CPE301 – SPRING 2019

MIDTERM 1

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:

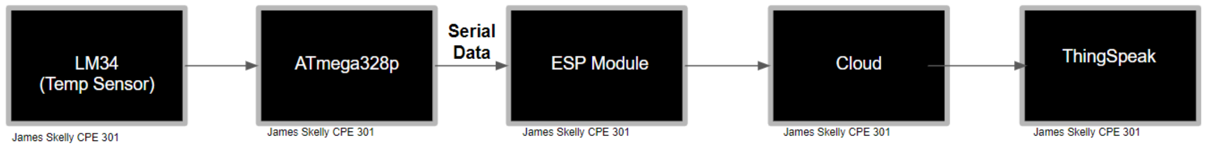
1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/Midterm, 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

List of Components:

- ATmega328P and Xplained Mini Board
- Atmel Studio 7
- Breadboard, jumper wire, USB cables
- ESP8266 Module
- ESPlorer Software (for communication with MCU)
- ESP8266Flasher Software (for flashing firmware to ESP)
- FTDI Chip and USART Module (for flashing firmware to ESP)

Block Diagram:



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

```c
/*
 James Skelly, DA3B C Code
 Displaying the temperature readout from LM34 on Data Visualizer Terminal every second.
 */

#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
```

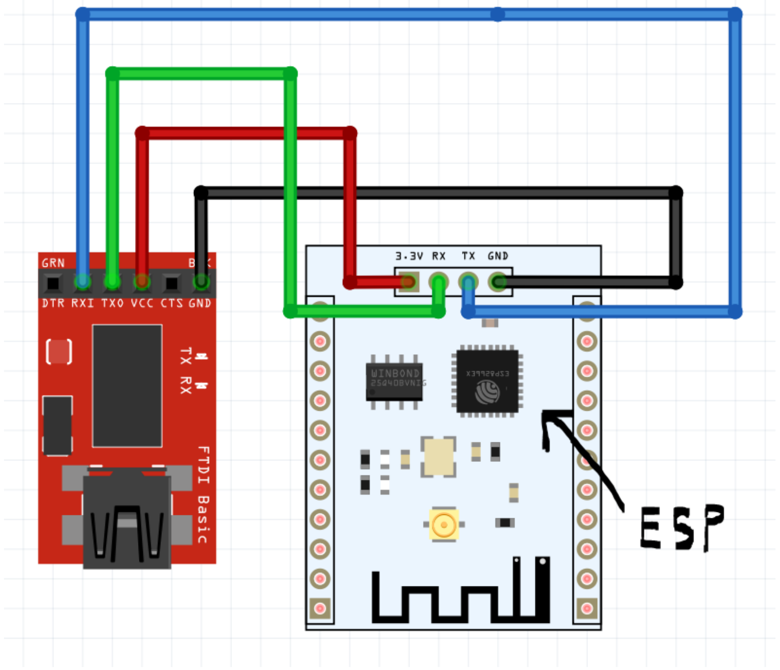
```
              {                                  // code to convert output to Fahrenheit

                     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

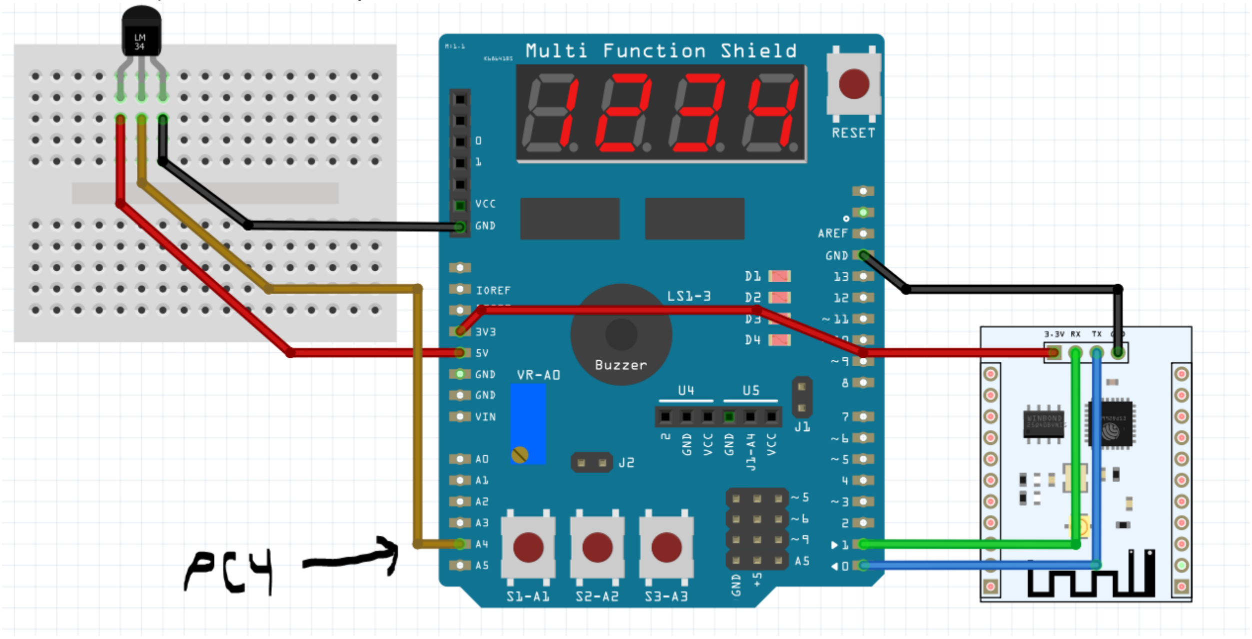
```
/*
 James Skelly, Midterm 1
 Display the temperature (vs. time) from LM34 as a graph on ThingSpeak.com
 */

#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
#define BAUD_PRESCALLER F_CPU/16/BAUD-1 // Baudrate prescaller

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

int overflow; // initialize overflow variable for delay

// function prototypes
void usart_init ();
void usart_send (unsigned char ch);
void usart_print (char* str);
void read_adc(void);  //Read LM34 using ADC
volatile unsigned int adc_temp; // Volatile raw temperature variable
volatile unsigned int tempF; // Volatile Fahrenheit temperature variable
char outs[256]; // String array used for sending USART commands
volatile char received_data; // String array used for receiving USART communication

int main (void)
{
         // 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)|     // ADC4 (PC4)
         (0<<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)|      // ADC4 (PC4)
         (0<<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

        usart_init (BAUD_PRESCALLER);
        _delay_ms(500);                                     // Delay to allow hardware to initialize

        while (1)
        {
                // Repeatedly read the temperature value from the ADC and print to Thingspeak

                // AT COMMANDS
                // Check connection with handshake
                char AT[] = "AT\r\n";
                // Set mode 1 => Station mode
                char AT_CWMODE[] = "AT+CWMODE=1\r\n";
                // Connect to Wifi, using correct network name and password
                char AT_CWJAP[] = "AT+CWJAP=\"WiFi_Network\",\"PASSWORD\"\r\n";
                // Single IP Address Mode
                char AT_CIPMUX[] = "AT+CIPMUX=0\r\n";
                // Connect to Thingspeak.com (port 80)
                char AT_CIPSTART[] = "AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n";
                // Set string length
                char AT_CIPSEND[] = "AT+CIPSEND=100\r\n";

                // send the previous commands
                _delay_ms(200);
                usart_send(AT);
                _delay_ms(5000);
                usart_send(AT_CWMODE);
                _delay_ms(5000);
                usart_send(AT_CWJAP);
                _delay_ms(3000);
                usart_send(AT_CIPMUX);
                _delay_ms(3000);
                usart_send(AT_CIPSTART);
                _delay_ms(3000);
                usart_send(AT_CIPSEND);
                _delay_ms(5000);



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

                        PORTC^= (1<<5);

                        read_adc(); // Read next ADC value from LM34
                        adc_temp = (adc_temp/1024.0)*500.0; // Convert to Fahrenheit
                        tempF = adc_temp;
                // Print Data to Thingspeak using provided link, website channel key, and field location
                        snprintf(outs,sizeof(outs),"INSERT WRITE KEY from THINGSPEAK HERE", tempF);
                        USART_tx_string(outs);//send data
                        _delay_ms(5000);
                        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
{
        UBRR0H = (unsigned char)(ubrr>>8);
        UBRR0L = (unsigned char)ubrr;
        // enable transmit, receive, interrupt
        UCSR0B = (1<<TXEN0) | (1 << RXEN0)| ( 1 << RXCIE0);
        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]);
}
```

4. SCHEMATICS

Some pins are excluded from the available ESP module in Fritzing. For example, the GPIO pins IO0 and IO2, the RESET pin, and the enable pins are not present. Those needed to be left off of the schematic for that purpose, but note that for programming (below), the GPIO pin IO0 was grounded.

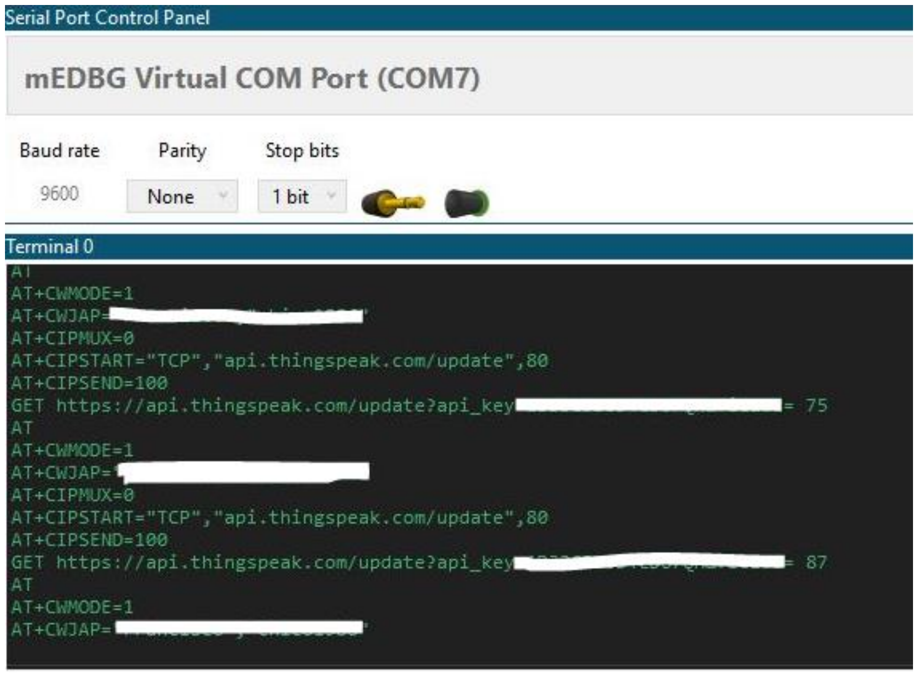


Here we see the LM34 connected to the multifunction shield, which is connected to the Xplained Mini and ATmega328P (not pictured, unavailable in fritzing). The LM34 feeds data to pin PC4 of the Mini (A4 on the shield), and the TX/RX pins of the shield transmit and receive data from the WiFi Module.



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

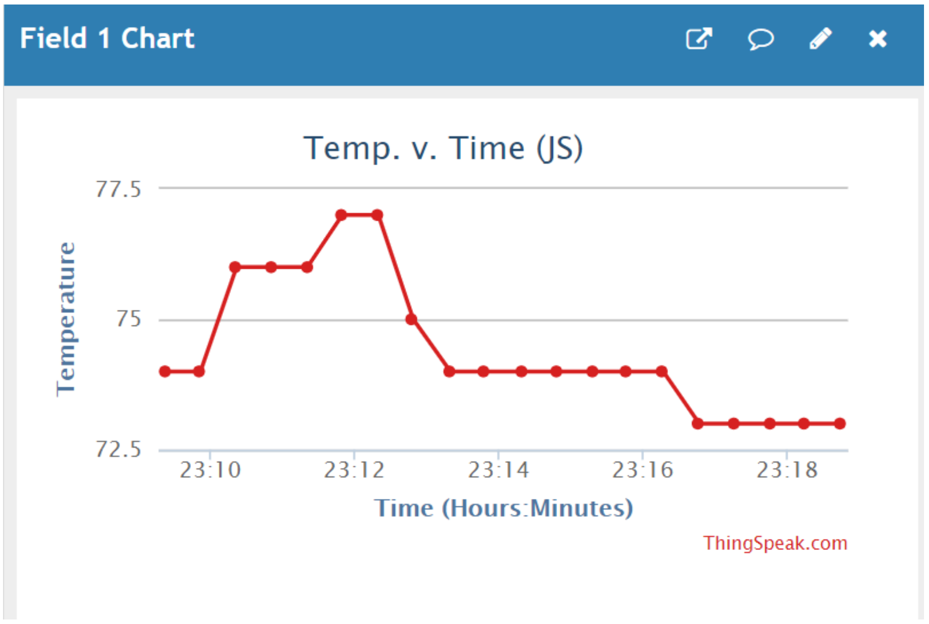
The data visualizer in Atmel Studio 7 is shown below, reading the temperature values out of the ESP module to the terminal. In the photo below, the wifi network used, the password, and the write key of my ThingSpeak channel are whited out for privacy.



Baud rate: 9600 | Parity: None | Stop bits: 1 bit

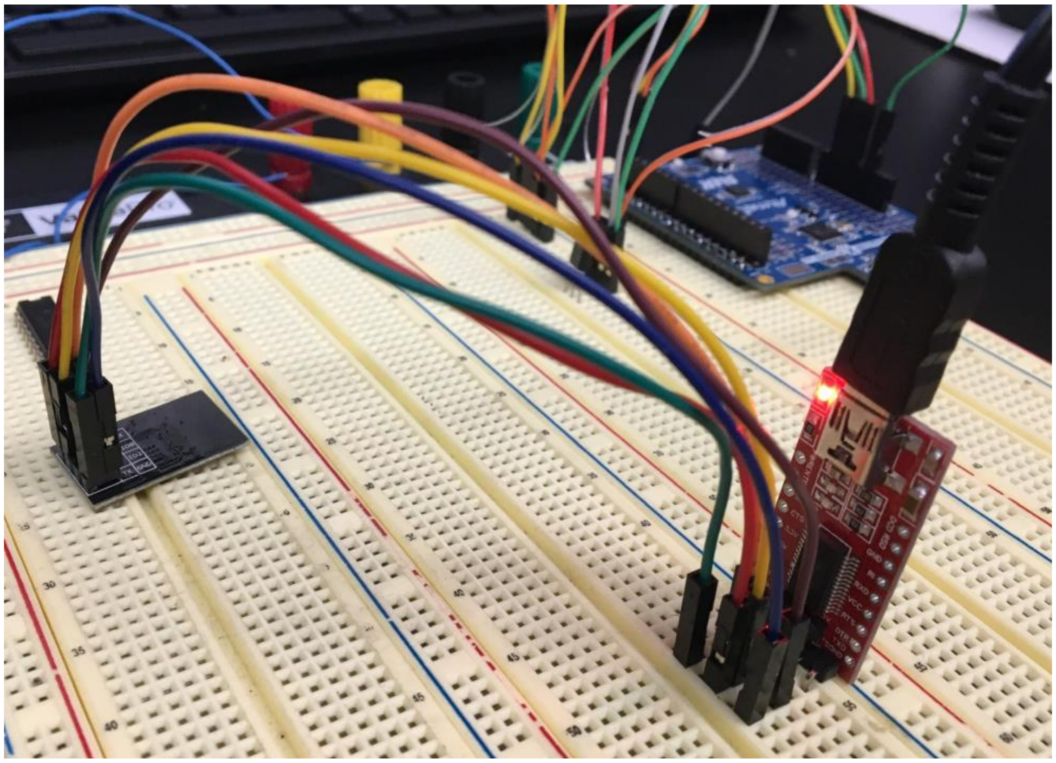
```
AT
AT+CWMODE=1
AT+CWJAP=[redacted]
AT+CIPMUX=0
AT+CIPSTART="TCP","api.thingspeak.com/update",80
AT+CIPSEND=100
GET https://api.thingspeak.com/update?api_key[redacted]= 75
AT
AT+CWMODE=1
AT+CWJAP=[redacted]
AT+CIPMUX=0
AT+CIPSTART="TCP","api.thingspeak.com/update",80
AT+CIPSEND=100
GET https://api.thingspeak.com/update?api_key[redacted]= 87
AT
AT+CWMODE=1
AT+CWJAP=[redacted]
```

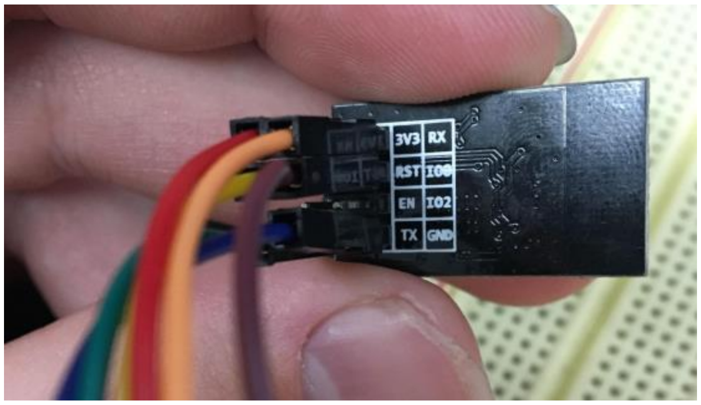
Shown here is the ThingSpeak graph after 20 data points were taken. The temperature shoots up in the graph when I hold my fingers over the LM34, heating it up. The time taken between points is roughly 25 seconds.



6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Below is the board setup (FTDI module connected to ESP module) used to flash the firmware onto the ESP module. A more detailed image of the wiring is found in the next image.

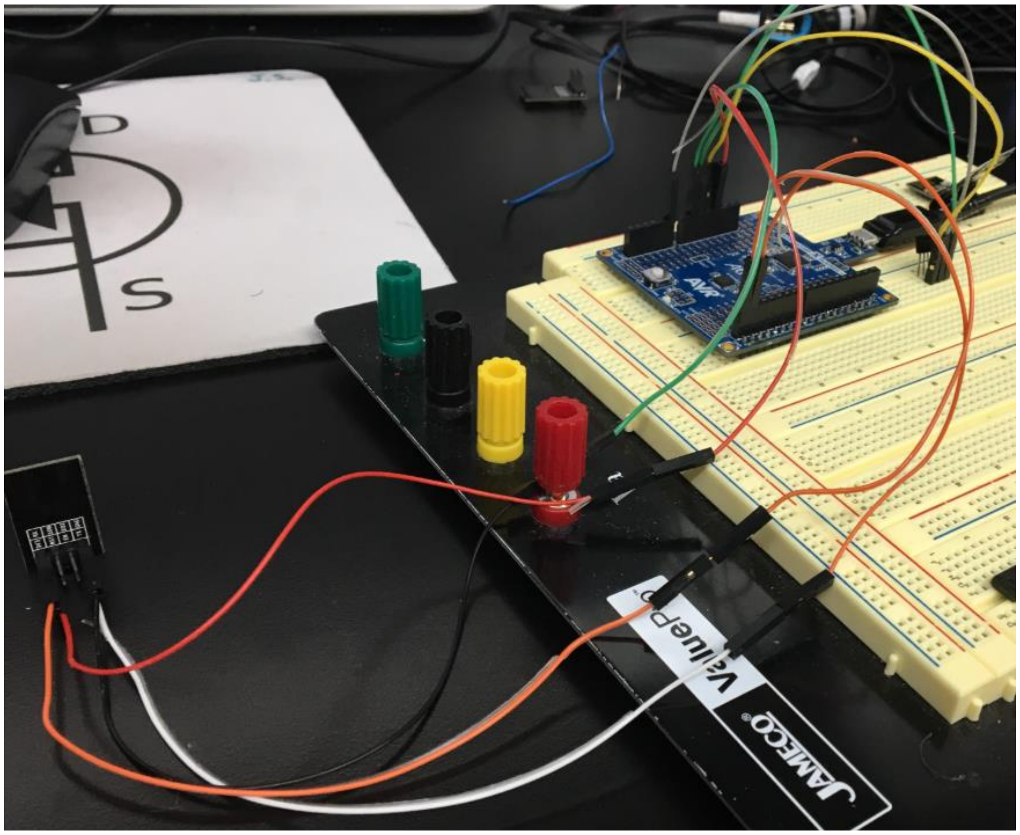


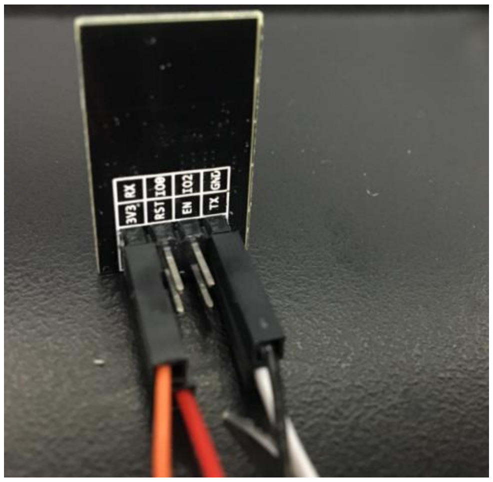


- 3V3 connects to VCC
- RST connects to GND (to reset)
- GPIO0 connects to GND (to flash)
- TX (ESP) connects to RX (FTDI)
- RX (EXP) connects to TX (FTDI)
- GND connects to GND

The setup described above is specific to flashing the firmware onto the ESP8266. Once flashed, the GPIO0 pin was disconnected from ground to take the ESP out of “programming mode”.

The board setup below was used to actually send data to ThingSpeak through the WiFi that our ESP module was connected to. Here we see the Xplained Mini board connected to the PC by USB. While the C program was running in Atmel Studio, the ESP module was connected in the configuration seen in the next picture.

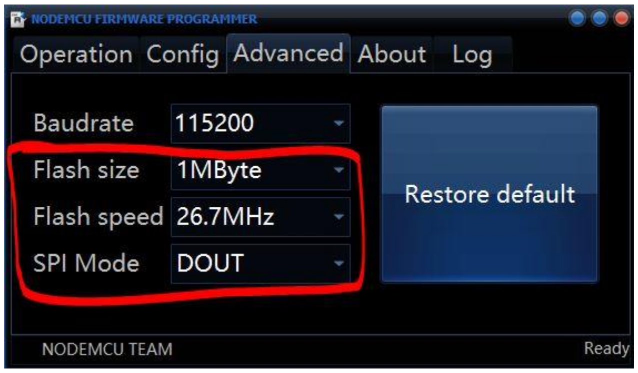




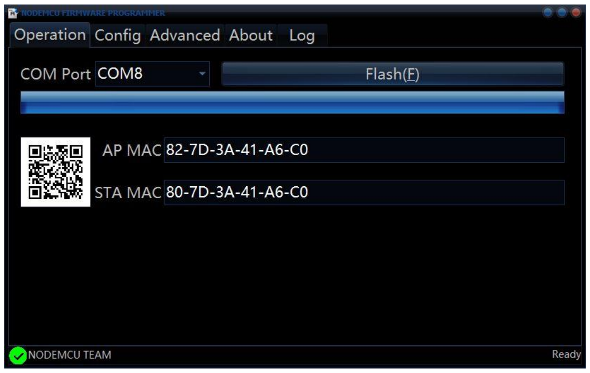
- 3V3 (ESP) connects to 3V3 (Mini)
- GND (ESP) connects to GND (Mini)
- TX (ESP) connects to PD0 (Mini)
- RX (ESP) connects to PD1 (Mini)
- Vs (LM34) connects to 5V (Mini)
- Vout (LM34 connects to PC4 (Mini)
- GND (LM34) connects to GND (Mini)

The next page will explain the process of flashing the ESP.

Advanced settings needed to be adjusted to properly flash the ESP8266 using the ESP8266Flasher, seen below.

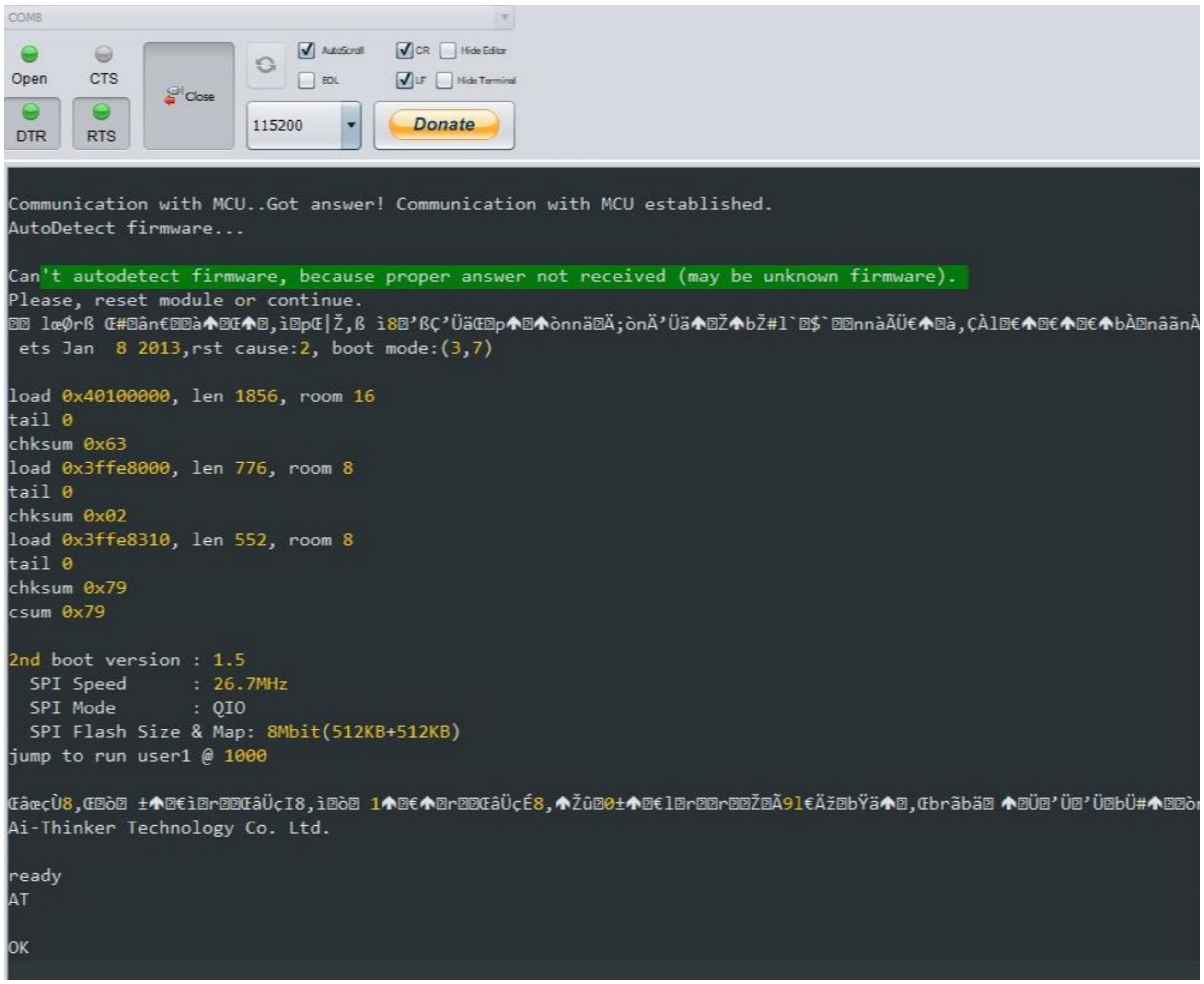


Baudrate 115200
Flash size 1MByte
Flash speed 26.7MHz
SPI Mode DOUT



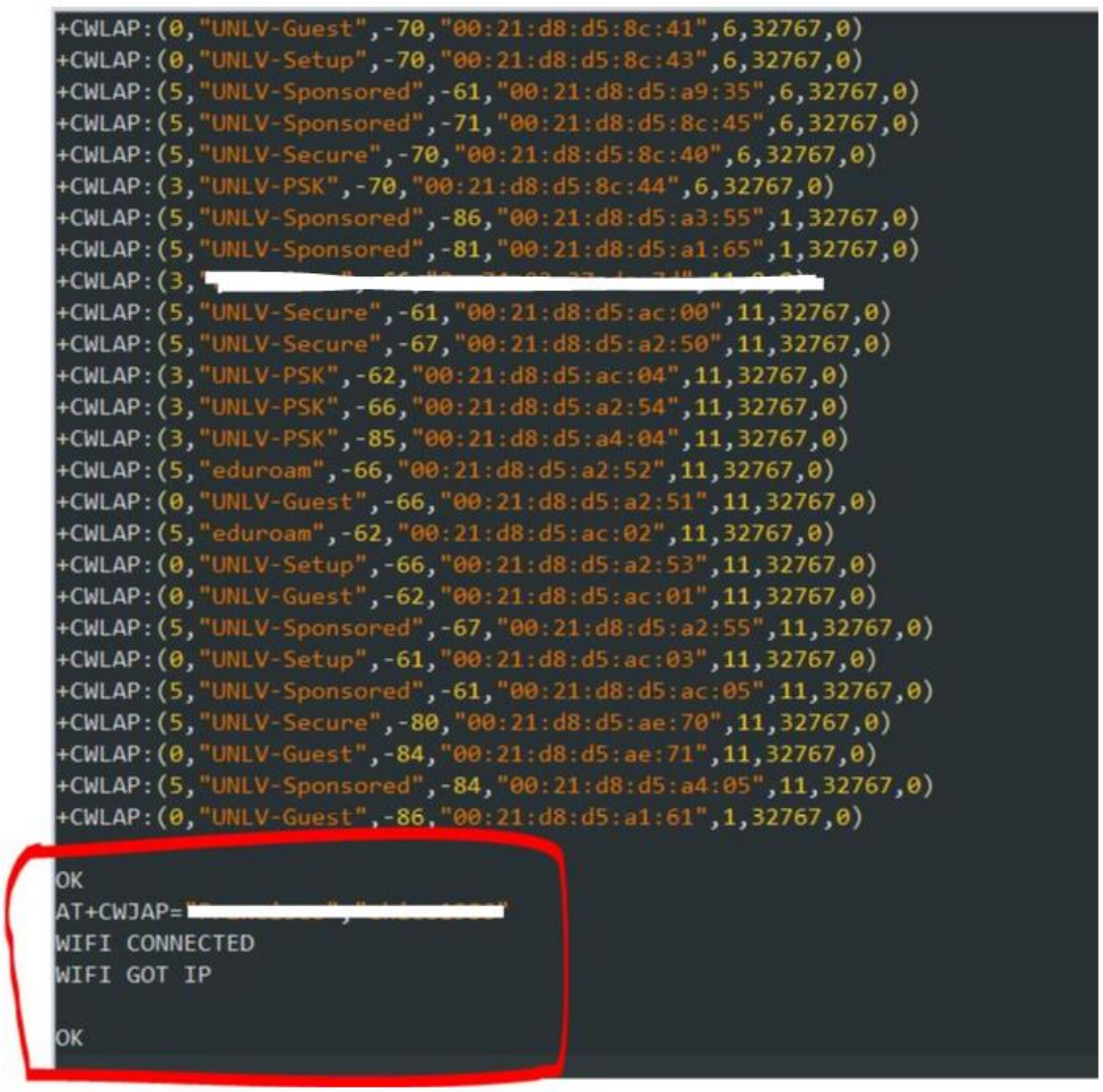
COM Port COM8
AP MAC 82-7D-3A-41-A6-C0
STA MAC 80-7D-3A-41-A6-C0

Once flashed, the ESPlorer software was used to communicate with the MCU.



```
Communication with MCU..Got answer! Communication with MCU established.
AutoDetect firmware...

Can't autodetect firmware, because proper answer not received (may be unknown firmware).
Please, reset module or continue.
�� lœØrß Œ#�ân€��à🡅��Œ🡅�,i�pŒ|Ž,ß i8�'ßÇ'ÜäŒ�p🡅�🡅ònnä�Ä;ònÄ'Üä🡅�Ž🡅bŽ#l`�$`��nnàÃÜ€🡅�à,ÇÀl�€🡅�€🡅�€🡅bÀ�nâãnÀ
 ets Jan  8 2013,rst cause:2, boot mode:(3,7)

load 0x40100000, len 1856, room 16
tail 0
chksum 0x63
load 0x3ffe8000, len 776, room 8
tail 0
chksum 0x02
load 0x3ffe8310, len 552, room 8
tail 0
chksum 0x79
csum 0x79

2nd boot version : 1.5
  SPI Speed      : 26.7MHz
  SPI Mode       : QIO
  SPI Flash Size & Map: 8Mbit(512KB+512KB)
jump to run user1 @ 1000

ŒâæçÙ8,Œ��ò� ±🡅�€ì�r��ŒâÜçI8,ì�ò� 1🡅�€🡅�r��ŒâÜçÉ8,🡅Žû�0±🡅�€l�r��r��Ž�Ã9l€ÄŽ�bŸä🡅�,Œbrãbä� 🡅�Ü�'Ü�'Ü�bÜ#🡅��ò
Ai-Thinker Technology Co. Ltd.

ready
AT

OK
```

Once communication with the MCU was established, a series of commands were sent to the module to test its operation and to ensure that the device could connect successfully to surrounding WiFi networks.



```
+CWLAP:(0,"UNLV-Guest",-70,"00:21:d8:d5:8c:41",6,32767,0)
+CWLAP:(0,"UNLV-Setup",-70,"00:21:d8:d5:8c:43",6,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-61,"00:21:d8:d5:a9:35",6,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-71,"00:21:d8:d5:8c:45",6,32767,0)
+CWLAP:(5,"UNLV-Secure",-70,"00:21:d8:d5:8c:40",6,32767,0)
+CWLAP:(3,"UNLV-PSK",-70,"00:21:d8:d5:8c:44",6,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-86,"00:21:d8:d5:a3:55",1,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-81,"00:21:d8:d5:a1:65",1,32767,0)
+CWLAP:(3,[redacted]
+CWLAP:(5,"UNLV-Secure",-61,"00:21:d8:d5:ac:00",11,32767,0)
+CWLAP:(5,"UNLV-Secure",-67,"00:21:d8:d5:a2:50",11,32767,0)
+CWLAP:(3,"UNLV-PSK",-62,"00:21:d8:d5:ac:04",11,32767,0)
+CWLAP:(3,"UNLV-PSK",-66,"00:21:d8:d5:a2:54",11,32767,0)
+CWLAP:(3,"UNLV-PSK",-85,"00:21:d8:d5:a4:04",11,32767,0)
+CWLAP:(5,"eduroam",-66,"00:21:d8:d5:a2:52",11,32767,0)
+CWLAP:(0,"UNLV-Guest",-66,"00:21:d8:d5:a2:51",11,32767,0)
+CWLAP:(5,"eduroam",-62,"00:21:d8:d5:ac:02",11,32767,0)
+CWLAP:(0,"UNLV-Setup",-66,"00:21:d8:d5:a2:53",11,32767,0)
+CWLAP:(0,"UNLV-Guest",-62,"00:21:d8:d5:ac:01",11,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-67,"00:21:d8:d5:a2:55",11,32767,0)
+CWLAP:(0,"UNLV-Setup",-61,"00:21:d8:d5:ac:03",11,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-61,"00:21:d8:d5:ac:05",11,32767,0)
+CWLAP:(5,"UNLV-Secure",-80,"00:21:d8:d5:ae:70",11,32767,0)
+CWLAP:(0,"UNLV-Guest",-84,"00:21:d8:d5:ae:71",11,32767,0)
+CWLAP:(5,"UNLV-Sponsored",-84,"00:21:d8:d5:a4:05",11,32767,0)
+CWLAP:(0,"UNLV-Guest",-86,"00:21:d8:d5:a1:61",1,32767,0)

OK
AT+CWJAP=[redacted]
WIFI CONNECTED
WIFI GOT IP

OK
```

Once the above step was completed, the FTDI was disconnected, the board was configured and connections were made from the LM34 to the Mini, and from the Mini to the ESP module.

7. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=bu3qmjpEGCs

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