

## Contents

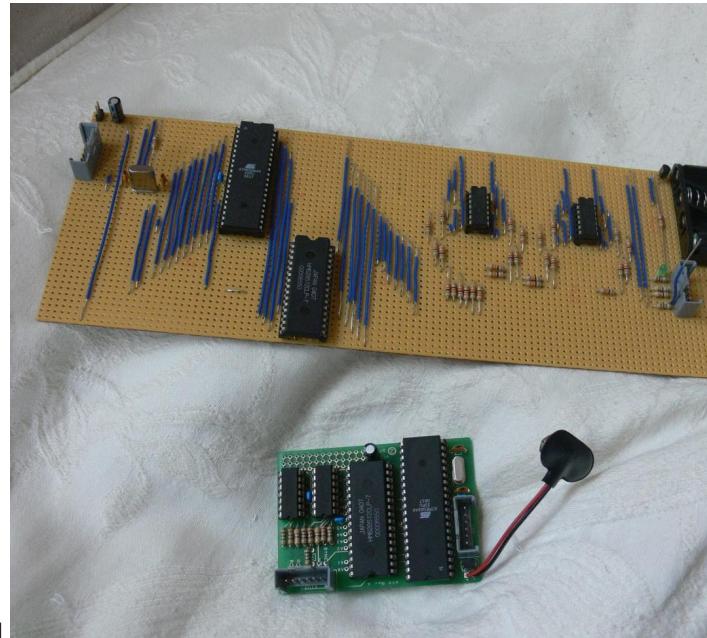
Orígenes	2
¿Clones?	6
Proyectos	13
Raspberry en los medios	19
Iniciativas similares	20
¿Raspberry Pi vs Arduino?	21
OpenSource y licencias abiertas	24
Equipo	24
Dónde y qué comprar	27
Materiales	28
Arquitectura	28
Sistemas operativos disponibles	31
Instalación	31
Problemas	34
Mantenimiento	35
Manejando tu Raspberry Pi	35
Acceso directo	39
Usos	42
Hacking	54
Vigilancia	55
Simuladores	55
Exteriores	56
Aprendiendo a programar	57
Scratch	60

<b>Utilizando scripts</b>	<b>62</b>
<b>Python</b>	<b>67</b>
<b>Aplicaciones hechas en python</b>	<b>70</b>
<b>APIS : conectando con el mundo exterior</b>	<b>73</b>
<b>Python y la cámara: openCV</b>	<b>77</b>
<b>Mathematica en nuestra Raspberry Pi</b>	<b>78</b>
<b>Electronica</b>	<b>79</b>
<b>GPIO</b>	<b>82</b>
<b>Precauciones</b>	<b>83</b>
<b>Conectado un pulsador</b>	<b>87</b>
<b>Complementos (Addons)</b>	<b>109</b>
<b>RaspiFAQ</b>	<b>111</b>

## Orígenes

La intención era crear un sistema barato que nos permitiera enseñar a programar a niños y adultos.

- En 2005, la Universidad de Cambridge notó como cada vez menos estudiantes quería estudiar informática
- Pensaron que la solución podía ser un ordenador superbarato con el que pudieran jugar, aprendiendo en el camino.



- Se empezó a construir en 2006, pero era difícil
- En 2009 ya existía la tecnología necesaria y se creó “Raspberry Pi Fundation” administrada por Eben Upton
- En 2011 aparecen los primeros prototipos y se ven factibles modelos de 25\$ y 35\$. Aparece el modelo B Beta

Utiliza un diseño avanzado

## Versiónes

Version 3 (1.2GHz quad core 1Gb Wifi y bluetooth)

Version 0 (1Ghz 512Mb)

Versión 2 B (900MHz quad core y 1Gb)

Versión B+ 2014

Versión B (512M y ethernet)

Versión A (256MB)

La llegada de los clones

## Refencias

Artículo de la wikipedia sobre RaspBerry Pi

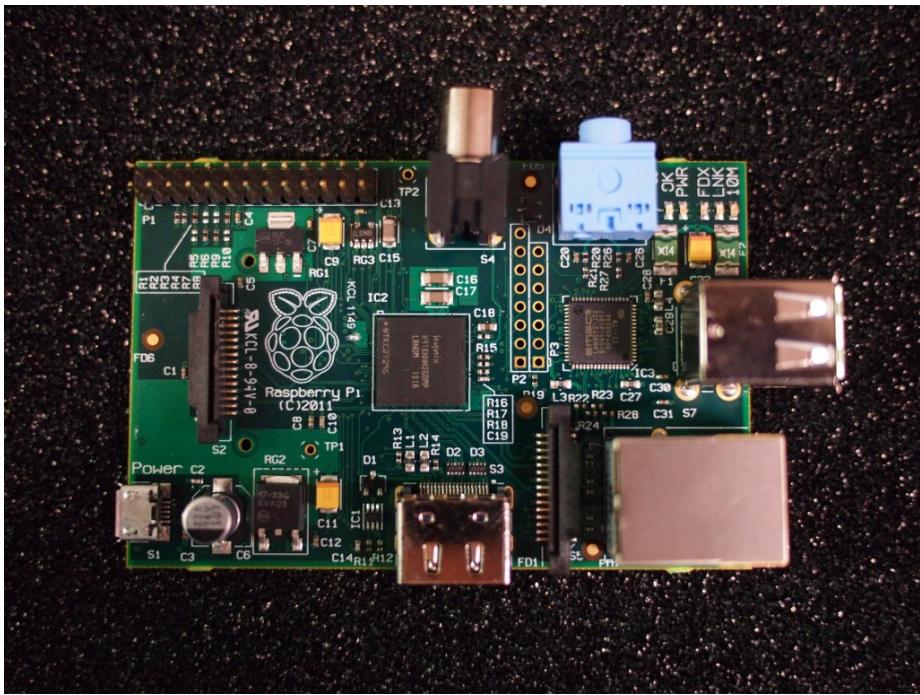


Figure 1: Placa beta de Raspberry Pi

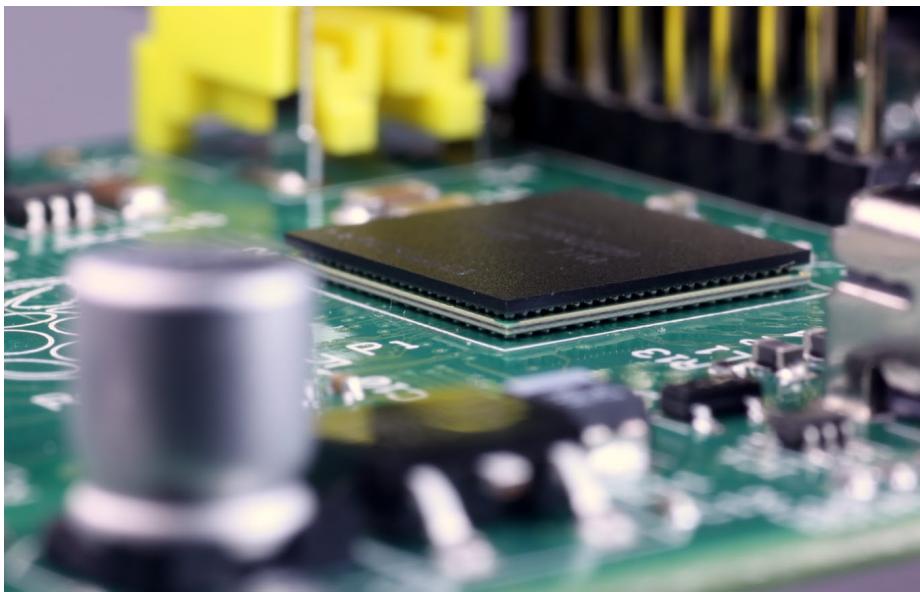


Figure 2: Diseño avanzado

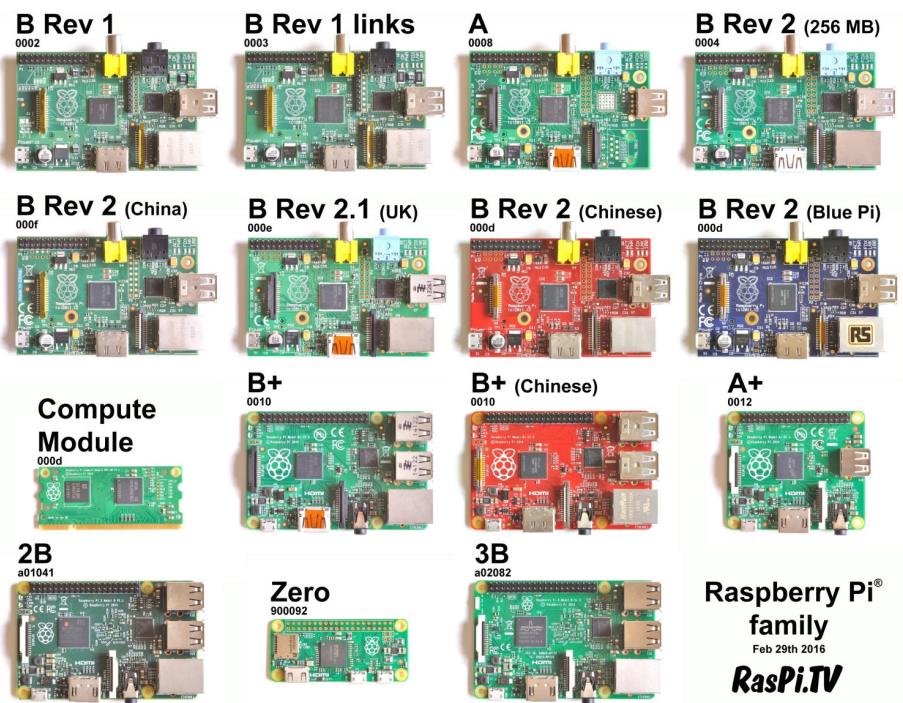


Figure 3: Diferentes modelos de Raspberry Pi

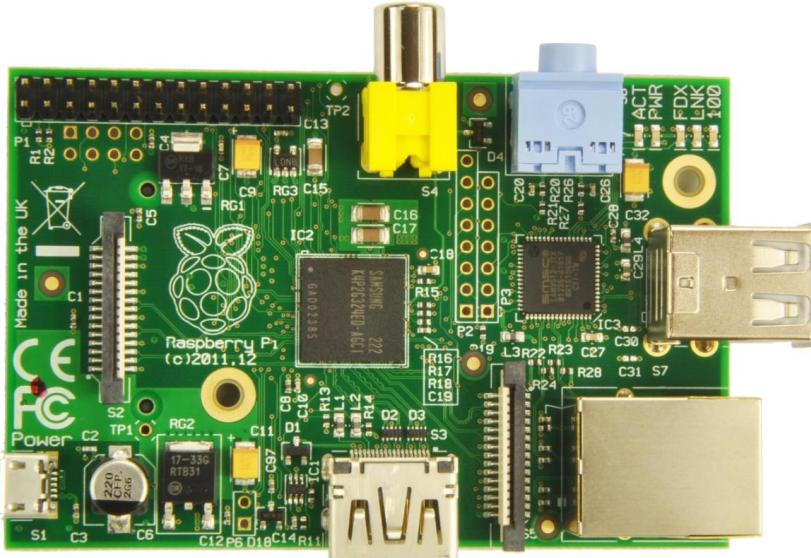


Figure 4: modeloB

Products at [Raspberry.org](http://Raspberry.org)

Tabla de los distintos modelos y sus características

Comparativa Raspberry 2

## ¿Clones?

- BeagleBoard ARM Cortex-A8 256MB 89€
- OLinuXino Cortex A8 1GHz 512 MB 55€

UDOO=raspberry+arduino 6 ARM Cortex-A9 CPU Dua/Quad core 1GHz 1GB

Cubieboard ARM cortex-A8 y 512 MB 49\$

Carambola 8devices (32Mb RAM) 22\$

Arduino Tre e Intel Galileo

Nanode y waspmote

Banana Pi

PCDuino



Figure 5: beagle

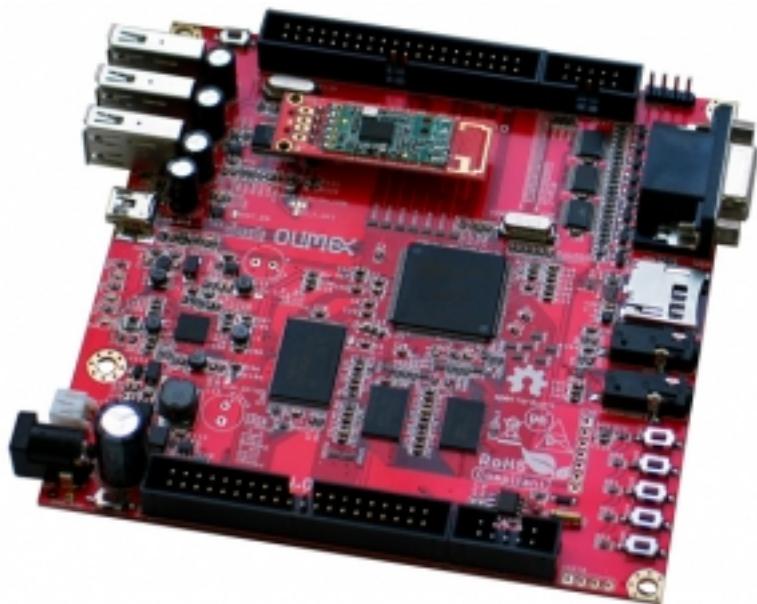


Figure 6: Olixunio

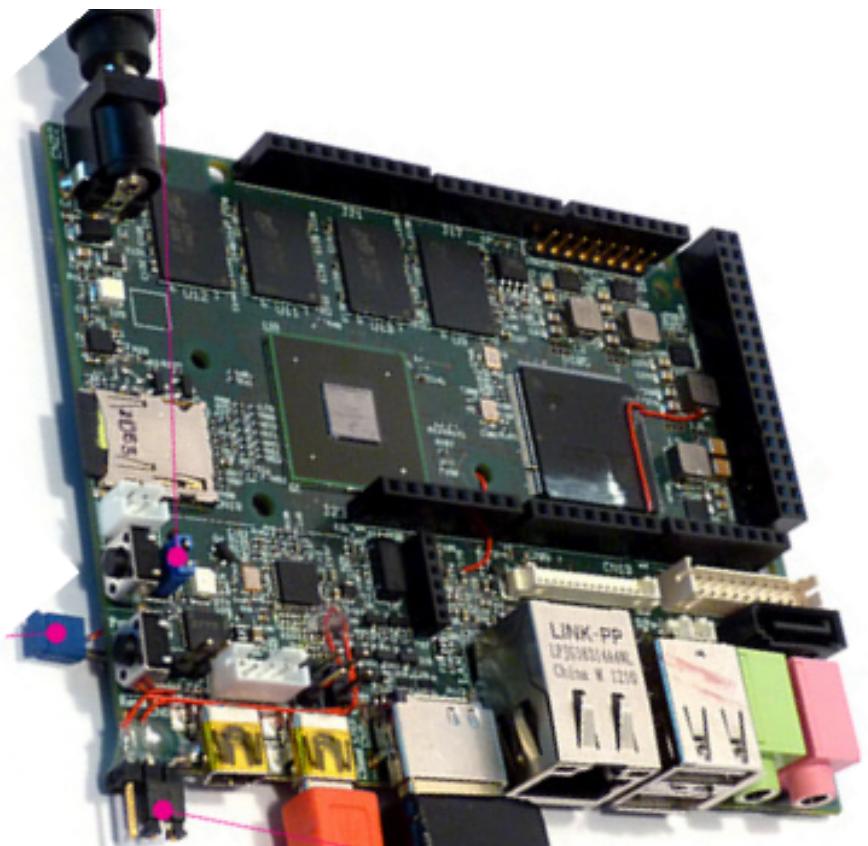


Figure 7: udoo

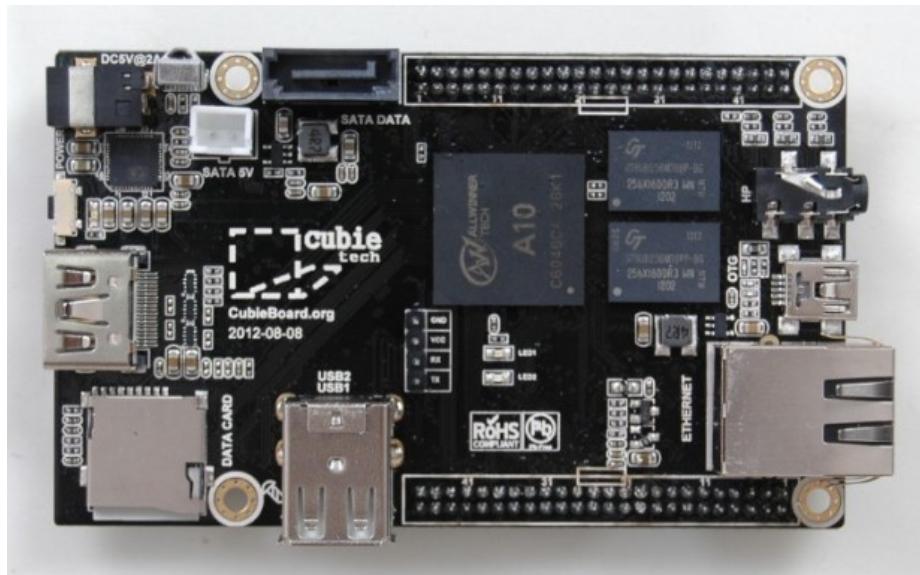


Figure 8: Cubieboard

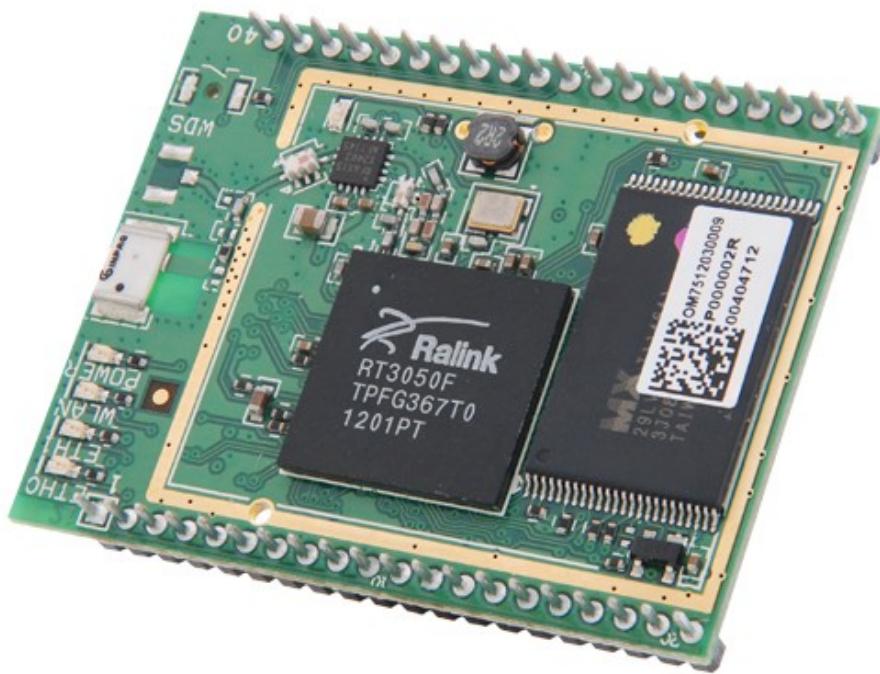


Figure 9: Carambola



Figure 10: Banana Pi

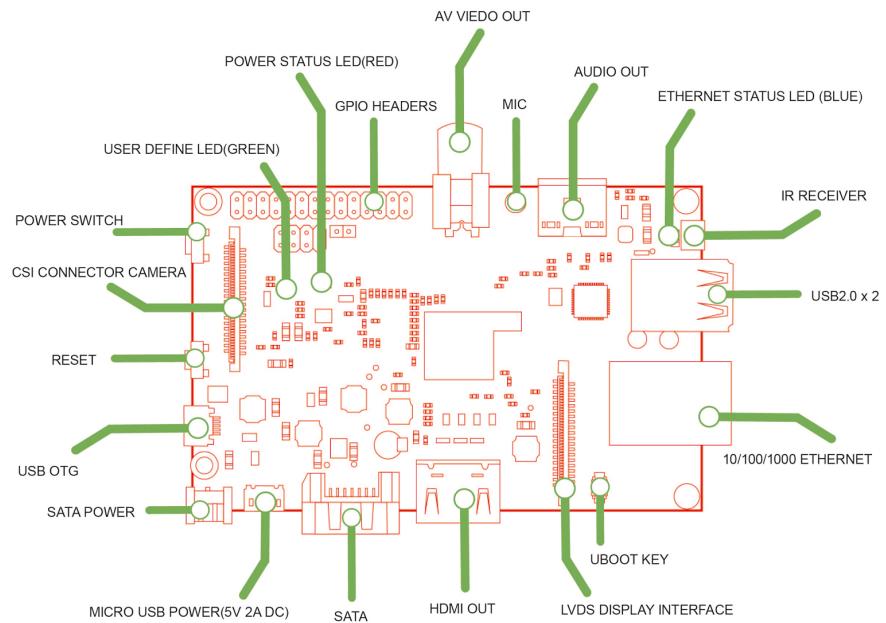
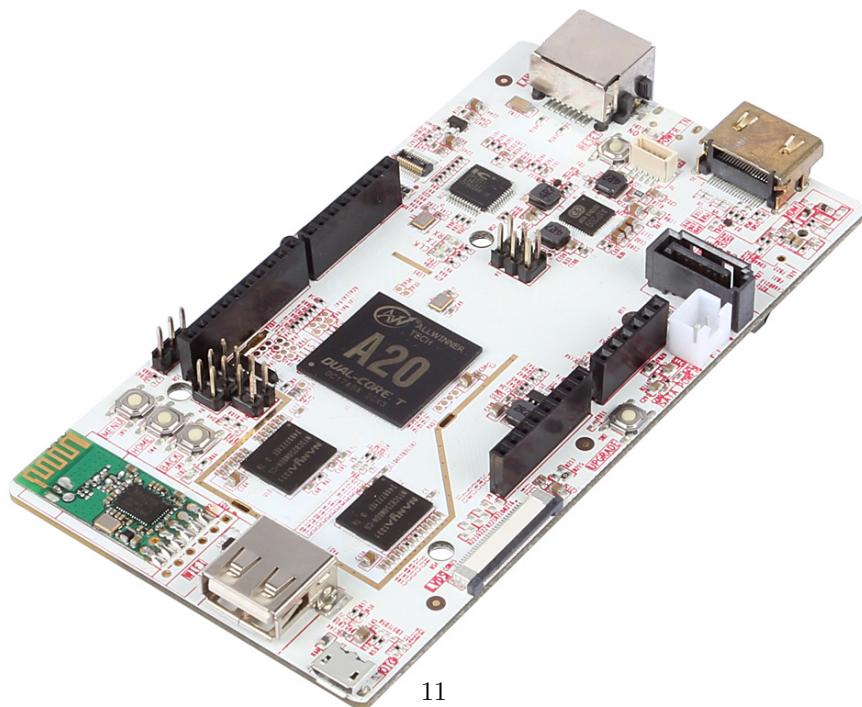


Figure 11: Esquema de Banana Pi



	Pin 1	Pin 2
3.3V1		
I2C-SDA		
I2C-SCL		
GCLK		
GND		
GPIO 0		
GPIO 2		
GPIO 3		
3.3V2		
SPI MOSI		
SPI MISO		
SPI CLK		
GND		
	Pin 25	Pin 26

Figure 12: GPio Banana Pi  
12

...

## Refencias

Alternativas

Otras placas similares

Comparativa entre clones

## Proyectos

### Aulas informatica de bajo coste



Figure 13: Aula con Raspberry Pi

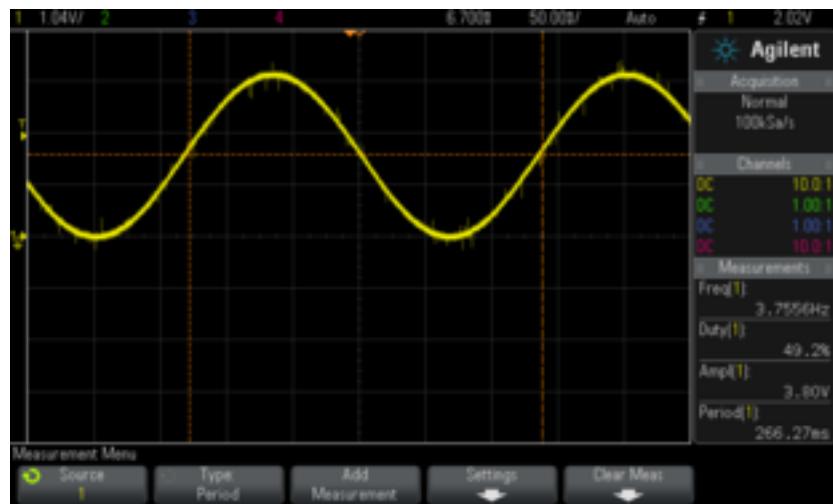


Figure 14: DAC con Raspberry Pi

Instrumentación de Laboratorio

MediaCenter

Robots

Supercomputación

Minería de BitCoin

Cámaras

Juegos



Figure 15: Placa para conectar Lego y Raspberry Pi

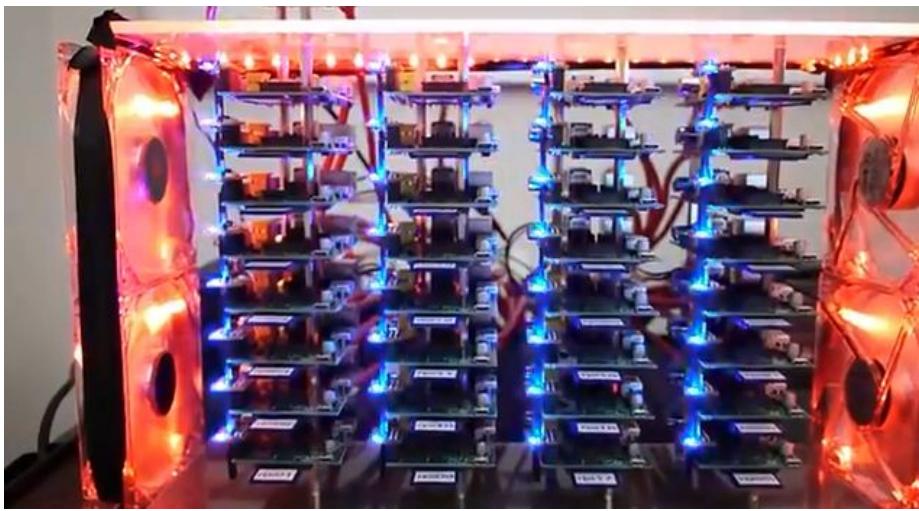


Figure 16: Cluster formado por 64 Raspberry Pi



Figure 17: Placa dedicada a minería de Bitcoin basada en Raspberry Pi



Figure 18: snapPiCam: cámara basada en Raspberry Pi



Figure 19: El clásico Doom corriendo en una Raspberry Pi



Figure 20: minecraft



Figure 21: Láser que se mueve aleatoriamente



Figure 22: ligth painting



Figure 23: Sistema de iluminación



Figure 24: Kano

**Minecraft**

**Instrumentos de tortura**

**Arte**

**Iluminación**

**Kano: portátil basado en raspberrypi**

**Raspberry en los medios**

Han sido muchas las películas y series donde han aparecido últimamente Raspberry Pi

- Serie Mr. Robot

Vídeo de Mr. Robot

- CSI Cyber (video)

## ¿Cómo lo usan?

- Mr. Robot: utilizan como Datalogger y para inyectar datos erróneos en el sistema de temperatura para
- CSI: Bridge para capturar datos

## Iniciativas similares

Existen proyectos similares en su origen y que han tenido gran difusión

### Micro micro:Bit

Está previsto que llegue a todos los escolares británicos de 11 y 12 años.



Figure 25: microbit

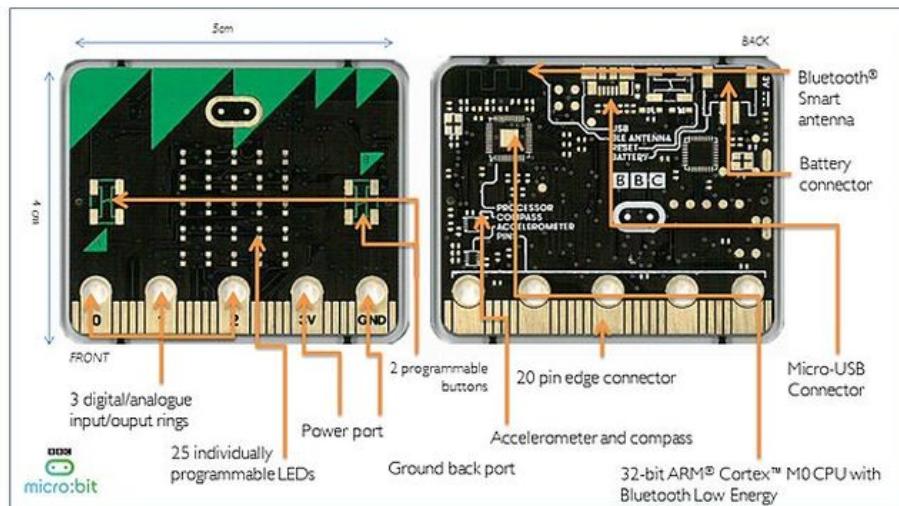


Figure 26: microbit

- Es similar a arduino
- Enseña a programar
- Enseña electrónica
- Ampliable

Vídeo

### One Laptop Per Child (OLPC)

Un ordenador superbarato y que se puede utilizar en condiciones complejas: sin electricidad (tiene un generador de manivela) y capaz de crear su propia red de comunicaciones



Figure 27: olpc

### Arduino

¿Qué decir de arduino?

### ¿Raspberry Pi vs Arduino?

- ¿Son incompatibles?
- ¿Para qué sirve cada uno?
- Arduino es más robusto
- Raspberry es más inteligente



Figure 28: aulaOLPC



Figure 29: arduino



Figure 30: vs

	<b>Arduino Uno</b>	<b>Raspberry Pi Model B</b>
<b>Price</b>	\$30	\$35
<b>Size</b>	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
<b>Memory</b>	0.002MB	512MB
<b>Clock Speed</b>	16 MHz	700 MHz
<b>On Board Network</b>	None	10/100 wired Ethernet RJ45
<b>Multitasking</b>	No	Yes
<b>Input voltage</b>	7 to 12 V	5 V
<b>Flash</b>	32KB	SD Card (2 to 16G)
<b>USB</b>	One, input only	Two, peripherals OK
<b>Operating System</b>	None	Linux distributions
<b>Integrated Development Environment</b>	Arduino	Scratch, IDLE, anything with Linux support

Figure 31: vs

# **OpenSource y licencias abiertas**

## **OpenSource**

- ¿qué es?
- ¿sólo código?
- ¿para qué me puede servir?
- ¿proyecto pequeños?
- Linux
- Arduino

## **Licencias**

- ¿ Licencia abierta o Copyright?
- ¿ Por qué compartir?
- Licencias CC charla de psicobyte (OSL)

Para saber más sobre el tema:

Charla de Psicobyte (OSL)

# **Equipo**

## **Componentes Obligatorios**

- Fuente de alimentación con conector micro-USB con al menos 2A
- Tarjeta SD de 2 GB o mejor de 4Gb. Se recomienda de clase 10 por su velocidad
- Cable de red ethernet

Y si la vas a usar como un ordenador

- Cable HDMI. Existen conversores, pero no todos funcionan bien con cables largos
- Teclado y ratón USB. Mejor un combo que sólo gastará 1 USB
- Un monitor HDMI

## **Opcionales**

- Una caja o carcasa(para evitar problemas)
- Wifi USB
- Hub USB con alimentación, así podremos añadir más dispositivos y evitaremos cargar la potencia de la Raspberry

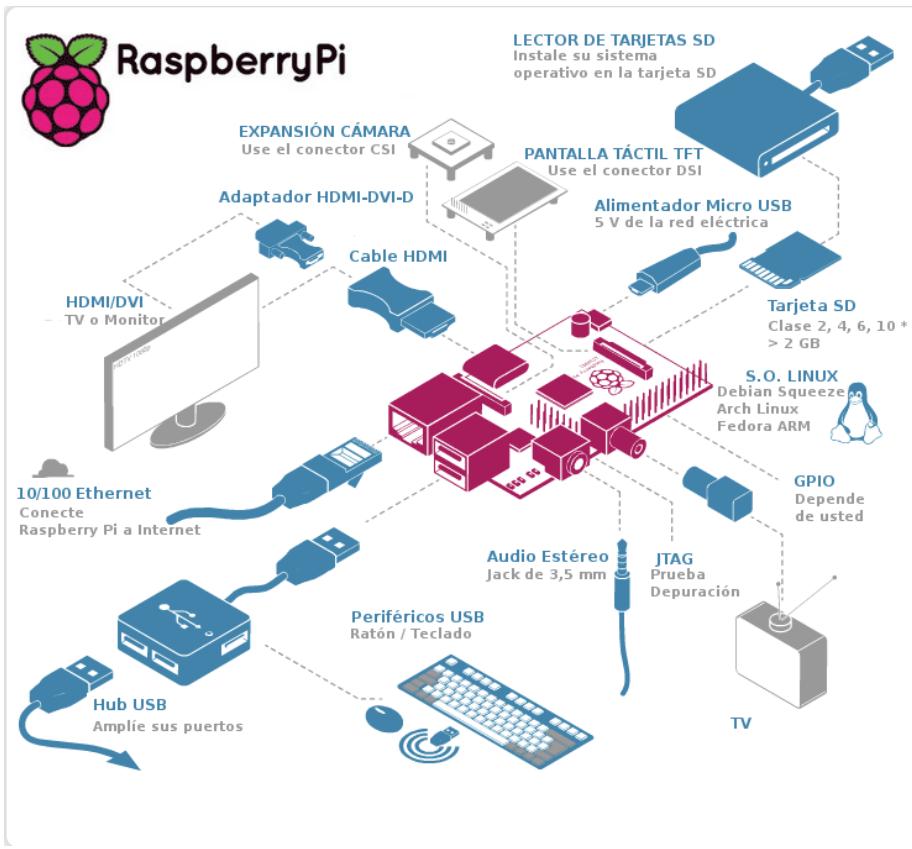


Figure 32: Esquema completo de montaje de una Raspberry Pi



Figure 33: Carcasa hecha con Lego

¿Más carcasas?

## Dónde y qué comprar

### Tiendas para comprar

- Raspberry.org
- www.electan.com
- www.amazon.es
- www.bricogeek.com
- www.raspipc.com ¿Algún sitio donde comprar en Granada?

### ¿Qué comprar?

Mejor un kit

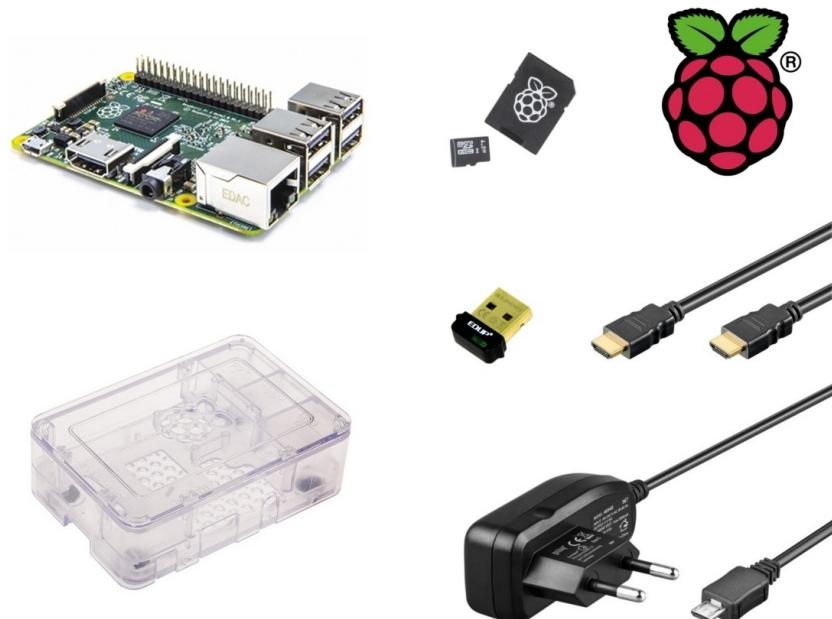


Figure 34: Kit Raspberry Pi

## Materiales

- Raspberry Pi
- Tarjeta SD de al menos 4Gb
- Alimentación de 5V y al menos 2A (mejor si son 2.5A)
- Caja para evitar cortocircuitos
- Monitor HDMI (o VGA con adaptador)
- Teclado y raton USB (mejor si es inalambrico, porque consume menos)
- Conexión a la red: Cable ethernet o dongle Wifi
- Hub USB alimentado externamente (para evitar cargar demasiado a la Raspberry)

## ¿Dónde encontrarlos?

- Kit base
- Shield electronica
- Kit amazon 1
- Kit Amazon 2 1
- Kit Amazon 3

## Arquitectura

Un diagrama simplificado de la arquitectura de Raspberry Pi

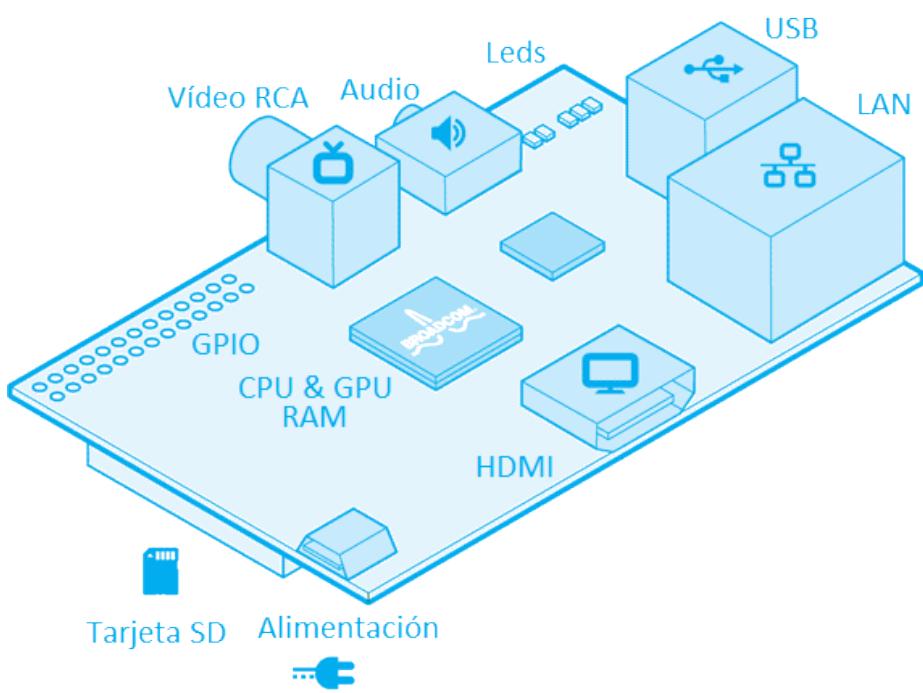


Figure 35: Arquitectura de la Raspberry Pi

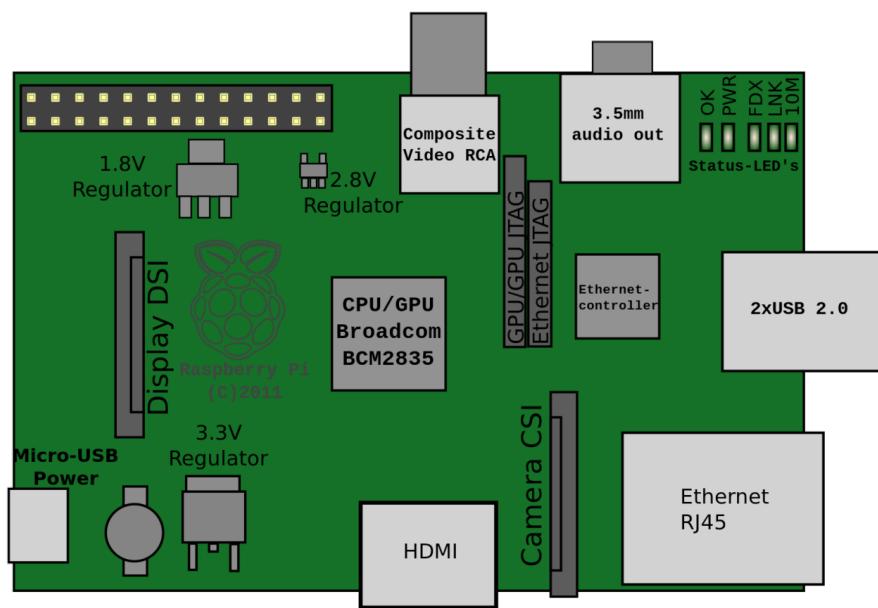


Figure 36: Bloques de la Raspberry Pi

## **Sistemas operativos disponibles**

**Noobs**

**Raspbian**

**Ubuntu Mate**

**Snappy Ubuntu Core**

**Windows 10 IOT Core**

**OSMC**

**Chrome OS**

**Android**

**(¿Dónde encontrarlos?)**

## **Instalación**

*¿Qué necesitamos?*

- Formatear tarjeta ([http://www.sdcard.org/downloads/formatter\\_4/](http://www.sdcard.org/downloads/formatter_4/))
- Descargamos la imagen del sistema que queramos <http://www.raspberrypi.org/download>
- ¿Qué imagen usar?
  - Empecemos con Noobs
  - Instalación de Noobs
- ¡¡¡Arrancar!!!
- Configuración

## **Configuración**

**sudo raspi-config**

(Puede variar algo según la versión)

Una vez configurado podemos abrir el entorno visual con

**startx**

En cualquier momento podemos volver a reconfigurar

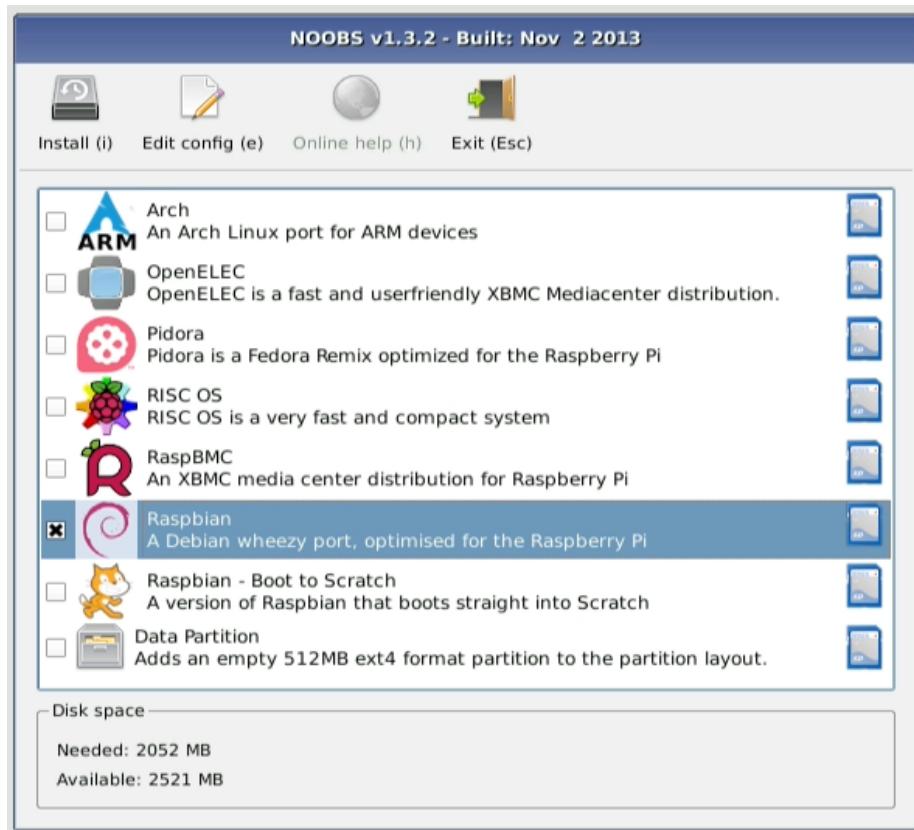


Figure 37: noobs

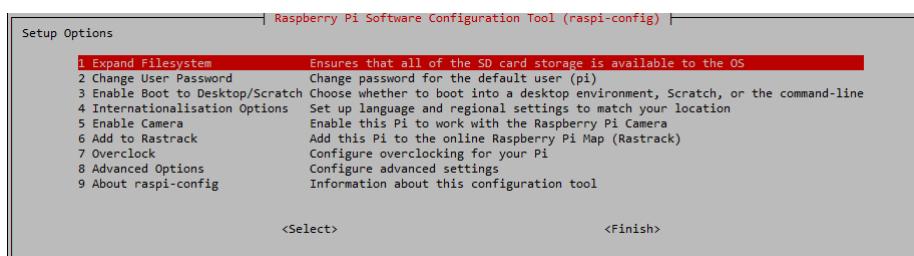


Figure 38: config

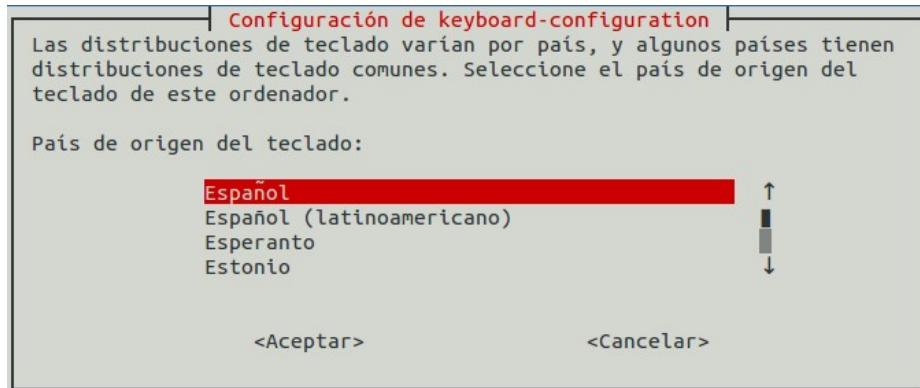


Figure 39: teclado

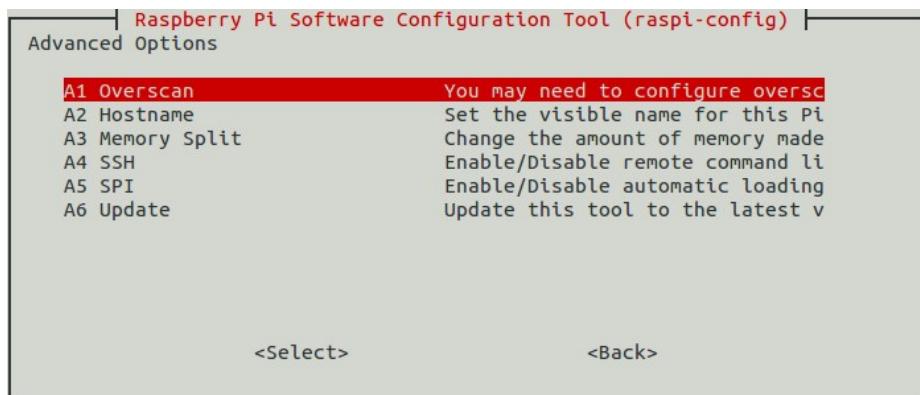


Figure 40: avanzado

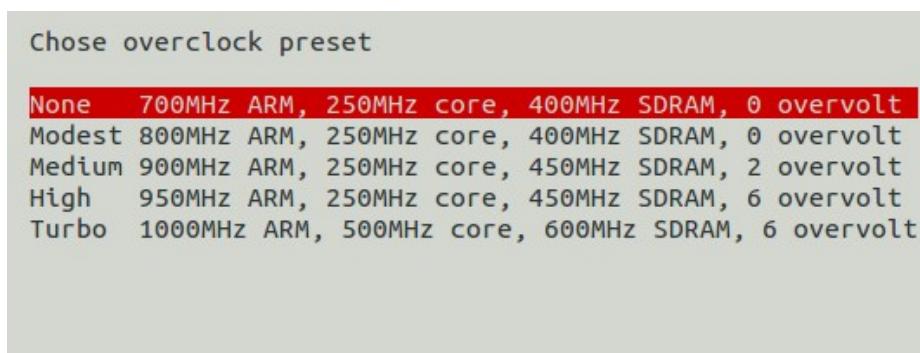


Figure 41: overclock

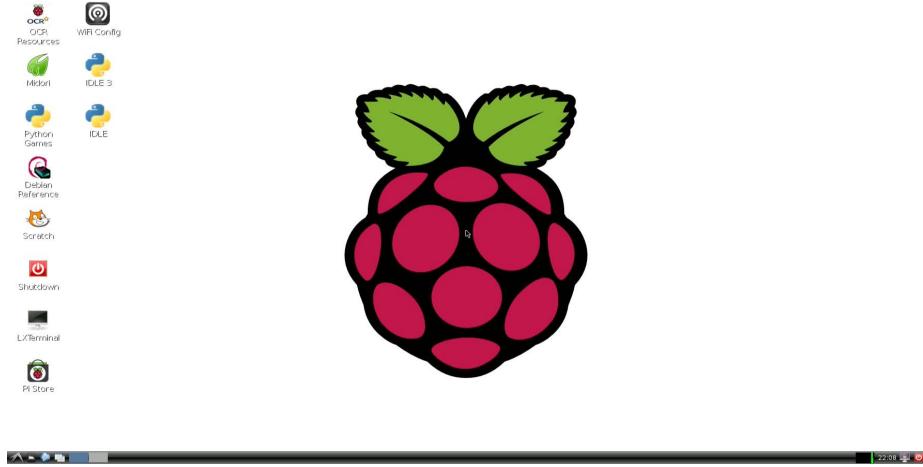


Figure 42: Arrancamos el entorno visual con startx

```
sudo raspi-config
```

## Problemas

Siempre podemos encontrarnos con problemas. Veamos los más frecuentes

### Alimentación

Necesitamos un mínimo de 2A

### Velocidad de la tarjeta

Se recomienda velocidad 10, una velocidad menor da problemas a bloqueos

### Espacio en disco

Al menos 2Gb por sistema operativo, mejor 4Gb

### No se ve nada en el monitor

¿Lo arrancaste con el monitor conectado? Es necesario arrancar con el monitor conectado

## Mantenimiento

Una vez instalado el sistema, necesitamos de vez en cuando actualizarlo.

### Actualización (update)

Des un terminal/consola

Para buscar cambios

`sudo apt-get update`

Para instalar estos cambios

`sudo apt-get upgrade`

Para actualizar el sistema

`sudo apt-get dist-upgrade`

Para instalar un paquete determinado

`sudo apt-get install paquete`

(siempre podemos instalar desde la herramienta visual “Añadir programas” en el menú Preferencias)

### Cuidados eléctricos

- No existe protección en los terminales, con lo que es muy, muy sencillo quemar la placa.
- Cuidado con colocar la placa sobre un instrumento o superficie metálica. Mejor usar una caja
- Cuidado con los dispositivos que conectamos, pudieran demandar más potencia de la que le puede dar

## Manejando tu Raspberry Pi

Como sabes es una máquina Linux, con lo que podrás manejarla igual que se maneja cualquier otra máquina Linux

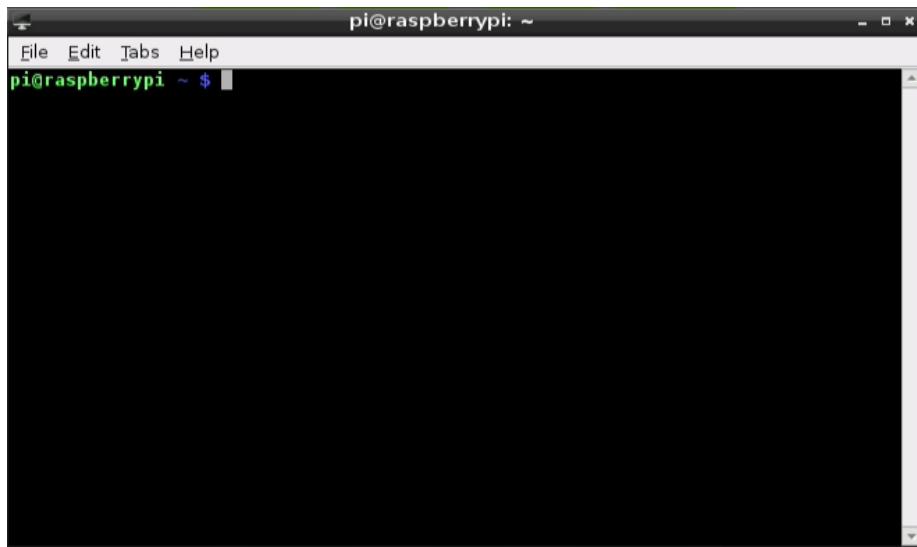


Figure 43: console

## Consola (línea de comandos)

### Comandos básicos:

- **ls** : muestra los archivos y directorios ( **ls -l** para más detalles y **ls -a** para mostrar todos)
- **cd** : cambia de directorio (**cd ~** nos lleva a nuestro directorio home y **cd ..** sale del directorio actual)
- **chmod** : cambia los permisos de un fichero/directorio (**chmod ugo-w fichero** quita todos los permisos de escritura)
- **pwd** : nos dice el directorio actual
- **mv** : mueve directorios/ficheros a un nuevo destino
- **rm** : borra directorios/ficheros
- **mkdir** : crea un directorio
- **passwd** : cambia la contraseña del usuario actual
- **ps -ef** : muestra los procesos en ejecución
- **top** : administrador de tareas
- **clear** : borra todo el contenido del terminal
- **df** : muestra el % de disco ocupado
- **nano** : editor de texto básico
- **vi** : editor de texto avanzado pero complejo
- **du** : muestra lo que ocupa un directorio (**du -s \*** muestra lo que ocupa un directorio y todo lo que contiene)
- **sudo halt** apaga la raspberry
- **sudo shutdown -h now** apaga la raspberry

- **history** : muestra todos los comandos que se han ejecutado antes. Podemos ejecutar el comando de la posición n, con !n . Las teclas abajo/arriba del cursor nos permiten iterar por los comandos usados.
- La tecla Tabulador nos permite completar el nombre del fichero/directorio
- **man comando**: Para obtener ayuda sobre comando
- Para hacer fichero script: añadimos los comandos, chmod u+x fichero y para ejecutarlo ./fichero

## Usuarios

El usuario por defecto es “**pi**” con contraseña “**raspberry**”

## Cuidado con sudo

Nos da todo el poder del usuario administrador (**root**)

## Interface gráfico

Para arrancar el interface gráfico (si no está arrancado) usaremos

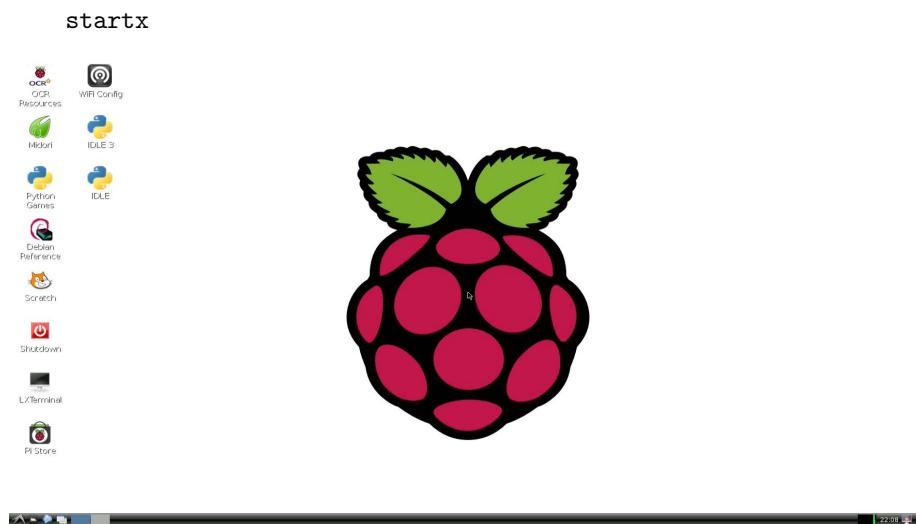


Figure 44: startx

## Acceso remoto

### Conexión directa

#### ssh

Es el protocolo de acceso por consola

Entramos en la configuración avanzada

```
sudo raspi-config
```

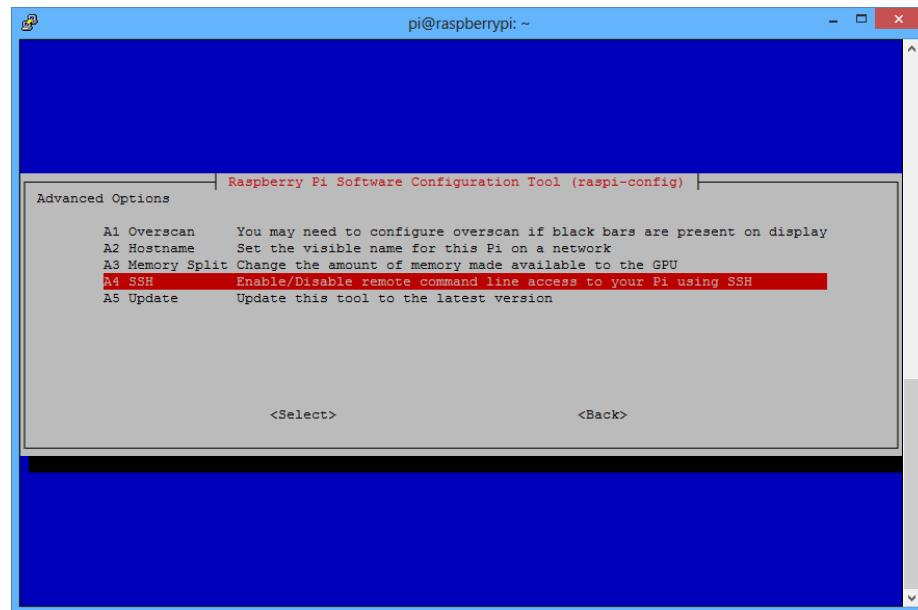


Figure 45: ssh

Podemos hacerlo también por comandos con

```
sudo service ssh start  
sudo insserv ssh
```

Ahora podremos conectarnos remotamente con ssh

```
ssh pi@192.189.0.123
```

O bien usando algún software como Putty

Conviene cambiar la contraseña para evitar que cualquiera pueda acceder

## vnc

VNC es un protocolo que nos permite acceder remotamente al escritorio de otra máquina.

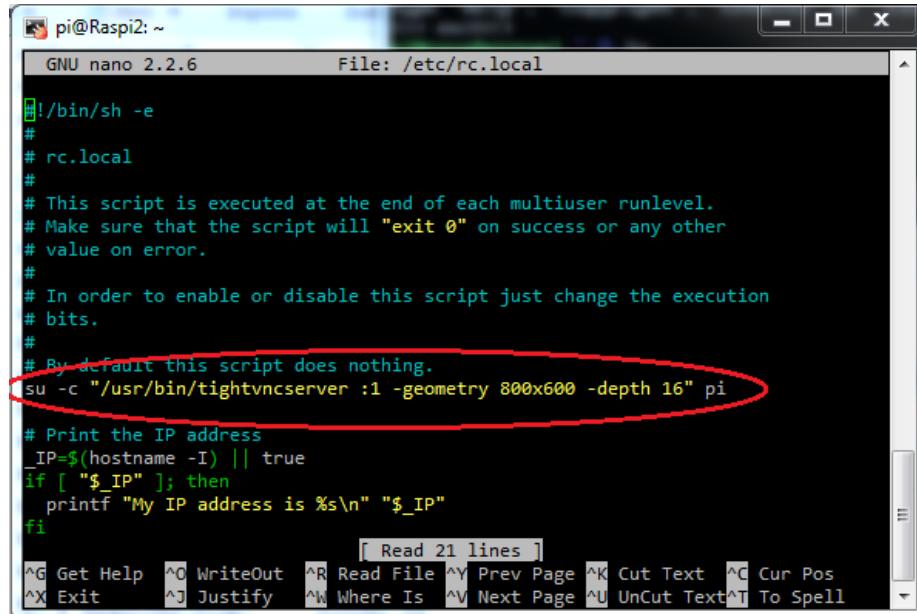
En las nuevas versiones de Raspbian podemos activar VNC desde la configuración (raspi-config).

Si no está disponible podemos instalarlo en nuestra Raspberry de manera sencilla con:

```
sudo apt-get install tightvncserver
```

Este software requiere que un servicio se ejecute al arrancar si queremos acceder en cualquier momento. Podemos instalarlo añadiendo la siguiente línea al archivo `/etc/rc.local`

```
su -c "/usr/bin/tightvncserver :1 -geometry 800x600 -depth 16" pi
```



```
pi@Raspi2: ~
GNU nano 2.2.6          File: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
su -c "/usr/bin/tightvncserver :1 -geometry 800x600 -depth 16" pi

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi
[ Read 21 lines ]
```

Figure 46: vnc

Ahora accederemos usando un cliente vnc

## Acceso directo

Vamos a configurar nuestra raspberry y un portátil con Ubuntu para facilitar al máximo la conexión y así no tener que utilizar muchos componentes. De

esta manera podremos trastear con un kit mínimo, evitando tener que usar un teclado, ratón y sobre todo un monitor.



Figure 47: Conexión directa entre Raspberry y Portatil

En concreto usaremos simplemente un cable de red (ethernet) y un cable micro-usb para alimentar la raspberry.

Con esta configuración no podemos consumir en total más de los 500mA que proporciona el USB.

Tendremos que modificar ficheros de configuración en el PC y en la raspberry.

Asumiremos que tenemos conexión a internet via Wifi y utilizaremos el cable ethernet para dar conectividad a la raspberry. Crearemos una red entre el portátil y la raspberry creando una subred distinta y haremos que el portátil actúe como gateway de esa red enrutando los paquetes hacia la raspberry y dándole acceso a internet.

Comencemos editando la configuración del pc, para lo que ejecutaremos en el pc:

```
sudo vi /etc/network/interfaces
```

y dejamos el contenido del fichero (la red que se usa normalmente es las 192.168.1.x de ahí que el gateway sea 192.168.1.1 que es el real)

Ahora vamos a editar la configuración de la raspberry. La forma más sencilla es editando los ficheros de configuración desde el pc, para lo que insertamos la tarjeta sd de la raspberry (obviamente con esta apagada) en el pc y ejecutamos en este:

```
sudo vi /media/10b4c001-2137-4418-b29e-57b7d15a6cbc/etc/network/interfaces
```

Quedando el mismo:

Ahora, colocamos la tarjeta sd en la raspberry y volvemos a encenderla

Conectamos el cable ethernet entre los dos

```
auto lo
iface lo inet loopback
iface eth0 inet static
    address 192.168.0.80
    netmask 255.255.255.0
    gateway 192.168.1.1
```

Figure 48: Configuración inicial de la red local

```
auto lo

iface lo inet loopback
#iface eth0 inet dhcp
iface eth0 inet static
    address 192.168.0.90
    netmaks 255.255.255.0
    gateway 192.168.0.80
```

Figure 49: Configuración final de la red local

En el PC hacemos comprobamos que la tarjeta eth0 está ok y con la ip correspondiente, haciendo

```
ifconfig /all
```

Veremos que aparece el interface eth0 con ip 192.168.0.80

Ahora vamos a hacer que el portátil actúe como router. Para ello ejecutamos los siguientes comandos

```
sudo su -
```

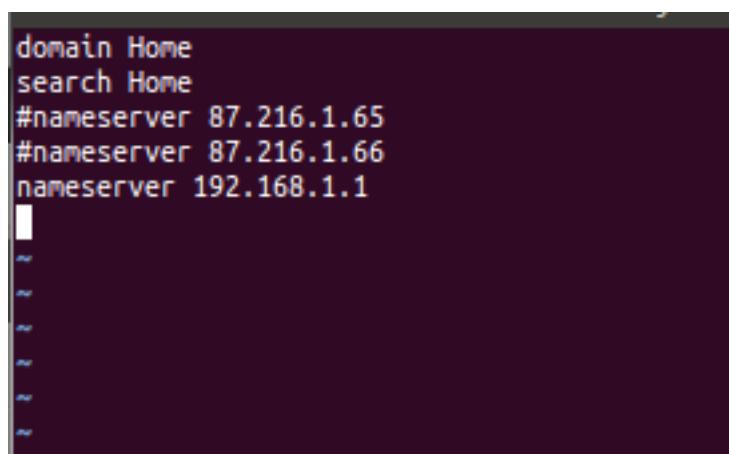
```
root@ubuntu-asus:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
root@ubuntu-asus:~# /sbin/iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

Por último editamos el fichero de configuración de DNS con

```
sudo vi /etc/resolv.conf
```

y lo dejamos así



```
domain Home
search Home
#nameserver 87.216.1.65
#nameserver 87.216.1.66
nameserver 192.168.1.1
```

Figure 50: Configuración de servidor de nombres

Ahora solo falta probar que tenemos conectividad, haciendo un ping

```
ping 192.168.0.90
```

Si todo es correcto ya podremos acceder via ssh o VNC

## Usos

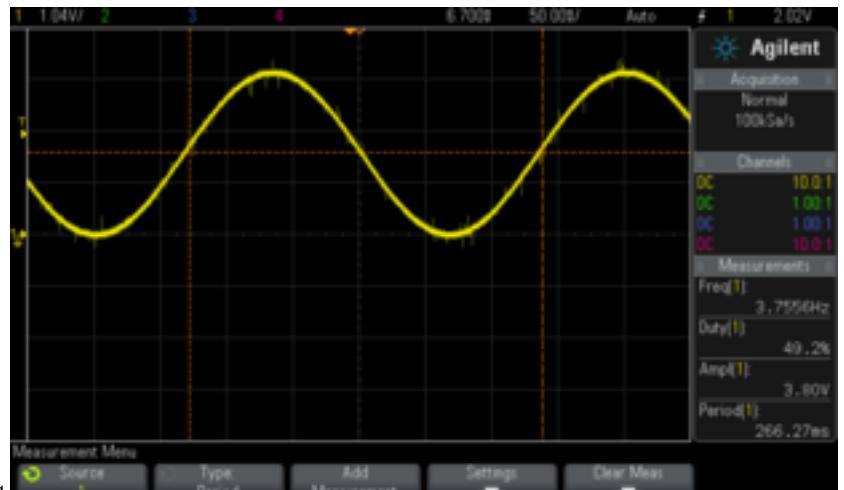
### MediaCenter

How to make a MediaCenter



Figure 51: home

## Laboratorio



- Generador de onda DAC
- Analizador de frecuencias

## Cerebro de una instalación domótica

Regulación de temperatura

**Servidor NAS**

**Cortafuegos/acceso remoto**

**Monitorización**

**Sistema aislado recogida de datos**

**Streaming**

- Radio wifi

**Hacking**

**Vigilancia**

**Enseñanza**

Distribución educativa

Aulas informatica

Servidor de aula (Kahn Academy Offline)

Raspeberry 35\$

SD 64Gb 50\$

Wifi USB 5\$

Caja 9\$

**Robótica**

Raspberry y Lego Minstorm

**Data crunching**

- Cluster de supercomputación
- Minería de BitCoin

**Cámara**

SnapPiCam



Figure 52: aula



Figure 53: Robot Lego controlado por Raspberry Pi

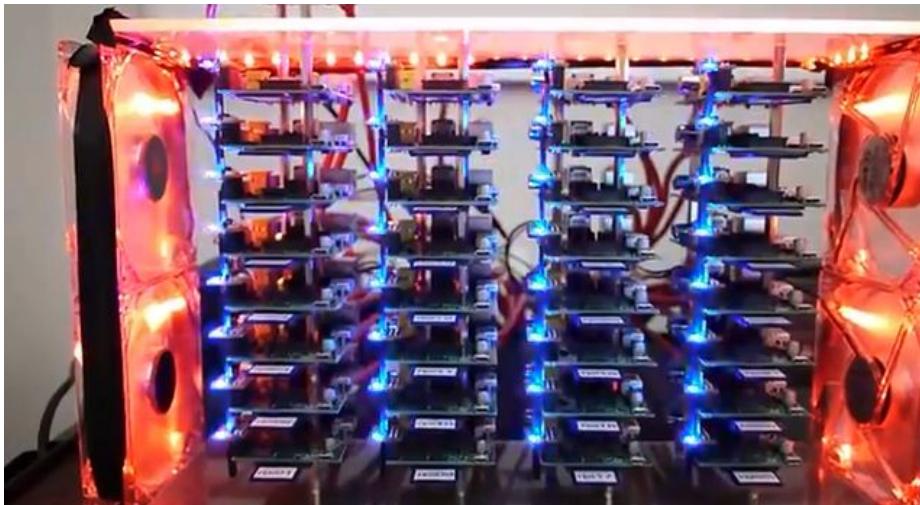


Figure 54: cluster



Figure 55: bitcoin



Figure 56: snapPiCam

### Máquina de juegos

- Mame
- Mini máquina recreativa
- Cómo instalar y jugar Doom
- Minecraft

### Instrumentos de tortura

- Juguete para tu gato
- Cuadro diabólico

### Arte

- Ligth Painting
- Iluminación
- Cortina luminosa



Figure 57: Doom



Figure 58: minecraft



Figure 59: Laser para jugar con gatos



Figure 60: cuadro diabólico



Figure 61: lighth painting



Figure 62: ilumnicación



Figure 63: cortina

## Portables

- Raspberry Pi Gameboy
- Super consola



Figure 64: super

## Teléfono

- RaspiPhone

## Coche

Datalogger de datos del coche

Seguidor de flotas

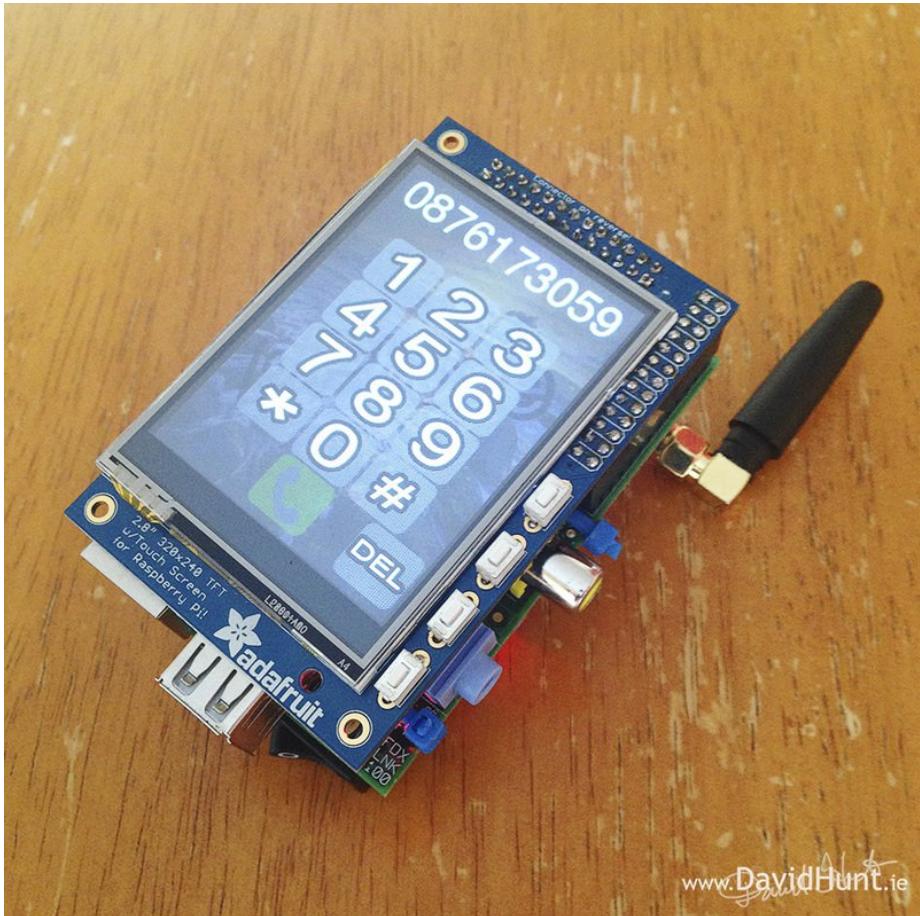


Figure 65: phone

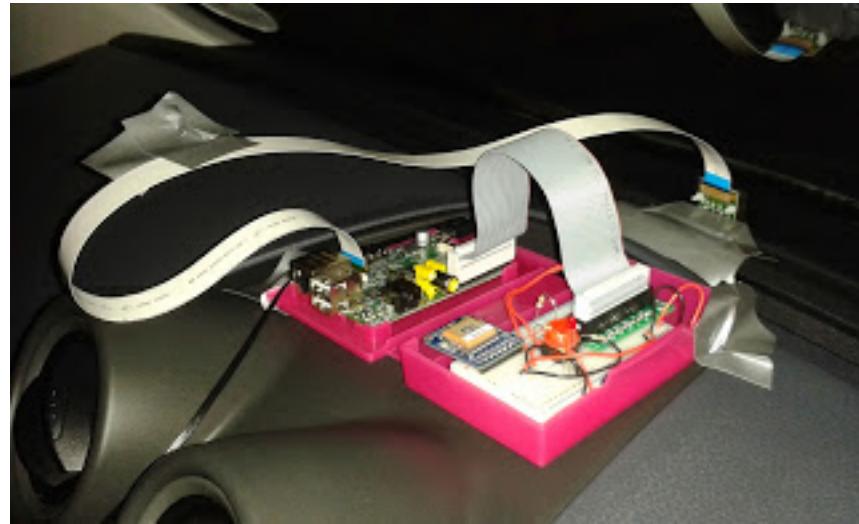


Figure 66: coche

## Hacking

Wifi sniffer(RP para agentes secretos)(wifi . . . )

Intercambio de contenidos

Cómo hackear una raspberry pi

Hacking RaspPi

## Usos hackers de raspberry

Escóndela dentro de una regleta Turn a Raspberry Pi Into a Super Cheap, Packet-Sniffing Power Strip

## Ejemplos de cine

Mr Robot

## Vigilancia

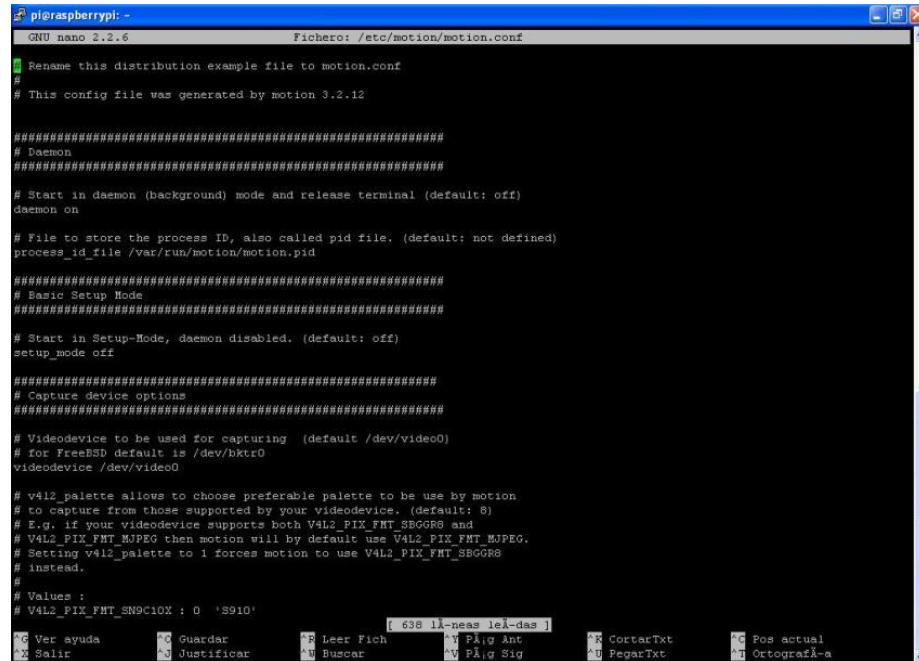
Podemos usar su cámara (la original o una USB)

Usaremos un software standard de Linux: motion

```
sudo apt-get install motion
```

Editamos la configuración

```
sudo nano /etc/motion/motion.conf
```



The screenshot shows a terminal window titled 'pi@raspberrypi: ~' with the command 'GNU nano 2.2.6' at the top. The file being edited is '/etc/motion/motion.conf'. The content of the file is a configuration for the motion daemon, including sections for daemon, basic setup mode, capture device options, and video device settings. The terminal window has a dark background with white text and includes a status bar at the bottom with various keyboard shortcuts.

```
pi@raspberrypi: ~
GNU nano 2.2.6          Fichero: /etc/motion/motion.conf

# Rename this distribution example file to motion.conf
#
# This config file was generated by motion 3.2.12

#####
# Daemon
#####

# Start in daemon (background) mode and release terminal (default: off)
daemon on

# File to store the process ID, also called pid file. (default: not defined)
process_id_file /var/run/motion/motion.pid

#####
# Basic Setup Mode
#####

# Start in Setup-Mode, daemon disabled. (default: off)
setup_mode off

#####
# Capture device options
#####

# Videodevice to be used for capturing (default /dev/video0)
# for FreeBSD default is /dev/bktr0
videodevice /dev/video0

# v4l2 palette allows to choose preferable palette to be used by motion
# to capture from those supported by your videodevice. (default: 8)
# E.g. if your videodevice supports both V4L2_PIX_FMT_SBGGR8 and
# V4L2_PIX_FMT_MJPEG then motion will by default use V4L2_PIX_FMT_MJPEG.
# Setting v4l2_palette to 1 forces motion to use V4L2_PIX_FMT_SBGGR8
# instead.
#
# Values :
# V4L2_PIX_FMT_SN9C10X : 0 'S910'
[ 638 líneas de más ]

AC Ver ayuda      GU Guardar      LF Leer Fich     PG Ant     CT CortarTxt     PA Pos actual
  S Salir          J Justificar    BU Buscar      PG Sig     PG PegarTxt     OA Ortografía
```

Figure 67: usando motion

Lo arrancamos

```
sudo motion -n
```

Podremos acceder a la imagen en vivo de la cámara con

```
http://raspberry_ip:8001
```

## Simuladores

Varios simuladores

## En windows

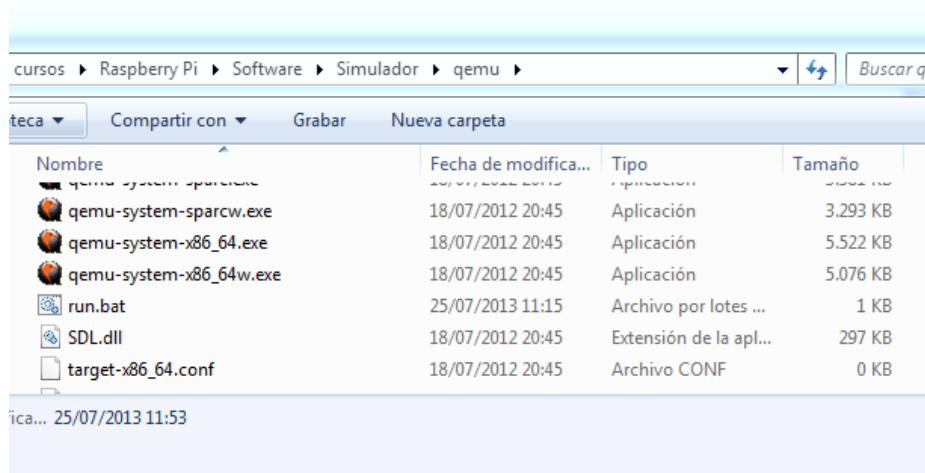


Figure 68: qemu

- Descargamos la imagen en <http://sourceforge.net/projects/rpiqemuwindows/>
  - Emulador qemu
  - Imagen (2012-07-15-wheezy-raspbian.img) o (<http://downloads.raspberrypi.org/download.php?file=/image/2012-07-15-wheezy-raspbian/2013-05-25-wheezy-raspbian.zip>)
- Ejecutamos

```
qemu-system-arm.exe -M versatilepb -cpu arm1176 -hda imagen/2013-09-25-wheezy-raspbian.qcow2
```

Vídeo

## En ubuntu

### Simulando en virtualBox

#### Simulando el Sense Hat

Sobre la velocidad....

## Exteriores

Su bajo peso, y los pocos periféricos de los que depende, facilitan su uso en entornos aislados

- Aislamiento
  - Temperatura
  - Baterias
  - ¿Espacio?
- astroPi



Figure 69: espacio

- ¿Volando? Autopilot

## Proyectos

- Portable Raspberry
- MiniPortatil

Más detalles en el libro “RP para agentes secretos”

## Aprendiendo a programar

Existen distintas alternativas, dependiendo de los conocimientos previos



Figure 70: espa



Figure 71: autopilot



Figure 72: portable



Figure 73: MiniPortatil con Raspberry Pi

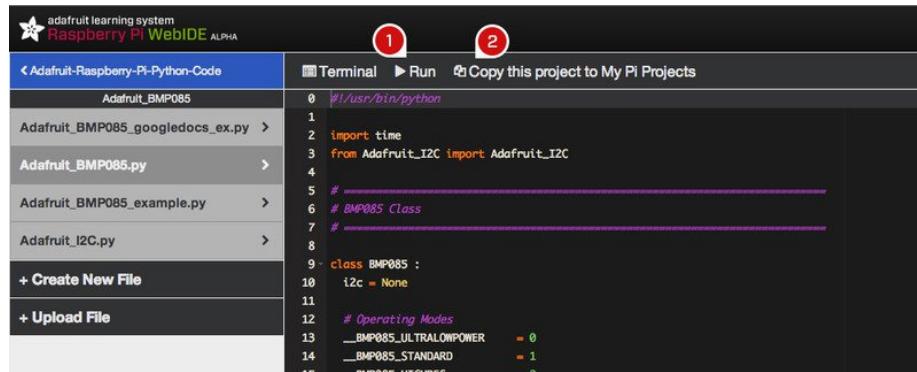
## Scratch: pensado para gente sin conocimiento

python: profesional, se necesitan otras habilidades (manejo de editores,...)

Otros: permiten aprender de una forma sencilla otros lenguajes.

Facilitan la instalación de los entornos, que suele ser lo más complicado

- Coder Instala un servidor con todo lo necesario para montar un aula de aprendizaje de programación web
  - Tiene varios proyectos de diferente complejidad, con tutoriales
- WebIDE Entorno Web que permite programar en Python, Ruby, Javascript y otros lenguajes



The screenshot shows the Adafruit Raspberry Pi WebIDE interface. At the top, there's a header with the Adafruit learning system logo, 'Raspberry Pi WebIDE ALPHA', and tabs for 'Terminal', 'Run', and 'Copy this project to My Pi Projects'. Below the header, the left sidebar shows a file tree with files like 'Adafruit\_BMP085', 'Adafruit\_BMP085\_googledocs\_ex.py', 'Adafruit\_BMP085.py', 'Adafruit\_BMP085\_example.py', and 'Adafruit\_I2C.py'. There are also '+ Create New File' and '+ Upload File' buttons. The right side is a terminal window with the following Python code:

```
#!/usr/bin/python
import time
from Adafruit_I2C import Adafruit_I2C
# -----
# BMP085 Class
# -----
class BMP085 :
    i2c = None
    # Operating Modes
    __BMP085_ULTRALOWPOWER = 0
    __BMP085_STANDARD = 1
    __BMP085_HIGHRES = 2
```

Figure 74: webide

## Scratch

Podemos probarlo online

Está pensado para enseñar a programar sin la complejidad de la sintaxis.

Otros similares

- Snap!
- Code.org

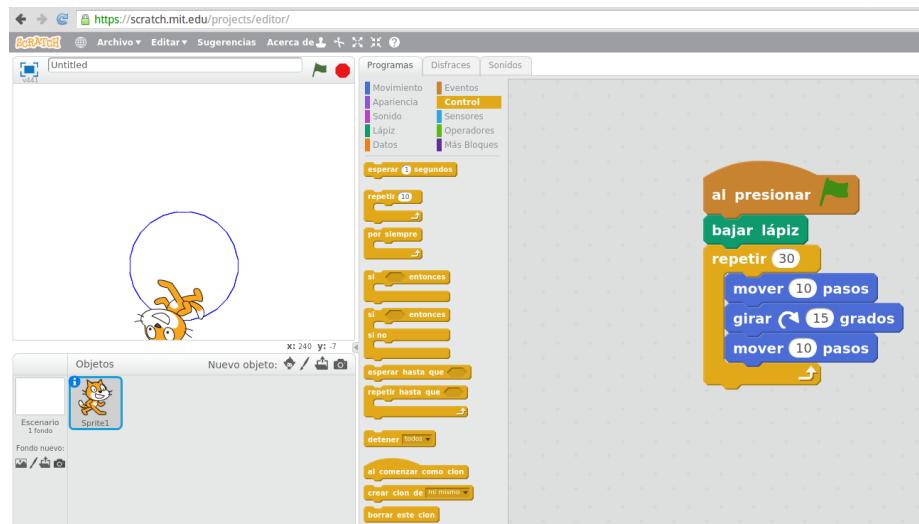


Figure 75: scratch



Figure 76: code

## Utilizando scripts

Los scripts son ficheros donde colocamos distintas órdenes que se irán realizando de forma consecutiva una tras otra

### Usando la cámara

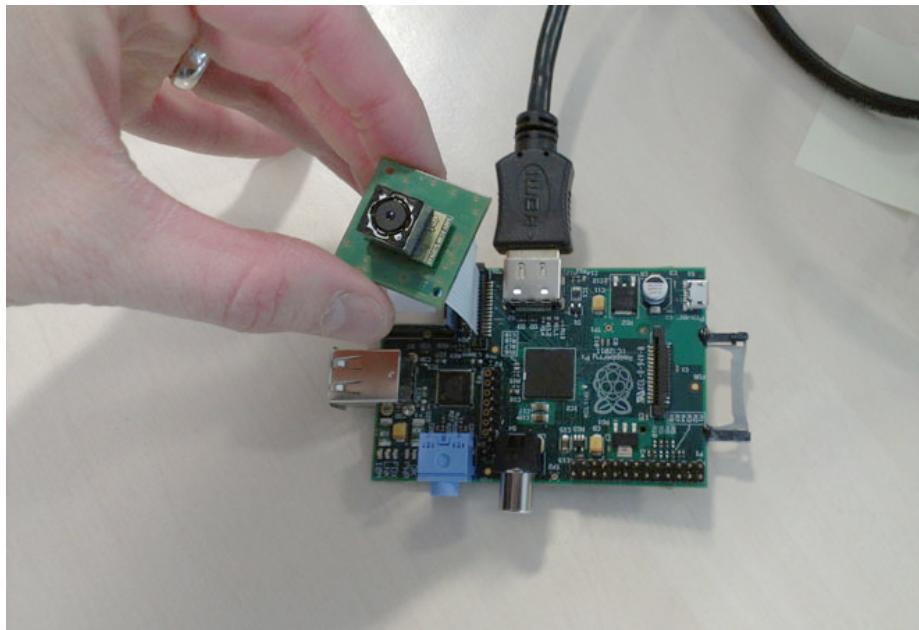


Figure 77: Camara de Raspberry Pi

La cámara tiene su propio conector, junto a las conectores GPIO.

Antes de poder utilizarla tenemos que activarla

```
sudo raspi-config
```

Tenemos 2 aplicaciones para usar la cámara

```
raspistill
```

Tomará imágenes fijas

```
raspivid
```

Grabará un vídeo

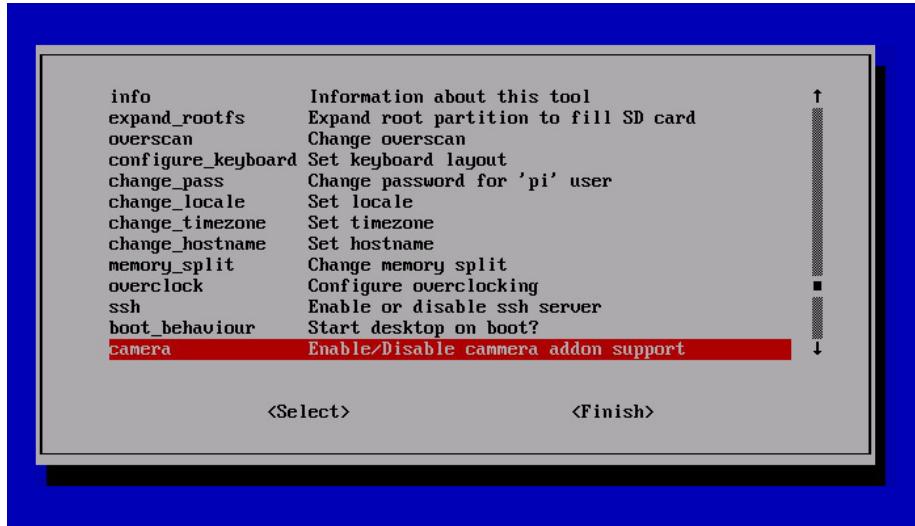


Figure 78: Configuración para activar la cámara

## Imágenes estáticas

Si queremos cambiar el retardo con el se captura, usamos la opción -t indicando el tiempo en milisegundos:

```
raspistill -o myimage.jpg -t 3000
```

Este programa tiene muchas opciones que podemos ver:

```

raspistill | less
-?, --help
: This help information
-w, --width
: Set image width <size>
-h, --height : Set image height <size>
-q, --quality : Set jpeg quality <0 to 100>
-o, --output : Output filename <filename>
-v, --verbose : Output verbose information during run
-t, --timeout : Time (in ms) before taking picture
(if not specified, set to 5s)
-th, --thumb
: Set thumbnail parameters (x:y:quality)
-d, --demo
: Run a demo mode
-e, --encoding : Output format (jpg, bmp, gif, png)
-tl, --timelapse : Timelapse mode. Takes a picture every <t>ms
-p, --preview : Preview window settings <'x,y,w,h'>

```

```

-f, --fullscreen : Fullscreen preview mode
-n, --nopreview : Do not display a preview window
-sh, --sharpness : Set image sharpness (-100 to 100)
-co, --contrast : Set image contrast (-100 to 100)
-br, --brightness : Set image brightness (0 to 100)
-sa, --saturation : Set image saturation (-100 to 100)
-ISO, --ISO
: Set capture ISO
-vs, --vstab
: Turn on video stablisation
-rot, --rotation : Set image rotation (90,180,270)
-hf, --hflip
: Set horizontal flip
-vf, --vflip
: Set vertical flip

```

Entre estas opciones podemos encontrar **-tl** que nos va a permitir tomar una imagen cada cierto tiempo. Con ello podemos generar una secuencia de imágenes con una sola línea de comando

```
raspistill -o myimage_%d.jpg -tl 2000 -t 25000
```

Una imagen cada 2 segundos durante 25 segundos Cada foto tendrá un número de secuencia

```

myimage_1.jpg
myimage_2.jpg
myimage_3.jpg
myimage_4.jpg
...

```

Si deseamos utilizar un formato de nombre más complejo, siempre podemos usar un script como el siguiente que además guardará las imágenes en una carpeta

```

SAVEDIR=/var/tlcam/stills
while [ true ]; do
filename=$(date -u +"%d%m%Y_%H%M-%S").jpg
/opt/vc/bin/raspistill -o $SAVEDIR/$filename
sleep 4;
done;

```

## Vídeo

raspivid nos va a permitir grabar vídeos. Para capturar 5s de vídeo en formato h264 utilizaremos:

```
raspivid -o video.h264
```

Si queremos capturar 10 segundos usaremos:

```
raspivid -o video.h264 -t 10000
```

Para ver todas las opciones disponibles podemos hacer

```
$raspivid | less
```

Para una documentación más detallada sobre las opciones del ejecutable se puede consultar el siguiente enlace

## Cámaras web

Podemos usar cámaras USB compatibles como la PS3 Eye.

Veremos si se ha detectado con:

```
$ ls -l /dev/video*
```

Si se detecta

```
pi@raspberrypi ~ $ ls -l /dev/video*
crw-rw---T 1 root video 81, 0 Jan 1 1970 /dev/video0
```

Figure 79: Camara USB detectada

Instalamos fswebcam

```
sudo apt-get install fswebcam
```

Que nos permitirá tomar una imagen con

```
fswebcam -d /dev/video0 -r 640x480 test.jpeg
```

Hagamos ahora un script para hacer un timelapse

```
#!/bin/bash
# Timelapse controller for USB webcam
DIR=/home/pi/timelapse
x=1
while [ $x -le 1440 ]; do
    filename=$(date -u +"%d%m%Y_%H%M-%S").jpg
    fswebcam -d /dev/video0 -r 640x480 $DIR/$filename
    x=$(( $x + 1 ))
    sleep 10;
done;
```

Podemos ver que se están realizando capturas de imágenes cada 10 segundos y como mucho se guardarán 1440 imágenes.

```
./runtimelapse
```

## Control remoto de camaras



Figure 80: Controlando una cámara profesional

También podemos controlar cámaras profesionales que suelen admitir conexión USB (como por ejemplo una Canon Rebel T4i / 650D)

Utilizaremos el software gphoto2 que instalaremos con

```
sudo apt-get install gphoto2
```

Podemos controlar casi todos los valores de exposición, ISO, etc de nuestra cámara remotamente, pero para no complicarnos vamos a suponer que la usamos en modo automático.

Podemos capturar una imagen, que se mantendrá en la cámara con:

```
$ gphoto2 --capture-image
```

Para tomar una imagen y enviarla a la raspberry usaremos

```
$ gphoto2 --capture-image-and-download
```

La librería gphoto2 por defecto guarda las imágenes en la memoria RAM de la Raspberry (no en la SD) con lo que es necesario que lo configuremos para evitar perderlas al cortar la alimentación.

```
$ gphoto2 --get-config /main/settings/capturetarget
```

Para establecer nuestro almacenamiento usaremos:

```
$ gphoto2 --set-config /main/settings/capturetarget=NuestroDirectorio
```

Veamos ahora como hacer un time-lapse, es decir capturar las imágenes cada cierto tiempo. Usaremos el siguiente comando.

```
$ gphoto2 --capture-image -F 1440 -I 30
```

Que almacenará en la cámara un máximo de 1440 imágenes tomadas cada 30 segundos

## Convertir fotos a vídeo

Una vez tengamos todas las imágenes podemos generar un vídeo con ellas.

Instalamos un software llamado mencoder que será el que genere el vídeo.

```
$ sudo apt-get install mencoder
```

Ahora generamos un fichero que contenga todas las imágenes que queremos unir en el vídeo

```
$ cd timelapse  
$ ls *.jpg > list.txt
```

Y ejecutamos memcoder con los parámetros adecuados (es una sola línea)

```
$ mencoder -nosound -ovc lavc -lavcopts vcodec=mpeg4:aspect=16/9:vbitrate=8000000 -vf so
```

Con esto generaremos un vídeo de 640x480 de resolución, con nombre timelapse.avi codificado en mpeg4, a 24 frame por segundo y con las imágenes cuyos nombres se incluyen en el fichero list.txt

Si queremos hacer un vídeo a partir de las imágenes tomadas con la cámara original de Raspberry usaremos el siguiente comando

```
$ mencoder -nosound -ovc lavc -lavcopts vcodec=mpeg4:aspect=16/9:vbitrate=8000000 -vf so
```

Hay que tener cuidado de no llenar el almacenamiento, puesto que este proceso consume mucho espacio

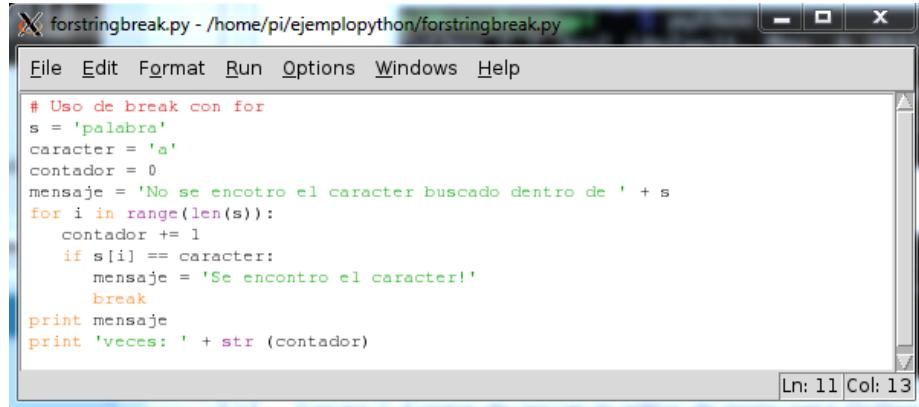
## Python

Es un lenguaje moderno de gran productividad, sencillo, potente y con millones de líneas ya desarrolladas que se pueden usar directamente por medio de paquetes instalables

Se utiliza en la web, en aplicaciones de escritorio, etc... Gran parte del interface de linux lo utiliza

Existen dos versiones de python ahora mismo: la rama 2.x y la 3.x Por sencillez vamos a usar la sintaxis de la rama 2.x

Podemos utilizar la herramienta Idle o python directamente para programar con él.



```
# Uso de break con for
s = 'palabra'
caracter = 'a'
contador = 0
mensaje = 'No se encontró el carácter buscado dentro de ' + s
for i in range(len(s)):
    contador += 1
    if s[i] == caracter:
        mensaje = 'Se encontró el carácter!'
        break
print mensaje
print 'veces: ' + str (contador)
```

Figure 81: Herramienta idle

Es más sencillo si escribimos nuestro código en un fichero (con cualquier editor de texto) y luego lo ejecutamos o bien abriéndolo con idle o haciendo:

```
python fichero.py
```

Veamos algunos ejemplos

## Operaciones numéricas y petición de datos al usuario

Código de Suma

```
# Programa que realiza la suma de dos valores
a=input('numero 1');
b=input('numero 2');
suma = a + b;
print (suma);
```

**Ejercicio:** cambia la operación a realizar

## Sentencias de control condicionales

Código de Bisiesto

```
# Programa que determina si un año es o no bisiesto
year = input('Introduzca el año: ');
if ((year%400)==0 or (year % 100) ==0 or (year%4)==0):
    print 'Es bisiesto!!';
```

```

else:
    print 'No es bisiesto!!!';

Código de días por mes

# Nos da los días que tiene el mes seleccionado
mes = input('Introduce el mes:');
year = input('Introduce el año:');
# Comprobamos si es entero
if type(mes) == int:
    # Comprobamos si esta entre 1 y 12
    if (mes>=1) and (mes<=12):
        if mes == 2:
            if(year%400 == 0) or (year%100 ==0) or (year %4 == 0):
                dias = 29;
            else:
                dias = 28;
        elif (mes==4) or (mes==6) or (mes==9) or (mes==11):
            dias = 30;
        else:
            dias = 31;
        print 'El mes '+str(mes) +' del año '+str(year)+ ' tiene '+str(dias)+ ' días';
    else:
        print 'El mes debe ser entre 1 y 12';
else:
    print 'El mes debe ser entero';

```

## Sentencias de control de repetición

Código de Buscando Caracteres

```

# Cuenta las veces que se repite un carácter en una palabra
word= 'palabra';
caracter = 'a';
contador=0;
mensaje='No se ha encontrado el carácter :('
for i in range(len(word)):
    if (word[i]==caracter):
        mensaje='se ha encontrado el carácter!!!!';
        contador=contador+1;

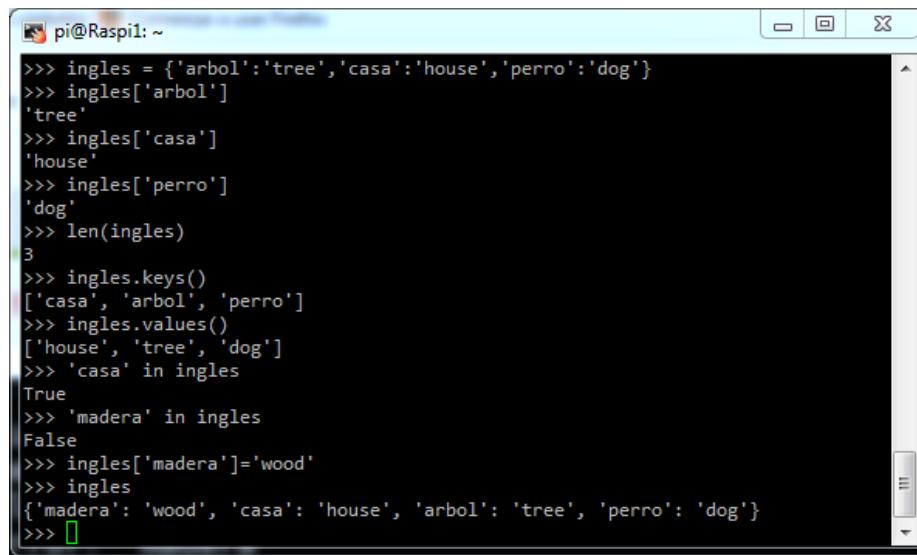
print mensaje;
print 'Se encontrado '+str(contador)+' veces';

```

**Ejercicio:** haz que el usuario pueda introducir la cadena donde buscar y el carácter

## Diccionarios que nos permitirán relacionar contenidos

Podemos introducir estas líneas en idle (las que empiezan por >>>)



```
pi@RaspPi1: ~
>>> ingles = {'arbol': 'tree', 'casa': 'house', 'perro': 'dog'}
>>> ingles['arbol']
'tree'
>>> ingles['casa']
'house'
>>> ingles['perro']
'dog'
>>> len(ingles)
3
>>> ingles.keys()
['casa', 'arbol', 'perro']
>>> ingles.values()
['house', 'tree', 'dog']
>>> 'casa' in ingles
True
>>> 'madera' in ingles
False
>>> ingles['madera'] = 'wood'
>>> ingles
{'madera': 'wood', 'casa': 'house', 'arbol': 'tree', 'perro': 'dog'}
>>> 
```

Figure 82: Usando Diccionarios

## Programa complejo

Veamos una implementación de un programa más elaborado como “Piedra, Papel o Tijera”

## Aplicaciones hechas en python

Utilizaremos la libre pyGTK que permite crear e interaccionar ventanas desde python

Tutorial de la OSL

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from gi.repository import Gtk

class MyWindow(Gtk.Window):

    def __init__(self):
        Gtk.Window.__init__(self, title="Hola Mundo!")
```

```

X piedrapapeltijera.py - /home/pi/ejemplospython/piedrapapeltijera.py
File Edit Format Run Options Windows Help
# Programa piedra papel o tijera
# Los numeros 0, 1 y 2 se corresponden con "piedra", "papel" o tijera de la forma:
# 0 - piedra
# 1 - papel
# 2 - tijera

# funciones de ayuda
import random

def number_to_name(number):
    # convertimos el numero al nombre
    if(number==0):
        return "piedra"
    elif (number==1):
        return "papel"
    elif (number==2):
        return "tijera"
    else:
        return "La elección no es correcta"

def name_to_number(name):
    # función para convertir el nombre en numero
    if(name=="piedra"):
        return 0
    elif (name == "papel"):
        return 1
    elif (name=="tijera"):
        return 2
    else:
        return -1

def pps(name):
    # convierte el nombre al numero usando name_to_number
    player_number=name_to_number(name);
    print "El jugador elige",number_to_name(player_number);
    # el ordenador elige una opción aleatoria con random.randrange()
    comp_guess=random.randrange(0,3);
    print "El ordenador elige",number_to_name(comp_guess);
    # diferencia entre player_number y comp_number
    outcome=player_number-comp_guess;
    if(outcome < 0):
        outcome=outcome+3;
    # usamos if/elif/else para determinar el ganador
    # restamos antes las dos opciones elegidas y hacemos un ajuste sumando 3 si negativa
    # Jugador  Ordenador  Ganador  Resta  Ajuste
    # 0          0        Empatan   0      0
    # 1          1        Empatan   0      0
    # 2          2        Empatan   0      0
    # 0          1        Ordenador -1      2
    # 0          2        Jugador   -2      1
    # 1          0        Jugador   1      1
    # 1          2        Ordenador -1      2
    # 2          0        Ordenador  2      2
    # 2          1        Jugador   1      1

    # muestra el resultado

    if(outcome==1):
        print "El jugador gana!\n"
    elif (outcome==2):
        print "El ordenador gana!\n"
    else:
        print "Jugador y ordenador empatan!\n"

    # Realizamos un test del programa con las tres opciones
    pps("piedra")
    pps("papel")
    pps("tijera")

```

Ln: 44 Col: 79

Figure 83: Ejemplo de piedra, papel o tijera

```

        self.button = Gtk.Button(label="Hazme click")
        self.button.connect("clicked", self.on_button_clicked)
        self.add(self.button)

    def on_button_clicked(self, widget):
        print("Hola Mundo!")

def main():
    win = MyWindow()
    win.connect("delete-event", Gtk.main_quit)
    win.show_all()
    Gtk.main()

main()

```

The image shows a terminal window and a GTK application window. The terminal window displays the following text:

```

seravb@Hercules ~/Dropbox/OSL/Cursos/2014/PythonAvanzado/con
ten
ido/curso-python-avanzado/Interfaces_gráficas_con_PyGTK/code
/01_Hola_Mundo (master) $ python 01_hola_mundo.py
Hola Mundo!
Hola Mundo!
Hola Mundo!

```

Below the terminal is a screenshot of a GTK application window titled "Hola Mundo!". The window contains a single button labeled "Hazme click".

Figure 84: gtk

Utilizaremos Glade para diseñar el interface

## PyGame

Si lo que queremos hacer es un juego podemos usar pyGame

Ejemplo sencillo

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-

```

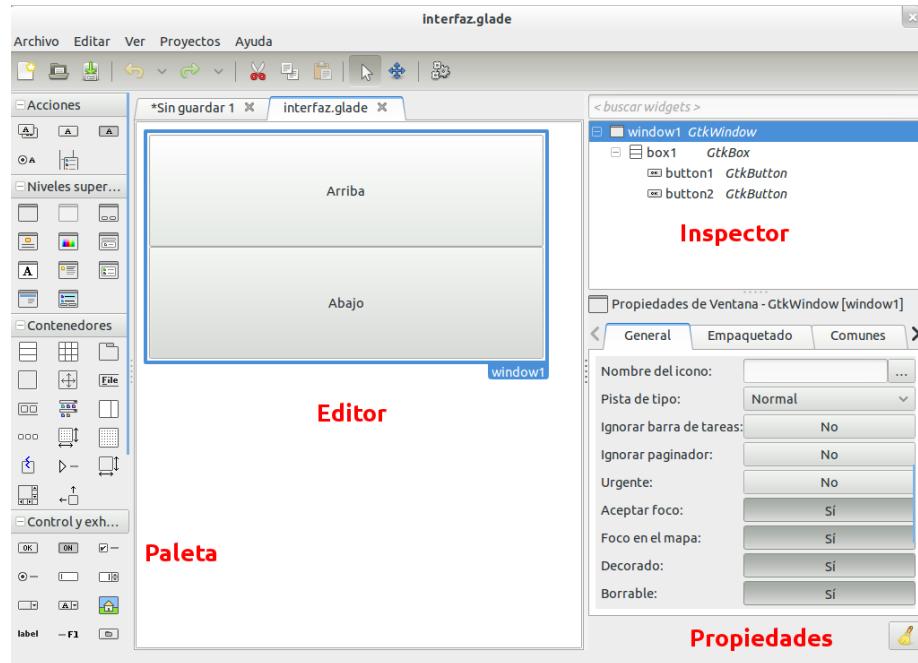


Figure 85: glade

```
# Importamos la librería
import pygame

# Iniciamos Pygame
pygame.init()

# Creamos una surface (la ventana de juego), asignándole un alto y un ancho
Ventana = pygame.display.set_mode((600, 400))

# Le ponemos un título a la ventana
pygame.display.set_caption("Hola Mundo")
```

Ejemplo de animaciones

Tutorial de la OSL

## APIS : conectando con el mundo exterior

Una de las ventajas de usar linux es que podremos integrarnos con muchos dispositivos para los que ya existen programas

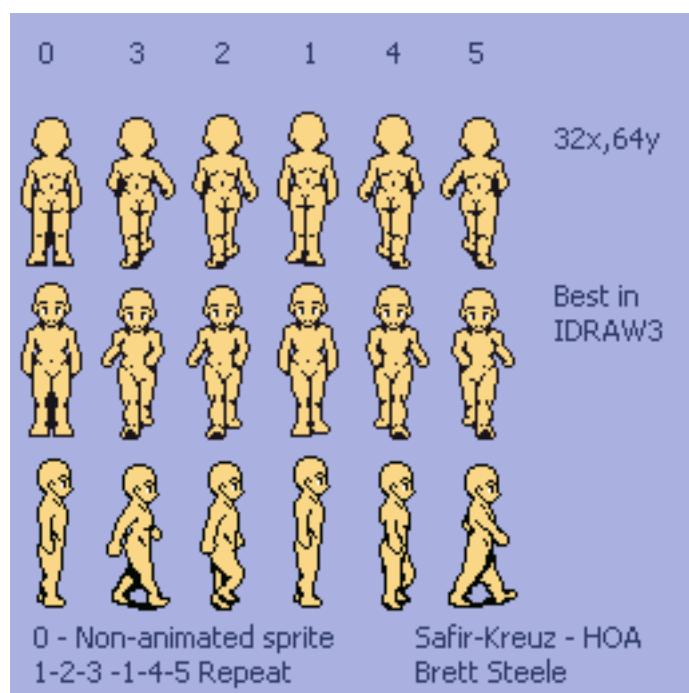


Figure 86: Ejemplo de animaciones

## Integración con gtalk

Vamos a crear un bot que realicen las acciones que le enviemos desde cualquier dispositivo

```
$ sudo apt-get install python-pip git-core python2.7-dev
```

Ahora actualizamos la lista de paquetes

```
$ sudo easy_install -U distribute
```

Procedemos ahora a instalar los paquetes que vamos a necesitar

```
$ sudo pip install Rpi.GPIO xmpppy pydns
```

Podemos descargar el código del enlace correspondiente.

En este código tendremos que cambiar los valores de las 3 variables a los correspondientes correos y contraseñas:

```
BOT_GTALK_USER = 'bot_username@gmail.com'  
BOT_GTALK_PASS = 'password'  
BOT_ADMIN = 'admin_username@gmail.com'
```

Una vez configurado, lo ejecutamos (como root debido a que usa GPIO).

```
$ sudo python ./raspiBot.py
```

y tendremos accesibles los siguientes comandos:

```
[pinon|pon|on|high] [pin] : activa el GPIO pin  
[pinoff|poff|off|low] [pin] : apaga GPIO pin  
[write|w] [pin] [state] : escribe el estado en GPIO pin  
[read|r] [pin]: lee el estado de GPIO pin  
[shell|bash] [arg1] : ejecuta el comando que sigue a "shell" o "bash"
```

## Reconocimiento de voz

Usaremos el API de Google, pero antes tenemos que digitalizar la voz para lo que instalaremos el paquete ffmpeg

```
$ sudo apt-get install ffmpeg
```

Con este paquete crearemos un fichero que enviaremos a google via su API para recuperar el texto. Veamos todo esto en un script al que llamaremos speech2text.sh y al que daremos permiso de ejecución (chmod +x speech2text.sh)

```
$#!/bin/bash
echo "Grabando.. Pulse Ctrl+C para parar."
arecord -D "plughw:1,0" -q -f cd -t wav | ffmpeg -loglevel panic -y -i - -ar 16000 -acodec flac
echo "Procesando..."
wget -q -U "Mozilla/5.0" --post-file file.flac --header "Content-Type:audio/x-flac; rate=16000" -O output.flac
```

```

echo -n "Dijiste: "
cat stt.txt
rm file.flac > /dev/null 2>&1

```

## Preguntando a WolphranAlpha

Instalamos lo necesario

```

$ apt-get install python-setuptools easy_install pip
$ sudo python setup.py build
$ sudo python setup.py

```

Y este es el código

```

#!/usr/bin/python
import wolframalpha
import sys
# Obtenemos una clave en http://products.wolframalpha.com/api/
# reemplaza la siguiente por la clave obtenida.
app_id='HY04TL-CLAVE'
client = wolframalpha.Client(app_id)
query = ' '.join(sys.argv[1:])
res = client.query(query)
if len(res.pods) > 0:
    texts = ""
    pod = res.pods[1]
    if pod.text:
        texts = pod.text
    else:
        texts = "No hay respuesta"
    # filtramos los caracteres
    texts = texts.encode('ascii', 'ignore')
    print texts
else:
    print "No hay respuesta"

```

## Haciendo que nos hable nuestra raspberry

Instalamos el reproductor mplayer

```
sudo apt-get install mplayer
```

Y con este script podemos hacer que nos lea lo que queramos

```

#!/bin/bash
say() { local IFS=+;/usr/bin/mplayer -ao alsa -really-quiet -noconsolecontrols "http://$1" say $* }

```

## Python y la cámara: openCV

Vamos a ver cómo utilizar librerías de reconocimiento de objetos para detectar formas y colores en imágenes provenientes de la cámara

Vídeo demostración

Usaremos la librería OpenCV, librería desarrollada para trabajar en sistemas de reconocimiento de imágenes.

Es una de las más utilizadas Funcionar independientemente de la fuente de las imágenes, (tiempo real o imágenes almacenadas)

Un proyecto sencillo: detectar la posición de unos círculos en la imagen.

(Detección de una pelota en el suelo. Basta con que sepamos “restar” el fondo a nuestra imagen)

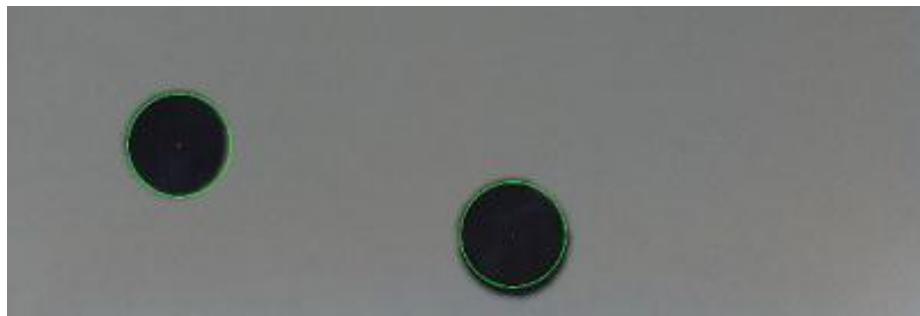


Figure 87: Detectando Círculos

Instalamos

```
sudo apt-get install python-opencv
```

Veamos el código necesario. Comenzaremos incluyendo los paquetes que vamos a utilizar:

```
import os
import cv2
import math
```

Vamos a reescalar todas las imágenes para así trabajar siempre con imágenes de un 1/4 del tamaño original. Podemos encontrar más detalles sobre las transformaciones geométricas disponibles en la documentación de openCV:

```
## Usamos el método resize para re-escalar
def resizeImage(img):
    dst = cv2.resize(img,None, fx=0.25, fy=0.25, interpolation =
cv2.INTER_LINEAR)
    return dst
```

Supondremos que capturamos la imagen con la cámara de la Raspberry con raspistill, pero si no fuera así, basta con cargar las imagen estática que tengamos (os.system ejecuta un comando del sistema):

```
## Capturamos la imagen con la cámara de la Raspberry Pi  
os.system("raspistill -o image.jpg")
```

Ahora cargaremos la imagen, primero en color y luego en escala de grises (ya vamos viendo la potencia de la librería, facilitándonos este tipo de cosas)

```
## Cargamos la imagen  
img = cv2.imread("/home/pi/Desktop/image.jpg")  
grey = cv2.imread("/home/pi/Desktop/image.jpg",0) #0 para la escala de grises
```

A partir de la imagen en grises obtendremos una imagen en blanco y negro utilizando un valor umbral de 50 y convirtiendo cualquier pixel que tenga un valor mayor en uno negro (255).

```
## convertimos la imagen en grises en una en blanco y negro  
ret, thresh = cv2.threshold(grey,50,255,cv2.THRESH_BINARY)
```

A partir de esta imagen, aplicamos un método llamado HoughCircle para encontrar el centro de los círculos. Es posible que tengamos que modificar estas valores para el caso concreto de nuestras imágenes.

```
## el método houghcircles encuentra el centro de los círculos  
circles = cv2.HoughCircles(thresh, cv2.cv.CV_HOUGH_GRADIENT, 1,75,param1=50,param2=13,min
```

El resultado será un conjunto de tuplas de 3 valores: x,y y el radio, que recorreremos para dibujar sobre la imagen los círculos detectados.

```
for i in circles[0,:]:  
    # dibujamos el círculos exterior  
    cv2.circle(img,(i[0],i[1]),i[2],(0,255,0),2)  
    # dibujamos el centro  
    cv2.circle(img,(i[0],i[1]),2,(0,0,255),3)
```

Sólo nos queda dibujar estas imágenes de vuelta en nuestra raspberry

```
## Re-escalamos las imágenes  
img = resizeImage(img)  
thresh = resizeImage(thresh)  
## las mostramos en pantalla  
cv2.imshow("thresh",thresh)  
cv2.imshow("img",img)
```

## Mathematica en nuestra Raspberry Pi

Hay una versión gratuita de Wolfram por defecto en Raspbian

# **Electronica**

**Cuidados**

**Adaptadores**

**Potencia**

**Motores**

Servo desde python

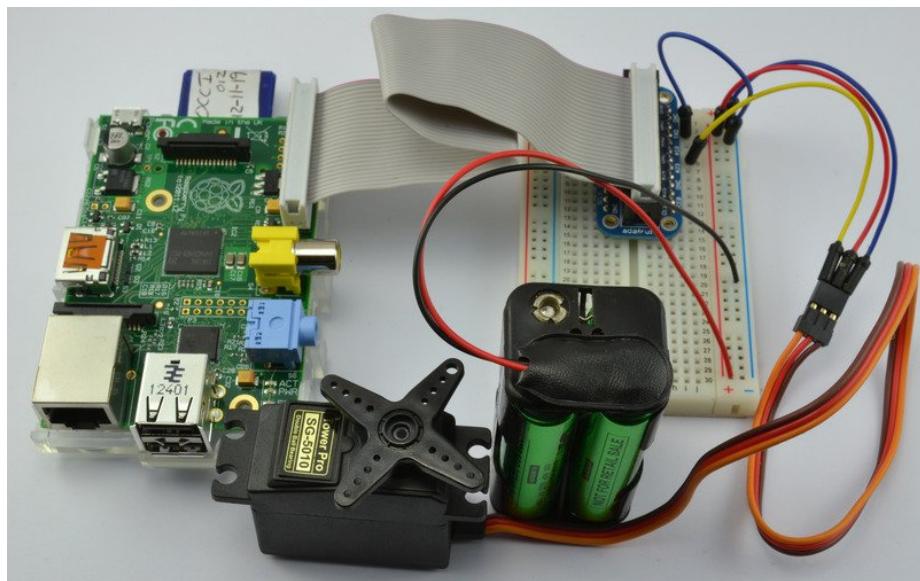


Figure 88: Servo desde python

Varios motores

**Sensores**

Sensores de temperatura digitales

**Algunos enlaces**

Veamos algunos enlaces interesantes sobre tutoriales de electrónica  
\* [http://www.sc.ehu.es/sbweb/electronica/elec\\_basic/default.htm](http://www.sc.ehu.es/sbweb/electronica/elec_basic/default.htm) \*

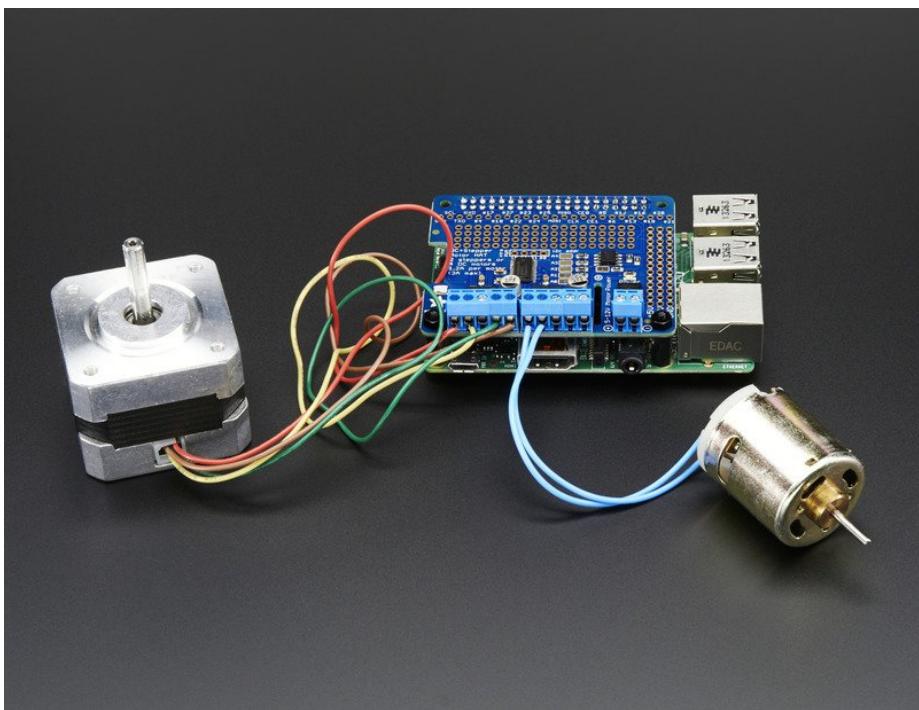


Figure 89: Controlando varios motores

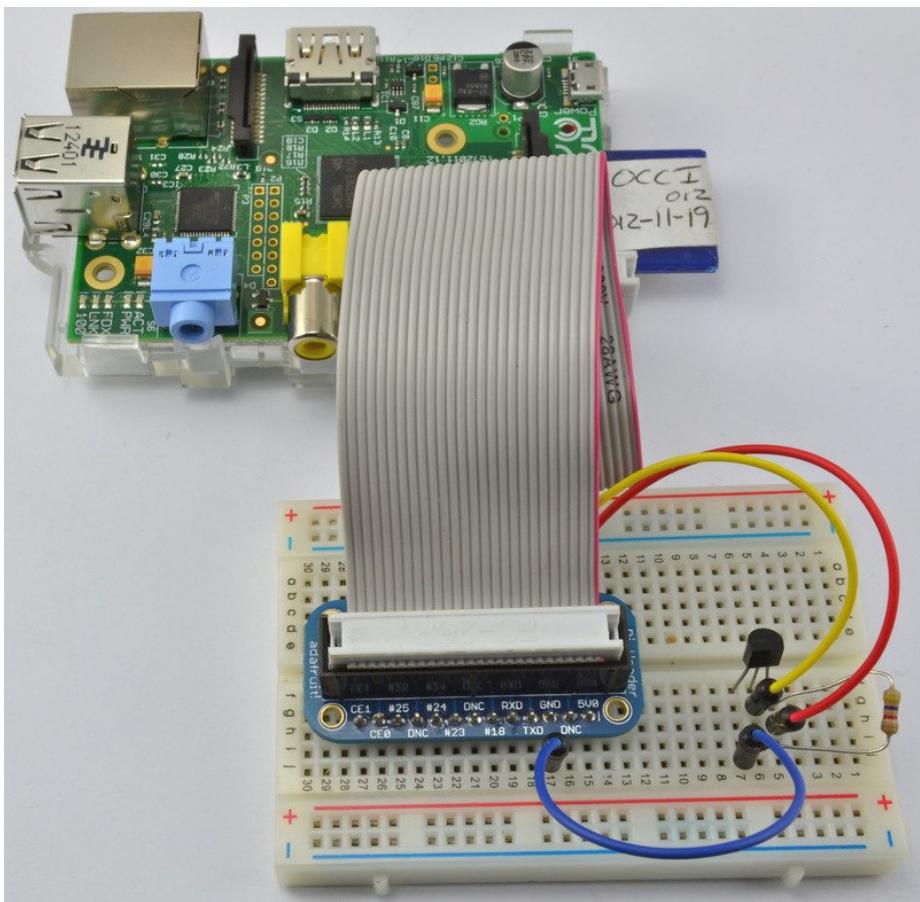


Figure 90: Sensores de temperatura

http://www.tutoelectro.com/ \* http://www.electronicafacil.net/circuitos/ \*  
 http://www.areatecnologia.com/ \* http://www.simbologia-electronica.com/ \*  
 Un instructable que empieza desde lo más fundamental y llega hasta montar  
 un circuito oscilador con un 555. Lo más probable es que cualquiera que  
 tenga una mínima inquietud por el tema se pueda saltar los primeros pasos,  
 pero en cualquier caso vale la pena. \* Otro instructable que explica diversos  
 componentes y sus símbolos en los esquemas \* Un repositorio de circuitos enorme,  
 \* Un listado de bloques básicos para entender y crear circuitos electrónicos \*  
 Estudios teóricos de electronica: el club de electrónica tiene montado un muy  
 completo grupo de tutoriales que abarcan desde los conceptos básicos de la  
 corriente eléctrica hasta ejemplos de circuitos y proyectos básicos, pasando por  
 componentes, como por ejemplo los transistores \* Página de documentación de  
 la tienda yourduino: Páginas interesantes que he visto tratan sobre manejo de  
 potencia con arduino, libros sobre

## GPIO

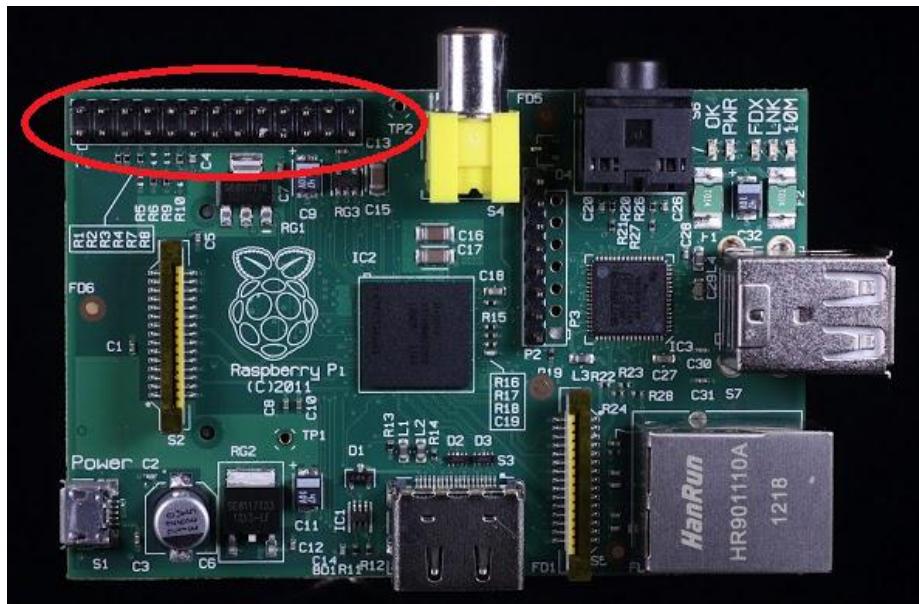


Figure 91: GPIO

- Son los pines que podemos usar como salidas o como entradas, pero siempre de tipo digital.
- Utilizan **3.3V**
- Podemos configurar cada uno como entrada o como salida

- Algunos de ellos se pueden usar como comunicaciones especializadas: SPI, I2C, UART

## Precauciones

- Antes de realizar cualquier tipo de conexión en los conectores o pines debemos de tener siempre la precaución de tener desconectada la alimentación de la Raspberry Pi.
- Evitaremos derivaciones eléctricas o cortos .
- Conviene recordar que los pines de la CPU de la placa están conectados directamente a los diferentes conectores y pines, con lo que cualquier cosa que hagamos sobre los pines la estamos haciendo directamente sobre la CPU.
- También hay que tener en cuenta que los pines GPIO no soportan 5 V, sólo 3.3V y un máximo de 16 mA, por lo que hay que tomar precauciones en este sentido.

## Pines

Hay que tener cuidado con no equivocarse. Podemos usar una etiqueta



Figure 92: Etiquetas para los pines

Las distintas versiones tienen algunos pines distintos

Las versiones de 40 pines

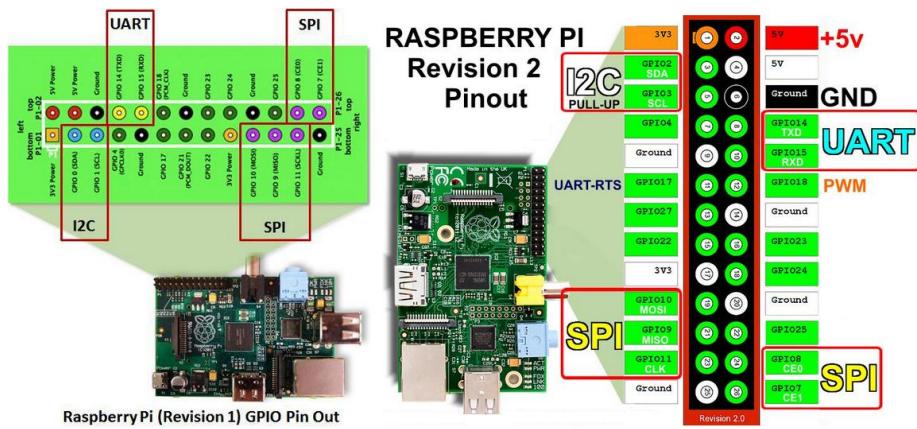


Figure 93: GPIO para la versión 2

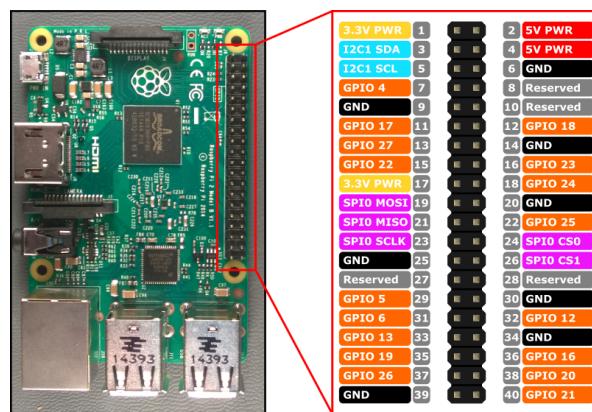


Figure 94: GPIO de 40 pines

## Librerías

Hay 4 librerías GPIO

- Shell (línea de comandos)
- Rpi. GPIO
- wiringPi (Gordon Henderson [wiringpi.com](http://wiringpi.com))
- BCM 2835

Veamos como llaman a los distintos pines

P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Figure 95: Nombre de los GPIOs

## Wiring

Para instalarlo tenemos que tener instalado parte del entorno de desarrollo de python

```
sudo apt-get install python-dev python-setuptools git-core
```

Descargamos el código (también podíamos haber descargado el fichero zip)

```
git clone git://git.drogon.net/wiringPi
```

La compilamos

```
./build
```

Y ya podemos udarla

```
gpio readall
```

wiringPi	GPIO	Phys	Name	Mode	Value
0	17	11	GPIO 0	IN	Low
1	18	12	GPIO 1	IN	Low
2	27	13	GPIO 2	IN	Low
3	22	15	GPIO 3	IN	Low
4	23	16	GPIO 4	IN	Low
5	24	18	GPIO 5	IN	Low
6	25	22	GPIO 6	IN	Low
7	4	7	GPIO 7	IN	Low
8	2	3	SDA	IN	High
9	3	5	SCL	IN	High
10	8	24	CE0	IN	Low
11	7	26	CE1	IN	Low
12	10	19	MOSI	IN	Low
13	9	21	MISO	IN	Low
14	11	23	SCLK	IN	Low
15	14	8	TxD	ALT0	High
16	15	10	RxD	ALT0	High
17	28	3	GPIO 8	IN	Low
18	29	4	GPIO 9	IN	Low
19	30	5	GPIO10	IN	Low
20	31	6	GPIO11	IN	Low

Figure 96: Leer el estado de todos los pines

## Conectando un led

Este es el esquema para conectar un led

El montaje sería

Hagamos un programa que parpadea el led conectado

```
import time
# Importamos la librería wiringpi
import wiringpi2
#Configuramos la numeración de los pines con respecto al
#estandar de la librería wiringpi (pin de entrada salida
#    GPIO0)

io = wiringpi2.GPIO(wiringpi2.GPIO.WPI_MODE_PINS)

#Configuramos el pin 0 como salida
io.pinMode(0,io.OUTPUT)

# Ciclo for que ejecutamos 3 veces
for x in range (0,3):
    io.digitalWrite(0,io.HIGH) #encendemos el led
    time.sleep(0.5) # esperamos medio segundo
    io.digitalWrite(0,io.LOW) # apagamos el led
    time.sleep(0.5) # esperamos medio segundo
```

Para ejecutar estos programas necesitamos permiso de administrador

```
sudo python blink.py
```

## Conectado un pulsador

Usando el código

## Usando GPIO

Instalamos la librería

```
sudo apt-get install python-dev python-rpi.gpio
```

El programa que los usa

```
import RPi.GPIO as GPIO
# Usamos la numeración de los GPIO no el numero de los pines
GPIO.setmode(GPIO.BCM)
GPIO.setup(7, GPIO.IN) # establecemos el GPIO 7 como entrada
```

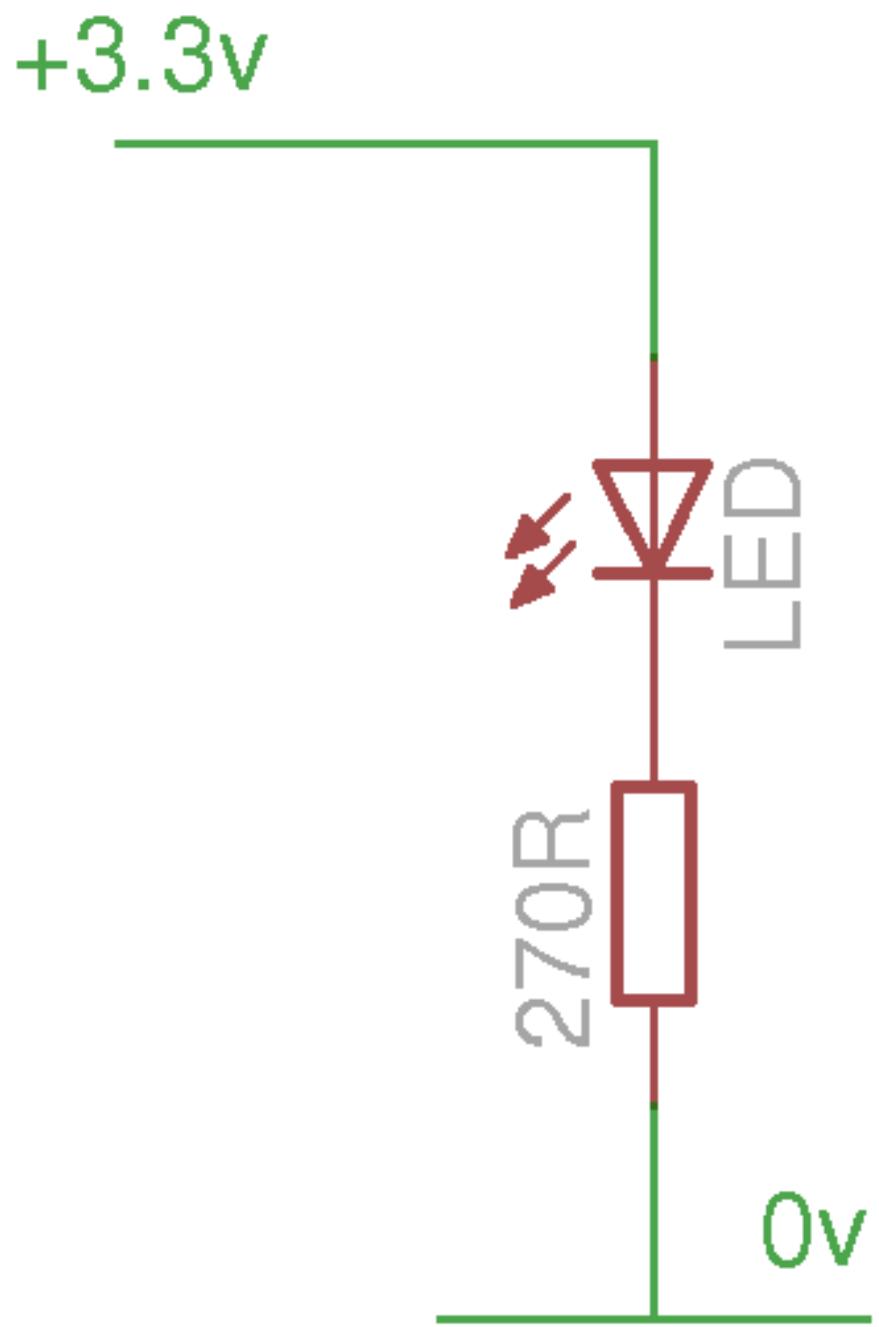


Figure 97: LED

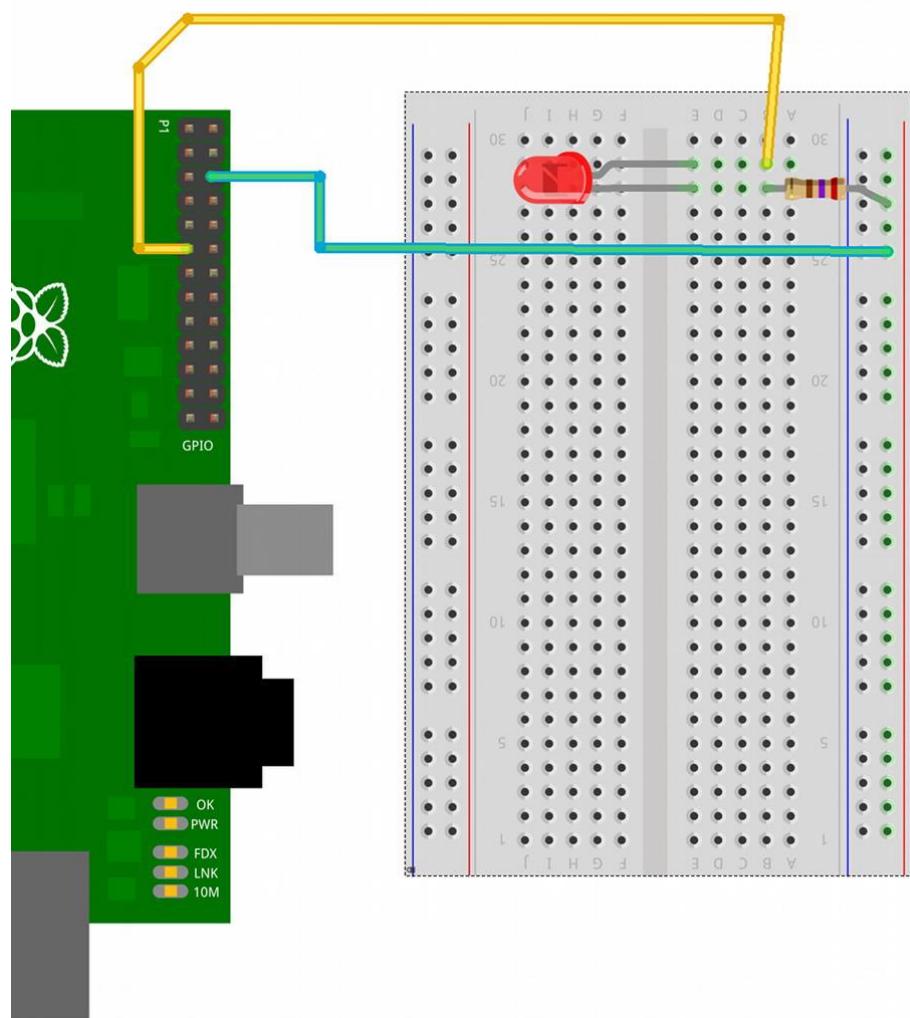


Figure 98: Esquema de conexión de un led

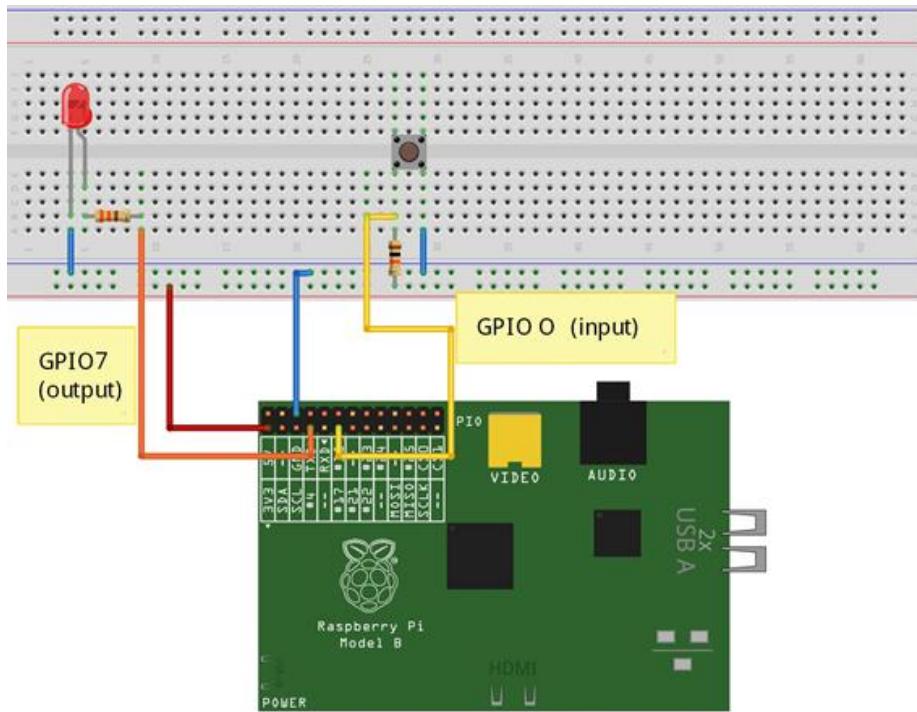


Figure 99: Conectando un pulsador

```
# Enciendo led según el estado de una entrada (pulsador)
import wiringpi2

io = wiringpi2.GPIO(wiringpi2.GPIO.WPI_MODE_PINS)

io.pinMode(7,io.OUTPUT) #Pin 7 como salida
io.pinMode(0,io.INPUT) # Pin 0 como entrada
io.pullUpDnControl(0,io.PUD_UP) # Habilito pulsador como entrada

while True: # ciclo infinito
    x=io.digitalRead(0) # Ve el estado del pulsador en 0
    if x==io.LOW: # Si esta pulsado, valor bajo
        io.digitalWrite(7,io.HIGH) # enciendo el led

    else:
        io.digitalWrite(7,io.LOW) # si no esta pulsado apaga el led
```

Figure 100: Código para usar un pulsador

```

GPIO.setup(8, GPIO.OUT) # establecemos el GPIO 8 como salida
input_value = GPIO.input(7) # recuperamos el valor de entrada
GPIO.output(8, True) # establecemos la salida en alto

```

O este ejemplo más complejo

```

import RPi.GPIO as GPIO
import time
# Usamos la posición en el conector
GPIO.setmode(GPIO.BCM)
# pin 11 (GPIO17) como output
GPIO.setup(11, GPIO.OUT)
var=1
print "Empezamos el bucle infinito"
while var==1 :
    print "Output False"
    GPIO.output(11, False)
    time.sleep(1) # esperamos un tiempo
    print "Output True"
    GPIO.output(11, True)
    time.sleep(1)

```

## Usando más potencia

En el caso bastante normal de que necesitemos más potencia de las que nos da un pin (16mA) Podemos utilizar un transistor. Veamos el montaje

A la salida de este transistor podemos conectar un relé para obtener aún más potencia

## Leyendo valores analógicos

Para leer valores analógicos usaremos electrónica externa, com pueden se esta placa o esta otra, ambas de 16 bits. El montaje es sencillo

## Usos de los GPIOs

- Encender apagar LEDs (no podemos aspirar a encender nada de mayor potencia directamente). Estas son las salidas digitales, capaces de estar en estado alto o bajo.
- Algunos de estos pines pueden generar PWM (modulación por ancho de pulso) protocolo que usan los servos.
- Detectar pulsaciones de botones/interruptores. Estas son las entradas digitales.
- Acceso al puerto serie por los terminales TX/TX
- Acceso al

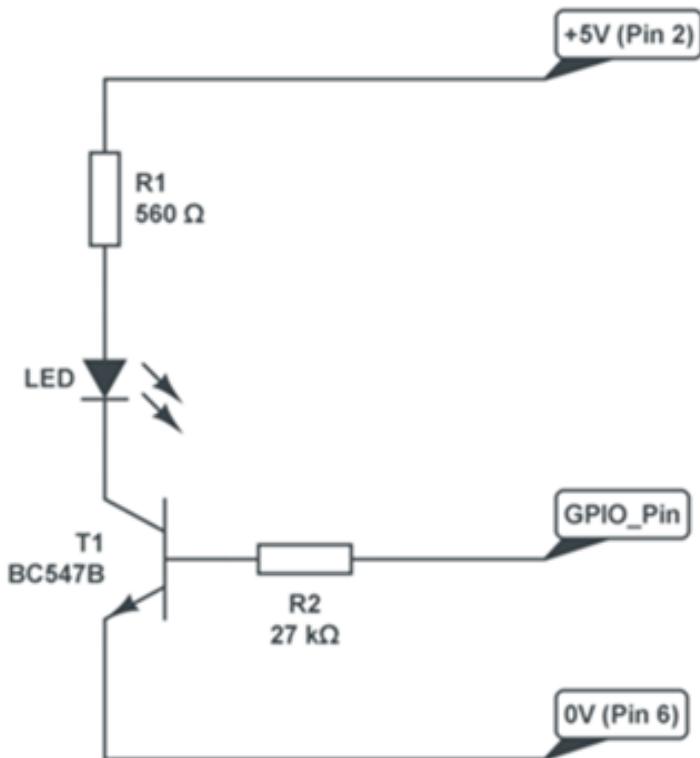


Figure 101: Conexión con transistor

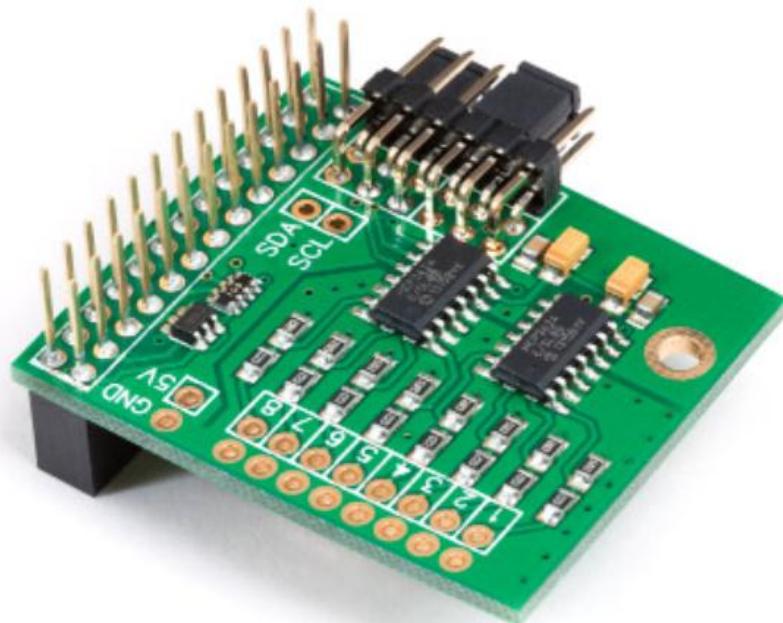


Figure 102: ADC

bus I2C, bus de comunicaciones usado por muchos dispositivos • Acceso al bus SPI, bus de comunicaciones similar al I2C pero con diferentes especificaciones

El bus I2C y SPI nos permiten conectar con dispositivos externos que nos expanden su funcionalidad. Es como si conectáramos periféricos a nuestra Raspberry.

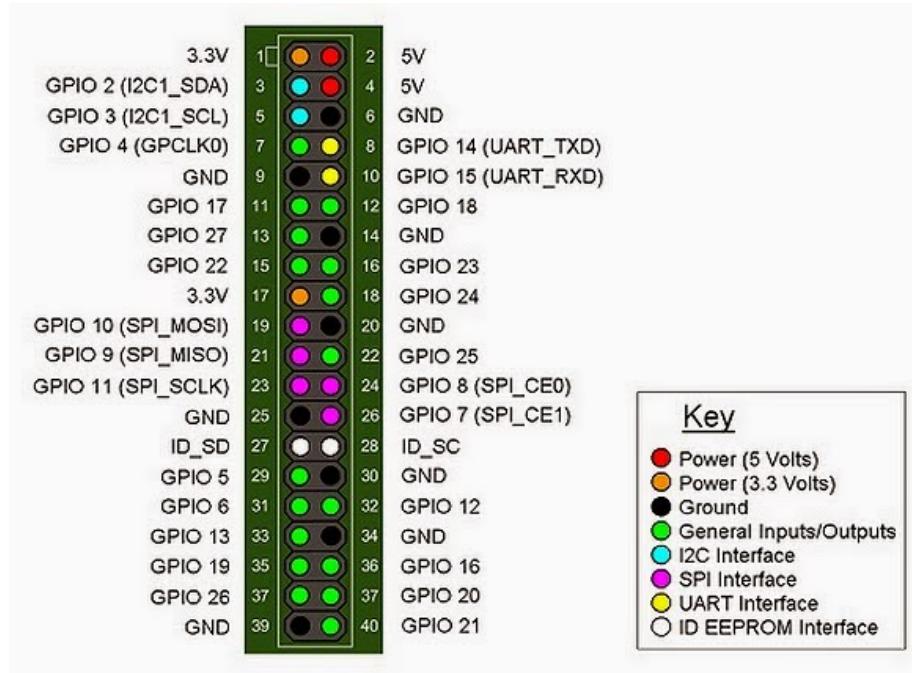


Figure 103: pins

- También están disponibles las líneas de alimentación de 5v y 3.3v y por supuesto tierra.
- Todos los pines se pueden configurar tanto de entrada como de salida.
- Algunos de los pines tienen una segunda función como por ejemplo los etiquetados como SCL y SDA utilizados para I2C y los MOSI, MISO y SCKL utilizados para conectar con dispositivos SPI.
- Hay que tener muy claro que todos los pines usan niveles lógicos de 3.3V y no es seguro conectarlos directamente a 5V, porque las entradas han de ser menores de 3.3V. Igualmente no podemos esperar salidas superiores a 3.3V.
- En caso de querer conectar con lógica de 5v tendremos que usar una electrónica para adaptar niveles.

- Existen dispositivos convertidores de niveles (level shifters) con diferentes tecnologías. Los más antiguos están formados por unas resistencias y unos transistores.

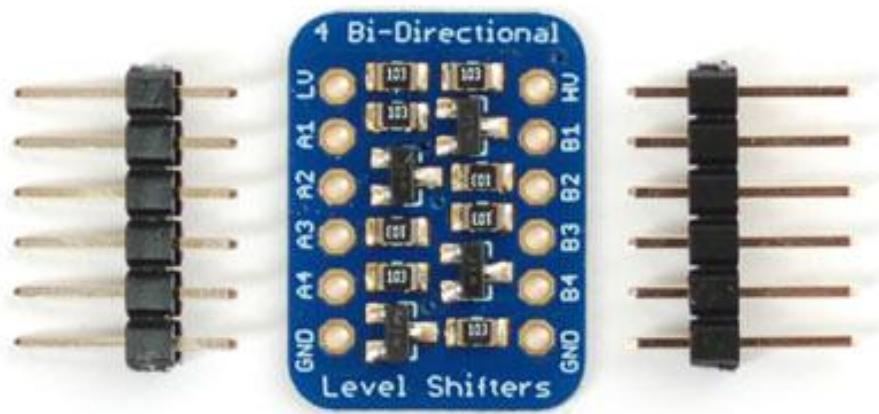


Figure 104: Conversores de niveles (shifter)

Para identificar más fácilmente los pines podemos usar una etiqueta

## Placas GPIO

### Clobber

- Es bastante arriesgado y complicado trabajar directamente con los pines del conector GPIO de la Raspberry.
- Existen en el mercado una gran variedad de placas que nos facilitan la vida.
- Algunas sólo nos facilitan la conexión.
- Otras nos proporcionan mayor funcionalidad.
- En cualquier caso ganamos en tranquilidad al usarlas.

Son simples adaptadores que nos facilitan la vida permitiendo conectar de manera sencilla con las placas de prototipo

3.3V	○ ○	5V
0 SDA	○ ○	N/C
1 SCL	○ ○	GND
4	○ ○	14 TXD
N/C	○ ○	15 RXD
17	○ ○	18
21	○ ○	N/C
22	○ ○	23
N/C	○ ○	24
10 MOSI	○ ○	N/C
9 MISO	○ ○	25
11 SCKL	○ ○	8
N/C	○ ○	7

Figure 105: etiqueta



Figure 106: Clobber

### PiPlate

Se trata de una placa de prototipo especialmente adaptada al tamaño de la Raspberry y que nos permite acceder de forma sencilla a los pines por nombre y funcionalidad.

### PiFace

- Tiene un fin claramente educativo,
- Incluye diferentes dispositivos
- Leds que se pueden activar independientemente,
- 2 relés para activar cargas de potencia y
- 4 pulsadores conectados a otras tantas entradas

### Slice of I/O

Se trata de una placa sencilla que nos permite acceder a 8 entradas y otras tantas salidas con la seguridad de que existe una electrónica que protege a nuestra Raspberry

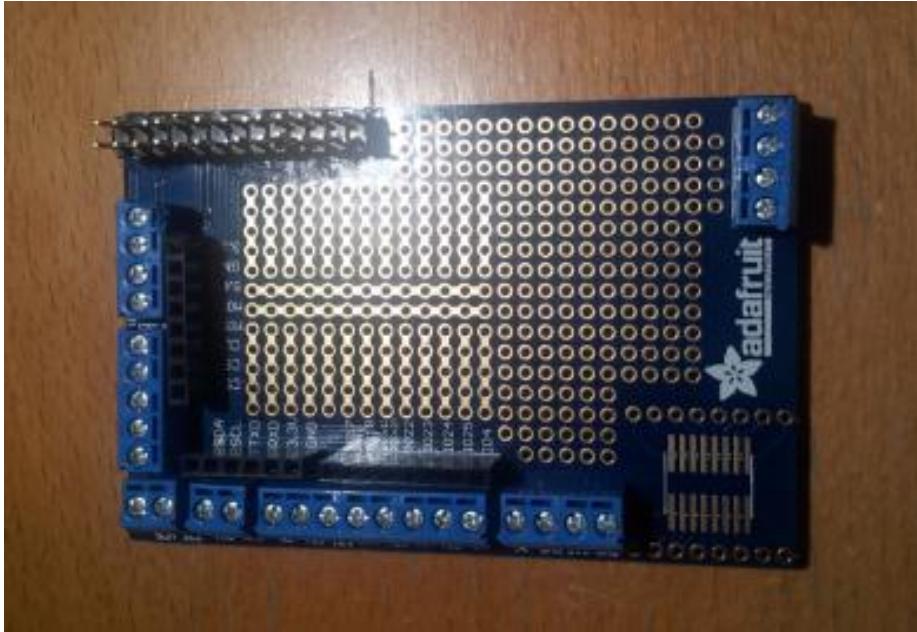


Figure 107: piplate

### Gertboard

Es una placa de desarrollo con una enorme cantidad de complementos, como son controladores de motores, ADC, DAC, 12 leds, 3 pulsadores y hasta un microcontrolador ATMega (similar a Arduino)

### RaspiRobot

- El manejo de motores es mucho más complejo que el manejo de leds.
- La programación es exactamente la misma,
- La electrónica necesaria para controlarlos es totalmente diferente.
- Si bien podemos conectar directamente un led a un pin de GPIO, conectar un motor es totalmente desaconsejable, por varias razones:
  - La primera porque los motores requieren de mayor potencia para funcionar,
  - Necesitaremos una electrónica capaz de gestionar estas potencias
  - Serán controladas desde los pines de la RaspBerry.
  - En caso de forzar la electrónica de alimentación de nuestra Raspberry a dar una mayor potencia podríamos quemarla.
  - El funcionamiento de los motores hace que estos generen al acelerar unas corrientes de inducción opuesto a las que les aplicamos

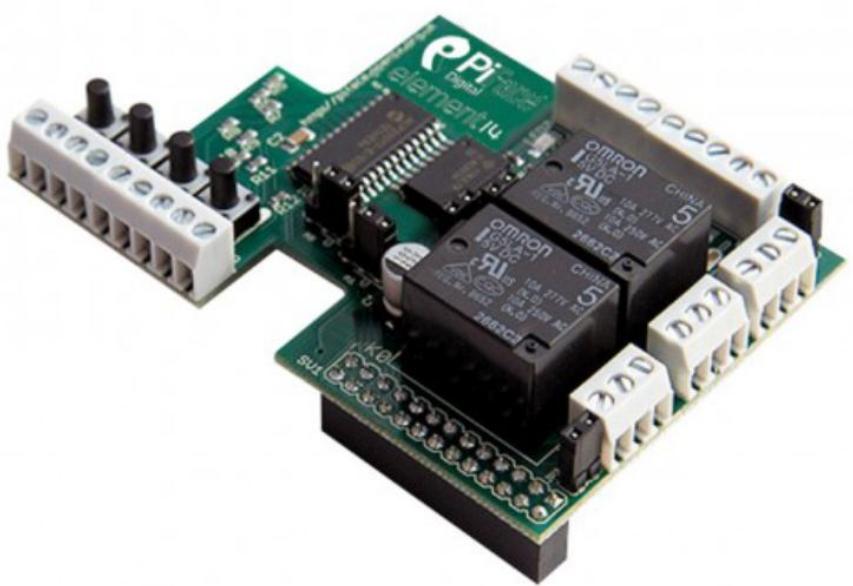


Figure 108: piface

## Board Overview

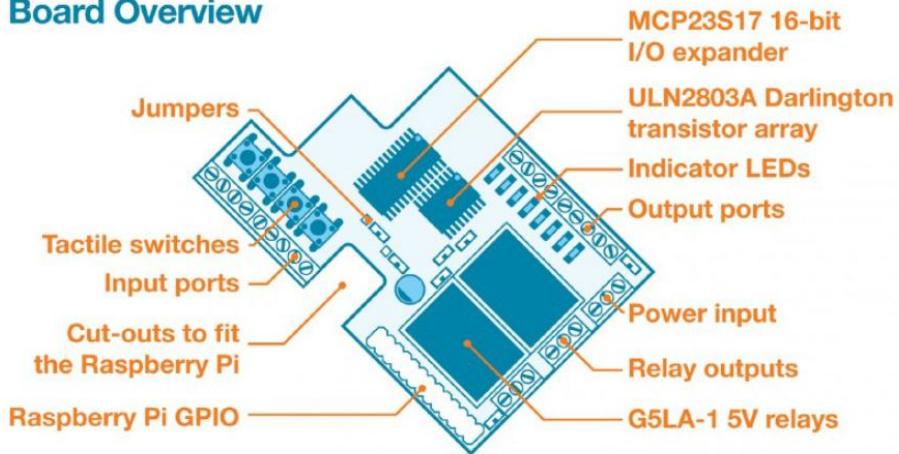


Figure 109: esquemapiface

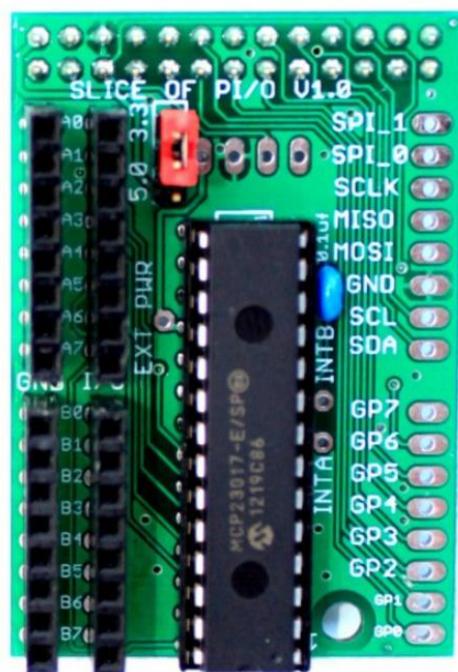


Figure 110: slice

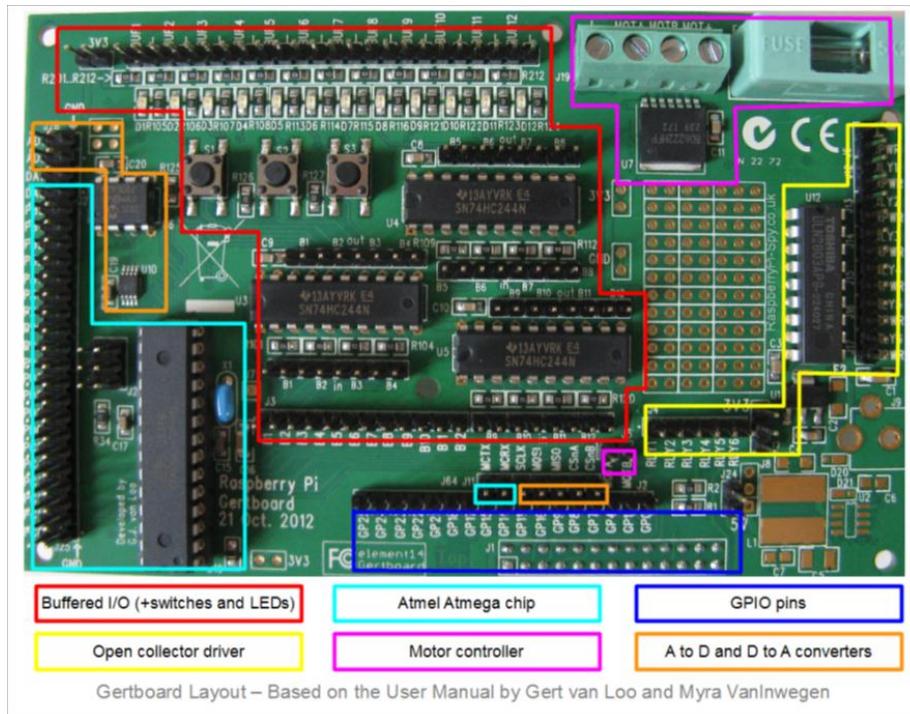


Figure 111: gertboard

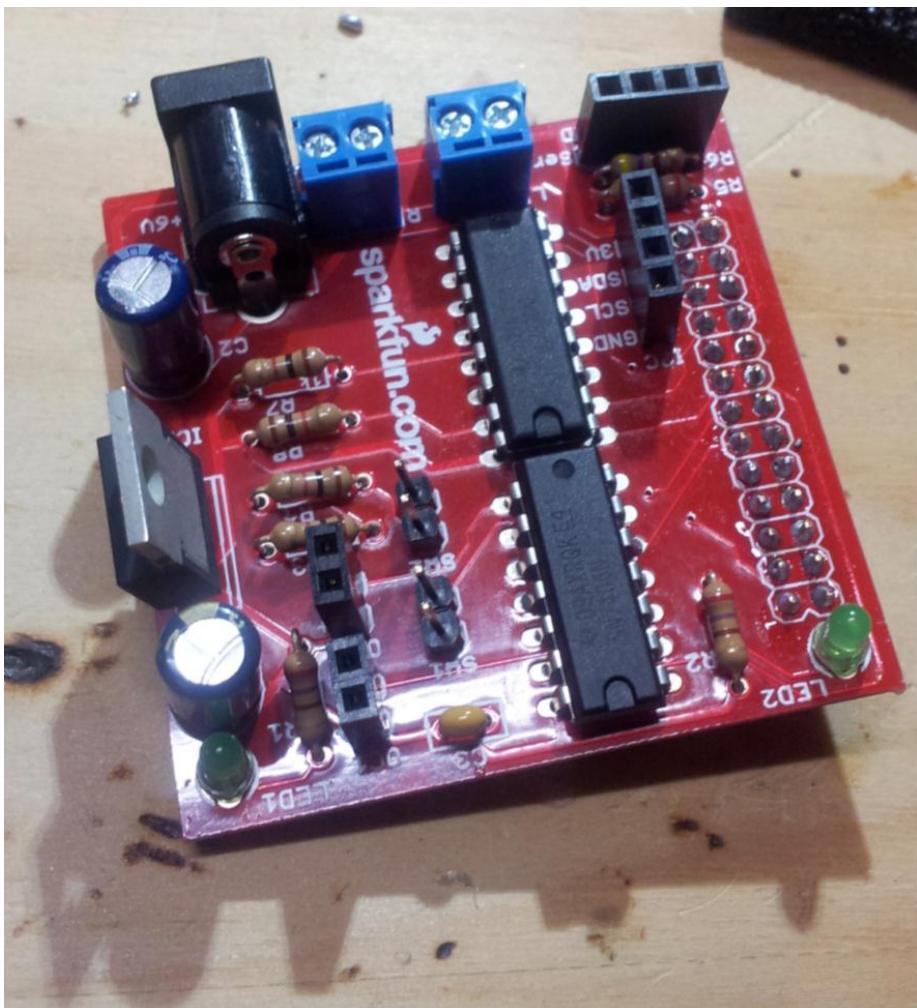


Figure 112: Raspirobot

para funcionar y que de no ser suprimidas podrían dañar la electrónica a la que están conectados.

En la web de raspbirobot vemos instrucciones de montaje

- Controla 2 motores,
- 2 leds,
- 2 entradas de pulsador,
- 2 salidas de colector abierto, para poder usar mayores potencias
- Conector I2C y
- otro serie

Descargamos la librería

```
wget https://github.com/simonmonk/raspirobotboard/archive/master.zip
```

y la instalamos

```
sudo python setup.py install
```

Un programa podría ser

```
from raspirobotboard import *
rr = RaspiRobot() # creamos el objeto
rr.set_led1(1) # activamos el led 1
rr.set_led2(0) # desactivamos el led 2
rr.set_oc1(1) # activamos la salida 1
rr.forward() # movemos los dos motores hacia adelante
rr.reverse() # movemos los dos motores hacia atrás
rr.left() # motor izquierdo hacia adelante, derecho hacia atrás
rr.right() # motor izquierdo hacia atrás, derecho hacia adelante
rr.stop() # los dos motores hacia atrás
rr.sw1_closed() # devuelver True o False según cerrado o abierto
```

### Steppers: motores paso a paso

Los motores paso a paso son motores que nos permiten una gran precisión de giro, pudiendo determinar su moviendo en grados.

Vamos a ver cómo usar el motor de la imagen, que tiene 4 bobinas. La placa de control es muy sencilla y necesita de 4 pines para controlarla (en realidad la placa sólo transforma la salida de los pines de raspberry en una señal de la potencia que necesita el motor)

Veamos como conectarla 5V (P1-02) GND (P1-06) Inp1 (P1-18) Inp2 (P1-22) Inp3 (P1-24) Inp4 (P1-26)

Vamos a ver ahora la programación.

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
```



Figure 113: stepper

```

StepPins = [24,25,8,7] # Pines que conectamos a la placa de control
for pin in StepPins: # configuramos todos los pines como salida
    GPIO.setup(pin,GPIO.OUT)
    GPIO.output(pin, False)
StepCounter = 0
WaitTime = 0.5
StepCount1 = 4
Seq1 = []
Seq1 = range(0, StepCount1) # Definimos la secuencia de giro
Seq1[0] = [1,0,0,0]
Seq1[1] = [0,1,0,0]
Seq1[2] = [0,0,1,0]
Seq1[3] = [0,0,0,1]
while 1==1: # realizamos un bucle infinito enviando la secuencia
    for pin in range(0, 4): #iteramos sobre los pasos de la secuencia
        xpin = StepPins[pin]
        if Seq[StepCounter][pin]!=0:
            GPIO.output(xpin, True)
        else:
            GPIO.output(xpin, False)
        StepCounter += 1
        time.sleep(WaitTime)

```

Veamos un ejemplo de su precisión

## Servos

Los servos son motores pensados para mantener una posición concreta, disponen de electrónica de control propia y a la se le indica la posición que deben mantener mediante un pulso que hay que enviar 50 veces por segundo.

El ancho de este pulso determina la posición a mantener, como podemos ver en la imagen adjunta.

La estabilidad de la posición depende de la precisión con la enviamos la señal de control.

Veamos un método para generar esta señal con python. Está pensada para controlar 2 servos:

```

def mover_servo(grados,servo):
    if servo==1: GPIO_servo=22
    elif servo==2: GPIO_servo=21
    # creamos el pulso
    pos_servo=(0.0000122*grados)+0.0002
    GPIO.output(GPIO_servo, True) #activamos la salida
    time.sleep(pos_servo) # esperamos la duración del pulso

```

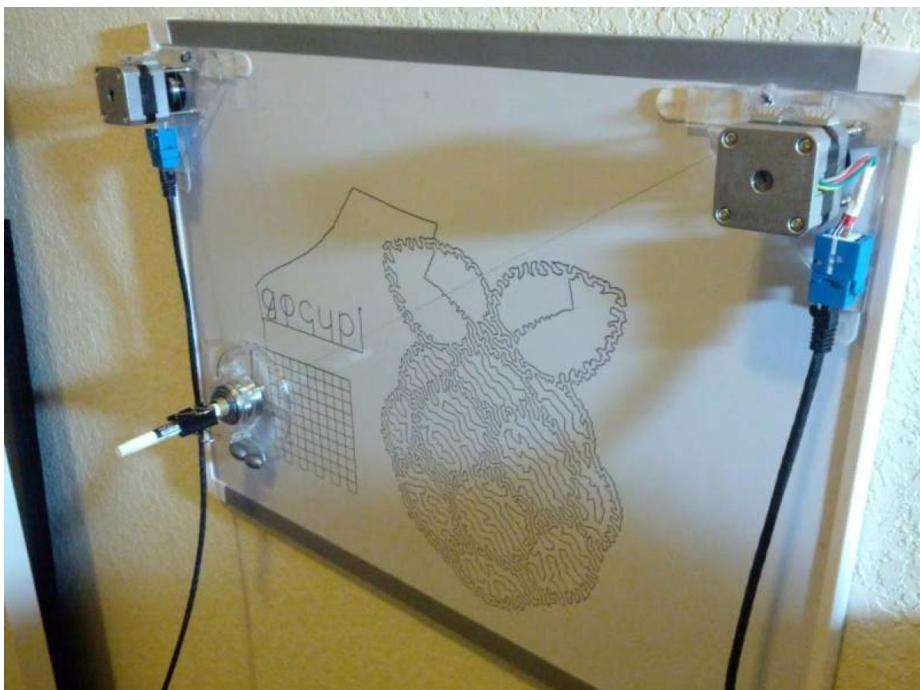


Figure 114: Robot polarplot

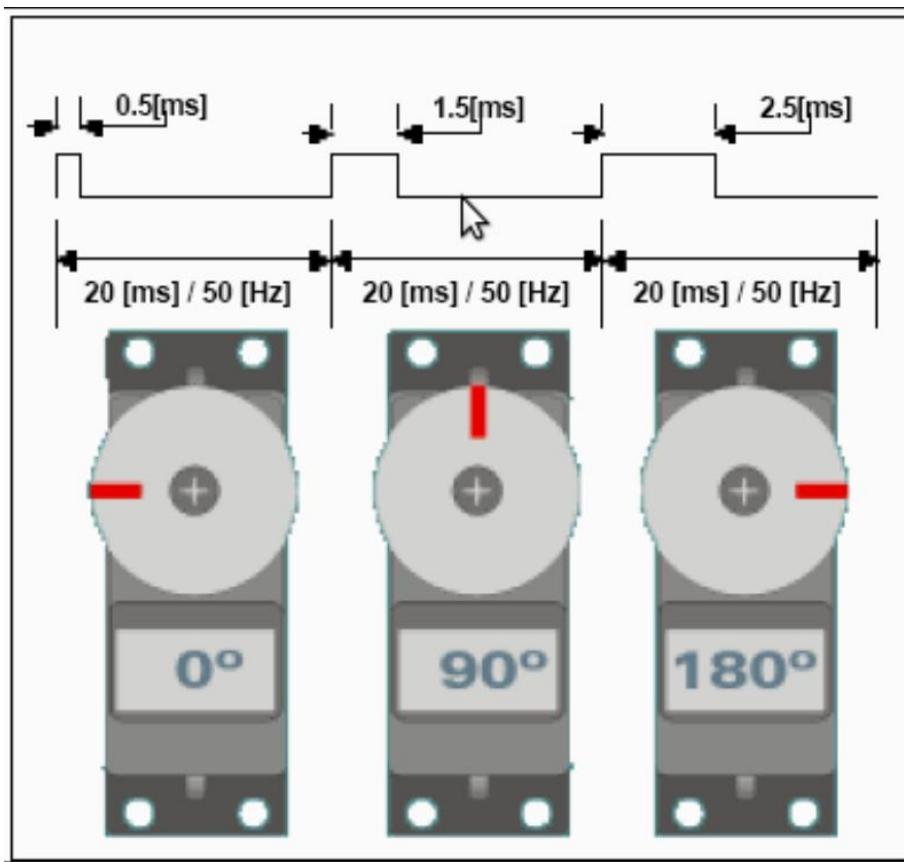


Figure 115: Control de servos

```
GPIO.output(GPIO_servo, False) # desativamos la señal porque el pulso ha terminado  
#esperamos el tiempo necesario hasta enviar el siguiente pulso  
time.sleep(0.0025-pos_servo)
```

Si lo probamos veremos que el servo vibra debido a la mala calidad de la señal por su falta de estabilidad. Python es un lenguaje interpretado y temporización que hemos hecho dependerá de la carga que tenga nuestra Raspberry

Podemos mejorar la calidad de la señal utilizando un programa escrito en C que producirá una mejor temporización.

## Uniendolo todo

Vamos a utilizar un par de servos para hacer que una cámara siga una cara

Estos son los pasos para instalar todo lo necesario

```
sudo apt-get update  
sudo apt-get install git python-opencv python-all-dev libopencv-dev  
sudo modprobe servoblaster  
git clone https://github.com/mitchtech/py_servo_facetracker
```

Y para ejecutarlo

```
cd py_servo_facetracker  
python ./facetracker_servo_gpio.py 0
```

## Complementos (Addons)

### Dispositivos I2C

### Otros

Conectando una pantalla 5110 <http://www.elecfeeks.com/6179.html> leds RGB Raspilight [http://www.adafruit.com/blog/2013/03/01/raspberry-pi-and-led-strips-piday-raspberrypi-raspberry\\_pi/](http://www.adafruit.com/blog/2013/03/01/raspberry-pi-and-led-strips-piday-raspberrypi-raspberry_pi/) [https://github.com/labatrockwell/raspberrypi-experiments/tree/master/Led\\_Strip\\_Library](https://github.com/labatrockwell/raspberrypi-experiments/tree/master/Led_Strip_Library) [http://www.adafruit.com/blog/2013/01/18/raspberry-pi-pwm-rgb-led-strip-piday-raspberrypi-raspberry\\_pi/](http://www.adafruit.com/blog/2013/01/18/raspberry-pi-pwm-rgb-led-strip-piday-raspberrypi-raspberry_pi/) <http://www.raspberrypi.org/phpBB3/viewtopic.php?t=3010>

10 cosas que conectar a una raspberry

Muchas más cosas que conectar a nuestras Raspberry

<http://www.raspberrypi-spy.co.uk/2013/03/top-10-things-to-connect-to-your-raspberry-pi/>



Figure 116: Opencv + Camara  
110

## **Revistas**

Revista The MagPy

Todos los números

Libro de proyectos

## **Libros**

- Programing the Raspberry Pi, Simon Monk
- Raspberry Pi for Secret Agents
- Raspberry Pi Gamming

## **Tutoriales**

Raspberry pi Class (instructables)

Vídeo tutorial Raspberry Pi (sparkfun)

Vídeo tutorial Raspberry Pi 2 (sparkfun)

Tutoriales de Adafruit

Tutoriales de sparkfun

## **Referencia**

wikipedia

Products at Raspberry.org

# **RaspiFAQ**

## **General**

- ¿Es openSource?  
Casi sí, pero lo será
- ¿De verdad cuesta 35\$?  
La placa sí, pero por si misma no es más que un pisapapeles Geek
- ¿Cómo la alimento?  
Por USB Micro (como los móviles) con 5v y al menos 1A (mejor 2A)

- ¿Puede funcionar con pilas?

Depende de las pilas, con una batería externa (como las de los móviles) Sí

- ¿Qué significan las luces?

```
PWR      5V alimentación ok
OK     Acceso a la SD
FDX     Ethernet Full Duplex conectada
LNK     Ethernet conectado
10M Ethernet de 100 Mbps conectada
```

- ¿Cómo debo apagar mi raspberry?

La mejor forma de apagarlas es usando el comando halt

```
sudo halt
```

ó

```
sudo shutdown -h
```

- ¿Se rompe si le quito la alimentación?

No debería pero pudiera ocurrir si se están escribiendo muchos archivos  
(es un tema de probabilidad)

- ¿Qué versión tengo?

Podemos saber la versión de Raspberry que tenemos usando el siguiente comando

```
cat /proc/cpuinfo
```

Obtendremos una información similar a esta

```
Processor      : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS      : 847.05
Features       : swp half thumb fastmult vfp edsp java tls
CPU implementer : 0x41
CPU architecture: 7
CPU variant   : 0x0
CPU part      : 0xb76
CPU revision  : 7
Hardware       : BCM2708
Revision       : 0002
Serial         : 0000000000abc0ab1
```

Según el valor que aparezca en el campo Revision tendremos una versión u otra

Existen 3 versiones:

Modelo y Revision	Hardware Revision Code de cpuinfo
Model B Revision 1.0	0002

```
Model B Revision 1.0 + ECN0001 (no fuses, D14 removed) 0003  
Model B Revision 2.0          0004, 0005, 0006
```

## Cacharreo (cables)

- ¿Puedo encender y apagar un led?  
Sí, pero con cuidado
- ¿Puede controlar un motor?  
No directamente, sí con una plaquita
- ¿Qué necesito para hacer un robot?  
Una placa controladora, y motores ...

## Compras

- ¿Dónde puedo comprar en Granada?

## Administrando (¡es linux!)

- ¿Cuál es el usuario por defecto?  
Es “pi”
- ¿Cuál es la contraseña por defecto del usuario pi?  
Es “raspberry”
- ¿Cuál es la contraseña del usuario root?

El usuario root no tiene contraseña para evitar acceso indeseados. Para ejecutar algún comando como root podemos usar el comando “sudo”

`sudo comando`

nos solicitará la contraseña del usuario actual

Si necesitamos por alguna razón permanecer como root (lo cual se desaconseja en todos los Linux) podemos usar

`sudo su -i`

ó

`sudo su -`

Cuando acabemos podemos salir con Ctrl-D o con “exit”