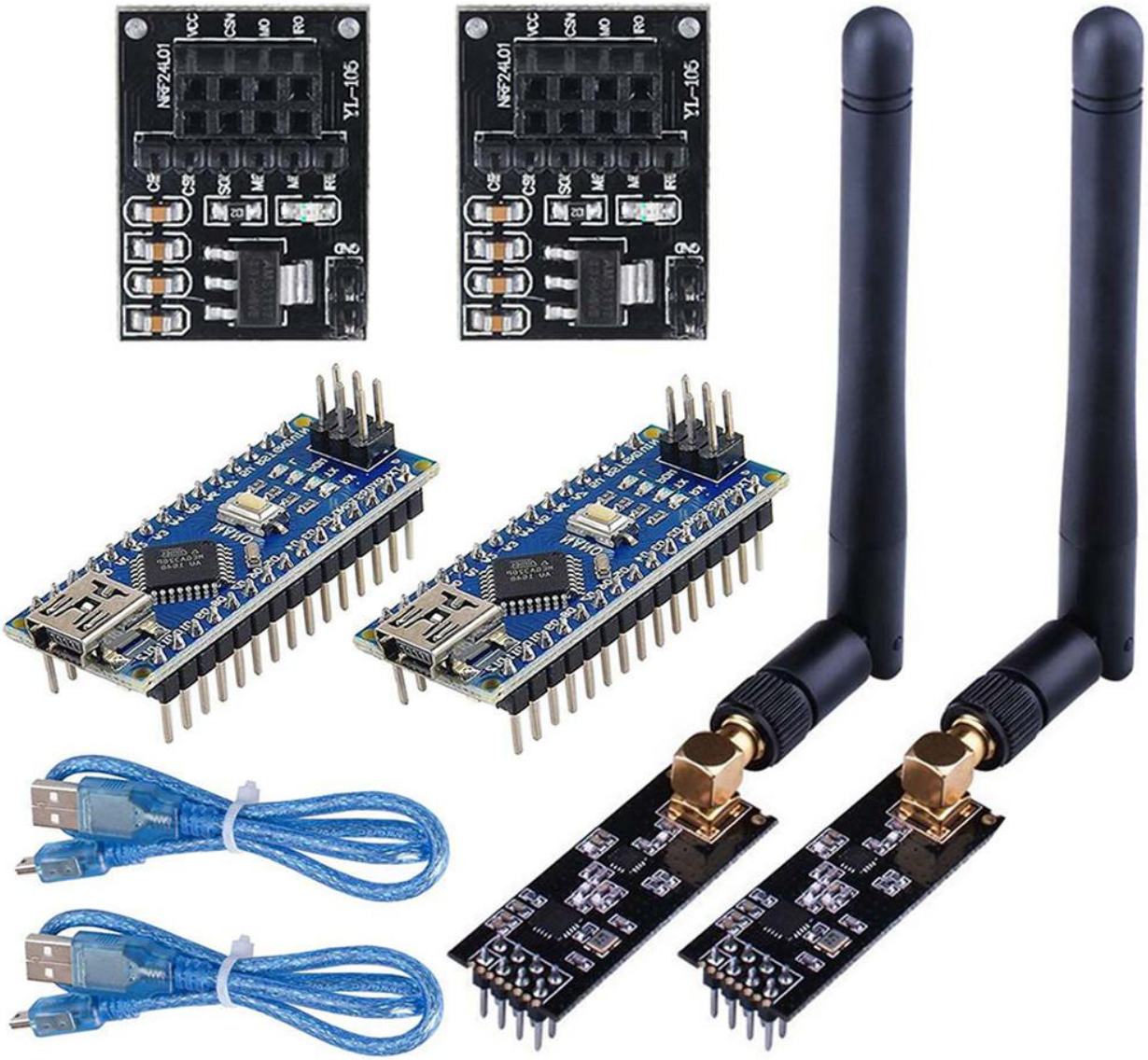
Preface

About Our Company

WayinTop, Your Top Way to Inspiration, is a professional manufacturer over 2,000 open source motherboards, modules, and components. From designing PCBs, printing, soldering, testing, debugging, and offering online tutorials, WayinTop has been committed to explore and demystify the wonderful world of embedded electronics, including but not limited to Arduino and Raspberry Pi. We aim to make the best designed products for makers of all ages and skill levels. No matter your vision or skill level, our products and resources are designed to make electronics more accessible. Founded in 2013, WayinTop has grown to over 100+ employees and a 50,000+ sq ft. factory in China by now. With our unremitting efforts, we also have expanded offerings to include tools, equipments, connector kits and various DIY products that we have carefully selected and tested.

US Amazon Store Homepage:
https://www.amazon.com/shops/A22PZZC3JNHS9L
CA Amazon Store Homepage:
https://www.amazon.ca/shops/A22PZZC3JNHS9L
UK Amazon Store Homepage:
https://www.amazon.co.uk/shops/A3F8F97TMOROPI
DE Amazon Store Homepage:
https://www.amazon.de/shops/A3F8F97TMOROPI
FR Amazon Store Homepage:
https://www.amazon.fr/shops/A3F8F97TMOROPI
IT Amazon Store Homepage:
https://www.amazon.it/shops/A3F8F97TMOROPI
ES Amazon Store Homepage:
https://www.amazon.es/shops/A3F8F97TMOROPI
JP Amazon Store Homepage:
https://www.amazon.co.jp/shops/A1F5OUAXY2TP0K

Wireless Communication System Based on Arduino Nano + NRF24L01



Overview

The arduino nano and NRF24L01 wireless transceiver modules can be used to realize wireless communication between modules. The applications of the system include data wireless transmission and reception, and remote wireless remote control. This tutorial focuses on the combination of two modules that can show you simple wireless data transfer.

Parts Required

2 x Arduino Nano Main Control Board

2 x NRF24L01 Wireless Transceiver Module

2 x Breadboard

2 x NRF24L01 Wireless Transceiver Module Regulator

N x Jumper Wires

Warm Tips:

Please copy the required code in the "The Code" folder.

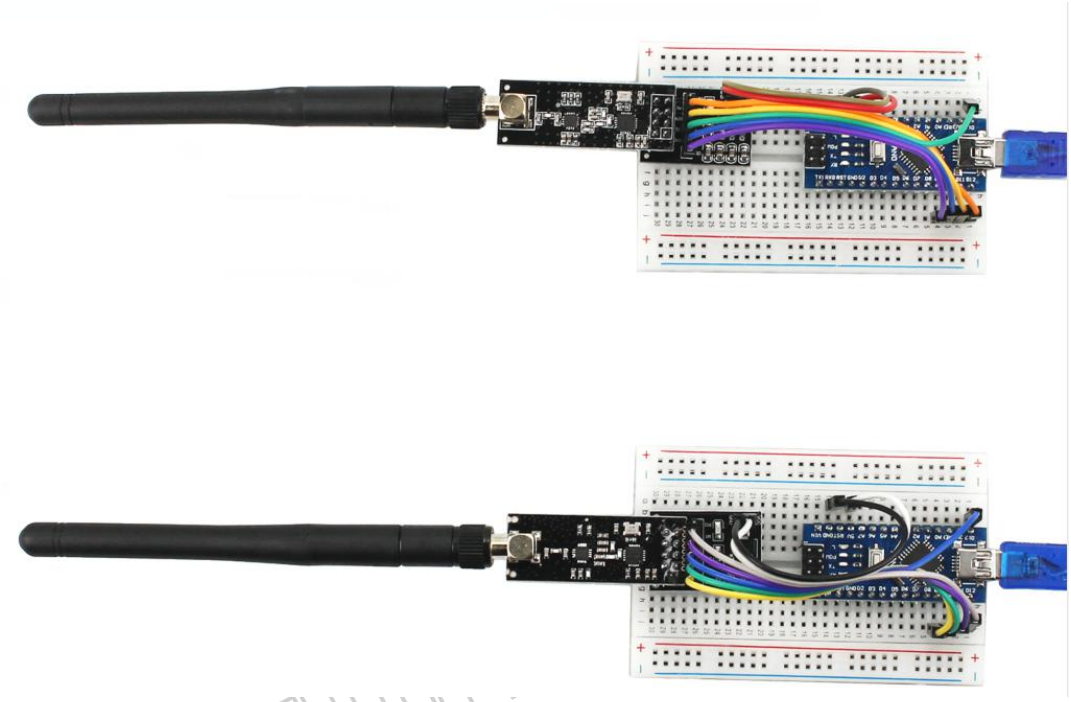
Pin Connection

First, the NRF24L01 wireless transceiver module needs to be connected to the NRF24L01 voltage regulator module. After the connection is completed, the pin connections of the NRF24L01 regulator module to the arduino nano as shown below:

NRF24L01 Regulator <----------> Arduino Nano

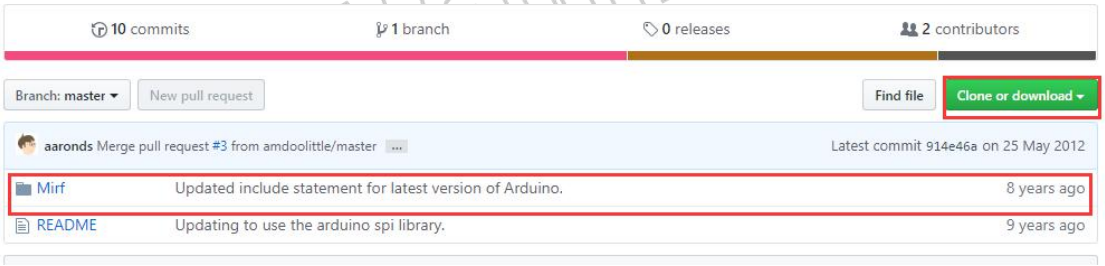
VCC <--------> 5V
GND <--------> GND
CE <----------> D9
CSN <---------> D10
MOSI<--------> D11
MISO<--------> D12
SCK <---------> D13
IRQ <---------> is not connected

Two sets need to be connected because nRF24L01 belongs to the mutual transfer module. Each chip is both a transmitter and a receiver. So the test needs 2pcs nRF24L01+ module and 2pcs arduino nano. The connection diagram is as shown below:



How to setup the wireless communication system?

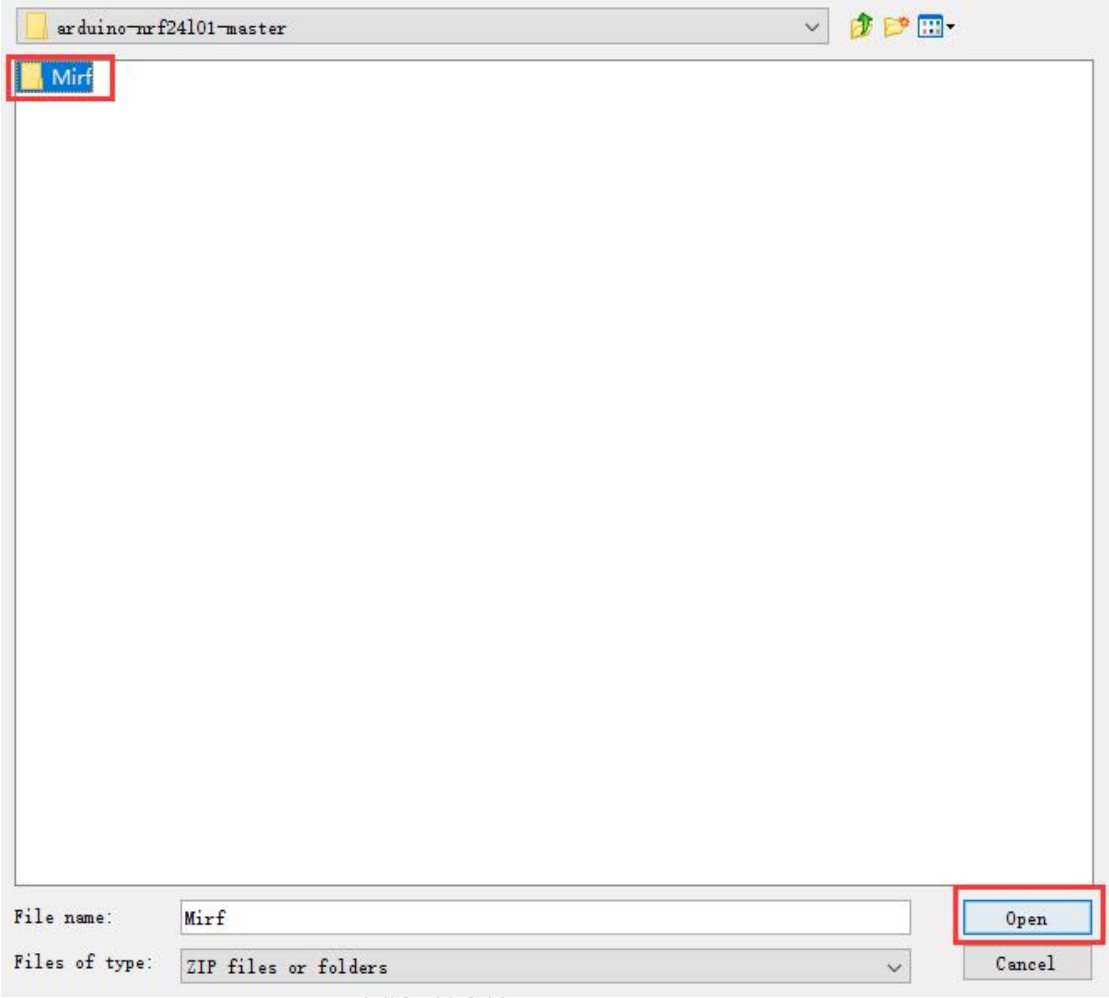
Step 1: You need to install the Mirf library before running the code, you can download it at https://github.com/aaronds/arduino-nrf24l01, as shown below:



Step 2: After the download is complete, unzip the folder and remember the way to extract it.

After the extraction is complete, open the arduino IDE and click on Sketch->Include Library->Add.ZIP Library, as shown below:

Step 3: In the pop-up window, find the file you just extracted, and select the Mirf file in the file, then click "Open" to complete the addition of the library file.



Step 4: For this tutorial we will have 2 separate codes: 1 for the Transmitter and one for the Receiver.

1) The code on the data transmitter is as shown below (the data sent by this tutorial is "123"):

```
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
void setup()
{
```

```
    Serial.begin(9600);

    Mirf.cePin = 9;                    //Set the CE pin to D9
    Mirf.csnPin = 10;          //Set the CE pin to D10
    Mirf.spi = &MirfHardwareSpi;
    Mirf.init();   //initialization nRF24L01

        //Set the receiving identifier "Sen01"
    Mirf.setRADDR((byte *)"Sen01");
    //Set the number of bytes sent and received at a time, here send an integer, write sizeof (unsigned int), actually equal to 2 bytes
    Mirf.payload = sizeof(unsigned int);
    //Sending channel, can fill 0~128, send and receive must be consistent. Mirf.channel = 3;
    Mirf.config();

        //Note that one Arduino writes Sender.ino and the other writes Receiver.ino.
        //The identifier here is written to Sender.ino
    Serial.println("I'm Sender...");
}
unsigned int adata = 0;
void loop()
{
    adata=123;
    //Since nRF24L01 can only send Mirf.payload data in a byte single byte array,
    //So all the data that needs to be transferred must be split into bytes.
    //Define a byte array to store pending data, because Mirf.payload = sizeof (unsigned int);
    //Actually the following is equal to byte data[2];
    byte data[Mirf.payload];

    //adata is unsigned int double-byte data and must be split.
    //Split the adata high and low eight:
    data[0] = adata & 0xFF;                    //low eight bits to data[0],
    data[1] = adata >> 8;                    //high eight bits to data[1]。

    //Settings send data to "serv1"
    Mirf.setTADDR((byte *)"Rec01");
    Mirf.send(data);
    //The next step can only be performed after the while loop function transmission is completed.
    while(Mirf.isSending()) {}
```

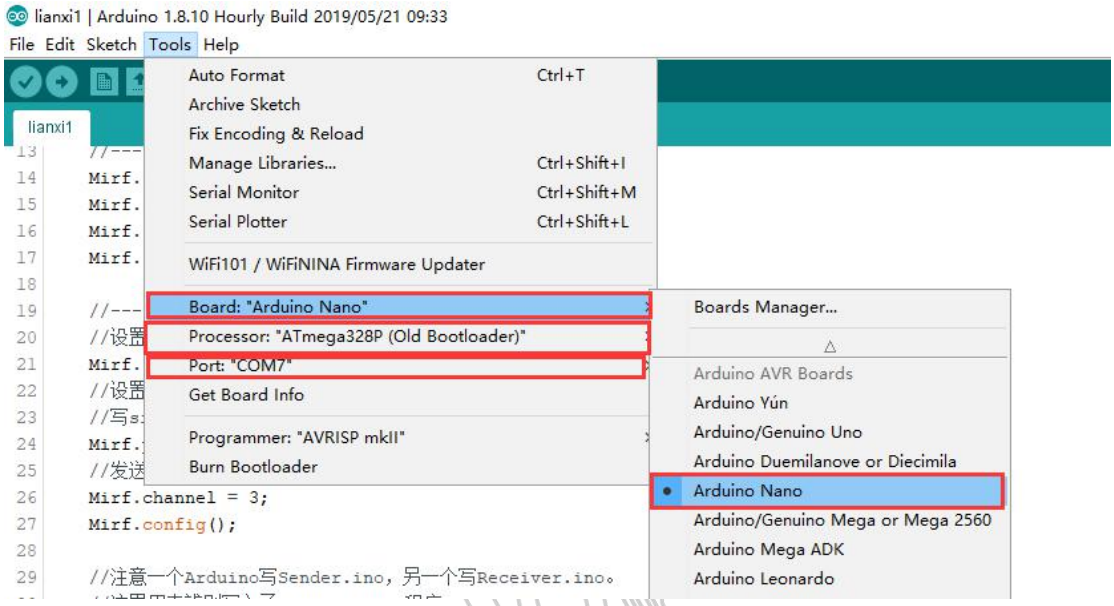
```
    delay(20);
}
```

2) The code on the data receiver is as shown below (the data received by this tutorial is "123")

```
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
//Define a variable adata to store the final result.
unsigned int adata = 0;

void setup()
{
    Serial.begin(9600);

    //---------Initial part, can't be modified at any time---------
    Mirf.cePin = 9;         //Set CE Pin to D9
    Mirf.csnPin = 10;       //Set CE Pin to D10
    Mirf.spi = &MirfHardwareSpi;
    Mirf.init();    //initialization nRF24L01

    //---------Configuration part, can be modified it at any time---------
    //Set the receiving identifier "Rev01"
    Mirf.setRADDR((byte *)"Rec01");
    //Set the number of bytes sent and received at a time, here sent an integer.
    //Write sizeof(unsigned int), which is actually equal to 2 bytes.
    Mirf.payload = sizeof(unsigned int);
    // Sending channel, can fill 0~128, send and receive must be consistent.
    Mirf.channel = 3;
    Mirf.config();

    //Note that one Arduino writes Sender.ino and the other writes Receiver.ino.
    //To identify the program written in Receiver.ino.
    Serial.println("I'm Receiver...");
}

void loop()
{
    //Define a scratchpad array with a size of Mirf.payload.
```

```
        byte data[Mirf.payload];
        if(Mirf.dataReady())        //Waiting the prepared receive data.
        {
            Mirf.getData(data);        //Receive data to data array.
            //data[1]< move left 8 bits and data[0] merge, reorganize data.
            adata = (unsigned int)((data[1] << 8) | data[0]);

                Serial.print("pin=");
                Serial.println(adata);
                //Can also output double-byte data.
                //Serial.write(data[1]);
                //Serial.write(data[0]);

        }
}
```

Step 5: Create two new files in the arduinoIDE, copy the two parts of the code into two files, and download them to the two arduino nano main control boards. The specific settings of ArduinoIDE are as shown below: After the setting is completed, you can download to the main control board.



After the download is completed, power the system at the transmitting end, and plug the system at the receiving end onto the computer, and then open the serial monitor of the arduinoIDE. Set the baud rate to 9600. Click on the data output area to see the received data, as shown below: