



PROJET PREMIÈRE ANNÉE

Laurent Fiack – laurent.fiack@ensea.fr
Bureau D212

Sommaire

1 Organisation

- Objectifs
- Évaluation
- Planning

2 Le protocole MIDI

- Pourquoi le MIDI ?
- La couche physique
- Les messages MIDI

3 Les sujets

- La veste lumineuse
- Le pédalier MIDI

4 Travail collaboratif avec Git

- Versioning, pourquoi ?
- Git quick start guide
- Github quick start guide
- Premier exemple
- Branch

5 Aujourd'hui

Organisation

Objectifs généraux

- Concevoir un prototype
- Doit contenir un microcontrôleur STM32
- Conception d'un PCB
 - Schéma documenté
 - Placement
 - Routage
 - Sérigraphie propre
- Écriture d'un Firmware
 - Code réutilisable
 - Documenté
- Gestion de projet sur github

Évaluation

- Revue de schéma
- Revue de routage
- Démo finale la dernière séance
- Qualité du code
- Évaluation du dépôt git
 - Organisation générale
 - README
 - Régularité des commits (individualisé!)

Évaluation du hardware

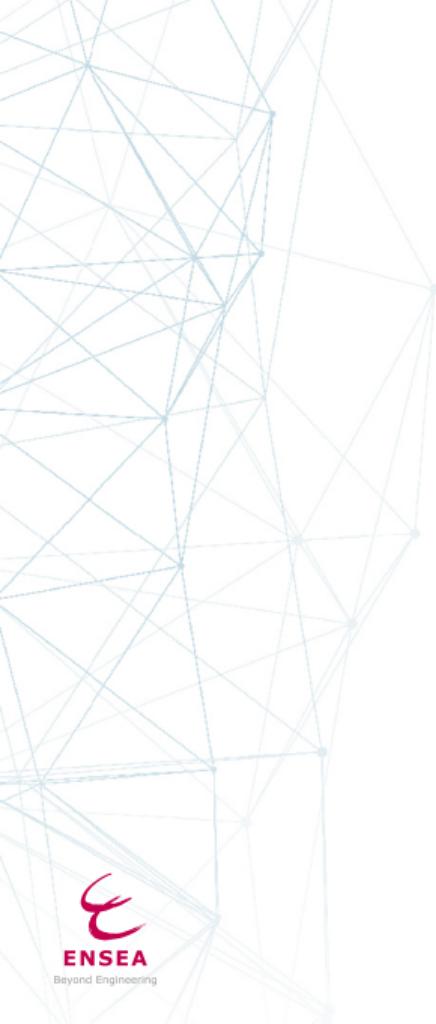
- **Organisation** : Noms de fichiers, le projet s'ouvre correctement...
- **Schéma** : Erreurs, lisibilité, commentaires, noms/refs composants, noms de signaux
- **Placement** : Capas, oscillateurs, connecteurs, HMI, test points
- **Routage** : Erreurs, vias, taille pistes, CEM (plans de masse...)
- **Finitions** : Sérigraphie, logo, visu 3D

Évaluation du firmware

- **Organisation** : Dossiers, noms de fichiers, signaux CubeMX...
- **Drivers** : Structures, noms variables, lisibilité, portabilité, documentation
- **Tests** : Lisibilité, documentation
- **Intégration** : Programme final, organisation du main...

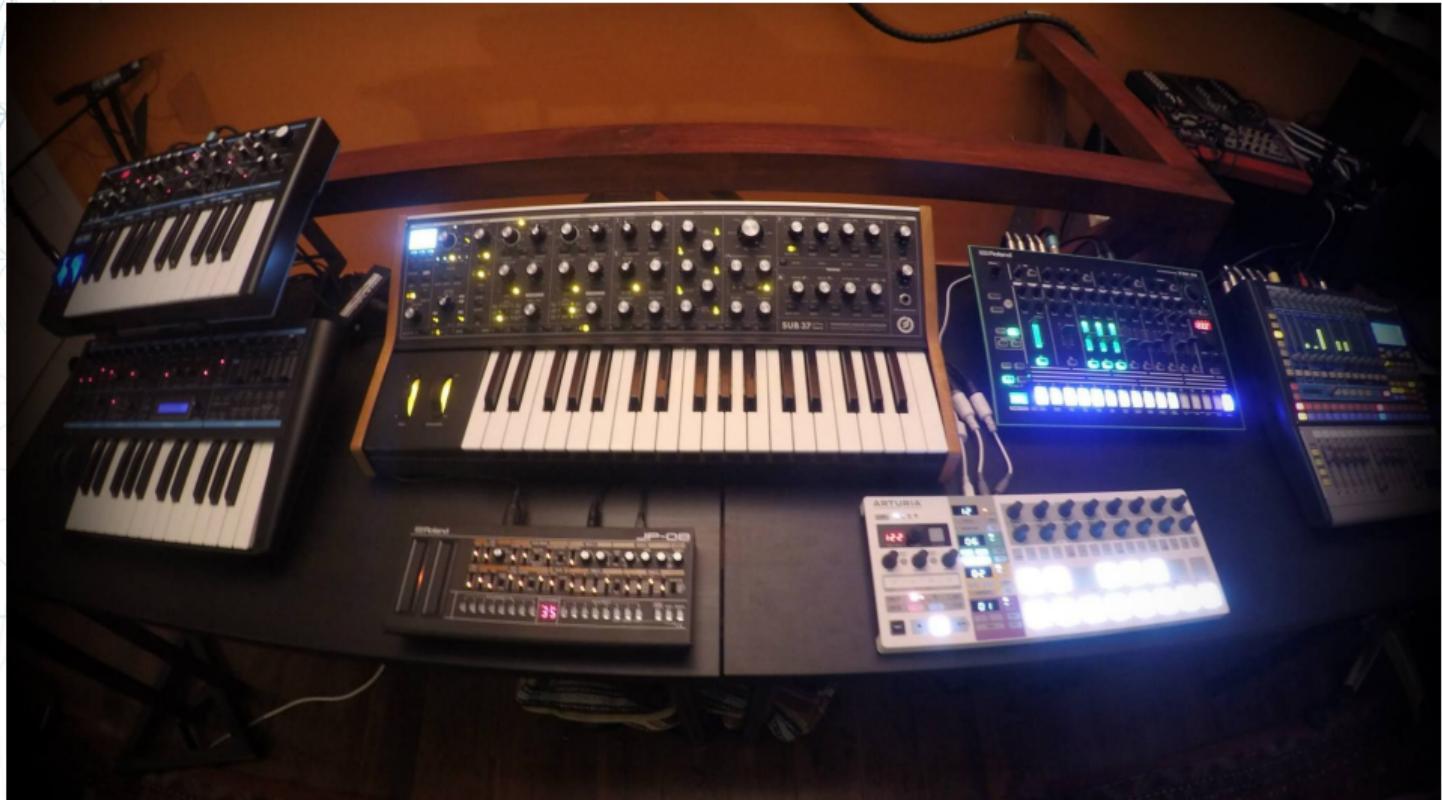
Planning

- **40 heures = 10 séances de 4h (c'est peu!)**
 - 1 **10/02** (prise en main projet/git/composants...)
(vacances février)
 - 2 **03/03** (schéma)
 - 3 **10/03** (revue + schéma)
(2 semaines sans projet)
 - 4 **31/03** (placement/routage)
 - 5 **07/04** (revue + routage, fichiers de fab=deadline stricte)
(vacances avril)
 - 6 **28/04** (soudure)
 - 7 **05/05** (firmware)
 - 8 **12/05** (firmware)
 - 9 **19/05** (firmware)
 - 10 **26/05** (2h finitions / 2h démo)

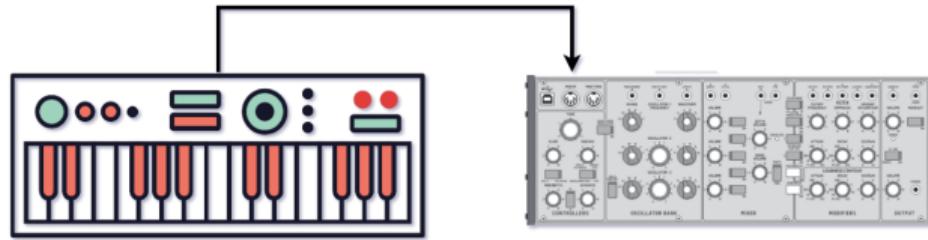


Le protocole MIDI

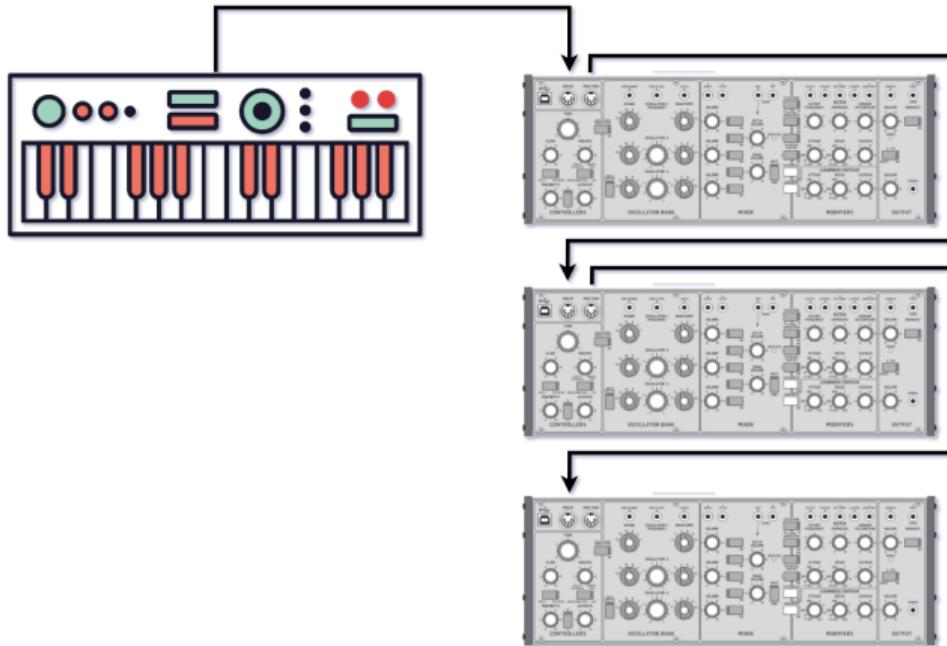
Connecter des trucs musicaux entre eux



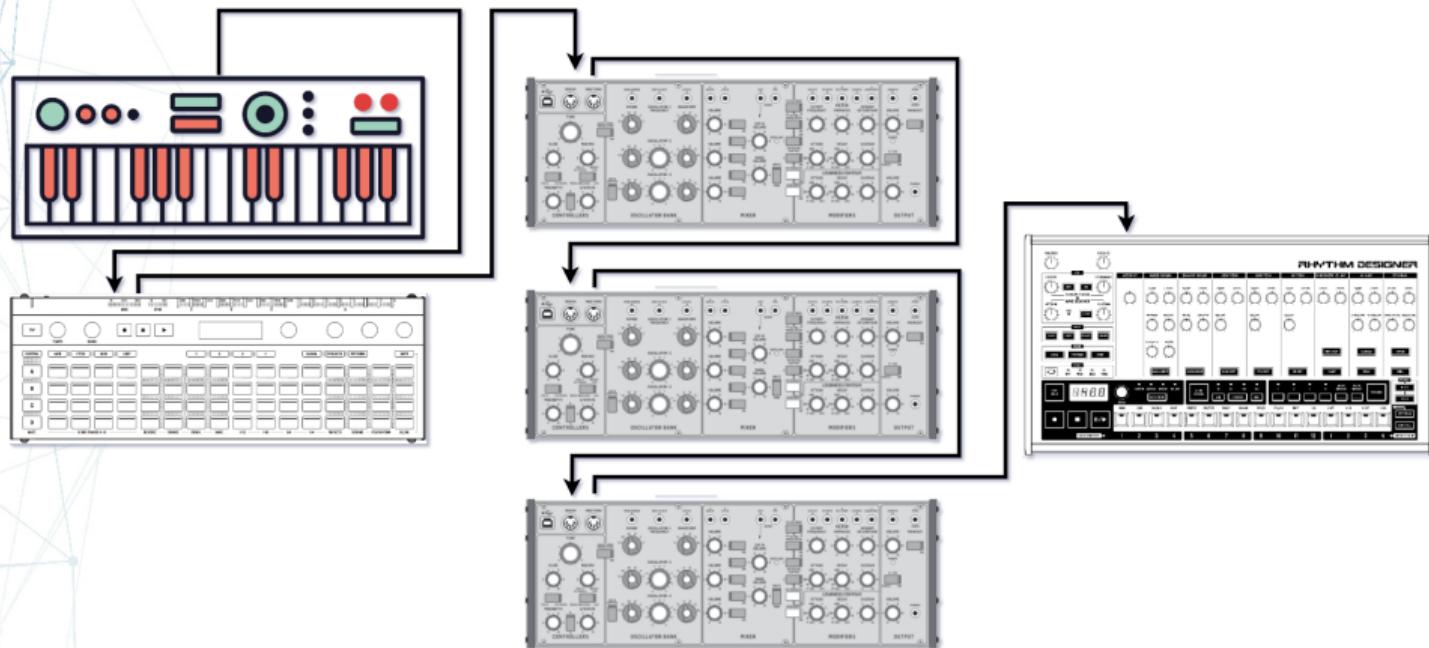
Clavier maître → synthétiseur



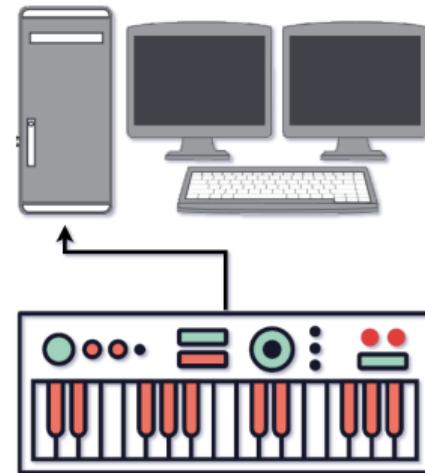
Plusieurs synthétiseurs



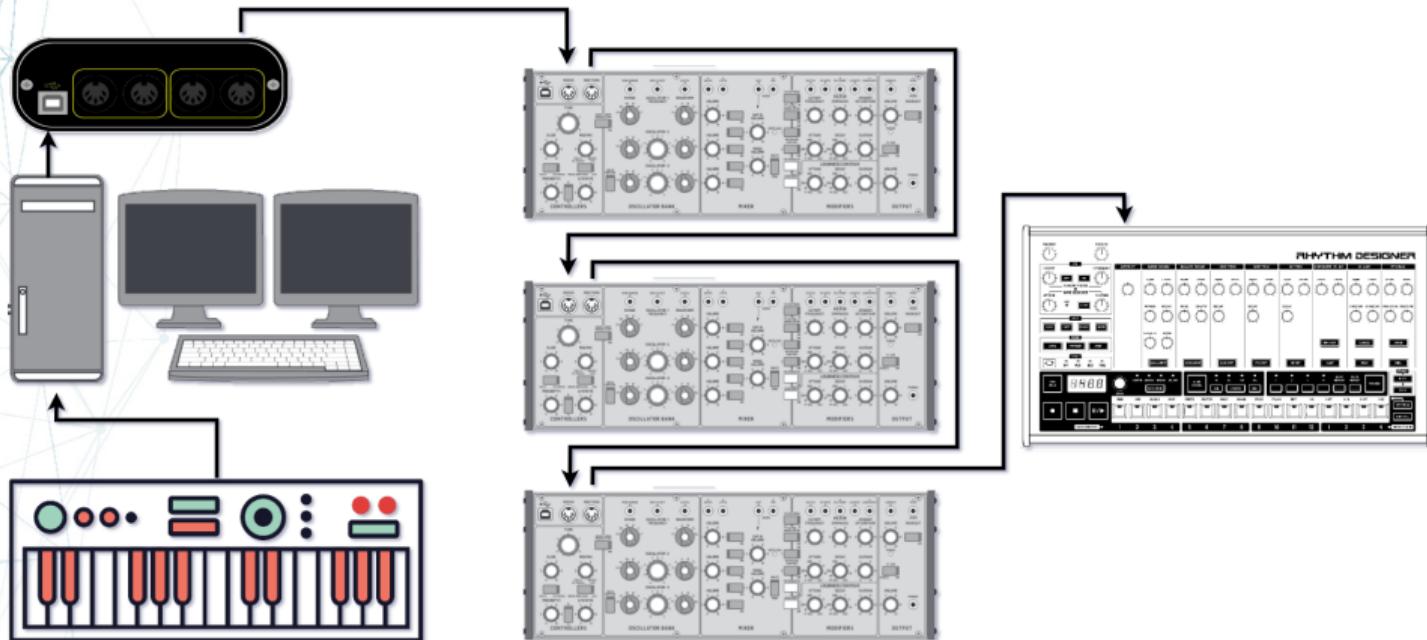
Séquenceur



Avec un PC



Studio

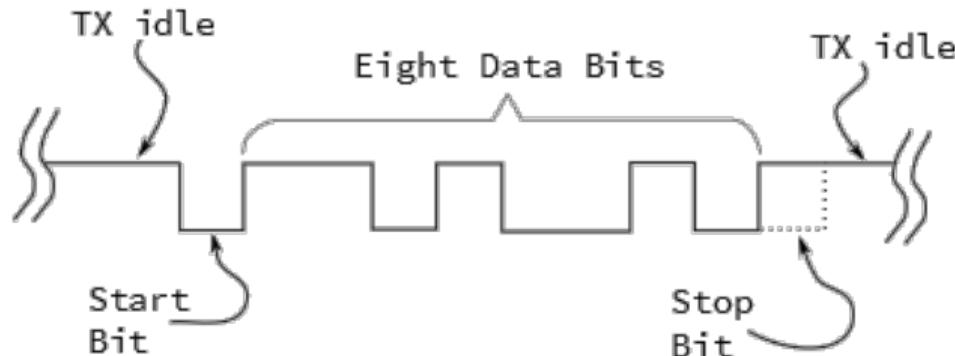


Historique

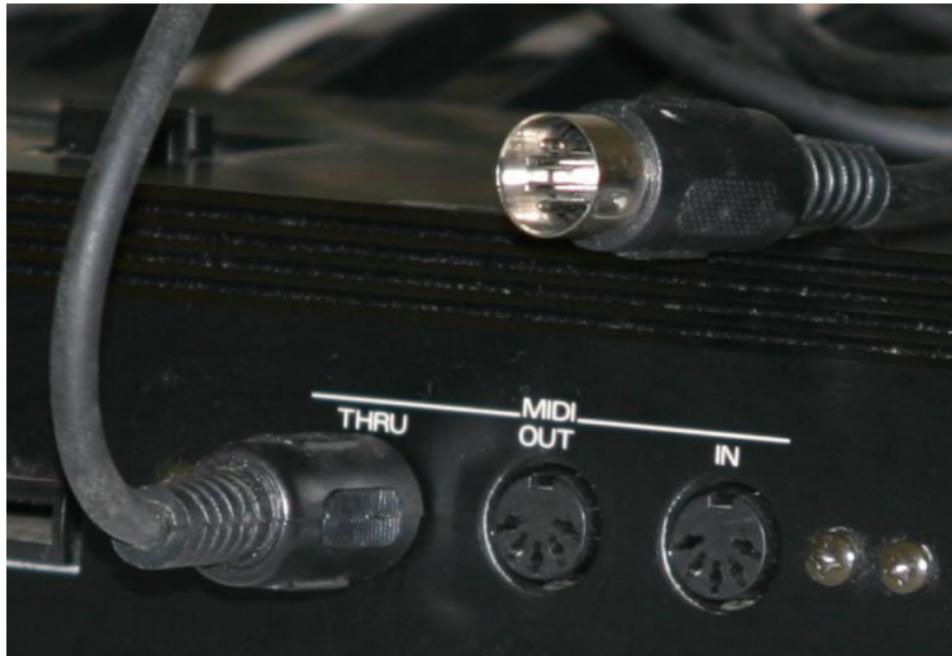
- Première spec en 1981 par Sequential Circuits et Roland
 - Rapidement accepté par les Japonais Yamaha, KORG, Kawai et les Americains Moog
- Standardisé par la MIDI Manufacturers Association (MMA) en 1983
- Connection entre Prophet 600 et Roland JP-6 en 83
- En 83 encore, TR-909 et séquenceur MSQ-700
- 1991 : format de fichier MIDI
- 1999 : USB Midi class compliant (USB 1.0 en 1996)
- 2020 : MIDI 2.0 bidir compatible MIDI 1.0

Protocole bas niveau

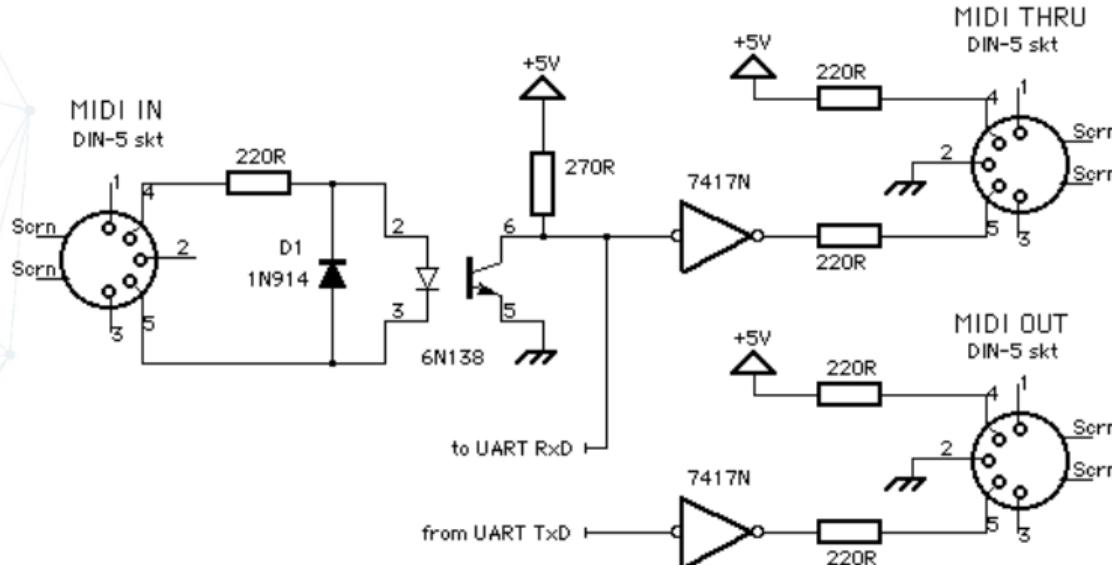
- Liaison Série Asynchrone (= UART)
- 31250 bits/seconde, 1 bit start, 1 bit stop, pas de parité
- Message MIDI = 1 octet de status (+ éventuellement data)
 - Octet Status : MSB = 1
 - Octet Data : MSB = 0
- Message classique = 3 octets : 1 Status, 2 Data
 - Environ 1 ms



Connectique : DIN 5 broches

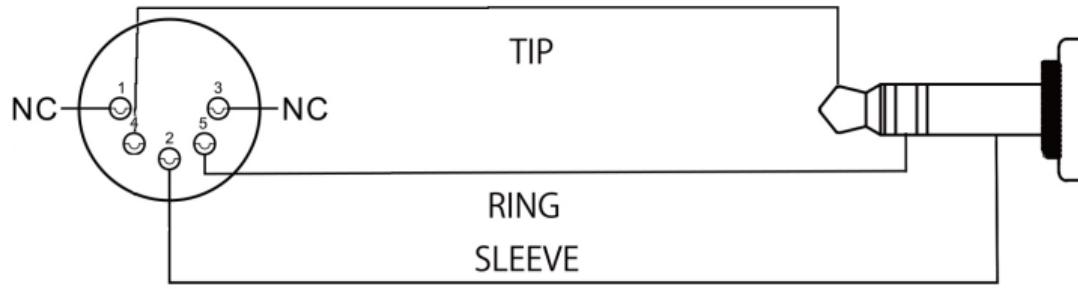


Câblage



- Utilisation d'un optocoupleur
- Évite les boucles de masse

Connectique alternative : Jack TRS 3.5mm



Tip —— Pin 4

Ring —— Pin 5

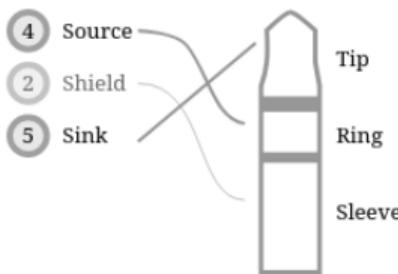
Sleeve —— Pin 2

- Standard recommande 2.5mm pour éviter la confusion avec un connecteur audio
- Assez peu respecté

Connectique alternative : Jack TRS 3.5mm

Type A (MIDI standard)

Korg, Make Noise



MIDI 4 (Source) > TRS RING

MIDI 2 (Shield) > TRS SLEEVE

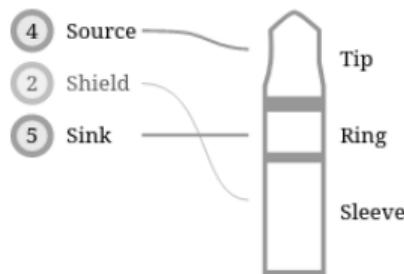
MIDI 5 (Sink) > TRS TIP

Used by manufacturers like Korg, Make Noise, Akai, and IK Multimedia (iRig MIDI 1). In 2018, it became the [MIDI standard](#)

The first known use of MIDI over TRS (the Line 6 MIDI Mobilizer in 2010) connected this way.

Type B

Arturia, Novation, 1010music



MIDI 4 (Source) > TRS TIP

MIDI 2 (Shield) > TRS SLEEVE

MIDI 5 (Sink) > TRS RING

Used by manufacturers like Arturia (BeatStep Pro), Novation (Launchpad Pro), and 1010music.

■ Deux standards

Messages MIDI

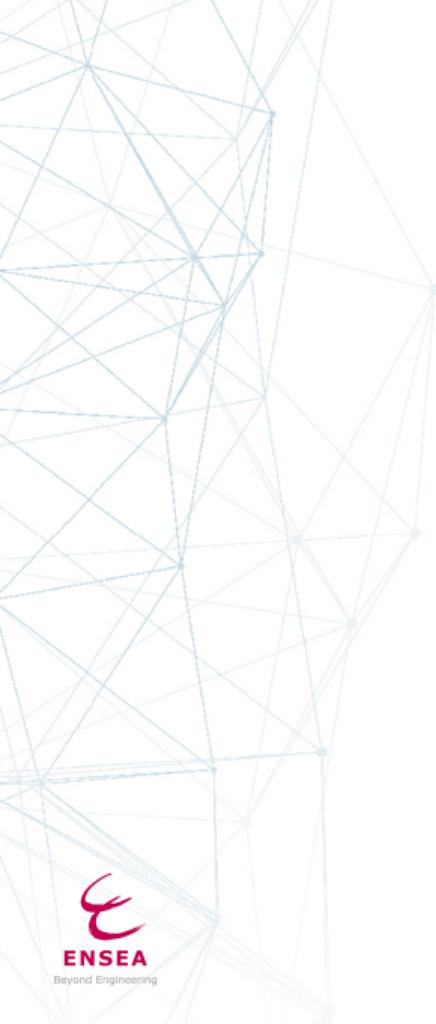
- 16 canaux

Message	Status	Data 1	Data 2
Note OFF	1000 CCCC	Note number	Velocity
Note ON	1001 CCCC	Note number	Velocity
Polyphonic Aftertouch	1010 CCCC	Note number	Pressure
Control Change (CC)	1011 CCCC	Control number (0-119)	Value (0-127)
Program Change	1100 CCCC	Program number	N/A
Channel Pressure	1101 CCCC	Pressure	N/A
Pitch Bend	1110 CCCC	LSB	MSB
System	1111 XXXX	Variable	Variable

- Ex: Jouer un C3 (60, 0x3C) sur le channel 2
 - 0x92 0x3C 0x7F
 - 0x82 0x3C 0x7F

Notes number

Octave	Note Names / Numbers (C3=60)											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-2	0	1	2	3	4	5	6	7	8	9	10	11
-1	12	13	14	15	16	17	18	19	20	21	22	23
0	24	25	26	27	28	29	30	31	32	33	34	35
1	36	37	38	39	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55	56	57	58	59
3	60	61	62	63	64	65	66	67	68	69	70	71
4	72	73	74	75	76	77	78	79	80	81	82	83
5	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107
7	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127	—	—	—	—



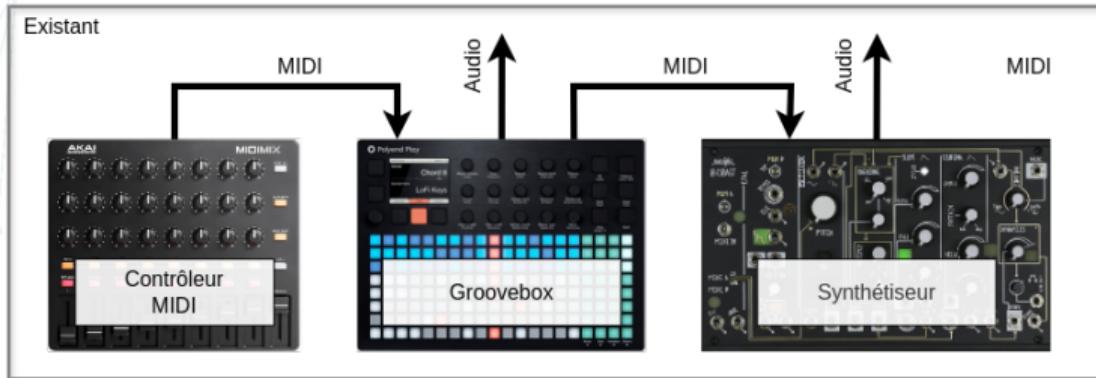
Les sujets

La veste lumineuse

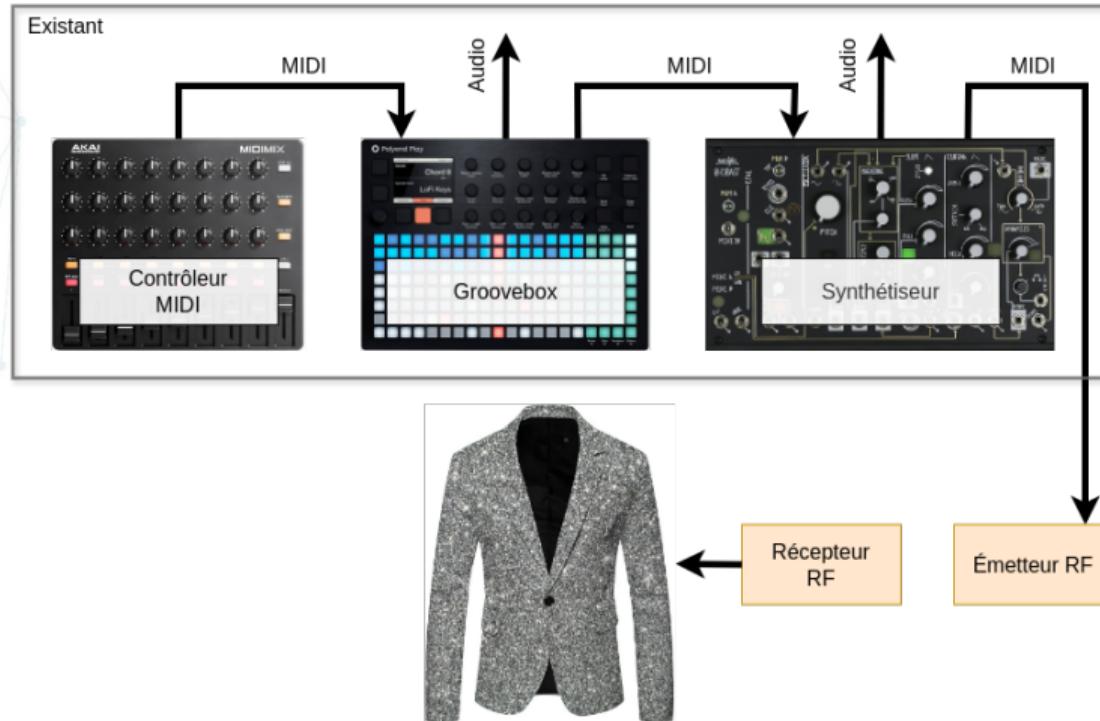
- **Objectif :** Veste lumineuse qui réagit à la musique



Détails : Mon setup actuel



Détails : Après votre projet



Organisation

- 6 étudiants (+2 chez N. Simond)
 - 3 sous-projets à faire en binôme
- 1 LED addressable NeoPixel WS2812B ; Programmer des patterns.
- 2 Communication MIDI sans-fil
- 3 Batterie rechargeable USB-C

Ressources wireless

- KiCad 7 STM32 Bluetooth Hardware Design (1/2 Schematic) - Phil's Lab #127
- KiCad 7 STM32 Bluetooth Hardware Design (2/2 PCB) - Phil's Lab #128
- STM32 Bluetooth Firmware Tutorial (Bring-Up) - Phil's Lab #129

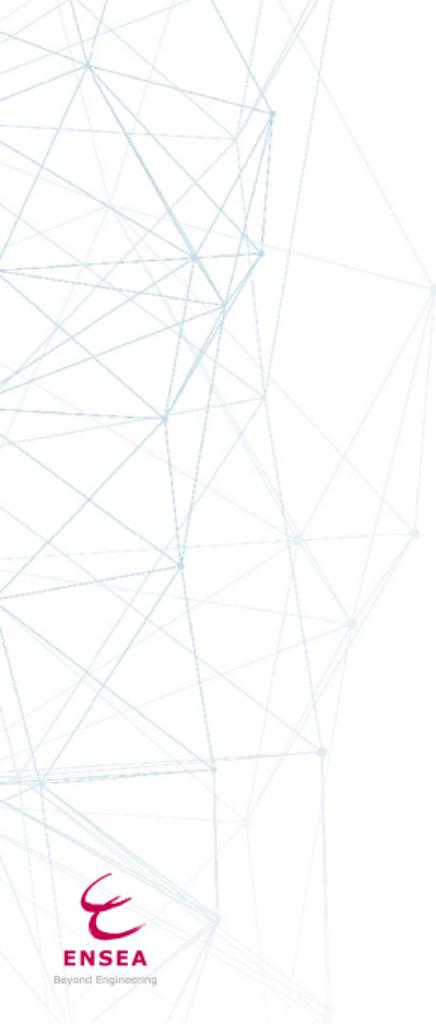
Le pédalier MIDI

- **Objectif :** Pédalier qui envoie en MIDI les messages suivants : PC et CC (pédale d'expression)
- **Détails :** Démonter un vieux multi-effet, redesigner toute l'électronique



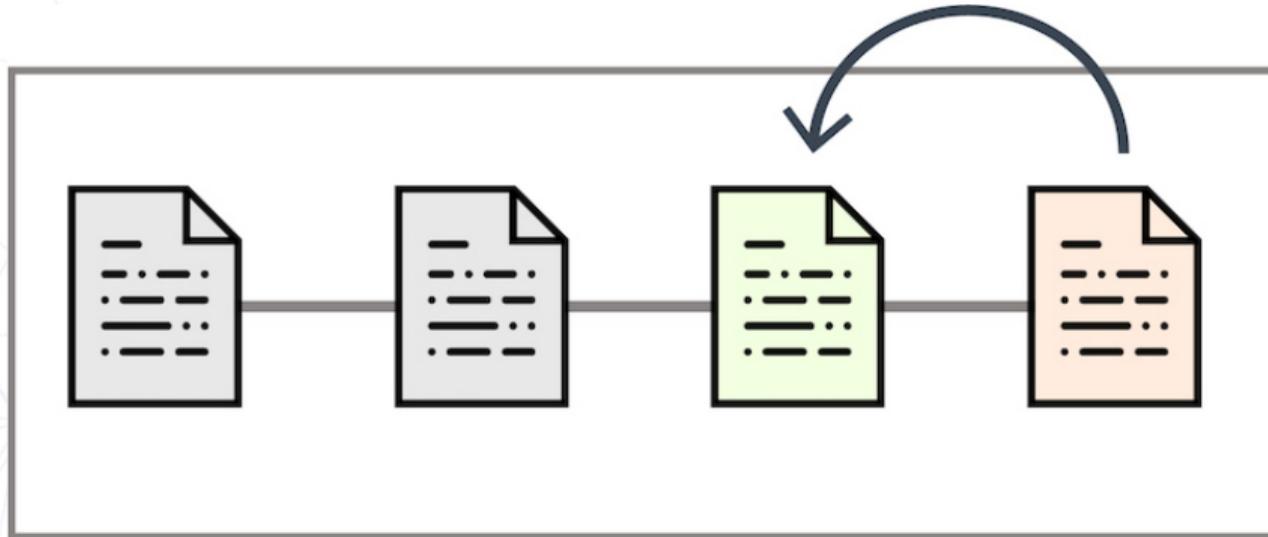
Organisation

- 4 étudiants
 - 2 sous-projets à faire en binome
- 1 Électronique et firmware
- 2 Batterie rechargeable USB-C
- Possibilité de dupliquer MIDI sans-fil



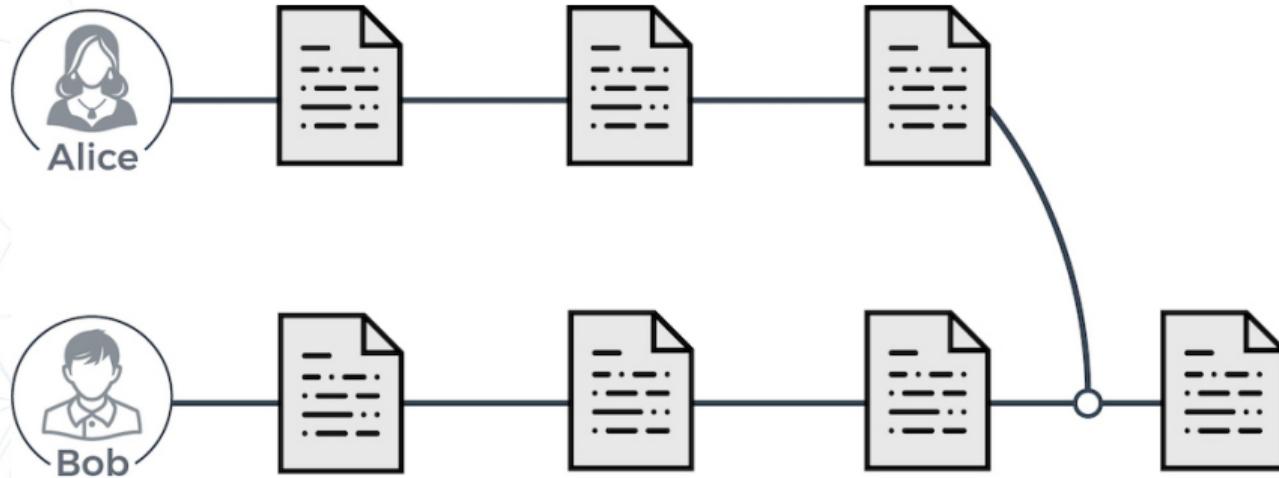
Travail collaboratif avec Git

Le droit à l'erreur



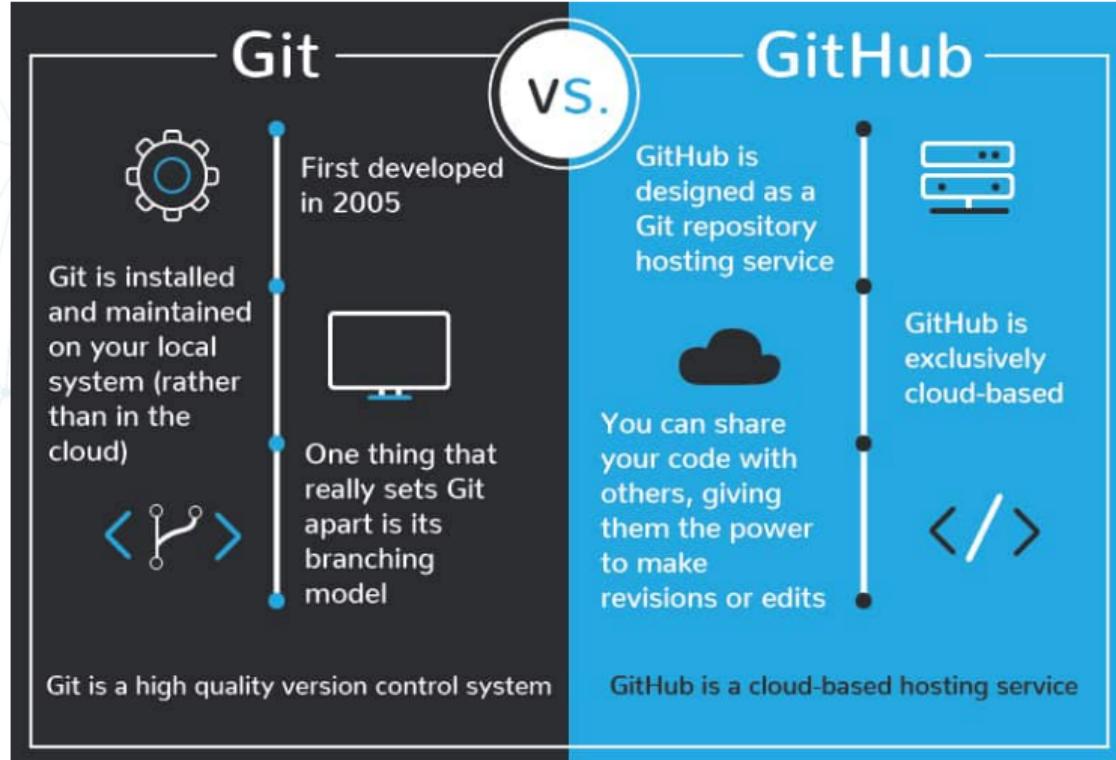
Retour à une version précédente de vos codes.

Working together

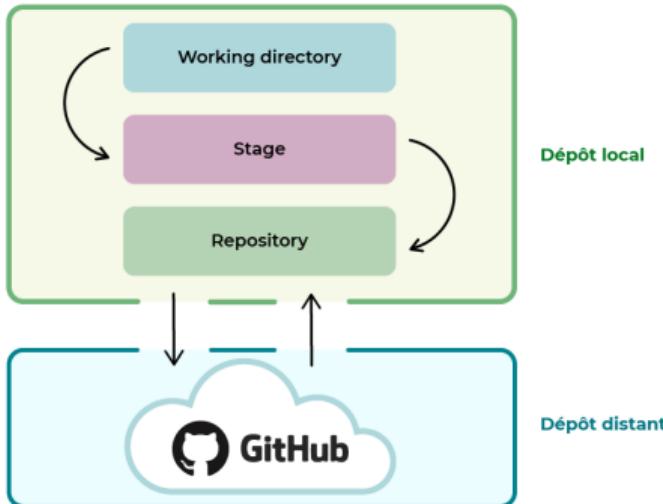


Travail sur le même projet sans gérer le travail de l'autre.

Git & Github : differences



Working flow



- Le répertoire de travail : Cette zone correspond au dossier du projet sur votre ordinateur.
- Stage ou Index : Cette zone est un intermédiaire entre le répertoire de travail et le dépôt. Il représente tous les fichiers modifiés que vous souhaitez voir apparaître dans votre prochaine version de code.
- Le Dépôt : Lorsque vous créez de nouvelles versions d'un projet, c'est dans cette zone qu'elles sont stockées.

Installer git

- Windows : <https://gitforwindows.org/>
- Linux : sudo apt install git
- Mac OS : brew install git

Git Configuration

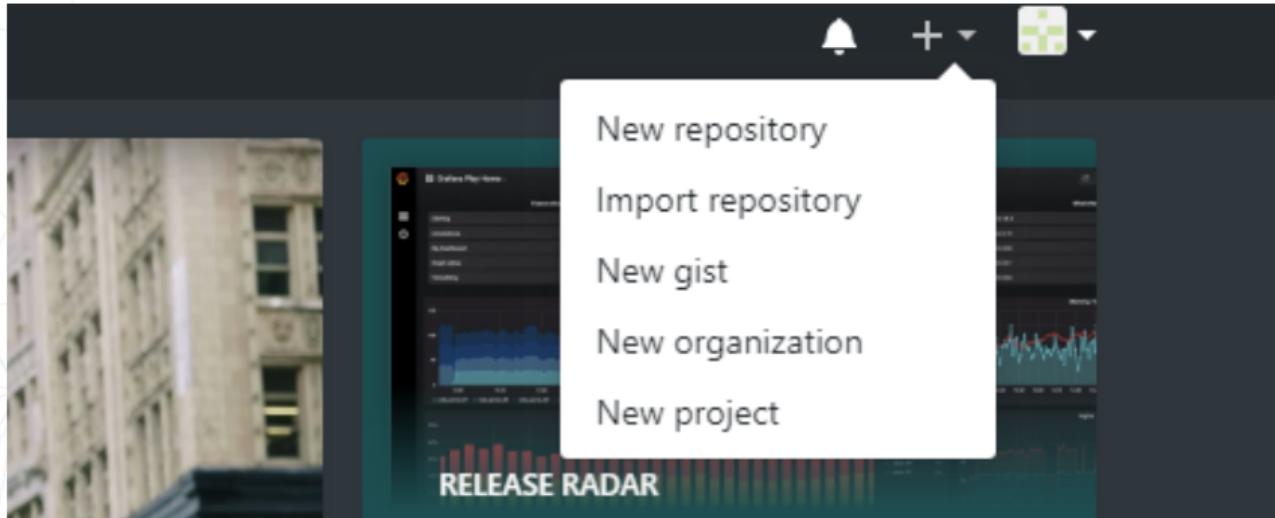
Name and email configuration :

```
git config --global user.name "John Doe"  
git config --global user.email johndoe@example.com  
git config --list
```

Create account on github

The screenshot shows the GitHub homepage with a dark blue background. At the top, there is a navigation bar with links for "Why GitHub?", "Team", "Enterprise", "Explore", "Marketplace", and "Pricing". On the right side of the header are "Search GitHub", "Sign in", and a red "Sign up" button. The main visual features a large, glowing blue globe with a network of pink lines representing global connections. In the bottom right corner, there is a small, colorful cartoon character wearing a space helmet, standing on a small patch of green grass. The central text on the page reads "Where the world builds software" in large white letters, followed by a subtext: "Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world." Below this text are two buttons: a white "Email address" input field and a green "Sign up for GitHub" button. At the bottom of the page, there are four statistics: "65+ million Developers", "3+ million Organizations", "200+ million Repositories", and "72% Fortune 50".

First repository



Public ou Privé

Dépôt public :

- Accessible à tous sur Internet.
- Excellent choix pour les projets open source, car il encourage la collaboration et la contribution d'une communauté mondiale.
- Les référentiels publics permettent une plus grande visibilité, permettant aux collaborateurs potentiels de trouver plus facilement et de s'engager dans votre travail.

Dépôt privé

- Limité à un groupe sélectionné d'individus.
- Pour les projets qui impliquent du code propriétaire, des données confidentielles ou des éléments sensibles.
- Les référentiels privés offrent un environnement sécurisé pour la collaboration au sein d'une équipe de confiance tout en gardant le projet caché aux yeux du public.
- Nécessite un abonnement GitHub payant, sauf si vous êtes étudiant.

Les fichiers importants

- README.md : Indique les informations clés de votre projet : description, environnement à utiliser, dépendances possibles et copyright. Il est affiché directement sur la page web de github.
- .gitignore : Il s'agit d'un fichier qui vous permet d'ignorer certains fichiers de votre projet Git, très utile lorsque vous ne voulez pas synchroniser des fichiers temporaires. Il existe des templates pour chaque type de projet en fonction des langages utilisés.

Create local repository

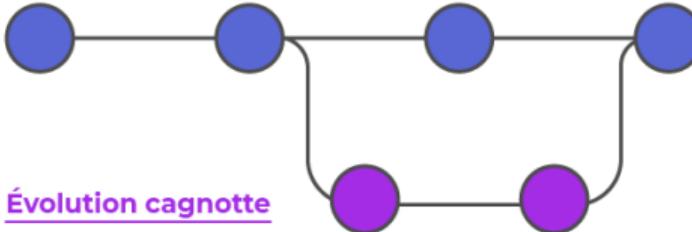
```
cd Documents/PremierProjet  
git init
```

Index your files

```
git add index.html styles.css
git commit -m "Ajout des fichiers html et css de base"
git remote add origin https://github.com/<user>/<repos>.git
git branch -M main
git push -u origin main
```

Branch

Application en production



Évolution cagnotte

Créer une nouvelle branche

```
git branch
```

```
git branch myNewBranch
```

```
git checkout myNewBranch
```

```
git commit -m "some important comments"
```

Supprimer une branche

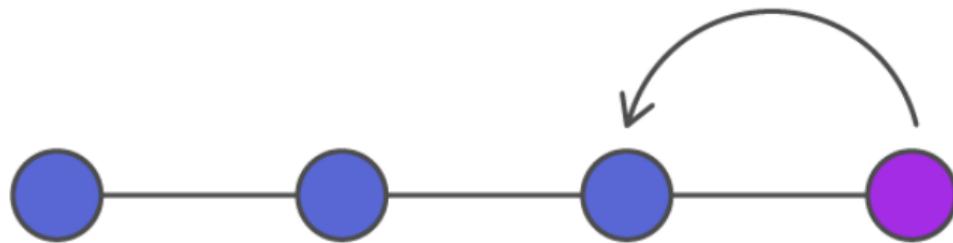
```
git branch -d myOldBranch
```

Merge

Fusionner myNewBranch into the main branch

```
git checkout main  
git merge myNewBranch
```

Reset



Reset

Revenir à une version spécifique :

- Syntaxe : `git reset [--hard] <commit>`
- Retour à '3050fc0' tout en gardant les changements dans le répertoire de travail :
`git reset 3050fc0`
- Retour à 'c0d30f3' et suppression des changements :
`git reset --hard c0d30f3`

Reset HEAD

Retour n version en arrière

- Syntaxe : `git reset [--hard] HEAD~<n>`
- Retour de 5 commits en arrière tout en gardant les changements dans le répertoire de travail :
`git reset HEAD~5`
- Retour de 3 commits en arrière et suppression des changements :
`git reset --hard HEAD~3`

Clone

- Cloner un répertoire existant :

```
git clone https://github.com/<user>/<repos>.git
```

- Mettre à jour un repertoire déjà cloné :

```
git pull [origin main]
```

Ce que vous devez retenir : Git puis Github

Répertoire local créé en premier :

```
git init  
git remote add origin https://github.com/<user>/<repos>.git
```

<<<coding>>>

```
git add .  
git commit -m "important comment"  
git push
```

Ce que vous devez retenir : Github puis Git

Répertoire github créé en premier :

```
git clone https://github.com/<user>/<repos>.git
```

```
<<<coding>>>
```

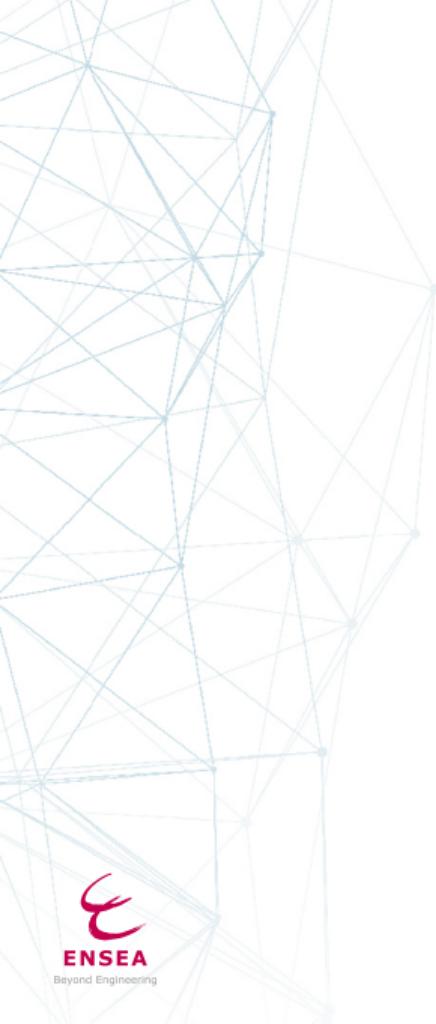
```
git add .
```

```
git commit -m "important comment"
```

```
git push
```

Pour aller plus loin : clé SSH

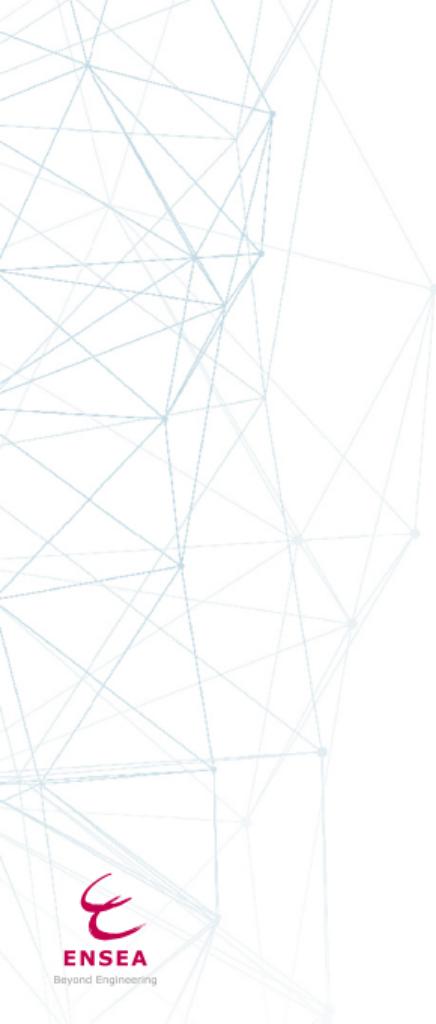
- Authentification par clé SSH :
 - Plus sécurisé,
 - Possible depuis un serveur distant,
 - Authentification par id/password non valable avec github,
 - Création obligatoire d'un token
- Procédure :
 - Génération de la clé ssh :
`ssh-keygen`
 - Récupération de la clé publique :
`cat ~/.ssh/id_rsa.pub`
 - Copie de la clé publique sur github :
 - Settings
 - SSH and GPG keys
 - New SSH key
- Changement :
 - URL de la forme `git@github.com:<user>/<repo>.git`



Aujourd'hui

Aujourd'hui

- Choix du projet, formation des groupes
- Création du dépôt git
 - M'ajouter au projet git : laurent.fiack@gmail.com
- Liste de fonctionnalité
- Schéma architectural
- Planning prévisionnel
- Choix des composants



Au boulot !