Using Lejos to Read the NXT Sensors

John Kelleher Brian Mac Namee

School of Computing, Dublin Institute of Technology

DT211-3 Robot Technology Fundamentals

Outline

Introduction

The NXT comes with four sensors:

1. the touch sensor
2. the sound sensor
3. the light sensor
4. the ultrasonic sensor

leJOS NXJ provides software abstractions of all these sensor types.

Hardware Connection and Software Instance Creation

- A physical sensor must be connected to a port
- The possibilities are: SensorPort.S1, S2, S3 or S4.
- To interface with a sensor you create an instance of the Lejos sensor class that abstracts the sensor type.
- This software object needs to know which sensor port the sensor it is interfacing to is connected to.
- You can pass this information to the sensor object as a parameter to the constructor when the object is being created.

Class Constructor

To use a touch sensor, you create an instance of it, using the constructor:

```
TouchSensor(SensorPort port)
```

Are we touching something?

To test if the touch sensor is pressed, you use the isPressed() method:

```
boolean isPressed()
```

Example

TouchTest.java

```
1  import lejos.nxt.*;
2
3  public class TouchTest {
4    public static void main(String[] args) throws
         Exception {
5      TouchSensor touch = new TouchSensor(SensorPort.S1);
6
7      while (!touch.isPressed()) ;
8      LCD.drawString("Finished", 3, 4);
9    }
10 }
```

Overview

- The Light Sensor enables your robot to distinguish between light and dark.
- It can read the light intensity in a room and measure the light intensity of colored surfaces.

Figure: Comparision of Human Vision and LightSensor Vision

Class Constructor

To use a light sensor, you create an instance of it using the constructor:

```
public LightSensor(SensorPort port)
```

Example

TouchTest.java

```
1  import lejos.nxt.*;
2
3  public class LightTest {
4    public static void main(String[] args) throws
         Exception {
5      LightSensor light = new LightSensor(SensorPort.S1);
6
7      while (true) {
8        LCD.drawInt(light.readValue(), 4, 0, 0);
9        LCD.drawInt(light.readNormalizedValue(), 4, 0, 1);
10       LCD.drawInt(SensorPort.S1.readRawValue(), 4, 0, 2)
             ;
11       LCD.drawInt(SensorPort.S1.readValue(), 4, 0, 3);
12     }
13   }
14 }
```

Sensor Overview

- The Sound Sensor can detect both decibels [dB] and adjusted decibel [dBA]. A decibel is a measurement of sound pressure.
  - dBA: in detecting adjusted decibels, the sensitivity of the sensor is adapted to the sensitivity of the human ear. In other words, these are the sounds that your ears are able to hear.
  - dB: in detecting standard [unadjusted] decibels, all sounds are measured with equal sensitivity. Thus, these sounds may include some that are too high or too low for the human ear to hear.

Sensor Overview

- The Sound Sensor can measure sound pressure levels up to 90 dB Ð about the level of a lawnmower.
- Sound pressure levels are extremely complicated, so the Sound Sensor readings on the NXT are displayed in percent %.
- The lower the percent the quieter the sound. For example:
  - 4-5% is like a silent living room
  - 5-10% would be someone talking some distance away
  - 10-30% is normal conversation close to the sensor or music played at a normal level
  - 30-100% are people shouting or music being played at a high volume

Modes DB and DBA

The Lejos SoundSensor class supports supports two modes: DB and DBA. These modes give different frequency response, so that it may be possible to get an idea of the frequency of a sound by switching between modes.

Class Constructors

There are two constructors:

Create a sound sensor using DB mode

SoundSensor(SensorPort port)

Create a sound sensor using DBA mode (if arg2 = true)

SoundSensor(SensorPort port, boolean dba)

Switching Modes

You can switch modes with:

```
void setDBA(boolean dba)
```

Example (using DB Mode only

SoundScope.java

```
1  import lejos.nxt.*;
2
3  public class SoundScope {
4    public static void main(String[] args) throws
         Exception {
5      SoundSensor sound = new SoundSensor(SensorPort.S1);
6
7      while (!Button.ESCAPE.isPressed()) {
8        LCD.clear();
9        for (int i = 0; i < 100; i++) {
10         LCD.setPixel(1, i, 60 - (sound.readValue() / 2))
               ;
11         Thread.sleep(20);
12       }
13     }
14   }
15 }
```

Overview

- The Ultrasonic sensor can measure the distance from the sensor to something that it is facing.
- The Ultrasonic Sensor measures distance in centimeters and in inches. It is able to measure distances from 0 to 255 centimeters with a precision of +/- 3 cm.
- The Ultrasonic Sensor uses the same scientific principle as bats: it measures distance by calculating the time it takes for a sound wave to hit an object and return Ð just like an echo.
- Large sized objects with hard surfaces return the best readings. Objects made of soft fabric or that are curved [like a ball] or are very thin or small can be difficult for the sensor to detect.
- *Note that two or more Ultrasonic Sensors operating in the same room may interrupt each otherÕs readings.

Class Constructor

The Lejos API provides the class UltrasonicSenor. By creating an instance of this class we can get distance readings from our Ultrasonic Sensor. We can create an instance of this class using the following constructor:

```
UltrasonicSensor(Port SensorPort)
```

Modes: continuous and ping

The Ultrasonic sensor operates in two modes, continuous (defalt) and ping.

Getting the Echo Readings

When in continuous mode the sensor sends out pings as often as it can and the most recently obtained result is available via a call to

```
int getDistance()
```

The return value is in centimeters. If no echo was detected, the returned value is 255. The maximum range of the sensor is about 170 cm.

Example (using Continuous Mode)

SoundScope.java

```
1  import lejos.nxt.*;
2
3  public class SonicTest {
4    public static void main(String[] args) throws
         Exception {
5      UltrasonicSensor sonic = new UltrasonicSensor(
           SensorPort.S1);
6
7      while (!Button.ESCAPE.isPressed()) {
8        LCD.clear();
9        LCD.drawString(sonic.getVersion(), 0, 0);
10       LCD.drawString(sonic.getProductID(), 0, 1);
11       LCD.drawString(sonic.getSensorType(), 0, 2);
12       LCD.drawInt(sonic.getDistance(), 0, 3);
13     }
14   }
15 }
```

Overview

When in ping mode, a ping is sent only when a call is made to

```
void ping()
```

Getting the Echo Readings

Invoking the ping() method switches the sensor into ping mode and sends a single ping and up to 8 echoes are captured. These echos may be read by making a call to:

```
int readDistances(int [] distances)
```

- You provide an integer array of length 8 that contains the data after the method returns.
- A delay of approximately 20ms is required between the call to ping and getDistances(). This delay is not included in the method.
- Calls to getDistances() before this period may result in an error or no data being returned.
- The normal getDistance() call may also be used with ping, returning information for the first echo.

Switching Back to Continuous Mode

Invoking ping() will disable the default continuous mode. To switch back to continuous mode, call

```
int continuous()
```

Today we looked at: