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, were 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 take 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.

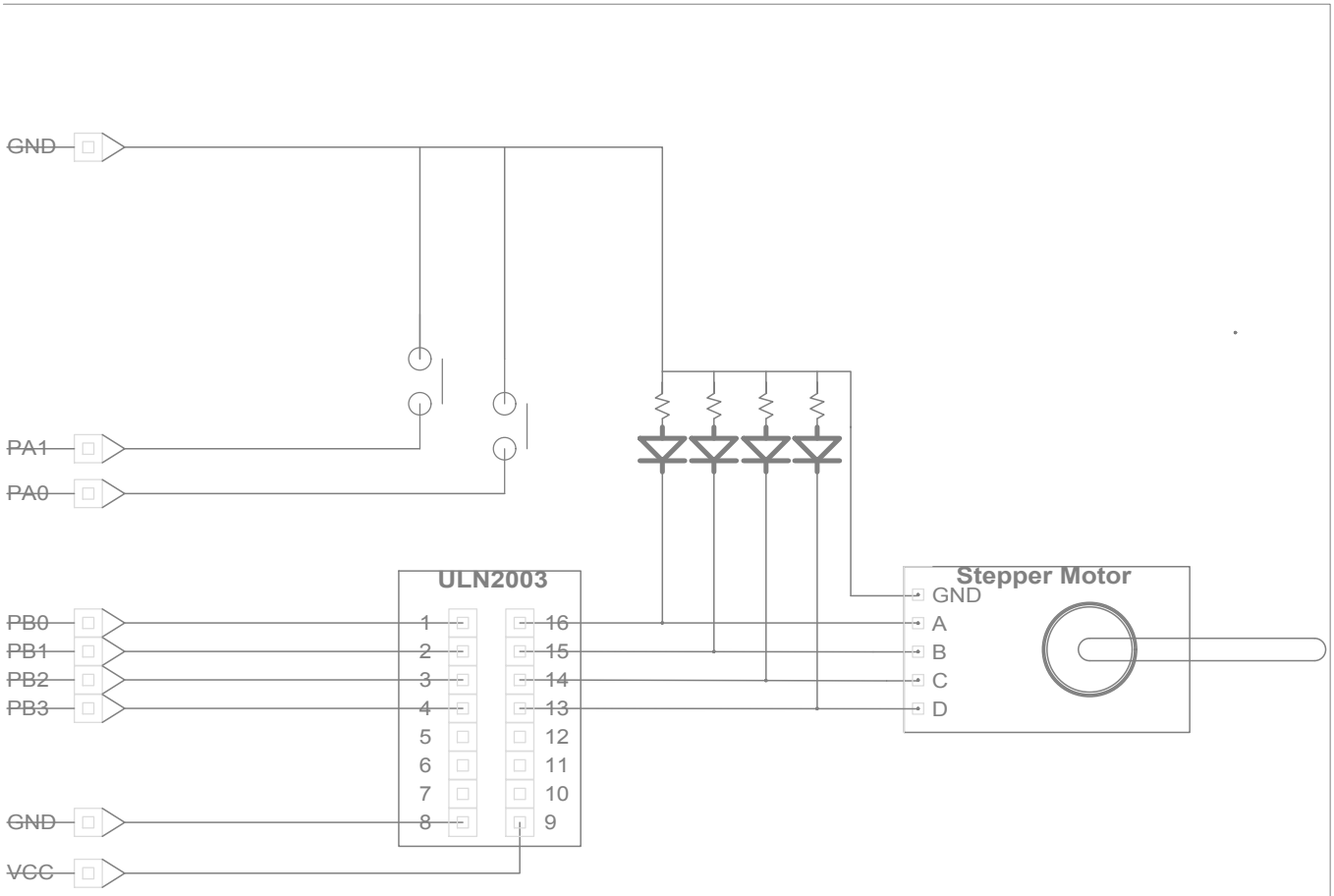

Figure 1: Circuit Diagram

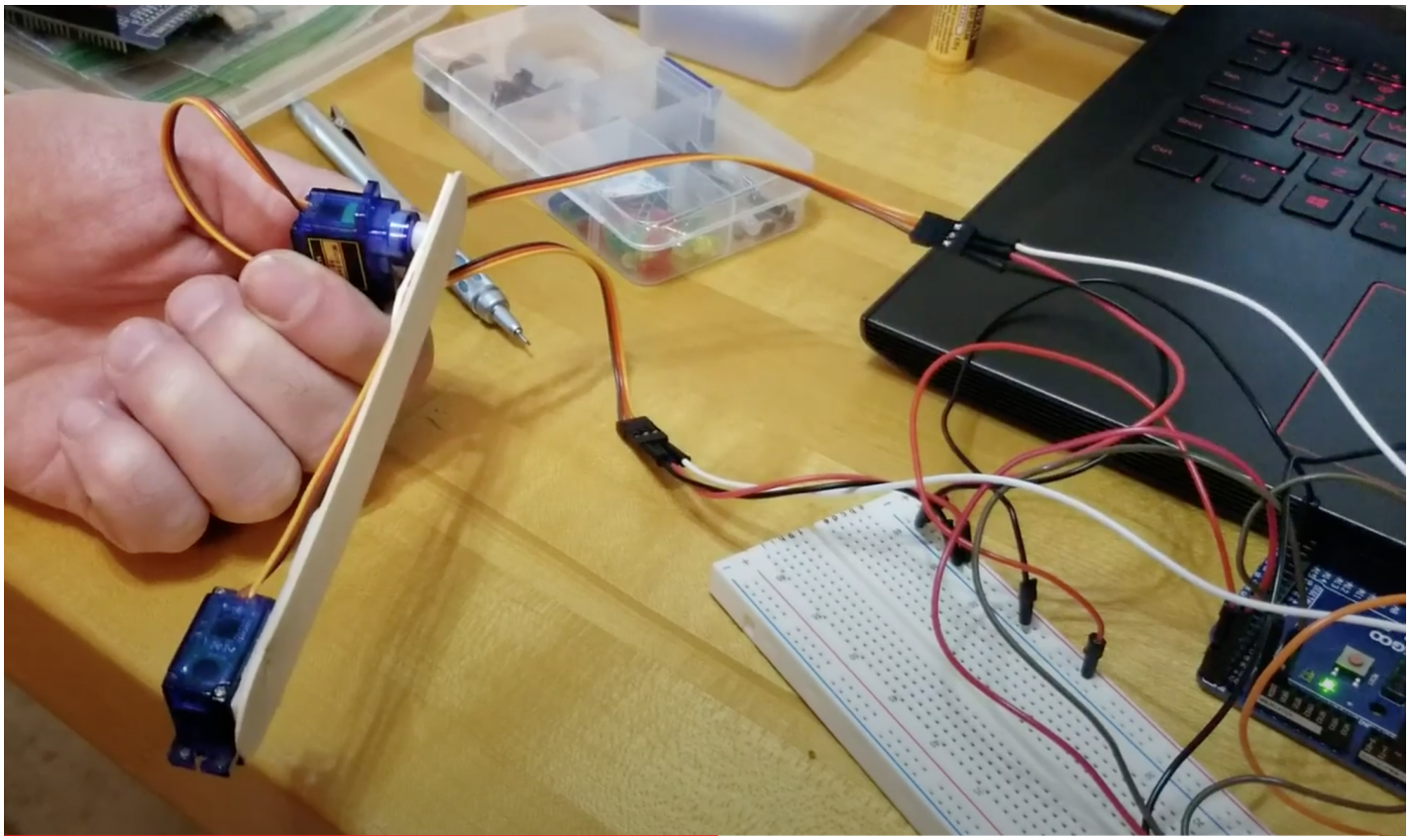

Figure 2: Breadboard and Conections

3 Source Code

```
// AUTHORS: A. LOGAN BARBER, IAN NAIL
// FILE NAME: Lab06.ino
// LAST UPDATED: 17 APRIL 2022
/*
 * PURPOSE: THIS FILE IS OUR SOLUTION FOR ME 4370 >> LAB 6.
 */

/*
 * STEPPER MOTOR DRIVER:
 *
 * STEPPER MOTOR DRIVER PIN: IN1
 *      PORT: B
 *      PIN: PB0
 *      DIGITAL PIN: 53
 *
 * STEPPER MOTOR DRIVER PIN: IN2
 *      PORT: B
 *      PIN: PB1
 *      DIGITAL PIN: 52
 *
 * STEPPER MOTOR DRIVER PIN: IN3
 *      PORT: B
 *      PIN: PB2
 *      DIGITAL PIN: 51
 *
 * STEPPER MOTOR DRIVER PIN: IN4
 *      PORT: B
 *      PIN: PB3
 *      DIGITAL PIN: 50
 */

 /*
  * BUTTON0:
  * DIGITAL PIN: 22
  * PORT: A
  * PORT PIN: 0
  */

 /*
  * BUTTON1:
  * DIGITAL PIN: 23
  * PORT: A
  * PORT PIN: 1
  */

/********** I N C L U D E S **********/
#include <stdint.h> // ALLOWS TO SPECIFICATION OF THE BIT SIZE OF THE NUMBER


/********** D E F I N E S **********/
#define DELAY 850
uint8_t u8_state = 0;

/********** G L O B A L  V A R I A B L E S **********/


// SET UP FUNCTION
```

```
void setup()
{
    // SET UP PORT B PINS FOR OUTPUT
    DDRB = 0x0F; // 0b00001111
    PORTB = 0x00; // 0b00000000

    // SETUP BUTTON PINS AS INPUTS
    // ENABLE INTERNAL PULL-UP RESISTOR FOR BUTTONS
    DDRA = 0x00; // 0b00000000
    PORTA = 0x03; // 0b00000011
}


// RUN THIS CODE FOREVER
void loop()
{
    // IF BUTTON0 IS LOW CHANGE STATE
    if((PINA & 0x01) == 0x00)
    {
        u8_state = 1;
    }

    // ELSE IF BUTTON1 IS LOW CHANGE STATE
    else if((PINA & 0x02) == 0x00)
    {
        u8_state = 2;
    }
    else
    {
        u8_state = 0;
    }

    // FORWARD
    if(u8_state == 1)
    {
        // TURN ON IN1
        PORTB |= 0x01; // 0b00000001
        delayMicroseconds(DELAY);

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

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

        // TURN OFF IN1
        PORTB &= ~(0x01); // 0b11111110
        delayMicroseconds(DELAY);

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

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

        // TURN ON IN4
```

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

120         // TURN OFF IN3
            PORTB &= ~(0x04); // 0b11111011
122         delayMicroseconds(DELAY);
    }
124
    // BACKWARD
126     else if(u8_state == 2)
    {
128          // TURN ON IN4
            PORTB |= 0x08; // 0b00001000
130         delayMicroseconds(DELAY);

132         // TURN ON IN3
            PORTB |= 0x04; // 0b00000100
134         delayMicroseconds(DELAY);

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

140         // TURN ON IN2
            PORTB |= 0x02; // 0b00000010
142         delayMicroseconds(DELAY);

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

148          // TURN ON IN1
            PORTB |= 0x01; // 0b00000001
150         delayMicroseconds(DELAY);

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

156         // TURN OFF IN1
            PORTB &= ~(0x01); // 0b11111110
158         delayMicroseconds(DELAY);
    }
160 }
```

../Lab06.ino