



**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)
- 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





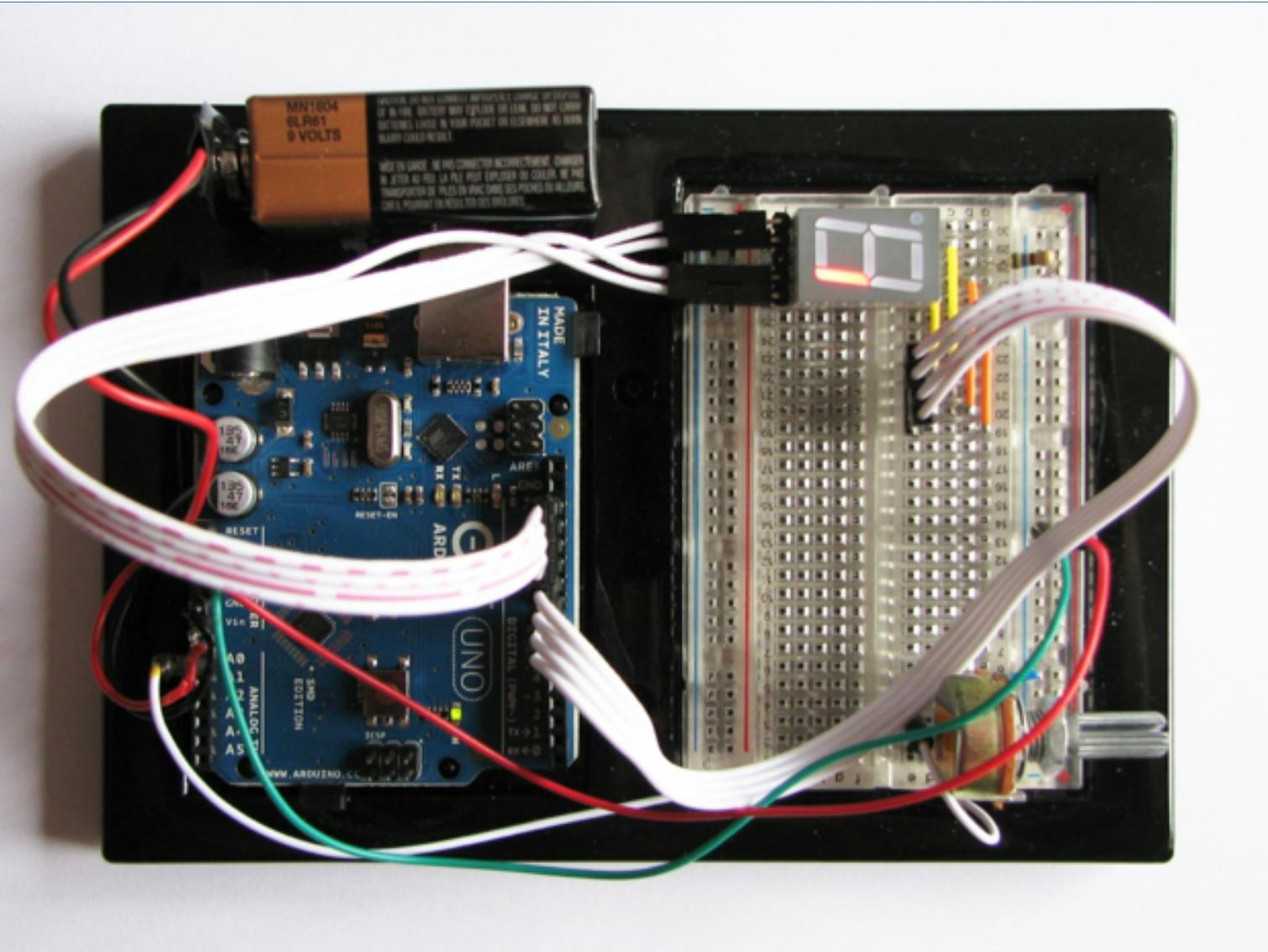
O “PROBLEMA”

Pinos GPIO

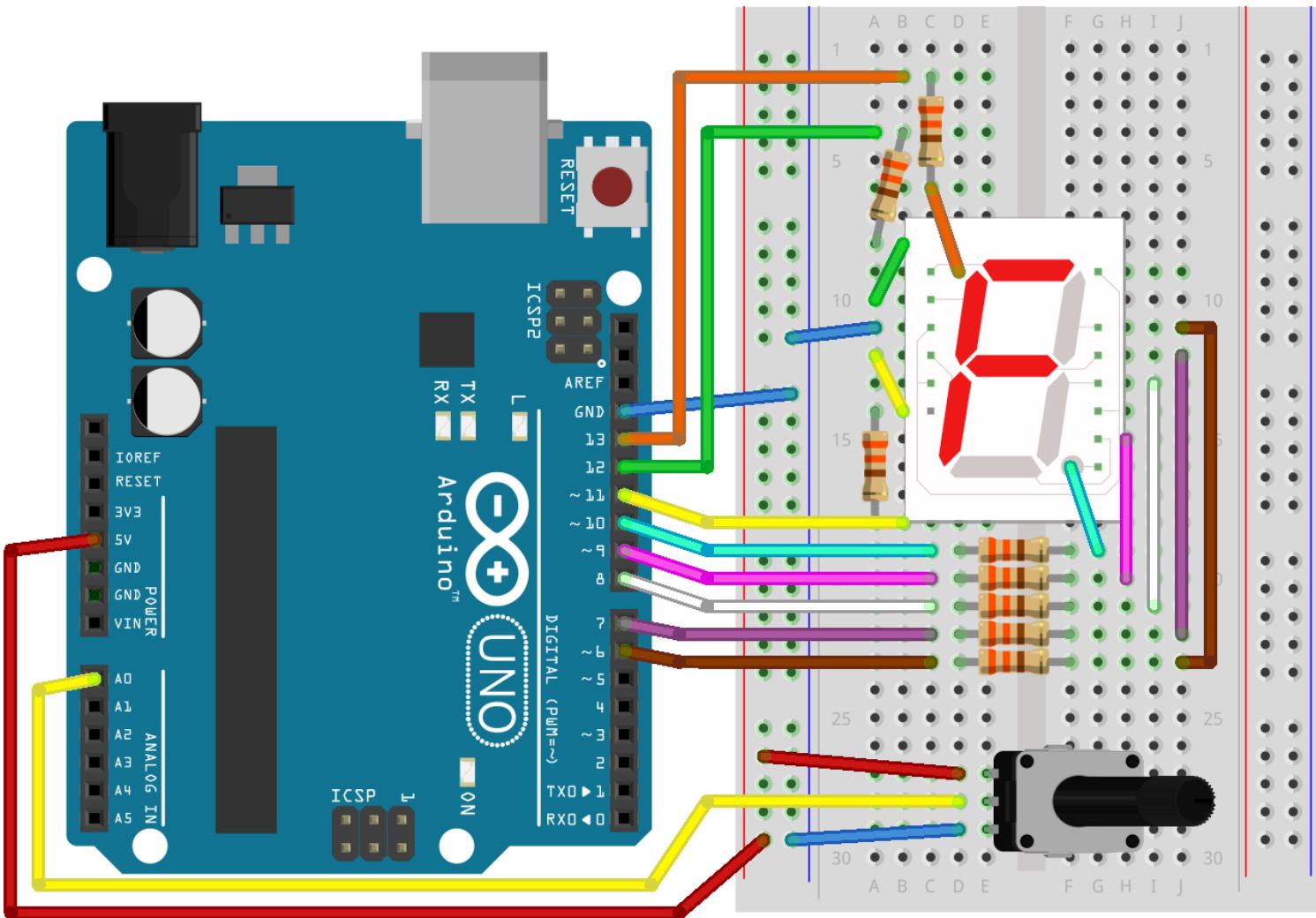
- Todas as placas interessantes para computação física possuem pinos **GPIO**
 - General Purpose I/O
- Pinos **digitais** I/O
 - 0/1 == HIGH/LOW
- Pinos **análogicos**
 - conversor A/D
- Pinos digitais com suporte a **PWM**
 - substituto saída analógica sem conversor D/A



DOJO COM ARDUINO



DOJO COM ARDUINO



fritzing

PCDuino



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

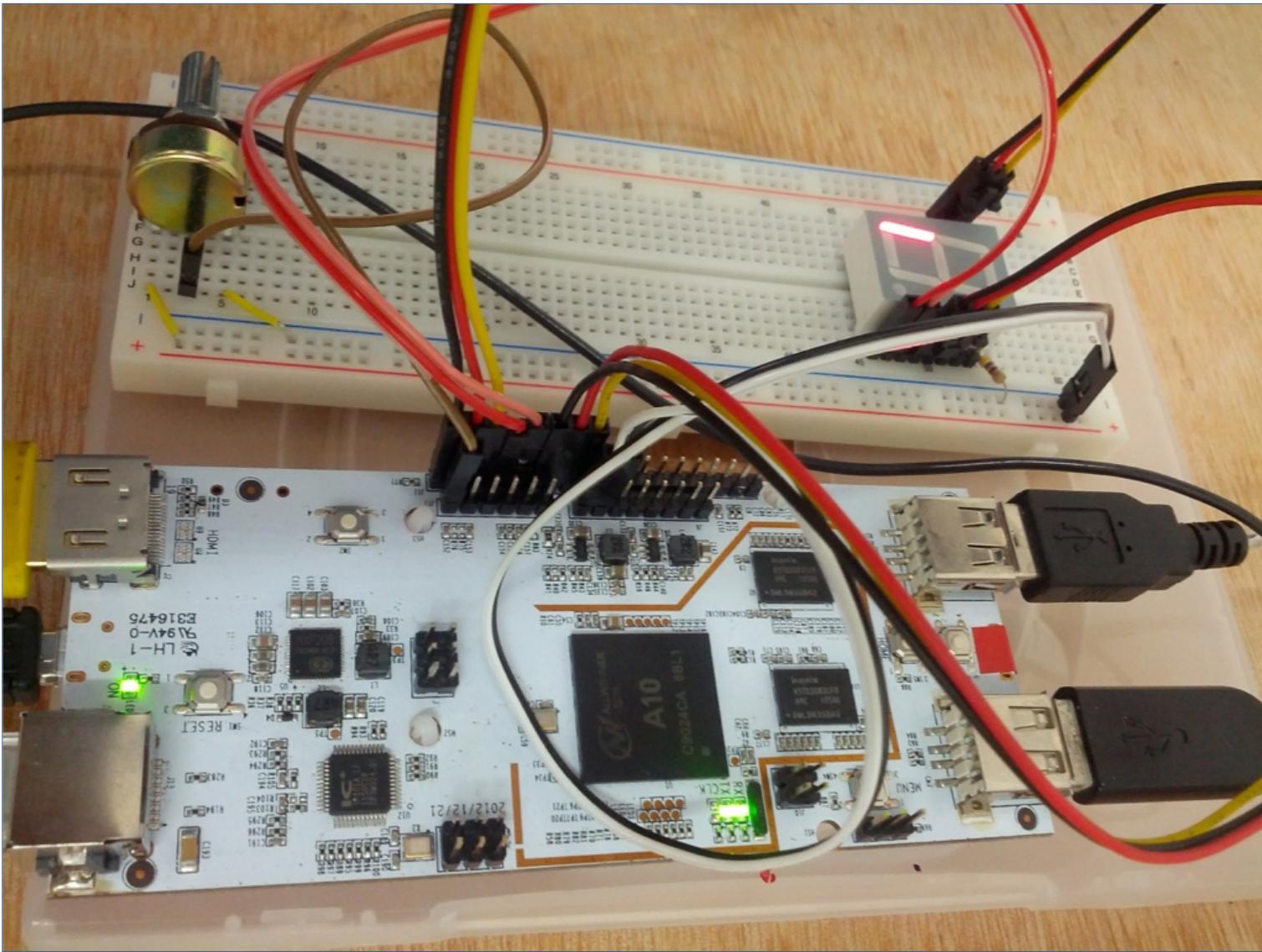
PCDUINO

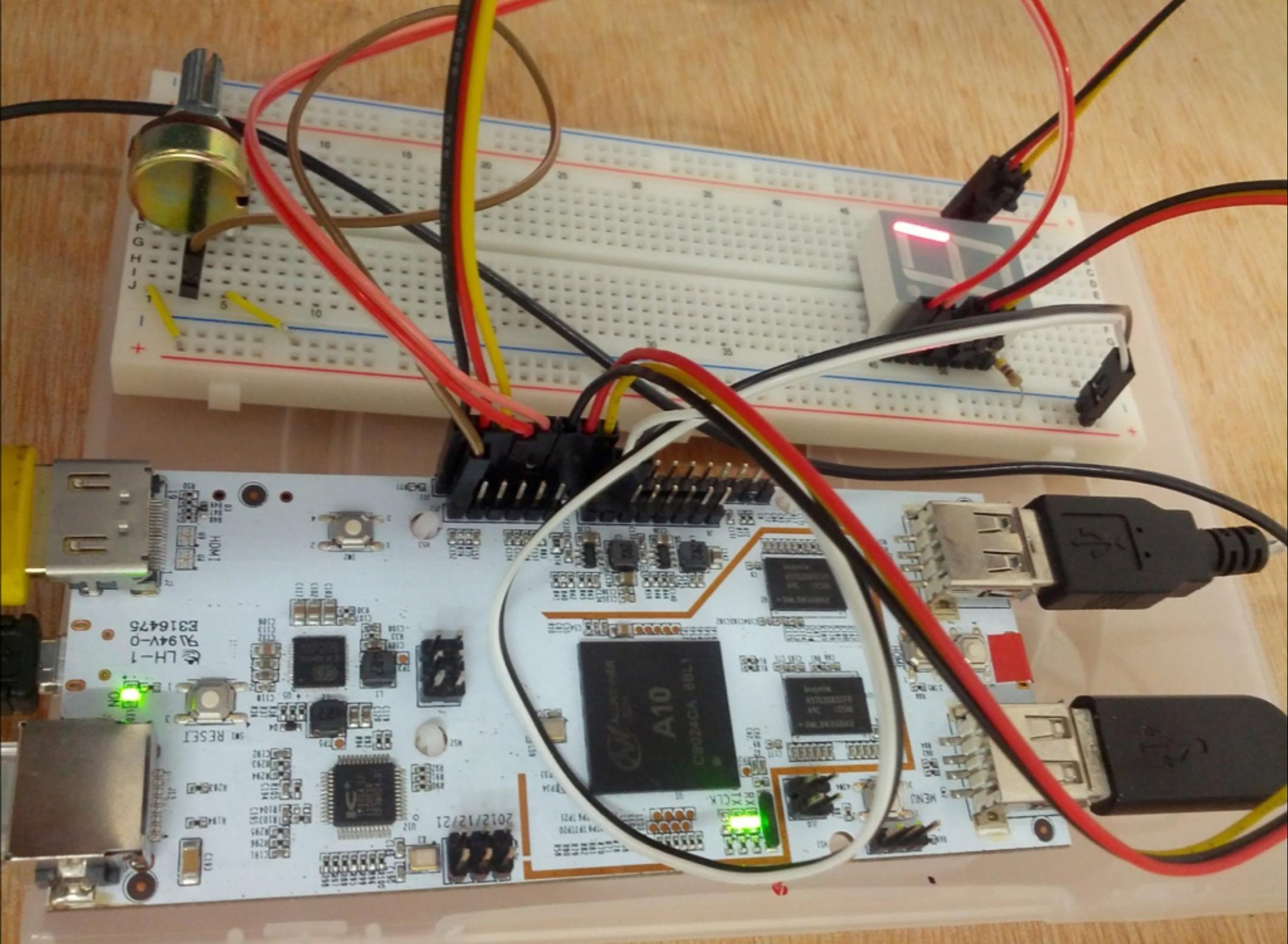


pinos físicos	
J12	A5
	A4
	A3
	A2
ADC 12 bits	A1
	A0
ADC 6 bits	
J11	1
	2
	3
	4
	5
	6
	7
	8
5V in	0
gnd	1
gnd	2
5 V	3
3.3 V	4
reset	5
5 V	6
	7
J9	1
	2
	3
	4
	5
	6
	7
	8
	9
	10 ~
	11 ~
	12
	13
	gnd
	ARef
	SDA
	SCL
J8	8
	9 ~
	10 ~
	11 ~
	12
	13
	gnd
	ARef
	SDA
	SCL
analog in	UART
	SPI
	I ² C
~ pino PWM: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

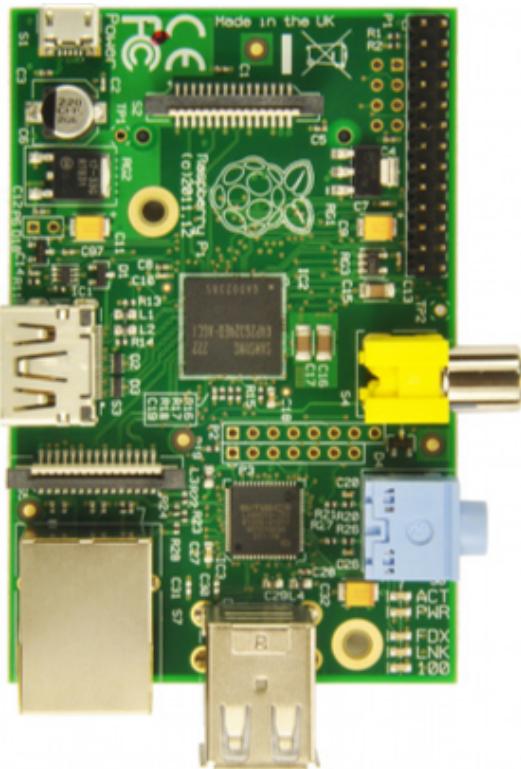


DOJO com PCDUino





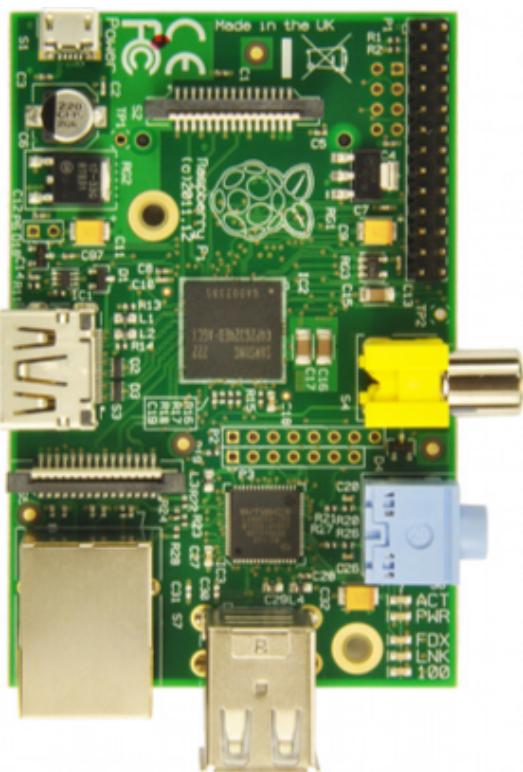
RASPBERRY Pi



	Raspberry Pi mod. B
SoC	Broadcom BCM2835
CPU	ARM11 (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	1
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

P1



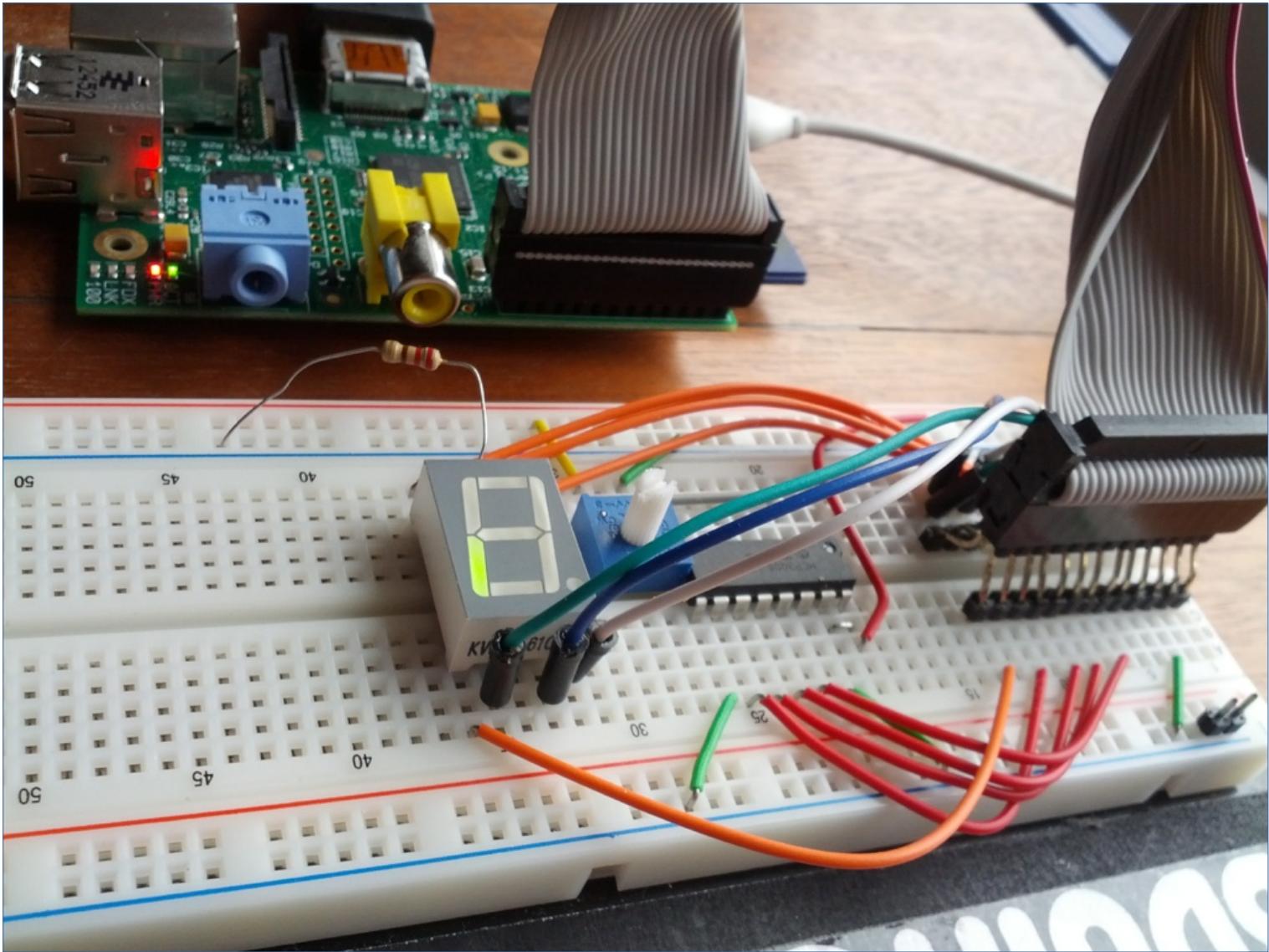
P1		
pinos físicos		
SDA	3.3 V * 0→2	5 V
SCL	3.3 V * 1→3	5 V
GPCLK0	4	gnd
	17	TXD
	* 21→27	14
	22	15
MOSI	3.3 V 10	PCM_CLK
MISO	9	18 ~
SCLK	11	gnd
	23	23
	24	24
	19	gnd
	20	25
	21	8
	22	CE0
	23	7
	24	CE1
	25	
	26	

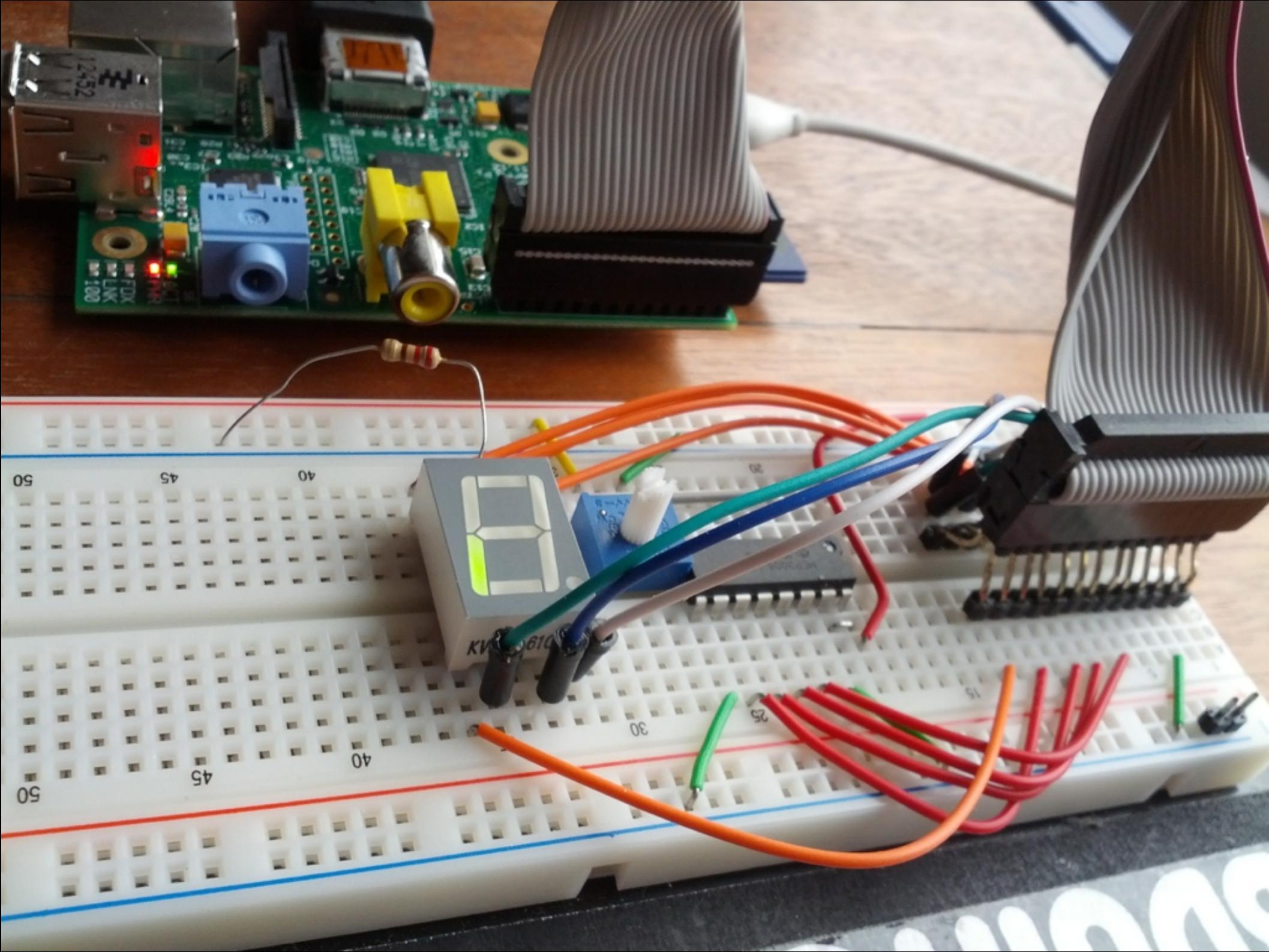
~ pino PWM: 12 (GPIO_18)

* pinos GPIO renumerados
rev.1→rev.2

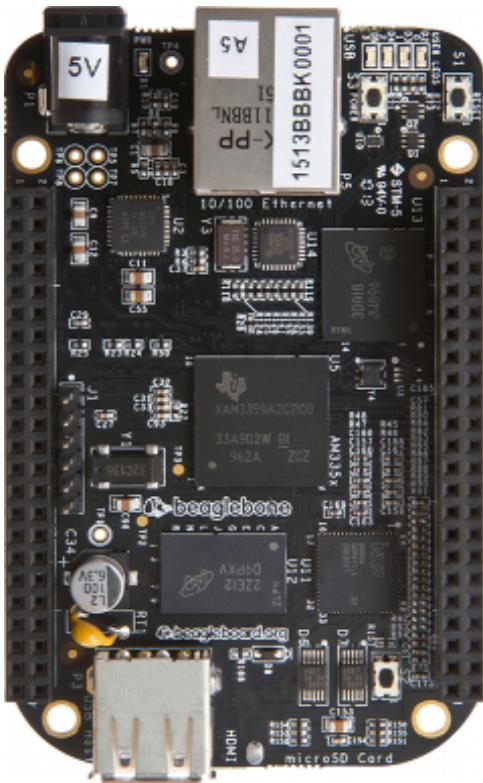


DOJO COM RASPBERRY Pi



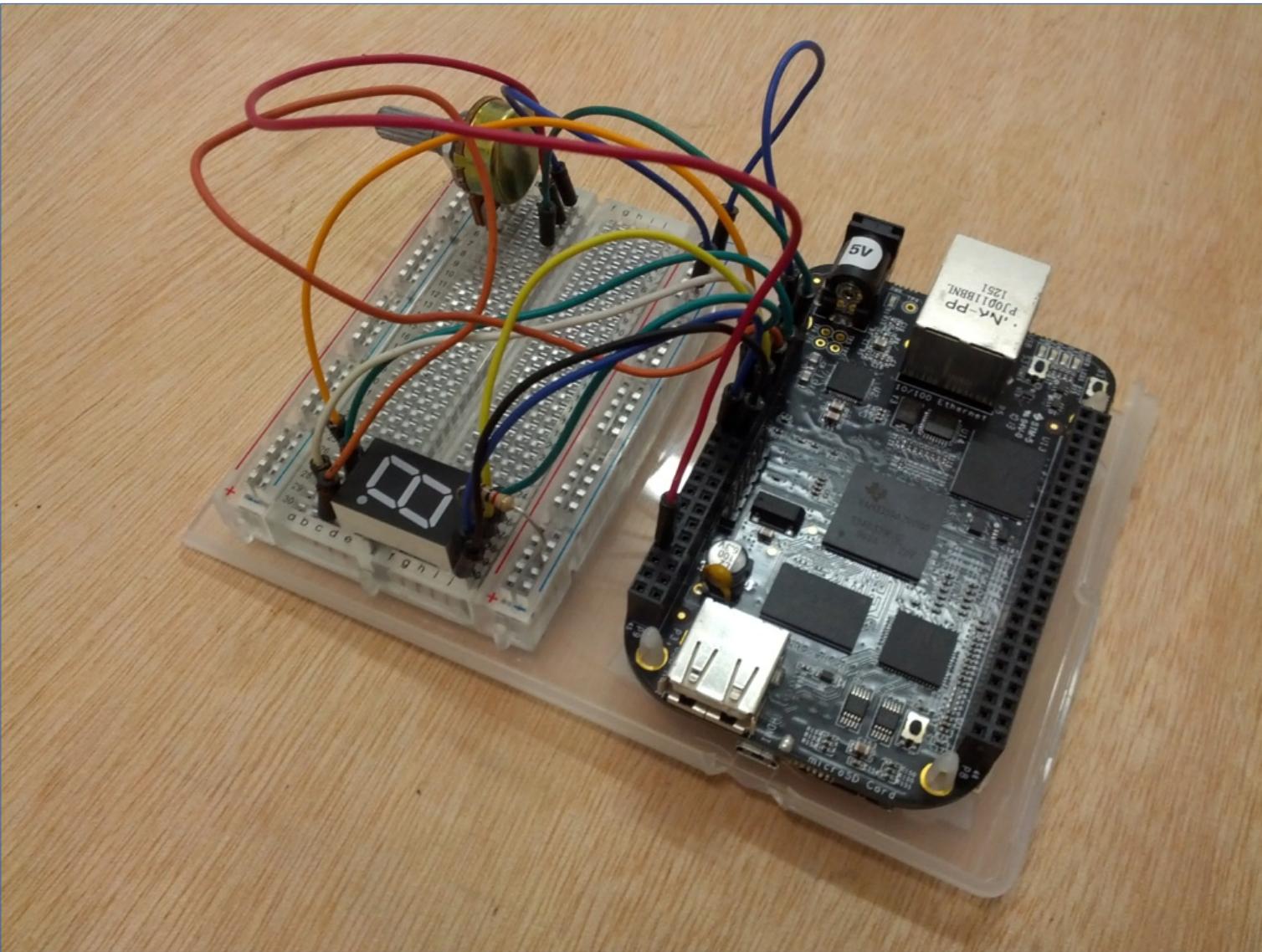


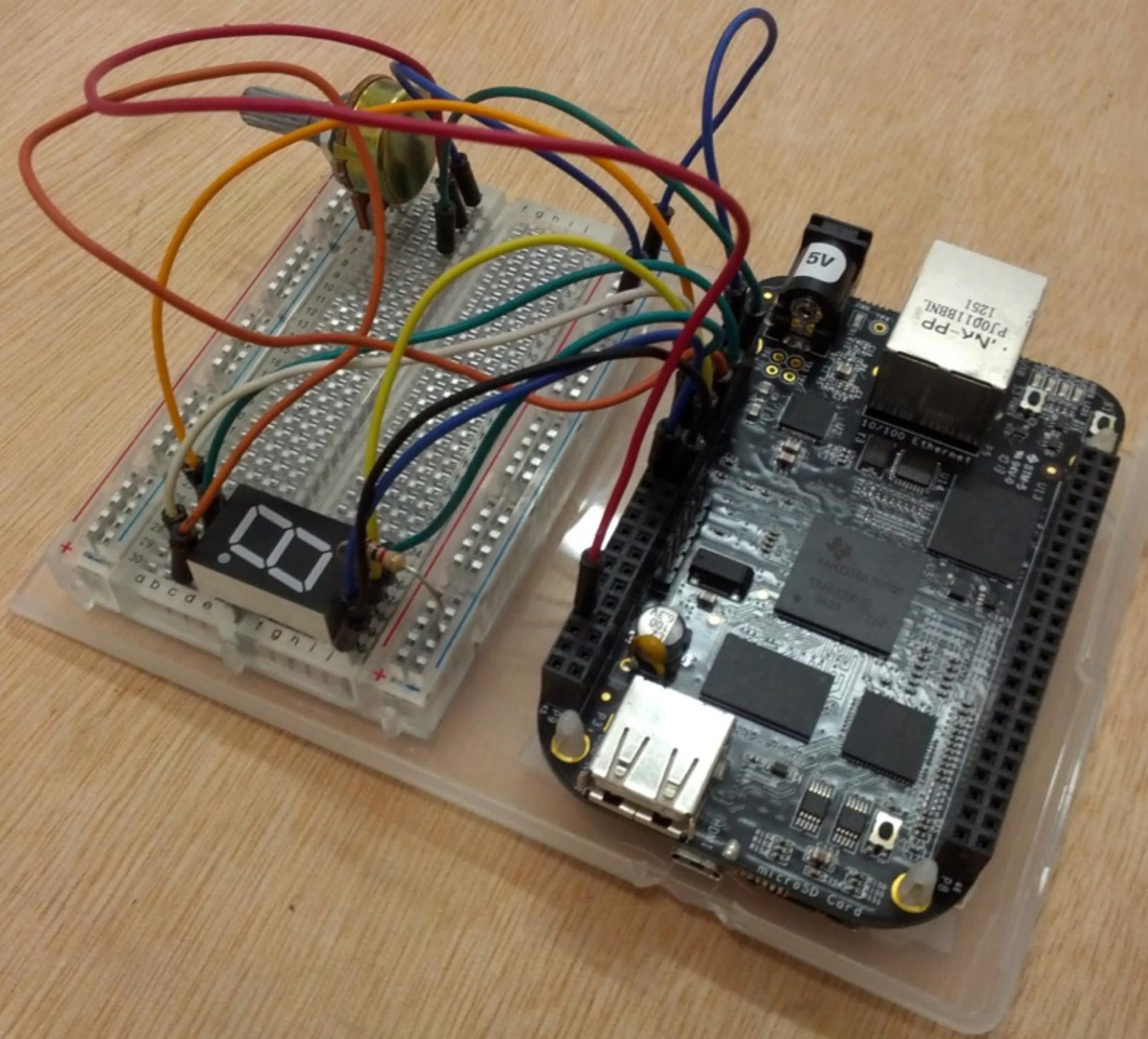
BEAGLEBONE BLACK



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

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)
```



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

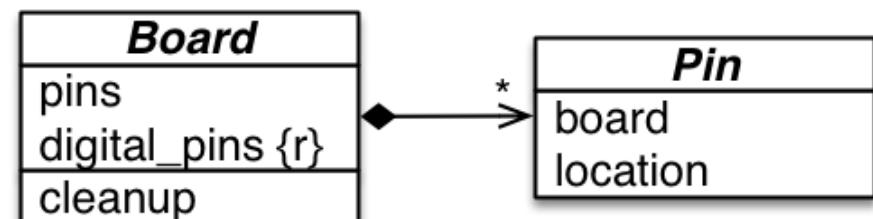
- Implementam métodos de controle específicos de cada placa
- Disponíveis em maio de 2014

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

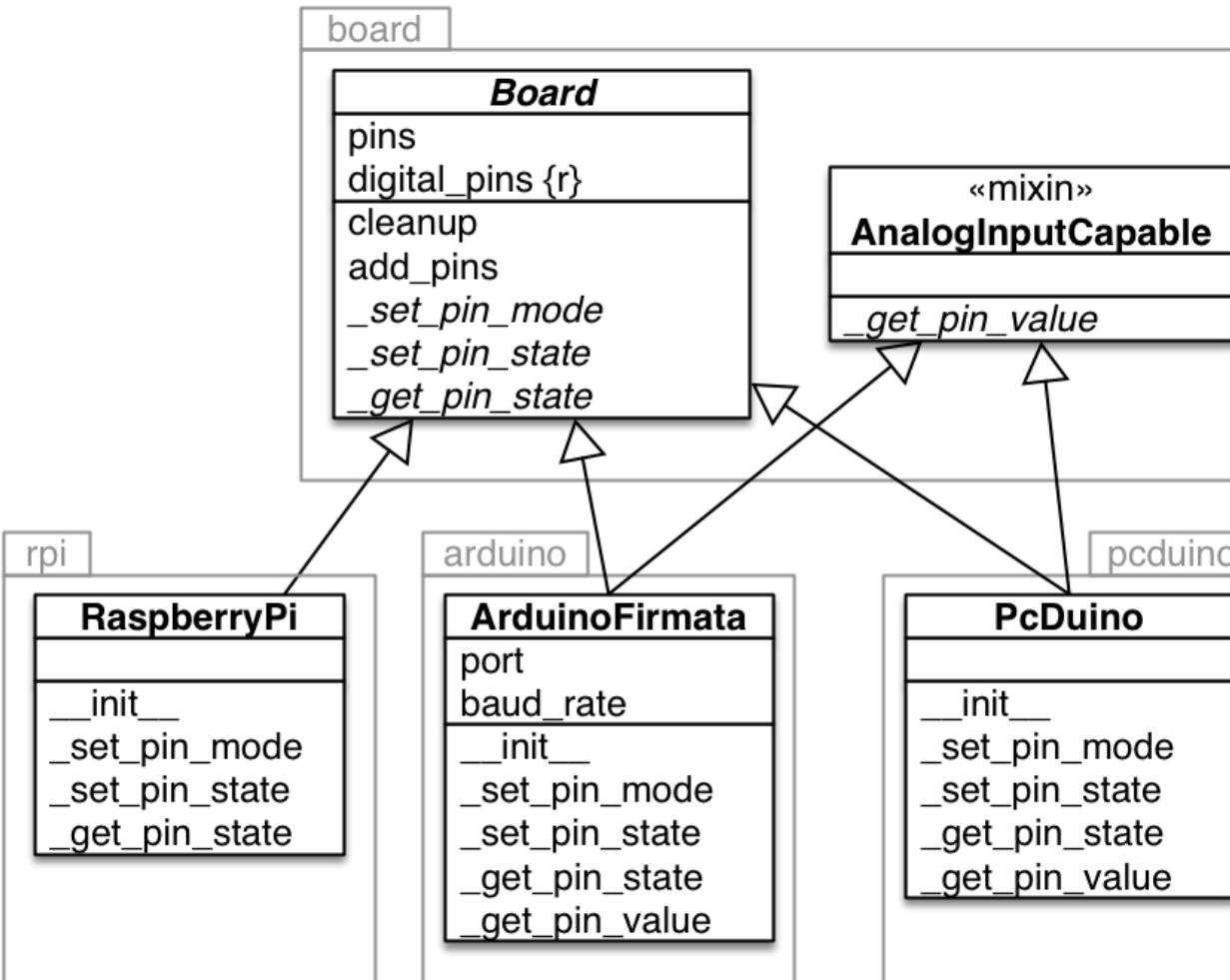


OBJETOS BÁSICOS

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



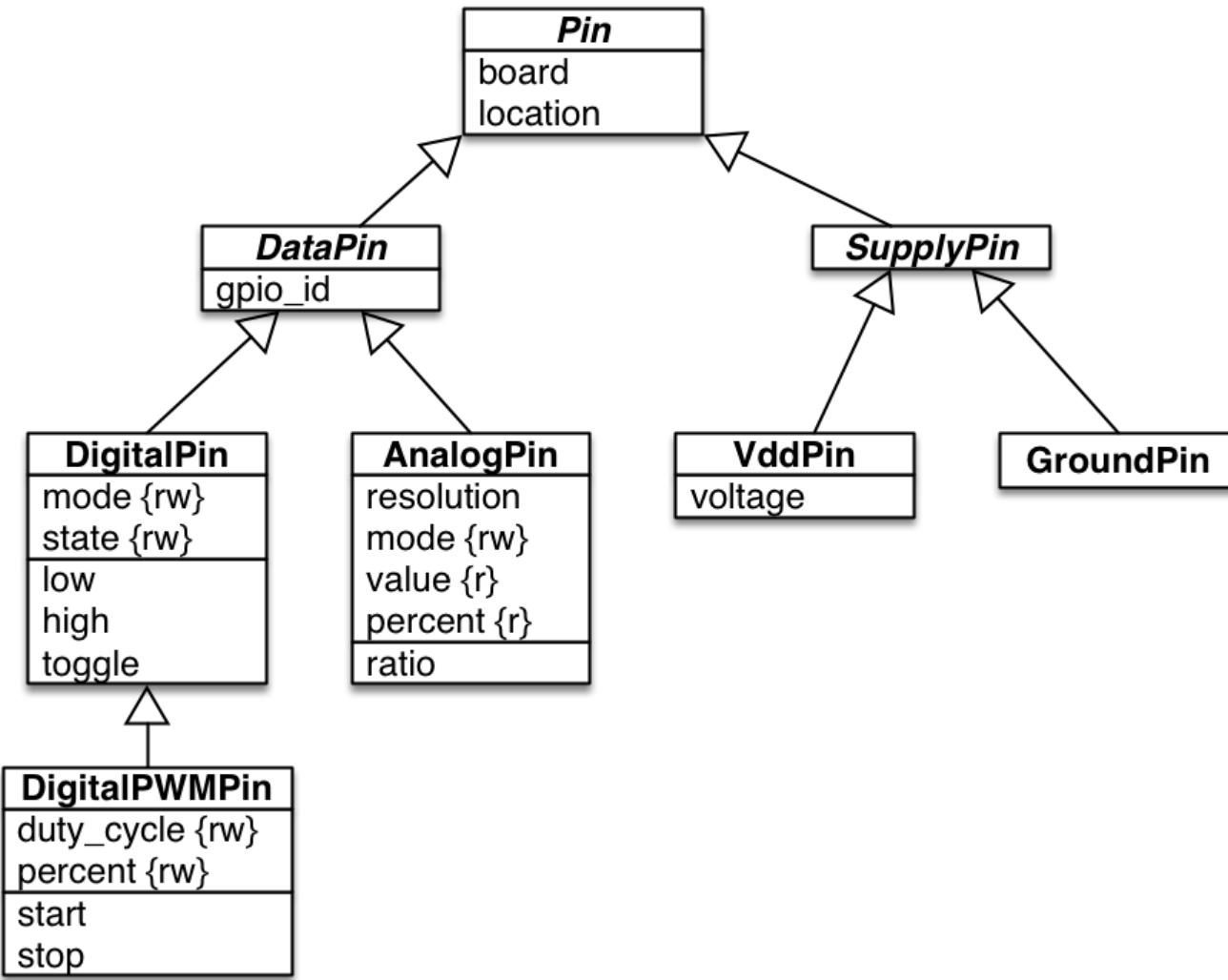
PLACAS



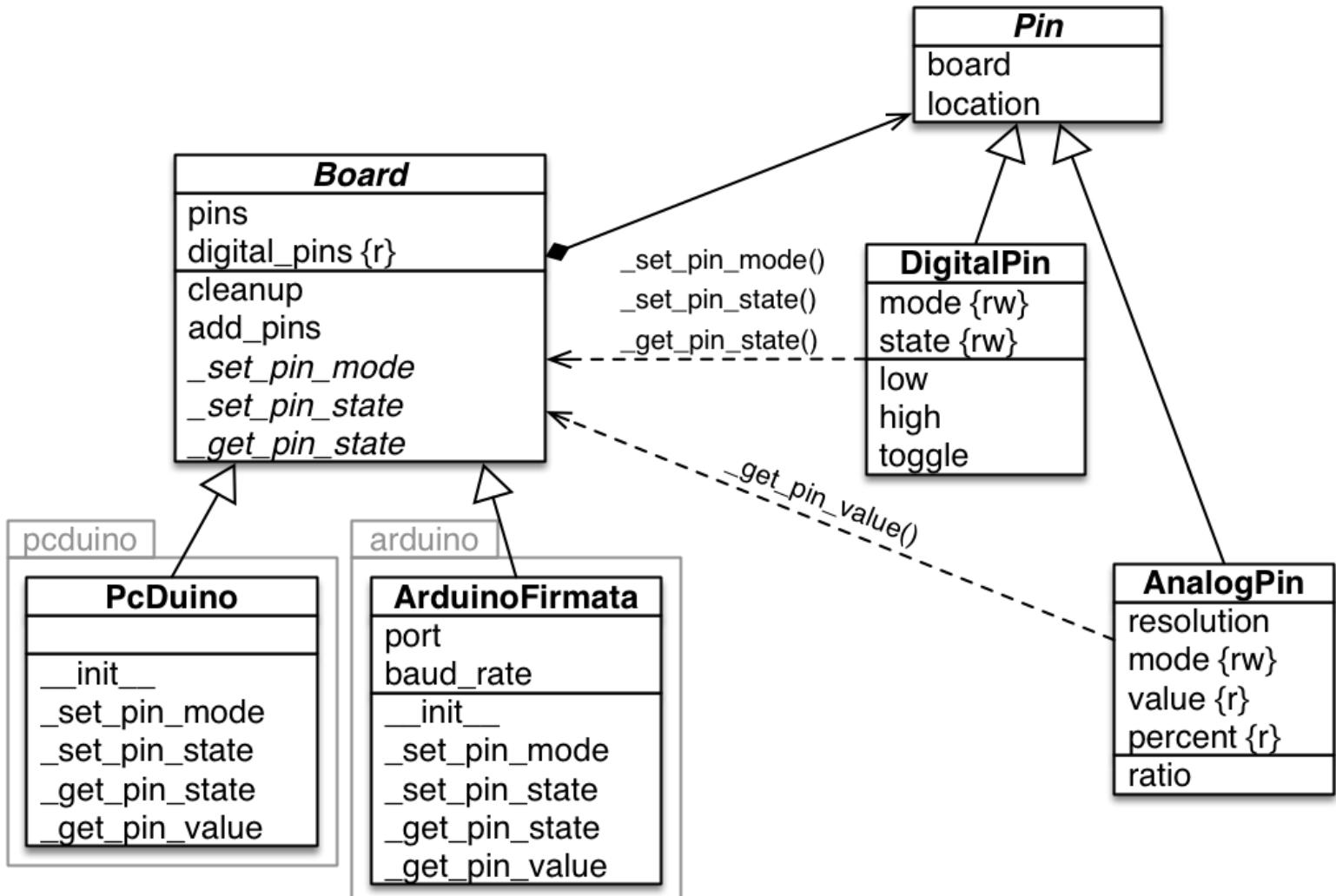
• • •



PINOS



COLABORAÇÕES

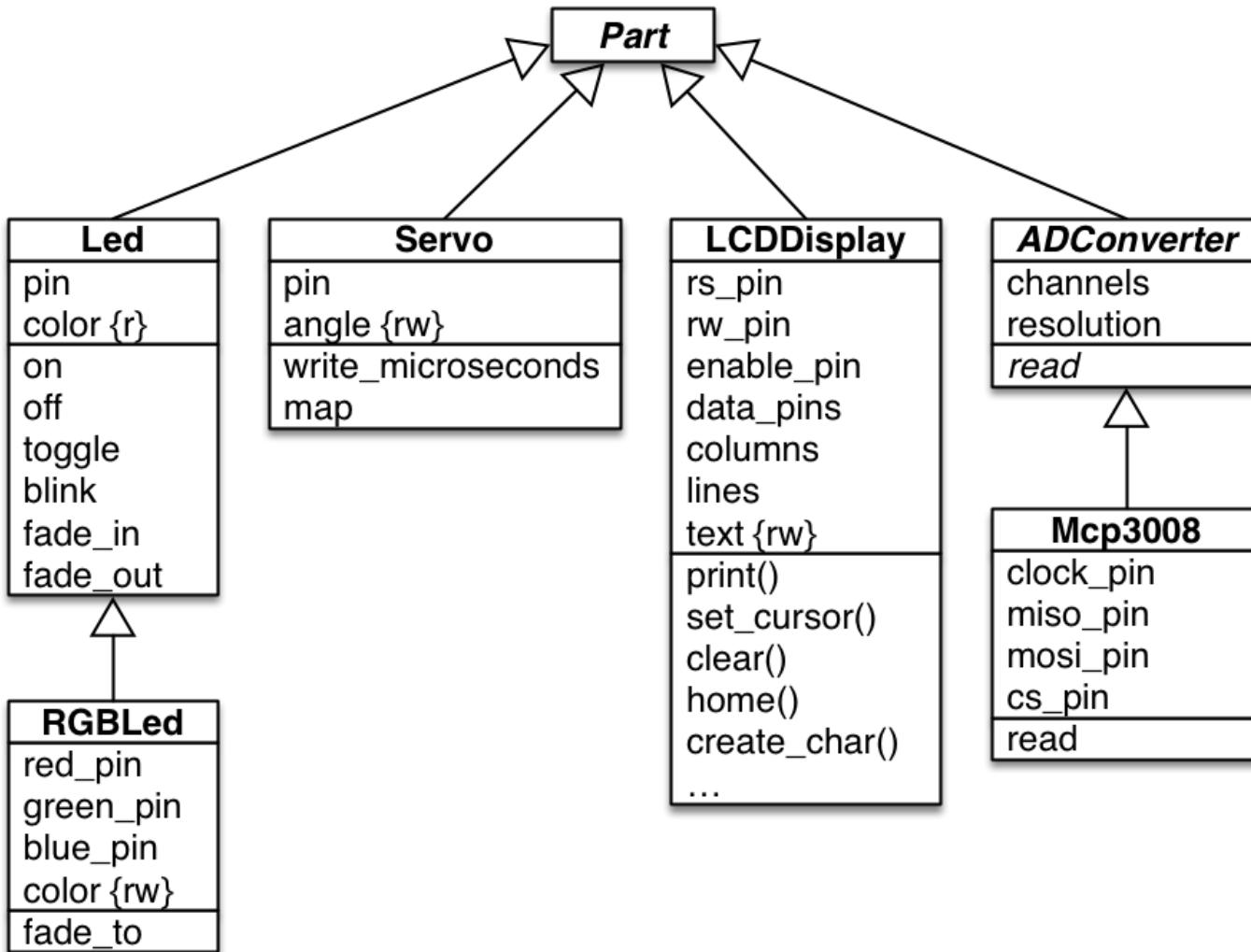


PLANOS

- Reestruturar sistema de testes
 - com mocks
 - com testes distribuídos em cluster de placas físicas
- Implementar pinos especializados: PWM, DAC, multi-função digital/analógico...
- Implementar protocolos: SPI, I2C...
- Implementar componentes



COMPONENTES



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 terça-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>

