MATLAB Interface to the LEGO Mindstorms NXT

This directory contains several MATLAB classes for communicating with the LEGO NXT through a Bluetooth Serial interface (see Bluetooth.pdf for details on setting up the Bluetooth connection).

Low-Level Interface

`nxtInterface`	The low-level interface to the LEGO NXT. It implements the basic "direct command" API. To see a list of available commands, type "help nxtInterface/Contents"

High-Level Interface

`legoNXT`	The high-level interface for communicating with the LEGO NXT and for keeping track of its state.
`outputPort`	A high-level interface for communicating with the LEGO NXT servomotors.
`inputPort`	A high-level generic interface for communicating with LEGO NXT sensors. It hides the details of whether a sensor is analog or digital.
`analogSensor`	A high-level interface for communicating with LEGO NXT Analog sensors.
`distanceSensor`	A high-level interface for communicating with the LEGO NXT ultrasonic digital sensor.

> Typically, you will only need to create the legoNXT object. This object will keep track of the inputPort, outputPort and nxtInterface objects for you.

The MATLAB interface has been developed and tested with R2007b (MATLAB version 7.5), but it should work with MATLAB version 7 and later. Please send questions or comments to mindstorms@mathworks.com

Useful links

LEGO Mindstorms home page:

http://mindstorms.lego.com/Overview/

LEGO Mindstorms NXT user forums:

http://forums.nxtasy.org/
http://news.lugnet.com/robotics/nxt/
http://thenxtstep.com/smf/

Running the Demos

To run the NXT demo GUI, using Bluetooth serial port COM40

```
>> nxtdemo('COM40')
```

To open a GUI using a test or a "dummy" LEGO NXT system

```
>> nxtdemo('test')
```

There is also a tribot demo (`demo_tribot`) that demonstrates how to control a robot programmatically. When the demo is run, the tribot moves forward until the touch sensor is depressed. When this happens, the tribot plays a "Woops" sound, backs up, and turns in a random direction (beeping all the while). After this action, it starts moving forward again. See `demo_tribot.m` for more information.

Sample Usage of a legoNXT Object

To create and use a legoNXT object

```
>> nxt = legoNXT('COM40');
Lego NXT object
 Serial port: COM17
 Automatic KeepAlive messages to NXT: OFF
 Input ports
   Port1: none
   Port2: none
   Port3: none
   Port4: none
 Output ports
   PortA: Stopped for 0.19 sec
   PortB: Stopped for 0.39 sec
   PortC: Stopped for 0.39 sec
```

To access the various ports, use the dot-notation

```
>> nxt.PortA
>> nxt.PortC
>> nxt.Port1
>> nxt.Port4
```

Tell the legoNXT object to periodically send "keep alive" messages to the LEGO NXT

```
>> set(nxt, 'AutoKeepAlive', 'on')
```

To see a list of properties

```
>> get(nxt)
```

Sample Usage – Motor Control

To specify the power setpoint for the motor at Port A

```
>> set(nxt.PortA, 'Power', 20)
```

To control the motor connected to Port A

```
>> start(nxt.PortA)
>> start(nxt.PortA, 30); % overrides the power level specified via SET
>> stop(nxt.PortA)
```

To make a controlled turn to the left

```
>> leftm = nxt.PortB;
>> rightm = nxt.PortC;
>> start(leftm, 10); start(rightm, 30);
>> pause(1);
>> stop(lefm); stop(rightm);
```

To see the list of methods for the output port

```
>> methods(nxt.PortA)
delete          get             outputPort      start           stop
display         getdata         set             startSchedule
```

To see help on a particular method

```
>> help outputPort/startSchedule
```

Sample Usage – Reading Sensors

To configure a sensor connected to Port 1

```
>> set(nxt.Port1, 'type', 'touch')
```

To see the list of allowed sensors and modes

```
>> help inputPort/set
```

To get data from a sensor

```
>> v = getdata(nxt.Port1)
v =
    20
```

Another way to get the data

```
>> touchSensor = nxt.Port1;
>> v = getdata(touchSensor)
v =
    20
```

Sample Usage – nxtInterface

You don't need to directly use nxtInterface for motor or sensor control since legoNXT does it for you. However, if you want to do things like play sound files or run programs, then you need the nxtInterface object.

Get the nxtInterface object being used by legoNXT

```
>> nxti = get(nxt, 'NxtInterface');
```

See the methods support by nxtInterface

```
>> methods(nxti)
Methods for class nxtInterface:
Contents          getOutputState    playSoundFile       setOutputState
delete            keepAlive         playTone            startProgram
display           lsGetStatus       resetMotorPosition  stopProgram
get               lsRead            resetScaledValue    stopSoundPlayback
getBatteryLevel   lsWrite           set
getInputValue     nxtInterface      setInputMode
```

Play a 1000 Hz tone for 300 milliseconds

```
>> playTone(nxti,1000,300)
```

Play a sound file (that is already present on the LEGO NXT)

```
>> playSoundFile(nxti, 'Hello.rso', 0)
```

Get the battery level of the LEGO NXT in millivolts

```
>> getBatteryLevel(nxti)
```

More details on the NXT "direct command" interface can be found by going to http://mindstorms.lego.com/Overview/NXTreme.aspx and downloading the "Bluetooth Developer Kit (BDK)".

Troubleshooting

Symptom: MATLAB is unable to find the legoNXT commands

You need to add the directory with the legoNXT files to your MATLAB path. Do the following at the MATLAB prompt:

```
>> addpath 'C:\My Documents\MATLAB\work\LegoBluetooth\'
>> savepath
```

(Replace '`C:\My Documents\MATLAB\work\LegoBluetooth\'` with the appropriate directory name in your system).

Symptom: The LEGO NXT is not responding to MATLAB commands

It is possible that the LEGO NXT has automatically turned itself off (went into sleep mode). If so:

1. Turn on the NXT by pressing the orange button on the NXT brick.
2. Go to your Bluetooth device manager and make sure that the Bluetooth serial connection to the NXT is still valid (see Bluetooth.pdf).

To prevent this from happening again, send "keep alive" messages using nxtInterface/keepAlive,

```
>> nxti = get(nxt, 'NxtInterface');
>> keepAlive(nxti);
```

or, make sure that the "AutoKeepAlive" parameter of the legoNXT object is 'on'.

```
>> set(nxt, 'AutoKeepAlive', 'on');
```

Symptom: The LEGO NXT is on, but it is still not responding to MATLAB commands

The Bluetooth serial connection may be in an invalid state. To reset this connection:

1. Restart the LEGO NXT (i.e., turn it off and on), and re-establish its Bluetooth serial connection to your computer.
2. Delete the legoNXT or nxtInterface objects. For example:
   ```
   >> delete(nxt);
   ```
3. Clear all existing serial port connections:
   ```
   >> delete(instrfindall);
   ```
4. Recreate the legoNXT object:
   ```
   >> nxt = legoNXT('COM40');
   ```

Symptom: Timeout errors (or invalid data) when reading sensor data in a timer callback

1. One potential problem is if the timer callback takes longer to run than the timer period (e.g., the timer period is 10 msec but the timer callback takes 15 msec to run). To avoid this problem, use 'FixedSpacing' as the execution mode instead of 'FixedRate' or 'FixedDelay'.

2. It is also important to keep in mind that the communication latency between the computer and the LEGO NXT (the time between sending a command to read the sensor and getting the sensor data) may be as long as 50 milliseconds. If your timer period is faster than this, you will encounter more timeout errors.

3. The legoNXT and nxtInterface objects resend commands if there is a communication problem. So, a call like `getdata(nxt.Port1)` can potentially take as long as 2 seconds. If you want a much faster latency, try the following:

- Reduce the number of retries, e.g:

```
>> nxti = get(nxt, 'NxtInterface');
>> set(nxti, 'NumRetries', 0)
```

- Reduce the default timeout for serial port communication. To do this, edit the constructor for the nxtInterface class:

```
>> edit nxtInterface\nxtInterface
```

  and change the default `timeoutSecs` value.

Symptom: Difficulty synchronizing two motors precisely.

Due to limitations in the NXT's "Direct command" interface, it is difficult to synchronize motors to start and stop at the same time. Two suggestions that might be helpful:

1. Issue the START and STOP commands directly to the output port objects. For example, if you are currently doing the following:

```
>> start(nxt.PortB); start(nxt.PortC);
```

   try this instead (it is slightly faster):

```
>> leftmotor = nxt.PortB; rightmotor = nxt.PortC;
>> start(leftmotor); start(rightmotor);
```

2. According to the NXT documentation, it is possible (in principle) to synchronize two motors using nxtInterface/setOutputState. We were not able to get this to work, but if you are comfortable with the low-level direct-command interface, you could give this a try.

Symptom: Difficulty with precise motor control (e.g., to make a quarter turn)

This difficulty is also a result of the NXT's "Direct command" interface. As with motor synchronization, it is possible to do this (in principle) using nxtInterface/setOutputState, but we were not able to get it to work reliably.

There is some evidence that using the LeJOS firmware on the NXT brick (http://lejos.sourceforge.net/) allows for more precise motor control when using the direct-command Bluetooth interface.