Université Paul Sabatier
Master 1 SME / 2016-2017
2016-2017

Techniques pour le Temps Réel

Prise en main de l'OS temps réel TRAMPOLINE

1 Installation de l'environnement de développement

Pour développer une application utilisant Trampoline et s'exécutant sur une carte STM32F4Discovery, plusieurs éléments sont nécessaires :

- L'environnement de développement de Trampoline (code source, utilitaire de compilation, bibliothèques spécifiques pour portage vers la carte)
- Chaîne de compilation pour le processeur de la carte
- outils de communication avec la carte pour transférer l'application de la machine de développement vers la carte

1.1 Environnement Trampoline

Le projet Trampoline est aujourd'hui hébergé sur `github.com`. Il est possible d'accéder à la page dédiée au projet en faisant une recherche avec les mots clefs "trampoline" et "RTOS".

1.1.1 Téléchargement des sources du projet

Pour récupérer l'environnement Trampoline sur votre poste de travail, on utilisera la commande :

```
git clone https://github.com/TrampolineRTOS/trampoline.git
```

1.1.2 Compilation et installation du compilateur `goil`

Les fichiers de description de l'application (`.oil`) doivent être compilés à l'aide du compilateur `goil`. Il est nécessaire de compiler ce compilateur et de l'installer sur votre session. Pour compiler `goil` :

- placez-vous dans le répertoire `trampoline/goil/makefile-unix`
- lancez le script python build.py avec la commande
  `python build.py`
- pour permettre d'utiliser `goil` depuis n'importe quel répertoire de votre compte utilisateur, il faut l'ajouter à la variable d'environnement `PATH`. Pour se faire, on éditera le fichier `.bashrc` qui est dans le répertoire racine de votre compte. On ajoutera en fin du fichier la ligne :
  `export PATH=$PATH:~/trampoline/goil/makefile-unix`
- Cette modification sera prise en compte lors de la prochaine ouverture d'un terminal. Pour prendre en compte la modification sans avoir à fermer puis ré-ouvrir le terminal, utiliser la commande :
  `source .bashrc`

1.2 Chaîne de compilation

Canonical met à disposition une chaîne de compilation pour processeur ARM (d'autres sources existes).

- Télécharger les fichiers d'installation pour linux à partir de
  `https://launchpad.net/gcc-arm-embedded`
- Décompresser l'archive récupérée (qui est au format `tar.bz2` normalement) avec la commande :
  `tar jxvf gcc-arm-none-eabi-4_9-2015q3-20150921-linux.tar.bz2`
- si ce n'est pas déjà le cas, placer le répertoire créé (`gcc-arm-none-eabi-4_9-2015q3`) dans la racine de votre compte.
- comme pour le compilateur `goil`, il faut que les outils de compilation soient exécutables depuis n'importe quel répertoire de votre compte. On ajoutera donc le répertoire contenant tous ces exécutables à la variable d'environnement PATH. Les exécutables sont dans le répertoire `gcc-arm-none-eabi-4_9-2015q3/bin`.

1.3 Outils stlink

Les outils *stlink* permettent à la machine de développement de s'interfacer avec la carte à des fins de programmation ou de debuggage. Par défaut, trampoline utilise l'utilitaire `st-flash`, qui fait parti de stlink, pour charger la carte. Une version pour linux de stlink est disponible sur `github.com`.

- Dans le répertoire racine de votre compte, récupérer le projet texane/stlink avec la commande :
  `git clone https://github.com/texane/stlink.git`
- Ces outils ont une dépendance avec la librairie libusb1.0, il faut donc récupérer les sources de cette librairie pour permettre la compilation des outils stlink
- comme pour les outils de compilation, il faut que les utilitaires de stlink soient exécutables depuis n'importe quel répertoire de votre compte. Il faut donc ajouter le répertoire de stlink à la variable d'environnement `PATH`.

2 Au travail !

Récupérer le répertoire `labs` et placer le dans votre répertoire `trampoline`. `labs` contient les sujet des manipulations ainsi qu'une copie du cours sur Trampoline (voir le sous répertoire `labs/labs_stm32F4_discovery/doc`). Les sujets sont dans les deux fichiers suivants :

- `labs.pdf` : contient le premier sujet à traiter. Le sujet (écrit par Jean-Luc Bechennec, créateur de trampoline) propose une mise-en-œvre progressive des concepts fondamentaux de Trampoline/OSEK-VSX.
- `etude_de_cas.pdf` : support pour la deuxième partie des TPs sur OSEK/VDX. Le fichier contient une étude de cas (extrêmement simplifiée) d'une application de régulation de distance de type "Adaptive Cruise Control" utilisée dans l'automobile. L'objectif est de comprendre les exigences temps réel de l'application et de proposer une architecture logicielle répondant à ces exigences.

Le répertoire `labs` contient également deux fichiers sources (`tp_ups.c` et `tp_ups.h`) d'une bibliothèque facilitant la configuration de la carte. Il faudra les copier dans le répertoire contenant les bibliothèques de la carte, ie :

`trampoline/machines/cortex/armv7em/stm32f407/stm32f4discovery`

La documentation de trampoline se trouve dans le répertoire

`trampoline/documentation`