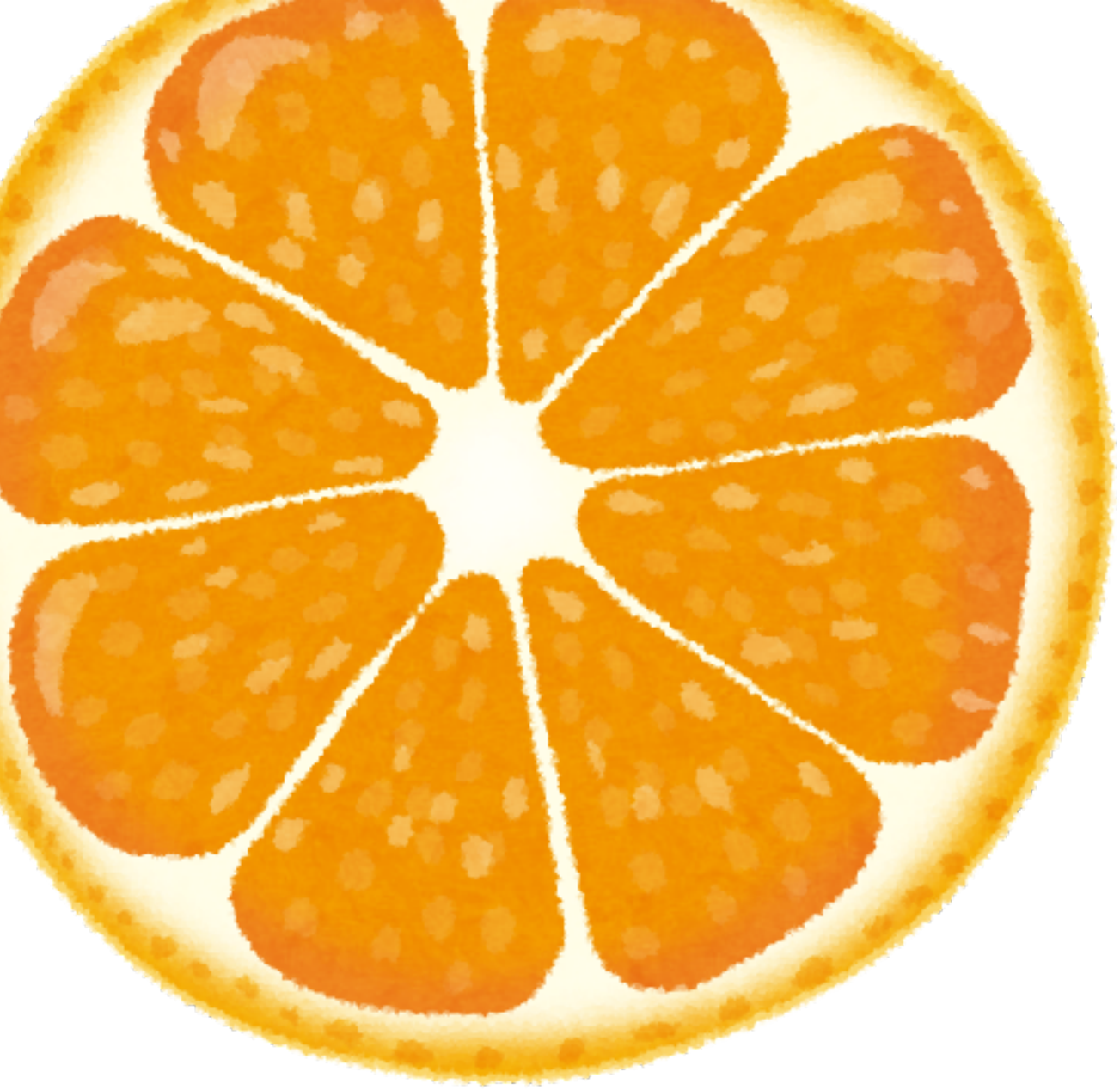
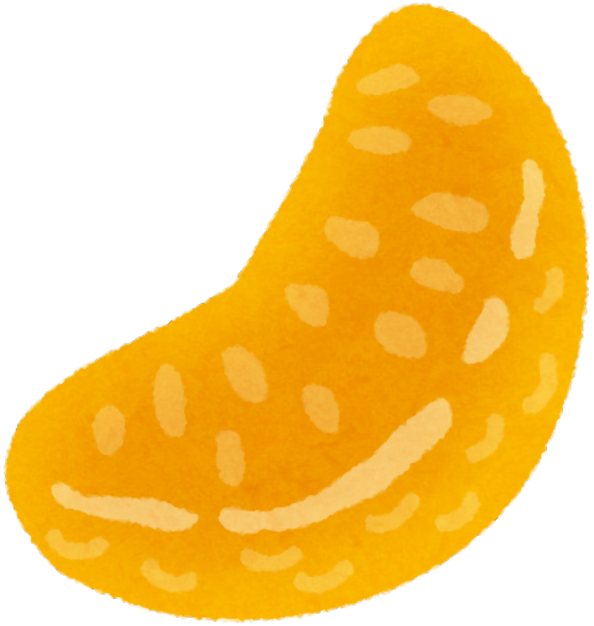


mikan

v1.12

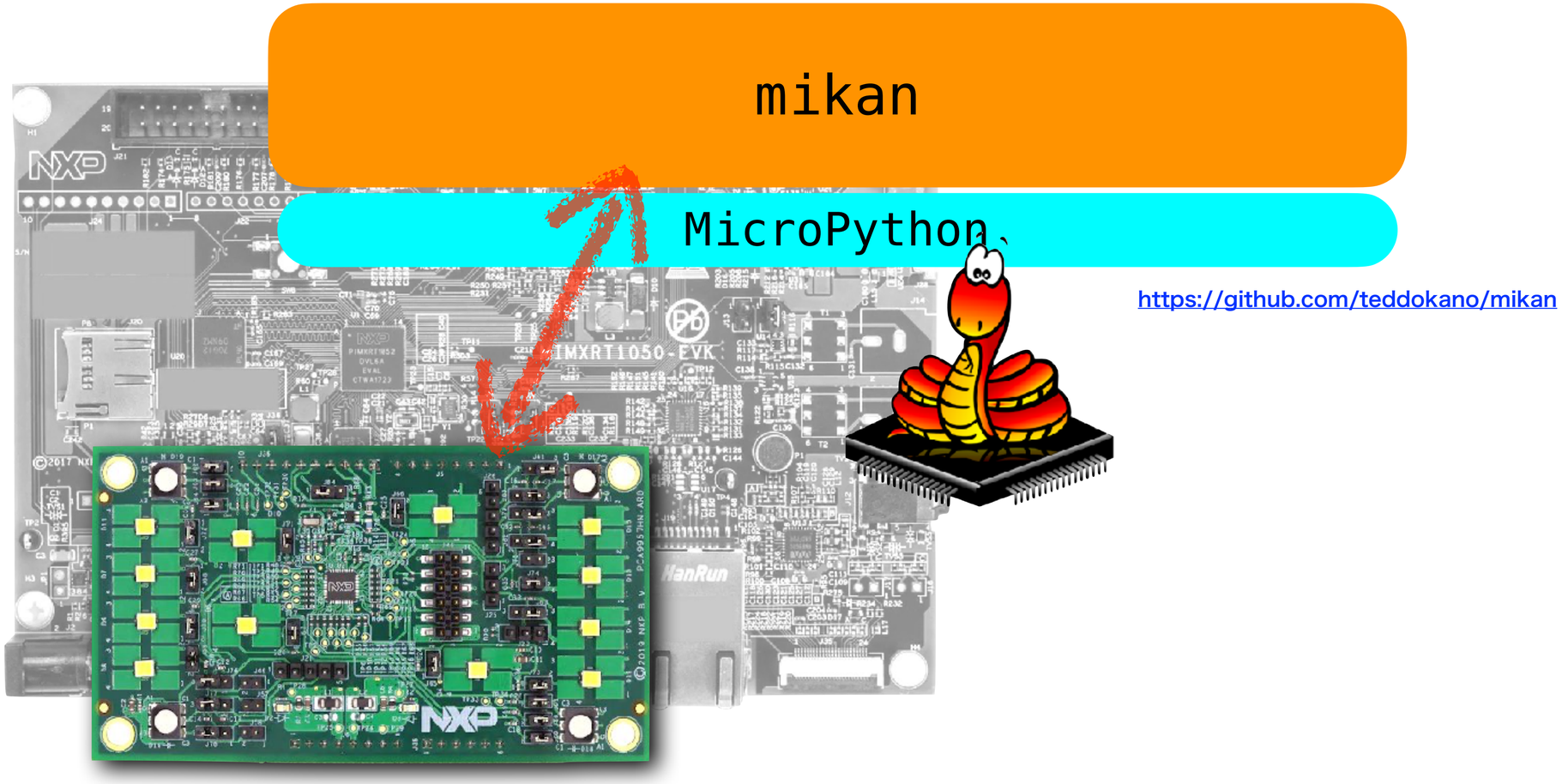
Document version: 1.12 - 2023 Dec 21

mikan?



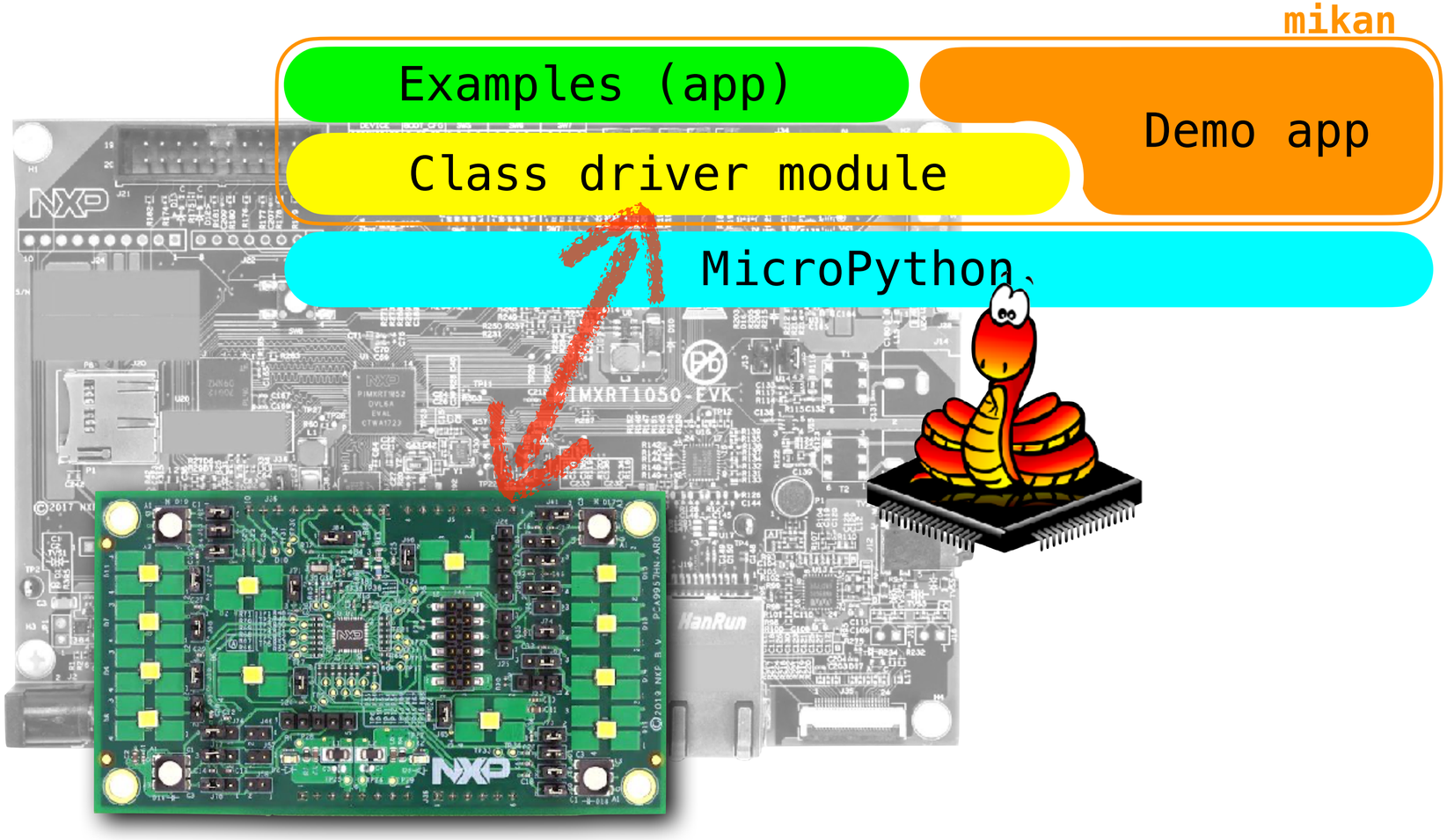
mikan?

Peripheral device driver for collection for MicroPython



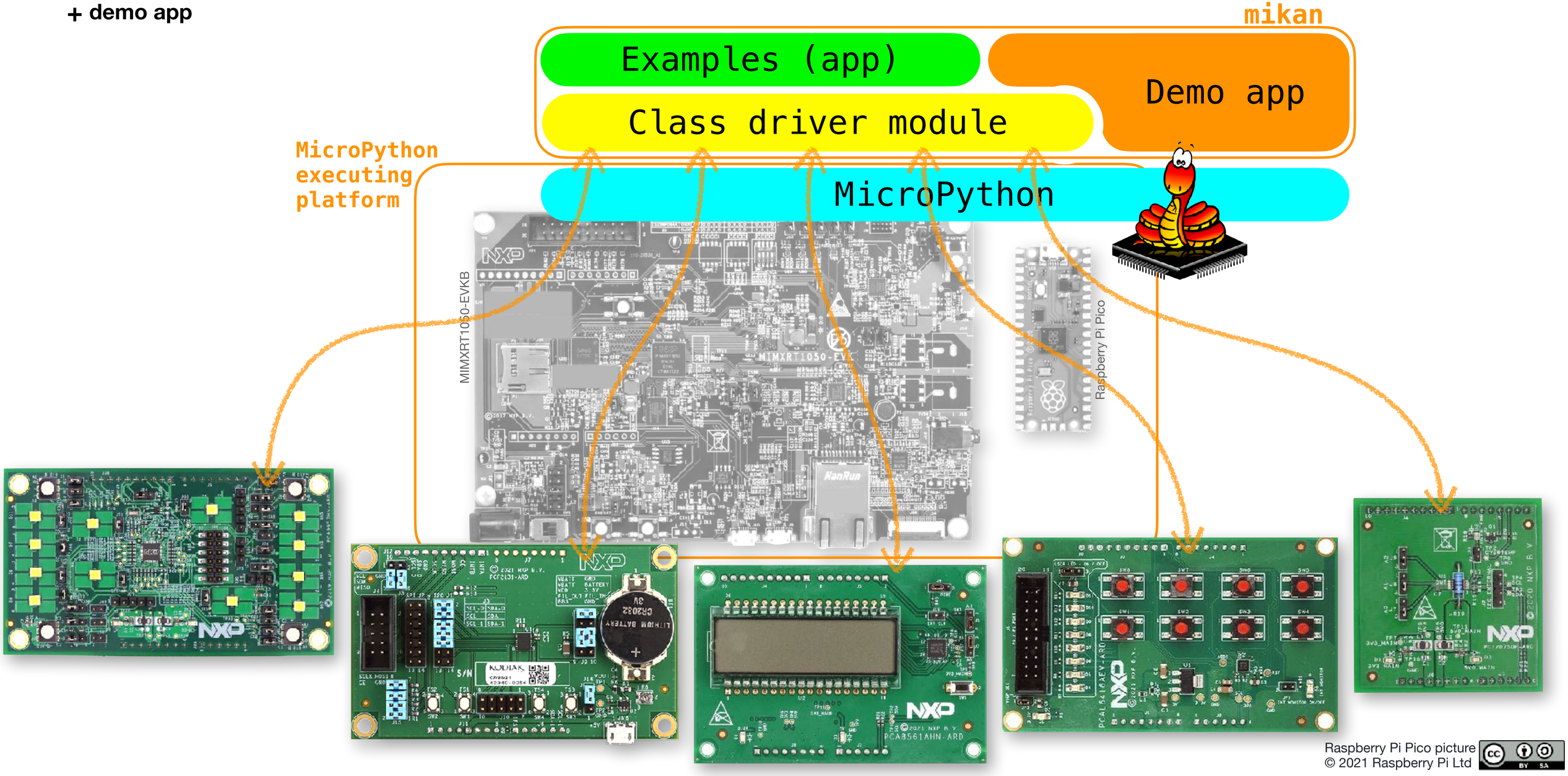
mikan?

Peripheral device driver for collection for MicroPython



mikan?

Peripheral device driver for collection for MicroPython
+ example code
+ demo app



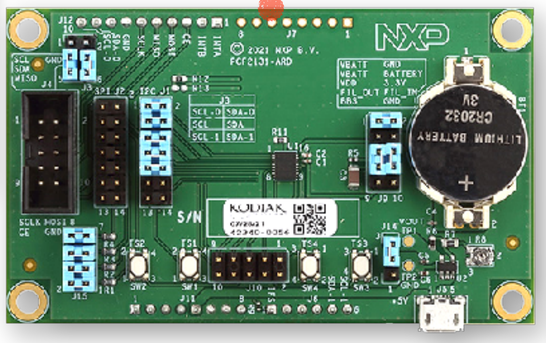
Raspberry Pi Pico picture
© 2021 Raspberry Pi Ltd

mikan?

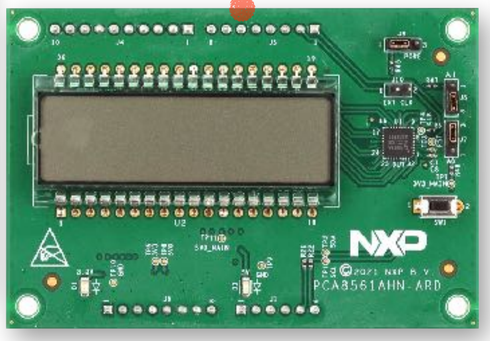
Enables quick evaluation/test of I²C/SPI devices and rapid prototyping
Can be written by simple Python code

```
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 )
```

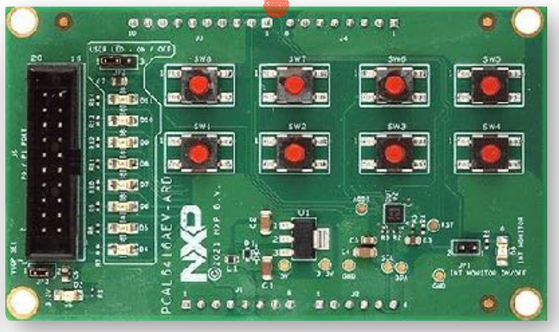
```
from machine    import I2C
from nxp_periph import PCF2131

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

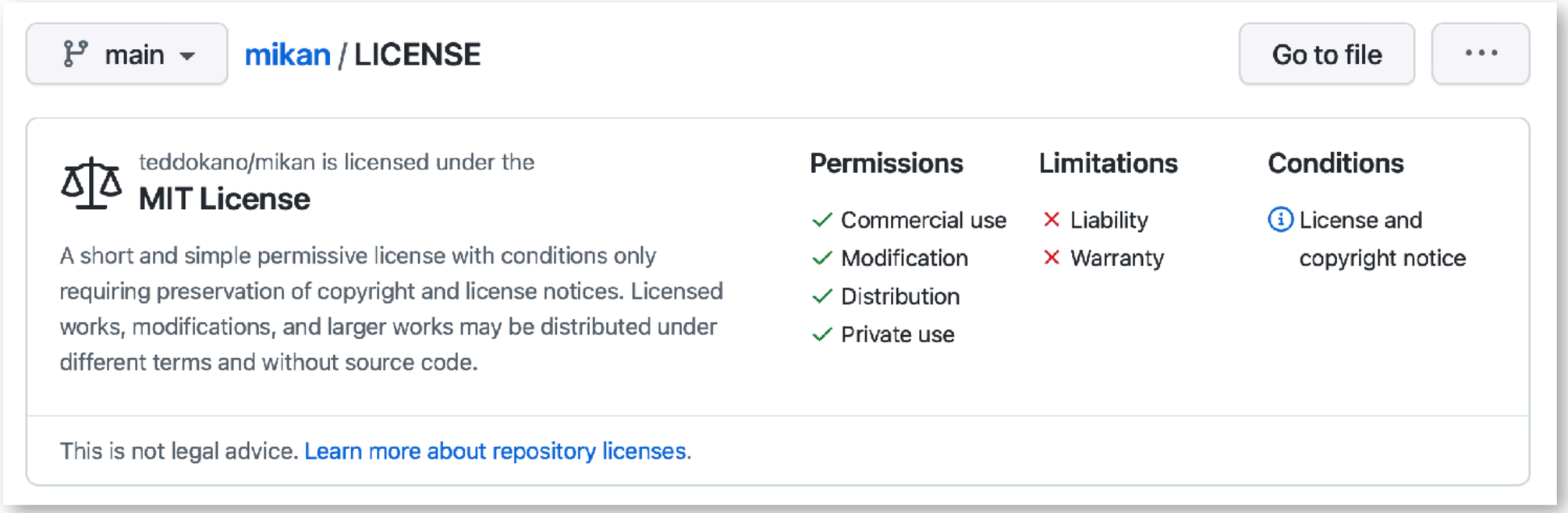
```
from machine    import I2C
from nxp_periph import PCT2075


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

← Prepare to use class libraries

← Make a class instance

← Call an instance method and print

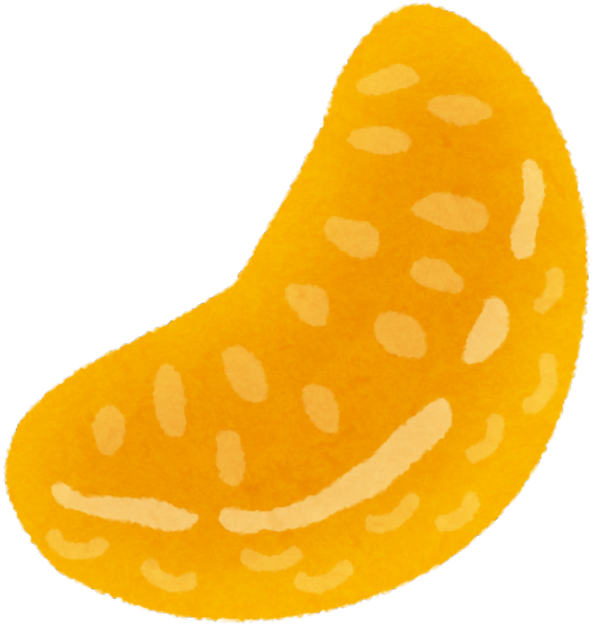
License



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

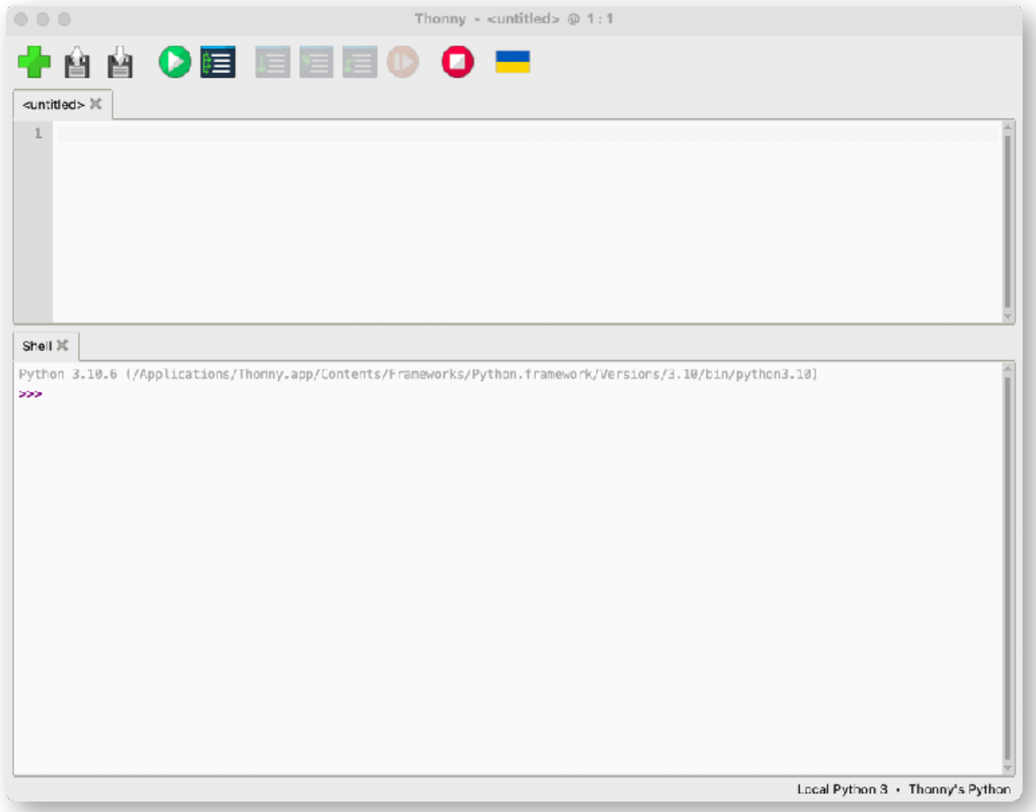
Get ready

mikan

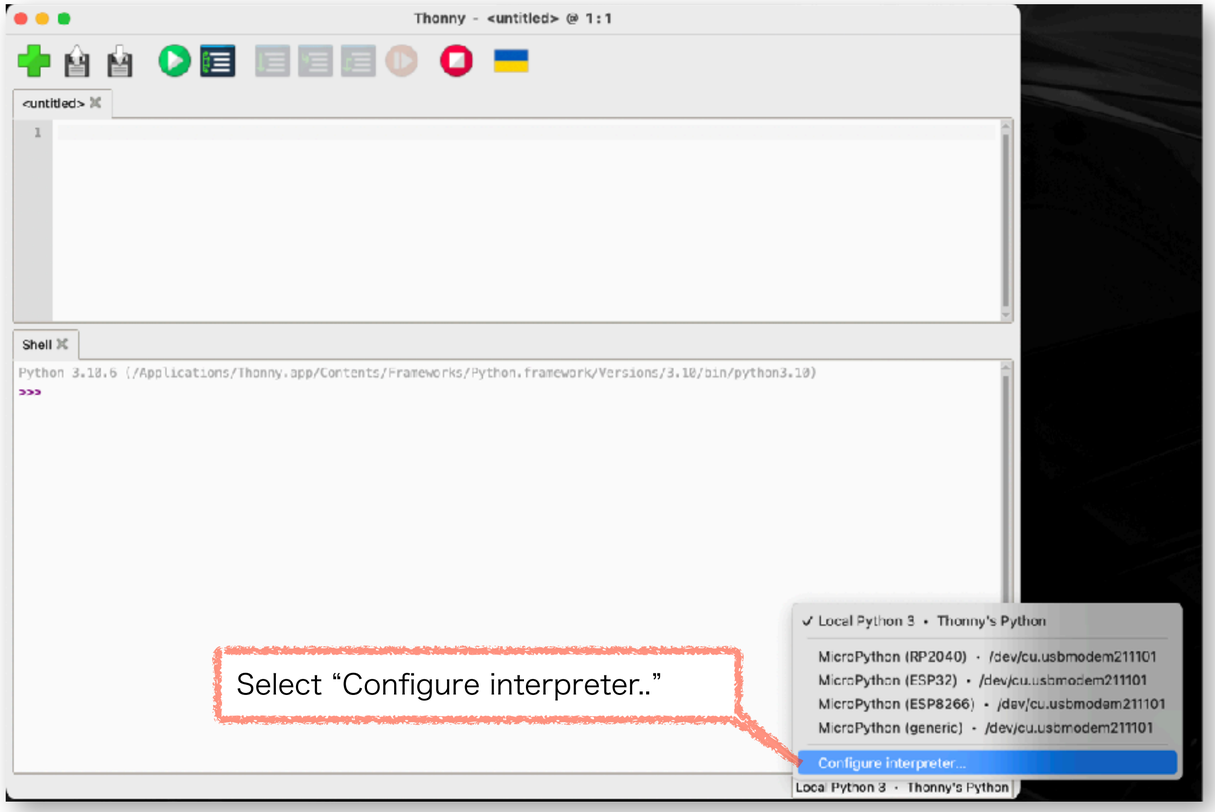


MicroPython executing environment setup

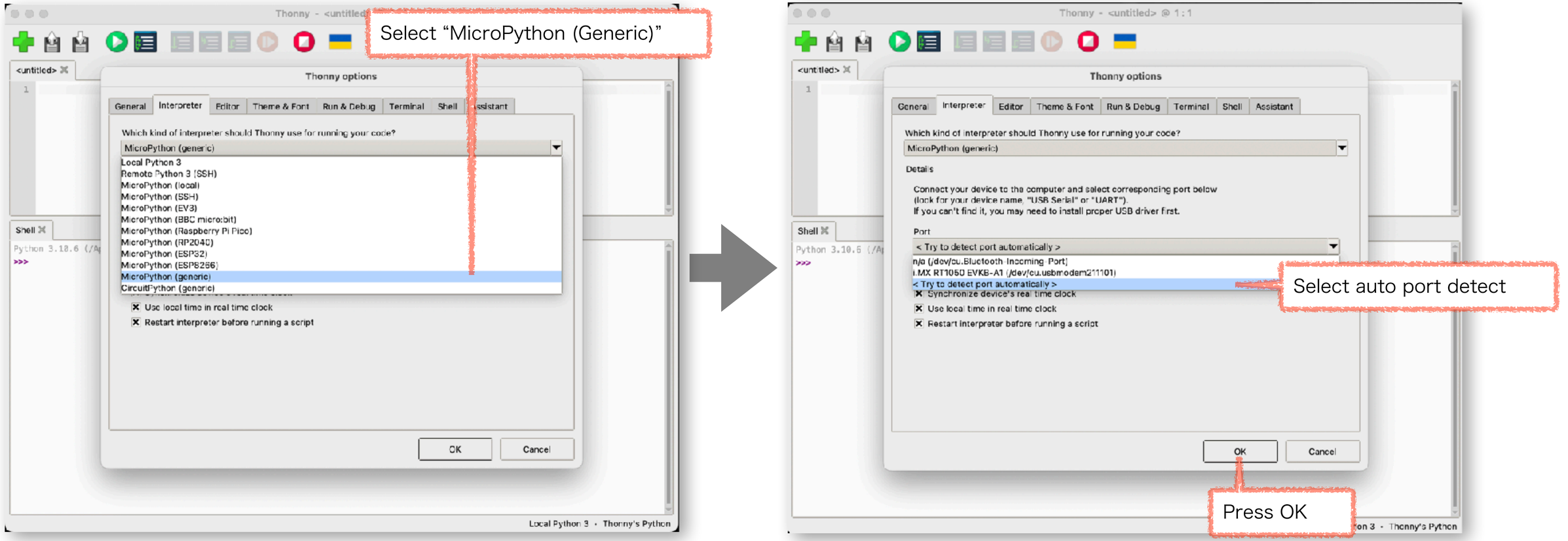
Install Thonny and launch



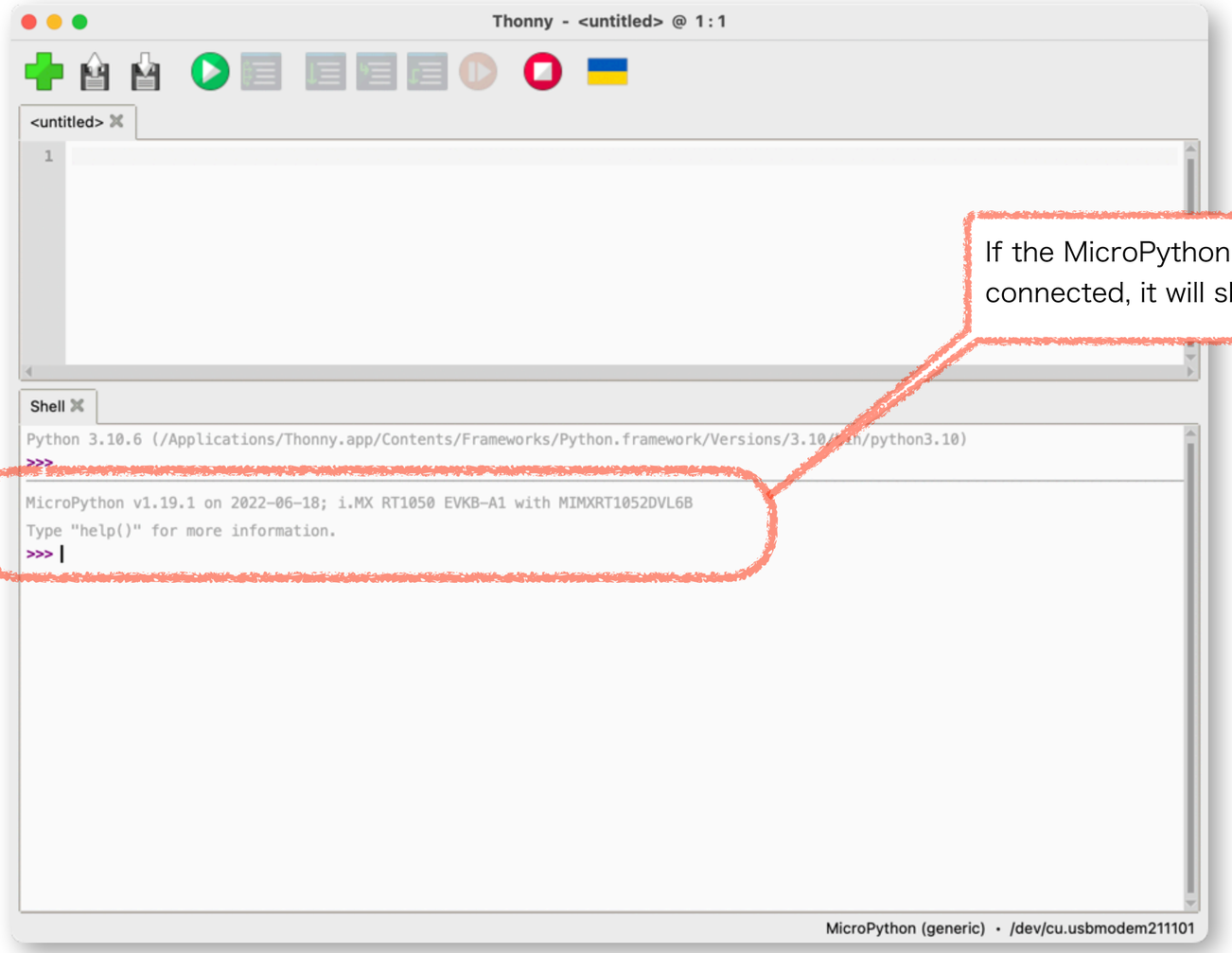
Click right-bottom pop-up menu



MicroPython executing environment setup



MicroPython executing environment setup

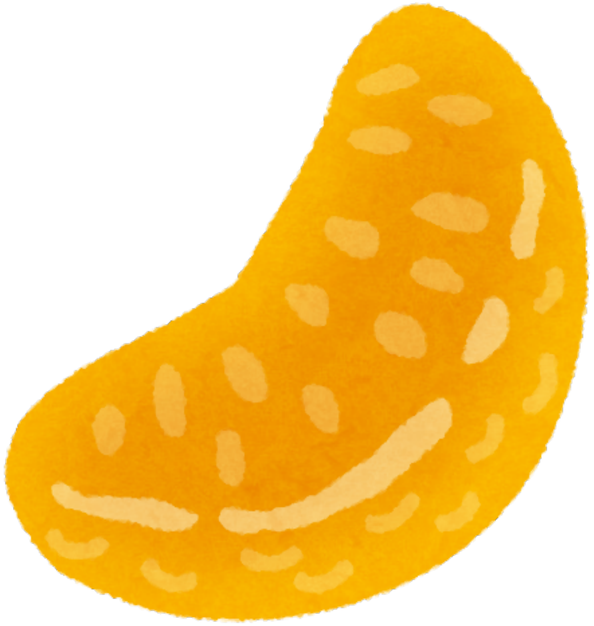


If the MicroPython installed IMXRT1050-EVKB is connected, it will show MicroPython message and prompt.

Now MicroPython is ready to run!

Basics of MicroPython

mikan



MicroPython?

https://micropython.org/

micropython.org

MicroPython DOWNLOAD DOCS DISCORD DISCUSSIONS WIKI STORE

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?

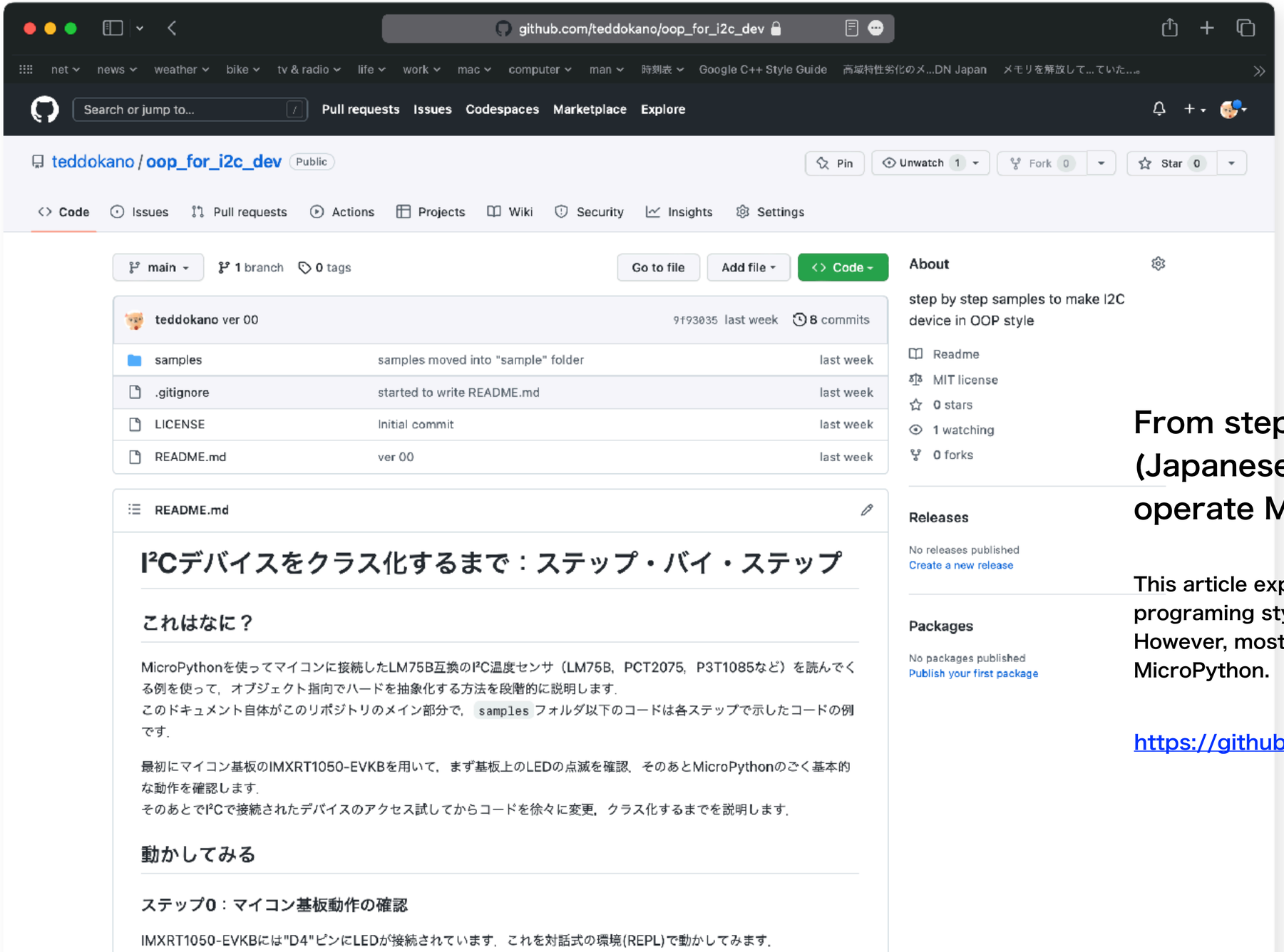
It’s popular!

- 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



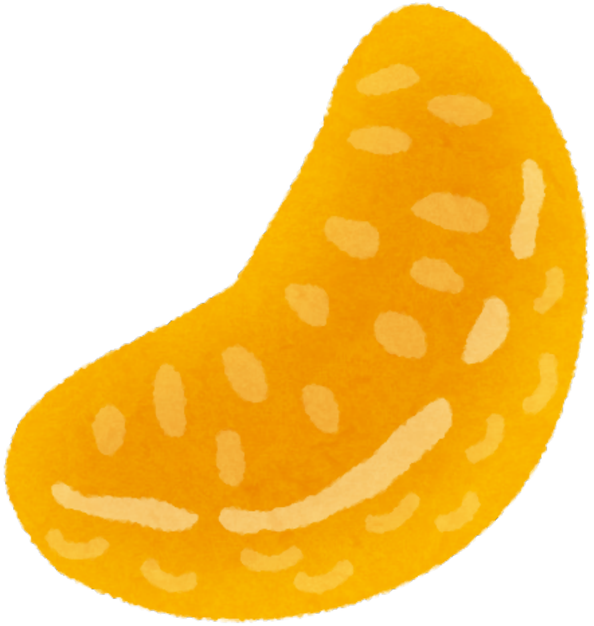
From step0 to step8 in [this sample code](https://github.com/teddokano/oop_for_i2c_dev) (Japanese) could be good samples to operate MicroPython

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

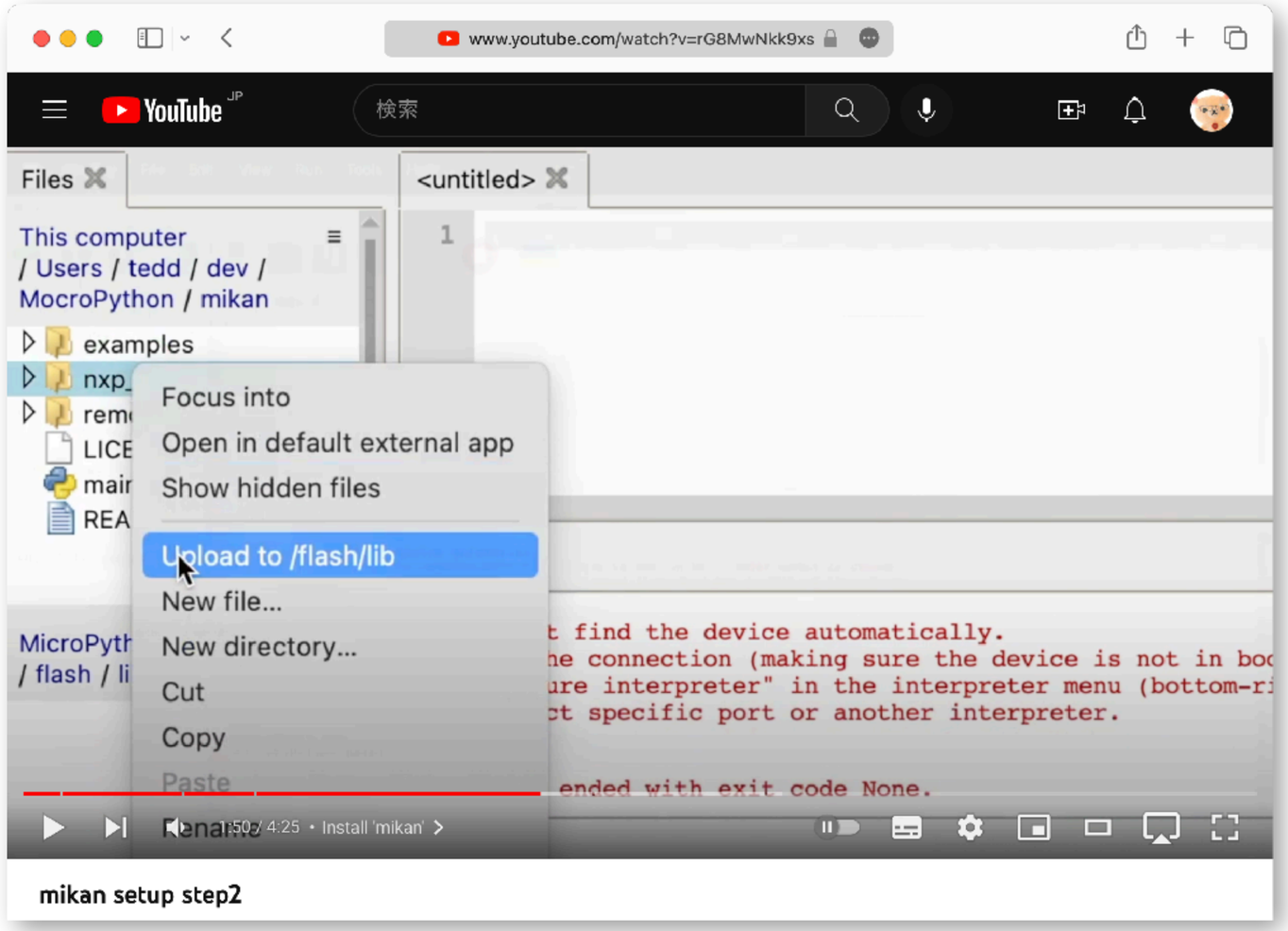
class driver

mikan



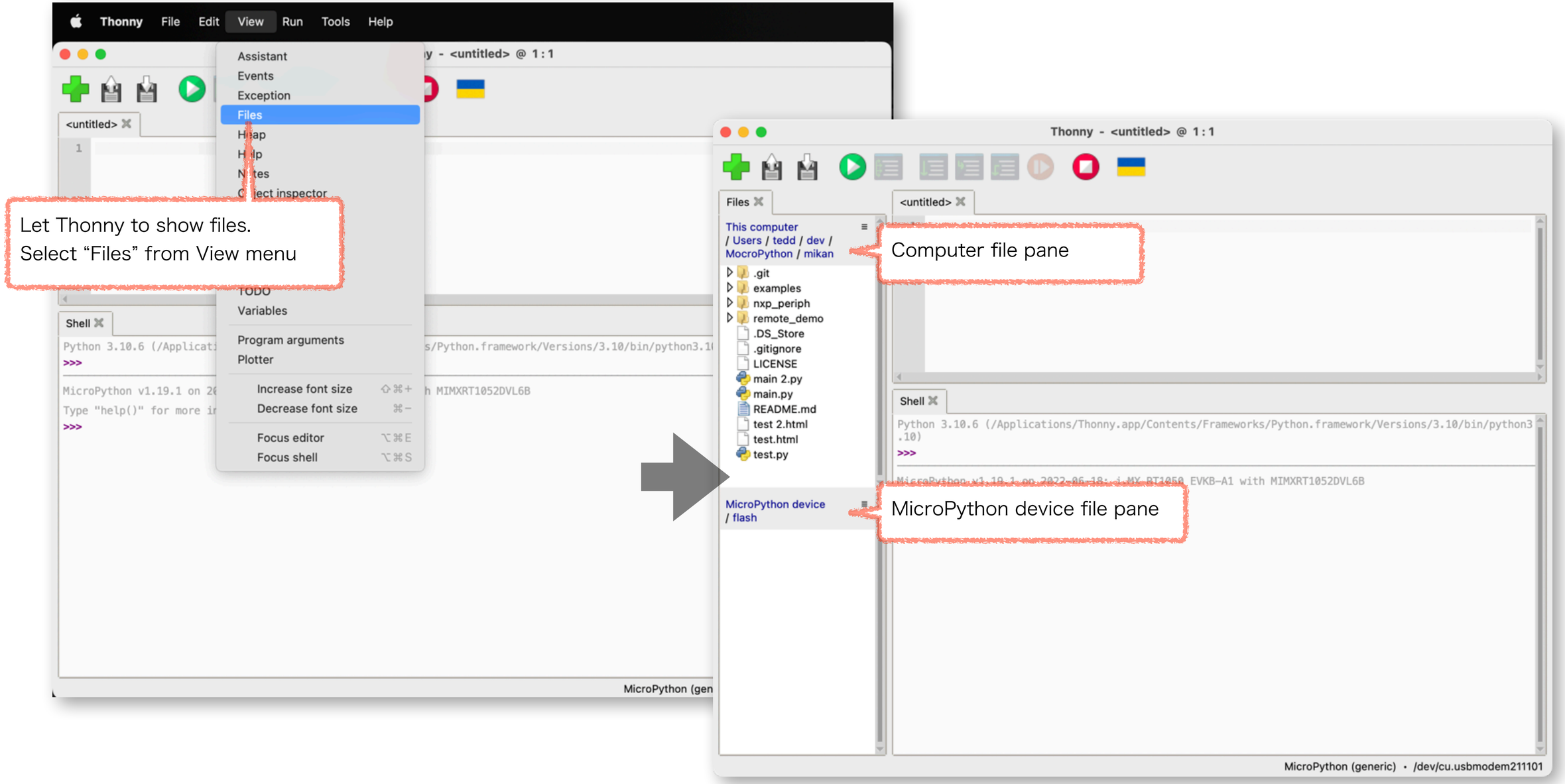
mikan install

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

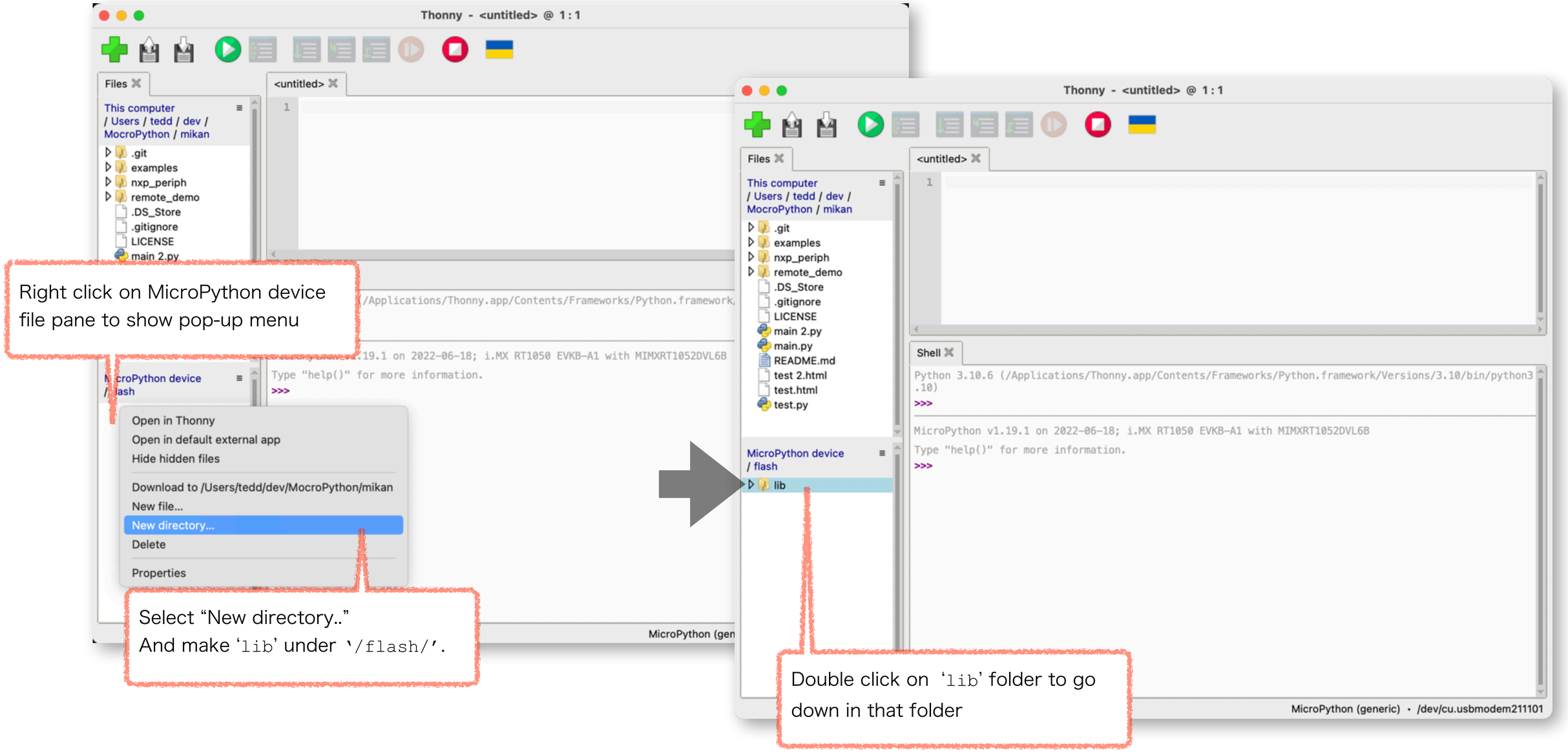


https://github.com/teddokano/mikan

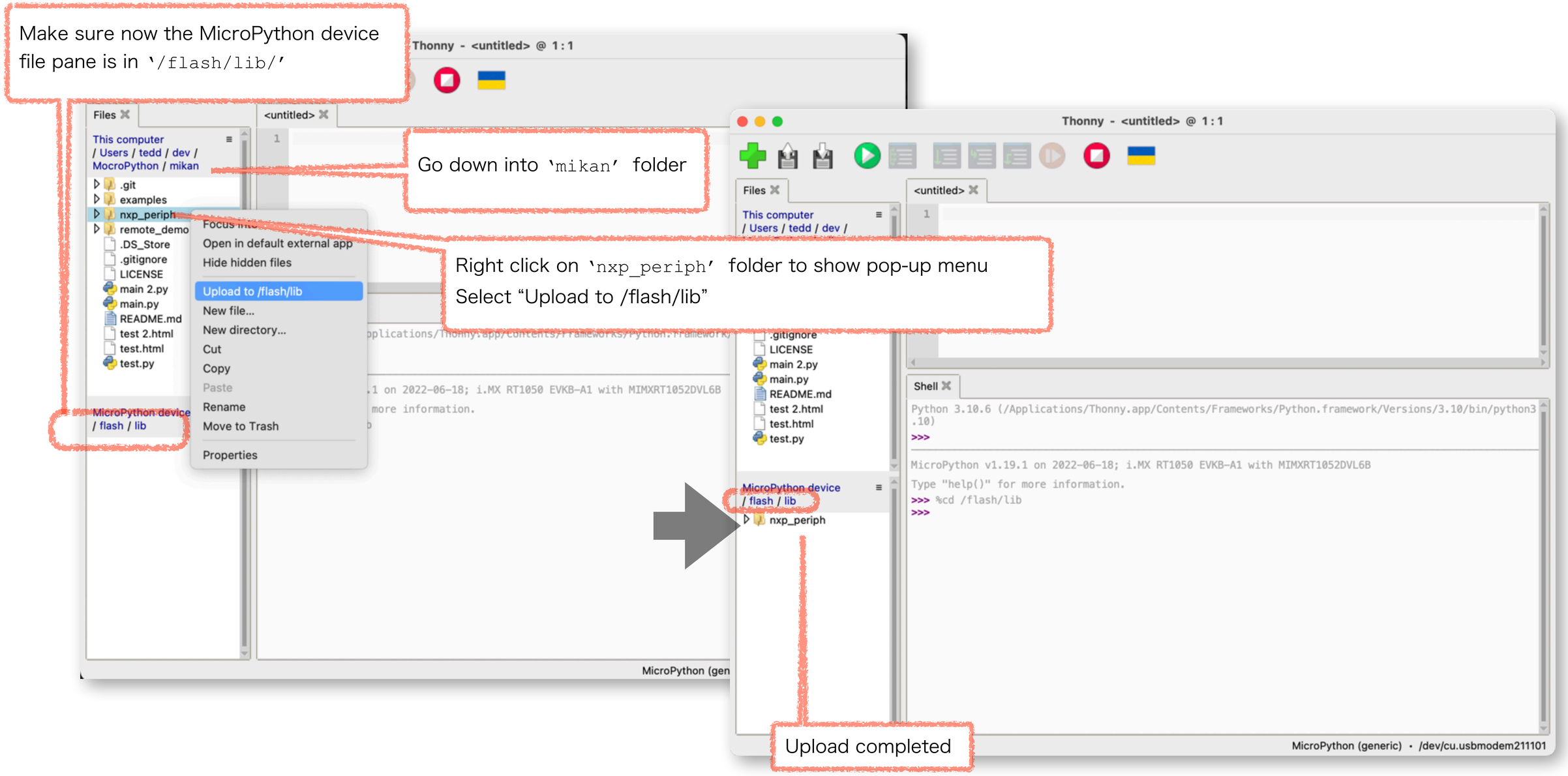
Installing mikan class library



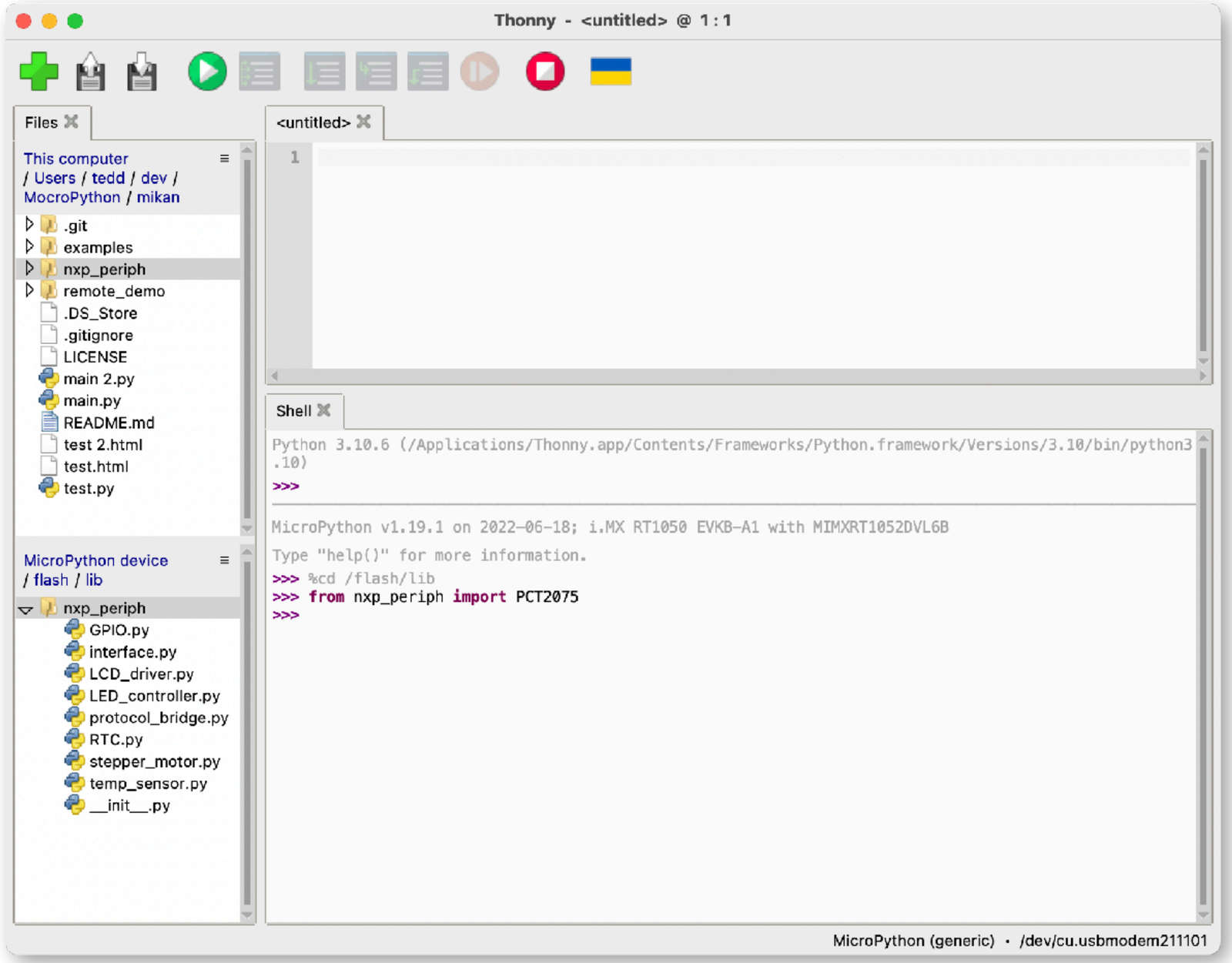
Installing mikan class library



Installing mikan class library

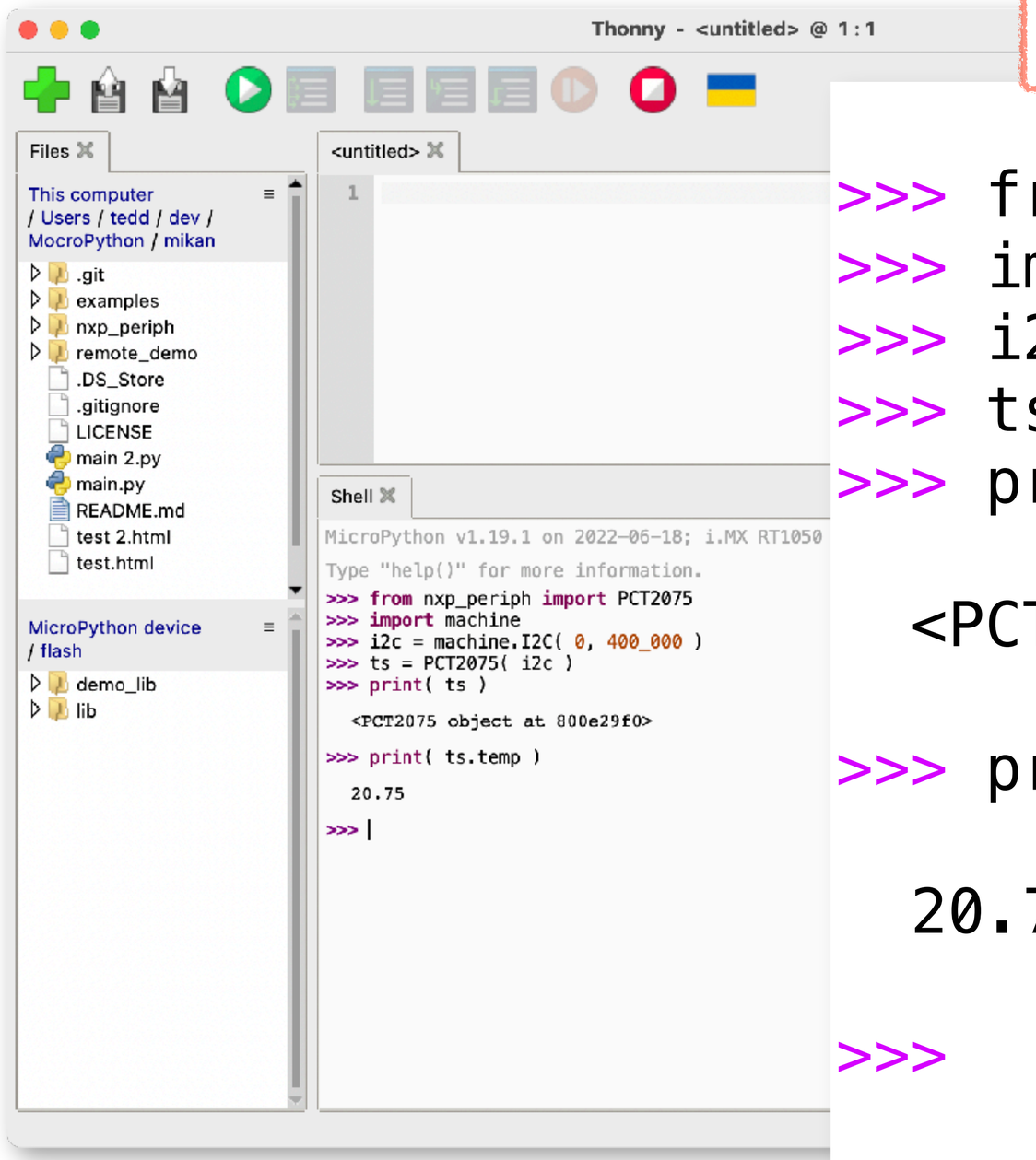


Installing mikan class library



mikan class library is ready!

Basic mikan operation

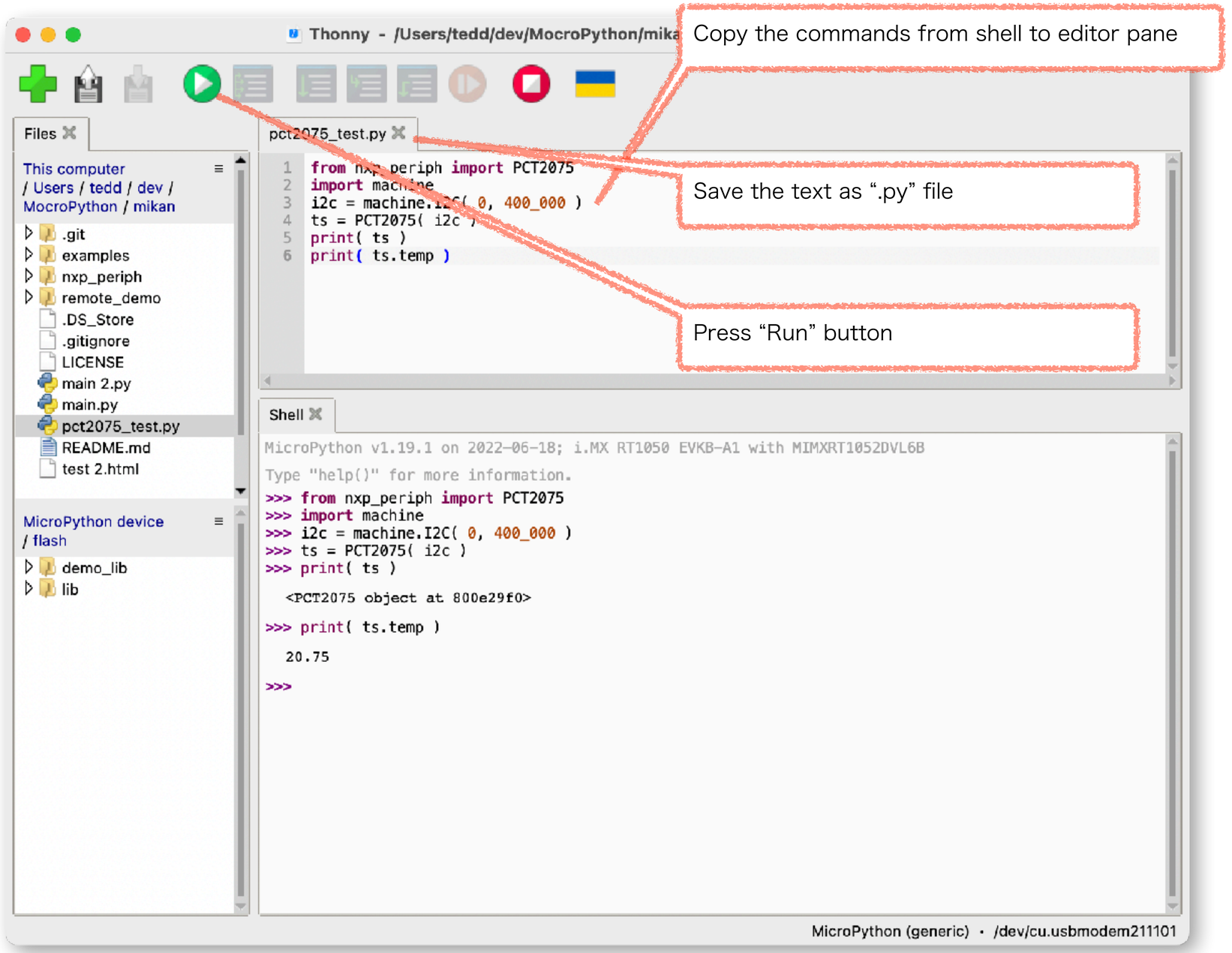


Type-in these commands after prompt (“>>>”) in shell window.

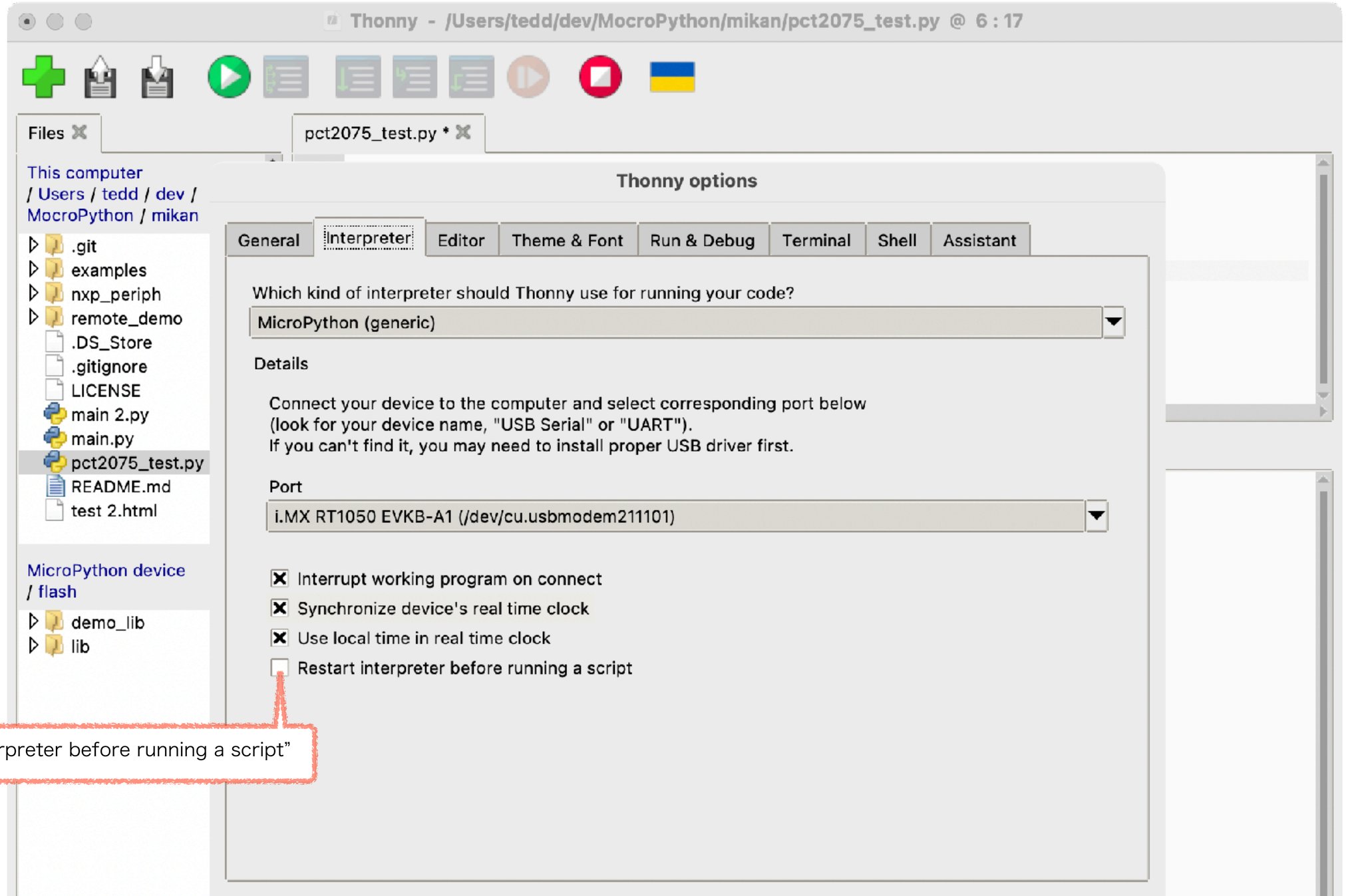
```
>>> from nxp_periph import PCT2075
>>> import machine
>>> i2c = machine.I2C( 0 )
>>> ts = PCT2075( i2c )
>>> print( ts )

  <PCT2075 object at 800e29f0>

>>> print( ts.temp )

  20.75

>>>
```

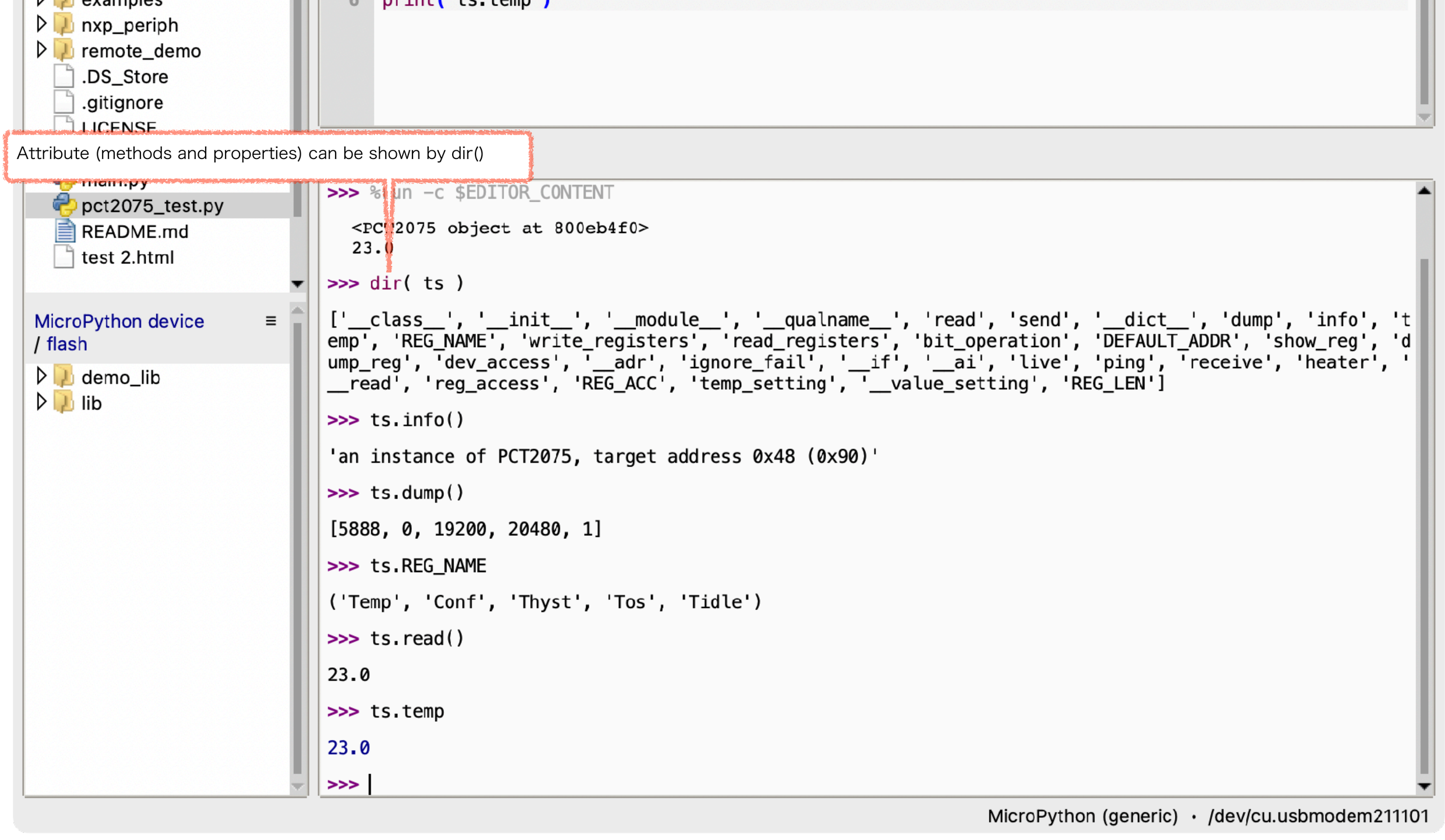
Temperature [˚C] will be shown

Basic mikan operation



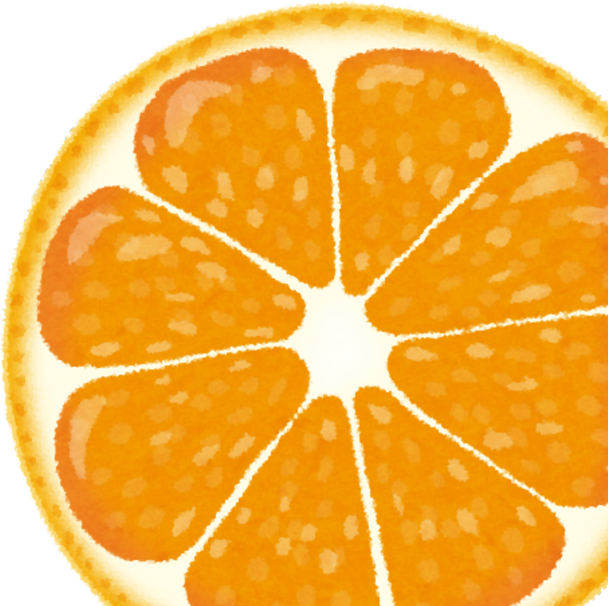
Basic mikan operation




print(ts.temp)
nxp_periph
remote_demo
.DS_Store
.gitignore
LICENSE
Attribute (methods and properties) can be shown by dir()
pct2075_test.py
README.md
test 2.html
MicroPython device
/ flash
demo_lib
lib
>>> %run -c $EDITOR_CONTENT
<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
>>>
MicroPython (generic) • /dev/cu.usbmodem211101

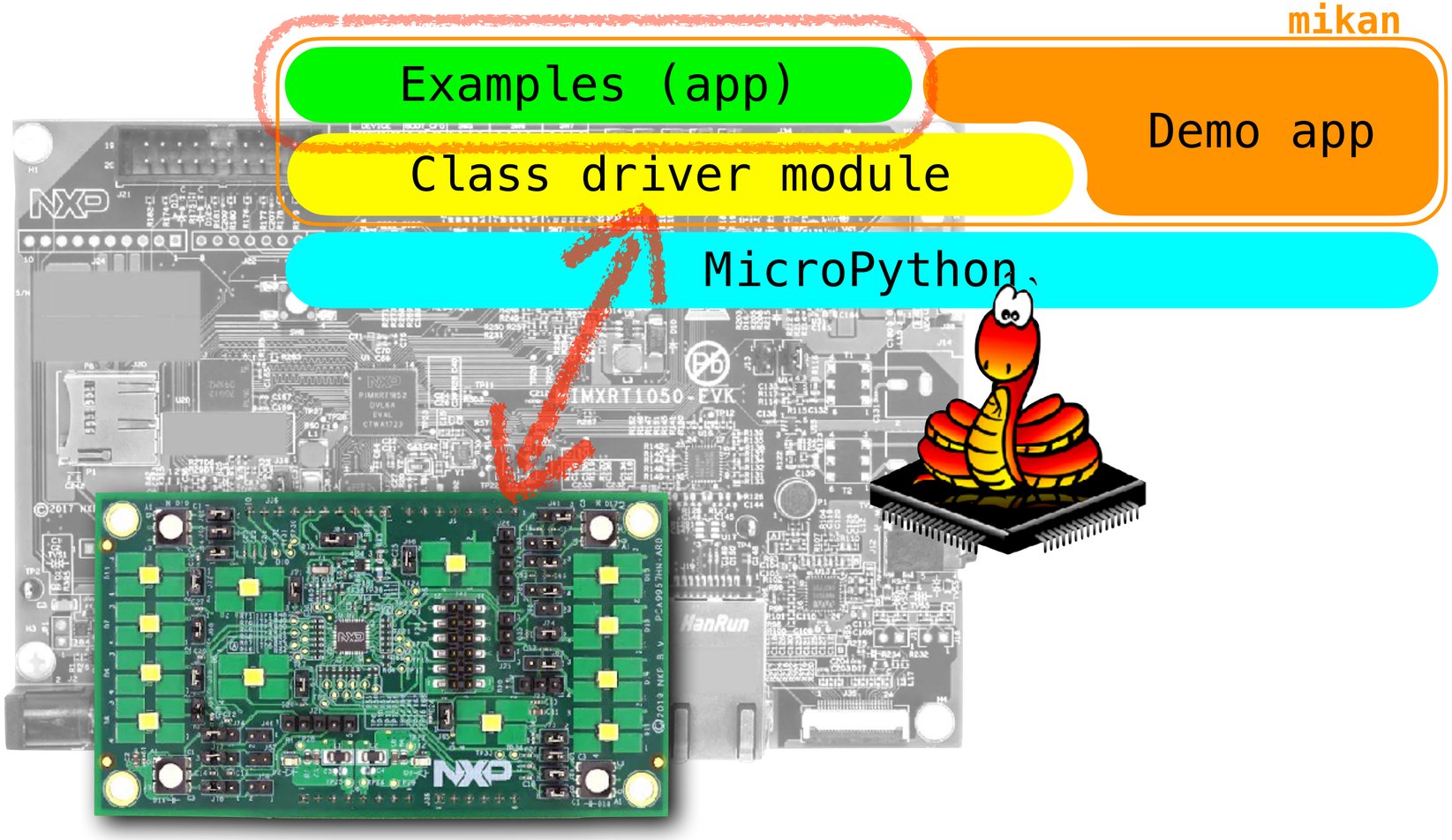
examples/

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

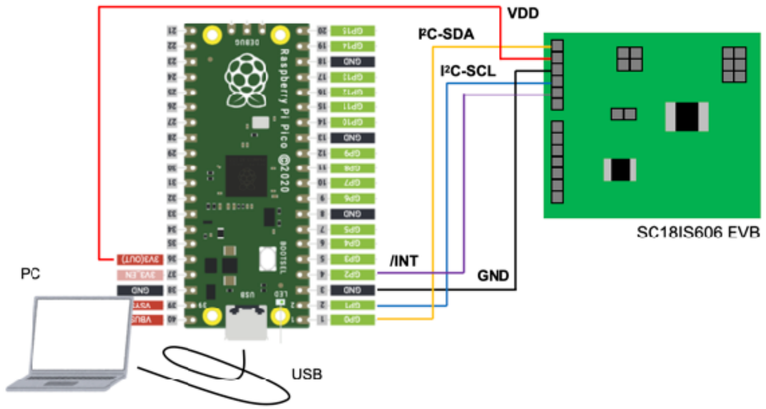
Category	File name	Feature	Target
LED	LED_controller.py	Simple sample: making an LED_controller instance and how PWM can be controlled	PCA9955B, PCA9956B, PCA9957, PCA9632
LED	LED_gradation_ctrl.py	Gradation control (hardware) feature demo	PCA9955B, PCA9957
LED	LED_instance.py	`Using another class to 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 I²C/SPI to UART protocol bridge through MicroPython's machine.UART equivalent APIs.	SC16IS7xx
Protocol bridge	protocol_bridge_SC18IS606_with_AT25010.py	Operate an I²C 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

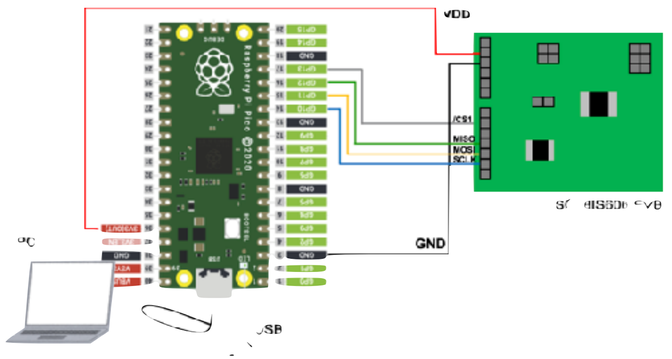
examples?

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





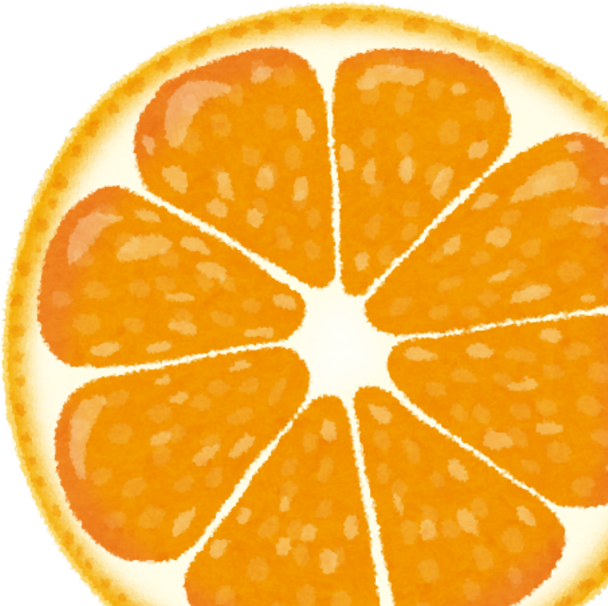
3. Put code to absorb hardware difference

```
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!

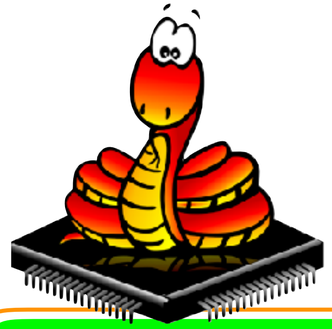
remote_demo/

mikan



remote_demo?

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



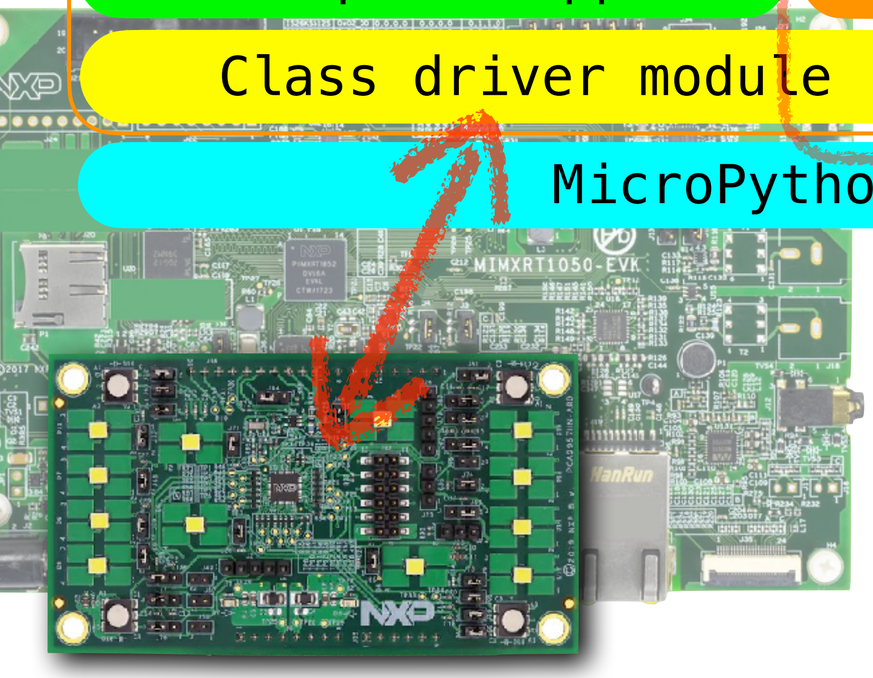
mikan

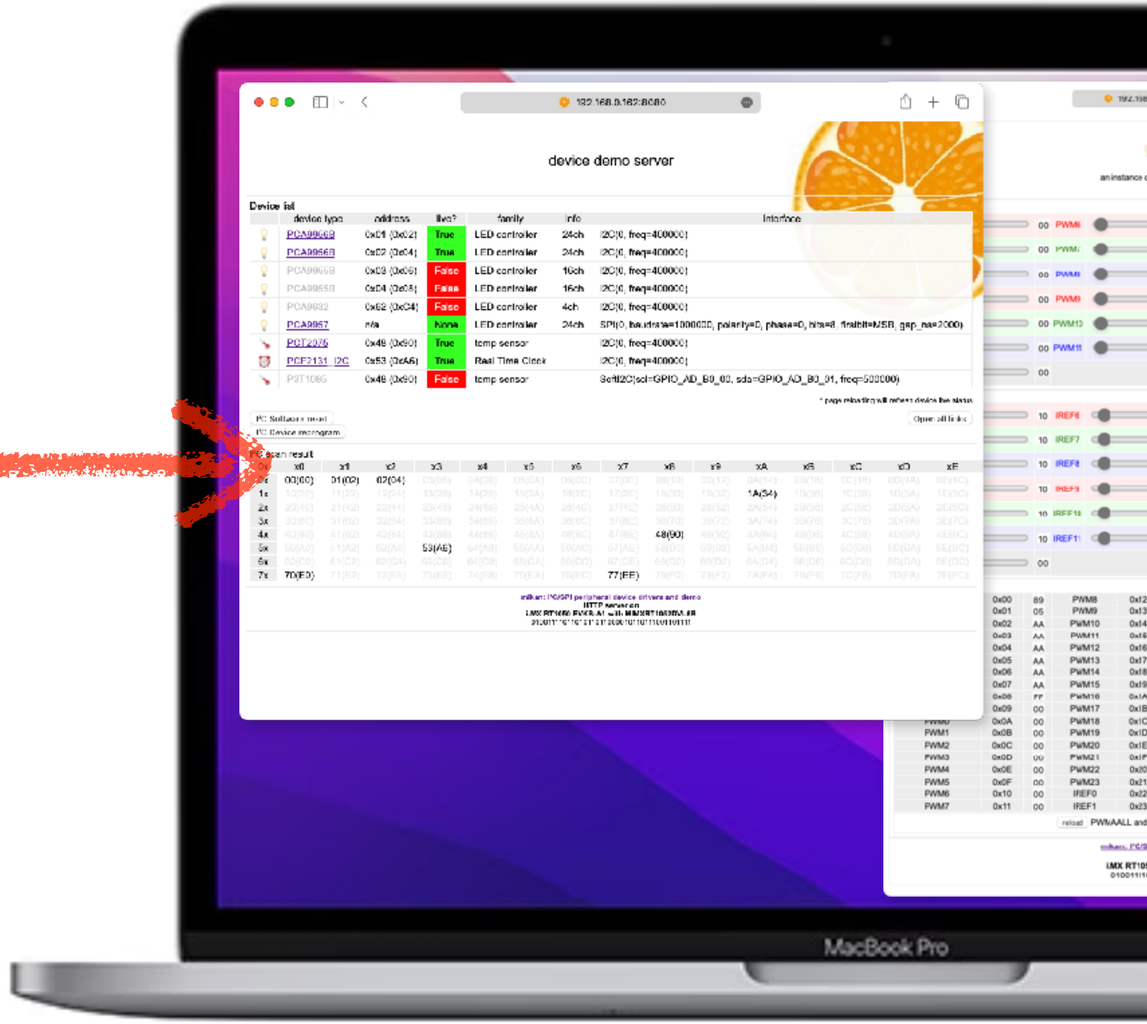
Examples (app)

Demo app

Class driver module

MicroPython



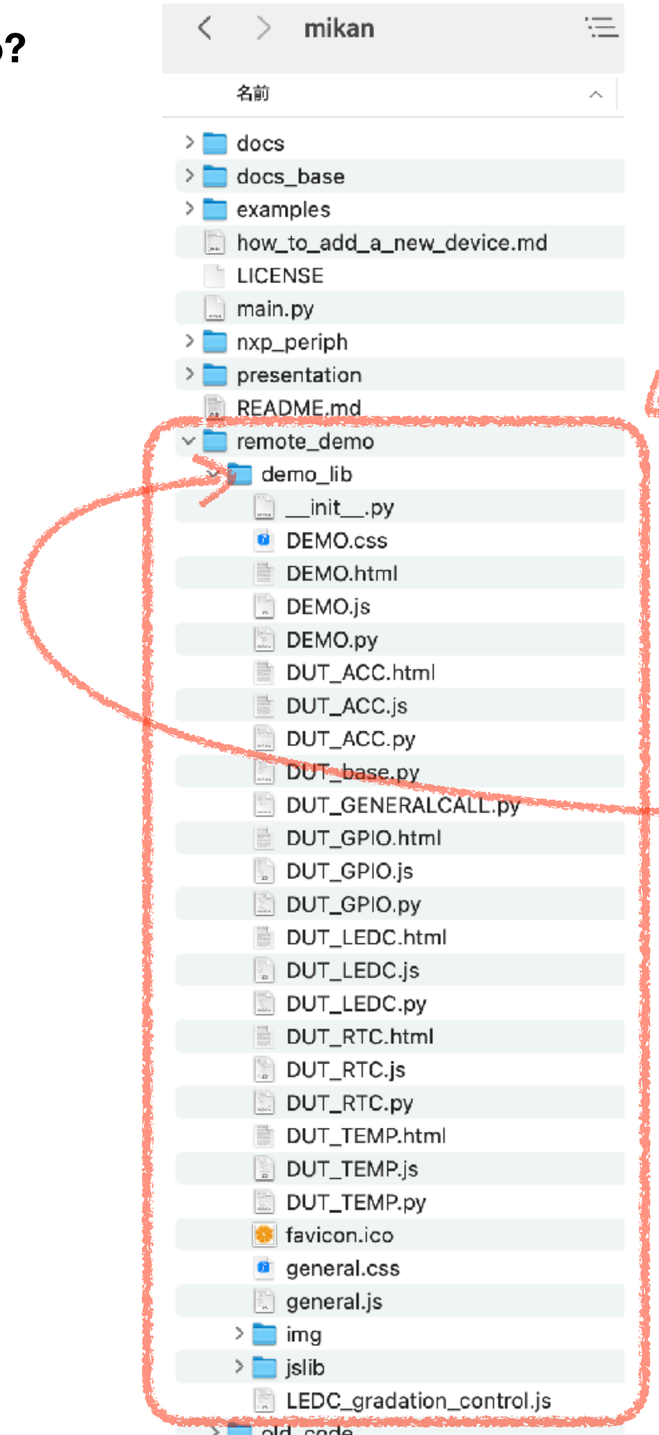


remote_demo?

Supported devices

Category	Target	Feature	mikan class driver	remote_demo support
LED controller	PCA9632	4ch, I²C	✔	yes (no ARD available)
LED controller	PCA9955B	16ch, I²C	✔	yes
LED controller	PCA9956B	24ch, I²C	✔	yes (no ARD available)
LED controller	PCA9957	24ch, SPI	✔	yes
RTC	PCF2131	I²C & 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	I²C/SPI → UART	✔	no (no ARD available)
Protocol bridge	SC18IS606	I²C → 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?

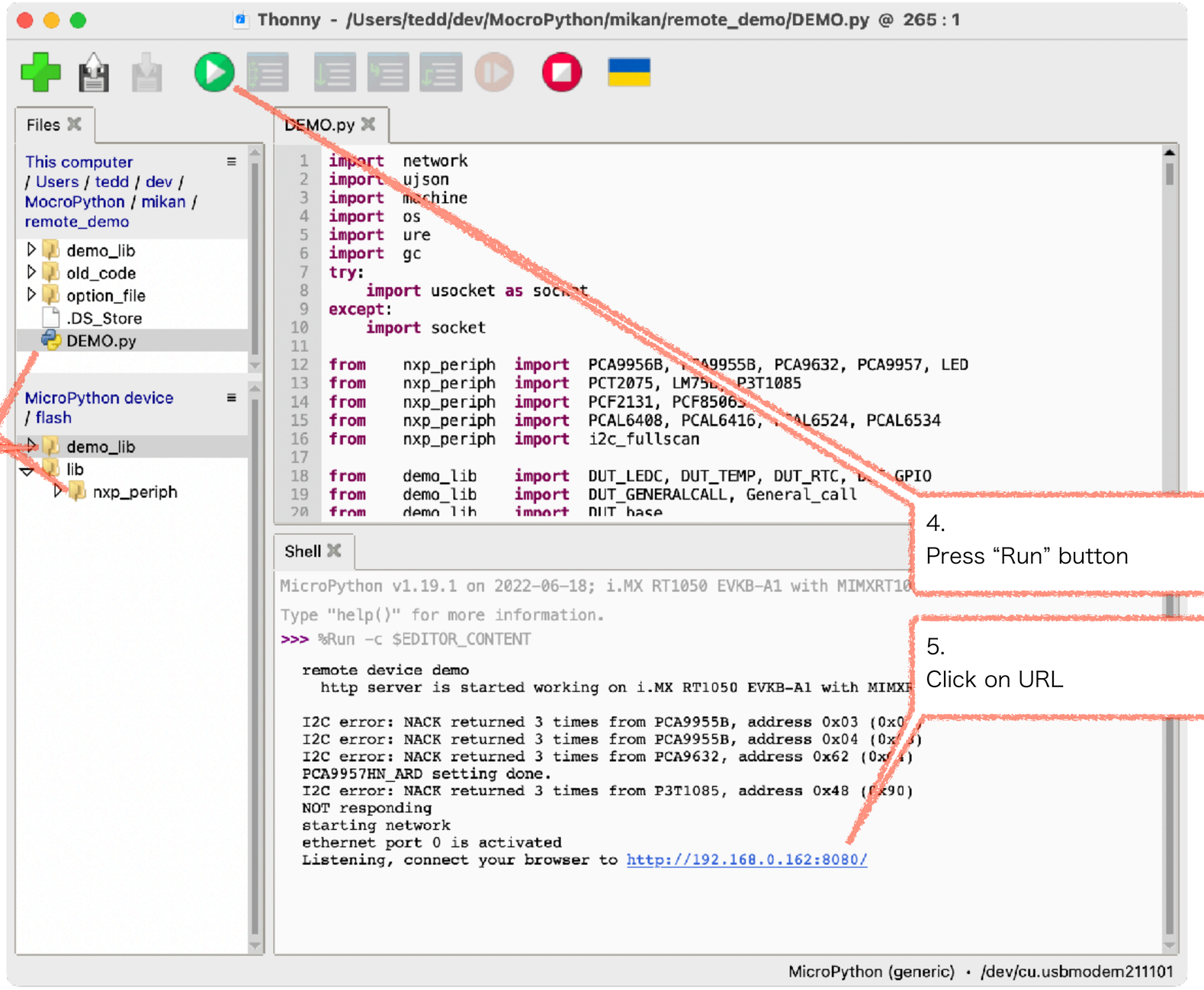
Steps to run

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

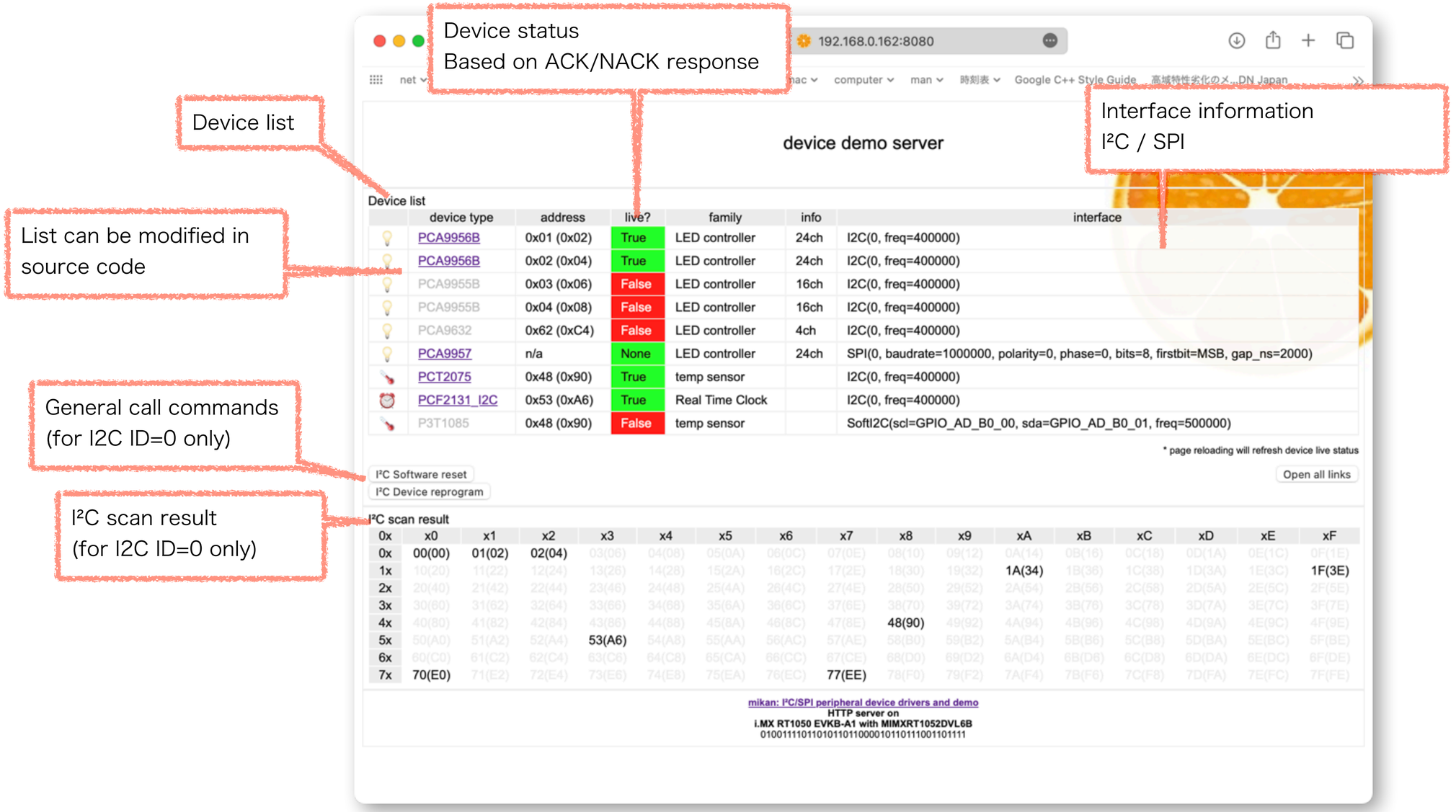


4.
Press “Run” button

5.
Click on URL

remote_demo?

UI on browser: Device list page



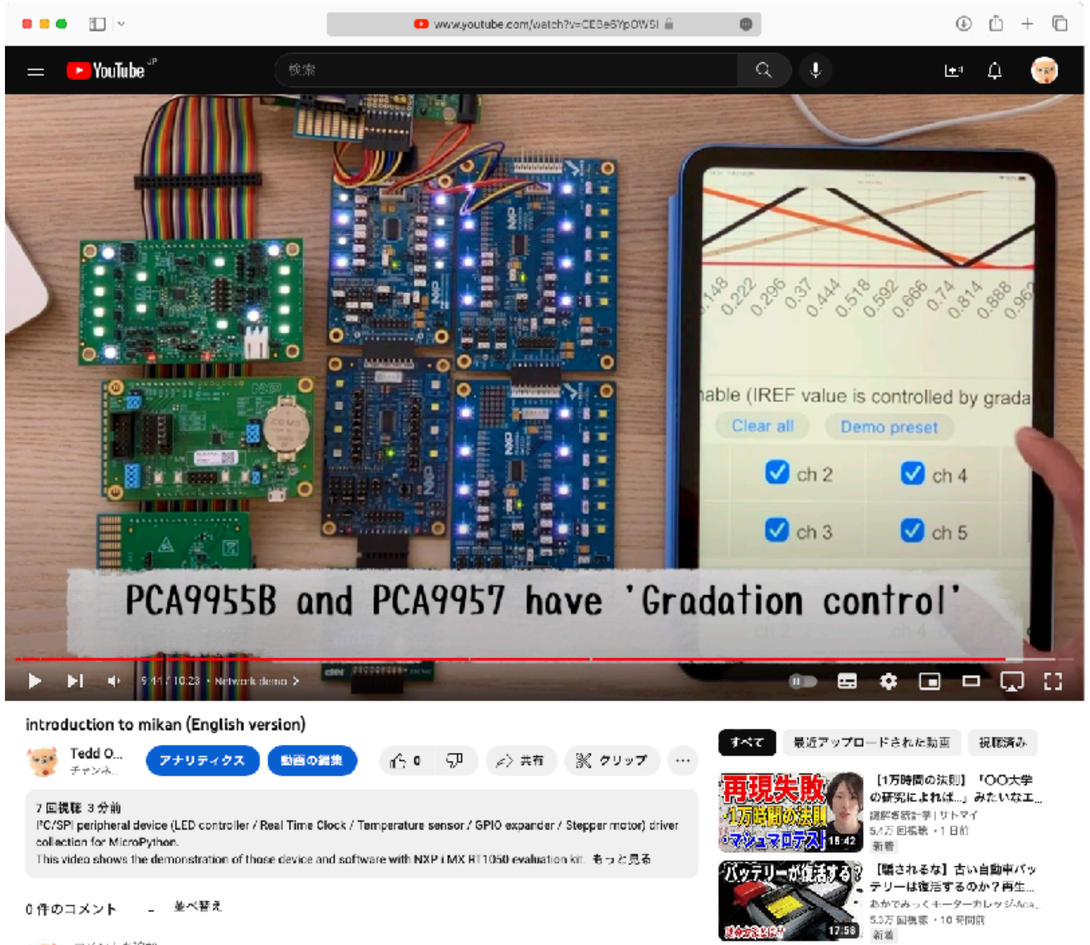
remote_demo?

UI on browser: example of device page

remote_demo?

Demo video available

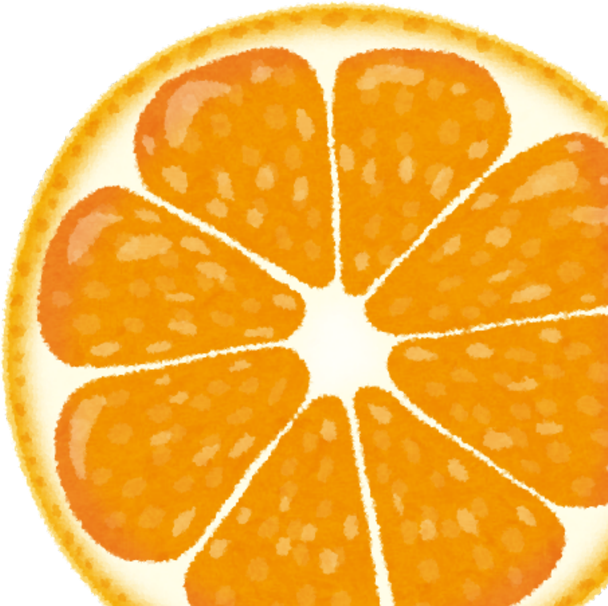
English version
https://youtu.be/CEBe6YpOWSI



Japanese version
https://youtu.be/WYQKIY8iBYM

How does it work?

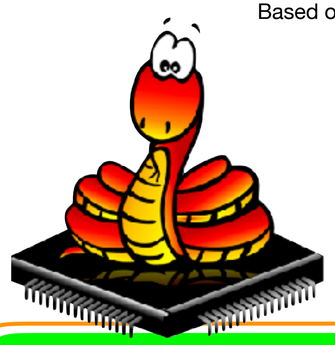
mikan



How does it work? : Applications

Web server application
using ‘mikan’ class driver
to control devices

Based on MicroPython example code



remote_demo/demo_lib/DEMO.py

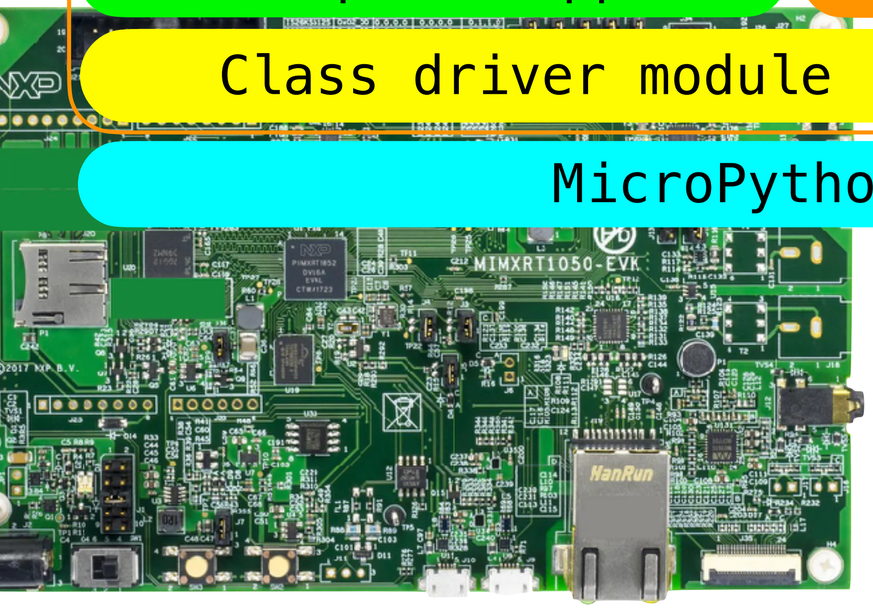
mikan

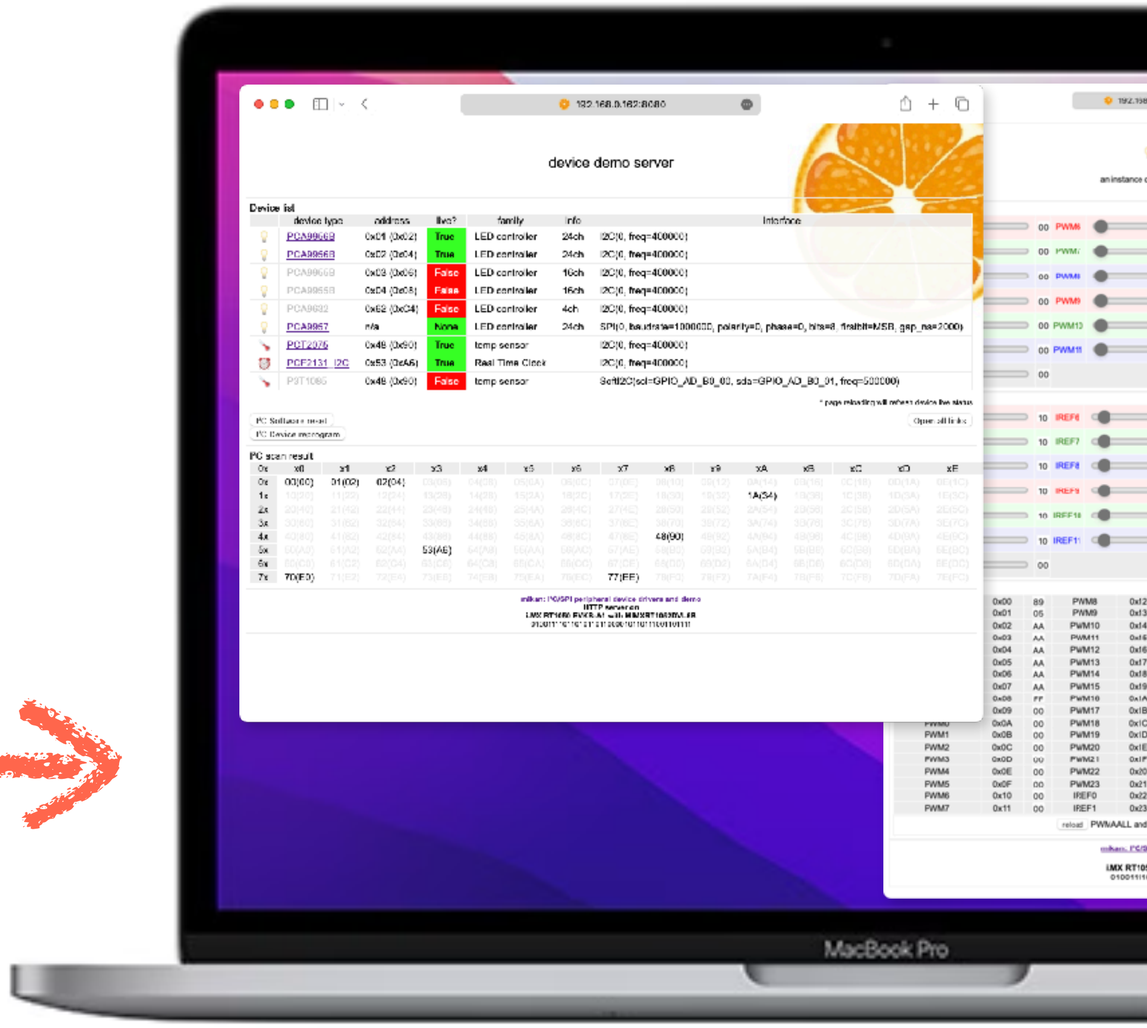
Examples (app)

Demo app

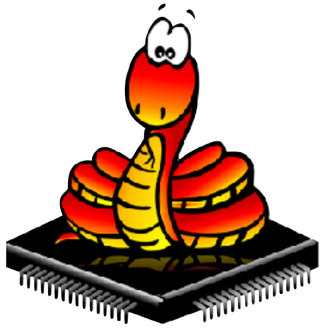
Class driver module

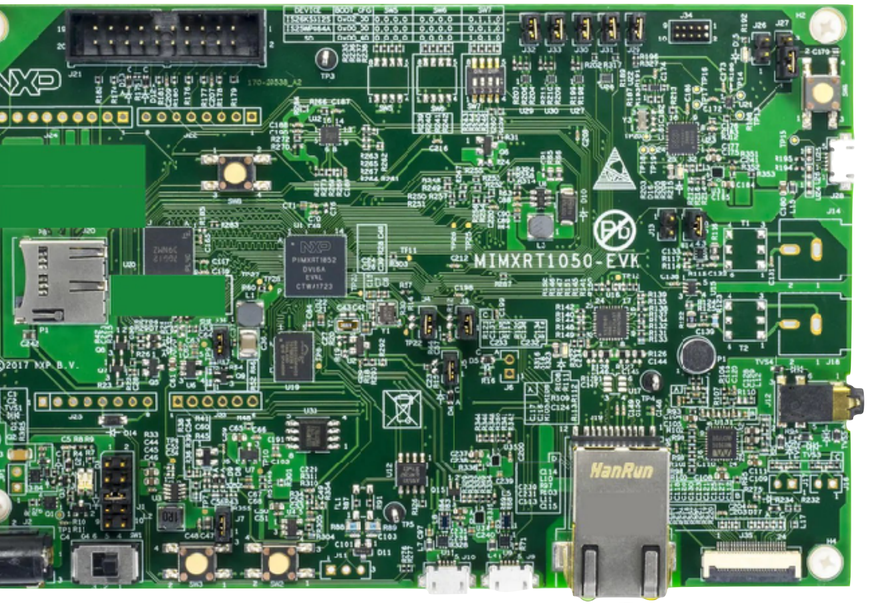
MicroPython



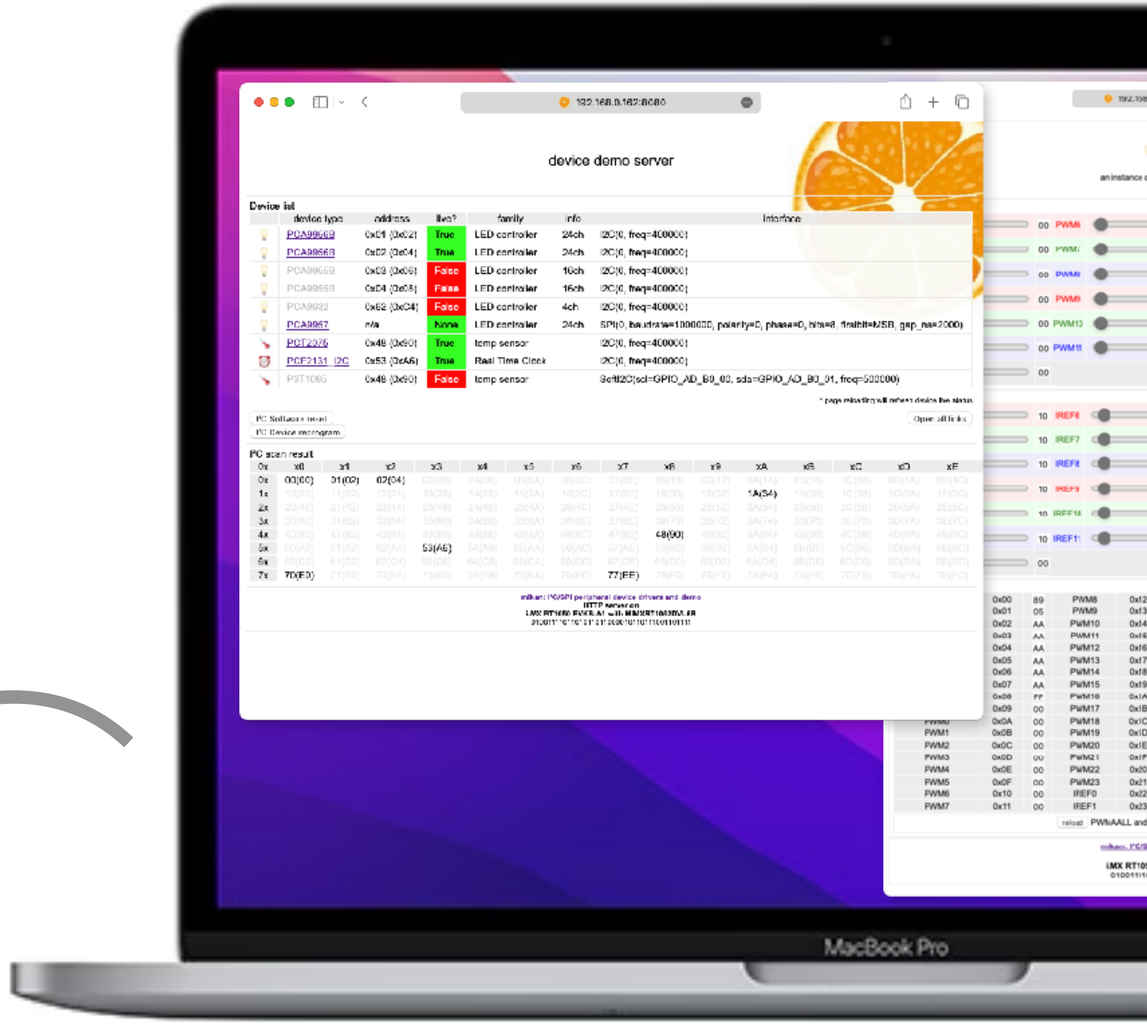


How does it work? : Hardware (using PC with a LAN cable)

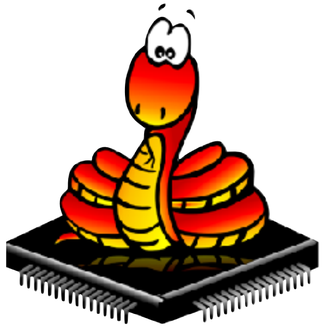


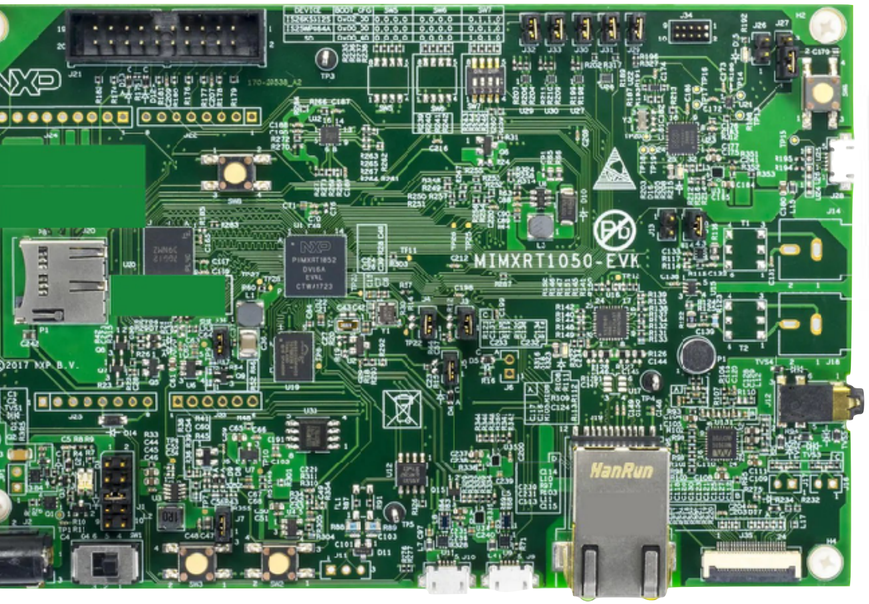


LAN cable



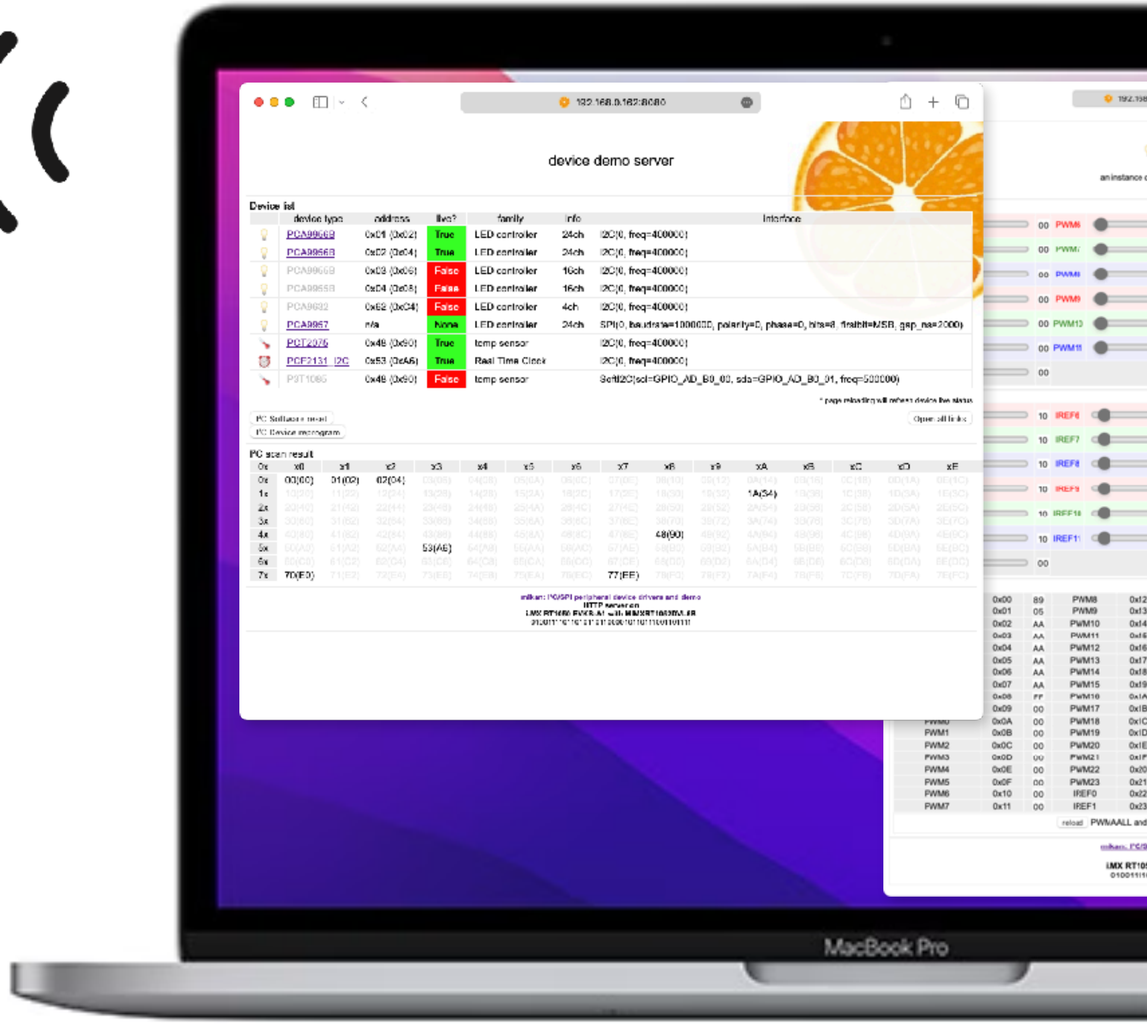
How does it work? : Hardware (using PC/Tablet/Smartphone with a wireless adapter)



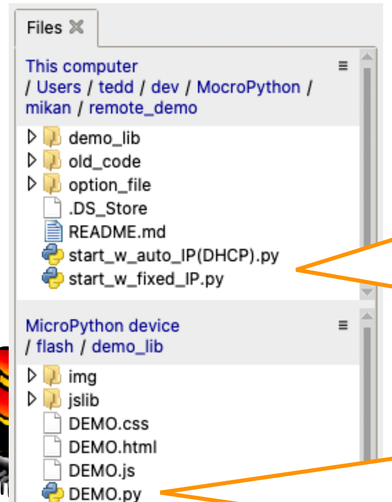


Wireless adapter
TL-WR902AC
(Client mode)

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`

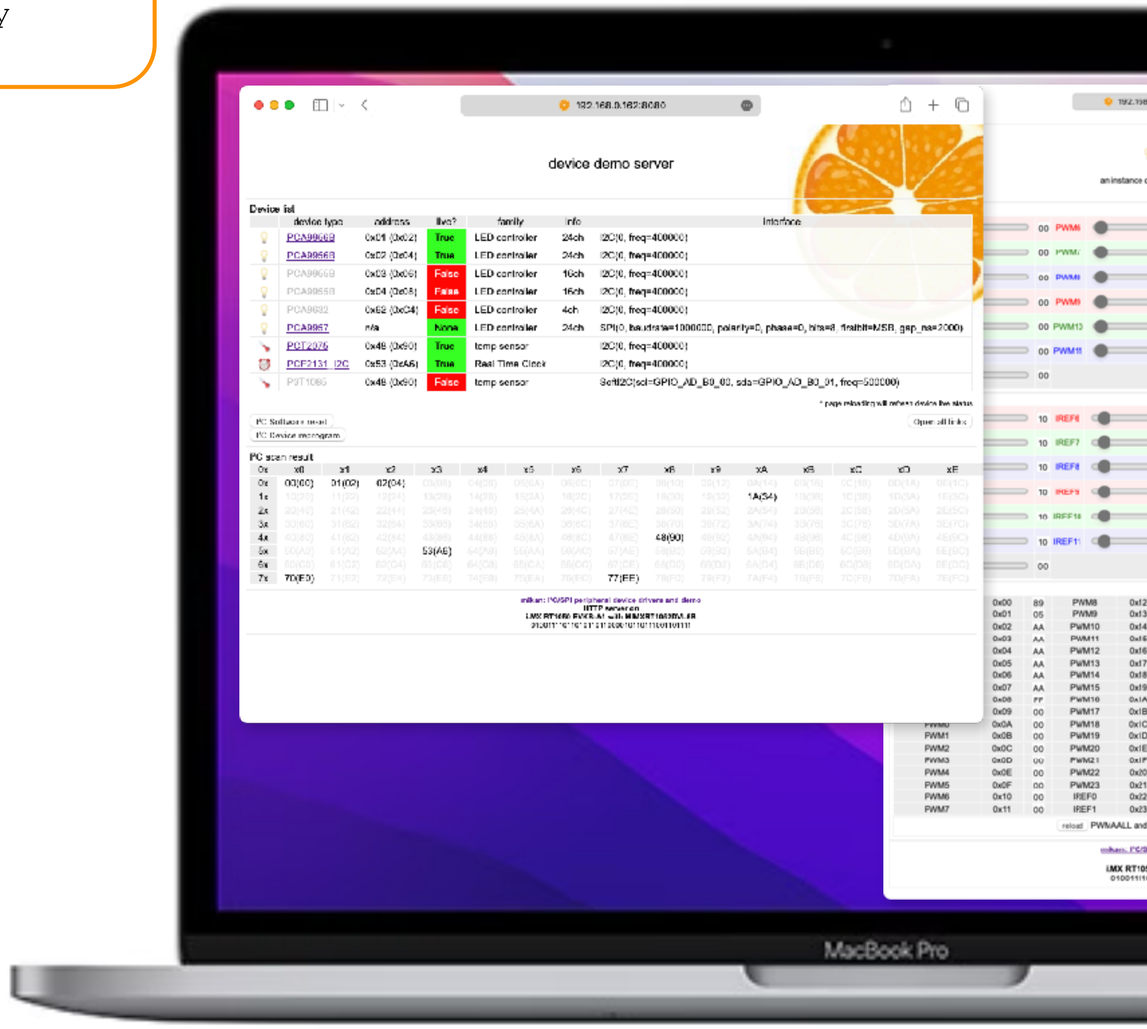
Main module of this demo app

`remote_demo/demo_lib/DEMO.py`

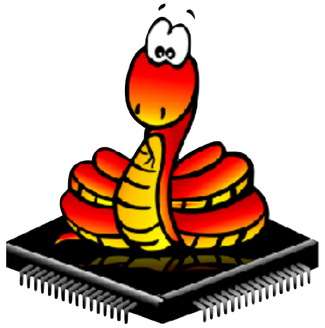
Local storage

`'remote_demo/demo_lib/'`

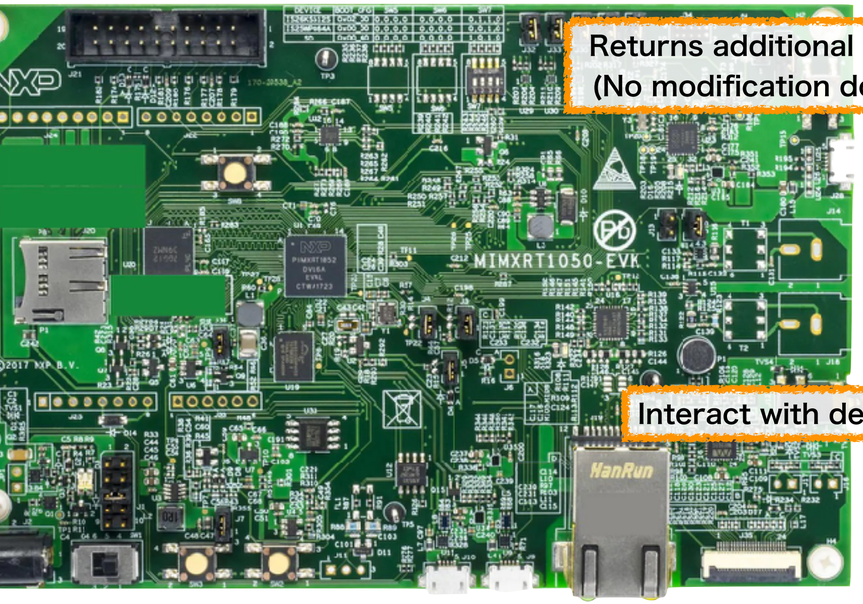
has `py`, `html`, `css` and `js` files



How does it work? : Software - communication



HTTP



Find device handling routine
Read and overwrite .html file

Returns additional files
(No modification done)

Interact with device

Browser requests HTML file

Serves HTML file

Request CSS, JS file

Serves CSS, JS file

Event notification with parameters

Return device response

User typed-in URL or clicked link

Page loading

Parse HTML and render

Found additional file requirement (CSS, JS, etc)

Start to listen user event after page loaded

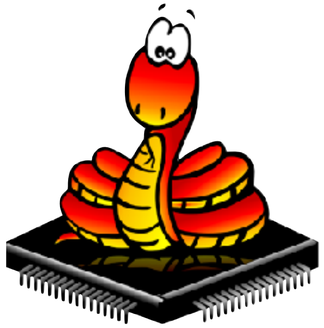
User event like button, slider, text input.

Ajax

Page contents update

MacBook Pro

How does it work? : Software - HTML overwrite



Find device handling routine
Read and overwrite .html file

Browser requests HTML file

User typed-in URL or clicked link

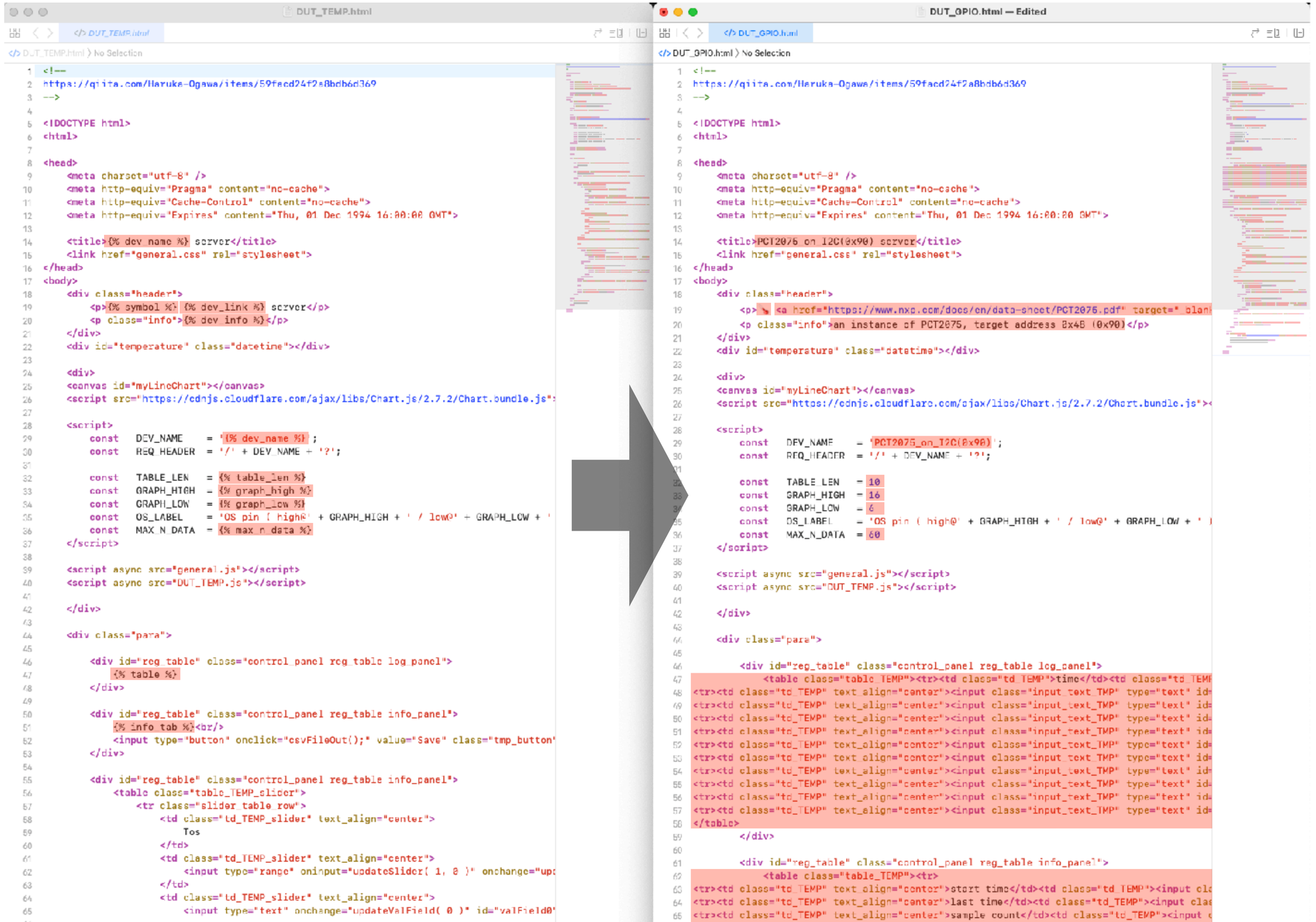
Serves HTML file

"{% xxx %}" part in "DUT_*.html" file are overwritten by "DUT_*.py" before serving

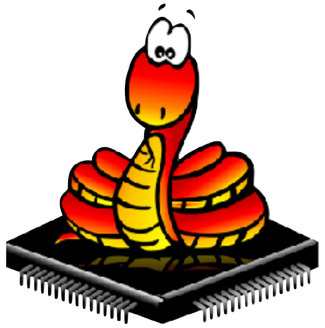

```
16 </head>
17 <body>
18     <div class="header">
19         <p>{% symbol %} {% dev_link %} server</p>
20         <p class="info">{% dev_info %}</p>
21     </div>
22     <div id="temperature" class="datetime"></div>
23
24     <div>
25     <canvas id="myLineChart"></canvas>
```

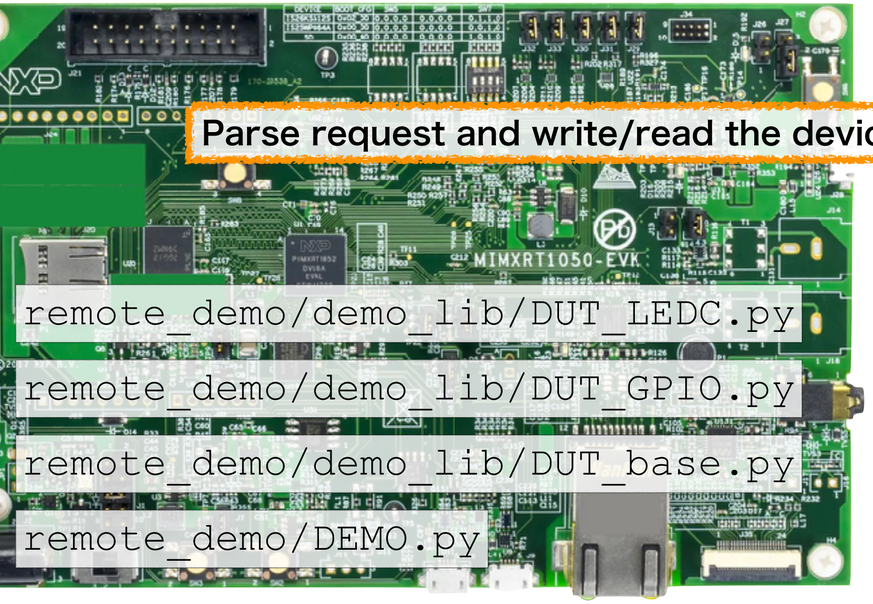
```
16 </head>
17 <body>
18     <div class="header">
19         <p> <a href="https://www.nxp.com/docs/en/data-sheet/PCT2075.pdf" target="_blank" rel="noopener
noreferrer">PCT2075</a> server</p>
20         <p class="info">an instance of PCT2075, target address 0x48 (0x90)</p>
21     </div>
22     <div id="temperature" class="datetime"></div>
23
24     <div>
25     <canvas id="myLineChart"></canvas>
```

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



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





Parse request and write/read the device

remote_demo/demo_lib/DUT_LEDC.py

remote_demo/demo_lib/DUT_GPIO.py

remote_demo/demo_lib/DUT_base.py

remote_demo/DEMO.py

Register update event

Update event on a widget

Register data

Update page contents

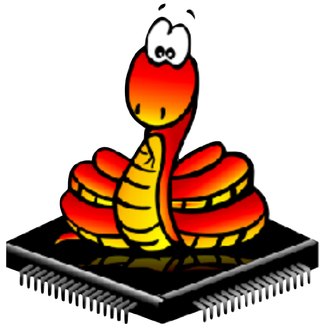
DUT_LEDC.js

LEDC_gradation_control.js

DUT_GPIO.js

general.js

How does it work? : Software - RTC operation



read date/time registers

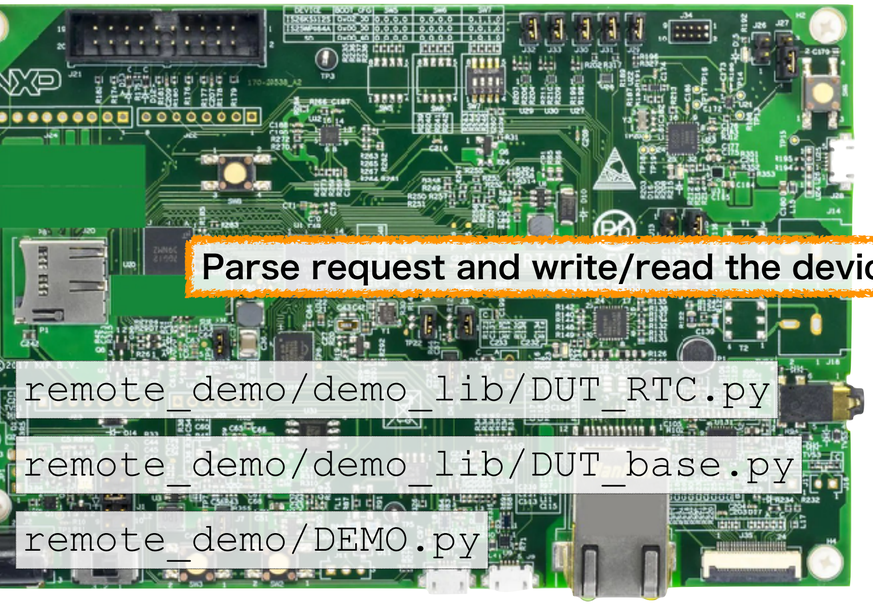
Register update event

Register data

Get current time

Update time display

every 1 second



Parse request and write/read the device

Register update event

Update event on widget

Register data

Update page contents

remote_demo/demo_lib/DUT_RTC.py

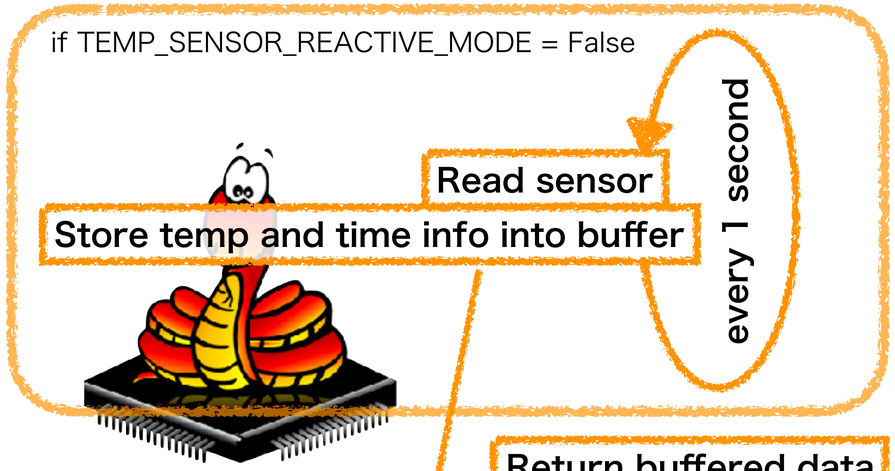
remote_demo/demo_lib/DUT_base.py

remote_demo/DEMO.py

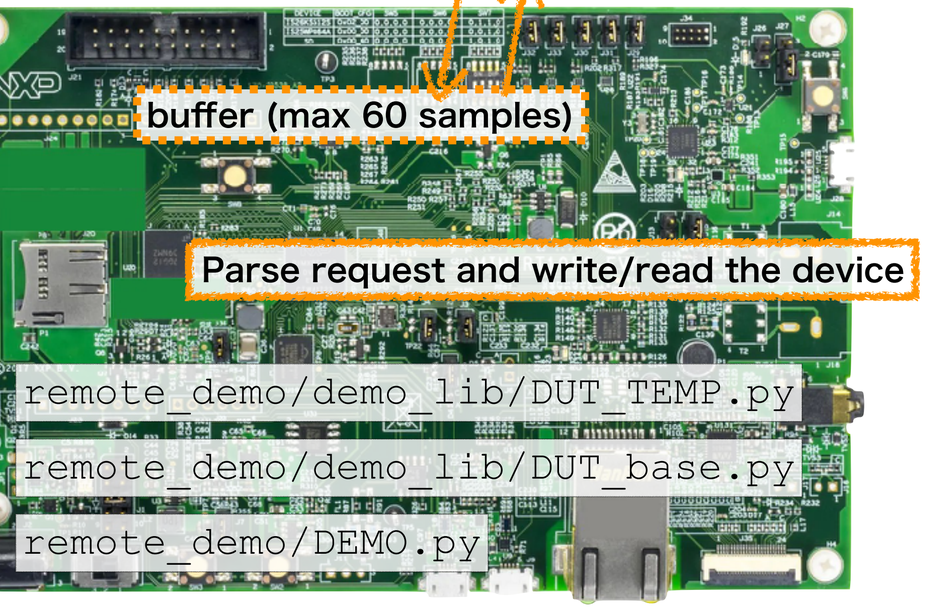
DUT_RTC.js

general.js

How does it work? : Software - temp sensor operation



Return buffered data



Register update event

Get current temp

Register data

Update page

Register update event

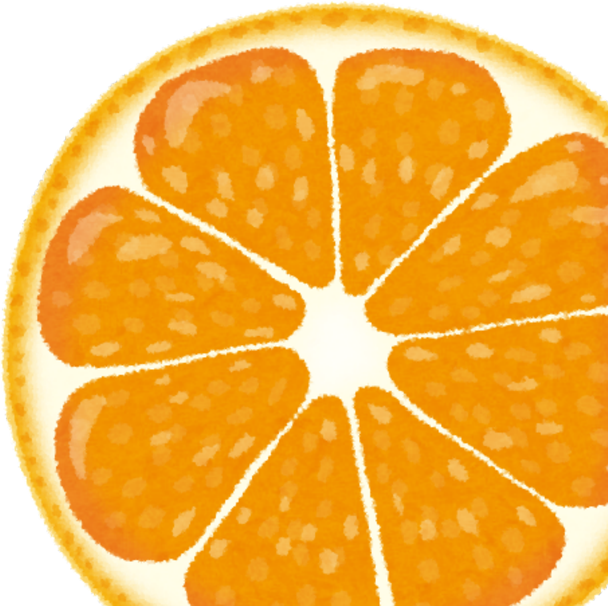
Update event on widget

Register data

Update page contents

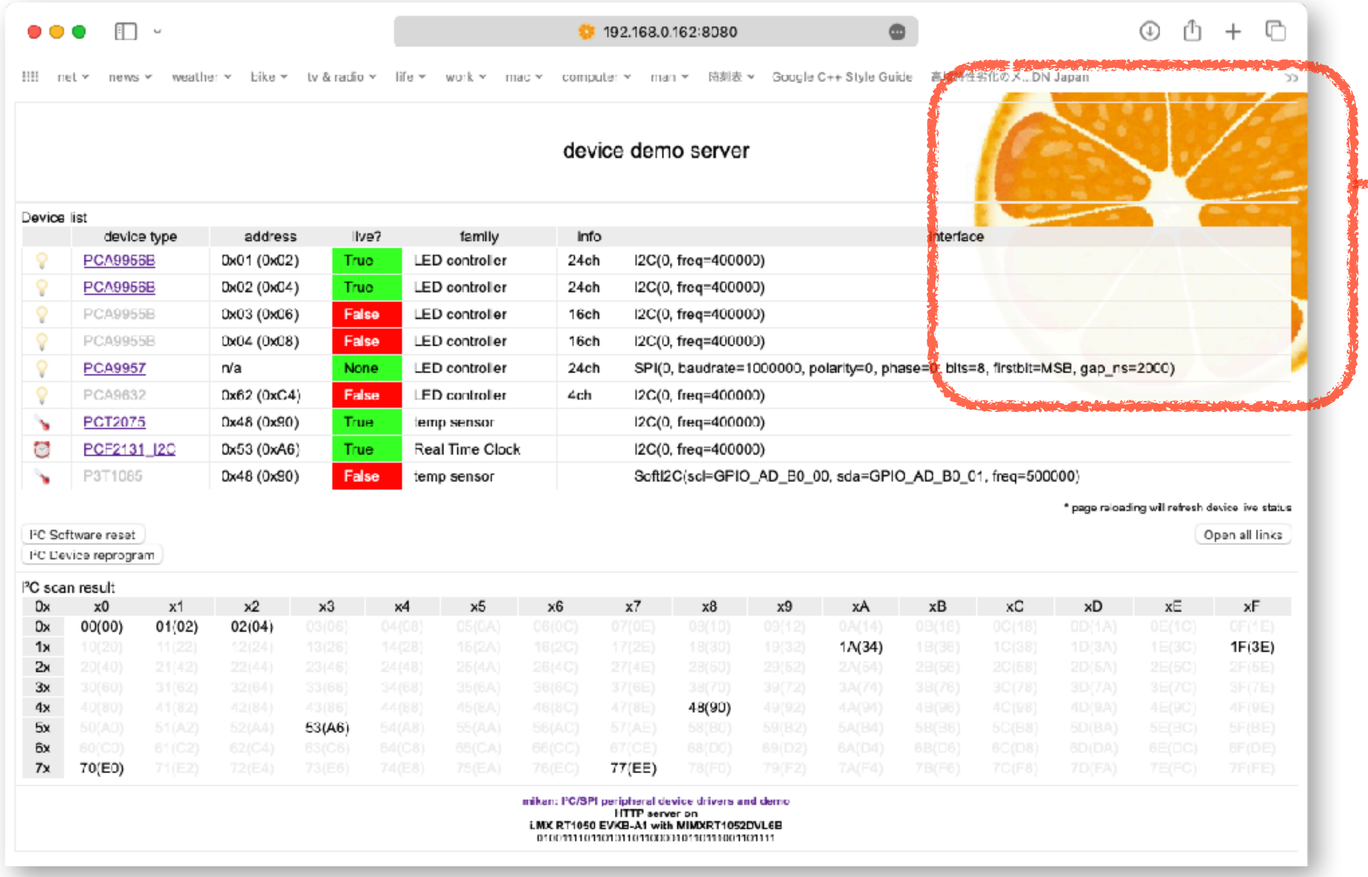
Customizing **remote_demo**

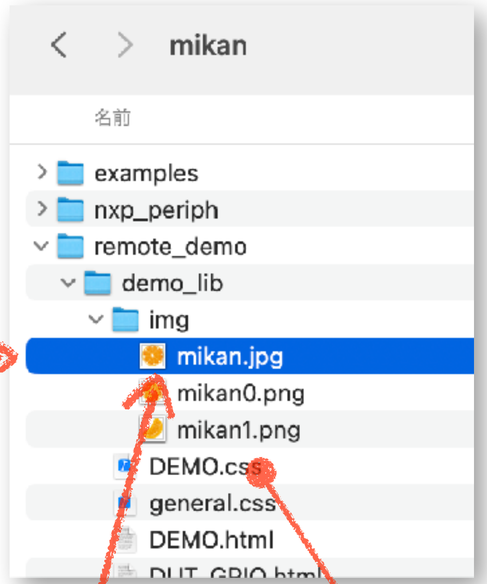
mikan



Changing logo image setting

Changing logo on right-top on device list page





remote_demo/demo_lib/DEMO.css

```
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

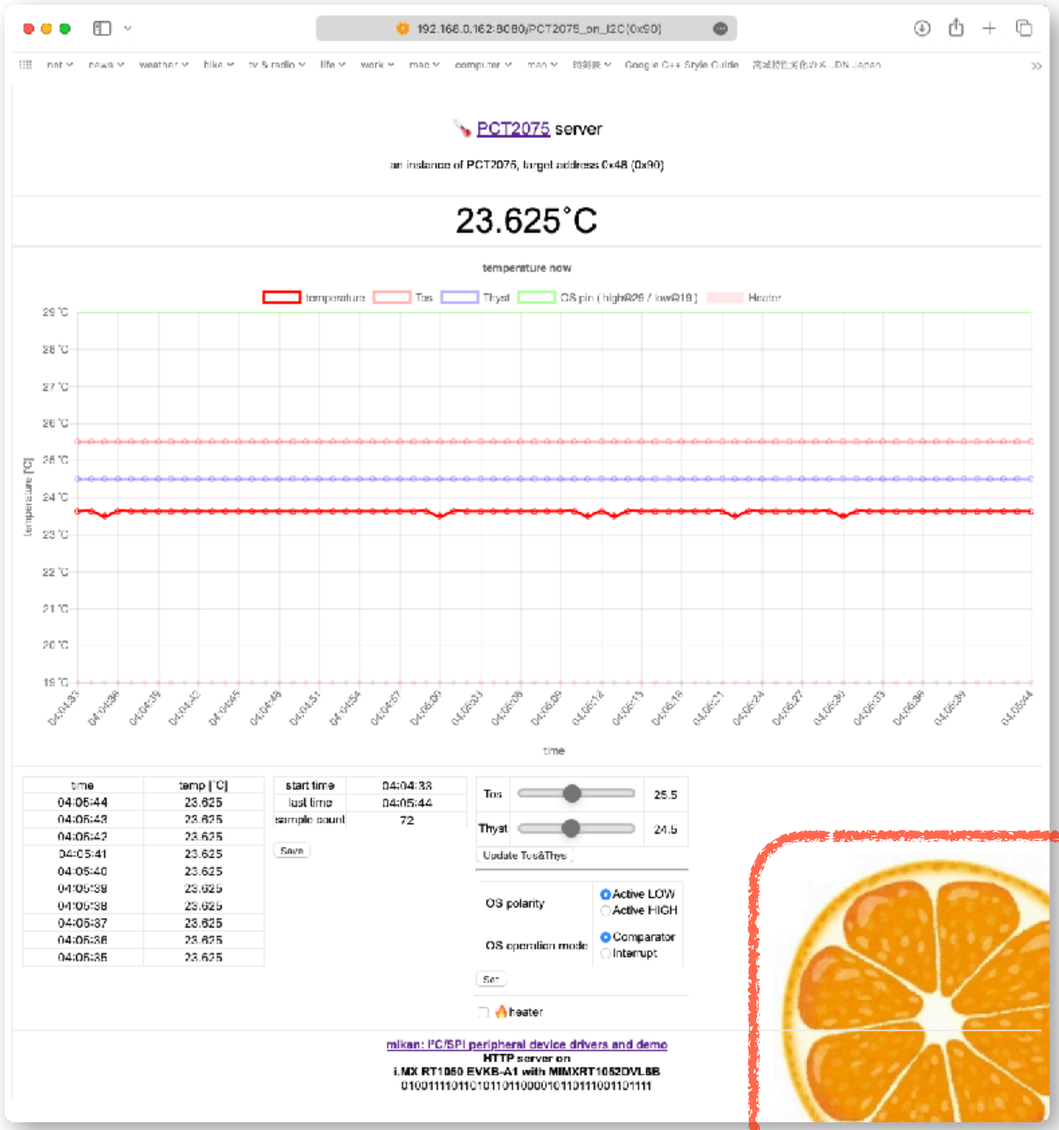
(1) Replace '`mikan.jpg`' file

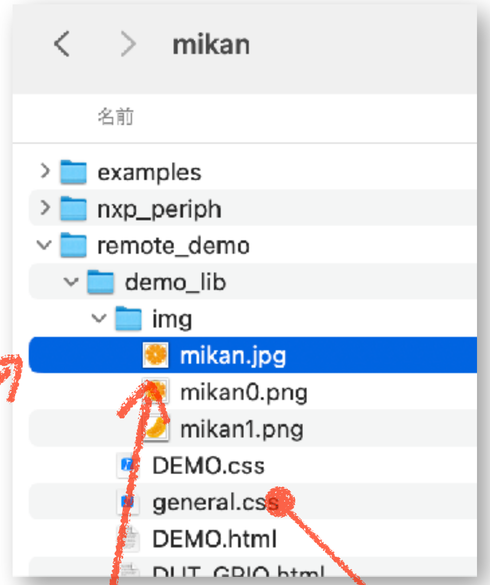
(2) Or put a new image file and modify 'background-image' URL.

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



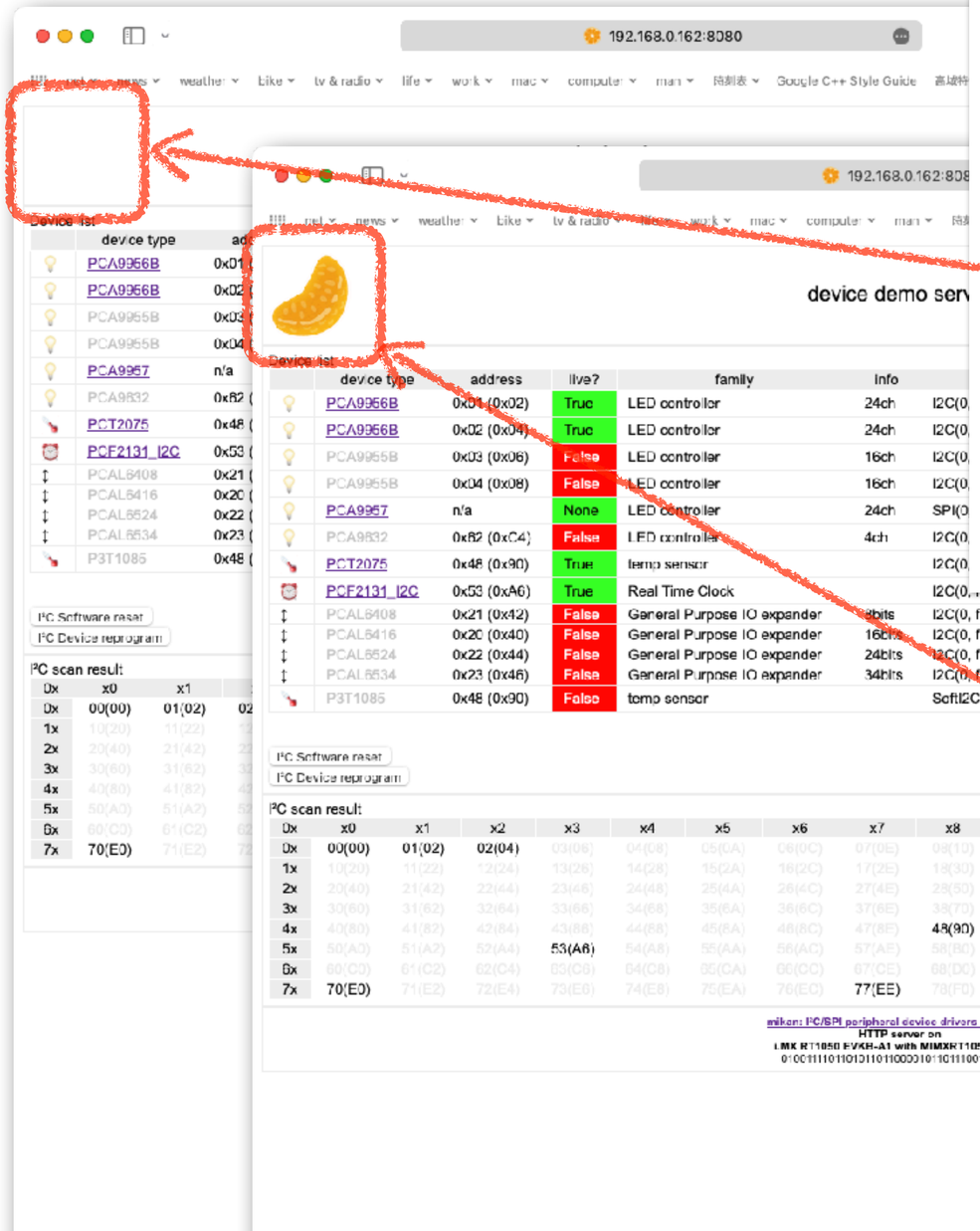


remote_demo/demo_lib/general.css

```
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



remote_demo/demo_lib/DEMO.html(default)

```
<!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>
```

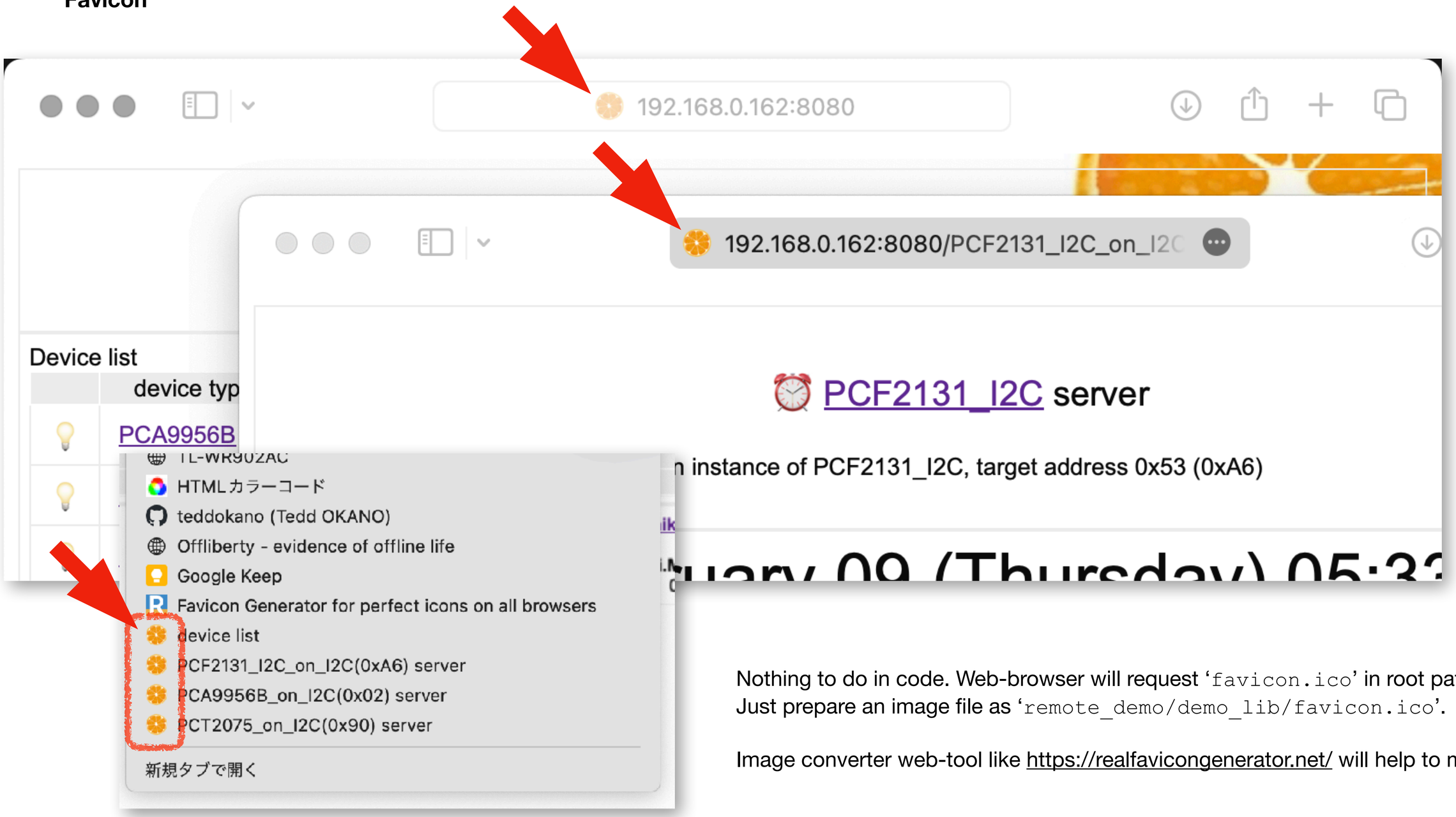
remote_demo/demo_lib/DEMO.html (modified)

```
<!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>
```

remote_demo/DEMO.py

Changing logo image setting

Favicon



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

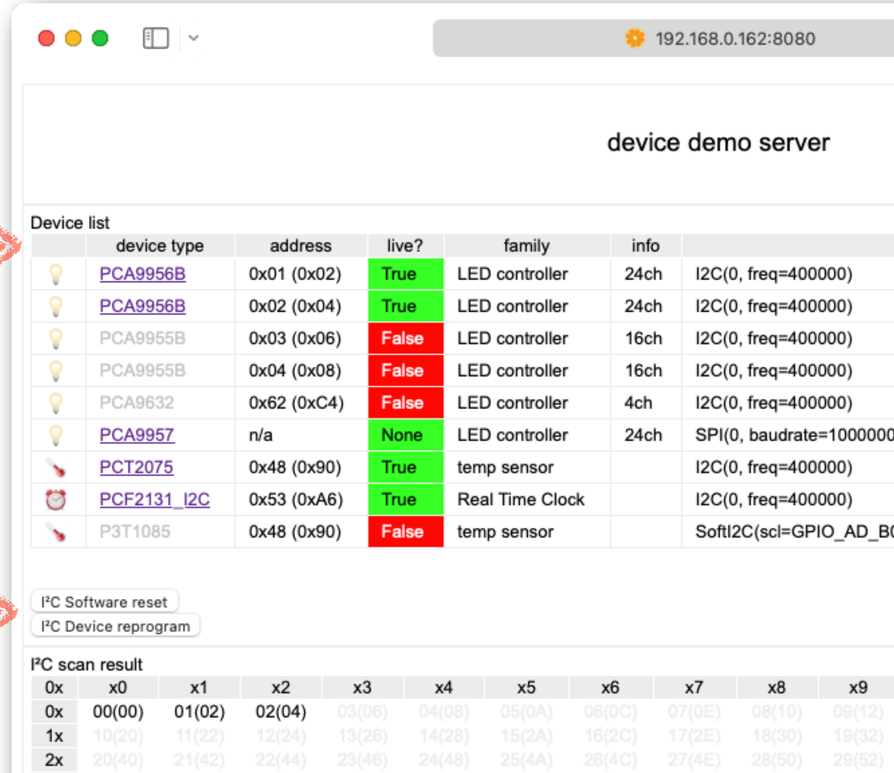
remote_demo/DEMO.py

```
def main( micropython_optimize = False ):
   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.

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

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

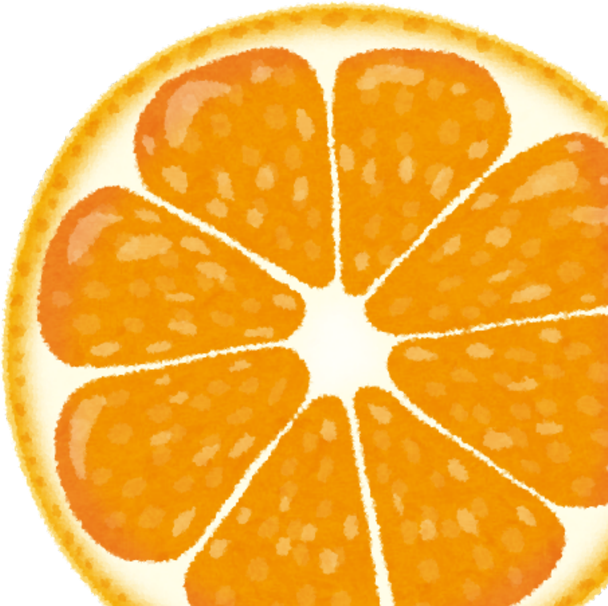
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>{% dev_name %} server</title>
    <link href="general.css" rel="stylesheet">
  </head>
```

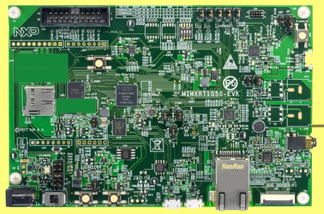
remote_demo/demo_lib/DUT_GPIO.html
remote_demo/demo_lib/DUT_LEDC.html
remote_demo/demo_lib/DUT_RTC.html
remote_demo/demo_lib/DUT_TEMP.html

remote_demo/demo_lib/general.css

NOTES

mikan





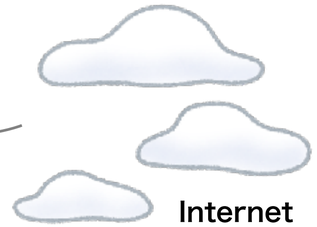
Ethernet

Wireless adapter

Wi-Fi

Access point

Ethernet

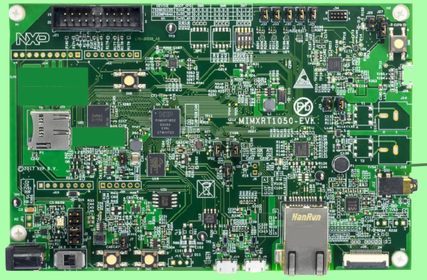


Internet

It was needing Internet connection

Arrange equipment, configure those

IMPROVED! from v1.9



Ethernet

From v1.9, no Internet required

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 〉 DUT_TEMP

```
1  import  machine
2  import  ure
3  import  ujson
4  import  micropython
5
6  from    nxp_periph  import  PCT2075, LM75B, P3T1085
7  from    nxp_periph  import  temp_sensor_base
8  from    demo_lib    import  DUT_base
9
10 TEMP_SENSOR_REACTIVE_MODE    = True  # Default: To operate multiple I2C devices on same bus
11 #TEMP_SENSOR_REACTIVE_MODE   = False # If only one device is connected on same I2C bus
12
13 class DUT_TEMP( DUT_base.DUT_base ):
14     APPLIED_TO      = temp_sensor_base
15     TABLE_LENGTH    = 10
16     SAMPLE_LENGTH   = 60
17     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

```python
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 geven, it performs write.
        If only 1 argument is geven, 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

        """
```

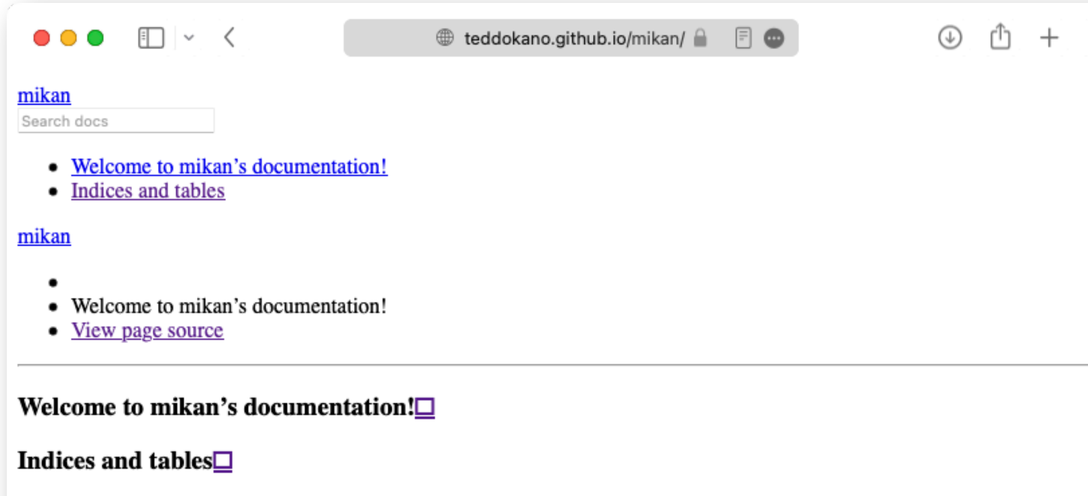
nxp_periph/temp_sensor.py

'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/

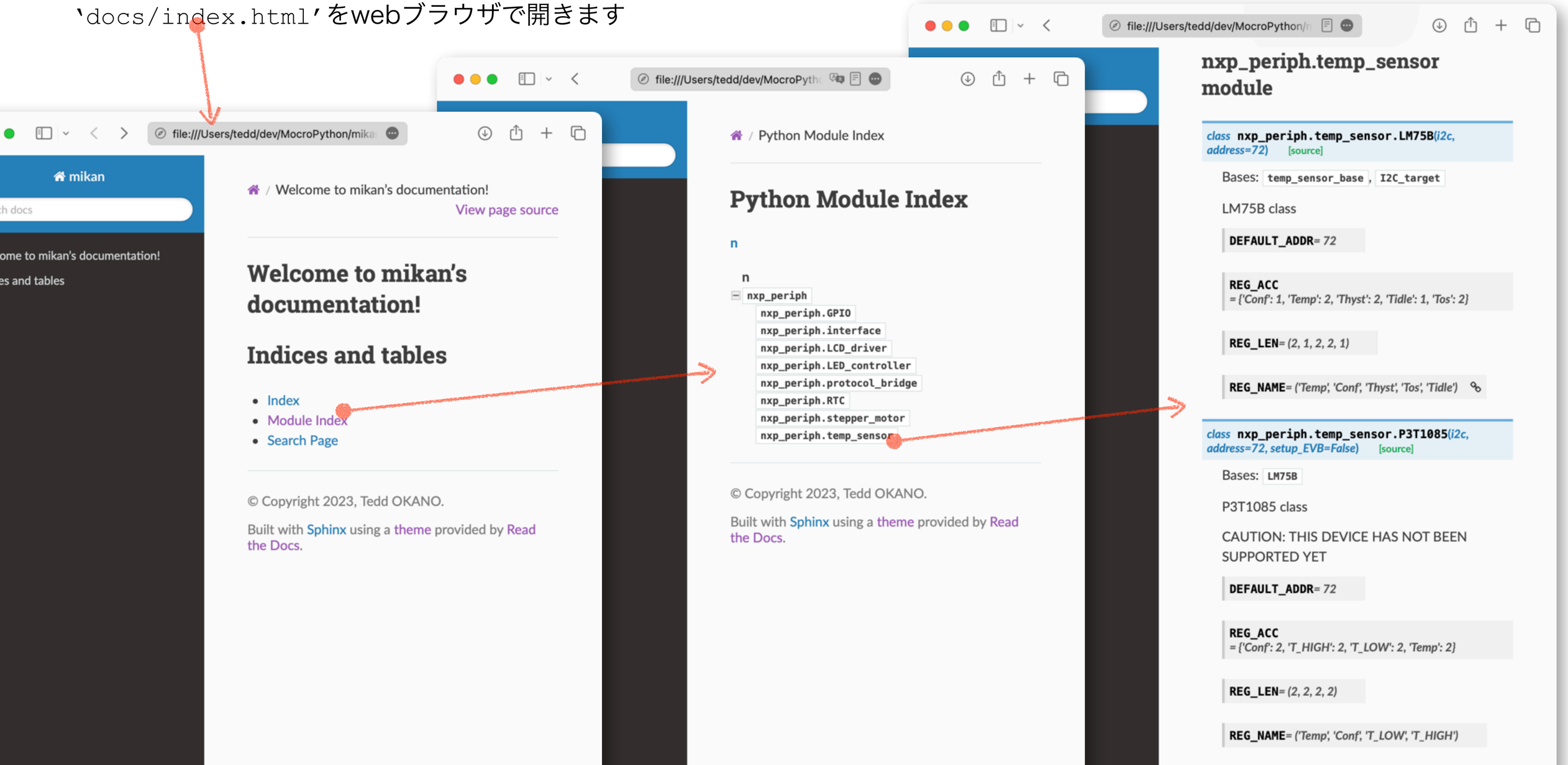
また，これらのドキュメントはクラスの (継承) 構造が前提になっているため少し難解です.
どのように書くのがいいのか悩み中



注意点 (+ 直近のデバッグ状況)

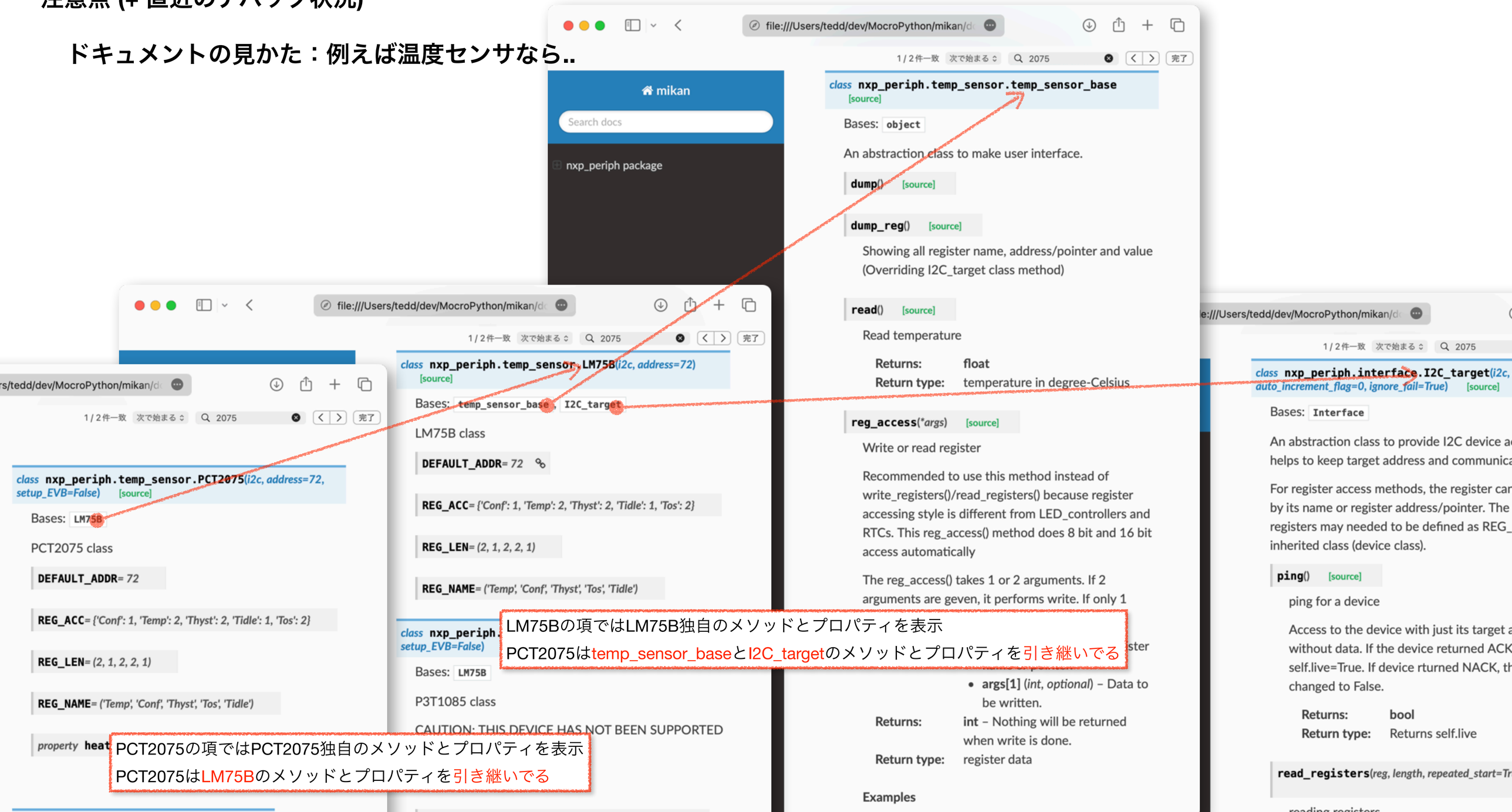
ドキュメントの見方：例えば温度センサなら..

'docs/index.html'をwebブラウザで開きます



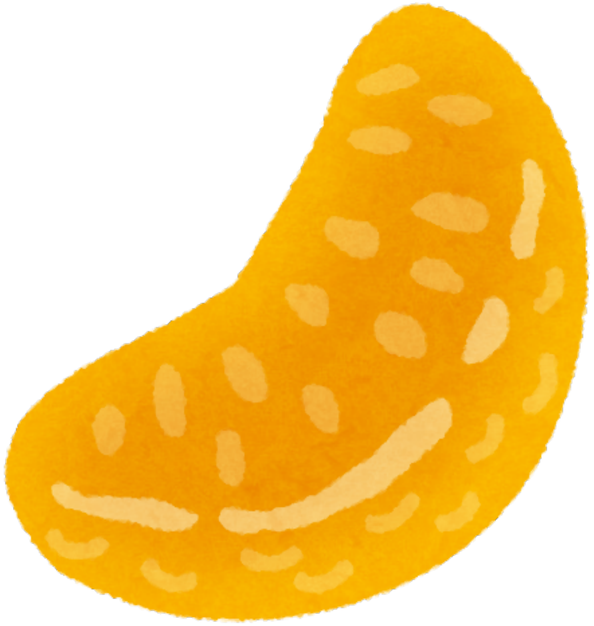
注意点 (+ 直近のデバッグ状況)

ドキュメントの見かた：例えば温度センサなら..



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/