CPE301 – SPRING 2019

Design Assignment DA5

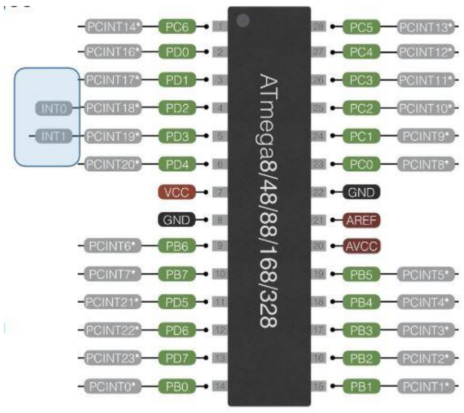
Student Name: Jett Guerrero
Student #: 5002377713
Student Email: guerrj1@unlv.nevada.edu
Primary Github address: https://github.com/guerrj1/Submission_DA.git
Directory: DA5 - https://github.com/guerrj1/Submission_DA/tree/master/DA5

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 CONNECTION BLOCK DIAGRAM w/ PINS

-ATMega328p
-Male to male wires
-Male to female wires
-Breadboard
-NRF24L01 Module
-FTDI Basic
-LM34 Temperature Sensor



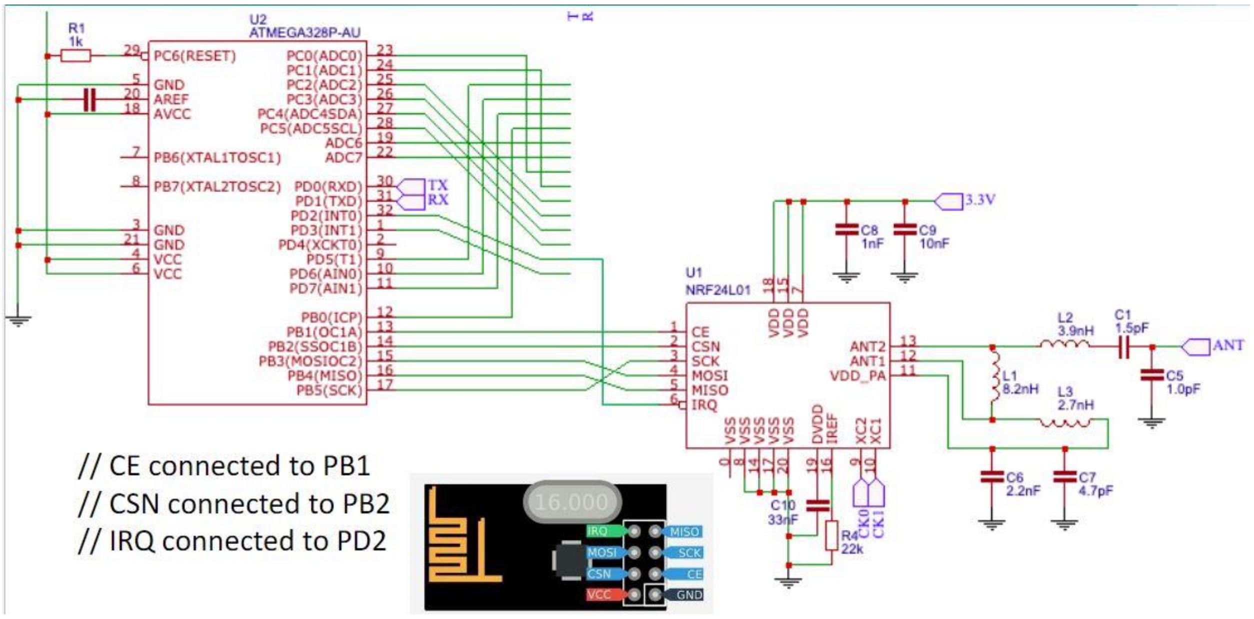
Atmega328P using PB0-PB5 for the ESP module. PC0 for input of LM34 sensor output.

2. DEVELOPED CODE OF TASK 1 C CODE

```c
//DA5

#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 "libraries\STDIO_UART.c"

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

//      Used in IRQ ISR
volatile bool message_received = false;
volatile bool status = false;
volatile uint8_t ADCvalue;              //Global variable, set to volatile if used with ISR
volatile unsigned char ADCtemp[5];

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();
	
	//	Initialize nRF24L01+ and print configuration info
	nrf24_init();
	print_config();
	
	//	Start listening to incoming messages
	nrf24_start_listening();
	
	nrf24_send_message(tx_message);
	
	adc_init();
	
	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(ADCtemp);
			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);
```

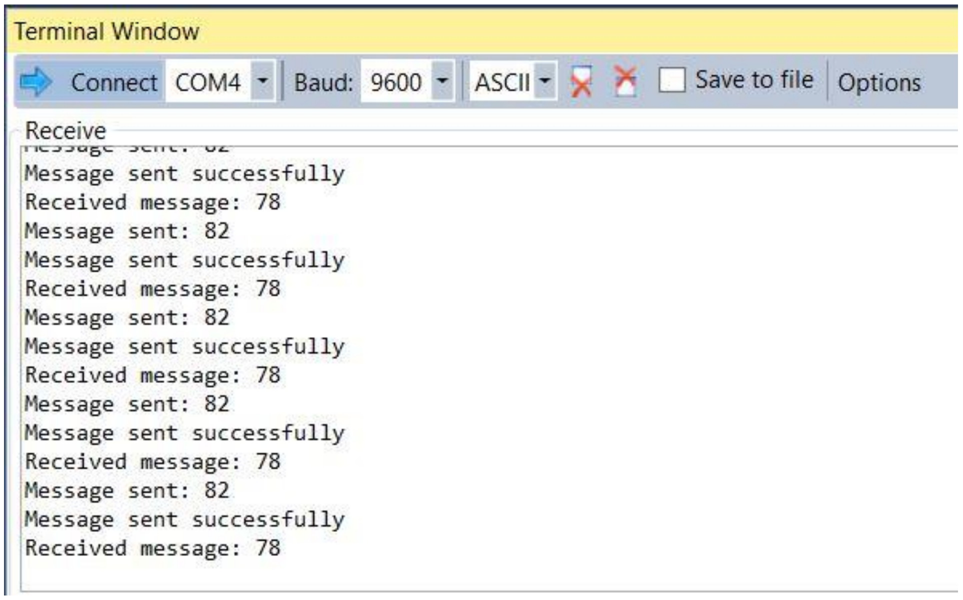
```
        printf("RF_SETUP             0x%x\n",data);
        nrf24_read(STATUS,&data,1);
        printf("STATUS        0x%x\n",data);
        nrf24_read(FEATURE,&data,1);
        printf("FEATURE              0x%x\n",data);
        printf("-------------------------------------------\n\n");
}

void adc_init(void)
{
        //ADC intializations
        ADMUX = 0;               //use ADC0
        ADMUX |= (1 << REFS0);  //use AVcc as the reference
        ADMUX |= (1 << ADLAR);  //Right adjust for 8 bit resolution
        ADCSRA |= (1 << ADATE); //Set ADC Auto Trigger Enable
        ADCSRB = 0;              //0 for free running mode
        ADCSRA |= (1 << ADEN);  //Enable the ADC
        ADCSRA |= (1 << ADIE);  //Enable Interrupts
        ADCSRA |= (1 << ADSC);  //Start the ADC conversion
        ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);  //set prescaler to 128
}

ISR(ADC_vect)
{
        volatile unsigned int j=0;
        char temp[5];

        ADCvalue = (ADCH<<1);  //shifts ADCH by 1 to the left
        itoa(ADCvalue, temp, 10);

        while(j<5)
        {
                ADCtemp[j] = temp[j]; //set temp as ADCtemp
                j++;  //increment the j temp
        }
}
```

3. SCHEMATIC



Atmega328P connection to the FTDI Basic and the NRF24L01 module.

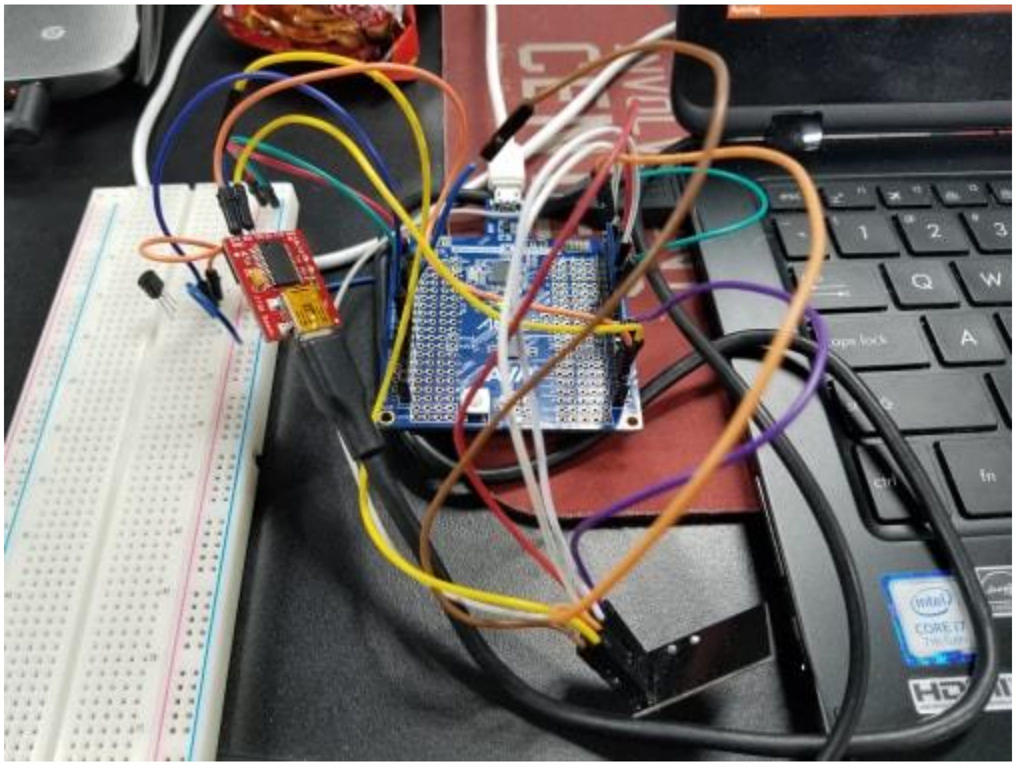
4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



```
Message sent successfully
Received message: 78
Message sent: 82
Message sent successfully
Received message: 78
Message sent: 82
Message sent successfully
Received message: 78
Message sent: 82
Message sent successfully
Received message: 78
Message sent: 82
Message sent successfully
Received message: 78
```

Terminal window of the SPI communication with another source.

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



Board set up for the SPI configuration.

6. VIDEO LINKS OF EACH DEMO

https://youtu.be/O378Oeloj74

7. GITHUB LINK OF THIS DA

https://github.com/guerrj1/Submission_DA/tree/master/DA5

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

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