

### Mini BiO

- Luciano Ramalho, programador-repentista
- Desenvolvedor Web desde 1994
  - IDG Now, Brasil Online, UOL/BOL, AOL Brasil...
- Professor
  - para profissionais: Python.pro.br
  - para crianças e adolescentes: Oficina Turing
- Palestrante: FISL, PyCon US, OSCON, TDC...
- Co-fundador do Garoa Hacker Clube

#### PINGO: FEITO NO GAROA

- Garoa Hacker Clube (garoa.net.br)
- Um hackerspace
  - == laboratório comunitário independente
- Em São Paulo (próximo do metrô Pinheiros)
- Mantido por uma associação sem fins lucrativos operada pelos associados
  - hoje somos 46 associados pagando mensalidades de R\$ 60 ou R\$ 100
  - isso paga o aluguel, a manutenção da casa e permite a compra de equipamentos





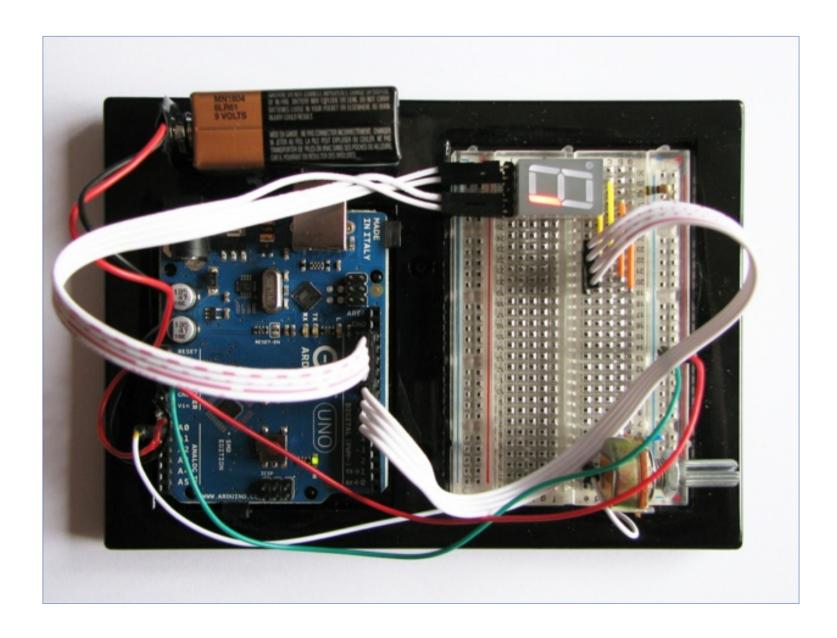
# O "PROBLEMA"

### Pinos GPio

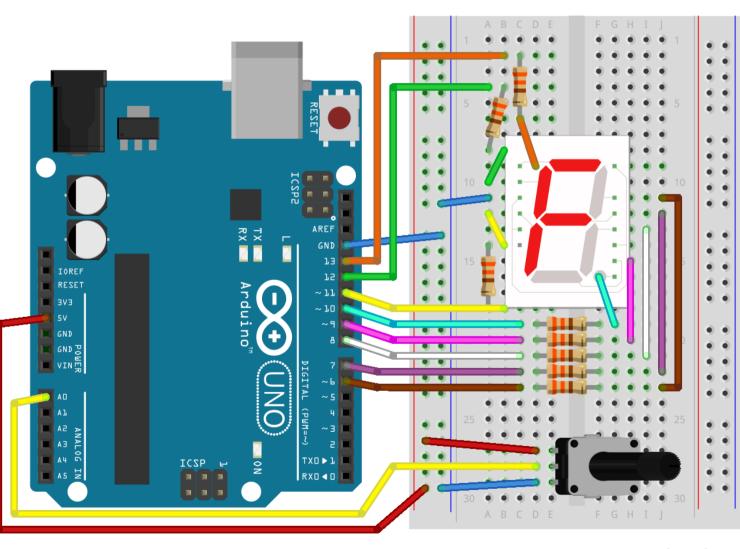
- Todas as placas interessantes para computação física possuem pinos GPIO
  - General Purpose I/O
- Pinos digitais I/O
  - 0/1 == HIGH/LOW
- Pinos analógicos
  - conversor A/D
- Pinos digitais com suporte a PWM
  - substituto saída analógica sem conversor D/A



# DOJO COM ARDUINO



# DOJO COM ARDUINO



HACKER CLUBE

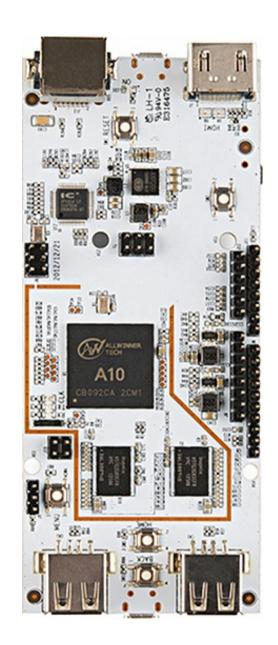
fritzing

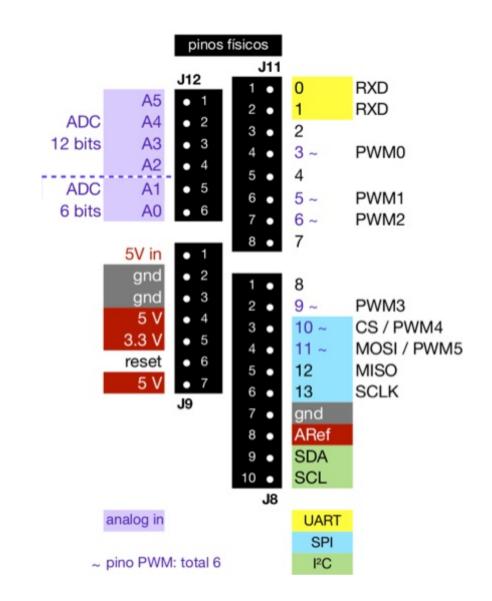
# **PCDUINO**



	pcDuino
SoC	Allwinner A10
CPU	ARM Cortex A8
<b>3. 3</b>	(ARMv7 + NEON SIMD)
GPU	Mali 400
clock	l GHz
so	GNU/Linux (vários)
	Android
RAM	I GB
Flash onboard	2 GB
SD card	micro-SD
GPIO	14
PWM	6
ADC	6
USB client	I
USB host	I
Ethernet	10/100
Wifi	X
HDMI	HDMI
video composto	X
audio	via HDMI
preço USD	59
Arduinos	1.7
open hardware	?

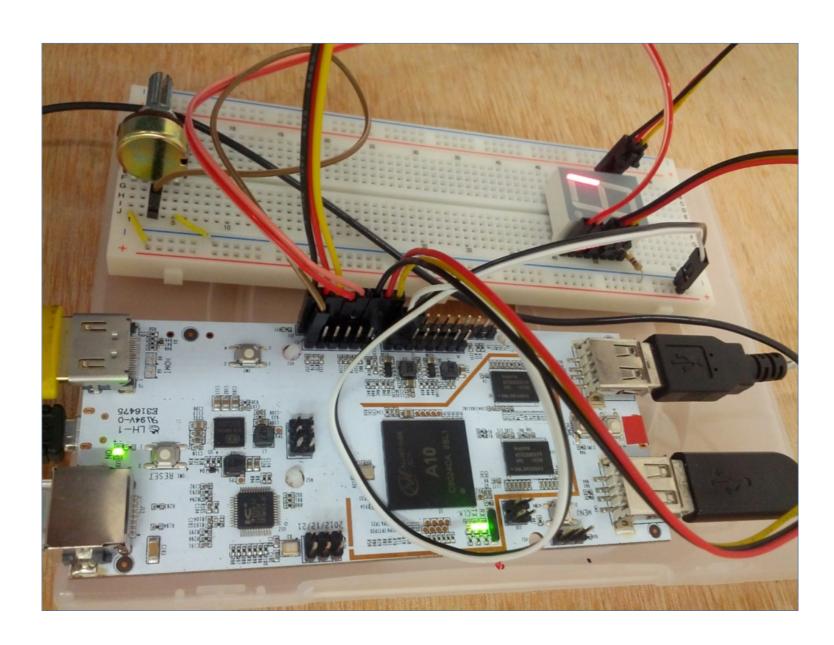
### **PCDUINO**



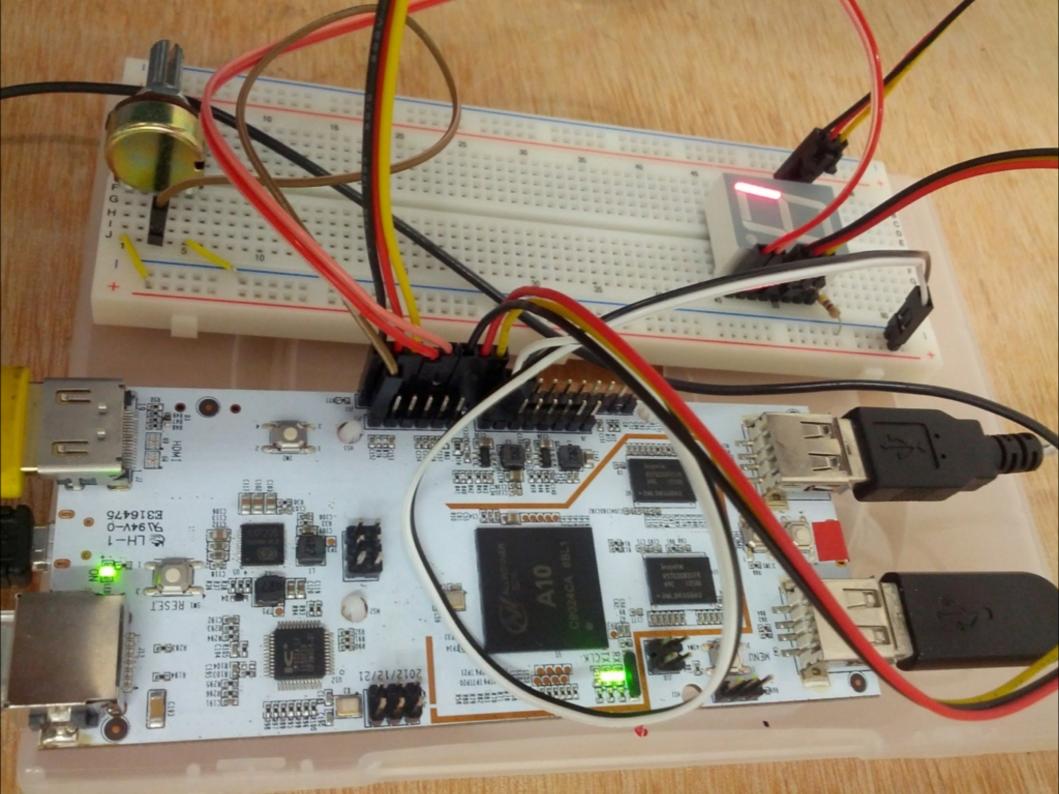


GAROA HACKER CLUBE

# DOJO COM PCDUINO



GAROA HACKER CLUBE



# RASPBERRY Pi

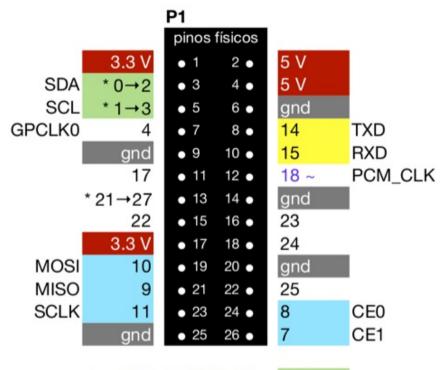


	Raspberry Pi mod. B	
SoC	Broadcom BCM2835	
CPU	ARMII (ARMv6)	
GPU	VideoCore IV	
clock	700 MHz	
so	GNU/Linux (vários)	
RAM	512 MB	
Flash onboard	X	
SD card	SD	
GPIO	17 (+4 no P5)	
PWM	I	
ADC	X	
USB client	X	
USB host	2	
Ethernet	10/100	
Wifi	X	
HDMI	HDMI	
video composto	RCA	
audio	HDMI + plug 3.5mm	
preço USD	35	
Arduinos	1.0	
open hardware	X	



#### RASPBERRY Pi



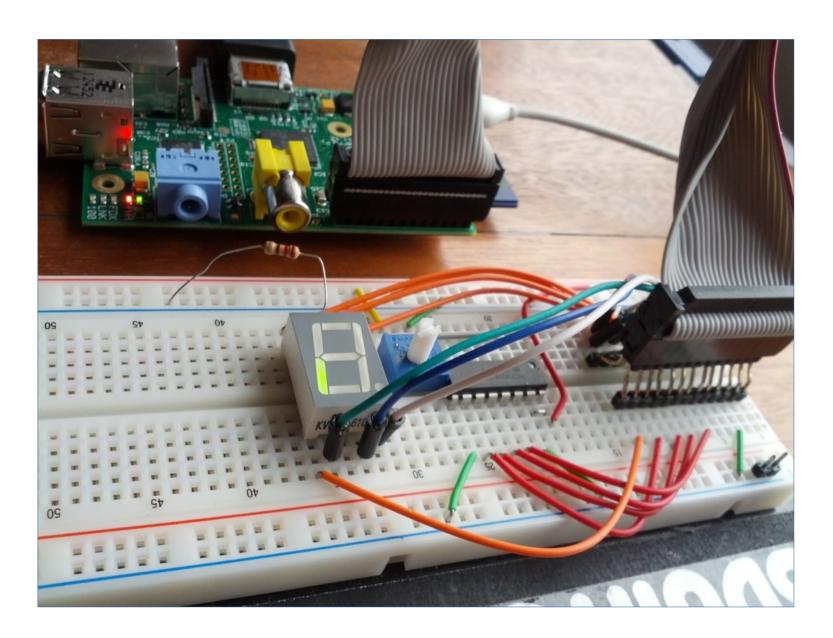


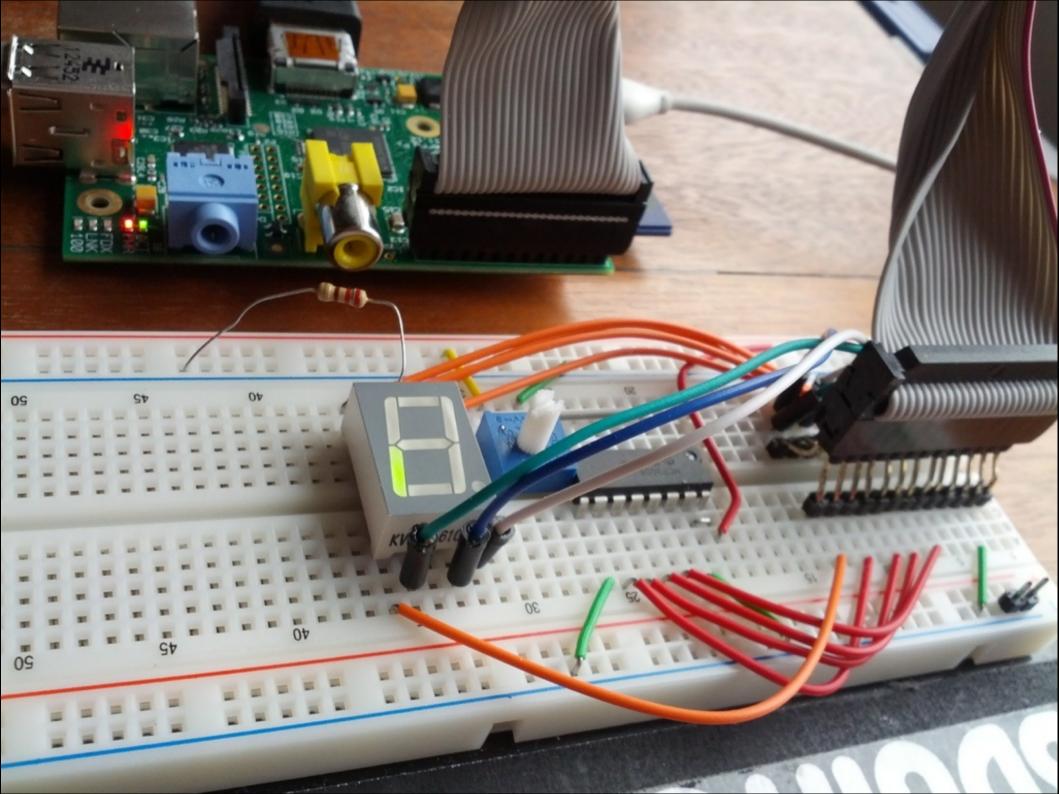
~ pino PWM: 12 (GPIO\_18)

\* pinos GPIO renumerados rev.1→rev.2

I<sup>2</sup>C UART SPI

# DOJO COM RASPBERRY Pi





# BEAGLEBONE BLACK



	Beaglebone Black	
SoC	Texas Instruments Sitara AM335x	
СРИ	ARM Cortex A8 (ARMv7 + NEON SIMD)	
GPU	PowerVR SGX 530	
clock	l GHz	
so	GNU/Linux (vários) Android	
RAM	512 MB	
Flash onboard	2 GB	
SD card	micro-SD	
GPIO	66	
PWM	8	
ADC	7	
USB client	I	
USB host	I	
Ethernet	10/100	
Wifi	×	
HDMI	micro-HDMI	
video composto	X	
audio	via HDMI	
preço USD	45	
Arduinos	1.3	
open hardware	<b>✓</b>	

#### BEAGLEBONE BLACK

P9 pinos físicos 3.3 V VDD 5 V SYS 5 V PWR BUT 10 • UART4 RXD 12 • UART4 TXD • 13 14 • 1-16 16 • I2C1\_SCL 18 • • 17 12C2 SCL • 19 20 • UART2 TXD 22 0 1-17 23 24 . \* 3-21 25 26 • 3-19 • 27 28 • SPI1\_D0 29 30 • SPI1\_SCLK · 31 32 · AIN4 • 33 34 • AIN6 35 36 • AIN<sub>2</sub> 38 • AIN0 40 • <sup>2</sup> CLKOUT2, 3-20 • 43 45
 46

gnd 3.3 V 5 V VDD 5 V SYS SYS RESETN 1-28 EHRPWM1A ~ EHRPWM1B ~ I2C1\_SDA 12C2 SDA UART2 RXD UART1\_TXD UART1\_RXD SPI1 CS0 ~ SPI1\_D1 VADC **AGND** AIN5 AIN3 AIN1 0-7 ~, 3-18 2 gnd gnd

513BBBK 000 dd-10/100 Etherne

pinos físicos gnd gnd 2 • 1-6 • 3 1-7 1-2 1-3 • 5 6 • TIMER4 TIMER7 TIMER5 TIMER6 10 • 1-13 12 • 1-12 ~ EHRPWM2B 0-2613 14 • 1-15 1-14 16 • 0-27 18 • 2-1 ~ EHRPWM2A 1-31 1-5 • 21 22 • 1-4 1-1 23 24 • 1-0 1-29 25 26 • 2-22 2-24 • 27 28 2-25 29 UART5\_CTSN UART5\_RTSN 31 32 • UART4 RTSN UART3\_RTSN 33
 34 UART4\_CTSN UART3\_CTSN 35
 36 **UART5 TXD** UART5 RXD 2-12 2-13 2-10 2-11 2-9 2-27 x-yy pino GPIOX\_YY UART

timers

~ pino PWM: 13, 19

**P8** 

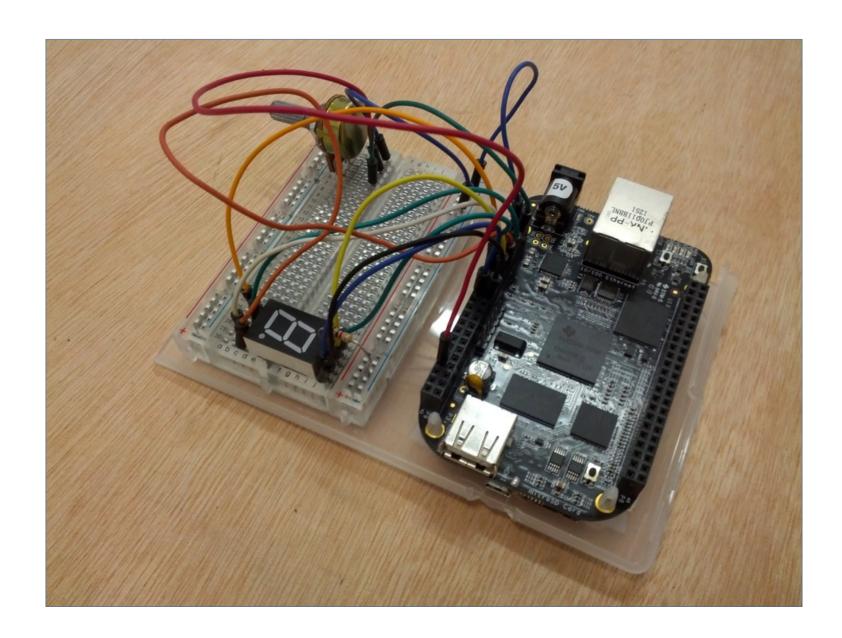
x-yy pino GPIOX\_YY

~ pino PWM: 14, 16, 28, 422 (0-7)

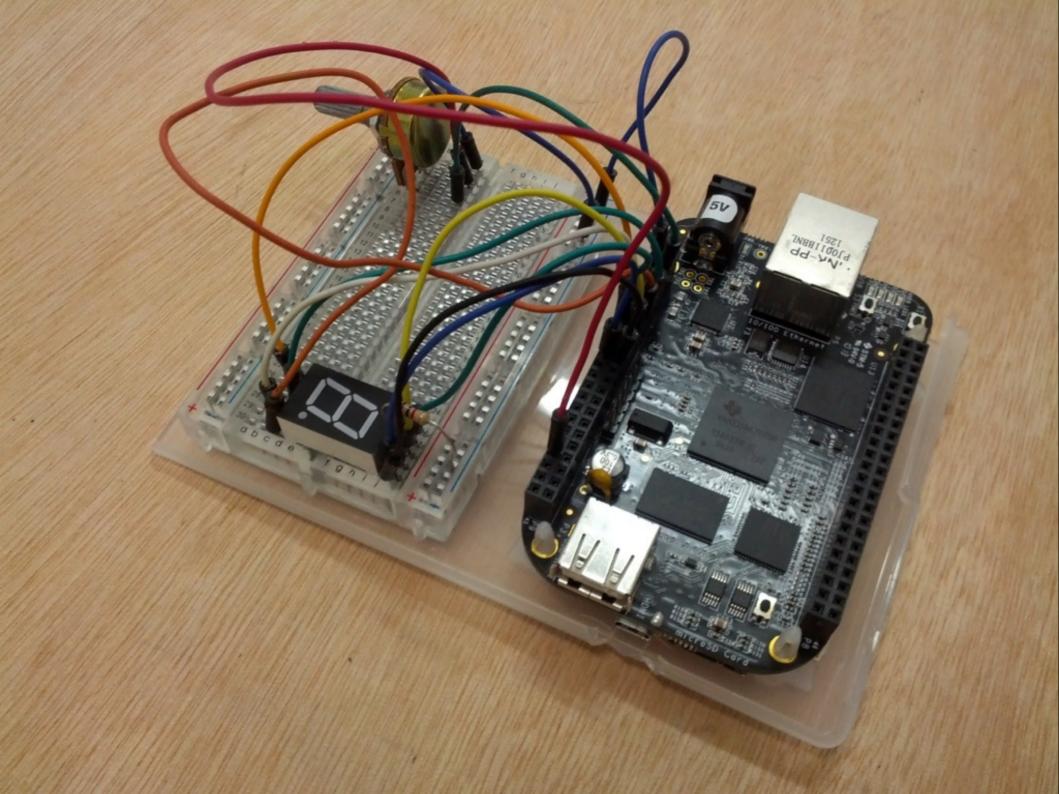
\* em uso: oscilador audio HDMI

<sup>2</sup> pino físico ligado a dois pinos lógicos analog input I<sup>2</sup>C UART SPI misc.

# DOJO COM BBB







#### PROGRAMANDO PINOS

- Quase todas essas placas são programáveis em **Python**
- Python embarcado, naquelas que rodam
   GNU/Linux embarcado
- Python em outro computador, controlando remotamente
  - ex: Arduino via serial com protocolo
     Firmata e biblioteca pyMata



```
import time, os
GPIO_PATH = os.path.normpath('/sys/devices/virtual/misc/gpio/')
ADC_PATH = os.path.normpath('/proc/')
INPUT = LOW = "0"
                                       SysFS: pinos GPIO
OUTPUT = HIGH = "1"
                                      mapeados em arquivos
def pin_mode(pin, mode):
   with open(GPIO_PATH+'mode/gpio%s' % pin, 'w') as f:
        f.write(mode)
def digital_write(pin, value):
   with open(GPIO_PATH+'pin/gpio%s' % pin, 'w') as f:
        f.write(str(value))
def analog read(pin):
   with open(ADC_PATH+'adc%d' % pin) as f:
        f.seek(0)
        return int(f.read(16).split(':')[1])
def setup():
   for i in range(18):
        pin_mode(i, OUTPUT)
                                       Nenhuma biblioteca
        digital_write(i, LOW)
                                      especial é usada neste
                                              script!
setup()
while True:
   for i in [0, 1, 7, 5, 4, 2]:
        digital_write(i, 1)
                                        PCDUINO
        delay = analog_read(5)/4096.0
        time.sleep(delay)
        digital_write(i, 0)
```

```
import atexit
                                          Duas bibliotecas
import time
import RPi.GPIO as GPIO
                                        específicas: RPi.GPIO
import spi
                                        e spi (feita no Garoa)
# executar cleanup ao sair
atexit.register(GPIO.cleanup)
# usar numeração lógica dos pinos
GPIO.setmode(GPIO.BCM)
DISPLAY = [17, 4, 9, 11, 7, 27, 22, 10]
SPI CLK = 18
SPI MISO = 23
SPI MOSI = 24
SPI CS = 25
conversor ad = spi.Mcp3008(SPI CLK, SPI MISO, SPI MOSI, SPI CS)
CANAL POTENCIOMETRO = 1
for led in DISPLAY[:6]:
   GPIO.setup(led, GPIO.OUT)
GPIO.output(led, 0)
   GPIO.output(led, 0)
while True:
   for led in DISPLAY[:6]:
       GPIO.output(led, 1)
       atraso = conversor_ad.read(CANAL_POTENCIOMETRO)/1000.0
       time.sleep(atraso)
       GPIO.output(led, 0)
```

```
import Adafruit_BBIO.GPIO as GPIO
import Adafruit_BBIO.ADC as ADC
ADC.setup()
from time import sleep
pinos = [16, 21, 22, 13, 12, 11]
for pino in pinos:
    GPIO.setup("P9_" + str(pino), GPIO.OUT)
while True:
    for pino in pinos:
        GPIO.output("P9_" + str(pino), GPIO.HIGH)
        tempo = ADC.read('P9_39')
        print tempo
        sleep(tempo)
        GPIO.output("P9_" + str(pino), GPIO.LOW)
```



#### BEAGLEBONE BLACK



# NOSSA SOLUÇÃO



DEMO

#### BLINK: MODO INTERATIVO

```
>>> from pingo import *
>>> ard = detect.MyBoard()
>>> ard
<ArduinoFirmata '/dev/tty.usbmodemfa1341'>
>>> ard.pins
{'0': <DigitalPin @0>, '1': <DigitalPin @1>, '2': <DigitalPin
@2>, '3': <DigitalPin @3>, '4': <DigitalPin @4>, '5': ...
<AnalogPin @A3>, 'A2': <AnalogPin @A 2>, 'A5': <AnalogPin</pre>
@A5>, 'A4': <AnalogPin @A4>}
>>> p13 = ard.pins[13]
>>> p13.mode = OUT
>>> p13.hi()
>>> p13.lo()
>>> from time import sleep
>>> p13.toggle()
>>> p13.toggle()
>>> p13.toggle()
>>> while True:
p13.toggle()
... sleep(.1)
```

#### BLINK: SCRIPT 1

while True:
 led.toggle()
 time.sleep(.1)



#### BLINK: SCRIPT 2

Detecta Arduino em

porta serial/USB

```
import time
import pingo

ard = pingo.arduino.get_arduino()
led = ard.pins[13]
led.mode = pingo.OUT

while True:
    led.toggle()
    time.sleep(.1)
```



#### BLINK: SCRIPT 3

```
import pingo
board = pingo.detect.MyBoard()
led = board.pins[13]
led.mode = pingo.OUT
```

import time

while True:

led.toggle()

time.sleep(.1)

Detecta qualquer

placa suportada



#### DOJO SCRIPT

```
board = pingo.detect.MyBoard()
print('board: %s' % board)
leds = board.digital_pins[6:13]
pot = board.pins['A0']
for led in leds:
    led.mode = pingo.OUT
while True:
    for led in leds:
        if led.location == 9:
            continue
        led.high()
        time.sleep(pot.ratio())
        led.low()
```

import time

import pingo

DigitalPins: 6 a 12

AnalogPin: 'AO'

Script compatível com qualquer placa com suporte a AnalogPin.

Para algumas placas, será preciso editar a localização dos pinos.



CONCEITOS

# PRINCÍPIOS

- API orientada a objeto
  - exploração interativa
  - fácil de estender para novos dispositivos
- Usabilidade
  - default: pinos identificados por localização física
  - atributos de conveniência: digital\_pins, toggle...
- Boas práticas
  - Python idiomático
  - testes automatizados (o maior desafio)



### **DRIVERS**

- Implementam métodos de controle específicos de cada placa
- Disponíveis em maio de 2014

driver	operação	funcionalidade
ArduinoFirmata	remota	digital + analógica
pcDuino	na placa	digital + analógica
Raspberry Pi	na placa	digital
UDOO	na placa	digital

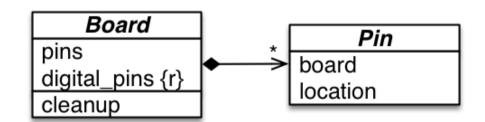


# OBJETOS BÁSICOS

- pingo.Board (placa)
  - pingo.arduino.ArduinoFirmata
  - pingo.rpi.RaspberryPi
  - pingo.pcduino.PcDuino

\_ ...

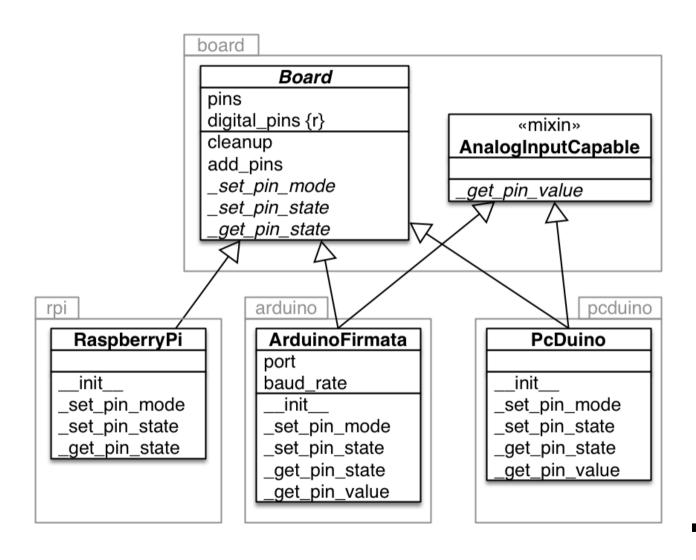
- pingo.Pin (pino)
  - pingo.DigitalPin
  - pingo.AnalogPin





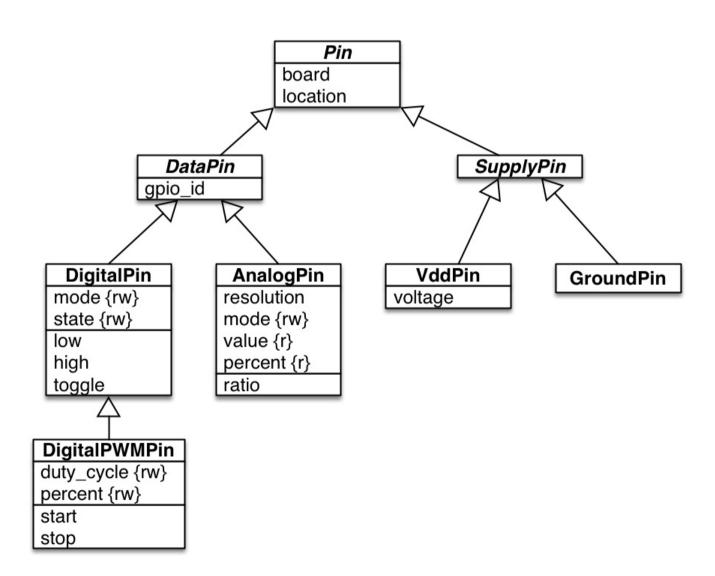
•

### **PLACAS**



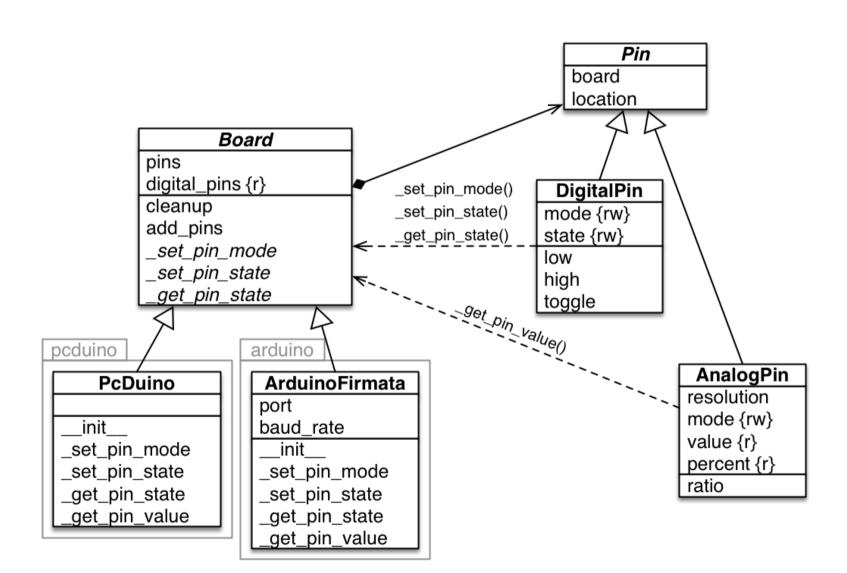
GAROA HACKER CLUBE

# Pinos



GAROA CLUBE

# COLABORAÇÕES

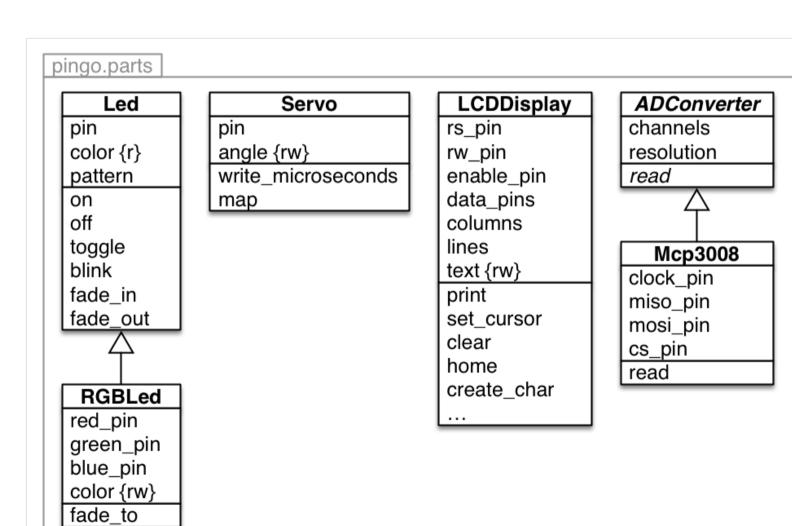


#### **PLANOS**

- Drivers para: Intel Galileo, Arduino Yún, BeagleBone Black, Arduino Tre
- Reestruturar sistema de testes
  - com mock e testes distribuídos em 'cluster' de placas físicas
- Implementar pinos especializados: PWM, DAC, multi-função digital/analógico...
- Implementar protocolos: SPI, I2C...
- Implementar componentes



# COMPONENTES: PACOTE PINGO PARTS



# CONTRIBUIÇÕES

 Lucas Vido, colaborador do Pingo, está contribuindo com o projeto PyFirmata, implementando o suporte a detecção automática de pinos do Firmata 2.2

#### Precisamos:

- usuários que experimentem, reportem falhas e sugiram melhorias
- especialistas em testes automatizados distribuídos em Python



# JUNTE-SE A NÓS!

- Projeto de software livre no estágio inicial
  - decisões de arquitetura interessantes
  - qualquer contribuição faz diferença
  - reuniões quinzenais: 2ª-feira, 19:30 no Garoa
- Site com documentação: http://pingo.io
- Repositório: http://github.com/garoa/pingo
- Google Groups: pingo-io http://groups.google.com/forum/#!forum/pingo-io



### GAROA: #COMOFAZ

- Todas as atividades são abertas a todos os interessados
  - **não** precisa se associar para participar
  - mas é preciso **participar** para poder se associar!
- Aberto quase todos os dias úteis a partir de 19:30 (às vezes no fim de semana)
- Veja a programação no site:



