RT1010Py

User Manual

Rev.1.5 Oct. 2025

olimex.com

Table of Contents

Revision

Revision	Date	By	Description
1.0	Oct. 2023	TsvetanUsunov	Initial document
1.1	Nov. 2023	TsvetanUsunov	Add more software ressource
1.2	Dec. 2023	TsvetanUsunov	UART 1,2,3 information
1.3	April 2025	TsvetanUsunov	usbtinyuf2 update
1.4	Oct. 2025	Meurisse D.	Adding content and details. Fix bytes() vs encoding.
1.5	Oct 2025	Meurisse D.	Fix typo. Update UEXT & fUEXT graphics. Install RT1010Py libraries. Append WS2812/NeoPixel support.

What is RT1010Py

RT1010Py is development board with MIMXRT1011DAE5A Cortex-M7 processor running at 500Mhz i.e. 4 times faster than RP2040.

RT1010Py runs MicroPython tanks to Robert Hammelrath (robert@hammelrath.com)

RT1010Py has these features:

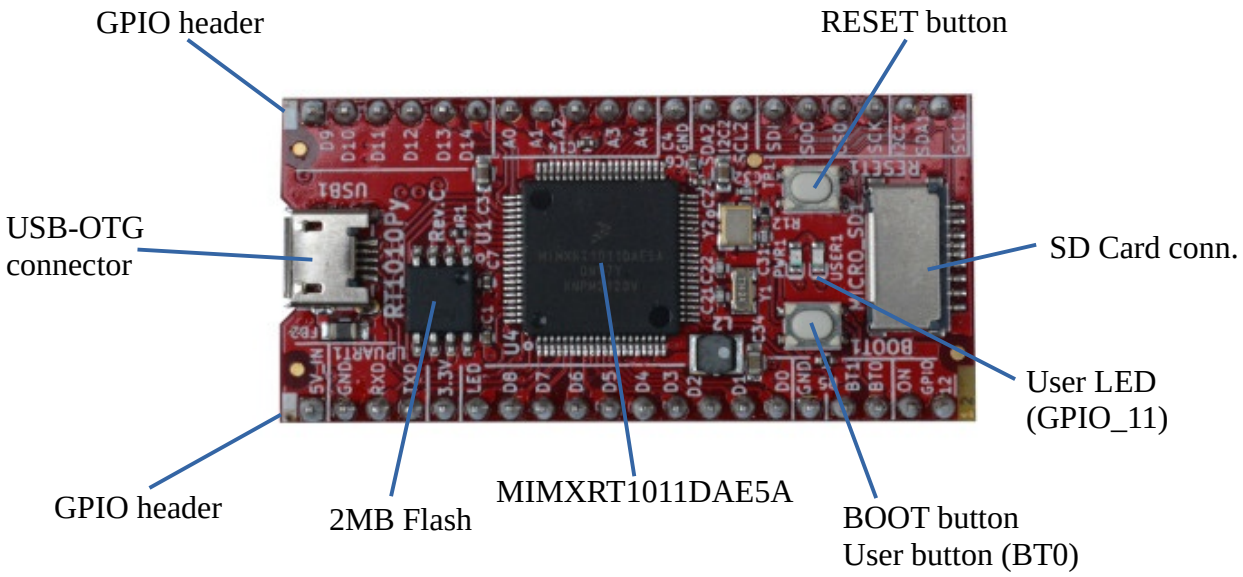
- MIMXRT1011DAE5A
- 128KB on board RAM
- 2MB SPI Flash
- two SPI
- two I2C
- four PWM bloc
- USB 2.0 OTG
- micro SD card connector
- RTC with 32.768 kHz crystal
- RESET button
- BOOT button (also act as User button)
- mUEXT connector with 3.3V, GND, I2C, SPI, and UART
- two GPIOs headers spaced at 22.86 mm (0.9”)
- Dimensions: 53.34 x 25.4 mm (2.1 x 1”)

Order codes for RT1010Py and accessories:

RT1010Py	RT1011 board running at 500Mhz with MicroPython
USB-CABLE-A-MICRO-1.8M	USB-A to micro cable
MICRO-SD-16GB-CLASS10	16GB microSD card
RT1010Py-DevKit	evaluation board for RT1010Py with two relays, two UEXT, USB-C
UEXT modules	There are temperature, humidity, pressure, magnetic field, light sensors. Modules with LCDs, LED matrix, Relays, Bluetooth, Zigbee, WiFi, GSM, GPS, RFID, RTC, EKG, sensors and etc.

Hardware

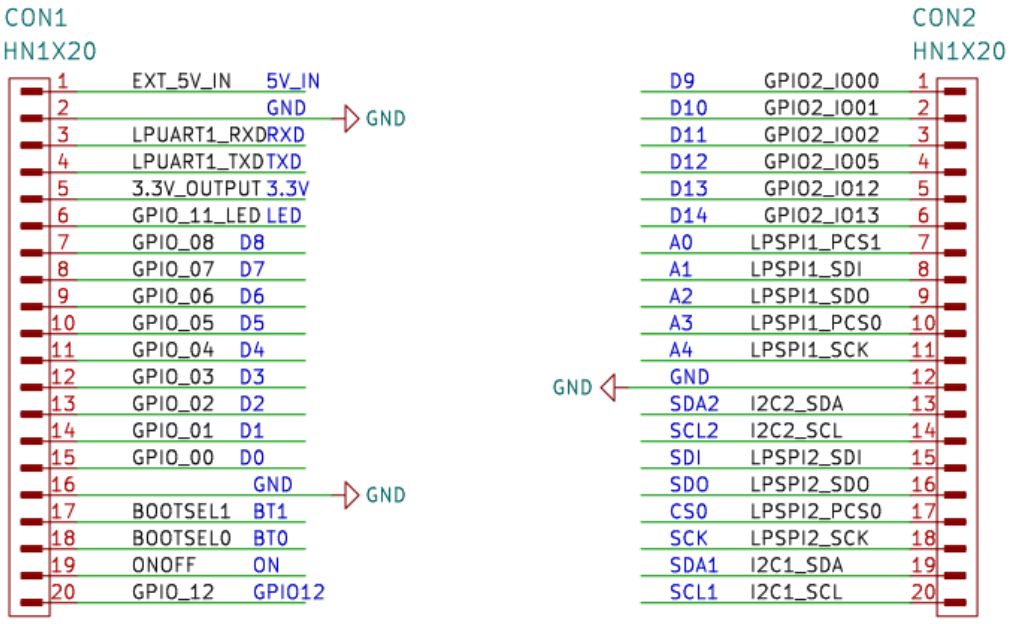
RT1010Py layout



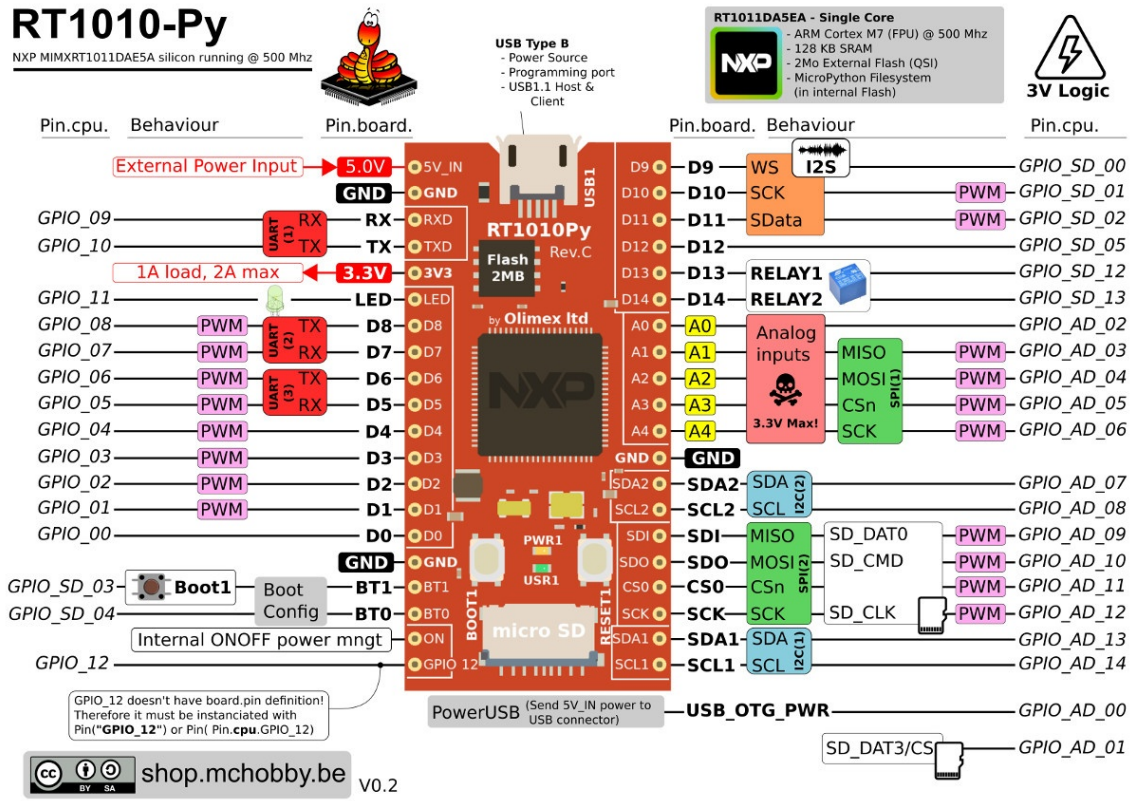
RT1010Py schematics

RT1010Py latest schematic is on GitHub

GPIO connectors



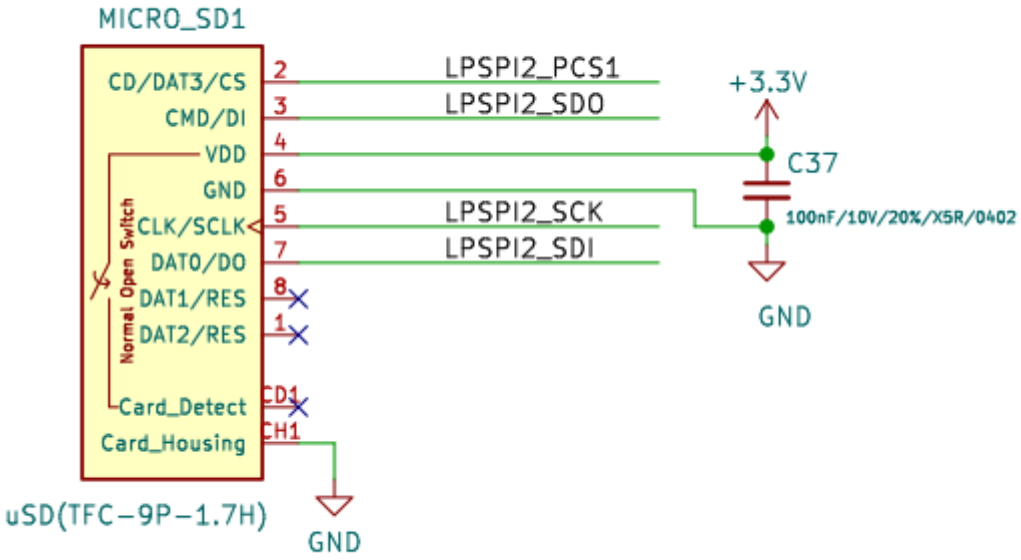
The following pinout is also available as DOCUMENTS/RT1010-py-pinout.jpg from the RT1010py repository.



SD card connector

The SD Card is connected to SPI2 bus. The SPI2 bus is also available on GPIOs connector and the fUEXT connector under the board (see pins SDI, SDO, CSK).

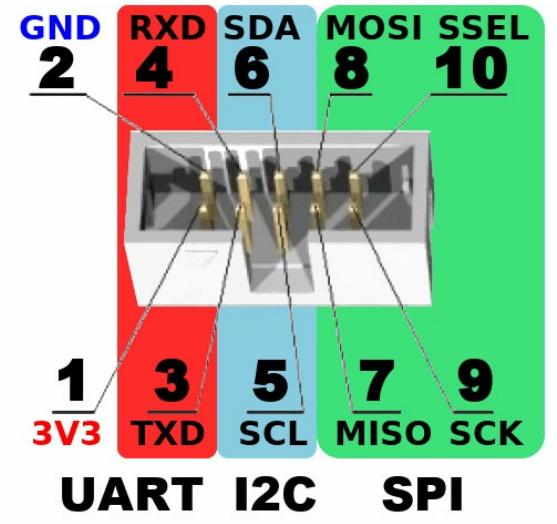
The SD Card & fUEXT bus are selected with the cpu pin LPSPI2_PCS1 (also known as `Pin.cpu.GPIO_AD_01`) whereas SPI2 bus on GPIOs is selected by LPSPI2_PCS0 (also known as `Pin.board.CS0`).



UEXT connector

UEXT connector stands for Universal EXTension connector and contain +3.3V, GND, I2C, SPI, UART signals. The original UEXT connector is 0.1” 2.54mm step boxed plastic connector.

All signals are with 3.3V levels.



As the boards become smaller and smaller some smaller packages were introduced too beside the original UEXT connector

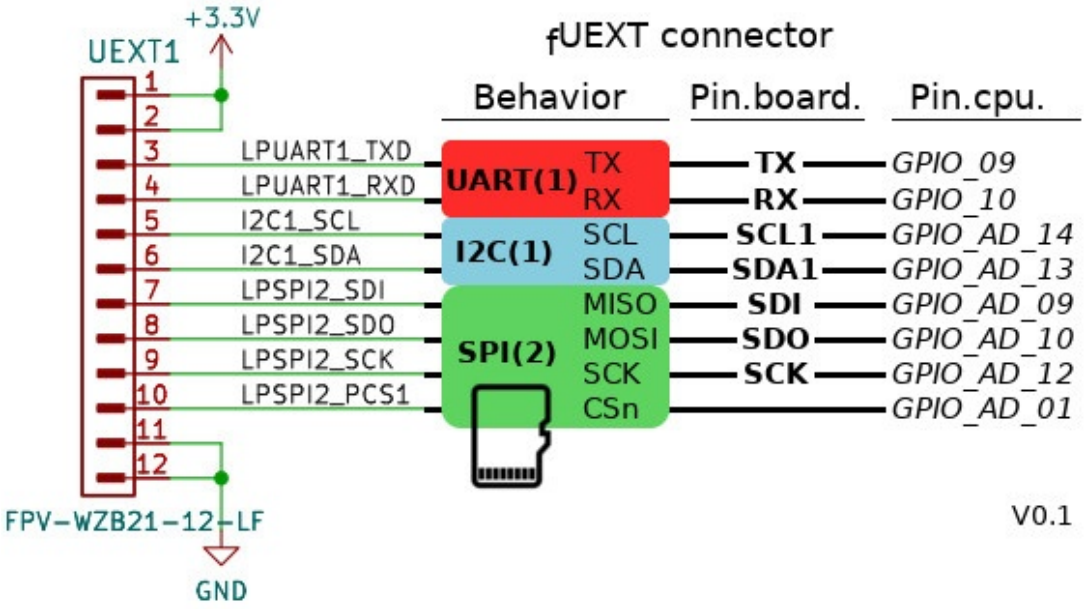
- mUEXT is 1.27 mm step boxed header connector which is with same layout as UEXT

- pUEXT is 1.0 mm single row connector (this is the connector used in RP2040-PICO30)

- fUEXT is Flat cable 12 pins, 0.5 mm step connector

Olimex Ltd. has developed number of MODULES with this connector. There are temperature, humidity, pressure, magnetic field, light sensors. Modules with LCDs, LED matrix, Relays, Bluetooth, Zigbee, WiFi, GSM, GPS, RFID, RTC, EKG, sensors and etc.

RT1010Py fUEXT (flat ribbon) connector:



Notes:

The fUEXT expose the SPI(2) bus used on the GPIO and the SD Card connector.

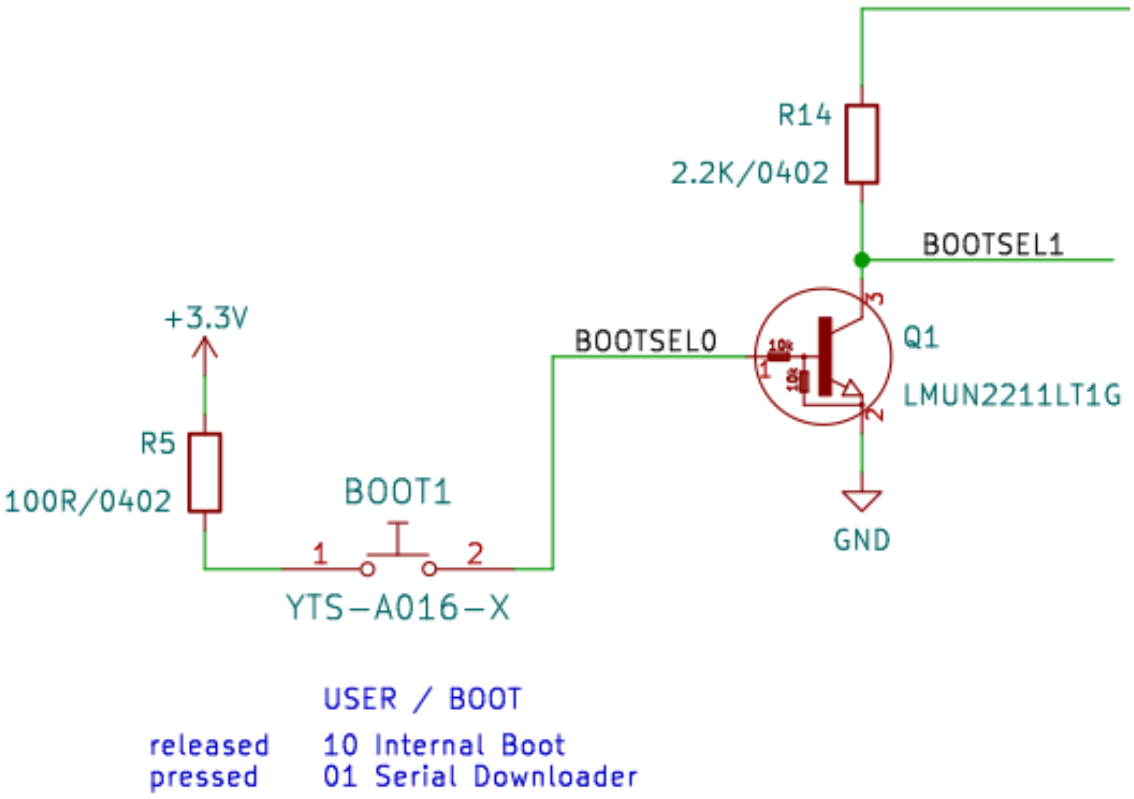
The fUEXT CSn signal is the same as the SD Card!

Consequence: the SPI(2) bus + CSn cannot be used on the fUEXT when the SDCard is also used!

Workaround: under MicroPython, the *Chip Select* line is managed by the user code. So the SPI(2) CSn can be ignored and replace with one of the free/unused TX, RX, SCL, SDA lines!

RT1010Py boot configuration

The BOOT1 button can also be used as user input button on Pin BT1.



Software

MicroPython bootloader and firmware installation:

OBSOLETE skip to tinyuf2 bootloader!

Detailed instructions how to install MicroPython bootloader and firmware are here: https://micropython.org/download/OLIMEX_RT1010/

- Get the files ufconv.py and uf2families.json from the micropython/tools directory, e.g. at https://github.com/micropython/micropython/tree/master/tools.
- Get the NXP program sdphost for your operating system, e.g. from https://github.com/adafruit/tinyuf2/tree/master/ports/mimxrt10xx/sdphost. You can also get them from the NXP web sites.
- Get the UF2 boot-loader package https://github.com/adafruit/tinyuf2/releases/download/0.9.0/tinyuf2-imxrt1010_evk-0.9.0.zip and extract the file tinyuf2-imxrt1010_evk-0.9.0.bin

Now you have all files at hand that you will need for updating.

1. Get the firmware you want to upload from the MicroPython download page.
2. Push and hold the "Boot" button, then press "Reset", and release both buttons.
3. Run the commands:

```
sudo ./sdphost -u 0x1fc9,0x0145 -- write-file 0x20206400 tinyuf2-imxrt1010_evk-0.9.0.bin
```

```
sudo ./sdphost -u 0x1fc9,0x0145 -- jump-address 0x20207000
```

Wait until a drive icon appears on the computer (or mount it explicitly), and then run:

```
python3 uf2conv.py <firmware_xx.yy.zz.hex> --base 0x60000400 -f 0x4fb2d5bd
```

You can put all of that in a script. Just add a short wait before the 3rd command to let the drive connect.

4. Once the upload is finished, push Reset again.

Using sudo is Linux specific. You may not need it at all, if the access rights are set properly, and you will not need it for Windows.

Once the generic boot-loader is available, this procedure is only required for the first firmware load or in case the flash is corrupted and the existing firmware is not functioning any more.

MicroPython Firmware

Oct. 5, 2025 : at the time of this manual update, the running version is v1.25.0 (preview)

Releases

v1.21.0 (2023-10-05) .hex / [.bin] / [Release notes] (latest)
v1.20.0 (2023-04-26) .hex / [.bin] / [Release notes]
v1.19.1 (2022-06-18) .hex / [.bin] / [Release notes]

Preview builds

v1.22.0-preview.31.g3883f2948 (2023-10-17) .hex / [.bin]
v1.22.0-preview.30.ge78471416 (2023-10-17) .hex / [.bin]
v1.22.0-preview.27.gc2361328e (2023-10-17) .hex / [.bin]
v1.22.0-preview.24.g51da8cc28 (2023-10-17) .hex / [.bin]
(These are automatic builds of the development branch for the next release)

RT1010Py USB tinyuf2 drag and drop bootloader installations

Robert Hammelrath did amazing work again! As of March 2025 RT1010Py have uf2 bootloader with virtual drive and people can program with simple drag and drop like on RaspberryPi PICO.

The new instructions are here.

1. You need the bootloader ZIP from Github extract it

2. Press and hold RESET then press BOOT button and release RESET button yellow LED will stay solid ON

Then execute this command:

```
$ sudo ./sdphost -u 0x1fc9,0x0145 -- write-file 0x20206400 tinyuf2-imxrt1010_evk-0.21.0.bin
Preparing to send 27888 (0x6cf0) bytes to the target.
(1/1)0%Status (HAB mode) = 1450735702 (0x56787856) HAB disabled.
Reponse Status = 2290649224 (0x88888888) Write File complete.
```

Then run the bootloader with:

```
$ sudo ./sdphost -u 0x1fc9,0x0145 -- jump-address 0x20207000

Status (HAB mode) = 1450735702 (0x56787856) HAB disabled.
```

At this point you will see new drive on your computer named RT1010BOOT

Download latest Micropython build in uf2 format from here At the time of writing this document it's

```
OLIMEX_RT1010-20250327-v1.25.0-preview.428.g50da085d9.uf2
```

and copy it to RT1010BOOT disk. Now you have Micropython up and running.

If you want to run the bootloader double click on RESET button.

Interacting with MicroPython boards

Thonny IDE

By far, Thonny is the most user friendly Python IDE for MicroPython board and very well suited for beginners.

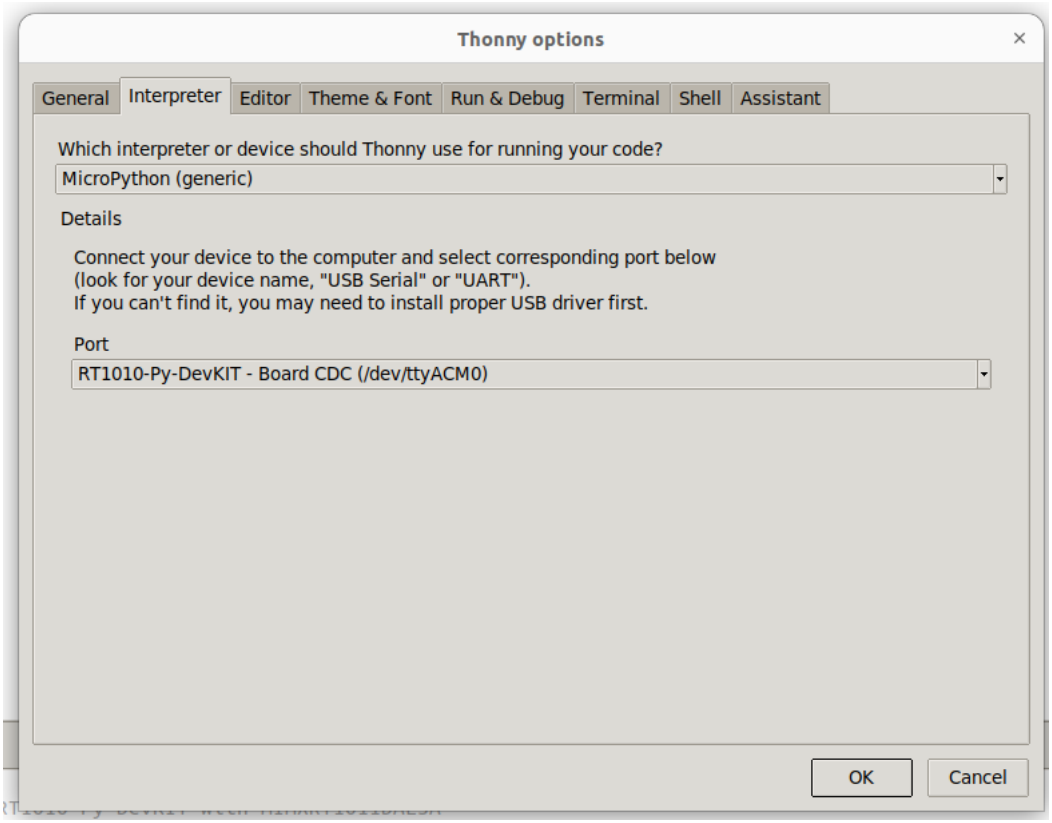
Thonny (thonny.org) is a multi-plateform code editor supporting Windows, Mac and Linux.

Thonny IDE also offers MicroPython programming through serial/usb-to-serial connection.

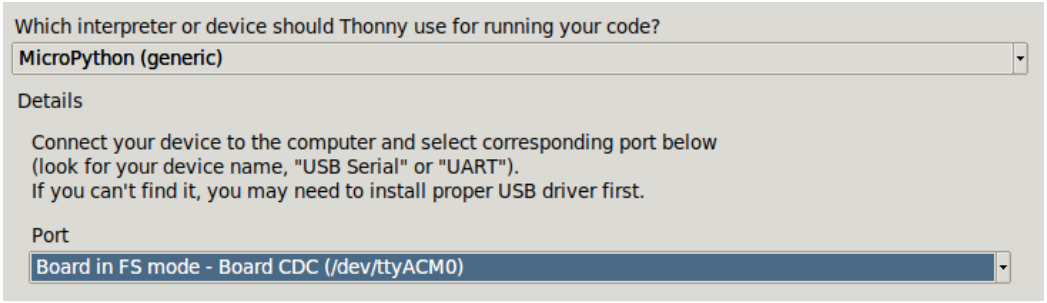
Install Thonny on Linux with:

```
$ sudo apt install thonny
```

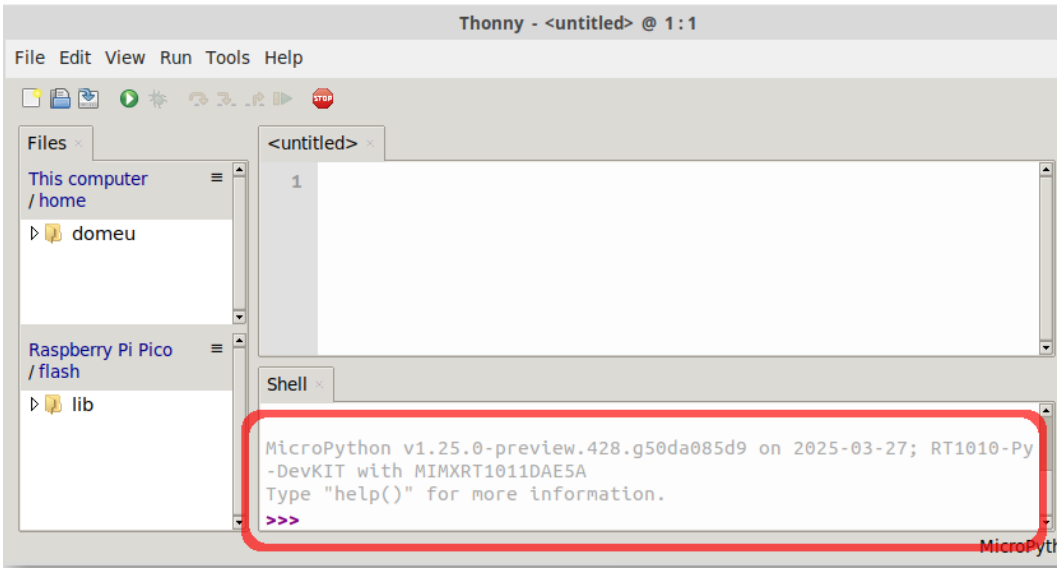
Plug USB cable to RT1010Py and run Thonny.
From the **Run** menu select the entry “**select interpreter...”**:



If the board type is not properly detected then select the usb-to-serial connection attached to the connected RT1010-py .



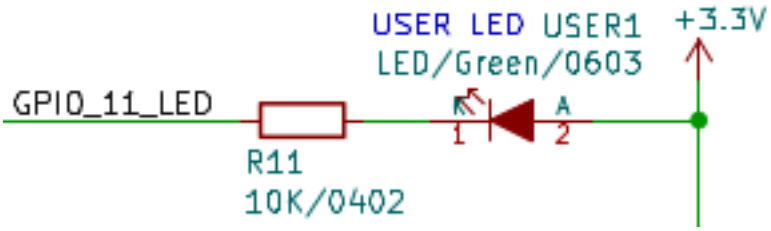
Once connected to the microcontroler, the **Shell** tab show the MicroPython REPL messages.



Now you are ready to make your first embedded hello world program i.e. to blink the USER LED. Select the following instructions in the **Shell** tab to make REPL executing them immediately.

```
from machine import Pin
led = Pin( "LED", Pin.OUT )
led.on()
led.off()
led.on()
led.off()
```

Due to the USER LED wiring, the LED lit when the output is LOW and vice-versa.



MPRemote official tool

MicroPython also released a tool named MPRemote, a Python tool for MicroPython development.

MPRemote uses the USB-to-Serial to manipulate the remote MicroPython plateform.

https://docs.micropython.org/en/latest/reference/mpremote.html

Despite the fact that MPRemote is a command line tool, it offers very powerful commands to transfer files to/from the MicroPython board, getting interactive REPL, executing computer script onto the remote MicroPython, made MicroPython executing its own filesystem script.

The bests features of MPRemote are:

- The ability to install an online MicroPython library with `mpremote mip install`. Eg: `mpremote mip install github:mchobby/esp8266-upy/modenv`
- The ability to mount a computer folder as if it were the MicroPython own filesystem. The `mpremote mount` command is a Great way to simplify MicroPython development.

MPRemote is a very handy tool to update a MicroPython filesystem.

RT1010Py libraries

Some RT1010Py extra features are supported with libraries.
The libraries must be installed on the RT1010Py board prior to use the related features.

Here is the list of extra libraries available for the RT1010py in this repository

- **ws2812**: controlling the WS2812/NeoPixel smart LED with SPI bus.

MicroPython MIP installer

MicroPython MPRemote computer tool is the easiest way to install online libraries. The **MPRemote MIP** command can be executed either from a computer, either from a network capable microcontroler. See MPRemote details on https://docs.micropython.org/en/latest/reference/mpremote.html .

On a WiFi capable plateform:

```
>>> import mip
>>> mip.install("github:OLIMEX/RT1010Py")
```

Or with the **mpremote** utility :

```
mpremote mip install github:OLIMEX/RT1010Py
```

Manual installation

Libraries can also be installed with your favorite tool (eg: Tonny IDE).

Copy all the files under `OLIMEX/RT1010Py/lib` (from repository) to the MicroPython RT1010Py board. The files must be stored into the `/lib` directory of the RT1010Py.

Create the `/lib` directory if not yet present on your MicroPython RT1010Py board.

MicroPython with RT1010Py

Processor frequency

```
import machine
machine.freq()
```

Delay and timing

```
import time

time.sleep(1)           # sleep for 1 second
time.sleep_ms(500)      # sleep for 500 milliseconds
time.sleep_us(10)       # sleep for 10 microseconds

start = time.ticks_ms() # get millisecond counter

delta = time.ticks_diff(time.ticks_ms(), start) # compute time difference
```

Timers

The id timer -1 is used to define a virtual timer handled by the board firmware.

Specifying the timer can be done by using the named “id” parameter.
The RT1010 provides 2 General Programmable Timers (GPT).

```
from machine import Timer

tim0 = Timer(-1)
tim0.init(period=5000, mode=Timer.ONE_SHOT, callback=lambda t:print(0))

tim1 = Timer(-1)
tim1.init(period=2000, mode=Timer.PERIODIC, callback=lambda t:print(1))
```

blinking LED with Timer

The pin name **LED** identifies the onboard green LED also identified as **GPIO_11**.

```
from machine import Pin, Timer
led = Pin("LED", Pin.OUT)
tim = Timer()

def tick(timer):
    global led
    led.toggle()

tim.init(freq=2.5, mode=Timer.PERIODIC, callback=tick)
```

GPIOs

valid GPIO names are: D0..D14, LED, A0..A4 , GPIO_00..GPIO_14, some of them duplicate for instance GPIO_00 is D0.

Output pins

A Pin defined as output have a Low Impedance allowing it to provide or sink some current to/from an external device.

```
from machine import Pin

led = Pin('LED',Pin.OUT)
led.on()
led.off()
led.toggle()

# Setting the state from a variable
state = True
led.value( state )

# Current state can also be read form the Pin
current_state = led.value()
```

Always being sure that output pin is not overloaded (shorted to ground or 3.3V) because this may degrade/destroy the microcontroler.

Using board constant

Note that pins can also be identified thanks to board constant. So `'LED'` string can be replace with **`Pin.board.LED`** constant.

```
from machine import Pin

led = Pin(Pin.board.LED,Pin.OUT)
led.toggle()
```

Available constants are the following. Most of them are visible on the board silkscreen.

A0	A1	A2	A3
A4	BT0	BT1	CS0
D0	D1	D10	D11
D12	D13	D14	D2
D3	D4	D5	D6
D7	D8	D9	LED
MCK	RELAY1	RELAY2	RX
SCK	SCK_RX	SCK_TX	SCL1
SCL2	SDA1	SDA2	SDI

SDO	SD_RX	SD_TX	TX
USB_OTG1_PWR	WS_RX	WS_TX	

Input pins

A Pin set as input have a High Impedance. So an external device can set the pin state to LOW (0 Volts) or HIGH (3.3V) without damaging the MCU.

```
from machine import Pin

d9 = Pin(Pin.board.D9, Pin.IN)
print( d9.value() ) # Returns 1 or 0 depending on Pin State
```

Input pin with PULL_UP / PULL_DOWN

A pin set as input can have internal Pull Up (or Pull Down) resistor activated.

When the Pull Up is activated, the input pin stays at 3.3V except if an external device (eg: push button) force the voltage to LOW.

```
from machine import Pin

d9 = Pin(Pin.board.D9, Pin.IN, Pin.PULL_UP)
print( d9.value() ) # Returns 1 or 0 depending on Pin State
```

Special GPIOs

Some GPIOs or pin have special behavior.

- **USB_OTG1_PWR**: output, default=LOW. When micro-USB is used as USB HOST (USB-OTG) this pin can be used to set 5V on the USB from the EXT_5V_IN (external 5V input)
- **GPIO_12**: do not confuse with D12. Can only be instantiated with a string id.
- **BT0**: Use as input pin read Boot1 button state (0=release, 1=pressed).
- **BT1**: Use as input pin read Boot1 button state (1=release, 0=pressed).
- **GPIO_11** or **LED**: control the green user LED. Set LOW to light-up the LED.
- **RELAY1** or **RELAY2**: Respectively D13 and D14, are used with RT1010-py-DevKit development board.
- **A0** to **A4**: those analog pins are also attached to SPI1 bus.
- **RX** and **TX**: respectively GPIO_09 and GPIO_10 of UART1
- **D5** and **D6**: Rx and Tx of UART3
- **D7** and **D8**: Rx and Tx of UART2
- **D9** to **D11**: are respectively attached to WS, SCK and SD of I2S(3) bus

PWM

By example, the following example will breathing LED connected to D1 GPIO:

```
import time
from machine import Pin, PWM

pwm = PWM(Pin('D1'))
pwm.freq(1000)

duty = 0
direction = 1
for x in range(8 * 256):
    duty += direction
    if duty > 255:
        duty = 255
        direction = -1
    elif duty < 0:
        duty = 0
        direction = 1
    pwm.duty_u16(duty * duty)
    time.sleep(0.001)
```

The PWM is generated from the MIMXRT FlexPWM hardware sub-modules.

Pin	Olimex RT1010PY	Pin	Olimex RT1010PY
D0	•	**D12**	F1/1/B
D1	F1/0/B	**D13**	F1/1/A
D2	F1/0/A	**D14**	•
D3	F1/1/B	**A0**	•
D4	F1/1/A	**A1**	F1/2/B
D5	F1/2/B	**A2**	F1/2/A
D6	F1/2/A	**A3**	F1/3/B
D7	F1/3/B	**A4**	F1/3/A
D8	F1/3/A	**SDI**	F1/3/X
D9	•	**SDO**	F1/2/X
D10	F1/0/B	**CS0**	F1/1/X
D11	F1/0/A	**SCK**	F1/0/X

- Fm/n/l: FLEXPWM module m, submodule n, channel l.
- The pulse at a X channel is always aligned to the period start.

According to MicroPython documentation, each FLEX submodule (used to generate PWM) or QTMR module may run at different frequencies. The PWM signal is created by dividing the pwm_clk signal by an integral factor accordingly to the formula:

```
   f = pwm_clk / (2**n * m)
```

with n from 0..7 and m in range 2..65535.

As MIMXRT1010 have a pwm_clk of 125MHz, the lowest frequency will be pwm_clk/2**23 = 15 Hz whereas the highest frequency with U16 resolution will be pwm_clk/2**16 = 1907 Hz.

Still accordingly to MicroPython documentation, the highest frequency with 1% resolution is pwm_clk/100 = 1.25 MHz. The highest achievable frequency for A/B channels is pwm_clk/3 = 41.6 MHz. The highest frequency for X channels and QTMR is pwm_clk/2 = 62.5 MHz

ADC

RT1010 have a 12bits Analog to Digital Converter with 10 to 11 bits accuracy (dixit MicroPython doc). The ADC converter then provide a result value from 0 to 4095. The maximum voltage resolution is 3.3V/4095 = 0.0008 V (0.8 mV).

Notice than MicroPython upscale the ADC reading to 16 bits (value between 0 and 65535) giving a common read method across all plateforms.

The following example reads analog value on A0 pin then convert it into voltage.

```
import time
from machine import ADC,Pin

adc = ADC(Pin.board.A0) # can also use ADC("A0")
while True:
 value = adc.read_u16() # read value, 0-65535 across voltage range 0.0v – 3.3v
 volts = 3.3 * value / 65535
 print( value, "=", volts )
 time.sleep( 0.5 )
```

UART

UART sends and receives data as Python bytes() Python object (so binary data).

When sending text content, the unicode string must be converted proper binary format. Don’t doing this will -sooner or later- creates encoding error.

Remarks:

- **USB_OTG1_PWR**: Default UART configuration is 8N1 (8bits data, no parity, 1 stop bit).
- UARTs are very sensitive to the **timeout** and **timeout_char** parameters while reading from the UART.

```
from machine import UART

uart1 = UART(1, baudrate=115200) # marked as Tx (GPIO_10) and Rx (GPIO_9)
uart1.write( "hello".encode("ASCII") )  # write 5 bytes
data = uart1.read(5)  # read up to 5 bytes, returns a bytes() object

uart2 = UART(2, baudrate=115200) #D7 (GPIO_7) - Rx, D8 (GPIO_8) – Tx
uart2.write( bytes( [0x4f, 0x6c, 0x69, 0x6d, 0x65, 0x78] ) ) # write Olimex
uart2.read(5) # read up to 5 bytes

uart3 = UART(3, baudrate=115200) #D5 (GPIO_5)- Rx, D6 (GPIO_6) – Tx
readed= uart3.read(20) # read up to 20 bytes
if readed != None:
    uart3.write( readed ) # send back to received data
```

Read the MicroPython UART documentation for full details on UART Class and initialization parameters.
https://docs.micropython.org/en/latest/library/machine.UART.html

SPI bus

SPI bus often uses a Chip Select pin (CS) to initiate transactions on the target device. Transaction starts by putting the CS pin LOW and terminates by putting CS pin HIGH.

Under MicroPython the CS pin must be managed in the user code!

Software SPI

Software SPI uses bit banging so it works on all pins.

```
from machine import Pin, SoftSPI

# construct a SoftSPI bus + ChipSelect signal
cs = Pin( 'D0', Pin.OUT, value=True ) # transaction not started
# polarity is the idle state of SCK
# phase=0 means sample on the first edge of SCK, phase=1 means the second
spi = SoftSPI(baudrate=100000, polarity=1, phase=0, sck=Pin('D1'), mosi=Pin('D2'),
miso=Pin('D3'))

spi.init(baudrate=200000) # set the baudrate
spi.read(10)            # read 10 bytes on MISO
spi.read(10, 0xff)      # read 10 bytes while outputting 0xff on MOSI

buf = bytearray(50)     # create a buffer
spi.readinto(buf)       # read into the given buffer (reads 50 bytes in this case)
spi.readinto(buf, 0xff) # read into the given buffer and output 0xff on MOSI

spi.write(b'12345')     # write 5 bytes on MOSI

buf = bytearray(4)      # create a buffer
cs.value( False )       # Start the transaction (example)
spi.write_readinto(b'1234', buf) # write to MOSI and read from MISO into buffer
cs.value( True )        # Ends the transaction

spi.write_readinto(buf, buf) # write buf to MOSI and read MISO back into buf
```

Hardware SPI

There are two Hardware SPIs . They can work up to 30Mhz clock.

The **SPI2** bus uses the pins **SDO** as mosi, **SDI** as miso, **SCK** controled with following ChipSelect

- **CS0** on GPIO header (Pin.board.CS0 = Pin.cpu.GPIO_AD_11)
- **CS1** on SD Card and UEXT connectors (Pin.cpu.GPIO_AD_01, no Pin.board equivalent)

The **SPI1** bus uses the pins A2 as SDO/mosi, A1 as SDI/miso, A4 as SCK. The recommended ChipSelect pin is **A3**.

```
from machine import SPI, Pin

cs = Pin( Pin.board.A3, Pin.OUT, value=True )
```

```
spi = SPI(1, 10000000)

cs.off() # Start transaction
spi.write( "Hello World".encode(ASCII) )
cs.on()  # End transaction
```

Read the MicroPython SPI documentation for full details on SPI class.
https://docs.micropython.org/en/latest/library/machine.SPI.html

I2C bus

I2C bus is very popular bus using 2 wires to carry the data and clock signals and allows multiples devices to share the same bus. I2C protocol use a 7bits address to exchange data with a target device.

I2C bus requires pull-up resistor (10 Kohms) on both SDA and SCL for its operation.

I2C bus also use two special bus state for ACK and ERROR. It is very convenient as it allows the microcontroler (and MicroPython) to be informed when something goes wrong with the target I2C device.

Software I2C

Software I2C (using bit-banging) works on all output-capable pins.

```
from machine import Pin, SoftI2C

i2c = SoftI2C(scl=Pin('D1'), sda=Pin('D2'), freq=100000)

# scan for active devices (returns a list)
print( i2c.scan() )

# read 4 bytes from device from address 0x3a. Returns a bytes()
data = i2c.readfrom(0x3a, 4)

# write binary buffer containing '12' to device with address 0x3a
i2c.writeto(0x3a, b'12')

buf = bytearray(10)  # create a buffer with 10 bytes
i2c.writeto(0x3a, buf)  # write the given buffer to device with address 0x3a
```

Hardware I2C

Two hardware I2C are available SDA1/SCL1 and SDA2/SCL2.

Usual I2C frequencies are 100 Khz and 400 Khz.

```
from machine import I2C

i2c = I2C(1, 400_000)
print( i2c.scan() )
i2c.writeto(0x76, b"Hello World")
```

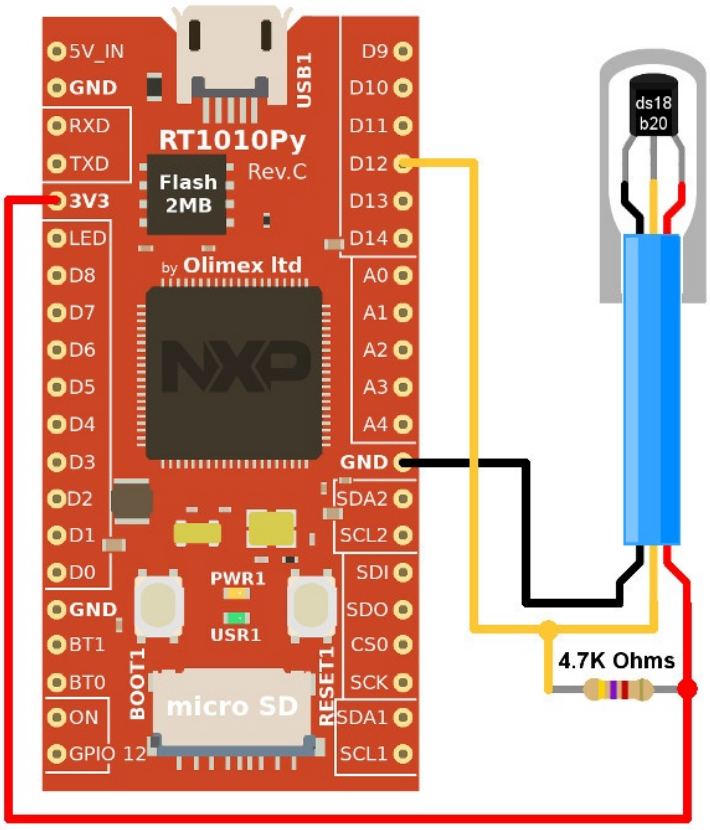
Read the MicroPython I2C documentation for full details on I2C class.
https://docs.micropython.org/en/latest/library/machine.I2C.html

OneWire bus

MicroPython can read OneWire devices like SNS-TMP-DS18B20-MAXIM.

The connection should be:

- Black to GND,
- Red to +3.3V,
- Yellow to D12. There must be also 4.7K resistor connected between D12 and +3.3V




```
from machine import Pin
import onewire, time, ds18x20

ow = onewire.OneWire(Pin.board.D12)
ds = ds18x20.DS18X20(ow)

roms = ds.scan()   # identify sensor by ROM ID

ds.convert_temp()  # Read sensors
time.sleep_ms(750) # Wait for data to be received

for rom in roms:
    print(ds.read_temp(rom)) # temperature in celcius
```

It is also possible to communicates directly on onewire bus.

```
from machine import Pin
import onewire

ow = onewire.OneWire( Pin.board.D12 ) # create a OneWire bus on GPIO12
ow.scan()           # return a list of devices on the bus
ow.reset()          # reset the bus
ow.readbyte()       # read a byte
ow.writebyte(0x12)  # write a byte on the bus
ow.write(b'123')    # write bytes on the bus
ow.select_rom(b'12345678')  # select a specific device by its ROM code
```

I2S bus

The I2S bus is used to send/receive audio data to/from an I2S capable device.

The Olimex RT1010Py offers two I2S bus, the bus 3 is dedicated to audio output only whereas the bus 1 can also receive audio stream (eg: from I2S microphone).

Board	ID	MCK	SCK_TX	WS_TX	SD_TX	SCK_RX	WS_RX
Olimex RT1010Py	1	D8	D6	D7	D4	D1	D2
Olimex RT1010Py	3	•	D10	D9	D11	•	•

Source: https://docs.micropython.org/en/latest/mimxrt/pinout.html

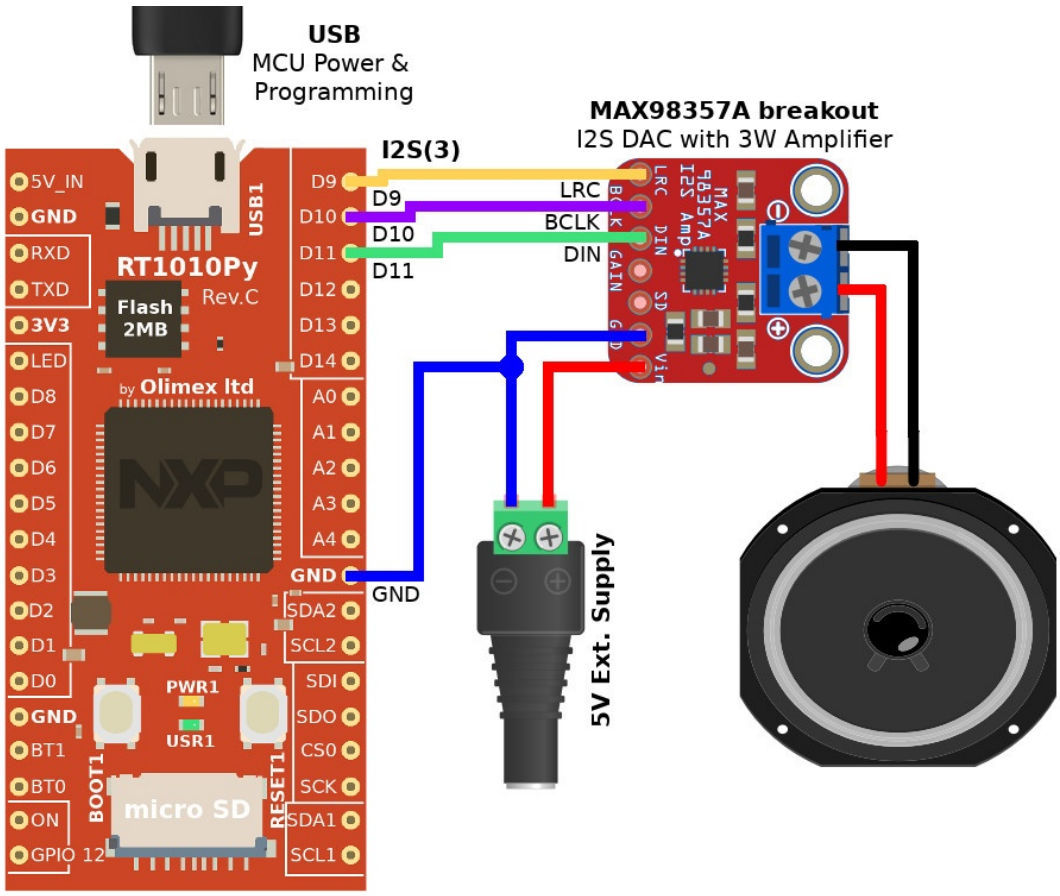
I2S audio output

An audio output I2S bus requires 3 lines to work properly:

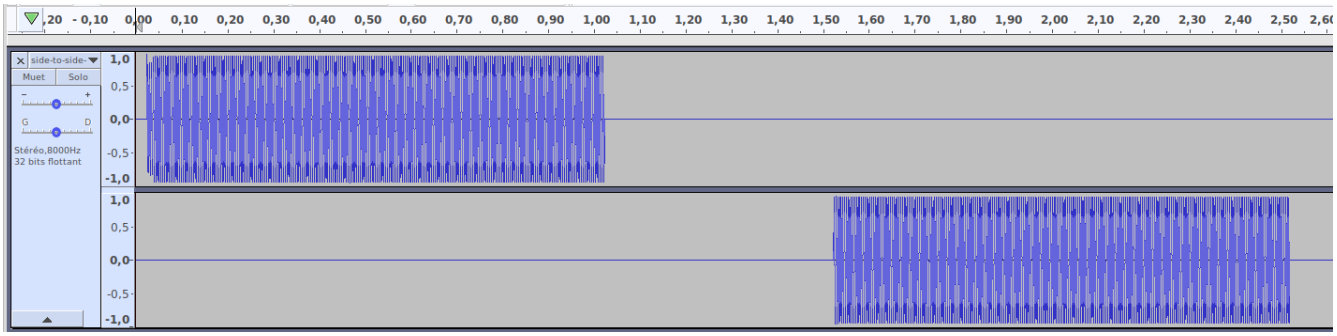
- **sck**: serial clock output
- **ws**: word clock output
- **sd**: serial data output.

```
from machine import I2S, Pin

i2s = I2S(3, sck=Pin('D10'),
             ws=Pin('D9'),
             sd=Pin('D11'),
             mode=I2S.TX, bits=16,format=I2S.STEREO, rate=44100, ibuf=4000)

# write buffer of audio samples to I2S device
i2s.write(buf)
```

The MAX98357A I2S amplified DAC can be wired as follow to the RT1010py. The stereo channels will be mixed and played on the single audio output of the MAX98357A.



The example here below is based on the work of MikeTeachMan (see links at end of section).
It uses the `side-to-side-8k-16bits-stereo.wav` file of 94 KB small enough to be stored in the Flash filesystem.
This file plays the same sound on left then on the right channel.



The following code is available in the RT1010py repository SOFTWARE/I2S_flash_play_audio.py .

```
# Play a WAV audio file out of a speaker or headphones
# RT1010py from Olimex together with MAX98357A I2S
#
# ... reduced comment ...
#
# Based on micropython-i2s-examples from miketeachman
```

```
#   https://github.com/miketeachman/micropython-i2s-examples
#   The MIT License (MIT)
#   Copyright (c) 2022 Mike Teachman
#   https://opensource.org/licenses/MIT

from machine import I2S, Pin

i2s = I2S(3, sck=Pin.board.D10, ws=Pin.board.D9, sd=Pin.board.D11, mode=I2S.TX,
          bits=16, format=I2S.STEREO, rate=8000, ibuf=5000 )

wav = open("side-to-side-8k-16bits-stereo.wav", "rb")
pos = wav.seek(44)  # advance to first byte of Data section in WAV file

# allocate sample array, memoryview used to reduce heap allocation
wav_samples = bytearray(1000)
wav_samples_mv = memoryview(wav_samples)

# continuously read audio samples from the WAV file and write them to an I2S DAC
while True:
    num_read = wav.readinto(wav_samples_mv)
    # end of WAV file?
    if num_read == 0:
        # end-of-file, exit reading loop.
        Break
    # PLay the buffer
    _ = i2s.write(wav_samples_mv[:num_read])

# cleanup
wav.close()
i2s.deinit()
print("Done")
```

Read the MicroPython I2S documentation for full details on I2S class and initialization parameters.

https://docs.micropython.org/en/latest/library/machine.I2S.html

See also the MicroPython I2S Examples on github

https://github.com/miketeachman/micropython-i2s-examples

https://github.com/miketeachman/micropython-esp32-i2s-examples

RTC : real time clock

RT1010 do have an internal clock. Once initialized that clock can be read to schedule tasks and alarms.

As long as the plateform is under power either via USB either via EXT_5V_IN (external 5V in) then the RTC will stays powered via the SNVS pin.

It is the SNVS (RTC) mode as described in the IMXRT1010CEC datasheet (see 4.1.6 Low power mode supply currents). This mode reduces the power consumption to 0,05W.

```
from machine import RTC

rtc = RTC()
rtc.datetime((2017, 8, 23, 1, 12, 48, 0, 0)) # set a specific date and time

# get date and time
print( rtc.datetime() )
```

SD card

A SD card with FAT filesytem can be used from MicroPython.

Thanks to MicroPython Virtual File System (vFs), the SD card can be mounted as a sub-folder of the existing MicroPython File System.

Attaching a SD card to MicroPython is done in 3 steps:

1. Instantiate the SPI bus and the ChipSelect pin attached to SD connector
2. Instanciate a SDCard driver and attach it to the SPI bus. This driver will expose a Unix block devices interface while managing physical read/write on the SPI bus.
3. Attach/mount the SDCard instance in the MicroPython filesystem.

Installing sdcard library

To use a SD Card, you will need the **sdcard.py** driver file written on the RT1010-py filesystem.

The **sdcard.py** is available from:

- The **micropython-lib** GitHub repository at the `micropython/drivers/storage/sdcard` path. Official source.
  https://github.com/micropython/micropython-lib/tree/master/micropython/drivers/storage/sdcard
- The **RT1010Py** GitHub repository at the `SOFTWARE` path. Copy of official version.
  https://github.com/OLIMEX/RT1010Py/tree/main/SOFTWARE

Once downloaded, the **sdcard.py** file can be copied to the RT1010-py filesystem with Thonny IDE

The **sdcard.py** driver file must be written to RT1010Py filesystem. Thonny IDE can be used to perfirm such operation.

The file can be copied to the root folder (best in the /lib subfolder more suitable for libraries).

Accessing the sdcard

The **sdtest_ol.py** script available from **RT1010Py** GitHub repository is adapted version of the official sdtest.py from Peter Hinch .

https://github.com/OLIMEX/RT1010Py/tree/main/SOFTWARE

The **sdtest_ol.py** test script mount the SD card and perform read/write tests on the medium.

The **sdmount.py** shows the minimalist script to mount the SD card.

```
# Minimalist code for Mounting the RT1010py SDCard under /sd MicroPython
# filesystem folder
```

```
#
# Requires the sdcard.py file available from official micropython-lib
#
from machine import SPI, Pin
from sdcard import SDCard
import gc, os

sd_spi = SPI(2,baudrate=20_000_000)
sd_cs = Pin( Pin.cpu.GPIO_AD_01, Pin.OUT, value=1 )
sd = SDCard( sd_spi, sd_cs )

print("Mounting SDcard to /sd" )
sd_vfs = os.VfsFat(sd)
os.mount(sd_vfs, "/sd")

print("List /sd files")
print(os.listdir("/sd"))

# Use  os.umount('/sd')  to unmount the folder
```

The result will be something like this:

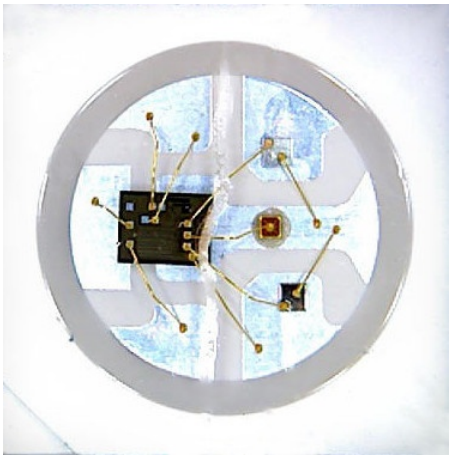
```
>>> import sdmount
Mounting SDcard to /sd
List /sd files
['System Volume Information', 'Agon-CPM2.2', 'basic_examples_tests', 'mos',
'autoexec.txt', 'bbcbasic.bin', 'README.md']
>>>
>>> # List the internal Flash (default)
>>> os.listdir()
['lib', 'sdmount.py', 'side-to-side-8k-16bits-stereo.wav']
>>>
>>> # List the root of MicroPython FileSystem
>>> os.listdir("/")
['flash', 'sd']
```

Notes:

- All Python file routine will work the same with the MicroPython Internal Flash filesystem and the mounted filesystem (a.k.a sd card). To access the sd card files, just prefix the filename with “/sd/”.
- Never forget to **`os.unmount( "/sd" )`** before removing the sd card. FAT is very sensitive to unclosed filesystem.

WS2812 (aka NeoPixel)

The WS2812 are LED light source with an internal circuit controlling the RGB color and a constant current driver for stable color rendering. The WS2812 are addressable LED meaning that each LED can have its own dedicated color.



The LEDs are daisy chained and color data is transmitted from one LED (Dout) to the following LED (Din). The first LED receives the data from the microcontroler on its DIN (data in) pin. Sending the data only requires on pin on the microcontroler. Notice that **logic level of data pin must be the same as supply voltage**!

WS2812 do use a time base protocol to transmit data. Unfortunately, the RT1010-Py MicroPython implementation do not have a specialized library yet! By chance, using a SPI MOSI (data out) pin at the right data rate with over-sampling can make the job!

The WS2812 are designed to run under 5V for maximum brightness (60mA for full white). The WS2812 can also be powered under 3.3V but offers weaker luminosity.

Install library

The support of the WS2812 is provided through a dedicated library. The library ws2812 must be installed on the MicroPython board prior running the WS2812 examples.

As NeoPixel native support is not available within the RT1010Py MicroPython implementation, the ws2812 library use over-sampling method over an hardware SPI bus clocked at the appropriate frequency. As the consequence, only the MOSI pin is in used whilst MISO and SCK are unused but reserved (because attached to the hardware SPI).

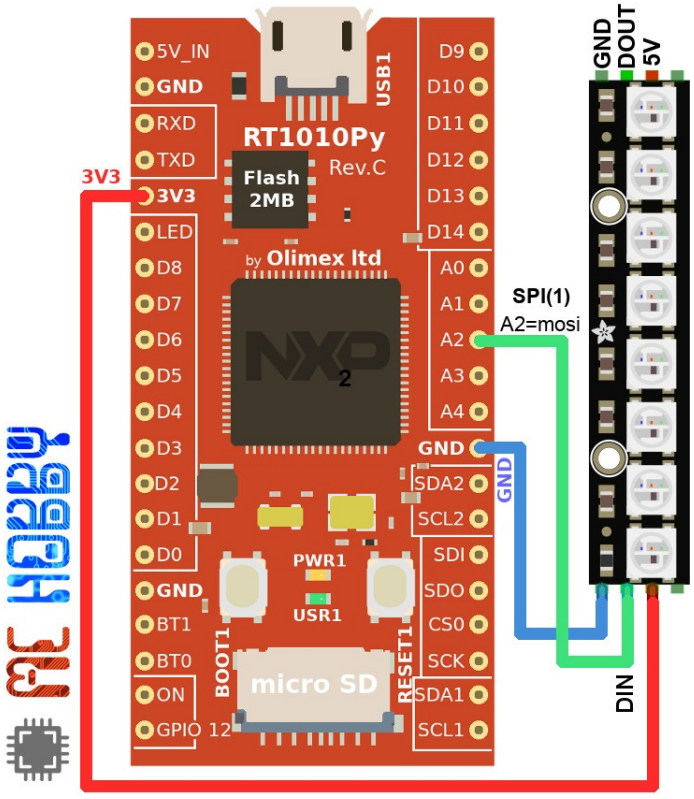
Check “RT1010Py libraries” entry for more information about the library installation.

Wiring

Using the WS2812 under 3.3V offers the most simple wiring. It is perfect to power a few LEDs from 3.3V board regulator.

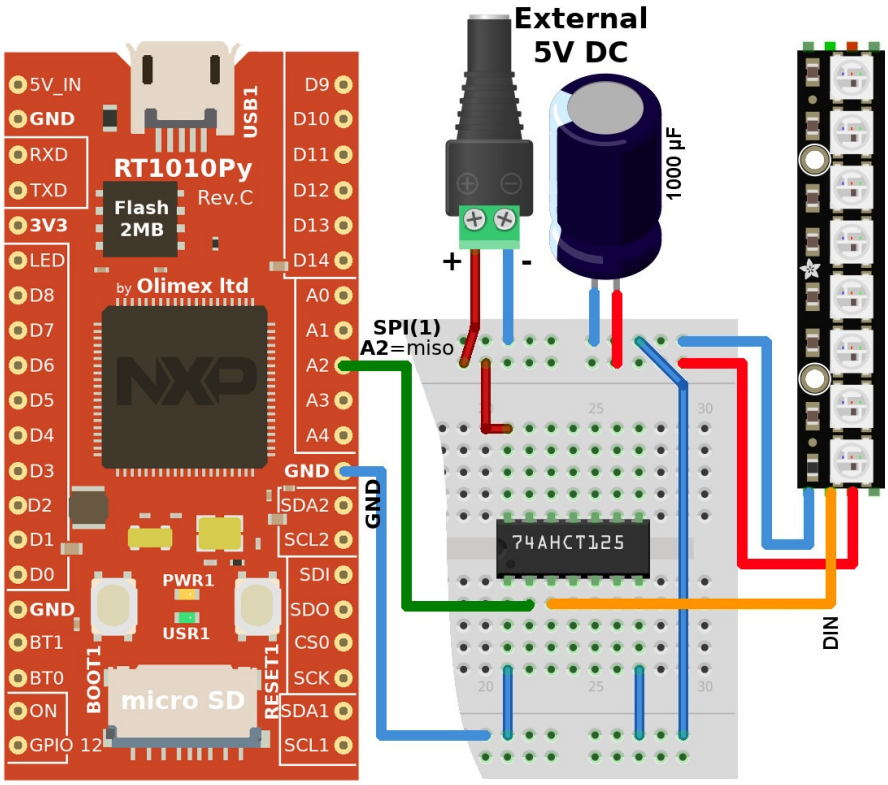
For more LEDs, it is strongly recommended to use the level shifting wiring option together with 5V supply. With 60mA max per LED, 1 meter Strip LED (60 LEDs/m) will draw up to 3.5 Amps!

3.3V wiring



5V level shifting wiring

The 74AHCT125 is used as high speed level shifting converting data from 3.3V to 5V.



Testing

The RT1010Py repository contains two major examples:

- SOFTWARE/**ws2812_test.py** : basic manipulation of a 8 LEDs stick (script detailled here below)
- SOFTWARE/**ws2812_fxdemo.py** : a few visual effect sample with a 8 LEDs stick.

```
from ws2812 import NeoPixel
from time import sleep

# NeoPixel( hardware_spi_bus_number, led_quantity )
# RT1010-Py SPI(1) => MISO=A1, MOSI=A2 (LED Output), SCK=A4
np = NeoPixel( spi_bus=1, led_count=8, intensity=1)

# Set the color of the very first LED from a
# (r,g,b) tuple. Each r,g or b in the range 0..255
np[0] = (255,0,0) # pure RED

# Color of remaining LEDs
#   (r,g,b) color can be composed with a color picker
#   https://www.w3schools.com/colors/colors_picker.asp
np[1] = (0, 255, 0) # pure GREEN
np[2] = (0, 0, 128) # half BLUE
np[3] = (255, 102, 0) # Orange
np[4] = (255, 0, 102) # Pink
np[5] = (153, 51, 255) # VIOLET
np[6] = (102, 153, 255) # Pastel BLUE
np[7] = (153, 255, 153) # Pastel GREEN
```

```
# Send color data over the LEDs
np.write()
sleep(2)

# A collection of colors
colors = [ (255,0,0), (0, 255, 0), (0, 0, 128), (255, 102, 0) , (255, 0, 102),
   (153, 51, 128), (102, 153, 128), (153, 255, 128) ]

for color in colors:
    # fill() replicate the color across all LEDs
    np.fill( color )
    np.write()
    sleep(2)

# Switch OFF the LEDs with BLACK color
np.fill( (0,0,0) )
np.write()
```

Watchdog

The once activated watchdog must be feed before its timeout otherwise the watchdog resets the microcontroler. In such case, the `machine.reset_cause()` will returns `WDT_RESET`.

```
from machine import WDT
dog = WDT( timeout=5000 ) # 5 sec before activation
dog.feed() # feed the watchdog
```

The **watchdog.py** script available from **RT1010Py** GitHub repository show a rudimentary example around the watchdog.

The script first read the last reset_cause then displays its value together with a user friendly label.

Then the watchdog is created and initialized. Watchdog immediately starts countdown. The feed() call must occurs before the watchdog reach 0 (otherwise the watchdog will resets the microcontroler).

The BOOT1 button is associated with the btn_press() callback. When pressed, the callback feed() the watchdog.

Wait 5 seconds without pressing the button to get the watchdog activated (microcontroler resets).

```
from machine import WDT, Pin
from machine import reset_cause, PWRON_RESET, SOFT_RESET, WDT_RESET

RESET_CAUSE = { PWRON_RESET:'PWRON_RESET', SOFT_RESET:'SOFT_RESET',
WDT_RESET:'WDT_RESET' }

rc = reset_cause()
print( "Reset cause: %i (%s)" % (rc, RESET_CAUSE[rc] ) )

print( "Set Watchdog to 5sec" )
print( "Press Boot1 to feed watchdog before" )
print( "  it resets the microcontroler" )

btn = Pin( Pin.board.BT1, Pin.IN ) # Boot1 button. 0=pressed, 1=released
dog = WDT( timeout=5000 ) # 5 sec before activation

def btn_press( btn ):
 global dog
 print( "Feed watchdog!" )
 dog.feed()

# Attach event to button
btn.irq( handler=btn_press, trigger=Pin.IRQ_FALLING )
```

Here some results produced by the script from a REPL session.

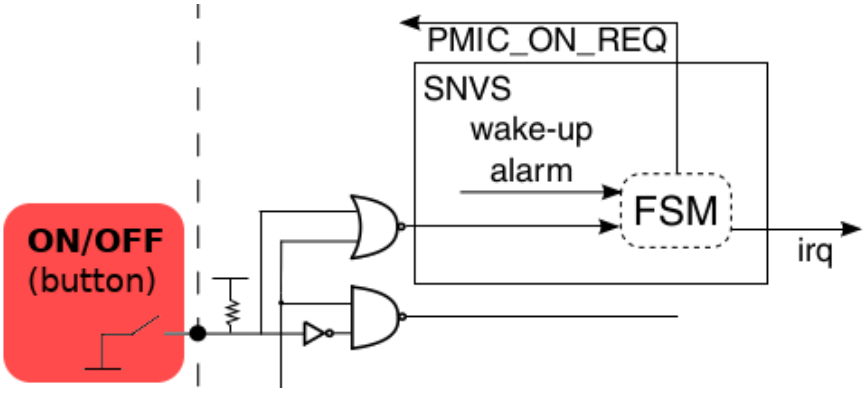
```
$ mpremote
Connected to MicroPython at /dev/ttyACM0
Use Ctrl-] or Ctrl-x to exit this shell

>>> import watchdog
```

```
Reset cause: 3 (WDT_RESET)
Set Watchdog to 5sec
Press Boot1 to feed watchdog before
  it resets the microcontroler
>>> Feed watchdog!
Feed watchdog!
Feed watchdog!
Feed watchdog!
device disconnected
```

Power Management

ON pin

That pin is used to control the Power Management Unit (see documentation about ONOFF MCU signal). The ON pin can be used to wake-up/shutdown the microcontroler.



The PM unit is driven by 3 configurations parameters:

- **debounce time**: 0, 50, 100, 500 ms. It is the minimal time to detect the button press.
- **off to on time**: while in OFF state, pressing button longer than “off to on” time will switch the state to ON.
- **Max time-out**: 5, 10, 15 sec or disabled. This is the minimal button press time to request a shutdown.

While in ON state, pressing the button longer than debounce time will raise the *Power Off* interrupt. Unfortunately, the *Power Off* interrupt is not available with the MicroPython script.

In its default MicroPython configuration, the ON pin can wake-up the MCU from deep sleep. The ON pin is activated by placing its to LOW (GND).

The current PMIC configuration can be read with the **`pmic_config_read.py`** example as shown the excerpt here below.

```
...
LPRC On-time : 500ms (0b0)
LPRC Debounce : 50ms (0b0)
LPRC Btn_Press_time : 5s (0b0)
...
```

With the current configuration, pushing the ON button (put the ON pin LOW) for 5 sec will shutoff the microcontroler (the PWR LED goes out).

Pressing again the ON button for ½ sec will switch on the microcontroler. MicroPython will perform a full boot cycle (execution of `boot.py` + `main.py`) but LPRC will survive the ON/OFF Power Management Unit cycle.

Modifying the configuration

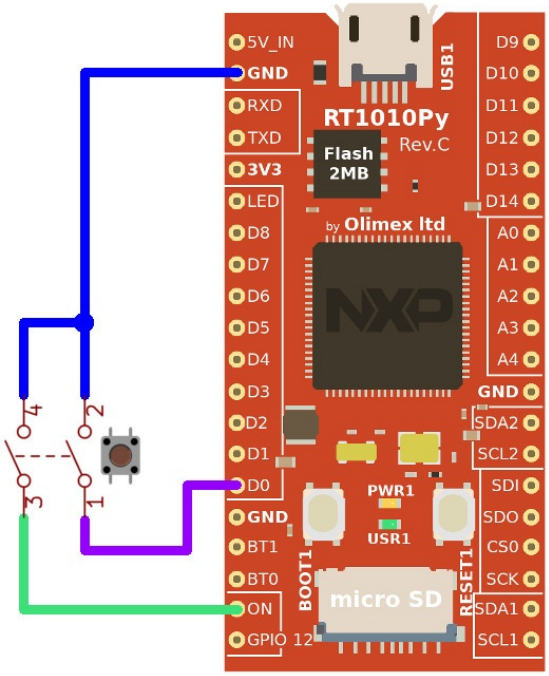
The configuration of the Power Management Unit can be updated by accessing the registers.

This is demonstrated in the **`pmic_config_set.py`** script that change the **off to on** time from 5sec to 10sec.

Power off interrupt

The internal MCU Power Off interrupt is not available to MicroPython script but it is possible to capture it anyway thank to an external GPIO.

A **bipolar push button** can be used to control the "ON" signal as well as an input pin (eg: D0) attached to an IRQ fired by the same button controlling the "ON" signal..



Do not connect the D0 pin directly on the ON pin because this will interfere with the off-to-on stage.

The following script will fire an IRQ each time a falling edge is detected on the “ON” signal.

```
from machine import Pin

def power_off_cb( obj ):
    print( "Power off pressed" )

# Check doc for more details on parameters
# https://docs.micropython.org/en/latest/library/machine.Pin.html
```

```
poff = Pin( Pin.board.D0, Pin.IN )
poff.irq( handler=power_off_cb, trigger=Pin.IRQ_FALLING, hard=True )
```

In this example, the message “Power off pressed” is displayed in the REPL session on the falling edge of the signal (when the button is pressed). The `hard` parameter creates an hardware irq which is more responsive. Notice that several irq calls may happens for a single push due transient effect, appropriate development method must be applied.

See the “ISR Rules” recommendations on the official MicroPython page:

https://docs.micropython.org/en/latest/reference/isr_rules.html

LPGPR register

The SNVS_LP General Purpose register are 4 registers (0..3) for a total of 128 bits/16bytes available for software application. The LPGPR registers retains data during the ON/OFF cycle.

Writing registers

The **`pmic_gpr_set.py`** script will write data into the LPGPR register with some helper functions. The excerpt here below shows the main activity of the script

```
# Store the Day,Month,Hour,Sec in the LPGPR[0]
rtc = RTC()
_year, _month, _day, _dow, _hour,_min, _sec, _ms = rtc.datetime()
data = bytes([_day, _month, _hour, _min])
print( "Encode %i/%i %i:%i" % (_day, _month, _hour, _min) )
print(  data, "=> LPGPR[0]")
LPGPR_set( 0, data )
print()

# Store 12 bytes message into remaining storage
sMessage = "OlimexRT1010"
data = sMessage.encode("ASCII")
print( "Encode message %s" % sMessage )
print( data, "=> LPGPR[1..3]" )
LPGPR_set( 1, data[0:4] )
LPGPR_set( 2, data[4:8] )
LPGPR_set( 3, data[8:12] )
```

The execution of the script show the following results in the REPL session:

```
Encode 13/10 1:12
b'\r\n\x01\x0c' => LPGPR[0]

Encode message OlimexRT1010
b'OlimexRT1010' => LPGPR[1..3]

Now you can inspect the content with pmic_gpr_get.py script!
Register will survive the On/OFF cycle
```

Reading registers

The **`pmic_gpr_get.py`** script will read the LPGPR registers with some helper functions.

```
# Display the content as raw content
print( "--- RAW ----------------------------------------" )
for i in range(4):
    data = LPGPR_get( i )
    print( "LPGPR[%i] :" % i, data )

print( "--- Binary -------------------------------------" )
for i in range(4):
    value = LPGPR_as_int( i ) # Get the value as u32int
    print( "LPGPR[%i] :" % i, pretty_bin32(value) )

print( "--- Bytes --------------------------------------" )
for i in range(4):
    value = LPGPR_get( i ) # Get the value as u32int
    print( "LPGPR[%i] :" % i, ", ".join( "%i" % _ for _ in value ) )
```

Here the result of the script clearly shows the stored text in the LPGPR register # 1, #2 and #3 (raw version). The bytes representation allows to read back the stored date/heure from LPGPR register #0.

```
--- RAW ----------------------------------------
LPGPR[0] : b'\r\n\x01\x0c'
LPGPR[1] : b'Olim'
LPGPR[2] : b'exRT'
LPGPR[3] : b'1010'
--- Binary -------------------------------------
LPGPR[0] : 00001101 : 00001010 : 00000001 : 00001100
LPGPR[1] : 01001111 : 01101100 : 01101001 : 01101101
LPGPR[2] : 01100101 : 01111000 : 01010010 : 01010100
LPGPR[3] : 00110001 : 00110000 : 00110001 : 00110000
--- Bytes --------------------------------------
LPGPR[0] : 13, 10, 1, 12
LPGPR[1] : 79, 108, 105, 109
LPGPR[2] : 101, 120, 82, 84
LPGPR[3] : 49, 48, 49, 48
```

know issue

After a ON/OFF power cycle, the LPGPR[3] register have been zeroed for an unknown reason.

```
--- RAW ----------------------------------------
LPGPR[0] : b'\r\n\x01\x0c'
LPGPR[1] : b'Olim'
LPGPR[2] : b'exRT'
LPGPR[3] : b'\x00\x00\x00\x00'
--- Binary -------------------------------------
LPGPR[0] : 00001101 : 00001010 : 00000001 : 00001100
LPGPR[1] : 01001111 : 01101100 : 01101001 : 01101101
LPGPR[2] : 01100101 : 01111000 : 01010010 : 01010100
LPGPR[3] : 00000000 : 00000000 : 00000000 : 00000000
--- Bytes --------------------------------------
LPGPR[0] : 13, 10, 1, 12
```

```
LPGPR[1] : 79, 108, 105, 109
LPGPR[2] : 101, 120, 82, 84
LPGPR[3] : 0, 0, 0, 0
```

Idle

The `machine.deepsleep()` is useful to reduce power consumption for a short or long period of time without impact on the performances. In idle mode, the peripherals are not suspended and resumes as soon as an interrupt occurs (or at most one millisecond later).

To reduce power consumption see `time.sleep()`, `time.sleep_ms()`, `machine.lightsleep()` (*not implemented on I.MXRT1010*) and `machine.deepsleep()`.

DeepSleep

The `machine.deepsleep()` can be used with a sleep time (in ms) or without parameter (sleep forever).

The sleep can be interrupted at any moment by putting the ON pin to LOW (GND) for at least debounce_time (that can be configured to 0, 50, 100, 500 ms).

When exciting deepsleep, MicroPython restart with a complete boot sequence meaning that `boot.py` is also executed before starting `main.py` .

The `machine.reset_cause` is `machine.PWRON_RESET` and the is memory cleared.

Libraries

MicroPython lib

https://github.com/micropython/micropython-lib

This is a repository of packages designed to be useful for writing MicroPython applications.

The packages here fall into categories corresponding to the four top-level directories:

- **python-stdlib**: Compatible versions of modules from The Python Standard Library. These should be drop-in replacements for the corresponding Python modules, although many have reduced functionality or missing methods or classes (which may not be an issue for most cases).
- **python-ecosys**: Compatible, but reduced-functionality versions of packages from the wider Python ecosystem. For example, a package that might be found in the Python Package Index.
- **micropython**: MicroPython-specific packages that do not have equivalents in other Python environments. This includes drivers for hardware (e.g. sensors, peripherals, or displays), libraries to work with embedded functionality (e.g. bluetooth), or MicroPython-specific packages that do not have equivalents in CPython.
- **unix-ffi**: These packages are specifically for the MicroPython Unix port and provide access to operating-system and third-party libraries via FFI, or functionality that is not useful for non-Unix ports.

Hardware Agnostic MicroPython drivers

https://github.com/mchobby/esp8266-upy

Contains a collection of +/-100 libraries supporting hardware sensors and actuators (the so called "*Driver*").

The idea behind "*Hardware Agnostic*" is to have library code free from any specific MicroPython platform. The collection is designed to be self documented! So the repository also contains wiring diagram and code examples.

The repository also includes 15+ drivers for Olimex UEXT modules.

Note:

MCHobby repositories also contains many MicroPython based project.
https://github.com/mchobby?tab=repositories

Awesome MicroPython

Collection of MycroPython resources at https://awesome-micropython.com/

AI

- MicroMLP - A micro neural network multilayer perceptron for MicroPython (used on ESP32 and Pycom modules).
- MicroPython-NeuralNetwork - Neural Network for MicroPython.
- upython-chat-gpt - ChatGPT for MicroPython.
- emlearn-micropython - Efficient Machine Learning engine for MicroPython.

Audio

- micropython-jq6500 - Driver for JQ6500 UART MP3 modules.
- KT403A-MP3 - Driver for KT403A, used by DFPlayer Mini and Grove MP3 v2.0.
- micropython-buzzer - Play Nokia compose and mid files on buzzers.
- micropython-dfplayer - Library to control the DFPlayer mini MP3 player module.
- micropython-dfplayer - Driver for DFPlayer Mini using UART.
- micropython-longwave - WAV player for MicroPython board.
- micropython-vs1053 - Asynchronous driver for VS1053b MP3 player.
- micropython-midi - A MIDI implementation example for MicroPython.
- upy-rtttl - Python Parser for Ring Tone Text Transfer Language (RTTTL).
- micropython-i2s-examples - Examples for I2S support on microcontrollers that run MicroPython.
- micropython-osc - A minimal OSC client and server library for MicroPython.
- micropython-sgtl5000 - Library for SGTL5000 Low Power Stereo Codec w/ Headphone Amp.
- umidiparser - MIDI file parser for MicroPython, CircuitPython and Python.
- micropython-tas2505 - MicroPython driver for the Texas Instruments TAS2505 Digital Input Class-D Speaker Amplifier.

Communications

APIs

- micropython-utelegram - Telegram API wrapper for MicroPython.
- uEagle - MicroPython Rainforest EAGLE client.
- micropython-youtube-api - YouTube API in MicroPython.
- micropython_esp8266_tweetbot - Tweet bot for MicroPython v1.8.4 (ESP8266).
- telegram-upy - Telegram API wrapper for MicroPython.
- micropython-thingspeak - Library for sending data to thingspeak.com from IoT devices running MicroPython (such as ESP8266).

- micropython_pushbullet - Simple example of how to use PushBullet with MicroPython on ESP8266.
- esp32-youtube-display - Display YouTube metrics using Google API and MicroPython.
- micropython-spotify-web-api - A library for using Spotify's web API from a IoT device with MicroPython.
- micropython_demo_bot - Little example of how to create a bot for Telegram.
- micropython-basicdweet - A python module for very basic APIs of the free dweet service.
- micropython-dweeter - A python module for messaging through the free dweet service.
- micropython-cryptodweet - A python module for very basic APIs of the free dweet service with encryption.
- micropython-linenotify - MicroPython library for sending notifications to Line Notify with ESP8266 and ESP32.

Authentication

- micropython-firebase-auth - Firebase Auth implementation for MicroPython.

Bluetooth

- PyBoard-HC05-Android - Pyboard HC05 Bluetooth adapter example application.
- uble - Lightweight Bluetooth Low Energy driver written in pure Python for MicroPython.
- MicroPythonBLEHID - Human Interface Device (HID) over Bluetooth Low Energy (BLE) GATT library for MicroPython.
- upyble - Command line tool for Bluetooth Low Energy MicroPython devices.
- micropython-xiaomi-ble-adv-parse - Passively retrieve sensor data from some Xiaomi Bluetooth Low Energy (BLE) sensors.
- mijia-temphum-upy - MicroPython library to read certain Xiaomi Mijia BLE temperature & humidity sensors.

CAN

- micropython-spacecan - Spacecan is a MicroPython implementation of the SpaceCAN protocol for embedded systems.
- Robomaster-Micropython - Robomaster S1 - MicroPython CAN BUS controller.
- micropython-mcp2515 - MicroPython MCP2515 driver, porting from Arduino MCP2515 CAN interface library.
- microPython_MCP2515 - A MicroPython library for the MCP2515 CAN bus controller.

Compression

- ufastlz - MicroPython wrapper for FastLZ, a lightning-fast lossless compression library.
- tamp - A low-memory, MicroPython-optimized, DEFLATE-inspired lossless compression library.

Cryptography

- ucryptography - Lightweight porting of pyca/cryptography to MicroPython based on ARM Mbed TLS.
- mpyaes - MicroPython module for AES encryption.
- micropython-aes - AES algorithm with pure python implementation.
- ucrypto - MicroPython package for doing fast RSA and elliptic curve cryptography, specifically digital signatures. ECDSA API design inspired from fastecdsa and implementation based on tomsfastmath.
- ucryptoauthlib - Lightweight driver for Microchip Crypto Authentication secure elements written in pure Python for MicroPython.
- embit - A minimal Bitcoin library for MicroPython and Python 3 with a focus on embedded systems.
- microotp - An ESP8266 MicroPython OTP Generator.
- micropython-rsa-signing - RSA signing on MicroPython.
- micropython-cryptomsg - A MicroPython module to encrypt and decrypt messages with AES CBC mode.
- mprsa - A MicroPython module for creating, importing, and exporting RSA keys in DER and PEM formats with PKCS#1, PKCS#8, and X.509/SPKI structures, and signing/verifying and encryption/decryption using blinding and SHA-1 and SHA-256 hashing algorithms.
- mpy-mbedtls - MicroPython bindings for some MbedTLS EC and x509 cert/csr functions.
- micropython-cryptocfb - A Python module to encrypt and decrypt data with AES-128 CFB mode.
- tscp - An endpoint-to-endpoint encryption based on Diffie-Hellman-Merkle with TLS1.3 styled handshake using MicroPython.

DNS

- ICantBelieveItsNotDNS - "I Can't Believe It's Not DNS!" (ICBIND) is an authoritative DNS server for the ESP8266 written in MicroPython.
- MicroDNSSrv - A micro DNS server for MicroPython to simply respond to A queries on multi-domains with or without wildcards (used on Pycom modules & ESP32).
- tinydns - Very simple DNS async server for MicroPython.
- micropython-captiveportal - Minimal async captive portal for MicroPython (compatible with uasyncio v3/MicroPython 1.13+ as well as earlier versions).
- Micropython-DNSServer-Captive-Portal - MicroPython WiFi AP Captive Portal with DNS and Web Server.

ESP-NOW

- mesh-espnow-micropython - Dynamic Secure Mesh for Collaborative Nodes of IoT devices.

Ethernet

- Official WIZnet5k - Driver for the WIZnet5x00 series of Ethernet controllers.
- micropy-ENC28J60 - ENC28J60 Ethernet chip driver for MicroPython (RP2).
- RP2040 Ethernet example - Ethernet driver, example Python code and YouTube.
- micropython-ch9121 - MicroPython library for controlling CH9121 Ethernet modules.

FTP

- micropython-ftplib - An FTP client library for MicroPython.
- FTP-Server-for-ESP8266-ESP32-and-PYBD - Small FTP server for ESP8266/ESP32/Pyboard on the MicroPython platform.
- MicroFTPServer - Minimal FTP Server that can run on an ESP8266 with MicroPython.
- micropython-uaioftp - Lightweight FTP library for MicroPython.
- FtpTiny-Micropython - Really small FTP server that runs in a thread.

GPS

- micropyGPS - Full featured GPS NMEA sentence parser.
- micropython-gnssl76l - MicroPython I2C driver for Quectel GNSS L76-L (GPS).
- mpy-agps - MicroPython implementation of assisted location services (AGPS).
- Asynchronous GPS driver - Receive and parse GPS data as a uasyncio task.

GSM

- micropython-upyphone - A GSM phone using Pyboard and SIM800l.
- micropython-sim800 - MicroPython driver for SIM800.
- sim800 - Library for interfacing with SIM800 module in MicroPython.
- MicroPython-AM7020 - MicroPython driver for AM7020 Narrowband Internet of Things (NBIoT) module.

HTTP

- mrequests - A HTTP client module (not only) for MicroPython with an API similar to requests.

IoT

- microhomie - MicroPython implementation of the Homie MQTT convention for IoT.
- uPyEcho - Emulated Belkin WeMo device that works with Amazon Echo (Alexa) using MicroPython on an ESP32.
- SonosRemote - A remote for Sonos installations running on an ESP8266 and using Sonos HTTP API.
- micropython-home-assistant - MicroPython-based scripts to extend your Home Assistant-driven home automation projects.
- micropython-iot - An approach to designing IoT applications using ESP8266, ESP32 or Pyboard D endpoints.

- iot-core-micropython - Use MicroPython to connect to Google Cloud IoT Core.
- SmartUPy - Controlling "Tuya-type" smart power outlets using MicroPython.
- aws-iot-GET-POST-loop - MicroPython code which uses the AWS IoT REST API to GET/POST device state info.
- sensor-mqtt-homeassistant - An ESP8266/ESP32 MicroPython-based sensor platform for GPIO, DHT, analog, LED and more. Includes remote updates for .py code from web server and MQTT/Home Assistant integration.
- micropython-ha-mqtt-device - MicroPython module which allows creating Entites for HomeAssistant using MQTT Discovery.
- ESP8266-Home-Assistant-Smart-Socket - This MicroPython project is to hack a Hyleton313 cheap WiFi smart socket.
- ESP8266-Home-Assistant-RGB-Bulb - This MicroPython project is to hack a TYWE3S board in a cheap WiFi RGB Bulb.
- uPyIoT - Connect an M5Stack ATOM running MicroPython to the Google Cloud Platform (GCP) to collect air-quality variables obtained from reading sensors.
- micropython-switchbot-thermometer-hygrometer - Read SwitchBot Thermometer/Hygrometer via Bluetooth.

IR

- micropython-necir - NEC infrared capture for TL1838 IR receiver LEDs.
- Micropython-IR - Pyboard infrared remote sniff and replay.
- micropython_ir - Nonblocking device drivers to receive from IR remotes and for IR "blaster" apps.
- micropython-amg88xx - Driver for Grid-EYE thermal infrared array sensor (Adafruit 3538).
- micropython-ys-irtm - MicroPython examples for YS-IRTM 5V NEC Infrared UART transceivers.
- esp8266_ir - Control IR signal by WebSocket.
- micropython_espX_IR_Transceiver - MicroPython ESP32 IR Transceiver.
- pico-ir - IR library for Raspberry Pi Pico.
- esp32-ir-remote - A MicroPython project for running ESP32 IR remotes.

LoRa

- loraE22 - A MicroPython class for the Ebyte E22 Series LoRa modules.
- micropython-lora - MicroPython library for controlling a Semtech SX127x LoRa module over SPI.
- micropython-aiolora - MicroPython library for controlling a Semtech SX127x LoRa module with asyncio API.
- micropython-rylr - MicroPython library for controlling Reyax LoRa modules (RYLR896, RYLR406).
- silvergeko_rfm9x - Porting to MicroPython of adafruit_rfm9x.py library.

LoRaWAN

- uPyLoRaWAN - ESP32 using MicroPython meets LoRa and LoRaWAN.
- SX127x_driver_for_MicroPython_on_ESP8266 - SX127x (LoRa transceiver) driver for (Micro)Python on ESP8266/ESP32/Raspberry Pi.
- LightLora_MicroPython - Lightweight Interrupt-driven Semtech SX127x Library for MicroPython.
- u-lora - Raspi-lora for MicroPython.
- sx127x_esp - Connect Ra-01 module base on LoRaTM sx127x chip to ESP8266/ESP32 under MicroPython.
- nanoserver - MicroPython embedded LoRaWAN server.
- micropySX126X - Semtech SX126X LoRa driver for MicroPython and CircuitPython.

MDNS

- micropython-mdns - A pure Python implementation of MDNS with support for Service Discovery.

Modbus

- micropython-modbus - MicroPython port of modbus-tk.
- micropython-modbus - Modbus Master library for MicroPython ESP32 devices. Based on pycom-modbus from Pycom.
- mp_modbus - Modbus library for MicroPython.
- micropython-modbus - ModBus TCP and RTU library supporting client and host mode. Based on pycom-modbus from Pycom.

MQTT

- micropython-mqtt - A 'resilient' asynchronous MQTT driver. Plus a means of using an ESP8266 to bring MQTT to non-networked targets.
- MQBoard - A micro-framework for using MQTT with asyncio on MicroPython boards, primarily on the ESP32.
- pysmartnode - MicroPython Smart Home framework.
- umqtt_aws_iot - Publish UMQTT messages with MicroPython to AWS IoT.
- sonoff-mqtt by davea - MicroPython scripts to control Sonoff/ESP8266 using MQTT.
- micropython-sonoff-switch - Implements an MQTT-controllable switch for the iTead Sonoff Switch using MicroPython.
- micropython-thingspeak-mqtt-esp8266 - Publish and Subscribe to ThingSpeak using MQTT with MicroPython running on ESP8266/ESP32 platforms.
- uMQTT - MQTT publish for MicroPython on the WiPy board.
- micropython-mqtt - Async MQTT library with auto reconnect for MicroPython devices such as the ESP32 or Pycom devices.

- micropython-adafruit-mqtt-esp8266 - Using MQTT to Publish/Subscribe to Adafruit IO. MicroPython/CircuitPython implementation on ESP8266/ESP32.
- MicropythonCayenneMQTTClient - A port of the Python Cayenne MQTT Client to MicroPython.
- mqtt_upython - MQTT Client using MicroPython on ESP8266.
- tinymqtt - Async MQTT client for MicroPython.
- micropython-mqtt-thingspeak - Publish and Subscribe to ThingSpeak using MQTT with MicroPython.

NBD

- unbd - Micro implementation of network block device (NBD) for MicroPython.

NFC

- micropython-nfc - Using NFC with MicroPython.
- micropython_pn532 - Driver for PN532 NFC/RFID breakout boards based on Adafruit CircuitPython (UART).
- NFC_PN532_SPI - Partial port of Adafruit CircuitPython to MicroPython of PN532 NFC/RFID control library (SPI).

NTP

- esp8266_ntp_webserver - MicroPython + ESP8266 + NTP + web server.
- micropython-ntpd - An implementation of an NTP daemon in MicroPython.
- micropython_ntpserver - An NTP server written for MicroPython.
- micropython-ntpclient - NTP client for MicroPython using uasyncio.

OneWire

- Official OneWire - For devices using the OneWire bus, eg Dallas DS18x20.
- Onewire_DS18X20 - Classes for driving the DS18x20 sensor with the OneWire protocol for Pycom MicroPython.
- micropython_arduino_control - MicroPython library to control an Arduino remotely, with corresponding Arduino code.

Onkyo EISCP

- eiscp-micropython - MicroPython port for the Onkyo-EISCP protocol used, among others, by Pioneer.

OTA

- micropython-ota-updater - OTA Updater for MicroPython.
- Micropython-ESP32-OTA - MicroPython updater based on rdehuyss/micropython-ota-updater.
- senko - Simplest OTA update solution for your MicroPython projects.

Radio

- micropython-radio - Protocols for nRF24L01 2.4GHz radio modules.
- micropython-rfsocket - MicroPython implementation of popular 433MHz-based RFSockets.
- Official nRF24L01 - Official driver for nRF24L01 2.4GHz radio modules.
- micropython_remote - Capture and replay 433MHz remote control codes. Control remote switched power adaptors.
- micropython-ys-rf34t - MicroPython examples using YS-RF34T 433MHz ASK/OOK UART transceivers.
- FM_Talkie - FM Walkie Talkie using RDA5820N.
- micropython-TEA5767 - MicroPython ESP8266/ESP32 driver for TEA5767 FM radio module.
- micropython-ppm-decoder - Utility for decoding an R/C receiver PPM frame signal.
- ESP32-433Mhz-Receiver-and-Tools - ESP32 433MHz receiver written in MicroPython and tools for Windows.
- ESP32-433Mhz-Transmitter - A pure MicroPython RF transmitter. You can create and add your own encoder.
- pico_jjy_tx - JJY transmitter for Raspberry Pi Pico W.
- pico_dcf77_tx - DCF77 transmitter for Raspberry Pi Pico W.

RC receiver

- micropython-ppm_reader - Library to decode PPM signals coming from a RC receiver.

REPL

- webrepl - MicroPython WebREPL.
- zepl - MicroPython WebREPL Console Application using ZeroMQ.
- jupyter_micropython_remote - Jupyter kernel to directly execute code on a MicroPython board over the serial/web REPL.
- FBConsole - Framebuffer console class for MicroPython.

RFID

- micropython-mfrc522 - Driver for NXP MFRC522 RFID reader/writer.
- micropython-wiegand - Wiegand protocol reader.
- urdm6300 - A MicroPython driver for the popular RDM6300 RFID card reader.

RPC

- ujrpc - JSON RPC for MicroPython.

RTC

- micropython-tinyrtc-i2c - Driver for DS1307 RTC and AT24C32N EEPROM.
- Micropython_TinyRTC - Driver for DS1307 RTC.
- micropython-mcp7940 - Driver for the Microchip MCP7940 RTC.

- micropython-ds1302-rtc - DS1302 RTC Clock driver for MicroPython.
- DS3231micro - MicroPython library for DS3231.
- micropython-ds1307 - MicroPython driver for DS1307 RTC.
- esp-ds3231-micropython - A DS3231 library for ESP8266/ESP32 with MicroPython.
- PCF8563_PythonLibrary - MicroPython library for NXP PCF8563 Real-time clock/calendar.
- DS3231 - MicroPython module for the DS3231 clock from Maxim Integrated.
- DS1307 - MicroPython driver for the DS1307 real time clock.
- micropython-DS3231-AT24C32 - MicroPython driver for DS3231 RTC.

Serial

- mpy-miniterm - Tool for seamless serial debug and file synchronisation with MicroPython devices via the serial REPL.
- MicroPython-MorseCode - International Morse Code using a microcontroller with MicroPython.
- I2C Slave - Uses the Pyboard's I2C slave mode to implement a full duplex asynchronous link. Principal use case is for ESP8266 which has only one UART.
- microSDI12 - A mini SDI-12 implementation for getting sensor info over RS-485.

Serialization

- micropython-msgpack - MessagePack serialisation library optimised for MicroPython.
- micropython-uprotobuf - A lightweight implementation of Google's Protocol Buffers (protobuf) for MicroPython.
- minipb - Mini Protobuf {de}serializer in pure Python.
- ucbor - Lightweight implementation of cbor for MicroPython.
- upy-msgpack - A lightweight MessagePack (de)serialization library (not only) for MicroPython.

SMTP

- uMail - A lightweight, scalable SMTP client for sending email in MicroPython.

Sockets

- XAsyncSockets - XAsyncSockets is an efficient Python/MicroPython library of managed asynchronous sockets.

SOCKS

- micropython-socks - MicroPython library implementing SOCKS server.

TCP

- us2n - MicroPython bridge between UART and TCP for the ESP32.

Telnet

- MicroTelnetServer - Simple telnet server for MicroPython and the ESP8266 allowing telnet clients access to the REPL.

Text-to-Speech

- micropython-SYN6988 - MicroPython library for the VoiceTX SYN6988 text to speech module.

VoIP

- uPyVoip - VoIP for MicroPython ESP32 with Interactive Voice Response.

Web

- MicroWebSrv - A micro HTTP web server that supports WebSockets, HTML/Python language templating and routing handlers, for MicroPython (used on Pycom modules & ESP32).
- MicroWebSrv2 - The last micro web server for IoTs (MicroPython) or large servers (CPython), that supports WebSocket, routes, template engine and with really optimized architecture (mem allocations, async I/Os).
- tinyweb - Simple and lightweight HTTP async server for MicroPython.
- upy-websocket-server - MicroPython (ESP8266) WebSocket server implementation.
- micropython-captive-portal - A captive portal demo for MicroPython.
- uPyPortal - A captive portal for MicroPython using ESP32 (Wemos).
- ESP8266WebServer - ESP8266 web server for MicroPython.
- microCoAPy - A mini client/server implementation of CoAP (Constrained Application Protocol) into MicroPython.
- micropyserver - MicroPyServer is a simple HTTP server for MicroPython projects.
- MicroRESTCli - A micro JSON REST web client based on MicroWebCli for MicroPython (used on Pycom modules & ESP32).
- micropython-noggin - A very simple web server for MicroPython.
- uwebsockets - MicroPython WebSocket implementation for ESP8266.
- microdot - The impossibly small web framework for MicroPython.
- micropython-nanoweb - Full async MicroPython web server with small memory footprint.
- MicroWebCli - A micro HTTP web client for MicroPython (used on Pycom modules & ESP32).
- micropython-configserver - Captive portal for MicroPython including a dumb DNS server and a web server to configure WiFi networks.
- micropython-aioweb - A minimalist asyncio web framework for MicroPython.
- thimble - A tiny web framework for MicroPython.
- CaptiveWebServer - Simple MicroPython web server for serving a website from a captive portal.
- micropython-urouter - A lightweight HTTP request routing processing support library based on MicroPython. The previous name was micro-route.

- wlan-relays - Very simple HTTP server written in MicroPython for controlling the pins of an ESP32 board.

WiFi

- HueBridge - Philips Hue Bridge.
- micropython-wifimanager - A simple network configuration utility for MicroPython on the ESP8266 board.
- WiFiManager - WiFi manager for ESP8266 - ESP12 - ESP32 - MicroPython.
- Micropython-ESP-WiFi-Manager - WiFi Manager to configure and connect to networks.
- mpy-wpa_supplicant - MicroPython module to connect to the nearest known Wifi AP.
- micropython-wifi_manager - WiFi Manager for ESP8266 and ESP32 using MicroPython.

Zigbee

- ZbPy - MicroPython IEEE802.15.4 / Zigbee parser.

Display

E-Paper

- micropython-ili9341 - SSD1606 active matrix ePaper display 128x180.
- micropython-waveshare-epaper - Drivers for various Waveshare ePaper modules.
- micropython-waveshare-epd - Waveshare ePaper Display driver for devices running Pycom-flavored MicroPython.
- ssd1675a - Driver for SSD1675-based e-paper displays.
- Inkplate-micropython - MicroPython driver for Inkplate boards.
- micropython-inkplate6 - MicroPython driver for the Inkplate 6.
- eInk-micropython - eInk library for Waveshare 4.3inch device on MicroPython.
- eink - An eInk, ePaper display driver for MicroPython and ESP32.
- micropython_DEPG0213BN - Pure MicroPython driver for the DEPG0213BN eInk display found on the TTGO T5 V2.3 ESP32 boards.
- uPyEINK - Control a Waveshare 7.5" E-INK display using an ESP32 running MicroPython.
- MicroPython-2.9-inch-ePaper-Library - MicroPython Display Driver for WaveShare 2.9inch e-Paper Display (B).

Fonts

- micropython-font-to-py - A Python 3 utility to convert fonts to Python source capable of being frozen as bytecode.
- writer - A simple way to render above Python fonts to displays whose driver is subclassed from `framebuf`.
- ssd1306big - A font for MicroPython on 128x64 pixel SSD1306 OLED display.
- framebuf2 - MicroPython FrameBuffer extension: larger and rotated font, triangles and circles.

- micropython_GT30L24T3Y_big5_font - MicroPython driver for reading BIG-5 Chinese characters from GT30L24T3Y / ER3303-1 SPI module.
- ttgo-hershey-fonts - MicroPython Hershey font demo for the TTGO-LCD board.
- packed-font - Memory efficient MicroPython fonts for the Pico Pi and SSD1306 OLED Display.

Graphics

- micropython-stage - A MicroPython port of the Stage game library.
- micropython-png - Derivative of PyPNG for use with MicroPython.
- mpy-img-decoder - PNG and JPEG decoder / parser / renderer in pure MicroPython.
- micropython-oled-progressbars - A collection of progress bars for use with ESP8266 and ESP32 on OLED displays.
- microplot - Simple MicroPython plotting package.
- micropython-microbmp - A small Python module for BMP image processing.

GUI

- lvgl - An object-oriented, component-based high-level GUI library with MicroPython binding.
- micropython-lcd160cr-gui - Simple touch-driven event based GUI for the Pyboard and LCD160CR colour display.
- micropython_ra8875 - MicroPython device driver and nano-GUI for RA8875 based displays.
- micropython-nano-gui - A tiny display-only GUI with a limited set of GUI objects (widgets) for displays whose display driver is subclassed from the `framebuf` class. With drivers for TFT, ePaper and OLED displays.
- micro-gui - Derived from nano-gui and supporting the same displays and hosts, this provides for user input via push buttons or a navigation joystick and an optional rotary encoder.
- TFT-GUI - A fast touch GUI for large displays based on SSD1963 controller with XPT2046 touch controller.
- micropython-nextion - Control Nextion displays using MicroPython.
- mp_lvgl_widgets - Widgets for the MicroPython Port of LVGL.
- micropython-core2 - Extends LV-MicroPython for the M5Stack CORE2 with MPU6886, ILI9342C, BM8563 and AXP192 drivers.

LCD Character

- Grove_RGB_LCD - Driver for SeeedStudio's Grove RGB LCD.
- lcdi2c - Driver for HD44780-compatible dot matrix LCDs.
- micropython-charlcd - Driver for HD44780-compatible LCDs.
- micropython-i2c-lcd - Driver for I2C 2x16 LCD Screens.
- pyboard-LCD-character-display - Pyboar driver for HDD44780-compatible 1602 LCDs.
- python_lcd - Driver for HD44780-compatible dot matrix LCDs.
- micropython-lcd - Class for controlling the HD44780 from a MicroPython Pyboard.

- HD44780-lcd-upy - MicroPython module for controlling a generic HD44780 LCD.
- LCM1602-14_LCD_Library - driver for AIP31068L 3.3 V I2C and SPI 1602 Serial Character LCDs.
- micropython-i2c-lcd - MicroPython package to control HD44780 LCD displays 1602 and 2004 via I2C.

LCD Graphic

- micropython-lcd-AQM1248A - ESP8266 driver for AQM1248A graphic LCD.
- micropython-pcd8544 - Driver for Nokia 5110 PCD8544 84x48 LCD modules.
- micropython-st7565 - Driver for ST7565 128x64 LCDs.
- micropython-st7920 - Library for simple graphic primitives on ST7920 128x64 monochrome LCD panel using ESP8266 and SPI.
- MicroPython_PCD8544 - ESP8266 driver for Nokia 5110 PCD8544.
- Official LCD160CR - Driver for official MicroPython LCD160CR display with resistive touch sensor.
- micropython-hx1230 - MicroPython library for HX1230 96x68 LCD modules.
- micropython-SHARP_Memory_Display - MicroPython driver for SHARP memory display.

LCD TFT

- micropython-ili9341 - Collection of drivers for TFT displays, ILI9341, SH1106, SSD1606, ST7735.
- micropython-ili934x - SPI driver for ILI934X series based TFT / LCD displays.
- MicroPython-ST7735 - ESP32 version of GuyCarvers's ST7735 TFT LCD driver.
- micropython-st7735 - Driver for ST7735 TFT LCDs.
- MicroPython_ST7735 - Driver for ST7735 128x128 TFT.
- SSD1963-TFT-Library-for-PyBoard-and-RP2040 - SSD1963 TFT Library for Pyboard and Raspberry Pi Pico.
- ST7735 - Driver for ST7735 TFT LCDs.
- micropython-ili9341 - MicroPython ILI9341 display & XPT2046 touch screen driver.
- st7789_mpy - Fast pure-C driver for MicroPython that can handle display modules on ST7789 chip.
- st7789py_mpy - Slow MicroPython driver for 240x240 ST7789 display without CS pin from AliExpress, written in MicroPython.
- micropython-ili9341 - MicroPython Driver for ILI9341 display.
- micropython-ili9341 - ILI9341 TFT driver for MicroPython on ESP32.
- st7789_mpy - Fast MicroPython driver for ST7789 display module written in C.
- st7789py_mpy - Driver for 320x240, 240x240 and 135x240 ST7789 displays written in MicroPython.
- ili9342c_mpy - ILI9342C Fast 'C' Driver for MicroPython (M5Stack Core).
- gc9a01py - GC9A01 Display driver in MicroPython.

- gc9a01_mpy - Fast MicroPython driver for GC9A01 display modules written in C.
- st7735-esp8266-micropython - An ESP8266 MicroPython library for ST7735 160x80, 128x128, 128x160 TFT LCD displays.
- TTGO-ST7789-MicroPython - MicroPython ST7789 display driver for TTGO T-Display ESP32 CP2104 WiFi Bluetooth Module 1.14 Inch LCD.
- st7735_micropython - ST7735 MicroPython drivers for 80x160, 128x128, 128x160 for ESP8266.
- ili934x-micropython - Library for using ILI9341 display drivers with MicroPython.
- micropython-st7735-esp8266 - MicroPython driver for ST7735 TFT displays on the ESP8266.
- st7789s3_esp_lcd - Fast ESP_LCD based MicroPython driver for the TTGO T-Display-S3 st7789 display written in C.
- s3lcd - ESP_LCD based MicroPython driver for ESP32-S3 Devices with ST7789 or compatible displays.
- thmi_py - MicroPython display driver for the LILYGO T-HMI written in Python.
- wt32sc01py - WT32SC01 Plus MicroPython Display Driver.
- st7789s3_mpy - MicroPython display driver for the TTGO T-Display-S3 ST7789 written in C.
- t-display-s3 - MicroPython display driver for the TTGO T-Display-S3 ST7789 written in Python.
- mp-ili9341 - MicroPython Driver for ILI9341 TFT Display.
- lvgl_esp32_gc9a01 - Driver for displays using the GC901 driver for use with LVGL MicroPython.

LED Matrix

- micropython-ht1632c - Driver for HT1632C 32x16 bicolor LED matrix.
- micropython-matrix8x8 - Driver for Adafruit 8x8 LED Matrix display with HT16K33 backpack.
- micropython-max7219 - Driver for MAX7219 8x8 LED matrix modules.
- micropython-wemos-led-matrix-shield - Driver for Wemos D1 Mini Matrix LED shield, using TM1640 chip.
- micropython-wemos-led-matrix - Driver for Wemos D1 Mini Matrix LED shield, using TM1640 chip.
- micropython-max7219 - MicroPython driver for MAX7219 8x8 LED matrix.
- MatrixDisplay - MicroPython module for work with MAX7219 LED matrix 8x8 display.

LED Segment

- LKM1638 - Driver for JY-LKM1638 displays based on TM1638 controller.
- max7219_8digit - Driver for MAX7219 8-digit 7-segment LED modules.
- micropython-max7219 - Driver for MAX7219 8-digit 7-segment LED modules.
- micropython-my9221 - Driver for MY9221 10-segment LED bar graph modules.
- micropython-tm1637 - Driver for TM1637 quad 7-segment LED modules.

- micropython-tm1638 - Driver for TM1638 dual quad 7-segment LED modules with switches.
- micropython-tm1640 - Driver for TM1740 8x8 LED matrix modules.
- micropython-tm1640 - MicroPython Library for 16 digits 7-segment displays controlled by a TM1640.
- TM74HC595 - Driver for shift register-controlled 5 pin display modules.
- micropython-tm1638spi - MicroPython Library for a popular board with 8 7-segment digits, 8 separate LEDs and 8 push buttons controlled by a TM1638.

LEDs

- micropython-morsecode - Blink an LED with Morse Coded message.
- micropython-p9813 - Driver for P9813 RGB LED used in SeeedStudio's Grove chainable RGB LED.
- micropython-ws2812-7seg - 7-segment display using WS2812 RGB LEDs.
- micropython-ws2812 - Driver for WS2812 RGB LEDs.
- Official APA102 - ESP8266 APA102/DotStar RGB LED driver.
- Official WS2811 - ESP8266 WS2811/NeoPixel RGB LED driver.
- tlc5940-micropython - Driver for TLC5940 16 channel LED driver.
- ws2812-SPI - An efficient MicroPython WS2812 (NeoPixel) driver.
- micropython-ws2801 - A MicroPython library to interface with strands of WS2801 RGB LEDs.
- tlc5947-rgb-micropython - Driver for the TLC5947 24 channel 12-bit PWM LED driver.
- Hybotics_Micropython_HT16K33 - MicroPython driver for the HT16K33, a LED matrix, 7-Segment Numeric, and 14-Segment Alphanumeric display driver IC.
- micropython-rgbled - This wrapper module aims to reduce the work needed to work with NeoPixel (WS2812) and DotStar (APA102) RGB LED strips and matrixes.
- micropython_fastled - Port of FastLED to MicroPython.
- micropython-rgb-led-driver - Tiny driver to control an RGB LED with PWM.
- micropython-dotstar - A MicroPython port of the Adafruit CircuitPython APA102/DotStar library.

OLED

- Grove_OLED - Driver for SSD1327 used by SeeedStudio's Grove OLED Display 1.12" v1.0.
- micropython-oled - Collection of drivers for monochrome OLED displays, PCD8544, SH1106, SSD1306, UC1701X.
- micropython-ssd1327 - Driver for SSD1327 128x128 4-bit greyscale OLED displays.
- micropython-ssd1351 - Driver for SSD1351 OLED displays.
- MicroPython_SSD1306 - ESP8266 driver for SSD1306 OLED 128x64 displays.
- Official SSD1306 - Driver for SSD1306 128x64 OLED displays.
- SH1106 - Driver for the SH1106 OLED display.
- micropython-ssd1309 - MicroPython SSD1309 Monochrome OLED Display Driver.
- sh1107-micropython - MicroPython driver for SH1107-based OLED display (64x128).

- SH1107 - Driver for SH1107 OLED displays (128x128 and 128x64 pixels).
- micropython-ssd1322 - MicroPython display driver for SSD1322 grayscale OLED.

Printer

- micropython-thermal-printer - The MicroPython port of Python Thermal Printer by Adafruit.

IO

ADC

- ads1x15 - Driver for the ADS1015/ADS1115 ADC, I2C interface.
- micropython-ads1015 - ADS1015 12-Bit and ADS1115 16-bit ADC, 4 channels with programmable gain, I2C interface.
- Micropython_ADS1115 - ADS1115 16-bit ADC, 4 channels with programmable gain, I2C interface.
- ADS7818 - Python class interfacing the ADS7818 AD-converter.
- micropython-ads1219 - MicroPython module for the Texas Instruments ADS1219 ADC.
- micropython-hx711 - MicroPython driver for HX711 24-Bit Analog-to-Digital Converter.
- MicroPython-ADC_Cal - ESP32 ADC driver using reference voltage calibration value from efuse.
- micropython-pcf8591 - MicroPython driver for PCF8591 ADC/DAC, I2C interface.
- hx711_mpy-driver - Micropython Driver for the HX711 weighing sensor.
- MCP342x_LoPy - MicroPython driver for the MCP342x ADC.
- micropython-ads1220 - MicroPython library for ADS1220 24-bit analog-to-digital converter.
- PCF8591_micropython_library - MicroPython library for PCF8591 8-bit ADC/DAC.

DAC

- micropython-mcp4725 - Driver for the MCP4725 I2C DAC.
- mcp4728 - Helper library for the Microchip MCP4728 I2C 12-bit Quad DAC.

GPIO

- micropython-inputs - Classes to count pulses, debounce digital inputs, and calculate moving averages of analog inputs for a MicroPython board.
- ubutton - A MicroPython library for controlling reading and debouncing pushbutton inputs, including "short" and "long" press callbacks.
- micropython-debounce-switch - MicroPython Class for Debouncing Switches.

IO-Expander

- micropython-mcp230xx - Driver for MCP23017 and MCP23008 GPIO expanders.
- micropython-mcp23017 - MicroPython driver for MCP23017 16-bit I/O Expander.

- micropython-pcf8574 - MicroPython driver for PCF8574 8-Bit I2C I/O Expander with Interrupt.
- micropython-pcf8575 - MicroPython driver for PCF8575 16-Bit I2C I/O Expander with Interrupt.
- ESP8266_MCP23S17 - MicroPython library for using the MCP23S17 16-bit I/O expander with the ESP8266.
- pcf8574 - MicroPython module for working with the PCF8574(A) I2C 8-bit I/O expander from NXP.

Joystick

- micropython-nunchuck - Driver for Nunchuk game controller, I2C interface.
- esp32-microgamepad-ble - Dual analog joystick on ESP32 over BLE (Nordic UART Service - NUS) using MicroPython.

Keyboard

- micropython-keyboard - 47 key keyboard running on a MicroPython Pyboard.
- pico-rgbkeypad - A Python class for controlling the Pimoroni RGB Keypad for Raspberry Pi Pico.
- micropython-aiobutton - A MicroPython module for asyncio button.

Potentiometers

- micropython-ad840x - MicroPython SPI-based manipulation of the AD series digital potentiometers AD8400, AD8402 and AD8403.
- mcp4131 - MicroPython module to control MicroChip's MCP4131 SPI digital potentiometer.
- MicroPython_DS1841 - MicroPython Driver for the DS1841 Potentiometer.
- MicroPython_DS3502 - MicroPython Driver for the DS3502 Potentiometer.

Power Management

- AXP202_PythonLibrary - MicroPython AXP202 Library.
- micropython_hourly_sleeper_library - A MicroPython library that enables an ESP8266 to sleep for hourly increments for a setup amount of hours.

PWM

- upwmcontroller - A MicroPython library for controlling PWM outputs in an asyncio loop, with features including fading and blinking.

Rotary Encoder

- micropython-rotary - MicroPython module to read a rotary encoder.
- uencoder - A MicroPython library for reading from a rotary encoder.
- encodermenu - Simple GUI menu for MicroPython using a rotary encoder and basic display.

- encoderLib - MicroPython library to handle a rotary encoder.
- rotary-encoder - MicroPython code to drive a KY-040 rotary encoder.
- micropython-encoder-knob - A very simple lightweight encoder knob library with button support.
- encoders - Short document explaining issues around encoder technology.
- asynchronous encoder driver - Interface an encoder to uasyncio code.
- micropython-8encoder - Driver for the I2C M5Stack 8-Encoder Unit
- micropython-quiic-twist - MicroPython Driver for Quiic Twist RGB Rotary Encoder.
- AS5600 - AS5600 MicroPython library for reading this magnetic sensor.

Shift Registers

- micropython-74hc595 - MicroPython driver for 74HC595 8-bit shift registers.
- MicroPython-SN74HCS264 - MicroPython Driver for SN74HCS264 8-Bit Parallel-Out Serial Shift Registers With Schmitt-Trigger Inputs and Inverted Outputs.

Waveform Generator

- Micropython-AD9833 - Pyboard driver for AD9833, SPI interface.
- Clock_Generators - Clock generators (Si5351 for now) toolbox.
- Signal_Generators - Signal generators (AD9833, AD9834, AD9850, ADF4351) toolbox.
- ad9850_signalgen - MicroPython library for AD9850 synthesizer.
- pico-wave-vibration-generator - A MicroPython-based frequency generator for Raspberry Pi Pico designed to create vibrations on solenoids or speakers, enabling wave experimentation and exploration at home.
- micropython-m5stack-dds - MicroPython driver for the M5Stack DDS frequency generator.
- AD9833-MicroPython-Module - MicroPython module to use the AD9833 programable waveform generator.

Mathematics

- uMath - Computer Algebra for microcontrollers.
- micropython-ulab - A NumPy-like fast vector module for MicroPython.
- micropython-fourier - Fast Fourier transform in MicroPython's inline ARM assembler.
- Filters - FIR filters using ARM Thumb assembler. Using an online utility you can go from a graph of required frequency response to a filter implementation.
- ulinalg - Small size matrix handling module with a few linear algebra operations specifically for MicroPython (Python 3).
- micropython-mtx - Fast Matrix Multiplication and Linear Solver on MicroPython.
- micropython-vec - Vector Operations on MicroPython.
- MicroPython_Statistics - Statistics module for MicroPython.
- MicroPython-Matrix - MicroPython basic matrix operations.
- uumpy - A subset of NumPy for MicroPython.

- upyuncertainties - Uncertainty calculations for MicroPython.
- umatrix - A matrix library for the MicroPython language.
- micropython-fractions - A MicroPython port of the CPython standard Fractions library.

Motion

DC Motor

- L298N - Driver for the L298N dual H-bridge motor controller.
- MicroPython-L298 - Drive L298 dual H-bridge with MicroPython.
- pyl298 - Driver for the L298 dual full-bridge motor controller.

Servo

- micropython-pca9685 - 16-channel 12-bit PWM/servo driver.
- micropython-servo - Library to control RC servos using direct PWM output in a tidy way.
- MicroPython_PCA9685 - MicroPython Driver for the PCA9685 PWM control IC, commonly used to control servos, LEDs and motors.
- MicroPython_MOTOR - MicroPython Helper for controlling PWM based motors.

Stepper

- micropython-stepper - Library to control common stepper drivers in a tidy way.
- micropython-upybbot - A4988 driver for bipolar stepper motors.
- uln2003 - Driver for 5V 28BYJ-48 stepper motors.
- micropython-multiaxis - Multiaxis with MicroPython ESP32 and DRV8825.
- ticlib - Driver for Pololu Tic stepper motor controllers.
- AccelStepper-MicroPython - AccelStepper Library for MicroPython - ESP32.
- pystepper - MicroPython Stepper Motor Sequence Control.
- uPySteppers - DIY rotating platform using an ESP32 connected to WiFi.
- microPython_AMIS-30543 - MicroPython library for Stepper Driver control using AMIS-30543 driver.
- micropython-drv8825 - Driver and example in MicroPython to control a stepper motor via a DRV8825 controller board.
- microPython_TMC5160 - A MicroPython library for the Trinamic TMC5160 Motion Controller.
- micropython-stepper-motor - Drive a 28BYJ-48 motor attached to a ULN2003 driver.

Sensors

Accelerometer Digital

- ADXL345-with-Pyboard - Driver for ADXL345 16g 3-axis accelerometer.
- adxl345_micropython - Driver for ADXL345 16g 3-axis accelerometer.

- MicroPython-LIS3DH - I2C driver for LIS3DH 3-axis accelerometer.
- micropython-lis2hh12 - I2C driver for LIS2HH12 3-axis accelerometer.
- MMA7660 - Driver for MMA7660 1.5g 3-axis accelerometer.
- ADXL345_spi_micropython - Library for interacting through the SPI protocol with an 'Analog Devices ADXL345' accelerometer from an MCU flashed with MicroPython.
- MicroPython_ADXL343 - MicroPython Driver for the Analog Devices ADXL343 Accelerometer.
- MicroPython_BMA220 - MicroPython Driver for the Bosch BMA220 Accelerometer.
- MicroPython_BMA400 - MicroPython Driver for the Bosch BMA400 Accelerometer.
- MicroPython_LIS3DH - MicroPython Driver for the LIS3DH 3-axis accelerometer.
- MicroPython_KX132 - MicroPython Driver for the Kionix KX132 Accelerometer.
- MicroPython_H3LIS200DL - MicroPython Driver for the ST H3LIS200DL Accelerometer.
- MicroPython_QMC5883L - MicroPython Driver for the QMC5883L Accelerometer.
- Micropython_MC3479 - MicroPython Driver for the MC3479 Accelerometer.
- MicroPython_MMA8451 - MicroPython module for the MMA8451 3-axis accelerometer.
- MicroPython_MMA8452Q - MicroPython Driver for the NXP MMA8452Q Accelerometer.
- msa301-micropython-driver - Homebrew MicroPython driver for MSA301 3-axis accelerometer. Tested on Raspberry Pico.

Air Quality

- CCS811 - CCS811 Air Quality Sensor.
- upython-aq-monitor - Air Quality monitor using PMS5003 sensor and WiPy.
- micropython-pms7003 - MicroPython driver for the PMS7003 Air Quality Sensor.
- pms5003_micropython - Driver for PMS5003 air quality sensor for MicroPython.
- micropython-pms5003-minimal - Driver for P air quality sensor for MicroPython.
- polly - SDS011 pollution sensor + Wemos D1 mini pro + MicroPython.

Barometer

- micropython-bme280 - Driver for the Bosch BME280 temperature/pressure/humidity sensor.
- micropython-bmp180 - Driver for Bosch BMP180 temperature, pressure and altitude sensor.
- mpy_bme280_esp8266 - Bosch BME280 temperature/pressure/humidity sensor.
- BME280 - MicroPython driver for the BME280 sensor, target platform Pycom devices.
- micropython-bmp280 - Module for the BMP280 sensor.
- micropython_bme280_i2c - A MicroPython module for communicating with the Bosch BME280 temperature, humidity, and pressure sensor.
- MicroPython-BME280 - Driver to digital sensor of Temperature, Pressure and Humidity.
- micropython-bmp180 - A module for MicroPython which provides a class for the BMP180 pressure sensor.
- BMP390 - MicroPython module for BMP390 pressure & temperature sensor.
- BMP180 - MicroPython module for BMP180 pressure & temperature sensor.

- MicroPython_BMP581 - MicroPython driver for the Bosch BMP581 pressure & temperature sensor.
- MicroPython_DPS310 - MicroPython Driver for the DPS310 Sensor.
- MicroPython_ICP10111 - MicroPython Driver for the TDK ICP-10111 Barometric Pressure and Temperature sensor.

Battery

- Micropython-LC709203F - A simple MicroPython library for LC709293F Fuel Gauge.

Biometric

- micropython-fingerprint - MicroPython library for reading Grow and ZhianTec fingerprint sensors.
- MAX30102-MicroPython-driver - A MAX30102 driver ported to MicroPython. It should also work for MAX30105.

Camera

- micropython-ov2640 - MicroPython class for OV2640 camera.
- Nikon-Trigger-for-MicroPython - Remote trigger for a Nikon camera using an IR LED. For Pyboard v1.1.
- micropython-camera-driver - OV2640 camera driver for MicroPython on ESP32.
- esp32-cam-micropython - MicroPython ESP32-CAM.
- uPyCam - Take a photo with an ESP32-CAM running MicroPython.
- OV2640_uPy - OV2640 camera library for MicroPython.
- MQTT-Cam - ESP32-CAM MicroPython MQTT AWS S3 Uploader.

Colour

- micropython-tcs34725 - Driver class for TCS34725 and TCS34727 color sensors.
- micropython-as7341 - MicroPython library for AS7341.
- MicroPython_ISL29125 - MicroPython Driver for the Intersil ISL29125 Color Sensor.
- TCS3200-MicroPython - A MicroPython driver and test programs for the TCS3200 color sensor.
- MicroPython_TCS3430 - MicroPython driver for the AMS TCS3430 Color and ALS sensor.

Compass

- micropython-esp8266-hmc5883l - 3-axis digital compass on the ESP8266.
- QMC5883 - Python class for the QMC5883 Three-Axis Digital Compass IC.
- microPython_AS5600L - MicroPython driver for AS5600L magnet rotary position sensor.

Current

- micropythonINA219 - Driver for INA219 current sensor.

- pyb_ina219 - Driver for INA219 current sensor.
- INA219 - INA219 MicroPython driver.
- TI_INA226_micropython - MicroPython driver for Texas Instruments INA226 power measuring IC.
- micropython-current-monitor - Current monitor using the INA219 and an SSD1306 OLED.

Distance IR

- micropython-gp2y0e03 - IR-LED distance measuring sensor using Sharp GP2Y0E03.
- micropython-vl6180 - Time-of-Flight sensor, ambient light sensor & IR emitter.

Distance Laser

- micropython-vl53l0x - Time-of-Flight laser-ranging sensor.
- Qwiic_TOF_Module_RFD77402 - Qwiic TOF Module (RFD77402) time-of-flight rangefinding module.
- VL53L0X - MicroPython Library for LiDAR Sensor VL53L0X.
- vl53l1x_pico - MicroPython driver for the VL53L1X ToF sensor.
- tf-luna-micropython - A simple MicroPython I2C library for TF-Luna LiDAR Module.
- vl53l5cx - MicroPython and CircuitPython Package for the VL53L5CX (4x4/8x8 ToF sensor array).
- VL6180X - MicroPython driver for the VL6180X sensor on the ESP32.

Distance Ultrasonic

- micropython-hcsr04 - Driver for HC-SR04 ultrasonic distance sensors.
- micropython-us100 - MicroPython driver for the US-100 sonar distance sensor.
- micropython-i2c-ultrasonic - MicroPython driver for the RCWL-9620-based M5 I2C Ultrasonic Distance Unit.

Dust

- pyGP2Y - MicroPython library for the Sharp GP2Y1014AU0F Dust Sensor.

Energy

- ATM90E26_Micropython - Driver for ATM90E26 energy metering device.
- MCP39F521 - ESP8266 scripts for reading MCP39F521 power monitors.
- micropython-p1meter - A ESP32 sensor to read a p1 electricity meter and publish this to MQTT and Home Assistant, written in MicroPython.
- esp32-solar2 - Simple solar regulator - MicroPython project.
- cs5490_micropython - MicroPython Driver for CS5490 Energy Monitor IC.

Gaseous

- micropython-MQ - Drivers for MQ series gas sensors.

- MQ135 - Driver for MQ135 gas sensor.
- CCS811 - Basic MicroPython driver for CCS811 on ESP8266 boards.
- micropython-scd30 - MicroPython I2C driver for Sensirion SCD30 CO2 sensor module.
- MicroPython_SCD4X - MicroPython I2C driver for Sensirion SCD40 and SCD41 CO2 sensors.
- micropython-sgp40 - MicroPython I2C driver for SGP40 VOC sensor module.
- MICS6814-Micropython-driver - ESP32 MicroPython driver for the Pimoroni MICS6814 breakout board.
- MicroPython_AGS02MA - MicroPython Driver for the AGS02MA TVOC sensor.
- SCD4x - MicroPython module for work with SCD4x (SCD40, SCD41) low power CO2, temperature & humidity electroacoustic sensor from Sensirion.
- ens160 - MicroPython module for work with ENS160 Digital Metal-Oxide Multi-Gas Sensor.

Humidity

- MicroPython_HTS221 - MicroPython Driver for the HTS221 Humidity Sensor.

Light

- MicroPython-SI1145 - SI1145 UV index, IR, visible light and proximity sensor.
- micropython-tsl2561 - Driver for the TSL2561 illumination sensor from TAOS / ams.
- mpy_bh1750fvi_esp8266 - ESP8266 driver for BH1750FVI sensor.
- bh1750 - BH1750 I2C digital light sensor driver.
- micropython-max44009 - MicroPython driver for the MAX44009 ambient light sensor.
- veml7700 - Library for MicroPython for VEML7700 light sensor.
- MicroPython_MAX44009_driver - MicroPython driver for MAX44009 light sensor.
- MicroPython-VEML6075 - Driver base for the VEML6075 UV light sensor.
- BH1750 - MicroPython module for the BH1750 ambient light sensor (ALS).
- veml7700 - MicroPython module for the VEML7700 ambient light sensor (ALS) from Vishay.

Magnetometer

- MicroPython_LIS2MDL - MicroPython Driver for the ST LIS2MDL Magnetometer sensor.
- MicroPython_LIS3MDL - MicroPython Driver for the ST LIS3MDL magnetometer.
- MicroPython_MLX90393 - MicroPython Driver for the MLX90393 Magnetometer.
- MicroPython_MMC5603 - MicroPython driver for the Memsic MMC5603 Magnetometer.
- MicroPython_BMM150 - MicroPython Driver for the Bosch BMM150 Magnetometer.
- MicroPython_MMC5983 - MicroPython Library for the Memsic MMC5983 Magnetometer.

Motion Inertial

- micropython-bmx055 - Driver for Bosch BMX055 IMU sensor.
- micropython-bno055 - Bosch Sensortec BNO055 9DOF IMU sensor, I2C interface.
- micropython-lsm9ds0 - LSM9DS0 g-force linear acceleration, Gauss magnetic and DPS angular rate sensors.

- micropython-mpu9250 - I2C driver for MPU9250 9-axis motion tracking device.
- micropython-mpu9x50 - Driver for the InvenSense MPU9250 inertial measurement unit.
- MPU6050-ESP8266-MicroPython - ESP8266 driver for MPU6050 accelerometer/gyroscope.
- py-mpu6050 - ESP8266 driver for MPU6050 accelerometer/gyroscope.
- micropython-mpu6886 - MicroPython I2C driver for MPU6886 6-axis motion tracking device.
- micropython-fusion - Sensor fusion calculates heading, pitch and roll from the outputs of motion tracking devices.
- flight_controller - MicroPython flight controller.
- micropython-bno055 - Bosch BNO055 driver for MicroPython. IMU with hardware sensor fusion.
- micropython-mpu6050-mqtt-streamer - Stream data from MPU6050 to MQTT server using MicroPython on ESP8266.
- upy-motion - A simple MPU6050 driver written in MicroPython.
- micropython-bno08x-rvc - MicroPython library for BNO08x.
- micropython-mpu9250 - MicroPython MPU-9250 (MPU-6500 + AK8963) I2C driver.
- MicroPython_ICM20948 - MicroPython Driver for the TDK ICM20948 Accelerometer/Gyro Sensor.
- MicroPython_BMI160 - MicroPython Driver for the Bosch BMI160 Accelerometer/Gyro Sensor.
- micropython-mpu6050 - MicroPython library for reading from MPU-6050 accelerometer and gyroscope modules.
- MicroPython_ICG20660 - MicroPython Driver for the TDK ICG20660 Accelerometer/Gyro sensor.
- MicroPython_BMI270 - MicroPython Driver for the Bosch BMI270 Accelerometer/Gyro Sensor.
- MicroPython_LSM6DSOX - MicroPython Library for the ST LSM6DSOX accelerometer/gyro Sensor.

Pressure

- ms5803-micropython - A MicroPython implementation of the driver for an MS5803 pressure & temperature sensor.
- MPL3115A2_MicroPython - MicroPython library for the MPL3115A2 altimeter.
- MicroPython_MMR902 - MicroPython Driver for the Mitsumi MMR902 Micro Pressure Sensor.
- MicroPython_MPL3115A2 - MicroPython driver for the NXP MPL3115A2 Pressure and Temperature sensor.
- MicroPython_MS5611 - MicroPython Driver for the TE MS5611 Pressure and Temperature Sensor.
- D6F-PH - MicroPython module for differential pressure sensor, D6F-PH (OMRON).

Proximity

- uPy_APDS9960 - MicroPython proximity library for ESP8266 using APDS9960.
- MicroPython_VCNL4010 - MicroPython Driver for the Vishay VCNL4010 Proximity and Ambient Light Sensor.

Radiation

- micropython-geiger - Geiger counter with MicroPython card.
- ESPGeiger - MicroPython library for the ESP8266 Geiger counter.

Soil Moisture

- micropython-chirp - Driver for the Chirp Soil Moisture Sensor.
- MicroPython-MiFlora - Xiaomi Mi Flora (aka flower care) BLE plant sensors (soil moisture/conductivity/light intensity/temperature).
- micropython-miflora - MicroPython library for Xiaomi Mi Flora BLE plant sensors.

Spectral

- AS726X_LoPy - MicroPython driver for the AS726X spectral sensor.
- MicroPython_AS7262X_driver - MicroPython driver for AS7262/AS7263 nano spectrometer sensor.

Temperature Analog

- micropython-max31855 - Thermocouple amplifier, SPI interface.
- max31856 - Precision thermocouple to digital converter with linearization, SPI interface.
- mcp9700 - Generic MicroPython driver for MCP9700.

Temperature Digital

- bme680-mqtt-micropython - Driver for BME680 gas, pressure, temperature and humidity sensor.
- LM75-MicroPython - Driver for LM75 digital temperature sensor, I2C interface.
- micropython-am2320 - Aosong AM2320 temperature and humidity sensor, I2C interface.
- micropython-dht12 - Aosong DHT12 temperature and humidity sensor, I2C interface.
- micropython-hdc1008 - Driver for the Texas Instruments HDC1008 humidity and temperature sensor.
- micropython-mcp9808 - Driver for the Microchip MCP9808 temperature sensor.
- micropython-mpl115a2 - Pyboard driver for the MPL115A2 barometric pressure sensor.
- micropython-sht30 - Driver for SHT30 temperature and humidity sensor.
- micropython-sht31 - Driver for the SHT31 temperature and humidity sensor.
- micropython-Si7005 - Driver for Si7005 relative humidity and temperature sensor.
- micropython-si7021 - SI7021 Temperature and humidity sensor, I2C interface.
- micropython-si7021 - SI7021 Temperature and humidity sensor, I2C interface.

- micropython-Si705x - Silicon Labs Si705x series of temperature sensors, I2C interface.
- micropython-Si70xx - Silicon Labs Si70xx series of relative humidity and temperature sensors, I2C interface.
- micropython-tmp102 - Driver for TMP102 digital temperature sensor.
- Official DHT11+DHT12 - ESP8266 driver for DHT11 and DHT12 temperature and humidity sensor.
- sht25-micropython - Driver for SHT25 temperature and humidity sensor.
- micropython-tmp1075 - Driver for the TI TMP1075 temperature sensor.
- micropython-sht11 - Driver for Sensirion SHT11 temperature and humidity sensor.
- micropython-lm75a - Driver for the NXP LM75A digital temperature sensor.
- BME680-Micropython - MicroPython driver for the BME680 sensor.
- htu21d-esp8266 - This is a MicroPython module / class to measure data from the HTU21D.
- HTU21D - Asynchronous driver for HTU21D temperature and humidity sensor.
- esp-sht3x-micropython - A SHT3x (SHT30/31/35) library for ESP8266/ESP32 with MicroPython.
- sht25-micropython - MicroPython implementation of API of SHT25 humidity and temperature sensor.
- micropython-sht30 - SHT30 sensor driver in pure Python based on I2C bus.
- micropython_ahtx0 - MicroPython driver for the AHT10 and AHT20 temperature and humidity sensors.
- sht85 - MicroPython driver for the Sensiron SHT85 humidity and temperature sensor.
- micropython-zacwire - MicroPython driver for the ZACwire protocol used in TSic 506F temperature sensors.
- MicroPython_HTU31D - MicroPython library for TE HTU31D temperature and humidity sensors.
- MicroPython_SHTC3 - MicroPython Driver for the Sensirion SHTC3 Temperature and Humidity Sensor.
- MicroPython_TMP117 - MicroPython Driver for the TMP117 Temperature Sensor.
- MicroPython_SI7021 - MicroPython Library for the Temperature and Humidity SI7021 Sensor.
- MicroPython_ADT7410 - MicroPython Driver for the Analog Devices ADT7410 Temperature Sensor.
- MicroPython_WSENTIDS - MicroPython library for the WSEN WSEN-TIDS temperature Sensor.
- MicroPython_HS3003 - MicroPython Driver for the Renesas HS3003 Temperature and Humidity Sensor.
- MicroPython_STTS22H - MicroPython Driver for the STTS22H Temperature Sensor.
- MicroPython_HTU21DF - MicroPython HTU21D-F Temperature & Humidity driver.
- MicroPython_SHT4X - MicroPython Driver fot the Sensirion Temperature and Humidity SHT40 and SHT45 Sensor.
- MicroPython_SHT20 - MicroPython Driver for the Sensirion SHT20 Temperature Sensor.

- MicroPython_MCP9808 - MicroPython Driver for the Microchip MCP9808 Temperature Sensor.
- MicroPython_HDC1080 - MicroPython driver for the TI HDC1080 Temperature and Humidity sensor.
- TMP117 - MicroPython module for the TMP117 temperature sensor from Texas Instruments.
- BME680 - MicroPython module for the BME680, Bosch low power gas, pressure, temperature & humidity sensor.
- SHT30 - MicroPython driver for the Sensirion SHT3x sensor.
- MicroPython_AS6212 - MicroPython Library for the ASM AS6212 Temperature Sensor.
- MicroPython_PCT2075 - MicroPython Driver for the NXP Semiconductors PCT2075 Temperature Sensor.

Temperature IR

- micropython-mlx90614 - Driver for Melexis MLX90614 IR temperature sensor.
- MicroPython_MLX90615_driver - MicroPython driver for Melexis MLX90615 IR temperature sensor.

Touch Capacitive

- micropython-mpr121 - Driver for MPR121 capacitive touch keypads and breakout boards.
- micropython-ttp223 - Examples using TTP223 capacitive touch module.
- micropython-TTP229-BSF - MicroPython ESP8266/ESP32 driver for TTP229-BSF 16-key capacitive keypad in serial interface mode.
- uFT6336U - MicroPython I2C driver for the Focus LCDs FT6336U capacitive touch panel controller IC.
- MicroPythonTrill - Trill touch sensor library for MicroPython.
- L58Touch - L58 Multi-Touch MicroPython Module.

Touch Resistive

- XPT2046-touch-pad-driver - Driver for XPT2046 touch pad controller used in many TFT modules.

Scheduling

- micropython-mcron - MicroCRON is a time-based task scheduling program for MicroPython.
- micropython-scron - SimpleCRON is a time-based task scheduling program inspired by the well-known cron program for Unix systems.
- Schedule - A scheduler for uasyncio based applications. Schedule events at specified times and dates.
- micropython-aioschedule - A persistent uasyncio scheduler that supports deepsleep between task runs.

Storage

Database

- uPyMySQL - Pure MicroPython MySQL Client.
- micropython-redis - A Redis client implementation designed for use with MicroPython.
- picoredis - A very minimal Redis client (not only) for MicroPython.
- micropg - PostgreSQL database driver for MicroPython.
- micropg_lite - PostgreSQL database driver for MicroPython, based on micropg but aiming to require less memory with some compromises in functionality.
- nmongo - MongoDB client for CPython and MicroPython, with MongoDB shell-like APIs.
- MicroPyDatabase - A low-memory JSON-based database for MicroPython.
- micropython-firebase-realtime-database - Firebase implementation for MicroPython optimized for ESP32.
- micropython-firebase-firestore - Firebase Firestore implementation for MicroPython.
- uSQLite - SQLite library module for MicroPython.

EEPROM

- micropython_eeprom - Cross-platform MicroPython device drivers for memory chips (EEPROM, FRAM, Flash, PSRAM).
- mb_24x256_512 - Very simple MicroPython module/driver for Microchip 24x256 and 24x512 I2C EEPROM devices.
- micropython-eeprom - MicroPython driver for AT24Cxx EEPROM.

Flash

- micropython_data_to_py - A Python 3 utility to convert an arbitrary binary file to Python source for freezing as bytecode in Flash.
- micropython-winbond - Interact with Winbond W25Q Flash chips via SPI.

FRAM

- micropython-fram - Pyboard driver for Ferroelectric RAM module.

PSRAM

- mb_PSRAM_64Mb_SPI - Very simple MicroPython module to use a generic 64Mbit PSRAM (ie Adafruit 4677) with a Raspberry Pi Pico (RP2040).

SRAM

- mb_23LC1024 - Very simple MicroPython module to use a Microchip 23LC1024 SPI SRAM with a Raspberry Pi Pico (RP2040).
- mb_47x16 - Very simple MicroPython module/driver for Microchip 47x16 EERAM devices (47L/47C).

Threading

- MicroWorkers - A micro workers class that easily manages a pool of threads to optimise simultaneous jobs and jobs endings, for MicroPython (used on Pycom modules & ESP32).

User Interface

- upymenu - MicroPython Menu for LCD Displays.

Community

- MicroPython Discussions on GitHub - GitHub discussions for all things related to MicroPython.
- MicroPython Forum (archive) - Archived community conversations on all things related to MicroPython.
- Discord - Get an invite to the MicroPython Discord server.
- MicroPython on Mastodon / Fediverse - Follow MicroPython in the Fediverse.
- MicroPython on Twitter - Follow MicroPython on Twitter for latest news and updates.
- MicroPython on Facebook - Like MicroPython on Facebook for competitions, news and updates.
- Melbourne MicroPython Meetup - Regular meetup at CCHS in Melbourne, Australia.

Tutorials

- uasyncio - Write asynchronous code which interfaces to hardware devices.
- Asynchronous drivers - Tutorial and code for asynchronous interfaces to switches, pushbuttons, encoders and ADCs.
- Pyboard micropower - Tutorial and code for low power applications on Pyboard 1.x and Pyboard D.
- 3D rotation with quaternions - Tutorial and code for the easy way to do 3D rotation.
- Miguel Grinberg - MicroPython and the Internet of Things.
- Bhavesh Kakwani - MicroPython videos + written tutorials.
- CoderDojo Twin Cities MicroPython - Full coding curriculum for teaching MicroPython to children.
- MicroPython Tutorials for ESP32 boards - Tutorials with code examples to learn the basic of MicroPython with ESP32 boards.
- Learn MicroPython with a Pi Pico board - Tutorials on MicroPython with the Raspberry Pi Pico / RP240 boards.

Books

- Programming with MicroPython: Embedded Programming with Microcontrollers and Python - By Nicholas H. Tollervey. ISBN 9781491972731.

- MicroPython for the Internet of Things: A Beginner's Guide to Programming with Python on Microcontrollers - By Charles Bell. ISBN 9781484231227.
- Beginning MicroPython with the Raspberry Pi Pico: Build Electronics and IoT Projects - By Charles Bell. ISBN 9781484281345.
- MicroPython Cookbook - By Marwan Alsabbagh. ISBN 9781838649951.
- Python for Microcontrollers: Getting Started with MicroPython - By Donald Norris. ISBN 9781259644535.
- Advanced Programming in MicroPython By Example - By Yury Magda. ISBN 9781090900937.
- MicroPython Projects - By Jacob Beningo. ISBN 9781789958034.
- Get Started with MicroPython on Raspberry Pi Pico - By Gareth Halfacree and Ben Everard. ISBN 9781912047864.
- MicroPython for Microcontrollers - By Günter Spanner. ISBN 9783895764370.
- MicroPython for the Raspberry Pi Pico W: A gentle introduction to programming digital circuits with Python - By Miguel Grinberg. ISBN 9798361302710.

Frameworks

- micrOS - MicroPython-based IoT Framework.
- terkin-datalogger - Flexible data logger application for MicroPython and CPython.
- perthensis - Perthensis: an asynchronous framework for MicroPython.
- meerkat - I2C Data Acquisition for MicroPython and Raspberry Pi.

Resources

- MicroPython - Project website. Test drive the Pyboard. Try MicroPython online with Unicorn.
- MicroPython on GitHub - Submit bug reports, follow and join in development on GitHub.
- MicroPython Official Documentation - For various ports, including quick reference, general information, examples and tutorials.
- MicroPython Wiki - Community generated documentation and examples of the features of MicroPython and the Pyboard.
- MicroPython Newsletter - Subscribe to the MicroPython newsletter for news and announcements including new features and new products.
- MicroPython Store - Where you can buy the Pyboard, housings, skins, books, connectors and peripherals.
- MicroPython on Wikipedia - MicroPython on Wikipedia.
- awesome-micropythons - The many forks & ports of MicroPython.

Development

Code Generation

- micropy-cli - Micropy CLI is a project management/generation tool for writing MicroPython code in modern IDEs such as Visual Studio Code.
- micropython-stubber - Generate and use stubs for different MicroPython firmwares to use with Visual Studio Code or any IDE and linter.
- micropython-stubs - Stubs of most MicroPython ports, boards and versions to make writing code that much simpler.
- micropy-stubs - Automatically Generated Stub Packages for Micropy-Cli and whomever else.
- micropython-extmod-generator - Generator for MicroPython external modules written in C.
- micropython-package-template - GitHub workflow supported MicroPython package template with deploys to the Python Package Index on a push to the main branch and test deploys to the Test Python Package Index on PRs.
- micropython-usermod - Online book about MicroPython external modules writen in C.

Debugging

- esp32-backtrace - ESP32 Exception Stack Backtrace Analyzer.
- micropython-aiosentry - Asynchronous Sentry.io micro client for MicroPython.
- micropython-usyslog - Simple remote syslog client for MicroPython.
- Asynchronous monitor - Use a Raspberry Pico and a logic analyser or scope to monitor asynchronous code.

IDEs

- BIPES - Web-based IDE for MicroPython with file manager, editor, code generation from blocks, IoT dashboard and Serial/USB/Bluetooth/WebREPL console on the web browser. Source: https://github.com/BIPES.
- ESP32-MPY-Jama - Tool for managing Espressif ESP32 microcontrollers with MicroPython.
- JetBrains IntelliJ/PyCharm MicroPython Plugin - Plugin for MicroPython devices in IntelliJ and PyCharm.
- MicroPython IDE for VSCode - MicroPython IDE for Visual Studio Code.
- MicroPython-REPLink for VSCode - Handy shortcuts for interacting with a MicroPython REPL terminal.
- MPRemote for VSCode - An extension to provide easy access to some of mpremote's functionality from within Visual Studio Code.
- Mu Editor - Code with Mu: a simple Python/MicroPython/CircuitPython editor for beginner programmers.
- Thonny IDE - Thonny: Python IDE for beginners.
- Pyboard File Manager - Pyboard File Manager: Windows GUI for Pyboard.py compatible devices.

- uPIDE - µPIDE is a simple IDE for MicroPython.

Logging

- micropython-ulogger - Lightweight log module customized for MicroPython.
- scd30logger - Sensirion SCD30 based CO2, Humidity and Temperature Logger for MicroPython.
- sht15logger - MicroPython Temperature and Humidity Logger using Sensirion SHT15.

Shells

Jupyter

- micropython-magic - MicroPython integrated into Jupyter notebooks.
- jupyter_upydevice_kernel - Jupyter kernel to interact with a MicroPython board over its REPL interface.

On Device

- upy-shell - A simple command line-based shell for MicroPython.
- Micropython-Editor - Small on-board editor for Pyboard, WiPy, ESP8266, ESP32, PyCom and Adafruit devices written in Python.

On Host

- rshell - Copy or sync files to boards, enter REPL from your terminal.
- ampy - Utility to interact with a MicroPython board over a serial connection.
- mpbridge - A file system bridge to synchronize and manage files on a device running MicroPython.
- mpfshell - A simple shell-based file explorer for ESP8266 and WiPy.
- mpsync - A tool that automatically synchronizes code to a MicroPython board.
- mpremote - Powerful official shell that supports mounting the host's current directory on the target. Run code without changing the target's filesystem.
- MPRemoteEditor - A simple Windows IDE for developing with MicroPython MPRemote devices.
- uPyExplorer - Explorer for MicroPython Device.
- mpr - Wrapper for MicroPython mpremote tool.