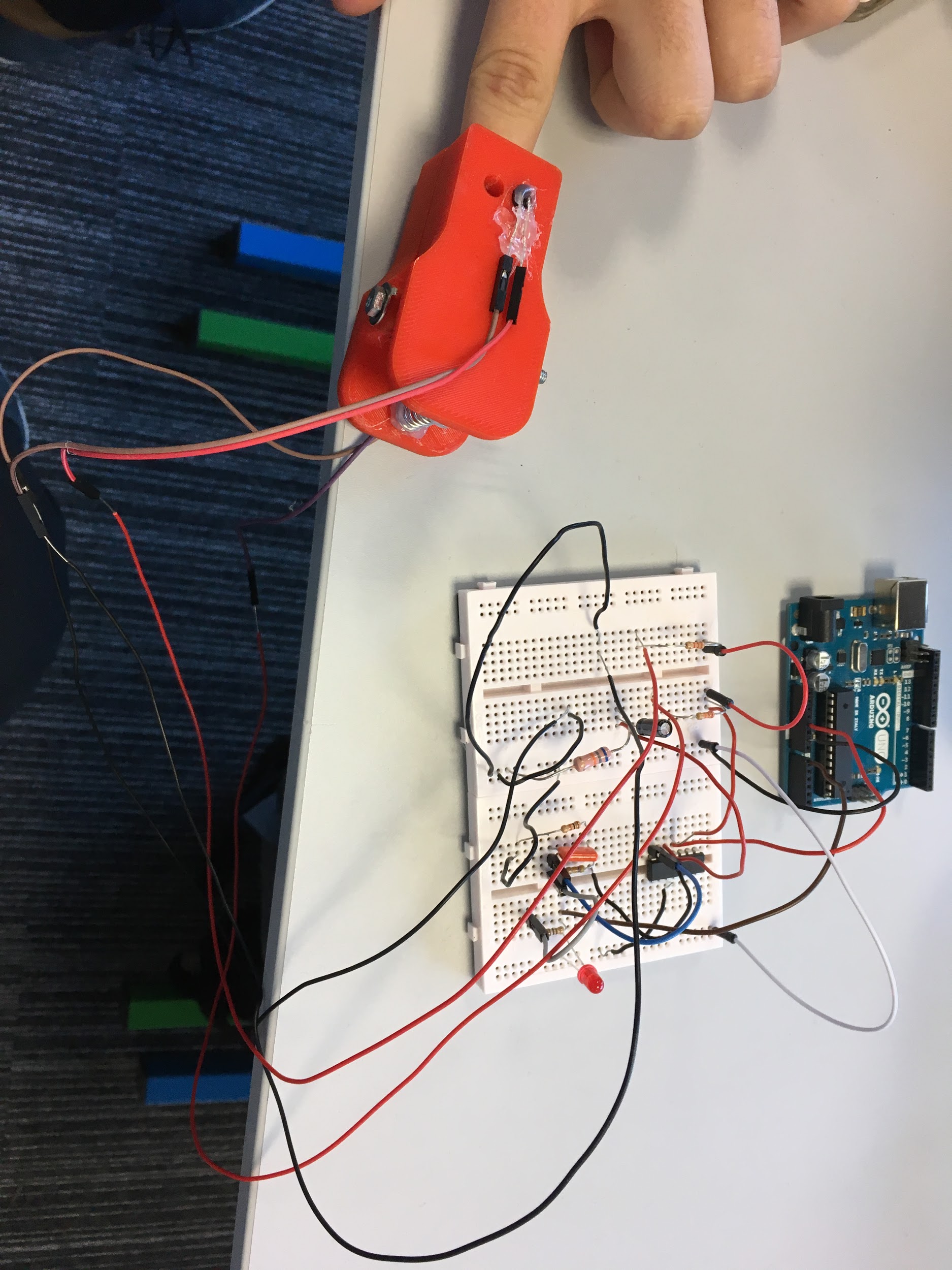
-------------------------------------------------------------------------------------------------------------------------------------

Documentation technique

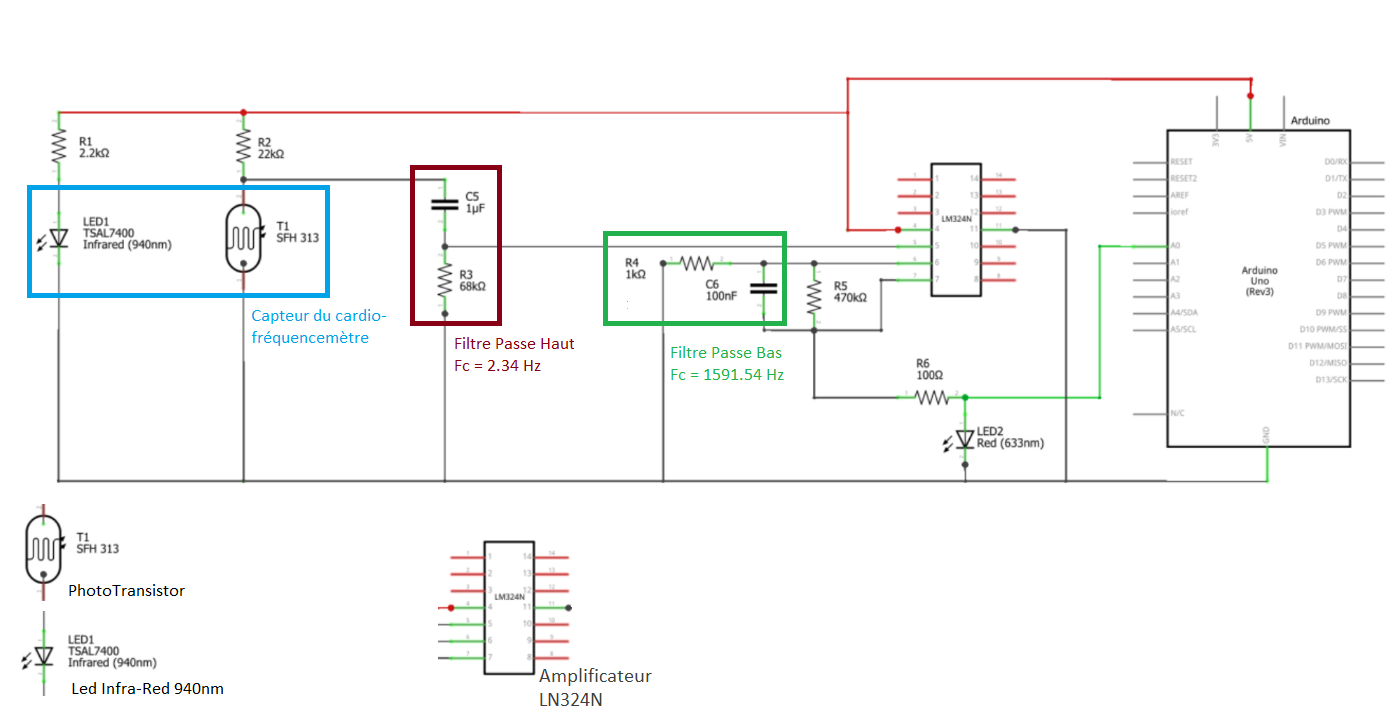
-------------------------------------------------------------------------------------------------------------------------------------

MODULE 1 :

Câblage du Module Cardio :



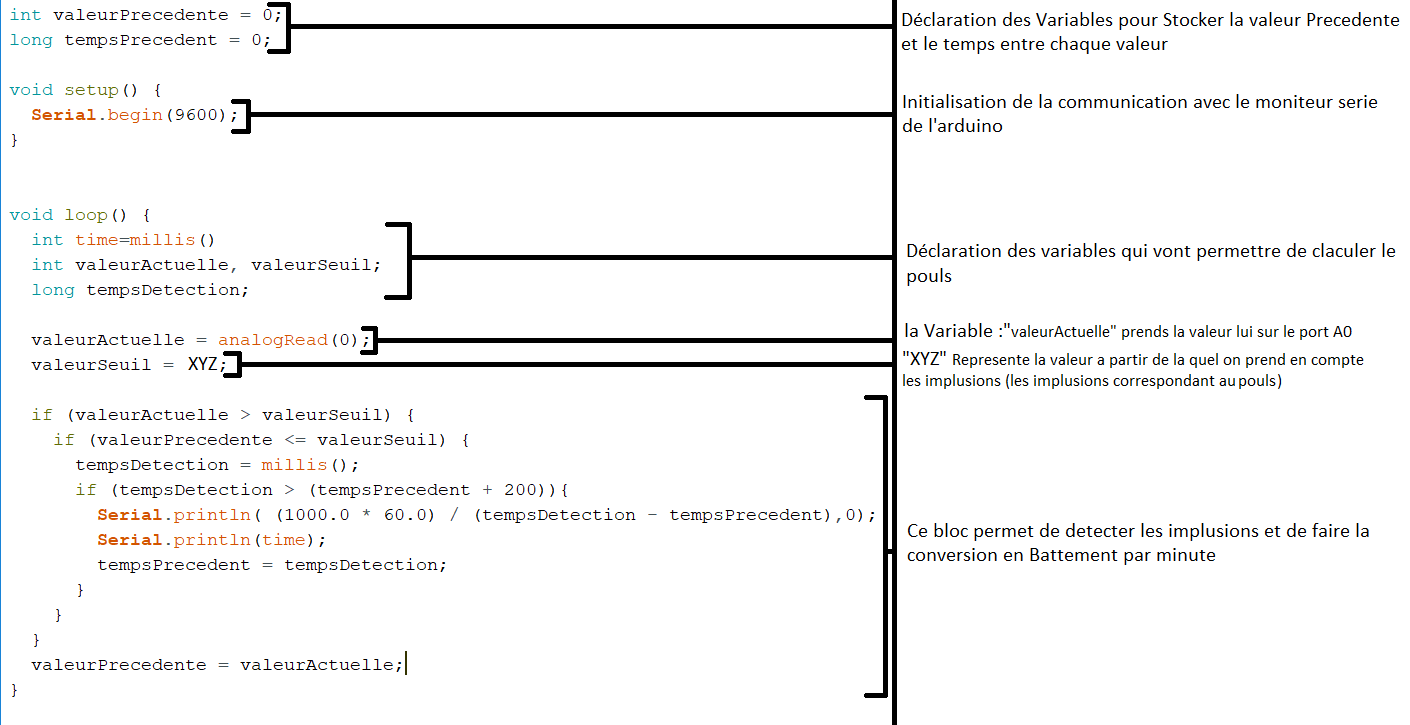
Aspect technique en vue schématique:



Dans ce montage nous utilisons un amplificateur opérationnel car la tension en sortie du Phototransistor est trop faible pour que l’on puisse l'analyser avec l'Arduino.

Dans la module Cardio nous récupérons le Pouls du Sujet Test grâce au Phototransistor, ce dernier va délivrer une tension qui sera dirigée vers la suite du montage lorsque des “ondes lumineuses” (selon la quantité de lumière) qui atteint le Phototransistor, il laissera passer plus ou moins de tension vu que nous voulons récupérer des ondes infra-rouges . Le phototransistor nous délivrera une tension très faible, c’est pourquoi nous avons un amplificateur de signal ce qui nous permet de booster notre signal en sortie de notre Phototransistor pour qu'il puisse être lu par notre Arduino.

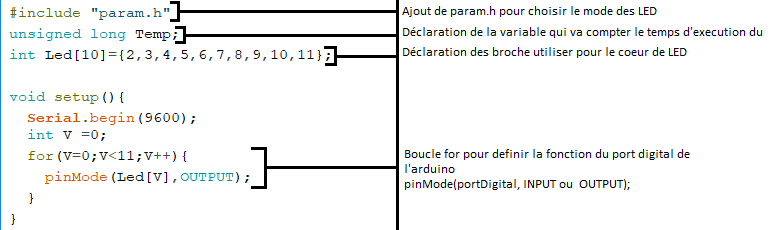
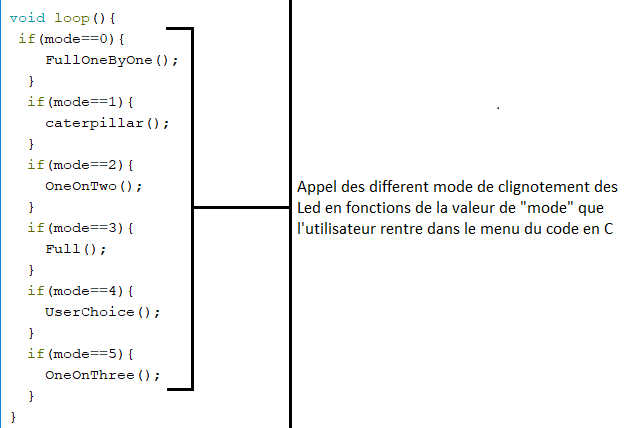
Code Permettant de calculer la Fréquence cardiaque :

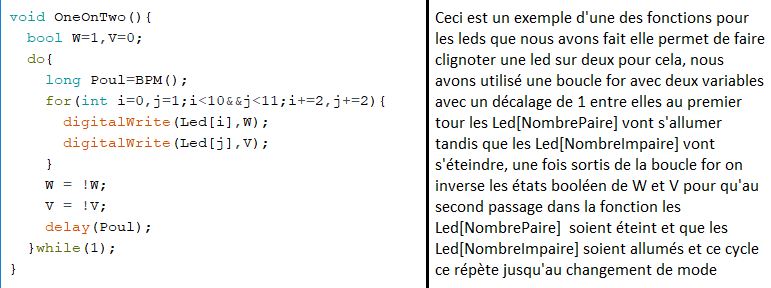


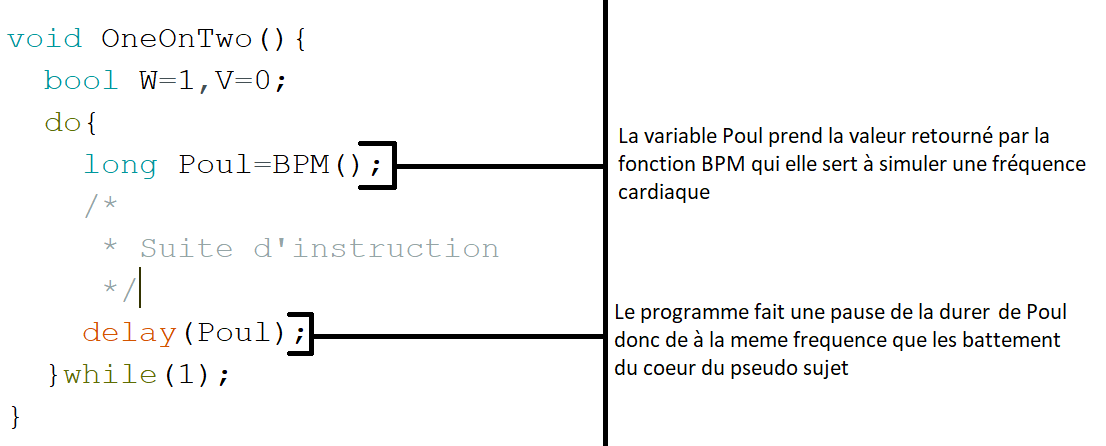
Dans la suite du projet nous n'utiliserons pas le module cardio pour avoir une fréquence cardiaque à cause de certain points techniques qui sont les suivants :

* Le manque de Précision du Phototransistor, en effet ce dernier ne prend en compte que les ondes infrarouges mais aussi les ondes du spectre lumineux ce qui crée des interférences, lorsqu’on mets notre doigt dans la pince on détecte bien un pouls mais si on déplace un peu le doigt ou la pince, ou qu’on modifie l'environnement autours de la pince notre valeur moyenne changera ce qui faussera complètement les résultats.
* Il est spécifié dans la grille d'évaluation d’avoir un simulateur de rythme cardiaque sur la carte Arduino (En C Arduino).

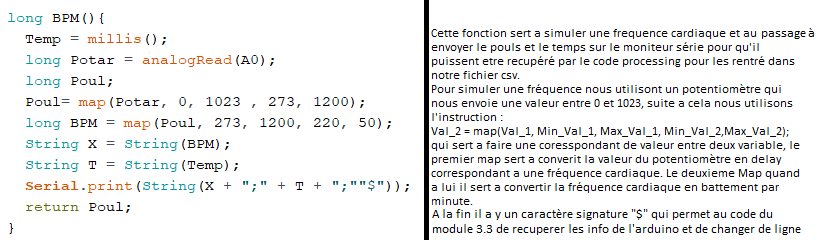
MODULE 2 :



Je ne vais pas décrire chaque mode de LED puisqu'ils sont essentiellement constitués de la même manière. Voir ci-dessous le mode qui permet d’allumer une LED sur deux.

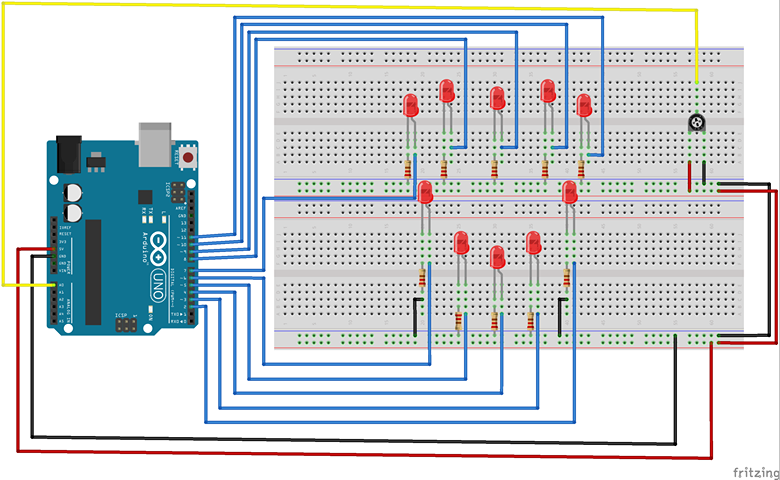


La Fonction BPM :

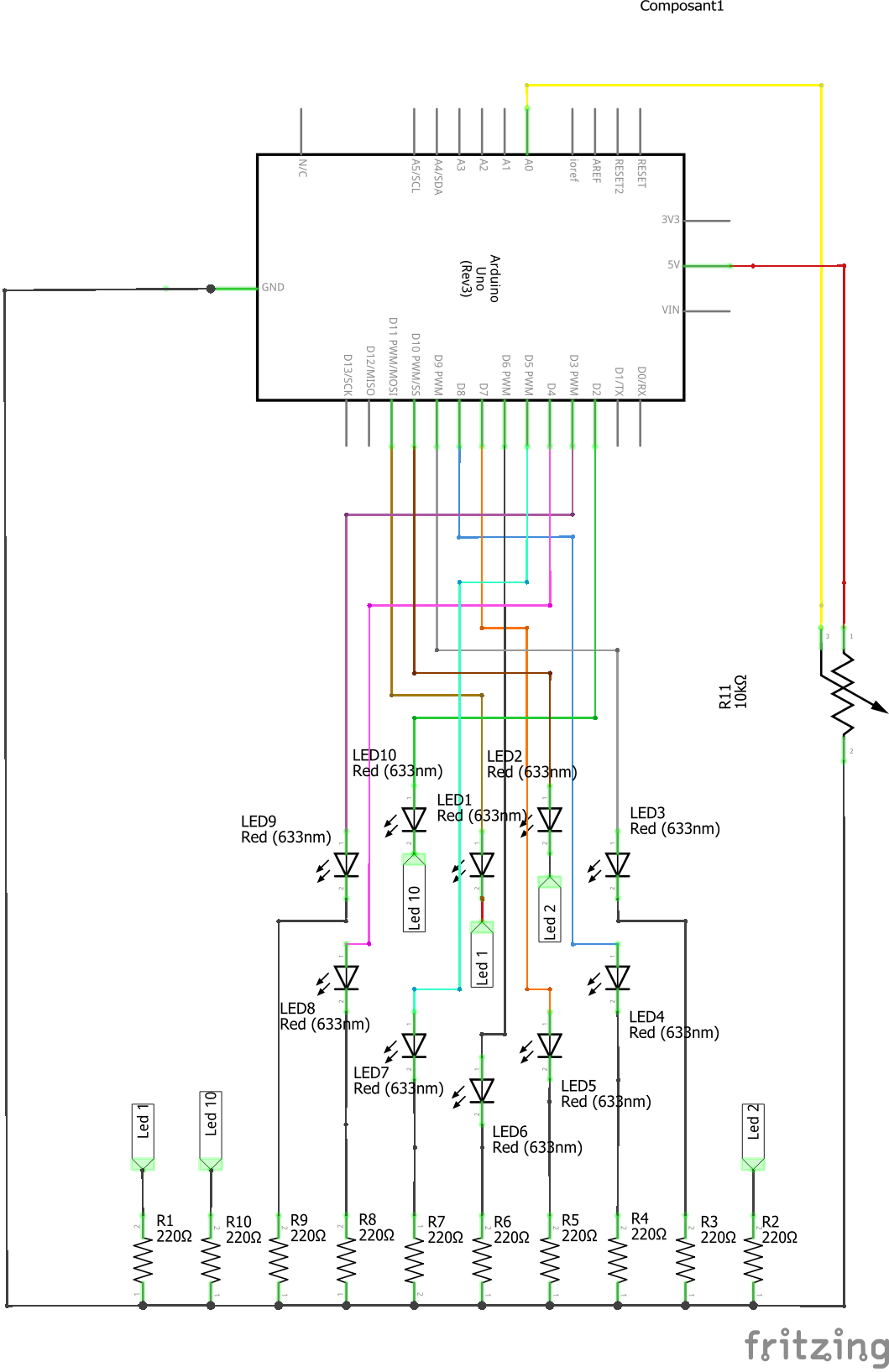


Vue platine du cœur de LED sous fritzing :

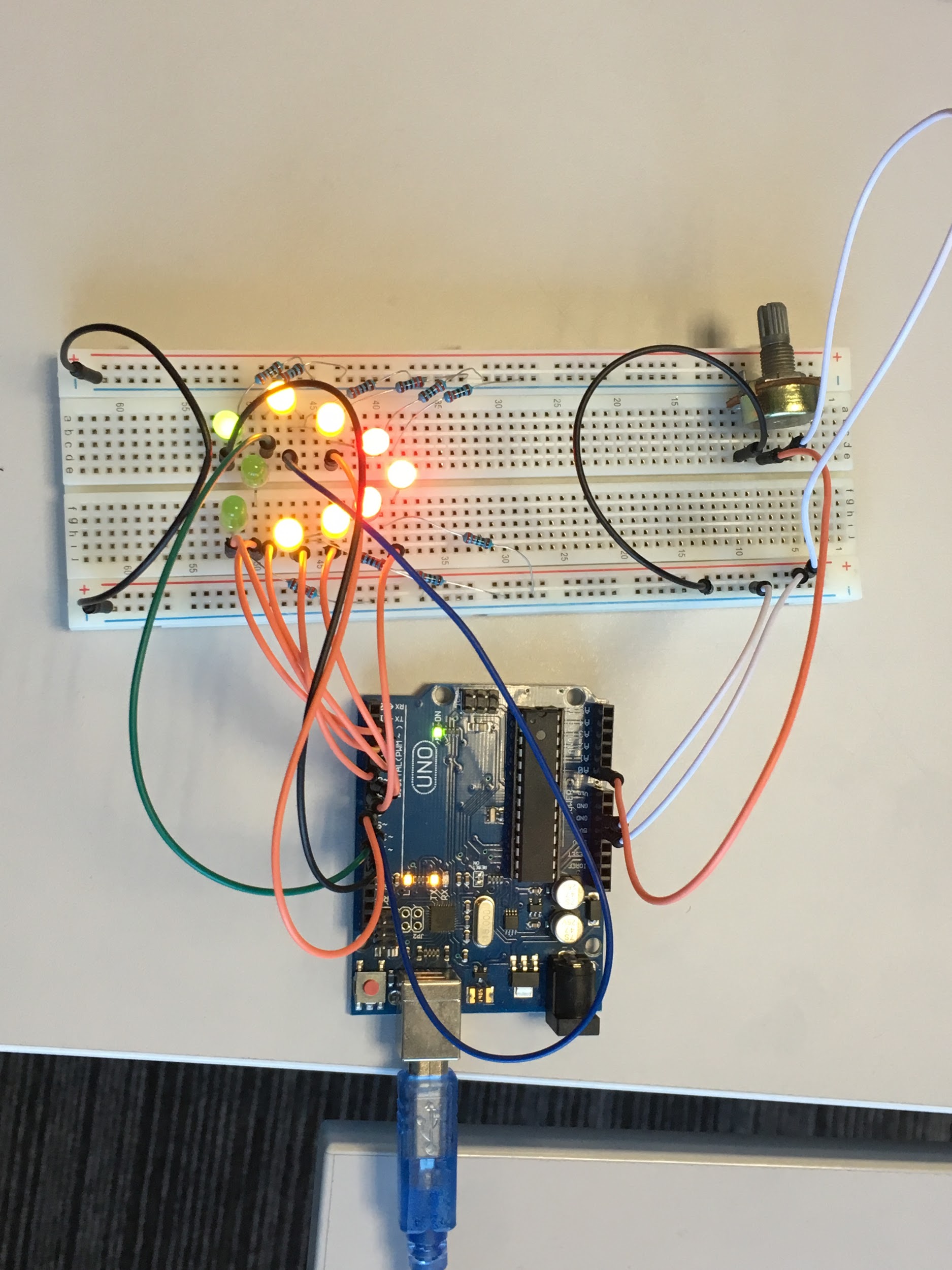
* Résistance utilisé 220Ω
* LED Rouge 633nm
* Potentiomètre 10KΩ



Vue Schématique du cœur de LED :



Câblage physique :



troisième partie :

Pour la création du menu pour l’affichage des LEDs, j’ai créé la fonction **ModeLED** ou les différents choix d’allumage sont énumérés. Ensuite on demande à l’utilisateur quel mode d’affichage il souhaiterait voir à l’aide d’un **scanf**. La valeur entrée par l’utilisateur est stockée dans une variable et ensuite envoyée à une fonction comportant un switch avec les différents code permettant les différents affichage. Le tout est codé dans un **do while** ce qui nous a permis d'insérer une fonction **Leave** qui retourne au menu initial (voir Module 4).

MODULE 3 :

Pour la réalisation du Module 3, nous avons tout d'abord téléchargé processing sur internet. Nous avons donc ensuite créé un fichier **.pde** dans processing et inséré le code fournis.

Par la suite, nous avons donc essayé de brancher l'Arduino avec le code et le montage du Module Cardio mais nous avons constaté que premièrement notre récupération de données ne s'arrêtait jamais et deuxièmement que les valeurs récupérées n’étaient pas précises et illogiques. Nous avons déduit que ces résultats étaient dus aux interférences et à l’utilisation de matériel pas forcément super précis.

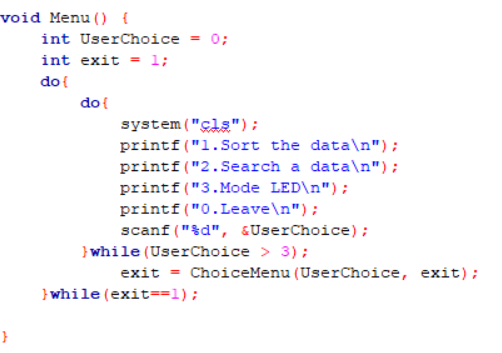
Pour remédier à ce problème, nous avons tout d'abord créé une variable **stop** dans le code processing que l’on a incrémenté à chaque boucle et qui s'arrête donc lorsque i est égal à 25. Nous avons choisi 25 pour que nous ayons un certain nombre de données sans que le processus ne prenne trop longtemps. Pour obtenir des valeurs plus cohérentes et plus réalistes nous avons réalisé un montage avec un code Arduino simulant des battements. Nous avons implanté un potentiomètre qui nous permet de faire varier le pouls entre 50 et 220 battements par minute.

Ensuite nous avons relancé le programme et nous avons obtenu un fichier **.csv** avec nos valeurs. Cependant il y avait un problème au niveau des retours à la ligne; les différentes valeurs enregistrées ne se mettaient pas systématiquement sur des lignes différentes. Nous avons donc modifié un autre ligne du code (**String SenVal = udSerial.readStringUntil('$')**, qui nous a permis de mettre après chaque relevé d’information le signe “$”, qui signifie donc le retour à la ligne.

MODULE 4 :

Tout d'abord, il était demandé de répartir les différentes fonctions de notre code dans différentes bibliothèques. Nous avons donc créés plusieurs librairies dont une pour les menus.

**Menu()**



Il fallait que l’utilisateur ait accès à plusieurs fonctionnalités par exemple l’affichage de nos LEDs ou les différentes actions possible sur les informations répertoriées dans le fichier csv. Nous avons donc créé un pré-menu comportant trois choix possible:

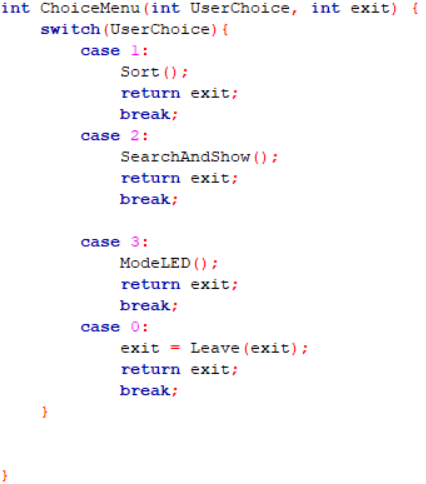
- Le tri des données

- La recherche de donnée

- Affichage des LEDs

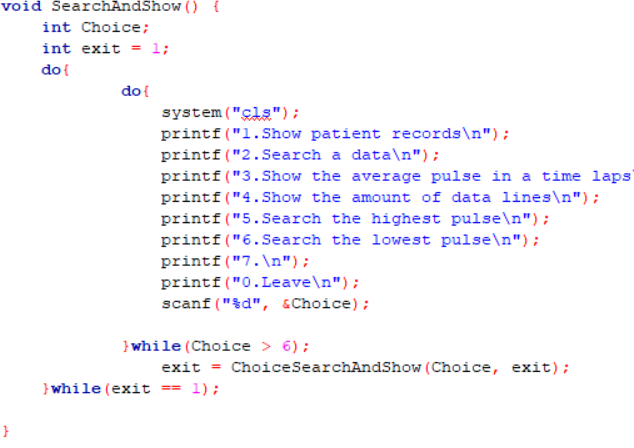
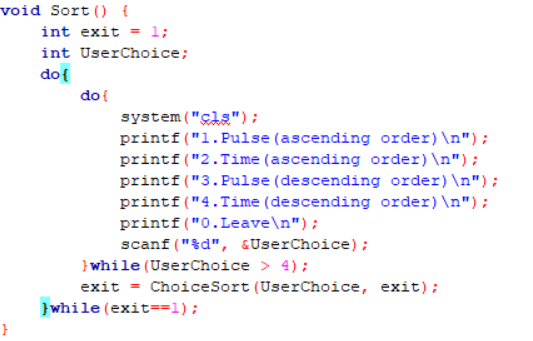
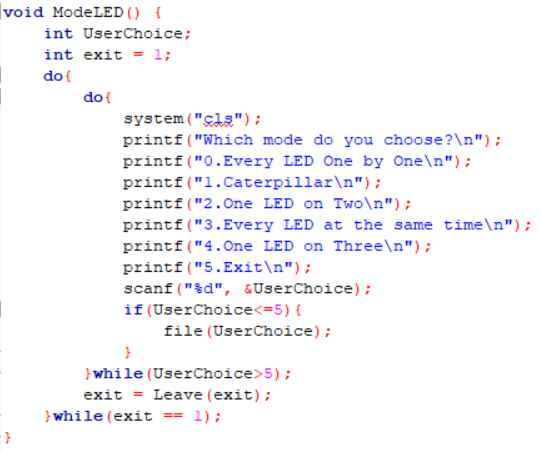
Ensuite l’utilisateur choisit l’action qu’il souhaite réaliser à l’aide d’un **scanf** qui envoie donc la variable **Userchoice** dans une fonction avec un switch comportant les fonctions des trois choix possibles.

**ChoiceMenu**()



Le nouveau menu souhaité est donc affiché dans la console et on peut de la même manière choisir plus précisément l’action souhaitée

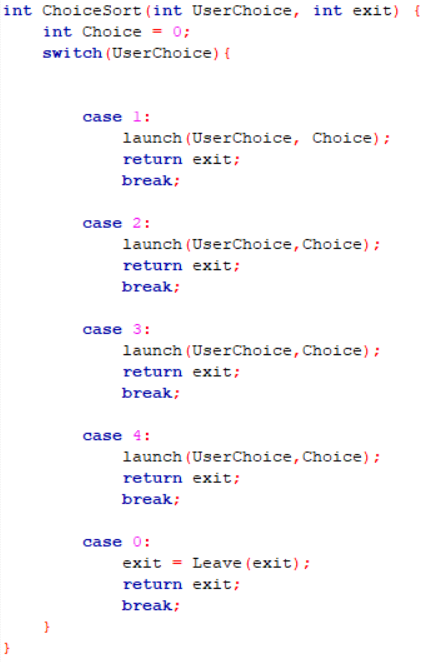
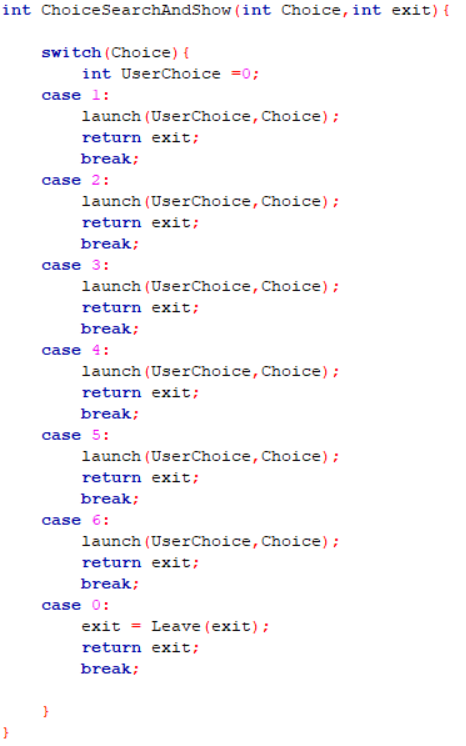
**Sort()** : **SearchAndShow(): ModeLED() :**

****

Chaque menu appel en fait une fonctioncomportant un switch ou chaque case appel également un fonction correspondante a la bonne action demandé.

Le **ModeLED** est un peu particulier car en fonction du choix de l’utilisateur, les fonctions qui sont dans dans le switch vont écrire dans un fichier header(**param.h**) qui va ensuite être récupérer par un code Arduino.

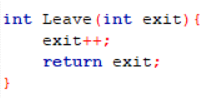
**ChoiceSort()** : **ChoiceSearchAndShow()** :

Tous les menu ont été mis dans une librairie ou il n’y avait uniquement les menus (menu.c/menu.h). Les fonctions avec les **switch** étaient dans une autre librairies où nous avons mis toutes nos fonctions réalisant une action ( actions.c/actions.h). Dans notre main il y a donc uniquement une fonction appelé Menu et c’est tout.

Dans nos menu, c’est à dire dans notre bibliothèque de menu, on a également ajouter une fonction **Leave** qui permet de revenir au menu précédent. Cela permet de revenir au menu précédent pour pouvoir faire une action supplémentaire sans devoir quitter le programme et recompiler.

**Leave()**

****

**ChoiceSort()** :

la valeur Choice vaut 0.

Switch Case :

La fonction **launch** appelle la fonction **contact\_infos** et **free\_contact\_infos**

**ChoiceSearchAndShow() :**

Switch Case :

lance la fonction ***launch***(UserChoice, Choice)

La fonction **launch** appelle la fonction **contact\_infos** et **free\_contact\_infos**

**contact\_infos {**

on ouvre le fichier .**csv** avec la fonction fopen, on le lit et on récupère avec fgets,

on a limité le nombre de variable par chaîne de caractère (ici BUFF\_SIZE = 128).

on alloue dynamiquement la place nécessaire à la lecture de toutes les données dans le .**csv**

on sépare grâce à un strtok (dont on a défini au préalable, le caractère de séparation (DELIM), et la taille maximale d’une chaîne(buff)).

une boucle a été fait au préalable et ensuite on rentre dans un switch case :

A chaque tour de la boucle, on incrémente le pointeur infos->poul avec la valeur du poul,

si dans le Menu, on a choisi de trier le pouls, alors on créer un tableau avec les valeurs du pouls.

on fait pareil pour le temps (en millisecondes)

si dans le Menu, on a choisi de trier le le temps, alors on créer un tableau avec les valeurs du temps.

une fois sorti de la boucle, les valeurs sont triées grâce à un **tri à bulle**  dans l’ordre croissant ou décroissant (selon ce que l’on a choisi dans le menu **ChoiceSort**.

Si l’on est passé par la menu **ChoiceSort** la variable Choice prend la valeur 0, du coup il y a un affichage de toutes les valeurs triées.

selon le choix réalisé dans le menu **ChoiceSearchAndShow**, la variable Choice a pris une valeur qui permet ou non de rentrer dans la boucle qui affiche la valeur maximale ou minimale du poul(on affiche soit la première soit la dernière valeur du tableau), le nombre de ligne dans le fichier .csv(on a incrémenter une variable j en fonction du nombre de tour que fait la boucle qui incrémente le pointeur infos->poul ou infos->millis, donc la valeur de j correspond au nombre de ligne, on affiche donc la valeur de j) .

on ferme le fichier avec fclose().

**}**

**free\_contact\_infos{**

On supprime l’espace prévu lors de l’allocation dynamique dans la fonction **contact\_infos**.

**}**

**Tri\_à\_bulle {**

Nous avons choisis un tri à bulle car nous n’avons pas besoin de gérer et de trier un très grand nombre de valeur, de plus aucune contrainte de temps n’as été émise.

De plus, ce tri possède un réel avantage, si l’on veut rajouter une donnée dans un tableau déjà trié, le programme n’aura qu’à faire un seul aller afin de disposer au bon endroit la valeur.

fonctionnement ;

on parcourt le tableau autant de fois que nécessaire afin de trier le tableau de données, ensuite on compare si la valeur 1 est plus grande (tri croissant) ou plus petite(tri décroissant) que la valeur 2;

on échange cette valeur si besoin jusqu’à ce qu’il n’y ai plus aucune valeur à déplacer**}**

**typedef struct {**

on déclare la variable poul et la variables millis

**}**on utilise le raccourci contact\_infos pour parler de la structure

**typedef enum{**

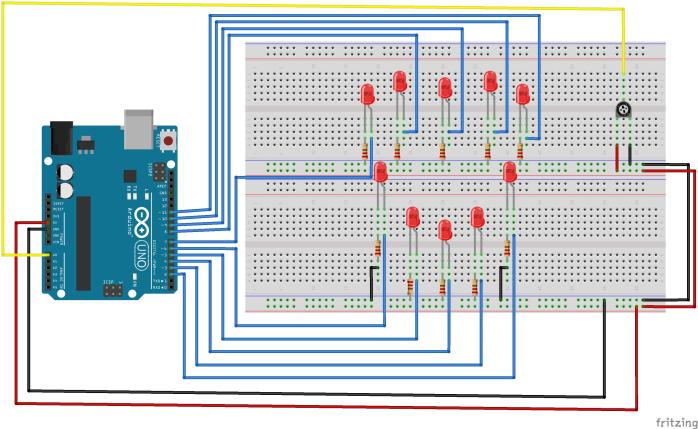
on déclare la variable POUL, MILLIS,NB\_TYPE

**}**on utilise le raccourci infos\_type pour parler de la structure

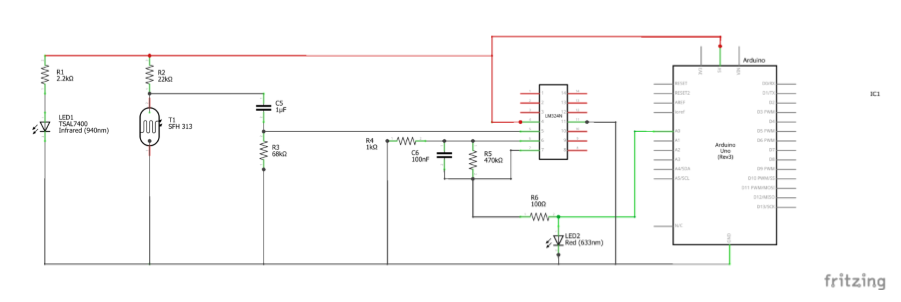
**-------------------------------------------------------------------------------------------------------------------------------------**

**MODULE ARDUINO**

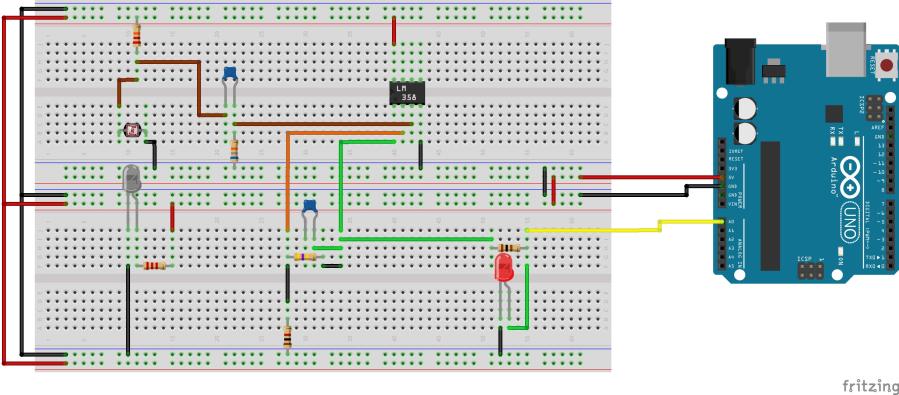
**-------------------------------------------------------------------------------------------------------------------------------------**

****

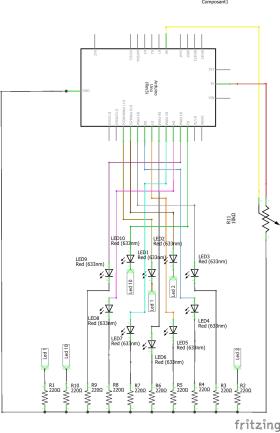
**Module cœur de LED**



**Module cardio**

****

**Module Cardio**

****

**Module cœur de LED**