

Setup for Remote iMobot Control in Ch

David Ko
Mechanical and Aerospace Engineering

October 12, 2011

1 The `imobotcomms` iMobot Remote Control Library

The `imobotcomms` library is a collection of functions geared towards controlling the motors and reading sensor values of an iMobot module via the Bluetooth wireless protocol. The functions are designed to be intuitive and easy to use. Various functions are provided to control or obtain the speed, direction, and position of the motors. The API includes C-style functions as well as a C++ class called `CiMobotComms` to facilitate C++ style api function calls.

This documentation introduces the basic computer setup required for controlling the iMobot, as well as several demo programs and a complete reference for all API function provided with the `imobotcomms` library.

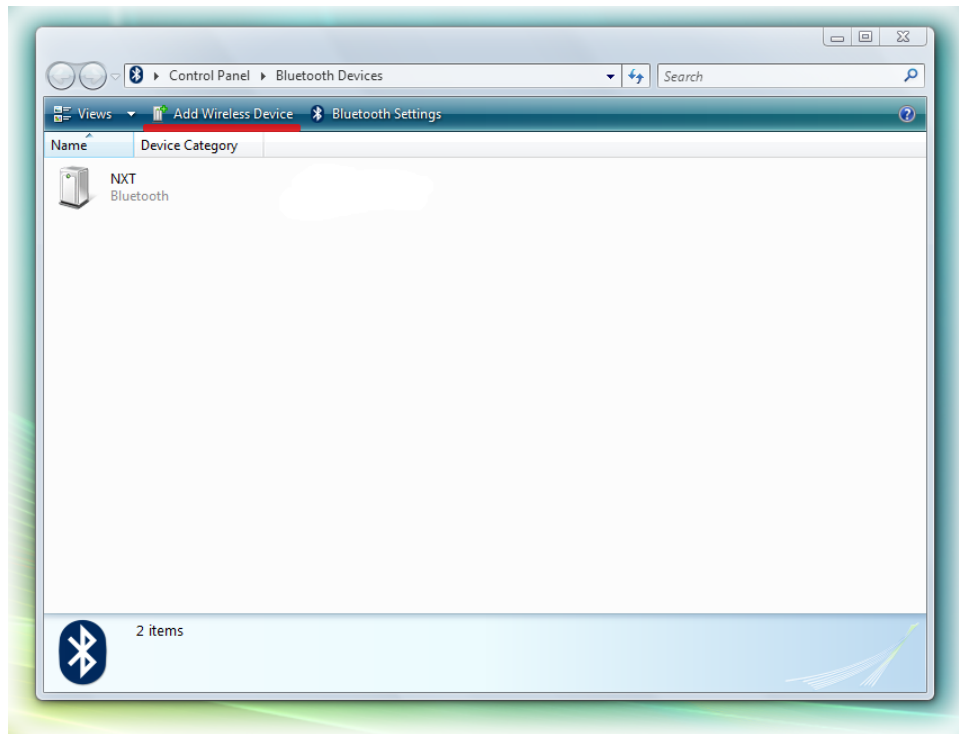
2 Bluetooth Pairing with the iMobot

To control the iMobot with the Bluetooth wireless protocol, the controlling computer must be equipped with Bluetooth. If the computer does not have Bluetooth built-in, an external USB Bluetooth dongle may be used. The following instructions are for a Windows 7 computer with built-in Bluetooth. The basic process is the same for Windows XP and Vista, although the screenshots may appear different.

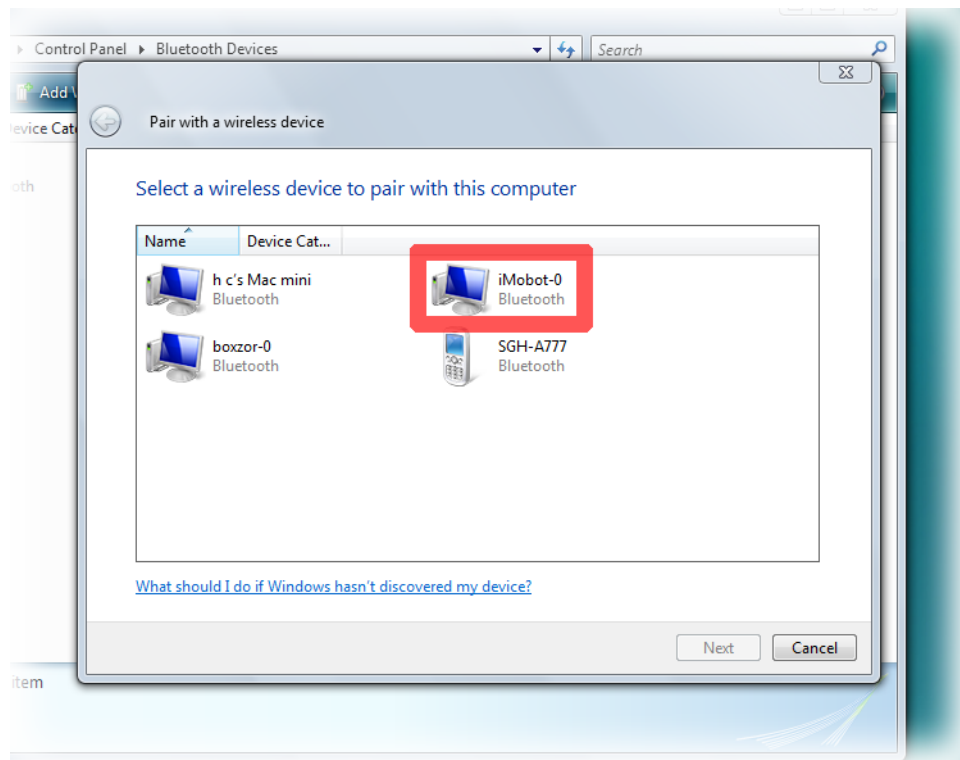
The first step is to open the Bluetooth applet by double-clicking the Bluetooth icon in the applet tray normally found at the bottom right of the screen, as shown highlighted in red in the following figure.



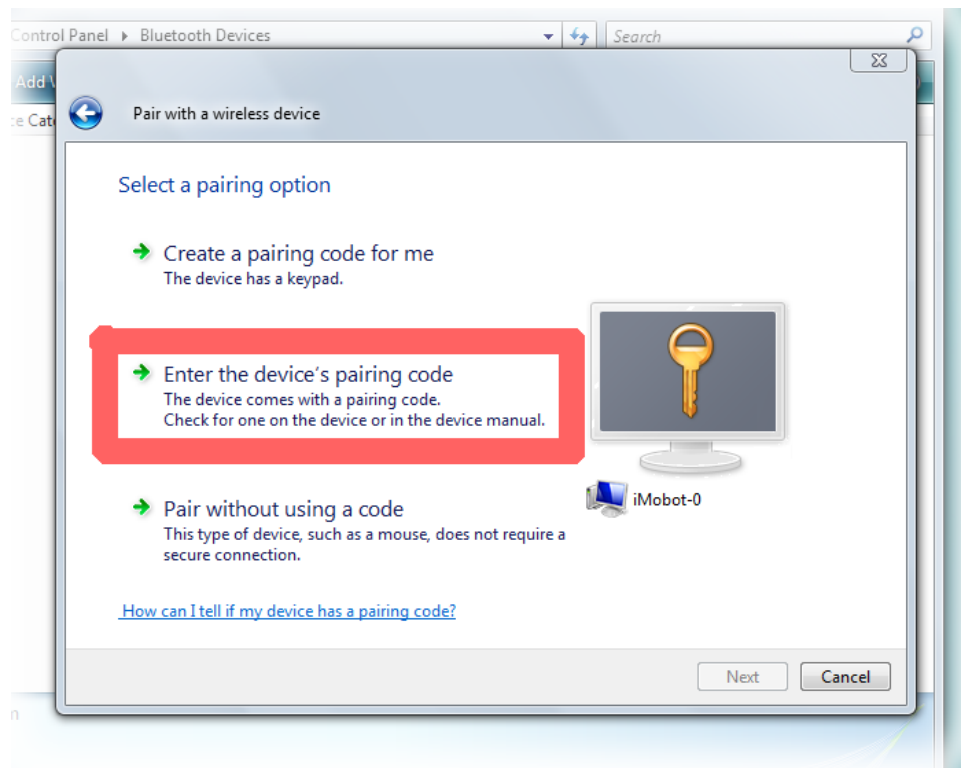
After double-clicking the icon, a new window will appear similar to the following.



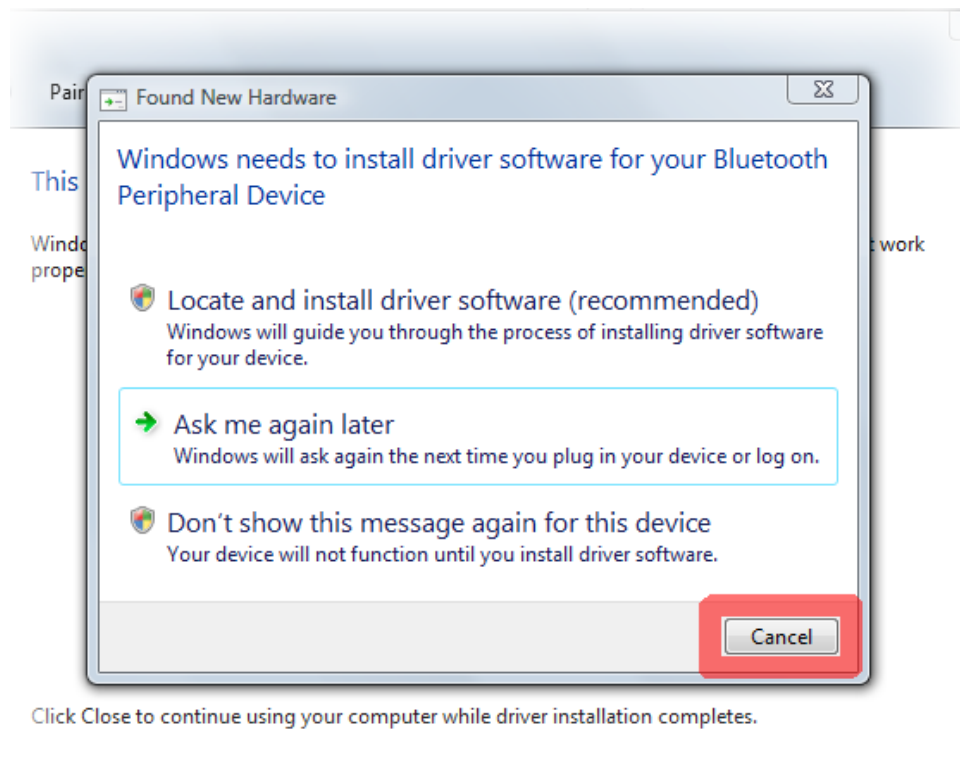
Next, click on the button labeled "Add Wireless Device" towards the top of the window. This will bring up the following dialog.



This dialog shows a list of all Bluetooth wireless devices that are in range. Among them should be the iMobot you wish to connect to. If the iMobot does not appear on this list, please ensure that the iMobot is within 10 meters of the connecting computer and that the iMobot is powered on. Double click on the icon representing the iMobot to proceed. Once you have double-clicked the icon, the following dialog box should appear.

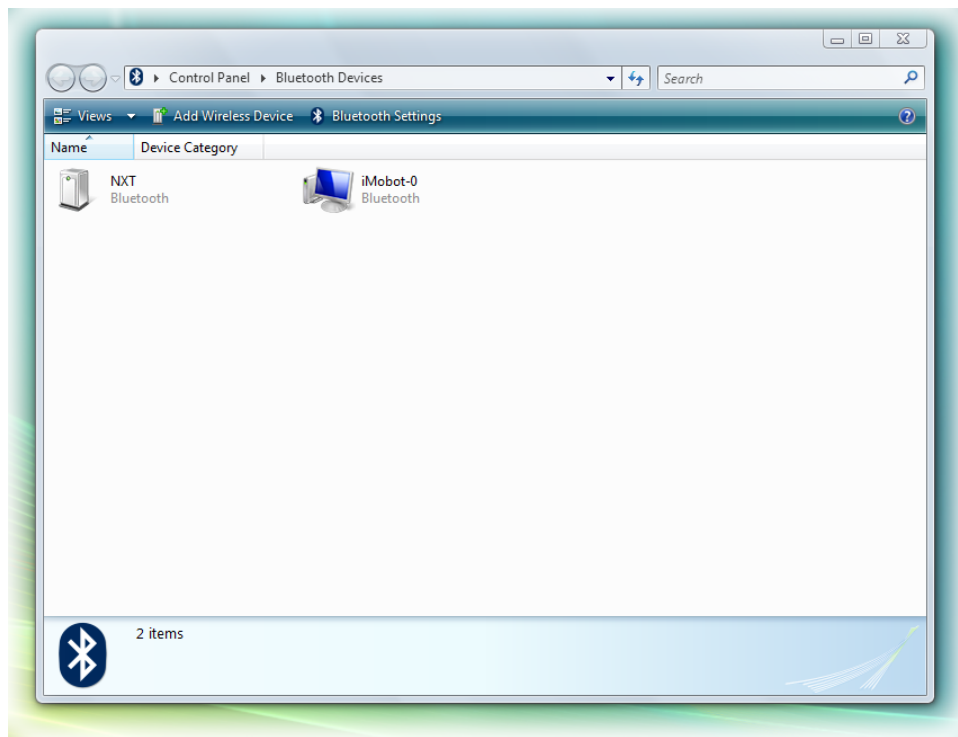


Select the second option, labeled “Enter the device’s pairing code”. iRobot modules come hard-coded with a default pairing code. When prompted for the pairing code, enter “1234”. Once the computer is paired with the iRobot, the following dialog box may pop up asking to install drivers.

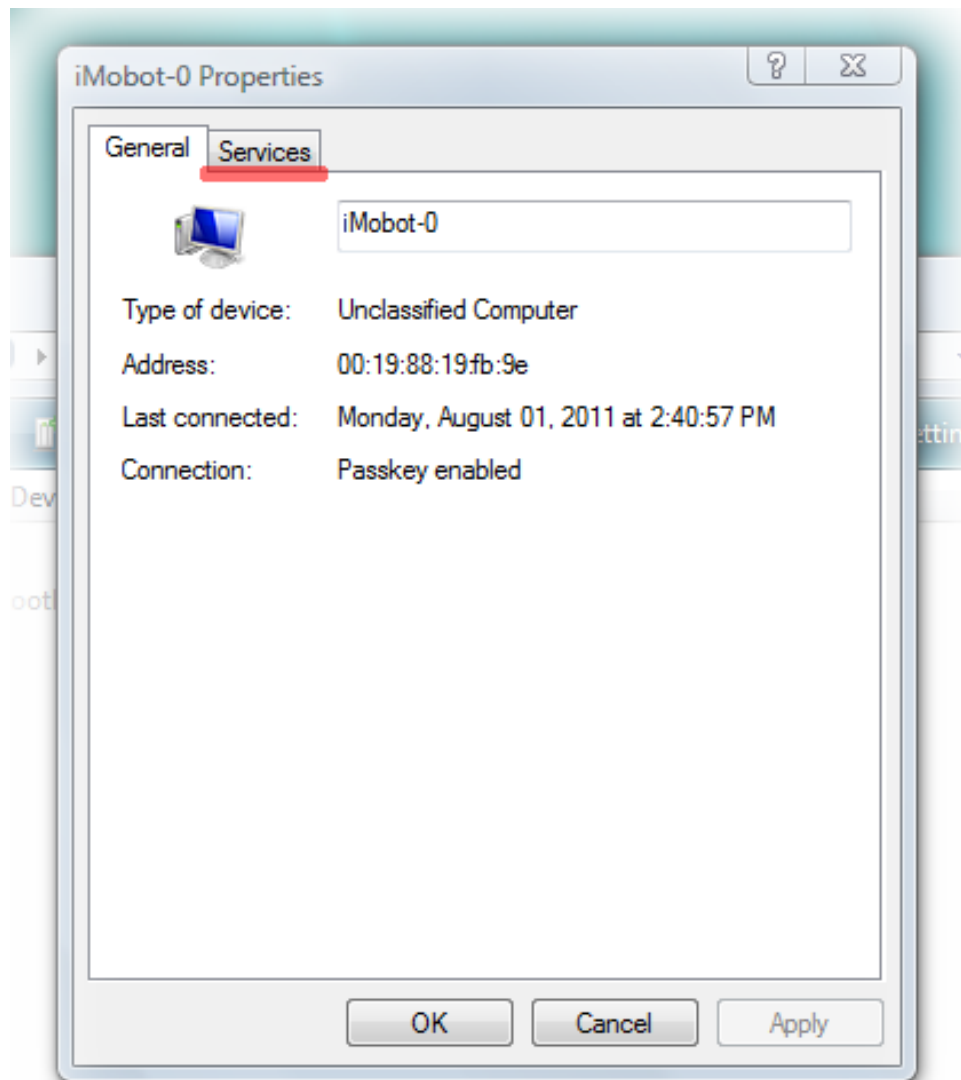


If the previously illustrated dialog box appears, just click the “cancel” button at the bottom right. No extra drivers are necessary for controlling the iMobot module.

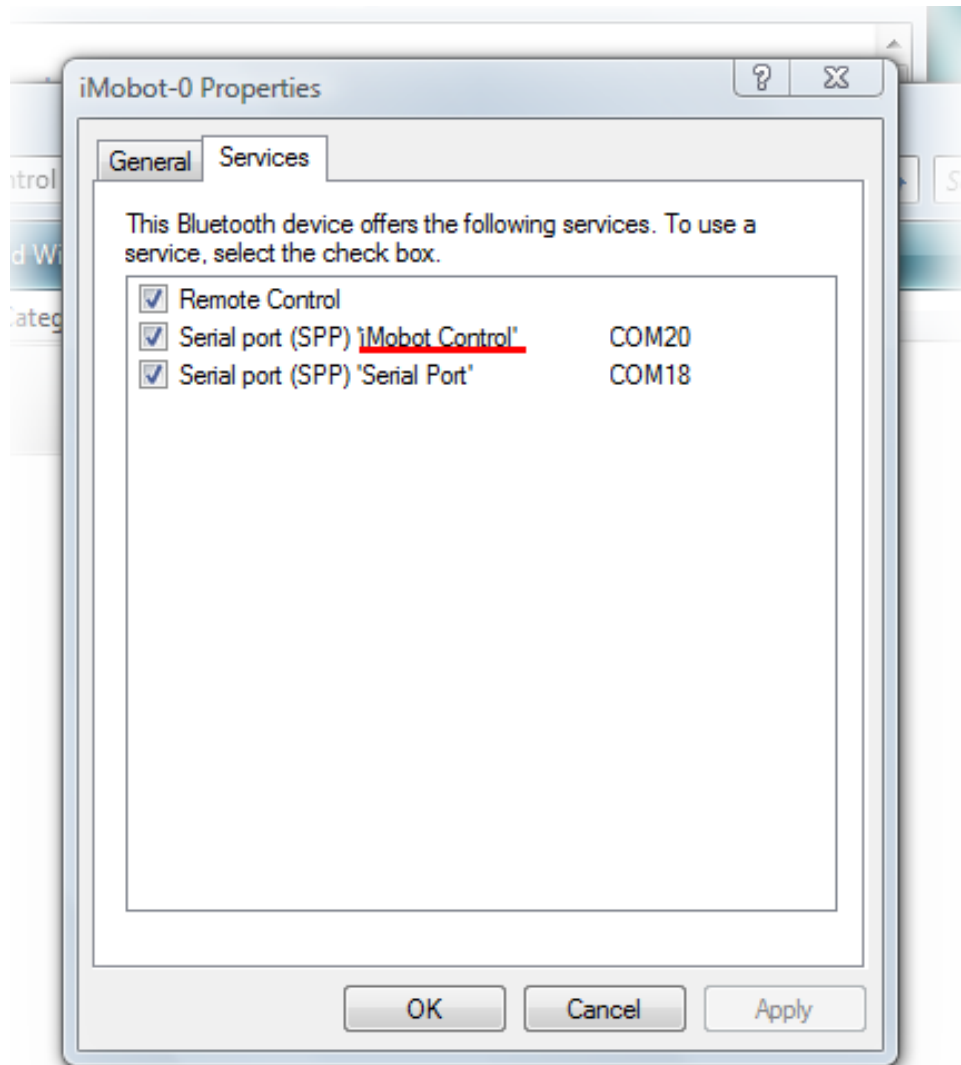
At this point, the following dialog should be shown.



The next step is to enable the iRobot control service. Double-click on the icon denoting the iRobot module to bring up the following dialog:



Click on the tab labeled "Services"



Ensure that the service titled “iMobot Control” is enabled. If it is not enabled, click on the check-box to enable it. Click on the “Ok” button to accept the changes and close the dialogs. The iMobot is now ready to be controlled with the iMobotComms library.

3 Basics of a Ch iMobot Program

To help the user become acquainted with the iMobot control programs, one sample program will be presented to illustrate the basics and minimum requirements

of an iMobot control program.

3.1 getting_started.ch Source Code

```
/* Filename: getting_started.ch */

#include <stdio.h>
#include <imobotcomms.h>

CiMobotComms robot;
/* Set the robot to "home" position, where all joint angles are 0 degrees. */
robot.moveZero();
robot.moveWait();

/* Rotate each of the faceplates by 90 degrees */
robot.moveJointTo(IMOBOT_JOINT3, 90);
robot.moveJointTo(IMOBOT_JOINT4, 90);

/* Wait for the movement to complete */
robot.moveJointWait(IMOBOT_JOINT3);
robot.moveJointWait(IMOBOT_JOINT4);
```

3.2 Demo Code for getting_started.ch Explained

The beginning of every iMobot control program will include header files. Each header file imports functions used for a number of tasks, such as printing data onto the screen or controlling the iMobot.

```
#include <stdio.h>          // Required for printf()
#include <imobotcomms.h>    // Required for iMobot control functions
```

Next, we must initialize the C++ class used to control the iMobot. This line initializes a new variable named `robot` which represents the remote iMobot module which we wish to control. This special variable is actually an instance of the `CiMobotComms` class, which contains its own set of functions called “methods” or “member functions”.

```
CiMobotComms robot;
```

The next line will use the `moveZero` member function. The `moveZero` function causes the iMobot to move all of its motors to the zero position.

```
robot.moveZero();
```

The majority of iMobot control functions do not wait for the robotic motions to complete before continuing. As such, if we want to wait for the robot to fully complete the requested motion before continuing with the rest of the program, we must use the `moveWait` function, as such.

```
robot.moveWait();
```

The next four lines of code command motors 3 and 4 to rotate 90 degrees, and then waits for the motors to stop moving.

```
robot.moveJointTo(IMOBOT_JOINT3, 90);  
robot.moveJointTo(IMOBOT_JOINT4, 90);  
robot.moveJointWait(IMOBOT_JOINT3);  
robot.moveJointWait(IMOBOT_JOINT4);
```

A iMobotComms API

The header file **libimobotcomms.h** defines all the data types, macros and function prototypes for the iMobot API library. The header file declares a class called **CiMobotComms** which contains member functions which may be used to control the robot.

Table 1: CiMobotComms Member Functions.

Function	Description
<code>CiMobotComms()</code>	The CiMobotComms constructor function. This function is called automatically and should not be called explicitly.
<code>~CiMobotComms()</code>	The CiMobotComms destructor function. This function is called automatically and should not be called explicitly.
<code>connect()</code>	Connect to a remote iMobot module. This function connects to an already-paired iMobot module in Microsoft Windows. This function does not currently work for non-Windows operating systems, such as Mac or Linux. For those operating systems, please use the <code>connectAddress()</code> function instead.
<code>connectAddress()</code>	Connect to an iMobot module by specifying its Bluetooth address.
<code>disconnect()</code>	Disconnect from an iMobot module.
<code>getJointDirection()</code> .	Gets a motor's currently assigned direction.
<code>getJointAngle()</code>	Gets a joint's angle.
<code>getJointSpeed()</code>	Gets a motor's speed.
<code>getJointState()</code>	Gets a motor's current status.
<code>isConnected()</code>	This function is used to check the connection to an iMobot.
<code>moveWait()</code>	Wait until all motors have stopped moving.
<code>pozeZero()</code>	Instructs all motors to go to their zero positions.
<code>setJointDirection()</code> .	Set the motor direction of a motor. Set to "0" for automatic direction, "1" for forward, and "2" for reverse.
<code>moveJointTo()</code>	Set the desired motor position.
<code>setJointSpeed()</code>	Sets a motor's speed.
<code>stop()</code>	Stop all currently executing motions of the iMobot.
<code>moveJointWait()</code>	Wait until the specified motor has stopped moving.
Compound Motions	These are convenience functions of commonly used compound motions.
<code>motionInchwormLeft()</code>	Inchworm gait towards the left.
<code>motionInchwormRight()</code>	Inchworm gait towards the right.
<code>motionRollBackward()</code>	Roll on the faceplates toward the backward direction.
<code>motionRollForward()</code> .	Roll on the faceplates forwards.
<code>motionStand()</code>	Stand the iMobot up on its end.
<code>motionTurnLeft()</code>	Rotate the iMobot counterclockwise.
<code>motionTurnRight()</code> ...	Rotate the iMobot clockwise.

CiRobotComms::connect()

Synopsis

```
#include <irobot.h>
int CiRobotComms::connect();
```

Purpose

Connect to a remote iRobot via Bluetooth.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function is used to connect to an iRobot. The iRobot must first be paired with the computer. This function currently only works in Microsoft Windows operating systems. For other operating systems, please use the `connectAddress()` function.

Example

Please see the example in Section 3.2 on page 9.

See Also

`connectAddress()`

CiMobotComms::connectAddress()

Synopsis

```
#include <imobot.h>
int CiMobotComms::connectAddress(const char* address, int channel);
```

Purpose

Connect to a remote iMobot via Bluetooth by specifying the specific Bluetooth address of the device.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

address The Bluetooth address of the iMobot.

channel The Bluetooth channel that the listening program is listening on. The default channel is channel 20.

Description

This function is used to connect to an iMobot.

Example

See Also

`connect()`

CiRobotComms::disconnect()

Synopsis

```
#include <irobot.h>
int CiRobotComms::disconnect();
```

Purpose

Disconnect from a remote iRobot.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function is used from disconnect to an iRobot. A call to this function is not necessary before the termination of a program. It is only necessary if another connection will be established within the same program at a later time.

Example

See Also

CiRobotComms::getJointDirection()

Synopsis

```
#include <irobot.h>
int CiRobotComms::getJointDirection(int id, int &direction);
```

Purpose

Get the speed of a joint on the iRobot.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

- `id` The joint number to pose.
- `direction` An integer variable. This variable will be overwritten with the current speed of the joint.

Description

This function is used to retrieve the motor's direction status. The valid status directions are

- 0: Automatic direction
- 1: Forward direction
- 2: Backward direction

Example

See Also

`setJointDirection()`

CiMobotComms::getJointAngle()

Synopsis

```
#include <imobot.h>
int CiMobotComms::getJointAngle(int id, double &position);
```

Purpose

Connect to a remote iMobot via Bluetooth.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

id The joint number to wait for.
positionA variable to store the current position of the iMobot motor. The contents of this variable will be overwritten with a value that represents the motor's angle in degrees.

Description

This function gets the current motor position of an iMobot's motor. The position returned is in units of degrees and is accurate to roughly ± 0.1 degrees.

Example

See Also

`connectAddress()`

CiRobotComms::getJointSpeed()

Synopsis

```
#include <irobot.h>
int CiRobotComms::getJointSpeed(int id, int &speed);
```

Purpose

Get the speed of a joint on the iRobot.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

- | | |
|--------------|---|
| id | The joint number to pose. |
| speed | The address of an unsigned short variable. This variable will be overwritten with the current speed of the joint. |

Description

This function is used to find the speed of a joint. This is the speed at which the joint will move when given motion commands. The values should be between 0 and 100.

Example

See Also

CiRobotComms::getJointState()

Synopsis

```
#include <irobot.h>
int CiRobotComms::getJointState(int id, int &state);
```

Purpose

Determine whether a motor is moving or not.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

- | | |
|--------------------|--|
| <code>id</code> | The joint number to pose. |
| <code>state</code> | An integer variable which will be overwritten with the current state of the motor. |

Description

This function is used to determine the current state of a motor. Valid states are:

- 0: The motor is idle.
- 1: The motor is moving.
- 2: The motor is heading towards a specified position.

Example

See Also

CiRobotComms::motionInchwormLeft()

Synopsis

```
#include <irobot.h>
int CiRobotComms::motionInchwormLeft();
```

Purpose

Perform the inch-worm gait to the left.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes the iRobot to perform a single cycle of the inchworm gait to the left.

See Also

`motionInchwormRight()`

CiRobotComms::motionInchwormRight()

Synopsis

```
#include <irobot.h>
int CiRobotComms::motionInchwormRight();
```

Purpose

Perform the inch-worm gait to the right.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes the iRobot to perform a single cycle of the inchworm gait to the right.

See Also

`motionInchwormLeft()`

CiRobotComms::isConnected()

Synopsis

```
#include <irobot.h>
int CiRobotComms::isConnected();
```

Purpose

Check to see if currently connected to a remote iRobot via Bluetooth.

Return Value

The function returns zero if it is not currently connected to an iRobot, or non-zero otherwise.

Parameters

None.

Description

This function is used to check if the software is currently connected to an iRobot.

Example

See Also

CiRobotComms::moveWait()

Synopsis

```
#include <imobot.h>
int CiRobotComms::moveWait();
```

Purpose

Wait for all joints to stop moving.

Return Value

The function returns 0 on success and non-zero otherwise.

Description

This function is used to wait for all joint motions to finish. Functions such as `poseJoint()` and `moveJoint()` do not wait for a joint to finish moving before continuing to allow multiple joints to move at the same time. The `moveWait()` or `moveJointWait()` functions are used to wait for robotic motions to complete.

Please note that if this function is called after a motor has been commanded to turn indefinitely, this function may never return and your program may hang.

Example

See the sample program in Section 3.2 on page 9.

See Also

`moveWait()`, `moveJointWait()`

CiRobotComms::moveZero()

Synopsis

```
#include <irobot.h>
int CiRobotComms::moveZero();
```

Purpose

Move all of the joints of an iRobot to their zero position.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function moves all of the joints of an iRobot to their zero position. Please note that this function is non-blocking and will return immediately. Use this function in conjunction with the `moveWait()` function to block until the movement completes.

Example

Please see the demo at Section 3.2 on page 9.

See Also

CiRobotComms::motionRollBackward()

Synopsis

```
#include <irobot.h>
int CiRobotComms::motionRollBackward();
```

Purpose

Use the faceplates as wheels to roll backward.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes each of the faceplates to rotate 90 degrees to roll the robot backward.

See Also

`motionRollBackward()`

CiRobotComms::motionRollForward()

Synopsis

```
#include <irobot.h>
int CiRobotComms::motionRollForward();
```

Purpose

Use the faceplates as wheels to roll forward.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes each of the faceplates to rotate 90 degrees to roll the robot forward.

See Also

`motionRollBackward()`

CiMobotComms::setJointDirection()

Synopsis

```
#include <imobot.h>
int CiMobotComms::setJointDirection(int id, int direction);
```

Purpose

Set's a motor's direction. In conjunction with `setJointSpeed()`, this function may be used to cause a motor to turn indefinitely.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

<code>id</code>	The joint number to move.
<code>direction</code>	A value indicating the desired direction.

Description

This function is used to set a motor's turn direction. Possible values for the direction are:

- 0: Automatic direction. This is the default setting.
- 1: Forward. If this value is used with a non-zero speed set using the `setJointSpeed()` function, the motor will turn forward indefinitely. : Backward. Similar to "1", except the motor will spin backward.

Example

See Also

CiMobotComms::moveJointTo()

Synopsis

```
#include <imobot.h>
int CiMobotComms::moveJointTo(int id, double position);
```

Purpose

Connect to a remote iMobot via Bluetooth.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

`id` The joint number to wait for.
`position` The absolute angle to move the motor to.

Description

This function commands the motor to move to a position specified in degrees at the current motor's speed. The current motor speed may be set with the `setJointSpeed()` member function. Please note that if the motor speed is set to zero, the motor will not move after calling the `moveJointTo()` function.

Example

Please see the example in Section 3.2 on page 9.

See Also

`connectAddress()`

CiMotorComms::setJointSpeed()

Synopsis

```
#include <imobot.h>
int CiMotorComms::setJointSpeed(int id, int speed);
```

Purpose

Get the speed of a joint on the iMotor.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

- `id` The joint number to pose.
- `speed` An unsigned short variable indicating the requested speed.

Description

This function is used to set the speed of a joint of an iMotor. Valid speed values range from 0 to 100. **Example**

See Also

CiRobotComms::motionStand()

Synopsis

```
#include <irobot.h>
int CiRobotComms::motionStand();
```

Purpose

Stand the robot up on a faceplate.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes the robot to motionStand up into the camera platform.

See Also

CiMotorComms::stop()

Synopsis

```
#include <imobot.h>
int CiMotorComms::stop();
```

Purpose

Stop all current motions on the iMotor.

Return Value

The function returns 0 on success and non-zero otherwise.

Description

This function stops all currently occurring movements on the iMotor. Internally, this function simply sets all motor speeds to zero. If it is only required to stop a single motor, use the `setJointSpeed()` function to set the motor's speed to zero.

Example

See Also

`setJointSpeed()`

CiMobotComms::motionTurnLeft()

Synopsis

```
#include <imobot.h>
int CiMobotComms::motionTurnLeft();
```

Purpose

Rotate the iMobot using the faceplates as wheels.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes the iMobot to rotate the faceplates in opposite directions to cause the robot to rotate counter-clockwise.

See Also

`motionTurnRight()`

CiRobotComms::motionTurnRight()

Synopsis

```
#include <irobot.h>
int CiRobotComms::motionTurnRight();
```

Purpose

Rotate the iRobot using the faceplates as wheels.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

None.

Description

This function causes the iRobot to rotate the faceplates in opposite directions to cause the robot to rotate clockwise.

See Also

`motionTurnLeft()`

CiMobotComms::moveJointWait()

Synopsis

```
#include <imobot.h>
int CiMobotComms::moveJointWait(int id);
```

Purpose

Wait for a joint to stop moving.

Return Value

The function returns 0 on success and non-zero otherwise.

Parameters

`id` The joint number to wait for.

Description

This function is used to wait for a joint motion to finish. Functions such as `poseJoint()` and `moveJoint()` do not wait for a joint to finish moving before continuing to allow multiple joints to move at the same time. The `moveJointWait()` or `moveJointWait()` functions are used to wait for robotic motions to complete.

Please note that if this function is called after a motor has been commanded to turn indefinitely, this function may never return and your program may hang.

Example

Please see the example in Section 3.2 on page 9.

See Also

`moveWait()`