# iRobot_Framework

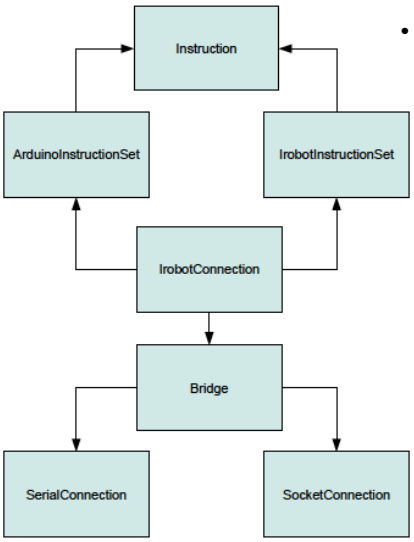
Robótica, Sensores y Actuadores

2017-2018

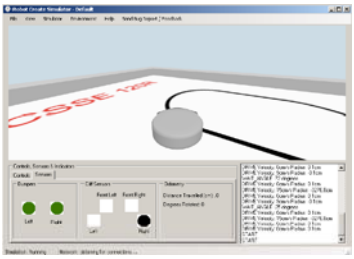
# iRobot_Framework

- iRobot_Framework es una máquina virtual que permite programar el iRobot Create en C/C++
- Aporta un lenguaje de programación de alto nivel (LPAN) para programar el iRobot Create.
  - Actúa como una máquina virtual que traduce las instrucciones de este lenguaje a los códigos que ejecuta el iRobot
  - Además se encarga de conectar por linea serie el iRobot y el procesador (en nuestro caso Raspberry Pi 2)
- Versiones
  - inicial: pPara Arduino, simulador, y BAM. Diseñada por Gorka Montero PFC (2012).
  - Actual: para Raspberry Pi2. Extendida por Borja Gamecho (2015).

# Versión inicial de las Librerías IRobotConnection

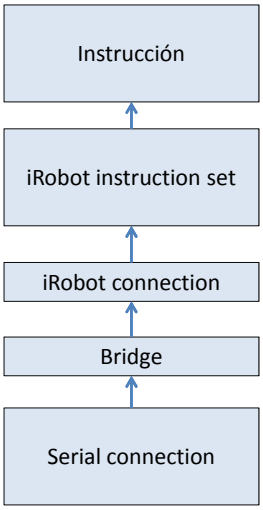



- Usa de forma “transparente” el mismo código en 3 escenarios:
  - Simulador (de Rose-Hulman Inst.)



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

# Versión actual de las Librerías IRobotConnection




- Genera código que se ejecuta sobre Raspberry Pi2.
- Este código traduce cada instrucción de alto nivel a comandos del iRobot.
- Abre una línea serie (entre Raspberry Pi 2 e iRobot Create ) y envía por ella los códigos asociados a cada instrucción
- Requiere una conexión física por línea serie entre Raspberry Pi 2 e iRobot Create

# Programación en C/C++

```cpp
#include <iostream>
#include "../libs/IRobotConnection.h"
#include <dos.h>

using namespace std;

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

    // Creamos un objeto robot que se conectará por el puerto x
    IRobotConnection robot("COMx");

    // Iniciamos la conexión
    printf("Connecting... ");
    robot.connect();
    printf("Done!!\n");

    // comando 128 start
    robot.start();
    Sleep(500);

    // comando 132 modo full
    robot.full();
    Sleep(500);

    ... resto de comandos ...

    robot.disconnect();
    return 0;
}
```

```cpp
// 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 2segundos a 200mm/s y paramos los motores

robot.driveDirect(200,200);
Sleep(2000);
robot.driveDirect(0,0);
```

# 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);~~
- ~~int queryList(char * sensorIdList, int size);~~
- ~~void PauseResumeStream(bool bolol);~~

# Librería iRobotConnection

~~Scripting~~
- ~~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);~~

Desaconsejable usarlos, paran el envío de datos por la línea serie 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);
- …

# 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 Radious | Signed 16 bit value [-32768 - 32768] |
| **Wall Signal, Cliffs x 4,** | **Unsigned 16 bit [0 – 4095]** |
| Requested Velocity x3 | Signed 16 bit value [-500, 500] |

# Instalación y uso del iRobot_Framework

# Puesta en marcha

- 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++

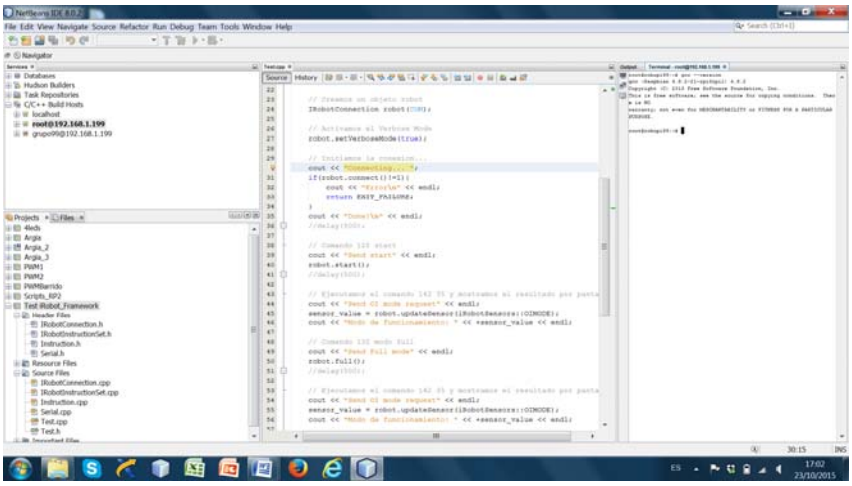
- Actualizar los enlaces simbólicos del sistema: /usr/bin/gcc y /usr/bin/g++
  > sudo ln –s /usr/bin/gcc-4.8 /usr/bin/gcc
  > sudo ln –s /usr/bin/g++-4.8 /usr/bin/g++

# Proceso de instalación

- Abrir NetBeans
- Importar el proyecto frameworktest.zip
  – seleccionar File—> Import Project —> From Zip
- frameworktest.zip está en eGela: Práctica 3
  – Contiene los ficheros:

- IRobotConnection.h
- IrobotInstructionSet.h
- Instruction.h
- Serial.h
- Test.h
- IRobotConnection.cpp
- IrobotInstructionSet.cpp
- Instruction.cpp
- Serial.cpp
- Test.cpp

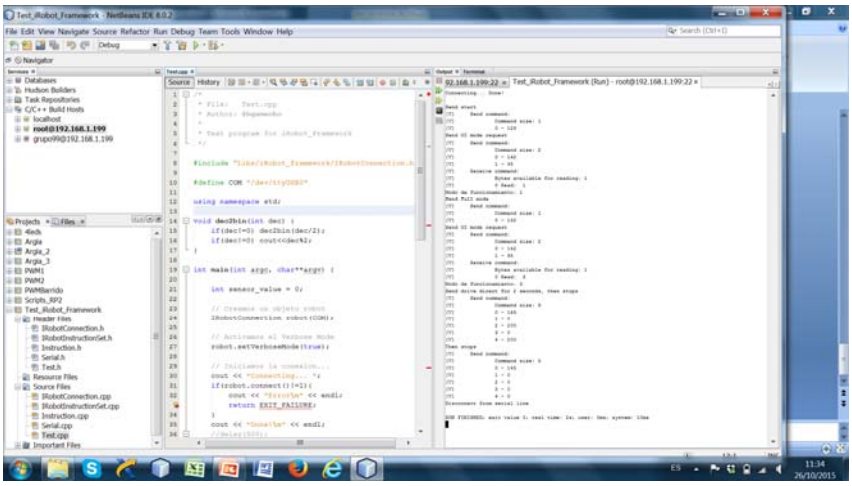
# Compilar y ejecutar Test.cpp



- Si se trabaja con la cuenta root
  - Verificar que está compilando para vuestra dirección IP (root@192.168.1.1XX)
- Si se trabaja con la cuenta GrupoXX
  - Verificar que está compilando para vuestra dirección IP (grupoXX@192.168.1.1XX)
  - Para dar al usuario GrupoXY permiso de ejecución en el adaptador línea-serie conectado al iRobot Create, ejecutar en el terminal remoto de Raspberry Pi2 :

    > sudo usermod -aG dialout grupoXY

# La ejecución producirá



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