Module 1 : Introduction et Concepts Fondamentaux

Cours Git et GitHub pour Ingénieurs en Électronique

Durée : 60 minutes

Slide 1 : Bienvenue 👋

Formation Git et GitHub

Pour Ingénieurs en Électronique

- Durée : 4 heures
- Format : Théorie + Pratique
- Objectif : Maîtriser le contrôle de version pour vos projets

Slide 2 : Vos formateurs 🎓

Présentations

Nicolas et Laurent

- Ingénieurs passionnés par le développement
- Expérience en gestion de projets.
- Utilisateurs quotidiens de Git et GitHub

Slide 3 : Pourquoi ce cours ? 🤔

Contexte pour les ingénieurs en électronique

Vos projets incluent :

- Code firmware (C, C++, Python)
- Scripts de test et d'automatisation
- Documentation technique
- Schémas et fichiers de conception
- Collaboration en équipe

Problèmes courants :

- "Ça marchait hier, qu'est-ce qui a changé ?"
- Versions multiples : `projet_v1.ino`, `projet_v2_final.ino`, `projet_v2_final_VRAIMENT.ino`
- Difficultés de collaboration
- Perte de code ou de modifications

Slide 4 : Qu'est-ce que Git ? 📚

Définition

Git est un système de contrôle de version distribué (DVCS - Distributed Version Control System)

Créé par : Linus Torvalds en 2005
Objectif : Gérer le développement du noyau Linux

Caractéristiques principales

- ✅ **Distribué** : Chaque développeur a une copie complète de l'historique
- ✅ **Rapide** : Opérations locales ultra-rapides
- ✅ **Flexible** : Supporte différents workflows
- ✅ **Fiable** : Intégrité des données garantie

Slide 5 : Contrôle de Version : Pourquoi ? 🎯

Avantages du contrôle de version

1. **Historique complet**
   - Qui a modifié quoi et quand
   - Pourquoi les modifications ont été faites
   - Retour à n'importe quelle version

2. **Collaboration efficace**
   - Travail simultané sur le même projet
   - Fusion automatique des modifications
   - Résolution de conflits structurée

3. **Expérimentation sans risque**
   - Branches pour tester de nouvelles idées
   - Retour arrière facile si problème
   - Isolation des fonctionnalités

4. **Sauvegarde et sécurité**
   - Copies multiples du code
   - Protection contre la perte de données
   - Traçabilité complète

Slide 6 : Git vs Autres Systèmes 📊

Comparaison

Caractéristique	Git	SVN	Dropbox/Drive
Distribué	✅	❌	⚠️
Travail hors ligne	✅	❌	⚠️
Branches légères	✅	❌	❌
Historique complet	✅	✅	❌
Résolution conflits	✅	⚠️	❌
Performance	⚡	🐌	🐌

Slide 7 : Git pour l'Électronique 🔌

Applications spécifiques

Projets typiques :

- Firmware pour microcontrôleurs (Arduino, ESP32, STM32)
- Scripts Python pour acquisition de données
- Code VHDL/Verilog pour FPGA
- Documentation technique (Markdown, LaTeX)
- Fichiers de configuration

Avantages pour l'électronique :

- Suivi des versions de firmware
- Collaboration sur projets complexes
- Documentation intégrée au code
- Gestion des releases et versions stables
- Intégration avec outils de CI/CD

Slide 8 : Git et GitHub : Quelle différence ? 🤷

Distinction importante

Git 🔧

- Logiciel de contrôle de version
- Fonctionne en local sur votre ordinateur
- Ligne de commande ou interface graphique
- Gratuit et open source

GitHub 🌐

- Plateforme web d'hébergement de dépôts Git
- Collaboration et partage
- Fonctionnalités sociales (issues, pull requests)
- Gratuit pour projets publics et privés

Alternatives à GitHub :

- GitLab
- Bitbucket
- Gitea (auto-hébergé)

Slide 9 : Concepts Clés - Le Dépôt 📦

Repository (Dépôt)

Définition : Un dépôt est un dossier contenant votre projet et tout son historique

```
mon-projet/
├── .git/              ← Dossier caché contenant l'historique
├── src/
│   ├── main.cpp
│   └── config.h
├── README.md
└── .gitignore
```

Types de dépôts :

- **Local** : Sur votre ordinateur
- **Remote** : Sur un serveur (GitHub, GitLab, etc.)

Slide 10 : Concepts Clés - Le Commit 💾

Commit (Validation)

Définition : Un commit est un instantané de votre projet à un moment donné

Anatomie d'un commit :

```
commit a3f5b2c8d1e9f7a6b4c2d8e1f3a5b7c9d2e4f6a8
Author: Jean Dupont <jean.dupont@example.com>
Date:   Thu Jan 16 14:30:00 2025 +0100

    Ajout du support du capteur DHT22

    - Implémentation de la lecture température/humidité
    - Ajout de la gestion d'erreurs
    - Mise à jour de la documentation
```

Composants :

- Hash unique (identifiant)
- Auteur et date
- Message descriptif
- Modifications apportées

Slide 11 : Concepts Clés - Les Branches 🌳

Branches

Définition : Une branche est une ligne de développement indépendante

Utilisations courantes :

- `main` ou `master` : branche principale stable
- `develop` : développement en cours
- `feature/nouvelle-fonctionnalite` : nouvelle fonctionnalité
- `bugfix/correction-capteur` : correction de bug
- `release/v1.2.0` : préparation d'une release

Slide 12 : Concepts Clés - Les États 🔄

Les trois états de Git

```
Working Directory  →  Staging Area  →  Repository
   (Modifié)          (Préparé)        (Validé)
```

1. **Working Directory** (Répertoire de travail)
   - Fichiers que vous modifiez
   - État : Modified
2. **Staging Area** (Zone de préparation)
   - Fichiers prêts à être commités
   - État : Staged
   - Commande : `git add`
3. **Repository** (Dépôt)
   - Fichiers validés dans l'historique
   - État : Committed
   - Commande : `git commit`

Slide 13 : Workflow Git de Base 🔁

Cycle de travail typique

```
1. Modifier des fichiers
   ↓
2. git add (préparer les modifications)
   ↓
3. git commit (valider les modifications)
   ↓
4. git push (envoyer vers le serveur)
```

Exemple concret :

```
# 1. Modifier main.cpp
vim main.cpp

# 2. Préparer le fichier
git add main.cpp

# 3. Valider avec un message
git commit -m "Ajout du support I2C"

# 4. Envoyer vers GitHub
git push origin main
```

Slide 14 : Installation de Git 💻

Windows

Git for Windows

- Télécharger : https://git-scm.com/download/win
- Installer avec les options par défaut
- Inclut Git Bash (terminal Unix-like)

macOS

Option 1 : Homebrew

```
brew install git
```

Linux

Debian/Ubuntu

```
sudo apt-get update
sudo apt-get install git
```

Fedora

```
sudo dnf install git
```

Slide 15 : Vérification de l'Installation ✅

Tester Git

```
# Vérifier la version installée
git --version

# Devrait afficher quelque chose comme :
# git version 2.43.0
```

Si la commande fonctionne, Git est installé ! 🎉

Slide 16 : Configuration Initiale ⚙️

Configuration de votre identité

Obligatoire avant le premier commit :

```
# Configurer votre nom
git config --global user.name "Jean Dupont"

# Configurer votre email
git config --global user.email "jean.dupont@example.com"

# Vérifier la configuration
git config --list
```

Pourquoi c'est important ?

- Chaque commit est signé avec ces informations
- Permet d'identifier l'auteur des modifications
- Utilisé par GitHub pour lier les commits à votre compte

Slide 17 : Configuration Recommandée 🔧

Paramètres utiles

```
# Éditeur par défaut (VS Code)
git config --global core.editor "code --wait"

# Couleurs dans le terminal
git config --global color.ui auto

# Nom de la branche par défaut
git config --global init.defaultBranch main

# Gestion des fins de ligne (Windows)
git config --global core.autocrlf true

# Gestion des fins de ligne (Mac/Linux)
git config --global core.autocrlf input
```

Slide 18 : Interfaces Git 🖥️

Options d'utilisation

1. Ligne de commande (CLI) ⌨️

- Plus puissant et flexible
- Recommandé pour ce cours
- Git Bash (Windows), Terminal (Mac/Linux)

2. Interfaces graphiques (GUI) 🖱️

- GitHub Desktop
- GitKraken
- Sourcetree
- VS Code (intégration Git)

3. Intégrations IDE 🔌

- VS Code
- Arduino IDE 2.0
- PlatformIO
- CLion

Recommandation : Apprendre la CLI d'abord, puis utiliser les GUI

Slide 19 : Aide et Documentation 📖

Obtenir de l'aide

```
# Aide générale
git help

# Aide sur une commande spécifique
git help commit
git commit --help

# Version courte de l'aide
git commit -h
```

Ressources en ligne :

- Documentation officielle : https://git-scm.com/doc
- Pro Git Book (gratuit) : https://git-scm.com/book/fr/v2
- GitHub Learning Lab : https://lab.github.com/
- Stack Overflow : https://stackoverflow.com/questions/tagged/git

Slide 20 : Exercice Pratique 1 🏋️

Installation et Configuration

Objectifs :

1. Installer Git sur votre machine
2. Configurer votre identité
3. Vérifier la configuration

Instructions :

```
# 1. Vérifier l'installation
git --version

# 2. Configurer votre identité
git config --global user.name "Votre Nom"
git config --global user.email "votre.email@example.com"

# 3. Vérifier la configuration
git config --list

# 4. Tester l'aide
git help status
```

Temps alloué : 10 minutes

Slide 21 : Vocabulaire Git Essentiel 📝

Termes à connaître

Terme	Définition
Repository	Dépôt contenant le projet et son historique
Commit	Instantané du projet à un moment donné
Branch	Ligne de développement indépendante
Merge	Fusion de deux branches
Clone	Copie locale d'un dépôt distant
Fork	Copie d'un dépôt sur votre compte GitHub
Pull	Récupérer les modifications du serveur
Push	Envoyer les modifications vers le serveur
Remote	Dépôt distant (sur GitHub par exemple)
HEAD	Pointeur vers le commit actuel

Slide 22 : Architecture Git

Comment Git stocke les données

Git utilise un système de snapshots, pas de différences

Avantages :

- Rapidité des opérations
- Intégrité des données (SHA-1)
- Efficacité du stockage (compression)

Slide 23 : Git pour Projets Embarqués 🤖

Cas d'usage spécifiques

Exemple : Projet Arduino

```
projet-arduino/
├── .git/
├── src/
│   ├── main.ino
│   ├── sensors.cpp
│   └── sensors.h
├── lib/
│   └── DHT/
├── docs/
│   ├── schema.pdf
│   └── README.md
├── .gitignore
└── platformio.ini
```

Bonnes pratiques :

- Versionner le code source
- Versionner la documentation
- Exclure les fichiers compilés (.hex, .bin)
- Exclure les dépendances téléchargeables
- Inclure les fichiers de configuration

Slide 24 : Fichiers à Versionner ou Non ✅❌

Que mettre dans Git ?

À VERSIONNER ✅

- Code source (.c, .cpp, .h, .py, .ino)
- Documentation (.md, .txt)
- Fichiers de configuration
- Scripts de build
- Schémas (formats texte : KiCad, Eagle XML)
- README et LICENSE

À NE PAS VERSIONNER ❌

- Fichiers compilés (.o, .hex, .bin, .elf)
- Fichiers temporaires (.tmp, .bak)
- Dépendances téléchargeables (node_modules, .pio)
- Fichiers IDE (.vscode, .idea)
- Fichiers système (.DS_Store, Thumbs.db)
- Données sensibles (mots de passe, clés API)

Solution : fichier `.gitignore`

Slide 25 : Modèle Mental de Git 🧠

Comprendre Git

Git n'est PAS :

- ❌ Un système de backup
- ❌ Un Dropbox pour code
- ❌ Compliqué si on comprend les concepts

Git EST :

- ✅ Un système de gestion d'historique
- ✅ Un outil de collaboration
- ✅ Un graphe de commits
- ✅ Puissant et flexible

Clé du succès :

- Comprendre les concepts de base
- Pratiquer régulièrement
- Faire des commits atomiques et descriptifs
- Ne pas avoir peur d'expérimenter (avec des branches)

Slide 26 : Récapitulatif Module 1 📋

Ce que nous avons appris

✅ Concepts fondamentaux

- Qu'est-ce que Git et pourquoi l'utiliser
- Différence entre Git et GitHub
- Vocabulaire essentiel

✅ Installation et configuration

- Installation de Git
- Configuration de l'identité
- Paramètres recommandés

✅ Architecture Git

- Les trois états (Working, Staging, Repository)
- Workflow de base
- Modèle de données

✅ Applications pour l'électronique

- Cas d'usage spécifiques
- Bonnes pratiques

Slide 27 : Questions ? 🙋

Discussion

Points à clarifier ?

- Concepts pas clairs ?
- Questions sur l'installation ?
- Cas d'usage spécifiques ?

Prochaine étape :
Module 2 - Commandes Git Essentielles

Slide 28 : Pause ☕

Pause de 5 minutes

Avant de continuer :

- Assurez-vous que Git est installé
- Vérifiez votre configuration
- Préparez votre terminal

Rendez-vous dans 5 minutes pour le Module 2 !

Notes pour le formateur 👨‍🏫

Timing suggéré

- Slides 1-8 : Introduction (15 min)
- Slides 9-17 : Concepts et installation (20 min)
- Slides 18-20 : Exercice pratique (15 min)
- Slides 21-26 : Approfondissement (10 min)

Points d'attention

- Vérifier que tous les étudiants ont Git installé
- Adapter les exemples aux projets des étudiants
- Encourager les questions
- Prévoir du temps pour le dépannage

Matériel nécessaire

- Projecteur
- Accès Internet
- Ordinateurs pour les étudiants
- Droits administrateur pour installation