howTo_doc Documentation

Poltergeist42

oct. 10, 2018

Contents:

1 README **1**

2 Informations générales sur le projet howTo_doc **3**
- 2.1 Description . . . 3
- 2.2 Comment utiliser ce document . . . 4
- 2.3 Arborescence du projet . . . 4

3 Comment documenter un projet **5**
- 3.1 Avant d'attaquer . . . 5
- 3.2 Les bonnes questions qu'il faut se poser pour documenter un projet . . . 6
- 3.3 Pour qui et pourquoi ? . . . 6
- 3.4 Quoi et comment ? . . . 6
- 3.5 Avant, pendant et après la documentation . . . 19
- 3.6 Les licences . . . 21
- 3.7 Les finitions . . . 22
- 3.8 Conclusions . . . 23

4 Bug et ToDo-list **25**
- 4.1 Description . . . 25
- 4.2 Model Type . . . 25
- 4.3 Bugs identifiés . . . 26
- 4.4 ToDo-list . . . 26

5 VoLAB **29**
- 5.1 Nous connaître . . . 29
- 5.2 Qui sommes nous ? . . . 29

CHAPITRE 1

README

CHAPITRE 2

Informations générales sur le projet howTo_doc

Auteur Poltergeist42
Relecteur MajorLee95
Projet howTo_doc
Version 20181010
dépôt GitHub https://github.com/volab/howto_doc
doc GitHub https://volab.github.io/howto_doc/
doc ReadTheDocs https://howto-doc.readthedocs.io

2.1 Description

Ce projet a pour objectif de fournir quelques clefs et idées pour aider à la documentation d'un projet.

Il ne s'agit pas d'une doctrine qu'il faut respecter absolument, mais simplement de la façon dont j'envisage la documentation. Tout au long de ce projet, je vais essayer de vous décrire mes petits trucs ainsi que quelques outils vous permettant de réaliser la documentation.

Ma philosophie étant que : projet et documentation sont indissociables l'un de l'autre, ce document vous présentera également différents éléments pour la gestion de l'organisation des projets.

2.2 Comment utiliser ce document

Ce document est construit sous la forme d'un site web. La navigation se fait par le menu sur la gauche de la fenêtre ou en cliquant sur la rubrique désirée sur la partie centrale de la fenêtre.

Pour revenir sur la page d'accueil, il faut cliquez sur : « Documentation howTo_doc » sur la gauche de la fenêtre (en haut et en bas).

Ce document est constitué de quatre éléments :

- README : Donne une brève description du projet et donne quelques informations supplémentaires (Auteur, Licence, Version, etc.).
- Comment documenter un projet : C'est l'élément principale traitant du sujet qui nous intéresse.
- Bug et ToDo-list : Cet élément sert à l'élaboration du projet (et à sa correction).
- VoLAB : Cet élément présente le VoLAB, un super FabLab situé à Vauréal dans le Val d'Oise.

2.3 Arborescence du projet

Pour aider à la compréhension de mon organisation, voici un bref descriptif de l'arborescence de se projet. Cette arborescence est à reproduire si vous récupérez ce dépôt depuis GitHub.

```
ProjectDir_Name         # Dossier racine du projet (non versionner)
|
+--project              # (Branch master) contient l'ensemble du projet en lui même
|   |
|   +--_1_userDoc       # Contiens toute la documentation relative au projet
|   |   |
|   |   \--source       # Dossier réunissant les sources utilisées par Sphinx
|   |
|   +--_2_modelisation  # Contiens tous les plans et toutes les modélisations du projet
|   |
|   +--_3_software      # Contiens toute la partie programmation du projet
|   |
|   \--_4_PCB           # Contient toutes les parties des circuits imprimés (routage,
|                       # Implantation, typon, fichier de perçage, etc.
|
\--webDoc               # Dossier racine de la documentation qui doit être publiée
    |
    \--html             # (Branch gh-pages) C'est dans ce dossier que Sphinx va
                        # générer la documentation a publié sur internet
```

CHAPITRE 3

Comment documenter un projet

3.1 Avant d'attaquer

Lorsque l'on fait une simple recherche sur internet avec les mots clefs :

```
comment documenter un projet
```

On se rend compte qu'il y a déjà beaucoup d'informations disponible sur ce sujet. Je n'hésiterai d'ailleurs pas un instant en vous renvoyant de temps à autres vers l'un ou l'autre de ces sites.

Liens_Web

- **http://cours.education/dufacilitateur/2016/07/24/atelier-comment-documenter-son-projet-v1/** # Une aide à la documentation proposée par le FacLab
- **http://carrefour-numerique.cite-sciences.fr/fablab/wiki/doku.php?id=aide:documenter** # Une aide à la documentation proposée par Le Carrefour Numérique
- **http://blog.wikifab.org/2016/07/19/pourquoi-documenter-et-partager-mon-projet/** # Une description de pourquoi, pour qui donnée par WikiFab
- **https://reso-nance.org/wiki/_media/projets/documentation-fablab.pdf** # Un petit PDF résumant les étapes de documentation
- **http://www.fao.org/capacity-development/resources/practical-tools/comment-documenter-et-partager-les-bonnes-p** # Comment capitaliser et partager vos bonnes pratiques pour générer le changement

3.1.1 Pourquoi encore un document sur « comment faire de la doc »

Tous d'abord parce qu'il est toujours intéressant d'avoir plusieurs points de vue sur un sujet lorsque l'on décide de s'y intéresser, pour pouvoir se forger sa propre opinion. Ensuite ce document n'est pas quelque chose que vous devrez suivre à la lettre pour avoir la garantie de faire une bonne documentation. Il s'agit simplement de vous décrire au mieux ma propre recette. Vous pourrez ainsi piocher (ou non) quelques idées pour définir votre propre méthode.

Ce document apporte en plus des éléments sur la façon de gérer et d'organiser un projet, ce qui peut être utile si vous n'avez pas encore défini votre propre méthodologie de gestion de projet.

3.2 Les bonnes questions qu'il faut se poser pour documenter un projet

Il y a deux thèmes importants dans la documentation :

— **Pour qui et pourquoi** : ce thème permet de définir la cible principale à qui s'adresse la documentation
— **Quoi et comment** : ce thème permet de définir les outils nécessaires à la documentation et à l'organisation de votre projet.

3.3 Pour qui et pourquoi ?

Dans la plupart des lectures que j'ai pu consulter, il revient une idée générale avec laquelle je ne suis pas d'accord. Cette idée, c'est :

> *« Il faut documenter afin de partager son travail avec d'autres personnes »*

Pour ma part, je pense que le travail de documentation doit d'abord être un travail égoïste. La documentation doit être faite en priorité pour soi, mais écrite pour quelqu'un qui ne sait pas de quoi il s'agit et qui découvrirai complètement votre projet. De cette façon la documentation sera plus complète et plus précise.

> **Exemple** Imaginez que vous soyez tout le temps en train de faire 50 trucs en même temps, plus les activités des enfants, le travail, les courses etc ... (dur à imaginer n'est-ce pas ? :p). Dans cette situation il peut arriver qu'un projet soit mis de coté pendant plusieurs mois voire plusieurs années.
>
> Lorsque vous allez enfin pouvoir reprendre votre projet, vous aurez oublié une grande partie du million de détails qui constituent votre projet. Trois choix se présentent alors :
> — Vous reprenez le projet depuis le début pour essayer de comprendre comment vous avez fait telle opération ou pourquoi vous aviez privilégié une solution plutôt qu'une autre.
> — Vous remettez votre projet sous les tas de poussière sous lesquel il était enterrer et vous vous empressez de l'oublier consciencieusement car le reprendre depuis le début représenterait un investissement en temps et en pressage de citron beaucoup trop important.
> — Vous ouvrez simplement la documentation que vous avez pris soin d'écrire tout au long du projet, ce qui vous permet de reprendre rapidement et facilement vos travaux.

Vous l'aurez compris, vous êtes le premier destinataire de la documentation que vous allez produire. Cependant, comme vous aurez pris la peine de vous faire une doc (et donc elle existe déjà), il ne vous reste plus qu'à la partager avec la Terre entière puisque que cela ne vous coûtera pas plus de travail.

3.4 Quoi et comment ?

Que faut-il mettre dans la documentation ?

A cette question ma réponse est : Tous ce que vous pouvez !

— Toutes les idées que vous avez eues, celles que vous avez expérimenté, pourquoi certaines n'ont pas été conservées, pourquoi les autres oui.
— Tous les croquis que vous avez dessinez sur un coin de nappe (avant de vous apercevoir qu'elle était en tissu ;))
— Vos succès, vos échecs, vos erreurs
— Les objectifs majeurs, mineurs et transitoires du projet
— La fiche de description du projet
— les plans de fabrication

— Les listes des matériaux avec leur provenance et leur prix
— les schémas électroniques, les typons et les PCB
— La liste des composants électronique avec leur provenance et leur prix
— le code source du programme
— Les sources des informations que vous utilisez pour avancer dans votre projet

En résumé, tous ce que vous avez produit ou utiliser pour réaliser votre projet.

Il ne faut jamais trop se faire confiance.

Si vous pensez :

> *« ça c'est bon, je m'en souviendrai. Je n'ai pas besoin de le noter ! »*

Ou :

> *« c'est parfaitement compréhensible, je n'ai pas besoin de l'expliquer ! »*

Vous pouvez être certains que NON vous ne vous en souviendrez pas et que NON ce n'est pas si compréhensible que ça quand vous n'avez pas touchez au projet depuis 6 mois. Il vous faut donc systématiquement noter et documenter même les choses évidentes et / ou apparemment simple.

Soyez clair et efficace dans vos explications.

N'hésitez pas à lire à haute voix ce que vous avez écrit. Cela vous aidera à définir si ce que vous avez écrit est :

— Suffisamment explicite
— Pertinent
— Pas trop compliqué. Pas besoin d'avoir une doc pour comprendre la doc (qui a dit : « meuble Ikéa » ?).
— Pas trop verbeux. Il s'agit d'une documentation, pas d'un essai littéraire. Il ne faut pas que les informations soit perdues dans un flot de blablas inutile.

3.4.1 Des images Oui, mais pas que !

L'ajout d'image, de croquis, de graphique ou d'illustration est toujours une bonne chose car cela peut aider à la compréhension. Attention toutefois à ne pas en abuser. Il ne faut pas transformer votre doc en BD ou en rébus.

— Ajoutez un support visuel que si ça amène une plus-value à votre doc.
— Accompagnez toujours vos illustrations d'une légende et / ou d'un commentaire.
— Ne pas concentrer trop d'informations visuelles au même endroit, car cela masque le texte qui est souvent plus important.

Petites Astuces

1. Pour identifier plus facilement à quel projet une image appartient (en particulier si vous publiez votre doc sur un WIKI), vous pouvez ajouter un préfixe au nom de votre image.

```
ex:

projet : "patate au four"

image : "carrotte_bleu.jpg"

Nom de l'image associé au projet
PAF_carrotte_bleu.jpg
```

3.4.2 Boite à idées

Une idée n'est bonne que si on s'en souvient !

C'est toujours une bonne pratique que de noté toutes vos idées dans un document. Vous pouvez avoir plusieurs boites à idée. Une globale dans laquelle vous allez décrire les idées qui vous permettront de commencer d'autres projets qui n'auront peut-être rien à voir les uns avec les autres. Et une par projet qui vous servira à améliorer ou à développer votre projet.

Petites Astuces

1. Tant que vous n'avez pas commencer le projet, laissez-le dormir dans la boite à idées. Cela vous évitera d'avoir des tonnes de dossiers projets vide qui ne contiennent en tout et pour tout que la description globale de votre idée.
2. Utilisez, pour votre boite à idées, la même structure / arborescence que pour vos projets.

   Souvent lorsqu'on a une idée, elle nous enthousiasme tellement qu'on a tendance à faire tout de suite des recherches dessus. Si vous avez organisé votre boite à idée comme un projet, vous allez pouvoir y stocker le fruit de vos recherches en attendant que votre idée devienne un vrai projet.

3.4.3 Documenter régulièrement

La plus grosse erreur que vous pouvez faire c'est de vouloir faire la documentation quand tous le reste sera terminé. Si vous vous dites cela, vous pouvez être certain que vous ne ferez pas de documentation. Voici par exemple certaines raisons qui vous en empêcherons :

— Il y a de grandes chances pour que vous enchainiez directement sur un autre projet.
— Vous n'aurez pas le temps car vous serez trop occupé ailleurs (souvenez-vous, la vie, le travail, les enfants tout ça).
— Faire la documentation en dernier équivaux à recommencer tout le projet depuis le début. On se retrouve dans la même situation que celle qui a été donné en exemple un peu plus haut. Je suis certain qu'à ce moment vous serez atteint d'un mal assez connu que l'on nomme : « La flemme ».

Pour éviter cette situation il faut documenter régulièrement vos travaux, consigner toutes vos expérimentations, prendre des notes sur vos manipulation (voir « *Le Journal de Manip* ») et si possible faire une fiche descriptive du projet à partir de laquelle vous pourrez définir plusieurs petits objectifs qui seront plus faciles à réaliser et donc à documenter.

3.4.4 Journal de manip, Bug et TODO list

Le Journal de manip

Un projet n'est jamais réalisé de façon linéaire. Vous serez souvent amené à vous détourner de votre objectif principal afin de découvrir de nouvelles choses, expérimenter une nouvelle technique, tester un outil ou une appli ou encore tester du matériel. Tout cela représente de l'expérience.

L'expérience, c'est comme les idées. Cela ne sert à rien si on ne les conserve pas. Pour éviter de perdre toutes ces précieuses expériences, vous pouvez créer **un journal de manip**.

Le journal de manip, est un document dans lequel vous allez noter tous ce que vous avez pu faire ou tester sur un sujet donné. Vous pouvez faire un journal de manip par projet, mais je vous conseil de faire un journal de manip global car au cours de vos différents projets vous allez certainement travailler sur des thèmes communs ou similaires.

Ce journal de manip vous servira également de support pour votre documentation.

Vous pouvez consulter mon propre journal de manip. Je ne l'ai pas organisé comme un journal dans lequel je saisie toutes mes manips au quotidien, mais comme un référentiel technique dans lequel je répertorie les éléments sur lesquels j'ai été amené à travailler une ou plusieurs fois.

A vous de trouver l'organisation qui vous convient le mieux pour la gestion de votre Journal de manip.

Bug et TODO list

Tout au long du développement de votre projet, vous allez le tester et le faire évoluer. Au cours de ces tests, il arrive souvent que l'on constate un défaut ou un point qu'il faudra améliorer. Si ces défauts ne sont pas bloquant, il n'est pas nécessaire d'interrompre le travail en cours. Cependant, pour ne pas oublier que ces choses sont à faire, je vous conseille de créer un document dans lequel tous ces problèmes sont référencés.

Voici le modèle que je me suis défini :

```
Model Type
==========

    :Date de saisie:        Date à laquelle la problématique a été identifiée
    :Date de traitement:    Date du traitement de la problématique
    :Cible:                 [userDoc, modelisation, software, PCB, autre]
    :Statu:                 [NONE, WIP, DONE]
    :Problématique:         Descriptif de la problématique
    :Traitement:            Descriptif du traitement de la problématique
```

Comme vous pouvez le constatez le modèle est assez simple. Les 2 seuls éléments qui peuvent poser problème sont :

- **Cible** : C'est ici que je renseigne la « catégorie » de l'élément impacté comme la doc, le programme, le matériel, etc. Ces éléments sont extraits de ma propre arborescence de projet que vous pourrez découvrir un peu plus bas.
- **Statu** : C'est ici que je renseigne l'état d'avancement du travail à faire
  - None : Le travail n'est pas commencé
  - WIP : (Work In Progress) Travail en cours
  - Done : Travail fini

Vous pouvez consulter le fichier Bug_ToDoLst de ce projet pour voir comment je l'utilise.

Tout comme moi, vous pouvez intégrer ce document à la documentation du projet.

N. B : N'hésitez pas à joindre tous vos documents de travail dans votre documentation car ils représentent de l'information que vous serez content d'avoir après une longue pause dans le projet

Petites Astuces

1. Il arrive souvent que nos idées nous viennent au cours d'un moment d'oisiveté, d'un moment de détente ou tous simplement quant ce n'est pas le moment. Pour être certain de conserver cette brillante idée jusqu'a ce que vous soyez enfin devant un PC, prenez l'habitude de toujours avoir avec vous un petit blocnote et un stilo. Vous pourrez ainsi preserver votre idée des courrant d'air en la gardant bien au chaud. Vous n'aurez plus ensuite qu'à l'étofer et à l'ajouter au reste de votre documentation.

3.4.5 Un peu d'organisation

De façon générale, vous ne pourrez pas travailler efficacement si vous ne tentez pas un tout petit peu de vous discipliner et d'organiser votre espace de travail et votre travail lui-même.

Il y a quelques bonnes pratiques que vous pouvez adopter. Vous les trouverez peut-être un peu contraignantes au début mais lorsque vous vous y serez habitué, vous serez content de retrouver toujours les mêmes types d'éléments au même endroit.

Un peu de paresse est bon pour la santé

Si vous faite de la programmation, vous avez peut-être déjà rencontré l'expression DRY (Don't Repeat Yourself) qui signifie : Ne te répète pas toi-même.

Il faut faire attention à ne pas se répéter. Il serait dommage de documenter 2 fois une partie du projet parce que cette partie en question est référencer à plusieurs endroits dans votre bazar (pas si) organisé.

On peut également étendre le concept à : ne répètes pas ce que les autres ont déjà dit. Il est inutile de faire du copier-coller (ou même de réécrire) quelque chose qui a déjà été écrit. Il suffit de mettre un lien dans votre documentation pointant vers l'endroit où l'information existe déjà.

N.B : N'oubliez pas de cités les sources et les auteurs des informations d'une tiers partie que vous incorporez dans vos documents.

Uniformiser les projets

Lorsqu'on travaille sur un projet, on peut être amené à manipuler de nombreux éléments différents comme :

- le code source d'un programme
- les schémas électroniques
- des plans de fabrication
- les docs techniques que vous avez récupérés à droite, à gauche
- Vos propres notes et documentation
- ETC.

Pour gérer ces documents, vous avez plusieurs solutions :

- **Tout réunir dans un seul dossier** : Vous aurez tous les éléments au même endroit mais il n'y aura certainement pas d'organisation logique
- **Tout répartir à différents endroits sur votre disque dur** : Vous aurez un semblant d'organisation mais il deviendra difficile de partager votre projet en l'état. Vous serez donc obligé de regrouper tous les éléments en un même endroit quand vous souhaiterez le diffuser. Créant ainsi un doublon des différents éléments et augmentant la difficulté de maintenir le projet.
- **Une troisième solution** et de créer un dossier pour le projet et de créer des sous répertoire pour l'organisation des différents documents. Vous aurez donc tous le projet dans un seul répertoire, une meilleure organisation du projet et une meilleure facilité de maintenance et de diffusion.

La troisième solution est la bonne, mais comment allez-vous organiser le prochain projet ? En créant un nouveau dossier principal et une nouvelle sous arborescence.

C'est à ce moment qu'il faut un peu de discipline (ce que j'appelle la paresse organisée). Vous devez-vous définir une arborescence standardisée dans laquelle vous aurez toujours les mêmes noms de dossiers et les mêmes modèles de fichiers. Ce qui vous permettra de ranger les différents types de documents toujours de la même façon quel que soit votre projet.

Vous devez utiliser la même arborescence dans tous vos projets pour vous faciliter le travail

Voici en exemple l'arborescence que je me suis défini

```
ProjectDir_Name         # Dossier racine du projet (non versionner)
|
+--project              # (Branch master) contient l'ensemble du projet en lui même
|  |
|  +--_1_userDoc        # Contient toute la documentation relative au projet
```

(suite sur la page suivante)

(suite de la page précédente)

```
|   |   |
|   |   \--source       # Dossier réunissant les sources utilisées par Sphinx
|   |
|   +--_2_modelisation  # Contient tous les plans et toutes les modélisations du projet
|   |
|   +--_3_software      # Contient toute la partie programmation du projet
|   |
|   +--_4_PCB           # Contient toutes les parties des circuits imprimés (routage,
|   |                   # Implantation, typon, fichier de perçage, etc.)
|   |
|   +--_5_techDoc       # Contient toutes les documentations techniques des différents␣
↪composants
|   |                   # ou éléments qui constituent le projet. Ces éléments sont␣
↪identifiés
|   |                   # par un liens Web dans la documentation. Ce dossier n'est pas
↪"poussé"
|   |                   # dans le dépôt distant (.gitignore).
|   |
|   +--_6_research      # Regroupe toutes les recherches relatives à l'élaboration ou␣
↪au
|   |                   # développement du projet. Ces éléments sont identifiés
|   |                   # par un liens Web dans la documentation. Ce dossier n'est pas
↪"poussé"
|   |                   # dans le dépôt distant (.gitignore).
|   |
|   \--_7_rushes        # Contient tous les éléments qui seront potentiellement␣
↪intégrés dans la
|                       # doc ou dans le projet. Ce dossier n'est pas "poussé" dans le␣
↪dépôt
|                       # distant (.gitignore).
|
\--webDoc               # Dossier racine de la documentation qui doit être publiée
    |
    \--html             # (Branch gh-pages) C'est dans ce dossier que Sphinx va
                        # générer la documentation a publié sur internet
```

Pour être certain d'utiliser toujours la même arborescence, vous devez limiter le nombre d'actions à faire à la main. Pour cela, vous avez 2 solutions :

- **Solution 1** : Créer un modèle de projet (avec toute son arborescence). A chaque nouveau projet, vous copiez le modèle à l'endroit où vous voulez créer votre projet et vous renommer cette copie avec le nom du projet.
- **Solution 2** : Vous vous faites un petit programme qui va créer pour vous tout les répertoires et sous répertoire nécessaires. Ce genre de programme est très simple à faire et ce quel que soit le langage de programmation que vous utilisez. Vous pouvez faire en sorte que ce programme mette en place une structure plus évoluée comme par exemple : initialiser GIT et Sphinx en même temps que la création de l'arborescence du projet.

La deuxième solution est plus compliquée à mettre en œuvre, mais elle vous facilitera vraiment le processus de création d'un nouveau projet.

Vous pouvez regarder mon projet ArboProject écrit en python. Ce programme me crée une arborescence, copie ou crée certains fichiers, initialise GIT et Sphinx dans la foulée. De plus je peux modifier mon arborescence en modifiant simplement un fichier JSON.

Petites Astuces

1. Si vous utilisez toujours la même arborescence, vous aurez forcement des dossiers

vides. Pour être sûre de ne pas passer votre temps à ouvrir ces dossiers vides ajoutez le suffixe "_v" au nom de chaque dossier. Vous pouvez même le faire directement dans votre modèle. De cette façon, lorsque vous ajoutez des données dans un dossier, il vous suffit de retirer se suffixe pour obtenir immédiatement un indicateur visuel sur l'état (data ou vide) d'un dossier.

```
Ex:
# Pas de données
|
\--_1_userDoc_v

# Dossier avec données
|
\--_1_userDoc
    |
    \--MaSupperDoc.txt
```

2. En plus d'une arborescence standardisée, il est aussi possible de créer un certain nombre de fichiers qui seront toujours structurés de la même façon. Je vous conseille de créer un fichier : README à la racine de votre projet. Ce fichier devrait contenir les éléments suivants :
   — Le ou les auteurs du projet (normalement vous)
   — Le ou les relecteurs / correcteurs de la documentation
   — La licence sous laquelle le projet est distribué
   — les informations sur la documentation et sur le dépôt (endroit de mise à disposition) du projet.
   — le numéro de version de la documentation
   — Une brève description du projet,
   — Quelques informations permettant le démarrage ou la prise en main du projet
   — Quelques informations supplémentaires comme par exemple l'arborescence du projet

   Mon programme "arboProject" me crée se fichier automatiquement. Je fais systématiquement apparaître ce fichier en premier dans ma documentation.

   Vous pouvez consulter le fichier README de ce projet pour voir à quoi cela ressemble.

Versionner et nommer les fichiers

Versionner des fichiers

L'un des problèmes que l'on rencontre souvent est :

> « *qu'elle est la bonne version du fichier à utiliser et comment le nommer de façon intelligente et compréhensible ?* »

Exemple Vous venez de terminer le travail sur un fichier. Étant certain que vous n'aurez plus à travailler dessus, vous le renommer en "final" : "*mon_super_fichier_final.txt*". Seulement, le lendemain, vous vous apercevez qu'en fait vous avez oublié de parler d'un truc important. Vous modifier votre fichier et là cette fois c'est sûr, c'est la version finale. Comme vous ne voulez pas écraser l'ancien fichier, vous l'enregistrer en tant que "*mon_super_fichier_final_final.txt*". Puis quelques mois après, vous faites une nouvelle modif alors vous enregistrez le fichier sous : "*mon_super_fichier_final_final_V1.txt*".

Je pense que vous avez compris où je voulais en venir.

Pour éviter ce genre de problème il existe une technique simple que l'on nomme : « l'horodatage ». Cette technique consiste à ajouter, en préfixe, la date en version courte au nom de votre fichier. Ce préfixe est formaté de la façon suivante :

```
AAAAMMJJ
```

Vous pouvez encore simplifier cette notation en considérant que vous ne modifierez pas ce fichier dans 100 ans ou dans 1000 ans. Cela donne :

```
AAMMJJ
```

Votre ficher aura alors la forme :

```
AAMMJJ_[Nom_du_fichier].ext

Ex:
    '180825_mon_super_fichier.txt'
```

Pour ma part je me contente de préfixer mes fichiers avec la date. Il peut malgré tout y avoir des situations ou vous souhaiterez avoir une information plus précise sur l'horodatage du fichier. La solution est alors d'ajouter l'heure en plus de la date :

```
AAMMJJ-HHMM[SS]_[Nom_du_fichier].ext
# N. B : on peut ne pas spécifier les secondes

Ex:
    '180825-1843_mon_super_fichier.txt'
```

Cette technique présente plusieurs avantages :

- Vous évitez les noms prêtant à confusion
- Vous pouvez repérer immédiatement la version du fichier que vous souhaitez consulter simplement en consultant son nom
- Lorsque, dans votre explorateur de fichier, vous classerez vos fichiers par nom, ceux-ci seront également automatiquement classé par ordre chronologique C'est tout l'intérêt d'écrire l'année en premier, suivi du mois !

Versionner tout un projet

Dans la vie d'un projet, il est parfois nécessaire de tester une partie sans défaire la partie déjà fonctionnelle. Pour cela, il faut pouvoir créer une version fonctionnelle et une version d'essais. La technique de l'horodatage est efficace au niveau des fichiers, mais pas au niveau des projets.

La solution est donc d'utiliser un logiciel gestion de version. Le plus connu (et le plus utiliser) est GIT. Les liens sont disponibles dans *Les outils de production*.

GIT permet de conserver toutes les versions de tous les fichiers. Il offre ainsi la possibilité de comparer la version actuelle avec une version antérieure d'un fichier. Il permet également de créer plusieurs branches de travail :

- La branche **"Master"** : C'est la branche principale (Obligatoire). Elle est utilisée pour les versions Stable et / ou fonctionnelles des projets.
- La Branche **"gh-pages"** : Cette branche devra obligatoirement être créer si vous utilisez la fonctionnalité de gestion / publication de la documentation de GitHub : github.io
- Les autres branches que vous devrez créer, sont utilisées pour les versions de développement ou pour tester de nouvelles fonctionnalités.

Il s'utilise en ligne de commandes et permet de porter vos projets vers des dépôts distant comme GitHub ou GitLab (voir *Les outils de publication des projets et de la documentation*)

Pour les allergiques à la ligne de commande, il existe Tortoise (Outils de production) qui est une interface graphique pour GIT. Il s'intègre dans le menu contextuel de Windows.

Version et révision

Liens_Web

- **https://fr.wikipedia.org/wiki/Version_d%27un_logiciel** # Page d'explication des versions logiciel
- **https://semver.org/lang/fr/** # Gestion sémantique de version

Il est important, pour vous aussi bien que pour d'autres personnes qui souhaiteraient utiliser votre projet ou consulter votre documentation, d'avoir des numéros de versions sur vos documents ou autres éléments du projet.

Il est toujours difficile de définir une bonne nomenclature pour affecter un numéro de version. Cette décision vous revient. Il y a plusieurs écoles sur ce point. La plus courante étant : « SemVer » (Semantic Versioning).

```
Majeur.Mineure.correctif

ex:
2.6.2   --> le correctif 3 (la numérotation commence à 0) pour la révision Mineure 6
↪de la version 2
```

Consultez les liens ci-dessus pour plus de détail.

Pour ma part je préfère ne pas utiliser SemVer car je trouve se procéder trop contraignant. Puisque mes projets sont construits dans le temps, j'utilise la date comme numéros de versions. Pour différencier les versions de développement des versions stables (en plus de ma gestion des Branches dans GIT), j'ajoute le suffixe « -dev » à la suite de la date.

```
Format:
AAAAMMJJ

ex:
20180913
# version stable

20180914-dev
# version en cours de développement
```

On constate que je pourrais simplifier mes numéros de version en supprimant les 2 premiers digits de l'année. Malheureusement, je n'y avais pas pensé quand j'ai adopté ce système de notation. Par soucis de cohérence, je continue donc avec ce format.

Vous comprenez pourquoi il faut y réfléchir avant d'adopté un système de notation. Si vous décidiez de changer votre façon d'affecter des numéros de version, sachez que cela pourrait vous gêner par la suite dans vos gestions de projets, mais cela reste acceptable tant que vous restez cohérent sur les différentes versions d'un même projet. **Un changement de nomenclature ne peut donc être effectué qu'a partir d'un nouveau projet.**

Je fais apparaître ces numéros de versions au début de mes programmes. Lorsqu'il s'agit d'un projet orienté « blablas » comme le présent document, je le place dans le document « README » qui est présent dans tous mes projets (voir : Uniformiser les projets).

Nommer les fichiers et les dossiers correctement

Beaucoup d'entre vous l'ignorent, mais on ne doit pas nommer les fichiers n'importe comment. Il y a des règles de syntaxe à respecter. Là où les systèmes d'exploitation font des pièges (Windows en particulier), c'est que bien qu'on ne doive pas nommer les fichiers n'importe comment, le système ne nous l'interdit pas.

Je vous encourage à consulter le site ci-dessous pour prendre connaissance des contraintes de nommage des fichiers.

Liens_Web

— **https://bpmi.geneses.fr/3-2-nommer-dossiers-fichiers/** # Page d'explication sur le nommage des fichiers

Voici un résumé de ce qu'il faut faire ou pas
- Pas d'espace
- Pas de caractère « bizarres » ou accentués
- Ne pas utiliser des noms trop longs
- Utiliser uniquement les 26 caractères de l'alphabet (Majuscule et / ou minuscule), les chiffres de 0 à 9 et les caractères "_" et "-".

Il est important de respecter ses règles de nommage car les outils que vous allez utiliser comme : Le Raspberry Pi (ou toutes les machines linux), Github, Wikimedia (et tous le WEB en général), Sphinx, Doxygen et bien d'autre encore, respectent ces règles.

Ne pas multiplier les copies d'un projet

Une chose importante à laquelle il faut être vigilant, c'est de s'assurer que vous travaillez toujours sur la même source d'un projet (ou de la documentation).

> **Exemple** Admettons que vous ayez le dossier de votre projet en local sur votre PC sur « *D :\Mon_Super_Projet* ». Vous pourriez envisager que votre disque dur risque de tombé en panne. Vous allez alors vouloir en mettre une copie sur le réseau : « *\serveur_en_reseau\Mon_Super_Projet* ». Lorsque vous publierez votre projet, vous allez à nouveau mettre une copie de votre projet sur un dépôt distant.

Dans cet exemple, vous vous retrouvez alors avec trois versions de votre projet. Ce qui signifie que vous allez devoir maintenir ces trois versions à chaque évolution de votre projet. Cela demande beaucoup de temps et de rigueur. Cela vous ajoute autant de risque de faire des erreurs comme : oublier de modifier l'une des versions ou encore modifier la mauvaise version.

Pour éviter cette gestion difficile, voici ce que vous pouvez faire.
- **Travail en réseau** :
  - Si vous êtes en entreprise, vous ne devez travailler que sur le serveur. Aucune copie en local sur votre poste. L'administrateur réseau de votre société fait des sauvegardes du serveur, mais pas de votre poste.
  - Si vous êtes un particulier, à moins d'avoir un logiciel qui sauvegarde vos données automatiquement sur un média externe comme par exemple « Cobian Backup », Je vous déconseille la copie d'un projet sur un média externe (ou un disque réseau), surtout si se média externe et également un support de travail. Vous éviterez ainsi de travailler sur deux versions en parallèle.
    Il peut arriver que l'on n'ait pas le choix, notamment dans le cas ou vos projets sont stockés bien au chaud sur un serveur et que vous souhaitiez malgré tout travailler dessus lorsque vous serrez en voyage. Dans ce cas 2 solutions s'ouvrent à vous :
    1. Utiliser un logiciel de synchronisation des données comme RSync ou FreeFile Sync. Vous devrez être vigilant, lors des synchronisations, que les données sont bien synchronisée dans le bon sens.
    2. Utiliser une solution de stockage dans le Cloud comme *OneDrive*, *DropBox* ou *Google Drive*. Ces trois services proposent des solutions gratuites ou payantes. La principale différence entre les offres gratuites et payantes est, la plupart du temps, l'espace mis à votre disposition.

    Si vous le pouvez, privilégiez la seconde solution car le travail est effectué automatiquement en tache de fond et vous aurez en plus des possibilités de sauvegarde et de restauration.
- **Publication sur un dépôt distant** :
  La gestion publique est toujours compliquée, car par principe on publie la version stable, même si cette version n'a pas toutes les fonctionnalités. Une version stable est une version fonctionnelle et n'ayant pas de problèmes bloquants.
  L'utilisation de GIT et d'un gestionnaire de dépôt distant comme Github ou Gitlab permet de s'affranchir de cette gestion des versions puisque comme indiqué dans Versionner tout un projet on travaille avec la notion

de branche. Il faut envisager GIT comme un arbre. Avec la branche Master pour le tronc et toutes vos autres branches (Dev, gh-pages, test_fonction_truc, etc.) comme un branchage avec les différentes ramifications.
N.B : Certains outils de gestion de projet ou certains IDE permettent nativement de « pousser » vos fichiers directement sur les dépôts distants.

Commenter et Documenter son code

Liens_Web
— **https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_de_documentation** #
Page d'explication et de démonstration des générateurs de code

Que vous soyez grand débutant ou un baroudeur tout terrain en programmation, vous allez forcément vous retrouvez devant la double problématique :

« Elle fait quoi cette saleté de méthode ? ! »

et

« Comment je vais faire pour documenter tout ce Bazar ? »

Pour la première question, la réponse est qu'il faut absolument mettre des commentaires dans votre code, ça vous aidera à mieux comprendre votre programme après une pause (plus ou moins longue) dans vos travaux. Souvenez-vous du : « Pour qui et pourquoi » évoqué plus haut.

Pour la deuxième question, il faudrait pouvoir reprendre chacune des Classes et Méthodes qui compose votre code pour pouvoir en faire une description complète et donc permettre à n'importe qui (vous par exemple) d'utiliser votre programme ou l'interface de programmation de votre application (API).

La thématique de ces deux questions, et la même. Il faudrait donc trouver une solution pour que les commentaires qu'on met dans du code soit automatiquement utilisés pour générer la documentation.

Heureusement, cette solution existe sous la forme de logiciels qui interprète les commentaires de votre code pour générer une documentation souvent disponible dans plusieurs format (html, PDF, EPUB). Ces Logiciels se nomment des **Générateur de documentation** voir le lien Web ci-dessus.

Il en existe plusieurs (quasiment 1 par langage) à vous de trouver votre bonheur. Voici un échantillon :
— Sphinx : pour Python
— Javadoc : pour Java
— Doxygen : multi langage, mais principalement pour le C / C++
Les commentaires insérés dans le code doivent respecter une certaine syntaxe (différente en fonction de l'outil utiliser).

Le document généré peut être utilisé et intégré dans votre propre documentation.

Petites Astuces

1. Les générateurs de documentation étant capable de générer du html, vous pouvez faire en sorte que ces documents soient directement publier sur Github.io ou ReadTheDoc.

   Il est même possible de gérer toute la documentation d'un projet avec ce genre de programme. C'est le cas pour tous mes projets (dont le présent document) pour lesquels j'utilise **Sphinx** (Générateur de documentation), **ReStructuredText** (Langage de balisage léger) et **Github.io** (support de publication en ligne).
2. Lorsque vous modifiez votre code, pensez à modifier les commentaires également car lorsque vous reprendrez votre code, il y a de grandes chances que vous ne lisiez que les commentaires.

Ne pas négliger la sécurité

Il ne faut jamais laisser des informations personnelles dans vos documentations, dépôts ou code !

Les identifiants et mots de passe, les codes bancaires, adresses et numéros de téléphones, clefs privées de chiffrement, sont autant d'informations que vous ne devez en aucun cas diffuser.

Si dans l'un de vos codes vous avez besoin de renseigner ce genre d'informations il faut les placer dans un dossier séparer (par exemple "Creds" ou "Credential") et vous assurer que seul votre code (en local) peut accéder à ce dossier et qu'il ne sera pas pousser avec le reste du projet sur les réseaux. GIT vous permet de d'ignorer les éléments qui sont renseigner dans le fichier ".gitignore" Vous devez donc ajouter "*/Creds/*" dans ce fichier, si votre dossier se nomme "Creds", pour qu'il ne soit pas pris en compte dans la gestion du projet.

3.4.6 Demander de l'aide de temps en temps

De la même façon qu'il est difficile de mener un projet entièrement seul, il peut être intéressant de demander l'aide d'une ou plusieurs personnes pour faire une documentation.

La première chose que je vous conseille de demander, si vous trouvez une personne de bonne volonté, c'est de demander que quelqu'un relise votre doc. Vous aurez ainsi un avis objectif sur ce qui est bien, ce qui ne l'est pas et sur les choses incompréhensibles qu'il sera bon de clarifier.

La seconde chose à demander, peut être que l'on vous aide à la prise en main de certains outils qui, si on n'en a pas l'habitude, peuvent être difficile à maitriser. Ces outils sont certainement utilisés dans les Fablabs ou les Hackerspaces. N'hésitez pas à vous y rendre pour trouver de l'aide.

3.4.7 Les outils et médias de diffusions

Comme pour tous travaux, il y a toujours plusieurs façons de les réaliser. En fonction de ce que l'on veut faire il faut utiliser le bon outil. La difficulté étant de savoir quels sont les outils qui existe et ce qu'ils font.

Voici donc les quelques outils que j'utilise. Il en existe d'autre, ce sera à vous de les découvrir et de vous les approprier.

Les outils de dessin

Il est toujours intéressant de pouvoir ajouter un croquis ou une image pour permettre d'illustrer un point particulier.

- Inkscape est un logiciel de dessin vectoriel. Ce logiciel Open-source permet de créer facilement de croquis, des schémas ou des images.
  On peut trouver facilement des tutos en faisant une recherche : Inkscape tuto
- GIMP est un logiciel de traitement d'image Bitmap. Ce logiciel Open-source permet de manipuler des images de la même façon que Photoshop d'Adobe.
  De nombreux tutos sont disponible : Gimp tuto
- XnView est un logiciel Open-source de traitement d'image par lot. Il vous permet de redimensionner, de convertir ou renommer vos images par lot

Les outils de production

- Fusion 360 : Il s'agit d'un logiciel de modélisation 3D de design industriel. Bien qu'étant un produit Autodesk (donc close source), il est gratuit pour les hobbyistes, les étudiants, les enthousiastes et les entrepreneurs réalisant un chiffre d'affaire de moins de 100 K$ par ans.
  La communauté est très active et il y a une chaine dédiée sur YouTube qui propose énormément de tutos.
- Kicad : Permet d'éditer des schémas électroniques, router des PCB, générer les typons et gérer le fichier de perçage. Ce logiciel open source à une forte communauté, ce qui permet de trouver de l'aide facilement.

- Un éditeur de texte : Je préfère travailler avec des éditeurs de texte simple permettant la coloration syntaxique, la numération automatique des lignes et l'édition de code.
  J'utilise VIM par ce qu'il est disponible sur toutes les plateformes (Windows, MacOs, Linux, BSD). C'est un outil un peu compliqué à utiliser car il pour l'utiliser il faut tout apprendre depuis le début. Même une tâche aussi simple qu'un copier / coller est compliqué.
  Si vous êtes sous Windows, vous pouvez utiliser Notepad ++ qui fera très bien le boulot.
  Si vous ne faites pas de code et que vous préférez le confort d'un éditeur de texte WYSIWIG comme Word ou Libre Office, cela ne pose aucun problème. Souvenez-vous simplement que ce genre de soft peut parfois vous réserver des surprises au niveau du formatage du texte lorsque vous l'exporterez vers un WIKI.
- Logiciel de gestion de version
  - GIT : Permet une gestion de version décentralisée (qui n'a pas besoin d'un serveur) c'est l'outil ultime du développeur. Il permet de garder une version stable et de tester ou de développer en parallèle sans remettre en question le bon fonctionnement de la branche stable. GIT permettant la gestion de version de fichier texte, vous pouvez l'utiliser pour gérer tous vos projets dont les éléments de construction génèrent des fichiers texte. (txt, xml, html, python, C Plus Plus, INI, etc.
  - Tortoise : Ce programme s'interface entre GIT et vous. Il s'intègre dans votre explorateur et vous permet de gérer toutes vos manipulations GIT à partir de votre souris, ce qui représente un sérieux atout pour tous les allergiques à la ligne de commande. Il ajoute en plus un indicateur visuel à vos fichiers pour que vous puissiez identifier les éléments qui ont été modifiés récemment et qui n'ont pas encore été versionnés
  - Autres : Il existe d'autre logiciel de versioning, comme Mercurial ou SVN. A vous de les découvrir et de choisir votre outil préféré.

Les outils de traduction des textes

Que ce soit dans la préparation ou dans la réalisation de votre projet, vous allez certainement devoir consulter des sites ou des docs écris dans une autre langue. Cela peut vite être très handicapant. Voici deux outils qui permettent de traduire : un mot, un texte, un document ou un site.

- Google Translate : Il s'agit de l'outil de Google que tous le monde connait.
- Deepl : Il s'agit d'un outil de traduction qui amène plus de nuance dans les traductions et qui génères des traductions de meilleur qualité que ces concurrent.

Les langages de balisage en texte clair (Plantext Markup Language)

Les langages de balisages en texte clair sont des langages qui doivent pouvoir être interprété informatiquement tout en en restant parfaitement lisible par les humains. Cela permet d'écrire un document sans avoir à se préoccuper de l'aspect esthétique. Seule une syntaxe peu contraignante est à respecter pour que le document puisse être interpréter par l'ordinateur.

Liens_Web

- **https://fr.wikipedia.org/wiki/Langage_de_balisage_l%C3%A9ger** # Petite explication sur les langages de balisage simples
- **https://fr.wikipedia.org/wiki/LaTeX** # Page Wiki de présentation du LaTeX

- reStructuredText et Markdown : Ces langages sont utilisés nativement par Github, ReadTheDoc et GitLab. Sphinx s'appuie sur le reStrucredText.
- LaTeX : il s'agit d'un langage permettant la création de document très élaboré. Il est très utilisé par la communauté scientifique pour ses possibilités d'écriture de systèmes mathématiques. Il est également de plus en plus utilisé dans la presse pour ses possibilités de gestion des mises en page.
  Attention Il s'agit d'un outil très puissant mais très compliqué à utiliser et à mettre en œuvre.
- MediaWiki et DokuWiki : Il y a actuellement deux types de moteur WIKI. Chacune utilisant une syntaxe différente.

Les générateurs de documentation

Liens_Web
— **https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_de_documentation** # Page WIKI sur les générateurs de documentation

Les générateurs de documentation (voir Commenter et Documenter son code) permettent de récupérer les commentaires et balises que vous intégrez dans votre code pour générer une documentation. Cette documentation peut être générée sous différents formats. Pour une publication sur le WEB, c'est le format **html** qui est le plus pratique.

Il en existe dans tous les langages, par exemples :
— Sphinx (pour Python)
— Javadoc (pour Java)
— Doxygen (pour le C/C++)
Cette documentation extraite de votre code améliore la qualité de votre projet tout en vous permettant une lecture plus confortable de votre interface.

Les outils de publication des projets et de la documentation

Liens_Web
— **https://fr.wikipedia.org/wiki/Wiki** # Page de définition d'un WIKI

Il y a plusieurs moyens à votre disposition pour publier vos projets et vos documentations sans obligatoirement devoir faire appel à un éditeur ou à un service marketing
— Les Wiki (moteur MediaWiki et moteur DokuWiki) : Les Wikis, dont le plus connu est certainement Wikipédia sont des espaces de publication et d'information collaboratifs, c'est à dire que ce que vous y écrivez peut-être modifié / corrigé / complété par d'autres personnes. Ils représentent une très bonne solution pour la présentation et la documentation des projets. En particulier si le projet a été réalisé dans le cadre d'un travail universitaire, d'un fablab ou de n'importe quel espace collaboratif.
— Les gestionnaires de dépôt distant (Github, Read The Docs, GitLab) : GitHub et Gitlab permettent tous les deux de partager des projets et leur documentation. Ils offrent tous les deux la possibilité de créer une page Wiki spécifique à votre projet. Seul Github propose de générer une page web regroupant toute votre documentation. ReadTheDoc ne permet que de créer une page web. C'est un bon complément à GitLab.
— Les sites de partage de projet en mode DIY (Faites le vous-même) : Ce sont des plateformes qui permettent de présenter des projets sous la forme d'un tutoriel pas à pas. Pour vous faire une idée, vous pouvez visitez les trois exemples ci-dessous :
  — Oui Are Makers
  — hackster.io
  — Instructables

3.5 Avant, pendant et après la documentation

Avant Vous devez vous créer une structure de travail, rassembler les différentes documentations que vous avez déjà et définir les objectifs de votre projet.

Ces éléments vous serviront à la fois : à la réalisation de votre projet et à l'établissement de votre documentation.

Petites Astuces

1. La création d'une fiche projet vous permettra non seulement de définir les objectifs de votre projet mais en plus, cela vous permettra de synthétiser vos idées.

   Lorsque vous aurez défini les objectifs de votre projet, identifiez pour chacun d'entre eux les taches et actions qui le composent. Cela vous permettra d'avoir un ensemble de petits objectifs permettant de simplifier le processus global de création.

   Ces sous-objectifs peuvent être intégrés dans la fiche projet et font partie intégrante de la documentation.

Pendant

1. Essayez d'écrire votre documentation en parallèle du développement de votre projet. Puisque le travail de rédaction prend toujours beaucoup de temps, il y a de grandes chances que vous soyez souvent amené à vous interrompre en cours d'écriture. Vous devez vous laisser des informations vous signalant qu'un paragraphe n'est pas encore terminé. Pour ma part, j'entoure les éléments en cours de rédaction (ou pour lesquels une modification est à effectuer) par deux blocs "[WIP]" (Work In Progress : Travail en cours). Ces blocs sont supprimés quand la partie en cours est complète

```
Ex:
[WIP]

Un bout de doc vachement bien mais encore incomplet ...

[WIP]
```

   Cela vous permet d'avoir immédiatement une information visuelle sur ce qui est complet.

   N.B : Même si vous avez supprimé ces blocs, vous pouvez faire des modifs ou autres améliorations sur les paragraphes déjà traiter. Pensez juste à vous laisser des indices si vous effectuez une refonte du paragraphe ou si vous y ajoutez un complément conséquent.

2. Une autre petite astuce est d'insérer des penses bêtes dans vos documents pour que vous vous rappeliez que vous vouliez parler d'un truc. Je mets ces penses bêtes dans les fichiers « Bug_ToDo_lst » (qu'il faut consulter régulièrement). Cependant, dans le feu de l'action, il m'arrive de les mettre directement dans mon document. Pour les identifier, je les entoures encore une fois par 2 blocs : [TODO] (à faire)

```
Ex:
[TODO]

Parler des trucs super important que j'oublie tout le temps

[TODO]

Ou:
[TODO]

Mettre la rubrique machin avant le paragraphe bidule-truc

[TODO]
```

3. Essayez d'avoir une construction logique dans votre document. Il faut que votre document soit facile à lire. Pour cela, il faut éviter de renvoyer le lecteur vers une autre partie du texte avant de revenir sur la partie actuelle.
4. Pour vous aider à naviguer facilement dans votre documentation, insérez une **Table de matières** dans votre documentation. Il est à noter que les outils de publication WEB (WIKI, Sphinx, etc.) ajoutent automatiquement cette table des matières.

   Si votre documentation n'est pas dynamique (PDF, papier, etc.), pensez à ajouter les numéros de pages en face de chaque rubrique. Les logiciels de traitement de textes (Word, Libre Office, etc.)

sont capables de générer cette table des matières en ajustant automatiquement les numéros de pages.

Pour que les logiciels fassent une partie du travail à votre place, il faut leur préparer le travail. Dans le cas des tables des matières, il faut utiliser le balisage des titres présent dans ces logiciels.

N. B. : Pour les WIKI ou pour Sphinx, les titres sont définis par des conventions syntaxiques.

5. N'oubliez pas de vous relire (souvent) pour vous assurer qu'il n'y a pas de doublon, de non-sen, d'incohérence ou simplement de coquille dans votre doc.
6. Définissez votre projet et tenez-vous y. Il est possible voir fréquent qu'un projet évolue en cours de route. Tant que vous ne cherchez pas à le faire évoluer en permanence. Essayez d'aller au bout du projet sous la forme que vous avez défini et par la suite, faites une nouvelle version du projet qui elle aussi sera défini selon des objectifs précis (ou à peu près). C'est la seule façon de pourvoir finir un projet et cela aide à ne pas se décourager (voir à abandonner) en cours de route.

Après Relisez votre document, faites-le relire par quelqu'un d'autre et lorsque tout semble correcte, essayez d'agrémenter un peu votre document. C'est particulièrement important si votre document est exclusivement textuel. Voir *Les finitions*.

3.6 Les licences

Il est important que vous attribuiez une licence à vos travaux. La documentation ne fait pas exception.

Par défaut, si vous ne définissez pas explicitement une licence pour vos projets ou vos documentations, c'est le Copyright (appelé « Droit d'auteur » en France) qui s'applique par défaut sur vos œuvres et créations. Dans ce cas une personne tiers ne pourra consulter et utiliser vos créations que dans les conditions de mise à disposition de l'œuvre que vous aurez défini.

Si votre projet n'a pas vocation à être "libre" ou open sources, assurez-vous d'en avoir bien définit la paternité en la signant avec votre nom ou un pseudonyme (à condition que ce dernier ne soit pas lui aussi utilisé par une tierce personne).

Si vous souhaitez mettre votre création à disposition de tout le monde et permettre à quiconque de la modifier et de la redistribuer selon ses propres conditions, il faut utiliser la licence X11 (licence MIT).

Vous pouvez consulter les différentes licences depuis les liens Web ci-dessous :

— Liste des licences

Pour ma part, j'utilise systématiquement la licence :

CC BY-NC-SA. 

Cette licence définit les termes suivants :

— **BY** : l'attribution. En cas d'utilisation ou de modification, l'auteur doit être cité
— **NC** : Pas d'utilisation commerciale. Une tierce personne ne peut pas vendre tout ou partie du projet ou de la documentation qui lui est rattaché.
— **SA** : Partage dans les mêmes conditions. Même si quelqu'un modifie le projet, il ne peut pas s'attribuer la paternité du projet ni changer le type de licence.

Je place cette licence systématiquement dans les fichiers "README.RST" présent dans tous mes projets.

3.7 Les finitions

Lorsque vous aurez enfin tout fini. Vous pourrez passer aux petits détails de dernières minutes, comme la décoration ou la publication de la doc et du projet.

3.7.1 Agrémenter sa documentation

Lorsque vous aurez terminé votre documentation, vous allez certainement vous trouver face à un bloc de texte monolithique. Pour rendre votre documentation plus attractive il vous suffit d'ajouter quelques images et / ou dessins.

Vous ne pouvez pas utiliser la première photo de chatons venus car vous ne le savez peut-être pas mais la plupart des photos que l'on trouve sur internet ne sont pas libre de droits. Vous n'avez donc pas les droits de les utiliser.

Pour trouver des images libres de droits, il vous suffit de faire la recherche suivante :

```
# Recherche avec mots clefs en Français
image domaine public gratuit
# ou encore
illustration domaine public

# Recherche avec mots clefs en Anglais
public domain pictures

# Recherche de dessins vectorisés et clipart
public domain vector
```

3.7.2 Publier sa documentation et son projet

La publication de votre documentation doit être envisagée dès le début de votre projet car les outils de rédactions sont différents selon que vous souhaitiez publier sur un WIKI, sur Github ou dans une revue scientifique.

N. B. : En plus des outils, la syntaxe des langages utilisés est également très différentes.

Les pièges à éviter

Même si cela parait être une façon simple et efficace de publier sa documentation, **il ne faut jamais la publier dans un blog ou un forum**.

- **Les blogs** : Un blog doit uniquement servir de vitrine. On peut y faire un article présentant son projet avec quelques photos et des liens qui pointent vers le dépôt et vers sa documentation.
- **Les forums** : Les forums représentent un lieu de partage et d'entre aide. Les posts qu'on y met permettent de publier quelques photos de l'évolution d'un projet, de donner ou de recevoir de l'aide, de présenter le projet fini ou de mettre une jolie photo avec des liens vers le dépôt et vers sa documentation.

La raison pour laquelle il ne faut pas publier un projet ou une documentation sur un forum ou un blog est simple. Sur ces médias, seuls les derniers articles sont affichés. Il faut alors savoir exactement ce qu'on cherche pour pouvoir retrouver un article précis. De plus, il est compliqué de devoir rééditer un article chaque fois qu'on veut faire évoluer la documentation. C'est même encore pire sur les forums car on risque d'avoir une suite dissolue d'éléments qu'il sera compliqué de réunir pour avoir une vision globale du projet.

3.8 Conclusions

Vous devriez maintenant avoir suffisamment d'informations pour pouvoir faire votre propre documentation.

Je vous conseille de faire ce travail de documentation car au-delà de vous aider en cas de reprise d'un projet, le travail de documentation vous aidera à fournir un meilleur travail sur votre projet.

N'hésitez pas à vous rendre dans un FabLab (VoLAB ? !) pour trouver de l'aide vous permettant d'aller au bout de vos projets.

CHAPITRE 4

Bug et ToDo-list

4.1 Description

Dans ce fichier sont renseigner les bugs identifiés et la liste des choses à faire.

1. Bugs identifiés

   A chaque fois qu'un bug est identifié, il doit être renseigner ici si il ne fait pas l'objet d'un traitement immédiat.

2. ToDo-list

   Ici doivent être renseigner la liste des tâches à faire. Il s'agit souvent de petites choses à fort potentiel d'oubli ou des tâches qui ne peuvent pas faire l'objet d'un traitement immédiat.

4.2 Model Type

Date de saisie Date à laquelle la problématique a été identifiée

Date de traitement Date du traitement de la problématique

Cible [userDoc, modelisation, software, PCB, autre]

Statu [NONE, WIP, DONE]

Problématique Descriptif de la problématique

Traitement Descriptif du traitement de la problématique

4.3 Bugs identifiés

Date de saisie

Date de traitement

Cible [userDoc, modelisation, software, PCB, autre]

Statu

Problématique

Traitement

4.4 ToDo-list

Date de saisie 180817

Date de traitement 180825

Cible userDoc

Statu Done

Problématique Dans la partie comment, ne pas oublier de parler du versioning et de l'horodatage

Traitement Traité dans la partie "Un peu d'organisation"

Date de saisie 180825

Date de traitement 180904

Cible userDoc

Statu Done

Problématique Dans finissions, parler des banques d'image libre de droits

Traitement Traité dans la sous rubrique : « Agrémenter sa documentation »

Date de saisie 180906

Date de traitement 180912

Cible userDoc

Statu Done

Problématique Dans la partie Comment, parler des problèmes de maintenance et de MAJ de la difficulté de gestion des source multiples

Traitement La rubrique : « Ne pas multiplier les copies d'un projet » a été créer spécialement pour cette problématique.

Date de saisie 180906

Date de traitement 180919

Cible userDoc

Statu Done

Problématique Dans Finissions / Publication, parler des problèmes liés à la publication sur les blogs ou les forums

Traitement La sous rubrique « Les pièges à éviter » a été créer dans Finissions / Publications.

Date de saisie 180912

Date de traitement 180914

Cible userDoc

Statu Done

Problématique Dans versioning, parler des numéros de versions (Doc et Projet)

Traitement Sous rubrique : « Version et révision » créer dans « Versionner tout un projet »

Date de saisie 180913

Date de traitement 180922

Cible userDoc

Statu DONE

Problématique Commenter et Documenter son code : Cette section est à intégrer après « Ne pas multiplier les copies d'un projet » et avant « Ne pas négliger la sécurité ».

Traitement Nouvelle section créer.

Date de saisie 180915

Date de traitement 180918

Cible userDoc

Statu Done

Problématique Dans « Uniformiser les projets », présenter l'arborescence complète d'un dossier projet.

Traitement L'arborescence a été compléter et une description de chacun des nouveaux éléments a été ajouter.

Date de saisie 180918

Date de traitement 180919

Cible userDoc

Statu Done

Problématique Dans README.rst, ajouter une section « Comment utiliser ce document »

Traitement Nouvelle rubrique ajoutée.

CHAPITRE 5

VoLAB

./Images/volab/logoVoLAB.jpg

5.1 Nous connaître

Le Site Web http://www.volab.org

Le Wiki http://www.vorobotics.com/wiki/index.php?title=Accueil

GitHub https://github.com/volab

Twitter https://twitter.com/vorobotics

Faceboock https://www.facebook.com/VoLab95/

5.2 Qui sommes nous ?

Le VoLAB, premier FabLab du Val d'Oise (depuis 2013), est un FabLab associatif portée par l'associassion VoRoBoTics situé Vauréal (95480).

Nous avons pour vocation le partage non marchand des connaissances et l'échange de compétences. Petits et gros projets se côtoient dans divers domaines comme :

Le travail du bois, du métal, l'électronique, la programmation, la sérigraphie, la couture, le scrapbooking et bien d'autres encore.

Les échanges dynamiques dans la bonne humeur et le respect mutuel permettent à chacun de partager et d'apprendre à son rythme

N'hésitez pas venir nous rendre visite.