CPE301 – SPRING 2019

Design Assignment 5

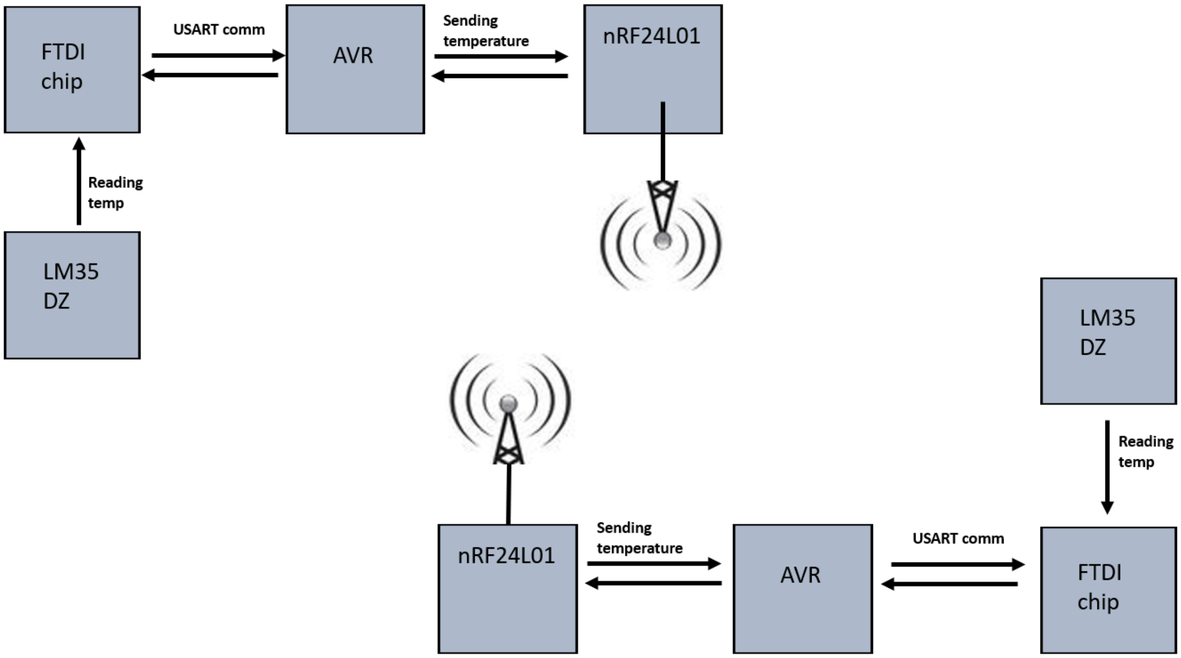
Student Name: Francisco Mata Carlos
Student #: 1012593607
Student Email: matacarl@unlv.nevada.edu
Primary Github address: https://github.com/chicosisco/da_sub.git
Directory: repository/cpe301/DesignAssignments/DA5

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

The components used for this assignment are the next:

a. Atmega328p Xplained Mini
b. Atmel Studio 7
c. FTDI chip
d. nRF24L01 single chip 2.4GHz Transceiver
e. LM35

Block diagram with pins used



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

1. Task 1

1. Interface the provided NRF24L01+ RF Module to the ATmega328p using the SPI interface. Using the earlier developed code for ADC, transmit the ADC value of the internal temperature sensor, or LM35 sensor between two RF Modules. The ATmega328p interfacing the RF Modules should alternate between TX and RX modes every 0.5 secs (hopefully they are not both at TX and RX modes in the same interval). The temperature of both ATmega328p's should be displayed on both ATmega328p's.

```
//      Set clock frequency
#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 "inc\STDIO_UART.c"

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

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

void ADC_init (void);
volatile unsigned char ADC_temp_val[5];
volatile uint8_t ADC_val_num;



int main(void)
{


        ADC_init();
```

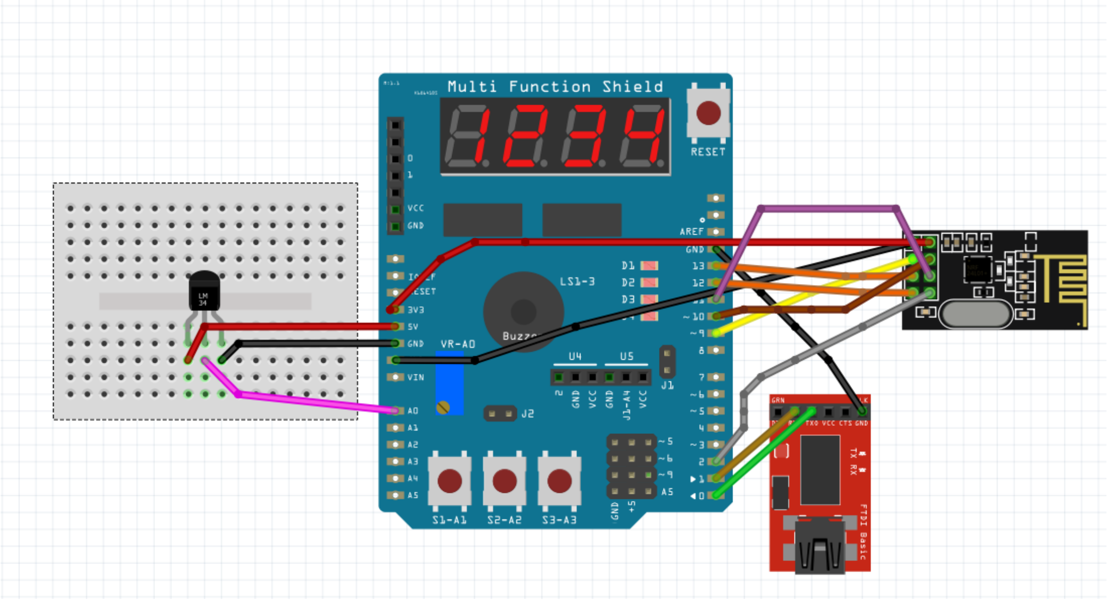
```c
       //     Set cliche message to send (message cannot exceed 32 characters)
       char tx_message[32];                     // Define string array
       strcpy(tx_message,"Hello World!"); // Copy string into array

       //     Initialize UART
       uart_init();

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

       //     Start listening to incoming messages
       nrf24_start_listening();

       strcpy(tx_message,"Hello World!"); // Copying string to array
       nrf24_send_message(tx_message);

       while (1)
       {
              if (message_received)
              {
                     //     Message received, print it
                     message_received = false;
                     printf("Received message: %s\n",nrf24_read_message());
                     //     Send message as response
                     _delay_ms(500);
                     status = nrf24_send_message(ADC_temp_val);
                     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");
}


//  Interrupt used to follow instructions below when conversion is done
ISR(ADC_vect)
{

       volatile unsigned int i=0;   // from characters to string
       char temp[4];

       ADC_val_num = (ADCH << 1);          // Shifts the value left to one place

       itoa(ADC_val_num, temp, 10);        // Converts integers to string
       // Takes ADCvalue, turns it into an ASCII representation
       // the ASCII representation will be stored under 'temp'
       // '10' represents the buffer

       while (i<4)                               // Transfers the temp string from itoa()
to ADCtemp
       {
              ADC_temp_val[i] = temp[i];
              i++;
       }
}

void ADC_init (void)
{
       // AVcc with external capacitor at AREF pin
       // ADC left adjust
       ADMUX |= (1<<REFS0)|(1<<ADLAR);

       //ADC enable
       // ADC Start Conversion
       // ADC Auto Trigger Enable
       // ADC Interrupt Enable
       // 128 prescaler=128
       ADCSRA |=(1<<ADEN)|(1<<ADSC)|(1<<ADATE)|(1<<ADIE)|(1<<ADPS2)|(ADPS1)|(ADPS0);
}
```

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

Same as above

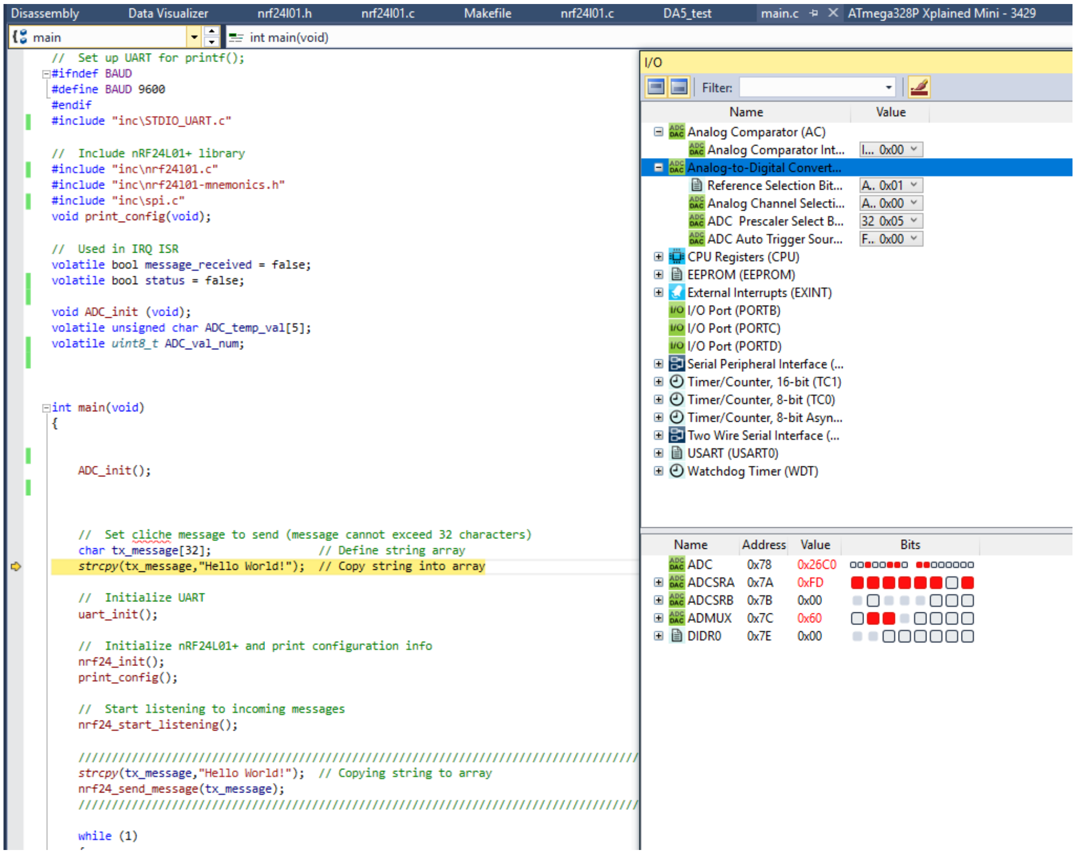
4. SCHEMATICS

Task 1



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

TASK 1

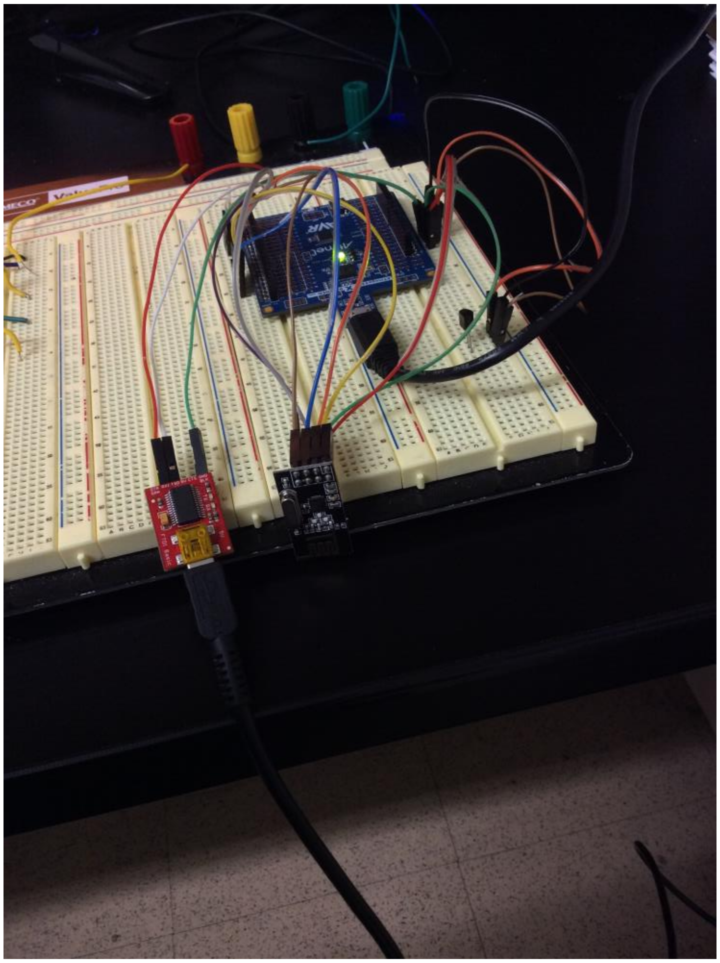


```c
//  Set up UART for printf();
#ifndef BAUD
#define BAUD 9600
#endif
#include "inc\STDIO_UART.c"

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

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

void ADC_init (void);
volatile unsigned char ADC_temp_val[5];
volatile uint8_t ADC_val_num;


int main(void)
{


    ADC_init();



    //  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();

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

    //  Start listening to incoming messages
    nrf24_start_listening();

    ////////////////////////////////////////////////////////////////////////////////
    strcpy(tx_message,"Hello World!");  // Copying string to array
    nrf24_send_message(tx_message);
    ////////////////////////////////////////////////////////////////////////////////

    while (1)
```

Name	Value
⊟ Analog Comparator (AC)	
Analog Comparator Int...	I... 0x00
⊟ Analog-to-Digital Convert...	
Reference Selection Bit...	A.. 0x01
Analog Channel Selecti...	A.. 0x00
ADC Prescaler Select B...	32 0x05
ADC Auto Trigger Sour...	F.. 0x00
⊞ CPU Registers (CPU)	
⊞ EEPROM (EEPROM)	
⊞ External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
⊞ Serial Peripheral Interface (...	
⊞ Timer/Counter, 16-bit (TC1)	
⊞ Timer/Counter, 8-bit (TC0)	
⊞ Timer/Counter, 8-bit Asyn...	
⊞ Two Wire Serial Interface (...	
⊞ USART (USART0)	
⊞ Watchdog Timer (WDT)	

Name	Address	Value	Bits
ADC	0x78	0x26C0	□□■□□■■□ ■■□□□□□□
⊞ ADCSRA	0x7A	0xFD	■■■■■■□■
⊞ ADCSRB	0x7B	0x00	□□□□□□□□
⊞ ADMUX	0x7C	0x60	□■■□□□□□
⊞ DIDR0	0x7E	0x00	□□□□□□□□

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



7. VIDEO LINKS OF EACH DEMO

Task 1 video:

https://youtu.be/iB9WnU45EDY

8. GITHUB LINK OF THIS DA

https://github.com/chicosisco/da_sub.git

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

"This assignment submission is my own, original work".
Francisco Mata Carlos