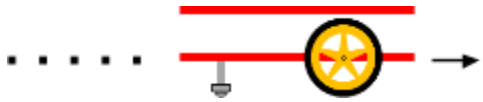
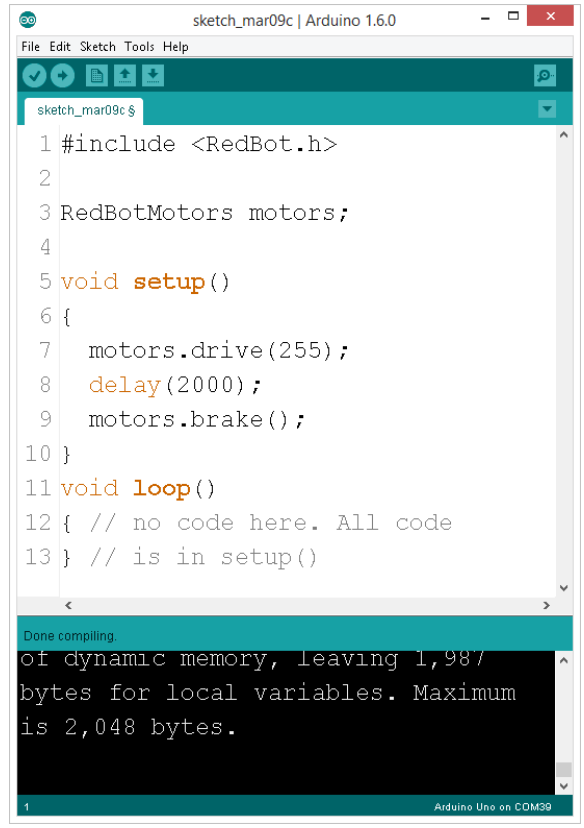
Name: _____ Date: _____

Characterizing `motors.drive(255)`



Introduction



The RedBot library has a number of pre-written functions and commands that make controlling the robot easier. One of these is the `RedBotMotors` object. The first line of code reads:

```
#include <RedBot.h>
```

This incorporates additional code into your program in the background and provides access to new functions and commands. Type this into your window. Make sure you pay attention to case (i.e. upper vs. lower case).

In second line of code, we create a `RedBotMotors` object called `motors`. This is done with the following line:

```
RedBotMotors motors;
```

Here, pay attention to both the case and the semicolon ";" at the end of the line.

Now, we're ready to start using the library. In the `setup()` function, you'll see two commands that we can use to control the speed of the robot.

```
motors.drive(255);
motors.brake();
```

The `motors.drive()` command drives both the left and right motors in the correct direction so that the robot drives forward. The 255 in between the parentheses is called an input parameter. It indicates the `drivePower` -- it is any integer value from 0 to 255.

The `motors.brake()` command stops the motors abruptly. Notice that the `motors.brake()` command does not have a parameter. You still need to have the parentheses (), though. This is what indicates that it is a command (also called a function).

Finally, you'll see a command in between the two motors commands:

```
delay(2000);
```

As implied, this command delays the program flow for 2000 time units. In this case 1 time unit is a millisecond. The Arduino microcontroller runs at **16 MHz**. So, if we didn't have a delay, it would drive the motors and then immediately stop the motors in less than a few microseconds!

Before uploading code, things to watch out for:

- Make sure that the XBEE_HW_SERIAL switch is in the down position pointed toward XBEE SW SERIAL.
- Make sure that the MOTOR switch is set to STOP (just so that it doesn't run off the table.
- The POWER must be set to ON.
- Motors are, in fact, plugged into the correct ports.

Now, click on the upload icon: After a few seconds, you should see the TX / RX LED lights on the RedBot Mainboard flash in rapid succession. This is the program being uploaded from the computer to your robot!

Un-plug the RedBot, set it down on the floor next to a meter stick.

Flip the **MOTOR** switch to **RUN**, and push the **RESET** button. Measure how far the robot drives.

Testing and Experimentation

Now that you know how to get the robot to drive, let's investigate! How does the `drivePower` relate to "real" measurements of speed / velocity? motor.drive() takes a parameter from 0 - 255. Come up with a relationship (equation) between `drivePower` and the actual speed of the robot.

Task: Devise an experiment to find a relationship between the drivePower (0 - 255) and the speed of your robot.

Be sure to start with a data table. What variables are you going to change? What will be your independent variable and what will be your dependant variable?

Analysis / Discussion

What is the relationship between robot speed and `drivePower`? Write out the equation with all of the proper units and replace the generic y=mx+b with meaningful variable names. What is the significance of the graph? What does the shape of the graph imply about this relationship?

Going Further

Now that you have a relationship for the way the robot behaves going forward, what about going backward. Is it symmetric? Try `drivePower` values from (0 to -255). This will cause the robot to drive backwards.

Qualitative Observations

As you test the robot, make notes about how the robot behaves. Does it always drive straight? Does it have a tendency to turn in one direction vs. another? What does that imply about the right motor vs. the left motor?

When you start off at full power, watch the wheels closely. What do you notice?