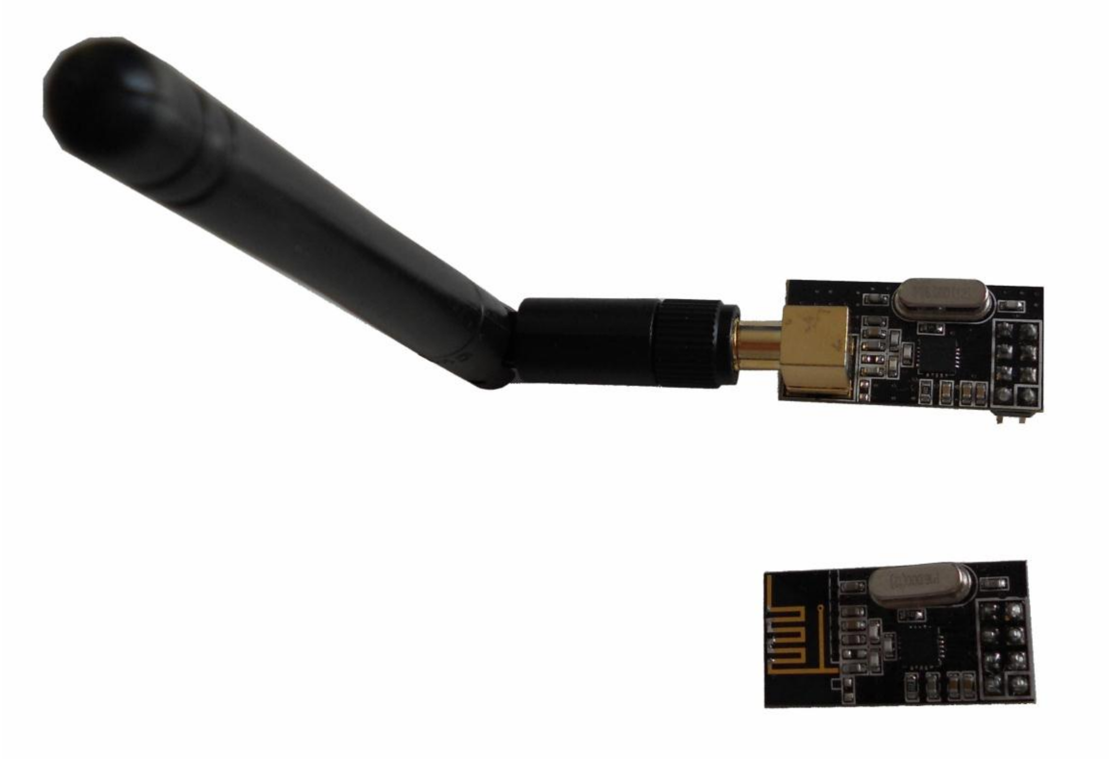
ELECHOUSE NRF24L01



Version 1.0, December, 2010

ELECHOUSE_NRF24L01 Module PINS

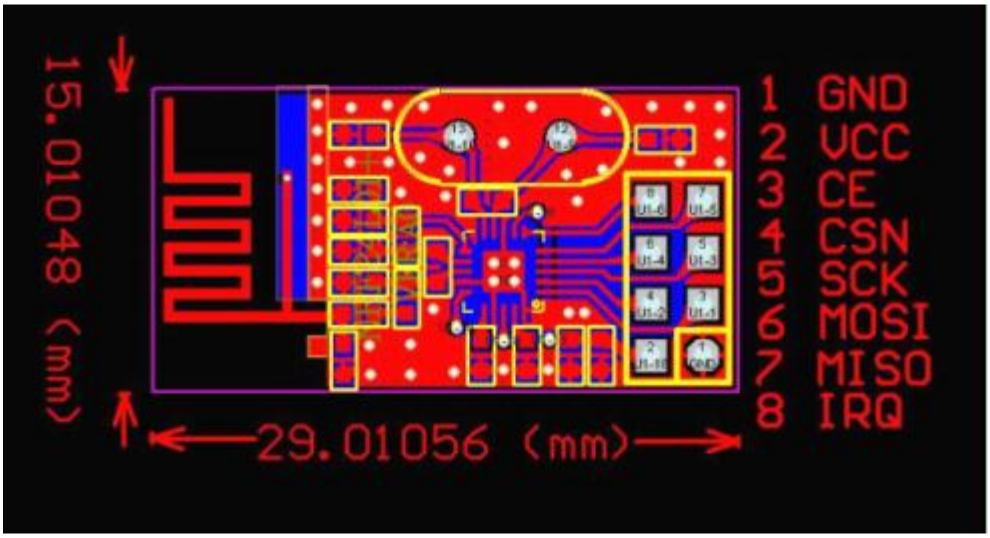


Figure 1 pins

VCC: *3.3V*

GND: *Ground*

CE: *Digital Input. Chip Enable Activates RX or TX mode. Need a resistor about 500 Ω in series when connecting to Arduino board. In the library ELECHOUSE_NRF24L01.h, the default setting is Digital pin 9.*

CSN: *Digital Input. SPI Chip Select. Connecting with Digital pin 10 of Arduino board through a resistor about 500 Ωin series.*

SCK: *Digital Input. SPI Clock. Connecting with Digital pin 13 of Arduino board through a resistor about 500 Ωin series.*

MOSI: *Digital Input. SPI Slave Data Input. Connecting with Digital pin 11 of Arduino board through a resistor about 500 Ω in series.*

MISO: *Digital Output. SPI Slave Data Output. Connecting with Digital pin 12 of Arduino board.*

IRQ: *Digital Output. Maskable interrupt pin, Active low. In the library ELECHOUSE_NRF24L01.h, the default setting is Digital pin 2.*

ELECHOUSE_NRF24L01.h Library

This library is designed to use NRF24L01 module on Arduino platform. Using the functions of the library, you can easily send and receive data by the NRF24L01 module. When using, you should copy the library folder to the path "\arduino-0018\arduino-0018\libraries\" first.

For the users who want to realize wireless communication through the module, you just need to know the following functions. And I think it is enough for most applications. If you want to study the chip NRF24L01 carefully, you can refer the other functions in the ELECHOUSE_NRF24L01.h library and the datasheet for details.

void Init(void)

Description

NRF24L01 initialization, including the setting of Spi pins, mode and the setting of IRQ, CE, CSN pins. It must be called before using other functions.

Parameters

None

Returns

None

void RegConfigSettings(void)

Description

Some common NRF24L01 register configuration, including RF, address width, auto ack enable. It should be called after Init and before RX/TX.

Parameters

None

Returns

None

*void RX_Setting(byte pipe_num, byte *pipe_address, byte pipe_addr_width, byte RX_pload_width)*

Description

NRF24L01 RX setting, including the pipe number for receiving and the corresponding pipe address, address width, RX payload width. It should be called before start RX mode. And it can be called several times for different pipe setting if you need.

Parameters

pipe_num: the number of pipe (0-5)

pipe_address: address corresponding to pipe_num

pipe_addr_width: address width corresponding to pipe_address (pipe 0-1 is 5; pipe 2-5 is 1)

RX_pload_width: RX payload width (should be equal to TX payload width)

Returns

None

void RX_ModeStart(void)

Description

Make NRF24L01 into RX mode.

Parameters

None

Returns

None

*void TX_Setting(byte *TX_addr, byte TX_addr_width, byte *TX_data_buf, byte TX_pload_width)*

Description

NRF24L01 TX setting, it should be called before start TX mode.

Parameters

TX_addr: TX destination address

TX_addr_width: TX address width(equal to the setting in SETUP_AW)

TX_data_buf: data buffer to send

TX_pload_width: TX payload width(equal to RX payload width)

Returns

None

void TX_ModeStart(void)

Description

Make NRF24L01 into TX mode.

Parameters

None

Returns

None

byte CheckSendFlag(void)

Description

Check whether transmits data successfully or not.

Parameters

None

Returns

TX flag, return 1 when TX successfully.

byte CheckReceiveFlag(void)

Description

Check receiving data or not.

Parameters

None

Returns

RX flag, return 1 when receiving data.

*byte ReceiveData(byte *rxBuffer, byte RX_pload_width)*

Description

Read data received from RX FIFO.

Parameters

rxBuffer: buffer to store data

RX_pload_width: RX data width

Returns

The pipe number which received data. It will return 0x07 if no data.

Demo

This demo shows that two NRF24L01 modules communicate wirelessly on Arduino platform. The PTX sends 0-31 circularly. The PRX receives the content and prints through serial monitor. When you practice this demo, first connect the corresponding pins of the module and Arduino board like figure 2 *and table 1 shows. Note for the pins CE, CSN, SCK, MOSI, there should be a resistor about 500 Ω in serials between the module and Arduino in case burning the chip.*

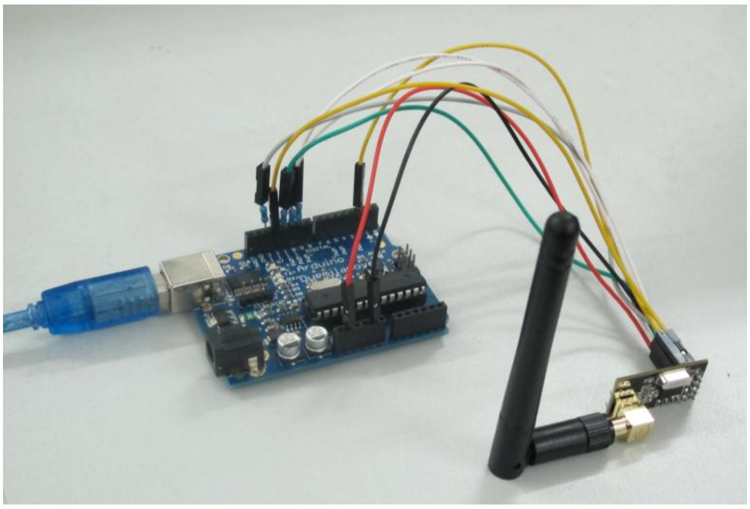
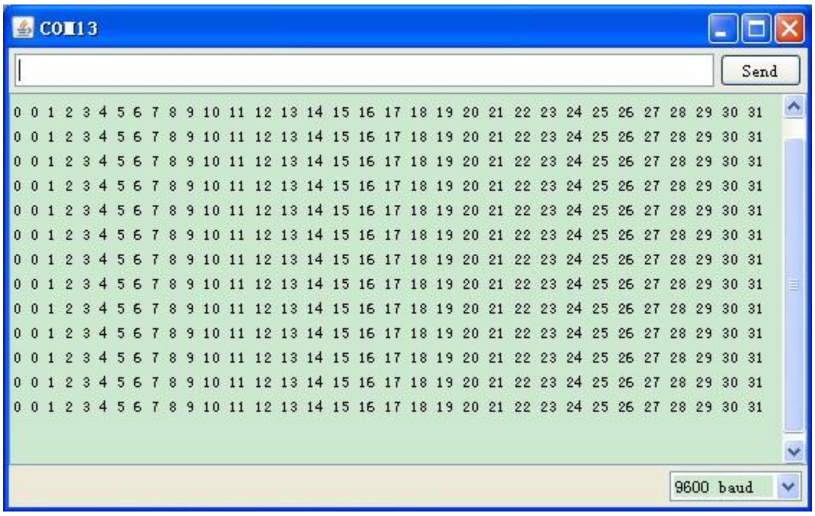


Figure 2 circuit connection

Table 2 pins corresponding

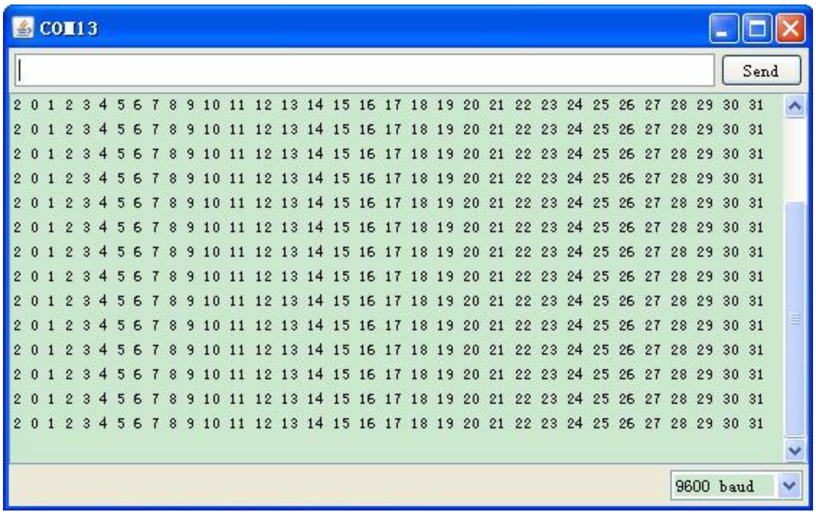
NRF24L01 module	*Arduino*
VCC	*3.3V*
GND	*GND*
CE	*Digital 9*
CSN	*Digital 10*
SCK	*Digital 13*
MOSI	*Digital 11*
MISO	*Digital 12*
IRQ	*Digital 2*

The demo result is like figure 3 and figure 4.



```
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

Figure 3 pipe 0



```
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
2 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

Figure 4 pipe 2

CODE

PTX Code

```
/*

    TX_ADR_WIDTH default is 5, modify the SETUP_AW setting in library file for other value

    TX_PLOAD_WIDTH should be equal to RX_PLOAD_WIDTH

*/
```

```
#include <ELECHOUSE_NRF24L01.h>


#define TX_ADR_WIDTH      5        // TX address width

#define TX_PLOAD_WIDTH 32      // TX payload width


byte TX_ADDRESS[TX_ADR_WIDTH] = {0x00,0x10,0x10,0x10,0x10};    // TX address

byte TX_buffer[TX_PLOAD_WIDTH]={0};          // data buffer to transmit

byte i,temp;


void setup()

{

   Serial.begin(9600);

   ELECHOUSE_nRF24L01.Init();            //initialization

   ELECHOUSE_nRF24L01.RegConfigSettings();    //RF,address width,enable AA

   for(i=0;i<TX_PLOAD_WIDTH;i++)

   {

        TX_buffer[i]=i;

   }

}


void loop()

{
  ELECHOUSE_nRF24L01.TX_Setting(TX_ADDRESS,TX_ADR_WIDTH,TX_buffer,TX_PLOAD_WIDTH);//TX setting
   ELECHOUSE_nRF24L01.TX_ModeStart();   //start TX
   temp=ELECHOUSE_nRF24L01.CheckSendFlag();   //TX flag
   Serial.print(temp,HEX);
   Serial.println("");
   delay(1000);
}
```

PRX Code

```
/*
  RX_ADR_WIDTH default is 5, modify the SETUP_AW setting in library file for other value
  RX_ADR_WIDTH should be equal to TX_PLOAD_WIDTH
*/
#include <ELECHOUSE_NRF24L01.h>


#define RX_ADR_WIDTH     5    // RX address width
#define RX_PLOAD_WIDTH    32    // RX payload width


byte RX_ADDRESS_P0[RX_ADR_WIDTH] = {0x00,0x10,0x10,0x10,0x10}; //pipe 0 address
byte RX_ADDRESS_P1[RX_ADR_WIDTH] = {0x01,0x10,0x10,0x10,0x10};   //pipe 1 address
byte RX_ADDRESS_P2[1] = {0x02};                      // pipe 2 address, MSBytes are equal to pipe 1
byte RX_buffer[RX_PLOAD_WIDTH]={0};              //RX buffer
byte i,temp;


void setup()

{

   Serial.begin(9600);

   ELECHOUSE_nRF24L01.Init();               //initialization

   ELECHOUSE_nRF24L01.RegConfigSettings();    //RF, address width, enable AA


  ELECHOUSE_nRF24L01.RX_Setting(0,RX_ADDRESS_P0,RX_ADR_WIDTH,RX_PLOAD_WIDTH);
  // pipe 0 address


  ELECHOUSE_nRF24L01.RX_Setting(1,RX_ADDRESS_P1,RX_ADR_WIDTH,RX_PLOAD_WIDTH);
  // pipe 1 address

   ELECHOUSE_nRF24L01.RX_Setting(2,RX_ADDRESS_P2,1,RX_PLOAD_WIDTH); //pipe 2
  address

   ELECHOUSE_nRF24L01.RX_ModeStart(); //start receive

}


void loop()

{

   temp=ELECHOUSE_nRF24L01.CheckReceiveFlag(); //check status

   if(temp)                                   // have data

   {

      temp=ELECHOUSE_nRF24L01.ReceiveData(RX_buffer,RX_PLOAD_WIDTH);//pipe number
  and RX data

      Serial.print(temp,HEX);
```

```
    Serial.print(' ',BYTE);
    for(i=0;i<RX_PLOAD_WIDTH;i++)
    {
      Serial.print(RX_buffer[i],DEC);
      Serial.print(' ',BYTE);
    }
    Serial.println("");
    ELECHOUSE_nRF24L01.RX_ModeStart();
  }
  delay(1);
}
```