Responding to Sensors

Add infrared distance sensors and reflectance sensors to your robot.

Site: iCODE
Course: Machine Science Guides (Arduino Version)
Book: Responding to Sensors
Printed by: Guest User
Date: Thursday, August 7, 2014, 02:27 PM

Contents

Responding to Sensors

In this section, you will connect two types of infrared sensors to your robot: an analog distance sensor and two infrared reflectance sensors. You will then program the robot to respond to input from these sensors. Figure 1 shows the three sensors mounted to the front of the robot chassis.

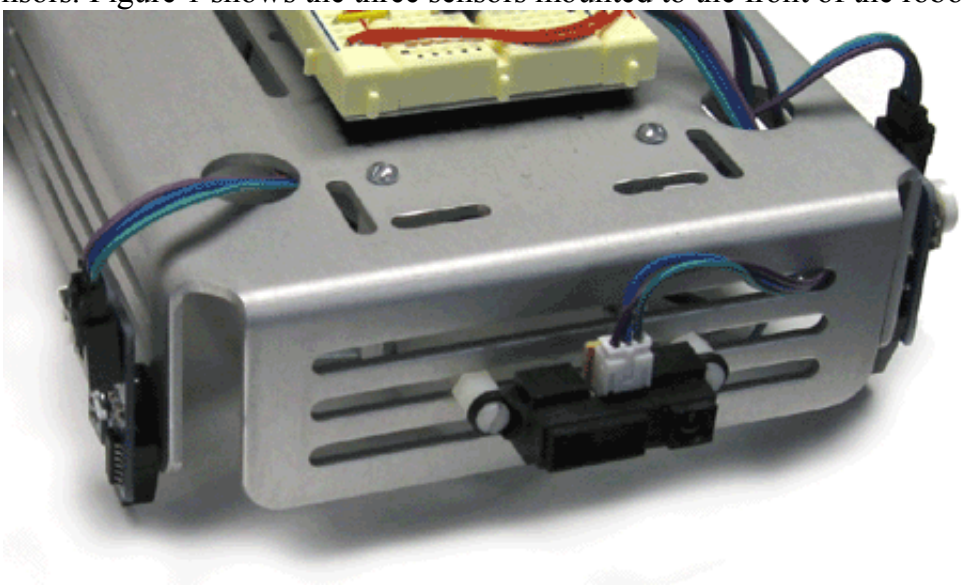

Figure 1. Sensors mounted to the front of the robot chassis.

Installing the Analog Distance Sensor

The analog distance sensor should be mounted to the slots on the front of the robot chassis, so that it can detect objects in the robot's path.

1. Using nylon standoffs, screws, and nuts, secure the sensor to the front of the chassis, as shown in Figure 2. ***NOTE: Make sure that the standoffs insulate the sensor from the chassis.***

2. Check to make sure that the sensor is directly centered in the slot before tightening the standoffs and screws.

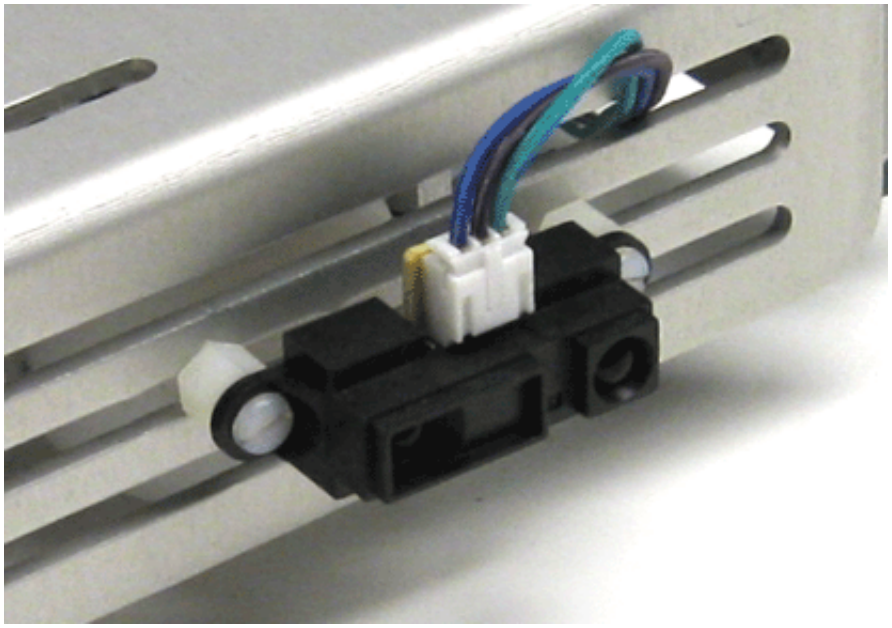

Figure 2. Analog distance sensor secured to the chassis.

Installing the Down Facing Sensors

The infrared reflectance sensors should be mounted toward facing down, and positioned near the front of the robot, so that they can detect a table edge before the robot encounters it.

1. Attach one reflectance sensor on the left side of the chassis, using a ½" 4-40 machine screws and nuts. Mount the sensor on the lowest slot, as shown in Figure 3.

2. Repeat step 1 to attach the second sensor in the same slot on the other side of the robot.

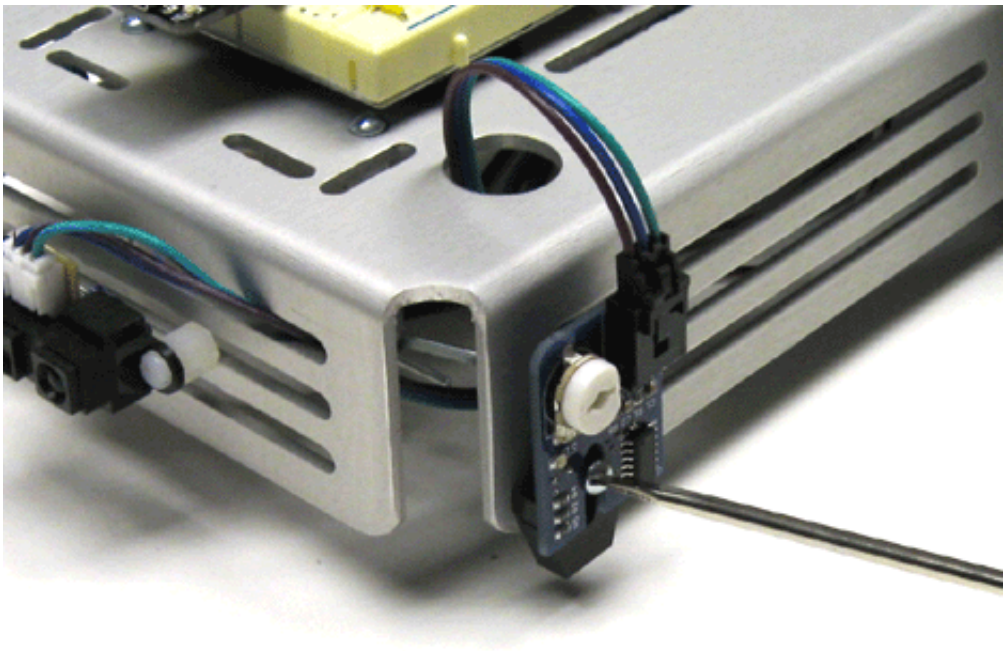

Figure 3. Securing the infrared reflectance sensor.

Wiring the Sensors

Each sensor is connected to the breadboard by a three-strand ribbon cable. Before connecting the sensors, examine these cables. Each has a green (signal) strand, a blue (power) strand, and a purple (ground) strand. The orientation of these strands is *critical* to the proper functioning of the down sensors. Using the schematic shown in Figure 4 as a guide, connect the three sensors to the breadboard.

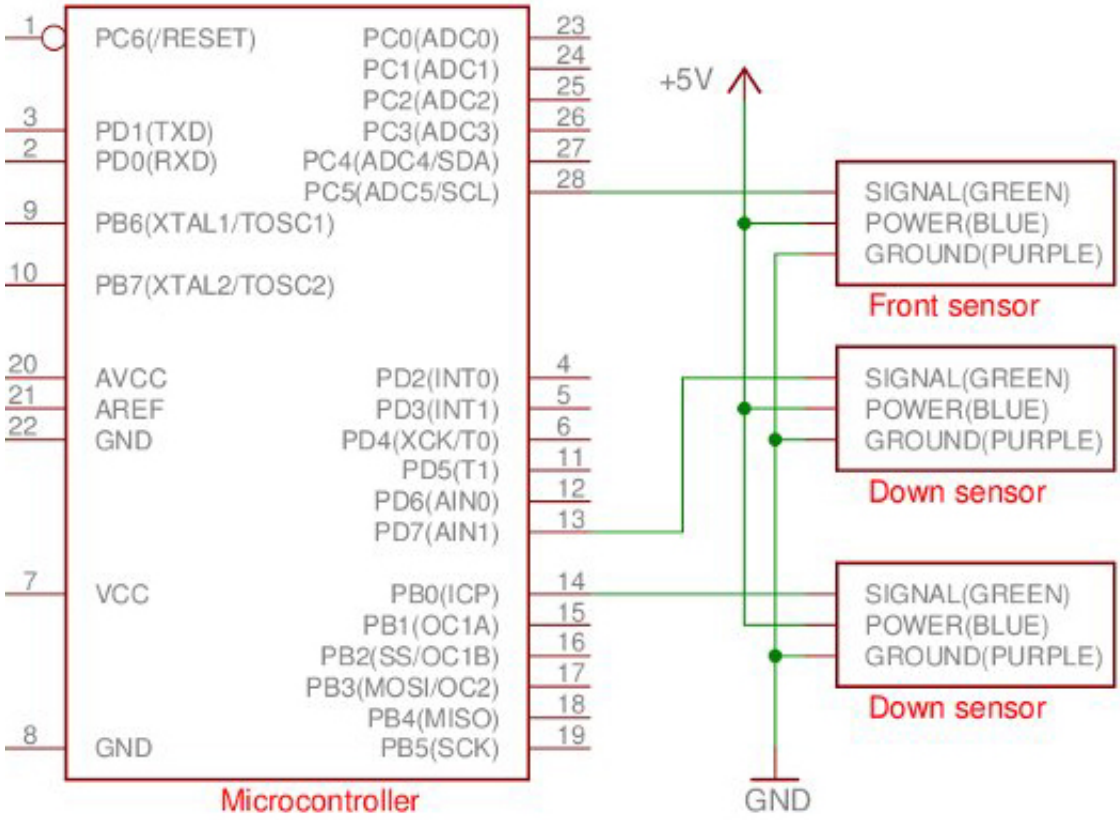

Figure 4. Robot sensor schematic.

Figure 5 shows one way that jump wires can be used to make the necessary connections to power, ground, and the microcontroller.

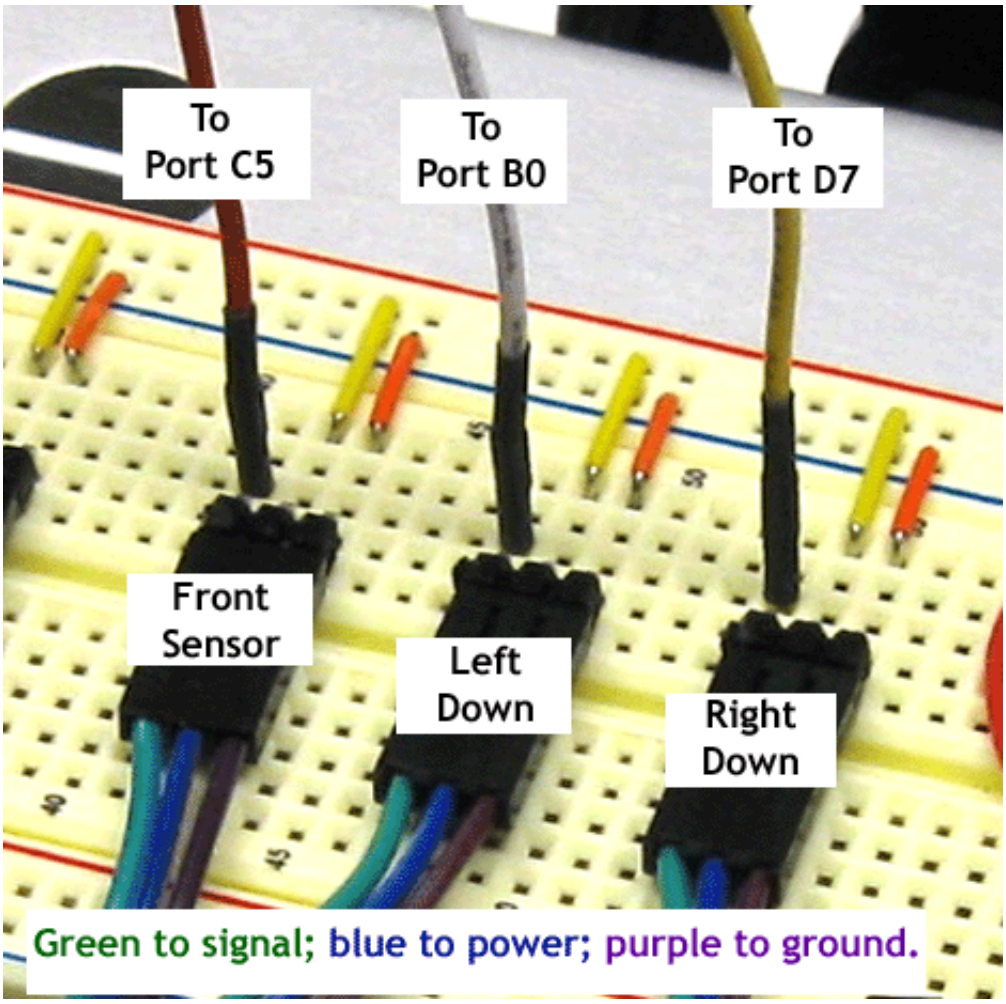

Figure 5. One way to connect the sensors.

Writing Code for the Front Sensor

The front sensor can detect objects at a distance of 5 to 150 centimeters (approximately 2 to 60 inches). Depending on the object's distance from the sensor, the sensor returns a value ranging from about 150 (no object is detected) to about 650 (object is at the minimum detectable distance from the sensor). When an object is extremely close to the sensor, the sensor cannot gauge its distance accurately. Sensors that return values in a continuous range, as opposed to discrete off/on values, are called *analog* sensors. The following code example displays the values returned by the front sensor on the ATmega Board's LCD.

1. Type or paste the following code into the editor window.

```
#include <mxapi.h>
#include <lcd.h>
#include <adc.h>

int main(void)
{
   int x=0;  //Declare a variable called x
   lcd_init();  //Initialize the LCD
   adc_init();  //Initialize analog input
   while(1==1)
   {
      x=adc_read(5); //Set x equal to analog value on
      lcd_decimal(FIRST_LINE, x, 4);   //Display x on LCD
      lcd_instruction(GOTO_LINE1);
      delay_ms(10);
   }
}
```

2. Compile and test your new code. Place your hand in front of the sensor at varying distances and see how the values on the LCD change.

Avoiding Walls and Obstacles

To make the robot respond to the front sensor, you need to introduce an if-else statement based on the value returned by the analog_read function. In the example below, the robot will stop if the sensor returns a value greater than 250.

1. Type or paste the following code into the editor window.

```
#include <mxapi.h>
#include <servo.h>
#include <adc.h>

int main(void)
{
    servo_init();              //Initialize servo motor control
    adc_init();                //Initialize analog sensor input
    while(1==1)                //Execute the following code repeatedly
    {
        if(adc_read(5)>250)     //If sensor detects a close object
        {
            servo_robot(STOP, 0);       //Make robot stop
        }
        else                        //If sensor does NOT detect a close
object
        {
            servo_robot(FORWARD, 100); //Make robot move forward
        }
    }
}
```

2. With your robot's wheels suspended off the ground, download this code to the ATmega Board. The wheels should move in the forward direction when there is no object in front of the robot. When you place your hand or another object within a few inches of the sensor, the wheels should stop moving.

Once you have confirmed that your robot wheels all stop when the front sensor detects an object, you are ready to test it by placing it on the floor and pointing it toward a wall or obstacle.

Continuous Object Avoidance

In addition to just stopping when it sees an obstacle, your robot can be programmed to make an evasive maneuver. The following code causes the robot to turn to the right if it sees an obstacle and continue driving.

1. **Type or paste the following code into the editor window.**

```
#include <mxapi.h>
#include <servo.h>
#include <adc.h>

int main(void)
{
    servo_init();        //Initialize servo motor control
    adc_init();          //Initialize analog sensor input
    while(1==1)          //Execute the following code repeatedly
    {
        if(adc_read(5)>250)      //If sensor detects a close object
        {
            servo_robot(SPIN_RIGHT, 100);  //Spin right for 500 ms
            delay_ms(500);
        }
        else                            //If sensor does NOT detect close
object
        {
             servo_robot(FORWARD, 100);      //Go forward
        }
    }
}
```

2. Download and test your new code.

If the robot continues to get stuck, reposition the front sensor or modify your code to improve its performance.

Avoiding the Edge of a Table

Using the two down sensors and the code in this section, your robot should be able to navigate around a table without falling off. When the down-facing sensors detect something below them (the table), the sensors send a HIGH signal to the microcontroller. When they do not detect the table, the signal line is LOW. Unlike the front-facing sensor, the down-facing sensors are referred to as *digital* sensors, because they only have two states, HIGH and LOW.

1. Using a small flat head screwdriver, gently turn each sensor's potentiometer as far as it can go in the counterclockwise direction. This sets the sensor for maximum sensitivity, helping to ensure that your down sensors will be able to detect the edge of the table. (More information about sensitivity adjustment is provided in the next section.)

2. Type or paste the following code into the editor window.

```
#include <mxapi.h>
#include <servo.h>

int main(void)
{
    servo_init();          //Initialize servo motor control
    while(1==1)            //Execute the following code repeatedly
    {
        if(pin_value(PORT_B0)==LOW)   //If left sensor detects table edge
        {
            //Move the robot in reverse for 1/2 second
            servo_robot(REVERSE, 30);
            delay_ms(500);
            //Spin the robot right for 1/4 second
            servo_robot(SPIN_RIGHT, 30);
            delay_ms(250);
        }
        else if(pin_value(PORT_D7)==LOW)  //If right sensor detects table
edge
        {
            //Move the robot in reverse for 1/2 second
            servo_robot(REVERSE, 30);
            delay_ms(500);
            //Spin the robot left for 1/4 second
            servo_robot(SPIN_LEFT, 30);
            delay_ms(250);
        }
        else
        {
            servo_robot(FORWARD, 30);         //Make the robot move
forward
        }
    }
}
```

3. Download and test your new code.

IMPORTANT NOTE

In this activity, there is the strong possibility that your robot could fall off the table, causing permanent damage to the sensors or other electronic components. For this reason, it is imperative to do all testing at the lowest speed. Until you are certain that your robot is working correctly, always have at least two people acting as "spotters" to ensure that the robot does not fall to the floor in the event that something doesn't work as planned.

Adjusting the Sensitivity

The sensitivity of the down sensors can be adjusted, using the on-board potentiometer, as shown in Figure 6. Turning the dial counterclockwise makes the device more sensitive, and turning it clockwise makes it less sensitive. The red LED glows to indicate when the device does not "see anything," meaning it is sending a LOW signal to the microcontroller. The sensitivity adjustment can be important when programming the sensor to detect the difference between light and dark colors, as shown in Figure 6.

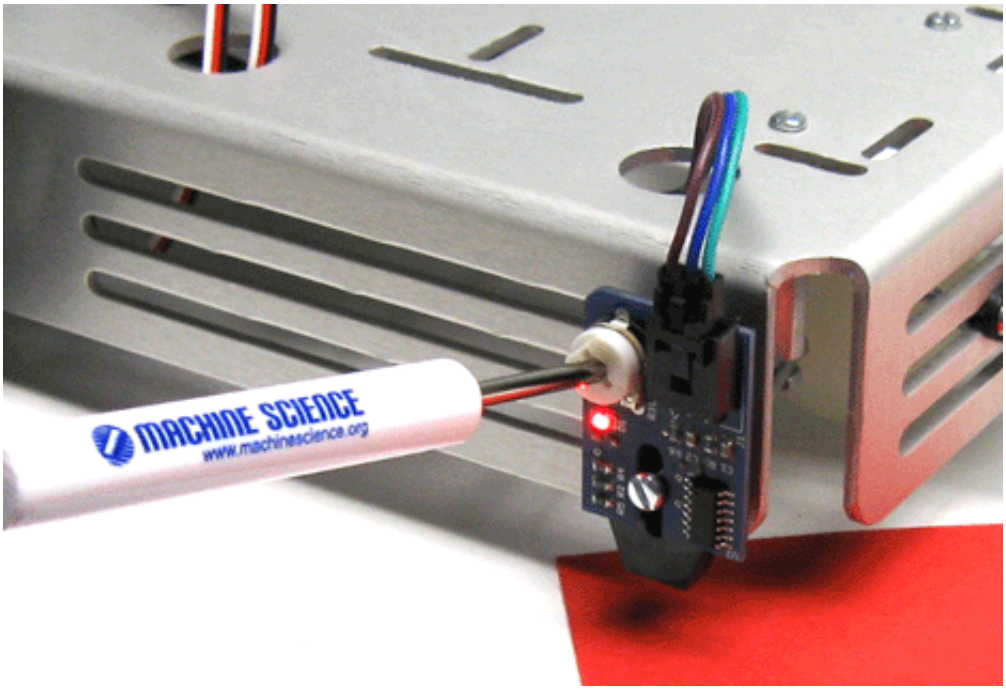

Figure 6. Adjusting the sensitivity of the down-facing sensor.

The mounting height of the sensor also affects how it performs. If the sensor is mounted too high, it may never send a HIGH signal, even when adjusted for maximum sensitivity. Likewise, if the sensor is mounted too low, it may not perform as expected. The ideal height is roughly one-eighth of an inch, or 4 to 5 millimeters.