MicroPython:
The Next Steps to World Domination

Steve Holden

CTO, Global Stress Index Limited

Introductions

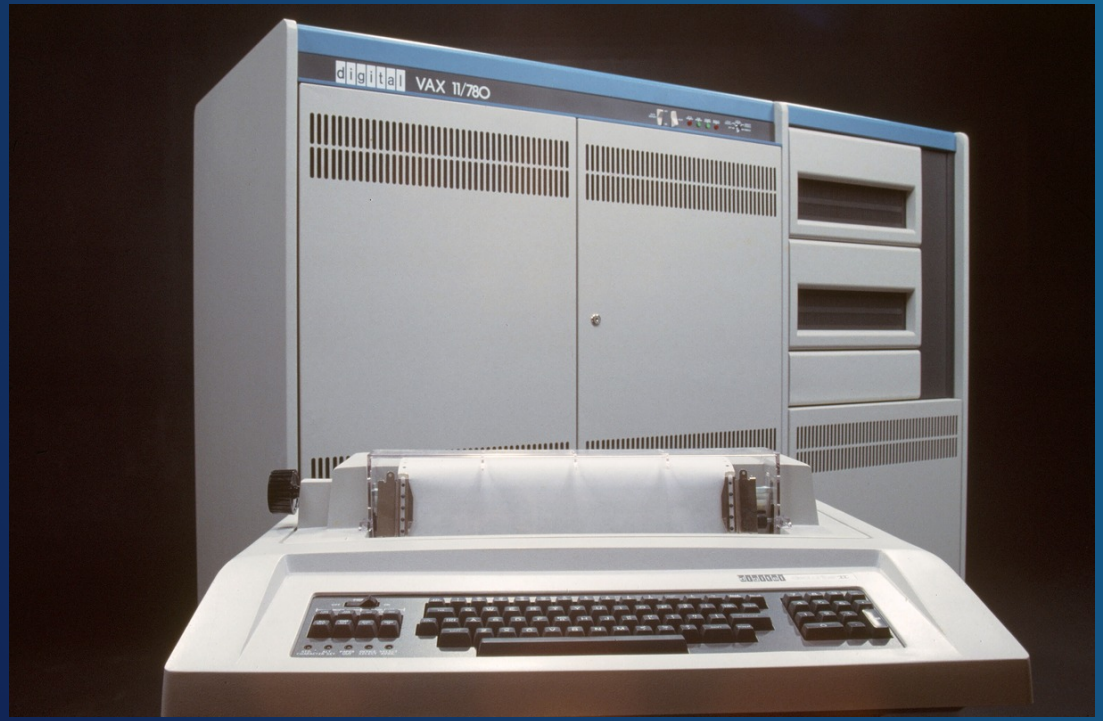
- Started programming in 1968
- Computational scientist by training
- Engineer at heart, electronics hobbyist
- Python user since Python 1.4
- Enjoy helping people to learn

Developments in Computing

SOME HISTORY

1948

1977

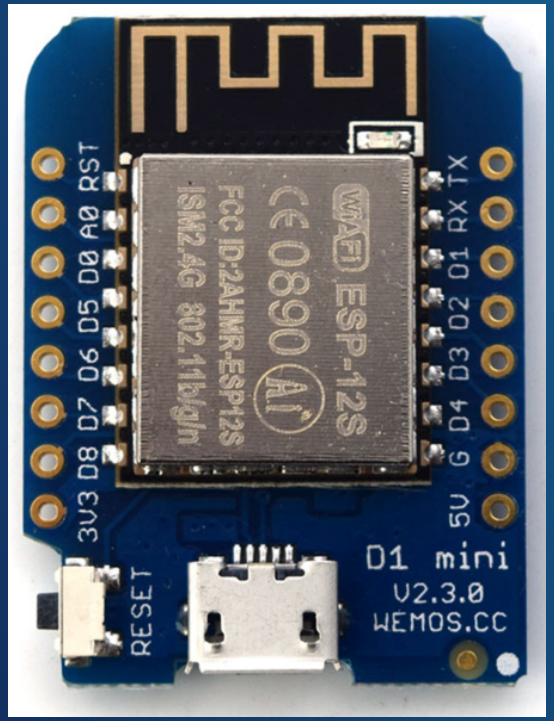


1984

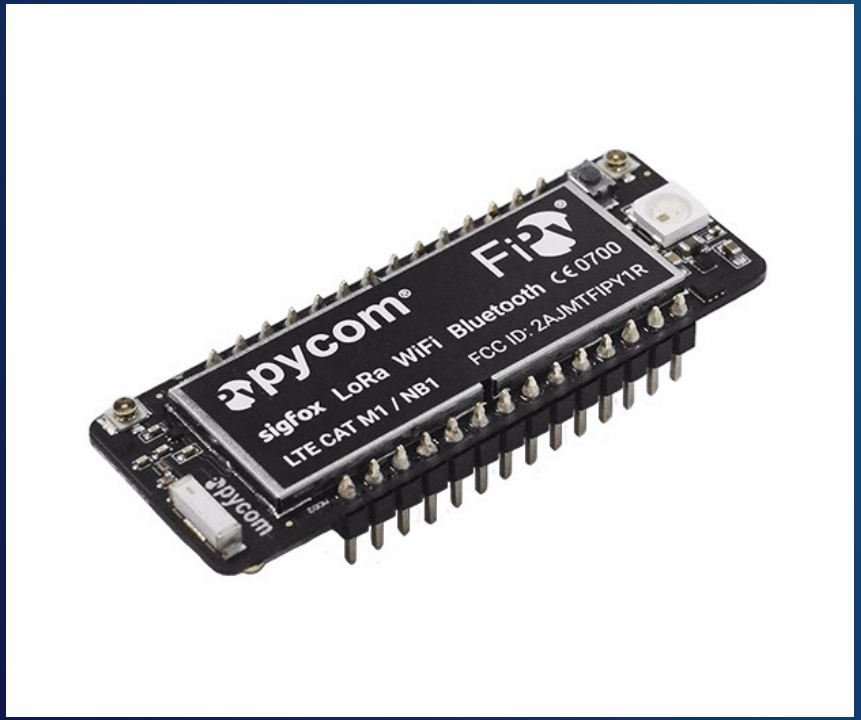

Archivio Composizione Esposizione Strumenti
System Disk
The Macintosh™ Finder
Bruce Horn and Steve Capps
512K
Cestino

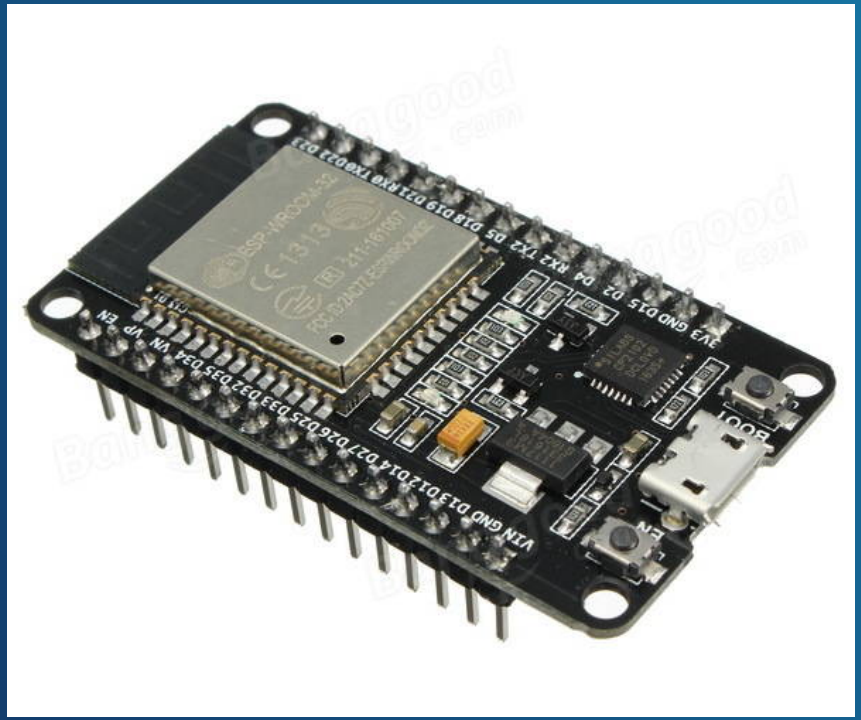
2015

2016

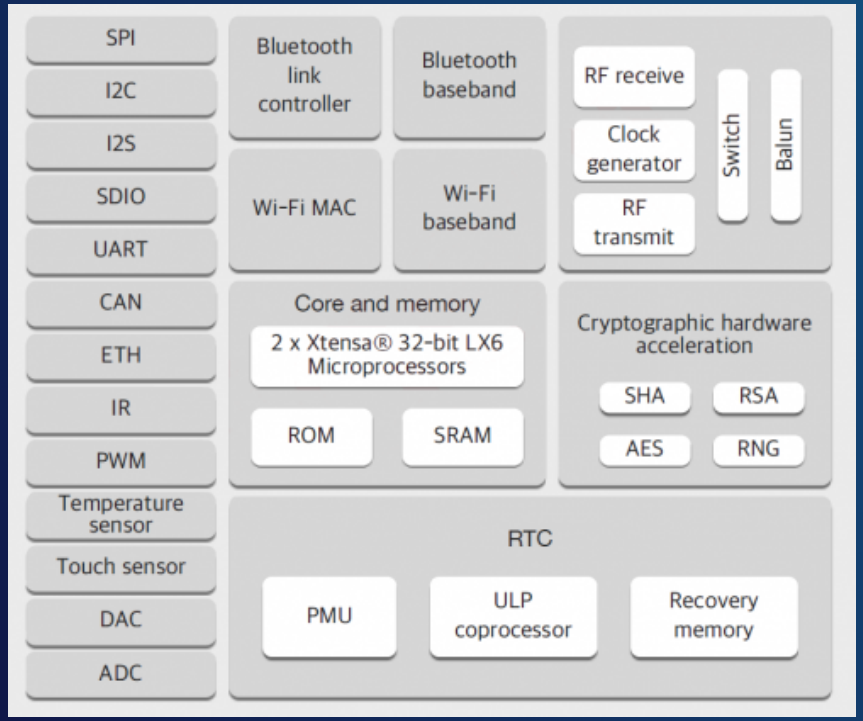

RST A0 D0 D5 D6 D7 D8 3V3
TX RX D1 D2 D3 D4 G 5V
FCC ID:2AHMR-ESP12S
ISM2.4G 802.11b/g/n
ESP-12S
0890
RESET
D1 mini
V2.3.0
WEMOS.CC

2017


pycom
sigfox LoRa WiFi Bluetooth
LTE CAT M1 / NB1
FCC ID: 2AJMTFIPY1R
FiPy



Some ESP32 Features



- Twin-core CPU (200 MIPS)
- Wifi and Bluetooth
- Crypto acceleration
- Touch sensing
- A-to-D and D-to-A
- Many comms interfaces

2020

Python’s Popularity

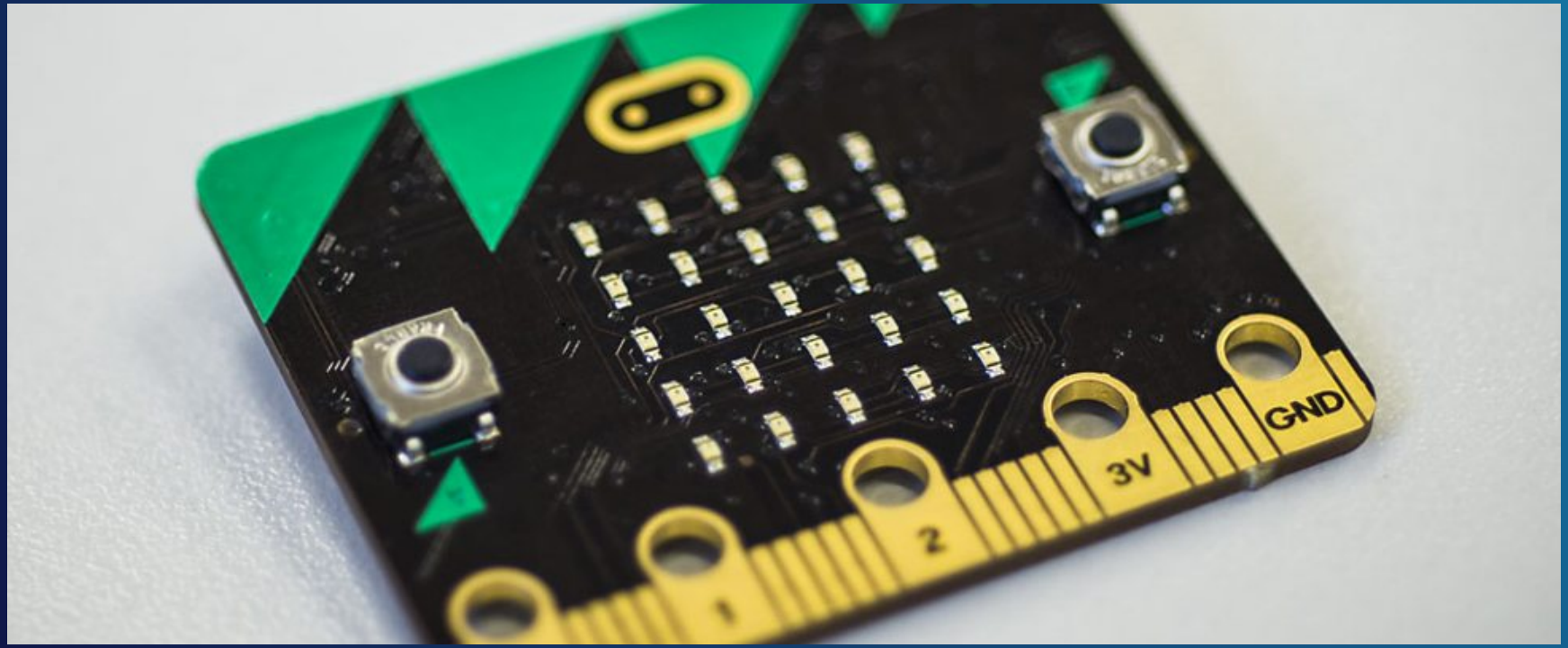
THE BIRTH OF THE MICRO:BIT

20-Year Plan for World Domination

1. Help grow Python's communities
2. Help underpin the Python Software Foundation
3. Profit? (and for whom?)

A million new users in a single year might help…

A Seminal Moment: 22.03.2016



The BBC micro:bit

- ARM Cortex M0 processor
- 256kB flash memory, 16kB static RAM
- Accelerometer and Magnetometer
- Bluetooth and USB connectivity
- 5 x 5 red LED matrix

Goal: Python in Everyone's Reach

Note to Purists

- We are talking about *learners*
- They do not need huge libraries
- Direct hands-on experience *enables*
- Large resources are not required
  - Large numbers of devices *are*

MicroPython

Damien George

Nick Tollervey

The Language Itself

HOW LIKE PYTHON IS IT?

MicroPython Features

- The *entire* Python 3.4 syntax, including
  - Exceptions
  - `with`, `yield from`, *etc.*
- Also adds 3.5's `async` and `await`
- Optional machine code!

- Types include str, bytes, bytearray, tuple, list, dict, set, frozenset, array.array, collections.namedtuple
- Also, naturally, classes and instances

Supported Systems

- PyBoard
- WiPy
- ESP8266
- ESP32
- STM32F4
- NUCLEO boards
- Espruino Pico

Filesystem

- A small amount of EPROM appears as **`/flash`**
- With hardware support you can also get **`/sd`**
  - One or the other becomes current directory
- Can access via an abbreviated **`os`** module
- **`open`** works as usual

Boot Sequence

- **`boot.py`** is run
- The USB interface is enabled
- **`main.py`** is run
- Most systems also allow a safe-mode boot
  - Helpful when you get **`boot.py`** wrong
- Can reset the filesystem to factory settings

Paste Mode

- Enter with `^E`
- Paste code in
- Terminate entry and parse code with `^D`

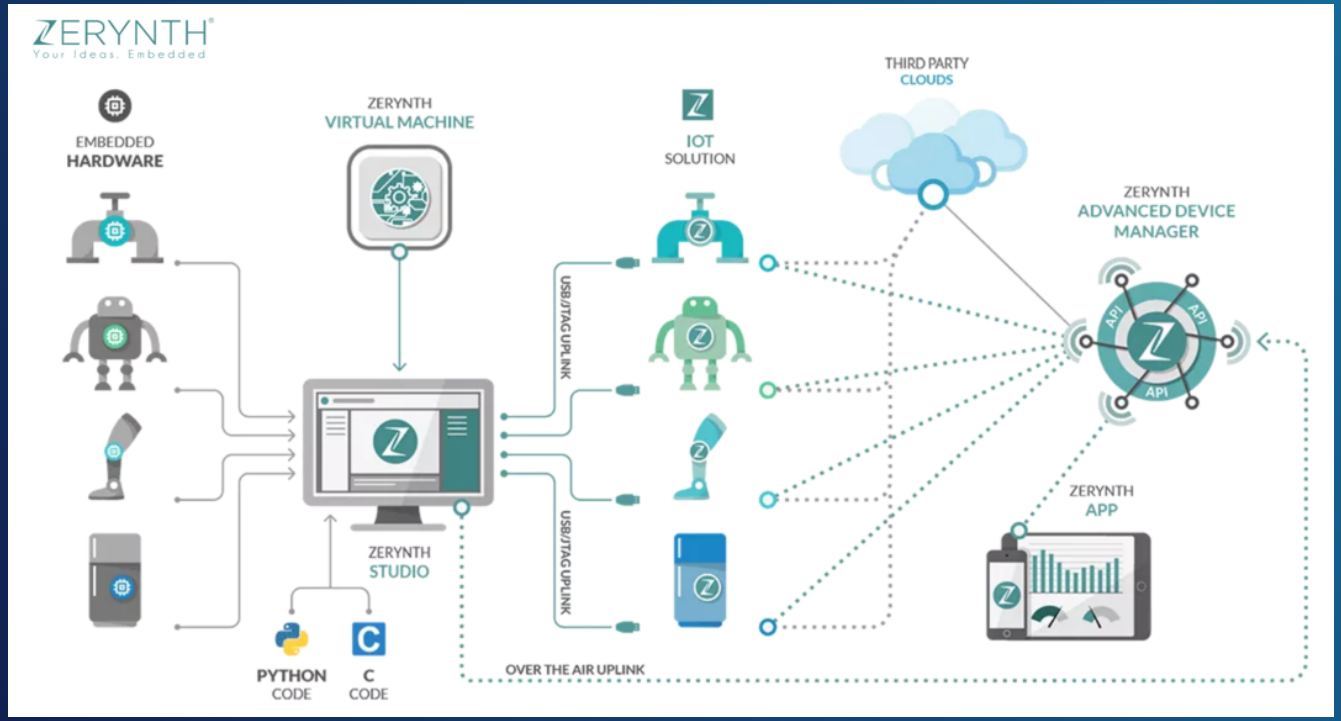
Installation (ESP8266 Systems)

```
# Get the MicroPython binary
wget http://micropython.org/resources/firmware/esp8266-20170823-v1.9.2.bin

# Ensure esptool is installed
pip install esptool

# configure for specific port
alias esptool="python -m esptool --port /dev/tty.wchusbserial1410"

# Erase flash and install the MicroPython image
esptool erase_flash
esptool --baud 115200 write_flash --flash_size=detect 0 \
                    esp8266-20170823-v1.9.2.bin
```

Zerynth’s Internet of Things Environment

COMMERCIAL SUPPORT

Zerynth: IoT with MicroPython and C



Zerynth MicroPython Features

- Studio: an IDE
- VM: real-time ‘OS’ – hardware independent
- Advanced Device Manager
  - OTA updates and remote procedure calls
- App: mobile phones to communicate directly with Python-programmed IoT devices

Zerynth Supported Systems

- ESP8266
- ESP32
- Microchip/Atmel SAM*
- Nordic NRF*
- NXP K64
- Particle Photon
- 33 different systems in all!

Show me the code!

SOME SIMPLE CODE EXAMPLES

Digital I/O

```
from machine import Pin

# Digital input
in_pin = Pin(0, machine.Pin.IN, machine.Pin.PULL_UP)
p = in_pin.value()

# Digital output
out_pin = machine.Pin(0, machine.Pin.OUT)
# Set of clear using the same method
out_pin.value(1)    # sets it
Out_pin.value(0)    # clears it
```

File Handling

```
# Files are stored in a FAT filesystem in flash memory or on an SD card

>>> f = open('data.txt', 'w')
>>> f.write('some data')
9
>>> f.close()

>>> f = open('data.txt')
>>> f.read()
'some data'
>>> f.close()
```

Simple Networking

```
def http_get(url):
    _, _, host, path = url.split('/', 3)
    addr = socket.getaddrinfo(host, 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (path, host),
'utf8'))
    while True:
        data = s.recv(100)
        if data:
            print(str(data, 'utf8'), end='')
        else:
            break
    s.close()
```

Hobby Servomotor Control

```
import pyb

s1 = pyb.Servo(1)    # create a servo object on position X1
s2 = pyb.Servo(2)    # create a servo object on position X2

s1.angle(45)         # move servo 1 to 45 degrees
s2.angle(0)          # move servo 2 to 0 degrees

# move servo1 and servo2 synchronously, taking 1500ms
s1.angle(-60, 1500)
s2.angle(30, 1500)
```

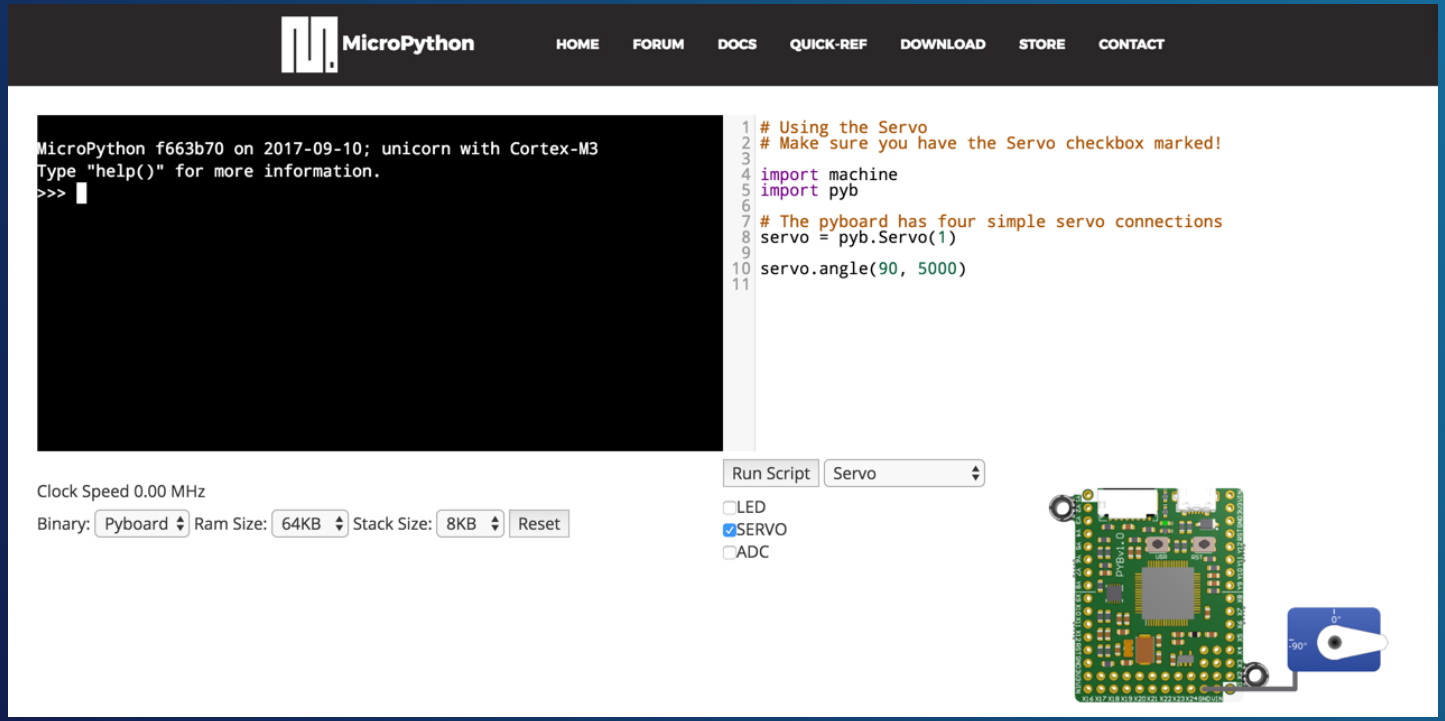
Interrupt Handling

```
>>> def callback(p):
...     print('pin change', p)

>>> from machine import Pin
>>> p0 = Pin(0, Pin.IN)
>>> p2 = Pin(2, Pin.IN)

>>> p0.irq(trigger=Pin.IRQ_FALLING, handler=callback)
>>> p2.irq(trigger=Pin.IRQ_RISING | Pin.IRQ_FALLING, handler=callback)
```

Power Control

```
import machine
# configure RTC.ALARM0 to be able to wake the device
rtc = machine.RTC()
rtc.irq(trigger=rtc.ALARM0, wake=machine.DEEPSLEEP)
# set RTC.ALARM0 to fire after 10 seconds (waking the device)
rtc.alarm(rtc.ALARM0, 10000)
# put the device to sleep
machine.deepsleep()
      ...
if machine.reset_cause() == machine.DEEPSLEEP_RESET:
    print('woke from a deep sleep')
else:
    print('power on or hard reset')
```

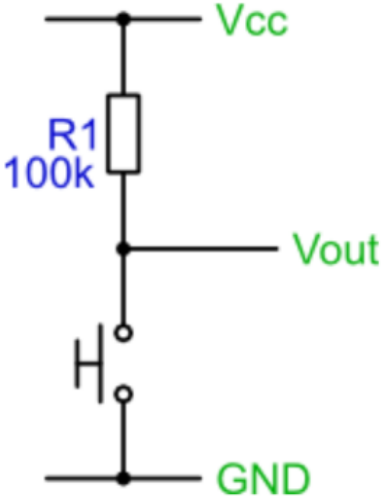
www.micropython.org/unicorn/



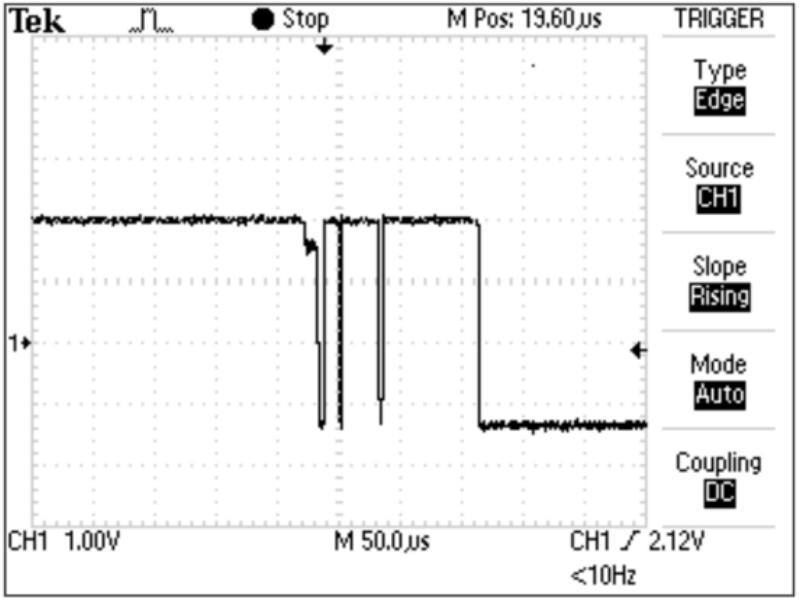
De-bouncing switches

A REAL-WORLD PROBLEM

Switches Aren’t Just On/Off



Simple switch pull-up circuit



Switch bounce produced on switch press

Solving The Problem

- Use a pure software solution
- Sample at regular intervals
  - I chose a 6ms sampling interval
- Switch is stable when N samples are the same
  - I chose N=12

Switch Class Definition

```
from machine import Timer, Pin

class Switch:

    def __init__(self, pin_no, name):
        self._state = 0
        self.value = 0
        self.pin = Pin(pin_no)
        self.name = name

    def output(self):
        "Return name of switch if closed, dot if open."
        return self.name if self.value else "."
```

Debouncer Class Definition (1)

```
class Debouncer:

    def __init__(self):
        self.switches = []
        self.timer = Timer(-1)
        self.timer.init(period=6, mode=Timer.PERIODIC, callback=self.tick)

    def register(self, switch):
        "Add a switch to the bank."
        self.switches.append(switch)
        switch.pin.init(mode=Pin.IN, pull=Pin.PULL_UP)
        return switch
```

Debouncer Class Definition (2)

```
def tick(self, _):
    "Examine input states and note debounced state changes."
    for switch in self.switches:
        bit = switch.pin.value()
        switch._state = ((switch._state << 1) | bit) & 0xfff
        # Latch state if last 12 samples were equal
        if switch._state == 0x000:   # switch pressed
            switch.value = True
        elif switch._state == 0xfff: # switch released
            switch.value = False
```

Switch Setup on ESP8266

```
def switches():
    "Report names of closed switches."
    return "".join(x.output() for x in (R, W, B, Y))

# Create a bank of four debounced switches
d = Debouncer()
Y = d.register(Switch(0, "Y")) # D3
R = d.register(Switch(14, "R")) # D5
W = d.register(Switch(12, "W")) # D6
B = d.register(Switch(13, "B")) # D7
```

Main Loop

```
# Background task loops reporting changes in any switch's state
state = switches()

while True:
    new_state = switches()
    if new_state != state:
        state = new_state
        print(state)
```

Live Demonstration!

Possibilities …

- Robot control
- Toys and games
- Weather stations
- Light patterns
- Science instrumentation/data collection

Final Thoughts

- Computers don't just belong in mathematics
- MicroPython gives children *direct, hands-on* experience
  - *Puts them in control*
- The younger they start, the more useful they will find computers

Questions?

Contact me:
sh@felix.com
@holdenweb

Slides and code available at
http://github.com/holdenweb/PyConIE2017

More Resources

github.com/micropython/

github.com/micropython/micropython/wiki

www.micropython.org/download

twitter.com/search?q=#micropython