LEGO NXT Robots using NXC – Part 3

Sensors

Off course you can connect sensors to NXT to make the robot react to external events. Before I can show how to do this, we must change the robot a bit by adding a touch sensor. As before, follow the Tribot instructions to build the front bumper.

Waiting for a sensor

Let us start with a very simple program in which the robot drives forwards until it hits something. Here it is:

```
task main()
{
	SetSensor(IN_1,SENSOR_TOUCH);
	OnFwd(OUT_AC, 75);
	until (SENSOR_1 == 1);
	Off(OUT_AC);
}
```

There are two important lines here. The first line of the program tells the robot what type of sensor we use. `IN_1` is the number of the input to which we connected the sensor. The other sensor inputs are called `IN_2`, `IN_3` and `IN_4`. `SENSOR_TOUCH` indicates that this is a touch sensor. For the light sensor we would use `SENSOR_LIGHT`. After we specified the type of the sensor, the program switches on both motors and the robot starts moving forwards. The next statement is a very useful construction. It waits until the condition between the brackets is true. This condition says that the value of the sensor `SENSOR_1` must be 1, which means that the sensor is pressed. As long as the sensor is not pressed, the value is 0. So this statement waits until the sensor is pressed. Then we switch off the motors and the task is finished.

Acting on a touch sensor

Let us now try to make the robot avoid obstacles. Whenever the robot hits an object, we let it move back a bit, make a turn, and then continue. Here is the program:

```
task main()
{
	SetSensorTouch(IN_1);
	OnFwd(OUT_AC, 75);
	while (true)
	{
		if (SENSOR_1 == 1)
		{
			OnRev(OUT_AC, 75); Wait(300);
			OnFwd(OUT_A, 75); Wait(300);
			OnFwd(OUT_AC, 75);
		}
	}
}
```

As in the previous example, we first indicate the type of the sensor. Next the robot starts moving forwards. In the infinite **while** loop we constantly test whether the sensor is touched and, if so, move back for 300ms, turn right for 300ms, and then continue forwards again.

Light sensor

Besides the touch sensor, you also get a light sensor, a sound sensor and a digital ultrasonic sensor with Mindstorms NXT system. The light sensor can be triggered to emit light or not, so you can measure the amount of reflected light or ambient light in a particular direction. Measuring reflected light is particularly useful when making a robot follow a line on the floor. This is what we are going to do in the next example. To go on with experiments, finish building Tribot. Connect light sensor to input 3, sound sensor to input 2 and ultrasonic sensor to input 4, as indicated by instructions.

We also need the test pad with the black track that comes with the NXT set. The basic principle of line following is that the robot keeps trying to stay right on the border of the black line, turning away from line if the light level is too low (and sensor is in the middle of the line) and turning towards the line if the sensor is out of the track and detects a high light level. Here is a very simple program doing line following with a single light threshold value.

```
#define THRESHOLD 40
task main()
{
	SetSensorLight(IN_3);
	OnFwd(OUT_AC, 75);
	while (true)
	{
		if (Sensor(IN_3) > THRESHOLD)
		{
			OnRev(OUT_C, 75);
			Wait(100);
			until(Sensor(IN_3) <= THRESHOLD);
			OnFwd(OUT_AC, 75);
		}
	}
}
```

The program first configures port 3 as a light sensor. Next it sets the robot to move forwards and goes into an infinite loop. Whenever the light value is bigger than 40 (we use a constant here such that this can be adapted easily, because it depends a lot on the surrounding light) we reverse one motor and wait till we are on the track again.

As you will see when you execute the program, the motion is not very smooth. Try adding a `Wait(100)` command before the **until** command to make the robot move better. Note that the program does not work for moving counter-clockwise. To enable motion along arbitrary path a much more complicated program is required.

Sound sensor

Using the sound sensor you can transform your expensive NXT set into a clapper! We are going to write a program that waits for a loud sound, and drives the robot until another sound is detected. Attach the sound sensor to port 2, as described in Tribot instructions guide.

```
#define THRESHOLD 40
#define MIC SENSOR_2
task main()
{
	SetSensorSound(IN_2);
	while(true)
	{
		until(MIC > THRESHOLD);
		OnFwd(OUT_AC, 75);
		Wait(300);
		until(MIC > THRESHOLD);
		Off(OUT_AC);
		Wait(300);
	}
}
```

We first define a `THRESHOLD` constant and an alias for `SENSOR_2`; in the main task, we configure the port 2 to read data from the sound sensor and we start a forever loop.
Using the **until** statement, the program waits for the sound level to be greater than the threshold we chose: note that `SENSOR_2` is not just a name, but a macro that returns the sound value read from the sensor.

If a loud sound occurs, the robot starts to go straight until another sound stops it.

The **wait** statements have been inserted because otherwise the robot would start and stop instantly: in fact, the NXT is so fast that takes no time to execute lines between the two **until** statements. If you try to comment out the first and the second **wait**, you will understand this better. An alternative to the use of **until** to wait for events is **while**, it is enough to put inside the parentheses a complementary condition, e.g.

```
while(MIC <= THRESHOLD).
```

There is not much more to know about NXT analog sensors; just remember that both light and sound sensors give you a reading from 0 to 100.

Ultrasonic sensor

Ultrasonic sensor works as a sonar: roughly speaking, it sends a burst of ultrasonic waves and measures the time needed for the waves to be reflected back by the object in sight. This is a digital sensor, meaning it has an embedded integrated device to analyze and send data. With this new sensor you can make a robot see and avoid an obstacle before actually hitting it (as is for touch sensor).

```
#define NEAR 15 //cm
task main()
{
	SetSensorLowspeed(IN_4);
	while(true)
	{
		OnFwd(OUT_AC,50);
		while(SensorUS(IN_4)>NEAR);
		Off(OUT_AC);
		OnRev(OUT_C,100);
		Wait(800);
	}
}
```

The program initializes port 4 to read data from digital US sensor; then runs forever a loop where robots goes straight until something nearer than NEAR cm (15cm in our example) is in sight, then backups a bit and begins going straight again.