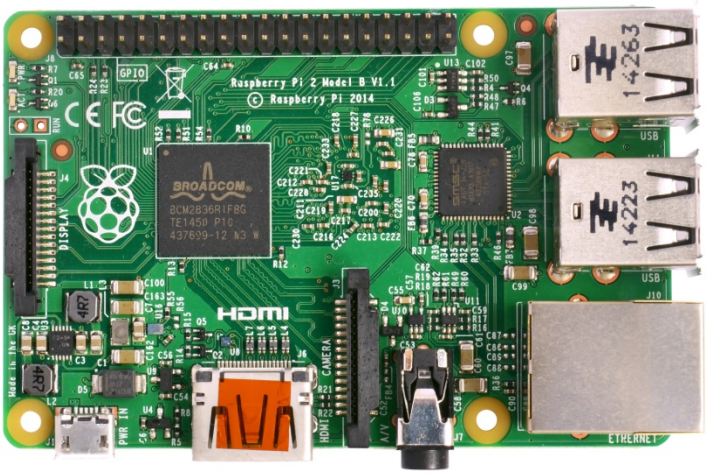
1 Instalación, configuración y puesta a punto del hardware y del software de Raspberry Pi 2

El objetivo de este capítulo es conocer y familiarizarse con el uso y mantenimiento de la Raspberry Pi 2 que usaremos en las siguientes prácticas para controlar el robot iRobot Create.

1.1 Instalación y configuración de la Raspberry Pi 2

Raspberry Pi 2[1] es un mini-ordenador basado en el microcontrolador de Broadcom BMC 2836 a 900Mhz con 1GB de RAM.



Raspberry Pi admite diferentes sistemas operativos. En esta asignatura usaremos el más popular, llamado "Raspbian"[2], una versión de la distribución Linux Debian, adaptada al hardware de la Raspberry Pi.

1.1.1 Verificación de la configuración de Raspberry Pi para el laboratorio de Robótica

En las prácticas de robótica usaremos placas de Raspberry Pi 2 para controlar el iRobot create. Las placas Raspberry Pi 2 están preconfiguradas para este laboratorio. En este apartado verificaremos si lo están y actualizaremos el sistema operativo. Si fuera necesario reconfigurarlas seguiríamos las instrucciones del Anexo 6.

Conectar a la placa Raspberry Pi del grupo:

un teclado y un ratón USB

un adaptador de HDMI-VGA a la pantalla disponible.

un adaptador USB-WIFI

una fuente de alimentación (la Raspberry Pi 2 usa alimentación de 5V a 2A aunque internamente trabaja con 3.3V).

Cada placa Raspberry Pi del laboratorio tiene pegada una etiqueta en la que aparece el nombre que la identifica con su nombre en la red: "ROBOPI**XY**".

Para hacer *login*, el usuario por defecto es "pi" con password "raspberry". Este usuario estará activo, aunque no se recomienda usarlo para las prácticas. Cada grupo tiene un usuario "grupoXY" con password "grupoXY", dónde XY es el número de grupo, XY: {01,02,...10}. Por otro lado, está activo el superusuario "root" con password "toor". No se debe cambiar el password de grupo ni el de root.

---

[1] Raspberry Pi 2: [https://www.raspberrypi.org/products/raspberry-pi-2-model-b/]

[2] Raspbian : [https://www.raspberrypi.org/documentation/raspbian/]

1.1.1.1 Iniciar sesión y comprobar la IP en la WiFi

```
robopiXY login: grupoXY
Password: grupoXY
…
grupoXY@robopyXY ~$
```

Comprobamos la asignación de dirección de red:

```
grupoXY@robopyXY ~$ ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 74:da:38:2e:2a:29
          inet addr:192.168.1.1XY  Bcast:192.168.1.255
    Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:12 overruns:0
    frame:0
          TX packets:8 errors:0 dropped:0 overruns:0
    carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1028 (1.0 KiB)  TX bytes:1070 (1.0 KiB)
```

Es necesario verificar que la IP es la del grupo XY: 192.168.1.1XY

1.1.1.2 Actualizar el sistema operativo

Es recomendable actualizar el sistema operativo a la última versión ejecutando los comandos:

```
~$ sudo apt-get update
~$ sudo apt-get upgrade
```

1.1.1.3 Apagar y reiniciar (halt and reboot)

Si necesitamos reiniciar el sistema usaremos:

```
~$ sudo shutdown –r now
o
~$ sudo reboot
```

Para apagar la Raspberry Pi usaremos la opción halt y después de un tiempo desenchufaremos el conector microUSB (halt):

```
~$ sudo shutdown –h now
o
~$ sudo halt
```

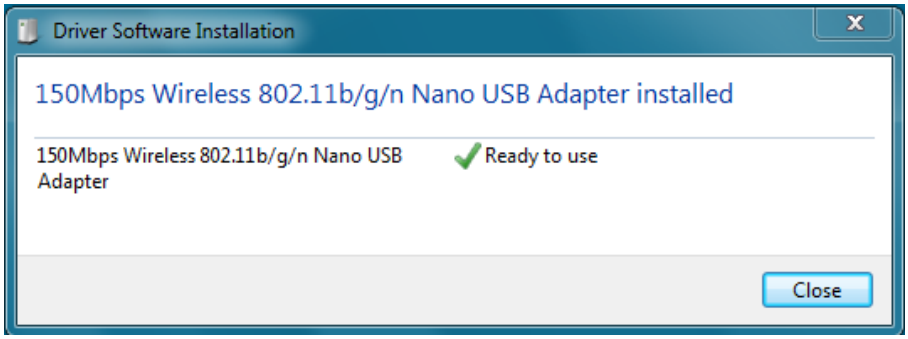
1.1.1.4 Problemas de instalación

Si estas pruebas no dan como resultado que se dispone del usuario “grupoXY” correcto, con el password “grupoXY”, y con la dirección IP correcta (192.168.1.1XY), tendremos que detectar el problema y, en último caso reinstalar el sistema operativo y crear de nuevo el usuario, tal como aparece en el ANEXO: Configuración de la Raspberry Pi para el Laboratorio de Robótica.

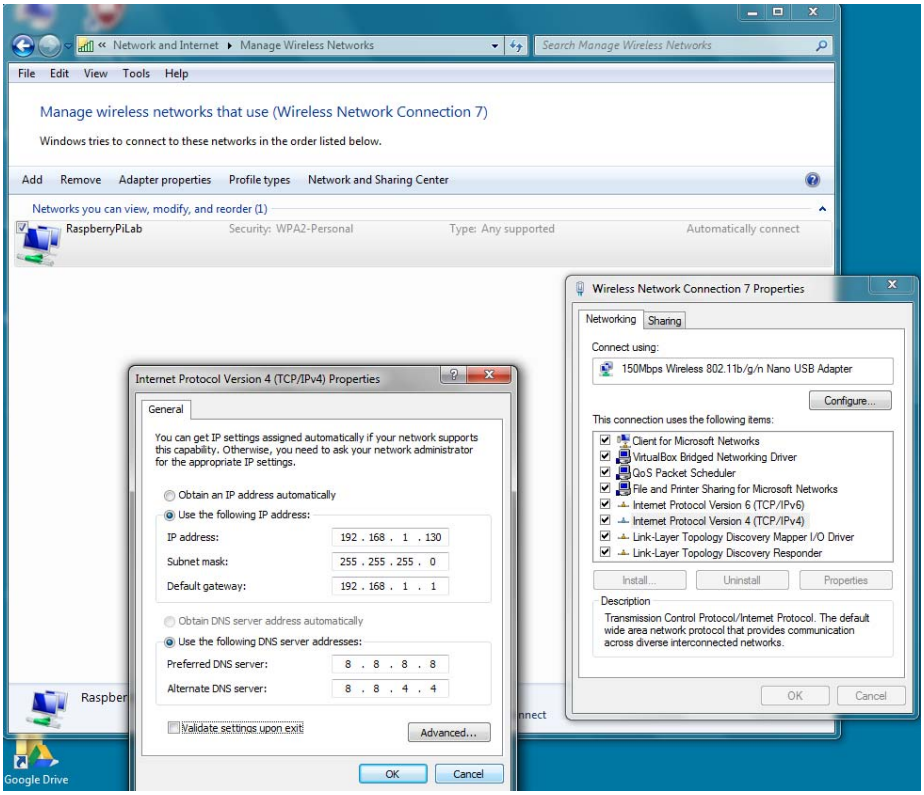
1.2 Conexión de la Raspberry Pi 2 con PC/Windows a través de WiFi

1.2.1 Preparación Windows 7

En el laboratorio se va a utilizar una red wifi llamada “RaspberryPiLab”. Es necesario utilizar un adaptador WiFi USB para conectarse a esa red.



Hay que configurar la dirección IP de los PC en un rango que permita trabajar a las Raspberry PI 2 sin conflictos 192.168.1.131 – 192.168.1.140.



Asignación de números de IP a cada grupo		
Grupo	Raspberry Pi	Puesto de trabajo
1	192.168.1.101	192.168.1.131

2	192.168.1.102	192.168.1.132
3	192.168.1.103	192.168.1.133
4	192.168.1.104	192.168.1.134
5	192.168.1.105	192.168.1.135
6	192.168.1.106	192.168.1.136
7	192.168.1.107	192.168.1.137
8	192.168.1.108	192.168.1.138
9	192.168.1.109	192.168.1.139
10	192.168.1.110	192.168.1.140

1.2.2 Conexión remota mediante SSH

Para usar la Raspberry PI 2 desde un sistema de escritorio como Windows necesitamos usar un cliente ssh para conectarnos de manera remota.

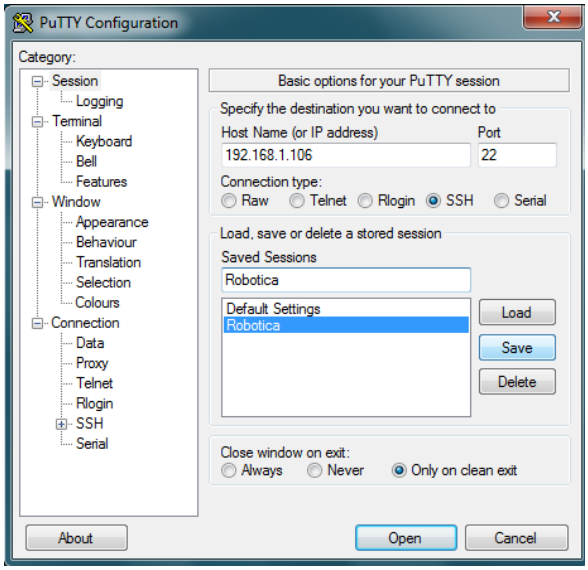
PuTTYes un software ligero que permite configurar de forma sencilla una conexión remota mediante el protocolo SSH (Secure Shell) indicando la dirección IP de nuestra Raspberry PI 2 y usando el puerto 22.

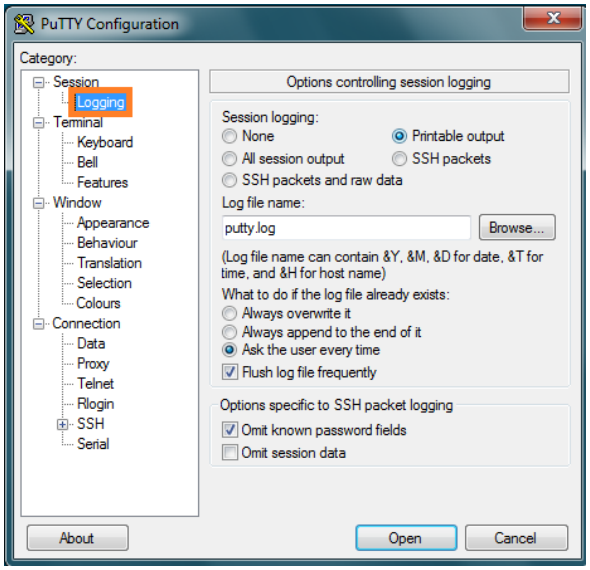
PuTTY[3] permite almacenar un Log con todos los comandos y respuestas recibidas de la sesión remota si se configura en:

```
Session -> Logging
```

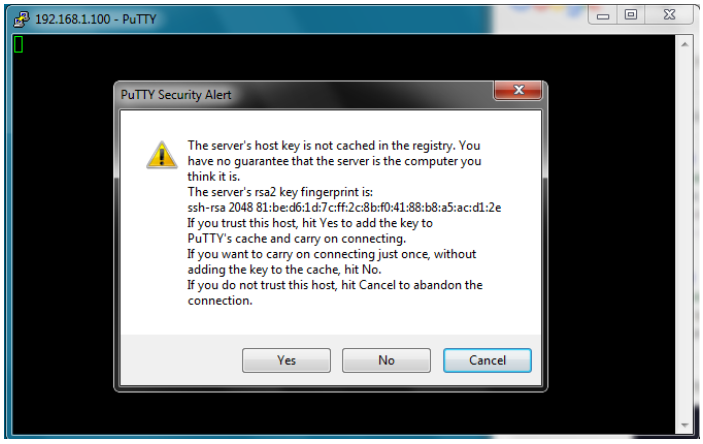
También permite asignar un nombre distinto a cada fichero de log creado en la opción "Log file name:". Por ejemplo: "&M&D&T_GrupoXY.log" guarda la fecha y el número del grupo como nombre de fichero.

[3] Putty. http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

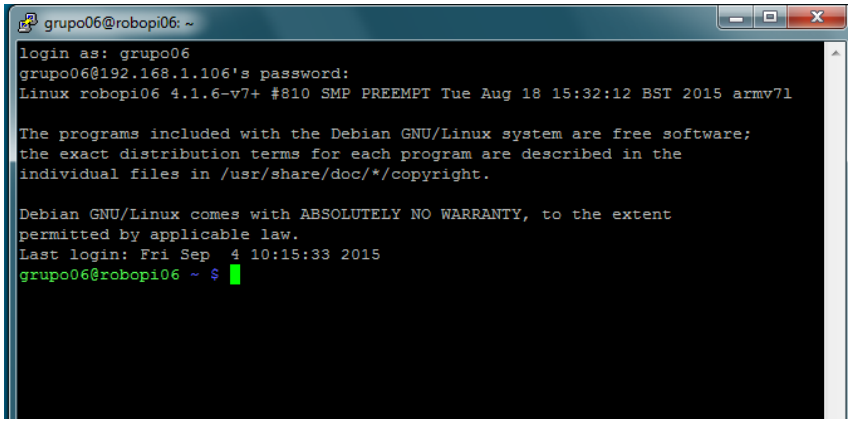




Cada vez que se conecta de forma remota un nuevo dispositivo, un mensaje de alerta notifica que el host es desconocido para el cliente.



Tras introducir el usuario y la contraseña podemos empezar a manejar la Raspberry Pi 2 desde la línea de comandos.



1.2.3 Programación en C en la Raspberry Pi

Raspbian incluye los compiladores gcc y g++ para compilar código en c/c++, editores, linkeditor, etc. Así pues, se pueden escribir programas en C/C++ y compilarlos sin necesidad de equipos exteriores. Para ello, pueden ser útiles los siguientes comandos:

Comando Raspbian	Función
`touch programa.c`	Crea un archivo
`nano programa.c`	Edita el archivo (Ctrl+O guardar, Ctrl+X Salir)
`gcc -Wall programa.c –o programa`	Compila en C el archivo programa.c
`g++ -Wall programa.c –o programa`	Compila en C++ el archivo programa.c
`./programa`	Ejecuta el programa

1.2.4 Ejercicio 1: "Hola mundo" en C

Para probarlos, escribe un programa "Hola Mundo" usando el lenguaje de programación C.

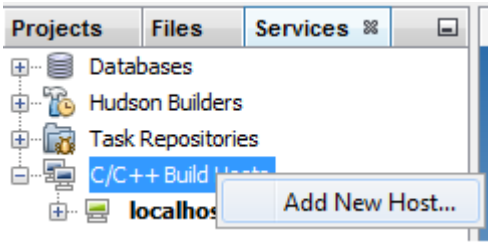
```
#include <stdio.h>
int main() {
    printf("Hola mundo");
    return 0;
}
```

Escribe otro programa para comprobar el funcionamiento de la función "scanf" para leer el teclado y la definición:

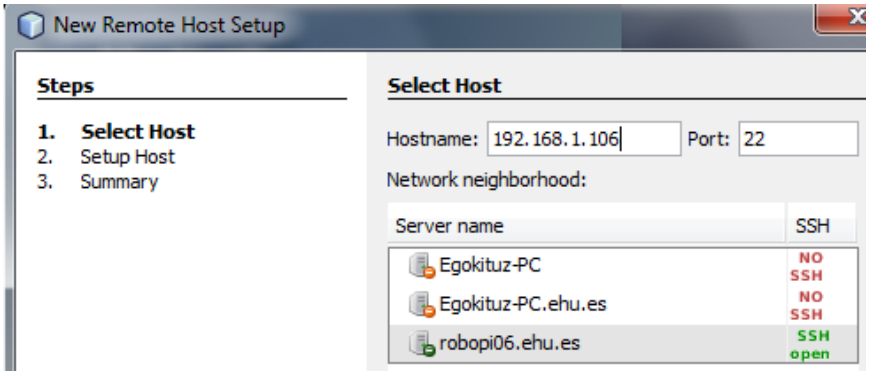
```
int main(int argc, char *argv[])
```

1.3 Programación Raspberry Pi 2 usando el entorno NetBeans

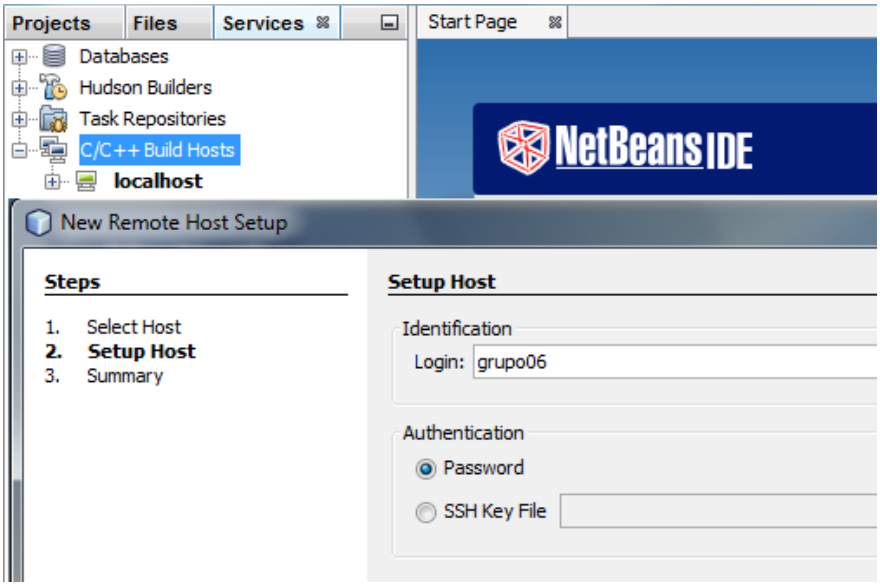
Para desarrollar programas en la asignatura recomendamos utilizar el entorno NetBeans[4] en Windows 7. Netbeans permite fijar un host remoto para compilar los programas[5].



Hay que escribir la dirección IP de la Raspberry Pi



El login es el usuario que tenemos asignado

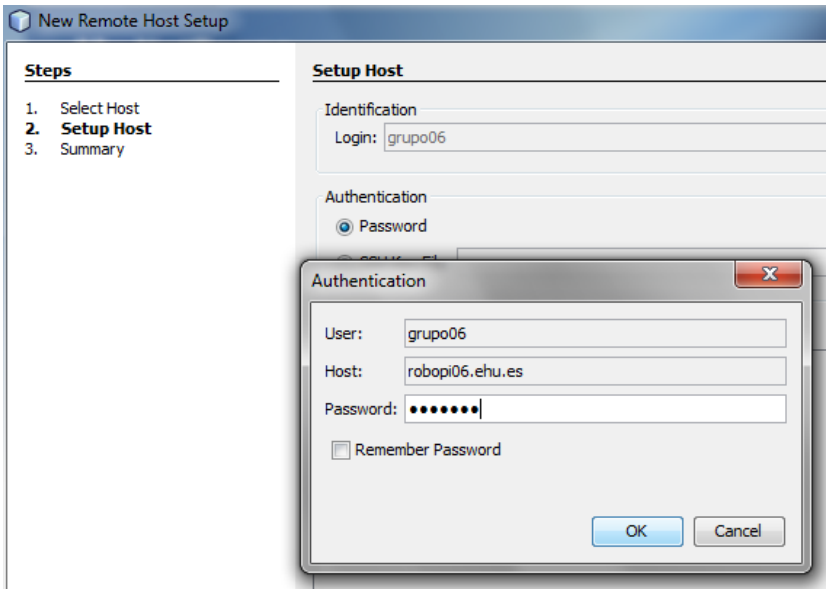


Hay que introducir el password asociado a la cuenta

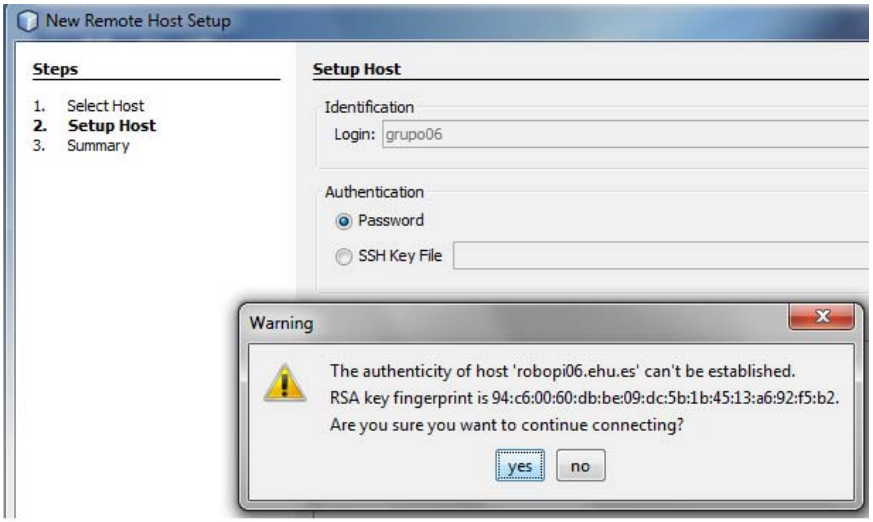
---

[4] Descarga NetBeans. https://netbeans.org/downloads/

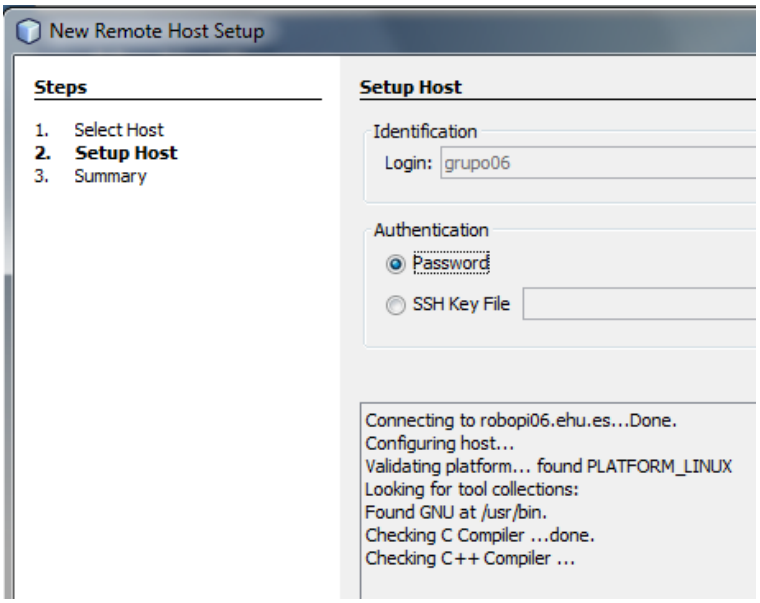
[5] NetBeans IDE Tutorial. https://netbeans.org/kb/docs/cnd/remotedev-tutorial.html



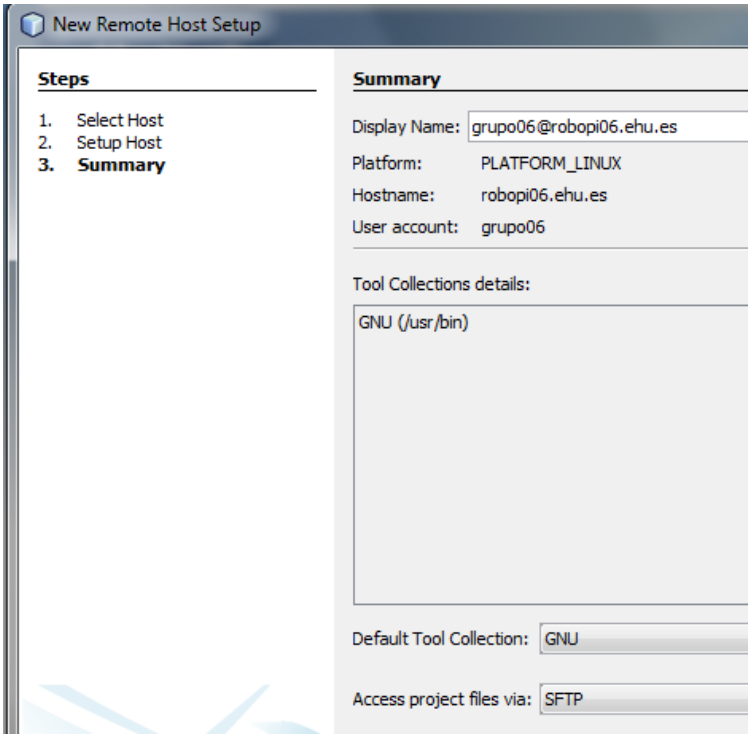
Siempre que se conecte por SSH a un host remoto aparece una advertencia de seguridad que hay que aceptar



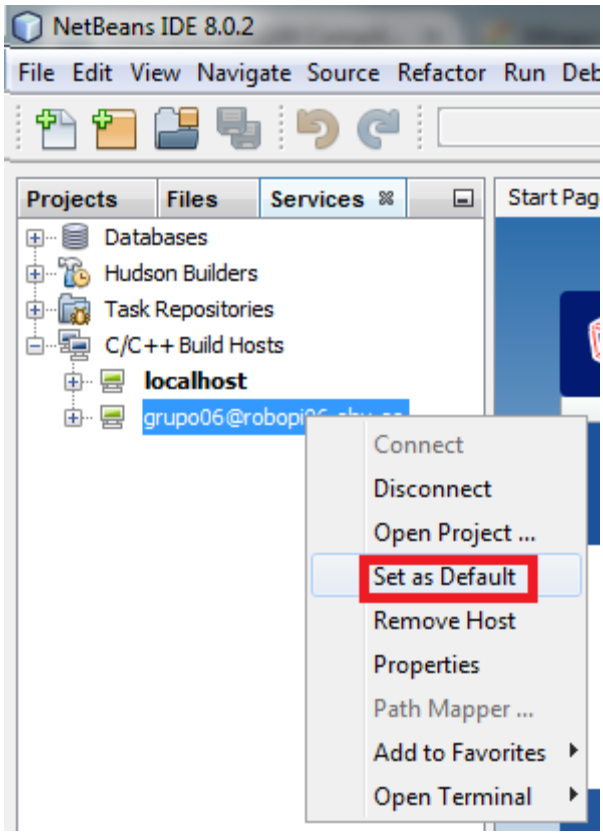
NetBeans detecta y configura el entorno de programación automáticamente para los compiladores que se encuentren instalados en la Raspberry PI



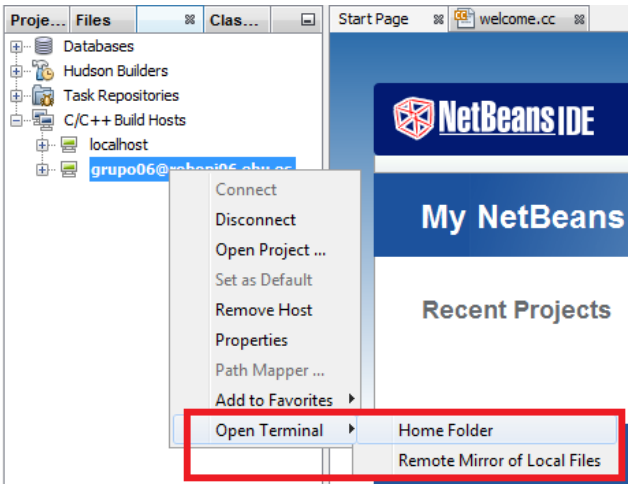
Hay que asegurarse de que use el protocolo SFTP para copiar archivos o puede haber problemas a la hora de crear un nuevo proyecto



Es recomendable configurar el nuevo "Host" por defecto para que los proyectos compilen directamente sobre la Raspberry Pi

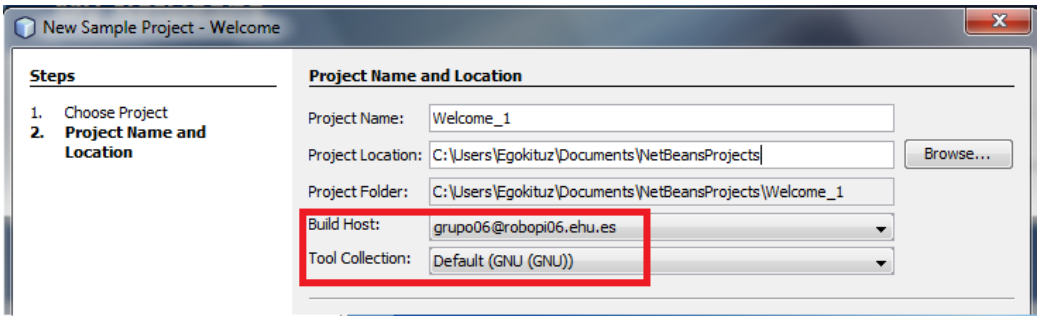


Es posible abrir un cliente SSH remoto usando las opciones “Open Terminal” de NetBeans.

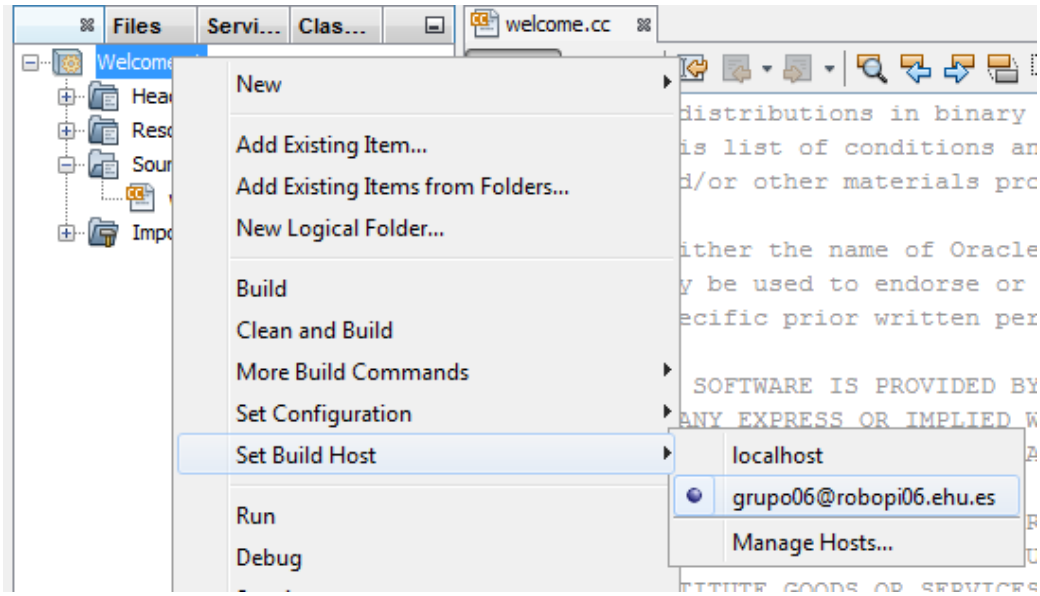


1.3.1 Comprobación

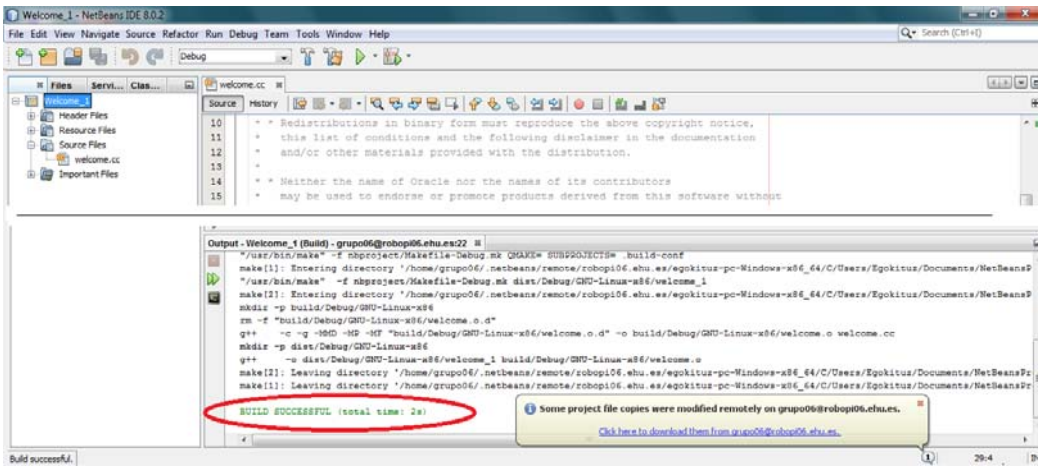
Probar que los programas de ejemplo incluidos en NetBeans se ejecutan correctamente



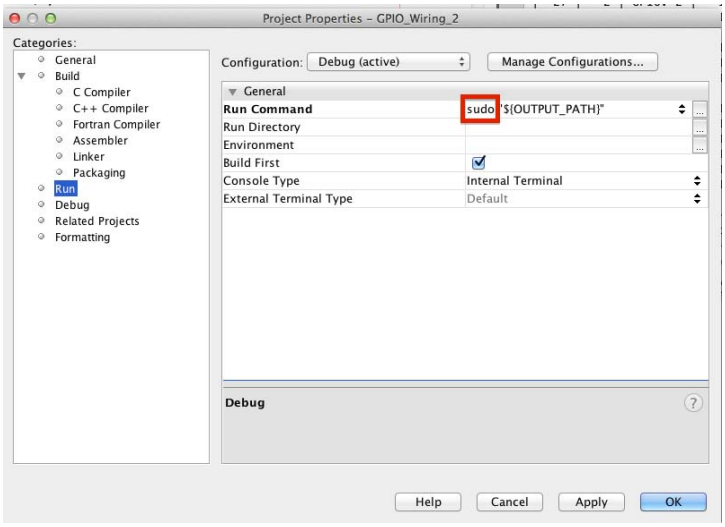
Hay que asegurarse que el "Host" es el adecuado antes de compilar



Si el compilador finaliza sin errores, aparecerá un mensaje en color verde



Para lanzar un programa como root desde NetBeans, hay que configurar la opción Run en las propiedades del proyecto



NetBeans deja el código ejecutable resultante de la compilación y linkedición en un directorio de la Raspberry Pi. Normalmente deberíamos ir a ese directorio y ejecutar con la orden

```
~$ ./programa
```

Para poder ejecutar el programa usando el "Run Projetc" de .NetBeans (lo que resulta mucho más cómodo que tener que ir al directorio que contiene el ejecutable):

En una sesión de Raspberry Pi hacer lo siguiente:

1. Pasar a ser root:

```
~$ sudo su
```

2. Poner password al root (todos usamos "toor" para que no haya problemas de olvidos):

```
~$ passwd root <R>
toor
```

En la ventana de NetBeans:

1. Services, C/C++ Built Host borrar el antiguo host y crear uno nuevo:

- Add new host:
- Host name: dirección IP
- Setup Host -> Name: **root**, Password: **toor**

Otra alternativa es establecer el host por defecto con el usuario root de nuestra Raspberry Pi

Ahora se puede compilar y ejecutar directamente con el triangulo verde "Run Project"

1.3.2 Prueba de generación de código en NetBeans-para Raspberry Pi

Para crear nuevos proyectos, lo más seguro es cargar y modificar uno de los ejemplos (tal como Wecome).

```
> File: New Project: Samples: Welcome
```

En algunos casos, para desarrollar las prácticas se os pasarán proyectos que contienen algunas de las funciones y estructuras de datos que hay que utilizar. Para completar un proyecto existente, lo mejor es usar “Import Project” en el menú File. De otra manera puede haber problemas para que encuentre los ficheros

Una vez instalado probar que los programas de ejemplo incluidos en NetBeans se ejecutan correctamente. Por ejemplo, el "Project welcome":

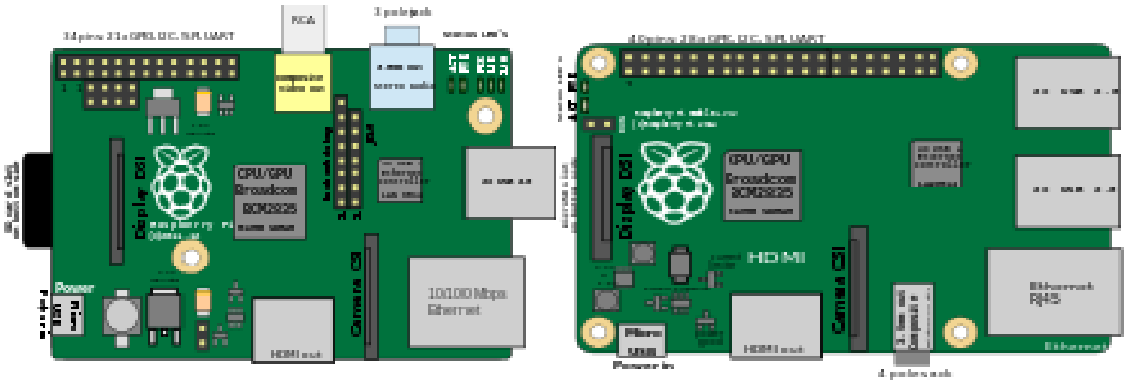
```
> File: New Project: Samples: Welcome
```

1.4 GPIO: Entradas/salidas de la Raspberry Pi 2

El puerto GPIO (General Purpose Input/Output) de la Raspberry Pi 2 tiene 40 pines con diferentes funciones configurables, que se pueden programar para interactuar con el exterior. De los 40 pines, 26 son GPIO y los otros son para alimentación o tierra (además de dos pines ID EEPROM que no conviene usar). Las entradas se pueden conectar a sensores o a otros dispositivos. Las salidas permiten desde encender un LED hasta enviar señales o datos a otros ordenadores. Por ejemplo, este puerto permite el uso de los periféricos del BMC2836 para comunicaciones línea serie, protocolos i2c, SPI, etc.

Características generales

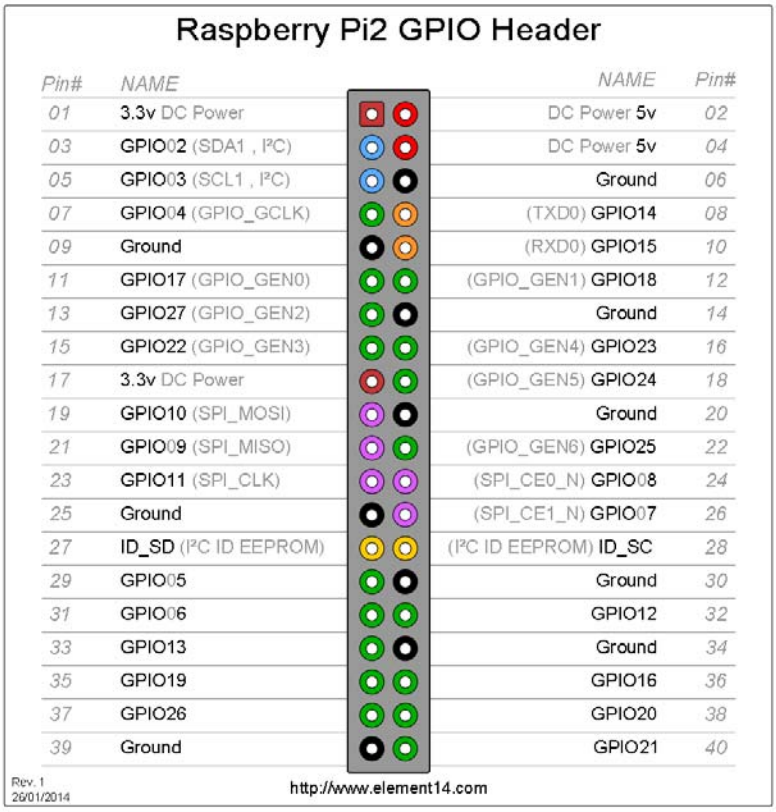
- Versión 1 (02/2012)
- 700 Mhz Single Core, ARM1176JZF-S, 256/512 MB RAM
- Versión 2 (F02/2015)
- 900 Mhz Quad Core , ARM Cortex-A7, 1 GB RAM
- Basado en el SOC BCM2835/BCM2836
- Entrada / Salida configurable: 26x GPIO, USART, I2C



http://www.jameco.com/Jameco/workshop/circuitnotes/raspberry_pi_circuit_note.html

1.4.1 Periféricos BCM2836

Para conectar dispositivos exteriores, usamos las asignaciones del conector de 40 pines de la Raspberri Pi. No obstante, es muy posible que el software que utilicemos para controlar las estradas salidas, asigne nombres diferentes a esos mismos pines.



A la hora de conectar componentes externos al puerto GPIO conviene tener muy en cuenta las siguientes **precauciones** para no dañar la Raspberry PI 2:

No introducir más de 3.3V en un pin GPIO
No consumir más de 3mA por los pines de salida, y asegurarse de protegerlos usando resistencias
No presionar o tocar el puerto GPIO con destornilladores o objetos metálicos
No alimentar la Raspberry Pi con más de 5V
No consumir más de 50mA de los pines a 3.3V
No consumir más de 250mA de los pines a 5V

Para más información se puede acceder a los siguientes enlaces:

Web oficial: https://www.raspberrypi.org/
Resources: https://www.raspberrypi.org/resources/
Forum: https://www.raspberrypi.org/forums/
Libros Springer: http://link.springer.com/search?query=Raspberry+Pi&facet-content-type=%22Book%22&from=SL

1.4.2 Acceso al puerto GPIO de la Raspberry Pi 2 desde Raspbian

Para programar la función de los pines de entrada salida hay que manipular los ficheros que se encuentran en la carpeta */sys/class/gpio* lo que **requiere permisos de root**.

Para saber si tiene permisos de acceso al grupo GPIO ejecutar el comando:

```
~$ groups grupoXY
```

Si responde

```
grupoXY: grupoXY audio video gpio
```

es que tiene permiso para usar GPIO. Si no fuera así, hay que añadir el usuario grupoXY al grupo GPIO:

```
~$ sudo usermod -aG gpio grupoXY
```

Ahora, repetir la verificación con el comando groups.

Para activar un pin en concreto hay que crear un acceso a él usando "export"

```
~$ echo 4 > /sys/class/gpio/export
```

A continuación hay que configurarlo para que se comporte como pin de entrada o salida (in/out)

```
~$ echo out > /sys/class/gpio/gpio4/direction
```

Finalmente se puede escribir 1 o 0 para asignarle un voltaje "High" o "Low" en caso de que esté configurado como salida en el paso anterior

```
~$ echo 1 > /sys/class/gpio/gpio4/value
```

O se puede leer el valor del voltaje en el pin usando el comando "cat"

```
~$ cat /sys/class/gpio/gpio4/value
```

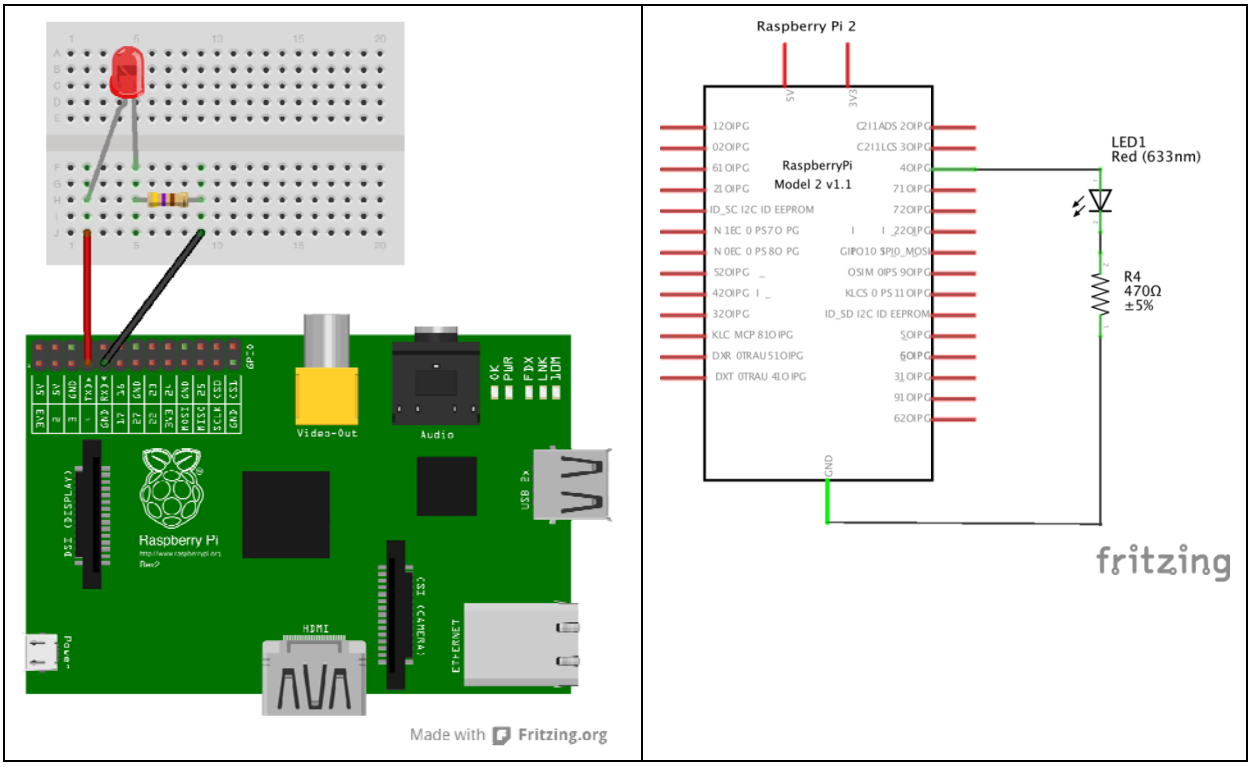
1.4.3 Prueba de uso de la GPIO

Lista de Materiales

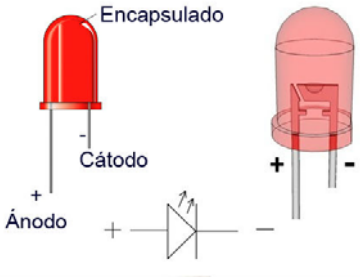
- Raspberry PI 2 (etiquetada)
- Caja Transparente (etiquetada)
- Fuente de alimentación 5V 2A
- Micro SD 8 GB con Raspbian precargado (etiquetada)
- 2 Adaptadores USB WiFi con etiquetas negra y verde
- Placa de montaje
- Cables de conexión (se suele usar rojo para la señal y negro para tierra/GND)
- 1 LED
- 1 pulsador
- dos resistencias de 470 Ω y una de 1KΩ
- Tira de 9 pines

1.4.3.1 Ejercicio 1:

Conectar a un GPIO un LED siguiendo el siguiente esquema:



Cuidado: Verificar que el ánodo (+, pata larga) del LED está conectado al cuarto pin de la fila interior (correspondiente al GPIO4) y el cátodo (- pata corta) a una resistencia de 470Ω al quinto pin de la misma fila (que es tierra/GND).



Encender y apagar el LED desde el terminal usando los comandos

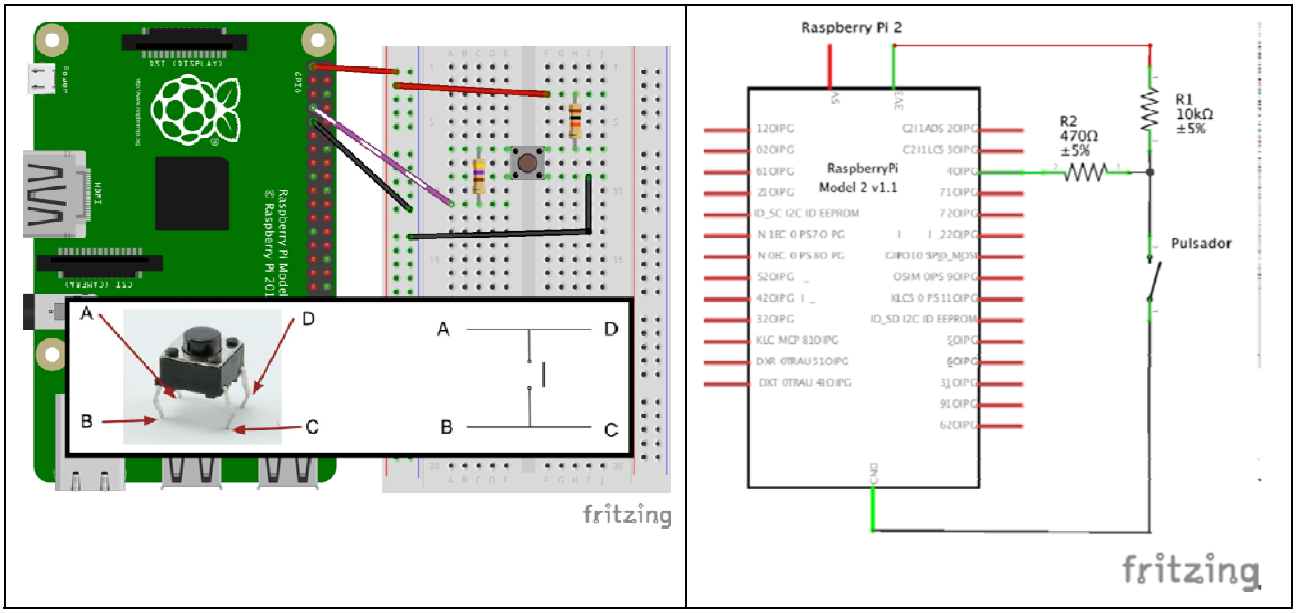
```
~$ echo 1 > /sys/class/gpio/gpio4/value
~$ echo 0 > /sys/class/gpio/gpio4/value
```

1.4.3.2 Ejercicio 2

Escribir un programa en C para controlar el encendido y apagado del LED usando la entrada estándar (teclado). Para ello se recomienda usar la librería "stdlib.h" y la función system que permite ejecutar comandos del Shell.

```
#include <stdlib.h>
..
int system("comando");
```

1.4.3.3 Ejercicio 3



Conectar el pulsador y verificar que una de sus patas (A) va al quinto pin de la fila interior (que es tierra/GND) y que de la otra pata ((B) va a al primer pin (3,3V) a través de una resistencia de 10KΩ y desde la misma pata B va, al tercer pin de la misma fila (correspondiente al GPIO3) a través de una resistencia de 470 Ω.

Programar el PIO4 con los comados

```
~$ echo 3 > /sys/class/gpio/export
~$ echo in > /sys/class/gpio/gpio3/direction
```

y verificar el funcionamiento del pulsador con el comando:

```
~$ cat /sys/class/gpio/gpio3value
```

1.4.3.4 Ejercicio 4

¿Puedes hacer un programa en C que obtenga el valor del pulsador con una llamada

```
system (cat /sys/class/gpio/gpio3value);?
```

En ese caso, modifica el programa del apartado 1.4.3.2 para que el LED empiece a parpadear a intervalos regulares de 0,5s cuando se pulse el pulsador y se apague completamente al pulsar de nuevo el pulsador.

1.5 Librería Wiring Pi

Para usar los pines del puerto J8 de la Raspberry Pi 2 vamos a usar la librería Wiring Pi[6] preparada para C/C++.

WiringPi es una librería de acceso a los pines de la GPIO escrita en C para el BCM2835 utilizado en la Raspberry Pi. Se puede utilizar desde C, C ++ y otros lenguajes.

1.5.1 Instalación de Wiring Pi

Instalar GIT

```
~$ sudo apt-get install git-core
```

Update & Upgrade Raspberry Pi

```
~$ sudo apt-get update
~$ sudo apt-get upgrade
```

- **Descargar el proyecto WiringPi**

Si ya ha sido descargado previamente devolverá el mensaje: "F*atal destination path 'wiringPi'" already exists and is not an empty directory*". Si es así se puede pasar al siguiente punto.

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

Entrar en la carpeta y actualizar el contenido

```
~$ cd wiringPi
~$ git pull origin
```

Compilar las librerías

```
~$ ./build
```

Comprobar que la librería funciona

```
~$ gpio -v
~$ gpio readal
```

Que devuelve la siguiente tabla, con la programación actual de los pines, donde[7]:

- BCM es el código asignado por la placa BCM GPIO
- wPi: código asignado por Wiring Pi
- Name: función del pin
- Mode: cómo está programado actualmente (IN/OUT)
- V: su estado lógico (1: high/true/, 0, low/false)
- Physical: posición en le conector (impares a la izquierda y pares a la derecha)

---

[6] Wiring Pi. http://wiringpi.com/

[7] Pins: http://wiringpi.com/pins/

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	IN	1	7 || 8	1	ALT0	TxD	15	14
		0v			9 || 10	1	ALT0	RxD	16	15
17	0	GPIO. 0	IN	0	11 || 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	1	13 || 14			0v		
22	3	GPIO. 3	IN	1	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

Para usar Wiring Pi es necesario llamara a los pines según la siguiente tabla. Hay que tener en cuenta que los nombres lógicos que asigna Wiring Pi no corresponden con los nombres físicos.

Por otro lado, Wiring PI lleva en funcionamiento desde las primeras versiones de Raspberry PI. El orden y disposición del GPIO ha ido cambiando para las distintas revisiones de la placa.

1.5.2 Uso de las librerías

WiringPi está compuesta por una serie de funciones que facilitan el uso de la entrada/salida al programar en C/C++ la Raspberry PI.

Es necesario en primer lugar incluir referencias a los archivos .h que definen las funciones que vamos a usar:

```
#include <wiringPi.h>
#include <softPwm.h> (para usar PWM)
```

En esta práctica haremos uso de las siguientes funciones[8]:

[8] El resto de funciones de la librería se pueden encontrar consultar en: http://wiringpi.com/reference/

Entrada/salida	
Función	Funcionamiento
wiringPiSetup ()	Configura el puerto GPIO para usar la numeración por defecto de la librería WiringPI
pinMode (Param1, Param2)	Selecciona si un pin funciona com entrada o salida: Param1: Identificador del pin; Param2: INPUT / OUTPUT
digitalWrite (Param1, Param2)	Establece el valor de un pin configurado de salida: Param1: Identificador del pin; Param2: HIGH / LOW
int digitalRead(Param)	Adquiere el valor de un pin configurado de entrada: Param: identificador del pin; Return: 1 - High / 0 - Low
delay(Param);	Pausa la ejecución de un programa. Param: Número de milisegundos
softPwmCreate(Param1, Param2, Param3);	Configura un pin de salida para el uso de la modulación de anchura de pulso (PWM): Param1: Identificador del pin; Param2: Valor inicial del pin; Param3: Rango máximo
softPwmWrite(Param1, Param2)	Establece el valor del PWM de un pin de salida: Param1: Identificador del pin; Param2: Valor del PWM respecto al rango

1.5.3 Compilación de programas con WiringPi

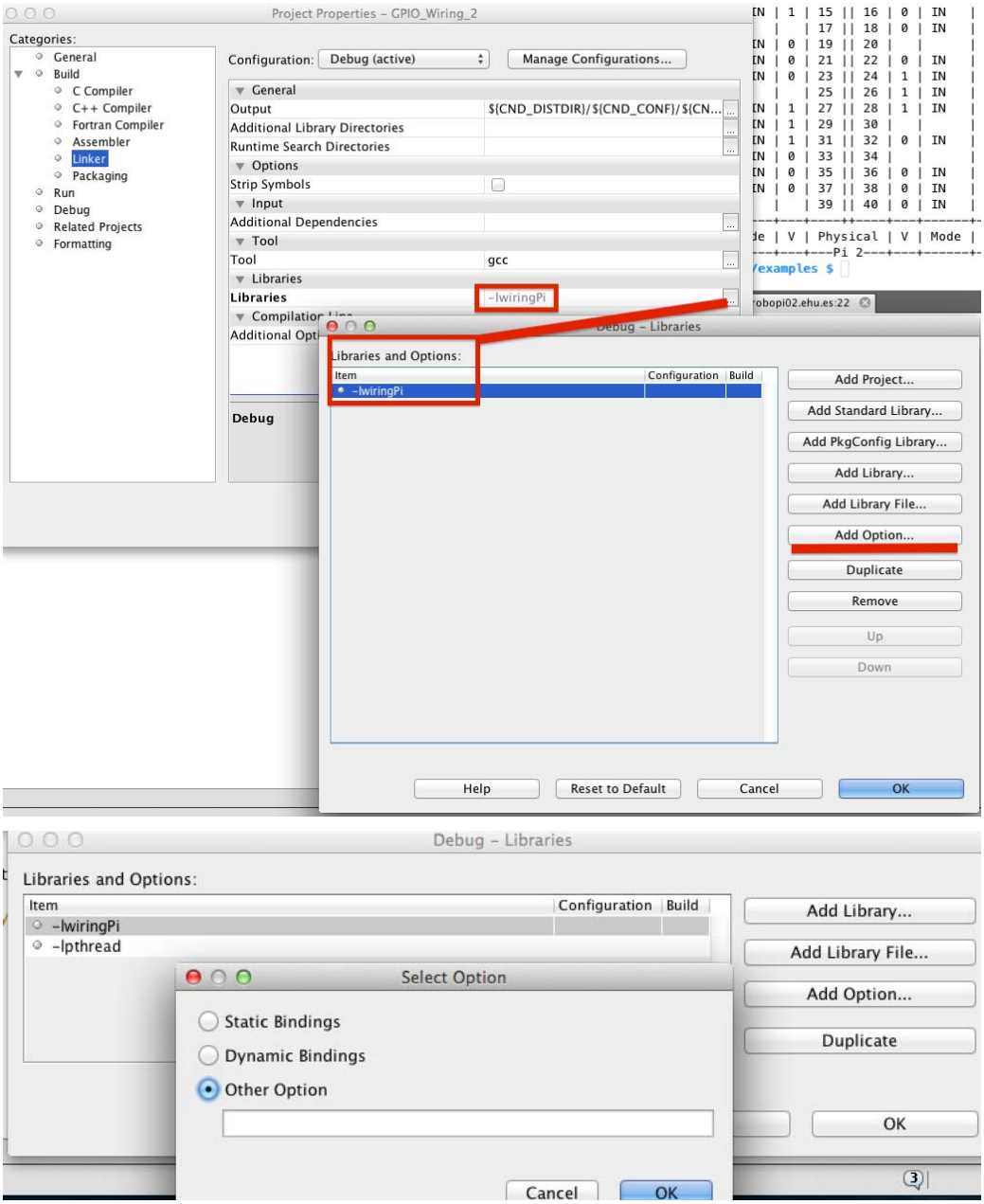
Los proyectos que usan WiringPi requieren que se haga referencia a las librerías estáticas que se instalan. Para ello hay que añadir "-lwiringPi" a las opciones de compilación:

```
~$ gcc -Wall myFile.c -o programa -lwiringPi
```

Si se usa la funcionalidad de PWM también hay que incluir la opción "-lpthread". Para ejecutar el programa hay que hacerlo como administrador:

```
~$ sudo ./programa
```

En NetBeans es necesario añadir las librerías de forma manual en las "propiedades del proyecto", apartado "Linker":



Escribir en la ventana Other Option

```
-lwiringPi
```

1.5.3.1 *Ejercicio 1: control de un LED mediante Wiring Pi*

Escribir un programa que encienda un LED conectado al pin físico 7 (pin lógico Wiring Pi: 7) y lea cíclicamente un pulsador conectado al pin físico 5 (pin lógico Wiring Pi: 9). Cuando se pulsa el pulsador se apaga el LED y el programa termina.

1.5.4 Modulación por anchura de pulso

La **modulación por anchura de pulso** (**PWM**, *pulse-width modulation*) de una señal es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$D = \tau / T$

- *D*: ciclo de trabajo
- τ: tiempo en que la función es positiva (ancho del pulso)
- *T*: período de la función

La principal desventaja que presentan los circuitos PWM es la posibilidad de que haya interferencias generadas por radiofrecuencia. Éstas pueden minimizarse ubicando el controlador cerca de la carga y realizando un filtrado de la fuente de alimentación.

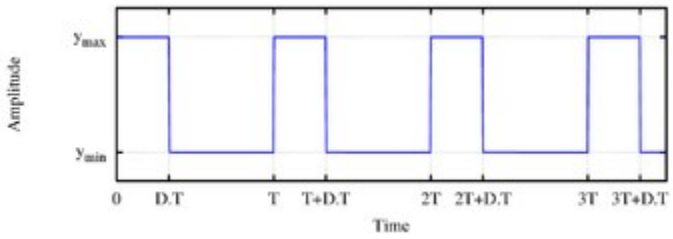


Fig.: una señal de onda cuadrada de amplitud acotada mostrando el ciclo de trabajo D.

1.5.4.1 PMW en Raspberry Pi2 con GPIO

```
#define LEDAMARILLO X //numero de salida GPIO a la que
    aplicamos el PWM
/*  Inicilización PWM
*   @softPwmCreate: función de activación del PWM en un pin
*   Param 1: número de pin GPIO de salida
*   Param 2: Límite inferior (mínimo permitido 0)
*   Param 3: Límite superior (máximo permitido 100)
*   Return: 0 si falla.
*/

if(softPwmCreate(LEDAMARILLO, 0, 100) != 0)
{
fprintf(stderr, "Error while creating PWM pin\n");
return 1;
}

int value_PWM = 0;
int delay_PWM = 100;

/*    Activación del led PWM
*    @softPwmWrite: escritura del GPIO PWM
*    Param 1: nombre pin GPIO PWM
*    Param 2: valor PWM (límitew inferior: 0, superior: 100)
*/
{
softPwmWrite(LEDAMARILLO, value_PWM);
delay(delay_PWM);//cada delay_PWM segundos cambia el led
}
```

1.5.4.2 Ejercicio 2: Control de la luminosidad de un LED mediante PWM de Wiring Pi

Escribir un programa que encienda un LED conectado al pin físico 7 (pin lógico Wiring Pi: 7). Conectar a ua esa salida una señal PWM que vaya aumentando de ancho de 0 a 200 y después de 200 a 0.

1.5.5 Uso de interrupciones con Wiring Pi

El sistema de consulta por encuesta mantiene al procesador ocupado aunque no se esté procesando nada. La versión más reciente del kernel de Wiring Pi (ampliada con el código de manejo de interrupciones GPIO) tiene una instrucción que permite establecer la espera por interrupción asociada a un pin de entrada y la rutina que se ejecutará cuando se produzca la interrupción. Esto libera el procesador para realizar otras tareas mientras espera. El anexo del capítulo **¡Error! No se encuentra el origen de la referencia.** contiene información detallada y ejemplos del uso de las interrupciones con Wiring Pi

Interrupciones	
Función	Funcionamiento
int wiringPiISR (int pin, int edgeType, void (*function)(void)) ;	Llama a la rutina de atención asociada al pin especificado. edgeType: INT_EDGE_FALLING, INT_EDGE_RISING, INT_EDGE_BOTH, o INT_EDGE_SETUP

El GPIO puede configurarse para interrumpir por flanco de subida, de bajada o ambos flancos de la señal entrante.

int wiringPiISR (int pin, int edgeType, void (*function)(void)) ;

Esta función apunta a la rutina/función de atención a la interrupción recibida en el pin especificado. El parámetro edgeType es **INT_EDGE_FALLING, INT_EDGE_RISING, INT_EDGE_BOTH,** o **INT_EDGE_SETUP**. Si se trata de **INT_EDGE_SETUP** no se producirá inicialización del pin -se supone que ya se ha configurado el pin en otro lugar (por ejemplo, con el programa **gpio**), pero si especifica uno de los otros tipos, el pin se exportará y se inicializará como se especifica (lo que se hace a través de una llamada adecuada al programa de utilidad **gpio**, que debe estar disponible).

El número de pin se puede suministrar en los modos wiringPi nativo, BCM_GPIO, físico o Sys. Esta función funciona en cualquier modo y no necesita privilegios de root para ejecutarse.

1.5.5.1 Ejercicio: control de un LED mediante Wiring Pi por interrupción

Escribir un programa que haga lo mismo que el anterior (1.5.3.1), pero en vez de leer el pulsador cíclicamente, espere una interrupción desencadenada por el pulsador.

¿Cuándo es mejor usar interrupciones que encuesta? ¿Por qué?

1.5.6 Threading con Wiring Pi

WiringPi proporciona funciones auxiliares que permiten gestionar interrupciones externas, lanzar un nuevo hilo desde un programa, y administrar la prioridad de programa (o hilo/thread). Los subprocesos se ejecutan simultáneamente con el programa principal y se pueden utilizar para múltiples propósitos. Para obtener más información sobre los hilos, se puede buscar "Posix Threads". Además aporta instrucciones de

bloqueo para implementar acceso a las variables y recursos compartidos en exclusión mutua. El anexo **¡Error! No se encuentra el origen de la referencia.** "**¡Error! No se encuentra el origen de la referencia.**" contiene información detallada y ejemplos del uso de threads con Wiring Pi.

Threads: concurrencia, comunicación y sincronización	
Función	Funcionamiento
int piHiPri (int prioridad);	Establece la prioridad del programa/hilo [0-90] (si se ejecuta como root)
int piThreadCreate (nombre);	Crea un subproceso cuyo c es una función previamente declarada usando **PI_THREAD**
piLock (int keyNum) ;	Bloquea el acceso a una variable. keyNum: [1-3]
piUnlock (int keyNum);	Desbloquea el acceso a una variable. keyNum: [1-3]

1.5.6.1 Ejercicio: Programación con threads de Wiring Pi

Escribir un programa que cree tres threads que escriban su nombre por pantalla. Para que no sigan la misma secuencia, introducir esperas diferentes en cada uno de ellos (por ejemplo, 1, 2 y 3 segundos).

6 ANEXO: Configuración de la Raspberry Pi para el Laboratorio de Robótica

Atención: estos ajustes sólo se realizarán si se ha verificad que la Raspberry Pi no estaba bien configurada para el laboratorio de Robótica

6.1 Conexiones

Si no está en su lugar, introducir la memoria microSD. La que usamos en el laboratorio viene formateada y con el Noobs 1.4 cargado. Si usamos una microSD vacía, habría que formatearla y descargar el SO desde la web.

Conectar

- un teclado y un ratón USB
- un adaptador de HDMI-VGA a la pantalla disponible.
- un adaptador USB-WIFI
- una fuente de alimentación. La Raspberry PI 2 usa alimentación de 5V a 2A aunque internamente trabaja con 3.3V.

Al conectar la fuente de alimentación se activa todo el sistema y se carga el SO. La primera vez aparece la lista de comandos de configuración (de 1 a 9). Para salir, ir a "finish" con el tabulador. Si se quiere entrar de nuevo, ejecutar el comando " sudo raspi-config".

El usuario por defecto es "pi" con password "raspberry". Este usuario estará activo, aunque no se recomienda usarlo para las prácticas. Lo usaremos cuando el usuario de prácticas falle, para reparar la cuenta de grupo. Cada grupo tiene un usuario "grupoXY" con password "grupoXY", dónde XY es el número de grupo.

6.1.1 Fijar IP para la red WiFi

La red que usamos en la asignatura se llama "RaspberryPiLab", utiliza el sistema WPA2_personal y tiene como contraseña "robopi2015".

Para mayor comodidad cada grupo tiene una dirección de IP fija para acceso a la red WiFi. Está configurada de manera que tiene relación con la etiqueta de la RaspberryPI de cada grupo. Para ello se modifica el archivo "interfaces":

```
~$ sudo nano /etc/network/interfaces
```

Por ejemplo para el equipo "robopi**XY**" (donde XY es el número de grupo) habría que añadir:

```
iface wlan0 inet static
address 192.168.1.1XY
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
```

6.1.2 Actualizar el sistema operativo

Es recomendable actualizar el sistema operativo a la última versión:

```
~$ sudo apt-get update
~$ sudo apt-get upgrade
```

Para actualizar la versión de Raspbian hay que ejecutar:

```
~$ sudo apt-get dist-upgrade
```

Una de las últimas actualizaciones ha cambiado el gestor de redes WiFi clásico (wpa-gui) por otro más amigable, que se puede obtener usando el comando:

```
~$ sudo apt-get install raspberrypi-net-mods
```

6.1.3 Añadir nuevo usuario

En la asignatura cada grupo tiene un usuario llamado grupoXY donde XY es el número del grupo. Si hace falta crearlo, usa el siguiente comando:

```
~$ sudo adduser grupoXY
```

Para que el nuevo usuario tenga permisos de administrador. Hay que añadir una línea con la configuración en el fichero "/etc/sudoers", similar a la línea de configuración del usuario pi. Primero abrimos el archivo:

```
~$ sudo nano /etc/sudoers
```

Se añade la siguiente línea con el número del grupo correspondiente al final del fichero:

```
~$ grupoXY ALL=(ALL) NOPASSWD: ALL
```

La sesión gráfica puede dar problemas si el nuevo usuario no se añade también a los grupos video y audio mediante los comandos:

```
~$ sudo usermod -aG video grupoXY
~$ sudo usermod -aG audio grupoXY
```

Para eliminar a un usuario:

```
sudo deluser –remove-home grupoXY
```

6.1.4 Cambiar password

El password por defecto será el mismo que el nombre del grupo. Se puede cambiar el password usando el siguiente comando:

```
~$ sudo passwd user_name
```

6.1.5 Cambiar el hostname

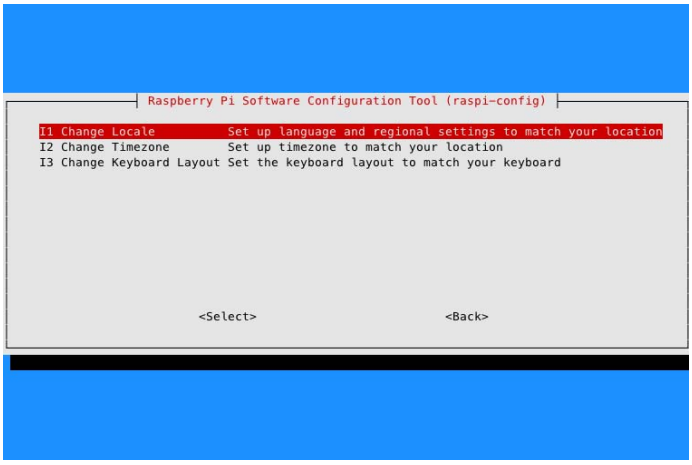
En cada máquina tiene pegada una etiqueta en la que aparece el nombre que la identifica: "ROBOPIXY". Para hacer coincidir el nombre de la etiqueta con el nombre virtual de la máquina en la red se cambia en las opciones de configuración:

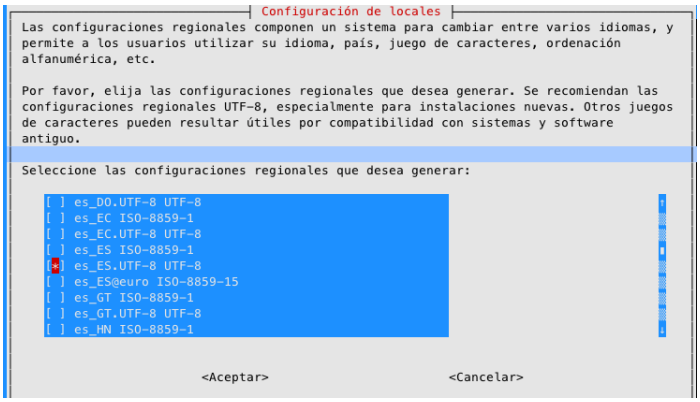
```
~$ sudo raspi-config
```

Seleccionar en (8) "Advanced options" la opción (A2) "Hostname" y escribir en minúsculas el nombre asociado, por ejemplo "robopiXY". El *prompt* de la máquina debería de verse de la siguiente manera:

```
grupoXY@robopiXY ~$
```

6.1.6 Configurar idioma





Para localizar el idioma y otros parámetros, debemos usar las opciones de configuración:

```
~$ sudo raspi-config
```

Seleccionando el menú de internacionalización (4), hay varias opciones para cambiar el idioma (I1 – I2 – I3):

- *Locale*: Traduce los mensajes del sistema, seleccionar la opción "es_ES.UTF-8 UTF-8" usando las flechas y activarlo usando la barra espaciadora
- *Timezone*: seleccionar la hora de la raspberry pi (Europa / Madrid)
- *Teclado*: Seleccionar "Spanish" a través de los sucesivos menús. Esto permite el uso de caracteres especiales tales como la ñ, pero no funciona si se accede a través de ssh sin conectar un teclado a la PI).

6.1.7 Actualizar Firmware de la Raspberry Pi

Tener el firmware de la Raspberry PI actualizado puede solucionar algunos problemas, como por ejemplo que la Raspberry PI no arranque cuando no hay un monitor conectado. Sin embargo, las versiones más recientes del firmware pueden fallar con algunas librerías. Por ello, esta operación puede hacer que la configuración actual de dispositivos no funcione. Hay que actualizar el firmware sólo si es necesario y con mucha precaución. Antes conviene comprobar la versión del firmware:

```
~$ /opt/vc/bin/vcgencmd version
```

resultado:

```
Aug 23 2015 19:17:06
Copyright (c) 2012 Broadcom
version 41b1f861989dc4d1bbe19a52361a248ae57ee82b (clean)
    (release)
```

Copiar el código de la versión por si hay que restablecerla.

Para actualizar el firmware a la última versión:

```
~$ sudo rpi-update
```

Para restablecer el firmware a la versión anterior:

```
~$ sudo rpi-update 41b1f861989dc4d1bbe19a52361a248ae57ee82b
```

7 Multithreading e interrupciones con Wiring Pi

7.1 Interupciones y multithreading con Wiring Pi[10]

WiringPi proporciona funciones auxiliares que permiten gestionar interrupciones externas, lanzar un nuevo hilo desde un programa, y administrar la prioridad de programa (o hilo/thread). Los subprocesos se ejecutan simultáneamente con el programa principal y se pueden utilizar para múltiples propósitos. Los hilos funcionan como los "Posix Threads".

7.2 Procesamiento concurrente (multi-threading)

WiringPi ofrece una interfaz simplificada para la implementación de los hilos Posix en Linux, así como un mecanismo (simplificado) para acceder a las exclusiones mutuas.

Usando estas funciones se puede crear un nuevo proceso (una función dentro del programa principal) que se ejecuta simultáneamente con el programa principal y pasarse entre ellos las variables de forma segura utilizando los mecanismos de exclusión mutua (mutex).

- **int piThreadCreate (nombre);**

Esta función crea un subproceso que ejecuta otra función en el programa que se ha declarado previamente mediante **PI_THREAD**. Esta función se ejecuta simultáneamente con su programa principal. Por ejemplo se puede hacer que esta función espere una interrupción mientras el programa continúa realizando otras tareas. El hilo puede indicar un evento o una acción mediante el uso de variables globales para comunicarse de nuevo con el programa principal u otros subprocesos.

Las funciones de hilo se declaran de la siguiente manera:

```
PI_THREAD (myThread)
{
.. Código que se ejecuta simultáneamente con el programa
    principal, probablemente en un bucle infinito}
}
```

y se inicia en el programa principal con

```
x = piThreadCreate (myThread);
if (x != 0) printf ("it didn't startn")
```

Esto no es más que una interfaz simplificada al mecanismo de subprocesos Posix que soporta Linux. Para saber más, se puede consultar el manual de los hilos Posix (man pthread).

```
piLock (int keyNum) ;
piUnlock (int keyNum) ;
```

Estas instrucciones permiten sincronizar las actualizaciones de las variables de su programa principal a cualquier subproceso que se ejecuta en su programa. KeyNum es

[10] Wiring Pi: GPIO Interface library for the Raspberry Pi http://wiringpi.com/reference/priority-interrupts-and-threads/

un número de 0 a 3 y representa una "clave". Cuando otro proceso intenta bloquear la misma clave, se paralizará hasta que el primer proceso haya desbloqueado esa clave.

Se necesita utilizar estas funciones para asegurarse de que se obtienen resultados válidos al intercambiar datos entre un programa principal y un hilo. De lo contrario, es posible que el hilo pueda despertar durante la copia de datos y cambiarlos de modo que la copia quede incompleta o no sea válida. Consultar el programa wfi.c en el directorio de ejemplos para ver un ejemplo.

7.3 Prioridad de programa o de hilo

int piHiPri (int prioridad);

Esta instrucción aumenta la prioridad del programa (o subproceso en un programa multihilo) y permite una planificación en tiempo real. El parámetro de **prioridad** va de 0 (predeterminado) a 99 (máximo). Esto no hace que el programa vaya más rápido, pero le da más tiempo cuando se están ejecutando con otros programas. El parámetro de prioridad es relativo a los otros, por lo que se puede establecer una prioridad de programa a 1 y otra prioridad a 2 y tendrá el mismo efecto que establecer una a 10 y la otra a 90 (siempre y cuando no se ejecuten otros programas con prioridades más altas).

El valor de retorno es 0 para éxito y -1 para error. Si se devuelve un error, el programa puede consultar la variable global errno, según las convenciones habituales.

Nota: Sólo los programas que se ejecutan como root pueden cambiar su prioridad. Si se llama desde un programa no root, no sucede nada.

7.4 Interrupciones

La versión más reciente del kernel (ampliada con el código de manejo de interrupciones GPIO) permite al programa esperar una interrupción. Esto libera el procesador para realizar otras tareas mientras espera. El GPIO puede configurarse para interrumpir por flanco de subida, de bajada o ambos flancos de la señal entrante.

- **int wiringPiISR (int pin, int edgeType, void (*function)(void)) ;**

Esta función apunta a la rutina/función de atención a la interrupción recibida en el pin especificado. El parámetro edgeType es **INT_EDGE_FALLING, INT_EDGE_RISING, INT_EDGE_BOTH, o INT_EDGE_SETUP**. Si se trata de **INT_EDGE_SETUP** no se producirá inicialización del pin -se supone que ya se ha configurado el pin en otro lugar (por ejemplo, con el programa gpio), pero si especifica uno de los otros tipos, el pin se exportará y se inicializará como se especifica (lo que se hace a través de una llamada adecuada al programa de utilidad gpio, que debe estar disponible).

El número de pin se puede suministrar en los modos wiringPi nativo, BCM_GPIO, físico o Sys.

Esta función funciona en cualquier modo y no necesita privilegios de root para ejecutarse.

La rutina/función se llamará cuando se active la interrupción. Cuando se dispara la interrupción, se borra del dispatcher antes de llamar a la función de atención, por lo que si una interrupción posterior se activa antes de terminar el procesamiento de la actual no se perderá. Sin embargo, sólo puede seguir una interrupción más. Si se dispara más de una interrupción mientras se está atendiendo a otra, entonces se ignorarán.

Esta función se ejecuta con prioridad alta (si el programa se ejecuta utilizando sudo o como root) y concurrentemente con el programa principal. Tiene acceso completo a todas las variables globales, manejadores de archivos abiertos, etc.

7.5 Ejemplos

7.5.1 Ejemplo de lectura de una entrada por encuesta

```
/* Programa encuesta.c
* Enciende un LED conectado al pin físico 7 (pin lógico
    Wiring Pi: 7) * y lee cíclicamente un pulsador
    (normalmente en alto) conectado al * * pin físico 5
    (pin lógico Wiring Pi: 9)
*****************************************************/
#include <stdio.h>
#include <wiringPi.h>
#include <softPwm.h>

#define LED_PIN 7
#define BUTTON_PIN 9

/*Programa principal*/
int main()
{
int pulsador=1;
wiringPiSetup();

pinMode(BUTTON_PIN,INPUT);
pinMode(LED_PIN,OUTPUT);

printf("El LED se apagará cuando se pulse el pulsador más de
    un segundo\n ");

      while (pulsador == 1)
      {
        digitalWrite(LED_PIN,HIGH);
        delay (800);
        digitalWrite(LED_PIN,LOW);
        delay (200);
        pulsador = digitalRead(BUTTON_PIN);
        printf("El estado del pulsador es: %d \n",
    pulsador);
      }

printf("Pulsador pulsado \n");
return 0;
}
```

7.5.2 Lectura de una entrada por interrupción

```
/* Programa interrupcion.c
* Enciende un LED conectado al pin físico 7 (pin lógico
    Wiring Pi: 7) * y espera a la interrupción generada por
    el pulsador (normalmente en * alto) conectado al pin
    físico 5 (pin lógico Wiring Pi: 9)
*************************************************************/

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

#define LED_PIN 7
#define BUTTON_PIN 9
static volatile int fin=0;

/*Rutina de atención a la interrupción*/
void esperaInterrupcion(void)
    {
    fin=1;
    printf("Se ha pulsado el pulsador: %d \n", fin);
    }

/*Programa principal*/
int main()
{
int i, x, led;
wiringPiSetup();
pinMode(LED_PIN,OUTPUT);

/*Activa la espera por interrupción*/
wiringPiISR(BUTTON_PIN, INT_EDGE_BOTH, &esperaInterrupcion);

/*Enciende el LED mientras fin sea falso*/
printf("LED encendido. Esperando interrupción\n ");
while (fin==0)
      {
        digitalWrite(LED_PIN,HIGH);
        delay (900);
        digitalWrite(LED_PIN,LOW);
        delay (100);
      }

return 0;
}
```

7.5.3 Uso de threads

```
/*************************************************
*Programa test threads.c
*Crea dos threads que se ejecutan en paralelo con el main
*************************************************/
//inclusión librerias necesarias para el programa
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
//definición costantes
#define GLOBAL_UNIT_TIME 1000 // unidad de pausa temporal

//Código de cada thread
PI_THREAD (thread1)
{
  while(true)
  {
…
  }
}

PI_THREAD (thread2)
{
  while(true)
  {
…
  }
}

//Main program
int main(void)
{
//creación de los threads
  printf("Main: creando threads\n");
  int r,v;
  /*  @piThreadCreate: creatión de un thread. Param: nombre
     del thread; Return: 0 si no falla  */
     r = piThreadCreate(thread1);
     v = piThreadCreate(thread2);

  if((r != 0) || (v != 0)
  {
    printf("Error en la creación de un thread!\n");
    return 1;
  }

  for(;;)// Los programas de control no terminan nunca
  {
        …
  }
  return 0;
}
```