ORJUBIN Alexis
RICO Jonathan

Université Toulouse III – Paul Sabatier

Professeur encadrant :
H. Leymarie - Electronique

M1 EEA ESET

Rapport de Projet - TER

Mise en place d'un TP sur les Unités de Mesure Inertielle

Année 2015-16

Sommaire

Présentation du projet

Présentation du sujet

Ce projet nous a été proposé par le professeur H. Leymarie, il porte sur la mise en œuvre d'une unité de mesure inertielle avec un micro-contrôleur compatible Arduino. L'intérêt de ce projet est d'être une première ébauche pour les TP d'étudiants en Master en alternance et en préparation agrégation de Physique. Il doit utiliser une approche ludique afin d'aider les étudiants à se familiariser avec les liaisons I2C, les accéléromètres/gyroscopes et l'environnement Arduino. Il doit également permettre à ces étudiants de développer une application utilisant une unité de mesure inertielle. Il a pour but de fournir une introduction sur le fonctionnement physique des capteurs, leur programmation et l'interprétation des données mesurées.

Travail demandé

Le cahier des charges est le suivant :

- Le projet doit être une application portable comportant un capteur d'accélération et de vitesse de rotation compatibles I2C.
- La réalisation physique doit être la plus simple possible pour être facilement reproductible : 9 exemplaires demandés.
- Afin d'assurer la tenue dans le temps des TP il faudra veiller à utiliser des composants trouvables facilement dans le commerce.
- Le projet abordera par étape la configuration de ces capteurs au cours d'un TP d'une durée de 4h : Il faudra dans un premier temps aborder la physique derrière ces capteurs, puis guider l'étudiant à travers la programmation de ces capteurs pour les aider à comprendre leur implémentation et à être capable de mettre en œuvre ces capteurs de par eux mêmes.
- Les sujets de TP ainsi que les 9 maquettes devront être opérationnels et prêt à l'emploi pour la date du 31 Mars.

Solutions apportées

Pour répondre à tous ces critères et profiter de l'occasion qui nous a été donnée de rendre ce TP ludique et passionnant plusieurs idées nous sont venues à l'esprit, voici celles qui ont été retenues :

- Fabrication d'un boîtier qui mesurera l'accélération linéaire et angulaire sur 3 axes et les transmettra par liaison sans fil à un récepteur rattaché à l'ordinateur de l'étudiant
- Réalisation d'un logiciel d'affichage (et enregistrement) graphique des données
- Mise en place d'expériences physiques afin de visualiser les accélérations sur les 3/6 axes et interprétation des résultats obtenus (réutilisation si possible de bancs de test de l'université)
- Codage d'un programme uC non fonctionnel, qui devra être complété par l'étudiant
- Conception d'un robot balancier pour démontrer une application des capteurs accéléromètriques

Répartition des tâches

Pour une meilleure efficacité, les tâches ont été réparties en fonction des compétences de chacun :

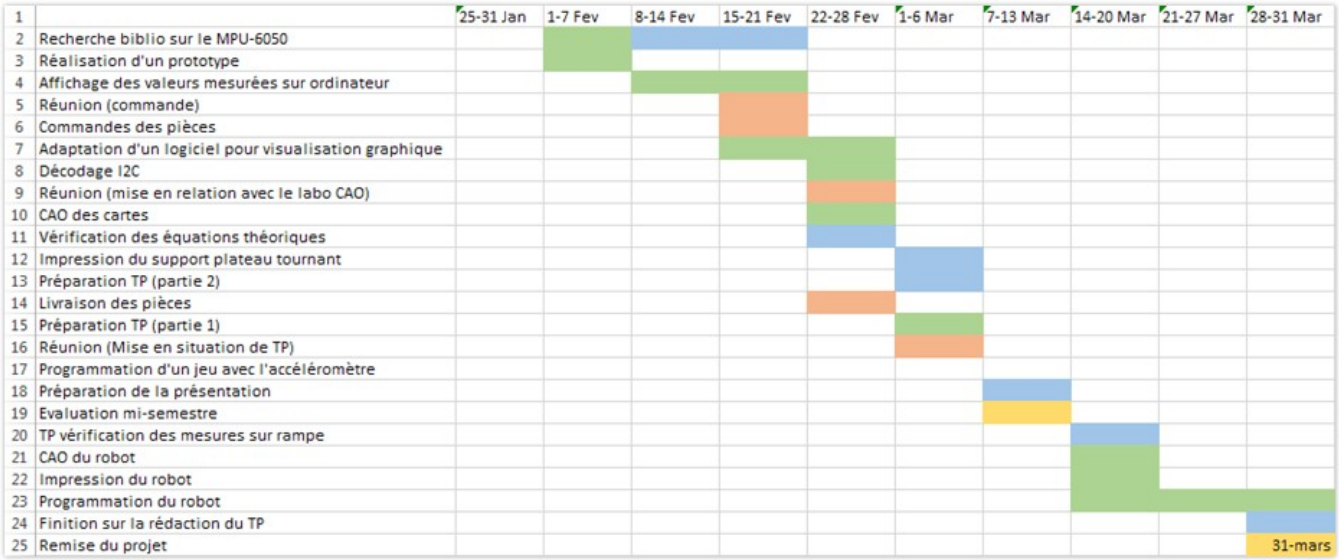
Ayant déjà travaillé sur la théorie des capteurs accéléromètriques, l'étude théorique du TP sera rédigée par A. ORJUBIN.

J.RICO se chargera de la partie programmation/implémentation.

J.RICO	A.ORJUBIN
TP implémentation / programmation	TP Etude statique / théorie d'opération
Affichage graphique	TP Etude dynamique
Conception des cartes	Gestion de projet / planification

Contrainte temporelle

Le projet devant être fini avant le 31 mars, il a fallu réaliser une planification des tâches relativement précise :



		25-31 Jan	1-7 Fev	8-14 Fev	15-21 Fev	22-28 Fev	1-6 Mar	7-13 Mar	14-20 Mar	21-27 Mar	28-31 Mar
2	Recherche biblio sur le MPU-6050										
3	Réalisation d'un prototype										
4	Affichage des valeurs mesurées sur ordinateur										
5	Réunion (commande)										
6	Commandes des pièces										
7	Adaptation d'un logiciel pour visualisation graphique										
8	Décodage I2C										
9	Réunion (mise en relation avec le labo CAO)										
10	CAO des cartes										
11	Vérification des équations théoriques										
12	Impression du support plateau tournant										
13	Préparation TP (partie 2)										
14	Livraison des pièces										
15	Préparation TP (partie 1)										
16	Réunion (Mise en situation de TP)										
17	Programmation d'un jeu avec l'accéléromètre										
18	Préparation de la présentation										
19	Evaluation mi-semestre										
20	TP vérification des mesures sur rampe										
21	CAO du robot										
22	Impression du robot										
23	Programmation du robot										
24	Finition sur la rédaction du TP										
25	Remise du projet										31-mars

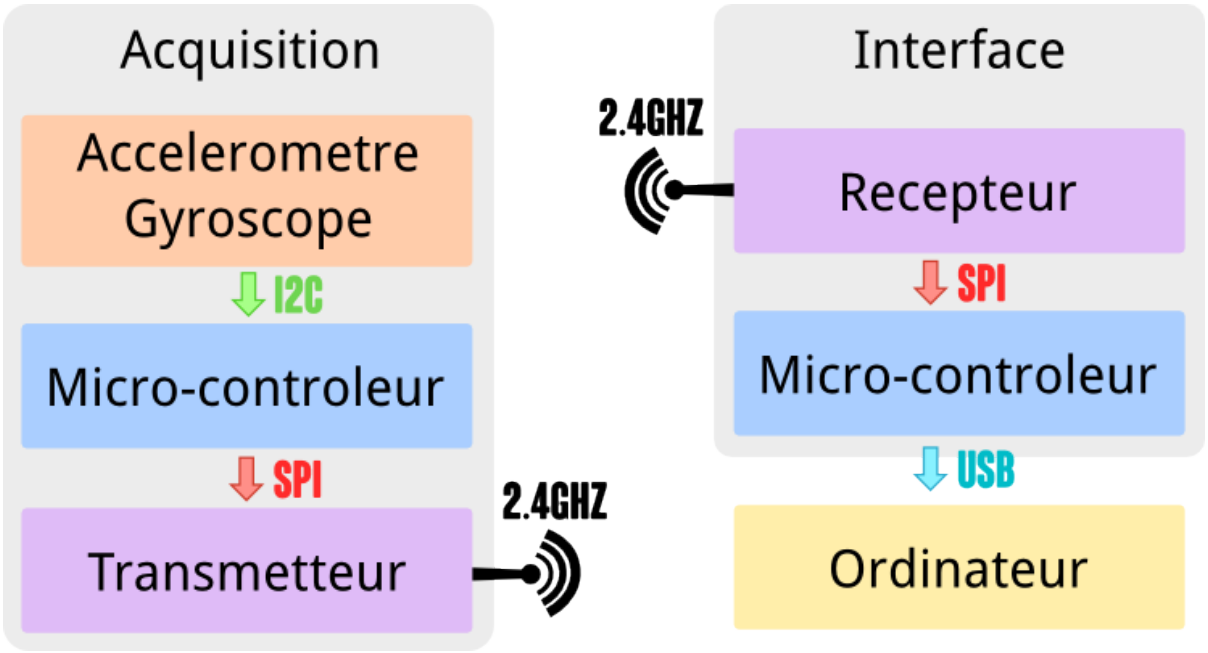
Développement des maquettes TP

Schéma fonctionnel

Selon le cahier des charges, la maquette TP devrait pouvoir faire une acquisition de données inertielles (accélération et vitesse de rotation) et la transmettre sans fil à un ordinateur afin d'avoir un affichage graphique des données.

On a donc une solution utilisant 2 cartes, une carte d'acquisition/transmission et une carte de réception/interface PC.

Le schéma fonctionnel du système est le suivant :



Pour une fabrication plus simple et plus rapide, le choix à été fait d'utiliser des modules pré-fabriqués réalisant chacun une des fonctions du système.

Dans la suite nous détaillons les différents modules utilisés :

Module Accéléromètre/Gyroscope

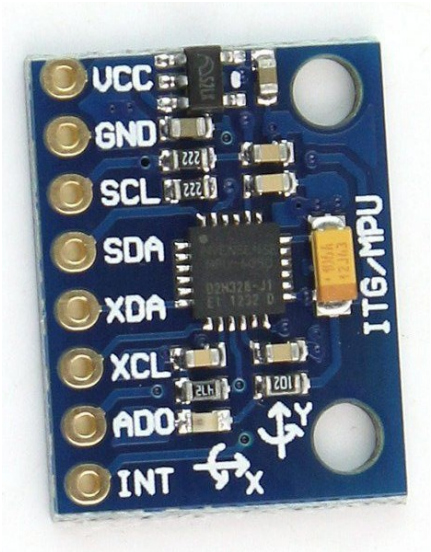
Pour des raisons de coût et de simplicité, nous avons choisi d'utiliser une puce intégrant un accéléromètre et un gyroscope sur le même substrat : Le MPU-6050 de Invensense.

Ce circuit intégré est interfaçable via I2C et fournit des mesures inertielles sur 16 bits signés. La sensibilité est réglable aussi bien sur l'accéléromètre que sur le gyroscope : de ± 2g à ± 16g et de ± 250°/s à ± 2000°/s

Le circuit fonctionne avec une alimentation de 3.3V et à des entrées compatibles TTL (5V).

Il existe des modules peu coûteux en provenance de Chine : disponibles sur des sites de vente comme *alibaba.com* et *ebay.com* à moins de 3$, ils intègrent ce circuit et un régulateur 3.3V avec les composants passifs requis.

Ces modules n'ont pas vraiment de référence ou de nom, nous avons choisi celui-ci :



Module micro-contrôleur

Le cahier des charges impose un système programmable par l'environnement de développement Arduino.

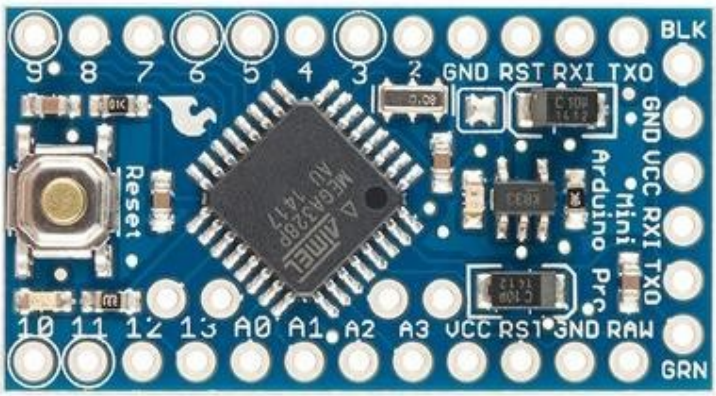
Nous avons donc choisi un module Arduino contenant le micro-contrôleur le plus commun (chez les cartes Arduino) : l'Atmega328 de chez Atmel.

Ce micro-contrôleur à les caractéristiques suivantes :

- Architecture RISC (famille AVR) 8 bits
- 32kB de memoire programme Flash
- 2kB de mémoire RAM
- Fonctionnement jusqu'à 20MHz
- 23 Entrées/Sorties
- Divers périphériques : ADC, I2C, SPI, UART etc.

Un des grands avantages du système *Arduino* est le bootloader : un programme de 0.5kB se charge de récupérer l'image binaire à flasher via le port série et auto-programme le micro-contrôleur. Il n'y a plus besoin d'un programmeur/linker supplémentaire.

Nous avons choisi le module *Arduino Pro Mini*, conçu et fabriqué par *Sparkfun* (mais lui aussi disponible sur *ebay* pour peu cher) : il intègre ce micro-contrôleur, un régulateur 5/3.3V et un crystal 16MHz sur un rectangle de 18 par 33mm



Module d'interface série

La carte *Arduino Pro Mini* ne contient pas d'interface USB, il faut donc utiliser une carte supplémentaire qui agit comme convertisseur Série → USB.

Le connecteur sur la carte *Arduino Pro Mini* est aussi connu sous le nom de connecteur FTDI (référence aux cables USB → Série contenant une puce fabriquée par *FTDI*). Il comporte les lignes TX et RX, une alimentation, une masse et une ligne de reset (*DTR*).

Nous avons donc choisi un module FTDI vendu sur le site *ebay.com* qui pourrait fonctionner en logique 3.3V (le système est alimenté en 3.3V) :

Modules de transmission/réception

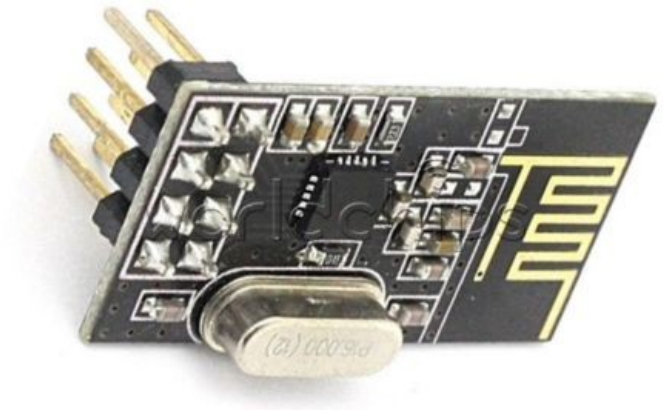
Nous avons choisi un circuit de transmission/réception (transceiver) 2.4GHz basse consommation, et qui disposait déjà de bibliothèques Arduino.

De plus la configuration est plus facile que les modules Xbee et se fait uniquement par programmation SPI (pas besoin de pré-programmer/associer les modules via un logiciel PC avant de pouvoir les utiliser).

Le circuit choisi est le *NRF24L01+* de *Nordic Semiconductor* :

- Solution low-cost à un seul IC, ne requiert pas de composants externes chers (seulement un crystal 16M et quelques passifs)
- Transmission/Réception GFSK sur la bande ISM 2.4GHz
- Interface SPI avec micro-contrôleur hôte
- Vitesse de transmission de 250kb/s, 1Mb/s et 2 Mb/s
- Très basse consommation : TX 11mA @ 0dBm, RX 13mA @ 2 Mbps

Nous avons choisi un module low-cost (moins de 3$) disponible sur *ebay* (basé sur un reference design de la datasheet du composant)

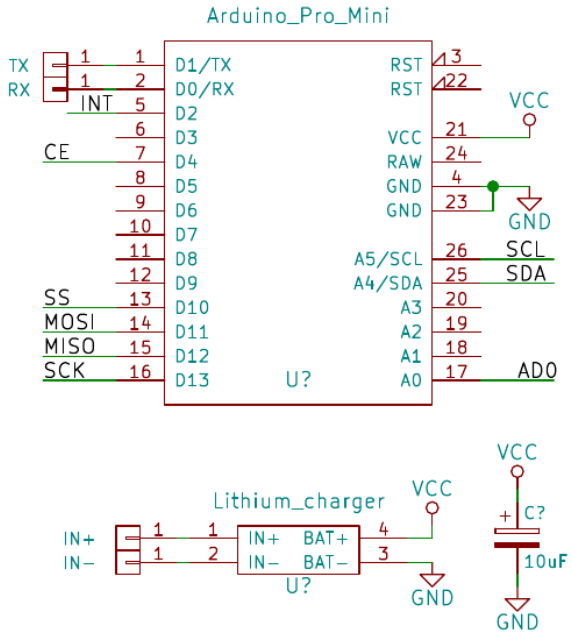


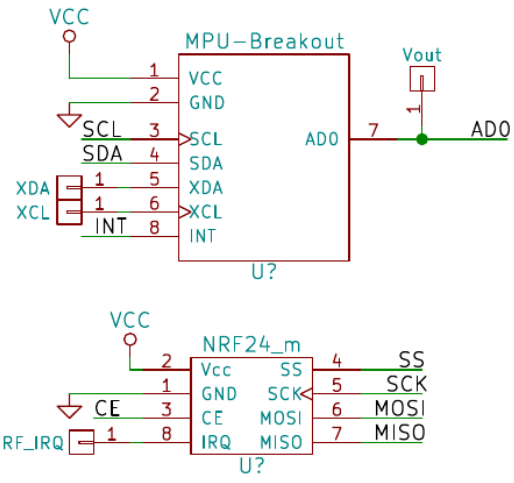
Conception des cartes Acquisition et Interface

Il a ensuite fallu réaliser une carte "hôte" pour accueillir et interconnecter ces modules afin de réaliser l'acquisition des données inertielles, la transmission et l'interfaçage avec l'ordinateur de l'étudiant.

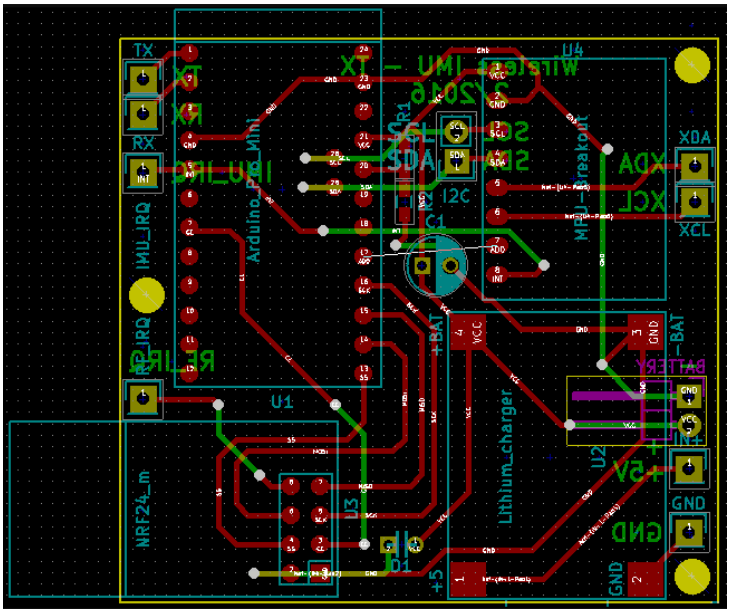
Nous avons utilisé le logiciel libre de CAO pour circuits imprimés *KiCad* pour dessiner le schéma et effectuer le routage des cartes.

Chaque carte *Acquisition* comporte un module *Arduino*, un module de mesure inertielle, un module de transmission 2.4GHz, un module de charge lithium-ion et une batterie LiPo.

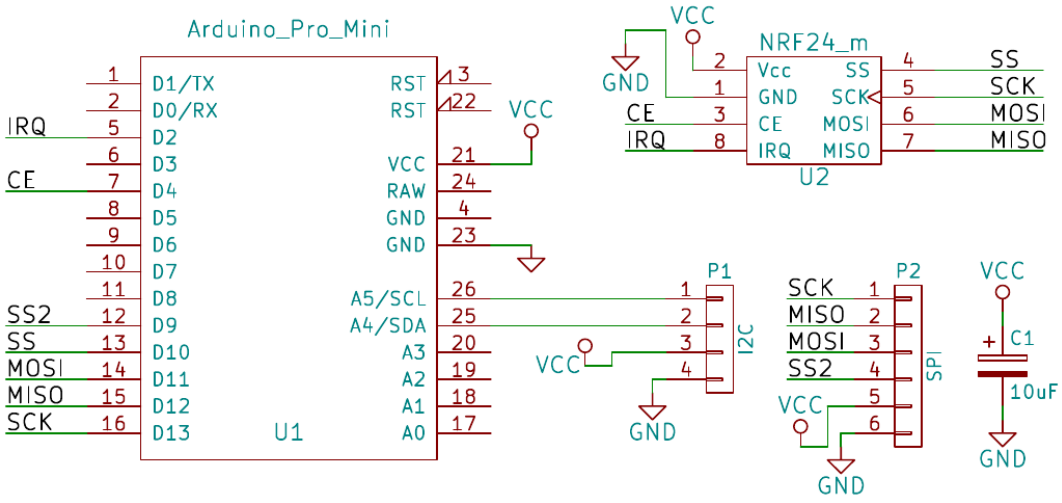




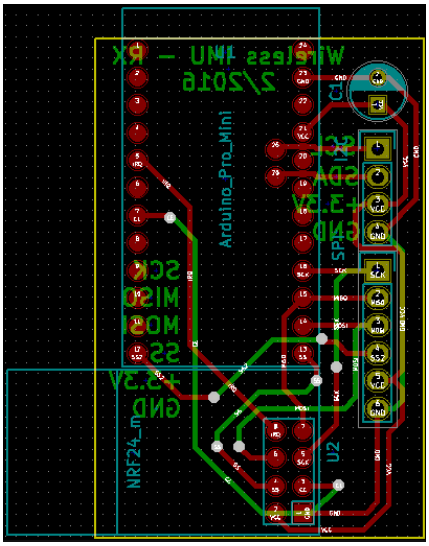
Grâce aux faibles dimensions des modules, la carte *Acquisition* tient sur un carré de 5cm de coté (à l'exception de l'antenne du module de transmission)



La carte *Interface* est encore plus petite, contenant seulement un module *Arduino* , un module de réception 2.4GHz et quelques connecteurs :



Pour gagner en rapidité lors de l'assemblage et la soudure des modules, le choix a été fait de monter les modules en surface, au lieu de devoir souder des connecteurs mâles/femelle pour chaque module. De plus avec cette méthode, les cartes ont un profil plus fin.



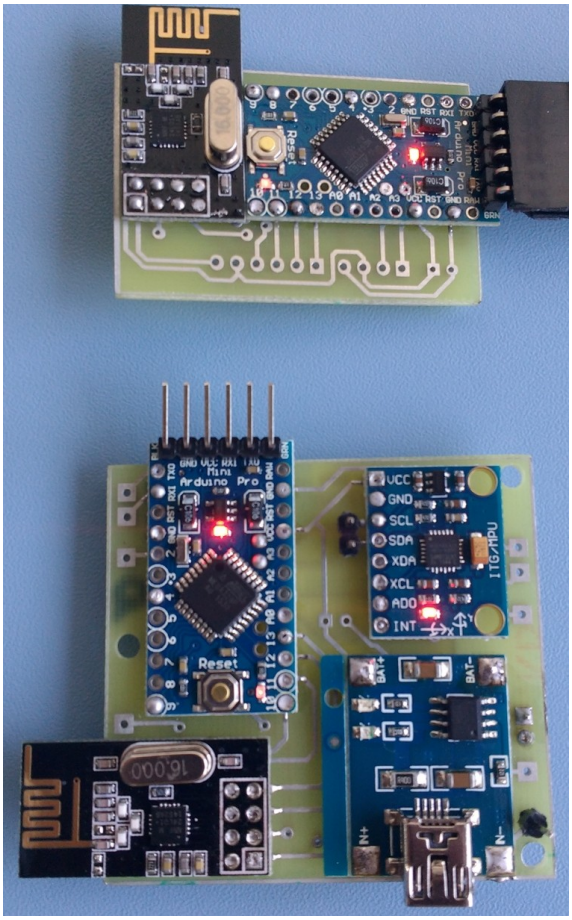
Fabrication des cartes

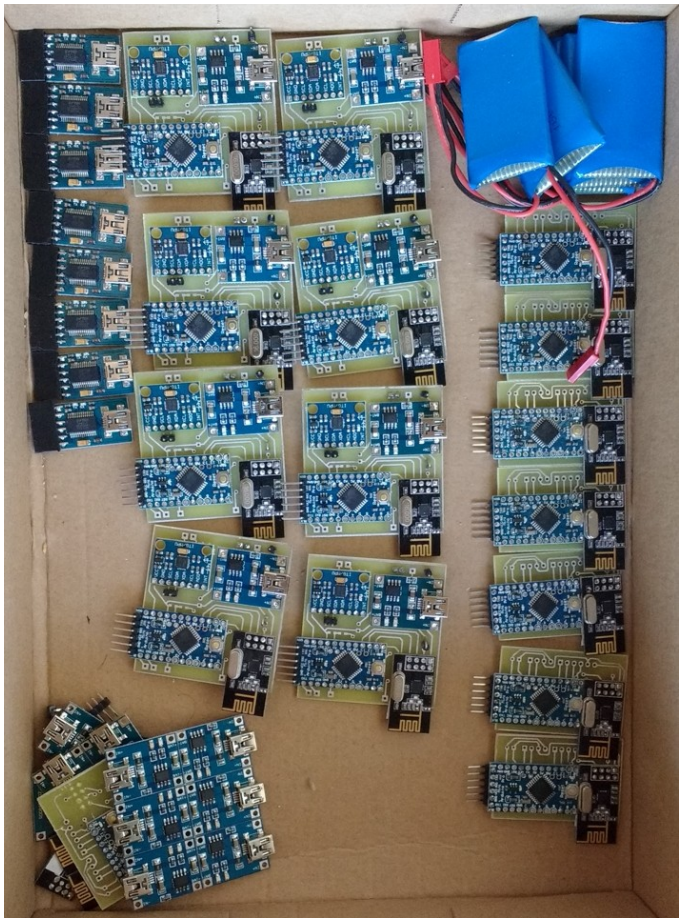
La fabrication des circuits imprimés à été prise en charge par le service d'électronique de l'université : Nous avons fourni les fichiers *Gerber* (standard industriel) et reçu les 2 prototypes deux jours plus tard.

Il a ensuite fallu souder les modules (et tester le système de montage en surface), vérifier le fonctionnement, rédiger le sujet de TP en fonction de ces cartes, tester le TP en présence de l'enseignant encadrant et enfin passer à la production finale.

Pour la production finale, les cartes ont été réalisées en double-couche avec trous métallisés mais sans silkscreen (pour réduire les coûts de production), d'où les inscriptions directement en cuivre.

Les cartes ont ensuite été testées une par une (acquisition, transmission, interface) puis fournies à l'enseignant.



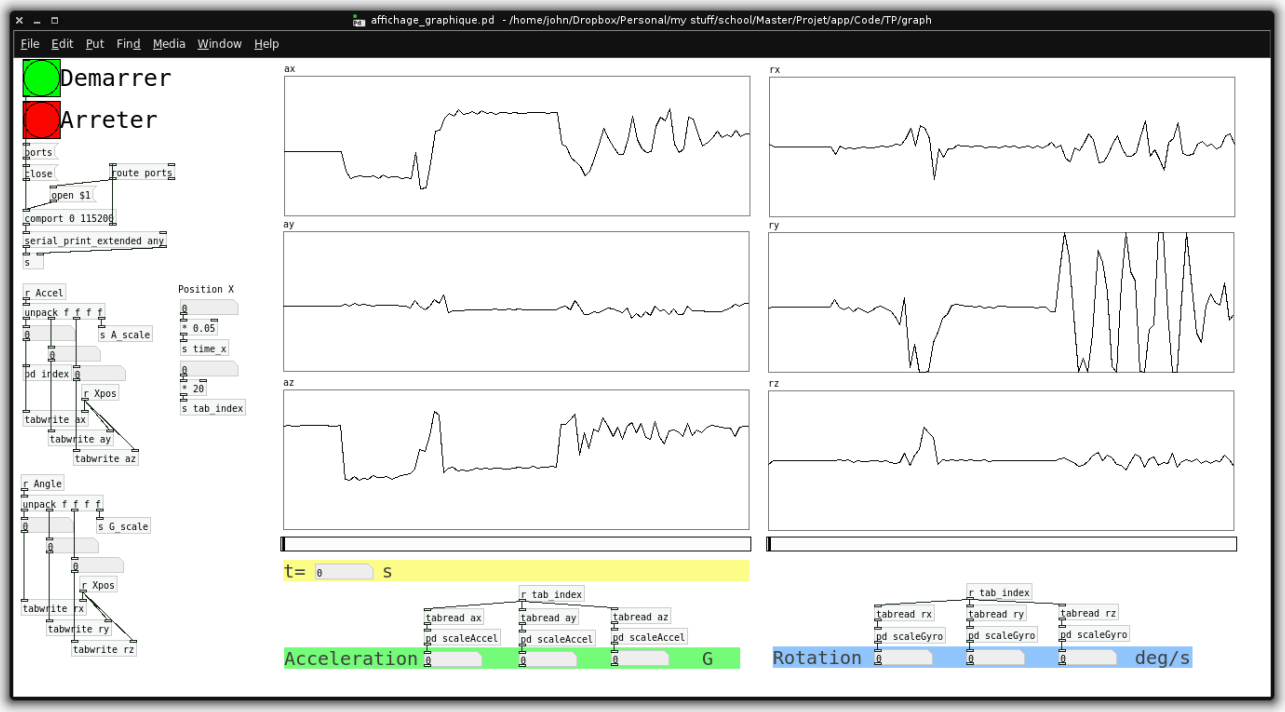


Développement logiciel

Programme d'affichage graphique

L'affichage graphique a été réalisé via le logiciel *PureData* (open-source), qui est un logiciel de programmation "visuelle" destiné au domaine artistique.

Les données inertielles (sans unité) sont récupérées du port série au format ASCII toutes les 50ms, converties en unités réelles puis affichées, une fenêtre graphique par axe :



L'interface est relativement simple d'utilisation : on démarre/arrête l'acquisition en cliquant sur le bouton vert/rouge, et on lit les valeurs numériques en déplaçant un curseur sous les axes

Programme d'acquisition / transmission

Le programme d'acquisition et de transmission (micro-contrôleur sur la carte *Acquisition*) suit une séquence d'initialisation puis envoie en boucle les données inertielles mesurées.

Le développement a été assez rapide grâce aux bibliothèques *Arduino* disponibles pour le module accéléromètre/gyro et le module de transmission.

Description rapide du programme (disponible en annexe) :

Début

Initialisation périphérique I2C

Initialisation port série (115.2kbaud)

Init. de l'accel/gyro, réglage de la sensibilité des capteurs

Init. du transceiver en mode transmission

Début de boucle

Mesure du temps avant la mesure inertielle

Récupération des valeurs des capteurs accel/gyro

Mise à jour de la structure de données a envoyer avec les nouvelles valeurs

Envoi de la structure de données via le transmetteur 2.4GHz

Attente de la fin des 50ms après la mesure

Fin de boucle

Programme de réception / interface

Le programme de réception des données qui tourne sur le uC de la carte *Interface* se contente juste de récupérer les données via le transceiver configuré cette fois-ci en réception et à les envoyer sur le port série en ASCII (simple ligne *printf*).

Rédaction du sujet de TP

Points à aborder pendant le TP

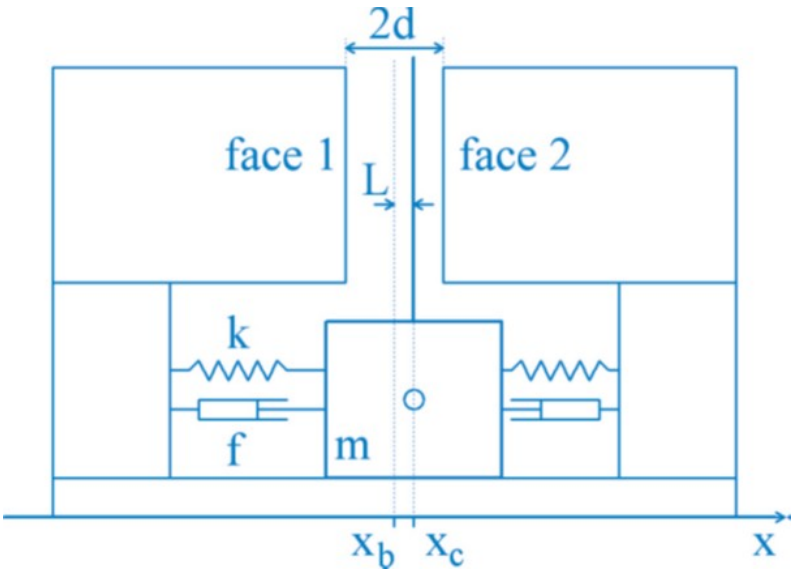
Au cours des réunions avec notre enseignant de projet nous avons établi les points à aborder dans nos sujets de TP. Nous avons pu décider quelles parties du capteur devraient être expliquées en détail et quelles autres données devraient être évoquées juste à titre informatif afin de respecter la durée imposée d'un TP de 4 heures.

Dans cette situation il devient primordial d'être concis et précis pour aborder l'essentiel et donner à l'étudiant les outils pour pouvoir configurer par la suite un capteur numérique de manière totalement autonome. Voici une synthèse des impératifs évoqués lors des réunions .

L'étudiant devra :

- Être familiarisé avec les liaisons/protocole I2C
- Être capable d'interpréter les données brutes en sortie du capteur
- Comprendre la méthode physique utilisée pour mesurer une accélération linéaire/angulaire
- Comprendre le code de configuration du MPU-6050 et être capable de l'adapter à ses besoins
- Être capable de mettre en œuvre une unité de mesure inertielle numérique dans une application réelle

Théorie de fonctionnement d'un accéléromètre MEMS



Modélisation d'un MEMS sur l'axe x

- xb : position au repos (par rapport au support)
- xc : position centrale
- k : raideur
- f : coefficient de frottement
- m : masse

L'équation du mouvement donne la relation entre le déplacement $L = x_c - x_b$ et l'accélération est obtenue en appliquant le principe fondamental de la mécanique dans le référentiel galiléen du sol :

$$m\vec{a}_c = \sum \vec{F}_{appliquées}$$

En projection sur l'horizontale (axe x) :

$$m\frac{d^2x_c}{dt^2} = -2k(x_c - x_b) - 2f\frac{d(x_c - x_b)}{dt}$$

Avec $x_c = (x_c - x_b) + x_b = L + x_b$, on obtient :

$$m\frac{d^2L}{dt^2} + m\frac{d^2x_b}{dt^2} = 2kL - 2f\frac{dL}{dt}$$

Où $\frac{d^2x_b}{dt^2} = a(t)$ est l'accélération du bâti de l'accéléromètre par rapport au sol que l'on veut mesurer.

Si l'on préfère, on peut raisonner dans le référentiel non galiléen du bâti en mouvement par rapport au sol : l'élongation *L* du ressort s'identifiant ici à la position du mobile par rapport au bâti, il vient en appliquant au mobile le théorème de la résultante cinétique dans le référentiel du bâti (donc, en tenant compte de la « force d'inertie d'entraînement » $-ma(t)$ qui s'ajoute aux autres forces physiques) :

$$m\frac{d^2L}{dt^2} = -2kL - 2f\frac{dL}{dt} - ma(t)$$

On obtient donc l'équation de mouvement en élongation :

$$\frac{d^2L}{dt^2} + \frac{2f}{m}\frac{dL}{dt} + \frac{2kL}{m} = -a(t)$$

L'analyse harmonique de la partie mécanique du capteur se fait sur l'équation complexe associée (transformation $\frac{d}{dt} \to j\omega$). On obtient :

$$(\omega_0^2 - \omega^2 + 2j\mu\omega_0\omega)\underline{L} = -\underline{a(t)} \to \frac{\underline{L}}{\underline{a}} = \frac{-\frac{m}{2k}}{1 + 2j\mu\frac{\omega}{\omega_0} + \left(j\frac{\omega}{\omega_0}\right)^2}$$

Avec la pulsation caractéristique $\omega_0^2 = \frac{2k}{m}$ et le paramètre d'amortissement $\mu = \sqrt{\frac{f^2}{2km}}$.

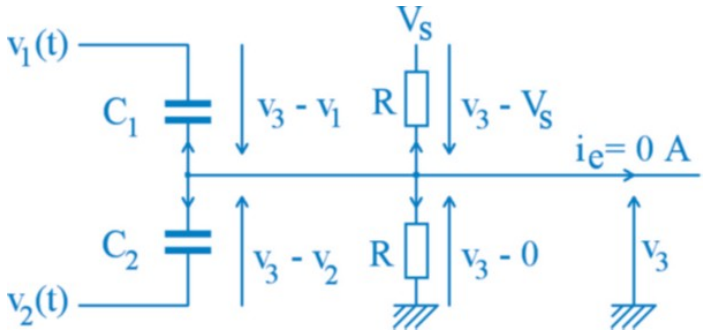
La fonction de transfert $\frac{\underline{L}}{\underline{a}}$ est de type passe-bas du second ordre.

Mesurer le déplacement *L* des lamelles du capteur différentiel revient donc à mesurer l'accélération du support, pourvu qu'elle soit constante ou lentement variable (c'est-à-dire avec $\omega \ll \omega_0$).

On a, en régime établi :

$$L \approx -\frac{m}{2k}a(t)$$

En passant l'analyse du MEMS comme un pont capacitif :



Conditionnement en pont du capteur différentiel de position

De façon synthétique, la tension de sortie se met sous la forme générale :

$$V_{out} = V_0 + S \times a$$

La tension de sortie de l'accéléromètre se présente « classiquement » sous une forme affine où V_0 est la tension « de repos » lorsque l'accélération à mesurer est nulle et *S* est la sensibilité de l'accéléromètre en $V \cdot m^{-1} \cdot s^2$.

$S' = S/g$ est la sensibilité en « V/g » (avec évidemment $g = 9{,}8\, m \cdot s^{-2}$).

Implémentation du capteur inertiel MPU-6050

Cette première partie du TP sert à introduire l'étudiant au système de développement *Arduino* et l'aider à implémenter le capteur inertiel *MPU-6050* de chez Invensense en mode communication I2C.

L'étudiant est guidé par étapes :

- Prise en main de l'environnement de développement *Arduino IDE*
- Installation des bibliothèques requises pour le transmetteur et l'IMU
- Test de la maquette et de la programmation du uC : clignottage d'une led
- Lecture de datasheet et modification du code avec l'adresse I2C du capteur
- Utilisation de la datasheet pour convertir les valeurs en unités réelles
- Mise en œuvre du code fourni pour la transmission sans fil et l'affichage graphique des données
- Visualisation de l'effet qu'ont les différentes sensibilités du gyroscope
- Décodage de l'adresse I2C du capteur avec un oscilloscope
- Analyse d'un relevé d'analyseur logique (trames I2C en hexadecimal dans un fichier CSV) et reconstruction des valeurs décimales transmises par le capteur au uC

Interprétation des mesures et étude physique

Au travers de plusieurs manipulations, l'étudiant apprend à utiliser et interpréter les mesures du capteur :

Il y a 3 étapes majeures : une étude statique, une étude dynamique sur rampe et une étude sur un tourne disque.

Étude statique :

- Orientation du capteur (carte *Acquisition*) selon les 3 axes X, Y et Z et correspondance des valeurs

Étude dynamique sur rampe :

- En utilisant le principe fondamental de la statique, mesure et détermination de l'inclinaison de la rampe
- "Chute" de la carte *Acquisition* sur la rampe : interprétation graphiques des différentes étapes de la chute/glissade
- Détermination par le calcul des coefficients de frottement de différentes matières (placées entre la carte et la rampe)

Étude sur le tourne-disque :

- Mesure des vitesses de rotation en deg/s avec les différentes vitesses
- Détermination de la vitesse en tours/min et comparaison avec la vitesse réglée
- Explication de la différence observée entre la vitesse mesurée et réglée

Les sujets de TP sont inclus en fin d'Annexe

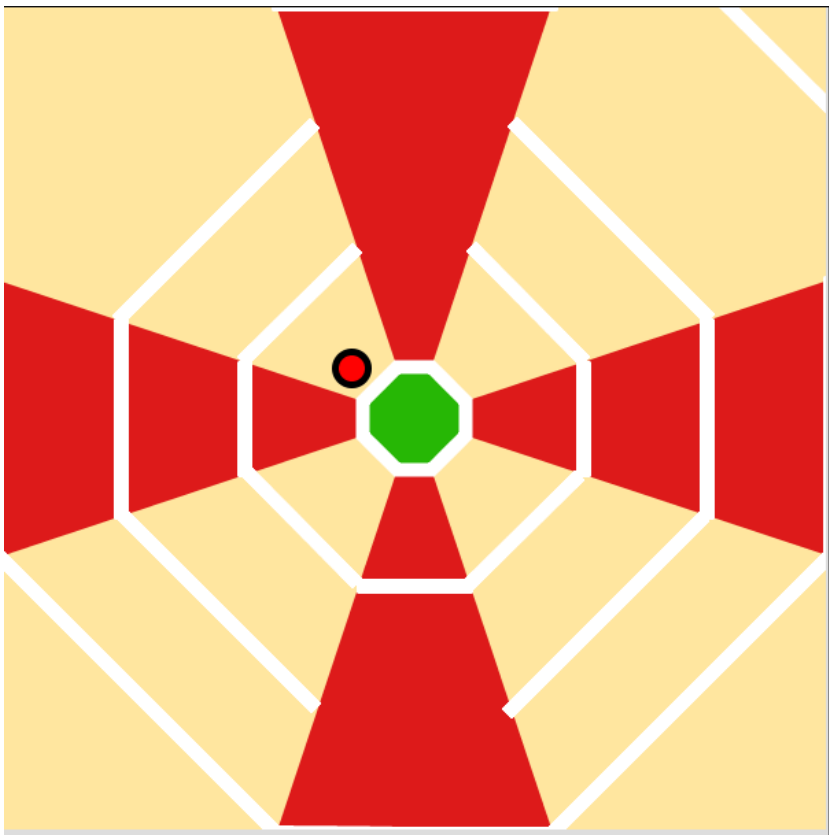
Réalisations supplémentaires

Jeu d'adresse

Comme le robot qui devait être développé ne l'a pas pu être à temps, nous avons trouvé un autre moyen de rendre le TP plus ludique : nous avons programmé un jeu d'adresse (clone de *SuperHexagon*).

Le principe est simple : l'étudiant doit bouger un point rouge pour sortir d'un labyrinthe octogonal sans toucher les murs en orientant la carte *Acquisition* dans le sens voulu.

Comme le jeu doit être réactif, le code d'acquisition/transmission a du être légèrement modifié afin de permettre une transmission sous les 20ms. Le code de la partie interface a lui aussi du être modifié pour des raisons de simplification du code du jeu : les données ne sont plus transmises en ASCII mais directement en binaire.



Robot balancier a pendule inversée

Finalement, nous avons aussi pu réaliser (quelques jours avant de rédiger ce rapport) le robot balancier que nous voulions produire au début comme démonstration.

Le corps du robot a été modélisé sous *SolidWorks* autour des dimensions des moteurs et des composants. Le corps est modulaire et est composé de plusieurs pièces qui s'emboîtent les unes dans les autres.

Les pièces ont été imprimées en 3D (matériau PLA) au *FabLab* de l'université.

L'électronique est quasiment la même que la carte *Acquisition* à la différence d'un pont en H pour le contrôle moteur (L293) et d'un régulateur boost pour l'utilisation de ce pont en H (fonctionnement 5V).

A l'heure de l'écriture du rapport, le robot arrive à se balancer mais la boucle de contrôle PID n'est pas encore optimisée (oscillation, etc.)

Le programme du robot fonctionne comme ceci :

Début

Initialisation des capteurs

Réglage du zéro des capteurs : position actuelle = position recherchée

Début de boucle

Lecture des capteurs

Détermination de l'angle du robot avec l'accéléromètre (2 axes)

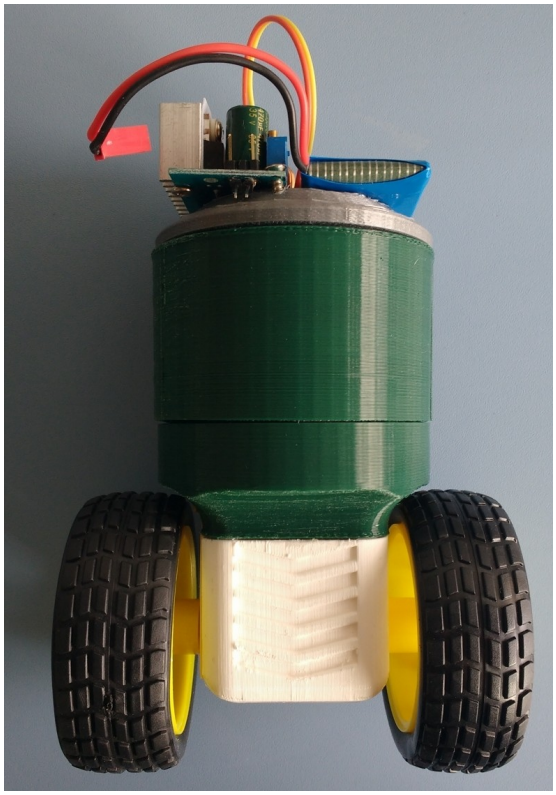
Détermination de la vitesse de rotation sur lui-même du robot (gyro 1 axe)

Filtrage Kalman avec ces 2 valeurs

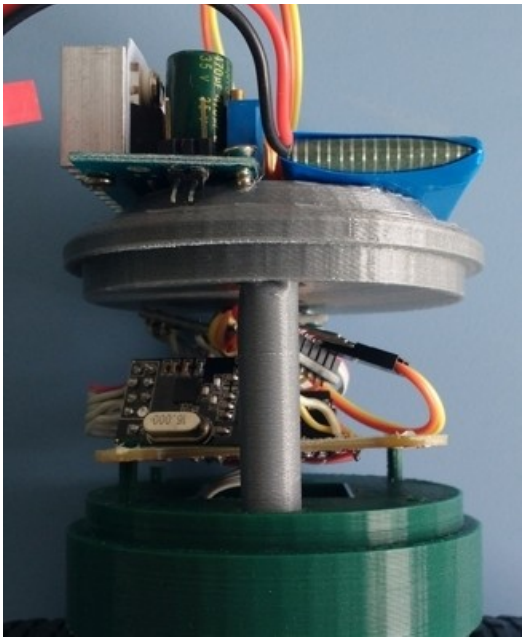
Mise a jour du PID

Activation des moteurs avec la commande de la boucle PID

Fin de boucle



Le robot a un double-capot à fermeture magnétique (pièces vertes sur le render) pour pouvoir accéder aux composants et programmer le uC.



Le robot contient les composants suivants :

- Régulateur boost (en haut a gauche)
- Batterie LiPoly (bleue en haut a droite)
- uC Arduino (vers le haut au milieu)
- Module de transmission réception (en noir)
- Carte pont en H (tout en bas)
- Module Accel/Gyro : non visible, vissé en haut du corps du robot (pièce grise)

Conclusion

Pour finir nous souhaitons remercier notre enseignant encadrant H. Leymarie de nous avoir fourni un sujet aussi passionnant et permis de réaliser un TP destiné à des étudiants comme nous.

Nous avons beaucoup appris de cette expérience, comme se mettre à la place de l'enseignant pour la réalisation des TP, nous en savons également plus au sujet des capteurs d'accélération numérique mais aussi au sujet des différentes étapes de conception de notre produit. L'aspect théorique, le dimensionnement des composants, l'étude numérique, ou même l'aspect financier sont autant d'aspects intéressants dont nous avons eu l'occasion de survoler lors de notre projet, ce qui nous permet d'entrevoir comment un projet peut être mené au sein d'une équipe professionnelle d'ingénieurs.

Bibliographie – Remerciements

Ce projet n'aurait pas pu être réalisé sans les personnes suivantes, nous tenons à les remercier pour leur aide précieuse :

- *ARMAJACH Guillaume* et *PETILLON Fanny*, sous la direction du *Pr. Laurent BERQUEZ*, qui ont fourni l'important travail préliminaire dont ce TP est très largement inspiré
- *Mr. LIBERT* pour l'accès à la salle de TP dont s'est inspiré notre étude théorique et pour le prêt du matériel utilisé dans l'étude théorique
- *Mr. BOITIER* pour la rapidité et la facilitation de la commande des composants
- *Mr. Guillaume MAFFRE* et son équipe du Service Commun d'Électronique pour sa réactivité et la rapidité à laquelle il a réalisé les cartes électroniques et leur prototypes

De plus, le code utilisé dans ce projet (et TP) a été plus rapide à réaliser grâce à l'utilisation de diverses bibliothèques disponibles sur Internet :

- Arduino_Pd, écrit par *Alexandros Drymonitis* :
  https://github.com/alexdrymonitis/Arduino_Pd
- MPU6050/I2CLib, écrit par *Jeff Rowberg* :
  https://github.com/jrowberg/i2cdevlib/
- RF24, écrit/modifié par *TMRh20*
  http://tmrh20.github.io/RF24/index.html

Annexes

Code TP n°1 :
Acquisition mesures inertielles, à compléter par les étudiants

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

#define IMU_ADDRESS 0x55 // Ecrire l'adresse trouvée à la place du 0x55

MPU6050 imu(IMU_ADDRESS);

int16_t ax, ay, az;
int16_t rx, ry, rz;

char serial_line[100] = {0};

void setup() {
  Wire.begin(); // Initialisation périphérique I2C

  Serial.begin(115200); // Initialisation port serie

  imu.initialize(); // Init. acceleromètre/gyroscope

  // Reglage de la sensibilité
  // imu.setFullScaleGyroRange(1);
  // imu.setFullScaleAccelRange(0);
}

void loop() {
        imu.getMotion6(&ax, &ay, &az, &rx, &ry, &rz); //Recuperation des donnees
inertielles

  // Calcul de l'acceleration lineaire et angulaire en fonction de la sensibilité
  // calcAccel(&ax, &ay, &az, imu.getFullScaleAccelRange());
  // calcAngle(&rx, &ry, &rz, imu.getFullScaleGyroRange());

  //Mise en forme de la trame a envoyer sur le port serie
  sprintf(serial_line, "Accel %d %d %d | Gyro %d %d %d\n", ax, ay, az, rx, ry, rz);
  Serial.print(serial_line); //Ecriture de la trame sur le port serie

  delay(100);
}

```

```
41 void calcAccel(int16_t* X, int16_t* Y, int16_t* Z, uint8_t scale) {
42   // Calcul de l'acceleration en milli-G
43   float valeur_lsb = 32768;
44
45
46 }
47
48 void calcAngle(int16_t* X, int16_t* Y, int16_t* Z, uint8_t scale) {
49     // Calcul de l'acceleration angulaire en centiemes de degres/s
50     float valeur_lsb;
51
52     // Changer les valeurs "valeur_lsb" en fonction de la datasheet
53     switch (scale) {
54         case 0:
55           valeur_lsb = 0;
56           break;
57         case 1:
58           valeur_lsb = 0;
59           break;
60         case 2:
61           valeur_lsb = 0;
62           break;
63         case 3:
64           valeur_lsb = 0;
65           break;
66         default:
67           valeur_lsb = 0;
68     }
69
70     *X = (long)*X * 100 / valeur_lsb;
71     *Y = (long)*Y * 100 / valeur_lsb;
72     *Z = (long)*Z * 100 / valeur_lsb;
}
```

Code TP Acquisition et Transmission

```
#include "I2Cdev.h"
#include "MPU6050.h"
#include <string.h>
#include <SPI.h>
#include "RF24.h"
#include "Wire.h"

#define SAMPLE_INTERVAL 50000

MPU6050 imu;

int16_t ax, ay, az;
int16_t rx, ry, rz;
unsigned long time_u;
char serial_line[100] = {0};

byte addresses[][6] = {"1Node","2Node"};

struct dataStruct{
  int ax;
  int ay;
  int az;
  int accelRange;
  int rx;
  int ry;
  int rz;
  int gyroRange;
}myData;

RF24 radio(4,10);

void setup() {
    Wire.begin(); // Initialisation I2C

    Serial.begin(115200); // Init. port serie

    // Init. de l'accelerometre/gyroscope MEMS
    imu.initialize();
    imu.setFullScaleGyroRange(0); // Changer la sensibilite ici
    imu.setFullScaleAccelRange(0);
    myData.accelRange = imu.getFullScaleAccelRange();
    myData.gyroRange  = imu.getFullScaleGyroRange();

    radio.begin(); // Init. du transceiver
```

```
45      radio.setPALevel(RF24_PA_LOW); // Demarrage a faible puissance
46      radio.openWritingPipe(addresses[0]); // Ouverture du canal de
47  transmission
48      radio.openReadingPipe(1,addresses[1]); // Ouverture du canal de reception
49      radio.stopListening(); // Arret de la reception
50  }
51
52  void loop() {
53      time_u = micros(); // Mesure du temps avant la mesure inertielle
54      imu.getMotion6(&ax, &ay, &az, &rx, &ry, &rz); // Mesure inertielle
55
56      // Calcul des valeurs reeles
57      // calcAccel(&ax, &ay, &az, imu.getFullScaleAccelRange());
58      // calcAngle(&rx, &ry, &rz, imu.getFullScaleGyroRange());
59
60      // Mise a jour de la structure de donnees a envoyer
61      myData.ax = ax; myData.ay = ay; myData.az = az;
62      myData.rx = rx; myData.ry = ry; myData.rz = rz;
63
64      // Trame serie Pure Data (lorsqu'on a pas de carte RX)
65      // sprintf(serial_line, "Accel %d %d %d %d\n",
66      //      myData.ax, myData.ay, myData.az, myData.accelRange);
67      // Serial.print(serial_line);
68
69      // sprintf(serial_line, "Angle %d %d %d %d\n",
70      //      myData.rx, myData.ry, myData.rz,  myData.gyroRange);
71      // Serial.print(serial_line);
72
73      // Envoi de la structure de donnees (env. 40ms)
74      radio.write(&myData, sizeof(myData));
75      // Verification que le temps entre chaque mesure = 50ms
76      while((micros() - time_u) < SAMPLE_INTERVAL);
77  }
78
79  void calcAccel(int16_t* X, int16_t* Y, int16_t* Z, uint8_t scale) {
80          uint16_t sensibilite = 32768;
81
82          sensibilite = sensibilite >> (scale+1);
83          *X = (long)*X * 1000 / sensibilite;
84          *Y = (long)*Y * 1000 / sensibilite;
85          *Z = (long)*Z * 1000 / sensibilite;
86  }
87
88  void calcAngle(int16_t* X, int16_t* Y, int16_t* Z, uint8_t scale) {
89      float sensibilite = 32768;
90
91      switch (scale) {
```

```
92            case 0:
93              sensibilite = 131;
94              break;
95            case 1:
96              sensibilite = 65.5;
97              break;
98            case 2:
99              sensibilite = 32.8;
100             break;
101           case 3:
102             sensibilite = 16.4;
103             break;
104           default:
105             sensibilite = 0; // valeur sensibilite non conforme
106       }
107
108       *X = (long)*X * 100 / sensibilite;
109       *Y = (long)*Y * 100 / sensibilite;
110       *Z = (long)*Z * 100 / sensibilite;
  }
```

Code TP Réception et Interface

```
#include <SPI.h>
#include "RF24.h"
#include <string.h>

byte addresses[][6] = {"1Node","2Node"}; // Adresses des 2 transceivers
char serial_line[100] = {0}; // Buffer pour la communication serie

// Structure de donnees contenant les
// mesures inertielles qui vont etre recues
struct dataStruct{
  int ax;
  int ay;
  int az;
  int accelRange;
  int rx;
  int ry;
  int rz;
  int gyroRange;
}myData;

RF24 radio(4,10); // Declaration du cablage du transceiver

void setup() {
  Serial.begin(115200); // Activation de la transmission serie

  radio.begin(); // Initialisation du transceiver
  radio.setPALevel(RF24_PA_LOW); // Demarrage a faible puissance
  radio.openWritingPipe(addresses[1]); // Ouverture du canal de
transmission
  radio.openReadingPipe(1,addresses[0]); // Ouverture du canal de reception
  radio.startListening(); // Demarrage de la reception
}

void loop() {
  if(radio.available()){ // Des qu'un paquet de donnees est recu
    while (radio.available()) {            // Tant qu'on recoit des paquets
      radio.read(&myData, sizeof(myData)); // Remplir la structure myData
    }

    // Transmission serie des mesures inertielles recues
    sprintf(serial_line, "Accel %d %d %d %d\n",
            myData.ax, myData.ay, myData.az, myData.accelRange);
    Serial.print(serial_line);
```

```
44
45      sprintf(serial_line, "Angle %d %d %d %d\n",
46              myData.rx, myData.ry, myData.rz,  myData.gyroRange);
47      Serial.print(serial_line);
48    }
  }
```

Première Partie : Programmation et communication I2C

I. Installation et prise en main du logiciel

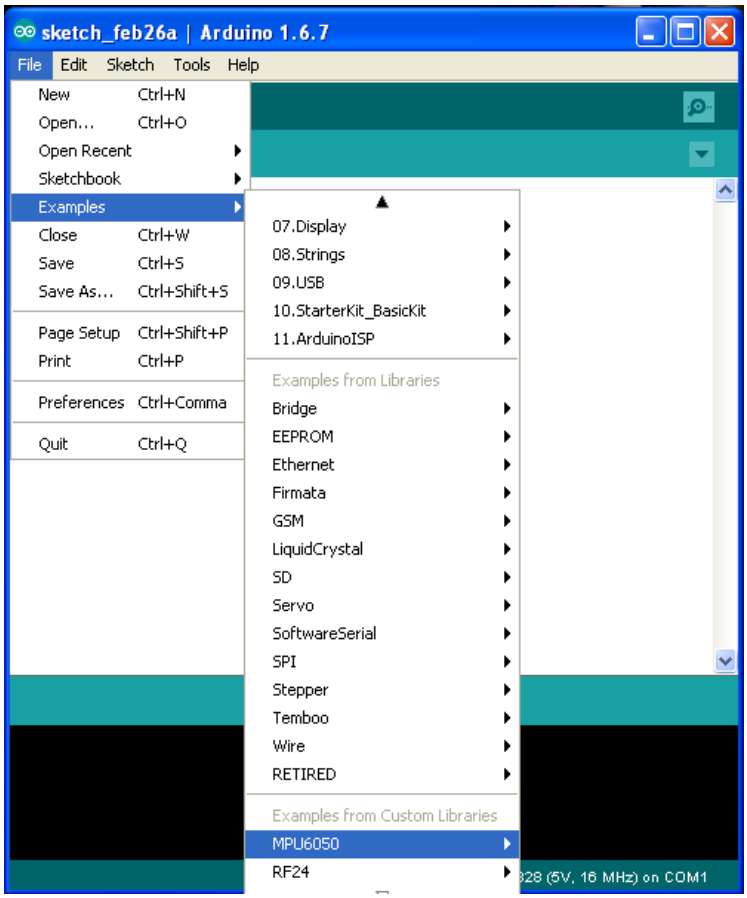
Pour pouvoir programmer le micro-contrôleur (Atmega328) qui est sur la carte IMU, on utilise l'IDE (environnement de développement intégré) et les bibliothèques Arduino, ainsi que deux autres bibliothèques open-source pour pouvoir communiquer avec l'accéléromètre/gyroscope (I2C) et le transmetteur sans fil (SPI).

On va commencer par télécharger et installer (si ce n'est pas encore fait) l'Arduino IDE :

- Aller sur http://arduino.cc/ et cliquer sur l'onglet *Download* puis sur *Windows Installer,* puis sur *Just download*.
- Installer puis lancer le logiciel, confirmer toutes les boites de dialogue qui s'ouvrent
- Une fois lancé, fermer l'Arduino IDE, un dossier *Arduino* devrait avoir apparu dans le dossier *Mes Documents*

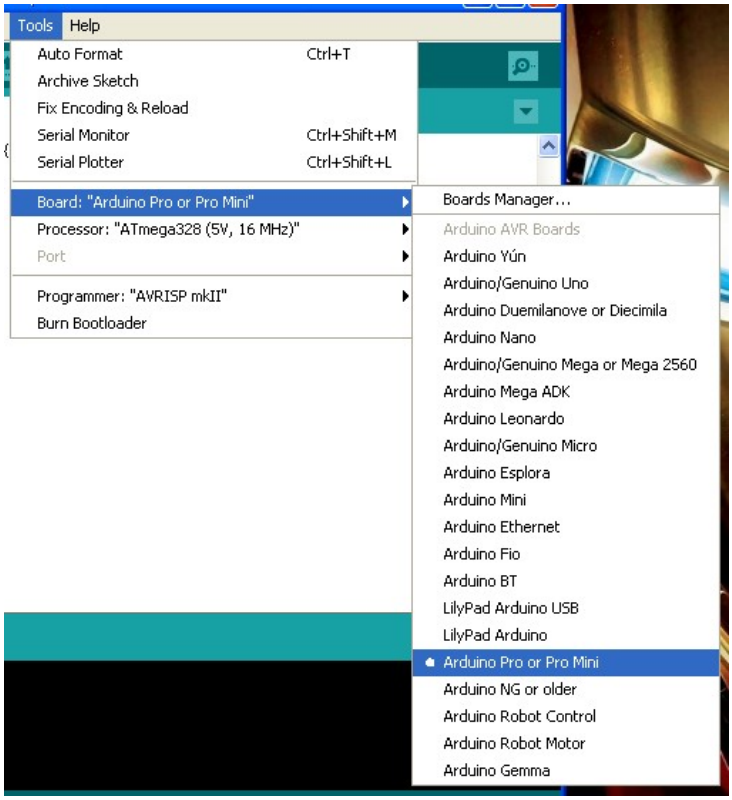
On va ensuite installer les deux bibliothèques nécessaires :

- Copier le contenu du dossier *bibliothèques* fourni avec le TP dans le dossier *libraries* qui est dans le dossier *Arduino.*
- Lancer l'Arduino IDE, regarder dans l'onglet *Fichier → Exemples,* il devrait y avoir deux nouveaux éléments: *MPU6050* et *RF24*

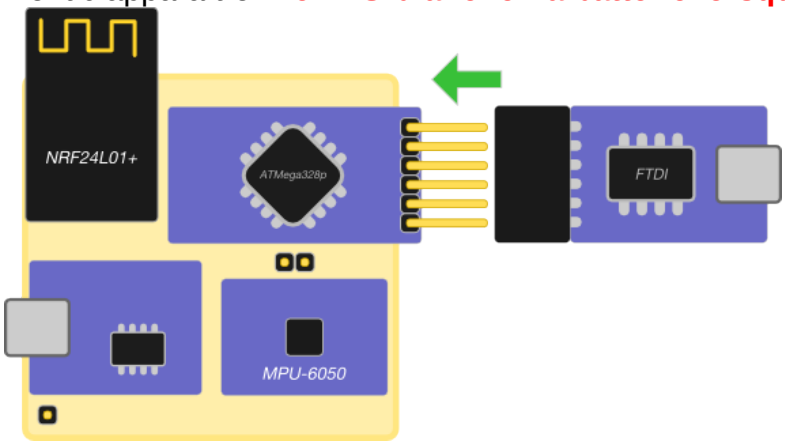


Une fois les bibliothèques installées, on programme le micro-contrôleur avec des instructions simples : allumer une LED 1 fois par seconde :

- Ouvrir *Fichier → Exemples → Basiques →Blink*
- Sélectionner la bonne configuration du micro-contrôleur en faisant *Outils → Carte → Arduino Pro ou Pro Mini*



- Noter les ports série présents sur le PC en regardant dans *Outils → Port*, puis brancher la carte IMU au PC (via la carte FTDI) et sélectionner le port série qui vient d'apparaître. **Ne PAS brancher la batterie lorsque la carte est reliée au PC**



- Flasher (ou programmer) le programme sur le micro-contrôleur en cliquant sur le bouton Téléverser (la flèche vers la droite)

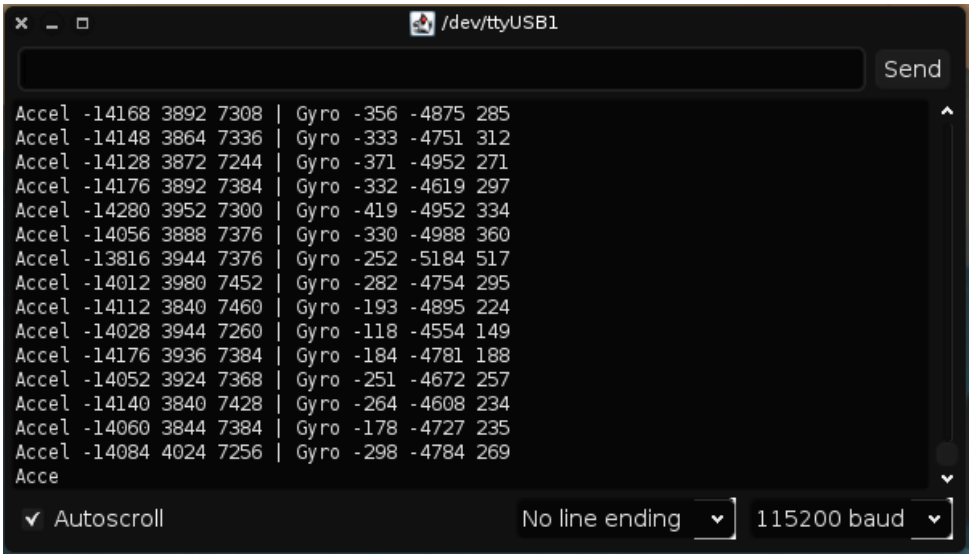
- La puce de programmation devrait clignoter un moment et l'Arduino IDE devrait afficher Téléversement *Terminé* dans la console (le carré noir en bas)
- Le micro-contrôleur devrait allumer la LED pendant 1s puis l'éteindre pendant 1s en boucle
- Fermer le logiciel

II. Adressage I2C

La puce utilisée est le *MPU-6050* de chez *Invensense*. Elle contient un accéléromètre 3 axes, un gyroscope 3 axes et une unité de calcul (DMP). Elle communique avec l'extérieur via une interface série I2C et peut avoir deux adresses au choix. Cette fonctionnalité est décrite à la page 33 de la datasheet (*MPU-6000.pdf*).

- Déterminer l'adresse en hexadécimal de la puce, sachant que la broche 9 (*AD0*) est connectée à la masse
- Copier le contenu du dossier *Code* fourni avec le TP dans le dossier *Arduino*
- Ouvrir Arduino IDE, puis ouvrir *imu_base* dans le menu *Fichier → Croquis*
- Changer l'adresse (#define…) à la valeur trouvée précédemment
- Cliquer sur le bouton *Téléverser* et attendre le message *Téléversement Terminé*
- Ouvrir la console série : (petite loupe a droite)

- Changer la vitesse a 115200 baud
- Les mesures de l'accéléromètre et du gyroscope devraient s'afficher et changer en fonction de l'orientation de la carte IMU



III. Calcul des valeurs réelles

Les valeurs affichées sont des valeurs dites "Pleine Echelle", elle n'ont pas d'unité et sont juste le résultat de la conversion de la tension de sortie des capteurs. Elles sont sur 15 bits (valeur max = 32768).

La datasheet du composant indique la valeur du bit de poids faible (*LSB*) : c'est la plus petite variation possible.

- Utiliser la datasheet (pages 12 et 13) pour compléter les fonctions *calcAngle* et *calcAccel* (attention, en anglais la virgule est juste un séparateur pour mieux lire les grands nombres, le séparateur décimal est le point)
- Une fois les fonctions complétées, activer ces fonctions en dé-commentant les lignes 30 et 31 du programme

```
// Calcul de l'acceleration lineaire et angulaire en fonction de
// calcAccel(&ax, &ay, &az, imu.getFullScaleAccelRange());
// calcAngle(&rx, &ry, &rz, imu.getFullScaleGyroRange());
```

- Les valeurs de l'accéléromètre devraient être d'environ 1g pour l'axe Z et -1g lorsqu'on met a l'envers la carte IMU
- Les valeurs du gyroscope devraient atteindre à peu près les ±250 degrés/s lorsqu'on fait tourner rapidement la carte IMU
- Si les valeurs ne semblent pas correspondre, relire les commentaires des fonctions calcAccel et calcAngle et vérifier les unités

IV. Transmission sans fil et Interface graphique

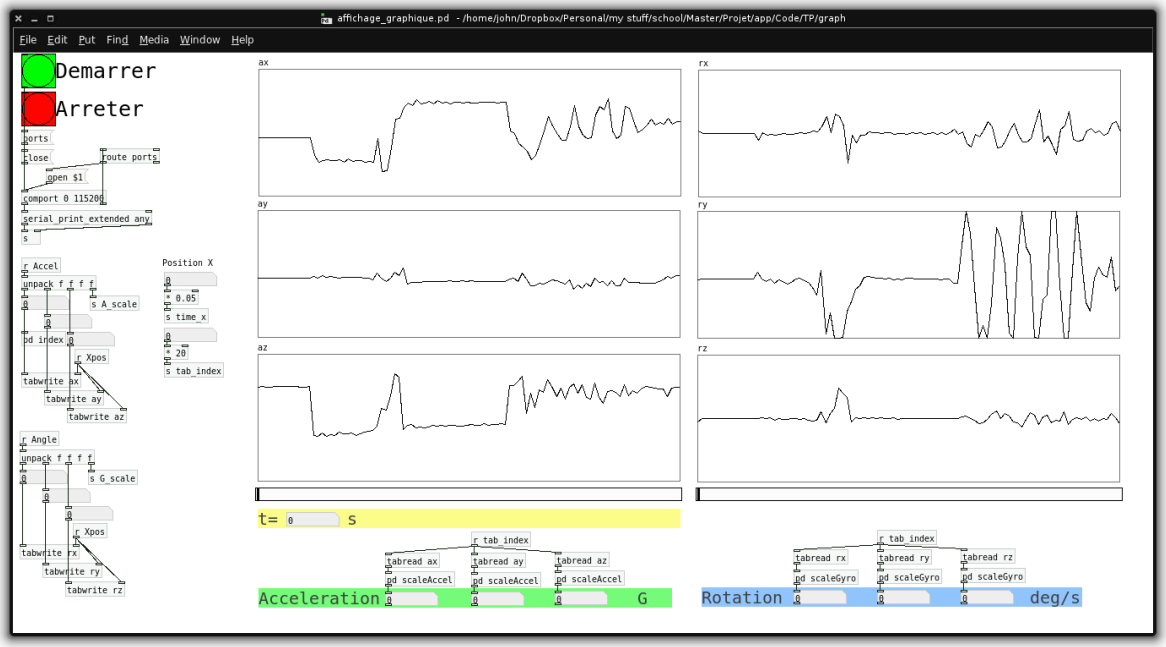
Avant d'utiliser l'affichage graphique des valeurs, on doit activer le transmetteur 2.4GHz intégré (nRF24L01+ de *Nordic*) et programmer la carte de réception :

- Ouvrir le code *imu_TX* : *Fichier → Croquis → imu_TX*
- Bien veiller a débrancher la batterie de la carte IMU – TX
- Modifier les adresses d'émission/réception (ligne 17) *1Node* et *2Node* par 5 lettres/chiffres au hasard (pour éviter d'envoyer des données chez le voisin)
- Flasher le code sur la carte IMU – TX

- Ouvrir le code *imu_RX* : *Fichier → Croquis → imu_RX*
- Débrancher la carte IMU – TX de la puce FTDI et brancher la carte IMU – RX
- Modifier les adresses (ligne 5) et mettre les mêmes que celles du code *imu_TX*
- Flasher le code sur la carte

Nous allons maintenant installer le logiciel de programmation/visualisation graphique (si ce n'est pas déjà fait) : *PureData*

- Télécharger le logiciel : http://sourceforge.net/projects/pure-data/files/pd-extended/0.43.4/Pd-0.43.4-extended-windowsxp-i386.exe/download
- Installer le logiciel, confirmer toutes les boites de dialogue
- Ouvrir le fichier *affichage_graphique.pd* dans le dossier *graph*



L'interface/code est séparée en 3 parties :

- A gauche : réception et le décodage des trames série envoyées par la carte RX
- Au milieu l'affichage graphique : à gauche l'accélération X, Y et Z et à droite la vitesse de rotation
- En bas des graphes : les curseurs, le temps *t* et les valeurs précises au temps *t*

Utilisation :

- Brancher la carte IMU – RX
- Brancher la batterie de la carte IMU – TX
- Cliquer sur *Démarrer* (Bouton vert)
- Les graphes devraient commencer a enregistrer les mouvements de la carte TX
- Faire tourner rapidement la carte TX sur l'axe X (mouvement du poignet analogue à utiliser un tournevis) : les axes sont indiqués sur la carte *MPU-6050*
- Cliquer sur *Arrêter*
- Utiliser les curseurs pour vérifier que la vitesse de rotation max. enregistrée sur l'axe X est bien de 250°/s et sature facilement

Réduction de la sensibilité du gyroscope :

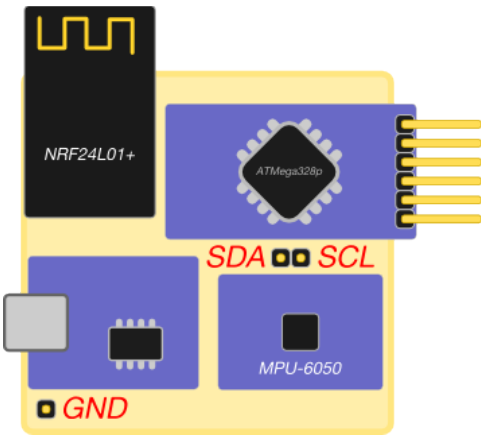
- Débrancher la carte RX de la puce FTDI
- Modifier la ligne 39 du programme *imu_TX* et mettre un 3 à la place du 0
- Débrancher la batterie de la carte TX
- Brancher la carte TX à la puce FTDI et flasher le code modifié
- Débrancher la carte TX et rebrancher la carte RX
- Brancher la batterie de la carte TX
- Cliquer sur *Démarrer* sur l'interface graphique
- Faire le même mouvement de poignet que précédemment avec la même force
- Cliquer sur *Arreter*
- Mesurer la vitesse de rotation max. , consulter les pages 12/13 de la datasheet et conclure sur l'opération effectuée

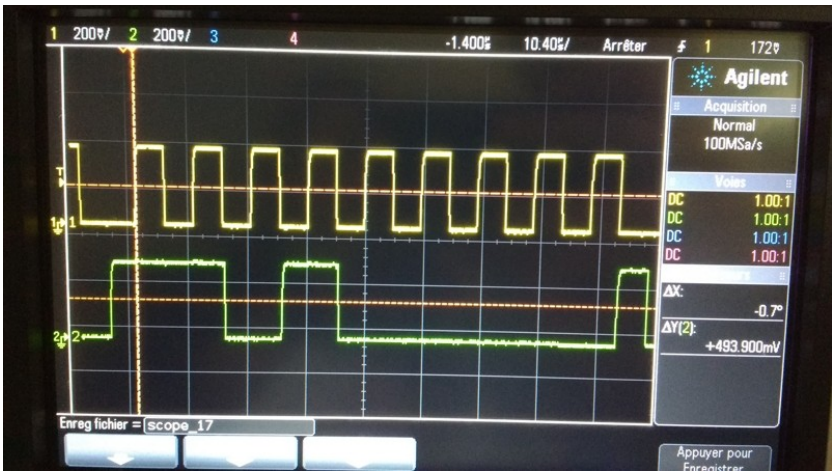
V. Protocole de communication (I2C)

Le gyroscope/accéléromètre communique via un protocole série appelé I2C Bus : Inter-Integrated Circuit Bus. Comme son nom l'indique, il est principalement utilisé pour communiquer entre différents circuits intégrés sur une même carte.

Nous allons d'abord visualiser et décoder un début de trame I2C à l'oscilloscope, puis analyser un relevé fait à l'analyseur logique (décodage plus facile) :

- Brancher les 2 voies de l'oscilloscope aux signaux *SDA* (Données) et *SCL* (Horloge), puis brancher la masse au pin *GND* sur la carte IMU – TX
- Allumer l'oscilloscope, appuyer sur le bouton *Auto* ou *Autoscale* pour réinitialiser les paramètres
- Modifier les paramètres (Temps et Tension), puis utiliser le bouton *Single* pour effectuer des acquisitions individuelles jusqu'à avoir une trame qui ressemble à peu près à la photo





Recopier (ou imprimer) la trame sur feuille et identifier les différents éléments : voir les pages 33 à 35 de la datasheet *MPU-6000.pdf*

- Conditions *START* et *STOP*
- Réponse de l'esclave (MPU-6050) au maître (ATMega328) : *ACK*
- 7 bits d'adresse
- Bit *Read/Write*

- Décoder les 7 bits d'adresse, les passer en hexadécimal et comparer avec l'adresse calculée précédemment

Décodage des mesures inertielles :

Un analyseur logique est un appareil digital, il ne mesure pas la tension mais seulement l'état logique des lignes à mesurer. Il est doté d'une grande mémoire et est utilisé pour debugger les systèmes numériques.
Il est aussi fourni la plupart du temps avec des analyseurs de protocoles (I2C, SPI, CAN etc..) qui décodent automatiquement les trames enregistrées et fournissent un fichier texte/csv contenant les données décodées en hexadécimal.
Ici, nous avons placé la carte IMU – TX dans une certaine position et effectué un relevé à l'analyseur logique, il va falloir retrouver cette position en décodant/convertissant les données relevées :

- Ouvrir le fichier *TX_i2c_decode.ods* dans le dossier *capture* fourni avec le TP
- Recopier sur feuille la première trame (entre le premier *Read* et le 2^{e} *Write*)
- A l'aide des pages 29 à 31 de la datasheet *MPU-6000-Register-Map.pdf*, identifier les données appartenant à chaque capteur et axe : accélération (X, Y, Z),

température, et rotation (X, Y, Z), sachant que les adresses des registres s'incrémentent automatiquement
- Assembler les MSB et LSB de chaque axe de l'accéléromètre, et sachant que la sensibilité est de 2g, calculer la valeur réelle en g
- Déduire la position de la carte dans l'espace au moment de la mesure

Bonus :
- En utilisant la formule page 30, calculer la température au moment de l'acquisition
- Attention : la valeur binaire est en complément à 2

Pour ceux qui veulent voir à quoi ressemble une interface d'analyseur logique, télécharger et installer *Logic* de *Saleae* ici : https://www.saleae.com/downloads
Il suffit de déplacer le fichier *TX_i2c.logicdata* dans la fenêtre du logiciel pour ouvrir les trames enregistrées. (zoom avec la molette et déplacement avec clic gauche)

Deuxième Partie : Étude de l'accélération mesurée

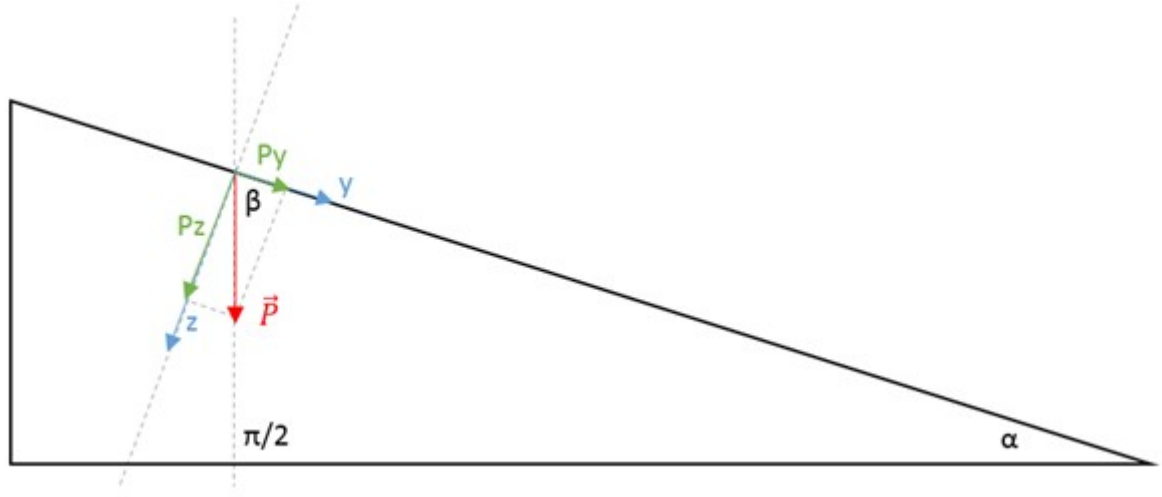
I. Étude Statique

- Brancher la batterie de la carte TX et la carte RX à l'ordinateur.
- Ouvrir le fichier *affichage_graphique.pd* dans le dossier *graph.*
- Visualiser les valeurs obtenues sur les 3 axes en posant la carte TX sur une surface plane et statique.
- Repérer dans quelle direction pointent les 3 axes, le repère est-il orthonormé ?
- Qu'obtient-on comme valeurs ? Expliquer vos résultats.

II. Étude Dynamique sur Tourne-disque

- Placer la carte TX sur le tourne-disque.
- Mettre en marche le tourne-disque.
- Visualiser les valeurs obtenues pour chacune des vitesses.
- Qu'obtient-on comme valeurs ? Expliquer vos résultats.
- Comparer les mesures à des distances différentes du centre du tourne-disque.
- A l'aide de vos mesures, le tourne-disque vous semble-t-il bien réglé ? Pourquoi ?

III. Étude Dynamique sur Rampe



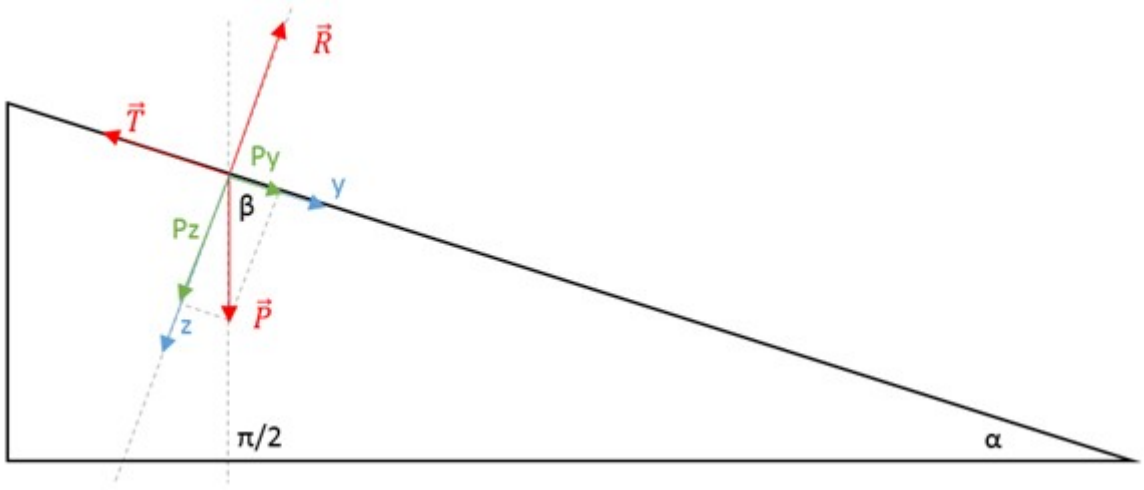
On va maintenant utiliser la carte TX afin de mesurer le coefficient de frottement de plusieurs surfaces. On peut ainsi modéliser le système comme vu sur le schéma ci-dessus.

- A l'aide du principe fondamental de la statique retrouver l'angle d'inclinaison de la rampe.

Données :

$$\vec{P} = Px.\vec{x} + Py.\vec{y} + Pz.\vec{z}$$
$$\cos\left(\frac{\pi}{2} - a\right) = \sin(a)$$
$$\sin\left(\frac{\pi}{2} - a\right) = \cos(a)$$

- Placer la carte TX sur la rampe.
- Relever la courbe qui se dessine lors de la descente de la carte TX. Repérer les différentes étapes d'accélération et expliquer.
- Visualiser les valeurs obtenues pour chacun des contacts des surfaces proposés (Aluminium/Aluminium, Aluminium/Bois, Aluminium/Téflon).
- Qu'obtient-on comme valeurs lors de la descente de la carte TX ?
- Bonus : Connaissant l'angle ainsi que les mesures d'accélération, quelles sont les coefficients de frottement respectifs des contacts Aluminium/Aluminium, Aluminium/Bois, Aluminium/Téflon ?



Données :

P le poids (m*g)
R la réaction du support (R)
T la force de frottement (T)

$fd = \frac{T}{R}$ Le coefficient de frottement

M = 149.3g

I. Étude statique

On obtient 1g selon l’axe perpendiculaire au plan ce qui est correct car l’on mesure les 9.81m/s de l’attraction terrestre.
Non le repère est inversé.

II. Étude Dynamique sur Tourne-disque

On obtient respectivement pour les vitesses 16, 33, 45, et 78 les valeurs 960, 1980, 2700, et 4680 en degré par seconde ce qui converties en tour par minute correspond bien aux vitesses du tourne-disque.

On observe aucun changement des mesures selon la distance de la carte TX par rapport au centre du tourne-disque ce qui parait logique car ce sont les forces de Coriolis que mesure le gyroscope.

On remarque que le tourne-disque va légèrement plus vite que la vitesse des 33T/min cela est dut principalement au fait que la vitesse réelle des tourne-disques est $33_{1/3}$ T/min mais possiblement dut quand on regarde les autres vitesses à la résistance du diamant lorsqu’il est en contact avec le vinyle.
Conclusion : Oui, le tourne-disque semble avoir été réglé volontairement sur une vitesse plus rapide ce qui correspond à la vitesse idéale lorsque le diamant est en contact.

III. Étude Dynamique sur Rampe

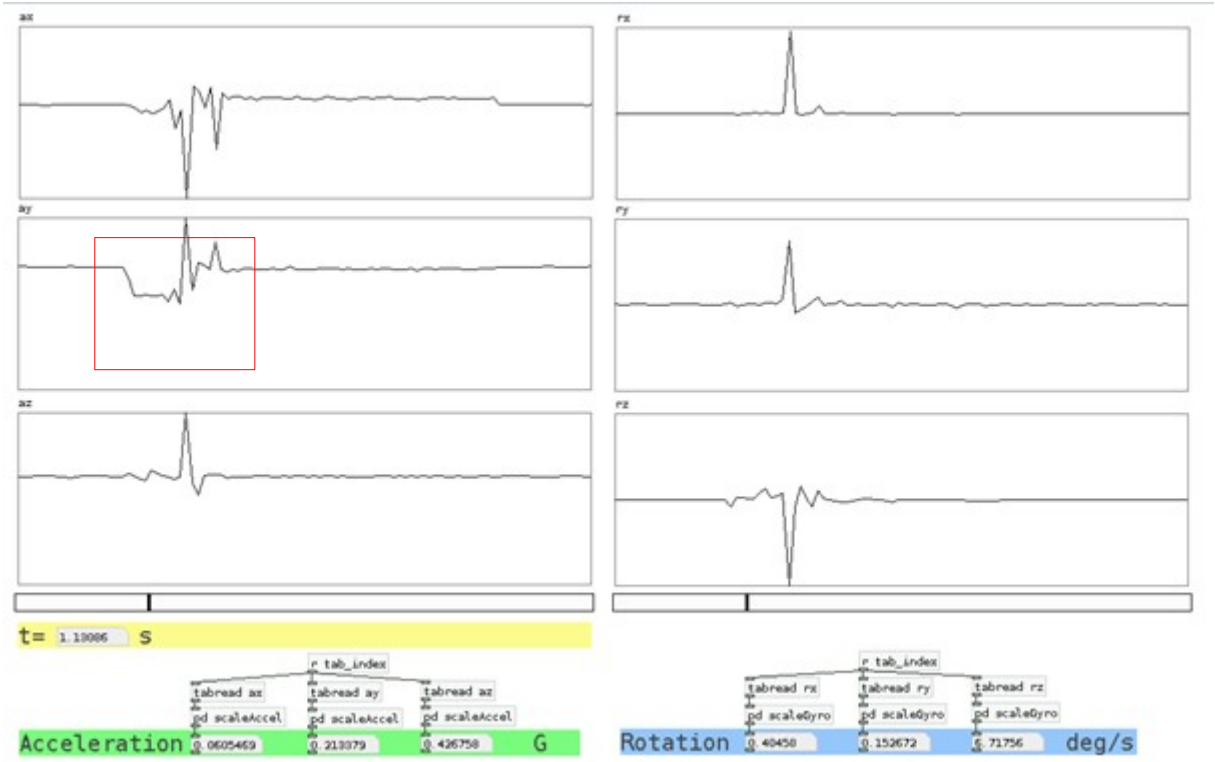
En projetant P sur les axes on obtient :

$Px.x = 0$
$Py.y = g.\cos(B) = g.\cos(Pi/2 – A) = g.\sin(A)$
$Pz.z = g.\sin(B) = g.\sin(Pi/2 – A) = g.\cos(A)$

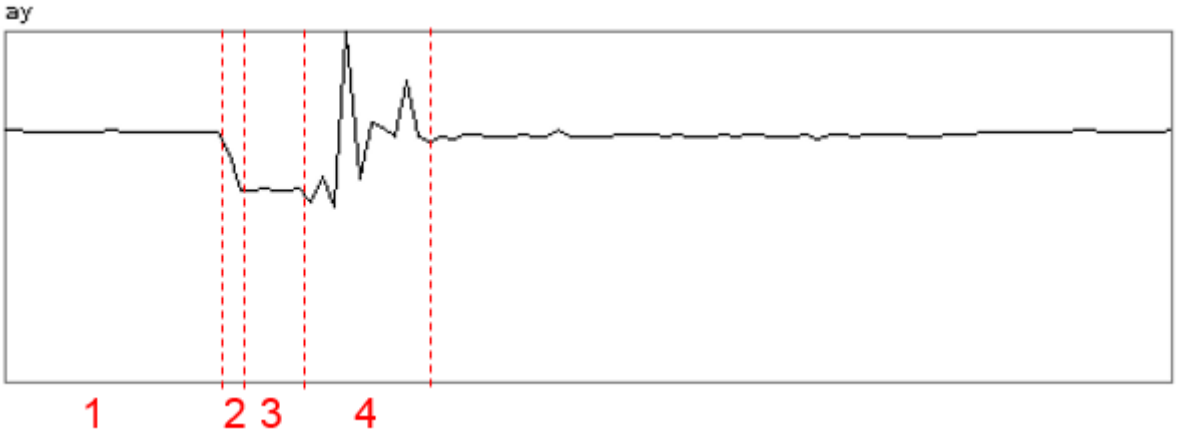
tan(B) = Pz/Py

B = arctan(Pz/Py) = 90 - A

A = 90 – arctan(Pz/Py)



Aluminium sur aluminium

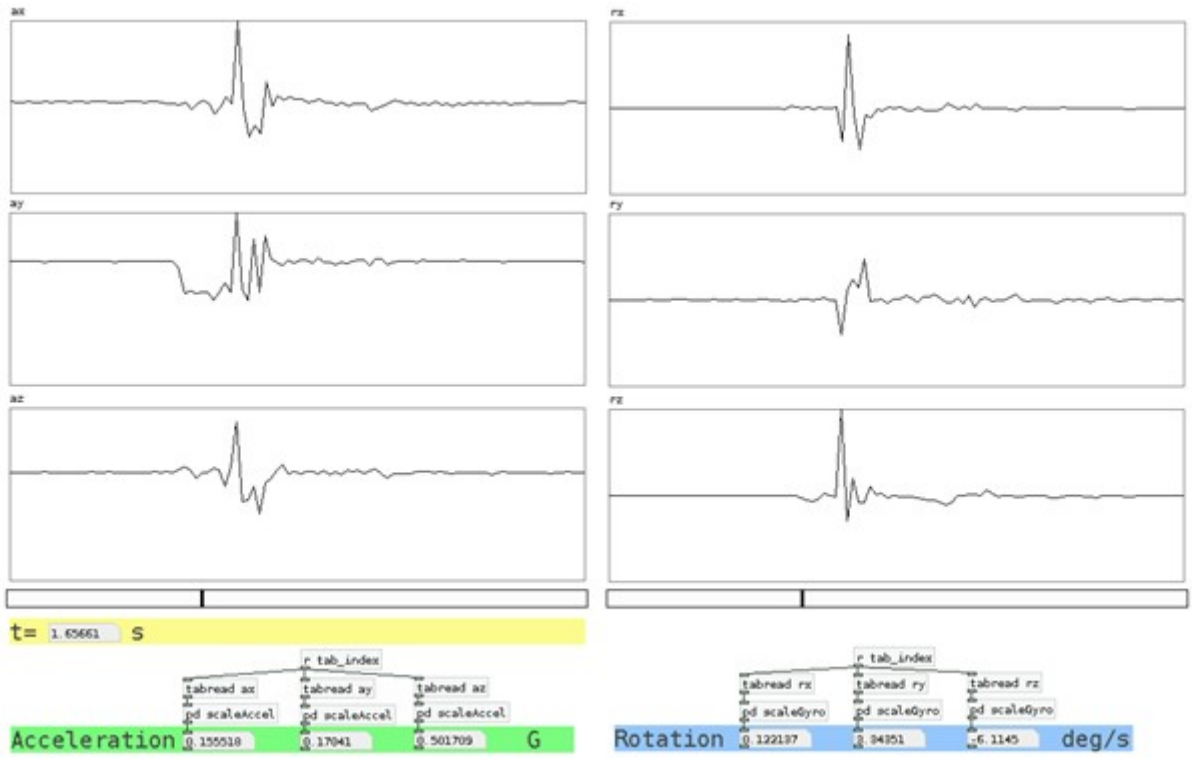


Zone 1 : Accélération constante >> La carte TX est statique.

Zone 2 : Accélération variante >> La carte TX commence à être lâchée.

Zone 3 : Accélération constante >> La carte TX glisse.

Zone 4 : Accélération variante >> La carte TX capte le choc brutal lorsque la carte s’arrête.



Téflon sur aluminium

En incluant cette fois la réaction du support et la force de frottement on obtient :
Sur x >> 0
Sur y >> (Py - T).y
Sur z >> (Pz – R).z
On veut la force de frottement lors du déplacement on va donc regarder l’accélération sur l’axe y.

m.ay = m.g.sin(A) – T.g
T = m.g.(sin(A) – ay)

Maintenant sachant que le coefficient de frottement est le quotient de la force de frottement sur la réaction du support on observe sur l’axe y :
m.az = m.g.cos(A) – R.g
R = m.g.(cos(A) - az)
Ce qui nous donne fd = T/R