Elizabeth Heider
Adrian Ruiz

CPE301 – SPRING 2018

Midterm 2

DO NOT REMOVE THIS PAGE DURING SUBMISSION:

The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E		
4.	SCHEMATICS		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
5.	SCREENSHOT OF EACH DEMO		
6.	VIDEO LINKS OF EACH DEMO		
7.	GOOGLECODE LINK OF THE DA		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

- 2 ATmegas 328p Microcontrollers
- NRF24L01 Transceiver
- LM34 Temperature Sensor

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

(TRANSMIT CODE)

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include "nrf24l01.h"
#include "ioe.h"

void setup_timer(void);
nRF24L01 *setup_rf(void);
void adc_init(void);

volatile bool rf_interrupt = false;
volatile bool send_message = false;
volatile int temp;
vchar outs[3];

int main(void) {
       usart0_init_();
       uint8_t to_address[5] = { 0x11, 0x11, 0x11, 0x11, 0x11 };
       bool on = false;
       DDRC = 0x00;
       sei();
       nRF24L01 *rf = setup_rf();
       setup_timer();

       while (true) {
              while(ADCSRA & (1<<ADSC));
              temp = ADC;
              if (rf_interrupt) {
                     rf_interrupt = false;
                     int success = nRF24L01_transmit_success(rf);
                     if (success != 0)
                     nRF24L01_flush_transmit_message(rf);
              }

              if (send_message) {
                     send_message = false;
                     on = !on;
                     nRF24L01Message msg;
                     sprintf(outs, "%d" ,  temp);
                     memcpy(msg.data, outs, 3);
                     msg.length = strlen((char *)msg.data) + 1;
                     nRF24L01_transmit(rf, to_address, &msg);
              }
       }

       return 0;
}

nRF24L01 *setup_rf(void) {
       nRF24L01 *rf = nRF24L01_init();
       rf->ss.port = &PORTB;
```

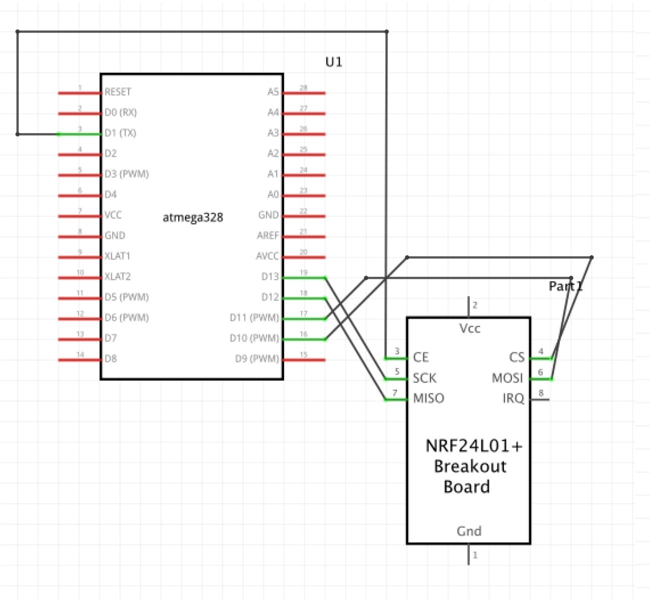
```
        rf->ss.pin = PB2;
        rf->ce.port = &PORTB;
        rf->ce.pin = PB1;
        rf->sck.port = &PORTB;
        rf->sck.pin = PB5;
        rf->mosi.port = &PORTB;
        rf->mosi.pin = PB3;
        rf->miso.port = &PORTB;
        rf->miso.pin = PB4;
        // interrupt on falling edge of INT0 (PD2)
        EICRA |= _BV(ISC01);
        EIMSK |= _BV(INT0);
        nRF24L01_begin(rf);
        return rf;
}

// setup timer to trigger interrupt every second when at 1MHz
void setup_timer(void) {
        TCCR1B |= _BV(WGM12);
        TIMSK1 |= _BV(OCIE1A);
        OCR1A = 15624;
        TCCR1B |= _BV(CS10) | _BV(CS11);
}

// each one second interrupt
ISR(TIMER1_COMPA_vect) {
        send_message = true;
}

// nRF24L01 interrupt
ISR(INT0_vect) {
        rf_interrupt = true;
}

void adc_init()
{
        ADMUX =0;
        ADCSRA= (1<<ADPS0) | (1<<ADPS1) | (1<<ADPS2);   //sample rate of 125kHz at 16MHz
        ADMUX |= (1<<REFS0);                                          //Set ADC reference
to AVCC
        ADMUX |= (1<<ADLAR);                                          //Left adjust ADC
Result to allow 8 bit reading

        ADCSRA |= (1<<ADEN);                                          //Enable ADC
        ADCSRA |= (1<<ADATE);                                                //set ADC
auto trigger enable
        ADCSRB = 0;                                                                 //0
for free running mode
        ADCSRA |= (1<<ADSC)  ;                                               //start A2D
Conversion

}
```

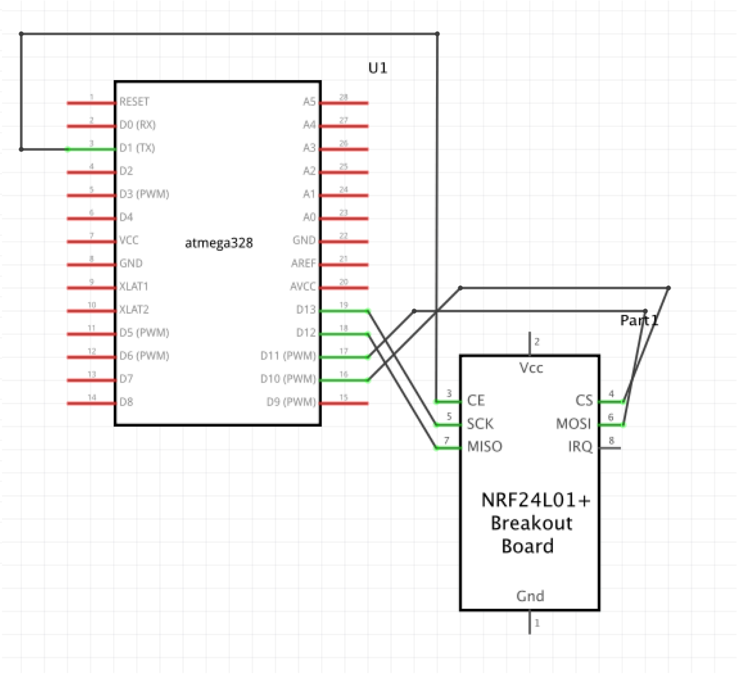
(RECEIVER CODE)

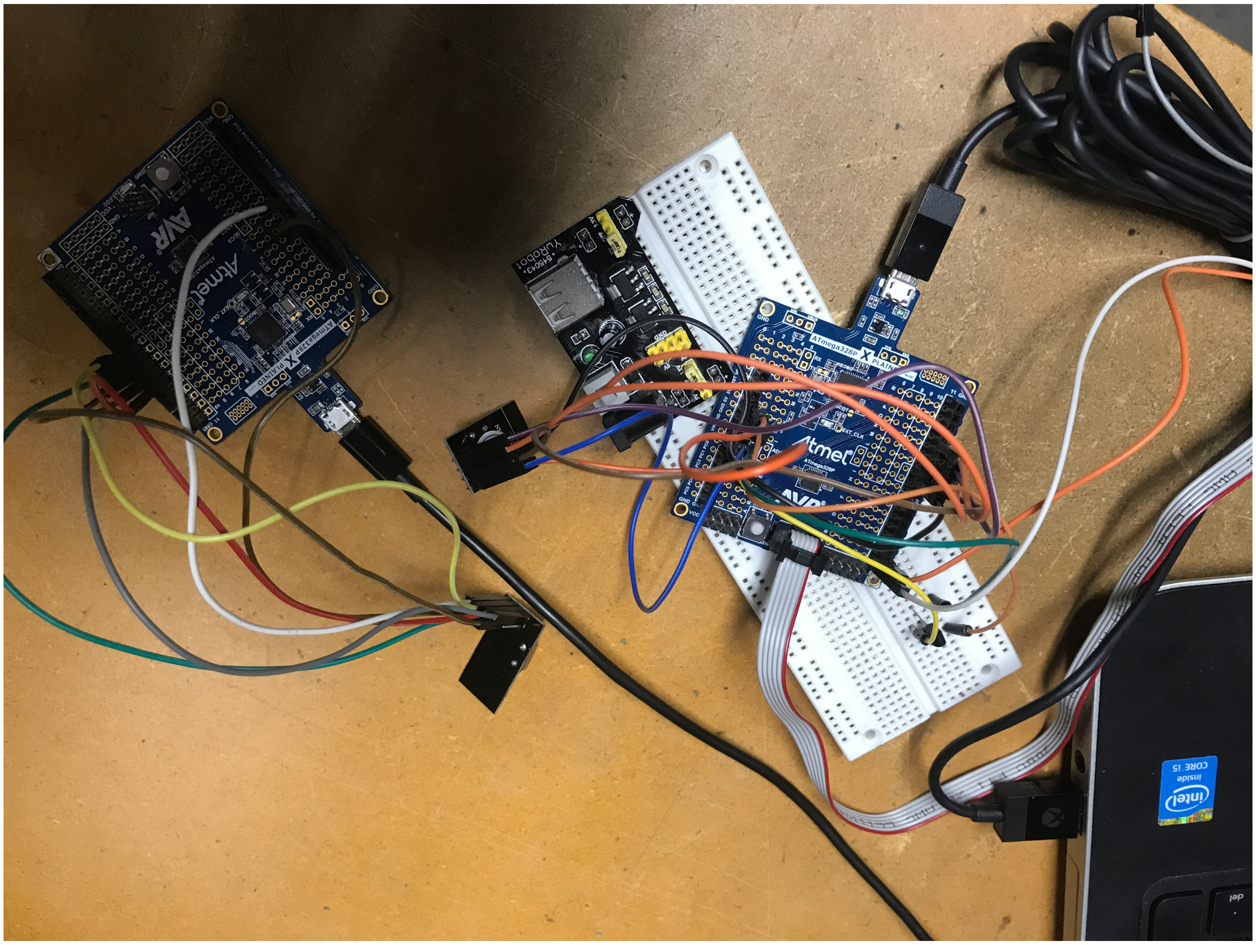
```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#ifndef F_CPU
#define F_CPU 16000000UL
#endif
#include <util/delay.h>
#include "nrf24l01.h"
#include "ioe.h"
#include "nrf24l01-mnemonics.h"


#define UBRR_2400 207

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);
void USART_INIT(unsigned int ubrr);
void USART_tx_string(char* data);

volatile bool rf_interrupt = false;
char* outs[20];

int main(void) {
        usart0_init_();
        _delay_ms(2000);

        uint8_t address[5] = { 0x11, 0x11, 0x11, 0x11, 0x11 };
        printm("Address INIT\r\n");
        prepare_led_pin();
        printm("LED PREP\r\n");

        sei();
        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);
                        }

                        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 of INT0 (PD2)
       EICRA |= _BV(ISC01);
       EIMSK |= _BV(INT0);
       nRF24L01_begin(rf);
       return rf;
}

void process_message(char *message) {
       printm(message);
}

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);
}

// nRF24L01 interrupt
ISR(INT0_vect) {
       rf_interrupt = true;
}

void USART_INIT(unsigned int ubrr){
       UBRR0H = (unsigned char)(ubrr>>8);
       UBRR0L = (unsigned char)ubrr;
       UCSR0B = (1<<TXEN0);

       UCSR0C = (1<<UCSZ00);
}

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

3. SCHEMATICS







4. VIDEO LINKS OF EACH DEMO

https://youtu.be/bVjIWe12VKk

5. GITHUB LINK OF THIS DA

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

“This assignment submission is my own, original work”.
Elizabeth Heider