CPE301 – SPRING 2019

Design Assignment 4B

Student Name: Geovanni Portillo
Student #: 8000603824
Student Email: portig1@unlv.nevada.edu
Primary Github address: https://github.com/portig1/submissions_E
Directory: submissions_E/DA/LAB4B/

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmel Studio 7
ATmega328PB Xplained mini

Figure 1-1. ATmega328P Xplained Mini Headers and Connectors

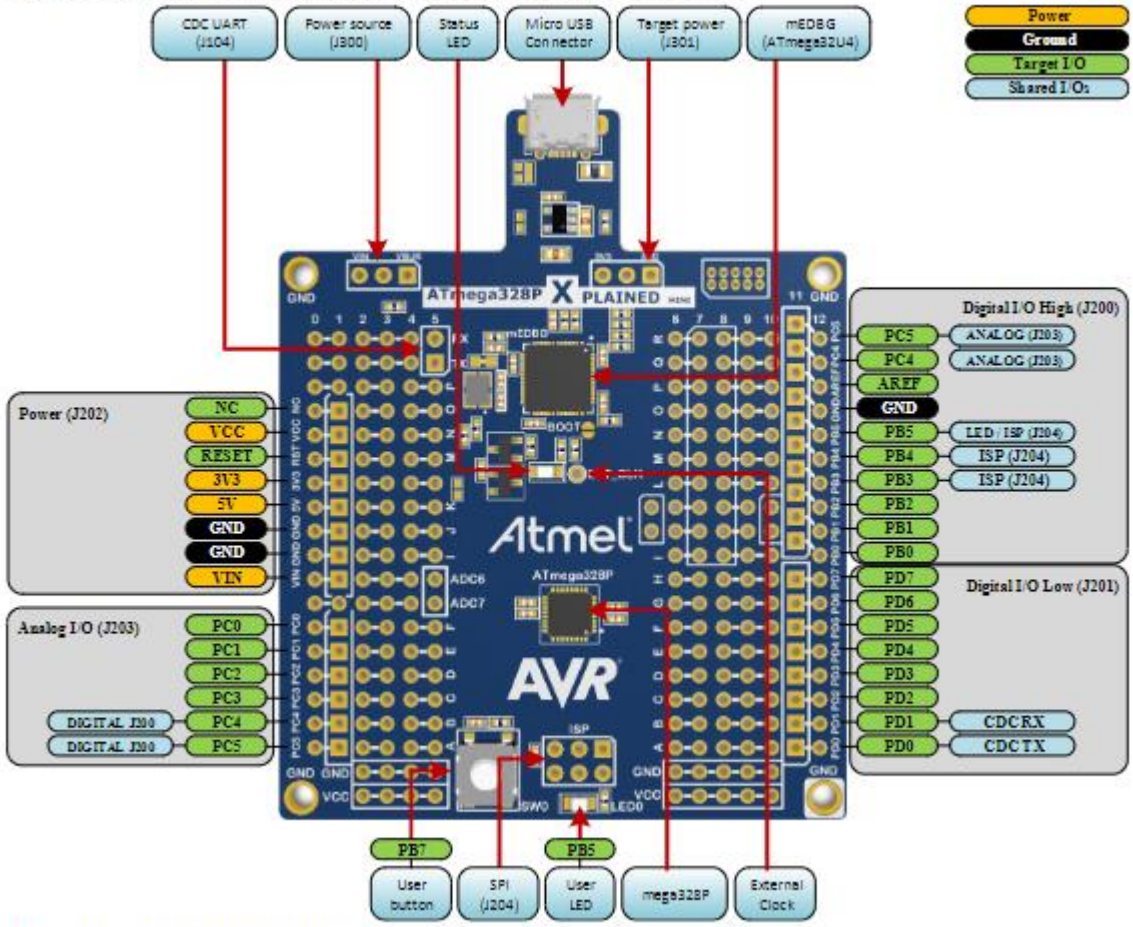
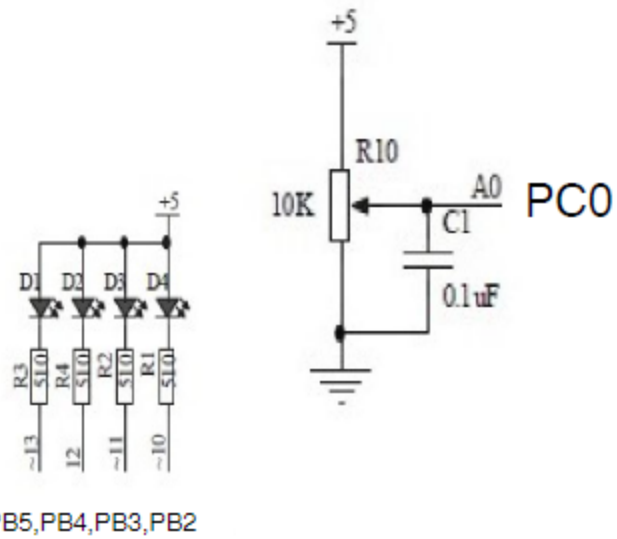


Table 1-1. Default Configurations

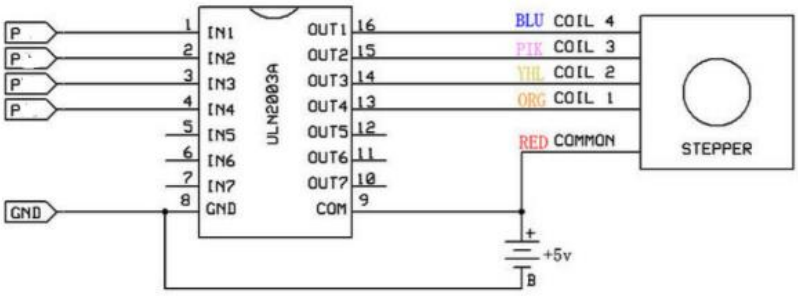


Schematic for shield LEDs (PB5:2) and Potentiometer (PC0)

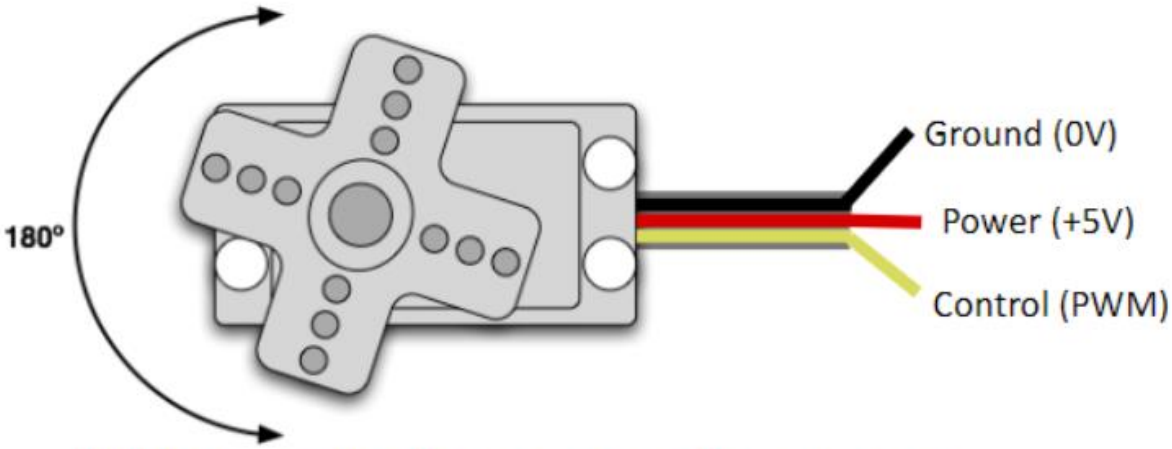
Model: 35BY48B06 - Unipolar Stepper Motor

Specifications:

- 5 VDC
- 7.5deg. step angle, 48 teeths
- Phase resistance: 10 Ohms
- Current: 500 mA



Stepper Motor Diagram



- PWM freq is 50 Hz (i.e. every 20 millisecs)
- Pulse width ranges from 1 to 2 millisecs
  - 1 millisec = full anti-clockwise position
  - 2 millisec = full clockwise position

Servo Diagram

2. INITIAL CODE OF TASK 1

```
#define F_CPU 16000000UL
#define BAUD_RATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUD_RATE * 16UL))) - 1)

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <util/delay.h>

void stepperDelay(int delayInMilliseconds);
void usart_init ();
void USART_send(unsigned char data);
void USART_putstring(char* StringPtr);

int main (void)
{
	DDRB = 0xFF;
	usart_init ();


	/** Setup and enable ADC **/
	ADMUX = (0<<REFS1)|    // Reference Selection Bits
	(1<<REFS0)|    // AVcc - external cap at AREF
	(0<<ADLAR)|    // ADC Left Adjust Result
	(0<<MUX2)|     // Analog Channel Selection Bits
	(0<<MUX1)|     // ADC0 (PC0) Potentionmeter
	(0<<MUX0);

	ADCSRA = (1<<ADEN)|    // ADC Enable
	(0<<ADSC)|     // ADC Start Conversion
	(0<<ADATE)|    // ADC Auto Trigger Enable
```

```
        (0<<ADIF)|     // ADC Interrupt Flag
        (0<<ADIE)|     // ADC Interrupt Enable
        (1<<ADPS2)|    // ADC Prescaler Select Bits
        (0<<ADPS1)|      // CLK/32
        (1<<ADPS0);

        int delay1 = 20; //20ms delay
        int delay2 = 40; //40ms delay
        int delay3 = 60;
        int delay4 = 80;
        int delay5 = 100;
        int delay6 = 120;
        int delay7 = 140;
        int delay8 = 160;
        int delay9 = 180;
        int delay10 = 200;

        while (1)
        {

                ADCSRA|=(1<<ADSC);   //start conversion
                while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish

                ADCSRA |= (1<<ADIF);

                int tempC = ADCL;
                tempC = tempC | (ADCH<<8);

                char output[20];
                snprintf(output, sizeof(output), "%d\r\n", tempC); //prints out
potentiometer value to serial terminal
                USART_putstring(output);

                //When the ADC value from the potentiometer is below 10, a duty cycle of 0%
will be output. Then above 10 and below 20, a duty cycle of 10%. This goes on until the
value is above
                //90 for which only a 95% duty cycle will be produced
                if((0 <= tempC) & (tempC < 10))
                {
                        PORTB=0x09;
                        stepperDelay(delay1);
                        PORTB=0x08;
                        stepperDelay(delay1);
                        PORTB=0x0C;
                        stepperDelay(delay1);
                        PORTB=0x04;
                        stepperDelay(delay1);
                        PORTB=0x06;
                        stepperDelay(delay1);
                        PORTB=0x02;
                        stepperDelay(delay1);
                        PORTB=0x03;
                        stepperDelay(delay1);
                        PORTB=0x01;
                        stepperDelay(delay1);
                }
                else if((10 <= tempC) & (tempC < 20))
                {
```

```
        PORTB=0x09;
        stepperDelay(delay2);
        PORTB=0x08;
        stepperDelay(delay2);
        PORTB=0x0C;
        stepperDelay(delay2);
        PORTB=0x04;
        stepperDelay(delay2);
        PORTB=0x06;
        stepperDelay(delay2);
        PORTB=0x02;
        stepperDelay(delay2);
        PORTB=0x03;
        stepperDelay(delay2);
        PORTB=0x01;
        stepperDelay(delay2);
}
else if((20 <= tempC) & (tempC < 30))
{
        PORTB=0x09;
        stepperDelay(delay3);
        PORTB=0x08;
        stepperDelay(delay3);
        PORTB=0x0C;
        stepperDelay(delay3);
        PORTB=0x04;
        stepperDelay(delay3);
        PORTB=0x06;
        stepperDelay(delay3);
        PORTB=0x02;
        stepperDelay(delay3);
        PORTB=0x03;
        stepperDelay(delay3);
        PORTB=0x01;
        stepperDelay(delay3);
}
else if((30 <= tempC) & (tempC < 40))
{
        PORTB=0x09;
        stepperDelay(delay4);
        PORTB=0x08;
        stepperDelay(delay4);
        PORTB=0x0C;
        stepperDelay(delay4);
        PORTB=0x04;
        stepperDelay(delay4);
        PORTB=0x06;
        stepperDelay(delay4);
        PORTB=0x02;
        stepperDelay(delay4);
        PORTB=0x03;
        stepperDelay(delay4);
        PORTB=0x01;
        stepperDelay(delay4);
}
else if((40 <= tempC) & (tempC < 50))
{
        PORTB=0x09;
```

```
        stepperDelay(delay5);
        PORTB=0x08;
        stepperDelay(delay5);
        PORTB=0x0C;
        stepperDelay(delay5);
        PORTB=0x04;
        stepperDelay(delay5);
        PORTB=0x06;
        stepperDelay(delay5);
        PORTB=0x02;
        stepperDelay(delay5);
        PORTB=0x03;
        stepperDelay(delay5);
        PORTB=0x01;
        stepperDelay(delay5);
}
else if((50 <= tempC) & (tempC < 60))
{
        PORTB=0x09;
        stepperDelay(delay6);
        PORTB=0x08;
        stepperDelay(delay6);
        PORTB=0x0C;
        stepperDelay(delay6);
        PORTB=0x04;
        stepperDelay(delay6);
        PORTB=0x06;
        stepperDelay(delay6);
        PORTB=0x02;
        stepperDelay(delay6);
        PORTB=0x03;
        stepperDelay(delay6);
        PORTB=0x01;
        stepperDelay(delay6);
}
else if((60 <= tempC) & (tempC < 70))
{
        PORTB=0x09;
        stepperDelay(delay7);
        PORTB=0x08;
        stepperDelay(delay7);
        PORTB=0x0C;
        stepperDelay(delay7);
        PORTB=0x04;
        stepperDelay(delay7);
        PORTB=0x06;
        stepperDelay(delay7);
        PORTB=0x02;
        stepperDelay(delay7);
        PORTB=0x03;
        stepperDelay(delay7);
        PORTB=0x01;
        stepperDelay(delay7);
}
else if((70 <= tempC) & (tempC < 80))
{
        PORTB=0x09;
        stepperDelay(delay8);
```

```
                PORTB=0x08;
                stepperDelay(delay8);
                PORTB=0x0C;
                stepperDelay(delay8);
                PORTB=0x04;
                stepperDelay(delay8);
                PORTB=0x06;
                stepperDelay(delay8);
                PORTB=0x02;
                stepperDelay(delay8);
                PORTB=0x03;
                stepperDelay(delay8);
                PORTB=0x01;
                stepperDelay(delay8);
        }
        else if((80 <= tempC) & (tempC < 90))
        {
                PORTB=0x09;
                stepperDelay(delay9);
                PORTB=0x08;
                stepperDelay(delay9);
                PORTB=0x0C;
                stepperDelay(delay9);
                PORTB=0x04;
                stepperDelay(delay9);
                PORTB=0x06;
                stepperDelay(delay9);
                PORTB=0x02;
                stepperDelay(delay9);
                PORTB=0x03;
                stepperDelay(delay9);
                PORTB=0x01;
                stepperDelay(delay9);
        }
        else
        {
                PORTB=0x09;
                stepperDelay(delay10);
                PORTB=0x08;
                stepperDelay(delay10);
                PORTB=0x0C;
                stepperDelay(delay10);
                PORTB=0x04;
                stepperDelay(delay10);
                PORTB=0x06;
                stepperDelay(delay10);
                PORTB=0x02;
                stepperDelay(delay10);
                PORTB=0x03;
                stepperDelay(delay10);
                PORTB=0x01;
                stepperDelay(delay10);
        }

    }
    return 0;
}
```

```
void usart_init (void)
{
	UBRR0H = (uint8_t)(BAUD_PRESCALLER >> 8);
	UBRR0L = (uint8_t)(BAUD_PRESCALLER);
	UCSR0B = (1 << RXEN0) | (1 << TXEN0);
	UCSR0C = (3 << UCSZ00);
}


void USART_send( unsigned char data) {

	while (!(UCSR0A & (1 << UDRE0))); //wait until UDR0 is empty
	UDR0 = data;                                  //transmit ch

}

void USART_putstring(char* StringPtr) {

	while (*StringPtr != 0x00) {
		USART_send(*StringPtr);
		StringPtr++;
	}

}

void stepperDelay(int delayInMilliseconds)
{
	TCNT1 = 0;
	OCR1A = (((F_CPU/64)/1000)*delayInMilliseconds)-1;
	TCCR1B = (1 << WGM12) | (1 << CS11) | (1 << CS10); //CTC mode, CLK/64 for maximum
delay of ~260ms
	while(TCNT1 < OCR1A); //Wait until TCNT is equal to OCR1A
	TCCR1B = 0; //Stop timer
}
```

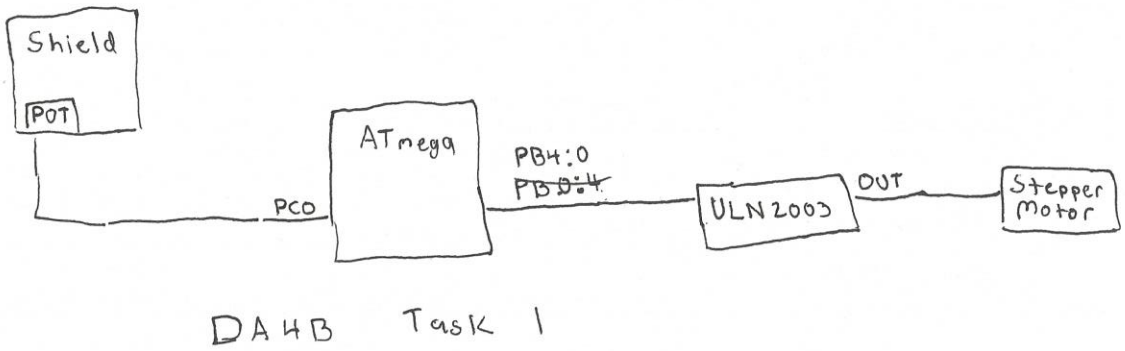
3. INITIAL CODE OF TASK 2

```
#define F_CPU 16000000UL
#define BAUD_RATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUD_RATE * 16UL))) - 1)
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

void adc_init();
int adc_read();
void usart_init ();
void USART_send(unsigned char data);
void USART_putstring(char* StringPtr);

int main(void)
{
	usart_init();
	adc_init();

	//Using information from slide 15-16 from Lecture 11 presentation
```

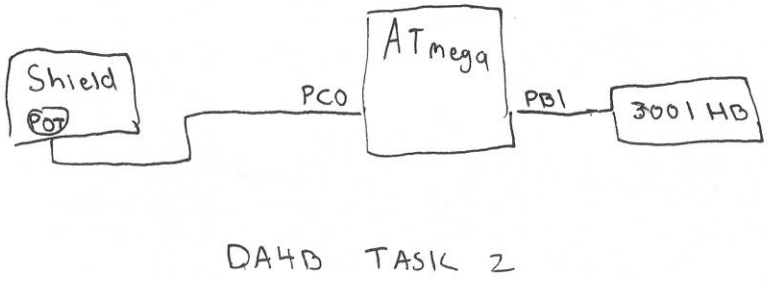
```
	TCNT1 = 0;		// Set timer1 count zero
	ICR1 = 39999;		// Set TOP count for timer1 in ICR1 register

	/* Set Fast PWM, TOP in ICR1, Clear OC1A on compare match, clk/8
	F = 2MHz, T = 0.5us
	For a period of 20ms, need 20ms/0.5us instructions = 40,000
	40,000 instructions per 20ms -> 2 instructions per 1us
	 */

	TCCR1A = (1<<COM1A1) | (1<<WGM11);
	TCCR1B = (1<<WGM13) | (1<<WGM12) | (1<<CS11);

	DDRB |= (1 << 1); //OC1A for ATmega328PB = PB1
	while(1)
	{
		//OCR1A will have a range of 1999 to 4999. The 3001HB Servo motor has a
maximum travel length listed of approx. 165° from 800us -> 2200us
		//Was able to push to 2500us before the motor would stop turning and buzz
		OCR1A = 1999 + (adc_read()*2.93) ; //Max adc value is 1023 and 3000/1023 ~=
2.93

	}
}

void adc_init() {
		/** Setup and enable ADC **/
		ADMUX = (0<<REFS1)|	// Reference Selection Bits
		(1<<REFS0)|	// AVcc - external cap at AREF
		(0<<ADLAR)|	// ADC Left Adjust Result
		(0<<MUX2)|	// Analog Channel Selection Bits
		(0<<MUX1)|	// ADC0 (PC0) Potentionmeter
		(0<<MUX0);

		ADCSRA = (1<<ADEN)|	// ADC Enable
		(0<<ADSC)|	// ADC Start Conversion
		(0<<ADATE)|	// ADC Auto Trigger Enable
		(0<<ADIF)|	// ADC Interrupt Flag
		(0<<ADIE)|	// ADC Interrupt Enable
		(1<<ADPS2)|	// ADC Prescaler Select Bits
		(0<<ADPS1)|	  // CLK/32
		(1<<ADPS0);
}

int adc_read()
{
	ADCSRA|=(1<<ADSC);	//start conversion
	while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish

	ADCSRA |= (1<<ADIF);

	int tempADC = ADCL;
	tempADC = tempADC | (ADCH<<8);

	char output[20];
	snprintf(output, sizeof(output), "%d\r\n", tempADC); //prints out potentiometer
value to serial terminal
	USART_putstring(output);
```

```
	return tempADC;
}

void usart_init (void)
{
	UBRR0H = (uint8_t)(BAUD_PRESCALLER >> 8);
	UBRR0L = (uint8_t)(BAUD_PRESCALLER);
	UCSR0B = (1 << RXEN0) | (1 << TXEN0);
	UCSR0C = (3 << UCSZ00);
}


void USART_send( unsigned char data) {

	while (!(UCSR0A & (1 << UDRE0))); //wait until UDR0 is empty
	UDR0 = data;                                  //transmit ch

}

void USART_putstring(char* StringPtr) {

	while (*StringPtr != 0x00) {
		USART_send(*StringPtr);
		StringPtr++;
	}

}
```

4. SCHEMATICS



Task 1 Schematic

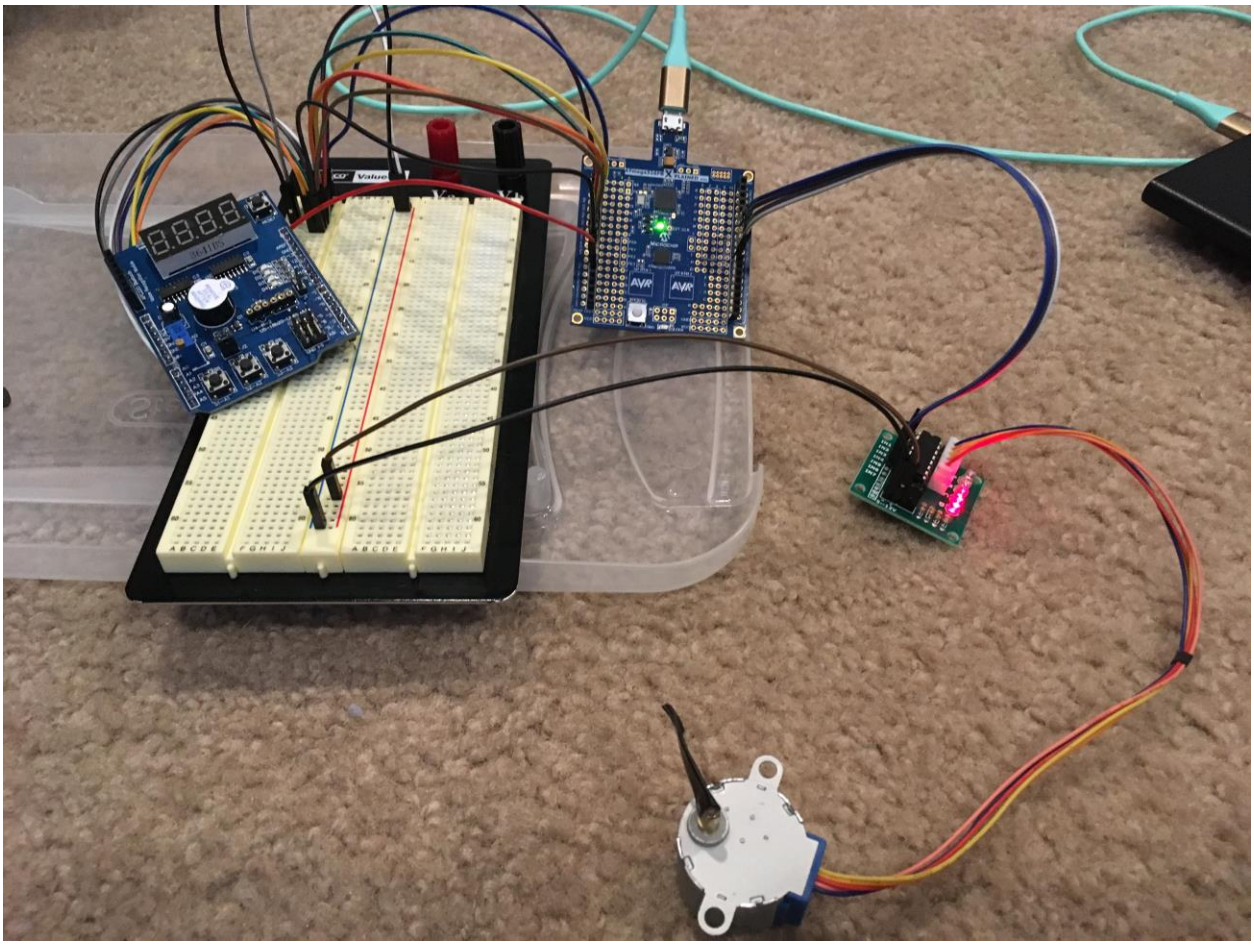


Task 2 Schematic

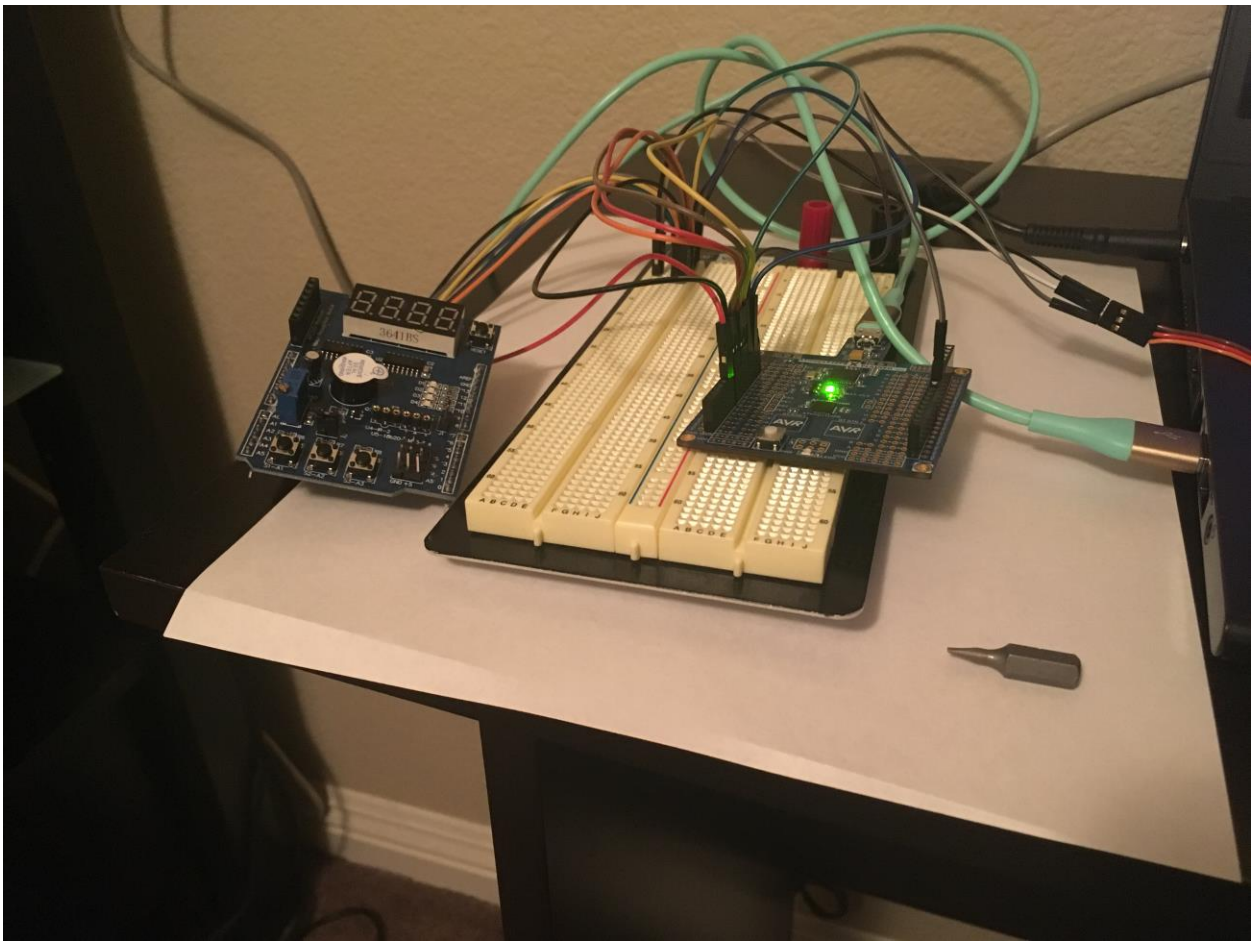
5. SCREENSHOTS OF EACH TASK OUTPUT

Refer to videos to see motor function controlled by potentiometer

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



Board setup for Task 1



Board setup for Task 2

7. VIDEO LINKS OF EACH DEMO

Task1: https://youtu.be/32tz3yPUxmA
Task2: https://youtu.be/ZcZeva9N44o

8. GITHUB LINK OF THIS DA

https://github.com/portig1/submissions_E/tree/master/DA/LAB4B

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Geovanni Portillo