Eric Mitchell
April 2, 2012

Application Note:
Control of a 180° Servo Motor with Arduino UNO Development Board

Abstract

This application note is a tutorial of how to use an Arduino UNO microcontroller to control an analog Servo motor through signals sent to the Servo motor's control line. This document will provide directions for wiring the Servo motor, the Arduino UNO and cover microcontroller programming techniques to control movement of the motor.

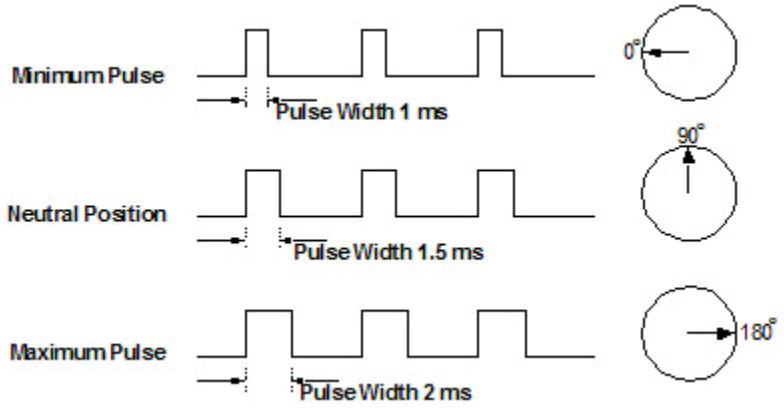
Introduction

Servo motors are small controllable motors that lend to implementation in many applications. There are Servos with many different speeds, sizes and torque capabilities, but all have 3 wires, power, ground and control. Servo motors are popular with hobbyists because they are inexpensive, $15-$100, and control of Servo motors with microcontrollers is universal for all models. Servo's receive pulse width modulated (PWM) signals to determine in which manner to move. There are many ways to send this signal to the motor; this application covers how to send the desired PWM signal to the Servo motor using the Arduino UNO microcontroller. The Arduino UNO is a popular microcontroller which comes on a development board to accelerate programming, provides simple interfacing with peripheral devices and connection with computers. The Arduino UNO chip is programmed with the Arduino programming language and Arduinio 1.0 software through a USB port on the board which plugs into a computer's USB port. The Arduino 1.0 software writes code from the software to the chip by uploading the file containing the desired code to the board. Once the chip has been programmed with the desired code, the chip can be removed from the development board and connected to any circuit. There

is much literature on this board and in the reference section of this document there are several places to find more information on this topic.

Specifications

Servo motors are controlled through the control line, usually a yellow or white wire. The pulse width of the signal sent to the Servo control wire determines how the motor will move, either clockwise or counter clockwise. The figure below shows how different pulse widths correspond with different position of the motor. When the Pulse Width is less than 1.5ms the motor will move to the 0° position and hold. When the Pulse Width is 1.5ms the motor will rotate to the 90° degree position and if the Pulse Width is greater than 1.5 ms the motor will rotate to the 180° position.



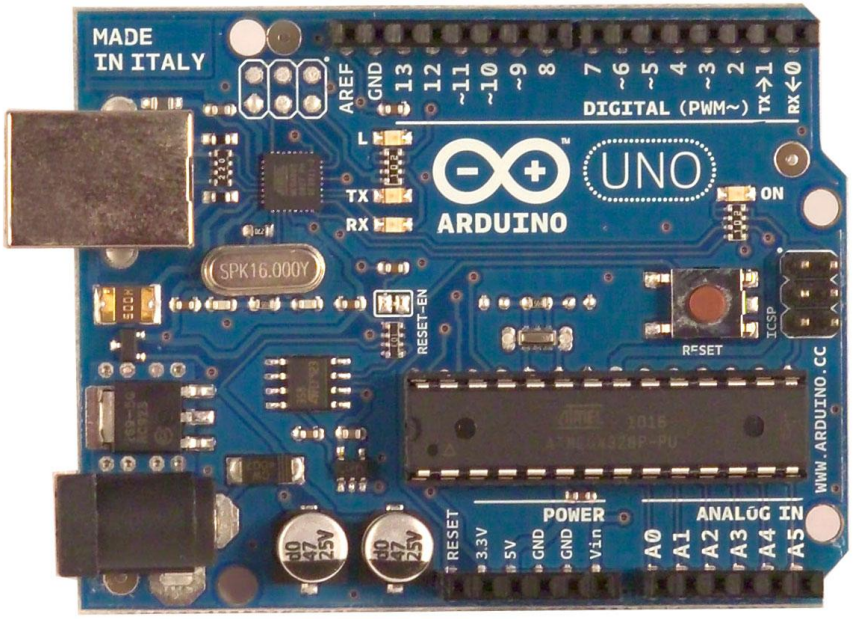
When the motor reaches the desired position it will hold there until a signal is sent to move. This is done in this application using the Arduino 1.0 coding software to write to one of the Arduino UNO's 5 PWM output pins. The PWM output pins on the development board can be written to with different pulse widths which are used to control the motor.

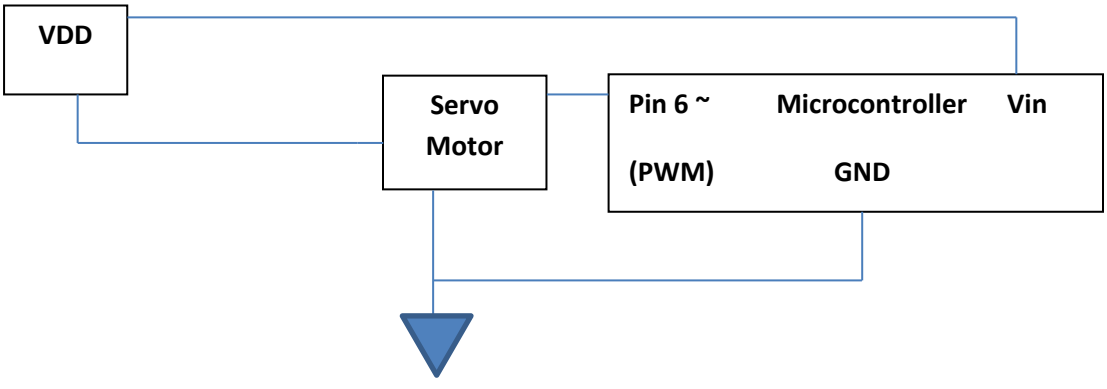
Hardware Configuration

The simplicity of the Servo and Arduino UNO connections is one of the major reasons for their popularity. The Servo motor has 3 wires as shown in the picture below. The red wire is for power and must be connected to a power supply of 4.8-6 Volts. The black wire is the ground wire and is

connected to ground. The third wire, usually white or yellow, is the control wire. This must be connected to one of the PWM outputs of the Arduino UNO (Pins: 3, 5, 6, 10, 11). Pin 6 will be used for this application note.

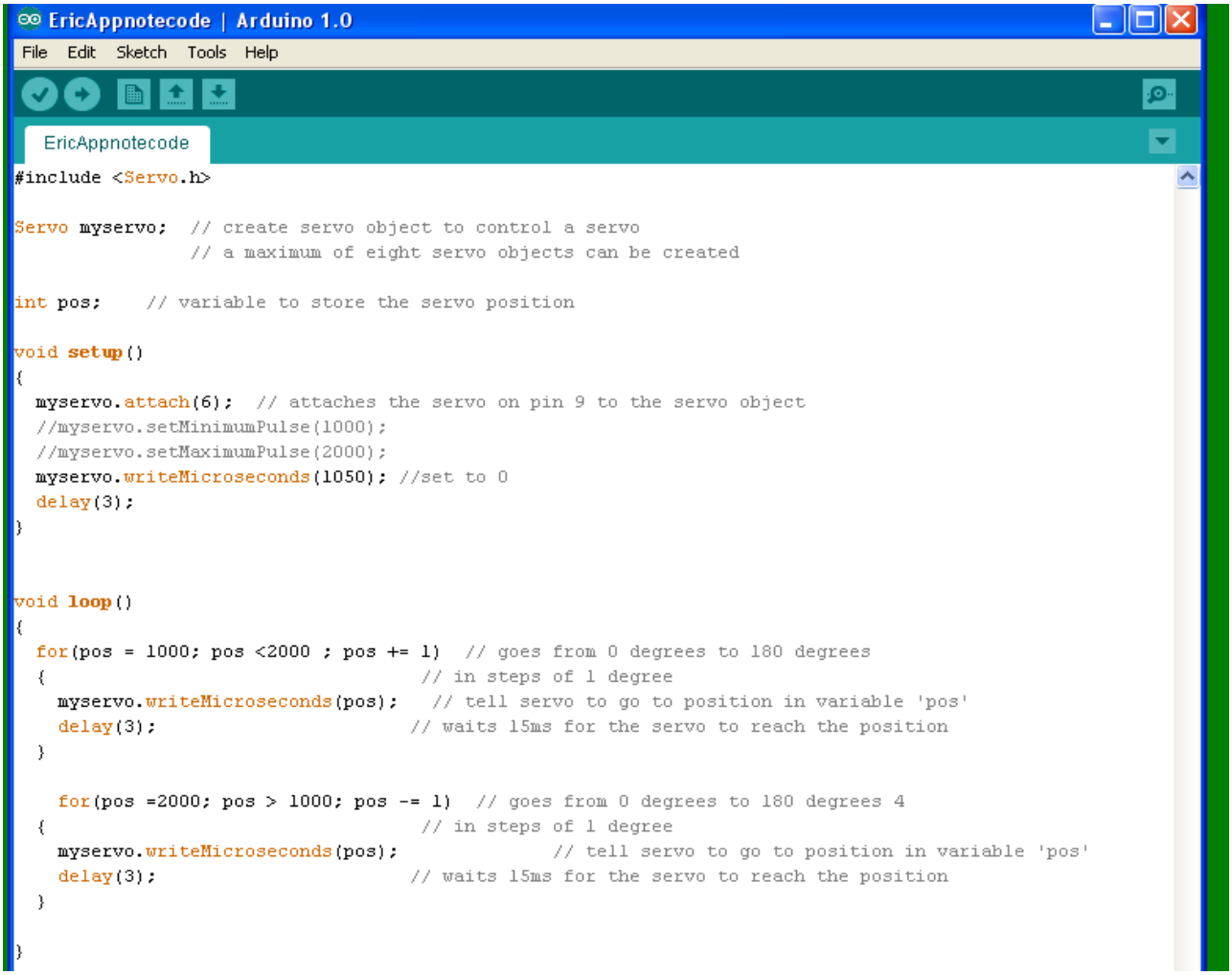
The Arduino UNO board, shown below, must be connected to power by either connecting the USB port on the board to a computer or wiring Vin to 3-5V and connecting GND to ground.





Software

There are many ways to program the microcontroller to control the operation of a Servo motor. This application will discuss how to make the Servo motor sweep from the 0° to the 180° continuously.



The above is a snapshot is from the Arduino 1.0 software and is the code used to perform the sweep operation. The comments on the code provide an explanation for each of the lines of code. This file is included in the reference section of this document. (1) The first step is to include the Servo.h library which contains commands for control of a Servo. The **void setup()** function is for attaching the servo to the Arduino board and also sets the Servo to its 90° position before

operation to set the Servo to a known position.. The **void loop()** statement controls the movement of the servo and runs continuously. The void loop contains two FOR loops, one for movement of the motor from 0° to 180° and the other from 180° to 0°. The first loop sets the PWM signal output to 1ms which moves the Servo to 0°. The pulse width is then incremented by 1us each time the loop is entered and the pulse width is written to port 6 until the pulse width reaches 2ms, the 180° position. There is also a delay of 15ms each time the pulse width is written to port 6 before sending the next signal to allow for the motor to move. The delay can be adjusted to shorten or lengthen the delay which corresponds to speeding up or slowing down the rotation. The second loops works similarly, but decrementing the pulse width from 2ms to 1ms which moves the motor from 180° to 0°.

Results

A Servo motor can be controlled with an Arduino UNO development board using the hardware and the software approach outlined above. With the program described running, and connections properly made, the Servo motor will continuously rotate 180°. The Servo motor can be coupled to a shaft which can be used for numerous applications. These methods can be modified slightly to move a motor in a variety of ways to fit another particular scenario. The ease of use of these 2 pieces of hardware offer a simple way to control a motor with a computer program.

References

1. Sample Arduino code for continuous 180 degree rotation

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
            // a maximum of eight servo objects can be created

int pos;    // variable to store the servo position

void setup()
{
 myservo.attach(6);  // attaches the servo on pin 9 to the servo object
 //myservo.setMinimumPulse(1000);
 //myservo.setMaximumPulse(2000);
 myservo.writeMicroseconds(1500); //set to 0
 delay(10);
}


void loop()
{
 for(pos = 1000; pos <2000 ; pos += 1)  // goes from 0 degrees to 180 degrees
 {                          // in steps of 1 degree
  myservo.writeMicroseconds(pos);   // tell servo to go to position in variable 'pos'
  delay(5);                 // waits 15ms for the servo to reach the position
 }

  for(pos =2000; pos > 1000; pos -= 1)  // goes from 0 degrees to 180 degrees 4
 {                          // in steps of 1 degree
  myservo.writeMicroseconds(pos);           // tell servo to go to position in variable 'pos'
  delay(3);                 // waits 15ms for the servo to reach the position
 }

}
```

2. Arduino Website: http://arduino.cc/

3. Arduino Servo Library:

http://arduino.cc/playground/ComponentLib/Servo

4. Servo Datasheet:

http://www.bizchip.com/servo.pdf