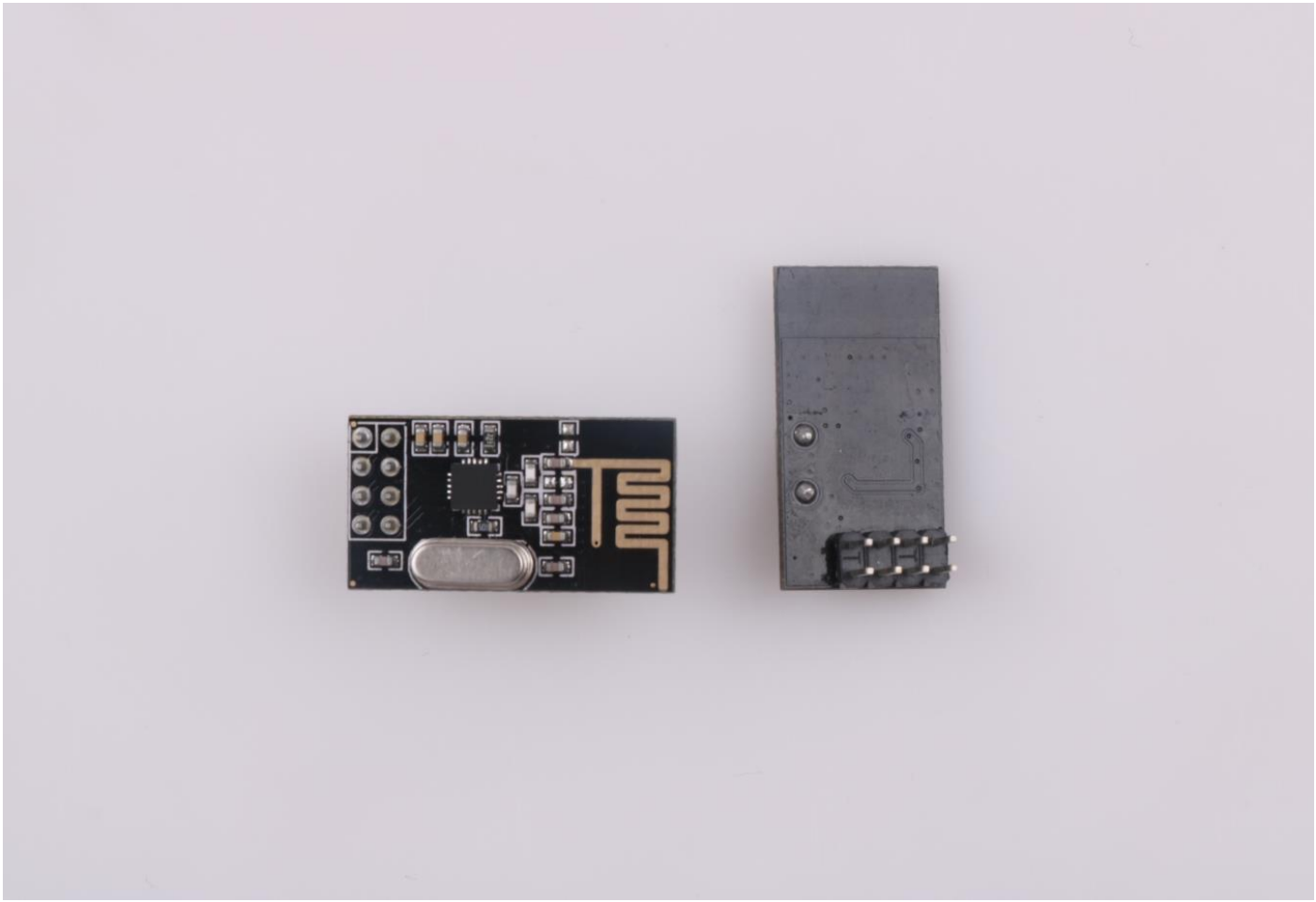
nRF24L01+

The nRF24L01+ module is a 2.4G wireless communication module developed by Nordic based on the nRF24L01 chip. Adopt FSK modulation and integrate Nordic's own Enhanced Short Burst protocol. Point-to-point or 1-to-6 wireless communication can be achieved. Wireless communication speed can reach up to 2M (bps). The NRF24L01 has four operating modes: transceiver mode, configuration mode, idle mode, and shutdown mode.The picture of nRF24L01+ as follows:



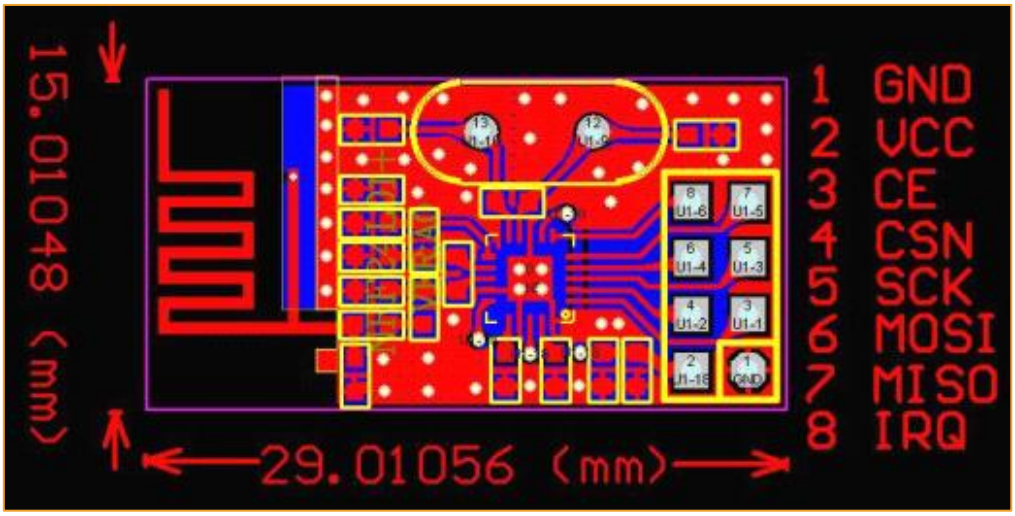
Module features

- 2.4GHz ,small size，15x29mm（including antenna）
- Support six-channel data reception
- Low working voltage: 1.9～3.6V
- Data transfer rate supports: 1Mbps、2Mbps
- Low power consumption design，working current at receiving is 12.3mA, 11.3mA at 0dBm power emission, 900nA at power-down mode
- Automatic retransmission function, automatic detection and retransmission of lost data packets, retransmission time and retransmission times can be controlled by software

- Automatic response function, after receiving valid data, the module automatically sends an answer signal

NRF24L01 chip details please refer to《nRF24L01 Datasheet.pdf》

Pin information



Pin	Symbol	Function	Direction
1	GND	GND	
2	+5V	Power Supply	
3	CE	Control Line At Working Mode	IN
4	CSN	Chip Select Signal, Low Level Working	IN
5	SCK	SPI Clock	IN
6	MOSI	SPI Input	IN
7	MISO	SPI Output	OUT
8	IPQ	Interrupt Output	OUT +

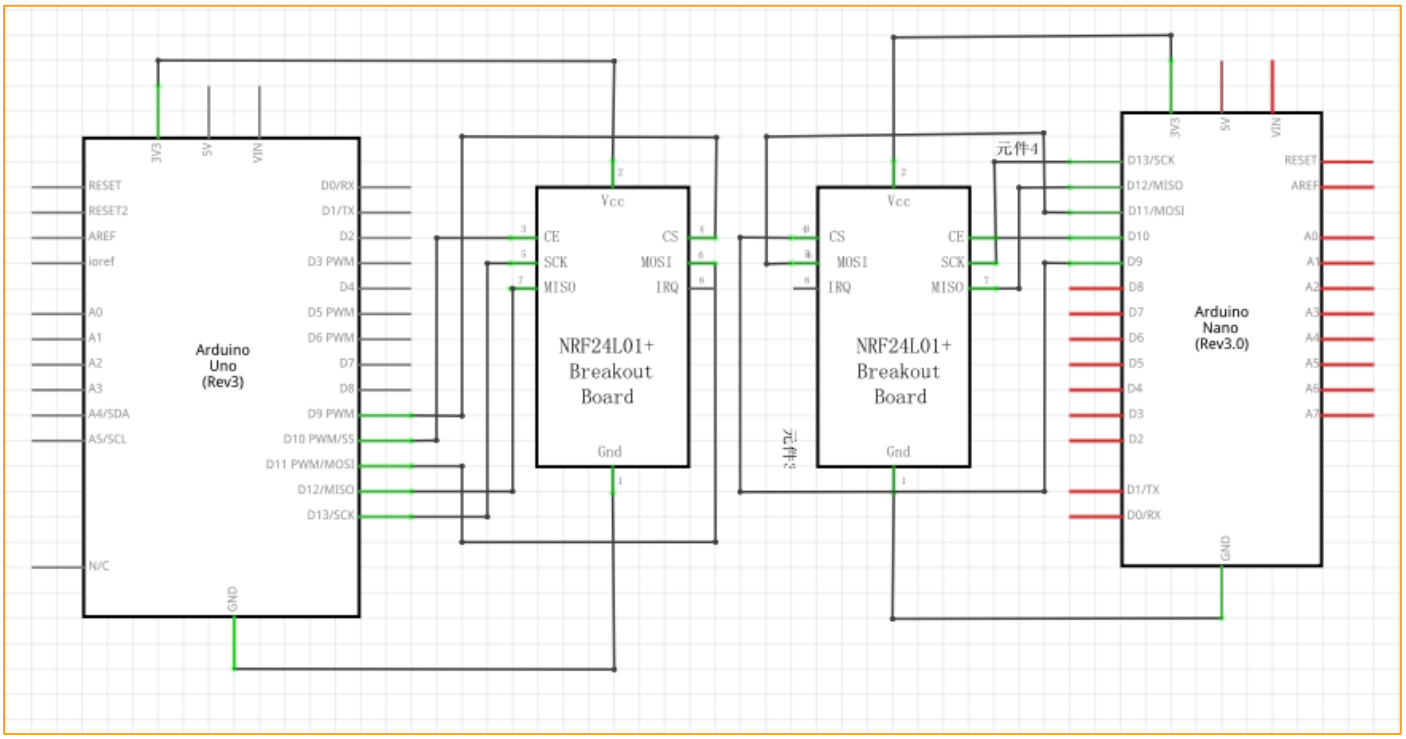
Experimental Purpose

1. Learn about nRF24L01+module and how to connect with Arduino.
2. How to use arduino and nRF24L01+ module to finish receiving and sending data.

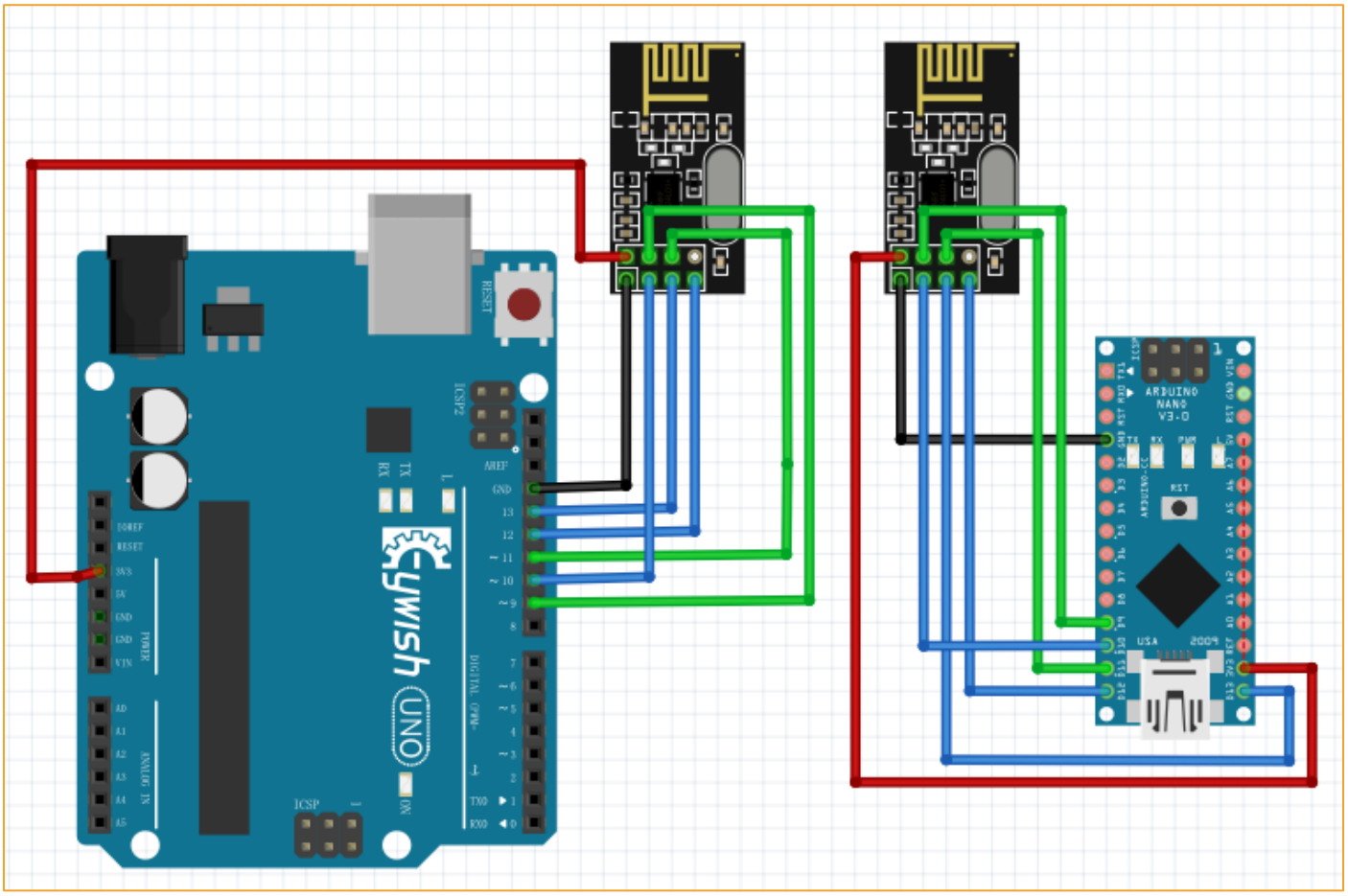
The components needed for this experiment

- Arduino UNO R3 Motherboard *1
- Arduino NANO Motherboard
- nRF24L01 Module*2
- Several wires

Experimental schematic diagram



Experimental connection diagram



Arduino and NRF24L01 connection

arduino Nano	nRF24L01
+3.3V	VCC
GND	GND
9pin	4pin CSN
10pin	3pin CE
11pin	6pin MOSI
12pin	7pin MISO
13pin	5pin SCK

arduino Uno	nRF24L01
+3.3V	VCC
GND	GND
9pin	4pin CSN
10pin	3pin CE
11pin	6pin MOSI

12pin	7pin MISO
13pin	5pin SCK

Program principle

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 nRF24L01 buffer area by the SPI port in time sequence.

3、Set CE to be high voltage for at least 10 μs and transmits data after a delay of 130 μs. 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 will set high and TX_PLD will not be cleared; If MAX_RT or 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 130μs 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.

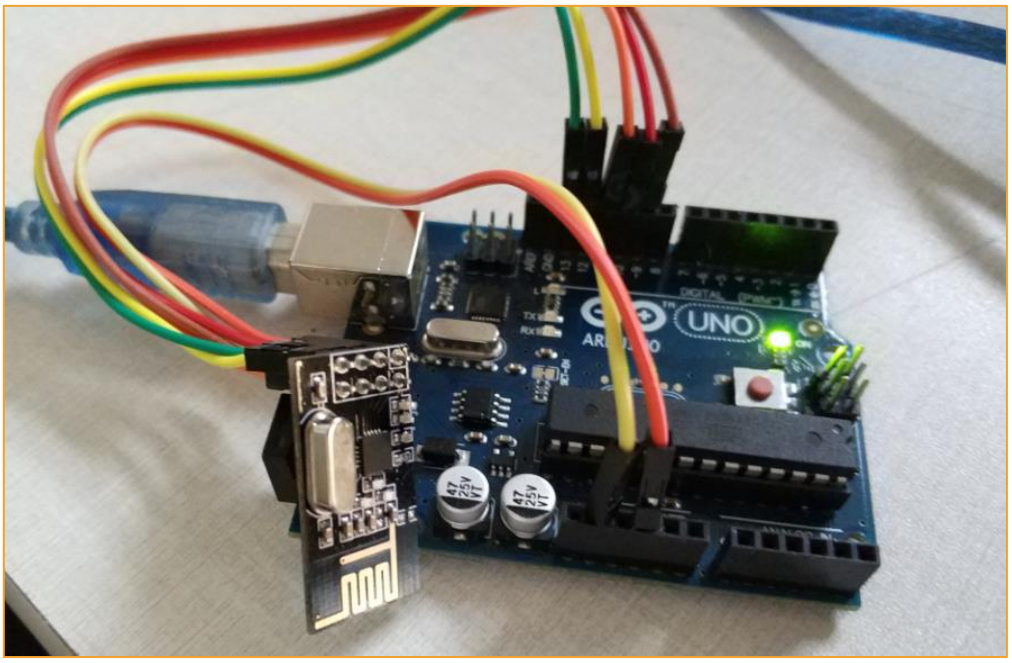
Sender program

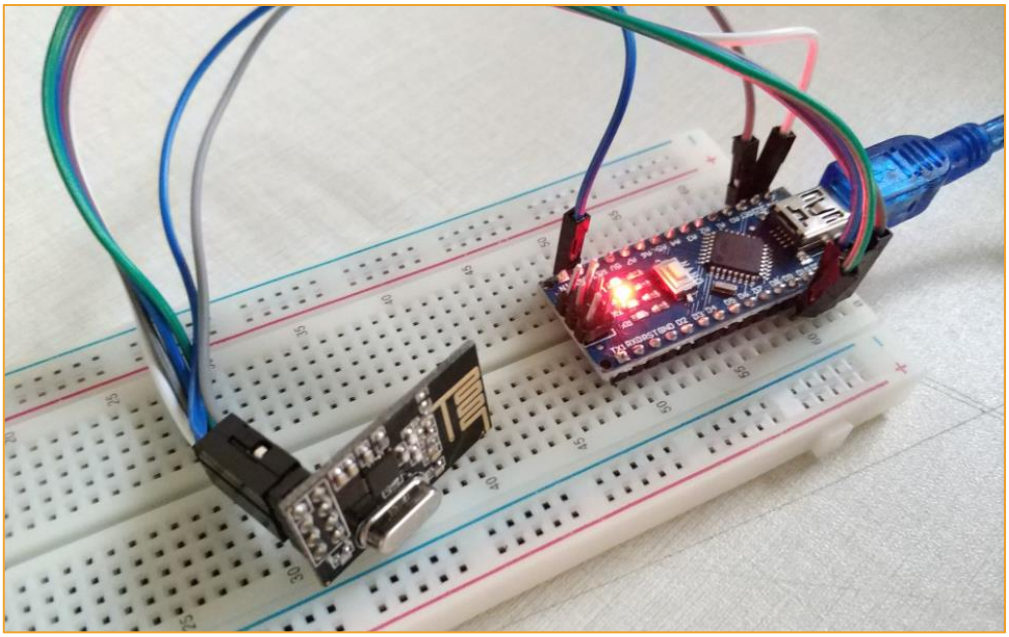
```
//Transmitter program
#include <SPI.h>
#include "Mirf.h"
#include "nRF24L01.h"
#include "MirfHardwareSpiDriver.h"
Nrf24l Mirf = Nrf24l(10, 9);
int value;

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 = 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((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);
}
```

Sender program

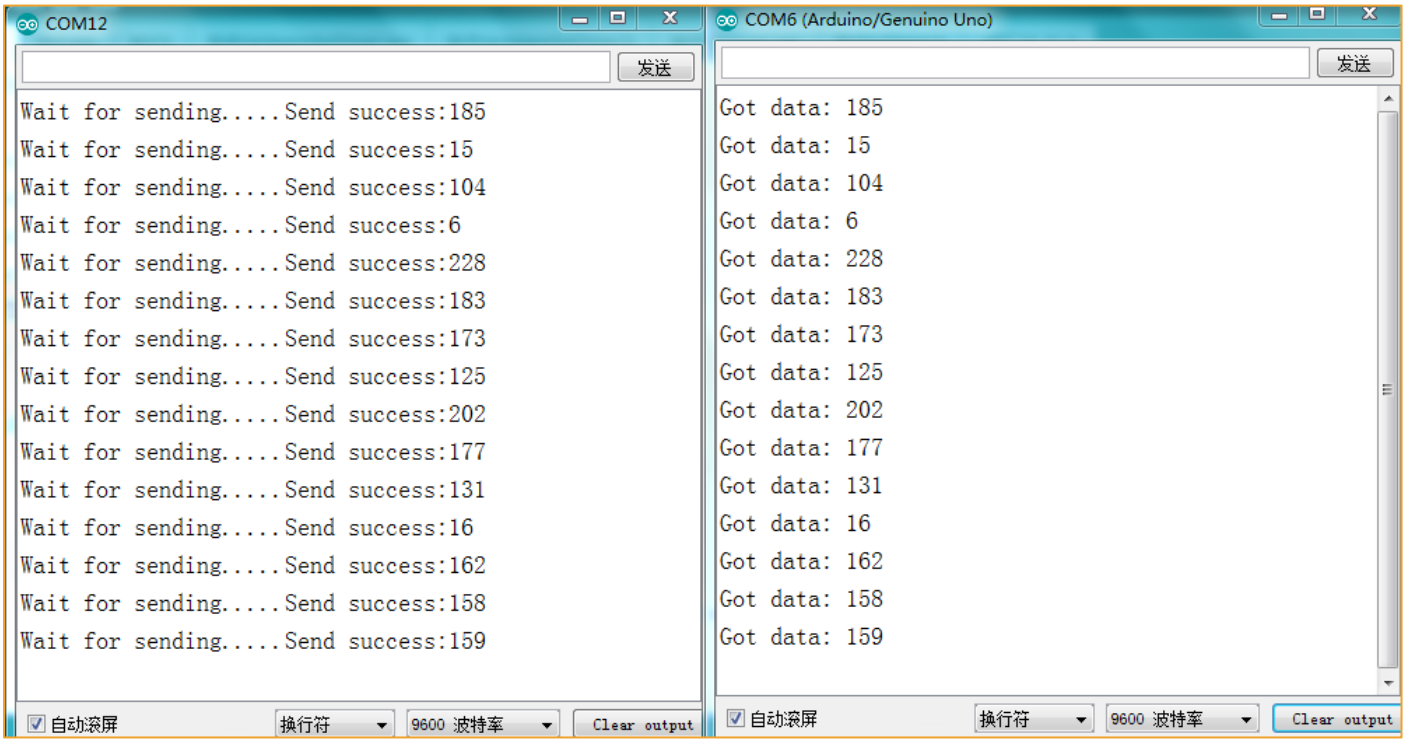
```
//Receiver program

#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 = 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((byte *) &value);
    Serial.print("Got data: ");
    Serial.println(value);
  }
}
```

Experimental





Result：


COM12
发送
Wait for sending.....Send success:185
Wait for sending.....Send success:15
Wait for sending.....Send success:104
Wait for sending.....Send success:6
Wait for sending.....Send success:228
Wait for sending.....Send success:183
Wait for sending.....Send success:173
Wait for sending.....Send success:125
Wait for sending.....Send success:202
Wait for sending.....Send success:177
Wait for sending.....Send success:131
Wait for sending.....Send success:16
Wait for sending.....Send success:162
Wait for sending.....Send success:158
Wait for sending.....Send success:159
自动滚屏
换行符
9600 波特率
Clear output
COM6 (Arduino/Genuino Uno)
发送
Got data: 185
Got data: 15
Got data: 104
Got data: 6
Got data: 228
Got data: 183
Got data: 173
Got data: 125
Got data: 202
Got data: 177
Got data: 131
Got data: 16
Got data: 162
Got data: 158
Got data: 159
自动滚屏
换行符
9600 波特率
Clear output