CPE301 – SPRING 2019

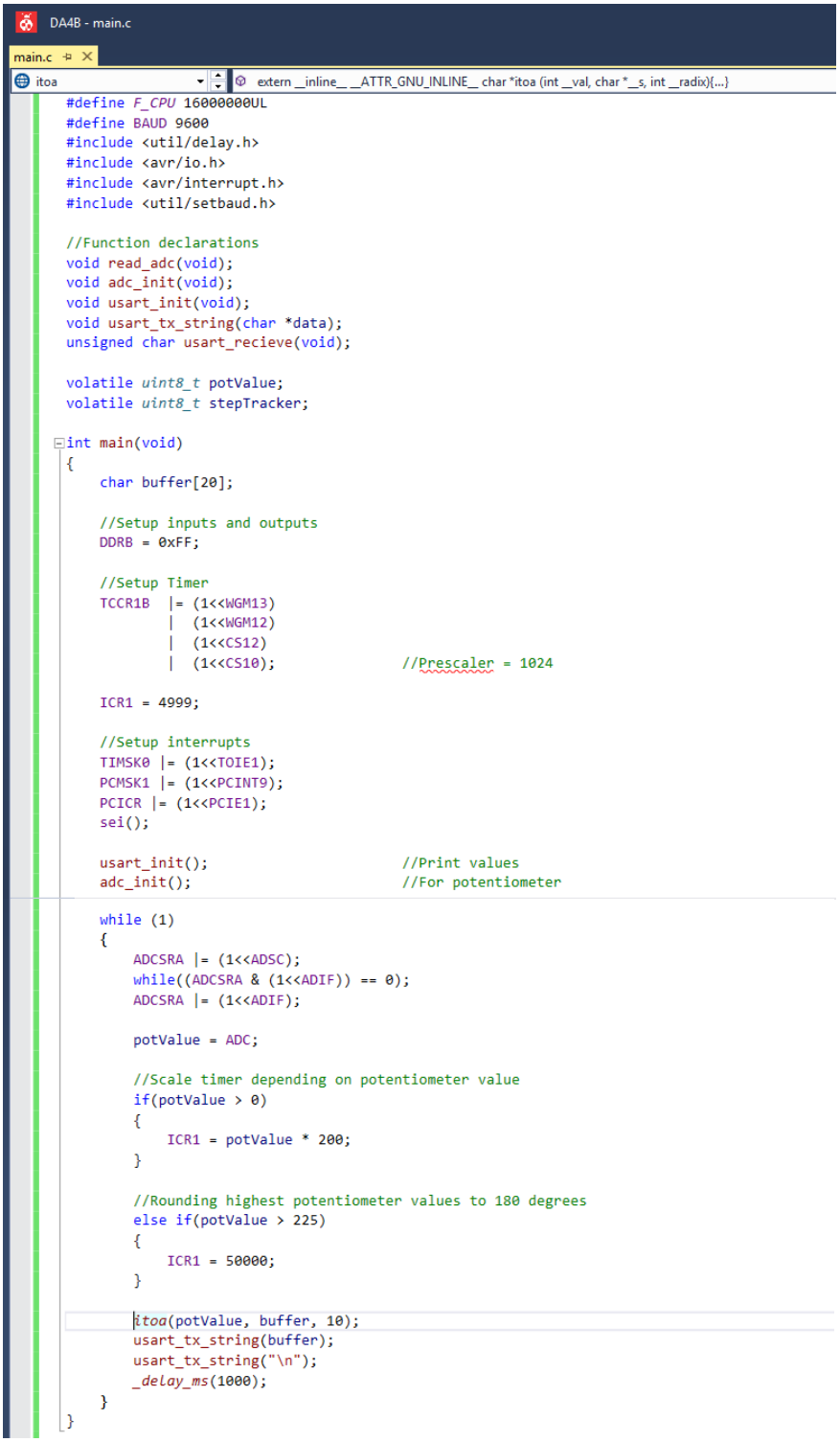
Design Assignment 4B

Student Name: John Duriman
Student #: 5002373995
Student Email: duriman@unlv.nevada.edu
Primary Github address: https://github.com/johnduriman/pirahnaplant.git
Directory: pirahnaplant/Design Assignments/DA4B/

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components used:
Atmega328P
Jumper Wires
Bread board
Stepper motor
Servo motor
ULN2003

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A




```c
#define F_CPU 16000000UL
#define BAUD 9600
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>

//Function declarations
void read_adc(void);
void adc_init(void);
void usart_init(void);
void usart_tx_string(char *data);
unsigned char usart_recieve(void);

volatile uint8_t potValue;
volatile uint8_t stepTracker;

int main(void)
{
    char buffer[20];

    //Setup inputs and outputs
    DDRB = 0xFF;

    //Setup Timer
    TCCR1B  |= (1<<WGM13)
            |  (1<<WGM12)
            |  (1<<CS12)
            |  (1<<CS10);              //Prescaler = 1024

    ICR1 = 4999;

    //Setup interrupts
    TIMSK0 |= (1<<TOIE1);
    PCMSK1 |= (1<<PCINT9);
    PCICR |= (1<<PCIE1);
    sei();

    usart_init();                      //Print values
    adc_init();                        //For potentiometer

    while (1)
    {
        ADCSRA |= (1<<ADSC);
        while((ADCSRA & (1<<ADIF)) == 0);
        ADCSRA |= (1<<ADIF);

        potValue = ADC;

        //Scale timer depending on potentiometer value
        if(potValue > 0)
        {
            ICR1 = potValue * 200;
        }

        //Rounding highest potentiometer values to 180 degrees
        else if(potValue > 225)
        {
            ICR1 = 50000;
        }

        itoa(potValue, buffer, 10);
        usart_tx_string(buffer);
        usart_tx_string("\n");
        _delay_ms(1000);
    }
}
```

```c
void adc_init(void)
{
    ADMUX = (0<<REFS1)  |                       //Reference Selection Bits
            (1<<REFS0)  |                       //AVcc - external cap at AREF
            (0<<ADLAR)  |                       //ADC Left Adjust Result
            (0<<MUX2)   |                       //Analog Channel Selection bits
            (0<<MUX1)   |                       //PC0 | PCINT8 | ADC0
            (0<<MUX0)   ;

    ADCSRA =    (1<<ADEN)   |                   //ADC Enable
                (0<<ADSC)   |                   //ADC Start Conversion
                (0<<ADATE)  |                   //ADC Auto Trigger Enable
                (0<<ADIF)   |                   //ADC Interrupt Flag
                (0<<ADIE)   |                   //ADC Interrupt Enable
                (1<<ADPS2)  |                   //ADC Prescalar Select Bits
                (0<<ADPS1)  |
                (1<<ADPS0)  ;
}

void usart_init(void)
{
    UBRR0H = UBRRH_VALUE;
    UBRR0L = UBRRL_VALUE;
    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00);         //8 bit data
    UCSR0B = _BV(RXEN0)  | _BV(TXEN0);          //Enable RX and TX
}

void usart_tx_string(char *data)
{
    while ((*data != '\0'))
    {
        while (!(UCSR0A & (1 << UDRE0)));
        UDR0 = *data;
        data++;
    }
}

ISR(TIMER1_OVF_vect)
{
    switch(stepTracker)
    {
        case '0':
            PORTB = 0x09;
            break;
        case '1':
            PORTB = 0x03;
            break;
        case '2':
            PORTB = 0x06;
            break;
        case '3':
            PORTB = 0x0c;
            break;
        default:
            break;
    }
    stepTracker++;
}
```

3. DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

Solution1

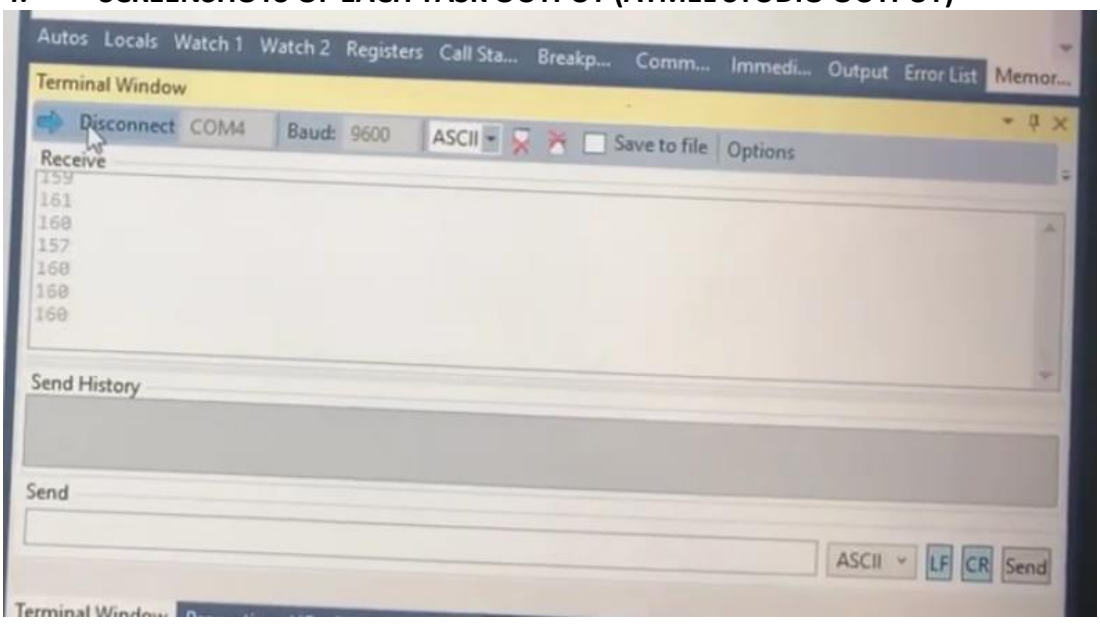
main.c*

usart_recieve — unsigned char usart_recieve(void)

```
#define F_CPU 16000000UL
#define BAUD 9600
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/setbaud.h>

//Function declarations
void read_adc(void);
void adc_init(void);
void usart_init(void);
void usart_tx_string(char *data);
unsigned char usart_recieve(void);

volatile uint8_t potValue;

int main(void)
{
    char buffer[20];

    //Setup inputs and outputs
    DDRB = 0xFF;                        //Make PORTB output

    //Setup Timer
    TCCR1A  |= (1<<WGM11)
            |  (1<<COM1A1)
            |  (1<<COM1B1);             //Non inverted PWM
    TCCR1B  |= (1<<WGM13)
            |  (1<<WGM12)
            |  (1<<CS11)
            |  (1<<CS10);               //Prescaler = 64 MODE 14(FAST PWM)
    ICR1 = 4999;

    //Setup interrupts
    PCMSK1 |= (1<<PCINT9);
    PCICR |= (1<<PCIE1);
    sei();

    usart_init();                       //Print values
    adc_init();                         //For potentiometer

    while (1)
    {
        ADCSRA |= (1<<ADSC);
        while((ADCSRA & (1<<ADIF)) == 0);
        ADCSRA |= (1<<ADIF);

        potValue = ADC;

        //0 degrees is at 97
        if(potValue > 0)
        {
            OCR1A = 97 + potValue;
        }
        //90  degrees is at 316
        else if(potValue > 75)
        {
            OCR1A = 316 + potValue;
        }
        //135 degrees is at 425
        else if(potValue > 150)
        {
            OCR1A = 425 + potValue;

        }
        //180 degrees is at 535
        //Rounding highest potentiometer values to 180 degrees
        else if(potValue > 225)
        {
            OCR1A = 535;
        }

        OCR1A = 97 + potValue;                      //0 degrees is at 97
        itoa(potValue, buffer, 10);
        usart_tx_string(buffer);
        usart_tx_string("\n");
        _delay_ms(1000);
    }
}
```

```
Solution1 - main.c*
main.c*
usart_recieve                    unsigned char usart_recieve(void)
        _delay_ms(1000);
    }
}

void adc_init(void)
{
    ADMUX = (0<<REFS1)  |                   //Reference Selection Bits
            (1<<REFS0)  |                   //AVcc - external cap at AREF
            (0<<ADLAR)  |                   //ADC Left Adjust Result
            (0<<MUX2)   |                   //Analog Channel Selection bits
            (0<<MUX1)   |                   //PC0 | PCINT8 | ADC0
            (0<<MUX0)   ;

    ADCSRA =    (1<<ADEN)   |               //ADC Enable
                (0<<ADSC)   |               //ADC Start Conversion
                (0<<ADATE)  |               //ADC Auto Trigger Enable
                (0<<ADIF)   |               //ADC Interrupt Flag
                (0<<ADIE)   |               //ADC Interrupt Enable
                (1<<ADPS2)  |               //ADC Prescalar Select Bits
                (0<<ADPS1)  |
                (1<<ADPS0)  ;
}

void usart_init(void)
{
    UBRR0H = UBRRH_VALUE;
    UBRR0L = UBRRL_VALUE;
    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00);         //8 bit data
    UCSR0B = _BV(RXEN0)  | _BV(TXEN0);          //Enable RX and TX
}

void usart_tx_string(char *data)
{
    while ((*data != '\0'))
    {
        while (!(UCSR0A & (1 << UDRE0)));
        UDR0 = *data;
        data++;
    }
}
110 %
```

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



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

6. VIDEO LINKS OF EACH DEMO

N/A

7. GITHUB LINK OF THIS DA

https://github.com/johnduriman/pirahnaplant.git

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

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