

**Computer Systems Engineering Technology**

**CST 120 – Embedded C Programming**

|  |  |
| --- | --- |
| Lab 5 – DC and Servo Motors | Name: Chris Thomas\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |
| Instructor: George Drouant | |
| Possible Points: 100 | |

# Purpose

In this lab, you will write/modify programs to control a small DC electric motor and a small servo. You will use the Atmega 328P to control a motor that requires more current to operate than the 328P can provide. We will use the protoboard power supply to provide power for the small DC motor. A history note - Check out the old advertisement at the end of the lab for the first microcontroller I used. I used it to build a controller for a prototype artificial kidney machine.

**Parts List**

UNO Board

Protoboard

Power Supply

DC Electric Motor with Fan

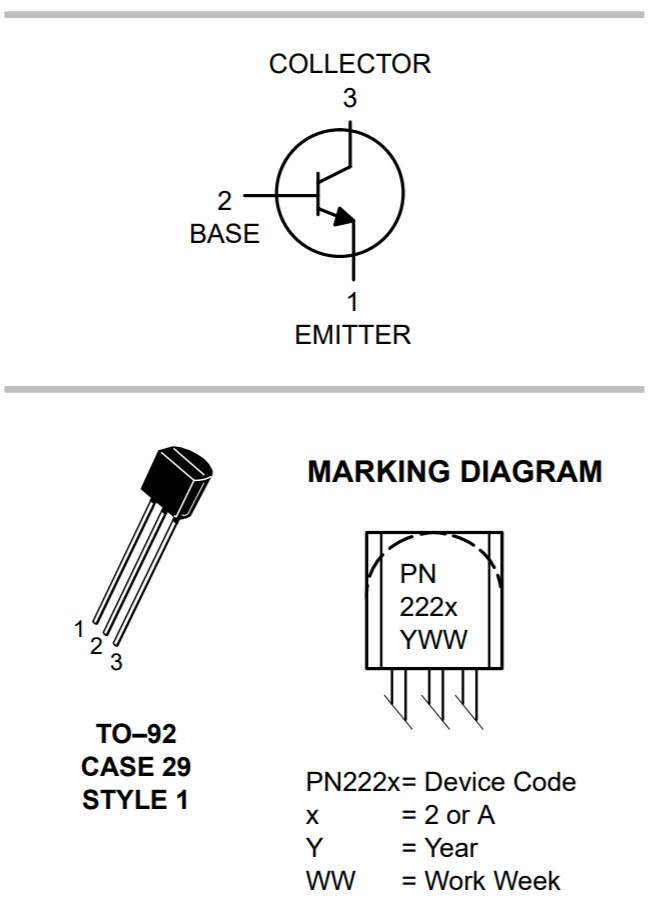
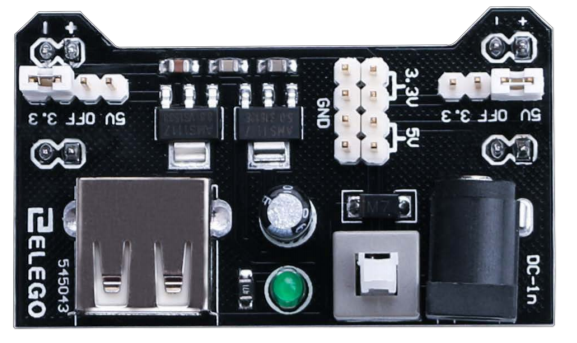
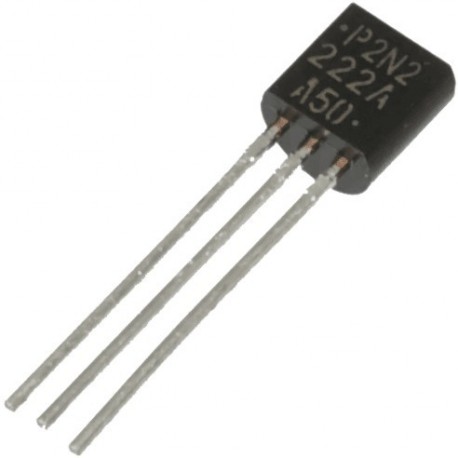
PN2222 NPN Transistor

Diode

100 Ohm Resistor

Servo

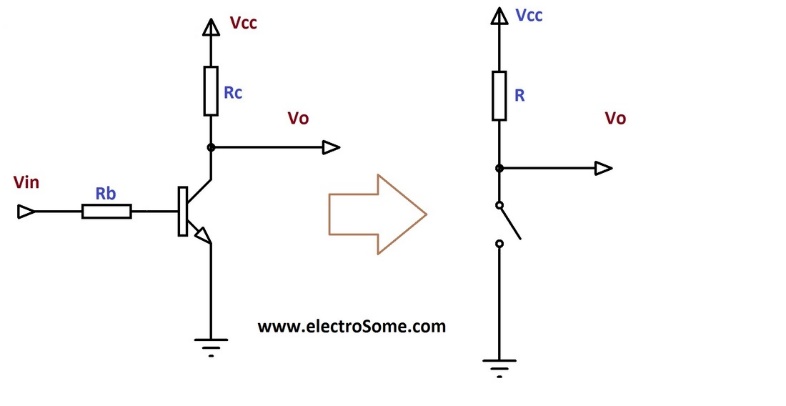
**Power Supply PN2222 NPN Transistor**

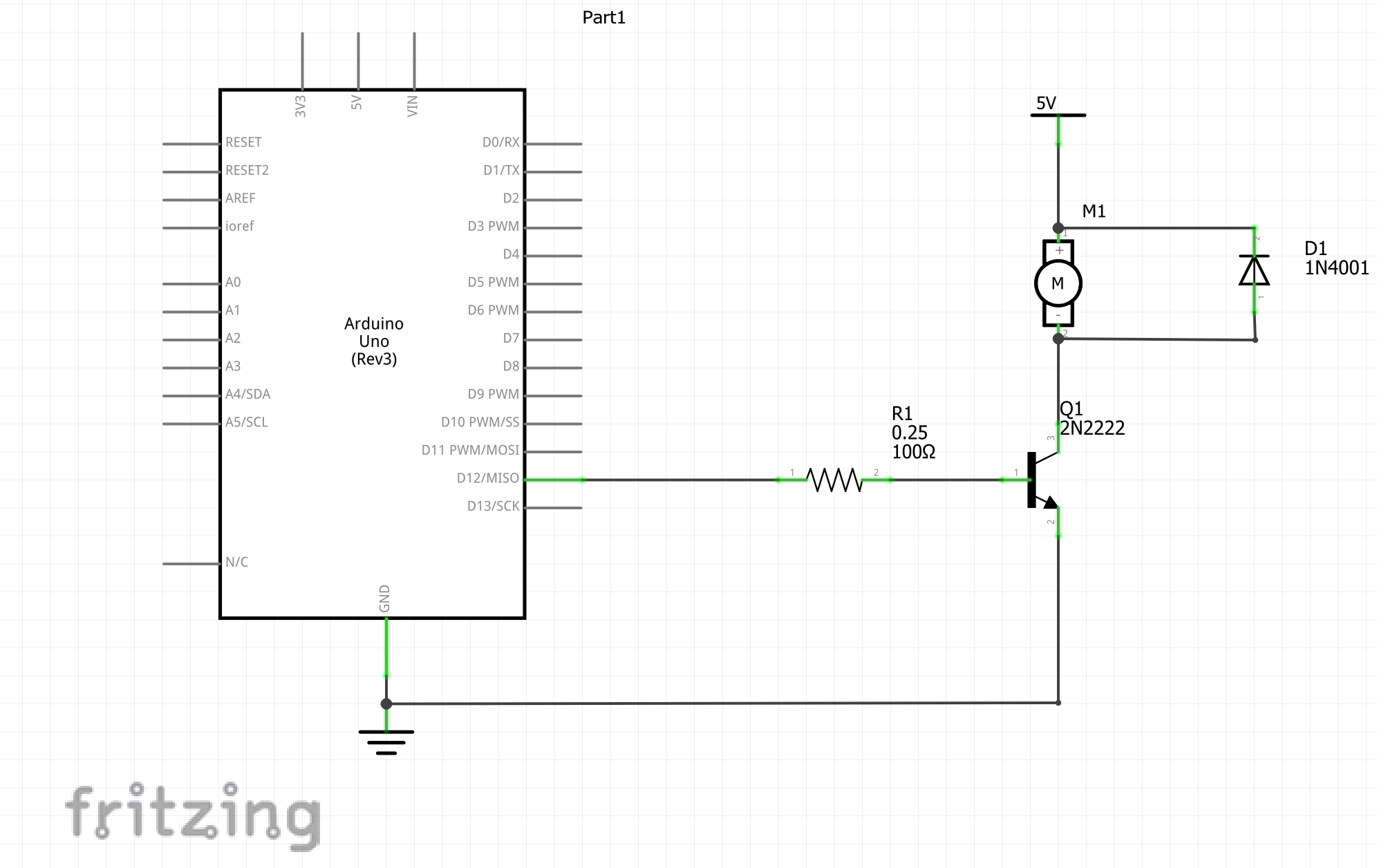
**Diode DC Motor with Fan**

**Part 1 – Control of DC Motor with a Solid-State Switch (NPN Transistor)**



In this part of the lab we will use a solid state (electronic) switch called a transistor. The basic idea behind a transistor switch is that a small current flowing into the base and out the emitter can control a much larger flow of current through the transistor from the collector to emitter. Current flowing into the base of the transistor is analogous to your figure pressing on the push button switch.



**Diode**



Build the circuit in the schematic above. Make sure that the power supply is turned off and the UNO board unplugged from the USB cable. Note that the power supply - **NOT** the UNO board - is supplying the 5V for the motor. Also, be aware that the ground of the UNO board IS connected to the ground of the power supply (ground is also labeled as the negative side of the power supply). The circuit element labeled D1 is a diode (see photo above). The diode is used to absorb voltage spikes created by the motor. You must place the diode with the silver band side of the diode connected to the “top” of the motor – the side of the motor connected to the power supply’s +5V terminal. The other side of the diode is connected to the junction of the motor and the collector terminal of the transistor. If the diode is installed backwards it is likely that the diode will be destroyed.

The transistor switch will be turned off and on by controlling the current flowing into the base terminal of the transistor. Digital Input/Output Pin 12 of the UNO board (PB4 of the 328P chip) will be used to control the base current. When a logic high is produced by Bit 4 of PORTB, current will flow into the base of the transistor and the “transistor switch” will close allowing current to flow through the motor turning the motor on. If Bit 4 PORTB is at a low logic level, no current will flow into the base and the “transistor switch” will be opened – no current will flow through the motor so the motor will be off. A 100 Ohm Resistor is used to connect the base terminal of the transistor to Digital I/O Pin 12 of the UNO board.

In the first part of this Lab the code will control the rotational speed of a small DC motor. The speed will be controlled by varying the duty cycle of a pulse width modulated signal (PWM) applied to the base of a transistor. The code will use one timer (Timer0) and two interrupts to produce the PWM signal. One interrupt will be generated when the timer’s count equals the number stored in the appropriate output compare register. This interrupt will determine how long the PWM signal is high. The second interrupt will be generated when Timer0 overflows (when it rolls over from 255 back to 0). This interrupt will set the length of the period for the PWM signal. There is a variable in the interrupt service routine (ISR) of this interrupt that allows us to select the number of overflows to count before the period of the PWS signal ends. That variable is initially set to 5.

## The Timer

We will use Timer0. Timer0 is an 8 bit timer meaning that it counts from zero to 255 and then overflows back to zero and continues counting. The timer also has two Comparators. You can use the comparators to trigger an event when the counter reaches a certain count. The timer can generate an interrupt on three events:

1. Timer Overflow
2. Timer equals Comparator A
3. Timer equals Comparator B

We will use the Timer Overflow and Comparator A interrupts in this lab.

Copy the code template from Canvas and paste it into a new Atmel Studio Project. Name the project “PWM\_Motor\_Control.” In multiple locations in the code you will see the word “BLANK.” Replace the word “BLANK” with the proper register, bit, instruction or keyword. Does the code build?

After the code builds, download it to the UNO and run the code. The program will ask you to input an integer speed setting between 35 and 155. Experiment with inputting several numbers.

What effect does the number you type have on the speed of the motor?

The higher the number the faster it spins

What happens to the speed of the motor if you enter the number 0?

It stops

What happens if you enter 255?

It spins even faster.

In the “if statement” of the Timer Overflow Interrupt Service Routine:

if(numOverflows == 5)

change the value of 5 to a value of 1. Re - Build and download the modified program. Is there any change in the performance of the motor?

A lower number gets the motor moving.

Does the motor rotate Faster or Slower than in the last test for a given entered number?

Explain your answer. It rotates faster since the number of overflows has been reduced to 1.

**What is the value of the PWM signal’s Duty Cycle?**

In order to calculate the Duty Cycle of the PWM signal we must first determine the period of the signal. In the code we are using the Timer0 overflow interrupt to set the period. Timer0 is an eight bit timer so it can count from 0 to 255 before it “overflows to 0” on the next count. In the case where “if(numOverflows == 5)” the period of the PWM signal is the time it takes for the timer to go through 5 overflows. With the if statement changed to “if(numOverflows == 1)” the period of the PWM signal is the time it takes the timer to go through the 256 steps to produce one overflow.

The second bit of information needed for the Duty Cycle calculation is the time the PWM signal is High. In the code the time the PWM signal is High can be determined from the value in the OCR0A Register and the Timer0 settings.

Recall from previous lectures:

Where:

*Clock Period =*

-> *Clock Period* in this case refers to the clock after it has been divided by the Prescaler.

In our case:

*Clock Period* = (1/(16MHz)) x 256 = 62.5nS x 256 = 16uS.

(equation 4)

Period of PWM Sig = Clock\_Period x # of counts to overflow x # overflows

= 16uS x 256 x 5= 20.48 mS

Equation 1 from above can be rearranged so that the amount of time it takes (Target Time Interval) for the Timer to count up to the number (Target Timer Count) stored in the OCR0A, or the time it takes for an overflow is given by:

*Target Time Interval* = (Target Timer Count +1) x Clock Period.

Use equation 5 to calculate the time that the PWM signal is High and the period of the PWM signal for the “if(numOverflows == 5)” case.

576 uS

The Duty Cycle is calculated with the equation below:

Duty Cycle (%) = (Time PWM signal is High **/** Period of PWM signal) x 100%

For the “if(numOverflows == 5)” case with OCR0A = 35:

Time the PWM signal is High = \_\_\_\_\_\_576 uS\_\_\_\_\_\_\_\_\_\_\_\_\_

The period of the PWM signal = \_\_\_\_\_\_\_20.48 mS\_\_\_\_\_\_\_\_\_\_\_

The Duty Cycle of the PWM signal =\_\_\_\_\_2.81%\_\_\_\_\_\_\_\_\_\_

For the “if(numOverflows == 5)” case with OCR0A = 155:

Time the PWM signal is High = \_\_\_\_\_\_\_\_2.5 mS\_\_\_\_\_\_\_\_\_\_\_

The period of the PWM signal = \_\_\_\_\_\_\_634.88 mS\_\_\_\_\_\_\_\_\_\_\_

The Duty Cycle of the PWM signal =\_\_\_\_\_39.3%\_\_\_\_\_\_\_\_\_\_

Lab Sign Off\_\_\_\_\_2.71\_\_\_\_\_\_\_\_\_\_\_

**Part 2 – Control of a Servo Motor**

# Introduction

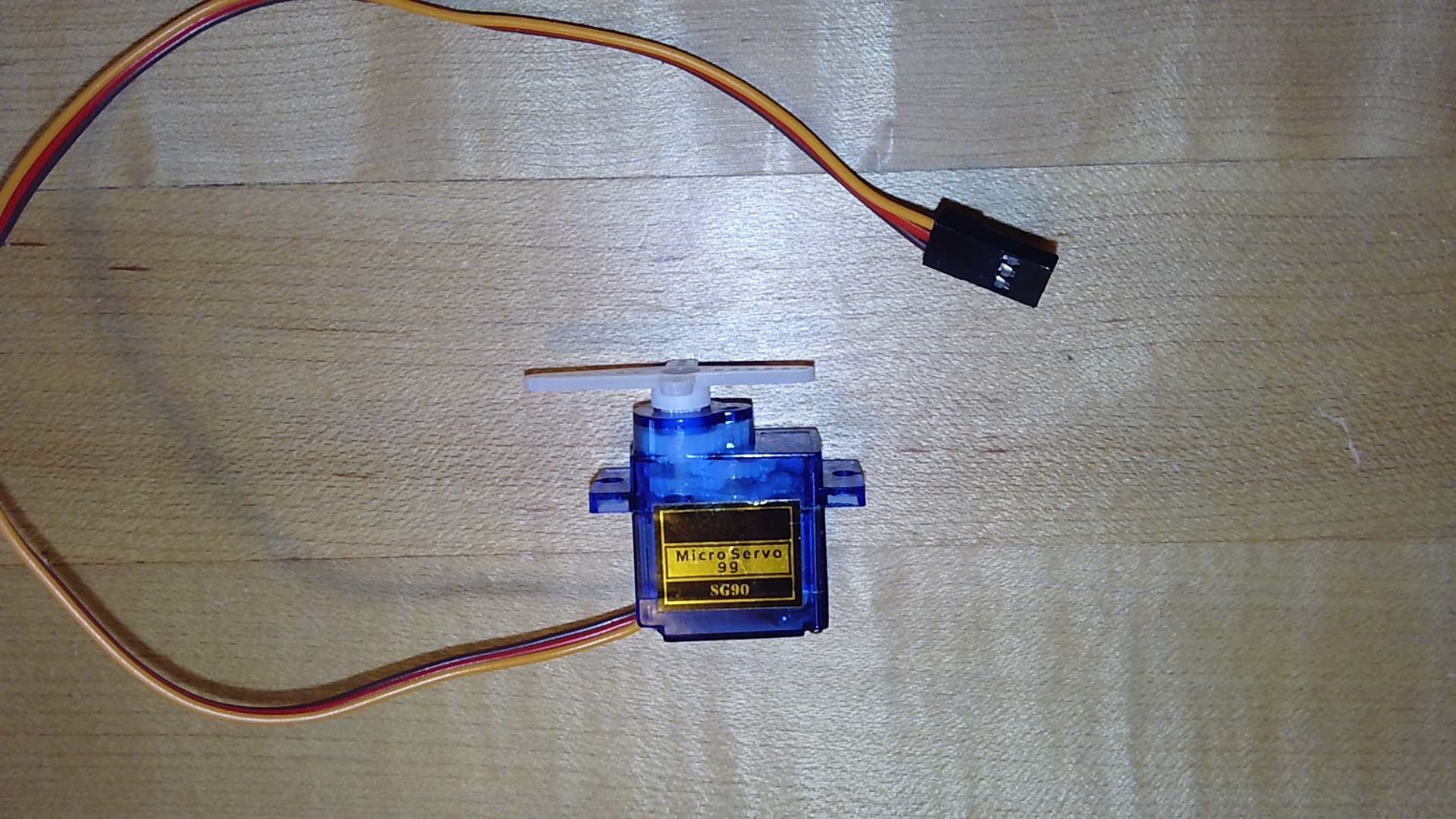
In this Part of the Lab we will use the Servo Motor in the Kit. We will investigate the basic operating principles of the servo motor.

|  |
| --- |
| **PARTS LIST** |
| UNO |
| Protoboard |
| Power Supply |
| Servo Motor |
| wires |

# SERVO MOTOR

The picture below shows the SERVO MOTOR that is supplied in the Kit. Note that only three wires are connected to the Servo Motor (orange – control signal, red - +5Volts, brown – ground). Servos are positioning motors. They do not continuously rotate like the motors used in Part 1 of this Lab. Our servo rotates between 0 and 180 degrees. The rotational position can be set to the nearest 1 or 2 degrees by the width of the control pulse sent over the orange wire.

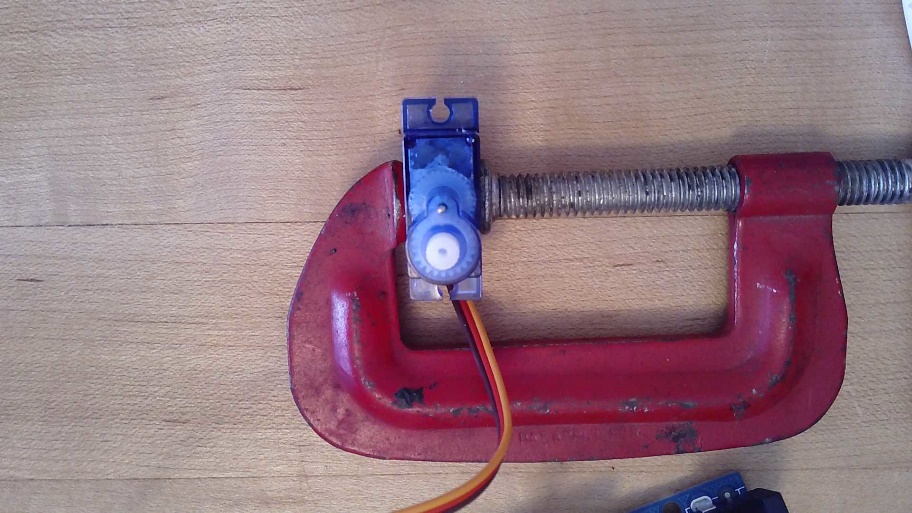
The internal positioning circuitry of the servo motor makes it straight forward to use with a microcontroller. A control signal is sent to the servo motor over the orange wire. The width of the control signal determines the position of the servo motor’s arm (called a “horn”). A pulse width of about 0.5 mS will rotate the arm clockwise (CW) – as seen looking down on the servo - to its maximum extent. A pulse of nominal width 2.5 mS will rotate the servo arm counter clockwise (CCW) to its maximum extent in that direction. The control pulses should occur every 15 to 25 mS or so. If the control pulses don’t occur within the required time interval the circuitry inside the servo will disconnect from the servo. The servo arm is free to turn if the disconnect happens. The position of the servo arm is dependent on the width of the control pulse and not so much on the time between the pulses (the period of the control pulses).



Servo Motor Setup

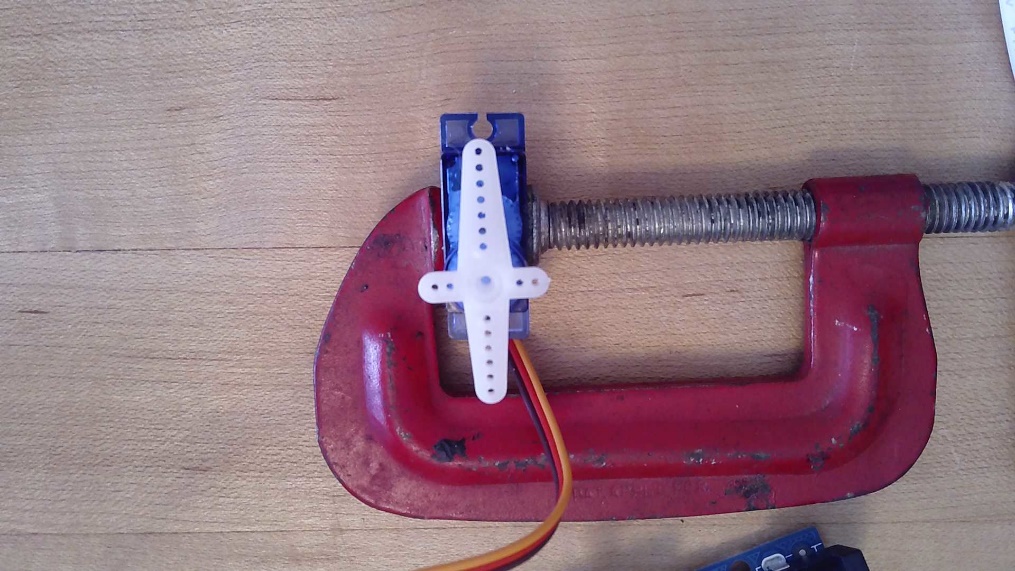
Plug the power supply into the protoboard. We will be using the 5V side of the power supply. Connect the ground from the power supply to ground of the UNO Board. Connect the “Brown” wire of the servo motor to ground. The “Red” wire of the servo should be connected to 5V from the power supply – not the UNO board’s 5V supply. The “Orange” wire from the servo should be connected to Digital I/O 12 on the edge of the UNO board.

The servo motor should look like this:



Notice that the servo motor arm has not been mounted to the shaft of the servo.

Power up the protoboard power supply connect the UNO to your computer with the USB cable. Use the same program that you used in Part 1 of this Lab. Input the number “35” when asked to input a speed. The servo will rotate to its most clockwise (CW) position. Press the large servo arm onto the shaft of the motor so that it is in the position shown below:



Now enter “155” as the speed. The servo arm should rotate 180 degrees counter clockwise (CCW) to its other limit. What value should be entered to position the arm at the 90 degree position?

95

Lab Sign Off\_\_\_\_\_\_\_\_\_\_\_2.71\_\_\_\_\_\_\_\_\_\_\_\_\_

***Completing the Lab***

Demo the DC Motor and the servo motor to the instructor and get checked off for this lab. Send in the lab handout with all questions answered (the word document) to the Assignments Section of Canvas.

Take a look at a microcontroller ad from the 1970’s on the next page. Notice the amount of memory that came with the microcontroller.

