Nathan Hanuscin

CPE301 – SPRING 2018

Design Assignment 4

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

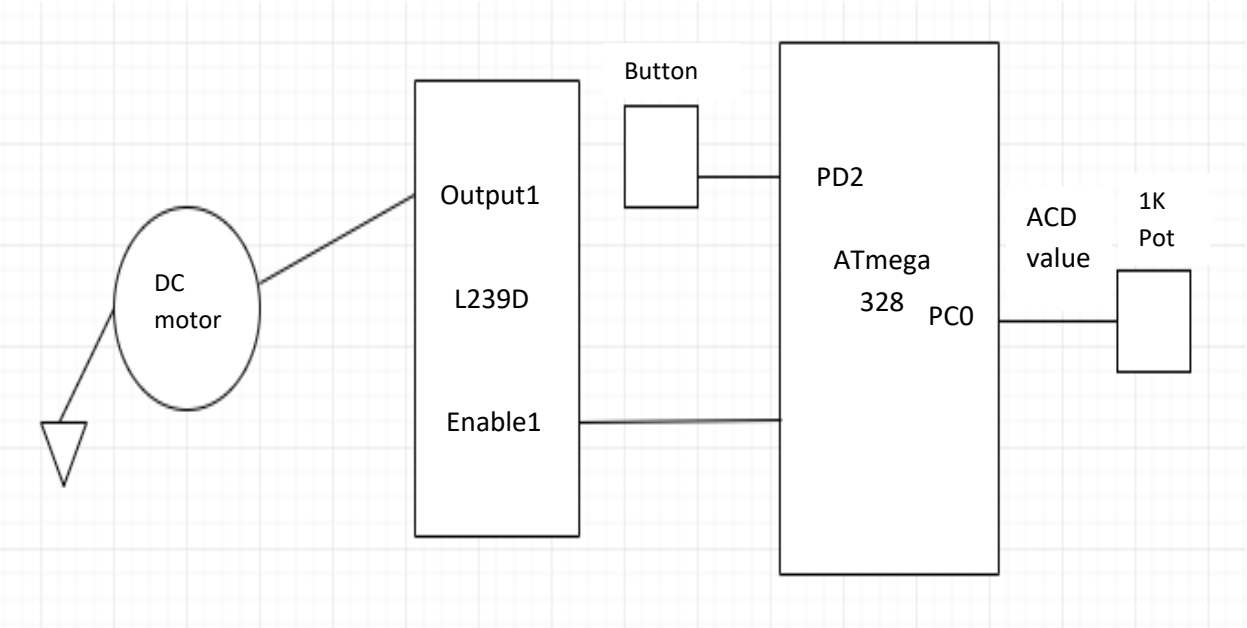
The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND 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		

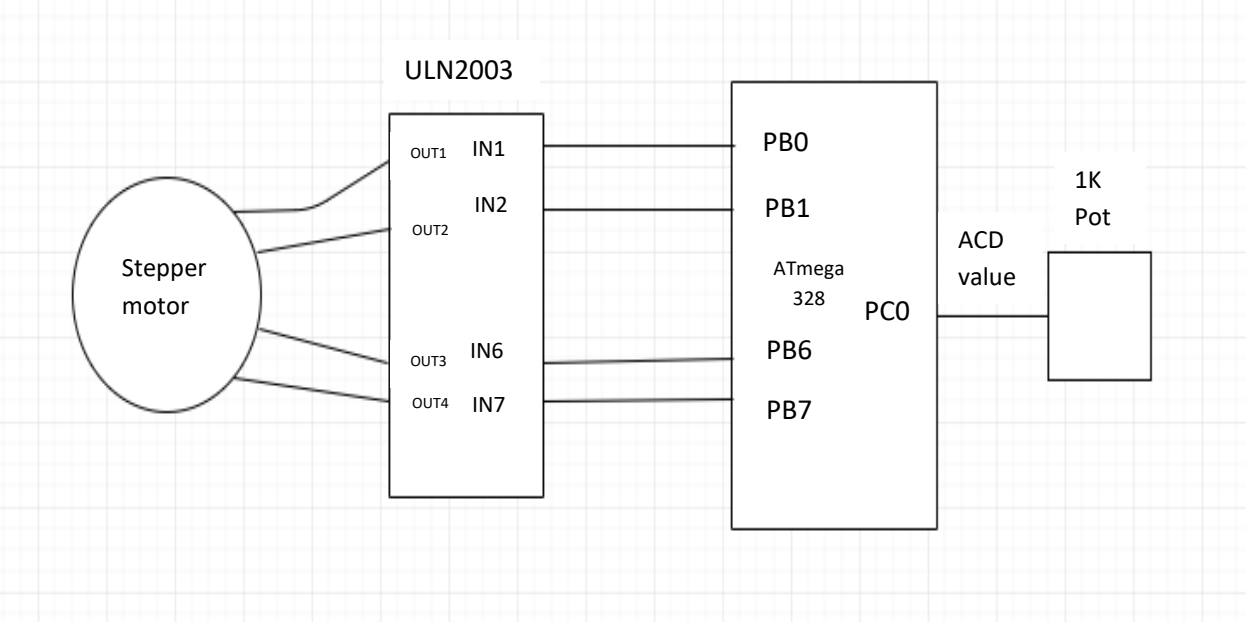
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

1K potentiometer
DC motor
4-lead Stepper motor
Servo motor

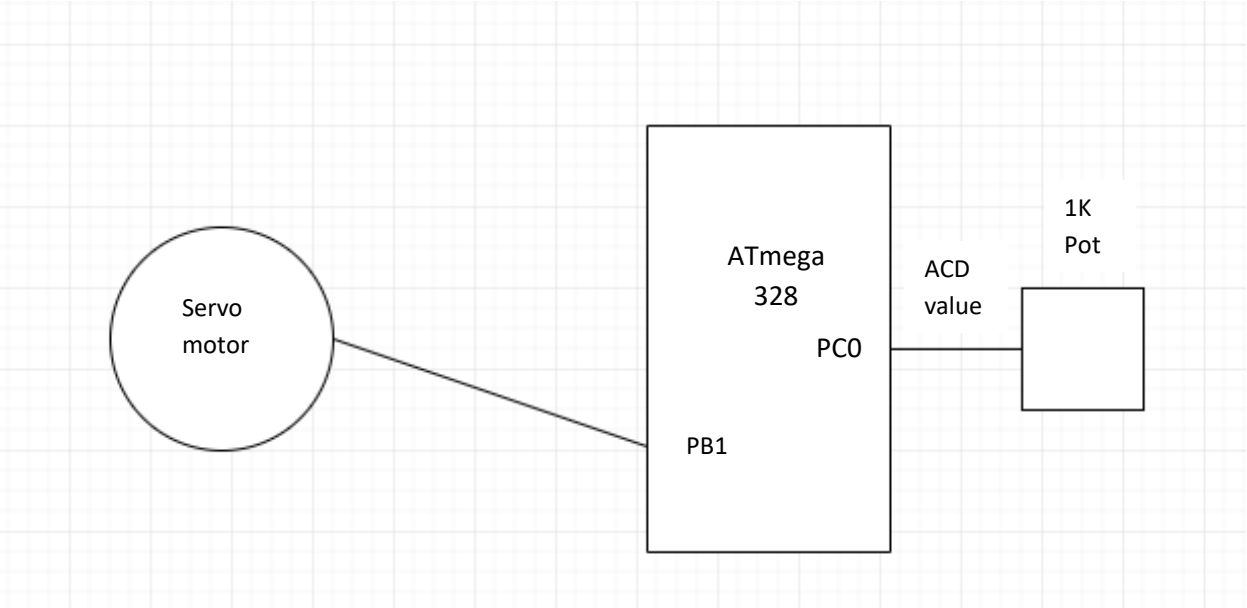
Task1 Block Diagram:



Task 2 Block Diagram:



Task 3 Block Diagram:



2. INITIAL/DEVELOPED CODE OF TASK 1/A

```
#include <avr/io.h>          //standard AVR header
int main (void)
{
   DDRB = 0xFF;              //make Port B an output
   DDRD = 0xFF;              //make Port D an output
   DDRA = 0;                 //make Port A an input for ADC input
   ADCSRA= 0x87;             //make ADC enable and select ck/128
   ADMUX= 0xC0;              //2.56V Vref, ADC0 single ended input
                             //data will be right-justified
   while (1){
     ADCSRA|=(1<<ADSC);      //start conversion
     while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish
     PORTD = ADCL;           //give the low byte to PORTD
     PORTB = ADCH;           //give the high byte to PORTB
   }
   return 0;
}
```

3. INITIAL/DEVELOPED CODE OF TASK 2/B

```
#define F_CPU    8000000UL       //XTAL = 8 MHz
#include "avr/io.h"
#include "util/delay.h"

int main ()
{
      DDRA = 0x00;
      DDRB = 0xFF;
      while (1)
      {
            if( (PINA&0x80) == 0)
            {
                  PORTB = 0x66;
                  _delay_ms (100);
                  PORTB = 0xCC;
                  _delay_ms (100);
                  PORTB = 0x99;
                  _delay_ms (100);
                  PORTB = 0x33;
                  _delay_ms (100);
            }
            else
            {
                  PORTB = 0x66;
                  _delay_ms (100);
                  PORTB = 0x33;
                  _delay_ms (100);
                  PORTB = 0x99;
                  _delay_ms (100);
                  PORTB = 0xCC;
                  _delay_ms (100);
            }
      }
}
```

4. INITIAL/DEVELOPED CODE OF TASK 3/C

```
#include <avr\io.h>
int main(void)
{
        //Port D 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;
    TCCR1AI=(0<<COM1A0)I(1<<COM1A1)I(0<<COM1B0)I(0<<COM1B1)I
    (0<<FOC1A)I(0<<FOC1B)I(1<<WGM11)I(0<<WGM10);
    TCCR1BI=(0<<ICNC1)I(0<<ICES1)I(1<<WGM13)I(1<<WGM12)I
    (0<<CS12)I(1<<CS11)I(0<<CS10);
    //start timer with prescaler 8
    for (;;)
    {
        if(bit_is_clear(PIND, 0))
        {
        //increase duty cycle
        OCR1A+=10;
        loop_until_bit_is_set(PIND, 0);
        }
        if(bit_is_clear(PIND, 1))
        {
        //decease duty cycle
        OCR1A-=10;
        loop_until_bit_is_set(PIND, 1);
        }
    }
}
```

5. MODIFIED CODE OF TASK 2/A from TASK 1/A

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

int main()
{

    ADCSRA = 0x87;          //ADC enable and ADC prescaler 128
    ADMUX = 0x60;           //AVcc as reference (5V) and left justified
    DDRB = 0xFF;            //set PORTB as output
    PORTD |= (1<<2);        //set up pull up resistor
    OCR1A = 0;              //0% duty cycle initially
    TCCR1B = 0x0D;          //prescaler of 1024
    TCCR1A = 0x83;          //non-inverting mode, fast PWM 10 bit
    EIMSK |= (1<<INT0);     //enable external interrupt 0
    EICRA |= (1<<ISC01);    //falling edge trigger
    sei();                  //enable interrupts

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

    }
}
ISR (INT0_vect)
{

    EIFR |= (1<<INTF0);     //reset flag
    if((PORTB & 0b00000001) == 0b00000000)      //check if sending enable signal
    {
        PORTB |= (1<<0);                        //toggle enable if not
        if(ADCH > 220)
        {
            OCR1A = 973;        //95% duty cycle
        }
        else
            OCR1A = 0;          //0% duty cycle
    }
    else
    {
        PORTB &= ~(1<<0);       //turn enable off
        OCR1A = 0;              //0% duty cycle
    }


}
```

6. MODIFIED CODE OF TASK 3/B from TASK 2/B

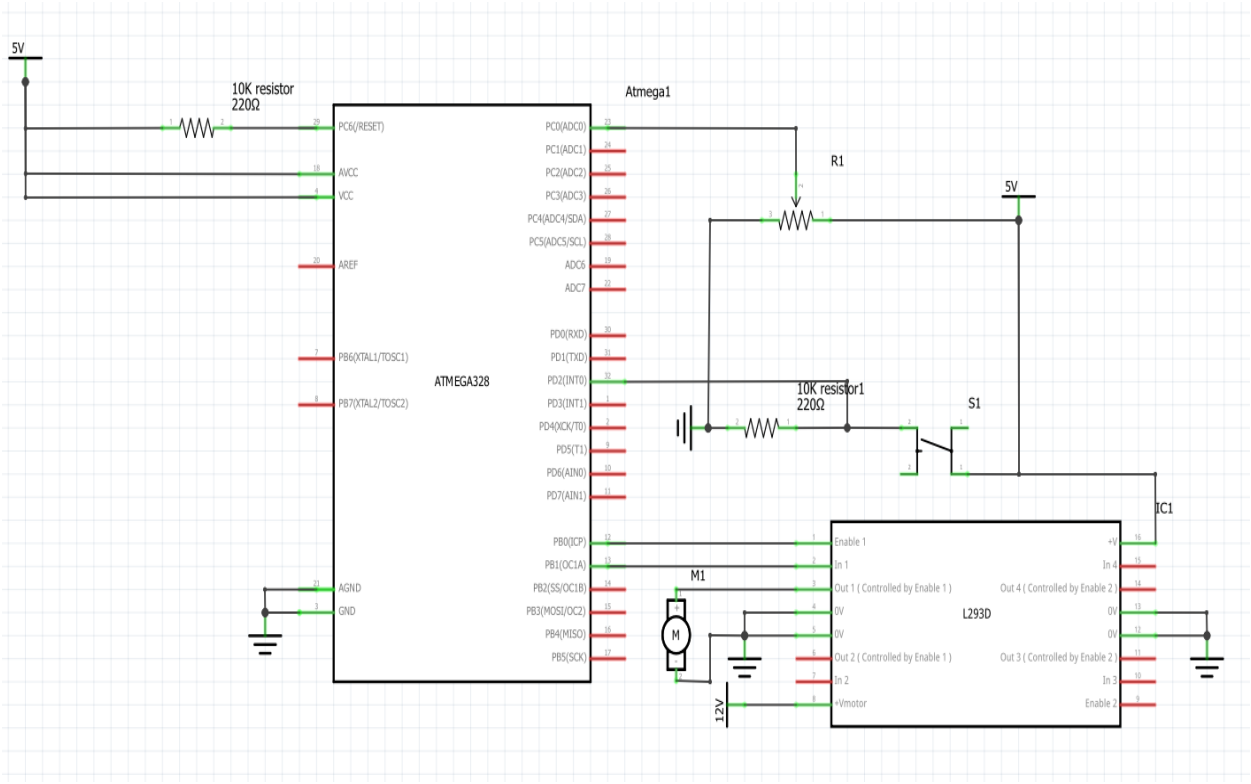
```
#include <avr/io.h>
#define F_CPU 8000000UL

int main(void)
{
    unsigned int speed;                                 //holder for ADCH value
    DDRC = 0x00;                                        //set port c as an input
    DDRD = 0xFF;                                        //set port d as an output
    DDRC = 0;                                           //set PC0 as input for ADC
    ADCSRA = 0x87;                                      //ADC enable and ADC prescaler 128
    ADMUX = 0x60;                                       //AVcc as reference (5V) and left justified
    TCCR1A = 0;                                         //initialize TCCR1A
    TCCR1B |= (1 << WGM12) | (1 << CS12) | (1 << CS10);     //mode CTC prescaler 1024
    OCR1A = 781;                                        //100ms second delay
    TCNT1 = 0;                                          //initialize counter


    while (1)
    {
        while ((TIFR1 & (1 << OCF1A)) == 0)
        {
            //wait for timer overflow
        }
        PORTD = 0x03;
        TIFR1 |= (1<< OCF1A);       //reset counter flag

        while ((TIFR1 & (1 << OCF1A))== 0)
        {
            //wait for timer overflow
        }
        PORTD = 0x42;
        TIFR1 |= (1<< OCF1A);       //reset counter flag

        while ((TIFR1 & (1 << OCF1A))== 0)
        {
            //wait for timer overflow
        }
        PORTD = 0xC0;
        TIFR1 |= (1<< OCF1A);       //reset counter flag

        while ((TIFR1 & (1 << OCF1A))== 0)
        {
            //wait for timer overflow
        }
        PORTD = 0x81;
        TIFR1 |= (1<< OCF1A);       //reset counter flag

        //rotation finished
        //get new delay value
        ADCSRA |= (1<<ADSC);            //start conversion
        while ((ADCSRA & (1<<ADIF)) == 0)
        {
            //wait for conversion to finish
        }
        speed = ADCH;                   //get ADCH value
        OCR1A = 781 + (speed*12);       //update speed 100ms min to 500ms max



    }
}
```

7. MODIFIED CODE OF TASK 4/C from TASK 3/C

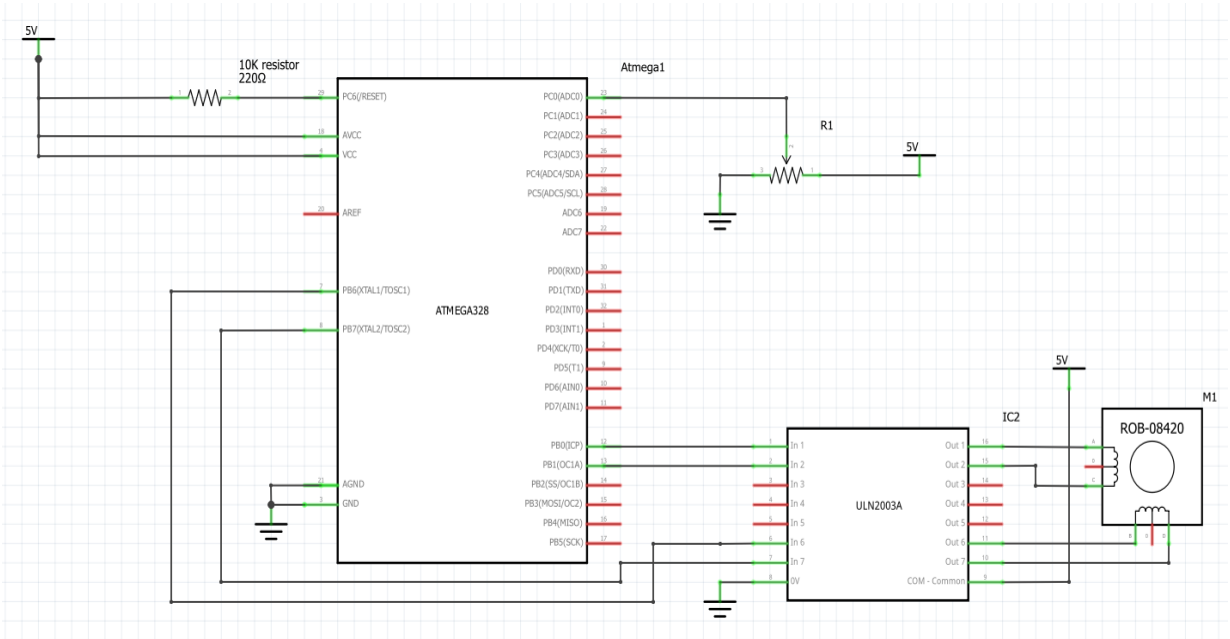
```c
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>

int main(void)
{
    DDRC = 0;                                           //set PC0 as input for ADC
    ADCSRA = 0x87;                                      //ADC enable and ADC pre-scaler 128
    ADMUX = 0x60;                                       //AVcc as reference (5V) and left justified
    TCCR1A |= (1<<COM1A1) | (1<<COM1B1)| (1<<WGM11);    //clear OC1A and OC1B on compare match
    TCCR1B |= (1<<WGM13)| (1<<WGM12)| (1<<CS11)|(1<<CS10);  //pre-scaler 64, fast PWM with ICR1 as top value
    ICR1 = 2500;                                        //fPWM = 50Hz, period 20ms
    DDRB= 0xFF;                                         //PortB as output
    unsigned int angle;
    while (1)
    {
        ADCSRA |= (1<<ADSC);                            //start conversion
        while ((ADCSRA & (1<<ADIF)) == 0)
        {
            //wait for conversion to finish
        }
        ADCSRA |= (1<<ADIF);                            //reset flag bit
        angle = ADCH;                                   //store ADC value into variable
        OCR1A = (angle/2)+125;                          //convert ADC value to PWM value
        _delay_ms(1500);                                //delay 1.5 seconds for motor to move
        //OCR1A value will range from 125 to approximately 250 which corresponds to a 1ms to 2ms pulse

    }
}
```

8. SCHEMATICS

Task 1 Schematic:

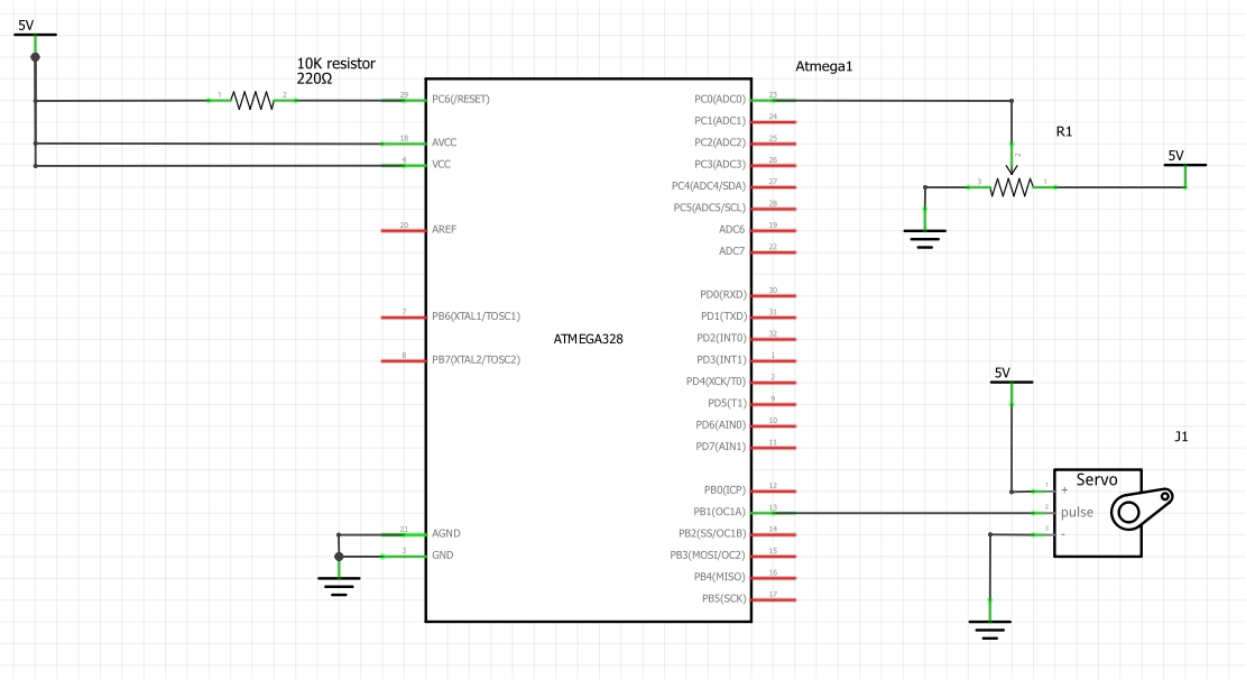


Task 2 Schematic:



The stepper motor in my lab kit only has 4 control wires plus one for powering it and fritzing only has 6 lead stepper motors, so I ignored the center tapped wires.

Task 3 Schematic

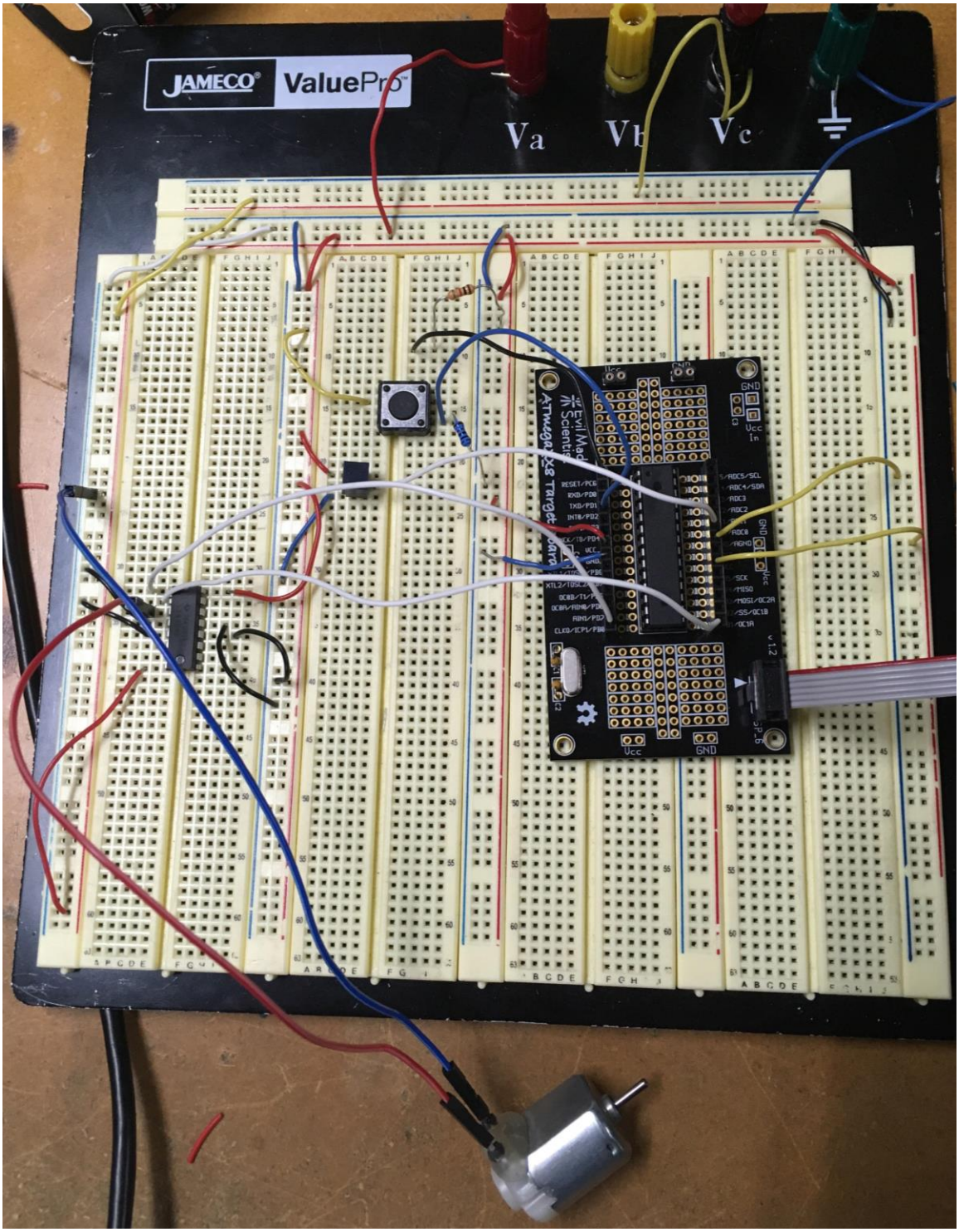


9. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

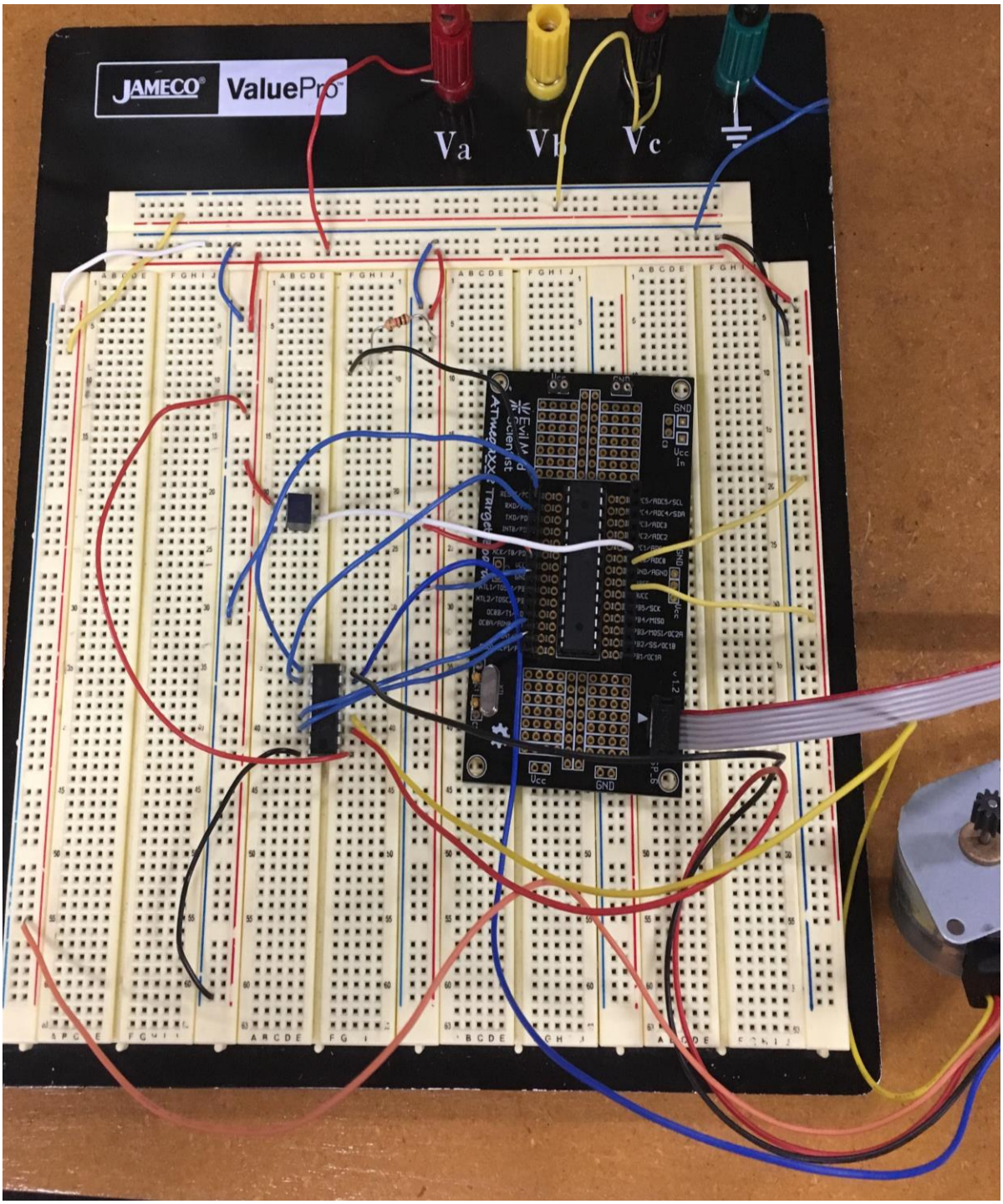
Refer to videos for outputs

10. SCREENSHOT OF EACH DEMO (BOARD SETUP)

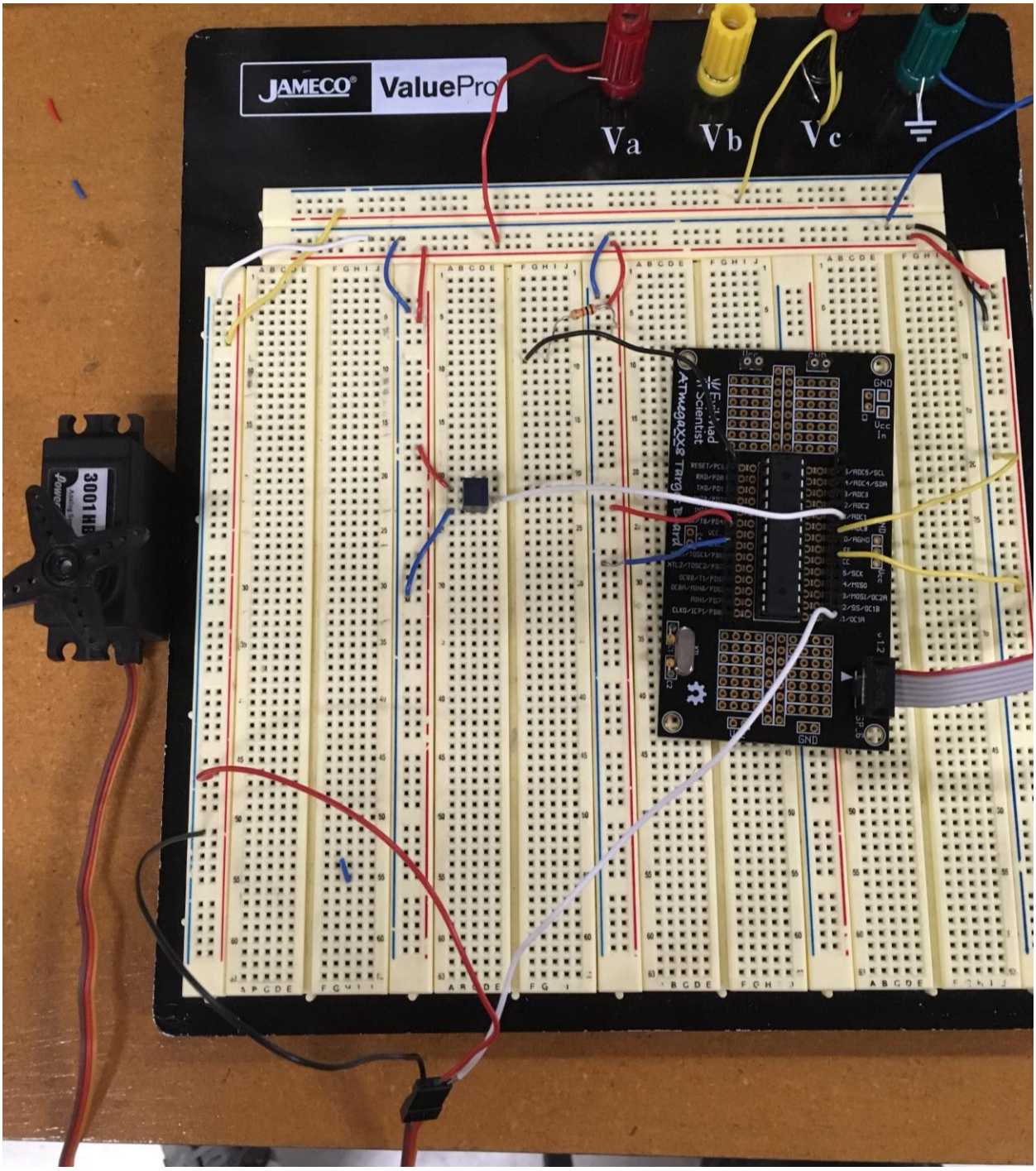
Task 1:



Task 2:



Task 3:



11. VIDEO LINKS OF EACH DEMO

Playlist - https://www.youtube.com/channel/UCX_dEuWexNMLRw5YqdTRQTg/playlists

Task1 - https://www.youtube.com/watch?v=C2g4so_71tw

Task2 - https://www.youtube.com/watch?v=7d5P36XGX1k

Task3- https://www.youtube.com/watch?v=fXSHr5Kn8aw

12. GITHUB LINK OF THIS DA

https://github.com/nhanuscin/submit/tree/master/DA4

Student Academic Misconduct Policy

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

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

Nathan Hanuscin