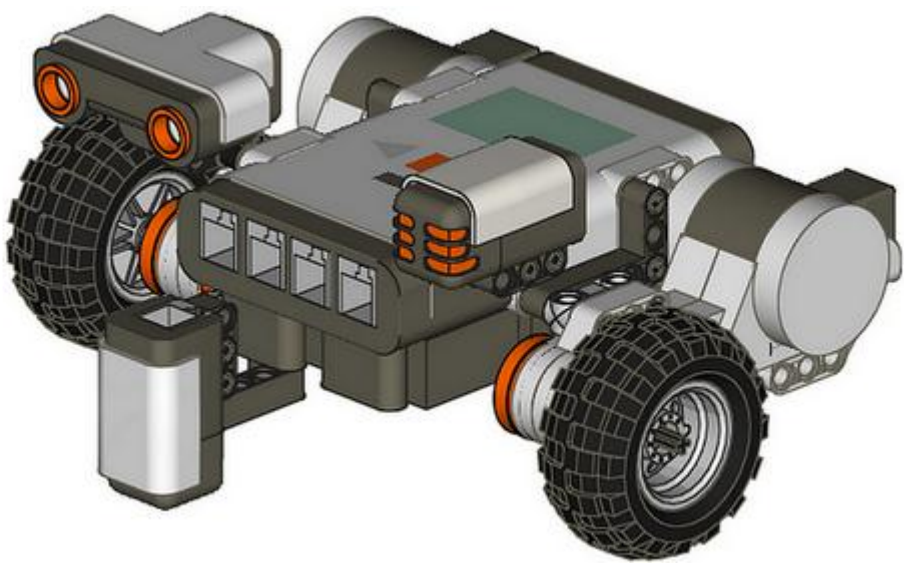
Programming the NXT with C#



School of Computer Science, University of Lincoln

University of Lincoln Computing Society

Getting Started

Now let's start to write a few simple programs which will control the NXT automatically without manual intervention. We are going to use a programming language called C# for this purpose, but your prior knowledge of programming is not necessary to complete these tutorials.

The programming environment should already be set-up for you. The most important window of the environment is, not surprisingly, there largest one – it contains the program code and this is where we will place our own lines of code. Some basic code is already there; the implemented program connects to the NXT and if the connection was successful, it simply waits for you to press the 'ESC' key. You should see some comments highlighted in green, which will provide more detail about each part of the program.

Connecting to your NXT

Before you can run a program for the first time, you will have to connect to your NXT. You can do this by completing the following steps:

1. Right click your Bluetooth icon (bottom right), and select 'Add a Device' (Figure 1).

2. Select your device from the available list and click 'next' (Figure 2).

3. Enter a password on both devices and wait for it to connect and install (Figure 3).

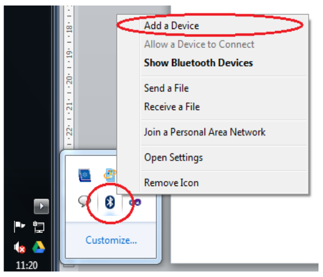


Figure 1: Add Bluetooth device

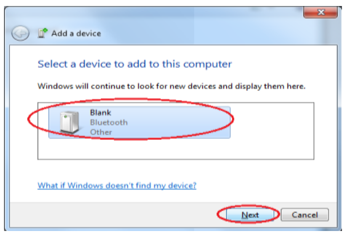

Add a device
Select a device to add to this computer
Windows will continue to look for new devices and display them here.
Blank
Bluetooth
Other
What if Windows doesn't find my device?
Next
Cancel

Figure 2: Connecting to NXT

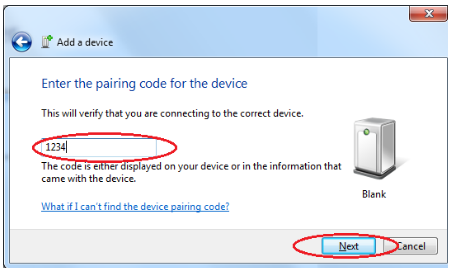
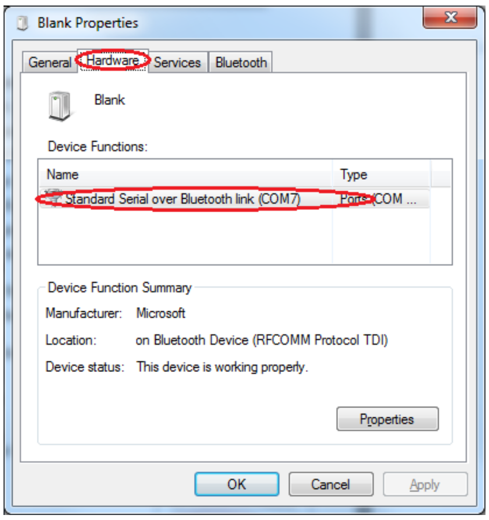

Add a device
Enter the pairing code for the device
This will verify that you are connecting to the correct device.
1234
The code is either displayed on your device or in the information that came with the device.
Blank
What if I can't find the device pairing code?
Next
Cancel

Figure 3: Enter Password

Now you have connected to your device you can see what COM port it is connected to, in a window address bar enter:

Control Panel\Hardware and Sound\Devices and Printers*your_Device_Name*

A window will show popup, click on the hardware tab and note down the Comport/s listed (Figure 4), you will need this when programming.


Blank Properties

General | Hardware | Services | Bluetooth

Blank

Device Functions:

Name	Type
Standard Serial over Bluetooth link (COM7)	Ports (COM ...

Device Function Summary

Manufacturer: Microsoft

Location: on Bluetooth Device (RFCOMM Protocol TDI)

Device status: This device is working properly.

Figure 4: Listed Comports

Now that you know what Comport your NXT has connected to you can now specify the Comport in the line that looks like the:

```
myNXT = new Robot("COM7");
```

To run your program press ‘F5’, it will take a moment to build the program. After a while a console window will appear with the information generated by your program.

C# Tutorial 1 – Movement

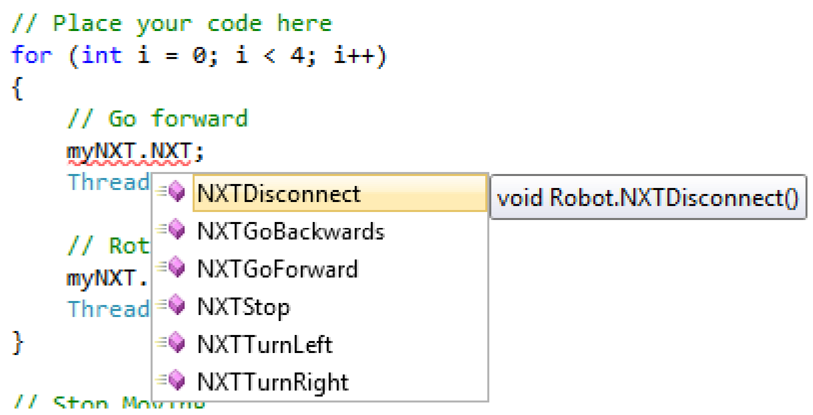
Now that you know how to build and run a program you can start sending commands to your NXT. The command to make your NXT go forward is

```
myNXT.NXTGoForward(100);
```

This will move your NXT forward with a speed of '100'. The robot will keep running this command until you send another one. To allow you to run commands one after another we can make use of

```
Thread.Sleep(Time in milliseconds);
```

This line of code pauses your program for a set amount of time let the command run for longer periods of time. There are many commands available to control your NXT in C# (e.g. GoForward, GoBackwards, TurnLeft and TurnRight) (Figure 5).



```
// Place your code here
for (int i = 0; i < 4; i++)
{
    // Go forward
    myNXT.NXT;
    Thread

    // Rot
    myNXT.
    Thread
}

// Stop Moving
```

Figure 5: NXT Commands

By using these commands the NXT can move in almost any direction you want it to.

Task 1 – Move in a Square

Making use of the `myNXT.NXTGoForward()`, `myNXT.TurnRight()` and `myNXT.NXTStop(motorport.portname)` commands we make the NXT more in a square.

You will you will first need to tell the NXT to go forward for 'X' amount of time, then you will need to make the NXT turn 90 degrees (this take about 300ms at a speed of 100). Add the following code just after the comment line: `// Place your code here` in the template program provided.

```
for (int i = 0; i < 4; i++)
{
     // Go forward
```

```
      myNXT.NXTGoForward(100);
      Thread.Sleep(500);

      // Rotate right
      myNXT.NXTTurnRight(100);
      Thread.Sleep(300);
}
```

So we know that out program has finished we can add make our NXT play a sound after it has completed the square. To do this we simply add the following line of code after the ‘For’ loop:

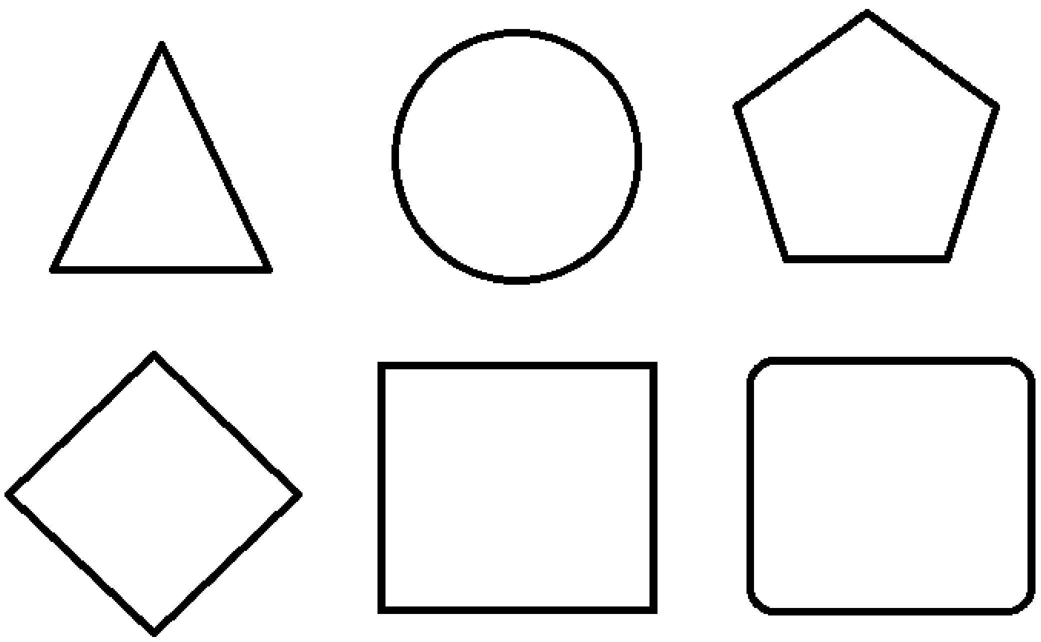
```
myNXT.DirectCommands.PlaySoundFile("Woops",false);
```

This command allows you play an sound that is on the NXT brick, you also have the option to loop the sound (True) or play only once (False).

If everything has gone according to plan your robot should move in a square and play a sound.

Task 2 – Doing more

Explore and experiment with the different commands and observe the behaviour of your NXT. Here is some shape you should try and make your NXT more in.



C# Tutorial 2 – Sound Sensor

The NXT comes with 4 different input ports and a variety of sensor you can use in your programs. The Sound sensor allows you detect the level of sounds in the surrounding area. We can use the Sound senor to enhance our previous program.

Here you will make your NXT move when it hears sounds and stop when it hears another sound. Before the NXT can hear the sounds we need to setup the sensor port. The code

```
myNXT.InPort2.Set(SensorType.SoundDB, SensorMode.RawMode);
```

sets ‘InPort2’ to be a soundDB sensor. For this program we are going to want to use the most up to date information from the sensor, to do this we use the command

```
myNXT.InPort2.Update();
```

Finally to get the data from the sensor we use the following code

```
int soundLevel = myNXT.InPort2.ScaledValue;
```

This code creates an integer variable called ‘soundLevel’ and populates it with the information from the sensor.

Task 1 – Move with the first sound, stop with the second

Here you will have to try and make your NXT move when it hears a sound, then stop when it hears another. This can be done with the following code

```
myNXT.InPort2.Set(SensorType.SoundDB, SensorMode.RawMode);
                myNXT.InPort2.Update();
                int soundLevel = myNXT.InPort2.ScaledValue;
```

First we must setup our sensor and make sure we get the most recent data, once we have this done we can test how loud the sound detected is.

```
if (soundLevel > 500)
                {
                 }
```

You can use this ‘if’ statement to determine how loud of a sound you would like to detect before running the rest of your program, you will need to place the rest of your code inside here. For the next part of the program you want your NXT to keep moving till it hears another sound, you can do this using a ‘While’ loop. Remember it is important that we always use the most up to date information; you will need to update your InPort after each loop. Place the following code into the ‘if’ statement (between the curly braces)

```
soundLevel = 0;

while (soundLevel < 500)
{
       // Enter your move commands here
       myNXT.InPort2.Update();
       soundLevel = myNXT.InPort2.ScaledValue;
}
myNXT.Stop(MotorPort.MotorAll);
```

You can input your own move commands just after the line `// Enter your move commands here`.

Task 2 – Doing more

Building on the program built in Task 1, try and make your NXT do different action depend on the level of sound it detects, try and add some sounds for each action.

Examples:

- Stop when it hears sound.
- Use different sound levels to make it turn.
- Use the different sound levels to determine what sound the NXT should make.

C# Tutorial 3 – Touch Sensor

Task 1 – Obstacle avoidance

Here you will are going to make your NXT change direction when it hits an obstacle. This can be done by checking the state of the touch sensor, if it is press then the NXT has hit an obstacle.

Start a new project using the template program, and as before in Tutorial 2 you will need to setup your InPort. This time you will need to set the sensor type to 'Switch'.

```
int mySwitch = 0;
myNXT.InPort1.Set(SensorType.Switch, SensorMode.RawMode);
mySwitch = myNXT.InPort1.RawADValue;
```

Now that you have the data from the sensor you can decide what to do, if it is pressed perform one set of instruction, if it is released perform a different set.

```
if (mySwitch < 200) // if mySwitch value is less than 200 it is pressed
{
       // Enter your move commands here
}
```

Remember to keep your InPort up to date by calling the update function as done in previous tutorials.

Task 2 – Doing more

Try building a program that makes your NXT perform different a range of different movements each time the Switch sensor is pressed. This could be done by using several 'if' statements together, and a variable to keep track of how many times the switch has been pressed.

Examples:

- Reverse and turn to the left
- Reverse and turn to the right
- Reverse and turn 180 degrees

C# Tutorial 4 – Ultrasonic Sensor

Task 1 – Obstacle Detection and Avoidance

For this tutorial you will be learning how detect obstacles before your NXT hit them, and react in similar ways with the Switch sensor. You will also learn how to make your NXT move in a random direction.

Start a new project using the template program, and as before in Tutorial 2 you will need to setup your InPort. This time you will need to set the sensor type to 'Sonar'.

```
int mySonar = 0.0;
myNXT.InPort4.Set(SensorType.Sonar, SensorMode.RawMode);
myNXT.InPort4.Update();
```

For this tutorial you will need to know how to create a random number, this can be done by using the 'Random' class. The following code creates a random number generator based off the current time.

```
Random myRandom = new Random(DateTime.Now.Millisecond);
```

You can now use the variable 'myRandom' to create random numbers; you simply use the following code whenever you want a new random number.

```
myRandom.Next(min value, max value)
```

You can use this to set the power values of the output ports; the following code is an example of randomly allocating power to an output port.

```
myNXT.OutPortB.SetOutputState(myRandom.Next(-100, -50), Mode.MotorOn,
RegulationMode.Motor_On, 0, RunState.Running);
```

Using everything you have learned so far try and create a program that will make your NXT move around the room and change direction when it detects and object.

Remember to update your sensor data each time you want a new reading.

Task 2 – Doing more

Alter your previous program to make your NXT play different sounds depending on how close it is to an object. Note you stop sounds by using the following line of code

```
myNXT.DirectCommands.StopSoundPlayback();
```

C# Tutorial 5 – Light Sensor

Task 1 – Line Following

Using everything you have learned so far try and create a program that will make your NXT follow a since coloured line round a track. To setup the InPort you can use the following code

```
int myShade = 0;
myNXT.InPort3.Set(SensorType.LightActive, SensorMode.RawMode);
myShade = myNXT.InPort3.ScaledValue;
```

Tip: You can use the colour of the track to make your NXT perform one action and the background colour to perform another action.