

*mini  
Livre Tuto*



**BANDES LED**  
*Montage et programmation*

---

*Nad Cosplay*  
mimigyaru.c♡m

# PRÉFACE

Mettre de la lumière dans un cosplay, c'est classe et ça apporte un truc en plus qui fait toujours son petit effet auprès du public. Mais il est parfois difficile de se lancer car l'électronique, ça peut paraître compliqué.

Dans ce livre, j'aborderai avec vous les « bandes LED », comment les connecter et les programmer pour créer des effets lumineux sur votre costume, vos accessoires ou même votre décor.

Il existe de nombreux types de bandes LED et c'est facile de s'y perdre quand on débute ! Ce livre a pour but de vous présenter celles que j'utilise principalement et de vous guider dans vos choix afin de mener à bien votre projet.

Je n'ai pas la prétention de tout connaître sur les bandes LED, loin de là, mais je vais essayer de partager avec vous mon expérience sur celles que j'utilise le plus souvent et comment je les fais fonctionner.

A l'issue de ce livre vous devriez avoir les clés pour dompter cet accessoire qui vous fera briller de mille feux en convention.



NAD

# SOMMAIRE

<b>I. Bien choisir sa bande LED</b> .....	4
1. Généralités .....	4
2. Avec 3 ou 4 pins ?.....	5
3. IP quésaco ?! .....	6
4. 30/m, 60/m, etc. : que choisir ! .....	7
<b>II. Les bandes RGB simples et efficaces</b> .....	9
1. Les alimenter .....	9
2. Les contrôler .....	10
<b>III. Les bandes WS2812b à programmer</b> ..	12
1. Les cartes de programmation .....	12
2. L'alimentation .....	14
3. La connexion .....	15
4. Les bases de la Programmation.....	20
5. Programmer vos LED.....	28
<b>IV. Aller plus loin</b> .....	37
1. Mise en pratique.....	37
2. DIY : Connecteur USB .....	39
3. Le mot de la fin.....	41

# I. Bien choisir sa bande LED

Pour correctement choisir sa bande LED, il faut savoir ce que l'on veut en faire !

En cosplay, vous ne serez que rarement branchés à une prise électrique donc oubliez les bandes 12V, même s'il existe des batteries 12V qui sont relativement discrètes.

Si vous avez besoin de programmer un peu avec des cartes de programmation, le 5V sera beaucoup plus adapté et largement suffisant.

Si la programmation vous fait peur (et je vais vous montrer par la suite que ce n'est pas une montagne infranchissable), il existe des bandes qui peuvent changer de couleur via une télécommande ou même avec une application sur téléphone !

Mais avant toute chose, détaillons ce qui compose une bande LED pour bien comprendre ce dont traite cet ouvrage.

## 1. Généralités

Une bande LED, ruban LED ou même bandeau LED est une bande flexible composée de LED et de petites résistances.

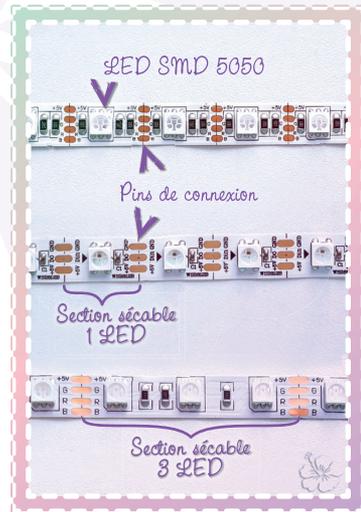
Celles que j'utilise éclairent en RGB (Red Green Blue - Rouge Vert Bleu en français) et sont équipées de LED SMD 5050 ("Surface-Mount Device", c'est-à-dire "Composant Monté en Surface"). Elles sont petites et extra plates, ce qui permet de les dissimuler plus facilement.

Les chiffres à côté de la mention "SMD" correspondent à leur taille en millimètres, séparés par un virgule : soit 5,0 mm x 5,0mm (et pas 50mm sinon ça serait une énorme LED).

Elles sont également sécables au niveau de leurs « pins » en cuivre (pastilles cuivrées).

Ces derniers servent d'interface de connexion entre les différentes parties de votre montage.

En fonction des bandes, il peut y avoir plusieurs LED sur la même section. Je préfère les bandes à une LED par section car on peut plus facilement gérer leur taille, même si elles sont un peu plus chères.



Cependant, peu importe votre nombre de LED par section, si vous voulez une bande qui peut changer de couleur, il faudra obligatoirement plus de 2 pins pour envoyer des données relatives aux différentes couleurs. En effet, avec 2 pins vous ne gérerez que l'allumage de vos LED étant donné qu'il faut au minimum un pin positif et un négatif pour que le courant circule. Ainsi, une bande à 2 pins n'éclairera que de la couleur de la LED.

## 2. Avec 3 ou 4 pins ?

Comme nous l'avons vu dans le chapitre précédent, les pins sont les petites pastilles cuivrées qui séparent les sections de LED. Alors combien en faut-il ?

La réponse est loin d'être simple, car si on veut rentrer dans les détails, il existe tellement de types de bandes ou types de connexions différentes qu'on s'y perd vite.

Comme ce tutoriel n'a pas pour but de tout vous expliquer, je vais juste vous parler des 2 types de bandes que j'utilise principalement.

Si j'ai besoin d'une bande LED programmable sur laquelle je peux agir sur chaque LED indépendamment les unes des autres, je vais prendre une bande adressable à 3 pins.

Si je veux juste une bande avec toutes les LED de la même couleur sans faire de la programmation, car il y a des télécommandes prévues à cet effet, je vais prendre une RGB classique à 4 pins.

**Donc pour résumer j'utilise :**

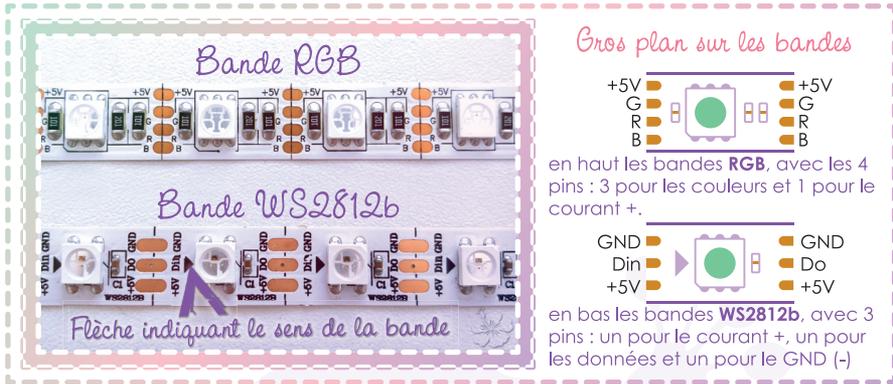
- 3 pins : programmation
- 4 pins : contrôle via une télécommande

Quand vous faites vos investigations sur internet et les différents sites marchands, il faudra toutefois utiliser d'autres termes pour la recherche que 3 pins ou 4 pins, pour être sûr de ne pas vous tromper car il existe une multitude de types de bandes. Et pour la suite de ce tutoriel, il faudra prendre les bonnes références sinon ça ne fonctionnera pas !

Ainsi on préférera le terme **WS2812B** pour les bandes 3 pins qui sont programmables et adressables LED par LED. Ces dernières utilisent la technologie PWM (Pulse Width Modulation, en français MLI, Modulation de Largeur d'Impulsion) : ce n'est pas forcément les meilleures au monde, mais pour le cosplay ça suffit largement surtout que le prix reste très abordable.

Pour les bandes à 4 pins, je tape : **USB RGB LED, DC 5V 5050 SMD**

Avec ça, vous êtes normalement sûr de tomber sur les bonnes bandes !



## 3. IP quésaco ?!

Vous pourrez aussi vous sentir perdus quand on va vous demander de choisir l'IP de la bande ! "IP" signifie Indice de Protection (**Ingress Protection** en anglais)

L'indice de protection est un standard au niveau international, relatif à l'étanchéité.

Il a pour objectif de classer le matériel électrique en fonction du niveau de protection qu'il offre contre l'intrusion de corps étrangers, qu'ils soient solides ou liquides.

Les indicateurs se présentent sous la forme des 2 lettres IP suivies de 2 chiffres. Le premier indique le niveau de protection contre les corps solides supérieurs à 12mm et le second indique le niveau de protection contre les projections d'eau.

Les indices les plus rencontrés sont les IP20, IP44 et IP65. L'IP20 est destiné à un usage exclusivement intérieur, alors que les IP44 et IP65 peuvent être utilisés partout.

Pour le cosplay, l'IP20 ou 30 sont suffisants : sur les indices plus élevés, du silicone entoure les LED et les pins de connexion. Il faut alors le retirer pour faire les soudures.

Cependant, si vous gardez la bande presque intacte et que vous avez juste à souder l'extrémité, les normes 44 (entourée de colle silicone) ou 65 (entouré d'un tube en silicone) restent une bonne alternative: les LED sont protégées, et ça évite de toucher aux petits composants qui sont sur la bande.

Au-delà, ça ne sert à rien, car même pour une séance photo sous l'eau, il faudra étanchéifier tout le reste avec des boîtiers étanches et ça devient vite compliqué.

Sachant que plus c'est étanche, plus c'est cher : pour le cosplay, n'allez pas au-delà de IP65 ! Par exemple, les bandes IP67 sont des IP20 dans un tube en PVC : donc si vous voulez la couper, vous allez vous retrouver avec une bande qui ne sera plus du tout étanche et que vous aurez payé beaucoup plus cher qu'une IP20 !

TABLE DES INDICES DE PROTECTION	
1 <sup>ER</sup> CHIFFRE protection contre les solides	2 <sup>ÈME</sup> CHIFFRE protection contre les liquides
 Aucune protection <b>0</b>	 Aucune protection <b>0</b>
 Protégé contre les corps solides supérieurs à 50 mm <b>1</b>	 Protégé contre les chutes verticales de gouttes d'eau <b>1</b>
 Protégé contre les corps solides supérieurs à 12,5 mm <b>2</b>	 Protégé contre les chutes de gouttes d'eau jusqu'à 15° de la verticale <b>2</b>
 Protégé contre les corps solides supérieurs à 2,5 mm <b>3</b>	 Protégé contre l'eau en pluie jusqu'à 60° de la verticale <b>3</b>
 Protégé contre les corps solides supérieurs à 1 mm <b>4</b>	 Protégé contre les projections d'eau de toutes directions <b>4</b>
 Protégé contre les poussières et autres résidus microscopiques <b>5</b>	 Protégé contre les jets d'eau de toutes directions à la lance <b>5</b>
 Totalement protégé contre les poussières <b>6</b>	 Protégé contre les forts jets d'eau de toutes directions à la lance <b>6</b>
<b>Exemple :</b>  +  IP <b>65</b>	 Protégé contre les effets de l'immersion temporaire jusqu'à 1m, pendant 30min <b>7</b>
 Matériel submersible dans en durée et en pression au-delà de 1 m <b>8</b>	
 Protégé contre les jets d'eau de toutes directions  Totalement protégé contre les poussières	

## 4. 30/m , 60/m, etc. : que choisir !

Les indications chiffre/m donnent simplement le nombre de LED qu'il y a pour 1 mètre de bande.

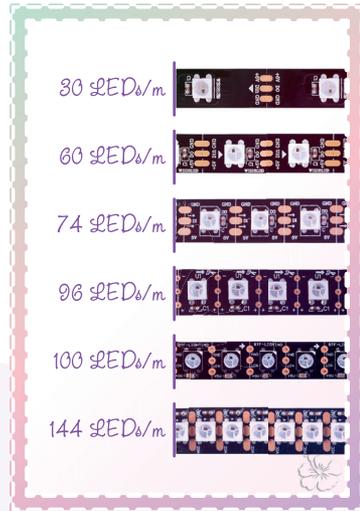
Par exemple : 30/m signifie qu'il y aura 30 LED pour 1 mètre de bande .

Personnellement, je choisis toujours au minimum 60 LED par mètre, je trouve que c'est un bon ratio. En dessous ça fait vraiment très clairsemé, au-delà c'est vraiment si vous avez envie d'avoir beaucoup de luminosité ou bien pour limiter le visuel des « points »

lumineux formés par les LED. Cependant pour éviter cela, il faudra rajouter un élément diffusant autour de votre bande, comme du polystyrène par exemple, car même avec le nombre maximum de LED par mètre présent sur le marché, on aura toujours cet effet de « points » lumineux.

Elles peuvent aller jusqu'à 144 LED par mètre, mais sachez que plus il y aura de LED par mètre plus votre installation sera gourmande en énergie !

À vous de bien déterminer vos besoins en fonction de votre projet.



## Conclusion

Bien choisir sa bande LED est la première étape pour mener à bien votre projet.

Pour résumer : si vous ne touchez pas trop en programmation, et que vous voulez la même lumière dans votre costume ou vos accessoires, prenez les bandes RGB à 4 pins. Avec des télécommandes ou des applications, vous les allumerez en toute simplicité.

En revanche, si vous voulez aller plus loin et mettre des effets d'allumage progressifs ou de choix de couleur LED par LED, il faudra prendre les bandes 3 pins WS2812b.

Sachez quand même qu'il existe un moyen de programmer les bandes 4 pins RGB, sinon on ne pourrait pas les faire fonctionner avec une télécommande. Cette dernière envoie des informations à un composant programmé en usine qui va agir sur la bande. Mais c'est plus complexe qu'avec les bandes 3 pins WS2812b, car vous aurez besoin de plus de composants pour que cela fonctionne correctement. Et avec plus de composants, le montage sera plus complexe et il faudra prévoir plus de place !

Le but de ce livre étant de rester accessible à tous, je ne vais pas y aborder la programmation des bandes 4 pins RGB.

Mais si vous êtes curieux, il existe plein de tutoriels sur la toile.



## II. Les bandes RGB simples et efficaces

Si vous avez peur de vous lancer dans de la programmation et que vous avez juste besoin de mettre de la lumière dans votre costume, ces bandes sont suffisantes. En 2 branchements, vous aurez votre costume éclairé sans même avoir besoin de brancher le fer à souder.

### 1. Les alimenter

La partie peut sembler la plus problématique mais s'avère être assez simple.

Vous avez sans doute déjà acheté une batterie externe pour recharger votre téléphone, car avec Pokémon GO vous ne pouvez pas jouer plus de 10 min sans tomber en rade !

Et bien, on va utiliser la même chose pour alimenter nos LED.

Ce qui est bien avec ces batteries 5V c'est qu'il en existe de toutes tailles et de toutes capacités à prix plus ou moins élevé ce qui vous laisse le choix !

Bien sûr plus votre costume aura de LED, plus il faudra une batterie avec une grande capacité (le nombre de milliAmpères-heure - mAh), sinon vous allez pouvoir l'allumer pendant 5 minutes puis il faudra recharger.



Qui dit plus grande capacité, dit souvent batterie plus chère. Mais avec 5000mAh vous avez de quoi tenir pas mal de temps, même avec 300 LED sur votre bande.

Il suffira simplement de connecter cette batterie à la bande LED via un contrôleur USB qui changera la couleur et les effets lumineux.

## 2. Les contrôler

Comme ces bandes sont constituées de LED RGB, il faut leur envoyer des informations sur la couleur que vous voulez. Pour cela, il existe des contrôleurs que l'on branche directement à la bande, et qui vous éviteront un montage compliqué avec une carte de programmation. Cependant, vous n'aurez pas des possibilités illimitées, de par le fait que ces bandes ne sont pas adressables LED par LED : changer la couleur aura pour effet de la changer pour toutes les LED en même temps.

Il existe plusieurs types de contrôleurs ayant leurs avantages et leurs inconvénients.

### On trouve principalement :

- des contrôleurs manuels
- des sans fil

L'avantage de ceux sans fil, c'est la télécommande. Grâce à elle, vous aurez accès directement à la couleur que vous voulez, vu qu'il suffira d'appuyer

sur le bouton correspondant. Vous aurez aussi (en fonction de la télécommande) accès à des effets de transition de couleur, de clignotement et de variation d'intensité des LED.



Ces manipulations sont moins pratiques avec le mini contrôleur manuel connecté directement à la bande LED car il ne possède que 3 boutons. Même s'il est possible d'accéder à toutes les couleurs et un large panel d'effets, comme il n'y a que 3 boutons, vous devrez les faire défiler une à une pour chercher la couleur ou l'effet que vous désirez. Mais son avantage par rapport à celui sans fil, est qu'il est petit et relié directement à la bande : aucun souci de signal parasite qui pourrait faire changer la couleur de votre bande LED à votre insu. En effet les contrôleurs sans fil utilisent une

connexion IR (infra-rouge) par télécommande, et si vous n'êtes pas bien alignés, ou trop loin du capteur (pour un décor par exemple) cela risque de mal fonctionner, voir ne pas fonctionner du tout. Et n'importe qui avec une télécommande du même type pourrait s'amuser à changer la couleur de votre bande. Après il ne faut pas non plus

être parano, ça reste quand même une bonne alternative si la programmation vous fait vraiment peur, malgré le tutoriel détaillé de la 3ème partie. Si vous optez pour la télécommande, il faudra prévoir de la mettre quelque part dans votre costume pour qu'elle soit accessible. Et de facto il faudra que le récepteur soit placé de façon que vous puissiez le pointer vous-même. Si vous le mettez dans le dos ça sera compliqué...

Attention avec les bandes RGB : certaines sont connectées en GRB (vert, rouge, bleu) : en soi, rien de bien grave, mais si vous connectez la bande à un contrôleur branché en RGB et relié à une télécommande, les couleurs seront inversées ! Pour rectifier le tir, il faudra juste inverser le fil rouge et le fil vert qui partent de votre connecteur.

Costume

## Conclusion

Les bandes RGB préprogrammées via contrôleur sont une solution simple pour ceux qui veulent rajouter de la lumière sans prise de tête. Avec elles vous n'aurez presque pas besoin de faire de la soudure car elles sont la plupart du temps vendues avec le système de connexion au module de contrôle. Après si vous voulez couper la bande pour répartir vos LED sur le costume, il faudra quand même passer par la case soudure. Il existe des fils vendus par 4 que vous pouvez utiliser pour relier vos bouts de bandes LED. L'avantage de ces fils est qu'ils sont déjà colorés pour ne pas vous tromper dans vos connexions.



De manière générale, il faut bien étudier comment vous allez disposer vos bouts de bandes afin de prévoir les longueurs de fils adaptées et surtout comment vous allez les cacher, car il n'y a rien de plus laid que des fils apparents.

Mais avec un peu de jugeote, on peut faire des choses sympas rien qu'avec ces bandes !

Cependant gardez bien à l'esprit que vous ne pourrez pas changer la couleur de chaque LED indépendamment avec ces bandes ! Cela n'est possible qu'avec une bande adressable que nous allons détailler maintenant. Celles-ci vont vous offrir beaucoup plus de possibilités, mais cela va demander un peu plus de technique.

## III. Les bandes WS2812b à programmer

Je pense que la partie la plus compliquée sera celle sur la programmation, car si vous n'en avez jamais fait ça risque d'être un peu du chinois, mais je vais essayer de vous donner les bases pour faire des effets sympas facilement et rapidement.

Et qui dit programmation dit carte à programmer, nous allons donc commencer par le choix de cette carte.

### 1. Les cartes de programmation

C'est l'élément indispensable pour faire fonctionner votre bande, sans elle rien n'est possible. Il existe différents types de cartes pour « contrôler » vos LED, de différentes marques, différents prix, différentes tailles, etc. on peut alors facilement se sentir perdu et penser que ça va être trop compliqué de choisir le bon modèle pour nos besoins.

L'électronique a fait un bon considérable depuis quelques temps et maintenant on peut trouver des cartes de programmation à des prix très abordables.

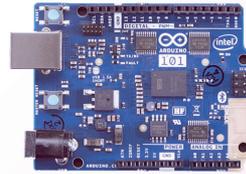
#### Les cartes Arduino



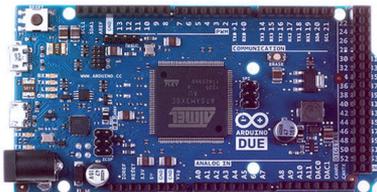
Leonardo



UNO



101



DUE



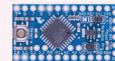
MEGA



Nano



Micro



Pro mini



Il existe plein de différentes cartes le plus souvent basées sur la technologie Arduino : les micro / les pro micro / les pro mini / les Uno etc.

Si l'envie vous prend d'en savoir plus, vous trouverez plein d'informations sur Internet !

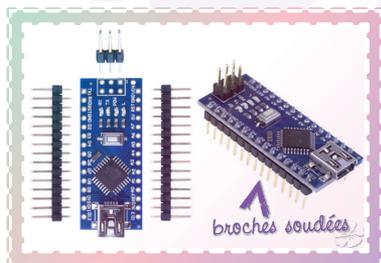
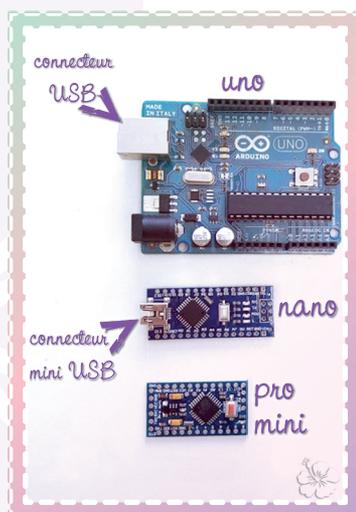
Mais je vous déconseille la Uno car elle est vraiment grosse et pour simplement contrôler des LED vous n'avez pas besoin d'un truc énorme, une simple carte Arduino (ou compatible Arduino) NANO fera l'affaire !

Les cartes officielles Arduino NANO avoisinent les 20€, mais vous trouverez des produits concurrents pour moins de 5€ ! Le tout est de bien vérifier qu'elles sont compatibles Arduino ou que le microcontrôleur est un **ATmega328P**.

Les "micro" et "pro micro"/ "mini" sont sensiblement comme les "nano" : c'est juste le microcontrôleur qui change.

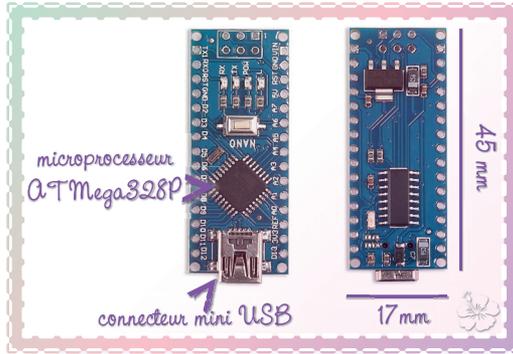
Bien que les "pro micro"/ "mini" présentent l'avantage d'être plus petites que les nano, elles n'ont pas de port mini USB, indispensable à la connexion avec un ordinateur, afin de transférer votre programme. Ces cartes nécessitent, pour ce faire, des programmeurs FTDI et là, ça devient plus compliqué. C'est pour cette raison que je ne les utilise pas d'autant plus que le gain de place n'est pas significatif par rapport à la taille de la batterie, rarement compacte, que vous intégrerez pour alimenter votre installation. Il est donc indispensable de prévoir un espace suffisant pour intégrer votre carte ainsi que votre batterie.

Pour votre premier achat, pensez à en prendre une vendue avec un câble USB to mini USB, par la suite vous pourrez acheter les cartes seules.



Dernier point, les "broches" : si vous n'avez pas besoin de souder votre carte sur un circuit imprimé, prenez-la avec des broches non soudées. Ainsi, elle sera toute fine et vous n'aurez pas besoin de mettre une protection en polystyrène, par exemple, pour protéger les broches et surtout votre costume !

Dans ce livre nous utiliserons la carte de type **Arduino NANO**. Sachez cependant que les branchements sont les mêmes pour tous les autres modèles.



## 2. L'alimentation

Pour alimenter le montage on utilise une batterie externe, comme pour les bandes RGB. Vous pouvez aussi utiliser des piles dans des boîtiers adaptés, mais l'avantage d'une batterie USB c'est qu'elle est rechargeable et qu'elle délivre précisément du 5V.

Cependant contrairement aux bandes RGB et à leur contrôleur, il est fortement déconseillé de connecter la bande LED à votre carte, et d'utiliser le port micro USB pour alimenter votre montage. La carte n'étant pas assez puissante pour délivrer plus de 200mA, si vous avez beaucoup de LED sur votre bande, vous allez griller la carte. Si vous n'avez que 10 LED à



contrôler ça passera, mais c'est préférable d'éviter de vous habituer à cette méthode et de directement prendre les bons réflexes pour alimenter vos bandes LED.

**Attention :** si vous avez vraiment beaucoup de LED, il faudra mettre plusieurs batteries, sinon vous n'aurez pas assez de puissance pour tout faire fonctionner : par exemple, ma robe de Katniss a 750 LED, pour toutes les faire fonctionner en même temps, j'ai dû utiliser 2 batteries qui alimentent chacune une partie du montage.

### 3. La connexion

Les réjouissances vont enfin commencer et il va falloir sortir le fer à souder !

C'est parti pour le tutoriel qui va vous servir de base pour tous vos prochains montages.

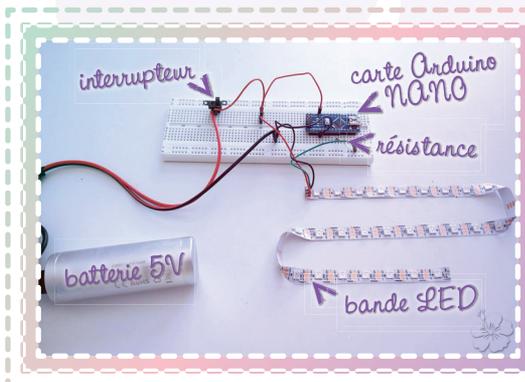
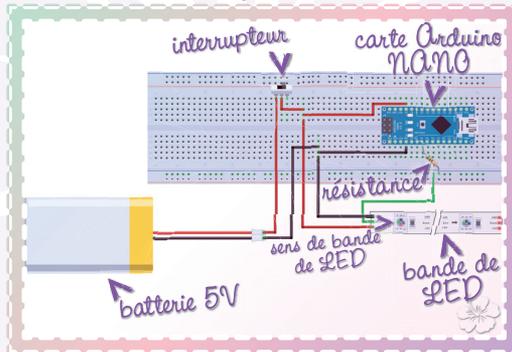
Dans un premier temps, on réalise le montage de base sur une plaque de test pour bien comprendre les connexions entre les différents éléments nécessaires au bon fonctionnement de notre bande LED.

Par convention, on utilisera des fils **rouges** pour le positif (5V), des fils **noirs** pour le négatif (GND) et des fils **verts** pour la Data (Din)

*Astuce*

Pour les explications je vais utiliser une plaque de test sans soudures pour électronique. Je vous invite à investir dans ce matériel : vous pouvez trouver des kits pour moins de 20€ avec la plaque, des câbles de connexion et divers composants électroniques utiles pour vos futurs projets, ça permet de tester votre montage avant de faire les soudures.

Voici le schéma de montage (réalisé avec le logiciel fritzing)



Résultat sur la plaque de test

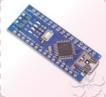
La batterie est connectée à la carte et à la bande pour les alimenter en courant.

On met un interrupteur entre la batterie et les composants pour faciliter l'allumage (sinon il faut brancher et débrancher le câble).

La carte est connectée à la bande pour lui envoyer les informations qui seront programmées à l'intérieur (couleur, effet, etc.)

Il est temps de sortir le fer à souder et l'étain pour réaliser le montage définitif, car vous n'allez pas intégrer la plaque de test dans votre costume !

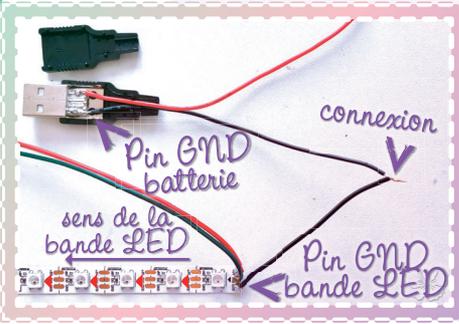
## Materiel

> Carte type Arduino NANO		> Bande LED WS2812b	
> Résistance 510 Ω (ohm)		> Interrupteur	
> Batterie USB 5V		> Prise USB (cf partie 4.2)	
> Gaine thermorétractable		> Fils électriques Ø ≈ 0,08 mm (rouge, vert et noir)	

Avant de débuter, si nécessaire, soudez 3 fils de couleur aux pins de la bande (**rouge** = +5V, **vert** = Din, **noir** = GND). **Attention au sens de la bande symbolisé par une flèche.**

On commencera par connecter le négatif de la batterie à la bande LED et à la carte Arduino. Puis on fera de même avec le positif en insérant un interrupteur entre la batterie et les 2 composants. Enfin on connectera la bande LED et la carte Arduino pour l'envoi des données programmées.

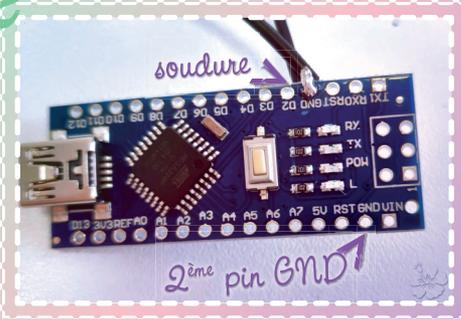
### 1 Connexion des fils GND



Enroulez le fil du pin GND de la bande LED avec le fil négatif du connecteur USB (fils noirs).

Si par la suite vous devez connecter plusieurs bandes ensemble, respectez bien le sens des flèches pour ne pas les connecter à l'envers !

2

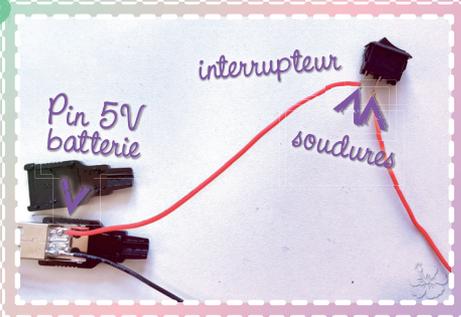


### Soudure du GND

Passer les fils noirs dans le trou du pin GND de la carte Arduino et soudez-les avec de l'étain.

Notez qu'il y a 2 pins GND sur la carte donc peu importe celui que vous choisirez, le résultat sera le même.

3

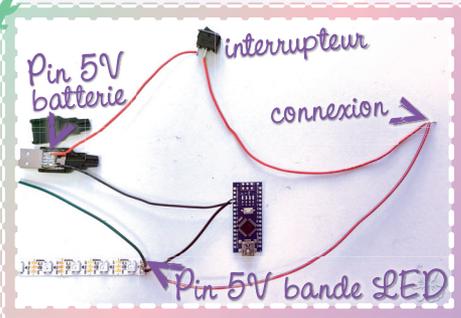


### Placement de l'interrupteur

Coupez le fil du pin 5V qui part de votre connecteur usb en 2 et insérez y l'interrupteur.

Cette partie est facultative mais vivement recommandée pour ne pas avoir à brancher et débrancher votre batterie à chaque fois que vous voulez allumer ou éteindre votre bande : un interrupteur c'est plus pratique.

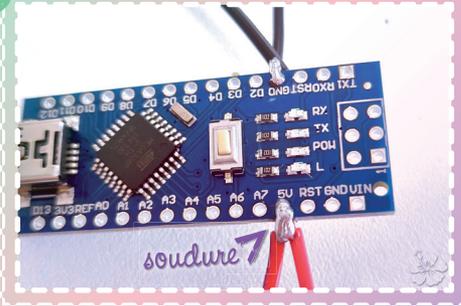
4



### Connexion des fils 5V

Enroulez le fil du pin 5V de la bande LED au fil qui vient de l'interrupteur pour les connecter entre eux. (fils rouges)

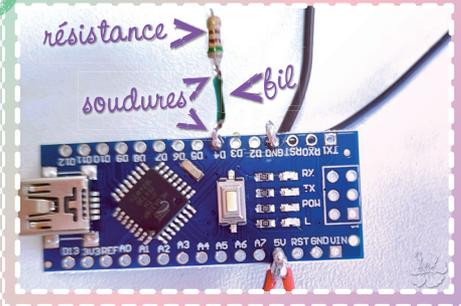
5



## Soudure du 5V

Passez les fils rouges dans le trou du pin 5V de la carte Arduino et soudez-les avec de l'étain.

6

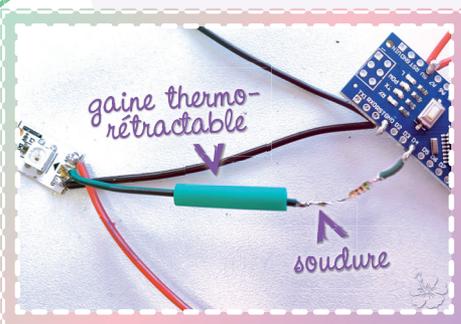


## Connexion de la résistance

Soudez la résistance de 510 ohms à un fil (vert) que vous soudez au pin D4 de la carte Arduino.

Il est préférable de ne pas directement souder la résistance au pin pour éviter que la branche ne se casse à la suite d'une torsion dans le costume. Après il m'arrive de le faire car c'est plus rapide mais il y a des risques.

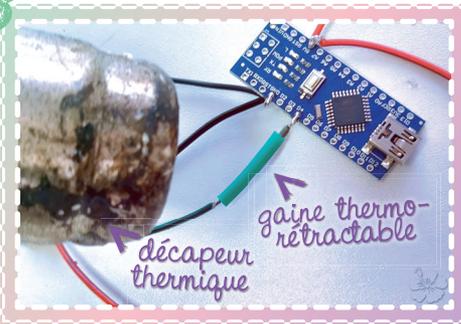
7



## Connexion à la bande LED

Soudez le fil Din (fil vert) de la bande LED à la résistance en insérant au préalable une gaine thermorétractable sur le fil.

8



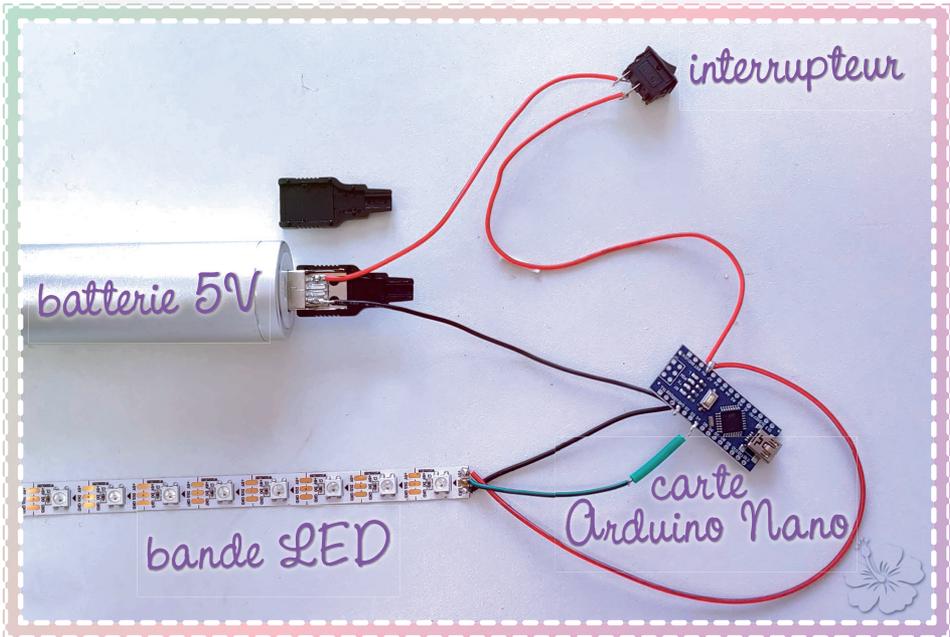
### Gaine Thermorétractable

Déplacez la gaine sur la résistance et chauffez-la avec un dépoteur thermique pour la rétracter et protéger votre résistance.

Voilà : le montage est fini ! Cependant si vous le branchez maintenant sans avoir téléchargé un programme dans la carte rien ne se passera !

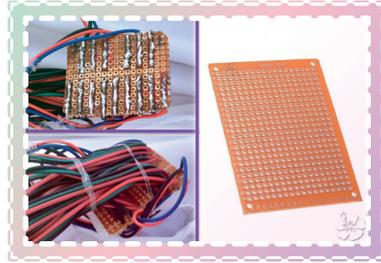
Mais au moins le montage et les soudures sont faits.

Ci-dessous le montage final pour bien voir les connexions :



Ici je vous ai montré comment faire les branchements ! Mais, ne vous dites pas que vous allez mettre vos LED à la fin : dans un cosplay, il faut réfléchir en amont afin de prévoir correctement les longueurs de fils, la place pour les composants et la batterie.

Il m'arrive aussi de me servir de plaques de circuits imprimés pour disposer mes petits composants comme les résistances et faire des ponts de connexions entre les câbles, surtout pour des montages en parallèle. Ce n'est pas obligatoire mais ça permet d'avoir des branchements plus organisés.



## 4. Les Bases de la Programmation

J'ai conscience que la programmation risque de poser des problèmes aux personnes qui n'en n'ont jamais fait ! Dans ce chapitre, nous allons aborder les notions de base, qui vont vous permettre de comprendre la partie sur la programmation à proprement dite des LED. Si vous vous y connaissez déjà en programmation, vous pouvez passer cette section et vous rendre directement à la partie 5.

Le langage Arduino est basé sur le C++\*, en programmation on utilise un langage spécifique pour que la machine comprenne les instructions qu'on lui envoie. Comme lorsque deux personnes veulent communiquer entre elles, si elles n'utilisent pas la même langue le message ne passera pas !

Afin que la machine exécute correctement le programme, il faut lui envoyer des instructions dans sa langue. Très tatillonne sur la syntaxe, il ne faut pas faire de fautes, sinon elle n'exécutera rien. Ainsi chaque instruction doit être précise et bien distincte.

Le point-virgule ( ; ) fait office de « séparateur » entre les différentes instructions.

On met généralement une instruction par ligne, ce n'est pas obligatoire mais c'est une convention d'écriture pour lire le code plus facilement.

Dans ce langage on rencontre principalement des variables, des constantes, des conditions, des boucles et des fonctions.

### a. Les variables

Une variable, c'est une petite information stockée en mémoire temporairement, qui comme son nom l'indique peut varier lors de l'exécution du code.

Elle est définie selon **3 caractéristiques** :

- **son nom** : qui permet de la repérer et de l'utiliser.
- **son type** : qui définit quelles données la variable peut contenir.
- **sa valeur** : qui correspond à l'information stockée dans la variable.

Lors de la définition d'une variable on renseigne le type de la donnée, puis son nom, et enfin on lui affecte sa valeur en séparant le nom et la valeur avec un " = ".

Exemple : `type_donnee ma_variable = sa_valeur;`

(le point virgule signifie la fin de l'instruction)

Il existe différents types de données dont voici les principaux à connaître :

- Les chaînes de caractères : `char` ; c'est du texte plus ou moins long.

Exemple : `char ma_variable = "Je suis un texte";`

Il est important de bien mettre son texte entre guillemets pour qu'il ne soit pas interprété comme une instruction de code.

- Les nombres entiers : `int` ; ce sont les nombres du type 1, 2, 3, 4, etc. On compte aussi parmi eux les entiers relatifs : -1, -2, -3...

Exemple : `int ma_variable = 12;`

**Attention :** si vous mettez une virgule dans votre valeur le programme va automatiquement la tronquer à l'unité et cela peu importe les chiffres après la virgule même si c'est 12.99999999999.

- Les nombres décimaux : `float` ; ce sont les nombres à virgule, comme 20,50. On peut stocker de nombreux chiffres après la virgule. Ces nombres doivent être écrits avec un point au lieu de la virgule (c'est la notation anglaise).

Exemple : `float ma_variable = 20.50;`

- Les booléens : `bool` ; c'est un type de variable à 2 états: vrai ou faux. Ils sont très utilisés et on écrit `true` pour vrai et `false` pour faux.

Exemple : `bool ma_variable = true;`

Si vos variables sont de type `int` ou `float`, vous pouvez vous en servir pour faire des calculs simples via les 4 opérateurs mathématiques de base : Addition (+) , Soustraction (-) , Multiplication (\*) et Division (/).

Ainsi dans notre programme, on pourra par exemple ajouter ou soustraire des valeurs aux nuances d'une couleur pour la faire varier petit à petit.

## b. Les constantes

Une constante, à la différence d'une variable, ne pourra jamais changer de valeur lors de l'exécution du code.

On utilise une constante plutôt qu'une valeur, pour savoir plus facilement de quoi on parle quand on lit le code. Et aussi, afin de pouvoir la modifier facilement en début de programme si sa valeur peut être amenée à changer, comme le nombre de LED sur votre bande par exemple.

Par convention on écrit les constantes en majuscule, ce n'est pas une obligation, mais ça permet de savoir au premier coup d'oeil qu'il ne s'agit pas d'une variable.

Pour la définir on ajoute le terme `const` devant son type ou on utilise `#define`. Je préfère `#define` car ça fonctionne aussi avec les constantes de type `char`.

Exemple : `const int NOMBRE_LED = 30;`

ou `#define NOMBRE_LED 30`

Notez qu'avec `#define` il ne faut pas mettre de `;` à la fin de la ligne. Le seul moment où l'on ne finit par une ligne par un `;` c'est quand elle commence par un `#`.

## c. Les conditions

Très utilisées en programmation, elles testent les instructions. On peut par exemple, demander au programme : **si** `ma_variable` est égale à 10, **alors** fais une action. Pour faire les tests, on utilise principalement les symboles suivants :

Symbole	Signification
<code>==</code>	égal à
<code>&gt;</code>	supérieur à
<code>&lt;</code>	inférieur à
<code>&gt;=</code>	supérieur ou égal
<code>&lt;=</code>	inférieur ou égal
<code>!</code>	différent de

Code  
Pour écrire un commentaire qui ne sera pas exécuté on met 2 slashes en début de ligne (`//`) ou on met tout le commentaire entre `/* */`

Pour écrire une condition, on utilise le "`if`" (**si** en anglais), puis on indique la condition à tester entre parenthèses, et enfin on met dans les accolades ce qui doit être exécuté si la condition est vraie. Ce qui, si on reprend l'exemple plus haut, nous donne :

```
if (ma_variable == 10) {  
    //code à exécuter si ma_variable vaut 10  
}
```

Au "`if`" on peut ajouter un "`else`" pour dire **sinon**. Le code mis dans les accolades à la suite du `else` sera exécuté si la condition définie dans le `if` est fausse.

```
if (ma_variable == 10) {  
    //code à exécuter si ma_variable vaut 10  
}else{  
    //code à exécuter si ma_variable n'est pas égale à 10  
}
```

Notez qu'on ne met pas de condition entre parenthèses pour le "else", car on n'exécute le code que si la condition du "if" n'est pas vraie.

Cependant si on veut rajouter une condition en plus entre le if et le else, on utilisera le "elseif", et de nouveau il faudra mettre une condition entre parenthèses :

```

if (ma_variable == 10) {
    //code à exécuter si ma_variable vaut 10
}else if (ma_variable >= 10) {
    //code à exécuter si ma_variable vaut 10 ou plus
}else{
    //code à exécuter si aucune des conditions n'est vraie
}

```

J'ai volontairement mis en 2<sup>ème</sup> condition "supérieur ou égal", car logiquement, les conditions 1 et 2 peuvent être toutes les 2 vraies si ma\_variable vaut exactement 10.

Cependant, si la condition 1 est remplie, le programme n'ira pas appliquer le code de la condition "supérieur ou égale à 10". Les conditions sont testées dans l'ordre de syntaxe (de haut en bas), si l'une d'entre elle est remplie, le programme ne testera pas les autres et passera à la suite du code. Donc ici, écrire "supérieur ou égal" dans la deuxième condition ne sert à rien, on peut simplement mettre supérieur (>).

En revanche, il est tout à fait possible d'imbriquer plusieurs "if" les uns dans les autres.

```

if (ma_variable > 10) {
    //code à exécuter si ma_variable vaut 10 ou plus
    if (ma_variable > 20) {
        //code à n'exécuter que si ma_variable vaut 20 ou plus
    }
}else if (ma_variable == 10) {
    //code à n'exécuter que si ma_variable vaut 10
}else{
    //code à exécuter si aucune des conditions n'est vraie
}

```

Cela offre des possibilités infinies de test. Il faut juste bien ordonner vos conditions pour que le programme les teste dans le bon ordre afin de répondre à vos attentes.

On peut également faire plusieurs tests en même temps, dans les parenthèses, pour éviter de devoir imbriquer trop de "if" les uns dans les autres. Cette méthode permet une lecture plus rapide et plus claire des conditions testées pour chaque demande.

Pour ce faire, on sépare chaque instruction par des symboles signifiant "ET" ou "OU" respectivement représentés par les symboles "&&" et "||" (PC : AltGr+6 / Mac : Shift+Alt+L).

On peut également demander si la condition est "fausse" avec un point d'exclamation (!) placé avant la condition entre parenthèses.

```
if (ma_variable > 10 && ma_variable < 20) {
    /*code à exécuter si ma_variable est supérieure à 10 ET
    inférieure à 20*/
}else if (ma_variable == 30 || ma_variable == 40) {
    /*code à exécuter que si ma_variable vaut exactement 30 OU
    exactement 40*/
}else if (!(ma_variable < 5)) {
    /*code à exécuter que si ma_variable est supérieure ou égale à 5*/
}else{
    /*code à exécuter si aucune des conditions n'est vraie*/
}
```

Écrire `!(ma_variable < 5)` équivaut à mettre `ma_variable >= 5`. Le ! est surtout utilisé avec des variables de type bool pour savoir si c'est vrai ou faux (`true/false`).

Pour "entrer" dans une condition il faut qu'elle soit vraie : donc si on veut rentrer dans une condition qui est "fausse" il faut la rendre "vraie" en la testant avec le point d'exclamation (qui va en quelque sorte "inverser" la valeur de la variable).

Exemple :

```
bool ma_variable = false;
if (!ma_variable){
    /*le code va s'exécuter car ma_variable vaut false et que
    !false donne true : cette condition est donc vraie*/
}
```

Il est aussi tout à fait possible d'imbriquer des parenthèses les unes dans les autres pour vos conditions, tout comme en mathématiques, pour signifier qu'un test doit être fait avant un autre.

Exemple :

```
if ((ma_variable > 10 && ma_variable < 20) || ma_variable ==
30 ) {
    /*code à exécuter si ma_variable est comprise entre 10 et 20
    OU si elle vaut 30*/
}else{
    //code à exécuter sinon
}
```

Pour aller plus loin, le `switch` permet aussi de tester la valeur d'une variable, c'est très utile si vous avez beaucoup de conditions : ça vous évite de faire trop de `elseif` et c'est plus lisible.

```
switch (ma_variable){
    case 1:
        /*code à exécuter si ma_variable vaut 1*/
        break;
    case 15:
        /*code à exécuter si ma_variable vaut 15*/
        break;
    default:
        /*code à exécuter si ma_variable n'est pas dans une ``case``
précédente*/
        break;
}
```

Le `break` est très important : ça ferme la "case" et sort de la boucle `switch` si on remplit la condition. Dans le `switch`, il ne faut pas oublier le `default`, même si vous ne mettez rien dedans, sinon le programme va vous signifier une erreur.

Avec tout ça vous devriez avoir les bases pour comprendre et utiliser les conditions dans vos programmes

## d. Les boucles

Une boucle est constituée d'une série d'instructions exécutées jusqu'à ce qu'un résultat particulier soit obtenu ou qu'une condition pré-déterminée soit remplie.

En C il existe le `while`, `do while` et `for`. Pour mes programmes j'utilise principalement la boucle `for` mais les 3 font à peu près la même chose.

Cependant, si on ne lui dit pas quand s'arrêter, le programme va tomber dans une boucle infinie, sans jamais passer à la suite et finira par faire planter la machine. Il est donc important d'associer à chaque boucle une condition, qui permettra au programme de savoir qu'elle est finie afin de passer à la suite. Il s'agit souvent d'un "compteur" que l'on va incrémenter (ou décrémenter) pour qu'à un moment il soit égal ou supérieur à la valeur qui permet de stopper la boucle.

Par exemple, pour appliquer un changement à toutes les valeurs comprises entre 0 et 10, on crée une boucle qui fera tourner notre code (entre les accolades) jusqu'à ce que le compteur, que l'on aura initié à 0 et qui sera incrémenté de 1 à chaque tour, arrive à 10. Dans la parenthèse on met notre condition (`compteur < 10`), qui indiquera au programme de s'arrêter quand le compteur aura atteint ou dépassé cette valeur.

Ce qui nous donne le code ci-dessous :

```
int compteur = 0;
while (compteur < 10){
    Serial.print("Le compteur vaut %d \n", compteur);
    compteur++;
}
```

Code

Pour modifier des valeurs (ajouter ou retirer) on utilise les symboles suivants :

Symbole	Signification
++	ajouter 1
--	retirer 1
+=2	ajouter 2
--2	retirer 2

**Serial.print**  
permet d'afficher du texte dans le moniteur série\* à l'exécution. Ici %d sera remplacé par la valeur de compteur à l'affichage. Le \n fait un retour à la ligne.

Info

Pour incrémenter ou décrétement de 3 en 3 on met +=3 ou -=3 etc.

Pour le `do while` et le `for`, la finalité est la même c'est juste la façon de rédiger qui va être différente, ainsi pour le `do while`, la condition sera mise à la fin :

```
int compteur = 0;
do {
    Serial.print("Le compteur vaut %d \n", compteur);
    compteur++;
}while (compteur < 10);
```

Code

Pour le `for`, l'initialisation, la condition et le compteur sont directement mis entre parenthèses, ce qui permet dès le début de savoir comment la boucle va fonctionner.

```
for (int compteur = 0 ; compteur < 10 ; compteur++){
    Serial.print("Le compteur vaut %d \n", compteur);
}
```

Code

Notez qu'on peut quitter une boucle à tout moment avec le mot clé "break" ou "return", qui peut être mis dans une condition `if` par exemple.

Le fait de choisir une écriture plus qu'une autre dépend surtout de ce qu'on veut faire. Dans le cas présent, retenir la boucle `for`, car c'est celle qui sera utilisée dans ce livre. Vous verrez que pour agir sur toutes vos LED en même temps, les boucles seront vos meilleures amies !

## e. Les fonctions

Les fonctions sont des petits bouts de programme que l'on définit une fois et que l'on peut appeler à tout moment. C'est très utile si vous avez besoin de répéter les mêmes actions à plusieurs endroits dans votre code.

Pour faire simple, si une variable vous permet de stocker une valeur, une fonction vous permettra de stocker une liste complète d'instructions réutilisables à l'infini.

On initialise une fonction comme une variable, par rapport à ce qu'elle doit retourner (`int`, `bool`, `char` ...). Et si elle ne retourne rien on utilisera de terme `void`. Comme pour une variable on lui donne un nom, puis on met entre parenthèses des paramètres séparés par des virgules. Ces paramètres seront remplacés par des valeurs lors de l'appel de la fonction. Pour finir on met des instructions entre les accolades comme pour les boucles. Outre le fait de ne pas avoir à recopier le code à chaque fois, elles permettent de changer les paramètres à chaque appel.

Exemple d'une fonction qui va retirer 5 au paramètre 1, multiplier par 3 le paramètre 2, et pour finir, additionner les 2 ensembles. Une fois déclarée il suffit de l'appeler avec son nom, tout en précisant les valeurs de ses paramètres. Elle exécutera alors tout le code entre les accolades en remplaçant les paramètres par leurs valeurs.

```
int calcul (int para1, int para2){
    para1 = para1 - 5;
    para2 = para2 * 3;

    return para1 + para2;
}

int nbr1 = calcul (10, 4); // nbr1 vaudra 17
int nbr2 = calcul (12, 2); // nbr2 vaudra 13
```

Le `return` définit ce que la fonction doit retourner comme résultat, mais vous verrez que pour les LED on ne sera pas obligé de mettre un `return`.

On pourra définir la fonction avec `void` et l'appeler pour agir sur les LED sans avoir besoin de retourner une valeur.

Normalement vous devriez en savoir assez pour commencer la section sur la programmation des LED.

Sachez que ces bases pourront vous servir par la suite pour tout programmer avec le logiciel Arduino, comme des servo-moteurs par exemple.

## 5. Programmer vos LED

### a. Avant de commencer

Dans un premier temps, rendez-vous sur le site officiel de Arduino pour télécharger la dernière version de leur éditeur : <https://www.arduino.cc/en/Main/Software>

Installez-le puis lancez-le.

Une fois lancé, le programme ouvrira directement un nouveau fichier avec les 2 fonctions de base : `setup` et `loop`, qui sont obligatoires pour tout programme Arduino.

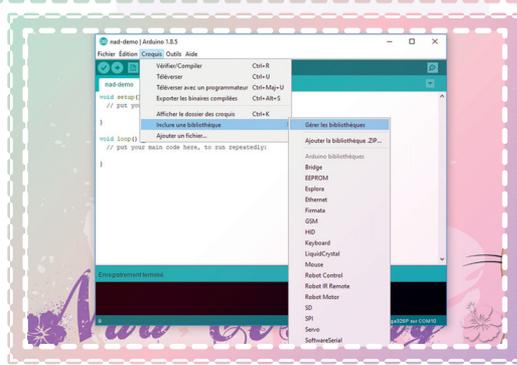
La fonction `setup` est toujours exécutée en premier, elle initialise votre programme, ici les paramètres de votre bande LED et permet d'inclure des effets sur vos LED qui n'auront lieu qu'une seule fois au démarrage.

La fonction `loop` s'exécute après la fonction `setup`, même si vous la déplacez avant dans le code, et va se répéter en boucle indéfiniment jusqu'à la coupure de l'alimentation, d'où son nom "`loop`" (**boucle** en anglais).

### b. La librairie

Afin de gérer les LED on utilise ce qu'on appelle des librairies (ou bibliothèques): ce sont des fichiers contenant du code avec des fonctions déjà pré-écrites qui évitent de devoir tout programmer : en gros ça facilite la tâche !

Ici on utilise FastLED mais il en existe d'autres : comme la NeoPixel de Adafruit. Mais si vous incluez la librairie Adafruit et que vous utilisez les fonctions de fastLED expliquées ci-dessous ça ne fonctionnera pas car chaque librairie a ses propres fonctions !



*Installer la librairie FastLED*

Avant de pouvoir l'inclure à votre programme il faut la télécharger car elle n'est pas de base dans le logiciel : allez dans Croquis > Inclure une Bibliothèque > Gérer les Bibliothèques.

Une fenêtre s'ouvre : tapez 'fastLED' dans le champ recherche puis installez le résultat via le bouton installation. Une fois installé, fermez la fenêtre.

On peut désormais inclure la librairie au début de notre fichier :

```
#include <FastLED.h>
```

## c. Mise en place des LED

Après avoir inclus la librairie, on va définir nos Constantes.

```
#define LED_PIN 4
#define NUM_LEDS 70
#define LED_TYPE WS2812
#define COLOR_ORDER GRB
```

`LED_PIN` correspond à la broche (pin) où est connectée la "data" (**fil vert**) de votre bande sur la carte : si vous n'indiquez pas le bon, ça ne fonctionnera pas !

`NUM_LEDS` indique le nombre de LED qui seront utilisées par le programme.

Il peut être inférieur au nombre de LED total si vous ne voulez pas toutes les utiliser.

Enfin le `LED_TYPE` et le `COLOR_ORDER` indique le type de la bande LED et l'ordre dans lequel sont indexées les couleurs. Ici on utilise des bandes de type **WS2812**, et sur ces dernières l'ordre des couleurs est **GRB**.

Ensuite, on définit le tableau **CRBG** qui va être utilisé pour stocker et manipuler les LED que l'on nommera "leds", puis on configure les LED dans la fonction `setup` :

```
CRGB leds[NUM_LEDS]; //le tableau contient 70 indexes

void setup() {
  FastLED.addLeds<LED_TYPE,LED_PIN,COLOR_ORDER>(leds, NUM_LEDS);
}
```

Un tableau est un ensemble de données ordonnées et accessibles par leur index : ici l'index est le numéro de la LED. Ce tableau va nous permettre de manipuler facilement les données propres à chaque LED (couleur, intensité etc.).

La ligne dans la fonction `setup`, indique à la bibliothèque FastLED qu'il y a une bande LED de type **WS2812** sur la broche 4 de la carte. Que cette dernière est encodée en **GRB** et que ces LED utilisent le tableau "leds" (défini plus haut), qui contient 70 entrées (une entrée par LED).

## d. Changer les paramètres d'une LED

Pour changer la couleur d'une LED, il faut modifier ses paramètres car par défaut elles sont toutes éteintes tant que vous ne leur attribuez rien.

Pour agir sur une LED, il faut simplement l'appeler via le tableau "leds" et son numéro d'entrée entre crochets afin de lui appliquer les changements désirés.

**Attention :** le tableau commence par l'entrée 0, ça veut dire que toutes les LED de la bande sont placées dans le tableau avec une clé égale à leur position -1.

Il existe plusieurs syntaxes pour changer la couleur des LED, voici les plus courantes :

```
leds[0] = CRGB::Red; //utilise le HTML Color Code
leds[1] = 0xFF0000; //utilise le code couleur hexadecimal

// utilise les valeurs R , G ,B comprise entre 0 et 255
leds[2] = CRGB(255, 0, 0);
leds[3].setRGB(255, 0, 0);
```

Peu importe la syntaxe choisie le résultat sera le même : les LED vont éclairer en rouge. Appliquons par exemple la couleur rouge sur la 1<sup>ère</sup> LED et du jaune sur la 4<sup>ème</sup>, et pour que cela soit effectif sur notre bande, on va l'écrire dans la fonction `loop`.

```
void loop() {
  leds[0] = CRGB::Red;    // première LED : en rouge
  leds[3] = CRGB::Yellow; // quatrième LED : en jaune
  FastLED.show();
}
```

On renseigne la couleur pour chaque LED, puis on applique les changements via la fonction `FastLED.show()` ; si vous n'écrivez pas cette ligne, rien ne se passera.

Cette fonction applique d'un coup tous les changements qui sont « au-dessus » : ici ça ne va pas allumer d'abord la led 1 puis la 4 : elles vont s'allumer en même temps.

Si vous voulez les allumer une à une il faudra utiliser la fonction `delay()` ; entre chaque attribution de couleur .

```
void loop() {
  leds[0] = CRGB::Red;
  FastLED.show();
  delay(1000);
}
```

```

    leds[3] = CRGB::Yellow;
    FastLED.show();
    delay(1000);
}

```

`delay()` ; fait une pause dans le programme : la valeur entre parenthèses correspond au temps en milliseconde de cette pause.

Si vous voulez faire clignoter une LED, il suffit simplement d'écraser ses paramètres de couleur en rappelant les fonctions `CRGB` et `FastLED.show()` ; après le `delay()` ;

```

void loop() {
    leds[0] = CRGB::Red;
    FastLED.show();
    delay(1000);

    leds[0] = CRGB::Yellow;
    FastLED.show();
    delay(1000);
} // la LED 0 alternera toutes les secondes entre le rouge et le jaune

```

## e. Agir sur toutes les LED en même temps

Afin de se faciliter la tâche, on peut utiliser des boucles pour appliquer à toutes les LED les mêmes paramètres. Cela s'avère très pratique si on en a beaucoup et que l'on veut, par exemple, leur attribuer à toutes la même couleur en même temps.

Pour ce faire, on crée une boucle `for`, dont le paramètre "i" va démarrer à 0 et augmenter de 1 en 1 (`i++`) jusqu'à ce qu'il ne soit plus inférieur à la variable `NUM_LEDS` (ici 30), qui correspond au nombre de LED total sur la bande.

Cette boucle va "tourner" sur elle-même tant que "i" reste inférieur à `NUM_LEDS`. A chaque tour elle ajoutera 1 à la valeur de "i" : 1<sup>er</sup> tour `i=0`, 2<sup>ème</sup> `i=1` etc. Quand "i" sera supérieur à `NUM_LEDS`, elle s'arrêtera car la condition ne sera plus remplie.

Ce qui nous donne le code suivant :

```

void loop() {
    for(int i = 0; i < NUM_LEDS; i++) { //début de la boucle
        leds[i] = CRGB::Red;
    } // fin de la boucle
    FastLED.show();
}

```

Ce code allume en rouge toutes les LED de la bande en même temps.

Si vous voulez faire clignoter toute la bande il suffit de faire 2 boucles séparées par un `delay()` ; en changeant la couleur dans chaque boucle.

```
void loop() {  
  for(int i = 0; i < NUM_LEDS; i++) { //début de la boucle  
    leds[i] = CRGB::Red;  
  } // fin de la boucle  
  FastLED.show();  
  delay(1000);  
  
  for(int i = 0; i < NUM_LEDS; i++) { //début de la boucle  
    leds[i] = CRGB::Yellow;  
  } // fin de la boucle  
  FastLED.show();  
  delay(1000);  
}
```

On peut aussi mettre le `FastLED.show()` ; dans la boucle, ce qui aura pour effet d'appliquer les changements un à un.

```
void loop() {  
  for(int i = 0; i < NUM_LEDS; i++) {  
    leds[i] = CRGB::Red;  
    FastLED.show();  
    leds[i] = CRGB::Black;  
    delay(50);  
  }  
}
```

Ici, à chaque "bouclage", on définit en rouge la couleur de la LED et on l'applique avec `FastLED.show()` ;. Ensuite on la passe en noir ("Black"), puis on fait un `delay()` ;. N'ayant pas mis un `FastLED.show()` ; après ce changement le programme va le garder en mémoire et l'appliquer à son prochain appel. Dans le cas présent lors du "bouclage" suivant, cela donnera un effet de LED qui se déplace le long de la bande. On allume la LED 0, puis on l'éteint quand on allume la LED 1 etc.

**Bon à savoir :** Black ne signifie pas que la LED va "éclairer" en noir mais qu'elle sera éteinte car le code RGB du noir est (0,0,0), ça veut dire que l'intensité de chaque diode sera à 0. Dans une LED RGB il y a 3 petites diodes, une **rouge**, une **verte** et une **bleue**. Pour obtenir d'autres couleurs, on joue sur l'intensité de chaque diode : plus on mettra un nombre proche de 255 pour une couleur, plus la diode brillera et *a contrario*

plus on mettra un nombre proche de 0 moins elle brillera : après c'est un peu comme en peinture, on mélange les couleurs primaires pour obtenir d'autres couleurs.

On peut également jouer sur l'intensité d'une couleur pour la faire varier et donner l'impression que la LED s'éteint petit à petit. Pour cela on va utiliser une fonction qui réduira proportionnellement les paramètres des 3 diodes de la LED, on aura alors l'illusion que la couleur perd en intensité.

Si on utilisait une fonction qui diminuait de 1 en 1 la nuance de chaque diode jusqu'à atteindre le 0, on aurait un changement de couleur avant d'avoir une LED totalement éteinte. Pour avoir du orange il faut 255 en rouge, 125 en vert et 0 en bleu : si on fait une fonction qui va baisser de 1 en 1 la valeur de chaque nuance, une fois arrivée à 125 en rouge et 0 en vert la LED n'aura plus une teinte orange mais rouge ! L'effet de perte de luminosité se sera transformé en changement de couleur. Pour avoir l'effet désiré, il faudrait baisser de 2 la nuance rouge à chaque fois qu'on baisse de 1 la verte... C'est pour ça que la librairie est utile car elle inclue directement une fonction qui fait ça en une ligne !

Pour jouer sur l'intensité d'une LED on utilise la fonction `fadeLightBy()` ;

Voici un exemple de code qui allume et éteint progressivement les LED pour avoir un effet de pulsation de la couleur sur la bande :

```
int fade = 5; // vitesse de l'effet
int lumin = 0; // luminosité (entre 0 et 255)
void loop(){
  for(int i = 0; i < NUM_LEDS; i++){
    leds[i] = CRGB(150,50,150); // violet
    leds[i].fadeLightBy(lumin);
  }
  FastLED.show();
  lumin = lumin + fade;

  if(lumin <= 0 || lumin >= 255){
    fade = - fade;
  }
  delay(10);
}
```

**Explications :** Dans un premier temps on définit nos 2 variables :

- "fade" qui correspond au « pas » de changement pour la luminosité ; plus il sera grand, plus la bande changera vite de luminosité
- "lumin" qui est la valeur de cette luminosité ; en initialisant à 0, on commencera avec une bande éteinte.

Dans `void loop` on crée notre boucle pour agir sur toutes les LED en même temps, en

leur attribuant la même couleur (ici un violet) et la même luminosité.

On applique les changements avec `FastLED.show()` ; , puis on change la valeur de "lumin" en lui ajoutant "fade". "lumin" s'incrémente alors de 5 en 5 à "bouclage".

Arrivée à 255 la bande aura atteint son intensité maximale. Il faut donc décrémenter la valeur de "lumin" pour qu'elle revienne à 0. À 0 il faudra de nouveau incrémenter "lumin" pour atteindre 255 et ainsi de suite pour obtenir l'effet d'oscillation infini.

L'astuce est d'inverser le signe de "fade" afin qu'il soit positif pour incrémenter ou négatif pour décrémenter "lumin". On écrit la condition « si la valeur de "lumin" est inférieure ou égale à 0 OU supérieure ou égale à 255 » alors on inverse le signe de "fade" en utilisant le signe - (moins) qui indique qu'on veut l'opposé : si "fade" est positif il va devenir négatif et s'il est négatif il va devenir positif car *moins et moins ça donne plus* en mathématiques.

### f. Créer des fonctions pour aller plus vite !

`FastLED` propose une gamme plutôt complète de fonctionnalités mais rien ne vous empêche de créer les vôtres pour vous épargner des programmes longs et répétitifs.

Par exemple, au lieu de devoir ré-écrire à chaque fois la boucle `for` pour changer la couleur de toutes les LED, on peut créer une fonction qui contiendra cette boucle, que l'on pourra appeler dès que l'on voudra changer la couleur de toutes nos LED.

On va définir une nouvelle fonction : "change\_coul" sera son nom et entre parenthèses on met la/les variable(s) propre(s) à la fonction. Comme "change\_coul" n'est amenée à retourner aucune valeur on ajoute `void` devant son nom.

Il est important d'écrire cette fonction hors des 2 fonctions de base `setup` et `loop`, sinon vous aurez une erreur à la compilation\*.

```
void change_coul(int r, int g, int b) {
    for(int i = 0; i < NUM_LEDS; i++) {
        leds[i] = CRGB(r,g,b);
    }
    FastLED.show();
}
```

Code

Ici on a mis la boucle `for` de changement de toutes les couleurs dans notre fonction "change\_coul" : ainsi pour changer la couleur de la bande il suffira d'appeler la fonction dans `loop` en renseignant les paramètres de la couleur désirée.

Pour connaître le code RGB d'une couleur rendez-vous sur un moteur de recherche et tapez "color picker" : vous aurez accès à un nuancier qui vous donnera, pour chaque couleur, ses valeurs R,G et B.

Astuce

Maintenant que la fonction est créée on va s'en servir dans la fonction `loop` pour changer la couleur de la bande en 1 seule ligne au lieu de 4.

```
void loop() {
  change_coul(255,0,0); //rouge
  delay(1000);
  change_coul(255,255,0); //jaune
  delay(1000);
}
```

Avec ça on aura le même résultat qu'en page 31, quand on a fait clignoter la bande mais en moins de lignes et en beaucoup plus simple à paramétrer.

Voici l'exemple d'un fichier de programmation complet.

Il est conseillé de placer vos fonctions avant les fonctions `setup` et `loop` afin d'éviter tous problèmes de compilation.

## Important

Les fonctions que vous créez devront être recopiées dans tous vos projets pour les utiliser, car elles ne sont pas dans la librairie FastLED. Vous pouvez les appeler autant de fois que vous voulez, mais il faudra les inclure dans chaque projet pour vous en resservir ! Si vous appelez une fonction qui n'est pas définie dans votre code, vous aurez une erreur lors de la compilation\*.

```
#include <FastLED.h>
#define LED_PIN 4
#define NUM_LEDS 70
#define LED_TYPE WS2812
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];
void change_coul(int r, int g, int b) {
  for(int i = 0; i < NUM_LEDS; i++) {
    leds[i] = CRGB(r,g,b);
  }
  FastLED.show();
}
void setup() {
  FastLED.addLeds<LED_TYPE,LED_PIN,COLOR_ORDER>(leds, NUM_LEDS);
}
void loop() {
  change_coul(255,0,0); //rouge
  delay(1000);
  change_coul(255,255,0); //jaune
  delay(1000);
}
```

Désormais vous savez tout, ou presque, sur la programmation des LED !

Sachez qu'il existe des exemples d'effet qui sont accessibles via Fichier>Exemples>-FastLED : essayez-les, servez-vous en de base pour créer vos propres enchaînements : mais n'oubliez pas de changer les variables au début du projet pour avoir le bon `LED_PIN`, `NUM_LEDS` etc.

Vous pouvez aussi en trouver sur internet : mais là, faites attention qu'ils utilisent bien la librairie FastLED. Après, rien ne vous interdit d'utiliser une autre librairie !

### f. Envoyer les données sur la carte

Une fois que votre code est fini, qu'il n'a pas d'erreurs, il faut l'envoyer sur la carte.

Avant toute chose vérifiez que le type de la carte est bien renseigné : allez dans Outils>Type de carte et sélectionnez Arduino Nano dans la liste. Normalement si vous avez acheté la bonne Arduino Nano le processeur est un ATmega328P. Regardez sur la boîte, si jamais c'est un ATmega168 changez-le dans Outils>Processeur.

Connectez la carte en USB avec le câble fourni à votre ordinateur.

Cliquez sur le bouton Téléverser



Attendez que le message Téléversement terminé soit affiché en bas dans la console. Si vous avez une erreur du type « Problème de téléversement vers la carte », vérifiez dans Outils que le Port est bien coché (Outils>Port), et si ça ne fonctionne vraiment pas essayez de changer le processeur.

Vous pouvez déconnecter votre carte et allumer votre montage : voilà, c'est fini !

## Conclusion

Avec un peu de pratique et en vous documentant sur internet, vous arriverez à faire vos propres programmes pour allumer vos LED selon vos envies.

La programmation est un savoir qui viendra petit à petit, au début faites des copier-coller de bouts de code et essayez de les analyser pour comprendre ce qui se passe. Je mettrai sur mon site des bouts de programmes que vous pourrez utiliser afin de faire des petits effets mais après il faudra se jeter à l'eau surtout si vous voulez personnaliser vos effets et leurs enchaînements.



## IV. Aller plus loin ...

### 1. Mise en pratique

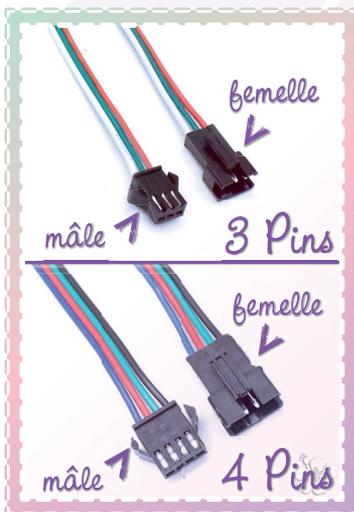
Les applications sont multiples : mettre de la lumière dans votre costume ou votre arme, enflammer votre robe, etc. Il faut juste bien poser votre projet sur le papier avant de vous lancer. Car ce n'est pas une fois votre arme finie qu'il faudra se dire : bon maintenant où je passe mes câbles ?!

#### a. Espaces de rangements

Pensez à où vous allez ranger les composants et par où vous allez faire passer vos câbles car malheureusement on n'a pas encore trouvé comment alimenter des LED sans fil. Pensez aussi à faire des petites « boîtes » ou « sacs » pour cacher vos cartes et batteries : c'est tout bête mais ça maintient tout en place et c'est plus esthétique.



#### b. Les connecteurs



Si vous avez des bouts de bande sur plusieurs zones de votre costume, il existe des connecteurs 3 ou 4 pins que vous pouvez utiliser pour relier les différentes parties de votre montage entre elles, si jamais vous avez envie que tout soit relié à la même batterie/carte. Par exemple si vous les avez mis dans le bas de votre costume avec un connecteur vous pourrez facilement relier le haut du costume sans avoir à faire des soudures entre les 2 parties : ainsi le transport et l'enfilage du costume sera plus simple : mais encore une fois, il faut bien penser les connexions et les longueurs de câbles en amont.

## c. Diffusion

Afin d'éviter de trop voir les points formés par les LED une fois allumées, vous pouvez utiliser des matériaux tels que le polystyrène ou la mousse comme diffusant : ça donne l'impression d'avoir une lumière répartie.

Vous pouvez également orienter vos LED à 90° pour ne voir que la lumière qui se dégage sur les côtés de la bande et ainsi limiter l'effet « points lumineux ».



## d. Les gaines thermorétractables



On peut les utiliser pour sécuriser les soudures entre 2 câbles : ça isole les parties où les fils sont à nu ! Il en existe de différents diamètres et vous pouvez trouver des packs bon marché sur les boutiques en ligne.

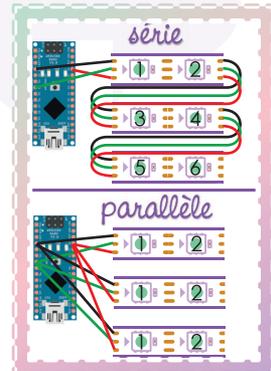
Pour les fixer, il faut simplement les placer sur votre bout de fil à nu et les chauffer avec un décapeur thermique.

Comme ce sont des tubes : pensez à les passer dans vos câbles avant de les souder entre eux, sinon vous ne pourrez plus enfiler vos gaines !

## e. Série ou Parallèle ?!?

Si vous avez besoin de répéter une séquence sur différentes parties de votre costume, pensez au montage en parallèle : ainsi vous aurez autant de LED n°1, n°2, n°3, etc. que vous aurez de sections de bandes. Lors de votre programmation vous aurez simplement à faire le code pour une bande pour que ça s'applique à toutes.

Par contre, il ne sera plus possible d'agir sur chaque LED indépendamment, donc il faut être bien sûr de ce que vous voulez faire !





Par exemple, pour l'effet de feu sur ma robe de Katniss, j'ai utilisé 25 bandes de 30 LED montées en parallèle. Ainsi j'ai juste eu à programmer mon effet pour 30 LED pour que cela s'applique à toutes les bandes. Une solution pratique car mon but était d'avoir le même effet partout !

## Conclusion

A vous de bien réfléchir à comment mettre en place votre projet, mais le mot d'ordre c'est : préparation !!! N'hésitez pas à faire un schéma du câblage complet de votre costume sur l'image de référence par exemple, pour bien définir toutes les zones à illuminer, comment les relier entre elles, où cacher les connectiques et le système d'allumage, où prévoir un interrupteur discret et accessible par vous-même pour éviter de devoir demander à vos amis de vous « allumer » à chaque fois. Une bonne préparation est toujours la clé de la réussite et avec de la pratique vous arriverez à faire des projets de plus en plus fous !

*Astuce*  
Pensez à vos marges de coutures car il est facile d'y passer des câbles ou fils. En rajoutant une couture parallèle, vous obtiendrez une sorte de gaine pour bien les camoufler.

## 2. DIY : Connecteur USB

### Matériel

> Prise USB en kit



> Fils électriques  
Ø ≈ 0,08 mm  
(rouge et noir)

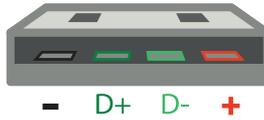


Dans la section III.3 pour alimenter le circuit on a besoin de le connecter à une batterie USB et pour gagner de la place on va créer nous-même le connecteur.

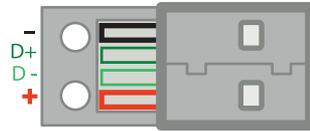
Le plus compliqué c'est de savoir où souder les fils : car un connecteur USB à 4 entrées, le positif et le négatif que nous allons utiliser, ainsi que la Data + et Data - qui servent à l'échange de données mais qui ne nous sont pas utiles pour l'alimentation.

Ci-contre 2 schémas explicatifs des 4 pins d'une prise USB.

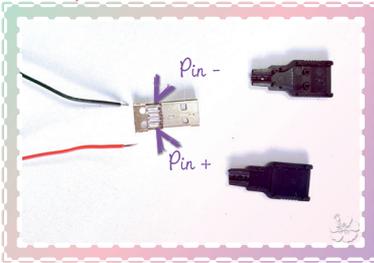
USB vu de face



USB vu de dessus

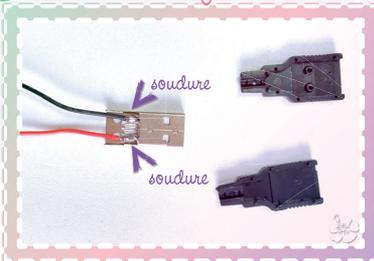


## 1 Préparation



Dans un premier temps, positionnez vos éléments sur une table en faisant de sorte que les fils soient correctement placés par rapport au schéma de la prise USB ci-dessus.

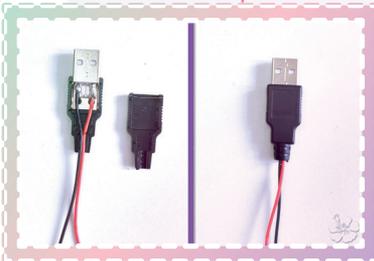
## 2 Soudure des fils



Soudez les fils aux pins à l'aide d'un fer à souder.

**Attention :** veillez à ne pas connecter le pin D+ avec le pin -, ainsi que le pin D- avec le pin +, en soudant vos fils : soyez précis !

## 3 Fermeture de la prise



Refermez votre prise USB en faisant passer les 2 fils par le trou.

Il ne vous reste plus qu'à connecter le fil rouge sur votre interrupteur et le fil noir sur la sortie de votre montage.

Si vous n'avez pas envie de faire votre connecteur USB maison, vous pouvez toujours acheter des câbles USB to DC et connecter votre montage à un connecteur DC.

### 3. Le mot de la fin !

Désormais vous connaissez les bases sur les bandes LED pour rajouter de la lumière dans vos costumes et autres créations.

Personnellement comme je m'y connais en programmation je préfère les bandes programmables, les autres, je les utilise principalement dans mes décors pour rajouter de la lumière d'ambiance quand je n'ai vraiment pas envie de me prendre la tête.

Si jamais j'ai besoin que la lumière de mon costume ait un allumage progressif, un changement de couleur ou n'importe quel autre effet, je vais sans hésiter prendre des bandes programmables car j'aurai beaucoup plus de liberté.

De même, si pour une prestation j'ai besoin d'agir sur la couleur ou les effets, je vais directement caler ma programmation sur ma prestation pour n'avoir qu'à allumer le tout à un instant T et ne plus me préoccuper d'actionner tel ou tel bouton pour faire un changement de couleur ou de luminosité, ça fait du stress en moins !

Par exemple dans ma robe de Katniss, j'ai rajouté des LED pour faire un effet de robe qui s'enflamme lors de la transformation : j'avais fait un programme qui était calé sur ma prestation pour que dès que j'actionne l'interrupteur les LED s'allument une à une de bas en haut et qu'elles restent allumées durant le temps de la transformation puis s'éteignent toutes seules une fois le changement fini.

La programmation laissera libre court à vos envies et n'est pas inaccessible.

C'est comme pour tout : il suffit de se lancer !

J'espère vraiment que ce livre aura été utile, et que son contenu vous aura appris des choses.

Si jamais vous avez des questions ou des remarques, retrouvez-moi sur les réseaux sociaux. Cherchez **nadcosplay** et je serai là !



Retrouvez également en complément de ce livre, des tutoriels en vidéo, des exemples de codes à télécharger pour vos projets ainsi que tout un tas d'autres informations pratiques sur les bandes LED en flashant le code ci-contre.

## Remerciements

Merci à Gectou4 et Alexandre d'avoir relu pour que les textes soient compréhensibles et les explications claires.

Merci à Douchan, Laura, Félonia, Orena et Guillaume d'avoir corrigé les (nombreuses) fautes.

Merci à Fabien de m'avoir fait découvrir les bandes WS2812b.

Et merci à tous ceux qui ont lu ce livre !



# GLOSSAIRE

**LED** : (en français : DEL : diode électroluminescente) est un composant électronique et optique, qui en étant traversé par du courant électrique, émet une lumière d'une intensité diffuse. Les LED consomment peu d'électricité.

**RGB** : de l'anglais « Red, Green, Blue », est, des systèmes de codage informatique des couleurs, le plus proche du matériel. Le codage RGB indique une valeur pour chacune de ces couleurs primaires. Il peut aussi être écrit en français sous la forme RVB (Rouge Vert Bleu).

**C++** : Langage de programmation compilé permettant la programmation sous de multiples paradigmes (comme la programmation procédurale, orientée objet ou générique).

Créé initialement par Bjarne Stroustrup dans les années 1980, le langage C++ est aujourd'hui normalisé par l'ISO.

**Compilation** : En informatique, elle désigne le processus de transformation d'un programme écrit dans un langage lisible par un humain en un programme exécutable par une machine. De manière plus générale, il s'agit de traduire un programme écrit dans un langage source en un programme écrit dans un langage cible.

**Moniteur série** : On le nomme ainsi car il utilise la communication dite "série" entre votre ordinateur et votre carte Arduino qui sont connectés via leurs ports "USB".

Il sert à avoir un retour de données afin de déboguer les dysfonctionnements dans les programmes en y affichant des données utiles au traitement du bogue.

Il s'ouvre en cliquant sur le bouton  en haut à droite du logiciel Arduino.

Pour l'utiliser, ajouter la ligne `Serial.begin(9600)` ; dans la fonction `setup` de votre programme. Puis rajouter la ligne `Serial.print(ma_variable)` ; dans votre code, pour voir le contenu de "ma\_variable" s'afficher dans le moniteur à l'exécution.

# MERCI POUR VOTRE SOUTIEN À BIENTÔT POUR DE NOUVEAUX TUTOS !



## Présentation

Plongez-vous dans l'univers des bandes LED pour illuminer vos créations.

Découvrez au travers de ce livre, comment facilement les utiliser, les connecter, les alimenter et les programmer.

Cet ouvrage traite principalement des bandes **RGB** classiques simples à mettre en place et des bandes **WS2812b**, certes un peu plus complexes à prendre en main, mais qui permettent de faire des effets beaucoup plus spectaculaires.

N'hésitez plus : l'illumination vous attend au détour de ce mini livre qui se veut accessible pour tous, avec des tutoriels en images détaillées et moult explications.

@NADCOSPLAY

Livre protégé par le droit d'auteur

PRIX CONSEILLÉ 10€

 *Nad Cosplay*  
mimigyaru.c♡m



     Nadcosplay   Nadcosplayvideo  nadcosplay@gmail.com