Elizabeth Heider

CPE301 – SPRING 2018

Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

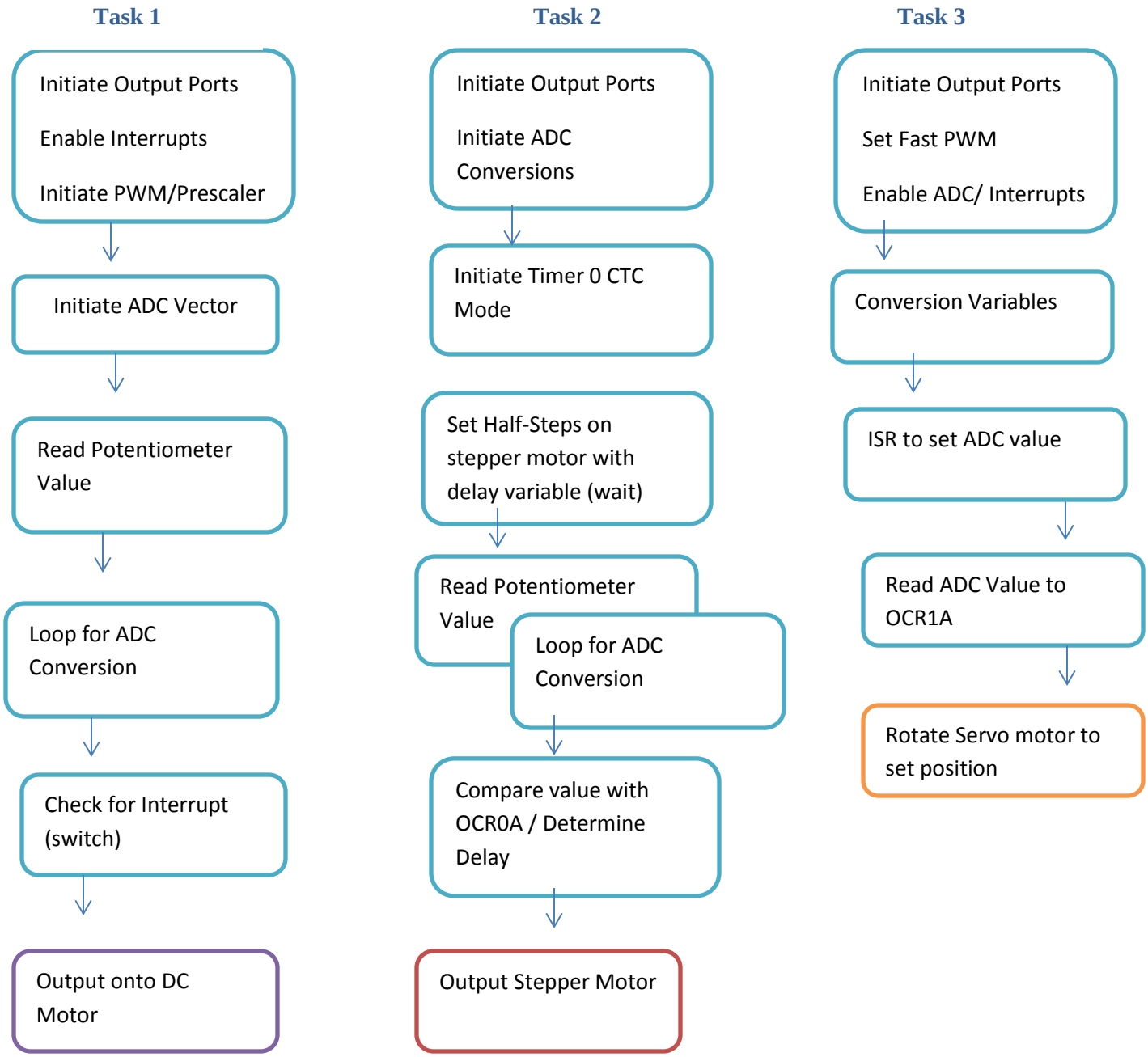
The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND FLOWCHARTN		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
4.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
5.	SCHEMATICS		
6.	VIDEO LINKS OF EACH DEMO		
7.	GITHUB LINK OF THE DA		

1. COMPONENTS LIST / FLOWCHART OF CODE

List of Components Used:

- Atmega328 Chip
- DC Motor
- Stepper Motor
- Servo Motor
- PushButton Switch
- Potentiometer



2. INITIAL/DEVELOPED CODE OF TASK 1

```
#define F_CPU 8000000UL // clock is 8MHz
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

void init_adc(void);
int control; // controlling the motor with potentiometer
int toggle = 0; // to toggle the motor with the switch

int main()
{
	DDRD = 0xFB; // set Port D as outputs, leave INT1(PD3) as interrupt
	PORTD = 0x00; // pull-up pins

	EIMSK = 0x02; // enable INT1
	EIFR = 0x02; // enable interrupt flag
	EICRA = 0X0C; // set interrupt on rising edge

	sei(); // enable interrupts
	init_adc(); // initiate adc stuff
	TCCR0A=0x83; // set fast PWM // clear OCR0A on MATCH
	TCCR0B=0x03; // set prescaler to 1024

	while (1)
	{
		while((ADCSRA&(1<<ADIF))==0); // wait for conversion
		control = ADC*80/100; // ADC Conversion
		OCR0A = control; // Output to converted value to 0CR0A

		if(toggle == 0) // when switch is NOT pressed (I THINK)
		{
			PORTD = 0x00; //honestly not sure what the hell I did but it worked?
		}
	}
}
void init_adc(void) // Initiate ADC function
	{
	ADMUX = (1<<REFS0); // Reference voltage at Aref
	ADCSRA = (1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
	// from ADCRSA we Enable ADC, Start Conversion, Set prescalar as 128
	}

ISR(INT1_vect)
{
	toggle ^= 1; // toggle switch on INT1 Interrupt
   }
```

3. MODIFIED CODE OF TASK 2/A

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>


void init_adc(void); // ADC functions
void timer0_init(); // ADC function for Timer 0 CTC

int control = 0; // converted ADC value to control motor speed

int main()
{

	DDRD = 0xFF; // Set Port D for outputs
	int wait = 0; // Wait is the delay

	init_adc(); // Initialize ADC conversions
	timer0_init(); // Initialize timer0 CTC function

	while(1)
	{
		while((ADCSRA&(1<<ADIF))==0);
		control = ADC*80/100; // ADC Conversion

		// Stepper Motor function in half stepper mode (7 commands vs 4)

		if (control < 1) // when control < 1 potentiometer is lowest voltage
		{
			wait = 0; // wait is the delay that controls the

			PORTD = 0x0C;
			_delay_ms (wait);
			PORTD = 0x04;
			_delay_ms (wait);
			PORTD = 0x06;
			_delay_ms (wait);
			PORTD = 0x02;
			_delay_ms (wait);
			PORTD = 0x01;
			_delay_ms (wait);
			PORTD = 0x09;
			_delay_ms (wait);
			PORTD = 0x08;
			_delay_ms (wait);
		}
		else if (control < 3)
		{
			wait = 100;

			PORTD = 0x0C;
			_delay_ms (wait);
			PORTD = 0x04;
			_delay_ms (wait);
			PORTD = 0x06;
			_delay_ms (wait);
			PORTD = 0x02;
```

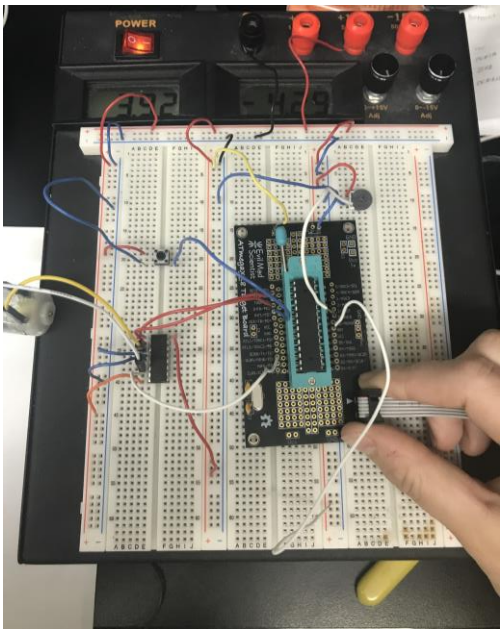
```
                _delay_ms (wait);
                PORTD = 0x01;
                _delay_ms (wait);
                PORTD = 0x09;
                _delay_ms (wait);
                PORTD = 0x08;
                _delay_ms (wait);
        }
        else if (control < 4)
        {
                wait = 50;

                PORTD = 0x0C;
                _delay_ms (wait);
                PORTD = 0x04;
                _delay_ms (wait);
                PORTD = 0x06;
                _delay_ms (wait);
                PORTD = 0x02;
                _delay_ms (wait);
                PORTD = 0x01;
                _delay_ms (wait);
                PORTD = 0x09;
                _delay_ms (wait);
                PORTD = 0x08;
                _delay_ms (wait);
        }

        else if (control > 4) // when control > 4 potentiometer at max value
        {
                wait = 10;

                PORTD = 0x0C;
                _delay_ms (wait);
                PORTD = 0x04;
                _delay_ms (wait);
                PORTD = 0x06;
                _delay_ms (wait);
                PORTD = 0x02;
                _delay_ms (wait);
                PORTD = 0x01;
                _delay_ms (wait);
                PORTD = 0x09;
                _delay_ms (wait);
                PORTD = 0x08;
                _delay_ms (wait);
        }

    }
}

void init_adc(void) // Initiate ADC function
{
    ADMUX = (1<<REFS0); // Reference voltage at Aref
    ADCSRA = (1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
    // from ADCRSA we Enable ADC, Start Conversion, Set prescalar as 128
}
```

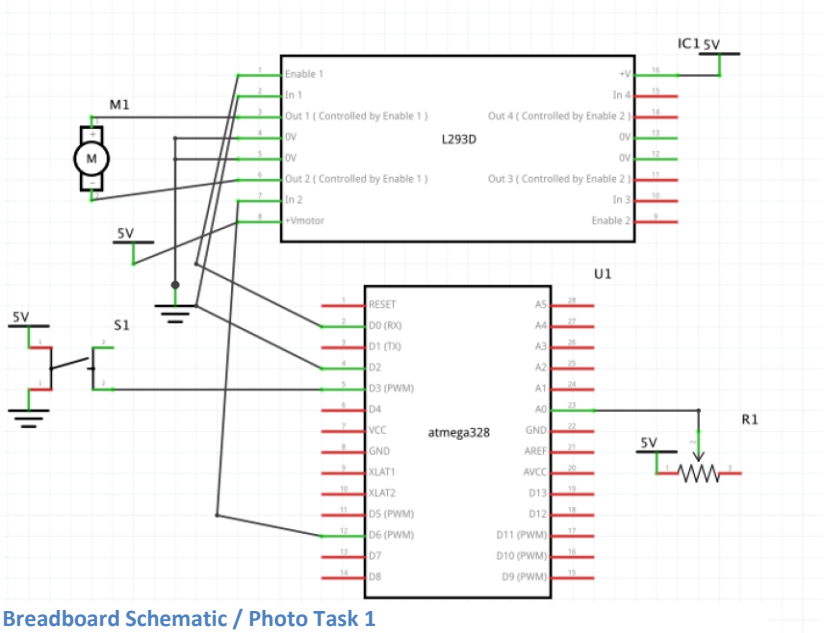
```
//  Using a timer in CTC mode to control the delay
void timer0_init()
{
	TCCR0B |= (1 << WGM12)|(1 << CS11)|(1 << CS10); // Prescalar 64 set CTC mode
	TCCR0A |= (1 << COM1A0); // Set timer in OCOA Pin Toggle Mode
	TCNT0 = 0; // Initialize Counter
	OCR0A = control; // OCROA reading ADC converted value
}
```

4. INITIAL/DEVELOPED CODE OF TASK 3

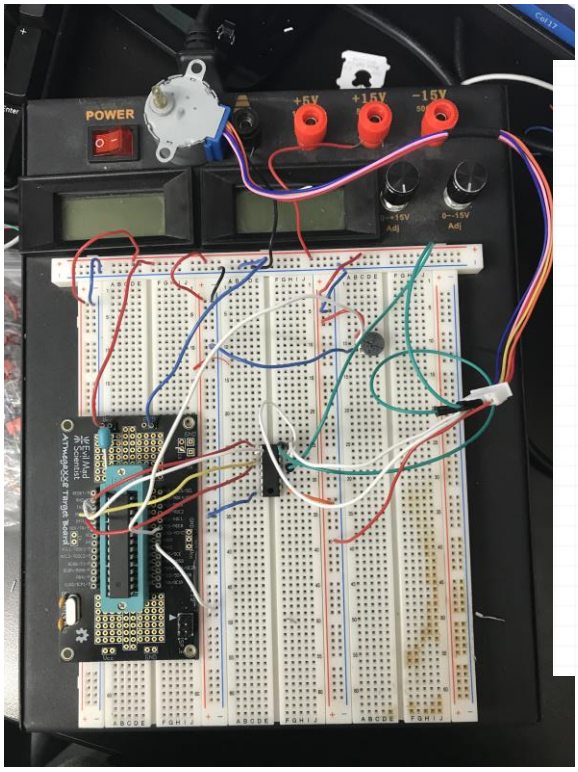
```
#include <avr/io.h>
#include <avr/interrupt.h>

static volatile uint16_t adc_val;

int main(void)
{
	// Set PORTB As output Port
	// Set Fast PWM, ICR1 at TOP, Update 0CR1A at Bottom
	DDRB |= 0xFF;
	TCCR1A |= (1<<COM1A0) | (1<<COM1A1) | (1<<WGM11);
	TCCR1B |= (1<<WGM12) | (1<<WGM13) | (1<<CS10);

	ICR1 = 19999; //Sets TOP to 19999.

	// Enable ADC, Enable Interrupt, Set Prescalar 16
	// Vref is Internal 1.1V Vref
	ADCSRA |= (1<<ADEN)| (1<<ADIE)| (1<<ADPS2);
	ADMUX |= (1<<REFS1) | (1<<REFS0);

	sei();

	// Start Conversions
	ADCSRA |= (1<<ADSC);

	// Create Variables For Conversion
	float upper = 2400;
	float lower = 800;
	float diff = (upper-lower)/1023;

	// Slope intercept form y = mx + b
	// YAY math!!
	while(1)
	{
		OCR1A = ICR1 - ((diff*adc_val) + lower);
	}
}

//Load value of ADC into adc_low.
//Set value to be used in setting OCR1A in while loop (main).
//Enable ADC Conversions
ISR(ADC_vect)
{
	uint8_t adc_low = ADCL;
	adc_val = (ADCH<<8) | adc_low;
	ADCSRA |= (1<<ADSC);
	}
```

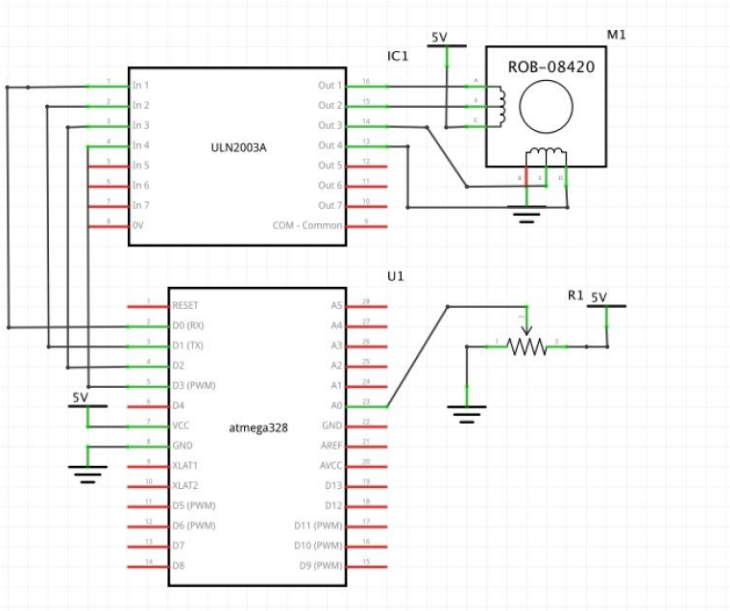
5. SCHEMATICS



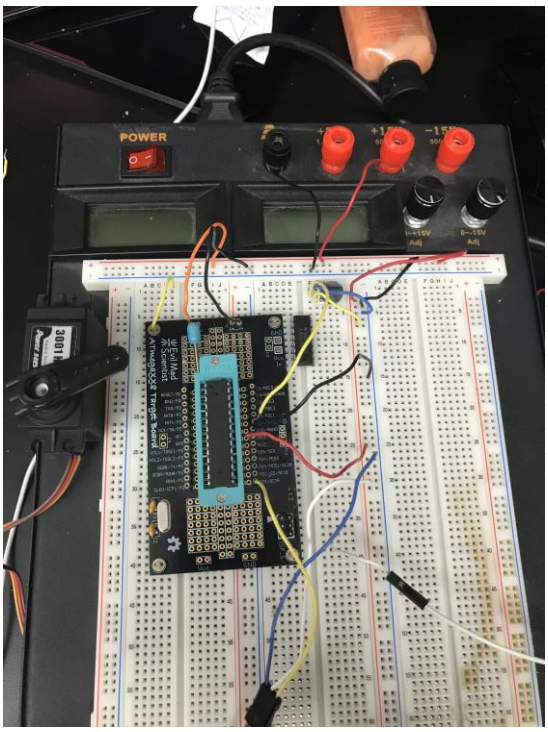


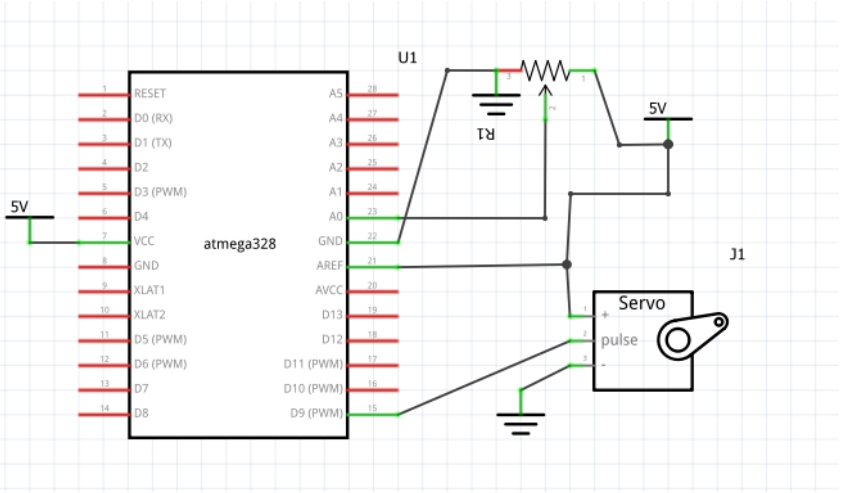
Breadboard Schematic / Photo Task 1





Breadboard Schematic / Photo Task 2





Breadboard Schematic / Photo Task 3

6. VIDEO LINKS OF EACH DEMO

TASK 1: https://youtu.be/doLIRM-jaXY
TASK 2: https://youtu.be/qRcXHsuEPYs
TASK 3: https://youtu.be/Qu70HH5voqM

7. GITHUB LINK OF THIS DA

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

"This assignment submission is my own, original work".
Elizabeth Heider