HeXart Care : Reference

[I/ Liste des modules et description 2](#_Toc530083738)

[Module Arduino 2](#_Toc530083739)

[Module Processing 2](#_Toc530083740)

[Module C et interface 2](#_Toc530083741)

[II/ Module Arduino 2](#_Toc530083742)

[Bibliothèque Hearth 2](#_Toc530083743)

[void mod1(); 2](#_Toc530083744)

[void mod2(); 2](#_Toc530083745)

[void mod3(); 2](#_Toc530083746)

[void mod4(); 2](#_Toc530083747)

[Fichier Param.h 2](#_Toc530083748)

[III/ Module Processing 3](#_Toc530083749)

[IV/ Module C et interface 3](#_Toc530083750)

[Bibliothèque csvparser 3](#_Toc530083751)

[struct CSVFile 3](#_Toc530083752)

[struct Data 3](#_Toc530083753)

[CSVfile\* readCSV(char\* filename, CSVfile\* ret); 3](#_Toc530083754)

[void freeCSVFile(CSVfile\* f); 4](#_Toc530083755)

[Data \*accessdata (int n, Data\*chainlist); 4](#_Toc530083756)

[void freechain(Data\*chain); 4](#_Toc530083757)

[Bibliothèque generate 4](#_Toc530083758)

[void generateParam (int choice); 4](#_Toc530083759)

[Bibliothèque input 4](#_Toc530083760)

[int choicemode(); 4](#_Toc530083761)

[Bibliothèque stats 5](#_Toc530083762)

[void dispFileData(struct CSVfile\* d); 5](#_Toc530083763)

[struct CSVfile\* sortMax(struct CSVfile\* d, int comparatorMode); 5](#_Toc530083764)

[struct CSVfile\* sortMin(struct CSVfile\* d, int comparatorMode); 5](#_Toc530083765)

[CSVfile\* getOnTimePeriod(struct CSVfile\* f, int tmin, int tmax); 6](#_Toc530083766)

[int getLineNumber(struct CSVfile\* f); 6](#_Toc530083767)

[int chainMax(struct CSVfile\* f); 6](#_Toc530083768)

[int chainMin(struct CSVfile\* f); 6](#_Toc530083769)

[Bibliothèque cinterface 6](#_Toc530083770)

[int cinterface(); 6](#_Toc530083771)

[void loop(struct CSVfile\* f); 7](#_Toc530083772)

[void freeAll(struct CSVfile\* f); 7](#_Toc530083773)

# I/ Liste des modules et description

## Module Arduino

Le module Arduino comprend tout le code qui sera, une fois complié, téléversé sur la carte arduino afin dans un premier temps de récupérer les informations du capteur infrarouge, de les traiter, de les envoyer à l’ordinateur et enfin de faire fonctionner les LEDs symbolisant chaque battement de cœur.

## Module Processing

Le module Processing n’inclue qu’un seul fichier source dont le rôle est de récupérer les informations envoyées par la carte afin de les écrire dans un fichier .csv pour ensuite être traitées par l’ordinateur.

## Module C et interface

Le module C et interface est le programme qui sert d’interface entre l’utilisateur et la partie technique. Son rôle comprend la récupération du fichier .csv produit par le module Arduino, le tri et la mise en forme des données et enfin l’interface utilisateur pour configurer le mode de clignotement des LEDs servant de représentation des battements du cœur.

# II/ Module Arduino

Bibliothèque Hearth

Description générale: Cette bibliothèque sert à créer les fonctions pour les différents modes d'affichage du cœur de LEDs.

void mod1();

Description : Allumage de chaque LEDs une par une puis extinction.

void mod2();

Description : Clignotement de 1 LED sur 2 une par une.

void mod3();

Description : Allumage de 1 LED sur 2 simultanément, puis extinction successive.

void mod4();

Description : Allumage successif de chaque LED puis extinction successive.

Fichier Param.h

Description générale : Permet de configurer la façon dont les LEDS vont s'allumer. Il va être généré par la bibliothèque generate.h.

# III/ Module Processing

Ce module processing sera donc fait avec le logiciel "Processing ". L'objectif de ce module et de récupérer les valeurs du pouls lues sur le port série de la carte arduino et de les stocker dans un fichier .csv que le programme va générer.

# IV/ Module C et interface

Bibliothèque csvparser

Description générale : Cette bibliothèque sert à traiter les fichiers .csv produits par le code processing et à en stocker les informations sous une forme optimisée.

struct CSVFile

Description : Représentation en C d’un fichier CSV et de son contenu.

Champs :

* size : contient la quantité de données chargées pour ce fichier.
* data : pointeur vers le premier élément d’une liste chaînée contenant toutes les données chargées du fichier .csv.

struct Data

Description : Représentation en C d’une donnée chargée depuis un fichier .csv. Cette structure est conçue pour être implémentée en liste chaînée.

Champs :

* nxt : pointeur vers l’élément suivant lorsque la structure est utilisée en liste chaînée.
* pulsenb : nombre de pulsations captées lors de la période représentée par la donnée
* time : temps passé lors de l’acquisition de la donnée depuis l’exécution du programme de capture de données.

CSVfile\* readCSV(char\* filename, CSVfile\* ret);

Description : Lit un fichier CSV formaté correctement et stock son contenu dans une variable dynamique de type CSVfile ou rajoute son contenu à une variable existante. Remarque : cette fonction crée un pointeur dynamique qu’il faut nettoyer avant la fermeture du programme ce qui se fait grâce à la fonction freeCSVFile.

Paramètres :

* Filename : chemin d’accès vers le fichier .csv avec le nom du fichier et l’extension. Le chemin peut être relatif.
* Ret : pointeur vers une structure CSVFile existante y pour rajouter les données prélevées dans le fichier .csv.

Renvoie : la fonction renvoie un pointeur vers la variable crée ou modifiée.

void freeCSVFile(CSVfile\* f);

Description : libère la mémoire dynamiquement allouée par la fonction readCSV.

Paramètre :

* f : pointeur vers l’adresse de la variable dynamiquement allouée par readCSV.

Data \*accessdata (int n, Data\*chainlist);

Description : permet d’accéder à la donnée positionnée à la n-ième position dans la liste chaînée. Remarque : la case 0 est inopérante, la numérotation commence donc à 1.  
Remarque bis : cette fonction est à proscrire et n’est présente que à des fins de débogage du programme.

Paramètres :

* n : numéro de la position de la donnée à accéder.
* chainlist : pointeur vers la liste chaînée où est stockée la donnée.

Renvoie : la fonction renvoie un pointeur vers la donnée correspondante.

void freechain(Data\*chain);

Description : permet de libérer la mémoire allouée par une liste chaînée seule.

Remarque : cette fonction est utilisée en interne par le programme et il est déconseillé de l’utiliser.

Paramètre :

* chain : pointeur vers le premier élément de la chaîne à désallouer.

Bibliothèque generate

void generateParam (int choice);

Description : permet de générer un fichier param.h correct avec un choix pré-paramétré.

Paramètre :

* choice : choix pré-paramétré à écrire dans le fichier

Bibliothèque input

Description générale : cette bibliothèque a pour unique but de permettre une entrée simplifiée par l’utilisateur du mode de clignotement des LEDs. Cette bibliothèque va donc de paire avec la bibliothèque generate.

int choicemode();

Description : permet à l’utilisateur d’entrer le mode de clignotement des LEDs souhaité sans générer le fichier param.h.

Remarque : cette fonction n’est pas utilisée actuellement et sera supprimé dans une prochaine version du logiciel.

Renvoie : la valeur du choix de l’utilisateur est renvoyée.

Bibliothèque stats

Description générale : cette bibliothèque sert à effectuer des opérations sur les listes chaînées afin de les trier, de rechercher à l’intérieur etc…

Remarque générale : toutes les fonctions de la bibliothèque dont le nom commence par un underscore ne doivent en aucun cas être utilisées. Elles sont utilisées de manière récursive par la bibliothèque mais ne sont certainement pas conçus pour être appelées à l’extérieur de la bibliothèque.

void dispFileData(struct CSVfile\* d);

Description : cette fonction affiche toutes les données contenues dans une variable de type CSVfile sans effectuer de modifications que ce soit aux données ou à l’ordre dans lequel elles sont rangées.

Paramètre :

* d : pointeur vers une variable de type CSVfile dont les données seront affichées à l’écran.

struct CSVfile\* sortMax(struct CSVfile\* d, int comparatorMode);

Description : cette fonction sert à trier les données d’une variable de type CSVFile dans l’ordre croissant en fonction de la valeur de la donnée ou du temps en fonction de la valeur de comparatorMode.

Paramètres :

* d : pointeur vers une variable de type CSVfile.
* comparatorMode : 1 pour une comparaison en fonction des valeurs des données et 0 pour une comparaison en fonction du temps.

Renvoie : la fonction renvoie une copie de d rangée selon les paramètres.

struct CSVfile\* sortMin(struct CSVfile\* d, int comparatorMode);

Description : cette fonction sert à trier les données d’une variable de type CSVFile dans l’ordre décroissant en fonction de la valeur de la donnée ou du temps en fonction de la valeur de comparatorMode.

Paramètres :

* d : pointeur vers une variable de type CSVfile.
* comparatorMode : 1 pour une comparaison en fonction des valeurs des données et 0 pour une comparaison en fonction du temps.

Renvoie : la fonction renvoie une copie de d rangée selon les paramètres.

CSVfile\* getOnTimePeriod(struct CSVfile\* f, int tmin, int tmax);

Description : sert à obtenir une copie des données de f sur une période de temps définie grâce aux paramètres.

Paramètres :

* f : la variable où sont prises les données.
* tmin : le temps minimum pris en compte.
* tmax : le temps maximal pris en compte.

Renvoie : la fonction renvoie une copie de f où seules les données relevées entre tmin et tmax sont présentes.

int getLineNumber(struct CSVfile\* f);

Description : permet d’obtenir le nombre de lignes conservées en mémoire.

Paramètre :

* f : la variable contenant les données.

Renvoie : le nombre de lignes présentes dans la mémoire.

int chainMax(struct CSVfile\* f);

Description : renvoie la valeur maximale trouvée dans les données du fichier.

Paramètre :

* f : la variable contenant les données.

Renvoie : la donnée la plus importante trouvée.

int chainMin(struct CSVfile\* f);

Description : renvoie la valeur minimale trouvée dans les données du fichier.

Paramètre :

* f : la variable contenant les données.

Renvoie : la donnée la moins importante trouvée.

Bibliothèque cinterface

Description générale : cette bibliothèque sert à appeler toutes les fonctions et à les exécuter dans le bon ordre. Elle a aussi le rôle de gérer l’interface utlisateur-machine.

int cinterface();

Description : première fonction du programme, son rôle est de préparer les variables pour la récupération des données dans le fichier.

Renvoie : le code de retour du programme : 0 si l’exécution s’est correctement déroulée et 1 si un problème a été détecté.

void loop(struct CSVfile\* f);

Description : fonction principale du programme, son rôle est de récolter les données du fichier CSV et d’appeler le menu lorsque cette action est requise.

Paramètre :

* f : fichier de données initialisé dans la fonction cinterface.

void freeAll(struct CSVfile\* f);

Description : fonction permettant de correctement libérer la mémoire du programme et ainsi d’éviter les fuites de mémoire.

Paramètre :

* f : fichier de données initialisé dans la fonction cinterface et utilisé dans la fonction loop.