Uso de la Raspberry Pi 2 en Robótica

Raspberry Pi 2

- es un mini-ordenador basado en el microcontrolador de Broadcom BMC 2836 a 900Mhz con 1GB de RAM.
  - [https://www.raspberrypi.org/products/raspberry-pi-2-model-b/]
- admite diferentes sistemas operativos.
  - usaremos “Raspbian”, (versión de la distribución Linux Debian, adaptada al hardware de la Raspberry Pi).
  - Raspbian : [https://www.raspberrypi.org/documentation/raspbian/]

Configuración de la Pi para el laboratorio de Robótica

- Usaremos Raspberry Pi 2 para controlar el iRobot create.
  - Las placas Raspberry Pi 2 están preconfiguradas
  - Conviene verificarlo y actualizar el sistema operativo (Si hay que reconfigurarlas -> ANEXO 1)
- 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 (a 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**".
- Login del usuario por defecto:
  - *login*: pi
  - Password raspberry
- Login del grupo
  - login: grupoXY
  - Password: grupoXY (dónde XY es el número de grupo, XY: {01,02,...10})
- Login del supersusuario
  - Login: root
  - password : toor

Verificación, mantenimiento y actualización

1. Iniciar sesión:

- robopiXY login: **grupoXY**
- Password: **grupoXY**

grupoXY@robopyXY ~$

2. Comprobar 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)
```

3. Actualizar el sistema operativo

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

4. Cómo apagar y reiniciar (halt and reboot)

- Reiniciar:
  - ~$ sudo shutdown –r now

  o
  - ~$ sudo reboot
- Apagar:
  - ~$ sudo shutdown –h now

  o
  - ~$ sudo halt

Modo de trabajo

Puesto de trabajo

- Sistema Operativo: Windows 7
- Entorno de desarrollo: NetBeans
- Lenguaje: C/C++
- Conexión WiFi a red “robopi2015”

Raspberry Pi 2

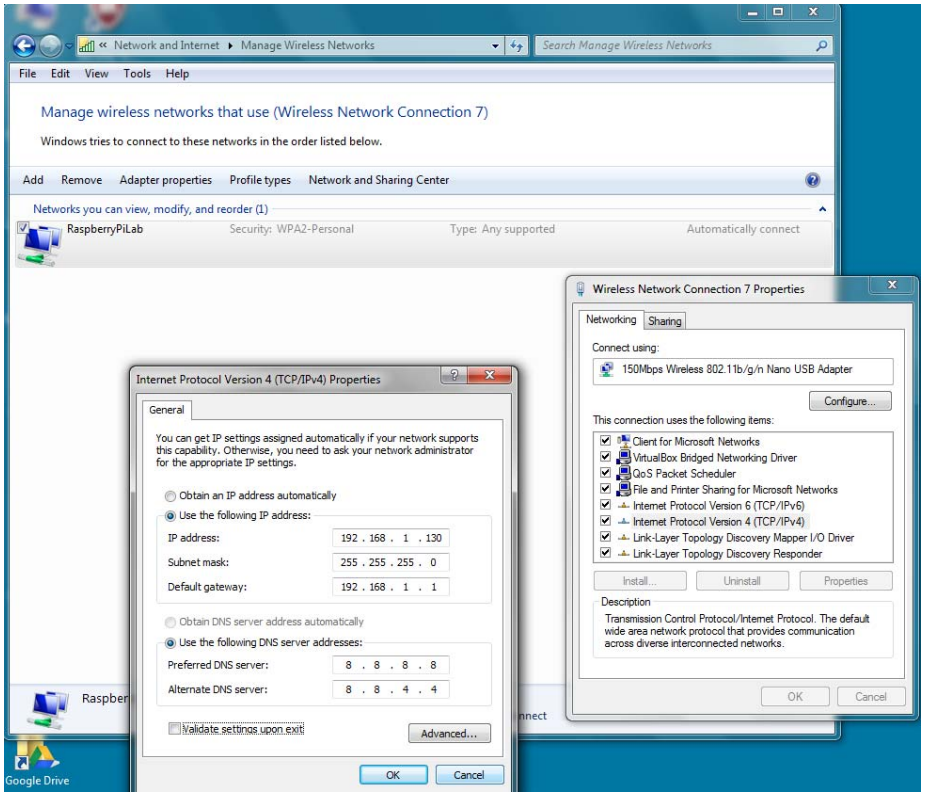
- Sistema Operativo: Raspbian
- Entorno de desarrollo: compiladores gcc/g++
- Conexión WiFi a red “robopi2015”

Router “RaspberryPiLab”
Pass: “robopi2015”

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

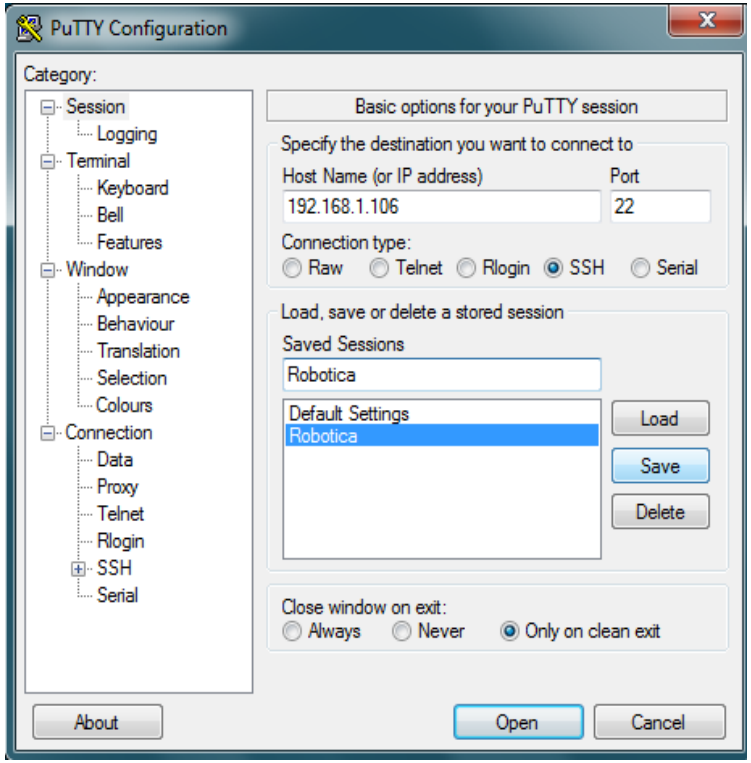
- Usaremos la red wifi “RaspberryPiLab” con dos adaptadores WiFi USB.
- Configurar la dirección IP de los PC en un rango que permita trabajar a las Raspberry PI 2 sin conflictos: [131–140].

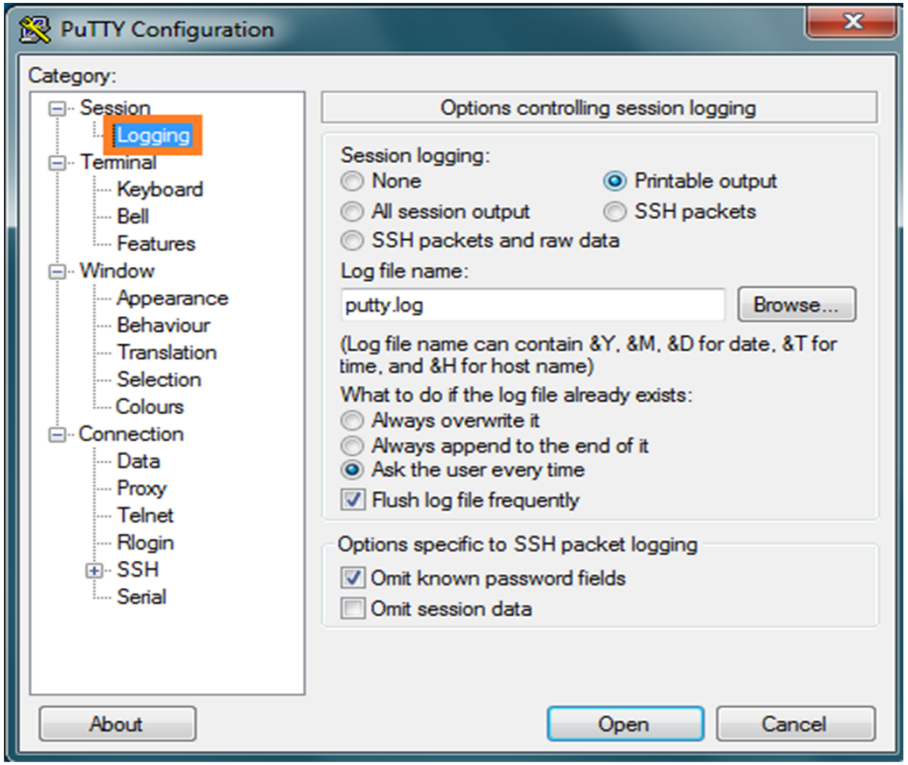
Asignación de números de IP a cada grupo		
Gr	**Raspberry Pi**	**PC**
1	192.168.1.101	192.168.1.131
2	192.168.1.102	192.168.1.132
3	192.168.1.103	192.168.1.133
4	192.168.1.104	192.168.1.134
5	192.168.1.105	192.168.1.135
6	192.168.1.106	192.168.1.136
7	192.168.1.107	192.168.1.137
8	192.168.1.108	192.168.1.138
9	192.168.1.109	192.168.1.139
10	192.168.1.110	192.168.1.140



Conexión remota SSH

- Usamos un cliente ssh para conectarnos de manera remota: **PuTTY**
  - conexión remota protocolo SSH (Secure Shell): dirección IP de la Raspberry PI 2; puerto 22.
  - permite almacenar un Log con los comandos y respuestas de la sesión remota si se configura en: Session -> Logging
- Se puede descargar gratuitamente desde: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html





Programación en C en la Raspberry Pi

- Se pueden escribir programas en C/C++ y compilarlos sin necesidad de equipos exteriores.
- Raspbian incluye editores, compiladores gcc y g++,, linkeditor, etc.

Ejercicio 1: "Hola mundo" en C

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

- 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

Programación Raspberry Pi 2 usando el entorno NetBeans

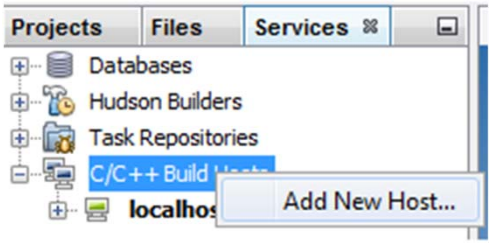
Para desarrollar programas utilizaremos NetBeans en Windows 7

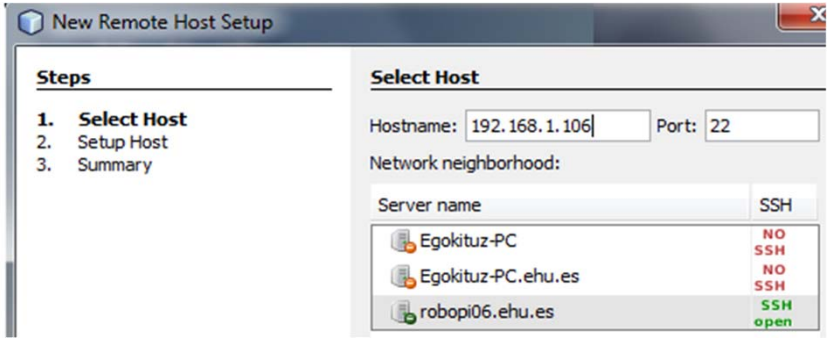
- Descarga NetBeC/C++ Remote Development - NetBeans IDE Tutorial: ans: https://netbeans.org/downloads/
- https://netbeans.org/kb/docs/cnd/remotedev-tutorial.html

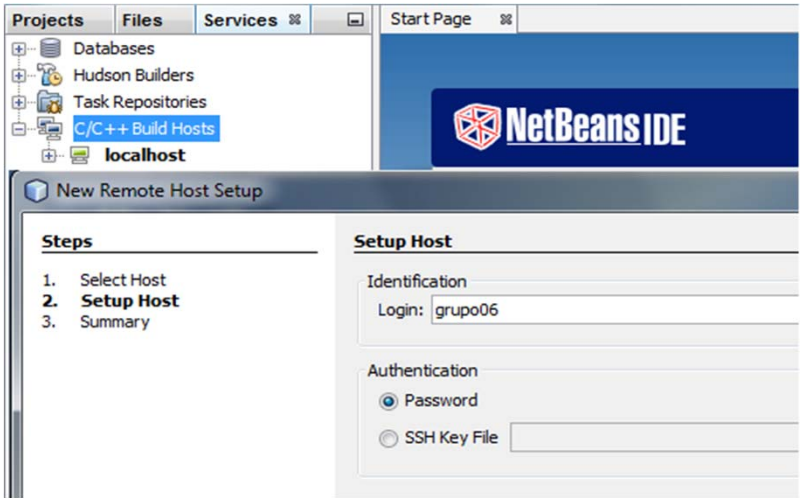
- Fijar un host remoto (la Raspberry Pi) para compilar los programas.
  - Hostname: dirección IP de la Raspberry Pi
  - login: el usuario que tenemos asignado
  - Password: el del grupo
  - aparece una advertencia de seguridad que hay que aceptar

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

- Dado que los compiladores y el código que se genera están en la Raspberry Pi, para compilar es necesario que haya una conexión activa

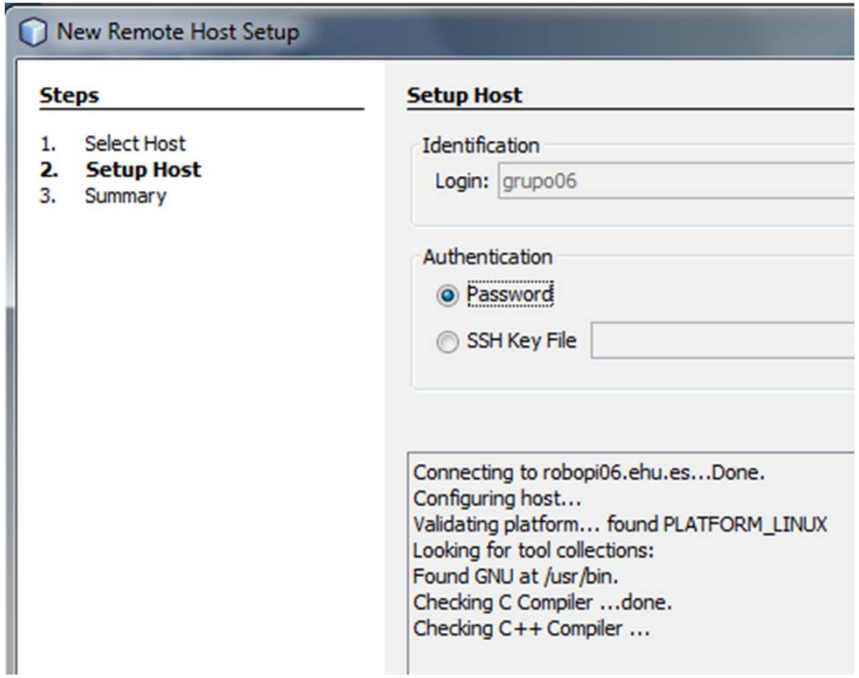


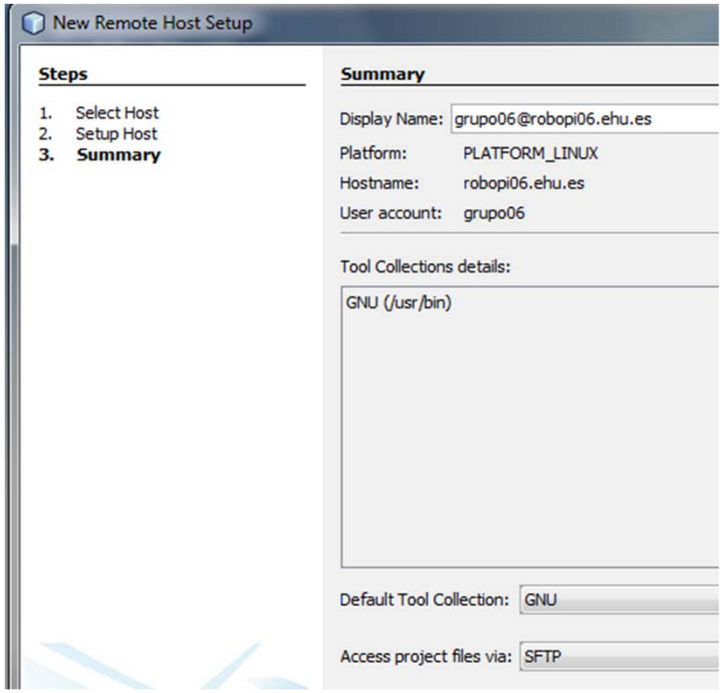




Programación Raspberry Pi 2 usando el entorno NetBeans

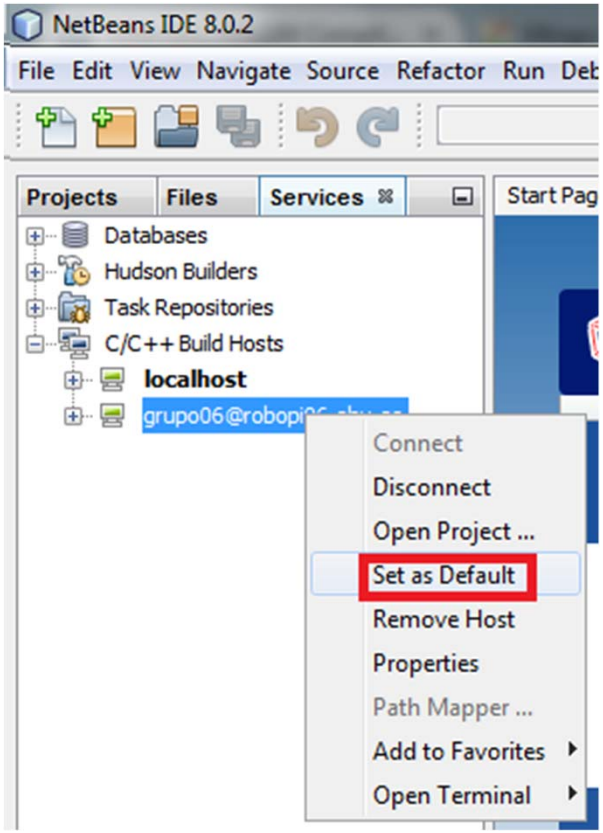
- Hay que asegurarse de que use el protocolo SFTP para copiar archivos (puede haber problemas a la hora de crear un nuevo proyecto)



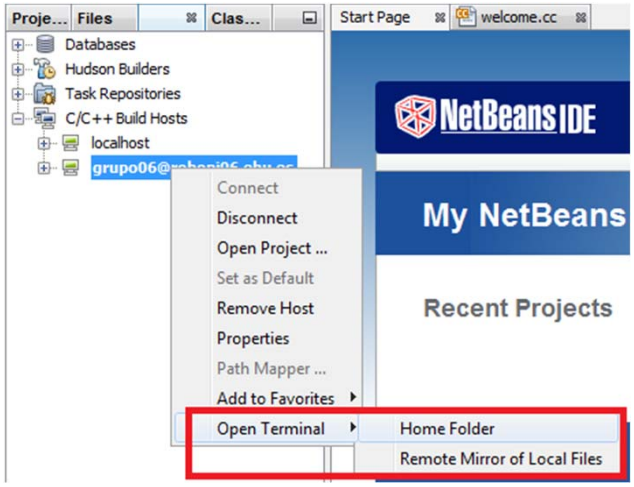


Programación Raspberry Pi 2 usando el entorno NetBeans

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

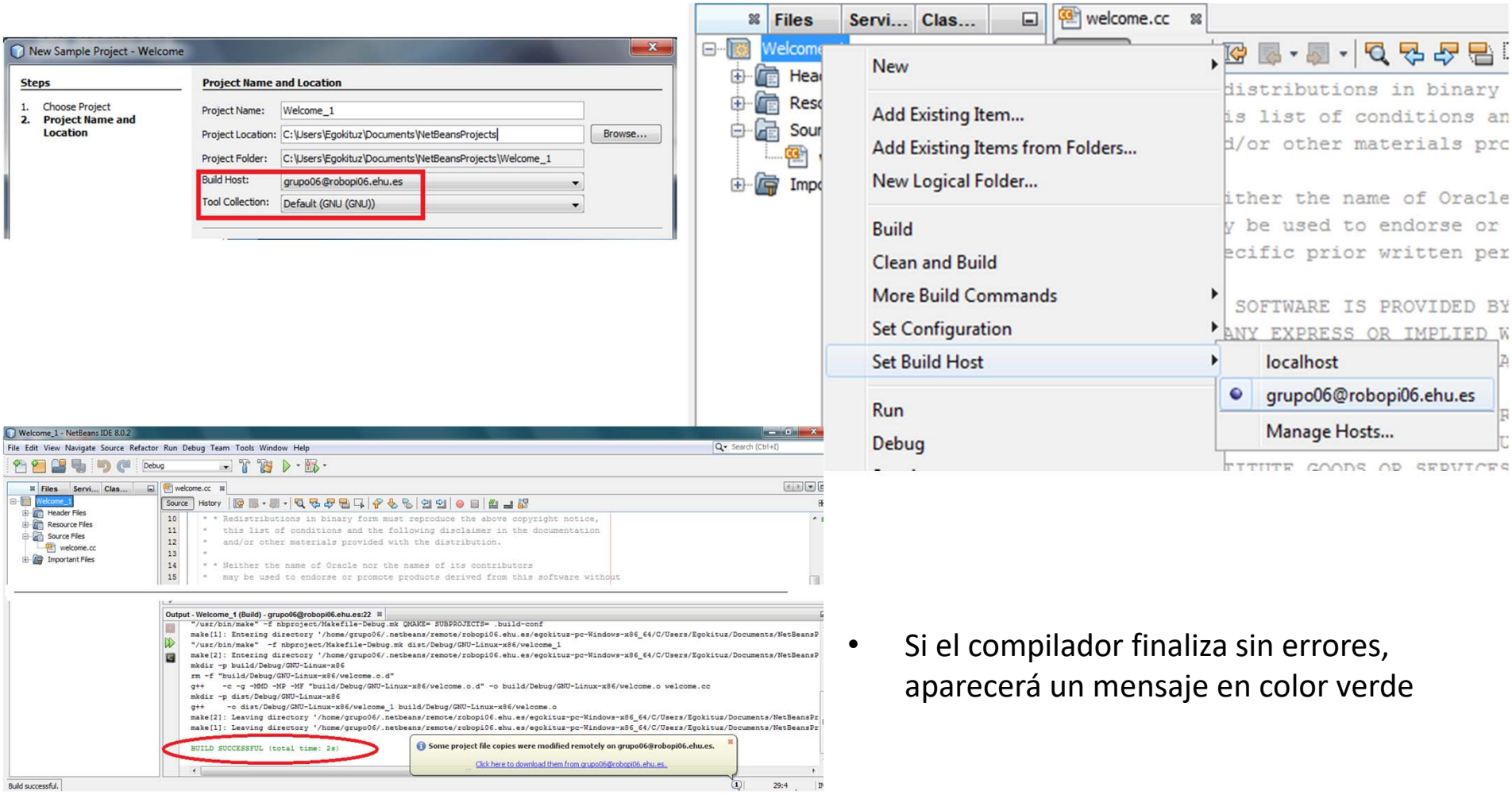


- Es posible abrir un cliente SSH remoto usando las opciones “Open Terminal” de NetBeans.
- Equivale a la conexión que habíamos hecho con PUTTY



Programación Raspberry Pi 2 usando el entorno NetBeans

- Hay que asegurarse que el “Host” es el adecuado antes de compilar



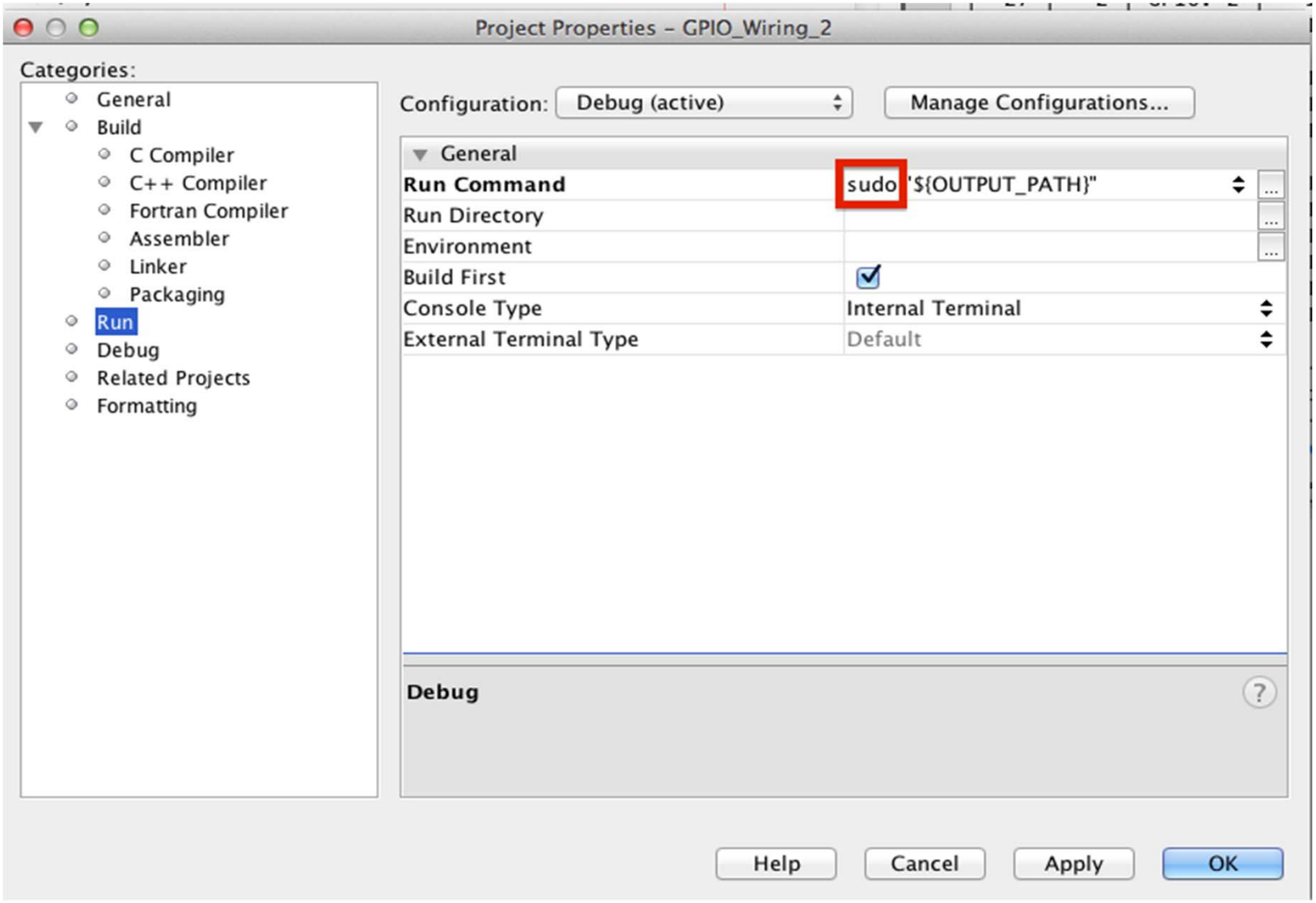
- Si el compilador finaliza sin errores, aparecerá un mensaje en color verde

Ejecución en la Raspeberry Pi

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:
       - **~$ passwd root <R> toor**
  - En la ventana de NetBeans:
    - 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"

Ejecución en la Raspeberry Pi



Prueba NetBeans-Raspberry Pi

- Para cargar nuevos proyectos
  a. Crear un nuevo proyecto modificando uno de los ejemplos:

     > File: New Project: Samples: Ejemplo_X

  b. Si partís de una plantilla usad:

     >File: "import project"

  De otra manera puede haber problemas para que encuentre los ficheros

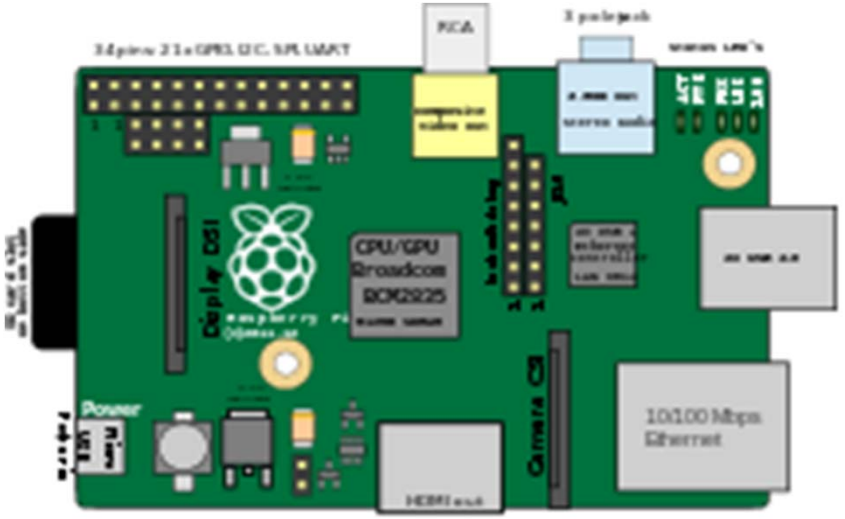
- Probad que los programas de ejemplo incluidos en NetBeans se ejecutan correctamente.
  - Por ejemplo el "Project welcome"

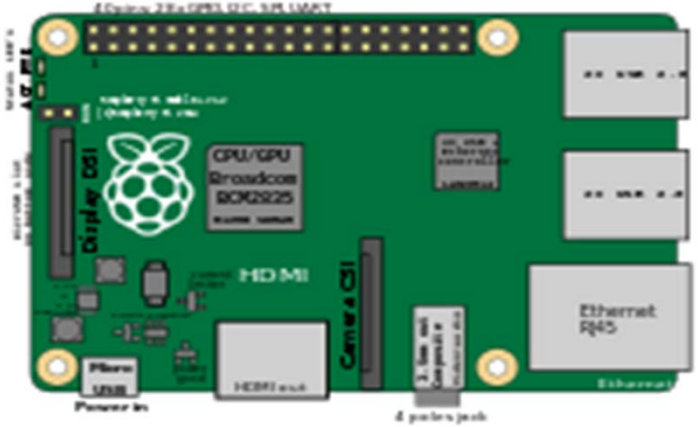
    > File: New Project: Samples: Welcome

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.
    - De los 40 pines,
    - 26 son GPIO y
    - los otros son para alimentación o tierra
    - además de dos pines ID EEPROM (que no convine usar).
  - Este puerto permite el uso de los periféricos del BMC2836 para comunicaciones línea serie, protocolos i2c, SPI, etc.

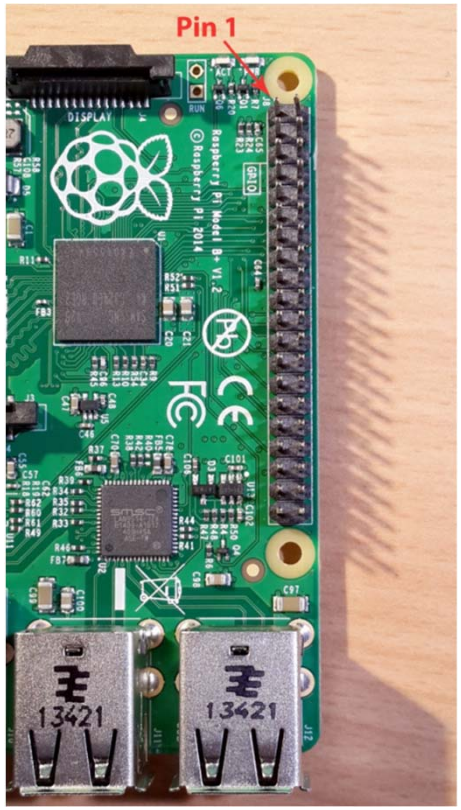
- Características generales
  - 900 Mhz Quad Core , ARM Cortex-A7, 1 GB RAM
  - Basado en el SOC BCM2835/BCM2836
  - Entrada / Salida configurable: 26x GPIO, USART, I2C





Periféricos BCM2836

- Asignaciones del conector de 40 pines de la Raspberri Pi.
- Es posible que el software que utilicemos para controlar las estradas salidas, asigne nombres diferentes a esos mismos pines.



Raspberry Pi2 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 1
26/01/2014

http://www.element14.com

Precauciones importantes:

- No introducir más de 3.3V en un pin GPIO
- No consumir más de 3mA por los pines de salida (asegurarse de protegerlos usando resistencias)

- No 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

Acceso al puerto GPIO desde la Raspbian

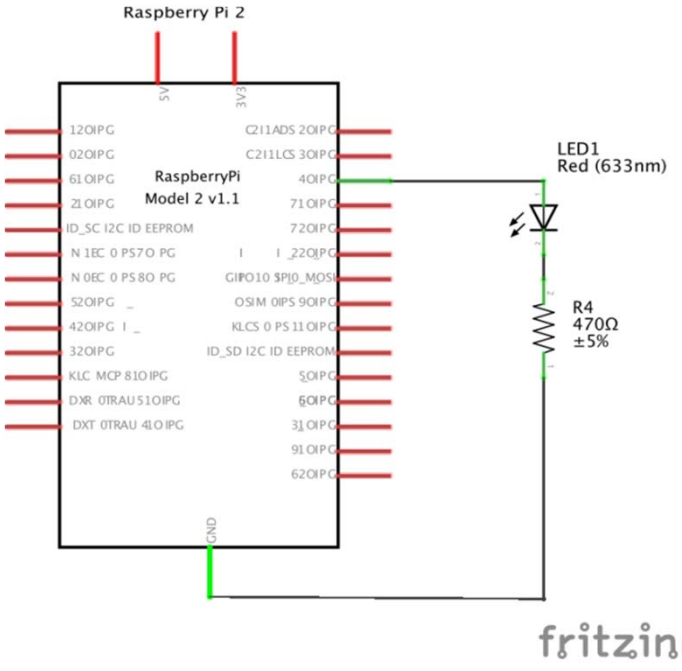
Para programar la función de los pines de entrada salida hay que manipular los ficheros que se encuentran en la carpeta */sys/class/gpio* lo que **requiere permisos de root**.

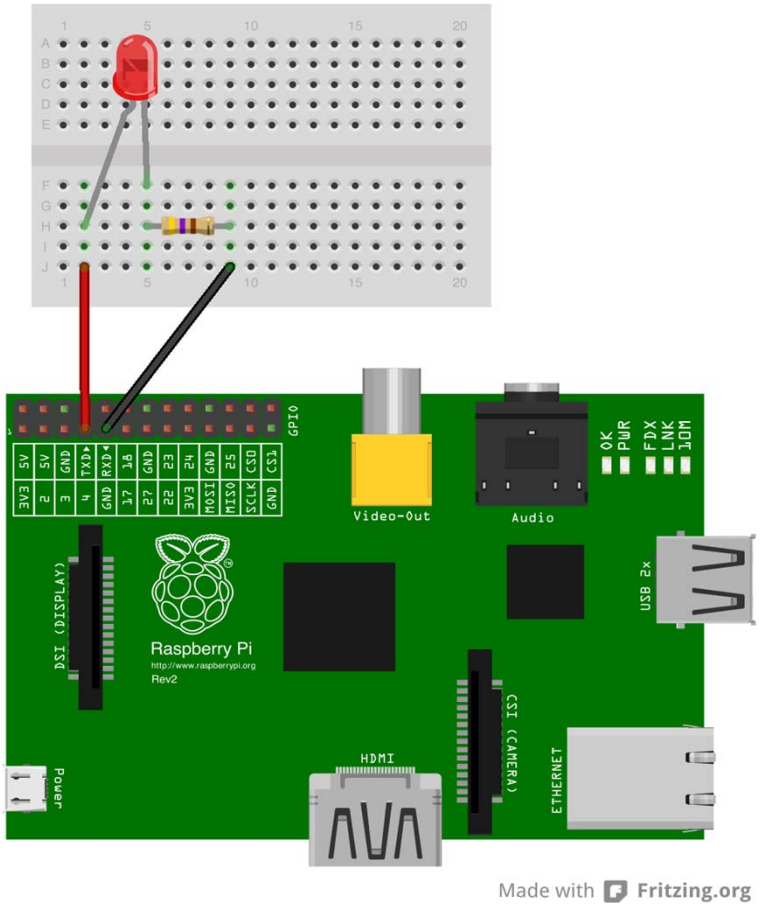
- Para saber si tiene permisos de acceso al grupo GPIO ejecutar:
  `~$ groups grupoXY`
- Si responde
  `grupoXY: grupoXY audio video gpio`
  es que tiene permiso para usar GPIO.
- Si no fuera así, hay que añadir el usuario grupoXY al grupo GPIO:
  `~$ sudo usermod –aG gpio grupoXY`
  Ahora, repetir la 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`
- Hay que configurarlo para que se comporte como pin de entrada o salida (in/out)
  `~$ echo out > /sys/class/gpio/gpio4/direction`
- En caso de que esté configurado como salida se puede escribir 1 o 0 para asignarle un voltaje “High” o “Low”.
  `~$ echo 1 > /sys/class/gpio/gpio4/value`
- Tanto si es una salida como si es una entrada se puede leer el valor del voltaje en el pin con el comando “cat”
  `~$ cat /sys/class/gpio/gpio4/value`

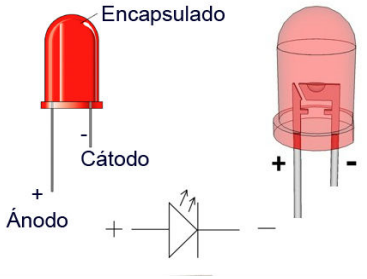
Prueba de salida de la GPIO

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
- Placa de montaje
- Cables de conexión (se suele usar rojo para la señal y negro para tierra/GND)
- 1 LEDs
- 1 pulsador
- dos resistencias de 470 Ω y una de 1KΩ
- Tira de 9 pines







Prueba de salida de la GPIO

Encender un led conectado al pin 7

1. Desde Raspbian

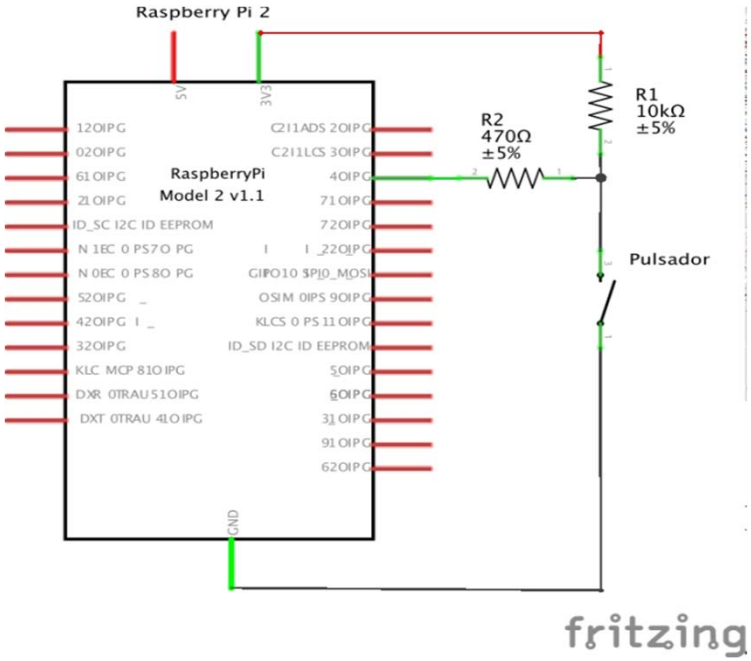
```
~$ echo 4 > /sys/class/gpio/export
~$ echo out >
  /sys/class/gpio/gpio4/direction
~$ echo {1/0} > /sys/class/gpio/gpio4/value
```

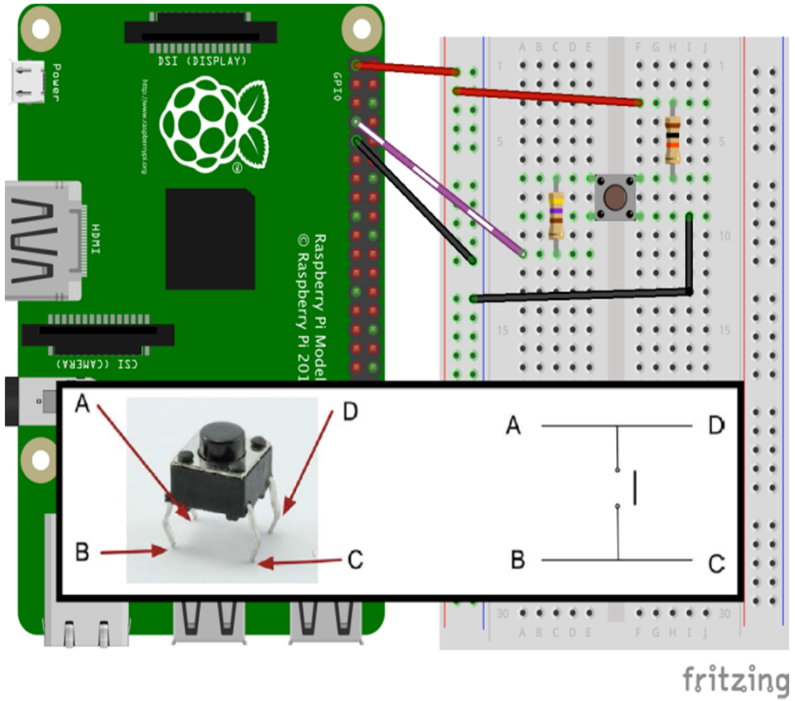
2. Desde un programa en C

- usar la función `int system("comando")` que permite ejecutar comandos del Shell.

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

Prueba de entrada por la GPIO





Prueba de entrada por la GPIO

Leer el estado de un pulsador conectado al pin 5 (correspondiente al GPIO3)

1. Desde Raspbian

- 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
```

2. Desde un Programa en C

- ¿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

Librería Wiring Pi

Usaremos la librería Wiring Pi para acceder desde C/C++ a los pines del puerto J8 de la Raspberry Pi 2: http://wiringpi.com/

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 se puede pasar al siguiente punto (devolverá el mensaje: "F*atal destination path 'wiringPi'" already exists and is not an empty directory*")

```
~$ 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
```

- ccv

Librería Wiring Pi

La orden:

```
~$ gpio readal
```

devuelve la siguiente tabla, con la programación actual de los pines, donde:

- 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)

```
+-----+-----+---------+------+---+---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 |   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 |
+-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
```

Uso de las librería Wiring Pi

- Wiring Pi 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 incluir referencias a los archivos .h que definen las funciones a usar:

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

- Algunas funciones que usaremos en el laboratorio de robótica:

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

Compilación de programas con WiringPi en Raspbian

- 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

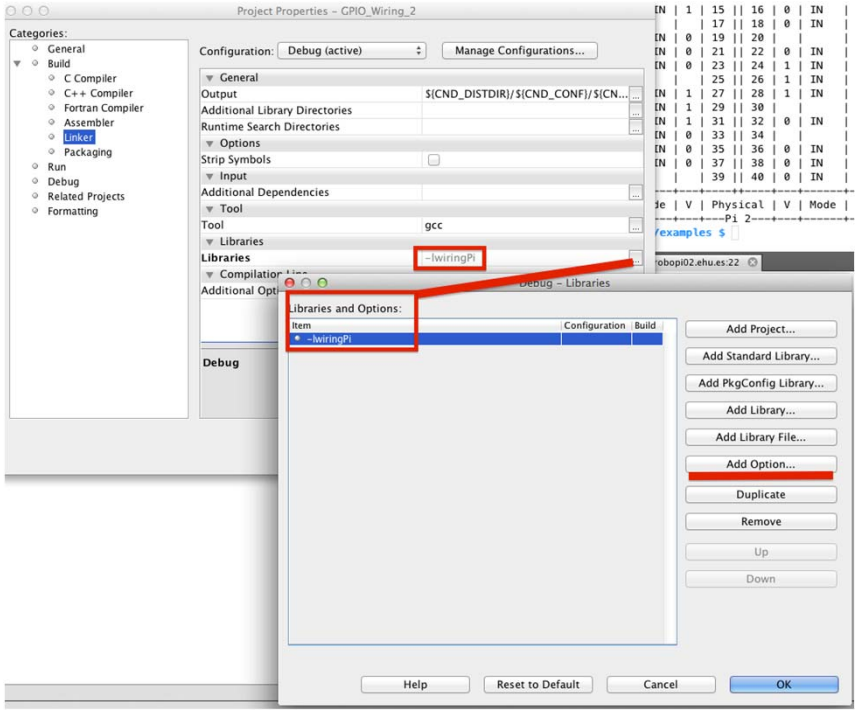
```
-lpthread
```

- Hay que ejecutar el programa como administrador:

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

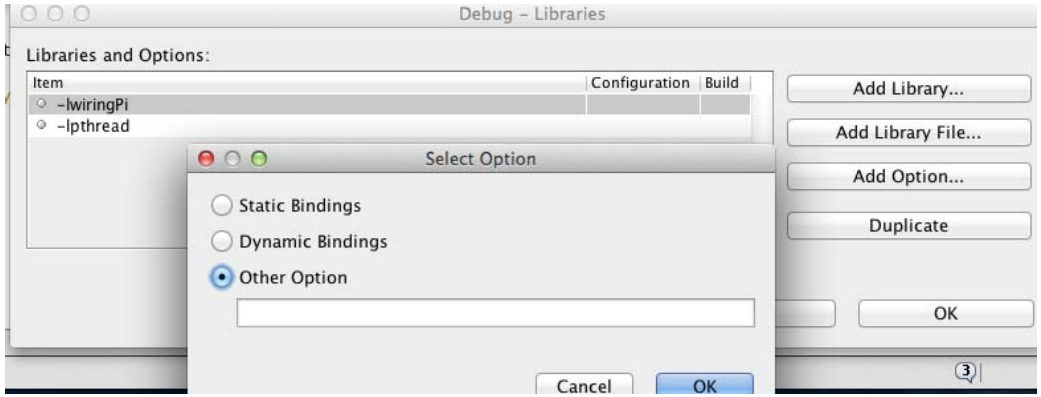
Compilación de programas con WiringPi en NetBeans

- Es necesario añadir las librerías en “propiedades del proyecto” > “Linker” de forma manual



- Escribir en la ventana "Other Option"

```
-lwiringPi
```



Gestión de interrupciones en Wiring Pi

- La consulta por encuesta mantiene al procesador ocupado aunque no se esté procesando nada.
- Wiring Pi permite establecer la espera por interrupción asociada a un pin de entrada
- cuando se produzca la interrupción se ejecutará una rutina de atención.

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

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;
}
```

Multithreading en Wiring Pi

- Wiring Pi permite desencadenar threads o hilos que se ejecutan concurrentemente con el programa principal.
- Además aporta instrucciones de bloqueo para implementar acceso a las variables y recursos compartidos en exclusión mutua.

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]

Uso de threads

```
/*Programa test threads.c :Crea 2 threads
que se ejecutan en paralelo con el main*/

//librerias necesarias para el programa
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>

//Código de cada thread
PI_THREAD (thread1) {
{
    while(true)
          {
          …
          }
}
PI_THREAD (thread2) {
{
    while(true)
          {
          …
          }
}
```

```
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 al crear un thread!\n");
    return 1;
    }
    for(;;)//Prog. de control: no termina
    {
    …
    }
return 0;
}
```

Materiales

- Putty
  - Cliente Secure SHell, linea de comandos remota
  - http://www.putty.org/
- NetBeans
  - Compilador C/C++
  - https://netbeans.org/downloads/
- Manuales
  - RaspberriPI para robótica
  - Presentación de la práctica