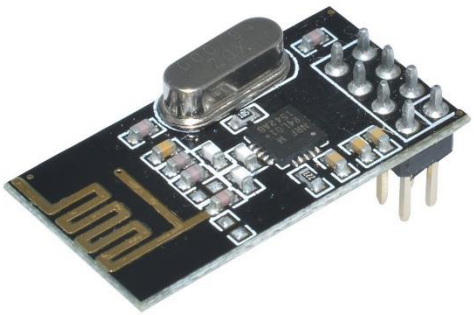
Communication sans fil Arduino - Tutoriel NRF24L01

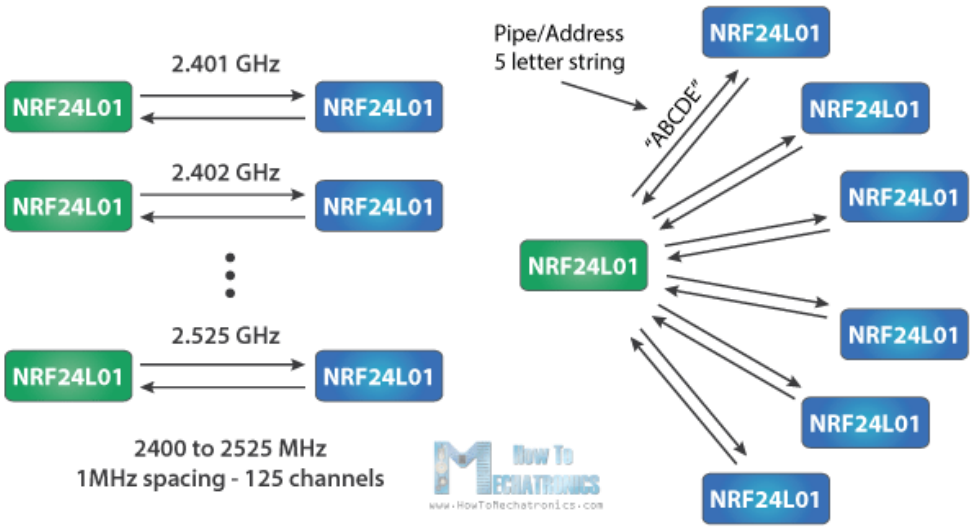
Introduction

Module émetteur-récepteur NRF24L01

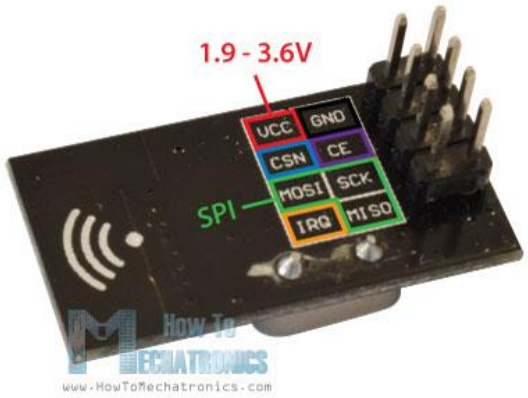
Let’s take a closer look at the NRF24L01 transceiver module. It uses the 2.4 GHz band and it can operate with baud rates from 250 kbps up to 2 Mbps. If used in open space and with lower baud rate its range can reach up to 100 meters.



The module can use 125 different channels which gives a possibility to have a network of 125 independently working modems in one place.

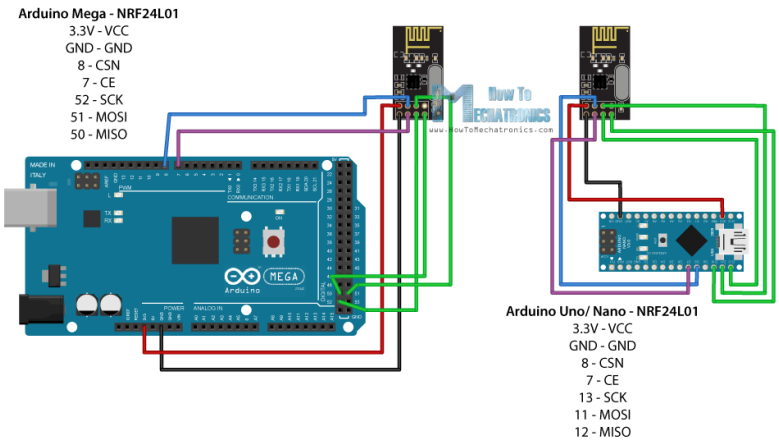


La consommation d'énergie de ce module est d'environ 12 mA pendant la transmission, ce qui est encore plus faible qu'une seule LED. La tension de fonctionnement du module est de 1,9 à 3,6V, mais la bonne chose est que les autres broches tolèrent la logique 5V, nous pouvons donc le connecter facilement à un Arduino sans utiliser de convertisseur de niveau logique.



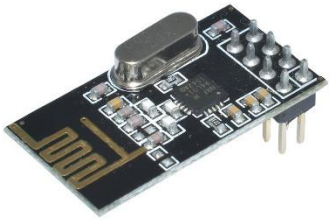
Trois de ces broches sont destinées à la communication SPI et doivent être connectées aux broches SPI de l'Arduino, mais notez que chaque carte Arduino a des broches SPI différentes. Les broches CSN et CE peuvent être connectées à n'importe quelle broche numérique de la carte Arduino et elles sont utilisées pour régler le module en mode veille ou actif, ainsi que pour basculer entre le mode de transmission ou de commande. La dernière broche est une broche d'interruption qui n'a pas besoin d'être utilisée.

Donc, une fois que nous avons connecté les modules NRF24L01 aux cartes Arduino, nous sommes prêts à créer les codes pour l'émetteur et le récepteur.

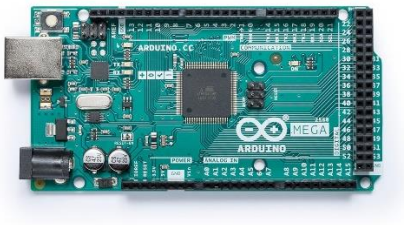


Composants matériels

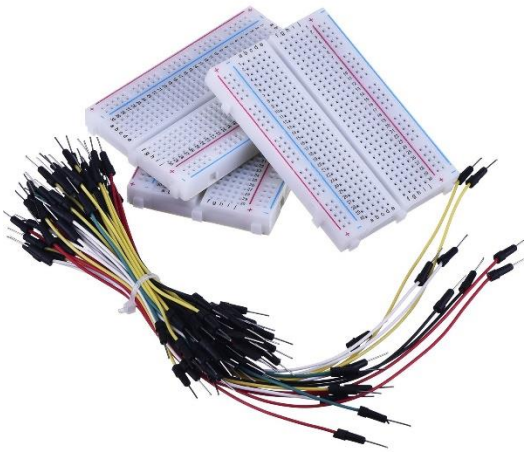
1. **NRF24L01 Transceiver Module**



2. **Arduino Board**



3. **Breadboard and Jump Wires**



Arduino NRF24L01 Codes

Nous devons d'abord télécharger et installer la bibliothèque RF24, ce qui rend la programmation moins difficile.

Voici les deux codes pour la communication sans fil et ci-dessous est leur description.

Code émetteur

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

void setup() {
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
}

void loop() {
  const char text[] = "Hello World";
  radio.write(&text, sizeof(text));
  delay(1000);
}
```

Code récepteur

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE, CSN

const byte address[6] = "00001";

void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
}

void loop() {
  if (radio.available()) {
    char text[32] = "";
    radio.read(&text, sizeof(text));
    Serial.println(text);
  }
}
```

La description:

Nous devons donc inclure le SPI de base et les bibliothèques RF24 nouvellement installées et créer un objet RF24. Les deux arguments ici sont les broches CSN et CE.

```
RF24 radio(7, 8); // CE, CSN
```

Ensuite, nous devons créer un tableau d'octets qui représentera l'adresse, ou le soi-disant canal à travers lequel les deux modules communiqueront.

```
const byte address[6] = "00001";
```

Nous pouvons changer la valeur de cette adresse en n'importe quelle chaîne de 5 lettres et cela permet de choisir à quel récepteur nous parlerons, donc dans notre cas, nous aurons la même adresse à la fois au récepteur et à l'émetteur.

Dans la section de configuration, nous devons initialiser l'objet radio et en utilisant la fonction radio.openWritingPipe (), nous définissons l'adresse du récepteur auquel nous enverrons les données, la chaîne de 5 lettres que nous avons précédemment définie.

```
radio.openWritingPipe(address);
```

De l'autre côté, au niveau du récepteur, en utilisant la fonction radio.setReadingPipe (), nous définissons la même adresse et de cette façon, nous permettons la communication entre les deux modules.

```
radio.openReadingPipe(0, address);
```

Ensuite, en utilisant la fonction radio.setPALevel (), nous définissons le niveau de l'amplificateur de puissance, dans notre cas, je le réglerai au minimum car mes modules sont très proches les uns des autres.

```
radio.setPALevel(RF24_PA_MIN);
```

Notez que si vous utilisez un niveau supérieur, il est recommandé d'utiliser des condensateurs de dérivation entre GND et 3,3 V des modules afin qu'ils aient une tension plus stable pendant le fonctionnement.

Ensuite, nous avons la fonction radio.stopListening () qui définit le module comme émetteur, et de l'autre côté, nous avons la fonction radio.startListening () qui définit le module comme récepteur.

```
// at the Transmitter
radio.stopListening();
```

```
// at the Receiver
radio.startListening();
```

Dans la section de boucle, au niveau de l'émetteur, nous créons un tableau de caractères auquel nous attribuons le message "Hello World". En utilisant la fonction radio.write (), nous enverrons ce message au récepteur. Le premier argument ici est la variable que nous voulons envoyer.

```
void loop() {
 const char text[] = "Hello World";
 radio.write(&text, sizeof(text));
 delay(1000);
}
```

En utilisant le «&» avant le nom de la variable, nous définissons en fait une indication de la variable qui stocke les données que nous voulons envoyer et en utilisant le deuxième argument, nous définissons le nombre d'octets que nous voulons prendre de cette variable. Dans ce cas, la fonction sizeof () obtient tous les octets des chaînes «texte». À la fin du programme, nous ajouterons 1 seconde de retard.

De l'autre côté, au niveau du récepteur, dans la section de boucle en utilisant la fonction radio.available (), nous vérifions s'il y a des données à recevoir. Si cela est vrai, nous créons d'abord un tableau de 32 éléments, appelé «texte», dans lequel nous enregistrerons les données entrantes.

```
void loop() {
  if (radio.available()) {
    char text[32] = "";
    radio.read(&text, sizeof(text));
    Serial.println(text);
  }
}
```

En utilisant la fonction radion.read (), nous lisons et stockons les données dans la variable «texte». À la fin, nous imprimons simplement du texte sur le moniteur série. Donc, une fois que nous avons téléchargé les deux programmes, nous pouvons exécuter le moniteur série sur le récepteur et nous remarquerons que le message «Hello World» est imprimé chaque seconde.