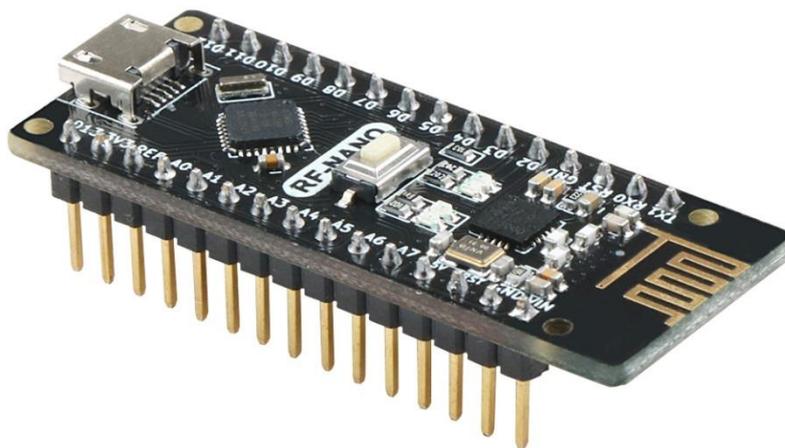


RF-NANO

Operating Instruction

V.1.1



Revision history

Date	Version	Description	Author
2019-7-25	V. 1. 0	Create a document	Abbott.Chen
2019-7-25	V. 1. 1	Modify the pictures	Abbott.Chen

Table of contents

Chapter 1 RF-NANO Introduce.....	4
1.1 RF-NANO overview.....	4
1.2 RF-NANO MCU introduce.....	6
1.2.1 Power Supply.....	7
1.2.2 Memory.....	7
1.2.3 Input and Output.....	7
1.2.4 Communication Interface.....	7
1.2.5 ATmega328 with NRF24L01+ of communication.....	8
1.2.6 Downloader.....	9
1.2.7 Attention.....	9
1.3 RF-NANO Driver installation.....	9
1.3.1 Installation IDE.....	9
1.3.2 Driver installation.....	12
Chapter 2 working principle of rf-nano.....	16
2.1 working principle introduction of rf-nano.....	16
2.2 Config word.....	17
2.3 RF-NANO work module.....	18
Chapter3 realizes the communication between rf-nano.....	19
3.1 Implement two rf-nano point-to-point communication.....	19
3.1.1 connect method.....	19
3.1.2 Principle of application.....	20
3.1.3 Program code.....	21
3.2 Implement multiple send to a receive communication.....	24
3.2.1 Experimental principle block diagram.....	24
3.2.2 program code.....	25
3.3 Implement a send to multiple receive communication.....	28
3.3.1 Experimental principle block diagram.....	28
3.3.2 program code.....	28
Chapter 4 Sets the transmitting power and data transmission rate.....	33
4.1 Set the transmitted power of RF-NANO.....	33
4.2 Set RF-NANO data transfer rate.....	33

1.1 RF-NANO overview

The board of rf-nano is integrated into NRF24L01+ chip, making it have unlimited transceiver function, which is equivalent to combining an ordinary NANO board and an NRF24L01+ module into one, as shown in figure 1-1-1. It is more convenient to use, and the small size rf-nano is exactly the same as the common pin of NANO board, convenient for transplantation.

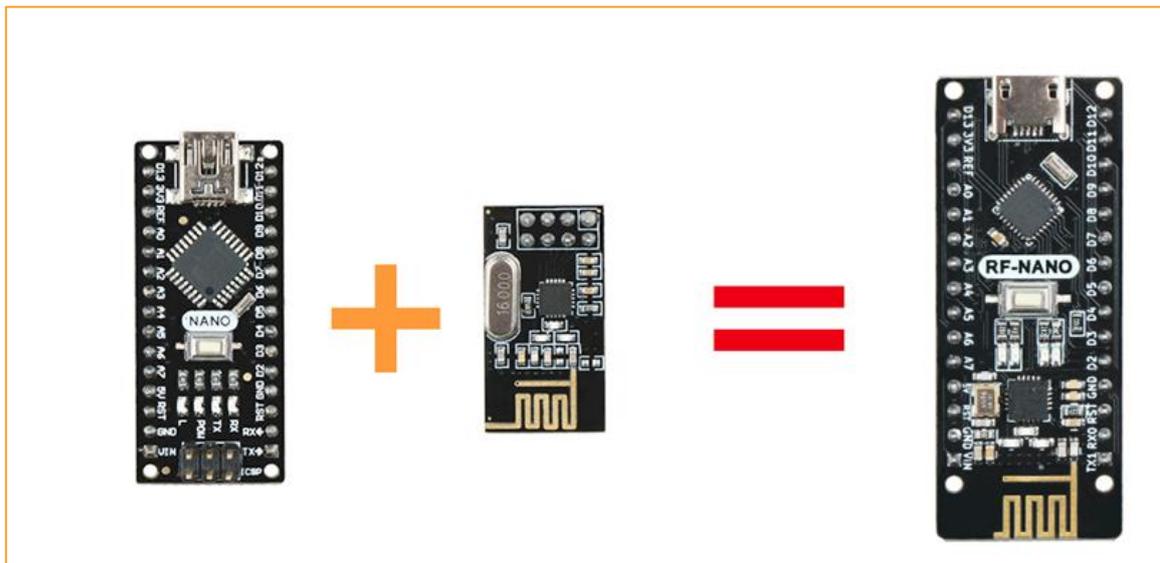


Figure 1-1-1

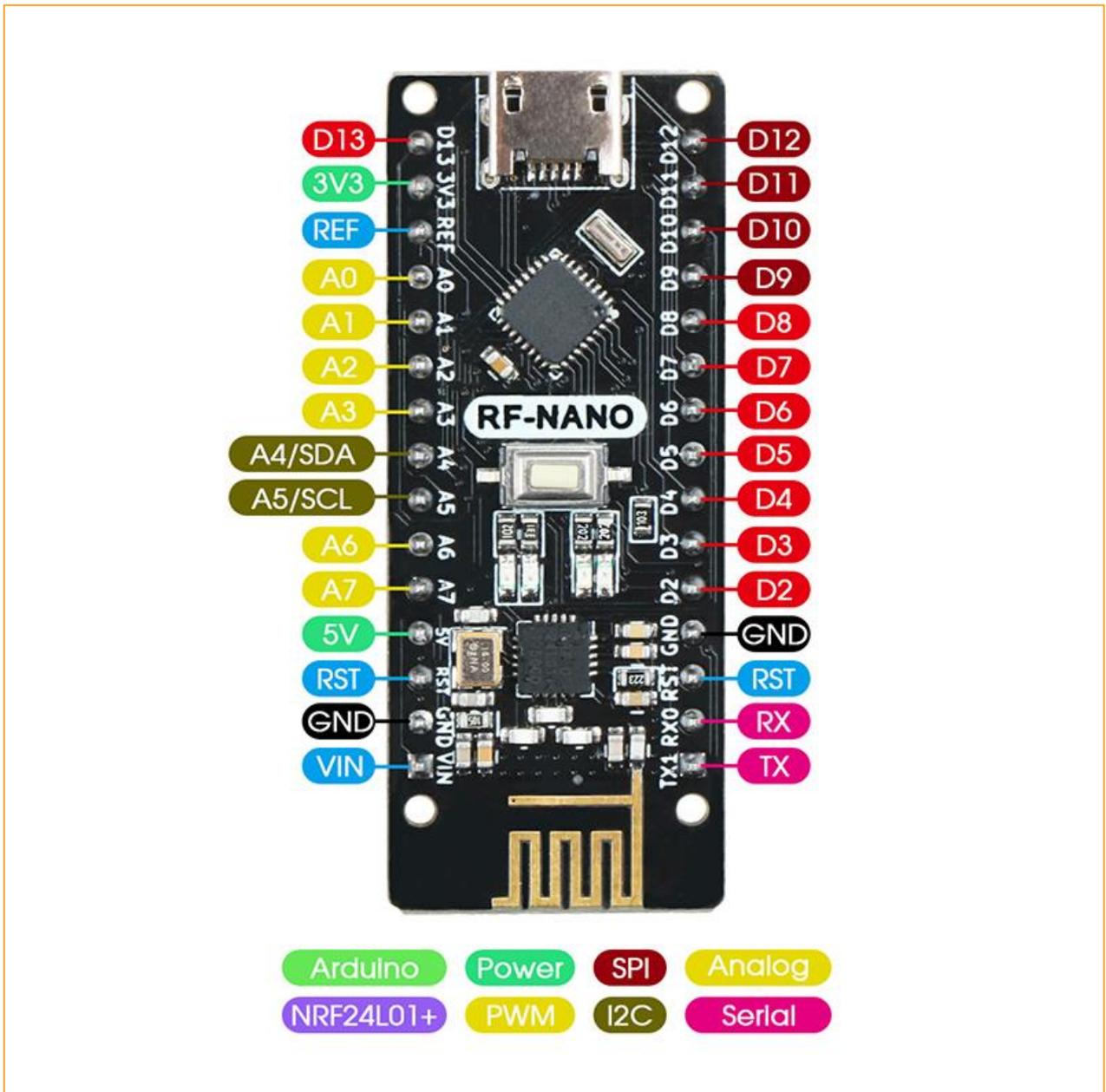


Figure 1-1-2

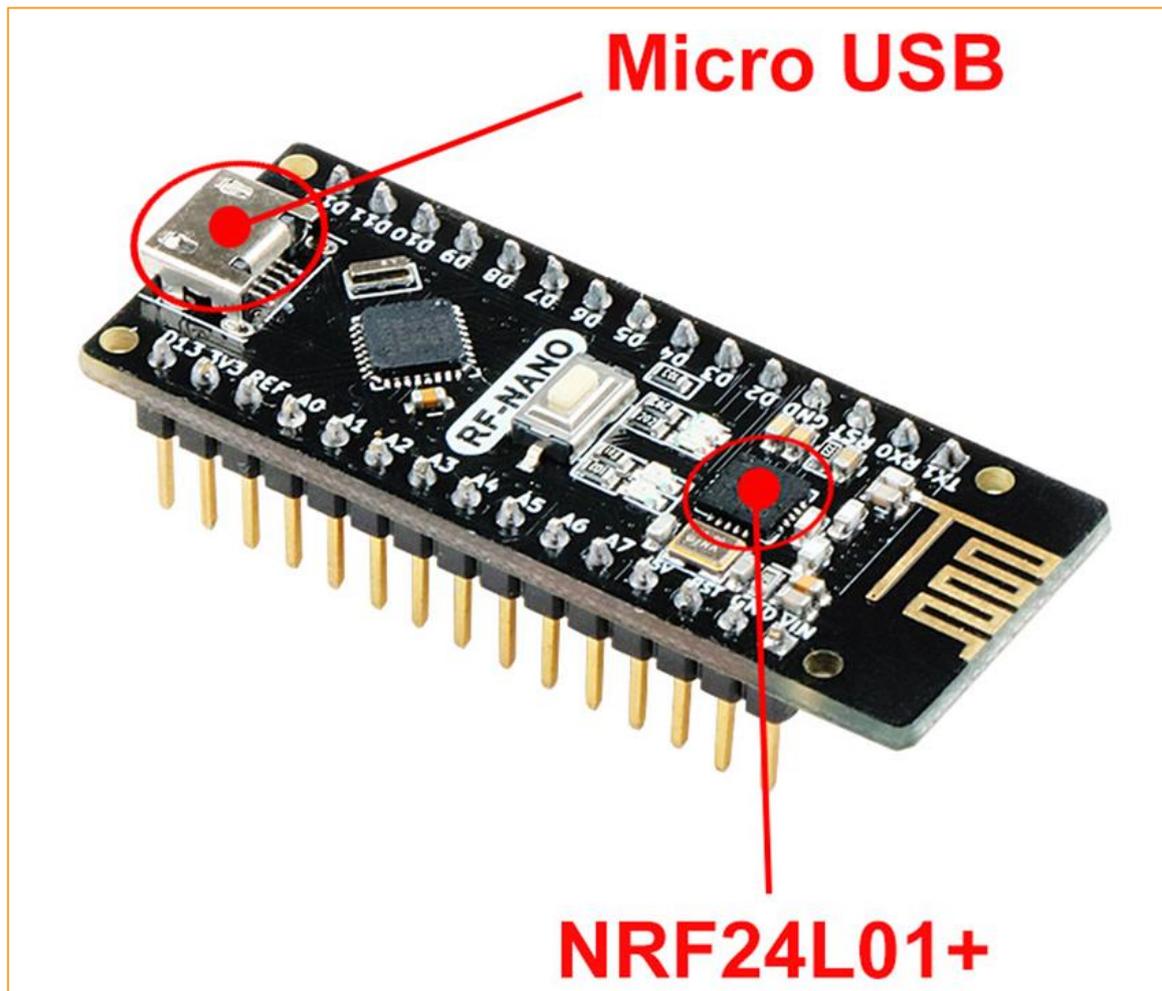


Figure 1-1-3

1.2 RF-NANO MCU introduce

Arduino rf-nano MCU is ATmega328(Nano3.0), usb-micro interface, with 14 channels of digital input/output (6 channels can be used as PWM output), 8 channels of analog input, a 16MHz crystal oscillator, a usb-micro port, an ICSP header and a reset button.

- ◆ The processor ATmega328
- ◆ Working voltage 5v
- ◆ Input voltage (recommended) 7-12v
- ◆ Input voltage (range) 6-20v
- ◆ Digital IO pin 14 (6 of which can be used as PWM output)
- ◆ Analog input pin 6
- ◆ IO pin DC 40 mA
- ◆ Flash Memory 16 or 32 KB (in which 2 KB for bootloader)
- ◆ SRAM 1KB or 2KB

- ◆ EEPROM :1KB (ATmega328)
- ◆ CH340 USB to serial port chip
- ◆ Clock 16 MHz

1.2.1 Power Supply

Arduino Nano power supply mode: mini-B USB interface power supply and external vin connection
7~12V external DC power supply

1.2.2 Memory

ATmega328 includes on-chip 32KB Flash, of which 2KB is used for Bootloader. There are also 2KB SRAM and 1KB EEPROM.

1.2.3 Input and Output

- ◆ 14 digital input and output ports: The working voltage is 5V, and each channel can output and access the maximum current of 40mA. Each channel is equipped with a 20-50K ohm internal pull-up resistance (not connected by default). In addition, some pins have specific functions.
- ◆ Serial signal RX (0), TX (1): It provides serial port receiving signal with TTL voltage level, connected to the corresponding pin of FT232R1.
- ◆ External Interrupt (No. 2 and No. 3): Trigger interrupt pin, which can be set to rising edge, falling edge or simultaneous trigger.
- ◆ Pulse Width Modulation PWM (3, 5, 6, 9, 10, 11): Provides 6 channel, 8-bit PWM outputs.
- ◆ SPI (10(SS), 11(MOSI), 12(MISO), 13(SCK)):SPI Communication Interface.
- ◆ LED (No. 13): Arduino is specially used to test the reserved interface of the LED. When the output is high, the LED is lit. When the output is low, the LED is off.
- ◆ 6 analog inputs A0 to A5: Each channel has a resolution of 10 bits (that is, the input has 1024 different values), the default input signal range is 0 to 5V, and the input upper limit can be adjusted by AREF. In addition, some pins have specific functions.
- ◆ TWI interface (SDA A4 and SCL A5): Supports communication interface (compatible with I2C bus).
- ◆ AREF: The reference voltage of the analog input signal.
- ◆ Reset: The microcontroller chip is reset when the signal is low.

1.2.4 Communication Interface

Serial port: The built-in UART of ATmega328 can communicate with external serial port through digital port 0 (RX) and 1 (TX).

1.2.5 ATmega328 with NRF24L01+ of communication

ATmega328 and NRF24L01+ are SPI communication, and the schematic diagram is shown in figure 1-2-1;

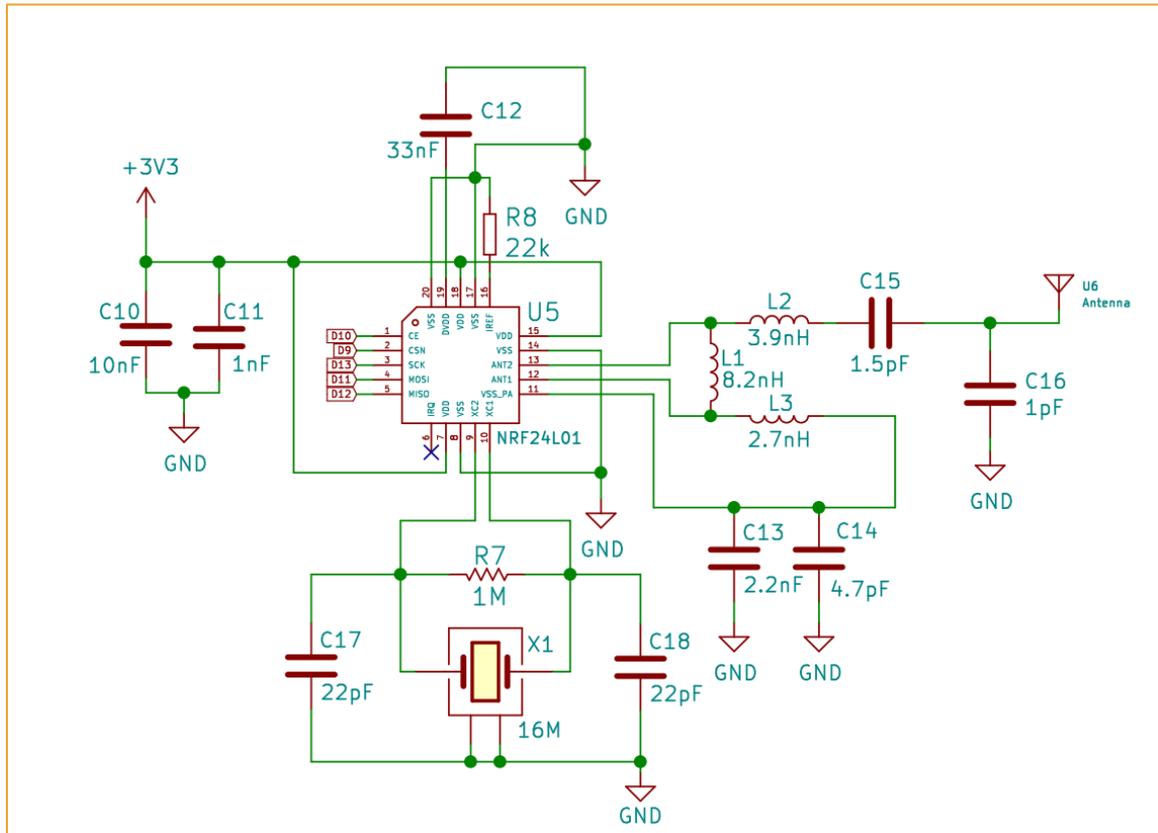


Figure 1-2-1

ATmega328 and NRF24L01+ chip pin connection:

ATmega328	NRF24L01+
+3.3V	VCC
GND	GND
D9	CSN
D10	CE
D11	MOSI
D12	MISO
D13	SCK

Note :ATmega328 already occupied pins D9,D10,D11,D12, and D13 cannot be reused

1.2.6 Downloader

The MCU on the Arduino Nano has a bootloader program, so you can download the program directly from the Arduino software. You can also directly download the program to the MCU through the ICSP header on the Nano.

1.2.7 Attention

The Arduino Nano provides an automatic reset design that can be reset by the host. In this way, the software can be automatically reset by the Arduino software in the program to the Nano, without pressing the reset button.

1.3 RF-NANO Driver installation

1.3.1 Installation IDE

ArduinoIDE is an open source software and hardware tool written by open source software such as Java, Processing, and avr-gcc. It is an integrated development environment that runs on a computer. It can write and transfer programs to the board. The major feature of the IDE is cross-platform compatibility for Windows, MacOSX, and Linux. Only a simple code base is needed, and the creators can create personalized home internet solutions through the platform, such as remote home monitoring and constant temperature control and so on.

In this tutorial, we use the version is 1.8.9, download address is :

<https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>, After opening the link, we can see the interface as shown in Figure 1-3-1. In this interface, we can see the different versions of the IDE and different operating environments. Everyone can download according to their own computer system, of course, There will be a downloaded installation package on our companion CD, but only the Windows version, because this tutorial is all running under Windows system.

1.8.1	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.0	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.13	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.12	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.11	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.10	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.9	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.8	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.7	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github

Figure 1-3-1 ArduinoIDE download interface

After the downloading, we will get a compressed package as shown in Figure 1-3-2. The compressed package will be decompressed. After decompression, the files in Figure 2.1.3 are extracted. The “drivers” is the driver software. When the “Arduino.exe” is installed, it will be Install the driver automatically. Because the installation of "arduino.exe" is very simple, it will not be explained here. It is recommended to exit the anti-virus software during the installation process, otherwise it may affect the installation of the IDE. After the installation is complete, click "arduino.exe" again to enter the IDE programming interface.

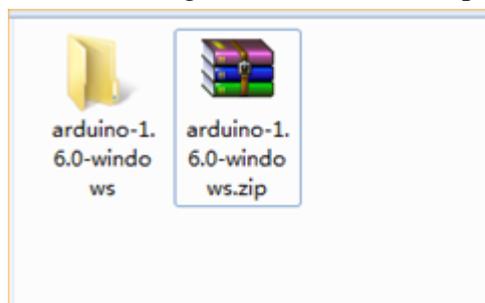


Figure 1-3-2 ArduinoIDE installation package

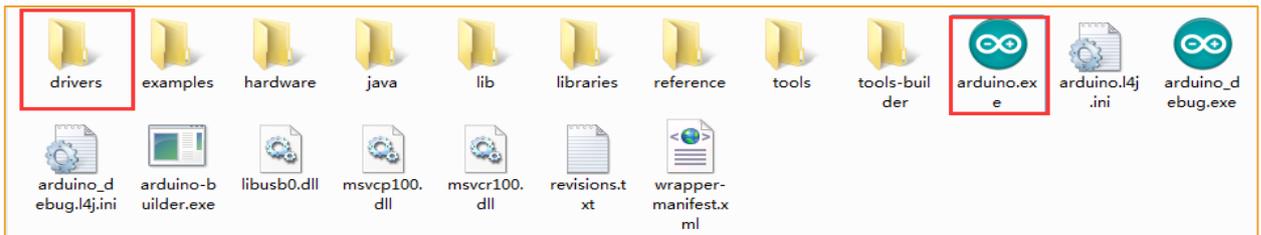


Figure 1-3-3 Extracted files

Arduino IDE After the installation, we hook up the Arduino motherboard and right-click on my computer (properties (device manager (look at the ports (COM and LTP), as shown in figure 1-3-4

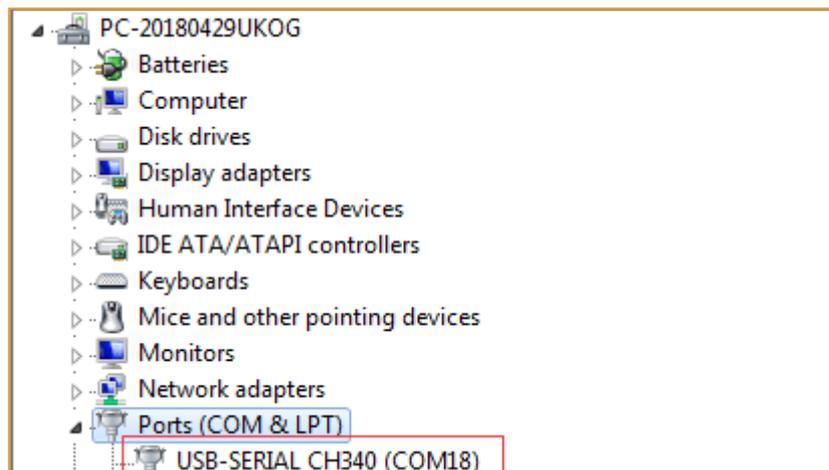


Figure 1-3-4

Then we open the IDE, select the corresponding development board model and port in the toolbar and it will work normally. If it appears as shown in figure 1-3-5, it shows that the computer does not recognize the development board and needs to install the driver itself

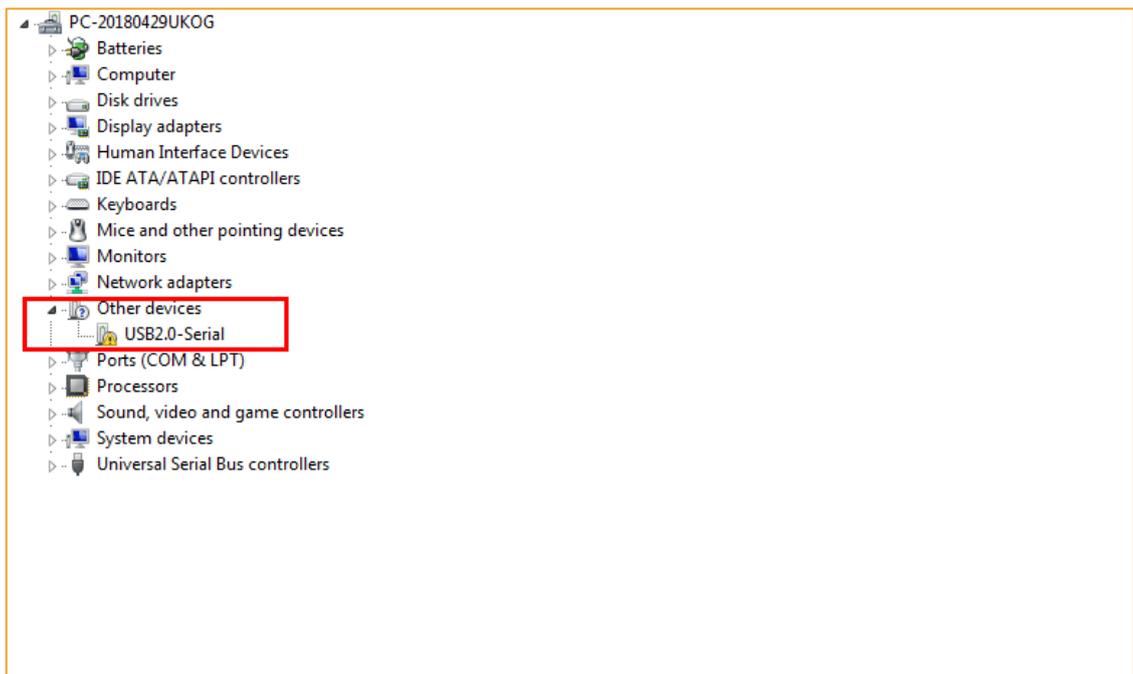


Figure 1-3-5

1.3.2 Driver installation

Windows7 System driver installation steps

1) Right-click "my computer" (open device manager (look at ports (COM and LPT)).At this point you will see a "USB serial port", right-click "USB serial port" and select the "update driver software" option.

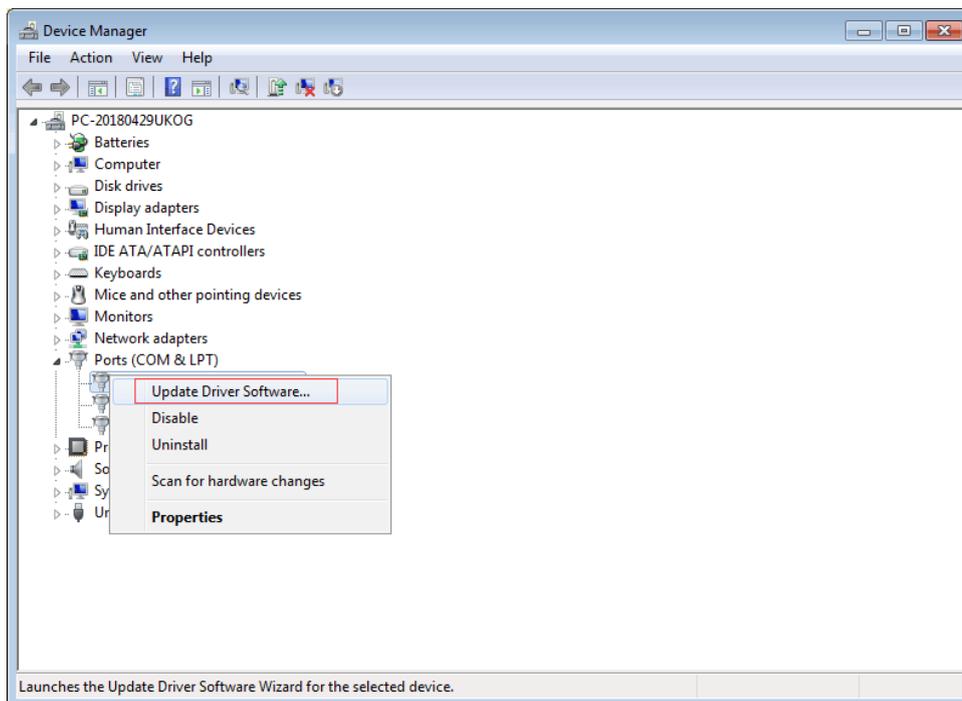


Figure 1-3-6

2) Next, select the browse my computer for driver software options

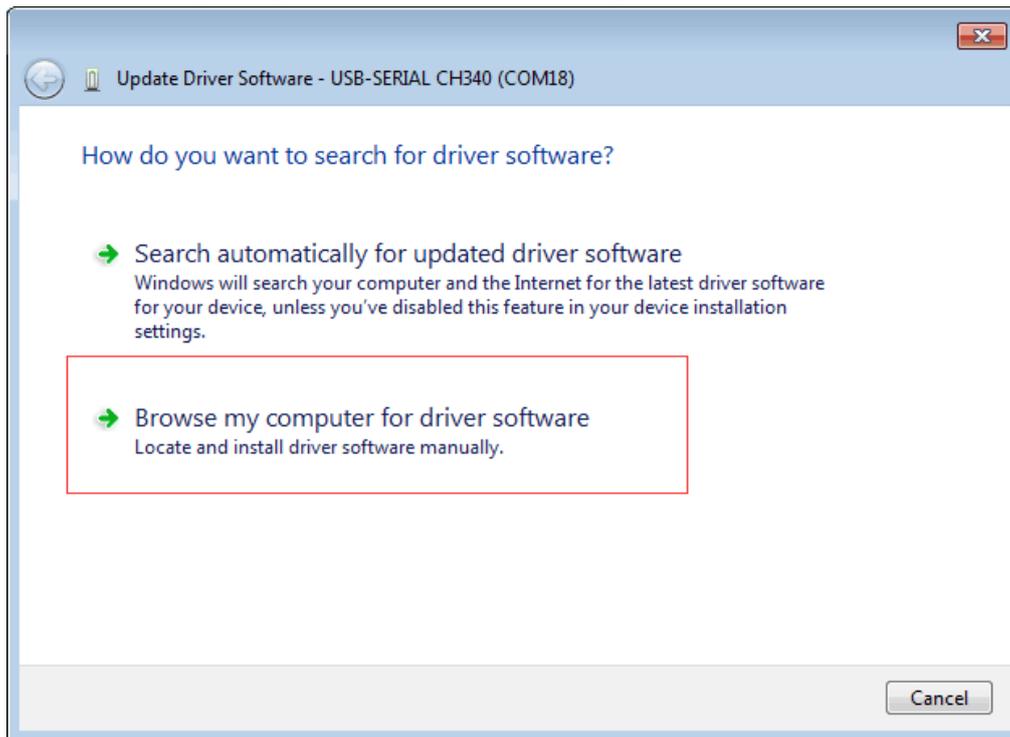


Figure 1-3-7

3) Finally, select the driver file named CH341SER_for_64bit_win7, which is located in the Arduino_Nano board driver folder. **Please select the corresponding driver version according to your computer system model**

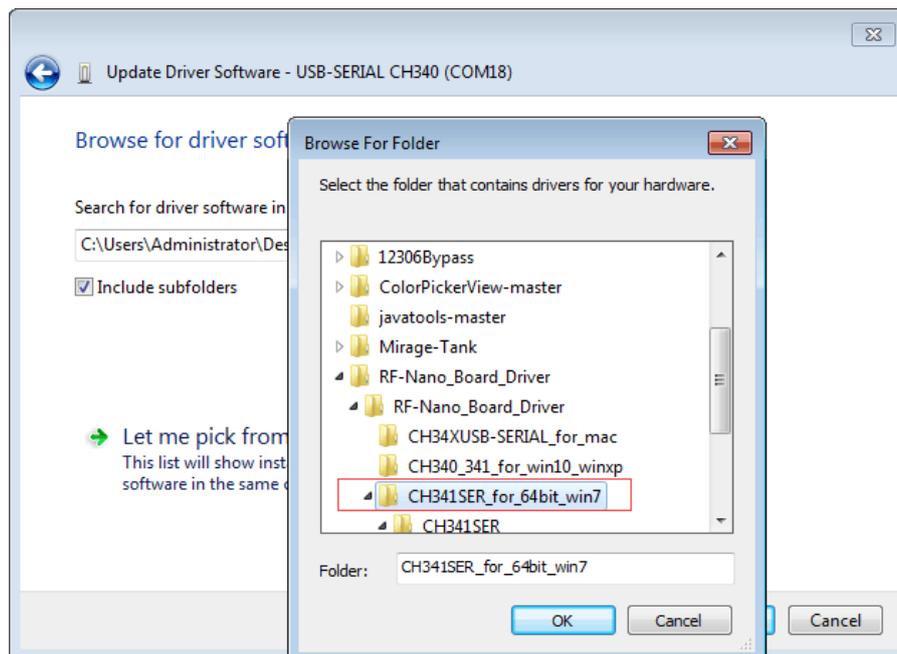


Figure 1-3-8

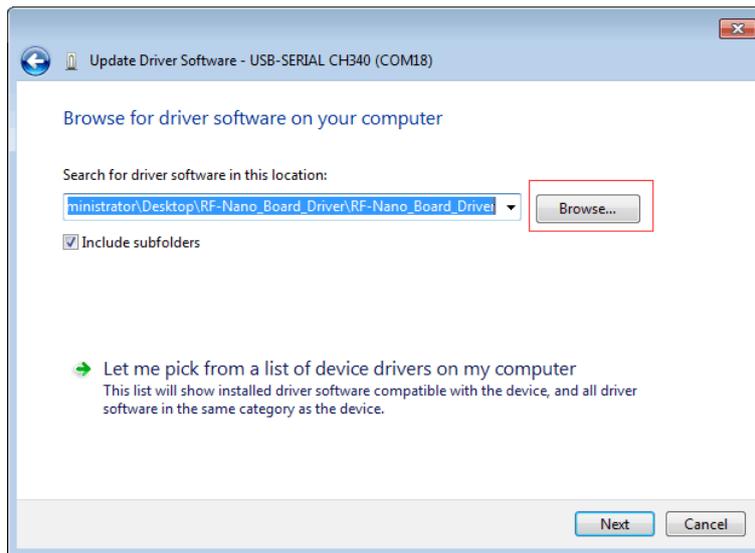


Figure 1-3-9

4) After successful installation, the following screen will appear to inform you that the driver is successful.

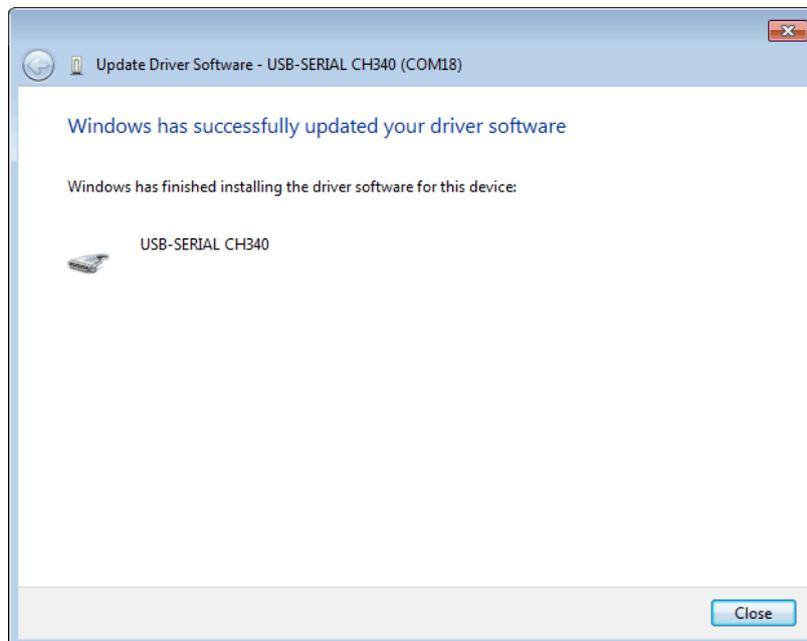


Figure 1-3-10

At this point, we can go back to the device manager interface and see that the computer has successfully recognized Arduino. As shown in figure 1-3-11 below, open the Arduino compilation environment and start the Arduino journey

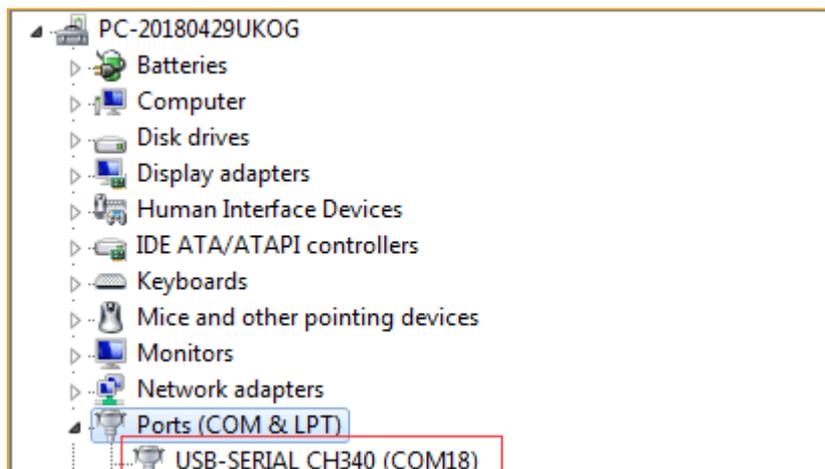


Figure 1-3-11

Note: in the Win10 system, part of the Arduino is in access to the computer (not the original chips are difficult to identify), after the system will automatically download the corresponding driver, don't need to install the driver himself, but in the Windows 7 system, you need to follow the steps above to manually install the driver as shown in the above we can see the USB serial port is identified as the COM15, but different computers may not be the same, you may be the COM4 COM5 etc., but the Arduino Nano this must be the same if you didn't find the USB Serial port, may be you install the wrong, or the system is not compatible

Windows8 System driver installation steps

If your computer is Windows8: before installing the driver, you should save the files you are editing because there will be several shutdowns during operation

- 1) press the Windows key + R
- 2) type shutdown. 00 exe/f/t/R/o
- 3) click ok button
- 4) the system will restart to choose an option screen
- 5) pick an option from the screen, select the troubleshooting
- 6) from the troubleshooting screen select advanced options to select Windows startup Settings from the advanced options screen
- 7) click the restart button
- 8) system to restart to
- 9) to select advanced startup options screen Disable driver signature enforcement
- 10) once the system starts, you can install the Arduino driver as with Windows7

If your computer is WindowsXP: the installation steps are basically the same as for Windows7, please refer to the above Windows7 installation steps.

Chapter 2 working principle of rf-nano

2.1 working principle introduction of rf-nano

The rf-nano can transmit and receive data.

When transmitting data: first set NRF24L01+ as transmission mode: then put the receiving node address TX_ADDR and effective data TX_PLD into NRF24L01+ cache by SPI port according to the time sequence. If the automatic reply is turned on, NRF24L01+ will enter the receiving mode immediately after transmitting data and receive the answering signal (the receiving address of the automatic reply should be the same as the receiving node address TX_ADDR). If receive reply, consider the communication success, TX_DS set high, at the same time TX_PLD from TX FIFO clear; If no reply is received, the data will be automatically retransmitted (automatic retransmission is enabled). If the number of times of retransmission (ARC) reaches the upper limit, MAX_RT will be set to a higher level, and the data in TX FIFO will be retained for subsequent retransmission. When MAX_RT or TX_DS are set high, the IRQ becomes low and interrupts, notifies MCU. When the launch is successful, if the CE is low, NRF24L01+ enters idle mode 1. If there is data in the send stack and CE is high, it will enter the next launch. If no data is sent on the stack and the CE is high, then idle mode 2 is entered.

When receiving data: NRF24L01+ is configured to receive mode first, followed by a delay of 130 microns before it enters the receiving state and waits for the data to arrive. When the receiver detects a valid address and CRC, the packet will be stored in RX FIFO. At the same time, the interrupt flag bit RX_DR is set to high, IRQ becomes low, generating an interrupt, and MCU is informed to fetch the data. If the automatic reply is on at this time, the receiver will enter the transmitting state to return the reply signal at the same time. When receiving successfully, if CE decreases, NRF24L01+ enters idle mode 1. Always go into standby or power down mode before writing registers. As shown in figure 2-1-1, SPI operation sequence diagram is given:

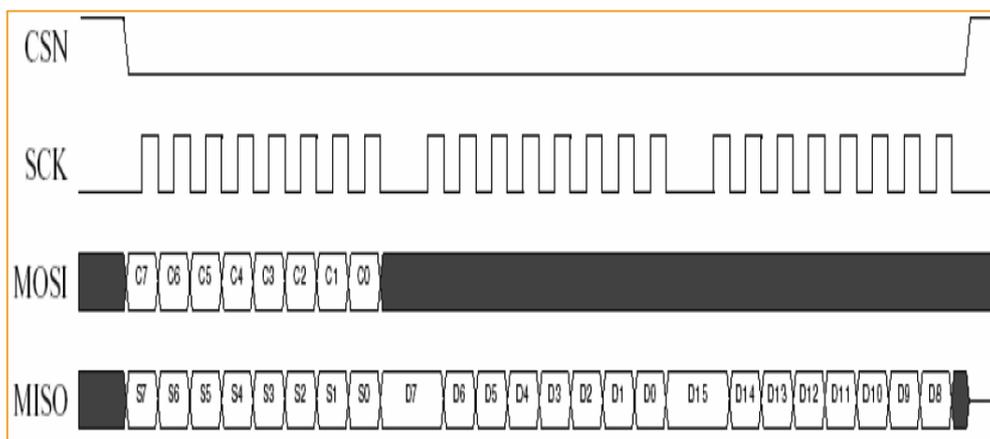


Figure 2-1-1 SPI read data

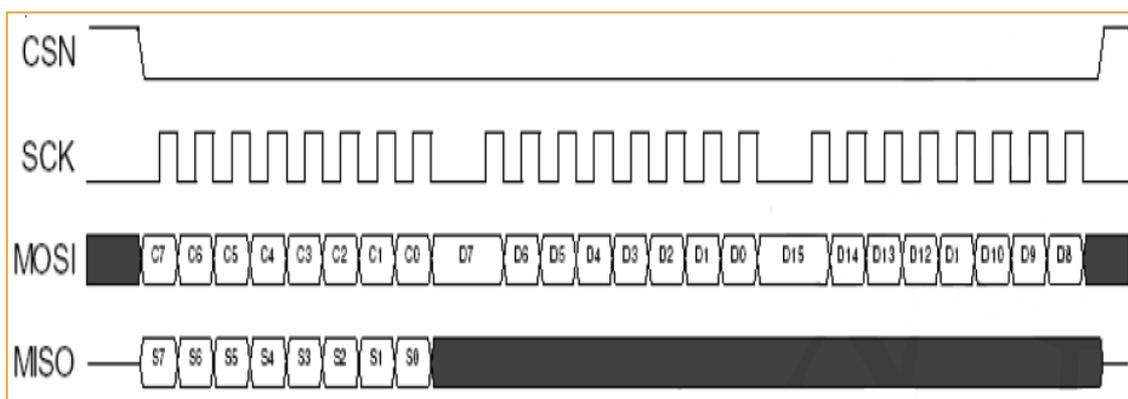


Figure 2-1-1 SPI write data

2.2 Config word

SPI port for synchronous serial communication interface, the maximum transfer rate of 10 Mb/s, when the transmission transfer low byte first, and then transmit high byte but for a single byte character, to send high to send low SPI related instructions with a total of eight, when using the control instruction by NRF24L01 + MOSI input from MISO output corresponding status and data information to the MCU nRF24L01 + are defined by the configuration register all the configuration of the word, the configuration register can be accessed through the SPI mouth NRF24L01 +0f the 25 configuration registers, the most common configuration registers are shown in table 2

Table 2: common configuration registers

Addresses (H)	Register name	Function

00	CONFIG	Set NRF24L01+ module
01	EN_AA	Set up the receiving channel and automatic reply
02	EN_RXADDR	Enable receive channel address
03	SETUP_AW	Set address width
04	SETUP_RETR	Set the time and frequency of automatic retransmission of data
07	STATUS	Status register, used to determine the working status
0A~0F	RX_ADDR_P0~P5	Set the receive channel address
10	TX_ADDR	Set the receive channel address
11~16	RX_PW_P0~P5	Sets the valid data width of the receive channel

2.3 RF-NANO work module

NRF24L01+ there is a state machine inside the chip, which controls the conversion between different working modes of the chip. NRF24L01+ can be configured as **Shutdown**、**Standby**、**Idle-TX**、**TX** and **RX** Five working modes. The state transition diagram is shown in figure 2-3-1.

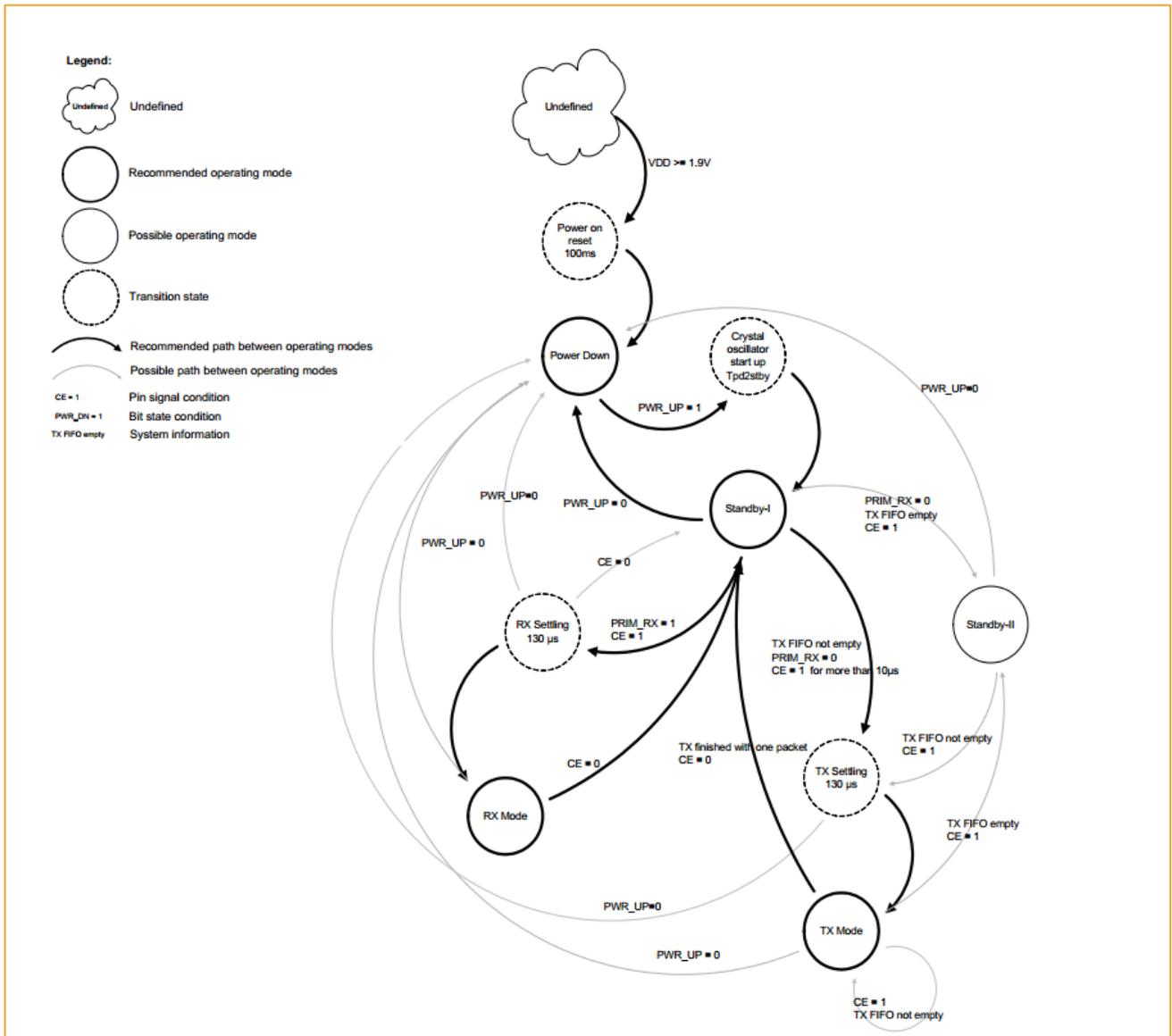


Figure 2-3-1

Chapter3 realizes the communication between rf-nano

3.1 Implement two rf-nano point-to-point communication

3.1.1 connect method

Prepare two rf-nano or an rf-nano and NRF24L01+ module, and an Arduino UNO R3(Arduino NANO V3.0), as shown in figure 3-1-1.



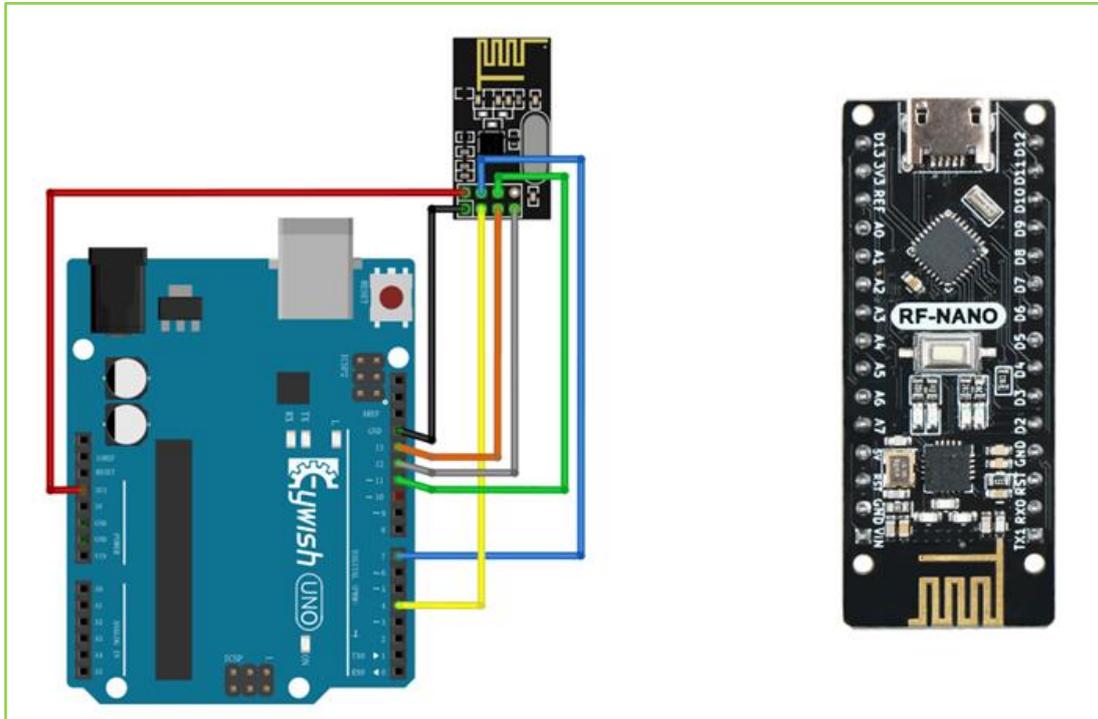


图 3-1-1

Arduino and NRF24L01+ wire way:

arduino Uno	NRF24L01+
+3.3V	VCC
GND	GND
7	CSN
4	CE
11pin	MOSI
12pin	MISO
13pin	SCK

3.1.2 Principle of application

Launch process

- 1、 Firstly, configure the nRF24L01 to transmit mode.
- 2、 Then write the address TX_ADDR of the receiving end and the data TX_PLD to be sent into the nRF24L01 buffer area by the SPI port in time sequence.

- 3、 Arduino configures the CE to be high for at least 10 μ s and transmits data after a delay of 130 μ s. If the auto answer is on, the nRF24L01 enters the receive mode immediately after transmitting the data and receives the answer signal. If a reply is received, the communication is considered successful.
- 4、 NRF24L01 will automatically set TX_DS high and the TX_PLD will be cleared from the transmit stack. If no response is received, the data will be automatically retransmitted. If the number of retransmissions (ARC_CNT) reaches the upper limit, MAX_RT is set high and TX_PLD will not be cleared; MAX_RT When TX_DS is set high, IRQ goes low to trigger MCU interrupt. When the last transmission is successful, if the CE is low, the nRF24L01 will enter standby mode.
- 5、 If there is data in the transmission stack and CE is high, the next transmission is started; if there is no data in the transmission stack and CE is high, the nRF24L01 will enter standby mode 2.

Receive data process

- 1、 When the nRF24L01 receives data, please configure the nRF24L01 to receive mode firstly.
- 2、 Then delay 130 μ s into the receiving state to wait for the arrival of data. When the receiver detects a valid address and CRC, it stores the data packet in the receiver stack. At the same time, the interrupt sign RX_DR is set high and the IRQ goes low to notify the MCU to fetch the data.
- 3、 If auto-response is turned on at this time, the receiver will enter the transmit status echo response signal at the same time. When the last reception is successful, if CE goes low, the nRF24L01 will go into idle mode 1.

3.1.3 Program code

Transmission data program code::

- **Code path:** RF-NANO Demo Program\Peer-to-peer Communication\Emitter\Emitter.ino
- **Code source**

```
//Transmitter program

#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
Nrf24l Mirf = Nrf24l(10, 9);
byte value;

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

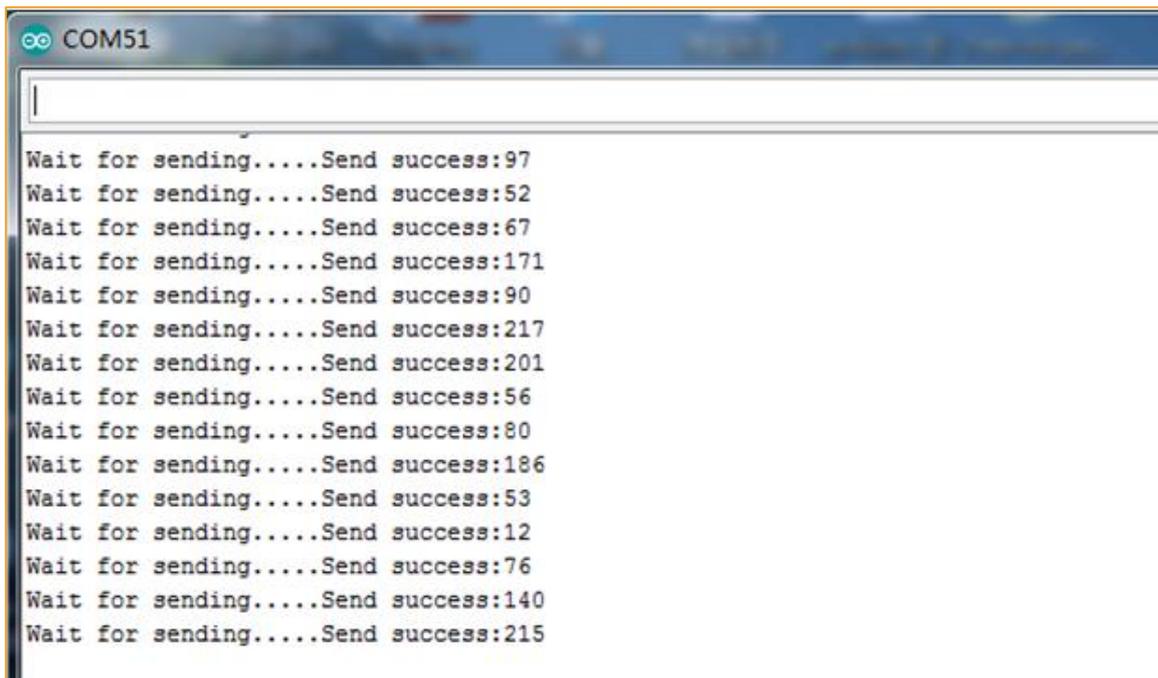
```

Mirf.spi = &MirfHardwareSpi;
Mirf.init();
//Set your own address (sender address) using 5 characters
Mirf.setRADDR((byte *)"ABCDE");
Mirf.payload = sizeof(value);
Mirf.channel = 90;           //Set the channel used
Mirf.config();
}

void loop()
{
  Mirf.setTADDR((byte *)"FGHIJ");           //Set the receiver address
  value = random(255);                       //0-255 random number
  Mirf.send(&value);                         //Send instructions, send random number value
  Serial.print("Wait for sending.....");
  while (Mirf.isSending())//Until you send successfully, exit the loop
  delay(1);
  Serial.print("Send success:");
  Serial.println(value);
  delay(1000);
}

```

Transmitting data from the transmitter:



```

COM51
Wait for sending.....Send success:97
Wait for sending.....Send success:52
Wait for sending.....Send success:67
Wait for sending.....Send success:171
Wait for sending.....Send success:90
Wait for sending.....Send success:217
Wait for sending.....Send success:201
Wait for sending.....Send success:56
Wait for sending.....Send success:80
Wait for sending.....Send success:186
Wait for sending.....Send success:53
Wait for sending.....Send success:12
Wait for sending.....Send success:76
Wait for sending.....Send success:140
Wait for sending.....Send success:215

```

Figure 3-1-2

Data receiving program code:

- **Code path :** RF-NANO Demo Program\Peer-to-peer Communication\Receive\ Receive.ino
- **Source code:**

```
//Receiver program
#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
Nrf24l Mirf = Nrf24l(10, 9);
byte value;
void setup()
{
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"FGHIJ"); //Set your own address (receiver address) using
5 characters
  Mirf.payload = sizeof(value);
  Mirf.channel = 90;           //Set the used channel
  Mirf.config();
  Serial.println("Listening..."); //Start listening to received data
}

void loop()
{
  if (Mirf.dataReady()) { //When the program is received, the received data is output
from the serial port
    Mirf.getData(&value);
    Serial.print("Got data: ");
    Serial.println(value);
  }
}
```

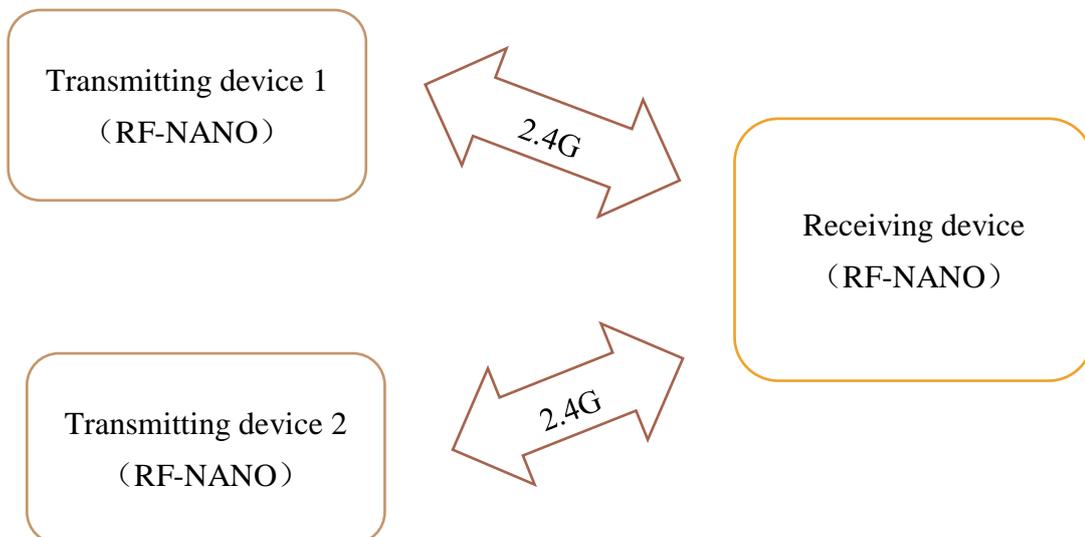
Data received by a receiving device

```
COM48  
  
Got MotorDrive data: 97  
Got MotorDrive data: 52  
Got MotorDrive data: 67  
Got MotorDrive data: 171  
Got MotorDrive data: 90  
Got MotorDrive data: 217  
Got MotorDrive data: 201  
Got MotorDrive data: 56  
Got MotorDrive data: 80  
Got MotorDrive data: 186  
Got MotorDrive data: 53  
Got MotorDrive data: 12  
Got MotorDrive data: 76  
Got MotorDrive data: 140  
Got MotorDrive data: 215
```

Figure 3-1-2

3.2 Implement multiple send to a receive communication

3.2.1 Experimental principle block diagram



3.2.2 program code

Transmitting data program 1 code:

- **Code path:** RF-NANO Demo Program\Multiple send to one receive communication\Emitter1\Emitter1.ino
- **Source code**

```
#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
Nrf24l Mirf = Nrf24l(10, 9);
int value;

void setup()
{
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  //Set your own address (sender address) using 5 characters
  Mirf.setRADDR((byte *)"ABCDE");
  Mirf.payload = sizeof(value);
  Mirf.channel = 10;           //Set the channel used
  Mirf.config();
}

void loop()
{
  Mirf.setTADDR((byte *)"FGHIJ");           //Set the receiver address
  value = 100;
  Mirf.send((byte *)&value);               //Send instructions, send random number value
  Serial.print("Wait for sending.....");
  while (Mirf.isSending()) delay(1);        //Until you send successfully, exit the loop
  Serial.print("Send success:");
  Serial.println(value);
  delay(1000);
}
```

Data sent by transmitter 1:

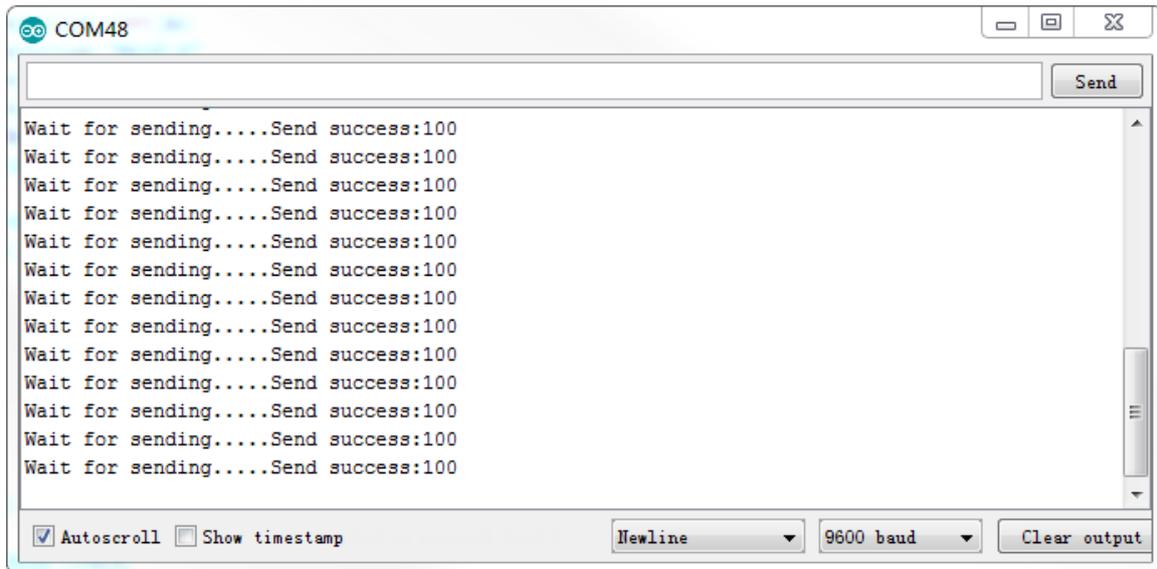


Figure 3-2-1

Launch data program 2 code:

- **Code path:** RF-NANO Demo Program\Multiple send to one receive communication\Emitter2\Emitter2.ino
- **Source code**

```
//Transmitter program

#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
Nrf24l Mirf = Nrf24l(10, 9);
int value;

void setup()
{
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  //Set your own address (sender address) using 5 characters
  Mirf.setRADDR((byte *)"ABCDE");
  Mirf.payload = sizeof(value);
  Mirf.channel = 20;           //Set the channel used
```

```

Mirf.config();
}

void loop()
{
  Mirf.setTADDR((byte *)"FGHIJ");           //Set the receiver address
  value = 200;                               //0-255 random number
  Mirf.send((byte *)&value);                //Send instructions, send random number value
  Serial.print("Wait for sending.....");
  while (Mirf.isSending()) delay(1);         //Until you send successfully, exit the loop
  Serial.print("Send success:");
  Serial.println(value);
  delay(1000);
}

```

Data sent by transmitter 2:

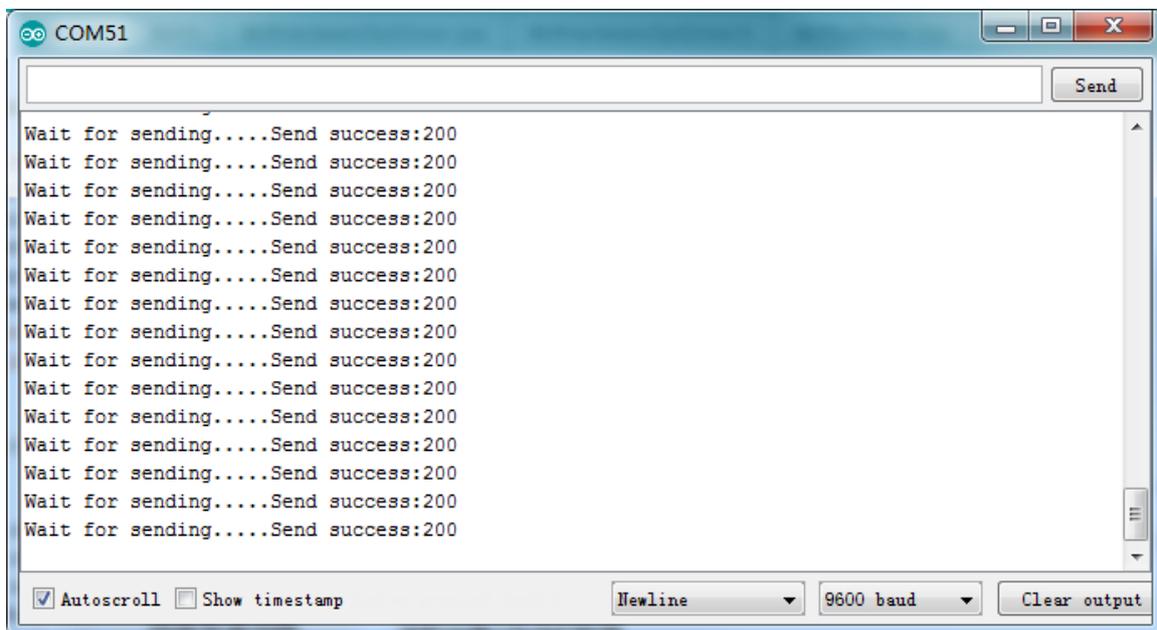


Figure 3-2-2

Receiving data program code:

- **Code path:** RF-NANO Demo Program\Multiple send to one receive communication
 \All_Receive\ All_Receive.ino
- **Source code**

Data received by a receiving device:

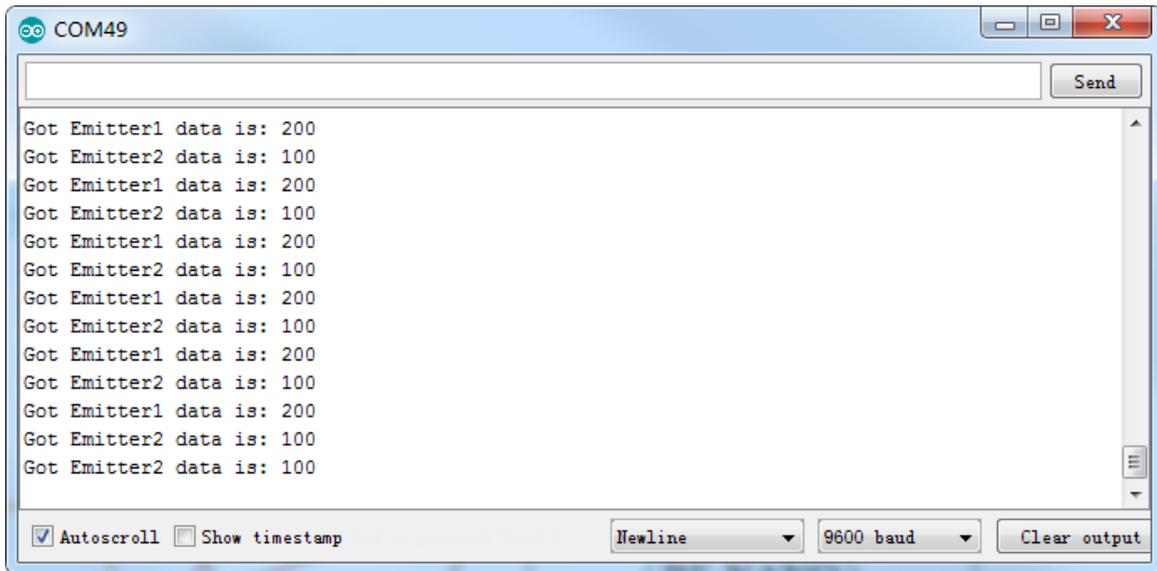
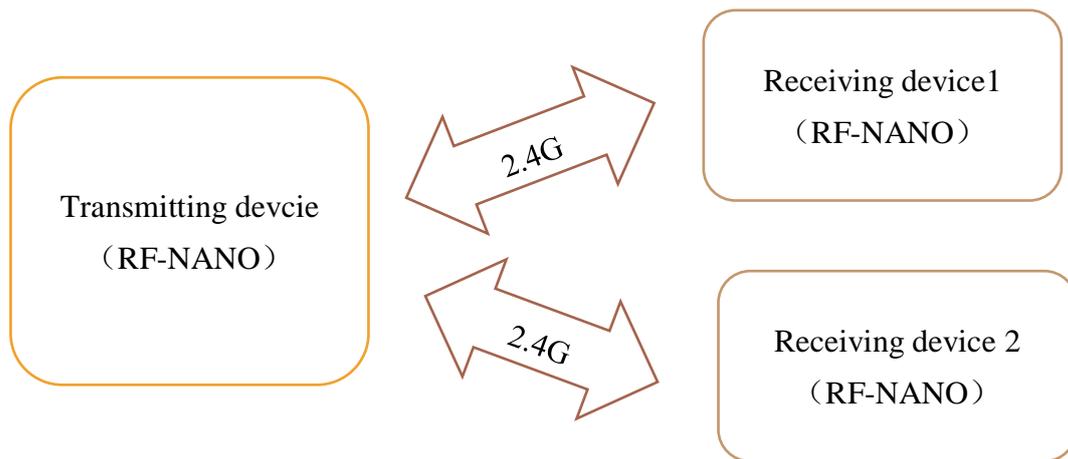


Figure 3-2-3

3.3 Implement a send to multiple receive communication

3.3.1 Experimental principle block diagram



3.3.2 program code

Transmit data program code:

- **Code path:** RF-NANO Demo Program\One send to multiple receive communication\All_Emitter\All_Emitter.ino
- **Source code**

```
//Transmitter program

#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
Nrf24l Mirf = Nrf24l(10, 9);
int value1,value2;

void setup()
{
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  //Set your own address (sender address) using 5 characters
  Mirf.setRADDR((byte *)"ABCDE");
}

void loop()
{
  Mirf.payload = sizeof(value1);
  Mirf.channel = 10;           //Set the channel used
  Mirf.config();
  if(Mirf.channel == 10)
  {
    Mirf.setTADDR((byte *)"FGHIJ");           //Set the receiver address
    value1 = 100;
    Mirf.send((byte *)&value1);           //Send instructions, send random number value
    Serial.print("Wait for sending.....");
    while (Mirf.isSending()) delay(1);           //Until you send successfully, exit the loop
    Serial.print("value1 Send success:");
    Serial.println(value1);
    delay(500);
  }
  Mirf.payload = sizeof(value2);
  Mirf.channel = 20;           //Set the channel used
  Mirf.config();
  if(Mirf.channel == 20)
  {
    Mirf.setTADDR((byte *)"KLMNO");           //Set the receiver address
```

```

value2 = 200;
Mirf.send((byte *)&value2);           //Send instructions, send random number value
Serial.print("Wait for sending.....");
while (Mirf.isSending()) delay(1);     //Until you send successfully, exit the loop
Serial.print("value2 Send success:");
Serial.println(value2);
delay(500);
}
}

```

Data sent by transmitter:

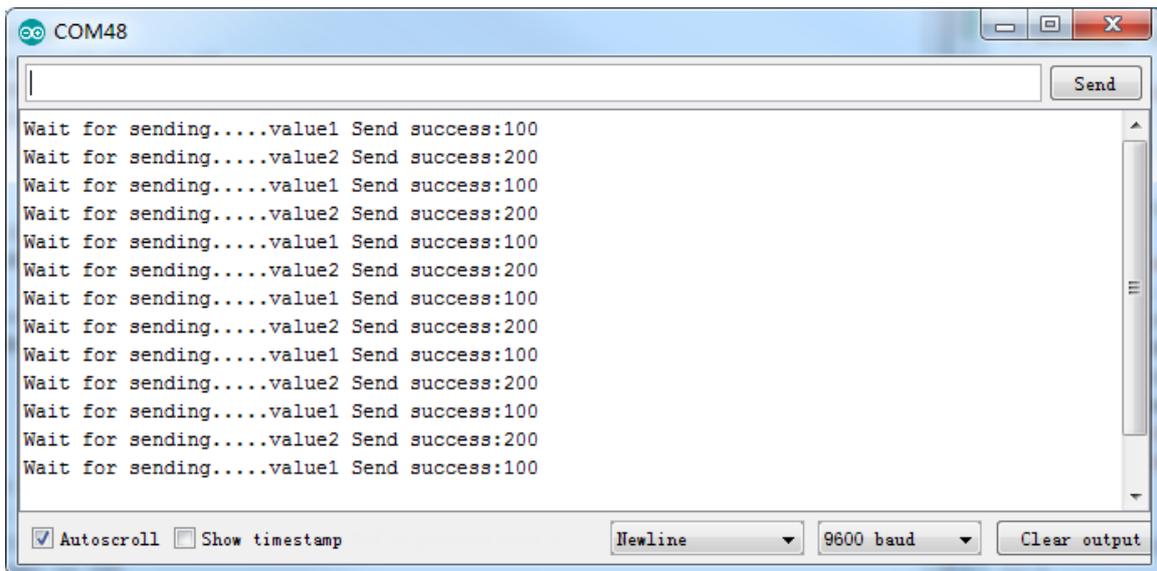


Figure 3-3-1

Receiving data program code 1:

- **Code path:** RF-NANO Demo Program\One send to multiple receive communication\Receive1\Receive1.ino
- **Source code**

```

//Receiver program
#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"

Nrf24l Mirf = Nrf24l(10, 9);

```

```
int value;

void setup()
{
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"FGHIJ"); //Set your own address (receiver address) using 5
characters
  Mirf.payload = sizeof(value);
  Mirf.channel = 10;           //Set the used channel
  Mirf.config();
  Serial.println("Listening..."); //Start listening to received data
}

void loop()
{
  Mirf.ceLow();
  Mirf.configRegister(RF_CH, 10); //switch channel 10
  Mirf.ceHi();
  if (Mirf.dataReady()) { //When the program is received, the received data is output from
the serial port
    Mirf.getData((byte *) &value);
    Serial.print("Receiver got data is: ");
    Serial.println(value);
  }
  delay(800);
}
```

Data received by receiving device 1:

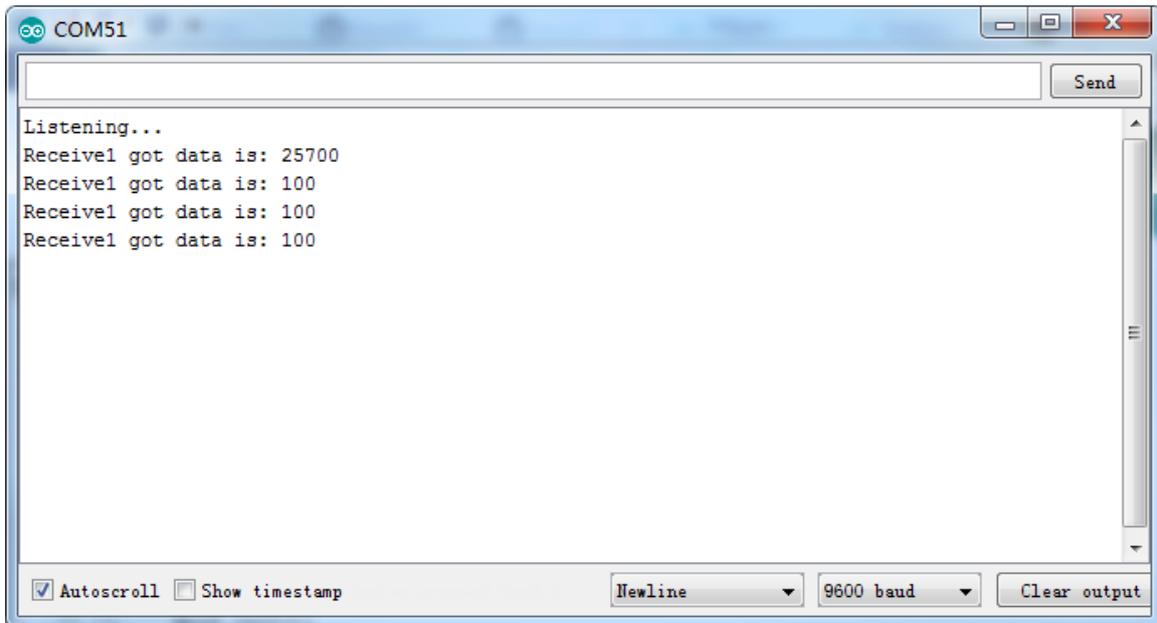


Figure 3-3-2

Receiving data program code 2:

- **Code path:** RF-NANO Demo Program\One send to multiple receive communication\Receive2\Receive2.ino
- **Source code**

Data received by receiving device 2:

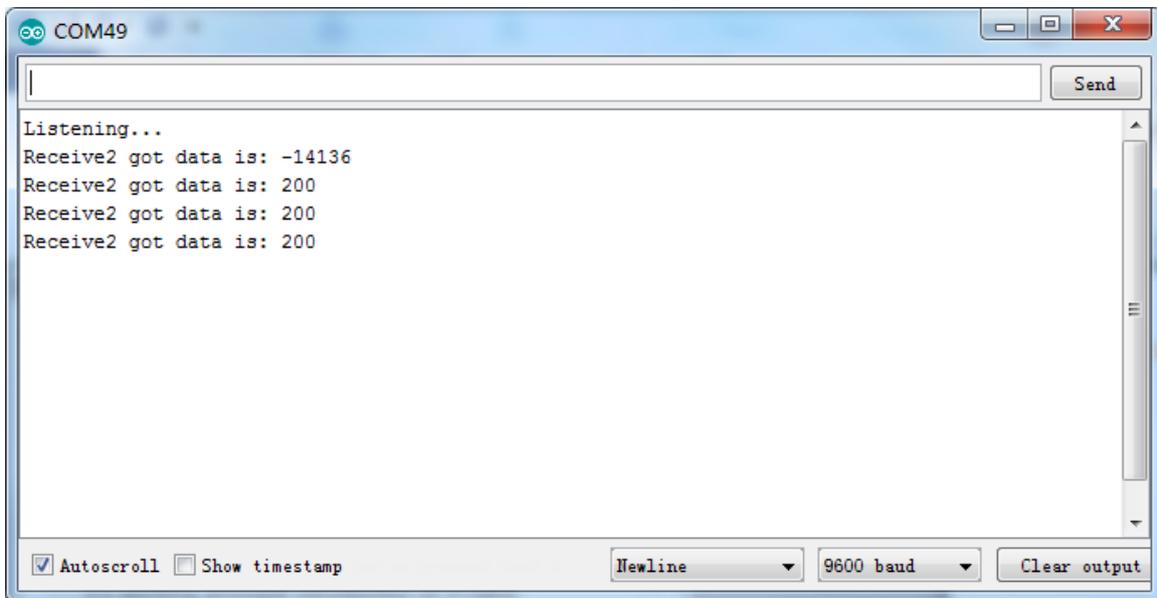


Figure 3-3-3

Chapter 4 Sets the transmitting power and data transmission rate

4.1 Set the transmitted power of RF-NANO

The transmitted power of RF-NANO has four levels: -18dBm, -12dBm, -6dBm and 0dBm. The greater the transmitting power, the farther the transmission distance; The transmitted power can be set in the software. Open the rf-nano experimental routine program. In mirf. CPP, there is a function to set the transmitted power :SetOutputRF_PWR(uint8_t val), which can set different transmitted power

```
void Nrf241::SetOutputRF_PWR(uint8_t val) //Set tx power :
0=-18dBm,1=-12dBm,2=-6dBm,3=0dBm,
{
    configRegister(RF_SETUP, (val<< RF_PWR) );
}
void Nrf241::config()
// Sets the important registers in the MiRF module and powers the module
// in receiving mode
// NB: channel and payload must be set now.
{
    configRegister(RF_CH, channel); // Set RF channel
    configRegister(RX_PW_P0, payload); // Set length of incoming payload
    configRegister(RX_PW_P1, payload);
    SetSpeedDataRates(0); //Select between the high speed data rates:250Kbps
    powerUpRx(); // Start receiver
    flushRx();
}
```

4.2 Set RF-NANO data transfer rate

RF-NANO data transmission rate has three levels, namely :1Mbps, 2Mbps, 250Kbps; Data transmission rate can be set in the software. Open the rf-nano experimental routine program. In mirf. CPP, there is a function to set the data transmission rate :SetSpeedDataRates(uint8_t val)。

```
void Nrf241::SetSpeedDataRates(uint8_t val) //Select between the high speed data
rates:0=1Mbps, 1=2Mbps, 2=250Kbps
{
    if(val>1)
    {
        configRegister(RF_SETUP, (1 << RF_DR_LOW) );
    }
    else
    {
```

```
configRegister(RF_SETUP, (val << RF_DR_HIGH) );  
}  
}
```