Práctica 2 Introducción al iRobot Create

Programación del iRobot Create mediante comandos y scripts

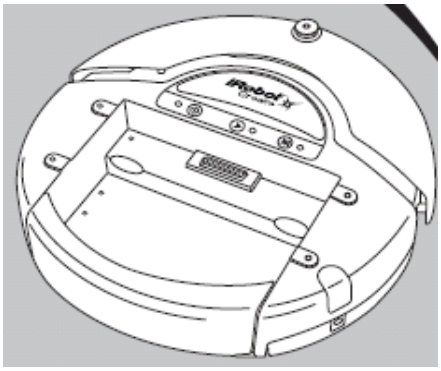
OBJETIVOS

- Conocer la arquitectura y la programación a bajo nivel del iRobot Create.
- Entender la conexión de la Raspberry Pi2 con el iRobot (alimentación y línea serie).
- Programar el iRobot mediante comandos numéricos y scripts.
- Enviar comandos al iRobot desde la Raspberry Pi a través de NetBeans.

MATERIAL NECESARIO

Hardware

- iRobot Create
- Cable serie PC-iRobot
- Cables de alimentación y conexión serie Raspberry Pi2 <---> IRobot



Software

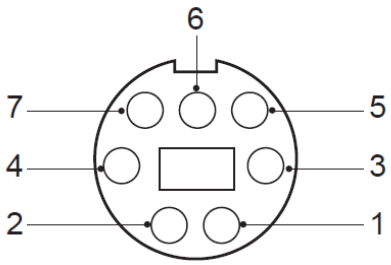
- Realterm
- Free Hex Editor Neo

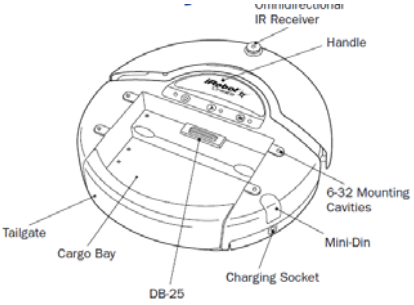
Manuales (en eGela)

- Guía del usuario: iRobot Create Owner's Guide
- Manual del lenguaje de comandos: iRobot Create Open Interface.

1. Conexión del iRobot al PC mediante el cable serie, en Windows 7

a. Conectar el **cable serie iRobot**[1] al conector Mini-DIN del puerto serie del iRobot Create y por el otro lado al cable Serie-USB. Y conectar este último a una entrada USB del PC. Encender iRobot Create.





b. Desde Windows: Inicio → "dispositivos e impresoras":

[1] El cable serie iRobot transforma el voltaje lógico de RXD y TXD (0v-5v) al voltaje requerido por el puerto serie RS232C del PC.

c. Aparecerá un dispositivo de nombre: “Prolific USB-to_Serial Comm Port” y **el nombre de puerto** entre paréntesis que usaremos para conectarnos a través de RealTerm.

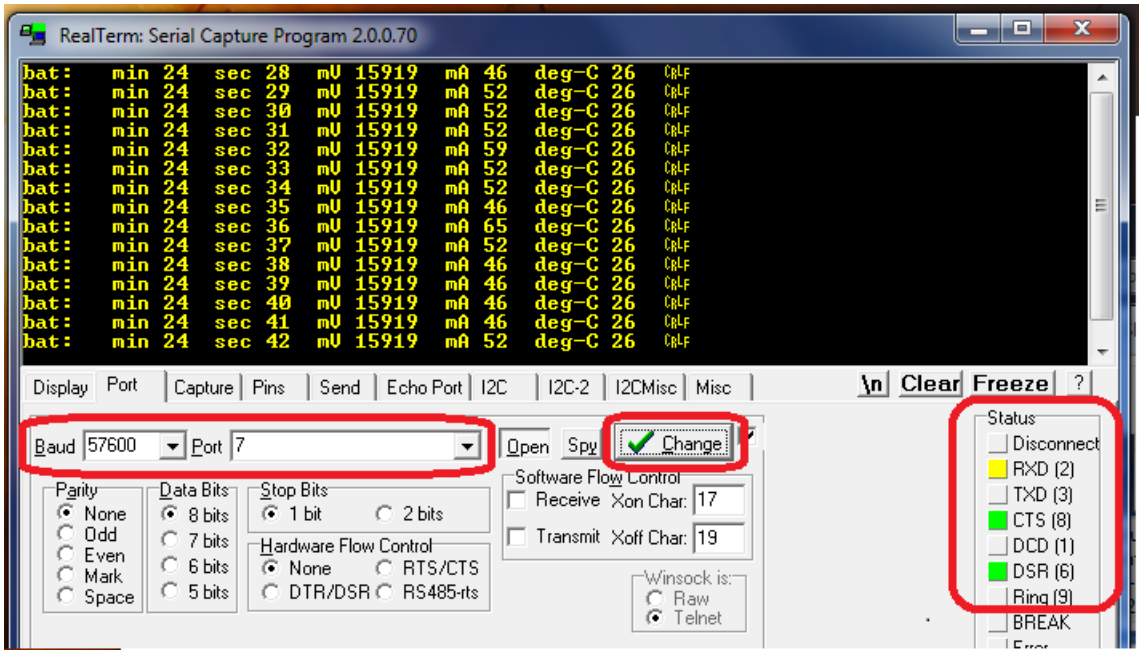
2. Programación con comandos. Configuración y uso de RealTerm

Abrid el programa RealTerm[2].

Si devuelve el error “Exception EOleSysError in Module realterm.exe” ejecutarlo como administrador.

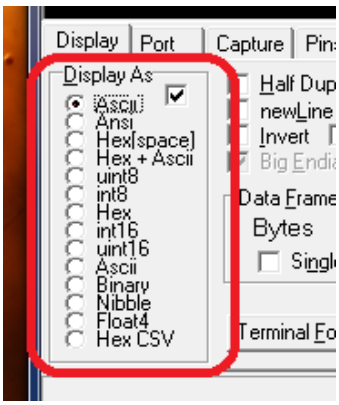
Configurad RealTerm para comunicarse con el iRobot por la línea serie, usando la pestaña **PORT**. Seleccionar el puerto COM en el que está enlazado el iRobot en Windows y cambiar la velocidad a 57600 Baudios.

Para aplicar los cambios hay que pulsar en el botón “Change”.



Visualización de los datos de iRobot desde RealTerm

En la pestaña **DISPLAY** están los modos de visualización de los datos que recibimos por línea serie.



Es importante tener en cuenta que solo vamos a poder apreciar correctamente cambios en los sensores si elegimos el modo “Binary”, “int16” o “uint16” en función de la codificación del sensor (no usar ASCII).

Tener en cuenta cómo se codifican los parámetros que se envían con dada comando, y los datos que devuelven los sensores (Ver Tabla 1, Create_Open_Interface_v2.pdf y Codificación de comandos de iRobot Create.pdf)

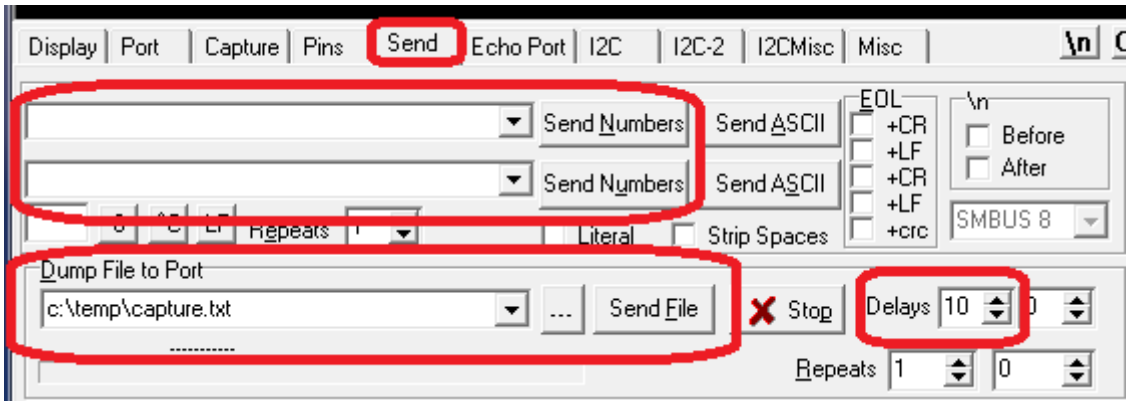
[2] Realterm es un terminal virtual diseñado para capturar, controlar y depurar flujos de datos binarios y depurar comunicaciones.

Tabla 1. Codificación OI Sensores iCreate

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]

Envío de Comandos

En la pestaña **SEND** debemos usar el botón "Send Numbers" para mandar los códigos de los comandos del robot. En el cuadro de opciones "Dump File to Port" podemos cargar un Script y enviárselo configurando un "Delay" de 10 ms entre código y código. Los archivos que se envían mediante la opción "Dump File to Port" de RealTerm solo deben contener los códigos OI de iRobot y estar editados desde un editor hexadecimal.



Ejemplo de comandos

Probad los siguientes comandos:

- Activar el LED de PLAY: 128 132 139 2 0 0
- Leer el estado del sensor de desnivel de la izquierda: 128 142 9 (observar cómo cambia la respuesta cuando se levanta el robot)
- Para hacer música con iRobot: enviar estos conjuntos de comandos por separado:
  - 128 132 (Pone el robot en el modo completo (full))
  - 140 0 4 62 12 66 12 69 12 74 36 (define la canción)
  - 141 0 (toca la canción)

Poned el robot sobre la caja de plástico para que no se mueva al poner en marcha los motores y enviar los siguientes comandos:

- Para hacer avanzar hacia adelante al iRobot: 128 131 137 0 100 128 0
- Para detener al iRobot: 128 131 137 0 0 0 0

Probar los comandos del documento iRobot Create Open Interface. En especial los que activan los motores: Drive y Drive Direct.

3. Programación con scripts. Hex Editor Neo

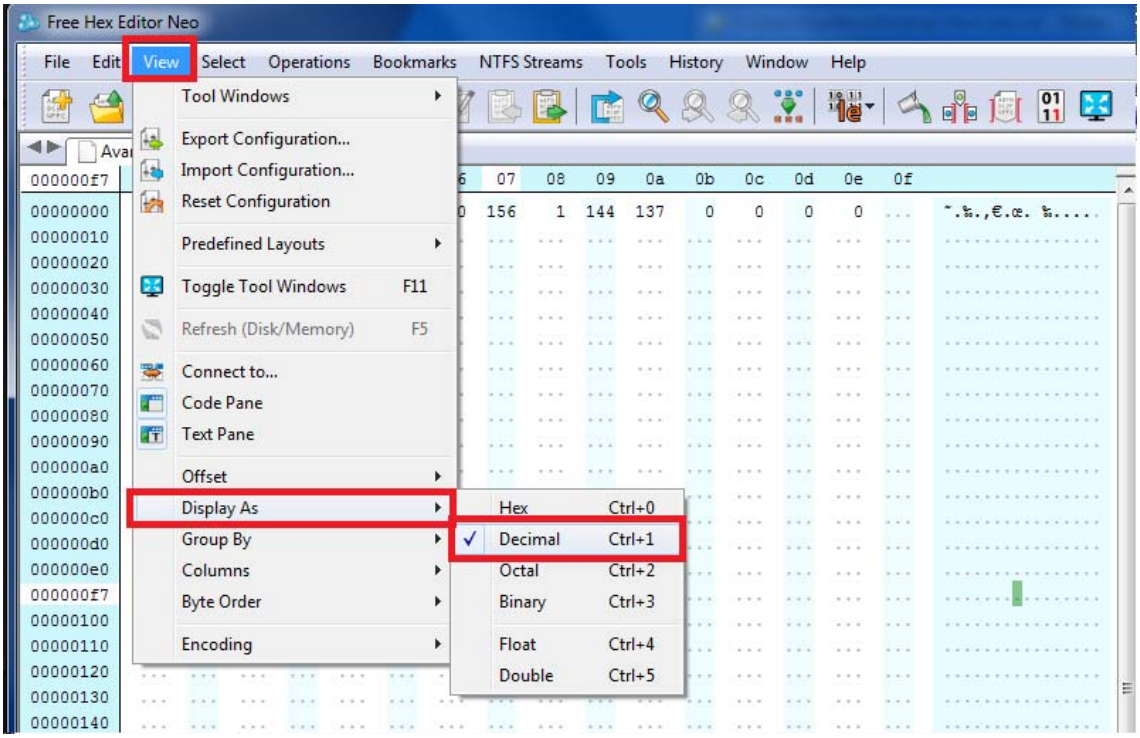
Un script es una serie de comandos que se ejecutan secuencialmente Se pueden hacer programas complejos agrupando los comandos en scripts.

- Un script consta de comandos OI y puede tener hasta 100 bytes de longitud.
- El primer código debe ser 153 y el segundo, el número de bytes que siguen dentro del script (entre 0 y 100). Se usa la longitud de 0 para borrar la secuencia de comandos actual. El resto de bytes son comandos del Open Interface y bytes de datos.
- La secuencia debe ser: [152] [Longitud de secuencia de comandos] [Opcode 1] [Opcode 2] [Opcode 3] etc.

El comando 152 sólo graba el script en memoria, pero no lo ejecuta. Para que se ejecute hay que enviar un 153 después del script. Si se mete el 153 en el propio script, este ciclará infinitamente (siempre que se meta después otro 153 para que se ejecute el script). Observad que el primer script se ejecuta una vez y el segundo cicla:

- 152 **12** 145 0 80 0 80 155 20 145 0 00 0 00 153
- 152 **13** 145 0 80 0 80 155 20 145 0 00 0 00 153

Los scripts contenidos en ficheros de texto se envían al iRobot mediante "Dump File to Port". Es necesario que los scripts sean binario puro, sin caracteres de control. Como la mayoría de editores (ej. NotePad) incluyen caracteres de control, usamos el editor hexadecimal "**Free Hex Editor Neo**" que dispone de la posibilidad de codificar los caracteres mediante un valor entero. Esto permite usar directamente la numeración de los comandos OI del manual de iRobot. Dentro del programa, se puede cambiar la codificación entre hexadecimal y decimal usando la combinación Ctrl+1 o el menú que aparece en la siguiente imagen:



Editar con Free Hex Editor Neo los siguiente scripts que aparecen en la página 15 del manual iRobot Create Open Interface y ejecutarlos:

- Drive 40cm and stop
- Toggle led and bump

- Drive in a square

Diseñar un script que permita leer los códigos IR enviados por la base de carga o el mando a distancia.

- Cuando funcione, enseñádselo al profesor
- Añadid el listado en el informe de la práctica, con comentarios.

4. Conexión del iRobot al PC mediante Wi-Fi a través de la Raspberry Pi

a. Conectar los cables de alimentación y línea serie entre Raspberry Pi 2 e iRobot.
b. Encender el iRobot y comprobar que se la Raspberry Pi 2 se enciende.
c. Abrir un nuevo proyecto en NetBeans 8.0.2 y copiar en el programa principal (por ejemplo welcome.cc) que está en "S*ource Files"* el Enviarls.cc que está en el Anexo 2 de este documento.
d. Crear un archivo de texto "myScript.txt" (que contendrá el script que se quiera enviar) y guardarlo en la carpeta "*Resource Files*" de NetBeans.

5. Programación con scripts

El programa **enviarls.cc** (en el Anexo II) lee los enteros contenidos en el fichero **myscript.txt** y se los envía como caracteres por la línea serie al iRobot. Se detiene cuando lee **-1**.
De esta manera, para enviar comandos o scripts al iRobot basta con meterlos en el fichero myscripts.txt terminando en -1 y ejecutar enviarls.cc.

Precauciones:

- En el proyecto de NetBeans hay que cambiar las opciones de Run (en Propiedades del Proyecto) de tal manera que en lugar de ejecutarse con *sudo "${OUTPUT_PATH}" "arg1" "arg2" "arg3" "arg4"*, se ejecute con: sudo "${OUTPUT_PATH}" "myScript.txt"
- Recordad que para que ejecute cualquier comando el robot debe estar en modo full: 128 132
- Recordad que todo script debe empezar por 152 y que para que se ejecute hay que enviar 153 después del script. Por ejemplo: 152 14 145 0 80 0 80 155 20 128 132 145 0 00 0 00 153 -1

Probar que funciona incluyendo en el fichero myscript.txt los siguientes scripts que aparecen en la página 15 del manual iRobot Create Open Interface:

- Drive 40 cm and stop
- Toggle led and bump
- Drive in a square

6. Desarrollo de un script propio

Diseñar un script que permita comparar los comandos Drive y Drive Direct. Por ejemplo, haciendo el mismo recorrido con uno y otro comando, y observando, midiendo y comentando los resultados.

- Puntúa la complejidad del script, la variedad de sensores y comandos usados y la creatividad.
- Cuando funcione, enseñádselo al profesor
- Añadid el listado en el informe de la práctica, con comentarios.

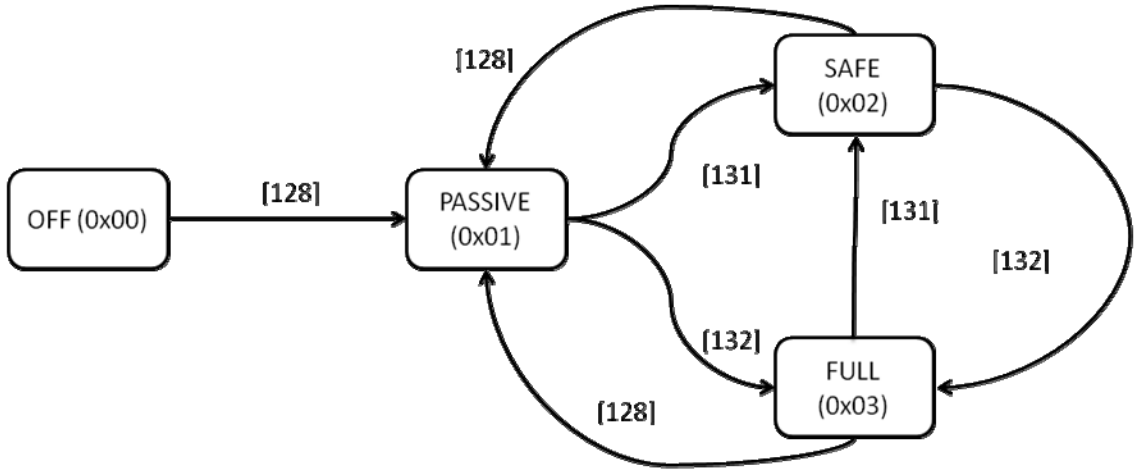
7. Informe a presentar

- Listar todos los comandos ejecutados y describir el comportamiento del robot para cada uno de ellos.
- Comentar y describir en detalle los scripts diseñados por el grupo.
- Responder a las siguientes preguntas:
  1. Comunicación iRobot-PC
     a. ¿Para qué se usa RealTerm en la programación de comandos para iRobot?
     b. ¿Qué hardware de comunicación entre iRobot y el PC se utiliza con RealTerm?
  2. Comandos de iRobot
     a. ¿Qué formato tiene un comando de iRobot?
     b. ¿Qué formato tiene los parámetros de un comando de iRobot? Poned un ejemplo de la transformación necesaria.
  3. Scripts
     a. ¿Para qué se usan los scripts?
     b. ¿Para qué sirve el 153 al principio de un script?
     c. ¿Para qué sirve el -1 al final de un script?
  4. Enviarls.cc
     a. ¿Para qué usamos el programa Enviarls.cc y por qué?
     b. Comentar el programa Enviarls.cc

Anexo I: iRobot Create Scripting

Modos de funcionamiento (OI Modes)

Modo	Código	Restricciones
Passive	[128] (0x80)	No funcionan: Motores, Sonido, Leds, Low Side Drivers, Digital Outputs.
Safe	[131] (0x83)	Control total excepto bajo condiciones de seguridad: Cliffs while moving, Wheel Drops & Charger plugged in and powered.
Full	[132] (0x84)	Ninguna



Consultar Modo: [142], [35]

Actuadores: Drive vs. Drive-Direct

Existen dos formas distintas de mover el robot:

- Drive mueve el robot a una velocidad y con un radio de giro.
- Drive-Direct controla cada rueda del robot de forma independiente.

Modo	Comandos	Rango
Drive	[137] Velocity 16bits Radius 16 bits	[-500 – 500 mm/s] [-2000 – 2000 mm]
Drive Direct	[145] V. Rueda Der. 16 bits V. Rueda Izq. 16 bits	[-500 – 500 mm/s] [-500 – 500 mm/s]

- Los parámetros de velocidad y Radio están codificados en “Complemento a 2”:
- Ejemplo “Velocidad máxima”:
  - +500 : 0x01 0xF4 || [1] [244] || 0000 0001 1111 0100
  - -500 : 0xFE 0x0C || [254] [11] || 1111 1110 0000 1100
- Valores por debajo de 30/-30 mm/s pueden no ser suficientes para mover el robot

Scripting + Eventos

- Scripts [152 – 153 - 154]:
  - Specify script [152] + bytes_script + commands*
  - Play Script [153]
  - Show Script [154]
- Eventos - Espera [155 – 156 – 157 – 158]
  - Durante la espera, el robot no cambia de estado, ni responde a los inputs, o línea serie. **Usar solo para Scripts**
  - Por Tiempo [155]: 1 Byte: 0 – 255 decenas de segundo
  - Por Distancia [156]: 2 Bytes: -32767 – 32767 mm
  - Por Ángulo [157]: 2 Bytes: -32767 – 32767 grados
  - Por Evento [158]: Código de evento

Sensores

- Comando para leer sensores [142] [packet id]
  - (página 13, 17-25 del manual Open Interface)
  - Ejemplo: [142] [7] para compobar Bumps and WheelDrops (registro de 8 bits)

Bumps and Wheel Drops

Bit	7	6	5	4	3	2	1	0
Sensor	n/a	n/a	n/a	Wheel-drop Caster	Wheel-drop Left	Wheel-drop Right	Bump Left	Bump Right

- Streaming de sensores [148]

Codificaciones de los Sensores iCreate

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]

Anexo 2: Programa Enviarls.cc

```
/* Envía los scripts contenidos en myScript.txt al iRobot Create*/

#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringSerial.h>

#define SERIAL "/dev/ttyUSB0"

void send_script(char *myScript);
int main(int argc, char**argv) {
    std::cout << "Script: " << argv[1] << std::endl;
    if(argc==2){
        send_script(argv[1]);
    }else{
        std::cout << "Error no argument found" << std::endl;
    }
    return EXIT_SUCCESS;
}

void send_script(char *myScript){
    int value, fd_serial, num_bytes;
    FILE * fd_script;
    fd_script = fopen(myScript, "r");
    if(fd_script < 0 ){
        std::cout << "Error opening the script file" << std::endl;
    }
    fd_serial =0;
    fd_serial = serialOpen(SERIAL, 57600);
    if(fd_serial < 0 ){
        std::cout << "Error opening serial line" << std::endl;
    }
    serialPutchar(fd_serial,  128);
    delay(10);
    std::cout << "escribiendo 128" << std::endl;
    serialPutchar(fd_serial,  132);
    delay(10);
    std::cout << "escribiendo 132" << std::endl;
    value = 0;
    num_bytes = 0;
    while(value!= -1){
        fscanf( fd_script, "%d", &value );

        if(value != -1){
            num_bytes++;
            std::cout << std::dec << value  << " " << std::endl;

            serialPutchar(fd_serial,  value);
            delay(10);
        }
    }
    std::cout <<  "Script ends: " << num_bytes << " bytes read" <<
std::endl;

    serialClose(fd_serial);
return;
}
```