

## USING nRF24L01+ & nRF24L01+ PA LNA WIRELESS MODULE WITH BREAKOUT BOARD



### Introduction

The nRF24L01+ module designed to operate in the 2.4 GHz worldwide ISM frequency band and uses GFSK modulation for data transmission. The data transfer rate is configurable and can be set to 250kbps, 1Mbps, or 2Mbps.

### Specification

|                            |                        |
|----------------------------|------------------------|
| Frequency Range            | 2.4 GHz ISM Band       |
| Maximum Air Data Rate      | 2 Mb/s                 |
| Modulation Format          | GFSK                   |
| Max. Output Power          | 0 dBm                  |
| Operating Supply Voltage   | 1.9 V to 3.6 V         |
| Max. Operating Current     | 13.5mA                 |
| Min. Current(Standby Mode) | 26µA                   |
| Logic Inputs               | 5V Tolerant            |
| Communication Range        | 800+ m (line of sight) |

**Objective:**

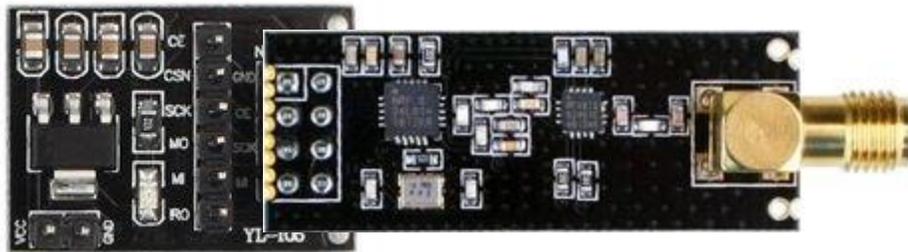
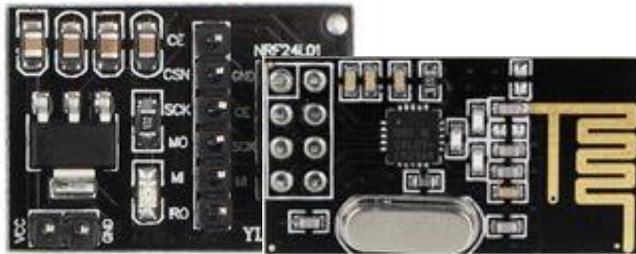
To establish connection and transmit data transfer between two nRF24L01 module.

**Note:**

Setting up nRF24L01 without breakout can be hit or miss due lack of power consistencies from Arduino. Breakout board will eliminate the voltage inconsistencies to nRF module. Adding external DC 5v power source to breakout board recommended.

**Connection Diagram:**

Attach Module on Breakout board as picture below.



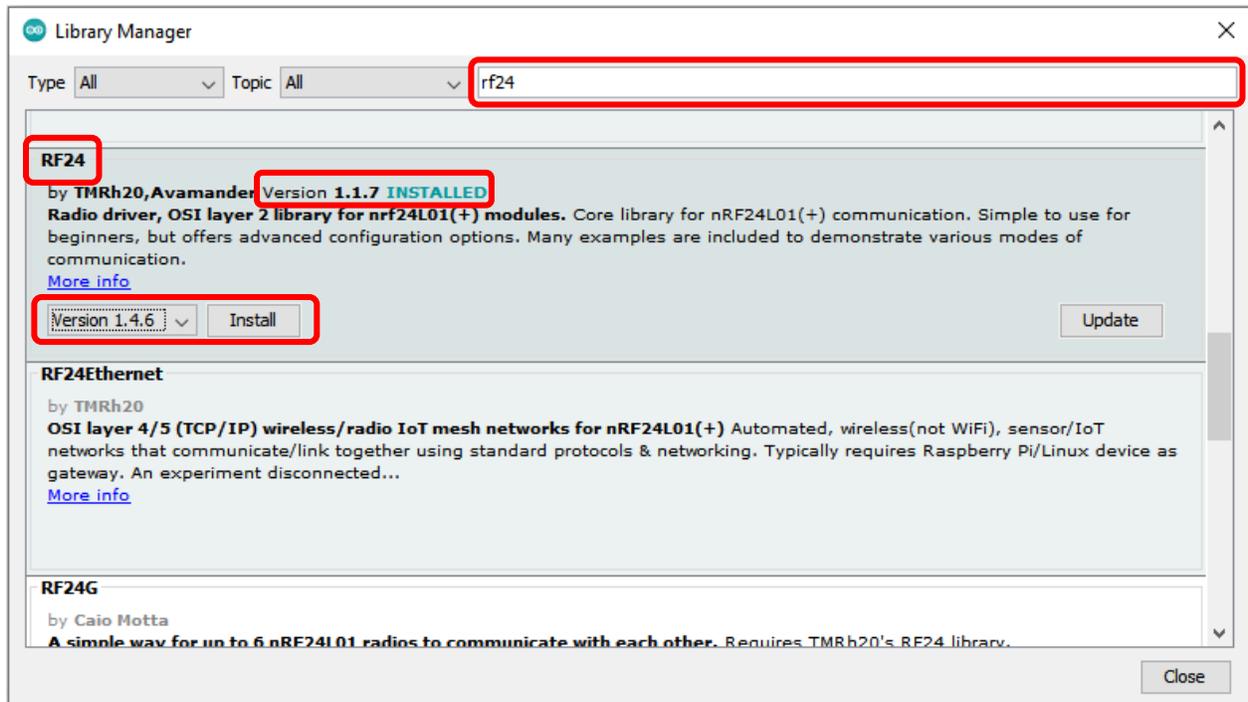
**Connection Table:**

5V external DC power sources are optional, if used with external supply positive terminal goes to VIN.

| Arduino UNO          | nRF24L01 Breakout board | <b>OPTIONAL: 5V DC Adapter</b> |
|----------------------|-------------------------|--------------------------------|
| <b>5V / VIN</b>      | VCC                     | Positive Terminal              |
| <b>GND</b>           | GND                     | Negative Terminal              |
| <b>D9</b>            | CE                      | -                              |
| <b>D10</b>           | CSN                     | -                              |
| <b>D13</b>           | SCK                     | -                              |
| <b>D11</b>           | MOSI                    | -                              |
| <b>D12</b>           | MISO                    | -                              |
| <b>No Connection</b> | IRQ                     | -                              |

## Installing Arduino Library:

In this guide, we will use [RF24-1.1.7.zip](#). User can download it thru Arduino IDE Library Manager or from our share drive. **Latest version are not tested**. Restart Arduino IDE after perform Library installation.

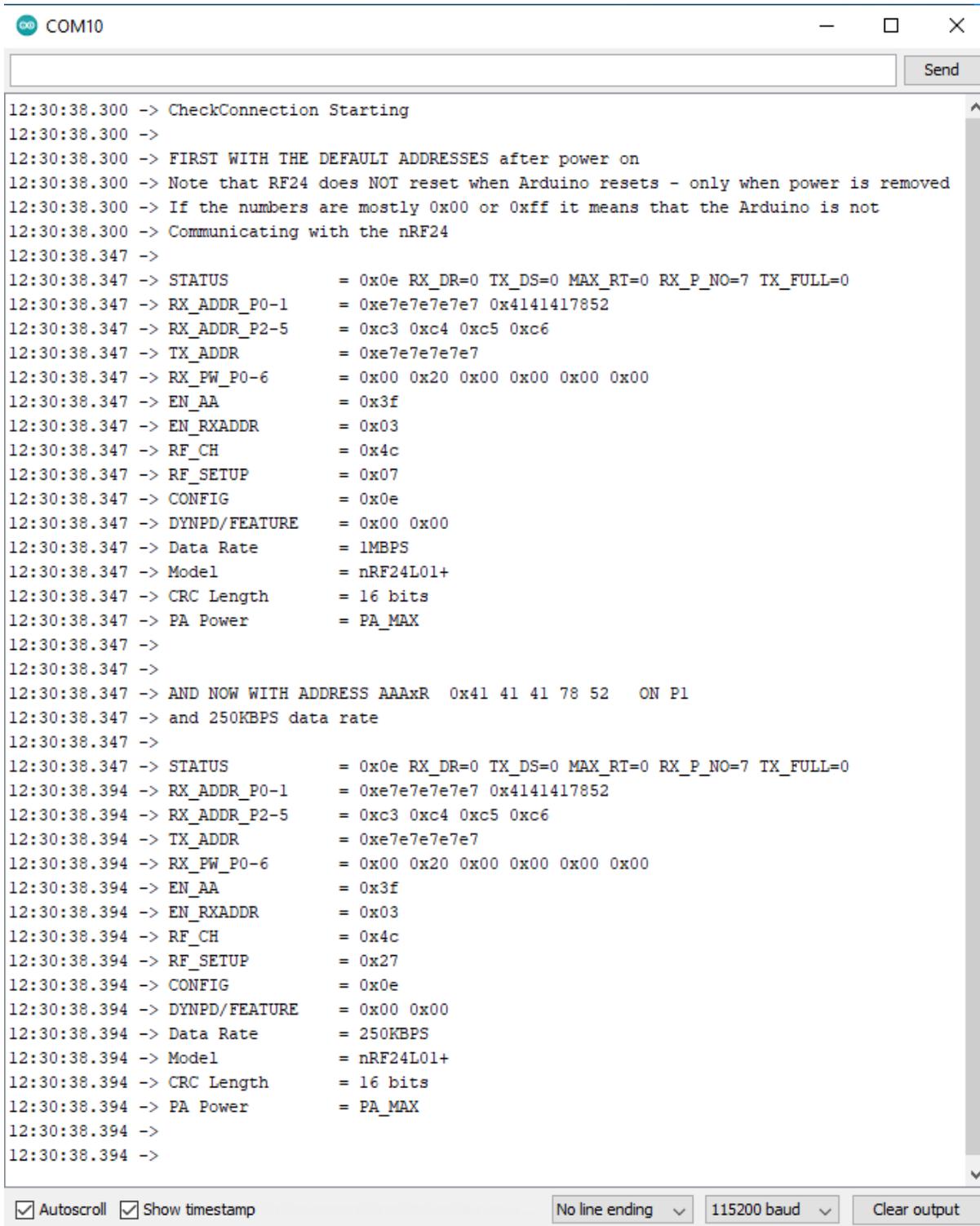


## A. Checking Connection/Setup.

1. **Upload** the provided **Check Connection** code to check whether the module hooked properly. Make sure **Board type** and **COM port** are correct before upload. Upload it on both Arduino.



2. Open **serial monitor** and check the data. Make sure your **baud rate** are set to **115200 bps**.
3. If you got mostly **0x00** or **0xff**, means there might be a **communication problem** between Arduino and nRF module. Check your connection and try again.



The screenshot shows a serial monitor window titled 'COM10'. The window contains a text area with the following text:

```

12:30:38.300 -> CheckConnection Starting
12:30:38.300 ->
12:30:38.300 -> FIRST WITH THE DEFAULT ADDRESSES after power on
12:30:38.300 -> Note that RF24 does NOT reset when Arduino resets - only when power is removed
12:30:38.300 -> If the numbers are mostly 0x00 or 0xff it means that the Arduino is not
12:30:38.300 -> Communicating with the nRF24
12:30:38.347 ->
12:30:38.347 -> STATUS           = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
12:30:38.347 -> RX_ADDR_P0-1      = 0xe7e7e7e7e7 0x4141417852
12:30:38.347 -> RX_ADDR_P2-5      = 0xc3 0xc4 0xc5 0xc6
12:30:38.347 -> TX_ADDR           = 0xe7e7e7e7e7
12:30:38.347 -> RX_PW_P0-6        = 0x00 0x20 0x00 0x00 0x00 0x00
12:30:38.347 -> EN_AA            = 0x3f
12:30:38.347 -> EN_RXADDR        = 0x03
12:30:38.347 -> RF_CH            = 0x4c
12:30:38.347 -> RF_SETUP         = 0x07
12:30:38.347 -> CONFIG           = 0x0e
12:30:38.347 -> DYNPD/FEATURE    = 0x00 0x00
12:30:38.347 -> Data Rate        = 1MBPS
12:30:38.347 -> Model            = nRF24L01+
12:30:38.347 -> CRC Length       = 16 bits
12:30:38.347 -> PA Power         = PA_MAX
12:30:38.347 ->
12:30:38.347 ->
12:30:38.347 -> AND NOW WITH ADDRESS AAAXR 0x41 41 41 78 52 ON P1
12:30:38.347 -> and 250KBPS data rate
12:30:38.347 ->
12:30:38.347 -> STATUS           = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
12:30:38.394 -> RX_ADDR_P0-1      = 0xe7e7e7e7e7 0x4141417852
12:30:38.394 -> RX_ADDR_P2-5      = 0xc3 0xc4 0xc5 0xc6
12:30:38.394 -> TX_ADDR           = 0xe7e7e7e7e7
12:30:38.394 -> RX_PW_P0-6        = 0x00 0x20 0x00 0x00 0x00 0x00
12:30:38.394 -> EN_AA            = 0x3f
12:30:38.394 -> EN_RXADDR        = 0x03
12:30:38.394 -> RF_CH            = 0x4c
12:30:38.394 -> RF_SETUP         = 0x27
12:30:38.394 -> CONFIG           = 0x0e
12:30:38.394 -> DYNPD/FEATURE    = 0x00 0x00
12:30:38.394 -> Data Rate        = 250KBPS
12:30:38.394 -> Model            = nRF24L01+
12:30:38.394 -> CRC Length       = 16 bits
12:30:38.394 -> PA Power         = PA_MAX
12:30:38.394 ->
12:30:38.394 ->

```

At the bottom of the window, there are several controls:  Autoscroll,  Show timestamp, a dropdown menu set to 'No line ending', a dropdown menu set to '115200 baud', and a 'Clear output' button.

## B. Transmitting and Receiving Data.

1. Upload **Simple\_TX** and **Simple\_RX** code to each respective Arduino. Make sure **Board type** and **COM port** are correct before upload.

**UNO A & nRF Module for Transmit data**

**UNO B & nRF Module for Receive data**

Simple\_TX | Arduino 1.8.18

```

1 // SimpleTx - the master or the transmitter
2
3 #include <SPI.h>
4 #include <nRF24L01.h>
5 #include <RF24.h>

```

Simple\_RX | Arduino 1.8.18

```

1 // SimpleRx - the slave or the receiver
2
3 #include <SPI.h>
4 #include <nRF24L01.h>
5 #include <RF24.h>

```

2. Open both **serial monitor**, our Transmitter nRF will send text 'Message (X)', where 'X' are number counting from 0 to 9. On Receiver nRF side, it will display text that received.

COM10

```

12:49:15.988 -> SimpleTx Starting
12:49:18.006 -> Data Sent - Message 0 - Acknowledge received
12:49:20.025 -> Data Sent - Message 1 - Acknowledge received
12:49:21.996 -> Data Sent - Message 2 - Acknowledge received
12:49:24.016 -> Data Sent - Message 3 - Acknowledge received
12:49:26.009 -> Data Sent - Message 4 - Acknowledge received
12:49:28.027 -> Data Sent - Message 5 - Acknowledge received
12:49:30.047 -> Data Sent - Message 6 - Acknowledge received
12:49:32.019 -> Data Sent - Message 7 - Acknowledge received
12:49:34.039 -> Data Sent - Message 8 - Acknowledge received
12:49:36.066 -> Data Sent - Message 9 - Acknowledge received
12:49:38.042 -> Data Sent - Message 0 - Acknowledge received
12:49:40.066 -> Data Sent - Message 1 - Acknowledge received
12:49:42.047 -> Data Sent - Message 2 - Acknowledge received

```

COM5

```

12:49:15.864 -> SimpleRx Starting
12:49:18.023 -> Data received - Message 0 - Acknowledge Sent
12:49:19.995 -> Data received - Message 1 - Acknowledge Sent
12:49:22.011 -> Data received - Message 2 - Acknowledge Sent
12:49:24.032 -> Data received - Message 3 - Acknowledge Sent
12:49:26.040 -> Data received - Message 4 - Acknowledge Sent
12:49:28.011 -> Data received - Message 5 - Acknowledge Sent
12:49:30.033 -> Data received - Message 6 - Acknowledge Sent
12:49:32.050 -> Data received - Message 7 - Acknowledge Sent
12:49:34.024 -> Data received - Message 8 - Acknowledge Sent
12:49:36.050 -> Data received - Message 9 - Acknowledge Sent
12:49:38.073 -> Data received - Message 0 - Acknowledge Sent
12:49:40.050 -> Data received - Message 1 - Acknowledge Sent
12:49:42.078 -> Data received - Message 2 - Acknowledge Sent

```

### nRF24L01+ Automatic Packet Handling Explanation.

1. Transmitter send **data** to Receiver.
2. Once **data** received thru Receiver it will wait for 130 μs.
3. Receiver will sent **Acknowledgement** to Transmitter.
4. Transmitter Receive **Acknowledgement**.
5. Data transmission successful.

