

## Lab 06

# Stepper Motor Pulley Control

Logan Barber, Ian Nail

April 18, 2022

*Ian Nail*

*Logan Barber*

ME-4370 - Stephen Canfield

# 1 Executive Summary

In this lab, the objective is to use the Arduino Mega 2560 to develop a product that makes use of the servo motors with a Pulse Width Modulated (PWM) signal. Two servo motors are used in making a robot arm. A joystick was used as the controller to the robot arm.

Servos are extremely useful in robotics. The motors are small, have built-in control circuitry, and are extremely powerful for their size. The output shaft of the servo is capable of traveling around 180 degrees.

To make use of the PWM functionality of the Mega 2560, the correct clock and compare bits need to be set. The data sheet of the ATMEGA 2560 gives instructions on the register values required for the different options. The fast PWM with a 256 pre-scaler is used. Two PWM signals are used to control two servo motors. One servo motor on PH6 and the other on PB5. The servo motors move according to the duty cycle of the PWM signal. If the duty cycle is 0%, the motor moves to the 0 degree position. If the duty cycle is 50%, it moves to the 180 degree position. The duty cycle is changed by setting the OCRxx bit to a new value, where xx is the timer and pin connection. OCR1A is Timer 1 and channel A that is mapped to PB5.

The joystick is what controls the movement of the motors. The joystick values are read from the analog to digital pins. The joystick is capable of y direction and x direction. The voltages are sent to the ADC pins and converted into signals to use in software. The software takes the signals and makes decisions on which direction to move the servo motors.

# 2 Analog to Digital Conversion

The Joystick is powered with 5 Vdd and GND. When the stick is moved analog voltages change on Vrx and Vry according to the movement of the stick. These analog values are sent to the ADC pins and the conversion to a digital number happens. That digital number is displayed onto the LCD. The software makes decisions on if the servo motor should be moved when the joystick is pressed.

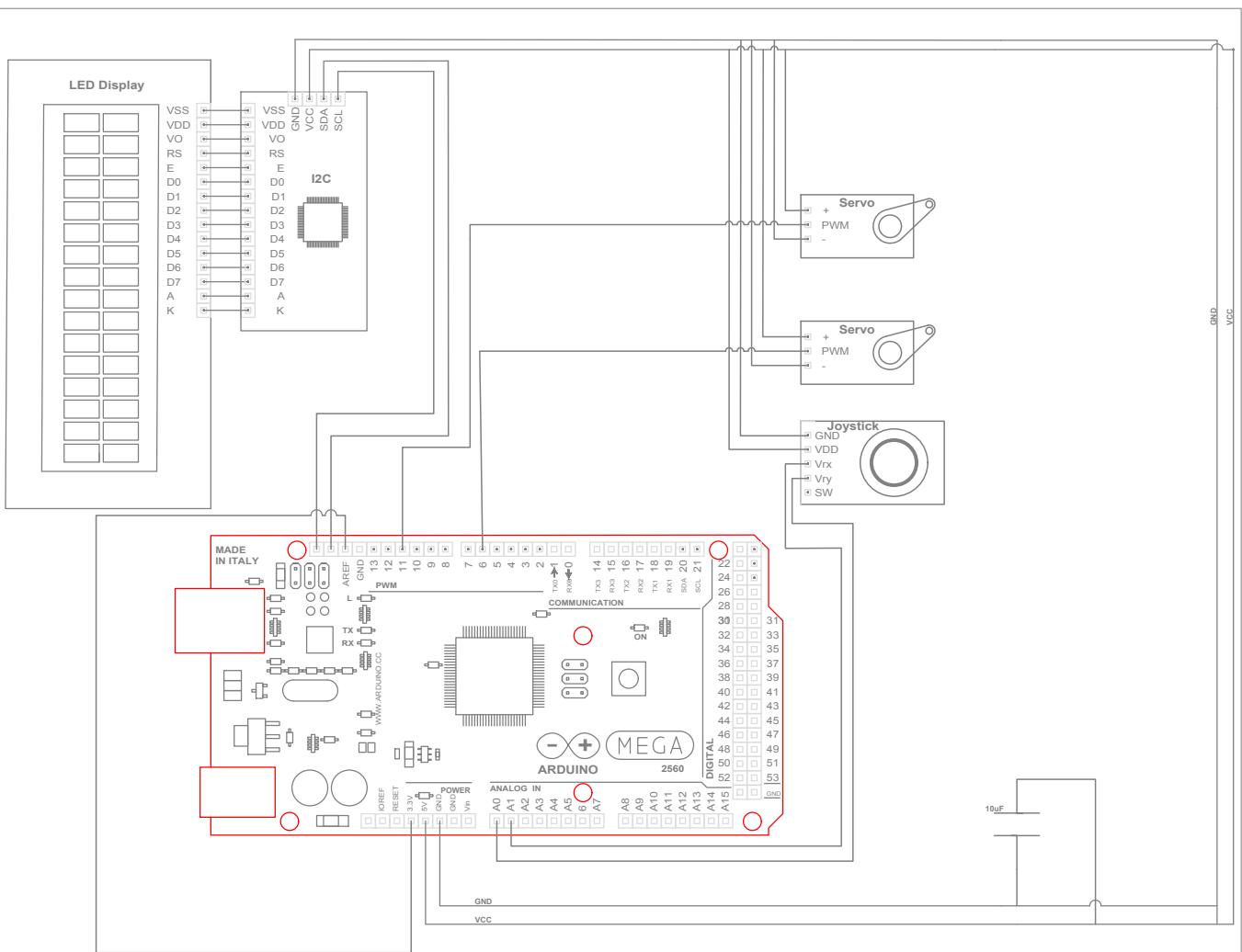


Figure 1: Circuit Diagram

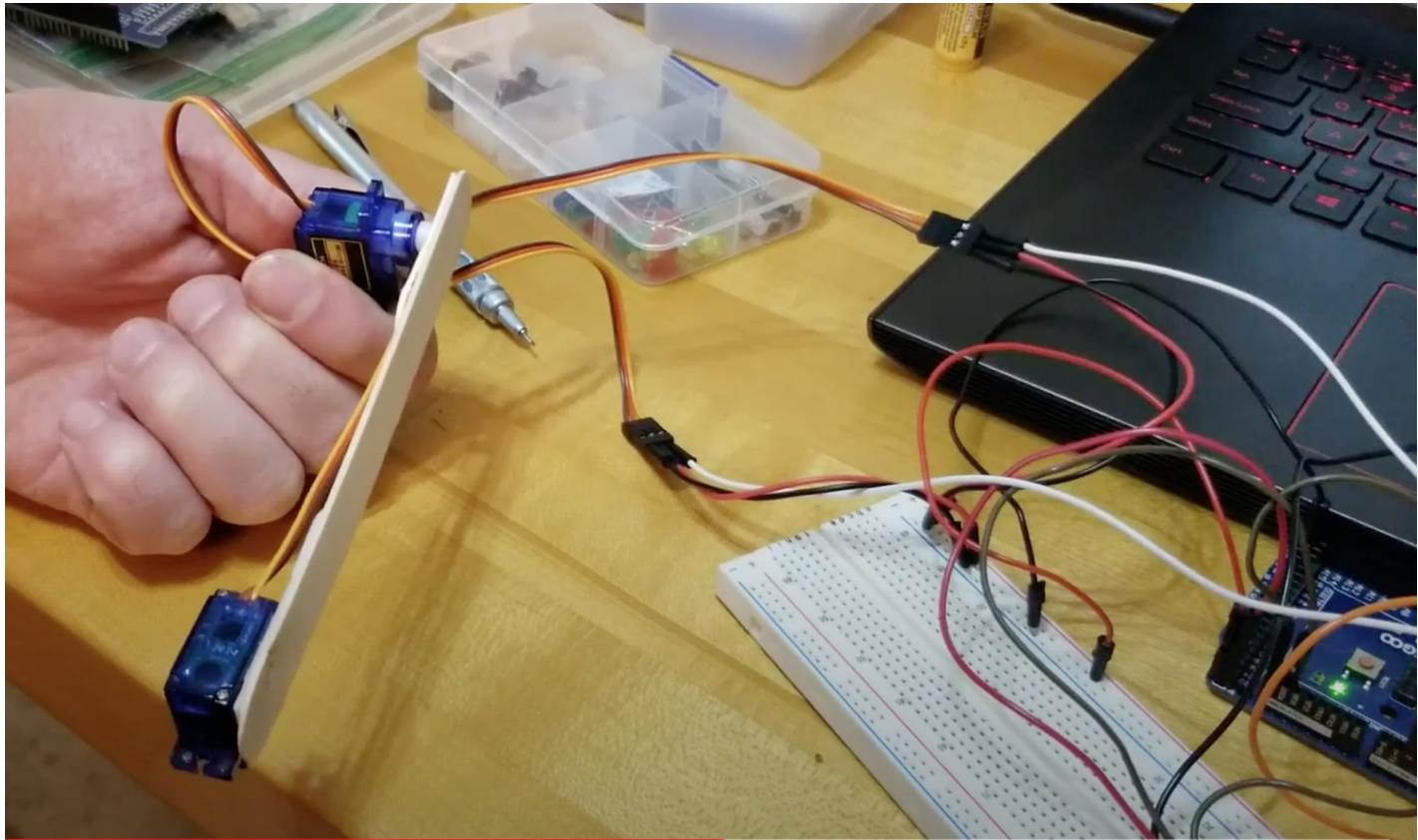


Figure 2: Breadboard and Connections

### 3 Source Code

```
0 // AUTHORS: A. LOGAN BARBER, IAN NAIL
1 // FILE NAME: Lab06.ino
2 // LAST UPDATED: 17 APRIL 2022
3 /*
4 * PURPOSE: THIS FILE IS OUR SOLUTION FOR ME 4370 >> LAB 6.
5 */
6 /*
7 * STEPPER MOTOR DRIVER:
8 *
9 * STEPPER MOTOR DRIVER PIN: IN1
10 * PORT: B
11 * PIN: PB0
12 * DIGITAL PIN: 53
13 *
14 * STEPPER MOTOR DRIVER PIN: IN2
15 * PORT: B
16 * PIN: PB1
17 * DIGITAL PIN: 52
18 *
19 * STEPPER MOTOR DRIVER PIN: IN3
20 * PORT: B
21 * PIN: PB2
22 * DIGITAL PIN: 51
23 *
24 * STEPPER MOTOR DRIVER PIN: IN4
25 * PORT: B
26 * PIN: PB3
27 * DIGITAL PIN: 50
28 */
29 /*
30 * BUTTON0:
31 * DIGITAL PIN: 22
32 * PORT: A
33 * PORT PIN: 0
34 */
35 /*
36 * BUTTON1:
37 * DIGITAL PIN: 23
38 * PORT: A
39 * PORT PIN: 1
40 */
41 /*
42 *****
43 *****
44 *****
45 *****
46 #include <stdint.h> // ALLOWS TO SPECIFICATION OF THE BIT SIZE OF THE NUMBER
47 *****
48 *****
49 *****
50 #define DELAY 850
51 uint8_t u8_state = 0;
52 *****
53 *****
54 // SET UP FUNCTION
```

```

58 void setup()
{
    // SET UP PORT B PINS FOR OUTPUT
60    DDRB = 0x0F; // 0b00000111
61    PORTB = 0x00; // 0b00000000
62
63    // SETUP BUTTON PINS AS INPUTS
64    // ENABLE INTERNAL PULL-UP RESISTOR FOR BUTTONS
65    DDRA = 0x00; // 0b00000000
66    PORTA = 0x07; // 0b00000011
}
67
68
70 // RUN THIS CODE FOREVER
void loop()
{
71
72    // IF BUTTON0 IS LOW CHANGE STATE
73    if((PINA & 0x01) == 0x01)
74    {
75        u8_state = 1;
76    }
77
78    // ELSE IF BUTTON1 IS LOW CHANGE STATE
79    else if((PINA & 0x02) == 0x02)
80    {
81        u8_state = 2;
82    }
83    else{
84        u8_state = 0;
85    }
86
87    // FORWARD
88    if( u8_state == 1)
89    {
90        // TURN ON IN1
91        PORTB |= 0x01; // 0b00000001
92        delayMicroseconds(DELAY);
93
94        // TURN OFF IN4
95        PORTB &= ~(0x08); // 0b11110111
96        delayMicroseconds(DELAY);
97
98        // TURN ON IN2
99        PORTB |= 0x02; // 0b00000010
100       delayMicroseconds(DELAY);
101
102        // TURN OFF IN1
103        PORTB &= ~(0x01); // 0b11111110
104        delayMicroseconds(DELAY);
105
106        // TURN ON IN3
107        PORTB |= 0x04; // 0b000000100
108        delayMicroseconds(DELAY);
109
110        // TURN OFF IN2
111        PORTB &= ~(0x02); // 0b11111101
112        delayMicroseconds(DELAY);
113
114        // TURN ON IN4
115
116

```

```

118     PORTB |= 0x08; // 0b00001000
      delayMicroseconds(DELAY);

120     // TURN OFF IN3
121     PORTB &= ~(0x04); // 0b11111011
      delayMicroseconds(DELAY);
122   }

124   // BACKWARD
125   else if ( u8_state == 2)
126   {
127     // TURN ON IN4
128     PORTB |= 0x08; // 0b00001000
      delayMicroseconds(DELAY);

129     // TURN ON IN3
130     PORTB |= 0x04; // 0b00000100
      delayMicroseconds(DELAY);

131     // TURN OFF IN4
132     PORTB &= ~(0x08); // 0b11110111
      delayMicroseconds(DELAY);

133     // TURN ON IN2
134     PORTB |= 0x02; // 0b00000010
      delayMicroseconds(DELAY);

135     // TURN OFF IN3
136     PORTB &= ~(0x04); // 0b11111011
      delayMicroseconds(DELAY);

137     // TURN ON IN1
138     PORTB |= 0x01; // 0b00000001
      delayMicroseconds(DELAY);

139     // TURN OFF IN2
140     PORTB &= ~(0x02); // 0b11111101
      delayMicroseconds(DELAY);

141     // TURN OFF IN1
142     PORTB &= ~(0x01); // 0b11111110
      delayMicroseconds(DELAY);
143   }
144 }
```

..../Lab06.ino