UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA
Y DISEÑO

15765 SISTEMAS EMPOTRADOS

PRESENTA: DR. EVERARDO INZUNZA GONZÁLEZ
Profesor – Investigador
Email:einzunza@uabc.edu.mx
Tel. 175 0744 Ext. 64321

Ensenada Baja California; a febrero del 2016.

COMPETENCIA DEL CURSO

Diseñar e implementar sistemas empotrados utilizando sistemas en tiempo real comerciales para la solución de problemas en forma analítica y eficiente, ordenada y disciplinada.

CONTENIDO DEL CURSO

I. Introducción a la arquitectura de sistemas de alta escala.

II. El Kernel en tiempo real y técnicas de interconexión.

III. Programación en C para microprocesadores empotrados de 32 Bits.

IV. Programación en Python para microprocesadores empotrados de 32 bits.

V. Programación en Octave para microprocesadores empotrados de 32 bits.

VI. Tratamiento digital de señales mediante procesadores empotrados de 32 bits.

VII. Procesamiento digital de imágenes con procesadores empotrados de 32 bits.

Evidencia del Desempeño (Proyecto Final)

- **Implementar un sistema embebido utilizando una plataforma de tiempo real** donde el alumno demuestre las habilidades adquiridas en el curso para la solución de un problema de medición, procesamiento de datos, así como entradas y salidas de datos (manejo de teclado, pantallas LCD/Touch, cámara digital).

Ejemplos de proyectos **(EMPLEANDO EL INTERNET DE LAS COSAS)**

Implementación de un GPS empleando un procesador embebido de 32 bits

Detector de sismos/ con monitoreo por internet

Automatización/alarma de una casa con monitoreo por internet

Estación climatológica/ con monitoreo por internet

Robot manipulador/controlado remotamente empleando el internet

Instrumento para medición de tres variables físico/químicas con procesamiento de los datos/internet

Conocimientos previos

- Diseño digital
- Microcontroladores
- Lenguaje ensamblador
- Procesamiento digital de señales
- Programación en C, C++, MatLab

BIBLIOGRAFÍA

Embedded Systems: Introduction to ARM® Cortex™-M Microcontrollers
Jonathan W Valvano
CreateSpace Independent Publishing Platform
Año 2012, Quinta Edición

Embedded Systems Architecture
Tammy Noergaard
Newnes
Año 2012, Segunda Edición

Embedded Systems Glossary
Michael Barr
Netrino Technical Library
Año 2007

Embedded systems design
Heath, Steve
EDN series for design engineers
2da edición
Año 2003

BIBLIOGRAFÍA

Raspberry Pi, User Guide
Eben Upton, Gareth Halfacree
Wiley
ISBN: 978-1-118-92166-1

Programming the Raspberry Pi, Getting Started with Python
Simon Monk
Mc-Graw Hill
ISBN: 978-0-07-180783-8

Raspberry Pi, Projects for the Evil Genius
Donald Norris
Mc-Graw Hill
ISBN: 978-0-07-182158-9

Adventures in Raspberry Pi
Carrie Anne Philbin
Wiley
ISBN: 978-1-119-04602-8

FORMA DE EVALUAR SISTEMAS EMPOTRADOS

- Exámenes teóricos y prácticos (4 parciales) 50%.
  (Promedio min. 60 para exentar ordinario).
- Laboratorio 20%
- Proyecto 20%
- Tareas 10%

➢ **REQUISITOS PARA ACREDITAR EL CURSO**

1. Haber acreditado el laboratorio.
2. Cumplir con las asistencias establecidas en el reglamento.
3. Haber entregado el proyecto Final. (Equipos de 3 personas máx.)

Nota: El alumno que no cumpla con alguno de los requisitos anteriores no tendrá derecho a calificación Ordinaria.

Dr. Everardo Inzunza G.

Nombre y firma del representante de grupo

Firma del profesor

FORMA DE EVALUAR LABORATORIO DE SIST. EMPOTRADOS

- **REQUISITOS PARA ACREDITAR EL LABORATORIO**

1. Cumplir con el 80% de asistencia.
2. Entregar el 90% de las prácticas (Experimentos y reportes).
3. Entregar proyecto final.
4. Aprobar examen práctico

Dr. Everardo Inzunza G.

Nombre y firma del representante de grupo

Firma del profesor

ÍNDICE DE PRESENTACIÓN

- INTRODUCCIÓN A LA ARQUITECTURA DE SISTEMAS DE ALTA ESCALA

❑ INTRODUCCIÓN

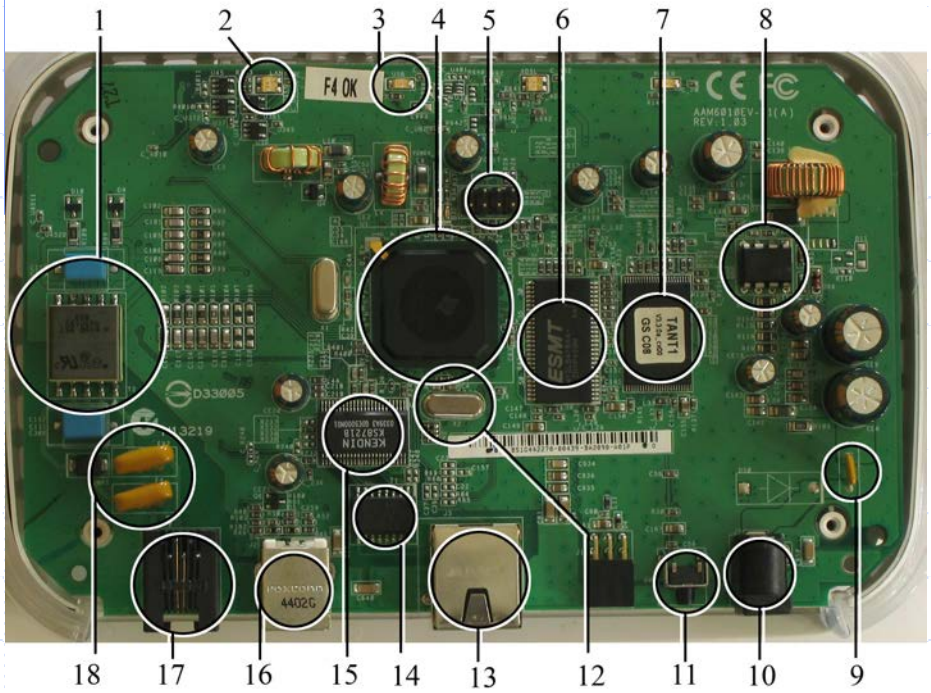
➢SISTEMA EMPOTRADO/EMBEBIDO

Un sistema embebido (anglicismo de embedded) o **empotrado** (integrado, incrustado) **es un sistema de computación** diseñado para realizar una o varias tareas dedicadas, usualmente en un sistema de computación en tiempo real [Michael Barr, 2007] y [Heath Steve, 2003].

Al contrario de lo que ocurre con las computadoras de propósito general (por ejemplo una PC) que están diseñadas para cubrir un amplio rango de necesidades, **los sistemas embebidos se diseñan para cubrir necesidades específicas**.

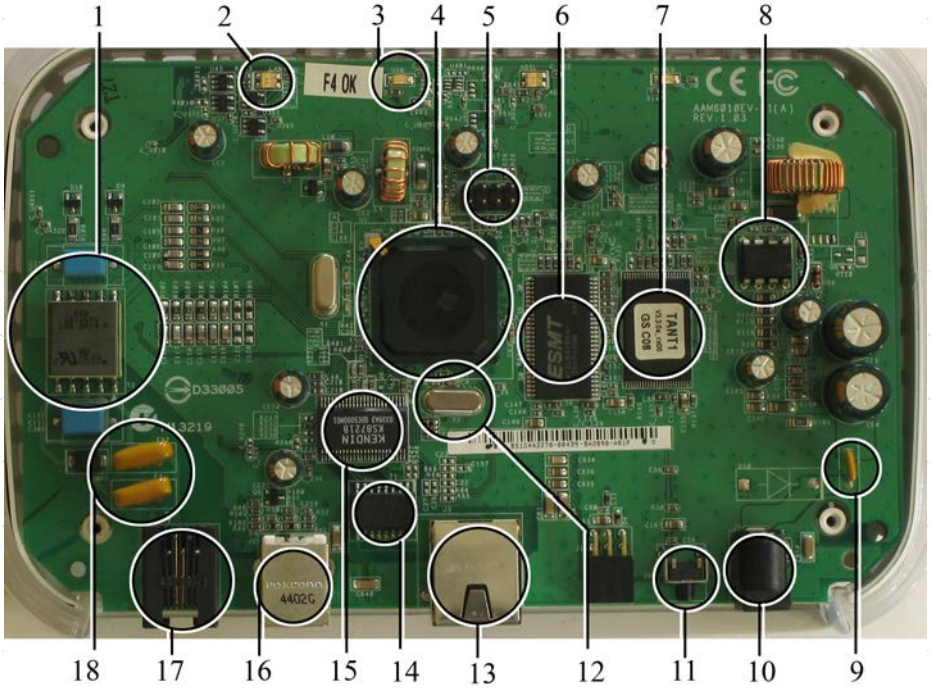
Sistemas embebidos

En un sistema embebido la mayoría de los componentes se encuentran **incluidos en una placa base** (la tarjeta de video, audio, módem, etc.) y muchas veces los dispositivos resultantes no tienen el aspecto de lo que se suele asociar a una computadora personal.



Un ejemplo de un sistema embebido. Las partes marcadas incluyen un **microprocesador** (4), **RAM** (6), y una **memoria flash** (7).

Detalles que incluye el ejemplo de sistema embebidos



Esta imagen muestra la parte interna de un Modem/Router Netgear DG632 ADSL. Este actua como un ruteador entre un puerto ethernet y una conexión a internet de banda ancha tipo ADSL. Las partes etiquetadas son:

1. Telephone decoupling electronics (for ADSL). **2**. Multicolour LED (displaying network status). **3**. Single colour LED (displaying USB status). **4.** Main processor, a TNETD7300GDU, a member of Texas Instruments AR7 product line. **5.** JTAG (Joint Test Action Group) test and programming port. **6.** RAM, a single ESMT M12L64164A 8 MB chip. **7.** Flash memory, obscured by sticker. **8**. Power supply regulator. **9.** Main power supply fuse. **10.** Power connector. **11.** Reset button. **12.** Quartz crystal. **13.** Ethernet port. **14.** Ethernet transformer, Delta LF8505. **15.** KS8721B ethernet PHY transmitter receiver. **16.** USB port. **17.** Telephone (RJ11) port. **18.** Telephone connector fuses.

Introducción a los sistema embebidos

Algunos ejemplos de sistemas embebidos podrían ser dispositivos como un taxímetro, un sistema de control de acceso, la electrónica que controla una máquina expendedora o el sistema de control de una fotocopiadora entre otras múltiples aplicaciones.

Por lo general los **sistemas embebidos se pueden programar** directamente en el **lenguaje ensamblador** del microcontrolador o microprocesador incorporado sobre el mismo, o también, utilizando los compiladores específicos, pueden utilizarse **lenguajes como C o C++**; en algunos casos, **cuando el tiempo de respuesta de la aplicación no es un factor crítico**, también pueden usarse lenguajes interpretados como **JAVA, PYTHON, OCTAVE.**

Introducción a los sistema embebidos

Debido a que los sistemas embebidos se pueden fabricar por decenas de millares o por millones de unidades, una de los principales objetivos es reducir los costos.

Los **sistemas embebidos** suelen usar un **procesador relativamente pequeño** y una memoria pequeña para ello.

Los primeros equipos embebidos que se desarrollaron fueron elaborados por IBM en los años 80's.

Un programa de sistema embebido normalmente se enfrenta a **tareas de procesamiento en tiempo real**.

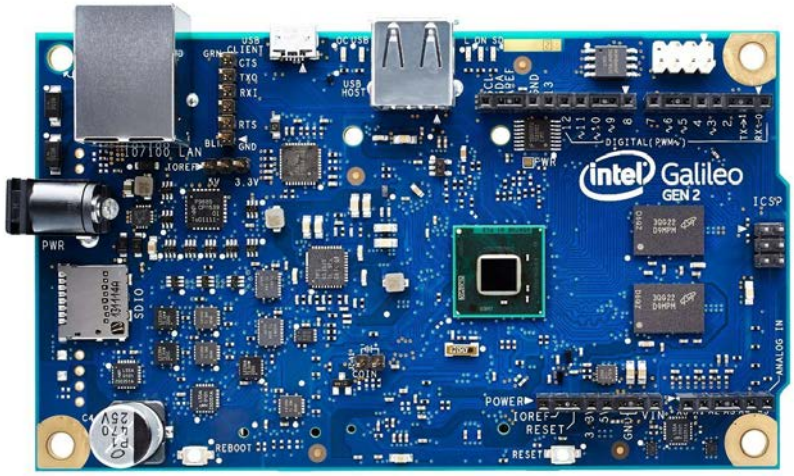
Introducción a los sistema embebidos

- Componentes de un sistema embebido

- En la parte central se encuentra un **Microprocesador**, o bien un **microcontrolador** ó un **DSP** (Digital Signal Processor) por lo general incluyen una **memoria RAM interna**.
- **Puertos de Comunicación** RS-232, Ethernet, SPI, I^2C, CAN, USB, HDMI, Wi Fi, GSM, GPRS, etc.
- En ocasiones puede contener una pantalla gráfica, táctil, LCD, alfanumérica, etc.
- **Módulo de Entradas/Salida**s analógicas y digitales.
- Módulo de **reloj.**
- El **subsistema de energía.**

Introducción a los sistema embebidos

- Ejemplos de sistemas embebidos (INTEL Galileo)



Intel Galileo Gen 2 Board Single GALILEO2

La tarjeta de desarrollo Intel® Galileo Gen 2 contiene un **procesador de aplicaciones Intel®** Quark **SoC X1000** (sistema en un chip (SoC)) Intel® Pentium® de 32-bit.

Es la **primera tarjeta con arquitectura Intel®** diseñada para ser compatible con software y hardware con cubiertas diseñadas **para el Arduino** UNO R3.

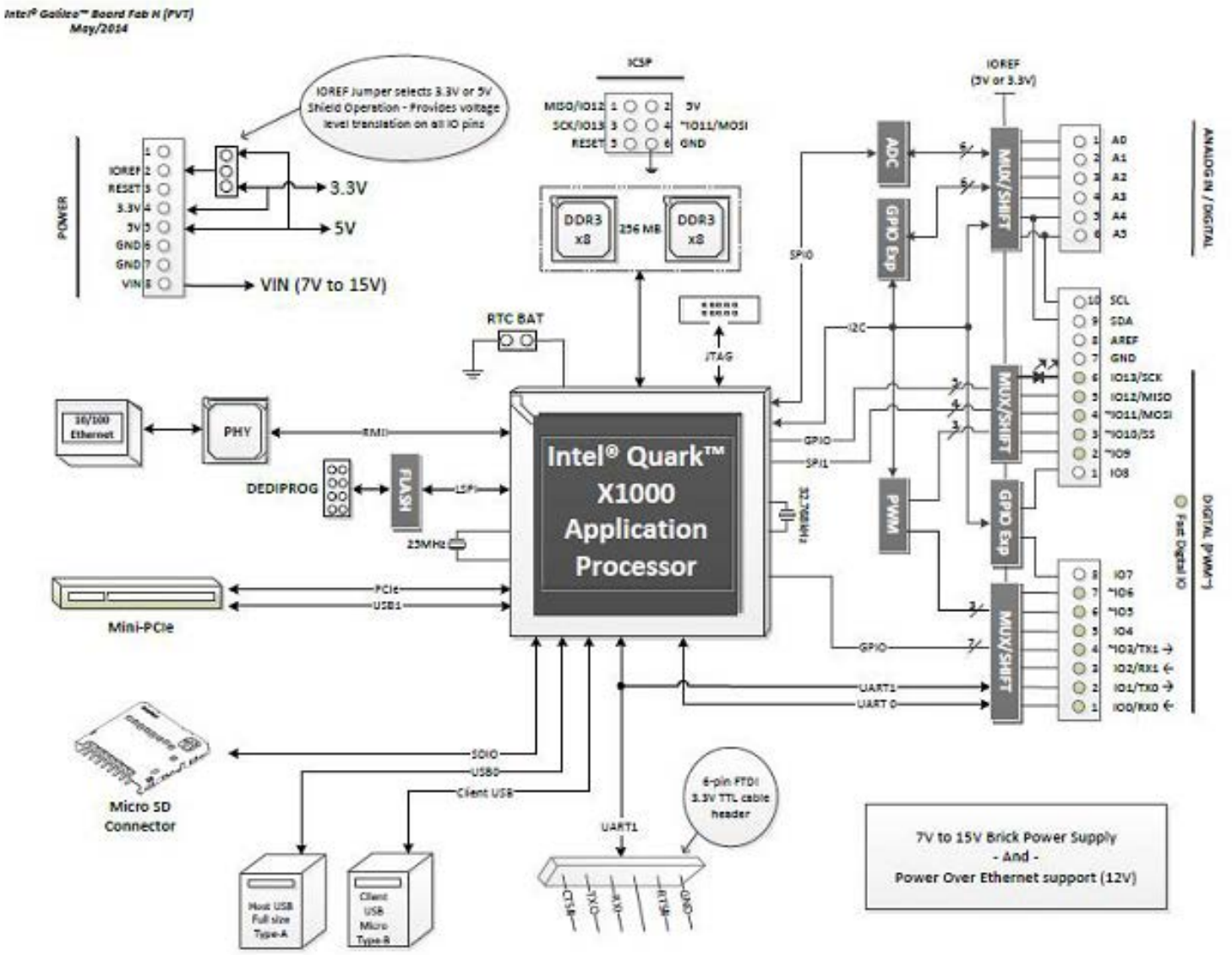
Esta plataforma proporciona la facilidad del desarrollo de la arquitectura Intel a través de soporte para **Microsoft Windows*, Mac OS*** y los sistemas operativos de host, **Linux***.

También trae la simplicidad del **software de entorno de desarrollo integrado (IDE) Arduino.**

Además contiene un ranura PCI Express, puerto Ethernet 100Mb, ranura Micro –SD, puerto UART, Puerto USB, NOR Flash de 8 MB y salida PWM.

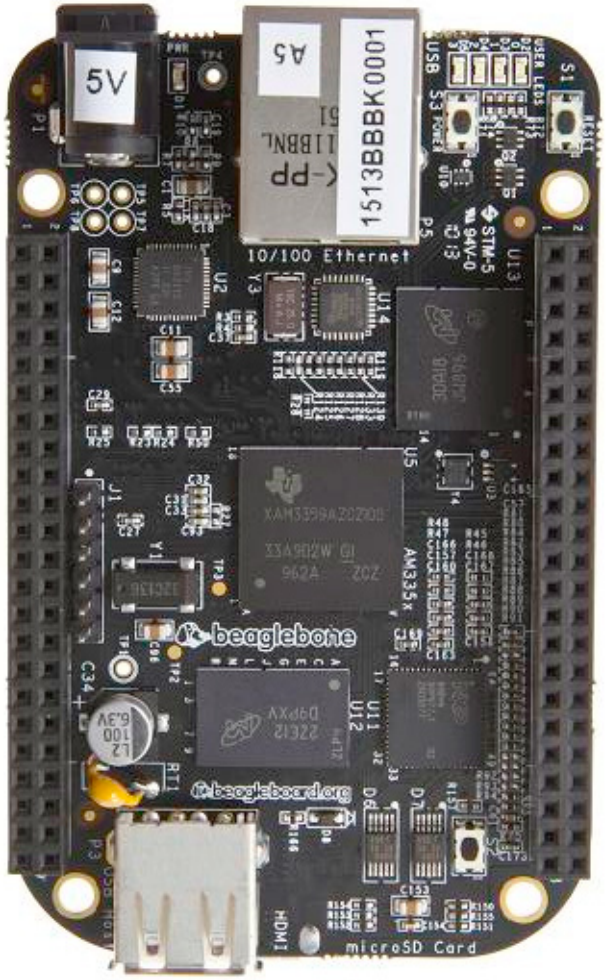
Introducción a los sistema embebidos

- Arquitectura de la tarjeta INTEL Galileo



Introducción a los sistema embebidos

- Ejemplos de sistemas embebidos – Beagle board ó Beaglebone



Procesador: AM335x 1GHz ARM® Cortex-A8
512MB DDR3 RAM
4GB 8-bit eMMC on-board **flash storage**
3D graphics accelerator
NEON floating-point accelerator
2x PRU 32-bit microcontrollers

Conectividad
USB client for power & communications
USB host
Ethernet
HDMI
2x 46 pin headers

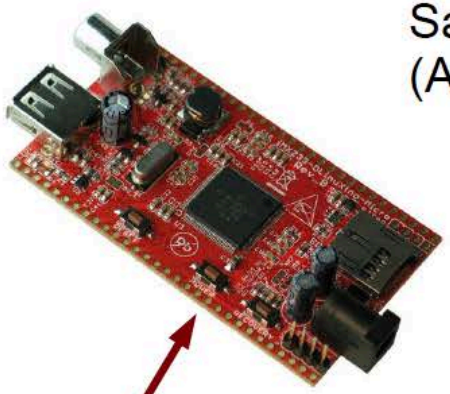
Compatibilidad de Software
Linux Debian
Android
Ubuntu
Cloud9 IDE on Node.js w/ BoneScript library
C, C++

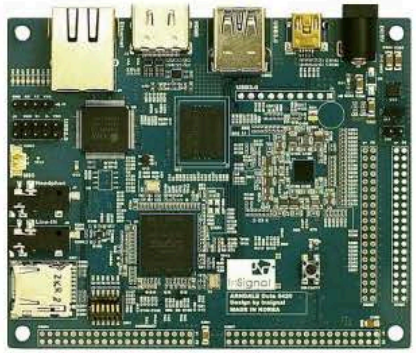
Introducción a los sistema embebidos

- Ejemplos de sistemas embebidos – Otras opciones interesantes (Con procesadores ARM)

Wandboard
Freescale iMX6

Arndale
Samsung Exynos Octa Core!!!
(ARM Cortex-A15)





OlinuXino
iMX233 (ARM9)

Utilite
Freescale iMX6

Rikomagic MK802
Rockchip RK3066

En placas: OlinuXino, o-doidx, arndale, wandboard, etc.

En caja: Cubie board, Rikomagic Cubox-I, Utilite, etc.

Introducción a los sistema embebidos

- Ejemplos de sistemas embebidos – Otras opciones interesantes con gran rendimiento computacional

SBC con Intel Atom Z530 a 1,6 GHz

TAREA #1

- Investigar la historia de Raspberry Pi
  - Fecha de pre-lanzamiento y lanzamiento a la venta
  - Inventores
  - País de orígen
  - Modelos fabricados y sus específicaciones técnicas
  - Sistemas operativos que pueden ejecutar

Introducción a los sistema embebidos

- Ejemplos de sistemas embebidos – Raspberry Pi 2 Modelo B

Puede ejecutar diferentes distribuciones de LINUX, así como WINDOWS 10.

Microprocesador ARM Cortex-A7
CPU 900MHz **Quad-Core**
Opcional **Overclocking a 1.1 GHz**
Memoria **RAM 1GB**

Tambien contiene:
4 puertos USB
40 pines GPIO
Puerto HDMI completo
Puerto Ethernet
Salida de audio y video compuesto combinado en Jack 3.5mm
Interface para cámara (CSI)
Interface para Display(DSI)
Ranura para Micro SD
Procesador gráfico VideoCore IV 3D

Introducción a los sistema embebidos

- Raspberry Pi 2

Es una computadora de placa reducida del tamaño de una tarjeta de crédito (System on a Chip) de bajo costo, desarrollada en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Utiliza una memoria Micro –SD como dispositivo de almacenamiento masivo.

Puede ejecutar diferentes distribuciones de LINUX para procesadores ARM

Raspbian – Derivado de Debian Wheezy

RISC OS 5

Arch Linux ARM – Derivado de Arch Linux

Pidora – Derivado de Fedora

Ubuntu Mate y Ubuntu Snappy Core

OpenELEC y OSMC – Linux pra Xbox

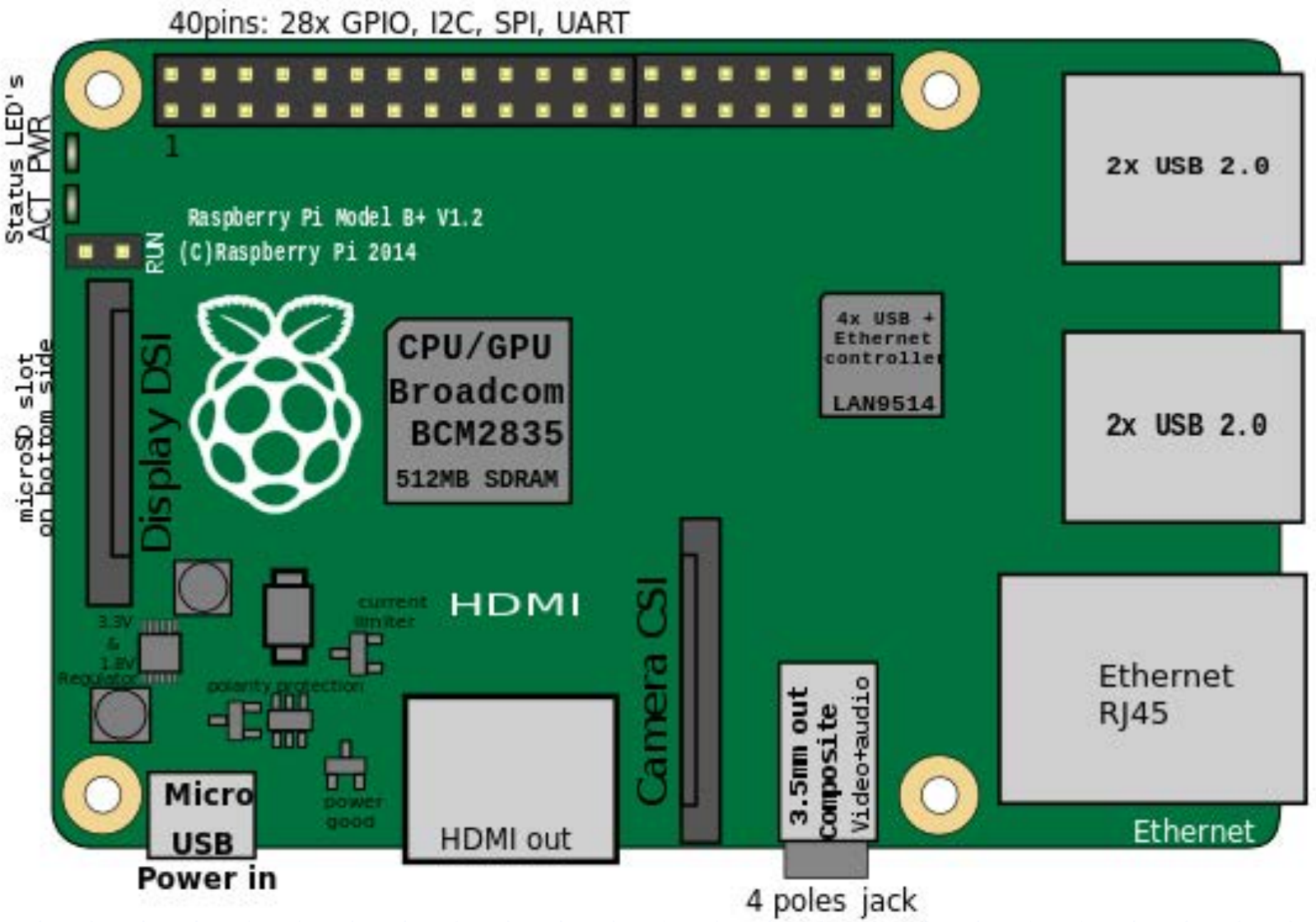
Microsoft **Windows 10**

Ejecuta el **lenguaje C** de forma **nativa**.

Promueve principalmente el **lenguaje de programación** (interprete) **Python**.

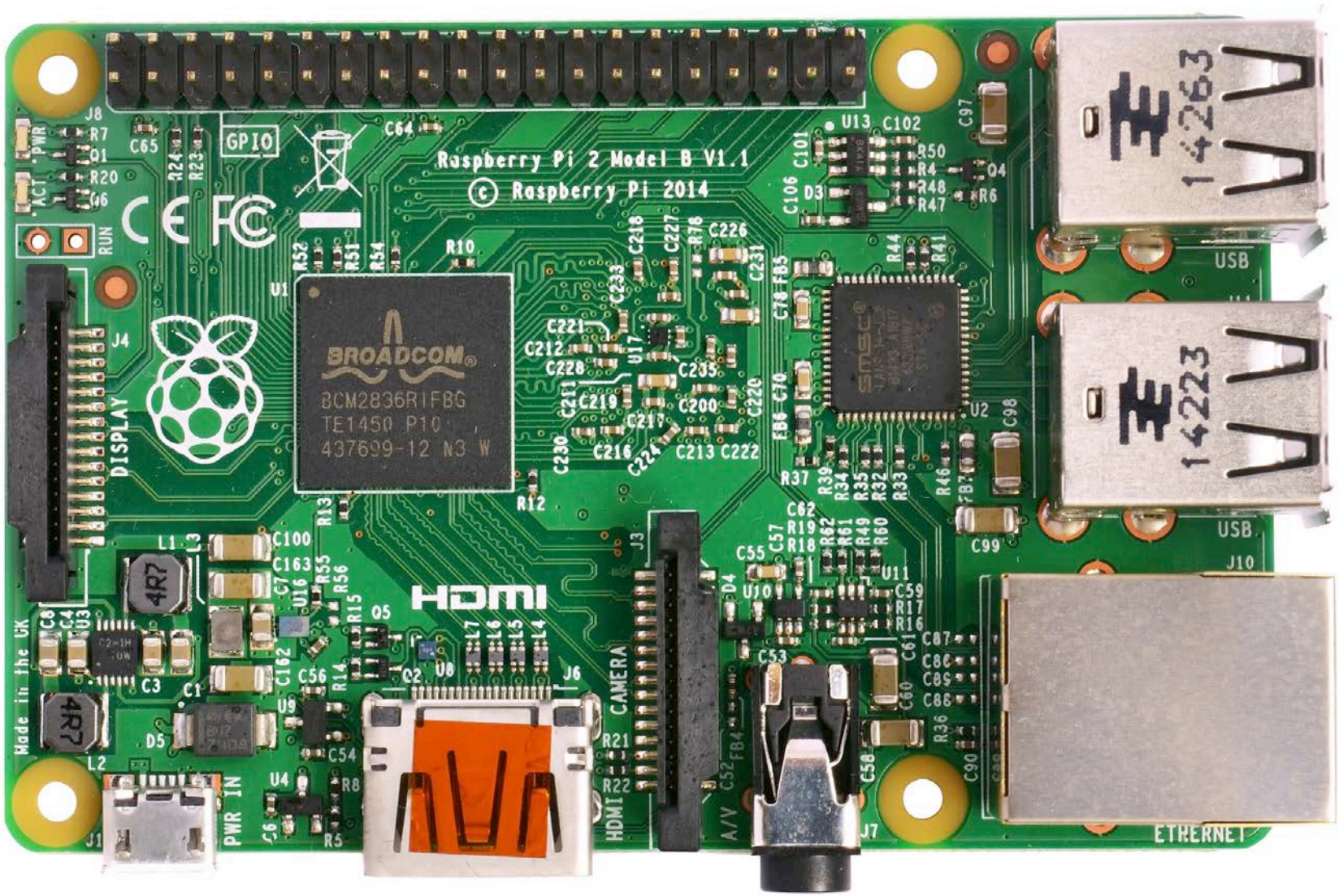
Introducción a los sistema embebidos

- Arquitectura de Raspberry Pi 2 Modelo B



Introducción a los sistema embebidos

- Arquitectura de Raspberry Pi 2 Modelo B



Introducción a los sistema embebidos

- Arquitectura de Raspberry Pi 2 Modelo B

Organización del Hardware

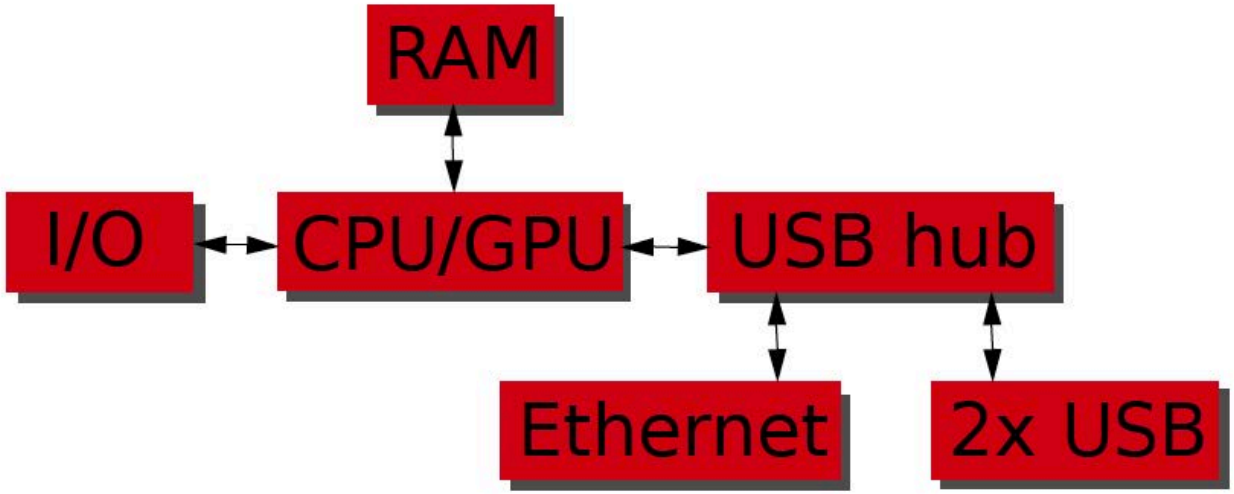


Diagrama de bloques del modelo B, en el Modelo A el puerto USB está conectado directamente al SoC

TAREA #2

- INVESTIGAR LA EVOLUCIÓN DE LOS MICROPROCESADORES **ARM**
  - Sus orígenes e historia.
  - Cual ha sido la clave del éxito de estos procesadores?
  - Principales compañías que utilizan microprocesadores ARM
  - Sistemas opertivos que ejecutan los microprocesadores ARM
  - Lenguajes de programación
  - Tecnologías que emplean los ARM

Introducción a los sistema embebidos

- Arquitectura de SoC (System on Chip)

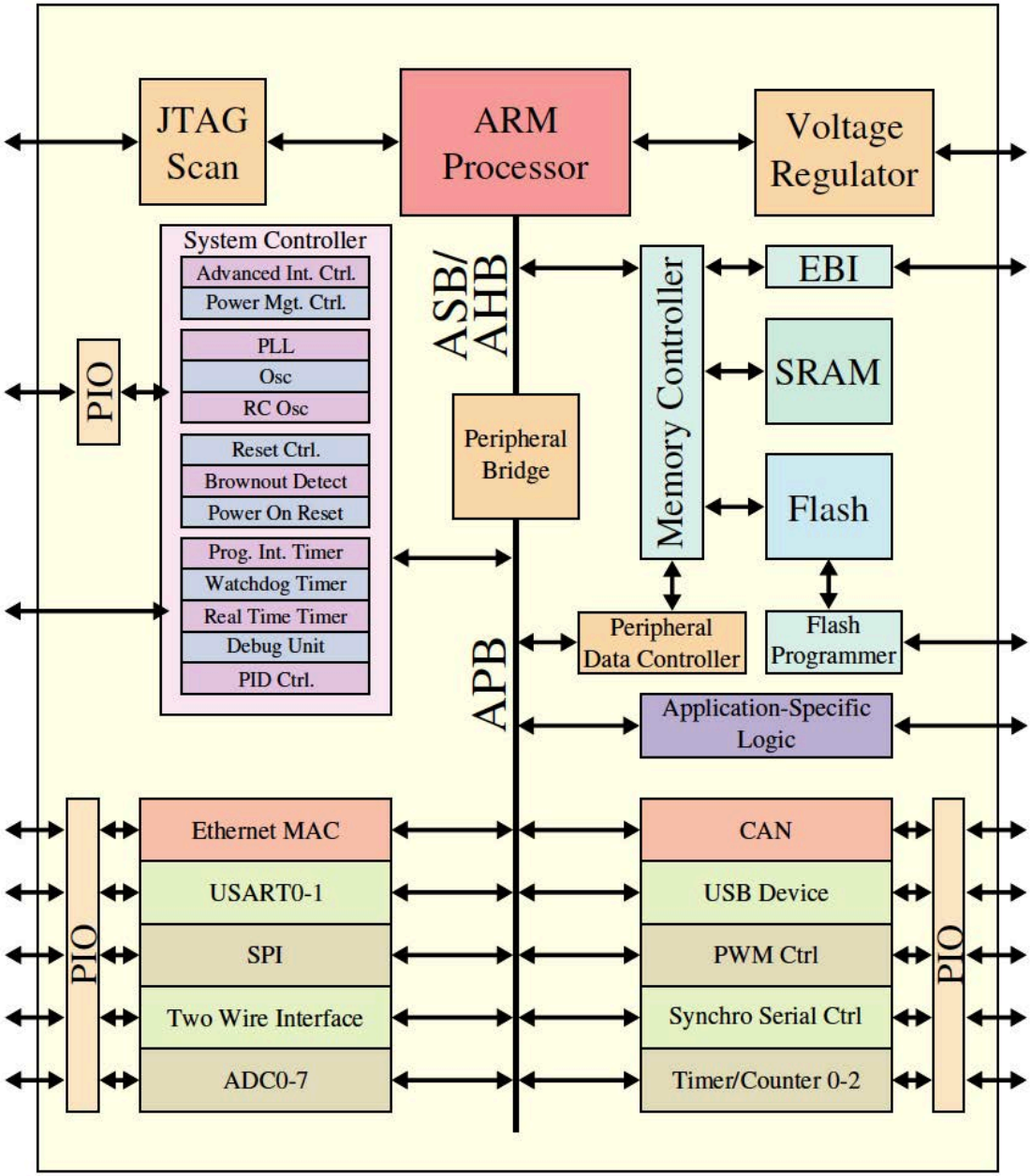
Describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos componentes de una computadora o cualquier otro **sistema informático** o **electrónico** en un único circuito integrado o chip.

El diseño de estos sistemas puede estar basado en circuitos de señal digital, señal analógica, o incluso de señal mixta (tanto analógica como digital), y a menudo módulos o sistemas de radio-frecuencia (módulos de comunicación inalámbrica: Wi-Fi, Bluetooth, etc.).

Un ámbito común de **aplicación de la tecnología SoC son los sistemas embebidos.**

Introducción a los sistema embebidos

- Arquitectura de SoC (System on a Chip)



Nomenclatura:

Advanced Microprocessor Bus Architecture (AMBA)

AMBA High-Speed Bus (AHB)

Advanced System Bus (ASB)

Arquitectura de un SoC basada en Un Microprocesador ARM

Introducción a los sistema embebidos

- Arquitectura de SoC (System on a Chip)

Un SoC estándar está constituído por:

- **Un microcontrolador , microprocesador o núcleo DSP .** Algunos SoCs – llamados MultiProcessor System-on-Chip (MPSoC) – son construídos con microprocesadores dotados de varios núcleos o bien más de un microprocesador.

- **Módulos de memoria digital** incluyendo parte o todos los tipos de memoria a continuación listados: ROM (memoria de sólo lectura), RAM (memoria de acceso aleatorio), EEPROM (memoria de sólo lectura programable y borrable eléctricamente) y Flash (memorias NAND de acceso muy rápido en comparación con los tradicionales -todavía hoy muy usados- soportes magnéticos).

Introducción a los sistema embebidos

- Arquitectura de SoC (System on a Chip)

Un SoC estándar está constituído por:

- **Generadores de frecuencia fija o señal de reloj**, como por ejemplo osciladores y/o lazos de amarre de fase o Phase Locked Loops (PLL por sus siglas en inglés).

- **Componentes periféricos** como contadores, temporizadores, relojes de tiempo real y generadores Power-on Reset (o PoR por sus siglas en ingles, dispositivos que reajustan un sistema electrónico al recibir una señal digital determinada, permitiéndole arrancar desde un estado conocido).

- **Controladores de comunicación** con interfaces normalmente estándar como USB, IEEE 1394/Firewire, Ethernet, UART, SPI, I^2C, Wi Fi, GPRS, etc.

Introducción a los sistema embebidos

- Arquitectura de SoC (System on a Chip)

Un SoC estándar está constituído por:

- **Controladores de interfaces analógicas**, incluyendo convertidores analógico a digital (ADC) y digital a analógico (DAC).
- **Subsistema de energía - Reguladores de voltaje** y circuitos de gestión eficaz de la energía.

Todos estos módulos están unidos de acuerdo a estándares industriales para la interconexión de buses, aunque pueden seguir directrices de carácter propietario como por ejemplo la especificación AMBA, arquitectura de bus diseñada por ARM .

Los controladores DMA (Direct Memory Access), dirigen la información entre interfaces externas y la memoria principal, evitando el paso innecesario de ésta a través del microprocesador para evitar que se incremente el volumen de trabajo del SoC.

MICROPROCESADORES ARM

ARM es una arquitectura RISC (Reduced Instruction Set Computer = Computadora con Conjunto Reducido de Instrucciones) de 32 bits y recientemente con la llegada de su versión V8-A también 64 Bits desarrollada por ARM Holdings.

Se llamó Advanced RISC Machine , y anteriormente Acorn RISC Machine .

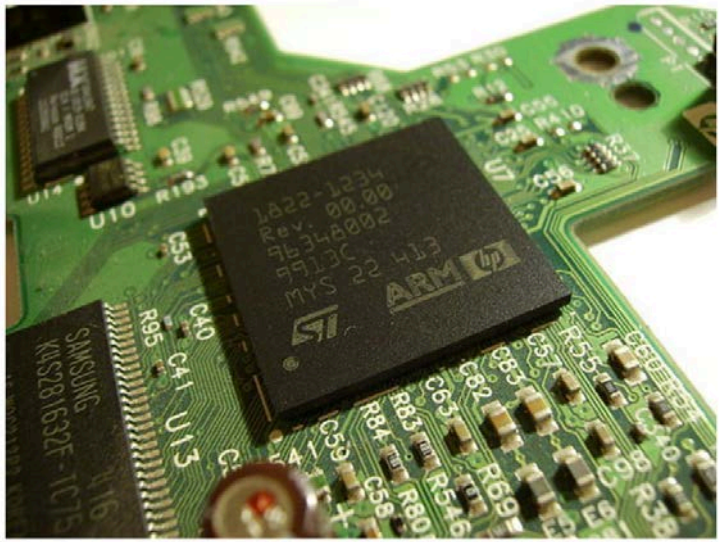
La arquitectura ARM tiene conjunto de instrucciones de 32 y 64 bits más ampliamente utilizado en unidades producidas.

Inicialmente los fabricaban Acorn Computers **para uso en computadoras personales.**

MICROPROCESADORES ARM

Los primeros productos basados en ARM eran los Acorn Archimedes, lanzados en 1987.

La relativa simplicidad de los procesadores ARM los hace ideales para aplicaciones de baja potencia. Como resultado, se han convertido en dominante en el mercado de la electrónica móvil e integrada, encarnados en microprocesadores y microcontroladores pequeños, de bajo consumo y relativamente bajo costo.



Procesador ARM en una impresora HP

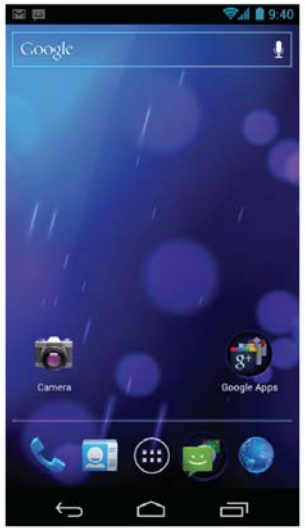
MICROPROCESADORES ARM

En 2005, alrededor del 98% de los *más de mil millones de teléfonos móviles* vendidos **utilizaban al menos un procesador ARM**.

Desde 2009, los procesadores ARM son aproximadamente el 90% de todos los procesadores RISC de 32 bits integrados.

MICROPROCESADORES ARM

Se utilizan generalmente en la electrónica de consumo, incluyendo PDA, tabletas, teléfonos móviles, Teléfonos inteligentes, Relojes Inteligentes, videoconsolas portátiles, calculadoras, reproductores digitales de música y medios (fotos, vídeos, etc.), y periféricos de ordenador como discos duros y routers .



Dispositivo androide

Mini tablet

iPhone & iWatch

MICROPROCESADORES ARM

- Advanced RISC Machines Holdings Limited (ARM) diseña y licencia procesadores.

- **Los clientes adquieren estas licencias** en forma de Intellectual Property (IP) junto con herramientas adicionales.

- **Los clientes tiene derecho adaptar** y complementar las IP

- Las adaptaciones son fabricadas:
  - Por el mismo cliente
  - Encargadas a terceros
  - Relicenciadas

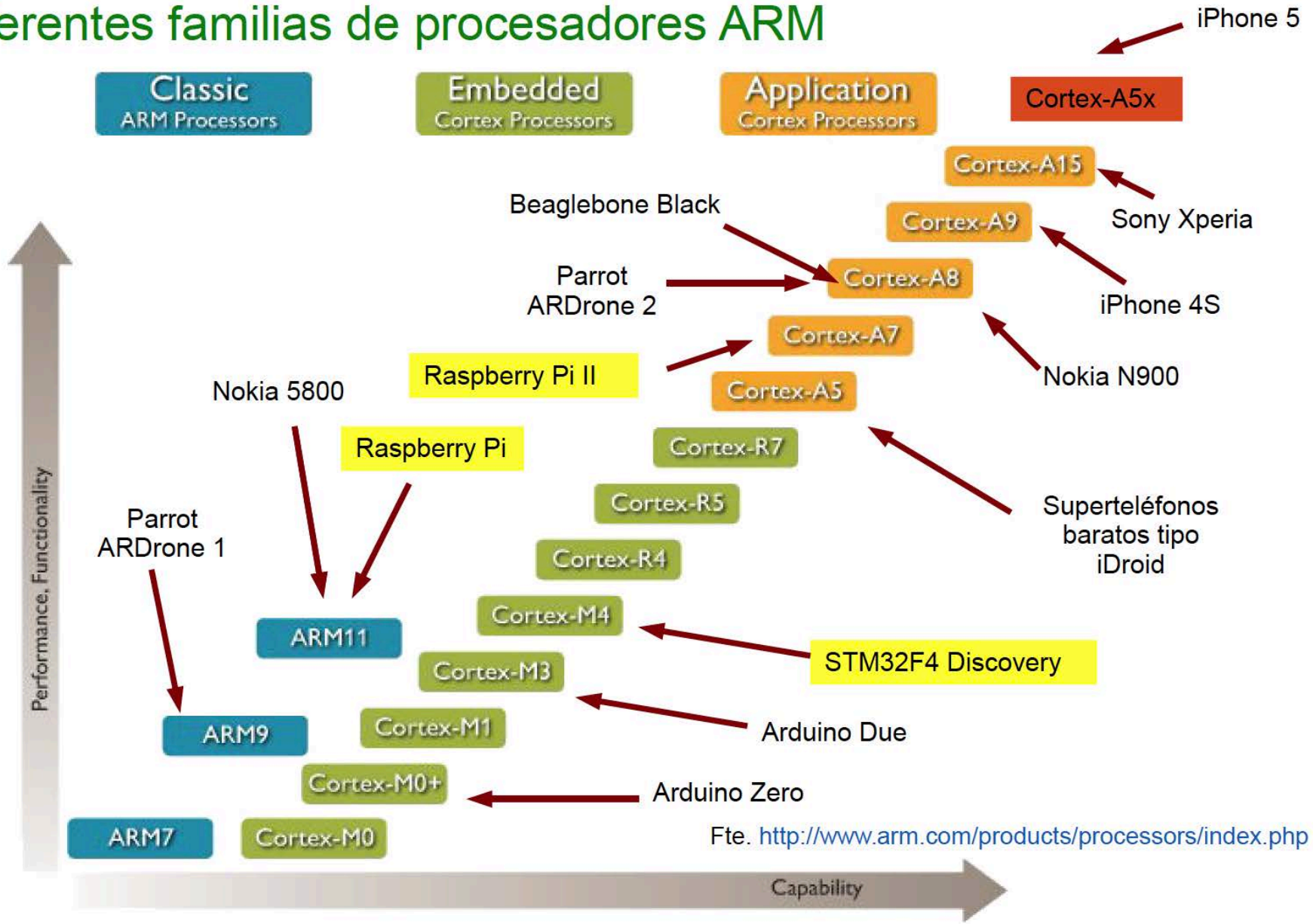
MICROPROCESADORES ARM

Un "SoC" ARM

Hay que ser conscientes de lo que se tiene en manos

MICROPROCESADORES ARM
La familia ARM Cortex

Diferentes familias de procesadores ARM



Fte. http://www.arm.com/products/processors/index.php

La familia ARM “Cortex”

- Notaciones previas a “Cortex”
  - ARM11, ARM9, ARM7TDMI
  - ¡un infierno para entenderlo!
- Cortex-A de Application
  - aplicaciones de alto rendimiento con sistemas operativos generales
  - tablets, e-books, móviles, smart-TV, ...
- Cortex-R de Real-time
  - sistemas de tiempo real-críticos y rendimiento equilibrado
  - robótica, impresoras, control de electrónico de motores, ...
- Cortex-M de Microcontroller
  - aplicaciones típicas de microcontroladores
  - lavadoras, microondas, mandos, nodos inalámbricos, ...

Elegir bien: David o Goliath

TÍPICO ERROR: no saber elegir lo adecuado

Elegir bien: David o Goliath

- David
  - muy ágil
  - algo debilucho
  - come poco
- Goliath
  - muy fuerte
  - movimientos lentos
  - come mucho

Elegir bien: David o Goliath

- David: microcontroladores ARM Cortex-M, Cortex-R y ARM7
  - airbag, frenos ABS, mando a distancia, wearables, sensores inalámbricos, lavadoras, microondas, equipo música, reloj inteligente básico, ...
  - no necesitan un sistema operativo de apoyo (pero ayuda mucho)
- Goliath: procesadores ARM Cortex-A, ARM11 y ARM9
  - teléfonos móviles, tablets, routers, reproductores multimedia, smartTV, NAS, servidores web sencillos, ..
  - usan siempre un sistema operativo de apoyo (Linux, Windows CE, QNX, VxWorks, ...)

NAS – Network Attached Storage

ARM Cortex-A: Linux + Raspberry Pi

- Goliath
  - procesador SoC ARM Cortex-A
  - se usa S.0. Linux: inmensas posibilidades
  - muy económico
  - fantástica comunidad
  - ***Open Source***

DIFERENCIAS ENTRE SoC y Microcontroladores

La diferencia principal de un SoC con un microcontrolador tradicional no debe pasarse por alto.

Los microcontroladores rara vez disponen de más de 100 Kilobytes de memoria RAM (de hecho, lo más frecuente es que las memorias -tanto la RAM como la flash - de un microcontrolador consten de unos pocos Kilobytes), y gran parte de estos son estructuras monochip,

Mientras que los SoC es usado para procesadores más potentes y de arquitectura más compleja, como son los que integran las computadoras y dispositivos actuales de telecomunicación que dependen de chips o módulos de memoria externos para ser eficaces.

Diferencias entre Arduino UNO versus Raspberry Pi

Recurso	ARDUINO UNO R3	Raspberry Pi 2 modelo B
CPU	Es un microcontrolador AVR de 8 bits Atmega 328P	Es un SoC Broadcom 2836 con un microprocesador ARM Cortex-A7 32 Bits
Sistema operativo	No ejecuta sistema operativo	Distribuciones LINUX, Windows 10
Precio en dólares (USA)	$ 25 Dlls	$42 Dlls
Tamaño	7.6x1.9x6.4 cm	8.6x5.4x1.7 cm
Memoria RAM	2 KB	1 GB
Frecuencia de Rejol	Hasta 20 MHz	Hasta 1.1 GHz
Red en la tarjeta	Ningua	10/100 Ethernet RJ-45
Voltaje de entrada	5-12V (Recomendado)	5V
Puertos USB	Uno	Cuatro

Diferencias entre Arduino UNO versus Raspberry Pi

Recurso	ARDUINO UNO R3	Raspberry Pi 2 modelo B
Multitarea	No	Si
Memoria Flash	32 KB	Micro SD 16/32 GB
Entorno de desarrollo	Arduino IDE, Processing	Python, C, C++, OpenCV, Octave, Scratch, cualquier lenguaje con soporte linux
Entradas analógicas	Si	No
Entradas/salidas digitales	Si, 20	Si, 26
Puertos seriales	Si	Si
Conectividad a internet	Si	Si
Procesador gráfico GPU	No	Si
Open Source	Si	Si

LA UNIÓN HACE LA FUERZA…

Dependiendo la aplicación o el proyecto, se elige el dispositivo más acorde a las necesidades…

MUCHAS GRACIAS!!!

ATTE.

DR. EVERARDO INZUNZA GONZÁLEZ

Email: einzunza@uabc.edu.mx

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA, ARQUITECTURA
Y DISEÑO

15765 SISTEMAS EMPOTRADOS

SEGUNDA PARTE

PRESENTA: DR. EVERARDO INZUNZA GONZÁLEZ
Profesor – Investigador
Email:einzunza@uabc.edu.mx
Tel. 175 0744 Ext. 64321

Hardware: Empezando con la Raspberry Pi 2

- Para empezar a trabajar con la Raspberry Pi, adicionalmente se necesita el siguiente hardware:

  - Memoria Micro SD HC 8GB
  - Fuente de alimentación 5V @ 2Amps mínimo con micro USB
  - Monitor o TV con entrada HDMI, o VGA
  - Cable HDMI
  - Convertidor HDMI-VGA (en caso de monitor VGA)
  - Teclado y mouse
  - Cable red ethernet (RJ-45)
  - Módulo USB-WIFI (Opcional)

HARDWARE NECERARIO

Cable red ethernet

Fuente de alimentación

Memoria Micro SD HC

Cable HDMI

Convertidor HDMI-VGA

Carcasa para Raspi

Teclado y mouse

Monitor

Módulo USB-WIFI

(Opcional)

El software: S.O. disponibles para la RPi

- Raspbian: distribución Linux basada en Debian Wheezy
  - la más “oficial”
- Pidora: distribución Linux basada en Fedora Remix
  - Fedora es el proyecto open de Red Hat
  - Red Hat es la distribución de referencia comercial/industrial
- Ubuntu Mate y Ubuntu Snappy Core
  - El Ubuntu típico y una optimización para IoT y similares
- OpenELEC y OSMC: Linux para XBMC (Xbox Media Center)
  - optimizado para ser un eficiente Home Theater
- RISC OS: no todo es Linux en el mundo (ni Windows, ni Mac,
- Microsoft Windows 10: esto tampoco es Linux





Sistema operativo a utilizar

- Para la Raspberry Pi usaremos Raspbian

- Para la computadora de desarrollo utilizaremos Windows/Mac OS/Linux

El software: empezar fácil con NOOBS

- NOOBS: New Out Of the Box Software
  - software a grabar en una tarjeta SD para la RPi
  - permite instalar, configurar y probar las anteriores distribuciones
  - descargable en http://www.raspberrypi.org/downloads/
  - es nuestra recomendación para empezar, (y mejor la "offline")

INSTALACIÓN DE RASPBIAN

Para la instalación del sistema operativo a la tarjeta SD de nuestra Raspi vamos a usar Windows. No se va a tratar de una instalación al uso, sino de introducir una imagen en la tarjeta de memoria. Una instalación convencional puede llevar más de 10 hora si queremos disponer de un entorno de escritorio, por lo que no se usará este método.

Primero nos vamos a la web de descargas de Raspberry Pi (http://www.raspberrypi.org/downloads) y descargamos una imagen del sistema operativo.

Raspbian "wheezy"

If you're just starting out, **this is the image we recommend you use**. It's a reference root filesystem from Alex and Dom, based on the Raspbian optimised version of Debian, and containing LXDE, Midori, development tools and example source code for multimedia functions.

Torrent	2012-12-16-wheezy-raspbian.zip.torrent
Direct download	2012-12-16-wheezy-raspbian.zip
SHA-1	514974a5fcbbbea02151d79a715741c2159d4b0a
Default login	Username: pi Password: raspberry

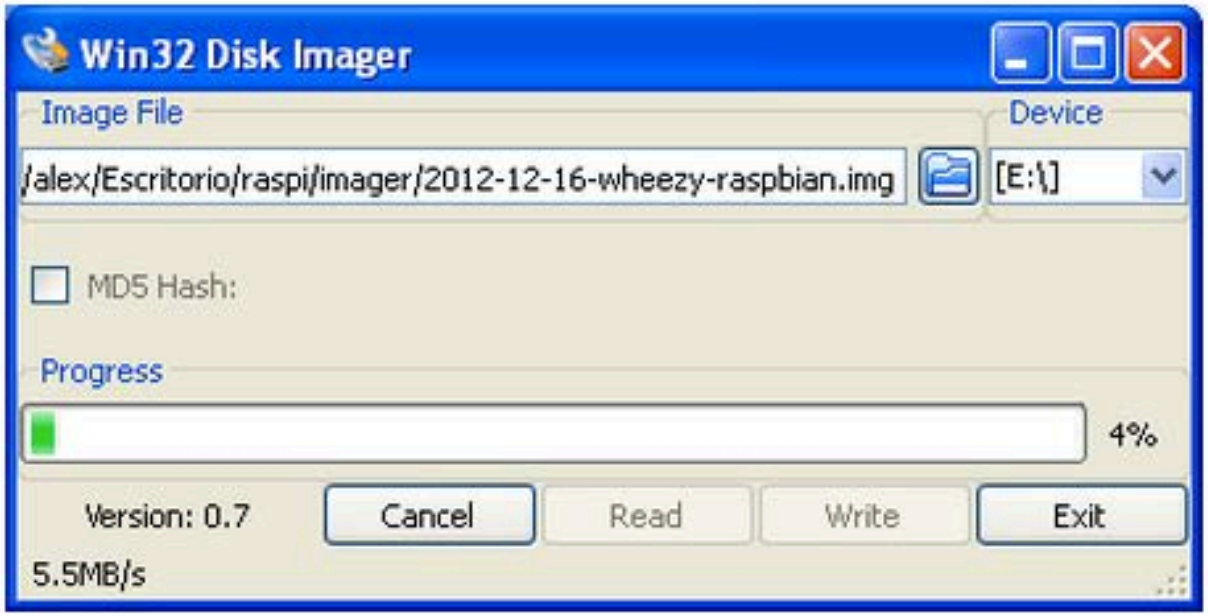
El archivo pesa cerca de 500 mb. Una vez descargado, extraemos su contenido (un archivo .img) a cualquier carpeta.

INSTALACIÓN DE RASPBIAN

Ahora nos descargamos Win32 Disk Imager (http://sourceforge.net/projects/win32diskimager/files/latest/download), una utilidad que nos permitirá meter la imagen de Raspbian en nuestra tarjeta SD. Abrimos el archivo y extraemos su contenido a cualquier carpeta. Finalmente se ejecuta Win32DiskImager

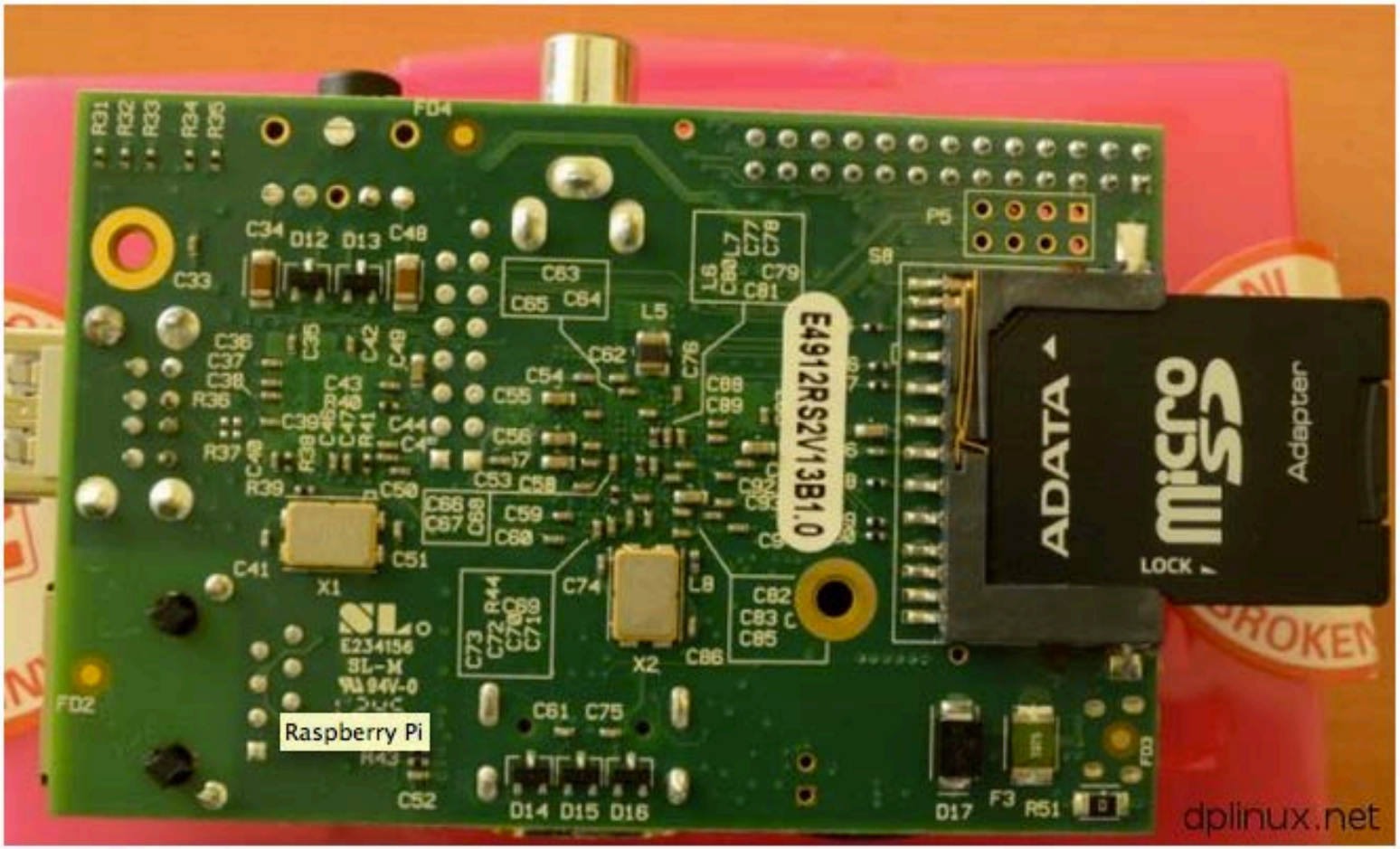
INSTALACIÓN DE RASPBIAN

Introducimos nuestra tarjeta SD en el lector de tarjetas del ordenador, seleccionamos la imagen que hemos extraído anteriormente y hacemos clic en “Write”.



INSTALACIÓN DE RASPBIAN

Una vez copiada la imagen, introducimos la tarjeta SD en nuestra Raspberry



Y ya podemos conectarla.

INICIANDO CON RASPBERRY PI

- Conectar monitor
- Conectar teclado y mouse
- Conectar cable de red ethernet
- Conectar módulo WiFi (opcional)
- Conectar fuente de alimentación 5V

Esperar a que arranque y se configure…

Formas de conectar/usar la Raspberry

- Tenemos dos formas de usar la Raspberry:

1. In situ conectando una pantalla, teclado y mouse.

2. Remotamente con SSH (Secure Shell)

II – El Kernel en tiempo real y técnicas de interconexión

Presenta:

Dr. Everardo Inzunza González

COMANDOS DE LINUX
para Raspberry Pi

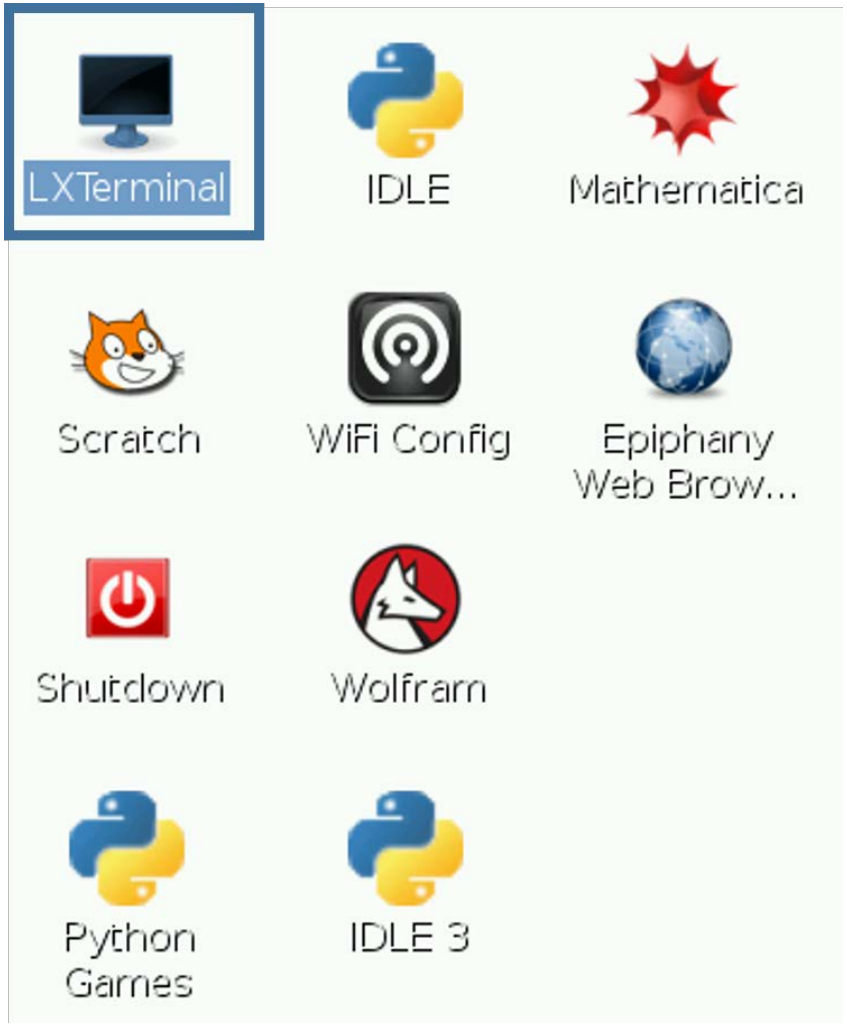
Terminal de comandos

En Raspbian, *LXTerminal* es la aplicación Terminal por default.

Básicamente, escribir líneas de comandos es el equivalente a utilizar el puntero del mouse para seleccionar y dar clicks, pero utilizando texto. Los comandos pueden combinarse para formar complejos Scripts que pueden realizar tareas de forma más eficiente que la mayoría del software tradicional.

Terminal de comandos

Para iniciar la aplicación damos doble click al icono.

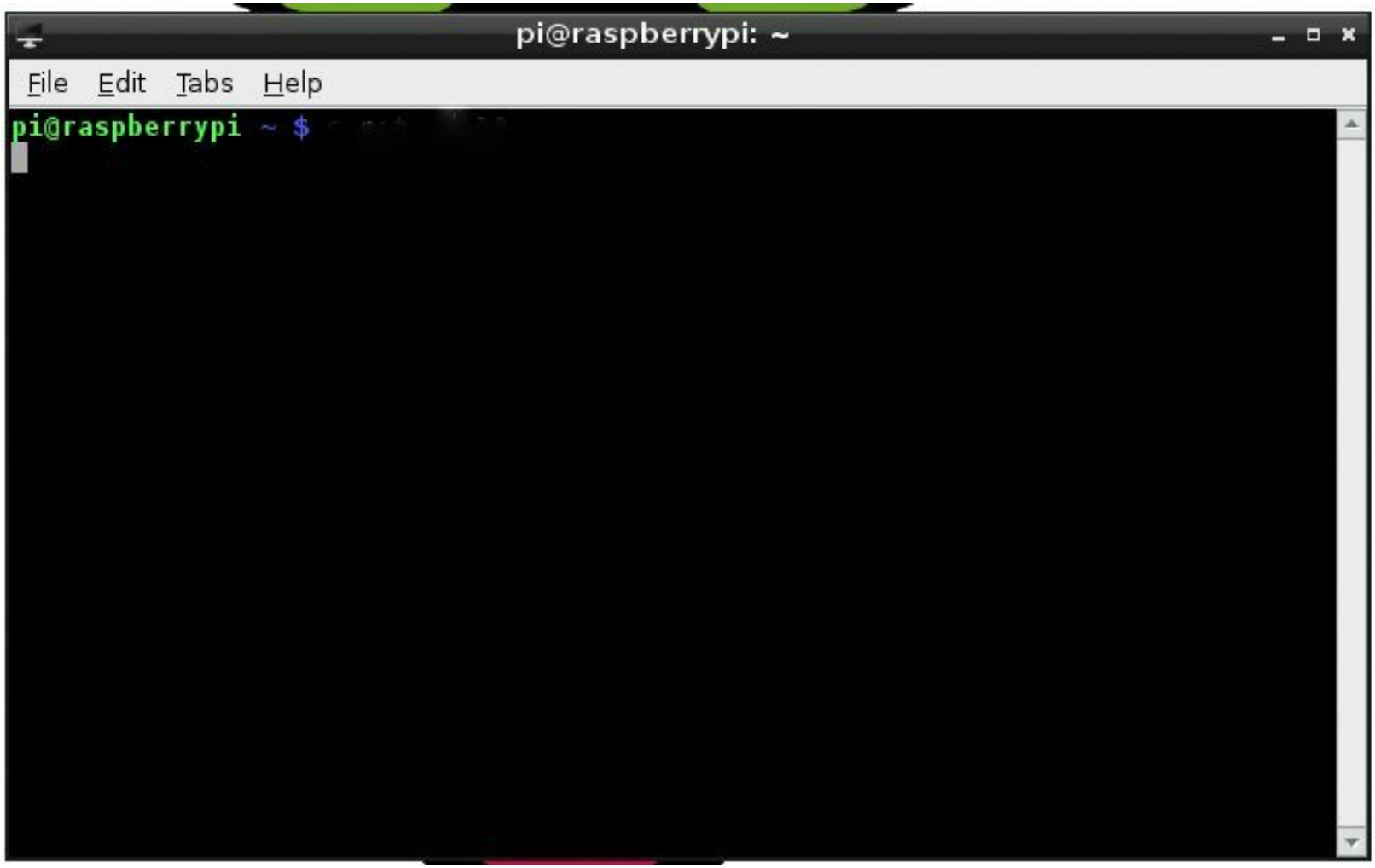


- Desde la ventana de comandos:

$ startx

Terminal de comandos

Al iniciar la Terminal lo primero que vemos es **pi@raspberrypi ~ $**.
Esto muestra tu nombre de usuario y nombre de host.



Comandos básicos

Para inciar se debe ejecutar el programa **LXTerminal** de Raspbian

```
pi@raspberrypi ~ $
```

- Para la navegación entre carpetas:
  - Imprimir en el terminal el directorio en el que te encuentras `pwd`
  - Ver el contenido del directorio `ls`
  - Cambiar a otro directorio `cd nombredeldirectorio`
- Para la creación de directorios usamos:
  - Para un solo directorio `mkdir nombredirectorio`
  - Para un árbol de directorios `mkdir -p /home/usuario/directorio1/directorio2`

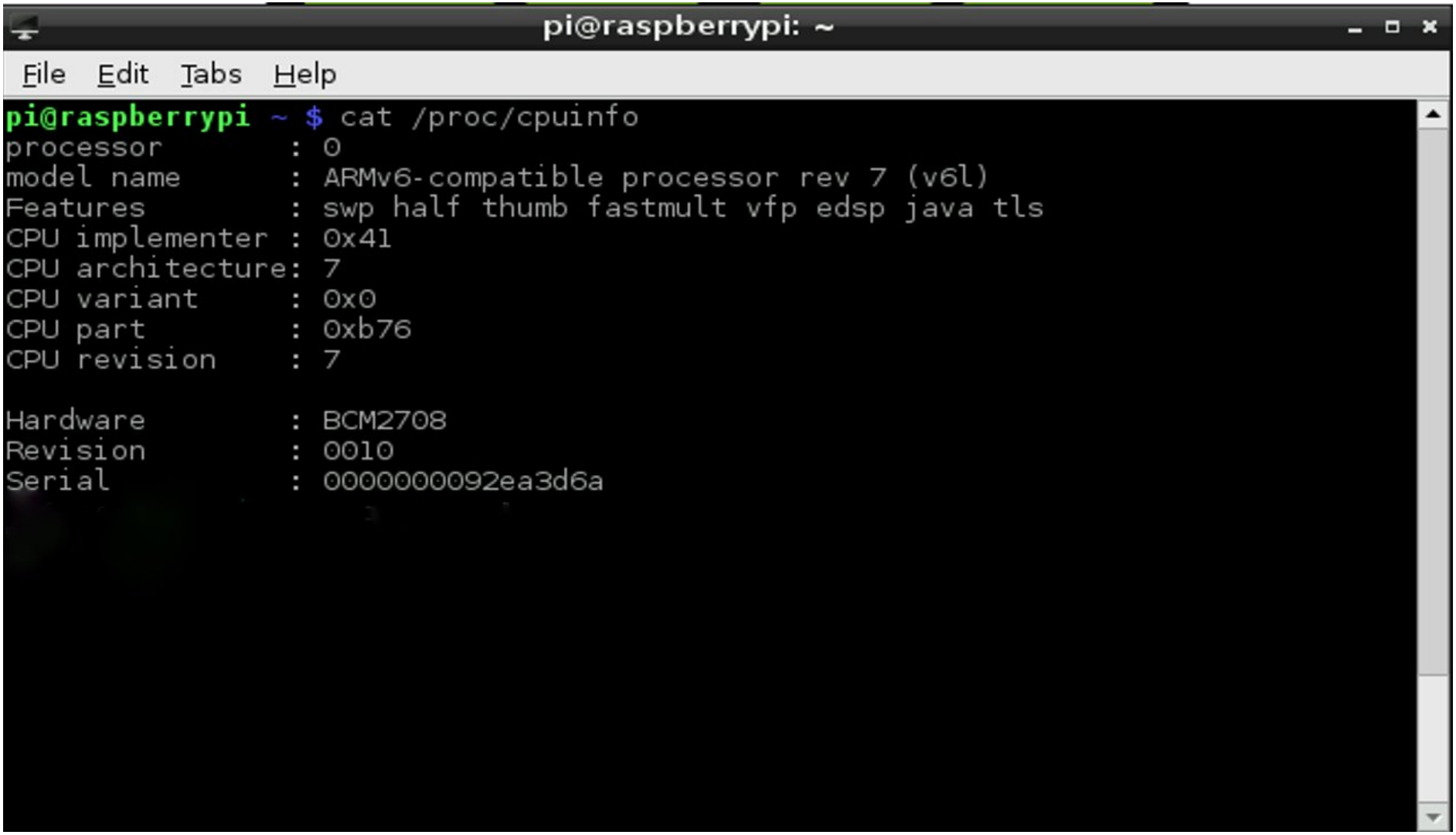
Para mostrar información de Hardware

- Para conocer la información hardware general: `cat /proc/cpuinfo`
- Para saber el estado de la memoria: `at /proc/meminfo`
- Para ver las particiones de la tarjeta de memoria o el disco duro: `cat /proc/partitions`
- Si queremos conocer la versión de nuestra Rasp: `cat /proc/version`
- Temperatura de la *CPU*: `vcgencmd measure_temp`
- Visualizar todos los dispositivos *USB* conectados: `lsusb`

Todos estos comandos nos permitirán **conocer con detalle todo lo que tiene nuestro dispositivo** y así saber qué podemos y qué no podemos hacer con él.

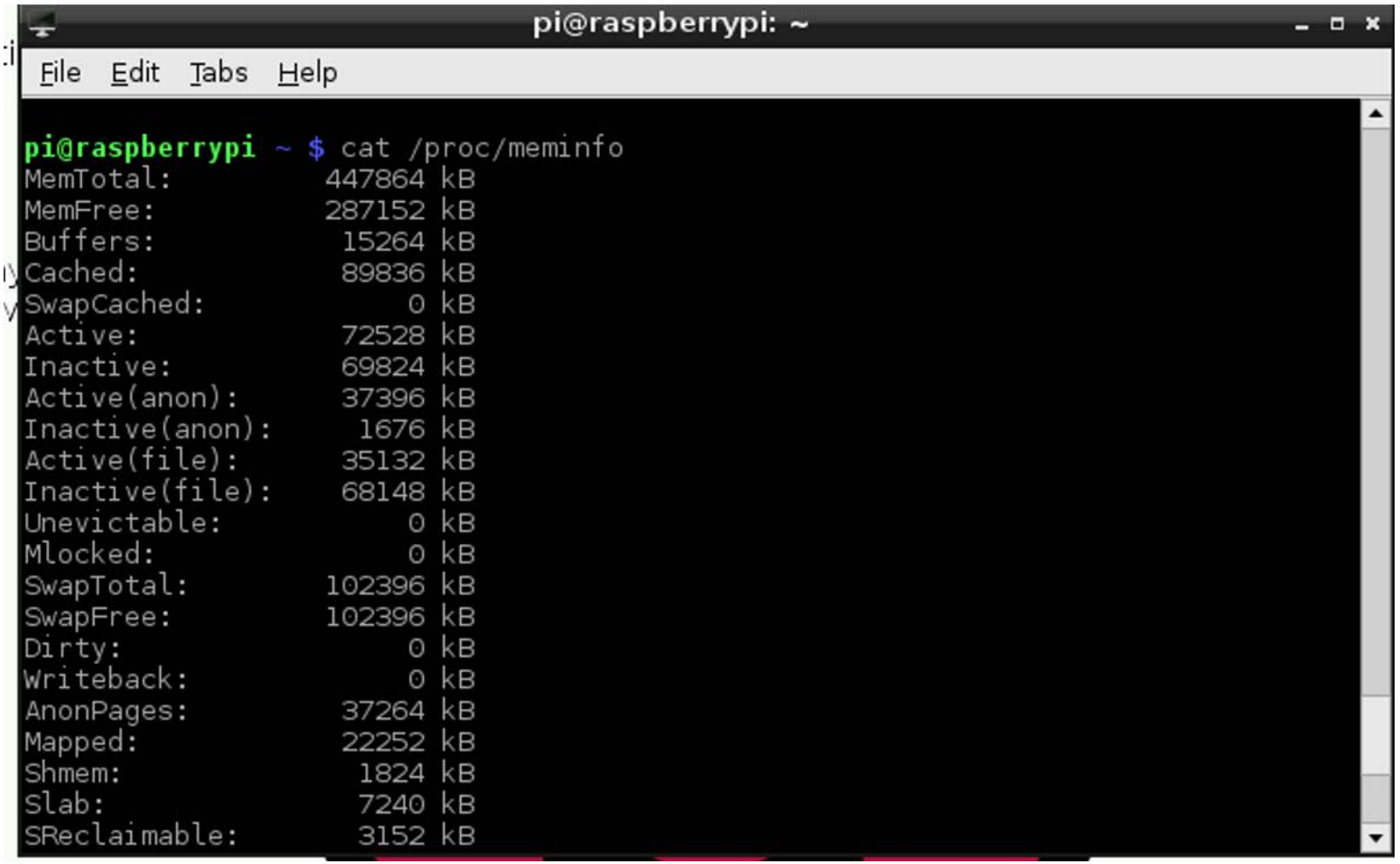
Para mostrar información de Hardware

Para saber detalles acerca del microprocesador de la Raspberry Pi, usamos **cat /proc/cpuinfo**



Para mostrar información de Hardware

Para saber detalles acerca de la memoria de la Raspberry Pi, usamos **cat /proc/meminfo**.

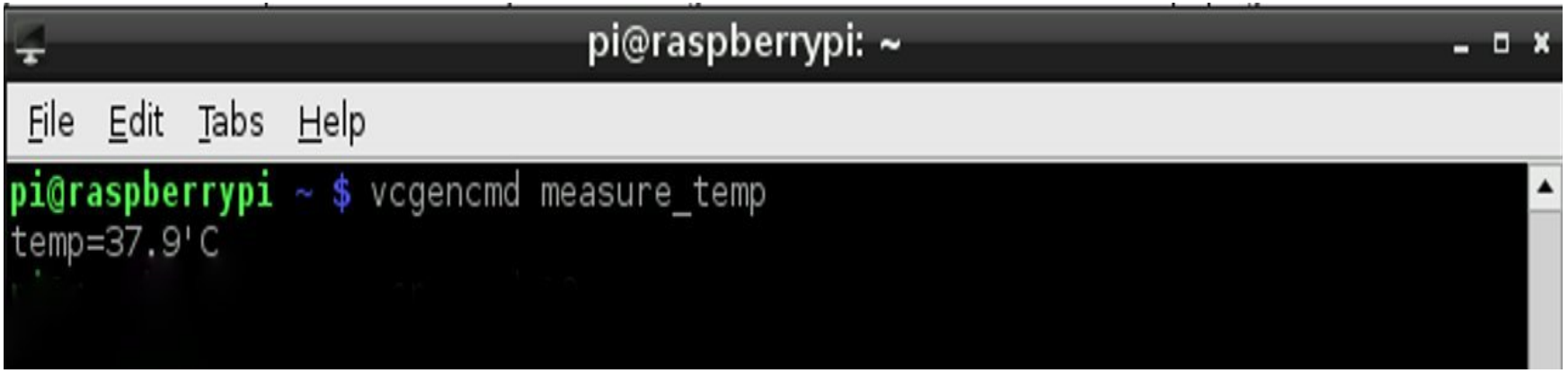


Para mostrar información de Hardware

Para saber las particiones del sistema, usamos el comando
cat /proc/partitions

Para mostrar información de Hardware

Para conocer la temperatura del CPU ejectuamos el comando **vgencmd measure_temp**



Comandos básicos

- **mv**-El comando "move" (mover) tiene dos propósitos en Linux: permitir que un archivo sea trasladado de un directorio a otro, y también permitir que los archivos sean renombrados. Esta última característica puede parecer fuera de lugar, pero en términos de Linux, el archivo se ha movido de un nombre a otro. El comando es invocado en la siguiente forma: `mv viejoarchivo nuevoarchivo`.
- **rm**-Abreviación para "remove" (remover), `rm` elimina archivos. Cualquier archivo (o lista de archivos) escrito después del nombre del comando será eliminado. Su equivalente en Windows es `del`, y los dos comandos comparten un mismo requisito, que es el cuidado que debe tomarse para asegurarse de eliminar el archivo correcto.
- **rmdir**-Por regla general, el comando `rm` no puede eliminar directorios por sí mismo. Por consiguiente, el comando `rmdir` es proporcionado para borrar los directorios una vez que éstos han quedado vacíos de archivos debido al comando `rm`.
- **mkdir**-Es el opuesto a `rmdir`. El comando `mkdir` crea nuevos directorios. Por ejemplo, escribir `mkdir micarpeta` en la terminal, creará un nuevo directorio llamado `micarpeta` bajo el directorio actual de trabajo. Al igual que `cd`, los directorios que se proporcionen al comando pueden ser relativos o absolutos.

Comandos básicos

Cat

Cat (de concatenar), es una maravillosa utilidad que nos permite visualizar el contenido de un archivo de texto sin la necesidad de un editor. Para utilizarlo solo debemos mencionarlo junto al archivo que deseamos visualizar:

Touch

Touch crea un archivo vacío, si el archivo existe actualiza la hora de modificación. Para crear el archivo prueba1.txt en /home, seria:

```
$ touch /home/prueba1.txt
```

Comandos básicos
Limpiar pantalla

Clear

Clear (de limpiar), es un sencillo comando que limpiara nuestra terminal por completo dejándola como recién abierta. Para ello ejecutamos:

```
$ clear
```

Comandos básicos

Para Visualizar documentación (ayuda) completa de los comandos linux

$ man nombre_comando

Ejemplo de los comandos básicos

Ls

Ls (de listar), permite listar el contenido de un directorio o fichero. La sintaxis es:

```
$ ls /home/directorio
```

El comando ls tiene varias opciones que permiten organizar la salida, lo que resulta particularmente útil cuando es muy grande. Por ejemplo, puedes usar -*a* para mostrar los archivos ocultos y -*l* para mostrar los usuarios, permisos y la fecha de los archivos. Así como para todos los comandos Linux, estas opciones pueden combinarse, terminando en algo como:

```
$ ls -la /home/directorio
```

Ejemplo de los comandos básicos

Cd

Cd (de *change directory* o cambiar directorio), es como su nombre lo indica el comando que necesitarás para acceder a una ruta distinta de la que te encuentras. Por ejemplo, si estas en el directorio /home y deseas acceder a /home/ejercicios, seria:

```
$ cd /home/ejercicios
```

Si estás en /home/ejercicios y deseas subir un nivel (es decir ir al directorio /home), ejecutas:

```
$ cd ..
```

Ejemplo de los comandos básicos

Mkdir

Mkdir (de *make directory* o crear directorio), crea un directorio nuevo tomando en cuenta la ubicación actual. Por ejemplo, si estas en /home y deseas crear el directorio ejercicios, sería:

```
$ mkdir /home/ejercicios
```

Mkdir tiene una opción bastante útil que permite crear un árbol de directorios completo que no existe. Para eso usamos la opción *-p*:

```
$ mkdir -p /home/ejercicios/prueba/uno/dos/tres
```

Ejemplo de los comandos básicos
Copiar archivos

Cp

Cp (de *copy* o copiar), copia un archivo o directorio origen a un archivo o directorio destino. Por ejemplo, para copiar el archivo prueba.txt ubicado en /home a un directorio de respaldo, podemos usar:

```
$ cp /home/prueba.txt /home/respaldo/prueba.txt
```

En la sintaxis siempre se especifica primero el origen y luego el destino. Si indicamos un nombre de destino diferente, cp copiará el archivo o directorio con el nuevo nombre.

Ejemplo de los comandos básicos
Copiar archivos

Cp

El comando también cuenta con la opción *-r* que copia no sólo el directorio especificado sino todos sus directorios internos de forma recursiva.
Suponiendo que deseamos hacer una copia del directorio /home/ejercicios que a su vez tiene las carpetas ejercicio1 y ejercicio2 en su interior, en lugar de ejecutar un comando para cada carpeta, ejecutamos:

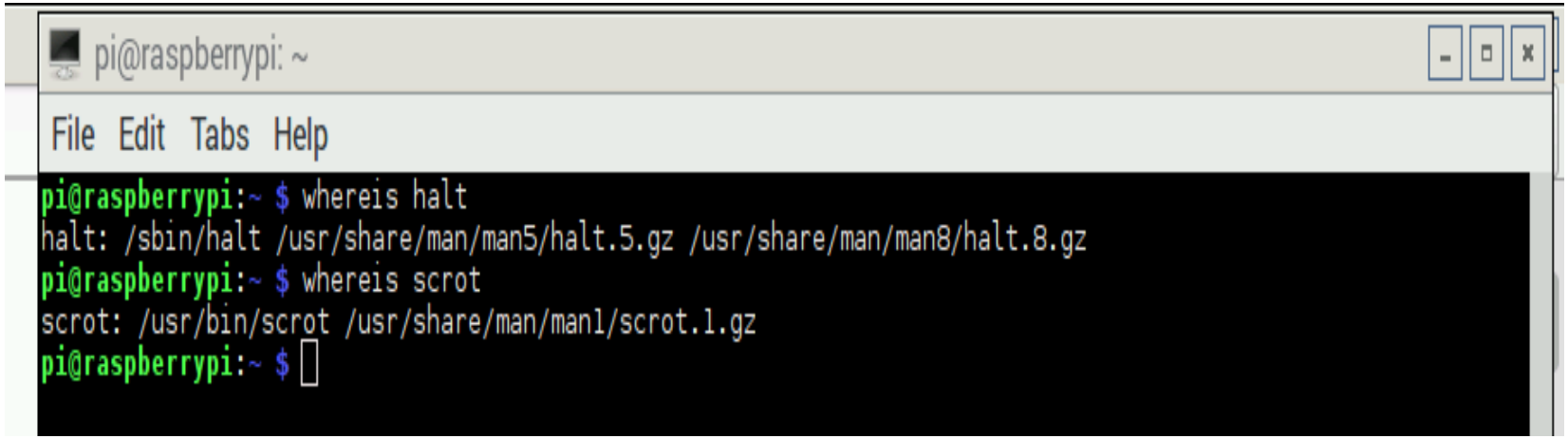
```
$ cp -r /home/ejercicios /home/respaldos/
```

Búsqueda de archivos

Comando **whereis**

Mostrar la ubicación de un fichero binario, de ayuda o fuente.

Ejemplo: **$ whereis halt**



Ejemplo de los comandos básicos
Mover archivos

Mv

Mv (de *move* o mover), mueve un archivo a una ruta específica, y a diferencia de *cp*, lo elimina del origen finalizada la operación. Por ejemplo:

```
$ mv /home/prueba.txt /home/respaldos/prueba2.txt
```

Al igual que *cp*, en la sintaxis se especifica primero el origen y luego el destino. Si indicamos un nombre de destino diferente, mv moverá el archivo o directorio con el nuevo nombre.

Ejemplo de los comandos básicos
Borrar archivos

Rm

Rm (de *remove* o remover), es el comando necesario para borrar un archivo o directorio. Para borrar el archivo prueba.txt ubicado en /home, ejecutamos:

```
$ rm /home/prueba.txt
```

Este comando también presenta varias opciones. La opción *-r* borra todos los archivos y directorios de forma recursiva. Por otra parte, *-f* borra todo sin pedir confirmación. Estas opciones pueden combinarse causando un borrado recursivo y sin confirmación del directorio que se especifique. Para realizar esto en el directorio respaldos ubicado en el /home, usamos:

```
$ rm -fr /home/respaldos
```

Ejemplo de los comandos básicos
Imprime directorio de trabajo

Pwd

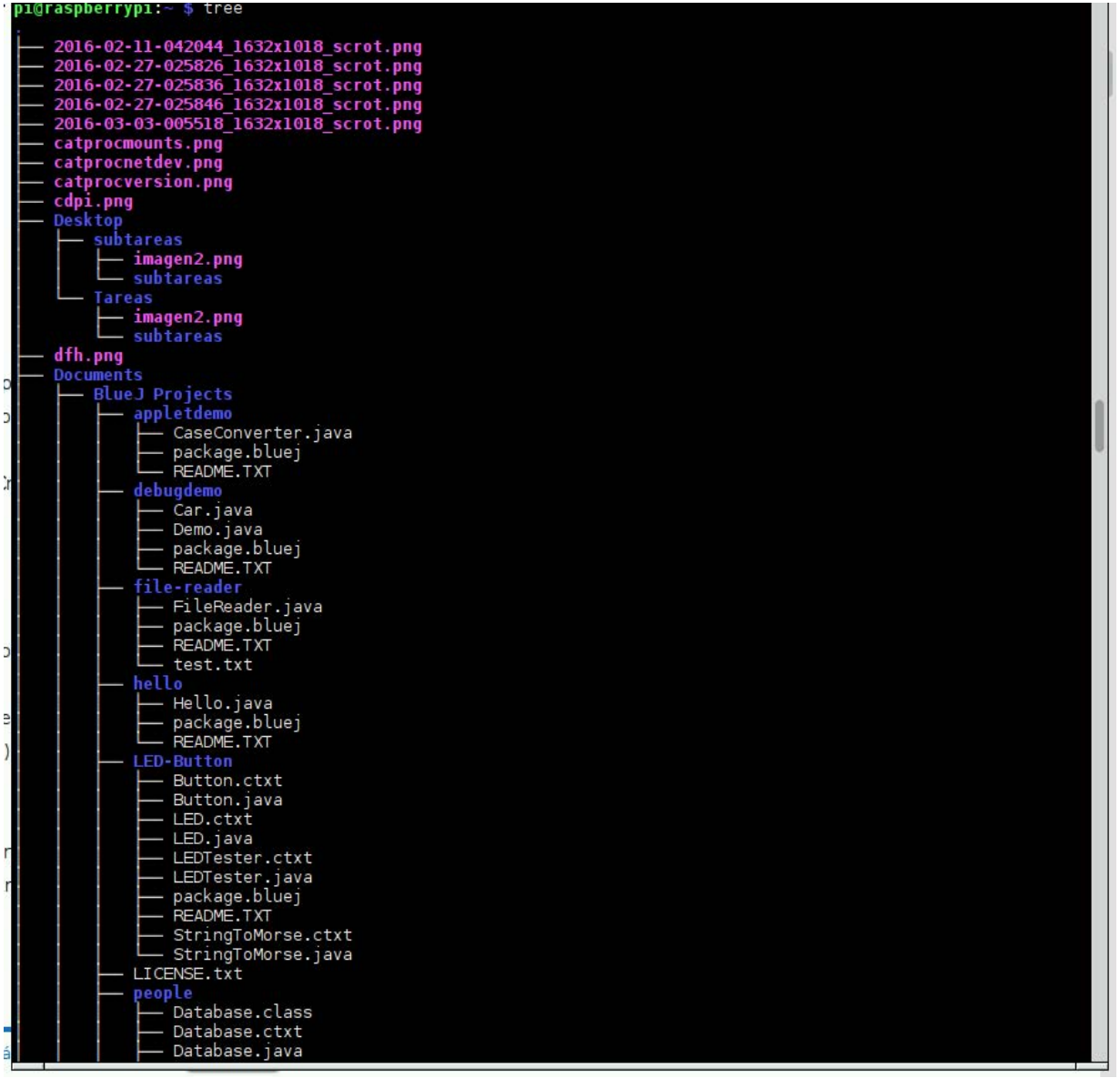
Pwd (de *print working directory* o imprimir directorio de trabajo), es un conveniente comando que imprime nuestra ruta o ubicación al momento de ejecutarlo, así evitamos perdernos si estamos trabajando con múltiples directorios y carpetas. Su sintaxis seria:

```
$ pwd
```

Ejemplo de los comandos básicos para manejo de archivos

Comando **tree**

Mostras los richeros y carpetas en forma de árbol comenzado por la raíz.



Ejemplo de los comandos básicos para manejo de archivos

Mostrar el tamaño de los farchivos y directorios ordenados por tamaño:

du –sk *| sort -rn

```
pi@raspberrypi:~ $ du -sk *| sort -rn
20284   Downloads
5304    Documents
1804    python_games
980     imss.pdf
928     Desktop
244     ventana3.png
196     youtu.goog.png
184     2016-03-07-234942_1776x952_scrot.png
160     p1.png
160     arcpngg.png
148     pdflect.png
148     htoppp2.png
108     okularr.png
104     nave.png
88      more.png
76      epiph.goog.png
72      mozzila.png
72      geannn.png
72      accesos.png
64      moverrr.png
64      edit.png
64      cattgene.png
60      autorun.inf
56      htopp.png
52      hwinff.png
52      epiffar.png
44      nousados.png
40      meminfoo.png
40      arboll.png
40      aptitudee.png
36      ice.goog.png
36      comandossss.png
28      dist.png
28      dispositivos.png
24      particmontadas.png
24      dfe.png
20      versssion.png
20      partitionss.png
16      pdos2.png
16      numeross.png
16      libreoff.png
16      cpuu.png
12      autocl.png
8       tempp.png
8       mover.png
8       espacioo.png
8       directorio1
4       Videos
4       Templates
4       respaldo
4       Public
4       prueba1.txt
4       Pictures
4       Music
pi@raspberrypi:~ $
```

Administrador de tareas o Monitor de actividades

Htop.

Htop es una herramienta para administrar los procesos del sistema de manera interactiva.

Por defecto no viene incluido en nuestro sistema, para instalarlo:

$ sudo apt-get install htop

Después de ser instalado, para ejecutarlo invoque:

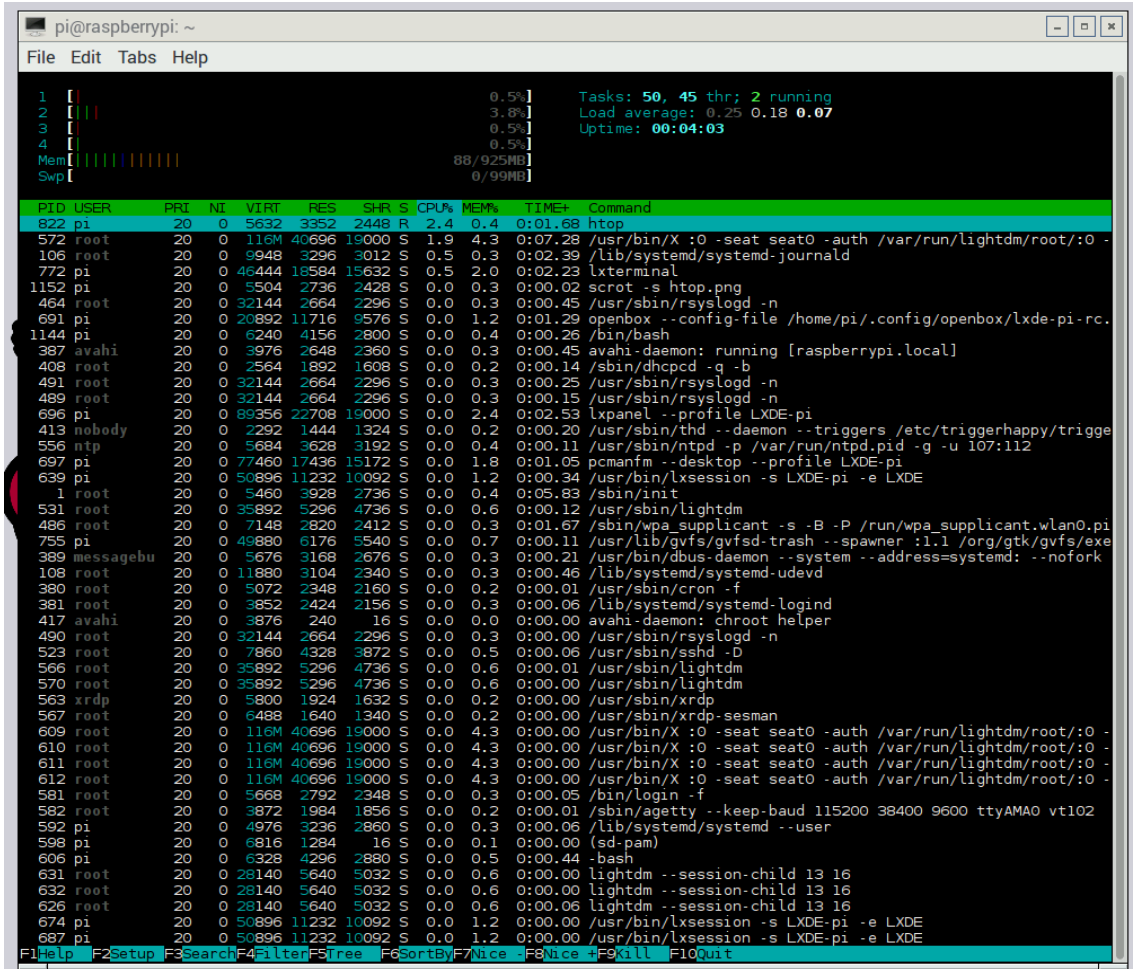
$ htop

Administrador de tareas o Monitor de actividades

Htop.

Muestra un listado de los procesos que estan actualmente en ejecución en todo el Sistema; estos se van actualizando en tiempo real ordenados por consumo de CPU.

Entre otras utilidades podemos ver el PID de un proceso, su usuario y el porcentaje de uso de procesador y memoria, entre otras.



Administrador de tareas o Monitor de actividades

Información del sistema

$ free ----- Revisa la RAM usada y libre

```
pi@raspberrypi:~ $ free
             total       used       free     shared    buffers     cached
Mem:        948016     901304      46712       9500      61148     675504
-/+ buffers/cache:     164652     783364
Swap:       102396          0     102396
```

El comando “sudo” (Super User Do)

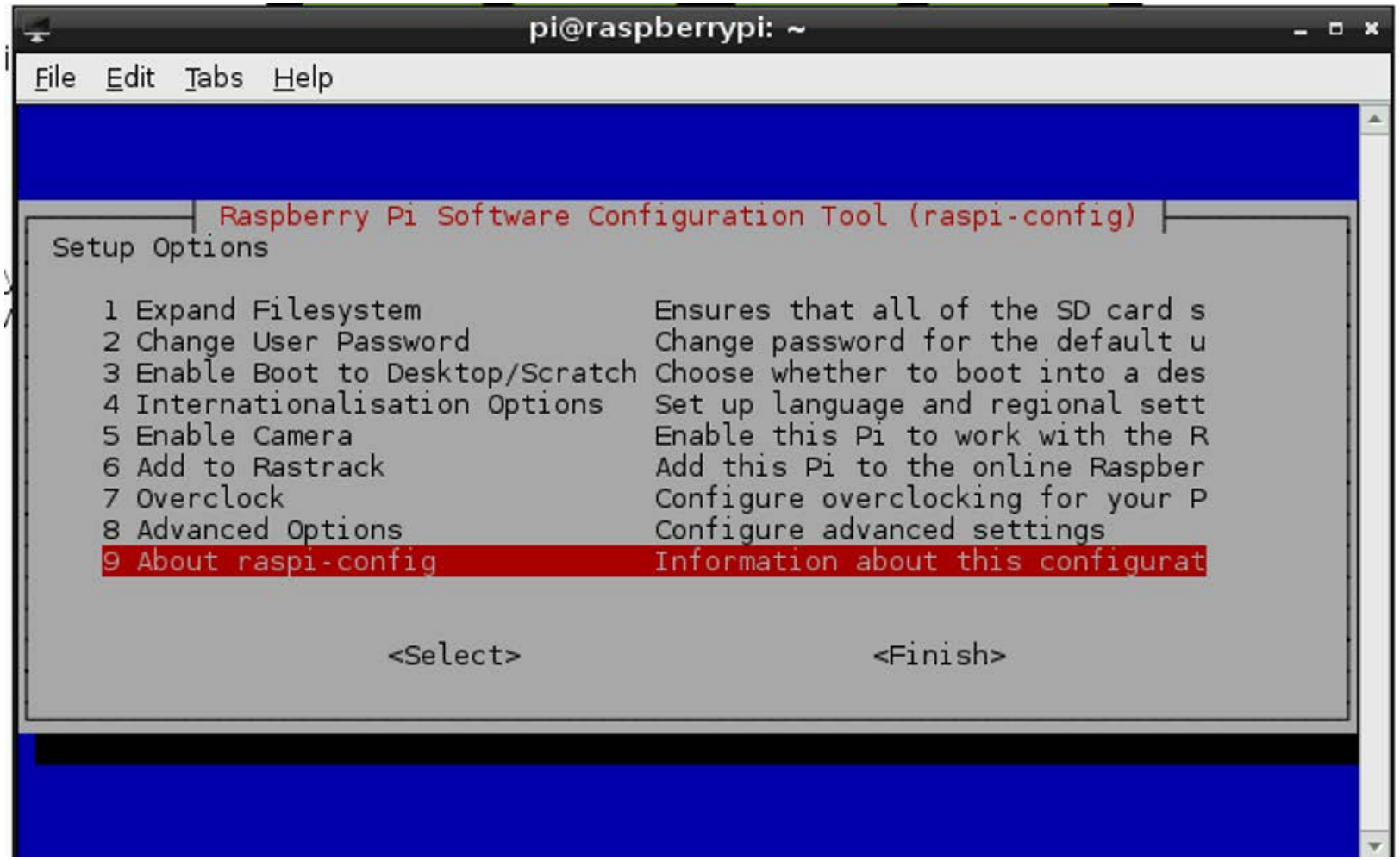
Quiza el comando más utilizado es **sudo**, esta palabra le **indica al sistema que el comando siguiente se va a ejecutar con permisos de “super usuario”**.

Este **es un nivel de acceso avanzado** similar a los permisos de “administrador” en sistemas Windows.

Este **comando es muy importante** y ampliamente utilizado, a continuación te mostramos unas instrucciones muy interesantes.

El comando “sudo” (Super User Do)

Con **$ sudo raspi-config** podemos acceder a las opciones de **configuración del sistema operativo**



Ejemplos de comando “sudo” (Super User Do)

Desde línea de comandos es posible **apagar o reiniciar el sistema**.

Con el comando **sudo shutdown –h now** se inicia automáticamente el proceso de **apagado del sistema**.

Con el comando **sudo shutdown –r now** es posible **reiniciar**,

También se puede incluir el tiempo o la hora para ejecutarse, por ejemplo **sudo shutdown –h 21-55** apagara el sistema a la hora indicada.

```
$ Init 0          (Apaga)

$ telinit 0

$ sudo halt

$ sudo poweroff

$ sudo reboot
```

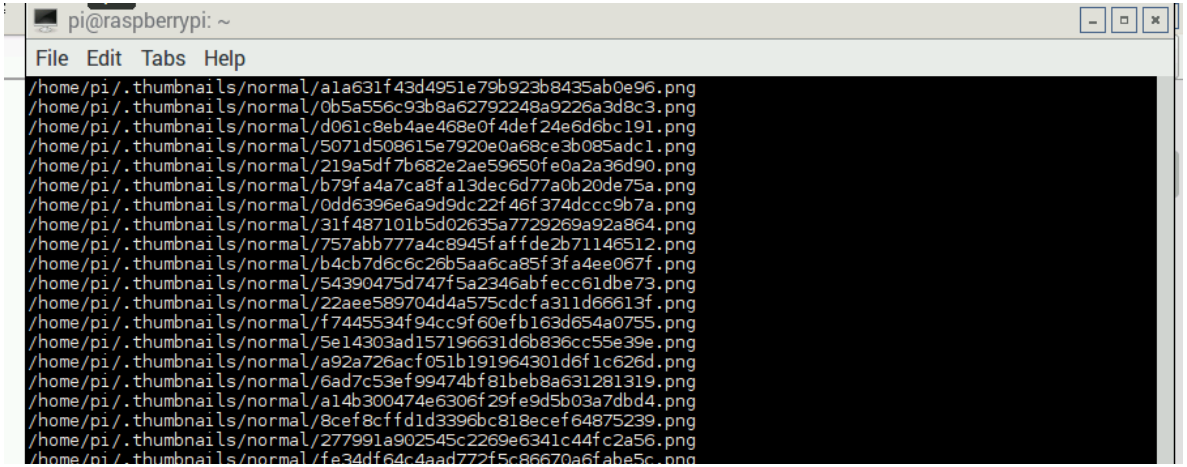
Ejemplo de comandos con “sudo” (Super User Do) Encontrar/Buscar Archivos

sudo find / -name file1 buscar archivo y directorio a partir de la raíz del sistema.

sudo find / -user user1 buscar archivos y directorios pertenecientes al usuario ‘user1’.

sudo find /home/user1 –name *.png

buca los ficheros con extension “.png” dentro del directorio “home/user1”



Ejemplo de comandos con "sudo" (Super User Do) Trabajo en Red (LAN, Wi Fi)

COMANDO "ping" (**P**acket **I**ternet **G**roper)

El comando PING es usado para probar la conexión y la latencia entre dos conexiones de red.

Es la mejor forma de probar la conectividad entre dos nodos.

Estas conexiones pueden ser conexiones de área local, área amplia o el Internet como un todo.

El comando ping envía paquetes de información a una dirección IP específica y luego mide el tiempo que se tarda en obtener una respuesta de la computadora o dispositivo especificado.

Ejemplo de comandos con "sudo" (Super User Do)
Trabajo en Red (LAN, Wi Fi)

EJEMPLOS COMANDO "ping"

Checa si está vivo (activo/funcionando) el host

$ ping google.com

PING google.com (74.125.200.102) 56(84) bytes of data.
64 bytes from plus.google.com (74.125.200.102): icmp_req=1 ttl=128 time=172 ms
64 bytes from plus.google.com (74.125.200.102): icmp_req=2 ttl=128 time=164 ms
64 bytes from plus.google.com (74.125.200.102): icmp_req=4 ttl=128 time=165 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3013ms
rtt min/avg/max/mdev = 164.618/167.289/172.010/3.364 ms

Ejemplo de comandos con "sudo" (Super User Do)
Trabajo en Red (LAN, Wi Fi)

EJEMPLOS COMANDO "ping"

Incrementa o decrementa el intervalo de tiempo entre paquetes
Ej. Espera 5 segundos antes de enviar el siguiente paquete

$ ping –i 5 google.com

Ej. Espera 0.1 segundo antes de enviar el siguiente paquete

$ sudo ping –i 0.1 google.com

Nota: Solamente el super usuario puede especificar tiempos menores a 0.2 Seg.

Ejemplo de comandos con "sudo" (Super User Do)
Trabajo en Red (LAN, Wi Fi)

EJEMPLOS COMANDO "ping"

Envía N paquetes y se detiene

$ ping –c 4 google.com

PING google.com (74.125.135.100) 56(84) bytes of data
.
64 bytes from plus.google.com (74.125.135.100): icmp_req=1 ttl=128 time=251 ms
64 bytes from plus.google.com (74.125.135.100): icmp_req=2 ttl=128 time=180 ms
64 bytes from plus.google.com (74.125.135.100): icmp_req=3 ttl=128 time=179 ms
64 bytes from plus.google.com (74.125.135.100): icmp_req=4 ttl=128 time=179 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 179.569/197.734/251.433/31.005 ms

Ejemplo de comandos con "sudo" (Super User Do) Trabajo en Red (LAN, Wi Fi)

EJEMPLOS COMANDO "ping"

Especificando direcciones de IP

$ ping 192.168.3.33 192.168.7.1 192.168.4.45

Nota: En caso de que algunas de las direcciones IP's no esté disponible, marcará los errores

Ejemplo de comandos con "sudo" (Super User Do)
Trabajo en Red (LAN, Wi Fi)

Tarjeta de Red Ethernet

Mostrar la configuración de una tarjeta de red Ethernet.

$ sudo ifconfig eth0

Activar (Habilita) una tarjeta de red Ethernet.

$ sudo ifup eth0

Des-Activar (Des-Habilitar) una tarjeta de red Ethernet.

$ sudo ifdown eth0

Ejemplo de comandos con "sudo" (Super User Do) Trabajo en Red (LAN, Wi Fi)

Redes Wi Fi

- Mostrar las redes Wi Fi Disponibles.

$ sudo iwlist scan

- Comprobar que está cargado correctamente el driver del adaptador Wi Fi

$ sudo iwconfig

Para configurar ahora la conexión a una red WiFi encriptada con WEP cuyo ESSID es "ssidpruebas" y cuya clave WEP de 104 bits en hexadecimal es "11223344556677889900aabbcc", haremos:

$ sudo iwconfig wlan0 essid ssidpruebas key 1122-3344-5566-7788-9900-aabb-cc

Actualizando Raspbian

Primero se tiene que **actualizar (descargar)** la lista de los paquetes con el siguiente comando:

$ sudo apt-get update

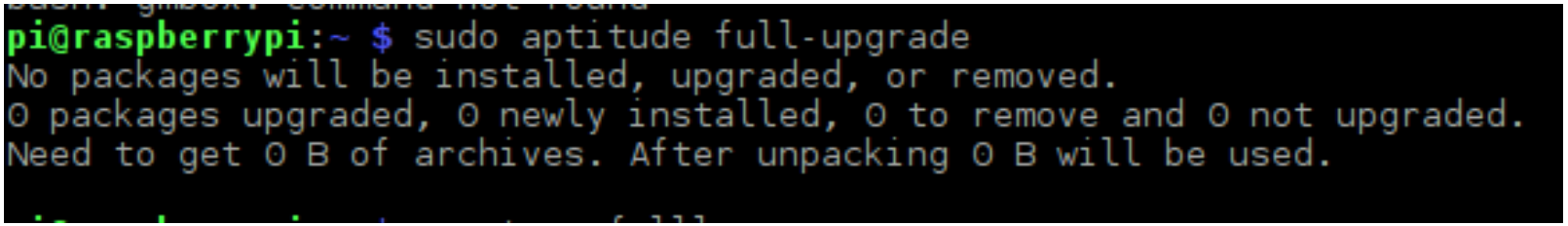
Luego se **instalan** todas las actualizaciones

$ sudo apt-get upgrade

Para **Actualizar el Kernel**

$ sudo rpi-update

Para **Actualizar los paquetes**



Liberando espacio Raspbian

Todos los archivos descargados de las actualizaciones (.deb) se guardan en: /var/cache/apt/archives

Para liberar espacio se debe ejecutar el siguiente comando:

$ sudo apt-get clean

Instalando software

Como alternativa a utilizar la **Pi Store** para descargar e instalar nuevo software, se puede utilizar el comando **apt-get**, que es el administrador de software incluido en cualquier distribución de **Linux basada en Debian** (incluye Raspbian)

Para **instalar un nuevo programa** usamos:

sudo apt-get install <software>

donde **<software>** es el nombre del programa que se va a instalar.

Instalando software

Para **instalar un paquete** ya descargado:

sudo dpkg –i paquete.deb

donde **paquete.deb** es el nombre del programa que se va a instalar.

Des-instalando software

Para **Des-instalar un programa (software)** usamos:

sudo apt-get remove <software>

donde **<software>** es el nombre del programa que se va a des-instalar.

El comando **remove**, tiene un hermano mas poderoso, que se llama purge (este último elimina todo).

sudo apt-get purge <software>

Buscando software o paquetes

Para **Buscar un programa (software)** usamos:

$ apt-cache search nombre_software

Para **saber si un paquete o programa (software)** está Instalado:

$ aptitude show nombre_software

Creando nuevo usuario en Raspi

Para **crear usuarios** adicionales en Raspberry

$ sudo adduser nombre_usuario

Después preguntará por el password para el usuario, dejar en blanco para no guardar password.

Para **borrar usuarios** en Raspbery

$ sudo userdel -r nombre_usuario

Historial de comandos

Para **visualizar el historial de comandos ejecutados** en la Raspberry

$ history

```
129  sudo apt-get install epiphany-bro
130  sudo apt-get install iceweasel
131  scrot -s nave.png
132  apt-get isntall geany
133  apt-get install geany
134  apt-get install emacs
135  apt-get install mplayer
136  sudo apt-get install mplayer
137  sudo apt-get install geany
138  scrot -s pdflect.png
139  scrot -s instalac.png
140  sodo apt-get install gimp
141  sudo apt-get install gimp
142  scrot-s geany.png
143  scrot -s geannn.png
144  wpa_gui
145  sudo apt-get update
146  sudo apt-get upgrade
147  scrot -s edit.png
148  sudo apt-get install file-roller
149  remove roller
150  sudo remove roller
151  scrot -s accesos.png
152  scrot -s tablaa.png
153  scrot -s COMANDOSS.png
154  scrot -s libreoff.png
155  scrot -s okularr.png
156  scrot -s epiffar.png
157  scrot -s mozzila.png
158  cat /proc/cpuinfo
159  scrot -s cattgene.png
160  at /proc/meminfo
161  cat /proc/meminfo
162  scrot -s meminfoo.png
163  cat /proc/partitions
164  scrot -s partitionss.png
165  cat /proc/version
```

Espacio en disco

Mostrar una lista de particiones montada:

$ df -h

```
pi@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       6.0G  4.1G  1.6G  73% /
devtmpfs        459M     0  459M   0% /dev
tmpfs           463M     0  463M   0% /dev/shm
tmpfs           463M  6.4M  457M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           463M     0  463M   0% /sys/fs/cgroup
/dev/mmcblk0p6   63M   20M   44M  31% /boot
tmpfs            93M  4.0K   93M   1% /run/user/1000
/dev/mmcblk0p5   30M  449K   28M   2% /media/pi/SETTINGS
pi@raspberrypi:~ $ scrot -s particmontadas.png
```

Ajustando hora y fecha

```
sudo raspi-config
```

Después seguimos estos **pasos**:

- Internationalization options
- Cambiar el 'Time Zone'
- Seleccionar el área geográfica (Europe)
- Seleccionar la ciudad o región (Madrid)
- Reiniciar la Raspberry.

Ajustar la hora

```
sudo date 08050822
```

donde:

- 08 es el mes
- 05 es el día
- 08 es la hora
- 22 son los minutos

TAREA #4

- **Investigar al menos 5 comandos** "Linux/ Raspberry/Raspbian adicionales para c/u de las **siguientes actividades**:

  - Información del sistema
  - Manejo o búsqueda de archivos y directorios
  - Manejo espacio de disco
  - Trabajo en red (LAN, Wi Fi)
  - Apagar o re-iniciar
  - Actualización de paquetes APT

Incluír EJEMPLOS de todos

Acceso a Internet con la RasPi

- Una de las características más atractivas de Raspberry Pi es su capacidad de conectarse a internet.

- Esto abre un gran abanico de posibles usos, aplicaciones dirigidas a domótica, telepresencia, monitoreo de red, etc.

- La conexión se puede realizar de forma cableada con Ethernet o inalámbrica utilizando un adaptador Wifi USB.

Acceso a Internet con la RasPi
-Conexión Ethernet

La conexión por Ethernet no debería dar mayor problema, solo se necesita conectar nuestro cable de red RJ-45 directamente de la Raspberry Pi **al router (con acceso a internet).**

En caso de que no se pueda conectar, verifique que su router está configurado para **DHCP** (Dynamic Host Configuration Protocol), el cual asigna aleatoriamente una IP a los dispositivos.

Acceso a Internet con la RasPi 2
-Conexión Wi Fi

Se debe contar con un adaptador que soporte el **chipset RTL8192cu ó RTL8188cus** , en este caso por ejemplo se utiliza un adaptador de la marca Realtek.

Acceso a Internet con la RasPi
-Conexión Wi Fi

El adaptador Wifi puede llegar a consumir mucha energía, es recomendable ver capacidad de la fuente de alimentación, se sugiere utilizar una fuente de 5V @ 2.5 Amps.

EXPOSICIONES DE COMANDOS LINUX

POR PARTE DE LOS ESTUDIANTES

III -Programación en C para microprocesadores empotrados de 32 Bits.

TAREA #5

- ***Investigar*** *los orígenes* del ***Lenguaje C***,
  - Quién lo desarrolló?
  - En que Año?
  - Donde?
  - Cuál era el objetivo?
  - Porqué se llama C?
  - Quién lo estandarizó?

SISTEMAS EMPOTRADOS

III -Programación en C para microprocesadores empotrados de 32 Bits.

Profesor: Dr. Everardo Inzunza G.

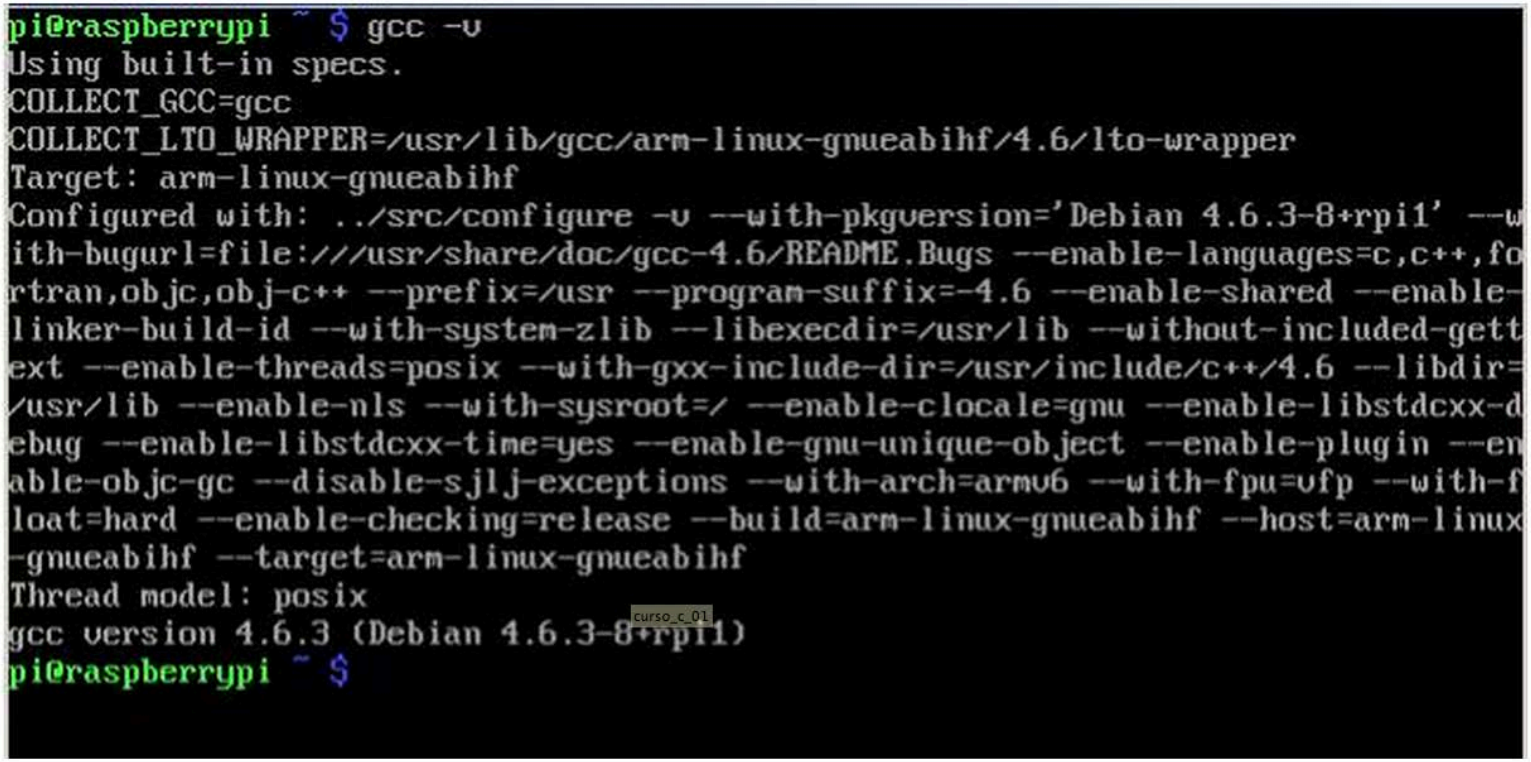
Programación C para RasPi

- Una característica de linux es su total integración con el lenguaje C.

- Se puede programar en C, sin necesidad de instalar software adicional.

Programación C para RasPi

- En la terminal de Linux se verifica que el entorno C está instalado.
- Ejecutamos **gcc –v**
- **A continuación se mostrará una serie de parámetros**



Programación C para RasPi

- Posteriormente borrar pantalla (**clear**)

- Nos cambiamos a la carpeta raíz de nuestro usuario (pi).

cd /home/pi

- Verificamos que estamos ahí **(pwd)**

- Creamos una carpeta “curso”

mkdir curso

- Verificamos que la carpeta esté creada **(ls –l)** o bien **ls**

- Entramos a la carpeta “curso” **(cd curso)**

- Verificamos (opcionalmente) de nuevo nuestra ubicación **(pwd)**

Programación C para RasPi

```
1 nota: este paso es opcional ya que el cursor de Linux
2 ya suele indicarnos donde estamos
```

- Ahora creamos un archivo vacío

touch hola.c

Verificamo que se ha creado y está vacío (ocupa 0 bytes)

ls –l

```
pi@raspberrypi ~ $ mkdir curso
pi@raspberrypi ~ $ ls -l
total 12
drwxr-xr-x 2 pi pi 4096 nov 20 08:46 curso
drwxr-xr-x 2 pi pi 4096 oct 28 23:54 Desktop
drwxrwxr-x 2 pi pi 4096 jul 20 20:07 python_games
pi@raspberrypi ~ $ cd curso/
pi@raspberrypi ~/curso $ pwd
/home/pi/curso
pi@raspberrypi ~/curso $ touch hola.c
pi@raspberrypi ~/curso $ ls -l
total 0
-rw-r--r-- 1 pi pi 0 nov 20 08:55 hola.c
pi@raspberrypi ~/curso $ _
```

Programación C para RasPi

- Lo editamos (usando el mini editor nano)

  nano hola.c

  - Escribimos “algo original” en nuestro primer programa

```
#include <stdio.h>

int main() {
   printf("Hola Mundo desde mi RasPi ....");
   return 0;
}
```

Programación C para RasPi

Programación C para RasPi

- Salimos y guardamos (Ctrl-x, Sí, Enter)
  - Escribimos “algo original” en nuestro primer programa
  - Verificamos que hola.c ya no ocupa 0 bytes

    ls –ls

  - Opcionalmente podemos verificar rápidamente el contenido del archivo hola.c

    more hola.c

```
pi@raspberrypi ~/curso $ ls -l
total 4
-rw-r--r-- 1 pi pi 92 nov 20 09:10 hola.c
pi@raspberrypi ~/curso $ more hola.c
#include <stdio.h>

int main() {
  printf ("Hola Mundo desde mi RasPi ....");
  return 0;
}
pi@raspberrypi ~/curso $
```

Programación C para RasPi

- Compilamos el programa hola.c

  gcc –o holapi hola.c

  (gcc es el llamado al compilador C de Linux)

  -o holapi indicamos el nombre de salida del archivo ejecutable

  Hola.c indicamos el nombre del archivo fuente

- **Vemos que es lo que se ha generado en /curso**

  ls -l

-Ejecutamos el programa

./holapi

Programación C para RasPi

- Para que cambie de renglón, agregamos \n

```
#include <stdio.h>

int main() {
printf("Hola Mundo desde mi RasPi ....\n");
return 0;
}
```

– editamos y corregimos

Programación C para RasPi

- compilamos de nuevo, obteniendo

```
pi@raspberrypi ~/curso $ gcc -o holaPi hola.c
pi@raspberrypi ~/curso $ ./holaPi
Hola Mundo desde mi RasPi ....
pi@raspberrypi ~/curso $ _
```

USANDO LOS PINES GPIO

- **Instalando** las librerías GPIO (WiringPi) Opción A

$ sudo apt-get update

$ sudo apt-get upgrate

$ sudo apt-get install git-core

Si marca algunos errores, asegúrese de tener acutalizada su Raspberry

Para descargar la librería WiringPi usando GIT

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

Posteriormente para instalar,

$ cd wiringPi

$ git pull origin

$ ./build

Sino marca errores, la librería está instalada correctamente y lista para usarse en lenguaje C

USANDO LOS PINES GPIO

- **Instalando** las librerías GPIO (WiringPi) Opción B

Descargar el archivo .tar del siguiente link:

https://git.drogon.net/?p=wiringPi;a=summary

Busque la palabra snapshot del lado derecho y haga click en el que se encuentra en la parte superior (1ro de la lista es la versión más reciente)

Se descargará un archivo .tar.gz, por ejemplo con el nombre:
wiringPi-170dce5.tar.gz (nota: este nombre puede cambiar)

Luego se descomprime este archivo

```
$ tar xfz wiringPi-170dce5.tar.gz
$ cd wiringPi-170dce5.tar.gz
$ ./build
```

Sino marca errores, la librería está instalada correctamente y lista para usarse en lenguaje C

USANDO LOS PINES GPIO

- **Probando** las librerías GPIO (WiringPi) Opción B

Ejecute el comando gpio para probar la instalación de la librería WiringPi

$ gpio –v

$ gpio readall

```
pi@raspberrypi ~/progs_c $ gpio -v
gpio version: 2.27
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Model 2, Revision: 1.1, Memory: 1024MB, Maker: Sony
  This Raspberry Pi supports user-level GPIO access via /dev/gpiomem.
pi@raspberrypi ~/progs_c $
```

USANDO LOS PINES GPIO

- **Probando** las librerías GPIO (WiringPi) Opción B

Ejecute el comando gpio para probar la instalación de la librería WiringPi

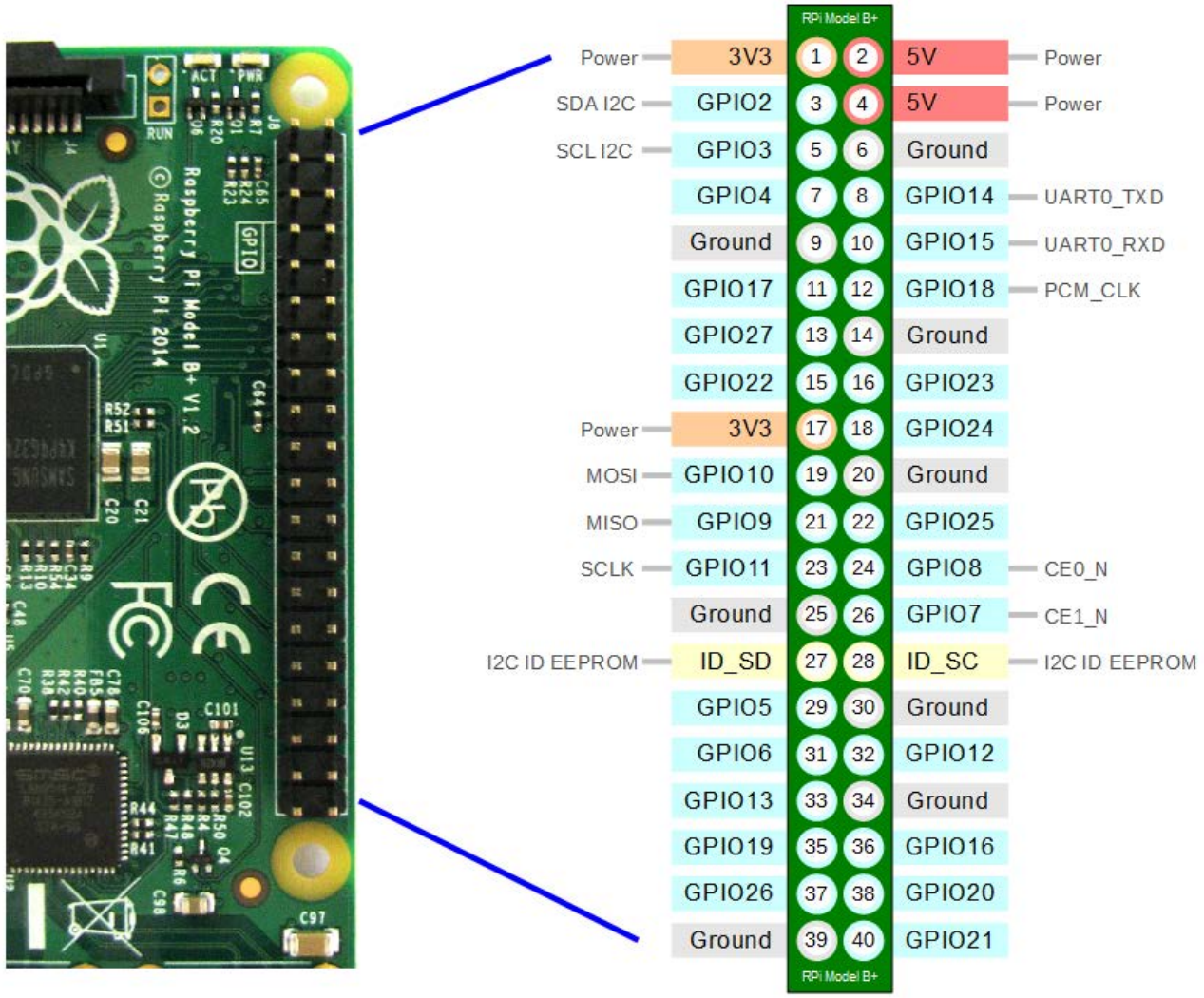
$ gpio –v

$ gpio readall

```
pi@raspberrypi ~/progs_c $ gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |  OUT | 0 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |  OUT | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
pi@raspberrypi ~/progs_c $
```

USANDO LOS PINES GPIO

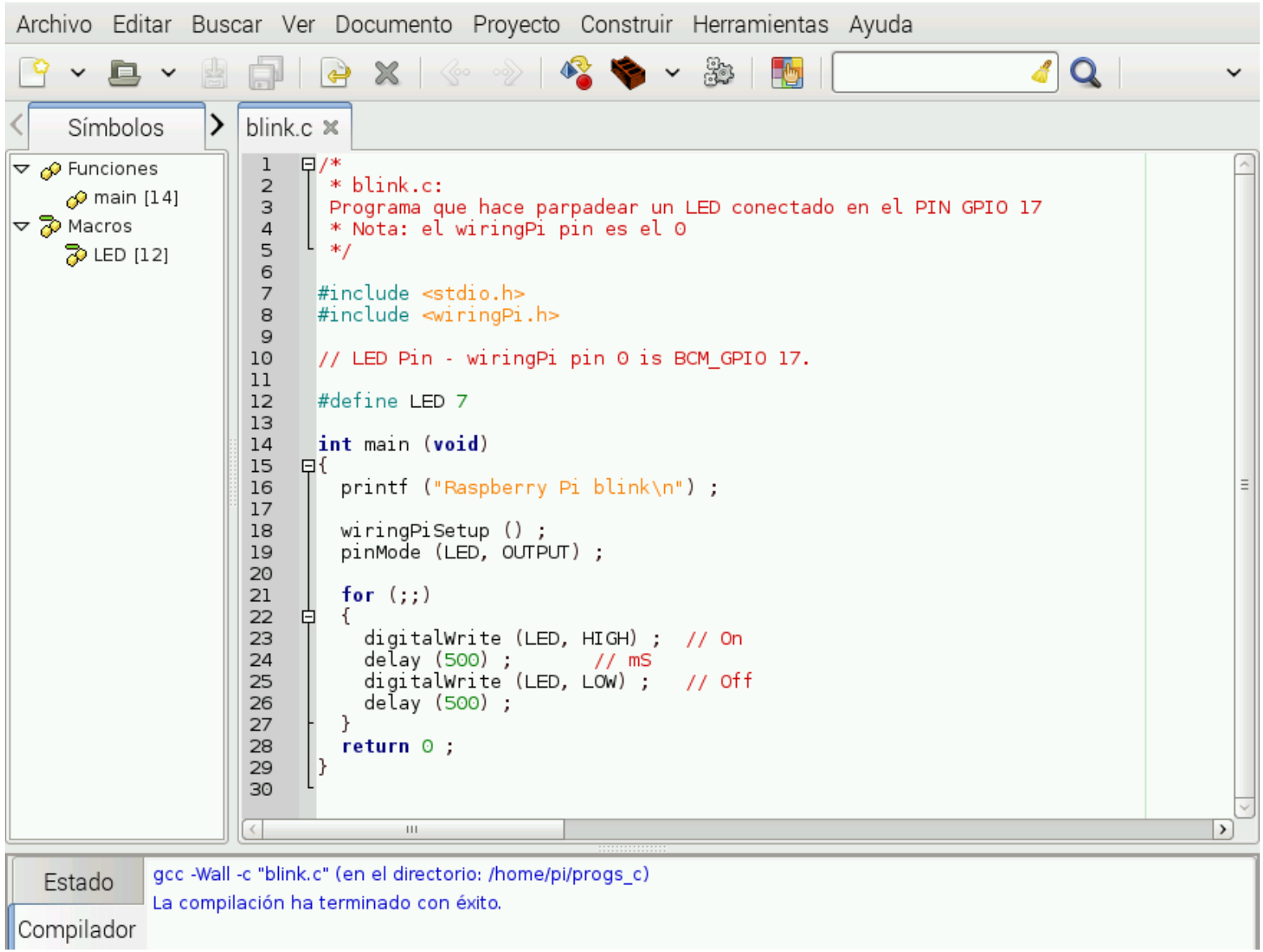
- Conectando los pines GPIO



USANDO LOS PINES GPIO

Se recomienda usar el Editor **Geany**

- **Ejemplo** de librerías GPIO (WiringPi) - **Parpadeo de un LED**



USANDO LOS PINES GPIO

- Compilando el programa

Se posiciona en la carpeta de trabajo (donde se encuentra su archivo fuente .c)

$ gcc -Wall -Winline -pipe -L/usr/local/lib blink.c -lwiringPi -lwiringPiDev -lpthread -lm -o blink

Nota: **blink.c** es el archivo fuente

blink es el archivo ejecutable

```
pi@raspberrypi ~/progs_c $ gcc -O3 -Wall -I/usr/local/include -Winline -pipe  -L/usr/local/lib
blink.c  -lwiringPi -lwiringPiDev -lpthread -lm -o blink
```

USANDO LOS PINES GPIO

- Compilando el programa

Se posiciona en la carpeta de trabajo (donde se encuentra su archivo fuente .c)

$ gcc -O3 -Wall -I/usr/local/include -Winline -pipe -L/usr/local/lib blink.c -lwiringPi -lwiringPiDev -lpthread -lm -o blink

Nota: **blink.c** es el archivo fuente

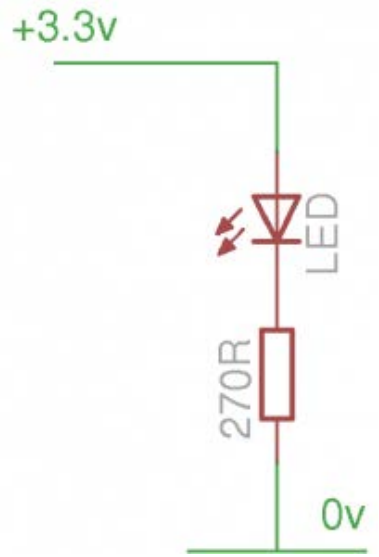
blink es el archivo ejecutable

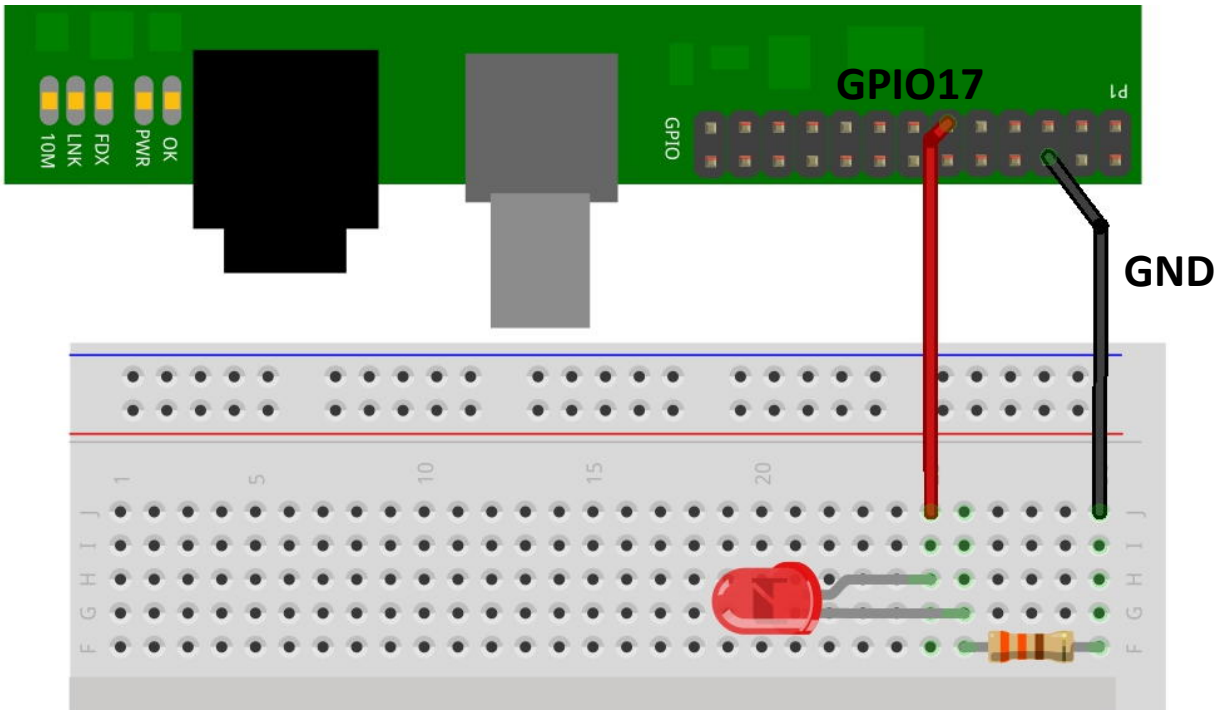
```
pi@raspberrypi ~/progs_c $ gcc -O3 -Wall -I/usr/local/include -Winline -pipe  -L/usr/local/lib
blink.c  -lwiringPi -lwiringPiDev -lpthread -lm -o blink
```

USANDO LOS PINES GPIO

- Conectando los pines GPIO – equivalencia WiringPi

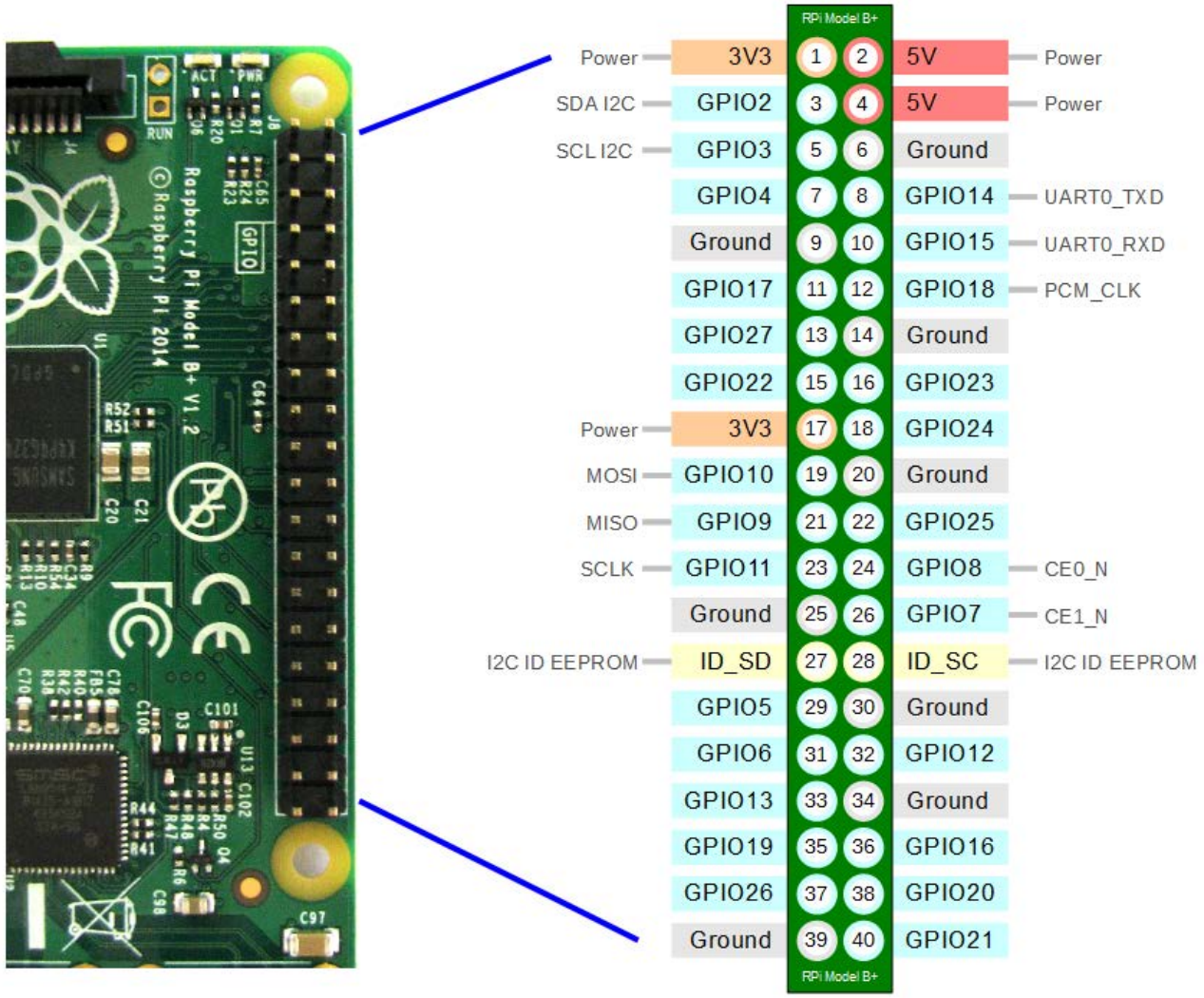
Blink.c





USANDO LOS PINES GPIO

- Conectando los pines GPIO



USANDO LOS PINES GPIO

- Identificando los pines GPIO – Equivalencia WiringPi Vs BCM

WiringPi: GPIO Pin Numbering Tables

http://wiringpi.com/

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

Note the differences between Revision 1 and Revision 2 Raspberry Pi's. The Revision 2 is readily identifiable by the presence of the 2 mounting holes.

The revision 2 Raspberry Pi has an additional GPIO connector, P5, which is next to the main P1 GPIO connector:

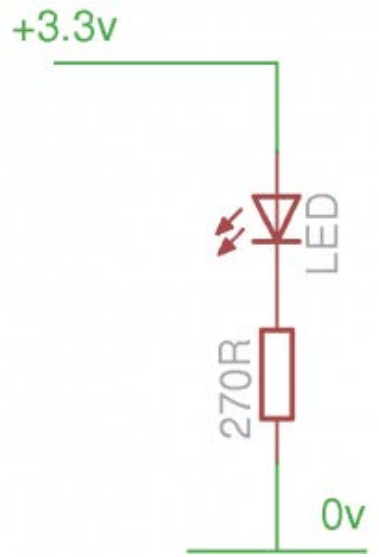
P5: Secondary GPIO connector (Rev. 2 Pi only)

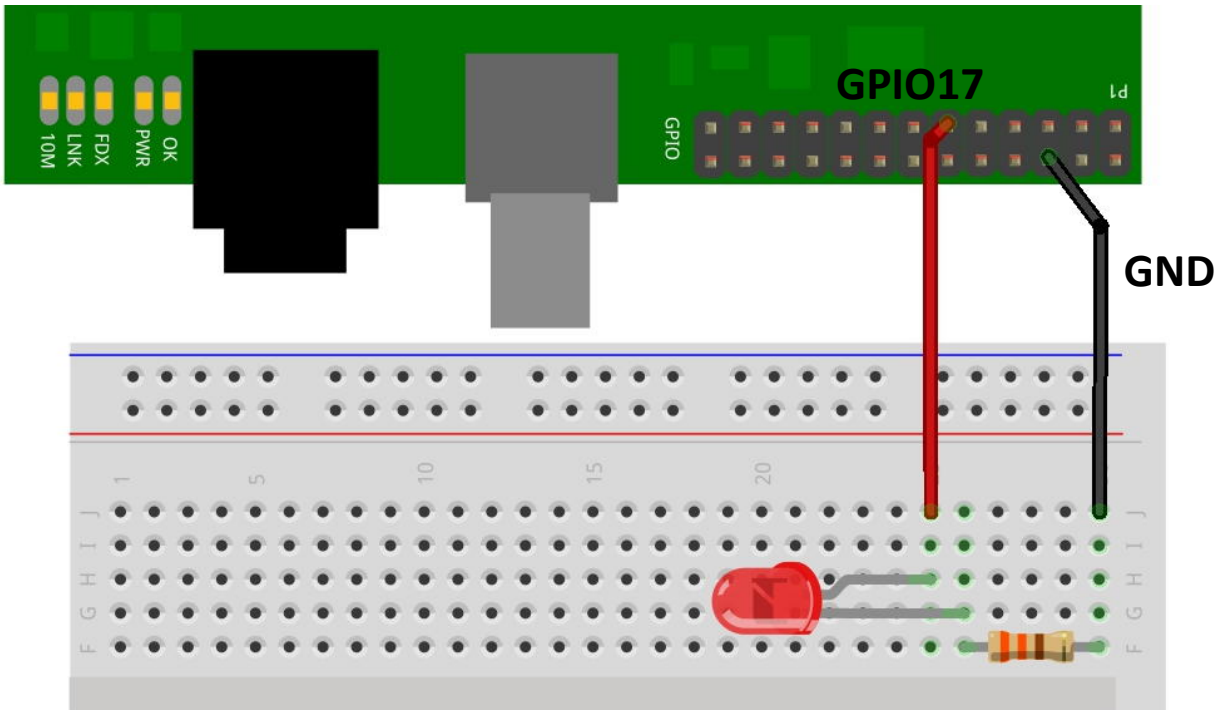
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		5v	1	2	3.3v		
17	28	GPIO8	3	4	GPIO9	29	18
19	30	GPIO10	5	6	GPIO11	31	20
		0v	7	8	0v		
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

USANDO LOS PINES GPIO

- Conectando los pines GPIO – equivalencia WiringPi

Blink.c





USANDO LOS PINES GPIO

- Secuencia de 8 LEDS

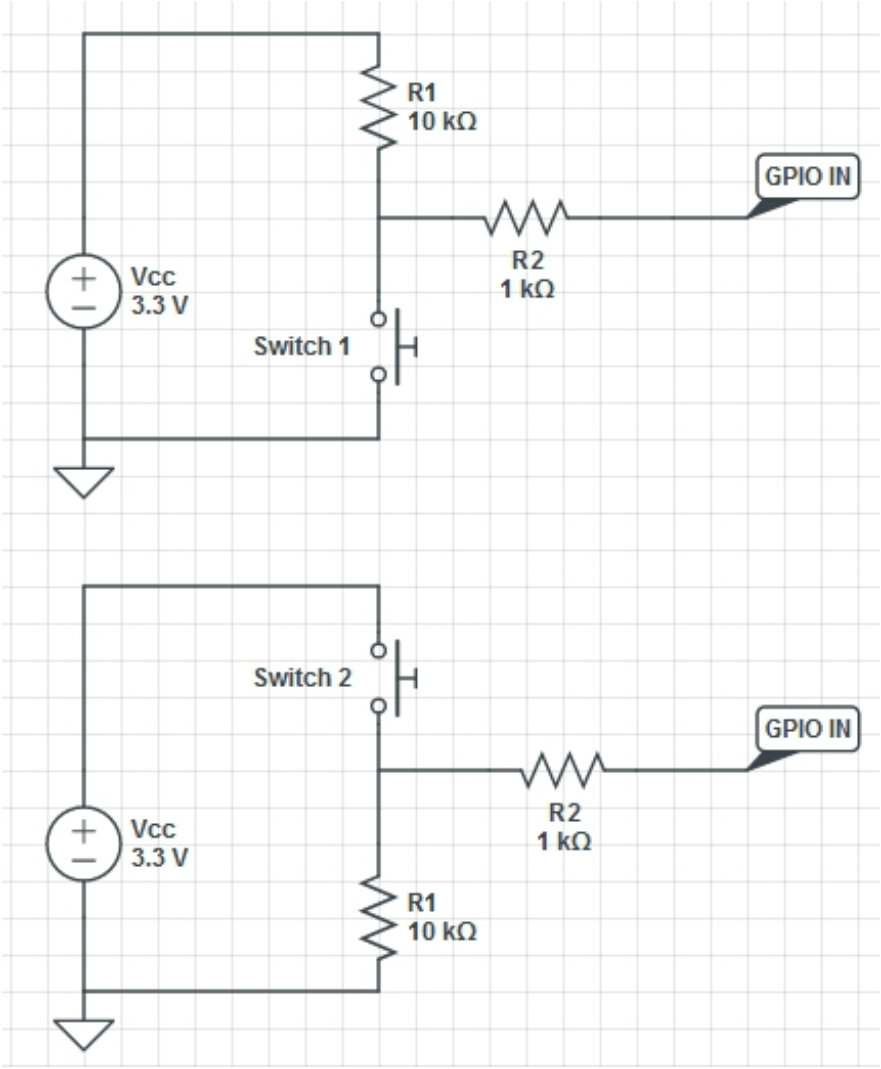
secuencia8.c

secuencia8.c

```c
/*
 * secuencia8.c:
 * Enciende una secuencia de LED utilizando los primeros
 * 8 WiringPi pines (GPIO) con un retardo inter secuencias
 * de 0.5 Segundos
 */

#include <stdio.h>
#include <wiringPi.h>

int main (void)
{
  int i, led ;
  clr();
  printf ("\n") ;
  printf ("Raspberry Pi - Secuencia de 8 LEDS\n") ;
  printf ("==================================\n") ;
  printf ("\n") ;
  printf ("Conecte 8 LEDS en los primeros 8 wiringPi pines.\n") ;

  wiringPiSetup () ;

  for (i = 0 ; i < 8 ; ++i)
    pinMode (i, OUTPUT) ;

  for (;;)
  {
    for (led = 0 ; led < 8 ; ++led)
    {
      digitalWrite (led, 1) ;
      delay (500) ;
    }

    for (led = 0 ; led < 8 ; ++led)
    {
      digitalWrite (led, 0) ;
      delay (500) ;
    }
  }
}
```

USANDO LOS PINES GPIO

- Pines GPIO como entradas



USANDO LOS PINES GPIO

- Pines GPIO como entradas

Observe la diferencia en:
wiringPisetup();

```c
#include <stdio.h>    // Used for printf() statements
#include <wiringPi.h> // Include WiringPi library!

// Pin number declarations. We're using the Broadcom chip pin numbers.
const int pwmPin = 18; // PWM LED - Broadcom pin 18, P1 pin 12
const int ledPin = 23; // Regular LED - Broadcom pin 23, P1 pin 16
const int butPin = 22; // Active-low button - Broadcom pin 22, P1 pin 15

const int pwmValue = 75; // Use this to set an LED brightness

int main(void)
{
    // Setup stuff:
    wiringPiSetupGpio(); // Initialize wiringPi -- using Broadcom pin numbers

    pinMode(pwmPin, PWM_OUTPUT); // Set PWM LED as PWM output
    pinMode(ledPin, OUTPUT);     // Set regular LED as output
    pinMode(butPin, INPUT);      // Set button as INPUT
    pullUpDnControl(butPin, PUD_UP); // Enable pull-up resistor on button

    printf("blinker is running! Press CTRL+C to quit.");

    // Loop (while(1)):
    while(1)
    {
        if (digitalRead(butPin)) // Button is released if this returns 1
        {
            pwmWrite(pwmPin, pwmValue); // PWM LED at bright setting
            digitalWrite(ledPin, LOW);     // Regular LED off
        }
        else // If digitalRead returns 0, button is pressed
        {
            pwmWrite(pwmPin, 1024 - pwmValue); // PWM LED at dim setting
            // Do some blinking on the ledPin:
            digitalWrite(ledPin, HIGH); // Turn LED ON
            delay(75); // Wait 75ms
            digitalWrite(ledPin, LOW); // Turn LED OFF
            delay(75); // Wait 75ms again
        }
    }

    return 0;
}
```

TAREA Prog en C #1

- Diseñar y construír una **alarma digital** utilizando la Raspberry Pi 2 y el lenguaje de programación en C

El sistema tendrá como entrada 5 sensores del tipo ON/OFF y una entrada para ACTIVAR/DESACTIVAR la alarma. Estas 5 entradas serán a través de los pines GPIO, pueden ser activadas en ALTO/BAJO, esto depende del tipo de sensor.

Cuando la alarma esté activada y cualquiera de los sensores se active, la alarma deberá encender una bocina o buzzer de 5V, una luz de emergencia (Led Rojo).

Cuando la alarma este desactivada, deberá ignorar todas las señales provenientes de los sensores.

TAREA Prog en C #2

- Diseñar y construír una arrancador secuencial de 6 motores corriente directa.

El sistema deberá tener un botón de **arranque** para que inicie el encendido secuencial y un botón de **paro** secuencial en orden inverso a la secuencia de arranque.

Además considere el uso de un botón de **paro de Emergencia** para apagar inmediatamente (al mismo tiempo) todos los motores.

Los motores deberán conectarse en los puertos GPIO por medio de una etapa de potencia.

El tiempo inter-secuencia es de 500 mS.

IV – PROGRAMACIÓN EN PYTHON PARA SISTEMAS EMPOTRADOS DE 32 BITS v 64 Bits

Presenta:

Dr. Everardo Inzunza González

Python

- **Python** es un **lenguaje de programación interpretado**, cuya filosofía hace hincapié en una sintaxis que favorezca el **código legible**.
- Se trata de un **lenguaje de programación multiparadigma**, ya que soporta orientación a objetos, programación imperativa y programación funcional.
- Es administrado por Python Software Foundation.
- Tiene licencia de **código abierto** (open source)
- **Es multiplataforma** (Se puede ejecutar en Windows, MAC, Linux).

Lenguajes de programación imperativa

- BASIC
- C
- Fortran
- Pascal
- Pauscal en español
- Perl
- PHP
- Lua
- Java
- Go

Lenguajes de programación funcional

- Java Script
- Java 8
- Scheme
- Erlang
- Rust
- Objective Caml
- Scala
- F#
- Haskell

Python es multiparadigma, ya que soporta Programación imperativa y funcional

Python

- Es un lenguaje de programación creado por **Guido Van Rossum** a principios de los años 90's **cuyo nombre está inspirado en el grupo de cómicos ingleses "Monty Python"**.

- Es un lenguaje similar a **Perl**, pero con una sintaxis muy limpia y que favorece un código legible.

- **Se trata de un lenguaje interpretado o de script, con tipado dinámico,** fuertemente tipado, multiplataforma y orientado a objetos.

Lenguaje interpretado o de script

- **Un lenguaje interpretado o de script** es aquel que **se ejecuta utilizando un programa intermedio llamado intérprete**, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados).

- **La ventaja de los lenguajes compilados** es que su ejecución es más rápida.

- Sin embargo los **lenguajes interpretados son más flexibles y más portables**.

Lenguaje interpretado o de script

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones.

Tipado dinámico

La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Fuertemente tipado

No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena `"9"` y el número `8`). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

¿Por qué Python?

Python es un lenguaje que todo el mundo debería conocer. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo, muy rápido.

La sintaxis de Python es tan sencilla y cercana al lenguaje natural que los Programas elaborados en Python parecen pseudocódigo.

Python no es adecuado sin embargo para la programación de bajo nivel o para aplicaciones en las que el rendimiento sea crítico.

Casos de éxito de Python

Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA, Industrias Light & Magic, y todas las distribuciones Linux, en las que Python cada vez representa un tanto por ciento mayor de los programas disponibles.

PROGRAMANDO EN PYTHON

El primer programa que vamos a escribir en Python es el clásico Hola Mundo, y en este lenguaje es tan simple como:

```
print "Hola Mundo"
```

Vamos a probarlo primero en el intérprete. Ejecuta python o ipython según tus preferencias, escribe la línea anterior y pulsa Enter. El intérprete responderá mostrando en la consola el texto `Hola Mundo`.

Escribiendo el primer programa en Python

Vamos ahora a crear un archivo de texto con el código anterior, de forma que pudiéramos distribuir nuestro pequeño gran programa entre nuestros amigos. Abre tu editor de texto preferido o bien el IDE que hayas elegido y copia la línea anterior. Guárdalo como hola.py, por ejemplo.

Ejecutar este programa es tan sencillo como indicarle el nombre del archivo a ejecutar al intérprete de Python

```
python hola.py
```

Escribiendo el primer programa en Python

Si utilizas Windows los archivos .py ya estarán asociados al intérprete de Python, por lo que basta hacer doble clic sobre el archivo para ejecutar el programa. Sin embargo como este programa no hace más que imprimir un texto en la consola, la ejecución es demasiado rápida para poder verlo si quiera. Para remediarlo, vamos a añadir una nueva línea que espere la entrada de datos por parte del usuario.

```
print "Hola Mundo"
raw_input()
```

De esta forma se mostrará una consola con el texto `Hola Mundo` hasta que pulsemos Enter.

Ejecutando en linux

También podríamos correr el programa desde la consola como si tratara de un ejecutable cualquiera:

```
./hola.py
```

```
$ python  nombre_archivo
```

Segundo programa en Python

programa2.py

```
1  #! /usr/bin/python
 2
 3 #Esto es un comentario
 4 #Importamos la librería time
 5 import time
 6 print("Hola Mundo")
 7 print("...")
 8 time.sleep(5)
 9 print("Pasados 5 segundos")
10 print("Hasta luego Mundo")
```

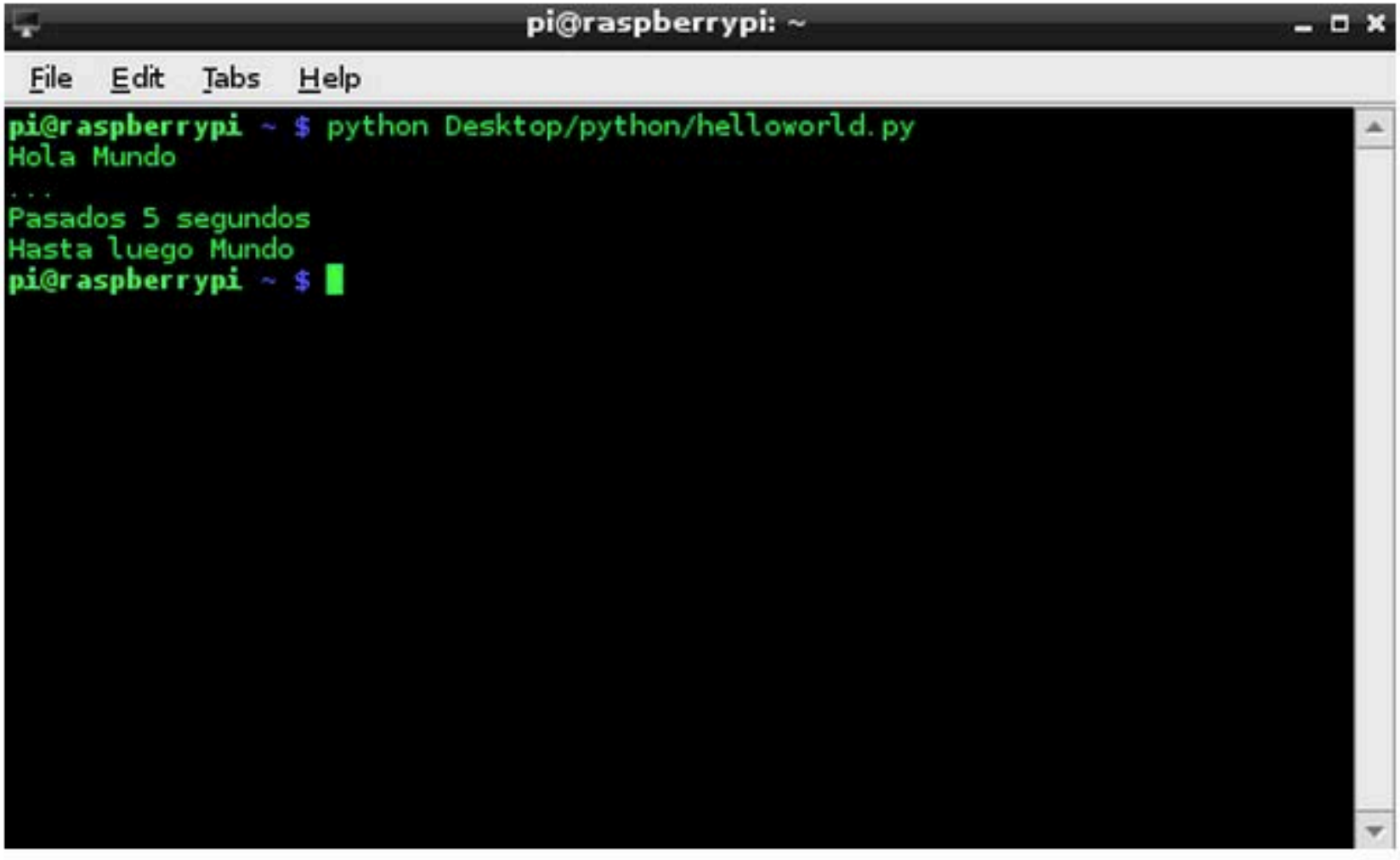
Shebang (#!)

Nota:

Es el par de caracteres que se encuentran al inicio de los Programas ejecutables Interpretados.

Para ejecutarlo

```
python ruta_archivo/nombre_archivo.py
```

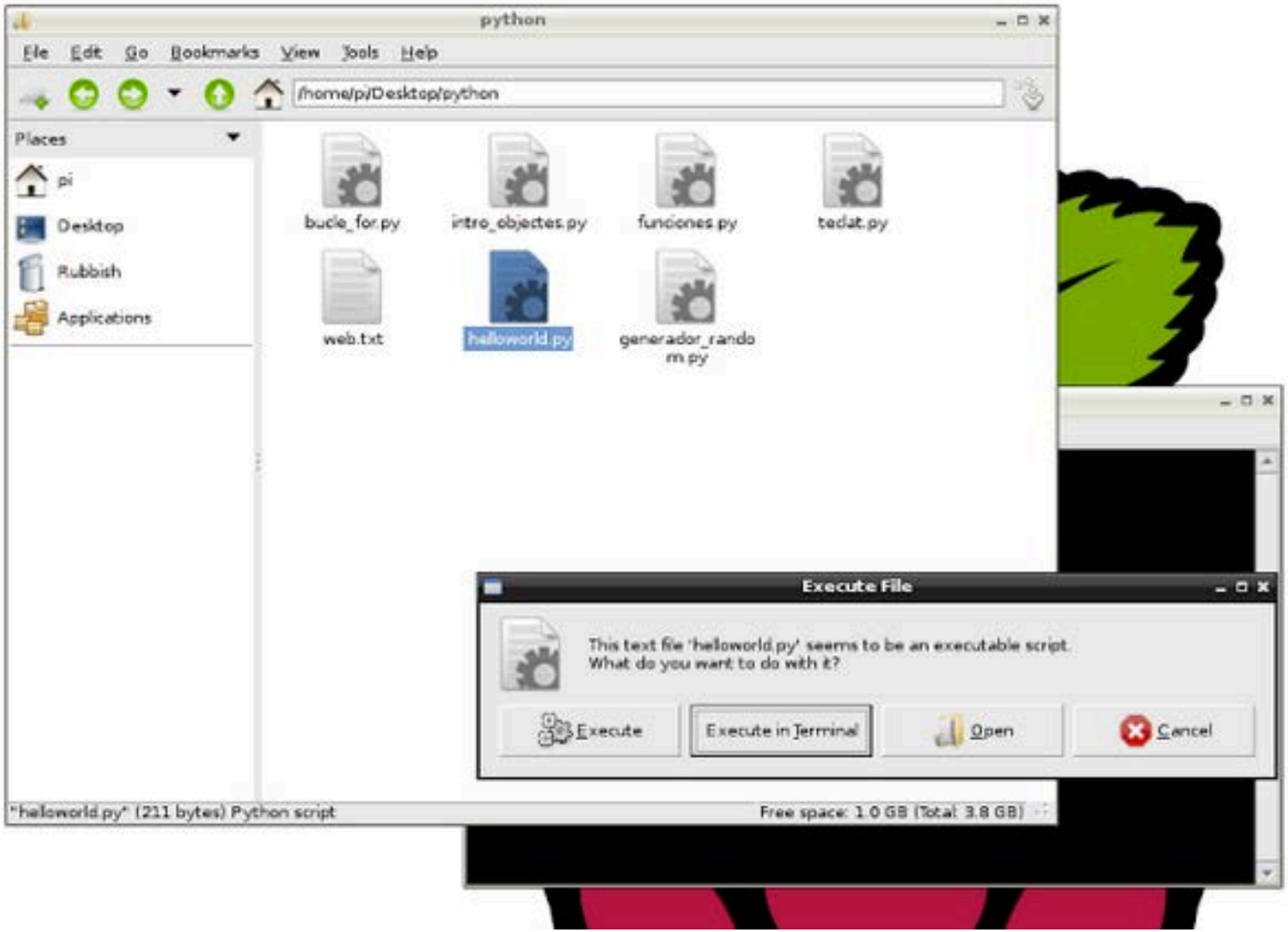


Haciendo ejecutable en linux

Si lo deseamos podemos configurar el archivo para que sea ejecutable directamente al pulsar sobre el fichero. Esto es posible gracias al "*shebang*" que pusimos en la primera linea. Un "*shebang*" no es más que un indicador que le indica a Linux con que programa debe ejecutar el script. Podemos poner diferentes opciones de optimización así como otros parámetros en el "*shebang*". Para poder ejecutar el archivo únicamente deberemos darle permisos de ejecución de este modo no hace falta llamar al interprete de Python. Podemos hacerlo usando el comando:

```
1 chmod +x ruta_archivo/nombre_archivo.py
```

Haciendo ejecutable en linux



Tipos básicos de datos en Python

En Python los tipos básicos se dividen en:

- Números, como pueden ser `3` (entero), `15.57` (de coma flotante) o `7 + 5j` (complejos)
- Cadenas de texto, como `"Hola Mundo"`
- Valores booleanos: `True` (cierto) y `False` (falso).

Vamos a crear un par de variables a modo de ejemplo. Una de tipo cadena y una de tipo entero:

```
# esto es una cadena
c = "Hola Mundo"

# y esto es un entero
e = 23

# podemos comprobarlo con la función type
type(c)
type(e)
```

Tipos de datos en Python - Reales

Reales

Los números reales son los que tienen decimales. En Python se expresan mediante el tipo `float`. En otros lenguajes de programación, como C, tenemos también el tipo `double`, similar a `float` pero de mayor precisión (double = doble precisión). Python, sin embargo, implementa su tipo `float` a bajo nivel mediante una variable de tipo `double` de C, es decir, utilizando 64 bits, luego en Python siempre se utiliza doble precisión, y en concreto se sigue el estándar IEEE 754: 1 bit para el signo, 11 para el exponente, y 52 para la mantisa. Esto significa que los valores que podemos representar van desde $\pm 2{,}2250738585072020 \times 10^{-308}$ hasta $\pm 1{,}7976931348623157 \times 10^{308}$.

Tipos de datos en Python - Reales

Para representar un número real en Python se escribe primero la parte entera, seguido de un punto y por último la parte decimal.

```
real = 0.2703
```

También se puede utilizar notación científica, y añadir una e (de exponente) para indicar un exponente en base 10. Por ejemplo:

```
real = 0.1e-3
```

sería equivalente a 0.1 x 10-3 = 0.1 x 0.001 = 0.0001

Tipos de datos en Python - Complejos

Los números complejos en Python se representan de la siguiente forma:

```
complejo = 2.1 + 7.8j
```

Operadores

Operadores aritméticos

Operador	Descripción	Ejemplo
+	Suma	`r = 3 + 2  # r es 5`
-	Resta	`r = 4 - 7  # r es -3`

Operador	Descripción	Ejemplo
-	Negación	`r = -7  # r es -7`
*	Multiplicación	`r = 2 * 6  # r es 12`
**	Exponente	`r = 2 ** 6  # r es 64`
/	División	`r = 3.5 / 2  # r es 1.75`
//	División entera	`r = 3.5 // 2 # r es 1.0`
%	Módulo	`r = 7 % 2  # r es 1`

Ejemplos

No obstante hay que tener en cuenta que si utilizamos dos operandos enteros, Python determinará que queremos que la variable resultado también sea un entero, por lo que el resultado de, por ejemplo, `3 / 2` y `3 // 2` sería el mismo: 1.

Si quisiéramos obtener los decimales necesitaríamos que al menos uno de los operandos fuera un número real, bien indicando los decimales

```
r = 3.0 / 2
```

```
r = float(3) / 2
```

Operadores a nivel de bit

Si no conocéis estos operadores es poco probable que vayáis a necesitarlos, por lo que podéis obviar esta parte. Si aún así tenéis curiosidad os diré que estos son operadores que actúan sobre las representaciones en binario de los operandos.

Por ejemplo, si veis una operación como `3 & 2`, lo que estais viendo es un and bit a bit entre los números binarios 11 y 10 (las representaciones en binario de 3 y 2).

El operador *and* (`&`), del inglés "y", devuelve 1 si el primer bit operando es 1 **y** el segundo bit operando es 1. Se devuelve 0 en caso contrario.

El resultado de aplicar and bit a bit a 11 y 10 sería entonces el número binario 10, o lo que es lo mismo, 2 en decimal (el primer dígito es 1 para ambas cifras, mientras que el segundo es 1 sólo para una de ellas).

Operadores a nivel de bit

El operador *or* (|), del inglés "o", devuelve 1 si el primer operando es 1 **o** el segundo operando es 1. Para el resto de casos se devuelve 0.

El operador *xor* u or exclusivo (^) devuelve 1 si uno de los operandos es 1 y el otro no lo es.

El operador *not* (~), del inglés "no", sirve para negar uno a uno cada bit; es decir, si el operando es 0, cambia a 1 y si es 1, cambia a 0.

Por último los operadores de desplazamiento (<< y >>) sirven para desplazar los bits n posiciones hacia la izquierda o la derecha.

Operadores a nivel de bit

Por último los operadores de desplazamiento (<< y >>) sirven para desplazar los bits n posiciones hacia la izquierda o la derecha.

Operador	Descripción	Ejemplo
&	and	r = 3 & 2 # r es 2
|	or	r = 3 | 2 # r es 3
^	xor	r = 3 ^ 2 # r es 1
~	not	r = ~3 # r es -4

<<	Desplazamiento izq.	r = 3 << 1 # r es 6
>>	Desplazamiento der.	r = 3 >> 1 # r es 1

Cadenas

Las cadenas no son más que texto encerrado entre comillas simples (`'cadena'`) o dobles (`"cadena"`). Dentro de las comillas se pueden añadir caracteres especiales escapándolos con `\`, como `\n`, el carácter de nueva línea, o `\t`, el de tabulación.

Ejemplo

```
a = "uno"
b = "dos"

c = a + b # c es "unodos"
c = a * 3 # c es "unounouno"
```

Booleanos

Estos son los distintos tipos de operadores con los que podemos trabajar con valores booleanos, los llamados operadores lógicos o condicionales:

Operador	Descripción	Ejemplo
and	¿se cumple a y b?	`r = True and False # r es False`
or	¿se cumple a o b?	`r = True or False  # r es True`
not	No a	`r = not True  # r es False`

Booleanos

Los valores booleanos son además el resultado de expresiones que utilizan operadores relacionales (comparaciones entre valores):

Operador	Descripción	Ejemplo
==	¿son iguales a y b?	`r = 5 == 3 # r es False`
!=	¿son distintos a y b?	`r = 5 != 3 # r es True`
<	¿es a menor que b?	`r = 5 < 3  # r es False`
>	¿es a mayor que b?	`r = 5 > 3  # r es True`
<=	¿es a menor o igual que b?	`r = 5 <= 5 # r es True`
>=	¿es a mayor o igual que b?	`r = 5 >= 3 # r es True`

CONTROL DE FLUJO

- **Sentencias condicionales**

if

La forma más simple de un estamento condicional es un `if` (del inglés si) seguido de la condición a evaluar, dos puntos (:) y en la siguiente línea e indentado, el código a ejecutar en caso de que se cumpla dicha condición.

```
fav = "mundogeek.net"
# si (if) fav es igual a "mundogeek.net"
if fav == "mundogeek.net":
    print "Tienes buen gusto!"
    print "Gracias"
```

Observe el uso de tabulación (indent) antes de los dos “print”

CONTROL DE FLUJO

- **Sentencias condicionales (If) – uso del indent**

Indent (Tab)

```
if fav == "mundogeek.net":
    print "Tienes buen gusto!"
print "Gracias"
```

CONTROL DE FLUJO

- **Sentencias condicionales (If) – uso del indent**

En otros lenguajes de programación los bloques de código se determinan encerrándolos entre llaves, y el indentarlos no se trata más que de una buena práctica para que sea más sencillo seguir el flujo del programa con un solo golpe de vista. Por ejemplo, el código anterior expresado en Java sería algo así:

```
String fav = "mundogeek.net";
if (fav.equals("mundogeek.net")){
    System.out.println("Tienes buen gusto!");
    System.out.println("Gracias");
}
```

CONTROL DE FLUJO

- **Sentencias condicionales (If… else)**

```
if fav == "mundogeek.net":
    print "Tienes buen gusto!"
    print "Gracias"

if fav != "mundogeek.net":
    print "Vaya, que lástima"
```

Uso de If y Else

```
if fav == "mundogeek.net":
    print "Tienes buen gusto!"
    print "Gracias"
else:
    print "Vaya, que lástima"
```

CONTROL DE FLUJO

- **Sentencias condicionales (If… elif…elif… else)**

```
if numero < 0:
    print "Negativo"
elif numero > 0:
    print "Positivo"
else:
    print "Cero"
```

`elif` es una contracción de *else if*, por lo tanto `elif numero > 0` puede leerse como "si no, si numero es mayor que 0". Es decir, primero se evalúa la condición del `if`. Si es cierta, se ejecuta su código y se continúa ejecutando el código posterior al condicional; si no se cumple, se evalúa la condición del `elif`. Si se cumple la condición del `elif` se ejecuta su código y se continua ejecutando el código posterior al condicional; si no se cumple y hay más de un `elif` se continúa con el siguiente en orden de aparición. Si no se cumple la condición del `if` ni de ninguno de los `elif`, se ejecuta el código del `else`.

CONTROL DE FLUJO

- **Sentencias condicionales (A if C else B)**

También existe una construcción similar al operador ? de otros lenguajes, que no es más que una forma compacta de expresar un `if else`. En esta construcción se evalúa el predicado C y se devuelve A si se cumple o B si no se cumple: *A* `if` *C* `else` *B*. Veamos un ejemplo:

```
var = "par" if (num % 2 == 0) else "impar"
```

Bucles ó ciclos

Mientras que los condicionales nos permiten ejecutar distintos fragmentos de código dependiendo de ciertas condiciones, los bucles nos permiten ejecutar un mismo fragmento de código un cierto número de veces, mientras se cumpla una determinada condición.

while

El bucle `while` (mientras) ejecuta un fragmento de código mientras se cumpla una condición.

```
edad = 0
while edad < 18:
    edad = edad + 1
    print "Felicidades, tienes " + str(edad)
```

CICLO while INFINITO

```
while True:
    entrada = raw_input("> ")
    if entrada == "adios":
        break
    else:
        print entrada
```

Otra forma de escribir el While infinito…

```
salir = False
while not salir:
    entrada = raw_input()
    if entrada == "adios":
        salir = True
    else:
        print entrada
```

CICLO for ... in

Ejemplo #1

```
secuencia = ["uno", "dos", "tres"]
for elemento in secuencia:
    print elemento
```

Ejemplo #2

```
1
2
3   # prueba ciclo for
4
5   secuencia=["uno","dos","tres"]
6   cont=0
7   for i in secuencia:
8       print i
9       cont=cont+1
10      print cont
11
```

```
pi@raspberrypi ~/progs_python $ python ciclofor.py
uno
dos
tres
pi@raspberrypi ~/progs_python $ python ciclofor.py
uno
1
dos
2
tres
3
```

CICLO for ... in

Ejemplo #3

```
1
2
3   # prueba ciclo for
4
5
6
7   num=10
8
9   for i in range(num):
10      print i
11
```

```
pi@raspberrypi ~/progs_python $ python ciclofor2.py
0
1
2
3
4
5
6
7
8
9
```

FUNCIONES

Una función es un fragmento de código con un nombre asociado que realiza una serie de tareas y devuelve un valor. A los fragmentos de código que tienen un nombre asociado y no devuelven valores se les suele llamar procedimientos. En Python no existen los procedimientos, ya que cuando el programador no especifica un valor de retorno la función devuelve el valor `None` (nada), equivalente al `null` de Java.

Además de ayudarnos a programar y depurar dividiendo el programa en partes las funciones también permiten reutilizar código.

En Python las funciones se declaran de la siguiente forma:

```
def mi_funcion(param1, param2):
    print param1
    print param2
```

FUNCIONES

También podemos encontrarnos con una cadena de texto como primera línea del cuerpo de la función. Estas cadenas se conocen con el nombre de *docstring* (cadena de documentación) y sirven, como su nombre indica, a modo de documentación de la función.

```
def mi_funcion(param1, param2):
    """Esta funcion imprime los dos valores pasados
    como parametros"""
    print param1
    print param2
```

Esto es lo que imprime el opeardor ? de iPython o la función `help` del lenguaje para proporcionar una ayuda sobre el uso y utilidad de las funciones. Todos los objetos pueden tener docstrings, no solo las funciones, como veremos más adelante.

FUNCIONES

Ejemplo #1

```
 1
 2
 3
 4  def funcion1(x,y):
 5      print x
 6      print y
 7
 8
 9  a=3
10  b=5
11  funcion1()
12
13
14  x=8
15  y=20
16  funcion1(x,y)
17
```

```
pi@raspberrypi ~/progs_python $ python funcion1.py
3
5
8
20
```

FUNCIONES

Para definir funciones con un número variable de argumentos colocamos un último parámetro para la función cuyo nombre debe precederse de un signo *:

Ejemplo #2

```
1  def varios(param1,param2, *otros):
2      for val in otros:
3          print val
4
5
6
7   varios("a","b")
8   varios("c","d","e")
9   varios("f","g","h","i")
```

```
pi@raspberrypi ~/progs_python $ python funcion2.py
e
h
i
```

FUNCIONES Variables Mutables e inmutables

Ejemplo #3

```
1   def f(x,y):
2       x=x+3
3       y.append(23)
4       print x, y
5
6
7   x=22
8   y= [22]
9
10  f(x,y)
11
12  print x,y
13
```

```
pi@raspberrypi ~/progs_python $ python funcion3.py
25 [22, 23]
22 [22, 23]
```

Como vemos la variable x no conserva los cambios una vez salimos de la función porque los enteros son inmutables en Python. Sin embargo la variable y si los conserva, porque las listas son mutables.

FUNCIONES Regreso de valores

Ejemplo #4

```
1   def sumar(x, y):
2       return x+y
3
4   print sumar(3,2)
5
```

```
pi@raspberrypi ~/progs_python $ python sumar.py
5
```

También podríamos pasar varios valores que retornar a return.

```
def f(x, y):
    return x * 2, y * 2

a, b = f(1, 2)
```

LISTAS

La lista es un tipo de colección ordenada. Sería equivalente a lo que en otros lenguajes se conoce por arrays, o vectores.

Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos, … y también listas.

Crear una lista es tan sencillo como indicar entre corchetes, y separados por comas, los valores que queremos incluir en la lista:

```
l = [22, True, "una lista", [1, 2]]
```

LISTAS - Accesando los elementos de una lista

Podemos acceder a cada uno de los elementos de la lista escribiendo el nombre de la lista e indicando el índice del elemento entre corchetes. Ten en cuenta sin embargo que el índice del primer elemento de la lista es 0, y no 1:

```
l = [11, False]
mi_var = l[0] # mi_var vale 11
```

LISTAS - Accesando los elementos de una lista

```
1
2   lista=[22, True, "Hola mundo",[2,3,]]
3
4   print lista
5   print lista[0]
6   print lista[1]
7   print lista[2]
8   print lista[3]
9
```

```
pi@raspberrypi ~/progs_python $ python lista1.py
[22, True, 'Hola mundo', [2, 3]]
22
True
Hola mundo
[2, 3]
```

LISTAS - Accesando los elementos de una lista

Si queremos acceder a un elemento de una lista incluida dentro de otra lista tendremos que utilizar dos veces este operador, primero para indicar a qué posición de la lista exterior queremos acceder, y el segundo para seleccionar el elemento de la lista interior:

```
l = ["una lista", [1, 2]]



mi_var = l[1][0] # mi_var vale 1
```

También podemos utilizar este operador para modificar un elemento de la lista si lo colocamos en la parte izquierda de una asignación:

```
l = [22, True]
l[0] = 99 # Con esto l valdrá [99, True]
```

LISTAS - Algunos detalles del uso de [] en Python

El uso de los corchetes para acceder y modificar los elementos de una lista es común en muchos lenguajes, pero Python nos depara varias sorpresas muy agradables.

Una curiosidad sobre el operador `[]` de Python es que podemos utilizar también números negativos. Si se utiliza un número negativo como índice, esto se traduce en que el índice empieza a contar desde el final, hacia la izquierda; es decir, con `[-1]` accederíamos al último elemento de la lista, con `[-2]` al penúltimo, con `[-3]`, al antepenúltimo, y así sucesivamente.

LISTAS - Algunos detalles del uso de [] en Python

Otra cosa inusual es lo que en Python se conoce como *slicing* o particionado, y que consiste en ampliar este mecanismo para permitir seleccionar porciones de la lista. Si en lugar de un número escribimos dos números `inicio` y `fin` separados por dos puntos (`inicio:fin`) Python interpretará que queremos una lista que vaya desde la posición `inicio` a la posición `fin`, sin incluir este último. Si escribimos tres números (`inicio:fin:salto`) en lugar de dos, el tercero se utiliza para determinar cada cuantas posiciones añadir un elemento a la lista.

```
l = [99, True, "una lista", [1, 2]]
mi_var = l[0:2]   # mi_var vale [99, True]
mi_var = l[0:4:2] # mi_var vale [99, "una lista"]
```

LISTAS - Algunos detalles del uso de [] en Python

Ejemplo 1

```
 1
 2   lista=[22, True, "Hola mundo",[2,3,],-55,100,50]
 3
 4   lista_parcial1=lista[0:3]
 5
 6   lista_parcial2=lista[0:5:2]
 7
 8   print lista_parcial1
 9   print lista_parcial2
10
```

```
pi@raspberrypi ~/progs_python $ python lista2.py
[22, True, 'Hola mundo']
[22, 'Hola mundo', -55]
```

Los números negativos también se pueden utilizar en un slicing, con el mismo comportamiento que se comentó anteriormente.

LISTAS - Algunos detalles del uso de [] en Python

```
l = [99, True, "una lista"]
mi_var = l[1:]  # mi_var vale [True, "una lista"]


mi_var = l[:2]  # mi_var vale [99, True]
mi_var = l[:]   # mi_var vale [99, True, "una lista"]
mi_var = l[::2] # mi_var vale [99, "una lista"]
```

LISTAS - Algunos detalles del uso de [] en Python

También podemos utilizar este mecanismo para modificar la lista:

```
l = [99, True, "una lista", [1, 2]]
l[0:2] = [0, 1] # l vale [0, 1, "una lista", [1, 2]]
```

TUPLAS

- Una tupla es una lista que no se va a poder modificar después de haberla creado.

- Las tuplas se definen como las listas salvo que en lugar de encerrarlas entre corchetes se encierran entre paréntesis.

- Los índices de las tuplas empiezan en 0 e índices negativos empiezan a contar desde el final de la lista.

TUPLAS

Todo lo que hemos explicado sobre las listas se aplica también a las tuplas, a excepción de la forma de definirla, para lo que se utilizan paréntesis en lugar de corchetes.

```
t = (1, 2, True, "python")
```

En realidad el constructor de la tupla es la coma, no el paréntesis, pero el intérprete muestra los paréntesis, y nosotros deberíamos utilizarlos, por claridad.

```
>>> t = 1, 2, 3
>>> type(t)
type "tuple"
```

TUPLAS

```
1  t=(1, 2, True, "python")
2
3  a=type (t)
4
5  print a
6  print t
7
```

```
pi@raspberrypi ~/progs_python $ python tupla1.py
<type 'tuple'>
(1, 2, True, 'python')
```

TUPLAS

Además hay que tener en cuenta que es necesario añadir una coma para tuplas de un solo elemento, para diferenciarlo de un elemento entre paréntesis.

```
>>> t = (1)
>>> type(t)

type "int"
>>> t = (1,)
>>> type(t)
type "tuple"
```

TUPLAS - Accesando los elementos de una Tupla

Para referirnos a elementos de una tupla, como en una lista, se usa el operador `[]`:

```
mi_var = t[0] # mi_var es 1
mi_var = t[0:2] # mi_var es (1, 2)
```

Podemos utilizar el operador `[]` debido a que las tuplas, al igual que las listas, forman parte de un tipo de objetos llamados secuencias.

TUPLAS - Accesando los elementos de una Tupla

```
#!/usr/bin/python
# Nombre de Fichero : tupla1.py

tupla = (1,[2,3,4],2,"Pedro");

for r in tupla:
    print str(r)

print "----------------------------"

for i in range(len(tupla)):
    print str(i),"->",str(tupla[i])

print "----------------------------"
print "El ultimo elemento de la tupla es %s" % tupla[-1]
print "----------------------------"
print "Una tupla el segundo y tercer",
print "elemento de la tupla original %s" % str(tupla[1:3])
print "Los tres ultimos elementos %s" % str(tupla[1:])
```

TUPLAS - Accesando los elementos de una Tupla

```
1
[2, 3, 4]
2
Pedro
-----------------------------
0 -> 1
1 -> [2, 3, 4]
2 -> 2
3 -> Pedro
-----------------------------
El ultimo elemento de la tupla es Pedro
-----------------------------
Una tupla el segundo y tercer elemento de la tupla original ([2, 3,
Los tres ultimos elementos ([2, 3, 4], 2, 'Pedro')
>>> list(tupla)
[1, [2, 3, 4], 2, 'Pedro']
>>> tuple(list(tupla))
(1, [2, 3, 4], 2, 'Pedro')
```

TUPLAS

- Las tuplas no se pueden modificar.

- Las tuplas se pueden trasformar a listas por medio del método list.
- Las listas se pueden trasformar a tuplas por medio del método tuple.
- La ventaja de las tuplas es que son más rápidas que las listas.

DIFDERENCIA ENTRE TUPLAS Y LISTAS

- Una lista no es lo mismo que una tupla.

- Ambas son un conjunto ordenado de valores, en donde este último puede ser cualquier objeto: un número, una cadena, una función, una clase, una instancia, etc.

- La diferencia es que las listas presentan una serie de funciones adicionales que permiten un amplio manejo de los valores que contienen.

- Basándonos en esta definición, puede decirse que **las listas son dinámicas**, mientras que las **tuplas son estáticas**.

DICCIONARIOS

Los diccionarios, también llamados matrices asociativas, deben su nombre a que son colecciones que relacionan una clave y un valor. Por ejemplo, veamos un diccionario de películas y directores:

```
d = {"Love Actually ": "Richard Curtis",
     "Kill Bill": "Tarantino",
     "Amélie": "Jean-Pierre Jeunet"}
```

El primer valor se trata de la clave y el segundo del valor asociado a la clave. Como clave podemos utilizar cualquier valor inmutable: podríamos usar números, cadenas, booleanos, tuplas, … pero no listas o diccionarios, dado que son mutables.

DICCIONARIOS

Los diccionarios, también llamados matrices asociativas, deben su nombre a que son colecciones que relacionan una clave y un valor. Por ejemplo, veamos un diccionario de películas y directores:

```
d = {"Love Actually ": "Richard Curtis",
     "Kill Bill": "Tarantino",
     "Amélie": "Jean-Pierre Jeunet"}
```

El primer valor se trata de la clave y el segundo del valor asociado a la clave. Como clave podemos utilizar cualquier valor inmutable: podríamos usar números, cadenas, booleanos, tuplas, … pero no listas o diccionarios, dado que son mutables.

DICCIONARIOS

La diferencia principal entre los diccionarios y las listas o las tuplas es que a los valores almacenados en un diccionario se les accede no por su índice, porque de hecho no tienen orden, sino por su clave, utilizando de nuevo el operador `[]`.

```
d["Love Actually "] # devuelve "Richard Curtis"
```

Al igual que en listas y tuplas también se puede utilizar este operador para reasignar valores.

```
d["Kill Bill"] = "Quentin Tarantino"
```

DICCIONARIOS -Ejemplo

```
1
2  d={"Love Actually ": "Richard Curtis",
3     "Kill Bill": "Tarantino",
4     "Amelie":  "Jean Pierre Jeunet"}
5
6  # Ejemplo de acceso
7
8  dato= d["Love Actually "]
9  print dato
10
11
12 # Ejemplo de asignacion
13
14 d["Kill Bill"]="Quentin Tarantino"
15
16 dato2=d["Kill Bill"]
17
18 print dato2
19
```

```
pi@raspberrypi ~/progs_python $ python dicciol.py
Richard Curtis
Quentin Tarantino
```

Matrices en Python (listas dentro de listas)

```
# La mas sencilla e intuitiva
matriz = []
for i in range(numero_filas):
    matriz.append([])
    for j in range(numero_columnas):
        matriz[i].append(None)
```

Matrices en Python (Ejemplo)

```
 1
 2   filas=4
 3   cols=3
 4
 5   matriz=[]
 6
 7   for i in range(filas):
 8       matriz.append([])
 9       for j in range(cols):
10           matriz[i].append(None)
11
12
13
14
15   print matriz
16
17
18   matriz2=[[1, 2, 3],[4, 5, 6], [7, 8 ,9]]
19
20   print matriz2
21
```

```
pi@raspberrypi ~/progs_python $ python matriz1.py
[[None, None, None], [None, None, None], [None, None, None], [None, None, None]]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

NÚMEROS EN PYTHON
Librería math

Se pueden manipular variables numéricas, de tipo:

INTEGERS, SHORTS, LONGS, FLOATS, y OTRAS.

```
>>> abs(-16)
```

```
>>> ceil(16.7)
```

will return an error, because the `ceiling` function is *not* in those default libraries.

```
>>> import math
>>> math.ceil(16.7)
```

NÚMEROS EN PYTHON
Librería math

Funciones especiales

math.ceil(*x*)

math.fabs(x)

math.factorial(x)¶

math.floor(x)

math.fmod(x, y)

math.frexp(*x*) mantisa y exponente

math.isnan(x)

Funciones de Potencia y logarítmicas

math.exp(*x*

math.log(x[, base])

math.log1p(x)

math.log10(x)

math.pow(x, y)

math.sqrt(x)

NÚMEROS EN PYTHON
Librería math

Funciones trigonométricas (radianes)

math.acos(x)

math.asin(x)

math.atan(x)

math.cos(x)

math.hypot(*x*, *y*)

math.sin(x

math.tan(x)

Funciones hiperbólicas

math.acosh(*x*)¶

math.asinh(*x*)

math.atanh(*x*)

math.cosh(*x*)¶

math.sinh(*x*)¶

math.tanh(x)

Conversión angular

math.degrees(x)

math.radians(x)¶

Constantes

math.pi

math.e

NÚMEROS EN PYTHON

Librería **numpy** **import numpy**

Es el corazón de las librerías para el cómputo científico en Python.

Proporciona un alto desempeño del arreglos multidimensionales.

Proporciona herramientas para trabajar con arreglos.

Es similar al manejo de arreglos con MATLAB.

Ver el link numpy para usuarios de MATLAB

http://scipy.github.io/old-wiki/pages/NumPy_for_Matlab_Users

NÚMEROS EN PYTHON

También existe la librería **numpy**

import numpy

Ejemplo para cálculo de la función mod

variable = numpy.fmod(dato,255)

NÚMEROS EN PYTHON

Manejo de arreglos con **numpy**

Creando arreglos

```
import numpy as np

a = np.array([1, 2, 3])  # Create a rank 1 array
print type(a)            # Prints "<type 'numpy.ndarray'>"
print a.shape            # Prints "(3,)"
print a[0], a[1], a[2]   # Prints "1 2 3"
a[0] = 5                 # Change an element of the array
print a                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])   # Create a rank 2 array
print b.shape                     # Prints "(2, 3)"
print b[0, 0], b[0, 1], b[1, 0]   # Prints "1 2 4"
```

NÚMEROS EN PYTHON

Manejo de arreglos con **numpy** **(Inicializando arreglos)**

```
import numpy as np

a = np.zeros((2,2))  # Create an array of all zeros
print a              # Prints "[[ 0.  0.]
                     #          [ 0.  0.]]"

b = np.ones((1,2))   # Create an array of all ones
print b              # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7) # Create a constant array
print c               # Prints "[[ 7.  7.]
                      #          [ 7.  7.]]"

d = np.eye(2)        # Create a 2x2 identity matrix
print d              # Prints "[[ 1.  0.]
                     #          [ 0.  1.]]"

e = np.random.random((2,2)) # Create an array filled with random values
print e                     # Might print "[[ 0.91940167  0.08143941]
                            #               [ 0.68744134  0.87236687]]"
```

NÚMEROS EN PYTHON

Manejo de arreglos con **numpy - Accesando arreglos**

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print a[0, 1]   # Prints "2"
b[0, 0] = 77    # b[0, 0] is the same piece of data as a[0, 1]
print a[0, 1]   # Prints "77"
```

NÚMEROS EN PYTHON

Manejo de arreglos con **numpy** **(accesando arreglos)**

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Two ways of accessing the data in the middle row of the array.
# Mixing integer indexing with slices yields an array of lower rank,
# while using only slices yields an array of the same rank as the
# original array:
row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
print row_r1, row_r1.shape  # Prints "[5 6 7 8] (4,)"
print row_r2, row_r2.shape  # Prints "[[5 6 7 8]] (1, 4)"

# We can make the same distinction when accessing columns of an array:
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print col_r1, col_r1.shape  # Prints "[ 2  6 10] (3,)"
print col_r2, col_r2.shape  # Prints "[[ 2]
                            #          [ 6]
                            #          [10]] (3, 1)"
```

NÚMEROS EN PYTHON

Manejo de arreglos con **numpy**

Operaciones artiméticas con Arreglos (elemento a elemento)

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print x + y
print np.add(x, y)

# Elementwise difference; both produce the array
# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print x - y
print np.subtract(x, y)

# Elementwise product; both produce the array
# [[ 5.0 12.0]
#  [21.0 32.0]]
print x * y
print np.multiply(x, y)

# Elementwise division; both produce the array
# [[ 0.2         0.33333333]
#  [ 0.42857143  0.5       ]]
print x / y
print np.divide(x, y)

# Elementwise square root; produces the array
# [[ 1.          1.41421356]
#  [ 1.73205081  2.        ]]
print np.sqrt(x)
```

NÚMEROS EN PYTHON

Manejo de arreglos con **numpy (Productos matriciales)**

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print v.dot(w)
print np.dot(v, w)

# Matrix / vector product; both produce the rank 1 array [29 67]
print x.dot(v)
print np.dot(x, v)

# Matrix / matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print x.dot(y)
print np.dot(x, y)
```

Blinking LED in Python

```
import RPi.GPIO as GPIO ❶
import time ❷

GPIO.setmode(GPIO.BCM) ❸
GPIO.setup(25, GPIO.OUT) ❹

while True: ❺
   GPIO.output(25, GPIO.HIGH) ❻
   time.sleep(1) ❼
   GPIO.output(25, GPIO.LOW) ❽
   time.sleep(1) ❾
```

❶ Import the code needed for GPIO control

❷ Import the code needed for for the sleep function

❸ Use the chip's signal numbers

❹ Set pin 25 as an output

❺ Create an infinite loop consisting of the indented code below it

❻ Turn the LED on

❼ Wait for one second

❽ Turn the LED off

❾ Wait for one second

Leyendo botón en Python

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN) ❶

count = 0 ❷

while True:
   inputValue = GPIO.input(24) ❸
   if (inputValue == True): ❹
      count = count + 1 ❺
      print("Button pressed " + str(count) + " times.") ❻
   time.sleep(.01) ❼
```

❶ Set pin 24 as an input

❷ Create a variable called `count` and store 0 in it.

❸ Save the value of pin 24 into `inputValue`

❹ Check if that value is True (when the button is pressed)

❺ If it is, increment the counter

❻ Print the text to the terminal

❼ Wait briefly, but let other programs have a chance to run by not hogging the processor's time

Leyendo botón en Python (Mejorado)

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN)

count = 0

while True:
    inputValue = GPIO.input(24)
    if (inputValue == True):
        count = count + 1
        print("Button pressed " + str(count) + " times.")
        time.sleep(.3) ❶
    time.sleep(.01)
```

❶ Helps a button press register only once

2da PARTE

Presenta:

Dr. Everardo Inzunza González

LIBRERÍA WIRING Pi en Python

Con Wiring pi, el cableado y las instrucciones para el manejo de los pines GPIO es muy similar que Arduino.

Esta librería permine maneja expansores de puertos de E/S

Link de la Versión 1 de la librería Wiring Pi

https://github.com/WiringPi/WiringPi-Python.git

Instalación de Wiring Pi 2

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install python-dev python-pip

sudo pip install wiringpi2

Verificando la instalación

sudo python

import wiringpi2

wiringpi2.piBoardRev()

```
pi@raspberrypi ~ $ sudo python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import wiringpi2
>>> wiringpi2.piBoardRev()
2
>>>
```

Configurando Wiring pi 2

import wiringpi2 as wiringpi

1. To initialise WiringPi pin mode

   ```
   wiringpi.wiringPiSetup()
   ```

2. To initialise GPIO mode

   ```
   wiringpi.wiringPiSetupGpio()
   ```

3. To initialise Physical P1 header pin mode

   ```
   wiringpi.wiringPiSetupPhys()
   ```

   # Solamente para Pi 1

Configurando los puertos:

wiringpi.pinMode(port_or_pin_number, X)

X = 0 for input,
X = 1 for output,
X = 2 for alternative function (e.g. PWM on port 18)

Ejemplos de configuraciones

```
01.  # GPIO port numbers
02.  import wiringpi2 as wiringpi
03.  wiringpi.wiringPiSetupGpio()
04.  wiringpi.pinMode(25, 0) # sets GPIO 25 to input
05.  wiringpi.pinMode(24, 1) # sets GPIO 24 to output
06.  wiringpi.pinMode(18, 2) # sets GPIO 18 to PWM mode
07.
08.  # wiringpi numbers
09.  import wiringpi2 as wiringpi
10.  wiringpi.wiringPiSetup()
11.  wiringpi.pinMode(6, 0) # sets WP pin 6 to input
12.  wiringpi.pinMode(5, 1) # sets WP pin 5 to output
13.  wiringpi.pinMode(1, 2) # sets WP pin 1 to PWM mode
14.
15.  # Physical P1 header pin numbers
16.  import wiringpi2 as wiringpi
17.  wiringPiSetupPhys()
18.  wiringpi.pinMode(22, 0) # sets P1 pin 22 to input
19.  wiringpi.pinMode(18, 1) # sets P1 pin 18 to output
20.  wiringpi.pinMode(12, 2) # sets P1 pin 12 to PWM mode
```

Leyendo pines de entradas

wiringpi.digitalRead(port_or_pin_number)

Por ejemplo leyendo el pin 25

wiringpi.digitalRead(25)

my_input = wiringpi.digitalRead(25)

```
01.  if my_input:
02.      print "Input on Port 25 is 1"
03.      # then you'd code what you want to happen when port 25 is HIGH
04.  else:
05.      print "Input on Port 25 is 0"
06.      # then you'd code what you want to happen when port 25 is LOW
```

Escribiendo datos binarios en las salidas GPIO

wiringpi.digitalWrite(port_or_pin_number, 0) # sets port/pin to 0 (0V)

wiringpi.digitalWrite(port_or_pin_number, 1) # sets port/pin to 1 (3.3V)

```
01. # GPIO port numbers
02. import wiringpi2 as wiringpi
03. from time import sleep
04. wiringpi.wiringPiSetupGpio()
05. wiringpi.pinMode(24, 1)       # sets GPIO 24 to output
06. wiringpi.digitalWrite(24, 0) # sets port 24 to 0 (0V, off)
07. sleep(10)                     # wait 10s
08. wiringpi.digitalWrite(24, 1) # sets port 24 to 1 (3V3, on)
09. sleep(10)                     # wait 10s
10. wiringpi.digitalWrite(24, 0) # sets port 24 to 0 (0V, off)
```

Ejemplo leyendo y escribiendo datos en los pines GPIO

```
01. import wiringpi2 as wiringpi
02. from time import sleep       # allows us a time delay
03. wiringpi.wiringPiSetupGpio()
04. wiringpi.pinMode(24, 1)      # sets GPIO 24 to output
05. wiringpi.digitalWrite(24, 0) # sets port 24 to 0 (0V, off)
06.
07. wiringpi.pinMode(25, 0)      # sets GPIO 25 to input
08. try:
09.     while True:
10.         if wiringpi.digitalRead(25):     # If button on GPIO25 pressed
11.             wiringpi.digitalWrite(24, 1) # switch on LED. Sets port 24 to 1 (3V3,
12.         else:
13.             wiringpi.digitalWrite(24, 0) # switch off LED. Sets port 24 to 0 (0V,
14.         sleep(0.05)                      # delay 0.05s
15.
16. finally:  # when you CTRL+C exit, we clean up
17.     wiringpi.digitalWrite(24, 0) # sets port 24 to 0 (0V, off)
18.     wiringpi.pinMode(24, 0)      # sets GPIO 24 back to input Mode
19.     # GPIO 25 is already an input, so no need to change anything
```

UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

Facultad de Ingeniería, Arquitectura y Diseño

Materia: **Sistemas Empotrados**

V Programación en Octave para sistemas embebidos.

VI Tratamiento digital de señales con Octave

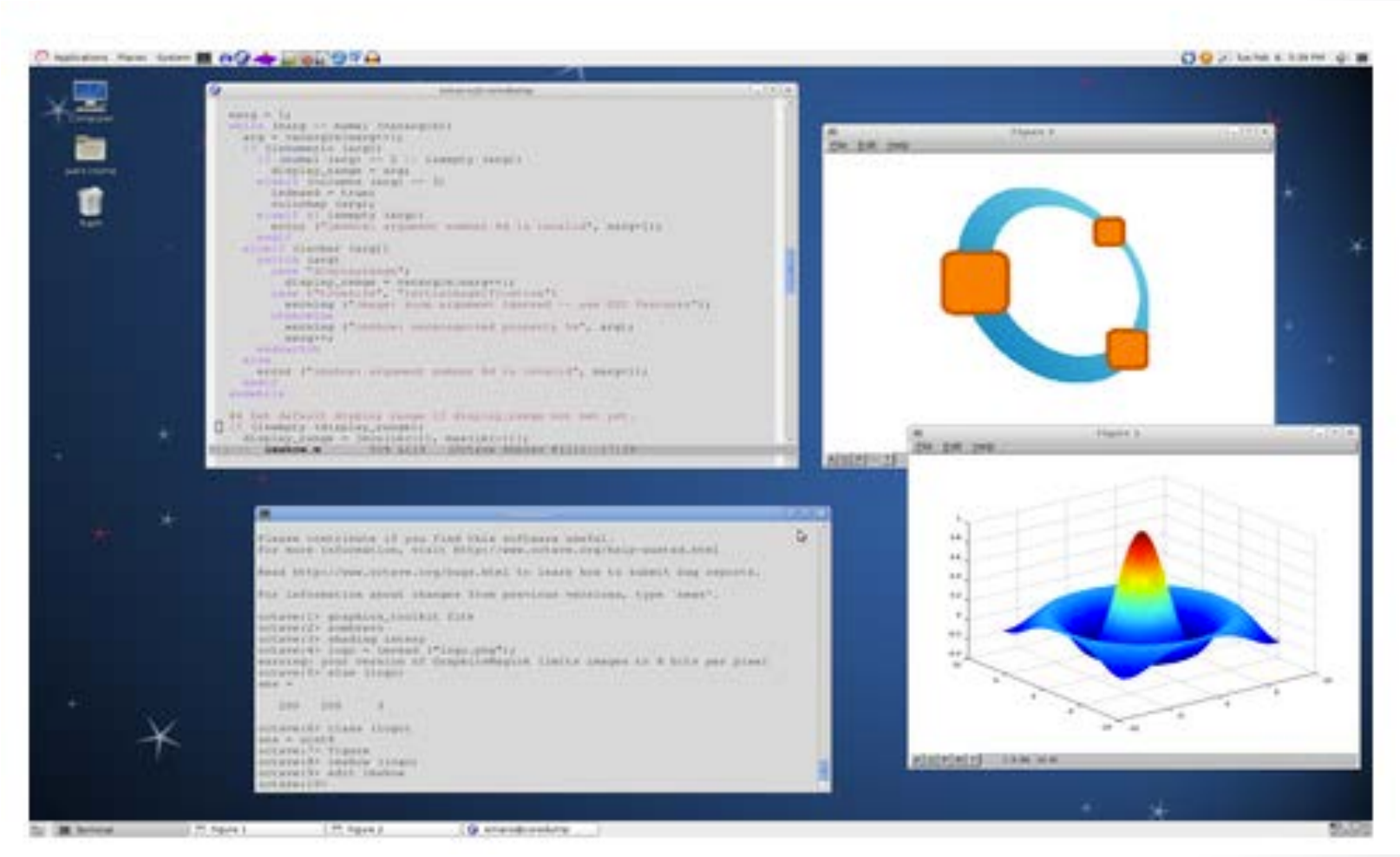
VII Procesamiento Digital de imágenes con Octave

Profesor

Dr. Everardo Inzunza González

Octave – Es un lenguaje interpretado de alto nivel para cómputo numérico.

Software libre equivalmente a MATLAB.



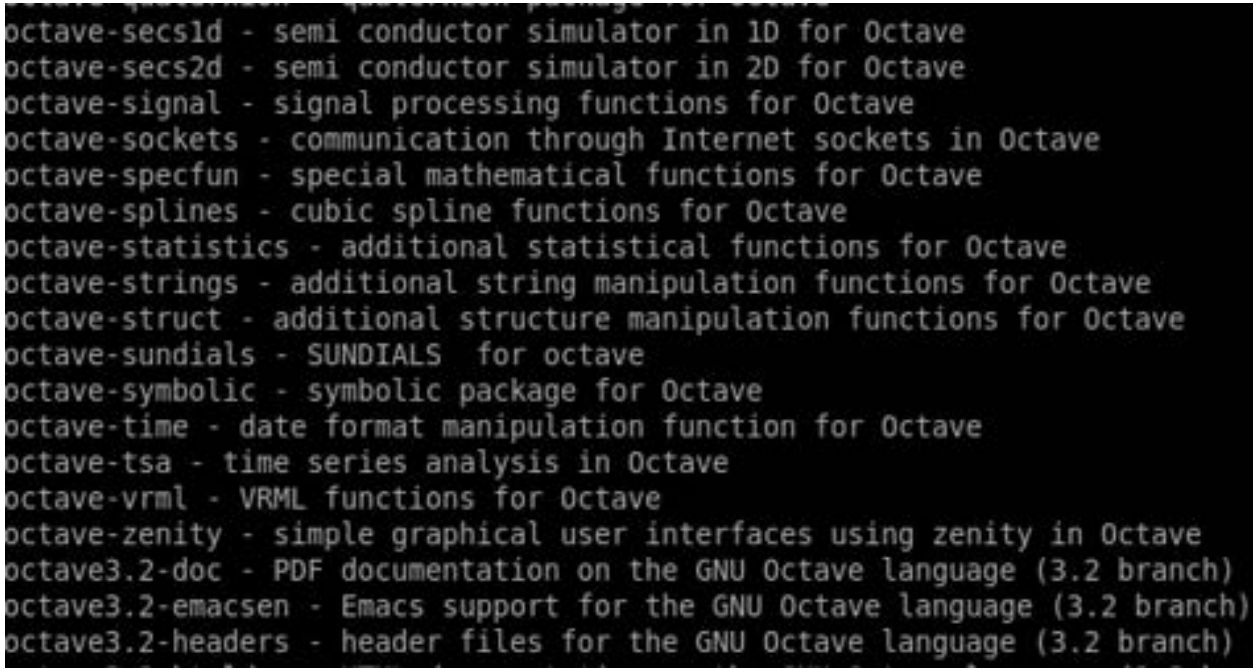
Instalación

$ sudo apt-get update

$ sudo apt-get upgrade

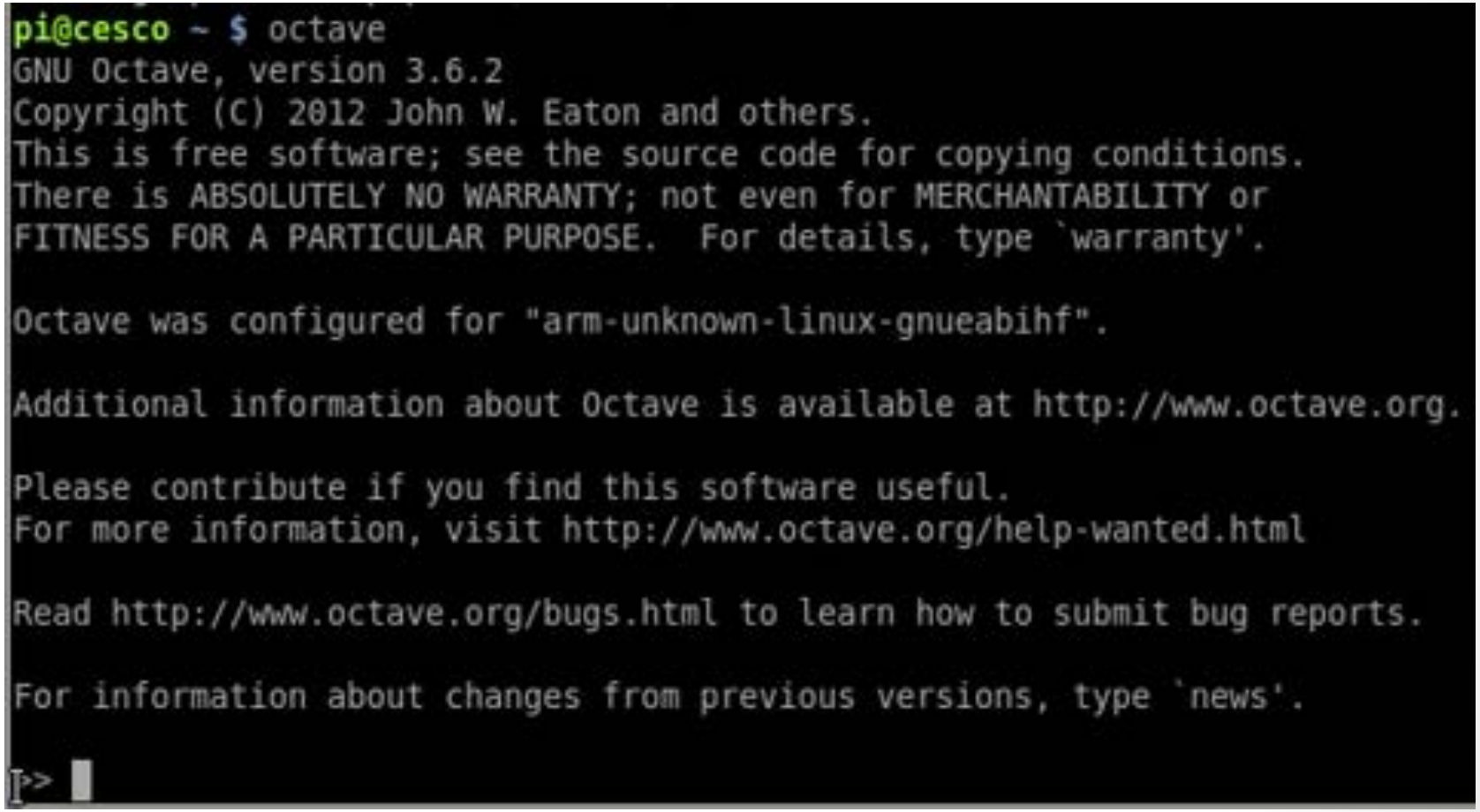
$ sudo apt-get install octave

$ sudo apt-cache search octave-



Instalación

$ sudo apt-get install octave-plplot octave-plot

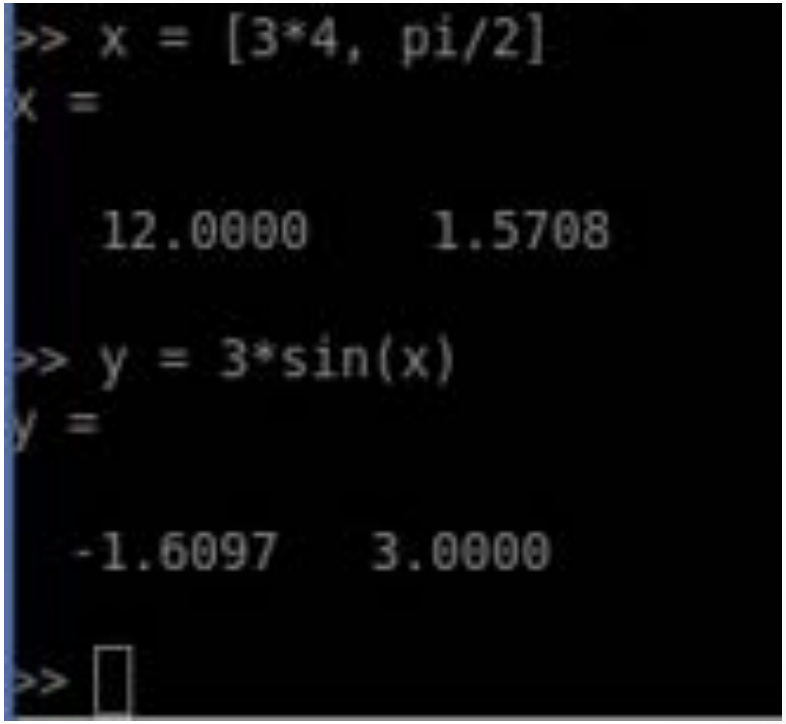


Ejemplos

- En Matlab

```
>> x = [3*4, pi/2]
x =
   12.0000    1.5708

>> y = 3*sin(x)
y =
   -1.6097    3.0000
```

En Octave



Ejemplos

- En Matlab

```
>> array = 1:2:9
array =
     1     3     5     7     9
```

En Octave

```
>> array = 1:2:9
array =

   1   3   5   7   9

>>
```

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

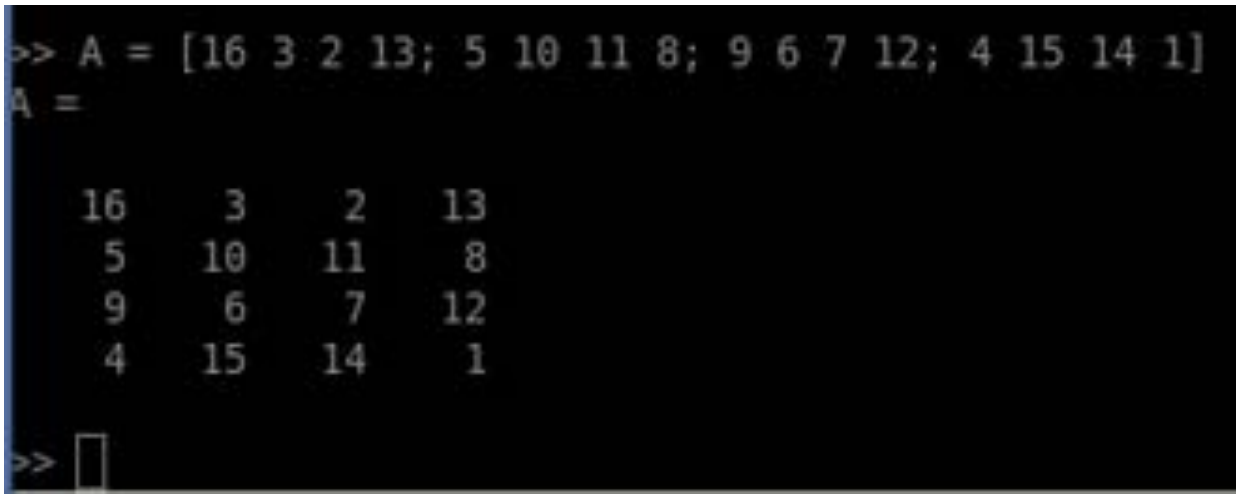
Ejemplos

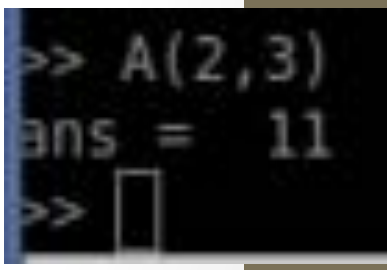
- En Matlab

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
 16  3  2 13
  5 10 11  8
  9  6  7 12
  4 15 14  1
```

```
>> A(2,3)
ans =
 11
```

- En Octave

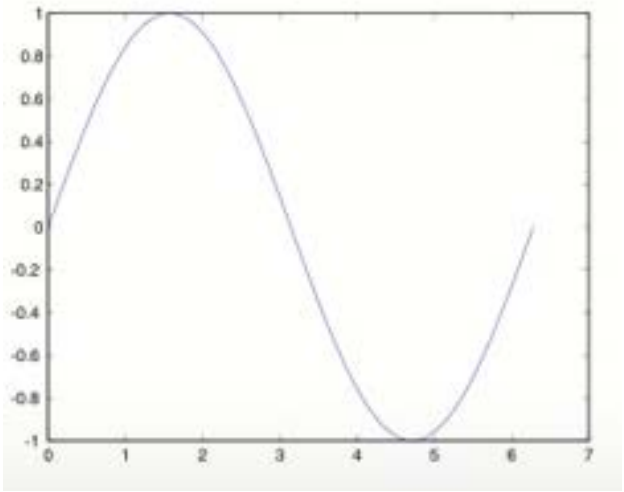




Ejemplos

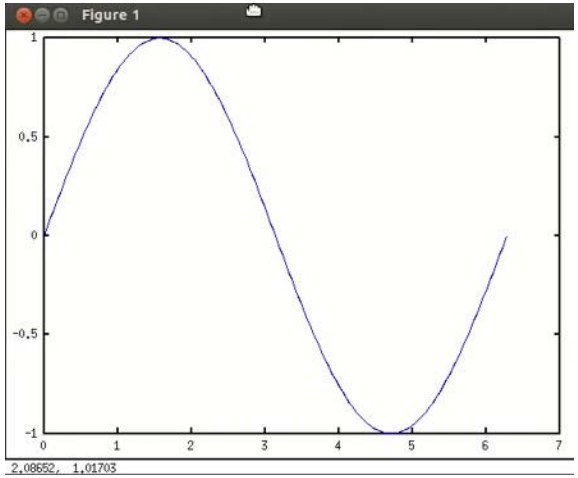
- En Matlab

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```



En Octave

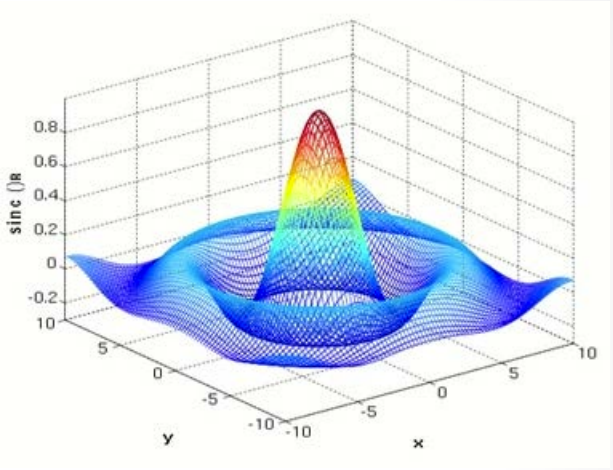
```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
>>
```



Ejemplos

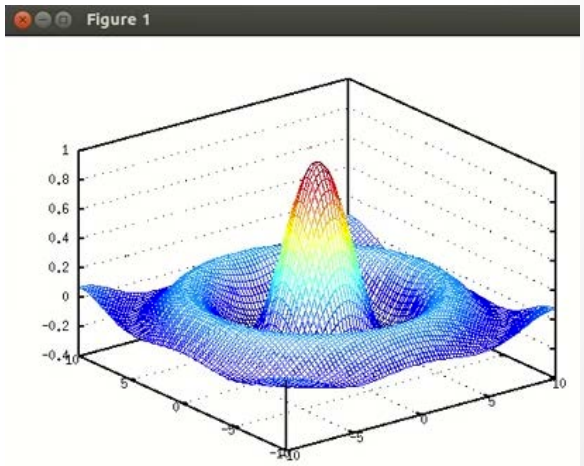
- En Matlab

```
[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
mesh(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
hidden off
```



- En Octave

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
>> [X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
>> f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
>> mesh(X,Y,f);
```



VI Tratamiento digital de señales con Octave

Procesamiento de audio en Octave

Instalar las siguientes librerías (toolboxes) de octave

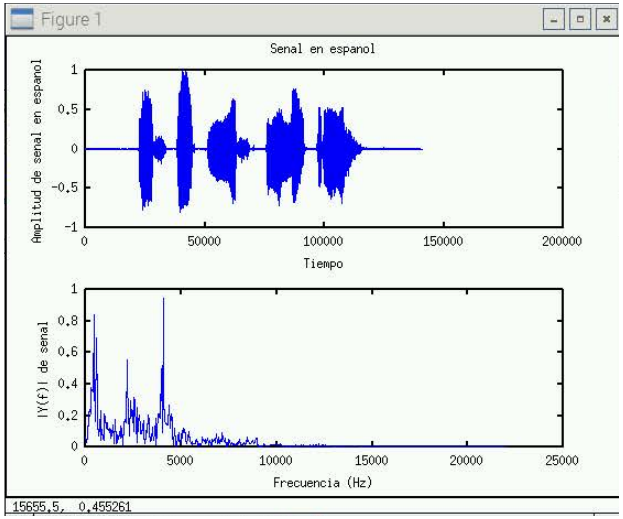
$ sudo apt –get update

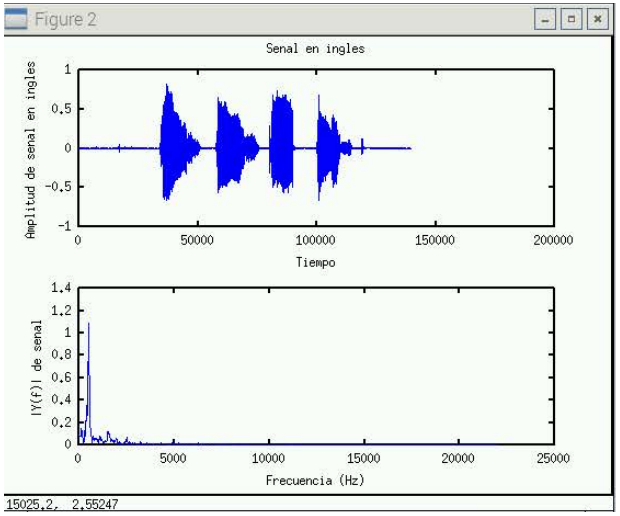
$ sudo apt-get upgrade

$ sudo apt-get install octave-**audio**

$ sudo apt-get install octave-**signal**

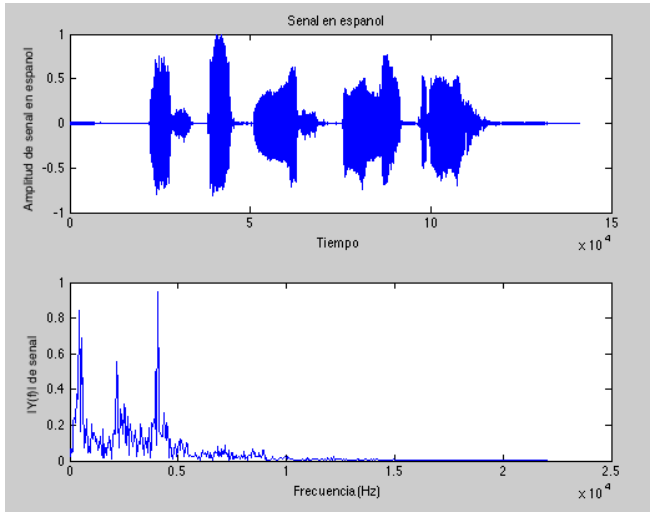
Procesamiento de audio en Octave

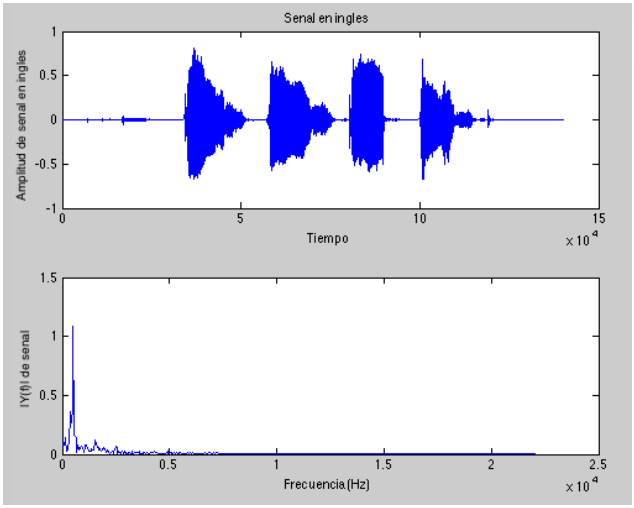




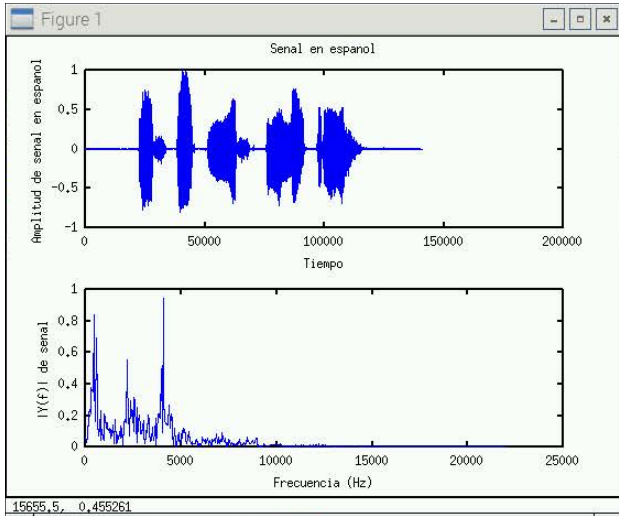

```
clear all
close all
clc

[y,Fs] = wavread('prueba1.wav'); % Lee audio 1 en espanol
nf=1024; %number of point in DTFT
        %y = datos, Fs = tasa de muestreo
        %sound(y,Fs); para reproducir el audio
acomodo= y'; % Calcula la transpuesta
subplot(2,1,1)
plot(acomodo)
title('Senal audio en espanol');
xlabel('Tiempo');
ylabel('Amplitud de senal de audio en espanol');

subplot(2,1,2)
Y = fft(y,nf);  % Calcula FFt de la senal de audio
f = Fs/2*linspace(0,1,nf/2+1); % calc eje de frecuencia
plot(f,abs(Y(1:nf/2+1))); % Grafica el espectro de frec
xlabel('Frecuencia (Hz)')
ylabel('|Y(f)| de senal')

[y,Fs] = wavread('prueba2.wav');  % Lee audio 2 ingles
nf=1024; %number of point in DTFT
        %y = datos, Fs = tasa de muestreo
        %sound(y,Fs); para reproducir el audio
acomodo= y';
figure(2)
subplot(2,1,1)
plot(acomodo)
title('Senal en ingles');
xlabel('Tiempo');
ylabel('Amplitud de senal en ingles');

subplot(2,1,2)
Y = fft(y,nf);
f = Fs/2*linspace(0,1,nf/2+1);
plot(f,abs(Y(1:nf/2+1)));
xlabel('Frecuencia (Hz)')
ylabel('|Y(f)| de senal')
```

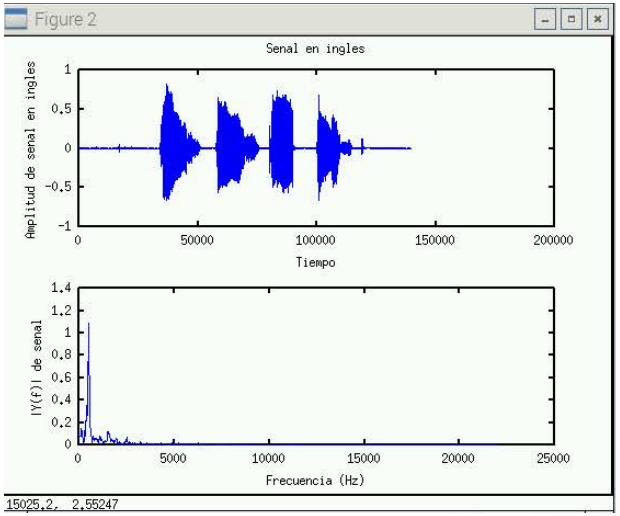
- En Matlab





- En Octave

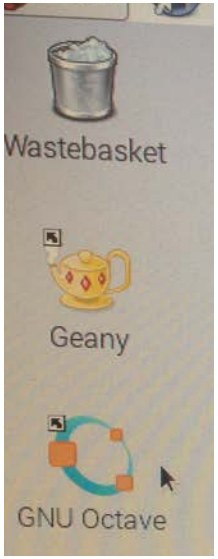




VII Procesamiento Digital de imágenes con Octave

Procesamiento de imágenes en octave

- **$ sudo apt –get update**
- **$ sudo apt-get upgrade**
- **$ sudo apt-get install octave-image**



Procesamiento de imágenes en octave

El procesamiento digital de imágenes sirve para mejorar la calidad o facilitar la búsqueda de información.

El proceso de filtrado obtiene imágenes mejorando ciertas características.

- Suavizar la imagen
- Eliminar ruido
- Realzar o detectar bordes

Imagen en escala de grises

Ejecutando unprograma en octave

- Octave:2>imagenesoctave

LXTerminal

- /home/pi - Geany

ew Document Project Build Tools Help

imagenesoctave.m

```
1   %Programa para manejo de imagenes en octave
2   warning off
3   clc
4   close all
5   clear all
6
7   imagen1=imread('lena.gif');%lee la imagen
8   imagen2=rgb2gray(imagen1);
9
10  subplot(1,2,1)
11  imshow(imagen1)
12  title('figura original')
13  subplot(1,2,2)
14  imshow(imagen2)
15  title('figura escala de gris')
```

Negativo de una imagen


```
1   %Programa para manejo de imagenes en octave
2   warning off
3   clc
4   close all
5   clear all
6
7   imagen1=imread('lena.gif');%lee la imagen
8   imagen2=imcomplement(imagen1);
9
10  subplot(1,2,1)
11  imshow(imagen1)
12  title('figura original')
13  subplot(1,2,2)
14  imshow(imagen2)
15  title('figura negativa')
```

Blanco y negro de una imagen


```
1   %Programa para manejo de imagenes en octave
2   warning off
3   clc
4   close all
5   clear all
6
7   imagen1=imread('lena.gif');%lee la imagen
8   threshold = graythresh(imagen1);
9   imagen2=im2bw(imagen1,threshold);
10
11  subplot(1,2,1)
12  imshow(imagen1)
13  title('figura original')
14  subplot(1,2,2)
15  imshow(imagen2)
16  title('figura en blanco y negro')
17
```

Operaciones aritméticas con imágenes

Suma de imagenes

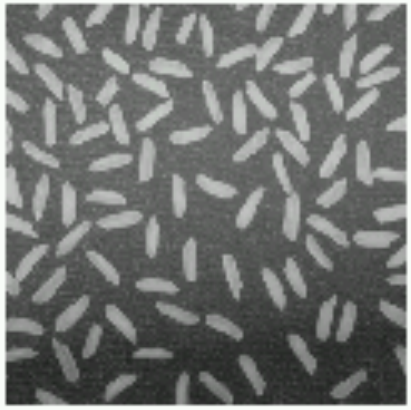
```
1
2   warning off
3   clc
4   close all
5   clear all
6
7   imagen1=imread('cameraman.tiff');  % Lee imagen1
8   imagen1num=double(imagen1);  % Convierte a numerica
9   imagen2=imread('rice.jpg');  % Lee imagen 2
10  imagen2num=double(imagen2);  % convierte a numerica
11
12  suma=imagen1num+imagen2num;  % suma las dos imagenes
13
14  suma_uint8=uint8(suma);  % convierte a UINT8
15
16
17
18  subplot(1,3,1)
19  imshow(imagen1)
20  title('Imagen Camera Man');
21  subplot(1,3,2)
22  imshow(imagen2)
23  title('Imagen de arroz');
24  subplot(1,3,3)
25  imshow(suma_uint8)
26  title('Suma de imagenes');
```

Operaciones aritméticas con imágenes

Resultado

Imagen Camera Man

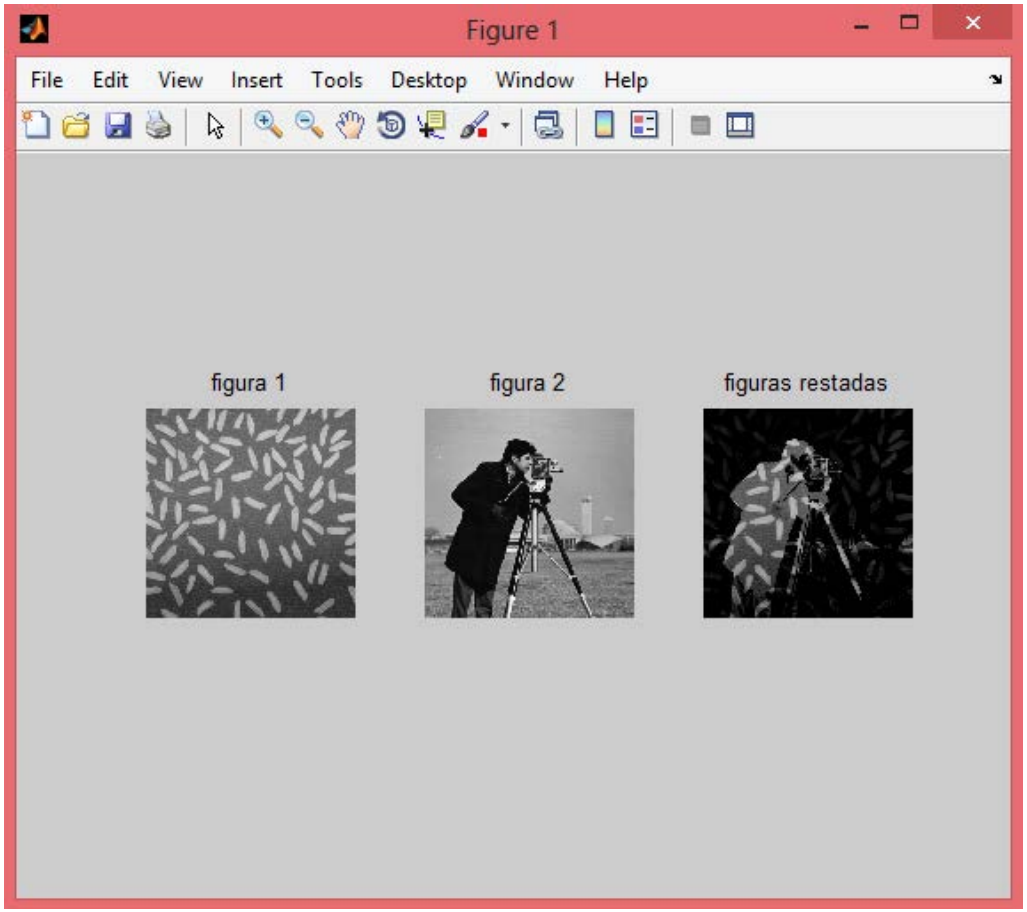
Imagen de arroz



Suma de imagenes

Operaciones aritméticas con imágenes

TAREA: CALCULAR FIGURA1-FIGURA2



Histograma de una imagen

Permite conocer la distribución de los niveles de gris y el contraste que presenta. El histograma de una imagen digital con L niveles de gris en el rango [0,L-1] es una función discreta de la forma:

$$h(r_k) = \frac{n_k}{N}$$

Donde:

r_k *es el* $k-$*eximo nivel de gris*

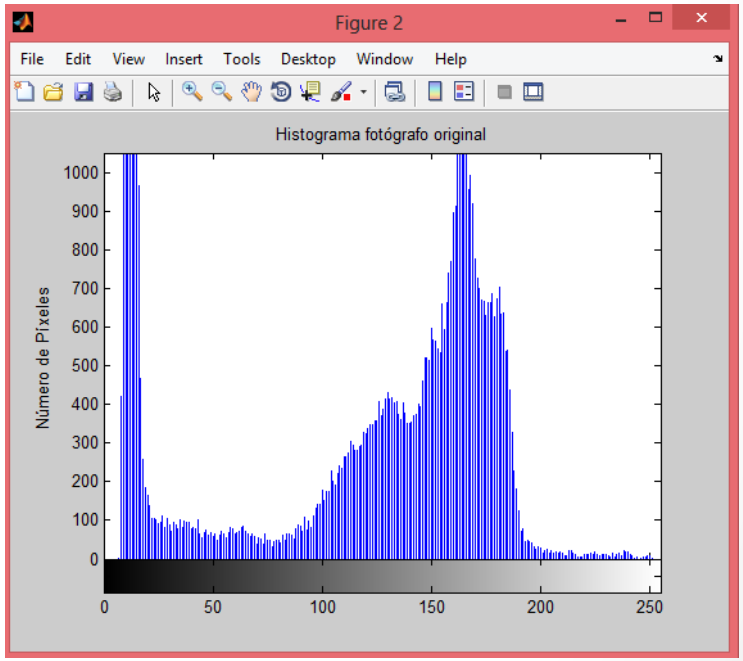
n_k *es el número de píxeles en la imagen con el nivel de gris* r_k

N *es el número total de píxel de la imagen*

$k = 0,1,2,...,L-1$ *niveles de gris*

Histograma de una imagen

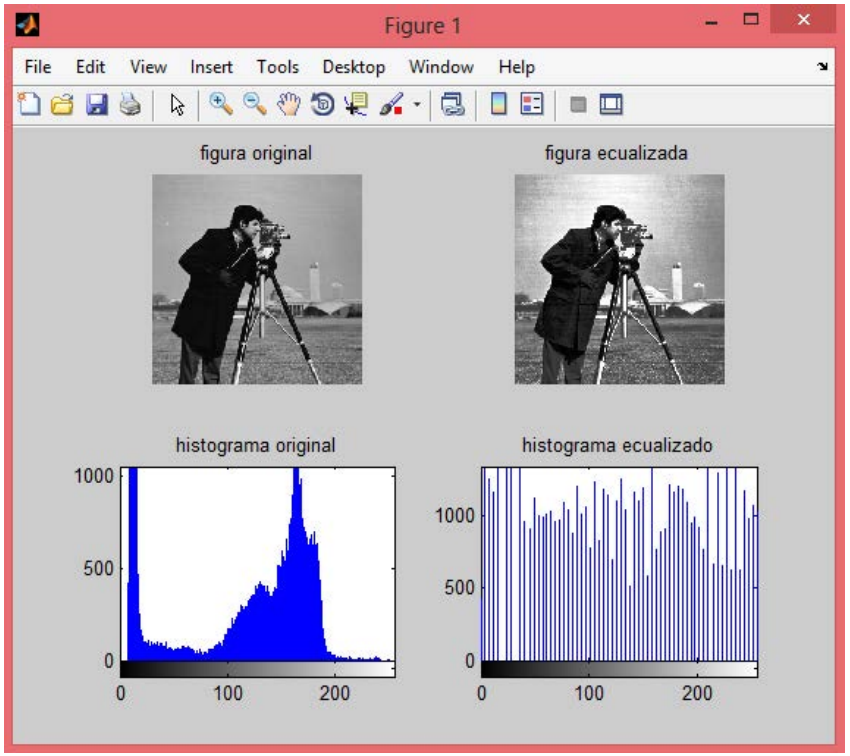
```
img=imread('cameraman.tif');
figure(1)
imshow(img)
figure(2)
Imgnum=double(img);
imhist(imgnum)
```



Ecualización del histograma

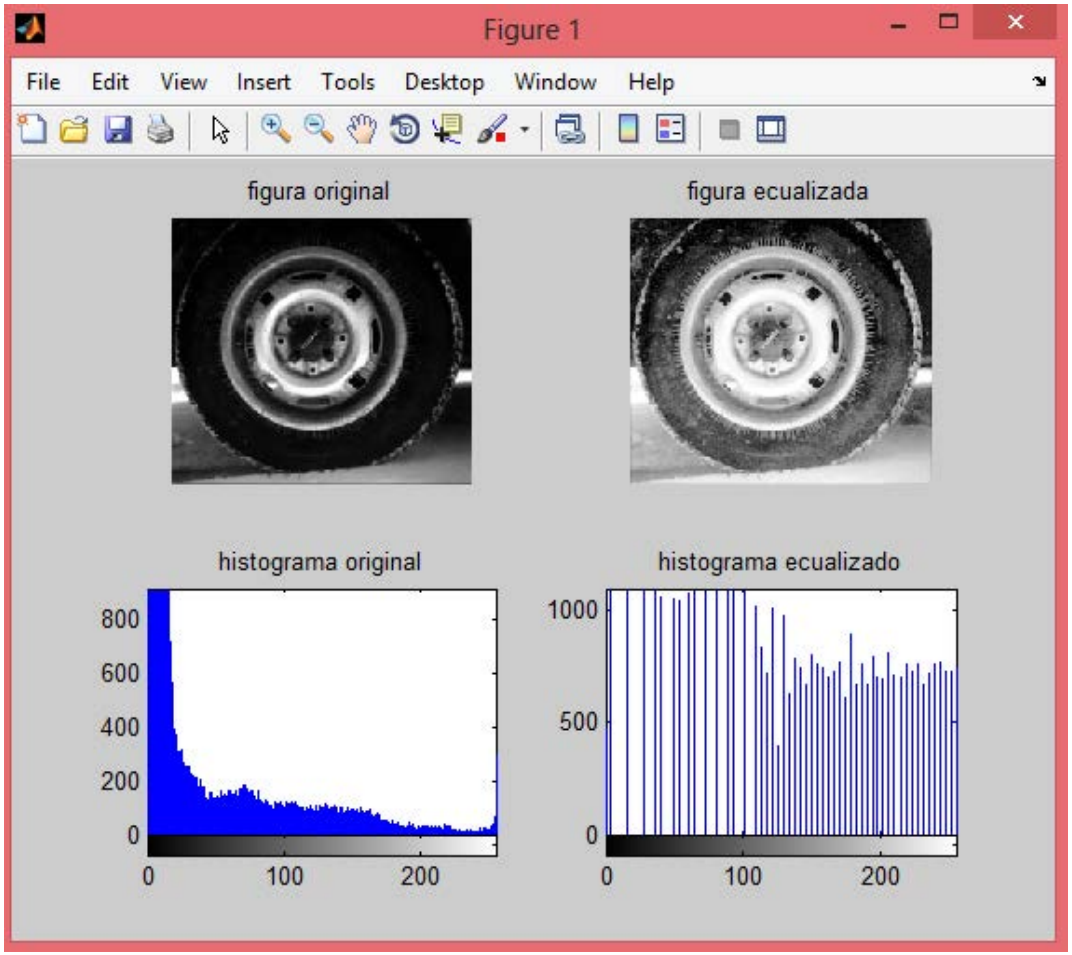
El fundamento se basa en realizar una conversión en el histograma de forma que la probabilidad de cualquier nivel de gris en la imagen sea idéntica.

$$F(r') = F(r) \rightarrow (r'+1)\frac{1}{I} = F(r) \rightarrow r' = F(r)\cdot I - 1$$ *Siendo* $F(r')$ *la función de distribución uniforme.*




```
img=imread('cameraman.tif');
subplot(2,2,1)
imshow(img)
title('figura original')
subplot(2,2,3)
imhist(img);
title('histograma original')
subplot(2,2,4)
imagen_ec=histeq(img);
imhist(imagen_ec)
title('histograma ecualizado')
subplot(2,2,2)
imshow(imagen_ec)
title('figura ecualizada')
```

Ecualización del histograma

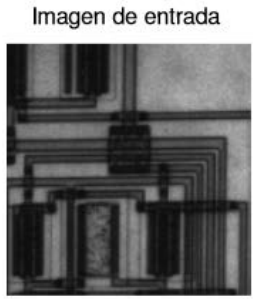


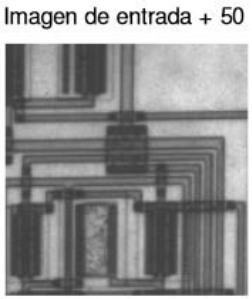

```
img=imread('tire.tif');
subplot(2,2,1)
imshow(img)
title('figura original')
subplot(2,2,3)
imhist(img);
title('histograma original')
subplot(2,2,4)
imagen_ec=histeq(img);
imhist(imagen_ec)
title('histograma ecualizado')
subplot(2,2,2)
imshow(imagen_ec)
title('figura ecualizada')
```

Brillo de una imagen

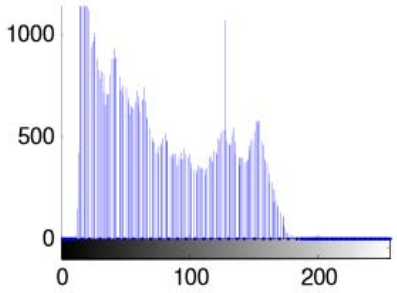
El brillo en una imagen nos indica la intensidad de los números digitales.
El brillo de una imagen está en función de:

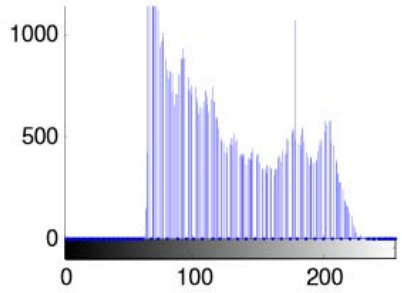
$$g(x,y) = f(x,y) + brillo$$





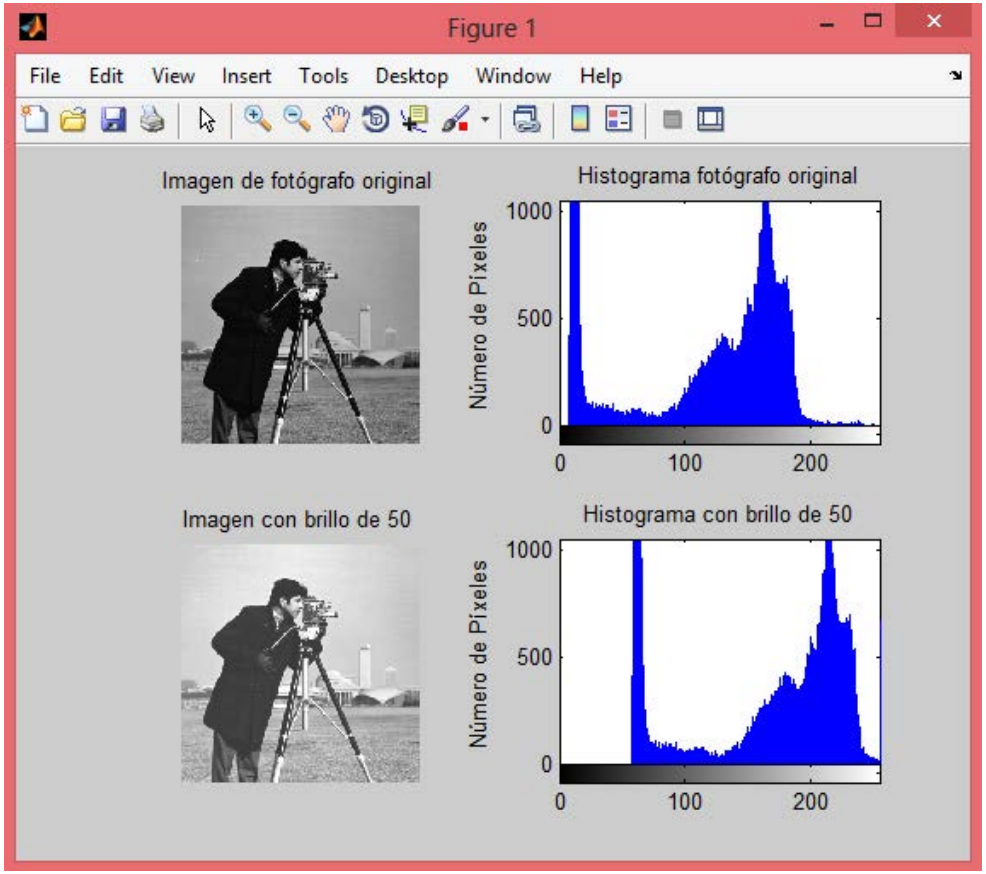
Donde brillo es constante para toda la imagen.





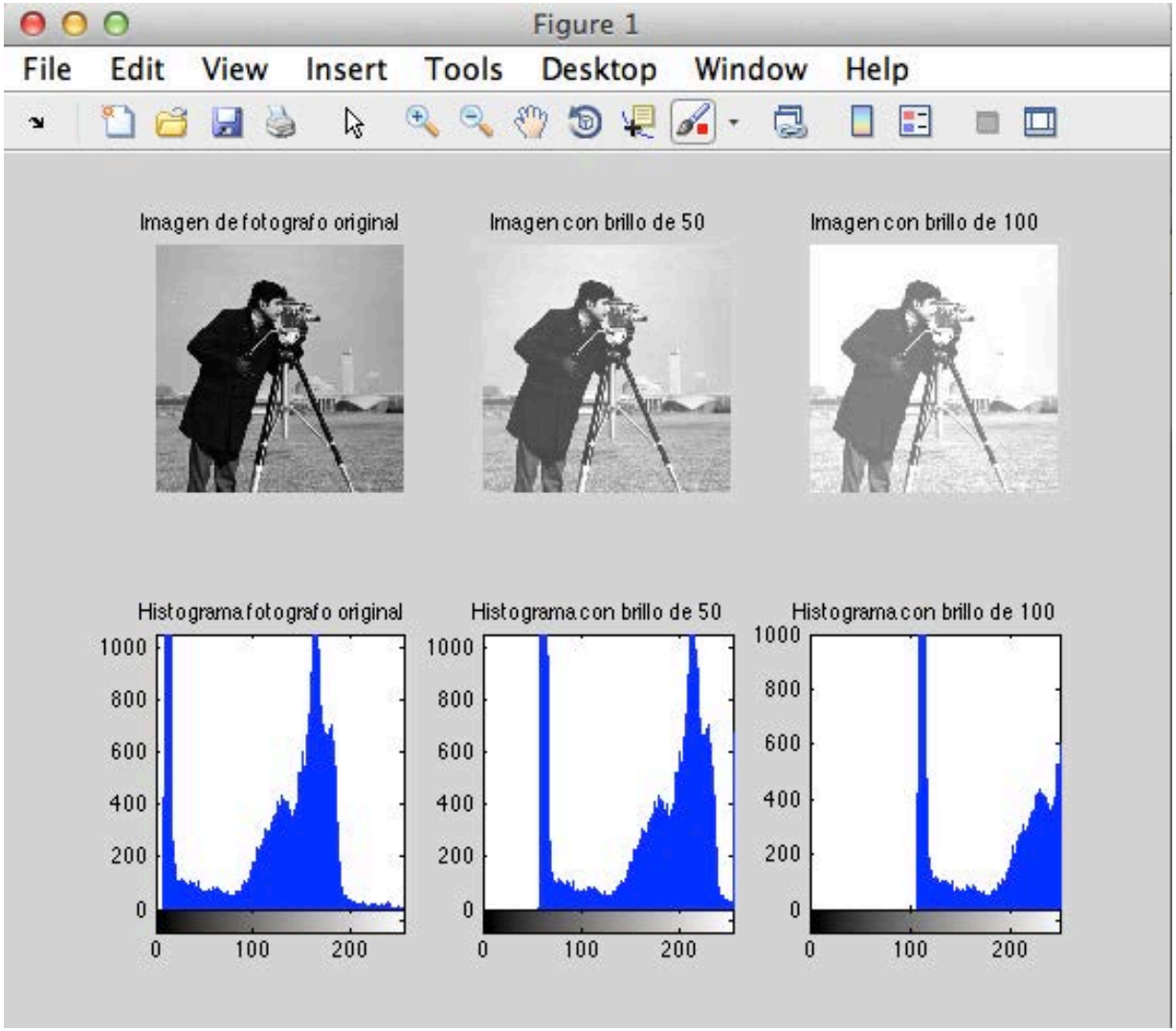
A medida de que la imagen parezca más clara, el histograma se mueve hacia valores más altos de los niveles de grises.

Brillo de una imagen(suma de una constante)

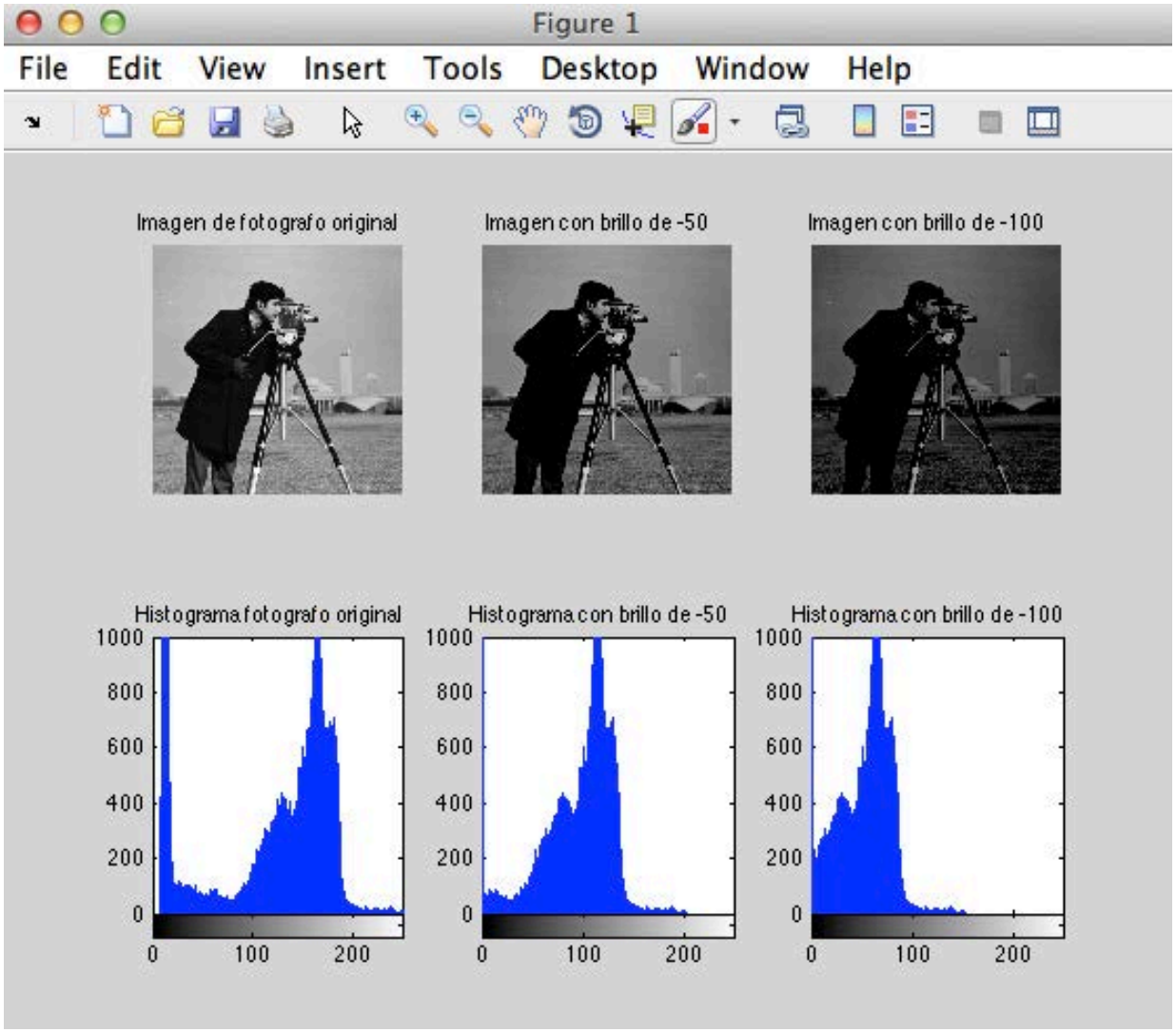



```
img=imread('cameraman.tif');
Imgnum=double(img);
subplot(2,2,1)
Imshow(img)
title('Imagen de fotógrafo original')
subplot(2,2,2)
hist1=imhist(imgnum);
imshow(hist1);
title('Histograma fotógrafo original')
ylabel('Número de Píxeles')
img_con_brillo=imgnum+50;
subplot(2,2,3)
imshow(uint8(img_con_brillo))
title('Imagen con brillo')
subplot(2,2,4)
hist2=imhist(img_con_brillo)
Imshow(hist2);
title('Histograma con brillo')
ylabel('Número de Píxeles‘)
```

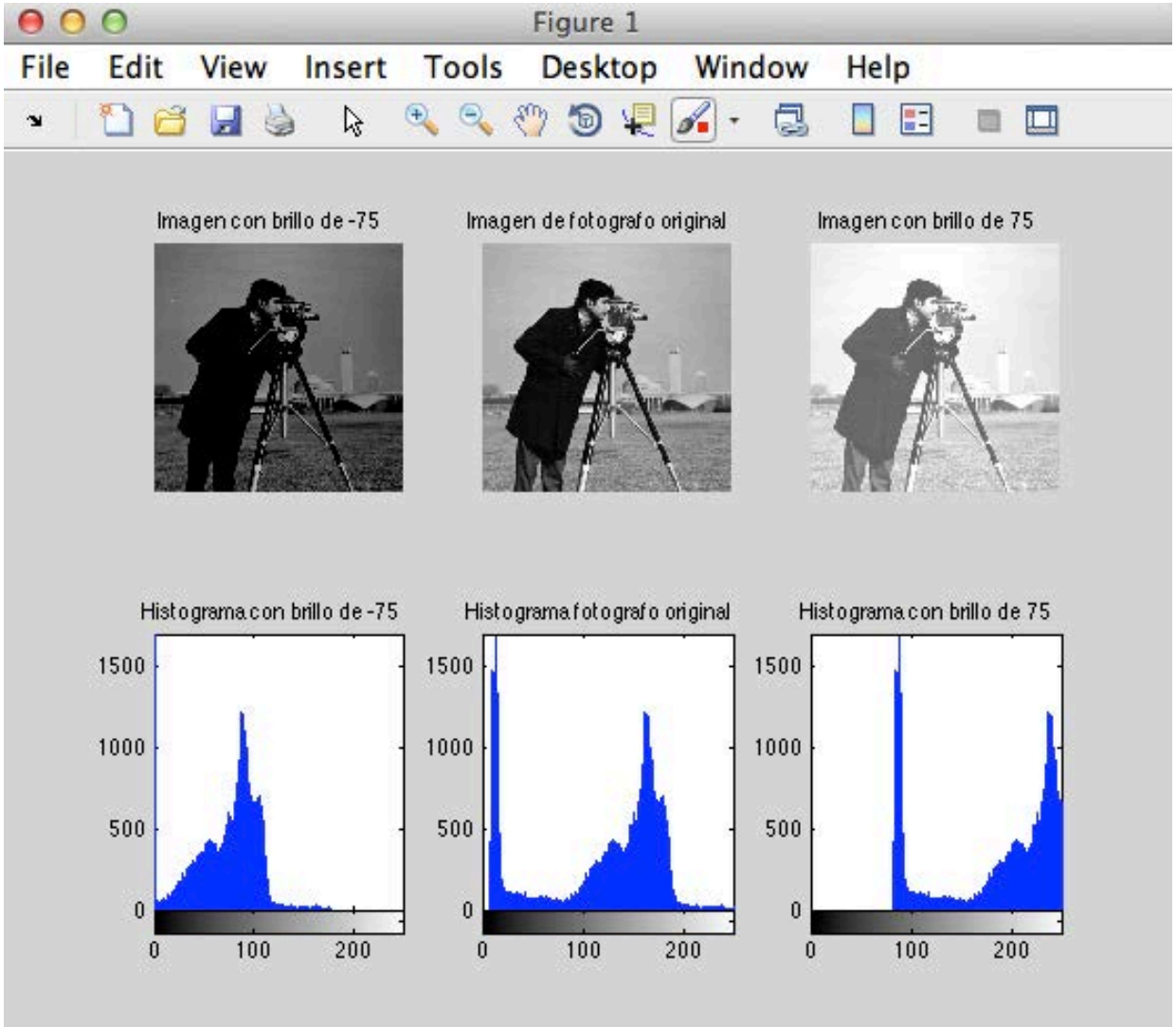
Brillo de una imagen(suma de una constante)



Brillo de una imagen(restas de una constante)



Brillo de una imagen(sumas y restas)



Filtrado de imágenes en el dominio de la frecuencia

Los filtros de frecuencia procesan una imagen sobre el dominio de frecuencia

Teorema de convolución

Transformada de Fourier
Multiplicar por la Función
Re-transformar al dominio espacial

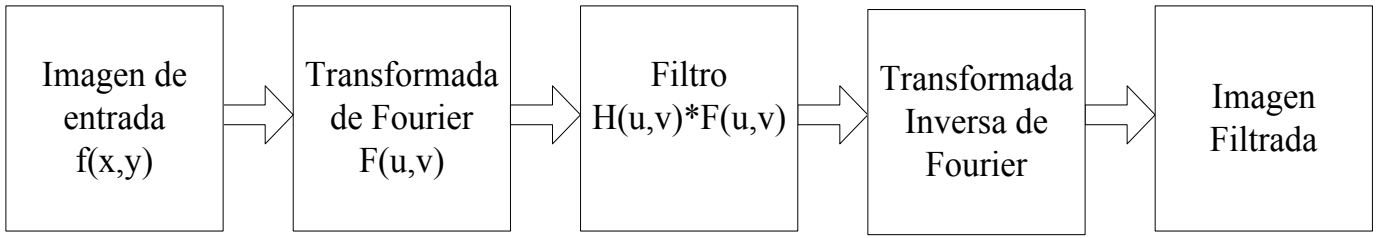


Figura 1. Etapas del procesamiento de imágenes en el dominio de la frecuencia

Filtrado de imágenes en el dominio de la frecuencia

Tipos de filtros

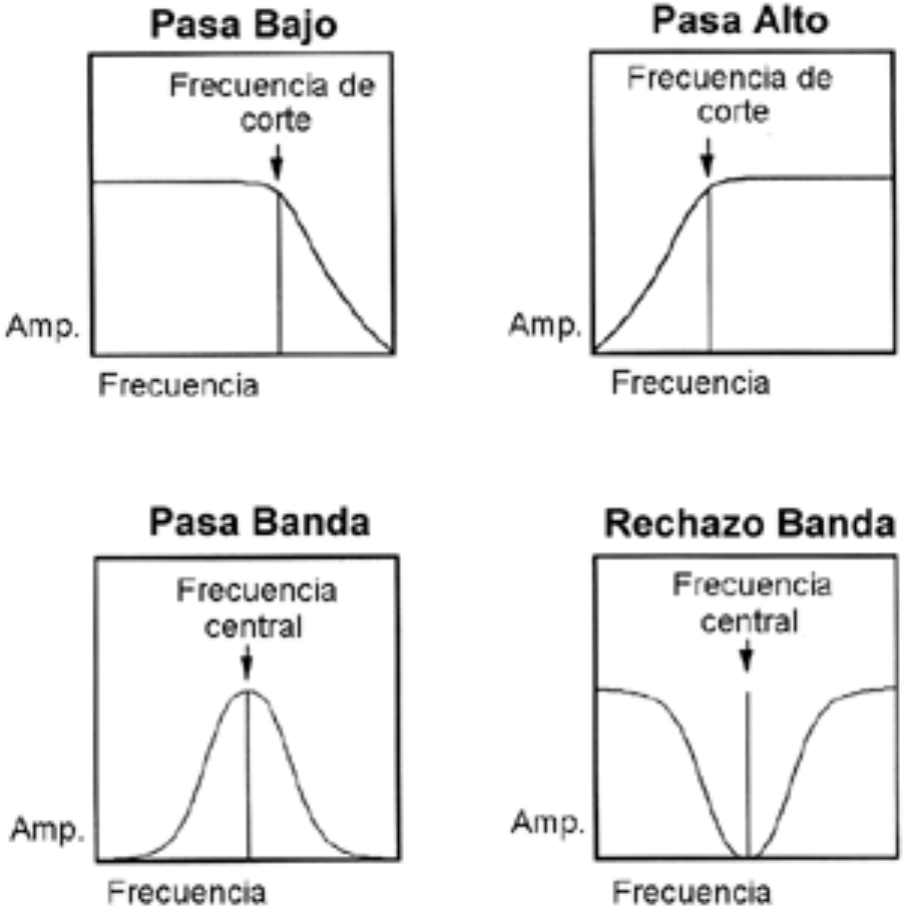
- Caja no-adaptable
  - Pasabajas
  - Pasa altas
  - Raíz cuadrada
  - Cuadrático
  - Sigma global
- Caja adaptable
  - Bit-Error
  - Sigma de caja local

Tipos de Ruido

- Poisson
- Gaussiano
- Sal y Pimienta

Filtrado de imágenes en el dominio de la frecuencia

Respuestas de algunos filtros



Filtrado de imágenes en el dominio de la frecuencia

Ventajas

- Método simple y sencillo de implementar.
- Fácil asociación del concepto de frecuencia con ciertas características de la imagen.
- Proporciona flexibilidad en el diseño de soluciones de filtrado.
- Rapidez en el filtrado al utilizar el Teorema de la Convolución.

Filtrado de imágenes en el dominio de la frecuencia

Desventajas

- Se necesitan conocimientos en varios campos para desarrollar una aplicación para el procesamiento de imágenes.

- El ruido no puede ser eliminado completamente.

Filtrado de imágenes en el dominio de la frecuencia

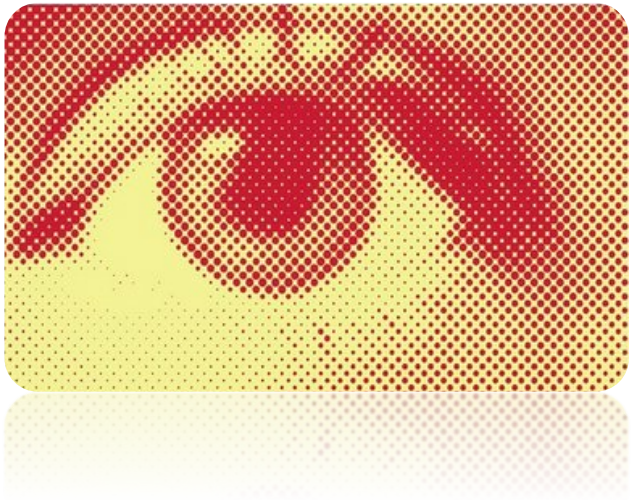
Desventajas

- Se necesitan conocimientos en varios campos para desarrollar una aplicación para el procesamiento de imágenes.

- El ruido no puede ser eliminado completamente.

Filtrado de imágenes en el dominio del espacio

Las operaciones de filtrado se llevan a cabo directamente sobre los pixeles de la imagen.



- Filtros lineales
- Filtros no lineales

Filtrado de imágenes en el dominio del espacio

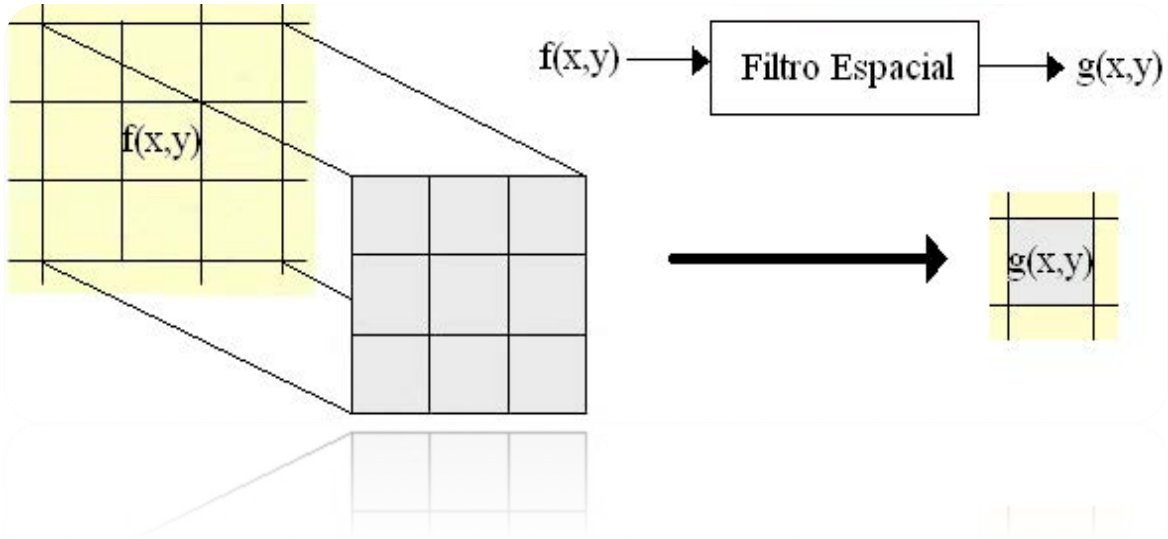
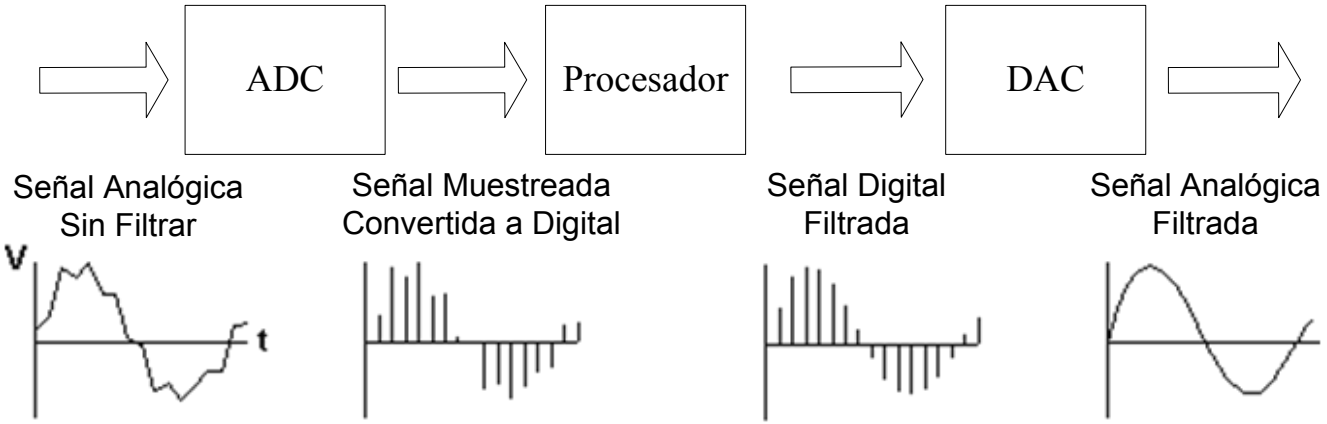


Figura 3. Matriz de coeficientes en el uso del Kernel.

Filtros digitales y analógicos

Filtrado de una señal.



Esquema básico del proceso de filtrado de una señal.

Ventajas de los filtros digitales

✓Un filtro digital es programable

✓Los filtros digitales pueden ser fácilmente diseñados, probados e implementados en un sistema empotrado.

✓Las características de los filtros analógicos, están sujetas a alteraciones y dependen de la temperatura. Los filtros digitales no sufren estos problemas.

Ventajas de los filtros digitales

✓A diferencia de los filtros analógicos, los digitales pueden manejar con mucha precisión las bajas frecuencias.

✓Mayor versatilidad a la hora de manipular la señal.

Ventajas de los filtros digitales

✓La producción en serie es el ser capaz de realizar cientos o miles de filtros idénticos. En el caso de filtros digitales esto se logra sin más que ejecutar el mismo programa en cada procesador. Con filtros analógicos esto no es posible.

Ventajas de los filtros digitales

- Para poder filtrar señales analógicas usando filtros digitales es necesario un interfaz analógico

- Efectos de precisión finita

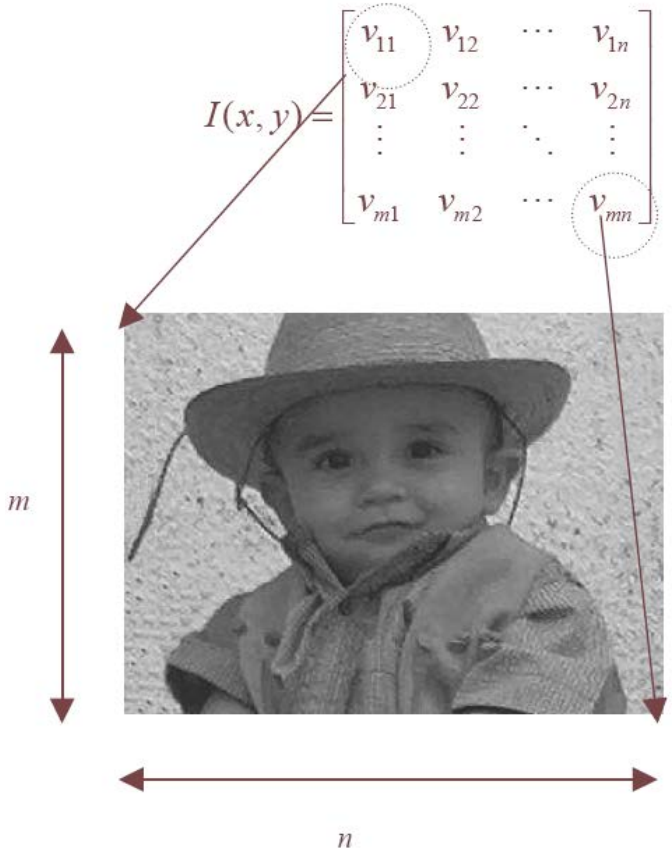
- Tiempo y costo del desarrollo del hardware

Filtros espaciales

El diseño de filtros espaciales se resume en calcular un conjunto de datos que definan una máscara, cuya Transformada de Fourier tenga el comportamiento de uno de los tres filtros básicos

Filtros espaciales

Imagen en escala de grises



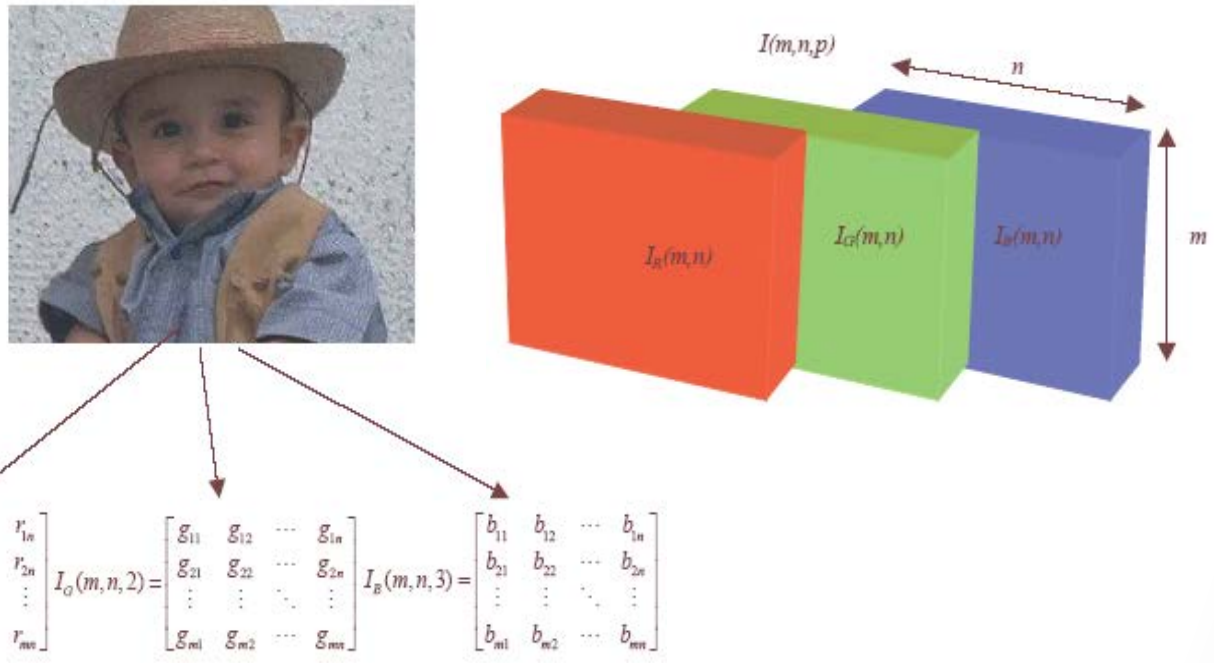
Representación de una imagen a escala de grises

Una imagen es representada por medio de una matriz bidimensional de *m* x *n* elementos en donde *n* representa el numero de píxeles de ancho y *m* el numero de píxeles de largo.

Filtros espaciales

Imagen a color RGB

Es representada por una imagen tridimensional (m x n x p)



Representación de una imagen a color en RGB.

TAREA

- Investigar la matriz de convolución de los siguientes filtros espaciales 3X3 e implementarlos en Octave:

- Filtro pasa bajas
- Filtro pasa altas

Hacer un ejemplo de cada filtro en octave