CPE301 – SPRING 2019

Design Assignment 5

Student Name: Rocky Yasuaki Gonzalez
Student #: 5003229733
Student Email: gonzar14@unlv.nevada.edu
Primary Github address: https://github.com/rockyg1995/ihswppdar.git
Directory: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA5

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 FLOW DIAGRAMS

Atmega328PB Xplained Mini
Micro USB Cable (Power Supply)
Breadboard
LM35
RF24L01
Male/Female Wires

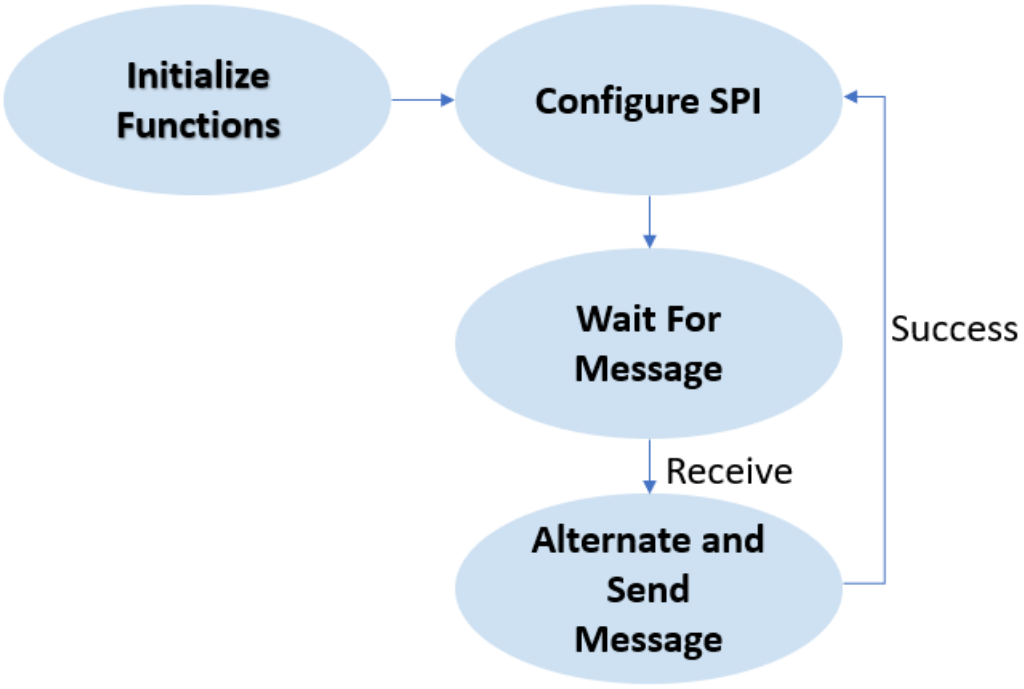

Figure 1 – Flow Chart for Coding Algorithm in Task

2. INITIAL/DEVELOPED CODE OF TASK

```
/*
 * DA5.c
 *
 * Created: 4/27/2019 7:16:05 AM
 * Author: RYG95
 */

#ifndef F_CPU
#define F_CPU 16000000UL
#endif

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

//     Set up UART for printf();
```

```
#ifndef BAUD
#define BAUD 9600
#endif
#include "STDIO_UART.h"

//     Include nRF24L01+ library
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"
#include "spi.h"
void print_config(void);

void adc_init(void);                    // Initialize Analog to Digital Converter
void read_adc(void);                    // Read temperature received from ADC
void USART_send(unsigned char data);    // Send individual char data into UDR0
void USART_putstring(char* StringPtr);  // Break string into individual chars and send

volatile float adc_temp;                // Stores ADC Value representing Temperature
char outs[20];                          // 'outs[]' used to store integer and float

//     Used in IRQ ISR
volatile bool message_received = false;
volatile bool status = false;

int main(void) {

       //     Set cliche message to send (message cannot exceed 32 characters)
       char tx_message[32];             // Define string array
       strcpy(tx_message,"Hello World!"); // Copy string into array

       //     Initialize UART
       uart_init();
       adc_init();
       float adc_tempf;

       //     Initialize nRF24L01+ and print configuration info
       nrf24_init();
       print_config();

       //     Start listening to incoming messages
       nrf24_start_listening();
       status = nrf24_send_message(tx_message);
       if (status == true) {
              printf("Message sent successfully\n")
              read_adc();
              adc_tempf = (ADCH << 8) + ADCL;    // 'T(C) = Vout/10mV', 'TOS = ADC - T(C)'
              adc_tempf = (9/5)*adc_tempf + 32;  // Converts Celsius to Fahrenheit
              snprintf(outs,sizeof(outs),"%3f\r\n", adc_tempf); // Store integer->string
              USART_putstring(outs);
       }

       while (1) {

              status = nrf24_send_message(tx_message);

              if (message_received) {

                     //     Message received, print it
                     message_received = false;
```

```
                        printf("Received message from geo: %s\n",nrf24_read_message());
                        //      Send message as response
                        _delay_ms(500);
                        if (status == true) printf("Message sent successfully\n");
                }
        }
}

//      Interrupt on IRQ pin
ISR(INT0_vect) {
        message_received = true;
}

void print_config(void) {
        uint8_t data;
        printf("Startup successful\n\n nRF24L01+ configured as:\n");
        printf("-----------------------------------------\n");
        nrf24_read(CONFIG,&data,1);
        printf("CONFIG              0x%x\n",data);
        nrf24_read(EN_AA,&data,1);
        printf("EN_AA               0x%x\n",data);
        nrf24_read(EN_RXADDR,&data,1);
        printf("EN_RXADDR           0x%x\n",data);
        nrf24_read(SETUP_RETR,&data,1);
        printf("SETUP_RETR          0x%x\n",data);
        nrf24_read(RF_CH,&data,1);
        printf("RF_CH               0x%x\n",data);
        nrf24_read(RF_SETUP,&data,1);
        printf("RF_SETUP            0x%x\n",data);
        nrf24_read(STATUS,&data,1);
        printf("STATUS              0x%x\n",data);
        nrf24_read(FEATURE,&data,1);
        printf("FEATURE             0x%x\n",data);
        printf("-----------------------------------------\n\n");
}

void adc_init(void) {
        ADMUX = (0<<REFS1)|(1<<REFS0)|     // Reference Select Bits, AVcc Ext cap at AREF
        (0<<ADLAR)|                        // ADC Left Adjust Result
        (0<<MUX3)|(1<<MUX2)|(0<<MUX1)|(1<<MUX0);  // Analog Channel 'ADC5' (PC5)

        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)|(0<<ADPS1)|(1<<ADPS0);          // ADC Prescaler Select Bits '32'
}

void read_adc(void) {
        unsigned char i = 4;                       // Set 'i' for iterations
        adc_temp = 0;                              // set float 'adc_temp'
        while (i--) {                              // Decrement 'i' until 4 samples take
                ADCSRA |= (1<<ADSC);               // If ADSC high (ADC Start Conversion)...
                while (ADCSRA & (1<<ADSC));        // Start ADC Conversion
                adc_temp += ADC;                   // Store analog value of current adc_temp
                _delay_ms(50);                     // delay 50ms for sampling
        }
```

```
        adc_temp = (adc_temp/4);                    // Average of 4 samples into adc_temp
}

void USART_send(unsigned char data) {               // Transmit ASCII value into UDR0
        while (!(UCSR0A & (1 << UDRE0)));           // Check UDRE0 'High' to break loop
        UDR0 = data;                                // Unsigned char serial data into UDR0
}

void USART_putstring(char* StringPtr) {             // Break string->chars, then USART_send()
        while (*StringPtr != 0x00) {                // Keep Looping until String Completed
                USART_send(*StringPtr);             // Send char pointed by string pointer
                StringPtr++;                        // Increment pointer next char location
        }
}
```

3. SCHEMATICS

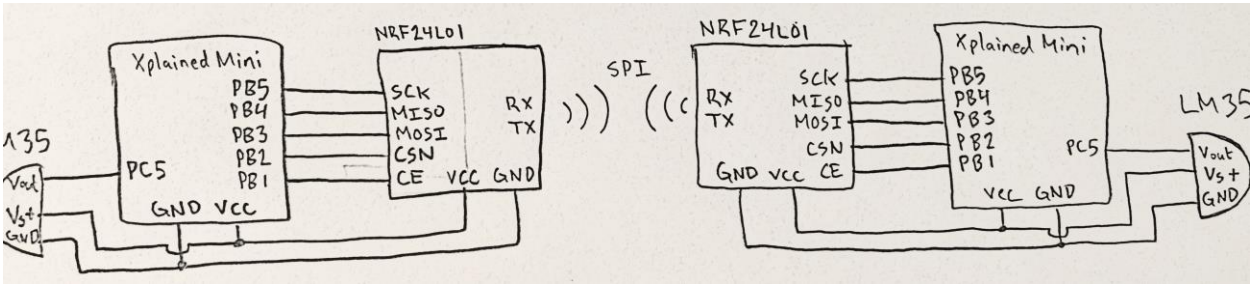

Figure 2 – Schematic of two Atmega328P/PB Xplained Minis + SPI Modules

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

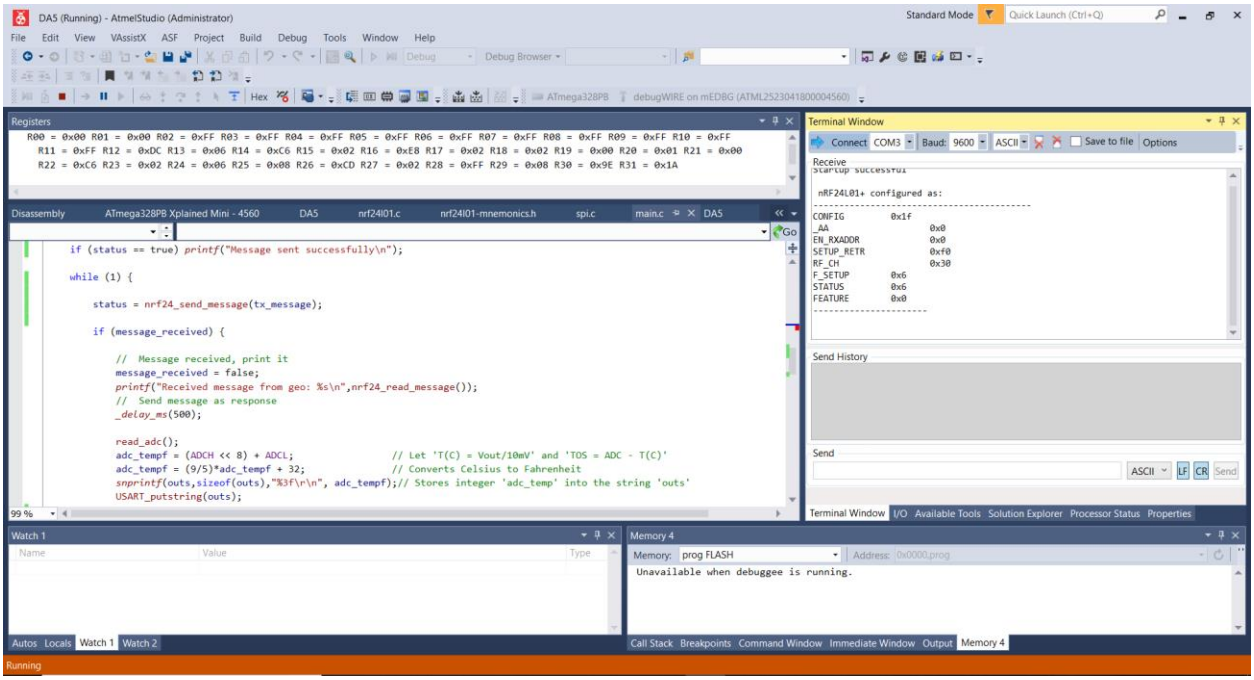

Figure 3 – Output Terminal for Configuration of SPI Module

There were problems communicating both devices to each after setting data to correct proper channels and opposite pipe addresses. We conclude that the assignment may have problems connecting Atmega328PB models since we've spoke to other classmates who've not had problems however had used the Atmega328P models.

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

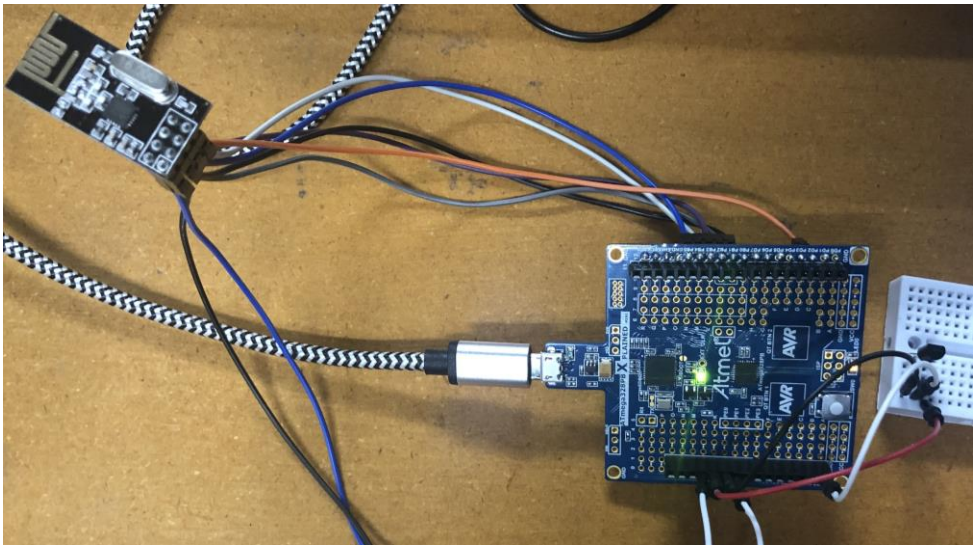

Figure 4 – Connecting NRF24L01 to the Xplained Mini + LM35 Temperature Sensor

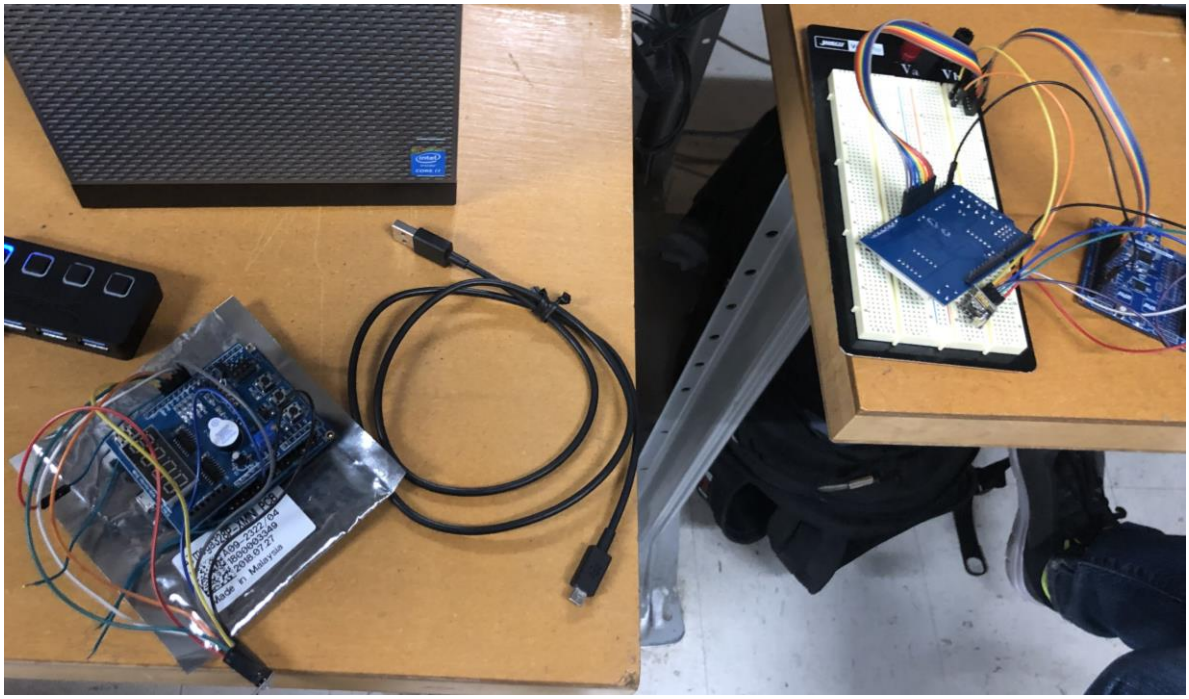

Figure 5 – Connecting two Xplained Minis/NRF24L01 Modules

6. VIDEO LINKS OF EACH DEMO

N/A

7. GITHUB LINK OF THIS DA

https://github.com/rockyg1995/ihswppdar/tree/master/DesignAssignments/DA5

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

"*This assignment submission is my own, original work*".
Rocky Gonzalez