Trace Stewart

CPE301 – SPRING 2018

Design Assignment X

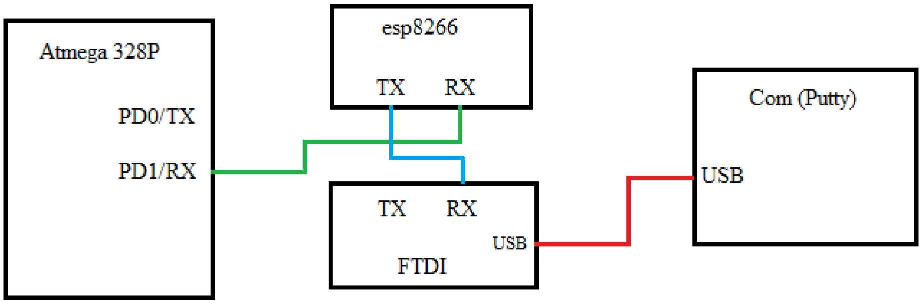
DO NOT REMOVE THIS PAGE DURING SUBMISSION:

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		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

The components used in this project were the Atmega xplained mini, a FTDI chip, the esp8266 wifi chip, and the lm34 temperature sensor/



I have the Atmega xplained mini writing to the esp8266(For some reason It would only work when I connected the RX port of the mini to the RX port of the esp8266) and the esp8266 writing to the FTDI chip which would display what the esp8266 was writing back on putty.

2. INITIAL/DEVELOPED CODE OF TASK 1/A

```
/*
 * DAMidterm.c
 *
 * Created: 3/26/2018 9:04:20 PM
 * Author : trace
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000L
#include <util/delay.h>
#include <stdlib.h>
#define BAUD  9600

volatile int ovrflw; // global variable for keeping track of # of times Timer0 overflows
volatile uint8_t ADCvalue; // Global variable, set to volatile if used with ISR

// functions
void initUART();
void writeChar(unsigned char c);
void writestring(char *c);
void writefloat(float c);

int main(void){

       initUART();          // Initialize UART

       // initialize ADC
       DDRC = 0;                   // Set PORTC as input for adc
       DIDR0 = 0x1;         // Disable digital input on ADC0 pin
```

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


        ADCSRA = 0x87;              // Enable ADC, system clock, 10000111
        ADCSRB = 0x0;        // Free running mode

        // initialize timer0 with starting value of 0, normal mode with no prescaler
        TCNT0 = 0;
        TCCR0A = 0;
        TCCR0B |= 2;

        // enable interrupts
        TIMSK0 |= (1 << TOIE0);            // enable overflow interrupt
        sei();                                    // enable global interrupts

        while (1);

        return 0;
}

void initUART() {
        unsigned int baudrate;

        // Set baud rate:  UBRR = [F_CPU/(16*BAUD)] -1
        baudrate = ((F_CPU/16)/BAUD) - 1;
        UBRR0H = (unsigned char) (baudrate >> 8);
        UBRR0L = (unsigned char) baudrate;

        UCSR0B |= (1 << RXEN0) | (1 << TXEN0);           // Enable receiver and transmitter
        UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00);  // Set data frame: 8 data bits, 1 stop
bit, no parity

}

void writeChar(unsigned char c) {
        UDR0 = c;                   // Display character on serial (i.e., PuTTY) terminal
        _delay_ms(10);              // delay for 200 ms
}

void writestring(char *c){
        unsigned int i = 0;
        while(c[i] != 0)
        writeChar(c[i++]);
}



// this interrupt service routine (ISR) runs whenever an overflow on Timer0 occurs
ISR (TIMER0_OVF_vect) {

        // Variable Declarations
        char output[6];                                        // Output string based on
ADC
        float temperature;                                     // Voltage received by ADC
then edited for Temperature
```

```
       char *AT = "AT \r\n";
       char *CIPMUX = "AT+CIPMUX=0 \r\n";
       char *ATCW = "AT+CWJAP=\"SSID\",\"Password\" \r\n";
       char *CIPSTART = "AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80 \r\n";
       char *CIPSEND = "AT+CIPSEND=44";
       char *SEND_DATA = "GET /update?key=M52FZABUR6UTS03B&field2=";
       char *ENTER = "\r\n";

       if (ovrflw == 7500) {

              ADCSRA |= (1 << ADSC);                   // Start conversion
              while((ADCSRA&(1<<ADIF))==0);     // Wait for conversion to finish

              ADCvalue = ADCH;                  // Only need to read the high value for 8
bit then equation for Fahrenheit
              temperature = (ADCvalue * 5.0 / 256) * 100;     // Temperature
              dtostrf(temperature, 0, 0, output);      // Float to char* conversion

              ovrflw = 0;                       // reinitialize ovrflw

       _delay_ms(200);
       initUART(); // initialize usart
       _delay_ms(500);
       writestring(AT);
       _delay_ms(2000);
       writestring(CIPMUX); // Mux at command
       _delay_ms(5000);
       writestring(ATCW); // Connect to wifi
       _delay_ms(5000);
       writestring(CIPSTART); // Connect to website
       _delay_ms(5000);
       writestring(CIPSEND); // Send characters of get command
       writeChar('\r');
       writeChar('\n');
       _delay_ms(5000);
       writestring(SEND_DATA); // Send get command
       _delay_ms(1000);
       writestring(output); // Send Temperature value
       _delay_ms(500);
       writestring(ENTER); // enter line
       _delay_ms(1000);

       for(int i=0; i < 1000; i++)
       {
       _delay_ms(1000);     // 16 minute loop
       }


       }
       else
       ovrflw++;     // increment ovrflw
```

3. SCHEMATICS

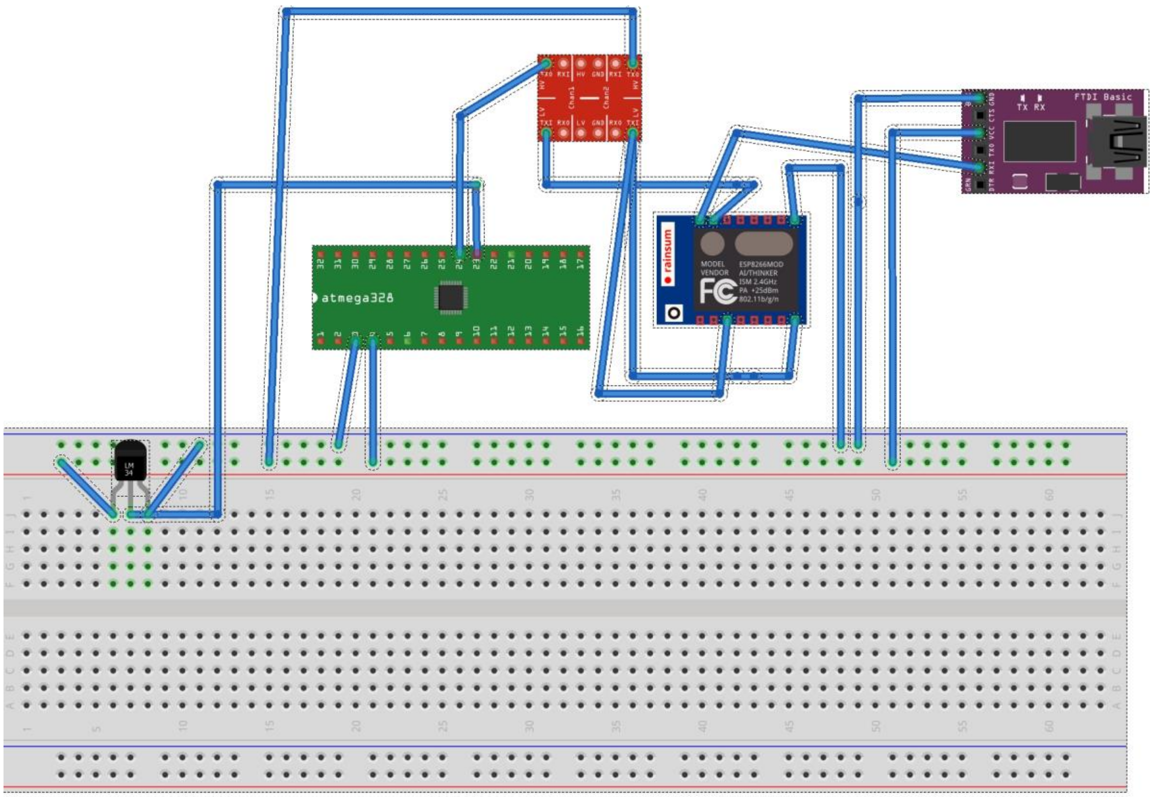
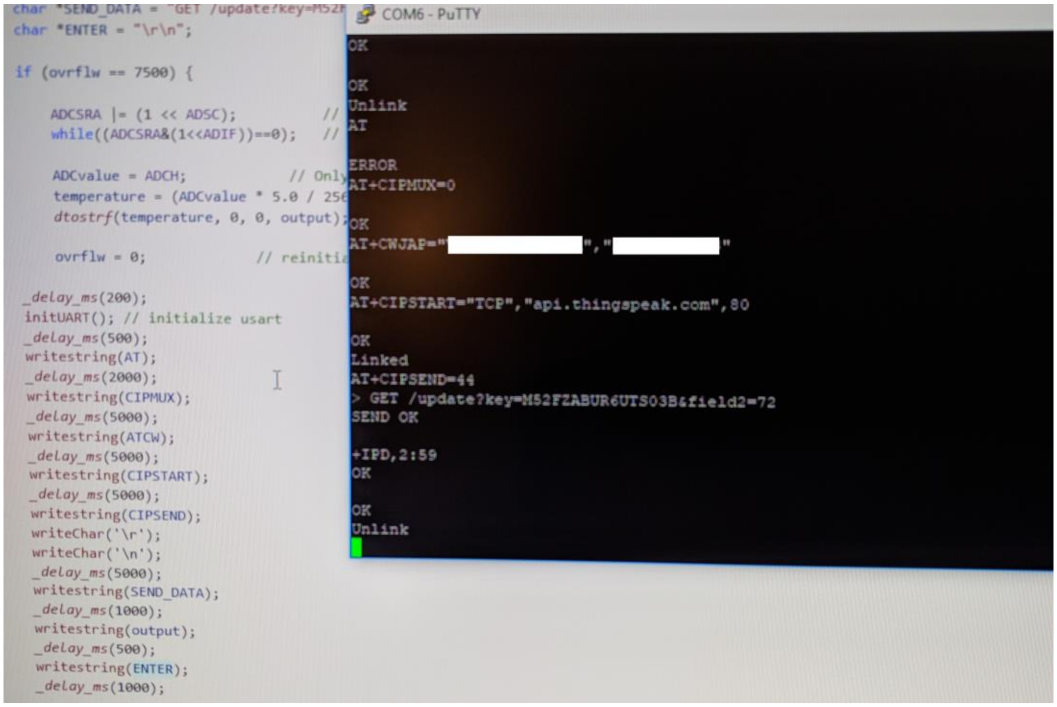


Figure 1: Fritzing Schematic

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

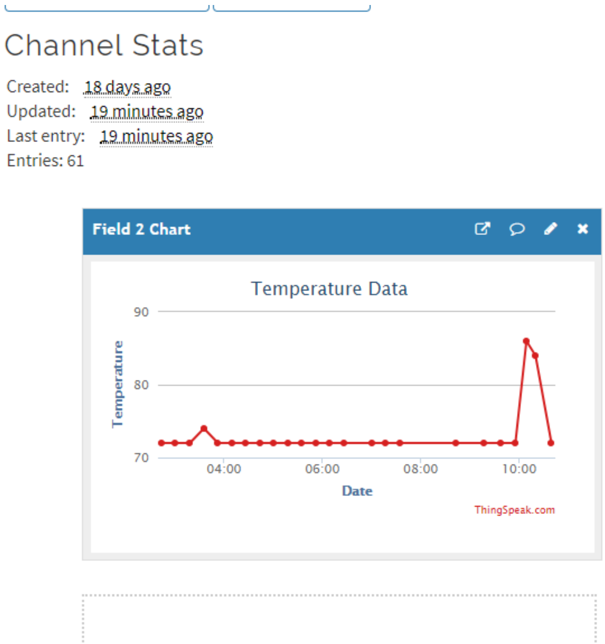


```
char *SEND_DATA = "GET /update?key=M52F
char *ENTER = "\r\n";

if (ovrflw == 7500) {

    ADCSRA |= (1 << ADSC);            //
    while((ADCSRA&(1<<ADIF))==0);    //

    ADCvalue = ADCH;                // Only
    temperature = (ADCvalue * 5.0 / 256
    dtostrf(temperature, 0, 0, output);

    ovrflw = 0;                 // reinitia

 _delay_ms(200);
 initUART(); // initialize usart
 _delay_ms(500);
 writestring(AT);
 _delay_ms(2000);
 writestring(CIPMUX);
 _delay_ms(5000);
 writestring(ATCW);
 _delay_ms(5000);
 writestring(CIPSTART);
 _delay_ms(5000);
 writestring(CIPSEND);
 writeChar('\r');
 writeChar('\n');
 _delay_ms(5000);
 writestring(SEND_DATA);
 _delay_ms(1000);
 writestring(output);
 _delay_ms(500);
 writestring(ENTER);
 _delay_ms(1000);
```

COM6 - PuTTY

```
OK

OK
Unlink
AT

ERROR
AT+CIPMUX=0

OK
AT+CWJAP="[redacted]","[redacted]"

OK
AT+CIPSTART="TCP","api.thingspeak.com",80

OK
Linked
AT+CIPSEND=44
> GET /update?key=M52FZABUR6UTS03B&field2=72
SEND OK

+IPD,2:59
OK

OK
Unlink
```

Figure 2: Screenshot of putty terminal



Channel Stats

Created: 18 days ago

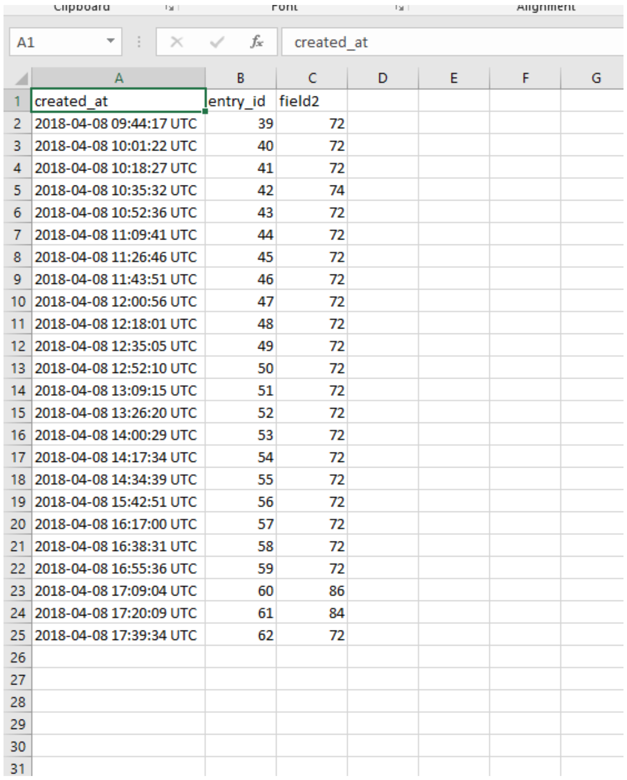
Updated: 19 minutes ago

Last entry: 19 minutes ago

Entries: 61

Figure 3: Temperature data graph on Thinkspeak

The temperature was very consistent so for the last couple of temperature reading I held onto the sensor to get some variance in the temperature as you can see with the spike at the end.



	A	B	C
1	created_at	entry_id	field2
2	2018-04-08 09:44:17 UTC	39	72
3	2018-04-08 10:01:22 UTC	40	72
4	2018-04-08 10:18:27 UTC	41	72
5	2018-04-08 10:35:32 UTC	42	74
6	2018-04-08 10:52:36 UTC	43	72
7	2018-04-08 11:09:41 UTC	44	72
8	2018-04-08 11:26:46 UTC	45	72
9	2018-04-08 11:43:51 UTC	46	72
10	2018-04-08 12:00:56 UTC	47	72
11	2018-04-08 12:18:01 UTC	48	72
12	2018-04-08 12:35:05 UTC	49	72
13	2018-04-08 12:52:10 UTC	50	72
14	2018-04-08 13:09:15 UTC	51	72
15	2018-04-08 13:26:20 UTC	52	72
16	2018-04-08 14:00:29 UTC	53	72
17	2018-04-08 14:17:34 UTC	54	72
18	2018-04-08 14:34:39 UTC	55	72
19	2018-04-08 15:42:51 UTC	56	72
20	2018-04-08 16:17:00 UTC	57	72
21	2018-04-08 16:38:31 UTC	58	72
22	2018-04-08 16:55:36 UTC	59	72
23	2018-04-08 17:09:04 UTC	60	86
24	2018-04-08 17:20:09 UTC	61	84
25	2018-04-08 17:39:34 UTC	62	72

Figure 4: Data exported from thinkspeak channel

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

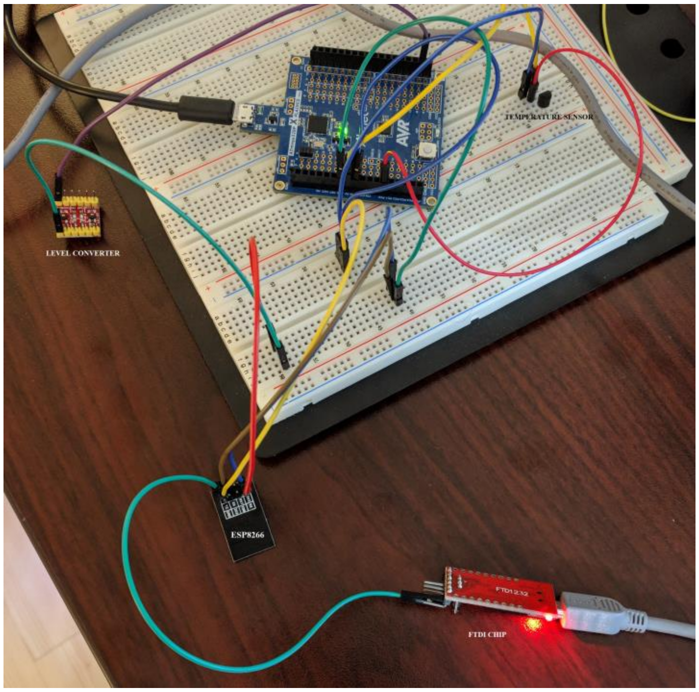


Figure 5: Board Setup

6. VIDEO LINKS OF EACH DEMO

None.

7. GITHUB LINK OF THIS DA

https://github.com/TraceStewart/epc103gnirps8102vlnu/tree/master/Midterm

ThinkSpeak Channel: https://thingspeak.com/channels/454833

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

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