MIDTERM 1

Student Name: Ron Joshua Recrio
Student #: 5003825419
Student Email: recrio@unlv.nevada.edu
Primary Github address: https://github.com/recrio/submissions
Directory: /Midterms/Midterm1

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 w/ PINS
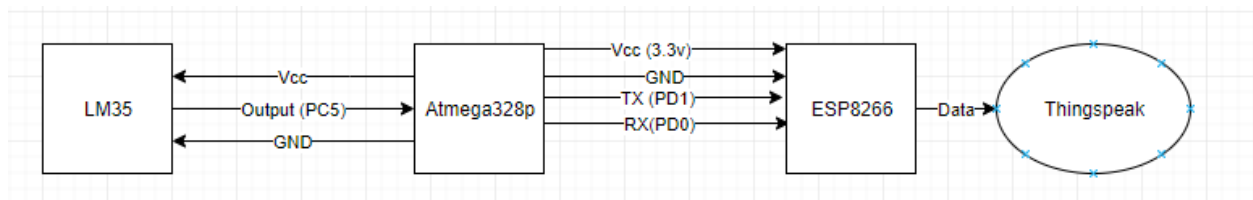
List of Components used
  ATmega328p
  LM35
  ESP8266
  USB to ESP8266 Module
Block diagram with pins used in the Atmega328P

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

```c
#define F_CPU 16000000UL
#define PRESCALAR 1024
#define BAUDRATE 9600
#define BAUD_PRESCALAR (((F_CPU / (BAUDRATE * 16UL))) - 1)
#define ONESEC (0xFFFF - ((F_CPU/PRESCALAR)*1) - 60)

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>

void USART_init(void); // initializes USART settings
void USART_sendChar(char ch); // sends a character
void USART_sendString(char* str); // sends a string
void TIMER_init(void); // initializes timer sequence for interrupts
void ADC_init(void); // initializes ADC settings

volatile int adc_temp;

int main(void)
{
      USART_init(); // initialize USART
      TIMER_init(); // initialize Timer/Interrupt
      ADC_init();

      while (1) // Loop forever
      {
      }
}

ISR (TIMER1_OVF_vect) {
      TCNT1 = ONESEC; // set the timer back
      ADCSRA |= (1<<ADSC); // start conversion
      while ((ADCSRA&(1<<ADIF))==0){} // Wait for conversion
      ADCSRA |= (1<<ADIF); // Clear Interrupt Flag
```

```c
        adc_temp = ADCL; // take in lower bits first
        adc_temp = adc_temp | (ADCH<<8); // take in upper bits
        adc_temp = adc_temp*(9/5) + 32;
        char temp[20]; // buffer
        snprintf(temp,sizeof(temp),"%d\r\n",adc_temp); // print to the buffer
        USART_sendString(temp); // send the temp out
}

void USART_init( void )
{
        UBRR0H = 0; // not needed
        UBRR0L = BAUD_PRESCALAR; // Baud Prescaler
        UCSR0C = _BV(UCSZ01) | _BV(UCSZ00); /* 8-bit data */
        UCSR0B = _BV(RXEN0) | _BV(TXEN0); /* Enable RX and TX */
}

void USART_sendChar(char ch) {
        while (!(UCSR0A & (1<<UDRE0))); // while data reg is not empty: hold
        UDR0 = ch; // place character into reg
}

void USART_sendString(char* str) {
        while ((*str != '\0')) { // while not the end of the string
                while (!(UCSR0A & (1<<UDRE0))); // while data reg is not empty: hold
                USART_sendChar(*str); //take in character to reg
                str++; // next character
        }
}

void TIMER_init(void) {
        TCNT1 = ONESEC; // ONESEC is the number to count up to 0xFFFF for 1 sec delay
        TIMSK1 |= (1 << TOIE0); // Enable Interrupt for Timer1
        sei(); // Enable Global Interrupt
        TCCR1B |= (1<<CS12)|(1<<CS10); // Start timer 1 and set prescaler to 1024
}

void ADC_init(void) {
        ADMUX |= (0<<REFS1)| // Reference Select
        (1<<REFS0)| // Selected AVcc
        (0<<ADLAR)| // Left Adjust Result OFF
        (1<<MUX2) | // Analog Channel Select
        (0<<MUX1) | // 1 0 1
        (1<<MUX0) ; // Channel 5 or PC5
        ADCSRA |= (1<<ADEN)| // Enable ADC
        (0<<ADSC)| // Do not start conversion
        (0<<ADATE)|// Auto Trigger Disabled
        (0<<ADIF)| // Interrupt Flag Cleared
        (0<<ADIE)| // Interrupt Disabled
        (1<<ADPS2)| // ADC Prescaler Select
        (0<<ADPS1)| // Set to
        (1<<ADPS0); // 32
}
```

3.   DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

```c
#define F_CPU 16000000UL
#define PRESCALAR 1024
#define BAUDRATE 115200
#define BAUD_PRESCALAR (((F_CPU / (BAUDRATE * 16UL))) - 1)

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <util/delay.h>

void USART_init(void); // initializes USART settings
void USART_sendChar(char ch); // sends a character
void USART_sendString(char* str); // sends a string
void TIMER_init(void); // initializes timer sequence for interrupts
void ADC_init(void); // initializes ADC settings
void AT_init(void); // initializes AT settings

volatile int adc_temp;

int main(void)
{
        USART_init(); // initialize USART
        AT_init(); // initialize AT setttings
        TIMER_init(); // initialize Timer/Interrupt
        ADC_init();  // initialize ADC

        while (1) // Loop forever
        {
        }
}

ISR (TIMER1_OVF_vect) {
        ADCSRA |= (1<<ADSC); // start conversion
        while ((ADCSRA&(1<<ADIF))==0){} // Wait for conversion
        ADCSRA |= (1<<ADIF); // Clear Interrupt Flag

        adc_temp = ADCL; // take in lower bits first
        adc_temp = adc_temp | (ADCH<<8); // take in upper bits
        adc_temp = adc_temp*(9/5) + 32; // change to fahrenheit
        char temp[20]; // temp buffer
        snprintf(temp,sizeof(temp),"%d\r\n",adc_temp); // print to the buffer
        //USART_sendString(temp);
        USART_sendString("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n"); //connect
to thingspeak
        _delay_ms(3000);
        USART_sendString("AT+CIPSEND=51\r\n"); // send data 51 characters
        _delay_ms(1000);
        USART_sendString("GET /update?key=DUEPKNU9WHKU2GQL&field1="); // update channel
using write key
        USART_sendString(temp); // value to input
        _delay_ms(1000);
        USART_sendString("AT+CIPCLOSE\r\n"); // end of send
        _delay_ms(1000);
        TCNT1 = 0; // set the timer back
}
```

```c
void USART_init( void )
{
        UBRR0H = 0; // not needed
        UBRR0L = 8; // used for 115200
        UCSR0C = _BV(UCSZ01) | _BV(UCSZ00); /* 8-bit data */
        UCSR0B = _BV(RXEN0) | _BV(TXEN0); /* Enable RX and TX */
}

void USART_sendChar(char ch) {
        while (!(UCSR0A & (1<<UDRE0))); // while data reg is not empty: hold
        UDR0 = ch; // place character into reg
}

void USART_sendString(char* str) {
        while ((*str != '\0')) { // while not the end of the string
                while (!(UCSR0A & (1<<UDRE0))); // while data reg is not empty: hold
                USART_sendChar(*str); //take in character to reg
                str++; // next character
        }
}

void TIMER_init(void) {
        TCNT1 = 0; // ONESEC is the number to count up to 0xFFFF for 1 sec delay
        TIMSK1 |= (1 << TOIE0); // Enable Interrupt for Timer1
        sei(); // Enable Global Interrupt
        TCCR1B |= (1<<CS12)|(1<<CS10); // Start timer 1 and set prescaler to 1024
}

void ADC_init(void) {
        ADMUX |= (0<<REFS1)| // Reference Select
                        (1<<REFS0)| // Selected AVcc
                        (0<<ADLAR)| // Left Adjust Result OFF
                        (1<<MUX2) | // Analog Channel Select
                        (0<<MUX1) | // 1 0 1
                        (1<<MUX0) ; // Channel 5 or PC5
        ADCSRA |= (1<<ADEN)| // Enable ADC
                        (0<<ADSC)| // Do not start conversion
                        (0<<ADATE)|// Auto Trigger Disabled
                        (0<<ADIF)| // Interrupt Flag Cleared
                        (0<<ADIE)| // Interrupt Disabled
                        (1<<ADPS2)| // ADC Prescaler Select
                        (0<<ADPS1)| // Set to
                        (1<<ADPS0); // 32
}

void AT_init(void) {
        USART_sendString("AT\r\n"); // Sends AT, expect OK
        _delay_ms(1000);
        USART_sendString("AT+CWMODE=1\r\n"); // Sends mode set to station, expect OK
        _delay_ms(1000);
        USART_sendString("AT+CWLAP\r\n"); // Send command to list Wifi networks, expect
list
        _delay_ms(4000);
        USART_sendString("AT+CWJAP=\"ATwifi\",\"ATpassword\"\r\n"); // Send command to
join guest wifi, expect OK
        _delay_ms(3000);
}
```
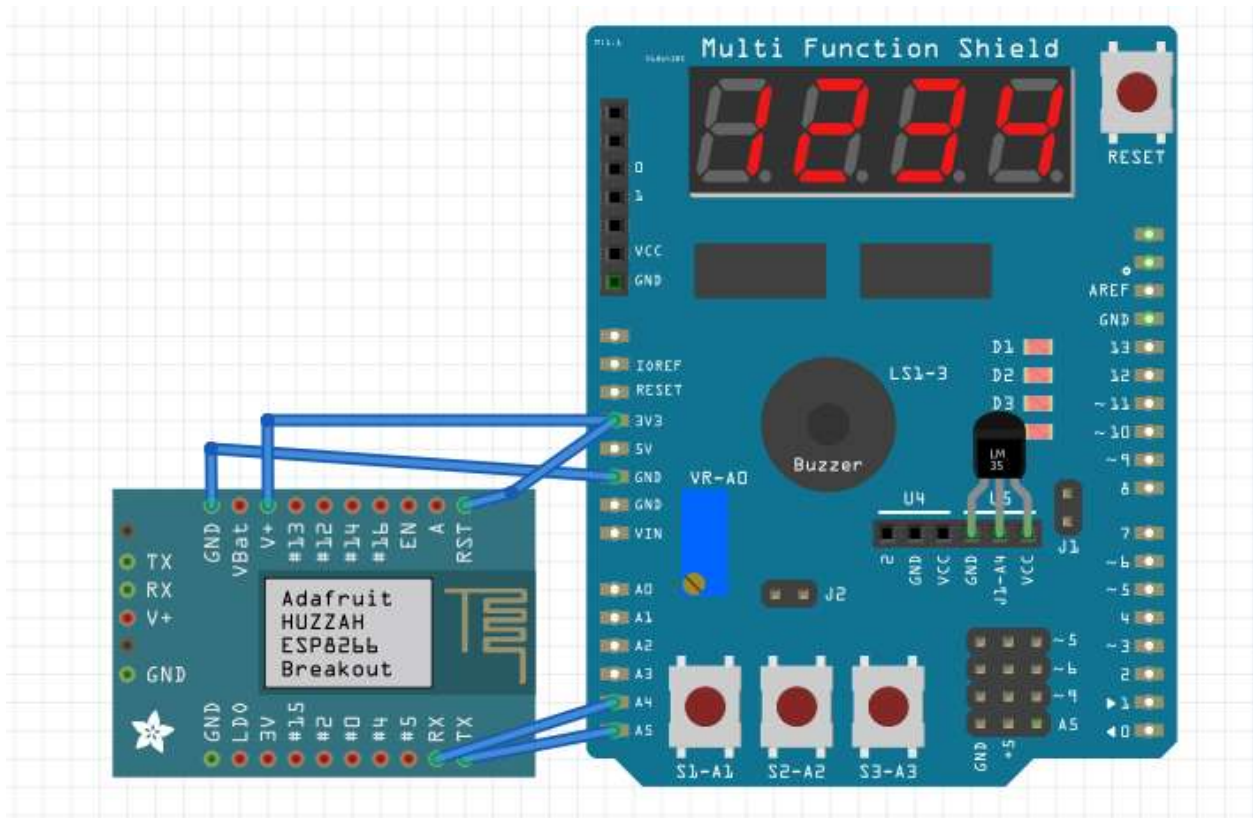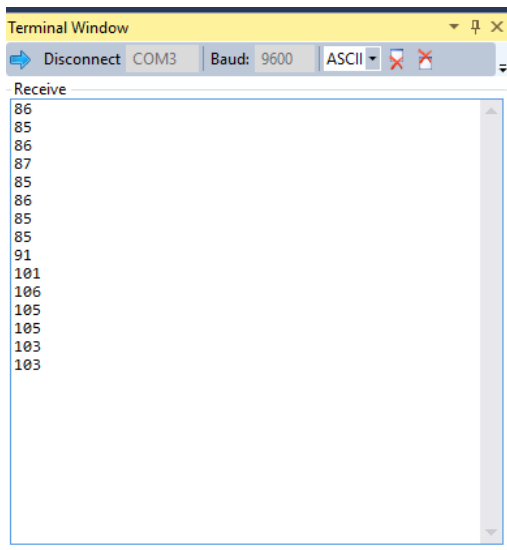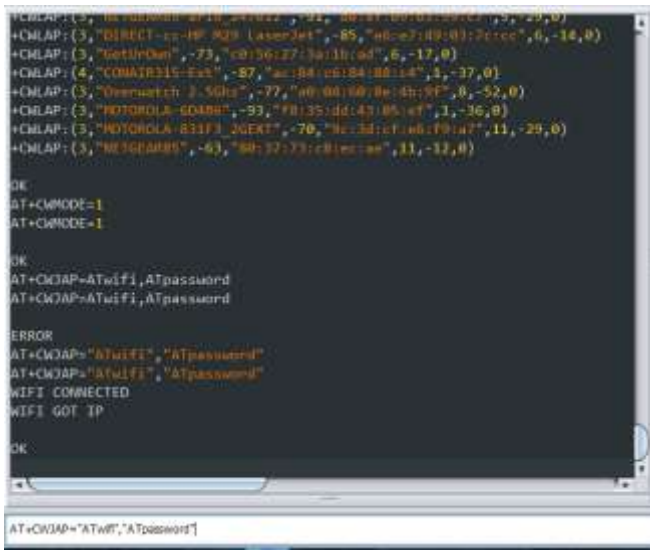
4. SCHEMATICS

*Not exact models

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)
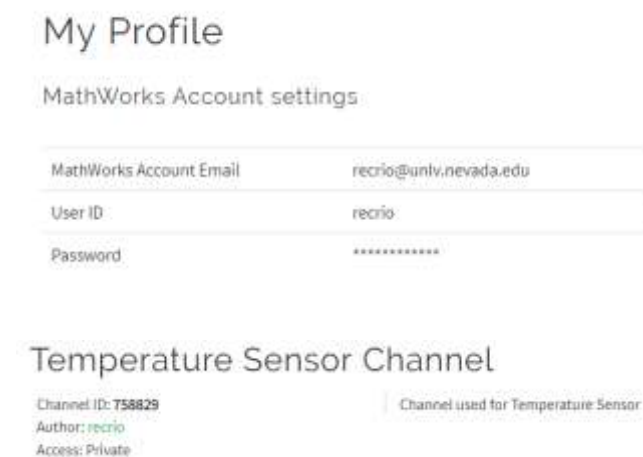
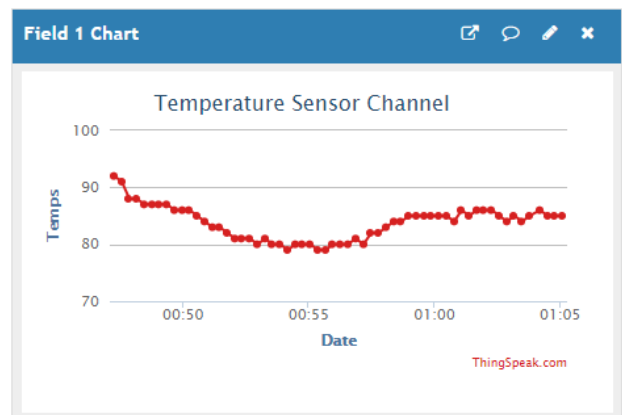1-2.) Terminal Output of LM35 using ADC   3.) AT firmware successfully working on Esplorer

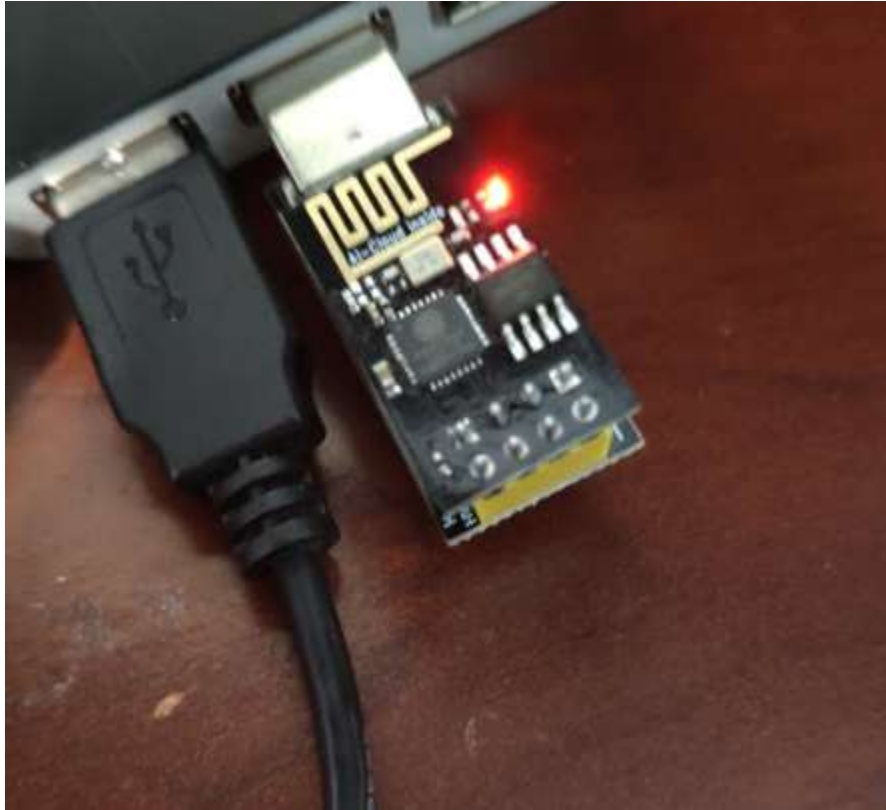4.) Signed up for Thingspeak and channel made   5-6.) Transmit temps to Thingspeak
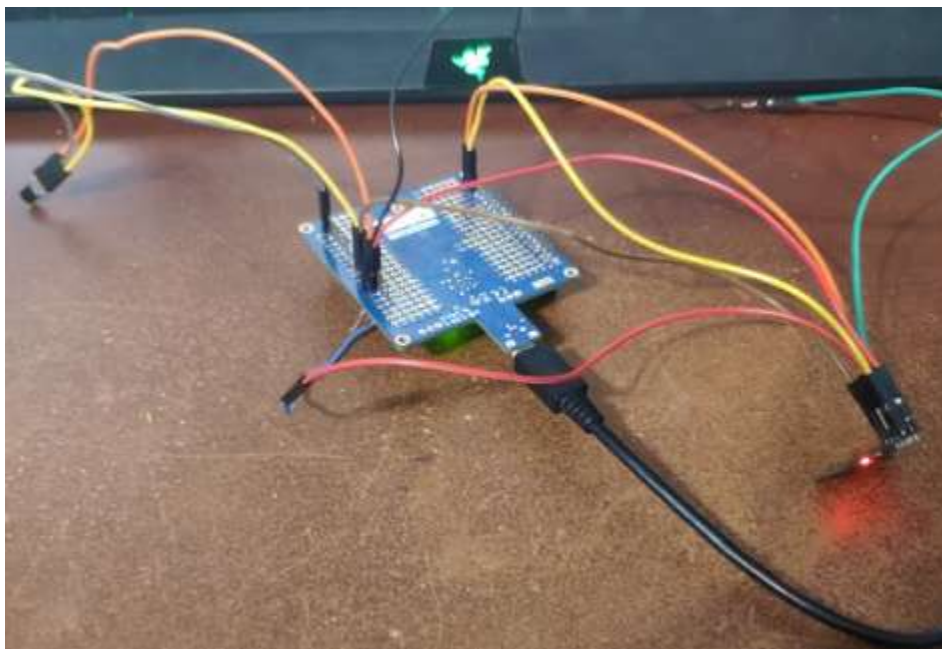
6. SCREENSHOT OF EACH DEMO (BOARD SETUP)
Programming ESP8266:

After Programming:

7.   VIDEO LINKS OF EACH DEMO

https://youtu.be/hEWqzvV_mSQ

8.   GITHUB LINK OF THIS DA

https://github.com/recrio/submissions/tree/master/Midterms/Midterm1

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

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

Ron Joshua Recrio