Full Throttle STEM™ at Eldora Speedway: Telemetric System

Challenge Problem and Resources

Developed by:

The teachers, students, and mentors in the
Gaming Research Integration for Learning Laboratory™ (GRILL™)
Summer 2013

TABLE OF CONTENTS

1. CHALLENGE PROBLEM: TELEMETRIC SYSTEM

Use a commercial off-the-shelf (COTS) computer based racing game and sensor technology to create a telemetric system that will import real time data to a virtual racing game. Telemetry is a key factor in modern motor sports, allowing race engineers to interpret data collection during a test or race to tune the car properly for optimal racing. Measurements on the racecar may include accelerations in three axes, temperature readings, wheel speed, GPS, and suspension displacement. Two-way telemetry also exists for motor sports. While pit crews can receive data about the car and the driver, the two-way telemetry also allows the pit crew to tune the engine while the car is racing on the track to change the car set-up.

1.1. THE TOOLS

Use tools such as X-Motor Racing (COTS game), sensor technology, computer scripting or coding and simple hardware.

1.2. THE CHALLENGE

Solutions to this challenge problem should include the creation of a telemetric system allowing the transmission of valuable data such as GPS, acceleration, temperature readings, wheel speed, and other pertinent data from a live race and integrate it into X-Motor Racing.

2. TUTORIALS

Wright Scholars, in collaboration with educators and the GRILL™ team, created the tutorials described below as possible solutions to solve the challenge problem. At the time of creation these were working tutorials; however, with software updates and changes in technology, additional steps may be required. Teachers are encouraged to communicate any issues, problems, or suggested changes to these tutorials to ensure the dissemination of helpful materials to support challenge problem implementation.

2.1. CONFIGURING XBEES AND WIRING AN RC CAR TO AN ARDUINO

This tutorial documents how to connect an RC car to microcontrollers enabling the car to operate through a virtual environment like Unity. There is a significant amount of coding required. In some cases, the code we have provided is in a separate text file.

Suggested materials for this tutorial include two XBees (S2 recommended; must be configured to communicate with each other), 2 XBee dongles, 1 Arduino board, and sufficient multicolored wires.

In order to ensure success for students and teachers to the greatest extent possible, we collected the supplemental files including libraries for use when tackling challenge problems. Please refer to the Full Throttle STEM™ At Eldora Speedway Implementation Guide.

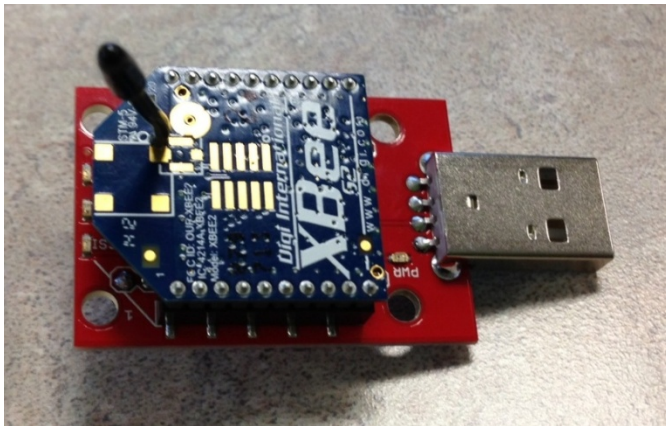
This tutorial includes the following topics:

- XBEE Configuration (Section 2.1.1)
- Arduino Programming(Section 2.1.2)
- Wiring the RC car to the Arduino and XBEE Radio (Section 2.1.3)

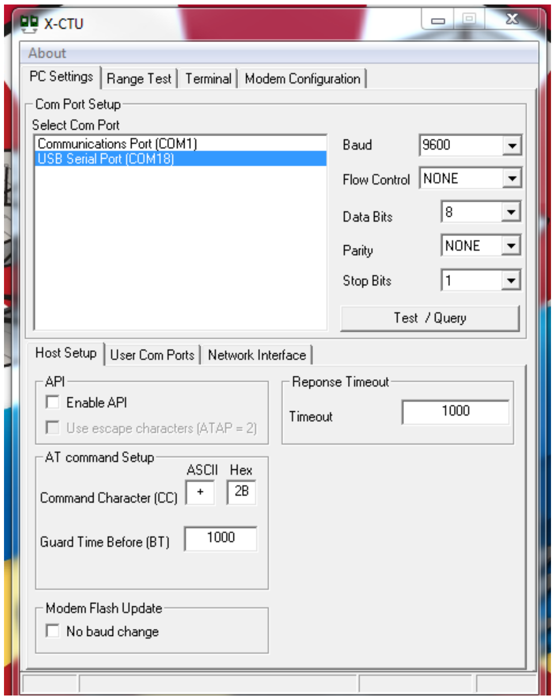
2.1.1. XBEE CONFIGURATION

Download and install the latest version of X-CTU Software from the following website: http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilities.

1. Choose an installer under the General Diagnostics, Utilities, and MIBs tab that will work best for your computer.
2. Insert one of the XBees into its dongle and connect it to the USB Port on your computer. This XBee will be the Router.



3. Open X-CTU Software.
4. Select a port from the Select Com Port list. It will not be Com 1.


X-CTU

About

PC Settings | Range Test | Terminal | Modem Configuration

Com Port Setup

Select Com Port

Communications Port (COM1)
USB Serial Port (COM18)

Baud 9600
Flow Control NONE
Data Bits 8
Parity NONE
Stop Bits 1

Test / Query

Host Setup | User Com Ports | Network Interface

API
Enable API
Use escape characters (ATAP = 2)

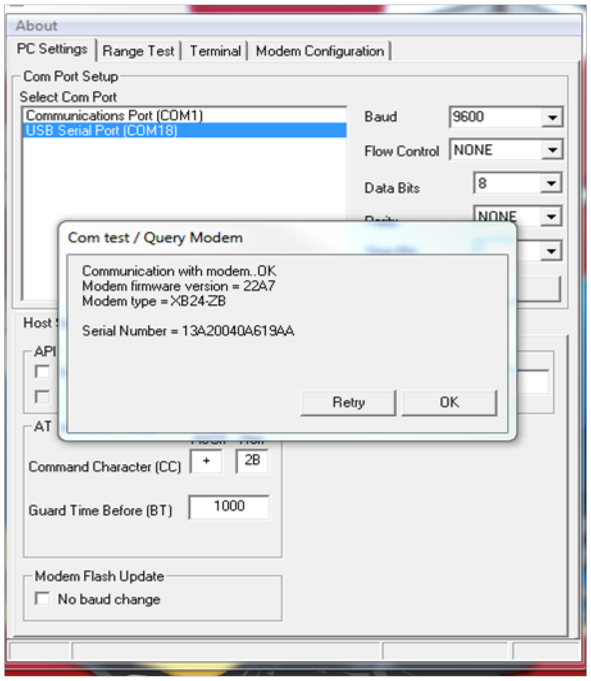
Reponse Timeout
Timeout 1000

AT command Setup

	ASCII	Hex
Command Character (CC)	+	2B
Guard Time Before (BT)	1000	

Modem Flash Update
No baud change

5. Click the Test/Query button. If successful, the window will appear similar to the one shown below, left.



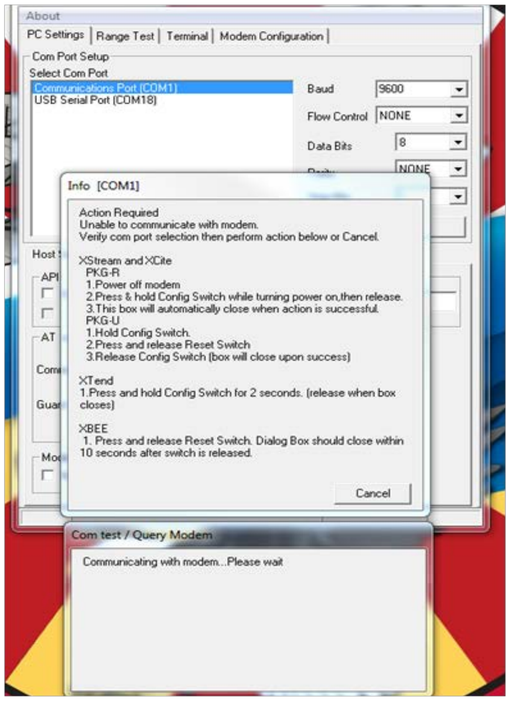
Com Port Setup
Select Com Port
Communications Port (COM1)
USB Serial Port (COM18)

Baud 9600
Flow Control NONE
Data Bits 8
Parity NONE

Com test / Query Modem

```
Communication with modem..OK
Modem firmware version = 22A7
Modem type = XB24-ZB

Serial Number = 13A20040A619AA
```

Command Character (CC) + 2B
Guard Time Before (BT) 1000
Modem Flash Update
No baud change



Com Port Setup
Select Com Port
Communications Port (COM1)
USB Serial Port (COM18)

Baud 9600
Flow Control NONE
Data Bits 8
Parity NONE

Info [COM1]

```
Action Required
Unable to communicate with modem.
Verify com port selection then perform action below or Cancel.

XStream and XCite
  PKG-R
  1.Power off modem
  2.Press & hold Config Switch while turning power on,then release.
  3.This box will automatically close when action is successful.
  PKG-U
  1.Hold Config Switch.
  2.Press and release Reset Switch
  3.Release Config Switch (box will close upon success)

XTend
1.Press and hold Config Switch for 2 seconds. (release when box
closes)

XBEE
 1. Press and release Reset Switch. Dialog Box should close within
10 seconds after switch is released.
```

Com test / Query Modem

```
Communicating with modem...Please wait
```

Successful!

Unsuccessful!

If unsuccessful, the window will appear similar to the one shown above, right.

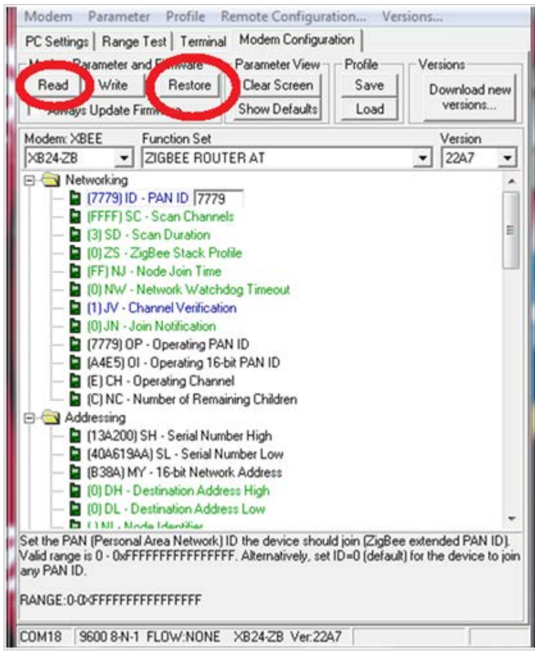
6. If your attempt is unsuccessful, unplug and Replug the dongle into the computer and close and reopen the X-CTU window. If the problem persists, there is an issue with the connection between the XBee and dongle or the dongle and your computer.

Note: Do not plug other devices into the USB ports so that you can easily identify the XBee's port name.

7. Click on the Modem Configuration tab.
8. Click Download New Versions… to install updates for the X-CTU software and click Done.
9. Plug in the second XBee and dongle and open a new X-CTU window. This XBee will be the Coordinator. Make sure to leave the Router XBee and its X-CTU window alone.
10. Select a port for the Coordinator Xbee from the Select Com Port list. Be sure to select the new com port this time and not the router's com port.
11. Click the Test/Query button.
12. In the Router X-CTU window:
    a. Click the Modem Configuration tab.
    b. Click the Read button.
    c. Click the Restore button.

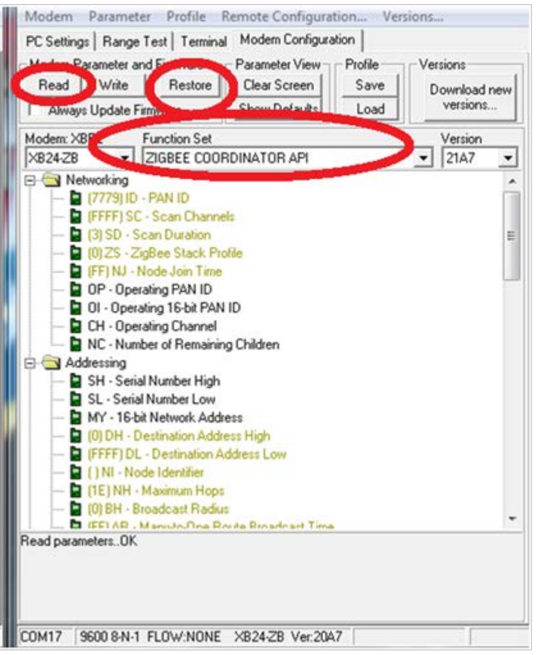
   d. Click the Read button again.
   e. Inside the Networking folder, set the PAN ID to an arbitrary number by clicking on it and typing in the number.
   f. Change the Channel Verification to 1-ENABLED.
   g. Inside the Addressing folder, set the Destination Address High value to 0.
   h. Click the Write button.

13. In the Coordinator X-CTU window:
   a. Click the Modem Configuration tab.
   b. Click the Read button.
   c. Click the Restore button.
   d. Click the Read button again.
   e. Change the Function Set from Zigbee Coordinator API to Zigbee Coordinator AT.
   f. Set this PAN ID to the same arbitrary number used in the Router window.
   g. Click the Write button.



```
Modem: XBEE    Function Set            Version
XB24-ZB        ZIGBEE ROUTER AT        22A7
Networking
  (7779) ID - PAN ID 7779
  (FFFF) SC - Scan Channels
  (3) SD - Scan Duration
  (0) ZS - ZigBee Stack Profile
  (FF) NJ - Node Join Time
  (0) NW - Network Watchdog Timeout
  (1) JV - Channel Verification
  (0) JN - Join Notification
  (7779) OP - Operating PAN ID
  (A4E5) OI - Operating 16-bit PAN ID
  (E) CH - Operating Channel
  (C) NC - Number of Remaining Children
Addressing
  (13A200) SH - Serial Number High
  (40A619AA) SL - Serial Number Low
  (B38A) MY - 16-bit Network Address
  (0) DH - Destination Address High
  (0) DL - Destination Address Low
Set the PAN (Personal Area Network) ID the device should join (ZigBee extended PAN ID).
Valid range is 0 - 0xFFFFFFFFFFFFFFFF. Alternatively, set ID=0 (default) for the device to join
any PAN ID.

RANGE:0-0XFFFFFFFFFFFFFFFF
COM18  9600 8-N-1  FLOW:NONE  XB24-ZB  Ver:22A7
```

Router

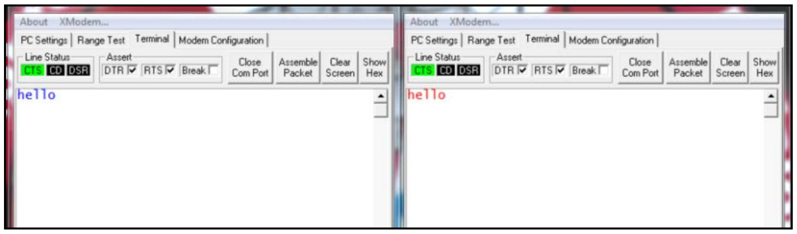


```
Modem: XBEE    Function Set            Version
XB24-ZB        ZIGBEE COORDINATOR API  21A7
Networking
  (7779) ID - PAN ID
  (FFFF) SC - Scan Channels
  (3) SD - Scan Duration
  (0) ZS - ZigBee Stack Profile
  (FF) NJ - Node Join Time
  OP - Operating PAN ID
  OI - Operating 16-bit PAN ID
  CH - Operating Channel
  NC - Number of Remaining Children
Addressing
  SH - Serial Number High
  SL - Serial Number Low
  MY - 16-bit Network Address
  (0) DH - Destination Address High
  (FFFF) DL - Destination Address Low
  ( ) NI - Node Identifier
  (1E) NH - Maximum Hops
  (0) BH - Broadcast Radius
Read parameters..OK
COM17  9600 8-N-1  FLOW:NONE  XB24-ZB  Ver:20A7
```

Coordinator

14. Click the Terminal tab in both X-CTU windows to verify that your XBees are communicating properly. Type a message in the Router window (do not press enter). The Coordinator

window should display the same text.



```
hello
```

```
hello
```

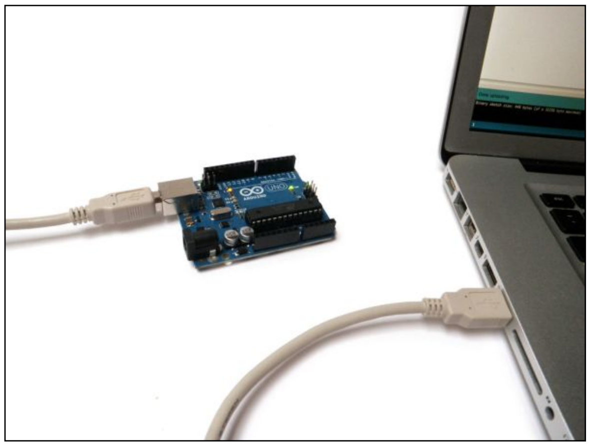
Note: If you are having trouble with XBee configuration, click the PC settings tab and set baud rate to the same number— usually 9600. In addition, you can watch the following XBee tutorials for more information:

- http://www.youtube.com/watch?v=mPx3TjzvE9U
- http://www.loveelectronics.co.uk/Tutorials/7/xbee-tutorial-how-to-set-up-your-xbees

2.1.2. ARDUINO PROGRAMMING

The microprocessor on the Arduino board is programmed using Arduino programming language and the Arduino development environment. You will need to download the software and upload the necessary code.

1. Download and install the latest version of Arduino software using the following link: http://arduino.cc/en/Main/Software
   Note: Choose a download that will work best for your computer.
2. Attach the Arduino board to the computer by connecting the USB adapter for the Arduino.



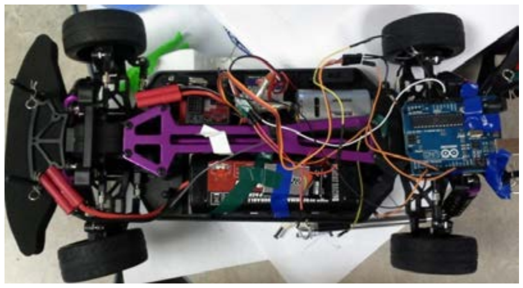
3. Launch the Arduino software, opening a window called a sketch.
4. Copy the following code into a new Arduino window. You can also find this code in the file Arduino_Xbee.txt.
   Note: This program will be for the Arduino attached to the RC car.

```
//Import libraries
```

```
#include <Servo.h>
String motorData="";
Servo servox;
Servo servoy;

//Set Initial Speed and turn to zero
int motorSpeed =90;
int motorTurn =90;

void setup(){
 Serial.begin(9600);
 servox.attach(9);
  servoy.attach(6);
}

void loop(){
 //If there is new data
 if(Serial.available()){
  int i = Serial.read();
  char c = i;
   motorData= motorData+c;
 }

 //keep motor moving at current speed and turn
 moveMotor(motorSpeed,motorTurn);

 //If the new data is correct
 if(motorData.substring(0,1)!="@"){
  motorData="";
 }

 //Once data is long enough, begin to change motor
 if(motorData.length()>7){
  //Serial.println("Step 1:"+motorData);
   checkEnding();
 }
}

//To make sure data was entered in correctly
void checkEnding(){
  if(motorData.substring(7)=":"){
    //Serial.println("Step 2:"+motorData);
    motorSpeed =StringToInt(motorData.substring(1,3));
    motorTurn =StringToInt(motorData.substring(5,8));
    //  Serial.print("Step 3:");
    //  Serial.print(motorSpeed);
    //  Serial.print(" ");
    //  Serial.println(motorTurn);
    moveMotor(motorSpeed,motorTurn);
    motorData="";
  }else{
   motorData="";
  }
}

//adjust speed and turn
void moveMotor(int Mspeed, int Mturn){
  servox.write(Mspeed);
  servoy.write(Mturn);
}

//convert and integer to a string.
```

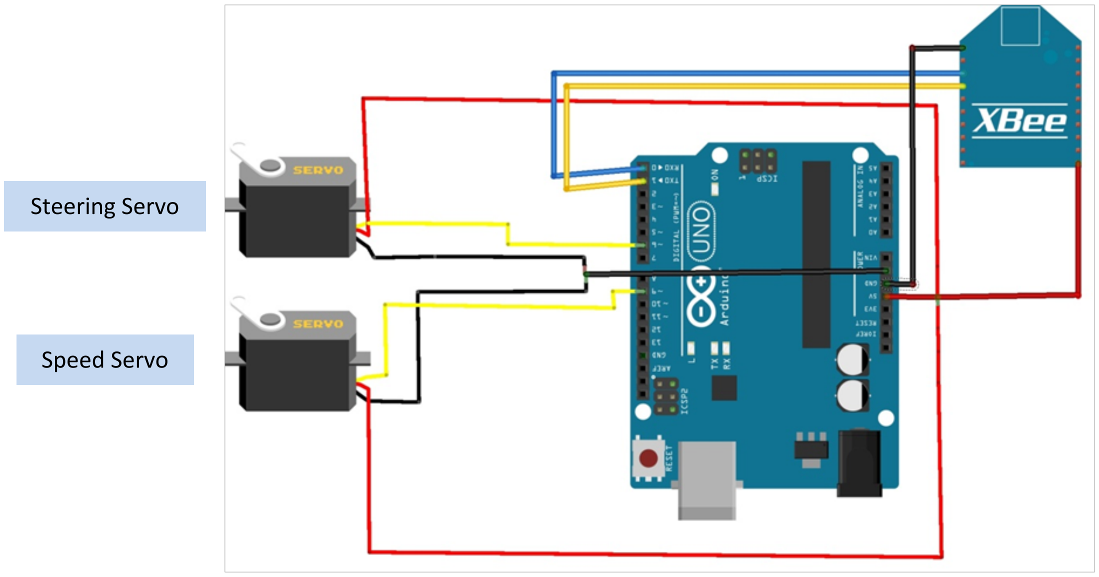
```
int StringToInt(String s){
  int solution =0;
  int counter = 2;
  int multiplier=1;
  while(counter>=0){
    if(s.substring(counter,counter+1)=="1"){
      solution= solution+multiplier*1;
    }else if(s.substring(counter,counter+1)=="2"){
      solution= solution+multiplier*2;
    }else if(s.substring(counter,counter+1)=="3"){
      solution= solution+multiplier*3;
    }else if(s.substring(counter,counter+1)=="4"){
      solution= solution+multiplier*4;
    }else if(s.substring(counter,counter+1)=="5"){
      solution= solution+multiplier*5;
    }else if(s.substring(counter,counter+1)=="6"){
      solution= solution+multiplier*6;
    }else if(s.substring(counter,counter+1)=="7"){
      solution= solution+multiplier*7;
    }else if(s.substring(counter,counter+1)=="8"){
      solution= solution+multiplier*8;
    }else if(s.substring(counter,counter+1)=="9"){
      solution= solution+multiplier*9;
    }
    multiplier = multiplier*10;
    counter--;
  }
  return solution;
}
```

2.1.3. WIRING THE RC CAR TO THE ARDUINO AND XBEE RADIO



Once you have the Arduino programmed, you will then need to wire the car with the rest of the hardware. Follow the Fritzing © schematic below as closely as possible. Please carefully read the following notes to wire the devices successfully:

- The wire colors in the diagram may not exactly match your model.
- Many of the wires may need extension wires added to allow the wires to span the distance from their origin to the Arduino board.
- The three red wires, one from each servo and one from the XBee, will need to be connected together with an additional wire; the additional wire will connect to the 5 volt pin on the arduino.
- The two black wires, one from each servo, will need to be connected together with an additional wire; the additional wire will connect to the ground pin.



Fritzing Schematic

To wire the XBee Dongle:

1. Connect the red wire to the 5-volt (5v) pin on the Arduino.
2. Use the black wire to connect the XBee ground pin to the ground (GND) pin on the Arduino.
3. Connect the blue wire to the XBee DOUT pin to the RX or 0 pin on the Arduino.
4. Connect the yellow wire to the XBee DIN pin to the TX or 1 pin on the Arduino.

Note: You may need to solder wires into the XBee dongle to secure them. Do not solder the XBee dongle with the XBee in it.

The RC car has two servo motors. Each servo motor has three wires coming out of it. One servo controls steering and the other controls speed. The servo attached to the steering mechanism (which has red, brown, and orange wires attached to it) is the top servo motor that attaches to pin 6. The servo attached to the speed mechanism (which has white, black, and red wires coming out of it) is the bottom servo motor that attaches to pin 9. (Depending on the RC car that is being used, the colors of the wires may change.)

To wire the top servo (steering control) motor:

1. Connect the red wire to the 5-volt (5v) pin on the Arduino.
2. Connect the brown wire to the ground (GND) pin on the Arduino.
   Note: The brown wire on the servo motor may be black.
3. Connect the orange wire to the 6 digital pin on the Arduino.

Note: The orange wire on the servo motor may be yellow or white.

To wire the bottom servo (speed control) motor:

4. Connect the red wire to the 5-volt (5v) pin on the Arduino.
5. Connect the black wire to the ground (GND) pin on the Arduino.
6. Connect the white wire to the 9-digital pin on the Arduino.

Note: The white wire on the servo motor may be yellow.