Brian Lopez

Nathan Hanuscin

CPE301 – SPRING 2018

Design Assignment 4

**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**

**Disclaimer: My partner for this assignment is Nathan Hanuscin**

* Atmega328P x 2
* FTDI x 2
* NRF24l01 x 2
* One LM34
* Jumper cables

1. **CODE FOR RECEIVER**

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <string.h>

#include <stdbool.h>

#include <stdio.h>

#include <avr/interrupt.h>

#include "nrf24l01.h"

volatile bool rf\_interrupt = false;

// Set up all of the SPI ports with the nRF struct and turn on int0 interrupt

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;

EICRA |= \_BV(ISC01);

EIMSK |= \_BV(INT0); // turn on int0 interrupt

nRF24L01\_begin(rf);

return rf;

}

void spi\_init() {

DDRB &= ~((1<<2)|(1<<3)|(1<<5)); //SCK, MOSI and SS as inputs

DDRB |= (1<<4); // MISO as output

SPCR &= !(1<<MSTR); // set as slave

SPCR |= (1<<SPR0)|(1<<SPR1); // divide clock by 128

SPCR |= (1<<SPE); // enable SPI

}

void init\_uart(){

// setting the baud rate based on FCPU and baudrate

UBRR0H =0x00;

UBRR0L =0x0C;

// enabling TX & RX

UCSR0B = (1<<RXEN0)|(1<<TXEN0); // enable receive and transmit

UCSR0A = (1<<UDRE0)|(1<<U2X0);

UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Set frame: 8data, 1 stop

}

void ADC\_init() {

ADMUX = 0; // read from port ADC0

ADMUX |= (1<<REFS0); // use Vcc for reference

ADCSRA |= (1<<ADPS2) | (1<<ADPS1); // prescalar of 64

ADCSRA |= (1<<ADEN); // enable ADC

ADCSRB = 0; // free running mode

}

void USART\_Transmit( char \*data)

{

while((\*data != '\0')) { // transmits all chars but null

while(!(UCSR0A & (1<<UDRE0))); // waits for transmit flag to clear

UDR0 = \*data; // transmit next char

data++; // move to next char

}

}

unsigned int readADC()

{

ADMUX &= ~(1<<ADLAR); // clear the adc value

unsigned int val = 0;

ADCSRA |= (1 << ADSC); // start adc

while(ADCSRA & (1<<ADSC)); // wait until adc is done

val = ADC;

val = val \* 0.427; // doing (5 \* 100 \* adc) / 1024, just simplified

return val;

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true; // turn on variable for while loop

EIFR |= (INTF0); // reset interrupt flag

}

int main(void)

{

init\_uart(); // set UART variables

ADC\_init(); // set ADC variables

*\_delay\_ms*(150);

USART\_Transmit("Started!\r\n");

*uint8\_t* address[5] = {0x01, 0x01, 0x01, 0x01, 0x01 }; // address for nRF

sei(); // turn on global interrupts

nRF24L01 \*rf = setup\_rf(); // initialze and setup nRF struct

nRF24L01\_listen(rf, 0, address);

*uint8\_t* addr[5];

nRF24L01\_read\_register(rf, 0x00, addr, 1);

while (1)

{

if (rf\_interrupt)

{

rf\_interrupt = false;

while (nRF24L01\_data\_received(rf)) {

nRF24L01Message msg;

nRF24L01\_read\_received\_data(rf, &msg); // gets data that was sent

USART\_Transmit((char \*)msg.data); // transmit the temperature value sent

USART\_Transmit("\r\n"); // transmit a line feed

}

nRF24L01\_listen(rf, 0, address);

}

}

return 0;

}

1. **CODE FOR TRANSMITTER**

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <string.h>

#include <stdio.h>

#include "nrf24l01.h"

#define UBRR\_9600 51 // for 8Mhz with .2% error

#define *F\_CPU* 8000000UL

#include <util/delay.h>

void spi\_init(void);

void setup\_timer(void);

nRF24L01 \*setup\_rf(void);

void adc\_init(void);

void read\_adc(void);

void USART\_init( unsigned int ubrr );

void USART\_tx\_string( char \*data );

volatile bool rf\_interrupt = false;

volatile bool send\_message = false;

volatile unsigned int adc\_temp;

char outs[20];

int main(void)

{

*uint8\_t* to\_address[5] = { 0x01, 0x01, 0x01, 0x01, 0x01 };

spi\_init();

USART\_init(UBRR\_9600); //Initialize the USART (RS232 interface)

USART\_tx\_string("Connected!\r\n"); //Display connected

*\_delay\_ms*(125); //wait a bit

sei();

nRF24L01 \*rf = setup\_rf();

adc\_init(); // start ADC

setup\_timer(); // setup timer for CTC every 1 second

while (true)

{

if (rf\_interrupt)

{

rf\_interrupt = false;

int success = nRF24L01\_transmit\_success(rf);

if (success != 0)

nRF24L01\_flush\_transmit\_message(rf);

}

if (send\_message)

{

read\_adc(); // read ADC value

send\_message = false;

nRF24L01Message msg;

*snprintf*(outs,sizeof(outs),"%3d\r\n", adc\_temp);

USART\_tx\_string(outs); // display temperature read

*memcpy*(msg.data, outs, 3);

msg.length = *strlen*((char \*)msg.data) + 1;

nRF24L01\_transmit(rf, to\_address, &msg); // transmit the temperature value

}

}

return 0;

}

void adc\_init(void)

{

/\*\* Setup and enable ADC \*\*/

ADMUX = 0; //select ADC0 Pin as input

ADMUX = (0<<REFS1)| //Reference Selection Bits

(1<<REFS0)| //AVcc - external cap at AREF

(1<<ADLAR); //ADC right Adjust Result

ADCSRA = (1<<ADEN)| //ADC ENable

(1<<ADSC)| //ADC Start Conversion

(1<<ADATE)| //ADC Auto Trigger Enable

(0<<ADIF)| //ADC Interrupt Flag

(0<<ADIE)| //ADC Interrupt Enable

(1<<ADPS2)| //ADC Prescaler of 64

(1<<ADPS1)|

(0<<ADPS0);

ADCSRB = 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 read\_adc(void)

{

adc\_temp = 0; //initalize temp to 0

ADCSRA |= (1<<ADSC); //start the conversion

while((ADCSRA & (1<<ADIF)) == 0);

{

//wait for conversion to finish

}

adc\_temp = ADCH; //get temp value

}

void spi\_init(void)

{

DDRB |= (1<<2)|(1<<3)|(1<<5); // SCK, MOSI and SS as outputs

DDRB &= ~(1<<4); // MISO as input

SPCR |= (1<<MSTR); // Set as Master

SPCR |= (1<<SPR0)|(1<<SPR1); // divided clock by 128

SPCR |= (1<<SPE); // Enable SPI

}

// setup timer to trigger interrupt every second when at 8MHz

void setup\_timer(void)

{

TCCR1B |= \_BV(WGM12);

TIMSK1 |= \_BV(OCIE1A);

OCR1A = 31250;

TCCR1B |= \_BV(CS12);

}

/\* INIT USART (RS-232) \*/

void USART\_init( unsigned int ubrr )

{

UBRR0H = (unsigned char)(ubrr>>8); //set baud rate

UBRR0L = (unsigned char)ubrr;

UCSR0B = (1 << TXEN0) | (1 <<RXEN0); // Enable receiver, transmitter

UCSR0C = (1 << UCSZ00) | (1 << UCSZ01); //asynchronous 8-bit data 1 stop bit

}

/\* SEND A STRING TO THE RS-232\*/

void USART\_tx\_string( char \*data )

{

while ((\*data != '\0'))

{

while (!(UCSR0A & (1 <<UDRE0)))

{

//wait for the transmit buffer to empty

}

UDR0 = \*data; //put the data into the empty buffer, which sends the data

*\_delay\_ms*(125); // wait a bit

data++;

}

}

// each one second interrupt

ISR(TIMER1\_COMPA\_vect)

{

send\_message = true;

TIFR1 |= (1<<OCF1A);

}

// nRF24L01 interrupt

ISR(INT0\_vect)

{

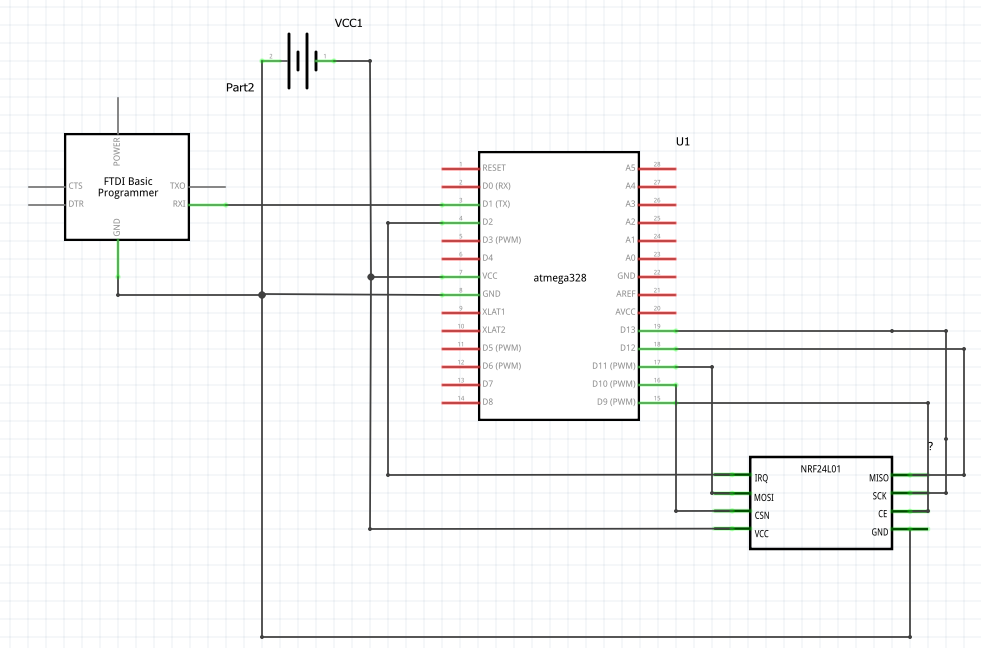
rf\_interrupt = true;

EIFR |= (1<<INTF0);

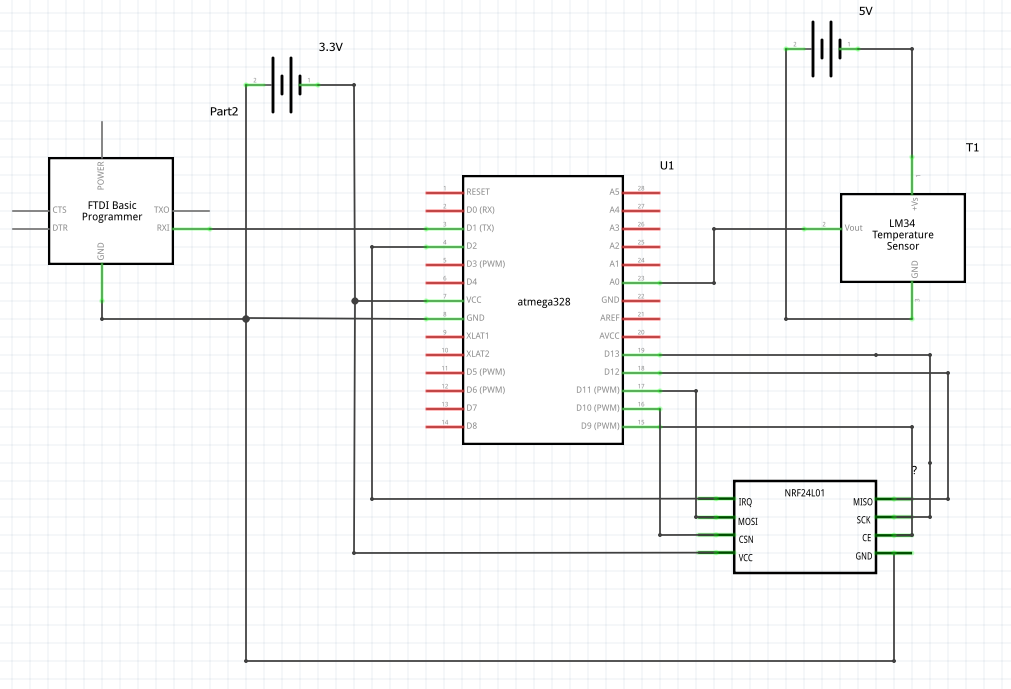
}

1. **SCHEMATICS**

**For Receiver:**

****

**For Transmitter:**

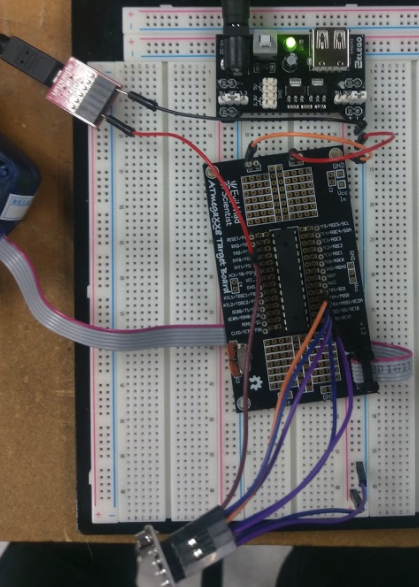
****

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

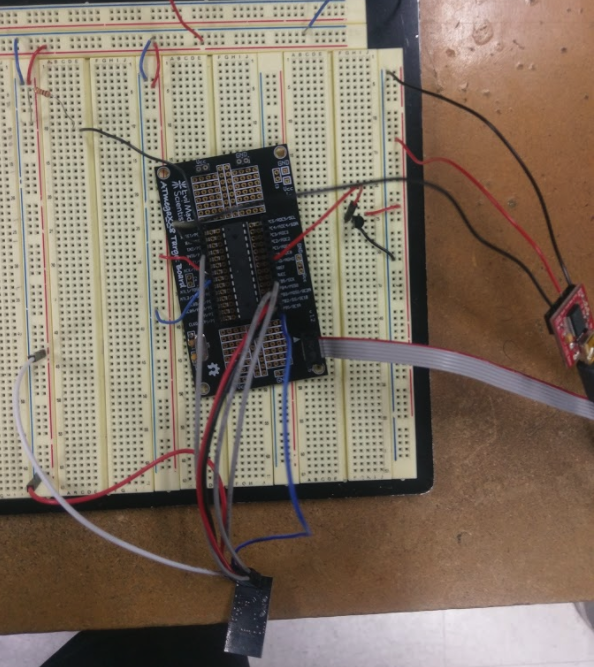
N/A

1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

**Receiver:**

****

**Transmitter:**

****

1. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/watch?v=BUFPaYAD4H8>

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Brian Lopez