Bryan Takemoto

CPE301 – SPRING 2018

Design Assignment Mid-Term

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		
2.	C CODE		
3.	FLOWCHART		
4.	THINGSPEAK GRAPH OF THE TEMPERATURE VIA UART [ESP8266]		
5.	VIDEO LINK EXPLAINING THE BREADBOARD AND OPERATION		

1. COMPONENTS LIST

Essential components used: Atmega328, LM34, FTDI, and ESP8266.

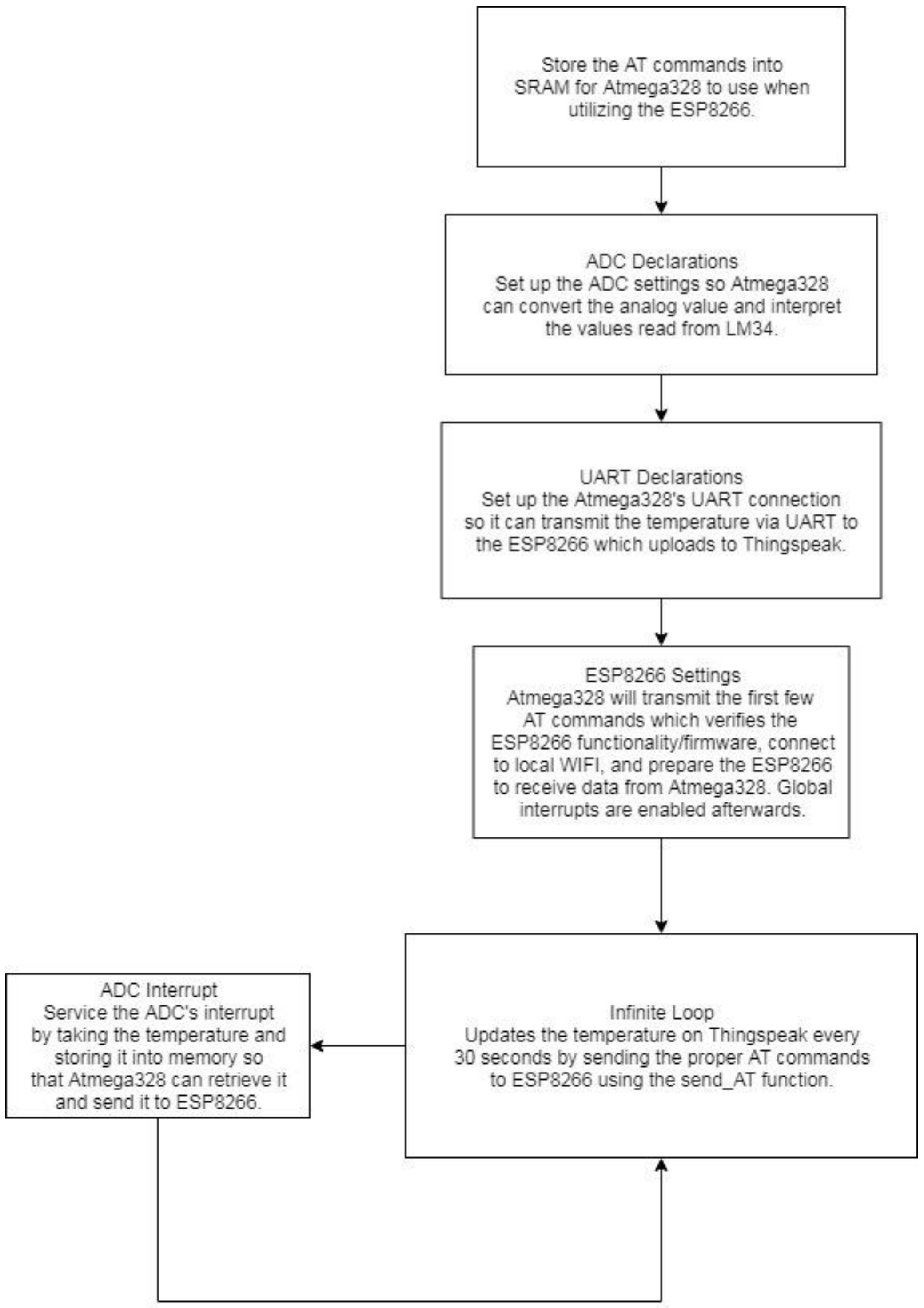
2. C CODE

```c
#define F_CPU 16000000UL
#include <avr/io.h>
#include <stdint.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>

#define FOSC 16000000                     // Clock speed
#define BAUD 115200                       // Desire baud rate
#define MYUBRR FOSC/8/BAUD-1              // Formula to set the baud rate
volatile uint8_t ADCvalue;                // Storage for the temperature in F
volatile unsigned char TEMP[5];           // ASCII temp value

// AT Commands
volatile unsigned char CWMODE[] = "AT+CWMODE=3\r\n";
volatile unsigned char WIFI[] = "AT+CWJAP=\"WIFI\",\"PASSWORD\"\r\n";
volatile unsigned char AT[] = "AT\r\n";
volatile unsigned char FIRM[] = "AT+GMR\r\n";
volatile unsigned char CIPMUX[] = "AT+CIPMUX=0\r\n";
volatile unsigned char CIPSTART[] = "AT+CIPSTART=\"TCP\",\"184.106.153.149\",80\r\n";
volatile unsigned char SIZE[] = "AT+CIPSEND=45\r\n";
volatile unsigned char SEND_DATA[] = "GET /update?key=54MRLC7ZQ32UD48T&field1=";
volatile unsigned char END[] = "\r\n\r\n";
void send_AT(volatile unsigned char AT[]);

int main(void)
{
       // ADC declaration
       ADMUX = 0;

       // Use ADC0
       ADMUX |= (1<<ADLAR);        // Left justified
       ADMUX |= (1<<REFS0);        // AVcc is reference
       ADCSRA |= (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0); // 16 MHz with prescalar of 128
       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
       UCSR0A |= (1<<U2X0);        // Double UART transmission speed
       UCSR0B |= (1<<TXEN0);                     // Enable transmitter
       UCSR0C |= (1<<UCSZ01) | (1<<UCSZ00);      // Frame: 8-bit Data and 1 Stop bit
```

```
	// ESP8266 settings

	_delay_ms(1000);
	send_AT(AT);

	_delay_ms(2000);	// Display firmware
	send_AT(FIRM);

	_delay_ms(2000);	// Select WIFI mode
	send_AT(CWMODE);

	_delay_ms(2000);	// Connect to local WIFI
	send_AT(WIFI);

	_delay_ms(10000);	// Enable connection
	send_AT(CIPMUX);

	sei();

	// Send temperature to Thingspeak server every 30 seconds
    while (1)
    {
		_delay_ms(500);		// Start a connection as client to Thingspeak
		send_AT(CIPSTART);

		_delay_ms(500);		// Specify the size of the data
		send_AT(SIZE);

		_delay_ms(1000);	// Send temperature
		send_AT(SEND_DATA);
		send_AT(TEMP);
		send_AT(END);
		_delay_ms(28000);
    }
	return 0;
}

// Interrupt subroutine for ADC value
ISR(ADC_vect)
{
	unsigned char i = 0x00;
	char temperature[5];
	ADCvalue = (ADCH<<1);		// Store the decimal value of the converted signal
	itoa(ADCvalue, temperature, 10);
	for(i = 0x00; i < 5; i++)
	{
		TEMP[i] = temperature[i];
	}
}

void send_AT(volatile unsigned char AT[])
{
	volatile unsigned char len = 0;
	volatile unsigned char i;
	while(AT[len] != 0)
	{
		len++;
	}
```

```
        for(i = 0x00; i < len; i++){
                while(!(UCSR0A & (1<<UDRE0)));      // Wait for the transmitter to finish
                UDR0 = AT[i];                       // Transmit the the new value
        }
}
```

3. FLOWCHART



Store the AT commands into SRAM for Atmega328 to use when utilizing the ESP8266.
ADC Declarations
Set up the ADC settings so Atmega328 can convert the analog value and interpret the values read from LM34.
UART Declarations
Set up the Atmega328's UART connection so it can transmit the temperature via UART to the ESP8266 which uploads to Thingspeak.
ESP8266 Settings
Atmega328 will transmit the first few AT commands which verifies the ESP8266 functionality/firmware, connect to local WIFI, and prepare the ESP8266 to receive data from Atmega328. Global interrupts are enabled afterwards.
Infinite Loop
Updates the temperature on Thingspeak every 30 seconds by sending the proper AT commands to ESP8266 using the send_AT function.
ADC Interrupt
Service the ADC's interrupt by taking the temperature and storing it into memory so that Atmega328 can retrieve it and send it to ESP8266.

4. THINGSPEAK GRAPH OF THE TEMPERATURE VIA UART [ESP8266]

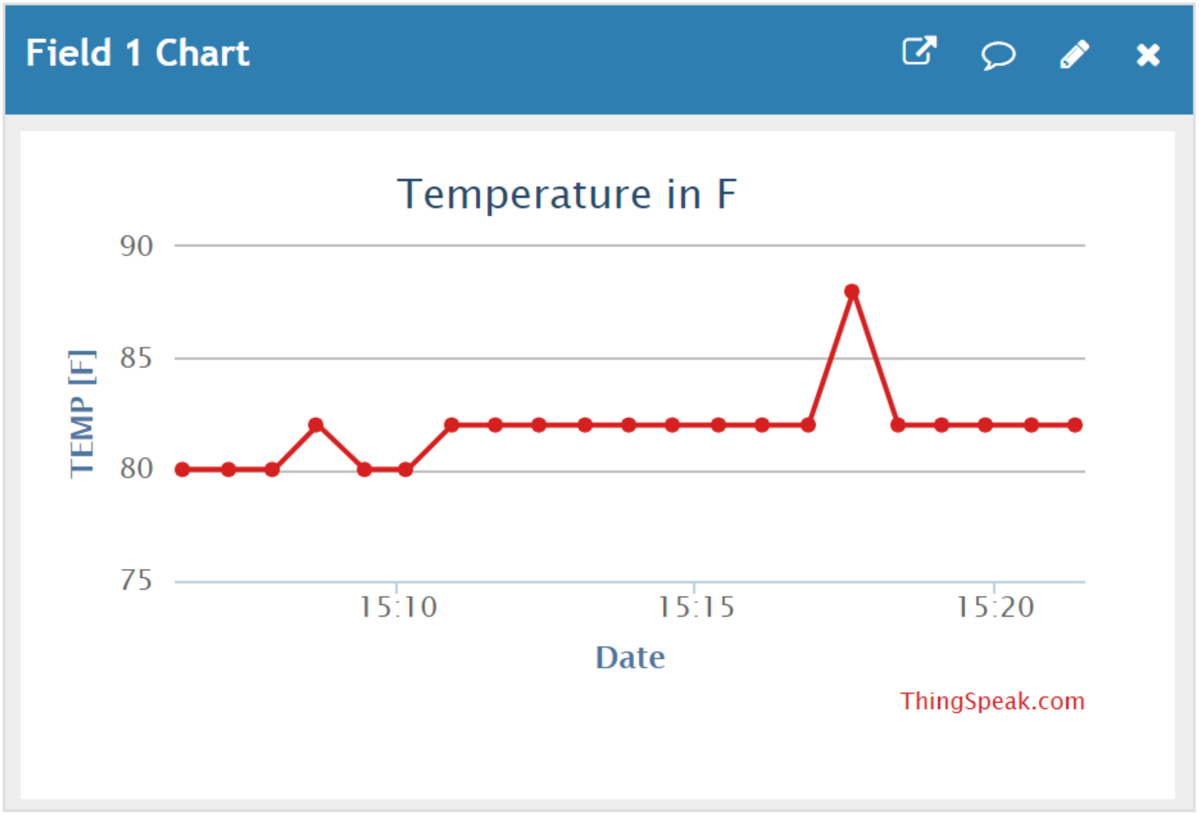


Image of the temperature graphed on Thingspeak. Notice that there is a temperature change at 15:17 after placing my finger on the LM34.

5. VIDEO LINK OF EXPLAINING THE BREADBOARD AND OPERATION

https://youtu.be/1meMs0GoMAU

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

"This assignment submission is my own, original work".
Bryan Takemoto