Bryan Takemoto

CPE301 – SPRING 2018

Design Assignment 3

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

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

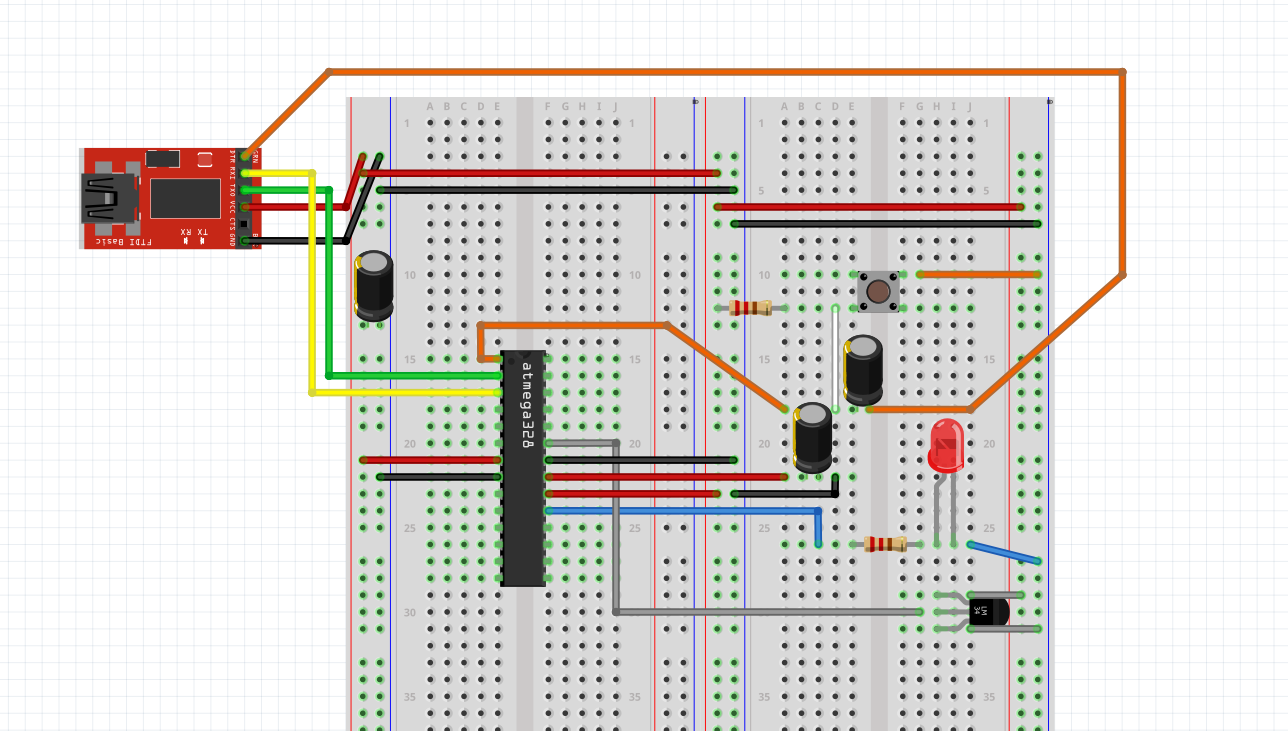
|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND BREADBOARD |  |  |
| 2. | SCHEMATIC |  |  |
| 3. | C CODE TO MEASURE TEMPERATURE (F) |  |  |
| 4. | FLOWCHART OF C CODE |  |  |
| 5. | SCREENSHOTS OF DATA VISUALIZER WAVEFORM |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **COMPONENTS LIST AND BREADBOARD**

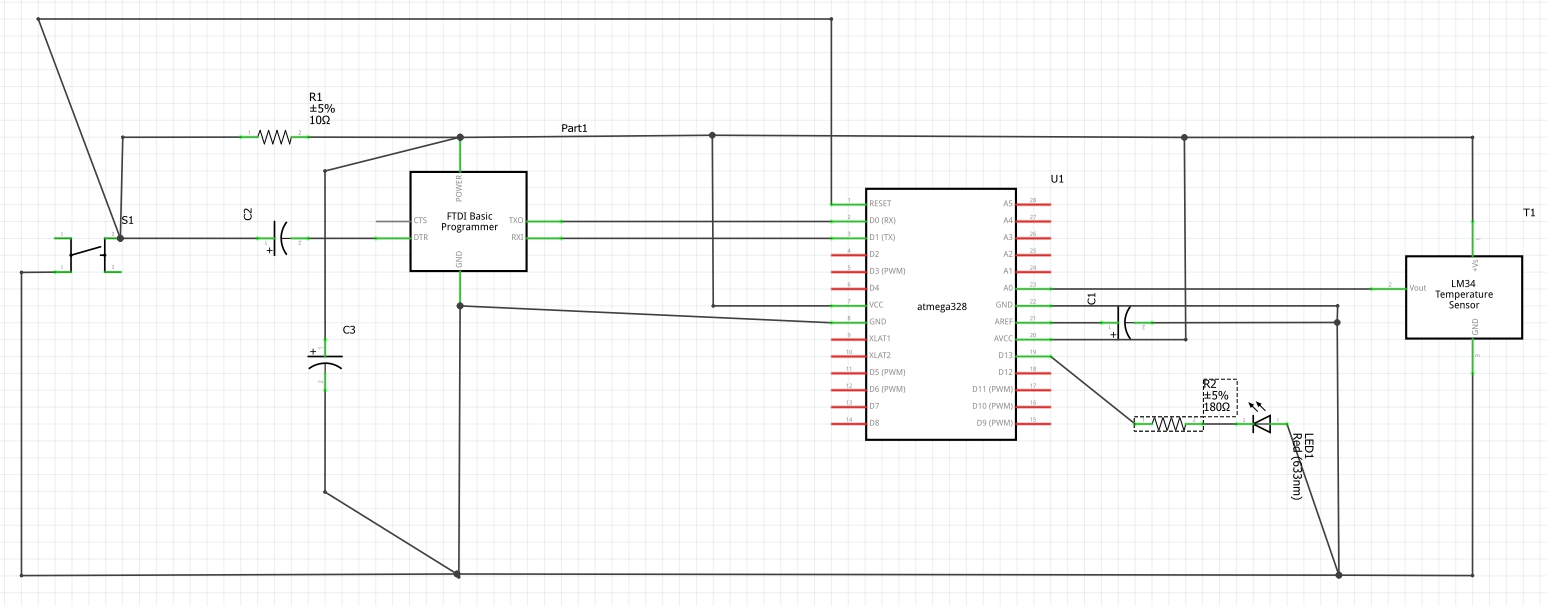
Components used:

1. Atmega328
2. 3 capacitors (0.1 μF – 100 μF) [Sufficient to safely store 5V]
3. 10K Ω and 200 Ω resistors
4. FTDI Basic 5V

Breadboard



1. **SCHEMATIC**



1. **C CODE TO MEASURE THE TEMPERATURE (F)**

#include <avr/io.h>

#include <stdint.h>

#include <avr/interrupt.h>

#define FOSC 8000000 // Clock speed

#define BAUD 9600 // Desire baud rate

#define MYUBRR FOSC/16/BAUD-1 // Formula to set the baud rate

volatile *uint8\_t* ADCvalue; // Storage for the temperature in F

int main(void)

{

// Port declarations

DDRB = (1<<5); // Set PORTB.5 as output

PORTB = 0x00; // Clear PORTB

// ADC declaration

ADMUX = 0; // Use ADC0

ADMUX |= (1<<ADLAR); // Left justified

ADMUX |= (1<<REFS0); // AVcc is reference

ADCSRA |= (1<<ADPS2) | (1<<ADPS1); // 8 MHz with prescalar of 64

ADCSRA |= (1<<ADATE); // Enable auto trigger

ADCSRB = 0; // Free running settings for auto trigger

ADCSRA |= (1<<ADEN); // Enable ADC

ADCSRA |= (1<<ADIE); // Enable ADC interrupt

ADCSRA |= (1<<ADSC); // Start conversion

// USART declaration

UBRR0H = ((MYUBRR)>>8); // Set baud rate for UPPER Register

UBRR0L = MYUBRR; // Set baud rate for LOWER Register

UCSR0B |= (1<<TXEN0); // Enable transmitter

UCSR0C |= (1<<UCSZ01) | (1<<UCSZ00);// Frame: 8-bit Data and 1 Stop bit

// F = 8 MHz

TCNT1 = 34286; // 65536-(8 MHz/256)

TIMSK1 = (1<<TOIE1); // Enable TIMER1 OVF interrupt

TCCR1A = 0x00;

TCCR1B = 0x04; // Start TIMER1 with prescalar 256

TCCR1C = 0x00;

sei(); // Enable global interrupts

while (1)

{

}

return 0;

}

// Interrupt subroutine for TIMER1 overflow (1 second)

ISR(TIMER1\_OVF\_vect)

{

TIFR1 = (1<<TOV1); // Clear TOV1 flag

PORTB ^= (1<<5); // Toggle LED

TCNT1 = 34286; // Reset TCNT1

while(!(UCSR0A & (1<<UDRE0))); // Wait for the transmitter to finish

UDR0 = ADCvalue; // Transmit the the new value

}

// Interrupt subroutine for ADC value

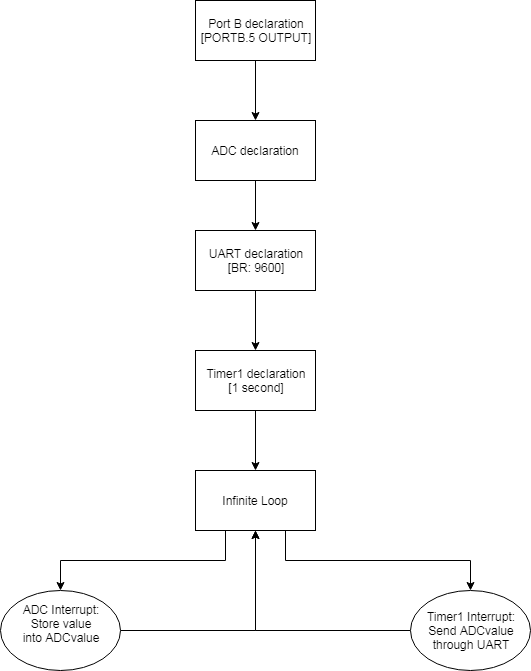
ISR(ADC\_vect)

{

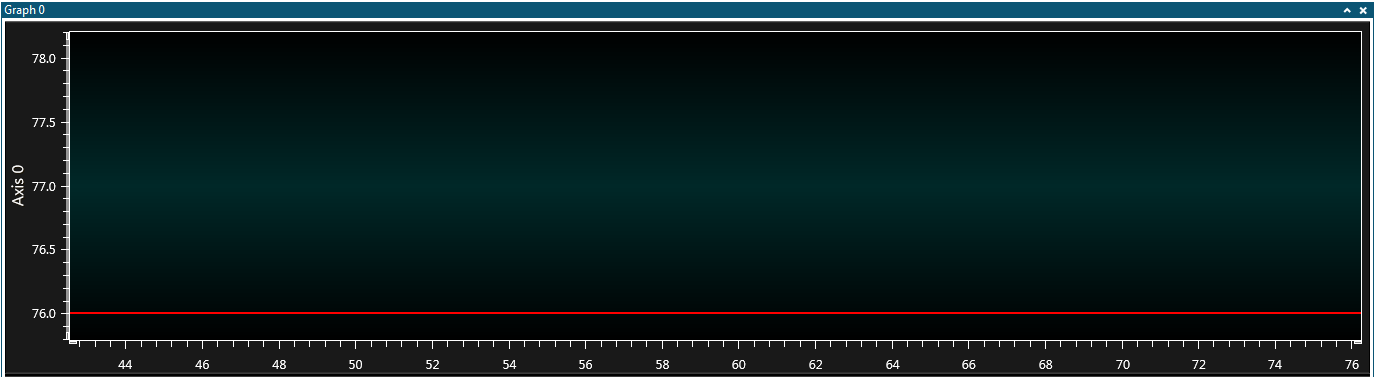
ADCvalue = (ADCH<<1);// Store the decimal value of the converted signal

} // Shift left by one to multiply by 2 and adjust the value

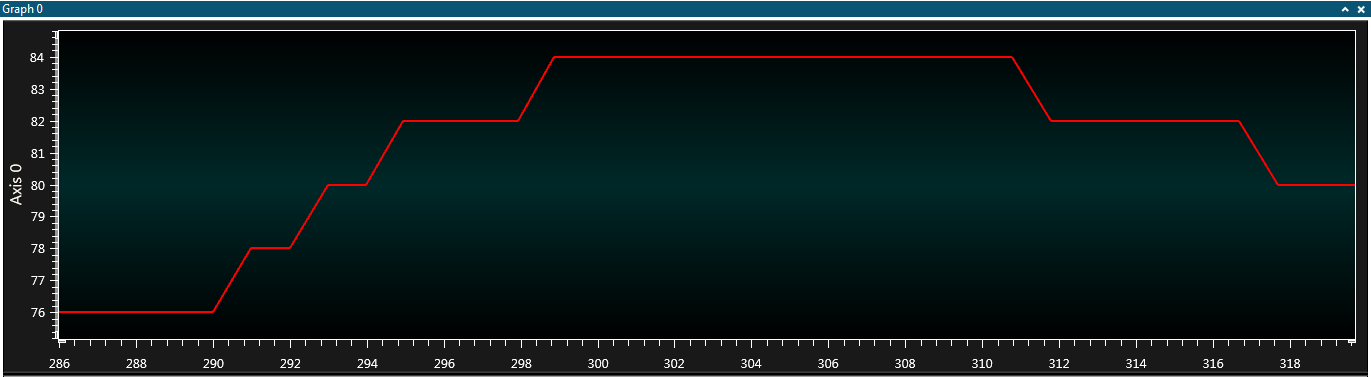
1. **FLOWCHART OF C CODE**



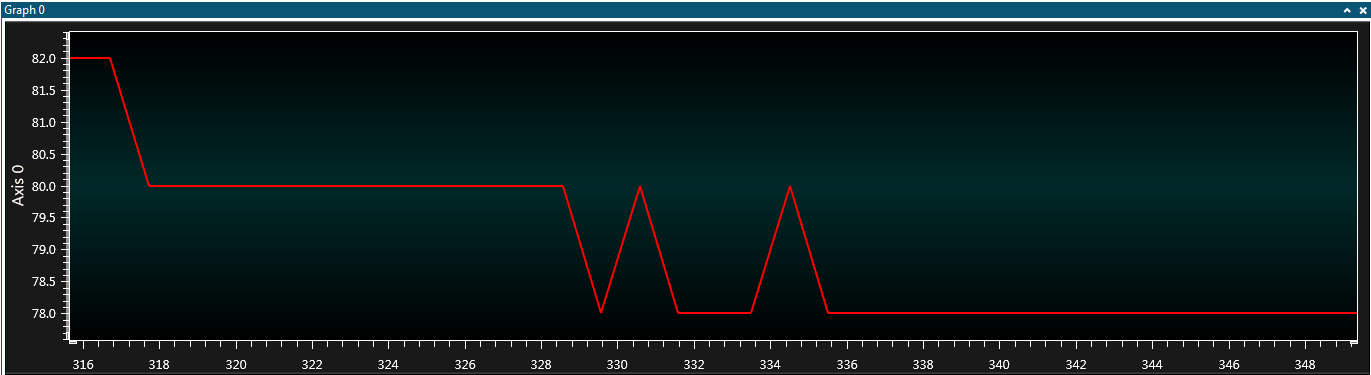
1. **SCREENSHOTS OF DATA VISUALIZER WAVEFORM**

****

*Figure 1: Steady measurement of room temperature*

****

*Figure 2: Temperature spikes after touching the LM34*

****

*Figure 3: Temperature returns to room temperature*

**Student Academic Misconduct Policy**

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

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

Bryan Takemoto