Trace Stewart
Guillermo Galvez

CPE301 – SPRING 2018

Midterm 2

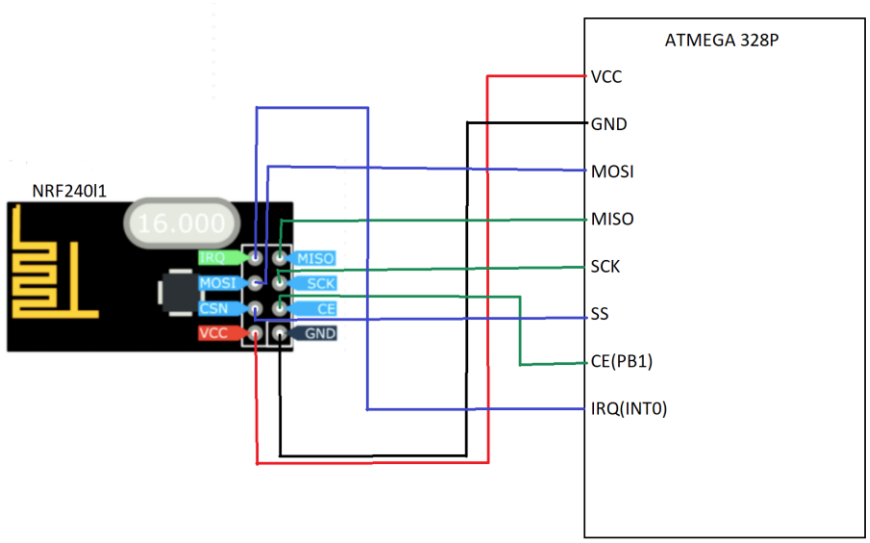
DO NOT REMOVE THIS PAGE DURING SUBMISSION:

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		

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

The Components used in this lab we 2 Atmega 328p's and 2 NRF24l01 modules.



2. INITIAL/DEVELOPED CODE OF TASK 1

```
/*
* Midterm2_Transmit.c
*
* Created: 4/12/2018 8:55:33 AM
* Author : trace / guillermo
*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"
#include <util/delay.h>

#define F_CPU 1000000UL

#define FOSC 16000000
#define BAUD  9600
#define MYUBRR FOSC/16/BAUD-1


void setup_timer(void);
```

```
nRF24L01 *setup_rf(void);
void initUART();
void ADC_Init();
void writeChar(char *c);
void USART_Init();

volatile bool rf_interrupt = false;
volatile bool send_message = false;
volatile uint8_t adcValue;

char seeTemp[3];
char *seeTemp2;


int main(void) {
    uint8_t to_address[5] = { 0x78, 0x78, 0x78, 0x78, 0x78 };
    bool on = false;



    ADC_Init();
    setup_timer();
    USART_Init();

    nRF24L01 *rf = setup_rf();


    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)
            {
            memcpy(msg.data, seeTemp2, 3);
            }

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

```
            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);
    TIMSK1 |= _BV(TOIE1);
    OCR1A = 15624;
    //OCR1A = 33000;
    TCCR1B |= _BV(CS10) | _BV(CS11);
}

//ADC initialization function
void ADC_Init()
{
// initialize ADC
        DDRC = 0;               // Set PORTC as input for adc
        DIDR0 = 0x1;        // Disable digital input on ADC0 pin
        ADMUX = 0;              // ADC0 (PC.0) used as analog input
        ADMUX |= (1 << REFS0);    // use AVcc as the reference
        ADMUX |= (1 << ADLAR);    // Right adjust for 8 bit resolution
```

```
        ADCSRA = 0x87;          // Enable ADC, system clock, 10000111
        ADCSRB = 0x0;       // Free running mode

        sei();                  //enable global interrupts


}//end ADC_Init

// each one second interrupt
ISR(TIMER1_COMPA_vect) {
    send_message = true;

            char temperature[6];

    TIFR1 |= (1 << TOV1);      //Clr Flag
    //fifteenPlus++;


    float  lm34_0;           //For ASCII Temp output

    ADCSRA |= (1 << ADSC);                          //Start Conversion
    while((ADCSRA & (1 << ADIF)) == 0);              //Wait for conversion to finish

    //Conversion to °F
    lm34_0 = (ADCH * 5.0 / 0x100) * 100.0;     //(ADC * 5 = 200 /256) * 100
    dtostrf(lm34_0, 3, 0, temperature);                    //Float to char conversion
        seeTemp2 = temperature;
}

// nRF24L01 interrupt
ISR(INT0_vect) {
    rf_interrupt = true;

}


void writeChar(char *c) {

int i = 0;
while(i<3){
    UDR0 = c[i];            // Display character on serial (i.e., PuTTY) terminal
    _delay_ms(10);          // delay for 200 ms
    i++;
}

}
```

```
void USART_Init()
{

    /*Set Baud Rate*/
    UBRR0H = (MYUBRR>>8);     //Shift MSB "top" of UBRR0H 0100 0100 >> 8 -> UBRR0H 0000 0000
    UBRR0L = MYUBRR;          //UBRR0L 0100 0100

    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);   //Enable Rec and Trans
    UCSR0B |= (1 << RXCIE0);                 //Enable Rec INT
    UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); //Set frame 8-bit, 1 STP
}//end USART_int



/*
* Midterm2_Receive.c
*
* Created: 4/15/2018 2:47:49 PM
* Author : trace / guillermo
*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>
#include <string.h>
#include <util/delay.h>
#include <stdlib.h>
#include "nrf24l01.h"
#include "nrf24l01-mnemonics.h"

#define F_CPU 1000000UL

#define FOSC 16000000
#define BAUD  9600
#define MYUBRR FOSC/16/BAUD-1


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);
void USART_Init();
void writestring(char *c);
void writeChar(unsigned char c);

volatile bool rf_interrupt = false;
```

```
int main(void) {

	USART_Init();	// Initialize UART

	bool on = false;

	uint8_t address[5] = { 0x78, 0x78, 0x78, 0x78, 0x78 };
	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);
				if(true){
				writestring("Temperature: ");
				writestring(msg.data);
				writestring("F");
				writestring("\r");

				}
			}

			nRF24L01_listen(rf, 0, address);
		}
	}

	return 0;
}

void USART_Init()
{

	/*Set Baud Rate*/
	UBRR0H = (MYUBRR>>8);	//Shift MSB "top" of UBRR0H 0100 0100 >> 8 -> UBRR0H 0000 0000
	UBRR0L = MYUBRR;		//UBRR0L 0100 0100

	UCSR0B |= (1 << RXEN0) | (1 << TXEN0);	//Enable Rec and Trans
	UCSR0B |= (1 << RXCIE0);			//Enable Rec INT
	UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); //Set frame 8-bit, 1 STP
}//end USART_int
```

```
void writeChar(unsigned char c) {
	UDR0 = c;		// Display character on serial (i.e., PuTTY) terminal
	_delay_ms(400);		// delay for 200 ms
}

void writestring(char *c){
	unsigned int i = 0;
	while(c[i] != 0)
	writeChar(c[i++]);
}


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;
}
```

3. SCHEMATICS

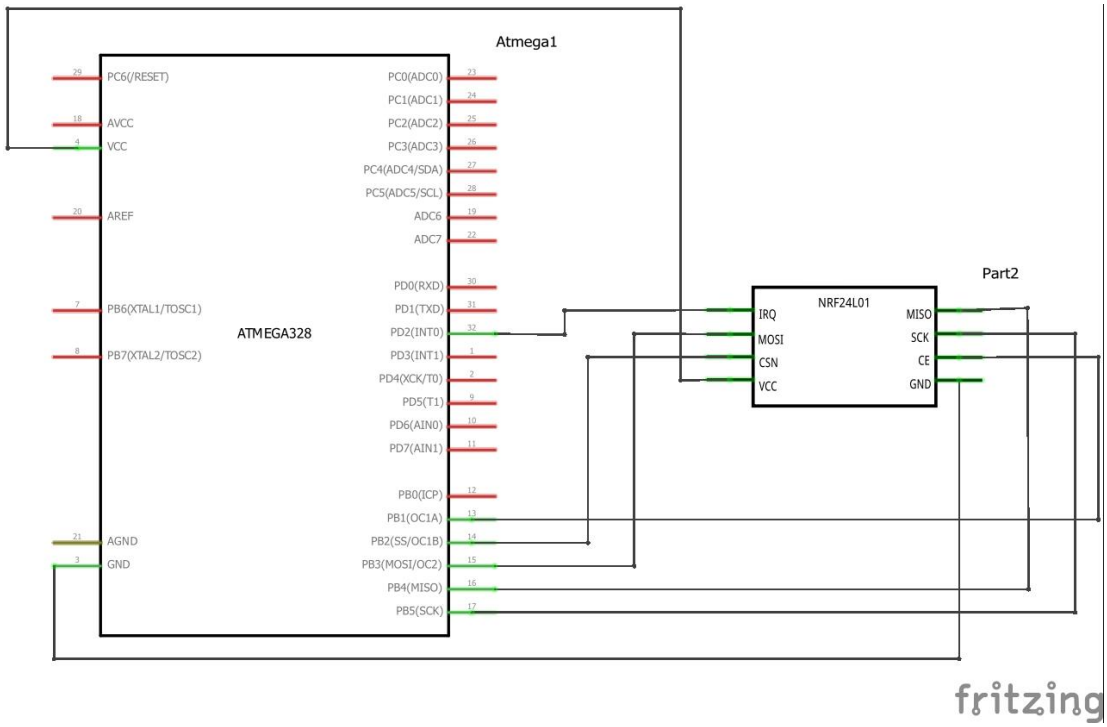


Figure 1: Receiver Schematic

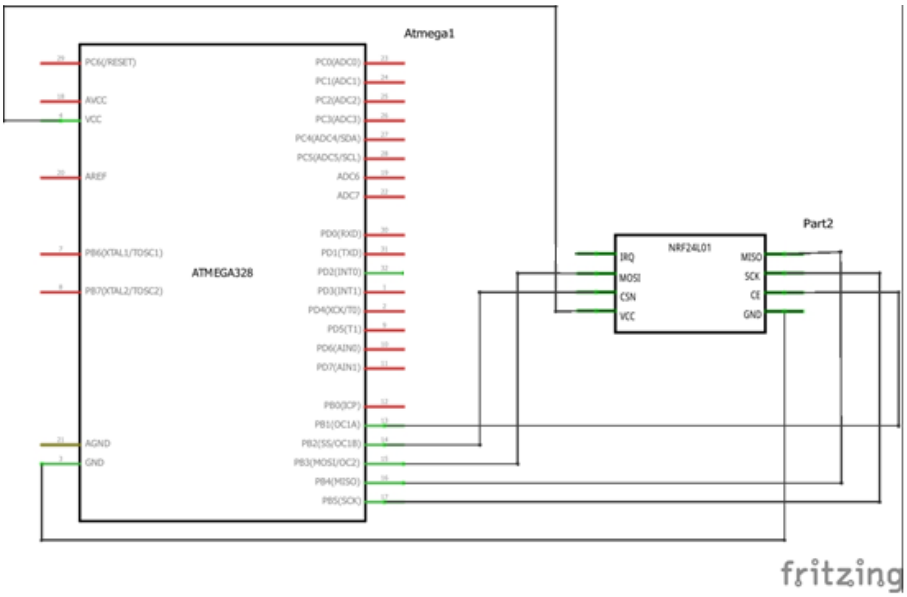


Figure 2: Sender Schematic

4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

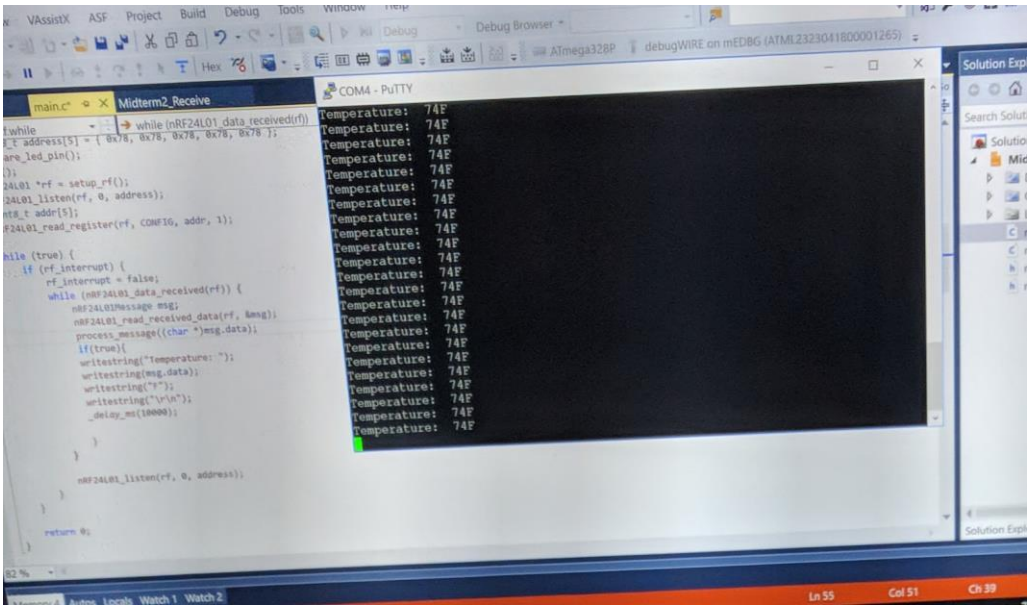


Figure 3: Putty output from Receiver

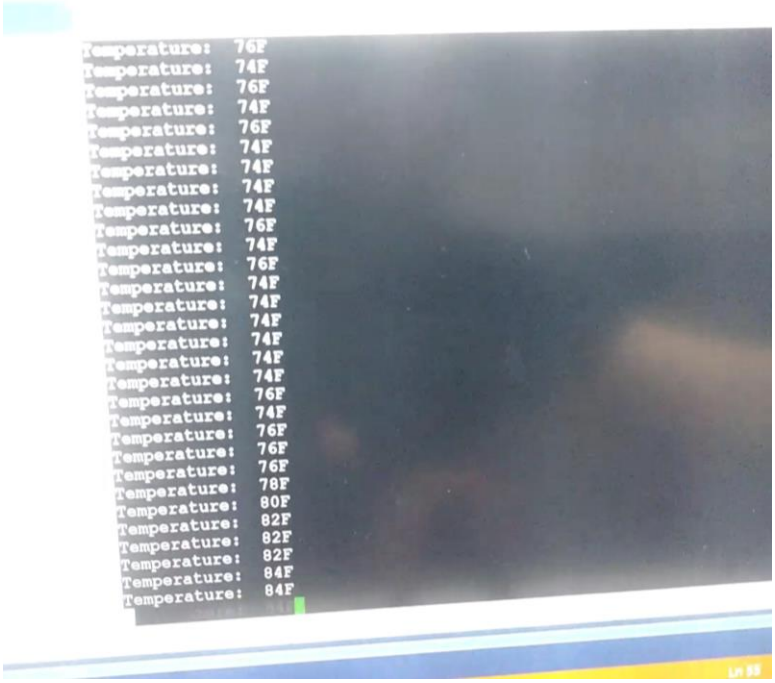


Figure 4: Putty output from receiver

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

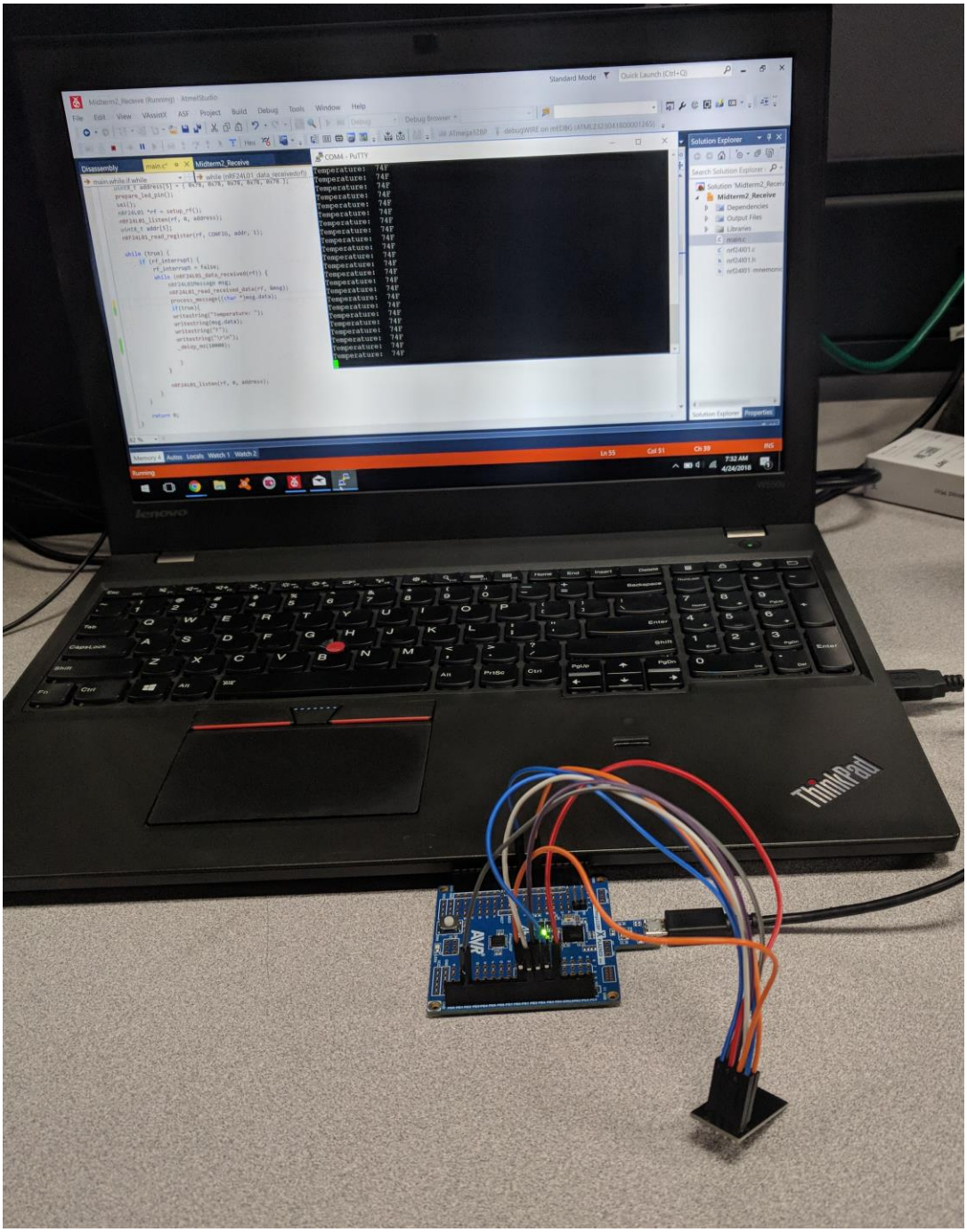


Figure 5: Receiver

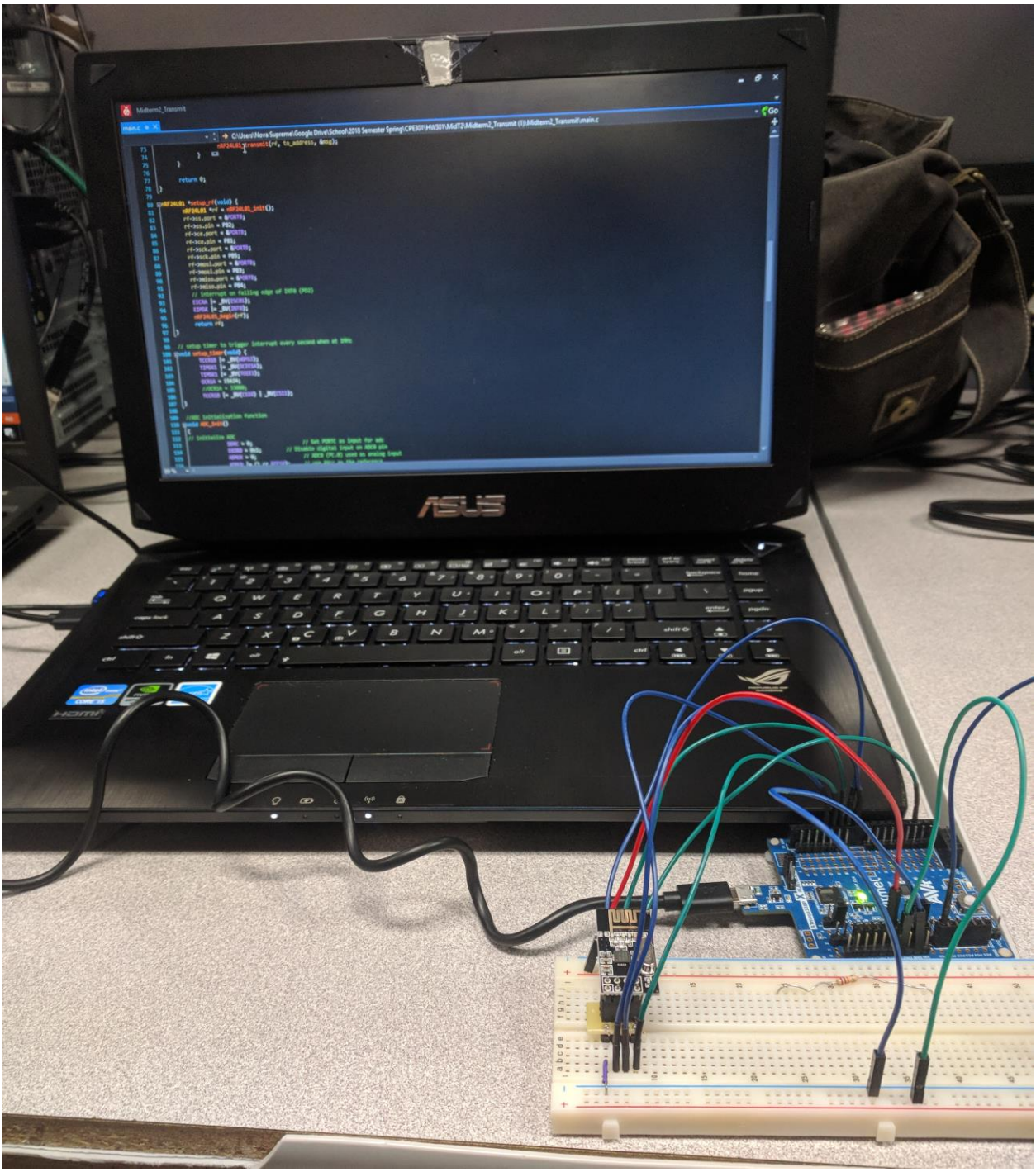


Figure 6: Sender

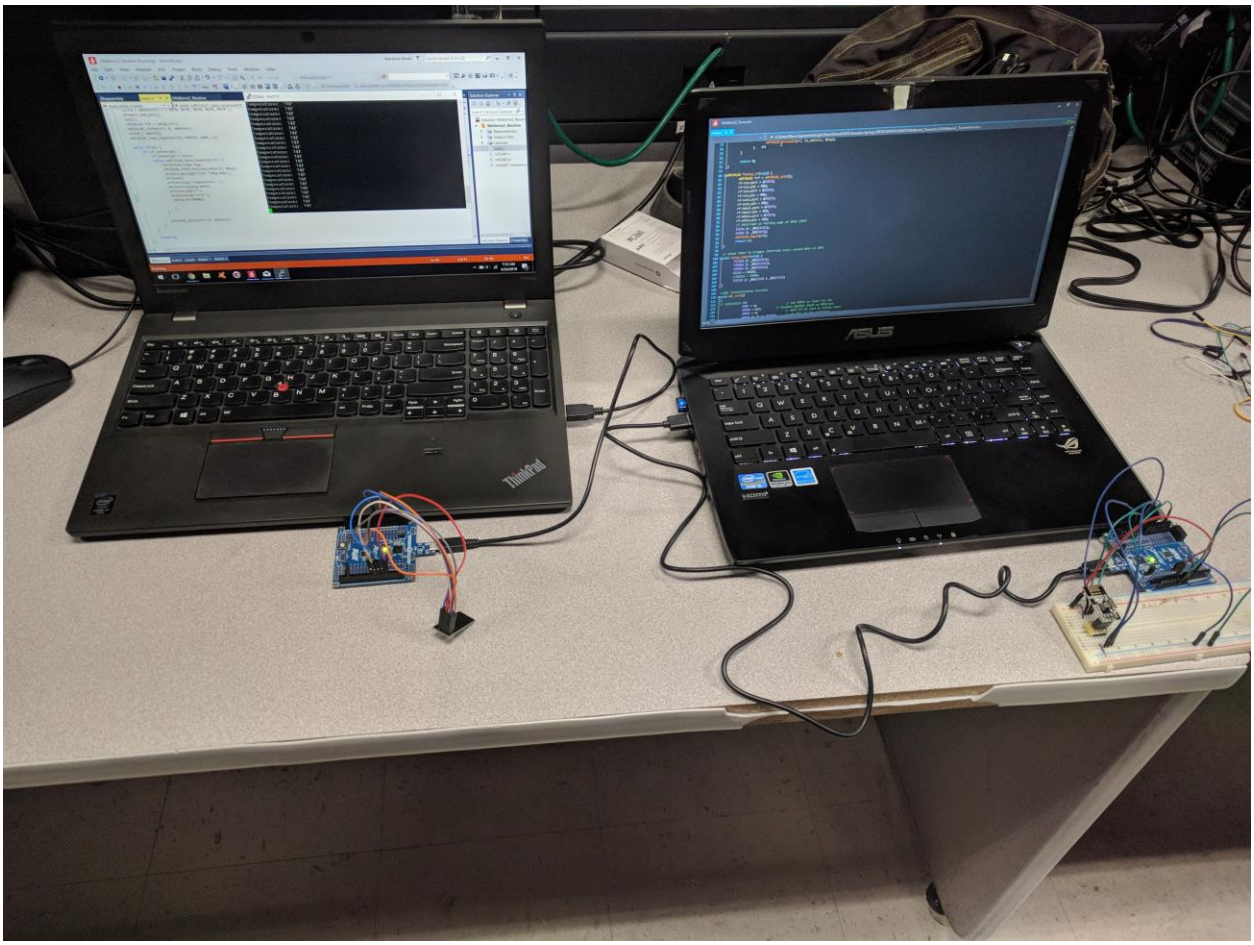


Figure 7: Sender and Receiver

6. VIDEO LINKS OF EACH DEMO

https://www.youtube.com/watch?v=PoDQDhIuC0Q – Trace Stewart Video

7. GITHUB LINK OF THIS DA

https://github.com/TraceStewart/epc103gnirps8102vlnu/tree/master/Midterm_2

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Trace Stewart
Guillermo Galvez