


Semana 2: RaspberryPi

SISTEMAS EMBEBIDOS 
(EMBEDED SYSTEMS)

Grado Dual en Industria Digital

Campus Vitoria

Curso 2020-2021



Deusto

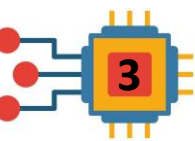
Facultad de Ingeniería
Ingeniaritza Fakultatea



***THE STORY OF THE \$35 COMPUTER
THAT CHANGED THE WORLD***

De donde viene la RaspberryPi ?

- Creada por Raspberry Pi Foundation (2008) y comercializada partir de Febrero 2012 (Eben Upton, Jack Lang, Alan Mycroft, and Rob Mullins): –
“**We provide low-cost, high-performance computers that people use to learn, solve problems and have fun.**”
- No es *Open Source Hardware*, aunque la mayor parte del código fuente si es Open Source al estar basado en Linux (p.e., el bootloader no)
- Mucha orientación a la comunidad y para educación (proyecto astro pi), siendo a su vez muy potente para aplicaciones en uso real.



Modelos



Raspberry Pi 4 Model B

Your tiny, dual-display, desktop computer



Raspberry Pi 3 Model A+

Our third-generation single-board computer, now in the A+ format



Raspberry Pi 3 Model B+

The final revision of our third-generation single-board computer



Raspberry Pi 3 Model B

Our third-generation single-board computer



Raspberry Pi 2 Model B

The Raspberry Pi 2 Model B is the second-generation Raspberry Pi



Raspberry Pi 1 Model B+

The Model B+ is the final revision of the original Raspberry Pi



Raspberry Pi 1 Model A+

The Model A+ is the low-cost variant of the Raspberry Pi



Raspberry Pi Zero W

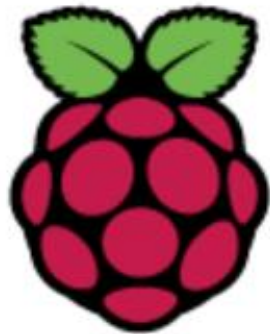
Single-board computer with wireless and Bluetooth connectivity



Raspberry Pi Zero

Our lowest-cost single-board computer

Sistemas Operativos: Oficial



Raspberry Pi OS (previously called Raspbian)

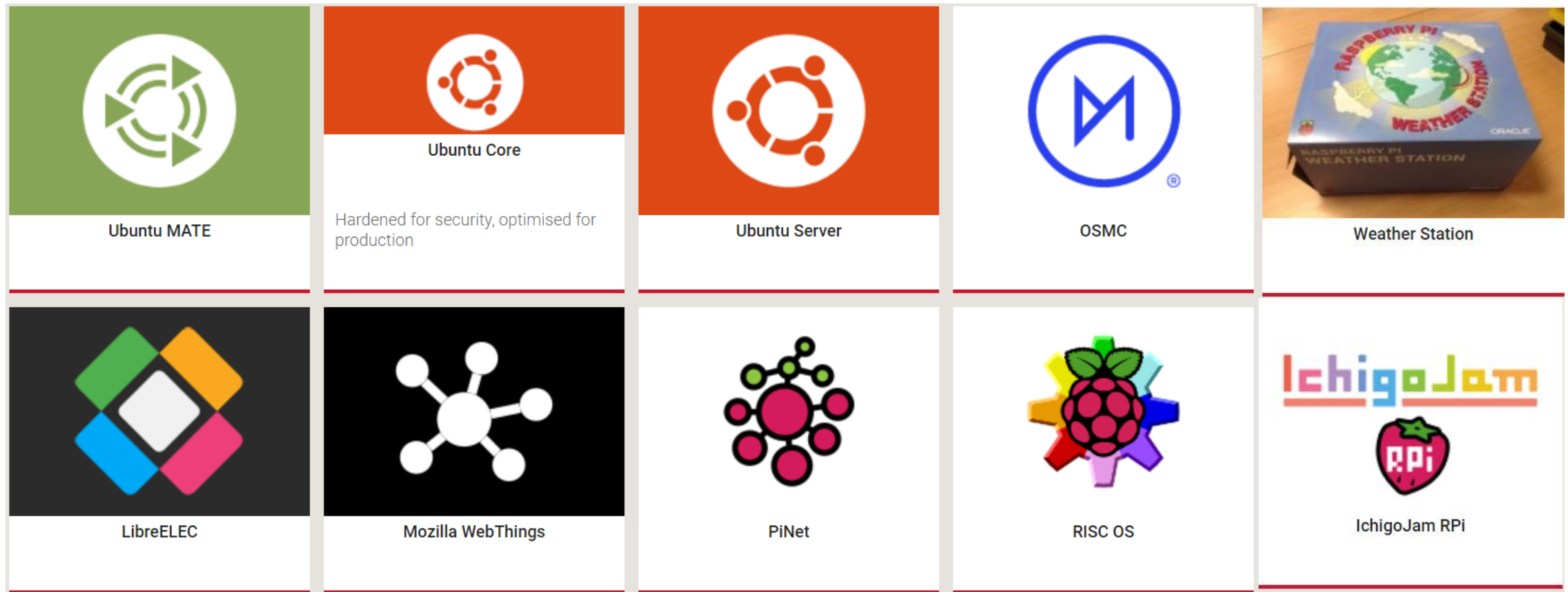
Foundation's official supported operating system



Raspberry Pi Desktop

The Raspberry Pi Desktop OS for PC and Mac - based on Debian Buster

Sistemas Operativos: 3rd Party



OpenElec / LibreElec: mayor rendimiento multimedia, para centro multimedia. Ideal para Kodi.

Info: <https://www.raspberrypi.org/downloads/>

- Distribuciones más populares:



Debian



Mandriva



Ubuntu



Novell/SuSE



Linspire



Red Hat



Fedora Core



Knoppix



Gentoo

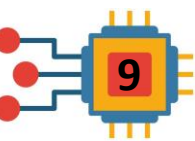
Actividad de clase

- Identificar los components de la Rpi 4B.
- Responder a las preguntas indicadas
- 15 minutos



Procesador

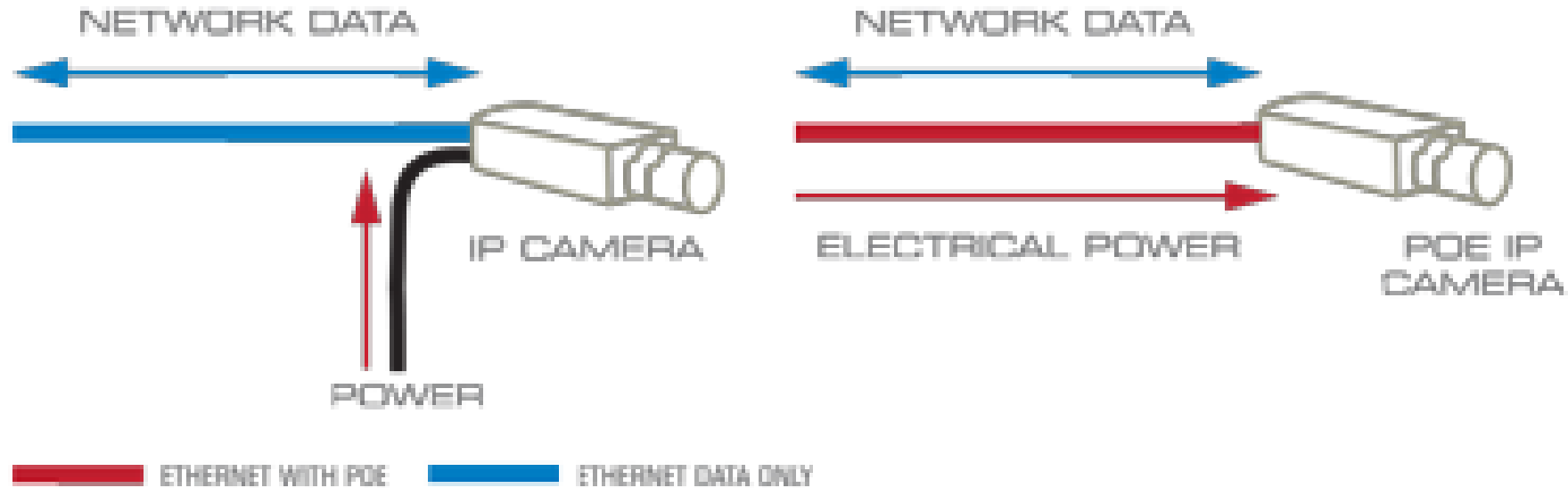
- Eficiente energéticamente: $\sim 5-7$ Watts
- Se puede hacer overclocking: ~ 2.147 GHz
- No necesita ventilación (aunque algún disipador es conveniente).
- Se puede crear una supercomputadora \rightarrow aquí si hace falta refrigeración ;)



Procesador

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz

Power Over Ethernet (PoE)



HATs



<https://magpi.raspberrypi.org/articles/best-raspberry-pi-hats>

Accesorios / Sensores



Grove - Digital Light Sensor



Grove - Light Sensor



Grove - Temperature and Humidity Sensor



Grove - Barometer Sensor



Grove - Dust Sensor



Grove - Gas Sensor



Grove - Temperature Sensor



Grove - Air Quality Sensor

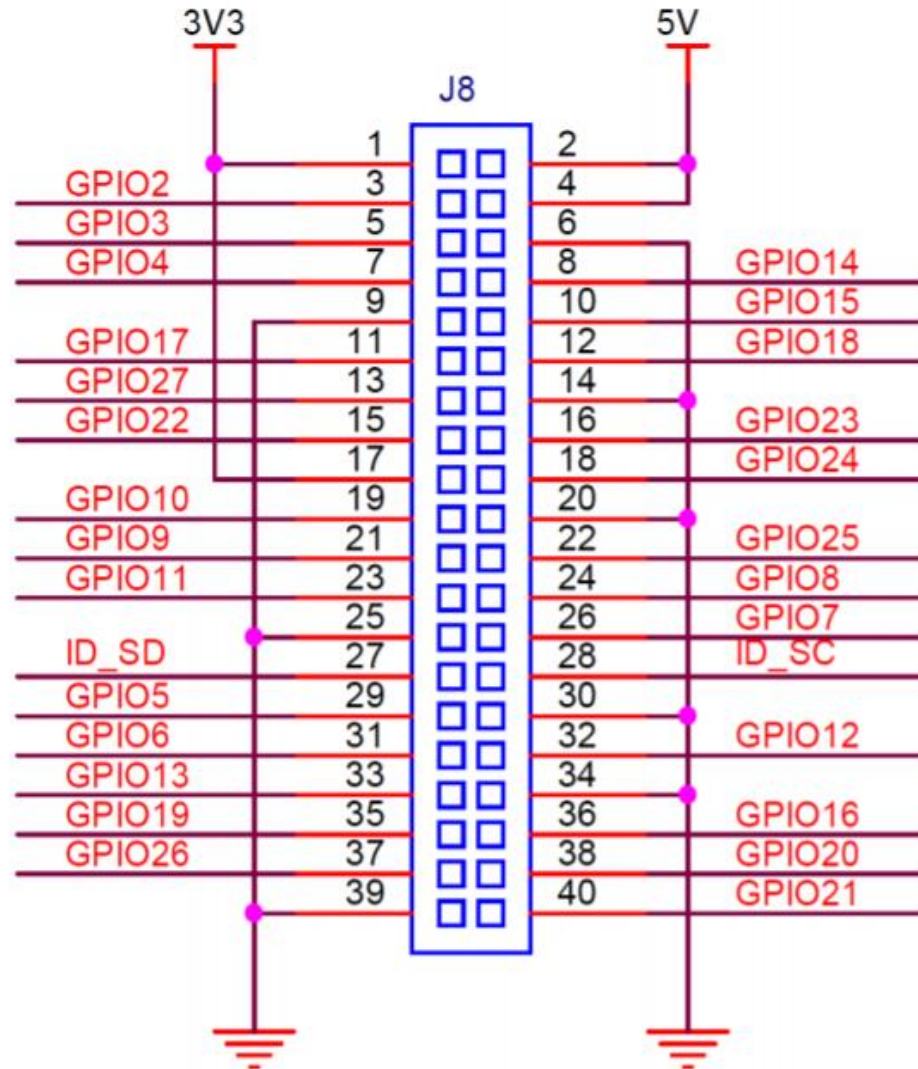


Grove - Temperature and Humidity Sensor Pro



Grove - Gas Sensor(O₂)

GPIO – General Purpose Input/Output



ID_SD and ID_SC PINS:

These pins are reserved for HAT ID EEPROM.

At boot time this I2C interface will be interrogated to look for an EEPROM that identifies the attached board and allows automatic setup of the GPIOs (and optionally, Linux drivers).

DO NOT USE these pins for anything other than attaching an I2C ID EEPROM. Leave unconnected if ID EEPROM not required.

Evaluacion del rendimiento (performance evaluacion)

- <https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b>

Scripting Languages

A scripting language is a computer programming language that is used to specify script files, which are interpreted directly by a runtime environment to perform tasks.

- used to automate the execution of tasks on the RPi, such as system administration, interaction, and even interfacing to electronic components using sysfs.

Scripting Languages

- **Bash scripting** : Is a great choice for short scripts that do not require advanced programming structures. Bash scripts are used extensively in this book for small, well-defined tasks, such as the timing code in the previous section. You can use the Linux commands in your Bash scripts.

Scripting Languages

- **Bash scripting** : Is a great choice for short scripts that do not require advanced programming structures. Bash scripts are used extensively in this book for small, well-defined tasks, such as the timing code in the previous section. You can use the Linux commands in your Bash scripts.
- **Lua** : Is a fast and lightweight scripting language that can be used for embedded applications because of its very small footprint . Lua supports the object-oriented programming (OOP) paradigm (using tables and functions) and dynamic typing, which is discussed shortly.
- **Perl** : Is a great choice for scripts that parse text documents or process streams of data. It enables you to write straightforward scripts and even supports the OOP paradigm.

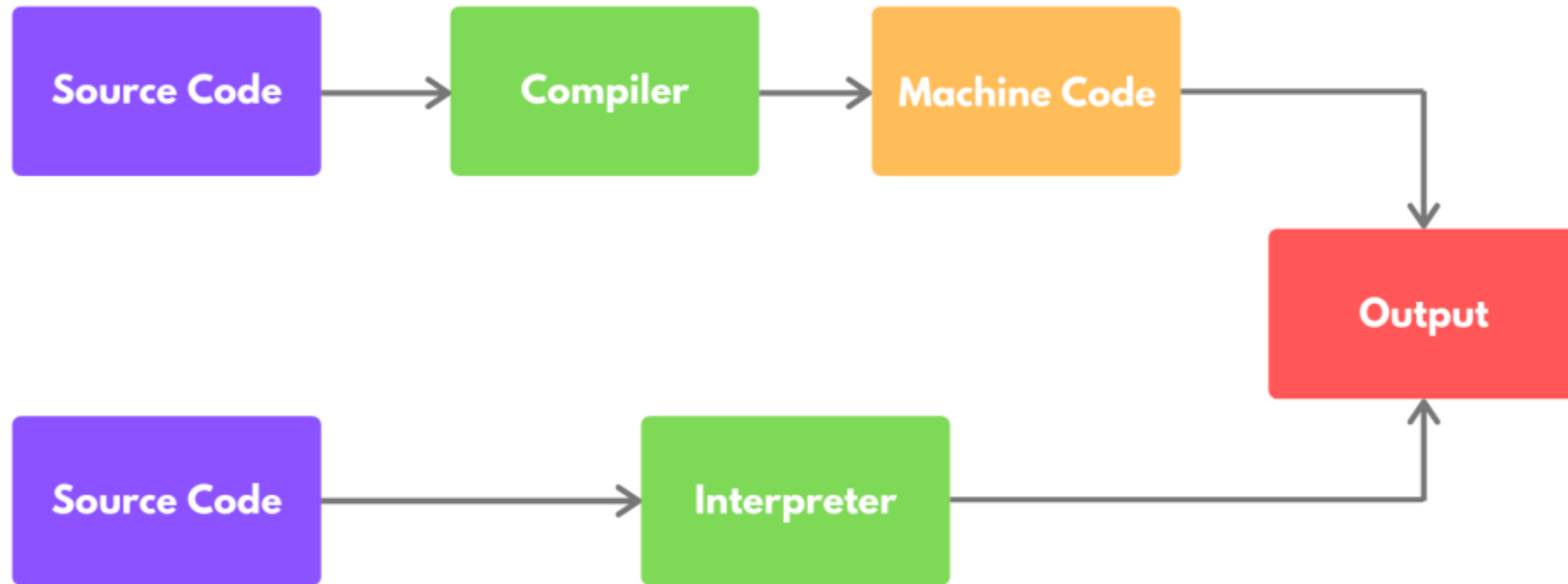
Scripting Languages

- **Python:**

Is great for scripts that need more complex structure and are likely to be built upon or modified in the future.

Python supports the OOP paradigm and dynamic typing.

Compiled VS Scripting



Dynamically-compiled Languages

- Java
- JavaScript
- C/C++

Scripting Languages

Table 5-2: Advantages and Disadvantages of Command Scripting on the RPi

ADVANTAGES	DISADVANTAGES
Perfect for automating Linux system administration tasks that require calls to Linux commands.	Performance is poor for complex numeric or algorithmic tasks.
Easy to modify and adapt to changes. Source code is always present and complex toolchains (see Chapter 7) are not required to make modifications. Generally, nano is the only tool that you need.	Generally, relatively poor/slow programming support for data structures, graphical user interfaces, sockets, threads, etc.
Generally, straightforward programming syntax and structure that is reasonably easy to learn when compared to languages like C++ and Java.	Generally, poor support for complex applications involving multiple, user-developed modules or components.
Generally, quick turnaround in coding solutions by occasional programmers or for prototyping.	Code is in the open. Direct access to view your code can be an intellectual property or a security concern.
	Lack of development tools (e.g., refactoring).

Table 5-1: Numeric Computation Time for 5,000,000 Iterations of the *n*-Body Algorithm on Raspbian (Jessie Minimal Image)

VALUE	TYPE	RPi 3 at 1.2 GHZ ¹	RPi 2 at 1 GHZ ²	RPi B+ at 1 GHZ ³	64-BIT i7 PC ⁴
C/C++	Compiled	1.00 × (6.5s)	1.00 × (9.3s)	1.00 × (10.0s)	1.00 × (0.61s)
C++11	Compiled	1.06 × (6.9s)	0.69 × (6.4s)	0.70 × (7.03s)	0.95 × (0.58s)
Haskell	Compiled	1.16 × (7.6s)	1.17 × (10.8s)	1.07 × (10.8s)	1.15 × (0.70s)
Java ⁵	JIT	1.52 × (9.94s)	1.45 × (13.4s)	2.29 × (23.0s)	1.36 × (0.83s)
Mono C#	JIT	2.72 × (17.8s)	2.47 × (22.9s)	3.62 × (36.4s)	2.16 × (1.32s)
Cython ⁶	Compiled	2.74 × (17.9s)	2.67 × (24.8s)	2.80 × (28.0s)	1.26 × (0.77s)
Node.js ⁷	JIT	2.76 × (18.1s)	6.23 × (57.7s)	50.1 × (503s)	6.54 × (3.99s)
Lua	Interpreted	20.2 × (132s)	21.2 × (197s)	25.7 × (258s)	34.3 × (20.9s)
Cython	Compiled	64.2 × (420s)	66.6 × (618s)	163 × (1633s)	58.0 × (34.4s)
Perl	Interpreted	92.6 × (601s)	81.5 × (756s)	171 × (1716s)	82.0 × (50.0s)
Python	Interpreted	94.1 × (616s)	89.9 × (834s)	198 × (1992s)	89.7 × (54.7s)
Ruby	Interpreted	147 × (962s)	140 × (1298s)	265 × (2662s)	47.4 × (28.9s)

Programacion de la Raspberry Pi

- Control de los GPIOs mediante la ventana de comandos:

https://elinux.org/RPi_GPIO_Code_Samples#Shell

Programacion de la Raspberry Pi

- Control de los GPIOs mediante Asembly language:

[http://www.science.smith.edu/dftwiki/index.php/Tutorial: Assembly Language with the Raspberry Pi](http://www.science.smith.edu/dftwiki/index.php/Tutorial:_Assembly_Language_with_the_Raspberry_Pi)

Link de interest

- TEDTalk: Creador de la Rpi:

<https://www.youtube.com/watch?v=58MhVYjjdTM>

Formas de romper la RaspberryPi

Nunca tocar la placa con las manos mientras esté alimentada: riesgo de CORTOCIRCUITO

No la desenchufes directamente, mejor cierra ordenadamente el SO (shutdown now)

No pongas la placa encima de superficies metálicas, así evitaremos un cortocircuito. Usa patas de goma o mejor una carcasa

No conectes circuitos que drenen o aporten mucha corriente por las GPIO, el máximo es 2 2-3 mA (en comparación Arduino permite hasta 40 mA, pudiendo alimentar circuitos por las GPIO, RPi no).
GPIO máximo 3.3V

Siempre repasa 2 o 3 veces la numeración de los pines GPIO: el uso de un PIN incorrecto puede quemar la placa

Mucho cuidado cuando trabajes en modo superusuario/root ya que puedes desconfigurar el sistema

Ante cualquier duda pide ayuda al profesor o consulta Internet en más de una fuente para verificar que la solución es correcta.
Extra-cuidado: los problemas de software son reversibles, los de hardware no.

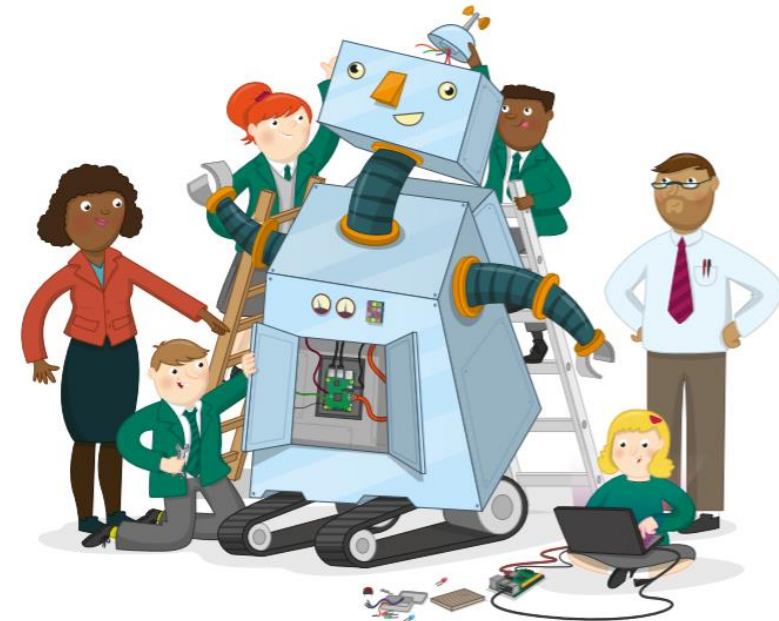


Illustration: www.raspberrypi.org