CPE301 – SPRING 2019

Design Assignment 5

Student Name: Armon Latifi
Student #: 2000698173
Student Email: latifa1@unlv.nevada.edu
Primary Github address: https://github.com/armonlatifi
Directory: https://github.com/armonlatifi/sub_da/tree/master/DA5

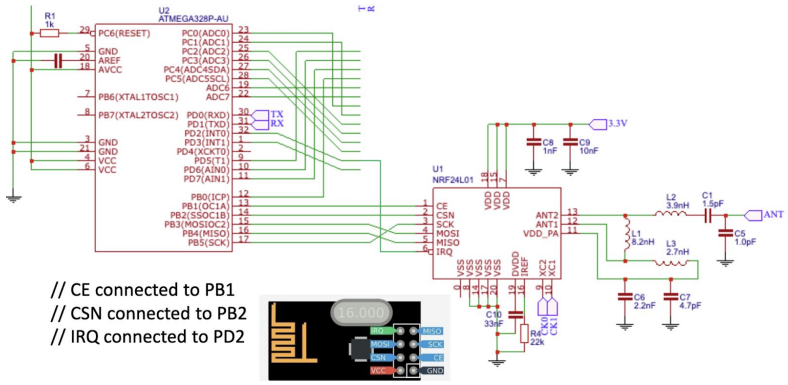
Submit the following for all Labs:

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

List of Components used:

- Assembler
- Simulator
- Debugger
- Breadboard
- Atmega328P
- Wires
- Xplained mini
- Micro usb
- Atmel studio 7
- LM34
- Nrf24l01



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

receive.c

```
#define F_CPU 16000000UL //set clock speed
#define BAUD 9600 //set baud rate
#define MYUBRR F_CPU/16/BAUD-1 //calculate Baud

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include <util/delay.h>
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"

nRF24L01 *setup_rf(void);
void process_message(char *message);
inline void prepare_led_pin(void);
inline void set_led_high(void);
inline void set_led_low(void);
volatile bool rf_interrupt = false;
void read_adc(void);
```

```
void adc_init(void);  //set up ADC
void USART_init( unsigned int ubrr );
void USART_tx_string(char *data); //prints stirng usart
volatile unsigned int adc_temp;
char outs[20]; //array


int main(void) {
	uint8_t address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
	prepare_led_pin();

		adc_init(); //initialize ADC
		USART_init(MYUBRR);
		USART_tx_string("Connected!\r\n"); //connection successful
		_delay_ms(125);
	sei(); //interrupts
	nRF24L01 *rf = setup_rf();
	nRF24L01_listen(rf, 0, address);
	uint8_t addr[5];
	nRF24L01_read_register(rf, CONFIG, addr, 1);
	while (true) {
		if (rf_interrupt) {
			rf_interrupt = false;
			while (nRF24L01_data_received(rf)) {
				nRF24L01Message msg;
				nRF24L01_read_received_data(rf, &msg);
				process_message((char *)msg.data);
				USART_tx_string(msg.data);
			}
			nRF24L01_listen(rf, 0, address);
		}
	}
	return 0;
}
nRF24L01 *setup_rf(void) {
	nRF24L01 *rf = nRF24L01_init();
	rf->ss.port = &PORTB;
	rf->ss.pin = PB2;
	rf->ce.port = &PORTB;
	rf->ce.pin = PB1;
	rf->sck.port = &PORTB;
	rf->sck.pin = PB5;
	rf->mosi.port = &PORTB;
	rf->mosi.pin = PB3;
	rf->miso.port = &PORTB;
	rf->miso.pin = PB4;

	//setup interrupts on falling edge
	EICRA |= _BV(ISC01);
	EIMSK |= _BV(INT0);
	nRF24L01_begin(rf);
	return rf;
}

void process_message(char *message) {
	if (strcmp(message, "ON") == 0)
	set_led_high();
```

```
        else if (strcmp(message, "OFF") == 0)
        set_led_low();
}
inline void prepare_led_pin(void) {
        DDRB |= _BV(PB0);
        PORTB &= ~_BV(PB0);
}
inline void set_led_high(void) {
        PORTB |= _BV(PB0);
}
inline void set_led_low(void) {
        PORTB &= ~_BV(PB0);
}

void adc_init(void)
{

        ADMUX = (0<<REFS1)|
                                        (1<<REFS0)|
                                        (0<<ADLAR)|
                                        (0<<MUX2)|
                                        (1<<MUX1)|
                                        (0<<MUX0);

        ADCSRA = (1<<ADEN)| //enables ADC
                                         (0<<ADSC)|
                                         (0<<ADATE)|
                                         (0<<ADIF)|
                                         (0<<ADIE)|
                                         (1<<ADPS2)|
                                         (0<<ADPS1)|
                                         (1<<ADPS0);
}
void read_adc(void) {
        unsigned char i =4;
        adc_temp = 0; //initialize
        while (i--) {
                ADCSRA |= (1<<ADSC);
                while(ADCSRA & (1<<ADSC));
                adc_temp+= ADC;
                _delay_ms(50);
        }
        adc_temp = adc_temp / 4;

}

void USART_init( unsigned int ubrr ) {
        UBRR0H = (unsigned char)(ubrr>>8);
        UBRR0L = (unsigned char)ubrr;
        UCSR0B = (1 << TXEN0);
        UCSR0C = (3 << UCSZ00);
}

void USART_tx_string( char *data ) {
        while ((*data != '\0')) {
                while (!(UCSR0A & (1 <<UDRE0)));
                UDR0 = *data;
```

```
                data++;
        }
}

ISR(INT0_vect) {
        rf_interrupt = true;
}
```

transmit.c

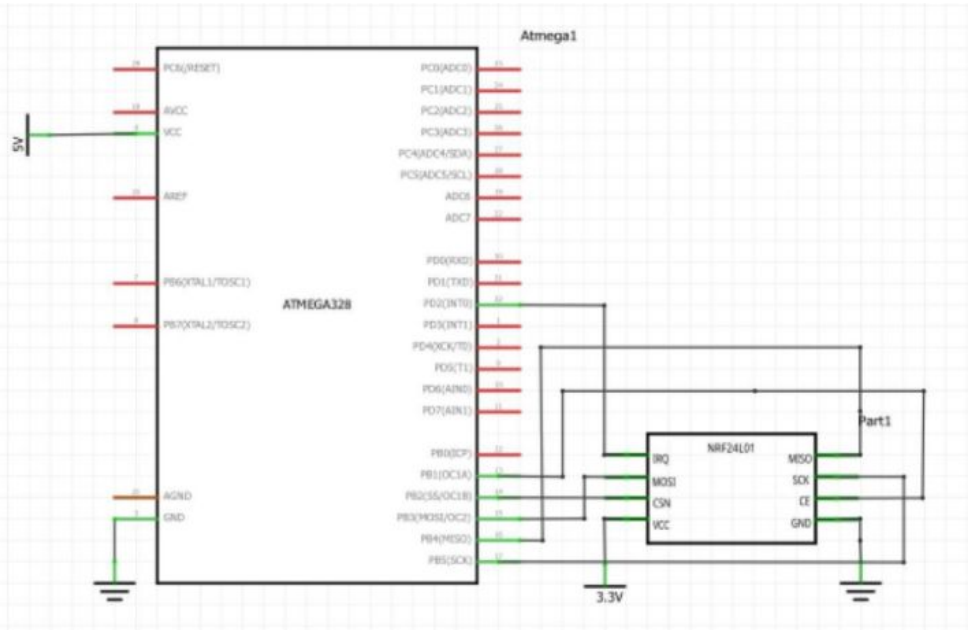
```
#define F_CPU 16000000UL //set clock speed
#define BAUD 9600 //set baud rate
#define MYUBRR F_CPU/16/BAUD-1

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include <util/delay.h>
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"

nRF24L01 *setup_rf(void);
void process_message(char *message);
inline void prepare_led_pin(void);
inline void set_led_high(void);
inline void set_led_low(void);
volatile bool rf_interrupt = false;
void read_adc(void);
void adc_init(void);
void USART_init( unsigned int ubrr );
void USART_tx_string(char *data); //prints string
volatile unsigned int adc_temp;
char outs[20];


int main(void) {
        uint8_t address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
        prepare_led_pin();
        adc_init();
        USART_init(MYUBRR);
        USART_tx_string("Connected\r\n"); //connection successful
        _delay_ms(125);
        sei(); //interrupts
        nRF24L01 *rf = setup_rf();
        nRF24L01_listen(rf, 0, address);
        uint8_t addr[5];
        nRF24L01_read_register(rf, CONFIG, addr, 1);
        while (true) {
                if (rf_interrupt) {
                        rf_interrupt = false;
                        while (nRF24L01_data_received(rf)) {
                                nRF24L01Message msg;
                                nRF24L01_read_received_data(rf, &msg);
                                process_message((char *)msg.data);
                                USART_tx_string(msg.data);
                        }
```

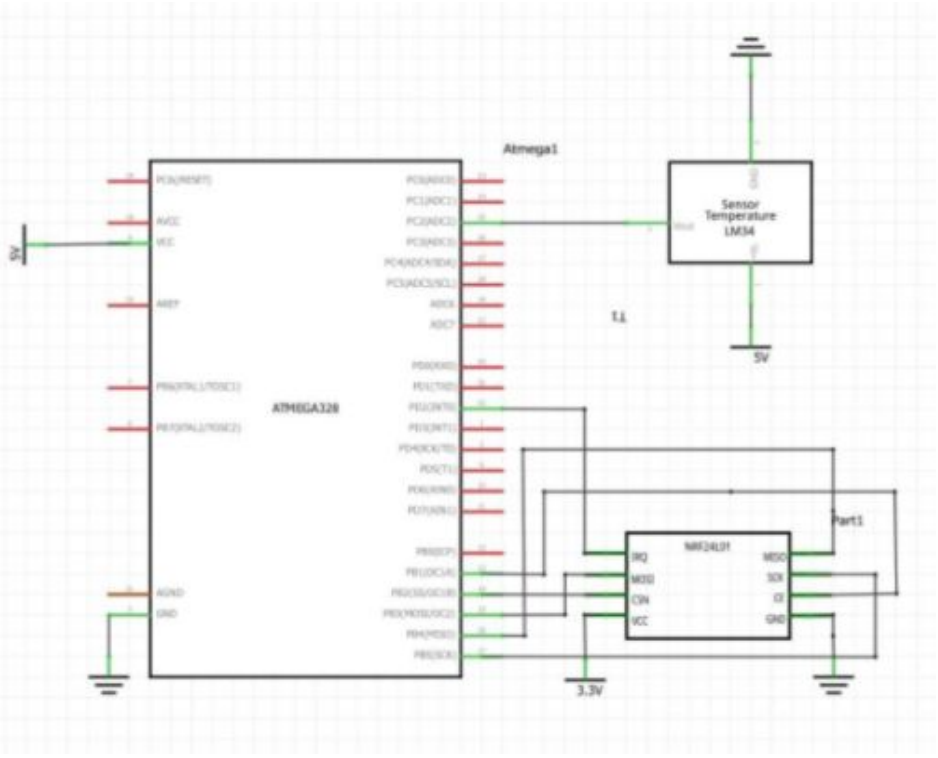
```
                        nRF24L01_listen(rf, 0, address);
                }
        }
        return 0;
}
nRF24L01 *setup_rf(void) {
        nRF24L01 *rf = nRF24L01_init();
        rf->ss.port = &PORTB;
        rf->ss.pin = PB2;
        rf->ce.port = &PORTB;
        rf->ce.pin = PB1;
        rf->sck.port = &PORTB;
        rf->sck.pin = PB5;
        rf->mosi.port = &PORTB;
        rf->mosi.pin = PB3;
        rf->miso.port = &PORTB;
        rf->miso.pin = PB4;
        //interrupt on falling edge
        EICRA |= _BV(ISC01);
        EIMSK |= _BV(INT0);
        nRF24L01_begin(rf);
        return rf;
}

void process_message(char *message) {
        if (strcmp(message, "ON") == 0)
        set_led_high();
        else if (strcmp(message, "OFF") == 0)
        set_led_low();
}
inline void prepare_led_pin(void) {
        DDRB |= _BV(PB0);
        PORTB &= ~_BV(PB0);
}
inline void set_led_high(void) {
        PORTB |= _BV(PB0);
}
inline void set_led_low(void) {
        PORTB &= ~_BV(PB0);
}

void adc_init(void)
{

        ADMUX = (0<<REFS1)|
                                        (1<<REFS0)|
                                        (0<<ADLAR)|
                                        (0<<MUX2)|
                                        (1<<MUX1)|
                                        (0<<MUX0);

        ADCSRA = (1<<ADEN)| //enables ADC
                                         (0<<ADSC)|
                                         (0<<ADATE)|
                                         (0<<ADIF)|
                                         (0<<ADIE)|
                                         (1<<ADPS2)|
```

```
                                        (0<<ADPS1)|
                                        (1<<ADPS0);

}
void read_adc(void) {
        unsigned char i =4;
        adc_temp = 0; //initialize
        while (i--) {
                ADCSRA |= (1<<ADSC);
                while(ADCSRA & (1<<ADSC));
                adc_temp+= ADC;
                _delay_ms(50);
        }
        adc_temp = adc_temp / 4; //take the average of a few samples

}

void USART_init( unsigned int ubrr ) {
        UBRR0H = (unsigned char)(ubrr>>8);
        UBRR0L = (unsigned char)ubrr;
        UCSR0B = (1 << TXEN0); //enable receiver
        UCSR0C = (3 << UCSZ00);
}

void USART_tx_string( char *data ) {
        while ((*data != '\0')) {
                while (!(UCSR0A & (1 <<UDRE0)));
                UDR0 = *data;
                data++;
        }
}

ISR(INT0_vect) {
        rf_interrupt = true;
}
```

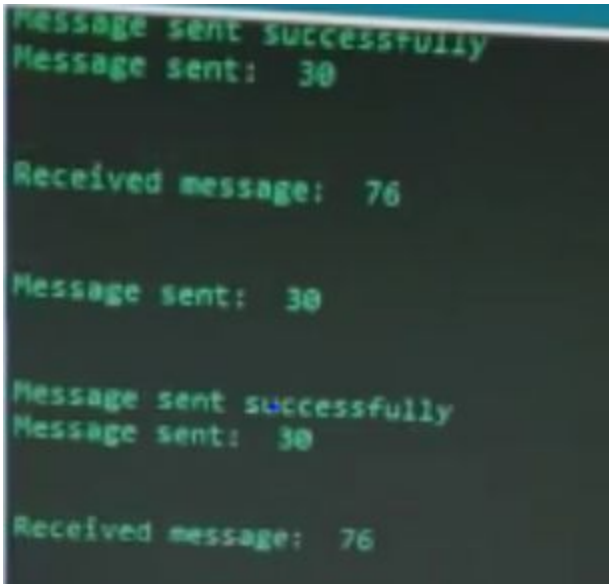
3. SCHEMATICS

Use fritzing.org

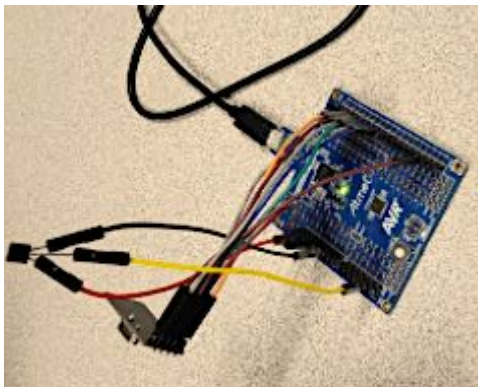




4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)



5. SCREENSHOT OF EACH DEMO (BOARD SETUP)



6. GITHUB LINK OF THIS DA

https://github.com/armonlatifi/sub_da/tree/master/DA5

Student Academic Misconduct Policy

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

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

Armon Latifi