NAME: BRIAN MEDRANO KIAER

JUSTINE JERYLL MACATO

TITLE: MULTIPLE LUX MEASUREMENT

GOAL:

- Implement TSL2591 Lux Sensors interfaced by I2C communication
- Use Transmitter NRF24L01 via serial communication to send sensor data to Receiver NRF24L01 and display on Terminal (Data Visualizer)
- Use TCA9548A 1-to-8 I2C Multiplexer with 2 TSL2591 Lux Sensors and send each Lux Sensor data through another transmitter NRF24L01 to the same Receiver NRF24L01.
- Display each Lux Sensor value via USART on the terminal (Atmel Studio Data Visualizer)

DELIVERABLES:

This project is intended to implement the TSL2591 along with RF communication to send multiple different light sensor measurements to provide the most accurate readings of the light in the room. These values can be incredibly important for several applications, for example an Ambient Light Sensor in cell phones or security sensors detecting a change in light for burglary alarms.

LITERATURE SURVEY:

Agriculture is a very important aspect of our lives. We may not think about it every day, but it is behind every grocery purchase that we get in a store. To grow food, we need light. There is a shortage in food production. The world's population is only growing larger. Emerging technologies are already being tested so that we don't need as much acreage to grow food. Scientists are growing food in labs. These scientists need to grow the food in a controlled manner. One variable they need to account for is how much light the plants need. In a natural system, the plants would only get light when the sun is out. Would having a constant light source allow the plants to grow faster? By using multiple different lux sensors we would be able to measure light accurately and can simulate our own production for natural resources.

Furthermore, there are many other applications that this project can necessarily help. Another example would be security. Safety is a tremendously valued factor in everyone's life. These lux sensors are able to also read infrared light accurately due to its high visual range. By taking advantage of such technology we could use this system to implement infrared security in many aspects. When a infrared light gets disrupted and the value of the light sensor changes that information will be sent to the central hub of the security system (Receiver) and can appropriately run a task for the received input such as turning on an alarm or calling authorities.

Using a TSL2591 Lux Sensor, we are able to read a dynamic range of light. By using the lux data that has been received we can implement and test different applications that would be appropriate such as the examples above. When using three lux sensors we are able to have the most accurate information and not display a skewed result.

The RF module allows us to communicate between different nodes. We used three different RF Modules, two for transmitting information and one for receiving information. By using USART we are able to see what is appropriately being sent and receive by displaying those calculated values on the Terminal. The lux sensor will read a value and the microcontroller will calculate an appropriate lux value depending on the gain and then the microcontroller will send that information to the RF module transmitter and the RF module receiver will receive this data and display is via USART.

The I2C interface is used to communicate with the three TSL2591 Lux Sensors. This interface allows these devices/sensors to be connected in a master-slave formatted connection. The master uses the SDA/SCL line to send and request data from the slave or another device connected to SDA/SCL line. However, the slave can only connect and communicate with the master. The communication between the two can only occur when the signal on SCL is high. By using different addresses we can implement multiple different slave-to-master communications. Most devices use 7 bits for the address while the 8th bit is used for the READ or WRITE (0 or 1) function of the device. The 9^{th} bit indicates an acknowledgment telling the slave that the master is reading if set, or not if 0. The only issue is when multiple devices have the same physical address we must approach this in a different manner.

The TCA9548A is used for I2C communication. It is a 1-to-8 multiplexer that is used to alternate different devices that use the same physical address. In this scenario, the TSL2591 have a fixed address at 0x29, because of this we need to use the TCA9548A for this project. The TCA9548A has a default address at 0x70, by changing the input pins A0,A1,A2 from 000-111 we can alternate between the addresses 0x70-0x77.

COMPONENTS:

TSL2591

The TSL2591 is a High Dynamic Range Digital Light Sensor. The lux calculations depending on the gain can range from 188uLux to 88,000 Lux. Furthermore, this sensor contains infrared and full spectrum diodes. Due to these diodes the sensor can differentiate between infrared, full-spectrum or human-visible light. The dynamic range is 600,000,000:1, also the current draw is extremely low so this also show the sensor is not resource heavy.

ATmega328p

The ATmega328P Xplained Mini is an 8-bit microcontroller designed by Atmel. This microcontroller implements either C code or AVR Assembly. There are 32 general purpose registers in the board. There's a huge variety of jobs that the Xplained Mini can do, for example we can implement ADC, PWM, Interrupts, and Timer implementation. The board also includes a

16-bit timer (Timer 1). The Xplained Mini Board also has a COM Port that can communicate serially so an FTDI chip is not needed for the RX/TX pins.

NRF24L01

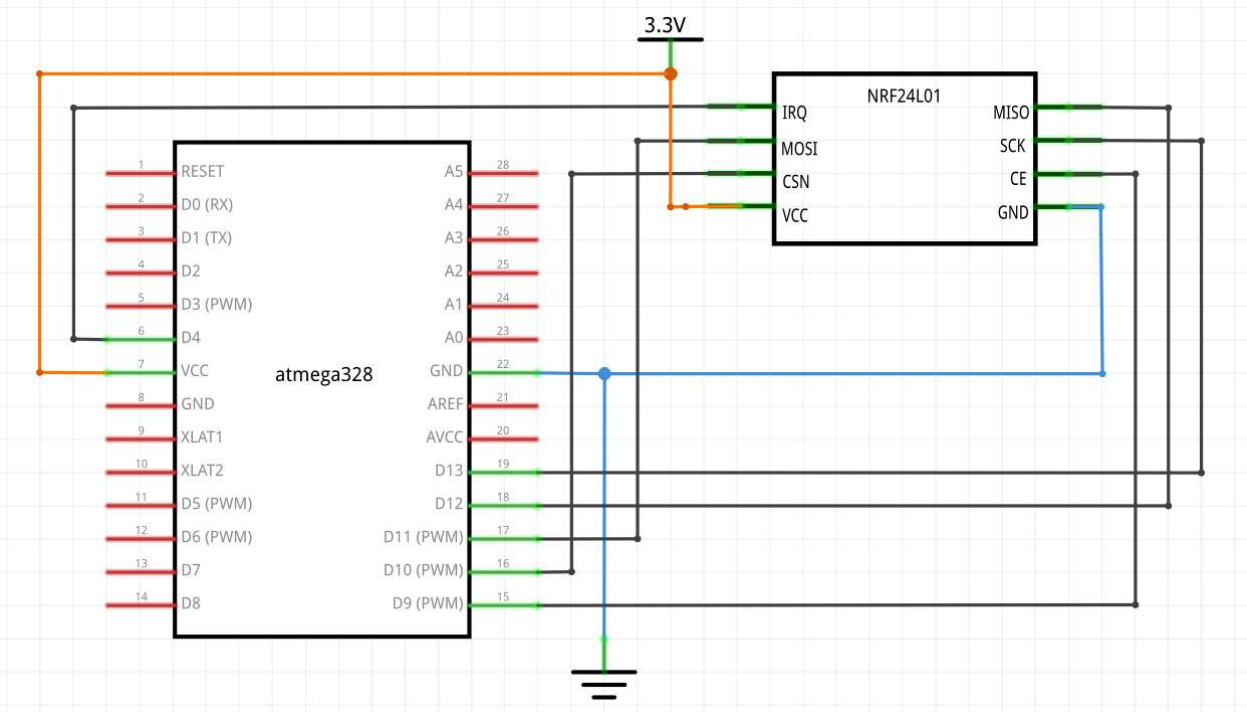
The nRF24L01+ is a ultra-low power RF Transceiver IC for the 2.4Ghz ISM band. The module has an advanced power management system which is beneficial for batters and power supplies for mobile applications. In the module, it consists of a 2.4GHz RF Transceiver, RF synthesizer, and baseband logic hardware protocol accelerator supporting a high-speed SPI interface.

TCA9548A

The TCA9548A module is 1-to-8 I2C Multiplexer. This multiplexer is responsible for implementing multiple I2C devices with the same address. By changing the channel input we are able to alternate between each channel/pipeline in the multiplexer.

SCHEMATICS:

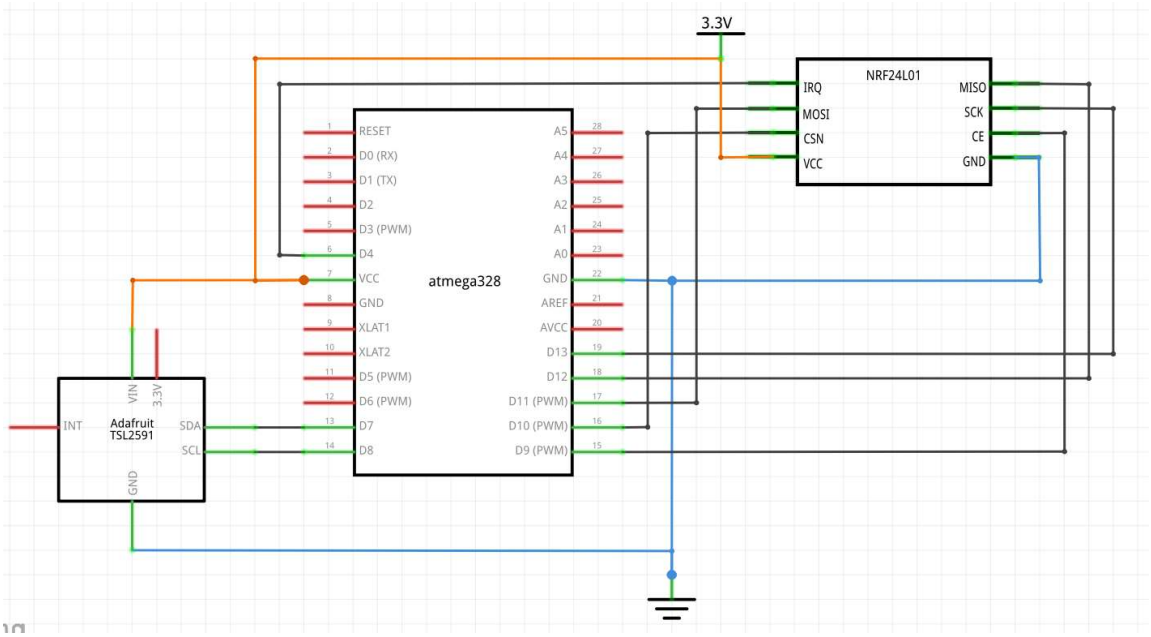
Receiver:



Schematic 1: Receiver

Here we have the schematic for the receiver. The Receiver uses 1 ATmega328P Xplained Mini Board with one nRF24L01/+ module powered with a 3.3 VCC. The board is powered via micro-USB to a USB 2.0 Port on the computer used.

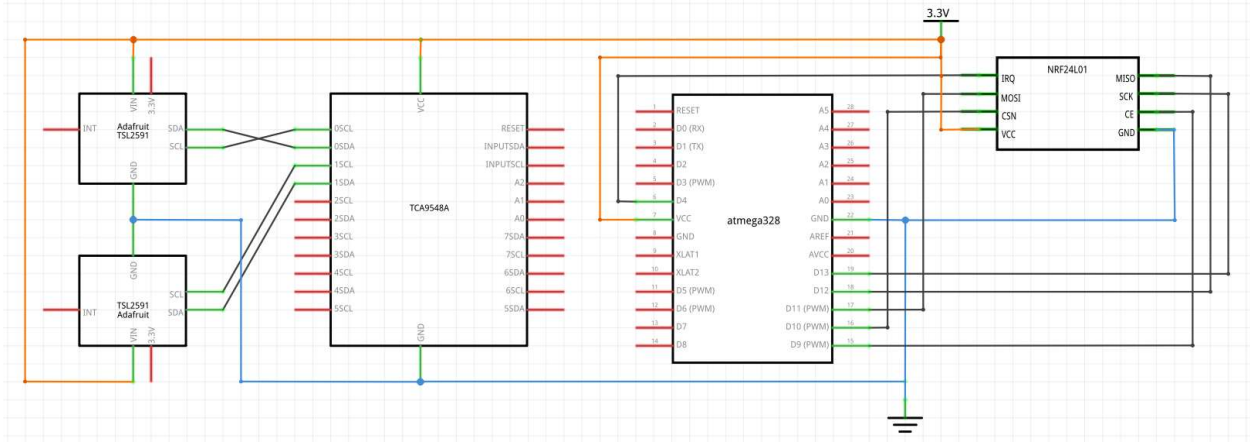
Transmitter 1 (NODE 1) :



Schematic 2: Node 1

Here we have the schematic for Node 1. Node 1 is primarily used for Tasks 1, 2, and 3. Connected is one TSL2591 Adafruit Lux Sensor, one ATmega328 Xplained Mini Board, and one nRF24L01/+ module powered with 3.3V. The board is powered via micro-USB to USB 2.0 on computer.

Transmitter 2 (NODE 2):



Schematic 3: Node 2

The last schematic is designed for Task 3 with two TSL2591 Lux Sensor devices, one TCA9548A Multiplexer, one nRF24L01/+, and one more ATmega328 Xplained Mini board.

INITIAL PCB:

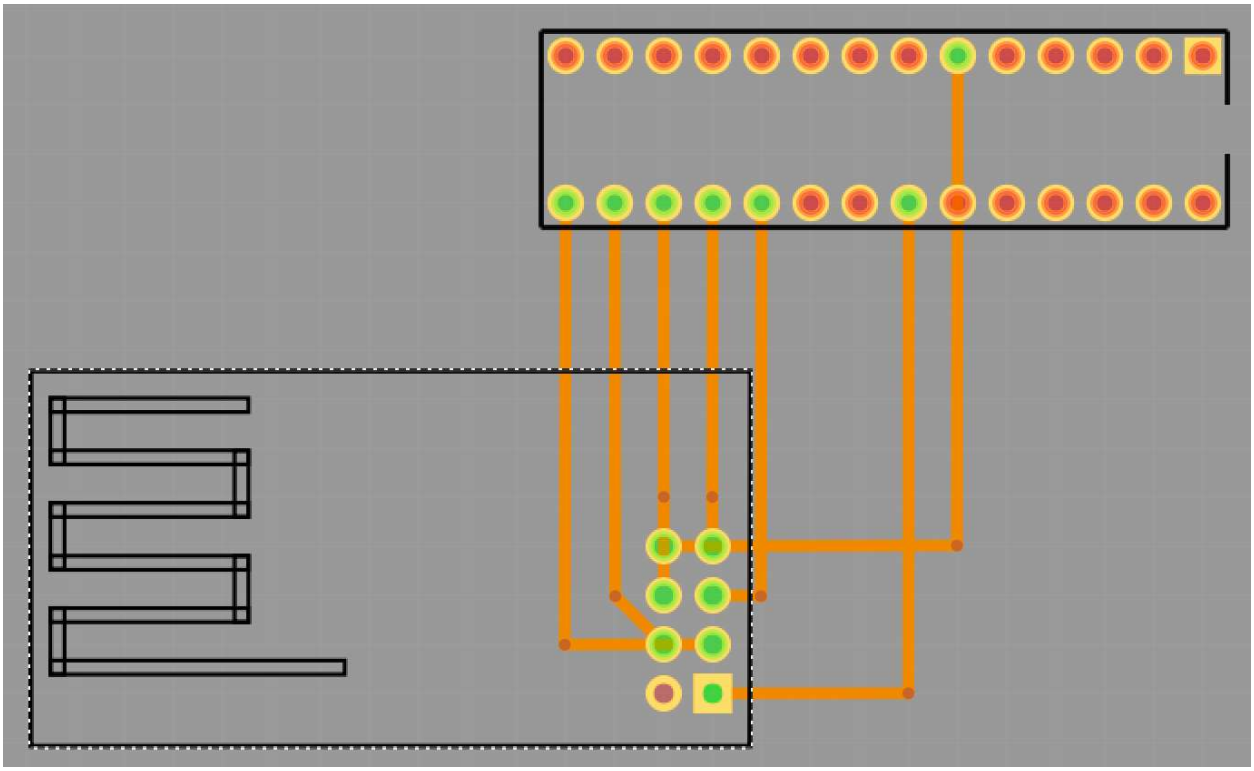


Figure 1. PCB of Receiver

Above we see that the ATmega328P board is connected to an nRF24L01/+ module. This PCB is designed to be the receiver for the whole system. This receiver is able to connect up to 6 other transmitters.

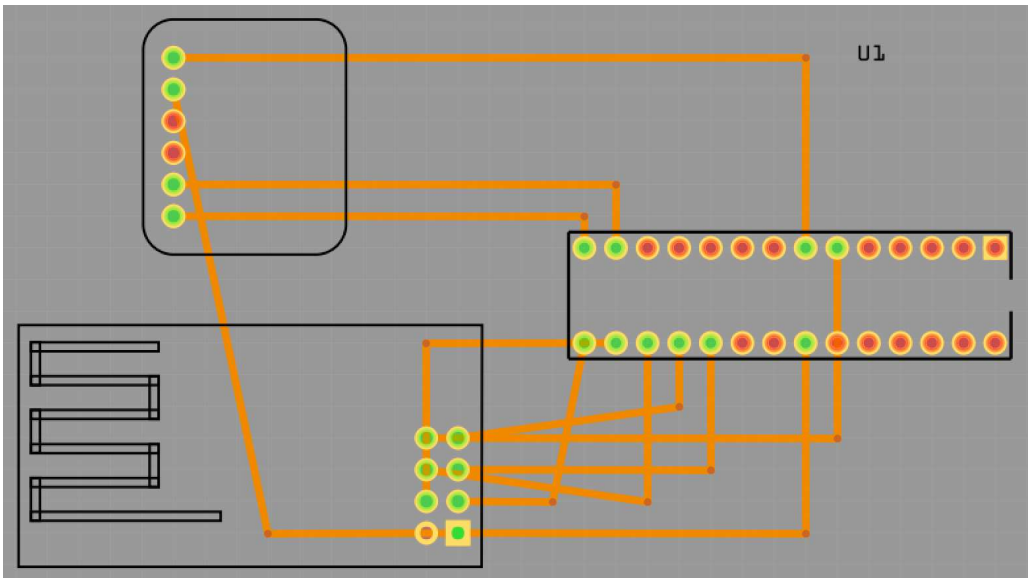


Figure 2. PCB of Transmitter 1

This PCB is designed for our first Transmitter that is responsible for Task 2. This task implements a nRF24L01/+ module and a TSL2591 Dynamic Range Lux Sensor. This reads lux values implemented by I2C communication and transmits that data through radio frequency and by specifying the correct address we can communicate with the receiver node previously mentioned.

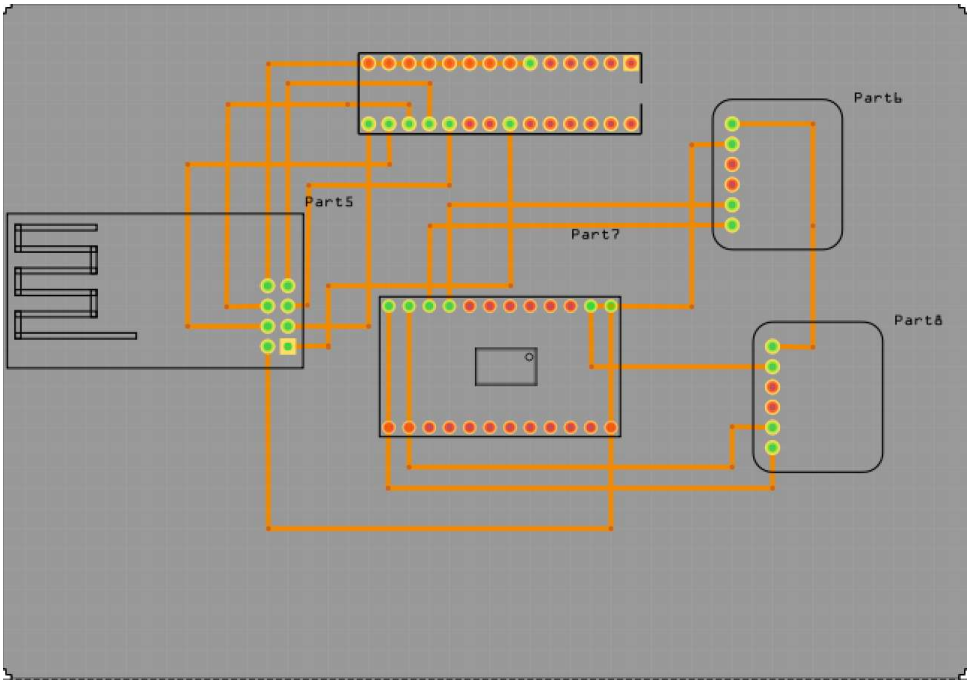


Figure 3. PCB of Transmitter 2

The PCB above is designed for Task 3 which is responsible for reading two LUX sensors, because the two lux sensors provided have the same physical address at 0x29, because of this we used a 1-to-8 Multiplexer to alternate between channels.

IMPLEMENTATION:

- TSL2591 Dynamic Range Lux Sensor interfaced via I2C to read values provided by the onboard infrared and full spectrum sensors and return values to microcontroller.
    - The I2C communication is implemented via a header file adapted from g4lvanix via Github.
    - USART and the onboard micro-USB COM Port build onto the ATmega328P Xplained Mini board for the Serial Communication.
- TCA9548A Multiplexer is used to interface multiple different I2C devices with the same physical address.
- nRF24L01/+ module is used to receive from and send data to the microcontroller
    - The RF module is interfaced through serial communications with the use of USART

    - Micro-USB COM Port was used to assure the correct baud rate, I/O, and information received was used
    - Data Visualizer built into Atmel Studio 7 was used for the assignment.
- ATmega328p Xplained Mini was used to link the TSL2591 and RF module and 1-to-8 MUX
    - The ATmega328p performed calculations to determine the angle of motion for the IMU
    - Sent trial information to the Wi-Fi module to ensure serial communications worked correctly

SNAPSHOTS/SCREENSHOTS*: (only links - do not embed images or videos in the document)

Snapshots: https://drive.google.com/open?id=1iIgksmAnz4dPIYtYAHqERxz6MZbWTwEd

TSL2591 Implementation Videos:

https://youtu.be/rgk-amaNQww

https://youtu.be/8t0dfonB40g

https://youtu.be/tPPbm4_hpYk

https://youtu.be/eug3BW66-UY

CODE:

RECEIVER

```
/*
 * FINAL_RECEIVER.c
 *
 * Created: 4/30/2018 1:22:08 PM
 * Author : brian
 */


#define F_CPU 16000000UL //16MHz
#define BAUD 9600 //Baud Rate
#define MYUBRR F_CPU/16/BAUD-1 //calculate Baud
#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 USART_init( unsigned int ubrr ); //initialize USART
void USART_tx_string(char *data); //Print String USART
volatile unsigned int adc_temp;
char outs[20]; //array


int main(void) {
	uint8_t address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
	prepare_led_pin();
	char str[80];
		USART_init(MYUBRR); // Initialize the USART (RS232 interface)
		USART_tx_string("Connected!\r\n"); // shows theres a connection with USART
		_delay_ms(125); // wait a bit


	sei();
	nRF24L01 *rf = setup_rf();
	nRF24L01_listen(rf, 0, address);
	uint8_t addr[5];
	nRF24L01_read_register(rf, CONFIG, addr, 1);
		int i = 0;
	while (true) {
		if (rf_interrupt) {
			rf_interrupt = false;

			while (nRF24L01_data_received(rf)) {
				nRF24L01Message msg;
				nRF24L01Message msg2;
				if(i == 0){
				nRF24L01_read_received_data(rf, &msg);
				process_message((char *)msg.data);
				}
				 if(i == 1){
				nRF24L01_read_received_data(rf, &msg2);
				process_message((char *)msg2.data);
				}
				 if(i==2)
				{
					USART_tx_string(msg.data); //print first node
					USART_tx_string(" ");
					USART_tx_string(msg2.data); //print second node
					USART_tx_string(" \r\n");
				}
				i++;
				if(i == 3) //reset counter
				i = 0;
			}
			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);
}



/* INIT USART (RS-232) */
void USART_init( unsigned int ubrr ) {
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
	UCSR0B = (1 << TXEN0); // Enable receiver, transmitter & RX interrupt
	UCSR0C = (3 << UCSZ00); //asynchronous 8 N 1
}

void USART_tx_string( char *data ) {
	while ((*data != '\0')) {
		while (!(UCSR0A & (1 <<UDRE0)));
		UDR0 = *data;
		data++;
	}
}
// nRF24L01 interrupt
ISR(INT0_vect) {
	rf_interrupt = true;
}
```

NODE 1:

```
/*
 * NODE1.c
 *
 * Created: 5/3/2018 10:45:09 AM
 * Author : kiaer
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include "nrf24l01.h"
#include "i2c_master.h"
#define BAUD 9600 //Baud Rate
#define MYUBRR F_CPU/16/BAUD-1 //calculate Baud

#define TSL2591_VISIBLE     (2)
#define TSL2591_INFRARED (1)
#define TSL2591_FULLSPECTRUM (0)

#define TSL2591_ADDR (0x29)
#define TSL2591_READBIT (0x01)

#define TSL2591_COMMAND_BIT (0xA0)
#define TSL2591_CLEAR_INT (0xE7)
#define TSL2591_TEST_INT (0xE4)
#define TSL2591_WORD_BIT (0x20)
#define TSL2591_BLOCK_BIT (0x10)

#define TSL2591_ENABLE_POWEROFF (0x00)
#define TSL2591_ENABLE_POWERON (0x09)
#define TSL2591_ENABLE_AEN (0x02)
#define TSL2591_ENABLE_AIEN (0x10)
#define TSL2591_ENABLE_NPIEN (0x80)

#define TSL2591_LUX_DF (408.0f)
#define TSL2591_LUX_COEFB (1.64f)
#define TSL2591_LUX_COEFC (0.59f)
#define TSL2591_LUX_COEFD (0.86f)

#define TSL2591_ENABLE (0x00)
#define TSL2591_CONFIG (0x01)
#define TSL2591_ID (0x12)
#define TSL2591_C0DATAL (0x14)
#define TSL2591_C0DATAH (0x15)
#define TSL2591_C1DATAL (0x16)
#define TSL2591_C1DATAH (0x17)

#define TSL2591_WRITE 0x52 //Adress for LUX Sensor at 0x29 with last bit for read or
write = 0x52
#define TSL2591_READ 0x53

void init_uart(uint16_t baudrate){
```

```
	uint16_t UBRR_val = (F_CPU/16)/(baudrate-1);

	UBRR0H = UBRR_val >> 8;
	UBRR0L = UBRR_val;

	UCSR0B |= (1<<TXEN0) | (1<<RXEN0) | (1<<RXCIE0);
	UCSR0C |= (1<<USBS0) | (3<<UCSZ00);
}

void uart_putc(unsigned char c){
	while(!(UCSR0A & (1<<UDRE0)));
	UDR0 = c;
}


void uart_puts(char *s){
	while(*s){
		uart_putc(*s);
		s++;
	}
}

void init_TSL2591(void){
	i2c_start(TSL2591_WRITE);
	i2c_write(TSL2591_COMMAND_BIT | TSL2591_ENABLE);
	i2c_write(TSL2591_ENABLE_POWERON | TSL2591_ENABLE_AEN |TSL2591_ENABLE_AIEN |
TSL2591_ENABLE_NPIEN);
	i2c_stop();

	i2c_start(TSL2591_WRITE);
	i2c_write(TSL2591_COMMAND_BIT | TSL2591_CONFIG);
	i2c_write(0x10);
	i2c_stop();
}

float getLux(void){
	float atime = 100.0f, again = 25.0f;
	uint16_t ch0, ch1;
	float cp1;
	float  lux1, lux2, lux;
	lux1 = lux2 = lux = 0;
	uint32_t x = 1;

	i2c_start(TSL2591_WRITE);
	i2c_write(TSL2591_COMMAND_BIT | TSL2591_C0DATAL);
	i2c_stop();

	i2c_start(TSL2591_READ);

	x = ((uint8_t)i2c_read_ack())<<8;
	x |= i2c_read_ack();
	x <<=16;
	x |= ((uint8_t)i2c_read_ack())<<8;
	x |= i2c_read_ack();
	i2c_stop();
	ch1 = x>>8;
	ch0 = x & 0xFFFF;
```

```
	cp1 = (uint32_t) (atime * again) / TSL2591_LUX_DF;
	lux1 = (uint32_t) ((float) ch0 - (TSL2591_LUX_COEFB * (float) ch1)) / cp1;
	lux2 = (uint32_t) ((TSL2591_LUX_COEFC * (float) ch0) - (TSL2591_LUX_COEFD *
(float) ch1)) / cp1;

	lux = (lux1 > lux2) ? lux1: lux2;

	return lux;
}

unsigned char i2c_read(unsigned char isLast)
{
	if(isLast == 0)
	TWCR = ( 1 << TWINT) | (1 << TWEN) | (1 <<TWEA);
	else
	TWCR = ( 1 << TWINT) | ( 1 << TWEN);
	while ((TWCR & ( 1 << TWINT)) == 0);
	return TWDR;

}
void setup_timer(void);
nRF24L01 *setup_rf(void);
volatile bool rf_interrupt = false;
volatile bool send_message = false;

void USART_init( unsigned int ubrr ); //initialize USART
void USART_tx_string(char *data); //Print String USART
volatile unsigned int adc_temp;
char outs[256]; //array

volatile char received_data;


int main(void) {
	uint8_t to_address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
	bool on = false;
	char buffer[6];
	float alux;
	char str[80];
	DDRD = (1<<PD5) |(1<<PD6)|(1<<PD7);
	PORTD = 0b11100000;

	USART_init(MYUBRR); // Initialize the USART (RS232 interface)

	i2c_init();
	init_TSL2591();

	USART_tx_string("Connected!\r\n"); // shows theres a connection with USART
	_delay_ms(125); // wait a bit

	sei();
	nRF24L01 *rf = setup_rf();
	setup_timer();

	while (true) {
		alux = getLux();
		dtostrf(alux, 6, 2, buffer);
```

```
                //     _delay_ms(10000);
            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){
                    strcpy(msg.data, "NODE1: ");
                //     memcpy(msg.data, buffer, 10);
                    strcat(msg.data, buffer);
                    _delay_ms(1000);
                    uart_puts(msg.data);
                    uart_puts("  \r\n");
                }
                else memcpy(msg.data, "OFF", 4);
                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);
}

/* INIT USART (RS-232) */
void USART_init( unsigned int ubrr ) {
        UBRR0H = (unsigned char)(ubrr>>8);
        UBRR0L = (unsigned char)ubrr;
```

```
	UCSR0B |= (1 << TXEN0) | (1 << RXEN0)| ( 1 << RXCIE0); // Enable receiver,
transmitter & RX interrupt
	UCSR0C |= (1<<UCSZ01) | (1 << UCSZ00);
}

void USART_tx_string( char *data ) {
	while ((*data != '\0')) {
		while (!(UCSR0A & (1 <<UDRE0)));
		UDR0 = *data;
		data++;
	}
}

// each one second interrupt
ISR(TIMER1_COMPA_vect) {
	send_message = true;
}
// nRF24L01 interrupt
ISR(INT0_vect) {
	rf_interrupt = true;
}
```

NODE 2:

```
/*
 * NODE2.c
 *
 * Created: 5/2/2018 12:16:57 PM
 * Author : kiaer
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include "nrf24l01.h"
#include "i2c_master.h"
#define BAUD 9600 //Baud Rate
#define MYUBRR F_CPU/16/BAUD-1 //calculate Baud

#define TSL2591_VISIBLE     (2)
#define TSL2591_INFRARED (1)
#define TSL2591_FULLSPECTRUM (0)

#define TSL2591_ADDR (0x29)
#define TSL2591_READBIT (0x01)

#define TSL2591_COMMAND_BIT (0xA0)
#define TSL2591_CLEAR_INT (0xE7)
#define TSL2591_TEST_INT (0xE4)
#define TSL2591_WORD_BIT (0x20)
#define TSL2591_BLOCK_BIT (0x10)

#define TSL2591_ENABLE_POWEROFF (0x00)
```

```
#define TSL2591_ENABLE_POWERON (0x09)
#define TSL2591_ENABLE_AEN (0x02)
#define TSL2591_ENABLE_AIEN (0x10)
#define TSL2591_ENABLE_NPIEN (0x80)

#define TSL2591_LUX_DF (408.0f)
#define TSL2591_LUX_COEFB (1.64f)
#define TSL2591_LUX_COEFC (0.59f)
#define TSL2591_LUX_COEFD (0.86f)

#define TSL2591_ENABLE (0x00)
#define TSL2591_CONFIG (0x01)
#define TSL2591_ID (0x12)
#define TSL2591_C0DATAL (0x14)
#define TSL2591_C0DATAH (0x15)
#define TSL2591_C1DATAL (0x16)
#define TSL2591_C1DATAH (0x17)

#define TSL2591_WRITE 0xE0  //address for I2C MUX (0x70) with last bit as a read or write
#define TSL2591_READ 0xE1

void init_uart(uint16_t baudrate){
	uint16_t UBRR_val = (F_CPU/16)/(baudrate-1);

	UBRR0H = UBRR_val >> 8;
	UBRR0L = UBRR_val;

	UCSR0B |= (1<<TXEN0) | (1<<RXEN0) | (1<<RXCIE0);
	UCSR0C |= (1<<USBS0) | (3<<UCSZ00);
}

void uart_putc(unsigned char c){
	while(!(UCSR0A & (1<<UDRE0)));
	UDR0 = c;
}


void uart_puts(char *s){
	while(*s){
		uart_putc(*s);
		s++;
	}
}

void init_TSL2591(void){
	i2c_start(TSL2591_WRITE);
	i2c_write(TSL2591_COMMAND_BIT | TSL2591_ENABLE);
	i2c_write(TSL2591_ENABLE_POWERON | TSL2591_ENABLE_AEN |TSL2591_ENABLE_AIEN |
TSL2591_ENABLE_NPIEN);
	i2c_stop();

	i2c_start(TSL2591_WRITE);
	i2c_write(TSL2591_COMMAND_BIT | TSL2591_CONFIG);
	i2c_write(0x10);
	i2c_stop();
}

float getLux(void){
```

```
	float atime = 100.0f, again = 25.0f;
	uint16_t ch0, ch1;
	float cp1;
	float  lux1, lux2, lux;
	lux1 = lux2 = lux = 0;
	uint32_t x = 1;

	i2c_start(TSL2591_WRITE);
	i2c_write(TSL2591_COMMAND_BIT | TSL2591_C0DATAL);
	i2c_stop();

	i2c_start(TSL2591_READ);

	x = ((uint8_t)i2c_read_ack())<<8;
	x |= i2c_read_ack();
	x <<=16;
	x |= ((uint8_t)i2c_read_ack())<<8;
	x |= i2c_read_ack();
	i2c_stop();
	ch1 = x>>8;
	ch0 = x & 0xFFFF;

	cp1 = (uint32_t) (atime * again) / TSL2591_LUX_DF;
	lux1 = (uint32_t) ((float) ch0 - (TSL2591_LUX_COEFB * (float) ch1)) / cp1;
	lux2 = (uint32_t) ((TSL2591_LUX_COEFC * (float) ch0) - (TSL2591_LUX_COEFD *
(float) ch1)) / cp1;

	lux = (lux1 > lux2) ? lux1: lux2;

	return lux;
}

unsigned char i2c_read(unsigned char isLast)
{
	if(isLast == 0)
	TWCR = ( 1 << TWINT) | (1 << TWEN) | (1 <<TWEA);
	else
	TWCR = ( 1 << TWINT) | ( 1 << TWEN);
	while ((TWCR & ( 1 << TWINT)) == 0);
	return TWDR;

}
void setup_timer(void);
nRF24L01 *setup_rf(void);
volatile bool rf_interrupt = false;
volatile bool send_message = false;

void USART_init( unsigned int ubrr ); //initialize USART
void USART_tx_string(char *data); //Print String USART
volatile unsigned int adc_temp;
char outs[256]; //array

volatile char received_data;


int main(void) {
	uint8_t to_address[5] = { 0x20, 0x30, 0x40, 0x51, 0x61 };
	bool on = false;
```

```
	char buffer[6];
	float alux;
	char str[80];
	DDRD = (1<<PD5) |(1<<PD6)|(1<<PD7);
	PORTD = 0b11100000;

	USART_init(MYUBRR); // Initialize the USART (RS232 interface)

	i2c_init();
	init_TSL2591();

	USART_tx_string("Connected!\r\n"); // shows theres a connection with USART
	_delay_ms(125); // wait a bit

	sei();
	nRF24L01 *rf = setup_rf();
	setup_timer();

	while (true) {
		alux = getLux();
		dtostrf(alux, 6, 2, buffer);

			//	_delay_ms(10000);
		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){
				strcpy(msg.data, "NODE2: ");
			//	memcpy(msg.data, buffer, 10);
				strcat(msg.data, buffer);
				_delay_ms(1000);
				uart_puts(msg.data);
				uart_puts("  \r\n");
			}
			else memcpy(msg.data, "OFF", 4);
			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);
}

/* INIT USART (RS-232) */
void USART_init( unsigned int ubrr ) {
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
	UCSR0B |= (1 << TXEN0) | (1 << RXEN0)| ( 1 << RXCIE0); // Enable receiver,
transmitter & RX interrupt
	UCSR0C |= (1<<UCSZ01) | (1 << UCSZ00);
}

void USART_tx_string( char *data ) {
	while ((*data != '\0')) {
		while (!(UCSR0A & (1 <<UDRE0)));
		UDR0 = *data;
		data++;
	}
}

// each one second interrupt
ISR(TIMER1_COMPA_vect) {
	send_message = true;
}
// nRF24L01 interrupt
ISR(INT0_vect) {
	rf_interrupt = true;
}
```

REFERENCE:

ATmega328p Datasheet: http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf

Adafruit TCA9548 1-to-8 I2C Multiplexer: https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout/overview

Adafruit TSL2591: https://www.adafruit.com/product/1980

I2C Library: https://github.com/g4lvanix/I2C-master-lib

NRF24L01/+: https://github.com/antoineleclair/avr-nrf24l01

USART Libraries: http://jump.to/fleury