Bryan Takemoto

CPE301 – SPRING 2018

Mid-Term 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 |  |  |
| 2. | C CODE OF TRANSMITTER |  |  |
| 3. | C CODE OF RECEIVER |  |  |
| 4. | BREADBOARD AND SCHEMATIC FOR TRANSMITTER |  |  |
| 5. | BREADBOARD AND SCHEMATIC FOR RECEIVER |  |  |
| 6. | PARTICIPANTS |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **COMPONENTS LIST**

List of Components used:

1. LM34
2. NRF24L01
3. FTDI BASIC
4. ATMEGA328
5. **C CODE OF TRANSMITTER**

#define *F\_CPU* 16000000UL // CPU Speed for delay

#define FOSC 16000000 // Clock speed

#define BAUD 115200 // Desire baud rate

#define MYUBRR FOSC/8/BAUD-1 // Formula to set the baud rate [Double Transmission Rate]

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdlib.h>

#include <stdint.h>

#include <stdbool.h>

#include <string.h>

#include "nrf24l01-mnemonics.h"

#include "nrf24l01.h"

void setup\_timer(void);

void UART\_TX(char \*data);

nRF24L01 \*setup\_rf(void);

volatile bool rf\_interrupt = false;

volatile bool send\_message = false;

volatile char ADCvalue;

int main(void)

{

// Variables

*uint8\_t* i = 0; // Iterative variable

char temp[5]; // Temperature string buffer

char LF = '\n'; // Line feed

DDRC = (1 << 5);

// NRF Settings

*uint8\_t* to\_address[5] = {0x02, 0x04, 0x06, 0x08, 0x0A}; // Device address

sei(); // Global interrupt enabled

nRF24L01 \*rf = setup\_rf(); // Initialize the NRF

setup\_timer();

// ADC Settings

ADMUX = 0; // Use ADC0

ADMUX |= (1 << REFS0); // AVcc is reference with ARef connected to external capacitor

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);// 16 MHz with prescalar of 128

ADCSRA |= (1 << ADATE); // Enable auto trigger

ADCSRB = 0; // Free running settings for auto trigger

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

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

// UART Settings

UBRR0H = ((MYUBRR) >> 8); // Set baud rate for UPPER Register

UBRR0L = MYUBRR; // Set baud rate for LOWER Register

UCSR0A |= (1 << U2X0); // Double UART transmission speed

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

UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Frame: 8-bit Data and 1 Stop bit

while (1)

{

// ADC Conversion

while((ADCSRA & (1 << ADIF)) == 0); // Wait for ADC conversion

ADCvalue = (ADC >> 1); // Assign the temperature

// Check if the message has been successfully transmitted

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; // Reset message flag

nRF24L01Message msg; // Message structure to be transmitted

i = 0; // Reset iterative variable

*itoa*(ADCvalue, temp, 10); // Convert integer value into ASCII

// Transmit the temperature to terminal [NULL terminated]

while(temp[i] != 0)

{

UART\_TX(&temp[i]); // Sends temperature to terminal

i++; // Size of the temperature string

}

UART\_TX(&LF); // Line feed

// NRF Transmission

msg.length = i + 1;// Save the length of the temperature string

*memcpy*(msg.data, temp, msg.length);// Copy the string into the struct

nRF24L01\_transmit(rf, to\_address, &msg);// Transmit the temperature

}

}

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;

EICRA |= \_BV(ISC01);

EIMSK |= \_BV(INT0);

nRF24L01\_begin(rf);

return rf;

}

void setup\_timer(void)

{

TCCR1B |= \_BV(WGM12); // CTC Mode

TIMSK1 |= \_BV(OCIE1A); // COMP Interrupt

OCR1A = 15624; // 16 MHz/1024

TCCR1B |= \_BV(CS10) | \_BV(CS12); // Prescalar 1024

}

// UART transmission function

void UART\_TX(char \*data)

{

while(!(UCSR0A & (1 << UDRE0))); // Wait for UART to be available

UDR0 = \*data; // Send the data

}

ISR(TIMER1\_COMPA\_vect)

{

send\_message = true;

PORTC ^= (1 << 5);

}

ISR(INT0\_vect)

{

rf\_interrupt = true;

}

1. **C CODE OF RECEIVER**

#define *F\_CPU* 16000000UL // CPU Speed for delay

#define FOSC 16000000 // Clock speed

#define BAUD 115200 // Desire baud rate

#define MYUBRR FOSC/8/BAUD-1 // Formula to set the baud rate [Double Transmission]

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdlib.h>

#include <stdint.h>

#include <stdbool.h>

#include <string.h>

#include "nrf24l01-mnemonics.h"

#include "nrf24l01.h"

nRF24L01 \*setup\_rf(void);

void process\_message(char \*message, *uint8\_t* len);

void UART\_TX(char \*data);

volatile bool rf\_interrupt = false;

int main(void)

{

// NRF Settings

*uint8\_t* address[5] = {0x02, 0x04, 0x06, 0x08, 0x0A};

//prepare\_led\_pin();

sei();

nRF24L01 \*rf = setup\_rf();

nRF24L01\_listen(rf, 0, address);

*uint8\_t* addr[5];

nRF24L01\_read\_register(rf, CONFIG, addr, 1);

// USART Settings

UBRR0H = ((MYUBRR) >> 8); // Set baud rate for UPPER Register

UBRR0L = MYUBRR; // Set baud rate for LOWER Register

UCSR0A |= (1 << U2X0); // Double UART transmission speed

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

UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Frame: 8-bit Data and 1 Stop bit

while (1)

{

if(rf\_interrupt)

{

rf\_interrupt = false;

while(nRF24L01\_data\_received(rf))

{

nRF24L01Message msg;

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

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

}

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;

EICRA |= \_BV(ISC01);

EIMSK |= \_BV(INT0);

nRF24L01\_begin(rf);

return rf;

}

void process\_message(char \*message, *uint8\_t* len)

{

*uint8\_t* i = 0; // Iterative variable

char LF = '\n'; // Line Feed

// Process the temperature and display to UART

for(i = 0; i < len; i++)

{

UART\_TX(&message[i]);

}

UART\_TX(&LF);

}

ISR(INT0\_vect)

{

rf\_interrupt = true;

}

// UART transmission function

void UART\_TX(char \*data)

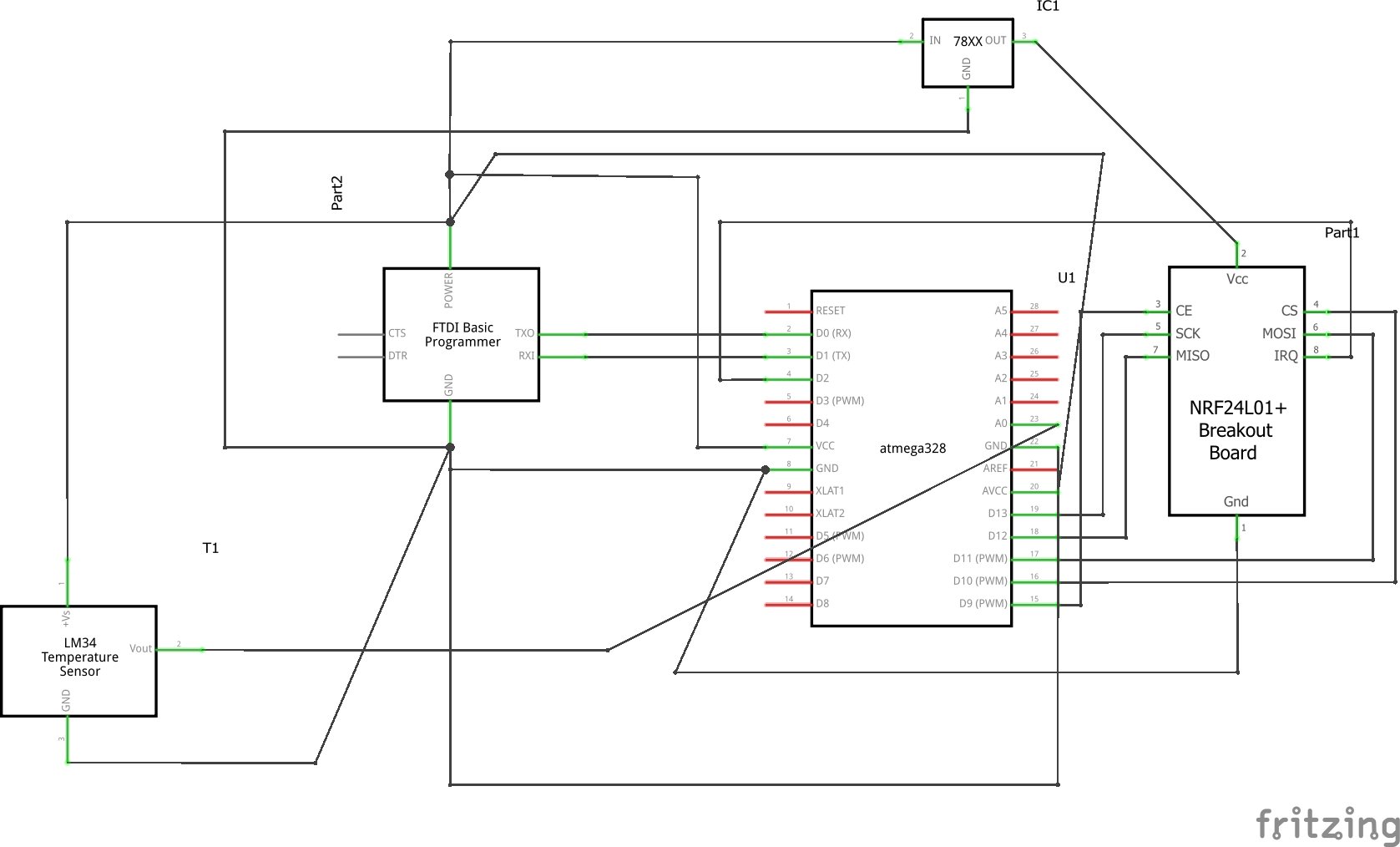
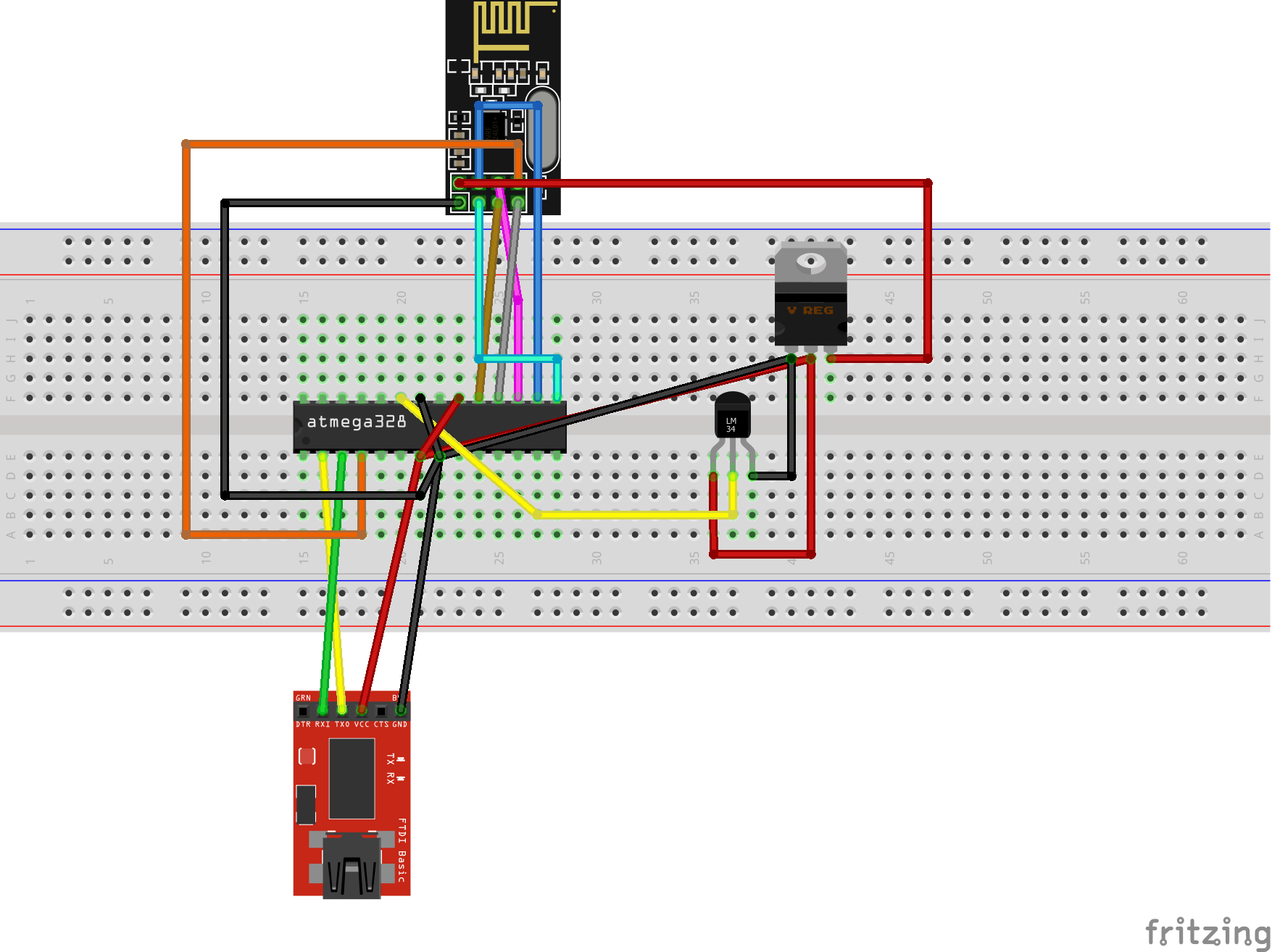
{

while(!(UCSR0A & (1 << UDRE0))); // Wait for UART to be available

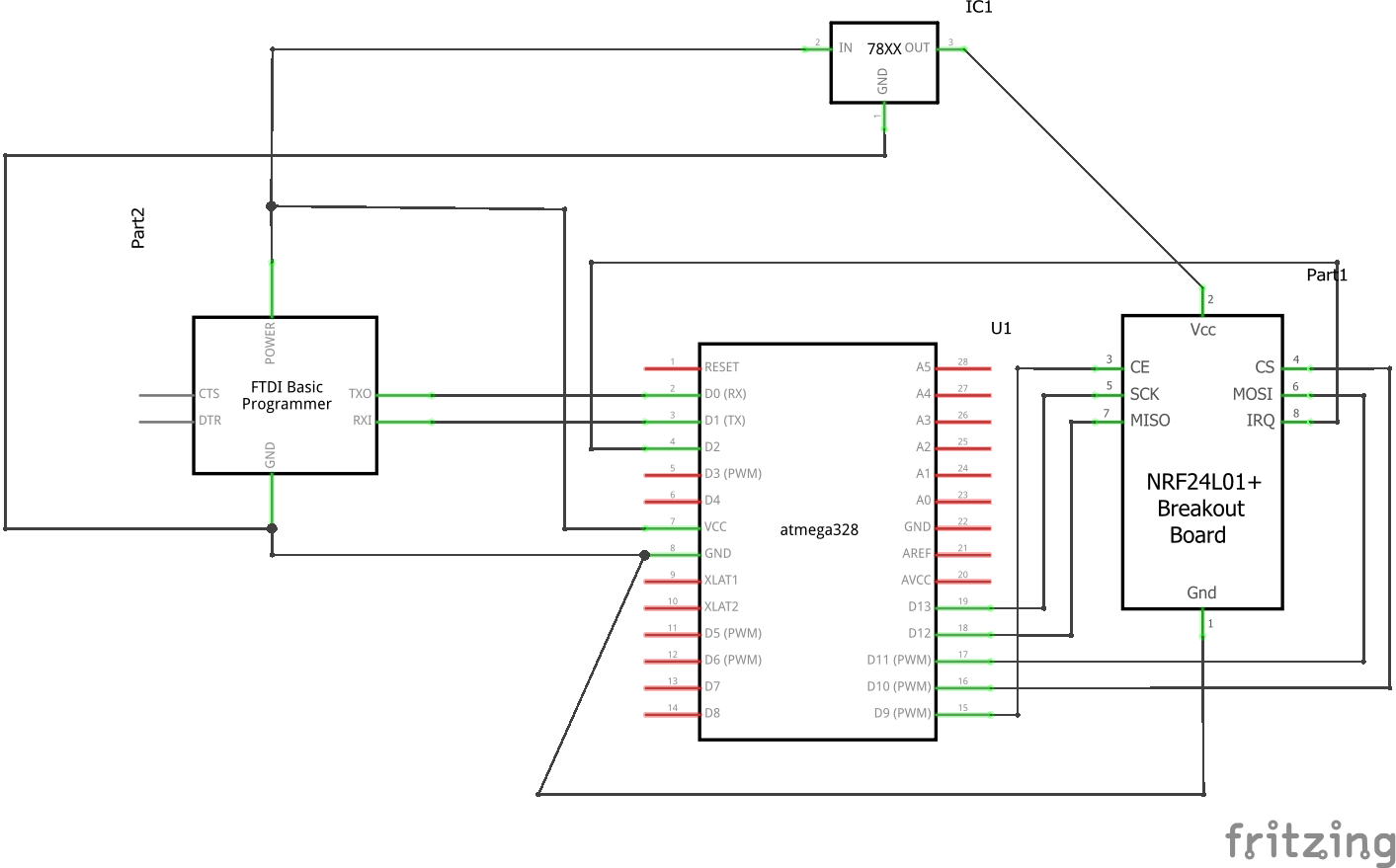
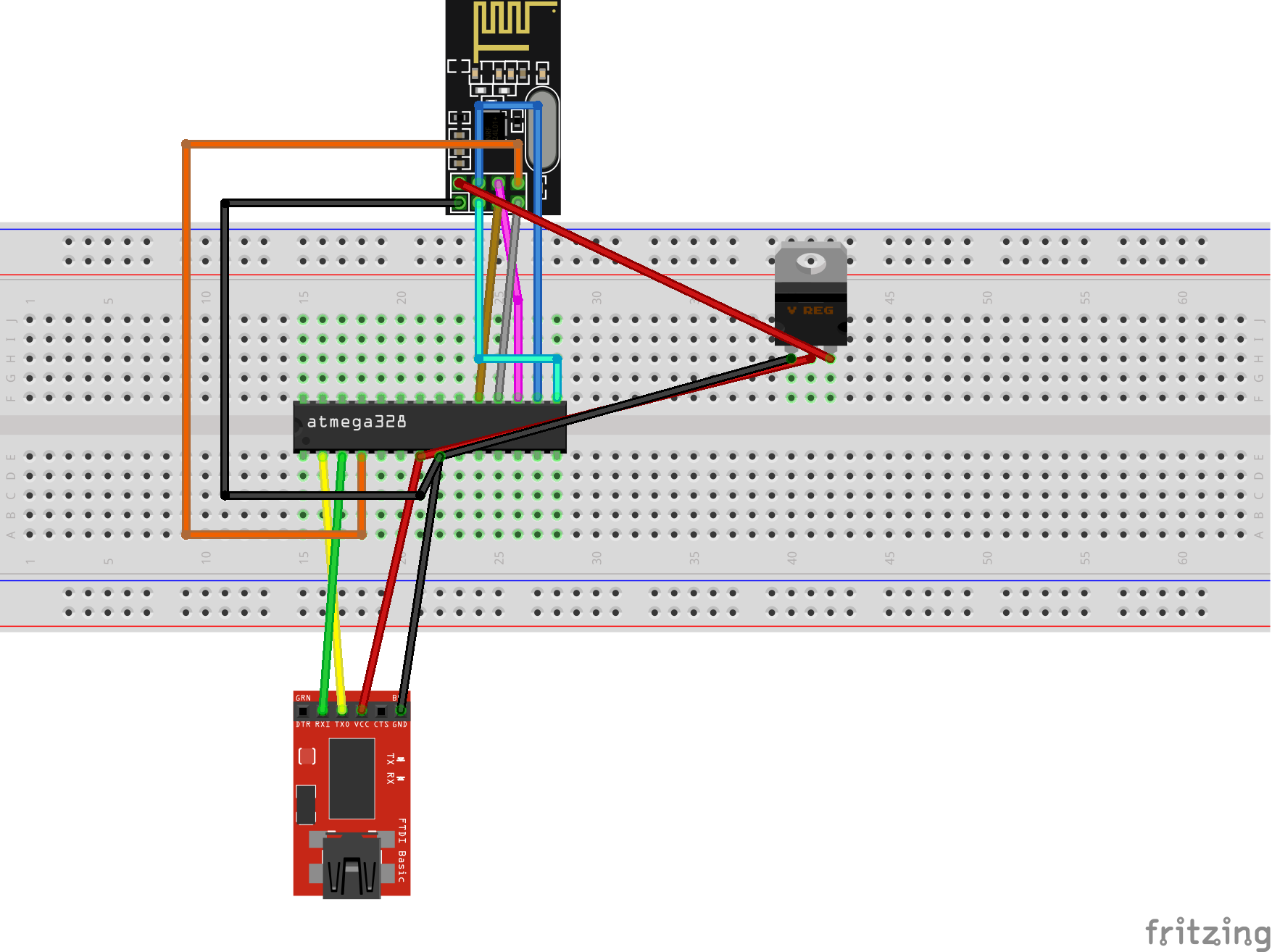
UDR0 = \*data; // Send the data

}

1. **BREADBOARD AND SCHEMATIC OF TRANSMITTER**



1. **BREADBOARD AND SCHEMATIC OF RECEIVER**



1. **PARTICIPANTS**

Tenniel, Argenis, and Jeffrey

**Student Academic Misconduct Policy**

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

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

BRYAN TAKEMOTO