Projet Allegro ECEcooked 2024
Livrable de mi-parcours du 07/04/2024

Sommaire

Page de garde

Dans la page de garde de votre livrable de mi-parcours, mettez l'intitulé du projet, votre groupe de TD, numéro d'équipe comme indiqué dans le drive Equipes par niveaux du projet Allegro S2 2023-2024 - Google Sheets, les noms et prénoms des coéquipier, et si possible une image sympa du projet (sans forcément faire un screenshot de celle du sujet du projet d'origine).

Sommaire

Le sommaire de votre livrable doit être une table des matières, telle que décrite dans le modèle du sommaire de ce tutoriel, avec des liens pour chaque chapitre de votre livrable, sans oublier de paginer votre livrable.

Synthèse du sujet du projet

En quelques lignes sans blabla, faites une synthèse du sujet du projet, sans copier-coller du sujet. Vous pouvez y introduire une image mais toujours sans copier-coller l'une des images du sujet (par exemple une image de votre thème choisi).

Structures de données : schémas mémoire

Comme écrit dans le sujet : « Cette année, vous disposez de beaucoup plus de temps que d'habitude pour pousser au maximum votre conception, c'est très important.

Objectif : Tout anticiper, améliorer, afin de tout prendre en compte et tout rendre simple à paramétrer, à coder, et à modifier etc. »

Tout d'abord, identifiez les structures de données nécessaires comme les suivantes :

- Listes Chaînées (Files, Piles) : Vous avez probablement identifié des fonctionnalités nécessitant l'utilisation de listes chaînées. C'est le cas par exemple des commandes : la cuisine doit traiter la plus ancienne en priorité afin que les clients ne s'impatientent pas !

- Allocation dynamique : Comme pour un vrai restaurant, il est difficile d'anticiper le nombre de tomates nécessaires pour la journée ! Heureusement, nos tomates ne prennent que quelques octets en mémoire, pas besoin de gros frigo. Mais, ne gaspillons pas la mémoire ! Nous créerons les tomates à la demande, via des allocations dynamiques. Mais une question persiste... où les stocker ? »

⚡ **Faites des schémas mémoire de vos structures de données mais sans code, y compris d'allocation dynamique nécessaire avec pointeurs, en expliquant dans quelles fonctionnalités en-dessous (exigences fonctionnelles) elles peuvent servir. Soyez le plus clair possible dans vos illustrations de schémas mémoire de vos structures de données, en indiquant concrètement leur utilisation**.

Exigences fonctionnelles prioritaires

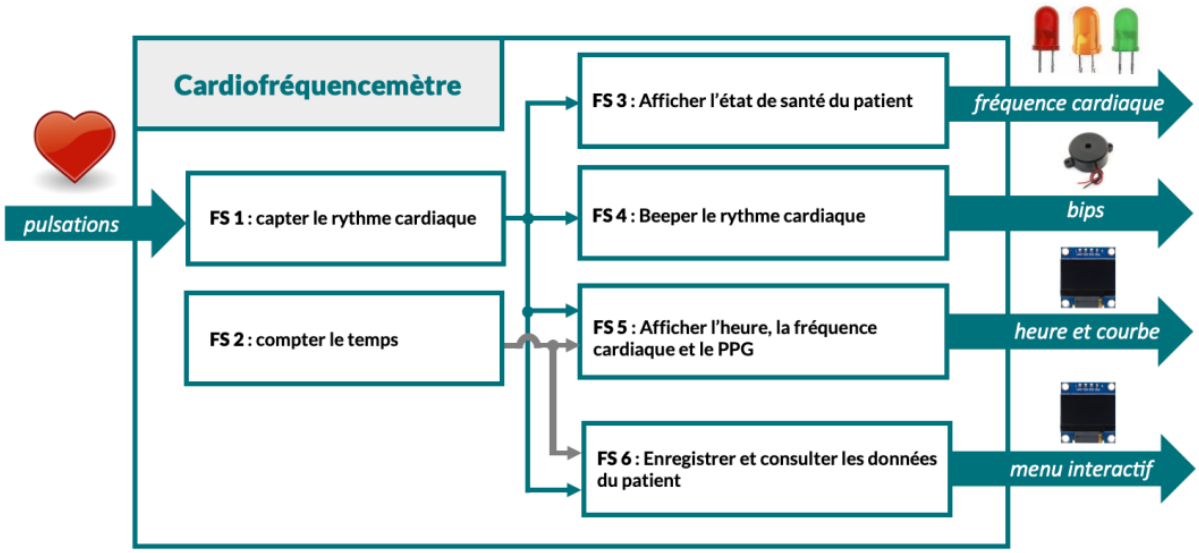
En reprenant le sujet du projet, citez au moins 10 exigences fonctionnelles qui vous semblent prioritaires, en vous inspirant du sujet.

Exemples à compléter avec le code (EF) et l'objectif pour chaque exigence fonctionnelle :

- **EF1** : Deux cuisinier.e.s travaillent ensemble pour cuisiner les commandes le plus rapidement possible dans un environnement qui rend la tâche parfois difficile !
- **EF2** : Votre jeu devra toujours se jouer à deux joueurs.
- **EF3** : Chaque joueur choisira son pseudo, qui sera affiché à différents moments du jeu, notamment lors de l'affichage des scores par joueur (Facile : saisie des pseudos dans le terminal, Difficile : saisie des pseudos dans l'interface graphique du jeu).

Diagramme fonctionnel

En vous inspirant du diagramme fonctionnel ci-dessous d'un projet d'électronique (même si rien à voir avec votre sujet ECEcooked), décrivez schématiquement votre diagramme fonctionnel de votre projet ECEcooked en vous servant des exigences fonctionnelles précédentes (un code et un titre par fonctionnalité) et les interactions entre elles, sans oublier si besoin les entrées et les sorties nécessaires :



Exigences techniques de chaque fonctionnalité

- Pour chaque fonctionnalité décrite ci-dessus, indiquez les exigences techniques nécessaires, en vous inspirant de celles mentionnées éventuellement dans le sujet et d'autres qui vous semblent nécessaires.

- Résumer les exigences techniques dans un tableau et établir un plan de test.

- Pour modèle d'exigences techniques, ci-dessous des exemples d'exigences techniques liés au diagramme fonctionnel ci-dessous d'un projet d'électronique, à adapter pour votre projet d'informatique, en précisant si besoin des conditions particulières :

■ La fonction secondaire **FS1** "capter le rythme cardiaque" se charge de capter le rythme cardiaque et de transmettre sa valeur à une unité de calcul.

Exigences techniques

ET1.1 Le logiciel du système ne doit pas utiliser une bibliothèque préconçue (exemple : **interdiction d'utiliser la bibliothèque Adafruit**). Le développement du programme doit être inédit et non commun à un autre groupe.

ET1.2 Le système doit comprendre uniquement un capteur iDuino SE049, qui a été choisi pour son peu de documentation. Il est interdit de changer de capteur.

ET1.3 Le système doit être robuste au changement de luminosité.

ET1.4 Le système doit fonctionner correctement sans limite de temps (pas de dérive de la fréquence mesurée).

ET1.5 La fréquence cardiaque doit être exprimée en battements par minute (bpm).

Exigence technique de la modularité du code

En dehors de exigences techniques des fonctionnalités, vous devez prouver que vous respectez la modularité du code, en décomposant votre programme en plusieurs fichiers de la manière suivante :

- Au moins un header .h contenant éventuellement une/des constante(s), une/des structure(s) et obligatoirement les prototypes des sous-programmes tous commentés comme il se doit. Idéalement, il faudrait un header par thème (Par exemple, un pour le cuisinier, un autre pour l'ingrédient de cuisine etc.) pour une meilleure modularité.
- Au moins un fichier source .c contenant l'implémentation des sous-programmes. Idem que les headers pour un fichier source par thème, pour meilleure modularité et donc une meilleure répartition des tâches entre coéquipiers.
- Un dernier fichier source ne contenant que le ***main*** dont le rôle est de lancer le jeu et appeler les différents sous-programmes.

⚡ Interdiction d'utiliser des variables globales anti-modularité, sous peine d'avoir des malus.

⚡ **Pour prouver cela en partie, je vous prie de faire un screenshot de la structure de votre projet sur CLion, toujours sans montrer de code, mais en expliquant le rôle de chaque fichier en une phrase**.

⚡ **Eviter les duplications de code et de données, etc.**

Illustrer certaines fonctionnalités développées

- Illustrer certaines fonctionnalités que vous avez développé avec Allegro, avec des screenshot pour preuves, mais sans code, en citant chacune des fonctionnalités développées et bien testées.

- Pour chacune de ces fonctionnalités développées (même code et titre que dans le diagramme fonctionnel au-dessus), faire un screehshot et **indiquer les résultats obtenus en montrant tous les cas de tests de validation effectués**.

Organigrammes ou Algorithmes de fonctionnalités

Ecrivez les organigrammes ou algorithmes de fonctionnalités déjà développées sur Allegro. Pour chaque organigramme ou algorithme (pas de code) de fonctionnalité, indiquez le titre (le même que dans le diagramme fonctionnel), en étant le plus précis possible et sans oublier les tests de validation.

Répartition des tâches par fonctionnalités

En fonction des fonctionnalités énoncées au-dessus, faire un tableau récapitulatif de la répartition des tâches des coéquipiers pour chacune de ces fonctionnalités.

GitHub

Indiquez le nom et le lien de votre Github Classroom, ou autre GitHub, avec des screenshot montrant vos contributions individuelles effectives à GitHub, ainsi que vos branches. Pour chaque commit, soyez clair dans le titre de chaque fonctionnalité postée sur GitHub. Attention : la contribution de chacun est essentielle dans votre Github, pour preuve d'une répartition des tâches dans le développement des fonctionnalités du code. En cas d'écart important de contribution, nous en tiendrons compte, en différenciant les notes entre coéquipiers !

Etudes documentaires (sources)

Etablissez une étude documentaire, en citant vos sources de manière précise comme il suit :

- Pour chaque site web consulté, y compris sur BoostCamp, indiquez bien le thème concerné et le lien ainsi que son/ses auteurs(s) si cité(s).
- En cas de contact(s) autre(s) que des sites web (exemples : personnes, livres etc.), les citer de manière explicite en expliquant pourquoi (thèmes abordés, le genre d'aide).