Monty Sourjah

CPE301 – SPRING 2018

Design Assignment Midterm2

**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 | Y |  |
| 2. | INITIAL CODE OF TASK 1/A | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E | Y |  |
| 4. | SCHEMATICS |  |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 5. | SCREENSHOT OF EACH DEMO | Y |  |
| 6. | VIDEO LINKS OF EACH DEMO | Y |  |
| 7. | GOOGLECODE LINK OF THE DA | Y |  |
|  |  |  |  |
|  |  |  |  |

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

List of Components used:

Arduino Mega2560 X2, NRF34L01 X2 Transreceiver,3.3 Power supply, Jumper wires LM34 temperature sensor.

Block diagram with pins used in the Atmega328P

Transmitter:



IRQ MISO PIN50 PIN21

NRF24L01

Trans receiver 2.4GHz

CSN PIN48

LM34 Temperature Sensor

Arduino Mega 2560 Board

CE PIN53

MOSI PIN51

SCK PIN52

3.3V GND GND A0 GND Signal 3.3V

GND

Voltage Regulator 3.3V

3.3V 9 Volts DC Input

Transmit Data

Receiver:



IRQ MISO PIN21 PIN50

NRF24L01

Trans receiver 2.4GHz

CSN PIN48 Display Data in the PC

Arduino Mega 2560 Board

CE PIN53

MOSI PIN51

SCK PIN52

3.3V GND GND

GND

Voltage Regulator 3.3V

3.3V 9 Volts DC Input

Receive Data

mp

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

Transmitter:

/\*

\* CpE301 - Midterm 2-TX.c

\*

\* Created: 17-Apr-18 3:28:59 PM

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include "nrf24l01.h"

#define FOSC 16000000 // Clock Speed

#define BAUD 9600

#define MYUBRR FOSC/16/BAUD - 1

void setup\_timer(void);

nRF24L01 \*setup\_rf(void);

void usart\_init();

int USART0SendByte(char u8Data);

void USARTSendStr(char\* \_str);

void adc\_init(void);

volatile bool rf\_interrupt = false;

volatile bool send\_message = false;

char\* tempTemplate = "TX: Temperature is %u degrees Fahrnheit\n";

int main(void)

{

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

*int16\_t* adc\_value;

*int8\_t* temp;

char printBuffer[64];

nRF24L01 \*rf = setup\_rf();

setup\_timer();

adc\_init();

usart\_init();

sei();

while (1)

{

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;

/\* Start the ADC conversion \*/

ADCSRA |= (1 << ADSC);

/\* Wait for completion. ADSC reads 1 while still in progress \*/

while (ADCSRA & (1<<ADSC));

/\* save ADC value to a variable \*/

adc\_value=ADCL;

adc\_value = (ADCH<<8) + adc\_value;

/\*

For LM34:

Vout = + 10.0 mV/ºF

So the temperature will be:

temp = adc\_value \* (5V/adcresolution) / 10.0mV

or:

\*/

temp = ((float)adc\_value \* 5 / 1024) / 0.010;

*sprintf*(printBuffer, tempTemplate, temp);

USARTSendStr(printBuffer);

nRF24L01Message msg;

/\* We will send only the value in alfa format to the Receiver \*/

*itoa*(temp, (char\*)msg.data, 10);

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

nRF24L01\_transmit(rf, to\_address, &msg);

}

}

}

nRF24L01 \*setup\_rf(void) {

nRF24L01 \*rf = nRF24L01\_init();

rf->ss.port = &PORTB;

rf->ss.pin = PINB2;

rf->ce.port = &PORTB;

rf->ce.pin = PINB1;

rf->sck.port = &PORTB;

rf->sck.pin = PINB5;

rf->mosi.port = &PORTB;

rf->mosi.pin = PINB3;

rf->miso.port = &PORTB;

rf->miso.pin = PINB4;

EICRA |= (1<<ISC01); // Falling edge generates interrupt

EIMSK |= (1<<INT0); // Use INT0 (PD2)

nRF24L01\_begin(rf);

return rf;

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true;

}

// setup timer to trigger interrupt every second when at 16MHz

void setup\_timer(void) {

TCCR1A = 0;

// set up timer with CTC mode and prescaling = 256

TCCR1B |= (1 << WGM12)|(1 << CS12);

// initialize counter

TCNT1 = 0;

/\*

Initialize compare value

With prescalar = 256, the frequency is 16,000,000 Hz / 256 -> period = 0.000016 sec

TimerCount=Requireddelay/period -1

For example, for 1 sec delay -> TimerCount = 1/0.000016 -1 = 62499

Let's consider this as the value for our delay.

\*/

OCR1A = 62499; // value for 1 second delay

// Enable compare interrupt

TIMSK1 |= (1 << OCIE1A);

}

// each one second interrupt

ISR(TIMER1\_COMPA\_vect) {

send\_message = true;

}

void usart\_init()

{

UBRR0H = (MYUBRR) >> 8;

UBRR0L = MYUBRR;

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

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

}

int USART0SendByte(char u8Data)

{

//wait while previous byte is completed

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

// Transmit data

UDR0=u8Data;

return 0;

}

void USARTSendStr(char\* \_str)

{

int thesize=*strlen*(\_str);

for (*uint8\_t* i=0; i<thesize;i++)

{

USART0SendByte(\_str[i]);

}

}

void adc\_init(void)

{

ADMUX = 0; // use ADC0

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

/\*

ADC operates within a frequency range between 50Kz and 200Kz

With a prescaler of 128 at CPU clock frequency of 16Mz, we will be in range:

F\_ADC=F\_CPU/128 = 125

So:

\*/

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // 128 prescaler for CPU@16Mhz

ADCSRB = 0; // 0 for free running mode

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

}

Receiver:

/\*

\* CpE301 - Midterm 2-RX.c

\*

\* Created: 17-Apr-18 3:39:13 PM

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

//#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "nrf24l01.h"

#include "nrf24l01-mnemonics.h"

nRF24L01 \*setup\_rf(void);

void process\_message(char \*message);

void usart\_init();

int USART0SendByte(char u8Data);

volatile bool rf\_interrupt = false;

char\* tempTemplate = "RX: Temperature is %s Fahrenheit\n";

void USARTSendStr(char\* \_str);

#define FOSC 16000000 // Clock Speed

#define BAUD 9600

#define MYUBRR FOSC/16/BAUD - 1

int main(void) {

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

usart\_init();

nRF24L01 \*rf = setup\_rf();

sei();

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);

/\* Write to UART TX for display on a RS232 terminal \*/

process\_message((char \*)msg.data);

}

nRF24L01\_listen(rf, 0, address);

}

}

return 0;

}

nRF24L01 \*setup\_rf(void) {

nRF24L01 \*rf = nRF24L01\_init();

DDRB |= (1<<DDB1);

rf->ss.port = &PORTB;

rf->ss.pin = PINB2;

rf->ce.port = &PORTB;

rf->ce.pin = PINB1;

rf->sck.port = &PORTB;

rf->sck.pin = PINB5;

rf->mosi.port = &PORTB;

rf->mosi.pin = PINB3;

rf->miso.port = &PORTB;

rf->miso.pin = PINB4;

// interrupt on falling edge of INT0 (PD2)

EICRA |= (1<<ISC01);

EIMSK |= (1<<INT0);

nRF24L01\_begin(rf);

return rf;

}

void process\_message(char \*message) {

char printBuffer[64];

*sprintf*(printBuffer, tempTemplate, message);

USARTSendStr(printBuffer);

}

// nRF24L01 interrupt

ISR(INT0\_vect) {

rf\_interrupt = true;

}

void usart\_init()

{

UBRR0H = (MYUBRR) >> 8;

UBRR0L = MYUBRR;

UCSR0B |= (1 << TXEN0); // Enable Transmitter

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

}

int USART0SendByte(char u8Data)

{

//wait while previous byte is completed

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

// Transmit data

UDR0=u8Data;

return 0;

}

void USARTSendStr(char\* \_str)

{

int thesize=*strlen*(\_str);

for (*uint8\_t* i=0; i<thesize;i++)

{

USART0SendByte(\_str[i]);

}

};

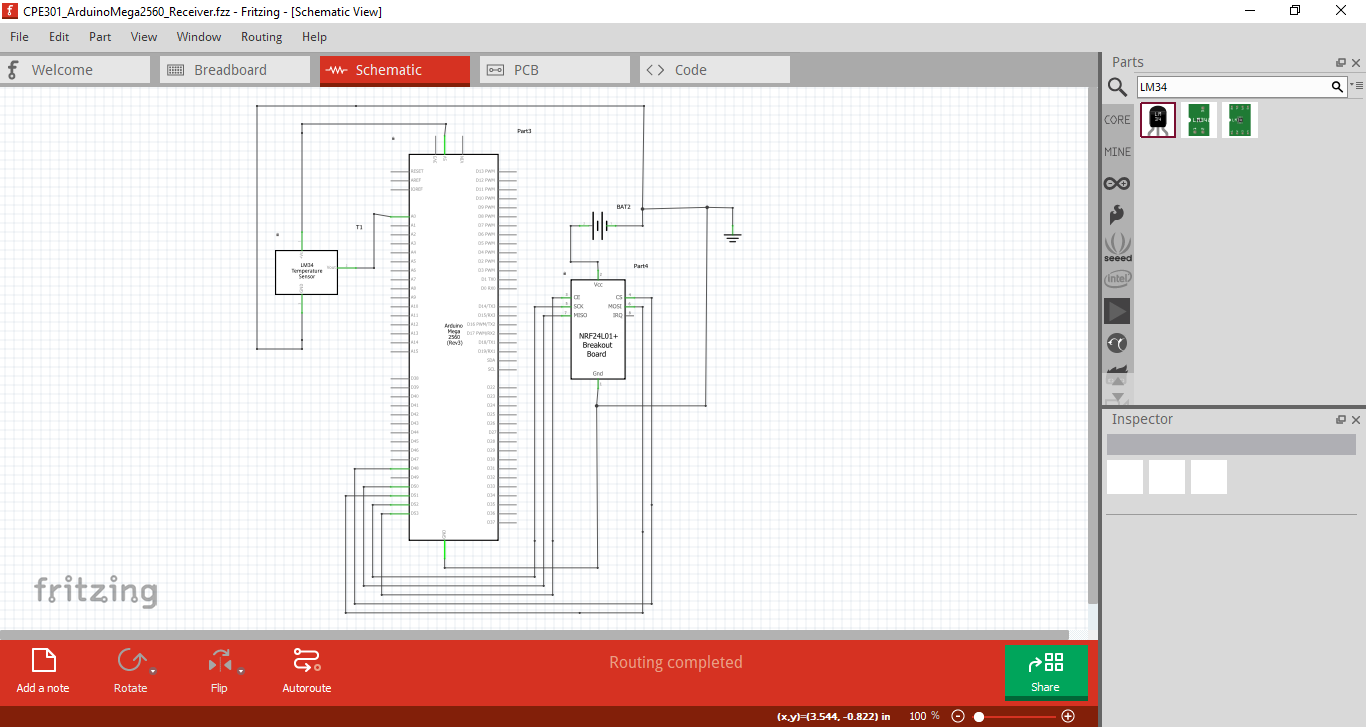
1. **MODIFIED CODE OF TASK 2/A from TASK 1/A**

Insert only the modified sections here. Use more sections if needed

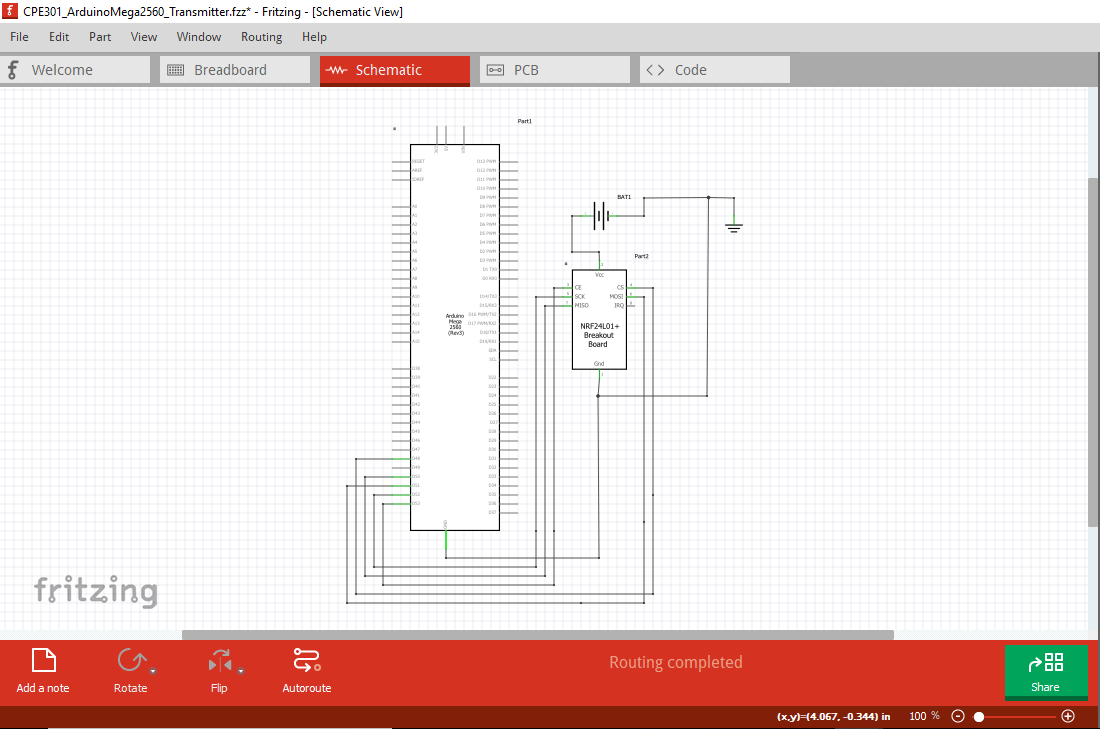
N/A

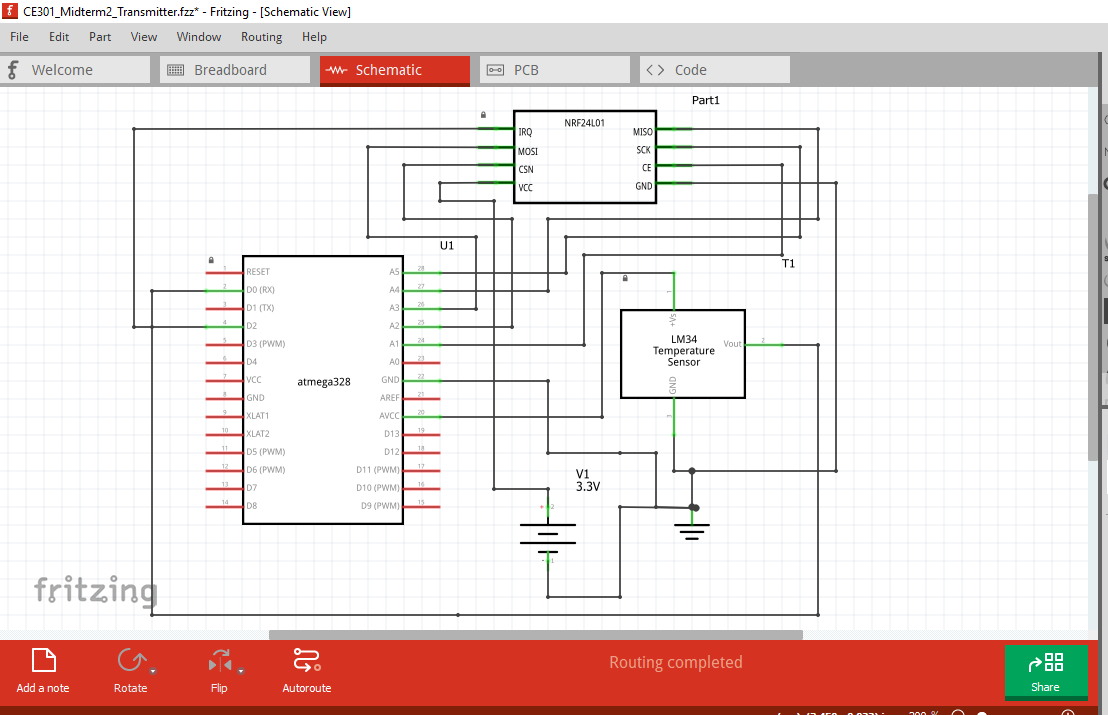
1. **SCHEMATICS**

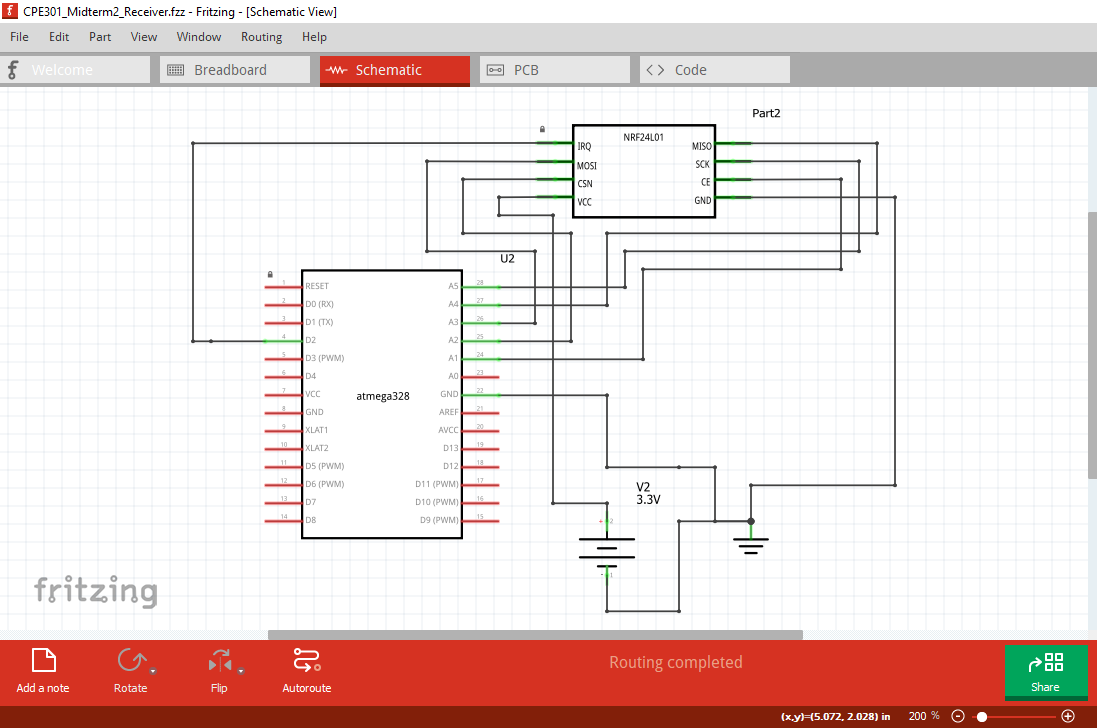
Transmitter: Arduino Mega 2560



Transmitter: Xplained Mini 328p

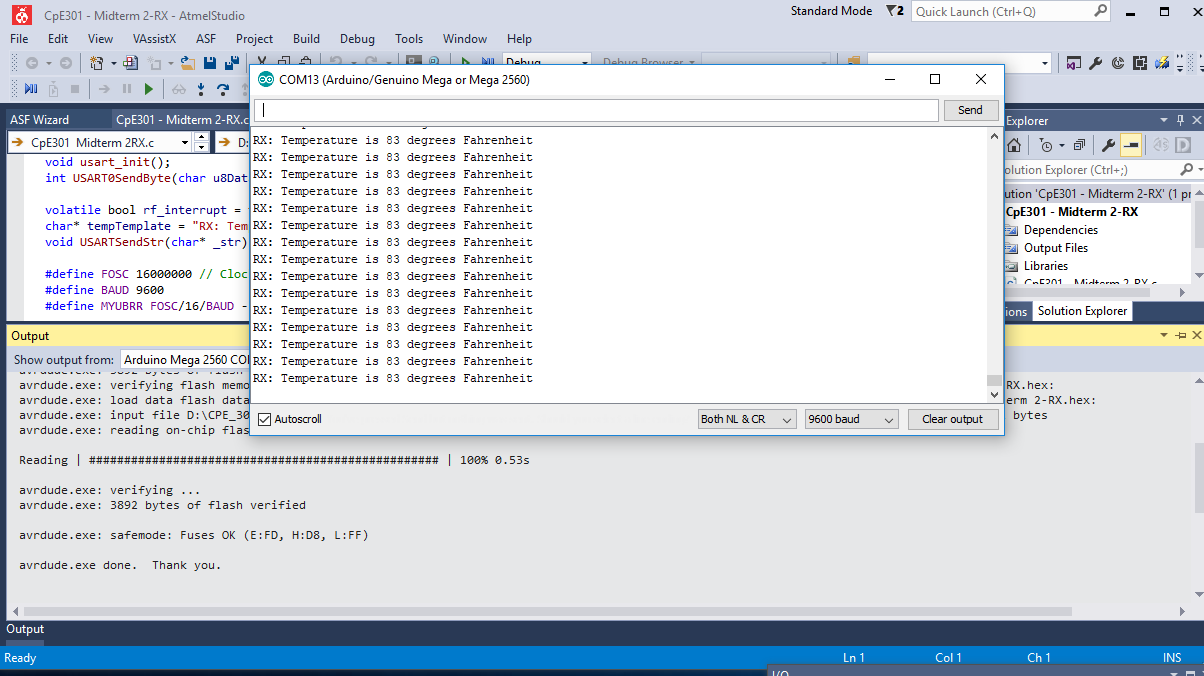




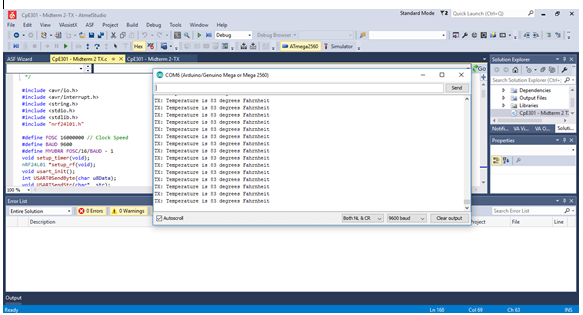


1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Receiver:

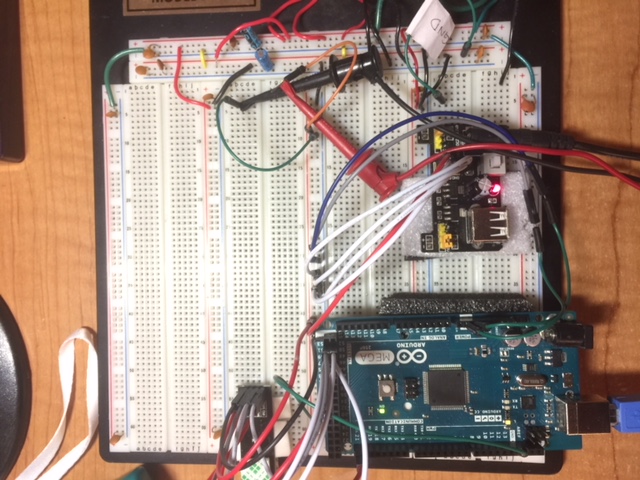


Transmitter:

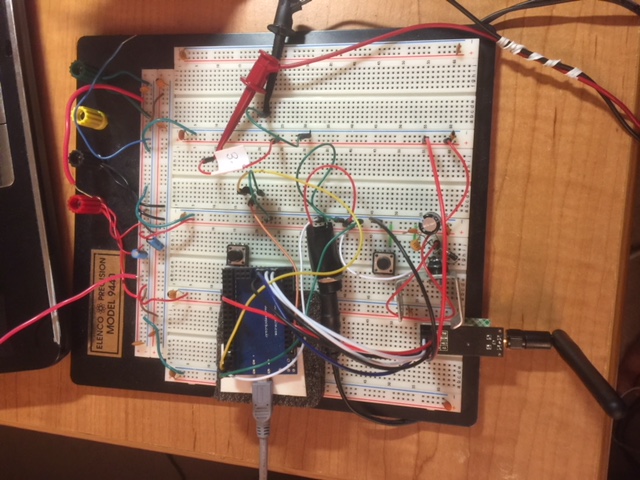


**SCREENSHOT OF EACH DEMO (BOARD SETUP)**

Receiver:



Transmitter:



1. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/watch?v=QAiWWb2t2L8>

1. **GITHUB LINK OF THIS DA**

<https://github.com/ballasl/CPE301_Midterm2>

**Student Academic Misconduct Policy**

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

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

Monty Sourjah