

الحمادي للإلكترونيات
ALHAMMADI FOR ELECTRONICS

AA102

NRF24L01 with Antenna 1100 Meters

V1.0.23.10.20

Preface

OurCompany

KUONGSHUN Electronic Company is a supplier and manufacturer of electronic components, it is committed to board and starter kit for Arduino, Raspberry PI, Smart Robot Car, 3D printer. It is also a collection of scientific research, design, production, maintenance and sales of high-tech enterprises, in the field of automation with professional standards and mature technology, we rapid rise in the field of foreign trade. Relying on technology and development, continuing to provide users with high-tech products, is our constant pursuit. Fully introduction of foreign advanced technology to enhance the value of our products.

Company gains users' praise for supplying first-class quality product and superb technical services, has now become the first choice of domestic and international procurement company.

Official Website: <https://www.kuongshun-ks.com>

AA102 NRF24L01 with Antenna 1100 Meters



Product Description

The NRF24L01 is a wireless transceiver module that adds a power amplifier and removable external antenna to greatly extend the effective communication range, which depends on whether the use of the environment is an open area and the presence of obstacles and the use of an external antenna or not, and which can be used with any microcontroller (MCU) for a variety of applications in wireless control.

Product Parameters

Supply voltage: DC 3V-3.6V

Maximum current: about 115mA

Operating frequency: 2.4GHz ~ 2.5GHz

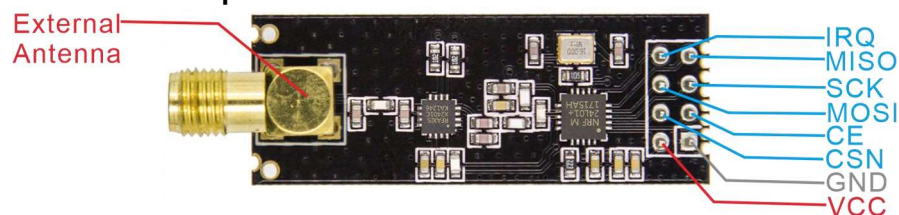
Communication distance: Maximum 1100 meters

Support up to six channels of data reception

Applications: remote control, wireless remote control, body sensing devices, home automation, etc.

Communication distance: maximum 1100 meters (using the environment of open flat land, without any obstacles and signal interference, the actual use of obstacles, and a variety of electromagnetic wireless signal interference, the distance is much less, in the indoor through the wall is less than 100 meters or less)

Interface Description

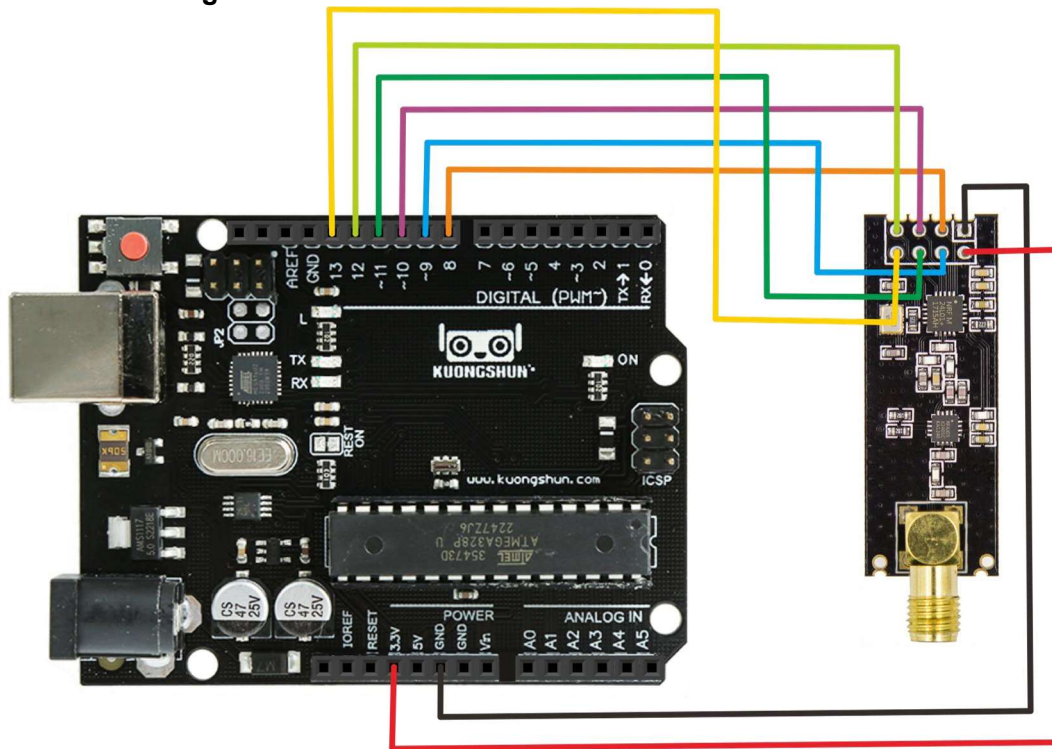


VCC: power supply DC 3V-3.6V

GND: Ground terminal (common ground with MCU)

CE: RX or TX mode selection
 CSN: SPI chip select signal
 SCK: SPI clock input
 MOSI: SPI data input pin
 MISO: SPI data output pin
 IRQ: Maskable interrupt pin, low level enable

Connection Diagram



Sample Code

The transmit mode test code is as follows.

```

/*
https://www.kuongshun-ks.com/
*/
/*****
**  SPI-compatible                               **
**  CE - to digital pin 8                       **
**  CSN - to digital pin 9 (SS pin)             **
**  SCK - to digital pin 10 (SCK pin)          **
**  MOSI - to digital pin 11 (MOSI pin)        **
**  MISO - to digital pin 12 (MISO pin)        **
**  IRQ - to digital pin 13 (MISO pin)         **
*****/
#include <SPI.h>
#include <NRF24L01.h>

```

```

//*****
#define TX_ADR_WIDTH 5 // 5 unsigned chars TX(RX) address width #define
TX_PLOAD_WIDTH 32 // 32 unsigned chars TX payload

unsigned char TX_ADDRESS[TX_ADR_WIDTH] =
{
    0x34,0x43,0x10,0x10,0x01
}; // Define a static TX address
unsigned char rx_buf[TX_PLOAD_WIDTH] = {0}; // initialize value
unsigned char tx_buf[TX_PLOAD_WIDTH] = {0};
//*****

void setup()
{
    pinMode(CE, OUTPUT);
    pinMode(SCK, OUTPUT);
    pinMode(CSN, OUTPUT);
    pinMode(MOSI, OUTPUT);
    pinMode(MISO, INPUT);
    pinMode(IRQ, INPUT);
    // attachInterrupt(1, _ISR, LOW); // interrupt enable
    Serial.begin(9600);
    init_io(); // Initialize IO port
    unsigned char status=SPI_Read(STATUS);
    Serial.print("status = ");
    Serial.println(status,HEX); // There is read the mode's status register, the default value should be
'E'
    Serial.println("*****TX_Mode Start*****");
    TX_Mode(); // set TX mode
}

void loop()
{
    int k = 0;
    for(;;)
    {
        for(int i=0; i<32; i++)
            tx_buf[i] = k++;
        unsigned char status = SPI_Read(STATUS); // read register STATUS's value
        if(status&TX_DS) // if receive data ready (TX_DS) interrupt
        {
            SPI_RW_Reg(FLUSH_TX,0);
            SPI_Write_Buf(WR_TX_PLOAD,tx_buf,TX_PLOAD_WIDTH); // write payload to TX_FIFO
        }
    }
}

```

```

        if(status&MAX_RT) // if receive data ready (MAX_RT) interrupt, this is retransmit than
SETUP_RETR
    {
        SPI_RW_Reg(FLUSH_TX,0);
        SPI_Write_Buf(WR_TX_PLOAD,tx_buf,TX_PLOAD_WIDTH); // disable standby-mode
    }
    SPI_RW_Reg(WRITE_REG+STATUS,status); // clear RX_DR or TX_DS or
MAX_RT interrupt flag
    delay(1000);
    }
}

```

```

//*****

```

```

// Function: init_io();

```

```

// Description:

```

```

// flash led one time,chip enable(ready to TX or RX Mode),

```

```

// Spi disable,Spi clock line init high

```

```

//*****

```

```

void init_io(void)

```

```

{
    digitalWrite(IRQ, 0);
    digitalWrite(CE, 0); // chip enable digitalWrite(CSN, 1); // Spi
    disable
}

```

```

/*****

```

```

* Function: SPI_RW();

```

```

*

```

```

* Description:

```

```

* Writes one unsigned char to nRF24L01, and return the unsigned char read

```

```

* from nRF24L01 during write, according to SPI protocol

```

```

*****/

```

```

unsigned char SPI_RW(unsigned char Byte)

```

```

{
    unsigned char i;

```

```

for(i=0;i<8;i++)                // output 8-bit
{
    if(Byte&0x80)
    {
        digitalWrite(MOSI, 1);
    }
    else
    {
        digitalWrite(MOSI, 0);
    }
    digitalWrite(SCK, 1);
    Byte <<= 1;                // shift next bit into MSB..
    if(digitalRead(MISO) == 1)
    {
        Byte |= 1;            // capture current MISO bit
    }
    digitalWrite(SCK, 0);
}
return(Byte);                    // return read unsigned char
}
/*****/

/*****
* Function: SPI_RW_Reg();
*
* Description:
* Writes value 'value' to register 'reg'
*****/
unsigned char SPI_RW_Reg(unsigned char reg, unsigned char value)
{
    unsigned char status;

    digitalWrite(CSN, 0);        // CSN low, init SPI transaction
    status = SPI_RW(reg);        // select register
    SPI_RW(value);                // ..and write value to it..
    digitalWrite(CSN, 1);        // CSN high again

    return(status);                // return nRF24L01 status unsigned char
}
/*****/

/*****
* Function: SPI_Read();
*
* Description:

```

```

* Read one unsigned char from nRF24L01 register, 'reg'
/*****/
unsigned char SPI_Read(unsigned char reg)
{
    unsigned char reg_val;

    digitalWrite(CSN, 0);          // CSN low, initialize SPI
                                   communication...
    SPI_RW(reg);                   // Select register to read from..
    reg_val = SPI_RW(0);           // ..then read register value
    digitalWrite(CSN, 1);         // CSN high, terminate SPI communication

    return(reg_val);              // return register value
}
/*****/

/*****/
* Function: SPI_Read_Buf();
*
* Description:
* Reads 'unsigned chars' #of unsigned chars from register 'reg'
* Typically used to read RX payload, Rx/Tx address
/*****/
unsigned char SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
    unsigned char status,i;

    digitalWrite(CSN, 0);         // Set CSN low, init SPI transaction status = SPI_RW(reg);    // Select
    register to write to and read status unsigned char

    for(i=0;i<bytes;i++)
    {
        pBuf[i] = SPI_RW(0);      // Perform SPI_RW to read unsigned char from nRF24L01
    }

    digitalWrite(CSN, 1);         // Set CSN high again

    return(status);              // return nRF24L01 status unsigned char
}
/*****/

/*****/
* Function: SPI_Write_Buf();
*
* Description:

```

```

* Writes contents of buffer '*pBuf' to nRF24L01
* Typically used to write TX payload, Rx/Tx address
/*****/
unsigned char SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
    unsigned char status,i;

    digitalWrite(CSN, 0);           // Set CSN low, init SPI transaction
    status = SPI_RW(reg);           // Select register to write to and read status unsigned char
    for(i=0;i<bytes; i++)           // then write all unsigned char in buffer(*pBuf)
    {
        SPI_RW(*pBuf++);
    }
    digitalWrite(CSN, 1);           // Set CSN high again
    return(status);                 // return nRF24L01 status unsigned char
}
/*****/

/*****/
* Function: TX_Mode();
*
* Description:
* This function initializes one nRF24L01 device to * TX mode, set TX address, set RX address for
  auto.ack, * fill TX payload, select RF channel, datarate & TX pwr.
* PWR_UP is set, CRC(2 unsigned chars) is enabled, & PRIM:TX.
*
* ToDo: One high pulse(>10us) on CE will now send this
* packet and expect an acknowledgment from the RX device.
*****/
void TX_Mode(void)
{
    digitalWrite(CE, 0);

    SPI_Write_Buf(WRITE_REG + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH); // Writes TX_Address
to nRF24L01
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); // RX_Addr0 same
as TX_Adr for Auto.Ack

    SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);           // Enable Auto.Ack:Pipe0
    SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01);       // Enable Pipe0
    SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a);      // 500us + 86us, 10 retrans...
    SPI_RW_Reg(WRITE_REG + RF_CH, 40);             // Select RF channel 40
    SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);        // TX_PWR:0dBm,
Datarate:2Mbps, LNA:HCURR

```

```
SPI_RW_Reg(WRITE_REG + CONFIG, 0x0e); // Set PWR_UP bit, enable CRC(2 unsigned chars) &
Prim:TX. MAX_RT & TX_DS enabled..
```

```
SPI_Write_Buf(WR_TX_PLOAD,tx_buf,TX_PLOAD_WIDTH);
```

```
digitalWrite(CE, 1);
```

```
}
```

The receive mode test code is as follows.

```
/*
```

```
https://www.kuongshun-ks.com/
```

```
*/
```

```
/**
```

```
// SPI-compatible **
```

```
// CE - to digital pin 8 **
```

```
// CSN - to digital pin 9 (SS pin) **
```

```
// SCK - to digital pin 10 (SCK pin) **
```

```
// MOSI - to digital pin 11 (MOSI pin) ** // MISO - to digital
pin 12 (MISO pin) **
```

```
// IRQ - to digital pin 13 (MISO pin) **
```

```
/**
```

```
#include <SPI.h>
```

```
#include <NRF24L01.h>
```

```
/**
```

```
#define TX_ADR_WIDTH 5 // 5 unsigned chars TX(RX) address width #define
```

```
TX_PLOAD_WIDTH 32 // 32 unsigned chars TX payload
```

```
unsigned char TX_ADDRESS[TX_ADR_WIDTH] =
```

```
{
```

```
0x34,0x43,0x10,0x10,0x01
```

```
}; // Define a static TX address
```

```
unsigned char rx_buf[TX_PLOAD_WIDTH];
```

```
unsigned char tx_buf[TX_PLOAD_WIDTH];
```

```
/**
```

```
void setup()
```

```
{
```

```
pinMode(CE, OUTPUT);
```

```
pinMode(SCK, OUTPUT);
```

```
pinMode(CSN, OUTPUT);
```

```
pinMode(MOSI, OUTPUT);
```

```
pinMode(MISO, INPUT);
```

```
pinMode(IRQ, INPUT);
```

```

// attachInterrupt(1, _ISR, LOW); // interrupt enable
Serial.begin(9600);
init_io(); // Initialize IO port
unsigned char status=SPI_Read(STATUS);
Serial.print("status = ");
Serial.println(status,HEX); // There is read the mode's status register, the default value should be
'E'
Serial.println("*****RX_Mode start*****R");
RX_Mode(); // set RX mode
}
void loop()
{ for(;;)
{
    unsigned char status = SPI_Read(STATUS); // read register STATUS's
value
    if(status&RX_DR) // if receive data ready
(TX_DS) interrupt
    {
        SPI_Read_Buf(RD_RX_PLOAD, rx_buf, TX_PLOAD_WIDTH); // read payload to
rx_buf
        SPI_RW_Reg(FLUSH_RX,0); // clear RX_FIFO for(int i=0; i<32; i++)
        {
            Serial.print(" ");
            Serial.print(rx_buf[i],HEX); // print rx_buf
        }
        Serial.println(" ");
    }
    SPI_RW_Reg(WRITE_REG+STATUS,status); // clear RX_DR or
TX_DS or MAX_RT interrupt flag
    delay(1000);
}
}

//*****
// Function: init_io();
// Description:
// flash led one time,chip enable(ready to TX or RX Mode),
// Spi disable,Spi clock line init high
//*****
void init_io(void)
{
    digitalWrite(IRQ, 0);
    digitalWrite(CE, 0); // chip enable digitalWrite(CSN, 1); // Spi
disable
}

```

```

}

/*****
* Function: SPI_RW();
*
* Description:
* Writes one unsigned char to nRF24L01, and return the unsigned char read
* from nRF24L01 during write, according to SPI protocol
*****/
unsigned char SPI_RW(unsigned char Byte)
{
    unsigned char i;
    for(i=0;i<8;i++)                // output 8-bit
    {
        if(Byte&0x80)
        {
            digitalWrite(MOSI, 1);    // output 'unsigned char', MSB to MOSI
        }
        else
        {
            digitalWrite(MOSI, 0);
        }
        digitalWrite(SCK, 1);        // Set SCK high..
        Byte <<= 1;                  // shift next bit into MSB..
        if(digitalRead(MISO) == 1)
        {
            Byte |= 1;                // capture current MISO bit
        }
        digitalWrite(SCK, 0);        // ..then set SCK low again
    }
    return(Byte);                    // return read unsigned char
}

/*****

/*****
* Function: SPI_RW_Reg();
*
* Description:
* Writes value 'value' to register 'reg'
*****/
unsigned char SPI_RW_Reg(unsigned char reg, unsigned char value)
{
    unsigned char status;

```

```

digitalWrite(CSN, 0);           // CSN low, init SPI transaction
status = SPI_RW(reg);         // select register
SPI_RW(value);                 // ..and write value to it..
digitalWrite(CSN, 1);         // CSN high again

return(status);               // return nRF24L01 status unsigned char
}
/*****/
/*****/
* Function: SPI_Read();
*
* Description:
* Read one unsigned char from nRF24L01 register, 'reg'
/*****/
unsigned char SPI_Read(unsigned char reg)
{
    unsigned char reg_val;

    digitalWrite(CSN, 0);      // CSN low, initialize SPI
                               // communication...
    SPI_RW(reg);               // Select register to read from..
    reg_val = SPI_RW(0);       // ..then read register value
    digitalWrite(CSN, 1);     // CSN high, terminate SPI communication

    return(reg_val);          // return register value
}
/*****/

/*****/
* Function: SPI_Read_Buf();
*
* Description:
* Reads 'unsigned chars' #of unsigned chars from register 'reg'
* Typically used to read RX payload, Rx/Tx address
/*****/
unsigned char SPI_Read_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
    unsigned char status,i;

    digitalWrite(CSN, 0);     // Set CSN low, init SPI transaction
    status = SPI_RW(reg);     // Select register to write to and read status unsigned char

    for(i=0;i<bytes;i++)
    {
        pBuf[i] = SPI_RW(0);  // Perform SPI_RW to read unsigned char from nRF24L01
    }
}

```

```

}

digitalWrite(CSN, 1);                // Set CSN high again

return(status);                      // return nRF24L01 status unsigned char
}
/*****/

/*****
 * Function: SPI_Write_Buf();
 *
 * Description:
 * Writes contents of buffer '*pBuf' to nRF24L01
 * Typically used to write TX payload, Rx/Tx address
 *****/
unsigned char SPI_Write_Buf(unsigned char reg, unsigned char *pBuf, unsigned char bytes)
{
    unsigned char status,i;

    digitalWrite(CSN, 0);            // Set CSN low, init SPI tranaction
    status = SPI_RW(reg);            // Select register to write to and read status unsigned char
    for(i=0;i<bytes; i++)            // then write all unsigned char in buffer(*pBuf)
    {
        SPI_RW(*pBuf++);
    }
    digitalWrite(CSN, 1);            // Set CSN high again
    return(status);                  // return nRF24L01 status unsigned char
}
/*****/

/*****
 * Function: RX_Mode();
 *
 * Description:
 * This function initializes one nRF24L01 device to * RX Mode, set RX address, writes RX payload width,
 * select RF channel, datarate & LNA HCURR.
 * After init, CE is toggled high, which means that
 * this device is now ready to receive a datapacket.
 *****/
void RX_Mode(void)
{
    digitalWrite(CE, 0);
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH); // Use the same
address on the RX device as the TX device

```

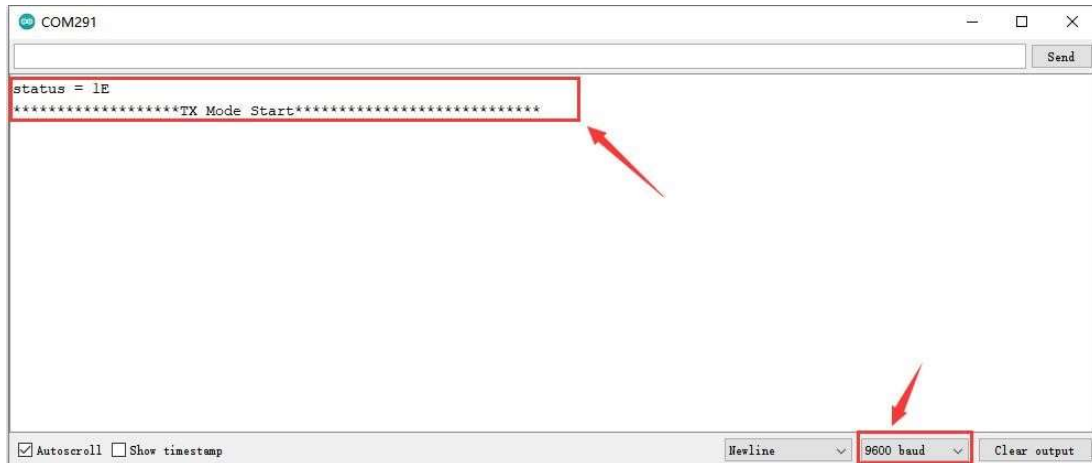
```

SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);    // Enable Auto.Ack:Pipe0
SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01); // Enable Pipe0
SPI_RW_Reg(WRITE_REG + RF_CH, 40);      // Select RF channel 40
SPI_RW_Reg(WRITE_REG + RX_PW_P0, TX_PLOAD_WIDTH); // Select same RX payload width as
TX Payload width
SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07); // TX_PWR:0dBm, Datarate:2Mbps,
LNA:HCURR
SPI_RW_Reg(WRITE_REG + CONFIG, 0x0f); // Set PWR_UP bit, enable CRC(2 unsigned chars) &
Prim:RX. RX_DR enabled..
digitalWrite(CE, 1);                    // Set CE pin high to enable RX device
// This device is now ready to receive one packet of 16 unsigned chars payload from a TX device
sending to address
// '3443101001', with auto acknowledgment, retransmit count of 10, RF channel 40 and datarate =
2Mbps.
}
/*****/

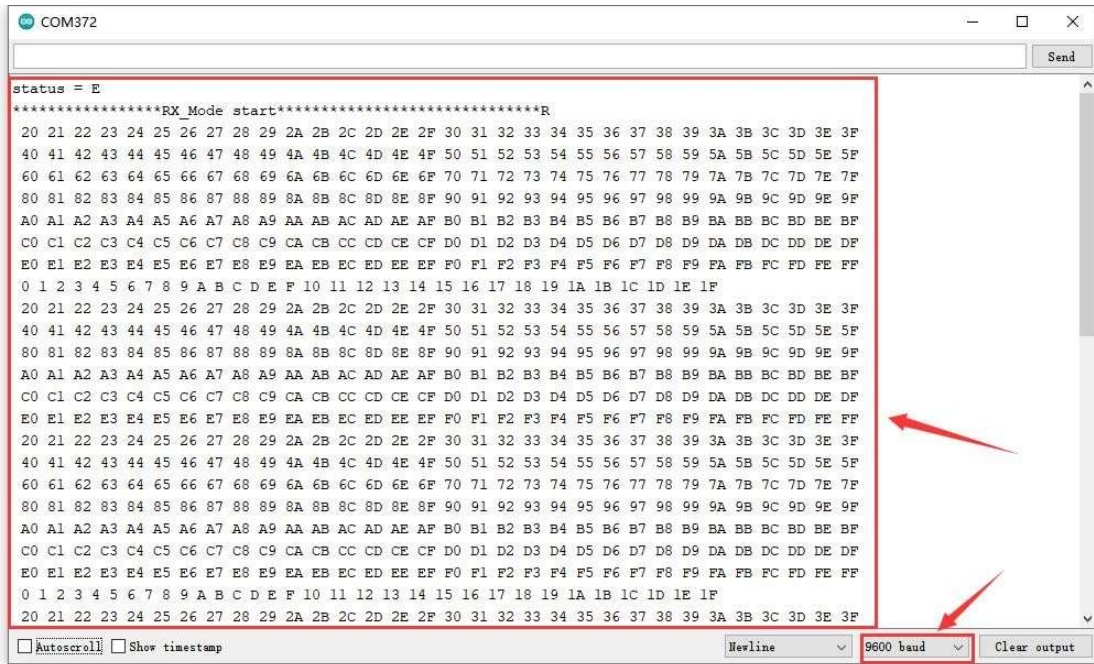
```

Test Result

Connect the wires according to the wiring method, the control board uploads the transmit mode test code, after power on, the module starts to send information, open the serial monitor, set the baud rate to 9600, and the monitor displays the following figure.



In accordance with the wiring method to connect the line, the control board to upload the receiving module test code, power on, the module will be able to receive the information sent by another module, open the serial port monitor, set the baud rate to 9600, the monitor will be able to display the received information, as follows.



Resources

Related code and library files are linked below:

XXXXXXXXXXXX