Setup for Remote iMobot Control in Ch

David Ko
Mechanical and Aerospace Engineering

September 6, 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.

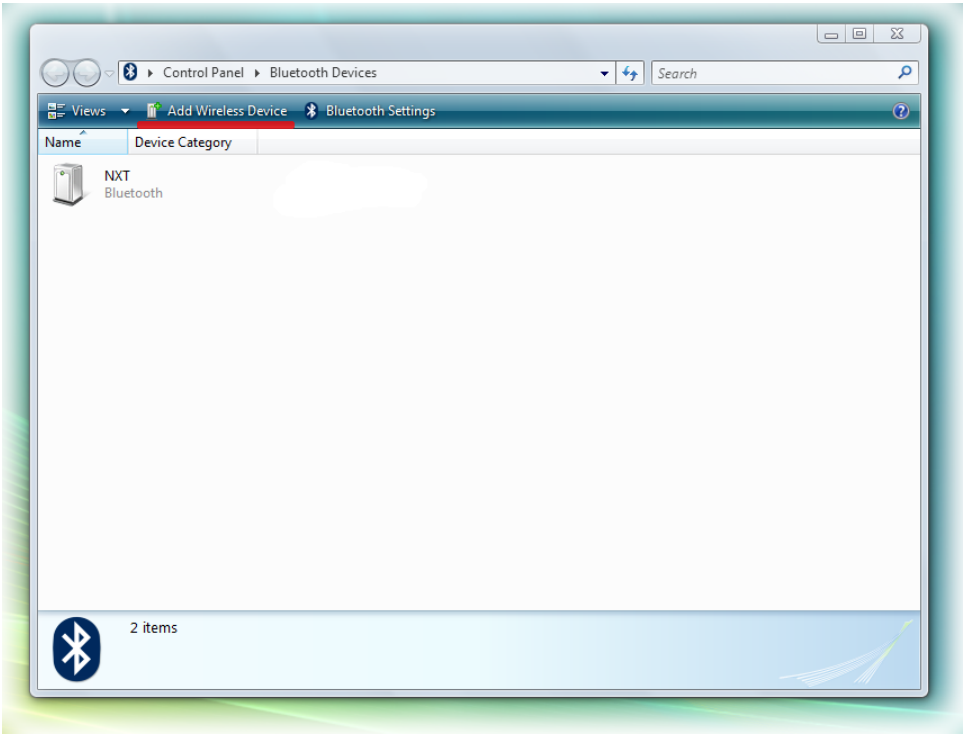
This documentation introduces the basic computer setup required for controlling the iMobot, as well as several demo programs and a complete reference for all API function provided with the `imobotcomms` library.

2 Bluetooth Pairing with the iMobot

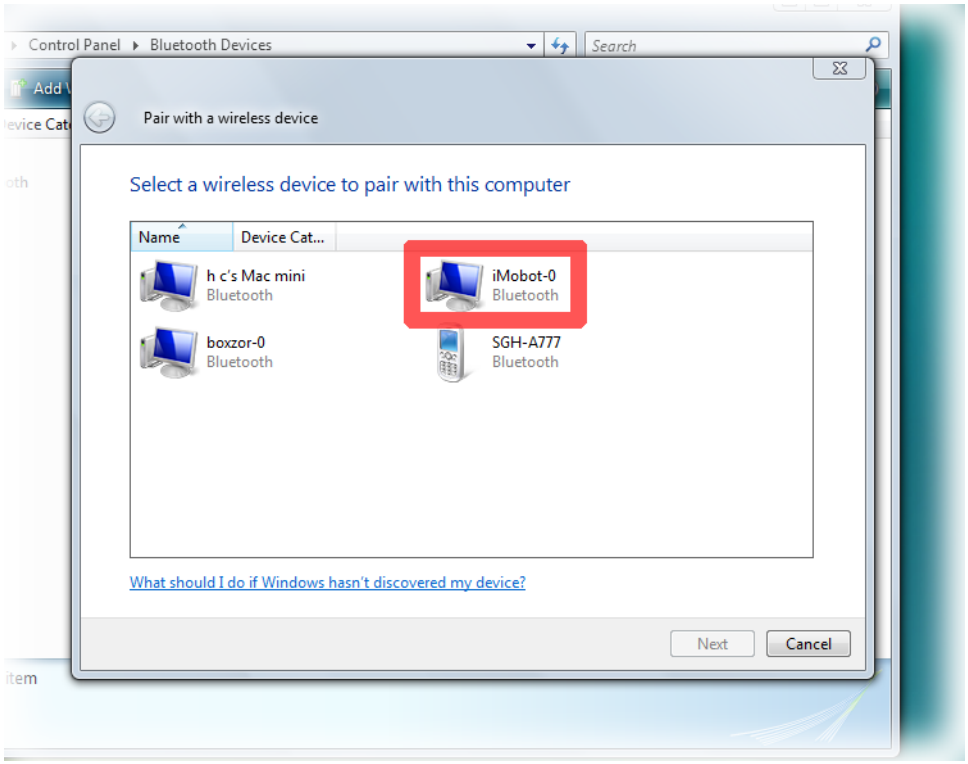
To control the iMobot with the Bluetooth wireless protocol, the controlling computer must be equipped with Bluetooth. If the computer does not have Bluetooth built-in, an external USB Bluetooth dongle may be used. The following instructions are for a Windows 7 computer with built-in Bluetooth. The basic 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.

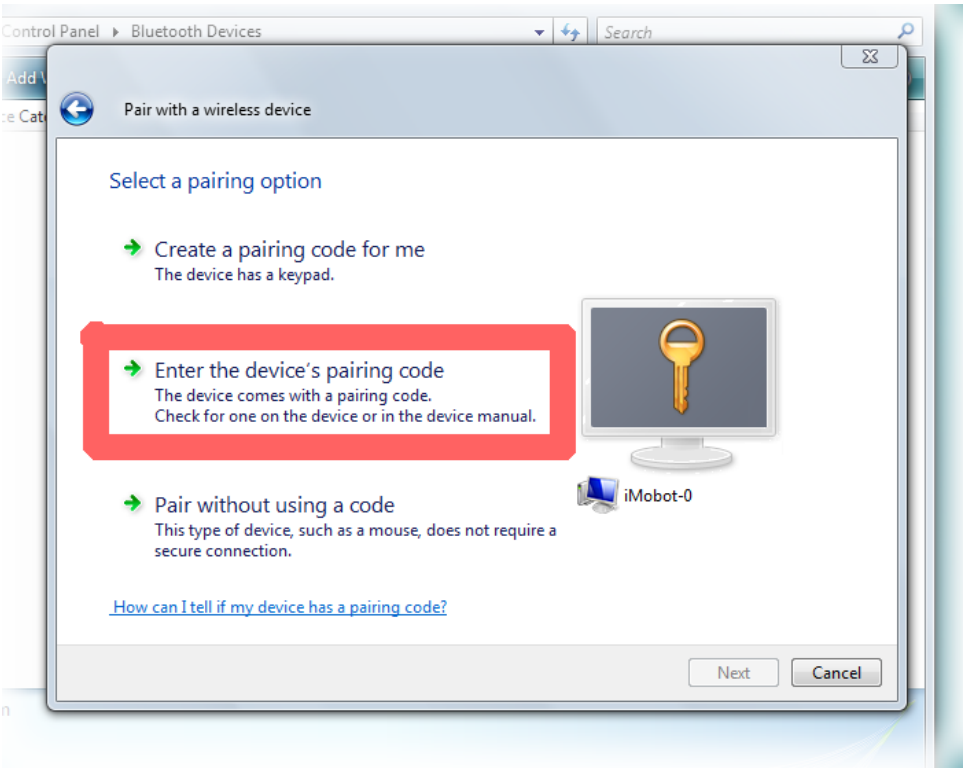
After double-clicking the icon, a new window will appear similar to the following.



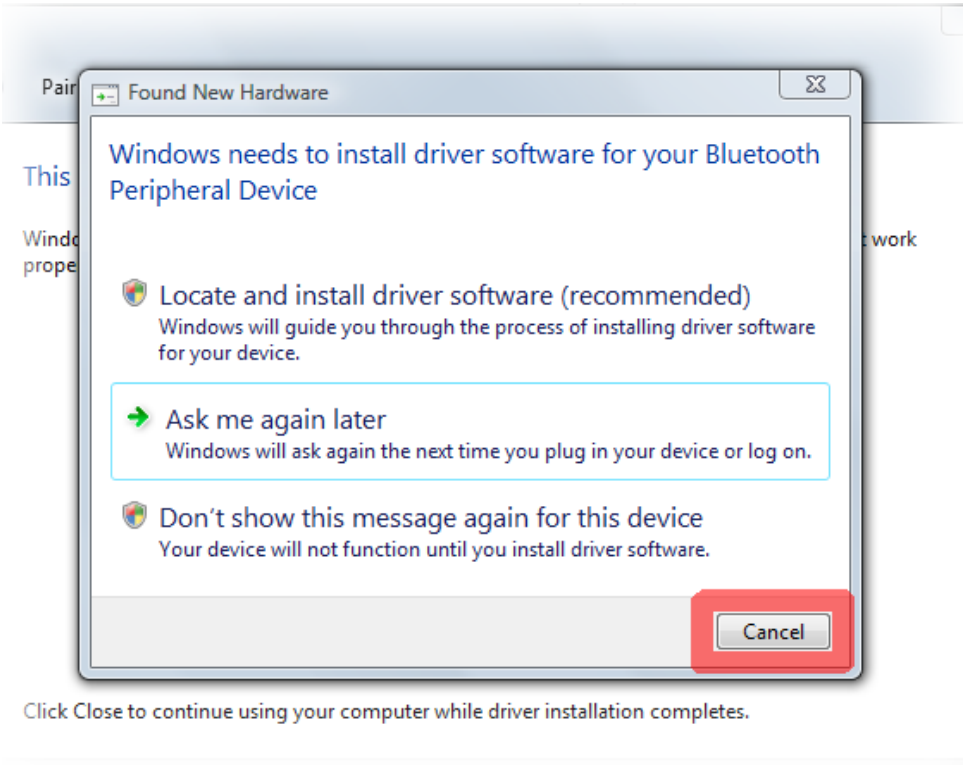
Next, click on the button labeled "Add Wireless Device" towards the top of the window. This will bring up the following dialog.



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.

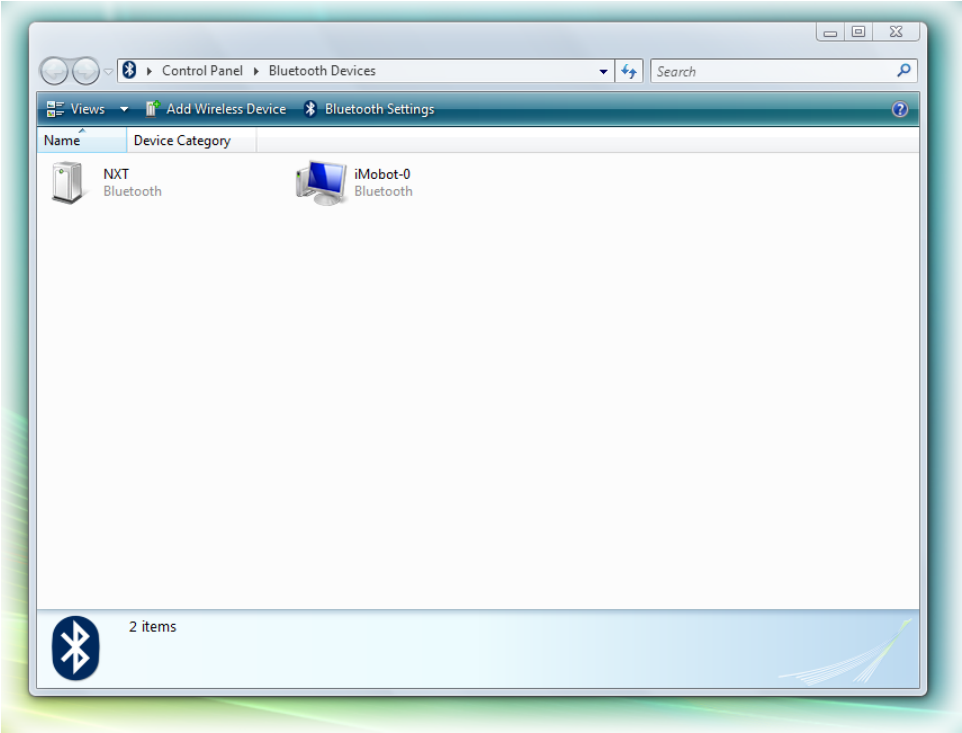


Select the second option, labeled "Enter the device's pairing code". iMobot modules come hard-coded with a default pairing code. When prompted for the pairing code, enter "1234". Once the computer is paired with the iMobot, the following dialog box may pop up asking to install drivers.

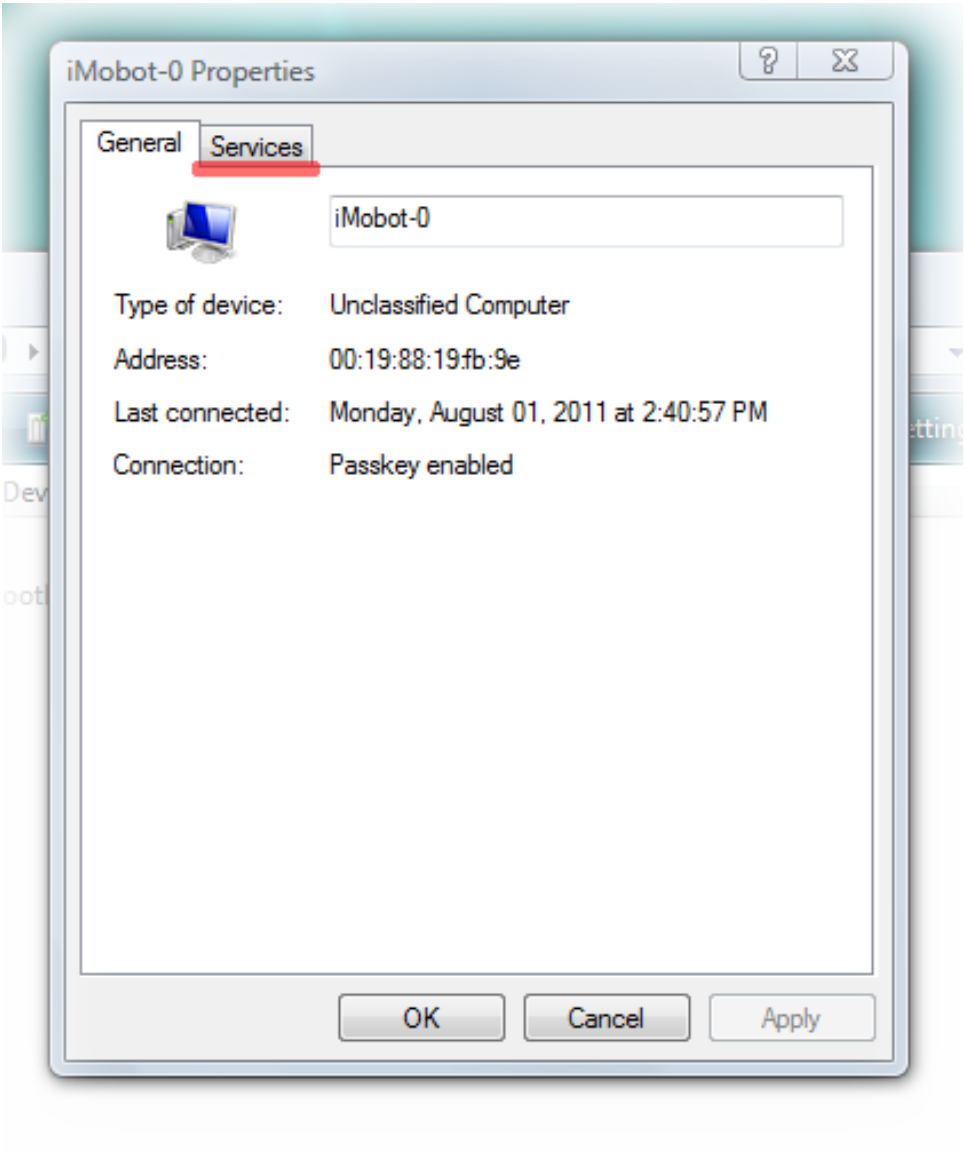


If the previously illustrated dialog box appears, just click the "cancel" button at the bottom right. No extra drivers are necessary for controlling the iMobot module.

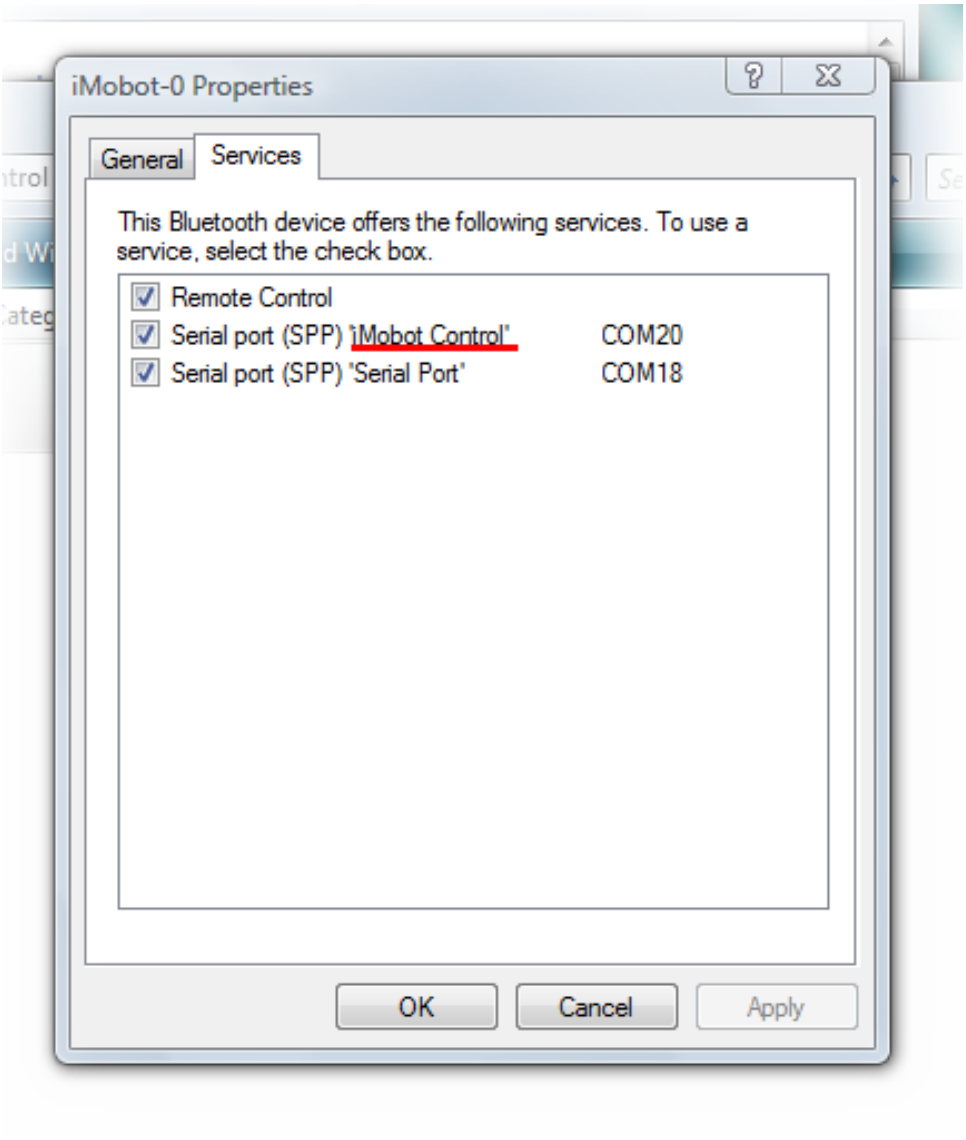
At this point, the following dialog should be shown.



The next step is to enable the iMobot control service. Double-click on the icon denoting the iMobot 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;
/* Connect to an already paired iMobot */
if(robot.connect()) {
  printf("Error connecting.\n");
}

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

/* Rotate each of the faceplates by 90 degrees */
robot.setMotorPosition(IMOBOT_MOTOR3, 90);
robot.setMotorPosition(IMOBOT_MOTOR4, 90);

/* Wait for the movement to complete */
robot.waitMotor(IMOBOT_MOTOR3);
robot.waitMotor(IMOBOT_MOTOR4);
```

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 `poseZero` member function. The `poseZero` function causes the iMobot to move all of its motors to the zero position.

```
robot.poseZero();
```

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.setMotorPosition(IMOBOT_MOTOR3, 90);
robot.setMotorPosition(IMOBOT_MOTOR4, 90);
robot.waitMotor(IMOBOT_MOTOR3);
robot.waitMotor(IMOBOT_MOTOR4);
```

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
`CiMobotComms()`	The CiMobotComms constructor function. This function is called automatically and should not be called explicitly.
`~CiMobotComms()`	The CiMobotComms destructor function. This function is called automatically and should not be called explicitly.
`connect()`	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 `connectAddress()` function instead.
`connectAddress()`	Connect to an iMobot module by specifying its Bluetooth address.
`disconnect()`	Disconnect from an iMobot module.
`getMotorDirection()` .	Gets a motor's currently assigned direction.
`getMotorPosition()` ..	Gets a joint's angle.
`getMotorSpeed()`	Gets a motor's speed.
`getMotorState()`	Gets a motor's current status.
`isConnected()`	This function is used to check the connection to an iMobot.
`moveWait()`	Wait until all motors have stopped moving.
`pozeZero()`	Instructs all motors to go to their zero positions.
`setMotorDirection()` .	Set the motor direction of a motor. Set to "0" for automatic direction, "1" for forward, and "2" for reverse.
`setMotorPosition()` ..	Set the desired motor position.
`setMotorSpeed()`	Sets a motor's speed.
`stop()`	Stop all currently executing motions of the iMobot.
`waitMotor()`	Wait until the specified motor has stopped moving.
Compound Motions	These are convenience functions of commonly used compound motions.
`inchLeft()`	Inchworm gait towards the left.
`inchRight()`	Inchworm gait towards the right.
`rollBackward()`	Roll on the faceplates toward the backward direction.
`rollForward()`	Roll on the faceplates forwards.
`stand()`	Stand the iMobot up on its end.
`turnLeft()`	Rotate the iMobot counterclockwise.
`turnRight()`	Rotate the iMobot clockwise.

CiMobotComms::connect()

Synopsis

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

Purpose
Connect to a remote iMobot 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 iMobot. The iMobot 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()`

CiMobotComms::disconnect()

Synopsis

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

Purpose
Disconnect from a remote iMobot.

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

Parameters
None.

Description
This function is used from disconnect to an iMobot. 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

CiMobotComms::getMotorDirection()

Synopsis

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

Purpose
Get the speed of a joint on the iMobot.

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
`setMotorDirection()`

CiMobotComms::getMotorPosition()

Synopsis

```
#include <imobot.h>
int CiMobotComms::getMotorPosition(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`	A 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()`

CiMobotComms::getMotorSpeed()

Synopsis

```
#include <imobot.h>
int CiMobotComms::getMotorSpeed(int id, int &speed);
```

Purpose
Get the speed of a joint on the iMobot.

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

CiMobotComms::getMotorState()

Synopsis

```
#include <imobot.h>
int CiMobotComms::getMotorState(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

`id`	The joint number to pose.
`state`	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

CiMobotComms::inchLeft()

Synopsis

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

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 iMobot to perform a single cycle of the inchworm gait to the left.

See Also
`inchRight()`

CiMobotComms::inchRight()

Synopsis

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

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 iMobot to perform a single cycle of the inchworm gait to the right.

See Also
`inchLeft()`

CiMobotComms::isConnected()

Synopsis

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

Purpose
Check to see if currently connected to a remote iMobot via Bluetooth.

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

Parameters
None.

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

Example

See Also

CiMobotComms::moveWait()

Synopsis

```
#include <imobot.h>
int CiMobotComms::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 `waitMotor()` 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(), waitMotor()`

CiMobotComms::poseZero()

Synopsis

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

Purpose
Move all of the joints of an iMobot 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 iMobot 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

CiMobotComms::rollBackward()

Synopsis

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

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
`rollBackward()`

CiMobotComms::rollForward()

Synopsis

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

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
`rollBackward()`

CiMobotComms::setMotorDirection()

Synopsis

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

Purpose
Set's a motor's direction. In conjunction with `setMotorSpeed()`, 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

`id`	The joint number to move.
`direction`	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 `setMotorSpeed()` function, the motor will turn forward indefinitely. : Backward. Similar to "1", except the motor will spin backward.

Example

See Also

CiMobotComms::setMotorPosition()

Synopsis

```
#include <imobot.h>
int CiMobotComms::setMotorPosition(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 `setMotorSpeed()` member function. Please note that if the motor speed is set to zero, the motor will not move after calling the `setMotorPosition()` function.

Example
Please see the example in Section 3.2 on page 9.

See Also
`connectAddress()`

CiMobotComms::setMotorSpeed()

Synopsis

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

Purpose
Get the speed of a joint on the iMobot.

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 iMobot. Valid speed values range from 0 to 100. **Example**

See Also

CiMobotComms::stand()

Synopsis

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

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 stand up into the camera platform.

See Also

CiMobotComms::stop()

Synopsis

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

Purpose
Stop all current motions on the iMobot.

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

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

Example

See Also
`setMotorSpeed()`

CiMobotComms::turnLeft()

Synopsis

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

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
`turnRight()`

CiMobotComms::turnRight()

Synopsis

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

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 clockwise.

See Also
`turnLeft()`

CiMobotComms::waitMotor()

Synopsis

```
#include <imobot.h>
int CiMobotComms::waitMotor(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 `waitMotor()` or `waitMotor()` 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()`