MicroPython
Python na mikrokontrolerze

Damian Gadomski
damgad@gmail.com
@damgad

MicroPython

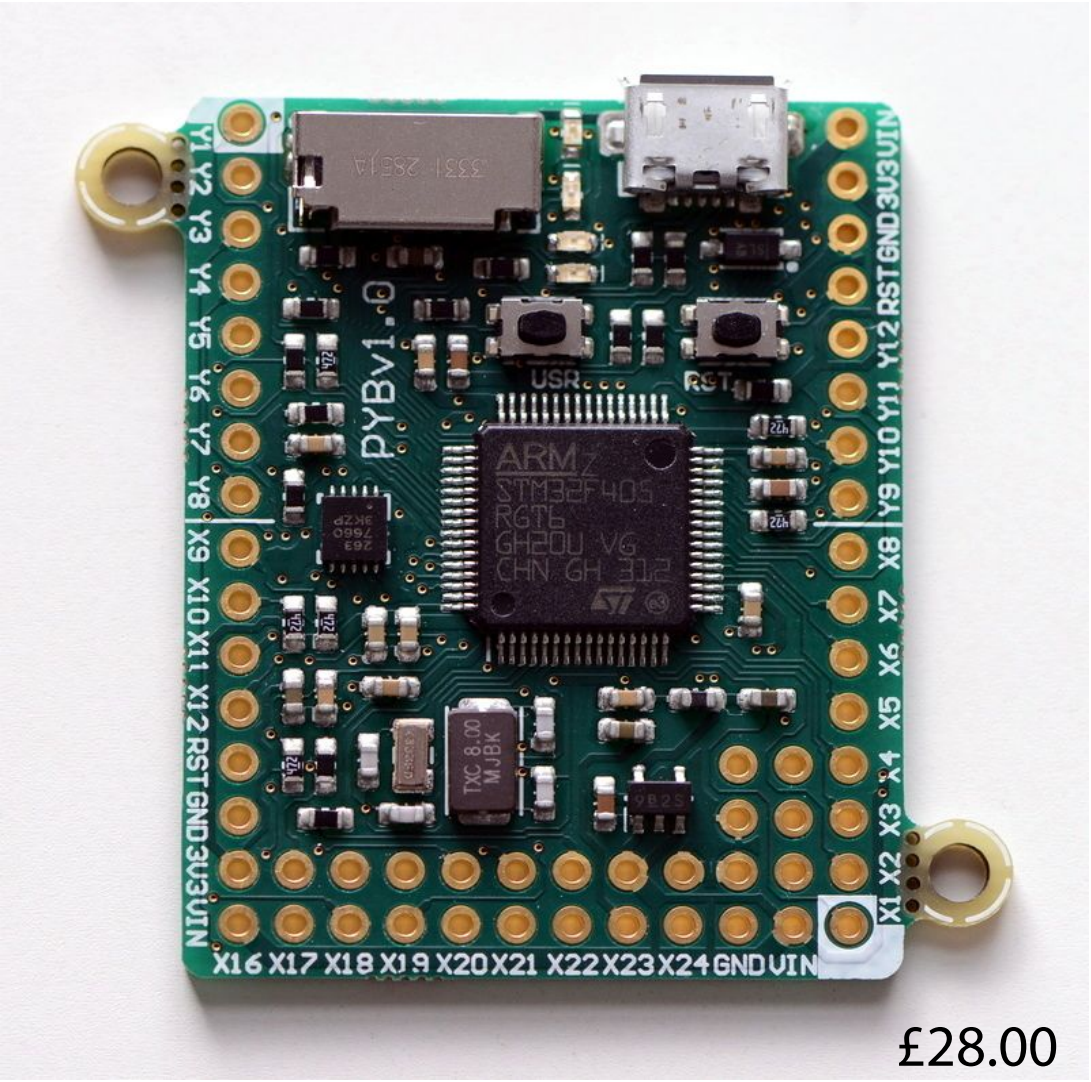
Python na ARM

Mini płytka ewaluacyjna
(diody, przyciski, GPIO, akcelerometr, RTC, USB)

Mini Demo

...

PyBoard



£28.00

ARM 32-bit Cortex-M4 CPU
(192KB RAM, 1MB ROM, do 168 MHz)

USB
micro SD

akcelerometr
RTC

GPIO, I2C, USART, DAC, ADC

Micro Python

Python 3.4

- zoptymalizowany dla mikrokontrolerów

- zaimplementowany cały język

Bez standardowych bibliotek

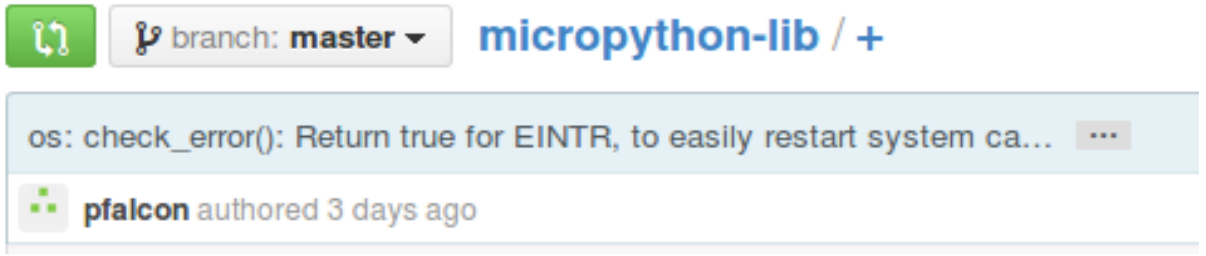
– są one stopniowo (intensywnie!) dopisywane

Biblioteki – wbudowanie w µP

- `cmath` – mathematical functions for complex numbers
- `gc` – control the garbage collector
- `math` – mathematical functions
- `os` – basic "operating system" services
- `select` – wait for events on a set of streams
- `struct` – pack and unpack primitive data types
- `sys` – system specific functions
- `time` – time related functions

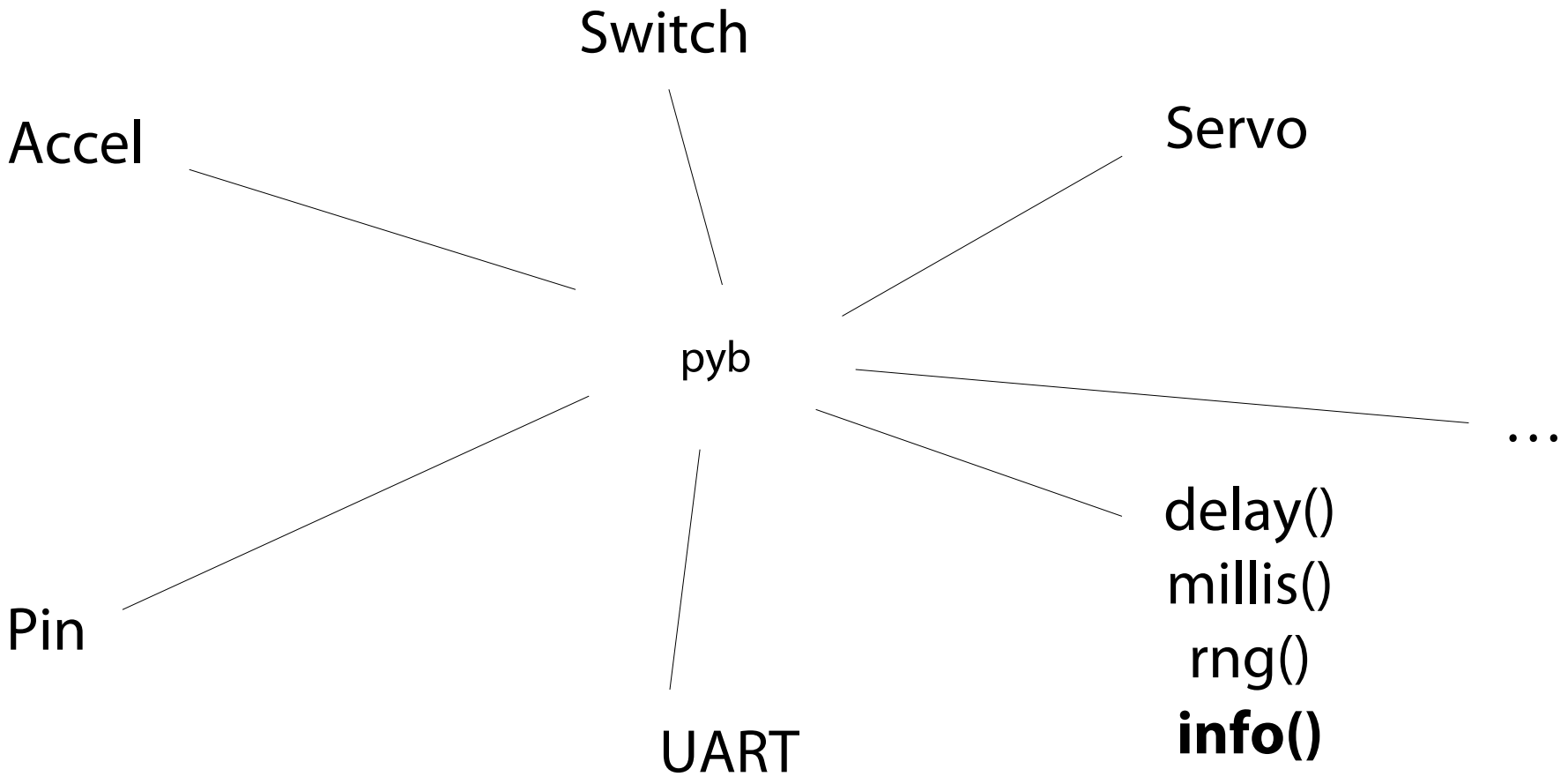
(micro)Biblioteki

Bardzo żywy temat:



- `ubinascii` – binary/ASCII conversions
- `uctypes` – access C structures
- `uhashlib` – hashing algorithm
- `uheapq` – heap queue algorithm
- `ujson` – JSON encoding and decoding
- `ure` – regular expressions
- `usocket` – socket module
- `uzlib` – zlib decompression

moduł pyb



Przerwania - Callbacki

```
pyb.Switch().callback(lambda: print("Nacisnieto mnie!"))
```

Przerwania - Callbacki

```
tim = pyb.Timer(4)
tim.init(freq=2)
tim.callback(lambda t:pyb.LED(1).toggle())
```

```
pyb.ExtInt(pin, pyb.ExtInt.IRQ_FALLING, pyb.Pin.PULL_UP, callback)
```

USB („deploy”)

System plików (sd lub flash) – wystarczy włożyć kartę

main.py
(boot.py)

Debugowanie utrudnione (zamontuj, zmień, odmontuj, reset)

Port micropythona na Unixa – nie do końca...

Demo „servo”

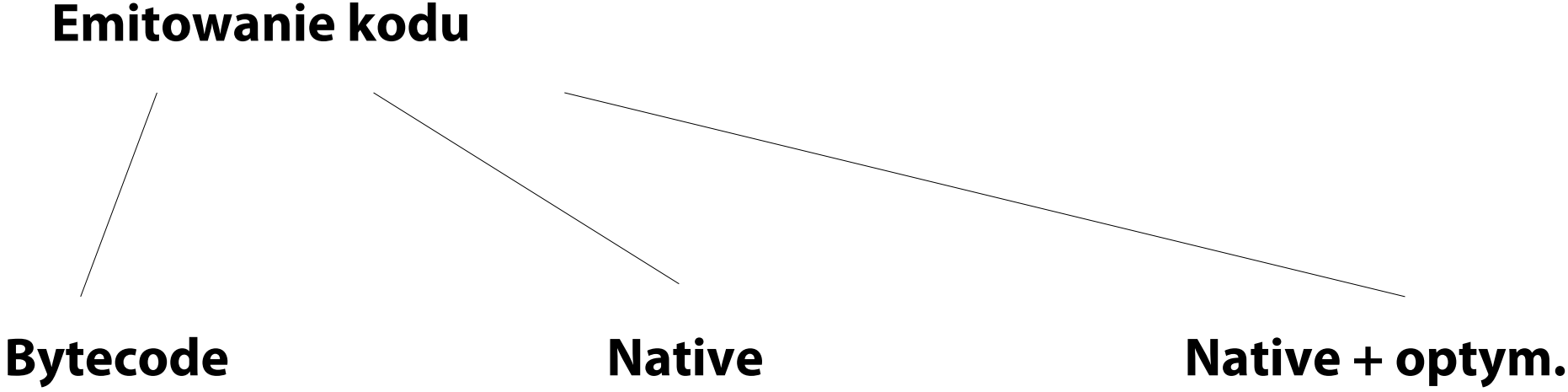
```
servo1 = pyb.Servo(1)
accel = pyb.Accel()

while True:
    servo1.angle(3*accel.x())
```

Kompilacja - wydajność

Kompilowanie kodu:

• Parsowanie → kompilator * 3 (zmienne, zakres, zajętość stosu, emitowanie kodu)



Wydajność – kompilacja

```
def speed_bytecode():
    for i in range(1000000):
        pyb.led(True)
        pyb.led(False)
```

44B RAM, 10.4 s.

```
@micropython.native
def speed_native():
    for i in range(1000000):
        pyb.led(True)
        pyb.led(False)
```

126B RAM, 6.3 s.

```
@micropython.viper
def speed_viper():
    for i in range(1000000):
        pyb.led(True)
        pyb.led(False)
```

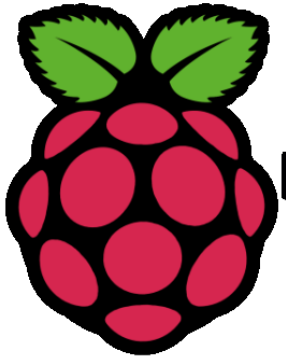
114B RAM, 5 s.

Jak się ma do tego RaspPi, Arduino?

(16 MHz)

Odpowiedni kod w C:

około 7 s



RaspberryPi

(700 MHz)

Odpowiedni kod w C:

około 300 ms

Python: 18 s

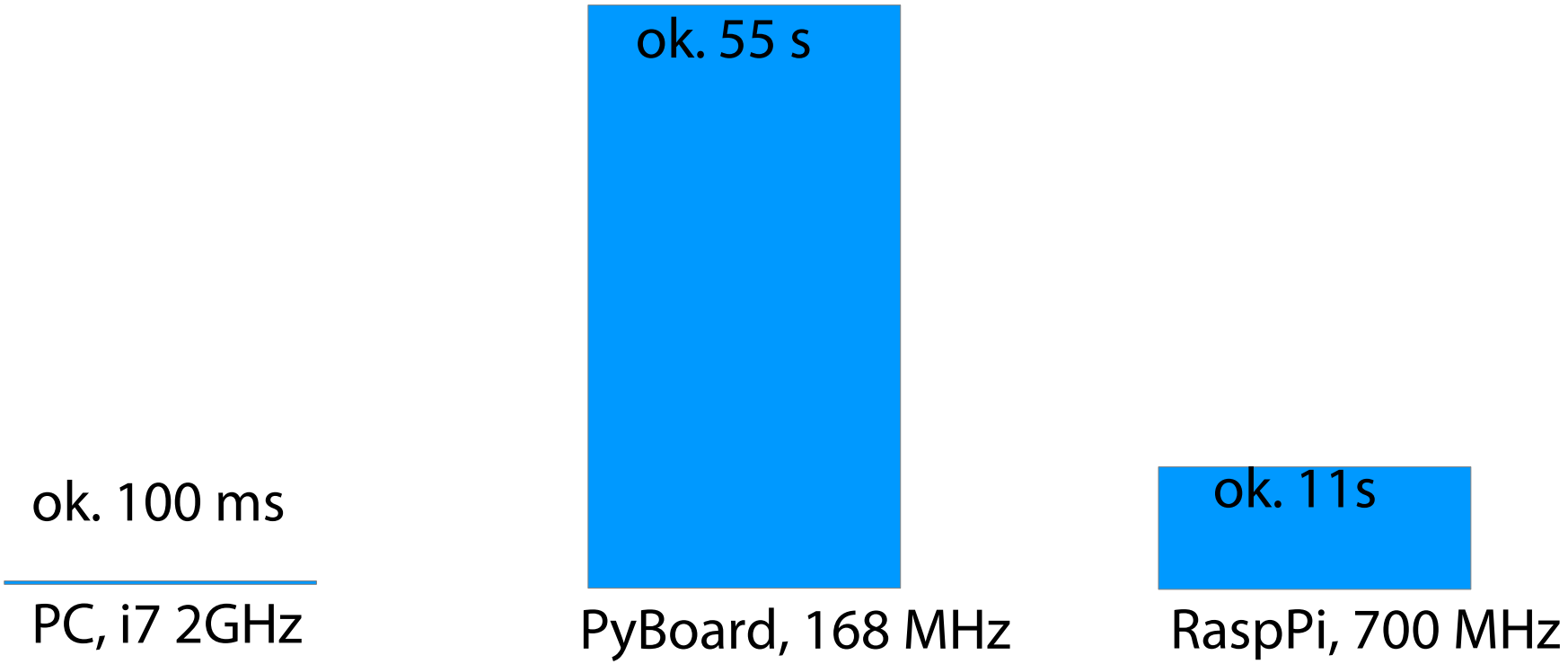
Assembler

```
@micropython.asm_thumb
def led_on():
    movwt(r0, stm.GPIOA)
    movw(r1, 1 << 13)
    strh(r1, [r0, stm.GPIO_BSRRL])
```

Niestety, pisanie w C możliwe tylko modyfikując firmware... :(

Wydajność

Test pierwszości (alg. Rabina – Millera)



Pobór energii

Zasilanie z USB, lub zasilacz 3,6 – 10V

Pobór prądu – od kilku do kilkudziesięciu mA

pyb.freq()

pyb.wfi()

wyłączyć USB

1000mAh /
30mA =~ 33h

Dziękuję!

PiO