Audie Escala

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. | INITIAL CODE OF TASK 2/B |  |  |
| 4. | FLOWCHART OF EACH TASK |  |  |
| 5. | SCREEN SHOTS OF EACH TASK SCHEMATIC |  |  |
| 6. | SCREENSHOTS OF COMPONENTS ON BREADBOARD |  |  |
| 7. | YOUTUBE/GOOGLECODE LINK OF THE MIDTERM |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

Two Atmega328P

Two NRF24L01

One LM34 Temperature Sensor

Two FTDI

1. **INITIAL/DEVELOPED CODE IN C OF TRAMSITTER**

#define *F\_CPU* 16000000UL

#define FOSC 16000000 // Clock Speed

#define BAUD 9600

#define MYUBRR FOSC/16/BAUD-1

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <stdint.h> // need for uint8\_t

#include <util/delay.h> //delay header

#include <string.h>

#include "nrf24l01.h"

#include <stdio.h>

void setup\_timer(void);

nRF24L01 \*setup\_rf(void);

void ADCinit(void);

void USARTinit(void);

void USARTsend(unsigned char);

static int uart\_putchar(char c, *FILE* \*stream)

{

if (c == '\n') uart\_putchar('\r', stream);

loop\_until\_bit\_is\_set(UCSR0A, UDRE0);

UDR0 = c;

return 0;

}

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

volatile float temp;

char c[10];

volatile *uint8\_t* ADCvalue;

volatile bool rf\_interrupt = false;

volatile bool send\_message = false;

int main(void) {

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

USARTinit();

ADCinit();

bool on = false;

sei();

nRF24L01 \*rf = setup\_rf();

setup\_timer();

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) {

send\_message = false;

on = !on;

nRF24L01Message msg;

if (on){

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

*printf*("Temperature: %s\n", c);

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

*\_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);

}

void USARTinit (void)

{

/\* set baud rate \*/

UBRR0H = (MYUBRR>>8); //high value of baud rate

UBRR0L = MYUBRR; // low value of baud rate

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

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

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

*stdout* = &mystdout;

}

void USARTsend(unsigned char Data) // function for sending data to the stream

{

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

UDR0=Data;

}

void ADCinit(void)

{

ADMUX |= (1 << REFS0); //use AVcc as ref

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // 128 Prescale for 16MHz

ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable

ADCSRB = 0; // Free running mode

ADCSRA |= (1 << ADEN); // Enable the ADC

ADCSRA |= (1 << ADIE); // Enable interrupts

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

}

ISR (ADC\_vect)

{

/\*

convert the read ADCvalue to temperature

500.0=>(Vref \* 100)=>(5V \* 100)

divide by 1024, the max for the ADC values (0-1024)

\*/

ADCvalue = ADC;

ADCvalue = (ADCvalue)\*(500.0/1024.0);

/\* converts value into ascii \*/

temp = ADCvalue;

}

// each one second interrupt

ISR(TIMER1\_COMPA\_vect)

{

send\_message = true;

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true;

}

1. **DEVELOPED CODE IN C OF RECIEVER**

#define *F\_CPU* 8000000UL

#define BAUD 9600

#include <stdio.h>

#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 init\_USART();

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

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

int main(void) {

*stdout* = &mystdout;

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

prepare\_led\_pin();

init\_USART();

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*("Temperature: ");

*printf*((char \*)msg.data);

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

*\_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 BAUD\_rate;

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

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

UBRR0H = (unsigned char) (BAUD\_rate >> 8); //shift top 8 bits into UBRR0H

UBRR0L = (unsigned char) BAUD\_rate; //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)

{

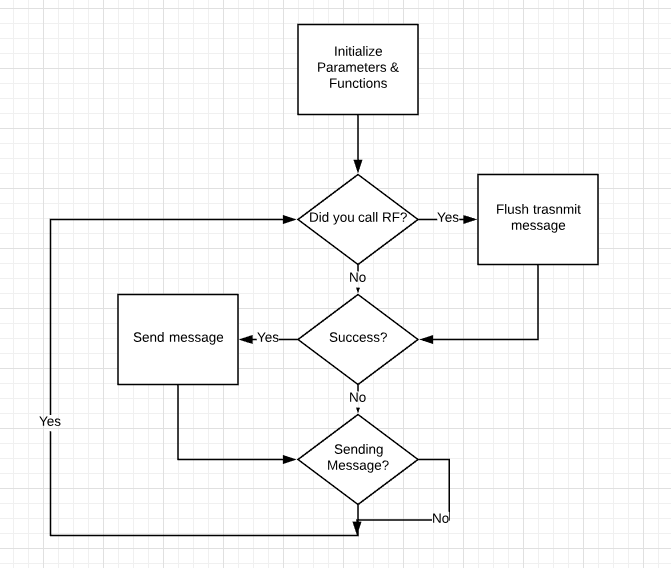
while(!(UCSR0A &(1<<UDRE0))); // wait for UDR to be clear

UDR0 = c; //send the character

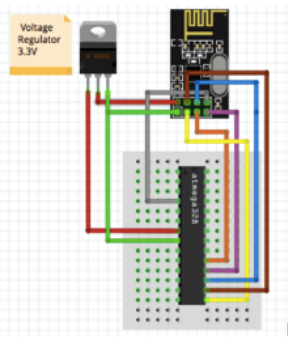
return 0;

}

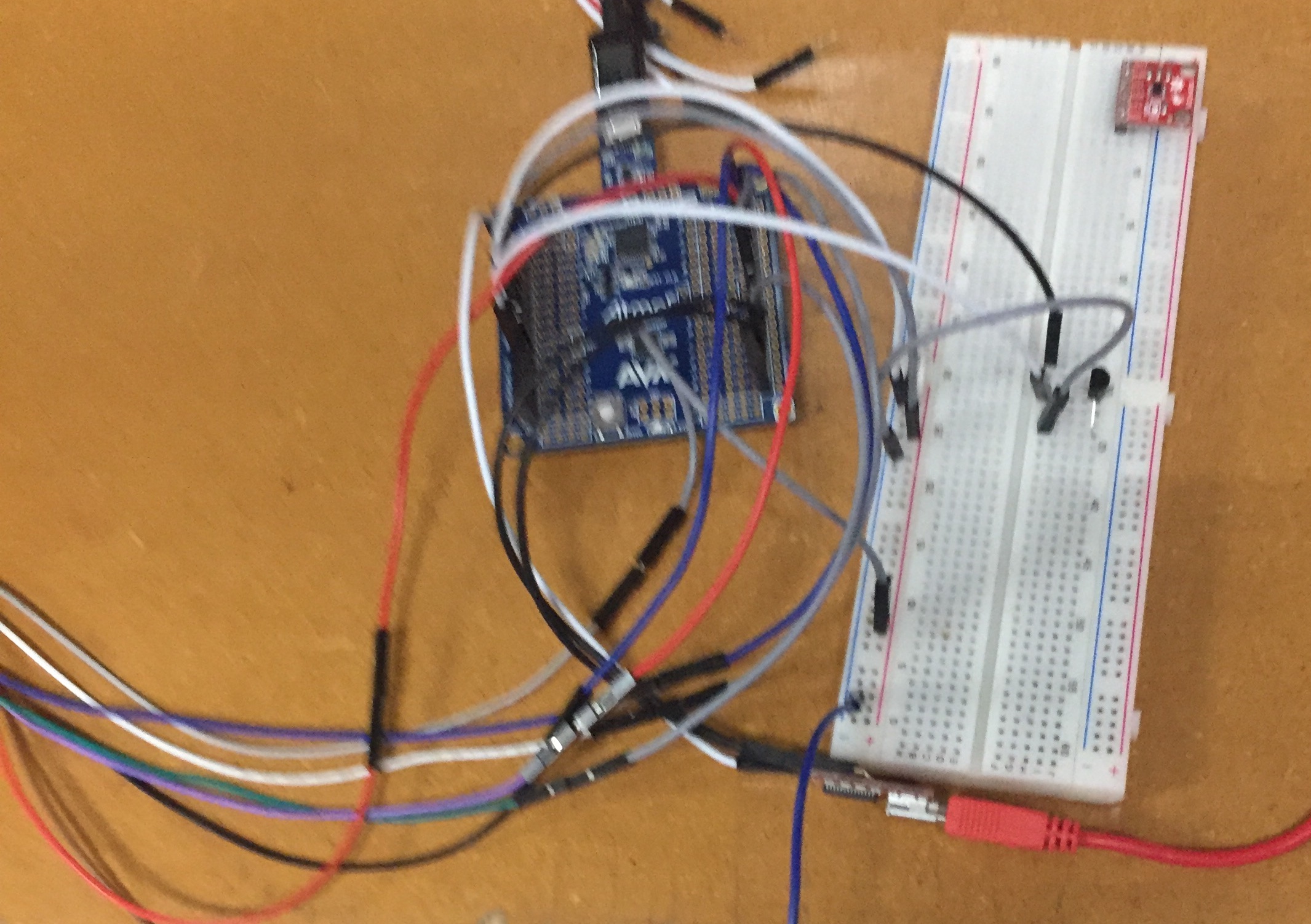
1. **FLOWCHART OF EACH TASK**



1. **SCREENSHOT OF EACH SCHEMATIC**



1. **SCREENSHOT OF EACH TASK**



1. **YOUTUBE/GITHUB LINK OF MIDTERM**

https://www.youtube.com/watch?v=1sYaCBC3vv4

https://github.com/escalaa/Midterm2

**Student Academic Misconduct Policy**

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

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

Audie Escala