
This talk might be recorded

How Pythons kill

- Backward pointing teeth grab the prey
- Then the victim is constricted like in the cartoons
- Blood flow is severely impeded and cardiac arrest typically follows


And now for
something
...
much smaller

Micropython

For rapid IoT prototyping

Per Buer, May 2020

Per Buer

@perbu

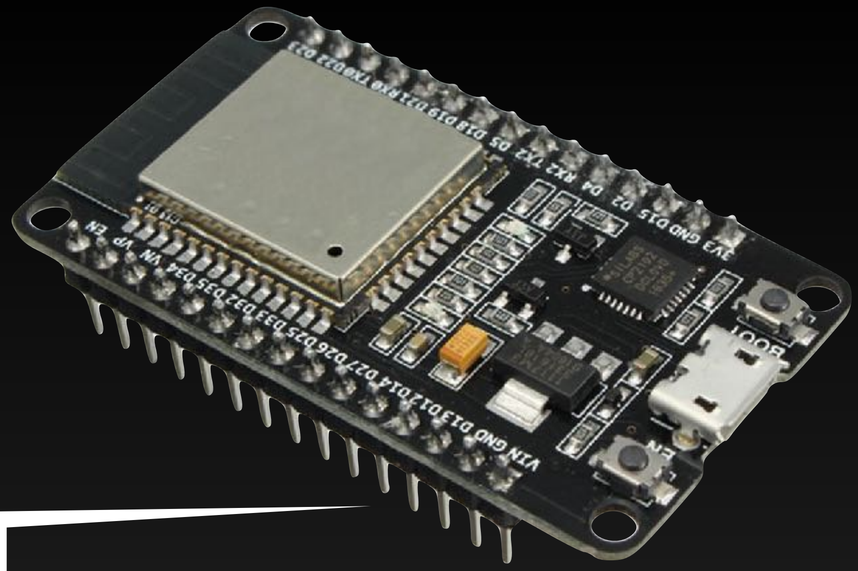
Linpro
Varnish Software
IncludeOS
KnowIT Objectnet

Talk overview

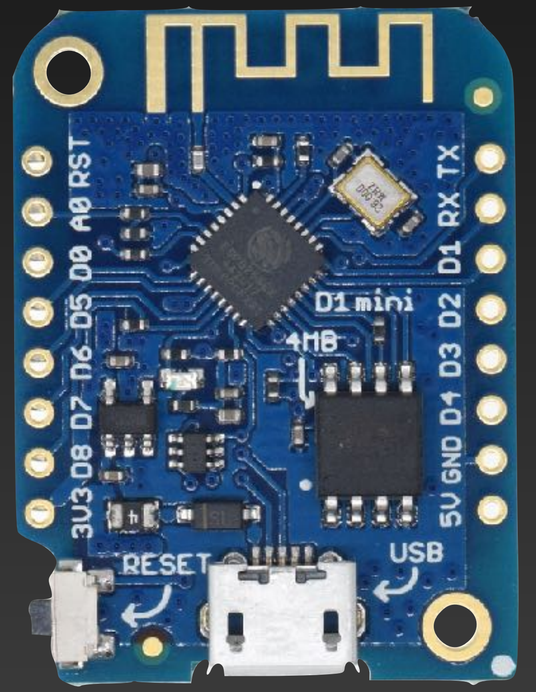
- What are microcontrollers?
- Python
- Micropython
  - Look ma! No compiler!
  - Finding a good workflow
- Build 1 - LEDs and movement sensors
- Build 2 - i2c devices
- Build 3 - MQTT and JSON - making an IoT device

MCUs

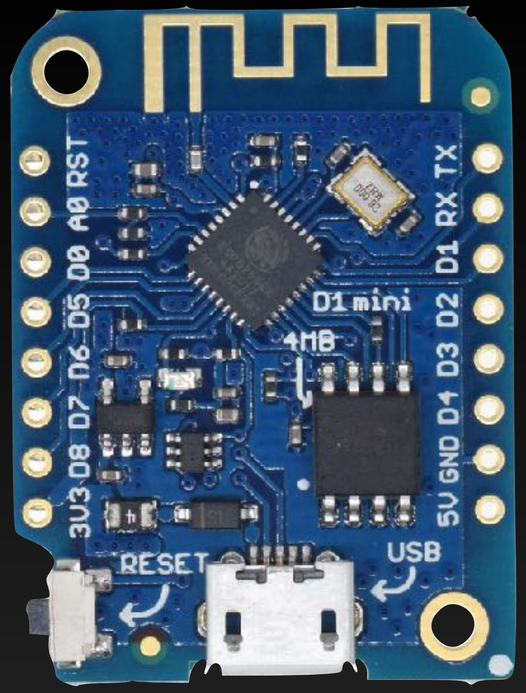
- Simple computers.
- No traditional OS. Runs “one thing”.
- Cheap, $2-$8
- GPIO allows you to connect other stuff
- The cool ones have WiFi - IoT

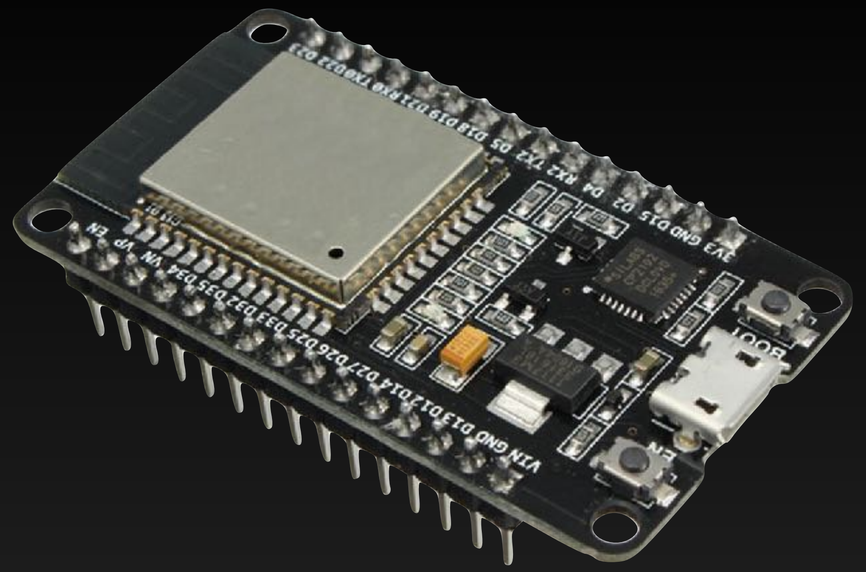


ESP32



ESP8266



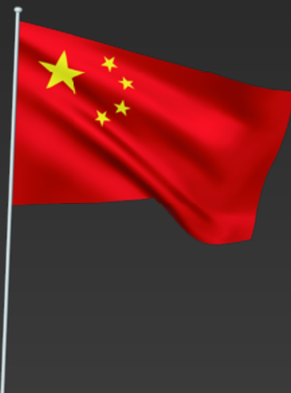


ESP8266

- Cheapest
- Only one analog pin
- Got popular as a wifi chip for the Arduino

ESP32

- Cheap
- Lots of pins
- GPIO pins are messy



Typical sensors
and actuators

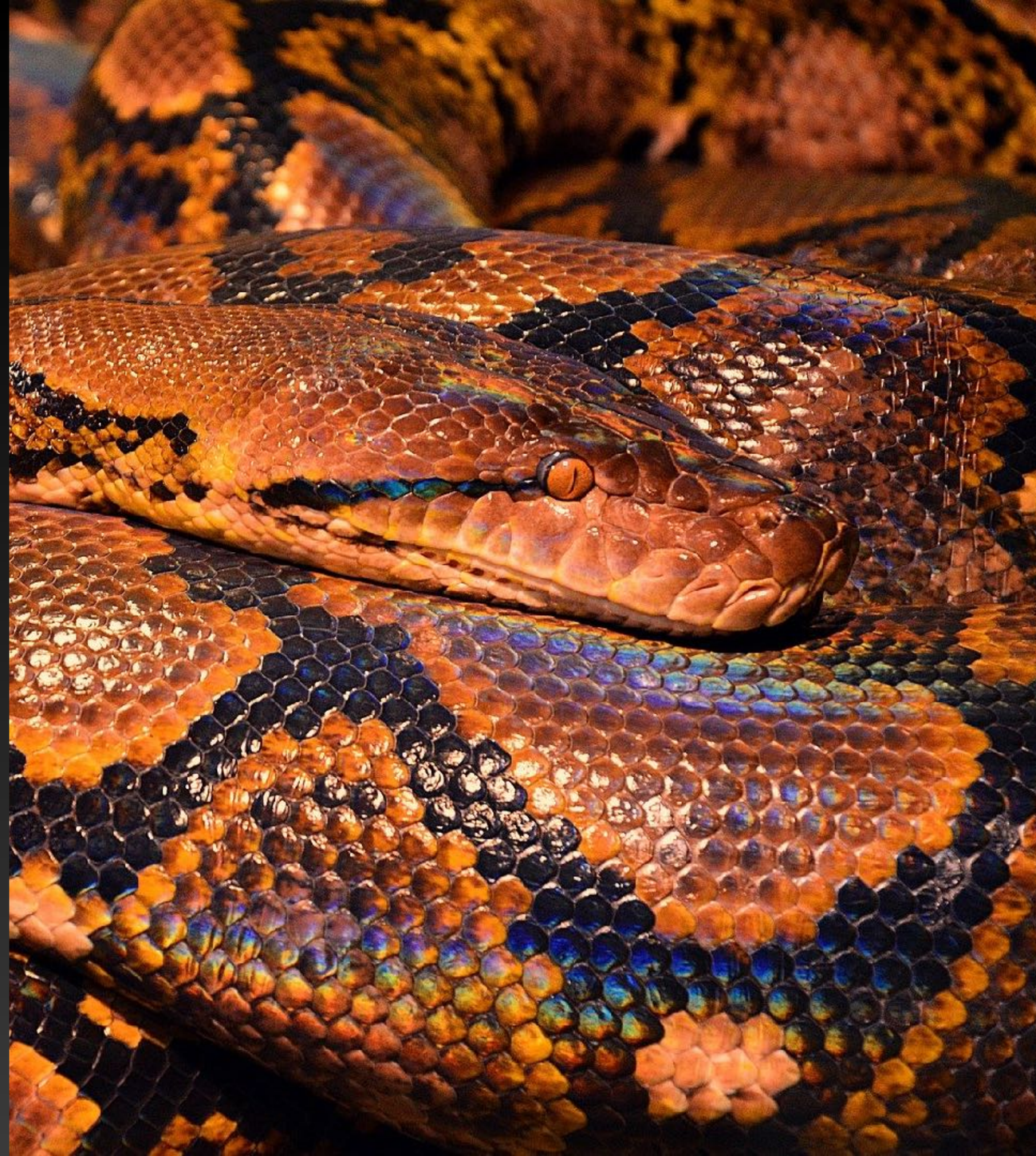
- Environment (Temperature, pressure, humidity)
- Distance (Sonic, IR)
- Motion (accelerometers, gyros)

- Relays for power control
- LEDs, Displays
- Servos, stepper motors

Motivation

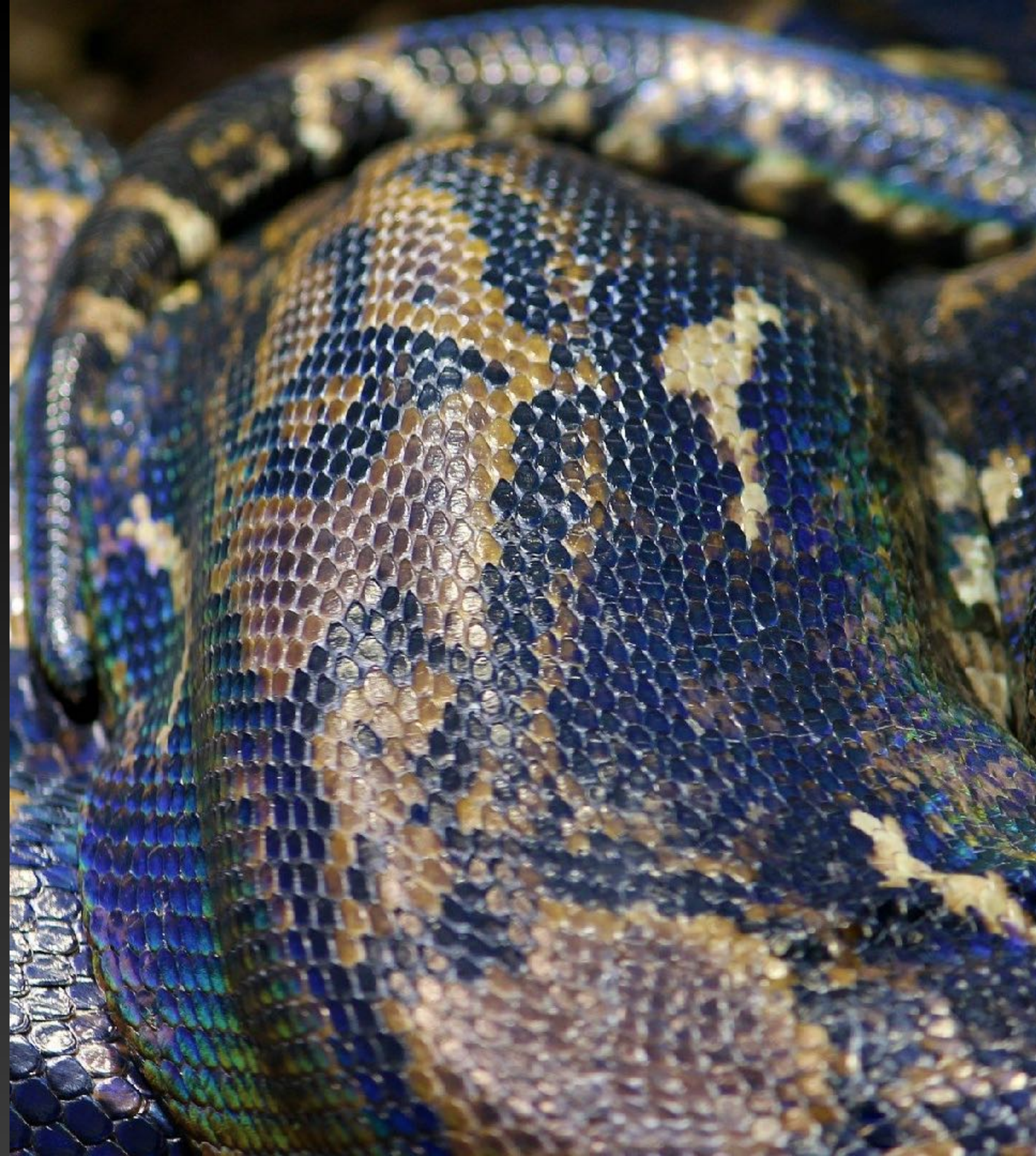
Why not prototype with Arduino?

- C and C++ is hard and dangerous
- We want to build prototypes. Quickly.
- Python’s easy syntax means this can be done by ~everybody



Limitations

- No real time code (GC, FreeRTOS)
- On-device security is very weak - no encryption.
- Higher power consumption
- Drivers aren’t as plentiful as on Arduino



Python

For non-pythonistas

Python

- Interpreted language from the early 90s
- Primary focus on readability of code - no clever stuff, please
- REPL allows to quickly tests constructs
- Let me show you.

Micropython

Micropython

- Reimplementation of Python 3.x with a tiny resource budget
- Supports 95% of Python
- 10-100x slower than C or C++
- Runs nicely on micro:bit, ESP8266, ESP32 and the Pyboard family

Installing Micropython

- Download the version of micropython you need
- Plug in the MCU - you'll get a new COM port (COM9, /dev/ttyUSB or /dev/cu.something)
- Install esptool
- For ESP8266 and ESP32:
  - Use esptool to flash
  - esptool.py --chip esp32 --port /dev/cu.SLAB_USBtoUART write_flash -z 0x1000 esp32spiram-idf4-20200405-v1.12-337-g312c69949.bin
  - esptool.py --port /dev/cu.Repleo-CH341-00004114 --baud 57600 write_flash --flash_size=detect 0 esp8266-20200324-v1.12-270-g38ccb4c64.bin
- You can use a terminal program to get the REPL prompt

Dynamic language

How do we write code for this?

- REPL allows you to interactively test stuff
- Code lives on a filesystem - main.py is executed on boot.
- With Arduino we'll have to compile and flash the program onto the device for each change
- With Micropython - you just run a file on the device:
  - $ pyboard —device /dev/cu.SLAB_USBtoUART hello.py

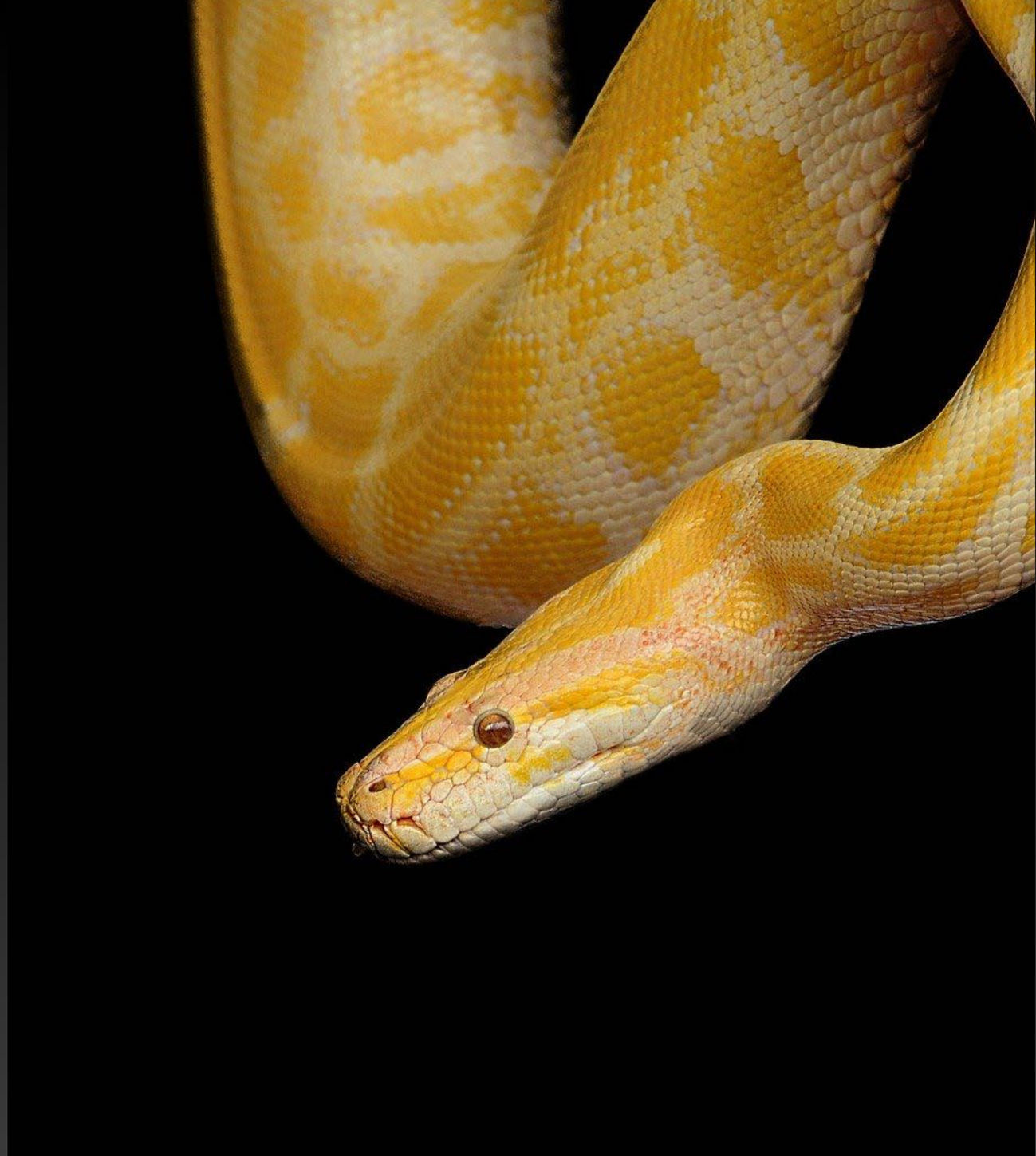
demo

Tools you'll need

- pyboard.py
  - For running foo.py on the device
  - Copying things to the device
- rshell
  - Good interactive shell for copy or sync folders to the device
  - Easy access to REPL
- screen or any other serial terminal
  - For direct access to the REPL

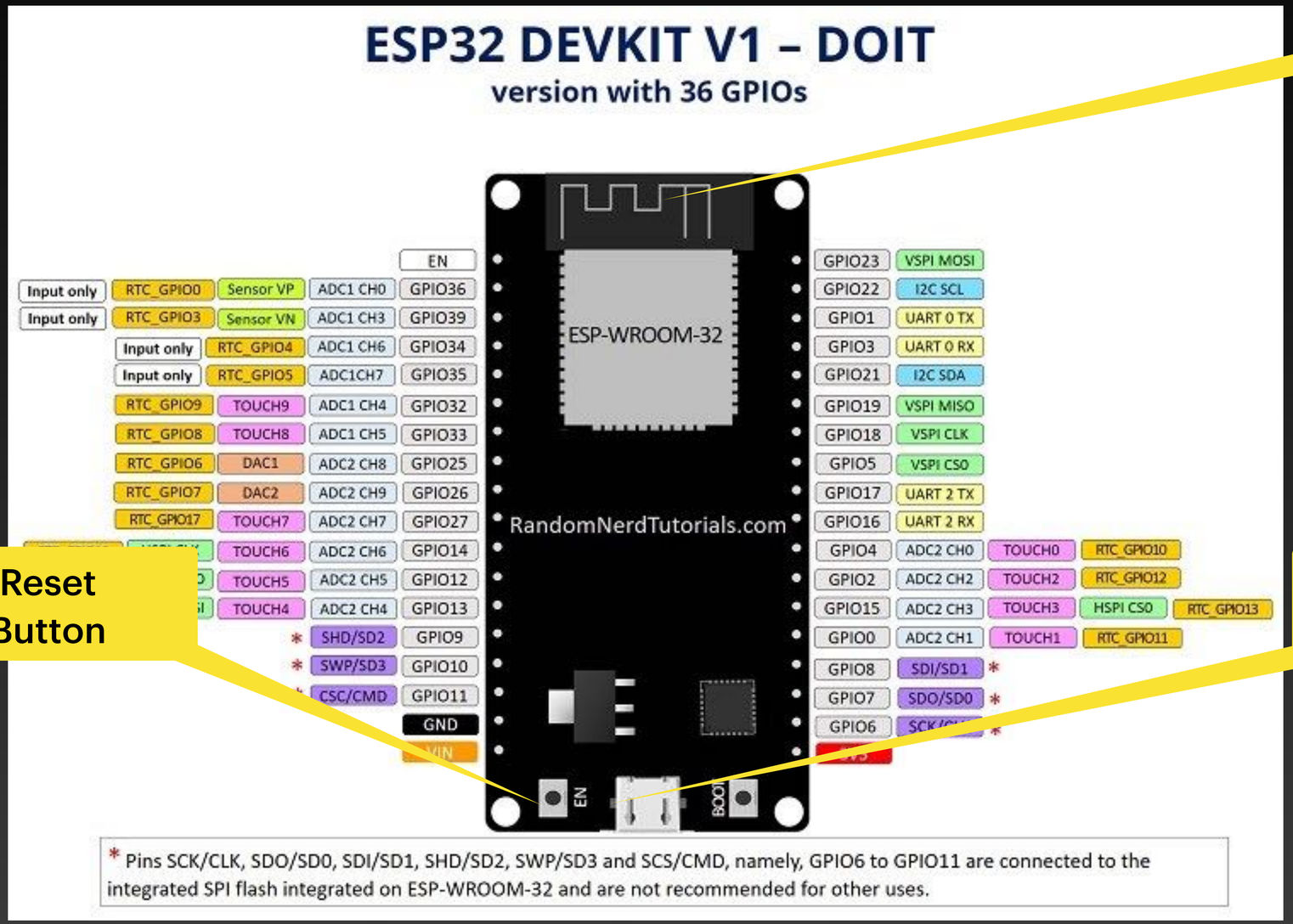
Let’s blink a LED

whenever I move around



Pinout

“esp32 devkit pinout” on google



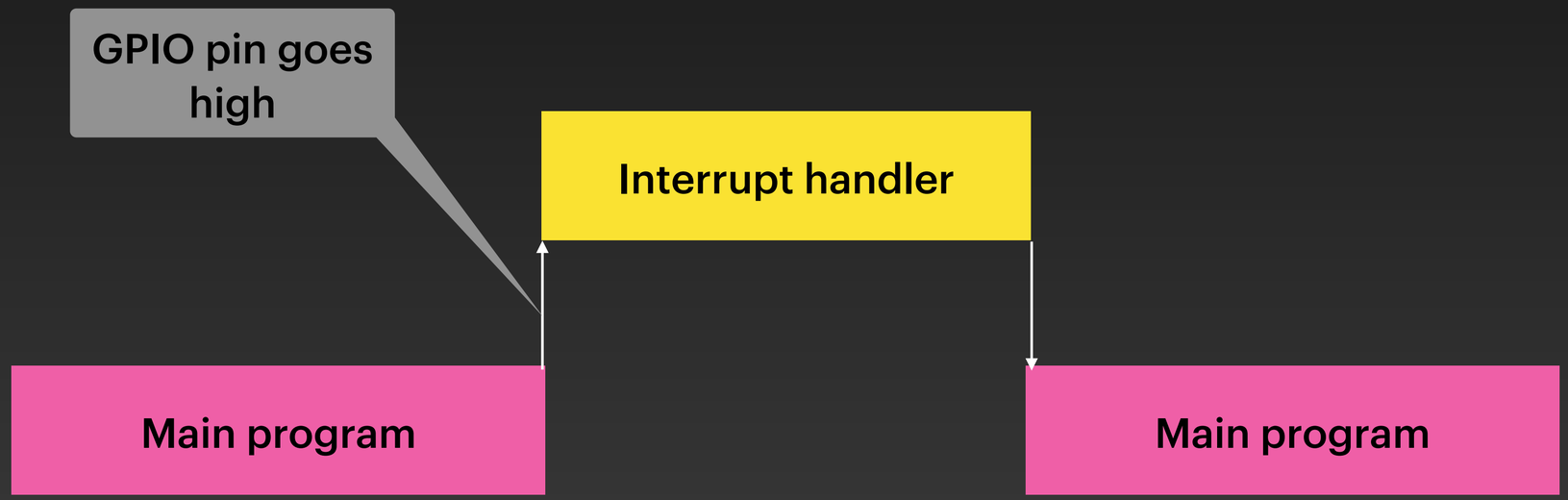
Wifi Antenna

Reset Button

MicroUSB connector

Interrupts

- When a GPIO pin changes state or a timer goes off you can run your code
- For GPIO state changes you can run the handler on:
  - Rising flank
  - Falling flank
  - Any change



```
from machine import Pin
from time import sleep

pir = Pin(23, Pin.IN)
led = Pin(2, Pin.OUT)


def handle_interrupt(pin):
    print('Movement detected.')
    led.on()
    sleep(1)
    led.off()

print('Setting up interrupt handler...')
pir.irq(trigger=Pin.IRQ_RISING, handler=handle_interrupt)

while True:
    sleep(1)
```

Built in LED

Notice the lack of ()

Just keep on swimmin’

Questions so far?

Let's connect more stuff with

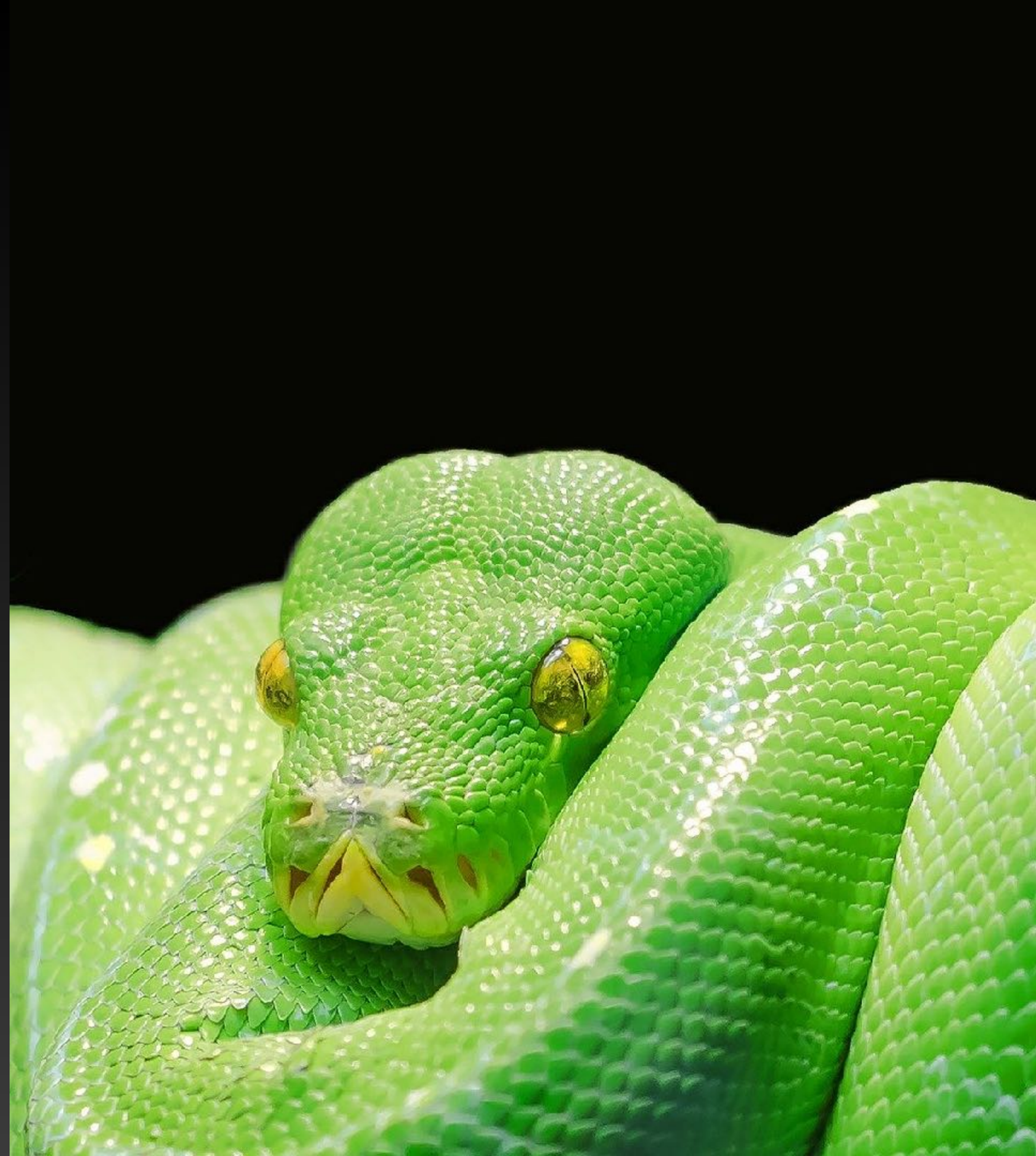
i²c

i²c bus

- Connect many device to two pins - sda and scl
- sda for data
- scl for clock

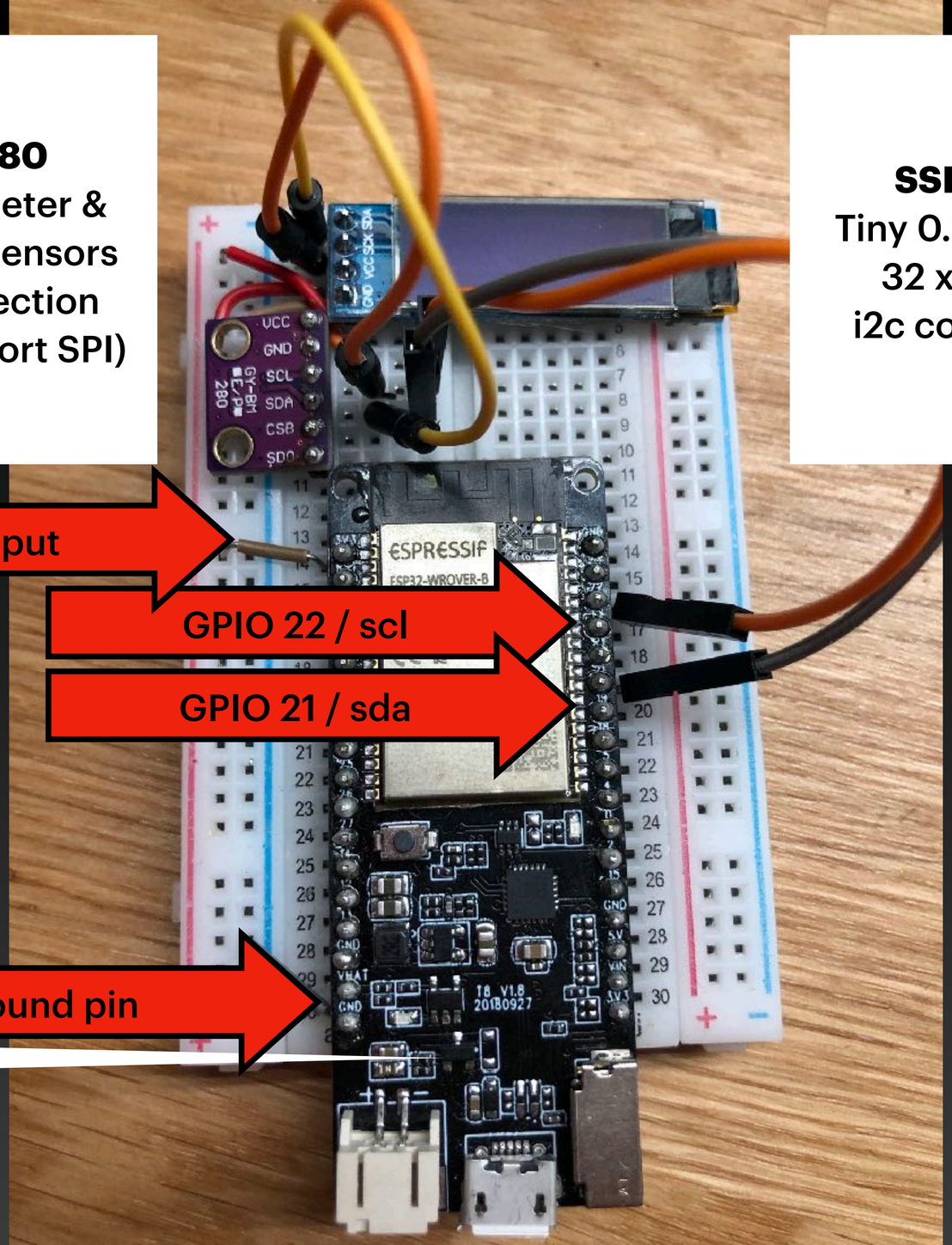
Build 2

- ESP32
- BMP280 sensor
  - Connected to the i²c bus
  - Temperature
  - Pressure
- OLED Display
  - Same i²c bus



BMP280
Thermometer & pressure sensors
i2c connection
(Also support SPI)

SSD1306
Tiny 0.96” OLED
32 x 128 px
i2c connected



Note:
The pin markings don’t match the GPIO number of the ESP32s.
Find the relevant pinout diagram.

Notice that this ESP32 looks completely different. It has a SD card slot and a battery connector

Lets explore the i²c bus

```
from machine import I2C, Pin
i2c = I2C(sda=Pin(21), scl=Pin(22))
print(i2c.scan())
```

Code dump ahead

Setup code

```
from machine import Pin, I2C
import time
import ssd1306 # We need to place this on the device
import bmp280  # We need to place this on the device

# Constants:
pinScl    = const(22)
pinSda    = const(21)
addr_oled = const(60)   # 0x3c
addr_bmp  = const(118)  # 0x76
h_size    = const(32)     # display heigh in pixels
w_size    = const(128)    # display width in pixels

# Variables
oled_conn = False
bmp_conn = False
temp = 0
pr = 0
hum = 0

oled = None
bmp = None

i2c = I2C(scl=Pin(pinScl),
          sda=Pin(pinSda))
```

Init code

```
print('Scanning i2c bus...')
devices = i2c.scan()
if len(devices) == 0:
    print("No i2c devices found.")
else:
    print('i2c devices found:', len(devices))
    for device in devices:
        if device == addr_oled:
            print('OLED at ', addr_oled)
            oled_conn = True
            oled = ssd1306.SSD1306_I2C(w_size, h_size, i2c, addr_oled)
        if device == addr_bmp:
            print('BMP at ', addr_bmp)
            bmp_conn = True
            bmp = bmp280.BMP280(i2c_bus=i2c, addr=addr_bmp)
            bmp.use_case(bmp280.BMP280_CASE_WEATHER)
            bmp.force_measure()
        print(device)
```

Initialize the OLED display

Initialize the sensor

Disable power saving

Main loop

```
while True:
    if bmp_conn:
        print("BMP280 values:")
        temp = bmp.temperature
        pr = bmp.pressure
        print("temp:", temp)
        print("pr:", pr)

    if oled_conn:
        oled.fill(0)
        if bmp_conn:
            oled.text("Temperature: {}".format(temp), 0, 0)
            oled.text("Pressure: {}".format(pr), 0, 10)
            oled.show()
        else:
            oled.text("BMP280 N/A", 0, 0)
            oled.show()
    else:
        print('No i2c display’)
    time.sleep_ms(5000)
```

Networking

No fun without it

```
import time
import network

def wlan_connect():
    ap = network.WLAN(network.AP_IF)
    ap.active(False)
    sta = network.WLAN(network.STA_IF)
    sta.active(True)
    if not sta.isconnected():
        print('Connecting')
        sta.connect('ssid', 'password')
        while not sta.isconnected():
            time.sleep(0.1)
            print('.', end='')
        print('Connected!')
    print('Network:', sta.ifconfig())

wlan_connect()
```

boot.py
Gave me a hard time

- “boot.py” runs when your MCU boots up
- If it hangs rshell is not able to connect to the MCU
- Put networking in main.py

JSON in Micropython

```
import ujson
obj = {
    'temperature': temp,
    'pressure': pressure
}

json_str = ujson.dumps(obj)
```

MQTT in Micropython

MQTT
In Micropython

- Simple pub/sub protocol
- Developed for oil pipes
- ISO/IEC 20922
- Very common in IoT
- Payload is typically JSON

```
from umqtt.robust import MQTTClient

mqtt_client = MQTTClient(uuid,
        mqtt_server)

mqtt_client.connect()
mqtt_client.publish(topic, json_str)
```

Last demo

- Using test.mosquitto.org which is publicly available
- TLS is not an issue with Micropython
- mosquitto_sub -h test.mosquitto.org -t 'knowit/objectnet/hardwareguild/micropython0'

demo

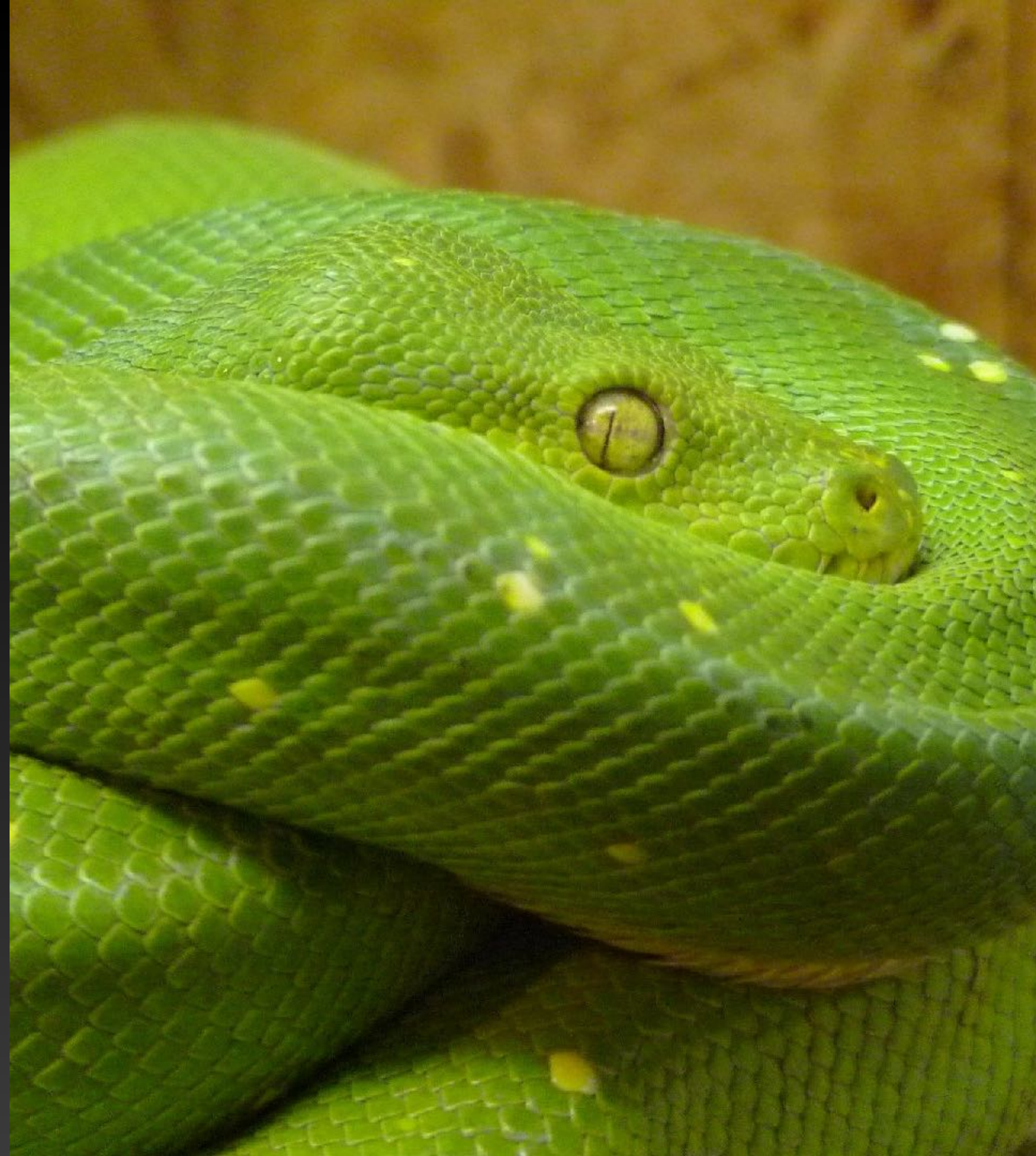
Copy file to ‘main.py’ in order to persist across reboots

Experiences

- Getting the sensor up is particularly easy compared to C/C++-enviroments. i2c.scan() is awesome
- Things like automatically pulling code from Github is easy so OTA needs no tools.
- No fucking YAML

Summing up

- Micropython is amazing for prototyping IoT devices. Iterate quickly.
- Use the REPL when connecting sensors or testing networking
- rshell and pyboard are nice
- On-device security is non-existing




Questions
https://github.com/perbu/micropython-talk-2020