CPE301 – SPRING 2019

Design Assignment 3, Part B

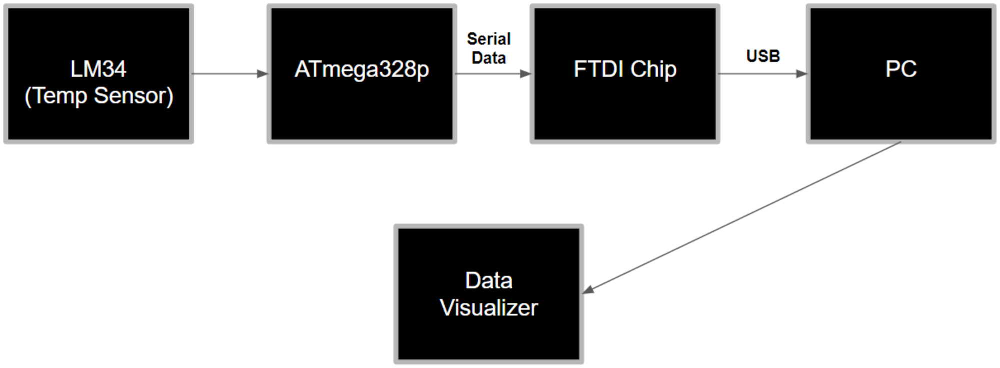
Student Name: James Skelly
Student #: 2000945485
Student Email: skellj1@unlv.nevada.edu
Primary Github address: https://github.com/skellj1/submission_da
Directory: skellj1/submission_da

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

Components used include FTDI chip and board for UART communication, Atmel Studio 7, Xplained Mini board, Atmel Data Visualizer, jumper wires for connections from mini to FTDI chip and board, LM34 temperature sensor, breadboard, iphone for recording.



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

```
/*
 James Skelly, DA3B C Code
 */

#define F_CPU 16000000UL    // define the frequency of the cpu to 16 MEG
#define BAUD_RATE 9600             // sets Baud rate, the rate of bit transfer

// Include the necessary headers
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

int overflow; // initialize overflow variable for delay

// void function prototypes
void usart_init ();
void usart_send (unsigned char ch);

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)|     // ADC5 (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 Prescaler Select Bits
   (0<<ADPS1)|         // ADC5 (PC5)
   (1<<ADPS0);

       // Setup and enable timer0 //
       TCCR0B |= (1<<CS02) | (1<<CS00);   // set prescaler to 1024
       TIMSK0 |= (1<<TOIE0);              // interrupt enabled on compare match A
       sei();                      // enable interrupts
       TCNT0 = 0;                  // resets timer0


       while (1)
       {
              ADCSRA|=(1<<ADSC);   //start conversion
                     while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish

                     ADCSRA |= (1<<ADIF);

       if (overflow == 61)         // generates 1 second delay using interrupt
              {                                  // code to convert output to Fahrenheit
```

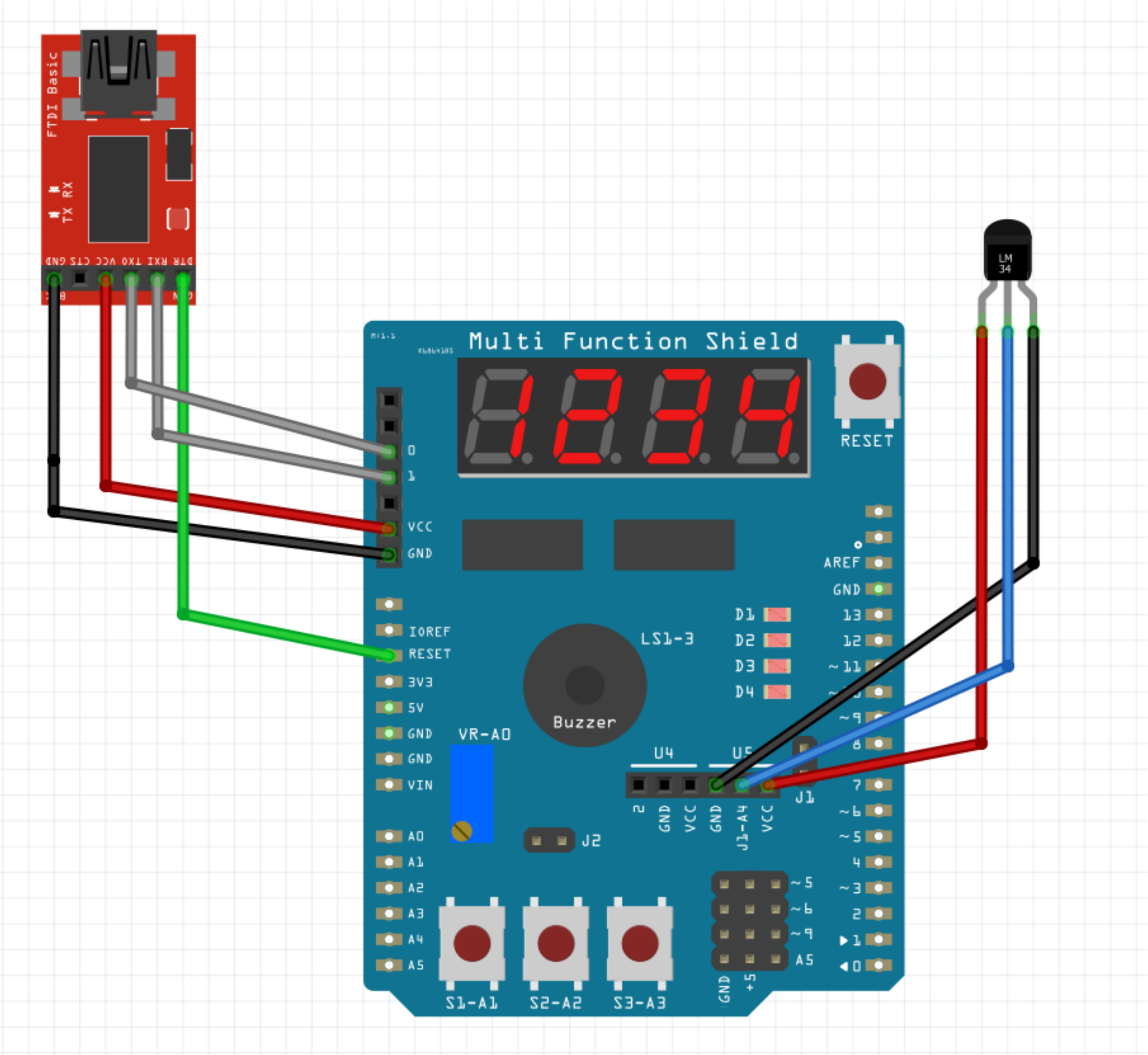
```
                     int a = ADCL;
                     a = a | (ADCH<<8);
                     a = (a/1024.0) * 5000/10;
                     usart_send((a/100)+'0');
                     a = a % 100;
                     usart_send((a/10)+'0');
                     a = a % 10;
                     usart_send((a)+'0');
                     usart_send('\r');
                     overflow=0;          // resets the overflow increment value to 0
              }
       }
       return 0;
}

ISR (TIMER0_OVF_vect)              // Interrupt subroutine
{
       while(!(TIFR0 & 0x01) == 0);       // while timer interrupt flag is high...
       {
              TCNT0 = 0;                  // ... reset timer0
              TIFR0 = 1;                  // ... reset the interrupt flag
              overflow++;                 // ... increment overflow variable
       }
}

void usart_init (void)             //     function to initialize the USART
{
       UCSR0B = (1<<TXEN0);
       UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);
       UBRR0L = F_CPU/16/BAUD_RATE-1;
}

void usart_send (unsigned char ch) // function to transit characters to PC
{
       while (! (UCSR0A & (1<<UDRE0)));   //wait until UDR0 is empty
       UDR0 = ch;                                                   //transmit ch
}

void usart_print(char* str)               // function to print out characters on PC
{
       int i = 0;
       while(str[i] != 0)
              usart_send(str[i]);
}
```

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

Not Applicable for this assignment.

4. SCHEMATICS

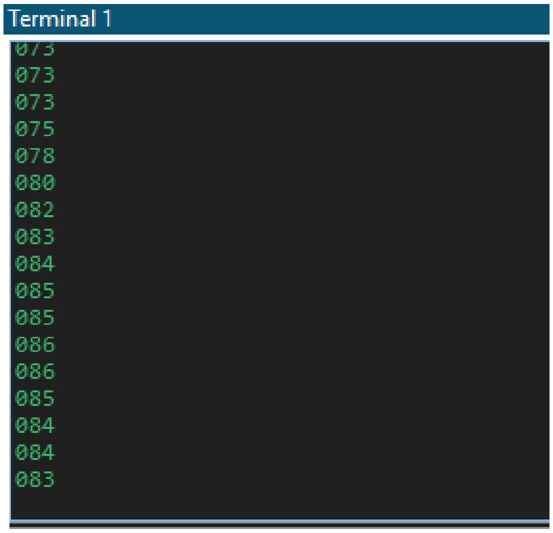
Below is the schematic of the multifunction shield with connections to the FTDI chip used for the UART communication in this assignment, and to the LM34 temperature sensor. The actual shield contains the same pins that were used for this assignment, but the shield itself was not used. In the **Board Setup** section, the photo of my breadboard setup shows that the Xplained Mini headers were routed directly to the FTDI board and to the LM34 with no connections to the shield.



Notice that the LM34 is connected backwards. This chip in the datasheet has a bottom view pinout diagram, so the chip needs to be connected opposite of the way we would assume it should be connected. The blue wire is Vout of the LM34. The red wire is power, and the black wire is ground.

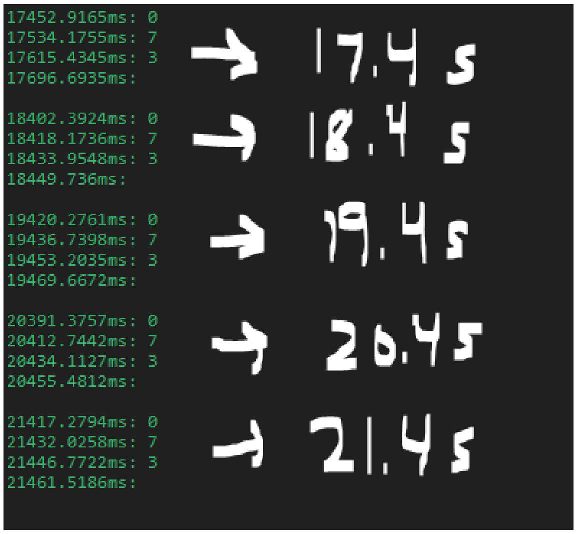
5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1 Output



Terminal 1

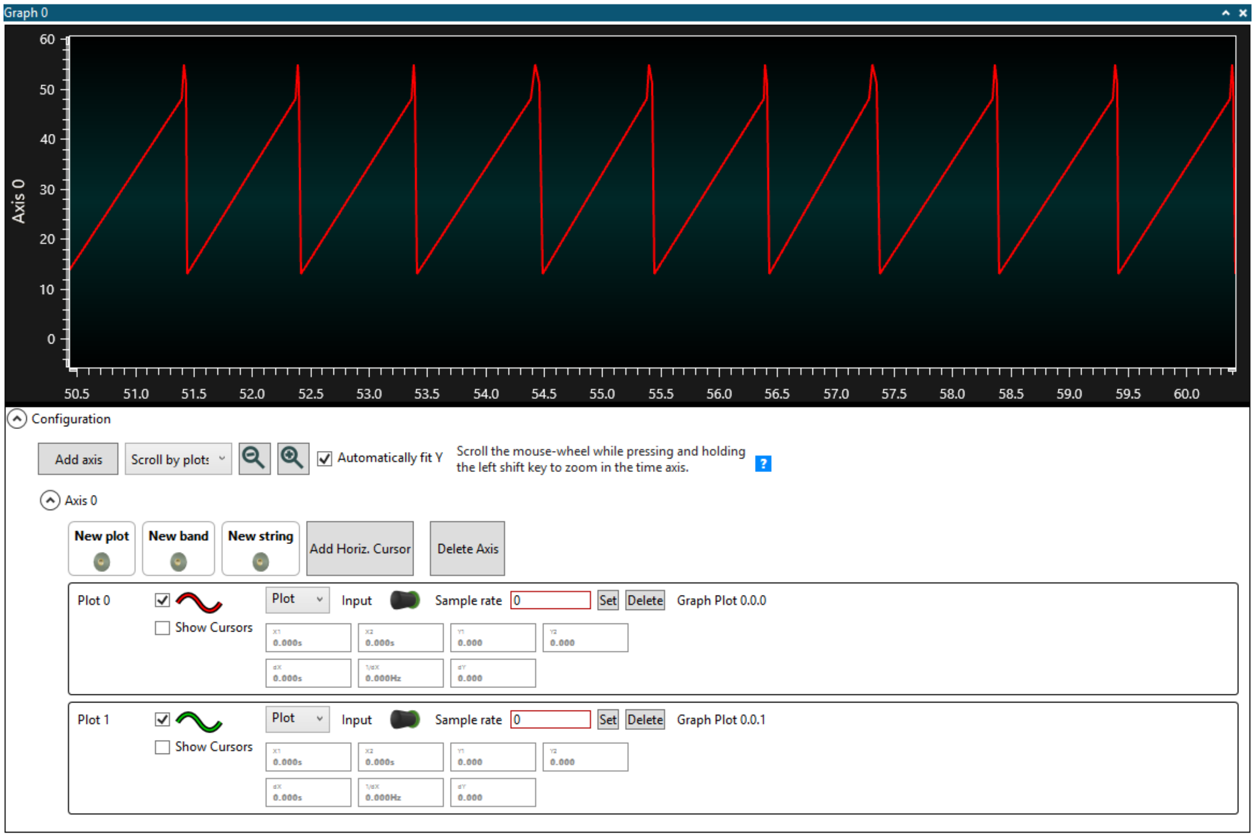
```
073
073
073
075
078
080
082
083
084
085
085
086
086
085
084
084
083
```



```
17452.9165ms: 0
17534.1755ms: 7
17615.4345ms: 3
17696.6935ms:

18402.3924ms: 0
18418.1736ms: 7
18433.9548ms: 3
18449.736ms:

19420.2761ms: 0
19436.7398ms: 7
19453.2035ms: 3
19469.6672ms:

20391.3757ms: 0
20412.7442ms: 7
20434.1127ms: 3
20455.4812ms:

21417.2794ms: 0
21432.0258ms: 7
21446.7722ms: 3
21461.5186ms:
```

Temperature rises on the LM34 when a person squeezes it with their finger, adding body head.

Task 2 Output



☑ Automatically fit Y

Scroll the mouse-wheel while pressing and holding the left shift key to zoom in the time axis.

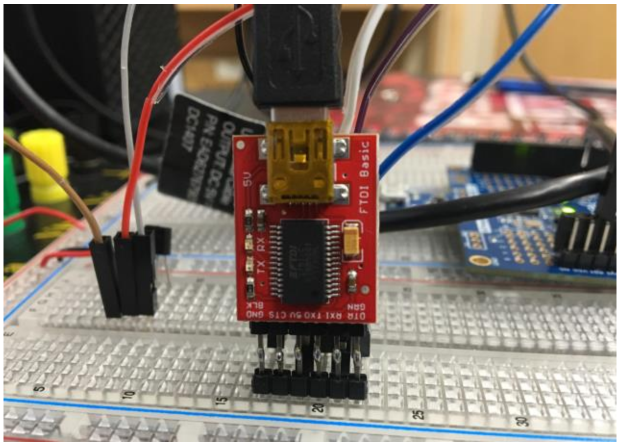
Plot 0 ☑ □ Show Cursors Plot Input Sample rate 0 Graph Plot 0.0.0

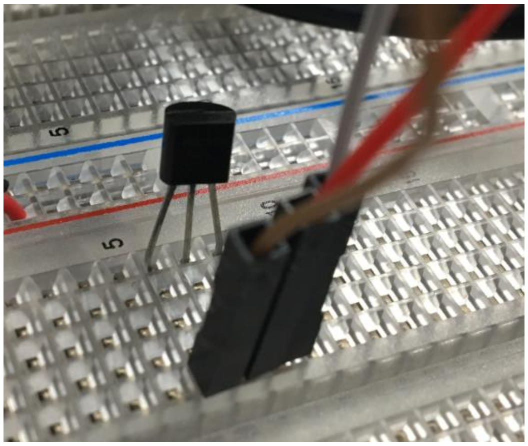
X1 0.000s X2 0.000s Y1 0.000 Y2 0.000
dX 0.000s 1/dX 0.000Hz dY 0.000

Plot 1 ☑ □ Show Cursors Plot Input Sample rate 0 Graph Plot 0.0.1

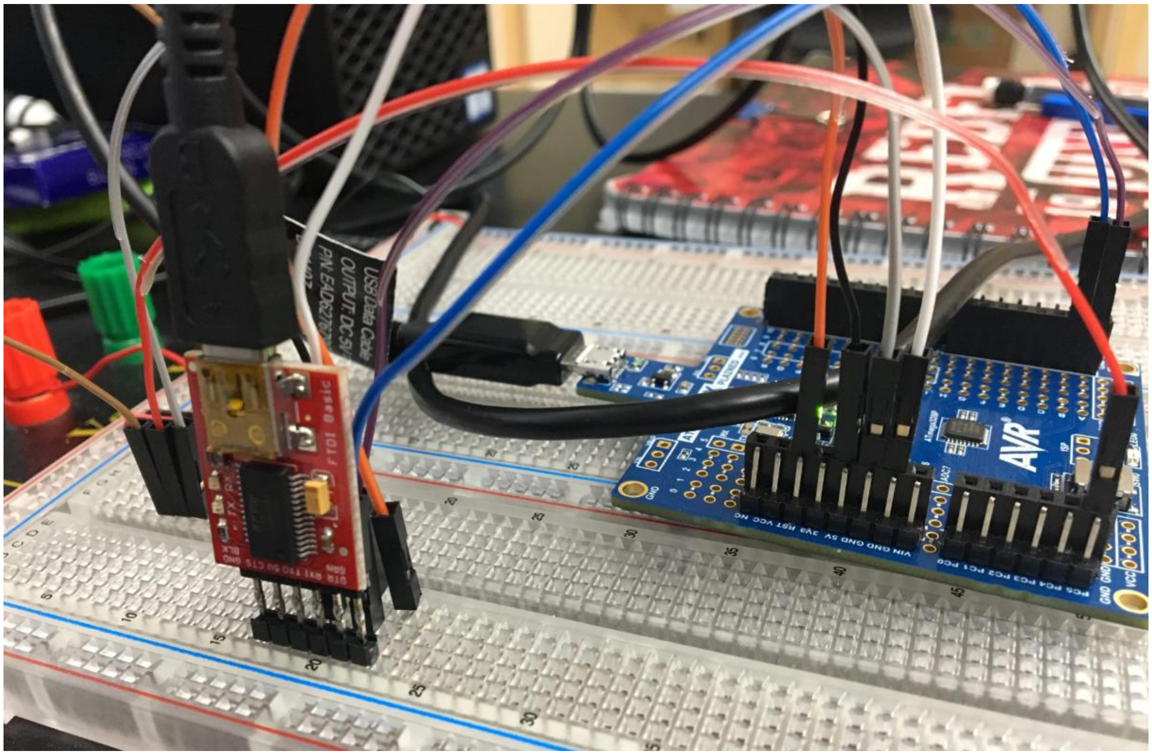
X1 0.000s X2 0.000s Y1 0.000 Y2 0.000
dX 0.000s 1/dX 0.000Hz dY 0.000

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)





Above, we see the board setup of the FTDI chip and the LM34. Below, we see the connections through jumper cable from the Xplained Mini to both the FTDI and LM34. The FTDI board came with attached female headers, so two sets of male headers were soldered together to form an extended male header with leads that would connect nicely to the breadboard.



The shield was not used in the assignment. The footprint for the LM34 is backwards on the shield. To avoid confusion and to avoid frying the LM34, the shield was excluded, and the LM34 was simply connected to the breadboard externally.

7. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=F8dKqkSNSvs

8. GITHUB LINK OF THIS DA

https://github.com/skellj1/submission_da

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

“*This assignment submission is my own, original work*”.
James W. Skelly