Jeffrey Razon

CPE301 – SPRING 2018

Design Assignment MIDTERM

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

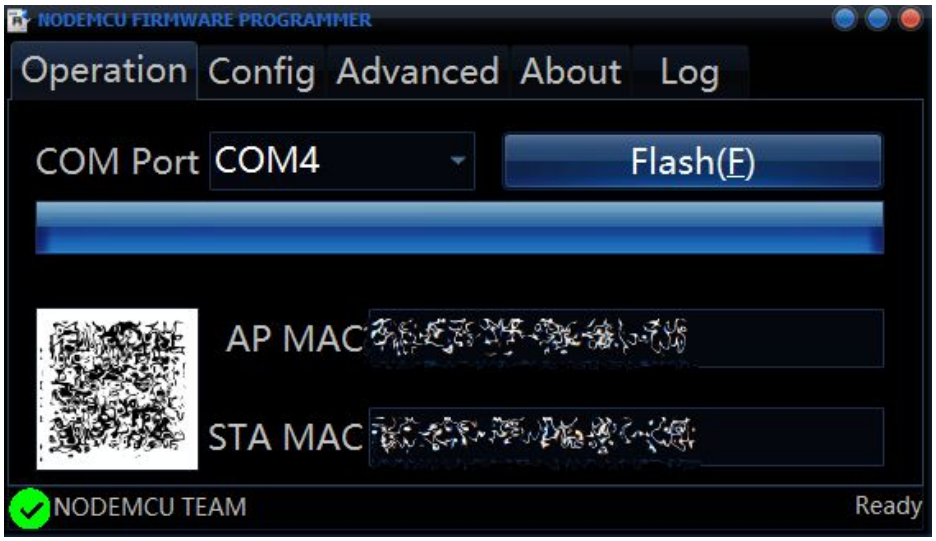
The student understands that all required components should be submitted in complete for grading of this assignment.

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		

Task 1/A. Program the ADC of ATmega328/p to read the LM34/35 temperature sensor.
FINISHED IN DA03, REFER TO DA03 DOC OR REFER TO FULL CODE LATER IN DOCUMENT

Task 2/B. Display the value to UART.
FINISHED IN DA03, REFER TO DA03 DOC

Task 3/C. Make sure the AT Firmware is downloaded into the ESP8266-01 module.
a) Proof of successful firmware download (green check bottom-left)
- distorted QR box and AP/STA MAC for safety purposes, FTDI was in COM4



Task 4/D. Register for a free Thingspeak account with MATHWORK. Setup and get the channel Key.
a) Proof of ThingSpeak account creation

My Profile

MathWorks Account settings

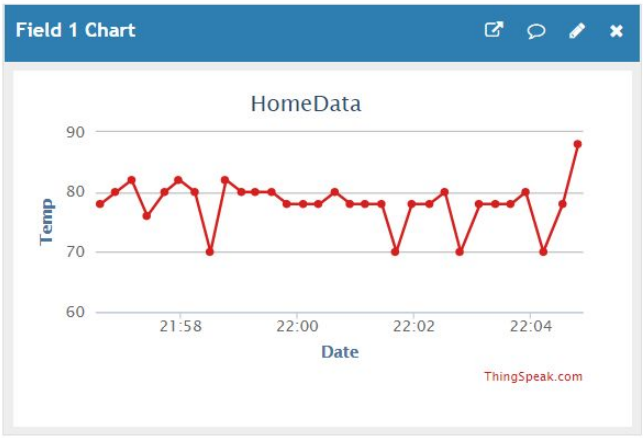
MathWorks Account E mail	razonj1@unlv.nevada.edu
User ID	razonj1
Password	**********

Task 5/E. Transmit temperature sensor value to ESP8266-01 through UART port using AT Commands.
REFER TO FULL CODE LATER IN DOCUMENT
Utilized this site for AT commands:
https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/

Task 6/F. Display the temperature sensor value as a graph in Thingspeak
a) Graph on ThingSpeak
Notes:
- Normal temperature was around 78°-80° F.
- Temperature rises when pinching LM34.
- Temperature drops when placed cold object on top of LM34.

ThingSpeak™

Created: about 4 hours ago
Updated: less than a minute ago
Last entry: less than a minute ago
Entries: 31



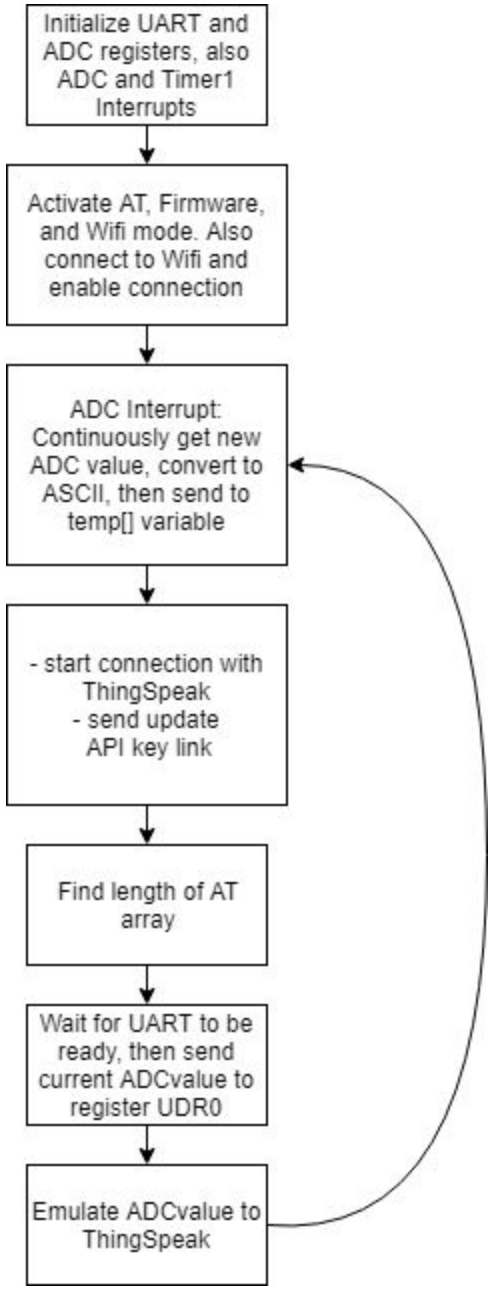
Full Code:

```
#define F_CPU 16000000UL

#include <stdlib.h>
#include <avr/io.h>
#include <stdint.h> // needed for uint8_t
#include <util/delay.h> // delays
#include <avr/interrupt.h>

//calc baud rate
#define FOSC 16000000              // Clock speed
#define BAUD 115200
#define MYUBRR FOSC/8/BAUD-1

//AT commands for ESP
volatile unsigned char AT[] = "AT\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"; //ip of thingspeak
volatile unsigned char SEND_DATA[] = "GET /update?key=4YEZ22JG6Q4EMQEC&field1="; //update thingspeak

volatile unsigned char CIPSIZE[] = "AT+CIPSEND=45\r\n"; //send data
volatile unsigned char CWMODE[] = "AT+CWMODE=3\r\n"; //wifi mode
volatile unsigned char CONNECTWIFI[] = "AT+CWJAP=\"WIFI NAME\",\"WIFI PASSWORD\"\r\n"; //connect to ap
volatile unsigned char FIRMWARE[] = "AT+GMR\r\n"; //view version info
volatile unsigned char BREAK[] = "\r\n\r\n"; //assures end of reading temp

//global variables
volatile uint8_t ADCvalue;
volatile unsigned char temp[5];

//prototypes
void initUart();
void send_AT(volatile unsigned char AT[]);

int main(void)
{

    ADMUX = 0; // use ADC0
    ADMUX |= (1 << REFS0); // use AVcc as the reference
    ADMUX |= (1 << ADLAR); // Right adjust for 8 bit resolution

    ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // 128 prescale for 16Mhz
    ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable
    ADCSRB = 0; // 0 for free running mode
    ADCSRA |= (1 << ADEN); // Enable the ADC
    ADCSRA |= (1 << ADIE); // Enable Interrupts
    ADCSRA |= (1 << ADSC); // Start the ADC conversion

    initUart();

    _delay_ms(1500);
    send_AT(AT); //at
    _delay_ms(1500);
    send_AT(FIRMWARE); //firmware
    _delay_ms(1500);
    send_AT(CWMODE); //wifi mode
    _delay_ms(1500);
    send_AT(CONNECTWIFI); //connect with WiFi
    _delay_ms(5000);
    send_AT(CIPMUX); //enable

    sei();
    while (1)
    {
        _delay_ms(500);
        send_AT(CIPSTART); // start connection
        _delay_ms(500);
        send_AT(CIPSIZE); // size
        _delay_ms(500);
        send_AT(SEND_DATA);
        send_AT(temp); //temperature
        send_AT(BREAK);
    }
    return 0;
}
```

```
// Interrupt subroutine for ADC value
ISR(ADC_vect) {
    unsigned char i = 0;
    char tmptemp[5];

    ADCvalue = (ADCH << 1);
    itoa(ADCvalue, tmptemp, 10); //convert char to ascii
    for(i = 0 ; i < 5 ; i++)
        temp[i] = tmptemp[i]; //move converted ascii
}

void send_AT(volatile unsigned char AT[]) {
    volatile unsigned char a;
    volatile unsigned char ATlength = 0;

    while(AT[ATlength] != 0)
        ATlength++; // find length

    for(a = 0 ; a < ATlength ; a++)
    {
        while(!(UCSR0A & (1<<UDRE0)));  // wait for UART
        UDR0 = AT[a];                   // transmit char array
    }
}
```

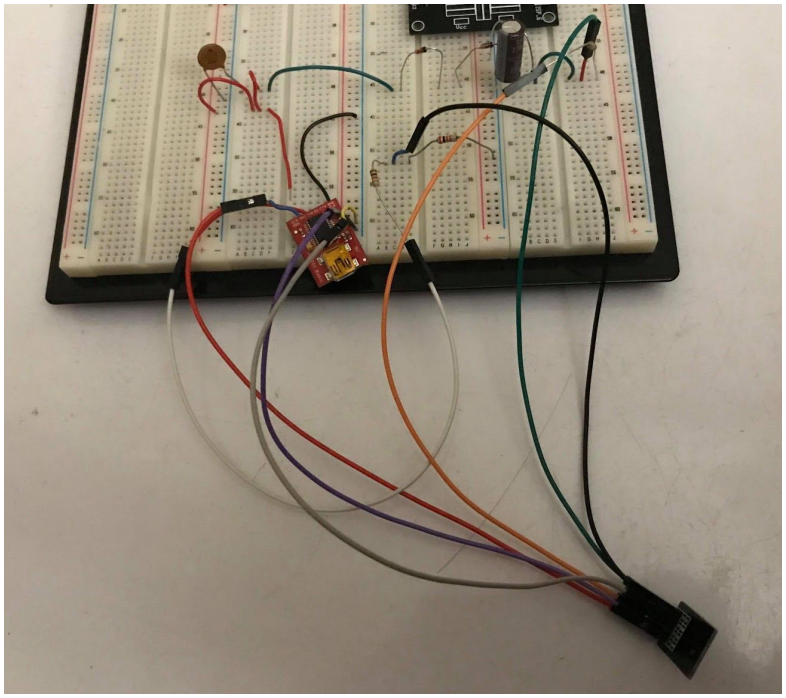
Flowchart:

a) Flowchart

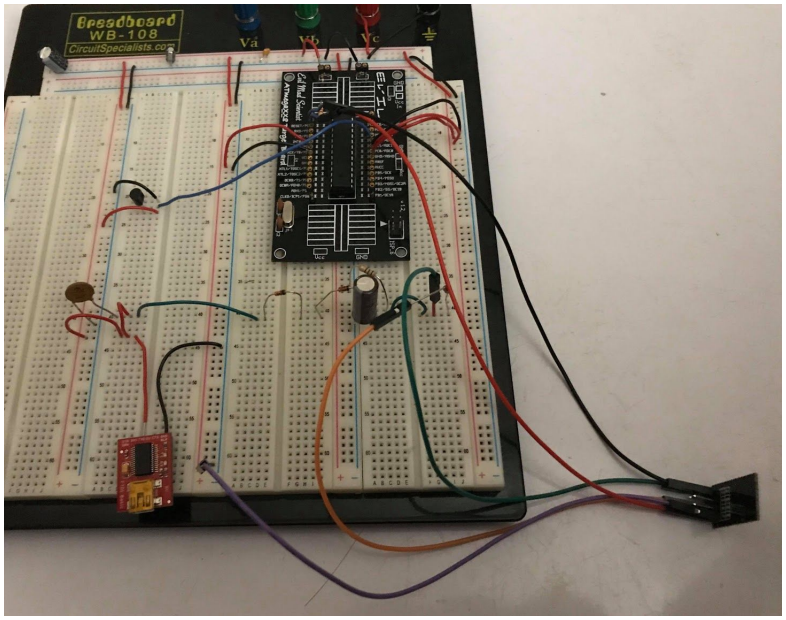


Physical Set-up:

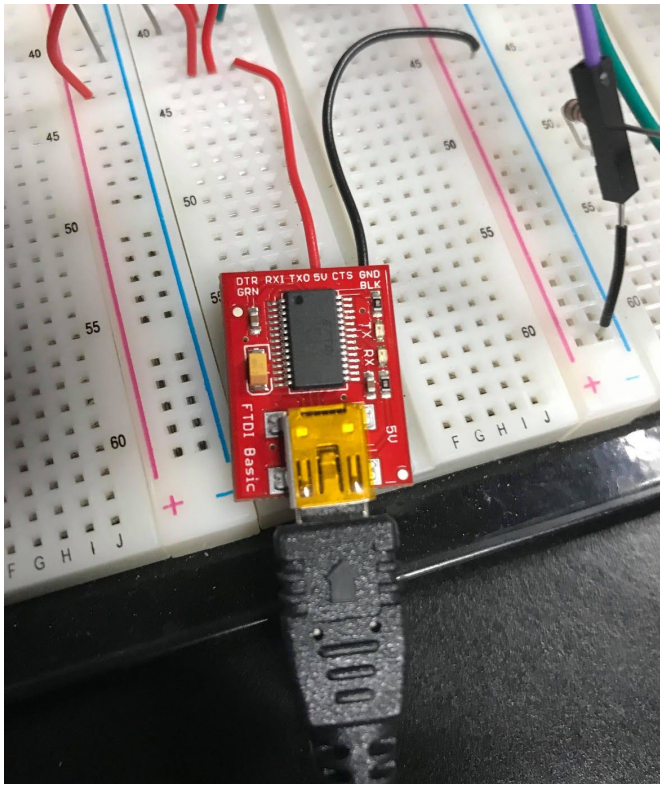
a) Circuit to install ESP-01 firmware (FTDI, ESP)



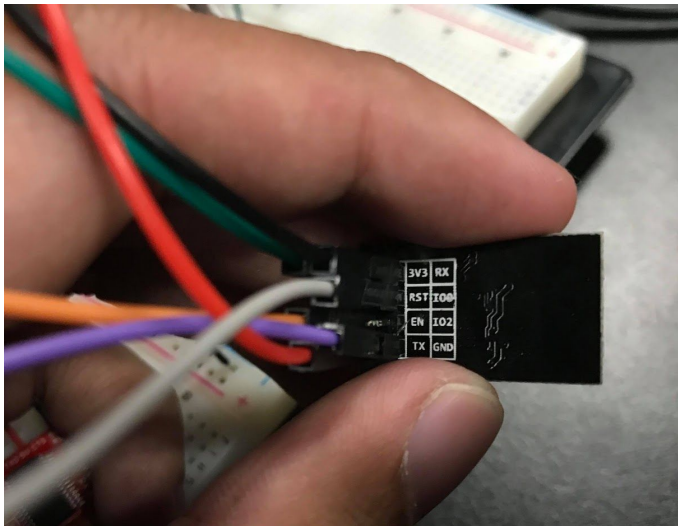
b) Full Breadboard circuit (ATMega328P, FTDI chip, ESP)



c) FTDI chip



d) ESP01 chip



GITHUB LINK: https://github.com/JeffinVegas/EmbSys.git
YOUTUBE LINK: In the videos_DA_MIDTERM.txt file

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

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