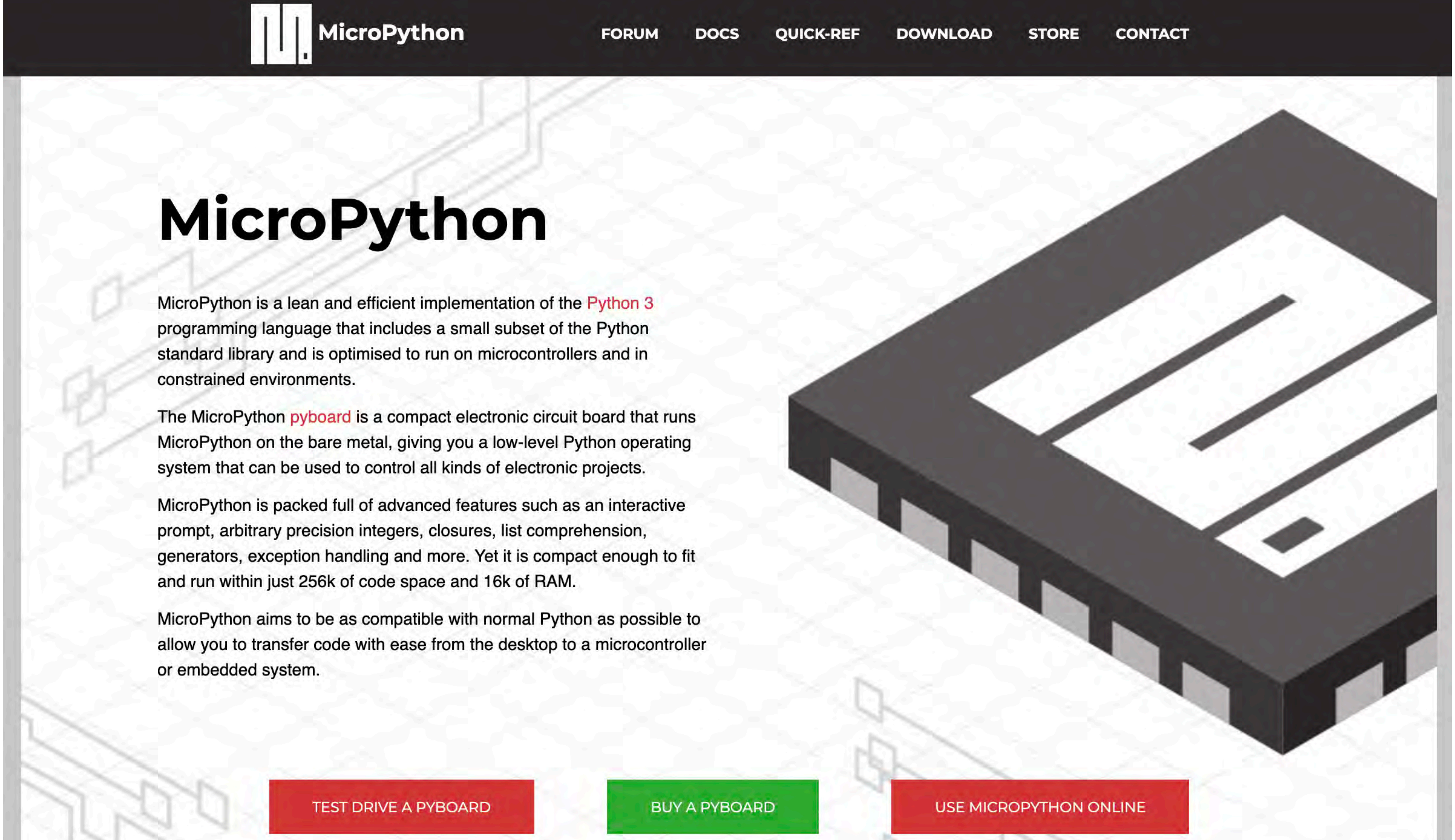
Adventures with: ESP32, MicroPython, Deepsleep and other gotchas

Stuff I wish someone had told me when I started down this path

Andy Harris, 8th December 2020



MicroPython

MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments.

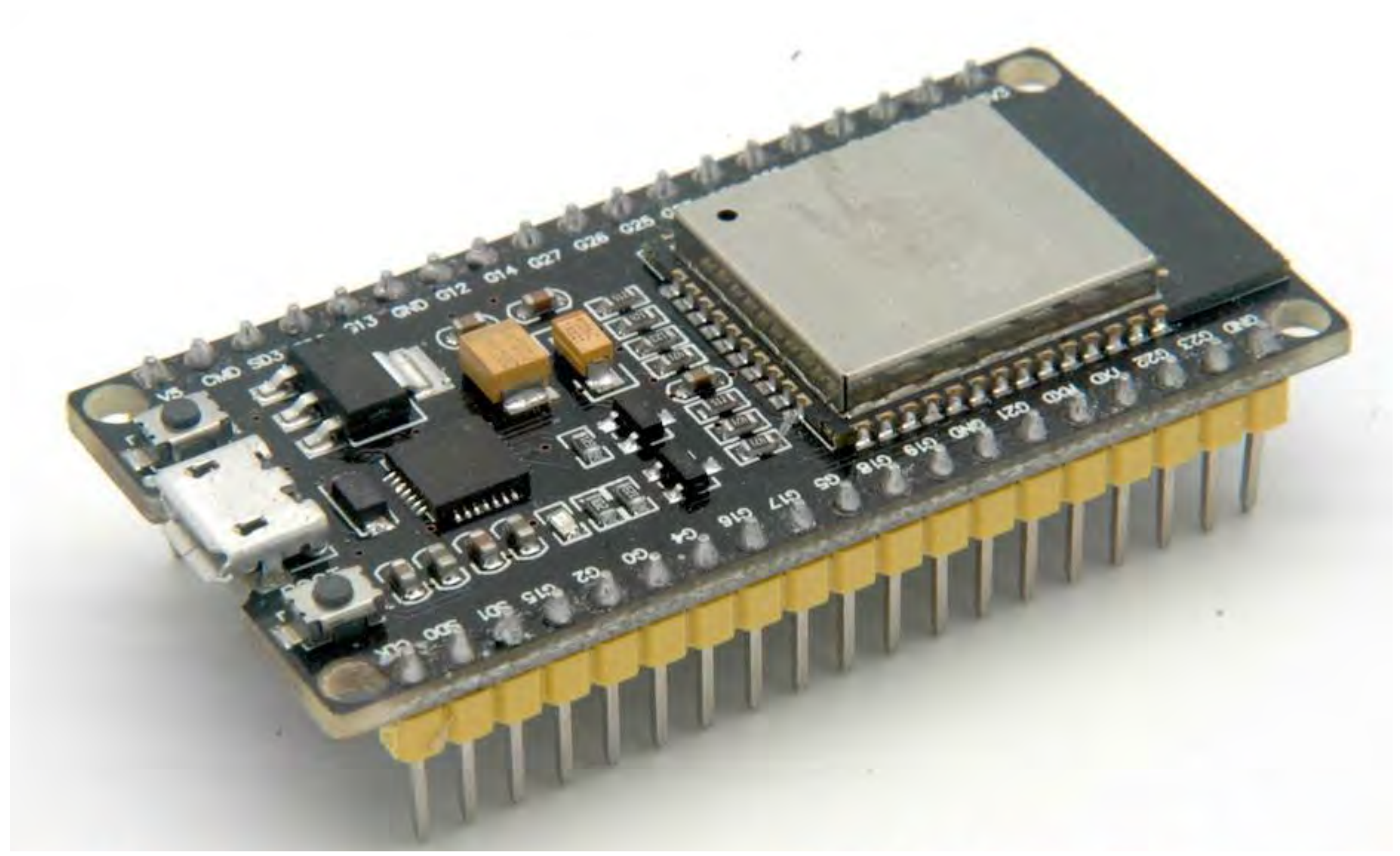
The MicroPython pyboard is a compact electronic circuit board that runs MicroPython on the bare metal, giving you a low-level Python operating system that can be used to control all kinds of electronic projects.

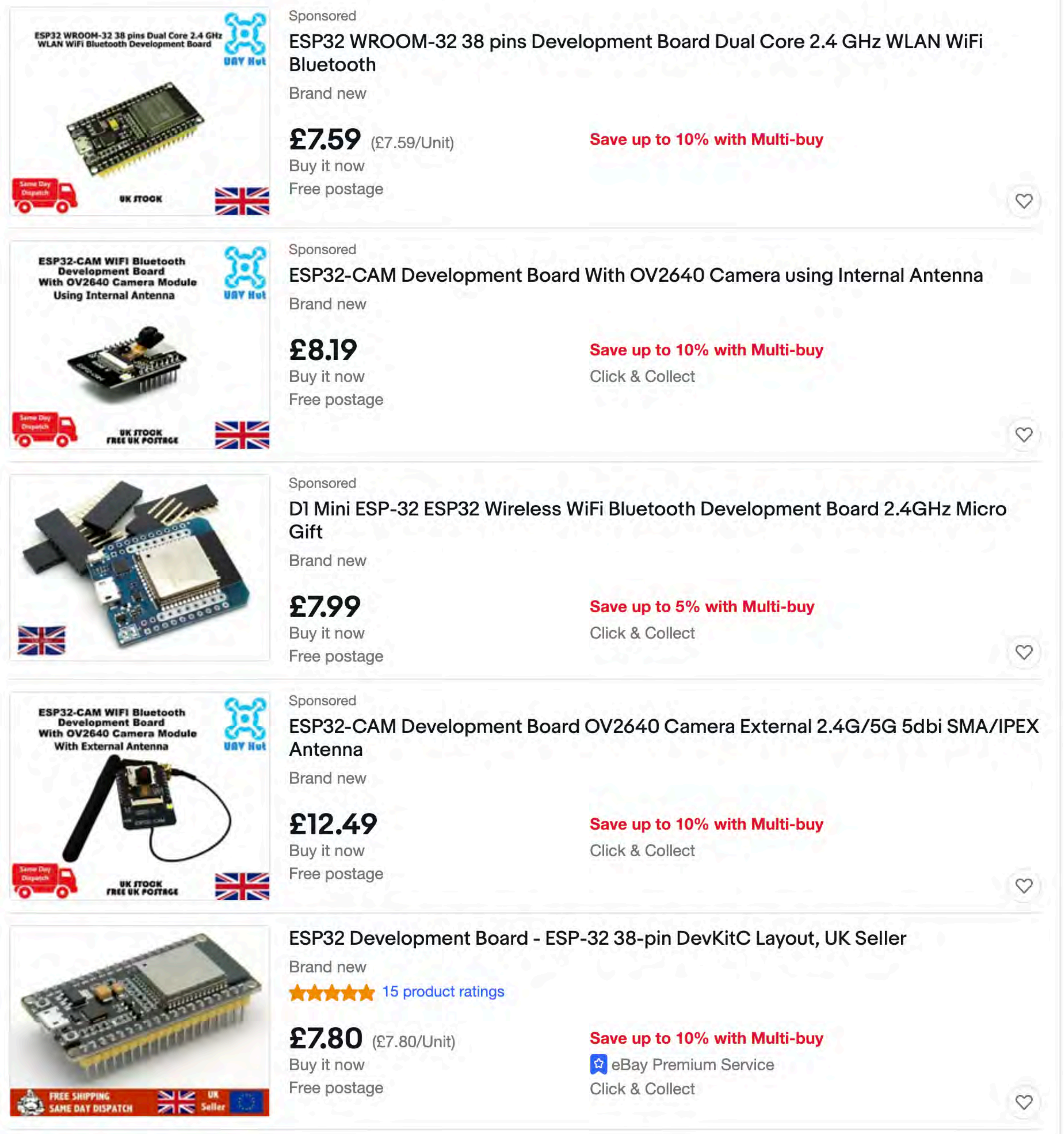
MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.

The ESP32

Double cheap





Sponsored

ESP32 WROOM-32 38 pins Development Board Dual Core 2.4 GHz WLAN WiFi Bluetooth

Brand new

£7.59 (£7.59/Unit)

Save up to 10% with Multi-buy

Buy it now

Free postage

Sponsored

ESP32-CAM Development Board With OV2640 Camera using Internal Antenna

Brand new

£8.19

Save up to 10% with Multi-buy

Buy it now

Click & Collect

Free postage

Sponsored

D1 Mini ESP-32 ESP32 Wireless WiFi Bluetooth Development Board 2.4GHz Micro Gift

Brand new

£7.99

Save up to 5% with Multi-buy

Buy it now

Click & Collect

Free postage

Sponsored

ESP32-CAM Development Board OV2640 Camera External 2.4G/5G 5dbi SMA/IPEX Antenna

Brand new

£12.49

Save up to 10% with Multi-buy

Buy it now

Click & Collect

Free postage

ESP32 Development Board - ESP-32 38-pin DevKitC Layout, UK Seller

Brand new

★★★★★ 15 product ratings

£7.80 (£7.80/Unit)

Save up to 10% with Multi-buy

Buy it now

eBay Premium Service

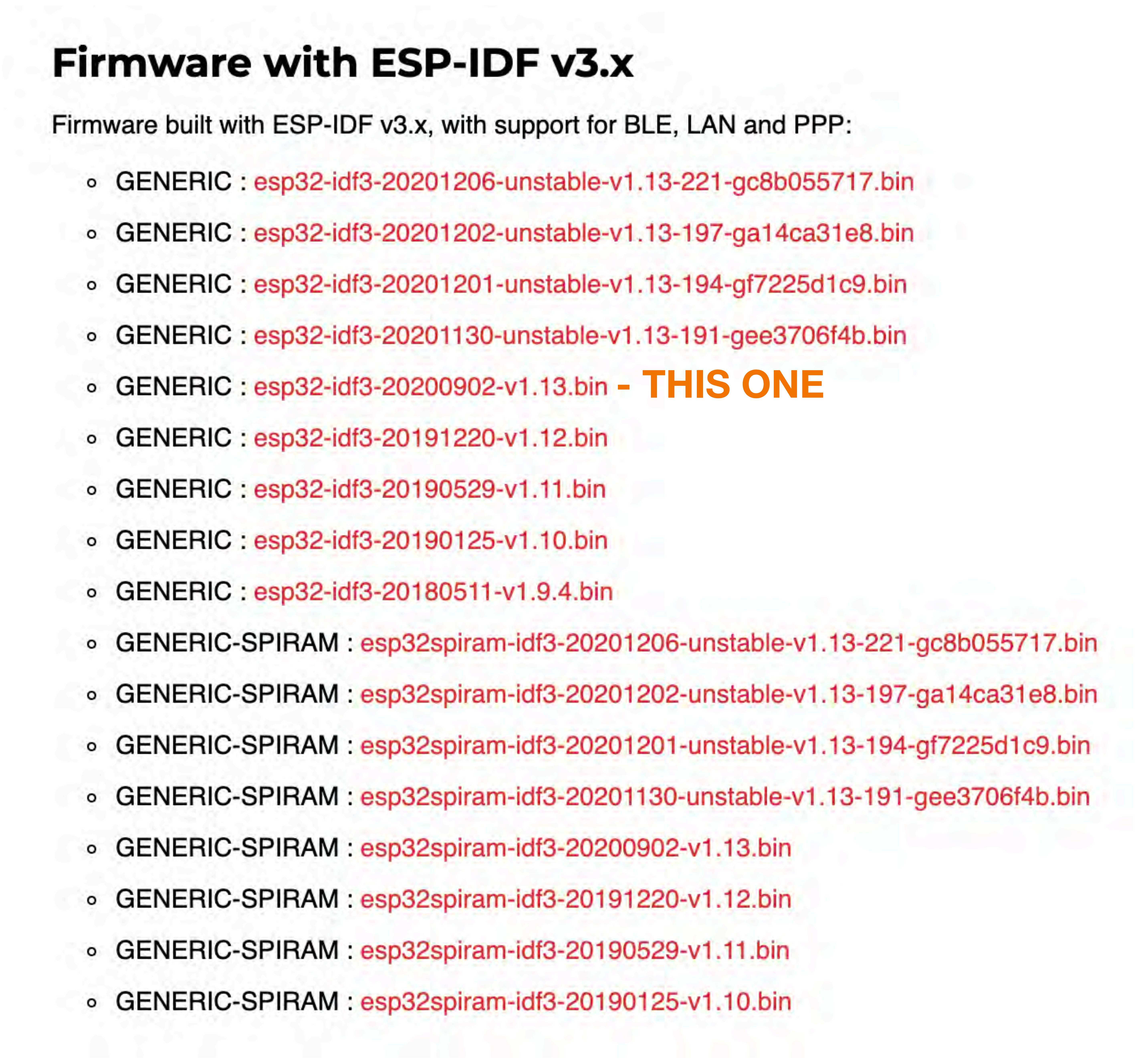
Free postage

Click & Collect

WHAT?

Python but smaller

- Python 3.4 Syntax
- Python 3.5 asyncio and await



Firmware with ESP-IDF v3.x

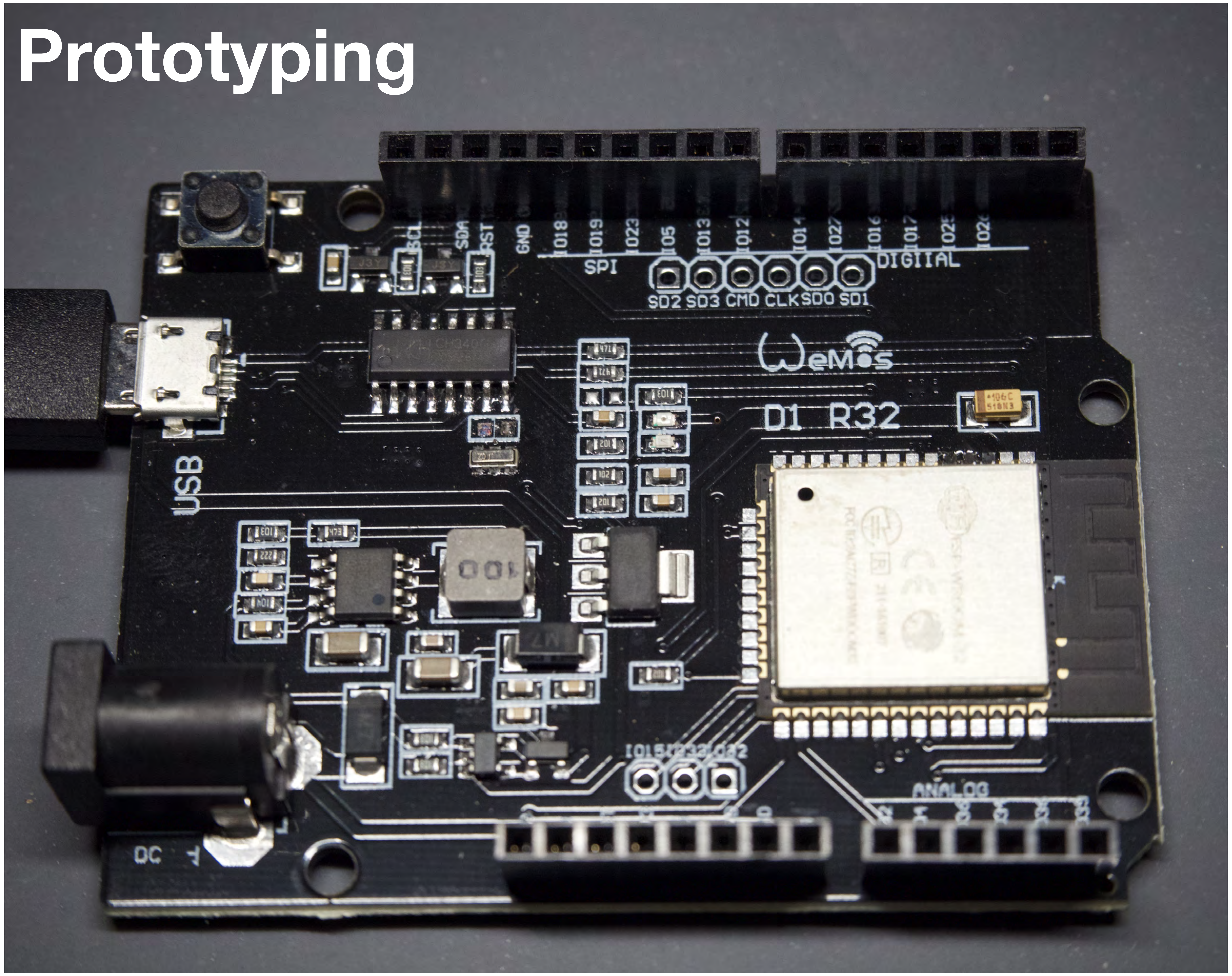
Firmware built with ESP-IDF v3.x, with support for BLE, LAN and PPP:

- GENERIC : esp32-idf3-20201206-unstable-v1.13-221-gc8b055717.bin
- GENERIC : esp32-idf3-20201202-unstable-v1.13-197-ga14ca31e8.bin
- GENERIC : esp32-idf3-20201201-unstable-v1.13-194-gf7225d1c9.bin
- GENERIC : esp32-idf3-20201130-unstable-v1.13-191-gee3706f4b.bin
- GENERIC : esp32-idf3-20200902-v1.13.bin - THIS ONE
- GENERIC : esp32-idf3-20191220-v1.12.bin
- GENERIC : esp32-idf3-20190529-v1.11.bin
- GENERIC : esp32-idf3-20190125-v1.10.bin
- GENERIC : esp32-idf3-20180511-v1.9.4.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20201206-unstable-v1.13-221-gc8b055717.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20201202-unstable-v1.13-197-ga14ca31e8.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20201201-unstable-v1.13-194-gf7225d1c9.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20201130-unstable-v1.13-191-gee3706f4b.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20200902-v1.13.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20191220-v1.12.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20190529-v1.11.bin
- GENERIC-SPIRAM : esp32spiram-idf3-20190125-v1.10.bin

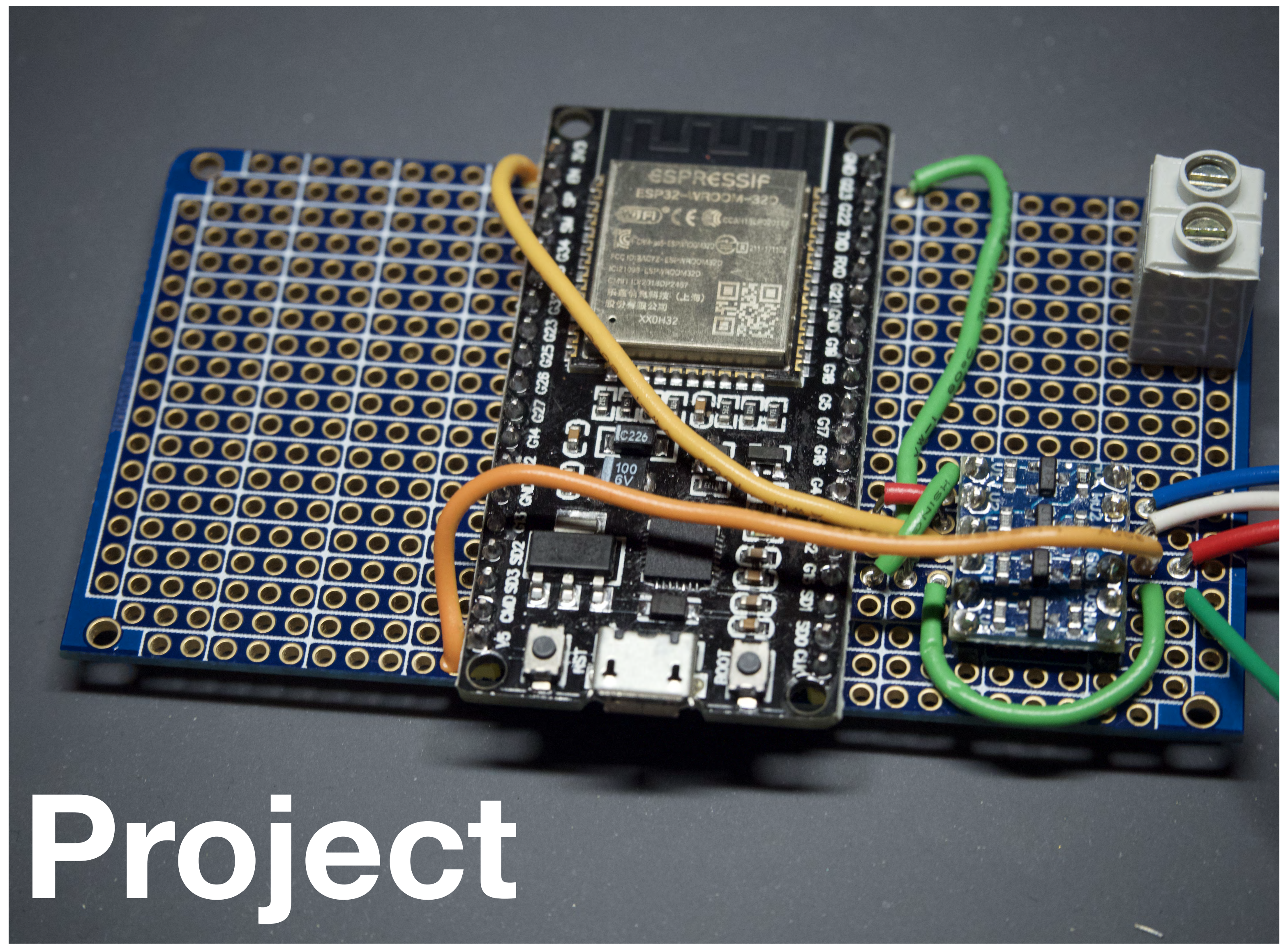
UPs and DOWNs

Weighing in against Arduino style C++

- Disadvantages:
    - Not so granular control over the hardware
    - Speed (unless you know the tricks)
- Advantages
    - Get stuff done quicker - many libraries, especially networking and data formats
    - Multi-threading is easy
    - Easy debugging
    - Field updates over the net


Prototyping
WeMos
D1 R32

Dev Board
CAUTION


Project
ESPRESSIF

Garden Light

Solar Controls

Bluetooth Gate

Solar Lights

Winter Mode

- One Light (Two in Summer)
- Sleep for 8 hours @11.1V
- Evening off @11.3V
- DeepSleep during off periods

ESP32

Built-in Peripherals - what you get for your £7

- Wifi and Bluetooth
- Soft and Hard (SPI, I2C), One wire, SD Flash, UARTS
- Hall Effect, Internal Temperature, Capacitive Touch
- 4 Timers, PWM, ADC
- Real Time Clock & RMT for precision pulses
- Deepsleep and ultra low energy processor

ESP32-PoE-ISO
POE VERSION

Getting Started

Installing MicroPython

Getting out of trouble (De-Bricking)

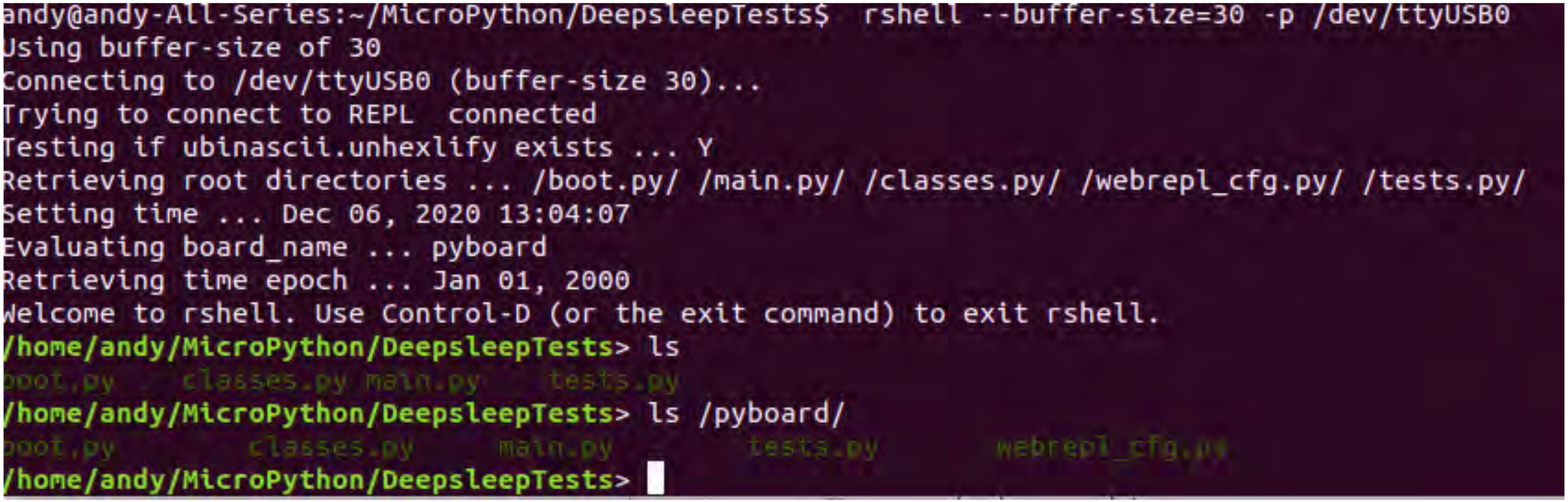
Download and install

Its simple …

- esptool --chip esp32 --port /dev/ttyUSB0 write_flash -fm dio -z 0x1000 esp32-idf3-20200902-v1.13.bin
- rshell --buffer-size=30 -p /dev/ttyUSB0
- Works perfectly on a new ‘out of the bag ESP32’

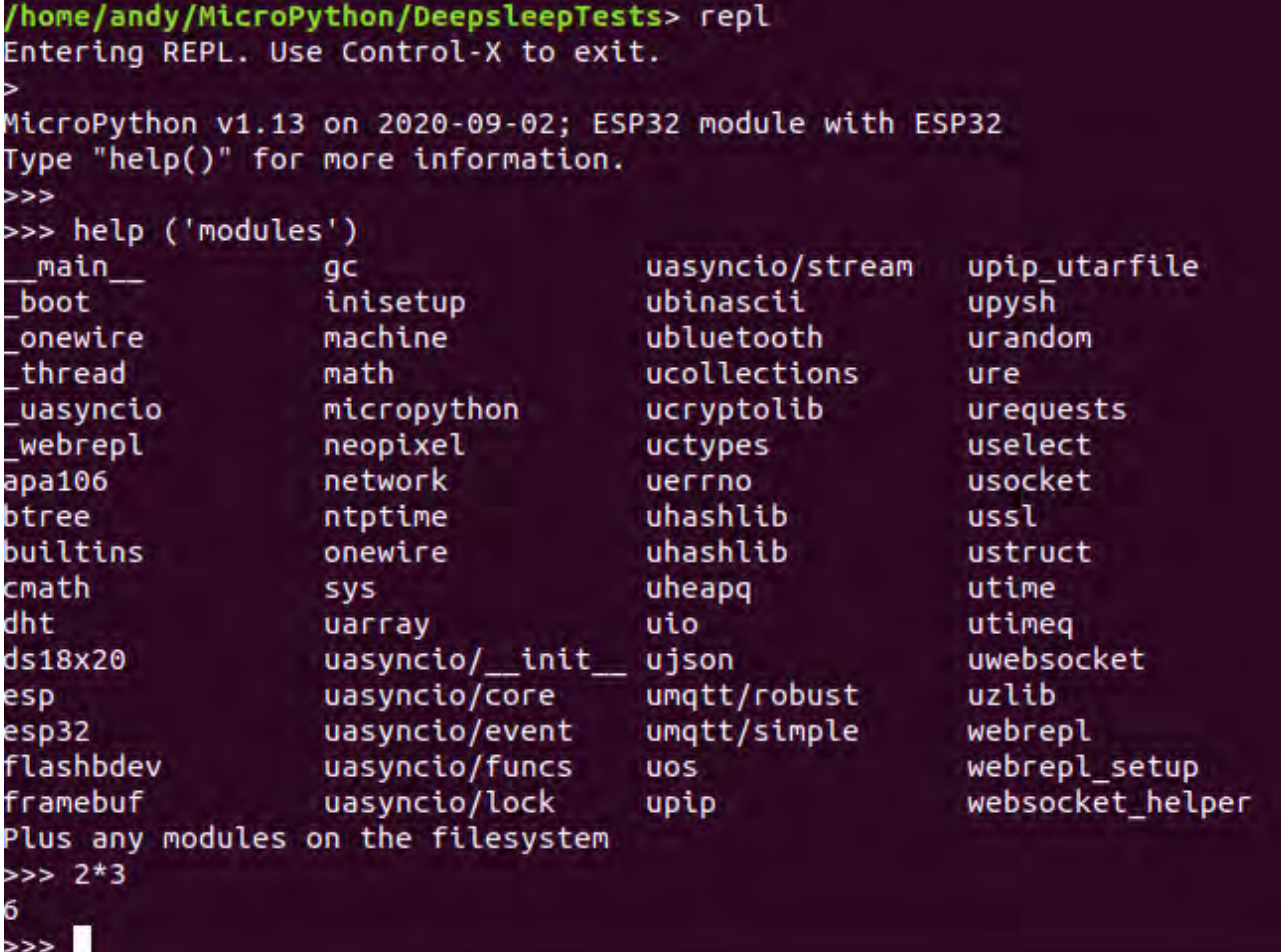
RSHELL

Between File System and REPL



```
andy@andy-All-Series:~/MicroPython/DeepsleepTests$  rshell --buffer-size=30 -p /dev/ttyUSB0
Using buffer-size of 30
Connecting to /dev/ttyUSB0 (buffer-size 30)...
Trying to connect to REPL  connected
Testing if ubinascii.unhexlify exists ... Y
Retrieving root directories ... /boot.py/ /main.py/ /classes.py/ /webrepl_cfg.py/ /tests.py/
Setting time ... Dec 06, 2020 13:04:07
Evaluating board_name ... pyboard
Retrieving time epoch ... Jan 01, 2000
Welcome to rshell. Use Control-D (or the exit command) to exit rshell.
/home/andy/MicroPython/DeepsleepTests> ls
boot.py    classes.py main.py    tests.py
/home/andy/MicroPython/DeepsleepTests> ls /pyboard/
boot.py         classes.py      main.py         tests.py        webrepl_cfg.py
/home/andy/MicroPython/DeepsleepTests>
```

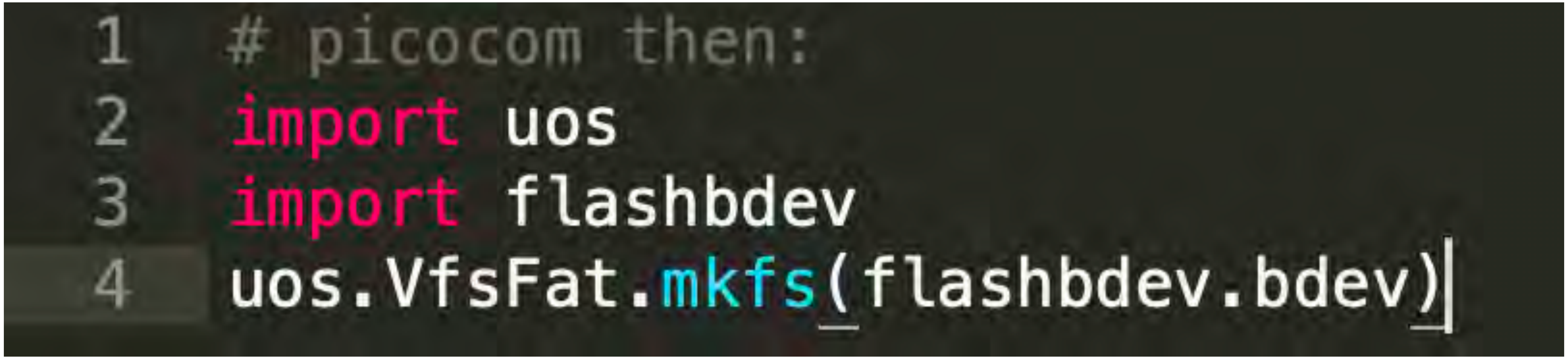
REPL



```
/home/andy/MicroPython/DeepsleepTests> repl
Entering REPL. Use Control-X to exit.
>
MicroPython v1.13 on 2020-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
>>> help ('modules')
__main__          gc                uasyncio/stream   upip_utarfile
_boot             inisetup          ubinascii         upysh
_onewire          machine           ubluetooth        urandom
_thread           math              ucollections      ure
_uasyncio         micropython       ucryptolib        urequests
_webrepl          neopixel          uctypes           uselect
apa106            network           uerrno            usocket
btree             ntptime           uhashlib          ussl
builtins          onewire           uhashlib          ustruct
cmath             sys               uheapq            utime
dht               uarray            uio               utimeq
ds18x20           uasyncio/__init__ ujson             uwebsocket
esp               uasyncio/core     umqtt/robust      uzlib
esp32             uasyncio/event    umqtt/simple      webrepl
flashbdev         uasyncio/funcs    uos               webrepl_setup
framebuf          uasyncio/lock     upip              websocket_helper
Plus any modules on the filesystem
>>> 2*3
6
>>> 
```

BRICKING IT

Recovery

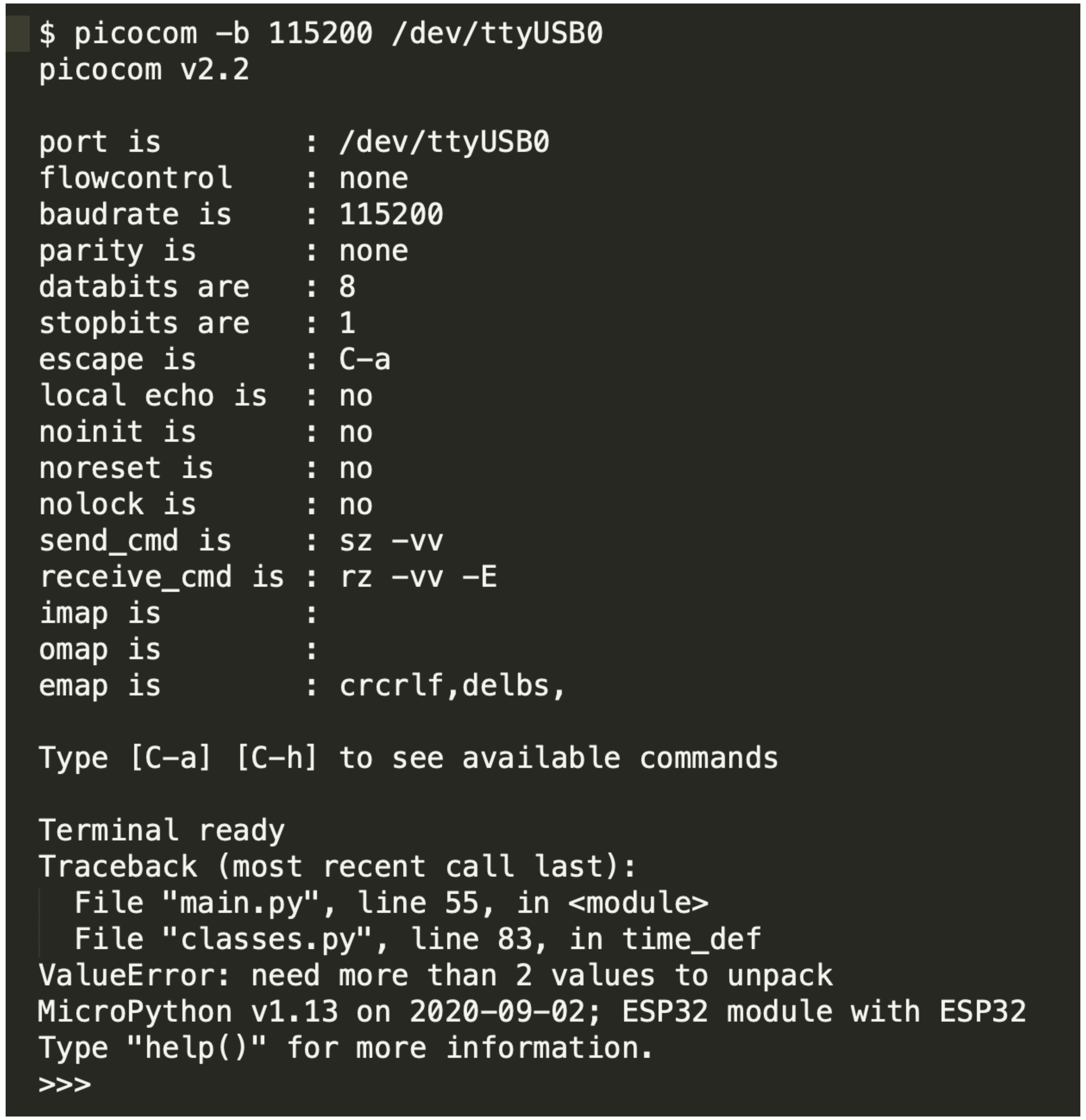


```
# picocom then:
import uos
import flashbdev
uos.VfsFat.mkfs(flashbdev.bdev)
```

BRICKING IT

picocomm

Control-C Interrupts



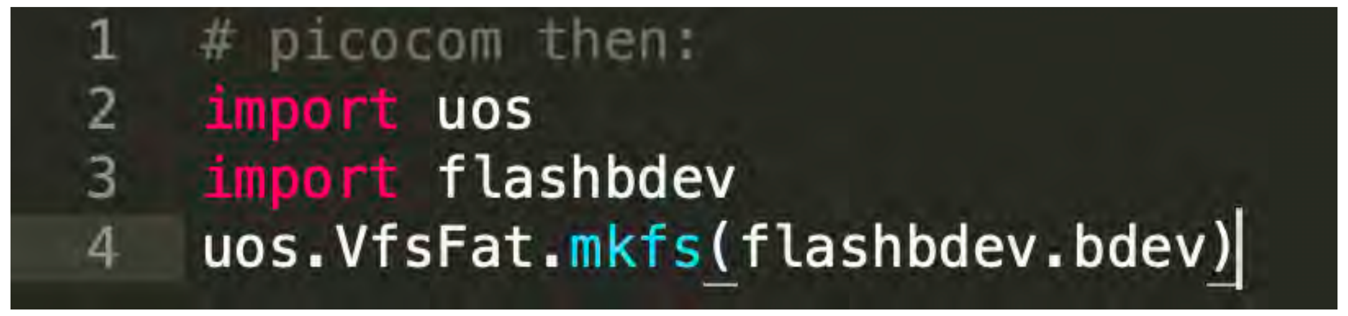
```
$ picocom -b 115200 /dev/ttyUSB0
picocom v2.2

port is         : /dev/ttyUSB0
flowcontrol     : none
baudrate is     : 115200
parity is       : none
databits are    : 8
stopbits are    : 1
escape is       : C-a
local echo is   : no
noinit is       : no
noreset is      : no
nolock is       : no
send_cmd is     : sz -vv
receive_cmd is  : rz -vv -E
imap is         :
omap is         :
emap is         : crcrlf,delbs,

Type [C-a] [C-h] to see available commands

Terminal ready
Traceback (most recent call last):
  File "main.py", line 55, in <module>
  File "classes.py", line 83, in time_def
ValueError: need more than 2 values to unpack
MicroPython v1.13 on 2020-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
```

Commands

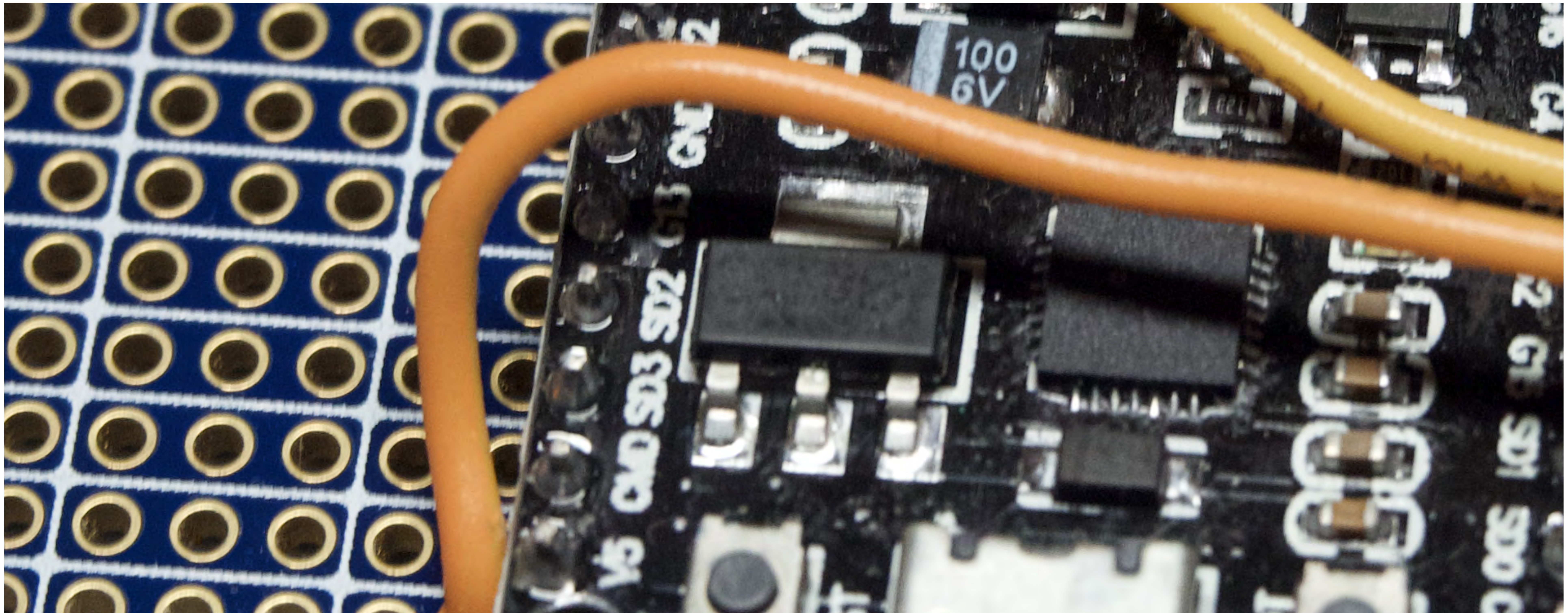


```python
# picocom then:
import uos
import flashbdev
uos.VfsFat.mkfs(flashbdev.bdev)
```

REALLY BRICKING IT

The CMD/GND gotcha

- There are two general layouts for the Dev Boards.
- It is very hard to read the difference between CMD and GND for the pin next to 5V.

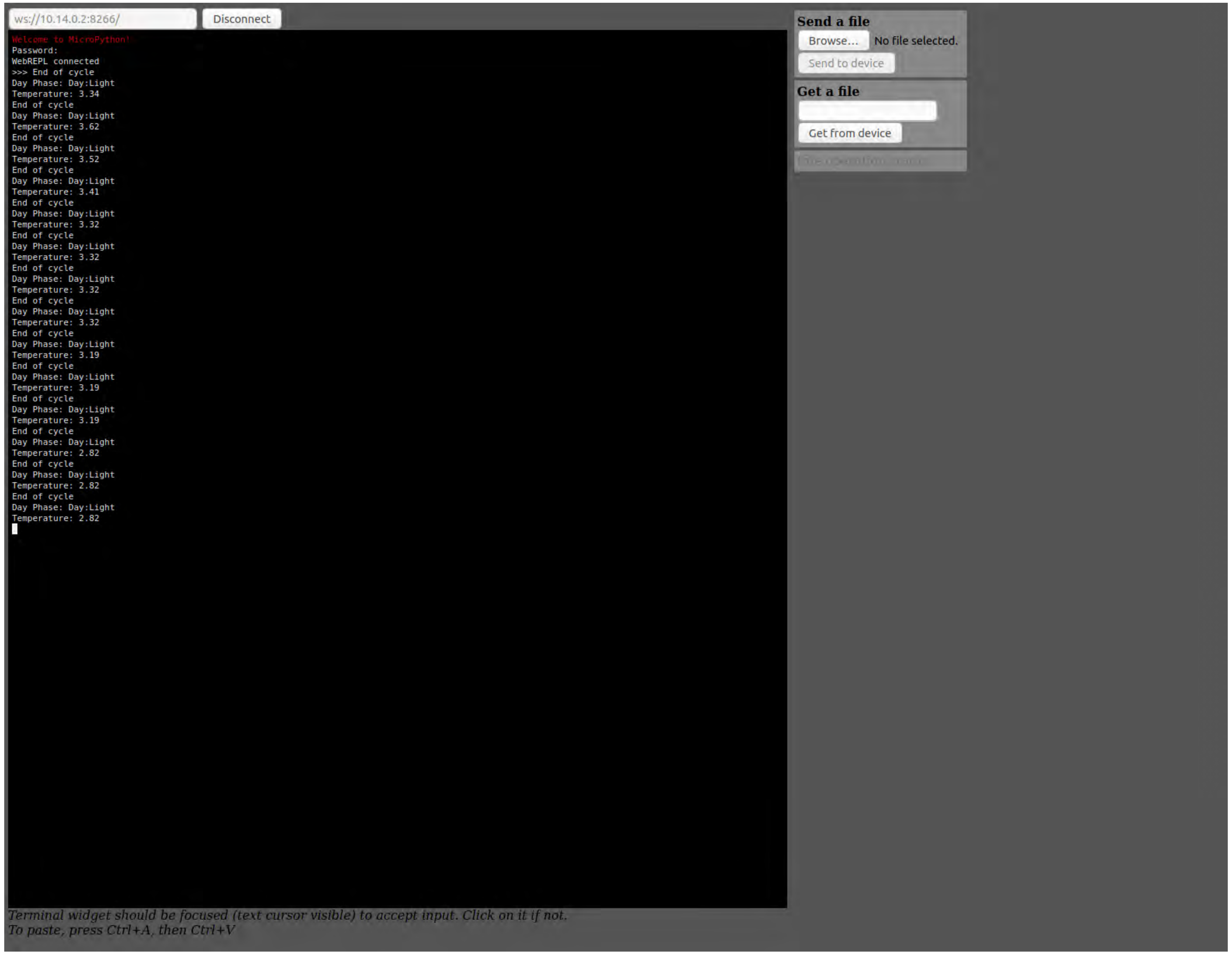


WEBREPL

And its gotchas

Allows for over the net interactions and file transfers.

Useful for live debug

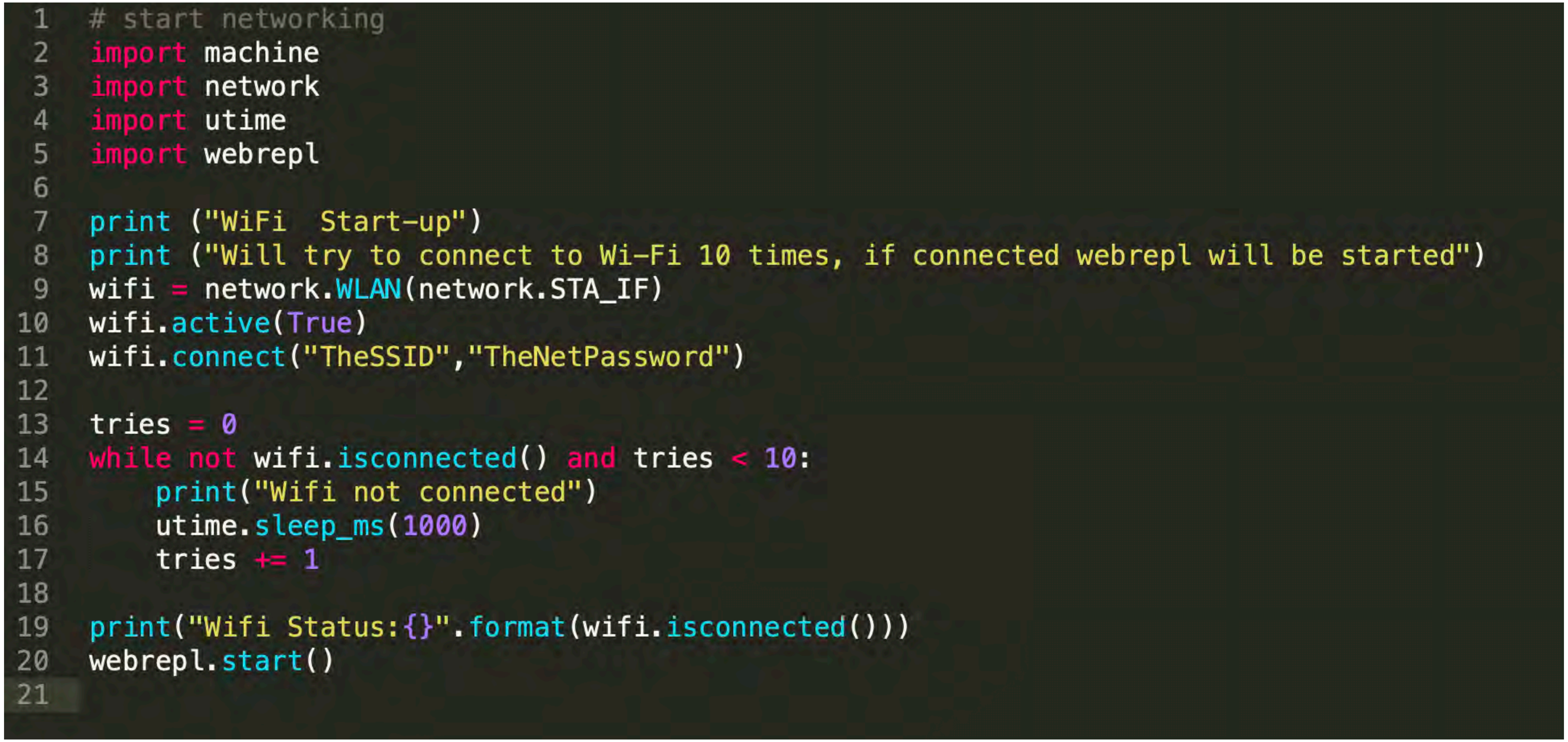


```
Welcome to MicroPython!
Password:
WebREPL connected
>>> End of cycle
Day Phase: Day:Light
Temperature: 3.34
End of cycle
Day Phase: Day:Light
Temperature: 3.62
End of cycle
Day Phase: Day:Light
Temperature: 3.52
End of cycle
Day Phase: Day:Light
Temperature: 3.41
End of cycle
Day Phase: Day:Light
Temperature: 3.32
End of cycle
Day Phase: Day:Light
Temperature: 3.32
End of cycle
Day Phase: Day:Light
Temperature: 3.32
End of cycle
Day Phase: Day:Light
Temperature: 3.32
End of cycle
Day Phase: Day:Light
Temperature: 3.19
End of cycle
Day Phase: Day:Light
Temperature: 3.19
End of cycle
Day Phase: Day:Light
Temperature: 3.19
End of cycle
Day Phase: Day:Light
Temperature: 2.82
End of cycle
Day Phase: Day:Light
Temperature: 2.82
End of cycle
Day Phase: Day:Light
Temperature: 2.82
```

Terminal widget should be focused (text cursor visible) to accept input. Click on it if not.
To paste, press Ctrl+A, then Ctrl+V

You would think

That this is a valid way of doing WEBRepl (in boot.py)



```python
# start networking
import machine
import network
import utime
import webrepl

print ("WiFi  Start-up")
print ("Will try to connect to Wi-Fi 10 times, if connected webrepl will be started")
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect("TheSSID","TheNetPassword")

tries = 0
while not wifi.isconnected() and tries < 10:
    print("Wifi not connected")
    utime.sleep_ms(1000)
    tries += 1

print("Wifi Status:{}".format(wifi.isconnected()))
webrepl.start()

```

But Actually

This works - and it has to be in main.py

```
lan = network.LAN(mdc = Pin(23), mdio = Pin(18), phy_type = network.PHY_LAN8720, phy_addr=0, clock_mode=network.ETH_CLOCK_GPIO17_OUT)
lan.active(1)

print ("IF CONFIG: {}".format(lan.ifconfig()))

sleep_ms(500)
webrepl.start()
```

DEEPSLEEP

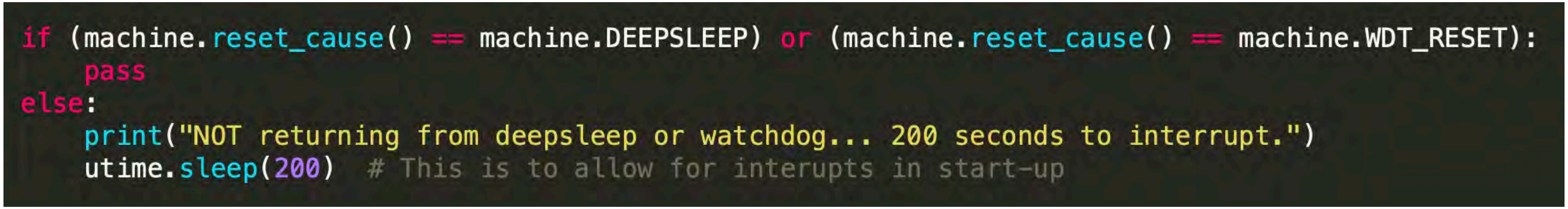
Entering ultra low power mode (50uA)

```
def deepsleep(sleeptime,timestr,ipstr):
    http_get("http://wbgw.thirtover.com/sendevent/TestLights/Deepsleep?val={}&ip={}".format(timestr,ipstr))
    print("Just before deepsleep: {} ip: {}".format(timestr,ipstr))
    #utime.sleep_ms(sleeptime) # whilst testing
    machine.deepsleep(sleeptime)
```

sleeptime in mS

Returning From Deepsleep

Its like a reboot Jim - but not as we know it



```
if (machine.reset_cause() == machine.DEEPSLEEP) or (machine.reset_cause() == machine.WDT_RESET):
    pass
else:
    print("NOT returning from deepsleep or watchdog... 200 seconds to interrupt.")
    utime.sleep(200)  # This is to allow for interupts in start-up
```

Feed the dog

Its worth it

```
wdt = machine.WDT(timeout=(sleep_time*cycle_time)+5000)  # 5 seconds more than (light) sleep
wdt.feed()
```

ANALOG IN

Is a bit touchy

```
def __init__(self,pin,ts_pin):
    self.adc = machine.ADC(machine.Pin(pin))
    self.adc.atten(machine.ADC.ATTN_11DB)
    self.adc.width(machine.ADC.WIDTH_10BIT)
```

HTTP is a Breeze

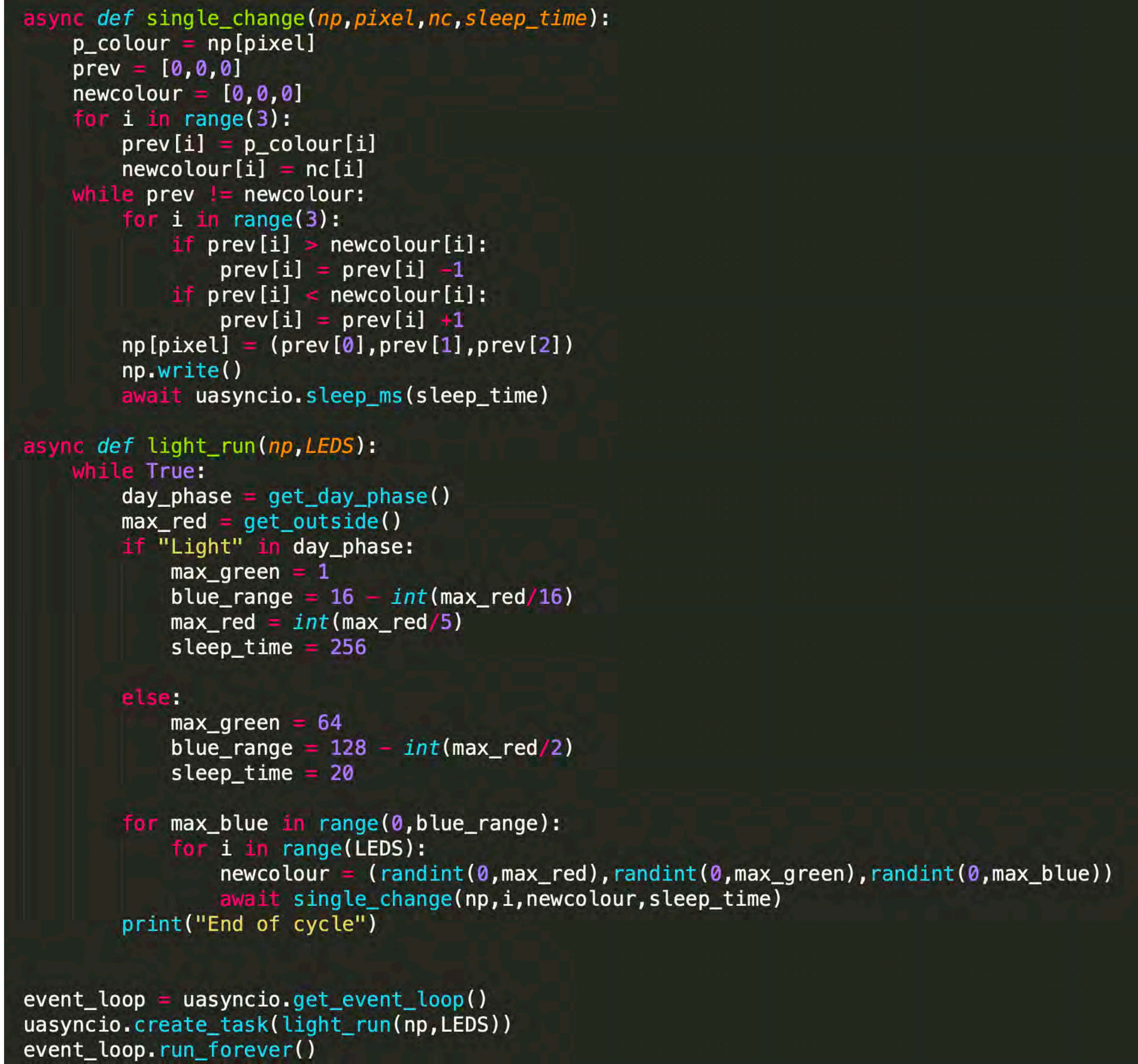
urequests

```python
def get_item(url, tag='val'):
    try:
        res = urequests.get(url)
        return res.text.split('<{}>'.format(tag))[1].split('</{}>'.format(tag))[0]
    except:
        print ("get_item FAILED")
        return ""
```

ASYNCIO

The joy of threads

WEBRepl
still works!



```
async def single_change(np,pixel,nc,sleep_time):
    p_colour = np[pixel]
    prev = [0,0,0]
    newcolour = [0,0,0]
    for i in range(3):
        prev[i] = p_colour[i]
        newcolour[i] = nc[i]
    while prev != newcolour:
        for i in range(3):
            if prev[i] > newcolour[i]:
                prev[i] = prev[i] -1
            if prev[i] < newcolour[i]:
                prev[i] = prev[i] +1
        np[pixel] = (prev[0],prev[1],prev[2])
        np.write()
        await uasyncio.sleep_ms(sleep_time)

async def light_run(np,LEDS):
    while True:
        day_phase = get_day_phase()
        max_red = get_outside()
        if "Light" in day_phase:
            max_green = 1
            blue_range = 16 - int(max_red/16)
            max_red = int(max_red/5)
            sleep_time = 256

        else:
            max_green = 64
            blue_range = 128 - int(max_red/2)
            sleep_time = 20

        for max_blue in range(0,blue_range):
            for i in range(LEDS):
                newcolour = (randint(0,max_red),randint(0,max_green),randint(0,max_blue))
                await single_change(np,i,newcolour,sleep_time)
        print("End of cycle")


event_loop = uasyncio.get_event_loop()
uasyncio.create_task(light_run(np,LEDS))
event_loop.run_forever()
```

BLUETOOTH (BLE)

Has got better since …

```python
def bt_irq_beacon(event, data):
  if event == _IRQ_SCAN_RESULT:
    addr_type, addr, iscon, rssi, adv_data = data
    if ibeacon_id in ubinascii.hexlify(adv_data):
        # this is very likely to be an ibeacon
        if (ubinascii.hexlify(addr)) in ibeacons:
            print('KNOWN {} - type:{} addr:{} rssi:{} data:{}'.format(ibeacons.index(ubinascii.hexlify(addr)), addr_type, ubinascii.hexlify(addr), rssi, ubinascii.hexlify(adv_data)))
            if int(rssi) > threshold:
                indicate(ibeacons.index(ubinascii.hexlify(addr)))
            else:
                print ("Signal: {} below threshold".format(rssi))
        else:
            print('NOT      - type:{} addr:{} rssi:{} data:{}'.format(addr_type, ubinascii.hexlify(addr), rssi, ubinascii.hexlify(adv_data)))
  elif event == _IRQ_SCAN_COMPLETE:
    # Scan duration finished or manually stopped.
    np[0] = (0,0,4)
    np.write()
    print('scan complete')
```

ONEWIRE

Double Easy

```
#Init OneWire
def init_onewire():
    ds = ds18x20.DS18X20(onewire.OneWire(temp_sensor_pin))
    roms = ds.scan()
    for rom in roms:
        print("Found: {}".format(rom))
    return ds, roms

def read_temperature():
    try:
        ds.convert_temp()
        utime.sleep_ms(750)
        temperature = ds.read_temp(roms[0])
    except:
        temperature = 100 # return a high temperature to force a fast fan speed for safety
        init_onewire() # try to recover the onewire bus
    return temperature
```

ESP32 MicroPython conclusions

Learn the basics in a afternoon

- Easier to debug than Arduino C++, has the ability to debug in situ over the network
- Network and data wrangling much easier
- Multi-threading is easy to comprehend
- WEBRepl is password protected, REPl is not.
- Would need MPY for commercial projects