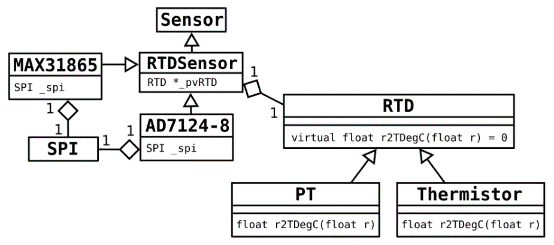
Architecture de l’ensemble du système

Architecture d’un driver de température avec une interface spi MAX31865. (capteur rtd)

- Au niveau du main() on scrute les changements (clock, time, alarm) et nous avons une instance de connecsens(initialisation(), lookAtResetSource, process, EndOfExecution)

On scrute en permanence avec le while(1).

Ensuite le programme passe à une instance **static** ConnecSenS \_env

- Au niveau connecsens vous faites beaucoup de choses (CSV, network, sensor …) Initialisation …

Nous avons une instance Sensor \*pvSensor

- Au niveau Sensor (la classe capteurs) à partir d’ici il découle notre architecture figure1 le point d’entrée est le MAX31865. En passant par la classe factory (design pattern) qui crée une instance pour chaque capteur avec Sensor.

Question/Réponse

//***mais Sensor est une classe virtual on peut quand même créer une instance alors ? si on fait un pointeur ?***

Non, il n’est pas possible de créer une instance d’une classe virtuelle, il n’est donc pas possible de créer directement une instance de la classe Sensor. Cette classe Sensor définie l’interface commune à tous les capteurs, elle nous permet de traiter de manière identique des capteurs de type différents, mais qui présentent des fonctionnalités de base communes (les fonctions définies dans cette classe Sensor). Et la classe ConnecSenS utilise uniquement ces fonctionnalités de base.

Il est uniquement possible de créer des instances de classes dérivées de Sensor et dont toutes les fonctions sont définies (dont plus aucune n’est purement virtuelle). La classe MAX31845 est une telle classe, qu’il est possible d’instancier. La liste de classes instanciables est dans la « factory ».

Comme toute classe dérivée d’une classe mère est cette classe mère plus autre chose (c’est une augmentation de la classe mère), il est possible d’utiliser la classe fille en place de la classe mère. Il est donc possible d’utiliser une instance MAX31865 partout où une instance Sensor est attendue.

Il est possible de créer des pointeurs de type Sensor, car la création d’un pointeur ne provoque pas la création d’une instance de la classe pointée. Un pointeur est simplement une variable 32bits (ici) qui contient l’adresse qui, si tout va bien, contient l’adresse d’une instance qui peut être considérée être de type Sensor, ou bien est 0 (NULL).

Débugger avec des logs ou le debugger/STLINK

**Réponse :**

Il existe deux méthodes principales :

La méthode basique, mais parfois très utile, est de créer des messages de log, comme par exemple : **log\_debug**(**\_logger**, "Invalid number of wires.");

Il existe une multitude d’exemples de telles lignes de log dans le programme. Vous pouvez ensuite les supprimer une fois les tests terminés si ces lignes ne sont plus utiles. Il est alors commode de rediriger également les messages de log vers une sortie série de nœud et de les lire en temps réel au moyen d’un adaptateur USB vers série TTL.

Je vous renvoie encore une fois vers la documentation sur le développement pour le nœud ConnecSenS.

La seconde méthode, et la plus puissante, et d’utiliser la sonde de debug JTAG. Il est alors possible de placer des breakpoints (points d’arrêt) et de faire de l’exécution pas à pas du programme.

Il existe probablement des tutoriels sur le WEB si vous n’êtes pas familier avec cette méthode. Elle n’est pas bien compliquée, seuls quelques concepts de base sont mis en jeu.

Le mode le plus courant d’utilisation de la sonde de debug est d’utiliser des « breakpoints ». Si vous double cliquez à la gauche d’un numéro de ligne dans un des éditeurs de texte il est possible d’activer ou de désactiver un breakpoint. Vous pouvez également passer par le menu « Run ».

Lors de l’exécution en mode debug, avec la sonde de débogage, le programme s’arrête lorsqu’un break point est rencontré. Il est alors possible de lire la valeur de différente variables et de procéder à une exécution pas à pas pour suivre finement l’évolution du programme et de son environnement.

**Le fichier Versionning**

il nous demande de changer la version et nous dit qu’il manque le fichier version-rev.h



Mais ça n’empêche pas en rien la compilation

**Réponse :**

Le fichier « version-rev » est normalement généré avant la compilation par un programme Java appelé par Attolic. Pour qu’il soit généré, il faut modifier la configuration de compilation d’Atollic.

Pour que le programme Java fonctionne, il faut que les sources fassent partie d’un dépôt git. C’est probablement le cas, vos sources sont probablement le fruit du clonage du dépôt git de la forge UCA.

Ce fichier permet, en particulier, d’intégrer le numéro de révision git dans le numéro de version du logiciel du nœud.

Pour vous assurer que la configuration est correcte :

1. Dans l’explorateur de projet « Project Explorer », faites un clic droit sur le projet « application ».
2. Puis cliquez sur *Properties→C/C++ Build→ Settings→ Build Steps*.
3. Dans la liste déroulante « Configuration » en haut de la fenêtre des « properties » sélectionnez l’option « All configurations ».
4. La ligne de saisie *Pre-build Steps→ Command* doit contenir une valeur. Si ce n’est pas le cas, saisissez alors la valeur suivante : ${TOOLCHAIN\_PATH}/../../ide/jre/bin/java -classpath ${ProjDirPath}/scripts GetRevision ${ProjDirPath}/Inc/version-rev.h
5. Cliquez sur le boutton « Apply » ou « Ok » si vous avez changé la valeur.