19/11/18



Projet fondamentaux scientifiques

sous-titre

Présenté par : Delpon Leo   
LAbarbe Xavier  
Casenave ines

Nom de l’entreprise

# Projet fondamentaux scientifiques

## Introduction du projet

### Contexte

Après un sabotage industriel, Hexart Care se retrouve sans aucune sauvegarde de leur projet de lecteur portatif de fréquence cardiaque. De plus le concepteur du projet a disparu. L’entreprise fait appel aux étudiants de première année afin de pouvoir relancer le projet et le finir sachant qu’on dispose de peu d’informations.

### Objectif

Notre objectif est de recréer le projet en nous aidant des quelques ressources disponibles. Nous devrons livrer une pince portative lectrice de fréquence cardiaque fonctionnelle. De plus nous aurons un montage à faire pour afficher le pouls par un allumage de LEDs.

### Perimetre

Pour réaliser la pince on devra se concentrer sur une partie électronique avec des montages, et une autre partie centrée sur de la programmation.

### Description fonctionnelle des besoins

#### Module 1

1. **Construire un cardiofréquencemètre** avec méthode de **photo pléthysmographie** (détecter battements du cœur en mesurant le volume de sang grâce à une source de lumière et d’un détecteur). Nous avons à disposition un schéma et des pinces imprimées.
2. Programmer la carte Arduino et l’organiser en deux fichiers : main.c, cardio.c, cardio.h. Ce programme contiendra des fonctions pour récolter infos et calculs pour avoir le pouls.

* On doit avoir le nombre de millisecondes depuis le démarrage du programme Arduino et le pouls calculé étant donné les informations récupérées par le récepteur IR

#### Module 2

**Réaliser un affichage original du pouls à travers un cœur de LEDS rouges**

* On veut que le cœur soit allumé à chaque battement de cœur détecté. Chaque LED doit être contrôlée de façon unitaire par l’Arduino

1. Modéliser le montage électronique sur **Fritzing** et de le réaliser
2. Programmer sur la carte Arduino les LEDS avec fichier param.h

* Voir comment allumer les LEDS
* Créer un module en C pour gérer le cœur ( cœur.c, cœur.h, param.h)

1. Programmer en Langage C (pas en C Arduino). Créer le fichier param.h du 2. Organisé en 3 fichiers (ou plus)

* Main.c : contient main et lance le menu
* Menu.c (+.h) : toutes les fonctions nécessaires pour créer menus de paramétrages du cœur de LEDS
* generationCode.c (+.h) : fonctions pour créer param.h contenant paramétrage du cœur de LEDS. Le fichier est indispensable pour la compilation du module.

#### Module 3

**On veut récupérer les valeurs du pouls lues sur la sortie série de l’Arduino et les enregistrer dans un fichier CSV.**

* Télécharger Processing sur la machine qui fera les relevés
* Ouvrir avec Processing le fichier ConvertisseurSerialCSV.pde
* Modifier la ligne udSeria = new Serial(this, Serial.list()[0], 9600)
* Exécuter le programme pour récupérer les informations et générer un programme

#### Module 4

**Programmation de code en Langage C. On veut charger en mémoire les infos du fichier CSV générés et traiter les données pour afficher certaines informations à partir de données brutes.**

1. Définir la structure de données pour charger en mémoire toutes les infos du fichier .csv
2. Code organisé en

* main.c
* menu.c (+menu.h) : fonctions pour afficher les menus
* data.c (+data.h) : fonctions pour manipuler la structure de données et faire la lecture des fichiers
* actions.c (+.h ) : fonctions pour traiter les données

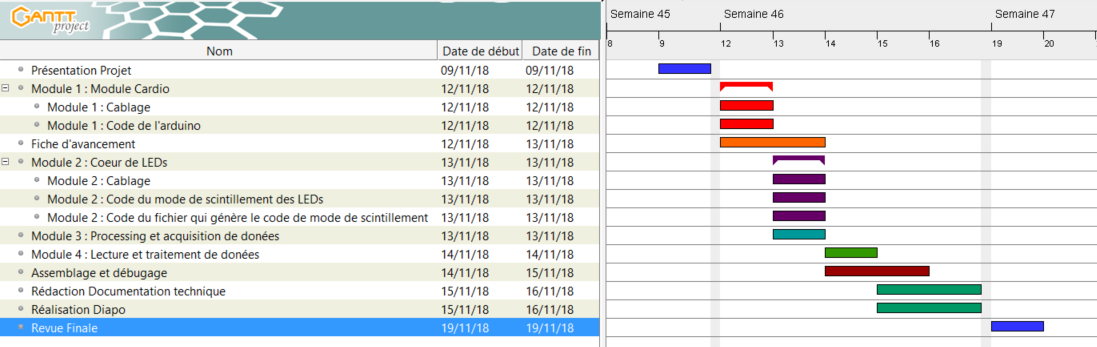
1. Le programme doit démarrer, lire le fichier .csv et le charger en mémoire, il doit afficher un menu avec les fonctionnalités :

* Afficher les données dans l’ordre du fichier .csv
* Afficher les données en ordre croissant/décroissant (selon le temps/pouls)
* Rechercher et afficher les données pour un temps particulier
* Afficher la moyenne de pouls dans une plage de temps donnée
* Afficher le nombre de lignes de données actuellement en mémoire
* Rechercher et afficher les max/min de pouls (avec le temps associé)
* Quitter l’application

## Répartition du travail a

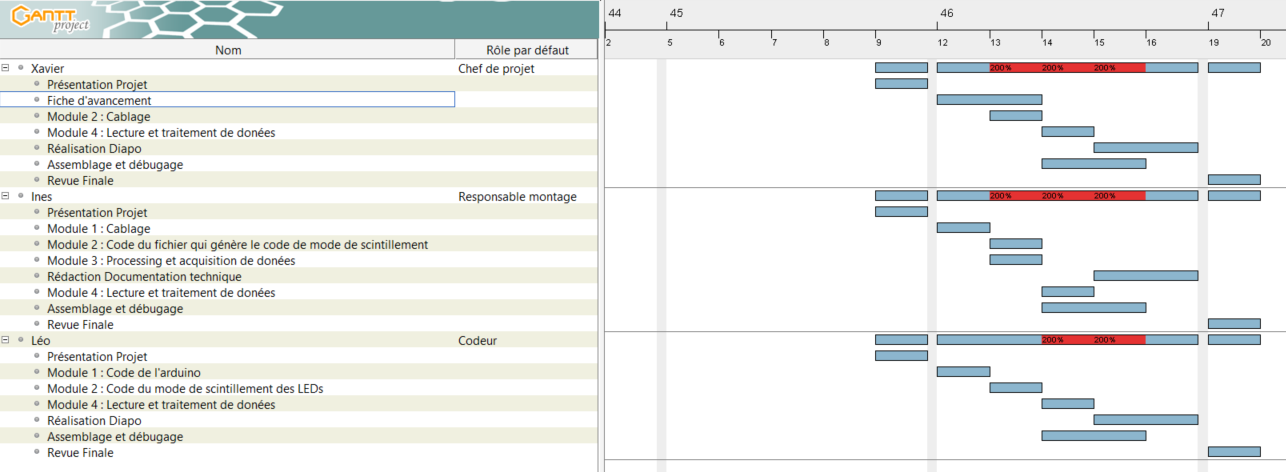
### Répartition des modules sur la semaine

Nous nous sommes mis comme objectif de réaliser un module par jour et de réserver le vendredi à la finalisation des programmes et de la rédaction du compte rendu technique.



### Répartition des tâches dans les modules

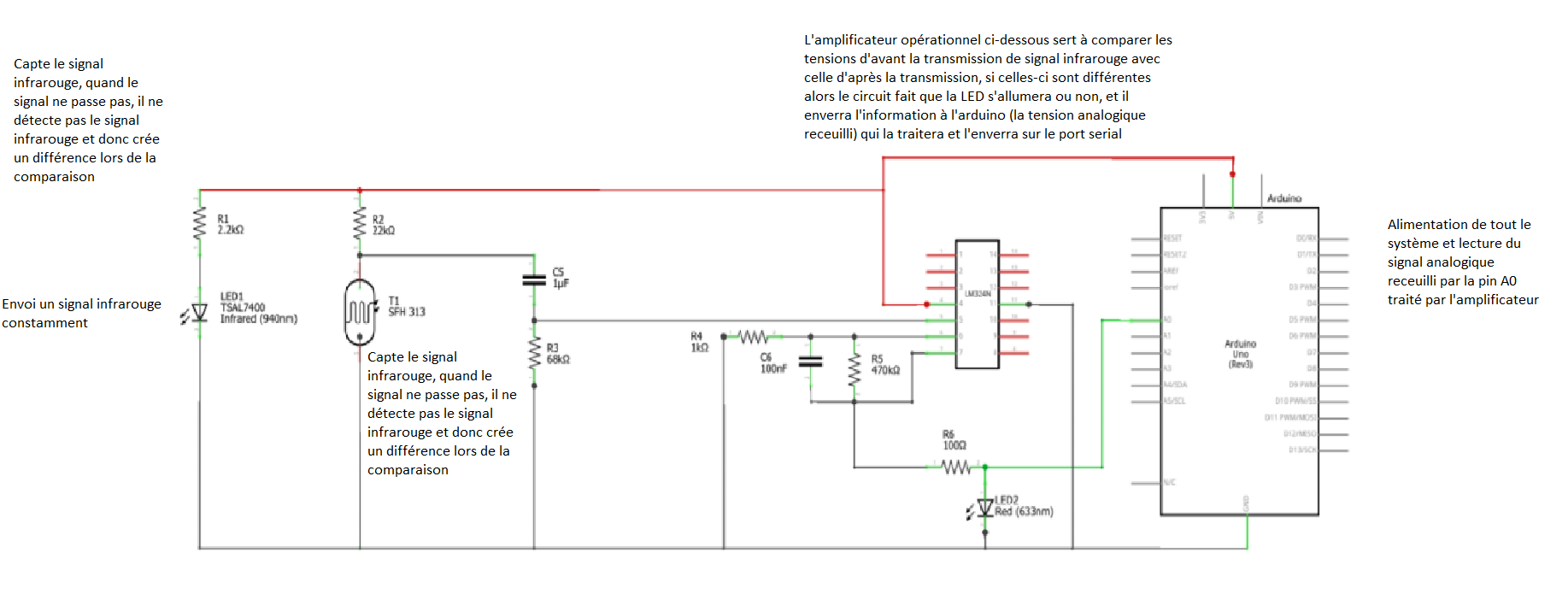
Chaque membre du groupe avait une tâche assignée dans chaque module. Dans l’ensemble du projet, nous avons voulu que tous les membres travaillent au moins une fois sur un domaine qu’ils ne maîtrisaient pas encore très bien. Le but était de renforcer nos connaissances en électronique et en programmation en langage C.



## Realisation des besoins

### Module 1

Pour ce module nous avons réalisé le montage ci-dessous :



Connexion de chaque composant :

Arduino -> A0 sur entrée LED rouge/ Une ligne de 5V et une ligne de GND

LED rouge -> entrée : A0/Sortie : résistance 100Ω

Résistance 100Ω -> sortie LED rouge et Port 3 du LM358

LM358 -> Port 3 ; Port 8 ; Port 7 ; Port 5

Condensateur 100nF -> 1er -> entrée : Port 3 LM/ sortie : résistance 1kΩ

2ème -> entrée : Condensateur 1µF/sortie : GND et Port 7 LM

Condensateur 1µF -> entrée : résistance 48kΩ/ sortie : Condensateur 100nF et Port 8

Résistance 68kΩ -> entrée : condensateur 1µF/ sortie : GND

Phototransistor -> entrée : Port 6/ sortie : Port 5 et LED infrarouge

LED infrarouge -> entrée : Phototransistor/ sortie : résistance 2.2kΩ

Résistance 2.2 kΩ -> 5V et LED infrarouge

Le but de ce module est de capter le pouls par réception de lumière infrarouge. Le système va percevoir les battements de cœurs de l’utilisateur (via le transmetteur infrarouge et son récepteur)

L’amplificateur va comparer la tension d’entrée dans le récepteur (soit celle d’après la photorésistance) et celle après la réception.

En fonction du résultat de celle-ci deux options vont en découler :

Si les tensions d’avant et après sont égales alors la LED ne s’allumera pas et cela ne correspondra pas à une pulsion cardiaque, l’Arduino recevra donc par la pin A0 une tension analogique qui ne correspondra pas à un rythme cardiaque et ne l’utilisera donc pas dans ses calculs ni dans le ficher csv que nous devons envoyer dans le port sérial.

Si les tensions d’avant et après ne sont pas égales alors la LED s’allumera et cela correspondra à une pulsion cardiaque, l’Arduino recevra donc par la pin A0 une tension analogique qui correspondra à un rythme cardiaque que nous devrons convertir en pouls

Ce code cardio.h sert à capter et à calculer le pouls détecté avec le système infrarouge avec la fonction void capteur\_et\_calcul(float a, long b, int c)

Le code source **cardio.c** qui exécute la fonction void capteur\_et\_calcul (float tension, long time, int max\_v

On retrouve la fonction void qui était sous forme de prototype dans le cardio.h

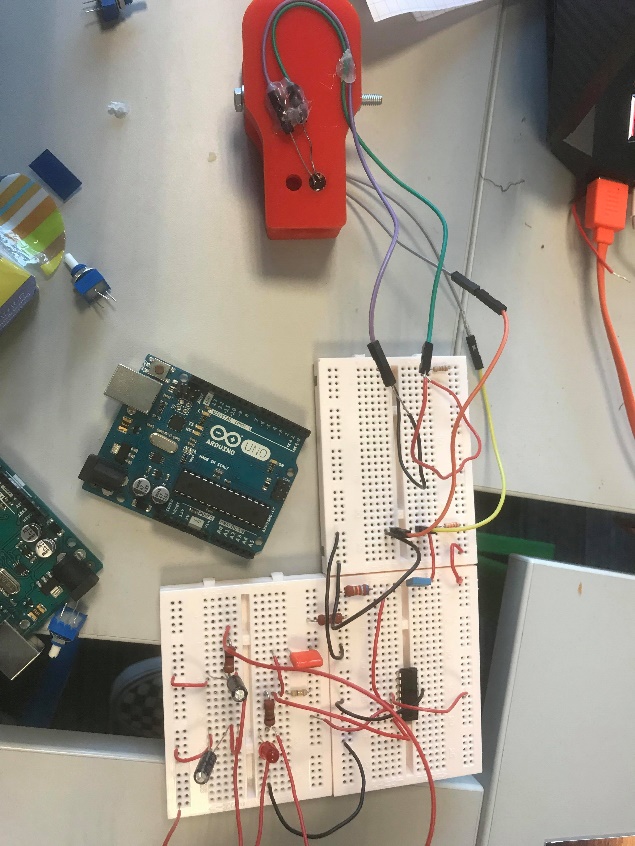
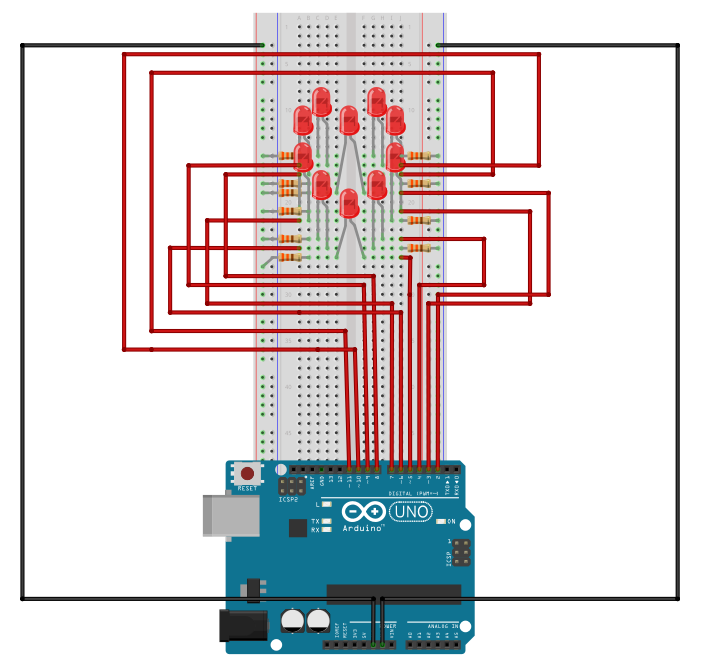


Photo du montage

### Module 2

Ce module est celui du cœur de LEDs. On doit créer un circuit avec des LEDs agencées en forme de cœur et le but est de les faire allumer suivant le pouls reçu, et suivant le mode choisi (d’allumage de LEDs).

Montage sur Fritzing du cœur de LED



Dans le code **heart.h** on définit les prototypes des fonctions des modes d’allumages des LEDs :

void allLeds(long w, int n, int i);

void chenilleArt(long w, int n);

void one\_on\_Two(long w, int j, int i);

void one\_on\_three(long w, int n, int i);

void whatever\_yall\_want(long w, int l, int i);

Dans le code heart.c on a donc 5 modes d’allumages :

- Toutes les LEDs sont allumées à chaque battement perçu : **void allLeds(long w, int n, int i)**

- Les LEDs sont allumées les unes après les autres en mode « chenille » : **void chenilleArt(long w, int n, int i)**

- Une LED sur deux est allumée : **void one\_on\_Two(long w, int j, int i)**

- Une LED sur trois est allumée : **void one\_on\_three(long w, int n, int i)**

- La LED choisie est allumée : **void whatever\_yall\_want(long w, int l, int i)**

Cette programmation est du code Arduino, après avoir choisi le mode voulu dans le menu (programmation C) on renvoi le mode dans l’Arduino avec le header param.h.

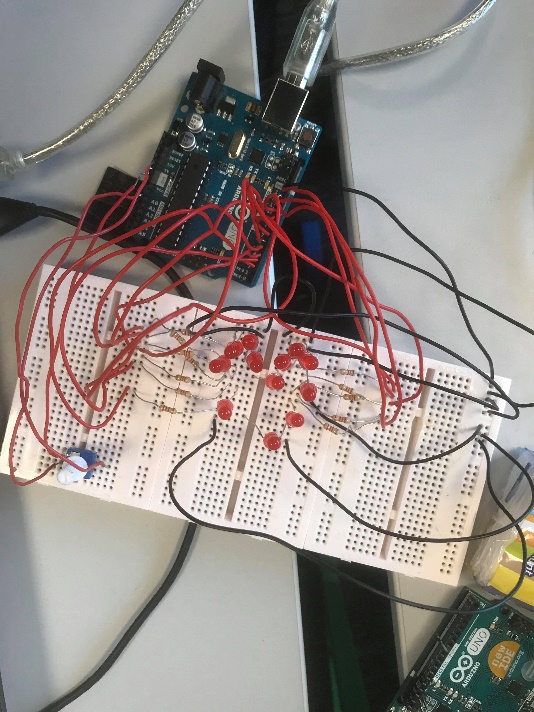
Le header param.h vient du mode choisi dans menu.c :

**Menu.h** définit le prototype de la fonction int choixMode(int a);et menu.c appelle cette dernière.

**Param.h**permet d’envoyer le mode choisi en programme C dans la carte Arduino.

Dans le **generatecode.c**, nous avons tout d'abord créer une fonction qui ouvre le fichier puis une fonction qui écrit dedans avec la fonction void setup (FILE\* file, int mode\_data) et void write (FILE\* a, int mode\_data)

Montage du cœur de LEDs



### Module 3

L’Arduino enverra ensuite dans un fichier .csv le temps à laquelle les mesures sont prises ainsi que le pouls par le port Série de l’Arduino

Un code en C sur un ordinateur qui réceptionnera le fichier en .csv s’occupera d’afficher les résultats.

Pour le module 3, nous avons dû tout d'abord installer l'IDE processing pour utiliser le langage processing. Nous avons donc recopié le code fournis par le document et l'avons modifé préalablement.

En outre, nous avons modifié cette ligne : **String SenVal = udSerial.readString(); en String SenVal = udSerial.readStringUntil(';');**

Ce programme sert à collecter toutes les données récupérées par l'Arduino et les implémenter dans un fichier au format .csv en écrivant sur chaque ligne du fichier ce que l’arduino écrit sur le port série jusqu’à qu’elle envoie un ‘ ;’ .

### Module 4

Dans le fichier **« action.c »** il y a les algorithmes de tri et de recherche. Nous avons choisi d’utiliser la méthode du tri à bulle car ce dernier était relativement efficace lors des tris. Le fichier **« action.h »** contient les prototypes adéquats pour action.c.

Le fichier **« data.c »** réalise les fonctions nécessaires pour la transformation d’un fichier CSV en tableau de structure. La première étape était la lecture du fichier, puis le comptage de ligne, ensuite l’utilisation d’un automate qui va sélectionner les éléments dont nous avons besoin. Cet automate range dans deux tableaux différents le temps enregistré et la pulsion.

Enfin, la fonction de transformation d'un fichier csv en tableau de structure.

Le fichier **"data.h"** définit des structures, des tableaux de structures et les prototypes des fonctions contenues dans **"data.c".**

Dans le fichier **"main.c"** est écrit les fonctions contenues dans menu.c et menu\_bis.c avec choixMode(int a), GlobalArray globalarray = NULL, et ShowingModeChoice(mode, globalarray)

Dans le fichier **"menu\_bis.c"** nous avons codé un menu pour le mode de tri et un switch pour les appels de fonctions.

Dans le fichier **"menu\_bis.h"** nous avons codé le prototype du menu bis.

En ce qui concerne la complexité des algorithmes de tri et de recherche, elle ne dépasse pas nln(n).

## Bilan Schématique

