Brandon Wade

CPE301 – SPRING 2016

Design Assignment 5

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

Atmel Studio 7.0

ATMega328P Chip

Breadboard

DC Motor

Stepper Motor

Servo Motor

ILD74

ULN2003

TIP120

Resistors (varied sizes)

Servo

STEPPER

0v

5v

ULN2003

1 16

2 15

3 14

4 13

8 9

5v

5v

DC

TIP120

B C E

ILD74

1 8

2 7

Atmega328P

PC0 PD6

PB0

PB1

PB2

PB3

POT

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

// Brandon Wade

#define F\_CPU 8000000UL //XTAL = 8MHZ

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ENABLE 0

volatile uint8\_t ADCvalue; // Global variable, set to volatile if used withIS

void adc\_init (void)

{

ADMUX = 0; // use ADC0

ADMUX |= (1 << REFS0) | (1 <<REFS1); // use AVcc as the reference

ADMUX |= (1 << ADLAR); // Right adjust for 8 bit resolution

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (0 << ADPS0); // 64 prescale for 8Mhz

ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable

ADCSRB = 0; // 0 for free running mode

ADCSRA |= (1 << ADEN); // Enable the ADC

ADCSRA |= (1 << ADIE); // Enable Interrupts

ADCSRA |= (1 << ADSC); // Start the ADC conversion

}

ISR(ADC\_vect)

{

ADCvalue = ADCH; // only need to read the high value for 8 bit

}

void timer\_init (void)

{

TCCR0A = 1<<COM0A1|1<<COM0A0|1<<WGM00; // PWM mode, set/reset on OCRxA

TCCR0B = 0<<WGM02 |0<<CS02|0<<CS01|1<<CS00; // PWM mode with 1 pre scalar

}

int main()

{

float f;

float i;

timer\_init(); // initialize timer

adc\_init(); // initialize adc

sei(); // enable global interrupts

DDRD = 0xFF; //make PORTB output pin

DDRC = 0x00;

while(1)

{

f = (float)ADCvalue \* (1.1 / 1024) \* .01;

i = f \* 1000; // conversion to change duty cycle

OCR0A = i \* 210; // update duty cycle over time

}

return 0 ;

}

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | INITIAL CODE OF TASK 2/A |  |  |

// Brandon Wade

#define F\_CPU 8000000UL //XTAL = 8MHZ

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ENABLE 0

volatile uint8\_t ADCvalue; // Global variable, set to volatile if used withIS

void adc\_init (void)

{

ADMUX = 0; // use ADC0

ADMUX |= (1 << REFS0) | (1 <<REFS1); // use AVcc as the reference

ADMUX |= (1 << ADLAR); // Right adjust for 8 bit resolution

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (0 << ADPS0); // 64 prescale for 8Mhz

ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable

ADCSRB = 0; // 0 for free running mode

ADCSRA |= (1 << ADEN); // Enable the ADC

ADCSRA |= (1 << ADIE); // Enable Interrupts

ADCSRA |= (1 << ADSC); // Start the ADC conversion

}

ISR(ADC\_vect)

{

ADCvalue = ADCH; // only need to read the high value for 8 bit

}

void timer\_init (void)

{

OCR0A = 125; // 10% duty start

TCCR0A = 1<<COM0A1|1<<COM0A0|1<<WGM00; // PWM mode, set/reset on OCRxA

TCCR0B = 0<<WGM02 |0<<CS02|0<<CS01|1<<CS00; // PWM mode with 1 pre scalar

}

void delay(int j)

{

for (int i = 0; i <j; i++)

{

\_delay\_us(1); // generates variable timer based off j passed

}

}

int main()

{

adc\_init(); // initialize adc

sei(); // enable global interrupts

DDRC = 0x00;

DDRB = 0xFF; // output ports

while(1)

{

PORTB = 0X06;

delay(ADCvalue \* 169 + 20000); // empirically derived delay

PORTB = 0x0C;

delay(ADCvalue \* 169 + 20000); // empirically derived delay

PORTB = 0x09;

delay(ADCvalue \* 169 + 20000); // empirically derived delay

PORTB = 0x03;

delay(ADCvalue \* 169 + 20000); // empirically derived delay

}

return 0 ;

}

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | INITIAL CODE OF TASK 3/A |  |  |

// Brandon Wade

#define F\_CPU 8000000UL //XTAL = 8MHZ

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ENABLE 0

volatile uint8\_t ADCvalue; // Global variable, set to volatile if used withIS

void adc\_init (void)

{

ADMUX = 0; // use ADC0

ADMUX |= (1 << REFS0) | (1 <<REFS1); // use AVcc as the reference

ADMUX |= (1 << ADLAR); // Right adjust for 8 bit resolution

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (0 << ADPS0); // 64 prescale for 8Mhz

ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable

ADCSRB = 0; // 0 for free running mode

ADCSRA |= (1 << ADEN); // Enable the ADC

ADCSRA |= (1 << ADIE); // Enable Interrupts

ADCSRA |= (1 << ADSC); // Start the ADC conversion

}

ISR(ADC\_vect)

{

ADCvalue = ADCH; // only need to read the high value for 8 bit

}

void timer\_init (void)

{

OCR0A = 125; // 10% duty start

TCCR0A = 1<<COM0A1|1<<COM0A0|1<<WGM00; // PWM mode, set/reset on OCRxA

TCCR0B = 0<<WGM02 |0<<CS02|0<<CS01|1<<CS00; // PWM mode with 1 pre scalar

}

void delay(int j)

{

for (int i = 0; i <j; i++)

{

\_delay\_us(1);

}

}

int main()

{

adc\_init(); // initialize adc

sei(); // enable global interrupts

//PORTD pins as input

DDRD = 0x00;

//Enable internal pull ups

PORTD = 0xFF;

//Set PORTB1 pin as output

DDRB = 0xFF;

//TOP = ICR1;

//output compare OC1A 8 bit non inverted PWM

//Clear OC1A on Compare Match, set OC1A at TOP

//Fast PWM

//ICR1 = 20000 defines 50Hz pwm

ICR1 = 20000;

TCCR1A|=(0<<COM1A0)|(1<<COM1A1)|(0<<COM1B0)|(0<<COM1B1)|(0<<FOC1A)|(0<<FOC1B)|(1<<WGM11)|(0<<WGM10); //TCCR1A = 0x82

TCCR1B|=(0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(1<<WGM12)|(0<<CS12)|(1<<CS11)|(0<<CS10);

while(1)

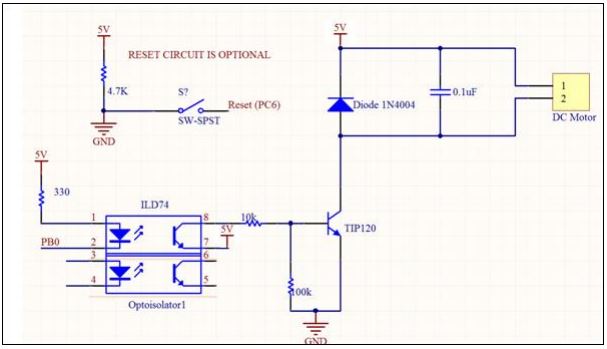
{

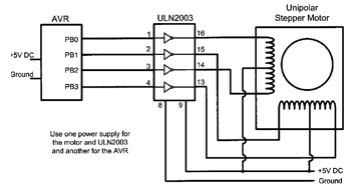
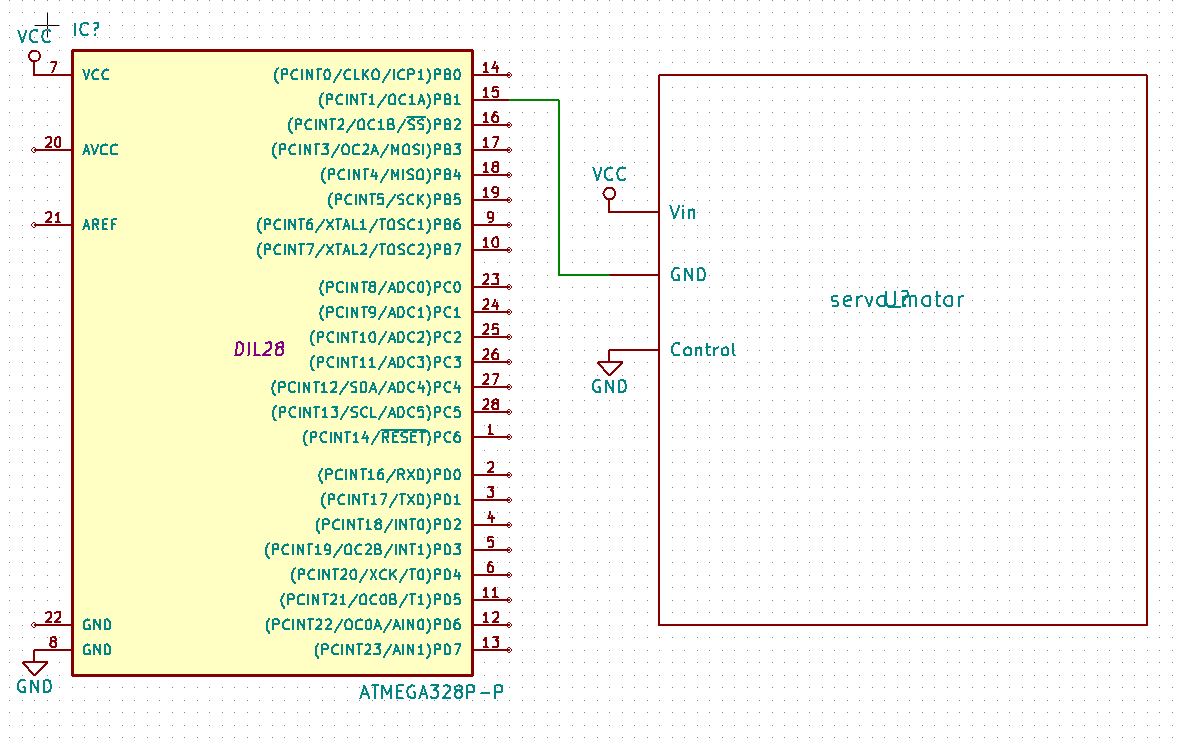
OCR1A=ADCvalue \* 21 + 100; // empirically derived values to change duty cycle

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |

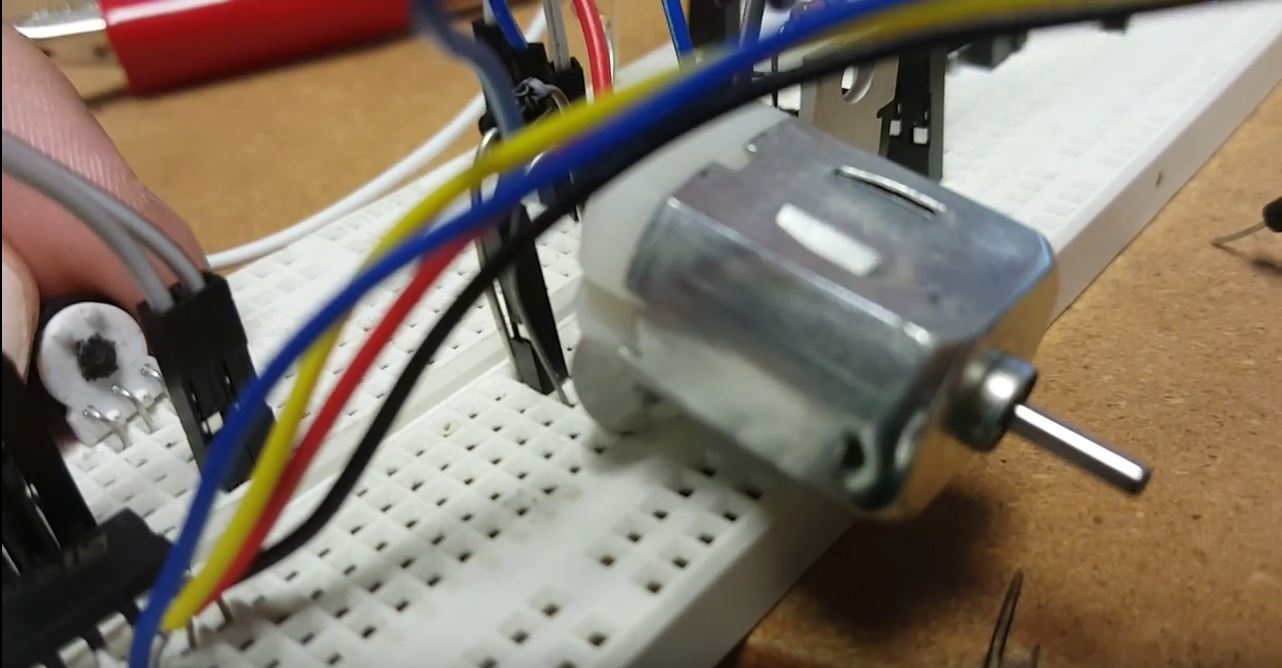




DC (top), Stepper (middle), and Servo (bottom) schematics. All tasks included potentiometer connected to PC0 pin of ATMega328P

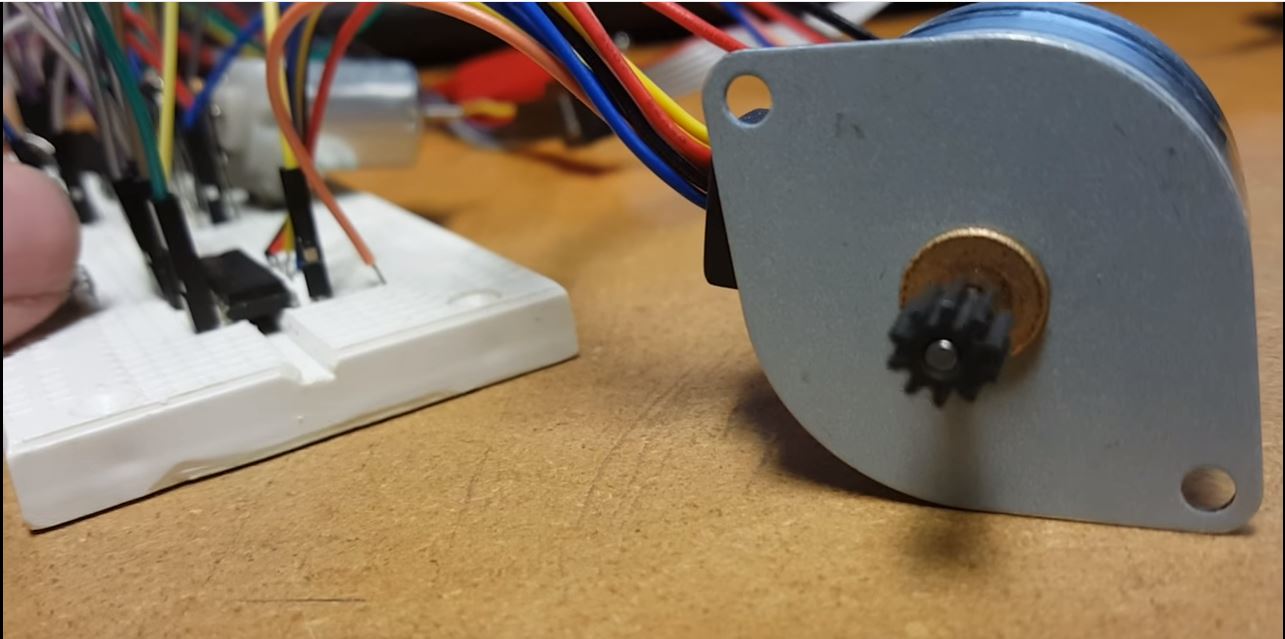
|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

TASK 1/A: DC Motor



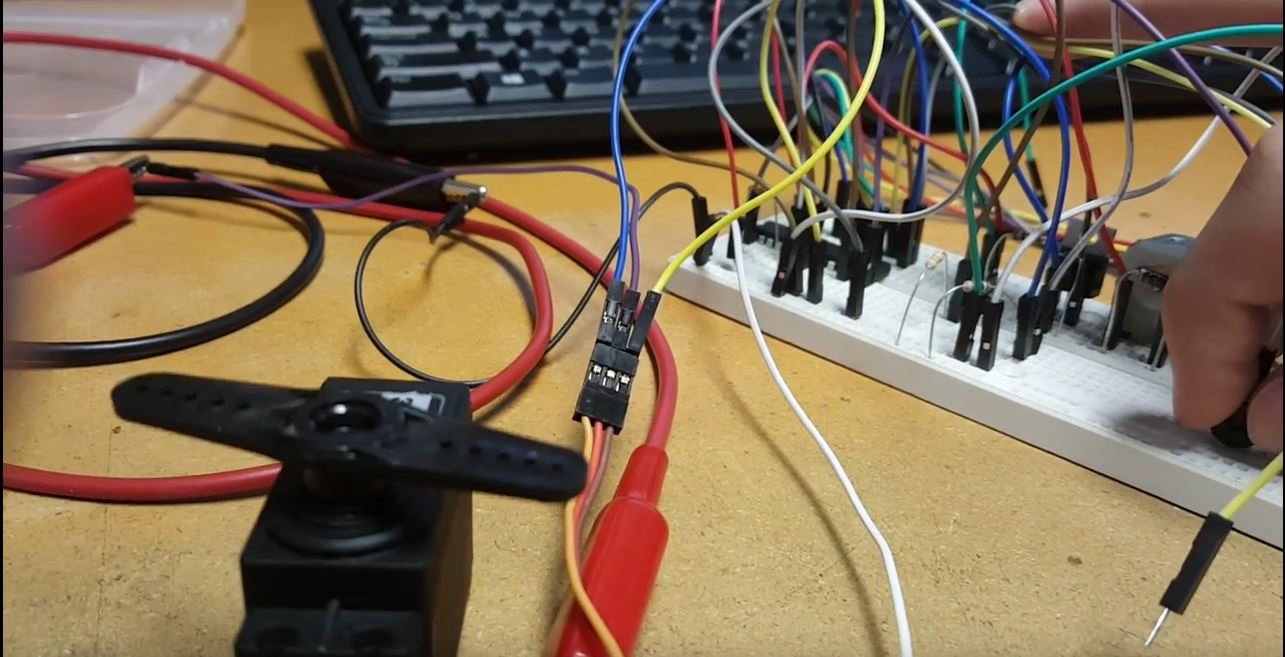
Potentiometer on the left, controls speed of DC motor.

TASK 2/A: Stepper Motor



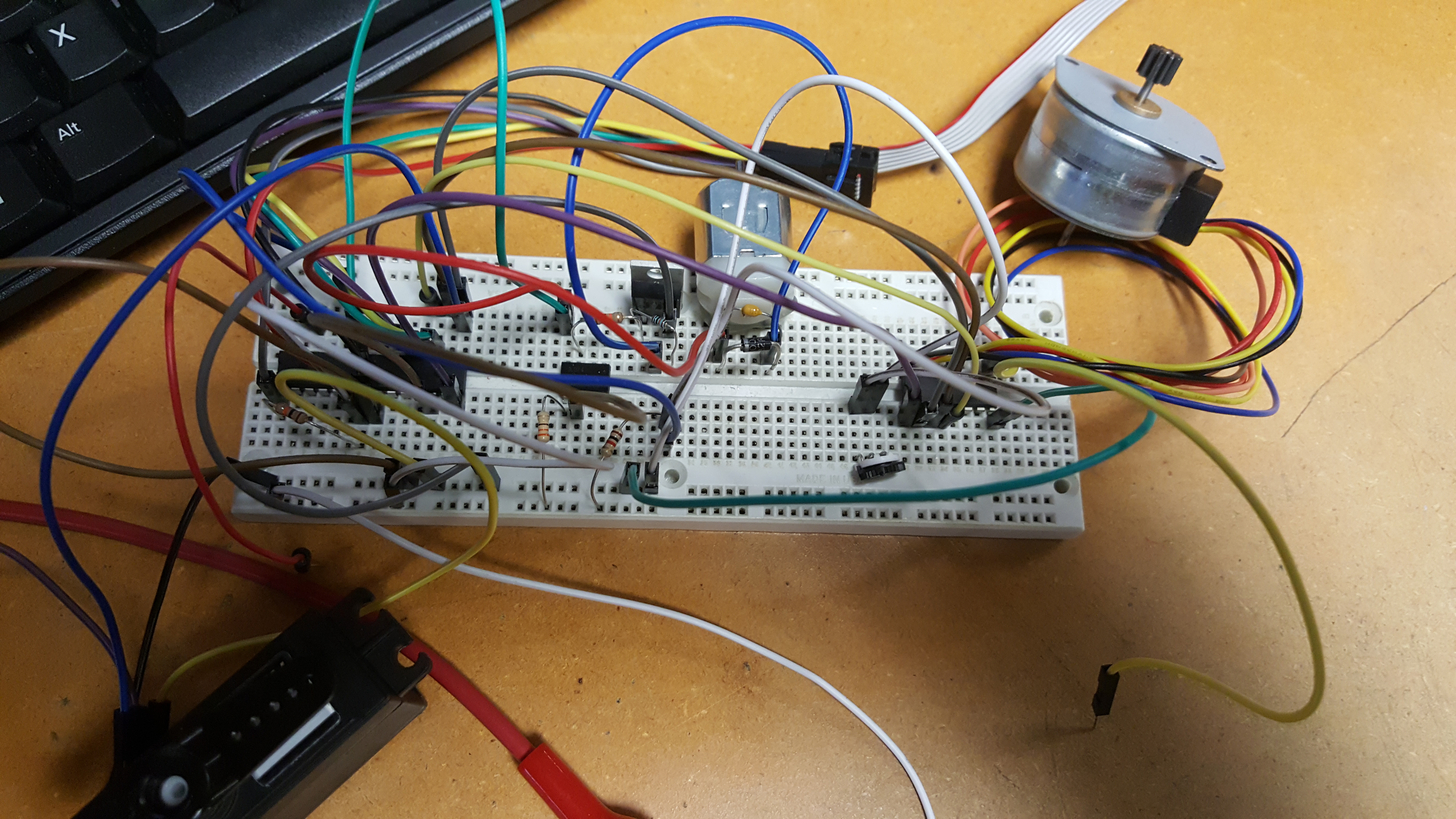
Potentiometer on the left, controls speed of stepper motor. ULN2003 located to right of potentiometer.

TASK 3/A: Servo Motor



Potentiometer on the right, controls position of servo motor. Blue wire indicates control signal, purple wire is 6V, and yellow wire is GND (connectors in middle).

|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT OF EACH DEMO |  |  |



All demonstrations implemented on single board. Servo to the left, DC in the middle, and Stepper towards the right. Potentiometer is located just below ULN2003 device.

|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| Task1 - <https://www.youtube.com/watch?v=bvW-NKz1yRw>  Task 2 - <https://www.youtube.com/watch?v=rrOxQSRi89c>  Task 3 - <https://www.youtube.com/watch?v=oAY2EDIXV5k> | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| <https://github.com/wadeb1/KF3HF6ZFMP.git> | | | |

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Brandon Wade