Connected Humber Hardware Meetup MicroPython on Ultra-Cheap Devices

Version 1.0 Rob Miles

Devices

- ESP8266 – very cheap: 2 pounds. WiFi and good performance. Not many pins.
- ESP32 – slightly more expensive: 5 pounds or so. WiFi, Bluetooth, Dual Core, lots of pins that are easy to configure

MicroPython

It’s a free download from here. There are versions for ESP32 and ESP8266

```
https://micropython.org/download
```

Note: Don’t use the latest ESP32 version (1.12) if you want to use Azure IoT Hub because the secure socket support (which Azure needs) is broken in 1.12. Use 1.11 instead.

Use the esptool to install MicroPython on devices. You can get the tool from here. It’s a Python program, so you’ll need Python too.

```
https://github.com/espressif/esptool
```

The commands to put an image onto an ESP 32 device is:

```
esptool.py --chip esp32 --port com4 erase_flash
esptool.py --chip esp32 --port com4 --baud 460800 write_flash -z 0x1000 file.bin
```

You’ll have to select the serial (com) port yourself and download the binary file (see the highlighted elements). If you are using an Apple or Linux device the port will look different – something like /dev/ttyUSB0.

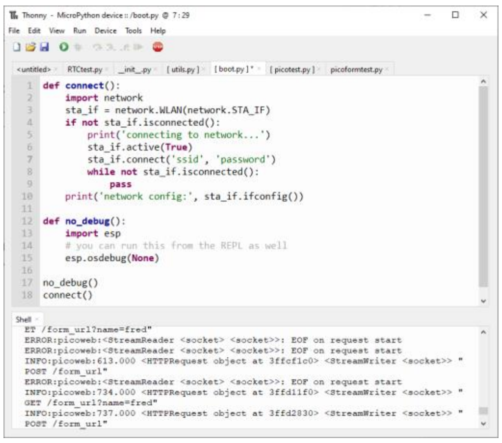
Once you have installed MicroPython you can talk to it via a serial terminal. It supports the REPL prompt so you can type in Python code and run it. But we advise you to take a look at Thonny. It’s a Python IDE (Integrated Development Environment) that is available for Windows, MAC and Linux.

Thonny

Get Thonny from here. The download links are at the top right of this page.

https://thonny.org/

You can write and debug Python programs on your computer, but you can also download and run Python programs in an attached MicroPython device.



Thonny - MicroPython device :: /boot.py @ 7:29
File Edit View Run Device Tools Help
<untitled> RTCtest.py _init_.py [utils.py] [boot.py]* [picotest.py] picoformtest.py
Shell

```
def connect():
    import network
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to network...')
        sta_if.active(True)
        sta_if.connect('ssid', 'password')
        while not sta_if.isconnected():
            pass
    print('network config:', sta_if.ifconfig())

def no_debug():
    import esp
    # you can run this from the REPL as well
    esp.osdebug(None)

no_debug()
connect()
```

```
ET /form_url?name=fred"
ERROR:picoweb:<StreamReader <socket> <socket>>: EOF on request start
ERROR:picoweb:<StreamReader <socket> <socket>>: EOF on request start
INFO:picoweb:613.000 <HTTPRequest object at 3ffcf1c0> <StreamWriter <socket>> "
POST /form_url"
ERROR:picoweb:<StreamReader <socket> <socket>>: EOF on request start
INFO:picoweb:734.000 <HTTPRequest object at 3ffd11f0> <StreamWriter <socket>> "
GET /form_url?name=fred"
INFO:picoweb:737.000 <HTTPRequest object at 3ffd2830> <StreamWriter <socket>> "
POST /form_url"
```

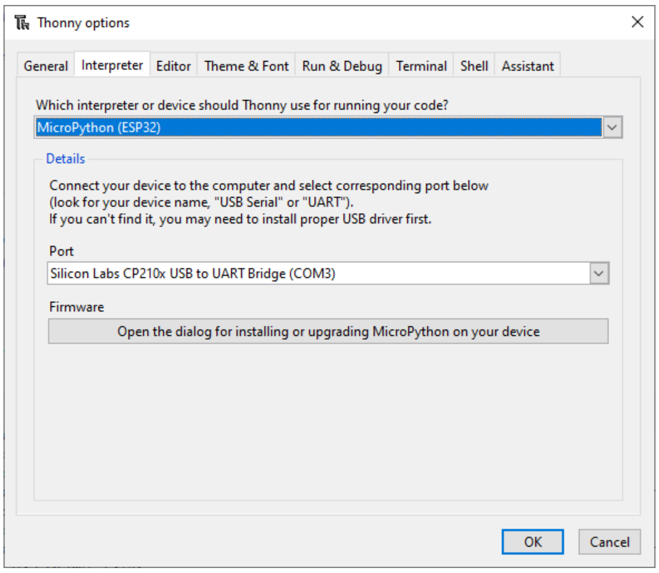
This is Thonny in action. I’m editing the Python file boot.py which runs in the device when it starts. This version of boot.py connects the device to the local WiFi. Note that need to add your ssid and password to this program.

These will be stored in clear text inside your device. So if anyone gets hold of the hardware they could read them back.

You use the Shell window to talk to Python running in your device.

Connect Thonny to an attached device

Select Tools>Options to open the options menu. Then select the Interpreter tab.



Which interpreter or device should Thonny use for running your code?
MicroPython (ESP32)

Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port
Silicon Labs CP210x USB to UART Bridge (COM3)

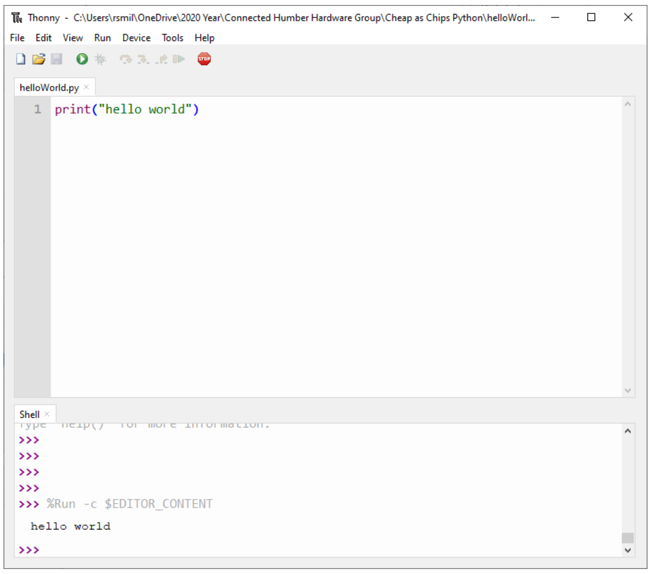
Firmware

Here you tell Thonny that you are using the MicroPython interpreter and specify which serial port it is connected to.

You can also use this dialog to install MicroPython on devices, but to do this you have to configure Thonny and tell it where the espytool program is on your system.

Once you have made your settings you just need to click OK to save them and they will be used each time you use Thonny.

Run a Python program from Thonny



Thonny - C:\Users\rsmil\OneDrive\2020 Year\Connected Humber Hardware Group\Cheap as Chips Python\helloWorl...

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

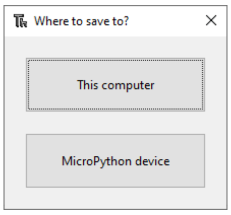
```
Type "help()" for more information.
>>>
>>>
>>>
>>>
>>> %Run -c $EDITOR_CONTENT
 hello world
>>>
```

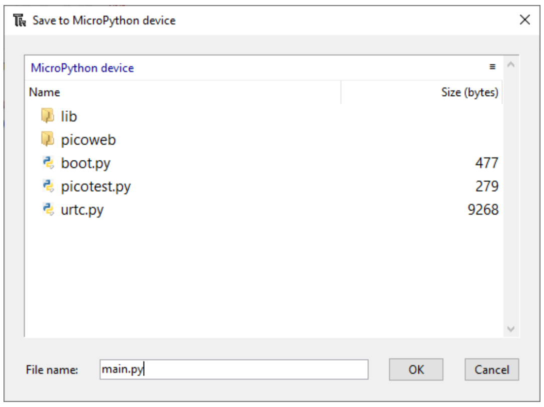
Once you have connected Thonny to a device you can run a Python program in the device by just opening the file and pressing the green Run button. This will cause the program to be downloaded into the device and start running.

You can use the File>Save command to save a local copy of the file or to save it onto your device:

Any printed output from the file will appear in the shell window at the bottom. You can type in input to your program here too.

Make Python programs run when your device is powered on





Name	Size (bytes)
lib	
picoweb	
boot.py	477
picotest.py	279
urtc.py	9268

File name: main.py

If you select File>Save you will be asked where you want to save the currently selected program file.

You can save to your computer or to the MicroPython device.

boot.py runs when your machine starts

main.py runs when boot.py has finished.

If a programming session gets stuck you can reset your Python device by clicking the red STOP button in the user interface.

What do you want to do next?

Hardware Control

MicroPython can control hardware pins on your device. Take a look here:

```
http://docs.micropython.org/en/latest/library/machine.Pin.html
```

Coloured LEDS

MicroPython supports NeoPixels for coloured lights. Take a look here:

```
http://docs.micropython.org/en/latest/esp8266/tutorial/neopixel.html
```

MQTT Support

You can use a MicroPython device to talk MQTT. There's a great description here:

```
https://boneskull.com/micropython-on-esp32-part-2/
```

Azure IOT Hub Support

MicroPython can talk to an Azure IOT Hub. Take a look here:

```
http://blogs.recneps.org/post/Connecting-the-ESP-8266-to-Azure-IoT-Hub-using-MQTT-and-MicroPython
```

Web Server

You can use the device as a web server and write Python code that responds to user web requests. Start here:

```
https://github.com/pfalcon/picoweb
```

Using a Real Time Clock Device

The DS3231 device is a fine Real Time Clock. You can find code to use it here:

```
https://github.com/adafruit/Adafruit-uRTC
```

BME280 environmental sensor

If you want to sense temperature, humidity and air pressure you can use the BME280. There's a driver here:

```
https://github.com/SebastianRoll/mpy_bme280_esp8266
```

Version 1.0 Rob Miles March 2020