Travail sur STM32
Projet SuperBeeLive

Apprentie : **Olivia SERENELLI-PESIN**
Maître d'apprentissage : **Sébastien DRUON**

Table des matières

Chapitre 1

Présentation d'une implémentation sous STM32

1.1 Contexte

Le but de ce travail de découverte du microcontrôleur STM32 est de pouvoir utiliser toutes les cartes STM32 sans dépendre d'un seul IDE ou logiciel tier : nous voulons être indépendants pour la création et l'utilisation de nos codes et nous voulons passer que par des outils libres. Pour toutes les installations et tests effectués sur ce document, nous sommes sur une machine sous Ubuntu.

1.2 Les étapes : de la compilation à l'implémentation

Comme pour la programmation en C classique, nous allons utiliser gcc pour compiler notre code. Avant tout, précisons que nous n'utiliserons pas les librairies standards du C pour coder sur notre STM : celles-ci sont standards dans la mesure où nous les utilisons sur un ordinateur, elles viennent avec le Kernel et les Drivers, ce qui n'est pas le cas avec les microcontrôleurs. Cependant, la programmation C sous STM32 possède ces propres standards : les librairies HAL et LL. Elles contiennent tous les liens entre le software et l'hardware et nous évitent de connaître précisemment dans quelle case mémoire écrire pour activer/désactiver les composants de notre STM. Pour débuter nous allons nous en servir, mais au fur et à mesure nous nous efforcerons de nous en séparer.

Notre toolchain pour débuter sous STM32 sera composée de :

— CubeMX, pour nous faciliter la création des fichiers de bases (Makefile, src, Driver etc) en fonction de la carte que nous utilisons.
— GCC, et plus précisement gcc-arm-eabi-none, pour compiler notre code
— STTLINK, pour téléverser (flasher) le code sur notre STM
— OpendOCD, pour débugger notre code une fois sur la STM

À mesure de la prise en main de la toolchain, nous tenterons de nous passer de l'outil CubeMX.

Chapitre 2

Installation des outils

2.1 Installation de CubeMx

Pour l'installer, il suffit de se créer un compte sur le site officiel de STM et de télécharger l'outil en passant par ce lien : https ://www.st.com/en/development-tools/stm32cubemx.html Une fois le .zip téléchargé, décompressez-le et suivez les instruction d'installation sur le README. Notons qu'une installation de Java est nécessaire, sous Ubuntu la commande suivante permettra de l'installer :

```
apt-get install default-jre
```

2.2 Installation de GCC

Faites une installation basique suivant votre OS. Dans notre cas, nous allons faire la commande

```
apt-get install gcc
```

Et il ne faut pas oublier le paquet qui nous permet de compiler pour notre carte STM :

```
apt-get install gcc-arm-none-eabi
```

Nous voilà prêt pour compiler notre code.

2.3 Installation de stlink

L'installation (cf https ://freeelectron.ro/installing-st-link-v2-to-flash-stm32-targets-on-linux/) de stlink va se faire via le répot Github de texane.

```
cd ~myusername
mkdir stm32
cd stm32
git clone https://github.com/texane/stlink
cd stlink
cmake .
make
sudo cp st-* /usr/local/bin
sudo cp etc/udev/rules.d/49-stlinkv* /etc/udev/rules.d/
```

Chapitre 3

Le Hello World du STM

Une fois tous les outils installés, nous pouvons faire l'exemple le plus basique possible. Commençons par lancer le logiciel STM32Cube.

```
cd ./usr/local/STMicroelectronics/STM32Cube/STM32CubeMX/
./STM32CubeMX
```

Cliquez sur "Access to Board Selector" dans "Start my project from STBoard". Sélectionnez la carte que vous souhaitez utiliser. Dans notre cas, ce sera la stm32f3discovery. Vous êtes directement redirigé dans l'onglet "Pinout & Configuration". Vous pouvez ici configurer les sorties et entrées de la carte. Pour faire simple, nous allons sélectionner la led bleue en PE8. Dans le menu de gauche, allez dans System Core -> PE8 et changez le User Label en "LED". Ce nom est totament arbitraire et vous pouvez choisir de le nommer comme vous le souhaitez. Une fois fait, allez dans l'onglet "Project Manager" et donnez un nom à votre projet dans "Project Name". Choisissez ensuite le dossier dans lequel vous allez l'enregistrer. Ensuite, sélectionnez "Makefile" dans le menu déroulant "Toolchain / IDE" Enfin, cliquez sur "Generate Code" tout en haut à gauche.

Vous pouvez maintenant retrouver votre code main.c dans le dossier que vous avez indiqué juste avant et dans Src. Editez le avec l'éditeur de votre choix (Vim...) et allez dans la boucle while qui permet d'exécuter le code pour y écrire :

```
while(1) {
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin) ;
    HAL_Delay(500) ;
}
```

La première ligne permet d'allumer la Led demandée, ici celle que nous avons nommé LED. La seconde ajoute un délais de 500ms.

Une fois le code écrit, on sauvegarde et on compile avec le fichier Makefile généré automatiquement par STM32Cube.

Un fois compilé, il faut se rendre dans le dossier build pour y retrouver le code binaire généré.

Maintenant que nous avons le code prêt, il est temps de le téleverser dans son microcontrôleur. Branchez la carte et faites la commande dmesg pour voir si la carte a bien été connectée. Vous pouvez faire la commande suivante afin de récupérer les informations sur la carte :

```
st-flash --debug read dummy.file 0 256
```

Ensuite, on va taper la commande suivante pour envoyer le code sur la STM32

```
st-flash write test.bin 0x8000000
```

Avec test.bin qui est le nom du .bin généré. Et voilà, le code est sur la carte !