MT 1

By: Bruce

Primary Github address: https://github.com/dcon99/CMX2345.git/

Directory: All Design Assignments/MT1

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**

-Atmega328P

-ESP8266-01

-FTD1232

-LM35DZ Temperature Sensor

-Breadboard

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

/\*

\* Midterm1.c

\*

\* Created: 10/27/2019 9:55:47 PM

\* Author : bruce

\*/

/\*

This program uses the Analog to Digital Converter (ADC) of the ATmega328p along with

the USART and Timer1 to convert the analog signal from an LM35DZ (Temperature Sensor in Celsius)

into a digital one, convert that data into a readable temperature value, and then transmit

said data to a channel through TeamSpeak via an ESP8266-01 Wi-Fi Module every 15-20 seconds.

\*/

#define *F\_CPU* 8000000UL //Sets the clock speed of the MCU

#define BAUD 9600 //Sets the baud rate for the UART to transmit

#define BAUDRATE ((*F\_CPU*) / (BAUD \* 8UL)-1) // Set Baud Rate Value for UBRR

/////////////////////////////////////////////////////////////////////////////////////////////////////

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdlib.h>

#include <stdio.h>

/////////////////////////////////////////////////////////////////////////////////////////////////////

char String[]="The Current Temperature Is: "; //Character array for string

char LineBreak[]="\r\n"; //New line and return string array for a neater transmission

//unsigned int number = 65; //Int declared for UART

//double n = 1.3689; //Double/Float declared for UART

char String\_num[]; //String array to hold an int after converting to string for int

char String\_flt[]; //String array to hold a double value after converting to string

volatile *uint8\_t* time\_ovf; //Integer to hold the amount of times the timer overflows

float temperature; //Float variable used to hold the measured analog value from the LM35DZ

char Degrees[]="° F"; //Used to make the output temperature value readable as Fahrenheit

char CIPMUX[]="AT+CIPMUX=1"; //AT command string for the ESP8266 for setting up a Single or Multi-line IP connection (value of '0' for single)

char CWJAP[]="AT+CWJAP=\""; //AT command string for the ESP8266 used to scan for available access points

char SSID[]="\COCOA9500\","; //A character array used to hold the name of the access point the user wishes to connect to

char Password[]="\"Arduc4nt3@\""; //A character array used to hold the password for the access point the user wishes to connect to.

char CIPSTART[]="AT+CIPSTART=4,\"TCP\",\"184.106.153.149\",80"; //AT command string for the ESP8266 used to start a connection with ThingSpeak in TCP mode

char CIPSEND[]="AT+CIPSEND=4,77"; //AT command used to send a number of bits through the ESP8266 to ThingSpeak

char TS\_COMMAND[]="GET /update?api\_key="; //Character array used to the first part of the API key command for a ThingSpeak channel

char API\_KEY[]="C3ULAMG0GEL8PX8E&field1="; //Character array that holds the ThingSpeak channel API key

char CIPCLOSE[]="AT+CIPCLOSE"; //AT command string to close the connection with ThingSpeak after data transmission

/////////////////////////////////////////////////////////////////////////////////////////////////////

//A function used to initialize Timer1 of the ATmega328P with a prescale of 64

void timer1\_init()

{

//Set up timer with a prescale of 64

TCCR1A |= (0<<COM1A1) | (0<<COM1A0);

TCCR1B |= (1<<CS11)|(1 << CS10);

//Initialize counter

TCNT1 = 0;

//Enable overflow interrupt

TIMSK1 |= (1 << TOIE1);

//Initialize overflow counter variable

time\_ovf = 0;

}

//Used to initialize the UART of the ATmega328P with a baud rate of 9600 and enables global interrupts.

void initialize\_UART(){

UBRR0H = (*uint8\_t*)(BAUDRATE>>8);

UBRR0L = (*uint8\_t*)(BAUDRATE);

UCSR0B = (1<<RXEN0)|(1<<TXEN0);

UCSR0C = ((1<<UCSZ00)|(1<<UCSZ01));

sei();

}

//Function used to send one character (8-bits) at a time through USART

void USART\_send( unsigned char data){

while(!(UCSR0A & (1<<UDRE0)));

UDR0 = data;

}

//Function used to send converted numbers through a string array through USART

void USART\_send\_num( unsigned int data){

while(!(UCSR0A & (1<<UDRE0)));

UDR0 = data;

}

//Function used to receive data from USART

unsigned char USART\_receive(void){

while(!(UCSR0A & (1<<RXC0)));

return UDR0;

}

//Function used to transmit an array of characters (string) through USART one character at a time.

void USART\_putstring(char\* StringPtr){

while(\*StringPtr != 0x00){

USART\_send(\*StringPtr);

StringPtr++;}

}

//Function used to transmit an array of characters of an integer converted into a string through USART one character at a time.

void USART\_putnumber(char\* String\_num, unsigned int temperature){

String\_num = (*utoa*(temperature,String\_num,10)); //Converts an int to a string

while(\*String\_num != 0x00){

USART\_send\_num(\*String\_num);

String\_num++;}

}

//Function used to transmit an array of characters of an double converted into a string through USART one character at a time.

void USART\_putflt(char\* String\_flt, float temperature){

String\_flt = *dtostrf*(temperature,0,2,String\_flt); //Converts a double to a string

while(\*String\_flt != 0x00){

USART\_send\_num(\*String\_flt);

String\_flt++;}

}

//Function used to initialize the ADC with an adc-prescaler of 64 and using a reference voltage of 5V

void ADC\_init(){

ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(0<<ADPS0)); //Prescaler at 64 so we have an 125Khz clock source

ADMUX |= (0<<REFS1)|( 1 << REFS0 );

}

//Interrupt Service Routine used for Timer1 Overflow.

ISR(TIMER1\_OVF\_vect)

{

//Keep track of number of overflows

time\_ovf++;

}

/////////////////////////////////////////////////////////////////////////////////////////////////////

//This function is used to start a single-conversion from an ADC of the user's choice and store the converted value in the temperature variable.

void ADC\_Read( *uint8\_t* channel )

{

// Select the ADC channel to be read.

ADMUX |= channel;

// Turn on the ADC.

ADCSRA |= ( 1 << ADEN );

// Start the conversion.

ADCSRA |= ( 1 << ADSC );

while( ADCSRA & ( 1 << ADIF ) );

temperature= ADC; //read upper 8bits

temperature = ((temperature/2)\*1.8)+32; //Convert converted value to Fahrenheit.

}

/\*

//Optional function if the average of 10 temperature readings is to be sent to ThingSpeak

void average\_temp(){

int temp\_array[10];

int average\_sum = 0;

for(int i=0;i<10;i++){

ADC\_Read(0);

temp\_array[i] = temperature;

temperature = 0;

}

for(int n=0;n<10;n++){

average\_sum += temp\_array[n];

}

average\_sum = average\_sum/10;

temperature = ((average\_sum/2)\*1.8)+32;

average\_sum = 0;

}

\*/

int main(void)

{

initialize\_UART(); //Initializes UART

timer1\_init(); //Initializes Timer1

ADC\_init(); //Initialize ADC

ADC\_Read(0); //Do one conversion (slowest conversion) for a table conversion process during the main loop.

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//ESP8266-01 Initialization

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

USART\_putstring(CIPMUX);

USART\_putstring(LineBreak);

*\_delay\_ms*(500);

USART\_putstring(CWJAP);

USART\_putstring(SSID);

USART\_putstring(Password);

USART\_putstring(LineBreak);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////

while (1)

{

if(time\_ovf >= 130){ //Uses increments of 4 for a timer1 overflow interrupt of roughly 1 second. (130 is roughly 33 seconds)

ADC\_Read(0); //Using ADC 0

USART\_putstring(String);

USART\_putflt(String\_num, temperature);

USART\_putstring(Degrees);

USART\_putstring(LineBreak);

USART\_putstring(LineBreak);

*\_delay\_ms*(2000);

USART\_putstring(CIPSTART);

USART\_putstring(LineBreak);

*\_delay\_ms*(10000);

USART\_putstring(CIPSEND);

USART\_putstring(LineBreak);

*\_delay\_ms*(2000);

USART\_putstring(TS\_COMMAND);

USART\_putstring(API\_KEY);

USART\_putflt(String\_num,temperature);

USART\_putstring(LineBreak);

*\_delay\_ms*(20000);

USART\_putstring(TS\_COMMAND);

USART\_putstring(API\_KEY);

USART\_putflt(String\_num,temperature);

USART\_putstring(LineBreak);

*\_delay\_ms*(2000);

USART\_putstring(CIPCLOSE);

USART\_putstring(LineBreak);

*\_delay\_ms*(2000);

TCNT1 = 0;

time\_ovf = 0;

}

else;

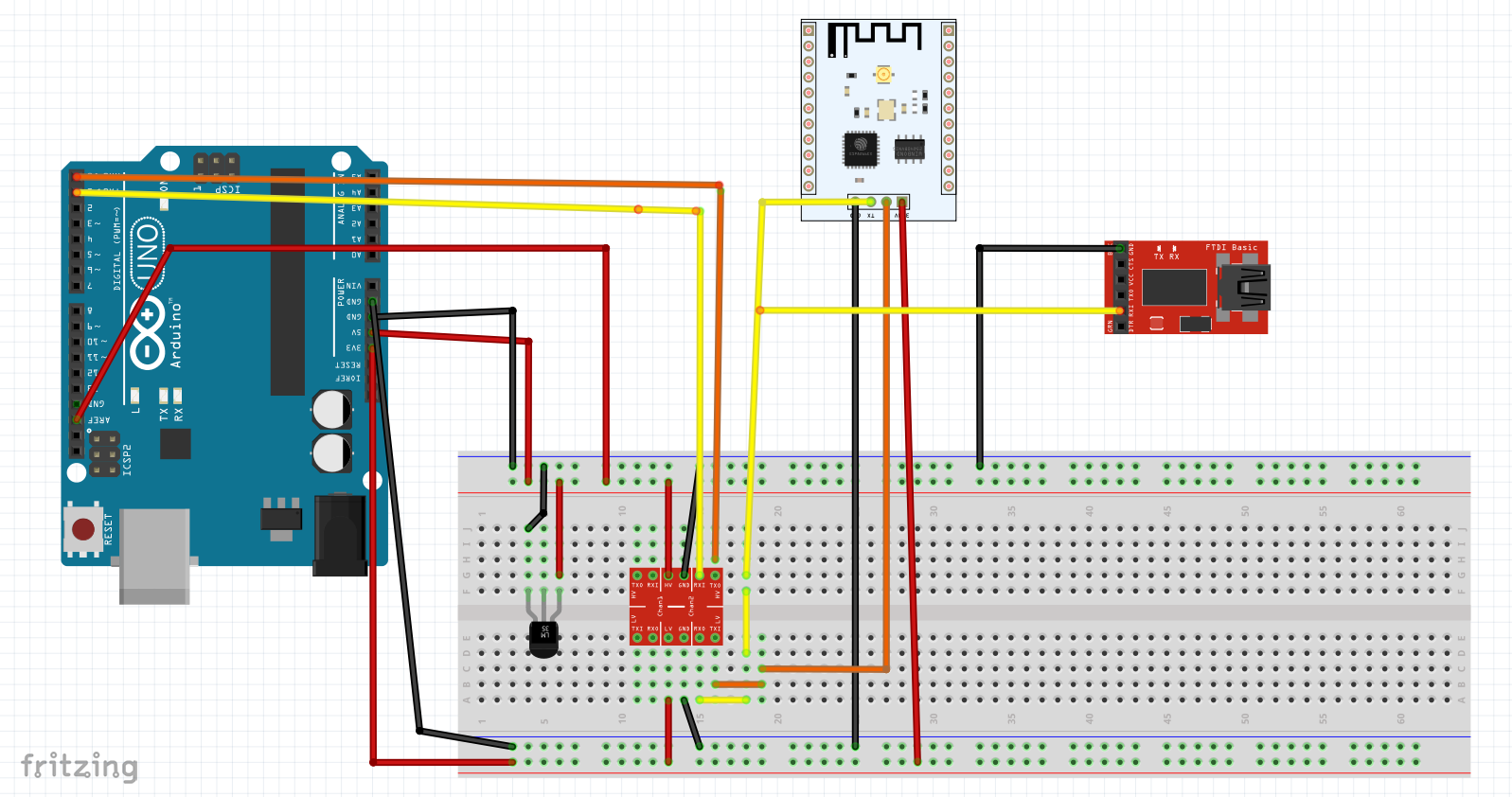
}

}

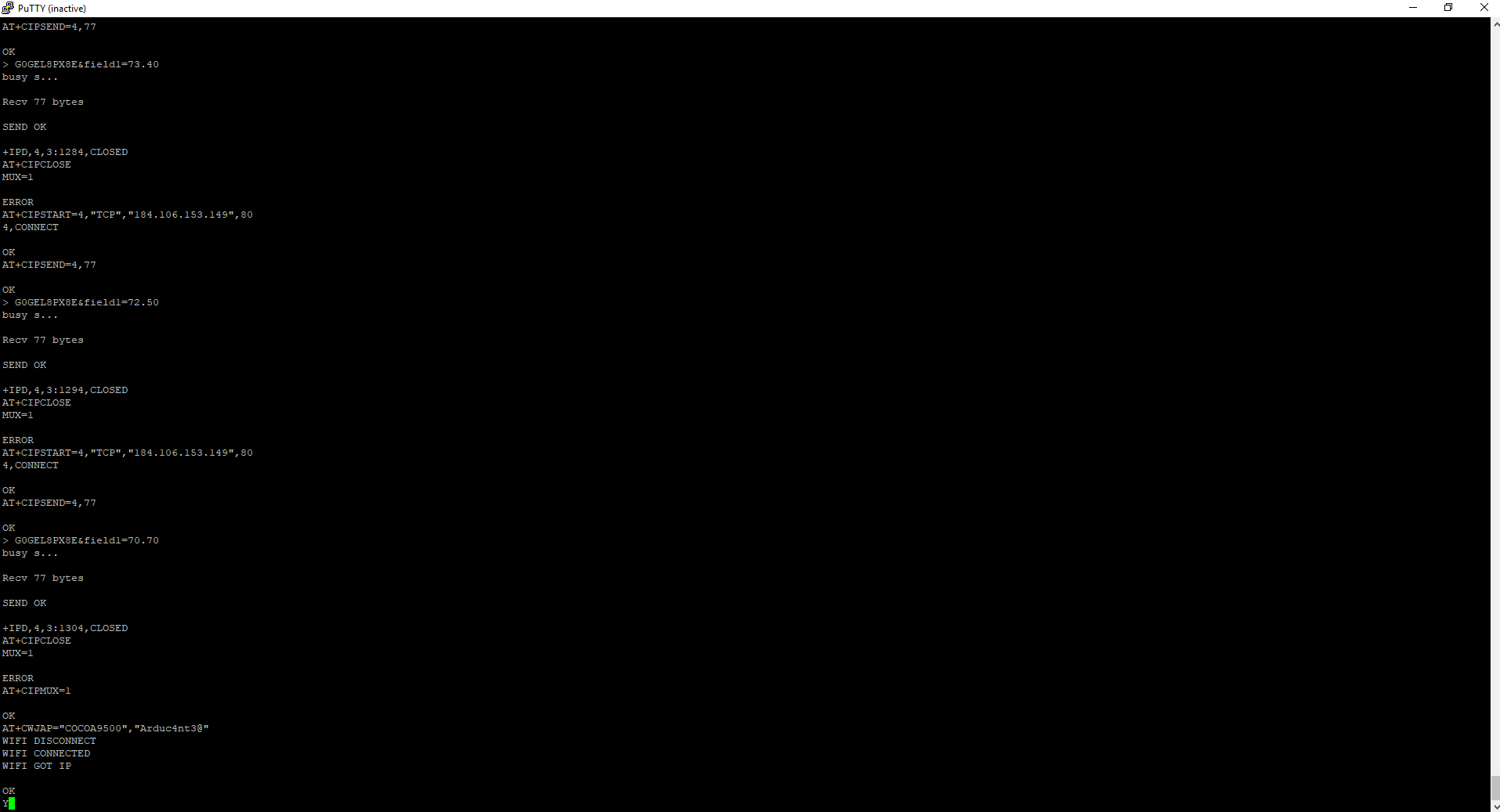
1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

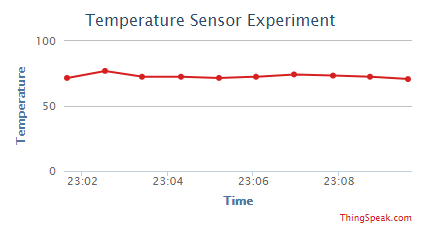
N/A

1. **SCHEMATICS**

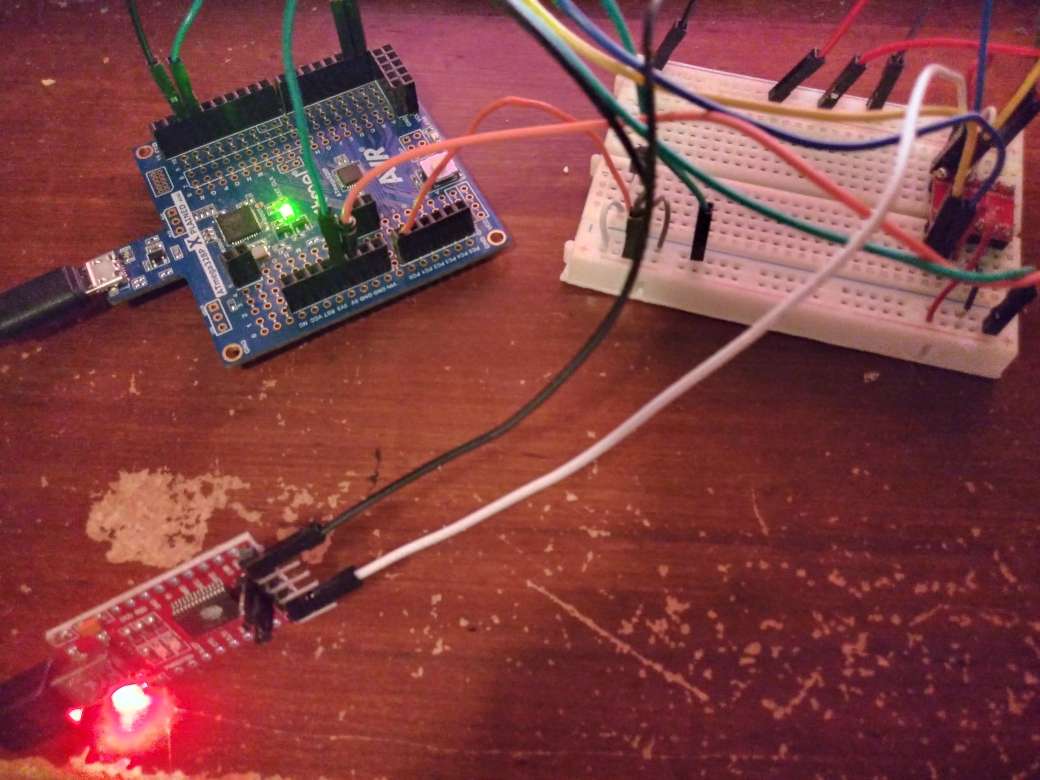


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

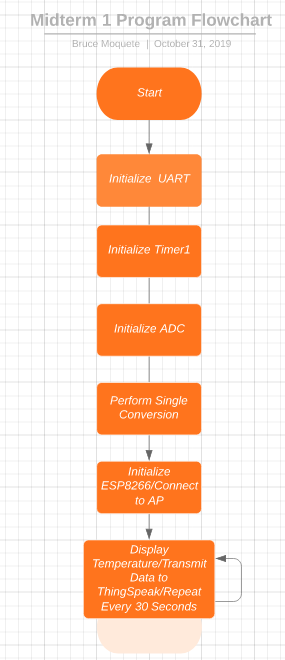




1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**







1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/VQzcMPmQWz0

1. **GITHUB LINK OF THIS DA**

https://github.com/dcon99/CMX2345.git/

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Bruce