

Introduction to MicroPython for Raspberry Pi Pico Part 2 - Software

Hamilton Python Users Group
12 Dec 2022
Ian Stewart

Version 2 of the presentation. This corrects diagrams where the Pull-Up and Pull-Down resistors were swapped.

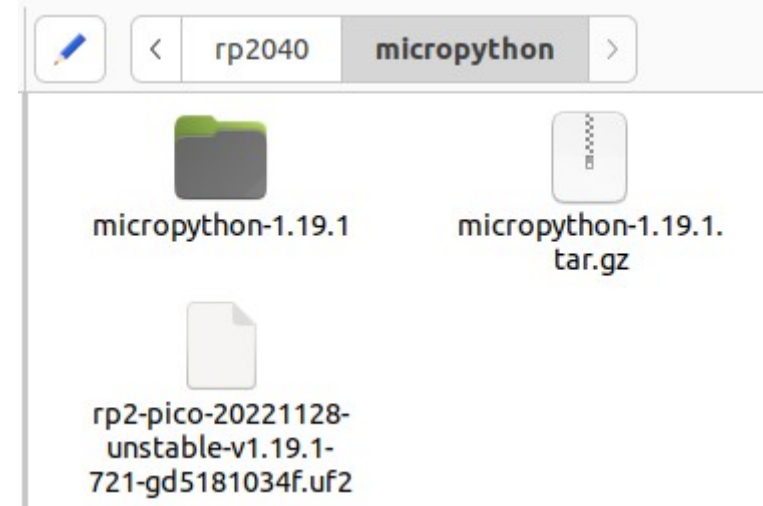
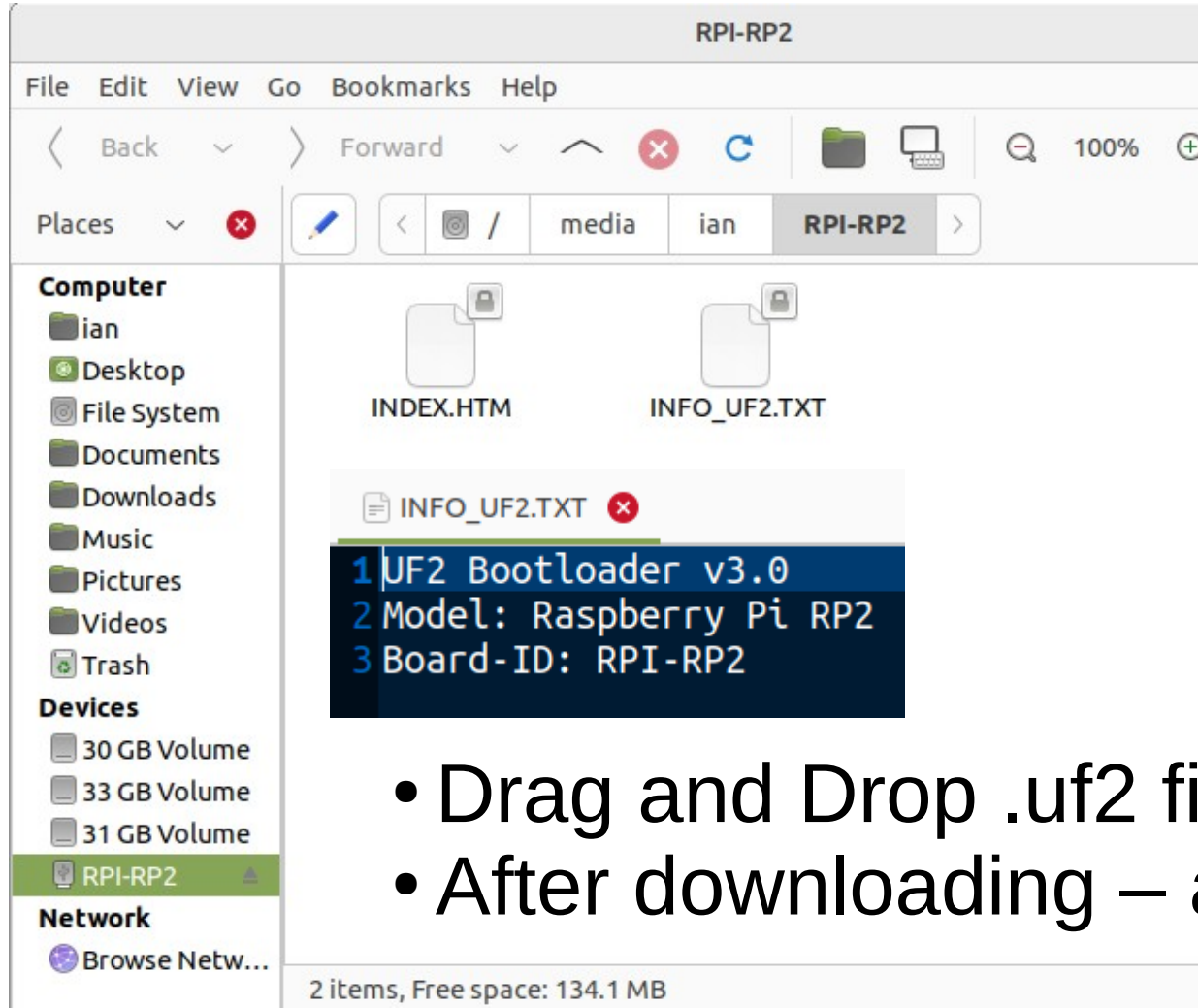
What is MicroPython?

- Full implementation of the Python 3 programming language that runs directly on embedded hardware like Raspberry Pi Pico.
- Interactive prompt (the REPL) to execute commands immediately via USB Serial port, and a built-in filesystem.
- The Pico port of MicroPython includes modules for accessing low-level chip-specific hardware.
- Documentation:
 - <https://www.raspberrypi.com/documentation/microcontrollers/micropython.html>
 - Demo of how to install to Pico module
- Website:
 - <https://micropython.org>
- Download Micropython for Pico:
 - <https://micropython.org/download/rp2-pico/>

What is Thonny?

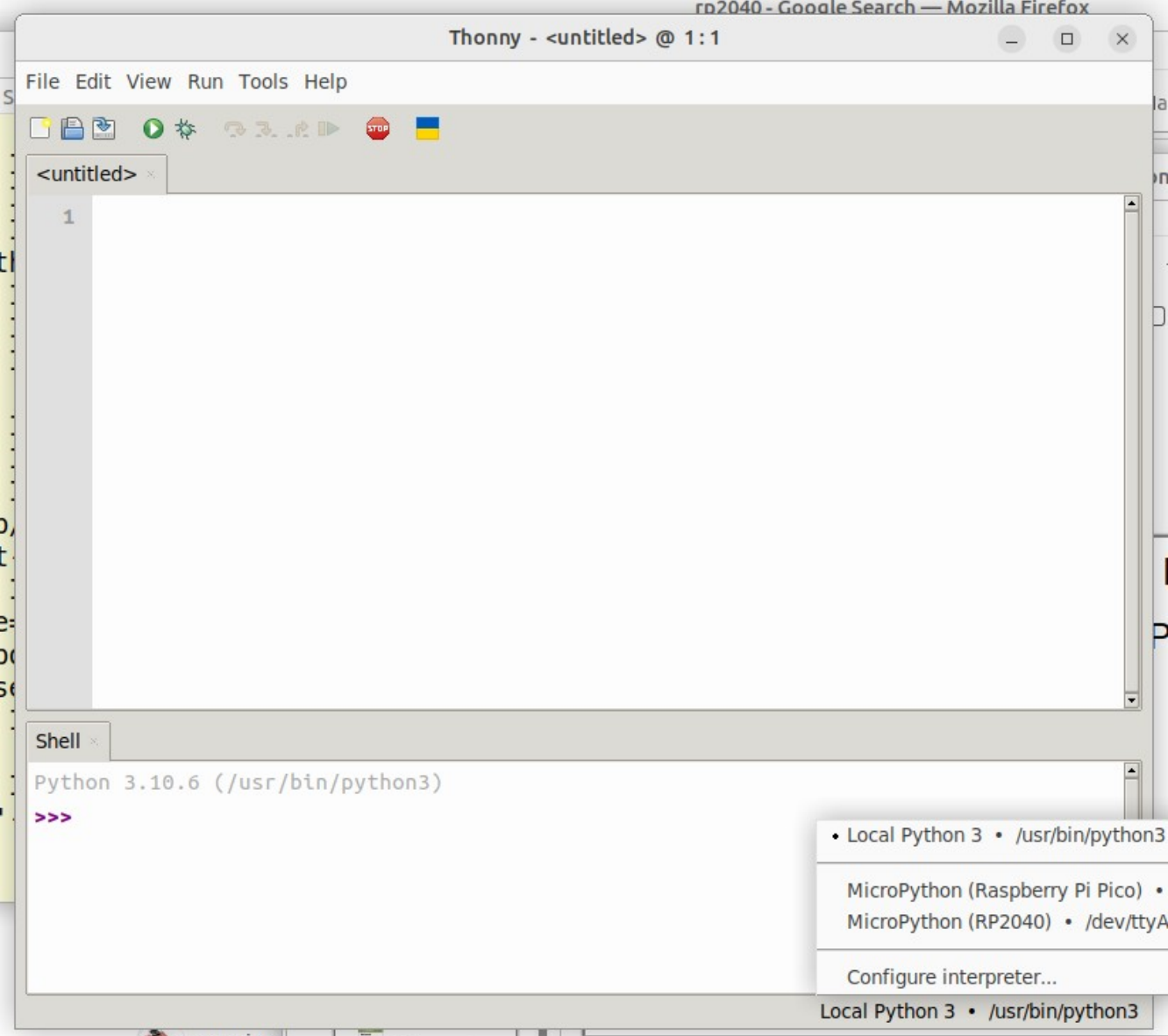
- Python IDE for Beginners.
- Written in Tkinter
- Website:
 - <https://github.com/thonny/thonny/wiki/Linux>
- Installation:
 - `sudo apt install python3-tk thonny`
 - `pip install thonny`

Plug in USB cable and release button



- Drag and Drop .uf2 file into RPI-RP2 folder
- After downloading – auto reboots

Thonny



Select
connection to
Raspberry Pi
Pico

Thonny not connecting to Pico via USB?

Error message:

```
Unable to connect to /dev/ttyACM0: [Errno 13] could not open port /dev/ttyACM0: [Errno 13] Permission denied: '/dev/ttyACM0'
```

Try adding yourself to the 'dialout' group:

```
> sudo usermod -a -G dialout <username>
```

(NB! You may need to reboot your system after this!)

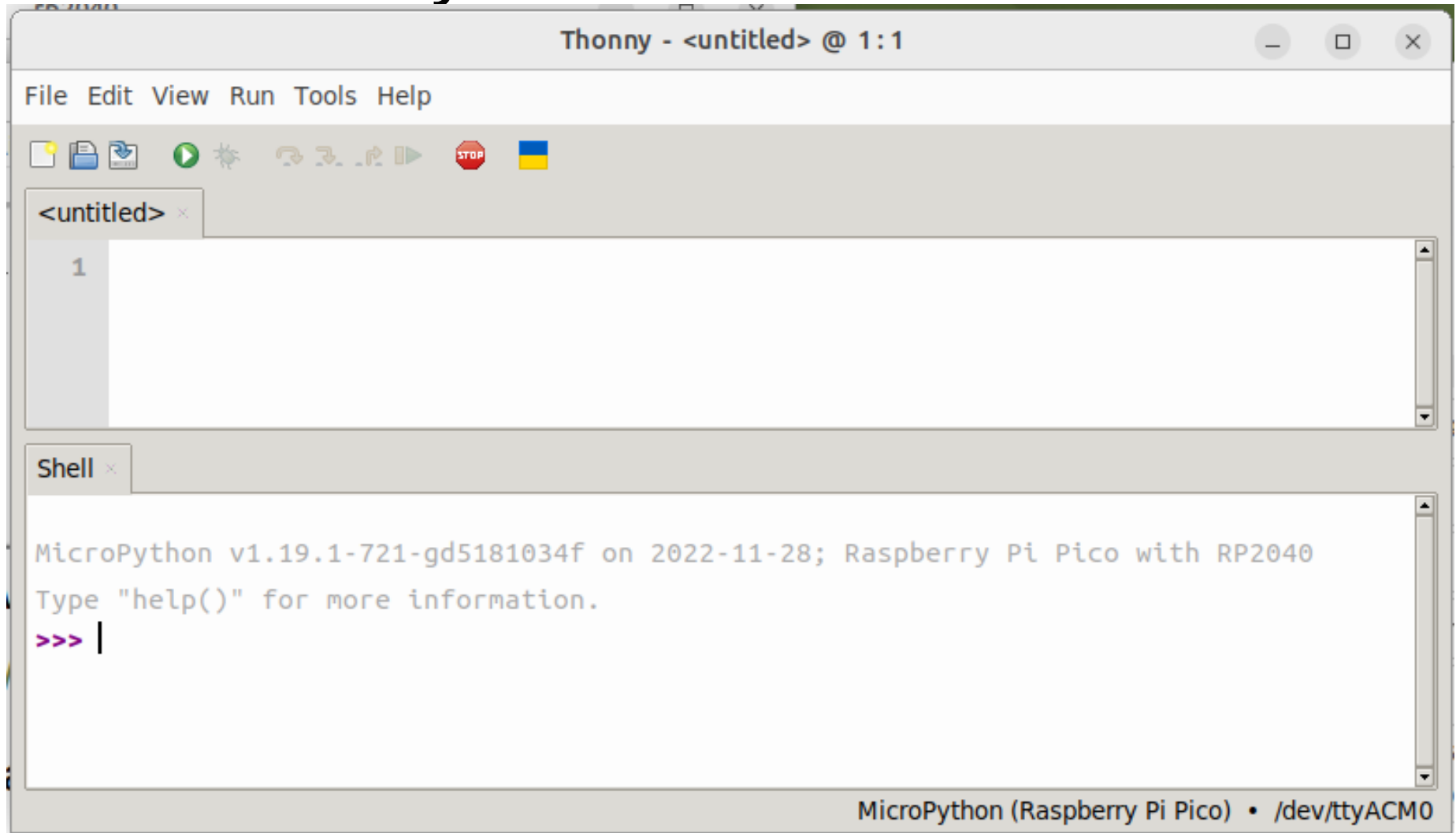
|

Process ended with exit code 1.

Allow Account to have Dialout:

- `ian@dell:~$ sudo usermod -a -G dialout ian`
- `[sudo] password for ian:`
- `ian@dell:~$ reboot`

MicroPython REPL >>> on Pico



MicroPython >>> help() 1/3

Welcome to MicroPython!

For online help please visit <https://micropython.org/help/>.

For access to the hardware use the 'machine' module. RP2 specific commands are in the 'rp2' module.

Quick overview of some objects:

- machine.Pin(pin) -- get a pin, eg machine.Pin(0)

- machine.Pin(pin, m, [p]) -- get a pin and configure it for IO mode m, pull mode p
methods: init(..), value([v]), high(), low(), irq(handler)

- machine.ADC(pin) -- make an analog object from a pin
methods: read_u16()

MicroPython >>> help() 2/3

`machine.PWM(pin)` -- make a PWM object from a pin
methods: `deinit()`, `freq([f])`, `duty_u16([d])`, `duty_ns([d])`

`machine.I2C(id)` -- create an I2C object (`id=0,1`)
methods: `readfrom(addr, buf, stop=True)`, `writeto(addr, buf, stop=True)`
`readfrom_mem(addr, memaddr, arg)`, `writeto_mem(addr, memaddr, arg)`

`machine.SPI(id, baudrate=1000000)` -- create an SPI object (`id=0,1`)
methods: `read(nbytes, write=0x00)`, `write(buf)`, `write_readinto(wr_buf, rd_buf)`

`machine.Timer(freq, callback)` -- create a software timer object
eg: `machine.Timer(freq=1, callback=lambda t:print(t))`

MicroPython >>> help() 3/3

Pins are numbered 0-29, and 26-29 have ADC capabilities

Pin IO modes are: Pin.IN, Pin.OUT, Pin.ALT

Pin pull modes are: Pin.PULL_UP, Pin.PULL_DOWN

Useful control commands:

- CTRL-C -- interrupt a running program

- CTRL-D -- on a blank line, do a soft reset of the board

- CTRL-E -- on a blank line, enter paste mode

For further help on a specific object, type `help(obj)`

For a list of available modules, type `help('modules')`

MicroPython >>> help("modules")

```
>>> help(modules")
```

__main__	framebuf	uasyncio/funcs	ujson
_boot	gc	uasyncio/lock	umachine
_boot_fat	math	uasyncio/stream	uos
_onewire	micropython	ubinascii	urandom
_rp2	neopixel	ucollections	ure
_thread	onewire	ucryptolib	uselect
_uasyncio	rp2	uctypes	ustruct
builtins	uarray	uerrno	usys
cmath	uasyncio/___init___	uhashlib	utime
dht	uasyncio/core	uheapq	uzlib
ds18x20	uasyncio/event	uio	

Plus any modules on the filesystem

```
>>>
```

>>> dir()

```
['machine', '__thonny_helper', 'usys', '__name__', 'rp2']
```

>>> dir(machine)

```
['__class__', '__name__', '__dict__', 'ADC', 'I2C', 'I2S',  
'PWM', 'PWRON_RESET', 'Pin', 'RTC', 'SPI', 'Signal',  
'SoftI2C', 'SoftSPI', 'Timer', 'UART', 'WDT', 'WDT_RESET',  
'bitstream', 'bootloader', 'deepsleep', 'dht_readinto',  
'disable_irq', 'enable_irq', 'freq', 'idle', 'lightsleep',  
'mem16', 'mem32', 'mem8', 'reset', 'reset_cause',  
'soft_reset', 'time_pulse_us', 'unique_id']
```

>>> dir(rp2)

```
['__class__', '__name__', 'const', '__dict__', '__file__',  
'Flash', 'PIO', 'StateMachine', 'asm_pio_encode',  
'PIOASMEError', 'PIOASMEmit', 'asm_pio', '_pio_funcs']
```

>>> dir()

```
>>> import sys
```

```
>>> import sys
```

```
>>> dir(sys)
```

```
['__class__', '__name__', '__dict__', 'argv', 'byteorder',  
'exit', 'implementation', 'maxsize', 'modules', 'path',  
'platform', 'print_exception', 'ps1', 'ps2', 'stderr',  
'stdin', 'stdout', 'version', 'version_info']
```

```
>>> dir(sys)
```

```
['__class__', '__name__', '__dict__', 'argv', 'byteorder',  
'exit', 'implementation', 'maxsize', 'modules', 'path',  
'platform', 'print_exception', 'ps1', 'ps2', 'stderr',  
'stdin', 'stdout', 'version', 'version_info']
```

```
>>> sys.
```

```
>>> sys.version_info  
(3, 4, 0)
```

```
>>> sys.version  
'3.4.0; MicroPython v1.19.1-721-gd5181034f on 2022-11-28'
```

```
>>> sys.platform  
'rp2'
```

>>> OS.

```
>>> import os
```

```
>>> dir(os)
['__class__', '__name__', 'remove', '__dict__', 'VfsFat',
'VfsLfs2', 'chdir', 'getcwd', 'ilistdir', 'listdir',
'mkdir', 'mount', 'rename', 'rmdir', 'stat', 'statvfs',
'umount', 'uname', 'unlink', 'urandom']
```

```
>>> os.uname()
(sysname='rp2', nodename='rp2', release='1.19.1',
version='v1.19.1-721-gd5181034f on 2022-11-28 (GNU 12.1.0
MinSizeRel)', machine='Raspberry Pi Pico with RP2040')
```

>>> OS.

```
>>> os.getcwd()  
'/'
```

```
>>> os.listdir()  
['adc_temp.py', 'blink.py', 'get_info.py',  
'pin16_blink.py', 'pwm_pin16.py', 'pwm_pin25.py',  
'pwm_pin25_dimmer.py', 'rtc_time.py', 'switch.py',  
'switch_interrupt.py', 'switch_interrupt_routine.py']  
>>>
```



Python programs written by me and
saved on the Pico's 2MB of flash
RAM

First Programs...

```
>>> print ("hello world")  
hello world
```

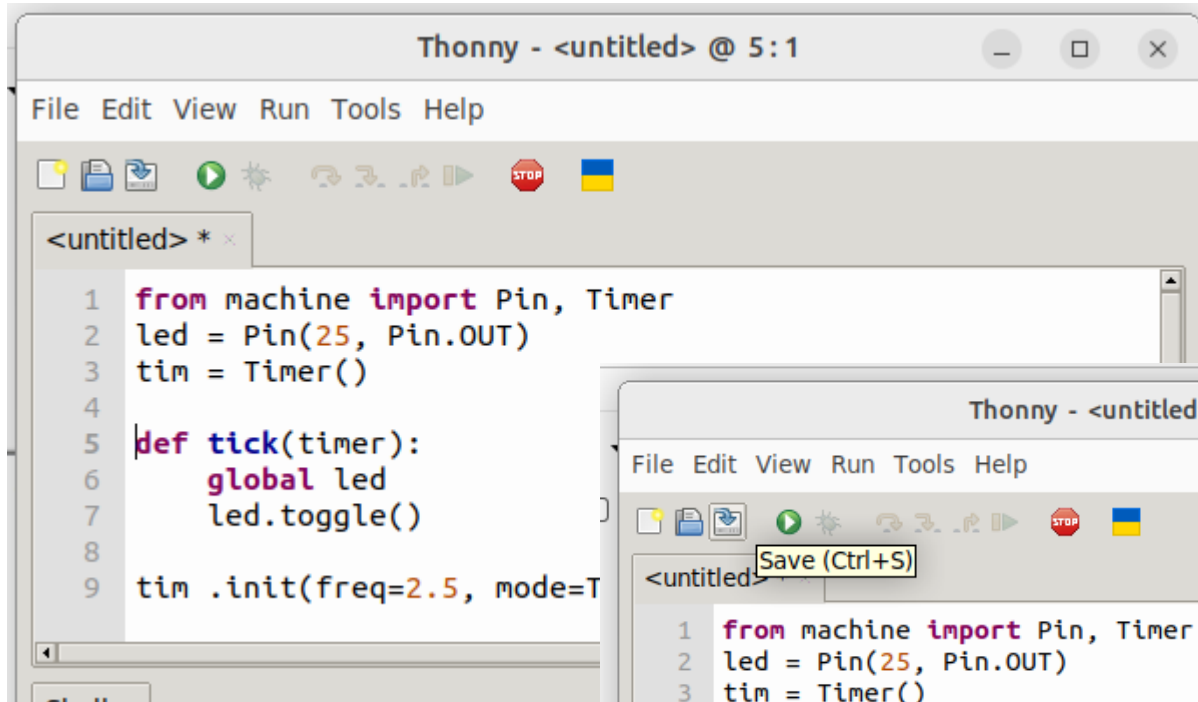
```
>>> from machine import Pin  
>>> led = Pin(25, Pin.OUT)  
>>> led.on()  
>>> led.off()
```

Pin25 is for the LED on-board the Pico module

```
>>> from machine import Pin, Timer  
>>> led = Pin(25, Pin.OUT)  
>>> tim = Timer()  
>>> def tick(timer):  
    global led  
    led.toggle()
```

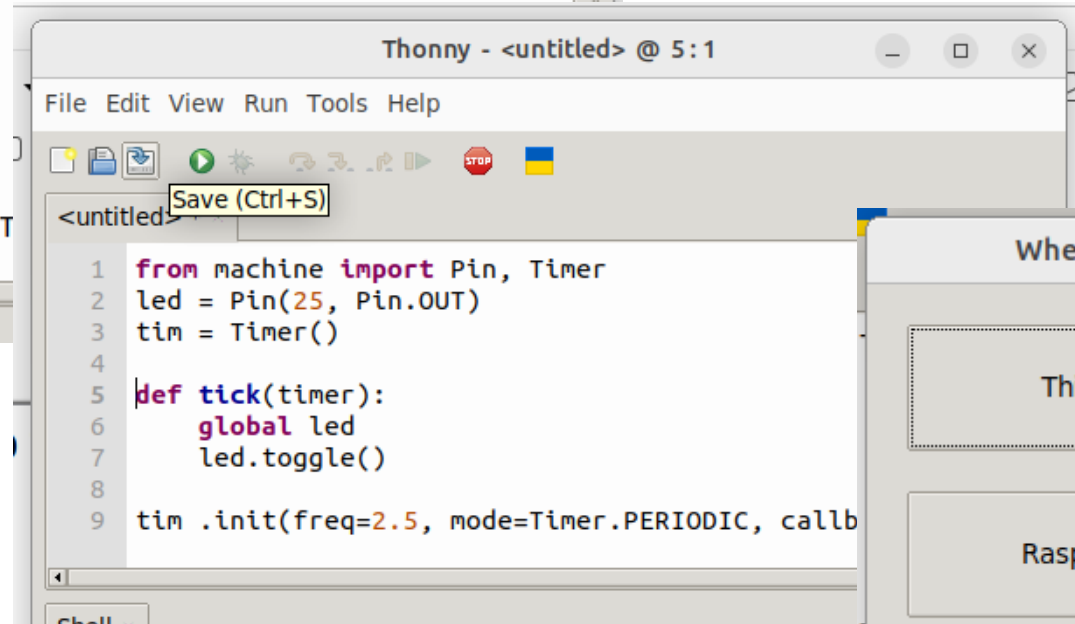
```
>>> tim .init(freq=2.5, mode=Timer.PERIODIC, callback=tick)
```

Saving program to Pico flash



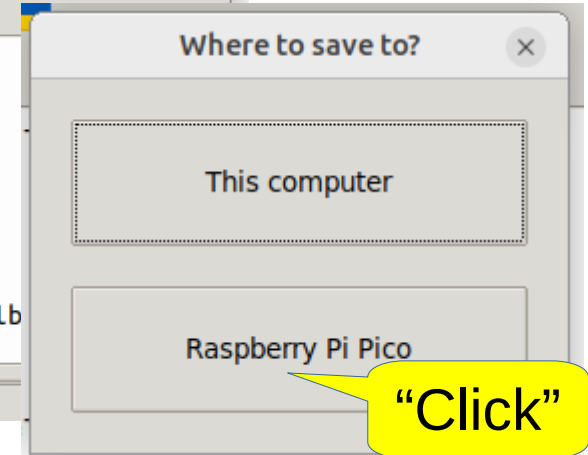
```
Thonny - <untitled> @ 5:1
File Edit View Run Tools Help

<untitled> * x
1 from machine import Pin, Timer
2 led = Pin(25, Pin.OUT)
3 tim = Timer()
4
5 def tick(timer):
6     global led
7     led.toggle()
8
9 tim .init(freq=2.5, mode=Timer.PERIODIC, callback=tick)
```

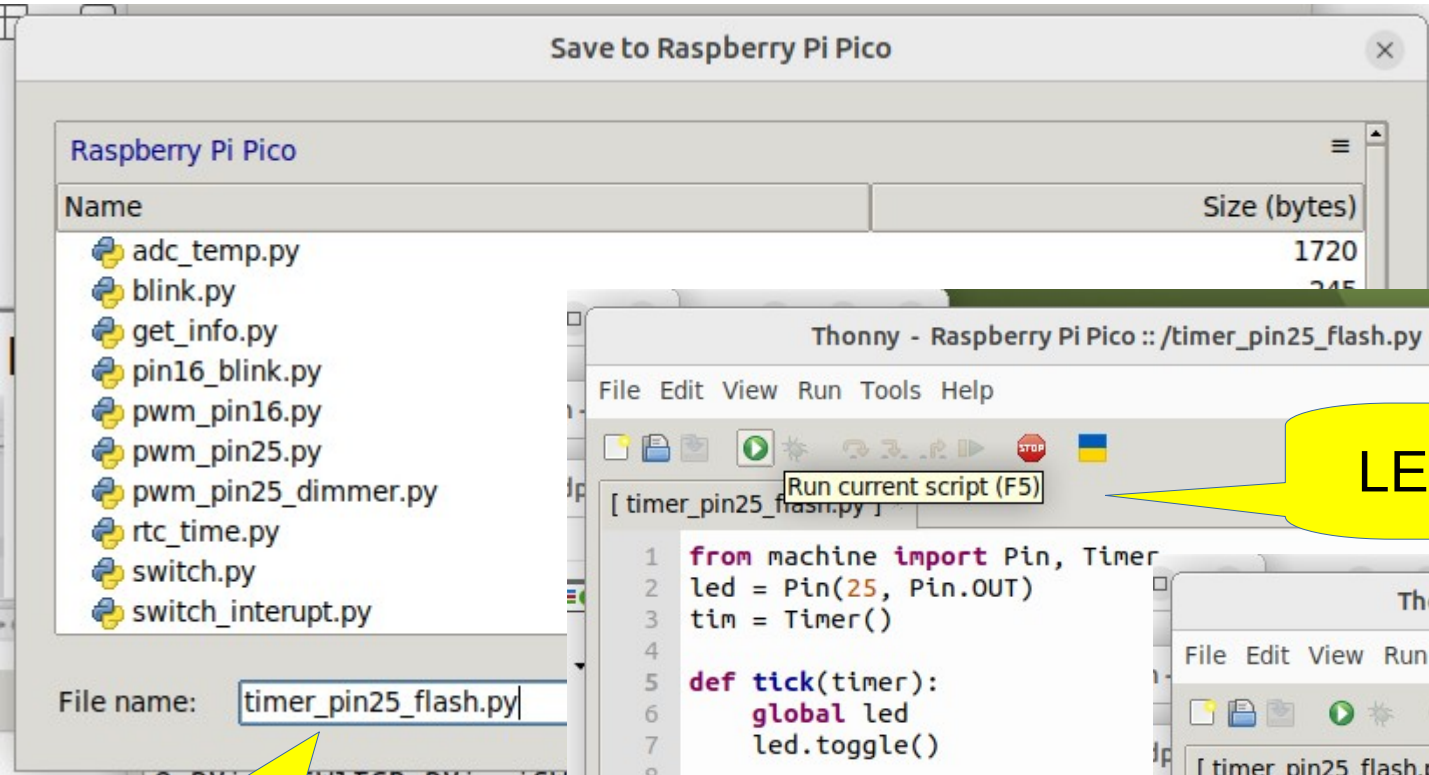


```
Thonny - <untitled> @ 5:1
File Edit View Run Tools Help

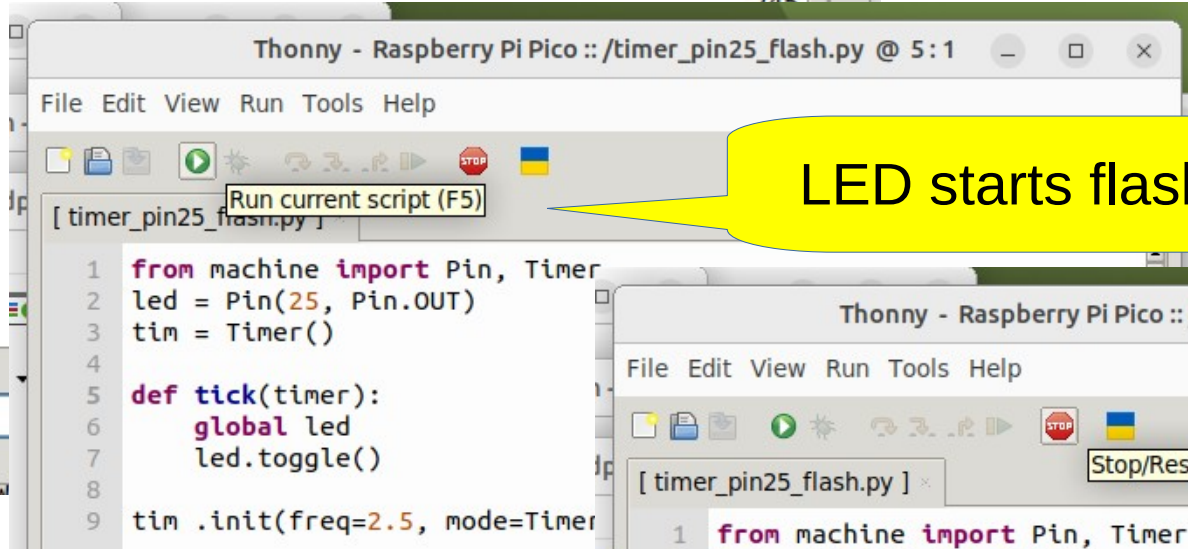
<untitled>
1 from machine import Pin, Timer
2 led = Pin(25, Pin.OUT)
3 tim = Timer()
4
5 def tick(timer):
6     global led
7     led.toggle()
8
9 tim .init(freq=2.5, mode=Timer.PERIODIC, callb
```



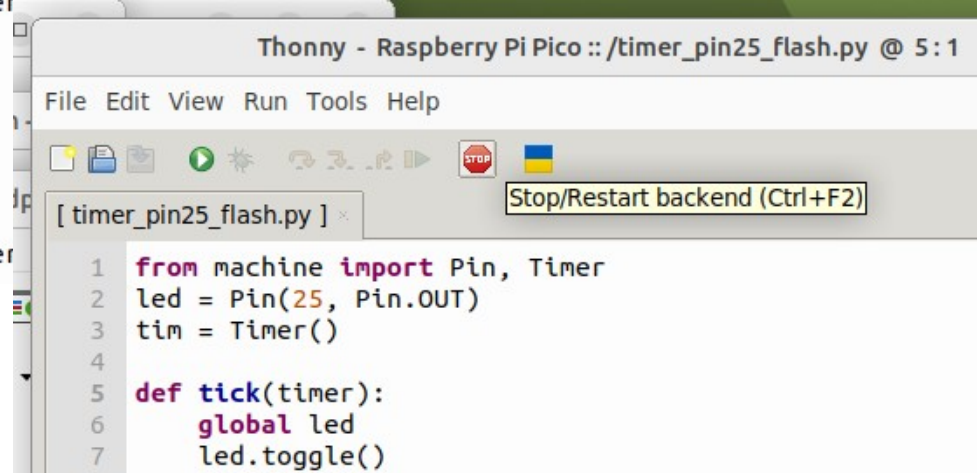
Saving program to Pico flash. Start / Stop



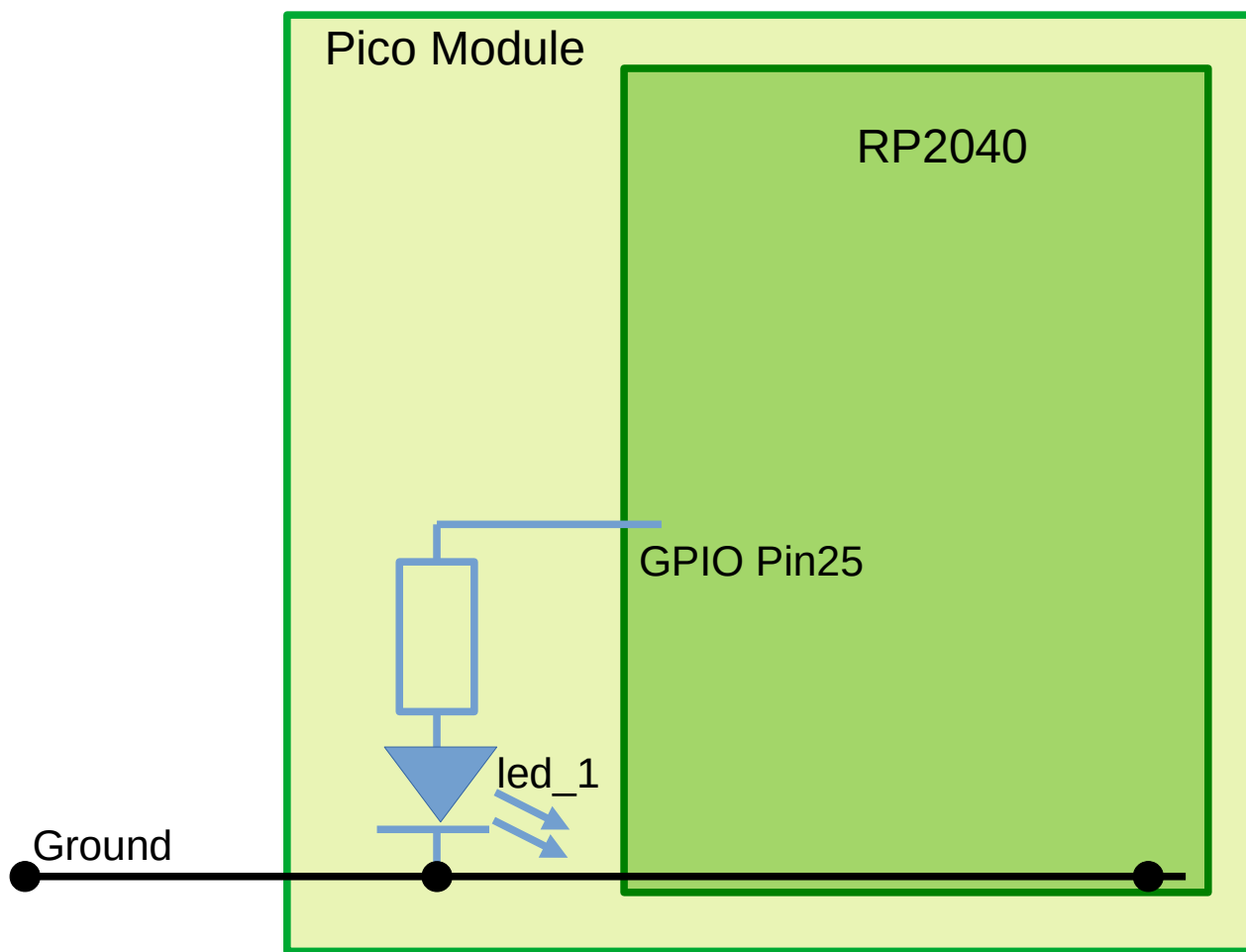
Give program a name



LED starts flashing

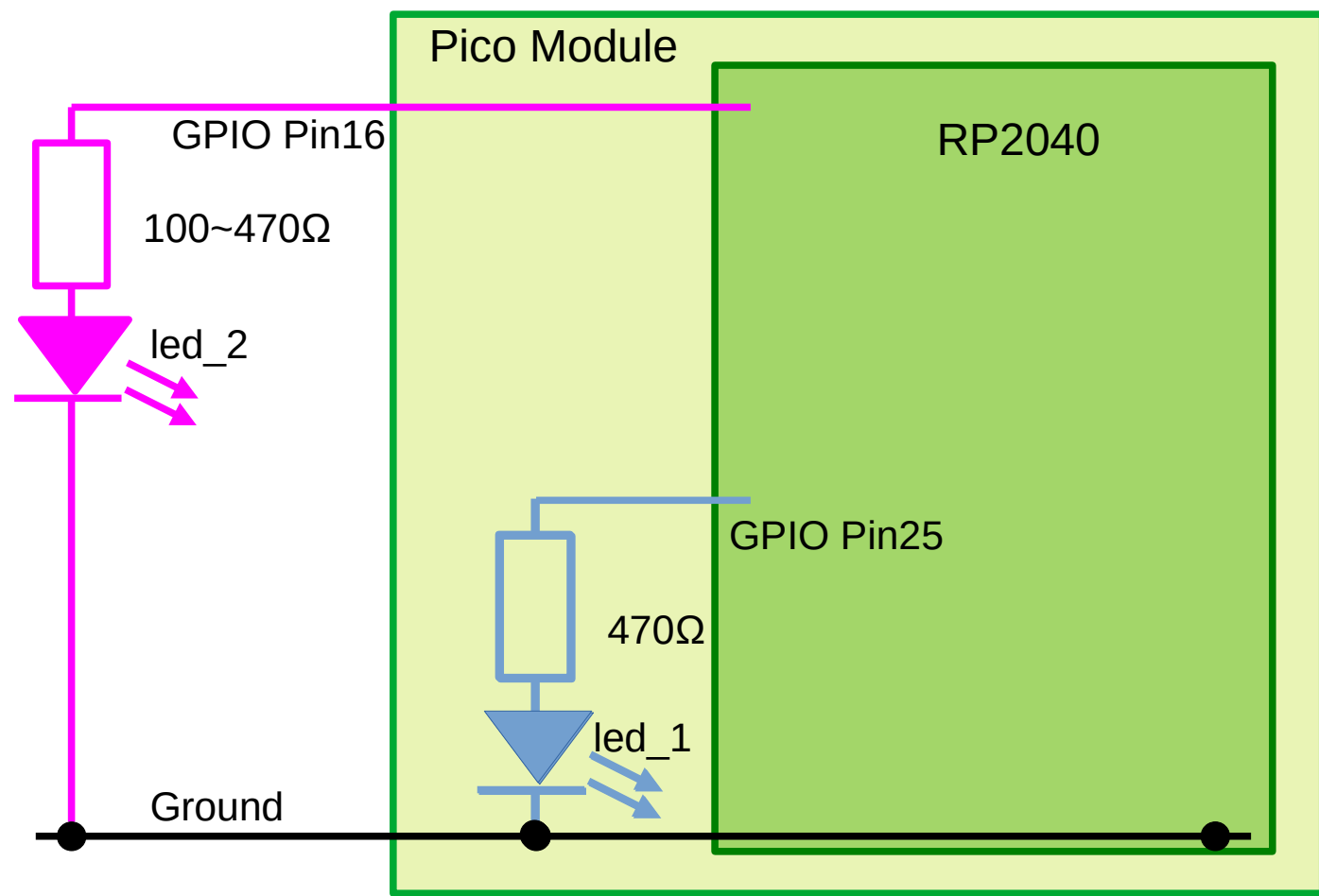


GPIO Pin OUT on Pico Module



```
from machine import Pin  
led_1 = Pin(25, Pin.OUT)
```

```
Turn ON Led 1:  
led_1.value(1)  
led_1.on()
```

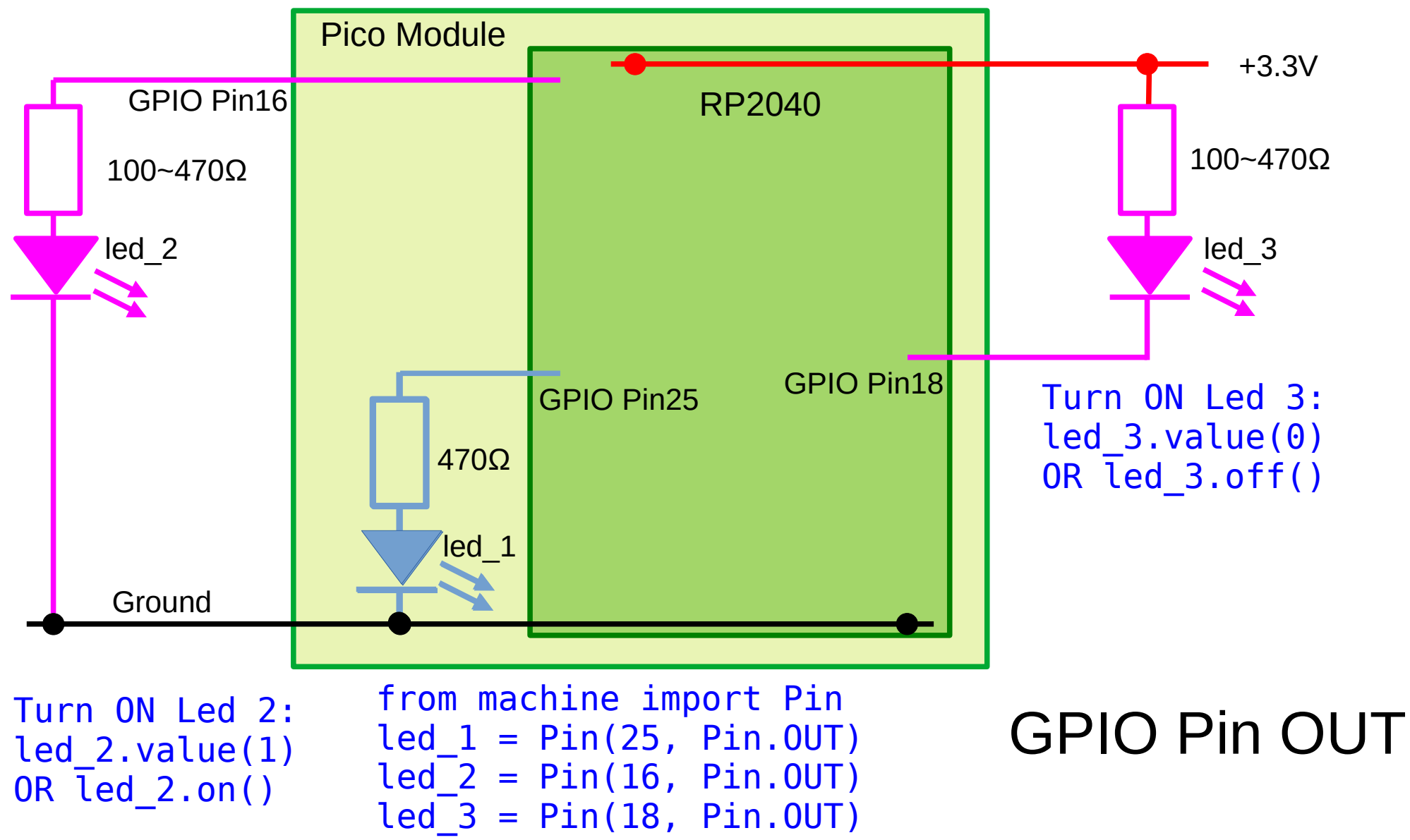


GPIO Pin OUT
on Pico Module.

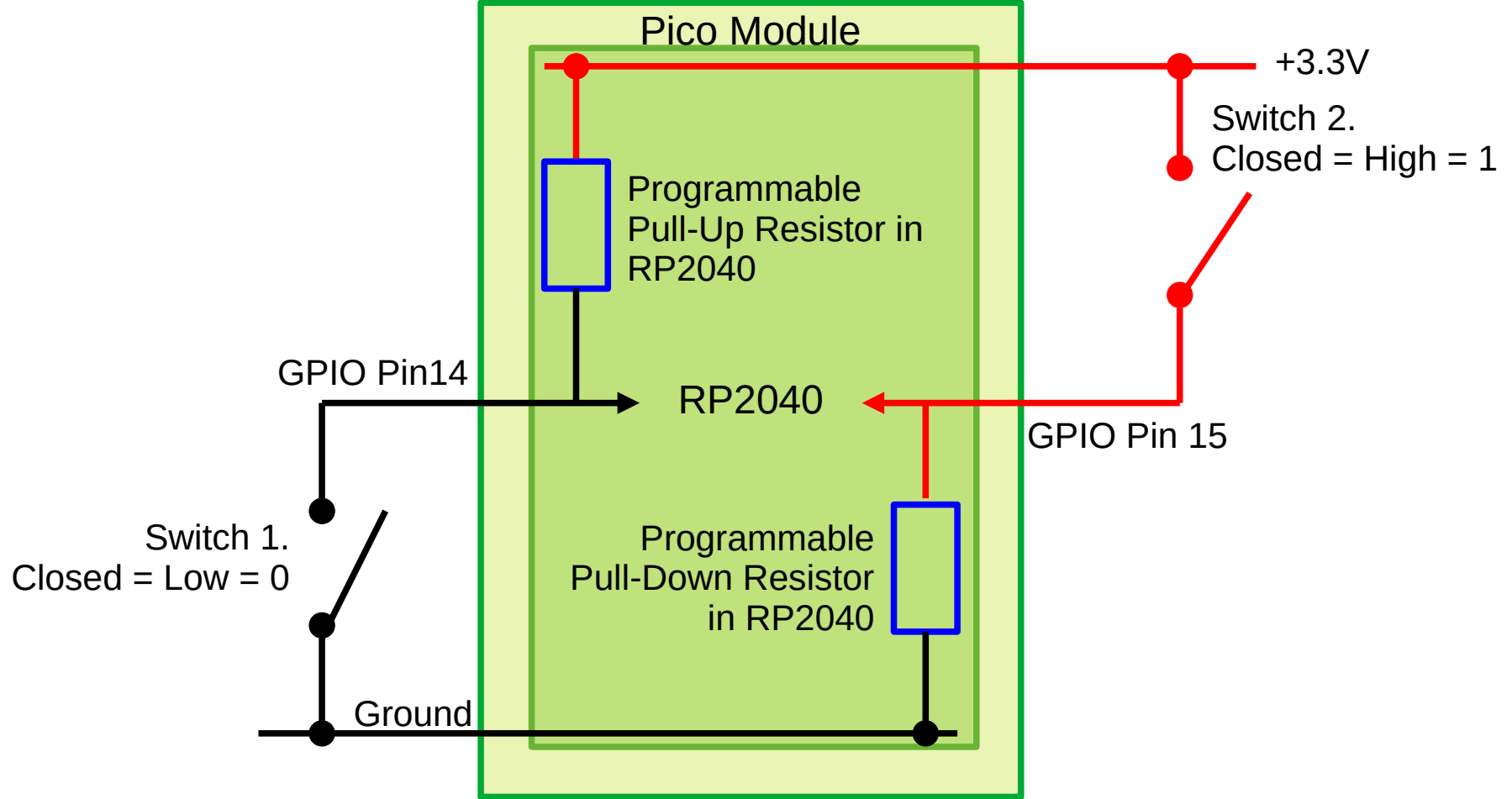
GPIO Pin OUT
External

Turn ON Led 2:
`led_2.value(1)`
OR `led_2.on()`

```
from machine import Pin  
led_1 = Pin(25, Pin.OUT)  
led_2 = Pin(16, Pin.OUT)
```



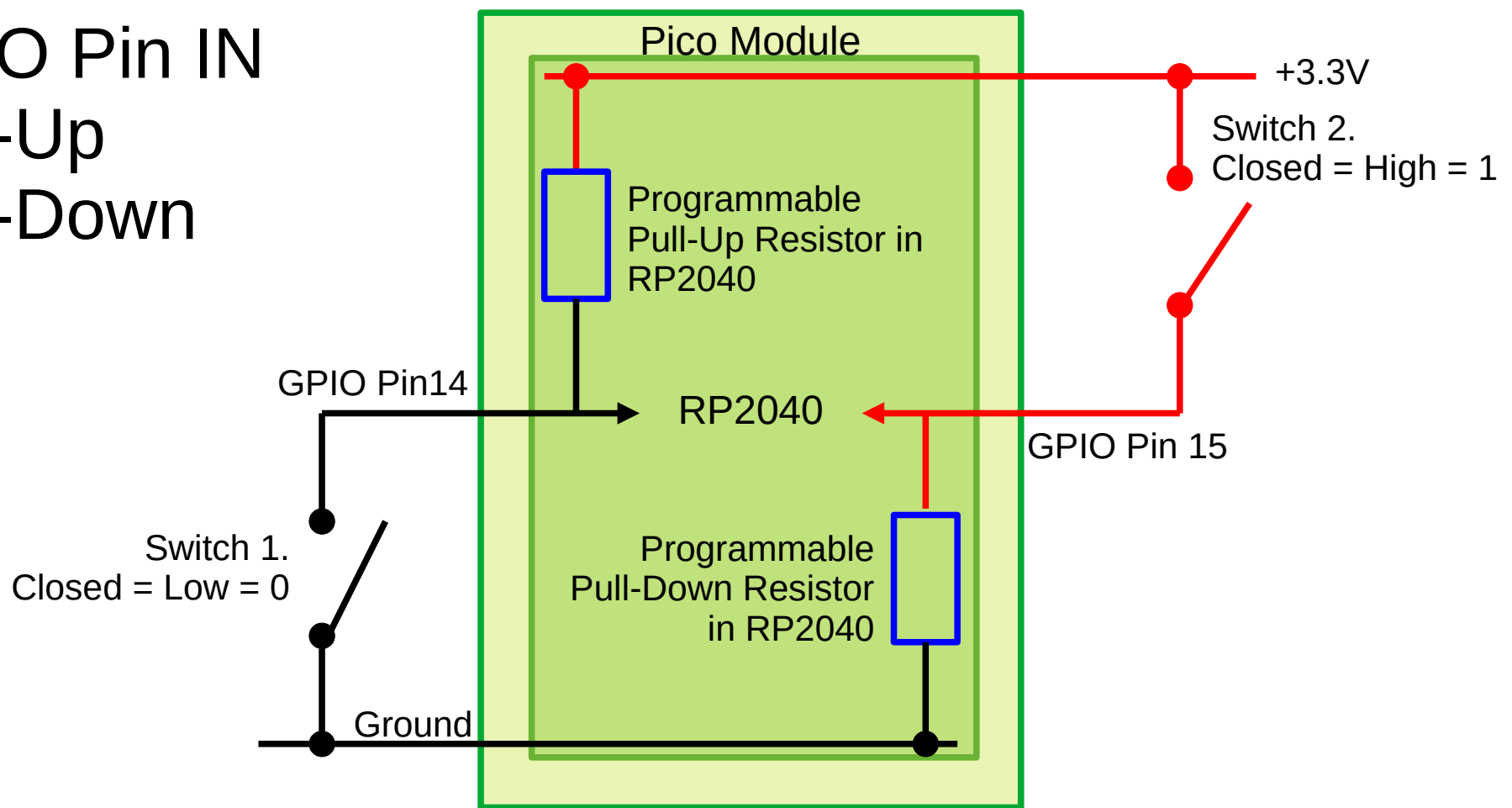
GPIO Pin IN Pull-Up / Pull-Down



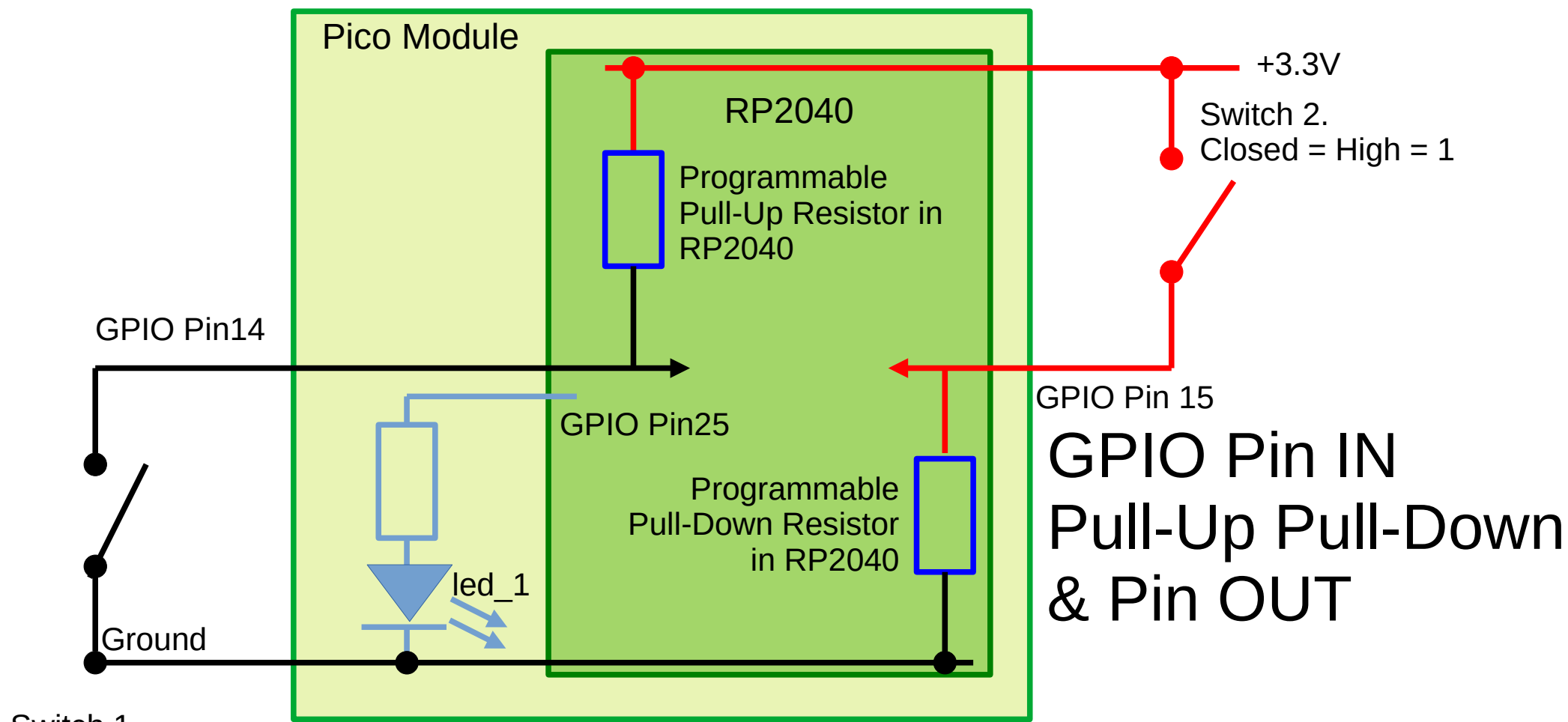
GPIO Pin IN

Pull-Up

Pull-Down



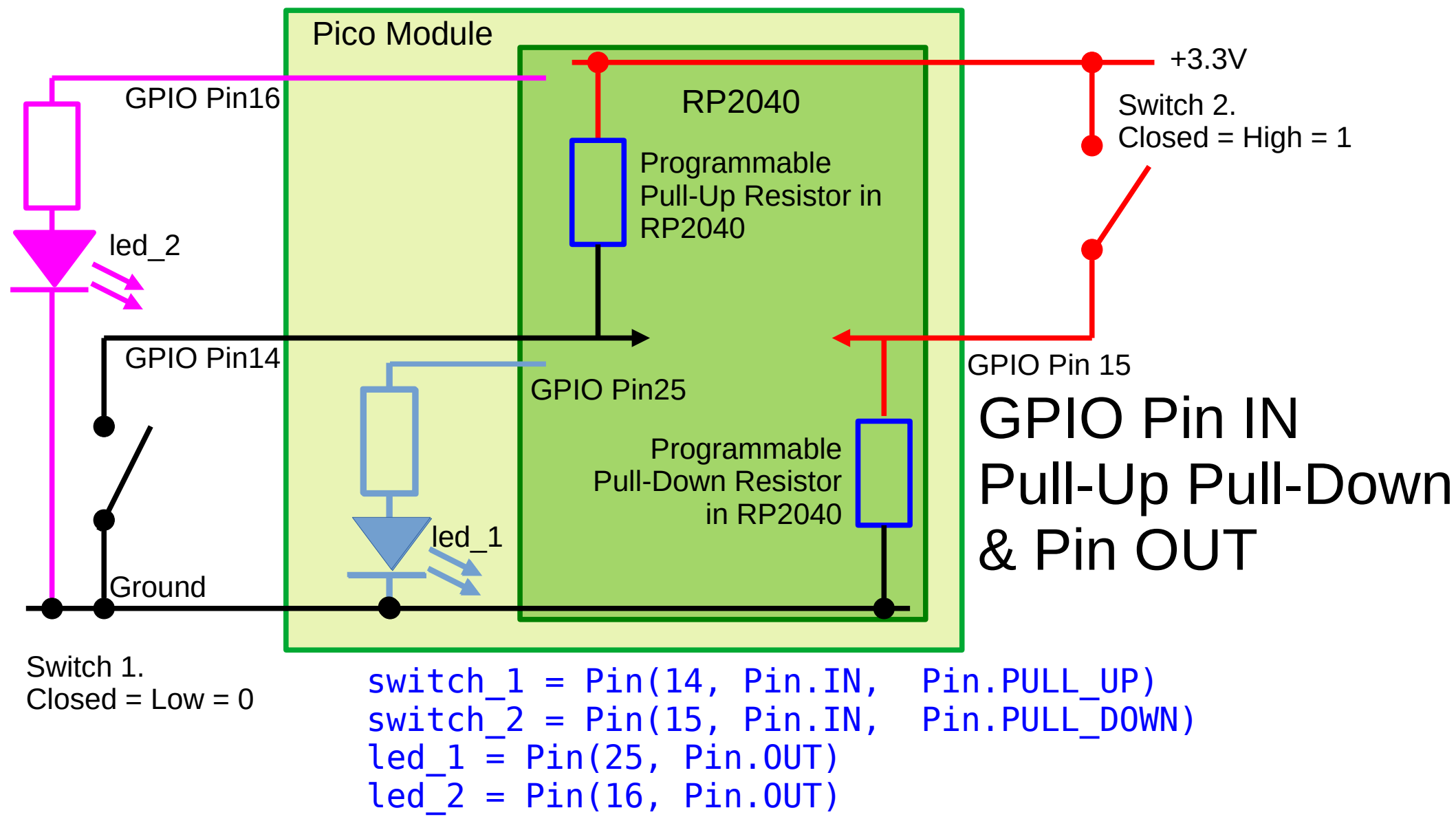
```
from machine import Pin
switch_1 = Pin(14, Pin.IN, Pin.PULL_UP)
switch_2 = Pin(15, Pin.IN, Pin.PULL_DOWN)
```

GPIO Pin IN
Pull-Up Pull-Down
& Pin OUT

Switch 1.
Closed = Low = 0

```
from machine import Pin
switch_1 = Pin(14, Pin.IN, Pin.PULL_UP)
switch_2 = Pin(15, Pin.IN, Pin.PULL_DOWN)
led_1 = Pin(25, Pin.OUT)
```



MicroPython

Demos:

- RTC
- Switches
- LEDs
- PWM
- Timer
- Debouncing
- Review MicroPython 1.19 source code