CPE301 – FALL 2019

Design Assignment 3

Student Name: Cody Jones

Student #: 5002863328

Student Email: Jonesc30@unlv.nevada.edu

Primary Github address: https://github.com/Jonesc30/Submission

Directory: Submission

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Atmega328PB

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

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define BAUDRATE 9600

#define BAUD\_PRESCALLER (((F\_CPU / (BAUDRATE \* 16UL))) - 1)

void USART\_init(void); //function to initialize USART

unsigned char USART\_receive(void); //function to receive through USART

void USART\_transmit(unsigned char data); //function to send through USART

void USART\_putstring(char\* StringPtr); //function to scan through the string

float cj = 2.500; //float number chosen for assignment

char String[] = "DA3A"; //string for task

char newline[] = "\n"; //so that I can create a new line

char buffer[5]; //buffer created for the random int number that will be 5 digits

char cj\_buffer[4]; //buffer created for float number chosen above

*uint8\_t* OVF\_COUNT = 0; //initialize the overflow count for interrupt

*uint8\_t* OVF\_LIMIT = 250; //set the limit the count can reach to set 1 sec delay

int main(void)

{

USART\_init(); //initialize USART

TCCR0A = 0x00; //normal operation

TCCR0B |= (1 << CS02); //set prescalar to 256

TCNT0 = 16; //TOP = 256-250 = 16

TIMSK0 |= (1 << TOIE0);

sei(); //enable interrupt

while (1)

{ //sit here waiting for interrupt

}

}

void USART\_init(void)

{

UBRR0H = (*uint8\_t*)(BAUD\_PRESCALLER >> 8);

UBRR0L = (*uint8\_t*)(BAUD\_PRESCALLER);

UCSR0B = (1 << RXEN0) | (1 << TXEN0);

UCSR0C = (3 << UCSZ00);

}

unsigned char USART\_receive(void)

{

while(!(UCSR0A & (1 << RXC0)));

return UDR0;

}

void USART\_transmit(unsigned char data)

{

while(!(UCSR0A & (1 << UDRE0)));

UDR0 = data;

}

void USART\_putstring(char\* StringPtr)

{

while(\*StringPtr != 0x00)

{

USART\_transmit(\*StringPtr);

StringPtr++;

}

}

// C program for implementation of ftoa()

// reverses a string 'str' of length 'len'

void reverse(char \*str, int len)

{

int i=0, j=len-1, temp;

while (i<j)

{

temp = str[i];

str[i] = str[j];

str[j] = temp;

i++; j--;

}

}

// Converts a given integer x to string str[]. d is the number

// of digits required in output. If d is more than the number

// of digits in x, then 0s are added at the beginning.

int intToStr(int x, char str[], int d)

{

int i = 0;

while (x)

{

str[i++] = (x%10) + '0';

x = x/10;

}

// If number of digits required is more, then

// add 0s at the beginning

while (i < d)

str[i++] = '0';

reverse(str, i);

str[i] = '\0';

return i;

}

// Converts a floating point number to string.

void ftoa(float n, char \*res, int afterpoint)

{

// Extract integer part

int ipart = (int)n;

// Extract floating part

float fpart = n - (float)ipart;

// convert integer part to string

int i = intToStr(ipart, res, 0);

// check for display option after point

if (afterpoint != 0)

{

res[i] = '.'; // add dot

// Get the value of fraction part upto given no.

// of points after dot. The third parameter is needed

// to handle cases like 233.007

fpart = fpart \* *pow*(10, afterpoint);

intToStr((int)fpart, res + i + 1, afterpoint);

}

}

ISR (TIMER0\_OVF\_vect)

{

OVF\_COUNT++; //increment the overflow counter

if (OVF\_COUNT == OVF\_LIMIT) //check to see if the limit was reached

{

USART\_putstring(String); //print string to serial monitor (terminal window)

USART\_putstring(newline); //go to next line

*itoa*(*rand*(), buffer, 10); //convert random int number (base 10) to a string

USART\_putstring(buffer); //print out the converted string stored in buffer

USART\_putstring(newline); //go to next line

ftoa(cj, cj\_buffer, 2); //convert float number (2 digits after decimal precision) to a string

USART\_putstring(cj\_buffer); //print out the converted string stored in buffer

USART\_putstring(newline); //go to next line

USART\_putstring(newline); //go to next line

OVF\_COUNT = 0; //reset overflow counter

}

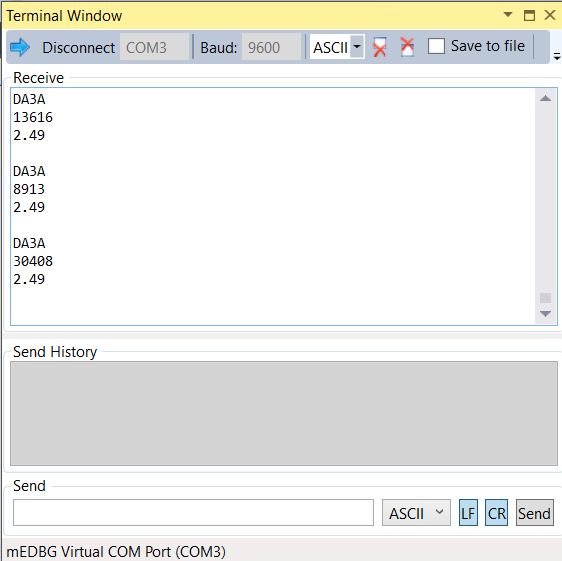
TCNT0 = 16; //reset TOP

}

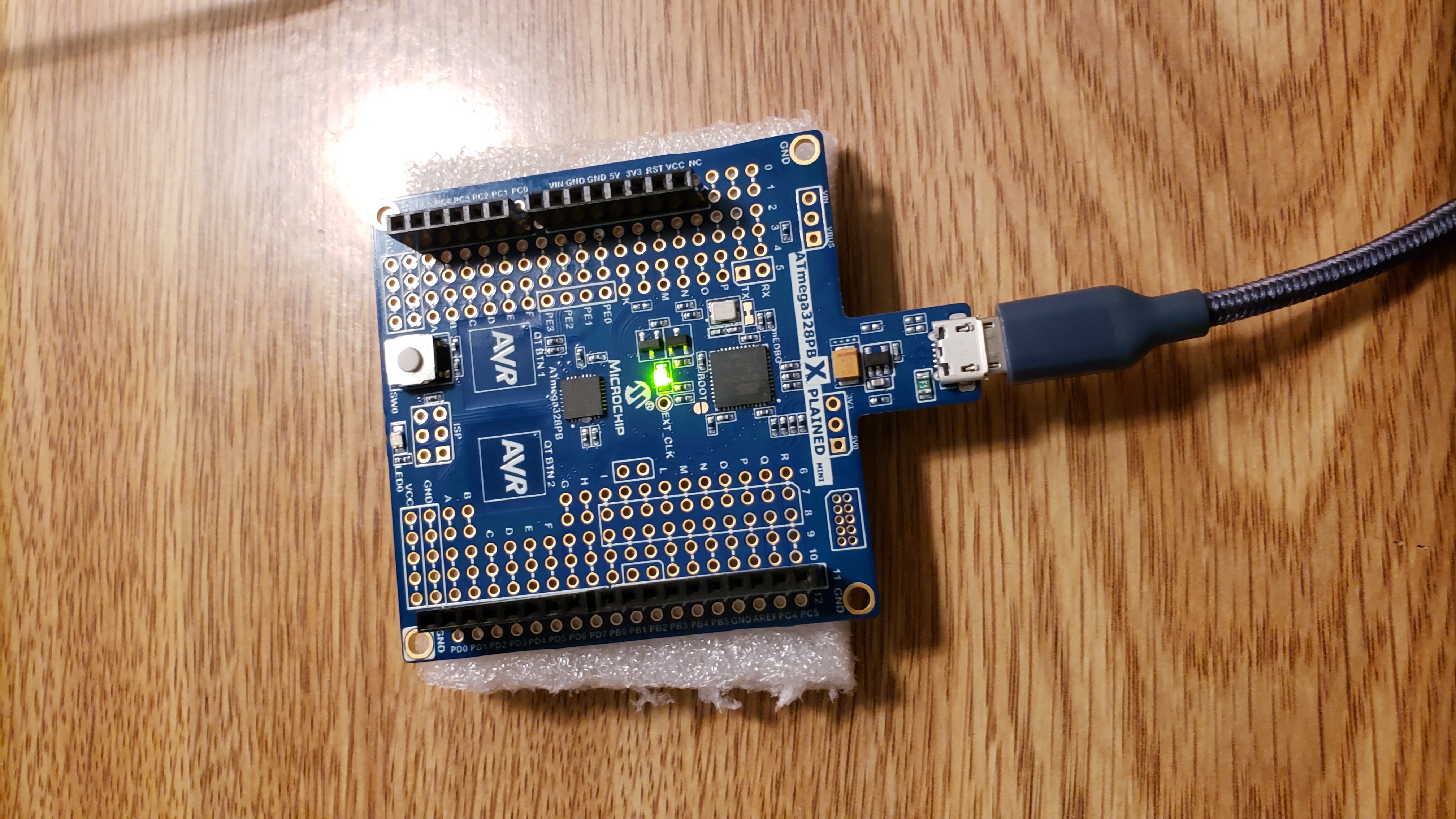
1. **SCHEMATICS**

Use fritzing.org

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



1. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/watch?v=sY73ztCTLSw>

1. **GITHUB LINK OF THIS DA**

<https://github.com/Jonesc30/Submission/tree/master/DesignAssignments>

**Student Academic Misconduct Policy**

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

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

Cody Jones