Rapport de DEEP

Digital Embedded Electronics Project

**Livraison finale**

|  |  |
| --- | --- |
| **Nom du projet :** | **Vincent Colin** |
| **Etudiants :** | **AUGEREAU Evan**  **CAILLAUD Armand** |

Tout ce qui est surligné en jaune est à modifier/adapter/supprimer. Vous pouvez ajouter à ce rapport toute partie que vous jugeriez intéressante.

Ce rapport doit être exporté en PDF, inséré à l’archive zip livrée, avec ce nom (non respect de ces consignes : -1 point) :

**[DEEP]\_NOM1\_Prenom1\_NOM2\_Prénom2.pdf**

Ce rapport doit être synthétique et s’adresse à un public averti quant aux termes techniques utilisés.

* Problèmes logiciels
* 90 -> 180
* Pixy -> block struct pas constant / led OK / Problemes C
* Algo trouvé ?

# Compléments choisis

Ce chapitre 1 doit rester sur la première page de ce rapport.

6 points de la note de projet DEEP sont associés à ces compléments au choix.   
Pour chaque complément, l’obtention de tout ou partie des points indiqués dans la colonne ‘pts’ est conditionnée :

* au bon remplissage de l’annexe correspondante dans la suite de ce rapport
* aux critères d’évaluations détaillés dans ces annexes

Pour maximiser vos chances d’obtenir ces 6 points, il est conseillé de viser 8 ou 9 points de compléments ! NB : (en police Wingdings, on obtient : en tapant x :  , en tapant o : )

|  |  |  |
| --- | --- | --- |
| **Complément (voir l’annexe correspondante à la fin du rapport)** | **pts** |  |
| A- utilisation d'un **a**nalyseur logique pour déchiffrer des trames | 2 |  |
| B- design de PC**B** |  |  |
| avec bluepill | 3 |  |
| Composants CMS et microcontrôleur nu | 4 |  |
| C- design **C**AO d'un boîtier | 2 |  |
| D- **d**ocumentation doxygen du code source | 1 |  |
| E- mesure de conso et d’**é**nergie selon scénarios | 2 |  |
| F- enregistrement de paramètres en **f**lash | 1 |  |
| G- **g**estion de version du code source / **G**it ou SVN | 1 |  |
| T- jeu de **t**ests pour valider une fonctionnalité software ou hardware | 1 | o |
| Nombre de points ciblés au maximum : | 7 / 6 | |

Commencez par une brève introduction.

# Cahier des charges

Rappelez ici votre cahier des charges (vous pouvez réutiliser le paragraphe du livrable intermédiaire, et le mettre à jour si besoin.)

Il faut décrire très précisément les limites du projet, et les contraintes imposées au développement.

Vous pouvez imaginer ici un contexte « client », et évoquer l’expression d’un besoin auquel vous répondez avec votre projet.

# Manuel d'utilisation

Vous devez présenter ce manuel du point de vu utilisateur final (les outils de développement n’ont pas leur place ici !).

Vous devez décrire ici les modalités d'utilisation de votre réalisation de manière à ce qu'il puisse être mis en œuvre et essayé rapidement.

Pour un jeu d'échec par exemple, il faudra expliquer comment le joueur doit saisir les coordonnées de déplacement. Ou encore, si l'affichage n'est pas explicite, il faudra l'expliquer ici.

Au besoin, vous pouvez vous appuyer sur une représentation 3D ou une photo.

Précisez quelles sont les contraintes d’alimentation à respecter pour utiliser votre réalisation.

Votre manuel d’utilisation est complet si l’utilisateur qui ne connaissait pas votre produit n’a aucun doute sur ce qu’il peut faire et comment il peut le faire. N’hésitez pas à faire relire cette partie par un autre groupe !

# Mes ports

Mettez à jour le tableau de l’affectation des ports du microcontrôleur dans votre application (ou recopiez le si rien n’a changé depuis le rapport intermédiaire) :

|  |  |
| --- | --- |
| **Pin** | **rôle** |
| **PA0** |  |
| **PA1** |  |
| **PA2** | UART2 Tx |
| **PA3** | UART2 Rx |
| **PA4** |  |
| **PA5** |  |
| **PA6** |  |
| **PA7** |  |
| **PA8** |  |
| **PA9** |  |
| **PA10** |  |
| **PA11** |  |
| **PA12** |  |
| **PA13** | (non dispo - SWDIO) |
| **PA14** | (non dispo - SWDCLK) |
| **PA15** | non dispo sur certaines bluepill |
| **PB0** |  |
| **PB1** |  |
| **PB2** | (non dispo) |
| **PB3** |  |
| **PB4** |  |
| **PB5** |  |
| **PB6** |  |
| **PB7** |  |
| **PB8** |  |
| **PB9** |  |
| **PB10** |  |
| **PB11** |  |
| **PB12** |  |
| **PB13** |  |
| **PB14** |  |
| **PB15** |  |
| **PC13** | (LED bluepill) |
| **PC14** | (Quartz 32kHz bluepill) |
| **PC15** | (Quartz 32kHz bluepill) |

# Description d'un algorithme du programme

Vous devez représenter ici au moins l’une des tâches de votre programme, en respectant le formalisme de représentation d’un algorithme. Elle peut se faire sous forme schématique (algorigramme, diagramme d'activité UML, …) ou sous forme textuelle (pseudo code, langage C, …). Une machine à états est la bienvenue.

# Structure du programme

Cette partie doit lister et décrire l'ensemble des fichiers du projet que vous avez produits, ainsi que leurs principales fonctions.

Un exemple est donné ci-après.

## Fichier « uart.c »

Ce module permet de manipuler l'UART pour envoyer et recevoir des caractères sur la liaison série RS232 de la carte.

|  |  |  |
| --- | --- | --- |
| **Fonction** | **Nom du développeur** | **Description** |
| UART\_testerReception | M. Dupond | Permet de savoir si un caractère a été reçu |
| UART\_recevoirOctet | M. Dupond | Permet d'obtenir un caractère (bloquant jusqu'à réception de caractère) |
| UART\_envoyerOctet | Mme. Dupont | Permet d'envoyer un caractère |

# Tests

Dans cette partie vous décrivez les différents tests que vous avez effectués avec le résultat du test.

Ce rapport de test permet de savoir à quel stade du développement se trouve votre réalisation.

|  |  |  |
| --- | --- | --- |
| **Intitulé du test** | **Description de ce qu’il faut faire pour jouer le test et de ce qu’on doit observer** | **Observation obtenue et conclusion** |
| Test d’alimentation du microcontrôleur : | Mesure au voltmètre ; les entrées d’alimentation doivent être à 5V | OK |
| Test de la liaison Tx -> PC | Test de la liaison électrique entre la broche utilisée pour une communication série en sortie de l’UART (TX) et la broche de réception de l’ordinateur | OK |
| Test d’envoi unidirectionnel entre le microcontrôleur et l’ordinateur d’un caractère | un caractère doit être envoyé sur la broche d’émission du microcontrôleur, il doit être reçu sur le PC et doit apparaître sur un terminal. | NON  Caractère non reçu |
| Test de la led « validation » | Lorsque l’utilisateur presse le bouton « validation », la led correspondante doit s’allumer pendant 500ms. Cette vérification peut être réalisée à l’oscilloscope. | KO  La led s’allume seulement pendant 400ms. |

# Cahier de suivi

Pour chaque date (chaque séance de travail, pendant ou en dehors des créneaux prévus à l’agenda), notez dans ce cahier de suivi les tâches réalisées, les réalisateurs, les difficultés rencontrées, l’état d’avancement de la réalisation…

L’objectif de cette démarche est de vous inciter à prendre du recul sur le déroulement de votre projet. Cette habitude est essentielle pour l’ingénieur qui doit savoir quantifier son temps de travail et apprendre à anticiper les durées des tâches.

Reprenez et complétez le cahier de suivi du livrable intermédiaire.

|  |  |  |
| --- | --- | --- |
| Date | Tâches, réalisateurs, difficultés rencontrés. | A faire la prochaine fois |
| 03/01 | Lucky Luke : routage de la carte Saloon 2.0  Jolly Jumper : développement du module logiciel Abreuvoir.c/h.  Difficulté rencontrée : le bug dans le fichier Dalton.c nous a fait perdre du temps… Nous l’avons résolu en lisant la documentation (plus vite que notre ombre). | Lucky Luke : DRC, validation du routage, envoi en fabrication.  Jolly Jumper : Test du module logiciel Abreuvoir.c/h |
|  |  |  |

# État d'avancement et analyse du projet réalisé

Dans cette partie, vous rendez compte de l'avancement de votre projet.

Le cas échéant, nous vous demandons de lister les points à terminer (telle partie à écrire, telle fonctionnalité à ajouter, tel sous-programme à tester, …).

Vous devez ANALYSER ce que vous avez produit :

Exemples :

* Si vous deviez refaire le projet, que feriez-vous autrement ?
* Quel moyen de débogage aurait-il fallu utiliser plus tôt ou différemment ?
* Quelles limites gênantes et pour quels composants ?

# Conclusion

Cette partie est disponible pour une brève conclusion.

Cette partie peut aussi servir à mettre en valeur un apprentissage, par exemple une difficulté que vous avez surmontée.

**Les annexes qui suivent dépendent des compléments que vous avez choisis.**

**Supprimez les annexes qui ne vous concernent pas, et complétez celles qui vous concernent.**

Annexe B – Complément « design de PCB »

Cette annexe facultative sera à renseigner si vous avez choisi ce complément.

Après avoir suivi les étapes du document **« Complément – Design de PCB »**, complétez cette annexe.

Auto-validez votre design en vérifiant les points suivants :

|  |  |
| --- | --- |
| Nous penserons à joindre à **l’archive zip livrable** : notre fichier SchDoc, notre fichier PcbDoc ainsi que le PDF généré par Altium contenant schéma, vue des couches de routage et vue 3D. |  |

|  |  |
| --- | --- |
| Les connecteurs sont accessibles |  |
| Un plan de masse est présent sur chaque couche de cuivre |  |
| Les GND sont correctement tous reliés par des pistes (même si le plan de masse les regroupe ensuite). |  |
| Le DRC (Design Rules Check) passe sans aucune erreur |  |
| Présence d'un connecteur UART (Rx + Tx + GND), parce que c'est toujours utile |  |
| Si le microcontrôleur est placé en CMS : présence d’un connecteur de programmation (SWDIO + SWDCLK + GND) |  |

Si mon PCB a été conçu pour être réalisé à l’ESEO, il respecte les contraintes du processus de fabrication (plus d’informations dans le document « Validation PCB avant fabrication.pdf » :

|  |  |
| --- | --- |
| Pas d’arrivée en TOP non soudables sur des composants traversants |  |
| Noms des étudiants et site (Angers/Vélizy/Dijon) visibles sur le TOP. |  |
| Carte aux dimensions raisonnables (pas trop d’espace libre !) |  |
| Pistes >= 12 mils ; et au moins 25 mils lorsque c’est possible |  |
| Clearance >= 20mils lorsque c’est possible. (ponctuellement >= 10mils) Clearance des plans de masse réglée à 40 mils. |  |

Au-delà les critères indiqués ci-dessus, les critères permettant de juger votre réalisation sont les suivants. Autoévaluez en une note de 0 à 100 votre design :

|  |  |
| --- | --- |
| Un schéma correct | \_\_\_/100 |
| Qualité du placement (un bon placement garanti un bon routage) | \_\_\_/100 |
| Qualité du routage | \_\_\_/100 |

* Présenter ici des extraits du schéma (capture d’écran) en expliquant le rôle des composants / périphériques.
* Ajoutez une vue 2D du routage (plans de masse masqués)
* Ajoutez une vue 3D du PCB (ou deux si la vue Bottom apporte des informations complémentaires)
* Évoquez vos éventuelles difficultés rencontrées.

Après avoir généré les fichiers Gerber de votre design, estimez le coût de production de ce PCB chez un fabricant en y déposant l’archive zip contenant ces fichiers (chez Eurocircuit, JLCPCB, ou tout autre constructeur)… vous pouvez ajouter ici une capture d’écran de cette estimation.

Annexe C – Complément « Design CAO d'un boîtier »

Cette annexe facultative sera à renseigner si vous avez choisi ce complément.

En marge du développement hardware et software de l'électronique, il est souvent nécessaire d'intégrer cette électronique dans un environnement mécanique.

Savoir réaliser un boîtier prototype est une compétence complémentaire intéressante.

Vous pouvez suivre ce tutoriel à votre rythme pour découvrir le logiciel Solidworks, ou utiliser un autre logiciel de votre choix :

<https://campus.eseo.fr/pluginfile.php/150246/mod_resource/content/1/607698-apprenez-a-utiliser-solidworks.pdf>

Les critères d'évaluations de ce complément sont :

- le soin apporté dans le design 3D

- la complexité

- l'adéquation des choix avec l'ergonomie, les contraintes de montage, ...

Attention, la **réalisation concrète** de l'objet conçu **n'entre pas** en compte dans l'évaluation.

Toutefois, pour certains projets, une réalisation de l'objet sera indispensable au bon fonctionnement de votre projet : attention, l'atelier mécatronique réserve sa capacité de fabrication à des projets où les pièces concernées sont de **taille raisonnable** et amènent un **élément fonctionnel essentiel.**

(en d'autres termes, on ne fabrique pas de boîtier "juste pour faire joli").

Voici quelques éléments à fournir pour présenter votre travail ; pour chaque pièce :

* Nous avons utilisé le logiciel \_\_\_\_\_\_\_ pour réaliser \_\_\_\_\_\_\_.
* Fournir les vues 2D ou 3D des pièces réalisées.
* Décrire les matériaux ciblés / mode de production ciblé (découpe laser, impression 3D, usinage, découpe manuelle, injection plastique, emboutissage…)
* Mise en contexte / Explication du rôle de ces pièces / de la raison d’être de certains trous ou particularités mécaniques.
* (Eventuellement) : Mises en plan de ces pièces.
* Réponse à la question : « Cette pièce a-t-elle été réalisée ? »
  + Si oui, vous pouvez joindre une photo
  + Si oui, y’a-t-il eu des difficultés pour les réaliser ou pour les mettre en œuvre ?

Annexe D – Complément « Documentation Doxygen du code source »

Cette annexe facultative sera à renseigner si vous avez choisi ce complément.

Dans vos codes sources, un minimum de commentaires est attendu.

Toutefois, nous vous invitons avec ce complément à systématiser les commentaires en suivant le format Doxygen.

Après avoir consulté ce tutoriel en ligne (ou un autre !!!) : <https://franckh.developpez.com/tutoriels/outils/doxygen/>

* Documentez votre code au format doxygen

Indiquez ici la liste des fichiers où vous avez documenté le code au format doxygen :

|  |
| --- |
|  |

* Compressez dans une archive nommée « Site\_doxygen.zip » le site généré par doxywizard.
* Insérez cette archive dans votre archive livrable (NOM1\_prenom1\_NOM2\_prenom2.zip).

Complétez cette annexe par un éventuel retour que vous auriez à faire suite à ce travail. (avis / rapport d’étonnement / …)

Annexe G – Complément « Gestion de version du code source »

Travailler à plusieurs sur un code source rend souvent pertinent le recours à un gestionnaire de version. Cet outil est également nécessaire pour développer plusieurs branches, pour suivre les versions livrées aux clients, ou pour factoriser certains développements liés à plusieurs projets.

Pour toutes ces raisons, il est primordial d’entraîner nos étudiants à la pratique des outils de gestion de version.

Dans le cadre du DEEP, nous n’entrerons pas dans les détails, mais nous vous encourageons à pratiquer par vous-même cette compétence clé, très appréciée chez les stagiaires (tenez-vous le pour dit !)

Consultez le document « Complément G – Gestion de version du code ».

Si besoin, vous pouvez consulter ce tutoriel :   
<https://openclassrooms.com/fr/courses/7162856-gerez-du-code-avec-git-et-github>

Si vous avez choisi ce complément, vous devez fournir dans cette annexe :

* Capture d’écran donnant une vue sur les révisions (avec les dates et auteurs visibles des commits).
* Retour d’expérience (sous la forme d’un ou plusieurs paragraphes) : que vous a apporté cette expérience. Aviez-vous déjà pratiqué la gestion de version par ailleurs…

L’évaluation de ce complément prendra en considération la quantité et l’étalement temporel des commits (il ne s’agit pas simplement d’y publier le code source final la veille de la soutenance).