



Module NRF24L01 Arduino : caractéristiques, librairies, et explications (tutorial avec exemples de code arduino)

NRF24L01
CARACTÉRISTIQUES,
TUTORIAL
AVEC EXEMPLES
DE CODES
ARDUINO

 PassionElectronique.fr

Envie de faire communiquer des Arduino entre eux ? Ou de commander un drone fait-maison ? Alors rien de tel qu'un émetteur/récepteur **NRF24L01** pour y arriver ! Car avec lui, vous pourrez transmettre des informations par radio, sur la bande des 2,4 GHz, le plus simplement du monde ! C'est d'ailleurs ce que nous allons voir ici, dans ce tuto 😊

Mais avant tout, nous allons **tout d'abord passer en revue toutes les caractéristiques du NRF24L01** (version + et PA LNA), afin de pouvoir vous en servir correctement dans vos futures applications. Par ailleurs, ce tutorial vous montrera comment vous servir de la librairie « nRF24 », disponible dans le gestionnaire de bibliothèque de l'IDE Arduino.

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

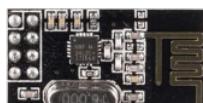
Nous verrons ensuite comment réaliser des émissions et réceptions de données de différentes manières, allant du simple « Hello World » en passant par l'échange de données croisées entre arduino (en quasi « full duplex »), pour arriver à la construction d'un « réseau arborescent », avec la librairie « NRF24 Network ». En bref, que des applications pratiques vraiment sympas, pour apprendre facilement comment se servir d'un NRF24L01 !

Nota : n'étant pas expert en la matière, il se peut qu'il y ait quelques imprécisions ou erreurs, qui se soient glissées. Si tel est le cas, n'hésitez pas à m'en faire part en bas, en zone commentaire, afin que je puisse les corriger, au besoin. N'hésitez pas également à y poser vos questions, si jamais vous avez des interrogations sur telle ou telle partie de ce tutorial ! J'y répondrais dans la mesure du possible, en fonction de mes connaissances, et dès que j'aurais le temps ! Enfin, gardez toujours à l'esprit que ce tuto s'adresse avant tout aux débutants en électronique, voulant intégrer une communication radio sans fil à leur projet, avec des NRF24L01+.

Sommaire [Masquer]

1. Explication rapide, concernant le NRF24
2. Caractéristiques du NRF24L01 (avec ou sans antenne)
3. Comment raccorder le NRF24L01 à l'Arduino (alimentation, bus SPI, chip select, ...)
4. Librairie nRF24 (gestionnaire de bibliothèques IDE Arduino)
5. Tuto NRF24L01 : unidirectionnel (exemple « Hello World »)
6. Tuto NRF24L01 : bidirectionnel (notion de « pipe »)
7. Tutorial NRF24L01 : montage en réseau / arborescence, avec nœuds (tree mode et nodes mesh networking)
8. [Ajout] Câblage d'un NRF24 PA LNA sur Arduino Mega
9. [Ajout] Problèmes fréquents, avec les modules nRF24L01 PA LNA
10. Conclusion

APERÇU



DESCRIPTION

Module NRF24L01 (avec antenne intégrée)

LIEN ACHAT



Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

APERÇU	DESCRIPTION	LIEN ACHAT
	Module NRF24L01 + PA + LNA (avec antenne externe)	

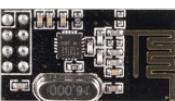
1. Explication rapide, concernant le NRF24

La puce nRF24 est un circuit intégré de chez Nordic Semiconductor (voir la [page documentaire du nRF24L01, avec le datasheet](#)). En fait, il s'agit ni plus ni moins que d'un module d'émission / réception radio, opérant sur la bande des 2,4 GHz (comme le WiFi, ou le Bluetooth). Mais ce qui fait toute sa beauté, c'est qu'il se pilote le plus simplement du monde. En effet, un simple petit Arduino permet d'en prendre le contrôle, via le port SPI.

Sachez que le nRF24 est décliné en 2 versions courantes, actuellement :

- **Le nRF24L01+, avec son antenne intégrée au PCB** (reconnaissable avec ses pistes de cuivre en zigzag)
- **Le nRF24L01+ PA LNA, avec son antenne externe** (raccordable avec un connecteur à visser, type SMA)

Comme vous vous en doutez : la version avec antenne intégrée sera moins performante (ou aura moins de portée, si vous préférez) que la version avec antenne externe. Bien entendu, suivant les applications que vous envisagerez d'en faire, un modèle sera plus avantageux que l'autre. Histoire de vous donner quelques repères, voici les portées approximatives, que l'on peut atteindre avec ces modules NRF24L01, dans la pratique :

APERÇU	MODÈLE	PORTEE EN MILIEU « CLOS »	PORTEE « EN PLEIN AIR »
	NRF24L01+	~ 25 m	~ 50 m
	NRF24L01+ PA LNA	~ 50 m	~ 400 m

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

À noter qu'en théorie, sous certaines conditions (terrain dégagé, sans perturbateurs à proximité, ...), on pourrait communiquer jusqu'à 800 mètres, voir 1,1 kilomètres. Mais en pratique, les conditions idéales ne sont pas toujours réunies. C'est pourquoi je vous ai mis, dans le tableau ci-dessus, des valeurs plus « sûres », afin d'avoir quelque chose de réaliste. Cela étant dit, il faut savoir que **ces distances sont également fonction du débit de communication**. Ainsi, si vous envoyez vos données à 2 mégabits par seconde (2 Mbps), vous irez moins loin qu'à vitesse réduite, de 250 kilo bits par seconde (250 kbps), par exemple.

Sachez toutefois que le constructeur communique sur les valeurs suivantes, dans des conditions optimales :

MODÈLE	DISTANCE MAX À 250	DISTANCE MAX À 1	DISTANCE MAX À 2
	KBPS	MBPS	MBPS
NRF24L01+ PA LNA	1100 m	750 m	520 m

Très important : ces modules nRF24 ont tendance à faire « chuter » la tension d'alimentation des Arduino, lorsque branchés directement dessus. Il faudra donc toujours rajouter un condensateur, entre les fils VCC et GND des NRF24L01, afin de stabiliser leur tension d'alimentation. Sinon, vous risquez fort de vous retrouver avec des bugs de communication inexplicables, ou des fonctionnements erratiques, dus à cela. C'est pourquoi on trouve couramment sur internet des petites cartes « intermédiaires » pour NRF24, permettant de mieux « gérer » leur alimentation.

2. Caractéristiques du NRF24L01 (avec ou sans antenne)

Ici, nous allons voir une à une **les principales caractéristiques du NRF24L01+ (avec antenne intégrée) et du NRF24L01+ PA LNA (avec antenne externe)**. Nous verrons : la fréquence d'utilisation (canaux), la vitesse de pilotage via arduino (dataRate), le niveau d'émission, les tensions d'alimentation et de communication, ... et j'en passe. Comme vous l'aurez compris, il y aura pas mal de choses à voir ici !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

notions, et revenez-y au une autre fois, ultérieurement, lorsque vous aurez besoin de vraiment renforcer certaines de ces connaissances théoriques. Ainsi, tout cela vous semblera moins pénible 😊

2.1 - Fréquence de transmission (2,4 GHz et plus)

Le NRF24L01 est prévu pour fonctionner dans la plage de 2,4 GHz à 2,525 GHz, par « pas » de 1 MHz. Chaque pas de 1 MHz est ce qu'on appelle ici un canal. Du coup, on dispose ici de 126 canaux, permettant de communiquer sur la fréquence de notre choix, entre 2,4 à 2,525 MHz.

N° DU CANAL	FRÉQUENCE DE TRANSMISSION
Canal 0	2,4 GHz
Canal 1	2,401 GHz
Canal 2	2,402 GHz
...	...
Canal 124	2,524 GHz
Canal 125	2,525 GHz

À noter qu'il y a d'autres appareils qui peuvent émettre ou recevoir sur ces fréquences. Je pense notamment au Bluetooth et au WiFi, qui eux aussi fonctionnent dans la bande des 2,4 GHz. Du coup, **il s'agira d'ajuster la fréquence de communication de vos nRF24, en fonction des appareils environnants, si besoin est.**

Concernant le WiFi, il couvre usuellement les fréquences allant de 2,412 GHz à 2,484 GHz. Ceci nous laisse donc de la « place », entre 2,485 et 2,525 GHz pour communiquer sans risque d'interférences avec lui.

Concernant le Bluetooth, il en va de même. En effet, les fréquences comprises entre 2400 MHz et 2483,5 MHz peuvent être utilisées, nous laissant donc « le champ libre » entre 2484 MHz et 2525 MHz.

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

WiFi, ou de toute autre clé ou appareil Bluetooth ! Sinon, gare aux interférences et aux fonctionnements aléatoires !

2.2 - Vitesse de transmission des données, par radio

Le nRF24 permet la transmission de données à vitesse plus ou moins rapide, selon ses besoins, et la portée que l'on souhaite atteindre. Par défaut, nous avons le choix entre trois valeurs possibles :

- Vitesse de 250 kbps (vitesse la plus lente, qui « porte » le plus loin)
- Vitesse de 1 Mbps
- Vitesse de 2 Mbps (vitesse la plus rapide, qui « porte » le moins loin)

Ces valeurs sont exprimées en kbps (kilo bits par seconde), ou Mbps (méga bits par seconde).

Et comme indiqué précédemment, **vouloir émettre à grande vitesse a un inconvénient majeur : une plus faible portée de signal, et un plus grand risque d'erreurs, dues aux interférences.** Il faut donc toujours adapter la vitesse de transmission de données à son besoin, et ne pas chercher à aller trop vite !

2.3 - Niveau du signal

Autre paramètre ajustable avec les puces nRF24L01 : le niveau de signal. En fait, vous aurez le choix entre plusieurs puissances d'émission possibles, allant de très faible, à particulièrement fort. Et le fait d'avoir une antenne intégrée ou une antenne externe fera toute la différence, si vous cherchez à porter loin.

À noter également que certains nRF24 sont dotés d'amplificateur, noté « PA ». Ainsi, si vous voyez marqué « NRF24L01 PA LNA » sur votre module radio, alors vous saurez qu'il s'agit là d'un modèle avec amplificateur de puissance de signal (PA signifiant « Power Amplification »). Accompagné de la mention LNA, signifiant « Low Noise Amplifier », vous aurez là un des meilleurs émetteurs récepteurs qui soit, car hyper puissant, tout en étant le moins sensible possible aux perturbations environnantes.

Concernant notre NRF24L01, voici des valeurs de rapport de puissance que j'ai pu trouver :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

NIVEAU DE SIGNAL	NRF24L01+	NRF24L01+ PA	NRF24L01+ PA LNA
BAS (low)	-12 dBm	-4 dBm	0 dBm
HAUT (high)	-6 dBm	1 dBm	3 dBm
MAXIMAL (max)	0 dBm	4 dBm	7 dBm

Pour rappel, le « dBm » est en quelque sorte un comparateur de puissance, faisant le rapport entre une puissance mesurée et 1 milliwatt. Ainsi :

- 0 dBm correspond à une puissance de 1 milliwatt (soit 1 mW, ou encore, 0,001 W)
- Une augmentation de 3 dB correspond à un doublement de la puissance (soit 2 mW)
- Une diminution de 3 dB correspond à une division de la puissance par deux (soit 0,5 mW)

Bien entendu, si vous souhaitez monter en puissance, faites très attention à ce que votre alimentation soit suffisamment puissante, et bien stabilisée. Sinon, vous risquez de voir apparaître des bugs ou problèmes de communication, comme c'est le cas lorsqu'on essaye d'alimenter ses montages directement sur le 3,3 volts des Arduino.

Enfin, dites nous que côté consommation, le niveau de puissance va forcément influer sur le courant consommé. En effet :

- Un NRF24L01+ consomme environ 14 mA en émission
- Un NRF24L01+ PA LNA peut facilement monter à 140 mA, en transmission

Bien sûr, ce ne sont ici que quelques chiffres indicatifs, qui en réalité seront à pondérer, en pratique (car également fonctions de la quantité de données échangées et de la vitesse d'échange, en plus de l'amplification de puissance).

2.4 - Mémoire du NRF24L01+ : limite à 32 octets par message

Petite parenthèse au sujet de la **mémoire intégrée au nRF24L01** : cette puce n'intègre que **32 octets de mémoire, en émission comme en réception, et par canaux** (sachant qu'un nRF24 peut gérer jusqu'à 6 canaux de communication simultanément). C'est pourquoi vous verrez

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

télémétrie, sans problème ! Qui plus est, rien n'empêche d'envoyer plusieurs messages à la suite, pour compléter l'envoi de données 😊

2.5 - Tunnels de communication (« Pipe ») : sens de communication, et adressage

Comme évoqué précédemment, les nRF24 sont limités à 6 canaux de communication simultanés. Ceux-ci s'appellent des « pipe », en anglais (pour ma part, je les appelle des « tunnel » de communication).

Basiquement, un nRF24 ne peut communiquer qu'avec 6 autres nRF24 au maximum (mais nous verrons comment dépasser cette limite en fin d'article, en créant un réseau de NRF24L01). Mais attention, car sur ces 6 pipes, un seul est utilisable en émission/réception (les autres ne pouvant qu'écouter, et non émettre).

Par ailleurs, il faut savoir que chaque pipe aura une adresse qui lui est propre, codée sur 5 octets (40 bits). Mais que seuls 2 pipes sur 6 sont encodés sur 5 octets. Car les 4 autres pipes n'auront qu'une « adresse » à un octet, qui seront en fait complétés avec quatre premiers octets pris sur le second pipe. Vous n'avez rien compris ? Ne vous inquiétez pas, car tout se résume avec le tableau suivant :

N° DU PIPE	NOM DU PIPE	PEUT ÉMETTRE ?	PEUT RECEVOIR ?	TAILLE DE L'ADRESSE
1	PIPE0	X	X	5 octets (40 bits)
2	PIPE1		X	5 octets (40 bits)
3	PIPE2		X	1 octet*
4	PIPE3		X	1 octet*
5	PIPE4		X	1 octet*
6	PIPE5		X	1 octet*

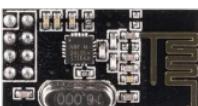
(*) les 4 octets manquants sont pris sur le PIPE1 (car les pipes 1 à 5 partagent leurs 32 premiers bits, pour arriver à 40 bits)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

- Si PIPE2 = 02 (1 octet)
- Et si PIPE3 = 54 (1 octet)
- Alors :
 - l'adresse de PIPE2 sera F1F2F3F402 (les 4 premiers octets de PIPE1, complétés avec « 02 »)
 - l'adresse de PIPE3 sera F1F2F3F454 (les 4 premiers octets de PIPE1, complétés avec « 54 »)

Tout cela vous semble compliqué ? Hum... effectivement, ça l'est un peu ! Mais pour faire simple, dites vous simplement que vous avez la liberté de choisir l'adresse qui vous plait entre 0000 et FFFF pour les PIPE0 et PIPE1, et que pour les autres, ce sera en fonction de PIPE1. Tout simplement 😊

Le saviez-vous ? Chaque tunnel peut prendre le nom qu'il veut (ou presque !), dans la limite de 5 caractères alphanumérique (d'un octet chacun). Par exemple, vous pourriez très bien choisir les noms d'adresses suivants pour vos pipes : « 00001 », « TOTO1 », « 1Node », « ABCDE », « ZYXWV », ou encore 0xAABBCCCDDEELL (en notation hexadécimale, avec le « LL » derrière, signifiant simplement qu'il s'agit d'un « très » grand nombre, dans un programme Arduino). Mais dans tous les cas, pour rappel, les PIPE2 à 5 devront avoir la même « racine » que le PIPE1 (comme vu juste avant).

APERÇU	DESCRIPTION	LIEN ACHAT
	Module NRF24L01 (avec antenne intégrée)	
	Adaptateur d'alim (pour NRF24L01+ et version +PA+LNA)	
	Module NRF24L01 + PA + LNA (avec antenne externe)	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)

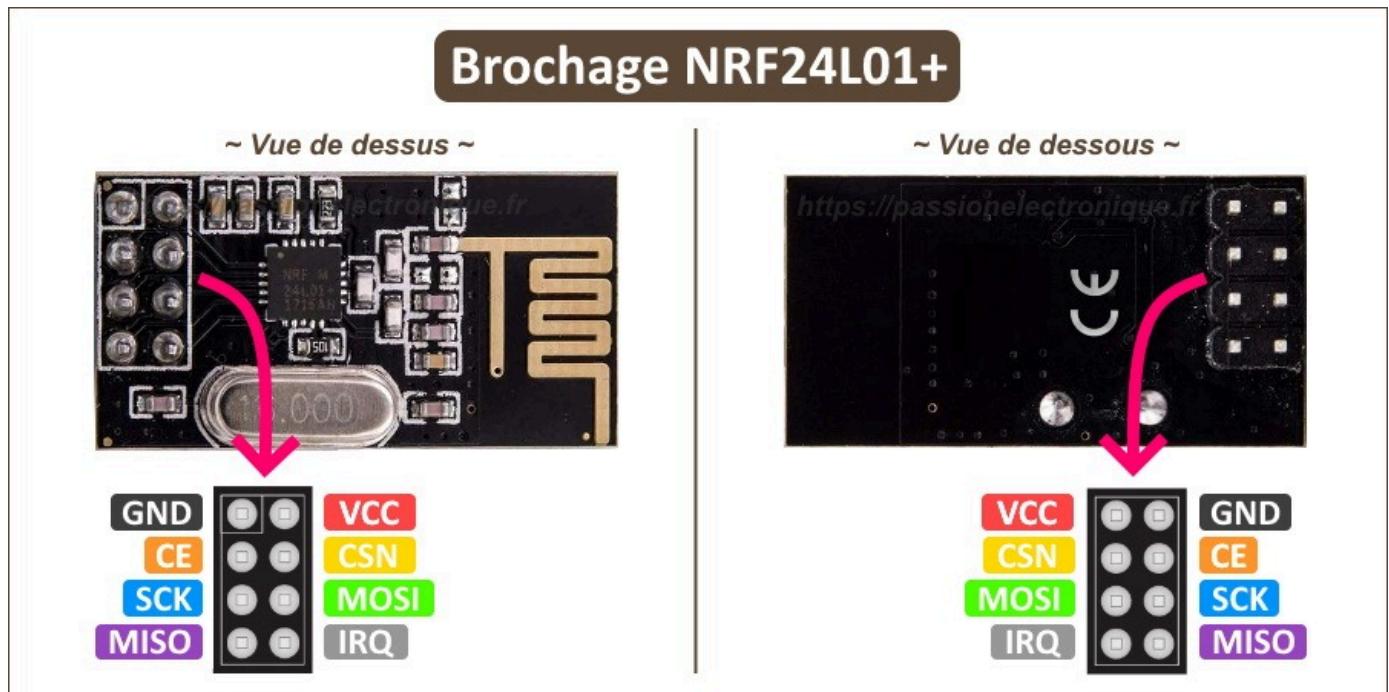
3. Comment raccorder le NRF24L01 à l'Arduino (alimentation, bus SPI, chip select, ...)

Maintenant que nous avons vu la partie « théorique », passons à la pratique ! Ici, nous verrons en détail le brochage des modules NRF24L01+ (dans sa version « compacte », et sa version PA LNA), son alimentation (très importante, sinon vous risquez fort d'avoir des fonctionnements erratiques), et son branchement à l'arduino (que ce soit un Uno, un Nano, ou un Mega).

3.1 - Brochage (NRF24L01+ pinout)

Qu'il s'agisse du « nRF24L01+ » ou du « nRF24L+01 PA LNA », ils sont tous deux équipés d'un connecteur à 8 broches. Celui-ci comprend deux choses : l'alimentation du module en lui-même, et les broches de communication, pour dialoguer avec lui.

Voici comment cela se présente pour le NRF24L01+ (version avec antenne intégrée au PCB) :

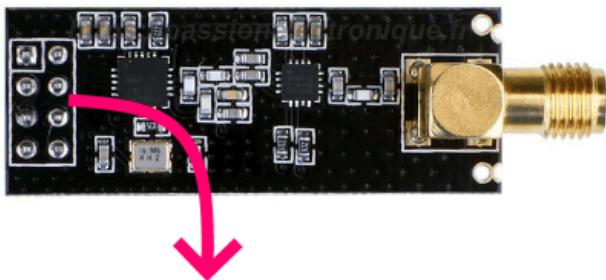


Et pour le modèle NRF24L01+ PA LNA (version avec antenne externe, raccordable sur le connecteur doré) :

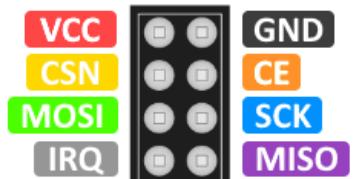
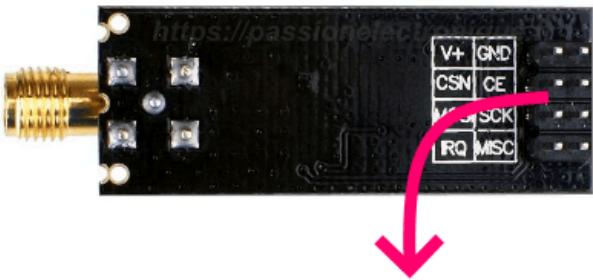
Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Brochage NRF24L01+ PA LNA

~ Vue de dessus ~



~ Vue de dessous ~



Comme vous le voyez, ces deux brochages sont strictement identiques. Par contre, leur « puissance » d'alimentation sera à adapter, en fonction du niveau d'émission, et de la capacité de votre arduino à fournir de l'énergie à ce module.

BROCHE	SIGNIFICATION	RÔLE
VCC	–	Alimentation du module (1,9 et 3,6 V)
GND	–	Alimentation du module (0 V)
SCK	Serial Clock	Horloge communication SPI
MISO	Master In Slave Out	Voie de communication nRF24 -> arduino
MOSI	Master Out Slave In	Voie de communication arduino -> nRF24
CE	Chip Enable	Active le mode RX ou TX (émission / réception)
CSN	Chip Select Not	Active le module nRF24, lorsque mis à la masse (port SPI)
IRQ	Interrupt ReQuest	Permet d'interagir avec un microcontrôleur, si besoin

3.2 - Alimentation électrique du module NRF24L01

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Mais attention, si jamais vous tirez ces 3,3 volts d'un arduino. Car les « pics » de courant des modules nRF24 risqueront fort de faire chuter cette tension, notamment au moment des émissions de données. C'est pourquoi **il faudra à minima rajouter un condensateur de 10 µF, aux bornes d'alimentation du module nRF24, pour lisser sa tension.**

Bien évidemment, si jamais votre puce nRF24 est configurée pour émettre à pleine puissance, alors il vaudrait mieux puiser cette énergie autre part que sur l'alim de votre arduino. L'idéal est d'ailleurs d'utiliser **ces petites cartes vendues dans le commerce, permettant l'alimentation intermédiaires des NRF24L01+**. Ainsi, ils pourront fonctionner sur du 3,3 volts bien stable, pour émettre correctement.

Au final, il faut donc :

- Ajouter un condensateur électrolytique, pour le lissage de la tension (10 à 100 µF, par exemple) -> **idéal si la puissance d'émission est minimale**
- Ou ajouter une « plaquette d'alimentation intermédiaire », spécialement conçue pour gérer l'alimentation des modules nRF24 -> **indispensable si la puissance d'émission est moyenne ou élevée**

Visuellement, voici les montages que vous pouvez réaliser, reflétant ces deux options d'alimentation :



Le saviez-vous ? Même si un NRF24L01+ ne peut s'alimenter qu'avec 3,6 volts tout au plus, vous pourrez toutefois fonctionner en 0V/+5V sur toutes ses autres broches. En fait, seule son alimentation doit respecter ce seuil de tension. Du coup, cela rend parfaitement compatible les modules nRF24 avec nos traditionnels Arduino Uno, Arduino Nano ou Arduino Mega au niveau des broches de communication

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Au niveau du **raccordement du NRF24L01+ à l'Arduino**, rien de plus simple ! Car il suffit simplement de **relier les broches une à une, directement**. À noter toutefois que la pin IRQ est peu souvent utilisée, donc presque jamais câblée, dans les applications simples. En fait, elle n'est utilisée que si vous souhaitez vous servir d'interruption matérielle, au niveau de votre microcontrôleur, par exemple.

Bien évidemment, chaque Arduino a des ports et pins qui lui sont propres. C'est pourquoi, selon le modèle que vous utiliserez (Uno, Nano, Mega, ...), les raccordements ne se feront pas sur les mêmes pins d'entrées/sorties. Mais pour faire simple, voici un tableau récapitulatif, vous montrant où brancher chaque broche du nRF24, sur votre carte microcontrôleur :

ARDUINO	VCC	GND	SCK	MISO	MOSI	CE	CSN	IRQ
Uno/Nano	+ 3,3 V	GND	13	12	11	7	8	n.c.
Mega	+ 3,3 V	GND	52	50	51	7	8	n.c.

Nota : n.c. = non connecté

Concernant les broches CE et CSN, il y a bien d'autres endroit où vous pourriez les brancher. Je pense notamment aux broches D9 et D10, par exemple. Mais avant de faire quel que changement que ce soit, réfléchissez bien à l'usage que vous ferez des autres pins. Par exemple, si vous utilisez un servomoteur piloté via ondes radios avec un nRF24L01+, sachez que la librairie « servo.h » qui permet de les piloter ne permet pas de les brancher sur n'importe quelles broches. Et justement, ceux-ci ne peuvent être « attachés » qu'aux bornes 9 et 10 d'un arduino uno, ou arduino nano. Comme quoi, il faut toujours lister tous ses besoins, avant d'attribuer telle ou telle pin d'entrée ou sortie à tel ou tel périphérique !

IMAGE	DESCRIPTION	LIEN
	Caméra WiFi extérieur, pour vidéosurveillance (modèle double, avec partie fixe + mobile 360°)	Voir le prix

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

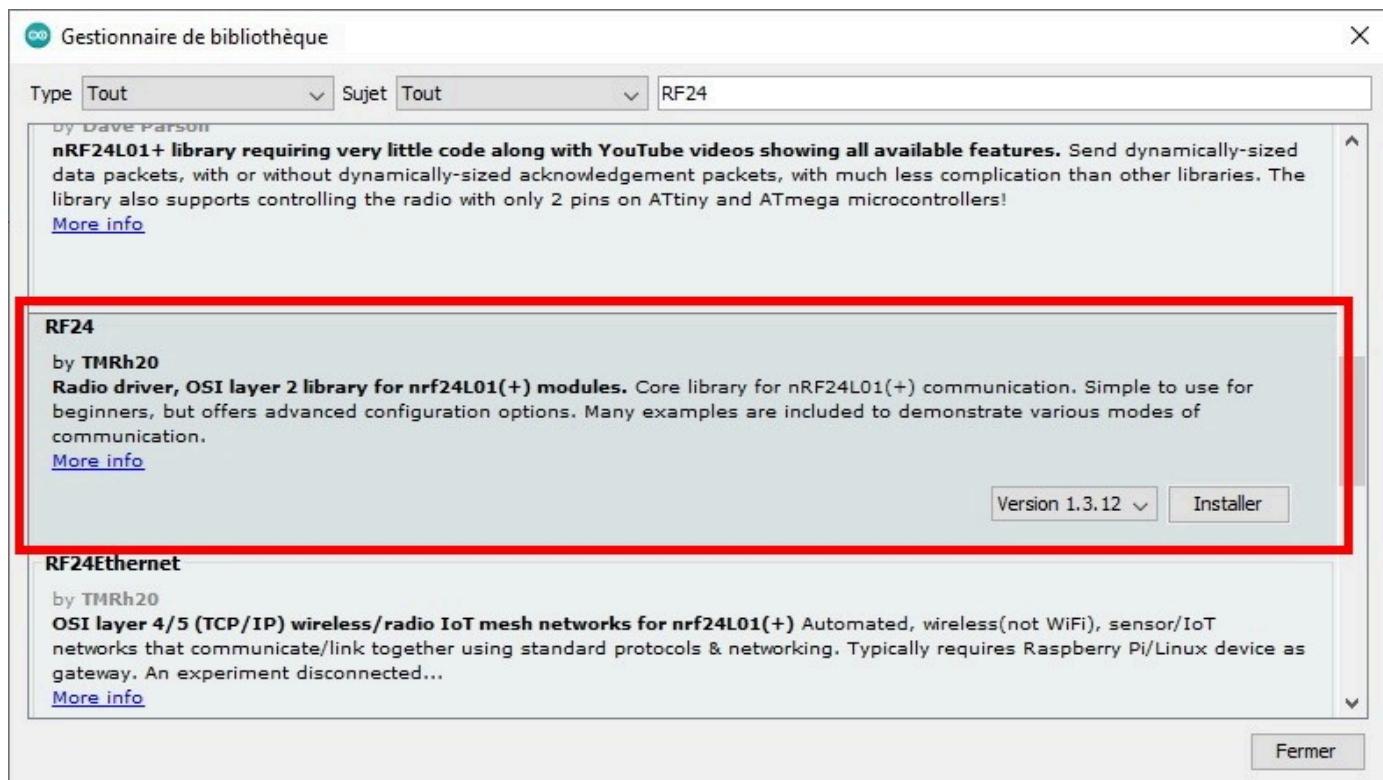
[Accepter](#) [Refuser](#)

Dans cette section, je vais vous parler d'une librairie déjà toute prête, et disponible dans le « Gestionnaire de Bibliothèques » de l'IDE Arduino. Il s'agit de la librairie NRF24 (voir lien [github nRF24 library](#)). Celle-ci contient tout ce dont nous avons besoin pour contrôler les échanges entre nos arduino et nos modules NRF24L01+, et ce, en toute simplicité. Nous verrons ici comment initialiser ce dernier, et les principaux paramétrages possibles. Alors, en avant !

4.1 - La librairie utilisée, pour piloter le NRF24L01+

Avant tout, il sera nécessaire d'installer la librairie nRF24, disponible dans le gestionnaire de bibliothèque, de votre IDE Arduino. Pour cela, il vous suffit d'ouvrir cette fenêtre (depuis Arduino IDE -> Menu Outils -> Gérer les bibliothèques), et de taper NRF24 dans la zone de recherche.

Ensuite, il ne vous restera plus qu'à sélectionner le package ayant comme sous-titre « by TMRh20 », et de cliquer sur « Installer » (attention à ne pas vous tromper avec tous les autres noms de package, comme le nRF24 Ethernet, ou le nRF24 Network, qui sont seulement complémentaires, et prévu pour d'autres usages). Voici d'ailleurs ce que vous devriez avoir à l'écran, avec le « bon » package visible au milieu :



Au niveau de la documentation de cette librairie NRF24, vous trouverez tout plein d'infos intéressantes ici :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

- **Détail des fonctions de la library NRF24** (pour voir toutes les propriétés utilisables, avec cette bibliothèque)

À présent, voyons comment instancier cette librairie, dans nos programmes arduino.

4.2 - Instanciation librairie nRF24 dans Arduino IDE

Pour instancier cette librairie nRF24, rien de plus simple ! Car il suffit d'inclure 2 libraires à tous nos projets de communication radio, à savoir :

- La librairie SPI, nécessaire pour les échanges Arduino <-> nRF24
- Et la librairie NRF24, en elle-même

Cela se fait avec les 2 lignes de code suivantes :

```
1 | #include <SPI.h>
2 | #include <RF24.h>
```

Ensuite, il faut procéder à l'instanciation de la librairie NRF24 en elle-même, en précisant où sont raccordées les pins CE et CSN sur l'arduino (les autres étant « immuables », si je puis dire). Pour ma part, j'ai relié :

- la broche CE à la broche D7 de mon arduino nano
- et la broche CSN à la broche D8 de mon arduino nano

Au niveau du code de programmation, voici comment cela se déroule :

```
1 | #define pinCE    7          // On associe la broche "CE" du NRF24L01 à la sortie
2 | #define pinCSN   8          // On associe la broche "CSN" du NRF24L01 à la sortie
3 | RF24 radio(pinCE, pinCSN); // Instanciation du NRF24L01
```

Enfin, dans la fonction « setup() » de notre programme arduino, il ne reste plus qu'à faire appel à

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

```
1 void setup() {  
2     radio.begin();  
3     ...  
4 }
```

Nous voilà fin prêt pour piloter notre émetteur récepteur radio ! Mais avant d'envoyer ou recevoir la moindre donnée, nous devons préalablement paramétrer certaines choses. Et c'est ce que nous allons voir, dès à présent !

4.3 - Sélection du canal de communication (setChannel)

Comme vu dans les caractéristiques du NRF24L01 plus haut, vous pouvez **choisir une fréquence de communication allant de 2400 MHz à 2525 MHz, par pas de 1 MHz**. Ceci, afin de prendre une fréquence libre, avec le moins de perturbations possibles.

Pour ce faire, il suffit de faire appel à la fonction « setChannel » dans votre programme arduino, pour définir cette fréquence. À noter que cette dernière sera basée sur 2400 MHz, à laquelle viendra s'ajouter un nombre situé entre 0 et 125. Ainsi :

- le channel 0 correspond à la fréquence de 2400 + 0, soit 2400 MHz
- le channel 1 correspond à la fréquence de 2400 + 1, soit 2401 MHz
- ...
- le channel 124 correspond à la fréquence de 2400 + 124, soit 2524 MHz
- le channel 125 correspond à la fréquence de 2400 + 125, soit 2525 MHz

Côté code arduino, ceci tient en une seule ligne (que vous pourrez mettre dans la partie « setup » de votre programme arduino) :

```
1 | radio.setChannel(x);           // en remplaçant « x » par une valeur comprise entre 0
```

4.4 - Choix du niveau de transmission du signal radio (setPALevel)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

- RF24_PA_MIN : pour un niveau d'émission minimal (idéal si vous n'avez pas besoin de communiquer à longue distance, ou si vous alimentez votre module nRF24L01 depuis l'alimentation 3,3 V de votre arduino)
- RF24_PA_LOW : niveau d'émission bas
- RF24_PA_HIGH : niveau d'émission moyen/haut
- RF24_PA_MAX : pour émettre à grande distance (jusqu'à 1,1 km de distance, dans des conditions optimales). C'est **le mode idéal si vous avez besoin de communiquer avec votre drone, avion, fusée, voiture, ou autre engin radiocommandé, avec le maximum de portée radio**. Mais attention, car dans ce cas, il faudra ajouter le fameux module d'alimentation intermédiaire, entre votre arduino, et votre module NRF24L01+.

Au niveau du code, il suffit d'utiliser la fonction « setPAlevel » pour définir la puissance d'émission, en spécifiant le niveau souhaité :

```
1 | radio.setPAlevel(xxx); // en remplaçant « xxx » par RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH
```

4.5 - Indication du débit d'émission réception (setDataRate)

Au niveau des vitesses de transmission de données, vous aurez le choix entre 3 cadences d'émission/réception:

- 250 kbps (kilo bits par seconde) : **c'est le débit de transmission le plus faible, mais qui permet d'aller le plus loin possible, en terme de portée de signal radio**
- 1 Mbps (méga bits par seconde)
- 2 Mbps (méga bits par seconde) : **c'est le débit le plus rapide, mais qui limitera la portée du signal radio** (à cause des interférences, notamment)

En résumé : plus vous essayerez « d'aller vite » et moins vous irez loin, en termes de portée du signal. Et inversement : plus vous transmettrez lentement, et plus vous pourrez émettre à « longue » distance.

Dans le code arduino, ce choix de vitesse s'écrit de la manière suivante :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

4.6 - Ouverture d'un tunnel pour de l'émission radio (setWritingPipe)

Lorsqu'on veut émettre d'un NRF24L01 à l'autre, il faut établir un canal de communication. Ce canal s'appelle un « pipe », en anglais (perso, je l'appelle aussi « tunnel ») de communication).

Pour envoyer des données d'un nRF24 à un autre, il faut donc tout d'abord donner un nom à ce tunnel (sur 5 caractères), et définir son sens. À noter, au passage, qu'un NRF24L01+ ne peut avoir qu'un seul canal d'émission au maximum (contrairement à la réception, qui elle, comme nous allons le voir après, peut se faire au travers de 5 voire 6 tunnels).

Pour ma part, voici comment je retranscris tout ça, dans l'entête du code de programmation arduino :

```
1 #define tunnel1 "PIPE1"          // On définit un "nom de tunnel" (5 caractères), poi
2 #define tunnel2 "PIPE2"
3 #define tunnel3 "PIPE3"
4 #define tunnel4 "PIPE4"
5 #define tunnel5 "PIPE5"
6 #define tunnel6 "PIPE6"
7
8 const byte adresses[][6] = {tunnel1, tunnel2, tunnel3, tunnel4, tunnel5, tunnel6};
```

Et un peu plus loin, dans la fonction setup ou loop, on ouvre le ou les canaux qui nous intéressent :

```
1 radio.openWritingPipe(adresses[0]);      // Ouverture du "tunnel1" en ÉCRITURE (émission)
```

Ainsi, dans cet exemple, le tunnel n°1 sera défini comme un canal permettant au nRF24 d'émettre des données, par ondes radios.

Nota : chaque nRF24 possède 6 pipes (6 tunnels de communication). Le 1^{er} est le

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

l'exception du PIPE0, qui, encore une fois, est le seul pouvant permettre d'émettre des données).

4.7 - Ouverture d'un tunnel pour la réception de données (setReadingPipe)

À l'image de la fonction précédente, la fonction **setReadingPipe** permet de spécifier un tunnel de communication, mais dans le sens « réception », cette fois-ci. Étant donné que le module NRF24L01 dispose de 6 pipes (ou tunnels de communication), vous pourrez recevoir des infos provenant de 6 autres nRF24, si le cœur vous en dit ! Ou plus exactement 5, si jamais vous souhaitez garder un canal pour émettre.

Au niveau du code, voici comment on pourrait initialiser tout cela, si on souhaitait recevoir des informations de 5 autres nRF24 :

```
1 #define tunnel1 "PIPE1"      // Pour l'envoi de données
2 #define tunnel2 "PIPE2"      // Pour la réception de données provenant d'un autre
3 #define tunnel3 "PIPE3"      // Pour la réception de données provenant d'un autre
4 #define tunnel4 "PIPE4"      // Pour la réception de données provenant d'un autre
5 #define tunnel5 "PIPE5"      // Pour la réception de données provenant d'un autre
6 #define tunnel6 "PIPE6"      // Pour la réception de données provenant d'un autre
```

Et plus loin, dans votre programme, l'appel de cette fonction pourra se faire comme ceci (pour la partie « écoute », donc) :

```
1 radio.openReadingPipe(1, adresses[1]); // Ouverture du "tunnel2" en LECTURE (réco
2 radio.openReadingPipe(2, adresses[2]); // Ouverture du "tunnel3" en LECTURE (réco
3 radio.openReadingPipe(3, adresses[3]); // Ouverture du "tunnel4" en LECTURE (réco
4 radio.openReadingPipe(4, adresses[4]); // Ouverture du "tunnel5" en LECTURE (réco
5 radio.openReadingPipe(5, adresses[5]); // Ouverture du "tunnel6" en LECTURE (réco
```

Bien sûr, vous pouvez vous servir des pipes comme vous le souhaitez, dans la mesure où vous respectez notamment ces deux choses là :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

- premier pipe en réception (en utilisant le code suivant : `radio.openReadingPipe(0, adresses[0]);`)
- Les 3ème à 6ème pipes ont une « adresse partagée » avec le 2ème pipe. C'est pourquoi ceux-ci devront avoir la même « racine » dans leur « nom » que le 2ème pipe (ici, la racine est correspond aux 4 premières lettres du 2ème pipe, à savoir les lettres « PIPE » ; en bref, seul la 5^{ème} lettre change, pour ces derniers pipes)

Si vous avez besoin de plus d'explications sur les « pipes », n'hésitez pas à revenir sur la partie théorique, présentée un peu plus haut, dans cet article.

4.8 - Définition du sens de communication, entre l'arduino et le NRF24L01+ (startListening et stopListening)

Avant d'effectuer la moindre lecture (read) ou écriture (write), il faudra toujours vous assurer d'être dans le bon « mode de fonctionnement ». Pour ce faire, il faudra tout simplement :

- Utiliser la fonction « startListening » si vous souhaitez pouvoir ensuite écouter via un pipe (avec la fonction « read »)
- Utiliser la fonction « stopListening » si vous souhaitez pouvoir plutôt émettre via un pipe (avec la fonction « write »)

Dans votre programme arduino, vous pourrez donc utiliser les fonctions suivantes, au besoin :

```
1 | radio.startListening();      // permet de pouvoir utiliser la fonction « read » par
2 | // ou
3 | radio.stopListening();      // permet de pouvoir utiliser la fonction « write » ens
```

4.9 - Émission et réception de données radio (read / write)

Dernières fonctions, et non des moindres : read et write. Celles-ci **permettent respectivement de pouvoir lire ce qu'on a reçu, ou d'envoyer un message, par onde radio.**

Mais avant tout, souvenez-vous d'une chose, abordée plus haut : le buffer (tampon de mémoire)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Exemple de code pour lire un message reçu (variable simple) :

```
1 | char message[32];
2 | radio.startListening();
3 |
4 | if (radio.available()) {           // On vérifie si un message est en attente
5 |     radio.read(&message, sizeof(message));    // Si oui, on le charge dans :
6 | }
```

Exemple de code pour envoyer un message (variable simple) :

```
1 | char message[32];
2 | message = "Mon message à envoyer !";      // Dans la limite de 32 octets (32 cara
3 |
4 | radio.stopListening();
5 | radio.write(&message, sizeof(message));      // Envoi du contenu stocké dan
```

Bien sûr, comme je vous disais, on n'est pas limité à une variable ou une chaîne de caractère. Car on peut utiliser « n'importe quoi », tant que ça fait moins de 32 octets en mémoire. On peut donc transmettre des ensembles de valeurs (struct), contenant des byte, boolean, int, char, ... où je ne sais quoi d'autre, tant que ça ne dépasse pas 32 octets !

Besoin d'un exemple pour illustrer tout ça ? Alors voici un bout de programme, permettant de définir un groupe de données, de type « struct » :

```
1 | struct DonneesAenvoyer {
2 |     int joystickGauchePotX; // int = 2 octets (0..65535)
3 |     int joystickGauchePotY;
4 |     int joystickDroitePotX;
5 |     int joystickDroitePotY;
6 |     boolean boostBtn;      // boolean = 1 octet (0..255)
7 |     byte offsetGauche;    // byte = 1 octet (0..255)
8 |     byte offsetDroite;
9 | };
10| DonneesAenvoyer donnees;
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

données sans soucis, car nous sommes en dessous de la taille limite du tampon de notre NRF24L01+.

Et pour envoyer ou recevoir ces données, avec ce type de structure, le codage est tout aussi simple. Car il suffit d'écrire :

- `radio.read(&donnees, sizeof(donnees));` **pour charger la totalité des données reçues en une seule fois**
- `radio.write(&donnees, sizeof(donnees));` **pour envoyer toutes les données de la « struct », à la fois**

À noter que chacune de ces données peut être lue ou chargée individuellement, en codant comme ceci :

- exemple d'enregistrement d'une nouvelle valeur dans la structure =>
`donnees.joystickGauchePotX = analogRead (A0);`
- exemple de lecture d'une donnée particulière, contenue dans la structure => byte
`leftOffset = donnees.offsetGauche;`

Nota : si jamais tout cela vous semble quelque peu compliqué, n'hésitez pas à y revenir plus tard, après avoir vu des exemples concrets. Ainsi, toute cette partie théorique vous semblera plus facile à assimiler. Dans tous les cas, ne vous faites pas des noeuds dans le cerveau, ça ne sert à rien de vouloir faire entrer les choses en force 😊

APERÇU	DESCRIPTION	LIEN ACHAT
	Module NRF24L01 (avec antenne intégrée)	
	Adaptateur d'alim (pour NRF24L01+ et version +PA+LNA)	

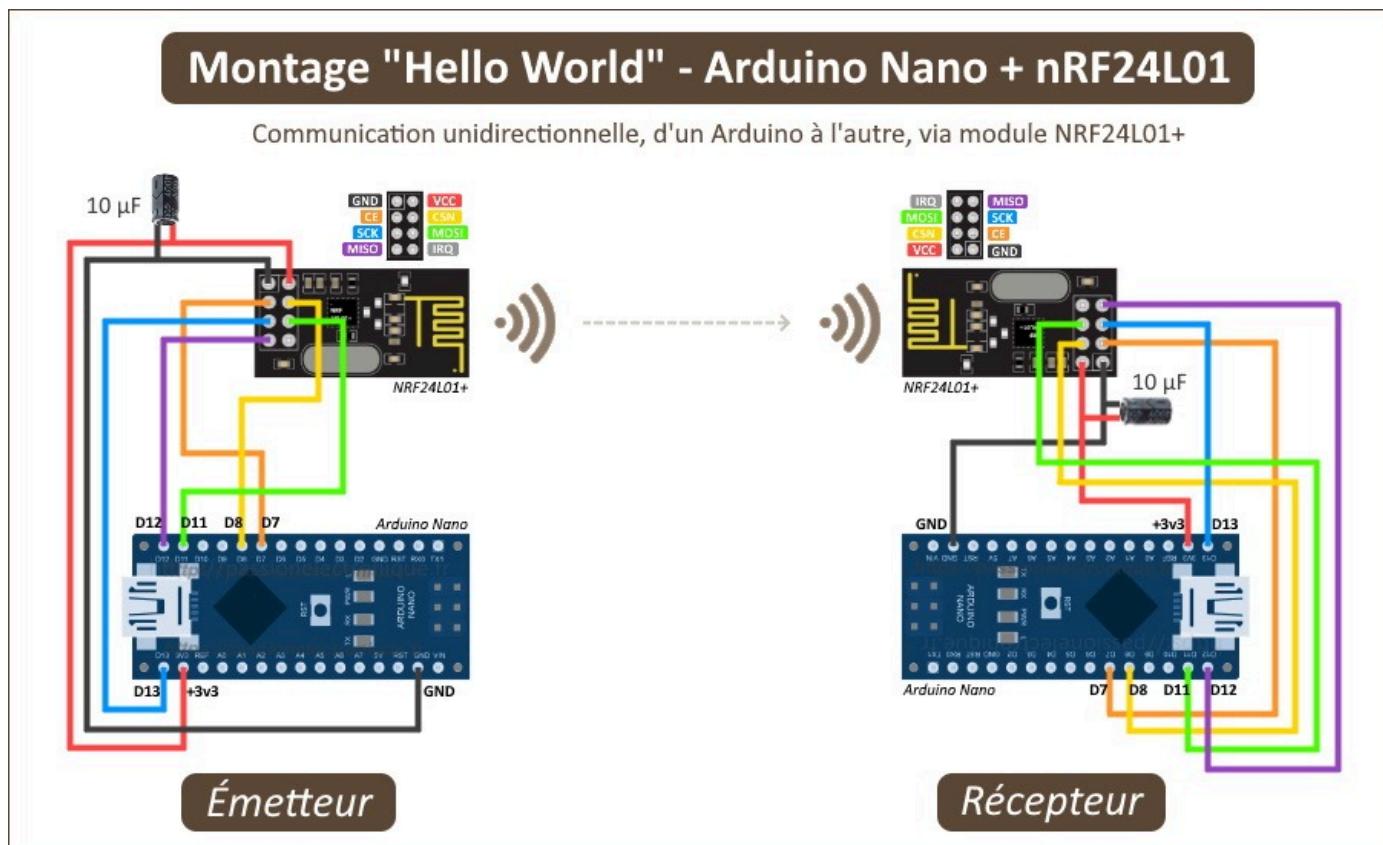
Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

5. Tuto NRF24L01 : unidirectionnel (exemple « Hello World »)

Premier exemple : réaliser l'envoi d'un message « Hello World » d'un arduino à l'autre. En résumé :

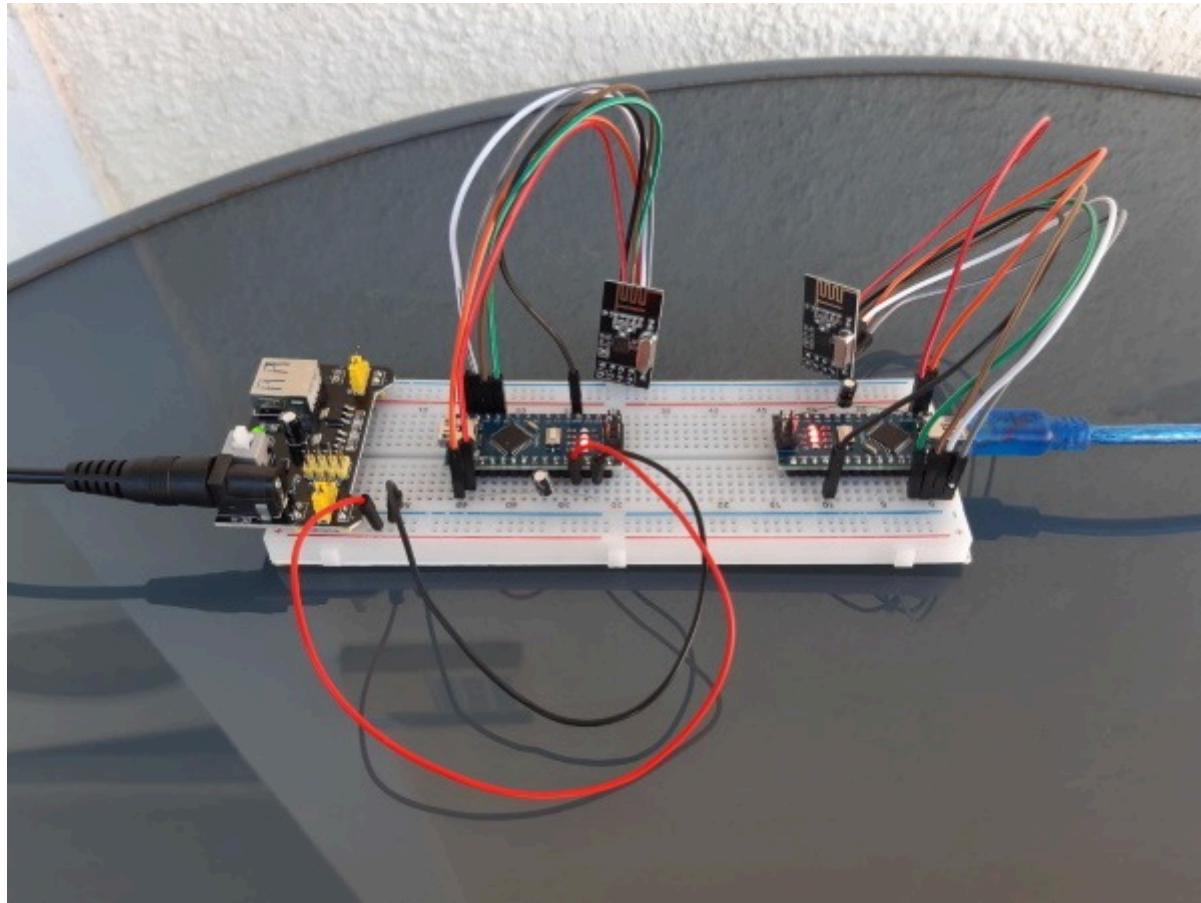
- le 1^{er} arduino enverra un message toutes les secondes, contenant la chaîne de caractères « Hello World !! »
- le 2^{ème} arduino sera à l'écoute de données radio, et si il en reçoit, il les affichera sur le moniteur série de l'IDE Arduino

Pour ce faire, voici le montage à réaliser :



Et ce que ça donne en photo, une fois câblé sur une breadboard (notez bien que le 2^{ème} arduino, faisant office de récepteur, devra être relié au PC, pour voir les données reçues, via le moniteur série) :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)



Côté programme Arduino, vous trouverez ci-après les 2 programmes animant cet exemple (un pour l'émetteur, et un autre pour le récepteur). Il faudra garder le câble de programmation branché sur « l'arduino récepteur », pour les infos reçues sur l'interface série (via le moniteur série de votre IDE Arduino).

Code de programmation émetteur Hello World :

Fichier: HelloWorldNRF24L01-Emetteur

Description: Emission d'un "Hello World" via un NRF24L01

Auteur: Passion-Électronique

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Acceptor Refuser

Code de programmation récepteur Hello World :

Fichier: HelloWorldNRF24L01-Recepteur

Description: Réception d'un message "Hello World" depuis un autre arduino nano,
Auteur: Passion-Électronique

Librairie utilisée : <https://github.com/nRF24/RF24>

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Acceptor Refuser

```

20
21 #define pinCE 7           // On associe la broche "CE" du NRF24L01 à la sortie
22 #define pinCSN 8          // On associe la broche "CSN" du NRF24L01 à la sortie
23 #define tunnel "PIPE1"    // On définit le "nom de tunnel" (5 caractères) à tra-
24
25 RF24 radio(pinCE, pinCSN); // Instanciation du NRF24L01
26
27 const byte adresse[6] = tunnel; // Mise au format "byte array" du nom du tunne-
28 char message[32];             // Avec cette librairie, on est "limité" à 32 caractères
29
30 void setup() {
31     // Initialisation du port série (pour afficher les infos reçues, sur le "Moniteur de sé-
32     Serial.begin(9600);
33     Serial.println("Récepteur NRF24L01");
34     Serial.println("");
35
36     // Partie NRF24
37     radio.begin();           // Initialisation du module NRF24
38     radio.openReadingPipe(0, adresse); // Ouverture du tunnel en LECTURE, avec le "0" comme index
39     radio.setPALevel(RF24_PA_MIN); // Sélection d'un niveau "MINIMAL" pour communiquer
40     radio.startListening();   // Démarrage de l'écoute du NRF24 (signifiant qu'il peut recevoir)
41 }
42
43 void loop() {
44     // On vérifie à chaque boucle si un message est arrivé
45     if (radio.available()) {
46         radio.read(&message, sizeof(message));           // Si un message est arrivé
47         Serial.print("Message reçu : "); Serial.println(message); // ... et on l'affiche
48     }
49 }
```

Une fois lancé, vous devriez obtenir quelque chose comme ça :

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** HelloWorldNRF24L01-Recepteur | Arduino 1.8.13 (Windows Store 1.842.0)
- File Menu:** Fichier, Edition, Croquis, Outils, Aide
- Sketch Area:** The code for the receiver is displayed:

```
1 /*  
2  *  
3  *  
4  *  
5  *  
6  *  
7  *  
8  *  
9  *  
10 Fichier:      HelloWorldNRF24L01-Recepteur  
11 Description:  Réception d'un message "Hello  
12 Auteur:       Passion-Electronique  
13  
14 Librairie utilisée : https://github.com/nan0825  
15  
16 Cr  e le 19.03.2021  
17 */  
18 #include <SPI.h>  
19 #include <RF24.h>  
20  
21 #define pinCE  9           // On associe la  
22 #define pinCSN 10          // On associe la  
23 #define tunnel  "PIPE1"     // On d  finit le  
24  
25 RF24 radio(pinCE, pinCSN); // Instanciation  
26  
27 const byte adresse[6] = tunnel; // Mise a  
28 char message[32];             // Avec c  
**/
```

Notes at the bottom:
T  l  viseusement
Le croquis utilise 3172 octets (10%) de l'espace de
Les variables globales utilisent 292 octets (14%)
- Serial Monitor:** A window titled "Récepteur NRF24L01" is open, connected to "COM3". It displays the received messages:

```
R  cepteur NRF24L01  
Message re  u : Hello World !!!  
Message re  u : Hello World !!!
```
- Bottom Bar:** Contains checkboxes for "D閏lacement automatique" and "Afficher l'historique", and buttons for "Nouvelle ligne", "9600 baud", and "Effacer la sortie".

Si jamais rien ne s'affiche, vérifiez bien tout votre câblage, et vérifiez bien que vous n'ayez pas oublié de mettre le condensateur de « soutien d'alimentation », aux NRF24L01.

Du reste, comme vous avez pu le constater, il n'y a rien de plus simple ici. Car **en synthèse**, on **ne fait qu'ouvrir une voie de communication radio (appelée pipe ou tunnel)**, et on y fait **transiter des infos, d'un arduino vers l'autre, via les NRF24L01+**. À présent, nous allons corser un peu les choses, en voyant comment échanger des informations radio de manière bidirectionnelle, afin que chaque Arduino puisse envoyer des données à l'autre. C'est parti !

APERÇU	DESCRIPTION	LIEN ACHAT
	Module NRF24L01 (avec antenne intégrée)	
	Adaptateur d'alim (pour NRF24L01+ et version +PA+LNA)	
	Module NRF24L01 + PA + LNA (avec antenne externe)	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#).

Acceptor **Refuser**

6. Tuto NRF24L01 : bidirectionnel (notion de « pipe »)

Maintenant que nous avons vu comment envoyer des messages dans un sens, on va le faire... dans les 2 sens !

Ici, je vous propose de réaliser un montage totalement symétrique. En fait, il s'agira simplement de **faire varier l'angle d'un servomoteur branché sur un arduino, en fonction de la valeur du potentiomètre branché sur l'autre arduino**. Et vice-versa, pour obtenir un fonctionnement croisé.

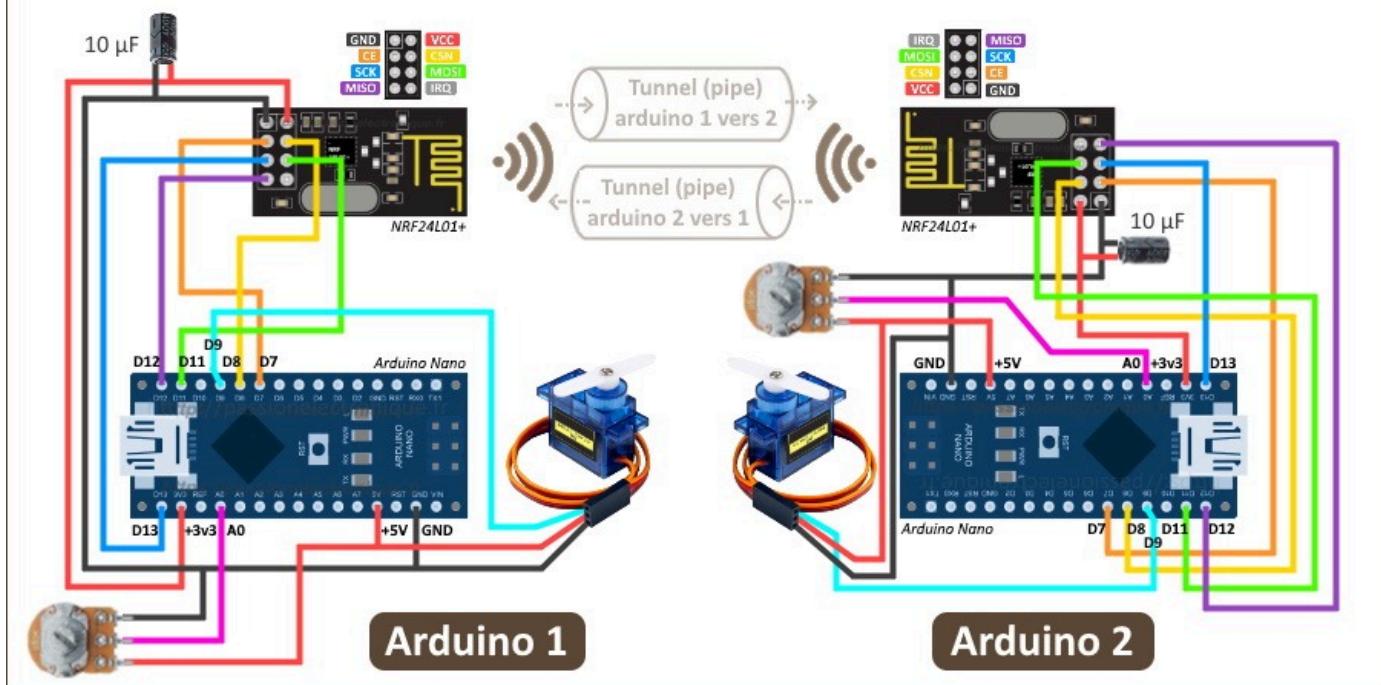
C'est pas clair ? Voici une petite vidéo du montage « en action », illustrant tout ceci :



Au niveau du montage électronique en lui-même, voici ce que vous aurez à réaliser, si vous souhaitez reproduire ce que j'ai fait :

Communication entre deux Arduino (bidirectionnel)

Communication bidirectionnelle, entre 2 Arduino Nano, via modules NRF24L01+



Au niveau du code arduino, voici le programme de l'émetteur :

Fichier: CommunicationBidirectionnelleNRF24L01-Arduino1

Description: Pilotage d'un servomoteur à distance, depuis un autre Arduino, via
Auteur: Passion-Électronique

Librairie utilisée : <https://github.com/nRF24/RF24>

Créé le 20.03.2021

* /

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

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

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Acceptor Refuser

```

25 #define pinPOT A0           // On associe le point milieu du potentiomètre à l'entrée
26
27 #define tunnel1 "PIPE1"     // On définit un premier "nom de tunnel" (5 caractères)
28 #define tunnel2 "PIPE2"     // On définit un second "nom de tunnel" (5 caractères)
29
30 RF24 radio(pinCE, pinCSN); // Instanciation du NRF24L01
31 Servo servomoteur;        // Instanciation d'un objet pour contrôler le servomoteur
32
33 const byte adresses[][6] = {tunnel1, tunnel2}; // Tableau des adresses de tunnel
34
35 int valeurPotLocal;       // Variable contenant la valeur du potentiomètre
36 int valeurAngleServoLocal; // Variable contenant la valeur de l'angle du servomoteur
37 int valeurAngleServoDistant; // Variable contenant la valeur de l'angle du servomoteur
38
39 void setup() {
40   pinMode(pinPOT, INPUT);      // Déclaration de la pin "pinPOT" en entrée
41   servomoteur.attach(pinSERVO); // Liaison de la pin "pinSERVO" au servomoteur
42
43   radio.begin();              // Initialisation du module NRF24
44   radio.openWritingPipe(adresses[0]); // Ouverture du "tunnel1" en ÉCRITURE
45   radio.openReadingPipe(1, adresses[1]); // Ouverture du "tunnel2" en LECTURE
46   radio.setPALevel(RF24_PA_MIN); // Sélection d'un niveau "MINIMAL" pour la puissance
47
48   servomoteur.write(90);      // Rotation du servomoteur pour le mettre en position initiale
49   delay(2000);               // puis démarrage du programme
50 }
51
52 void loop() {
53   // ***** ENVOI *****
54   radio.stopListening();      // On coupe la réception
55   valeurPotLocal = analogRead(pinPOT); // On lit la valeur du potentiomètre
56   valeurAngleServoDistant = map(valeurPotLocal, 0, 1023, 0, 180); // On convertit la valeur en angle
57   valeurAngleServoDistant = 2*(int)(valeurAngleServoDistant/2); // Léger arrondissement
58   radio.write(&valeurAngleServoDistant, sizeof(valeurAngleServoDistant)); // ... et on l'envoie
59   delay(5);                  // avec une courte pause
60
61   // ***** RÉCEPTION *****
62   radio.startListening();    // On coupe la transmission
63   if(radio.available()) {    // On recherche si il y a quelque chose à lire
64     while (radio.available()) { // Si une transmission est en cours
65       radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal)); // Lecture de l'angle
66       servomoteur.write(valeurAngleServoLocal); // ... et on l'affiche
67     }
68     delay(20);                // avec une courte pause
69   }

```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)

Et le code de programmation du récepteur :

Fichier: CommunicationBidirectionnelleNRF24L01-Arduino2
Description: Pilotage d'un servomoteur à distance, depuis un autre Arduino, via
Auteur: Passion-Électronique

Librairie utilisée : <https://github.com/nRF24/RF24>

Créé le 20.03.2021

```
/*
#include <SPI.h>
#include <RF24.h>
#include <Servo.h>

#define pinCE    7      // On associe la broche "CE" du NRF24L01 à la sortie d'adresse
#define pinCSN   8      // On associe la broche "CSN" du NRF24L01 à la sortie de sélection
#define pinSERVO 9      // On associe la broche "SIGNAL" du SERVO à la sortie de commande
#define pinPOT    A0     // On associe le point milieu du potentiomètre à l'entrée analogique

#define tunnel1 "PIPE2" // On définit un premier "nom de tunnel" (5 caractères)
#define tunnel2 "PIPE1" // On définit un second "nom de tunnel" (5 caractères)
/* ----- NOTEZ L'INVERSION de PIPE1 et PIPE2, entre celui-ci et l'autre montage ! */

RF24 radio(pinCE, pinCSN); // Instanciation du NRF24L01
Servo servomoteur;         // Instanciation d'un objet pour contrôler le servomoteur

const byte adresses[][][6] = {tunnel1, tunnel2}; // Tableau des adresses de tunnel

int valeurPotLocal;          // Variable contenant la valeur du potentiomètre
int valeurAngleServoLocal;   // Variable contenant la valeur de l'angle du servo local
int valeurAngleServoDistant; // Variable contenant la valeur de l'angle du servo distant

void setup() {
    pinMode(pinPOT, INPUT); // Déclaration de la pin "pinPOT" en entrée
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

```

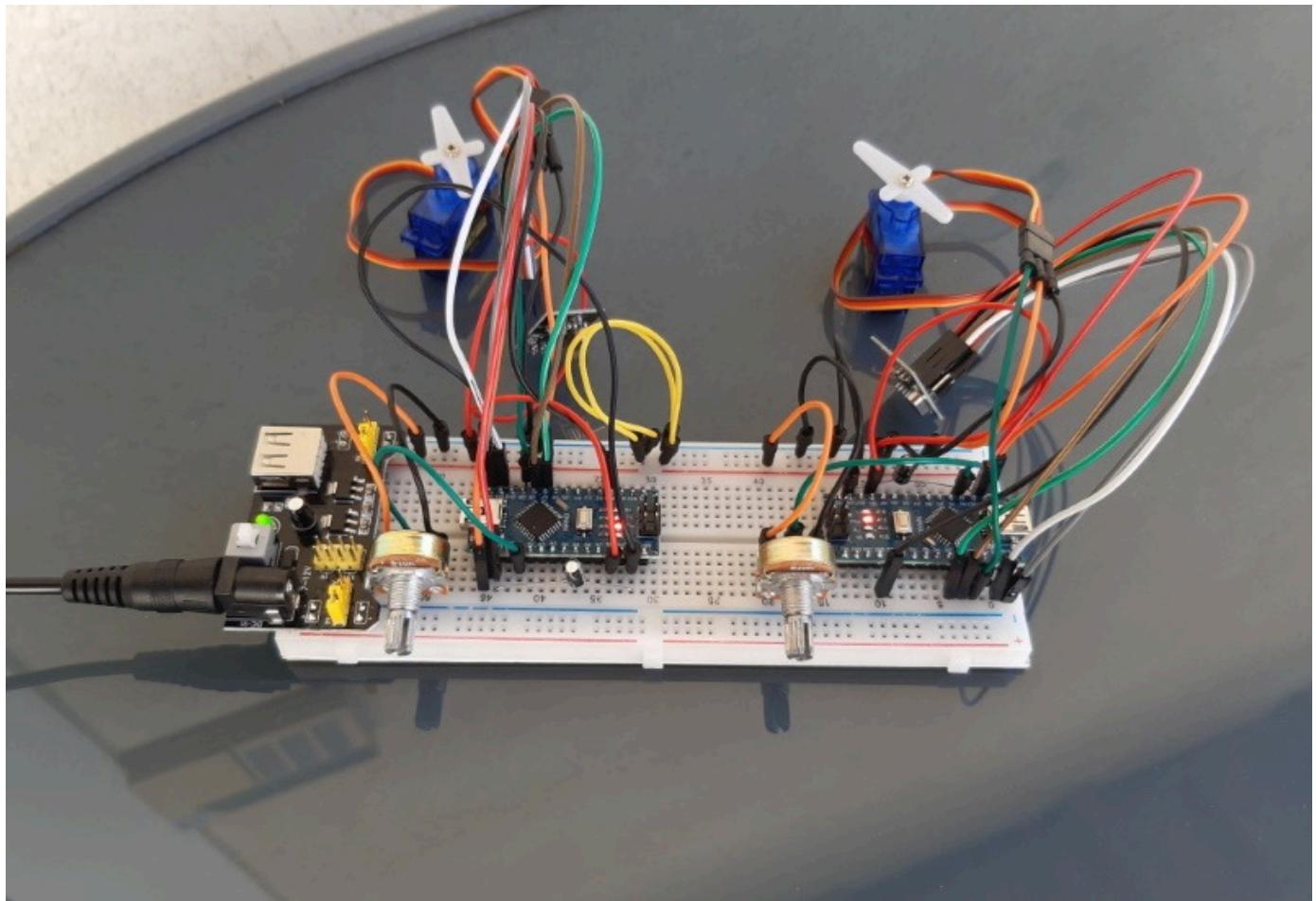
47 radio.setPALevel(RF24_PA_MIN);           // Sélection d'un niveau "MINIMAL" pour
48
49 servomoteur.write(90);      // Rotation du servomoteur pour le mettre en position i
50 delay(2000);                // puis démarrage du programme
51 }
52
53 void loop() {
54 // ***** ENVOI *****
55 radio.stopListening();          // On coi
56 valeurPotLocal = analogRead(pinPOT);    // On li·
57 valeurAngleServoDistant = map(valeurPotLocal, 0, 1023, 0, 180); // On coi
58 valeurAngleServoDistant = 2*(int)(valeurAngleServoDistant/2); // Léger
59 radio.write(&valeurAngleServoDistant, sizeof(valeurAngleServoDistant)); // ... et ...
60 delay(5);                      // avec ...
61
62 // ***** RÉCEPTION *****
63 radio.startListening();          // On coi
64 if(radio.available()) {          // On re...
65   while (radio.available()) {     // Si un ...
66     radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal)); // Lecture
67     servomoteur.write(valeurAngleServoLocal);                         // ... et ...
68   }
69   delay(20);                   // avec ...
70 }
71 delay(5);
72 }

```

Comme vous avez dû le constater, les deux programmes sont quasiment des copies conformes. À ceci près, toutefois, que les adresses de tunnel de communication sont inversés, d'un programme à l'autre (ce qui est normal, car, à travers un tunnel, les informations ne circulent que dans un sens).

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)



Et maintenant que vous savez faire échanger des données de manière bidirectionnelle, d'un NRF24L01 à l'autre, il nous reste plus qu'une dernière étape à voir : la communication d'une multitude de nRF24 entre eux !

IMAGE	DESCRIPTION	LIEN
	<p>Caméra WiFi extérieur, pour vidéosurveillance (modèle double, avec partie fixe + mobile 360°)</p>	Voir le prix

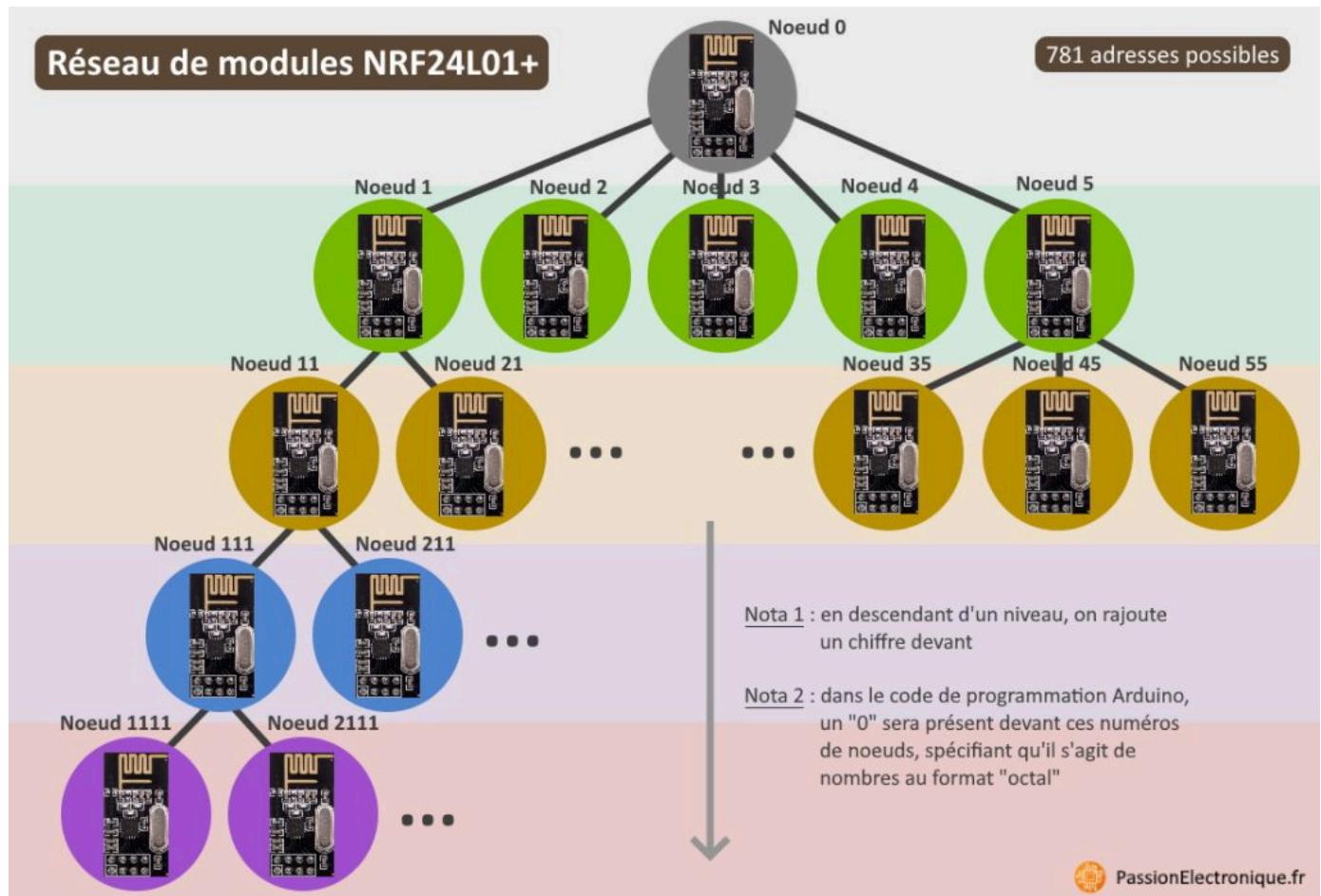
7. Tutorial NRF24L01 : montage en réseau / arborescence, avec nœuds (tree mode et nodes mesh networking)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)

très nombreuses perspectives, non ? Mais avant de vous montrer comment coder tout ça sous Arduino, il faut que je vous explique 2 ou 3 choses.

Pour commencer, il faut savoir que nous allons ici créer un réseau (comme le réseau WiFi, si vous voulez, mais en plus simple). Chaque module nRF24 aura sa propre adresse, mais attention, pas n'importe laquelle (il y aura moins de liberté que dans les exemples précédents). En fait, chaque module radio sera virtuellement raccordé à l'autre en arborescence (« tree », en anglais), comme visible ci-dessous :



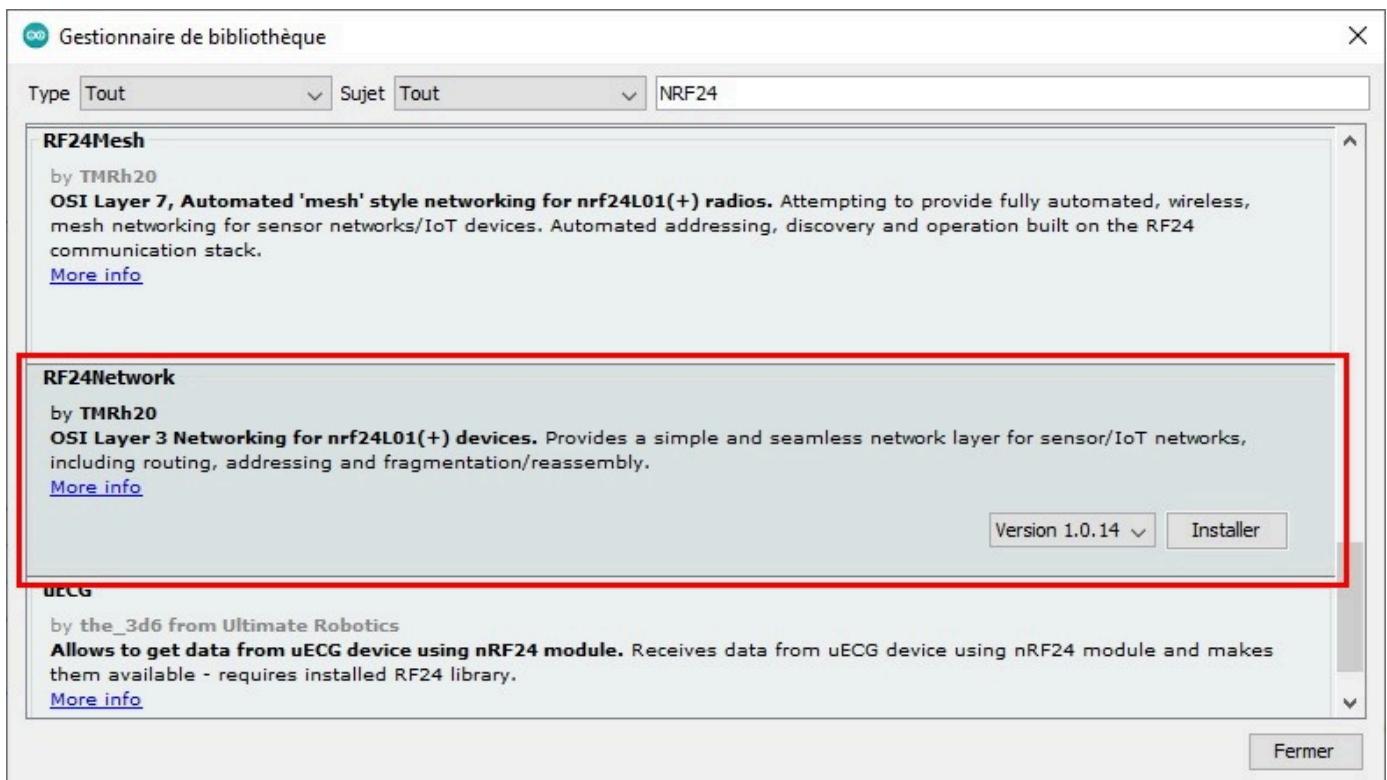
La règle à savoir, concernant l'adressage de chaque module, est que :

- il n'y a qu'un module nRF24 « maître » (l'adresse 0)
- chaque nRF24 ne peut avoir que 5 modules nRF24 « enfant », sous lui
- on est limité à 5 niveaux d'arborescence (en sachant que le 1^{er} niveau ne contient que le module maître, avec l'adresse zéro)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Remarque importante : chaque module a une adresse chiffrée, écrite au format octal (donc ni en binaire, ni en décimal, ni en hexadécimal, ... mais seulement des chiffres allant de 0 à 7). **Au niveau du programme arduino, un nombre octal s'écrit avec un « 0 » devant.** Du coup, lorsque vous voyez « 00 », cela veut dire le chiffre 0 en format octal. De même, si vous voyez par exemple « 0130 », cela veut dire qu'il s'agit du nombre 130 écrit au format octal. Ne croyez donc pas que j'ai rajouté des « 0 » pour m'amuser dans les programmes qui suivent ! Et de votre côté, ne les oubliez pas !

À noter que nous allons utiliser ici une librairie supplémentaire, à celle de base (la nRF24). Il s'agit en fait de **la librairie nRF24 Network**, permettant de créer des réseaux de modules NRF24L01 ! Elle est donc à utiliser en plus de la librairie nRF24 que nous avons utilisé jusqu'à présent (voir plus haut). Pour l'installer, il suffit également d'aller dans le « Gestionnaire de bibliothèques » de votre Arduino IDE, mais au lieu de taper « NRF24 », il va falloir taper « NRF24Network », cette fois-ci. Et comme la dernière fois, il faudra choisir celle qui est sous-titrée « by TMRh20 ».



Maintenant que nous avons vu toute la partie théorique, passons à la pratique ! Pour ce faire, **je vais reprendre exactement le même montage, réalisé juste avant (communication**

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

communiquant entre eux, libre à vous d'en mettre autant que vous voulez (mais dans la limite des 781 modules, bien évidemment !).

À présent, voici le code de l'arduino ayant l'adresse « 00 » (donc « 0 », écrit au format octal) :

Fichier: NetworkNRF24L01-Arduino1

Description: Construction d'un réseau communicant, via des NRF24L01 connectés à
Auteur: Passion-Électronique

Librairie utilisée : <https://github.com/nRF24/RF24>

Créé le 21.03.2021

```
/*
#include <SPI.h>
#include <RF24.h>
#include <RF24Network.h>
#include <Servo.h>

#define pinCE      7      // On associe la broche "CE" du NRF24L01 à la sortie d'entraînement du servomoteur
#define pinCSN     8      // On associe la broche "CSN" du NRF24L01 à la sortie de sélection du réseau
#define pinSERVO   9      // On associe la broche "SIGNAL" du SERVO à la sortie d'entraînement du servomoteur
#define pinPOT     A0     // On associe le point milieu du potentiomètre à l'entrée de sélection du réseau

RF24 radio(pinCE, pinCSN);      // Instanciation du NRF24L01
RF24Network network(radio);    // Nota : "Network" utilise la librairie "radio"
Servo servomoteur;             // Instanciation d'un objet pour contrôler le servomoteur

// Réseau
const uint16_t noeud0 = 00;      // Valeur "0" écrite au format "octal" (d'où l'autorisation de deux zéros)
const uint16_t noeud1 = 01;      // Valeur "1" écrite au format "octal" (d'où l'autorisation de deux zéros)
const uint16_t noeud11 = 011;    // Valeur "11" écrite au format "octal" (d'où l'autorisation de deux zéros)
const uint16_t noeud12 = 012;    // Valeur "12" écrite au format "octal" (d'où l'autorisation de deux zéros)
uint16_t numeroDeCeNoeud = noeud0;

uint16_t noeudCible = noeud1;    // Ici, le noeud sera donc le "noeud0", et le noeud cible sera le noeud1
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

```

44 unsigned long donnees; // Variable contenant les données
45
46 void setup() {
47   SPI.begin();
48
49   pinMode(pinPOT, INPUT); // Déclaration de la pin "pinPOT" en entrée
50   servomoteur.attach(pinSERVO); // Liaison de la pin "pinSERVO" au servomoteur
51
52   radio.begin(); // Initialisation du module NRF24
53   radio.setPALevel(RF24_PA_MIN); // Sélection d'un niveau "MINIMAL" pour
54   radio.setDataRate(RF24_250KBPS); // Vitesse : RF24_250KBPS, RF24_1MBPS, ...
55   network.begin(64, numeroDeCeNoeud); // Canal 64 choisi arbitrairement (valeu
56
57   servomoteur.write(90); // Rotation du servomoteur pour le mettre en position !
58   delay(2000); // puis démarrage du programme
59 }
60
61 void loop() {
62   network.update();
63
64   // ***** RÉCEPTION *****
65   while (network.available()) { // Si un paquet est arrivé
66     RF24NetworkHeader nHeader(numeroDeCeNoeud);
67     network.read(nHeader, &valeurAngleServoLocal, sizeof(valeurAngleServoLocal));
68     servomoteur.write(valeurAngleServoLocal); // ... et l'envoie
69   }
70   delay(5);
71
72   // ***** ENVOI *****
73   valeurPotLocal = analogRead(pinPOT); // On lit la position
74   valeurAngleServoDistant = map(valeurPotLocal, 0, 1023, 0, 180); // On convertit
75   valeurAngleServoDistant = 2*(int)(valeurAngleServoDistant/2); // Léger arrondi
76   RF24NetworkHeader nHeader(noeudCible);
77   network.write(nHeader, &valeurAngleServoDistant, sizeof(valeurAngleServoDistant))
78   delay(5); // avec une petite pause
}

```

Et le code de l'arduino ayant l'adresse « 01 » (donc « 1 », écrit au format octal) :

```
/* _____ - // / _ _ _ _ _
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#).

Accepter Refuser

Fichier: NetworkNRF24L01-Arduino2
Description: Construction d'un réseau communicant, via des NRF24L01 connectés à
Auteur: Passion-Électronique

Librairie utilisée : <https://github.com/nRF24/RF24>

Créé le 21.03.2021

```
/*
#include <SPI.h>
#include <RF24.h>
#include <RF24Network.h>
#include <Servo.h>

#define pinCE    7      // On associe la broche "CE" du NRF24L01 à la sortie de l'entité
#define pinCSN   8      // On associe la broche "CSN" du NRF24L01 à la sortie de l'entité
#define pinSERVO 9      // On associe la broche "SIGNAL" du SERVO à la sortie de l'entité
#define pinPOT   A0     // On associe le point milieu du potentiomètre à l'entrée A0

RF24 radio(pinCE, pinCSN);      // Instanciation du NRF24L01
RF24Network network(radio);    // Nota : "Network" utilise la librairie "radio"
Servo servomoteur;             // Instanciation d'un objet pour contrôler le servomoteur

// Réseau
const uint16_t noeud0 = 00;      // Valeur "0" écrit au format "octal" (d'où l'autre nom)
const uint16_t noeud1 = 01;      // Valeur "1" écrit au format "octal" (d'où l'autre nom)
const uint16_t noeud11 = 011;    // Valeur "11" écrit au format "octal" (d'où l'autre nom)
const uint16_t noeud12 = 012;    // Valeur "12" écrit au format "octal" (d'où l'autre nom)
uint16_t numeroDeCeNoeud = noeud1;
uint16_t noeudCible = noeud0;    // Ici, le noeud sera donc le "noeud1", et le noeud cible sera donc le noeud0

unsigned long valeurPotLocal;      // Variable contenant la valeur du potentiomètre
unsigned long valeurAngleServoLocal; // Variable contenant la valeur de l'angle du servomoteur local
unsigned long valeurAngleServoDistant; // Variable contenant la valeur de l'angle du servomoteur distant
unsigned long donnees;             // Variable contenant les données

void setup() {
  SPI.begin();

  pinMode(pinPOT, INPUT);          // Déclaration de la pin "pinPOT" en entrée
  servomoteur.attach(pinSERVO);    // Liaison de la pin "pinSERVO" au servomoteur bidirectionnel

  radio.begin();                  // Initialisation du module NRF24
  radio.setPALevel(RF24_PA_MIN); // Sélection d'un niveau "MINIMAL" pour la puissance de transmission
}
```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

```

58     delay(2000);                                // puis démarrage du programme
59 }
60
61 void loop() {
62     network.update();
63
64     // ***** RÉCEPTION *****
65     while (network.available()) {                // Si une
66         RF24NetworkHeader nHeader(numeroDeCeNoeud);
67         network.read(nHeader, &valeurAngleServoLocal, sizeof(valeurAngleServoLocal));
68         servomoteur.write(valeurAngleServoLocal);    // ... et ...
69     }
70     delay(5);
71
72     // ***** ENVOI *****
73     valeurPotLocal = analogRead(pinPOT);          // On lis...
74     valeurAngleServoDistant = map(valeurPotLocal, 0, 1023, 0, 180);    // On co...
75     valeurAngleServoDistant = 2*(int)(valeurAngleServoDistant/2);      // Léger
76     RF24NetworkHeader nHeader(noeudCible);
77     network.write(nHeader, &valeurAngleServoDistant, sizeof(valeurAngleServoDistant))
78     delay(5);                                    // avec ...
}

```

Nota : je vous ai rajouté des nœuds inutilisés dans le code (noeud11 et noeud12). Ils sont juste là pour vous montrer comment simplement rajouter d'autres nRF24 à l'arborescence. Bien sûr, libre à vous d'adapter tout cela, en fonction de vos besoins !

APERÇU	DESCRIPTION	LIEN ACHAT
	Module NRF24L01 (avec antenne intégrée)	
	Adaptateur d'alim (pour NRF24L01+ et version +PA+LNA)	
	Module NRF24L01 + PA + LNA (avec antenne externe)	

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Comme j'ai eu pas mal de questions au sujet du NRF24 PA LNA (celui avec antenne externe), notamment au sujet de son raccordement sur l'arduino mega, j'ai décidé de vous ajouter cette partie.

Ici, vous trouverez un exemple de câblage, vous montrant **comment raccorder un NRF24 PA LNA à votre Arduino Méga**. Vous allez voir, il n'y a rien de bien sorcier !

Remarque importante : notez bien la présence d'une **alimentation externe EN PLUS de l'éventuelle connexion USB**. Celle-ci est quasi indispensable, du fait que **le modèle nRF24 PA LNA est bien plus gourmand en énergie que son homologue sans antenne externe, non amplifié**. Du coup, n'oubliez jamais de rajouter cette source d'énergie supplémentaire, afin que l'ensemble puisse bien fonctionner. Sinon, vous risquez fort d'avoir des soucis de communication !

Pour ma part, j'ai réalisé le câblage ci-dessus en double exemplaire, pour tester le programme « Hello World » présenté un peu plus haut. En pratique, je vous confirme que sans alim

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Composants utilisés ici :

- 2 x cartes **Arduino Méga 2560**
- 2 x modules radio **NRF24 PA LNA**
- 2 x modules d'**alimentation pour NRF24**
- 2 x blocs d'**alimentation externe 7,5V / 1A** (DC 5,5/2,5)

Nota : j'insiste bien sur le fait qu'un Arduino seul, qu'il s'agisse d'un Uno, Mega, ou autre, aura beaucoup de mal à fournir toute l'énergie nécessaire, pour que le nRF24 PA LNA puisse fonctionner correctement. D'où la nécessité d'ajouter une alim externe, lorsqu'on fait ses essais.

9. [Ajout] Problèmes fréquents avec les modules

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Suite à vos nombreux retours, je vais vous recenser ici bon nombre de problèmes fréquents qu'on peut rencontrer, **en branchant un module nRF24L01 PA LNA sur un Arduino**. Cela vous permettra peut-être d'expliquer pourquoi vous n'arrivez pas forcément à faire marcher votre module radio NRF sur votre arduino ! Ainsi, vous arriverez peut-être à vous en dépatouiller, avec ce qui suit 😊

9.1 - Mauvaise soudure d'usine

Ne riez pas, c'est plus fréquent qu'on croit ! Pour preuve, ça m'est arrivé il y a deux jours seulement, **avec des modules NRF24 PA LNA, entièrement neufs**. Du coup, au moindre signe de fonctionnement anormal, il ne faut surtout pas hésiter à examiner toutes les soudures des composants « préassemblés » que vous avez acheté, avant de jeter l'éponge !

Et comme une image est souvent bien plus parlante que les mots, voici l'exemple type d'une soudure mal faite, et qui engendrait, de mon côté, des bugs intermittents de communication :

Ici, **une mauvaise soudure sur la broche GND du module d'alimentation de mon PA LNA m'a causé des fonctionnements erratiques**. Sur un autre module, j'avais dû ressouder une autre broche, car la soudure n'était pas bien faite, et m'a causé une certaine instabilité de transmission.

Mon conseil : dès que vous constatez des comportements bizarres ou anormaux, n'hésitez pas à

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

9.2 - Alimentation pas suffisamment puissante

Autre souci, et certainement celui qui vous causera le plus de soucis, si vous n'en tenez pas compte : la **nécessité de fournir suffisamment de puissance aux modules PA LNA (avec antenne externe)**, comparé aux modules « sans antenne » externe (ou plutôt intégrée au PCB, devrais-je dire). Car les modèles NRF24 PA LNA sont bien plus énergivores, et nécessitent donc une alimentation plus puissante.

Or, les régulateurs de tension sur les plaquettes Arduino ont des limites qui peuvent facilement être franchies, une fois un module nRF24 branché dessus. Du coup, une alimentation « de soutien » pour votre arduino est quasi indispensable, si vous souhaitez que tout fonctionne bien. Sinon : gare aux comportements bizarres, et fonctionnements erratiques !

Pour vous illustrer cela, voici un exemple de « ce qu'il faut faire », et « ne pas faire » (pour éviter toute surprise) :

Et croyez-moi, une alimentation « complémentaire » peut faire toute la différence. Car l'absence d'alimentation de soutien explique la plupart du temps les problèmes que vous rencontrez avec

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Rappel : comme évoqué plus haut dans ce tuto, **un condensateur électrolytique à proximité des broches VCC et GND des modules NRF24 permet de leur assurer une bien meilleure stabilité, notamment à « haut débit ».** D'ailleurs, ceci cumulé à l'ajout d'une alimentation externe de soutien, vous permettra très certainement d'atteindre les meilleures performances qui soient, en terme d'émission réception radio.

9.3 - Fils dupont mal branchés, ou qui ne serrent pas suffisamment

Plus généralement, **faites bien attention aussi aux fils dupont, qui, suivant leur qualité de fabrication ou fréquence d'utilisation, peuvent quelquefois être source de « faux-contact ».**

C'est en tout cas ce que j'ai pu remarquer de mon côté, avec le temps. Et croyez-moi là encore : il n'y a rien de plus agaçant que de chercher une panne « pendant des heures » du côté logiciel, pour s'apercevoir au final, que le problème venait d'un fil dupont à moitié déconnecté, ou pas assez « serré » !

Du coup, si vous avez un fonctionnement anormal, n'hésitez pas à vérifier vos raccordements filaires 😊

9.4 - Interférences électromagnétiques

Plus rare, mais autant vous en faire part : **les interférences électromagnétiques peuvent également perturber les émissions réceptions d'ondes radio.** C'est pourquoi, dans certains cas, il pourrait être intéressant d'effectuer un blindage tout autour du PCB du NRF24 PA LNA, à relier à la masse (GND), pour évacuer tout parasitage à ce niveau.

Bien sûr, il faudra alors penser à bien s'assurer qu'aucun composant électronique ne touche le blindage, afin d'éviter tout risque de court-circuit.

Nota : pour voir à quoi ce blindage pourrait ressembler, je vous mets en lien 2 photos ci-après, partagées par celui qui a conçu la librairie arduino utilisée dans ce tuto : [lien 1](#) et [lien 2](#). Ainsi, cela vous montre comment blinder un module NRF24 PA LNA, lorsque c'est nécessaire !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

APERÇU	DESCRIPTION	LIEN ACHAT
	Module NRF24L01 (avec antenne intégrée)	
	Adaptateur d'alim (pour NRF24L01+ et version +PA+LNA)	
	Module NRF24L01 + PA + LNA (avec antenne externe)	

10. Conclusion

Nous voici au terme de ce long article, vous donnant **les bases du NRF24L01 et plusieurs exemples d'utilisation avec code arduino**. J'espère que vous aurez pu apprendre ou découvrir plein de choses intéressantes ici, pour rendre vos futurs projets 100% communicant !

Pour ma part, je vais plancher sur la création d'une télécommande basée sur ce nRF24, pour piloter une voiture radiocommandée ! Il ne me reste plus qu'à trouver du temps pour faire tout cela, et vous le mettre en forme dans un nouvel article ! Je vous dis donc à très bientôt 😊

Jérôme.

À lire aussi : [tutoriel sur le convertisseur analogique numérique ADS1115](#) !

Ce contenu vous plaît ? Alors abonnez-vous à la Newsletter pour ne rien louper !

 [Recevoir la Newsletter !](#)

JEROME

Passionné par tout ce qui touche à l'électronique, sans toutefois en être expert, j'ai à cœur de vous partager ici, peu à peu, tout ce que j'ai appris, découvert, réalisé, et testé jusqu'à présent ! En espérant que tout cela

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)

99 commentaires sur “Module NRF24L01 Arduino : caractéristiques, librairies, et explications (tutorial avec exemples de code arduino)”

ANTOINE

31 mars 2021 à 07:25

Bonjour Jérôme,

Tous mes remerciements pour ce super tuto au sujet du NRF24L01 qui n'est pas très simple à utiliser, je l'avoue.

Pour ma part j'aimerais utiliser cette transmission de façon un peu différente.

À partir d'un émetteur (que émetteur), je voudrais envoyer les mêmes données sur 3 récepteurs distincts.

De par le mode de communication de Nordic, le « ShockBurst », que je comprend comme un accusé de réception, n'y aura-t-il pas de problème de « télescopage » si je n'utilise qu'un seul canal d'émission ?

Merci par avance pour votre réponse.

Cordialement

Antoine

Jérôme

31 mars 2021 à 08:23

Salut Antoine !

Tu as mis le doigt exactement où il faut ! Car en effet, dans ton cas, il faudra désactiver l'accusé de réception (Ack), pour pouvoir envoyer des infos à plusieurs récepteurs ayant la même adresse. Pour ce faire, avec la librairie nRF24, il te suffit de rajouter la commande « setAutoAck(false); » dans la fonction « setup » de ton code arduino, tout simplement.

Par contre, je t'avouerai que je n'ai jamais utilisé mes NRF24L01+ comme ça,
donc ce reste à vérifier !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

ANTOINE

31 mars 2021 à 16:29

Ouahou, quelle réactivité !

Un grand merci pour ta réponse aussi précise que rapide. Je vais faire des essais et ne manquerai pas de t'informer du résultat, cela pourra peut-être aider quelqu'un d'autre.

A titre d'infos, je suis entrain de réaliser avec l'aide d'un ami ferronnier, un simulateur dynamique 360° dans les 3 directions (avion, grand 8, ou même automobile), ce qui oblige la liaison par radio car les collecteurs tournants seraient trop complexes.

S'il y a des amateurs ...

En attendant je vais un peu fouiller sur ton site qui semble très intéressant.

Bien cordialement

Antoine

Jérôme

31 mars 2021 à 19:52

Ça marche, avec plaisir !

De mon côté, je vais bosser sur la réalisation de télécommandes RC, plus ou moins sophistiquées, pour le pilotage de voitures radiocommandées. Comme ça, chacun pourra trouver des exemples concrets d'application des émetteurs/récepteurs radio avec arduino, ce qui sera moins barbant que la théorie 😊

En tout cas, je te dis à bientôt !

Jérôme.

PAUL CHOPIN

22 avril 2021 à 17:40

Bonjour,

Je galère depuis quelques jours sur le NRF24L01. Je commençais à désespérer. Je vais donc essayer à nouveau grâce à votre « Tuto » qui me paraît très bien documenté et expliqué. Enfin quelque chose de clair sur un sujet qui n'est pas si simple. Je

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Paul

Jérôme

22 avril 2021 à 17:58

Bonsoir Paul !

Vous avez raison de persister, car ce module est vraiment très intéressant à utiliser, notamment dans les projets « sans fil » requérant un peu de distance (là où le bluetooth montre ses limites). Du coup, bon courage à vous !

Jérôme.

Mitch

5 mai 2021 à 18:01

Merci de cette démonstration qui a certainement dû vous demander beaucoup de temps.

Électronicien de base, j'ai quasiment tout compris, reste à mettre en pratique.

Seule petite coquille en début d'article dans le tableau des distances en fonction des débits, c'est l'inverse : quand la distance augmente le débit baisse.

Encore merci.

Jérôme

5 mai 2021 à 18:21

Bonsoir Mitch !

Tout d'abord merci pour ton retour, qui fait vraiment plaisir.

Du reste, tu as parfaitement raison ! J'ai commis une erreur d'écriture, en renseignant les valeurs « à l'envers » dans le tableau distance/débit du haut. Du coup, c'est bon, c'est corrigé 😊

Encore merci à toi pour la remontée de cette petite coquille, et bon courage à toi, pour la mise en pratique de tout ça !

Jérôme.

Ollivier

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

tout est super clair ...

... mais, j'ai une question sur la réception de la structure. Si je comprends bien, « byte leftOffset = offset.offsetGauche; » implique que l'on a reçu l'ensemble des données dans une structure que l'on a défini juste avant la lecture. Voici la question : doit-on définir la struct complètement ? (nom, type et ordre des données). Dans l'exemple ci-dessus, j'en ai déduit que l'on recevait les données dans la « struct offset » ...

Merci encore pour cette page 😊

Jérôme

24 mai 2021 à 14:02

Salut à toi !

Oui, tu as bien pigé le truc. Il faut effectivement définir la structure supportant les données avant tout, pour pouvoir ensuite envoyer ou recevoir l'intégralité des données sous cette forme. Et comme tu l'as bien compris, on peut lire ou écrire chacune des données présentes à l'intérieur individuellement, mais seulement une fois que cette structure a été déclarée.

Si tu veux un exemple d'utilisation concret de tout cela, n'hésite pas à checker les programmes arduino que j'ai écrit pour cette petite télécommande RF : <https://passionelectronique.fr/radiocommande-arduino-rc1/>. Car cela t'illustrera parfaitement comment utiliser une structure, aussi bien en émission qu'en réception, sur un projet concret.

Par contre, suite à ce que tu m'as écrit, je viens de m'apercevoir qu'il s'est glissé une petite coquille dans ce tuto sur le nRF24L01 ! En effet, je ne voulais pas écrire « offset.offsetGauche; », mais bien « donnees.offsetGauche; ». Car « donnees » était le nom que j'avais donné à l'instance de cette structure, quelques lignes plus haut, et je voulais conserver ce nom. Je corrige donc cela tout de suite, afin que ça n'embrouille personne davantage ! Désolé !

Jérôme.

Seb

15 juin 2021 à 07:28

Bonjour Jérôme, et merci pour ce Tuto qui m'a mis sur les rails.

Cela fait pas mal de temps que je cherche à relier en radio ma station météo qui ne

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

J'essaye de mettre en communication une UNO avec une MEGA (si il devait y avoir une subtilité).

J'ai donc utilisé ton code du chapitre 5, mais RIEN, il ne se passe rien du tout.

Les modules sont à 50cm l'un de l'autre. Je n'ai pas filtré l'alimentation, les deux sont alimentés par une alim stabilisée à 5VDC (utilisation des plugs d'alim).

j'ai recontrôlé je ne sais combien de fois le câblage... RIEN!

J'ai donc cherché le moyen de tester le module et utilisé le sketch 'Test_Rx' de la librairie RF24Lite. Mais là aussi rien... la com semble s'établir avec le chip via le SPI, mais rien de plus...

As-tu un moyen de contrôler chaque modules? un à un ?

PS: sauf erreur de ma part tu indiques que le wi-fi couvre 2,400 GHz à 2,483 GHz donc jusqu'au canal 83 et non 13 !?

Jérôme

15 juin 2021 à 12:04

Salut Seb !

Perso, lorsque j'ai un problème tel que le tien, je vérifie scrupuleusement :

- que le câblage du port SPI et du CE/CSN soit bien correct
- que la déclaration des entrées/sorties dans le code arduino corresponde bien avec le câblage effectué
- que les tensions arrivent bien au niveau de chaque module nRF24+PA/LNA (dans ton cas : que les convertisseurs 3,3 volts embarqués sur les cartes d'alim des NRF24 fournissent bien la tension attendue)

Car comme toi certainement, je n'ai aucun moyen de contrôler chaque module individuellement.

Aussi, ce que je ferais, c'est de prendre 2 arduinos identiques (par exemple deux Arduino Uno), que je sais parfaitement bien fonctionner. Ensuite, je brancherais un module nRF24L01 sur chacun, afin de vérifier l'état de fonctionnement des nRF24, tranquillement (car même neufs, il peut y avoir des surprises, aussi bien au niveau des cartes d'alim, que des nRF24L01).

N.B. : il ne faut pas confondre le canal 13 d'un WiFi, avec le 13ème canal

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

a+, et bon courage à toi 😊

Jérôme.

fabfab

25 juin 2021 à 22:38

Bonjour !!

Bravo pour le cours que tu as fais vraiment très bien et clair.

J'ai une petite question sur l'émission de hello world : est-ce que c'est normal quand je n'inclus pas « setDataRate » que je ne reçoive pas le message, et quand je l'inclus tout est OK ?

Merci.

fabfab

25 juin 2021 à 22:44

J'ai oublié de te dire que je n'ai pas de condo de 10uf. C'est peut-être pour cela !!

Jérôme

26 juin 2021 à 06:34

Salut Fabien !

Possible que ce soit effectivement dû à l'absence de condensateur sur l'alim du NRF24L01+ ! Car de mon côté, lors de mes essais, j'avais eu des comportements bizarres sans celui-ci. En tout cas, refait tes essais avec ce condo, car cela pourrait peut-être résoudre ton problème 😊

@+, bon courage !

Jérôme.

fabfab

29 juin 2021 à 21:51

Bonjour et merci pour ton retour. C'est bon problème réglé pour la transmission unidirectionnelle. Mais maintenant, je galère avec le bidirectionnel. Je ne reçois rien des deux côtés. Je ne sais pas si ça vient des tunnels ou pas. Alors j'ai fait un copier/coller de tes codes, mais toujours rien. Je t'avoue que je commence à caler un peu et pourtant j'ai fait pas mal

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Jérôme

30 juin 2021 à 07:12

Re !

Théoriquement, si ça fonctionne dans un sens, ça doit fonctionner dans les 2 sens ! À moins qu'il y ait un soucis de mauvais contact quelque part, ou un problème au niveau d'un module nRF24 en lui-même.

Il y a tout de même 2 choses qui me viennent à l'esprit, et que je testerai dans un cas pareil :

il faudrait tester le code unidirectionnel dans un sens, puis dans l'autre (c'est à dire faire un premier essai avec ton 1er nRF24 en émetteur et le 2nd en récepteur, puis dans un second temps, avec ton 1er nRF24 en récepteur et le 2nd en émetteur)

si le câblage est fait sur breadboard, je bougerai les fils de place, histoire d'être sûr qu'il n'y ait pas de mauvais contact quelque part

Parce qu'après, mise à part un soucis physique sur la puce nRF24L01 ou d'interférences avec le signal 2,4 GHz, je ne vois pas d'où peut provenir le problème !

À suivre 😊

Bon courage,

Jérôme.

fabfab

4 juillet 2021 à 09:20

Bonjour

Merci encore pour ton retour. Je vais essayer l'inversion de l'émetteur, le passer en récepteur et vice versa. Je ne pense pas que ça vient du câblage, car je ne suis pas sur une breadboard et j'ai soudé les câbles. Si cela ne fonctionne pas, j'ai deux autres modules. Je testerai avec les nouveaux modules. Je ferai un petit retour pour dire si ça a

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Jérôme

4 juillet 2021 à 09:40

Salut Fab !

Oui, ça m'intéresse de savoir du coup ! Alors à bientôt 😊

Jérôme.

Marc-Antoine

26 juin 2021 à 17:48

Salut, voila j'ai copié les deux codes pour l'émetteur et le récepteur, j'ai installé la librairie RF24 mais mon moniteur série ne m'affiche que « message reçu: », et je n'ai pas « Hello Word !!! ». C'est dû à la librairie SPI que je n'ai peut-être pas installé ? Car je ne sais pas laquelle installer. Pourrais-tu me la préciser stp ?

Merci d'avance.

Jérôme

26 juin 2021 à 20:28

Bonsoir !

Pour la librairie SPI.h, il n'y a rien à installer (car elle est native dans l'IDE arduino).

Par contre, c'est bizarre que ça t'affiche le texte « message reçu » (qui signifie que la liaison émetteur/récepteur est bonne), sans aucune donnée derrière.

La première explication serait que tu n'aies pas mis le condensateur de 10 µF en parallèle de l'alim, ce qui provoquerait des fluctuations de tension, et du coup, des pbs de communication.

Sinon, tu as peut-être un faux contact quelque part (si tu as fait ton montage sur breadboard, par exemple), qui ferait que ça marche par intermittence.

Perso, je commencerais par vérifier ces 2 points là !

Voilà ! Bon courage 😊

Jérôme.

Matthieu

27 juin 2021 à 19:04

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

données de 6 nRF24 (chaque nRF24 étant branché sur sa carte Mega). Sur ton tuto, cela semble possible et assez simple. Or j'ai fait plusieurs essais avec ces lignes de code :

```
// Sur nRF24 "récepteur"
const byte adresse[6][6] = {"Node1", "Node2", "Node3", "Node4",
                            "Node5", "Node6"};
radio.begin();
radio.setChannel(100);
radio.setPALevel(RF24_PA_LOW);
radio.openReadingPipe(0, adresse[0]);
radio.openReadingPipe(1, adresse[1]);
radio.openReadingPipe(2, adresse[2]);

// Sur nRF24 "émetteur"
const byte adresse[6][6] = {"Node1", "Node2", "Node3", "Node4",
                            "Node5", "Node6"};
radio.begin();
radio.setChannel(100);
radio.setPALevel(RF24_PA_LOW);
radio.openWritingPipe(adresse[1]);
```

Dans cette configuration, cela fonctionne impeccablement, mais lorsque je change « adresse[1] » en « adresse[2] » par exemple dans le programme du nRF24 « émetteur », cela ne fonctionne plus. De la même manière, seul « radio.openReadingPipe(1, adresse[1]); » sur nRF24 « récepteur » fonctionne. Si je change cela en « radio.openReadingPipe(2, adresse[1]); », cela ne fonctionne plus (avec « adresse[1] » sur nRF24 « émetteur » bien sûr).

Aurais-tu une solution ? Sinon, je pense utiliser la même adresse « const byte adresse » pour mes 6 nRF24 « émetteur », en espérant que les données ne soient pas envoyées en même temps...

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

me basant sur ton tuto.

Désolé pour ce long message (j'espère assez clair) et encore un grand merci.

Jérôme

27 juin 2021 à 20:08

Bonsoir Matthieu !

Qu'entends-tu par « cela ne fonctionne plus » ? Est-ce que tu veux dire que tu n'as aucune donnée qui semble transiter d'un émetteur au récepteur, ou bien que tu as une erreur à la compilation, lorsque tu essayes d'uploader le programme ?

Du reste, évite d'utiliser les même noms pour les émetteurs, pour éviter tout conflit de communication (comme tu l'avais déjà soupçonné !).

Enfin, les « #include » dépendent des librairies utilisées dans l'IDE d'arduino. Certaines peuvent être indispensables à rajouter, tout comme être déjà incluses, ou encore être totalement inutiles avec telle ou telle bibliothèque. Pour ma part, j'ai mis dans ce tuto tous les include qui étaient « préconisés » par celui qui a publié la librairie nRF24, utilisée ici.

À bientôt 😊

Jérôme.

Matthieu

28 juin 2021 à 17:26

Bonjour Jérôme. Merci pour ta réponse rapide.

Depuis le nRF24 « émetteur », j'envois un texte court, et dès que le nRF24 « récepteur » reçoit quelque chose, il l'affiche immédiatement sur le port série. Dans la configuration proposée dans mon ancien message, cela fonctionne parfaitement (100% de réussite), c'est à dire que je vois sur le port série (branché au nRF24 « récepteur »), le message que j'ai envoyé du nRF24 « émetteur ». Lorsque je transforme « radio.openWritingPipe(adresse[1]); » en « radio.openWritingPipe(adresse[2]); » sur le nRF24 « émetteur », cela ne fonctionne plus. Je n'ai pas de problème d'upload, mais le nRF24 « récepteur » ne reçoit plus le moindre caractère, car rien ne s'affiche sur le port série. De plus, dans cette configuration, par exemple « radio.write(« Coucou », sizeof(« Coucou »)) » présent sur le nRF24 « émetteur » renvoie la valeur « False ». Voilà si tu as une idée de piste à

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Jérôme

28 juin 2021 à 18:19

Re,

Arf... j'avoue ne pas comprendre pourquoi ça ne marche pas, désolé ...

Par contre, en creusant la question, j'ai trouvé un exemple de code que tu pourrais tester, car il devrait fortement t'intéresser. Il s'agit d'un programme arduino permettant de mettre en œuvre jusqu'à 6 émetteurs différents, communiquant sur 1 seul récepteur (comme tu voulais faire, en somme). Au passage, ce code est fourni par celui qui a créé la librairie RF24 utilisée dans ce tuto. Ce programme est téléchargeable ici : https://nrf24.github.io/RF24/examples_2MulticeiverDemo_2MulticeiverDemo_8ino-example.html (pour info : ce code est commun aux émetteurs et au récepteur, et il suffit de changer la variable « role » avec la valeur 0,1,2,3,4,5, ou 'R' pour dire si l'arduino doit se comporter en tant qu'émetteur n°0 à n°5, ou en tant que récepteur).

Et en regardant de plus près, je m'aperçois par ailleurs qu'il utilise la fonction « radio.available(&pipe) » plutôt que « radio.available() », ce qui permet de récupérer l'identifiant de celui qui émet, en quelque sorte. Je pense que ce programme pourrait vraiment t'intéresser, et répondre à tes besoins.

Du coup : pourrais-tu essayer ce programme, et voir s'il fait ton bonheur ?

@+, et tiens moi au courant 😊

Jérôme.

Matthieu

22 juillet 2021 à 17:03

Bonjour Jérôme.

Pour replacer le contexte, je voulais qu'un nRF24 « récepteur » (branché sur un ESP32) puisse recevoir des données de 6 nRF24 (chaque nRF24 étant branché sur sa carte Mega). Le code ci-dessus marchait lorsque j'utilise le pipe n°1 (avec cette ligne de code : radio.openWritingPipe(adresse[1]); pour le nRF24 « émetteur »). Cependant lorsque j'utilise les autres pipes, cela ne fonctionne plus

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

pipes par des variables de type : const uint64_t. (Après je n'ai pas assez de connaissances pour comprendre le pourquoi...)

Avec ces noms de pipes dans le code « émetteur » et « récepteur », cela fonctionne parfaitement : (c.a.d que le nRF24 « émetteur » peut utiliser toutes les adresses des différents pipes) :

```
const uint64_t pipe01 = 0xE8E8F0F0A1LL;
```

```
const uint64_t pipe02 = 0xE8E8F0F0A2LL;
```

```
const uint64_t pipe03 = 0xE8E8F0F0A3LL;
```

...

Je suis allé plus loin dans mes essais. Le nRF24 possède 3 buffers de 32 octets, c'est à dire qu'il peut garder en « mémoire », 3 messages de 32 octets chacun. Si un message arrive ensuite, (avant qu'un « radio.read() » du nRF24 « récepteur » ne soit lu par le programme) il ne sera pas pris en compte. De plus, les messages envoyés ne se mélangent jamais. En tout cas, j'ai essayé de faire en sorte que les messages soient corrompus, mais ce n'est jamais arrivé, et ce, même avec la même adresse de pipe. J'ai 2 nRF24 « émetteur » ayant la même adresse de pipe, qui envoient un message différent lorsque j'appuie sur un bouton poussoir. Le nRF24 « récepteur » (qui est branché sur un ESP32, mais cela n'a aucune influence je pense), ne « lit » (avec un radio.available() puis un radio.read(message, sizeof(message));) ce qu'il a éventuellement reçu uniquement lorsque j'appuie aussi sur un bouton poussoir. Les 2 messages sont toujours arrivés non corrompus, et même si ceux-ci faisaient 32 octets. Je pense que l'on peut utiliser des nRF24 ayant la même adresse et que cela ne va pas forcément poser problème, surtout si les nRF24 n'envoient pas des messages de manière très soutenue. En tout cas, c'est ce que laisse supposer les essais que j'ai fait.

Voilà si cela peut aider quelqu'un et merci encore Jérôme pour m'avoir aidé à résoudre mon problème.

Jérôme

22 juillet 2021 à 18:07

Salut Matthieu !

Mille mercis pour ce retour à la fois complet et précis, qui permettra à coup sûr d'aider un maximum de gens ici, qui cherchent à utiliser tous

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Très bonne soirée à toi 😊
Jérôme.

PIERRIC DOUBLET

28 juin 2021 à 06:42

Top j'ai pas tout compris mais j'ai adoré la clarté !

J'essaye de transmettre les valeurs d'un TDS (en ppm) d'un capteur qui se base sur la température d'un DS18B20. À la réception sur un affichage LCD, j'ai la température et la valeur en ppm qui s'affichent en alternance !!! Une idée ? Merci d'avance et bonne continuation.

Jérôme

28 juin 2021 à 07:52

Salut Pierrick !

Je n'aurais sous la main qu'un seul exemple à te soumettre, te montrant comment envoyer des infos analogiques/numériques d'un émetteur à un récepteur, via ondes radio. Il s'agit d'un radiocommande arduino que j'avais réalisé il y a quelques temps (consultable ici : <https://passionelectronique.fr/radiocommande-arduino-rc1/>). Tu pourrais peut-être t'inspirer du principe de transmission de données, pour transmettre tes infos de température et TDS d'un côté à l'autre.

En espérant que cela puisse t'aider un peu !

Jérôme.

bodedio

15 juillet 2021 à 01:16

Juste merci pour tous ces efforts dans l'esprit du partage

Jérôme

15 juillet 2021 à 15:44

Merci pour ce message ! Et c'est avec plaisir !

Jérôme.

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Bravo Jérôme, toutes mes félicitations pour la clarté de ton article, digne d'un enseignant. Ce que j'ai apprécié c'est que ce n'est pas une traduction prise sur Internet comme on peut, souvent, le voir avec peu d'explication concrète. Merci pour ton aide.

Jérôme

19 juillet 2021 à 20:03

Merci Jean-Pierre pour ce message chaleureux !

Et compte sur moi pour enrichir ce site un maximum, afin d'aider un maximum de personnes à faire leurs premiers pas en électronique !

Au plaisir 😊

Jérôme.

Simon

21 juillet 2021 à 08:52

Bonjour, Merci pour le tuto c'est très bien présenté.

J'ai essayé de mon côté mais rien n'y fais. Le matériel est tout neuf, j'ai fait plusieurs les même branchements que toi, j'ai le même code, et pourtant, je reçois seulement une suite de « message reçu » sans le « hello world » derrière. Il y a donc bien réception d'un message mais il est vide, et je ne comprend pas pourquoi. Je fonctionne sur les channels entre 100 et 125, et j'ai le module intermédiaire pour stabiliser la tension.

Merci de ton aide

Simon

Jérôme

21 juillet 2021 à 09:15

Salut Simon !

Pourrais-tu rajouter un condensateur de 10 ou 100 µF (en le soudant par exemple) directement sur le module NRF24L01 (après la petite carte d'alim, donc), au moins au niveau de l'émetteur ?

Parce que ton problème vient peut-être du fait que ton alim fluctue un peu trop,
~~lorsque le module nRF24 est actif~~

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Simon

22 juillet 2021 à 15:38

Salut, merci de ta réponse rapide 😊

J'ai résolu une partie du problème, il fallait que je branche l'adaptateur sur le 5v, pas sur le 3v3. J'ai encore un petit soucis qui est que je capte pas tout le temps, mais ça vient probablement des fils, je vais essayer avec le condo voir si ça change et sinon je souderai le tout pour avoir un truc solide.

Merci !! @+

Simon

Jérôme

22 juillet 2021 à 15:54

Ça marche, tiens nous au courant !

À bientôt 😊

Jérôme.

GPToky

15 août 2021 à 18:24

Bonsoir Jérôme,

Merci beaucoup pour le tutoriel et toutes les informations que tu nous donnes. J'ai actuellement un projet d'envergure moyenne à base d'Arduino : créer un genre de système permettant de capter 9 paramètres via divers capteurs (genre de Datalogger). De plus, chaque unité devrait pouvoir envoyer les réponses de trois questionnaires différents. Le but étant d'en fabriquer 30 (donc 30 Datalogger) ^^' et une de plus permettant de capter toutes ces informations et les stocker sur carte SD ou autre.

Penses-tu que les NRF24 pourraient supporter cette quantité d'informations ?

Jérôme

15 août 2021 à 19:28

Salut à toi !

Dans l'absolu, je te dirais que oui ! Parce qu'on peut sans problème monter un réseau allant jusqu'à 781 nRF24 communiquant entre eux (comme vu un peu plus

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Maintenant, la principale contrainte serait certainement la fréquence à laquelle seraient envoyées/reçues les données par chacun des NRF24. Car cela constitue une limite physique, au delà de laquelle on ne pourrait aller. Qui plus est, le débit reste également fonction des distances qui les séparent. Du coup, je ne saurais t'en dire plus, si ce n'est d'essayer de voir en pratique qu'elles pourraient être les limites d'un tel réseau 😊

Du reste, pour la partie datalogger, j'ai fait un tuto sur le sujet, qui pourrait peut-être te donner des idées ([article datalogger sur carte SD](#)).

Voilà, et désolé de ne pouvoir t'en dire plus, ou de ne pouvoir être plus précis !

N.B. : perso, j'avais dans l'idée de concevoir une mini station météo, avec plusieurs capteurs sans fils. Du point de vue réseau, ça rejoint un peu ton idée de système multi-capteurs. Mais comme j'ai énormément de priorités à gérer dans l'immédiat, je n'ai pas encore eu le temps de me pencher sur un tel montage. Celat étant dit, ... à voir lequel de nous aura créé son projet avant l'autre 😊

À bientôt ! Et surtout, bon courage à toi, et amuse toi bien !!

Jérôme.

Yannick

25 août 2021 à 21:09

Bonjour,

Très bonne explication !!! Et je me sers de ton exemple pour créer une télécommande.

Mais je ne comprends pas 2 petites choses. Pourquoi dans les exemples unidirectionnels et bidirectionnel, tu ne précises pas radio.setDataRate(xxx) et radio.setChannel(x) sur ton code arduino ?

Merci de m'éclairer.

Jérôme

26 août 2021 à 06:58

Salut Yannick !

En fait, tu peux très bien ne pas appeler ces fonctions, si tu veux faire simple et rapide. Car ces fonctions sont automatiquement appelées lorsque tu exécutes la

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

À noter que ces valeurs sont propres à la librairie arduino utilisée ici, et que cela pourrait être différent, si tu utilisais une autre bibliothèque.

Cela étant dit, je pense effectivement qu'il vaudrait mieux prendre l'habitude de toujours appeler ces fonctions en début de chaque programme arduino, afin de définir soi-même les fréquences et vitesse de fonctionnement que l'on souhaite (ce que j'aurais dû faire dans les deux premiers exemples, tu as raison, même si cela n'avait pas d'utilité particulière dans ceux-ci !).

Voili voilou !

Jérôme.

Adrien

[5 septembre 2021 à 21:14](#)

Bonjour,

Est ce possible d'augmenter le nombre d'enfants qui ,si j'ai bien comprit, est au maximum de 5?

Jérôme

[6 septembre 2021 à 06:43](#)

Salut Adrien !

En fait, tu n'es pas limité à 5 enfants, si tu optes pour un montage en arborescence (comme détaillé dans le paragraphe 7). Ainsi, tu pourrais monter jusqu'à 780 enfants, en quelque sorte.

Bonne journée à toi !

Jérôme.

Adrien

[7 septembre 2021 à 06:24](#)

Oui mais justement c'est marqué « chaque nRF24 ne peut avoir que 5 modules nRF24 « enfant », sous lui ». On peut effectivement avoir plus de 5 enfants mais obligatoirement si les autres « parents » sont actifs?

Jérôme

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Mais je pense qu'il y a confusion entre la notion de parent/enfant, et le nombre de connexions radio que peut avoir un module nRF24 avec d'autres (qui pour rappel est : 1 en émission/réception, et les 5 autres, en écoute seulement).

Car le mot parent n'est qu'une manière d'exprimer les choses. En fait, dans le cas d'un montage en arborescence, chaque parent (qui peut avoir 5 enfants) est lui-même enfant d'un autre parent (sauf pour le premier, qui est tout en haut de la « pyramide »). Donc au final :

- si tu veux savoir si un module nRF24 peut avoir plus de 5 enfants, alors la réponse est NON
- mais si tu veux savoir, si un module nRF24 peut communiquer avec plus de autres 5 modules nRF24, alors la réponse est OUI

En fait, dans le cas d'une architecture en arborescence, tu pourrais très bien imaginer avoir 1 récepteur et 780 émetteur (ou vice-versa, car tout est libre ici). Le tout est simplement de leur donner des adresses spécifiques, qui respectent bien le plan en arborescence. Et comme tu l'as bien compris, chaque module nRF24 doit être alimenté, et branché sur un Arduino ou autre, faisant tourner la librairie nRF24-Network.

En espérant que ce soit plus clair 😊

À bientôt !

Jérôme.

Adrien

12 septembre 2021 à 16:11

Merci beaucoup pour ton aide, je vais bientôt essayer et si je bloque je reviendrais vers toi 😊

Stephane

6 septembre 2021 à 12:49

Bonjour Jérôme,

Merci pour ce génial tuto, je suis en train d'essayer de communiquer entre un Arduino ATMEGA328P et un ESP32... et ton tuto traite tous les points avec clarté. Je pense donc pouvoir m'en sortir beaucoup plus facilement maintenant.

Bien à toi

...

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Jérôme

6 septembre 2021 à 14:57

Ah cool, si ça peut t'aider ! Par contre, je t'avouerai que je n'ai pas testé avec un ESP32. En espérant qu'il n'y ait pas de soucis à ce niveau !

@+

Jérôme.

Pierre

13 octobre 2021 à 13:59

Bonjour voici quelques temps que je cherche en vain une manière de faire fonctionner les modules RF24 malgré de nombreux échecs, cependant votre documentation est bien plus développée comparé à d'autres blogs, je vous en remercie malheureusement cela ne fonctionne toujours pas, j'ai vérifié plusieurs fois le câblage et relu les explications mais rien, si vous avez une idée du problème je suis preneur merci d'avance 😊

Cordialement FEBVAY Pierre

(PS: j'utilise deux arduino méga et mes modules RF24 sont tous deux équipés du module d'alimentation intermédiaire)

Jérôme

14 octobre 2021 à 20:20

Bonsoir Pierre !

Je vais refaire le même circuit que toi demain si j'ai le temps, au plus tard ce weekend, pour voir si j'ai le même problème avec des Arduino Méga + nRF24 avec module d'alimentation intermédiaire.

Je te tiens au courant.

À très vite !

Jérôme.

Jérôme

16 octobre 2021 à 10:11

Re

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)

même via sa petite carte d'alim, embrochable dessus. Car je viens de faire des essais de mon côté, et sans une alim externe au niveau de l'arduino, il peut y avoir des problèmes de communication. Comme quoi... le fait que ça marche, ou ne marche pas, tient parfois à peu de choses ! Et il faut persévérer !

Du coup, je pense que, dès que j'aurais le temps, je complèterai mon article en ajoutant :

- un paragraphe pour expliquer comment raccorder le nRF24 PA LNA (via sa petite carte d'alim) à un Arduino Méga, pour ceux qui auraient des doutes
- et un autre paragraphe, sur les problèmes qu'on rencontre le plus fréquemment avec ces modules NRF, et comment passer outre

[MISE À JOUR] C'est fait, j'ai rajouté les 2 paragraphes en question !

Voilà !

Jérôme.

PIERRE

22 octobre 2021 à 16:56

Merci beaucoup je vais regarder et je te tiens au courant ! 😊

Petite suggestion pour ton site : pourrais-tu faire un post expliquant la liaison SPI plus en détail ? Tu l'utilise avec tes câbles MISO et MOSI si je ne me trompe pas mais j'aimerais en savoir un peu plus si possible.

En tout cas merci beaucoup ton code est très clair et tes explications très compréhensibles, bravo !

Cordialement FEBVAY Pierre

PIERRE

22 octobre 2021 à 17:58

RF : je viens de tester après avoir tout désassemblé et pris le temps de

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Jérôme

22 octobre 2021 à 18:36

Ah, super ! Comme quoi, la frontière entre « y'a rien qui marche » et « tout fonctionne nickel » est parfois mince !
En tout cas, bravo pour ta persévérance !

Bonne soirée.

Jérôme.

Jérôme

22 octobre 2021 à 18:34

Oui, c'est prévu ! Je m'étais noté de faire un article « détaillé » sur les liaisons séries (UART, I2C, SPI, OneWire, ...). En fait, c'est le temps qui me manque cruellement. Du coup, je fais au mieux, en sachant que j'ai pas mal d'ébauches d'articles qui attendent également d'être achevées !

Mais ça viendra un jour 😊

Jérôme.

David

20 novembre 2021 à 16:25

Bonjour Jérôme,

Merci pour ce tuto.

Est ce que l'on peut utiliser un nRF24 PA LNA sur un arduino MEGA en émetteur et des nRF24L01 + sur uno ou mega.

J'ai en projet de réaliser une radiocommande pour des camions RC (je vais m'inspirer de ton tuto), ma question est la suivante :

est-il possible d'utiliser un seul émetteur pour commander différents récepteurs indépendamment via une sélection par le programme de l'émetteur ?

Merci d'avance.

David

Jérôme

20 novembre 2021 à 18:18

— — — — —

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Et qu'il s'agisse d'Arduino Uno ou Mega n'a pas d'influence ici, d'après moi (puisque'on dialogue simplement en SPI, autant sur l'un que sur l'autre).

Par contre, perso, je te déconseillerais de mixer les modèles nRF24L01+ et nRF24L01+PA+LNA, car le bon fonctionnement n'est franchement pas garanti.

En effet, il peut y avoir des bugs de transmission, entre ces différents modules (parfois dus aux librairies utilisées, parfois dus au fait que les nRF24L01+ ne sont pas vraiment des modèles « + », malgré ce qu'en disent les vendeurs).

Maintenant, si tu aimes bidouiller, tu peux toujours essayer avec un premier émetteur +PA+LNA et un seul récepteur +. Et voir ce que ça donne, avant de passer à « plus grande échelle ».

Enfin, tu peux effectivement utiliser un seul émetteur avec plusieurs récepteurs. Mais par contre, si ces derniers ont la même adresse, il faudra que tu désactives la « demande d'accusé de réception », au niveau de l'émetteur ; cela se fait via la commande « setAutoAck(false) », dans le code arduino.

Voilà !

Bonne soirée.

Jérôme.

david

21 novembre 2021 à 09:13

Merci jérôme.

Juste pour être sûr : peut-on sélectionner le récepteur auquel l'émetteur s'adresse ?

Pour mon projet, je voudrais pouvoir commander 4 camions (allumés en même temps) avec une seule radio. Il faut donc que je puisse sélectionner depuis l'émetteur quel engin je veux faire bouger.

David

Jérôme

21 novembre 2021 à 10:02

Salut 😊

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Par exemple : si tu envoies un message à plusieurs récepteurs allumés en même temps, mais que tu souhaites qu'un seul des récepteur le considère pour lui, alors une solution simple est d'intégrer quelque chose dans ton message qui fasse que le destinataire comprenne que c'est pour lui (et pour que tous les autres récepteurs l'ignore).

Pour bien comprendre, tu pourrais imaginer quelque chose à l'image de la poste :

- l'adresse (postale) permet de différentier tout le monde (les destinataires)
- mais s'il y a plusieurs destinataires (habitants) à la même adresse, alors tu pourrais rajouter quelque chose qui fasse qu'ils sachent à qui s'adresse ta correspondance (le n° d'appart, par exemple, ou plus basiquement, le nom et prénom de la personne visée).

Voilà !
Jérôme.

Nicolas Bouchet

23 novembre 2021 à 12:02

Bonjour, étant débutant dans le domaine arduino, j'aimerai avoir un peu d'aide avec ce module. Je voudrais envoyer beaucoup de données (64 octets) entre un arduino et un autre arduino. Et grâce a vos explications très complètes qui mon permis de réussir à comprendre la base de fonctionnement du nrf24l01, j'arrive à bien les faire communiquer sans problème. Cependant je n'arrive pas à pouvoir envoyer et recevoir autant de données (64 octets) sachant qu'on peut émettre que 32 octets par structure. Donc je me suis dit qu'il fallait envoyer deux structures mais le soucis c'est que quand j'envoie les deux structures l'une après l'autre, au moment de la réception les valeurs des deux structure se mélangent. Je vous remercie si quelqu'un aurait une solution la plus simple, même si c'est jamais très simple dans la programmation. Pour résumer mon but c'est d'envoyer deux structures de 32 octets chacune et de recevoir ces deux structures sur l'autre arduino sans avoir un mélange de valeurs à la réception.

Cdt Nicolas

Jérôme

23 novembre 2021 à 12:36

Salut Nicolas,

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Sinon, s'il y a vraiment mélange des infos, j'essayerai de travailler avec une vitesse de transmission plus lente, afin de voir si le problème ne vient pas d'une vitesse trop élevée, en pratique.

Bon courage 😊

Jérôme.

Laurent M.

23 décembre 2021 à 15:18

Bonjour Jérôme,
Un seul mot : BRAVO !
Ce tuto est tooop !

Jérôme

23 décembre 2021 à 16:14

Merci, et de rien ! C'est partagé avec plaisir !

Daniel CLEMENT

2 janvier 2022 à 17:12

Un grand merci pour ce tuto très accessible pour quelqu'un de niveau moins que moyen.

Jérôme

2 janvier 2022 à 19:01

De rien ! Mais ne pense surtout pas que ton niveau est « moins que moyen », si tu as pigé ce tuto ! Car le contenu présenté ici n'était pas forcément si facile que ça à assimiler, de prime abord. Donc si tu as trouvé ce tuto « très accessible », c'est déjà que tu as un niveau que bien d'autres aimeraient avoir 😊

Alors ne te dévalorise pas, et à bientôt !

Stéphane

17 janvier 2022 à 15:26

Bonjour Jérôme,

Merci beaucoup pour ce tutoriel !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

```
RF24NetworkHeader nHeader(noeudCibleA);  
network.write(nHeader, &dataToSendA, sizeof(dataToSendA));  
delay(5);  
  
RF24NetworkHeader nHeader(noeudCibleB);  
network.write(nHeader, &dataToSendB, sizeof(dataToSendB));  
delay(5);
```

Je reçois cette erreur : redeclaration of 'RF24NetworkHeader nHeader'

Savez-vous comment je peux faire ?

Merci !

Jérôme

17 janvier 2022 à 15:39

Salut Stéphane !

Le message d'erreur t'indique en fait que tu ne peux pas redéclarer 2 fois le même nom de variable (en l'occurrence « nHeader »). Du coup, il te suffit simplement de donner 2 noms différents, à ta variable. Ainsi, tu pourrais par exemple réécrire ton code de la manière suivante :

```
RF24NetworkHeader nHeaderA(noeudCibleA);  
network.write(nHeaderA, &dataToSendA, sizeof(dataToSendA));  
delay(5);  
  
RF24NetworkHeader nHeaderB(noeudCibleB);  
network.write(nHeaderB, &dataToSendB, sizeof(dataToSendB));  
delay(5);
```

Ainsi, tu ne devrais plus avoir ce message d'erreur !

@+

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Stéphane

17 janvier 2022 à 16:45

Ça marche !!

Merci beaucoup Jérôme !

Jérôme

17 janvier 2022 à 17:20

De rien, au plaisir 😊

Michel

18 janvier 2022 à 08:51

Merci Jérôme pour ce tuto très bien réalisé. Tous les bricoleurs aimeraient avoir pour tous les modules Arduino ce genre de tuto, car parfois les pages GitHub sont étourdissantes pour les néophytes comme moi. Mais je sais que c'est un travail important pour développer cela.

J'ai réalisé un petit réseau pour gérer le fonctionnement de mes 9 volets roulants. J'ai pas mal galéré avec les problèmes que tu as signalés. Si j'avais eu ton tuto avant, j'aurais gagné beaucoup de temps. Par contre, je n'utilise que 2 tunnels (émission, réception) pour l'ensemble ; le central envoie une structure dans laquelle un champ est l'adresse du volet, un autre est le code de la manœuvre à exécuter. Le central orchestre tout, volet par volet, de sorte qu'il n'y a aucun conflit. C'est une solution intéressante, car simple et non limitée en nombre de points.

Jérôme

18 janvier 2022 à 09:08

Salut Michel !

Tout d'abord merci à toi, pour ce retour !

Et oui, ta solution est excellente, et bien pensée. Car effectivement, si on s'arrange à mettre l'identifiant de « qui dit quoi » dans les données en elles-mêmes, on peut grandement simplifier les choses. Par contre, cela impose de bien réfléchir à son protocole de communication, afin d'éviter toute cacophonie ! Et je vois que tu y es parfaitement arrivé, donc un grand bravo à toi, pour cette « petite » installation parfaitement réussie !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Stéphane

18 janvier 2022 à 18:11

Bonjour Jérôme,

J'aurais encore besoin de votre aide 😊

J'aimerais envoyer des données différentes à plus de 6 nœuds depuis un seul transmetteur.

Je parviens à envoyer depuis le noeud 0 des données au noeud 1 jusqu'au noeud 5, mais je n'arrive pas à envoyer au noeud 11 etc. Par exemple, si j'ai deux récepteurs, le code suivant pour le transmetteur fonctionne si noeudCibleB à la valeur 02, mais pas avec la valeur 011 (en ayant bien modifié le code pour 1er récepteur pour que « numéroDeCeNoeud » soit bien 011 dans ce cas) :

```
RF24NetworkHeader nHeaderA(noeudCibleA=01);  
network.write(nHeaderA, &valA, sizeof(valA));  
  
RF24NetworkHeader nHeaderB(noeudCibleB);  
network.write(nHeaderB, &valB, sizeof(valB));
```

Vous voyez ce que je veux dire ?

Merci !

Jérôme

18 janvier 2022 à 18:56

Bonsoir !

C'est bizarre, car tout devrait bien fonctionner !

Mais pour être sûr de bien comprendre : tu as 3 appareils branchés en réseaux, avec une topologie telle que :

- il y a 1 émetteur au nœud 00
- il y a 1 récepteur au nœud 01
- et il y a 1 autre récepteur au nœud 011

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

« branché » au nœud 01 n'était pas alimenté au moment de tes essais entre le nœud 00 et le 011 ? (car c'est lui qui est chargé du relayage des infos)

Jerome.

Stéphane

19 janvier 2022 à 11:34

Merci beaucoup pour ton retour, oui ça marche bien en fait ! C'était un problème d'alimentation insuffisante (j'utilisais une pile 9V pour un émetteur et 2 récepteurs, tous équipés de module d'alimentation).

Merci !

Lionel R.

20 janvier 2022 à 11:12

Bonjour Jérôme,

Tout d'abord un grand merci pour ce super tutorial en français !

Je m'arrache les cheveux pour faire communiquer 2 arduino nano.

J'utilise 2 arduino nano. J'ai équipé mes modules NRF24L01+ d'un module intermédiaire d'alimentation (au lieu d'utiliser un condensateur) pour garantir la stabilité de l'alimentation. Mes 2 arduinos sont branchés à mon PC via prise USB.

J'ai lu bcp de posts disant qu'il fallait une alimentation externe supplémentaire.

J'ai donc mesuré la tension sur les 2 modules NRF24L01+ à l'aide d'un multimètre et j'obtiens 3.29V d'un côté et 3.28V de l'autre, donc je pense qu'on peut exclure tout problème d'alimentation insuffisante (correct ?).

J'ai aussi plusieurs fois vérifié mon câblage sans trouver d'erreur et effectué mes tests avec d'autres module NRF24 sans succès.

J'ai enfin utilisé la commande `radio.printPrettyDetails()` pour obtenir les infos concernant les module NRF24 :

Sur l'émetteur :

SPI Frequency	= 10 Mhz
Channel	= 2 (~ 2402 MHz)

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

```

Address Length          = 5 bytes
Static Payload Length  = 32 bytes
Auto Retry Delay       = 250 microseconds
Auto Retry Attempts    = 3 maximum
Packets lost on current channel = 0
Retry attempts made for last transmission = 0
Multicast               = Disabled
Custom ACK Payload      = Disabled
Dynamic Payloads        = Disabled
Auto Acknowledgment    = Enabled
Primary Mode            = TX
TX address              = 0x3145504950
pipe 0 ( open ) bound   = 0x3145504950
pipe 1 ( open ) bound   = 0xc2c2c2c2c2
pipe 2 (closed) bound   = 0xc3
pipe 3 (closed) bound   = 0xc4
pipe 4 (closed) bound   = 0xc5
pipe 5 (closed) bound   = 0xc6

```

Sur le récepteur :

```

SPI Frequency           = 10 Mhz
Channel                 = 2 (~ 2402 MHz)
RF Data Rate             = 250 KBPS
RF Power Amplifier       = PA_MIN
RF Low Noise Amplifier  = Enabled
CRC Length               = 16 bits
Address Length           = 5 bytes
Static Payload Length    = 32 bytes
Auto Retry Delay         = 1500 microseconds
Auto Retry Attempts      = 15 maximum
Packets lost on current channel = 0
Retry attempts made for last transmission = 0
Multicast               = Disabled
Custom ACK Payload      = Disabled

```

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

```
pipe 0 ( open ) bound      = 0x3145504950
pipe 1 ( open ) bound      = 0xc2c2c2c2c2
pipe 2 (closed) bound      = 0xc3
pipe 3 (closed) bound      = 0xc4
pipe 4 (closed) bound      = 0xc5
pipe 5 (closed) bound      = 0xc6
```

P.S. au passage j'ai constaté que mes 2 cartes arduino, pourtant strictement identiques, avaient des valeurs par défaut différentes pour le channel et le data rate, ce que j'ai résolu en définissant ces valeurs dans mon script.

Aurais-tu une idée ou une piste pour identifier la cause ?

Jérôme

20 janvier 2022 à 11:41

Salut Lionel,

En fait, en premier, j'éliminerai le doute sur l'alimentation, en rajoutant un condensateur. Car ton multimètre ne te donne peut-être qu'une valeur moyenne de ta tension d'alimentation, et ne te permet pas de voir de potentielles chutes de tensions brèves et passagères, qui seraient à l'origine de tous tes problèmes.

Sinon, je n'ai malheureusement aucune autre explication à te donner, quant à ce dysfonctionnement... désolé !

Bon courage à toi,

Jérôme.

Lionel R.

21 janvier 2022 à 08:22

J'ai fini par changer l'une des 2 cartes arduino et oh miracle, cela fonctionne. L'alimentation n'était pas en cause ...

Donc les différences au niveau de la configuration par défaut de l'une des 2 cartes était la cause.

C'est bon à savoir ...

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Alors pour commencer, « génial tes explications » 😊 Et bonjour !

Je suis en cours d'apprentissage du langage C et d'arduino.

780 module connectés excellent, serait-il possible de faire un équivalent avec d'autres moyens de communication (genre infrarouge, wifi et bluetooth), question théorique (je chercherai de mon côté).

Et ensuite, je suis sur esp32 et la bibliothèque « servo.h » me dit que c'est que pour avr (donc puce arduino).

Aurais-tu une autre bibliothèque pour esp (esp32 et esp8266) ? Merci.

J'ai commandé des arduino nano avec le 24L01 intégré, je referai des test avec ceux là.

Je dévore ton site depuis 2 jours.

J'attends les prochains articles de ton site avec bcp de plaisir.

Jérôme

9 février 2022 à 12:41

Salut !

Alors, pour l'instant, je ne suis pas assez calé pour te répondre au niveau de l'infrarouge et du bluetooth, ainsi que sur l'ESP32, malheureusement. Du coup, je préfère te dire que « je n'en sais rien », plutôt que te dire des bêtises, ou des approximations.

Désolé !

Du reste, bon courage à toi dans ton apprentissage, et à bientôt !

Jérôme 😊

Peaceful Crumble

10 février 2022 à 09:26

Bonjour Jérôme,

Super tutoriel, comme tout ce qui est sur ton site ! Merci !

Je vois que dans le programme « hello world » de l'émetteur, tu n'as pas défini le débit de données ni le canal Est-ce parce que il y a une valeur par défaut ?

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Jérôme

10 février 2022 à 09:55

Salut !

Oui, c'est exactement ça ! En fait, il y a des valeurs par défaut dans cette librairie, qui sont :

- 1 Mbps, pour le débit de données
- et 76, pour le canal de transmission

En fait, j'ai simplement laissé ces paramètres « secondaires » de côté dans mes « exemples de base », afin que chacun puisse faire ses premiers pas avec, sans prise de tête. Mais bien évidemment, libre à chacun de prendre ensuite le contrôle du nRF24 plus en profondeur, en jouant sur toutes les autres fonctionnalités disponibles 😊

Jérôme.

Gus

20 juin 2022 à 09:53

Bonjour,

Pour un projet étudiant nous essayons de faire fonctionner ces modules (en suivant vos explications à la lettre 😊). Mais le module récepteur n'affiche que le « message reçu » et pas le « hello world ». Nous avons pourtant vérifié notre câblage et rajouté un condensateur en parallèle. Que faire de plus ?

Jérôme

20 juin 2022 à 11:09

Salut Augustin !

Alors, toutes les fois où j'ai rencontré ce problème, l'origine se situait au niveau de l'alimentation électrique.

En fait, les problèmes de transmission constatés étaient dus :

- soit à une alimentation insuffisamment stable, auquel cas un simple condensateur pouvait pallier à problème (en lissant les fluctuations de

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

nos cartes Arduino, qui ne peuvent malheureusement pas toujours délivrer toute la puissance nécessaire)

Donc, à ta place, je commencerai par examiner de près ce qui se passe côté alim, avant de chercher plus loin 😊

Bon courage à toi.

Jérôme.

Luc ROLLAND

27 décembre 2022 à 18:14

Bonjour,

D'abord un très grand merci pour la qualité de ce tutoriel.

Je bute sur l'adressage arborescent : les enfants conservent les 4 premiers octets (donc ceux de gauche) de l'ID de canal. Cela va pour le premier niveau : si Master 0x000000000000 : enfants 0x0000000001 – 0x0000000005. Mais pour l'enfant 0x0000000001 les 4 premiers octets (ceux de gauche) restent à 0x00000000 !? Ca marche si on prend les 4 derniers octets (ou décalage de 8 bits vers la gauche) on obtient alors 0x00000001 et ça fonctionnera pour les niveaux suivants ...

Il y a donc quelque chose que je n'ai pas compris, pourrais-tu éclairer ma lanterne ?

D'avance merci.

Luc

Jérôme

27 décembre 2022 à 19:50

Salut Luc !

Je pense que tu mélanges les adressages en arborescence, avec l'adressage des pipes (qui, pour certains d'entre eux, conservent leurs 4 premiers octets). En fait, ce sont deux choses différentes, même si elles sont relatives à l'adressage toutes les deux.

Concernant l'adressage en arborescence, il est typé de la manière suivante :

- Niveau 0 : l'adresse du nœud maître est 0

• Niveau 1 : l'adresse du nœud père est 1

• Niveau 2 : l'adresse du nœud père est 2

• Niveau 3 : l'adresse du nœud père est 3

• Niveau 4 : l'adresse du nœud père est 4

• Niveau 5 : l'adresse du nœud père est 5

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

- 13, 23, 33, 43, et 53 pour l'enfant 3
- 14, 24, 34, 44, et 54 pour l'enfant 4
- 15, 25, 35, 45, et 55 pour l'enfant 5
- Niveau 3 : les adresses des enfants des petits-enfants du maître sont :
 - 111, 211, 311, 411, et 511 pour le petit-enfant 11
 - 112, 212, 312, 412, et 512 pour le petit-enfant 12
 - ...
 - 155, 255, 355, 455, et 555 pour le petit-enfant 55
- Niveau 4 : même logique que précédemment (on ajoute un chiffre allant de 1 à 5, devant l'adresse des parents, pour obtenir l'adresse de l'enfant)
 - 1111, 2111, 3111, 4111, et 5111 pour le petit-petit-enfant 111
 - ...
 - 1555, 2555, 3555, 4555, et 5555 pour le petit-petit-enfant 555

En espérant avoir pu t'éclairer un peu, et surtout, bien répondu à ta question 😊

Jérôme.

Paul MAGALON

26 septembre 2023 à 20:15

Bonjour Jérôme et félicitations pour ce tuto « grand modèle ».

J'ai une petite question technique a vous soumettre. Voila ...

J'aimerai envoyer a partir d'un module 'maître' une information identique a un cinquantaine de modules 'esclaves' distants d'une centaine de mètres. Ces modules 'esclaves' ne feront qu'écouter les ordres du 'maître' et ne renverront pas d'accusé de réception. Pour le module 'maître' je vais évidemment utiliser un circuit nRF24 PALNA avec une alimentation bien calibrée, mais par contre, et c'est la qu'arrive ma question, pour chacun des modules 'esclave', le nRF24 de base (avec antenne gravée dans le circuit imprimé) est-il suffisant ou faudra t-il que j'utilise aussi des nRF24 PALNA ?

Merci d'avance pour votre réponse et bonne continuation.

Paul.

Jérôme

27 septembre 2023 à 07:21

Salut Paul !

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

fonctionne pas. Du coup, cela t'éviteras peut-être d'acheter une cinquantaine de modules qui ne correspondraient pas à tes besoins, par ailleurs 😊

Bon courage à toi !

Jérôme.

JiHelB

18 décembre 2023 à 10:01

Bonjour Jérôme et merci pour ce tuto plein d'enseignement, comme d'habitude.

J'ai une petite incompréhension : j'ai cru comprendre qu'on pouvait envoyer des messages avec un accusé de réception automatique. Comment fait-on ? Comment sait-on si le message a été reçu ou non ?

Merci d'avance si tu peux éclairer ma vieille lanterne.

Et surtout bonne continuation,

JiHelB

Jérôme

18 décembre 2023 à 11:14

Salut !

Alors, avec la librairie que j'ai partagé dans cet article, l'activation/désactivation d'accusé de réception automatique se fait via la fonction `setAutoAck()` ; tu trouveras [pas mal d'infos ici](#), au besoin). À noter que l'AR est actif par défaut, donc on ne s'en soucis pas vraiment (sauf à vouloir le désactiver, en fait).

Du reste, il va falloir que je me replonge dans la documentation, sinon je risquerais te dire des bêtises. Cela dit, je crains de ne pas avoir le temps dans l'immédiat, car j'ai énormément de travail en retard, et qui urge ! Désolé ...

Bon courage à toi,

Jérôme.

Maxime

27 décembre 2023 à 22:09

Bonjour Jérôme,

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Cependant, j'ai pris part de votre réponse à Gus et j'ai rajouté un module d'alimentation intermédiaire pour stabiliser la tension sur les 2 modules et j'ai branché les 2 Arduino UNO sur une multiprise qui elle même est branché sur secteur pour plus de tension. J'ai également regardé toute mes soudures et ils sont tous bonne.

Que puis-je faire de plus ?

Merci d'avance.

Maxime.

Jérôme

28 décembre 2023 à 07:00

Salut Maxime,

Hum... alors, dans un cas comme le tien :

- je ferai vérifier le câblage et connexions par un autre (parfois, un œil « externe » peut remarquer des choses auxquelles on n'a pas fait attention)
- je déplacerai le montage, « loin » de tout appareil électrique (pour voir s'il n'y a pas des interférences particulières, qui viendraient perturber la communication)
- j'essaierai avec d'autres cartes NRF24, de même type (pour voir s'il n'y a pas un problème matériel), mais également de modèles différents (PA LNA, et non PA LNA, je veux dire)

En espérant que cela puisse t'aider, sinon je crains de ne pouvoir t'aider davantage

...

Bon courage à toi,

Jérôme.

Romain

29 décembre 2023 à 11:15

Bonjour Jérôme,

Auriez vous s'il vous plaît un exemple de code pour actionner un relai avec un bouton poussoir (un appuis on / un deuxième appuis off) utilisable avec le module NRF24L01 PA LNA ?

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Non, désolé ... par contre, je me note de faire un projet en ce sens, dès que j'aurais du temps (genre un émetteur à 4 boutons poussoirs, avec récepteur à 4 relais). Comme ça, vous aurez tous un exemple de réalisation et de codage en support, pour la mise en pratique des NRF24L01 PA LNA !

Par contre, ce n'est pas pour tout de suite, malheureusement (j'ai trop de boulot en retard et urgent, là ...).

Encore désolé,
Jérôme.

Romain

29 décembre 2023 à 17:22

Merci beaucoup pour ton retour, j'attends ton projet avec impatience. Je continue mes recherches de mon côté. Bonne journée à toi.

Jérôme

29 décembre 2023 à 18:51

Ça marche ! Très bonne soirée à toi !

Xavier

26 janvier 2024 à 10:42

Bonjour Jérôme

Tuto impressionnant par sa qualité et ses détails.

Je commence à m'intéresser à ce type de communication mais j'ai une incompréhension avec les canaux et leurs fréquences. Au paragraphe 2.1 tu écris « canal 1 freq 2400MHz » et un peu en dessous « fréquences allant de [2,401 GHz (canal 1) à 2,483 GHz (canal 13)] ». Le début c'est canal « 0 freq 2400 MHz » ? et d'où vient cette fréquence de 2483 MHz qui devrait être 2413MHz !!! Au paragraphe 4.3 je trouve le canal 0 à 2400 MHz !

Rien de très important. Merci pour une éventuelle explication et correction du texte si besoin.

A bientôt

Xavier

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Salut Xavier !

Excellent remarque, très pertinente ! Depuis que j'ai publié cet article, personne n'avait relevé cette erreur !!

En effet, c'est bien 126 canaux (et non 125) qui nous permettent de choisir une fréquence entre 2400 et 2525 MHz. Il y a donc 126 channels, allant de 0 à 125.

Par contre, j'ai enlevé mon aparté sur les canaux WiFi (qui vont de 1 à 14 maintenant), car c'était source de confusion, avec les canaux du NRF24. Par contre, j'ai laissé apparaître les fréquences WiFi, qui vont de 2412 à 2484 MHz (source : https://fr.wikipedia.org/wiki/Liste_des_canaux_Wi-Fi), pour bien montrer leur « chevauchement ».

Voilà !

Et encore merci, pour cette remontée d'info 😊

Jérôme.

SAHL

10 mars 2024 à 22:58

Merci beaucoup de ce tutoriel, votre petit mot à la fin m'a beaucoup touché, vous partagez vos connaissances et en même temps vos valeurs.

Jérôme

11 mars 2024 à 08:35

Merci, et de rien ! C'est partagé avec plaisir !

Laisser un commentaire

Votre adresse e-mail ne sera pas publiée. Les champs obligatoires sont indiqués avec *

Commentaire *

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

Accepter Refuser

Nom *

E-mail *



Enregistrer mon nom, mon e-mail et mon site dans le navigateur pour mon prochain commentaire.

Résolvez cette soustraction : 75 - = 72

[Laisser un commentaire](#)

Afin de filtrer au maximum les messages de type "spam" ou "inappropriés", **chaque commentaire est soumis à modération, et validé manuellement**. Du coup, il se peut que certains commentaires ne soient pas publiés, ou sinon, avec un peu de retard. Par ailleurs, j'ai malheureusement plus de messages à traiter que de temps pour y répondre : c'est pourquoi je

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)

[Accepter](#) [Refuser](#)

© Copyright 2020-2024, **PassionElectronique.fr**
Tous droits réservés.

Certains liens présents sur ce site sont affiliés. Cela permet, même si ça reste très limité, d'aider au financement de ce site. Sachez que ceux-ci ne vous coûtent rien, et que vous restez totalement libre de les utiliser ou pas. Du reste, merci à vous tous, notamment pour vos nombreux messages bienveillants, et chaleureux ! Encore merci !

Le site passionelectronique.fr participe au Programme Partenaires d'Amazon, un programme développé pour permettre aux sites internet de percevoir une rémunération grâce à la création de liens vers le site officiel Amazon.fr. En tant que Partenaire Amazon, je réalise un bénéfice sur les achats remplissant les conditions requises.

Raccourcis : **Tous les articles, Mentions légales, Politique de confidentialité, et Page contact**

Nous utilisons des cookies pour vous garantir la meilleure expérience sur notre site web. Si vous continuez à utiliser ce site, nous supposerons que vous acceptez sans réserve notre [politique de confidentialité](#)