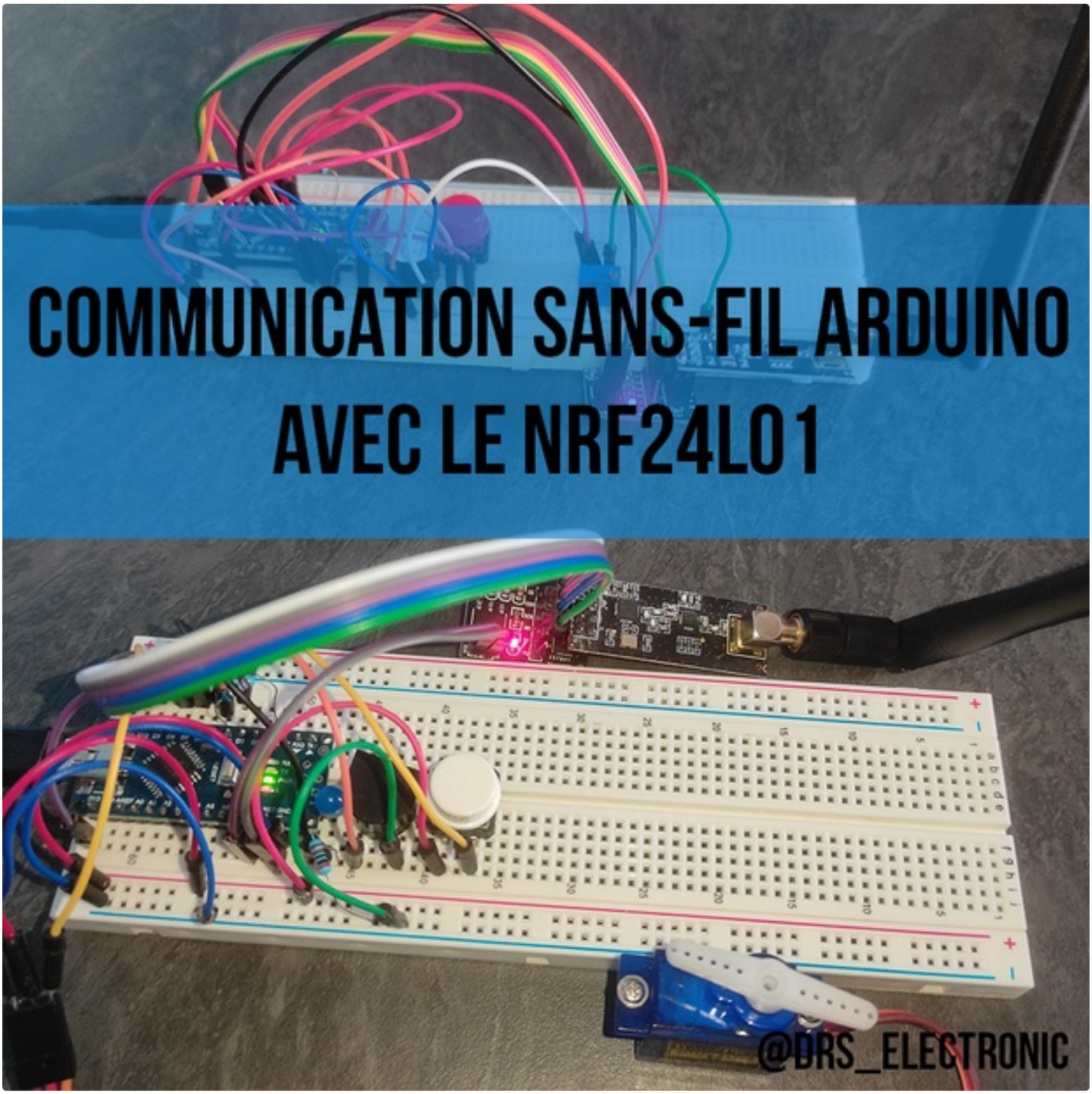
Faire De La Communication Sans-fil Arduino Avec Le NRF24L01

by DRS_electronic

Salut à tous dans cet instructable je vais vous présenter comment faire de la communication sans fil avec Arduino avec le module de communication sans-fil NRF24L01,pour vous montrer les capacités de ce dernier voici un exemple qui est complet, qui permet d'envoyer différent type d'informations comme des valeurs analogiques avec le potentiomètre ou autre entrée du type Tout ou Rien TOR qui envoie des informations digitales tel que les boutons poussoir sur ma breadboard, le tout fonctionnant en full-duplex c'est-à-dire communiquer simultanément dans les deux sens, je vais vous présenter comment réaliser cela mais avant intéressons-nous à ce module NRF24L01.

Comme d'habitude cette instructables est aussi en format vidéo :vidéo

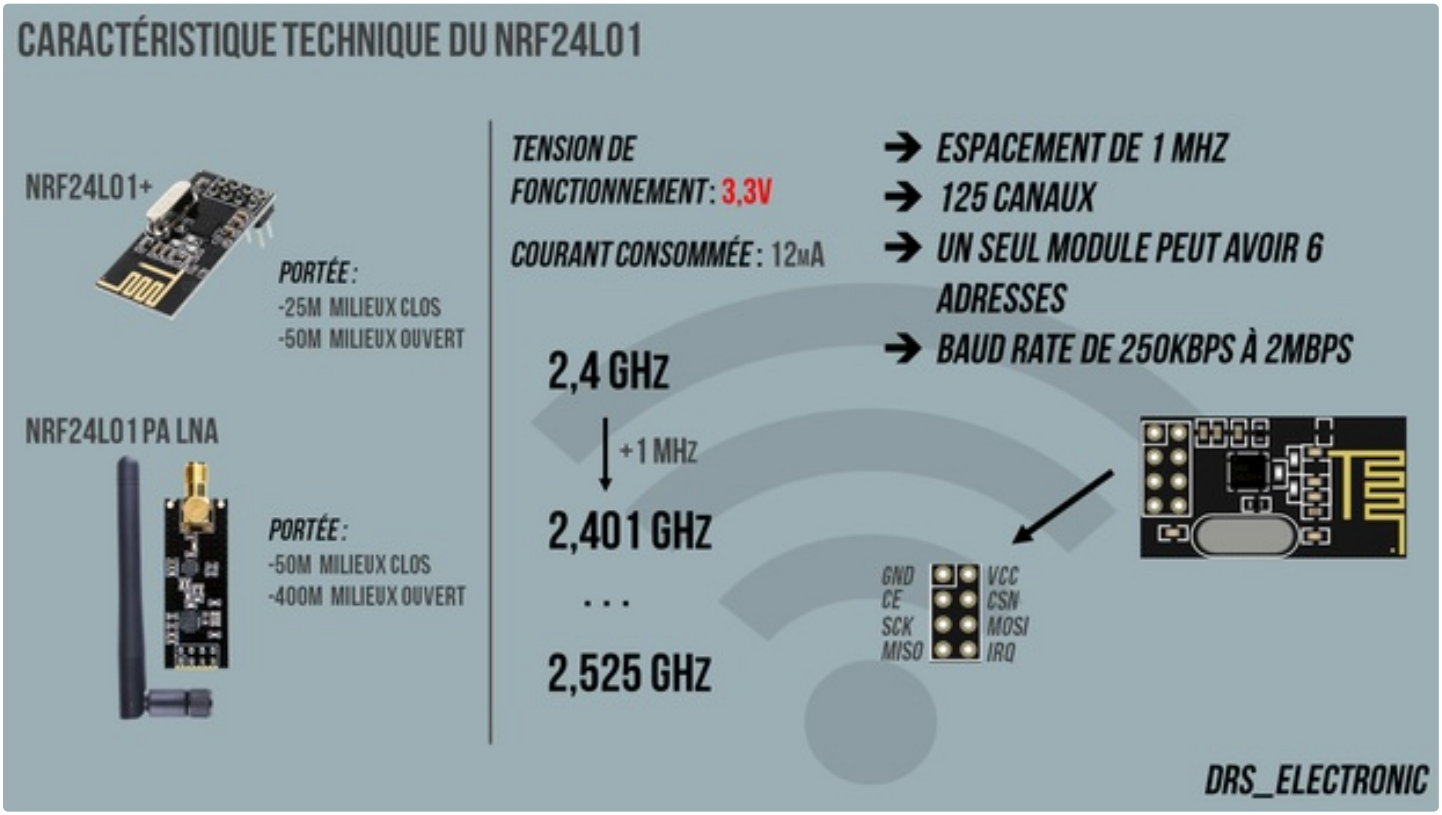


https://youtu.be/WKwQMj6M-hM

Step 1: Présentation Du Module NRF24L01

Le module NRF24L01 est un émetteur-récepteur de faible puissance d'émission qui communique des informations via la bandes ISM de 2.4GHz tel que la WiFi, il dispose de 125 Canaux avec un espacement de 1MHz ce qui veut dire que le module peut aller de 2.4GHz à 2.525GHz, chaque canal peut disposer de plus de 6 adresses, je vous laisse déjà imaginer les possibilités qu'offre ce module, d'autant plus qu'il consomme très peu de courant de l'ordre de la dizaine du mA voir moins selon les modes d'utilisations , il opère à la tension de 3.3V ce qui veut dire qu'il faudrait un étage d'adaptation de

tension avec l'Arduino car il fonctionne à 5V, ce module communique en protocole de terrain SPI, il existe deux version du NRF24L01, il existe le module NRF24L01 classique il peut atteindre une portée de 50m en lieu clos et la version dont je dispose avec une antenne et amplificateur qui peut théoriquement aller à 1Km, dans les fait c'est 200 à 400m selon l'environnement électromagnétiques ou vous habiter et bien moins en milieux clos.



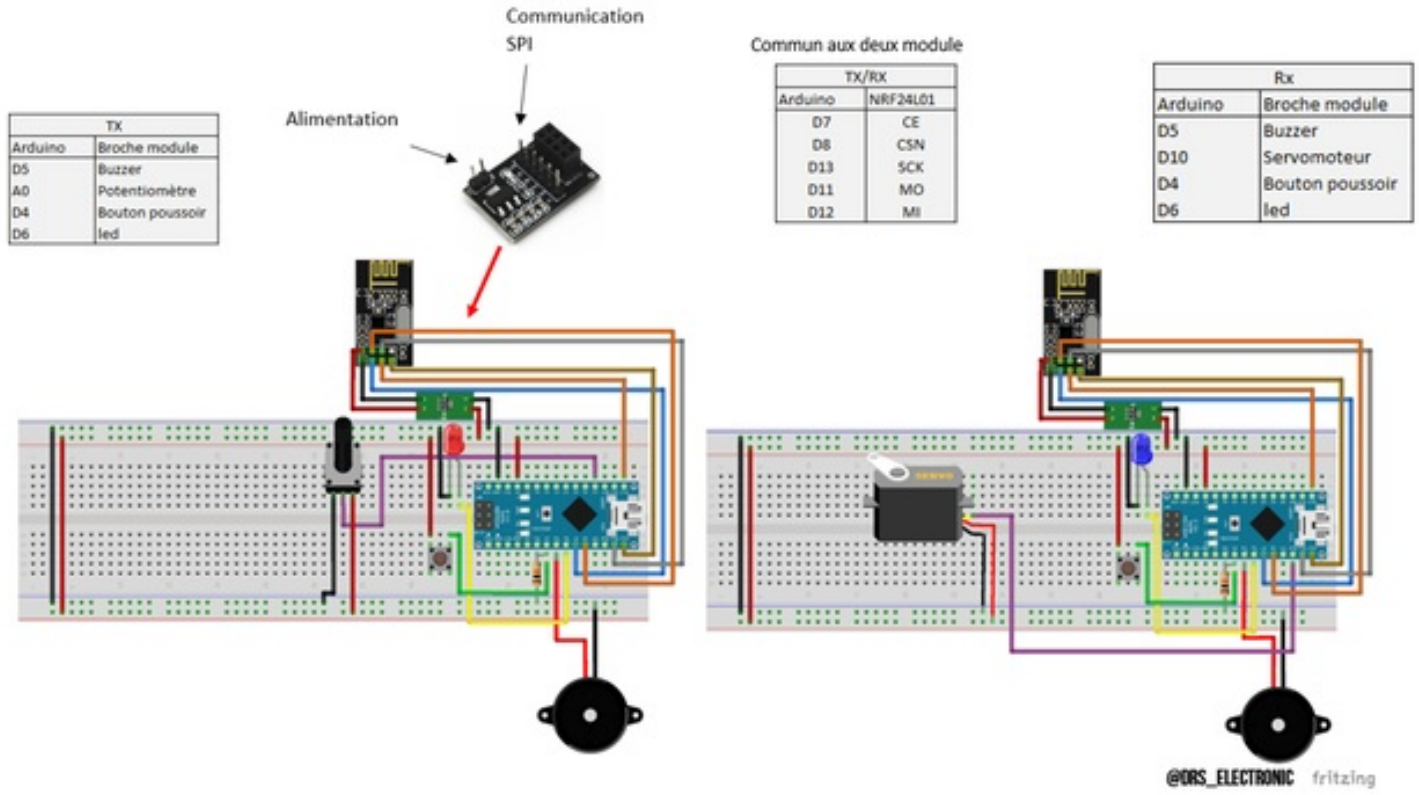
Step 2: Branchement Du Module

Pour la partie câblage il va nous falloir deux module NRF24L01 pour ma part je dispose du nrf24l01 PA LNA qui embarque une antenne omnidirectionnelle de 3db qui à plus de portée que le module classique, alors faite très attention ce module n'opère pas avec les mêmes niveaux de tension que l'Arduino, le nrf24l01 fonctionne à une tension de 3.3V et l'Arduino 5V, pour ce faire il existe mille et une solution mais la plus simple et sûr c'est d'acheter des petits modules qui intègre un LM1117 abaisseur de tension spécialement fait pour le nrf24l01, ça coûte peu chère et pour la fonction réalisé cela vous évite beaucoup de problème, sur le schéma je n'avais pas la représentation du module abaisseur de tension mais l'idée y est, je vous ai mis le screen du module à côté (cf. photo). Pour l'Arduino j'utilise un Arduino nano avec un µC Atmega168 cela fonctionne avec Arduino uno et pro pour les autres versions des µC de la famille AVR il faut se référer au datasheet des µC et de la librairie car les broche de bus SPI ne sont pas les mêmes et certaine librairies ne prennent pas en compte certain µC. Concernant le câblage entre l'Arduino et le NRF24L01 les broches sont communes entre l'émetteur et le récepteur ainsi que la led et le bouton poussoir, pour le module émetteur qui sera différencié par sa led rouges il y aura un potentiomètre de 10k qui va commander le servomoteur de la partie récepteur, pour la partie récepteur toujours le même schéma sauf qu'il y aura un servomoteur qui lui va réceptionner le signal venant de l'émetteur, avec cela vous avec un exemple complet qui envoie et reçoit des informations digitales et analogique le tout en full duplex, bien sûr vous pouvez remplacer les boutons poussoir et potentiomètre par des capteurs ou autre chose de votre souhait tout dépend de votre utilisation, cela peut être une télécommande pour un avion ou voiture télécommandé ou bien un système de sécurité ou de domotique pour n'en citer que quelque un.

Des difficultés à faire fonctionner votre NRF24L01 ?

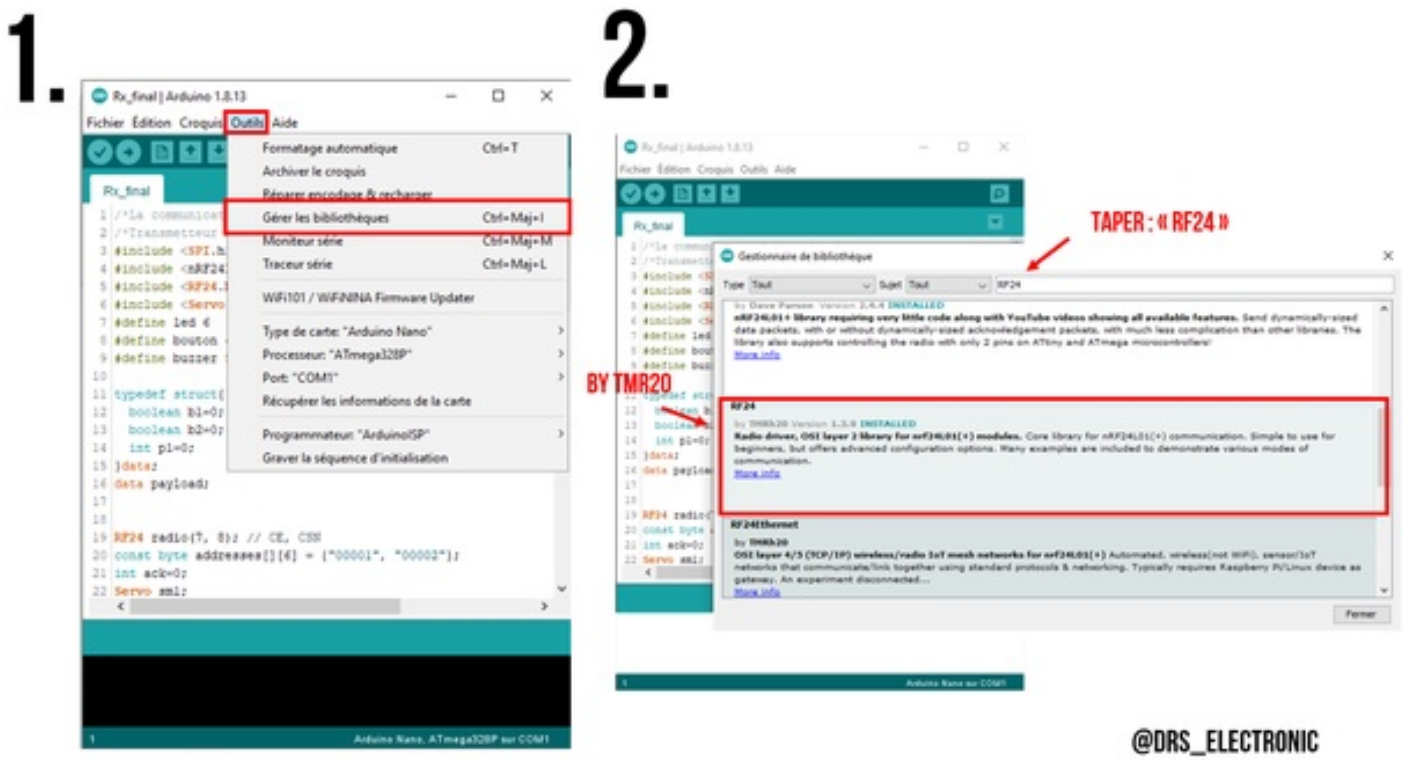
Pour ceux qui rencontrerait des problèmes à faire fonctionner leurs modules vous pouvez rajouter un condensateur de

10µF à électrolyte entre le Vcc et le gnd du module nrf24l01, généralement ce genre de problème disparaît quand on passe sur une carte PCB souder il y a moins de problème de CEM qu'avec une breadboard et également vous pouvez alimenter votre Arduino et module sur une alimentation 5v stabilisé ou simplement le 5V 1A d'un chargeur de téléphone il y aura dix fois moins de perturbation que l'alimentation USB de pc classique (évite les perturbations en mode communs et autres parasites).



Step 3: Installation De La Libraire

Nous allons voir maintenant comment installer la librairie pour utiliser le nrf24l01, alors la librairie que je vais vous présenter a été développé par TMRh20 (cf. lien) c'est une librairie très complète et très bien documenté un travail énorme a été réalisé, je vous laisse parcourir la documentation pour ceux qui souhaitent exploiter d'autre fonctionnalité de la librairie tel que la gestion d'erreur CRC, la modification des canaux de transmission etc, alors il faut aller dans outil/gérer les bibliothèque/ dans la barre de recherche taper « RF24 » et installer la dernière version proposé par « By TMRh20 », fermer et réouvrir Arduino et voilà vous l'avez installé, sans plus tarder passons à la partie code.



Step 4: Code

Pour une question de praticité je vais commenter le code sans détailler les deux programmes et de plus il y à beaucoup de similitudes entre les deux codes mis à part que l'émetteur aura un potentiomètre et le récepteur un servomoteur, **les deux codes sont en pièces jointes.**

Rentrons dans le vif du sujet, il va avoir deux code un pour l'émetteur qui est modélisé par la led rouge et un récepteur avec la led bleu, quand je parle d'émetteur et récepteur c'est pour les différencier mais au final un seul module fait à la fois émetteur et récepteur en même temps vu qu'ils communiquent en full-duplex.

Importation des librairies et déclaration :

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Servo.h>
#define led 6<br>
#define bouton 4
#define buzzer 5
```

Premièrement nous allons importer les libraires nécessaires et définir les broches que nous allons utiliser, pour le récepteur et l'émetteur la led en D6, le buzzeur en D5 et le bouton en D4, ensuite pour l'émetteur nous allons définir le servomoteur en A0.

La structure :

```
typedef struct{
  boolean b1=0;
  boolean b2=0;
  int p1=0;
}data;
data payload;
```

J'ai créé une structure, dedans il va contenir toutes les informations que nous allons souhaiter envoyer simultanément mais également les informations que nous allons recevoir je lui ai donné comme nom payload je reviendrais dessus plus bas.

Initialisation des variables :

```
RF24 radio(7, 8); // CE, CSN<br>const byte addresses[][6] = {"00001", "00002"};
int ack=0;
Servo sm1;
```

Ce coup-ci pour les deux modules ont déclarés les broches CE et CSN du NRF24L01, ensuite ont définis les deux adresses de notre module par défaut 1 et 2 rien de savant, et enfin une variable ack du type integer qui nous sera utile par la suite et pour la partie récepteur il ne faut surtout pas oublier de déclarer le servomoteur.

Setup :

```
void setup() {
  sm1.attach(10);
  pinMode(bouton, INPUT);
  pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(potar, INPUT);
  radio.begin();
  radio.setChannel(125);
  radio.openWritingPipe(addresses[1]); // 00002
  radio.openReadingPipe(1, addresses[0]); // 00001
  radio.setPALevel(RF24_PA_MIN);
}
```

Dans le setup ont déclares les entrées et sorties quoi de plus classique, pour le récepteur ont déclares que le servomoteur est branché à la broche digitale 10 et pour l'émetteur déclarer le potentiomètre en entrées, ensuite on lance la communication avec l'argument begin, ensuite j'ai choisi le canal 125 pour le NRF24L01 ce qui le fera fonctionner à la fréquence de 2.525GHz une fréquence différente de celle utilisé par la Wi-Fi vu qu'il opère sur la bande ISM (fréquence max de la WiFi 2.472GHz pour le canal 13), je n'ai jamais vu sensiblement la différence en terme de qualité d'émission et réception mais cela a toujours fonctionné sans encombre, ensuite on ouvre un writingPipe c'est ce qui permet d'envoyer des informations et un ReadingPire qui permet de réceptionner les informations, ont fait la même chose pour les deux mais en inversant l'adresse. Et enfin très important je fixe le niveau de transmission au minimum car c'est le mode qui est le plus efficace et moins gourmand en courant, cela ne sert à rien de mettre le niveau de transmission au max car les perturbations seront maximales et même le condensateur de 10µF préconisé auparavant ne suffira plus pour faire fonctionner correctement votre module et par-dessus le marché vous constaterez une baisse de portée de votre module qui sera non négligeable à vous de voir.

Loop :

```
void loop() {
  delay(5);
  /****************[ EMIT ]***************************/
  radio.stopListening();
  emitting();
  /*****************************************************/
  delay(5);
  /**************[RECEPTION]************************/
  radio.startListening();
  reception();
  /*****************************************************/
}
```

Il n'y a pas grand-chose dans la boucle j'ai fait le choix de passer par des fonctions pour rendre le code plus clair, pour l'émetteur je fais un startListening qui reçoit les informations et à l'inverse dans l'autre code il faut stopper l'écoute et émettre l'information, et faire la même chose mais dans l'autre sens c'est ce qui va nous permettre d'émettre et de recevoir simultanément l'informations.

Fonction réception :

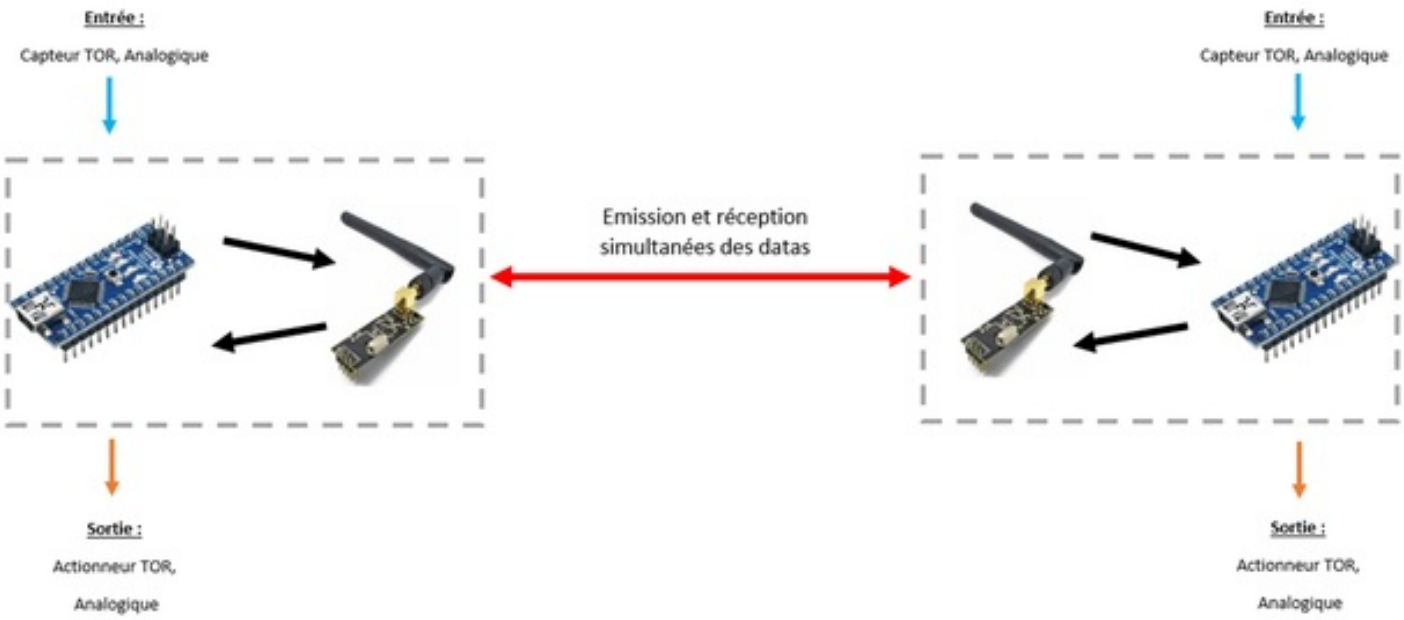
```
void reception(){
 if(radio.available()){
    radio.read(&payload, sizeof(payload));

    if(ack==0){ /*Si la communication est établie alors la fonction startcom est lancé*/
     ack=100;
     startcom();
    }
    map(payload.p1,0,1023,0,255);
    sm1.write(payload.p1);
    if (payload.b1 == HIGH){
      digitalWrite(led, HIGH);
    }
    else {
      digitalWrite(led, LOW);
    }
  }
  else{
    if(ack >1){
      ack = 0;
    }
  }
}
```

Pour la fonction réception avec l'argument available() on vérifie s'il y quelque chose dans le registre de réception du nrf24l01, si oui on lis les informations avec l'argument read en passant en argument la structure payload dont j'ai parlé au début du code, ensuite j'ai rajouté quelque ligne qui sont plus gadgets qui consiste à faire buzer si la communication est établie un peu à la manière des ESC de drone pour ceux qui connaissent, ensuite pour les deux modules ont vérifie si l'état du bouton poussoir de l'autre module est à l'état haut pour allumer la led, pour le récepteur il y à la gestion du servomoteur, ou ont écrit la valeur du potentiomètre dans le servomoteur.

Fonction émission :

```
void emitting(){
  payload.b2 = digitalRead(bouton);
  radio.write(&payload, sizeof(payload));
}
```

Dans la fonction émission c'est très simple ont lis l'états des entrées digitales avec digital Read et Analog Read pour celle analogique tout en pensant à bien l'affecter dans la structure payload les valeurs, la méthode payload permet d'envoyer toutes les informations que l'on souhaite au NRF24L01 en passant qu'un seul argument qui contient tout c'est très utiles et simplifie beaucoup le code, et enfin ont écrit le tout dans la fonction write en passant en argument le fameux payload et puis voilà téléversé le bon code dans le bon module et voilà !



https://www.instructables.com/FPP/M4F4/KJFPTE0E/FPPM4F4KJFPTE0E.ino

Download

https://www.instructables.com/F69/UVLT/KJFPTE0F/F69UVLTKJFPTE0F.ino

Download