CpE301 - Design Assignment 4

Design Assignment 4:
DUE: TBA
The goal of the assignment is to develop the above code to do the following:

1. 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.
2. 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.
3. 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.

Submission:

The following are required for successful completion of the design assignment:
    a. AVR C code that has been compiled and working.
    b. The C code should be well documented with explanation of every instruction.
    c. A word document that contains the flow chart of the assembly code along with the snapshots of the schematics, components connected on the breadboard and screenshots.

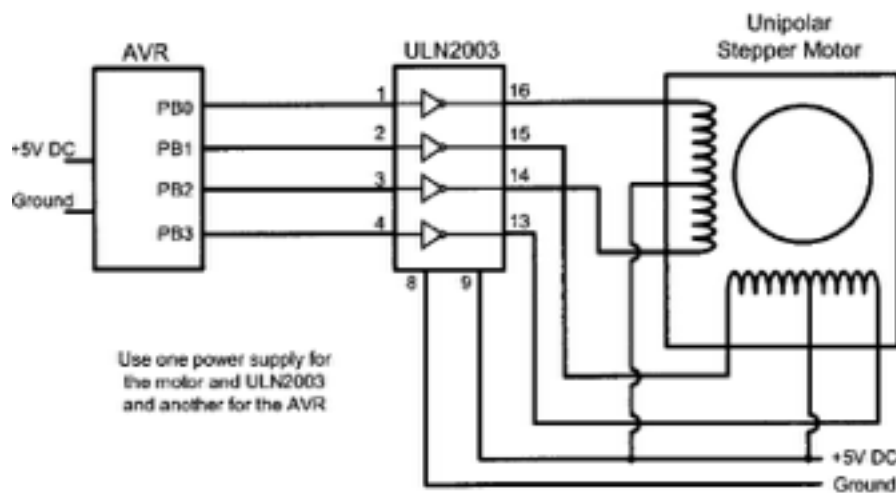Task 1: ADC 33.3%, Task 1: UART 33.3%, Task 2: 33.3%.

NOTES:
DC MOTOR CONTROL:
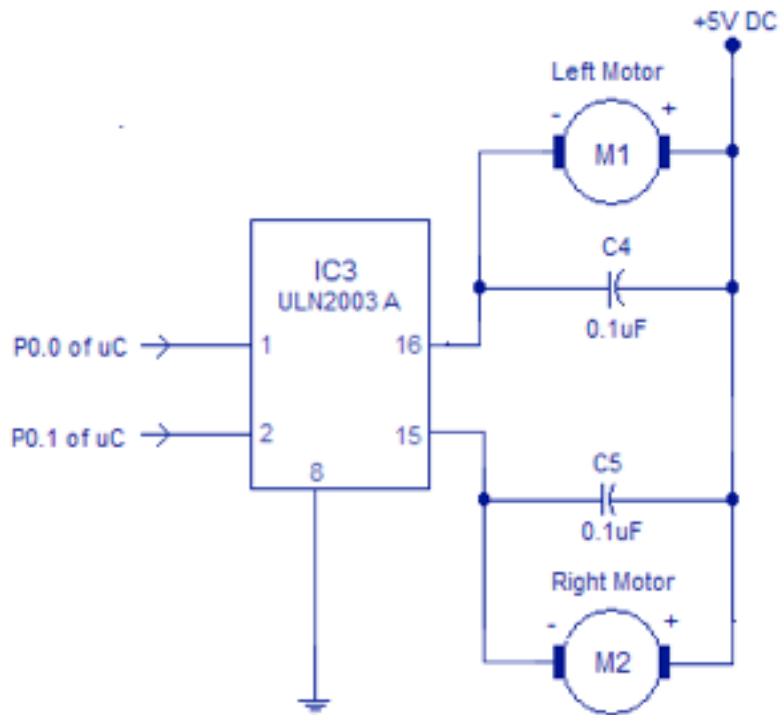
STEPPER MOTOR CONTROL:
Write a program to monitor the status of SW and perform the following:
(a) If SW = 0, the stepper motor moves clockwise.
(b) If SW = 1, the stepper motor moves counter clockwise.

```c
#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);
        }
    }
}
```

SERVO MOTOR:

Servo Motor PWM Timing Diagram
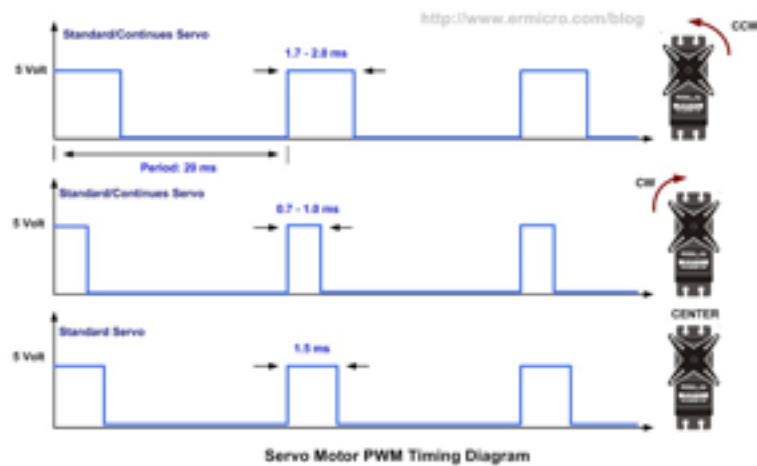
```c
#include <avr\io.h>
int main(void)
{
    //Port D pins as input
    DDRD=0x00;
    //Enable internal pull ups
    PORTD=0xFF;
    //Set PORTB1 pin as output
    DDRB=0xFF;
```

```
            //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;
            TCCR1A|=(0<<COM1A0)|(1<<COM1A1)|(0<<COM1B0)|(0<<COM1B1)|
            (0<<FOC1A)|(0<<FOC1B)|(1<<WGM11)|(0<<WGM10);
            TCCR1B|=(0<<ICNC1)|(0<<ICES1)|(1<<WGM13)|(1<<WGM12)|
            (0<<CS12)|(1<<CS11)|(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);
                    }
            }
}
```

PS: Use the A2D program below as a reference to read the pot values.

```
#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;
}
```