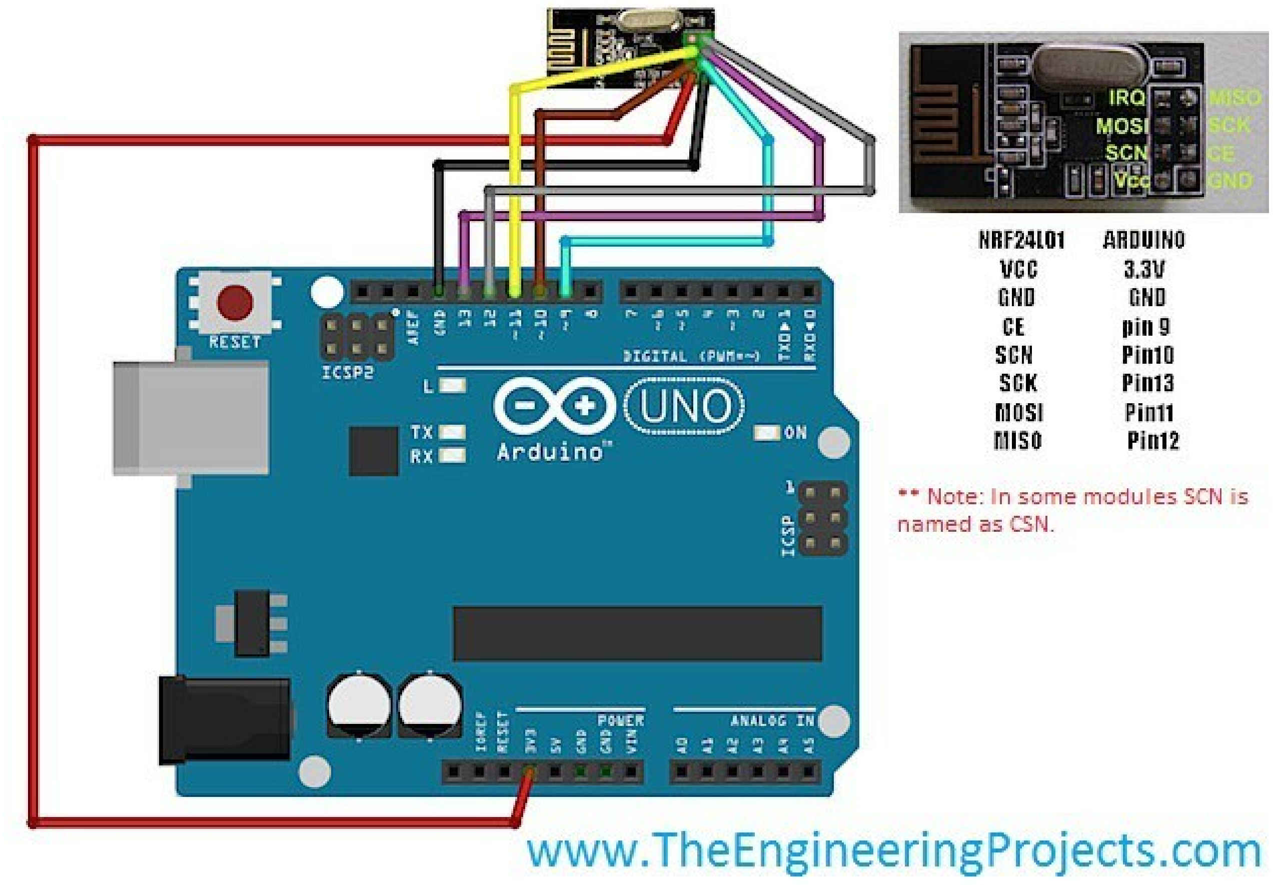
Nrf24l01 - First Video
https://www.youtube.com/watch?v=D40cgHyBLL4&t=61s
First adapt your wire colors by filling the box below….



NRF24L01	ARDUINO
VCC	3.3V
GND	GND
CE	pin 9
SCN	Pin10
SCK	Pin13
MOSI	Pin11
MISO	Pin12

** Note: In some modules SCN is named as CSN.

	MISO 12
MOSI 11	SCK 13
SCN 10	CE 9
3.3V	GND

G13121110 9 8

G 13 12 11 10 9 8

The code uses 7,8 to SCN and CE. Change to use 9,10.

From YOUTUBE - Published on Feb 20, 2016
Do you want to add wireless capability to your Arduino Projects? Using the NRF24L01+ module is a very easy and reliable way to do so. This tranceiver module works at the 2.4GHz band and it is extremely easy to use with any Arduino board, like the Arduino Uno, the Arduino Mega or the Arduino Nano. The cost of module is less than 3$ which makes this module irresistible!

WHERE TO BUY
----

1. NRF24L01: http://bit.ly/NRF24L01B

2. Cheap Arduino Uno: http://bit.ly/Cheap_Uno

3. Powerbank: http://bit.ly/PowerBank_XiaoMi

4. Wires: http://bit.ly/WiresArduino

Full disclosure: All of the links above are affiliate links. I get a small percentage of each sale they generate. Thank you for your support!

In this video we are going to build a simple project just to demonstrate how easy it is to add wireless capability to our Arduino Projects. I have two Arduinos here. This one is sending some data every second, and the other one is receiving the data and displaying it at the serial monitor. As you can see this one way communication is working fine and the range is very good, I can easily get more than 10 meters! The theoretical range that we can achieve is about 100 meters.

The NRF24L01 module is a low cost bi-directional transceiver module. The cost of it is less than 3$! You can find a link for it in the description of the video. It operates at the 2.4GHz band and it can achieve at a data rate of 2Mbits! Impressive isn't it? It uses the SPI interface in order to communicate with Arduino, so it is very easy to use with it. We have to connect 7 of the 8 pins of the module in order to make it work with Arduino.

Unfortunately we can't plug the module in the breadboard so we are going to use male to female wires in order to connect the module to Arduino. Pin number 1 of the module is GND. You have to connect it to Arduino Ground. The next pin is Vcc. You have to connect it to the 3.3V output of the Arduino Uno. Be careful! Do not connect it to 5V or you will destroy your module! The third pin is named CE and you can connect it to any digital pin you like. In this example I am going to connect it to digital pin 7. Pin 4 is CS and you can connect to any digital pin as well. I am going to connect to digital pin 8. The next pin is SCK which goes to digital pin 13 of the Arduino Uno. The next pin is MOSI which goes to digital pin 11 and the last pin in MISO which goes to digital pin 12. That's it!

LIBRARY
----
https://github.com/TMRh20/RF24

CODE OF THE PROJECT **(Use 9,10 instead of 7,8)**
----
http://educ8s.tv/nrf24l01/

ABOUT EDUC8S.TV

Educ8s.tv is a Youtube channel and website which is dedicated in developing high quality videos about DIY hardware and software projects. In this channel we develop projects with Arduino, Raspberry Pi, we build robots and simple electronic circuits. Check out our website as well for more information: http://www.educ8s.tv

Código do transmitter

```
#include <SPI.h>
#include "RF24.h"

RF24 myRadio (7, 8);  // <===== CHANGE 9,10
byte addresses[][6] = {"0"};

struct package
{
  int id=1;
  float temperature = 18.3;
  char  text[100] = "Text to be transmitted";
};


typedef struct package Package;
Package data;


void setup()
{
  Serial.begin(115200);
  delay(1000);
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setDataRate( RF24_250KBPS ) ;
  myRadio.openWritingPipe( addresses[0]);
  delay(1000);
}

void loop()
{
  myRadio.write(&data, sizeof(data));

  Serial.print("\nPackage:");
  Serial.print(data.id);
  Serial.print("\n");
  Serial.println(data.temperature);
  Serial.println(data.text);
  data.id = data.id + 1;
  data.temperature = data.temperature+0.1;
  delay(1000);

}
```

--
Código do receiver
--

```
#include <SPI.h>
#include "RF24.h"
RF24 myRadio (7, 8); // <===== CHANGE 9,10
struct package
{
  int id=0;
  float temperature = 0.0;
  char  text[100] ="empty";
};
byte addresses[][6] = {"0"};

typedef struct package Package;
Package data;

void setup()
{
  Serial.begin(115200);
  delay(1000);
  Serial.println("Receiver");
  myRadio.begin();
  myRadio.setChannel(115);
  myRadio.setPALevel(RF24_PA_MAX);
  myRadio.setDataRate( RF24_250KBPS ) ;
  myRadio.openReadingPipe(1, addresses[0]);
  myRadio.startListening();
}
```

```
void loop()
{

  if ( myRadio.available())
  {
    while (myRadio.available())
    {
      myRadio.read( &data, sizeof(data) );
    }
    Serial.print("\nPackage:");
    Serial.print(data.id);
    Serial.print("\n");
    Serial.println(data.temperature);
    Serial.println(data.text);
  }

}
```

Exercises
1. Write code to measure and show in Serial Monitor (receiver) the number of lost packages (start to count from the first one).
2. Change the data to send multiple package of 1k data. Remove the initial debug messages from the receiver, measure the data rate and print it.
3. Send a IMAGE from one arduino to another. Write a code in Python to receive the data from the Arduino receiver and show the image. The sender get an image from the computer by using the serial port.

Second Example is the Video from Ralph Bacon Send (and Receive) with 2 pipes. Execute the code. Exercises: 1) The sender number is used to blink the led from 0 (100ms) to 255 (1 second). The receiver will confirm the number and the sender check it, as well the receiver will send another random number to the sender show it and it also send back to the receive to ack it. The sender has no blink led. The receiver led to blink could be connected to pin 7.

---

2) Connect a Ultrasonic sensor or a DHT sensor to the sender. Then send the value to the receiver node.

Advanced Tutorial #1
https://www.youtube.com/watch?v=61kWj6zu4Uw
1) Explain the SCAN, run it from the library examples Scanner. Which channels are free and which channels are quite busy ?
2) Execute the transfer example from the NRF library to measure the data rate
3) Write a code to send/receive and blink a RGB LED. Create a scale of colors as a function of signal Quality on one of the nodes. Measure the maximum distance in multiple scenarios (indoor and outdoor)
4) Together with another group, Write a code to connect 4 NRF24L01. One is the transmitter and the others have sensors to send data. The sensors could be DHT, Ultrasonic, and ldr.

5) Modify video example (next video) to measure the temperature and send it. As well to set a alarm event. The receiver could send the alarm temperature to start to send data.

Published on Jan 2, 2016
NRF24L01 chips use the 2.4GHz band. This band is used by many other devices like Access Points or Smartphones. In this video I show how to find empty channels and discuss all the prerequisites for a successful communication between two NRF24L01 modules.
In the next video I will continue with the software part: defining data structures, establish a protected to-way communication, and give you some hints for increasing the range of the devices. This communication system is used in my robot to transfer commands in one direction and telemetry data in the other.
I use the Arduino IDE and the built-in RF24 library.
Useful lings are: https://youtu.be/hI4JGDB7WtU
https://youtu.be/BjId_6tlYvE
http://www.himix.lt/arduino/arduino-a...