

iRobot_Framework

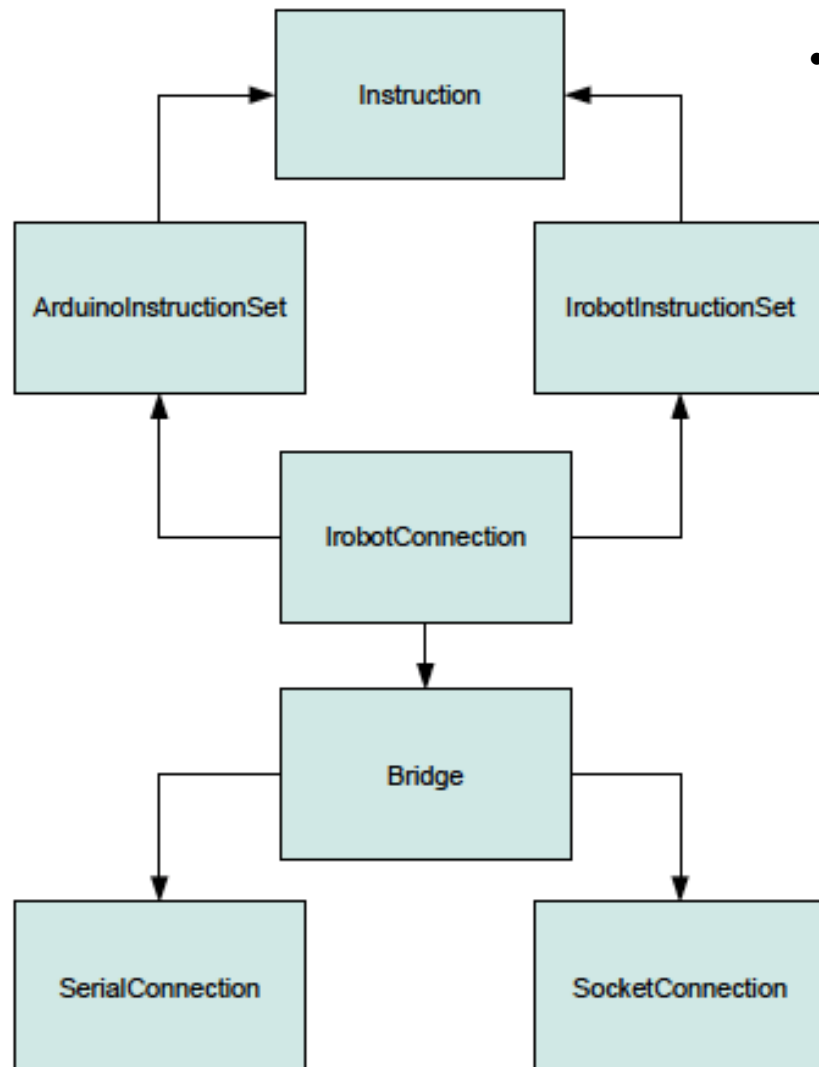
Robótica, Sensores y Actuadores
2019-2020

1. Estructura y uso del iRobot_Framework

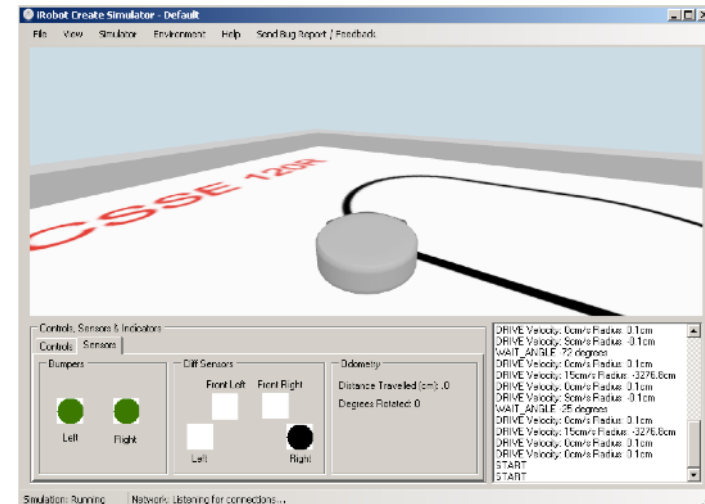
iRobot_Framework

- iRobot_Framework
 - es una máquina virtual que permite programar el iRobot Create en LAN C/C++
 - Aporta instrucciones de alto nivel, parametrizables (que el lenguaje de scripts no tiene)
- Versiones
 - inicial: Para Arduino, simulador, y BAM. Diseñada por Gorka Montero PFC (2012).
 - actual: para Raspberry Pi2. Extendida por Borja Gamecho (2015).

Librerías iniciales de IRobotConnection



- Usa de forma “transparente” el mismo código en 3 escenarios:
 - Simulador (de Rose-Hulman Inst.)



- Control mediante el módulo BAM / Línea serie
- Control mediante Arduino: extensión del código OI para sensores/actuadores adicionales

Aspecto de un programa de iRobot en C/C++

```
1  #include <iostream>
2  #include "../libs/IRobotConnection.h"
3  #include <dos.h>
4
5  using namespace std;
6
7  int main(int argc, char * argv[])
8  {
9
10     // Creamos un objeto robot que se conectará por el puerto x
11     IRobotConnection robot("COMx");
12
13     // Iniciamos la conexión
14     printf("Connecting... ");
15     robot.connect();
16     printf("Done!!\n");
17
18     // comando 128 start
19     robot.start();
20     Sleep(500);
21
22     // comando 132 modo full
23     robot.full();
24     Sleep(500);
25
26     ... resto de comandos ...
27
28     robot.disconnect();
29     return 0;
30 }
```

```
// Ejecutamos el comando 142 35 y mostramos el resultado por pantalla
cout << "Modo de funcionamiento: " << robot.updateSensor(iRobotSensors::OIMODE) << endl;

// Comando 132 modo full
robot.full();
Sleep(500); // Esperamos medio segundo a que cambie de modo

// Ejecutamos el comando 142 35 y mostramos el resultado por pantalla
cout << "Modo de funcionamiento: " << robot.updateSensor(iRobotSensors::OIMODE) << endl;

// Avanzamos durante 2 segundos a 200mm/s y paramos los motores
robot.driveDirect(200,200);
Sleep(2000);
robot.driveDirect(0,0);
```

iRobot_Framework

Funcionamiento:

- Traduce un conjunto de instrucciones (de alto nivel) en otro (de más bajo nivel)
- Para cada instrucción de alto nivel
 - la traduce a códigos de operación de iRobot Create
 - envía los códigos por la línea serie
 - espera la respuesta (si la hay)
- Estructura del software:
 - iRobot_framework se edita en NetBeans sobre PC/Windows
 - compila y genera código C/C++ en RaspberryPi/Linux
 - la máquina virtual se ejecuta sobre RaspberryPi

Contenido del SRC de iRobot_Framework

- Instruction.cpp e Instruction.h
- IRobotConnection.cpp e IRobotConnection.h
- IRobotInstructionSet.cpp e
IRobotInstructionSet.h
- Serial.cpp y Serial.h

Serial.ccp y Serial.h

- Gestión de la conexión serie y envío y recepción de caracteres
 - `Serial::Serial(void);`
 - `Serial::Serial(const char * type);`
 - `serial::~~Serial(void);`
 - `int serial::connect();`
 - `int serial::send(char *data, int size);`
 - `int Serial::receive(char *buffer, size_t size);`
 - `void Serial::disconnect();`
 - `void Serial::setVerboseMode(int val);`

Instruction.cpp e Instruction.h

- Estructura de las instrucciones que se envían

```
Instruction & Instruction::operator=( const
Instruction &aux )
{
    if (instruction != nullptr) free(instruction);
    instruction = new char[aux.length-1];
    for (int i = 0; i < aux.length ; i++)
    {
        instruction[i] = aux.instruction[i];
    }
    length = aux.length;
    response = aux.response;
    return *this;
}
```

IRobotConnection.cpp e IRobotConnection.h

Construye cada una de las instrucciones que se envían. P. ej.:

- ```
void IRobotConnection::drive(int speed,
int radius)
{
 Instruction aux =
iRobotInstructionGenerator.drive(speed,radius
);
 connection.send(aux.instruction,
aux.length);
}
```
- ```
int IRobotConnection::updateSensor( char
sensorId )
{
    Instruction aux =

iRobotInstructionGenerator.updateSensor(sens
orId);
    connection.send(aux.instruction,
aux.length);

    connection.receive(buffer,aux.response);
    if (aux.response == 2)
return(createInt(buffer));
    return buffer[0];
}
```

Si queremos crear una nueva instrucción debemos añadir, por ejemplo:

- En Irobot Connection.h
 - Void NuevaInstruccion (type parameters)
- En Irobot Connection.h
 - ```
void IRobotConnection::NuevaInstruccion()
{
 Instruction aux =
iRobotInstructionGenerator.NuevaInstruccion();
 connection.send(aux.instruction, aux.length);
 connection.receive(buffer,aux.response);
}
```

# Librería iRobotConnection

## Modos:

```
void connect();
void start(); // código 128
void control(); // código 130 Ya no disponible en iRobot Create
void safe(); // código 131 No recomendable
void full(); // código 132
```

## Navegación:

```
void drive(int speed, int radius);
void driveDirect(int rightVelocity, int leftVelocity);
```

## Sensores:

```
int updateSensor(char sensorId);
void stream(char* sensorIdList, int size); No lo usamos
int queryList(char* sensorIdList, int size); No lo usamos
void PauseResumeStream(bool bolol); No lo usamos
```

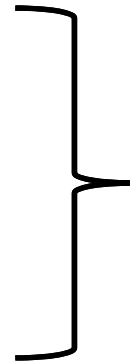
# Librería iRobotConnection

Scripting ~~No lo usamos~~

~~void script(int \*commandList, int size);~~  
~~void playScript();~~  
~~void showScript();~~

~~Espera eventos~~

~~void waitTime(int seconds);~~  
~~void waitDistance(int mm);~~  
~~void waitAngle(int degrees);~~  
~~void waitEvent(int eventId);~~



Es desaconsejable usarlos porque el programa en C no tiene mecanismos para esperar hasta que el evento se cumple

Otros:

void leds(int ledBit, int ledColor, int ledIntensity);  
void song (int songNumber, int songSize, char \*song);  
void playSong(int songNumber);  
...

# IRobotInstructionSet.cpp

Define cada una de las instrucciones que la maquina virtual es capaz de traducir

```
Instruction
 iRobotInstructionSet::updateSensor(
 char sensorId)
{
 int miSize = 2;
 Instruction aux;
 aux.instruction = (
 char*)malloc(miSize * sizeof(
 char));
 aux.instruction[0] =
 iRobotInstructions::SENSORS;
 aux.instruction[1] = sensorId;
 aux.response =
 sensorReturn(sensorId);
 aux.length = miSize;
 return aux;
}
```

Para crear una nueva intrucción:

```
Instruction
 iRobotInstructionSet::Nueva
 Instruccion (type param1, type
 param2,...)
{
 int miSize = numero_bytes;
 Instruction aux;
 aux.instruction = (
 char*)malloc(miSize * sizeof(
 char));
 aux.instruction[0] =
 iRobotInstructions::CODIGO_iROBOT
 ;
 aux.instruction[1] = Param1;
 aux.instruction[2] = Param2;
 ...
 aux.response =
 funcion_respuesta);
 aux.length = miSize;
 return aux;
}
```

# IRobotInstructionSet.h

Contiene las tablas de traducción

Códigos de los sensores. P. ej.:

- ...
- `static const char`  
  `BUMPERS_AND_WHEELDROPS = (char) 7;`
- `static const char WALL = (char) 8;`
- ...

Códigos de operación de iRobot. P. ej.:

- ...
- `static const char SENSORS = (char) 142;`
- `static const char DRIVE_DIRECT = (char)`  
  `145;`
- ...

Códigos de los eventos. P. ej.:

- ...
- `static const int RIGHT_BUMP = 7;`
- `static const int WALL = 9;`
- `static const int LEFT_CLIFF = 11;`
- ...

Número de bytes devueltos por cada sensor

- `int sensorReturn( char sensor);`  
  ...  
  `Instruction updateSensor(char`  
  `sensorId);`  
  ...  
  `Instruction driveDirect(int`  
  `rightVelocity, int leftVelocity);`  
  ...

# Uso al consultar sensores

```
Int value = updateSensor(char code);
```

Ejemplo:

```
Char value = updateSensor(iRobotSensors::BUMPERS_AND_WHEELDROPS);
printf("BUMPERS AND WHEELDROPS %s \n",value);
```

Codes:

- iRobotSensors::CLIFFLEFT
- iRobotSensors::CLIFFFRONTLEFT
- iRobotSensors::CLIFFFRONTRIGHT
- iRobotSensors::CLIFFRIGHT
- iRobotSensors::BUMPERS\_AND\_WHEELDROPS
- iRobotSensors::WALL
- iRobotSensors::DISTANCE
- iRobotSensors::ANGLE
- ...

# Codificaciones Sensores iCreate

Desde `updateSensor(char code)`; siempre obtenemos un `Int` (con signo), en cada caso habrá que hacer un casting al tipo de datos adecuado para poder tratar la información de los sensores correctamente.

Ejemplo:

```
Char value = updateSensor(iRobotSensors::BUMPERS_AND_WHEELDROPS);
printf("Front Caster WheelDrop: %d \n",value & 0x10);
```

| Sensor                            | Codificación                         |
|-----------------------------------|--------------------------------------|
| Bumps & WheelDrops                | Máscara de bits                      |
| Wall, Cliffs x 4, Virtual Wall    | 1 bit value (El menos significativo) |
| Infrared                          | 1 Byte [0,255]                       |
| Distance, Angle, Requested Radius | Signed 16 bit value [-32768 - 32768] |
| <b>Wall Signal, Cliffs x 4,</b>   | <b>Unsigned 16 bit [0 – 4095]</b>    |
| Requested Velocity x3             | Signed 16 bit value [-500, 500]      |



## 2. Instalación y programación del iRobot\_Framework

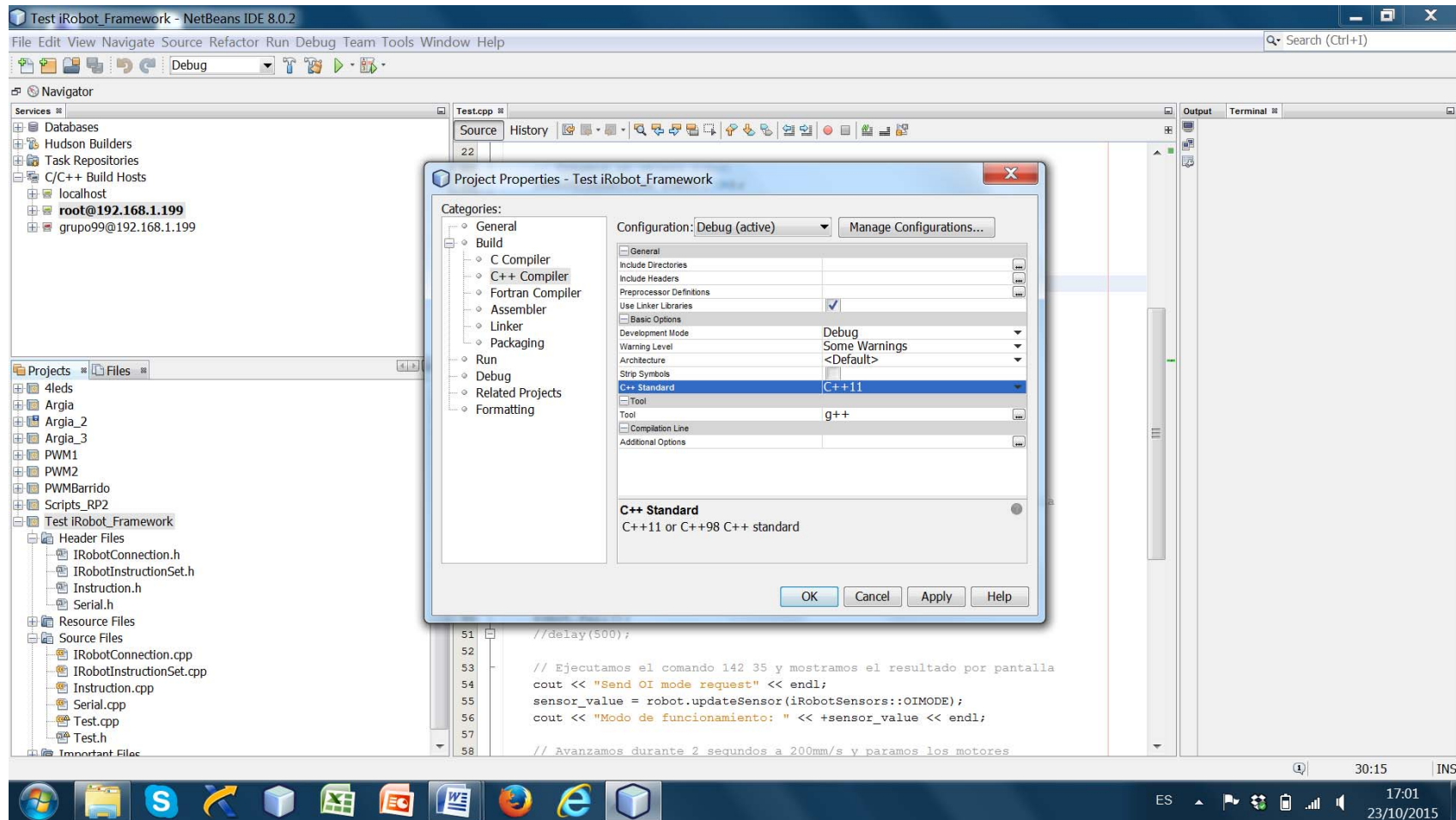
# Instalación de los compiladores y linkeditores en RaspberryPi

- Instalar el compilador de Raspberry Pi2, versión 4.8
  - > sudo apt-get install gcc-4.8
  - > sudo apt-get install g++-4.8
- Borrar los enlaces simbólicos del sistema antiguos
  - > sudo rm /usr/bin/gcc
  - > sudo rm /usr/bin/g++
- Actualizar los enlaces simbólicos del sistema: /usr/bin/gcc y /usr/bin/g++
  - > sudo ln -s /usr/bin/gcc-4.8 /usr/bin/gcc
  - > sudo ln -s /usr/bin/g++-4.8 /usr/bin/g++

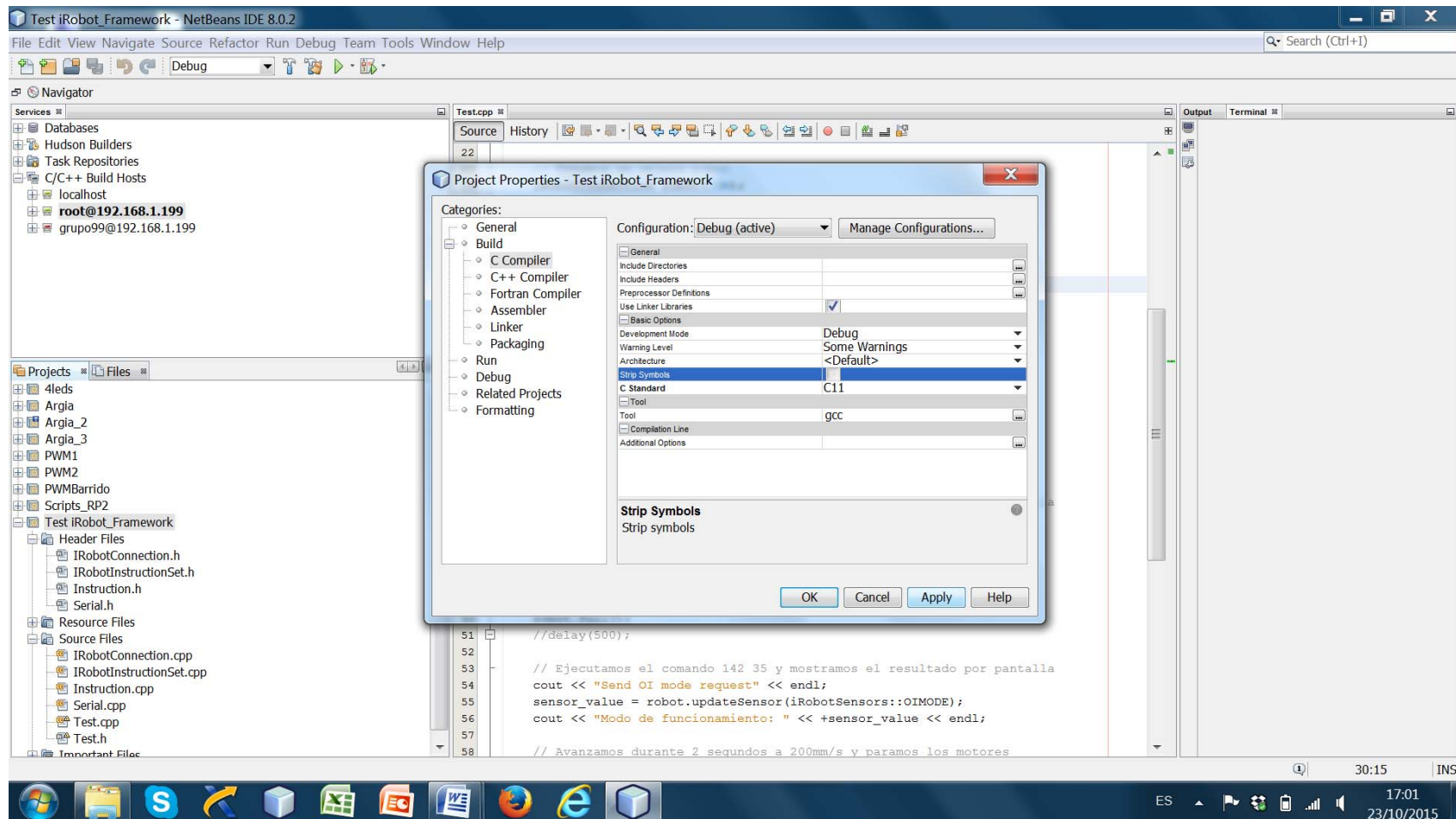
# Proceso de instalación

- Abrir NetBeans
- Importar proyecto desde zip
  - NetBeans:  
File > Import Project > From Zip
  - Importar el proyecto Test\_iRobot\_Framework802.zip
- Cambiar el build-host al que corresponda a cada grupo:  
(XXXX@192.168.1.XXX)
- El SRC Contiene:
  - IRobotConnection.h
  - IrobotInstructionSet.h
  - Instruction.h
  - Serial.h
  - Test.h
  - IRobotConnection.cpp
  - IrobotInstructionSet.cpp
  - Instruction.cpp
  - Serial.cpp
  - Test.cpp

# En Project properties>linker>libraries añadir -lwiringPi

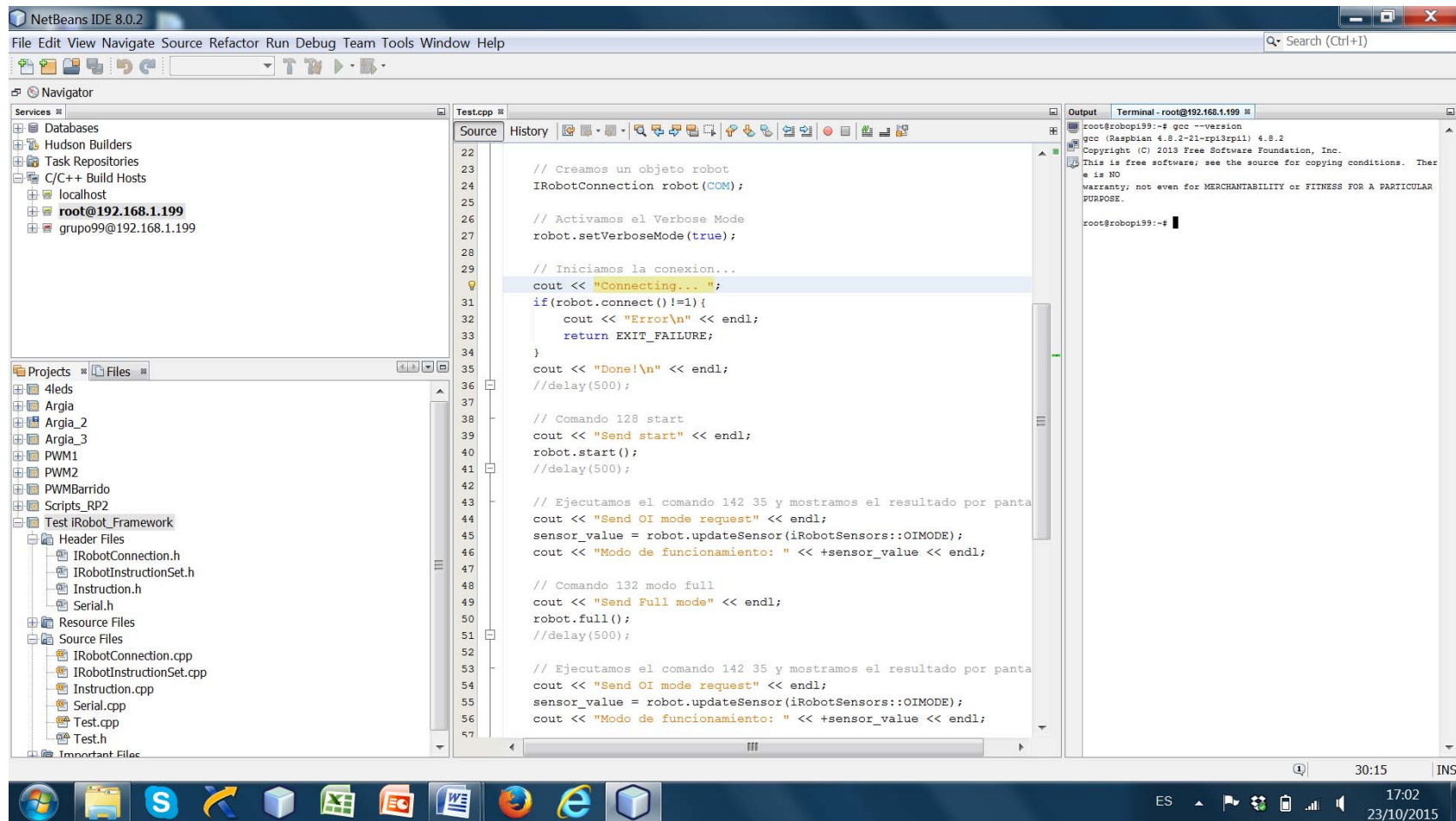


En Project properties > C compiler seleccionar C11 y en > C++ Compiler seleccionar C++11

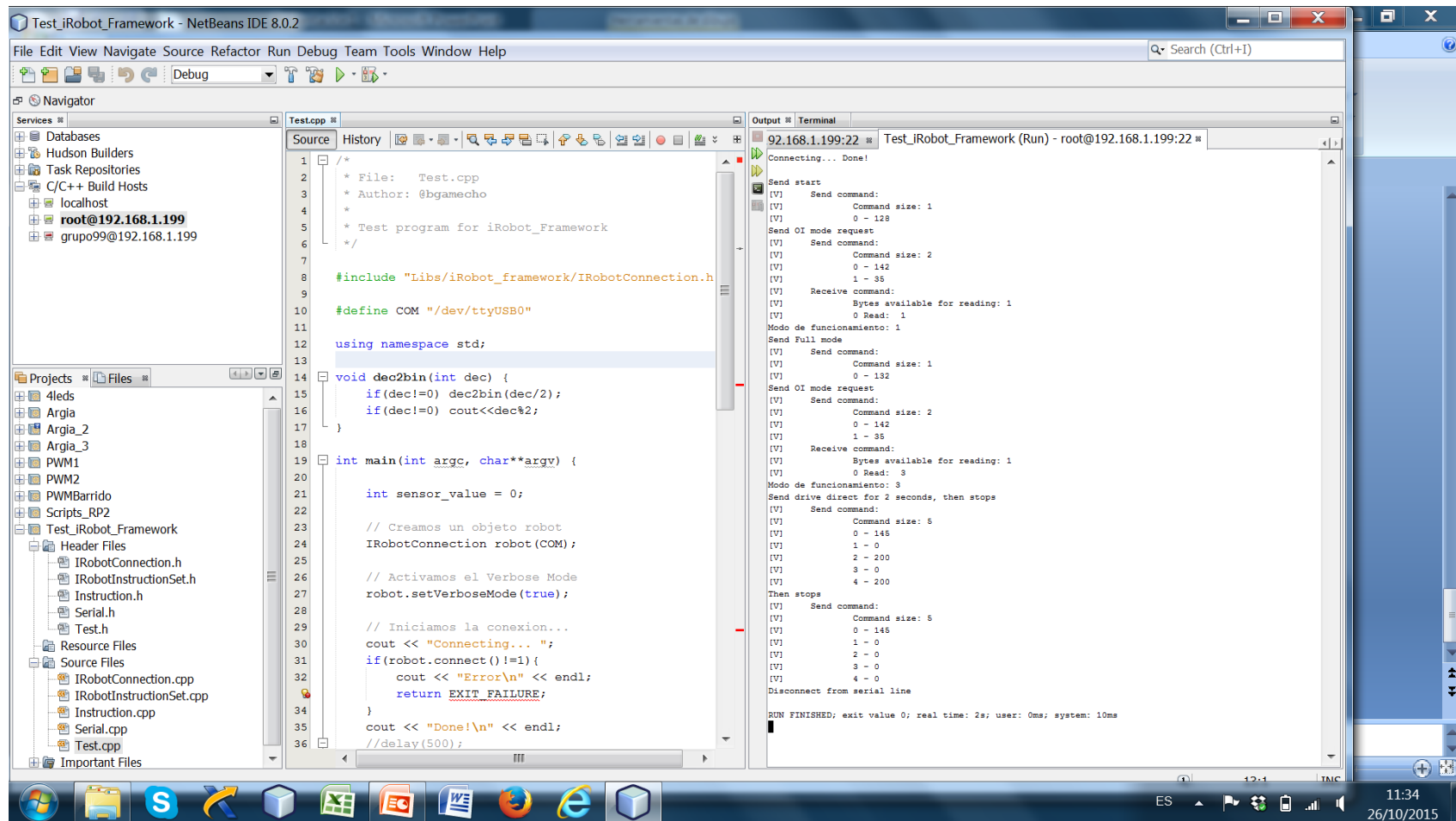


- Ejecutar en el terminal remoto de Raspberry Pi2 (para dar al usuario GrupoXY permiso de ejecución en el adaptador línea-serie conectado al iRobot Create):  
> `sudo usermod -aG dialout grupoXY`
- Host de compilación
  - Asignar host: vuestra RaspberryPi:  
`root@192.168.1.1XX`
  - Verificar que está compilando para vuestra dirección IP (`root@192.168.1.1XX` o `grupoXX@192.168.1.1XX`)-

# Compilar y ejecutar Test.cpp



# La ejecución producirá



Test\_iRobot\_Framework - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Team Tools Window Help

Search (Ctrl+I)

Services

- Databases
- Hudson Builders
- Task Repositories
- C/C++ Build Hosts
- localhost
- root@192.168.1.199
- grupo99@192.168.1.199

Projects

- 4leds
- Argia
- Argia\_2
- Argia\_3
- PWM1
- PWM2
- PWMBarrido
- Scripts\_RP2
- Test\_iRobot\_Framework
  - Header Files
    - IRobotConnection.h
    - IRobotInstructionSet.h
    - Instruction.h
    - Serial.h
    - Test.h
  - Resource Files
  - Source Files
    - IRobotConnection.cpp
    - IRobotInstructionSet.cpp
    - Instruction.cpp
    - Serial.cpp
    - Test.cpp
- Important Files

Test.cpp

```
1 /*
2 * File: Test.cpp
3 * Author: @bgamecho
4 *
5 * Test program for iRobot_Framework
6 */
7
8 #include "Libs/iRobot_framework/IRobotConnection.h"
9
10 #define COM "/dev/ttyUSB0"
11
12 using namespace std;
13
14 void dec2bin(int dec) {
15 if(dec!=0) dec2bin(dec/2);
16 if(dec!=0) cout<<dec%2;
17 }
18
19 int main(int argc, char**argv) {
20
21 int sensor_value = 0;
22
23 // Creamos un objeto robot
24 IRobotConnection robot(COM);
25
26 // Activamos el Verbose Mode
27 robot.setVerboseMode(true);
28
29 // Iniciamos la conexion...
30 cout << "Connecting..." << endl;
31 if(robot.connect() !=1){
32 cout << "Error\n" << endl;
33 return EXIT_FAILURE;
34 }
35 cout << "Done!\n" << endl;
36 //delay(500);
```

Output

92.168.1.199:22 Test\_iRobot\_Framework (Run) - root@192.168.1.199:22

Connecting... Done!

Send start

[V] Send command:

[V] Command size: 1

[V] 0 - 128

Send OI mode request

[V] Send command:

[V] Command size: 2

[V] 0 - 142

[V] 1 - 35

[V] Receive command:

[V] Bytes available for reading: 1

[V] 0 Read: 1

Modo de funcionamiento: 1

Send Full mode

[V] Send command:

[V] Command size: 1

[V] 0 - 132

Send OI mode request

[V] Send command:

[V] Command size: 2

[V] 0 - 142

[V] 1 - 35

[V] Receive command:

[V] Bytes available for reading: 1

[V] 0 Read: 3

Modo de funcionamiento: 3

Send drive direct for 2 seconds, then stops

[V] Send command:

[V] Command size: 5

[V] 0 - 145

[V] 1 - 0

[V] 2 - 200

[V] 3 - 0

[V] 4 - 200

Then stops

[V] Send command:

[V] Command size: 5

[V] 0 - 145

[V] 1 - 0

[V] 2 - 0

[V] 3 - 0

[V] 4 - 0

Disconnect from serial line

RUN FINISHED; exit value 0; real time: 2s; user: 0ms; system: 10ms

11:34 26/10/2015



- A partir de este momento ya se pueden escribir programas en C/C++ usando las instrucciones de iRobot proporcionadas por iRobot\_Framework