

Summer School on Rescue Robots

8th International UJI Robotics School

Practical Session with Lego Mindstorms

Tutorial

**Antonio Morales
Javier Felip
Juan Carlos García**

**Robotic Intelligence Lab
Universitat Jaume I**

1. Introduction.

This guide introduces the reader in the world of LEGO Mindstorms RCX robots. First, there is an overview of the hardware components of the robots, motors, sensors and CPU, and next there is a description of the basics of NQC programming language, how to control the motors, sensors and other useful tricks. Finally, there are some basic exercises to train the knowledge acquired through this tutorial.

2. Introducing the LEGO Mindstorms.

A LEGO Mindstorms robot is composed of three principal components: sensors, motors and the CPU also called Brick or RCX. Next we explain these components separately.

The RCX brick.

The computing core of a LEGO Mindstorms robot is the RCX brick. It contains a programmable microcontroller able to read up to three sensors inputs and to write on three output ports. The programs are loaded in the microcontroller from the PC using a infrared interface. Figure 1 shows a picture of LEGO Mindstorms RCX brick. Ports for sensors are numbered for 1 to 3, and ports for motors are named from A to C. It also has four buttons that can have different purposes depending on the operating system installed on the brick. With the default O.S. installed, button labelled **RUN** is used to start the execution of the program indicated on the LCD screen; **Prgn** is used to select a program among a set of up to 5 pre-loaded programs; **View** is used to show the value of the motor or sensor which number is showed in the screen; finally **On-Off** turns the brick on and off. Power supply is provided by six type AA (1.5 volts) batteries.



Figure 1: The Mindstrom RCX brick

ROM size	16K
RAM size	512
Speed	16MHz @ 5V
A/D Conversion	8 8-bit
Serial port	1

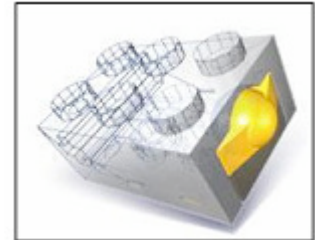
Figure 2: Characteristics of the CPU embedded in the brick.

Sensors

We will see in this section the standard types of sensors that can be mounted with the LEGO Mindstorms Kit, moreover we explain their most significant characteristics and most common uses.

Touch sensors:

These sensors return a boolean value. If the sensor value is 1, the sensor has been pressed, and when sensor value is 0, the sensor has not been pressed. Its most common use is the collision detection. It is possible to connect some sensors on the same port obtaining as a result the logical AND of all connected sensors.



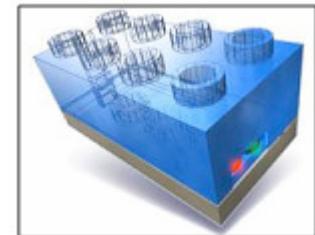
Rotation sensors:

This type of sensors returns a value that counts the number of turns. Often, this is connected to the motors to measure distances and to make odometry calculations. [Hurb] explains the detailed operation of these sensors.



Light sensors:

This sensor returns a value proportional to the amount of light it receives. It is the most used because of its capability to detect colour at a very short range with the ambient light. [Gasp] exposes internal details and [Ang] purposes a modification to improve the sensibility of the sensor.



The sensors are connected to the ports labelled 1 to 3 of the brick. Each port allows only one sensor, with the exception of the touch sensors that can be connected to the same port obtaining as a result the logical AND of the connected sensors. To avoid problems of value inversion, all the connections should be done with the cable pointing to the centre of the brick.

Motors

Several types of motors can be connected to RCX. [Hurb2] presents a complete comparative between them. The model of motors currently provided in the basic kit is 43336 (see figure 3), but is also possible that in old kits a motor of model 2838 is provided. Figure 3 shows the main types of motors.



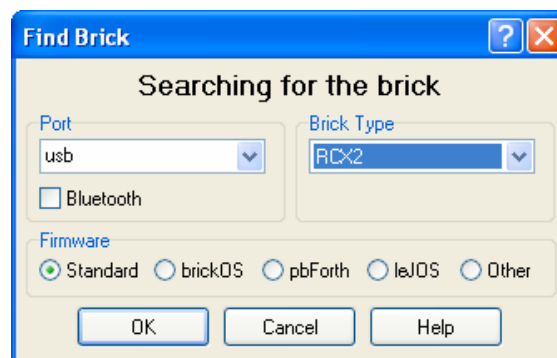
Figure 3 : Different models of Lego motors. From left to right: 2838, 43362, 2986, 5292, 47154

The operate them, the motors must be connected in the ports of RCX named A, B, and C. Though it should not be a problem it is recommended to perform the connexion in the same way it was explained for sensors. In this way, we avoid the inversion of polarity that would cause the motors to rotate in the opposite sense to that expected.

3. LEGO NQC Programs

NQC (Non Quite C) is a programming language for LEGO robots developed by Dave Baum. It is a simplified version of C. It uses the usual control structures, functions, subroutines and variables, moreover offers all necessary tools to control motors, sensors and all of the LEGO RCX capabilities.

To program, we will use the NQC Brick Command Center programming environment. Firstly, we should configure the environment to detect the robot, to do that, we have to put the values shown on the next figure.



It is not necessary to establish a connection with the robot before starting to program. The connection is only necessary to download the code to the Brick. To connect with the robot, turn on the NQC Center and use the FindBrick button:



Once connected with the robot other buttons of the toolbar will be enabled. To download a program to the Brick, we have to select one of the five program slots and then download the code to the robot by using the download button or by pressing F6 key, if there is any compiling mistake, it will be shown and we have to fix it before downloading the program successfully.



As said before, the language syntax is exactly equal to C syntax with some exceptions and limitations due to the limited hardware of the Brick. All details about NQC programming can be found in the NQC guide by Dave Baum and John Hansen:

http://bricxcc.sourceforge.net/nqc/doc/NQC_Guide.pdf

Next, we present the basic aspects of this programming language, motor control, sensor control and multi-task programming.

4. Starting to program

Before starting to program a robot, we must know its physical configuration. The configuration used for this introduction tutorial is described below:

Motors:

OUT_A : Right motor
OUT_B: Not used
OUT_C: Left motor

Sensors:

SENSOR_1: Right touch sensor
SENSOR_2: Not used
SENSOR_3: Left touch sensor

4.1 Motor control

Motors are identified by the following constant values:

OUT_A, OUT_B, OUT_C

The functions that control the motors need a parameter that contains the list of motors to be used. Under these lines there are the functions to control the motors and examples of how to pass the parameters.

Motor activation

On: On(OUT_A+OUT_C)
On forward: OnFwd(OUT_A+OUT_B+OUT_C)
On backward: OnRev(OUT_A+OUT_C)

Motor deactivation

Brake: Off(OUT_A+OUT_B) //Brakes and blocks the motor
Deactivate: Float(OUT_A+OUT_C)//Just turns off the motor without blocking it

Way of rotation:

Reverse: Toggle(OUT_A)
Forward: Fwd(OUT_A)
Backward: Rev(OUT_A)

Speed control:

SetPower (OUT_B , 5); //The speed have to be a number between 0 and 7

Example

This program turns on motor A for 3 seconds and then stops it.

```
task main(){
    On(OUT_A);
    Wait(300);
    Off(OUT_A);
}
```

Exercise

Write a program that makes the robot move following a squared path.

4.2 Sensor control

Sensors are identified by the following reserved words:

SENSOR_1, SENSOR_2, SENSOR_3

To poll a sensor value, you can use its reserved word. The following code puts the value of the sensor in a variable called "a":

```
a = SENSOR_2;
```

The functions used to control the sensors can affect only one sensor at a time. Then the sensor identifiers can not be combined using the + operator to pass as a function parameter.

Sensor configuration

SetSensor(SENSOR_1 , SENSOR_TOUCH)

Constant	Sensor type	Mode	ClearSensor()
SENSOR_TOUCH	Touch	1(pressed) 0(not pressed)	No
SENSOR_LIGHT	Light	0 dark, 100 light	No
SENSOR_ROTATION	Rotation	Rotation counter	Yes
SENSOR_PULSE	Touch	Pulse counter	Yes
SENSOR_EDGE	Touch	Edge counter	Yes
SENSOR_CELSIUS	Temperature	Celsius degrees	No
SENSOR_FARENHEIT	Temperature	Fahrenheit degrees	No

ClearSensor(SENSOR_2) //Resets the sensor counter when it is possible

Example

This code makes the robot move forward until one of the sensors detects a collision:

```
task main(){
    SetSensor(SENSOR_1 , SENSOR_TOUCH)
    SetSensor(SENSOR_3 , SENSOR_TOUCH)

    while ( SENSOR_1==0 && SENSOR_3==0){
        OnFwd(OUT_A+OUT_C);
    }
    Float(OUT_A+OUT_C);
}
```

Exercise

Make a program that makes the robot move forward avoiding the obstacles on its way, using the touch sensors.

4.3 Multitask programming

Although the hardware limitations of the LEGO RCX, it offers us the possibility of defining several tasks and run them concurrently.

Example

This program has two tasks that run concurrently during a minute. The first task plays a sound whenever one of the touch sensor are pressed. The second task plays a sound every second.

```
task task1(){
    while(1){
        while (SENSOR_1==1 || SENSOR_2==1){
            PlaySound (SOUND_CLICK);
        }
    }
}

task task2(){
    while(1){
        PlaySound (SOUND_DOUBLE_BEEP);
        Wait(100);
    }
}

task main(){
    start task1;
    start task2;

    Wait(6000);

    stop task1;
    stop task2;
}
```

Exercise

Make the same program on 4.2 but multitask. One task moves the robot forward and the other task have to detect and avoid obstacles.

References

- [Ang]** Frank Angeli's report about light sensor
<http://www.crynwr.com/lego-robotics/light-sensor.html>
- [Gasp]** Michael Gasperi's personal page dedicated to RCX light sensor
<http://www.plazaeath.com/usr/gasperi/light.htm>
- [Hurb]** Philippe Hurbain's personal web dedicated to the RCX rotation sensor
<http://www.philohome.com/sensors/legorot.htm>
- [Hurb2]** Philippe Hurbain's personal web dedicated to the RCX motors analysis
<http://www.philohome.com/motors/motorcomp.htm>

Practical exercise with Lego MindStorms.

In the first part of the session you have learned and trained the basic skills to program a simple mobile robot. In this second part the goal is to implement the execution of a service task.

The robot is a fire-fighter that must look for fires and put them out. The fires will be candles. First, you should install the fan module and plug in the light sensors. Then, the configuration of the robot will be as follows:

Motors:

OUT_A : Right motor
OUT_B: Fan motor
OUT_C: Left motor

Sensors:

SENSOR_1: Right light sensor
SENSOR_2: Center light sensor
SENSOR_3: Left touch sensor

The exercise is organized in three scenarios with increasing difficulty.

1. There is only one candle. The robot must locate it and put it out.
2. There are many candles, and some of them can be fired at any time. The robot must work in continuous mode, wandering around while looking for fires, and putting them out when it finds them.
3. There is moving candle. The robot must follow them until it puts it out.

You must program the robot to accomplish its task in these three scenarios.