Nick Harris

CPE 301

Dr. Venki

Mid-Term 1

/\*

\* Harrin2\_Midterm.c

\*ThingSpeak Channel ID: 896249

\* Created: 10/26/2019 2:32:33 PM

\* Author : Nick Harris

\*/

#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>

#include <string.h>

char string\_num[]; //Holds a number after string conversion

char string\_float[]; //holds a decimal value after string conversion

float temp; //Holds the measurement of the temp sensor

char string\_deg[] = "°F"; //temperature will be measured in Fahrenheit

char string[] = "The Temperature is: "; //string to announce temp

char linebreak[] = "/r/n"; //end of line character

volatile *uint8\_t* time\_ovf; //holds number of overflows of timer

char cipmux[] = "at+cipmux = 1"; //string for esp to set up single or multi line connection

char cwjap[] = "at+cwjap =\""; //string for esp to look for access points

char passwrd[] = "\"Your Password""; //holds password for access point

char ssid[] = "\Your ssid\","; //holds id/name of access point

char api\_key[]="6QDAP0MBCBB5LTE8&field1="; //holds thinkspeak channel key

char ts\_command[]="GET /update?api\_key="; //update key

char cipstart[]="AT+CIPSTART=4,\"TCP\",\"184.106.153.149\",80"; //string to strart communication with thinkspeak

char cipclose[]="AT+CIPCLOSE"; //string to close communication with thinkspeak

char cipsend[]="AT+CIPSEND=4,77"; //string command to sends bits to thinkspeak

//Initializes timer1

void init\_TIMER1()

{

//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 with a baud rate of 9600 and enables global interrupts.

void init\_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 at a time through USART

void USART\_send( unsigned char 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;

}

//sends converted string numbers through usart

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

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

UDR0 = data;

}

//function sends string through usart

void usart\_putstring(char\* StringPtr){

while(\*StringPtr != 0x00){

USART\_send(\*StringPtr);

StringPtr++;}

}

//converts a string integer and sends through usart

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 send an array of characters of converted floating point value into a string through USART

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 init\_adc(){

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

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

}

//function starts convertion of adc and stores it to temp

void read\_adc( *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.

}

//Interrupt Service Routine used for Timer1 Overflow.

ISR(TIMER1\_OVF\_vect)

{

//Keep track of number of overflows

time\_ovf++;

}

ISR(ADC\_vect)

{

PORTD = ADCL;

PORTB = ADCH;

ADCSRA|=(1<<ADSC); //start conversion

while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish

ADCSRA |= (1<<ADIF);

int a = ADCL;

a = a | (ADCH<<8);

a -= 289;

*itoa*(a, buffer, 10);

USART\_putstring(buffer);

USART\_putstring("\n");

*\_delay\_ms*(1000);

}

int main(void)

{

init\_TIMER1();

init\_UART();

init\_adc();

read\_adc();

USART\_putstring(cipmux);

USART\_putstring(linebreak);

*\_delay\_ms*(1000);

USART\_putstring(cwjap);

USART\_putstring(ssid);

USART\_putstring(passwrd);

USART\_putstring(linebreak);

/\*ADCSRA= 0x86; //make ADC enable and select ck/64

ADMUX= 0xC8; //1.1V Vref, temp, right-justified, internal temp. sensor

DDRB = (1<<5);

DDRD = 0x00;

ADCSRA |= (1<<ADSC);

sei();\*/

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, temp);

USART\_putstring(string\_deg);

USART\_putstring(linebreak);

USART\_putstring(linebreak);

*\_delay\_ms*(2000);

USART\_putstring(cipstart);

USART\_putstring(linebreak);

*\_delay\_ms*(20000);

USART\_putstring(cipsend);

USART\_putstring(linebreak);

*\_delay\_ms*(3000);

USART\_putstring(ts\_command);

USART\_putstring(api\_key);

USART\_putflt(string\_num,temp);

USART\_putstring(linebreak);

*\_delay\_ms*(30000);

USART\_putstring(ts\_command);

USART\_putstring(api\_key);

USART\_putflt(string\_num,temp);

USART\_putstring(linebreak);

*\_delay\_ms*(3000);

USART\_putstring(cipsend);

USART\_putstring(linebreak);

*\_delay\_ms*(3000);

TCNT1 = 0;

time\_ovf = 0;

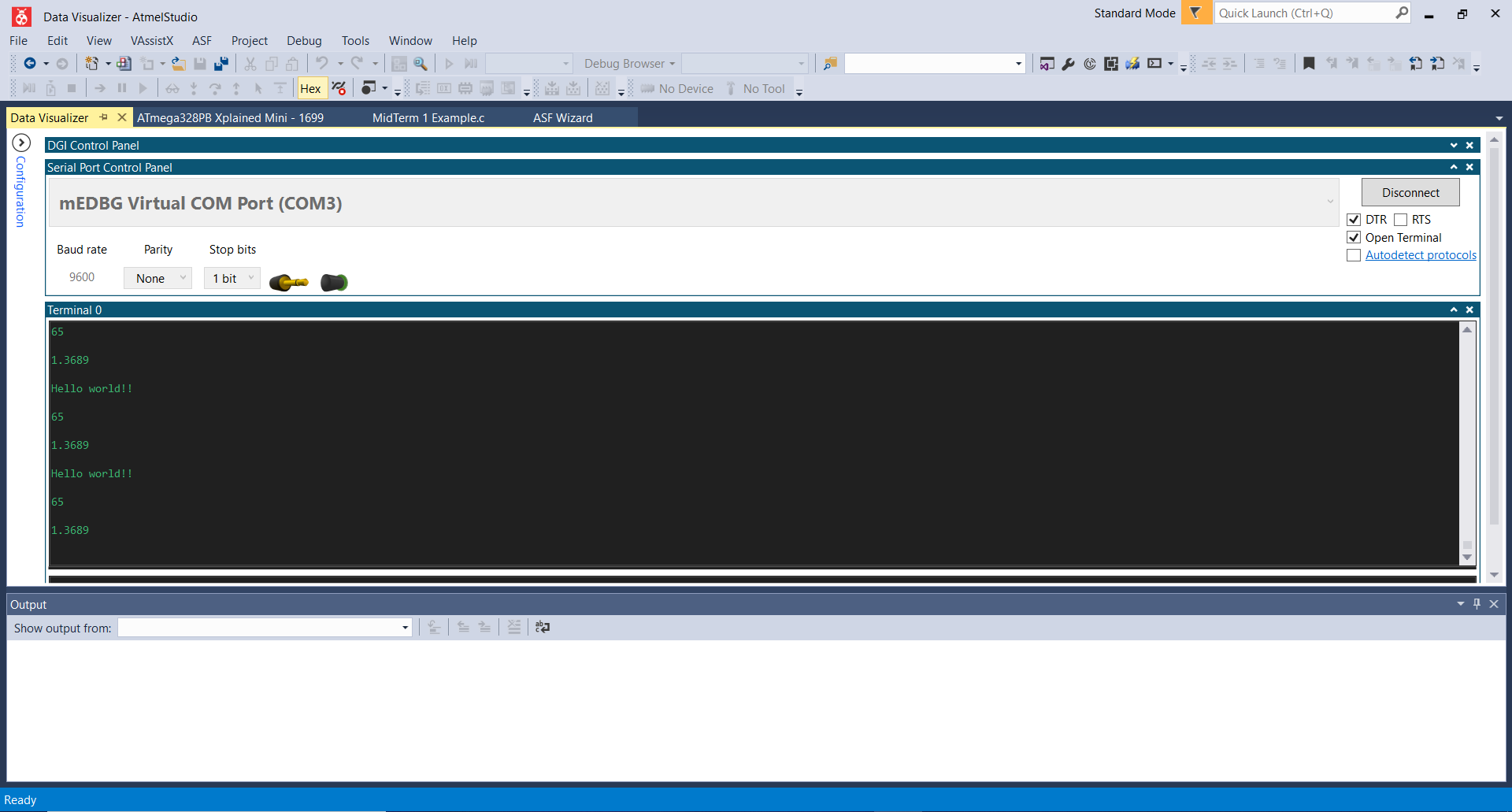
}

else;

}

return 0;

}



API Read key: 694KFK5IJ3OLS1KD

Read Channel: GET <https://api.thingspeak.com/channels/896249/feeds.json?api_key=694KFK5IJ3OLS1KD&results=2>

