Local Connectivity

nRF24L01+

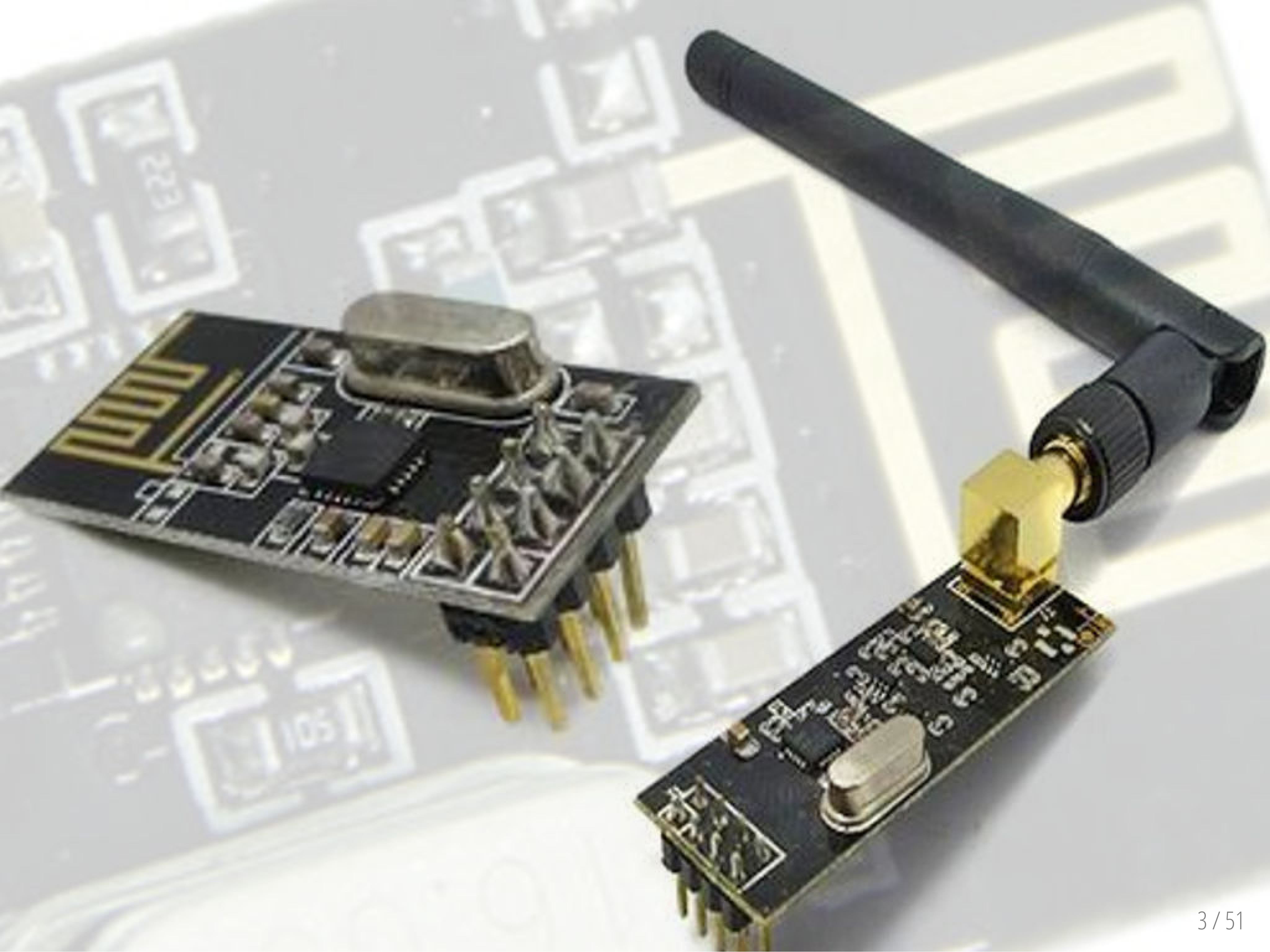
Eueung Mulyana

https://eueung.github.io/012017/nrf24

CodeLabs | Attribution-ShareAlike CC BY-SA

Outline

- Introduction
- Getting Started - Preparation
- Getting Started - Code & Play
- Simple Remote Control
- Gateway



Introduction

nRF24L01+

nRF24L01+ is a highly integrated, ultra low power (ULP) 2Mbps **RF transceiver** IC for the 2.4GHz ISM (Industrial, Scientific and Medical) band 2.400 - 2.4835GHz.

The Nordic **nRF24L01+** integrates a complete 2.4GHz RF **transceiver**, RF **synthesizer**, and **baseband logic** including the hardware protocol accelerator (Enhanced ShockBurst) supporting a high-speed **SPI** interface for the application controller.

With peak RX/TX currents lower than 14mA, a sub uA power down mode, advanced power management, and a 1.9 to 3.6V supply range, the nRF24L01+ provides a true ULP solution enabling months to years of battery life from coin cell or AA/AAA batteries.

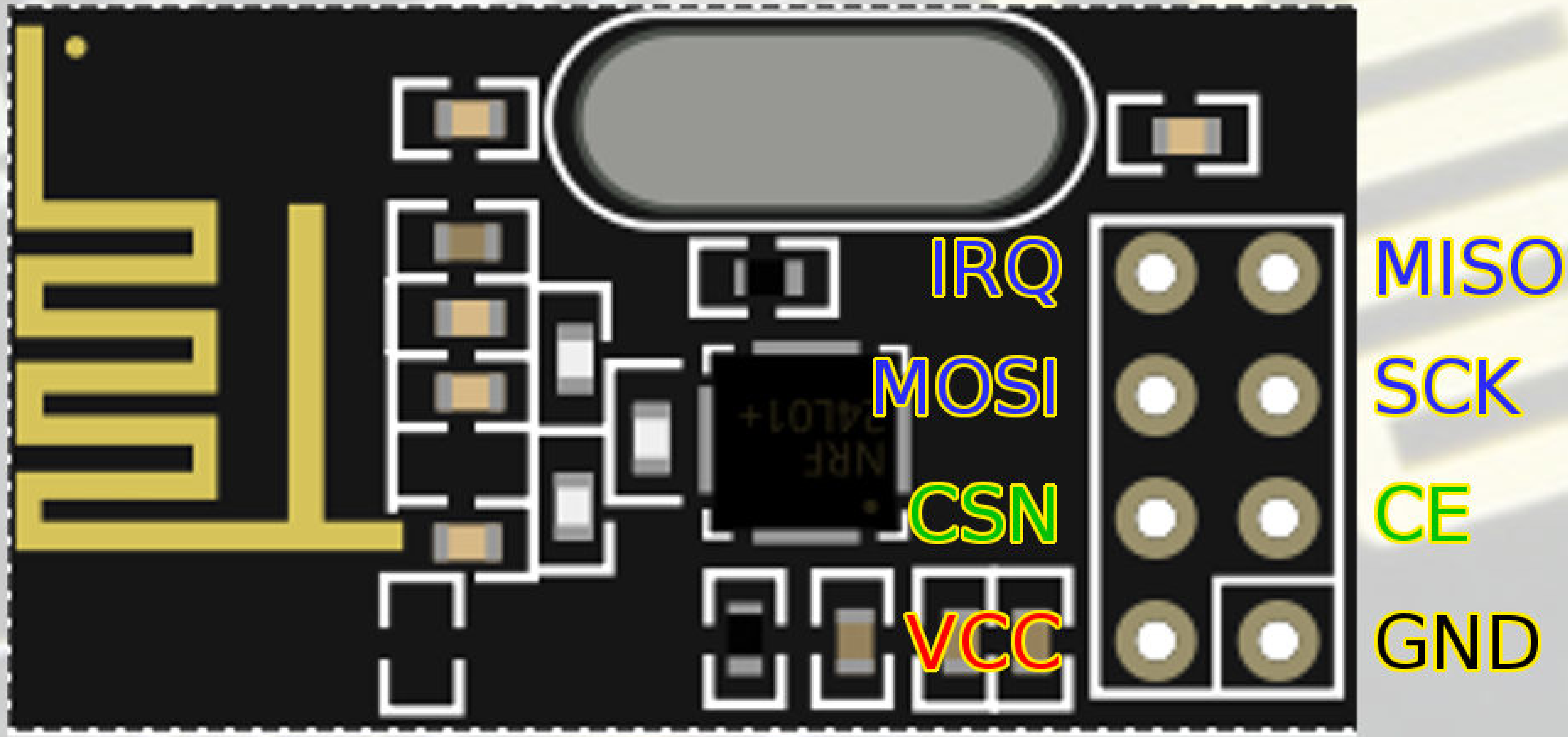
Ref: Nordic Semiconductor

nRF24L01+

1. ISM Frequency Band at 2.400 - 2.4835 GHz (Spacing at 1 or 2 MHz, GFSK)
2. 126 RF Channels
3. Air Data Rate Configurable to 2 Mbps (Options: 250 kbps, 1 Mbps)
4. 4-Pin Hardware **SPI**
5. 5V Tolerant Inputs
6. 6 Data Pipe MultiCeiver for 1:6 star networks

Notes: Power still at 3.3V!

Pin Map



Important Notes

Radio is sensitive to **Noises**! Make sure that the circuit (wire, solder, etc.) is stable.

Anything **fluxtuates** is bad!

Preparation

Getting Started

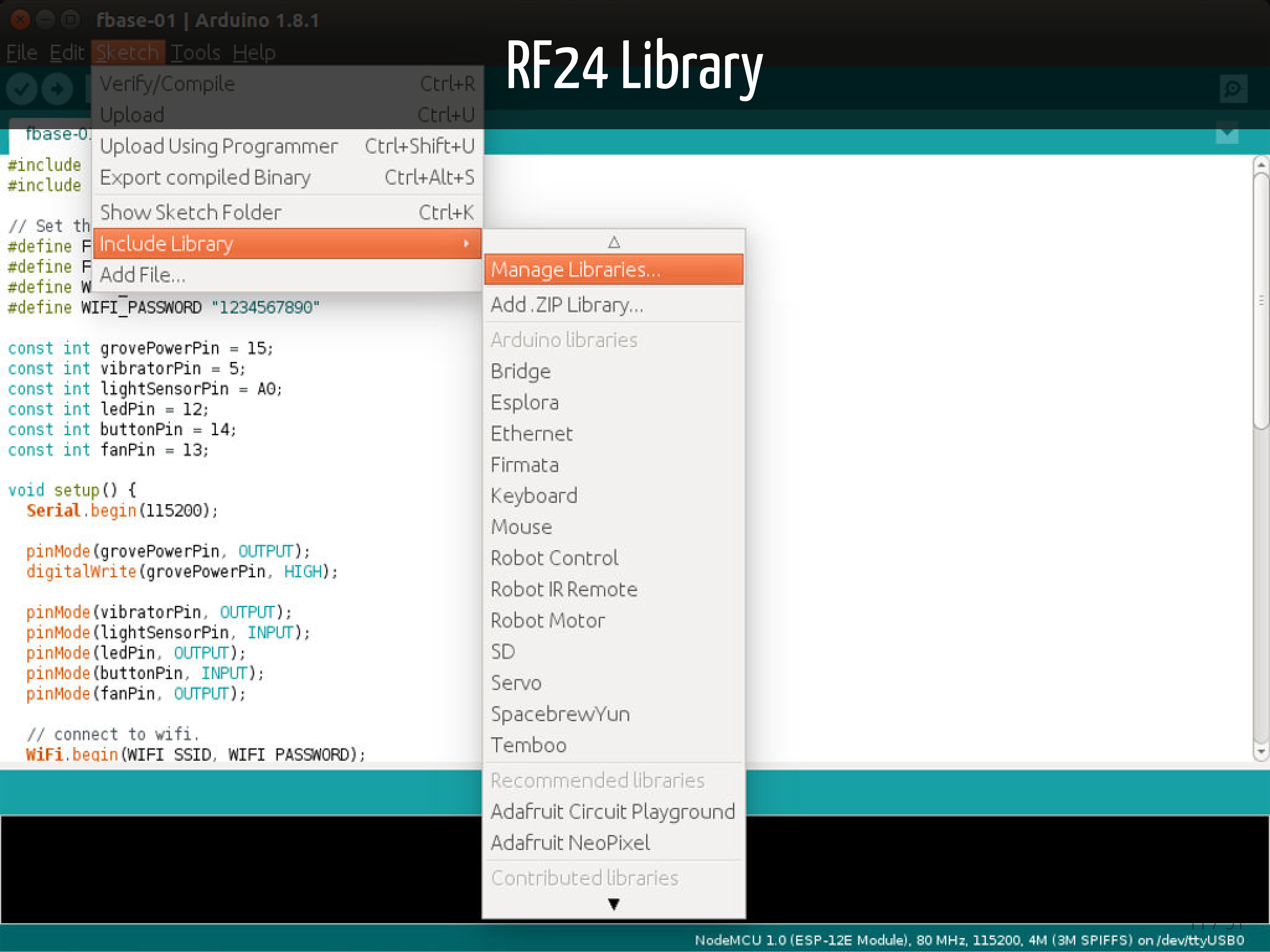
Getting Started

Arduino IDE, NodeMCU & Nano

1. Install RF24 Library
2. Prepare the First Node - **NodeMCU**
3. Prepare the Second Node - Arduino **Nano**

This setup is for demo purpose only. Can be any MCUs.

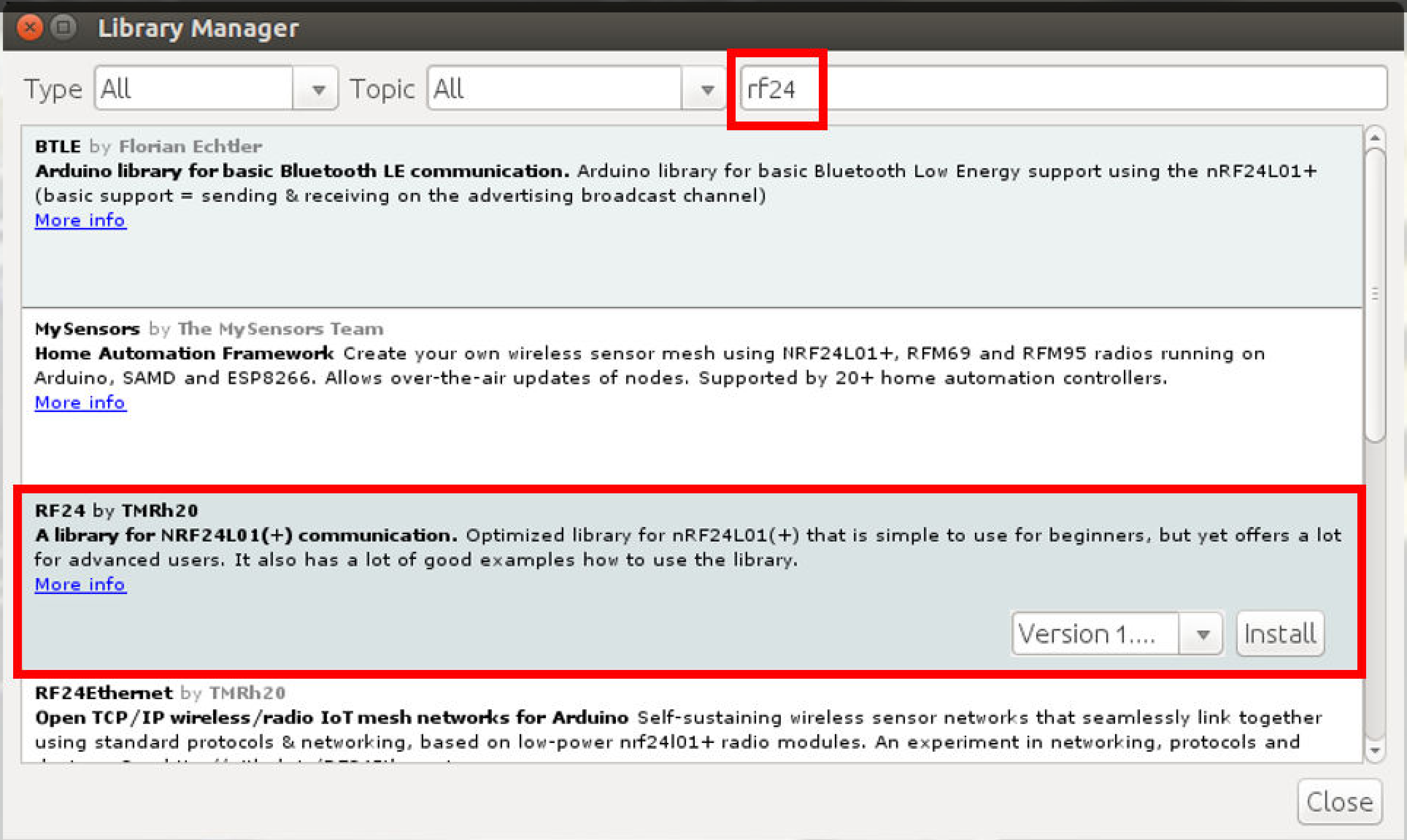
RF24 Library



```
#include
#include

// Set th
#define F
#define F
#define W
#define WIFI_PASSWORD "1234567890"

const int grovePowerPin = 15;
const int vibratorPin = 5;
const int lightSensorPin = A0;
const int ledPin = 12;
const int buttonPin = 14;
const int fanPin = 13;

void setup() {
  Serial.begin(115200);

  pinMode(grovePowerPin, OUTPUT);
  digitalWrite(grovePowerPin, HIGH);

  pinMode(vibratorPin, OUTPUT);
  pinMode(lightSensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  pinMode(fanPin, OUTPUT);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, 115200, 4M (3M SPIFFS) on /dev/ttyUSB0

RF24 Library



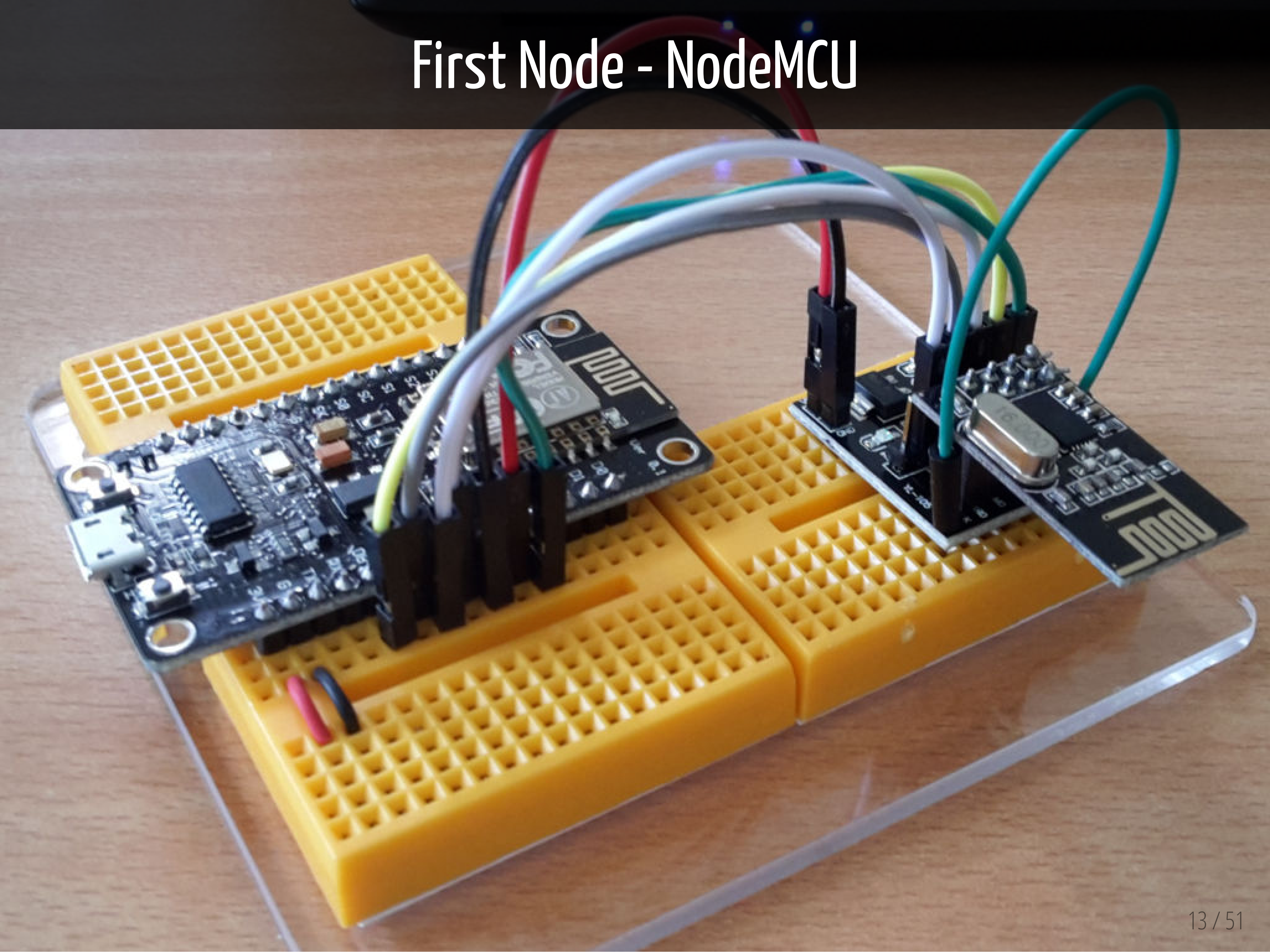
BTLE by Florian Echtler
Arduino library for basic Bluetooth LE communication. Arduino library for basic Bluetooth Low Energy support using the nRF24L01+ (basic support = sending & receiving on the advertising broadcast channel)
More info

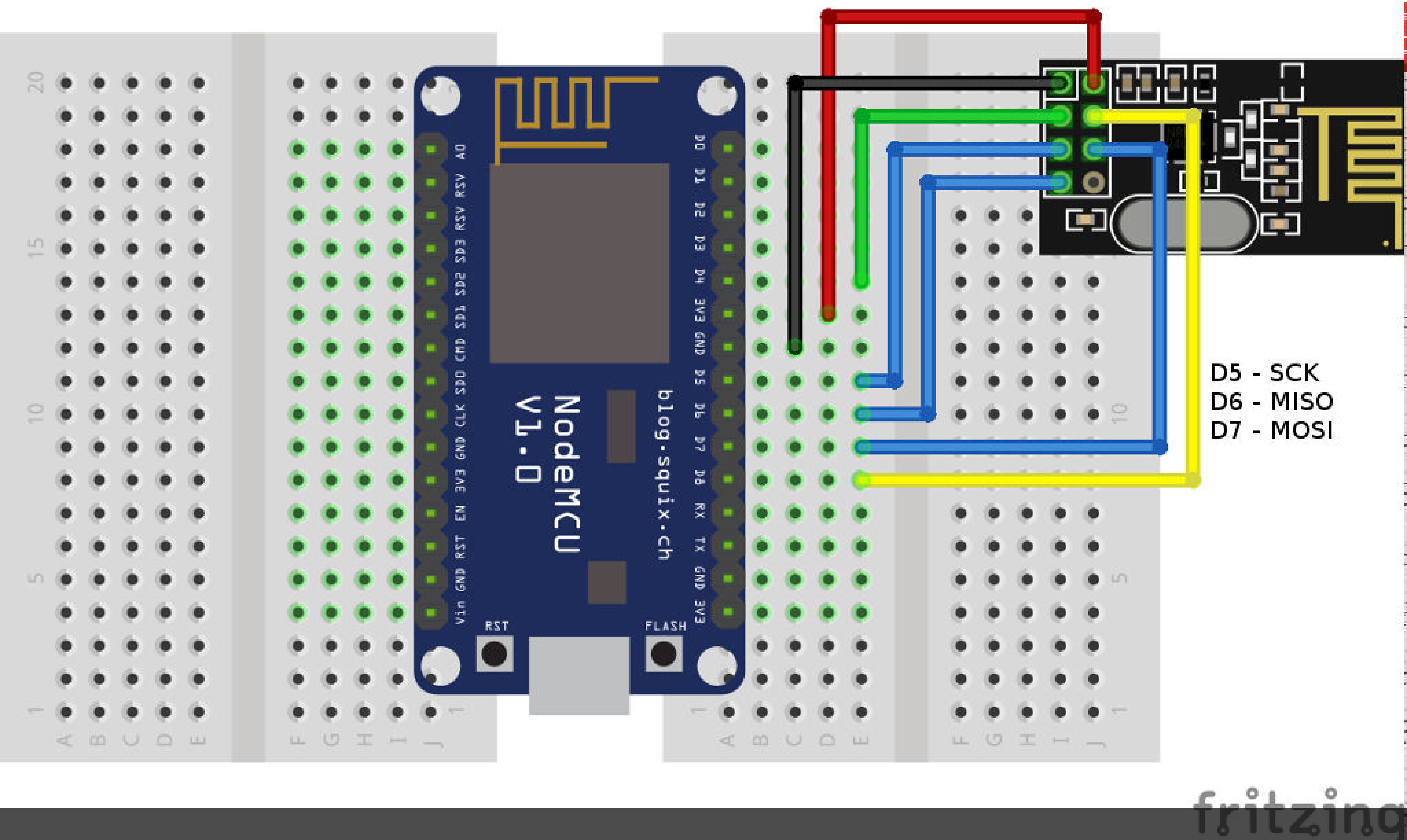
MySensors by The MySensors Team
Home Automation Framework Create your own wireless sensor mesh using NRF24L01+, RFM69 and RFM95 radios running on Arduino, SAMD and ESP8266. Allows over-the-air updates of nodes. Supported by 20+ home automation controllers.
More info

RF24 by **TMRh20**
A library for NRF24L01(+) communication. Optimized library for nRF24L01(+) that is simple to use for beginners, but yet offers a lot for advanced users. It also has a lot of good examples how to use the library.
More info

RF24Ethernet by TMRh20
Open TCP/IP wireless/radio IoT mesh networks for Arduino Self-sustaining wireless sensor networks that seamlessly link together using standard protocols & networking, based on low-power nrf24l01+ radio modules. An experiment in networking, protocols and

First Node - NodeMCU

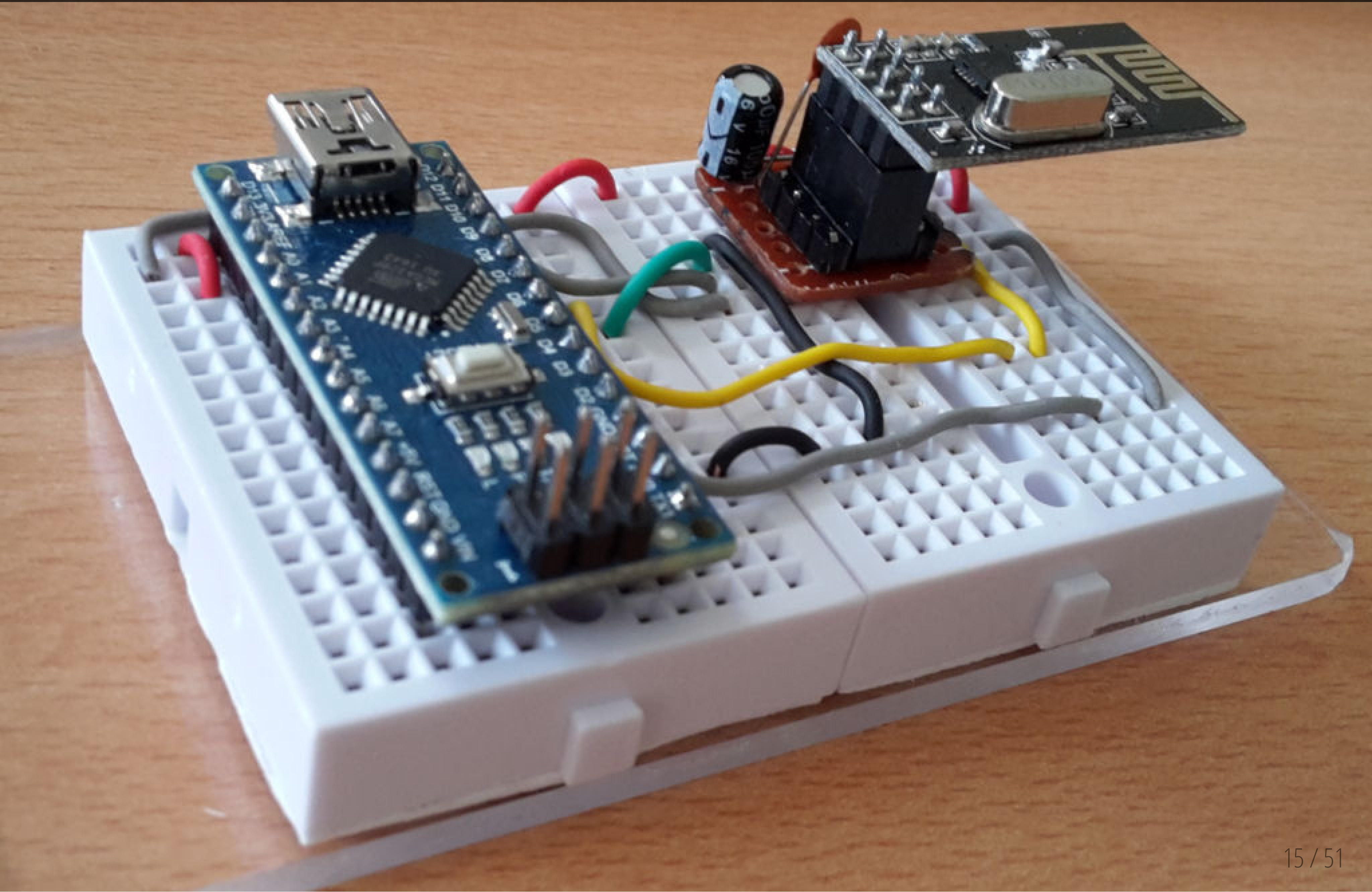


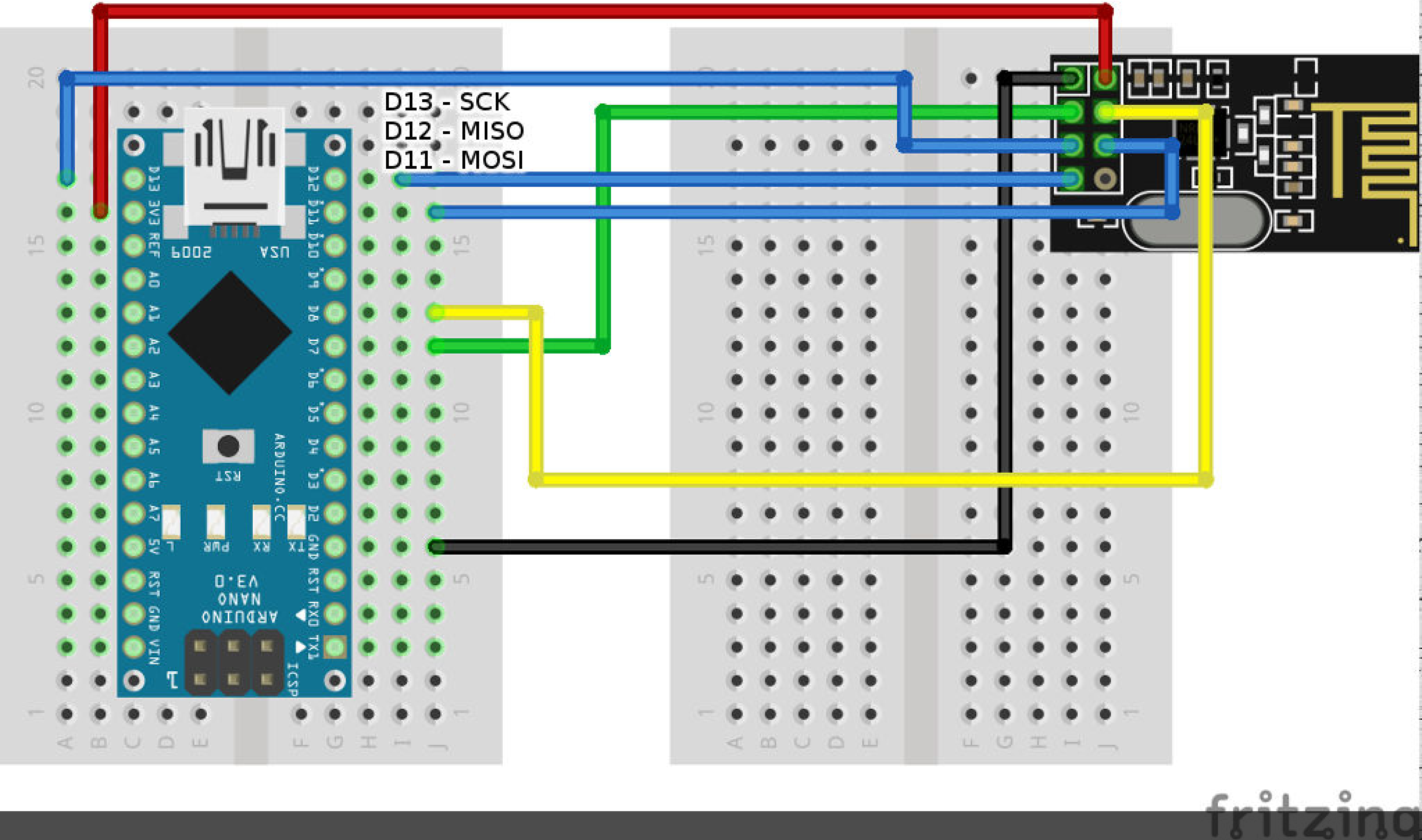


D5 - SCK
D6 - MISO
D7 - MOSI

First Node - NodeMCU

Second Node - Nano




D13 - SCK
D12 - MISO
D11 - MOSI

Second Node - Nano

Code & Play

Getting Started

Simple Transmit & Receive

NodeMCU - **Transmit** | Nano - **Receive**

Ref: Example Sketches

NodeMCU

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

RF24 myRadio (2, 15);

byte addresses[][6] = {"1Node"};
int dataTransmitted;

void setup()
{
  Serial.begin(115200);
  delay(1000);
  Serial.println(F("RF24/Simple Transmit data Test"));

  dataTransmitted = 100;

  myRadio.begin();
  myRadio.setChannel(108);
  myRadio.setPALevel(RF24_PA_MIN);

  myRadio.openWritingPipe( addresses[0]);
  delay(1000);
}

void loop()
{
  myRadio.write( &dataTransmitted, sizeof(dataTransmitted) );

  Serial.print(F("Data Transmitted = "));
  Serial.print(dataTransmitted);
  Serial.println(F(" No Acknowledge expected"));
  dataTransmitted = dataTransmitted + 1;
  delay(500);
}
```

NodeMCU

Serial

```
1384, room 16
tail 8
chksum
Data Transmitted = 100 No Acknowledge expected
Data Transmitted = 101 No Acknowledge expected
Data Transmitted = 102 No Acknowledge expected
Data Transmitted = 103 No Acknowledge expected
Data Transmitted = 104 No Acknowledge expected
Data Transmitted = 105 No Acknowledge expected
...
```

Nano

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

RF24 myRadio (7, 8);

byte addresses[][6] = {"1Node"};
int dataReceived;

void setup()
{
  Serial.begin(115200);
  delay(1000);
  Serial.println(F("RF24/Simple Receive data Test"));

  myRadio.begin();
  myRadio.setChannel(108);
  myRadio.setPALevel(RF24_PA_MIN);

  myRadio.openReadingPipe(1, addresses[0]);
  myRadio.startListening();

}

void loop()
{
  if (myRadio.available())
  {
    while (myRadio.available())
    {
      myRadio.read( &dataReceived, sizeof(dataReceived) );
    }

    Serial.print("Data received = ");
    Serial.println(dataReceived);
  }
}
```

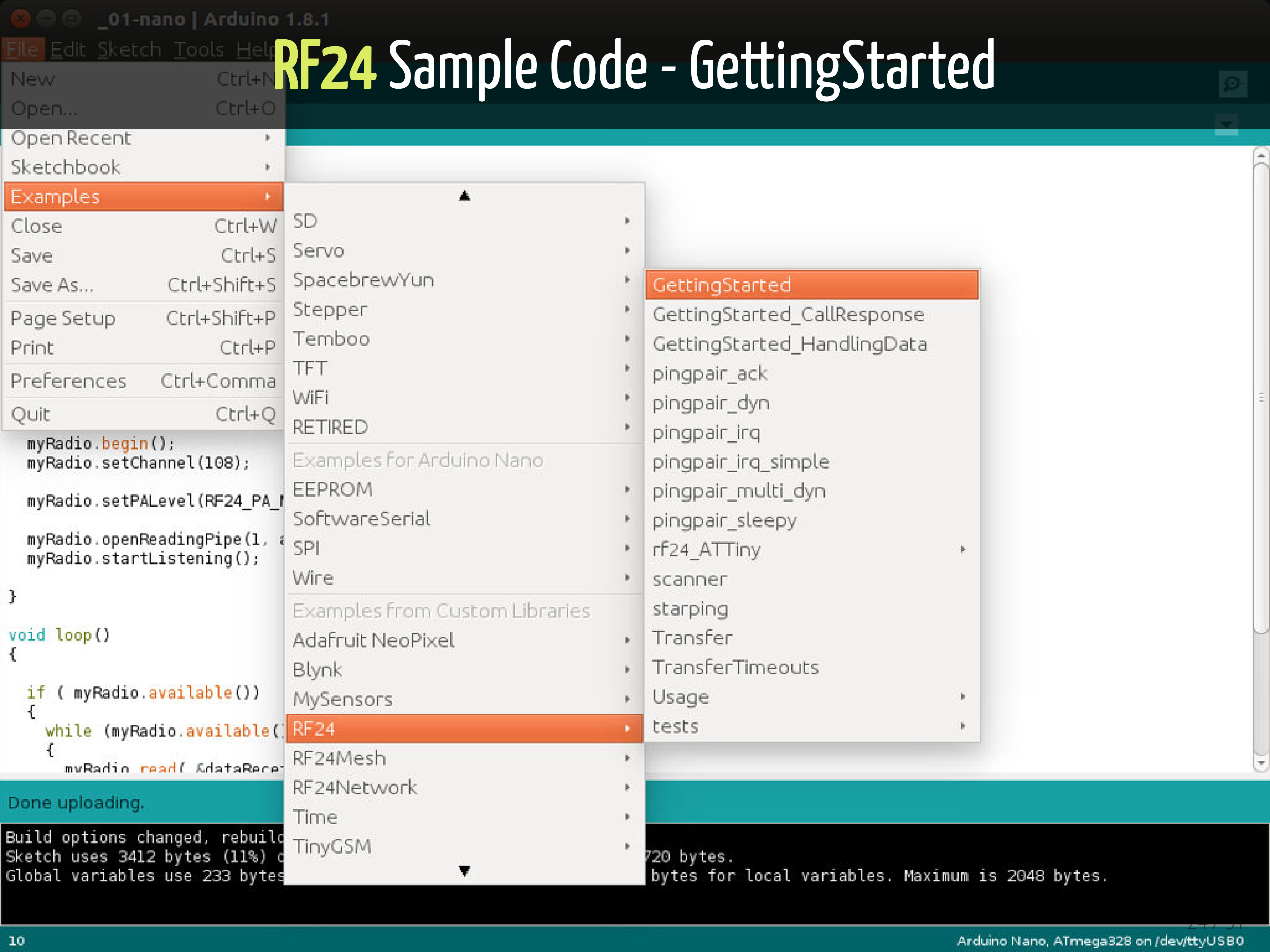
Nano

Serial

```
RF24/Simple Receive data Test
Data received = 100
Data received = 101
Data received = 102
Data received = 103
Data received = 104
Data received = 105
...
```

RF24 Sample Code

RF24 Sample Code - GettingStarted



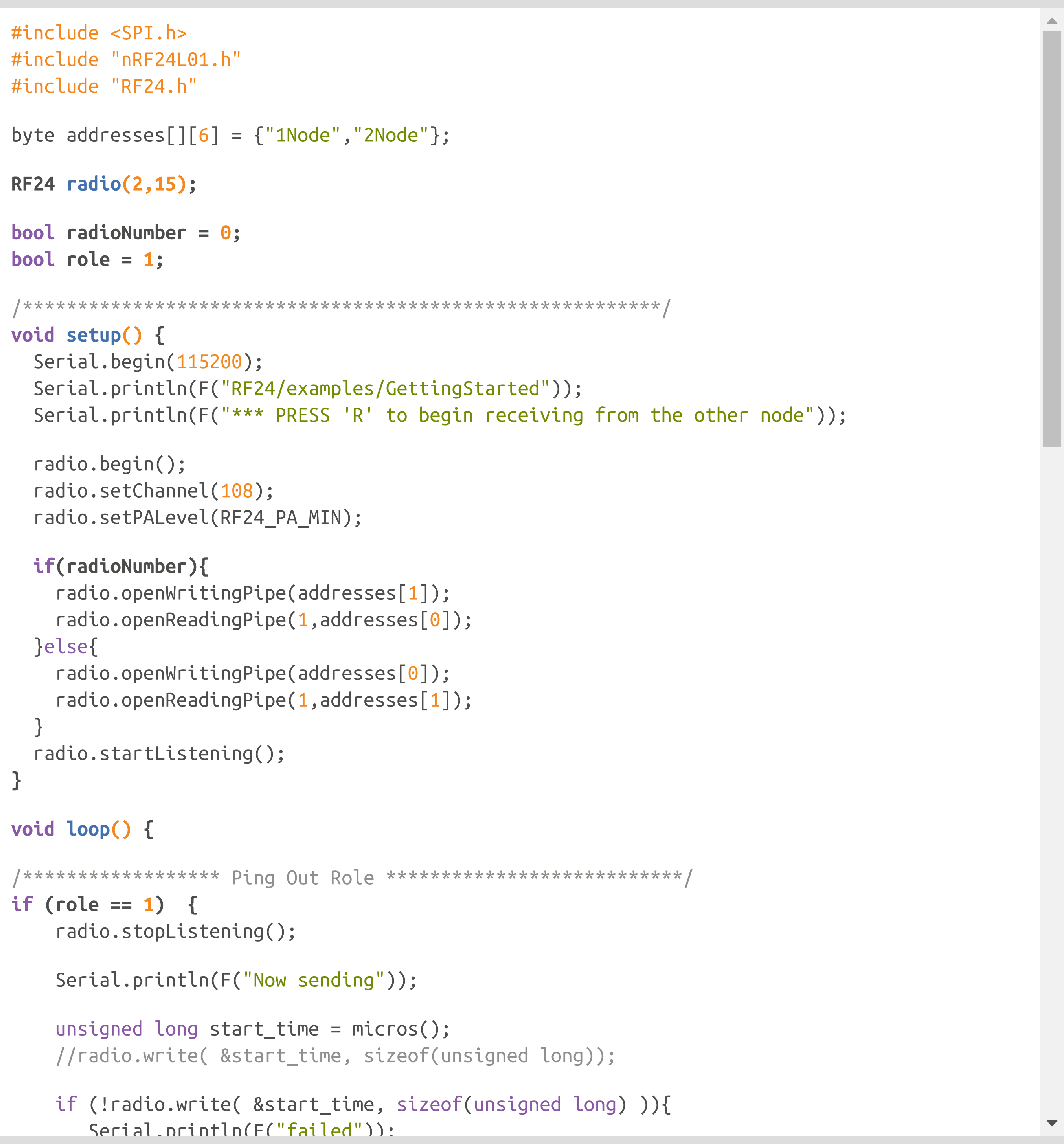
```
  myRadio.begin();
  myRadio.setChannel(108);

  myRadio.setPALevel(RF24_PA_I

  myRadio.openReadingPipe(1, a
  myRadio.startListening();

}

void loop()
{

  if ( myRadio.available())
  {
    while (myRadio.available(
    {
      myRadio.read( &dataRece
```

Done uploading.

```
Build options changed, rebuild
Sketch uses 3412 bytes (11%) o                              720 bytes.
Global variables use 233 bytes                              bytes for local variables. Maximum is 2048 bytes.
```

10

Arduino Nano, ATmega328 on /dev/ttyUSB0



```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

byte addresses[][6] = {"1Node","2Node"};

RF24 radio(2,15);

bool radioNumber = 0;
bool role = 1;

/**********************************************************/
void setup() {
  Serial.begin(115200);
  Serial.println(F("RF24/examples/GettingStarted"));
  Serial.println(F("*** PRESS 'R' to begin receiving from the other node"));

  radio.begin();
  radio.setChannel(108);
  radio.setPALevel(RF24_PA_MIN);

  if(radioNumber){
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1,addresses[0]);
  }else{
    radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1,addresses[1]);
  }
  radio.startListening();
}

void loop() {

/****************** Ping Out Role ***************************/
if (role == 1)  {
    radio.stopListening();

    Serial.println(F("Now sending"));

    unsigned long start_time = micros();
    //radio.write( &start_time, sizeof(unsigned long));

    if (!radio.write( &start_time, sizeof(unsigned long) )){
       Serial.println(F("failed"));
```

NodeMCU

NodeMCU

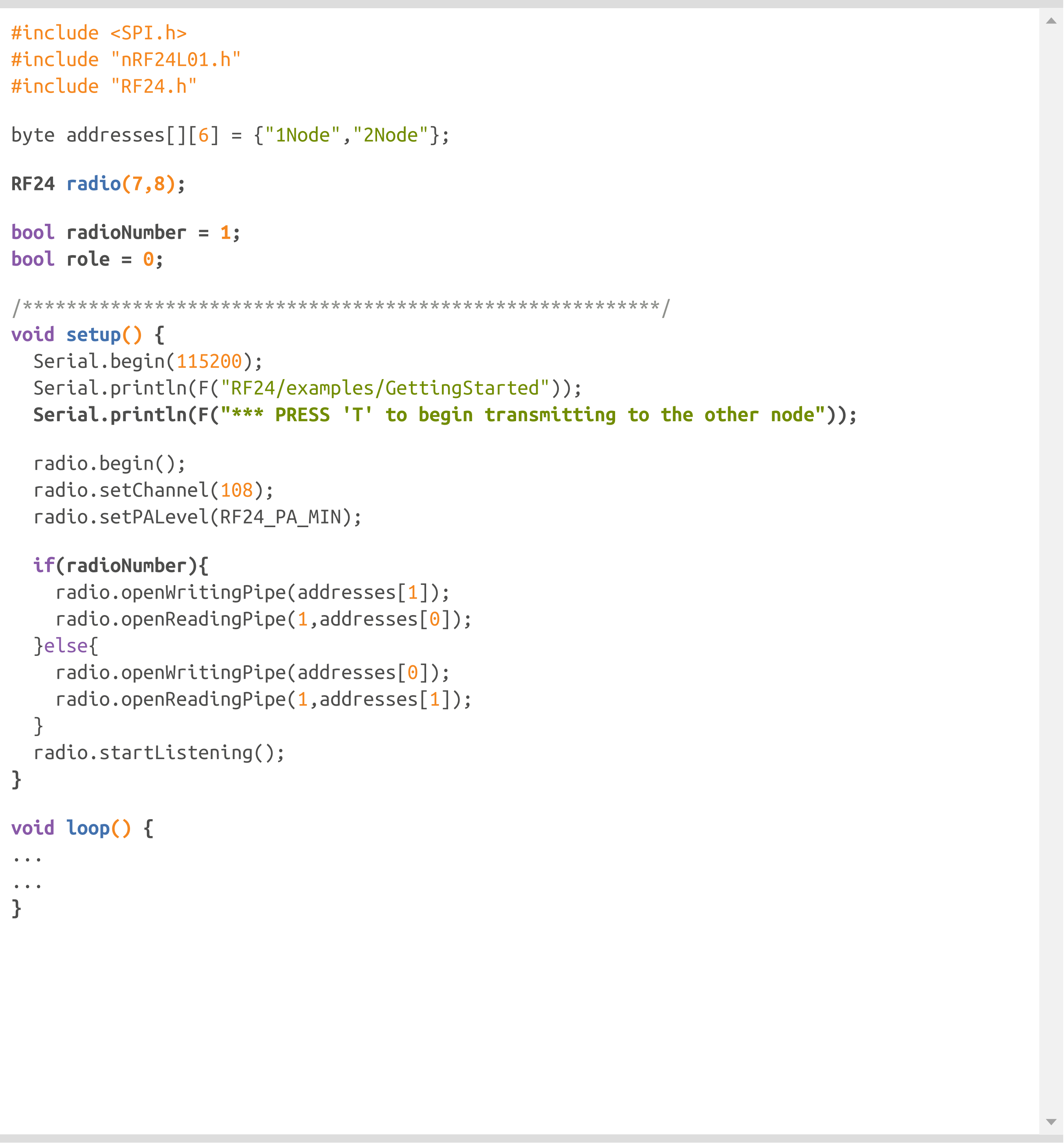
Serial

^$#%$#@*&%)# Why??

Nevermind for now!
Unplug NodeMCU, Plug-In Nano ..

```
Now sending
failed
Failed, response timed out.
Now sending
failed
Failed, response timed out.
Now sending
failed
Failed, response timed out.
Now sending
failed
Failed, response timed out.
Now sending
failed
Failed, response timed out.
Now sending
failed
Failed, response timed out.
...
```

Nano



```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

byte addresses[][6] = {"1Node","2Node"};

RF24 radio(7,8);

bool radioNumber = 1;
bool role = 0;

/**********************************************************/
void setup() {
  Serial.begin(115200);
  Serial.println(F("RF24/examples/GettingStarted"));
  Serial.println(F("*** PRESS 'T' to begin transmitting to the other node"));

  radio.begin();
  radio.setChannel(108);
  radio.setPALevel(RF24_PA_MIN);

  if(radioNumber){
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1,addresses[0]);
  }else{
    radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1,addresses[1]);
  }
  radio.startListening();
}

void loop() {
...
...
}
```

Nano

Serial

Get Back to NodeMCU, Switch It On!

```
RF24/examples/GettingStarted
** PRESS 'T' to begin transmitting to the other node

# After NodeMCU Switched ON
Sent response 9284083
Sent response 10286475
Sent response 11288847
Sent response 12291268
Sent response 13293653
...
```

NodeMCU

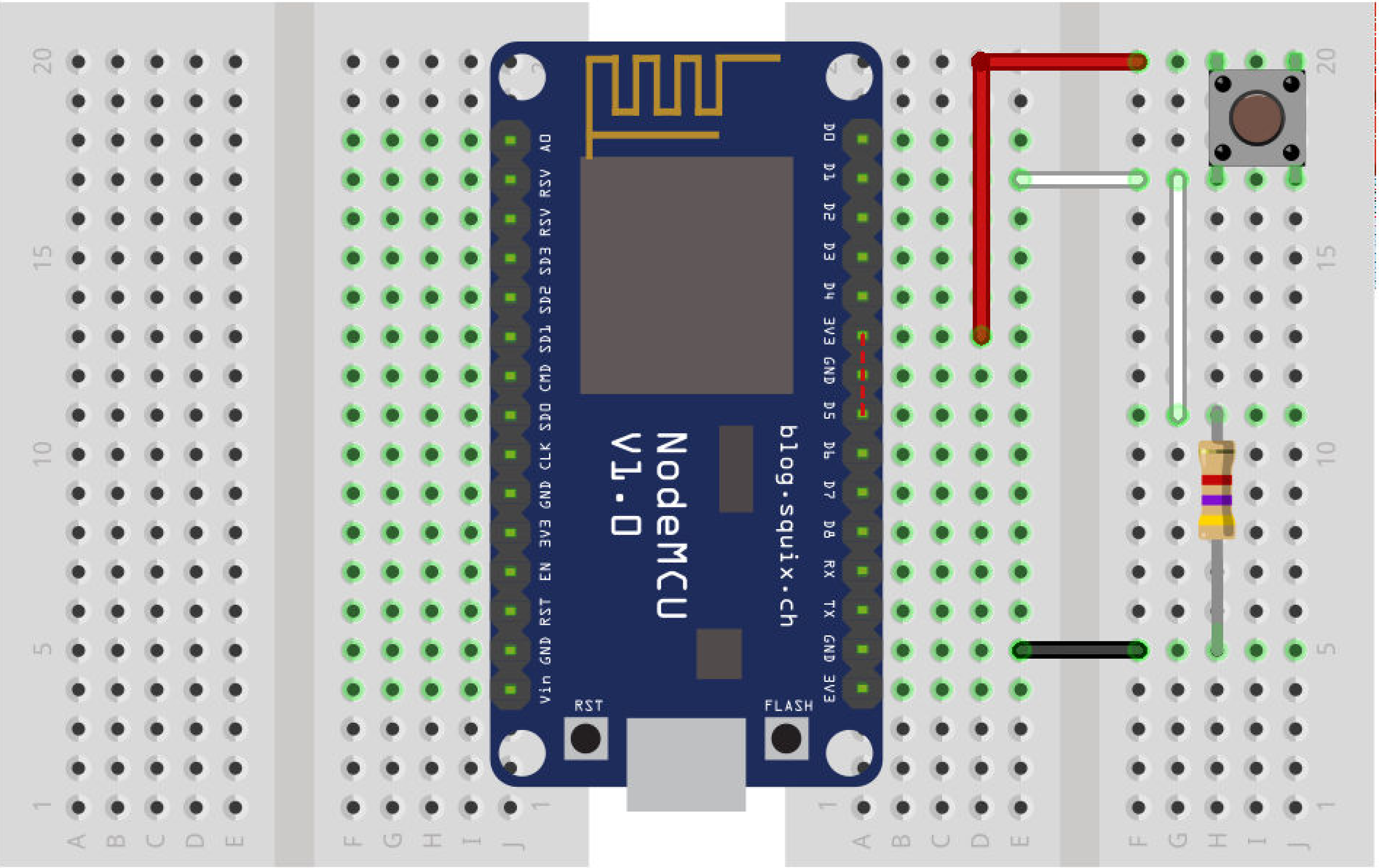
Serial - Take 2

Find Another Serial Console..

Now It Looks Good.. Explain!

```
Now sending
Sent 18612291, Got response 18612291, Round-trip delay 1828 microseconds
Now sending
Sent 19614686, Got response 19614686, Round-trip delay 1840 microseconds
Now sending
Sent 20617552, Got response 20617552, Round-trip delay 1803 microseconds
Now sending
Sent 21619866, Got response 21619866, Round-trip delay 1800 microseconds
Now sending
Sent 22622153, Got response 22622153, Round-trip delay 1806 microseconds
Now sending
Sent 23624535, Got response 23624535, Round-trip delay 1840 microseconds
...
```

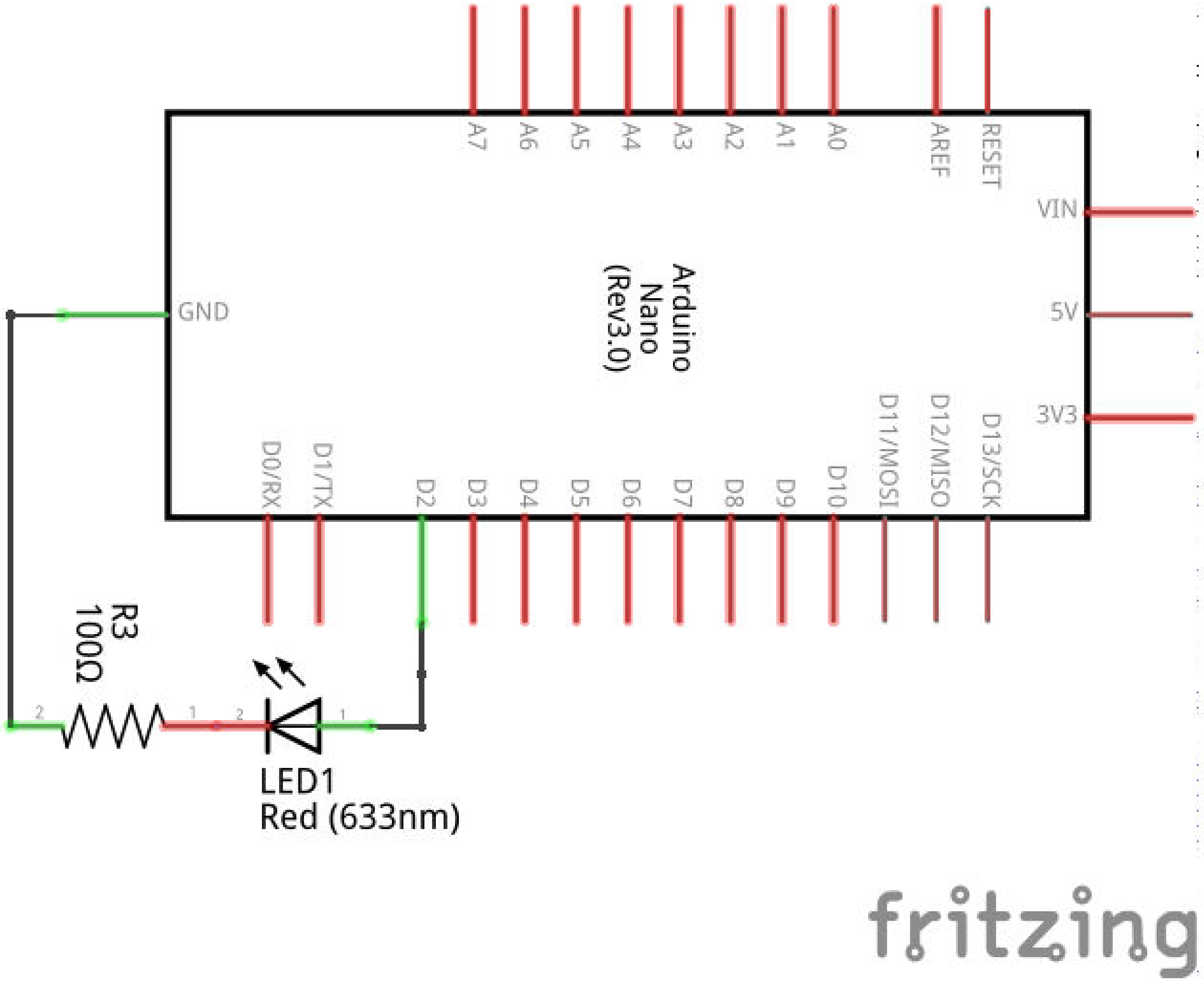
Simple Remote Control



NodeMCU
V1.0
blog.squix.ch
RST
FLASH
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3V3 EN RST GND Vin
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
A B C D E F G H I J
1 5 10 15 20

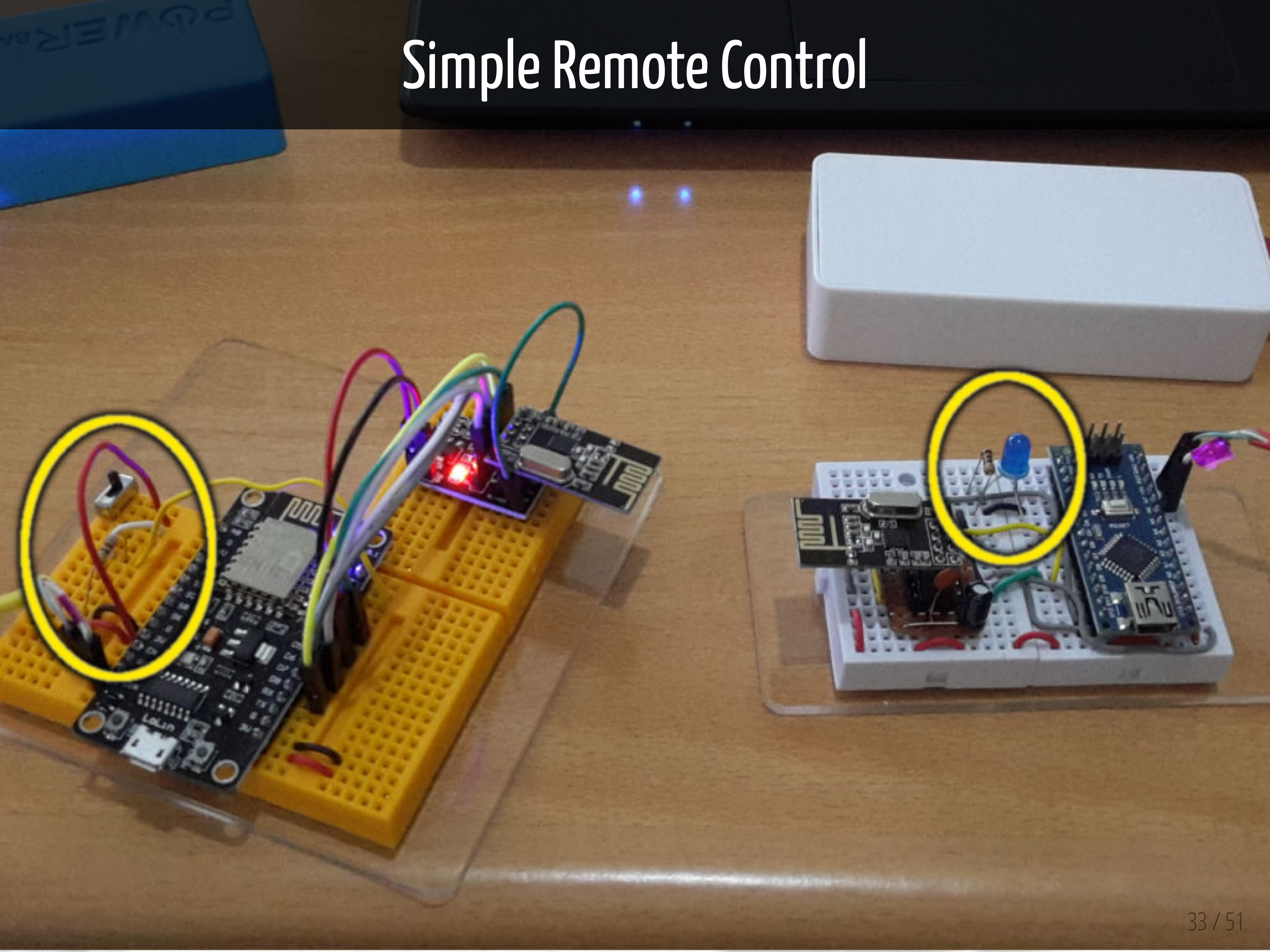
NodeMCU - Remote Controller

fritzing



Nano - Local Controller

Simple Remote Control



NodeMCU

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

RF24 myRadio (2, 15);
const int SW1 = 5;

byte addresses[][6] = {"1Node"};
int dataTransmitted;
int button;

void setup()
{
  pinMode(SW1, INPUT);
  dataTransmitted = 10;
  button = 0;

  Serial.begin(115200);
  delay(1000);

  myRadio.begin();
  myRadio.setChannel(108);
  myRadio.setPALevel(RF24_PA_MIN);

  myRadio.openWritingPipe( addresses[0]);
  delay(1000);
}

void loop()
{
  int newButton = digitalRead(SW1);
  if (newButton != button) {
    button = newButton;

    if (button == HIGH){
      dataTransmitted = 20;
    }
    else {
      dataTransmitted = 10;
    }
    myRadio.write( &dataTransmitted, sizeof(dataTransmitted) );
    Serial.print(F("Data Transmitted = "));
    Serial.println(dataTransmitted);
```

NodeMCU

Serial

After Some ON-OFFs

```
1384, room 16
tail 8
chksum
Data Transmitted = 20
Data Transmitted = 10
Data Transmitted = 20
Data Transmitted = 10
Data Transmitted = 20
Data Transmitted = 10
Data Transmitted = 20
```

Nano

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

RF24 myRadio (7, 8);
const int LED = 2;

byte addresses[][6] = {"1Node"};
int dataReceived;

void setup()
{
  pinMode(LED, OUTPUT);

  Serial.begin(115200);
  delay(1000);

  myRadio.begin();
  myRadio.setChannel(108);
  myRadio.setPALevel(RF24_PA_MIN);

  myRadio.openReadingPipe(1, addresses[0]);
  myRadio.startListening();
}

void loop()
{
  if (myRadio.available())
  {
    while (myRadio.available())
    {
      myRadio.read( &dataReceived, sizeof(dataReceived) );
    }

    Serial.print("Data received = ");
    Serial.println(dataReceived);

    if (dataReceived == 10) {
      digitalWrite(LED, LOW);
    } else {
      digitalWrite(LED, HIGH);
    }
  }
```

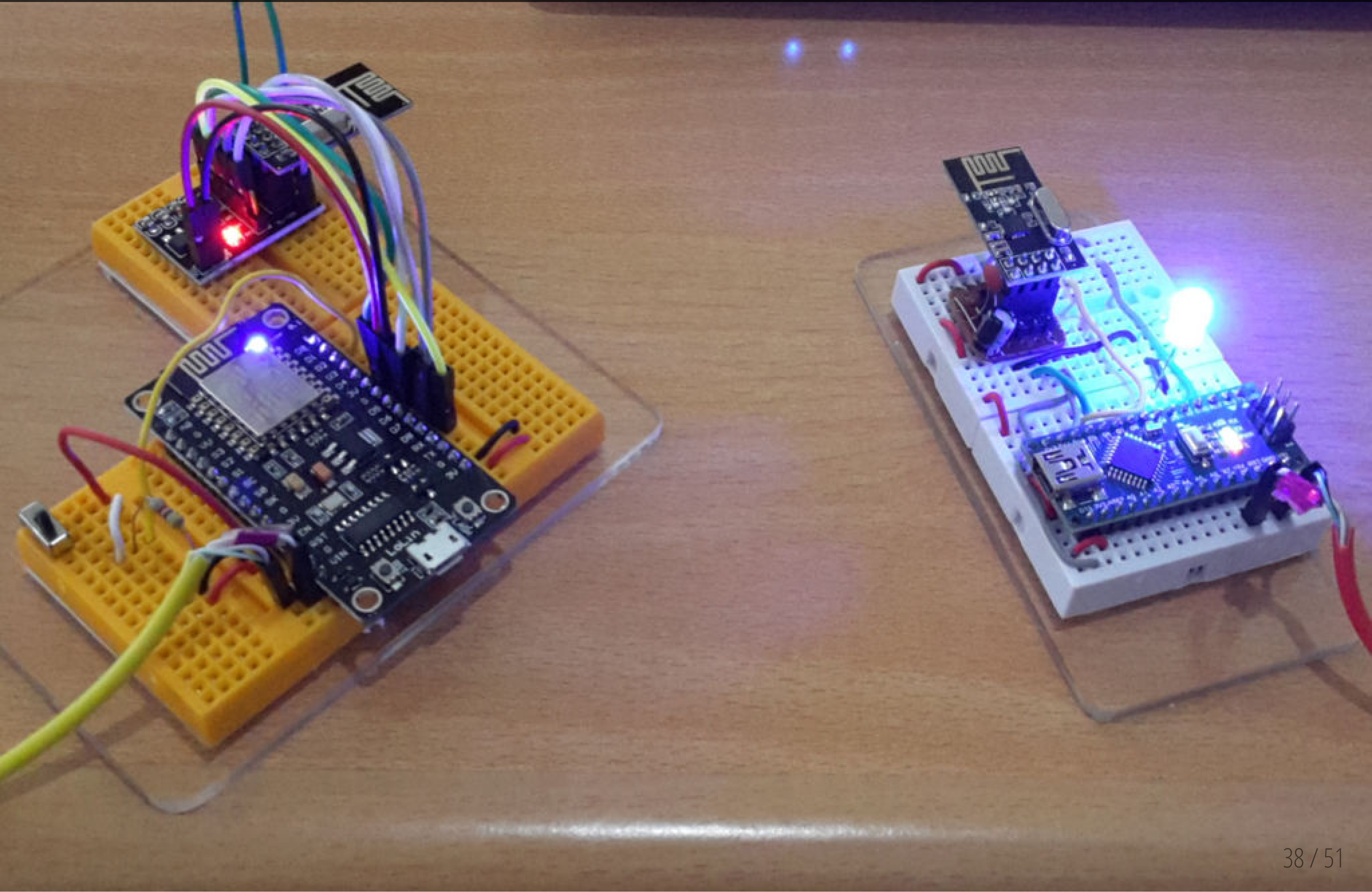
Nano

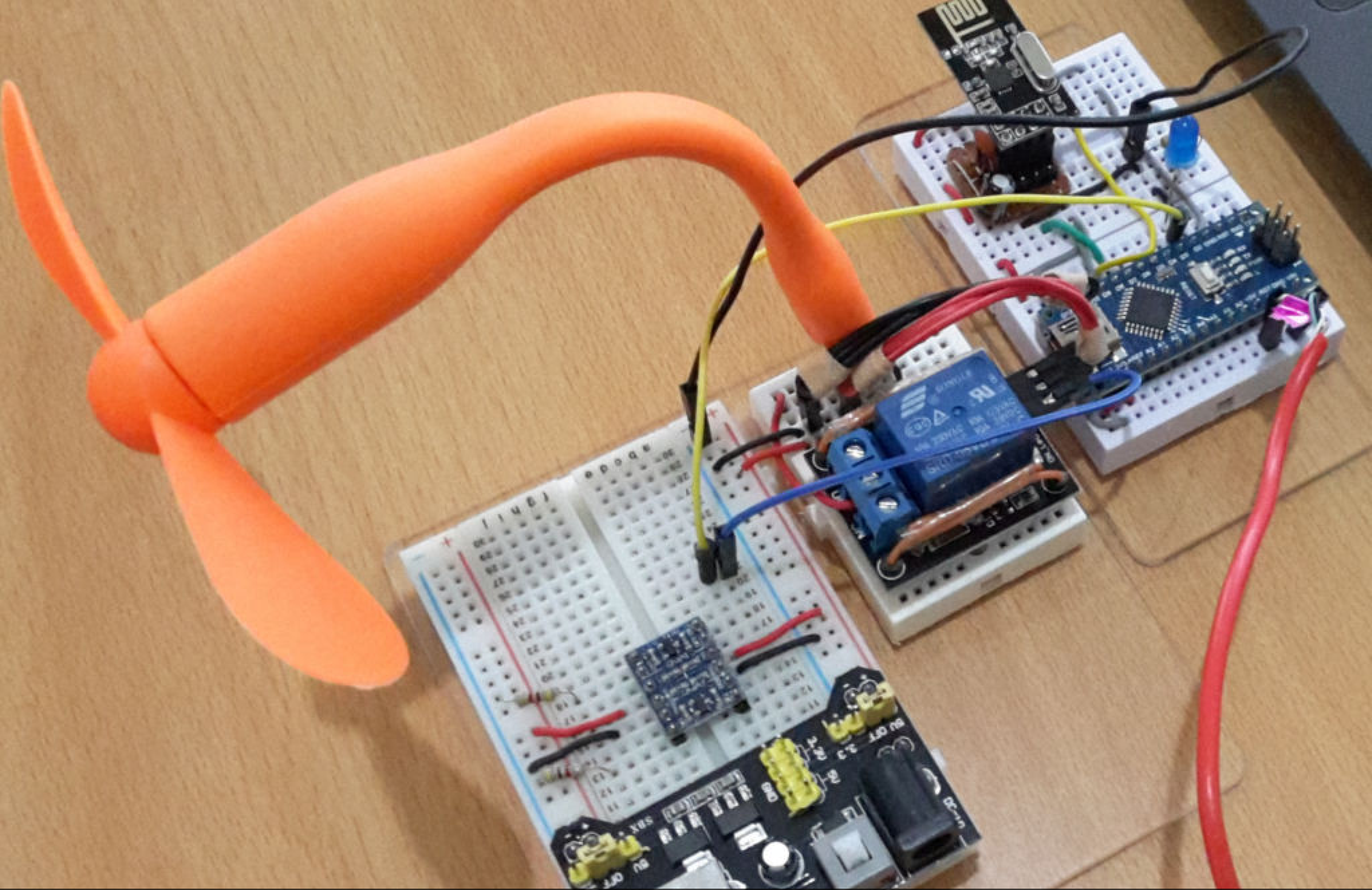
Serial

After Some ON-OFFs

```
Data received = 20
Data received = 10
Data received = 20
Data received = 10
Data received = 20
Data received = 10
Data received = 20
Data received = 10
Data received = 20
Data received = 10
```

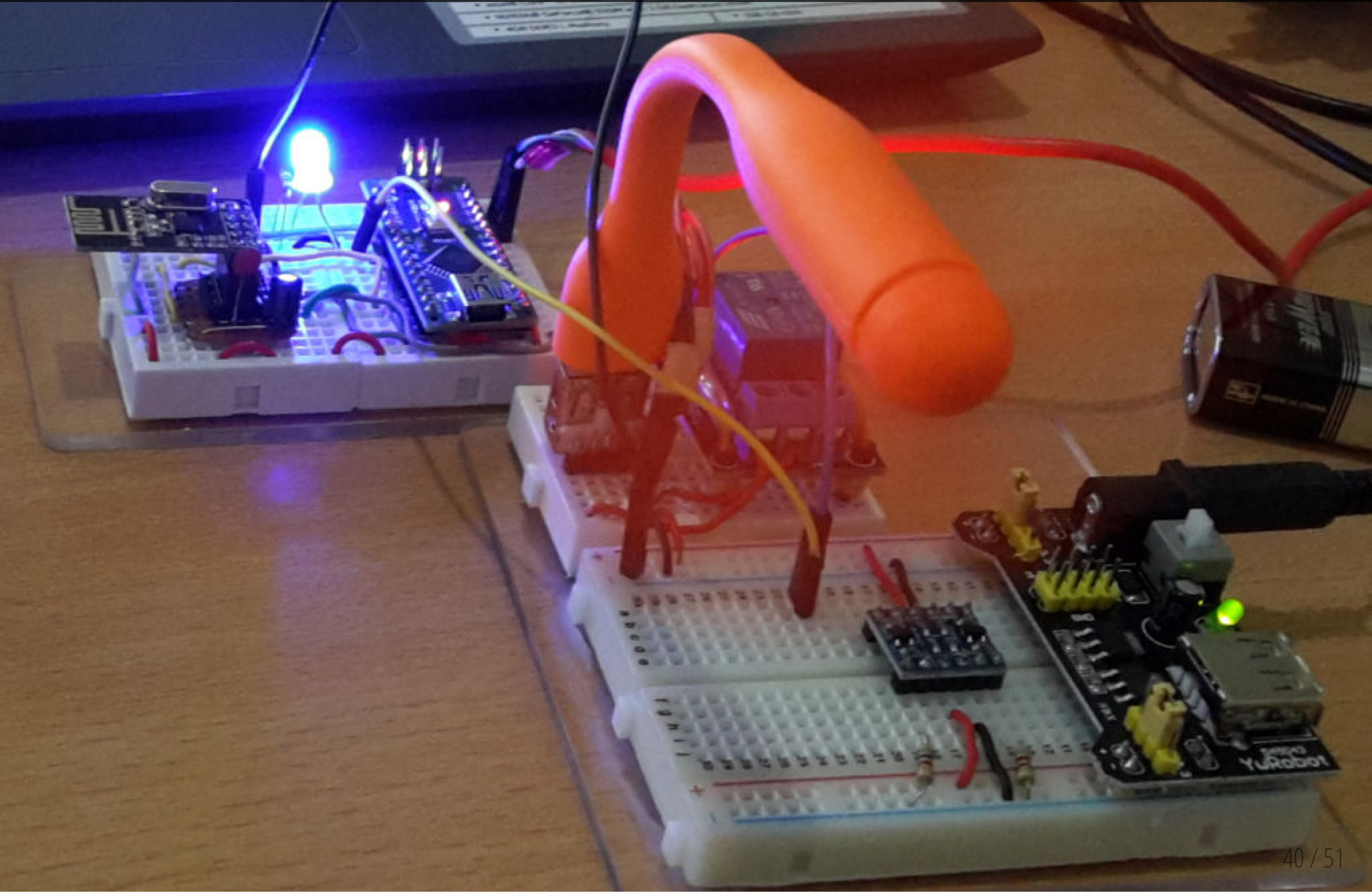
Simple Remote Control





Nano - Attach a Device

Nano - Attach a Device



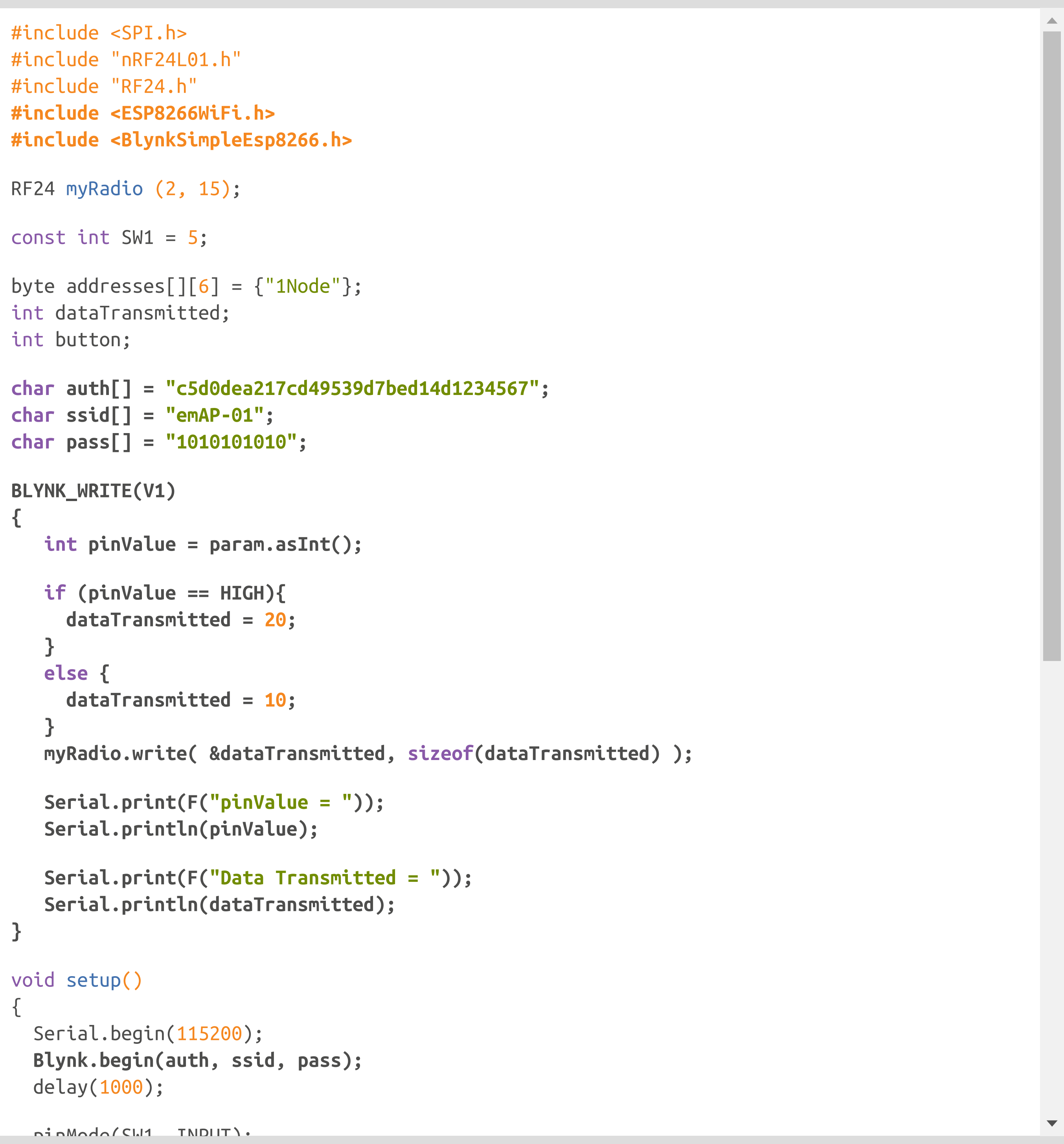
Connecting to Blynk Cloud

Gateway

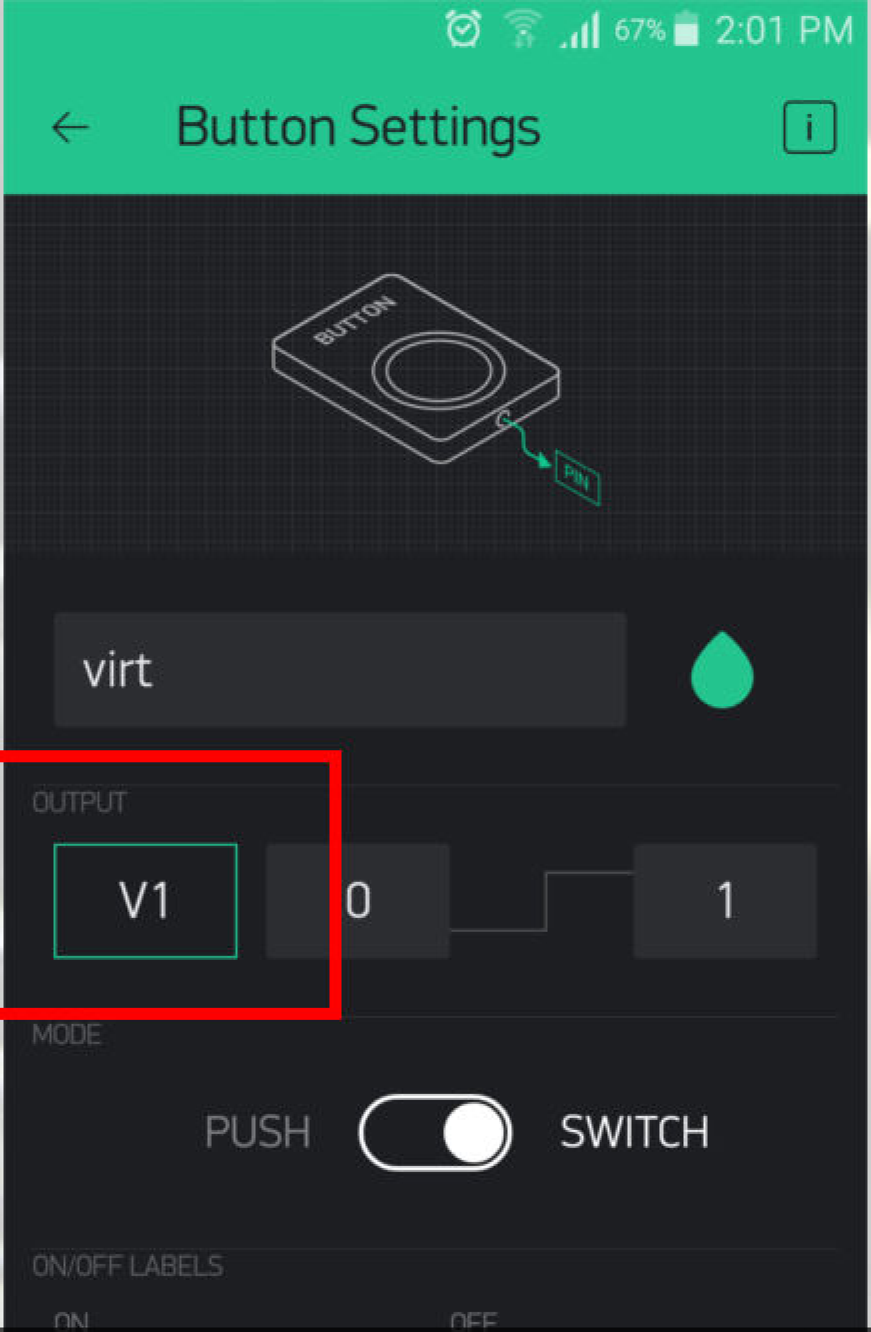
Notes

This is only an example of integration of local-connected **sensors** and **actuators** to other (cloud-based) services. This is applicable not only for **Blynk** or **Firebase**, but also for other services.

NodeMCU



```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

RF24 myRadio (2, 15);

const int SW1 = 5;

byte addresses[][6] = {"1Node"};
int dataTransmitted;
int button;

char auth[] = "c5d0dea217cd49539d7bed14d1234567";
char ssid[] = "emAP-01";
char pass[] = "1010101010";

BLYNK_WRITE(V1)
{
   int pinValue = param.asInt();

   if (pinValue == HIGH){
     dataTransmitted = 20;
   }
   else {
     dataTransmitted = 10;
   }
   myRadio.write( &dataTransmitted, sizeof(dataTransmitted) );

   Serial.print(F("pinValue = "));
   Serial.println(pinValue);

   Serial.print(F("Data Transmitted = "));
   Serial.println(dataTransmitted);
}

void setup()
{
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  delay(1000);

  pinMode(SW1, INPUT);
```



Button Settings

virt

OUTPUT

V1 | 0 | 1

MODE

PUSH ● SWITCH

ON/OFF LABELS

ON: ON

OFF: OFF

Blynk Button with Virtual Pin V1

NodeMCU

Serial

After Some ON-OFFs via Physical Button and **Blynk** Virtual Button

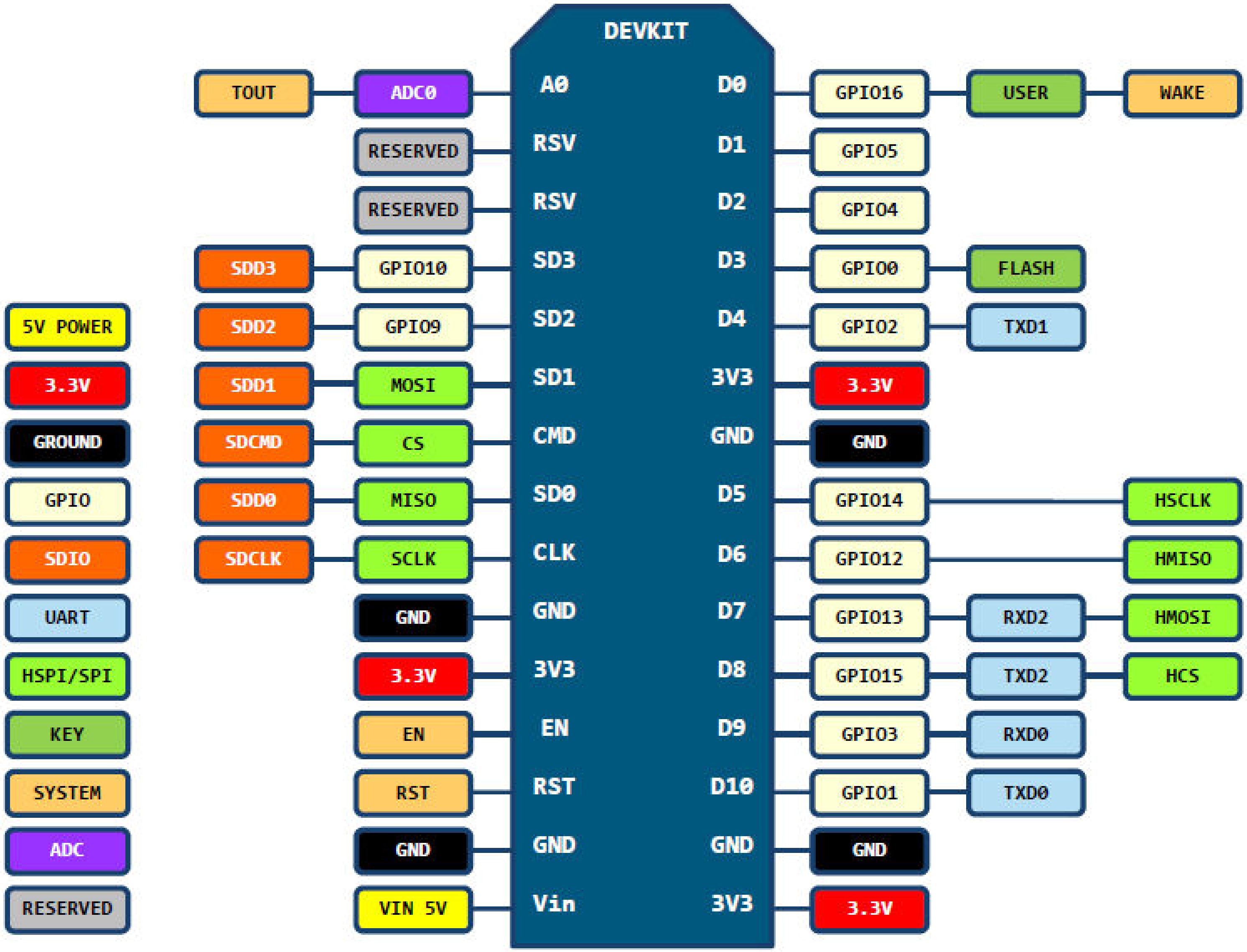
```
1384, room 16
Data Transmitted = 20
Data Transmitted = 10
pinValue = 1
Data Transmitted = 20
pinValue = 0
Data Transmitted = 10
pinValue = 1
Data Transmitted = 20
pinValue = 0
Data Transmitted = 10
pinValue = 1
Data Transmitted = 20
pinValue = 0
Data Transmitted = 10
pinValue = 1
Data Transmitted = 20
Data Transmitted = 20
Data Transmitted = 10
pinValue = 0
Data Transmitted = 10
Data Transmitted = 20
pinValue = 1
Data Transmitted = 20
pinValue = 0
Data Transmitted = 10
Data Transmitted = 10
```

Refs/Resources

Refs/Resources

1. Nordic Semiconductor
2. Example Sketches @arduino-info
3. Connecting the Radio | MySensors
4. nRF24/RF24: Optimized fork of nRF24L01 for Arduino & Raspberry Pi/Linux Devices

PIN DEFINITION NodeMCU V1.0 Pin Map



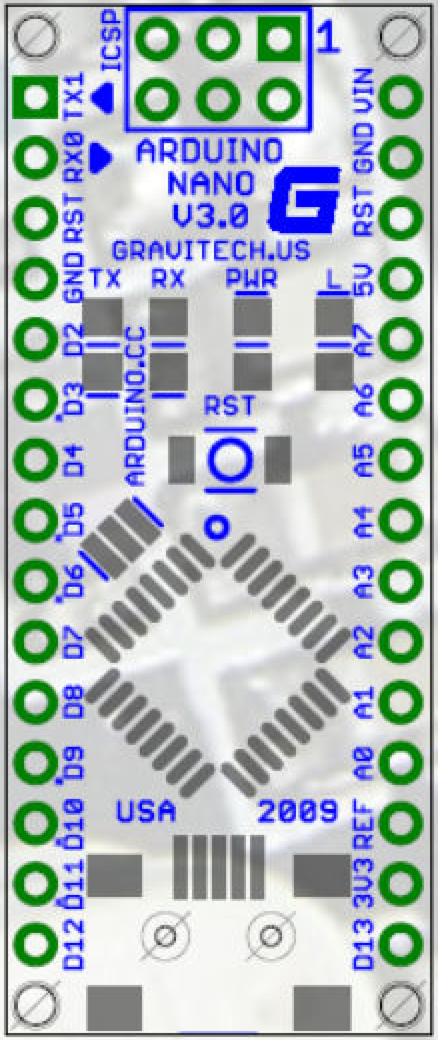
DEVKIT

TOUT	ADC0	A0	D0	GPIO16	USER	WAKE
	RESERVED	RSV	D1	GPIO5		
	RESERVED	RSV	D2	GPIO4		
SDD3	GPIO10	SD3	D3	GPIO0	FLASH	
SDD2	GPIO9	SD2	D4	GPIO2	TXD1	
SDD1	MOSI	SD1	3V3	3.3V		
SDCMD	CS	CMD	GND	GND		
SDD0	MISO	SD0	D5	GPIO14		HSCLK
SDCLK	SCLK	CLK	D6	GPIO12		HMISO
	GND	GND	D7	GPIO13	RXD2	HMOSI
	3.3V	3V3	D8	GPIO15	TXD2	HCS
	EN	EN	D9	GPIO3	RXD0	
	RST	RST	D10	GPIO1	TXD0	
	GND	GND	GND	GND		
	VIN 5V	Vin	3V3	3.3V		

- 5V POWER
- 3.3V
- GROUND
- GPIO
- SDIO
- UART
- HSPI/SPI
- KEY
- SYSTEM
- ADC
- RESERVED

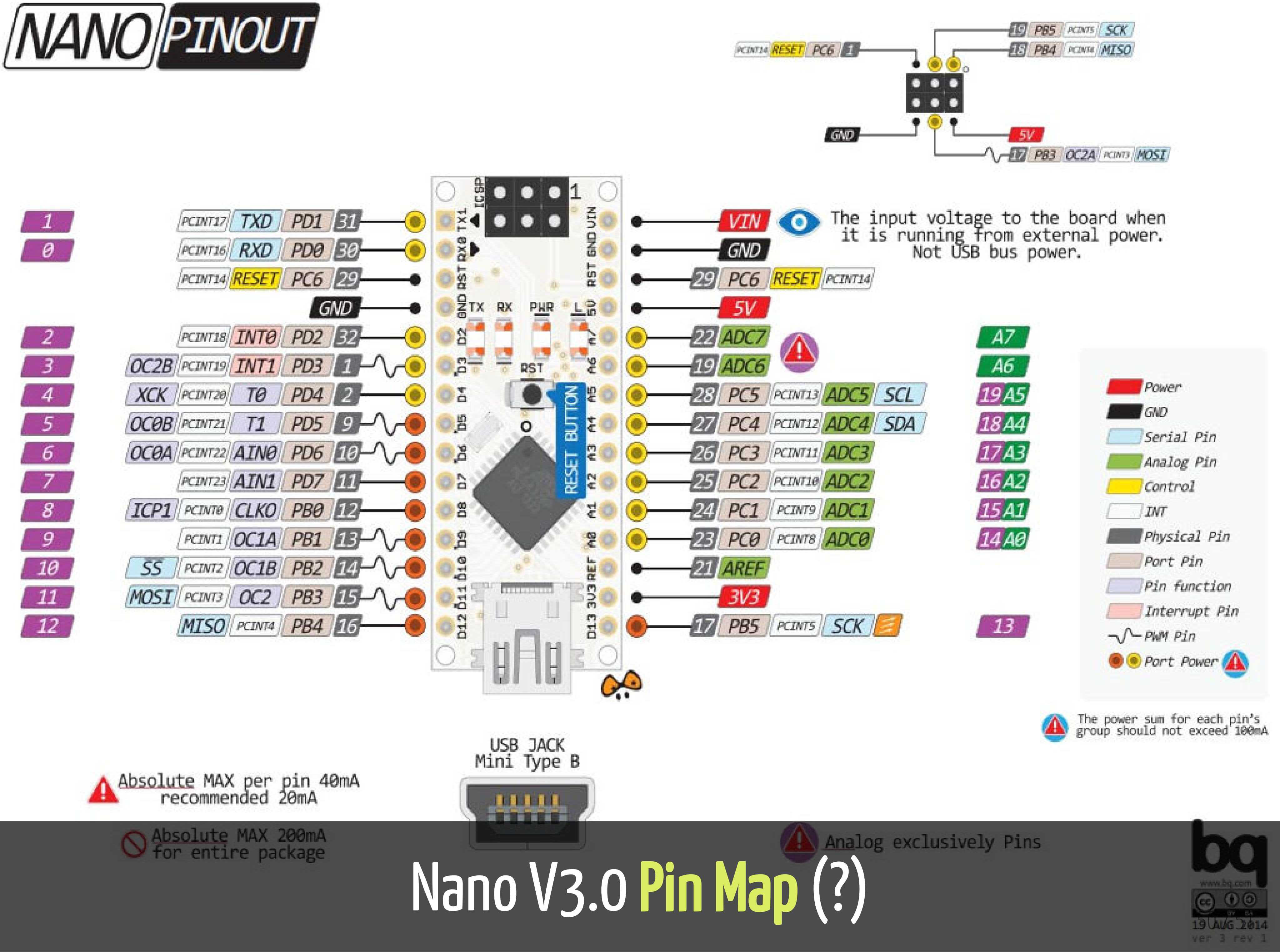
D0(GPIO16) can only be used as gpio read/write, no interrupt supported, no pwm/i2c/ow supported.

Nano V3.0 Pin Map



D1/TX	(1)	(30)	VIN
D0/RX	(2)	(29)	GND
RESET	(3)	(28)	RESET
GND	(4)	(27)	+5V
D2	(5)	(26)	A7
D3	(6)	(25)	A6
D4	(7)	(24)	A5
D5	(8)	(23)	A4
D6	(9)	(22)	A3
D7	(10)	(21)	A2
D8	(11)	(20)	A1
D9	(12)	(19)	A0
D10	(13)	(18)	AREF
D11	(14)	(17)	3V3
D12	(15)	(16)	D13

Nano V3.0 Pin Map (?)



END

Eueung Mulyana

https://eueung.github.io/012017/nrf24

CodeLabs | Attribution-ShareAlike CC BY-SA