Full Throttle STEM™ Rover: Travel with Autonomous Obstacle Avoidance

Challenge Problem and Resources

Developed by:

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

1. CHALLENGE PROBLEM: ROVER TRAVEL WITH AUTONOMOUS OBSTACLE AVOIDANCE .......................... 3
   1.1. THE TOOLS .......................................................................................................................... 3
   1.2. THE CHALLENGE .................................................................................................................. 3

2. TUTORIALS .................................................................................................................................. 4
   2.1. AUTONOMOUS OBSTACLE AVOIDANCE .......................................................................... 4
   2.1.1. Uploading the NewPing library 5.6.1 ........................................................................... 4
   2.1.2. Connecting the Ultrasonic SENSORS ........................................................................... 5
   2.1.3. Wiring the DC Motors .................................................................................................. 6
   2.1.4. Programming the Arduino ............................................................................................ 7
   2.1.5. Increasing Rover Intelligence ....................................................................................... 10

1. CHALLENGE PROBLEM: ROVER TRAVEL WITH AUTONOMOUS OBSTACLE AVOIDANCE

The rover will have sensors that allow it to be autonomous, avoiding obstacles in its way as it attempts to drive forward.

1.1. THE TOOLS

Students will need a rover, an Arduino board, ping sensors, and the Arduino IDE.

1.2. THE CHALLENGE

Ping sensors will provide information to the Arduino that will direct the rover to make appropriate turns before moving forward.
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. AUTONOMOUS OBSTACLE AVOIDANCE

Utilizing Arduino programming and ultrasonic distance sensors, a vehicle can reverse or change direction when it detects an object to be a certain distance away from the sensor. This tutorial explains the basic setup and describes the process of using ping sensors to enable the rover to be an autonomous vehicle. It has pictures of wiring from the Arduino to the breadboard on the rover. This section also includes a copy of the code for activating the ping sensors and using them to influence the movement of the motor.

Suggested tools and materials for this challenge problem include Arduino Uno, Arduino Motor Shield, Ping HC-SR04 sensors (two), Breadboard, Several Wires (approximately 10-15), USB Cable, Assembled Rover Kit, Arduino IDE software, and the NewPing library, which is available on this webpage (http://playground.arduino.cc/Code/NewPing).

This tutorial includes the following topics:

- Installing the NewPing library (Section 2.1.1)
- Arduino hardware (2.1.2)
- Wiring the DC Motors (2.1.3)
- Sample Code (2.1.4)
- Increased Rover Intelligence (2.1.5)

2.1.1. UPLOADING THE NEWPING LIBRARY 5.6.1

You will need to use the sample code provided in the NewPing Library to run the ping sensors.

To upload the library into the Arduino IDE:
1. Save the NewPing folder to any location.
2. Open the Sketch tab in the new Ardunio sketch window.
3. Select Import Library.
4. Go to the location where you saved the NewPing folder and select open.
5. To use this library in code, it is necessary to import it at the beginning of the code.
   a. Select Sketch.
   b. Select Import Libraries.
   c. Select NewPing.

2.1.2. CONNECTING THE ULTRASONIC SENSORS

The ultrasonic sensor uses sonar to determine distance to an object, similar to the activity of bats and dolphins. You must connect the ultrasonic sensors to Arduino Uno, as illustrated below, prior to programming it. Once connected, you can upload the code to make the rover reverse direction when it detects an obstacle. Two examples of code are included in Section (2.1.4).

Note: There is an Arduino motor shield on top of the Arduino Uno.
2.1.3. WIRING THE DC MOTORS

The only major design requirement is that the rover must have two DC motors. You must wire these motors to the Arduino motor shield ports A and B. These ports are on the corner opposite the reset button on the motor shield. By doing this, the Arduino board and its program can control the motors.

The rover pictured below was built with a robotics kit purchased from the Robot Shop. Notice that the ping sensors are mounted above the rest of the base plate, allowing for unobstructed “sight” when operational. The rover in this picture has three ping sensors and has an XBee shield.
2.1.4. PROGRAMMING THE ARDUINO

The code provided in this section enables the rover to reverse direction when it detects an obstacle.

**Note:** The NewPing library (tutorial included above) must be installed properly and the Ping sensors must be connected correctly for the code to work.
To program the Arduino:

1. Open the Arduino program on your computer. If Arduino is not loaded, go to http://arduino.cc/en/Main/Software and download the version that suits your computer (i.e. Mac or PC, 32 bit or 64 bit, etc.). Follow the instructions for downloading and installation.

2. Once you open Arduino, insert the following code into the code field. You can also find this code in the file Arduino_Ping1.txt (see supplemental files).

```c
#include <NewPing.h>

boolean forward;

#define TRIGGER_PIN 7 // Arduino pin 7 tied to trigger pin on the front ultrasonic sensor.
#define ECHO_PIN 6 // Arduino pin 6 tied to echo pin on the front ultrasonic sensor.
#define MAX_DISTANCE 200 // Max dist we want to ping for, in cm. Max sensor dist is rated at 400-500cm.

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing setup of pins and maximum distance.
NewPing sonar2(4, 5, MAX_DISTANCE); // pins assigned to trigger and echo of rear ultrasonic sensor

void setup() {
  Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
  pinMode(13, OUTPUT);
  pinMode(11, OUTPUT);
  forward = false;
  digitalWrite(11, HIGH);
}

void loop() {
  delay(50);
  // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
  unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
  Serial.println("Ping: ");
  int front = uS / US_ROUNDTRIP_CM;
  // Convert ping time to distance in cm and print result (0 = outside set distance range)
  Serial.print(front);
  Serial.print(" cm");
  Serial.print(" ");
  delay(50);
}
```
unsigned int uS2 = sonar2.ping();  // Send ping, get ping time in microseconds (uS).

int back = uS2 / US_ROUNDTRIP_CM;

// Convert ping time to distance in cm and print result (0 = outside set distance range)
Serial.print(back);
Serial.println("cm");

if (forward) {
    digitalWrite(13, HIGH);
    if (front < 10)
        flipSwitch();
}

if (!forward) {
    digitalWrite(13, LOW);
    if (back < 10)
        flipSwitch();
}

void flipSwitch() {
  switch(forward) {
    case true:
      forward = false;
      break;
    case false:
      forward = true;
      break;
    default:
      break;
  }
}
3. Press the Verify button (blue icon with the check mark underneath the File in the option bar) to make sure the code has no inaccuracies.

4. Connect one side of the USB-to-Arduino wire connector to the USB port of the computer and the other side to the Network connector port on the Arduino board.

5. Before uploading the code to the Arduino, click Tools → Serial Port and select any of the Serial Port options EXCEPT for COM1.

   **Note**: COM1 is strictly for the computer, so it will not transfer the code to the Arduino. If the Serial Port option originally chosen does not work, try out the other options (except for COM1).

6. Click Tools → Board and select the Arduino board you are using (in this case, the Arduino Uno).

   **Note**: Once you have chosen the Serial Port and board, press the Upload button on the tool bar of the Arduino program (the arrow pointing to the right). The code should automatically start working and the ping sensor should trigger the motors direction change.

7. To see the ping sensor measurements, click the Serial Monitor button (blue icon with a magnifying glass on the far right side of the tool bar on the Arduino program). This will open a second window. Check the box next to Autoscroll (on the bottom left corner of the Serial Monitor window) to follow all new readings of the ping sensor.

8. Once the code works, go to File → Save As… to save the code and name the file.

---

### 2.1.5. INCREASING ROVER INTELLIGENCE

The following code and wiring diagrams provide for a much more intelligent rover that, instead of simply reversing to avoid obstacles, will be able to know whether to turn left or right to avoid a given object. This will enable the rover to practice object avoidance while still maintaining its original path.

**Note**: Additional required materials include one additional ultrasonic range sensor (four prong), one additional breadboard, and additional wires.

**To wire the sensors:**

The wire setup should be mounted so that each sensor has an unobstructed field of view. The wiring diagram is below.
To program the Arduino:

1. Open the Arduino program on your computer. If Arduino is not loaded, go to http://arduino.cc/en/Main/Software and download the version that suits your computer (i.e. Mac or PC, 32 bit or 64 bit, etc.). Follow the instructions for downloading and installation.

2. Once you open Arduino, insert the following code into the code field. You can also find this code in the file Arduino_Ping2.txt (see supplemental files).

```c
#include <NewPing.h>
#define TRIGGER_PIN 7
#define ECHO_PIN 6
#define MAX_DISTANCE 200
```
NewPing sonar (TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); //pins assigned to front sensor

NewPing sonar2 (5, 4, 200); //pins assigned to left sensor

NewPing sonar3 (3, 2, 200); //pins assigned to right sensor

int fromOther = 0;

void setup() {
    Serial.begin(9600); //Open serial monitor to view ping
    // set channel a
    pinMode (12, OUTPUT);
    pinMode (3, OUTPUT);
    pinMode (9, OUTPUT);
    // set channel b
    pinMode (13, OUTPUT);
    pinMode (11, OUTPUT);
    pinMode (8, OUTPUT);
    delay (1000);
    //turn brake off
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);
    //set motors forward
    digitalWrite(12, HIGH);
    digitalWrite(13, HIGH);
    //set motors speed
    digitalWrite(3, 150);
    digitalWrite(11, 150);
}

void loop() {
    delay(50);
    // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
    unsigned int uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
    Serial.print("Ping: ");
    int dist = (uS / US_ROUNDTRIP_CM);
// Convert ping time to distance in cm and print result (0 = outside set distance range)
Serial.print(dist);
Serial.print(" cm");
Serial.print(" ");
delay(50);

// Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
unsigned int uS2 = sonar2.ping(); // Send ping, get ping time in microseconds (uS).
Serial.print("Ping 2: ");
int dist2 = (uS2 / US_ROUNDTRIP_CM);
// Convert ping time to distance in cm and print result (0 = outside set distance range)
Serial.print(dist2);
Serial.print(" cm");
Serial.print(" ");
delay(50);

// Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.
unsigned int uS3 = sonar3.ping(); // Send ping, get ping time in microseconds (uS).
Serial.print("Ping 3: ");
int dist3 = (uS3 / US_ROUNDTRIP_CM);
// Convert ping time to distance in cm and print result (0 = outside set distance range)
Serial.print(dist3);
Serial.println("cm");

//***A*** Nothing front
if (dist > 30 || dist == 0) {
  if (fromOther == 1) {
    delay(500);
  }
  digitalWrite (12, HIGH);
  digitalWrite (13, HIGH); // Go forward
  fromOther = 0;
}

//***B*** Something front, something left, nothing right
else if ((dist2 < 30 && dist2 > 0) && ((dist3 > dist2 && dist3 > 30) || dist3 == 0)) {
    digitalWrite (12, LOW);
    digitalWrite (13, HIGH);    //Turn right
    fromOther = 1;
}

/***C*** Something front, something right, nothing left

else if ((dist3 < 30 && dist3 > 0) && ((dist3 < dist2 && dist2 > 30) || dist2 == 0)) {
    digitalWrite (12, HIGH);
    digitalWrite (13, LOW);     //Turn left
    fromOther = 1;
}

/***D*** Something front, something left, something right

else {
    digitalWrite (12, LOW);
    digitalWrite (13, LOW);     //Go Backwards
    fromOther = 1;
}

3. Press the Verify button (blue icon with the check mark underneath the File in the option bar) to make sure the code has no inaccuracies.

4. Connect one side of the USB-to-Arduino wire connector to the USB port of the computer and the other side to the Network connector port on the Arduino board.

5. Before uploading the code to the Arduino, click Tools → Serial Port and select any of the Serial Port options EXCEPT for COM1.

   **Note**: COM1 is strictly for the computer, so it will not transfer the code to the Arduino. If the Serial Port option originally chosen does not work, try out the other options (except for COM1).

6. Similarly, click Tools → Board and select the Arduino board you are using (in this case, the Arduino Uno). Once you have chosen the Serial Port and Board, press the Upload button on the tool bar of the Arduino program (the arrow pointing to the right). The code should automatically start working and the ping sensor should trigger the rover to change direction.
7. To see the ping sensor measurements, click the Serial Monitor button (blue icon with a magnifying glass on the far right side of the tool bar on the Arduino program). This will open a second window. Check the box next to Autoscroll (on the bottom left corner of the Serial Monitor window) to follow all new readings of the ping sensor.

8. Once the code works, go to File → Save As... to save the code and name the file.

The list below provides additional resources to assist you:

- Sample rover kit with all necessary parts to mount an Arduino Board and associated sensors:
  

- Sample rover kit that comes with a preinstalled Arduino Uno Board:
  

- Provides a basic diagram of set-up and an example of code:
  

- Provides an example of a basic code for ping sensors:
  

- Interesting project that deals with autonomous gardening vehicles including sample code and design instructions:
  
  http://www.instructables.com/id/Seed-Spreading-Robot/step21/Seed-Spreader-Brackets/