



**API “UNIVERSAL”  
PARA HARDWARE HACKING**

# 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](http://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





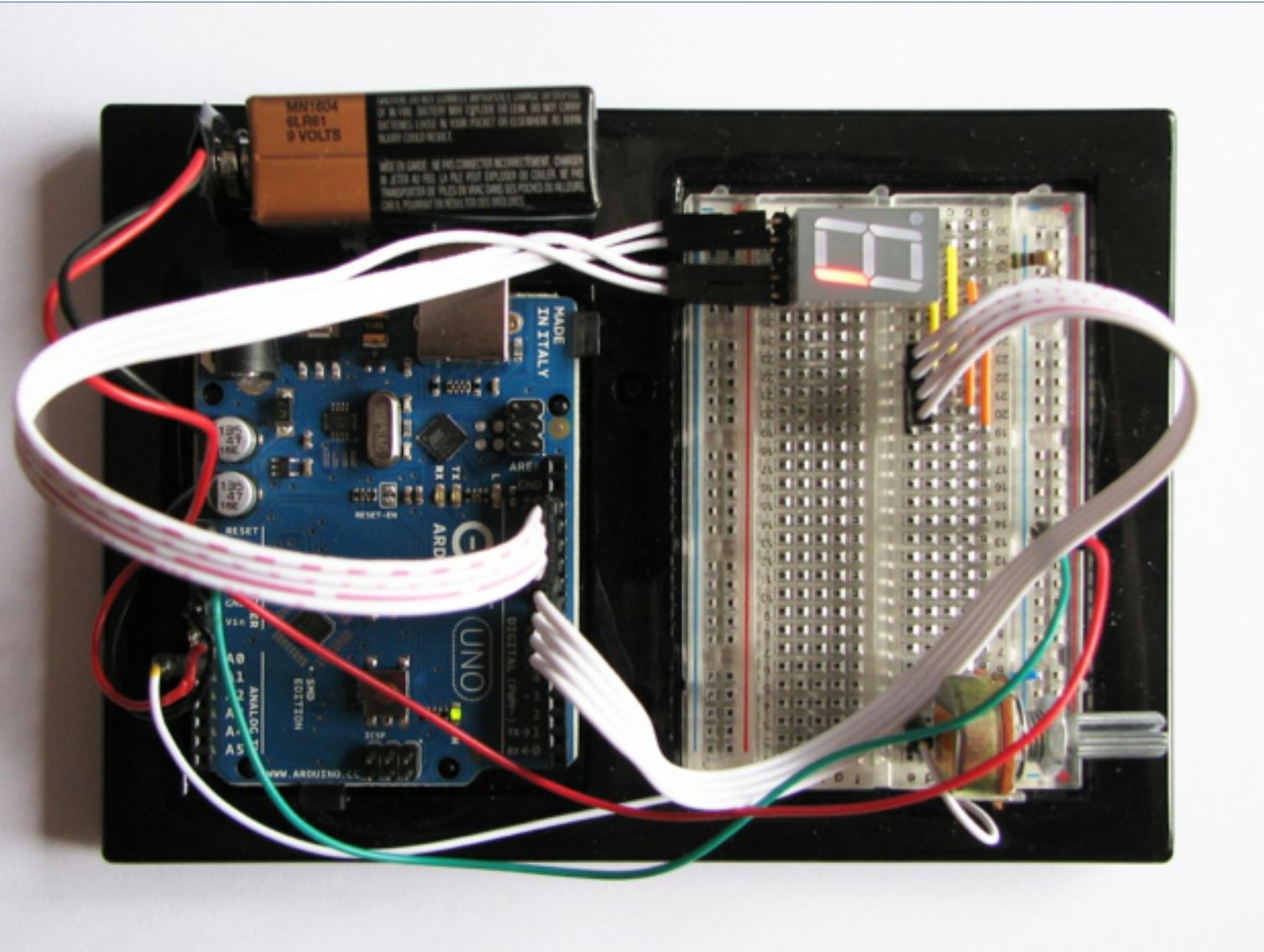
O “PROBLEMA”

# Pinos GPIO

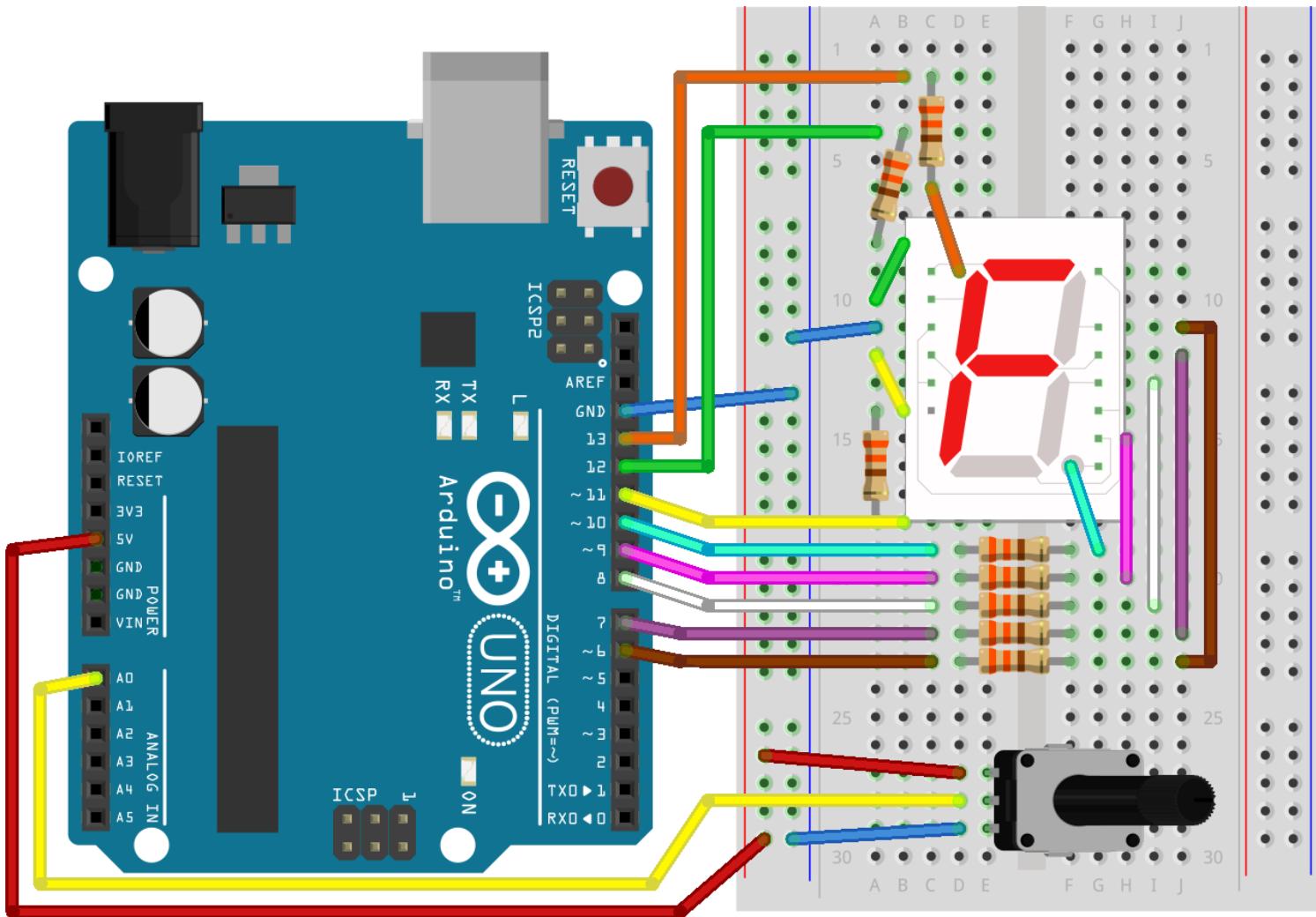
- Todas as placas interessantes para computação física possuem pinos **GPIO**
  - General Purpose I/O
- Pinos **digitais** I/O
  - 0/1 == HIGH/LOW
- Pinos **análogicos**
  - conversor A/D
- Pinos digitais com suporte a **PWM**
  - substituto saída analógica sem conversor D/A



# DOJO COM ARDUINO



# DOJO COM ARDUINO



fritzing

# PCDuino



	pcDuino
<b>SoC</b>	Allwinner A10
<b>CPU</b>	<b>ARM Cortex A8 (ARMv7 + NEON SIMD)</b>
<b>GPU</b>	Mali 400
<b>clock</b>	<b>1 GHz</b>
<b>OS</b>	GNU/Linux (vários) Android
<b>RAM</b>	<b>1 GB</b>
<b>Flash onboard</b>	<b>2 GB</b>
<b>SD card</b>	<b>micro-SD</b>
<b>GPIO</b>	14
<b>PWM</b>	6
<b>ADC</b>	6
<b>USB client</b>	1
<b>USB host</b>	1
<b>Ethernet</b>	10/100
<b>Wifi</b>	X
<b>HDMI</b>	<b>HDMI</b>
<b>video composto</b>	X
<b>audio</b>	via HDMI
<b>preço USD</b>	59
<b>Arduinos</b>	1.7
<b>open hardware</b>	?

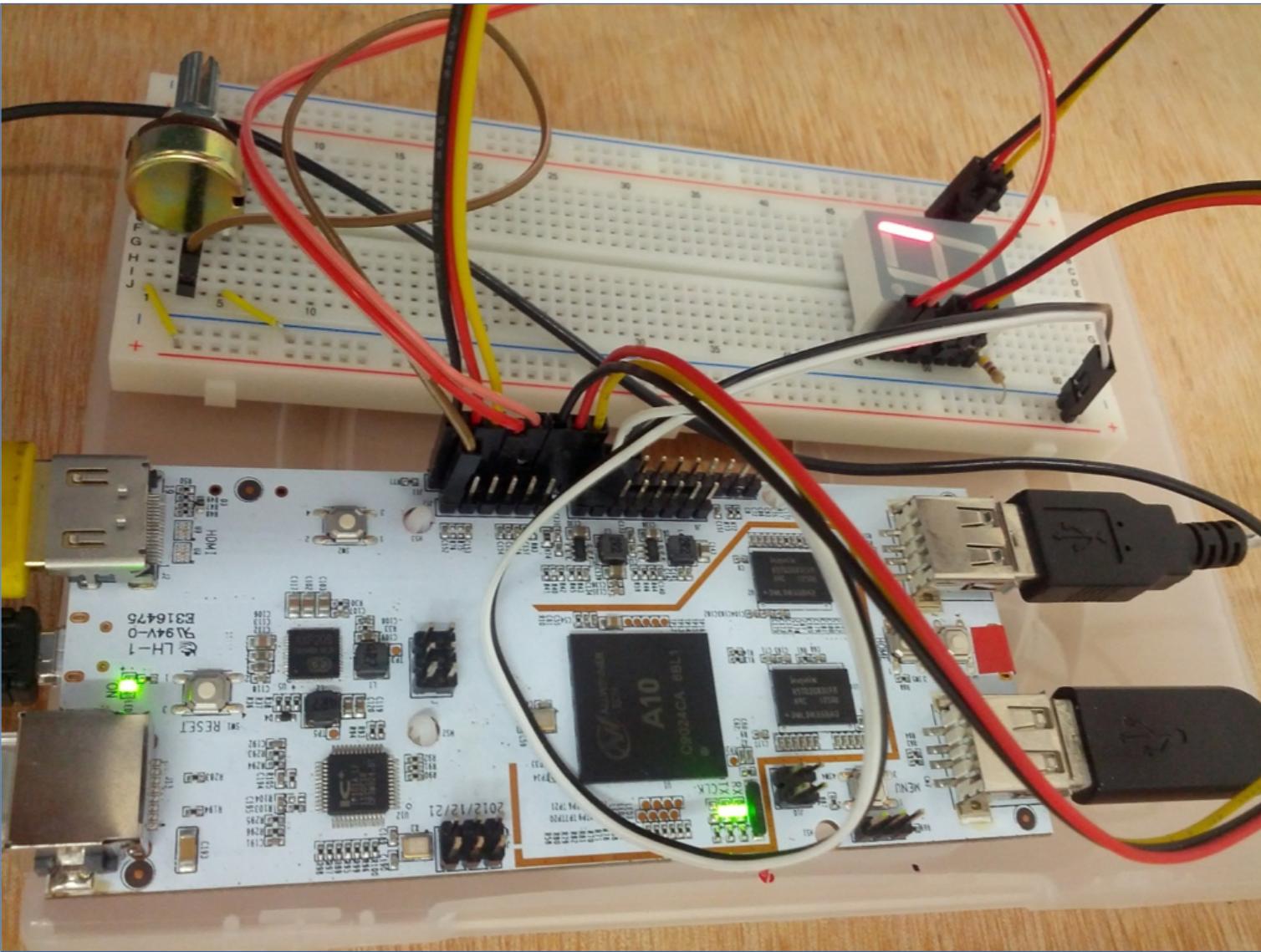
# PCDUINO

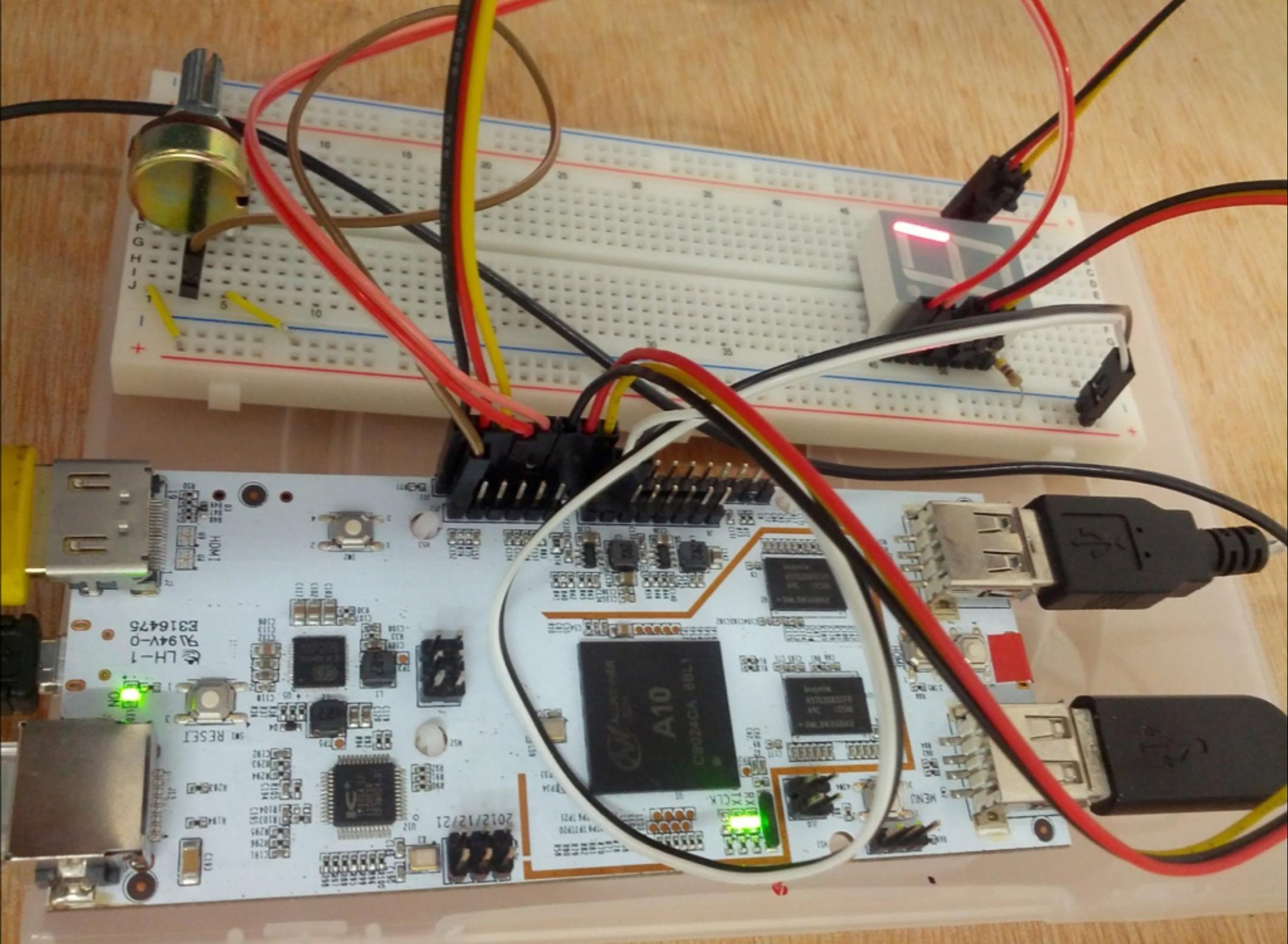


pinos físicos	
J12	A5
	A4
	A3
	A2
ADC 12 bits	A1
	A0
ADC 6 bits	
J11	1
	2
	3
	4
	5
	6
	7
	8
5V in	0
gnd	1
gnd	2
5 V	3
3.3 V	4
reset	5
5 V	6
	7
J9	1
	2
	3
	4
	5
	6
	7
	8
	9
	10 ~
	11 ~
	12
	13
	gnd
	ARef
	SDA
	SCL
J8	8
	9 ~
	10 ~
	11 ~
	12
	13
	gnd
	ARef
	SDA
	SCL
analog in	UART
	SPI
	I <sup>2</sup> C
~ pino PWM: total 6	

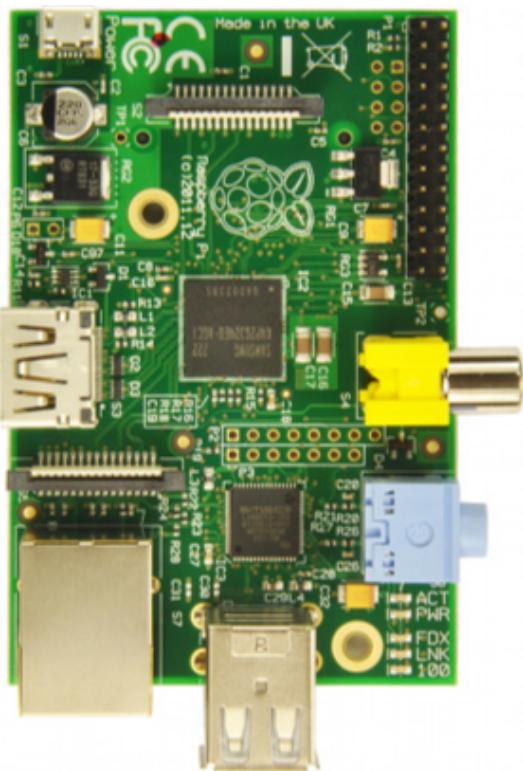


# DOJO com PCDUino





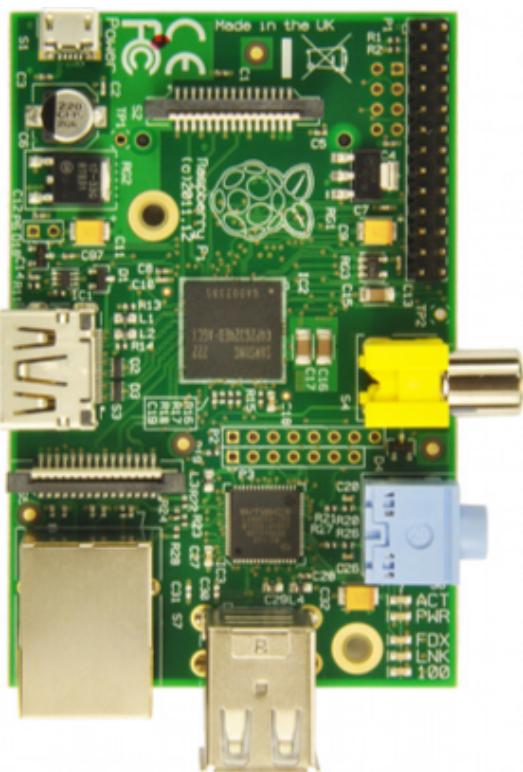
# RASPBERRY Pi



	Raspberry Pi mod. B
<b>SoC</b>	Broadcom BCM2835
<b>CPU</b>	ARM11 (ARMv6)
<b>GPU</b>	VideoCore IV
<b>clock</b>	700 MHz
<b>SO</b>	GNU/Linux (vários)
<b>RAM</b>	512 MB
<b>Flash onboard</b>	X
<b>SD card</b>	SD
<b>GPIO</b>	17 (+4 no P5)
<b>PWM</b>	1
<b>ADC</b>	X
<b>USB client</b>	X
<b>USB host</b>	2
<b>Ethernet</b>	10/100
<b>Wifi</b>	X
<b>HDMI</b>	HDMI
<b>video composto</b>	RCA
<b>audio</b>	HDMI + plug 3.5mm
<b>preço USD</b>	35
<b>Arduinos</b>	1.0
<b>open hardware</b>	X

# RASPBERRY Pi

P1



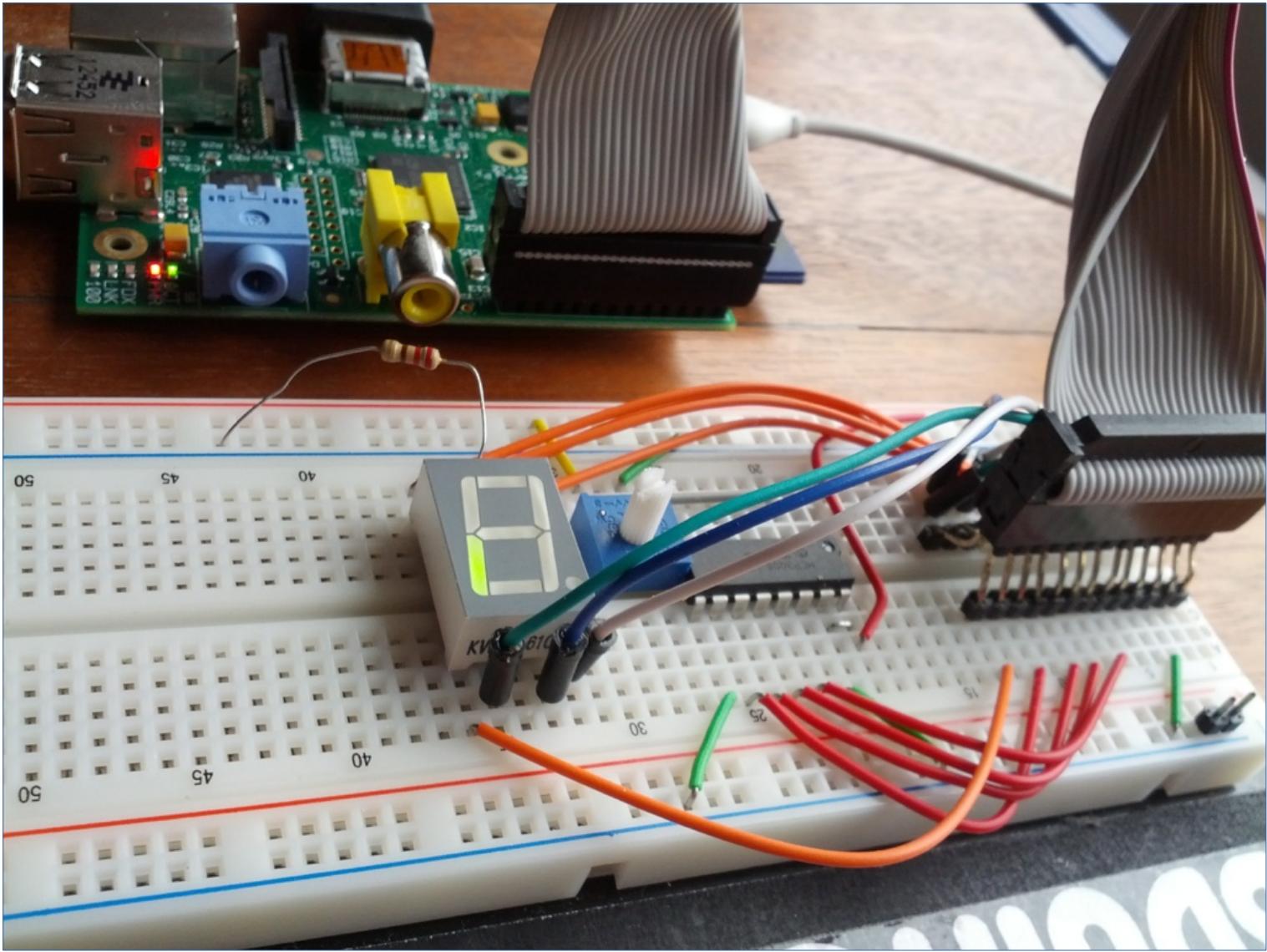
P1		
pinos físicos		
SDA	3.3 V * 0→2	5 V
SCL	3.3 V * 1→3	5 V
GPCLK0	4	gnd
	17	TXD
	* 21→27	14
	22	15
MOSI	3.3 V 10	PCM_CLK
MISO	9	18 ~
SCLK	11	gnd
	23	23
	24	24
	19	gnd
	20	25
	21	8
	22	CE0
	23	7
	24	CE1
	25	
	26	

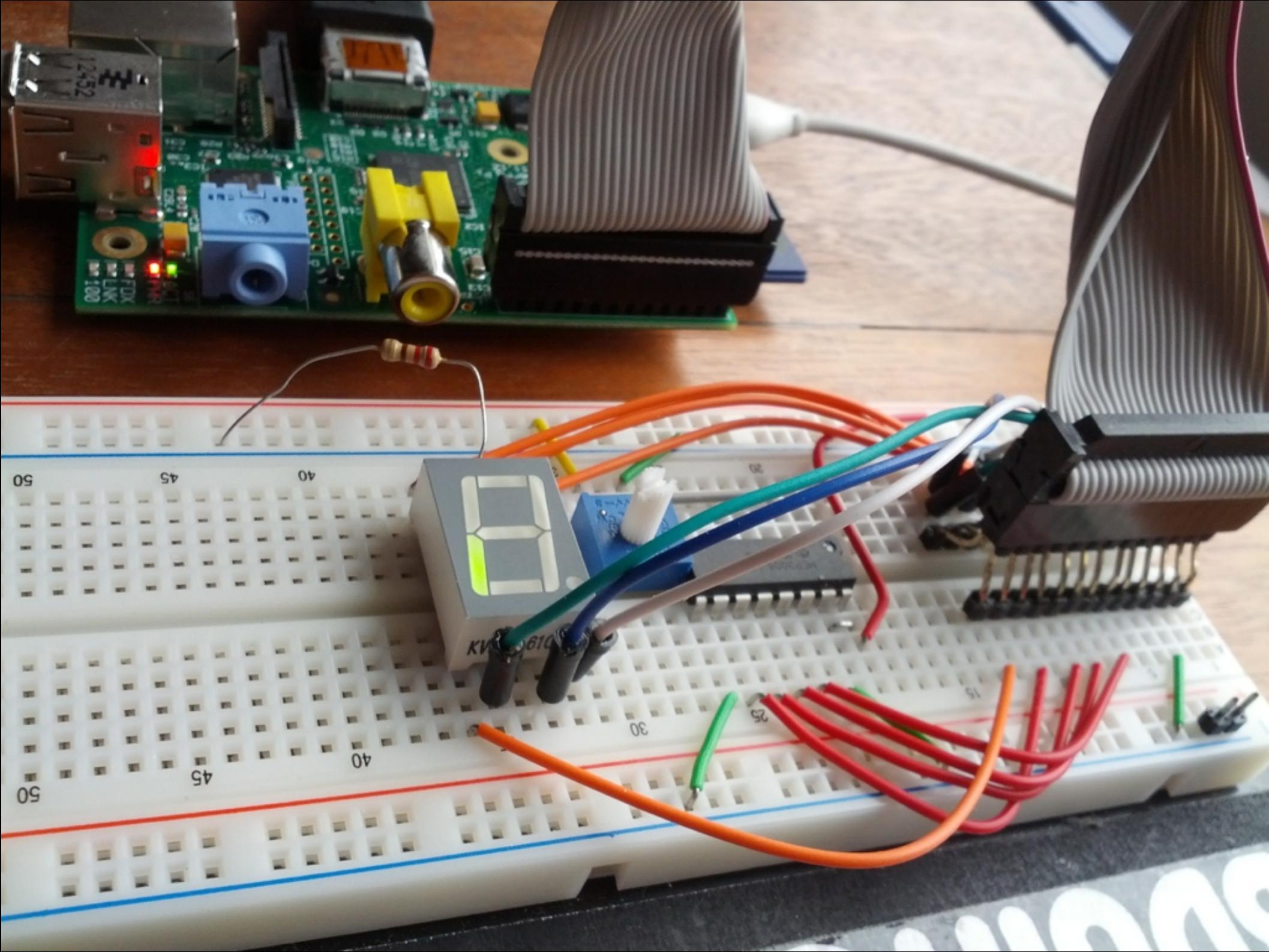
~ pino PWM: 12 (GPIO\_18)

\* pinos GPIO renumerados  
rev.1→rev.2

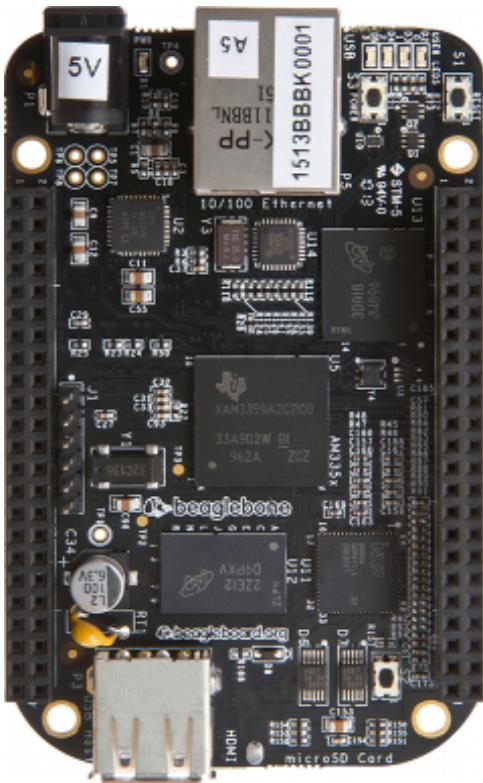


# DOJO COM RASPBERRY Pi





# BEAGLEBONE BLACK



	<b>Beaglebone Black</b>
<b>SoC</b>	Texas Instruments Sitara AM335x
<b>CPU</b>	<b>ARM Cortex A8</b> <b>(ARMv7 + NEON SIMD)</b>
<b>GPU</b>	PowerVR SGX 530
<b>clock</b>	<b>1 GHz</b>
<b>OS</b>	GNU/Linux (vários) Android
<b>RAM</b>	512 MB
<b>Flash onboard</b>	<b>2 GB</b>
<b>SD card</b>	<b>micro-SD</b>
<b>GPIO</b>	<b>66</b>
<b>PWM</b>	<b>8</b>
<b>ADC</b>	<b>7</b>
<b>USB client</b>	<b>1</b>
<b>USB host</b>	<b>1</b>
<b>Ethernet</b>	10/100
<b>Wifi</b>	<b>X</b>
<b>HDMI</b>	micro-HDMI
<b>video composto</b>	<b>X</b>
<b>audio</b>	via HDMI
<b>preço USD</b>	<b>45</b>
<b>Arduinos</b>	<b>1.3</b>
<b>open hardware</b>	<b>✓</b>

# BEAGLEBONE BLACK

P9		
pinos físicos		
gnd	• 1	2 •
3.3 V	• 3	4 •
VDD 5 V	• 5	6 •
SYS 5 V	• 7	8 •
PWR_BUT	• 9	10 •
UART4_RXD	• 11	12 •
UART4_TXD	• 13	14 •
1-16	• 15	16 •
I2C1_SCL	• 17	18 •
I2C2_SCL	• 19	20 •
UART2_TXD	• 21	22 •
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 •
AIN2	• 37	38 •
AIN0	• 39	40 •
2 CLKOUT2, 3-20	• 41	42 •
gnd	• 43	44 •
gnd	• 45	46 •

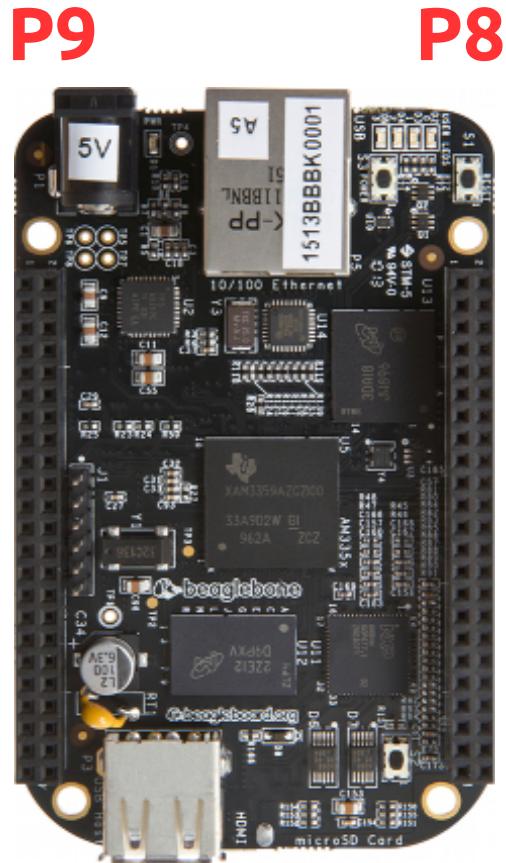
x-yy pino GPIOX\_YY

~ pino PWM: 14, 16, 28, 42<sup>2</sup> (0-7)

\* em uso: oscilador audio HDMI

<sup>2</sup> pino físico ligado a dois pinos lógicos

analog input
I <sup>C</sup>
UART
SPI
misc.



P8		
pinos físicos		
gnd	• 1	2 •
	• 6	7 •
	• 2	3 •
TIMER4	• 5	6 •
TIMER5	• 7	8 •
	• 13	12 •
~ EHRPWM2B	• 11	14 •
	• 15	16 •
0-27	• 17	18 •
~ EHRPWM2A	• 19	20 •
	• 30	22 •
1-4	• 21	24 •
	• 23	26 •
1-0	• 25	28 •
	• 27	30 •
2-22	• 29	32 •
	• 31	34 •
2-23	• 33	36 •
UART5_CTSN	• 35	38 •
UART4_RTSN	• 37	40 •
UART4_CTSN	• 39	42 •
UART5_TXD	• 41	44 •
	• 45	46 •

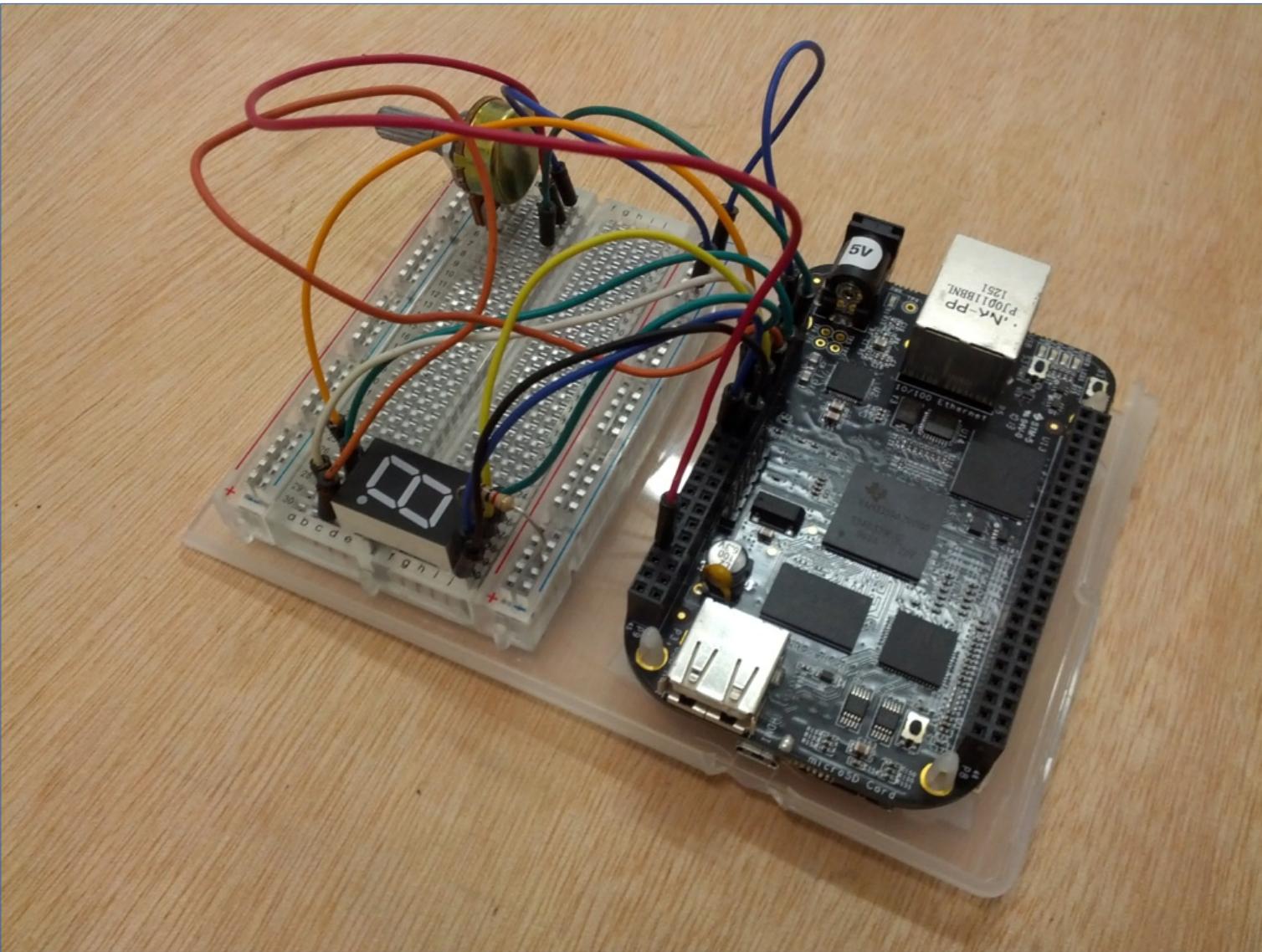
x-yy pino GPIOX\_YY

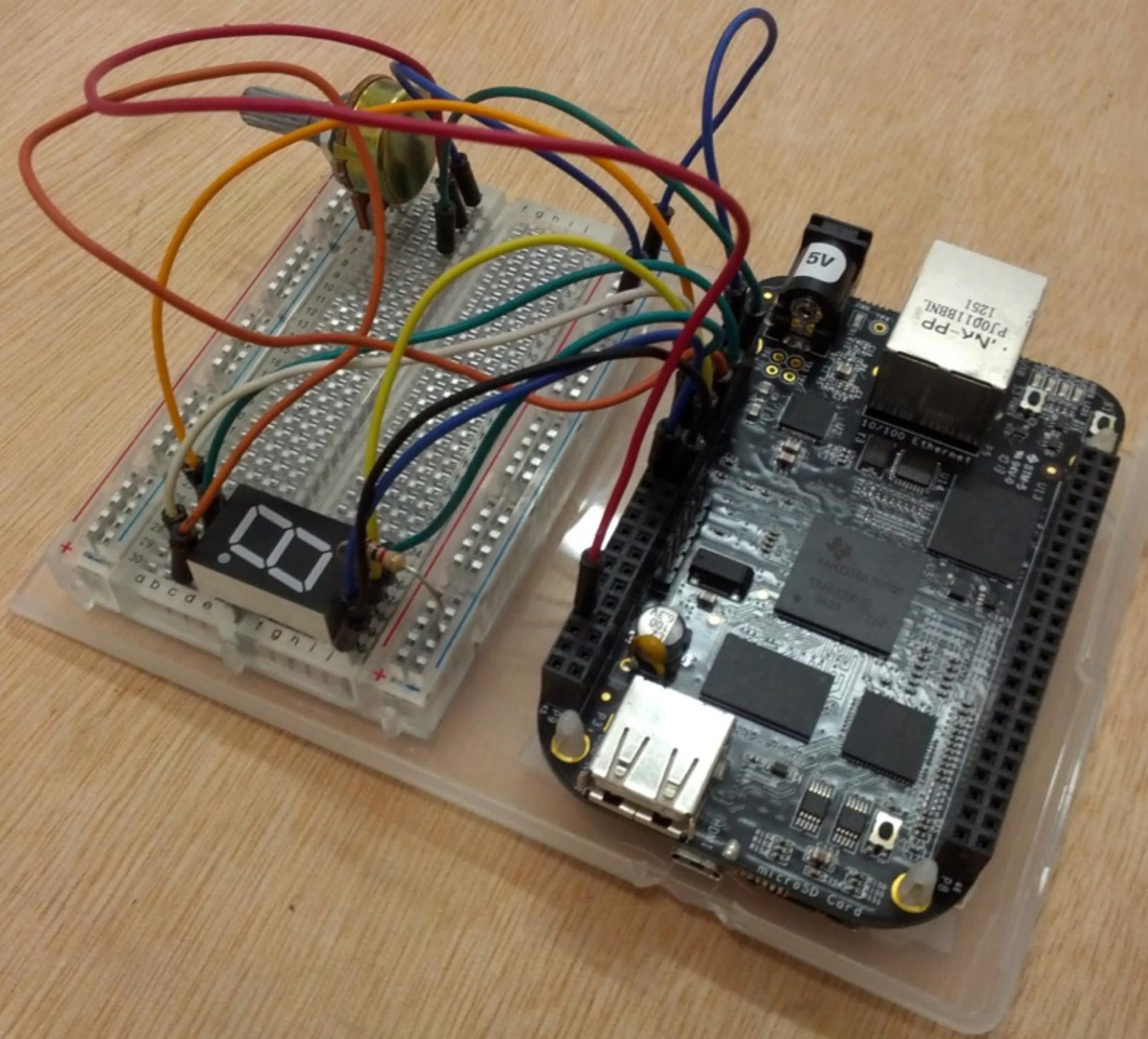
~ pino PWM: 13, 19

UART

timers

# 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 **pyFirmata**



```

import time, os

GPIO_PATH = os.path.normpath('/sys/devices/virtual/misc/gpio/')
ADC_PATH = os.path.normpath('/proc/')

INPUT = LOW = "0"
OUTPUT = HIGH = "1"

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)
        digital_write(i, LOW)

setup()
while True:
    for i in [0, 1, 7, 5, 4, 2]:
        digital_write(i, 1)
        delay = analog_read(5)/4096.0
        time.sleep(delay)
        digital_write(i, 0)

```

SysFS:  
pinos GPIO  
mapeados  
em arquivos

Nenhuma biblioteca  
especial é usada neste  
script!



# PCDUINO

```
import atexit
import time
import RPi.GPIO as GPIO
import spi

# 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)

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)
```

Duas bibliotecas  
específicas: RPi.GPIO e  
spi (feita em casa)

# RASPBERRY Pi



Biblioteca específica:  
**Adafruit\_BBIO**

```
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: <DigitalPin @5>, 6: <DigitalPin @6>, 7: <DigitalPin @7>, 8: <DigitalPin @8>, 9: <DigitalPin @9>, 10: <DigitalPin @10>, 11: <DigitalPin @11>, 12: <DigitalPin @12>, 13: <DigitalPin @13>, 'A1': <AnalogPin @A1>, 'A0': <AnalogPin @A0>, 'A3': <AnalogPin @A3>, 'A2': <AnalogPin @A2>, 'A5': <AnalogPin @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

```
import time  
import pingo
```

```
ard = pingo.arduino.ArduinoFirmata('/dev/tty.usbmodemfa1341')  
led = ard.pins[13]  
led.mode = pingo.OUT  
  
while True:  
    led.toggle()  
    time.sleep(.1)
```

Criar instância de  
ArduinoFirmata



# BLINK: SCRIPT 2

```
import time
import pingo

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

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

Detecta Arduino em  
porta serial/USB



# BLINK: SCRIPT 3

```
import time
import pingo

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

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

Detecta qualquer  
placa suportada



# DOJO SCRIPT

```
import time
import pingo

board = pingo.detect.MyBoard()
print('board: %s' % board)
pot = board.pins['A0']
leds = board.digital_pins[6:13]

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()
```

AnalogPin: 'A0'

DigitalPins: 6 a 12

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

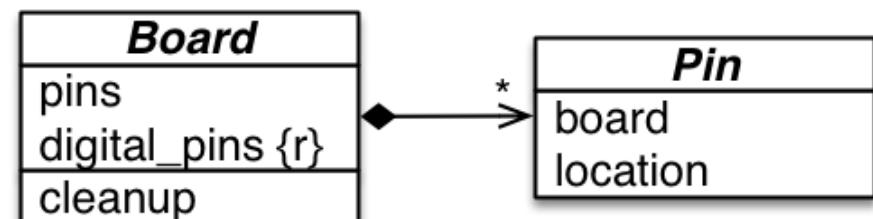
- Implementam métodos de controle específicos de cada placa
- Disponíveis em maio de 2014

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

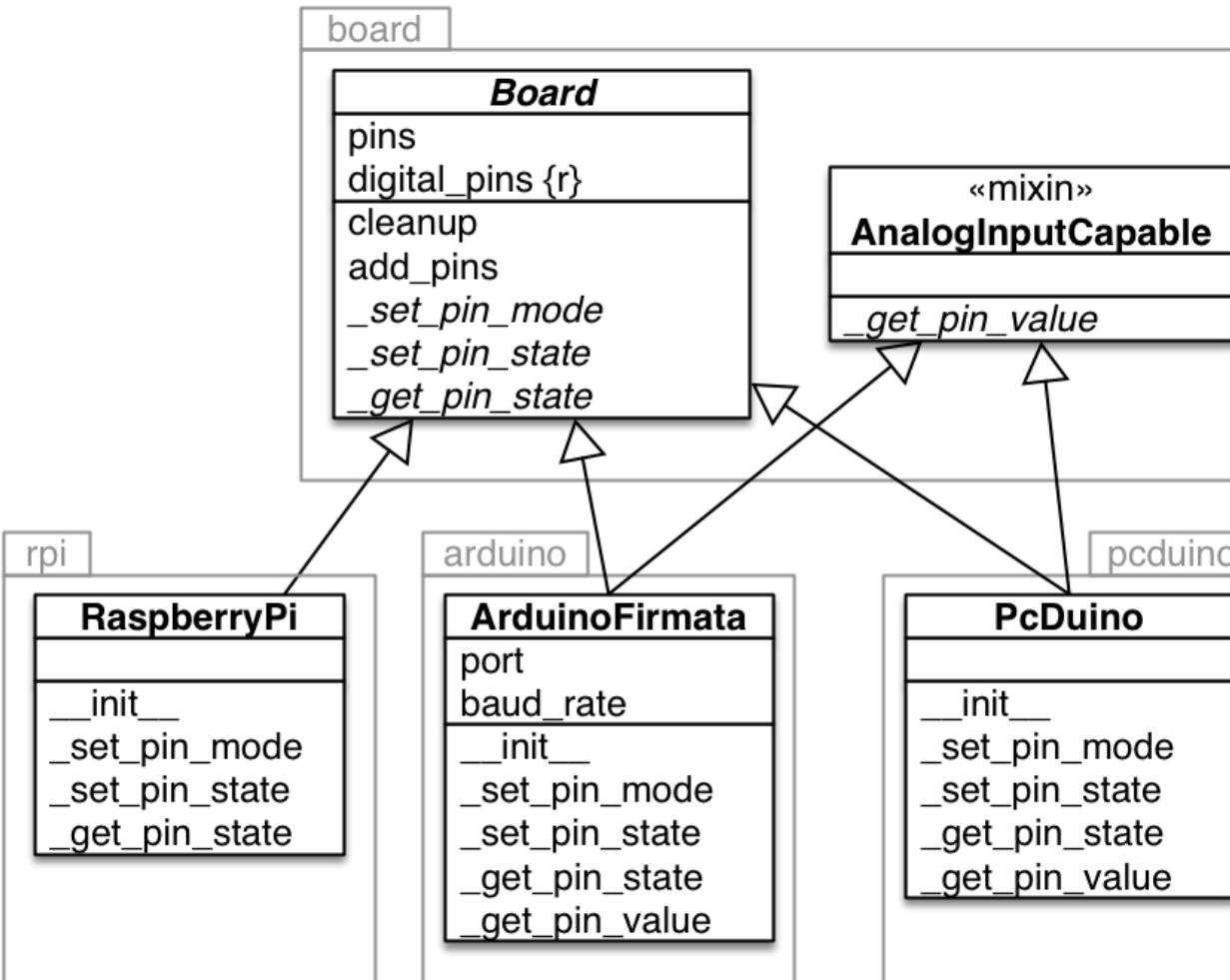


# OBJETOS BÁSICOS

- pingo.Board (placa)
  - pingo.arduino.ArduinoFirmata
  - pingo.rpi.RaspberryPi
  - pingo.pcduino.PcDuino
  - ...
- pingo.Pin (pino)
  - pingo.DigitalPin
  - pingo.AnalogPin
    - ...



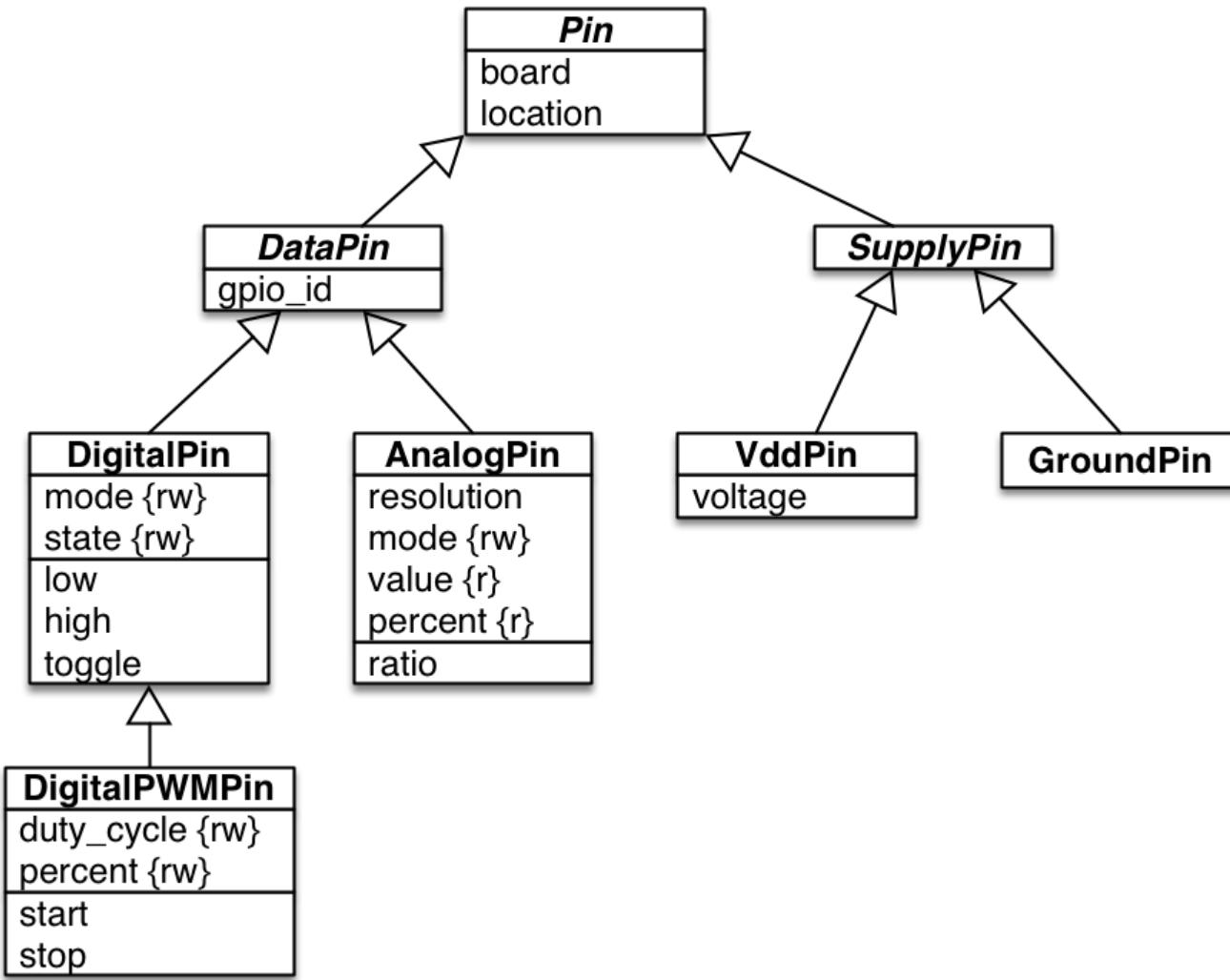
# PLACAS



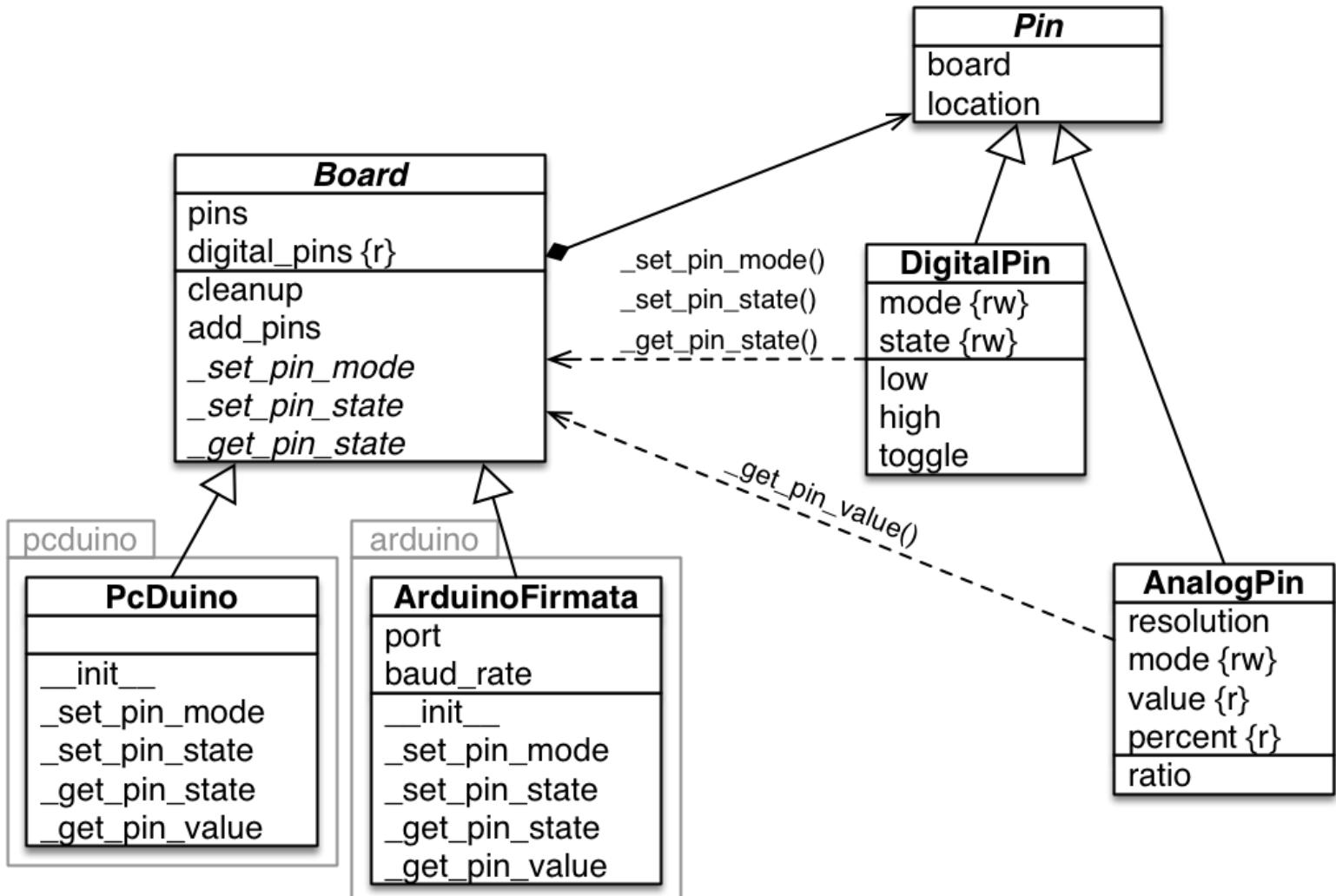
• • •



# PINOS



# COLABORAÇÕES

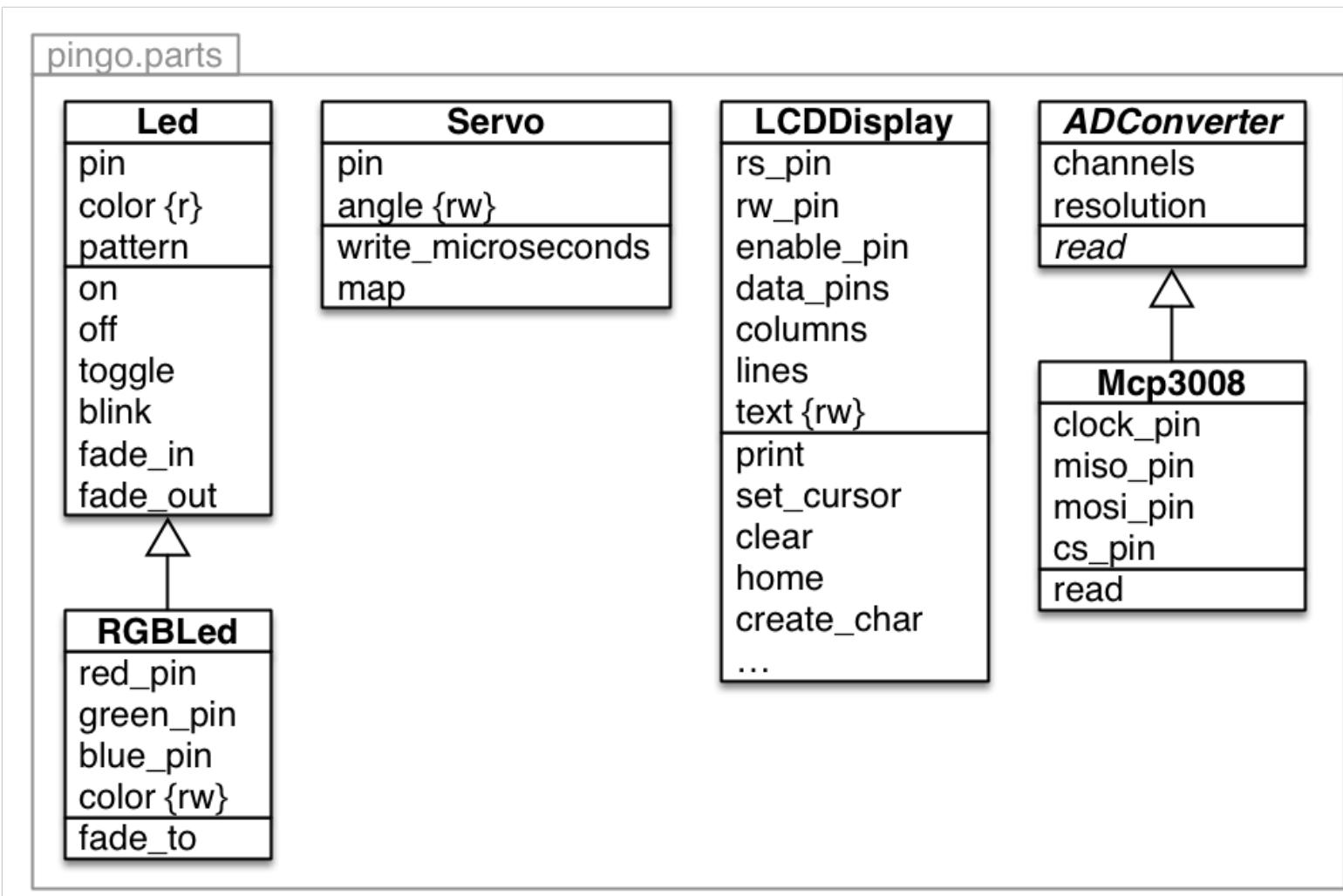


# PLANOS

- 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: 1ª e 3ª 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 todos os dias úteis a partir de 19:
  - às vezes também no fim de semana
- Veja a programação no site:

<http://garoa.net.br>

