CPE301 – SPRING 2022

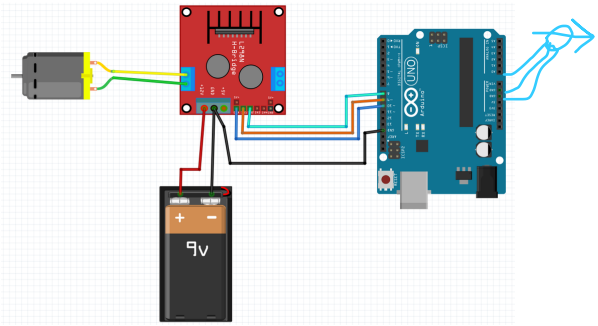
Design Assignment 5

Student Name: David Nakasone
Student #: 2001646072
Student Email: nakasd3@unlv.nevada.edu
Primary Github address: https://github.com/davenakasone
Directory: https://github.com/davenakasone/cpe301_David_Nakasone

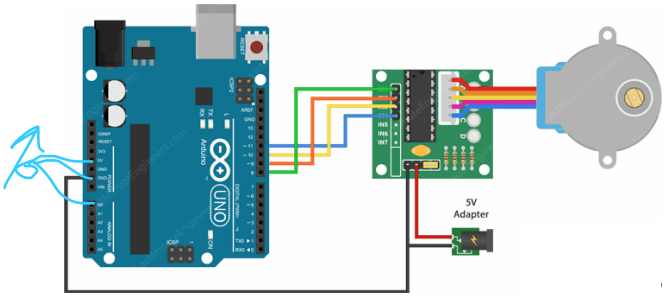
1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

See the pin assignments in the code heading of each task

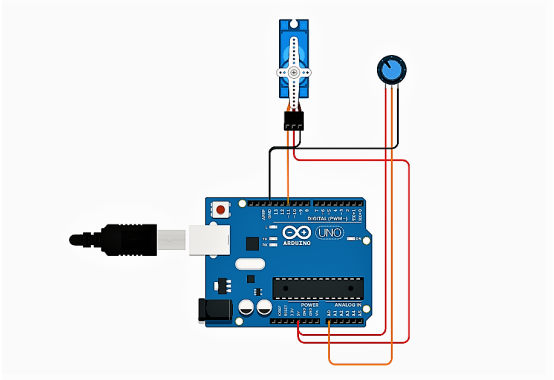
Task1, DC motor



Task2, stepper motor

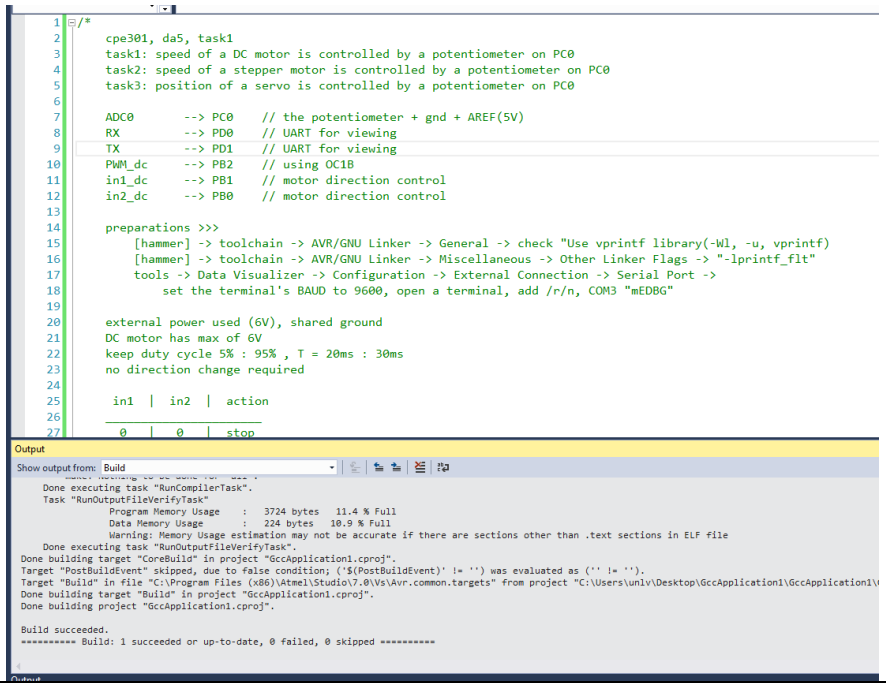


Task3, servo motor



2. INITIAL/MODIFIED/DEVELOPED CODE, all tasks

Task1, C code, with pins:




```
/*
    cpe301, da5, task1
    task1: speed of a DC motor is controlled by a potentiometer on PC0
    task2: speed of a stepper motor is controlled by a potentiometer on PC0
    task3: position of a servo is controlled by a potentiometer on PC0

    ADC0        --> PC0    // the potentiometer + gnd + AREF(5V)
    RX          --> PD0    // UART for viewing
    TX          --> PD1    // UART for viewing
    PWM_dc      --> PB2    // using OC1B
    in1_dc      --> PB1    // motor direction control
    in2_dc      --> PB0    // motor direction control

    preparations >>>
        [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u,
vprintf)
        [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-
lprintf_flt"
        tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
```

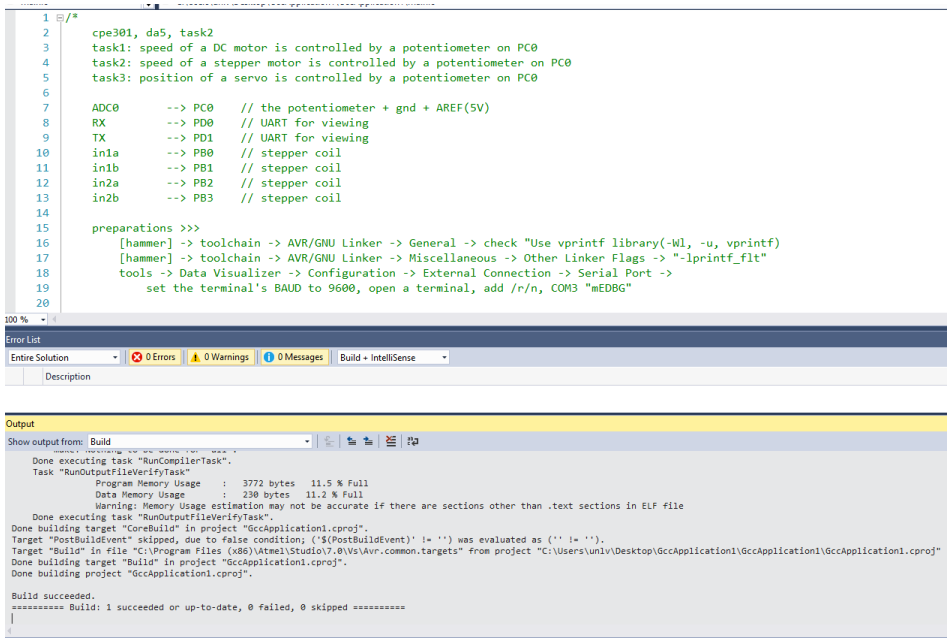
```
            set the terminal's BAUD to 9600, open a terminal, add /r/n, COM3 "mEDBG"


    external power used (6V), shared ground
    DC motor has max of 6V
    keep duty cycle 5% : 95% , T = 20ms : 30ms
    no direction change required


     in1  |  in2  |  action
    ________________________
      0   |   0   |  stop
      0   |   1   |  CW
      1   |   0   |  CCW
      1   |   1   | illegal


    T = 25ms, f_pwm = 1/T = 40 Hz, N = 64
    f_pwm = f_cpu / (N * [1 + ICR1]),, --> ICR1 as top @ 6249
*/


#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE ((F_CPU / (BAUD * 16UL)) - 1)
#define HELP_BUF 128
#define DEBUG_UART 123


#define ADC_MAX 1023     // highest raw ADC value, the minimum is 0
#define DC_WIDTH 6249    // sets ICR1 to get required frequency
#define DC_MIN 0         // for motor enable, 0% duty cycle
#define DC_MAX 6138      // for motor enable, 98% duty cycle ... maps 6x


#define DC_PWM 2     // motor enable using PWM, PB2
#define DC_IN_1 1    // controls motor direction, PB1
#define DC_IN_2 0    // controls motor direction, PB0
#define PIN_POT 0    // potentiometer on PC0


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

```c
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <util/atomic.h>
#include <util/delay.h>

unsigned char helper[HELP_BUF];    // UART debugging
uint16_t adc_raw;                  // gets raw ADC value
uint16_t motor_duty;               // sets duty cycle using OCR1B of motor PWM enable

void read_adc (void);
void usart_putc (char send_char);
void usart_puts (const char* send_str);


int main(void)
{
    memset(helper, '\0', HELP_BUF);
    adc_raw = 0;
    motor_duty = DC_MIN;

    PORTC |= (1 << PIN_POT);                        // activate pull-up resistor for potentiometer
reading
    ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));    // 16Mhz/128 = 125Khz the ADC reference clock
    ADMUX |= (0<<REFS0);                            // Voltage reference from AREF
    ADCSRA |= (1<<ADEN);                            // turn on ADC
    ADCSRA |= (1<<ADSC);                            // get first conversion done

    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);       // turn on USART module, receive and transmit enabled
    UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);     // configure: asynchronous, 8-bit data, 1-bit stop
    UBRR0H = (BAUD_PRESCALE >> 8);               // sets baud rate, UBRR0 = { f_cpu / (16*BAUD) } - 1,
has 4 msb
    UBRR0L = BAUD_PRESCALE;

    DDRB = (1 << DC_PWM) | (1 << DC_IN_1) | (1 << DC_IN_2);             // outputs to motor driver
```

```
    PORTB |= (1 << DC_IN_1);                                        // direction is set
    ICR1 = DC_WIDTH;                                                // frequency of motor PWM
enable is set
    TCCR1A = (1 << COM1B1) | (1 << WGM11);                          // timer1, fast PWM, mode 14,
non-inverting, OC1B duty cycle
    TCCR1B = (1 << WGM13) | (1 << WGM12) | (1 << CS11) | (1 << CS10);    // prescale 64


    usart_puts("\r\n");
    usart_puts("         initialization complete, program begins...\r\n");
    usart_puts("\r\n");
    sei();


    while(1)
    {
        read_adc();
        motor_duty = 6 * adc_raw;
        if (motor_duty > DC_MAX || motor_duty < DC_MIN)
        {
            motor_duty = DC_MIN;
        }
        OCR1B = motor_duty;


        #ifdef DEBUG_UART
            sprintf(helper, "adc_raw=  %9d  ,  motor_duty=  %9d\r\n",
                adc_raw, motor_duty);
            usart_puts(helper);
        #endif
    }
    return EXIT_FAILURE;
}



////~~~~


void read_adc (void)
```

```
{
    ADMUX &= 0xF0;                    // clears channels
    ADMUX |= PIN_POT;                 // set new channel to read
    ADCSRA |= (1 << ADSC);            // starts a new conversion
    while(ADCSRA & (1 << ADSC)){}     // wait until the conversion is done
    adc_raw = ADCW;                   // gets the ADC value
}



////~~~~



void usart_putc (char send_char)
{
    // wait until data register not empty
    while ((UCSR0A & (1 << UDRE0)) == 0) {}
    UDR0 = send_char;
}



////~~~~



void usart_puts (const char* send_str)
{
    // send the string, char by char, until send_str[x] == NULL
    while (*send_str)
    {
        usart_putc(*send_str++);
    }
}



////////~~~~~~~~END>  main.c
```

Task2, C code, with pins:



```
/*
    cpe301, da5, task2
    task1: speed of a DC motor is controlled by a potentiometer on PC0
    task2: speed of a stepper motor is controlled by a potentiometer on PC0
    task3: position of a servo is controlled by a potentiometer on PC0

    ADC0        --> PC0    // the potentiometer + gnd + AREF(5V)
    RX          --> PD0    // UART for viewing
    TX          --> PD1    // UART for viewing
    in1a        --> PB0    // stepper coil
    in1b        --> PB1    // stepper coil
    in2a        --> PB2    // stepper coil
    in2b        --> PB3    // stepper coil

    preparations >>>
        [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u, vprintf)
        [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-lprintf_flt"
        tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
            set the terminal's BAUD to 9600, open a terminal, add /r/n, COM3 "mEDBG"
```

```
/*

    cpe301, da5, task2

    task1: speed of a DC motor is controlled by a potentiometer on PC0

    task2: speed of a stepper motor is controlled by a potentiometer on PC0

    task3: position of a servo is controlled by a potentiometer on PC0


    ADC0        --> PC0    // the potentiometer + gnd + AREF(5V)

    RX          --> PD0    // UART for viewing

    TX          --> PD1    // UART for viewing

    in1a        --> PB0    // stepper coil

    in1b        --> PB1    // stepper coil

    in2a        --> PB2    // stepper coil

    in2b        --> PB3    // stepper coil


    preparations >>>

        [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u,
vprintf)

        [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-
lprintf_flt"

        tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
```

```
            set the terminal's BAUD to 9600, open a terminal, add /r/n, COM3 "mEDBG"


    control the steps, 10ms to ?


    1a | 1b | 2a | 2b
    _________________
    1  | 0  | 0  | 0
    0  | 1  | 0  | 0
    0  | 0  | 1  | 0
    0  | 0  | 0  | 1
    ...
*/
//#define TEST 3000
#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE ((F_CPU / (BAUD * 16UL)) - 1)
#define HELP_BUF 128
#define DEBUG_UART 123

#define ADC_MAX 1023          // highest raw ADC value, the minimum is 0
#define STEP_FASTEST 1500     // fastest by smallest pause between steps
#define STEP_SLOWEST 10000    // slowest by longest pause between steps
#define STEP_1 0x8
#define STEP_2 0x4
#define STEP_3 0x2
#define STEP_4 0x1

#define PINS_STEP 0xF   // the stepper coil pins on PB[0:3]
#define PIN_POT 0       // potentiometer on PC0

#include <avr/interrupt.h>
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
```

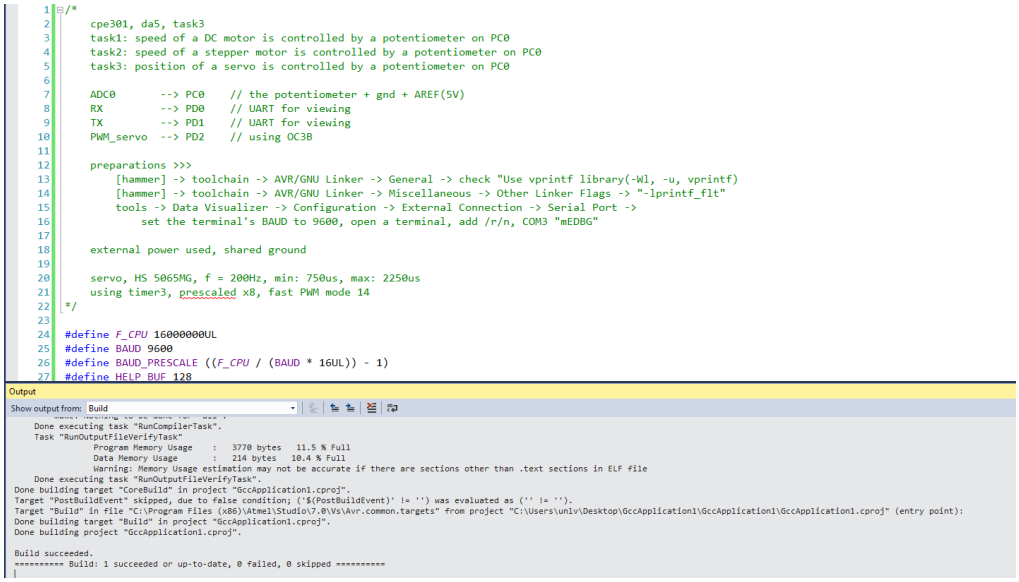
```c
#include <util/atomic.h>
#include <util/delay.h>

unsigned char helper[HELP_BUF];    // UART debugging
uint16_t adc_raw;                  // gets raw ADC value
uint16_t stepper_pause_us;         // amount of time to pause between step

void make_delay_us();
void read_adc (void);
void usart_putc (char send_char);
void usart_puts (const char* send_str);


int main(void)
{
    memset(helper, '\0', HELP_BUF);
    adc_raw = 0;
    stepper_pause_us = STEP_SLOWEST;

    PORTC |= (1 << PIN_POT);                          // activate pull-up resistor for potentiometer
reading
    ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));    // 16Mhz/128 = 125Khz the ADC reference clock
    ADMUX |= (0<<REFS0);                              // Voltage reference from AREF
    ADCSRA |= (1<<ADEN);                              // turn on ADC
    ADCSRA |= (1<<ADSC);                              // get first conversion done

    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);       // turn on USART module, receive and transmit enabled
    UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);     // configure: asynchronous, 8-bit data, 1-bit stop
    UBRR0H = (BAUD_PRESCALE >> 8);               // sets baud rate, UBRR0 = { f_cpu / (16*BAUD) } - 1,
has 4 msb
    UBRR0L = BAUD_PRESCALE;

    DDRB = PINS_STEP;    // the stepper coil pins as output
    PORTB = 0;           // motor begins turned off

    usart_puts("\r\n");
```

```
usart_puts("         initialization complete, program begins...\r\n");
usart_puts("\r\n");
sei();

while(1)
{
    read_adc();
    stepper_pause_us = (((STEP_FASTEST-STEP_SLOWEST)/ADC_MAX) * adc_raw) + STEP_SLOWEST;
    if (stepper_pause_us > STEP_SLOWEST ||
        stepper_pause_us < STEP_FASTEST  )
    {
        stepper_pause_us = STEP_SLOWEST;
    }
    PORTB = STEP_1;
    #ifdef TEST
        _delay_us(TEST);
    #else
        make_delay_us();
    #endif
    PORTB = STEP_2;
    #ifdef TEST
        _delay_us(TEST);
    #else
        make_delay_us();
    #endif
    PORTB = STEP_3;
    #ifdef TEST
        _delay_us(TEST);
    #else
        make_delay_us();
    #endif
    PORTB = STEP_4;
    #ifdef TEST
        _delay_us(TEST);
    #else
```

```
            make_delay_us();
        #endif


        #ifdef DEBUG_UART
            sprintf(helper, "adc_raw=  %9d  ,  stepper_pause_us=  %9d\r\n",
                adc_raw, stepper_pause_us);
            usart_puts(helper);
        #endif
    }
    return EXIT_FAILURE;
}



////~~~~



void make_delay_us()
{
    uint16_t counter = 0;
    while (counter < stepper_pause_us)
    {
        _delay_us(1);
        counter++;
    }
}



////~~~~



void read_adc (void)
{
    ADMUX &= 0xF0;                    // clears channels
    ADMUX |= PIN_POT;                 // set new channel to read
    ADCSRA |= (1 << ADSC);            // starts a new conversion
    while(ADCSRA & (1 << ADSC)){}     // wait until the conversion is done
    adc_raw = ADCW;                   // gets the ADC value
```

```
}



////~~~~



void usart_putc (char send_char)
{
    // wait until data register not empty
    while ((UCSR0A & (1 << UDRE0)) == 0) {}
    UDR0 = send_char;
}



////~~~~



void usart_puts (const char* send_str)
{
    // send the string, char by char, until send_str[x] == NULL
    while (*send_str)
    {
        usart_putc(*send_str++);
    }
}



////////~~~~~~~~END>  main.c
```

Task3, C code, with pins:



```
/*
    cpe301, da5, task3
    task1: speed of a DC motor is controlled by a potentiometer on PC0
    task2: speed of a stepper motor is controlled by a potentiometer on PC0
    task3: position of a servo is controlled by a potentiometer on PC0

    ADC0        --> PC0    // the potentiometer + gnd + AREF(5V)
    RX          --> PD0    // UART for viewing
    TX          --> PD1    // UART for viewing
    PWM_servo   --> PD2    // using OC3B

    preparations >>>
        [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u, vprintf)
        [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-lprintf_flt"
        tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
            set the terminal's BAUD to 9600, open a terminal, add /r/n, COM3 "mEDBG"

    external power used, shared ground

    servo, HS 5065MG, f = 200Hz, min: 750us, max: 2250us
    using timer3, prescaled x8, fast PWM mode 14
*/

#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE ((F_CPU / (BAUD * 16UL)) - 1)
#define HELP_BUF 128
```

```
Output
Show output from: Build
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage     :   3770 bytes   11.5 % Full
                Data Memory Usage        :   214 bytes   10.4 % Full
                Warning: Memory Usage estimation may not be accurate if there are sections other than .text sections in ELF file
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "GccApplication1.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\unlv\Desktop\GccApplication1\GccApplication1\GccApplication1.cproj" (entry point):
Done building target "Build" in project "GccApplication1.cproj".
Done building project "GccApplication1.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```


```
/*
    cpe301, da5, task3
    task1: speed of a DC motor is controlled by a potentiometer on PC0
    task2: speed of a stepper motor is controlled by a potentiometer on PC0
    task3: position of a servo is controlled by a potentiometer on PC0

    ADC0        --> PC0    // the potentiometer + gnd + AREF(5V)
    RX          --> PD0    // UART for viewing
    TX          --> PD1    // UART for viewing
    PWM_servo   --> PD2    // using OC3B

    preparations >>>
        [hammer] -> toolchain -> AVR/GNU Linker -> General -> check "Use vprintf library(-Wl, -u,
vprintf)
        [hammer] -> toolchain -> AVR/GNU Linker -> Miscellaneous -> Other Linker Flags -> "-
lprintf_flt"
        tools -> Data Visualizer -> Configuration -> External Connection -> Serial Port ->
            set the terminal's BAUD to 9600, open a terminal, add /r/n, COM3 "mEDBG"

    external power (5V) used, shared ground
    servo, HS 5065MG, f = 200Hz, min: 750us, max: 2250us
    using timer3, prescaled x8, fast PWM mode 14
```

```
*/

#define F_CPU 16000000UL
#define BAUD 9600
#define BAUD_PRESCALE ((F_CPU / (BAUD * 16UL)) - 1)
#define HELP_BUF 128

#define ADC_MAX 1023
#define SERVO_WIDTH 10000    // f_pwm  = 200 Hz,  200 = f_cpu / (N * (TOP+1)  -> TOP = 9999
#define SERVO_START 1400     // 750us min,  f = f_cpu / N = 2e6, T = 1/f = 0.5us  ->  750/0.5 =
1500...don't max out
#define SERVO_STOP 4600      // 2250us max, "                                    "  2250/0.5 =
4500...don't max out
#define SERVO_HANG 100       // ms, change based on step size, the wait time
#define SERVO_RETURN 500     // time for reset
#define SERVO_RANGE (SERVO_STOP - SERVO_START)


#define PIN_SERVO 2    // servo uses PD2 for PWM
#define PIN_POT 0      // potentiometer on PC0

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

volatile unsigned char helper [HELP_BUF];
uint16_t adc_raw;
uint16_t servo_counter;

void read_adc (void);
void usart_putc (char send_char);
void usart_puts (const char* send_str);
```

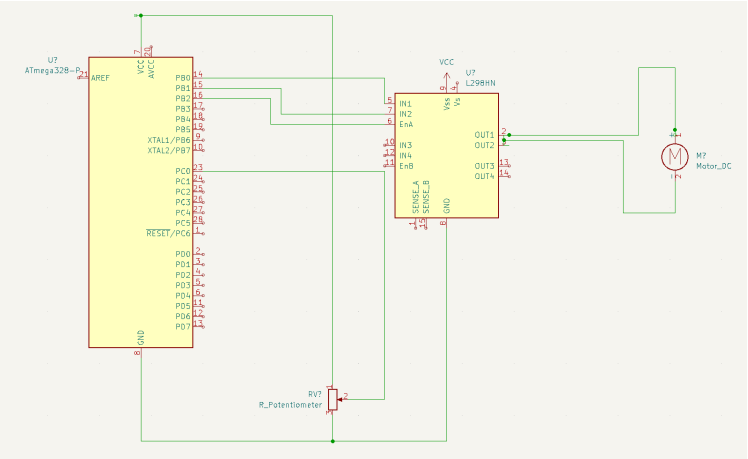
```
int main(void)
{
    memset(helper, '\0', HELP_BUF);
    adc_raw = 0;
    servo_counter = SERVO_START;


    PORTC |= (1 << PIN_POT);                        // activate pull-up resistor for potentiometer
reading
    ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));    // 16Mhz/128 = 125Khz the ADC reference clock
    ADMUX |= (0<<REFS0);                            // Voltage reference from AREF
    ADCSRA |= (1<<ADEN);                            // turn on ADC
    ADCSRA |= (1<<ADSC);                            // get first conversion done

    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);       // turn on USART module, receive and transmit enabled
    UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);     // configure: asynchronous, 8-bit data, 1-bit stop
    UBRR0H = (BAUD_PRESCALE >> 8);               // sets baud rate, UBRR0 = { f_cpu / (16*BAUD) } - 1,
has 4 msb
    UBRR0L = BAUD_PRESCALE;

    DDRD |= (1 << PIN_SERVO);                           // dedicated PWM pin for timer 3 as output
    PORTD |= (1 << PIN_SERVO);                          // this must be done to disable OC4B, the
pin is shared
    ICR3 = SERVO_WIDTH;                                 // the "TOP", makes the time period, OCR3A
makes duty cycle
    TCCR3A |= (1 << COM3B1) | (1 << WGM31);             // non inverting, mode14
    TCCR3B |= (1 << WGM33) | (1 << WGM32) | (1 << CS31);    // mode 14, prescale 8

    usart_puts("\r\n");
    usart_puts("         initialization complete, program begins...\r\n");
    usart_puts("\r\n");
    sei();

    while(1)
```

```
    {
        read_adc();
        servo_counter = (SERVO_RANGE / ADC_MAX) * adc_raw + SERVO_START;
        if (servo_counter > SERVO_STOP ||
            servo_counter < SERVO_START )
        {
            servo_counter = SERVO_START;
            _delay_ms(SERVO_RETURN);
        }

        OCR3B = servo_counter;

        sprintf(helper, "adc=  %9d  ,  servo:  %9d\r\n",
            adc_raw, servo_counter);
        usart_puts(helper);
        _delay_ms(SERVO_HANG);
    }
    return EXIT_FAILURE;
}


////~~~~


void read_adc (void)
{
    ADMUX &= 0xF0;                  // clears channels
    ADMUX |= PIN_POT;               // set new channel to read
    ADCSRA |= (1 << ADSC);          // starts a new conversion
    while(ADCSRA & (1 << ADSC)){}   // wait until the conversion is done
    adc_raw = ADCW;                 // gets the ADC value
}


////~~~~
```

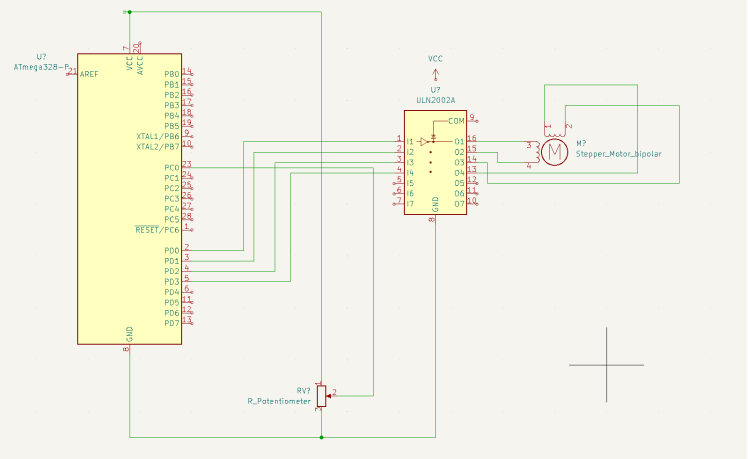
```
void usart_putc (char send_char)
{
    // wait until data register not empty
    while ((UCSR0A & (1 << UDRE0)) == 0) {}
    UDR0 = send_char;
}



////~~~~



void usart_puts (const char* send_str)
{
    // send the string, char by char, until send_str[x] == NULL
    while (*send_str)
    {
        usart_putc(*send_str++);
    }
}



////////~~~~~~~~END>  main.c
```

3. SCHEMATICS

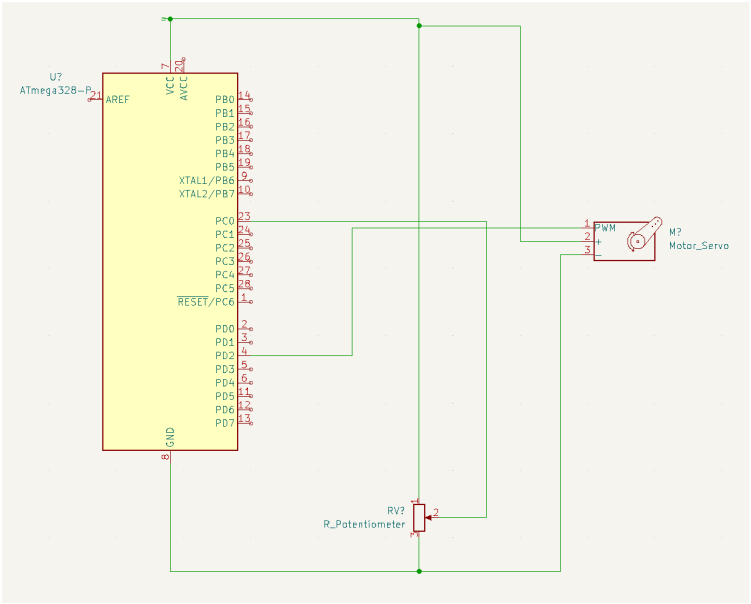
Task1, schematic



Task2, schematic

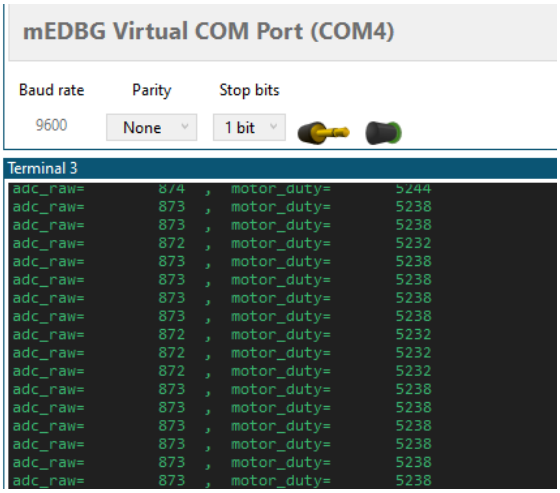


Task3, schematic

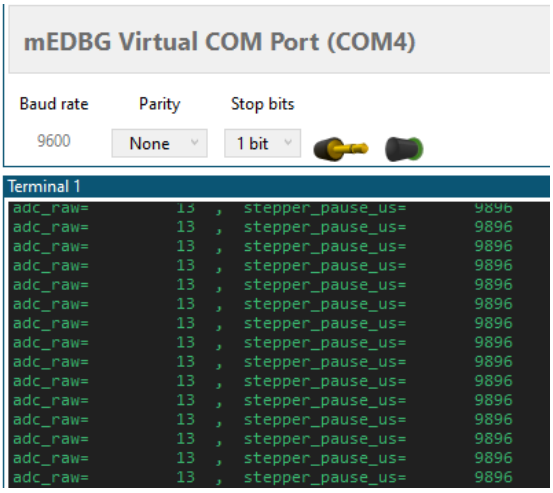


4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

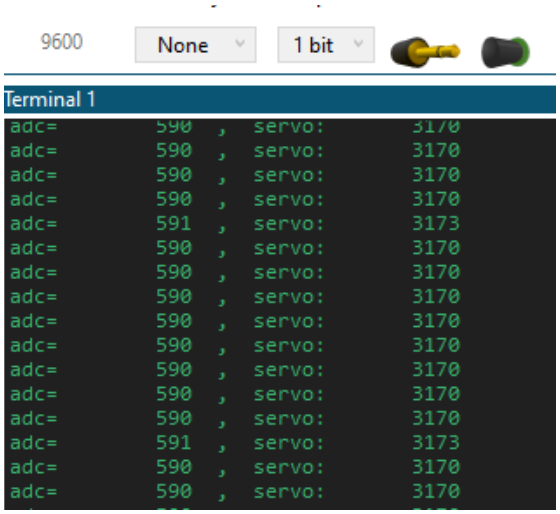
Task1, UART



Task2, UART

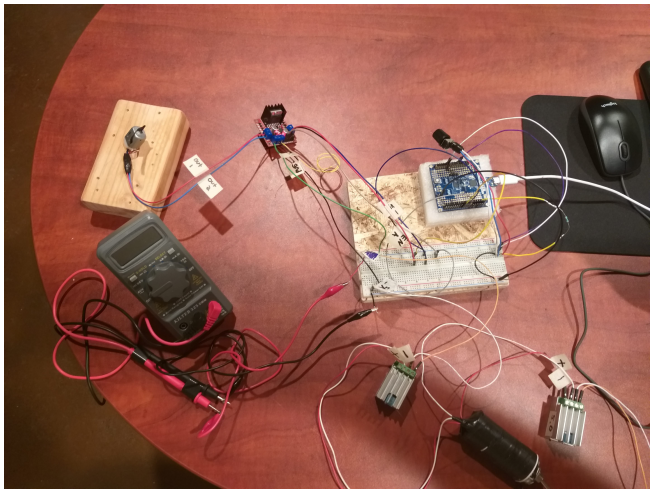


Task3, UART

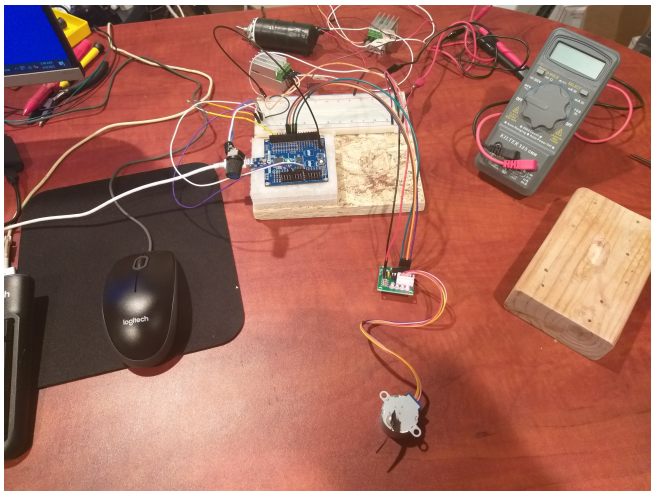


5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

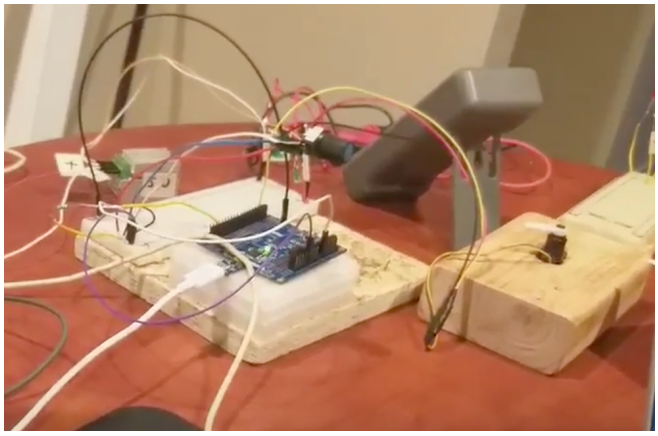
Task1, board



Task2, board



Task3, board



6. VIDEO LINKS OF EACH DEMO

task 1:https://youtu.be/4EG0HvC5S48
task 2:https://youtu.be/ESgJvJrX7Pc
task 3:https://youtu.be/Fn3XybCrqNg

7. GITHUB LINK OF THIS DA

https://github.com/davenakasone/cpe301_David_Nakasone/tree/main/Design_Assignmentz/DA5

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

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