# Python sur microcontrôleur

## Un rêve ?

Matthieu Falce

Inria Lille – Equipe Mjolnir

April 22, 2015

# Plan

# Projet Touchit

Figure : Stimtac

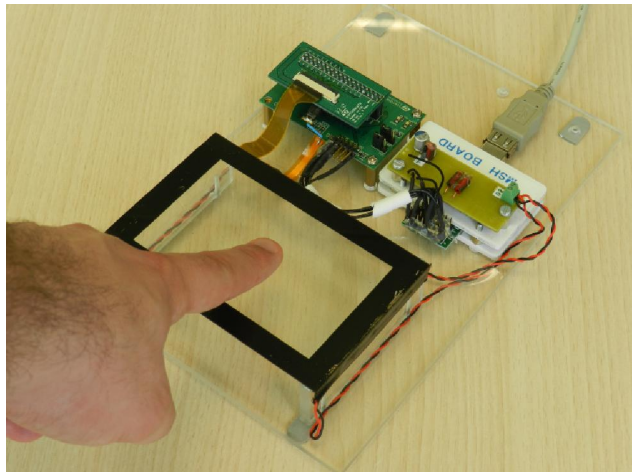


Figure : Stimtac transparent

- Création d'un dispositif de stimulation tactile
- Portage du dispositif sur une autre catégorie de microcontrôleurs
- Ajout d'un capteur capacitif pour la détection des doigts

# Projet Touchit

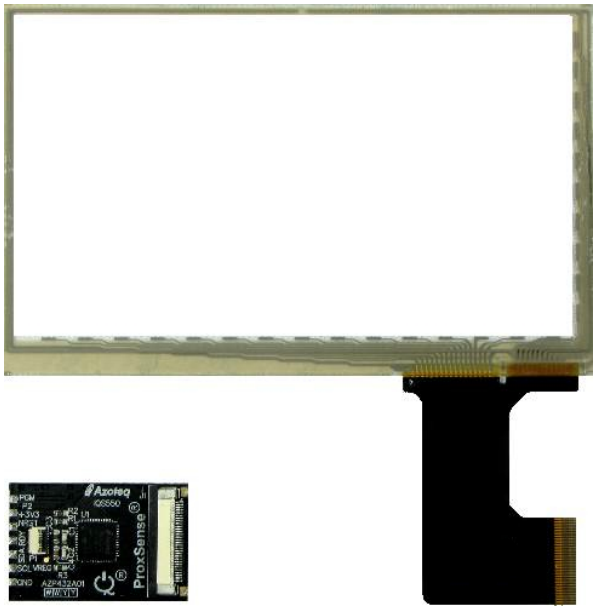



Figure : Capteur capacitif

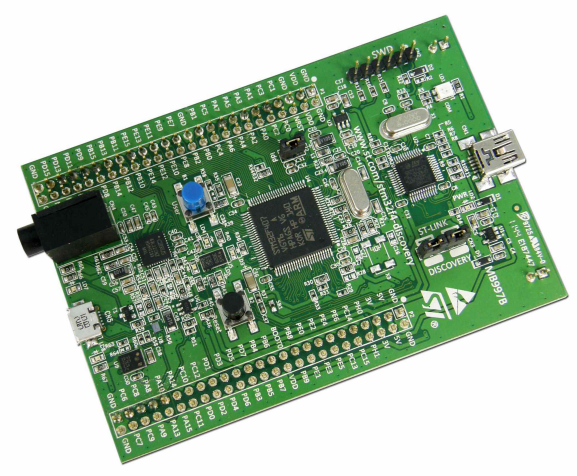


Figure : STM32F4 discovery

# Python

- langage de programmation haut niveau
- syntaxe claire et facile

Example :

```
for i in range(10):
    print(str(i) + " - Python YEAH!")
```

# Un microcontrôleur ?

"Un ordinateur en miniature pour intéragir avec le monde réel"

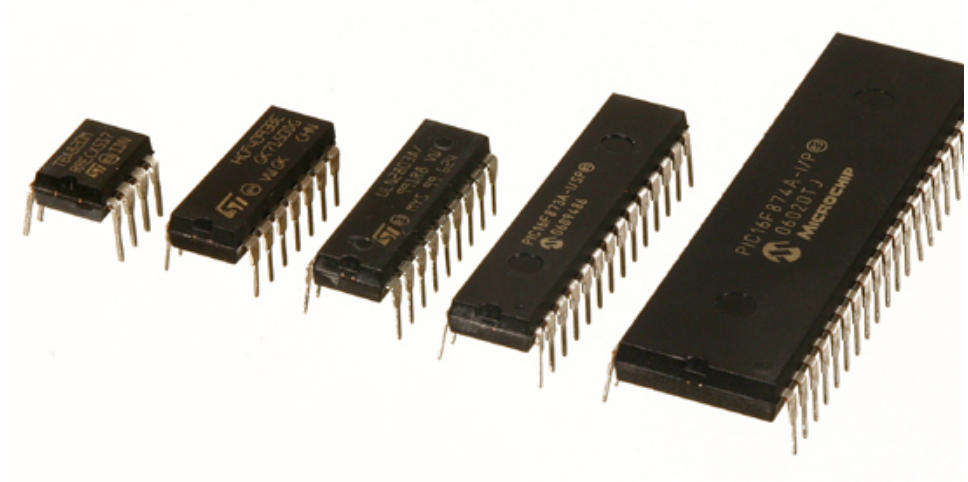



Figure : Exemples de microcontrôleurs

# Arduino – AVR Atmel

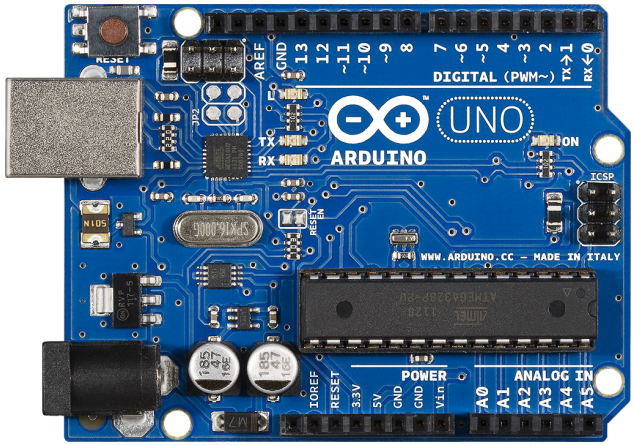
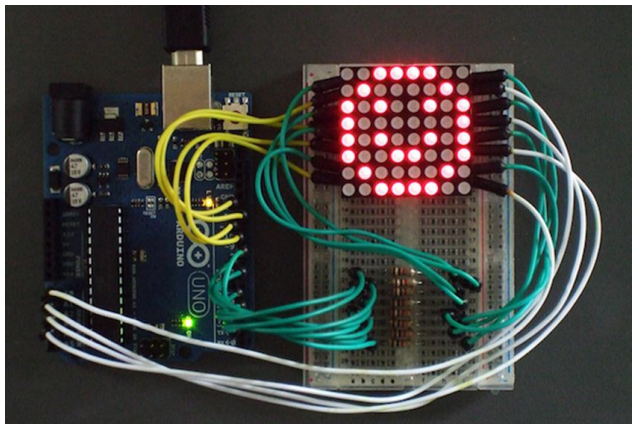
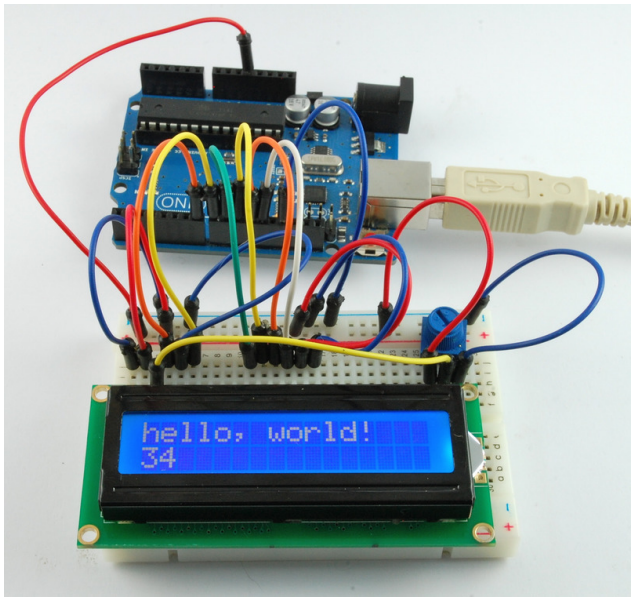


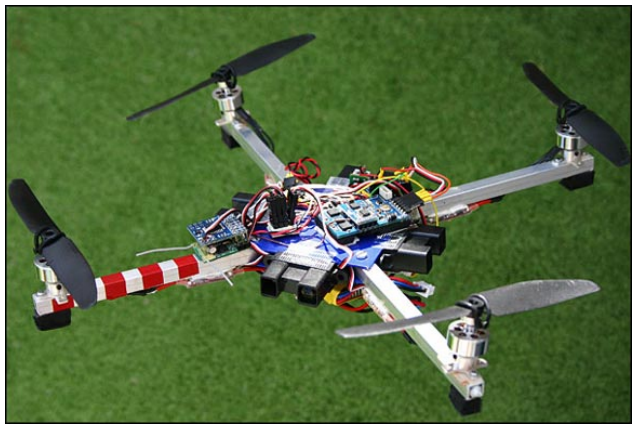

Figure : Arduino Uno

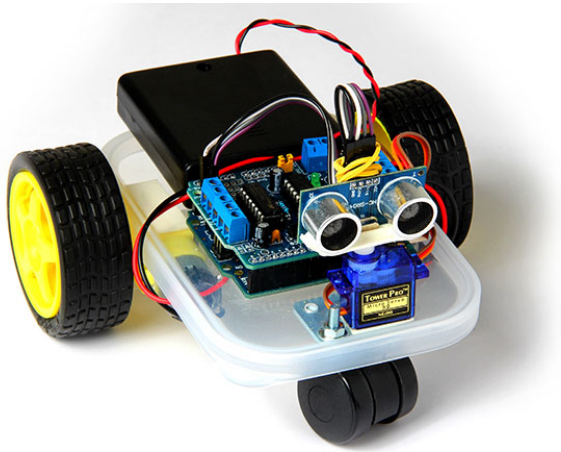
# Arduino – AVR Atmel





# Arduino – AVR Atmel





# Arduino – AVR Atmel

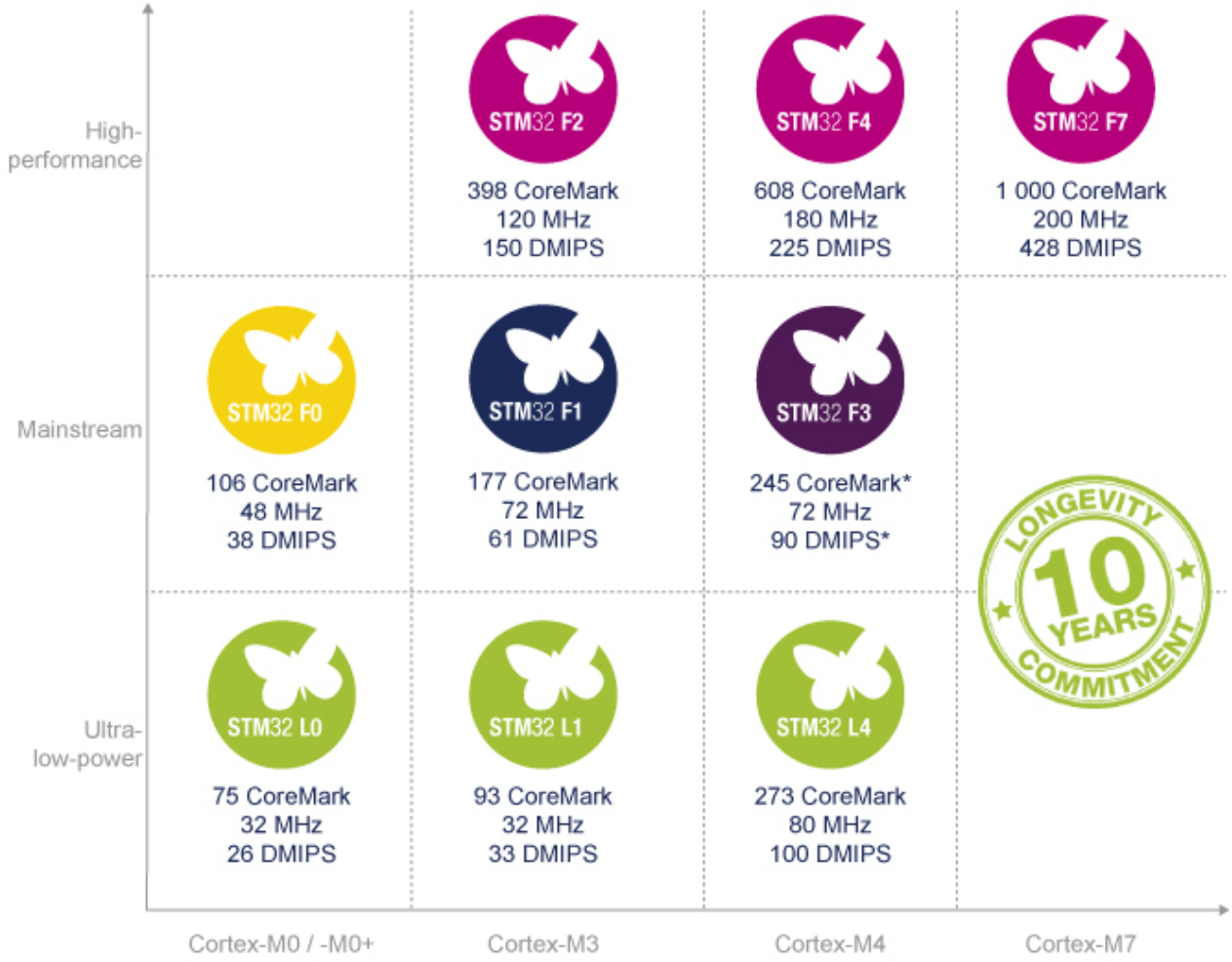
```
// source: blink example arduino

int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

# STM32




* from CCM-SRAM

# STM32


High-performance
Mainstream
Ultra-low-power
STM32 F2
398 CoreMark
120 MHz
150 DMIPS
STM32 F4
608 CoreMark
180 MHz
225 DMIPS
STM32 F7
1 000 CoreMark
200 MHz
428 DMIPS
STM32 F0
106 CoreMark
48 MHz
38 DMIPS
STM32 F1
177 CoreMark
72 MHz
61 DMIPS
STM32 F3
245 CoreMark*
72 MHz
90 DMIPS*
LONGEVITY
10
YEARS
COMMITMENT
STM32 L0
75 CoreMark
32 MHz
26 DMIPS
STM32 L1
93 CoreMark
32 MHz
33 DMIPS
STM32 L4
273 CoreMark
80 MHz
100 DMIPS
Cortex-M0 / -M0+
Cortex-M3
Cortex-M4
Cortex-M7
* from CCM-SRAM

# STM32

- Utilisé dans l'industrie
- Bien plus puissant que les arduino
- Peut utiliser un OS temps réel
- Possibilité de débugage

# STM32

- Utilisé dans l'industrie
- Bien plus puissant que les arduino
- Peut utiliser un OS temps réel
- Possibilité de débugage

Avec de grands pouvoirs viennent de grandes responsabilités...

# STM32

```
// source: http://pramode.net/fosstronics/minimal-c-program.txt

#define GPIOB_CRH  (*((volatile unsigned long*)(0x40010c00 + 0x4)))
#define GPIOB_BSRR  (*((volatile unsigned long*)(0x40010c00 + 0x10)))
#define RCC_APB2ENR  (*((volatile unsigned long*)(0x40021000 + 0x018)))


__asm__(".word 0x20001000");
__asm__(".word main");
main()
{
        unsigned int c = 0;

        RCC_APB2ENR = (1 << 3);
        GPIOB_CRH = 0x44444414;

        while(1) {
                GPIOB_BSRR = (1 << 9); // ON
                for(c = 0; c < 100000; c++);
                GPIOB_BSRR = (1 << 25); // OFF
                for(c = 0; c < 100000; c++);
        }

}
```

# STM32

```
// source: http://sigalrm.blogspot.fr/2012/09/stm32f4-discovery-quick-start-guide.html

#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"
GPIO_InitTypeDef  GPIO_InitStructure;
int main(void)
{
    /* GPIOG Periph clock enable */
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

    /* Configure PD12 in output mode */
    GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_12;
    GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_OUT;
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
    GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOD, &GPIO_InitStructure);

    /* Set PD12 high */
    GPIO_SetBits(GPIOD, GPIO_Pin_12);

    /* Do nothing */
    while (1) {
    }
}
```

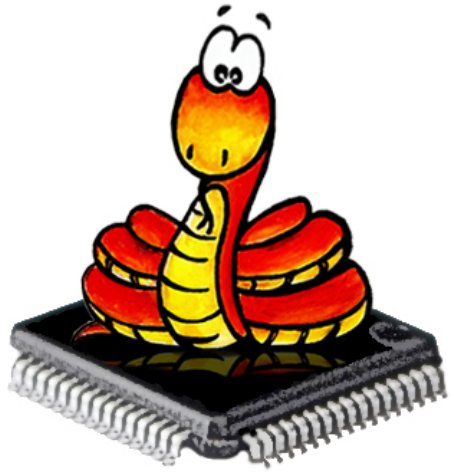
> *“Les psychiatres le détestent.*
> *Cet homme a découvert un secret simple pour ne pas devenir fou en utilisant la STM32.*
>
> *— Pr John Doe*

# Réduire le ticket d'entrée ?

Utilisation de langages haut niveau

- STM32F4
  - Python (MicroPython)
- STM32F1
  - Javascript (espruino)
  - Lua (eLua)

# MicroPython



- Python 3.4
- La plupart des fonctionnalités supportées (asyncio, décorateurs, POO)
- Interpréteur
- Code lancé au démarrage
- Communauté très active

# MicroPython

```
import pyb

led1 = pyb.LED(1)

while True:
    led1.toggle()
    pyb.delay(1000)
```

# MicroPython

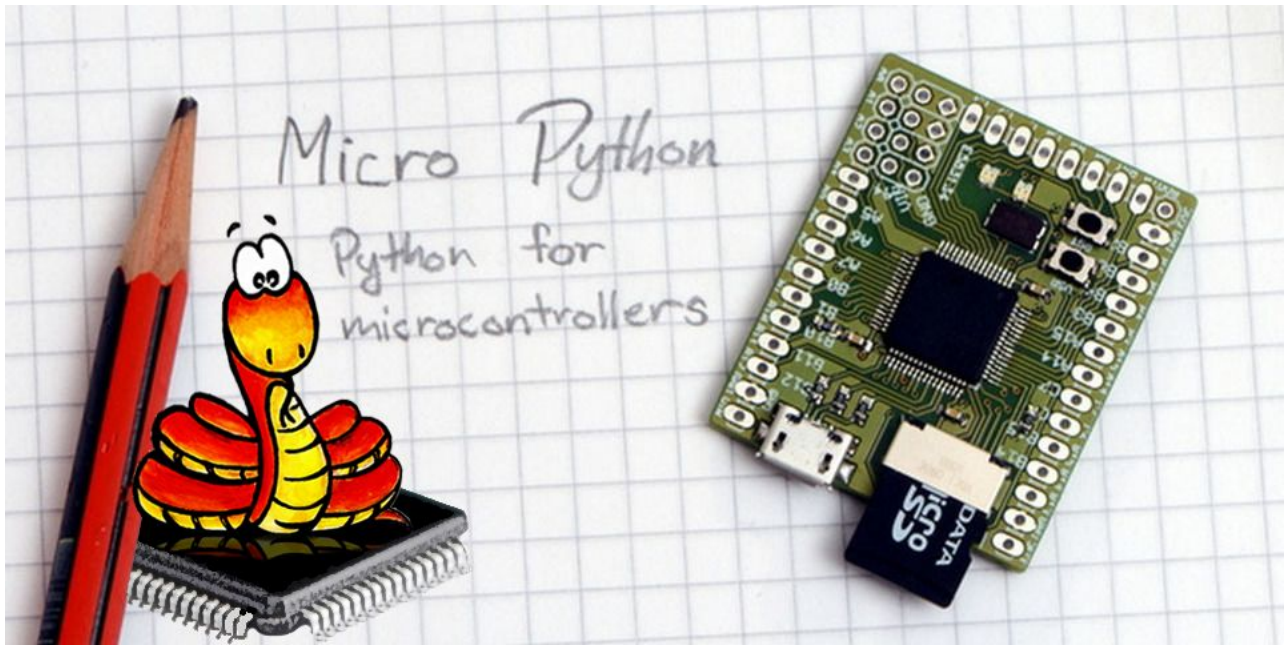
```
import pyb

led1 = pyb.LED(1)

while True:
    led1.toggle()
    pyb.delay(1000)
```

# Le projet

- Kickstarter lancé par Damien George en 2013
- £97,803 récoltés
- *Pyboard* (STM32F405)
- Libres et disponibles sur github



# Le projet

- plus de 30 cartes visées
- 5 ports fonctionnels actuellement
- STMDiscovery, CEB40

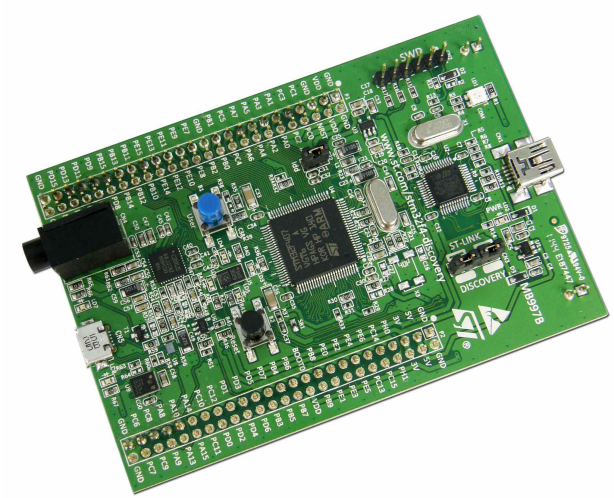


Figure : STM32F405 Discovery

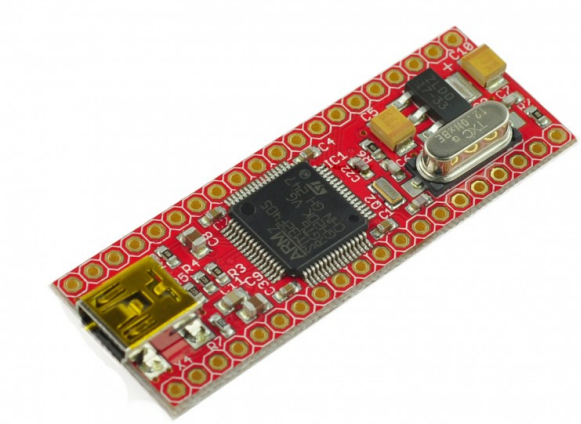


Figure : CERB40

# MicroPython – Avantages

- interpréteur
- utilise la syntaxe python 3.4 classique
- modulable
- possibilité d'utiliser du code natif (optimisé)

# MicroPython – Inconvénients

- pas d'allocation de mémoire dans les interruptions
- instabilité (problèmes de reboot, $I^2C$ instable)
- vitesse
- pas de threads (MAIS utilisation de `select` ou asynchrone)

# Glossaire

- HID:
  - Human Interaction Device
  - Données en *little endian*
  - Norme USB
  - Descripteur : comment sont formés les messages
  - Report : les messages

# Démos

Démos sur le CERB40

- Flashage de MicroPython
- Lecture d'un port analogique (joystick)
- Création d'une souris
- Création d'un joystick

# Conclusion

Quelle place pour cet outil ?

- Pas pour des applications critiques
- Prototypage
- Découverte des STM32

# Conclusion

Quelle place pour cet outil ?

- Pas pour des applications critiques
- Prototypage
- Découverte des STM32

C++ on Arduino Pro Mini

(Atmega328p @ 18MHz)

```
#include <Arduino.h>
void setup() {
    Serial.begin(115200);
    uint32_t endTime = millis() + 10000;
    uint32_t count = 0;
    while (millis() < endTime)
        count++;
    Serial.print("Count: ");
    Serial.println(count);
}
void loop() {
}
```

Count: 4,970,227

Micropython on pyboard

(stm32f407 @ 168MHz)

```
def performanceTest():
    millis = pyb.millis
    endTime = millis() + 10000
    count = 0
    while millis() < endTime:
        count += 1
    print("Count: ", count)
performanceTest()
```

Count: 2,890,723

Source:

https://github.com/micropython/micropython/wiki/Performance