CPE301 – SPRING 2022

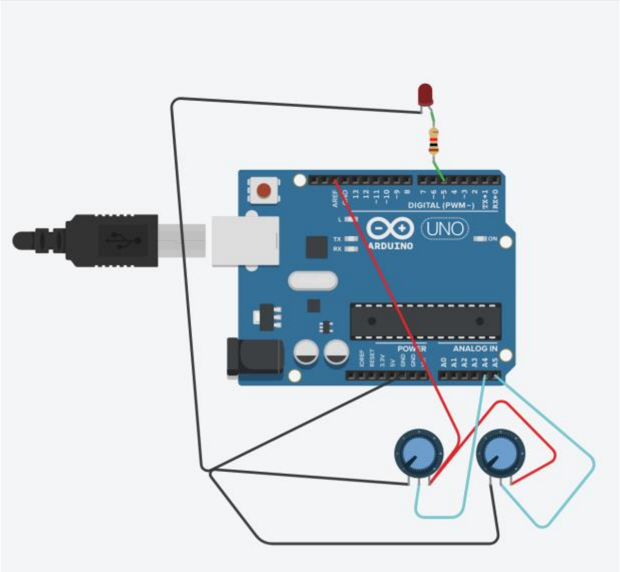
# Design Assignment 3

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

List of Components used: Atmega 328pb and shield
Block diagram with pins used in the Atmega328P:

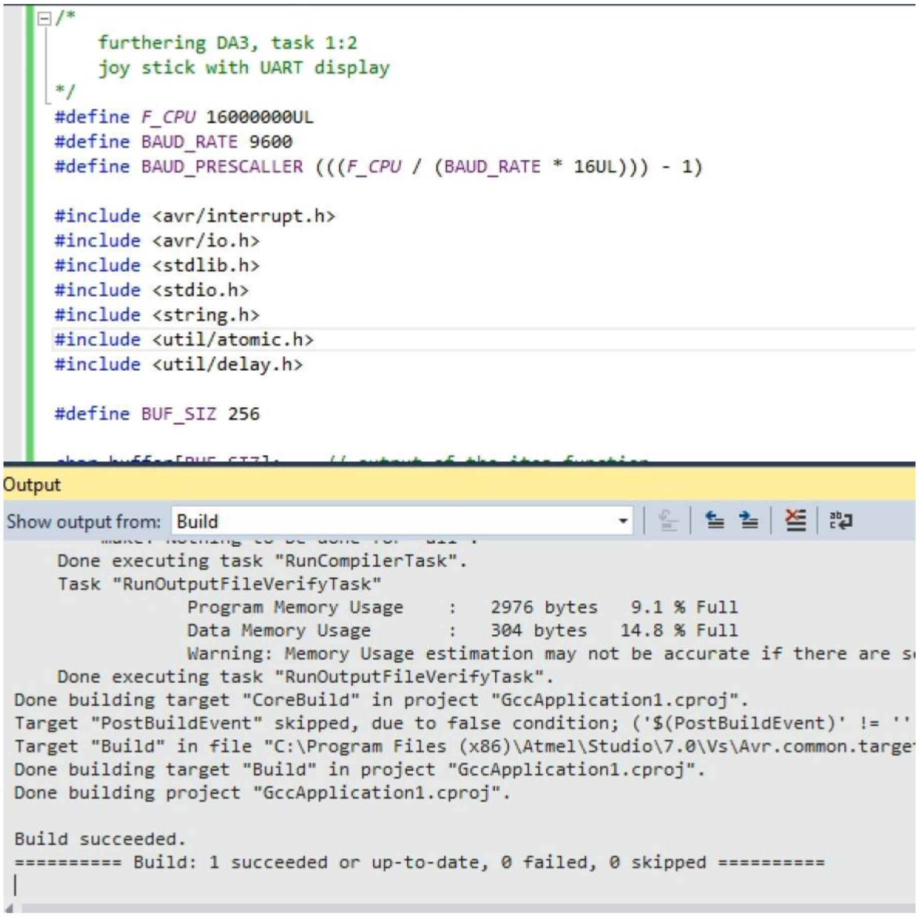
The joystick is 2 potentiometers



Using PC.4 PC.5 [x, y], PD.5 as output for LED

## 2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1:3

Task 1 and 2, successful build



```
/*
    furthering DA3, task 1:2
    joy stick with UART display
*/
#define F_CPU 16000000UL
#define BAUD_RATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUD_RATE * 16UL))) - 1)

#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>

#define BUF_SIZ 256
```

```
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage    :   2976 bytes   9.1 % Full
                Data Memory Usage       :   304 bytes   14.8 % Full
                Warning: Memory Usage estimation may not be accurate if there are s
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "GccApplication1.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != ''
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targe
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 ==========
```

Code for task1 an task2:

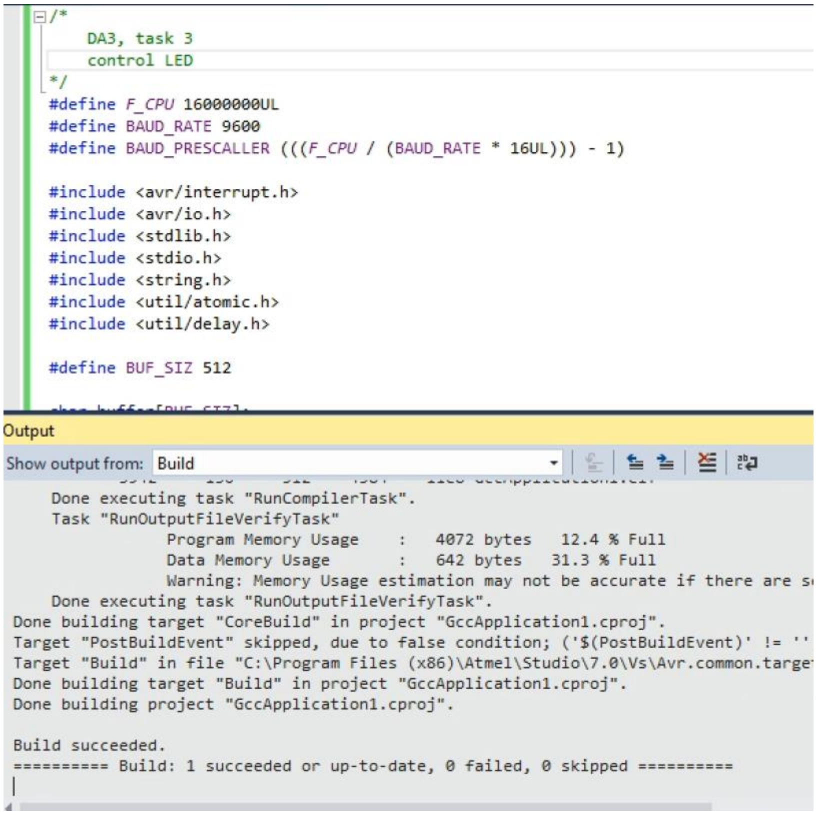
```
/*

   furthering DA3, task 1:2

   joy stick with UART display

*/

#define F_CPU 16000000UL

#define BAUD_RATE 9600

#define BAUD_PRESCALLER (((F_CPU / (BAUD_RATE * 16UL))) - 1)



#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>

#define BUF_SIZ 256

char buffer[BUF_SIZ];    // output of the itoa function

void adc_init(void);
volatile uint16_t read_adc(uint8_t channel);
void USART_init(void);
void USART_send( unsigned char data);
void USART_putstring(char* str_ptr);


int main(void)
{
    volatile uint16_t adc_val_x;      // ADC value, x position
    volatile uint16_t adc_val_y;      // ADC value, x position
    volatile float val_x;             // voltage on x position
    volatile float val_y;             // voltage on y position
    memset(buffer, '\0', BUF_SIZ);
    adc_init();
    USART_init();

    while(1)
    {
        adc_val_x = read_adc(4);        // read x position
        val_x = 5.0 * ((float)adc_val_x / 1023.0);    // find the voltage, x position
        adc_val_y = read_adc(5);        // read y position
        val_y = 5.0 * ((float)adc_val_y / 1023.0);    // find the voltage, y position
        snprintf(buffer, BUF_SIZ, "[x,y]:  [%d, %d] ADC  ,  [%0.3f, %0.3f] volts\r\n",
            adc_val_x, adc_val_y, val_x, val_y);
        USART_putstring(buffer);        // send coordinates
        USART_send('\n');
        _delay_ms(1000);                // limits data flow
    }
    return EXIT_FAILURE;
}
```

```
////~~~~



void adc_init(void)

{

    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);                //initial conversion because this one is the slow

}



////~~~~



volatile uint16_t read_adc(uint8_t channel)

{

    ADMUX &= 0xF0;                  // clears channels

    ADMUX |= channel;               // set new channel to read

    ADCSRA |= (1<<ADSC);            // starts a new conversion

    while(ADCSRA & (1<<ADSC)){}    // wait until the conversion is done

    return ADCW;                    // returns the ADC value of the chosen channel

}



////~~~~



void USART_init(void)

{

    //UBRR0L = F_CPU/16/BAUD_RATE-1;            // the baud rate is set

    UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8);    // set baud

    UBRR0L = (uint8_t)(BAUD_PRESCALLER);

    UCSR0B = (1 << TXEN0);                      // transmit enabled

    UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);         // async, 8-bit, 1-bit stop

}



////~~~~
```

```
void USART_send( unsigned char data)
{
    while(!(UCSR0A & (1<<UDRE0))){}// wait until ready
    UDR0 = data;
}

////~~~~

void USART_putstring(char* str)
{
    int ii = 0;
    while(str[ii] != '\0')
    {
        USART_send(str[ii]);
        ii++;
    }
}

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

Task 3, successful build



```
/*
    DA3, task 3
    control LED
*/
#define F_CPU 16000000UL
#define BAUD_RATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUD_RATE * 16UL))) - 1)

#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>

#define BUF_SIZ 512

char buffer[BUF_SIZ];
```

```
    Done executing task "RunCompilerTask".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage    :   4072 bytes   12.4 % Full
                Data Memory Usage       :   642 bytes   31.3 % Full
                Warning: Memory Usage estimation may not be accurate if there are s
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "GccApplication1.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != ''
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targe
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 ==========
```

Code for task3:

```
/*

   DA3, task 3

   control LED with joystick

*/

#define F_CPU 16000000UL

#define BAUD_RATE 9600

#define BAUD_PRESCALLER (((F_CPU / (BAUD_RATE * 16UL))) - 1)


#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>


#define BUF_SIZ 512
```

```
char buffer[BUF_SIZ];

void adc_init(void);
volatile uint16_t read_adc(uint8_t channel);
void USART_init(void);
void USART_send( unsigned char data);
void USART_putstring(char* str_ptr);
void timer1_update (volatile uint16_t freq_x, volatile uint16_t duty_y);


int main(void)
{
    DDRD = (1 << 5);                  // output LED is on PD.5
    PORTD = 0;                        // turn the LED off
    volatile uint16_t adc_val_x;      // ADC value, x position
    volatile uint16_t adc_val_y;      // ADC value, x position
    memset(buffer, '\0', BUF_SIZ);
    adc_init();
    USART_init();

    while(1)
    {
        adc_val_x = read_adc(4);                // read x position
        adc_val_y = read_adc(5);                // read y position
        USART_putstring("\r\n");
        timer1_update(adc_val_x, adc_val_y);    // sets PWM
    }
    return EXIT_FAILURE;
}


////~~~~


void timer1_update (volatile uint16_t freq_x, volatile uint16_t duty_y)
{
    uint16_t get_x = freq_x;
    uint16_t get_y = duty_y;
```

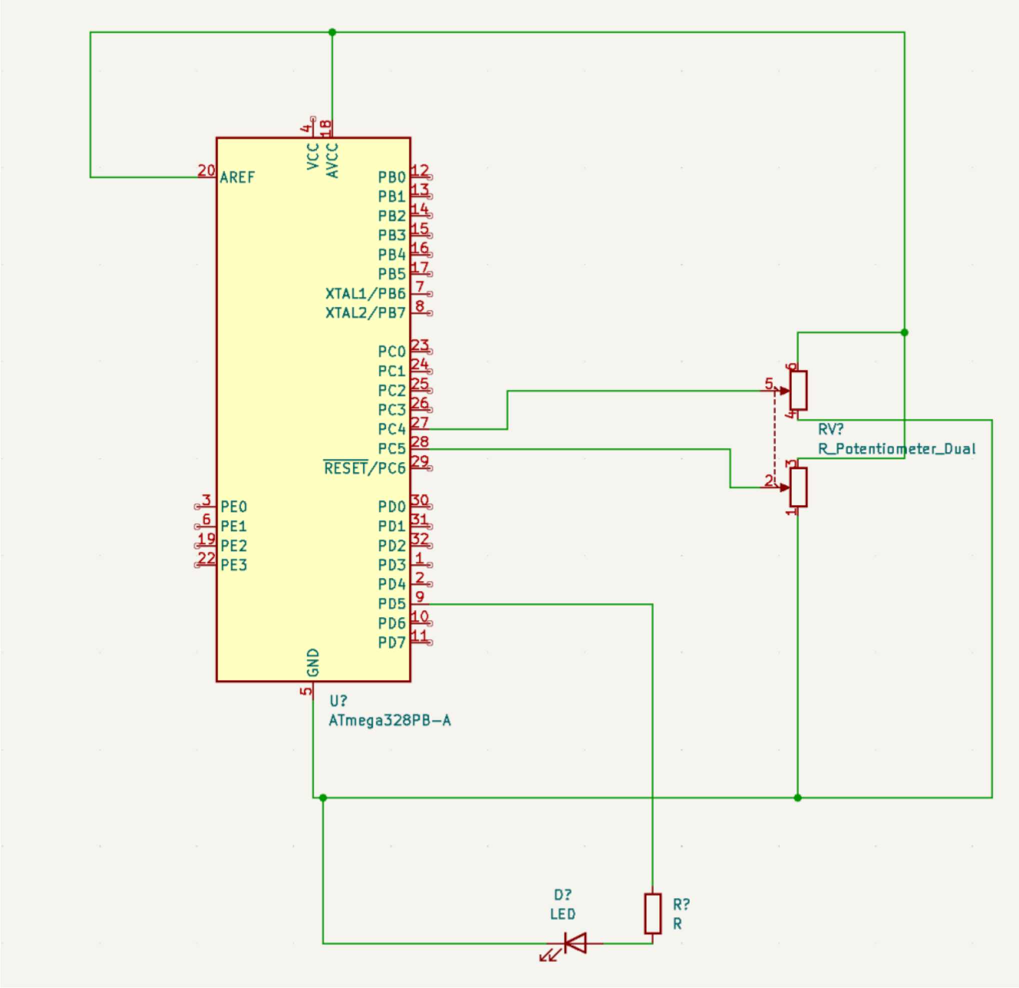
```
// handle corner cases
if (get_y == 0)
{
    PORTD = 0;    // turn LED off, no duty cycle
    return;
}
if (get_y == 1023)
{
    PORTD = (1 << 5);    // turn LED on, all on time
    return;
}
if (get_x < 0) {get_x = 1;}
if (get_x > 1023) {get_x = 1022;}
if (get_y < 1) {get_y = 1;}
if (get_y > 1023) {get_y = 1022;}
snprintf(buffer, BUF_SIZ, "[x,y]:  [%d, %d] ADC\r\n", get_x, get_y);
USART_putstring(buffer);

// contain the frequency to range
float f_frequency = 60 * (get_x / 1023.0);
if (f_frequency > 60.0) {f_frequency = 60.0;}
if (f_frequency < 1.0) {f_frequency = 1.0;}
//
//f_frequency = 30.0;
snprintf(buffer, BUF_SIZ, "f_frequency:  %0.3f\r\n", f_frequency);
USART_putstring(buffer);
//

// OCR1A = (F_CPU / (2 * N * f_osc)) - 1
float f_compare_top = (16000000.0 / ( 256.0 * f_frequency)) - 1.0;
uint16_t compare_top = (uint16_t)f_compare_top;
if (compare_top < 10) {compare_top = 10;}
if (compare_top > 31250) {compare_top = 31250;}
//
//compare_top = 40000;
snprintf(buffer, BUF_SIZ, "f_compare_top:  %0.3f\r\n", f_compare_top);
USART_putstring(buffer);
snprintf(buffer, BUF_SIZ, "compare_top:  %d\r\n", compare_top);
```

```
    USART_putstring(buffer);
    //

    float f_duty_cycle = ((float)get_y / 1023.0) * f_compare_top;
    uint16_t duty_cycle = (uint16_t)f_duty_cycle;
    if (duty_cycle < 5) {duty_cycle = 5;}
    if (duty_cycle > compare_top) {duty_cycle = compare_top;}
    //
    //duty_cycle = 20000;
    snprintf(buffer, BUF_SIZ, "f_duty_cycle:  %0.3f\r\n", f_duty_cycle);
    USART_putstring(buffer);
    snprintf(buffer, BUF_SIZ, "duty_cycle:  %d\r\n", duty_cycle);
    USART_putstring(buffer);
    //

    PORTD = (1 << 5);      // turn LED on
    OCR1A = compare_top;
    TIMSK1 = (1 << OCIE1A);                   // compareA flag
    TCNT1 = 0;
    TCCR1B |=
        (1 << WGM12) |                        // ctc mode
        (1 << CS12)  |                        // prescale 256
        (0 << CS11)  |
        (0 << CS10);                          // time starts
    while (TCNT1 < duty_cycle) {}             // wait on time
    PORTD = 0; // turn LED off
    while ((TIFR1 & (1 << OCF1A)) == 0) {}    // wait off time
    TIFR1 |= (1 << OCF1A);                    // clear flag
    TCCR1B = 0;                               // stop time
}


////~~~~


void adc_init(void)
{
    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

}



////~~~~



volatile uint16_t read_adc(uint8_t channel)

{

  ADMUX &= 0xF0;            // clears channels

  ADMUX |= channel;         // set new channel to read

  ADCSRA |= (1<<ADSC);        // starts a new conversion

  while(ADCSRA & (1<<ADSC)){}   // wait until the conversion is done

  return ADCW;             // returns the ADC value of the chosen channel

}



////~~~~



void USART_init(void)

{

  //UBRR0L = F_CPU/16/BAUD_RATE-1;        // the baud rate is set

  UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8);   // set baud

  UBRR0L = (uint8_t)(BAUD_PRESCALLER);

  UCSR0B = (1 << TXEN0);              // transmit enabled

  UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);       // async, 8-bit, 1-bit stop

}



////~~~~



void USART_send( unsigned char data)

{

  while(!(UCSR0A & (1<<UDRE0))){}// wait until ready

  UDR0 = data;

}
```

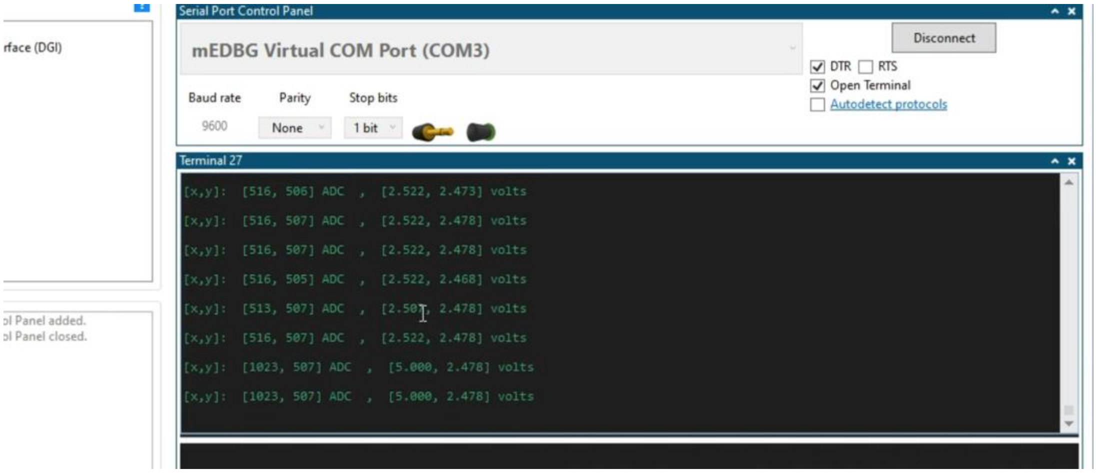
```
////~~~~


void USART_putstring(char* str)
{
    int ii = 0;
    while(str[ii] != '\0')
    {
        USART_send(str[ii]);
        ii++;
    }
}


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

## 3. SCHEMATICS



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

Task 1:2



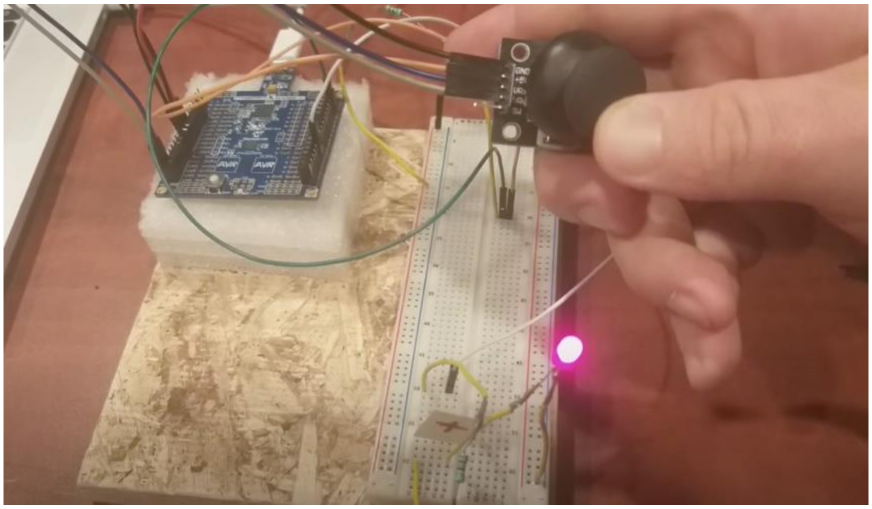
mEDBG Virtual COM Port (COM3)

Baud rate: 9600
Parity: None
Stop bits: 1 bit

☑ DTR ☐ RTS
☑ Open Terminal
☐ Autodetect protocols

```
[x,y]:  [516, 506] ADC  ,  [2.522, 2.473] volts
[x,y]:  [516, 507] ADC  ,  [2.522, 2.478] volts
[x,y]:  [516, 507] ADC  ,  [2.522, 2.478] volts
[x,y]:  [516, 505] ADC  ,  [2.522, 2.468] volts
[x,y]:  [513, 507] ADC  ,  [2.507, 2.478] volts
[x,y]:  [516, 507] ADC  ,  [2.522, 2.478] volts
[x,y]:  [1023, 507] ADC  ,  [5.000, 2.478] volts
[x,y]:  [1023, 507] ADC  ,  [5.000, 2.478] volts
```

Task 3

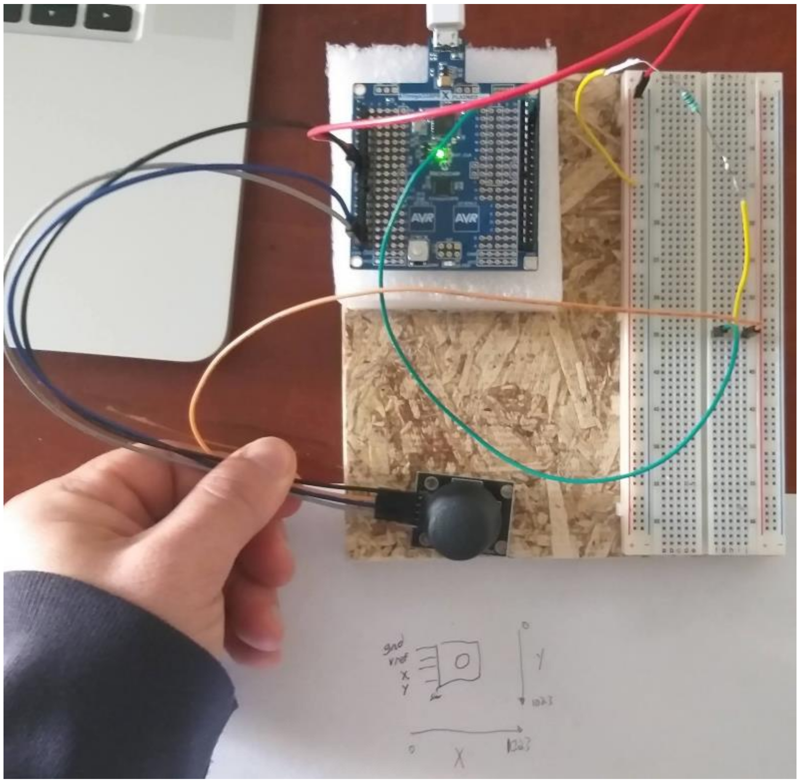


Conversions in progress:

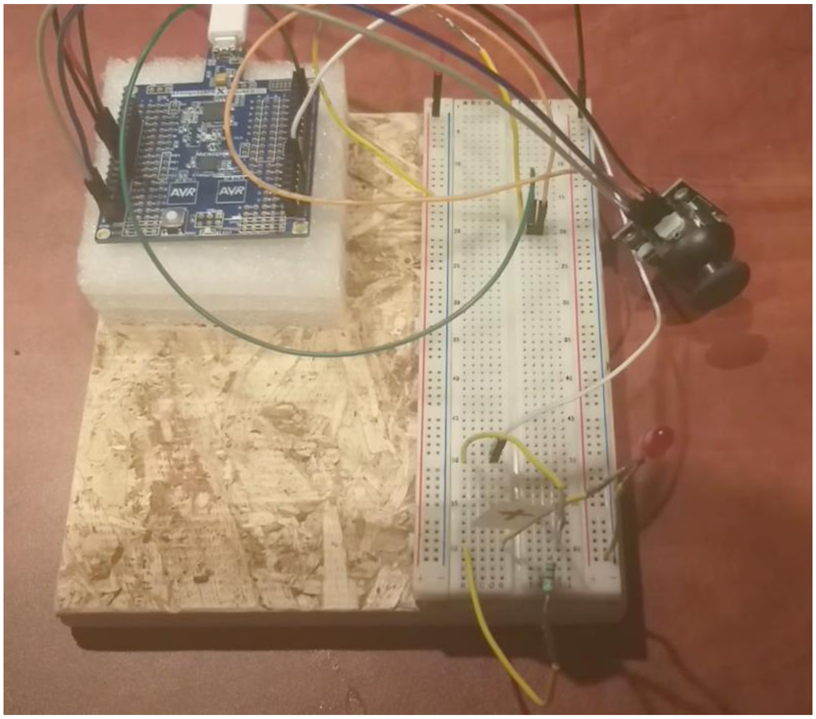
```
[x,y]:  [1023, 508] ADC
f_frequency:  60.000
f_compare_top:  519.833
compare_top:  519
f_duty_cycle:  258.138
duty_cycle:  258

[x,y]:  [509, 1] ADC
f_frequency:  29.853
f_compare_top:  1045.783
compare_top:  1045
f_duty_cycle:  1.022
duty_cycle:  5

[x,y]:  [516, 1023] ADC
f_frequency:  30.264
f_compare_top:  1031.582
compare_top:  1031
f_duty_cycle:  1031.582
duty_cycle:  1031
```

### 5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

The circuit prepared for UART transmission:



The circuit controlling an LED:



## 6. VIDEO LINKS OF EACH DEMO

task1 and 2, UART results:
https://youtu.be/90Kqhe97igs

task3, LED control:
https://youtu.be/CJQ9EV1pl5I

## 7. GITHUB LINK OF THIS DA

https://github.com/davenakasone/cpe301_David_Nakasone/tree/main/Design_Assignments/DA3

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

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