Lab 06

Stepper Motor Pulley Control

Logan Barber, Ian Nail

May 5, 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 a stepper motor. The stepper motor is the driving mechanism for a pulley system. Two buttons control the direction of the stepper motor. One of the buttons will make the pulley go up and the other will make the pulley go down.

Stepper motors are extremely useful in mechatronic systems. Stepper motors are not useful for high speed systems, but good for simple low speed systems; at low speeds, stepper motors have high torque. Stepper motors are useful due their very accurate open-loop control system, however, this comes at the expense of weight. Stepper motors are very heavy since they have magnets with coils of wire in them. An advantage of stepper motors is that, if one of the coils is powered, it takes a very large external force to rotate the shaft. This is because the electromagnetic force will lock onto the shaft's magnet causing this locking effect.

We used the ULN2003 Stepper Motor Driver PCB to as the driver. The driver board accepts a four bit command from the microcontroller and in turn applies the necessary power pulse to step the motor. At the heart of the driver is a ULN2003AN integrated circuit. The board can supply between 5V to 12V to the motor from an independent power supply. It also has a bank of LED's that correspond to the input signals received from the controller. They provide a nice visual when stepping.

The motor steps when a specific combination of inputs are driven from the microcontroller. This is just a pulse of power, just enough to get the motor to step. This driver uses a very simple protocol. Applying a signal to an input pin causes power to be sent to the motor on a corresponding wire.

2 Notes on circuit diagram and breadboard connections.

The electrical components used in the stepper motor circuitry are as follows:

- Stepper Motor Model: 28KYJ-48
- Two push buttons
- ULN2003 PCB

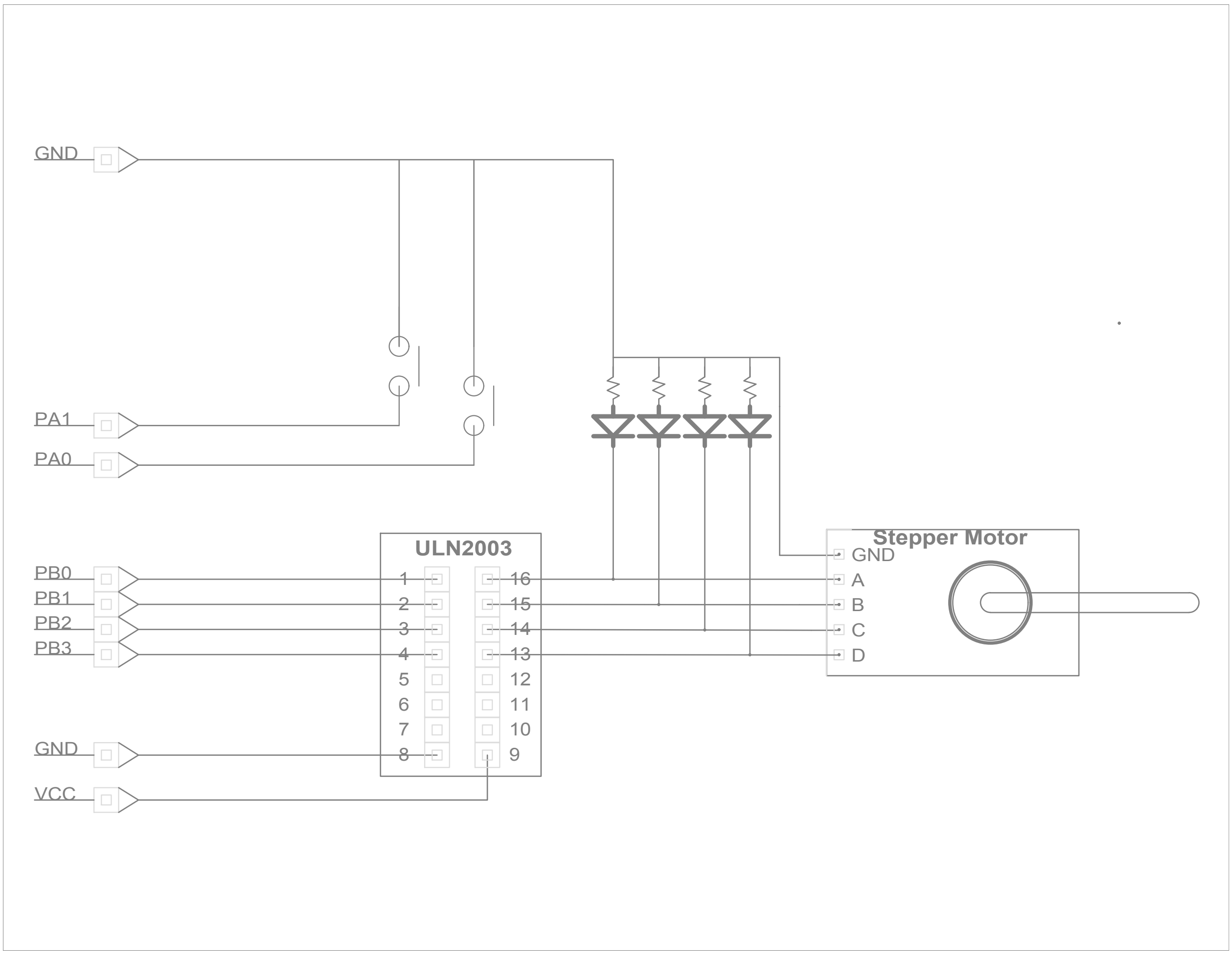

Figure 1: Circuit Diagram

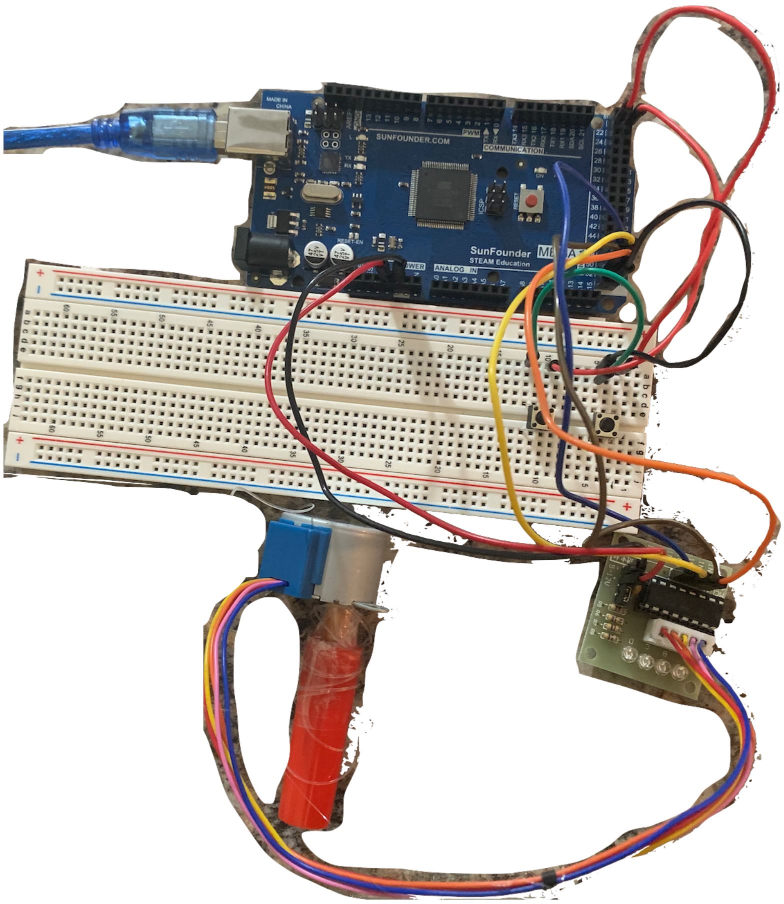

Figure 2: Breadboard and Connections

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
  */

/********** INCLUDES **********/
#include <stdint.h> // ALLOWS TO SPECIFICATION OF THE BIT SIZE OF THE NUMBER


/********** DEFINES **********/
#define DELAY 950
uint8_t u8_state = 0;

/********** GLOBAL VARIABLES **********/


// 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
        delayMicroseconds(DELAY);

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

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

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

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

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

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

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

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

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

../Lab06.ino