User Manual

For

RF905 Transceiver Module(ME109)

Description

This wireless module board adapts high-performance Nordic VLSI NRF905 radio chip, the maximum transmission data rates up to 50Kbps with GFSK, sensitivity to-100dBm, high reliability, it is widely used in various occasions, short-range wireless communications (such as wireless meter reading, industrial remote control, low-power handheld equipment, etc.)

Features:

- Operating voltage: 1.9 ~ 3.6V, (3.3V is recommended)
- Work on the 433/868MHz
- The maximum operating rate of 50kbps, support GFSK modulation
- A lower current consumption
- Programmable control of output power, for all the support frequencies of up to +10 dBm
- Standard DIP spacing interfaces for embedded applications
- Transmission Distance: open to the actual transmission distance 200-300 meters (depending on the specific situation of the environment and communication baud rate settings, etc.)
- Module Size: 32mm * 19mm (the size of non-SMA head and antenna)

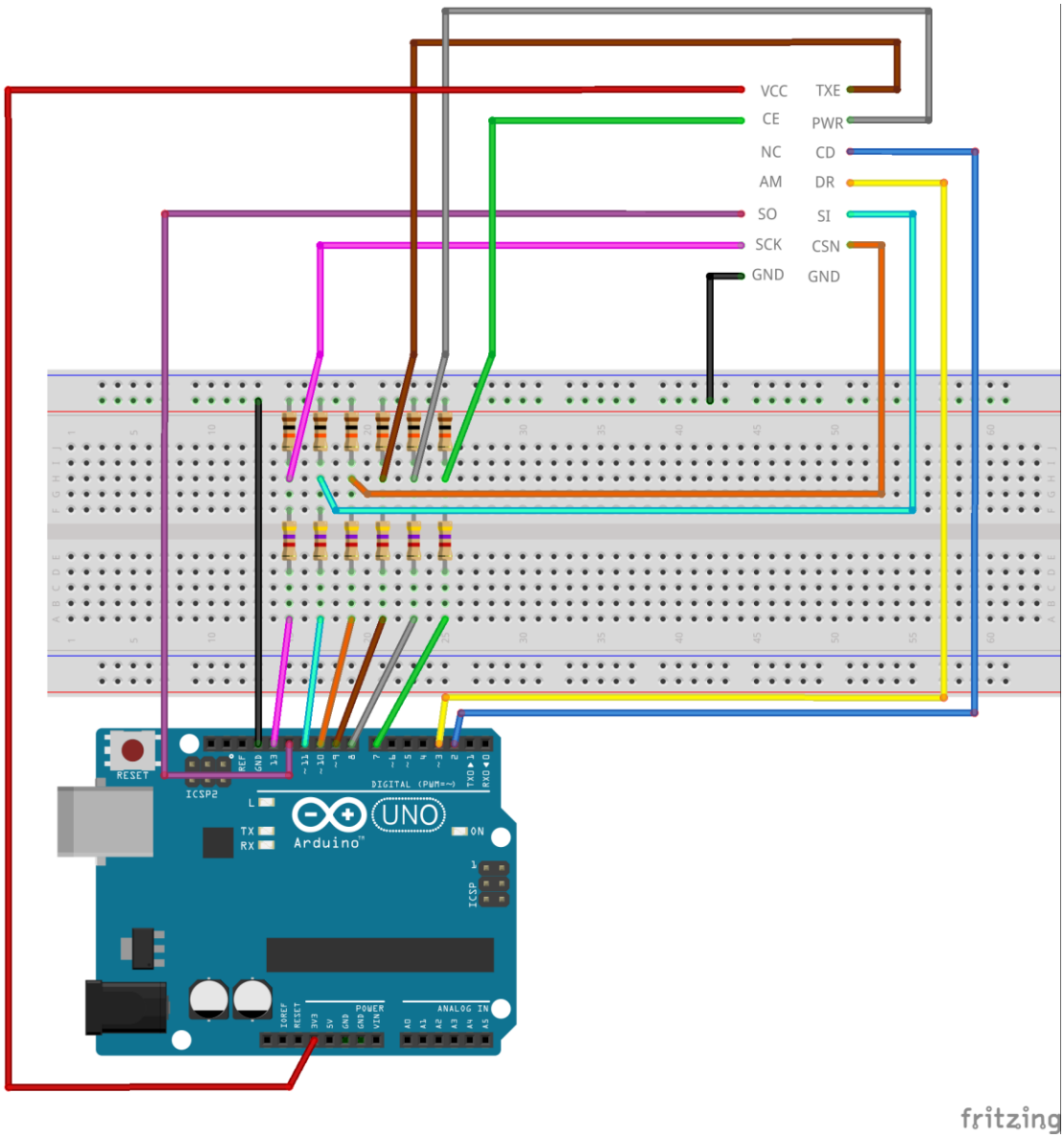
Pinout

nRF905 Pin	Arduino Uno pin connected	Pin Description
VCC	3.3V	Power(3.3V)
CE	7	Standy-High=TX/RX mode,Low=standby
TXE	9	TX or Rx mode-High=TX,Low=RX
PWR	8	Power up – High = on, Low = off
CD	2	Carrier detect – High when a signal is detected, for collision avoidance
AM	-	Address Match – High when receiving a packet that has the same address as the one set for this device, optional since state is stored in register, not used by this library
DR	3	Data Ready – High when finished transmitting/High when new data received, optional since state is stored in register, if interrupts are used this pin must be connected **NOTE:** On Arduino Mega change INTERRUPT_NUM to 5 in nRF905_config.h and for Arduino Yun it should be changed to 0.
SO	12	SPI MISO
SI	11	SPI MOSI
SCK	13	SPI SCK
CSN	10	SPI SS
GND	GND	Ground

Example

The nRF905 is not 5V compatible, so some level conversions will need to be done with the Arduino outputs, a simple 470R resistor will do the trick, only TXE, CE, PWR, SI, SCK and CSN pins need level conversion (not CD, AM, DR and SO).

Or you can download the Library from here, which contains fully example and annotation.
https://github.com/zkemble/nRF905



**********Code Begin*********

```
/*
 * Wireless serial link
 *
 * 7 -> CE
 * 8 -> PWR
 * 9 -> TXE
 * 2 -> CD
 * 3 -> DR
 * 10 -> CSN
 * 12 -> SO
 * 11 -> SI
 * 13 -> SCK
 */
```

```
#include <nRF905.h>
#include <SPI.h>

#define PACKET_TYPE_DATA  0
#define PACKET_TYPE_ACK      1

#define MAX_PACKET_SIZE (NRF905_MAX_PAYLOAD - 2)
typedef struct {
 byte dstAddress[NRF905_ADDR_SIZE];
 byte type;
 byte len;
 byte data[MAX_PACKET_SIZE];
} packet_s;

void setup()
{
 // Start up
 nRF905_init();

 // Put into receive mode
 nRF905_receive();

 Serial.begin(9600);

 Serial.println(F("Ready"));
}

void loop()
{
 packet_s packet;

 // Send serial data
 byte dataSize;
 while((dataSize = Serial.available()))
 {
    // Make sure we don't try to send more than max packet
size
    if(dataSize > MAX_PACKET_SIZE)
       dataSize = MAX_PACKET_SIZE;

    packet.type = PACKET_TYPE_DATA;
    packet.len = dataSize;

    // Copy data from serial to packet buffer
```

```
    for(byte i=0;i<dataSize;i++)
       packet.data[i] = Serial.read();

    // Send packet
    sendPacket(&packet);

    // Receive mode
    nRF905_receive();

    // Wait for ACK packet
    byte startTime = millis();
    while(1)
    {
       bool timeout = false;
       while(1)
       {
          if(getPacket(&packet)) // Get new packet
             break;
          else if((byte)(millis() - startTime) > 50) // 50ms
timeout
          {
             timeout = true;
             break;
          }
       }

       if(timeout) // Timed out
       {
          Serial.println(F("TO"));
          break;
       }
       else if(packet.type == PACKET_TYPE_ACK) // Is packet
type ACK?
          break;
    }
 }

 // Put into receive mode
 nRF905_receive();

 // Wait for data
 while(1)
 {
    if(getPacket(&packet) && packet.type == PACKET_TYPE_DATA)
```

```
// Got a packet and is it a data packet?
    {
        // Print data
        Serial.write(packet.data, packet.len);

        // Reply with ACK
        packet.type = PACKET_TYPE_ACK;
        packet.len = 0;
        sendPacket(&packet);

        // Put into receive mode
        nRF905_receive();
    }
    else if(Serial.available()) // We've got some serial data,
need to send it
        break;
 }
}

// Send a packet
static void sendPacket(void* _packet)
{
 // Void pointer to packet_s pointer hack
 // Arduino puts all the function defs at the top of the file
before packet_s being declared :/
 packet_s* packet = (packet_s*)_packet;

 // Convert packet data to plain byte array
 byte totalLength = packet->len + 2;
 byte tmpBuff[totalLength];
 tmpBuff[0] = packet->type;
 tmpBuff[1] = packet->len;
 memcpy(&tmpBuff[2], packet->data, packet->len);

 // Set address of device to send to
 //nRF905_setTXAddress(packet->dstAddress);

 // Set payload data
 nRF905_setData(tmpBuff, totalLength);

 // Send payload (send fails if other transmissions are going
on, keep trying until success)
 while(!nRF905_send());
}
```

```
// Get a packet
static bool getPacket(void* _packet)
{
 // Void pointer to packet_s pointer hack
 // Arduino puts all the function defs at the top of the file
before packet_s being declared :/
 packet_s* packet = (packet_s*)_packet;

 byte buffer[NRF905_MAX_PAYLOAD];

 // See if any data available
 if(!nRF905_getData(buffer, sizeof(buffer)))
    return false;

 // Convert byte array to packet
 packet->type = buffer[0];
 packet->len = buffer[1];

 // Sanity check
 if(packet->len > MAX_PACKET_SIZE)
    packet->len = MAX_PACKET_SIZE;

 memcpy(packet->data, &buffer[2], packet->len);

 return true;
}
```

********Code End*******