CPE301 – SPRING 2019

MIDTERM 1

Student Name: Daniel Senda
Student #: 5002362633
Student Email: sendad1@unlv.nevada.edu
Primary Github address: https://github.com/dsenda/Smiles
Directory: 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/DA, 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:
Atmel Studio 7
ATmega328P Xplained Mini
FTDI chip
ESP8266 ESP-01 Serial Wireless WiFi Transceiver Receiver Module
OPEN SMART USB to ESP8266 ESP 01 Wi Fi Adapter Module
Breadboard
Jumper wires

Block diagram about the transmission connections:

2. INITIAL CODE OF TASK DA3a

```
// c_code_temperature_on_serial2usb.c
// Daniel Senda
// Monitors LM34 to display temperature on serial terminal every 1s.
// Uses a timer with interrupt for the 1s delay.

#define       F_CPU 16000000UL     // Needed Definitions.
#define       BAUD_RATE 9600
#include      <avr/io.h>           // Needed Libraries.
#include      <util/delay.h>
#include      <avr/interrupt.h>

volatile int O_F = 0;              // Used for overflow count.
int num = 0;                       // Used for while loops.

void usart_init ();
void usart_send (unsigned char ch);
void initiate_timer()
{
       TCCR0A = 0b10;              // Sets CTC mode.
       TCCR0B = 0b00000101;        // Sets 1024 pre-scaler.
       OCR0A = 255;                // Sets output compare register A to 255.
       TIMSK0 |= (1 << OCIE0A);    // Enables output compare match interrupt.
       sei();                      // Enables global interrupts.
       TCNT0 = 0;                  // Resets timer.
}
ISR(TIMER0_COMPA_vect)             // Keeps count of overflows.
{
       O_F++;                      // Increments overflow count.
}

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)|                  // ADC4 (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 Pre-scaler Select Bits.
	(0<<ADPS1)|
	(1<<ADPS0);
	initiate_timer();           // Function that initiates timer.
	while (1)
	{
		ADCSRA|=(1<<ADSC);                  // Starts conversion.
		while((ADCSRA&(1<<ADIF))==0);       // Waits for conversion to finish.
		ADCSRA |= (1<<ADIF);
		num = 0;                            // Resets num count.
		TCNT0 = 0;                          // Resets timer.
		while (num <= 0)
		{
			if (O_F >= 61)
			{
				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');
				num++;                      // Increments num to exit while loop.
				O_F = 0;                    // Resets overflow count.
			}
		}
	}
	return 0;
}

void usart_init (void)                  // Initializes USART.
{
	UCSR0B = (1<<TXEN0);
	UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);
	UBRR0L = F_CPU/16/BAUD_RATE-1;
}

void usart_send (unsigned char ch)      // Transmits characters to computer.
{
	while (! (UCSR0A & (1<<UDRE0)));    // Waits until UDR0 is empty.
	UDR0 = ch;                          // Transmit ch.
}
```

```
void usart_print(char* str)              // Prints characters on computer.
{
       int i = 0;
       while(str[i] != 0)
       {
              usart_send(str[i]);
       }
}
```

3. DEVELOPED/MODIFIED CODE FOR MIDTERM1

The following is the code that accomplishes the required tasks for the Midterm1 assignment.

```
// Midterm1_c_code.c
// Daniel Senda
// This code reads temperature data from the sensor and sends this data to
// the Thingspeak server to create a temperature vs time graph.

#define       F_CPU 16000000UL     // Defines frequency of CPU.
#define       BAUD_RATE 9600       // Sets BAUD rate.
#include      <avr/io.h>           // Included libraries.
#include      <util/delay.h>
#include      <avr/interrupt.h>

volatile int O_F = 0;              // Used for overflow count.
int num = 0;                       // Used for while loops.
int tempf = 0;                     // Used for temperature data.
char outs[256];                    // Used for sending USART commands.

void usart_init (void);            // Function declarations.
void initiate_timer(void);
void at_cmd_init (void);
void usart_send (char *ch);

int main (void)
{
       usart_init ();       // Initializes USART communication
                            // Sets up and enables 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 (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 Pre-scaler Select Bits.
       (0<<ADPS1)|
       (1<<ADPS0);
       initiate_timer();    // Function that initiates timer.
       while (1)
       {
              ADCSRA|=(1<<ADSC);                // Starts conversion.
              while((ADCSRA&(1<<ADIF))==0);     // Waits for conversion to finish.
              ADCSRA |= (1<<ADIF);
              sei();                            // Enables global interrupts.
```

```
		num = 0;				// Resets num count.
		O_F = 0;				// Resets overflow count.
		TCNT0 = 0;				// Resets timer.
		while (num <= 0)
		{
			if (O_F >= 61)
			{
				cli();			// Disables global interrupts.
				at_cmd_init ();		// Initializes AT commands.
				tempf = ADCL;		// Records temp sensor data.
				tempf = tempf | (ADCH<<8);
				tempf = (tempf/1024.0) * 5000/10;
				snprintf(outs,sizeof(outs),"GET
https://api.thingspeak.com/update?api_key=QJIVD22Q8N49KTCH&field1=%3d\r\n", tempf);
				usart_send (outs);
				_delay_ms(5000);
				num++;			// Increments num to exit while loop.
			}
		}
	}
	return 0;
}

ISR(TIMER0_COMPA_vect)			// Keeps count of overflows.
{
	O_F++;				// Increments overflow count.
}

void initiate_timer(void)		// Initializes timer.
{
	TCCR0A = 0b10;			// Sets CTC mode.
	TCCR0B = 0b00000101;		// Sets 1024 pre-scaler.
	OCR0A = 255;			// Sets output compare register A to 255.
	TIMSK0 |= (1 << OCIE0A);	// Enables output compare match interrupt.
	sei();				// Enables global interrupts.
	TCNT0 = 0;			// Resets timer.
}

void at_cmd_init (void)			// Initializes AT commands.
{
	char AT[] = "AT\r\n";						// Checks AT connect.
	char AT_CWMODE[] = "AT+CWMODE=1\r\n";				// Sets device mode, 1 =
									Station mode.
	char AT_CWJAP[] = "AT+CWJAP=\"SSD\",\"Password\"\r\n";		// Connects to WiFi using
									SSID and Password.
	char AT_CIPMUX[] = "AT+CIPMUX=0\r\n";				//Sets device for single IP
									Address Mode.
	char AT_CIPSTART[] = "AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n";// Starts
									TCP connection to
									Thingspeak.com at port 80.
	char AT_CIPSEND[] = "AT+CIPSEND=100\r\n";			// Gives upcoming string
									length.
	_delay_ms(200);
	// Sends commands listed above
	usart_send(AT);
	_delay_ms(5000);
	usart_send(AT_CWMODE);
	_delay_ms(5000);
```

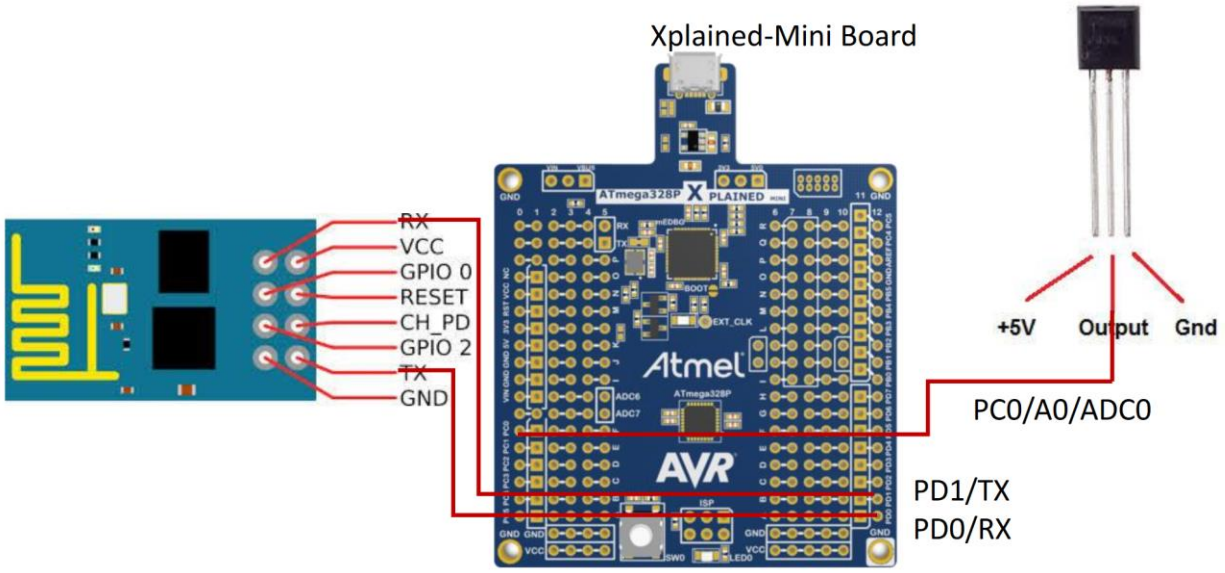
```
        usart_send(AT_CWJAP);
        _delay_ms(3000);
        usart_send(AT_CIPMUX);
        _delay_ms(3000);
        usart_send(AT_CIPSTART);
        _delay_ms(3000);
        usart_send(AT_CIPSEND);
        _delay_ms(5000);
}

void usart_init (void)      // Initializes USART.
{
        UCSR0B = (1<<TXEN0);
        UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);
        UBRR0L = F_CPU/16/BAUD_RATE-1;
}

void usart_send (char *ch)  // Sends characters out.
{
        while ((*ch != '\0'))
        {
                while (!(UCSR0A & (1 <<UDRE0)));
                UDR0 = *ch;
                ch++;
        }
}
```

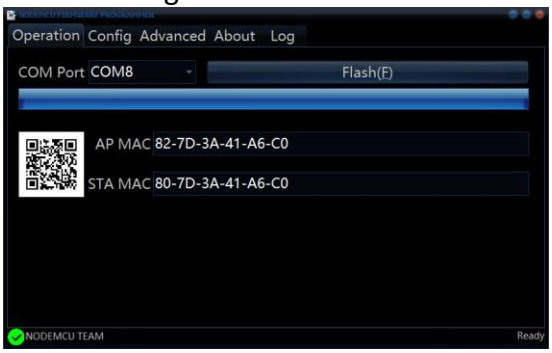
4. SCHEMATICS

The following schematic from the slides describe the connections that are made relatively well.

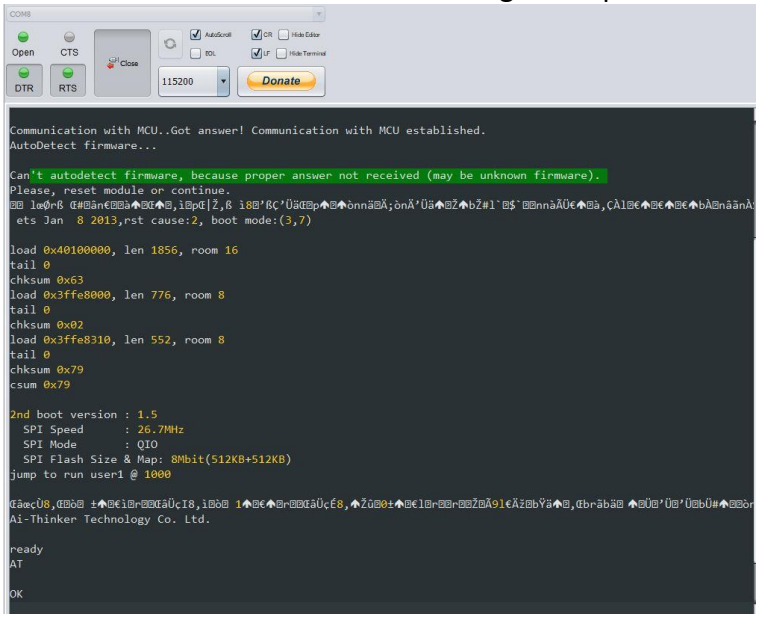


5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

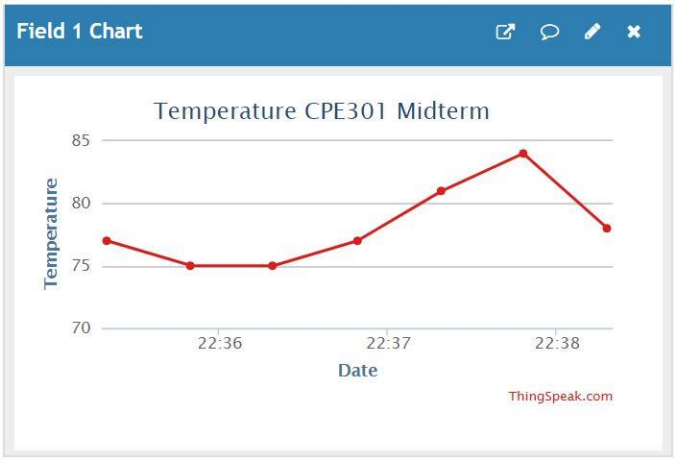
The following is a screenshot shows that the ESP module was programmed with the firmware.



The WiFi module was then tested using the ESplorer and it worked correctly.

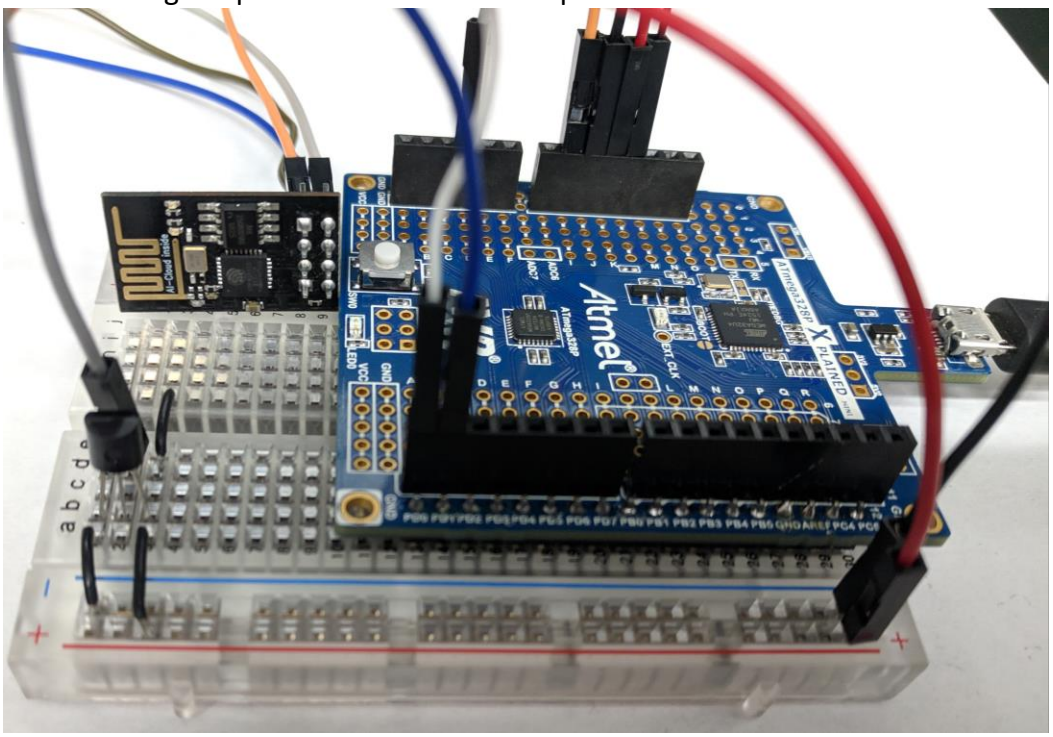


After everything was connected and the program was ran, the following graph was created on ThingSpeak using the supplied data from the temperature sensor.



6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

The following is a picture of the board setup:



7. VIDEO LINKS OF EACH DEMO

https://youtu.be/X9WeNYekWOw

8. GITHUB LINK OF THIS DA

https://github.com/dsenda/Smiles/tree/master/Midterm1

Student Academic Misconduct Policy

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

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

Daniel Senda