Controlling the Robot

Use the ATmega Board to control a robot with two servo motors.

Site: iCODE
Course: Machine Science Guides (Arduino Version)
Book: Controlling the Robot
Printed by: Guest User
Date: Thursday, August 7, 2014, 02:28 PM

Contents

Understanding Front, Back, Left, and Right

Before moving ahead, take a moment to review Figure 1, which identifies the left motor and the right motor, as well as the definitions used in this tutorial for forward, reverse, left, and right movements.

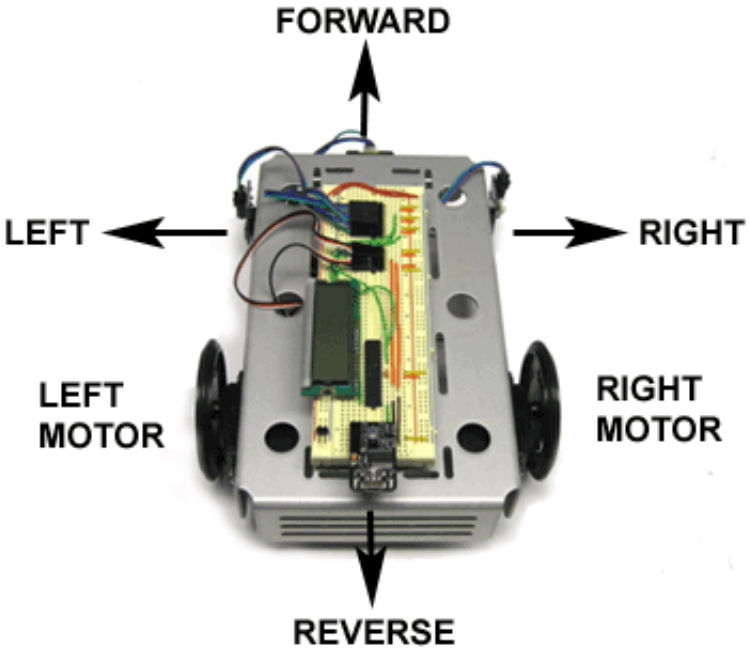

Figure 1. Direction conventions used in this tutorial.

Controlling One Motor

The direction and speed of any servo motor connected to Port D2 (MOTORA) or Port D3 (MOTORB) can be controlled with a function called servo_motor. This function takes two instructions: 1) the identity of the motor (MOTORA or MOTOR B), and 2) the direction and speed of the desired rotation (-100 to +100). Positive (+) speed values produce clockwise rotation, and negative (-) speed values produce counterclockwise rotation. The allowable range is -100 (full speed counterclockwise) to +100 (full speed clockwise). In the code example below, the servo_motor function tells the motor connected to Port D2 (MOTORA) to turn clockwise at full speed.

IMPORTANT NOTE

Before programming your ATmega Board, place the robot on a book or another stable object so that its wheels can turn freely (without moving the robot). Be sure that the battery is connected properly, and check all electrical connections before moving the power switch to the ON position.

1. If you have not already done so, launch the Arduino IDE.
2. In the editor window, type or paste the following code:

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

int main(void)
{
    servo_init();                //Initialize servo motor control
    servo_motor(MOTORA, +100);   //Move MOTORA clockwise at full speed
    while(1==1);                       //End the program
}
```

3. Compile and test your code.

The right motor should spin clockwise until you switch the power off. Before moving on to the next section, try changing the speed of the motor a few times. Remember that +100 is full speed and 0 is a full stop. After each change, compile and then download the new code. If you encounter strange behavior from the motors, please consult the Adjusting the Servo Motors page at the end of this tutorial.

Reversing the Motor's Direction

Using the code in this section, you will learn how to drive the motor in one direction and then another direction for a specified period of time. To do this, you need to introduce a new programming structure, called a delay statement. Delays can be specified either in milliseconds (ms or 1/1000ths of a second) or in microseconds (us or 1/1,000,000ths of a second). For a delay measured in milliseconds, use a delay_ms statement; for a delay measured in microseconds, use a delay_us statement.

1. In the editor window, type or paste the following code:

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

int main(void)
{
    servo_init();                //Initialize servo motor control
    servo_motor(MOTORA, +100);   //Move motor A clockwise at full speed
    delay_ms(3000);              //Delay for 3000 milliseconds
    servo_motor(MOTORA, -100);   //Move motor A counterclockwise at full
speed
    while(1==1);                 //End program
}
```

2. Compile and download this code file.

Before moving on to the next section, program the motor to turn clockwise for five seconds, then counterclockwise for four seconds, then clockwise again.

Controlling Both Motors

In this section, you will write code to drive both motors simultaneously. This is a prelude to placing the robot on the floor and turning it loose!

1. In the editor window, type or paste the following code:

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

int main(void)
{
    servo_init();                //Initialize servo motor control
    servo_motor(MOTORA, +100);   //Move motor A clockwise at full speed
    servo_motor(MOTORB, +100);   //Move motor B clockwise at full speed
    delay_ms(3000);              //Delay for 3 seconds
    servo_motor(MOTORA, 0);      //Stop motor A
    servo_motor(MOTORB, 0);      //Stop motor B
    while(1==1);                 //End program
 }
```

2. Compile and download this code file.

Both sets of wheels should now turn for three seconds and then stop. Note that, with both motors spinning in the same direction (in this case clockwise), if you were to place the robot on a flat surface (which you will do later), it would turn in place—not move forward or backward.

Moving the Robot to the Floor

OK. Now the big moment. You are ready to place your robot's wheels on the floor and see it move!

1. 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
    servo_motor(MOTORA, -30);   //Move motor A counterclockwise at speed
30
    servo_motor(MOTORB, +30);   //Move motor B clockwise at speed 30
    delay_ms(3000);             //Delay for 3 seconds
    servo_motor(MOTORA, 0);     //Stop motor A
    servo_motor(MOTORB, 0);     //Stop motor B
    while(1==1);                //End program
}
```

2. Program the microcontroller and confirm that the wheels turn as expected while it is above the table.
3. Turn off the robot.
4. Remove the USB cable from the microcontroller.
5. Place the robot on the floor and make sure the path in front of it is clear for several feet.
6. Switch the robot on and confirm that it moves forward for three seconds, and then stops.

Navigating by Dead Reckoning

Dead reckoning is the process of estimating a robot's position based upon a previously determined position and advancing from that position based upon time or speed. Modifying the code shown below, you will instruct your robot to navigate a 3-foot by 3-foot square and then stop. Before writing this program, you may wish to mark a 3-foot by 3-foot square foot square on the floor, using masking tape.

1. Type or paste the following code into the editor window, changing the delay times to make your robot move forward 3 feet and complete a 90-degree turn to the right. It may take several iterations to determine the correct delay timing to complete this task.

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

int main(void)
{
    servo_init();               //Initialize servo motor control
    servo_motor(MOTORA, -30);   //Move motor A counterclockwise at speed
30
    servo_motor(MOTORB, +30);   //Move motor B clockwise at speed 30
    delay_ms(5000);             //Delay to move forward 3 feet
    servo_motor(MOTORA, -30);   //Move motor A counterclockwise at speed
30
    servo_motor(MOTORB, -30);   //Move motor B counterclockwise at speed
30
    delay_ms(1000);             //Delay to turn right 90 degrees
    servo_motor(MOTORA, 0);     //Stop motor A
    servo_motor(MOTORB, 0);     //Stop motor B
    while(1==1);                //End the program
}
```

2. Compile and download your code.
3. Observe the movement of your robot to determine if you need to increase or decrease the delay times.
4. Update your code with new delay times so that your robot moves forward exactly 3 feet and turns exactly 90 degrees to the right.

After you have determined the correct delay times, add additional code to allow your robot to move in a 3-foot by 3-foot square.

Making an Infinite Loop

In this step, you will introduce a loop structure, called a while loop, to make your code repeat forever (or at least until the battery runs out!) A while loop executes a series of commands while (i.e. for as long as) a certain condition is true. The condition can be any mathematical statement that is either true or false. In this case, you will use a condition that is always true: 1==1. (Note that, in C, a condition of this type is expressed with two equals signs, not just one.)

1. Type or paste the following code into the editor window, filling in the required directions:

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

int main(void)
{
    servo_init();     //Initialize servo motor control
    while(1==1)       //Execute the following code repeatedly
    {
        servo_motor(MOTORA, -30); //Move motor A counterclockwise at
speed 30
        servo_motor(MOTORB, +30); //Move motor B clockwise at speed 30
        delay_ms(5000);           //Delay to move forward 3 feet
        servo_motor(MOTORA, -30); //Move motor A counterclockwise at
speed 30
        servo_motor(MOTORB, -30); //Move motor B counterclockwise at
speed 30
        delay_ms(1000);           //Delay to turn right 90 degrees
    }
}
```

2. Compile and download your new code.

Simplifying Robot Movements

Keeping track of the clockwise and counterclockwise settings needed to produce each robot motion can be cumbersome. To simplify these controls, you can use another function, called servo_robot. The servo_robot function controls the motors on Port D2 and Port D3 at the same time, producing forward and reverse motions, as well as left and right turns, as shown below.

1. 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)                         //Begin infinite loop
    {
        servo_robot(FORWARD, 30);       //Move robot forward at 30 speed
        delay_ms(5000);                 //Delay to move forward
        servo_robot(SPIN_RIGHT, 30);    //Spin robot to the right at 30
speed
        delay_ms(800);                  //Delay to turn right
    }
}
```

2. Compile and test your new code.

The following arguments may be passed to the servo_robot function: FORWARD, REVERSE, SPIN_LEFT, SPIN_RIGHT, and STOP.

IMPORTANT NOTE

For the servo_robot function to work properly, the left motor must be connected to Port D2, and the right motor must be connected to Port D3. If your robot is not responding correctly to the servo_robot function (going reverse instead of forward, and spinning left instead of right), turn off the microcontroller and carefully disconnect the two servo motor plugs and reverse them, moving the lead on Port 0 to Port 1 and vice versa. Be sure to keep the white (signal), red (power), and black (ground) wires when reconnecting the motor leads.

Adjusting the Servo Motors

Every continuous rotation servo motor has a "center point," at which it transitions from clockwise to counterclockwise rotation. Close to where the power, ground, and signal wires connect to the motor housing, there is a circular opening with an adjustment screw recessed just below the surface, as shown in Figure 2. The position of the adjustment screw determines the motor's center point. If your servo motors are performing erratically (rotating when they should stopped or rotating at inconsistent speeds), they probably need to be adjusted.

1. Program your microcontroller to send a "stop" signal to the motor -- e.g. servo_motor(0, 0) or servo_robot(STOP, 0).

2. Using a Phillips head screwdriver, gently turn the adjustment screw until the motor shaft stops rotating.

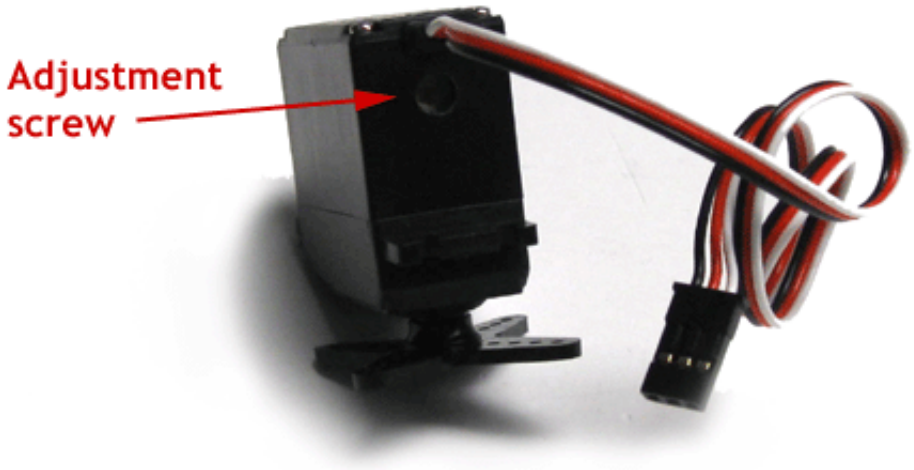

Figure 2. Servo motor adjustment screw.