Jeffrey Razon

CPE301 – SPRING 2018

Design Assignment 04

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

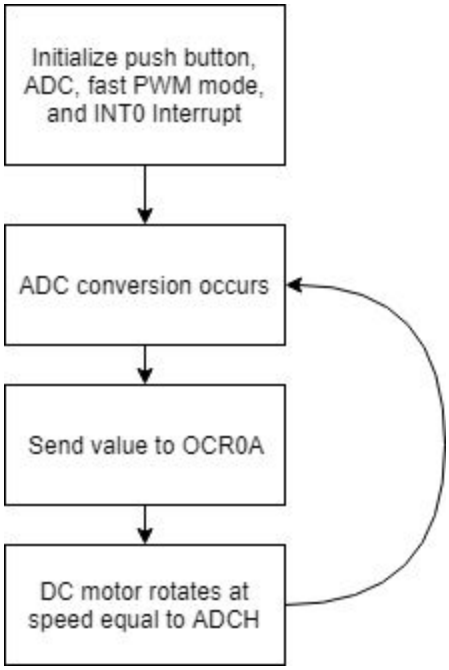
The student understands that all required components should be submitted in complete for grading of this assignment.

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

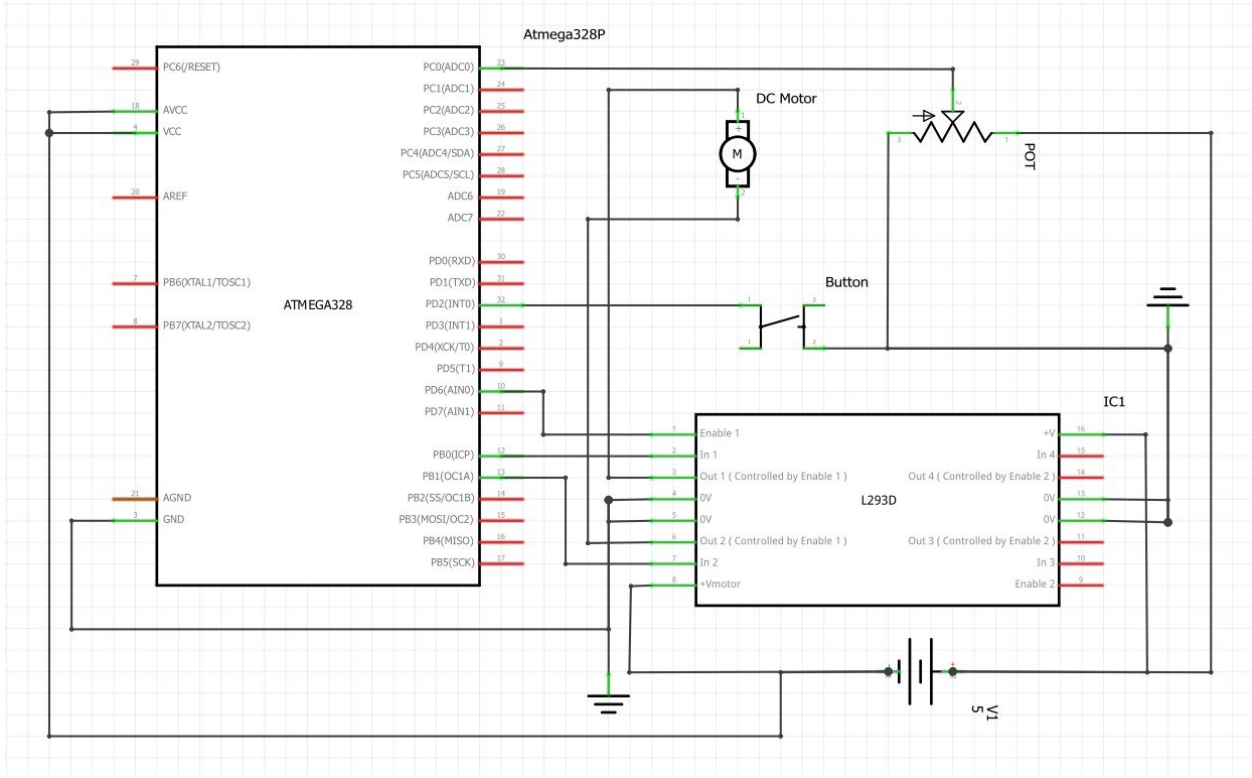
Task 1/A: Write an AVR C program to control the speed of the DC Motor using a potentiometer connected to any of the analog-in port. Use an interrupt on a button to stop and start the motor at each click. The minimum speed of the motor should be 0 when pot is minimum and maximum should be 95% of PWM value.

```
#define F_CPU 8000000L

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

#define BUTTONPORT PORTD
#define BUTTONDDR DDRD
#define BUTTON_PIN PD2

volatile unsigned int n; // true/false value to determine if motor is on/off

// this interrupt turns the motor on/off
ISR(INT0_vect)
{
    EIFR |= (1 <<INTF0); // clear int flag
    PORTB ^= (1 << PORTB0);
}

int main(void)
{
    // set ports
    DDRB = 0xFF;
    BUTTONPORT = (1 << BUTTON_PIN); // pull-up
    BUTTONDDR = (1 << BUTTON_PIN); // set PD2 as input (INT0 interrupt)
    DDRD |= (1 << PORTD6);  // PD.6 (OC0A) is an output

    // ADMUX and ADC config
    ADMUX = 0; // use ADC0
    ADMUX |= (1 << REFS0); // use AVcc as the reference
    ADMUX |= (1 << ADLAR); // Right adjust for 8 bit resolution
    ADCSRA = 0x87; // enables ADC, sets prescaler for ADC conversion
    ADCSRB = 0x00; // free running mode

    OCR0A = 0;
    TCCR0A |= (1 << COM0A1); // non-inverting mode
    TCCR0A |= (1 << WGM01) | (1 <<WGM00);   // fast PWM mode
    TCCR0B |= (1 << CS01); // prescaler 8

    // INT0 interrupt settings
    EIMSK = (1 << INT0);                    // enables INT0 interrupt
    EICRA = (1 << ISC01) | (1 << ISC00);    // triggers INT0 on rising edge

    n = 0; //initialize motor as turned off
    PORTB = 0x01;   //hbridge.forward

    sei(); // enable interrupts

    while (1)
    {
        ADCSRA |= (1 << ADSC); // start conversion
        while ( (ADCSRA&(1<<ADIF)) == 0 ) ; // wait for conversion to finish
        OCR0A = ADCH;
    }
}
```

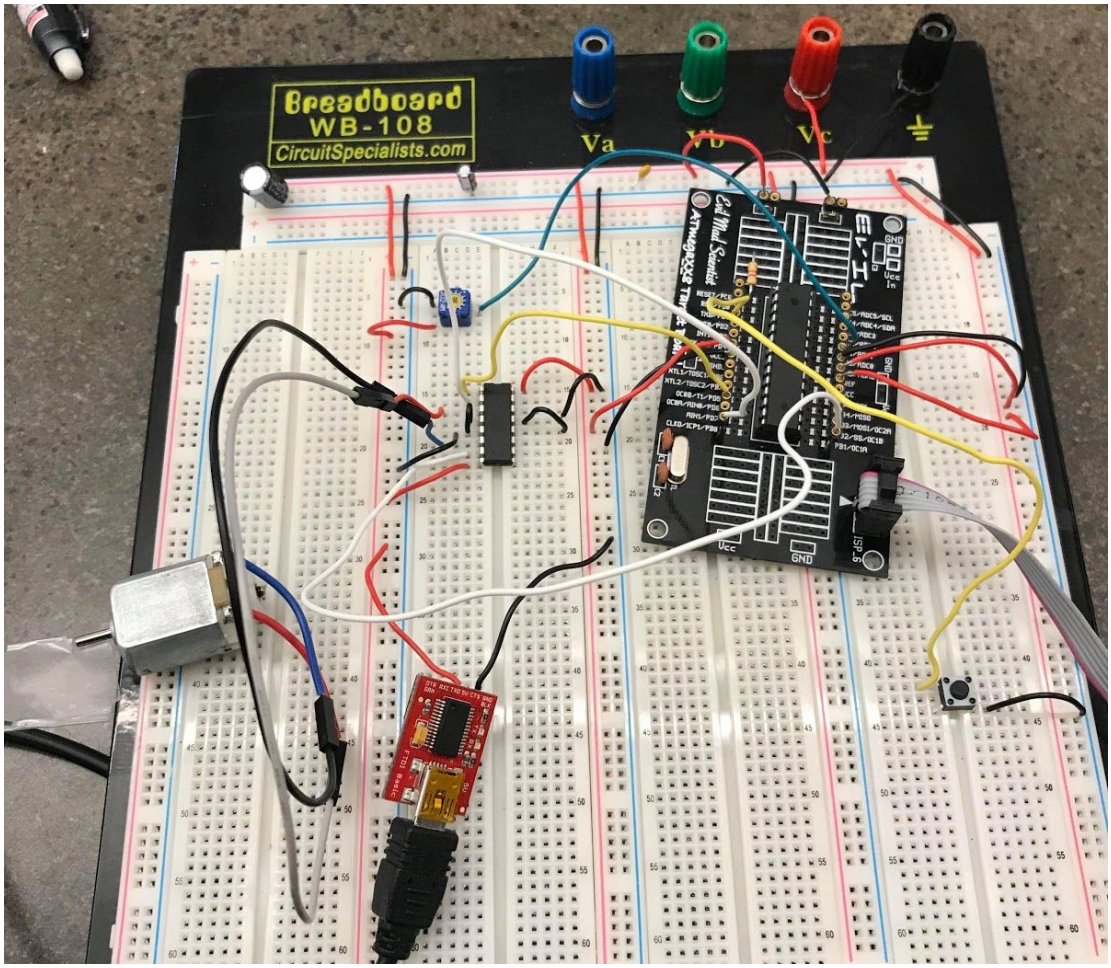
Flowchart:



Schematics:



Breadboard:

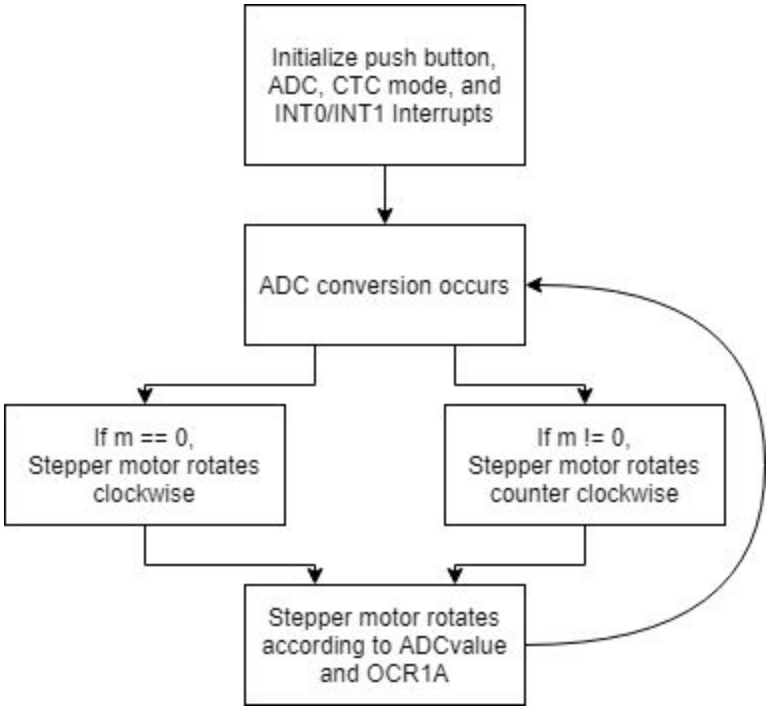


Task 2/B: Write an AVR C program to control the speed of the Stepper Motor using a potentiometer connected to any of the analog-in port. Use a timer in CTC mode to control the delay.

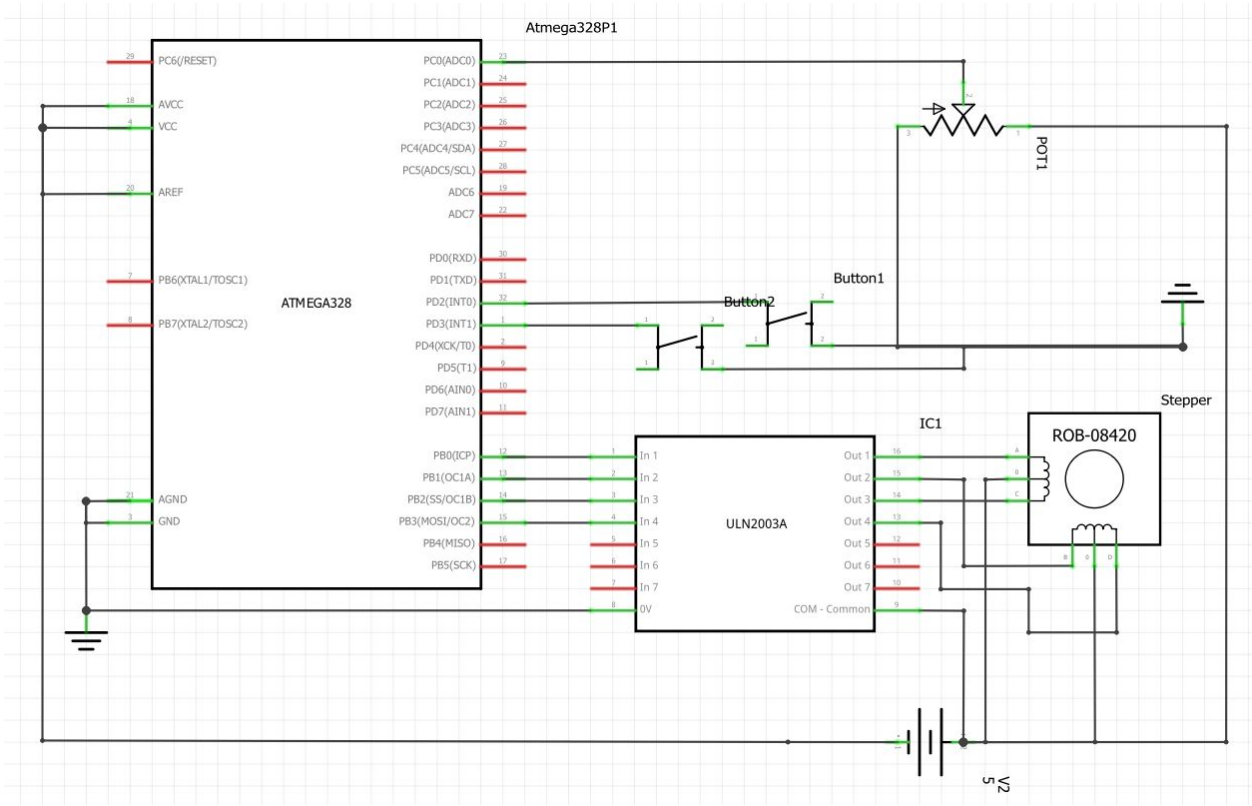
```
#define F_CPU 8000000UL

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

#define BUTTONPORT PORTD
#define BUTTONDDR DDRD
#define BUTTON_PIN PD2
#define BUTTON_PIN_B PD3

void Clockwise(); //clockwise function
void CClockwise(); //counter-clockwise function
void delaytime(); // ctc delay function

volatile unsigned int n; // true/false value to determine if motor is on/off
volatile unsigned int m; // true/false value for motor direction
volatile uint8_t ADCvalue; // Global variable, set to volatile if used with ISR

// this interrupt turns the motor on/off
ISR(INT0_vect)
{
    EIFR |= (1 <<INTF0); // clear int flag
    if(n == 0)
        n = 1; // turn on motor
    else
        n = 0; // turn off motor
}

// this interrupt changes motor's direction
ISR(INT1_vect)
{
    EIFR |= (1 <<INTF1); // clear int flag
    if(m == 0)
        m = 1; // change to counter-clockwise
    else
        m = 0; // change to clockwise
}

int main(void)
{
    // set ports
    DDRB = 0x0F;
    BUTTONPORT = (1 << BUTTON_PIN) | (1 << BUTTON_PIN_B);   // pull-up
    // set PD2 and PD3 as input (INT0 and INT1 interrupt)
    BUTTONDDR = (1 << BUTTON_PIN) | (1 << BUTTON_PIN_B);

    // ADMUX and ADC config
    ADMUX = 0; // use ADC0
    ADMUX |= (1 << REFS0); // use AVcc as the reference
    ADMUX |= (1 << ADLAR); // Right adjust for 8 bit resolution
    ADCSRA = 0x87; // enables ADC, sets prescaler for ADC conversion
    ADCSRB = 0x00; // free running mode

    // set timer
    TCCR1B |= (1 << WGM12)|(1 << CS11)|(1 << CS10); // CTC mode, prescaler 64

    // INT0 and INT1 interrupt settings
    EIMSK = (1 << INT0) | (1 << INT1);      // enables INT0 and INT1 interrupts
    EICRA = (1 << ISC01) | (1 << ISC00);    // triggers INT0 on rising edge
    EICRA = (1 << ISC11) | (1 << ISC10);    // triggers INT1 on rising edge
    n = 0; //initialize motor as turned off

    sei(); // enable interrupts
```

```
    while (1)
    {
        ADCSRA |= (1 << ADSC); // start ADC conversion
        while ( (ADCSRA&(1<<ADIF)) == 0 ); // wait for conversion to finish
        ADCvalue = ADCH; // classify ADCvalue has high 8 bits
        if(n != 0 && m == 0) // if turned on and m = 0,
        {
            Clockwise(); // call to move motor clockwise.
            _delay_ms(1);
        }
        if(n != 0 && m != 0) // Otherwise,
        {
            CClockwise();// call to move motor counter-clockwise
            _delay_ms(1);
        }
    }
}

void delaytime()
{
    TCNT1 = 0; // initialize counter
    OCR1A = ADCvalue * 100; // initialize top
    while(!(TIFR1 & (1 << OCF1A))); // waits for overflow
    TIFR1 |= (1 << OCF1A); // clears overflow flag
}

void Clockwise()
{
    PORTB = 0x06;
    delaytime();
    PORTB = 0x0C;
    delaytime();
    PORTB = 0x09;
    delaytime();
    PORTB = 0x03;
    delaytime();
}

void CClockwise()
{
    PORTB = 0x06;
    delaytime();
    PORTB = 0x03;
    delaytime();
    PORTB = 0x09;
    delaytime();
    PORTB = 0x0C;
    delaytime();
}
```

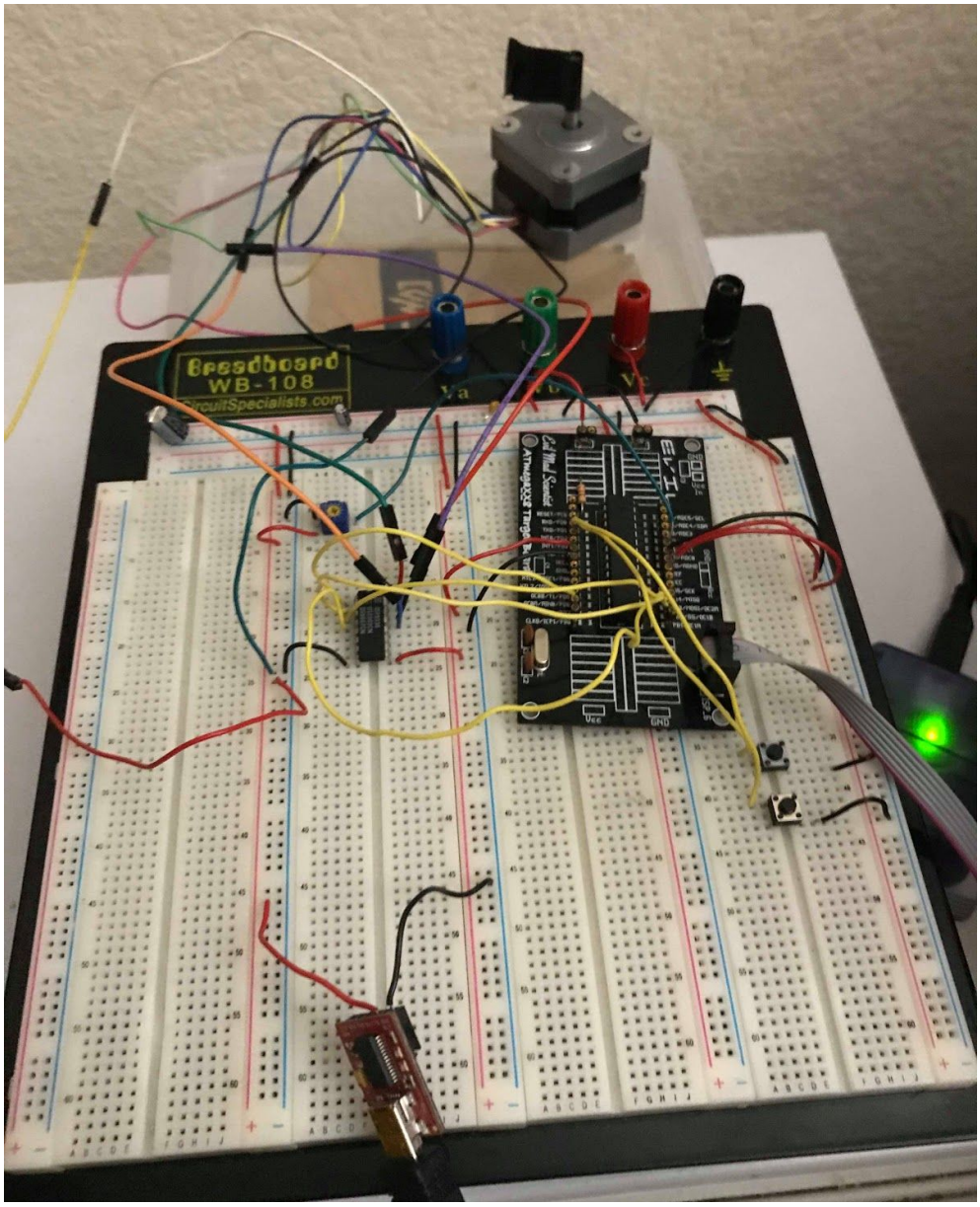
Flowchart:



Schematics:



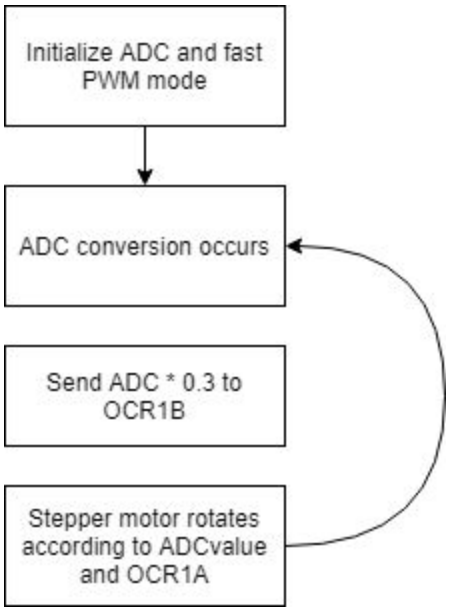
Breadboard:



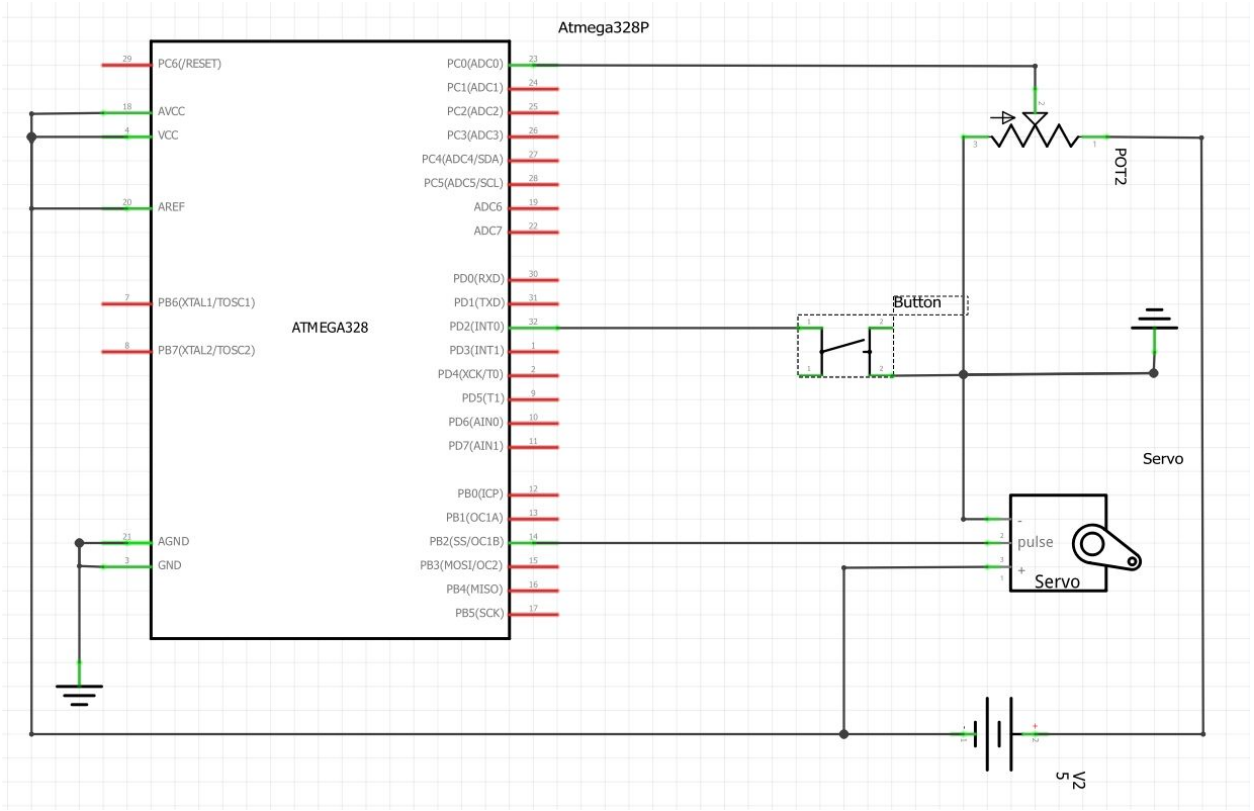
Task 3/C: Write an AVR C program to control the position of the Servo Motor using a potentiometer connected to any of the analog-in port. When pot value is 0 the servo is at position 0 deg. and when pot value is max (approx. 5V) the servo is at position 180 deg.

```
#define F_CPU 8000000L

#include <avr/io.h>
#include <util/delay.h> // delays

int main(void) {

    // set ports
    DDRB = 0xFF;

    // ADMUX and ADC config
    ADMUX = 0; // use ADC0
    ADCSRA = 0x87; // enable ADC, system clock used for A/D conversion
    ADCSRB = 0x00; // free running mode

    // timer config, ICR1 = TOP
    TCCR1A |= (1 << COM1A1) | (1 << COM1B1) |  (1 << WGM11); // Non-inverted PWM
    TCCR1B |= (1 << WGM13) | (1 << WGM12);  // Fast PWM mode
    TCCR1B |= (1 << CS11) | (1 << CS10); // Prescaler 64
    ICR1 = 2500; // top

    while (1)
    {
            ADCSRA |= (1 << ADSC); // start ADC conversion
            while( (ADCSRA&(1<<ADIF)) == 0 ); // wait for conversion to finish
            OCR1B = ADC * 0.3; // approximate value for pot. to rotate servo 180 degrees
    }
}
```

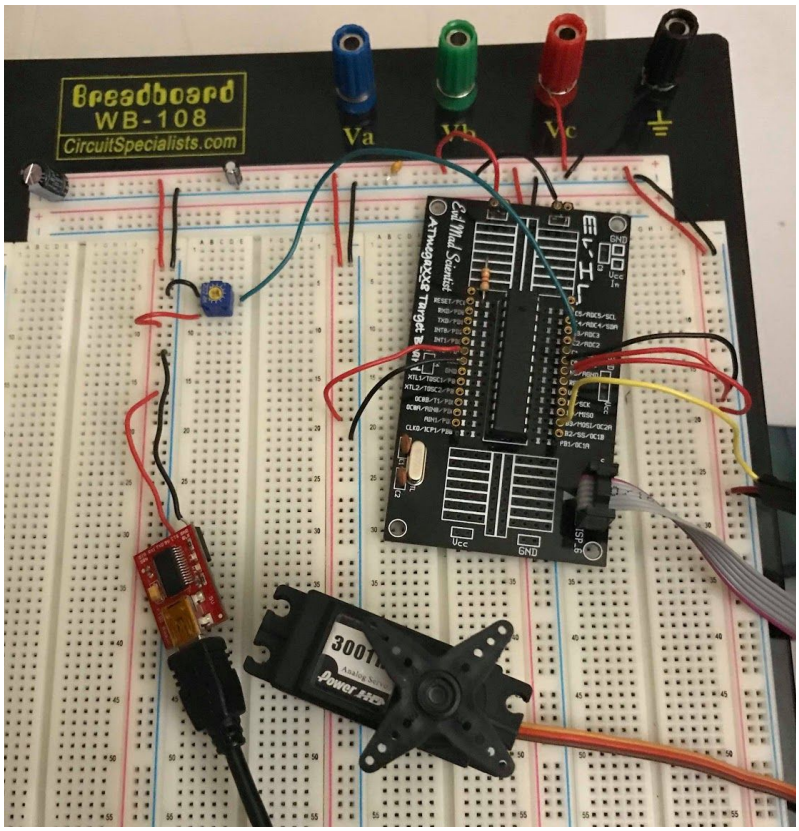
Flowchart:



Schematics:



Breadboard:



GITHUB LINK: https://github.com/JeffinVegas/EmbSys.git
YOUTUBE LINK: In the videos_DA04.txt file

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

"*This assignment submission is my own, original work*".
Jeffrey Razon