CPE301 – SPRING 2019

Design Assignment DA3b

Student Name: Daniel Senda
Student #: 5002362633
Student Email: sendad1@unlv.nevada.edu
Primary Github address: https://github.com/dsenda/Smiles
Directory: DA3b

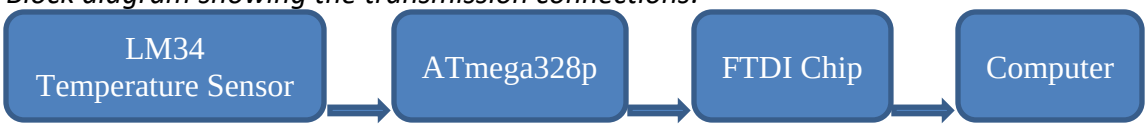
Submit the following for all Labs:

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

List of Components used:
Atmel Studio 7
ATmega328P Xplained Mini
FTDI chip
Breadboard
Jumper wires

Block diagram showing the transmission connections:



2. INITIAL CODE OF TASK DA3b

Not applicable, there is no initial code.

3. DEVELOPED CODE OF TASK DA3b

The following is the code that accomplishes the desired tasks that are required.

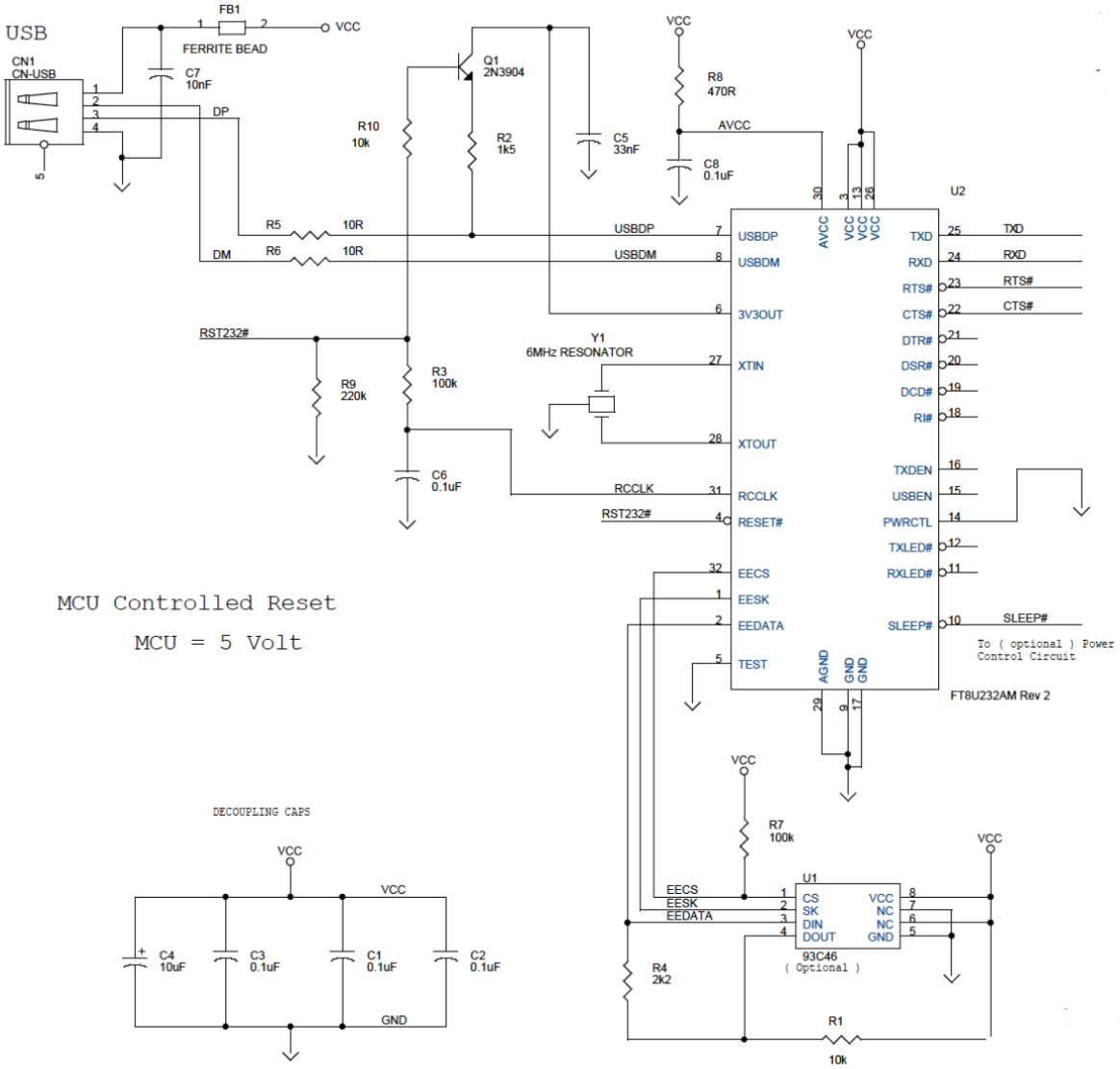
```
// c_code_temperature_on_serial2usb.c
// Daniel Senda
// Monitors LM34 to display temperature on serial terminal every 1s.
// Uses a timer with interrupt for the 1s delay.

#define       F_CPU 16000000UL     // Needed Definitions.
#define       BAUD_RATE 9600
#include      <avr/io.h>           // Needed Libraries.
#include      <util/delay.h>
#include      <avr/interrupt.h>

volatile int O_F = 0;                    // Used for overflow count.
int num = 0;                             // Used for while loops.

void usart_init ();
void usart_send (unsigned char ch);
void initiate_timer()
{
       TCCR0A = 0b10;              // Sets CTC mode.
       TCCR0B = 0b00000101;        // Sets 1024 pre-scaler.
       OCR0A = 255;                // Sets output compare register A to 255.
       TIMSK0 |= (1 << OCIE0A);    // Enables output compare match interrupt.
       sei();                      // Enables global interrupts.
       TCNT0 = 0;                  // Resets timer.
}
ISR(TIMER0_COMPA_vect)             // Keeps count of overflows.
{
       O_F++;                      // Increments overflow count.
}
```

```c
int main (void)
{
	usart_init ();
	// Setup and enable ADC
	ADMUX = (0<<REFS1)|		// Reference Selection Bits.
	(1<<REFS0)|			// AVcc - external cap at AREF.
	(0<<ADLAR)|			// ADC Left Adjust Result.
	(1<<MUX2)|			// Analog Channel Selection Bits.
	(0<<MUX1)|			// ADC4 (PC5).
	(1<<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 Pre-scaler Select Bits.
	(0<<ADPS1)|
	(1<<ADPS0);
	initiate_timer();		// Function that initiates timer.
	while (1)
	{
		ADCSRA|=(1<<ADSC);			// Starts conversion.
		while((ADCSRA&(1<<ADIF))==0);		// Waits for conversion to finish.
		ADCSRA |= (1<<ADIF);
		num = 0;				// Resets num count.
		TCNT0 = 0;				// Resets timer.
		while (num <= 0)
		{
			if (O_F >= 61)
			{
				int a = ADCL;
				a = a | (ADCH<<8);
				a = (a/1024.0) * 5000/10;
				usart_send((a/100)+'0');
				a = a % 100;
				usart_send((a/10)+'0');
				a = a % 10;
				usart_send((a)+'0');
				usart_send('\r');
				num++;			// Increments num to exit while loop.
				O_F = 0;		// Resets overflow count.
			}
		}
	}
	return 0;
}

void usart_init (void)			// Initializes USART.
{
	UCSR0B = (1<<TXEN0);
	UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);
	UBRR0L = F_CPU/16/BAUD_RATE-1;
}

void usart_send (unsigned char ch)	// Transmits characters to computer.
{
	while (! (UCSR0A & (1<<UDRE0)));	// Waits until UDR0 is empty.
```

```
        UDR0 = ch;                          // Transmit ch.
}

void usart_print(char* str)                 // Prints characters on computer.
{
        int i = 0;
        while(str[i] != 0)
        {
                usart_send(str[i]);
        }
}
```

4. SCHEMATICS

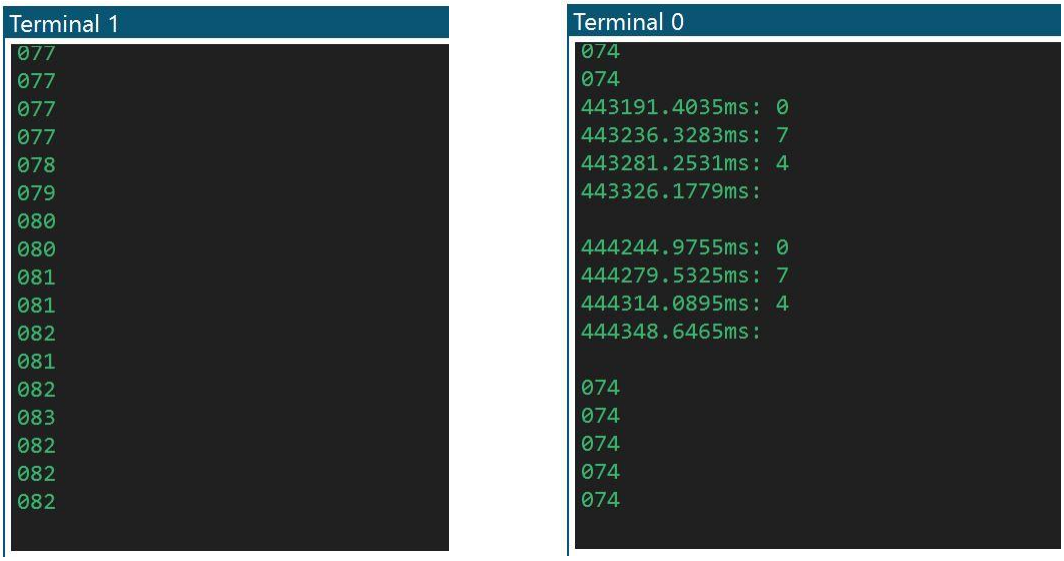
The following schematic from the slides describe the connections that are made relatively well.



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

TASK 1

The following is a screenshot of the output which demonstrates that it is operating properly.

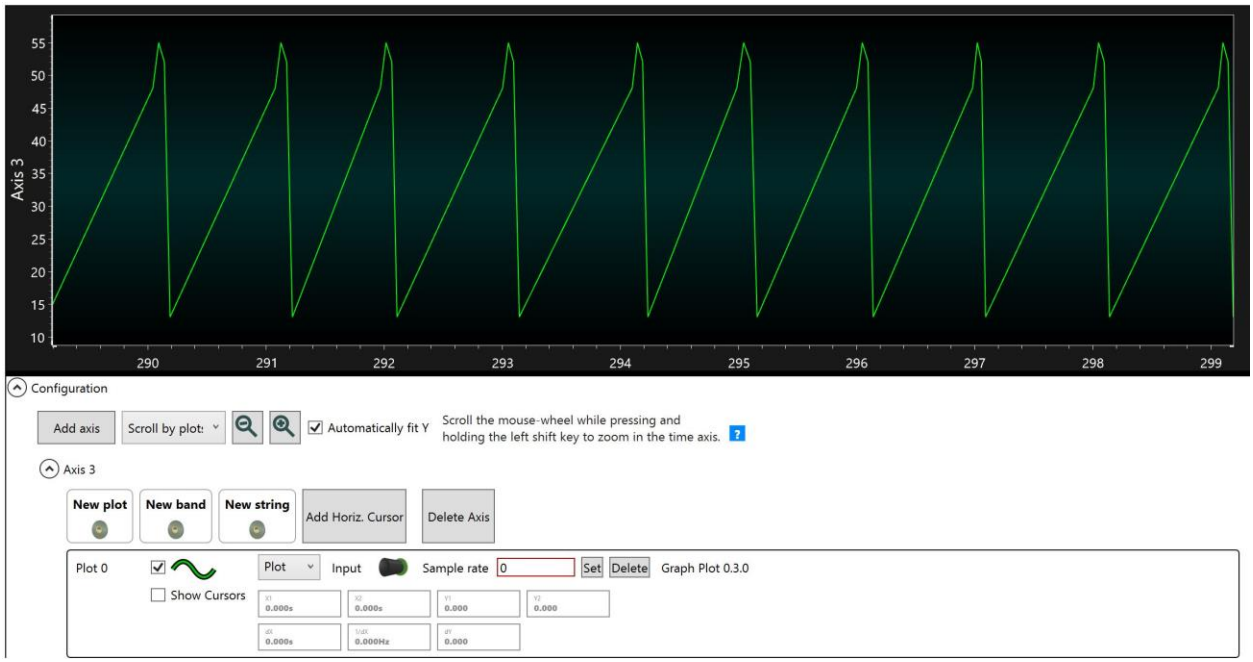


The picture on the left shows how the temperature rose when the student held the temperature sensor with his fingers.

As can be seen from the timestamp, the time between each temperature value print-out is almost exactly 1s. (Picture to the right)

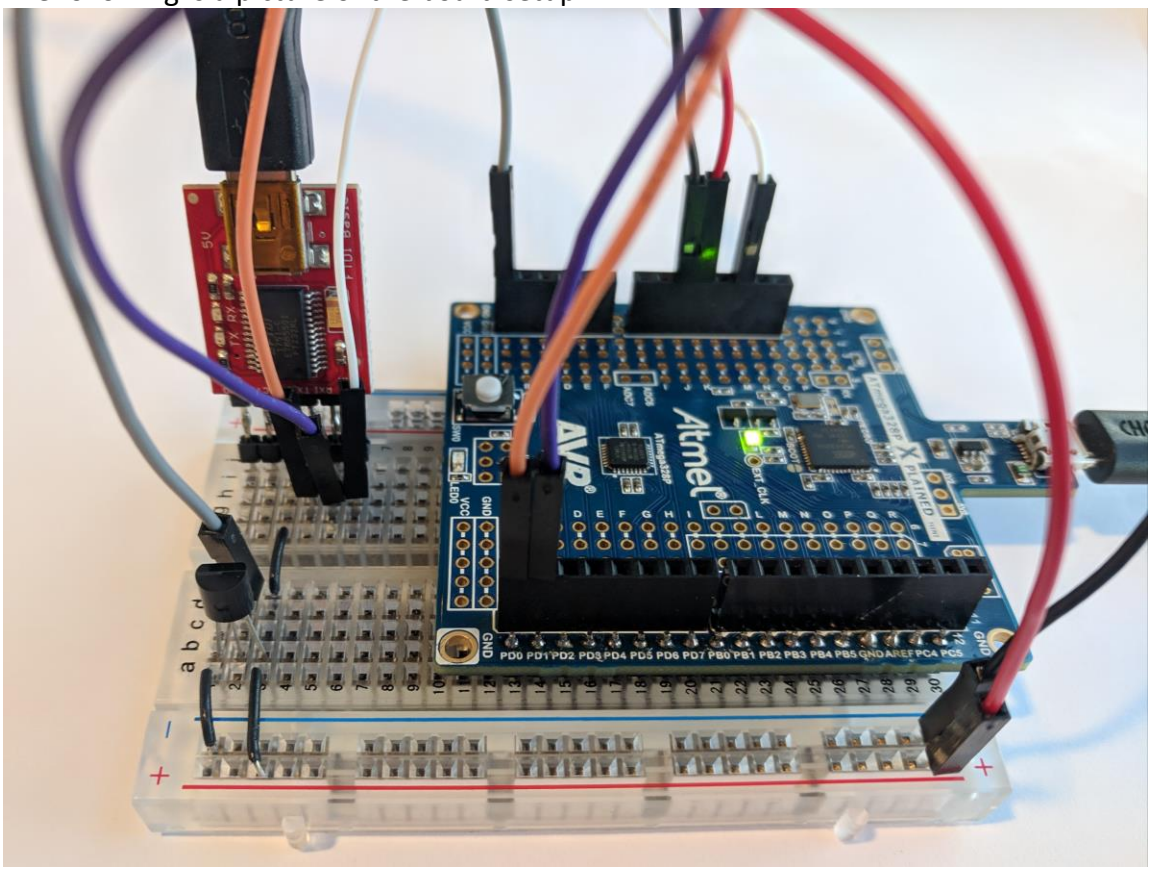
TASK 2

The following is the waveform output (from the ATMEL Studio Data Visualizer) that shows the values in time (outputs data every 1s).



6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

The following is a picture of the board setup:



7. VIDEO LINKS OF EACH DEMO

https://youtu.be/thshZi8K3ao

8. GITHUB LINK OF THIS DA

https://github.com/dsenda/Smiles/tree/master/DA3b

Student Academic Misconduct Policy

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

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

Daniel Senda