**Projet fondamentaux scientifiques**



**« Le cardiofréquencemètre »**

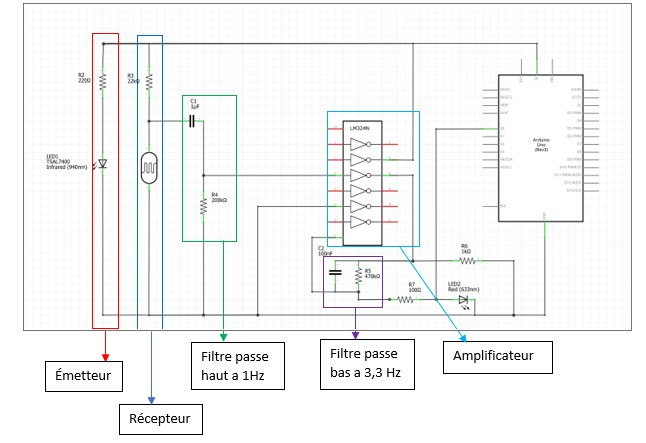
3.1 Module cardio :

3.1.1) Définition des attentes

L’objectif de cette partie est de réaliser un montage électronique couplé à un programme permettant de détecter les battements du cœur et de les envoyer, couplés à la durée de mesure, sur le moniteur série de l’Arduino.

Cette partie sera considérée comme réussie si nous obtenons des valeurs cohérentes lors de la mesure du pouls et si elles sont bien envoyées sur le moniteur. De plus, le programme devra être modulaire.

3.1.2) Application

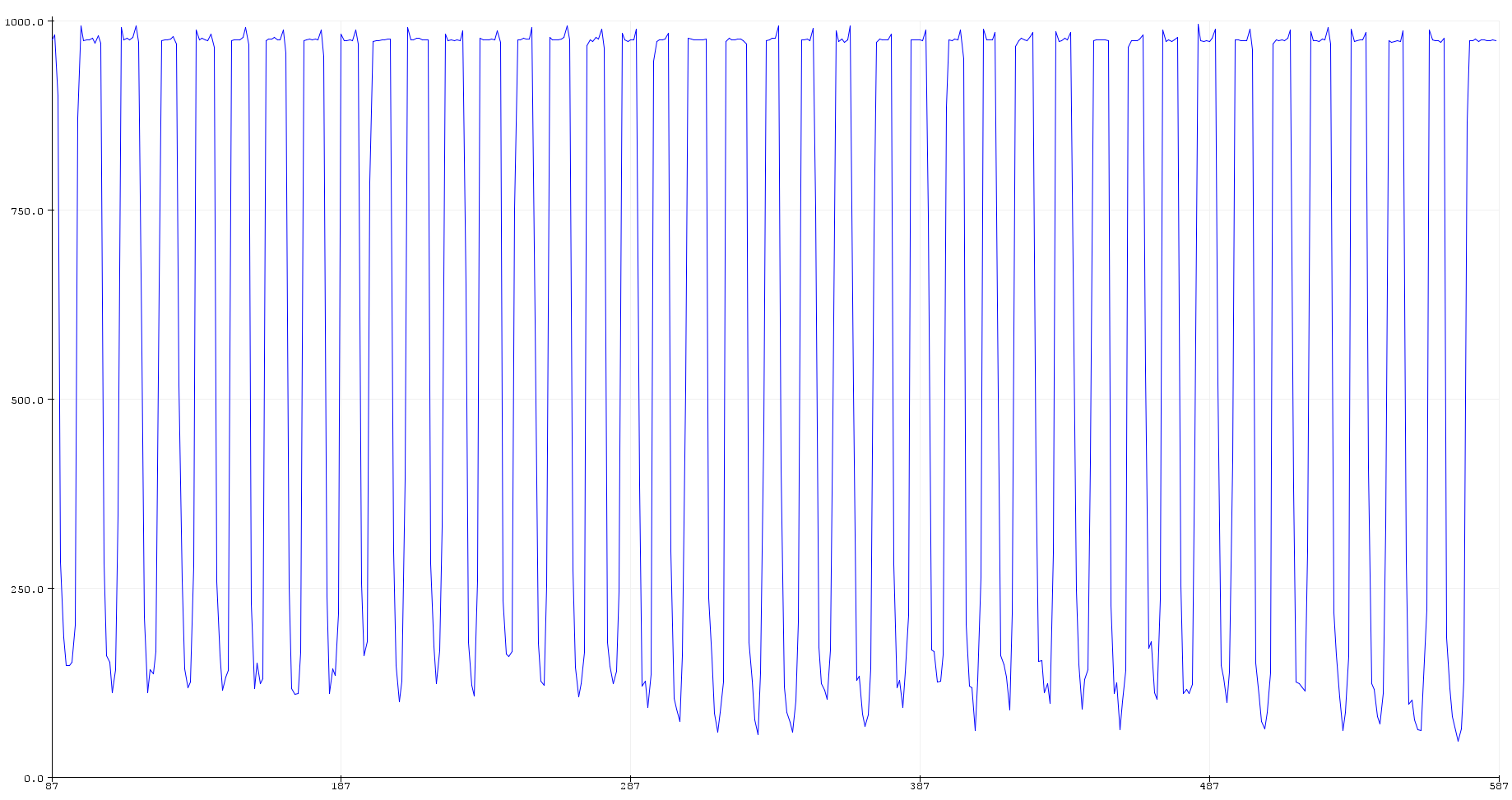


Le schéma ci-dessus représente un capteur cardiofréquencemètre :

- Il est composé de deux filtres, passe-haut et passe-bas qui permettent de garder les fréquences variant entre 1Hz et 3.3Hz pour ne garder que les fréquences utiles et pour supprimer les fréquences qui pourraient perturber le signal de sortie.

Le composant LM324N est un amplificateur : il permet d’améliorer le signal c'est-à-dire qu’il permet de mieux visualiser une pulsation cardiaque.

Voici 6 pulsations du cœur



Ce schéma est une simulation de pulsation cardiaque effectuée avec un potentiomètre.

Voici le programme modulaire mis en œuvre dans ma partie :

[Code Arduino 1\_2](https://github.com/LeBescherelle/Project-1/tree/master/Code_Arduino_module_1__2)

3.1.3) Interprétation des résultats

|  |  |  |
| --- | --- | --- |
|  | Réaliser un programme modulaire qui détecte les battements du cœur. | |
|  | Attendues | Obtenues |
| Issues | Lorsqu’on place son doigt dans la pince, les capteurs produisent un signal cohérent. Le programme applique la programmation modulaire (main.c cardio.c et cardio.h). | On ne récupère pas de valeurs cohérentes en utilisant le capteur infrarouge que nous avons intégré dans la pince. Nous avons donc décidé de faire une preuve de concept consistant à simuler plusieurs pulsations cardiaques à l’aide d’un potentiomètre. |

|  |  |  |
| --- | --- | --- |
|  | Envoyer les valeurs acquises sur le port série de l’Arduino. | |
|  | Attendues | Obtenues |
| Issues | Les valeurs des capteurs sont envoyées au moniteur série de l’Arduino en continu avec la durée en millisecondes ET la valeur du pouls. | On acquiert dans le moniteur série des valeurs cohérentes émanant du potentiomètre. |

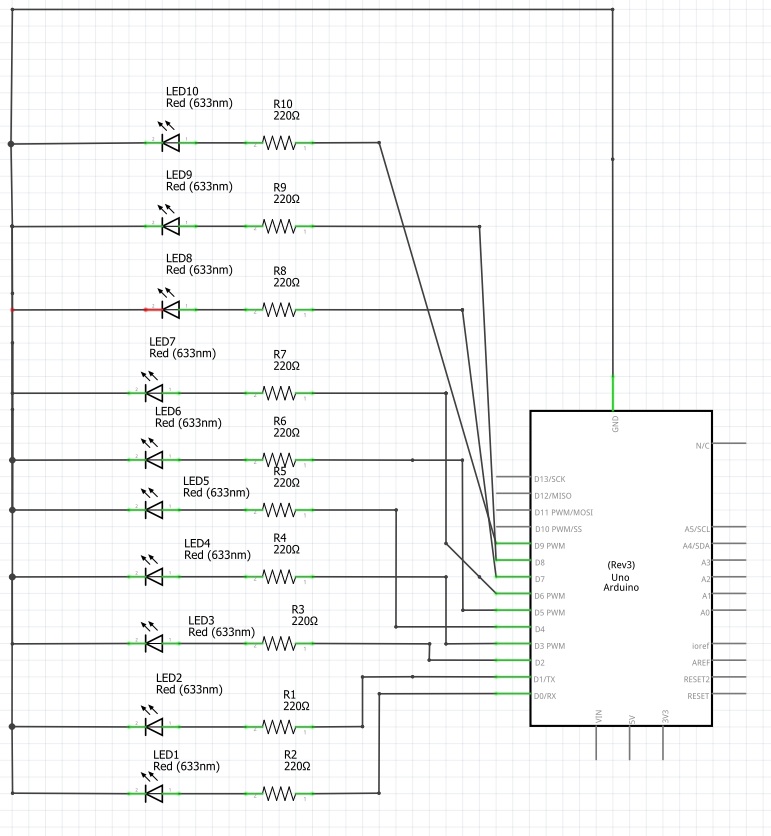
3.2 Module cœur de LEDS :

3.2.1) Définition des attentes

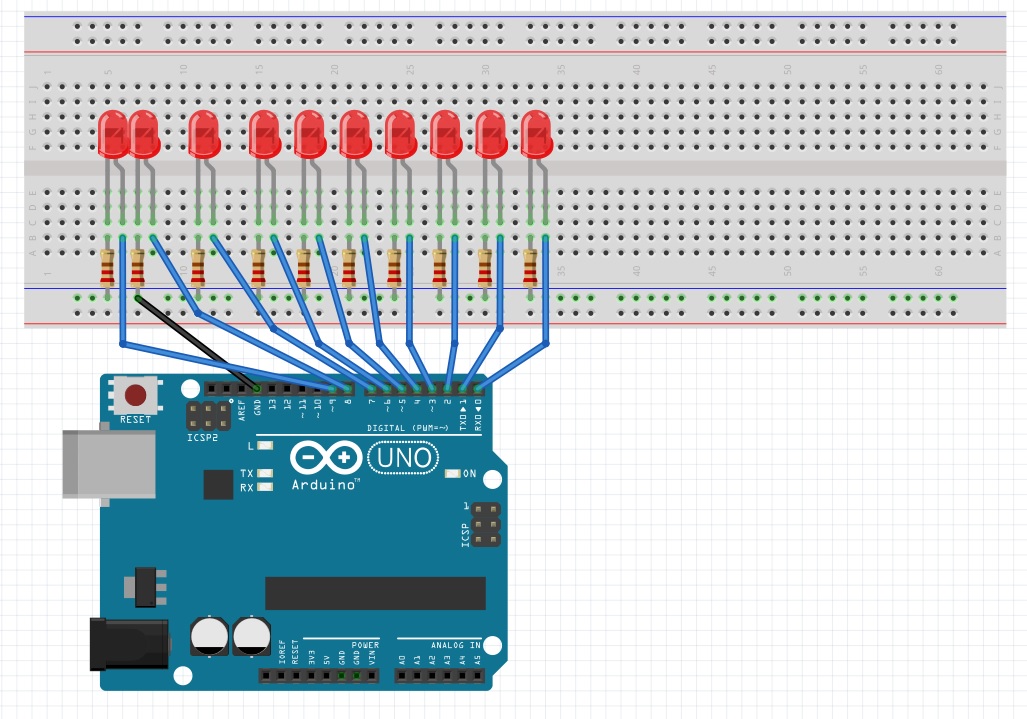
Le but de ce module est de pouvoir réaliser une “animation lumineuse” du rythme cardiaque mesuré dans le module précédent. Pour cela, nous allons piloter un ensemble de LEDs avec la carte Arduino et les faire clignoter selon plusieurs modes au rythme de la fréquence cardiaque mesurée dans le module 1.

Il sera en outre possible de changer d’animation lumineuse grâce à un programme en C générant une bibliothèque Arduino “param.h”.

Voici le schéma de branchement des LEDs que nous devrions avoir (schématisé sur Fritzing).



3.2.2) Application



Voici le montage de nos LEDs sur Fritzing (note : sur ce schéma, les LEDs sont branchées sur les ports de 0 à 9 alors qu’en pratique nous les avons branchées sur les ports de 3 à 12 mais le branchement reste similaire).

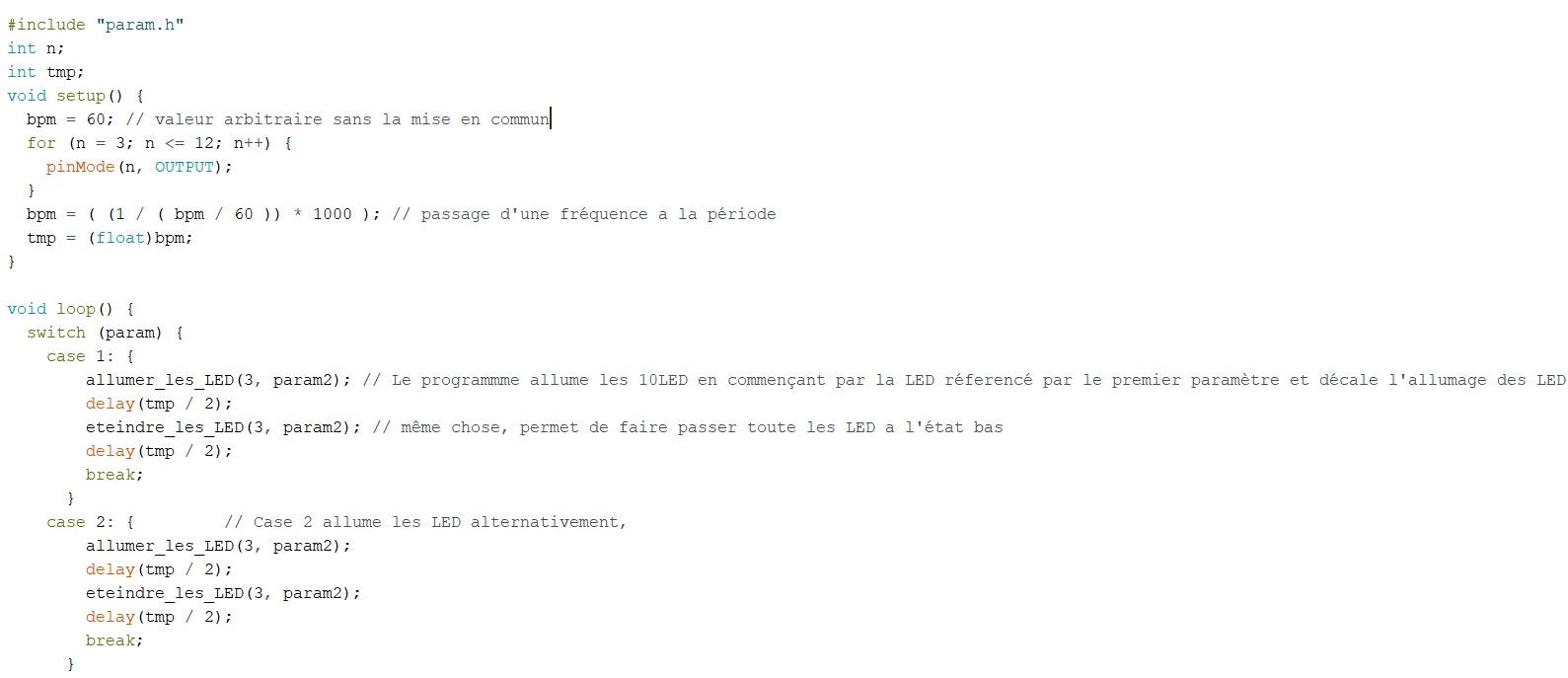
Ensuite, nous avons pu réaliser 3 modes de clignotement des LEDs :

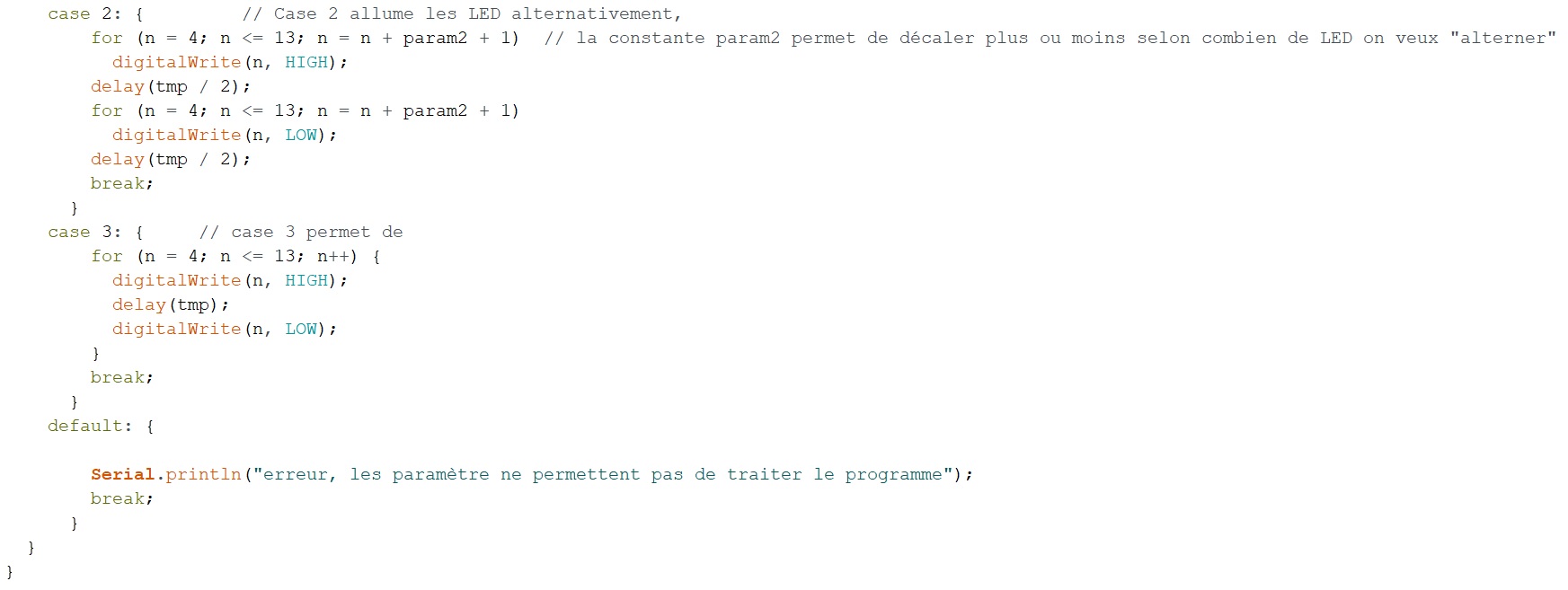
* Elles peuvent clignoter toutes en même temps ;
* Elles peuvent clignoter alternativement (certaines sont allumées d’autres, non) ;
* Elles peuvent clignoter en “chenille” c’est-à-dire l’une après l’autre (toujours par rapport au rythme cardiaque).

Les modes de clignotements implémentés ont tous été filmés lors de leur fonctionnement, les vidéos sont accessibles ci-dessous :

Voilà le programme réalisé dans un premier temps sous Arduino :





Ce programme fonctionne en théorie mais dans la pratique, étant donné que la carte Arduino doit servir à piloter le premier module également (récupérer les valeurs, ...), l’utilisation de la fonction delay n’est pas judicieuse (car la fonction delay va stopper l’Arduino le temps du delay, et le temps doit pouvoir être réparti entre les deux “pilotages”).

Une solution serait de définir un temps de “swapping” comme un temps spécifique attribué à chaque traitement (module 1 et module 2) afin d’alimenter un module, puis l’autre module très peu de temps après (ce qui donnera l’illusion que les deux modules sont alimentés simultanément en fixant un temps alloué très faible et permettrait à ce programme de fonctionner).

Néanmoins, nous avons préféré une autre solution lors de la mise en commun, c’est de pouvoir piloter les LEDs à chaque fois que le module 1 détectera une valeur, c’est à ce moment que nous pourrons allumer puis éteindre les LEDs en fonction du rythme cardiaque.

Voici à quoi ressemble le programme relié du module 1 et 2 :

[Code Arduino\_Modules 1 et 2](https://github.com/LeBescherelle/Project-1/tree/master/Code_Arduino_module_1__2)

On pilote le choix du mode de clignotement des LEDs par un petit programme réalisé en C qui permet de générer le param.h qui contient la définition de deux constantes :

* Param donne la valeur de la constante de choix principale ;
* Param2 est nécessaire lorsqu’on choisit le mode alterné pour pouvoir faire clignoter la LED, Param2 est initialisé (lors du lancement du programme à 0 par défaut, et à 1 lorsque Param = 2)

La troisième partie du module était de rédiger un programme en C qui va servir à afficher un menu de sélection puis à générer le fichier param.h. On a donc le programme suivant :

[Code C\_Module 2](https://github.com/LeBescherelle/Project-1/tree/master/Code_C_module_2)

|  |  |  |
| --- | --- | --- |
|  | Afficher un menu utilisateur dirigeant vers les différentes fonctions du programme. | |
|  | Attendues | Obtenues |
| Issues | Dirige et applique les modes :  - normal,  - alterné,  - chenillard… | Il est possible de naviguer dans le menu et d’appliquer les différents choix offerts. |

3.2.3) Interprétation des résultats

|  |  |  |
| --- | --- | --- |
|  | Allumer les LEDs en fonction des battements du cœur. | |
|  | Attendues | Obtenues |
| Issues | Le cœur s’allume quasi-instantanément à chaque battement. | Le rythme d’allumage des LEDs correspond à la fréquence cardiaque simulée. |

|  |  |  |
| --- | --- | --- |
|  | Créer différentes illuminations pour les LEDs et les piloter individuellement. | |
|  | Attendues | Obtenues |
| Issues | Les illuminations  attendues doivent être implémentées dans un fichier header. | Toutes les illuminations ont été implémentées et testées. |

|  |  |  |
| --- | --- | --- |
|  | Générer un fichier texte param.h. | |
|  | Attendues | Obtenues |
| Issues test 1 | Après le choix du mode des LEDs, cela génère un fichier texte qui fonctionne sur Arduino. | Le premier fichier généré est correct mais lors de la seconde génération, le contenu précédent n’est pas écrasé, le nouveau contenu est ajouté en plus. |
| Issues test 2 | Après le choix du mode des LEDs, cela génère un fichier texte qui fonctionne sur Arduino. | Le param.h est correct et reconnu par l’IDE Arduino. |

3.3 Module Processing et acquisition de données.

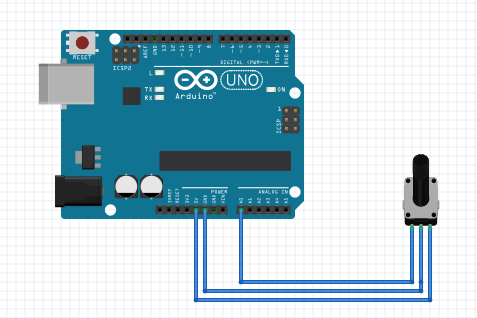
3.3.1) Définition des attentes

Le but de cette partie est de **récupérer** les données **acquises** par l’Arduino et de les classer au sein d’un **fichier .csv** généré automatiquement. Ce fichier pourra ensuite être lu manuellement dans un tableur ou par le programme programmé dans la partie 3.4.

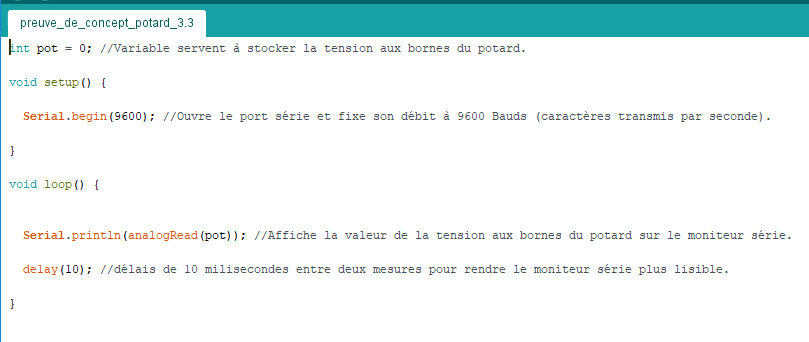
3.3.2) Application

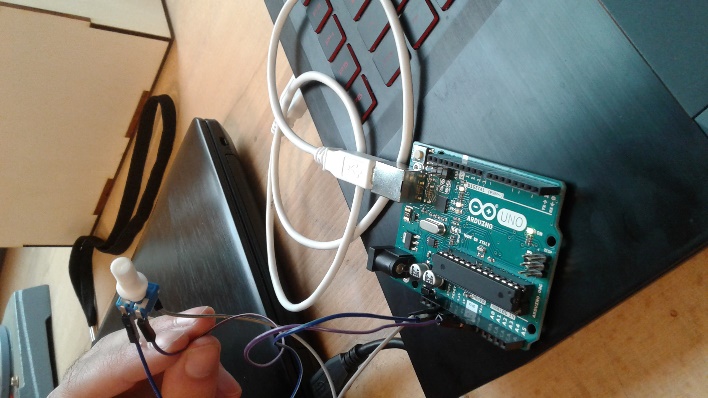
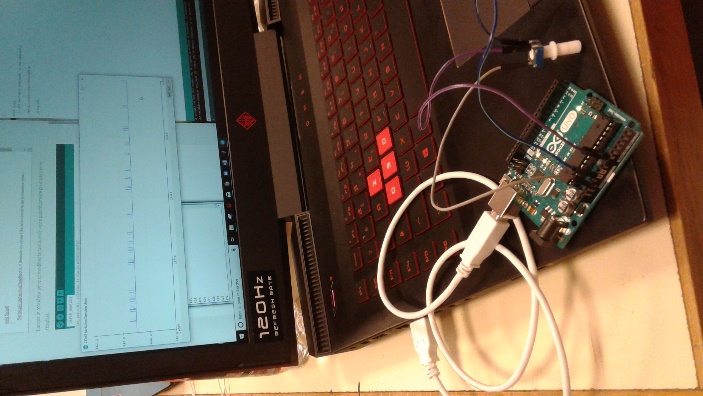
Nous avons créé une **preuve de concept** pour apprendre les tenants et aboutissants de notre partie, soit la méthode pour récupérer des données depuis l’Arduino et les organiser dans un .csv.

Nous avons branché un potentiomètre à une carte Arduino pour manipuler.



Nous avons ensuite rédigé un programme pour envoyer la valeur de la tension aux bornes du potentiomètre dans le moniteur série. Nous obtenons les sorties suivantes sur le moniteur série :





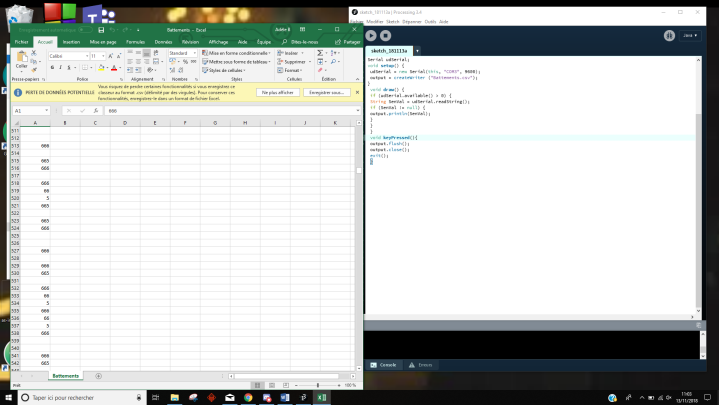


[](https://www.google.fr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwip0N6rudHeAhWQyYUKHf9UD6gQjRx6BAgBEAU&url=https://en.wikipedia.org/wiki/Processing_(programming_language)&psig=AOvVaw2nG6UHtsQyDC35JxItwXsw&ust=1542201204070541)Après avoir réussi à récupérer ces valeurs, il faut les envoyer dans un fichier .csv. Pour ce faire, nous allons utiliser **Processing,** une bibliothèque Java couplée à un IDE du même nom et qui est notamment connue pour permettre de créer simplement des rendus graphiques.

Dans notre cas, nous avons utilisé Processing pour récupérer les valeurs envoyées au moniteur série lors de l’exécution du programme Arduino pour ensuite créer un fichier nommé « *battements.csv* » pour les contenir.

Voici le lien GitHub du code réalisé sur Processing :

[Processing\_Module 3](https://github.com/LeBescherelle/Project-1/blob/master/Code_Processing_module_3/sketch_181116a.pde)



La preuve de concept est donc parfaitement fonctionnelle car elle répond aux attentes formulées au début de la partie 3.3, soit être capable de récupérer les données acquises par une carte Arduino et les ordonner dans un fichier .csv.

3.3.3) Interprétation des résultats

*Acquérir des données*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Preuve de concept |  | Cardiofréquencemètre |  |
|  | Attendues | Obtenues | Attendues | Obtenues |
| Issues | Suivre l’évolution de l’état d’un périphérique branché à l’Arduino en temps réel. | Objectif atteint | Suivre l’évolution de la fréquence cardiaque au cours du temps. |  |

*Mémoriser et organiser les données dans un fichier .csv.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Preuve de concept |  | Cardiofréquencemètre |  |
|  | Attendues | Obtenues | Attendues | Obtenues |
| Issues | Les données acquises sont classées dans l’ordre chronologique  dans un fichier tableur .csv. | Objectif atteint | Les données acquises sont classées dans l’ordre chronologique  dans un fichier tableur .csv. |  |

3.4 Module lecture et traitement de données.

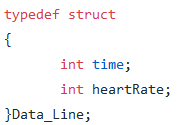
3.4.1) Définition des attentes

Le but de cette partie est de réaliser un logiciel codé en C et permettant de récupérer les données émanant de l’Arduino et triées dans un fichier .csv pour pouvoir les manipuler.

Le programme doit permettre d’afficher les données du .csv dans le bon ordre, de les trier de manière croissante ou décroissante selon le critère du pouls ou du temps. De plus il doit être possible de rechercher un pouls pour un temps spécifié ou s’en approchant le plus. Enfin, il doit être possible de calculer la moyenne des pouls mesurés sur une plage de temps définie pas l’utilisateur et de connaitre précisément combien d’entrés contient le dossier .csv.

3.4.2) Application

**Enregistrement des données :** Au lancement du programme, les données du fichier .csv sont lues et enregistrées dans la mémoire du programme.

Chaque ligne du fichier est enregistrée dans une structure qui comprend une variable pour stocker le temps et une autre pour stocker la valeur de la fréquence cardiaque qui lui est associée :

Pour stocker l’ensemble des données, le programme enregistre chaque structure dans un tableau. On a ainsi un tableau dynamique de structure.

Chaque case du tableau contient donc une structure qui contient les données d’une ligne du fichier .csv. La taille du tableau est augmentée au fur et à mesure qu’on lit les lignes du fichier .csv en lui allouant dynamiquement de l’espace mémoire.

Ainsi l’espace mémoire alloué au tableau dépend de la taille du fichier .csv et donc de la quantité de données qui doivent être stockées.

**Afficher les données dans l’ordre du fichier .csv :** Pour cette fonctionnalité, le tableau de données n’est pas utilisé car c’est dans celui-ci que sont effectués les tris demandés par l’utilisateur.

Or on souhaite afficher les données dans l’ordre du fichier .csv, donc une fonction s’occupe de lire de nouveau le fichier .csv et d’afficher les valeurs au fur et à mesure sur la console.

**Afficher les données par ordre croissant/décroissant :** Pour chaque tri choisi par l’utilisateur (croissant ou décroissant), un menu permet de choisir si le tri doit être effectué en fonction du temps ou en fonction du pouls.

Une fois le choix fait par l’utilisateur, le tableau de données est trié comme demandé. Ce tri s’effectue grâce à un algorithme de tri par sélection. Sa complexité est de O(n²) ce qui est suffisant dans notre contexte car il n’y aura pas des centaines de milliers de valeurs à trier.

Une fois le tableau trié, les valeurs sont affichées dans la console.

**Rechercher et afficher les données pour un temps particulier :**  La valeur en millisecondes du temps demandé par l’utilisateur permet d’effectuer une recherche dans le tableau de données.

La recherche se fait grâce à un algorithme de recherche séquentielle, on compare chaque valeur afin de trouver celle qui sera soit exactement le temps demandé par l’utilisateur soit la valeur inférieure qui s’en rapproche le plus.

Ainsi on peut afficher le pouls pour un temps demandé, soit le temps demandé existe et on peut ainsi donner la valeur exacte du pouls à cet instant T, soit le temps demandé n’existe pas et on prend donc le dernier pouls enregistré à partir du temps demandé par l’utilisateur.

Par exemple :

« A 1500ms, le dernier pouls enregistré était de 60 bpm à 1400ms. ».

**Afficher la moyenne de pouls dans une plage de temps donnée :** Pour afficher la moyenne de pouls dans un intervalle de temps donné, on utilise un algorithme de recherche séquentielle.

Cet algorithme va récupérer toutes les valeurs de pouls présentes dans l’intervalle de temps et en déduire la moyenne.

**Afficher le nombre de ligne de données actuellement en mémoire :** Au début du programme une fonction calcule la taille du tableau de données qui a été rempli avec les valeurs du fichier csv. Cette fonction enregistre la taille du tableau dans une variable qui est ensuite conservée pendant tout le programme. Cette variable est utilisée pour interagir sur le tableau de données.

Pour afficher le nombre de ligne de données actuellement en mémoire, il suffit donc d’afficher la valeur de cette variable.

**Rechercher et afficher le max/min de pouls :** Pour cette fonctionnalité on utilise de nouveau un algorithme de recherche séquentielle afin de trouver la valeur maximum et minimum de pouls.

Ces deux valeurs sont ensuite affichées dans la console.

**Quitter le programme :** Pour cette fonctionnalité on utilise tout simplement la fonction exit() avec comme paramètre EXIT\_SUCESS ou 0 qui permet de quitter le programme en disant que celui-ci s’est fermé sans erreur (mais par choix de l’utilisateur).

3.4.3) Interprétation des résultats

|  |  |  |
| --- | --- | --- |
|  | Afficher les données dans l’ordre du fichier .csv. | |
|  | Attendues | Obtenues |
| Issues | Au lancement du programme, le fichier .csv est importé dans le .c et chacune de ses entrées est copiée dans l’ordre au sein d’un tableau. | Le .csv est bien importé dans le .c et il est lu dans le bon ordre. |

|  |  |  |
| --- | --- | --- |
|  | Afficher les données en ordre croissant/décroissant (selon le temps, selon le pouls). | |
|  | Attendues | Obtenues |
| Issues | Obtenir des valeurs triées dans l’ordre croissant selon le temps ou le pouls.  Obtenir des valeurs triées dans l’ordre décroissant selon le temps et le pouls. | Les tris sont bien implémentés et produisent le résultat escompté. |

|  |  |  |
| --- | --- | --- |
|  | Rechercher et afficher les données pour un temps particulier. | |
|  | Attendues | Obtenues |
| Issues | Pouvoir connaître la fréquence cardiaque à l’instant T. | Il est possible de rechercher un pouls en spécifiant l’instant. |

|  |  |  |
| --- | --- | --- |
|  | Afficher la moyenne de pouls dans une plage de temps donnée. | |
|  | Attendues | Obtenues |
| Issues | Calcule la moyenne de tous les pouls compris entre deux bornes temporelles T1 et T2. | Il est possible de spécifier les bornes T1 et T2 et le calcul de la moyenne des pouls est correct. |

|  |  |  |
| --- | --- | --- |
|  | Afficher le nombre de lignes de données actuellement en mémoire. | |
|  | Attendues | Obtenues |
| Issues | Le nombre de lignes affichées doit être le même que le nombre de lignes dans le .csv. | Le décompte du nombre de lignes est implémenté et est pleinement fonctionnel. |

|  |  |  |
| --- | --- | --- |
|  | Rechercher et afficher les max/min de pouls (avec le temps associé). | |
|  | Attendues | Obtenues |
| Issues | Affiche la valeur la plus élevée et la plus faible associées au temps T où elles « ont eu lieu ». | La recherche de la valeur maximale et minimale du pouls avec le temps associé est fonctionnelle. |

|  |  |  |
| --- | --- | --- |
|  | Quitter l’application | |
|  | Attendues | Obtenues |
| Issues | L’application se ferme à l’exécution de la commande correspondante. | L’appui sur la touche 8 ferme le programme. |