

Nrf24L01-2.4GHz-HowTo

RF24L01 2.4GHz Radio/Wireless Transceivers How-To

[print this page](#)

UPDATES: (Comments,Critique to terry@yourduino.com)

- **PROBLEMS getting nRF24L01 Working: [CLICK HERE](#)**
- [Pinout for TMRh20 Library](#) usage added. See table below
- Example sketches:
 - [Getting Started Example from the RF24 Library](#)
 - [Bare Minimum Example](#)
 - [Transmit Joystick data to another Arduino/nRF24L01 to run Servos Updated](#)

Having two or more Arduinos be able to communicate with each other wirelessly over a distance opens lots of possibilities:

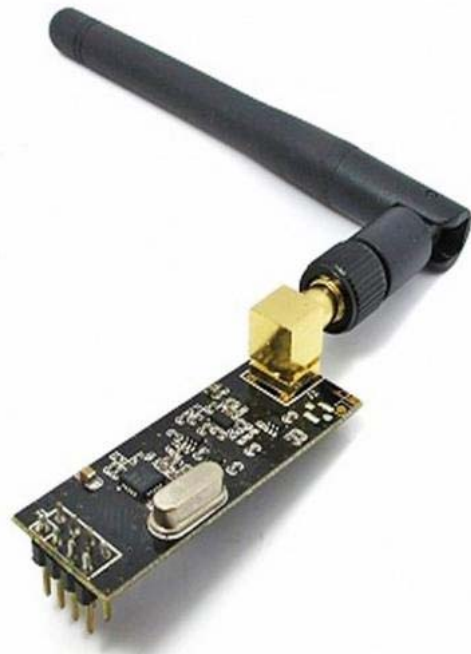
- Remote sensors for temperature, pressure, alarms, much more
- Robot control and monitoring from 50 feet to 2000 feet distances
- Remote control and monitoring of nearby or neighborhood buildings
- Autonomous vehicles of all kinds

These are a series of low-cost 2.4 GHz Radio modules that are all based on the Nordic Semiconductor nRF24L01+ chip.

[\(Details\)](#) The Nordic nRF24L01+ integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed SPI interface for the application controller. The low-power short-range (50-200 feet or so) Transceiver is available on a board with Arduino interface and built-in Antenna for less than \$3! There is also a high-power version (right).

nRF24L01 Modules (Left to Right)

- With built-in Antenna
- Base module with voltage regulator and bypass capacitors for stability
- With added Transmit Power Amplifier, Low Noise Receiver Preamplifier, External Antenna

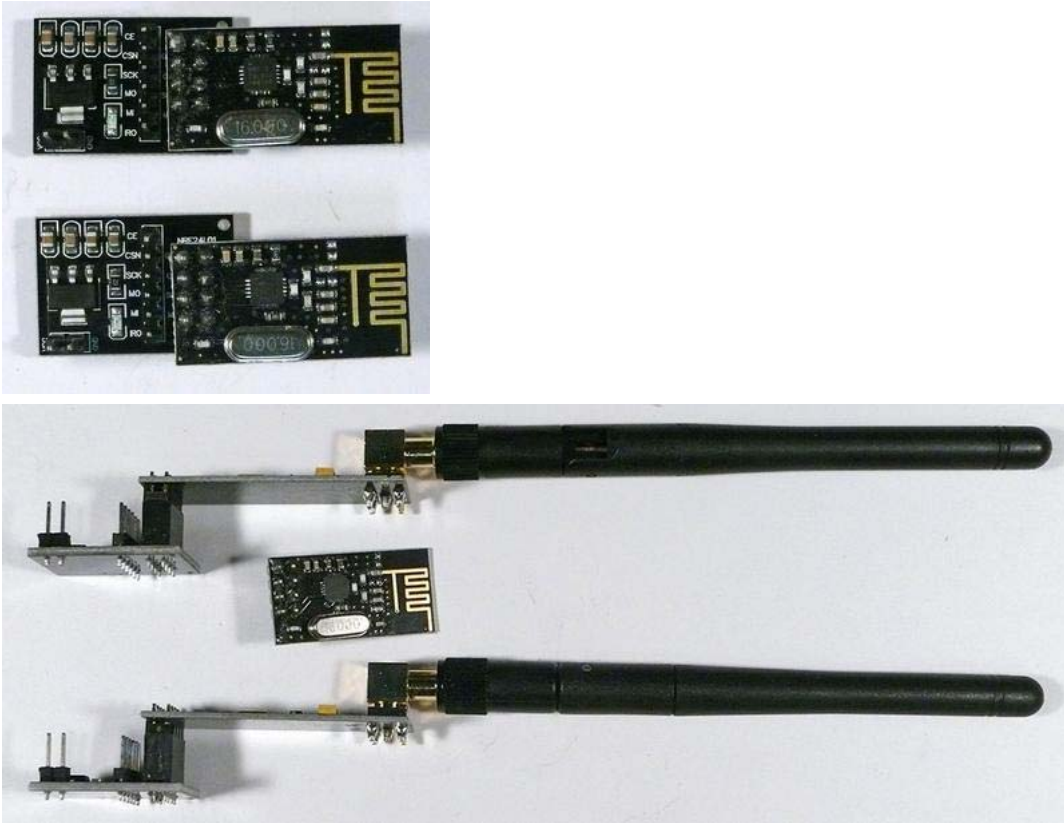


Click images for details and example prices. Widely available on Ebay, Amazon too.

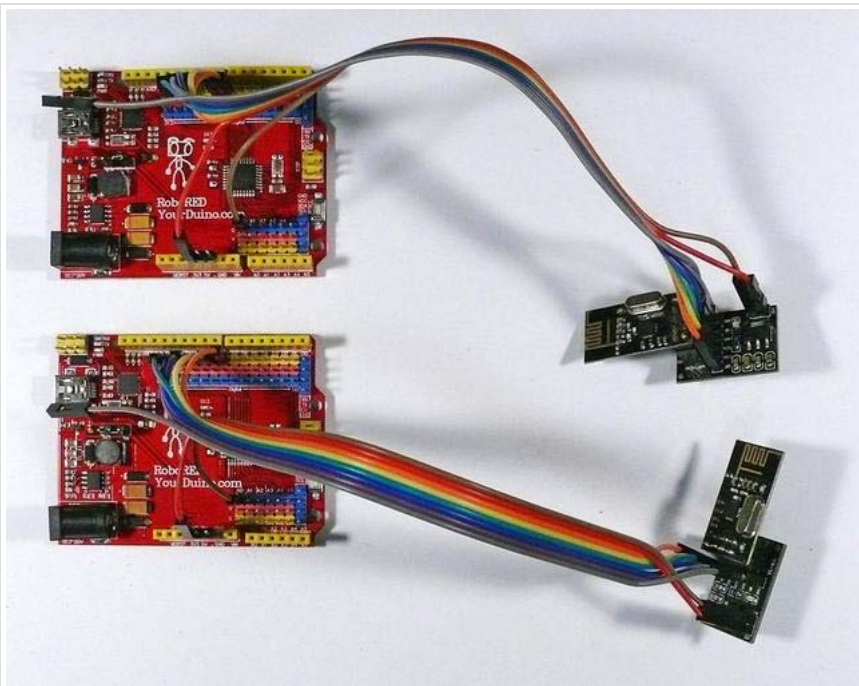
Below are a couple of kits with multiple nRF24L01 versions.

- 3 low-cost nRF24L01 with built-in Antenna, 2 base modules(left)

- 2 High Power nRF24L01 with external Antennas, plus 1 Low-cost nRF24L01, 2 base modules (right)

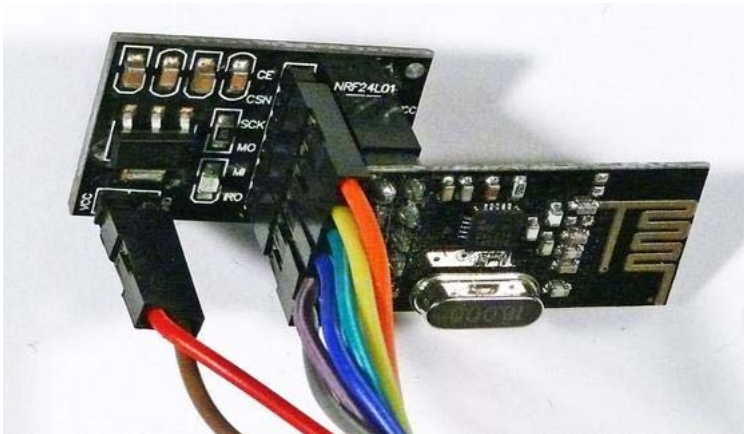
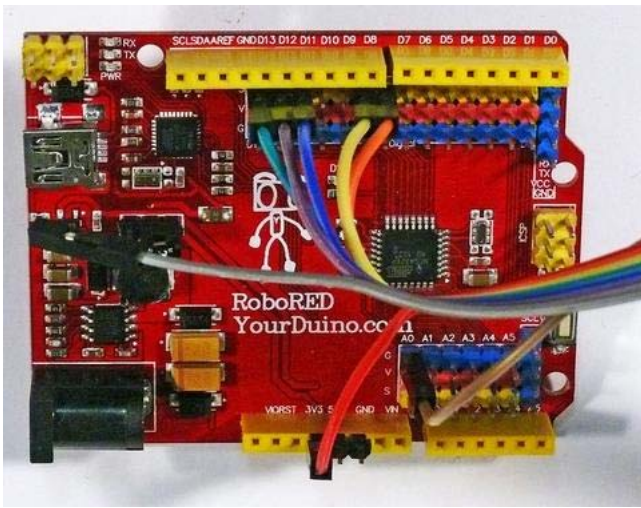


Examples of Arduino to nRF24L01 Connections (Pin Table with colors below):



Low-Power nRF24L01s with Base Modules connected to YourDuino RoboREDS

NOTE! If using the Base Module, power it from 5V not 3.3 like in these photos...



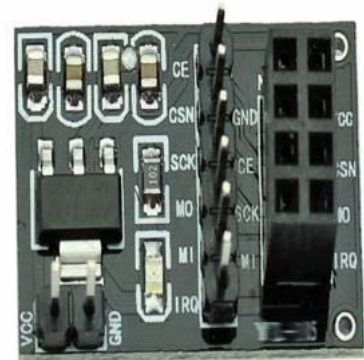
NOTE! If using the Base Module, power it

from 5V not 3.3 like in these photos...

NOTE! Power Problems:

Many users have had trouble getting the nRF24L01 modules to work. Many times the problem is that the 3.3V Power to the module does not have enough current capability, or current surges cause problems. Here are suggestions:

- Use the RF24 Library from TMRH20 (below) and set power low to minimize power requirements: `radio.setPALevel(RF24_PA_MIN);` Space the two radios about a meter apart. After you have things working, and if you know you have enough 3.3V current (up to 250 mA or more) then try higher power. The possibilities are: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX
- Connect a 3.3 uF to 10 uF (MicroFarad) capacitor directly on the module from +3.3V to Gnd (Watch + and - !) [Some users say 10 uF in parallel with 0.1uF is best] This is especially important if you are connecting the module with jumper wires. Or you are using the regular Arduino UNO which provides only 50 mA at 3.3V
- Use a [YourDuinoRoboRED](#) Arduino UNO compatible, which has an added 3.3V regulator (But add a .1 uF capacitor on the radio module).
- There are also nice low-cost base modules like [THIS](#) (RIGHT) that you can plug nRF24L01 modules into. These have a 3.3V regulator built in, AND good power bypass capacitors. You can also find these on Ebay etc. They make it easier to get started and keep operation reliable.
- A separate 3.3V power supply ([Maybe this one?](#))
- If you design a printed circuit board that the module plugs into, add .1uf and 10uf capacitors close to the GND and 3.3V pins. See the schematic of the base board shown on the right on [this page](#). as an



example.

These instability problems are particularly noticeable when 3.3V power comes from a UNO, MEGA, Nano etc. that has only 50 ma of 3.3V power available. Newer boards like the [YourDuinoRoboRED](#) have 350 ma or more available and can run even the high-power modules directly.

nRF24L01 SOFTWARE AND LIBRARIES: (Jump Ahead IF you're ready for that...)

Range

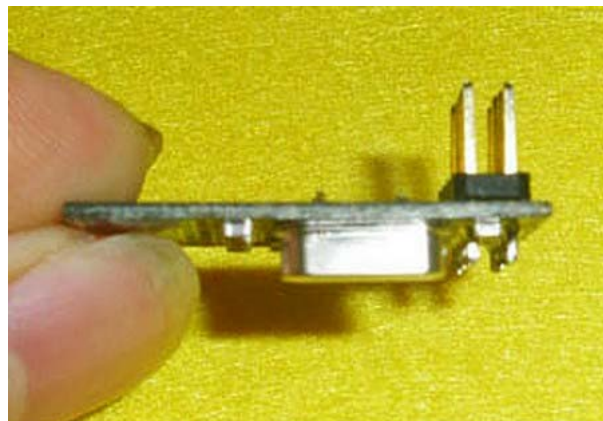
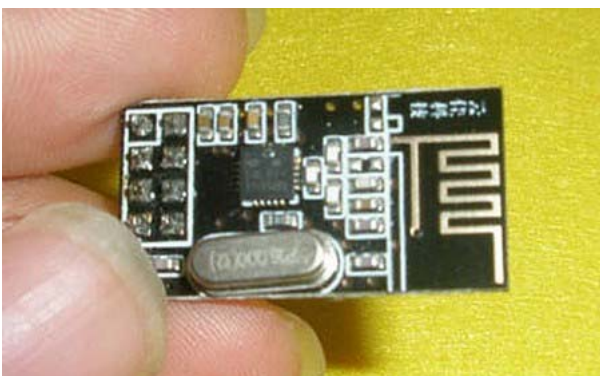
Range is very dependent on the situation and is much more with clear line of sight outdoors than indoors with effects of walls and materials. The usual distance quoted by different suppliers for the low-power version module with the single chip is 200 Feet or 100 Meters. This is for open spaaaaaaace between units operating at a Data Rate of 250KHz. Indoors the range will be less due to walls etc... The example with `radio.setPALevel(RF24_PA_LOW)`; will be only 10 feet or so. But reliable. You can do these things to get better range:

1. Make sure you have good 3.3V power (not just plain UNO, Mega etc. on USB power). You can use a separate 3.3V supply, or a Base Module powered from 5V that has a 3.3V regulator, or a YourDuino RoboRED.
2. **After** you have **good** 3.3V power, set the RF "Power Amplifier Level" to MAX. Like this:`radio.setPALevel(RF24_PA_MAX)`; Or try intermediate settings first: `RF24_PA_MIN`, `RF24_PA_LOW`, `RF24_PA_HIGH` and `RF24_PA_MAX`
3. Set `radio.setDataRate(RF24_250KBPS)`; *Fast enough.. Better range*
4. Set `radio.setChannel(108)`; 2.508 Ghz - Above most Wifi Channels

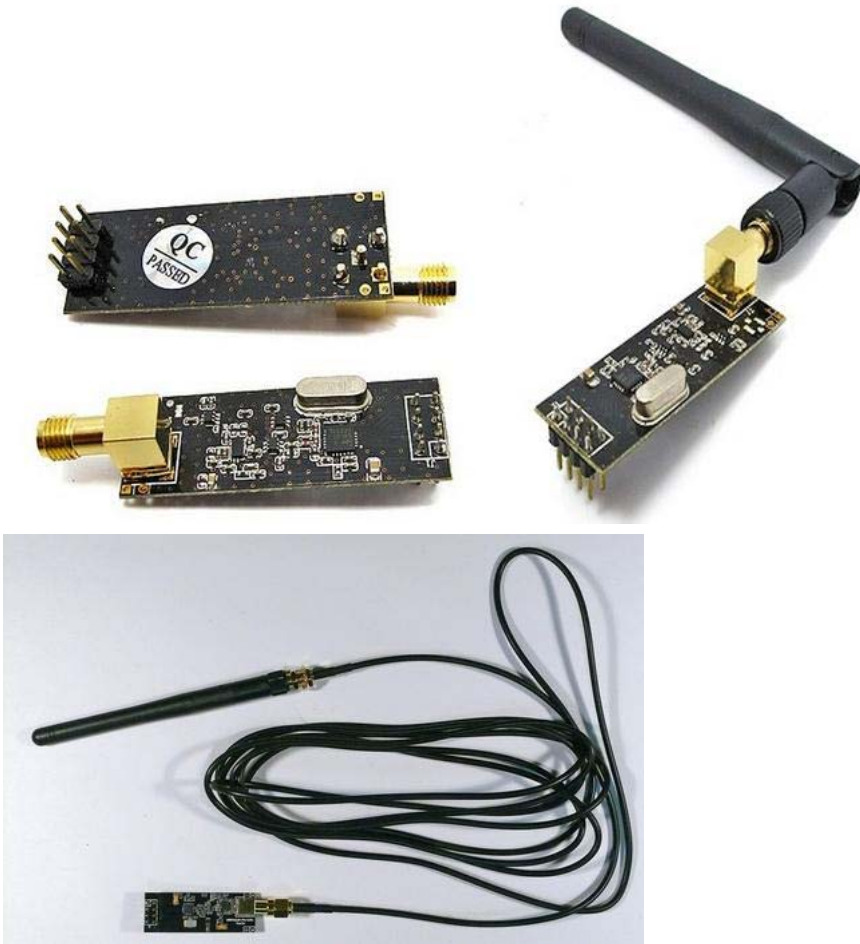
We suggest you test two units at your actual locations before making a decision. There are units with an Antenna Preamplifier for the receiver and transmitter power amplifier and external antenna. The range between that type unit and several low-power units will be better than between two low-power units. Every situation is a little different and difficult to get an exact number without actual tests.

[Link to nRF24L01+ Data Sheet.](#) You don't have to, but if you want to understand more about what you can do with this "little" radio, download the data sheet. In particular you may want to read pages 7-8-9 (For Overview and Features), and page 39 (MultiCeiver, which allows 6 Arduinos to talk to a Primary Arduino in an organized manner). Fortunately the board-level products we have take care of many of the physical and electrical details and Antenna Impedance Matching etc., and this library takes care of lots of register initialization and operational details.

There are other types of nRF24L01 modules which add Transmitter power amplifiers and Receiver preamplifiers for longer distances.. up to 1 Km (3000 feet). [See them all here.](#) These modules use an external antenna which can be a simple directly-attached one or a cable-connected antenna with more gain or directivity. Here's what some of these look like:



Above is the low-power version, with it's built-in zig-zag antenna. On the right you can see the pins sticking down (up in this photo) that connect to Arduino. Later we will show the pinout. **NOTE!! Different Libraries use different pinouts from nRF24L01 to Arduino!!**



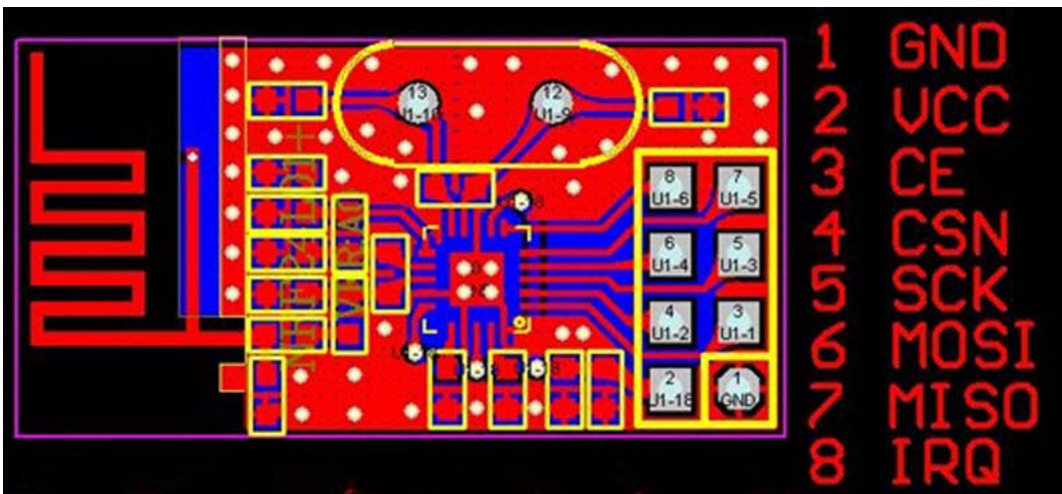
Above is the version with Transmit Power amplifier and Receive Preamplifier. Our low-cost antenna is on the unit in the middle. The same 8 pins connect to Arduino and the base module. Same software is used. On the right is the antenna connected with a 3M cable. Connectors are "Reverse SMA".

Here's a [link to a Home-Brew antenna design](#):

These transceivers use the 2.4 GHz unlicensed band like many WiFi routers, BlueTooth, some cordless phones etc. The range is 2.400 to 2.525 Ghz which is 2400 to 2525 MHz (MegaHz). The nRF24L01 channel spacing is 1 Mhz which gives 125 possible channels numbered 0 .. 124. WiFi uses most of the lower channels and we suggest using the highest 25 channels for nRF24L01 projects.

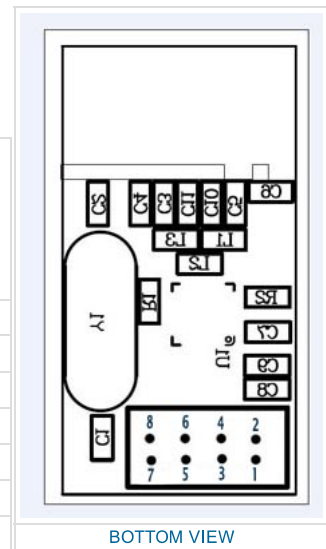
Transceivers like these both send and receive data in 'packets' of several bytes at a time. There is built-in error correction and resending, and it is possible to have one unit communicate with up to 6 other similar units at the same time. The RF24 Network Library extends this to multiple 'layers' of interconnected transceivers.

These amazing low-cost units have a lot of internal complexity but some talented people have written Arduino libraries that make them easy to us. They all use the same pinout as shown in the following diagram, which is a TOP VIEW (Correction!):



Here are details of the Pinout and connections to Arduino (updated):

Signal	RF Module PIN	Cable COLOR	"Base Module" PIN	Arduino pin for TMRh20 RF24 Library	Arduino pin for RF24 Library	Arduino pin for Mirf Library	MEGA2560 pin for RF24 Library	Arduino Pin for RH_NRF24 RadioHead Library	MEGA2560 Pin for RH_NRF24 RadioHead Library
GND	1	Brown	GND	GND	GND *	GND	GND *	GND *	GND *
VCC	2	Red	VCC	3.3 V	3.3V *	3.3V	3.3V *	3.3V *	3.3V *
CE	3	Orange	CE	7	9	8	9	8	8
CSN	4	Yellow	CSN	8	10	7	53	10	53
SCK	5	Green	SCK	13	13	13	52	13	52
MOSI	6	Blue	MO	11	11	11	51	11	51
MISO	7	Violet	MI	12	12	12	50	12	50
IRQ	8	Gray	IRQ	-	2		per library	N/C	N/C



- NOTE!! Most * problems with intermittent operation are because of insufficient current or electrical noise on the 3.3V Power supply. The MEGA is more of a problem with this. Solution: ADD bypass capacitors across GND and 3.3V ON the radio modules **or use the Base Modules** shown above. One user said, "Just Solder a 100nF ceramic cap across the gnd and 3.3v pins direct on the nrf24l01+ modules!" Some have used a 1uF to 10uF capacitor.
- NOTE: Pin 8 IRQ is Unused by most software, but the RF24 library has an example that utilizes it.

The COLOR is for optional color-coded flat cable [such as THIS](#). Photos above show an example.

NOTE: These units VCC connection **must** go to 3.3V not 5.0V, although the Arduino itself may run at 5.0V and the signals will be OK. The NRF24L01+ IC is a 3.3V device, but its I/O pins are 5 V tolerant , which makes it easier to interface to Arduino/YourDuino.

Arduino UNO and earlier versions have a 3.3V output that can run the low-power version of these modules (See Power Problems at the top of this page!), but the high-power versions must have a separate 3.3V supply or use a Base Module with a 3.3V regulator. The [YourDuino RoboRED](#) has a higher power 3.3V regulator and can be used to run the high-power Power Amplifier type module without a separate 3.3V regulator.

nRF24L01 SOFTWARE AND LIBRARIES:

We will show an example of transmit and receive software below, and there are many examples on the RF24 Library download page. You will need a library of software to run the nRF24L01 radios. There has been a progression of improvements to the RF24 libraries by many people. TMRh20 (Who IS he/she, anyway??) has done a great job on the current library and network additions. There are lots of details but you can ignore many of them that the library will take care of.

Get TMRh20's excellent RF24 Library that supports both Arduino and RaspberryPi:

NOTE!! If you have an earlier version of an RF24 Library, COMPLETELY REMOVE IT and install this version.

Read about it and Download it [HERE](#): (Click "Download ZIP" on the upper right of the page)

Read the detailed documentation [HERE](#)

Read his BLOG with advanced projects [HERE](#)

Once you have downloaded the ZIP, you should see a folder called RF24-master.ZIP. Change the name of this file to just RF24.ZIP. Double click on the ZIP and you should see a folder inside also called RF24-master. Rename this to just RF24 as well.

Then see our page about installing libraries [HERE](#):

When you have the library installed, you can run some examples. A ways below we discuss the individual commands (methods) in the RF24 Library and show how to use them to make a simple transmit-receive system. But to get you started we suggest you start with one of the Examples in the RF24 Library from TMRh20. See this page:

<https://github.com/TMRh20/RF24/tree/master/examples> When you install the RF24 library, those examples are loaded on your computer. You don't have to download them.

We suggest you start with, um, "GettingStarted". You need to set up two arduinos, with two nRF24L01's like in the photo above. Separate them by 5 feet or so, though. . You can run two instances of the Arduino IDE on the same machine, with two nRF24L01's on separate USBs with separate COM#. I suggest this sequence:

1. Connect one nRF24L01 to your first Arduino according to the table above (Use the TMRh20 RF24 Library column connections).
2. Start the Arduino IDE .. connect the first Arduino with USB. Find the COM number (Tools>Port) and select it.
3. Load the GettingStarted example (<File>Examples>RF24) and make sure it compiles OK. UPLOAD to your Arduino
4. Open the Serial Monitor (right side if the dark green bar) and SET IT TO 115200 . Close it and open it again. you should see:

```
RF24/examples/GettingStarted
*** PRESS 'T' to begin transmitting to the other node
```

6. Connect the second nRF24L01 to your second Arduino with same connections according to the table above.
7. Start another instance of the Arduino IDE .. connect second Arduino with a second USB cable. Find the COM number (Tools>Port) and select it.
8. Load the example and make sure it compiles OK. UPLOAD to your Arduino
9. Open the Serial Monitor (right side if the dark green bar) and SET IT TO 115200 . Close it and open it again. you should see the Same output.
10. Close the Serial Monitor, look at the code in the second IDE window and CHANGE one line to look like this:

```
/****** User Config *****/
/** Set this radio as radio number 0 or 1      */
bool radioNumber = 1;
```

12. Upload that version to the second Arduino.
13. Now, open both Serial Monitor windows, and type "T" in the top of the first Arduino window.
14. You should see some thing like:

```
RF24/examples/GettingStarted
*** PRESS 'T' to begin transmitting to the other node
```

```
*** CHANGING TO TRANSMIT ROLE -- PRESS 'R' TO SWITCH BACK
Now sending
Sent 523513060, Got response 523511232, Round-trip delay 1828 microseconds
Now sending
Sent 524519544, Got response 524517728, Round-trip delay 1816 microseconds
```

And the other Serial Port window should look like:

```
RF24/examples/GettingStarted
*** PRESS 'T' to begin transmitting to the other node
Sent response 523511232
Sent response 524517728
Sent response 525521620
```

You can reverse the actions by type R in the transmitting side and then T in the opposite side.

OK? You can always revert to this test when things seem to stop working. NOTE: See the "POWER PROBLEMS" section above if nothing works!

You can run some other examples, but when you want to understand the RF24 library and it's Methods better, see the section below.

See: Other Example Sketches at top of this page. Also see the very good example by Robin on [THIS PAGE](#).

The reader will find more details and additional examples here

RF24 LIBRARIES: Information, Documentation, Network, Mesh Network and more! TMRH20 has been busy!

[RF24 Library](#) : Main Page. Download ZIP file at right

RF24 Library detailed [Documentation](#)

RF24 Library [CLASS Documentation](#) (details of a functions and their valid parameters)

RF24 Library [Examples](#) Many examples of the capabilities of the RF24 Library

[RF24 Network System](#) A many-node Network Library

[RF24 library for mesh networking](#) A Mesh Network Library

[RF24 Library for transmitting Audio](#)

RF24 LIBRARIES: Commonly Used Commands (Methods):

NOTE: If you wish, see ALL the details [HERE](#).

We will try to understand the commands to set up and operate an nRF24L01 radio and build a very simple working example.

First we will look at the minimum needed to set up and transmit or receive data. Later we will write more complete examples and get them working with real radios..

CREATE AND START UP A RADIO (For either transmit or receive):

RF24 (uint8_t _cepin, uint8_t _cspin) Create a radio and set the Arduino pins to be used for CE and CS)

EXAMPLE: (Create an instance of a radio, specifying the CE and CS pins.)

RF24 myRadio (7,8); "myRadio" is the identifier you will use in the following examples

NOTE: The following pins are fixed and unchangeable. AND Vcc (supply voltage MUST be 3.3V)
SCK to Arduino pin 13

MOSI to Arduino pin 11
MISO to Arduino pin 12

NOTE: In all following commands, you must use the same identifying name you used in the RF24 statement that created the radio. ("**myRadio**" in the example above) You will use that identifier followed by (dot) followed by the method (command), followed by parameters. This is "object oriented programming". The **OBJECT** in this case is the Radio, and there are many **METHODS** that can operate on the Object. A **METHOD** is much like a Function or a Subroutine.

EXAMPLE: **myRadio.begin();** Start up the actual radio module with the "begin" method

NOTE: "PIPES" : This is often confusing. nRF24L01 uses "pipes" that connect from transmitter to receiver. Pipes have an address you need to set. The Transmitter pipe must have the same address as the Receiver pipe. Later it's possible to use multiple "pipes" at once

EXAMPLE OF RECEIVING DATA:

myRadio.openReadingPipe (1, const uint8_t *address) Pipe number (usually 1), pipe address (which is usually 5 bytes in an array structure).

EXAMPLE:

byte addresses[][6] = {"1Node"}; Create address for 1 pipe.

myRadio.openReadingPipe(1, addresses[0]); Use the first entry in array 'addresses' (Only 1 right now)

myRadio.startListening (); Turn on the receiver and listen for received data. You **MUST** have opened a reading pipe **FIRST**.

```
if( myRadio.available() ) Check for available incoming data from transmitter
{
while ( myRadio.available() ) While there is data ready
{
myRadio.read( &myData, sizeof(myData) ); Get the data payload (You must have defined that already!)
}
myRadio.stopListening(); stop listening
```

EXAMPLE OF TRANSMITTING DATA:

byte addresses[][6] = {"1Node"}; Create address for 1 pipe.

myRadio.openWritingPipe(1, addresses[0]); Use the first entry in array 'addresses' (Only 1 right now)

myRadio.write(&myData, sizeof(myData))

We have written working examples of Transmit and Receive sketches using these methods. [See them HERE](#)

SET SOME OTHER OPERATING OPTIONS AND PARAMETERS:

NOTE: Many of these methods have default values you can usually use.

radio.printDetails();

Prints out a LOT of debugging information.

radio.setDataRate(RF24_250KBPS);

speed RF24_250KBPS for 250kbs, RF24_1MBPS for 1Mbps, or RF24_2MBPS for 2Mbps. 250K Bits per second gives longest range.

radio.setPALevel(RF24_PA_MAX);

Set Power Amplifier (PA) level to one of four levels: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX

The power levels correspond to the following output levels respectively: NRF24L01: -18dBm, -12dBm, -6dBm, and 0dBm

0 dBm is equal to 1 milliwatt. Each 3 dB is a 2:1 ratio. 6 dB is 4:1 10 dB is 10:1 So -18 dBm is 0.0000158489 watts !

To calculate all this dBm Stuff (Which us old Radio Engineers love) See [THIS](#).

The "High Power" nRF24L01 modules [like THIS](#) have a gain of 100, so their output is +20 dBm (100 milliwatts)

radio.setChannel(108);

Which RF channel to communicate on, 0-124 Can operate on frequencies from 2.400GHz to 2.524GHz.

Programming resolution of channel frequency is 1Mhz

This is the same unlicensed band WiFi operates in (WiFi uses 2.400 to 2.500 GHz). Usually frequencies above channel 100 are best.

NOTE: In most countries the allowed frequencies are from 2.400GHz to 2.483.5GHz which you must not exceed. In USA it's best to use channels from 70 to 80.

You can scan the channels in your environment to find a channel that is clear... See [this Scanner sketch](#).

radio.enableDynamicPayloads();

Enable custom length payloads on the acknowledge packets. Ack payloads are a handy way to return data back to senders without manually changing the radio modes on both units.

radio.setRetries(15,15);

Set the number and delay of retries upon failed transmit. Parameters:

delay: How long to wait between each retry, in multiples of 250us, max is 15. 0 means 250us, 15 means 4000us.

count: How many retries before giving up, max 15

radio.setCRCLength(RF24_CRC_16);

length: RF24_CRC_8 for 8-bit or RF24_CRC_16 for 16-bit Cyclic Redundancy Check (Error checking)

NEWS: RaspberryPi is now supported! I need to show a working example... Phew..

RadioHead: A very full-featured Library with support for may different radios, not just nRF24L01:

<http://www.airspayce.com/mikem/arduino/RadioHead/index.html>

Another interesting example: Chat between radios:

<https://github.com/stanleyseow/RF24/tree/master/examples>

[HERE is another good How-To example](#) from Robin on arduino.cc

----- (COPY: Post for people having nRF24L01 Problems) -----

nRF24L01 Intermittent / No operation.

ALWAYS check 3.3V power. Many times intermittent operation is due to power supply regulation issues. Even though the average current may be less than 15ma, apparently there are quick transients when each transmit burst happens. The first examples in the RF24 TMRH20 library run low power and this is less of a problem.

I used to have nRF24L01 problems. Worked one day / one minute, failed the next. Now I put a 0.1uf AND 10uf capacitor right from GND to 3.3V pins on the modules, and things are MUCH better. An excellent solution is the base module shown at the top of this page.

These modules are designed to plug into a baseboard that usually has significant bypass capacitors. If you have them on the end of jumper wires you need extra bypass capacitors.

Try it and let us know here!

----- (END COPY) -----

(<https://www.wikispaces.com/user/view/khbilen>)



Working fine with Nano, not even initialised with Mega.

kbilen (<https://www.wikispaces.com/user/view/khbilen>) Dec 17, 2016



Thanks for this very informative wiki. I followed it step by step; used RadioHead as software library. Installed on Nano (clone), connected cables as shown (for generic Arduino). It worked at once, no problem. Two Nanos can communicate as I like. But I need the server to run on Mega. So, followed the same routine. Connected the pins differently as shown on the table for Mega. It fails at the very beginnng and I receive Init Failed message. OK, may be it needs a cap on 3.3V I said. I soldered 3.3 mF cap on +/- pins directly on the module. No chance, same problem. When I take back it to Nano, it works at once, no problem. On Mega, nothing. Do you think using TMRH20 will help? It looks moe like HW thing to me. Should I keep increasing uFarads up to 100, like one guy did? Should I try using 5V DC, hoping there is more stable current over there?



(<https://www.wikispaces.com/user/view/khbilen>)

kbilen (<https://www.wikispaces.com/user/view/khbilen>)
Dec 17, 2016

OK. Solved. Tried 100uF, and 100uF+0.1uF in parallel. No chance. Then looked backed into the code again. There is one line about Leonardo initialisation; commented out, it says:
RH_NRF24 nrf24(8, 10); For Leonardo, need explicit SS pin
For Nano I was using 8,10 pin connection (not definition) as per table above; so I said let me try explicit definition as:

```
RH_NRF24 nrf24(8, 53);
```

Bingo, everything works fine. For the moment. I

have not done extensive tests, so should not commit on stability, etc. But I can send/receive messages OK.



(<https://www.wikispaces.com/user/view/khbilen>)

khbilen (<https://www.wikispaces.com/user/view/khbilen>)

Dec 17, 2016

OK. Solved. Tried 100uF, and 100uF+0.1uF in parallel. No chance. Then looked backed into the code again. There is one line about Leonardo initialisation; commented out, it says:

```
RH_NRF24 nrf24(8, 10);
```

For Leonardo, need explicit SS pin

For Nano I was using 8,10 pin connection (not definition) as per table above; so I said let me try explicit definition as:

```
RH_NRF24 nrf24(8, 53);
```

Bingo, everything works fine. For the moment. I have not done extensive tests, so should not commit on stability, etc. But I can send/receive messages OK.

(<https://www.wikispaces.com/user/view/MrPaulo2>)



Unable to get radio working... Help? :)

MrPaulo2 (<https://www.wikispaces.com/user/view/MrPaulo2>) Nov 30, 2016

Hi,

Terry - thanks for the tutorial, it is easy to follow and very informative... but I've been unable to make it work. I've resorted to the simplest "Bare Minimum" example and suffer from the same issue each time. The call to

```
myRadio.write( &dataTransmitted, sizeof(dataTransmitted) )
```

always returns 0 (error).

I have the suggested base modules, and so I'm assuming 3V3 power isn't an issue?

I'm using a RobotDyn Arduino Uno R3 (and a RobotDyn Mega2560 as the receiver).

One question that will help me trouble shoot... if there is only one nRF24L01 transmitting and no receiver, should I still expect the write command to not error (or is there some ACK that it looks for). That may help me know if it is the tx or tx&rx combination that isn't working.

I've got printDetails output - but not sure it adds anything:

```
STATUS = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
```

```
RX_ADDR_P0-1 = 0x65646f4e31 0xc2c2c2c2c2
```

```
RX_ADDR_P2-5 = 0xc3 0xc4 0xc5 0xc6
```

```
TX_ADDR = 0x65646f4e31
```

```
RX_PW_P0-6 = 0x20 0x00 0x00 0x00 0x00 0x00
```

```
EN_AA = 0x3f
```

```
EN_RXADDR = 0x03
```

```
RF_CH = 0x6c
```

```
RF_SETUP = 0x01
```

```
CONFIG = 0x0e
```

```
DYNPD/FEATURE = 0x00 0x00
```

```
Data Rate = 1MBPS
```

```
Model = nRF24L01+
```

```
CRC Length = 16 bits
```

```
PA Power = PA_MIN
```

So... where can I look next?

Thanks for any assistance,
Paul.



(<https://www.wikispaces.com/user/view/MrPaulo2>)

MrPaulo2

(<https://www.wikispaces.com/user/view/MrPaulo2>)

Nov 30, 2016

OK - answered my own question with a bit more research...

here:<http://forum.arduino.cc/index.php?topic=421081>

(<http://forum.arduino.cc/index.php?topic=421081>)

(<http://forum.arduino.cc/index.php?topic=421081>) shows that

"Data validation

The nRF24s automatically include sophisticated systems to identify whether the data received matches the data that was sent. If the data is not received correctly the RX will not show data available() and will not send an acknowledgment. That means the TX will consider the transmission to have failed. You will see in the example programs that the TX

is instructed automatically to retry the transmission up to 5 time before giving up. (The max is 15)."

(<https://www.wikispaces.com/user/view/VangNguyen1992>)



Project not running...

VangNguyen1992 (<https://www.wikispaces.com/user/view/VangNguyen1992>) Nov 15, 2016

Hi,
I'm Vang, thank you for your project.
I've followed the steps in your project, but the device can not operating. I use Arduino Uno (pin 7, 8, 11, 12, 13) and RF NRF24L01 + PA LNA 2.4GHz, RF is supplied 3.3V (with 1uF capacitors) from Arduino and RF24 library.
But code not running :
"
Failed, response timed out.
Now sending
failed
Failed, response timed out.
Now sending
"
Can you help me?
Thanks!



(<https://www.wikispaces.com/user/view/LuisTolentino>) LuisTolentino

(<https://www.wikispaces.com/user/view/LuisTolentino>)
Nov 18, 2016

I have the same problem



(<https://www.wikispaces.com/user/view/VangNguyen1992>) VangNguyen1992

(<https://www.wikispaces.com/user/view/VangNguyen1992>)
Nov 27, 2016

Can anyone help me !!!

(<https://www.wikispaces.com/user/view/chrisj21>)



Voltage supply dropping too quickly

chrisj21 (<https://www.wikispaces.com/user/view/chrisj21>) Sep 19, 2016

Terry - excellent info on this website. Well done and thanks.

My questions about voltage usage. I've built a transmitter / receiver now (prototype) using the NRF24L01 and breakout board. In the transmitter, I have a 12V 23A size battery, LM7805 voltage regulator to drop to 5V (with 10uF and 0.1uF on voltage in and out, transmitter parts as noted above, atmega 328, capacitors, crystal, reset button and in my case a keypad (to drive a scoreboard).

Problem: The voltage drops from 12V to 10V quite quickly making it inoperative. Any suggestions?

(<https://www.wikispaces.com/user/view/Bokio>)



Capacitor to stabilize 3.3V

Bokio (<https://www.wikispaces.com/user/view/Bokio>) Sep 4, 2016

I have tried several capacitors 100nF, 1uF, 10uF but no response but than I tried a 100uF and ... it worked. I have to try bigger capacitor and maybe they work better but don't give up trying till it works ;)

(<https://www.wikispaces.com/user/view/kroome88>)



thank you TERRY...

kroome88 (<https://www.wikispaces.com/user/view/kroome88>) Sep 3, 2016

nice, thank you for this tutorial... got it working ... just following the instructions... lesson learned: never under estimate "RADIO signal" ... the note on POWER is very important!!!! cheers!!

(<https://www.wikispaces.com/user/view/Sucrac>)



Thank you.

Sucrac (<https://www.wikispaces.com/user/view/Sucrac>) Aug 22, 2016

Just popped on to say thank you. I really appreciate everything you've done here.

(<https://www.wikispaces.com/user/view/science102>)



Controlling servos with potentiometers over nRF24L01

science102 (<https://www.wikispaces.com/user/view/science102>) Jul 25, 2016



I am trying to control 7 servos with 7 different potentiometers (as part of a robot arm - lynxmotion al5d) using the nRF24L01 . What I am trying to do is similar to the arduino "knob" example for servos except with more servos and potetiometers. I do not understand how to incorporate this information into what I need to do. If anyone could please tell me how to do this it would be greatly appreciated since I am not too good at coding. If you could, please try to upload a segment of code as an example as I am not highly familiar with the terminology. Help is greatly appreciated and needed.

(<https://www.wikispaces.com/user/view/remaadhe>)



Sending and receiving data more than one.

remaadhe (<https://www.wikispaces.com/user/view/remaadhe>) Jul 12, 2016



hi sir, i have to control the robot car project. Where the rear wheels in the form of a dc motor is controlled with flex sensors and front wheels in the form of a servo motor controlled by the accelerometer sensor. Robot controller with a car connected with nrf24l01. if one of the sensors to control I can. but if it combines both to control I can not. Both sensors transmit data simultaneously . dc motors and servo motors confused receive data so that its movement is chaotic. How should I fix this? can you help me.? Can I send my program to you and you see.? Please

(<https://www.wikispaces.com/user/view/Tobyku>)



connecting tft touchscreen and nrf24L01+ to arduino

Tobyku (<https://www.wikispaces.com/user/view/Tobyku>) May 12, 2016



Thank you very much for the detailed description on the nrf24L01+, I got it working very vast with this website.

You state that pin 11, 12 and 13 cannot be changed. Can you explain why?

I am asking because I would like to also connect a cheap tft touchscreen, which unfortunately also uses these pins. I would like to power an arduino with a battery (making it portable) and with the screen added it would greatly help me with testing and understanding the NRF24. Transmitting data could commence when pressing the screen and received data could be easily shown.

The touchscreen I mean is this one:

http://www.banggood.com/2_4-Inch-TFT-LCD-Shield-Touch-Board-Display-Module-For-Arduino-UNO-p-940735.html (http://www.banggood.com/2_4-Inch-TFT-LCD-Shield-Touch-Board-Display-Module-For-Arduino-UNO-p-940735.html)

Thanks for your reply



(<https://www.wikispaces.com/user/view/Sucrac>) Sucrac (<https://www.wikispaces.com/user/view/Sucrac>)

Aug 22, 2016

They're SPI pins

<https://www.arduino.cc/en/Reference/SPI>

(<https://www.arduino.cc/en/Reference/SPI>) for more info. (You probably have this information, but this is for others who may read later.)

Contributions to <https://arduino-info.wikispaces.com/> are licensed under a [Creative Commons Attribution Non-Commercial 3.0 License](https://creativecommons.org/licenses/by-nc/3.0/). 

Portions not contributed by visitors are Copyright 2017 Tangient LLC

[TES: The largest network of teachers in the world](https://www.tangient.com/)