Qwerkbot Software API Quick Reference

This document is intended to give a brief overview of the methods available in the RobotClient class for controlling the Qwerkbot and associated GUI. In all sample usages, the RobotClient class is assumed to be instantiated as the "myRobot" object.

General Methods for Accessing and Controlling the GUI

Name: public boolean sleepUnlessStop(int *ms*)
Description: Causes the program to sleep for the specified number of milliseconds, unless the Play/Stop button is pressed, in which case it will exit. The method returns a boolean value; false if the method exits after time has elapsed, and true if the method exited because the Play/Stop button was pressed.
Sample Usages:

```
// Sleeps for 500 ms unless Play/Stop is pressed
myRobot.sleepUnlessStop(500);

// Sleeps for 500 ms during each loop cycle unless Stop is pressed, in which case the
// program breaks out of the while loop.
while(true) {
    if(myRobot.sleepUnlessStop (500))
    {
          break;
     }
}
```

Name: public boolean buttonState()
Description: Returns the value of the Play/Stop button – true if play/stop has been pressed, false if it has not been pressed or if it has been toggled to stop.
Sample Usages:

```
// Do the following code as long as Play has been pressed most recently
while(myRobot.buttonState())
{
 // Some code
}
```

Name: public boolean isPlaying()
Description: Returns true if the 'Play' button has most recently been pressed, false if 'Stop' has most recently been pressed. Note that the output is identical to buttonState.
Sample Usages:

```
// Do the following code as long as Play has been pressed
while(myRobot.isPlaying())
{
 // Some code
 }
```

Name: public boolean isStopped()
Description: Returns true if the 'Stop' button has most recently been pressed, false if ''Play' has most recently been pressed. Note that the output is the logical inverse of the buttonState method.
Sample Usages:
```
// If stop has been pressed, exit
while(true)
{
	if(myRobot.isStopped())
	{	break;		}
 }
```

Name: public void waitForPlay()
Description: Blocks program operation until the play button is pressed.
Sample Usages:
```
// Wait to start my actual program until play is pressed.
myRobot.waitForPlay();
/*  Rest of Program */
```

Name: public void waitForStop()
Description: Blocks program operation until the stop button is pressed.
Sample Usages:
```
// Wait for the stop button before turning off motors and such.
myRobot.waitForStop();
/*  Turn off motors, disconnect robot */
```

Name: public final void writeToTextBox(final String message)
Description: Writes the string specified by message to the GUI's textbox. Each call to this method writes the string to the next line of the textbox, and prepends a time-stamp to the string.
Sample Usages:
```
// Print hello world to the textbox
myRobot.writeToTextBox("Hello World!");
// Print the value of the variable sensor to the textbox
myRobot.writeToTextBox("The value of the rangefinder is " + sensor + ".");
```

Name: public final void clearTextBox()
Description: Clears all text written to the textbox using writeToTextBox
Sample Usages:
```
// Clear the textbox
myRobot.clearTextBox();
```

Name: public final int getTextFieldValueAsInt()
Description: Returns any value typed into the GUI's text field as an integer variable.
Sample Usages:
```
// Get the value of the text field
int textFieldValue = myRobot.getTextFieldValueAsInt();
```

Name: public final String getTextFieldValueAsString()
Description: Returns any value typed into the GUI's text field as a string
Sample Usages:
```
// Get the value of the text field as a string
String message = myRobot.getTextFieldValueAsString();
```

Methods for Controlling Motors

Name: public void moveMotor(int motorId, int velocity)
Description: Moves the motor specified by motorId to a velocity term set by velocity. Appropriate values for motorId are 0-3, corresponding to motor ports #0-3 on the Qwerk. Appropriate velocity values are -50,000 to 50,000. On a Qwerkbot, the left motor is motorId 0 and the right motor is motorId 1. To reflect this, the RobotClient class has two constants which can be used in lieu of numerical values.
Sample Usages:
```
// Move the left motor clockwise slowly
myRobot.moveMotor(0, 1000);  OR
myRobot.moveMotor(myRobot.leftMotor, 1000);

// Move the right motor counter clockwise quickly
myRobot.moveMotor(1, -25000);
```

Name: public void moveMotors(int motor0Velocity, int motor1Velocity)
public void moveMotors(int motor0Velocity, int motor1Velocity, int runTime)
Description: Allows the user to set both motor 0 and motor 1 simultaneously to different velocities. motor0Velocity and motor1Velocity have valid ranges from -50,000 to 50,000. When the method is called with the variable runTime, the motors will stop after the amount of milli-seconds specified by runTime has elapsed.
Sample Usages:
```
//  Move both left and right motors clockwise (causes the qwerkbot to spin right)
myRobot.moveMotors(20000, 20000);

// Move left motor counterclockwise and right motor clockwise (qwerbot will go straight)
myRobot.moveMotors(-15000, 15000);

// Move both motors clockwise for one second
myRobot.moveMotors(5000, 5000, 1000);
```

Name: public void stopMotors()
Description: Sets all motor velocities to 0, effectively stopping all motors.
Sample Usages:

```
// Stop motors
myRobot.stopMotors();
```

Methods for Controlling Servos

Name: public void setServo(int servoId, int position)
Description: Sets the servo specified by servoId to a position. Valid values for servoId are 0-15, corresponding to servo ports 0-15 on the Qwerk. Valid positions are 0-255.
Sample Usages:

```
//  Set servo 3 midway
myRobot.setServo(3, 127);

// Set servo 1 to one extreme
myRobot.setServo(1, 255);
```

Methods for Accessing the Analog Sensor Ports

Name: public short analog(int analogInputPortId)
Description: Returns the value of the analog sensor port specified by analogInputPortId. Valid values for analogInputPortId are 0-7, corresponding to analog ports 0-7 on the Qwerk. The returned value will be between 0 and 5000, and is the measured voltage in millivolts at the analog port.
Sample Usages:

```
// Read light sensor on port 0
short lightSense = myRobot.analog(0);
```

Name: public int batteryVoltage()
Description: Returns the value of the input battery voltage. The returned value will be between 0 and 30000, and is the measured voltage in millivolts at the analog port.
Sample Usages:

```
// Check the battery voltage
if(myRobot.batteryVoltage() < 5500)
{
        myRobot.writeToTextBox("WARNING: Battery Voltage less than 5.5Volts");
}
```

Methods for Accessing the Digital Sensor Ports

Name: public boolean digital(int digitalInputPortId)
Description: Returns the value on the digital input port specified by the digitalInputPortId. Valid values for digitalInputPortId are 0-3, corresponding to digital in ports 0-3 on the qwerk. The returned value will be true if a digital high is measured at the digital input port, while it will return false if there is a digital low.
Sample Usages:

```
// Measure the bump sensor on port 1
boolean bumpSense = myRobot.digital(1);
```

Methods for Controlling the Digital Output Ports

Name: public void setDigital(boolean state, int digitalOutputPortId)
Description: Sets the digital output port specified by digitalOutputPortId to on or off, depending on the boolean value of state. Valid values for digitalOutputPortId are 0-3, corresponding to digital in ports 0-3 on the qwerk.
Sample Usages:

```
// Turn on an attached LED on port 2
setDigital(true, 2);

// Turn off a pager motor on port 1
setDigital(false, 1);
```

Name(s):
public void setDigitalOn(int digitalOutputPortId)
public void setDigitalOff(int digitalOutputPortId)
Description: Depending on the method called, either turns the digital output specified by digitalOutputPortId on or off.
Sample Usages:

```
// Turn on an attached LED on port 2
setDigitalOn(2);

// Turn off a pager motor on port 1
setDigitalOff(1);
```

Methods for Controlling Onboard LEDs

Name(s):
public void setLEDOn(int ledId)
public void setLEDOff(int ledId)
public void setLEDBlinking(int ledId)
Description: Sets the LED specified by ledId to either on, off, or blinking depending on the method used. The valid range of ledId is 0-9, corresponding to the 10 green LEDs at the front of the Qwerk board labeled 0-9.
Sample Usages:

```
// Turn on LED 3
setLEDOn(3);

// Set LED 8 to blink
setLEDBlinking(8);
```

Methods for Controlling the External Speaker

Name: public void playSound(String filePath)
Description: Play the audio file specified by the String filePath. Note that only .wav files can be played at this time!
Sample Usages:

```
// Play the national anthem located in C:\MyMusic\NationalAnthem.wav
myRobot.playSound("C:/MyMusic/NationalAnthem.wav");
```

Name: public void saySomething(String text)
Description: Synthesizes the String 'text' into speech, and then plays that speech over the robot's speaker.
Sample Usages:

```
// Say that I am a silly robot
myRobot.saySomething("You are a silly robot.");

// Say the current state of analog port 3
myRobot.saySomething("The value of analog port 3 is " + myRobot.analog(3));
```

Name: public void playTone(int frequency, int amplitude, int duration)
Description: Play the tone at a given frequency and amplitude, and for a time specified by duration. Duration is in milli-seconds, so a duration of 1000 leads to a 1 second long tone. Amplitude determines the loudness of the tone, with a valid range up to 10,000. Frequency determines tone pitch, with higher frequency making for a higher pitch. For reference, the tones on a telephone dial are generally between 500 and 1000 Hz.
Sample Usages:

```
// Play a A4 note for 1.5 seconds at a loud volume
myRobot.playTone(440, 10000, 1500);
```