Trace Stewart

CPE301 – SPRING 2018

Design Assignment X

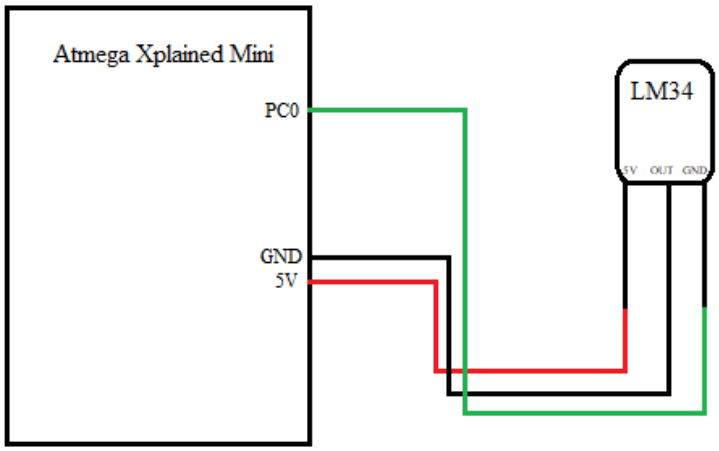
DO NOT REMOVE THIS PAGE DURING SUBMISSION:

The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E		
4.	SCHEMATICS		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
5.	SCREENSHOT OF EACH DEMO		
6.	VIDEO LINKS OF EACH DEMO		
7.	GOOGLECODE LINK OF THE DA		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

The components used for the assignment were the Atemga Xplained mini and the LM34 Temperature Sensor along with a micro usb cable to connect the mini to the computer.



2. INITIAL/DEVELOPED CODE OF TASK 1

```
/*
 * DA3.c
 *
 * Created: 3/18/2018 12:33:30 PM
 * Author : trace
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000L
#include <util/delay.h>
#include <stdlib.h>
#define BAUD  9600

volatile int ovrflw; // global variable for keeping track of # of times Timer0 overflows

// functions
void initUART();
void writeChar(unsigned char c);
void writestring(char *c);

int main(void){

	initUART();		// Initialize UART

	// initialize ADC
	DDRC = 0;			// Set PORTC as input for adc
	DIDR0 = 0x1;		// Disable digital input on ADC0 pin
	ADMUX = 0;			// ADC0 (PC.0) used as analog input
	ADMUX |= (1 << REFS0);	// use AVcc as the reference
```

```
	ADMUX |= (1 << ADLAR);      // Right adjust for 8 bit resolution


	ADCSRA = 0x87;              // Enable ADC, system clock, 10000111
	ADCSRB = 0x0;        // Free running mode

		// initialize timer0 with starting value of 0, normal mode with no pre
scaler
	TCNT0 = 0;
	TCCR0A = 0;
	TCCR0B |= 2;

	// enable interrupts
	TIMSK0 |= (1 << TOIE0);           // enable overflow interrupt
	sei();                                  // enable global interrupts


	while (1);

	return 0;
}

void initUART() {
	unsigned int baudrate;

	// Set baud rate:  UBRR = [F_CPU/(16*BAUD)] -1
	baudrate = ((F_CPU/16)/BAUD) - 1;
	UBRR0H = (unsigned char) (baudrate >> 8);
	UBRR0L = (unsigned char) baudrate;

	UCSR0B |= (1 << RXEN0) | (1 << TXEN0);          // Enable receiver and transmitter
	UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Set data frame: 8 data bits, 1 stop
bit, no parity

}

void writeChar(unsigned char c) {
	UDR0 = c;                   // Display character on serial (i.e., PuTTY) terminal
	_delay_ms(10);              // delay for 10 ms between each letter
}

void writestring(char *c){
	unsigned int i = 0;
	while(c[i] != 0)
	writeChar(c[i++]);
}

// this interrupt service routine (ISR) runs whenever an overflow on Timer0 occurs
ISR (TIMER0_OVF_vect) {

	// Variable Declarations
	char output[6];                                         // Output string based on
ADC
	char *label = "Temperature: " + '\0';     // Temperature String
	char *unit = " F" + '\0';                 // Degree String
	unsigned int adcVal;                      // 8-bit ADC value
	float temperature;                                      // Voltage received by ADC
then edited for Temperature
```

```
        if (ovrflw == 7500) {

                ADCSRA |= (1 << ADSC);                  // Start conversion
                while((ADCSRA&(1<<ADIF))==0);     // Wait for conversion to finish

                adcVal = ADCH * 9 / 5;                  // Only need to read the high
value for 8 bit then equation for Fahrenheit
                temperature = adcVal;       // Temperature
                dtostrf(temperature, 4, 1, output);      // Float to char* conversion

                // Print temperature to the terminal using UART
                writestring(label);
                writestring(output);
                writestring(unit);

                // Print end of line
                writeChar('\n');
                writeChar('\r');
                ovrflw = 0;                       // reinitialize ovrflw
}
else
ovrflw++;     // increment ovrflw
}
```

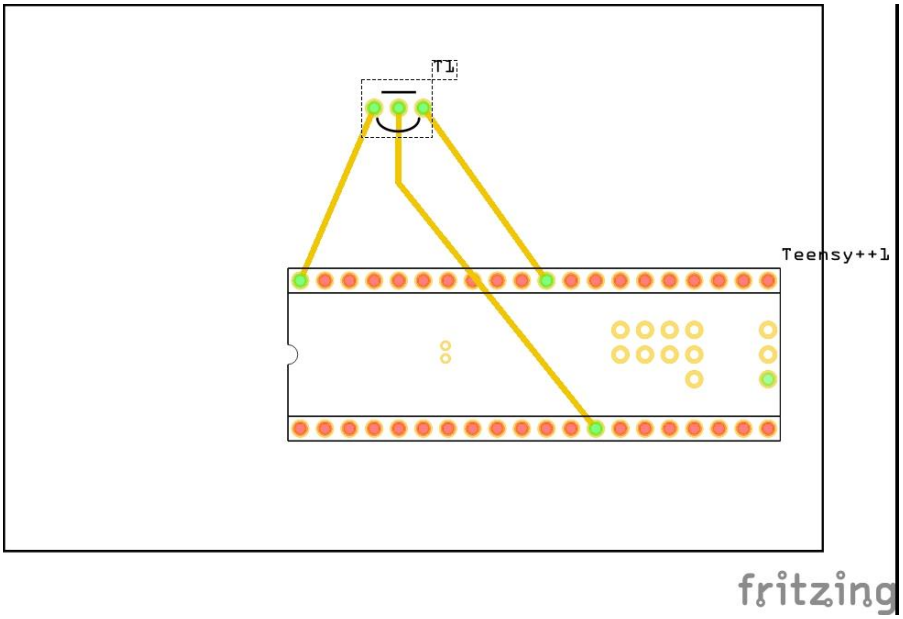
3. TASK 2

```
//This is what I changed in the above program to get the results from the data visualizer.

void writeChar(float c);
…
void writeChar(float c) {
        UDR0 = c;                       // Display character on serial (i.e., PuTTY) terminal
        _delay_ms(10);          // delay for 200 ms
}
/*
void writestring(char *c){
        unsigned int i = 0;
        while(c[i] != 0)
        writeChar(c[i++]);
}
*/
…
        writeChar(temperature);
                // Print temperature to the terminal using UART
                /*writestring(label);
                writestring(output);
                writestring(unit);

                // Print end of line
                writeChar('\n');
                writeChar('\r');*/
```

4. SCHEMATICS



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

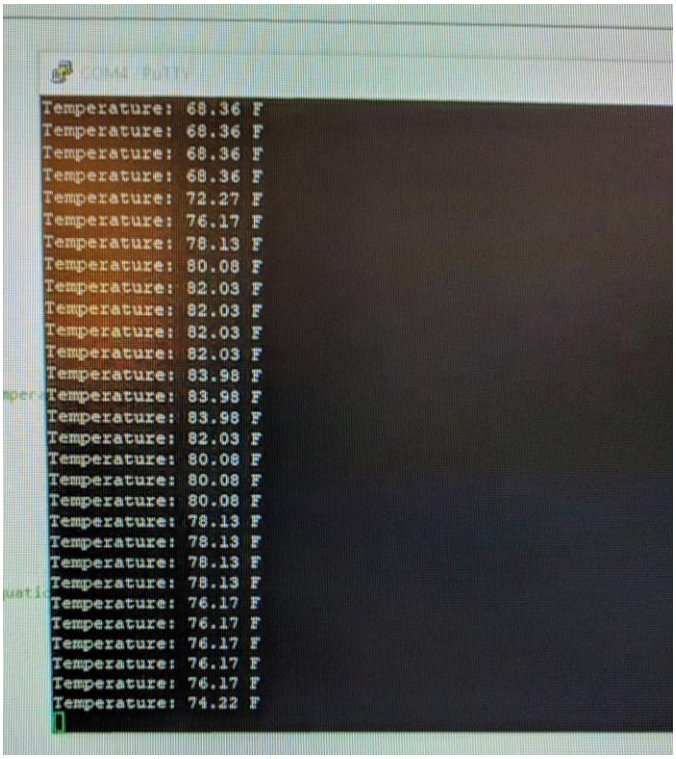


Figure 1: Temperature Increasing after holding onto the sensor (Putty)

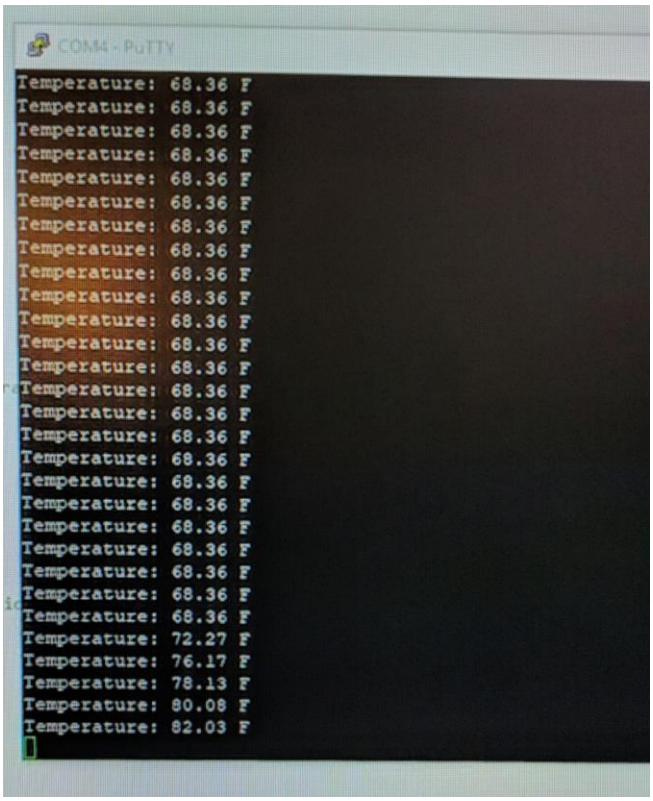


Figure 2: The temperature decreasing after letting go of the sensor (Putty)

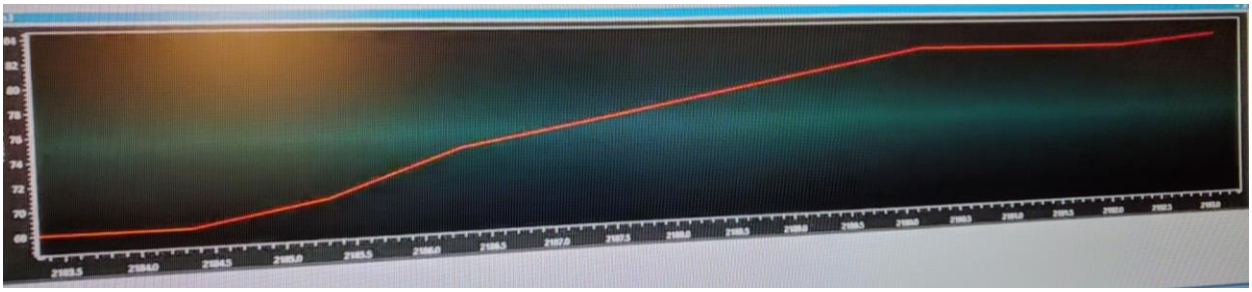


Figure 3:Temperature Increasing after holding onto the sensor (Data Visualizer)

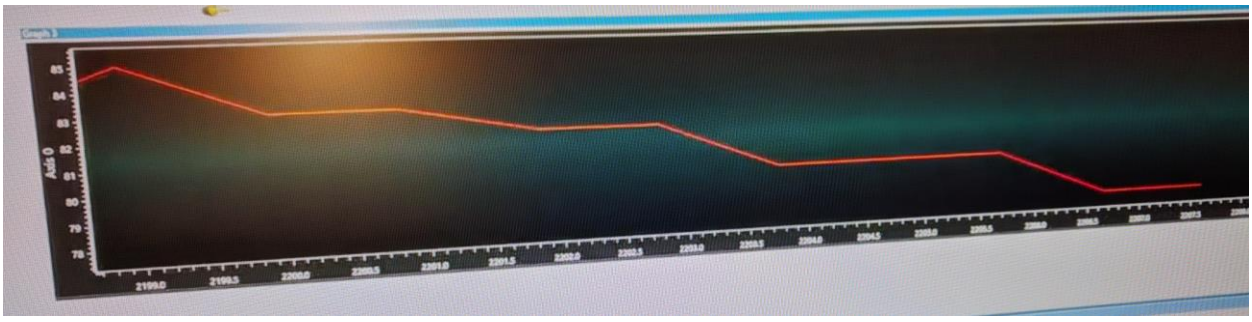


Figure 4: Temperature decreasing after holding onto the sensor (Data Visualizer)

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

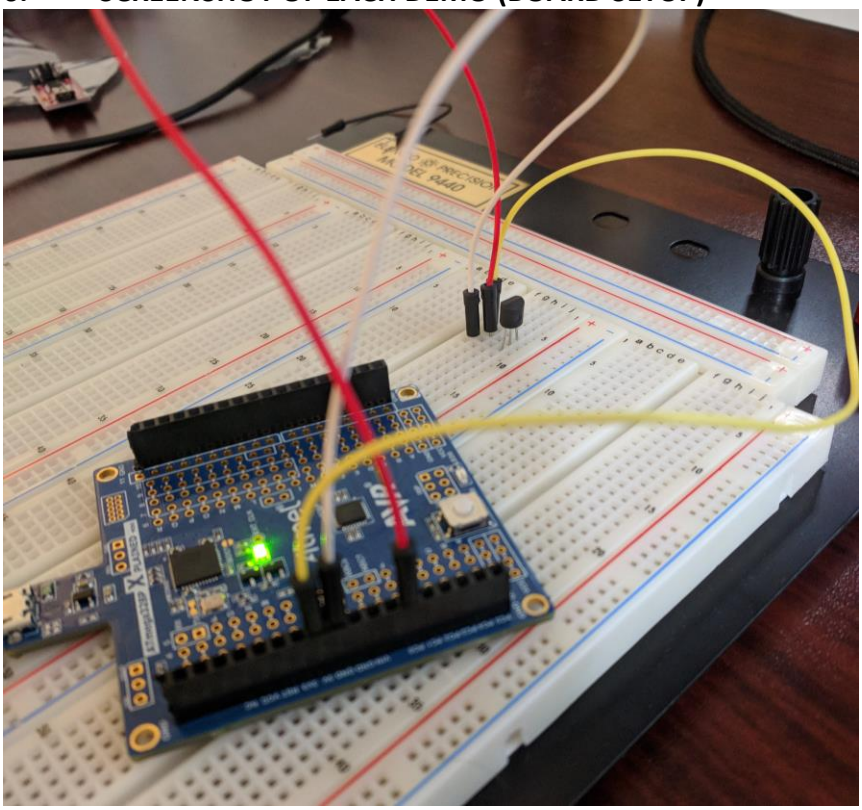


Figure 5: Side View of Setup

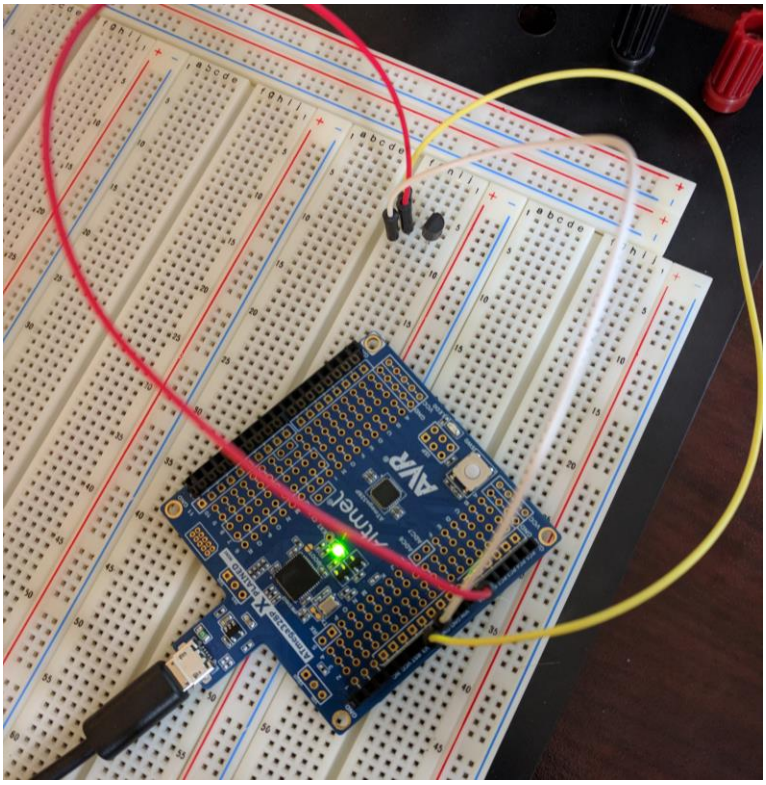


Figure 6: Top View of Setup

7. VIDEO LINKS OF EACH DEMO

Task 1: https://www.youtube.com/watch?v=iblZ6KprCC8
Task 2: https://www.youtube.com/watch?v=Cs3bv4uKRVw

GITHUB LINK OF THIS DA
https://github.com/TraceStewart/epc103gnirps8102vlnu/tree/master/DA3

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

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