
uPy_A*PDS9960LibraryDocumentation*
Release 1.0

Rune Langøy

Feb 06, 2023

Contents

1	Documentation	3
2	Dependencies	5
3	Installation	7
4	Examples	9
5	APDS9960 Example	11
6	APDS9900 Example	13
6.1	Hardware Set-up	13
6.2	Basics	13
6.2.1	Proximity	14
6.2.2	Light Sensing	14
6.3	Debug	14
7	Sphinx documentation	17
8	Contributing	19
9	Table of Contents	21
9.1	Proximity & Light Sensing	21
9.1.1	Proximity Examples	21
9.1.1.1	Simple Example	21
9.1.1.2	Regular Example	21
9.1.1.3	Simple IRQ	22
9.1.1.4	IRQ Example	23
9.1.2	Light Sensor Examples	24
9.1.2.1	Light sensing example	24
9.1.2.2	Ambient light IRQ	24
9.2	Debug	25
9.2.1	I2C Debug Example	25
9.3	Thonny IDE Tutorial	25
9.3.1	Download and install	25
9.3.1.1	Download up Micropython (ESP8266) dev enviroment	25
9.3.2	Start Thonny	26
9.3.3	Flash new firmware	26

9.3.3.1	Installing esptool.py	26
9.3.3.2	Seting up Micropython (ESP8266) dev enviroment	27
9.3.4	Running the examples	29
9.3.4.1	Uploading uPy_APDS9960	29
9.3.4.2	Running a example program	30
9.4	uPy_APDS9960 Module	32
10	Indices and tables	33

Another APDS9960 / GY-9960LLC / APDS9900 micro python library optimized for ESP8266 / ESP12-E for:

- Light Sensing (Ambient Light and RGB Color Sensing)
- Proximity Sensing

CHAPTER 1

Documentation

Complete documentation is hosted on the “Read the Docs” page upy-aps9960.readthedocs.io

CHAPTER 2

Dependencies

This driver depends on:

- [MicroPython](#)

Tested on:

Sensor: [GY-9960LLC](#)

Sensor: [APDS-9900](#)

Devboard: Node MCU v1.0 & Raspberry PI Pico

CHAPTER 3

Installation

- Flash the device with MicroPython
- Copy the folder uPy_APDS9960 and content (apds9960LITE.py) to the root folder for APDS9960 circuits
- Copy the folder uPy_APDS9900 and content (apds9900LITE.py) to the root folder for APDS9900 circuits

The steps above is desccribed in the [Thonny IDE tutorial](#).

CHAPTER 4

Examples

The examples in this repository use the NodeMCU devboard. If you use the devboard to use RPi Pico, please change the I2C interface as shown in the code below.

```
#Change I2C interface from:  
# i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))  
#to:  
i2c = machine.I2C(0, scl=machine.Pin(17), sda=machine.Pin(16))
```

Here is the NodeMCU Hookup.

CHAPTER 5

APDS9960 Example

```
import machine
from time import sleep_ms
from uPy_APDS9960.apds9960LITE import APDS9960LITE

#Init I2C Buss on RP2040
i2c = machine.I2C(0,scl=machine.Pin(17), sda=machine.Pin(16))

apds9960=APDS9960LITE(i2c)      # Enable sensor
apds9960.prox.enableSensor()    # Enable Proximit sensing

while True:
    sleep_ms(25) # wait for readout to be ready
    print(apds9960.prox.proximityLevel) #Print the proximity value
```


APDS9900 Example

```
import machine
from time import sleep_ms
from uPy_APDS9900.apds9900LITE import APDS9900LITE

#Init I2C Buss on RP2040
i2c = machine.I2C(0, scl=machine.Pin(17), sda=machine.Pin(16))

apds9900=APDS9900LITE(i2c)      # Enable sensor
apds9900.prox.enableSensor()    # Enable Proximit sensing

while True:
    sleep_ms(25) # wait for readout to be ready
    print(apds9900.prox.proximityLevel) #Print the proximity value
```

6.1 Hardware Set-up

Connect Vin to 3.3 V or 5 V power source, GND to ground, SCL and SDA to the appropriate pins to the Raspberry PI Pico

APDS9960	Name	Remarks	RPI PICO	Function
1	VIN	+3.3V Power	36	3V3
2	GND	Ground	GND	GND
3	SCL	I2C clock	22	GP17 (SCL)
4	SDA	I2C Data	21	GP16 (SDA)
5	INT	Interrupt	26	GP20

6.2 Basics

Of course, you must import the device and library :)

```
import machine
from time import sleep_ms
from uPy_APDS9960.apds9960LITE import APDS9960LITE
```

To set-up the device to gather data, initialize the I2C-device using SCL and SDA pins. Then initialize the library.

```
i2c = machine.I2C(0, scl=machine.Pin(17), sda=machine.Pin(16))
apds9960=APDS9960LITE(i2c)           # Poweron APDS9960
```

6.2.1 Proximity

Proximity functionalites is accessed torough the apds9960.prox member PROX

```
apds9960.prox.enableSensor()          # Enable Proximity sensing
sleep_ms(25)                          # wait for readout to be ready
print(apds9960.prox.proximityLevel)   # Print the proximity value
```

6.2.2 Light Sensing

Proximity functionalites is accessed torough the apds9960.als member ALS

```
apds9960.als.enableSensor()           # Enable Light sensor
sleep_ms(25)                          # Wait for readout to be ready
print(apds9960.als.ambientLightLevel) # Print the ambient light value
```

6.3 Debug

If things does not work try to run the script below to verify that it i2c communication with the apds9960 is working as expected

```
import machine
i2c = machine.I2C(0, scl=machine.Pin(17), sda=machine.Pin(16))

print('Scan i2c bus...')
devices = i2c.scan()

if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:', len(devices))

    for device in devices:
        print("Decimal address: ", device, " | Hexa address: ", hex(device))

        if(device==0x39): # APDS9960 Address = 0x39
            deviceID=i2c.readfrom_mem(devices[0], 0x92, 1) #Get deviceID
            deviceID=int.from_bytes(deviceID, 'big')         #Conv byte to int
            if(deviceID==0x29):
                deviceID=9900
            elif(deviceID==0x20):
                deviceID=9901
```

(continues on next page)

(continued from previous page)

```
else:
    deviceID=9960

print("Found ADPS-", deviceID)
```

If successful the output should be:

```
Scan i2c bus...
i2c devices found: 1
Decimal address: 57 | Hexa address: 0x39
Found ADPS- 9960
```

Note: Be aware if the output shows:

"many i2c devices was listed"	check if the i2c pins are allocated correctly
"No i2c device"	check if the power is correctly connected

The Device id can be 0xa8, 0xab 0x9c or 0x55.)

CHAPTER 7

Sphinx documentation

[Sphinx the Python Documentation Generator](#) is used for this documentation, if you like to build a local copy of the documentation install Sphinx :

```
python -m pip install sphinx
```

Crete html doc by

```
cd docs
make html
```

The html pages would be located at : docs/_build/html

CHAPTER 8

Contributing

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

9.1 Proximity & Light Sensing

9.1.1 Proximity Examples

Examples that demonstrates the use of apds9960 as a proximity sensor

9.1.1.1 Simple Example

Basic proximity test program.

Listing 1: examples/prox/simple_proximity_apds9960.py

```
1 import machine
2 from time import sleep_ms
3 from uPy_APDS9960.apds9960LITE import APDS9960LITE
4
5 #Init I2C Buss
6 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
7
8 apds9960=APDS9960LITE(i2c)      # Enable sensor
9 apds9960.prox.enableSensor()    # Enable Proximit sensing
10
11 while True:
12     sleep_ms(25) # wait for readout to be ready
13     print(apds9960.prox.proximityLevel)    #Print the proximity value
```

9.1.1.2 Regular Example

Example exposing more functions

Listing 2: examples/prox/proximity_apds9960.py

```
1 import machine
2 from time import sleep_ms
3 from uPy_APDS9960.apds9960LITE import APDS9960LITE
4
5 # Proximity Gain (PGAIN) values
6 APDS9960_PGAIN_1X = const(0)
7 APDS9960_PGAIN_2X = const(1)
8 APDS9960_PGAIN_4X = const(2)
9 APDS9960_PGAIN_8X = const(3)
10
11 # LED Drive values
12 APDS9960_LED_DRIVE_100MA = const(0)
13 APDS9960_LED_DRIVE_50MA = const(1)
14 APDS9960_LED_DRIVE_25MA = const(2)
15 APDS9960_LED_DRIVE_12_5MA = const(3)
16
17 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
18
19 apds9960=APDS9960LITE(i2c)
20 apds9960.prox.eLEDCurrent =APDS9960_LED_DRIVE_100MA
21 apds9960.prox.eProximityGain=APDS9960_PGAIN_8X
22 apds9960.prox.enableSensor()
23
24 sleep_ms(50)
25
26 while True:
27     sleep_ms(50)
28     print("proximity:", apds9960.prox.proximityLevel )
```

9.1.1.3 Simple IRQ

Example showing use of a hardware IRQ raised at a given proximity value

Listing 3: examples/prox/simple_irq_proximity_apds9960.py

```
1 import machine
2 from time import sleep_ms
3 from uPy_APDS9960.apds9960LITE import APDS9960LITE
4
5 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
6
7 apds9960=APDS9960LITE(i2c)
8 apds9960.prox.eLEDCurrent =0 # LED_DRIVE_100MA
9 apds9960.prox.eProximityGain =3 # PGAIN_8X
10 apds9960.prox.enableSensor()
11
12 #IRQ Functionalities
13 apds9960.prox.setInterruptThreshold(high=10,low=0,persistence=7)
14 apds9960.prox.enableInterrupt()
15
16 ProxThPin=machine.Pin(0, machine.Pin.IN ,machine.Pin.PULL_UP)
17
18 sleep_ms(50)
19
```

(continues on next page)

(continued from previous page)

```

20 while True:
21     sleep_ms(50)
22
23     if (ProxThPin.value() == 0):
24         print("proximity:", apds9960.prox.proximityLevel )
25         apds9960.prox.clearInterrupt()

```

9.1.1.4 IRQ Example

Example showing use of interrupts exposing more functionalities

Listing 4: examples/prox/irq_proximity_apds9960.py

```

1  import machine
2  from time import sleep_ms
3  from uPy_APDS9960.apds9960LITE import APDS9960LITE
4
5  # Proximity Gain (PGAIN) values
6  APDS9960_PGAIN_1X = const(0)
7  APDS9960_PGAIN_2X = const(1)
8  APDS9960_PGAIN_4X = const(2)
9  APDS9960_PGAIN_8X = const(3)
10
11 # LED Drive values
12 APDS9960_LED_DRIVE_100MA = const(0)
13 APDS9960_LED_DRIVE_50MA = const(1)
14 APDS9960_LED_DRIVE_25MA = const(2)
15 APDS9960_LED_DRIVE_12_5MA = const(3)
16
17 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
18 print("Lite APDS-9960 Proximity test ")
19
20 apds9960=APDS9960LITE(i2c)
21 apds9960.prox.eLEDCurrent =APDS9960_LED_DRIVE_100MA
22 apds9960.prox.eProximityGain=APDS9960_PGAIN_8X
23
24 apds9960.prox.enableSensor()
25 apds9960.prox.setInterruptThreshold(high=10,low=0,persistence=7)
26 apds9960.prox.enableInterrupt()
27
28 ProxThPin=machine.Pin(0, machine.Pin.IN ,machine.Pin.PULL_UP)
29
30 sleep_ms(50)
31
32 while True:
33     sleep_ms(50)
34
35     if (ProxThPin.value() == 0):
36         print("proximity:", apds9960.prox.proximityLevel )
37         apds9960.prox.clearInterrupt()

```

9.1.2 Light Sensor Examples

9.1.2.1 Light sensing example

Ambient Light and RGB Color Sensing test program.

Listing 5: examples/als/simple_light_apds9960.py

```
1 import machine
2 from time import sleep_ms
3 from uPy_APDS9960.apds9960LITE import APDS9960LITE
4
5 #Init I2C Buss
6 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
7
8 apds9960=APDS9960LITE(i2c)          # Enable sensor
9 print("Enable light Sensor")
10
11 apds9960.als.enableSensor()         # Enable Light sensor
12 apds9960.als.eLightGain=3           # x64 gain
13 #apds9960.prox.enableProximity()
14 sleep_ms(50)
15 print("Clear Light level: ", apds9960.als.ambientLightLevel)
16 print("Red   Light level: " , apds9960.als.redLightLevel)
17 print("Green Light level: ", apds9960.als.greenLightLevel)
18 print("Blue  Light level: " , apds9960.als.blueLightLevel)
19
```

9.1.2.2 Ambient light IRQ

Ambient Light IRQ test program.

Listing 6: examples/als/simple_light_irq_apds9960.py

```
1 import machine
2 from time import sleep_ms
3 from uPy_APDS9960.apds9960LITE import APDS9960LITE
4
5 #Init I2C Buss
6 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
7
8 apds9960=APDS9960LITE(i2c)          # Enable sensor
9 apds9960.als.enableSensor()         # Enable Light sensor
10 apds9960.als.eLightGain=3           # x64 gain
11 apds9960.als.setInterruptThreshold(high=100,low=0,persistence=7)
12 apds9960.als.enableInterrupt(True)  # Enable interrupt
13 apds9960.als.clearInterrupt()       # Clear interrupt
14 sleep_ms(50)
15
16 IrqThPin=machine.Pin(0, machine.Pin.IN ,machine.Pin.PULL_UP)
17 sleep_ms(50)
18
19 while True:
20     sleep_ms(50)
21
22     if(IrqThPin.value()==0):
```

(continues on next page)

(continued from previous page)

```

23     print("Ambient light level:", apds9960.als.ambientLightLevel )
24     apds9960.als.clearInterrupt()

```

9.2 Debug

9.2.1 I2C Debug Example

Test program for testing the I2C connection the the ap9960

Listing 7: examples/debug/i2c_test.py

```

1  import machine
2  i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
3
4  print('Scan i2c bus...')
5  devices = i2c.scan()
6
7  if len(devices) == 0:
8      print("No i2c device !")
9  else:
10     print('i2c devices found:', len(devices))
11
12     for device in devices:
13         print("Decimal address: ", device, " | Hexa address: ", hex(device))
14
15         if(device==0x39): # APDS9960 Address = 0x39
16             deviceID=i2c.readfrom_mem(devices[0],0x92, 1) #Get deviceID
17             print("Found ADPS9960: Device ID: ", deviceID)

```

9.3 Thonny IDE Tutorial

This is a short visual tutorial on how to use the [Thonny IDE](#) to flash the ESP8266 chip and upload and test the uPy_APD9960 library

9.3.1 Download and install

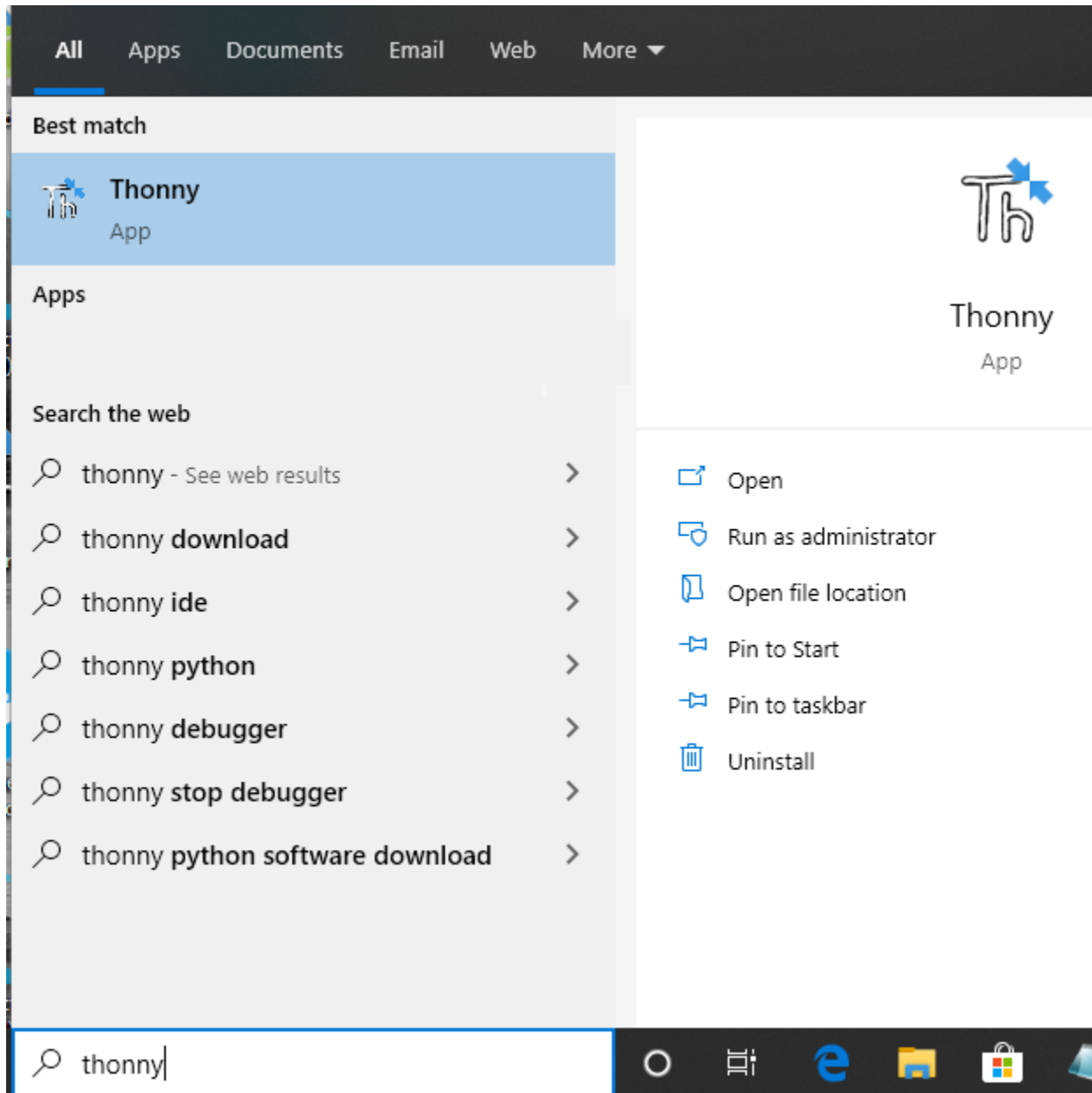
The latest version of tonny can be found at [thonny.org](#)

In this tutorial we uses the [windows version](#) and install Thonny on your computer

9.3.1.1 Download up Micropython (ESP8266) dev enviroment

Download latest [MicroPython firmware](#) for ESP8266

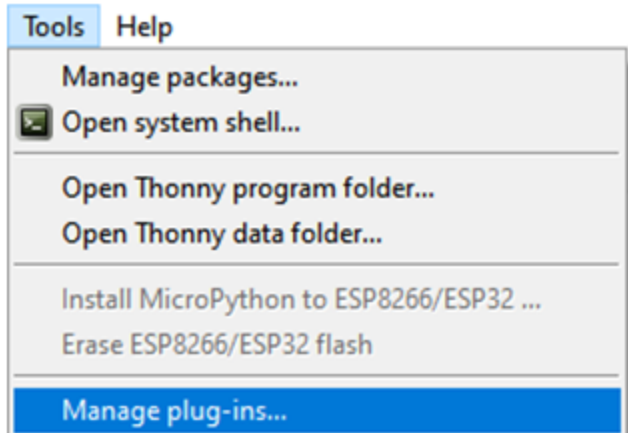
9.3.2 Start Thonny



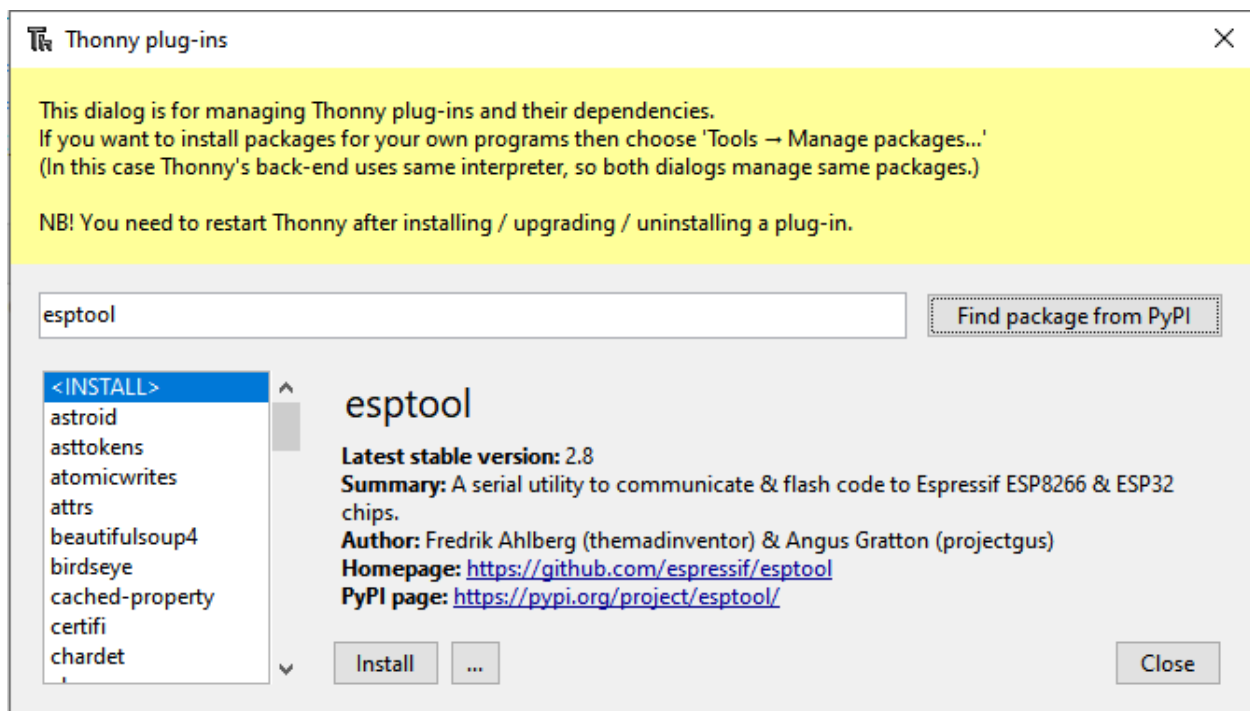
9.3.3 Flash new firmware

9.3.3.1 Installing esptool.py

From the menu “Tools” select “Manage Plut-ins...”



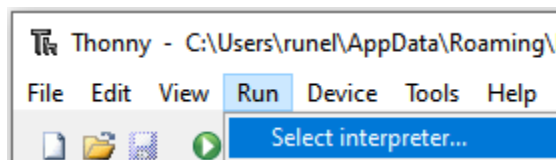
In the text field enter “esptool” and click the button “Find packages from PyPI”



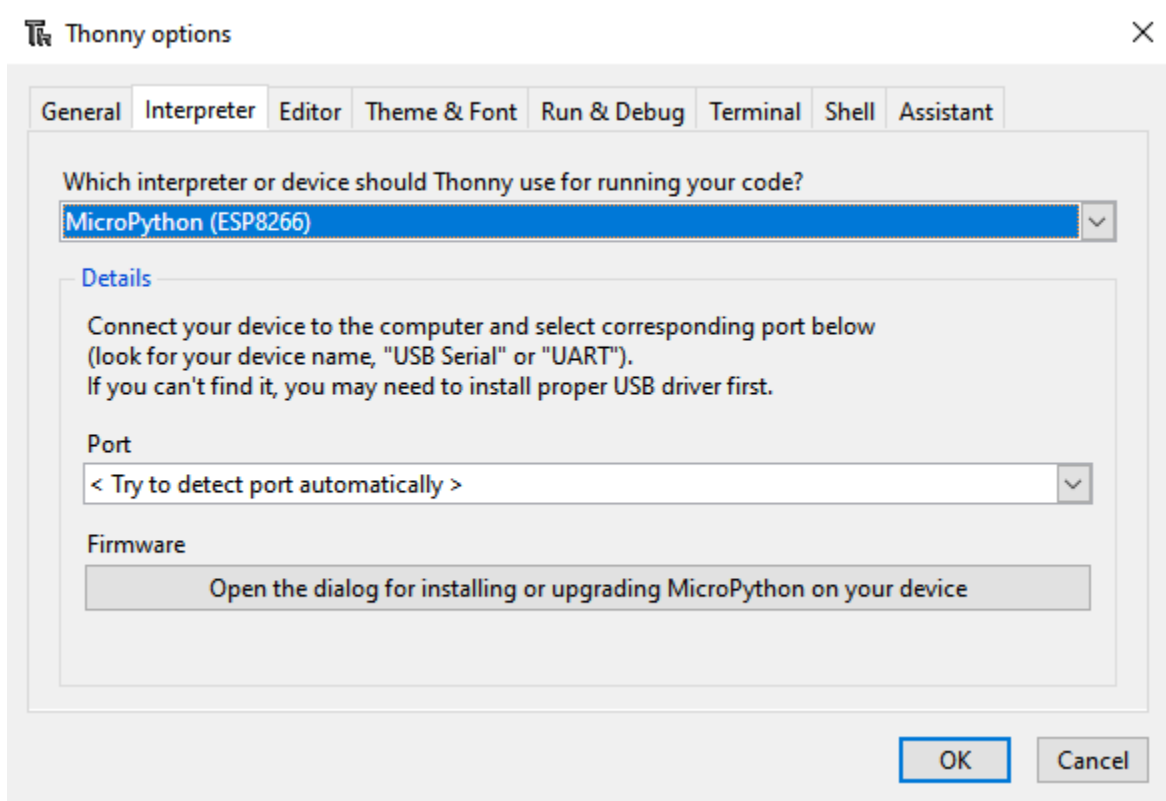
Click the “Install” button to finish the esptool installation

9.3.3.2 Setting up Micropython (ESP8266) dev environment

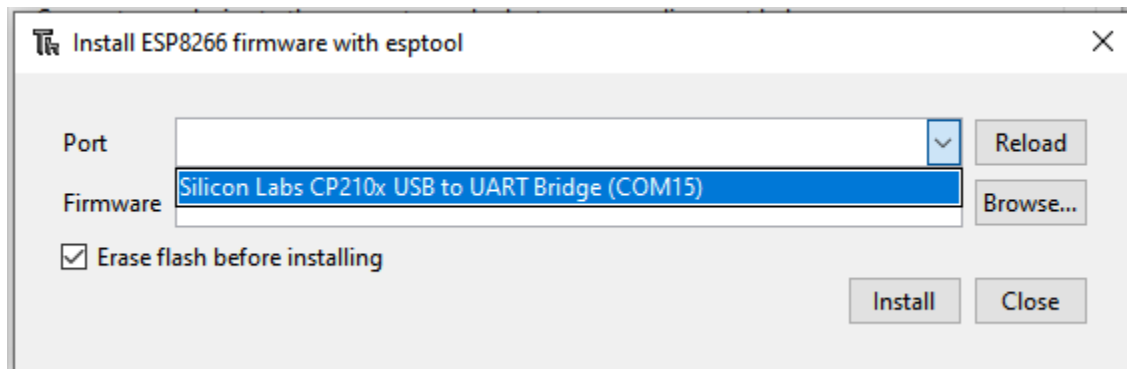
From the menu “Run” select “Select interpreter...”



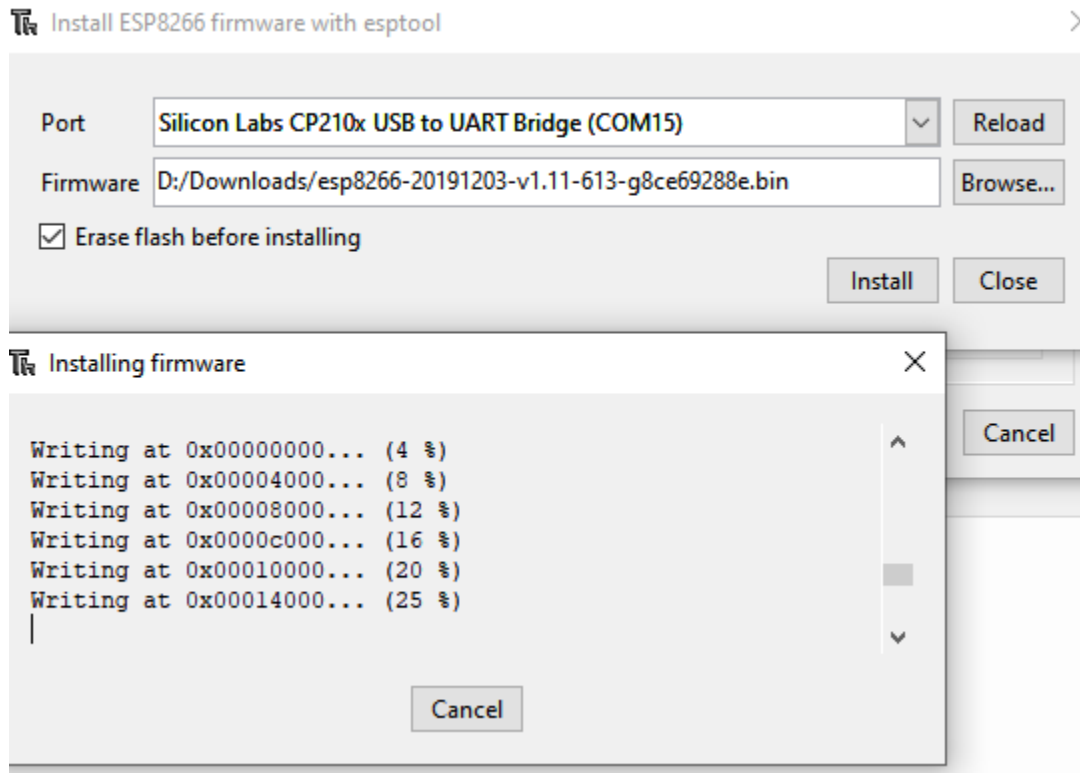
The ESP8266 firmware install/upgrade dialog is shown



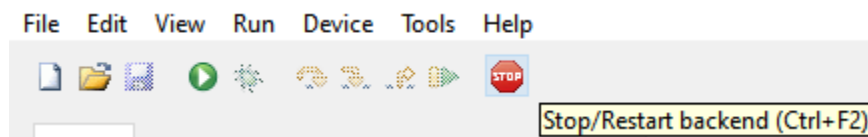
Make sure that the ESP8266 development kit is connected. Select the “port dropdown” arrow to select the serial port for flashing the ESP8266.



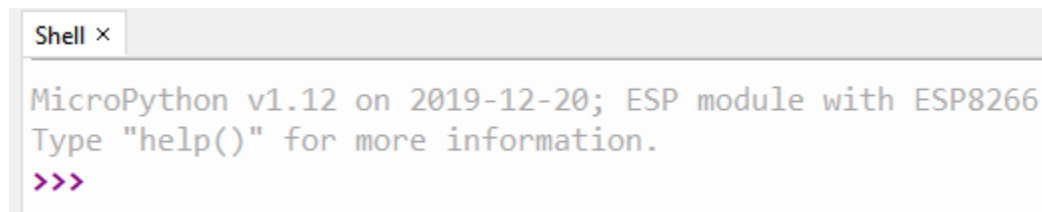
In the “Firmware” text field select the .bin file that was [downloaded](#) and click the install button



Click the stop icon to reset and connect to the ESP8266 board



Now you should be up and running as shown in the thonny shell windows

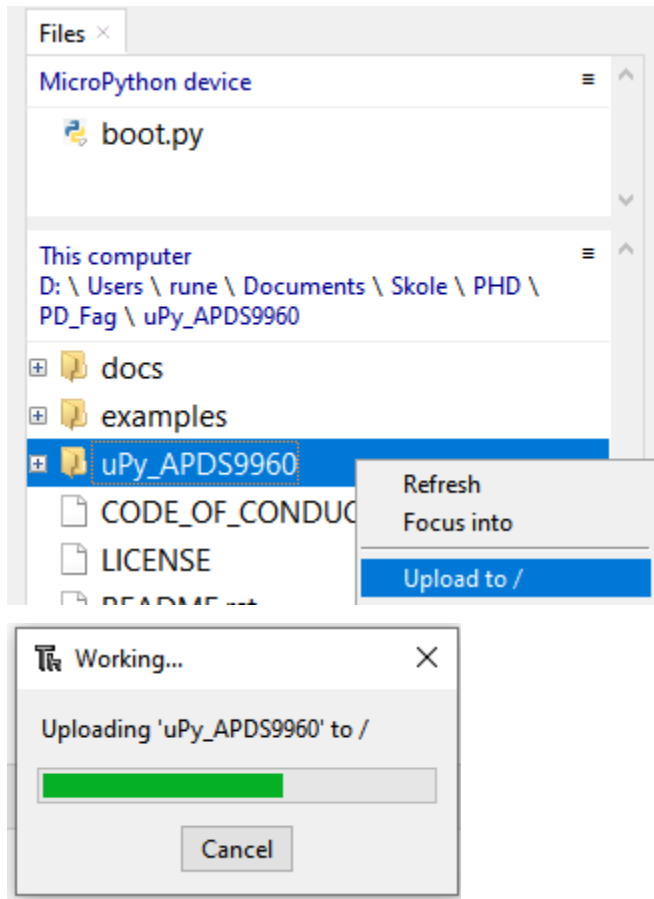


9.3.4 Running the examples

Start by uploading the uPy_APDS9960 library.

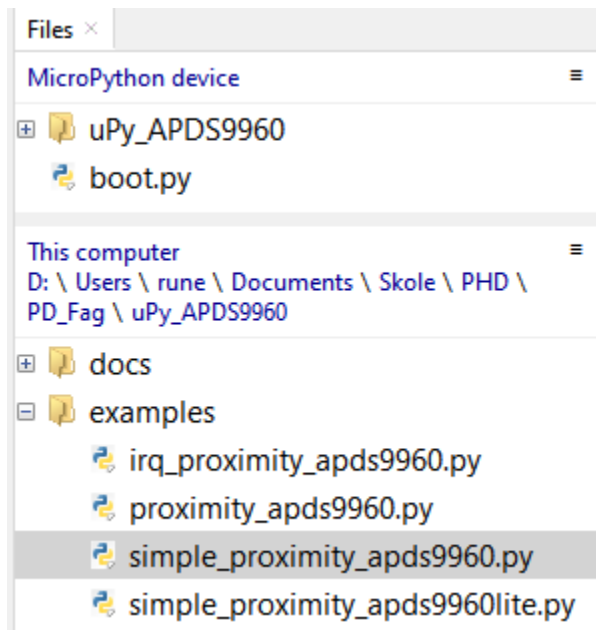
9.3.4.1 Uploading uPy_APDS9960

From the files windows under “This computer” right click on the folder “uPy_APDS9960” and from the dropdown menu select “Upload to /”



9.3.4.2 Running a example program

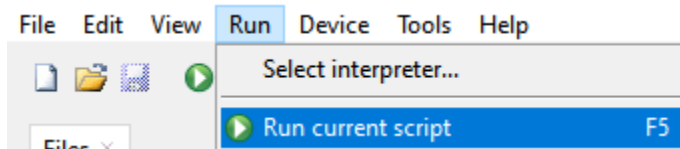
From the files windows under “This computer” click on the ‘+’ sign infront of the folder “examples” to expand it.



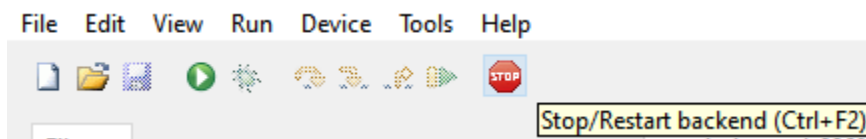
Double click on the file `simple_proximity_apds9960.py` and it will be open the the editor

```
simple_proximity_apds9960.py ×
1 import machine
2 from time import sleep_ms
3 from uPy_APDS9960.APDS9960 import APDS9960
4
5 #Init I2C Buss
6 i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
```

You are now ready to run the program entering F5 or selecting the menu “Run” and “Run current script”

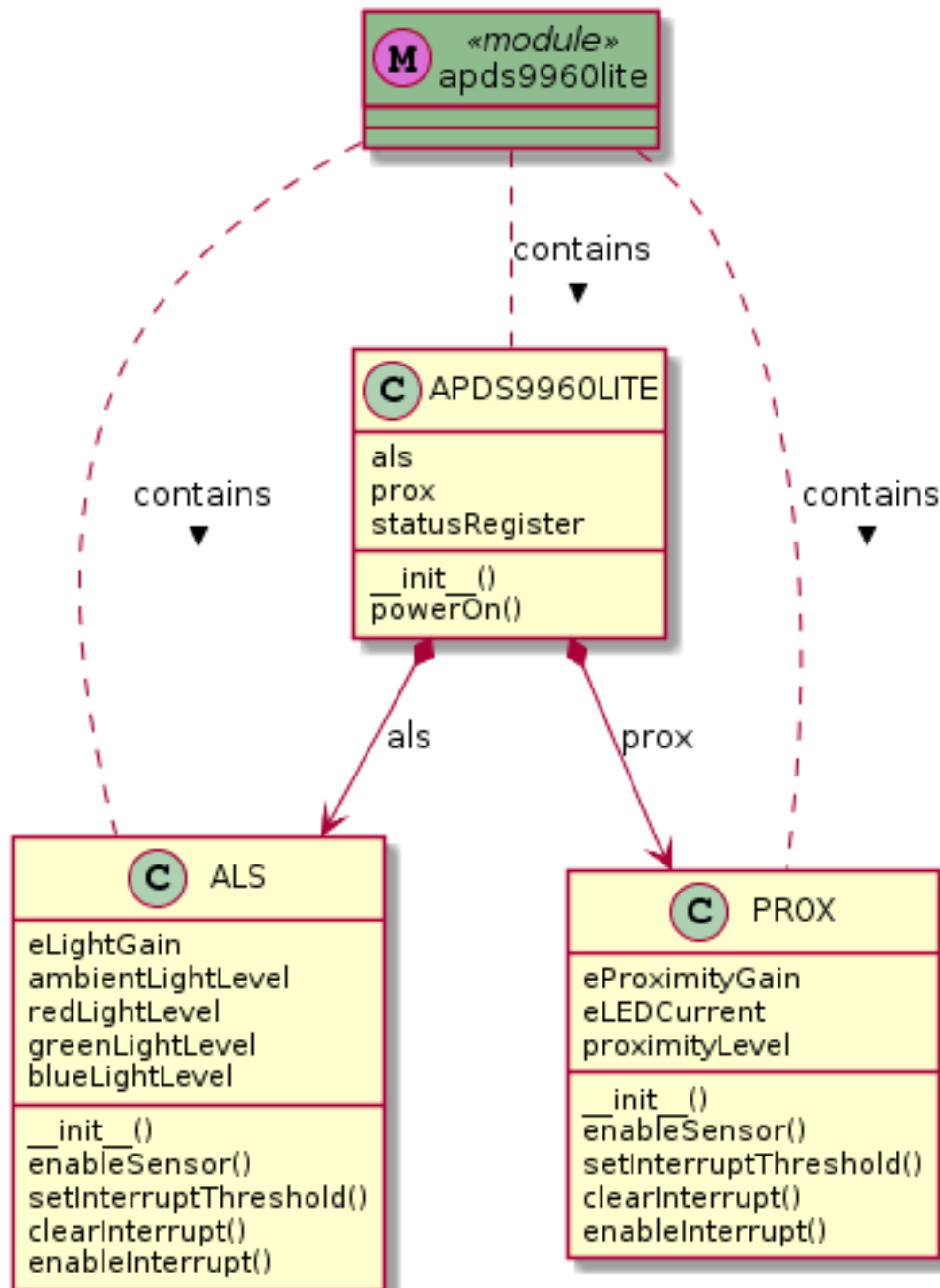


Click the stop icon to stop the program and return the command prompt



Have fun :)

9.4 uPy_APDS9960 Module



CHAPTER 10

Indices and tables

- `genindex`
- `modindex`
- `search`