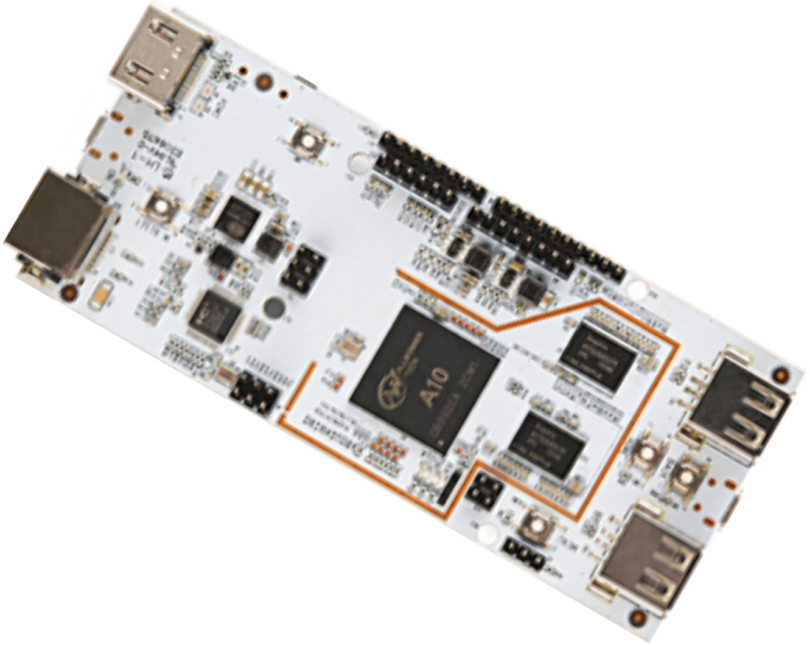


PiN GO

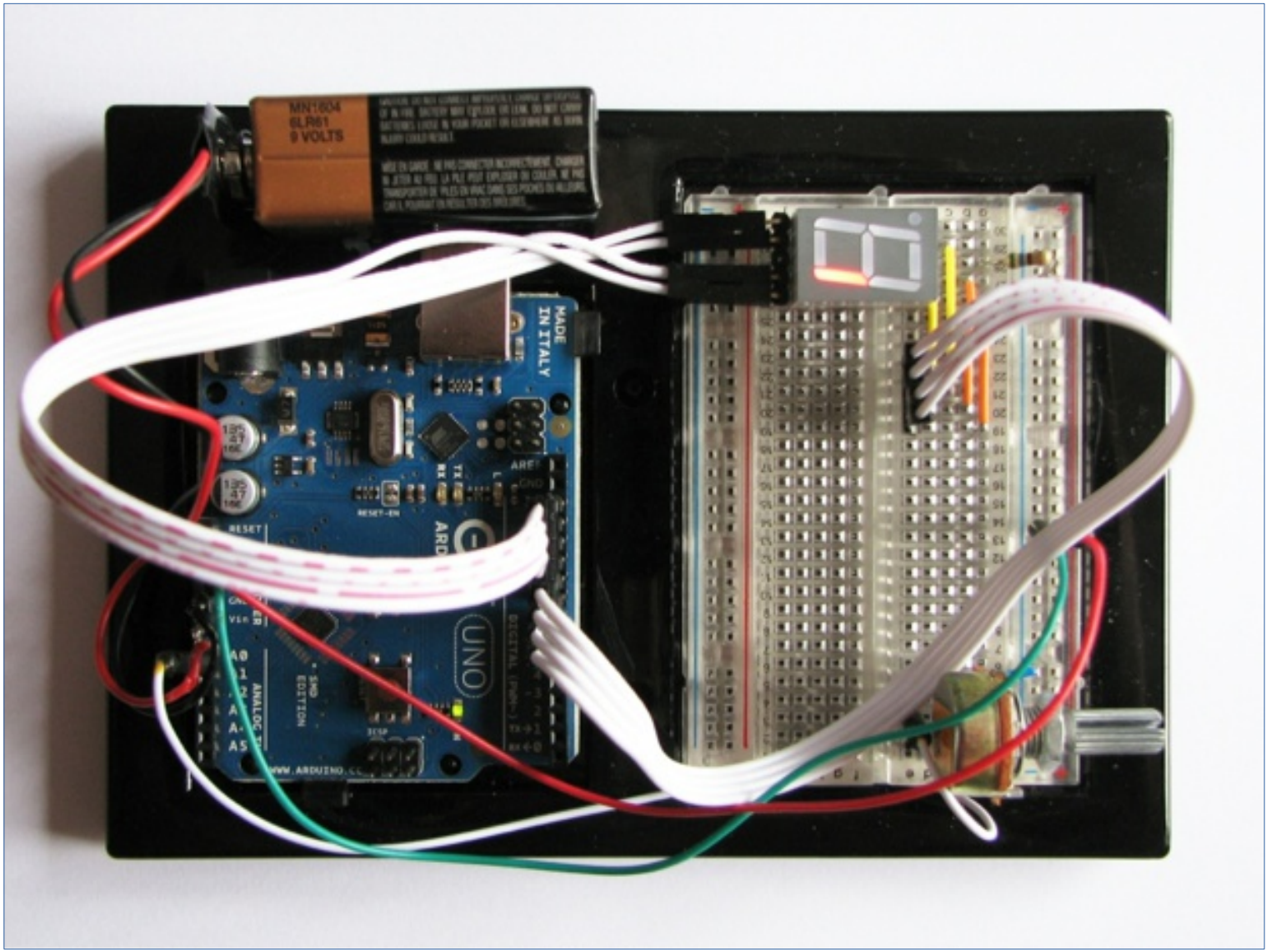
"UNIVERSAL" API FOR HARDWARE HACKING

THE “PROBLEM”

GPIO PINS

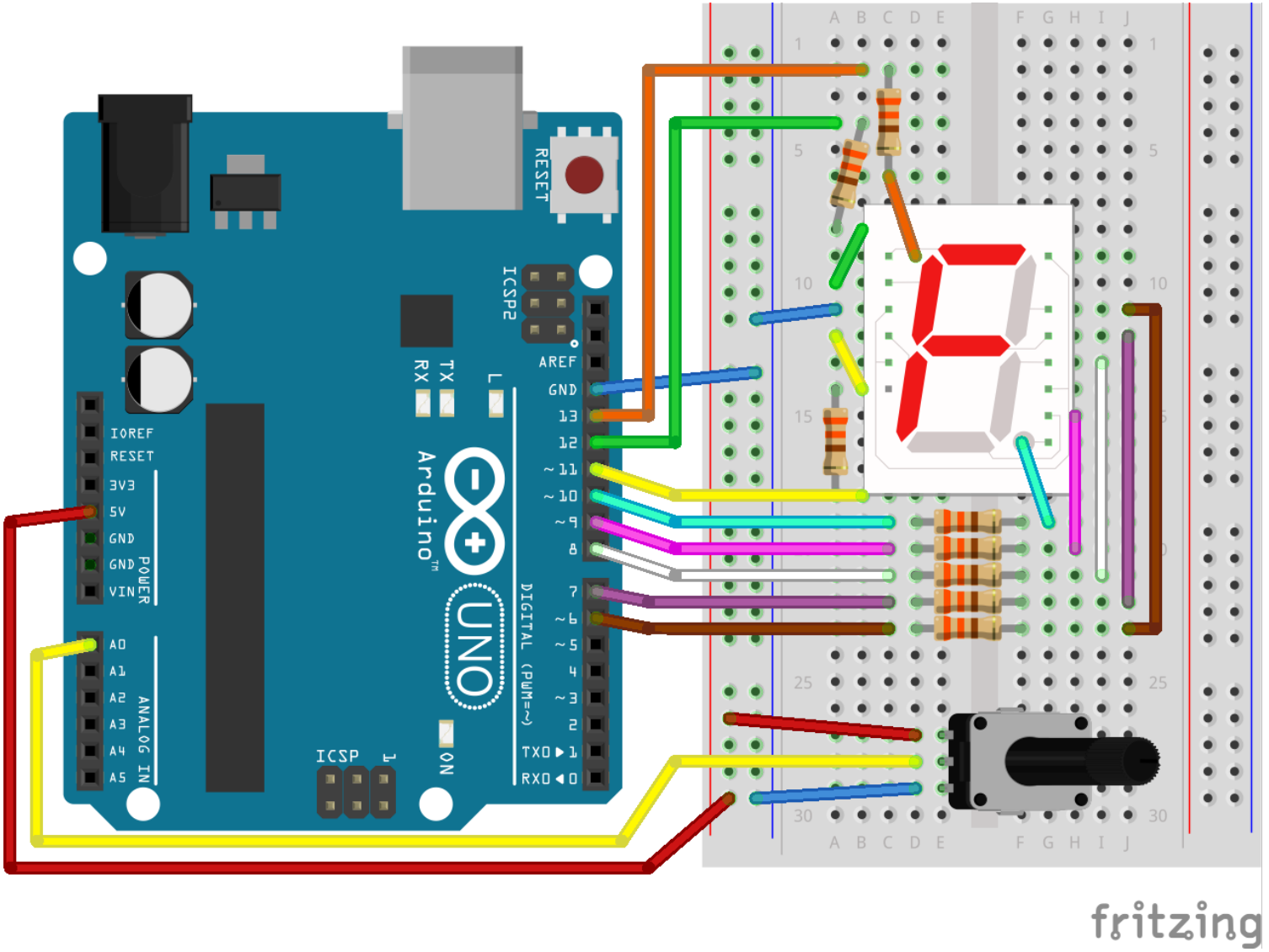
- All boards suitable for physical computing have **GPIO**
  - General Purpose I/O
- **Digital** I/O pins
  - 0/1 == HIGH/LOW
- **Analog** pins
  - A/D converter
- Digital pins with **PWM** support
  - Stand-in for analog output without a real D/A converter

ARDUINO CODING DOJO



GAROA HACKER CLUBE

ARDUINO CODING DOJO



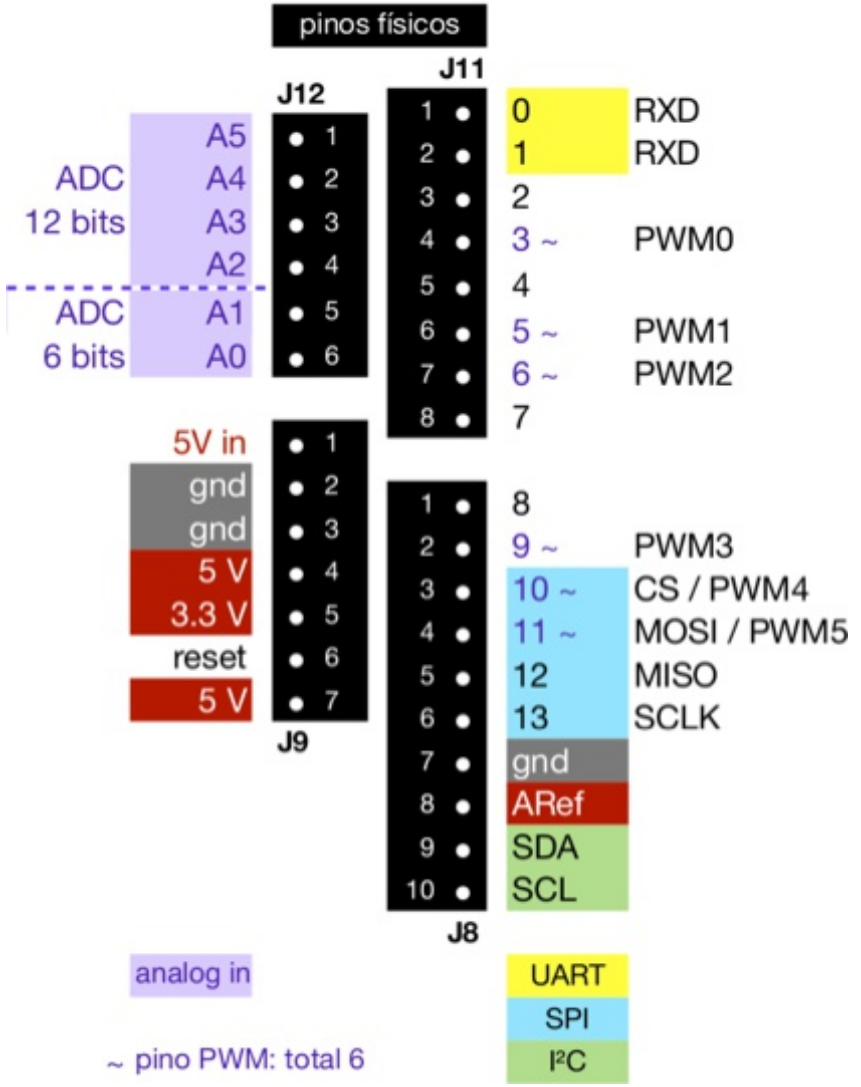
GAROA HACKER CLUBE

PCDUINO

	pcDuino
SoC	Allwinner A10
CPU	ARM Cortex A8 (ARMv7 + NEON SIMD)
GPU	Mali 400
clock	1 GHz
SO	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	✘
HDMI	HDMI
video composto	✘
audio	via HDMI
preço USD	59
Arduinos	1.7
open hardware	?

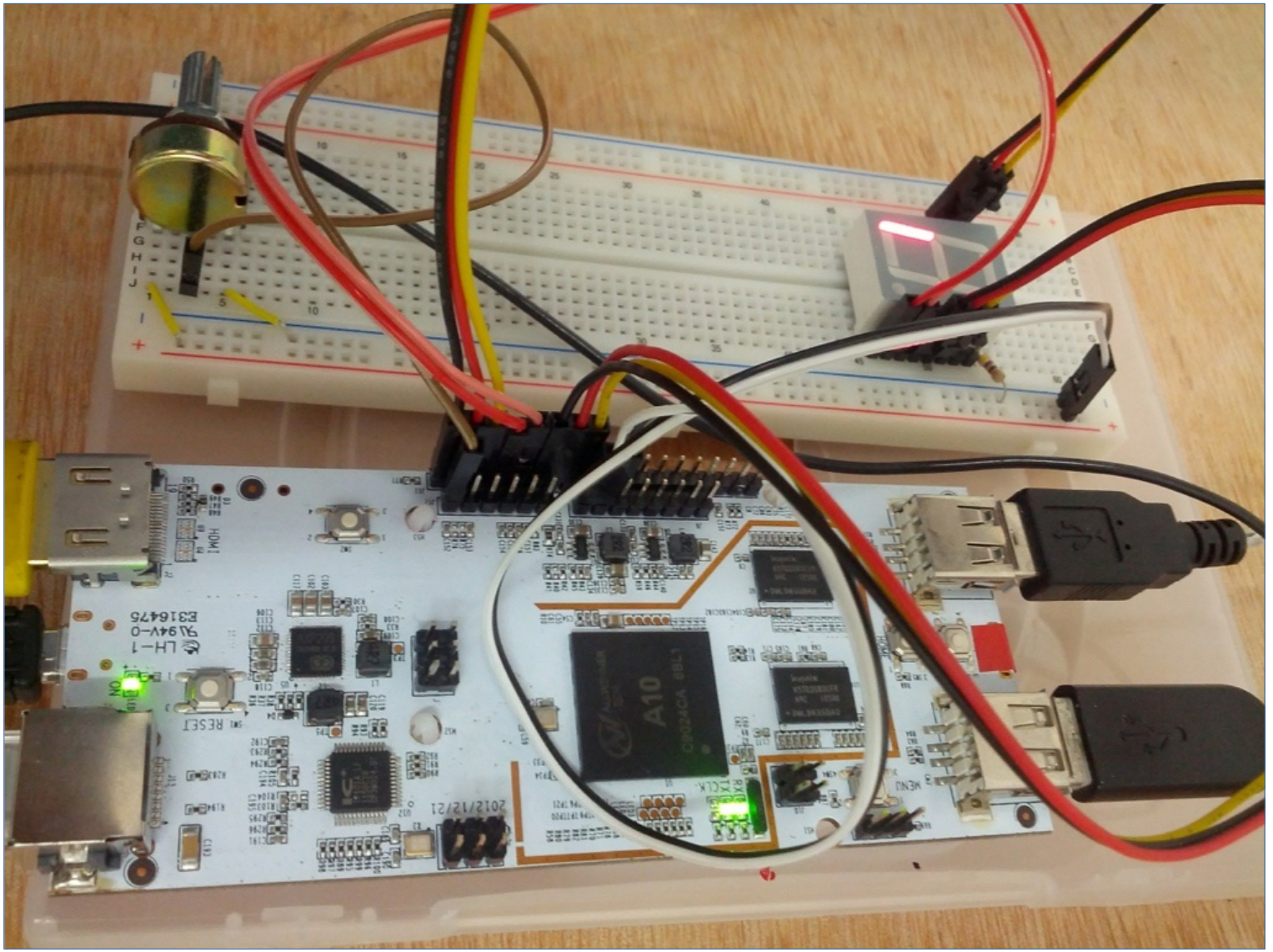
GAROA HACKER CLUBE

PCDUINO

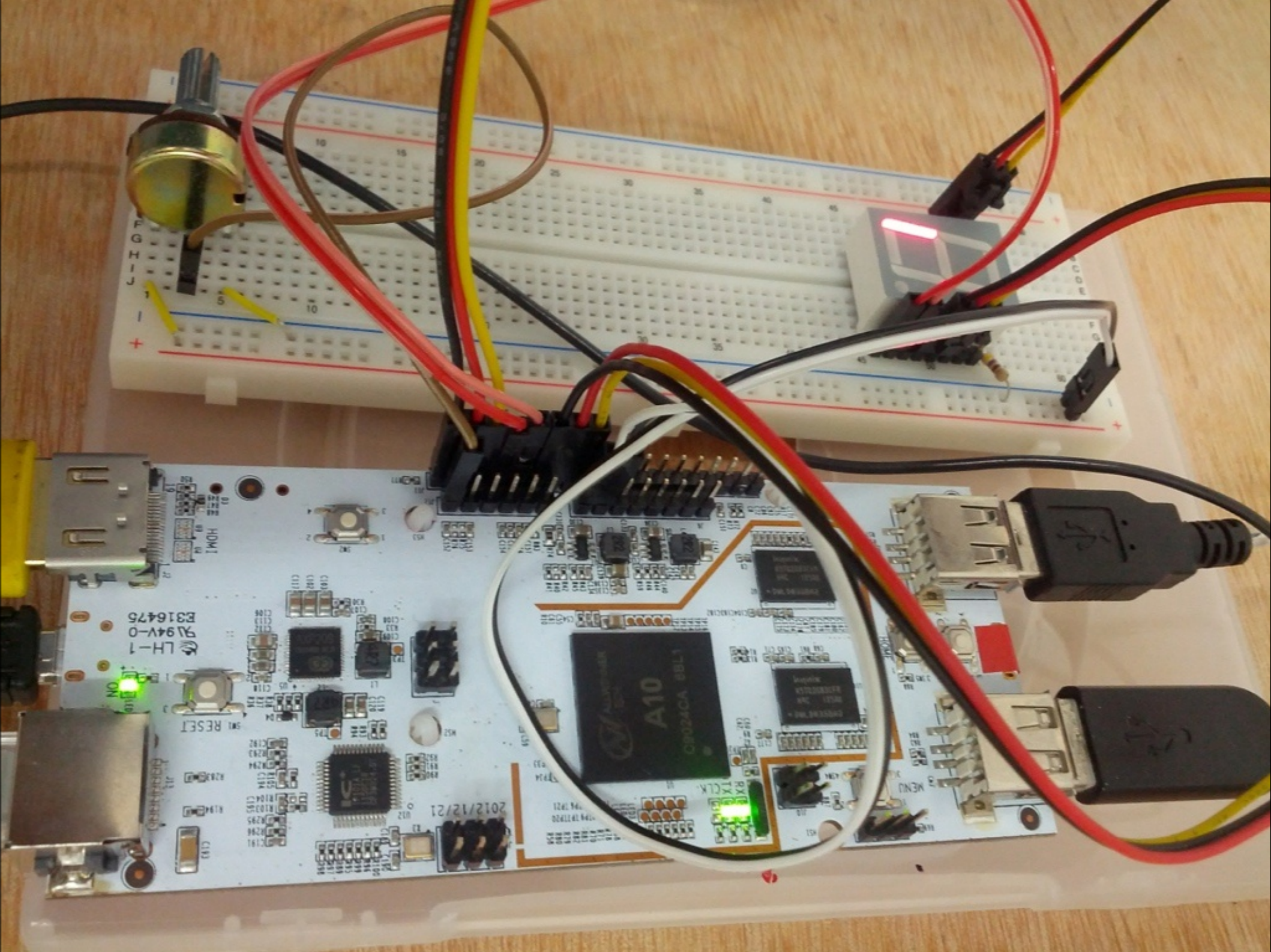


GAROA HACKER CLUBE

PCDUINO CODING DOJO



GAROA HACKER CLUBE


A10
2012/12/21

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	✘
SD card	SD
GPIO	17 (+4 no P5)
PWM	1
ADC	✘
USB client	✘
USB host	**2**
Ethernet	10/100
Wifi	✘
HDMI	**HDMI**
video composto	**RCA**
audio	**HDMI + plug 3.5mm**
preço USD	35
Arduinos	**1.0**
open hardware	✘

GAROA HACKER CLUBE

RASPBERRY Pi

P1

P1

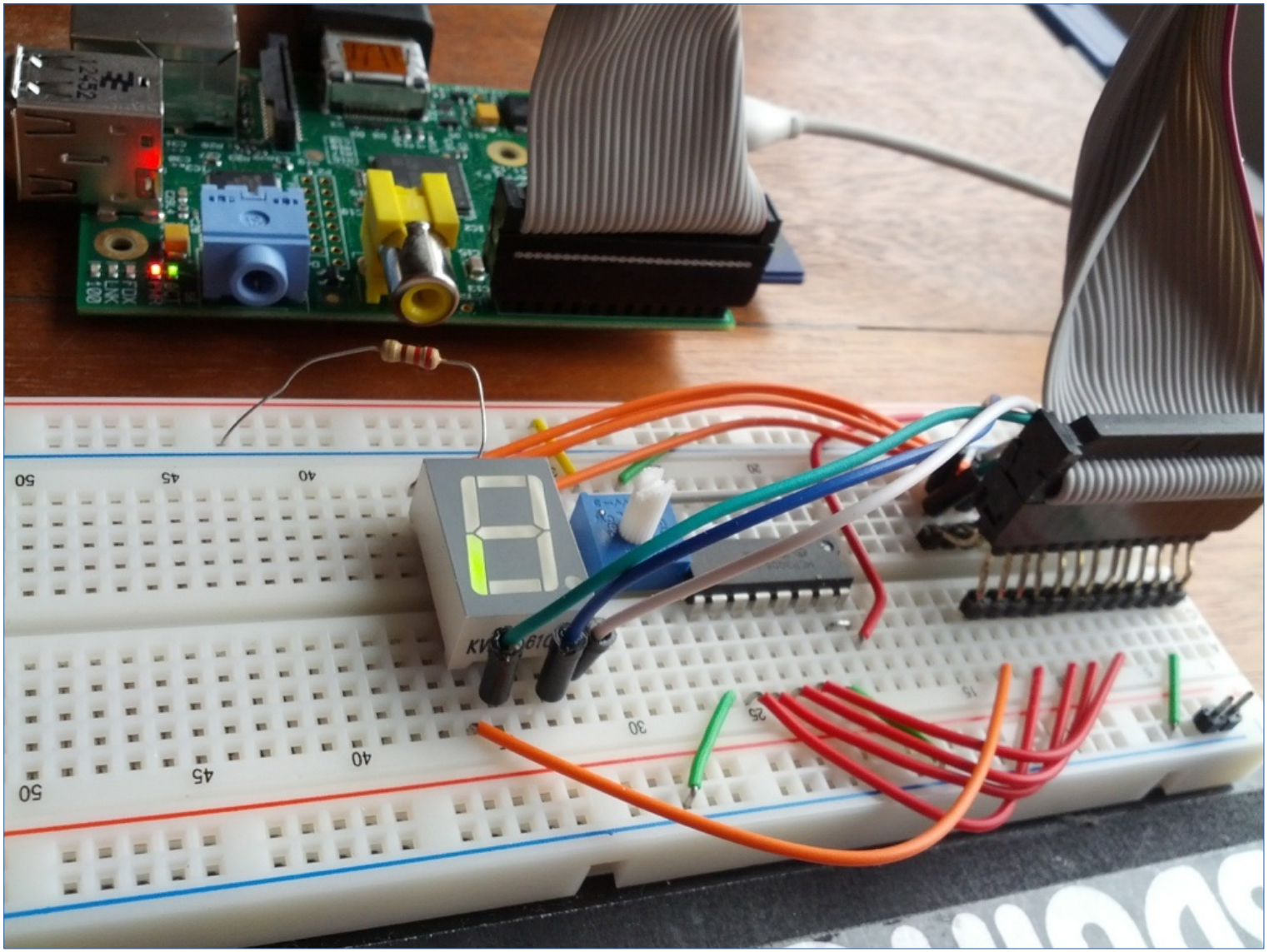
		pinos físicos			
	3.3 V	1	2	5 V	
SDA	* 0→2	3	4	5 V	
SCL	* 1→3	5	6	gnd	
GPCLK0	4	7	8	14	TXD
	gnd	9	10	15	RXD
	17	11	12	18 ~	PCM_CLK
	* 21→27	13	14	gnd	
	22	15	16	23	
	3.3 V	17	18	24	
MOSI	10	19	20	gnd	
MISO	9	21	22	25	
SCLK	11	23	24	8	CE0
	gnd	25	26	7	CE1

~ pino PWM: 12 (GPIO_18)
* pinos GPIO renumerados
rev.1→rev.2

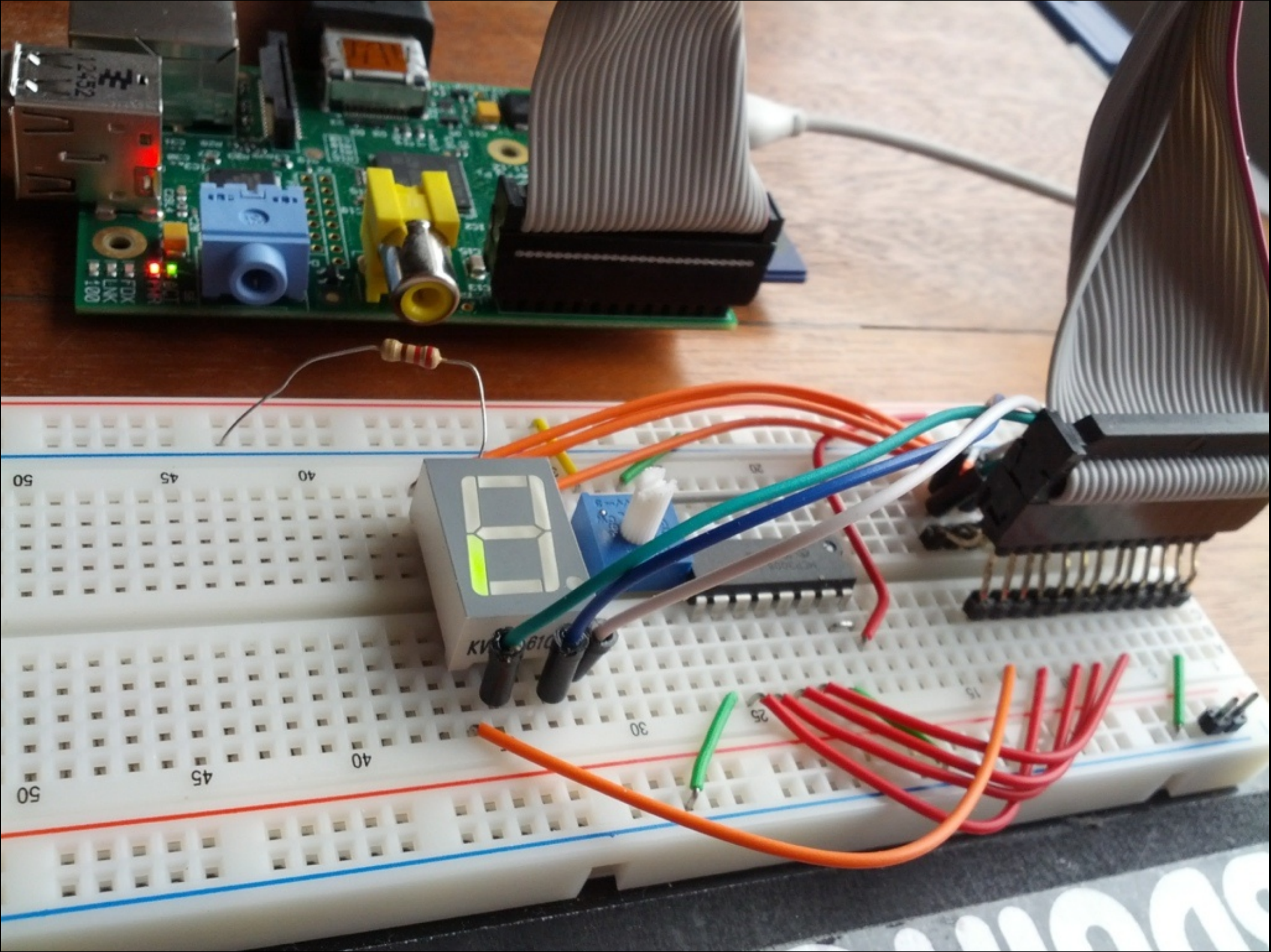
I²C
UART
SPI

GAROA HACKER CLUBE

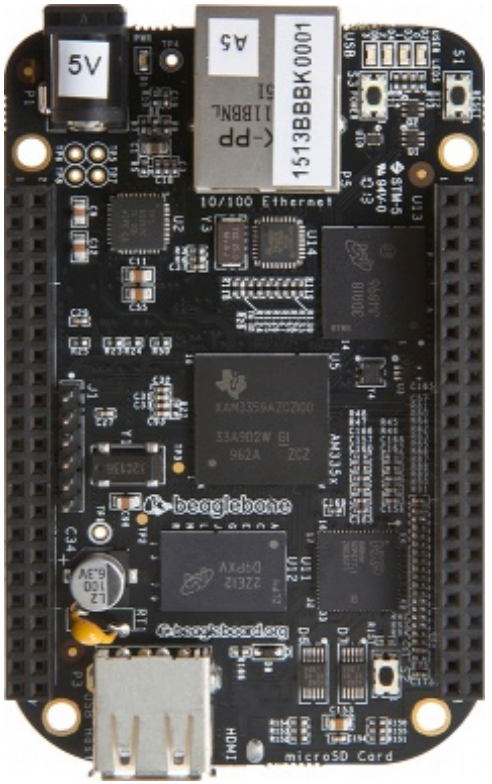
RASPBERRY Pi DOJO...



GAROA HACKER CLUBE



BEAGLEBONE BLACK



	Beaglebone Black
SoC	Texas Instruments Sitara AM335x
CPU	**ARM Cortex A8 (ARMv7 + NEON SIMD)**
GPU	PowerVR SGX 530
clock	**1 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	**1**
USB host	1
Ethernet	10/100
Wifi	✘
HDMI	micro-HDMI
video composto	✘
audio	via HDMI
preço USD	45
Arduinos	1.3
open hardware	✔

GAROA HACKER CLUBE

BEAGLEBONE BLACK

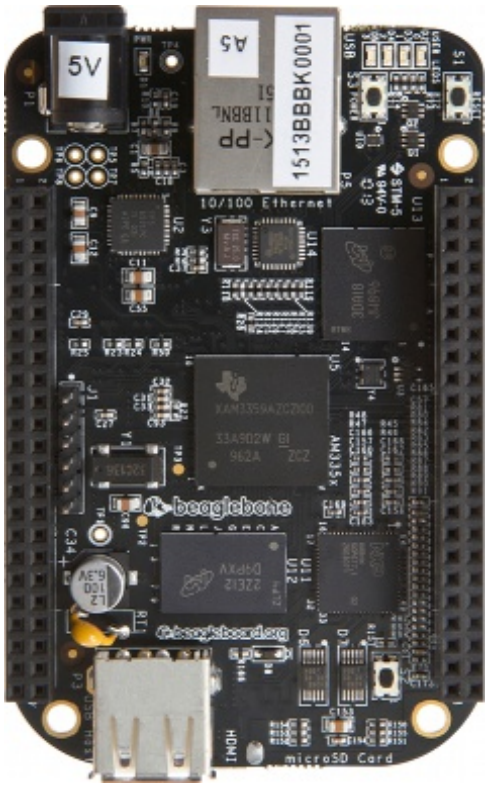
P9

	pinos físicos	
gnd	1 2	gnd
3.3 V	3 4	3.3 V
VDD 5 V	5 6	5 V VDD
SYS 5 V	7 8	5 V SYS
PWR_BUT	9 10	SYS_RESETn
UART4_RXD	11 12	1-28
UART4_TXD	13 14	EHRPWM1A ~
1-16	15 16	EHRPWM1B ~
I2C1_SCL	17 18	I2C1_SDA
I2C2_SCL	19 20	I2C2_SDA
UART2_TXD	21 22	UART2_RXD
1-17	23 24	UART1_TXD
* 3-21	25 26	UART1_RXD
3-19	27 28	SPI1_CS0 ~
SPI1_D0	29 30	SPI1_D1
SPI1_SCLK	31 32	VADC
AIN4	33 34	AGND
AIN6	35 36	AIN5
AIN2	37 38	AIN3
AIN0	39 40	AIN1
[2] CLKOUT2, 3-20	41 42	0-7 ~, 3-18 [2]
gnd	43 44	gnd
gnd	45 46	gnd

x-yy pino GPIOX_YY
~ pino PWM: 14, 16, 28, 42[2] (0-7)
* em uso: oscilador audio HDMI
[2] pino físico ligado a dois pinos lógicos

analog input
I²C
UART
SPI
misc.

P9 P8



P8

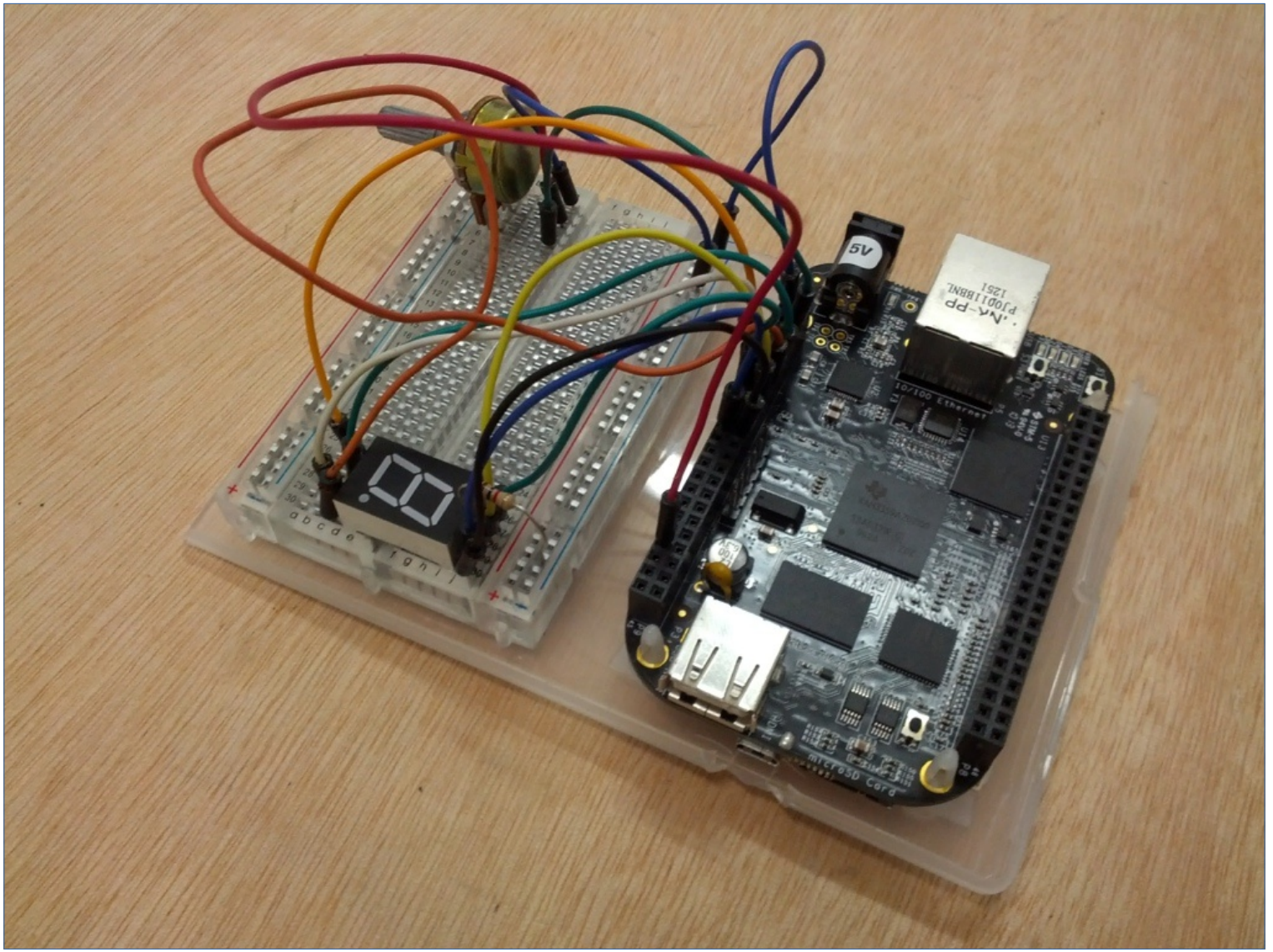
gnd	pinos físicos	gnd
1-6	1 2	1-7
1-2	3 4	1-3
TIMER4	5 6	TIMER7
TIMER5	7 8	TIMER6
1-13	9 10	1-12
~ EHRPWM2B	11 12	0-26
1-15	13 14	1-14
0-27	15 16	2-1
~ EHRPWM2A	17 18	1-31
1-30	19 20	1-5
1-4	21 22	1-1
1-0	23 24	1-29
2-22	25 26	2-24
2-23	27 28	2-25
UART5_CTSN	29 30	UART5_RTSN
UART4_RTSN	31 32	UART3_RTSN
UART4_CTSN	33 34	UART3_CTSN
UART5_TXD	35 36	UART5_RXD
2-12	37 38	2-13
2-10	39 40	2-11
2-8	41 42	2-9
2-6	43 44	2-27
	45 46	

x-yy pino GPIOX_YY
~ pino PWM: 13, 19

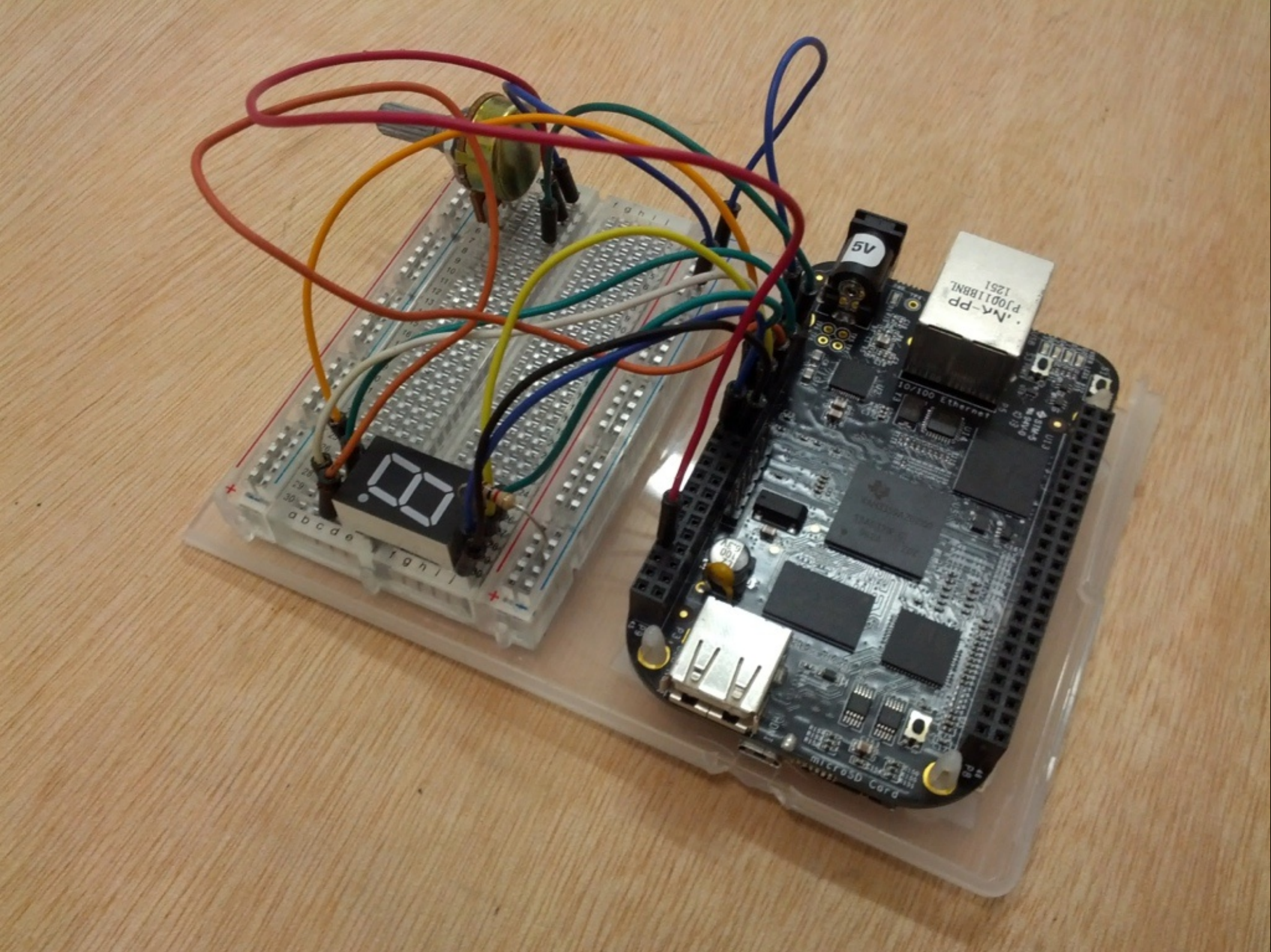
UART
timers

GAROA HACKER CLUBE

BBB CODING DOJO



GAROA HACKER CLUBE


5V
10/100 Ethernet
microSD Card

PROGRAMMING PINS

- Almost every board is programmable in **Python**
- Python onboard, when they run embedded **GNU/Linux**
- Remotely controlled by Python in **another** computer
  - ex: **Arduino** via **Firmata** over serial with the **PyMata** library

```
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: GPIO mapped on the filesystem

No special libraries used in this script

GAROA HACKER CLUBE

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)
```

Two specific libraries: RPi.GPIO and spi (homebrew)

RASPBERRY Pi

GAROA HACKER CLUBE

```
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)
```

Specific library: Adafruit_BBIO

GAROA HACKER CLUBE

BEAGLEBONE BLACK

OUR ANSWER

DEMO

INTERACTIVE MODE BLINK

```
>>> 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: <Digita lPin @7>, 8: <DigitalPin @8>, 9:
<DigitalPin @9>, 10: <DigitalPin @10>, 11: <DigitalPin @11>, 12:
<DigitalPin @12>, 13: <DigitalPin @13>, 'A1': <Analog Pin @A1>, 'A0':
<AnalogPin @A0>, 'A3': <AnalogPin @A3>, 'A2': <AnalogPin @A 2>, '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)
...
```

GAROA HACKER CLUBE

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)
```

GAROA HACKER CLUBE

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)
```

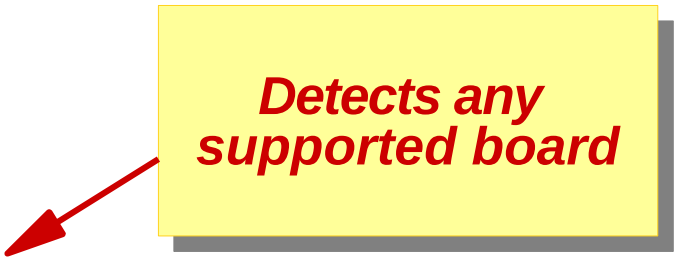
Detect Arduino over serial/USB

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)
```



GAROA HACKER CLUBE

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 to 12

This script should run on any supported board that has analog input pins.

Depending on the board layout, the pin ids may have to be changed.

GAROA HACKER CLUBE

CONCEPTS

PRINCIPLES

- Object oriented API
  - Enables interactive exploration
  - Easy to extend to new devices
- Usability
  - Default: pins identified by physical location
  - Convenience methods and attributes: toggle, digital_pins...
- Best practices
  - Idiomatic Python
  - Automated testing

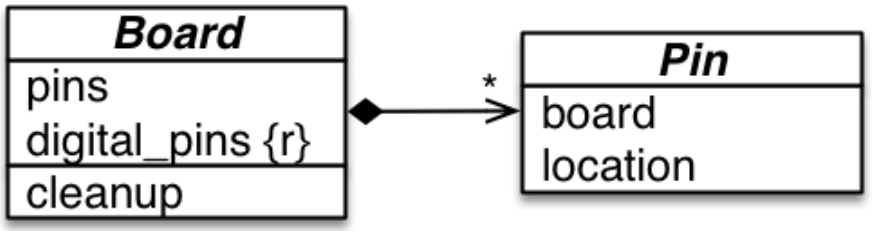
BOARD DRIVERS

- Drivers implement board-specific control methods
- Drivers available in June 2014:

driver	operation	functionality
ArduinoFirmata	remote	digital + analog
PcDuino	on board	digital + analog
Raspberry Pi	on board	digital
UDOO	on board	digital

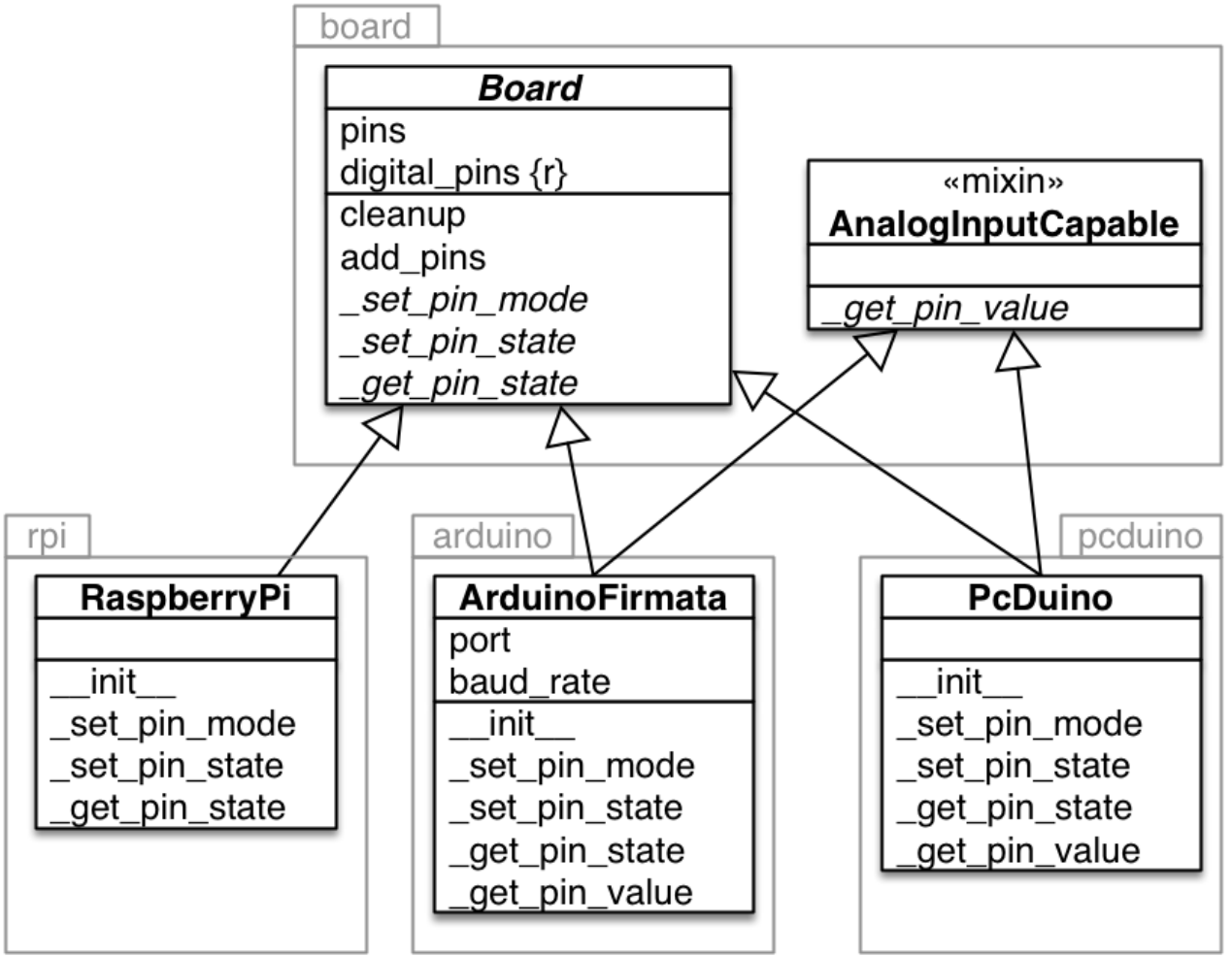
BASIC OBJECTS

- pingo.Board
  - pingo.arduino.ArduinoFirmata
  - pingo.rpi.RaspberryPi
  - pingo.pcduino.PcDuino
  - ...
- pingo.Pin
  - pingo.DigitalPin
  - pingo.AnalogPin
    - ...



GAROA HACKER CLUBE

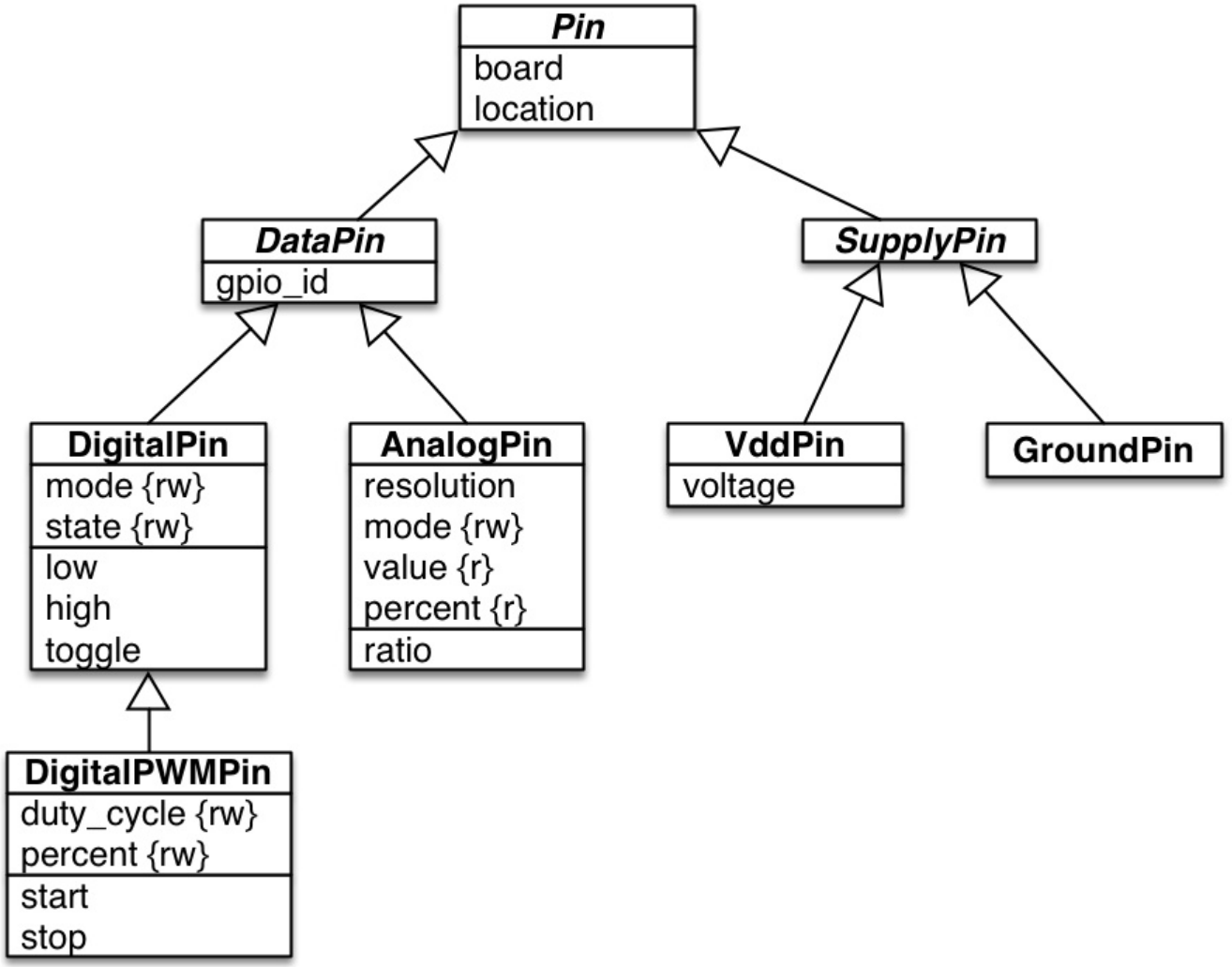
BOARDS



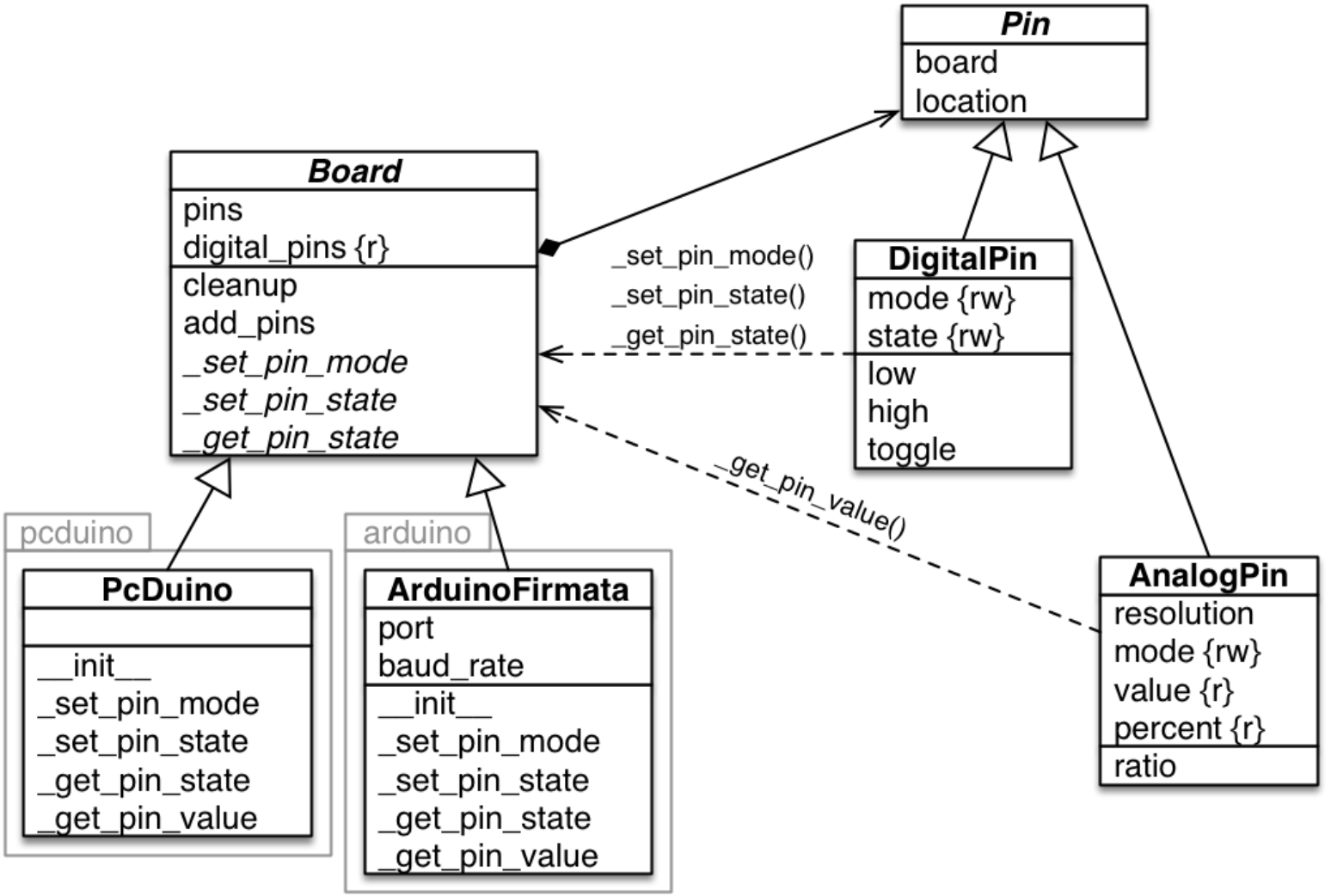
. . .

GAROA HACKER CLUBE

Pins



COLLABORATIONS

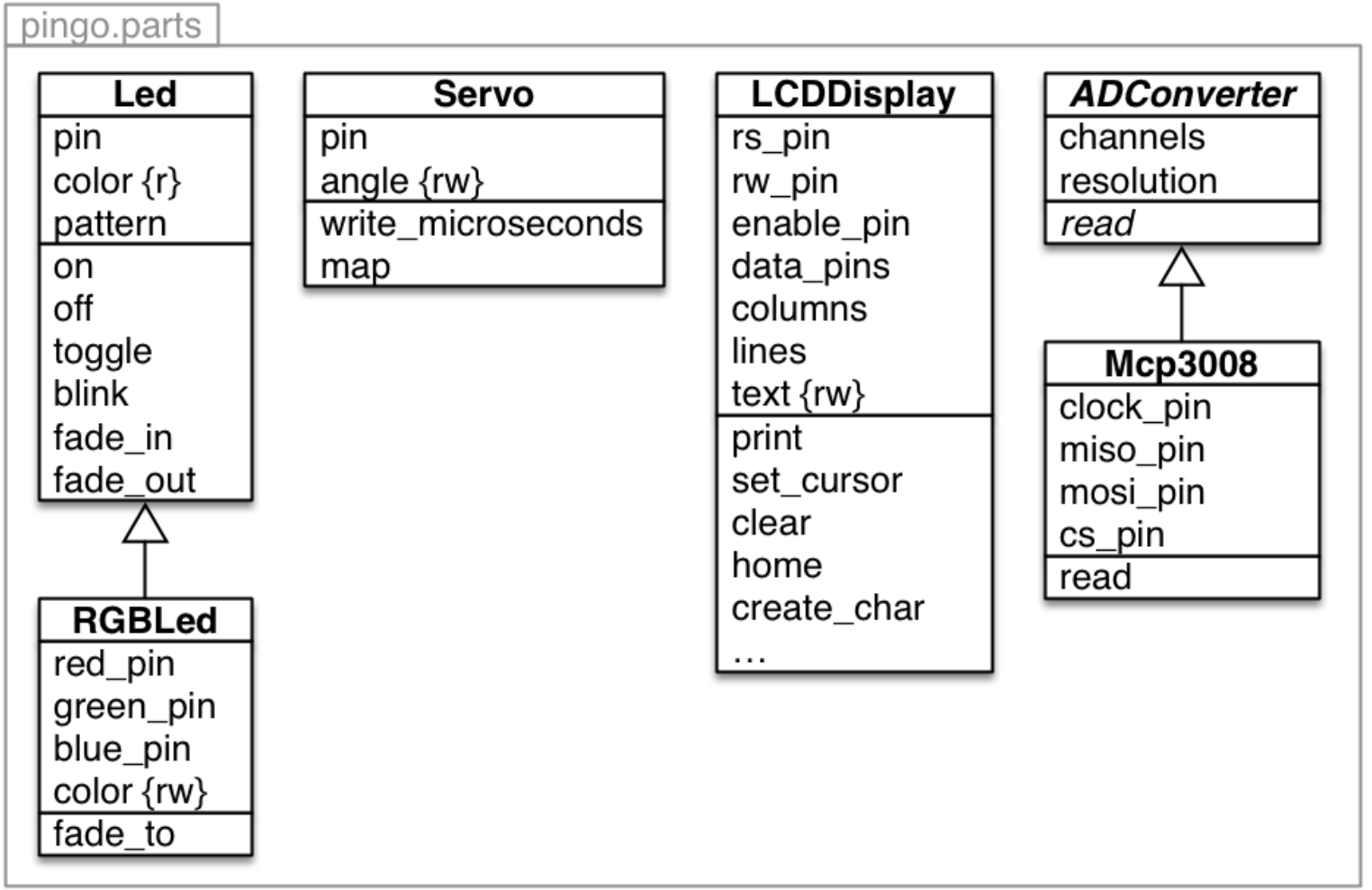


GAROA HACKER CLUBE

PLANS

- Drivers for: Intel Galileo, Arduino Yún, BeagleBone Black, Arduino Tre
- Reestructure tests
  - with tests distributed over 'cluster' of physical boards
- Implement specialized pins: PWM, DAC, multi-function digital/analog...
- Implement protocols: SPI, I2C...
- Implement parts

PiNGO PARTS PACKAGE



GAROA HACKER CLUBE

JOIN US!

- Free/libre project on initial stages
  - Interesting architectural decisions yet to be made
  - Any contribution is noted and makes a difference
- Docs: http://pingo.io
- Repo: http://github.com/garoa/pingo
- Google Groups: pingo-io
  http://groups.google.com/forum/#!forum/pingo-io