Brian Sheridan

Lab 3: Servo Motor Control

Performed on 4/22/15

Submitted on 5/6/15

Objective:

The goal of this lab was to learn how to use a servo motor, and control its position using the servo library and through manual control using pulse width modulation (PWM). The arduino built in library allows easy control of the servo through functions packaged in the servo library. PWM is accomplished through manually controlling an output pin and turn it on and off to emulate PWM.

Introduction:

The Arduino Uno microcontroller is an Atmel ATmega328 integrated onto a developer board to allow for easy prototyping with the chips many pins. The ATmeaga328 has 6 analog inputs, 14 digital inputs/outputs along with 3.3v, 5v and ground pins. Arduino also has a integrated development environment (IDE) for quick prototyping and coding which allows one to quickly design and test a project.

A standard servo motor is made up of a DC motor, potentiometer and a feedback control loop. The feedback look monitors the position of the servo, and generates a signal to keep the servo in the expected position through pulse width modulation. Positive and negative signals are used to move the servo left and right, based on the given position and the requested position.

Procedure:

Part 1 of the lab consisted of wiring the servo motor to the Arduino. 5V was connected to the red wire, GND to the black wire, and pin 10 to the orange wire. The servo was then programmed with the servo library to move to position 0, wait 2 seconds, then rotate to position 180, then wait 2 seconds and repeat the again.

```
#include <Servo.h>

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

int pos = 0;    // variable to store the servo position

const int MINPULSE = 600;
const int MAXPULSE = 2400;

void setup()
{
  myservo.attach(10, MINPULSE, MAXPULSE);  // attaches the servo on pin 9 to the servo object
}
```

```
void loop()
{
  myservo.write(0);
  delay(2000);

  myservo.write(180);
  delay(2000)
```

part 2 of the lab consisted of controlling the servo through pulse width modulation on pin 10. Pin 10 was raised high and low at a specified time to achieve the given position. A lookup table for each floor was created in an array to store the given high time needed to move the servo to the specified angle for each floor. The serial monitor was used to send the requested floor level to the Arduino. Based on the input character, the servo would then access the look up table and use the given values to move to the specified position.

```
#include <Servo.h>

const int NUMFLOORS = 5;
int delayTimes[NUMFLOORS+1];

int currentFloor = 1;

void setup()
{
  pinMode(10, OUTPUT);
  delayTimes[1] = 2000;
  delayTimes[2] = 1800;
  delayTimes[3] = 1600;
  delayTimes[4] = 1400;
  delayTimes[5] = 1200;

  Serial.begin(9600);
}


void loop()
{
  char myChar;
  unsigned long int highTime = 1500;
  unsigned long int lowTime = 18500;

  while(1){
    if(Serial.available()>0){
      myChar = Serial.read();
    }
    if(myChar>='1' && myChar <= '0' + NUMFLOORS){
      currentFloor =  myChar - '0';
    }
   highTime = delayTimes[currentFloor];
   lowTime = 20000 - highTime;

   digitalWrite(10, HIGH);
   delayMicroseconds(highTime);

   digitalWrite(10, LOW);
   delayMicroseconds(lowTime);

  }


}
```

Results:

Part 1 of the consisted of stetting up the servo using the servo.h library. This created the servo with the give min and max pulse on pin 10. Once the servo was set up, it was programmed to rotate to angle 0, wait 2 seconds then rotate to 180 degrees, then repeat. When the code was run the code worked exactly as expected, and completed a full 180 degree sweep. Though it might not have been exactly 180 degrees it was close enough given the tolerance of the servo.

Part 2 of the lab consisted setting up the servo without using the servo library. Instead manual pulse width modulation was used to control the angle the servo sweep. A look up table was created to store the exact pulse times for each desired floor on the elevator design. Next, the serial monitor was used to read in a number on the terminal, and sweep the servo to that floor which was done through reading the look up table. Overall this worked pretty well. When the floors were entered into the monitor, it swept just like a normal elevator monitor would. The precision was not nearly as precise due to using manual pulse width modulation but in the end it worked close enough.

Conclusion:

Overall this lab has shown how to create a servo object using the servo.h library included with the arduino, along with how to do it manually through pulse width modulation. It also showed how a look up table could be used to store values so a program does not need to continually calculate an equation, and can simply reference an array with predetermined values. Overall this lab was a success in learning how to use a servo with an Arduino Uno.