2018

PROJET UE 1
HeXart Care

PROJET 1 : Fondamentaux Scientifiques

Tuteur : KISTER Mathieu

Membres du groupe :

- POMMERY Tristan
- REGALL Nicolas
- SCHUBNEL Thomas
- CARRE Etienne

PROJET GROUPE 6

CESI.eXia - Lingolsheim

16/11/2018

SOMMAIRE

I. PARTIE 1 : RECHERCHE

Pour commencer cette fiche technique de projet, on va partir sur une analyse de type prosit avec cependant quelques étapes en moins.

Contexte

On souhaite recréer un cardio-fréquence mètre a partir des plans restants

Contraintes

Les contraintes de ce projet sont d'utiliser Arduino, Processing et de faire un programme en C pour traiter les données. Il faut aussi faire un montage électronique avec les composants requis.

Livrables

Un cardio-fréquence mètre fonctionnel en Arduino, ainsi qu'un exécutable processing transformant les données obtenues par l'Arduino en un fichier .csv, et un programme en C capable de traiter les données.

Avancement

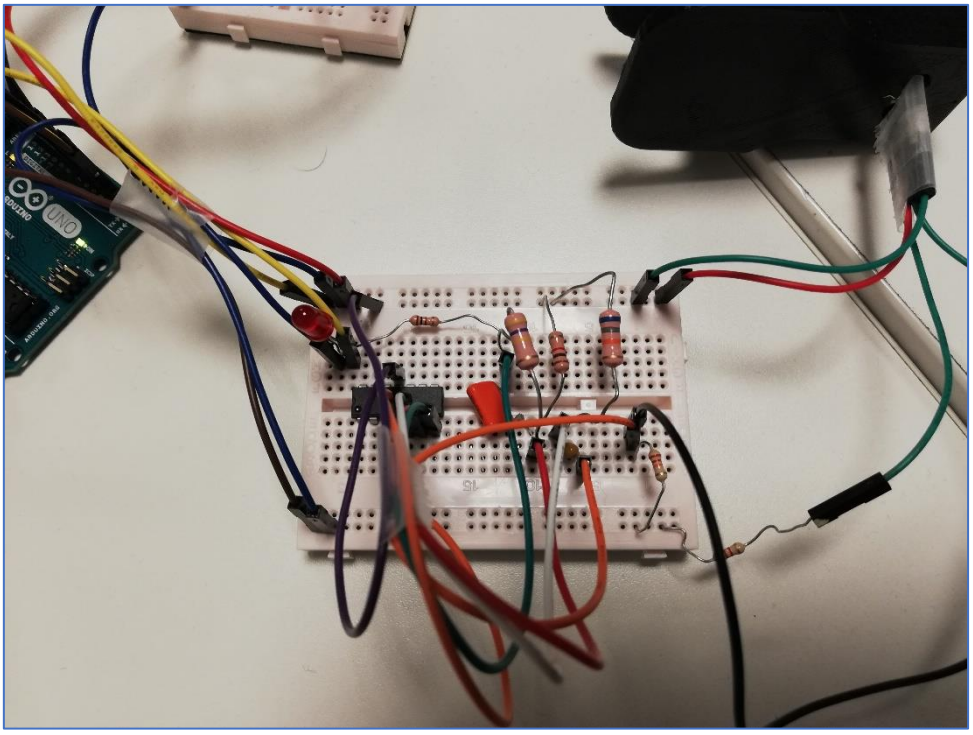
Pour voir la feuille d'avancement du projet, il faut aller la chercher sur GitHub dans la branche « master » et ouvrir le fichier PDF pour bénéficier de toutes les informations.

II. PARTIE 2 : EXECUTION DES TACHES

On va répartir la partie de réalisation du projet en différentes étapes distinctes, pour qu'on puisse distinguer aisément les différentes choses effectuées durant tout ce projet.

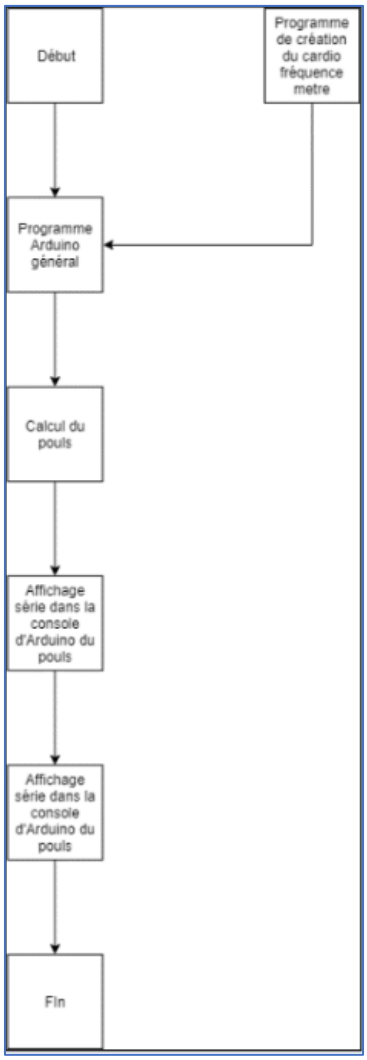
Module 3.1

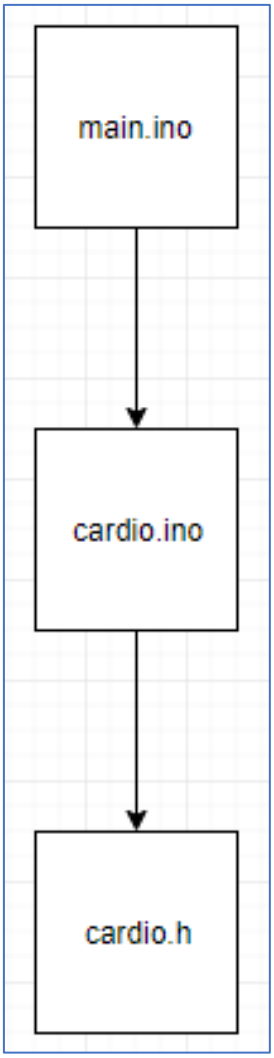
Le module 3.1 était le module cardio – fréquence mètre. Ce module a été découpé en plusieurs parties, avec tout d'abord la réalisation du montage fourni dans le guide. On a donc disposé les composants électroniques de manière équivalente pour être sur du bon fonctionnement, du circuit.



Pour plus d'explications sur ce schéma, il faut voir le dossier d'explications du schéma électronique du module 3.1 disponible dans le dépôt Git.

Ci-dessous l'algorigramme du module 3.1 et à coté la hiérarchie des dossiers / bibliothèques de celui – ci.





On peut donc voir que la bibliothèque cardio.h est appelée par le fichier cardio.ino qui déclare la fonction calculPouls(), car le fichier .h est un fichier d'en-tête, c'est-à-dire un fichier ou on définit les procédures de notre programme.

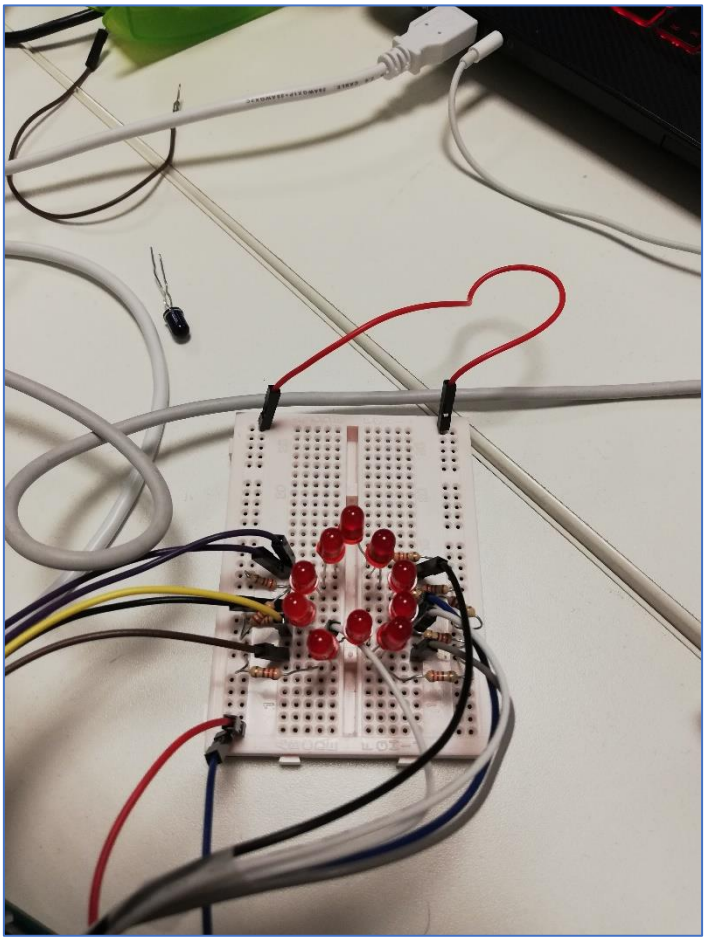
On a donc bien utilisé les void setup() et void loop() dans le main.ino et on a déclaré une seule fonction, la calculPouls () au lieu de 2 fonctions comme énoncé dans l'avancement. On a donc dans le cardio.ino tous les calculs nécessaires pour le calcul du pouls dans la fonction calculPouls().

Et enfin on a juste a appeler la fonction calculPouls() dans le void loop() pour qu'elle soit bien prise en compte lors de l'exécution du main.ino.

On obtiendra donc comme voulu le pouls ainsi que le temps depuis le début du programme.

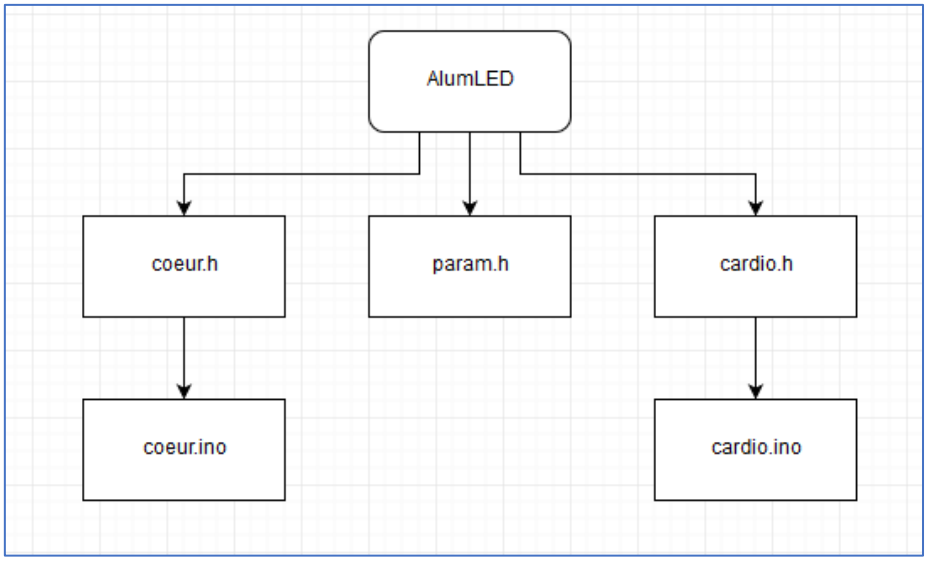
Module 3.2

Le module 3.2 était le module cœur. Ce module a été découpé en plusieurs parties, avec tout d'abord la réalisation du montage fourni dans le guide. On a donc disposé les composants électroniques de manière équivalente pour être sur du bon fonctionnement, du circuit. On s'est basé sur les schémas de fritzing (voir document des schémas électriques) et on a obtenu ceci.



Les explications sur le schéma sont disponibles dans le fichier de schémas électroniques du module 3.2 sur gitHub.

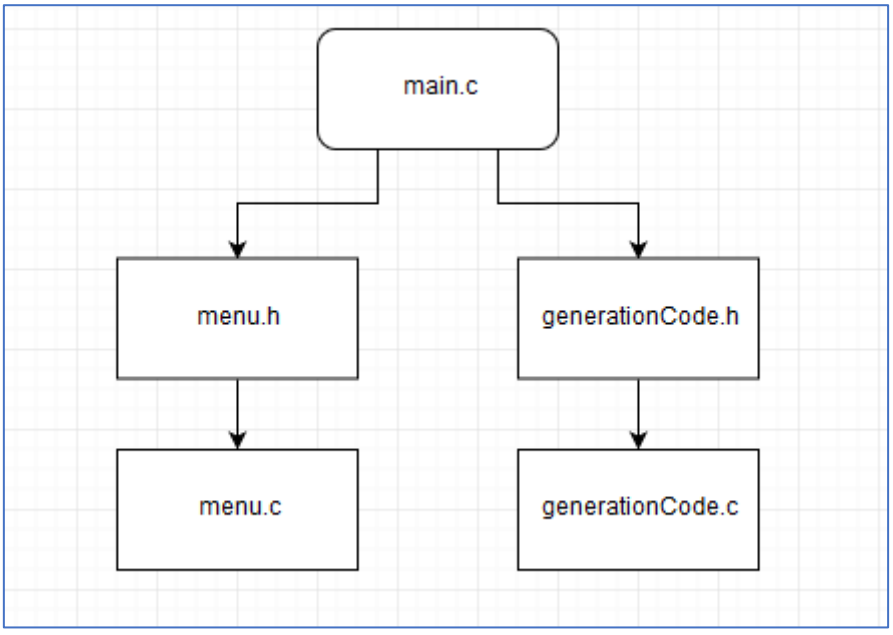
Le programme du cœur a été découpé en plusieurs parties avec comme pour le module 3.1, le main.ino en exécutable principal.



On a donc les différents fichiers nécessaires au bon fonctionnement du programme Arduino du cœur avec la bibliothèque cœur.h qui permet de définir les procédures nécessaires et ensuite la définition de cette procédure dans le cœur.ino. On retrouve également la partie cardio qui, couplée avec la partie cœur permet de faire clignoter les LED du cœur en fonction du pouls mesuré.

Il y a un nouveau fichier en tête qui est pris en compte, c'est le fichier param.h .

Le but de ce fichier est de définir quel type d'allumage des LED on veut obtenir. Cependant, ce fichier n'est pas modifié manuellement, il faut qu'il soit modifié par un autre programme mais celui-ci écrit en C. On peut voir ci-dessous le schéma montrant la hiérarchie des fonctions du programme en C permettant de générer le fichier param.h

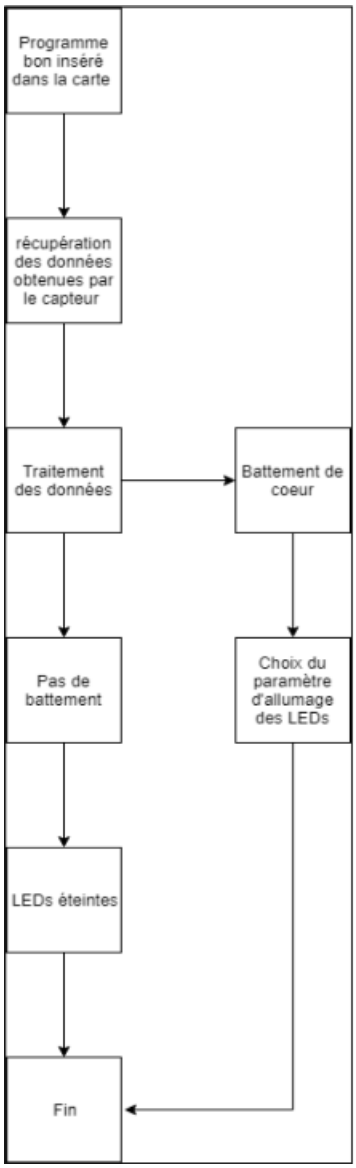


On a donc le main.c qui va appeler les différentes fonctions que l'on définit en dessous, c'est-à-dire celles du menu.c et du generationCode.c

Comme énoncé précédemment, les fichiers .h servent a déclarer les procédures de fonctions et on a dans les .c les definitions de nos procédures. C'est-à-dire dans le menu.c on aura les choix des menus d'allumage des LED. Et dans le générationCode on aura le programme pour générer le fichier param.h en fonction du menu choisi.

Pour plus d'informations relatives au programme, il est possible de voir les commentaires intégrés dans les différents codes.

Pour permettre de comprendre facilement le déroulement du programme, on peut se référer a l'arbre ci-dessous.

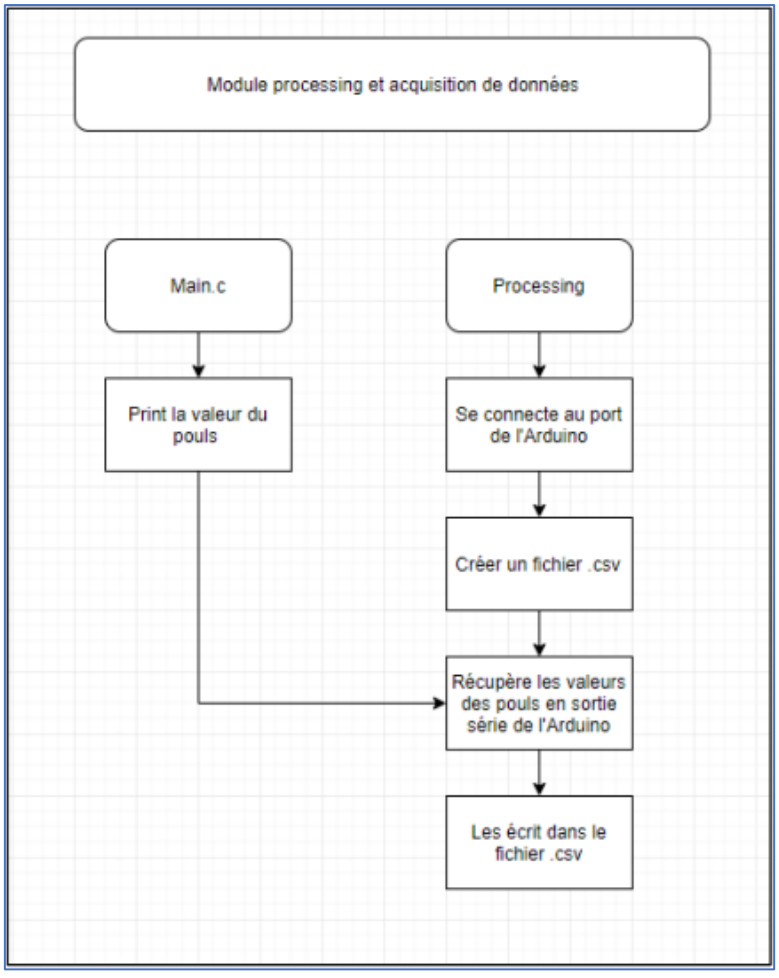


Cette partie a donc été ajoutée à la première et on obtient bien notre battement du cœur de LED en fonction du pouls mesuré.

Module 3.3

Le module 3 était le module processing. C'est-à-dire celui ou le programme processing récupère les informations envoyées sur le port série par l'Arduino et a la fermeture du fichier processing, le programme génère un fichier .csv (sous format excel) avec une colonne de battement et une colonne de temps. Et ce fichier pourra ensuite être récupéré dans le programme en C et les informations pourront être traitées en fonction de l'action choisie.

Ci-dessous le schéma montrant le déroulement du programme processing.



Module 3.4

Le module 3.4 était le module de traitement de données à proprement parler, c'est-à-dire que c'est le programme de ce module qui va récupérer le fichier .csv généré et qui va, selon le choix de l'utilisateur, trier les données, trouver le pouls maximum etc.

Pour cette partie, il a fallu utiliser des structures de données et utiliser des bibliothèques différentes des autres programmes.

On peut voir ci-dessous les différents schémas pour les structures utilisées, la repartition des bibliothèques et la modularisation des fichiers.

STRUCTURES UTILISEES

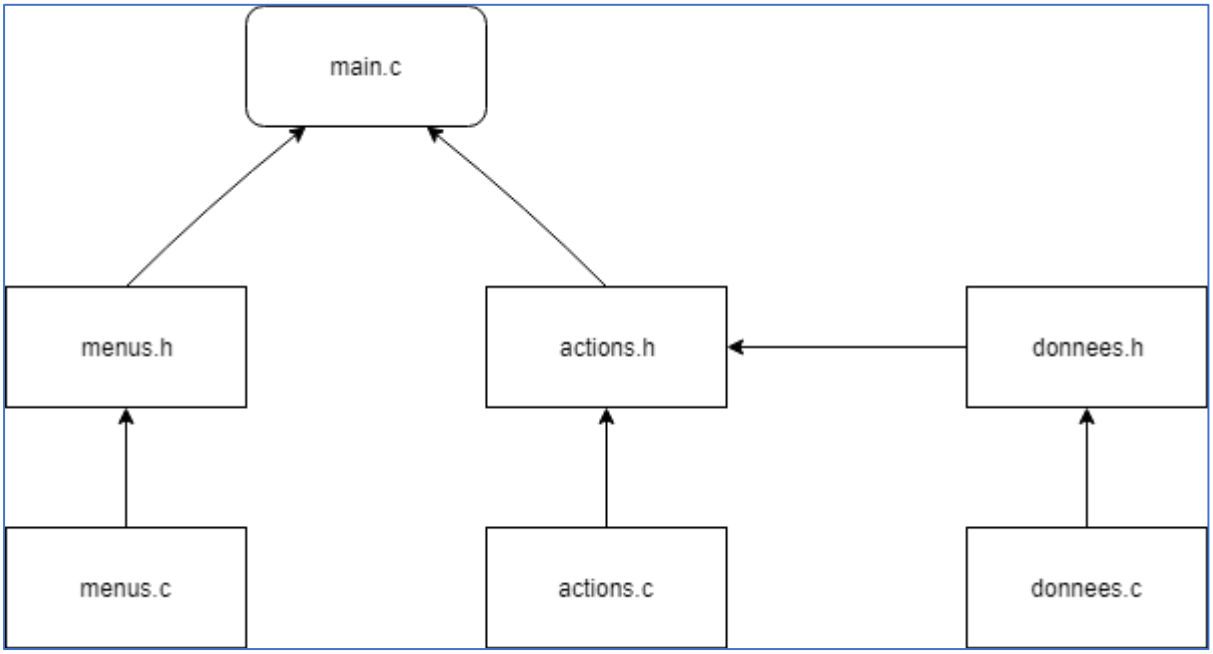
Ligne	**Element**	**File**
int frequence	Ligne donnee	Element *premier
int temps	Element *suivant	

REPARTITION DES BIBLIOTHEQUES

actions.h	**donnees.h**
void afficher_file(File *file)	typedef struct ligne Ligne
void rech_temps(File *file)	typedef struct Element Element
void rech_pouls_max(File *file)	typedef struct file File
void rech_pouls_min(File *file)	File *initialiser_file()
void moy_pouls(File *file)	void file_ajout(File *file, Ligne nvdonnee)
void tri_insertion(Ligne tab[], int taille)	Ligne file_retrait(File *file)
void sort(File *file, int taille, int choixMenu21)	void file_clear(File *file)
void renverse(File *file, int taille)	int lecture(File *file, int taille)

menus.h
int afficherMenu(int choixMenu)
int afficherMenu2(int choixMenu2)
int afficherMenu6(int choixMenu6)
int afficherMenu21(int choixMenu21)
int afficherMenu22(int choixMenu22)

MODULARISATION DES FICHIERS



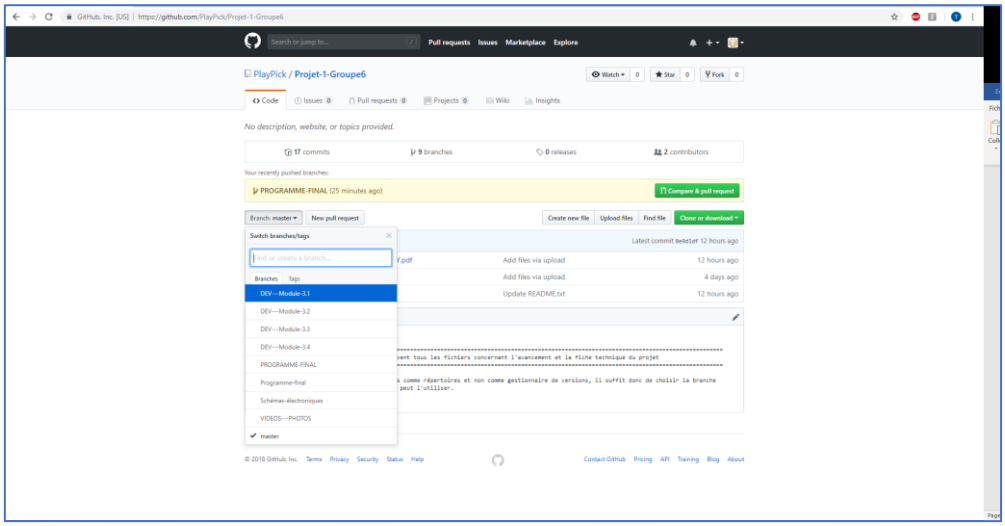
On a donc dans ce programme le programme du menu, qui va nous générer une liste de choix d'actions à effectuer, et en fonction du choix on exécute l'action. On a le fichier actions.c qui contient tous les types d'actions possibles, comme par exemple le tri, la recherche du maximum etc. Et dans le fichier donnees.c on a toutes les définitions de notre structure de données pour le stockage des informations, et donc pour pouvoir les réutiliser ensuite.

Pour plus d'informations sur le programme, il est possible de voir les commentaires de celui-ci dans la branche module 3.4 sur le dépôt Git.

Création du dépôt Git

Pour le partage de nos fichiers relatifs au projet, nous avons choisi d'utiliser GitHub. Mais qu'est-ce que GitHub ? GitHub est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels. Livrée en tant que logiciel à la demande (SaaS, Software as a Service), la solution GitHub a été lancée en 2008. Elle repose sur Git, un système de gestion de code open source créé par Linus Torvalds dans le but d'accélérer le développement logiciel.

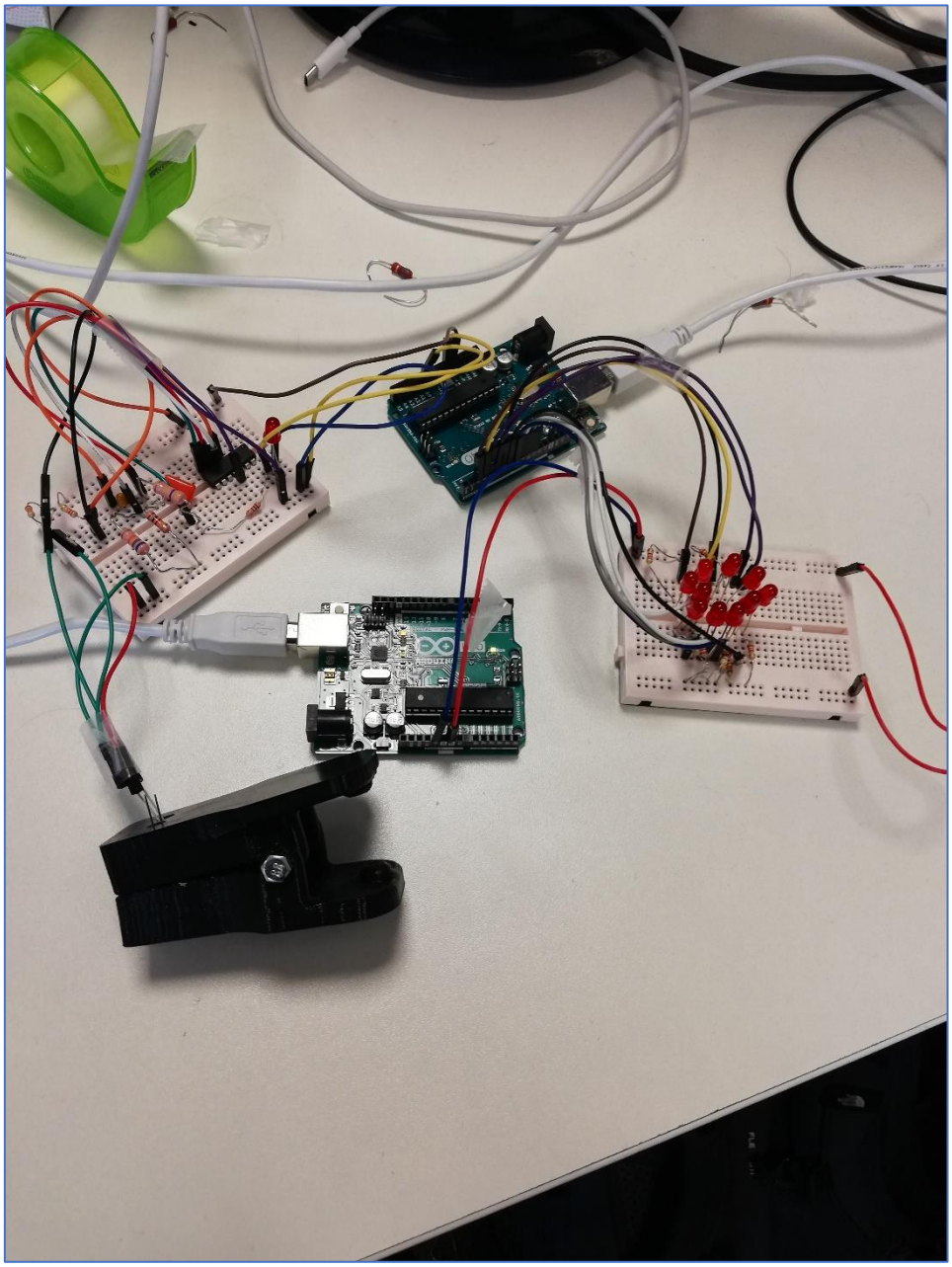
Le site est représenté comme ceci lorsqu'on crée un projet.



Nous n'avons pas utilisé git comme gestionnaire de versions mais comme un répertoire pour nos dossiers. Nous avons donc créé une branche relative a la branche principale « master » pour chaque dossier que nous voulions avoir. Par exemple le dossier DEV---Module 3.1 qui comprend les programmes pour le module 3.1 par exemple.

Création du montage

Ci-dessous il est possible de voir le montage final du projet. On a utilisé un second Arduino pour avoir moins d'éléments sur le ground (la terre). Car avoir trop d'éléments nous générait beaucoup d'interférences et ne nous permettait pas d'avoir des données plausibles. On a donc réparti équitablement les composants sur les pins ground et le problème fut réglé.

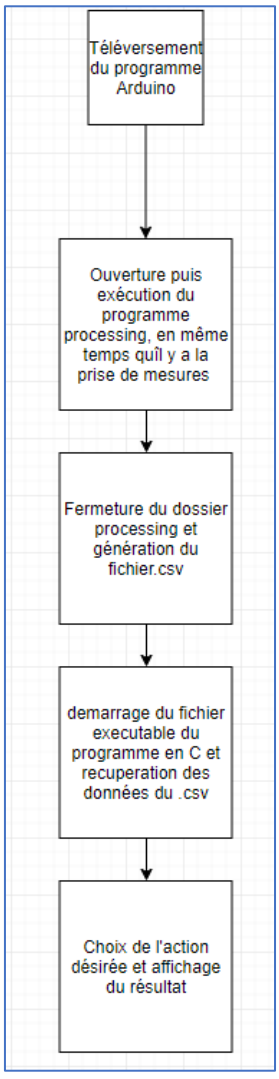


Mise en commun des modules

Après avoir créé les programmes pour les différents modules, il a fallu les combiner pour obtenir le programme final. Cette partie était simple car il n'a fallu combiner que les programmes Arduino. On a donc mis ensembles les deux main.ino et on a ajouté l'appel des fonctions dans le fichier Arduino final. Tout en prenant soin de modifier le programme pour qu'il fonctionne parfaitement.

Exécution du programme

Ci-dessous le diagramme expliquant le déroulement du programme de manière simple



Récapitulatif de l'avancement

Tout ce qui a été énoncé dans l'avancement a bien été exécuté sauf le respect de l'emploi du temps. En effet, de l'avance a directement été prise ce qui fait que l'on a pu plus facilement passer sur d'autres

tâches et donc les faire avancer rapidement également. Cela nous a permis d'avoir plus de temps pour analyser nos programmes et corriger les problèmes éventuels.

III. PARTIE 3 : BILAN ET CONCLUSIONS

Conclusion et retours sur les objectifs

Les objectifs du projet ont bien été atteints et ont été réalisé dans les temps donnés. Donc une conclusion de projet positive pour l'ensemble des membres du groupe.

Bilan critique du travail effectué

Le travail a bien été effectué, la prise d'avance nous a été d'une très grande aide car cela nous a permis d'assimiler les notions des autres modules, comme nous nous étions associés chacun à un module.

Synthèse du travail effectué et des résultats obtenus

Le travail effectué est celui demandé et les résultats obtenus sont très concluants.

Référence des méthodes et outils utilisés

On a pu utiliser des logiciels comme fritzing, tinkerCAD pour toute la modélisation électronique. L'IDE Arduino pour toute la programmation Arduino. GitHub pour le dépôt de fichiers, et enfin VSCODE, CodeBlocks, Clion pour toute la partie programmation en C (chacun avait son propre choix d'IDE).

Références bibliographiques et sitographies utilisées pour le projet

- Le guide de projet
- www.Arduino.cc
- www.openclassrooms.com

Et tous les éventuels sites visités pour les recherches sur des notions un peu floues