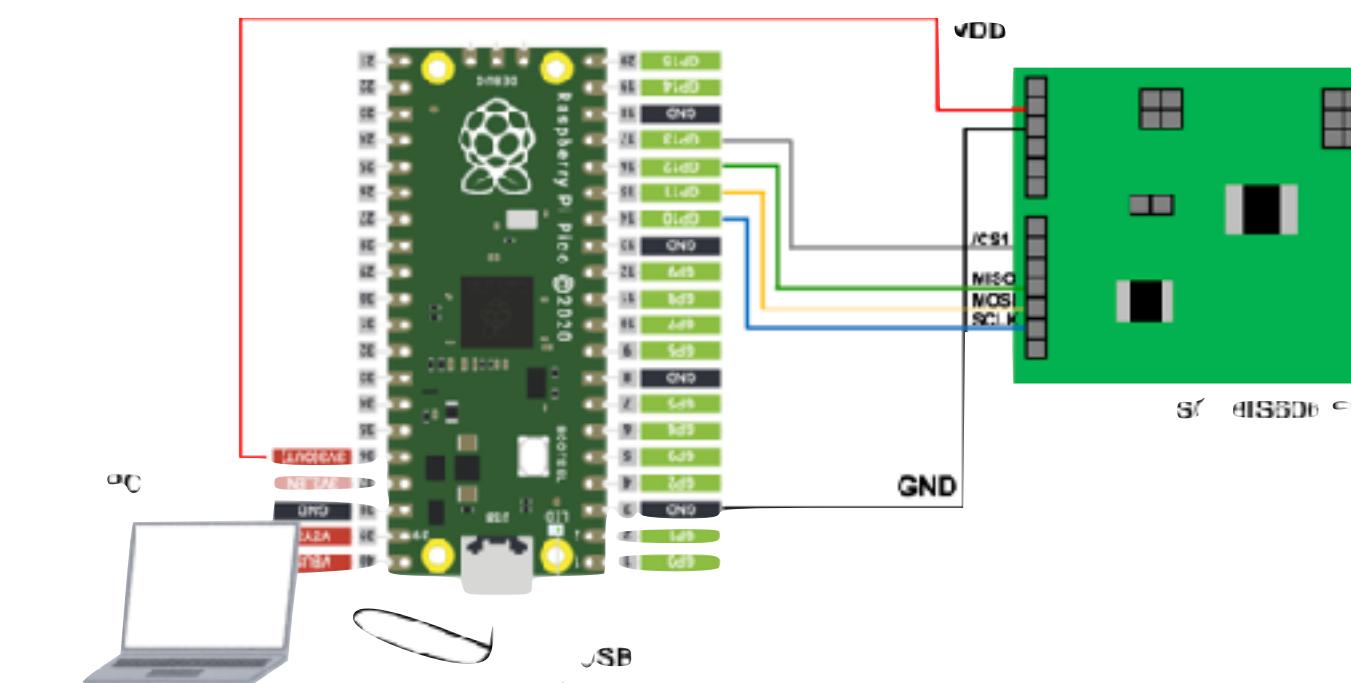
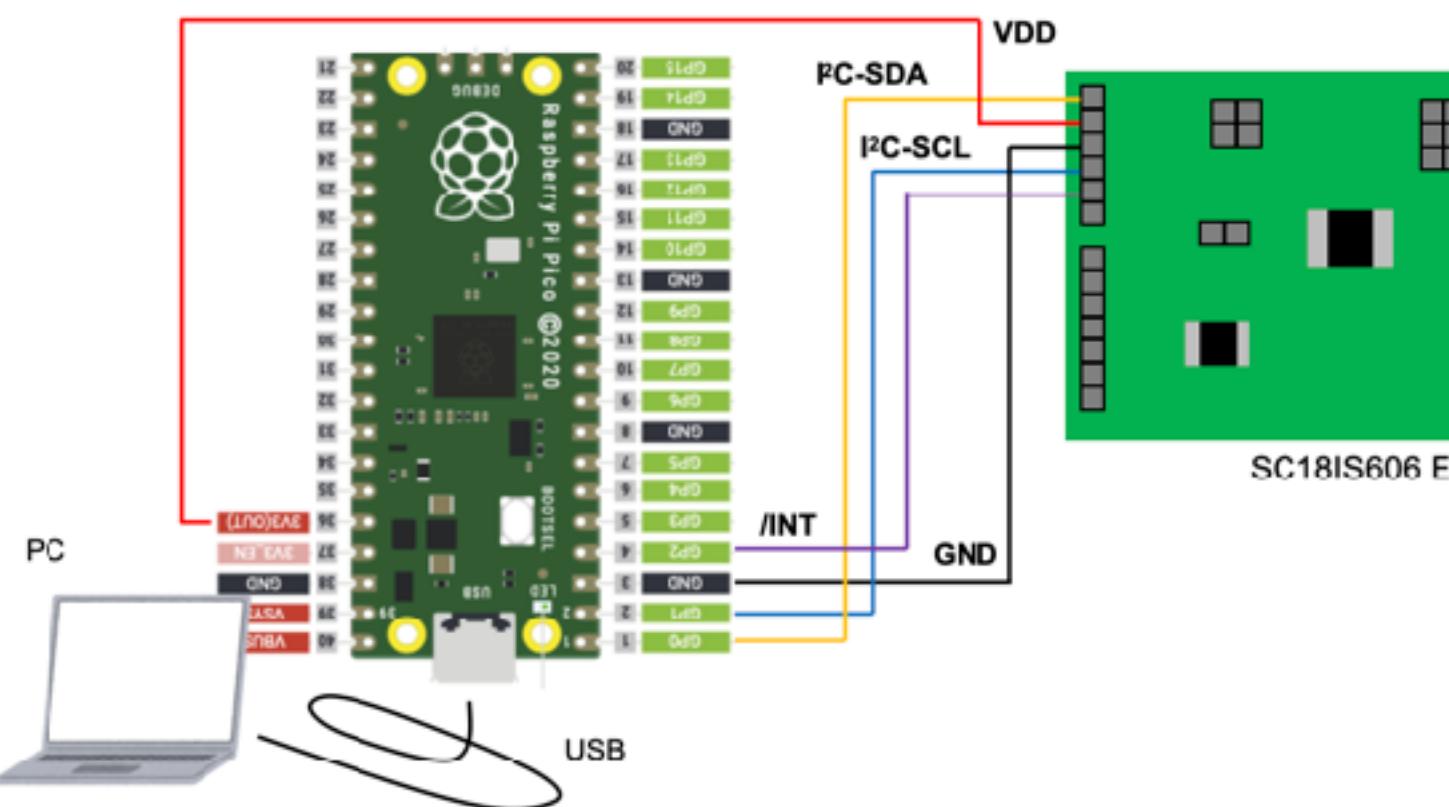
Compatibility: MicroPython abstracts MCU in very high level

If the code is needed to run on different MCU ...

1. Install mikan into 'lib' folder under `sys.path`

<https://github.com/teddokano/mikan#the-steps>

2. Connect hardware

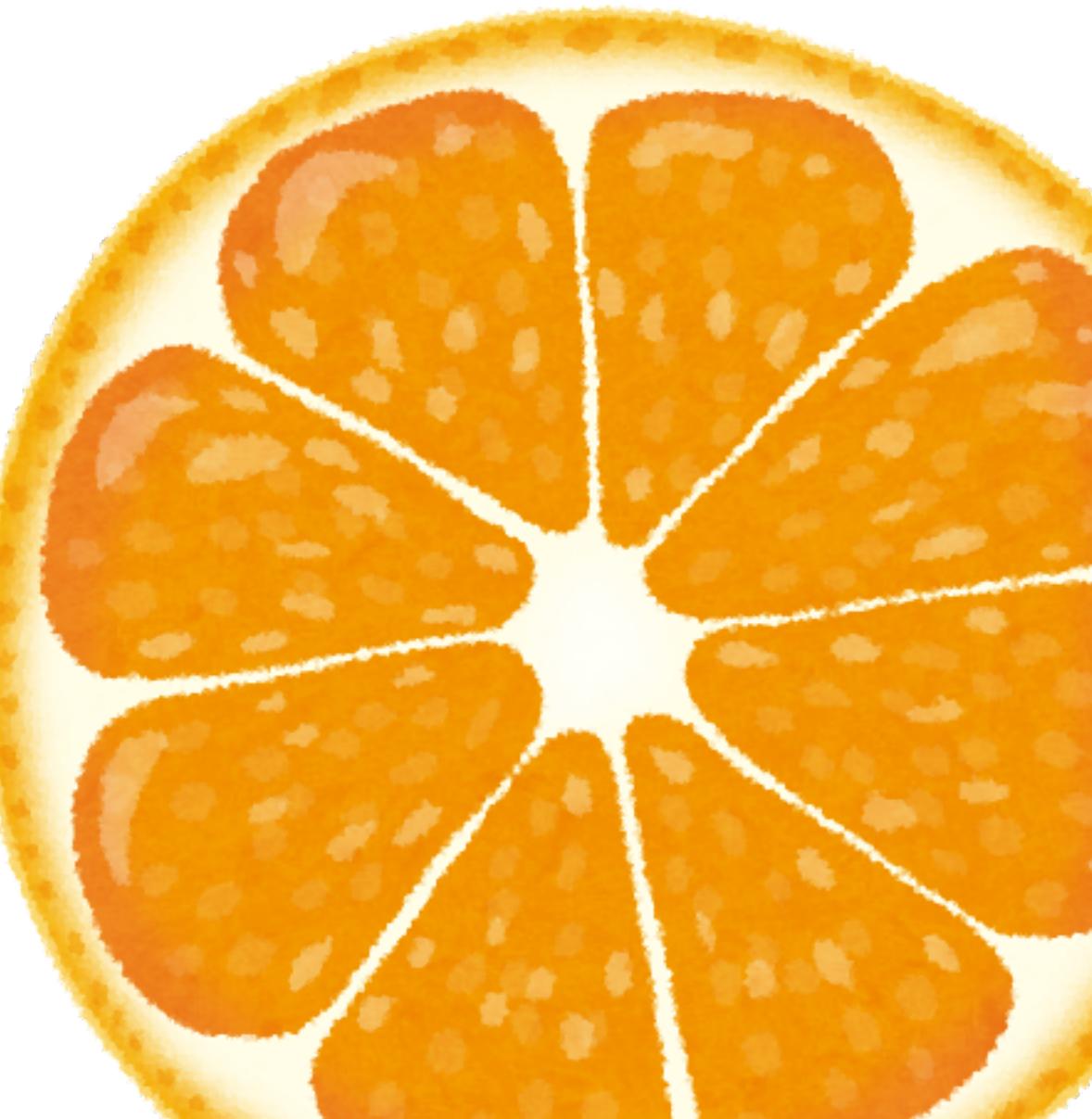


3. Put code to absorb hardware difference

```
if "Raspberry Pi Pico" in os.uname().machine:  
    i2c = I2C( 0, sda = Pin(0), scl = Pin(1), freq = I2C_FREQ )  
    spi = SPI( 1, SPI_FREQ, sck = Pin( 10 ), mosi = Pin( 11 ), miso = Pin( 12 ) )  
elif "MIMXRT" in os.uname().machine:  
    i2c = I2C( 0, freq = I2C_FREQ )  
    spi = SPI( 0, SPI_FREQ, cs = 0 )
```

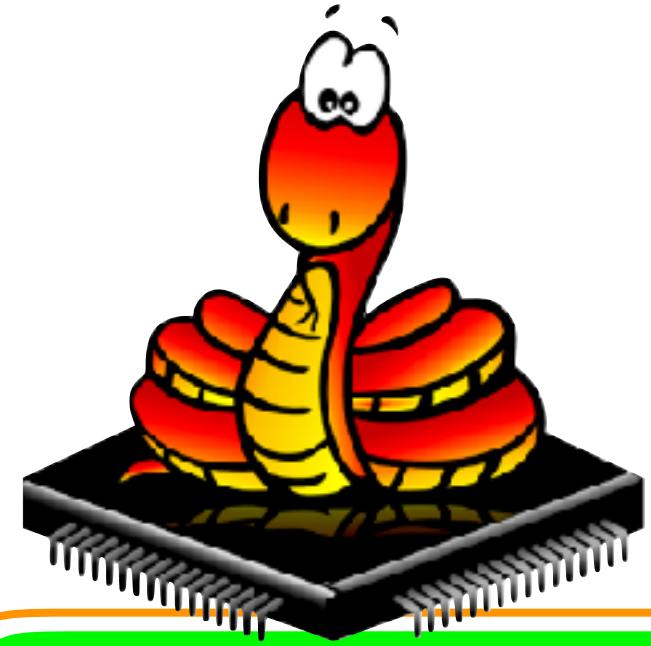
4. It should work!

remote_demo/
mikan



remote_demo?

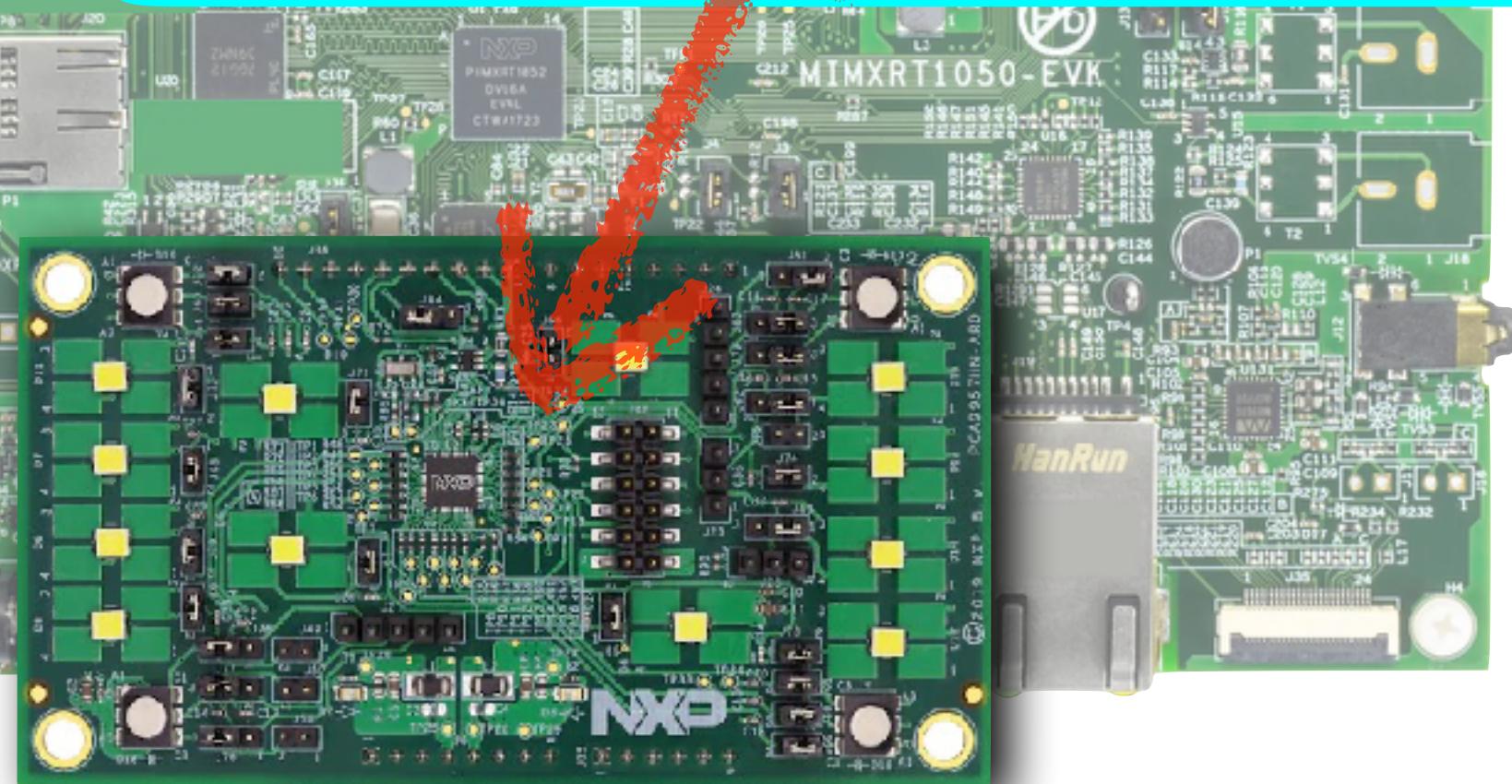
A demo software using mikan class drivers.
It can operate multiple target devices from web-browsers



Examples (app)

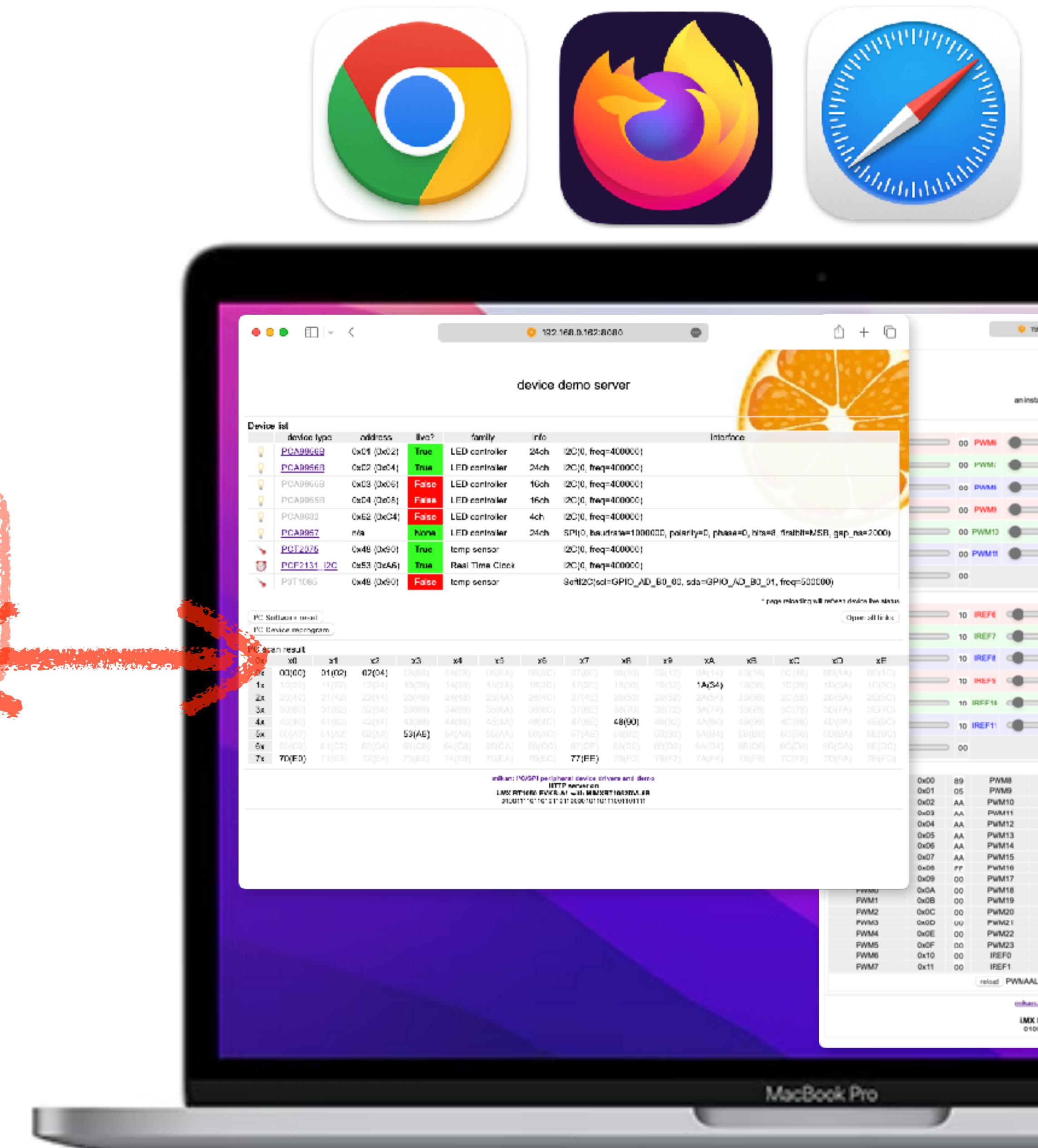
class driver module

MicroPython



mikan

Demo app

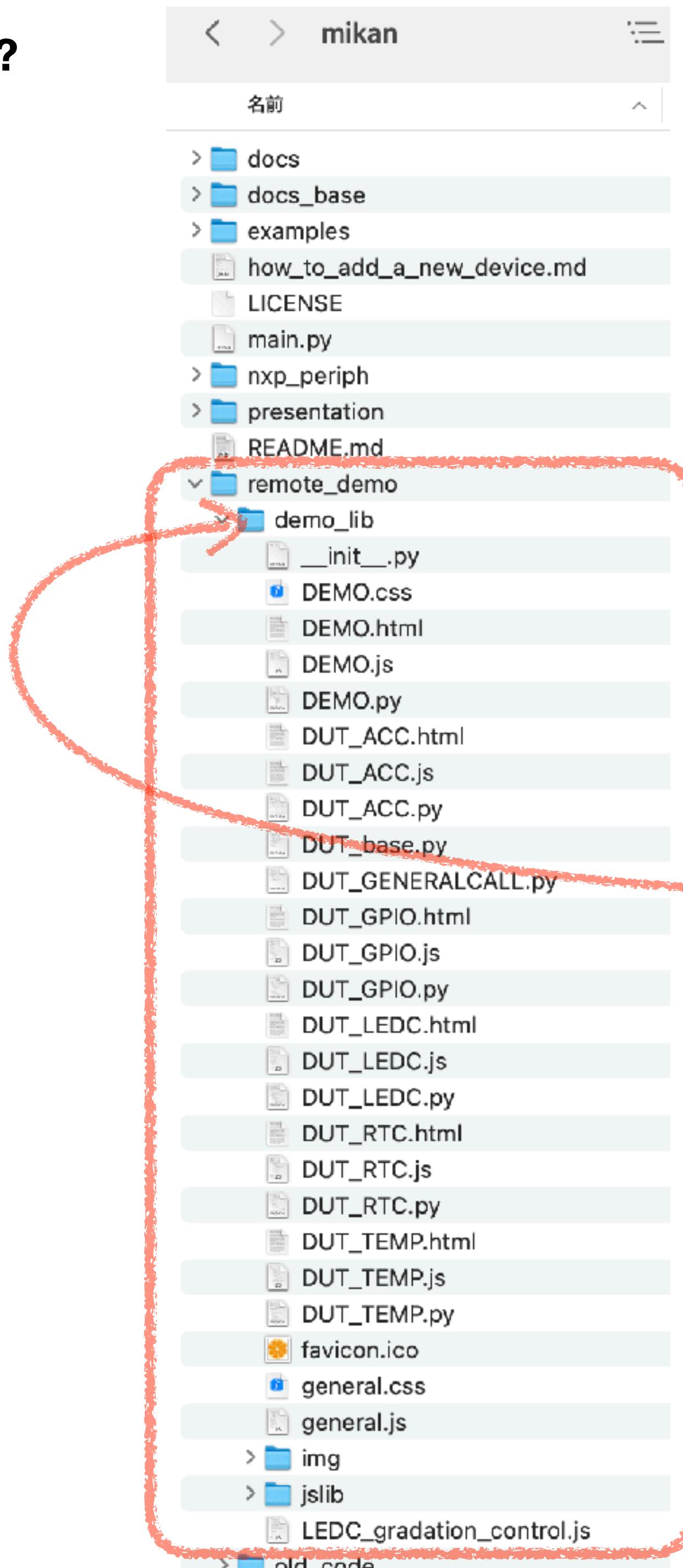


remote_demo?

Supported devices

Category	Target	Feature	mikan class driver	remote_demo support
LED controller	PCA9632	4ch, I ² C	✓	yes (no ARD available)
LED controller	PCA9955B	16ch, I ² C	✓	yes
LED controller	PCA9956B	24ch, I ² C	✓	yes (no ARD available)
LED controller	PCA9957	24ch, SPI	✓	yes
RTC	PCF2131	I ² C & SPI	✓	yes
RTC	PCF85063A		✓	yes
Temp sensor	PCT2075		✓	yes
Temp sensor	P3T1085		✓	yes
GPIO expander	PCA9554	8bit	✓	no (no ARD available)
GPIO expander	PCA9555	16bit	✓	no (no ARD available)
GPIO expander	PCAL6408	8bit	✓	yes
GPIO expander	PCAL6416	16bit	✓	yes
GPIO expander	PCAL6524	24bit	✓	yes
GPIO expander	PCAL6534	34bit	✓	yes
LCD driver	PCA8561		✓	no
Protocol bridge	SC16IS7xx	I ² C/SPI → UART	✓	no (no ARD available)
Protocol bridge	SC18IS606	I ² C → SPI	✓	no (no ARD available)
Stepper motor controller	PCA9629A		✓	no (no ARD available)
Accelerometer	FXOS8700	6 axis	✓	yes
Accelerometer	FXLS8974	3 axis	✓	yes

remote_demo?



remote_demo is a sample code of operating mikan class driver. The I²C/SPI devices can be operated through client application on web browser.

'demo_lib' folder is needed to be copied in path in MicroPython execute environment

remote_demo?

Steps to run

1.
'nxp_periph/' needed to be in '/
flash/lib/'

2.
Copy 'remote_demo/demo_lib' under
'/flash/'

1 and 2 are install steps. Not necessary to
do every time to run

3.
Choose (double click) 'remote_demo/
DEMO.py' under '/flash/'.
Source code will be shown in upper-
right pane

The screenshot shows the Thonny IDE interface with the following details:

- File Explorer (Left):** Shows the project structure:
 - This computer: /Users/tedd/dev/MicroPython/mikan/remote_demo
 - demo_lib
 - old_code
 - option_file
 - .DS_Store
 - DEMO.py
- MicroPython device (Bottom Left):** Shows the flash memory structure:
 - /flash
 - demo_lib
 - lib
 - nxp_periph
- Code Editor (Top Right):** Displays the content of DEMO.py:

```
1 import network
2 import ujson
3 import machine
4 import os
5 import ure
6 import gc
7 try:
8     import usocket as socket
9 except:
10     import socket
11 from nxp_periph import PCA9956B, PCA9955B, PCA9632, PCA9957, LED
12 from nxp_periph import PCT2075, LM75L, P3T1085
13 from nxp_periph import PCF2131, PCF8506
14 from nxp_periph import PCAL6408, PCAL6416, PCAL6524, PCAL6534
15 from nxp_periph import i2c_fullscan
16 from demo_lib import DUT_LED, DUT_TEMP, DUT_RTC, DUT_GPIO
17 from demo_lib import DUT_GENERALCALL, General_call
18 from demo_lib import DUT_hase
```
- Shell (Bottom Right):** Displays the MicroPython shell output:

```
MicroPython v1.19.1 on 2022-06-18; i.MX RT1050 EVKB-A1 with MIMXRT1050
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
remote device demo
    http server is started working on i.MX RT1050 EVKB-A1 with MIMXRT1050
I2C error: NACK returned 3 times from PCA9955B, address 0x03 (0x90)
I2C error: NACK returned 3 times from PCA9955B, address 0x04 (0x90)
I2C error: NACK returned 3 times from PCA9632, address 0x62 (0x90)
PCA9957HN_ARD setting done.
I2C error: NACK returned 3 times from P3T1085, address 0x48 (0x90)
NOT responding
starting network
ethernet port 0 is activated
Listening, connect your browser to http://192.168.0.162:8080/
```

4.
Press "Run" button

5.
Click on URL

remote_demo?

UI on browser: Device list page

The screenshot shows a web browser displaying the 'device demo server' at 192.168.0.162:8080. The interface includes:

- Device status**: Based on ACK/NACK response (highlighted by a red box).
- Interface information**: I²C / SPI (highlighted by a red box).
- Device list**: A table showing device details like device type, address, live status, family, info, and interface. One row for PCA9956B at address 0x02 is marked as 'True'. Another row for PCA9955B at address 0x06 is marked as 'False'. A note says 'List can be modified in source code' (highlighted by a red box).
- I²C scan result**: A table showing I²C scan results for addresses 0x to 7F. A note says 'General call commands (for I²C ID=0 only)' (highlighted by a red box).

Below the tables are buttons for 'I²C Software reset' and 'I²C Device reprogram', and a link 'Open all links'. The footer includes the URL 'mikan: I²C/SPI peripheral device drivers and demo', the HTTP server information 'HTTP server on i.MX RT1050 EVKB-A1 with MIMXRT1052DVL6B', and a long MAC address '01001110110101011000010110111001101111'.

	device type	address	live?	family	info	interface
1	PCA9956B	0x01 (0x02)	True	LED controller	24ch	I ² C(0, freq=400000)
1	PCA9956B	0x02 (0x04)	True	LED controller	24ch	I ² C(0, freq=400000)
1	PCA9955B	0x03 (0x06)	False	LED controller	16ch	I ² C(0, freq=400000)
1	PCA9955B	0x04 (0x08)	False	LED controller	16ch	I ² C(0, freq=400000)
1	PCA9632	0x62 (0xC4)	False	LED controller	4ch	I ² C(0, freq=400000)
1	PCA9957	n/a	None	LED controller	24ch	SPI(0, baudrate=1000000, polarity=0, phase=0, bits=8, firstbit=MSB, gap_ns=2000)
1	PCT2075	0x48 (0x90)	True	temp sensor		I ² C(0, freq=400000)
1	PCF2131_I²C	0x53 (0xA6)	True	Real Time Clock		I ² C(0, freq=400000)
1	P3T1085	0x48 (0x90)	False	temp sensor		SoftI ² C(scl=GPIO_AD_B0_00, sda=GPIO_AD_B0_01, freq=500000)

* page reloading will refresh device live status

[Open all links](#)

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	00(00)	01(02)	02(04)	03(06)	04(08)	05(0A)	06(0C)	07(0E)	08(10)	09(12)	0A(14)	0B(16)	0C(18)	0D(1A)	0E(1C)	0F(1E)
1x	10(20)	11(22)	12(24)	13(26)	14(28)	15(2A)	16(2C)	17(2E)	18(30)	19(32)	1A(34)	1B(36)	1C(38)	1D(3A)	1E(3C)	1F(3E)
2x	20(40)	21(42)	22(44)	23(46)	24(48)	25(4A)	26(4C)	27(4E)	28(50)	29(52)	2A(54)	2B(56)	2C(58)	2D(5A)	2E(5C)	2F(5E)
3x	30(60)	31(62)	32(64)	33(66)	34(68)	35(6A)	36(6C)	37(6E)	38(70)	39(72)	3A(74)	3B(76)	3C(78)	3D(7A)	3E(7C)	3F(7E)
4x	40(80)	41(82)	42(84)	43(86)	44(88)	45(8A)	46(8C)	47(8E)	48(90)	49(92)	4A(94)	4B(96)	4C(98)	4D(9A)	4E(9C)	4F(9E)
5x	50(A0)	51(A2)	52(A4)	53(A6)	54(A8)	55(AA)	56(AC)	57(AE)	58(B0)	59(B2)	5A(B4)	5B(B6)	5C(B8)	5D(BA)	5E(BC)	5F(BE)
6x	60(C0)	61(C2)	62(C4)	63(C6)	64(C8)	65(CA)	66(CC)	67(CE)	68(D0)	69(D2)	6A(D4)	6B(D6)	6C(D8)	6D(DA)	6E(DC)	6F(DE)
7x	70(E0)	71(E2)	72(E4)	73(E6)	74(E8)	75(EA)	76(EC)	77(EE)	78(F0)	79(F2)	7A(F4)	7B(F6)	7C(F8)	7D(FA)	7E(FC)	7F(FE)

mikan: I²C/SPI peripheral device drivers and demo
HTTP server on
i.MX RT1050 EVKB-A1 with MIMXRT1052DVL6B
01001110110101011000010110111001101111

remote_demo?

UI on browser: example of device page

The screenshot shows a web-based user interface for the PCA9956B device. At the top, a header bar includes links for 'net', 'news', 'weather', 'bike', 'tv', 'I2C(0x02)', and a search bar. A red box highlights the device name 'PCA9956B server' and its address 'an Instance of PCA9956B, target address 0x01 (0x02)'. Another red box highlights the 'Device info' section.

Below the header, the main content area is divided into sections:

- PWM registers:** A grid of 24 sliders labeled PWM0 through PWM23 and PWMALL. A red box highlights the first row of sliders. The first column of labels (PWM0, PWM1, PWM2, PWM3, PWM4, PWM5) is also highlighted.
- IREF registers:** A grid of 24 sliders labeled IREF0 through IREF23 and IREFALL. A red box highlights the first row of sliders. The first column of labels (IREF0, IREF1, IREF2, IREF3, IREF4, IREF5) is also highlighted.
- register table:** A table mapping register names to their addresses and values. A red box highlights the first row of the table. The columns include MODE1, MODE2, LEDOUT0, LEDOUT1, LEDOUT2, LEDOUT3, LEDOUT4, LEDOUT5, GRPPWM, GRPFREQ, PWM0, PWM1, PWM2, PWM3, PWM4, PWM5, PWM8, PWM9, PWM10, PWM11, PWM12, PWM13, PWM14, PWM15, PWM16, PWM17, PWM18, PWM19, PWM20, PWM21, PWM22, PWM23, IREF3, IREF4, IREF5, IREF6, IREF7, IREF8, IREF9, IREF10, IREF11, IREF12, IREF13, IREF14, IREF15, IREF16, IREF17, IREF18, IREF19, IREF20, IREF21, IREF22, IREF23, OFFSET, SUBADR1, SUBADR2, SUBADR3, ALLCALLADR, PWMALL, IREFALL, and EFLAG0-EFLAG5. A red box highlights the first row of the table.
- Type-in to write into register:** A text input field at the bottom of the register table table.
- Force to reload register values:** A button at the bottom left of the register table table.

At the very bottom, a note states: "reload PWMAALL and IREFALL are write-only register. ignore loaded value". The footer includes links for '時刻表', 'Google C++ Style Guide', '高域特性劣化のメ.. DN Japan', and a logo for 'mikan: I2C/SPI peripheral device drivers and demo'.

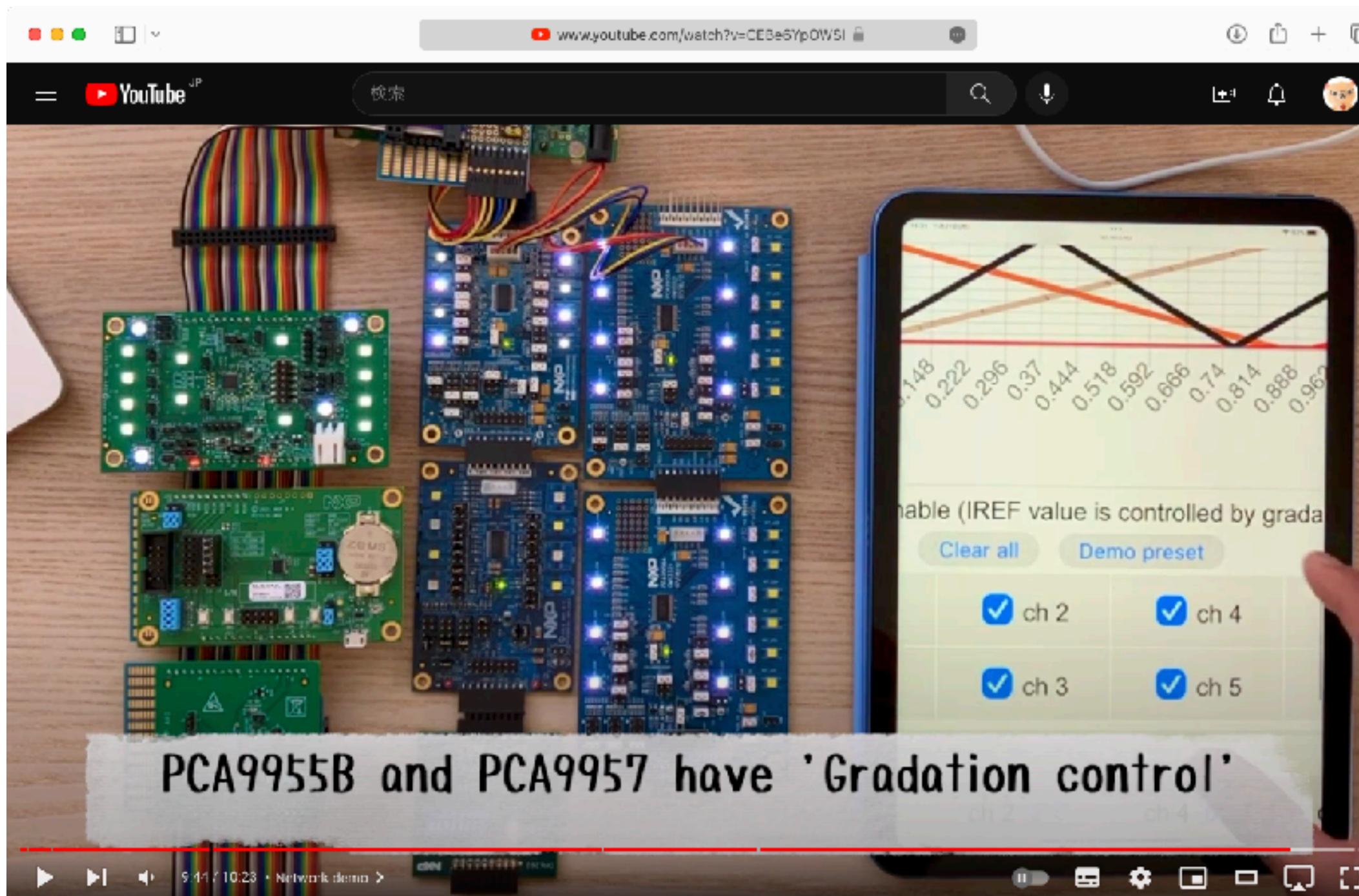
	0x00	89	PWM8	0x12	00	IREF3	0x25	10	IREF21	0x36	10
MODE1											
MODE2	0x01	05	PWM9	0x13	00	IREF4	0x26	10	IREF22	0x37	10
LEDOUT0	0x02	AA	PWM10	0x14	00	IREF5	0x27	10	IREF23	0x38	10
LEDOUT1	0x03	AA	PWM11	0x15	00	IREF6	0x28	10	OFFSET	0x39	10
LEDOUT2	0x04	AA	PWM12	0x16	00	IREF7	0x29	10	SUBADR1	0x3A	08
LEDOUT3	0x05	AA	PWM13	0x17	00	IREF8	0x2A	10	SUBADR2	0x3B	EE
LEDOUT4	0x06	AA	PWM14	0x18	00	IREF9	0x2B	10	SUBADR3	0x3C	EE
LEDOUT5	0x07	AA	PWM15	0x19	00	IREF10	0x2C	10	ALLCALLADR	0x3D	EE
GRPPWM	0x08	FF	PWM16	0x1A	00	IREF11	0x2D	10	PWMALL	0x3E	E0
GRPFREQ	0x09	00	PWM17	0x1B	00	IREF12	0x2E	10	IREFALL	0x3F	--
PWM0	0xA	00	PWM18	0x1C	00	IREF13	0x2F	10	EFLAG0	0x40	00
PWM1	0xB	00	PWM19	0x1D	00	IREF14	0x30	10	EFLAG1	0x41	00
PWM2	0xC	00	PWM20	0x1E	00	IREF15	0x31	10	EFLAG2	0x42	00
PWM3	0xD	00	PWM21	0x1F	00	IREF16	0x32	10	EFLAG3	0x43	00
PWM4	0xE	00	PWM22	0x20	00	IREF17	0x33	10	EFLAG4	0x44	00
PWM5	0xF	00	PWM23	0x21	00	IREF18	0x34	10	EFLAG5	0x45	00
	0x10	00	IREF0	0x22	10	IREF19	0x35	10			
	0x11	00	IREF1	0x23	10						

remote_demo?

Demo video available

English version

<https://youtu.be/CEBe6YpOWSI>



introduction to mikan (English version)



アナリティクス

動画の閲覧

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

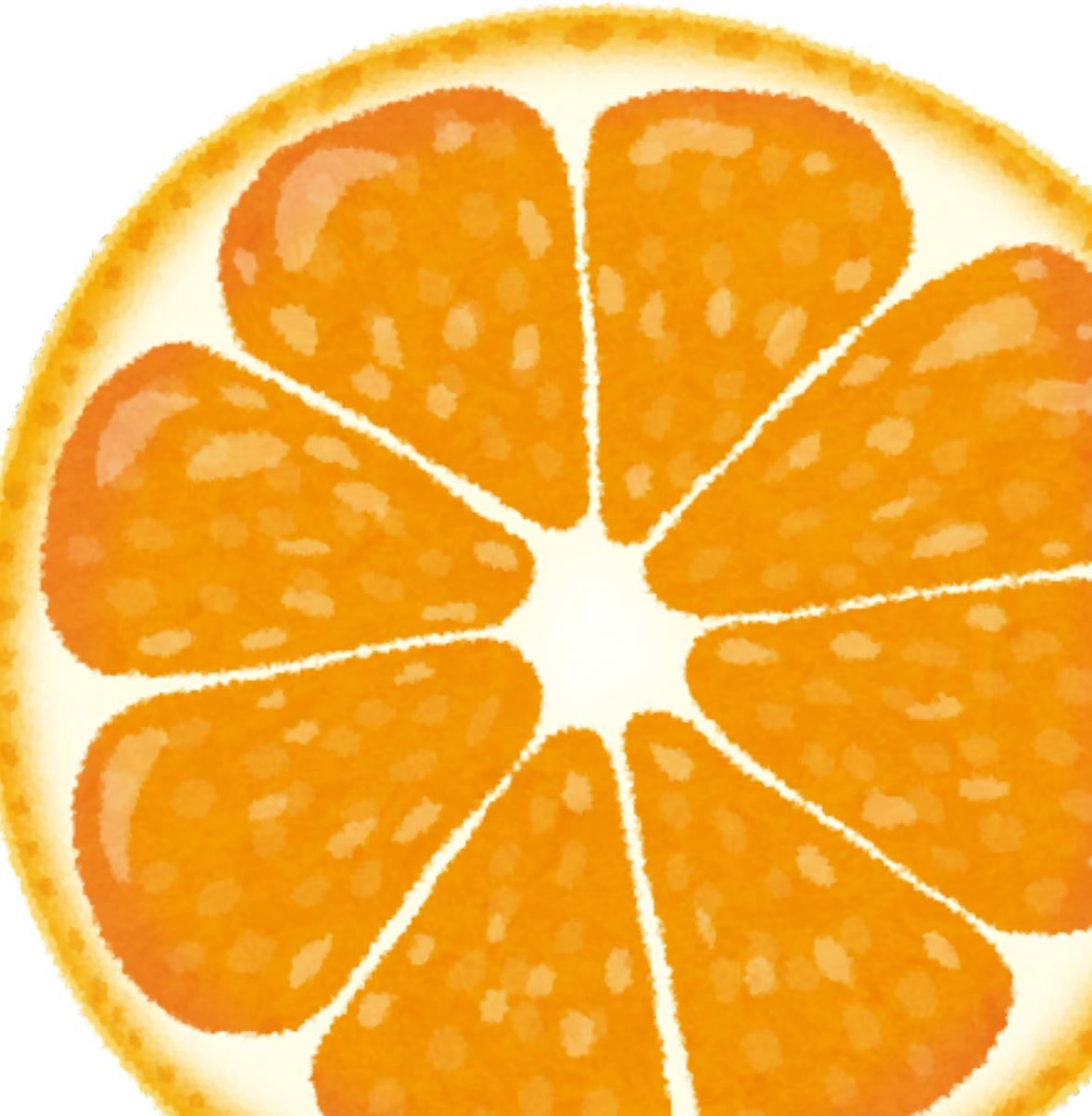
0

0

0

How does it work?

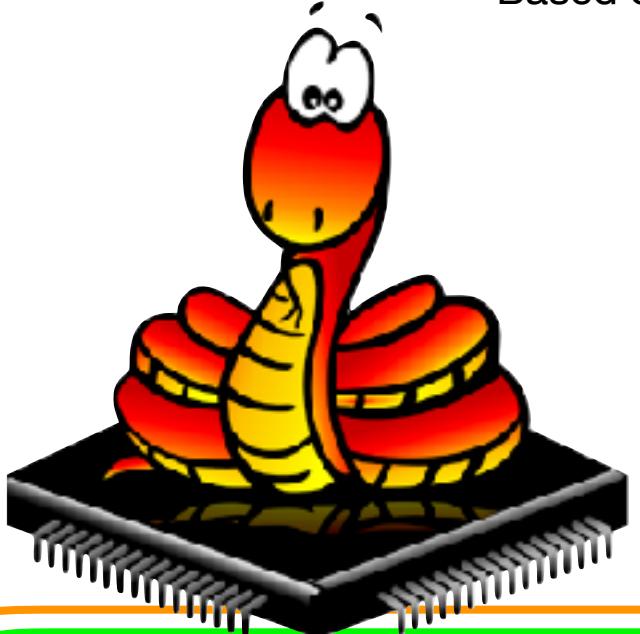
mikan



How does it work? : Applications

Web server application
using 'mikan' class driver
to control devices

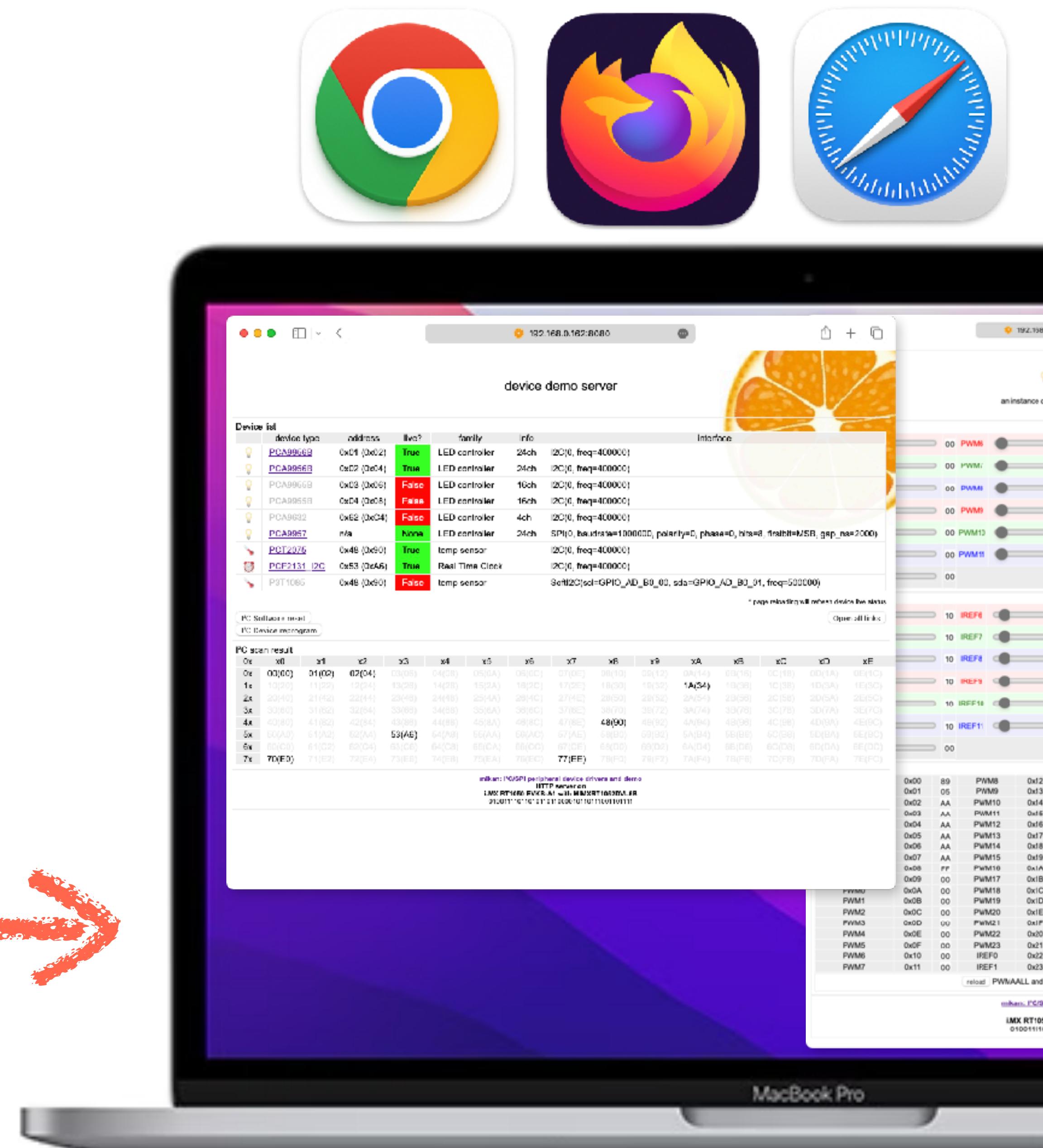
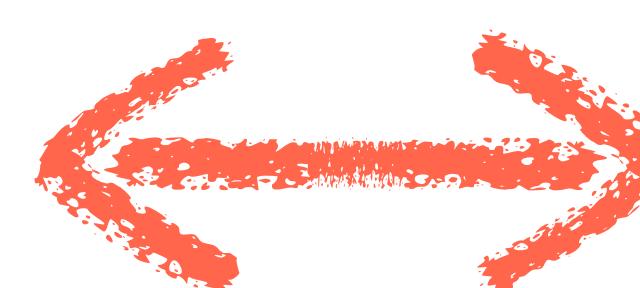
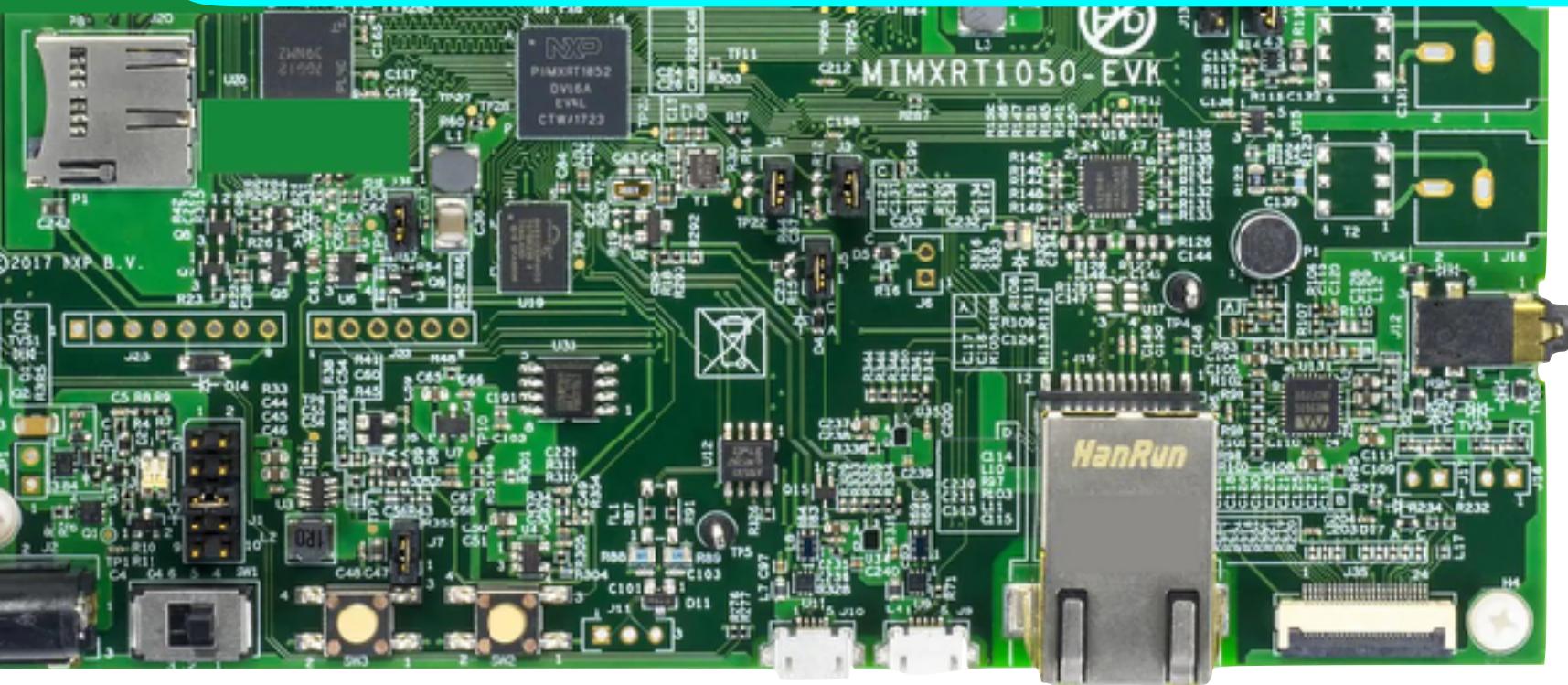
Based on MicroPython [example code](#)



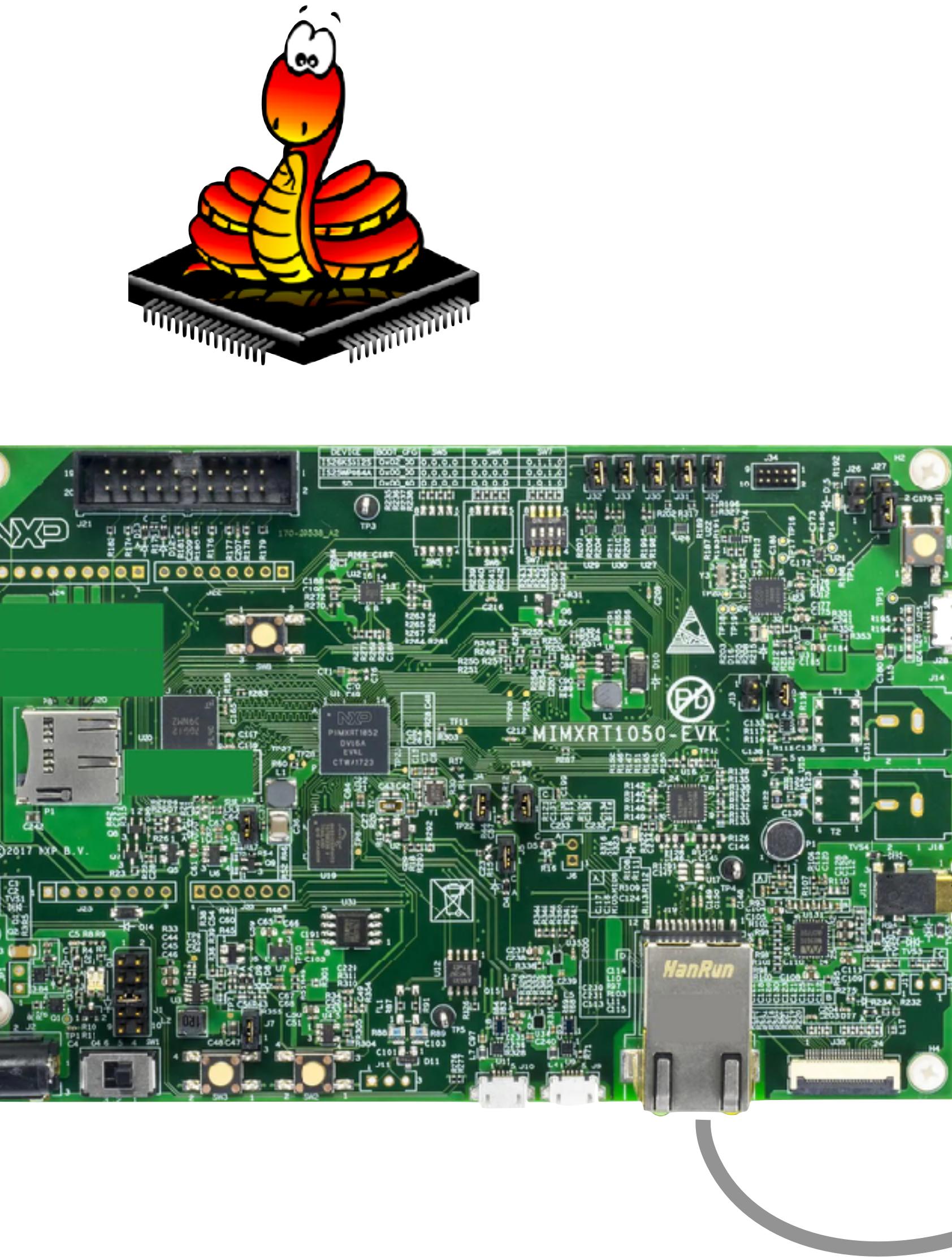
Examples (app)

Class driver module

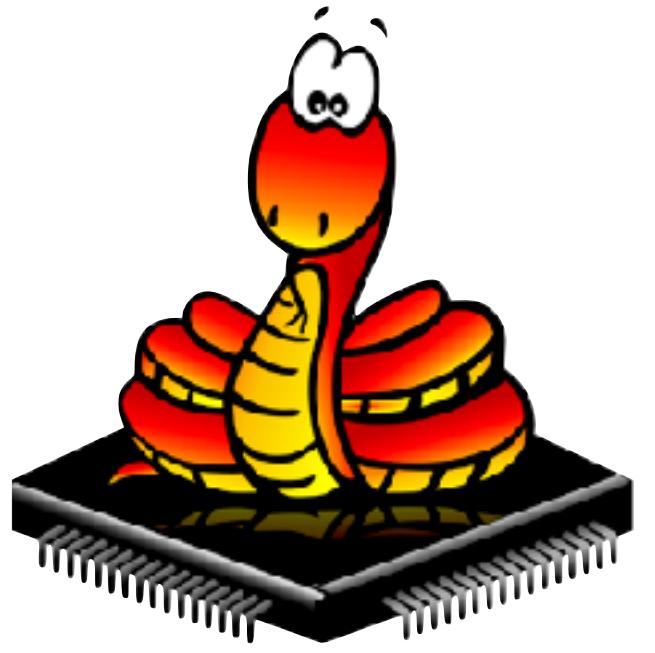
MicroPython



How does it work? : Hardware (using PC with a LAN cable)



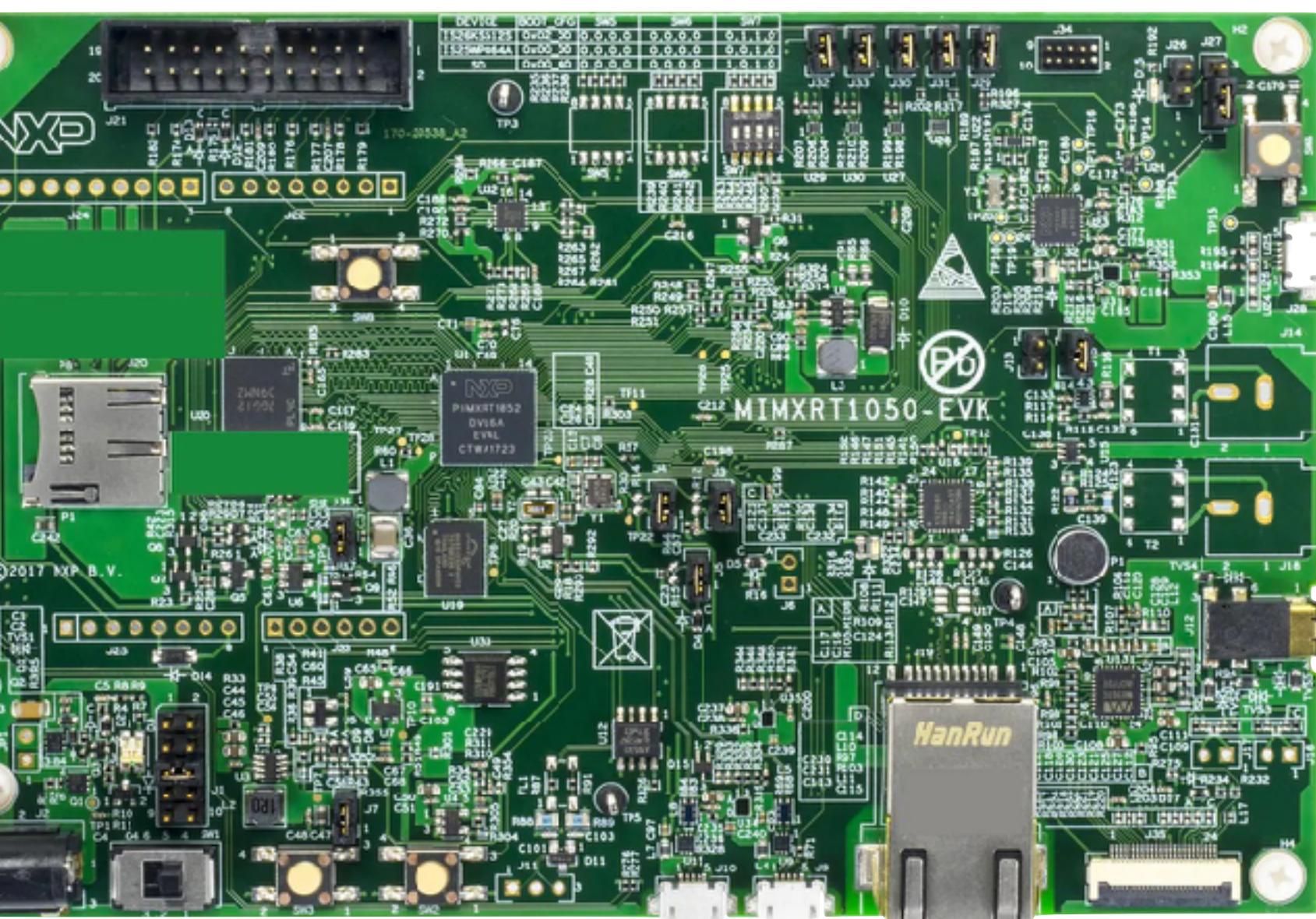
How does it work? : Hardware (using PC/Tablet/Smartphone with a wireless adapter)



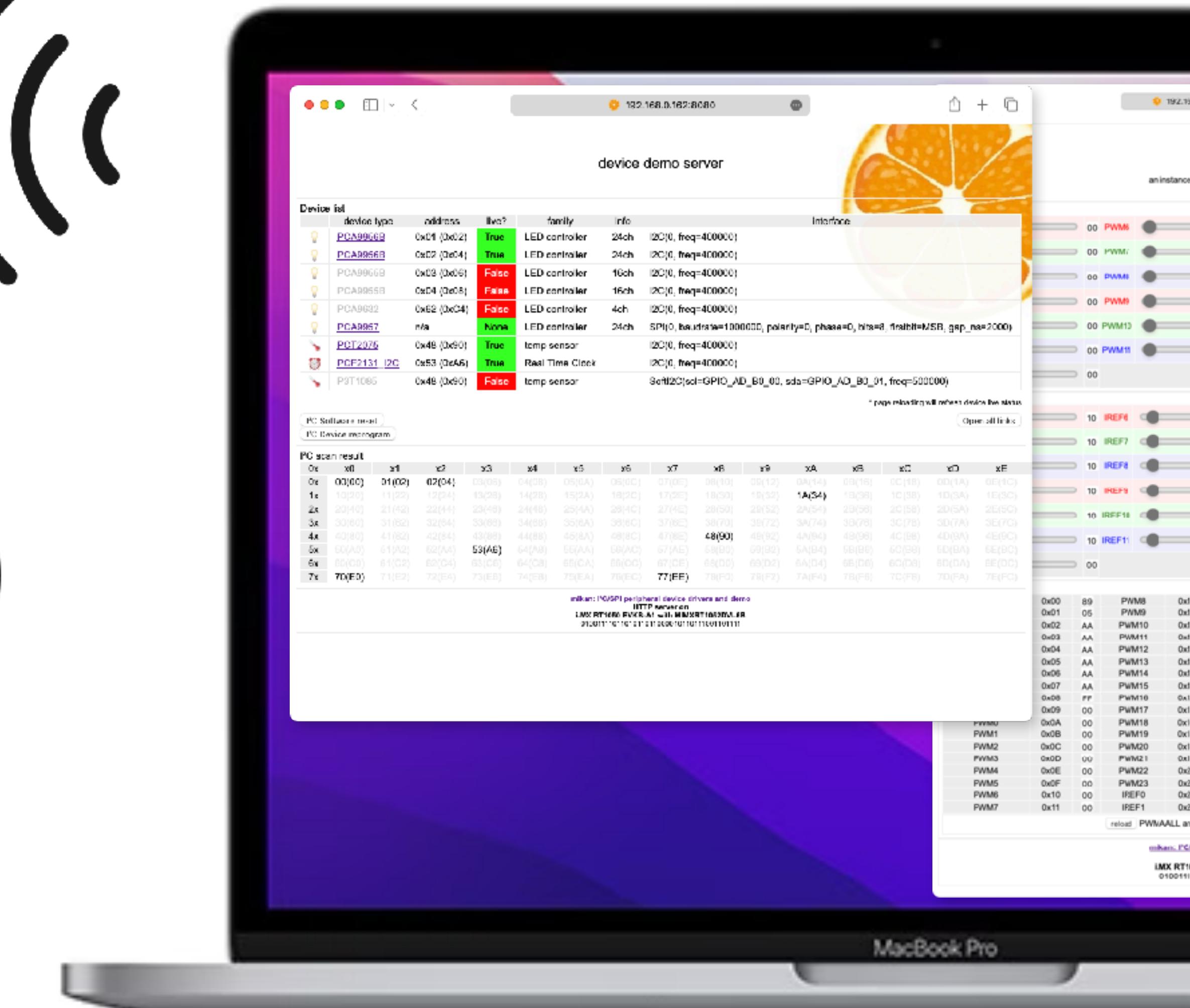
Wireless adapter

TL-WR902AC

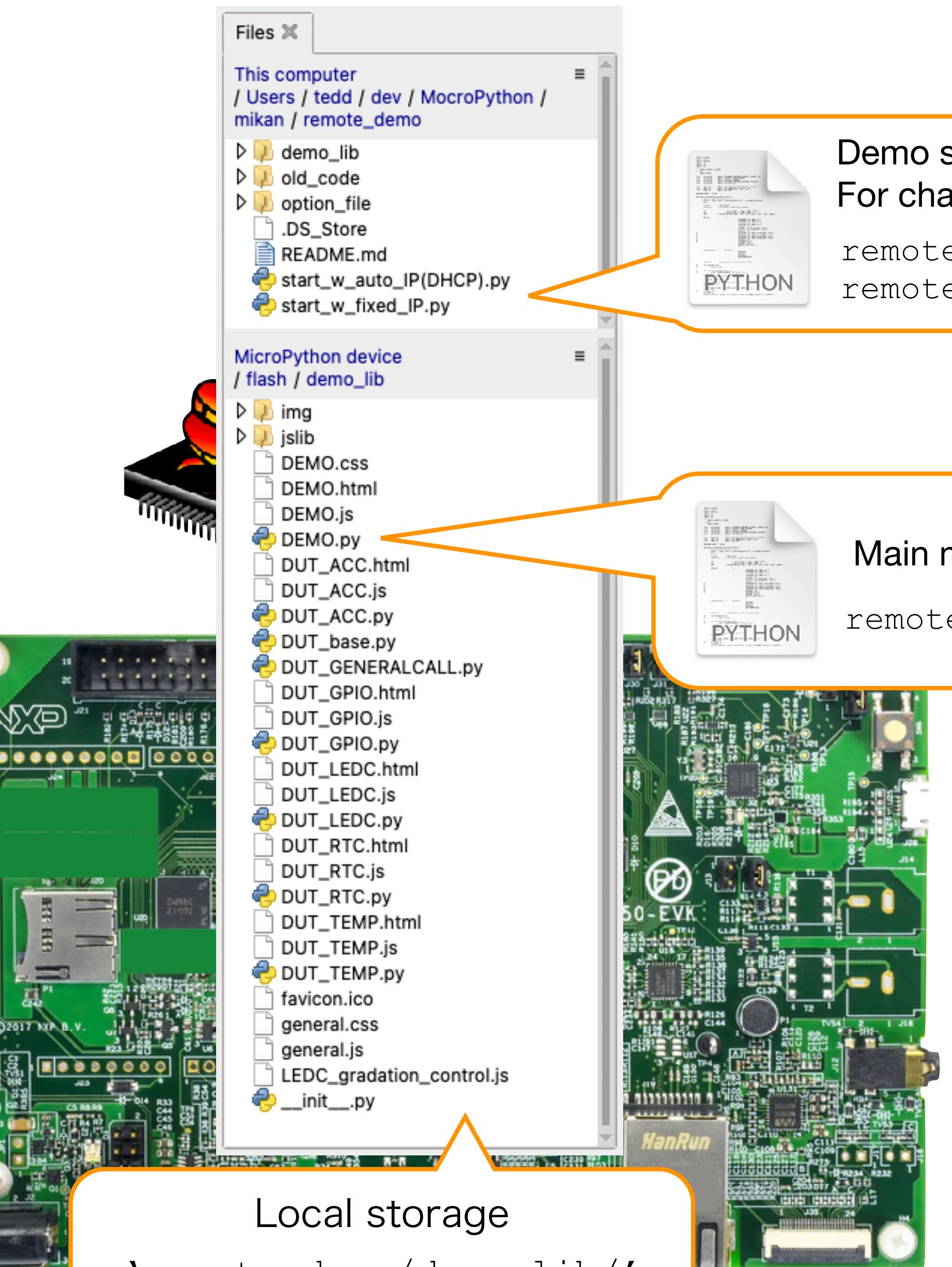
(Client mode)



LAN cable



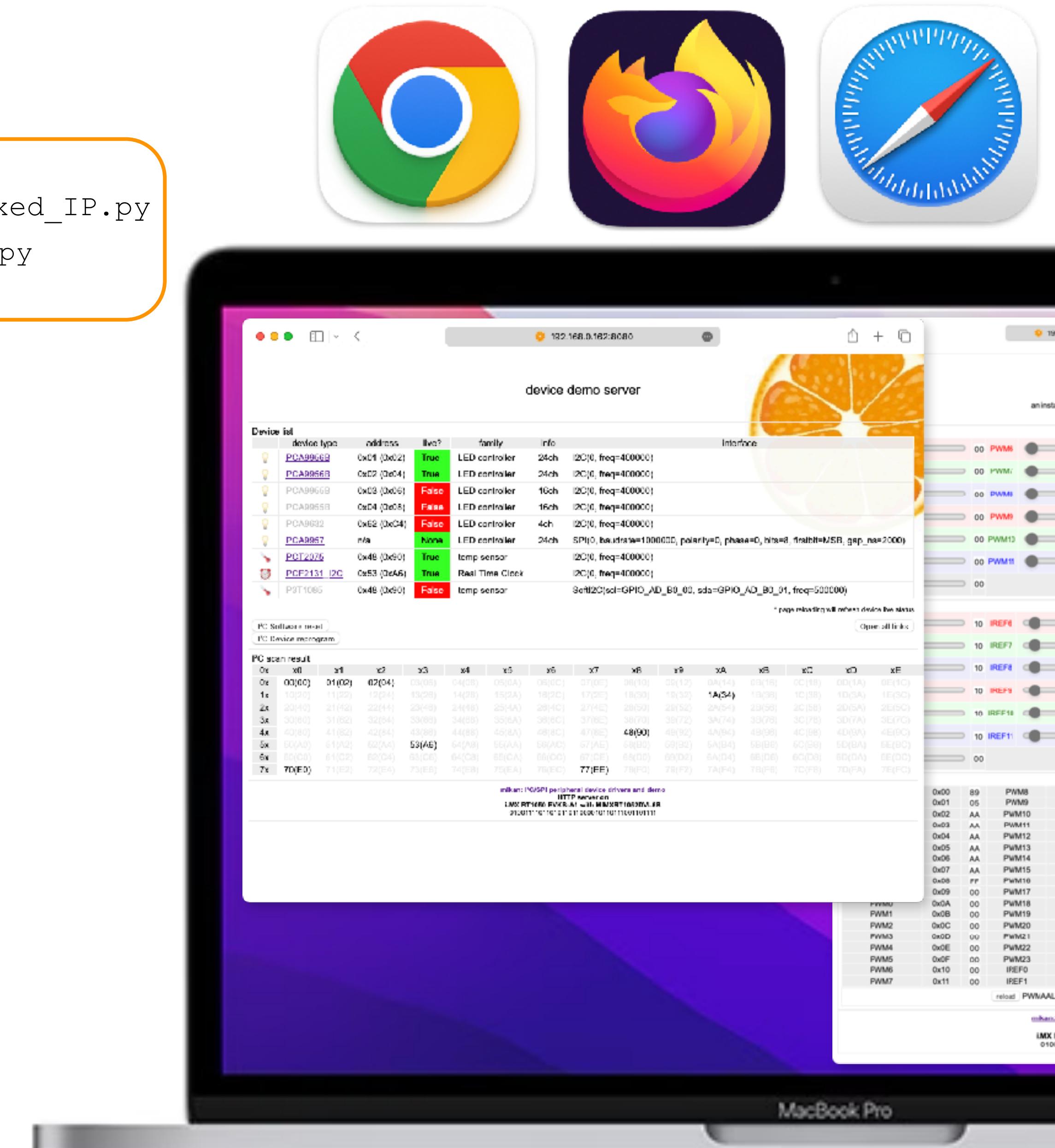
How does it work? : Software - files



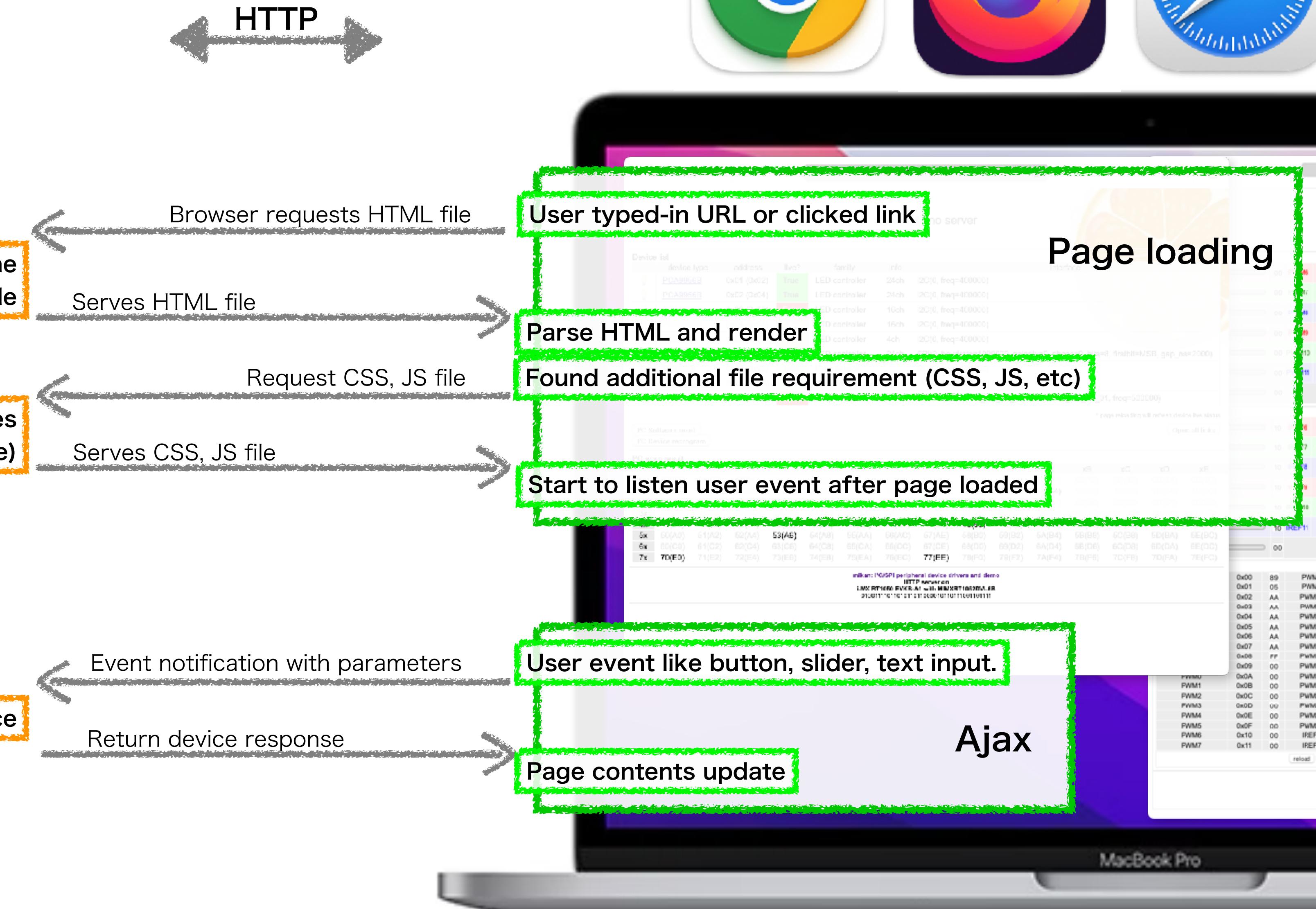
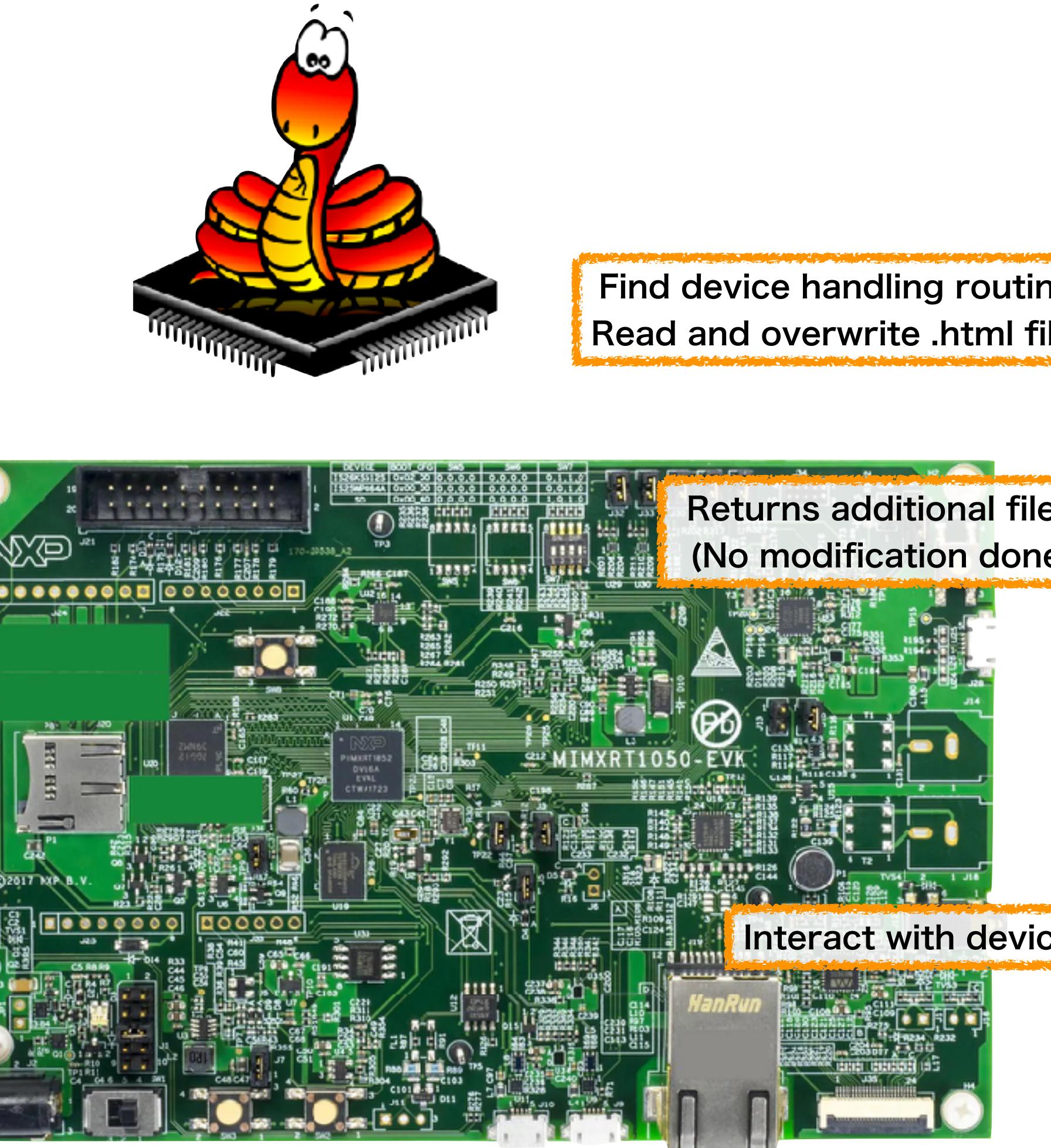
Demo start script: Run one of these scripts
For changing IP address, edit `start_w_fixed_IP.py`
`remote_demo/start_w_auto_IP(DHCP).py`
`remote_demo/start_w_fixed_IP.py`

Main module of this demo app
`remote_demo/demo_lib/DEMO.py`

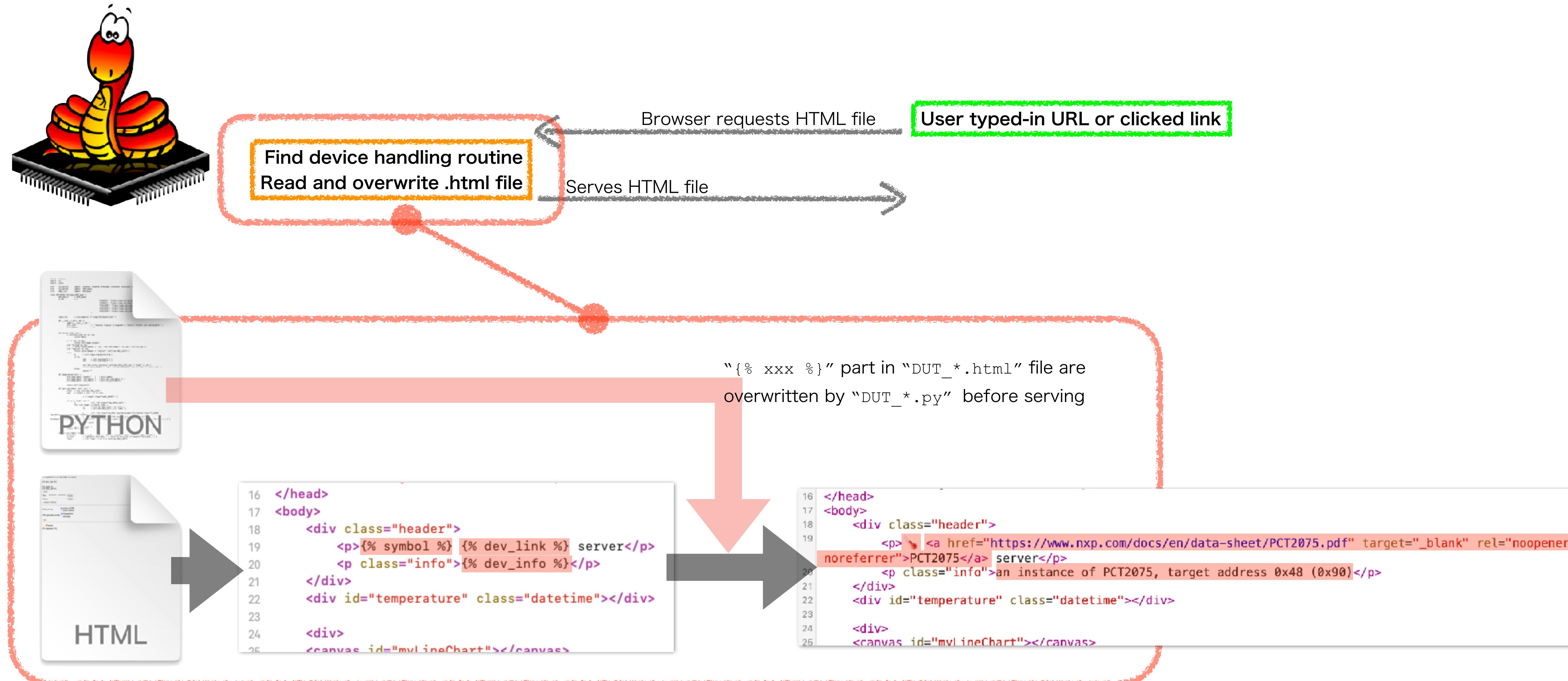
Local storage
`'remote_demo/demo_lib/'`
has py, html, css and js files



How does it work? : Software - communication



How does it work? : Software - HTML overwrite



How does it work? : Software - HTML overwrite (sample: PCT2075)

The screenshot shows two code editors side-by-side, illustrating the refactoring of a web application from using temperature data to using GPIO data.

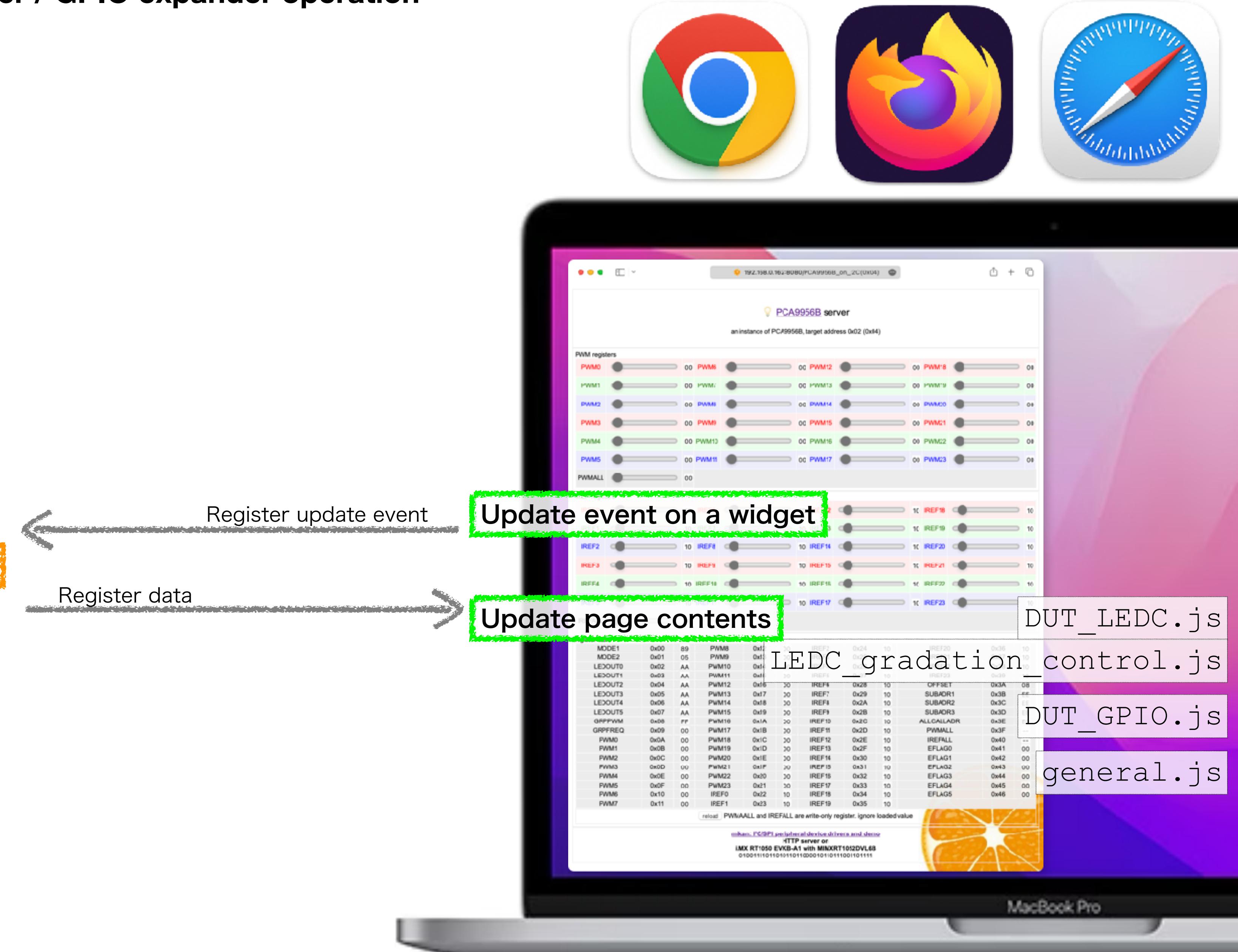
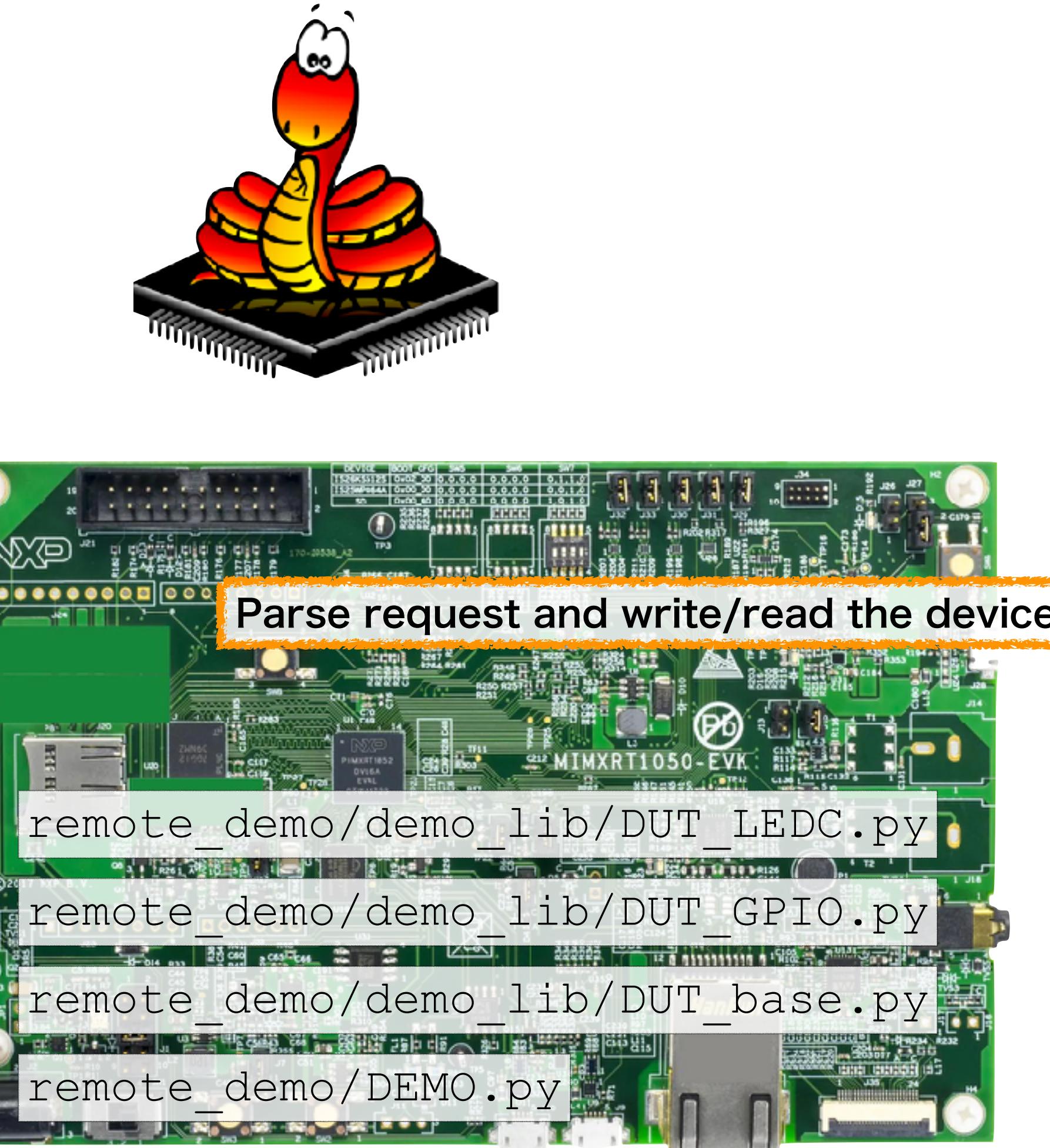
DUT_TEMP.html (Left Editor):

```
1 <!--
2 https://qiita.com/Haruka-Ogawa/items/59facd24f2a8bdb6d369
3 -->
4
5 <!DOCTYPE html>
6 <html>
7
8 <head>
9   <meta charset="utf-8" />
10  <meta http-equiv="Pragma" content="no-cache">
11  <meta http-equiv="Cache-Control" content="no-cache">
12  <meta http-equiv="Expires" content="Thu, 01 Dec 1994 16:00:00 GMT">
13
14  <title>% dev_name % server</title>
15  <link href="general.css" rel="stylesheet">
16 </head>
17 <body>
18   <div class="header">
19     <p>% symbol %> % dev_link %> server</p>
20     <p class="info">% dev_info %</p>
21   </div>
22   <div id="temperature" class="datetime"></div>
23
24   <div>
25     <canvas id="myLineChart"></canvas>
26     <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.2/Chart.bundle.js">
27
28 <script>
29   const DEV_NAME = '% dev_name %';
30   const REQ_HEADER = '/'+DEV_NAME+'?';
31
32   const TABLE_LEN = '% table_len %';
33   const GRAPH_HIGH = '% graph_high %';
34   const GRAPH_LOW = '% graph_low %';
35   const DS_LABEL = 'DS pin [ high@' + GRAPH_HIGH + ' / low@' + GRAPH_LOW + ']';
36   const MAX_N_DATA = '% max_n_data %';
37 </script>
38
39 <script async src="general.js"></script>
40 <script async src="DUT_TEMP.js"></script>
41
42 </div>
43
44 <div class="para">
45
46   <div id="reg_table" class="control_panel reg_table log_panel">
47     <% table %>
48   </div>
49
50   <div id="reg_table" class="control_panel reg_table info_panel">
51     <% info tab %><br/>
52     <input type="button" onclick="csvFileOut(); value="Save" class="tmp_button">
53   </div>
54
55   <div id="reg_table" class="control_panel reg_table info_panel">
56     <table class="table_TEMP_slider">
57       <tr class="slider_table_row">
58         <td class="td_TEMP_slider" text_align="center">
59           To s
60         </td>
61         <td class="td_TEMP_slider" text_align="center">
62           <input type="range" oninput="updateSlider( 1, 0 )" onchange="upc
63           </td>
64         <td class="td_TEMP_slider" text_align="center">
65           <input type="text" onchange="updateValField( 0 )" id="valField0"
```

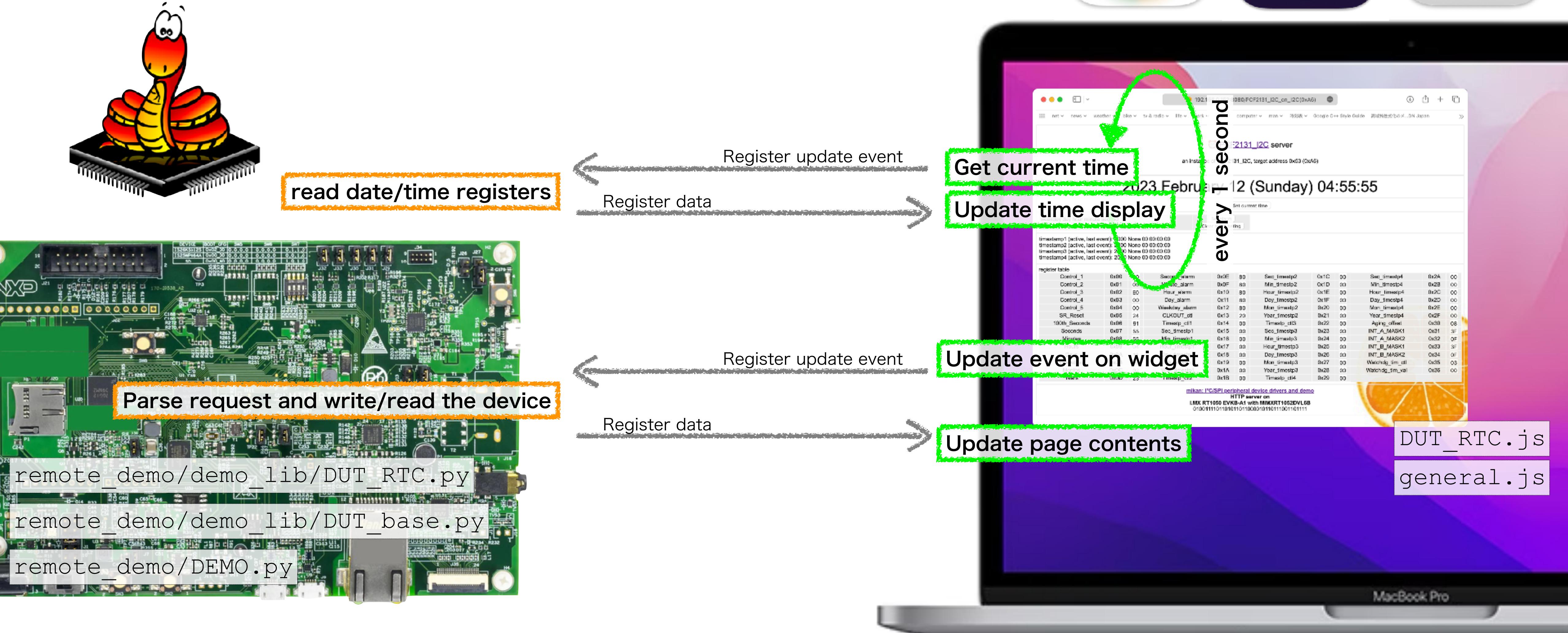
DUT_GPIO.html — Edited (Right Editor):

```
1 <!--
2 https://qiita.com/Haruka-Ogawa/items/59facd24f2a8bdb6d369
3 -->
4
5 <!DOCTYPE html>
6 <html>
7
8 <head>
9   <meta charset="utf-8" />
10  <meta http-equiv="Pragma" content="no-cache">
11  <meta http-equiv="Cache-Control" content="no-cache">
12  <meta http-equiv="Expires" content="Thu, 01 Dec 1994 16:00:00 GMT">
13
14  <title>PCT2075 on I2C(0x90) server</title>
15  <link href="general.css" rel="stylesheet">
16 </head>
17 <body>
18   <div class="header">
19     <p>% symbol %> <a href="https://www.nxp.com/docs/en/data-sheet/PCT2075.pdf" target=_blank>an instance of PCT2075, target address 0x4B (0x90)</a></p>
20     <p class="info">an instance of PCT2075, target address 0x4B (0x90)</p>
21   </div>
22   <div id="temperature" class="datetime"></div>
23
24   <div>
25     <canvas id="myLineChart"></canvas>
26     <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.7.2/Chart.bundle.js">
27
28 <script>
29   const DEV_NAME = 'PCT2075_on_I2C(0x90)';
30   const REQ_HEADER = '/'+DEV_NAME+'?';
31
32   const TABLE_LEN = 10;
33   const GRAPH_HIGH = 16;
34   const GRAPH_LOW = 6;
35   const DS_LABEL = 'DS pin ( high@' + GRAPH_HIGH + ' / low@' + GRAPH_LOW + ')';
36   const MAX_N_DATA = 60;
37 </script>
38
39 <script async src="general.js"></script>
40 <script async src="DUT_TEMP.js"></script>
41
42 </div>
43
44 <div class="para">
45
46   <div id="reg_table" class="control_panel reg_table log_panel">
47     <table class="table_TEMP"><tr><td class="td_TEMP">time</td><td class="td_TEMP">
48     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_0">
49     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_1">
50     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_2">
51     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_3">
52     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_4">
53     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_5">
54     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_6">
55     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_7">
56     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_8">
57     <tr><td class="td_TEMP" text_align="center"><input class="input_text_TMP" type="text" id="input_text_TMP_9">
58   </table>
59   </div>
60
61   <div id="reg_table" class="control_panel reg_table info_panel">
62     <table class="table_TEMP"><tr>
63       <td class="td_TEMP" text_align="center">start time</td><td class="td_TEMP"><input class="input_text_TMP" type="text" id="input_text_TMP_0">
64       <td class="td_TEMP" text_align="center">last time</td><td class="td_TEMP"><input class="input_text_TMP" type="text" id="input_text_TMP_1">
65       <td class="td_TEMP" text_align="center">sample count</td><td class="td_TEMP"><input class="input_text_TMP" type="text" id="input_text_TMP_2">
66     </tr>
67   </table>
68   </div>
```

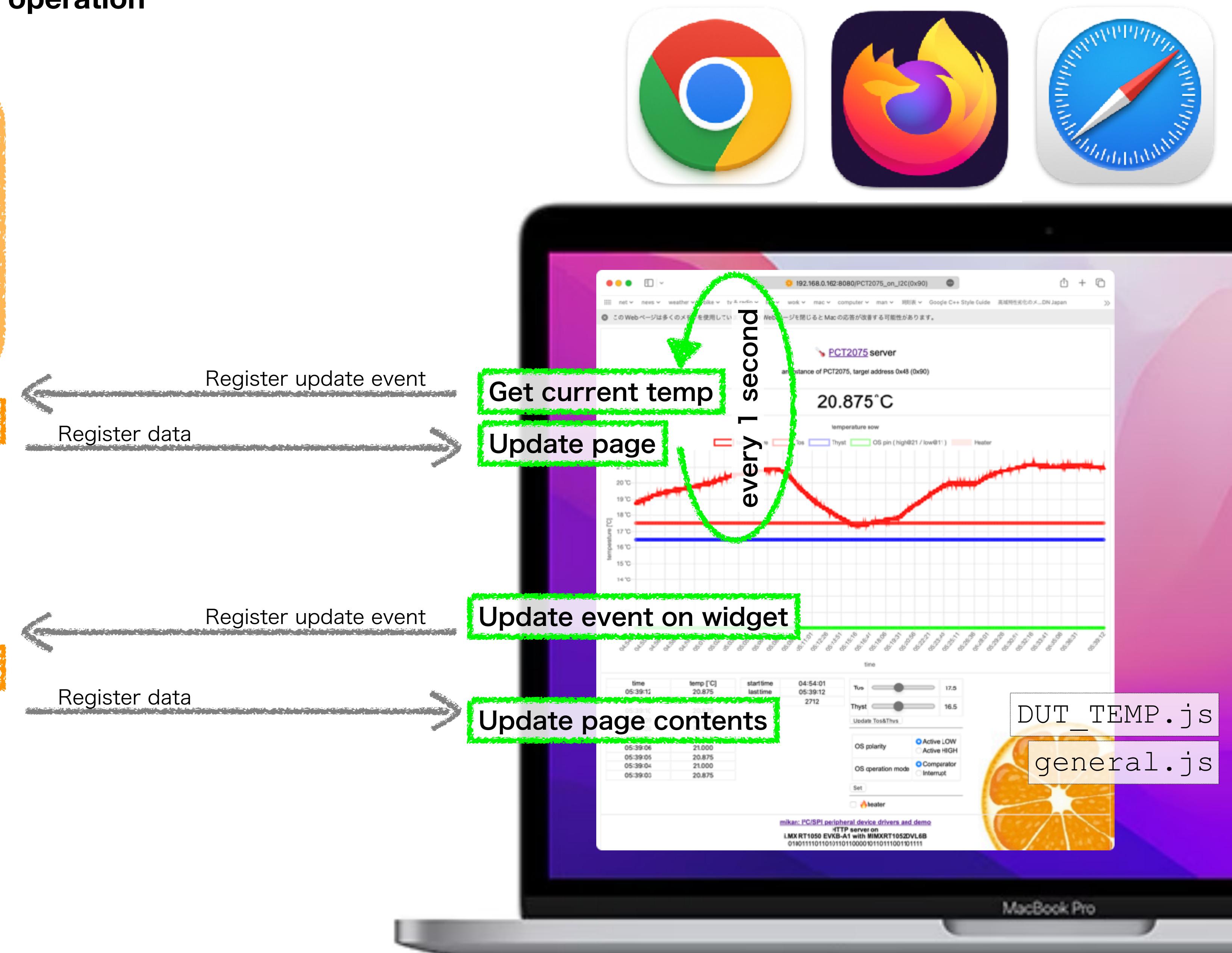
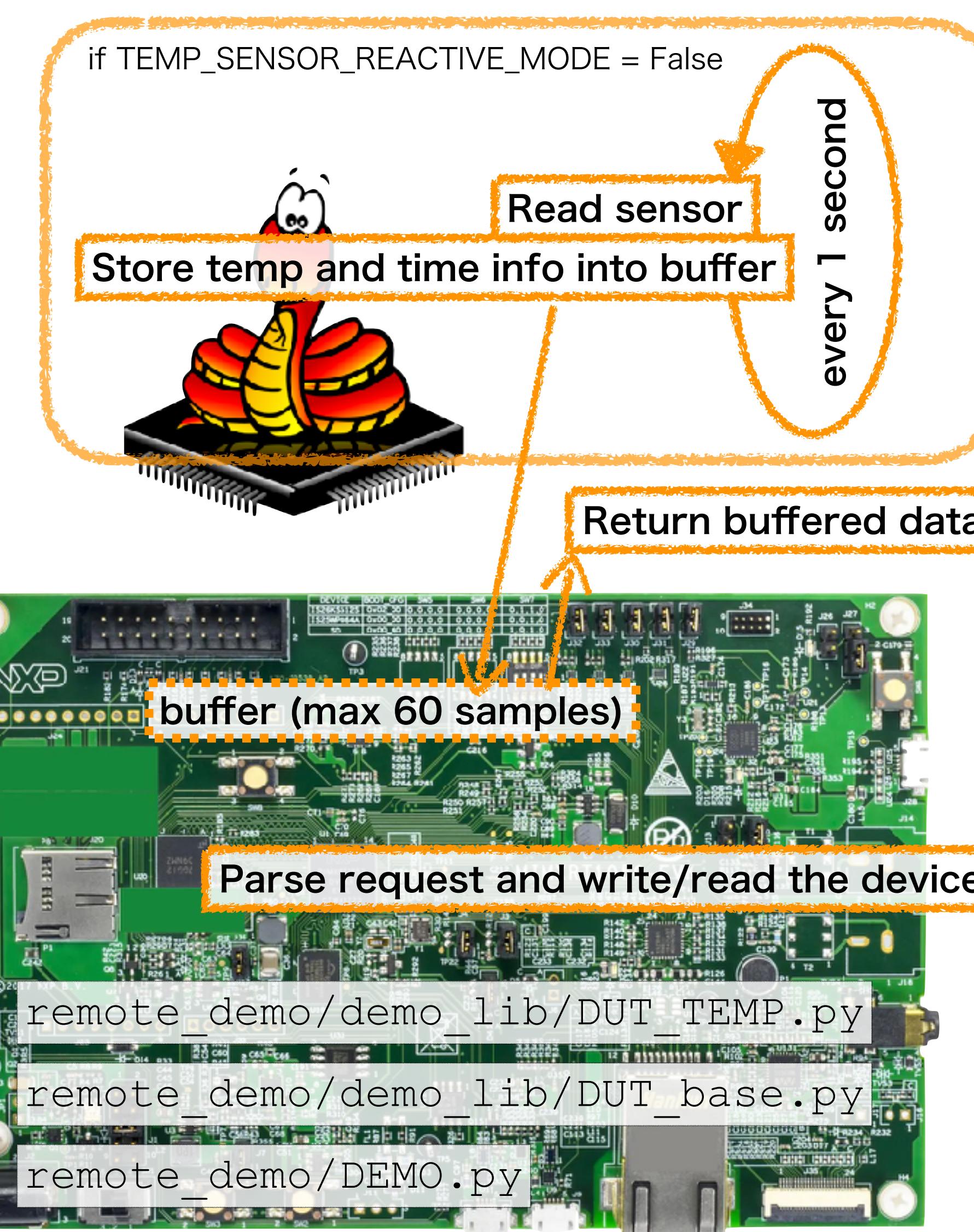
How does it work? : Software - LED controller / GPIO expander operation



How does it work? : Software - RTC operation

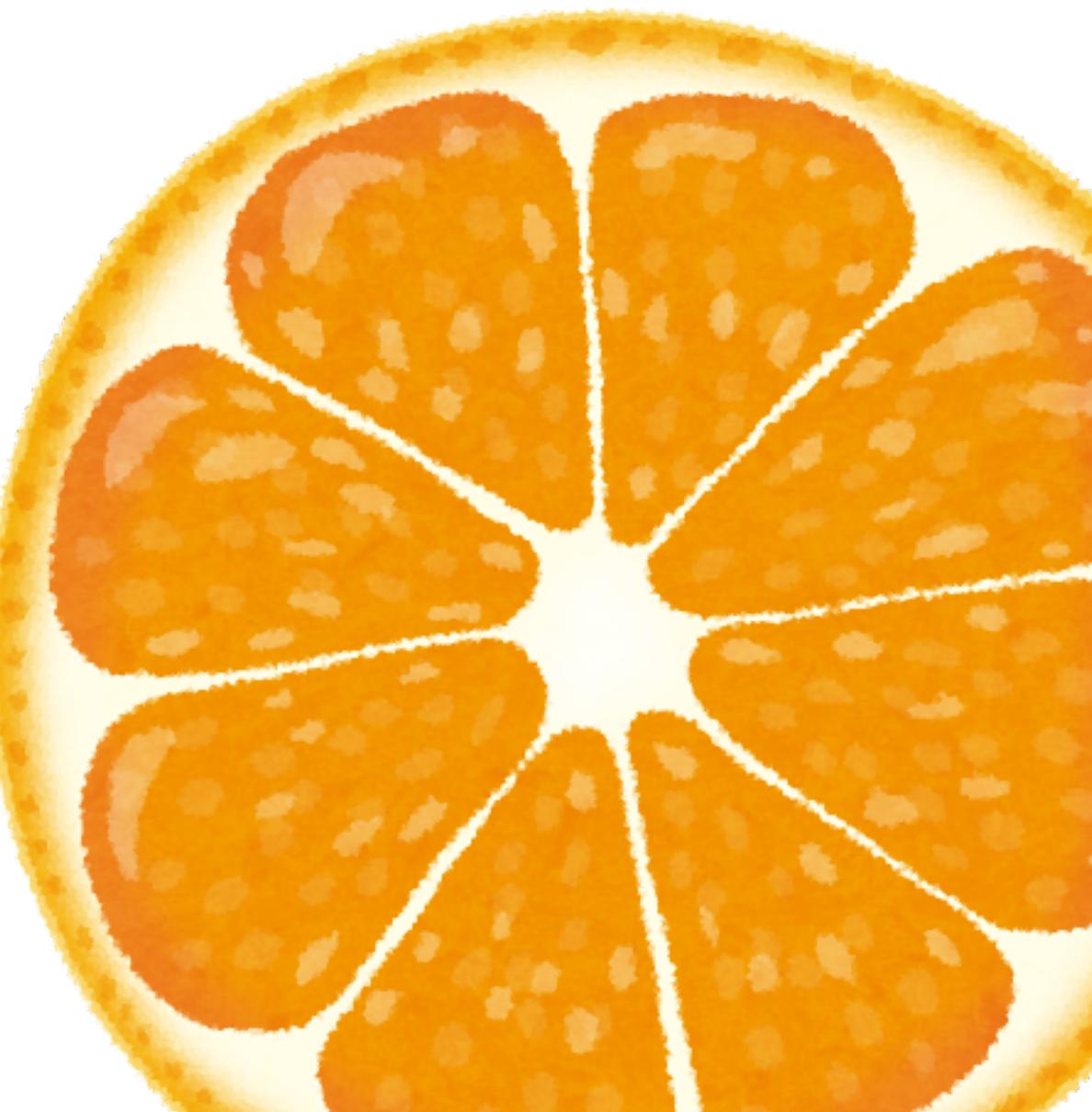


How does it work? : Software - temp sensor operation



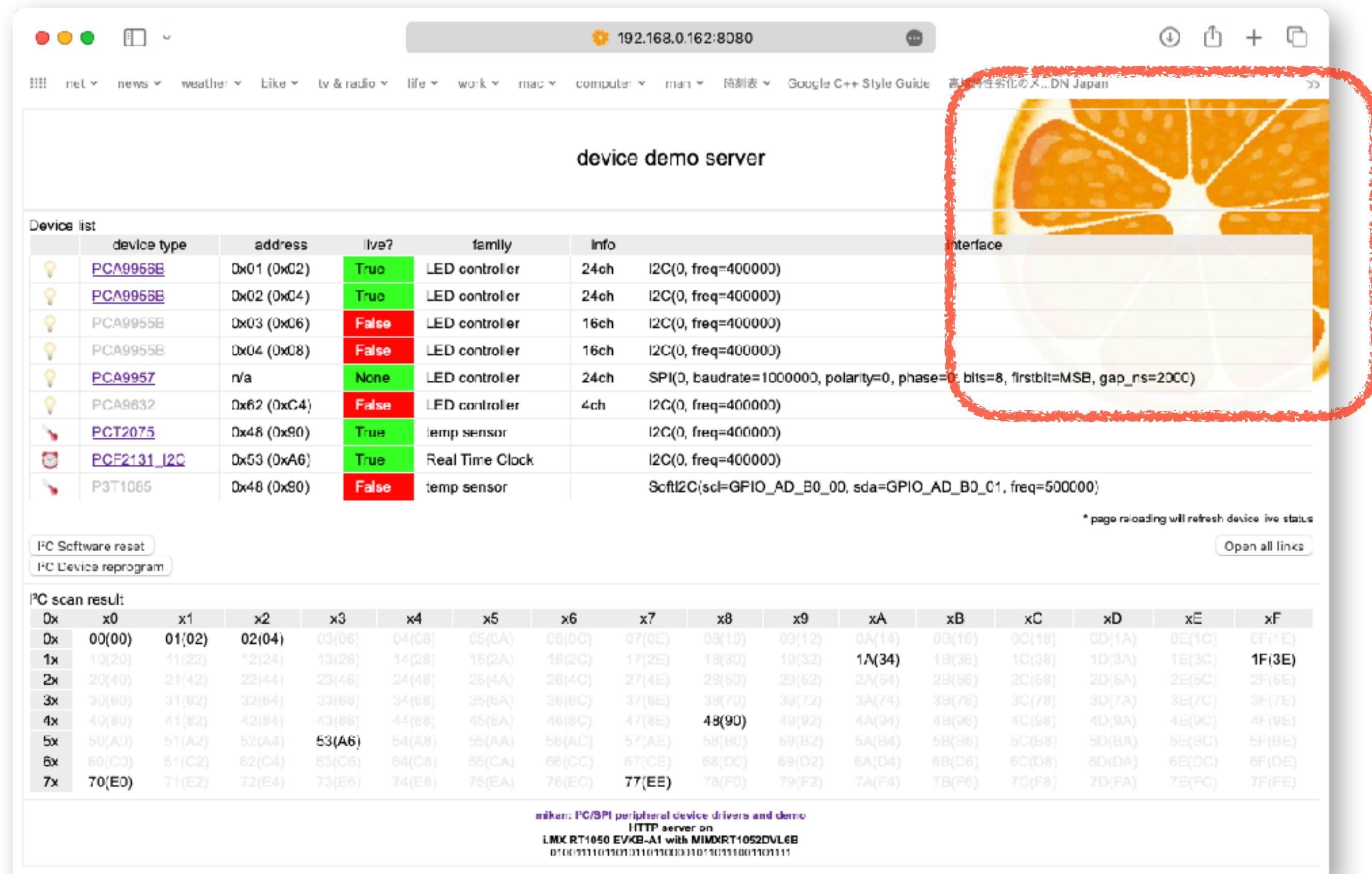
Customizing remote_demo

mikan

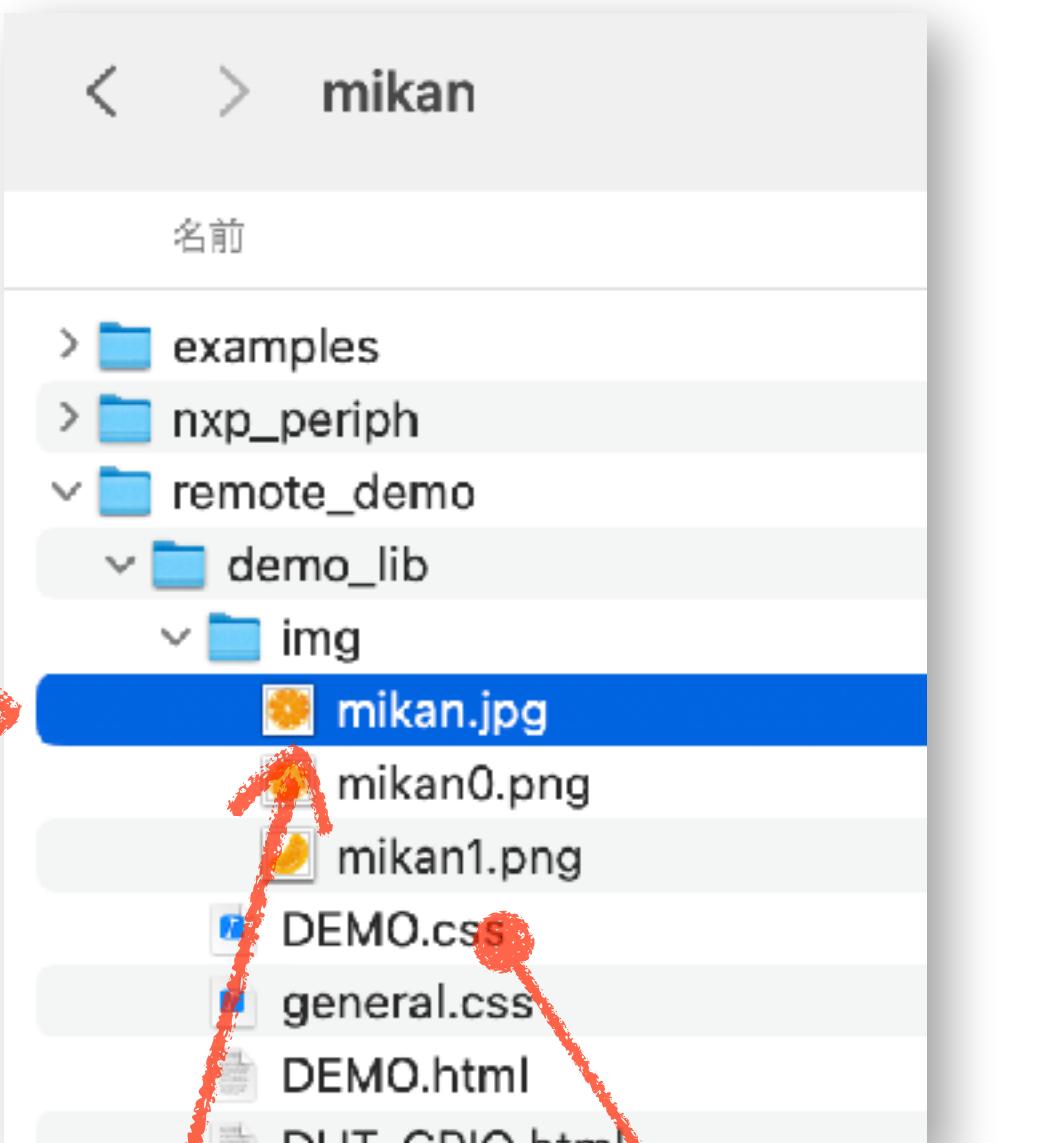


Changing logo image setting

Changing logo on right-top on device list page



The screenshot shows the 'device demo server' interface at 192.168.0.162:8080. It displays a device list with columns for device type, address, live?, family, info, and interface. Below the list is an I2C scan result table. At the bottom, there are links for I2C Software reset and I2C Device reprogram. The top right corner features a logo of an orange with segments removed, which is highlighted with a red circle.



The screenshot shows a file explorer window titled 'mikan' with a tree view of directory contents. A red arrow points from the highlighted logo image in the browser to the 'mikan.jpg' file in the 'img' folder. Another red arrow points from the 'mikan.jpg' file to the 'background-image' line in the 'DEMO.css' file.

```
html {
    font-size: 80%;
    font-family: Arial;
    display: inline-block;
    text-align: center;
}
body {
    font-size: 1.0rem;
    font-color: #000000;
    vertical-align: middle;
    background-image: url(img/mikan.jpg);
    background-position: 110% -10%;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 33%;
    background-color: #FFFFFF;
    opacity: 0.95;
}
div {
    border: solid 1px #EEEEEE;
```

The right-top logo image on device list page is defined in 'remote_demo/demo_lib/DEMO.css'.

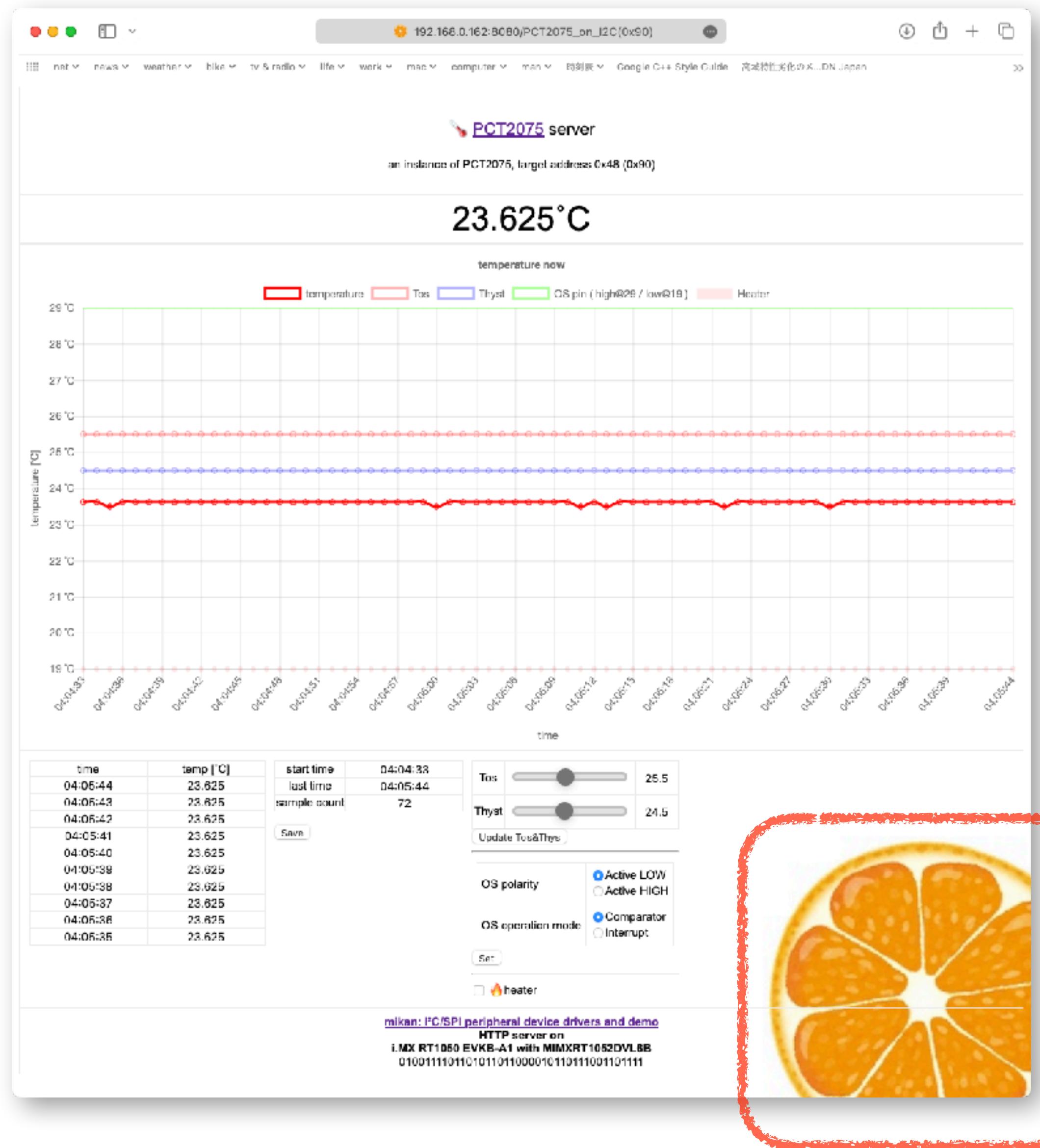
To change the logo

- (1) Replace 'mikan.jpg' file
- (2) Or put a new image file and modify 'background-image' URL.

In the remote demo, 'remote_demo/demo_lib/' will be root of file path.

Changing logo image setting

Changing logo on right-bottom on device page



The screenshot shows a file explorer window titled 'mikan'. It displays a directory structure under 'remote_demo/demo_lib/img'. A file named 'mikan.jpg' is highlighted. A red arrow points from the logo image in the device page to this file. Another red arrow points from the 'general.css' file in the 'general.css' folder to the corresponding line of CSS code in the text editor below.

```
html {
    font-size: 80%;
    font-family: Arial;
    display: inline-block;
    text-align: center;
}
body {
    font-size: 1.0rem;
    font-color: #000000;
    vertical-align: middle;
    background-image: url(img/mikan.jpg);
    background-position: bottom -10% right -10%;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 33%;
    background-color: #FFFFFF;
    opacity: 0.95;
}
canvas {
```

Changing logo image setting

Showing logo on left-top on device list page

The screenshot shows a web application for managing I2C devices. It has two main sections: 'device list' and 'I2C scan result'. The 'device list' section contains a table of 18 devices, each with its device type, address, live status, family, and additional information. The 'I2C scan result' section shows two tables of I2C addresses from 0x00 to 0x7F. A yellow citrus fruit logo is positioned in the top right of the main content area. Two specific areas are highlighted with red boxes: one around the top-left empty space and another around the logo image.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>device list</title>
    <link href="DEMO.css" rel="stylesheet">
  </head>
  <body>
    <script src="general.js"></script>
    <script src="DEMO.js"></script>

    <div class="header">
      <!--
        <div class="header_img">
          <img src = "img/mikan1.png" class="logo">
        </div>
      -->
      <div class="header_txt">
        <p>device demo server</p>
      </div>
    </div>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>device list</title>
    <link href="DEMO.css" rel="stylesheet">
  </head>
  <body>
    <script src="general.js"></script>
    <script src="DEMO.js"></script>
```

```
<div class="header">
  <div class="header_img">
    <img src = "img/mikan1.png" class="logo">
  </div>
  <div class="header_txt">
    <p>device demo server</p>
  </div>
```

Changing logo image setting

Favicon

Device list

device type
PCA9956B
TL-WR902AC
HTML カラーコード
teddokano (Tedd OKANO)
Offliberty - evidence of offline life
Google Keep
R Favicon Generator for perfect icons on all browsers
device list
PCF2131_I2C_on_I2C(0xA6) server
PCA9956B_on_I2C(0x02) server
PCT2075_on_I2C(0x90) server

Nothing to do in code. Web-browser will request 'favicon.ico' in root path
Just prepare an image file as 'remote_demo/demo_lib/favicon.ico'.
Image converter web-tool like <https://realfavicongenerator.net/> will help to make it easier.

Customizing device list

Be careful to listing multiple devices together

Put device instances in 'devices' list.
This list will appear in the device list page.

```
def main( micropython_optimize = False ):                                remote_demo/DEMO.py
    print( "remote device demo" )
    print( " http server is started working on " + os.uname().machine )
    print( "" )

    src_dir  = "demo_lib/"
    regex_file = ure.compile( r"GET /(\S+)\sHTTP" )

    i2c  = machine.I2C( 0, freq = (400 * 1000) )
    spi  = machine.SPI( 0, 1000 * 1000, cs = 0 )
    si2c = machine.SoftI2C( sda = "D14", scl = "D15", freq = (400 * 1000) )

    devices = [
        PCA9956B( i2c, 0x02 >>1 ),
        PCA9956B( i2c, 0x04 >>1 ),
        PCA9955B( i2c, 0x06 >>1 ),
        PCA9955B( i2c, 0x08 >>1 ),
        PCA9632( i2c ),
        PCA9957( spi, setup_EVB = True ),
        PCT2075( i2c, setup_EVB = True ),
        PCF2131( i2c ),
        # PCAL6408( i2c, 0x21, setup_EVB = True ),
        # PCAL6416( i2c, 0x20, setup_EVB = True ),
        # PCAL6524( i2c, 0x22, setup_EVB = True ),
        # PCAL6534( i2c, 0x23, setup_EVB = True ),
        PCF2131( spi ),
        PCF85063( i2c ),
        P3T1085( si2c ),
        General_call( i2c ),
    ]
```

'General_call' is a special class to
make an instance of I2C general call.

Device list						
	device type	address	live?	family	info	
💡	PCA9956B	0x01 (0x02)	True	LED controller	24ch	I2C(0, freq=400000)
💡	PCA9956B	0x02 (0x04)	True	LED controller	24ch	I2C(0, freq=400000)
💡	PCA9955B	0x03 (0x06)	False	LED controller	16ch	I2C(0, freq=400000)
💡	PCA9955B	0x04 (0x08)	False	LED controller	16ch	I2C(0, freq=400000)
💡	PCA9632	0x62 (0xC4)	False	LED controller	4ch	I2C(0, freq=400000)
💡	PCA9957	n/a	None	LED controller	24ch	SPI(0, baudrate=1000000)
⚡	PCT2075	0x48 (0x90)	True	temp sensor		I2C(0, freq=400000)
⌚	PCF2131_I2C	0x53 (0xA6)	True	Real Time Clock		I2C(0, freq=400000)
⚡	P3T1085	0x48 (0x90)	False	temp sensor		SoftI2C(scl=GPIO_AD_B0)

I²C Software reset I²C Device reprogram

I²C scan result

0x	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9
0x	00(00)	01(02)	02(04)	03(06)	04(08)	05(0A)	06(0C)	07(0E)	08(10)	09(12)
1x	10(20)	11(22)	12(24)	13(26)	14(28)	15(2A)	16(2C)	17(2E)	18(30)	19(32)
2x	20(40)	21(42)	22(44)	23(46)	24(48)	25(4A)	26(4C)	27(4E)	28(50)	29(52)

Be careful for order of making instances.

Some EVBs are not compatible to operate on same Arduino shield socket.

'setup_EVB = True' will set some pins to specific state.

Those disturb other EVB operation.

ie. PCAL6xxx-ARD, PCA995BTW-ARD, P3T1085UK-ARD

Customizing :general

CSS defines page design: Try modifying CSS

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>device list</title>
    <link href="DEMO.css" rel="stylesheet">
  </head>
```

remote_demo/demo_lib/DEMO.html

remote_demo/demo_lib/DEMO.css

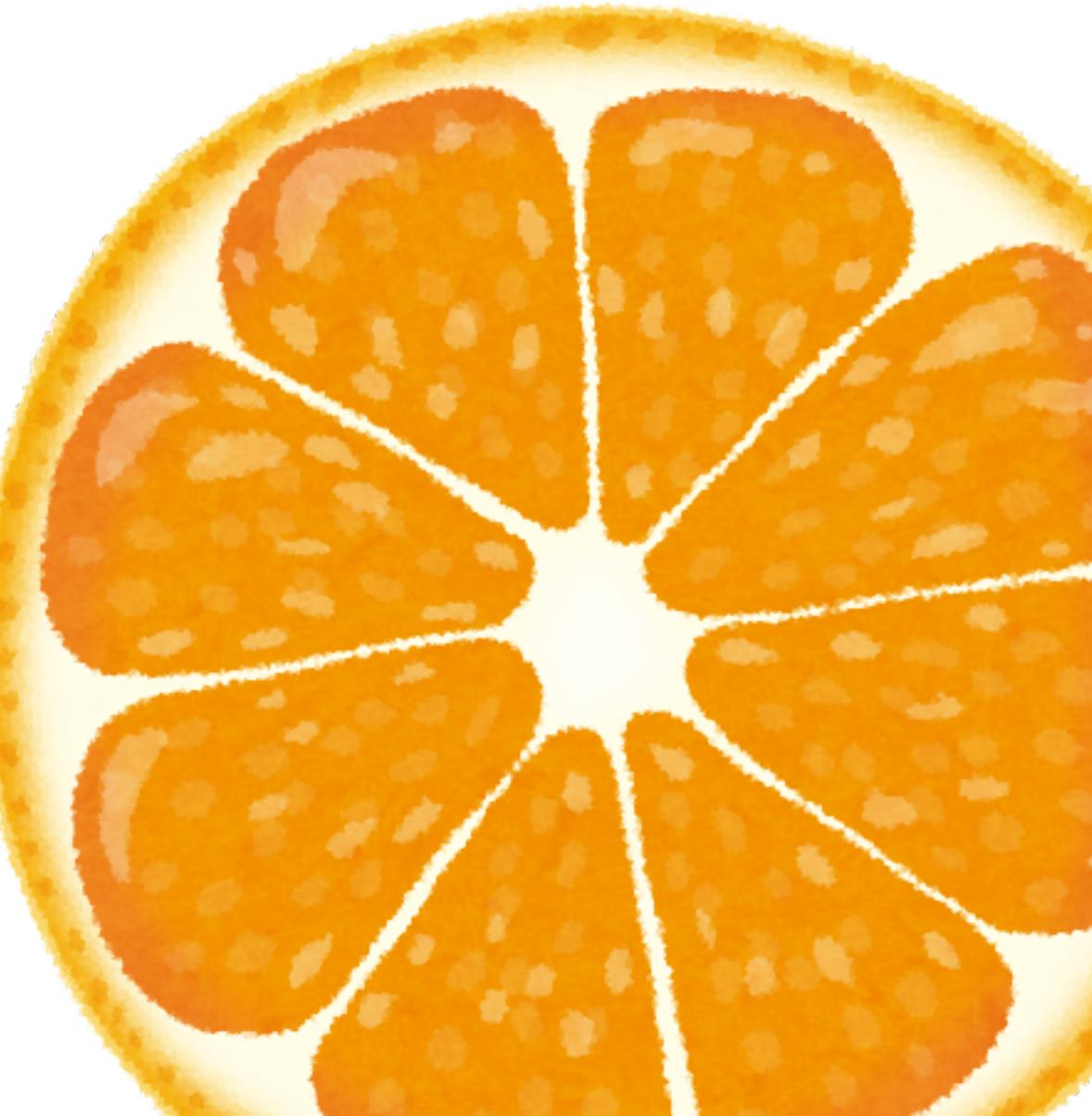
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>{% dev_name %} server</title>
    <link href="general.css" rel="stylesheet">
  </head>
```

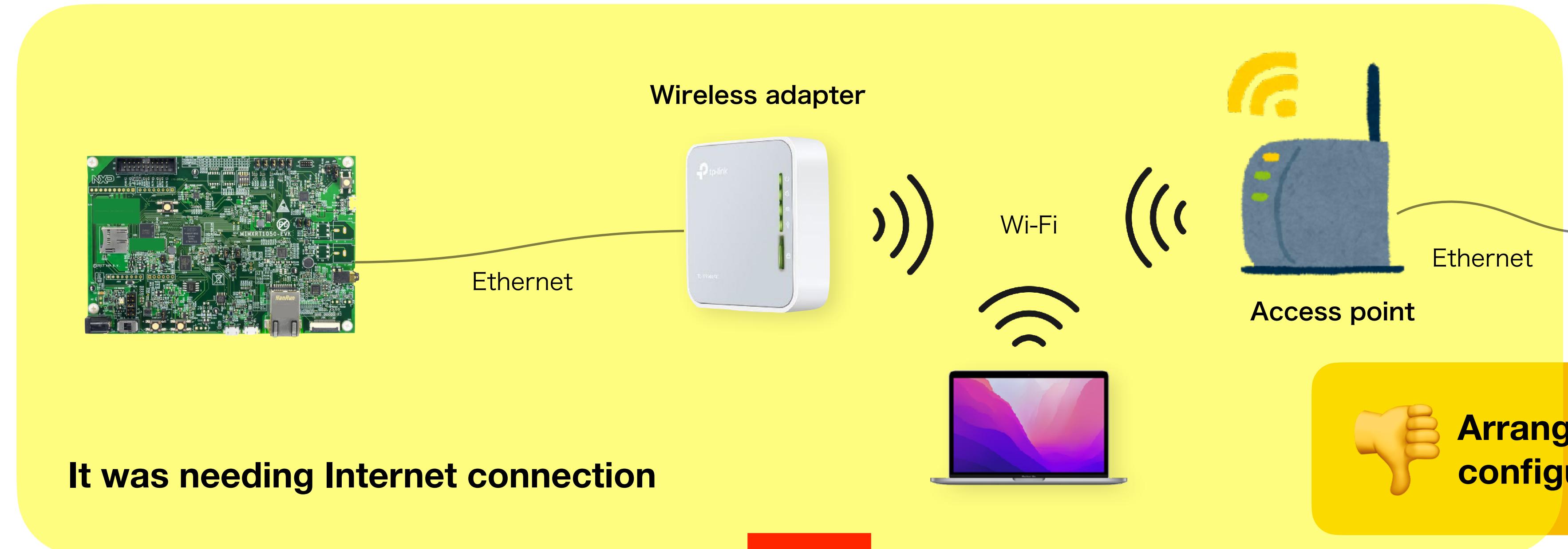
remote_demo/demo_lib/DUT_GPIO.html
remote_demo/demo_lib/DUT_LEDC.html
remote_demo/demo_lib/DUT_RTC.html
remote_demo/demo_lib/DUT_TEMP.html

remote_demo/demo_lib/general.css

NOTES

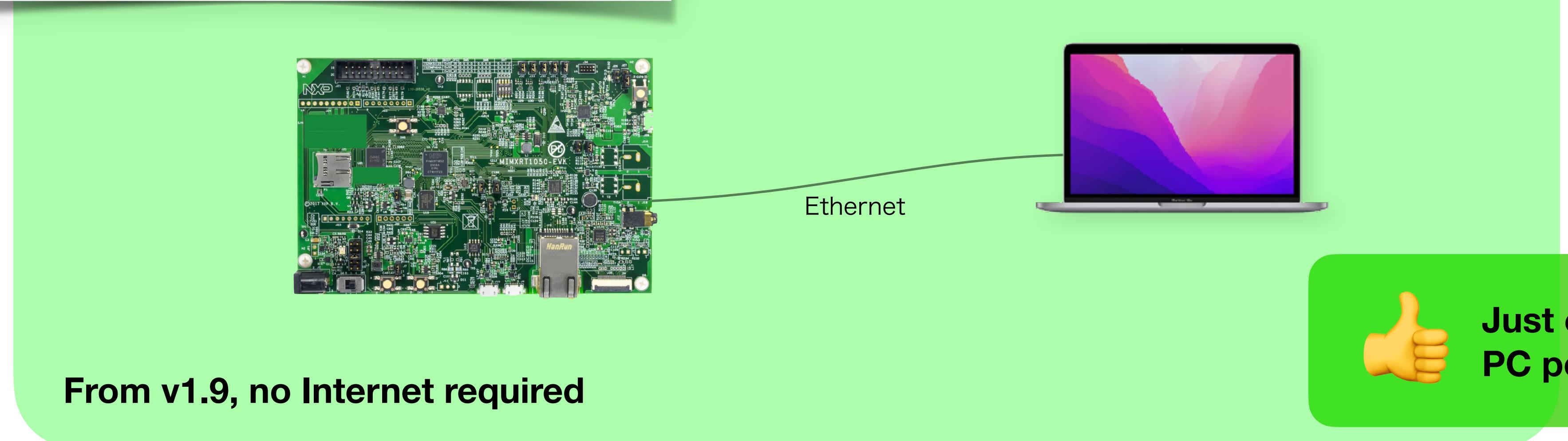
mikan





Arrange equipment,
configure those

IMPROVED! from v1.9



Just configure the
PC port

注意点 (+ 直近のデバッグ状況)

RTCと温度センサの同時動作でMicroPythonが停止：I²C通信中にI²C通信を始めてしまってた

remote_demoではブラウザからの要求で、 I²Cアクセスを行う。

しかし温度センサだけは例外で、 MCUのタイマ割り込みで温度を読み出していた。

RTCのレジスタの読み出し中に割り込みが発生しすると、それをきっかけにI²C通信が始まってしまう

TEMP_SENSOR_REACTIVE_MODE を追加。他のデバイスと同時にデモを行うときにはこのモードを使うことにより、上記の問題発生を抑える



```
DUT_TEMP.py
DUT_TEMP.py > C DUT_TEMP
import machine
import ure
import ujson
import micropython
from nxp_periph import PCT2075, LM75B, P3T1085
from nxp_periph import temp_sensor_base
from demo_lib import DUT_base
TEMP_SENSOR_REACTIVE_MODE = True # Default: To operate multiple I2C devices on same bus
#TEMP_SENSOR_REACTIVE_MODE = False # If only one device is connected on same I2C bus
class DUT_TEMP( DUT_base.DUT_base ):
    APPLIED_TO = temp_sensor_base
    TABLE_LENGTH = 10
    SAMPLE_LENGTH = 60
    GRAPH_HIGH = 30
```

remote_demo/demo_lib/DUT_TEMP.py

温度センサ単独で使うならこれを「False」で
そのほうがキレイなデータが取れます (^_^;

温度センサからのデータにはMicroPython側の
タイムスタンプが付いているため、リアクティ
ブ・モードにするとブラウザとMicroPython側
の時間のズレでデータ飛びが起こるかも知れな
い（データ重複は対応しているけど、データ飛
びは未対策）

注意点 (+ 直近のデバッグ状況)

ドキュメント：整備中

```
class temp_sensor_base():
    """
    An abstraction class to make user interface.

    def read( self ):
        """
        Read temperature

        Returns
        -----
        float : temperature in degree-Celsius
        """
        return self.__read()

    def reg_access( self, *args ):
        """
        Write or read register

        Recommended to use this method instead of write_registers()/read_registers()
        because register accessing style is different from LED_controllers and RTCs.
        This reg_access() method does 8 bit and 16 bit access automatically

        The reg_access() takes 1 or 2 arguments.
        If 2 arguments are given, it performs write.
        If only 1 argument is given, read is performed.

        Parameters
        -----
        args[0] : string or int
            Register name or pointer.
        args[1] : int, optional
            Data to be written.

        Returns
        -----
        int : register data
            Nothing will be returned when write is done.

    Examples
    -----
    self.reg_access( "Conf", 0 )    # writes 8 bit data (writes 16 bit data for P3T1085)
    value = self.reg_access( "Temp" ) # reads 16 bit data
    """


```

nxp_periph/temp_sensor.py

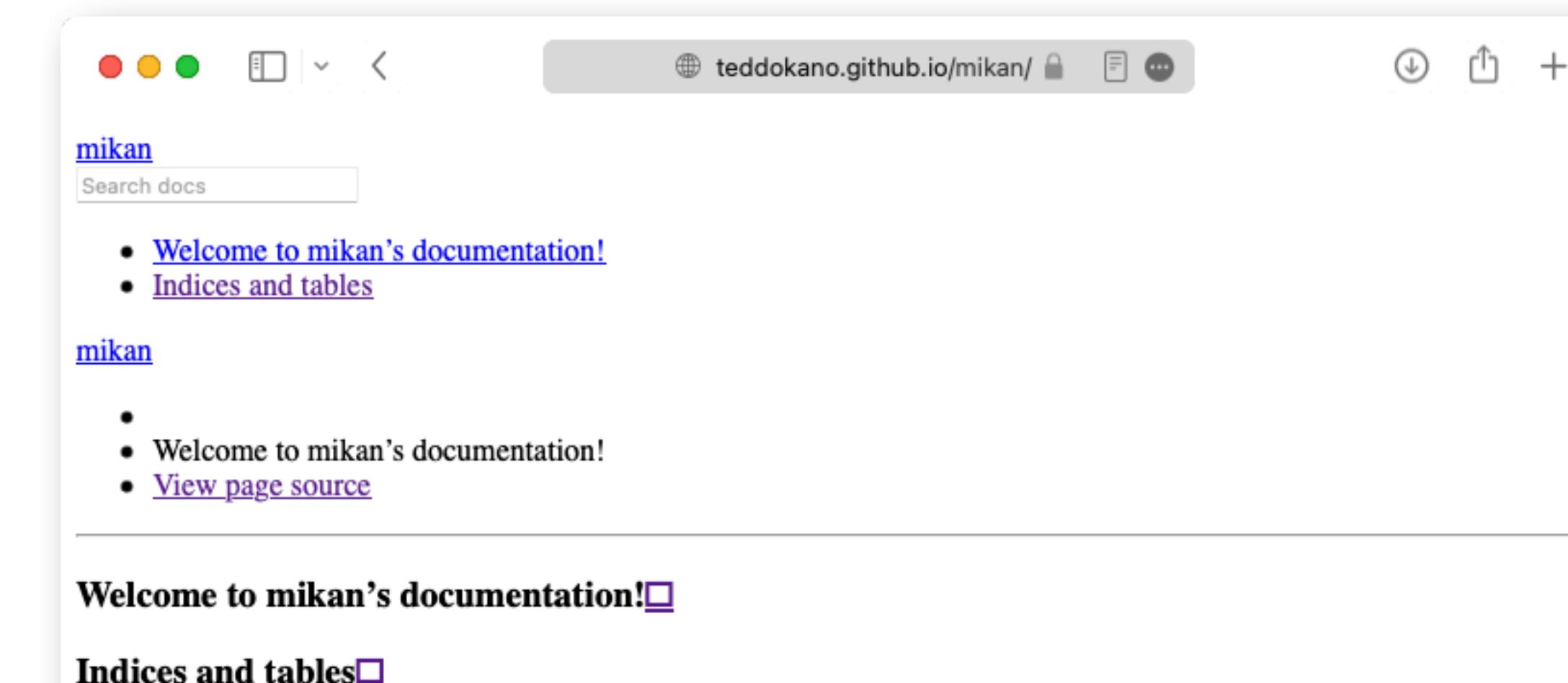
'nxp_periph/`フォルダ内のコードには
オンライン・ドキュメントあり

この内容と'examples/'内のコードを見れば使い方が分か
るようになっています

このオンライン・ドキュメントから生成したページもあるの
ですが、CSSがうまく効かず、キレイな表示ができていません

<https://teddokano.github.io/mikan/>

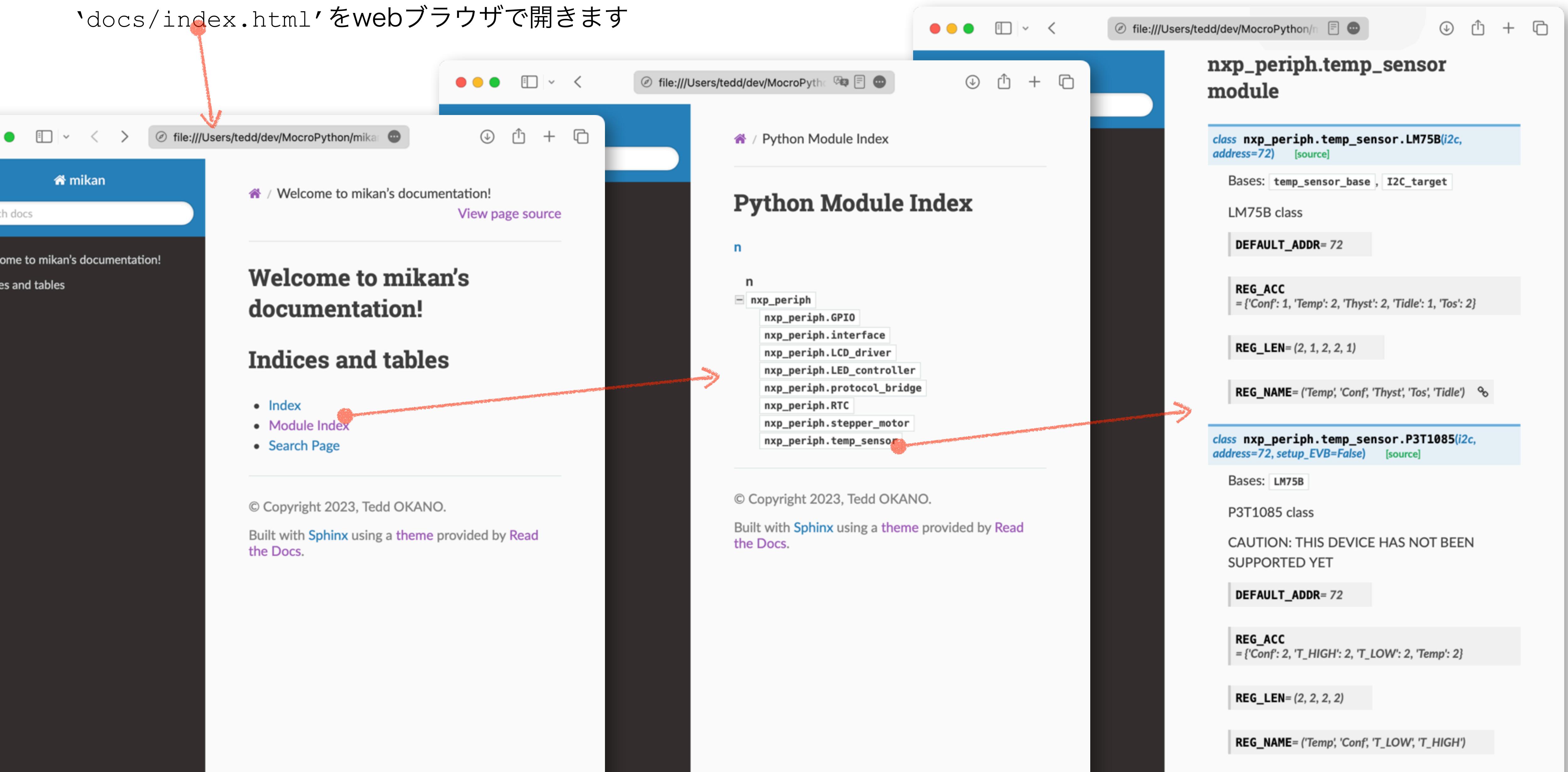
また、これらのドキュメントはクラスの(継承)構造が前提に
なっているため少し難解です。
どのように書くのがいいのか悩み中



注意点 (+ 直近のデバッグ状況)

ドキュメントの見方：例えば温度センサなら..

'docs/index.html' を web ブラウザで開きます



注意点 (+ 直近のデバッグ状況)

ドキュメントの見かた：例えば温度センサなら..

class `nxp_periph.temp_sensor.PCT2075`(`i2c, address=72, setup_EVB=False`) [source]
Bases: `LM75B`

`PCT2075` class
`DEFAULT_ADDR= 72`
`REG_ACC= {'Conf': 1, 'Temp': 2, 'Thyst': 2, 'Tidle': 1, 'Tos': 2}`
`REG_LEN= (2, 1, 2, 2, 1)`
`REG_NAME= ('Temp', 'Conf', 'Thyst', 'Tos', 'Tidle')`

property `heat` PCT2075の項ではPCT2075独自のメソッドとプロパティを表示
PCT2075はLM75Bのメソッドとプロパティを引き継いでる

class `nxp_periph.temp_sensor.LM75B`(`i2c, address=72`) [source]
Bases: `temp_sensor_base, I2C_target`

`LM75B` class
`DEFAULT_ADDR= 72`
`REG_ACC= {'Conf': 1, 'Temp': 2, 'Thyst': 2, 'Tidle': 1, 'Tos': 2}`
`REG_LEN= (2, 1, 2, 2, 1)`
`REG_NAME= ('Temp', 'Conf', 'Thyst', 'Tos', 'Tidle')`

CAUTION: THIS DEVICE HAS NOT BEEN SUPPORTED

class `nxp_periph.temp_sensor.temp_sensor_base` [source]
Bases: `object`

An abstraction class to make user interface.

`dump()` [source]
`dump_reg()` [source]

Showing all register name, address/pointer and value
(Overriding I2C_target class method)

`read()` [source]

Read temperature

Returns: float
Return type: temperature in degree-Celsius

`reg_access(*args)` [source]

Write or read register

Recommended to use this method instead of write_registers()/read_registers() because register accessing style is different from LED_controllers and RTCs. This reg_access() method does 8 bit and 16 bit access automatically

The reg_access() takes 1 or 2 arguments. If 2 arguments are given, it performs write. If only 1 argument is given, it performs read.

• `args[1] (int, optional) - Data to be written.`

Returns: int – Nothing will be returned when write is done.
Return type: register data

`ping()` [source]

ping for a device

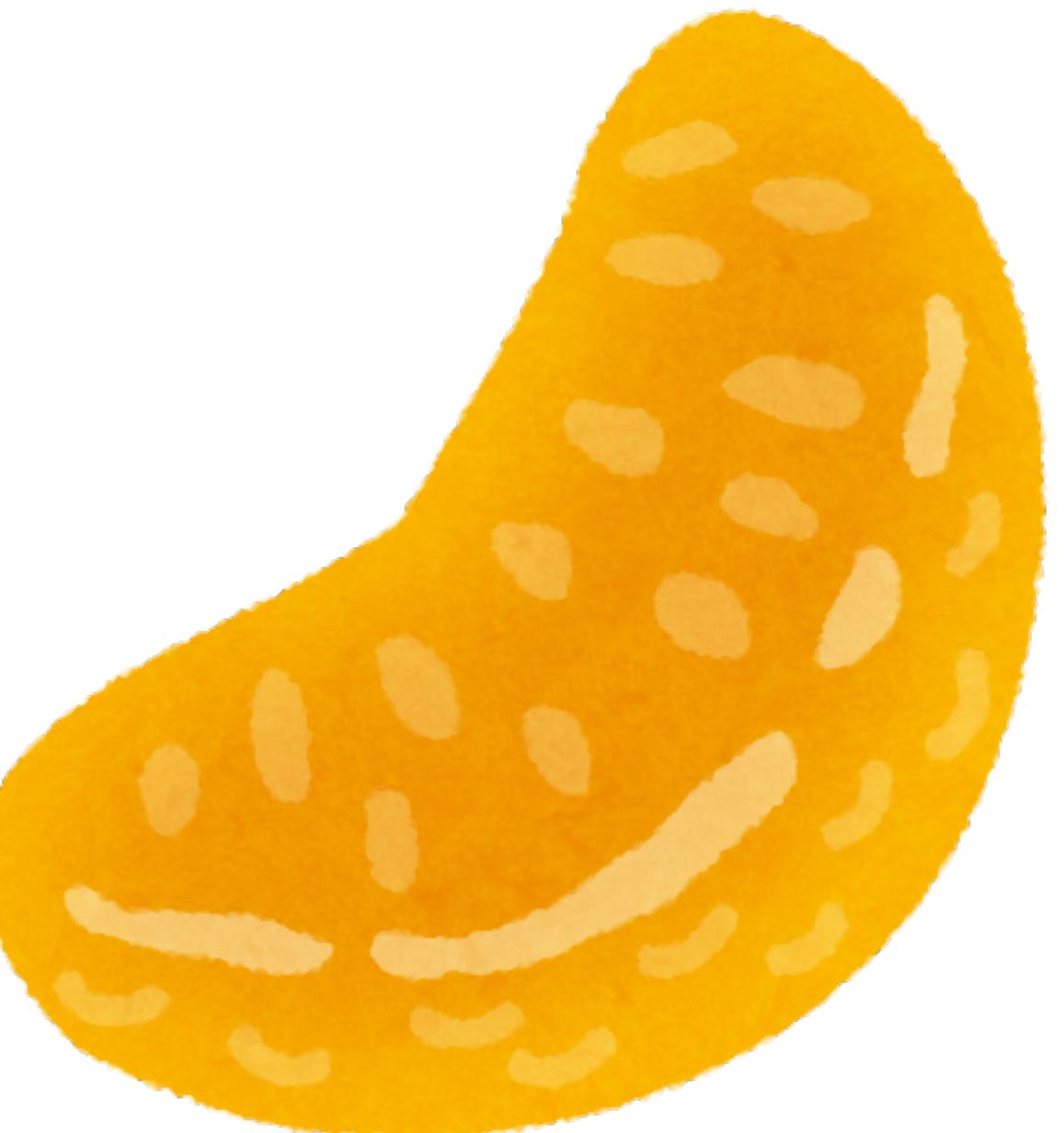
Access to the device with just its target address. If the device returned ACK without data. If the device returned NACK, self.live=False. If device returned PEC error, self.pec_error=True.

`read_registers(reg, length, repeated_start=True)` [source]

reading registers

Reference

mikan



Reference (J)

Title	URL
mikan	https://github.com/teddokano/mikan
Arduino® Shieldsソリューション	https://www.nxp.jp/design/development-boards/analog-toolbox/arduino-shields-solutions:ARDUINO-SOLUTIONS
MIMXRT1050-EVKB	https://www.nxp.jp/design/development-boards/i-mx-evaluation-and-development-boards/i-mx-rt1050-evaluation-kit:MIMXRT1050-EVK
MicroPython	https://micropython.org/
MicroPython Quick reference for the i.MXRT family	https://docs.micropython.org/en/latest/mimxrt/quickref.html
MicroPython Pinout for the i.MXRT machine modules	https://docs.micropython.org/en/latest/mimxrt/pinout.html#mimxrt-i2c-pinout
MicroPython MicroPythonとCPythonの違い	https://micropython-docs-ja.readthedocs.io/ja/latest/genrst/index.html
MicroPython Github	https://github.com/micropython/micropython
マイコン試作&学習 MicroPython教科書 Interface 2023年3月号	https://interface.cqpub.co.jp/magazine/202303/

