Damian Cisneros

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 |  |  |
|  |  |  |  |
|  |  |  |  |

Damian Cisneros

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

List of Components used:

ATmega328p

LM34

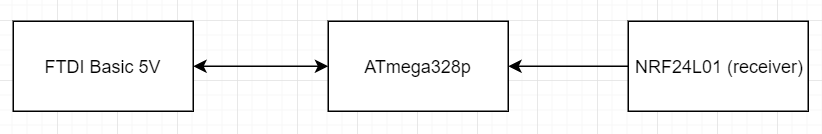
Breadboard

10kΩ resistor

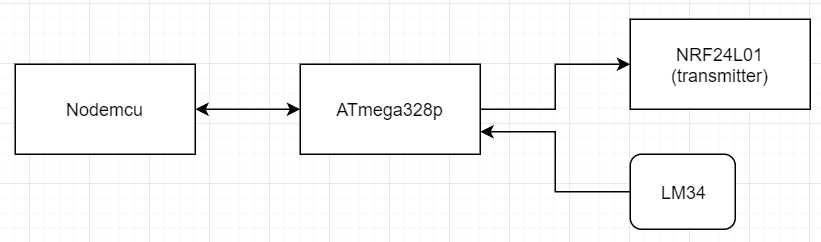
Nodemcu(UART receiving)

FTDI Basic 5V(UART sending)

2x NRF24L01 (1xreceive/1xsend)



Block diagram for receiving NRF circuit



Block diagram for sending NRF circuit

1. **INITIAL/DEVELOPED CODE OF MIDTERM 2 RECEIVER**

//tx

/\*

\* main.c

\*

\* Created: 4/20/2018 1:54:58 PM

\* Author : Damian Cisneros and Jiajian Chen

\* Description : This program monitors temperature using an LM34 sensor. It

\* sends the temperature through RF every 1s and

\* displays it on a serial terminal. Using 8Mhz clock

\*/

#define BAUD 9600

#define F\_CPU 8000000UL

#include <stdlib.h>

#include <stdlib.h>

#include <stdio.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <string.h>

#include <util/delay.h>

#include "nrf24l01.h"

void init\_USART**();**

void setup\_timer**(**void**);**

nRF24L01 **\***setup\_rf**(**void**);**

volatile bool rf\_interrupt **=** false**;**

volatile bool send\_message **=** false**;**

static int put\_char**(**char c**,** FILE **\***stream**);**

static FILE mystdout **=** FDEV\_SETUP\_STREAM**(**put\_char**,** **NULL,** \_FDEV\_SETUP\_WRITE**);**

int ADCvalue**;**

char c**[**10**];**

ISR**(**ADC\_vect**)**

**{**

ADCvalue **=** ADCH**;**

// only need to read the high value for 8 bit

**}**

int main**(**void**)** **{**

stdout **=** **&**mystdout**;** //set the output stream

ADMUX **=** 0**;** // use ADC0

ADMUX **|=** **(**1 **<<** REFS0**);** // use AVcc as the reference

ADMUX **|=** **(**1 **<<** ADLAR**);** // left adjust for 8 bit resolution

ADCSRA **|=** **(**1 **<<** ADPS2**)** **|** **(**1 **<<** ADPS1**)** **|** **(**0 **<<** ADPS0**);** // 128 prescale for 16Mhz

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

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

bool on **=** false**;**

init\_USART**();**

sei**();**

nRF24L01 **\***rf **=** setup\_rf**();**

setup\_timer**();**

**while** **(**true**)** **{**

ADCvalue**=((**ADCvalue**\***5**)/**256**)\***100**;**

**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**){**

dtostrf**(**ADCvalue**,**3**,**0**,**c**);**

memcpy**(**msg**.**data**,**c **,** 6**);**

printf**(**"Sending: "**);**

printf**(**c**);**

printf**(**"F"**);**

printf**(**"\n"**);**

\_delay\_ms**(**1000**);**

**}**

**}**

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 init\_USART**(){**

unsigned int BAUDrate**;**

//set BAUD rate: UBRR = [F\_CPU/(16\*BAUD)]-1

BAUDrate **=** **((**F\_CPU**/**16**)/**BAUD**)** **-** 1**;**

UBRR0H **=** **(**unsigned char**)** **(**BAUDrate **>>** 8**);** //shift top 8 bits into UBRR0H

UBRR0L **=** **(**unsigned char**)** BAUDrate**;** //shift rest of 8 bits into UBRR0L

UCSR0B **|=** **(**1 **<<** RXEN0**)** **|** **(**1 **<<** TXEN0**);** //enable receiver and trasmitter

// UCSR0B |= (1 << RXCIE0); //enable receiver interrupt

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

**}**

static int put\_char**(**char c**,** FILE **\***stream**)**

**{**

**while(!(**UCSR0A **&(**1**<<**UDRE0**)));** // wait for UDR to be clear

UDR0 **=** c**;** //send the character

**return** 0**;**

**}**

1. **INITIAL/DEVELOPED CODE OF MIDTERM 2 TRANSMITTER**

//rx

/\*

\* main.c

\*

\* Created: 4/20/2018 1:54:58 PM

\* Author : Damian Cisneros and Jiajian Chen

\* Description : This program monitors temperature using an LM34 sensor. It

\* reads the temperature received through RF every 1s and

\* displays it on a serial terminal. Using 8Mhz clock

\*/

#include <stdio.h>

#include <stdlib.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <string.h>

#define BAUD 9600

#define F\_CPU 8000000UL

#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**;**

//Used for printf to USART

static int put\_char**(**char c**,** FILE **\***stream**);**

static FILE mystdout **=** FDEV\_SETUP\_STREAM**(**put\_char**,** **NULL,** \_FDEV\_SETUP\_WRITE**);**

void init\_USART();

int main(void) {

stdout = &mystdout; //set the output stream

init\_USART();

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

prepare\_led\_pin();

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);

printf("Received: ");

printf("%" PRId8 "\n",(unsigned int) msg.data); //print temperature received

printf("F");

\_delay\_ms(1000);

}

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) {

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);

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true;

}

void init\_USART(){

unsigned int BAUDrate;

//set BAUD rate: UBRR = [F\_CPU/(16\*BAUD)]-1

BAUDrate = ((F\_CPU/16)/BAUD) - 1;

UBRR0H = (unsigned char) (BAUDrate >> 8); //shift top 8 bits into UBRR0H

UBRR0L = (unsigned char) BAUDrate; //shift rest of 8 bits into UBRR0L

UCSR0B |= (1 << RXEN0) | (1 << TXEN0); //enable receiver and trasmitter

// UCSR0B |= (1 << RXCIE0); //enable receiver interrupt

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

}

static int put\_char(char c, FILE \*stream)

{

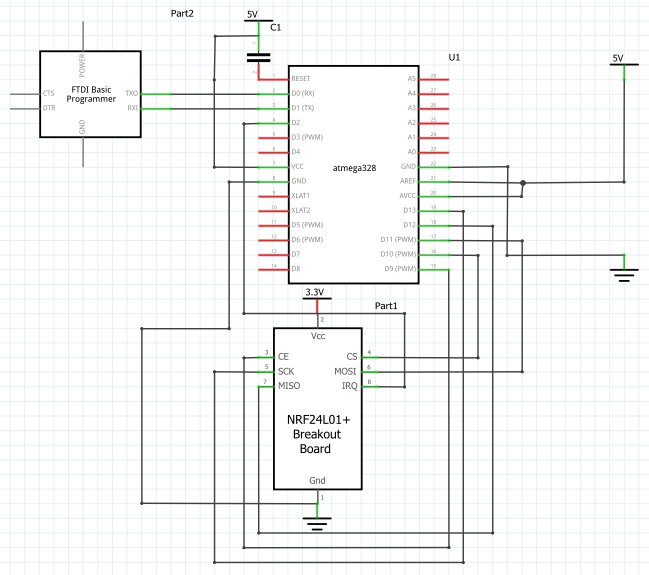
while(!(UCSR0A &(1<<UDRE0))); // wait for UDR to be clear

UDR0 = c; //send the character

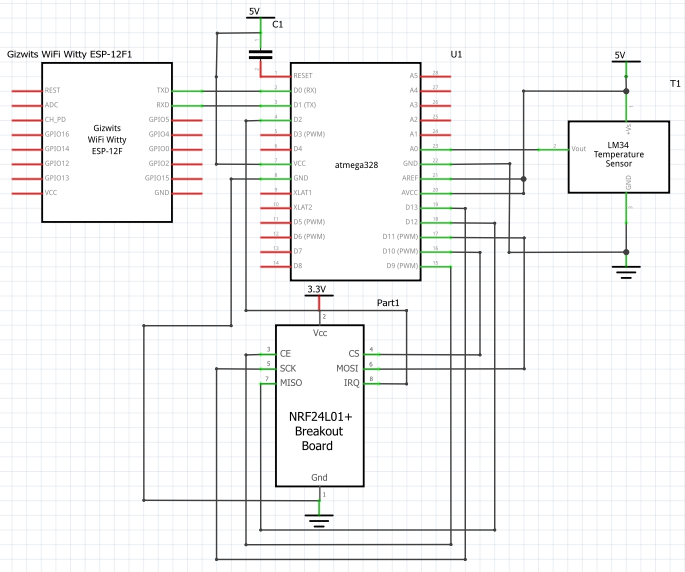
return 0;

}

1. **SCHEMATICS**



Receiver Schematic



Transmitter Schematic

1. **VIDEO LINKS OF EACH DEMO**

Midterm 2 Receiver and Transmitting - <https://youtu.be/_iO0WttPeco>

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”.