Blondet Thomas
Torck Quentin

RICM4

MONOSKI INTELLIGENT
Défy Foly 2015 :

POLYTECH
GRENOBLE

Contexte :

Le Défi Foly est une compétition de water-slide. Ce challenge consiste à parcourir à skis, monoski, snowboard ou avec tout engin glissant similaire, la plus grande distance sur le lac des Confins. Depuis 1986, ce sont plus de 150 sportifs par an qui ont tenté la traversée du lac, mais jamais réussi. L'année dernière Antoine Crochemore (MAT4) a réalisé une performance de 121 mètres et est arrivé 3ème au concours.

Cette année Polytech Grenoble participe encore une fois à l'épreuve, avec de nouveaux monoskis. Pour ce faire l'école a mobilisé 3 équipes. Une équipe Matériaux qui fabrique les monoskis afin qu'ile soient encore plus performant que l'année précédente, une équipe de 3I qui vont s'occuper de toute la partie électronique (ils vont installer différentes jauges sur le monoski, les configurer afin que celles-ci puissent envoyer des données de déformation du ski). Enfin il y à l'équipe RICM, qui se charge de l'instrumentation et de la partie transmission de données vers un serveur afin que ces celles-ci puissent être analysée plus facilement par les matériaux. Ces données seront analysées l'année prochaine par les matériaux afin de créer un monoski encore plus performant.

Objectifs :

L'objectif principal est de pouvoir récupérer les données transmises par les capteurs (température, pression, accéléromètre, gyroscope…). Pour ce faire nous nous sommes fixé différents objectifs :

→Le premier objectif est de pouvoir stocker les données sur une carte SD depuis la carte stm32, pour pouvoir récupérer ces données sur un ordinateur, et pouvoir les analyser avec différents outils.

→Le deuxième objectif est de créer un serveur web, qui intègre une base de données, relié à une interface graphique, afin de pouvoir analyser les données stockées et les transformer en graphes.

→ Le troisième objectif est de créer un parseur de données qui lit les données de notre carte SD pour les transformer en requêtes http et les envoyer vers notre base de donnée.

→ Le quatrième objectif consiste à créer une application qui permet de lancer et arrêter la capture sur la carte stm32 Nucleo. Cela permet d'éviter de devoir ouvrir le boitier contenant la carte STM32 lors de la descente du Défi Foly.

→ Enfin le dernier objectif consiste à réaliser une application qui contiendra l'objectif précédent ainsi que le parsing des données et l'envoi de celles-ci vers le serveur depuis l'application Android.

Comment Fonctionne Notre Projet ?

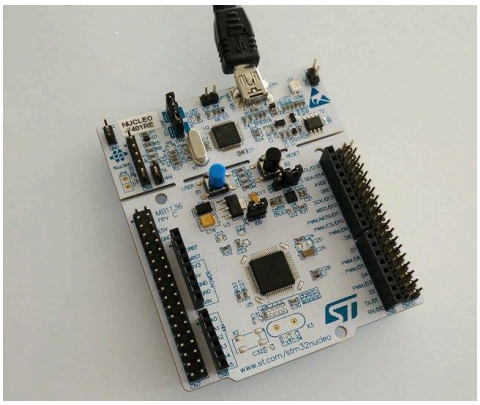
Pour démarrer la capture nous flashons notre programme sur la carte STM32. Quand le programme a été flashé une première fois sur la carte, il suffit de presser le bouton user pour lancer la capture. Lorsqu'on appuie une seconde fois sur ce bouton la capture s'arrête. Nous savons que la capture est en cours de fonctionnement car une led clignote. Pendant cette période de capture les données sont stockées sur la carte SD (Il y a un fichier par capture qui est créé). Une fois la capture effectuée nous récupérons les fichiers pour les transformer en requêtes http grâce à notre parser. Ces requêtes sont écrites dans un fichier Shell. Une fois le parsing des données effectué, nous exécutons le Shell grâce à la commande suivante :*/<nom du fichier>.sh* pour procéder à l'envoi des données sur le serveur. Une fois les données envoyées sur le serveur il suffit de se rendre sur la page suivante : http://54.93.89.31/grafana/#/dashboard/db/defi-foly-2015 pour pouvoir observer les graphes concernant les différents capteurs.

Travail Réalisé :

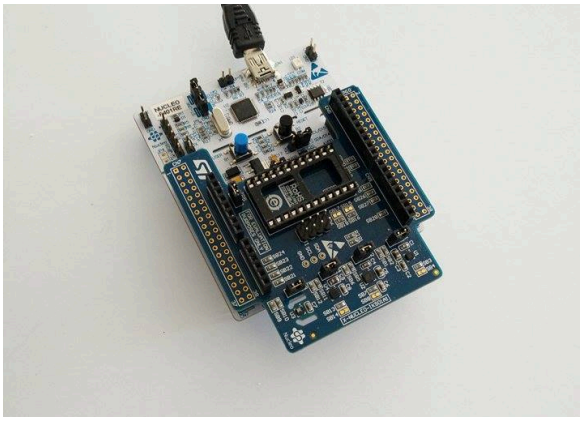
Détail de la partie carte :

1→ Description des différents composants :

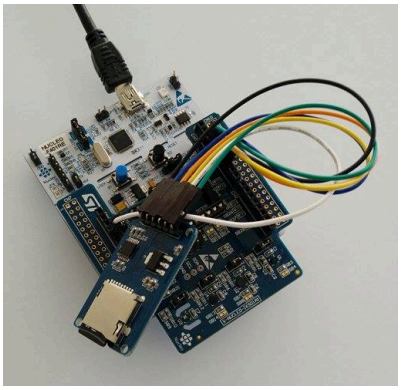
Le système embarqué sur le monoski est composé de plusieurs modules basés sur une carte STM32 Nucleo F401-RE. Cette carte fonctionne avec un CPU ARM®32-bit Cortex®-M4. Nous allons donc utiliser l'interface du site de ARMmbed pour créer et compiler nos programmes.



Cette base est surmontée du module X-NUCLEO-IKS01A1 qui contient une centrale inertielle ainsi que plusieurs autres capteurs (hygromètre, thermomètre, baromètre et magnétomètre). Ce module permet donc de réaliser toutes les mesures nécessaires dans notre projet.



Enfin, un lecteur de carte microSD est câblé grâce à une liaison SPI. Cette carte est la mémoire de notre système, toutes les mesures capturées seront inscrites dans un fichier « run.txt » qui permettra de délogger la course du monoski.



2→ Prise en main de la carte :

Après s'être documenté sur le fonctionnement de la carte STM32 Nucleo et sur son environnement, nous avons choisi d'utiliser le site d'ARMmbed pour créer notre projet. En effet, ce site est très pratique tant au niveau du compilateur qu'au niveau de la communauté active pour s'entraider. De plus, il est très simple d'inclure des bibliothèques externes dans le projet.
Nous avons donc dans un premier temps testé les différents programmes proposés par mbed pour la STM32 Nucleo.
Après avoir étudié le fonctionnement de la carte (utilisation des différents pins, de la LED et du bouton user) nous avons commencer à créer notre programme en débutant par l'utilisation de la carte microSD.

3→ Utilisation des bibliothèques :

Pour utiliser la carte STM32 ainsi que les modules nous avons besoin de plusieurs bibliothèques externes :

#include "mbed.h" : c'est la bibliothèque de base pour pouvoir utiliser les différents pins de la STM32. Nous avons donc accès au bouton, à la led et à toutes les autres interfaces telle que les pins pour la liaison SPI.

#include "SDFileSystem.h" : cette bibliothèque permet de gérer un système de fichier sur une carte SD. Elle contient aussi la bibliothèque FATFileSystem qui gère toutes les fonctions pour créer des dossiers et ouvrir et éditer des fichiers.

#include "x_cube_mems.h" : cette dernière bibliothèque sert à communiquer avec le module MEMS pour récupérer les mesures des capteurs.

Détail de la partie serveur :

1→ Création et configuration d'une machine linux sur le serveur d'amazone :

Il faut tout d'abord se rendre sur le site *https://eu-central-1.console.aws.amazon.com/ec2/v2/home?region=eu-central-1*. Puis il faut sélectionner l'onglet **launch instance** pour créer une instance de notre machine virtuelle. Il suffit de sélectionner une machine Ubuntu en version 14.4. Ensuite il faut configurer les accès pour pouvoir accéder à son instance de manière assez facile tout en la sécurisant au maximum. Une fois que ces paramètres sont configurés il suffit d'accéder en ssh à sa machine en tapant ***ssh -i clé.pem ubuntu@54.93.89.31.*** Quand nous somme arrivé sur cette machine nous somme parti d'une machine vierge, il a donc été nécessaire d'installer de nombreux packages de base. Enfin avant d'éteindre notre serveur nous effectuons un snapshot afin de ne pas perdre la configuration que l'on avait effectué.

2→ Installation et configuration d'InfluxDB (base de donnée) :

Dans un premier temps nous avons procédé à l'installation d'InfluxDB en tapant les commandes suivantes : ***wgethttp://s3.amazonaws.com/influxdb/influxdb_latest_amd64.deb***
sudo dpkg -i influxdb_latest_amd64.deb
Pour initialiser le serveur : ***sudo /etc/init.d/influxdb start***
Lorsque le serveur est initialisé il faut se rendre à l'adresse suivante pour commencer la configuration : ***54.93.89.31:8083.***
La première étape de configuration sur InfluxDB consiste à créer un utilisateur qui sera administrateur et lui mettre un mot de passe. Il faut ensuite supprimer l'utilisateur root.
Après avoir créé de nouveaux utilisateurs nous avons créé deux bases de données une base de données destinée aux test et une base de donnée destinée a notre projet. La base de donnée destinée au projet s'appelle monoski-data. Nous avons ensuite remplis notre base de donnée test en créant des tables et en ajoutant des valeurs à ces tables afin de voir le format des données JSON qui était requis. Ce format nous est en effet utile pour la partie parsing de données.

3→ Installation et configuration du serveur Apache :

Pour installer et configurer Apache, la première chose que nous avons effectué était de taper la commande suivante :***sudo apt-get install apach2*** puis nous avons effectué toutes les configurations indiquées dans cette page internet : http://doc.ubuntu-fr.org/apache2 (modifier les fichiers de configuration pour pouvoir créer des pages html et les afficher sur notre serveur).
Enfin quand on veut éteindre ou redémarrer notre serveur nous exécutons les commandes suivantes : ***sudo service apache2 start | stop | restart | status.*** Lorsque notre serveur est actif, nous arrivons sur notre page web via l'adresse IP ***54.93.89.31***. Il ne nous suffit plus que de configurer la page web pour afficher le site Grafana.

3→ Installation et configuration de Grafana :

Dans cette étape, nous voulons afficher notre page Grafana en entrant l'url suivante : http://54.93.89.31/grafana depuis un navigateur. Le but est d'arriver à une page affichant les graphes de température, de pression, d'accélération, du magnétomètre et du gyroscope. Pour ce faire nous avons dans un premier temps téléchargé sur notre serveur le package Grafana :

wget http://grafanarel.s3.amazonaws.com/grafana-1.9.1.zip

unzip grafana-1.9.1.zip sudo mv grafana-1.9.1 /opt

Quand le package est placé au bon endroit, nous modifions le fichier de configuration config.js pour que Grafana soit directement connecté à notre base de donnée InfluxDB :

```
datasources: {
        influxdb: {
          type: 'influxdb',
          url: "http://54.93.89.31:8086/db/monoski-data",
          username: 'username',
          password: 'motde passe',
        },
        grafana: {
          type: 'influxdb',
          url: "http://54.93.89.31:8086/db/grafana",
          username: 'username',
          password: 'motde passe',
          grafanaDB: true
        },
```

Apres avoir fait cette configuration Grafana sera normalement relié à notre base de donnée InfluxDB. Il nous reste encore a configurer Grafana pour qu'il puisse être accessible depuis notre serveur. Nous avons donc crée le script suivant :

```
cat > /tmp/grafana.conf <<EOF
Alias /grafana /opt/grafana-1.9.1

<Location /grafana>
Order deny,allow
Allow from 127.0.0.1
Allow from ::1
Allow from all
</Location>
EOF

sudo mv /tmp/grafana.conf /etc/apach2/conf.d/grafana.conf
```

Apres cette étape nous avons ouvert certain port de notre serveur pour éviter de bloquer l'accès à notre page Grafana. (Nous avions l'erreur *permission denied* qui apparaissait quand nous nous rendions sur notre page web) .

La dernière étape de configuration de Grafana était la suivante : configurer les graphes afin qu'ils affichent correctement les données en fonction du temps. Nous avons alors suivi un tutoriel que nous avons trouvé sur le site de Grafana.

A la fin de notre configuration le résultat est le suivant : http://54.93.89.31/grafana/#/dashboard/db/defi-foly-2015

Détail de la partie parseur et transmission de données :

1→ Etude du format de réception de donnée

Le format de sorti de la carte STM32 est le suivant : la première ligne du fichier contient le nom des capteurs séparés d'une tabulation. Ensuite le fichier est composé des valeurs des capteurs. Toutes ses valeurs sont egalement séparées par des tabulations.

2→ Création d'un parser

Nous avons choisi d'écrire notre parser en java pour l'integrer plus facilement dans notre application Android. Il commence par lire la première ligne pour récupérer les noms des capteurs qu'il stock dans des variables. Puis il récupère les valeurs des capteurs et les stock également dans d'autres variables. A chaque ligne lue, notre parser réécrit ces valeurs sous forme de requête http au format Json. Ces requêtes http sont écrites dans un script Shell.
Par exemple :
Format d'entrée :

```
998.992676  24.910000  41.980000  3  -3  995  -490  -1120  1050  149  -474  738
```

Format de sortie :

```
curl -X POST -d
'[{"name":"capteurs3","columns":["P","T","H","AccX","AccY","AccZ","GyrX","GyrY","GyrZ","MagX","MagY","Ma
gZ"],"points":[[998.992676,24.910000,41.980000,3,-3,995,-490,-1120,1050,149,-474,738]]}]'
'http://54.93.89.31:8086/db/monoski-data/series?u=quentin74100&p=monoski2015'
```

3→ Envois de requêtes vers le serveur

Pour l'envoi des requêtes http nous avons besoins d'ajouter les droits d'exécution pour l'utilisateur au script sortie.sh grâce à la commande **chmod u+x sortie.sh.** Nous exécutons enfin ce fichier pour envoyer toutes les requêtes au serveur.

Travail restant

Si ce projet est remis en jeux l'année prochaine, de nombreux points devront être améliorés. Il faudra par exemple revoir entièrement la partie applicative. Pour le moment notre application est capable de se connecter à un téléphone en bluetooth ou à une Arduino, mais n'est pas capable de se connecter à la STM32 Nucleo. Dans un second temps il faudra intégrer le parser dans l'application que nous avons réalisé pour qu'il transforme les données reçues en requêtes http. L'application devra alors commander la carte STM32 Nucleo pour lancer la capture et l'arrêter. Elle devra aussi récupérer le fichier de données via la liaison bluetooth et l'envoyer sur un serveur grâce à notre parser. On pourrait aussi penser à intégrer la localisation GPS du smartphone aux données pour obtenir le profile de la descente du skieur.

Les Problèmes rencontrés :

Durant ce projet, nous avons rencontré de nombreux problèmes. La plupart ont été résolus :
- Intégration de Grafana dans le serveur ; une fois installé nous n'avions pas accès à la page d'accueil.
- Incohérence entre le timestamp des données capturées et le timestamp d'InfluxDB.
- Incompatibilité de certaines bibliothèques ARM entre-elles.

Cependant l'impossibilité de détecter le shield Bluetooth de la STM32 Nucleo avec un autre appareil n'a pas été résolue ce qui a freiné notre projet quant à la partie applicative.

Conclusion :

En début de semestre nous avons eu la chance de choisir ce projet qui nous a immédiatement intéressé dès le dévoilement des sujets. En effet, étant tout 2 originaires de Haute-Savoie et skieurs, nous avions déjà connaissance de cette compétition qui est le Défi Foly. La collaboration active avec l'équipe des Matériaux que nous connaissons bien et les 3I a permis le très bon déroulement de ce projet inter-filière qui mettra en avant l'école lors du week-end du 18-19 avril.