Rapport de séance du Mardi 7 Janvier 2020 (séance 2):

La séance d'avant les vacances (Vendredi 20 Décembre 2019) a été annulée pour cause de mauvais temps.

Une chose que j'ai oublié de dire dans le rapport précédent: j'ai commencé le développement d'une application Java pour dessiner les pancakes. Pour me simplifier la vie, j'ai réutilisé le code du TP "ardoise magique" d'OpenClassrooms (https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/24857-tp-creez-une-ardoise-magique). J'ai réalisé ce TP à l'époque où j'apprenais le Java. Le code de l'application PC est donc basé sur celui du TP (mon code, pas la correction). Vous trouverez le code dans le dossier **src/pancakeDrawer**.

Pendant la semain "sans Arduino" et pendant les vacances, je n'ai pas pu tester le nouveau matériel. Je me suis donc occupé à écrire proprement des classes pour le code de fonctionnement des moteurs pas-à-pas. Vous trouverez ces codes dans le dossier src. En haut de votre fenêtre Github, vous trouverez un onglet **Wiki**. Il y a une documentation pour chacune des classes que j'ai écrites. Ne vous étonnez donc pas de l'absence de commentaires.

Pour avoir un code plus "propre", chaque classe est encapsulée dans le namespace pancar. Par exemple, pour accéder à la classe Stepper, il faudra écrire pancar::Stepper.

Les différentes classes écrites pendant l'intervalle entre les 2 séances:

Stepper: Contrôle les moteurs pas-à-pas

GroupStepper: Permet de contrôler un groupe de moteurs pas-à-pas. En effet si vous demandez à un moteur de faire n pas, le programme se bloque jusqu'à ce que les n pas soient effectués. Cela vient du fait qu'il n'y a qu'un seul fil d'exécution sur Arduino (monothread). L'astuce consiste donc à alterner les pas pour donner l'illusion que les 2 moteurs tournent en même temps, ce qui est vrai si on tient compte de l'inertie.

Vector: Permet de créer des tableaux dynamiques, dans le style des collections de Java ou des conteneurs de la bibliothèque standard du C++ (qui n'est évidemment pas disponible en Arduino). J'ai eu beau chercher, je n'ai pas trouvé la moindre bibliothèque pour faire ça (c'est pourtant très pratique). Ca va nous être utile pour le dessin du pancake, qui est en réalité une collection de points (la classe Point est à faire, à voir si Clara veut la faire). Peut-être que la classe **GroupStepper** va en hériter, à voir... Cette classe fonctionne par une récursivité sur des pointeurs de tableaux de pointeurs de référence de template. L'utilisation des pointeurs s'avère difficile en Arduino parce que comme on programme dans l'espace noyau, on ne bénéficie pas des protections de l'espace utilisateur (erreurs de segmentation...) et donc le code

est difficile à valider. Pour plus d'infos: https://fr.wikipedia.org/wiki/Espace_noyau https://fr.wikipedia.org/wiki/Espace_utilisateur https://wiki.osdev.org/Main_Page. D'autre part le compilateur fourni avec le logiciel Arduino (avr-g++) ne gère pas par défaut les exceptions (try, catch...). Il semblerait cependant qu'il soit possible d'activer cette fonctionnalité. Pour garder mon code le plus portable possible, j'ai décidé de ne pas utiliser les exceptions. Il en résulte malheureusement un code un peu moins propre, et ce d'autant plus qu'on programme dans l'espace noyau. J'ai prévu un pdf d'explication du code dans le dossier **documents divers**: **pancarvector - explications.pdf**.

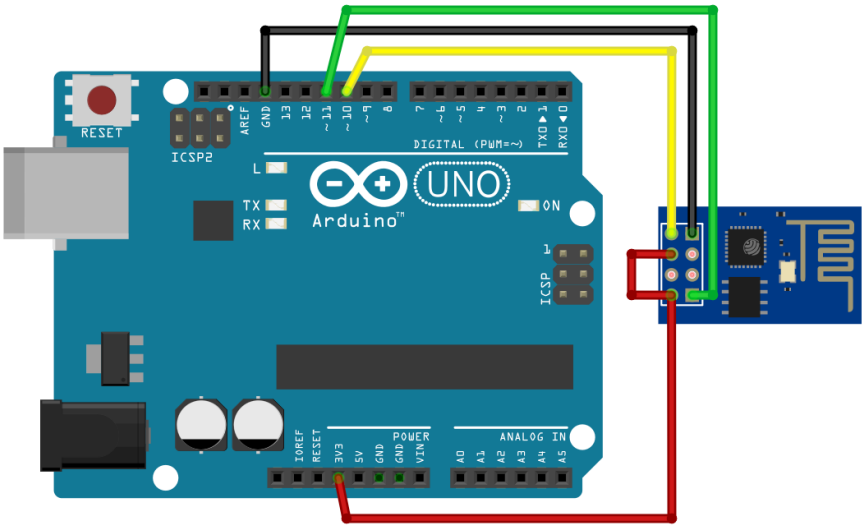
Pour information, j'avais utilisé le cours suivant pour apprendre le langage C++: https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c

Pendant la séance, je me suis demandé comment établir la communication entre mon programme Java sur PC et la carte Arduino. Comme dit plus haut, l'application PC a été faite au mois de Décembre. On verra après pour l'application Android. J'ai d'abord pensé naturellement au Bluetooth puisqu'on l'avait étudié pendant les séances de TD. Je suis donc allé emprunter un module HC-05 à l'enseignant. Voici comment j'ai configuré le module Bluetooth: nom : pancar, PIN : 1234, adresse physique : 20:15:09:29:35:46. Malheureusement, l'implémentation du Bluetooth s'avère difficile côté PC. En Java il existe la bibliothèque bluecove (v 2.1.0) mais elle n'était pas compatibles avec mon PC. Ce problème de bibliothèque couplé aux différents embêtements qu'on a eu en TP lorsqu'on essayait de faire fonctionner le Bluetooth avec l'enseignant (problème Apple, Huawei...) m'on fait prendre la décision suivante : on n'utilisera pas le Bluetooth mais le wifi.

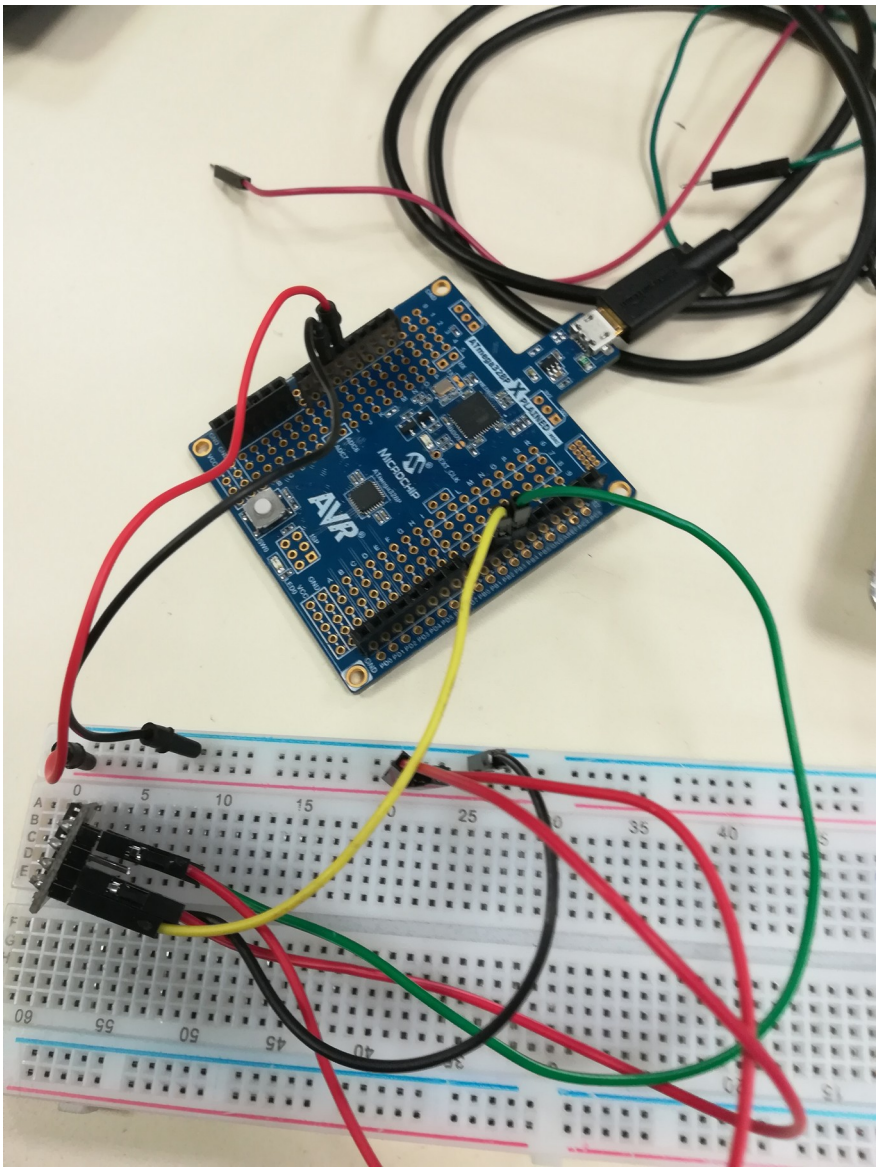
Pendant que j'essayais de faire fonctionner le wifi, Clara développait la classe Point pour Arduino.

L'avantage du wifi est qu'il suffit simplement d'utiliser les sockets pour communiquer avec la carte, ce qui est très facile à faire en Java. Pour le wifi, j'ai demandé à l'enseignant s'il avait à disposition le module wifi ESP8266 (modèle 1). Par chance il l'avait en réserve et j'ai donc pu commencer à travailler. Malheureusement ça se configure avec des commandes AT. https://f-leb.developpez.com/tutoriels/arduino/esp8266/debuter/.

Voici comment brancher le module ESP8266 (source : http://les-electroniciens.com/videos/arduino-ep16-installation-du-module-wifi-esp8266).



Après beaucoup de difficultés, je suis parvenu à brancher correctement le module wifi. En effet de nombreux tutoriels proposaient d'utiliser directement les pins séries (0 et 1) de l'Arduino, puis de communiquer directement avec le module. Il faut pour ça une bibliothèque spéciale que je n'arrivais pas à installer. Voici donc les bons branchements sur les pins 10 et 11.



Voici le programme pour envoyer des commandes AT au module wifi

```
#include <SoftwareSerial.h>

SoftwareSerial ESP8266(10, 11);

void setup() {
   Serial.begin(9600);
     ESP8266.begin(115200); // C'est la fréquence du ESP8266, j'ai eu beaucoup de difficultés à cause de ça
}

void loop() {
   if ( ESP8266.available() ) {
      Serial.write( ESP8266.read() );
   }
   if ( Serial.available() ) {
      ESP8266.write( Serial.read() );
   }
}
```

Voici les commandes AT pour envoyer une requête HTTP au module. Il faut d'abord connecter l'ordinateur au réseau wifi de la carte (**AI-THINKER_0649A7** chez moi). Vu qu'il n'y a pas de serveur DHCP, entrer manuellement les paramètres réseau : IP : 192.168.4.10, Passerelle : 192.168.4.1, Masque : 255.255.255.0, DNS : 8.8.8.8.
Commandes AT : https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/

Pour résumer, j'ai finalement réussi à faire passer un paquet HTTP (screenshot ci-dessus) après beaucoup de difficultés.

Il faudra maintenant mettre en place un protocole de communication entre mon « Pancake Drawer » Java sur PC et ma carte Arduino. Le problème est qu'il n'est pas possible à l'Arduino de faire 2 choses en même temps (ici écouter les ordres via wifi et faire autre chose en attendant). En effet le concept de **thread** n'existe pas en Arduino ; il existe bien une bibliothèque **protothread** mais ce n'est que du « simili-simultané » (j'ai essayé). Il va donc falloir gérer correctement l'attente et la réception des ordres.