CPE301 – SPRING 2019

Design Assignment 5

Student Name: Carlos D. Zuniga

Student #: 2000783713

Student Email: zunigc2@unlv.nevada.edu

Primary Github address: [https://github.com/blassen/](https://github.com/blassen/Dessign%20Assignments/)

Directory: Design Assignments/DA5

**Since I did not find a partner to do the assignment with, I borrowed a breadboard with the microcontroller from a class mate and did the assignment by myself.**

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

* Atmega328P
* Breadboard
* Resistors
* Power supply
* FTDI chip
* NRF24I01

1. **INITIAL/DEVELOPED CODE OF TASK 1/A**

**Transmit code:**

#include <avr/io.h>

#include <avr/interrupt.h>

#define *F\_CPU* 8000000UL

#define UBRR\_9600 51

#include <util/delay.h>

#include <stdbool.h>

#include <string.h>

#include "nrf24l01.h"

void setup\_timer(void);

nRF24L01 \*setup\_rf(void);

volatile unsigned int adc\_temp; //variable to send the ADC value

char outs[20];

volatile bool rf\_interrupt = false;

volatile bool send\_message = false;

void init\_uart(unsigned int ubrr)

{

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

UBRR0L = (unsigned char)ubrr;

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

UCSR0C = (1<<UCSZ00) | (1<<UCSZ01); //USART mode select

}

void USART\_tx\_string( char \*data ) { //print string

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

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

UDR0 = \*data;

data++;

}

}

void adc\_init(void)

{

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

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

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

(1<<ADLAR)| // ADC left Adjust Result

(0<<MUX2)| // Analog Channel Selection Bits

(0<<MUX1)| // ADC0 Pin

(0<<MUX0);

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 Select Bits

(0<<ADPS1)|

(1<<ADPS0);

}

/\* READ ADC PINS\*/

void read\_adc(void)

{

unsigned char i =4; //set i to 4- make 4 readings

adc\_temp = 0; //initialize ADC\_TEMP

while (i--)

{

ADCSRA |= (1<<ADSC);

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

adc\_temp += ADCH; //sum up 4 readings

*\_delay\_ms*(50);

}

adc\_temp = adc\_temp / 4; // Average of four samples

}

int main(void) {

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

bool on = false;

adc\_init(); //initialize ADC

sei();

nRF24L01 \*rf = setup\_rf();

setup\_timer();

while (true) {

read\_adc(); //keep reading temperate from LM34

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;

if (on)

{

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

*memcpy*(msg.data, outs , 3); //copy outs to msg.data

USART\_tx\_string((char \*)msg.data); //print msg.data

USART\_tx\_string("F\r\n"); //print F and line feed

}

else

*memcpy*(msg.data, "OFF", 4);

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;

}

**Receiver code:**

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <string.h>

#include "nrf24l01.h"

#include "nrf24l01-mnemonics.h"

#define *F\_CPU* 8000000UL

#define UBRR\_9600 51

#include <util/delay.h>

nRF24L01 \*setup\_rf(void);

volatile bool rf\_interrupt = false;

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(unsigned int ubrr)

{

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

UBRR0L = (unsigned char)ubrr;

UCSR0B = (1<<TXEN0) | (1<<RXEN0);

UCSR0C = (1<<UCSZ00) | (1<<UCSZ01);

}

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

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

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

UDR0 = \*data;

data++;

}

}

void ADC\_init ()

{

ADMUX = 0;

ADMUX |= (1<<REFS0);

ADCSRA |= (1<<ADPS2) | (1<<ADPS1) |(1<<ADEN);

ADCSRB = 0;

}

int main(void) {

init\_uart(UBRR\_9600);

spi\_init();

*\_delay\_ms*(500);

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

sei();

USART\_tx\_string("connected...\r\n");

nRF24L01 \*rf = setup\_rf();

nRF24L01\_listen(rf, 0, address);

*uint8\_t* addr[5];

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

while (true) {

if (rf\_interrupt) {

rf\_interrupt = false;

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

nRF24L01Message msg;

nRF24L01\_read\_received\_data(rf, &msg);

USART\_tx\_string((char \*)msg.data);

USART\_tx\_string("F\r\n");

}

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;

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

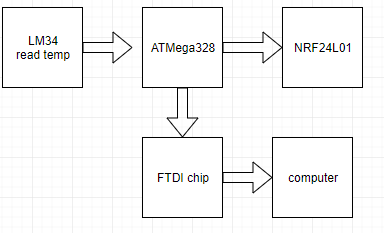
rf\_interrupt = true;

EIFR |= (INTF0);

}

1. **Flow chart**

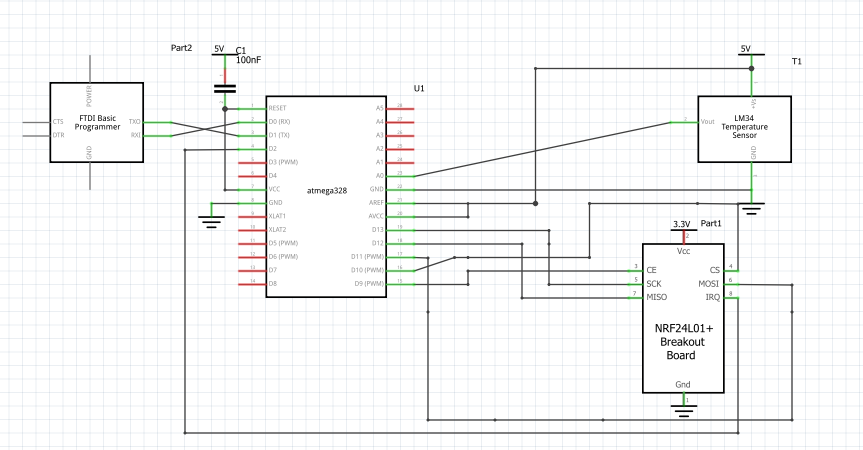
**Transmit:**



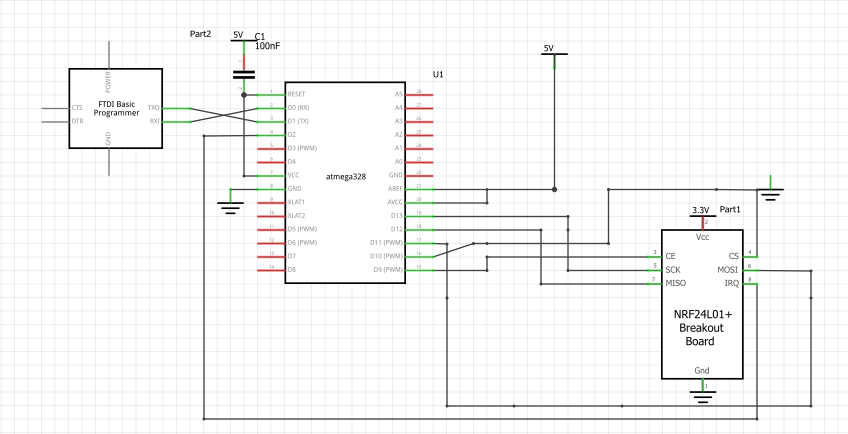
The LM34 is connected to the microcontroller through ADC0 pin and the microcontroller is connected to the FTDI chip to print the value to Putty as well as to the NRF24L01 chip to transfer the data to the receiver to print the temperature to Putty.

The receiver chart flow is identical to the transmit beside the temperature sensor.

1. Schematic- transmit:



Schematic- receiver:

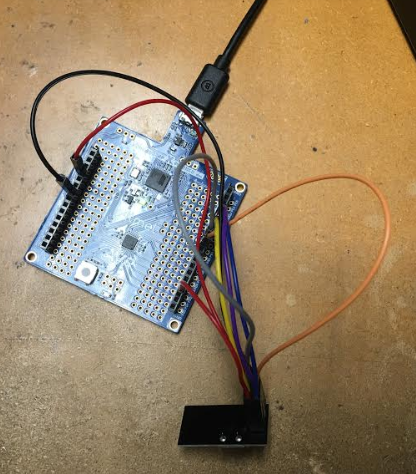


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Received Message (Temperature) and Sent Message(Temperature)



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



1. **VIDEO LINKS OF EACH DEMO**

<https://youtu.be/yNPQa4kFnAg>

1. **GITHUB LINK OF THIS DA**

[**https://github.com/blassen/Design**](https://github.com/blassen/Design)**Assignments/DA5**

**Student Academic Misconduct Policy**

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

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