ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

UNIVERSITÉ DU QUÉBEC

rapport technique

PRÉSENTÉ À L’ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

DANS LE CADRE DU PROJET DE FIN D’ÉTUDES

Restructuration du système de télécommandes pour les robots du club étudiant Capra

par

gervais, François

ST-onge, Rémi

présenté à

Jean-françois boland

montréal, le mercredi 9 décembre 2009

**<titre du rapport TECHNIQUE>**

<NOM, Prénom>

**RÉSUMÉ**

<Texte interligne simple>

**TABLE DES MATIÈRES**

Page

[INTRODUCTION 1](#_Toc185065390)

[CHAPITRE 1 <titre> 2](#_Toc185065391)

[1.1 <Titre> 2](#_Toc185065392)

[CHAPITRE 2 <titre> 3](#_Toc185065393)

[CHAPITRE 3 <Titre> 4](#_Toc185065394)

[3.1 <Titre> 4](#_Toc185065395)

[CHAPITRE 4 <titre> 5](#_Toc185065396)

[4.1 <Titre> 5](#_Toc185065397)

[CHAPITRE 5 <titre> 6](#_Toc185065398)

[5.1 <Titre> 6](#_Toc185065399)

[CHAPITRE 6 <titre> 7](#_Toc185065400)

[6.1 <Titre> 7](#_Toc185065401)

[CHAPITRE 7 <titre> 8](#_Toc185065402)

[7.1 <Titre> 8](#_Toc185065403)

[CHAPITRE 8 <titre> 9](#_Toc185065404)

[8.1 <Titre> 9](#_Toc185065405)

[CHAPITRE 9 <titre> 10](#_Toc185065406)

[9.1 <Titre> 10](#_Toc185065407)

[CHAPITRE 10 <titre> 11](#_Toc185065408)

[10.1 <Titre> 11](#_Toc185065409)

[CONCLUSION 12](#_Toc185065410)

[RECOMMANDATIONS <S’il y a lieu> 13](#_Toc185065411)

[ANNEXE I <titre> 14](#_Toc185065412)

[ANNEXE II <titre> 15](#_Toc185065413)

[ANNEXE III <titre> 16](#_Toc185065414)

[ANNEXE IV <titre> 17](#_Toc185065415)

[ANNEXE V <titre> 18](#_Toc185065416)

[ANNEXE VI <titre> 19](#_Toc185065417)

[ANNEXE VII <titre> 20](#_Toc185065418)

[ANNEXE VIII <titre> 21](#_Toc185065419)

[ANNEXE IX <titre> 22](#_Toc185065420)

[ANNEXE X <titre> 23](#_Toc185065421)

[APPENDICES <S’il y a lieu> 24](#_Toc185065422)

[LISTE DE RÉFÉRENCES 25](#_Toc185065423)

[BIBLIOGRAPHIE 26](#_Toc185065424)

**LISTE DES TABLEAUX**

Page

[Tableau 1.1 Titre](#_Toc150853242) x

**LISTE DES FIGURES**

Page

[Figure 1.1 Test](#_Toc150853834) x

|  |  |
| --- | --- |
| **LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES** | |
| **Nom :** | **Description :** |
| AI : | L'intelligence artificielle est défini par l’un de ses créateurs, Marvin Lee Minsky, comme « la construction de programmes informatiques qui s’adonnent à des tâches qui sont, pour l’instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau tels que : l’apprentissage perceptuel, l’organisation de la mémoire et le raisonnement critique »  http://fr.wikipedia.org/wiki/Intelligence\_artificielle |
| ARM | Nom du manufacturier produisant le microcontrôleur que nous utilison |
| Bloc d'alimentation: | Le bloc d'alimentation est la composante qui fournis les tensions exigés par tous les composantes du robots. Le bloc utilise les batteries DC 24v et génère des tensions de 24v, 12v, 5v et 3.3v. C'est tensions alimentent l'ordinateurs, l'écran, tous les capteurs ainsi que les récepteurs de télécommandes et le contrôleur. |
| Capteurs : | Le terme Capteur signifie, dans le cas de nos robots, l'ensemble de périphérique utilisé pour détecté des objet et obstacles ainsi que pour se positionner. Les robots du club étudiant Capra utilisent une caméra, laser SICK LMS291, une boussole et accéléromètre en 3 dimensions, ainsi qu'un GPS NovAtel possédant une borde différentielle. |
| Contrôleur: | Composante crée par le club étudiant Capra permettant d'effectuer un lien entre l'ordinateur des robots et les interfaces d'entrées et sortie utilisé par le robots. Le controleur utilise un lien CanBUS pour communiquer avec l'ordinateur et utilise un port série RS232 pour communiquer avec la « drive » de moteurs tout en ayant la possibilité de contrôler les divers entrées et sortie du système. Le contrôleur est aussi utilisé pour faire l'asservissement des moteurs du robots |
| Cortex-M3 | Famille du microcontrôleur utilisé |
| CRC | Cyclic Redondancy Check |
| DC: | Le courant continu (qui peut être abrégé par CC, ou DC, pour Direct Current), est un courant électrique unidirectionnel : le courant circule continuellement dans le même sens, le déplacement des électrons se fait toujours dans le même sens.  http://fr.wikipedia.org/wiki/Courant\_continu |
| DE2 | Carte de développement produite par Terasic |
| Diagramme de Gantt: | « Le diagramme de Gantt est un outil utilisé en ordonnancement et gestion de projet et permettant de visualiser dans le temps les diverses tâches composant un projet. Il permet de représenter graphiquement l'avancement du projet. »  <http://fr.wikipedia.org/wiki/Diagramme_de_Gantt> |
| Drive : | Le mot Drive vient de l'anglais et représente un contrôleur de moteur. Puisque notre robot contient déjà une pièce qui effectue le contrôle des moteurs, soit le contrôleur, nous utilisons le terme « drive » pour désigner le contrôleur de moteur Roboteq Ax1500. Il s'agit de la pièce qui gère la puissance de nos moteurs en lien avec des commandes RS232 reçu. |
| ÉLÉ | Nous parlons régulièrement de Batterie ÉLÉ, ou de la partie ÉLÉ du robot. Le robot est divisé en deux partie, soit par la partie électrique qui contient l'ordinateur, écran, bloc d'alimentation, le contrôleur ainsi que tous les capteurs utilisé dans le robots et bien entendu une batterie |
| HID | Human Interface Device |
| I/O | I/O vien de l'anglais, désignant les entrées et sortie d'un systeme. Il s'agit de l'abréviation de Input/Output |
| LED: | Une diode électroluminescente, abrégée sous les sigles DEL ou, plus couramment, LED (pour light-emitting diode) est un composant électronique capable d’émettre de la lumière lorsqu’il est parcouru par un courant électrique.  <http://fr.wikipedia.org/wiki/Diode_C3%A9lectroluminescente> |
| Microcontrôleur: | Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.  http://fr.wikipedia.org/wiki/Microcontr%C3%B4leur |
| MEC | La partie MEC du robot désigne la partie alimenté par la batterie MEC du robot. Elle est constitué de la « drive » de moteur et des moteurs eux même ainsi qu'une batterie. |
| Mode Intelligence artificielle | Le mode « Intelligence artificielle » est un mode de réception de la drive de moteur. Lorsque la drive est programmé en mode « Intelligence artificielle », elle s'attend à recevoir des données dans un protocole de communication RS232 qui ont le format !A7F. Ce qui signifie que le moteur A doit tourner à une vitesse de 7F (127) ce qui est la valeure maximal. |
| Mode Télécommande | Le mode « télécommande » est un mode de réception de la drive de moteur. Lorsque la drive est programmé en mode « télécommande », elle s'attend à recevoir des données sous forme de PWM, soit un signal de forme carré ou la largeur de la bande détermine la vitesse de chaque moteur. |
| NAK | Non Acknowledge |
| NES | Nintendo Entertainment System |
| Nintendo | Auteur de la console Nintendo Entertainment System |
| PID | Packet Identification |
| PWM : | La modulation de largeur d'impulsions (en anglais : Pulse Width Modulation, soit PWM), est une technique couramment utilisée pour synthétiser des signaux continus à l'aide de circuits à fonctionnement tout ou rien, ou plus généralement à états discrets.Le principe général est qu'en appliquant une succession d'états discrets pendant des durées bien choisies, on peut obtenir en moyenne sur une certaine durée n'importe quelle valeur intermédiaire. <http://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion> |
| Récepteur : | Le récepteurs est la composante qui reçois les signaux électro-magnétique provenant des télécommandes que nous utilisons. Nous avons deux récepteurs similaire qui sont utilisé pour récupérer les signaux provenant de la télécommande du robot ainsi que les signaux provenant de l'arrêt d'urgence sans fil. |
| RS232 : | Le RS-232 est une norme standardisant un bus de communication de type série sur trois fils minimum (électrique, mécanique et protocole). Disponible sur presque tous les PC jusqu'au milieu des années 2000.  http://fr.wikipedia.org/wiki/RS-232 |
| SPI | Serial Peripheral Interface |
| Télécommande : | La télécommande est le périphérique que nous utilisons pour contrôler nos robots à distance par une intervention humaine. Cette télécommande est très pratique quand vient le temps de déplacer nos robots ou de faire des démonstrations publique. |
| UART: | UART est un terme qui vient de l'anglais, ce sont les initiales de : Universal Asynchronous Receiver Transmitter. L'UART est donc un émetteur-récepteur asynchrone universel. En langage courant, c'est le composant utilisé pour faire la liaison entre l'ordinateur et le port série . L'ordinateur envoie les données en parallèle (autant de fils que de bits de données). Il faut donc transformer ces données pour les faire passer à travers une liaison série qui utilise un même fil pour faire passer tous les bits de données.  <http://fr.wikipedia.org/wiki/UART> |
| USB : | L’Universal Serial Bus (USB) est un bus informatique à transmission série servant à connecter des périphériques informatiques à un ordinateur. Le bus USB permet de connecter jusqu'à 127 périphériques à chaud et en bénéficiant du Plug and Play.  http://fr.wikipedia.org/wiki/Universal\_Serial\_Bushttp://fr.wikipedia.org/wiki/Courant\_continu |
| XBOX | Marque déposé de Microsoft pour sa console de jeux vidéo |

INTRODUCTION

Le club Capra a été fondé en 1996 par des étudiants, passionné de robotique, de l'école de technologie supérieure. Capra se spécialise dans la construction de robots autonome qui tentent de relever des défis au niveau de compétitions internationale dans le domaine de la navigation autonome. Grâce à l'implication d'une vingtaine de membre actif dans plusieurs domaines du génie tel que Mécanique, Électrique, Logiciel, Ti, Opérations et logistique ainsi qu'en production automatisée, le club est parvenu, cette été, à remporter un trophée pour une 3ieme place à la compétition de l'ION Automow (Institute of Navigation).

Pour le moment, le club Capra utilise deux robots fiables, soit RS3 et Herbinator. RS3, participe une compétition de parcours à obstacle dans lequel il doit parcourir un trajet qui lui est inconnu et qui est remplis d'embuche. Il doit atteindre la ligne d'arrivée, sans aucune intervention humaine plus rapidement que les autres équipes. Herbinator, pour sa part, est une tondeuse à gazon autonome capable de tondre des terrains de formes différentes tout en étant capable d'éviter les fleurs et le chien qui se ballade sur le terrain de la compétition.

Figure 2 RS3

Figure Herbinator

Les deux robots, qui effectue des taches bien différente, ont des caractéristiques électronique et logicielle très semblables. Ils utilisent une panoplie de capteurs de toute sorte reliée à un ordinateur qui effectue le traitement de toutes ces données pour s'orienter et se déplacer en toute quiétude.

Le club travaille présentement à améliorer et maintenir ces deux robots dans le but d'obtenir une première place lors d'une future compétition. Cette année, nous nous dirigerons vers

C’est donc par l’expérience des compétitions passés, les démonstrations de robotiques auxquelles nous participons dans le but de représenter l’école de technologie supérieure de Montréal ainsi que les nombreux tests effectué tout au long de l’année que nous avons remarqué plusieurs problèmes relié au système de gestion des télécommandes de nos robots.

Cette année, nous partirons, dans la semaine du 30 mai 2010, deux équipes distincte, avec deux robots distincts à deux compétitions qui se dérouleront en même temps en sol américain. L’une sera à détroit dans l’état du Michigan et l’autre à Dayton dans l’état de l’Ohio.

Au cours de la prochaine année, le club étudiant aimerait régler des problématiques de mobilité des robots en améliorant l'implémentation de la télécommande qui permet de contrôler le robot à distance. Ces télécommandes sont utilisé lors des divers événements publicitaires ou de démonstrations publique que nous participons ainsi que lors de la réalisation de test ou l'on doit déplacer les robots dans des endroits stratégiques.

# Analyse de la problématique

## Explication du système actuel

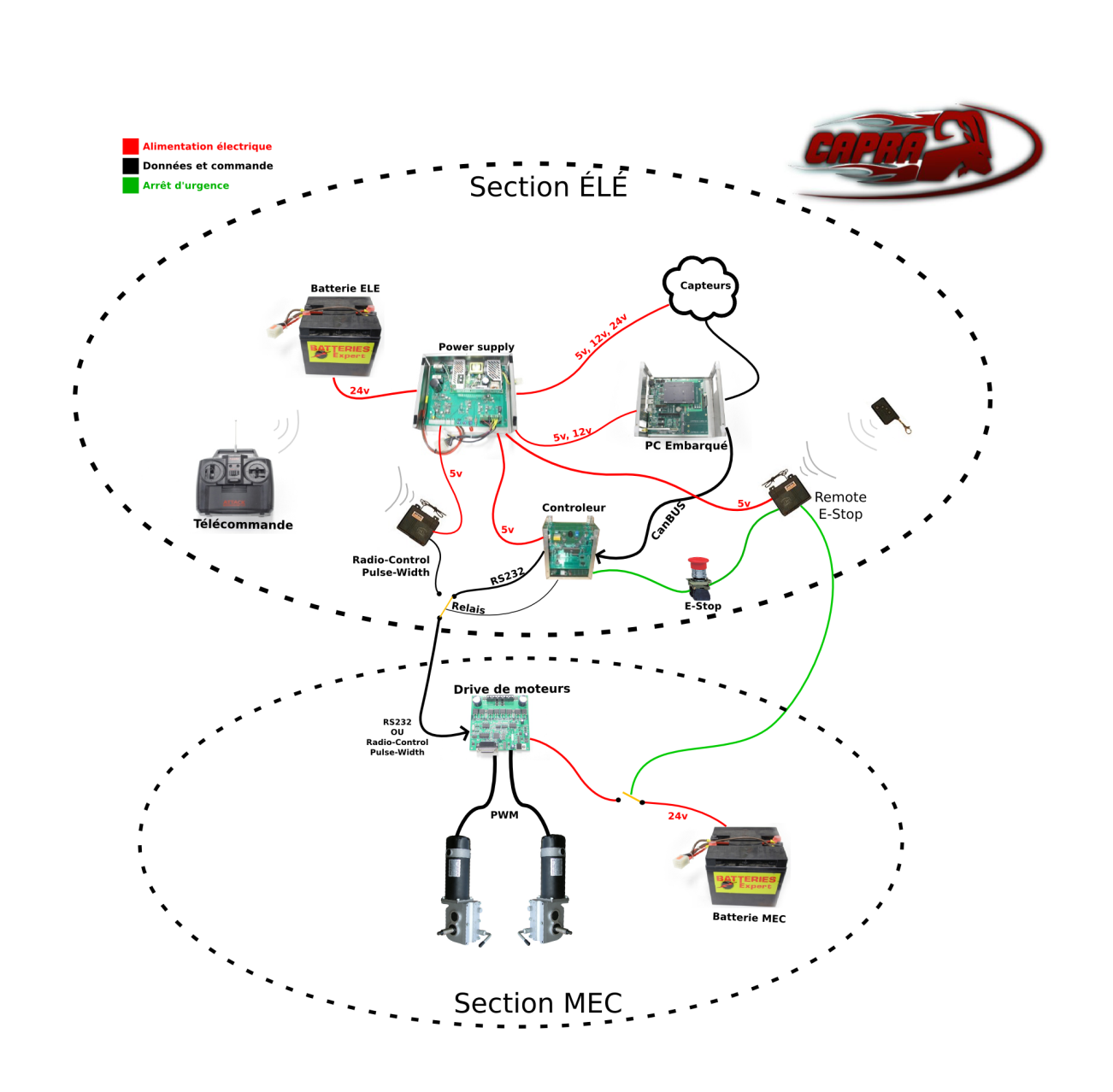
Pour bien être en mesure de comprendre les problèmes reliés au système de télécommande de nos robots, il importe de comprendre le fonctionnement de ceux-ci. Nous allons donc débuter par vous expliquer le principe de fonctionnement de nos robots. 

Figure Schéma actuel des robots

Ce diagramme représente les diverses composantes du robot en lien avec le système de télécommande.

Globalement, le système actuel est divisible en deux partie. Une partie ÉLÉ et une partie MEC. La partie MEC est alimentée par la batterie MEC et contient la « drive » de moteur ainsi que les deux moteurs. La partie ELE, pour sa part, est alimenté par la batterie ELE et contient l'ordinateur embarqué, tous les capteurs, le contrôleur, le bloc d'alimentation, ainsi que les deux télécommandes et récepteurs. Les alimentations des deux parties sont importantes parce qu'elles sont, en bonne partie, une sources des problèmes reliés aux télécommandes. Nous pouvons aussi apercevoir dans le schéma le système d'arrêt d'urgence en vert.

La partie la plus importante se trouve à être la chaine de commande des moteurs. Pour commencer, les données peuvent provenir de l’intelligence artificielle qui se trouve dans le PC embarqué du robot. Les commandes sont transmises au contrôleur par le biais d’un port CANBUS. Par la suite, le contrôleur transmet ces données à la drive de moteur par un lien RS232 en passant par un relais qui aiguille les diverses sources de données pouvant se rendre à la drive. Finalement, la drive de moteur exécute les commande en envoyant les bonnes tensions aux moteurs.

Dans un autre ordre d’idée, il est possible d’utiliser la télécommande des robots. Le récepteur de cette télécommande utilise deux PWM ou la modulation de la largeur de bande représente l’état des deux contrôles sur la télécommande. Notre drive de moteurs est en mesure d’interpréter ces signaux si elle est programmée dans le mode PWM. Donc, pour utiliser cette télécommande, le contrôleur doit envoyer un message à la drive pour la programmer dans le mode PWM et ensuite changer l’état du relais pour aiguiller ce dernier sur l’entrée du récepteur de la télécommande. Si nous désirons revenir en mode intelligence artificielle, le contrôleur modifie l’état du relais pour aiguiller l’entrée sur le lien RS232 provenant du contrôleur et envoie encore une fois un message à la drive de moteur de se programmer en mode RS232.

De plus, les alimentations relié au système ont aussi une importance dans la problématique que nous allons vous expliquer.

Selon le diagramme, nous constatons aussi que les divers récepteurs et pièces sont tous alimenté par le bloc d’alimentation principale à l’exception de la drive de moteur qui elle est alimenté par la batterie Mec du robot. La drive de moteur se trouve donc sur une alimentation séparé du reste du robot.

## Division en sous-problèmes

Cette structure nous apporte plusieurs problèmes de toute sorte.

Premièrement, la télécommande actuelle ne fonctionne que sur une très courte distance et la réception est, très souvent, mauvaise. Parfois le robot ne bouge plus du tout ou se déplace de manière très saccadé. La solution devra donc permettre une réception plus fiable et sur une plus grande distance.

De plus, la télécommande utilisée par Capra consume énormément et rapidement les piles. Plus les piles deviennent faible dans la télécommande, plus la réception est mauvaise par les robots. Les télécommandes utilisent chacune 8 piles AA et nous ne disposons pour l'instant que d'un seul chargeur permettant de charger 8 piles à la fois. Cette charge prend environ une journée complète. Pour charger les piles de nos deux télécommandes, nous devons impérativement les charger en deux jours, en s'assurant que quelqu'un soit présent au local du club étudiant pour effectuer une rotation des piles. La gestion de cette charge étant effectué par plusieurs personnes et la communication au sein du club étudiant étant relativement déficiente, la charge de ces piles est un véritable cauchemar. La solution devra donc faciliter le cycle de charges des piles de télécommandes.

Par la suite, les piles utilisées dans les télécommandes se déchargent par eux même. Si les piles sont oubliées dans la télécommande, malgré le fait que cette dernière soit éteinte, les piles se vident complètement en 24 heures. Si nous enlevons les piles de la télécommande, elles se vident tout de même par elle même en deux semaines. Il est donc impossible pour nous de tenir les piles chargé en tout temps. Si nous avons besoin d'utiliser la télécommande rapidement, les chances sont très faibles que nous ayons des piles chargées.

Ensuite, la configuration actuelle de la télécommande implique que la drive de moteur soit configuré en mode « Télécommande » pour traiter les signaux provenant de la télécommande et en mode « Intelligence artificielle » pour traiter les signaux provenant de l'intelligence artificielle. Cette programmation cause beaucoup de problèmes parce qu'elle n'est pas fiable et qu'il est difficile pour nous de nous assurer que la « drive » est dans le mode désirée. Il arrive fréquemment que nous devons brancher un ordinateur dans la « drive » pour la reprogrammer dans le but de pouvoir utiliser la télécommande. Cette programmation est expliqué par le fait que l'intelligence artificielle utilise le protocole RS232 pour communiquer avec la « drive » et la télécommande utilise un PWM (pulse width modulation) pour transmettre les vitesses de chaque moteurs. De plus, nous avons récemment appris, à l’aide du manuel d’utilisateur de la drive de moteur, que nous pouvons la programmer un maximum de 1000 fois. Pour l’instant, nous la programmons au minimum 3 fois à chaque fois que nous utilisons le robot et nous commençons aujourd’hui à avoir beaucoup de problèmes avec nos drives de moteurs qui sont probablement lié de près à cette indication. Le club Capra désire donc trouver une solution qui permet d'éliminer le besoin de reprogrammer la « drive » de moteur lorsqu'on passe du mode « télécommande » au mode « intelligence artificielle ».

Finalement, lorsque nous désirons utiliser la télécommande pour contrôler les robots, nous devons allumer l'ordinateur du robot, qui lui fait démarrer le bloc d'alimentation qui alimente toutes les composantes électronique du robot (capteur, contrôleur, écran, ordinateur, etc). Le fait de tout alimenter le robot lorsque nous n'utilisons que très peu de matériel devient problématique puisque les piles du robot se vident très rapidement. En mode télécommande, le robot devrait alimenter seulement les composantes nécessaires à la motricité sans alimenter les capteurs, l'ordinateur, l'écran ainsi que tous les autres périphériques inutile au mouvement en mode télécommandé.

## Étude de la complexité

<Texte>

# Choix d’une solution

## NES_controller.jpgJet d’idées

Notre idée de départ était d’intégrer une télécommande de Nintendo 8 bit.



Par la suite, nous avons pensé garder la même télécommande que nous avons actuellement pour nos robots mais en la branchant dans un système qui pourrais convertir le PWM en RS232.



Nous avons aussi analyser la possibilité d’utiliser une télécommande de Playstation 3 sans fil



Et finalement une télécommande de XBOX 360 sans fil

## Définition des concepts

Premièrement la télécommande de Nintendo à un grand avantage et un grand désavantage. Premièrement, elle possède un fil. Ce fil nous assure une communication parfaite sans aucune perte de donnée et qui est par le fait même extrêmement fiable. De plus, le protocole de communication de cette télécommande est très simple et facile à intégrer. Son désavantage est qu’elle possède un fil. En effet, notre portée avec le robot se trouve à être très limité. De plus, ces boutons sont de type « Tout ou rien » ce qui pourrais être dangereux dans le cas de notre robot puisque nous n’aurions aucun contrôle de la vitesse du robot mais seulement de la direction dans lequel il se dirige.

Nous avons ensuite analysé la possibilité de garder nos télécommandes actuelles en les modifiant afin de régler certain problèmes. Premièrement, cette télécommande devra être branchée dans une carte qui convertirais le PWM en RS232 pour éliminer le problème de reprogrammation de la drive de moteur. De plus, des piles au lithium ou au nickel devraient être installé pour augmenter la durée de vie de ces dernières. Le fait de modifier les piles devraient, en temps normal, augmenter la puissance de la transmission et par le fait même augmenter la portée du système.

Nous avons ensuite analysé la possibilité d’inclure une télécommande de Playstation 3 sans fil. Cette télécommande à un grand avantage, soit celui de d’avoir une pile qui à fait ces preuves. De plus, la charge de cette pile peut être assurée tout simplement par le branchement d’un câble USB avec un ordinateur personnel. Cette télécommande à par-contre un désavantage. Elle utilise, comme la plupart des télécommandes de sa génération, la technologie Bluetooth pour communiquer avec les récepteurs. Cette technologie bien qu’efficace ne possède une portée que d’environ 10m de rayon, ce qui est insuffisant pour nos besoins.

Finalement, la télécommande de XBOX 360 à aussi le même avantage que la télécommande de Playstation, soit le fait qu’elle possède une batterie au lithium intégré qui se charge par un simple câble USB relié à un ordinateur personnel. Cette télécommande possède un autre avantage de taille : Son protocole de transmission des données se fait par un lien de communication fermé pour lequel les détails ne sont pas vraiment disponibles par Microsoft mais nous savons qu’elle n’utilise pas la technologie Bluetooth. Après avoir effectué plusieurs test, nous nous sommes aperçu que la portée de cette télécommande dépassent amplement nos attentes et nos besoins.

## Choix d’une solution

Après de longue discutions, nous avons choisis d’intégrer la télécommande de XBOX à notre système. Ce choix étant risqué étant donné la complexité de l’intégration du USB dans un microcontrôleur, nous avons choisis d’implémenté en parallèle la conversion du PWM de nos commandes actuelles dans notre projet. Le choix de garder la télécommande RF actuelle implique de modifier cette dernière dans le but d’y intégrer une pile au lithium ou au nickel pour augmenter la durée de vie de ces piles ainsi que sa portée. Notre système fonctionnera comme suit : Nous aurons une carte dans laquelle nous brancherons les deux télécommandes (la télécommande de XBOX 360 et la télécommande RF actuelle des robots). Nous brancherons aussi dans cette carte un lien RS232 avec le contrôleur et un autre lien RS232 avec la drive de moteur. De cette manière, en temps normal, nous pourrons simplement répéter les données provenant du contrôleur vers la drive de moteur et ce sera complètement transparent pour l’intelligence artificielle. Les commandes obtenu provenant des télécommandes serons convertis en RS232 sous le même format utilisé par la contrôleur et un système de gestion de priorité (Multiplexeur) aura pour tache de diriger les bons signaux aux bon endroit. De plus, cette carte aura pour tâche d’alimenter les récepteurs des télécommandes et cette carte sera alimenté par la batterie MEC du robot. Ce qui aura pour conséquence d’obtenir la possibilité d’utiliser les télécommande avec tout simplement une batterie MEC et sans même avoir le besoin de démarrer la section ÉLÉ du robot. De plus, le système d’arrêt d’urgence sera modifié pour être controlé à partir de cette carte aussi dans le but de pouvoir débloquer les moteurs sans avoir besoin d’alimenté la section ÉLÉ du robot.

Le schéma suivant explique la solution retenue de manière plus visuelle :

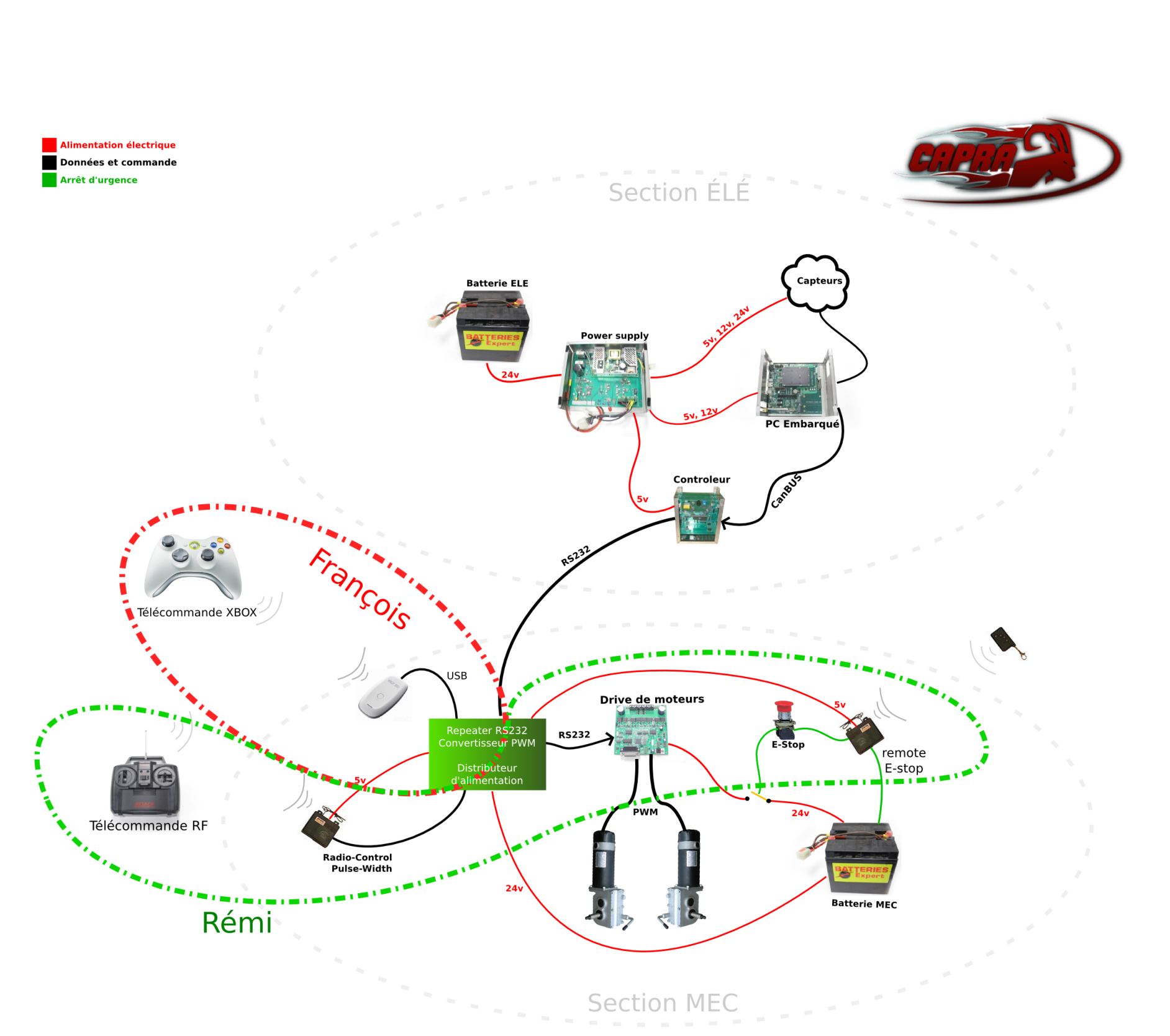


Figure Schéma de la solution

# Risques et opportunités

## Risques

* Les outils de compilation pour le microcontrôleur choisi ne sont pas disponibles.
* L’utilisation du ARM pourrait être plus complexe que prévue notamment au niveau des interruptions.
* La familiarisation des membres avec la programmation orientée objet dans un microcontrôleur peut être longue
* La création du circuit imprimé peut être plus longue que prévue
* Il pourrait être difficile de trouver un contrôleur USB à la fois facile d’utilisation, avec une bonne documentation et fournis sur une plateforme de prototypage.
* Dépendamment de la vitesse de la communication minimal entre le microcontrôleur et le contrôleur USB, il faudrait peut être produire nous même la plateforme de prototype ce que pourrait grandement jouer sur notre emploi du temps serré.
* La communication entre le microcontrôleur et le contrôleur USB pourrait être difficile à déverminer.
* Comme la vitesse des transactions USB est très rapide, certains problèmes liés à la fréquence élevée pourraient survenir. De plus, les transactions du bus seront difficiles à analyser.
* Plusieurs pièces devront être commandés sur Internet et ceci le plus tôt possible. Ceci rentre en conflit avec le fait que le contrôleur USB n’est pas choisi avant la moitié du projet. De plus le projet ne pourrait surement pas être fait dans les temps si certaines pièces arrivent endommagé ou si nous endommageons une pièce durant le développement.
* Les commandes venant de la télécommande XBOX devront être acquises par la lecture du code de pilotes du noyau linux et l’ingénierie inverse à l’aide d’un analyseur USB. Cette tâche pourrait possiblement être trop compliqué et prendre beaucoup plus de temps que prévue.

Opportunités

* La carte de développent DE2 est disponible à l’école et inclue un contrôleur USB. Celle-ci pourrait possiblement être utilisée.
* Batterie Expert est l’un des commanditaires du club étudiant Capra et pourrais nous donner des informations précieuse quant à la modification de la télécommande actuelle.
* L’existance d’un équipe logiciel au sein du club Capra pourra très probablement nous aider à planifier la gestion du code à l’intérieur du microcontrôleur.
* Labo circuit est aussi un des commanditaires du club étudiant Capra et pourrais possiblement nous fournir le circuit imprimé.

# Réalisation

## Modification de la télécommande actuelle

Pour débuter, nous nous sommes informé au près de Batterie Expert dans le but d’avoir plus amples informations au sujet des piles en général. Après avoir longuement discuté avec eux, nous avons fait un choix de piles. Nous avons choisis d’acheter un chargeur à batterie ainsi que de se faire fabriquer sur mesure un pack de batterie composé de 8 cellules au nickel permettant d’obtenir 2400mah, ce qui était près des spécifications des piles rechargeable que nous disposons au format AA à l’intérieur de notre télécommande. Par la suite, nous avons démonté et modifié l’intérieur de la télécommande à l’aide d’un Dremel dans le but d’enlever complètement le support de batterie AA à l’intérieur dans le but d’y intégré le pack de batteries. Puisque ces batteries ne peuvent maintenant plus être retirées de la télécommande, nous avons utilisé un connecteur à l’intérieur du compartiment à batterie qui permettra de brancher directement la batterie dans le chargeur sans la retirer du compartiment de la télécommande. Cette modification à été apporté a notre télécommande il y a maintenant plus de trois mois, et la pile est encore chargée! La durée de vie de cette pile est incroyable longue. De plus, comme prévu, la portée de la télécommande à été amélioré considérablement.

## Choix du microcontrôleur

Notre système comprenant maintenant deux port série, un entrée de capture de PWM ainsi que le besoin d’un support USB Host, nous avions besoin d’un microcontrôleur capable de gérer le tout. Nous avons regardé plusieurs microcontrôleurs soit un Atmega32, un Atmega128 et un Arm Cortex M3. Nous avons finalement choisis le Arm Cortex M3 pour deux raison. Pour commencer, sa puissance de calcul est définitivement plus grande que celle du Atmega32 et du Atmega128. Avec sa fréquence d’horloge cadencée à 72 MHz et plusieurs technologies telles que son pipeline et la possibilité de faire des accès DMA était très prometteur. De plus, ce microcontrôleur nous était inconnu et il est maintenant plus populaire que jamais puisqu’il fait aujourd’hui parti de plusieurs produit qui ont fait leurs preuves comme par exemple le IPhone de Apple qui utilise un autre microcontrôleur Arm très semblable au Cortex M3. Nous avons aussi choisis ce processeur par la possibilité de développer en C++ orienté objet, ce qui nous a permis de concevoir une architecture claire et qui resteras hautement maintenable dans le futur, ce qui est une chose souvent négligé au sein des clubs étudiants.

## Compilation et configuration du compilateur

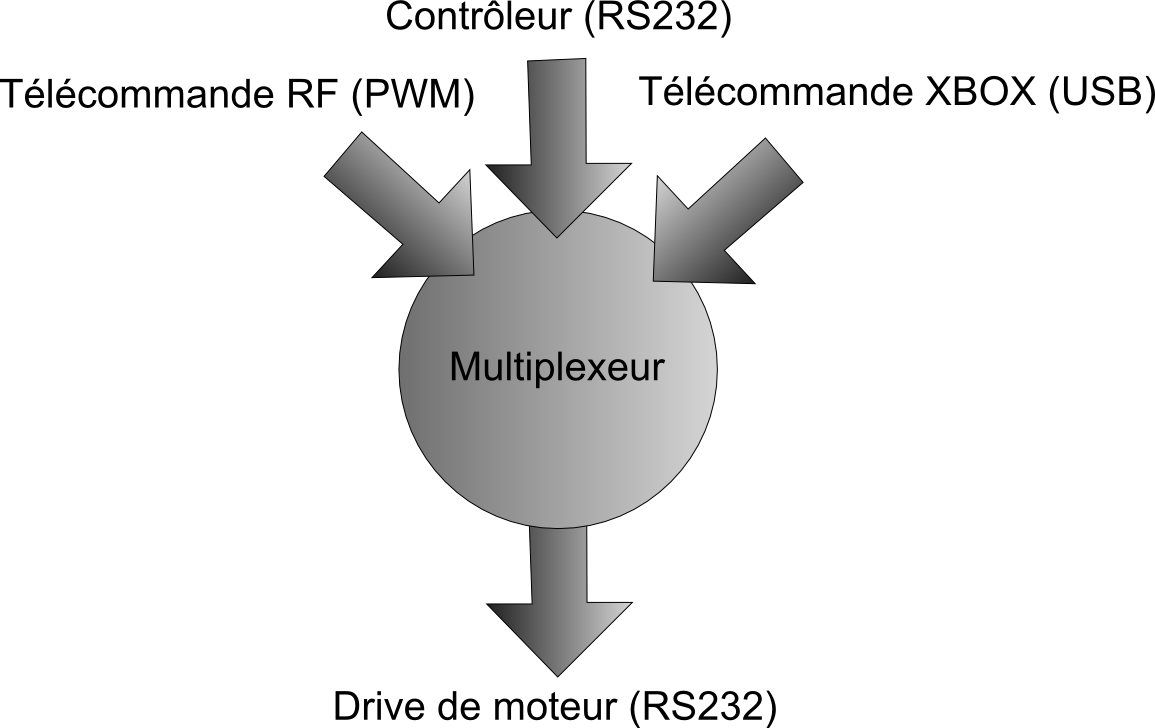
## Architecture de notre système (UML)

## Communication RS232

## Capture du PWM de la télécommande

## Implémentation du système de choix de priorité des sources

Notre carte à pour tache de recevoir des données provenant de diverses sources, soit, la télécommande de XBOX 360, du

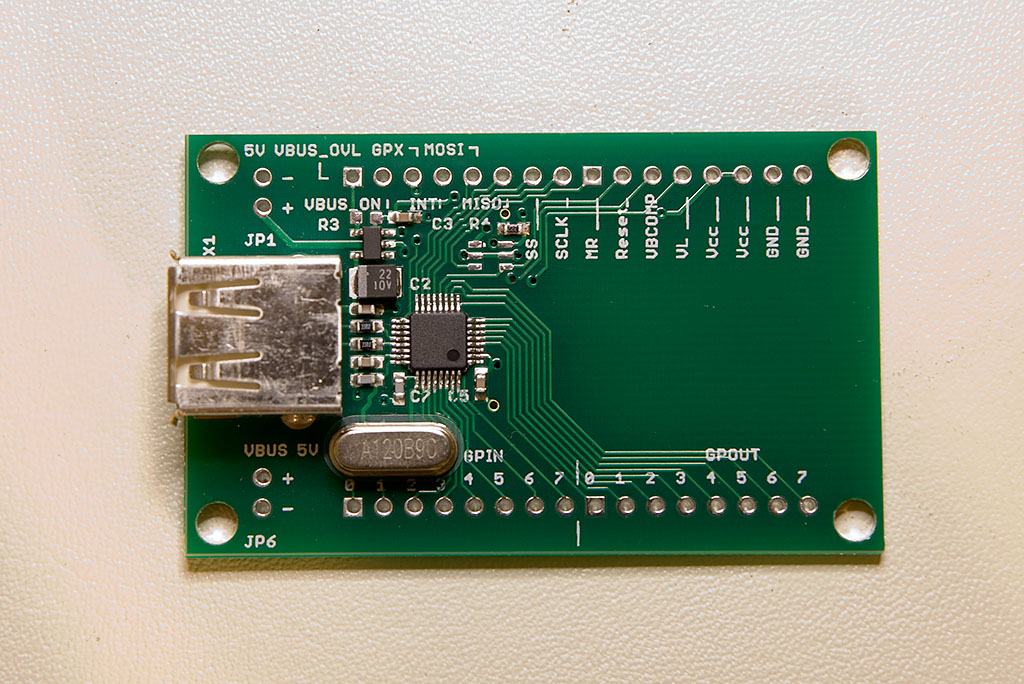


## Choix du contrôleur USB

Le choix du contrôleur n’est pas chose facile. Plusieurs critères doivent être tenu en compte. Dans notre cas, les principaux étaient les suivants en ordre de priorité :

* Support du mode de communication USB par interruption
* Facilité d’accès à la documentation
* Facilité de raccord au microcontrôleur
* Faible à moyen niveau d’abstraction pour ne pas perdre le coté éducatif

Le contrôleur choisi est le MAX3421E de la compagnie Maxim. Cette puce supporte tous les modes de communication USB soit Control, Bulk, Interrupt et Isochronous. De plus, elle est pourvue d’une fiche technique, d’un guide de programmation et d’un exemple couvrant l’énumération. Du coté de l’abstraction, ce contrôleur prend soin de la couche 1 de la communication. En d’autre terme, outre le fait d’envoyer le bon niveau sur les lignes de transmission, il s’occupe d’envoyer le PID, du numéro de séquence des paquets, et d’une certaine partie de la gestion d’erreur. Pour les curieux, ceci consiste principalement en le calcule du CRC et la vérification de celui-ci lors de la réception. De plus, si le CRC permet de détecter une erreur, le contrôleur fera lui-même la demande du même paquet à nouveau. Il est à noter que ceci peut avoir un effet pervers. En effet lors de la conception du pilote, certaines erreurs d’implémentation peuvent être cachées dû au fait que le contrôleur USB redemande automatiquement les paquets en cas d’erreur.



## Conception du pilote USB

### Détection du périphérique

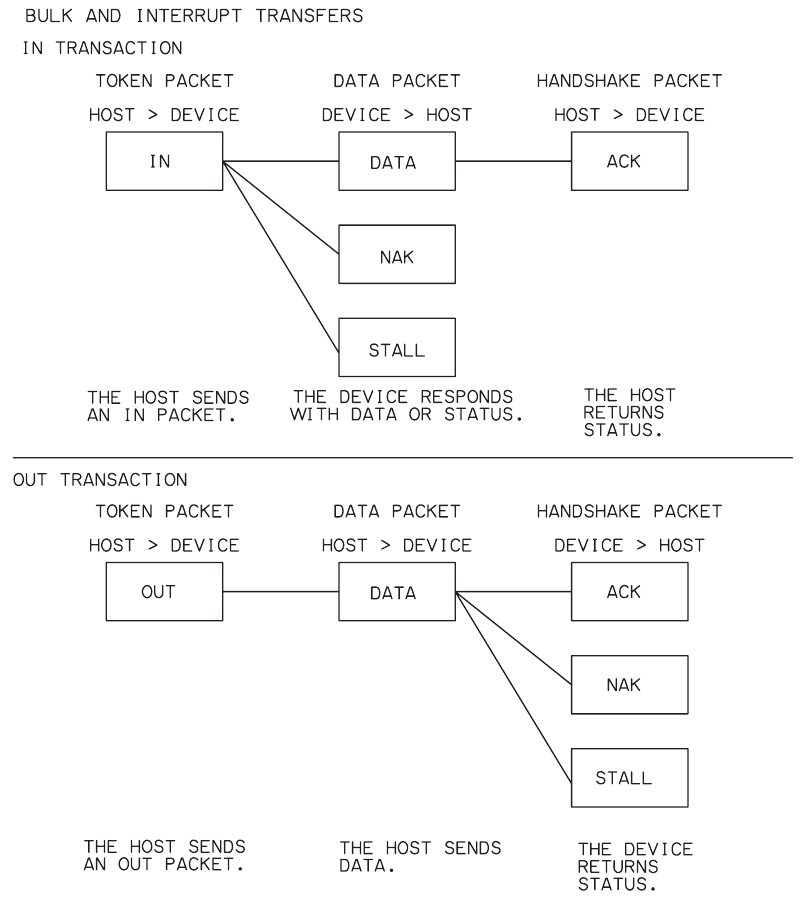
### Énumération du périphérique

### Requêtes HID

Tel que mentionné précédemment le USB comporte plusieurs mode de fonctionnement soit Control, Bulk, Interrupt et Isochronous. Cependant, ceci reste assez large quand on considère tous les périphériques USB disponible. C’est pourquoi ceux qui ont développé la spécification USB ont aussi mis en place des classes de périphérique. Ceci permet de simplifier le développement de pilotes car tous les périphériques faisant partie d’une même classe ont le même mode de fonctionnement. Par exemple le HID est une classe du mode de transmission Interrupt. Cette classe définit qu’un périphérique doit envoyer et recevoir l’information sous forme de rapport. Il est important de mentionner que la réception d’un rapport par un périphérique HID est optionnelle selon la norme. Par contre, la télécommande Xbox implémente cette fonctionnalité et in en sera question dans la section correspondante.

Afin de faire la demande d’un rapport, l’hôte doit envoyer un jeton au Endpoint Interrupt avec un bit indiquant une direction entrante. Une fois ce jeton reçu, le périphérique USB enverra un rapport si celui-ci à de l’information à partager. Dans le cas échéant, il envoie le paquet NAK. Le même principe est utilisé pour envoyé un rapport au périphérique seulement dans ce cas le bit de direction du jeton indique une direction sortante. Une fois ce jeton envoyé, l’hôte envoie le rapport au périphérique. Comme vous pouvez le constater, le bit de direction indique toujours une direction par rapport à l’hôte.

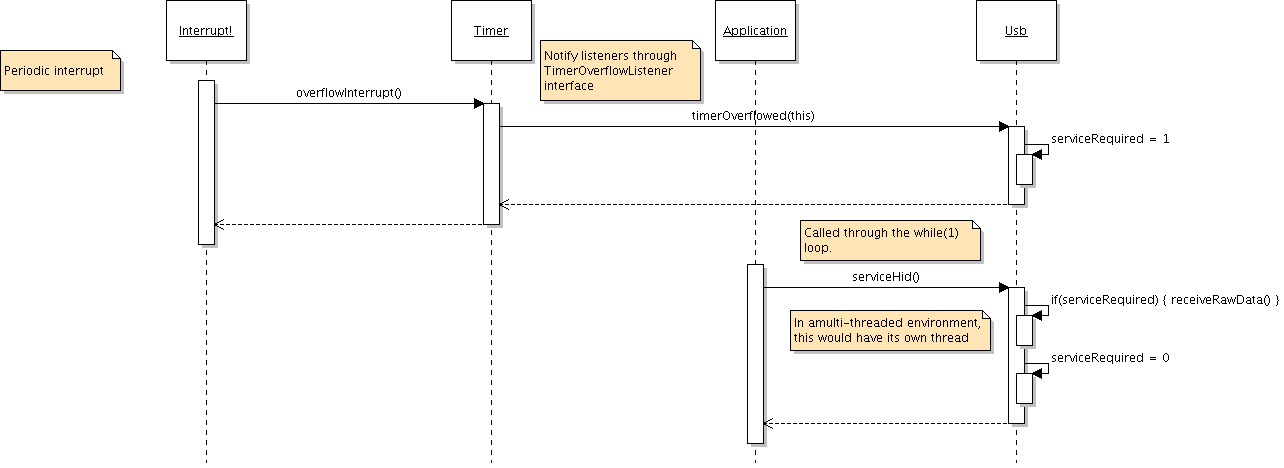
Pour une meilleure compréhension de la communication USB par interruption en général, la meilleure façon est probablement de jeter un œil au diagramme du livre « USB Complete ».



Il est important de noter que bien que le mode de transmission HID définit la communication par rapport, celui-ci ne définit pas le format des rapports sauf pour les périphériques courants. Plus d’information sur les rapports sont fournis dans la section correspondante.

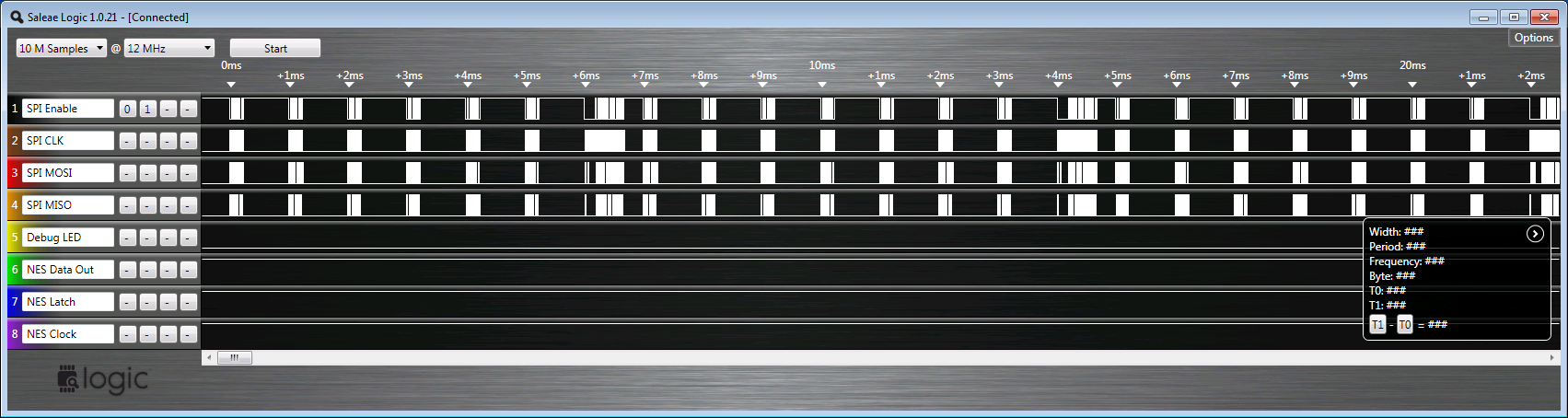
### Période des requêtes HID

Selon le standard USB, un périphérique se conformant à la norme HID doit fournir le temps entre les requêtes à l’hôte lors de l’énumération. Pour un bon fonctionnement du périphérique, ces temps doivent être respectés. Dans le cas de la télécommande du XBOX, celle-ci doit être interrogé tous les millisecondes. Le diagramme de séquence suivant indique la façon de faire au niveau du logiciel.



À tous les millisecondes, un timer déborde et met à jour un fanion indiquant que le service du périphérique est requis. Une fois ceci fait, la prochaine fois que le processeur aura le temps, il fera une requête à la télécommande.

En utilisant l’analyseur logique, il est possible de regarder les transactions entre le processeur et le contrôleur USB pour voir si ces délais sont respectés en pratique.



La communication entre le microcontrôleur et le contrôleur USB se fait en SPI. Lorsque nous ne voyons aucune activité sur le bus, c’est que les transactions son terminé pour cette période de temps. Nous remarquons que le processeur passe beaucoup de temps dans ces communications mais ceci est une facette que nous traiterons dans les prochaines sections.

### Temps alloué au service USB

Lors du service du périphérique USB, le microcontrôleur communique constamment avec le contrôleur USB par SPI. Il est donc possible de voir facilement le temps alloué pour ce service en regardant la communication entre ces deux puces. De plus, nous pouvons en déduire le temps libre pour les autres tâches du processeur. Sur cet ordre d’idée, il est important de réduire le temps des communications aux maximums car plus le facteur d’utilisation du processeur se rapproche de 1, plus nos chances de respecter le temps entre les services HID deviennent faible. Afin de mettre cela en perspective, une étude plus poussé de ces communications SPI s’impose.

Il est important de mentionner que notre analyse sera basée sur la communication la plus longue. Dans notre cas, cette communication se produit lorsqu’il y a un changement d’état de la télécommande de XBOX.





Les deux captures précédente démontre que sur une période de 1 milliseconde, il reste 93.5 microsecondes de libre au processeur. Ceci revient à dire que le service USB le plus long dure 0.9065 millisecondes. Comme nous connaissons la vitesse de transmission et le nombre d’octet transmit, il est possible d’en déduire le temps requis pour le traitement du USB. Voyons tout ceci en détail.

Cette communication est composée de 56 octets. La fréquence d’opération du SPI pour cet exemple est la fréquence des périphériques divisée par 128. Dans notre cas, a fréquence des périphérique est la même que la fréquence du processeur une fois multipliée, soit 72Mhz. Ceci donne une fréquence du SPI de 562.5 kHz. Avec 8 bits par octet, la communication de 56 octets devrait prendre 0.7964 milliseconde.

À l’aide de ce résultat nous en déduisons que le service USB pour la communication la plus longue requière 0.1101 millisecondes du processeur. Il est maintenant possible d’observer le facteur d’utilisation du processeur pour le service USB en fonction de la fréquence SPI de façon théorique. Nous faisons l’expérience sur la plage de fréquence supportée par le Cortex-M3 et possible d’atteindre avec le MAX3421E.

De ce graphique il est possible de déterminer le meilleur compromis en un coup d’œil. Il est bon de prendre note qu’un facteur d’utilisation de plus de 100% indique qu’il nous est impossible de finir notre service du périphérique en 1 millisecondes. Nous allons donc prendre constamment du retard ce qui risque de mettre le périphérique USB dans un mode d’opération imprévisible. Une fréquence du bus SPI de moins de 500 kHz est donc déconseillée.

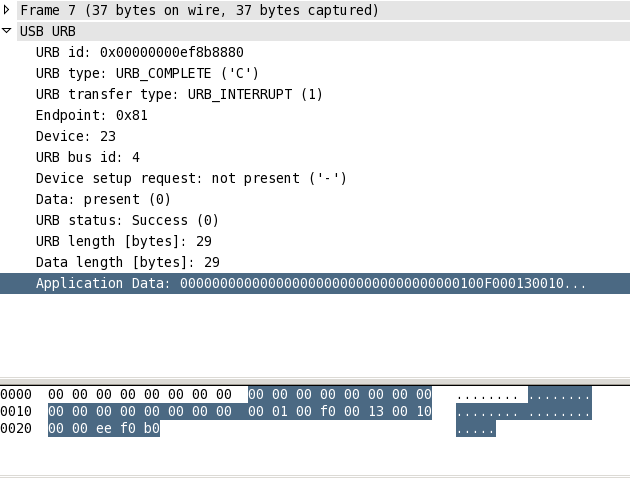
### Décodage des rapports HID

Lorsque nous faisons le service du périphérique USB, nous ne demandons en fait qu’un rapport de ce dernier. La norme USB à établie le format du rapport pour certains périphériques standard comme le clavier et la souris d’un ordinateur. Cependant, le constructeur est libre de choisir le format qu’il désire lors de la conception de son périphérique. Comme nous pouvons l’imaginer, Microsoft à utilisé son propre format pour sa télécommande.

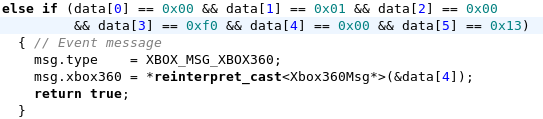
Afin de bien définir le format du rapport, j’ai utilisé le site officiel de l’ingénierie inverse du XBOX 360, free60.org. Celui-ci ne dispose d’information que pour la télécommande avec fil mais ceci est néanmoins en très bon point de départ. Ce site définit le rapport comme suit :

|  |  |  |  |
| --- | --- | --- | --- |
| **Offset** | **Length (bits)** | **Description** | **Windows driver** |
| 0x00.0 | 8 | Message type |  |
| 0x01.0 | 8 | Packet size (20 bytes = 0x14) |  |
| 0x02.0 | 1 | D-Pad up | D-Pad up |
| 0x02.1 | 1 | D-Pad down | D-Pad down |
| 0x02.2 | 1 | D-Pad left | D-Pad left |
| 0x02.3 | 1 | D-pad right | D-Pad right |
| 0x02.4 | 1 | Start button | Button 8 |
| 0x02.5 | 1 | Back button | Button 7 |
| 0x02.6 | 1 | Left stick press | Button 9 |
| 0x02.7 | 1 | Right stick press | Button 10 |
| 0x03.0 | 1 | Button LB | Button 5 |
| 0x03.1 | 1 | Button RB | Button 6 |
| 0x03.2 | 1 | Xbox logo button |  |
| 0x03.3 | 1 | Unused |  |
| 0x03.4 | 1 | Button A | Button 1 |
| 0x03.5 | 1 | Button B | Button 2 |
| 0x03.6 | 1 | Button X | Button 3 |
| 0x03.7 | 1 | Button Y | Button 4 |
| 0x04.0 | 8 | Left trigger | Z-axis down |
| 0x05.0 | 8 | Right trigger | Z-axis up |
| 0x06.0 | 16 | Left stick X-axis | X-axis |
| 0x08.0 | 16 | Left stick Y-axis | Y-axis |
| 0x0a.0 | 16 | Right stick X-axis | X-turn |
| 0x0c.0 | 16 | Right stick Y-axis | Y-turn |
| 0x0e.0 | 48 | Unused |  |

Afin de vérifier la concordance avec la télécommande sans fil, un analyseur USB fût utilise. En bref, cet outil permet d’afficher toutes les transactions USB faite par le noyau du système d’exploitation. Voici un rapport USB visualisé à l’aide d’un tel analyseur :

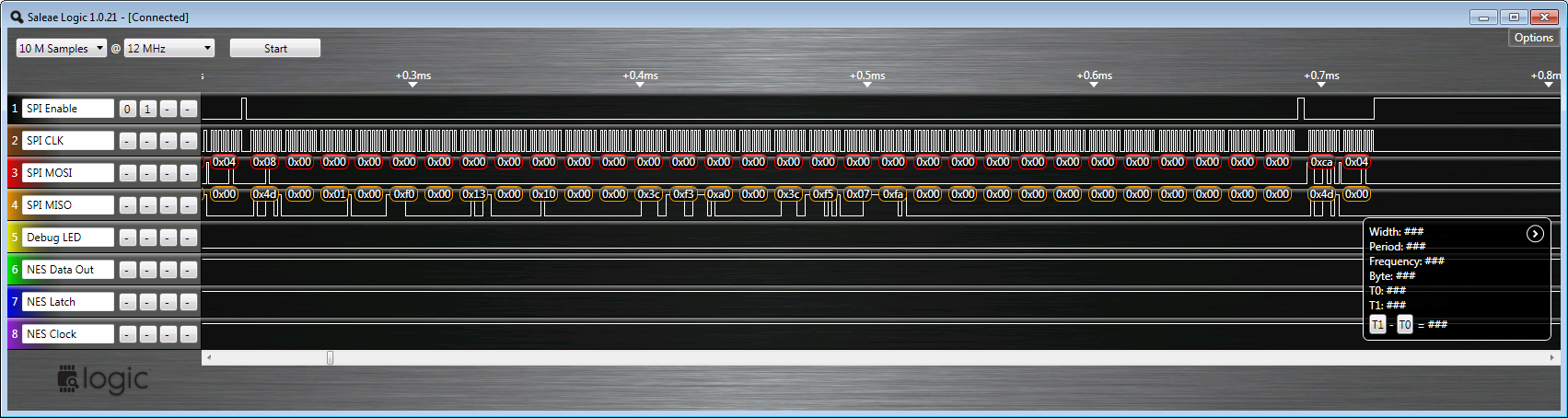


Lors que nous enfonçons le bouton A de la télécommande, nous recevons le message de la capture précédente. Il et important de mentionner que le rapport commence à l’adresse 0x18 sur les captures. Ce rapport en particulier commence donc par les octets 0x00, 0x01, 0x00, 0xF0, 0x00, 0x13. Selon le tableau faisant la description du rapport, le bit à la position 0x03.4 devrait être à 1. Cependant ce n’est pas le cas ici. Par contre, le bit 0x07.4 est à 1 ce qui nous donne une piste. L’analyse du code source du pilote xboxdrv-linux permettra de vérifier nos présomptions. En effet, ce code supporte la télécommande sans fil. La section traitant les évènements de la télécommande est particulièrement intéressante :



Cette condition vérifie l’information du rapport et ci celui-ci commence par la séquence 0x00, 0x01, 0x00, 0xF0, 0x00, 0x13, il transforme cette information en un message de la télécommande de type XBOX 360 \*avec\* fils mais avec un décalage de 4 octets. D’où le passage de l’adresse de la quatrième case de tableau. En d’autre terme, l’octet 4 correspond à l’octet 0 du rapport de la télécommande avec fil. Ceci est très intéressant car en regardant la capture prise avec l’analyseur USB nous constatons la même séquence de début de rapport. De plus lorsque nous avons enfoncé le bouton A, le bit à la position 0x07.4 était à 1. En tenant compte du décalage de 4 octets, ceci correspond au bit 0x03.4 du tableau ce qui est bien le bouton A.

Lors de l’implémentation du pilote, nous avons procédé à plusieurs autres tests qui ont confirmé la correspondance entre l’information reçu et le tableau de free60.org. La figure suivante est un rapport capturé à l’aide de l’analyseur logique et démontre la cas où le bouton A est enfoncé. Ce paquet est exactement le même que celui reçu avec l’analyseur USB.



### Rapports HID sortant

Comme introduit précédemment, la télécommande est conçue pour recevoir des rapports sortants. En d’autres termes, des rapports de l’hôte vers le périphérique. Dans notre cas, ces rapports sont utilisés essentiellement pour contrôler les LED sur la télécommande. Ceci est un détail important car par défaut les 4 LEDs flash indéfiniment ce qui est agaçant pour l’utilisateur même lors d’une courte utilisation.

Encore une fois, le format du rapport à dû être déterminé en utilisant plusieurs sources d’information. Le site Internet free60.org fût le point de départ. Ce site fournit un format de rapport sur 3 octets incluant une certaine commande. La liste de commande est la suivante :

|  |  |
| --- | --- |
| **Pattern** | **Description** |
| 0x00 | All off |
| 0x01 | All blinking |
| 0x02 | 1 flashes, then on |
| 0x03 | 2 flashes, then on |
| 0x04 | 3 flashes, then on |
| 0x05 | 4 flashes, then on |
| 0x06 | 1 on |
| 0x07 | 2 on |
| 0x08 | 3 on |
| 0x09 | 4 on |
| 0x0A | Rotating (e.g. 1-2-4-3) |
| 0x0B | Blinking\* |
| 0x0C | Slow blinking\* |
| 0x0D | Alternating (e.g. 1+4-2+3), then back to previous\* |

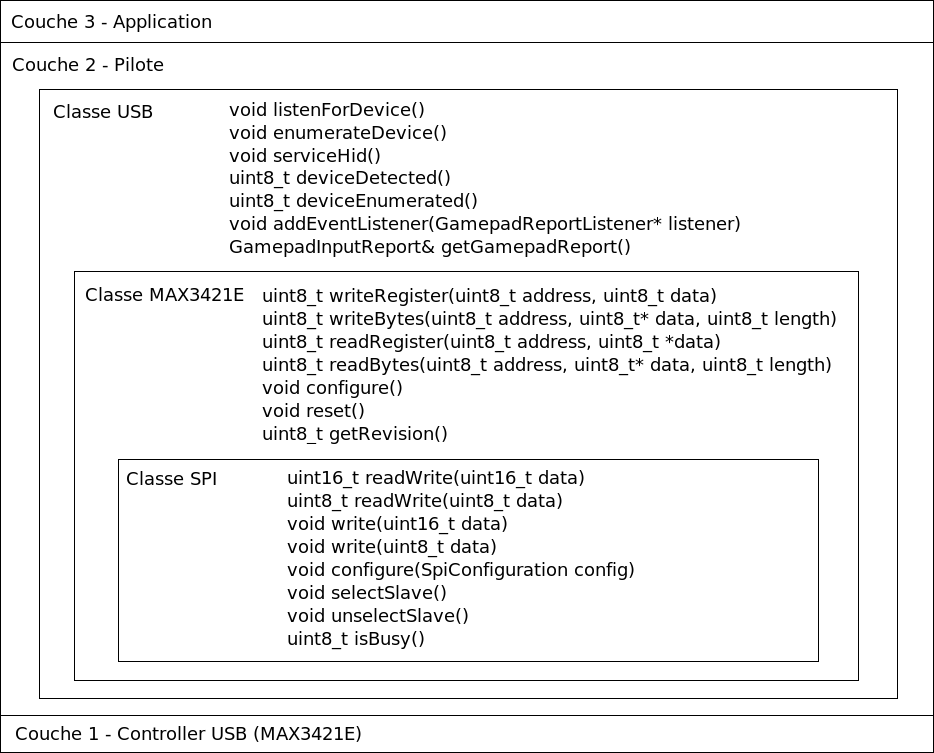
Cependant, avec l’analyseur USB nous ne voyons aucun message de ce type malgré le fait que la bonne LED est sélectionné sur la télécommande Xbox. Un coup d’œil dans le code source du pilote xpad du noyau linux à permit de trouvé que le rapport n’est pas sur 3 octets mais sur 12 et a le format suivant :

|  |  |
| --- | --- |
| **Position** | **Valeur** |
| 0x00 | 0x00 |
| 0x01 | 0x08 |
| 0x02 | 0x40 + (command % 0x0E) |
| 0x03 | 0x00 |
| 0x04 | 0x00 |
| 0x05 | 0x00 |
| 0x06 | 0x00 |
| 0x07 | 0x00 |
| 0x08 | 0x00 |
| 0x09 | 0x00 |
| 0x0A | 0x00 |
| 0x0B | 0x00 |

Dans notre application, seulement le code 0x01 est utilisé car le pilote support seulement une télécommande Xbox. Cependant, l’implémentation du pilote permet l’utilisation de n’importe laquelle de ces commandes.

### Vue d’ensemble du pilote USB

L’image suivante permet d’expliquer et de mettre en contexte le pilote USB qui a été développé.



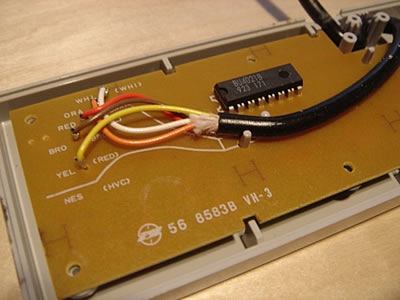
Un seul coup d’œil à ce diagramme permet de repérer les fonctions exposées de chacune des classes. Notez que l’utilisateur se situe à la couche application et ne fait des requêtes qu’à la classe USB.

### Aparté – L’interface avec le Nintendo

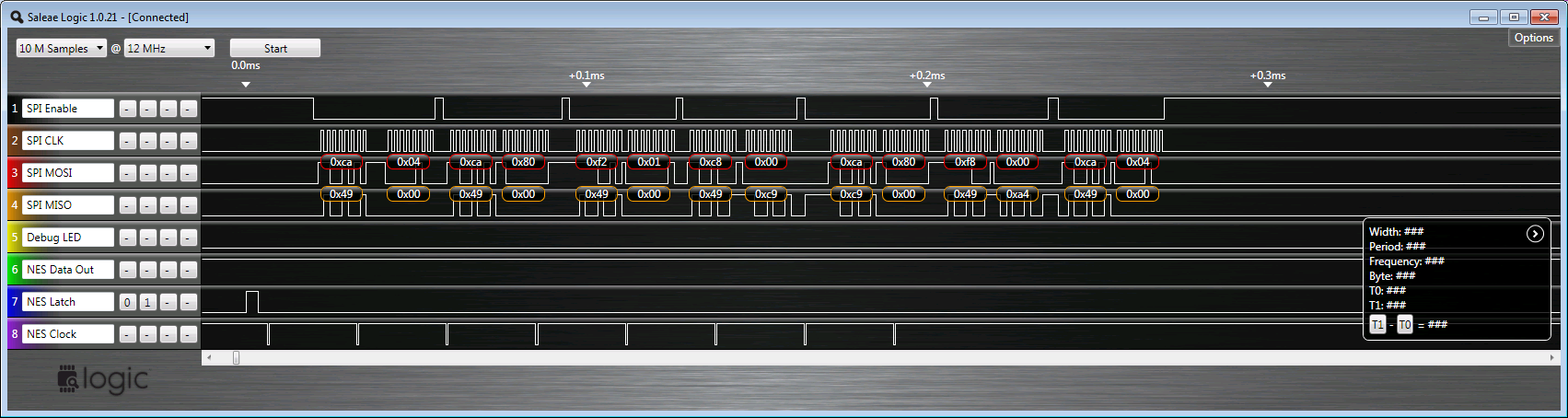
Afin de faire une bonne démonstration du processus de réception des commandes, une interface avec le Nintendo fût développée. Cette interaction est intéressante car ceci démontre la possibilité des faire le traitement des commandes en temps réel. De plus, une fois les commandes reçues et décodées, nous arrivons à la frontière entre le travail des deux participants de l’équipe. Cette interface pourra donc servir d’exemple lors de l’intégration des deux sections.

Le principe de fonctionnement de cette interface est assez simple. En bref, le microcontrôleur fait l’émulation d’une manette de Nintendo tout en recevant les commandes de la télécommande Xbox.

Pour bien comprendre l’implémentation de l’émulation, il faut jeter un œil à l’intérieur d’une manette de Nintendo. Celle-ci est composée d’un simple convertisseur série-parallèle. Chaque bouton sont individuellement relié à une entré du convertisseur. De plus, ces boutons sont actif bas. En d’autres termes, lorsque nous appuyons sur un bouton, le signal de sortie est 0.



Pour communiquer avec la manette, le Nintendo utilise 3 signaux soit le Latch, l’horloge et le signal de sortie pour l’état des boutons. Le Latch permet d’indiqué à la télécommande qu’un transfère va bientôt commencer et que celle-ci doit entrer en mémoire les boutons présentement enfoncés. Ensuite le Nintendo envoie 8 impulsions sur le signal d’horloge et à chaque impulsion, la manette doit lui envoyé l’état d’un bouton sur la ligne de sortie. L’ordre des de l’envoie des boutons est prédéterminé et est le suivant : A, B, SELECT, START, UP, DOWN, LEFT, RIGHT.

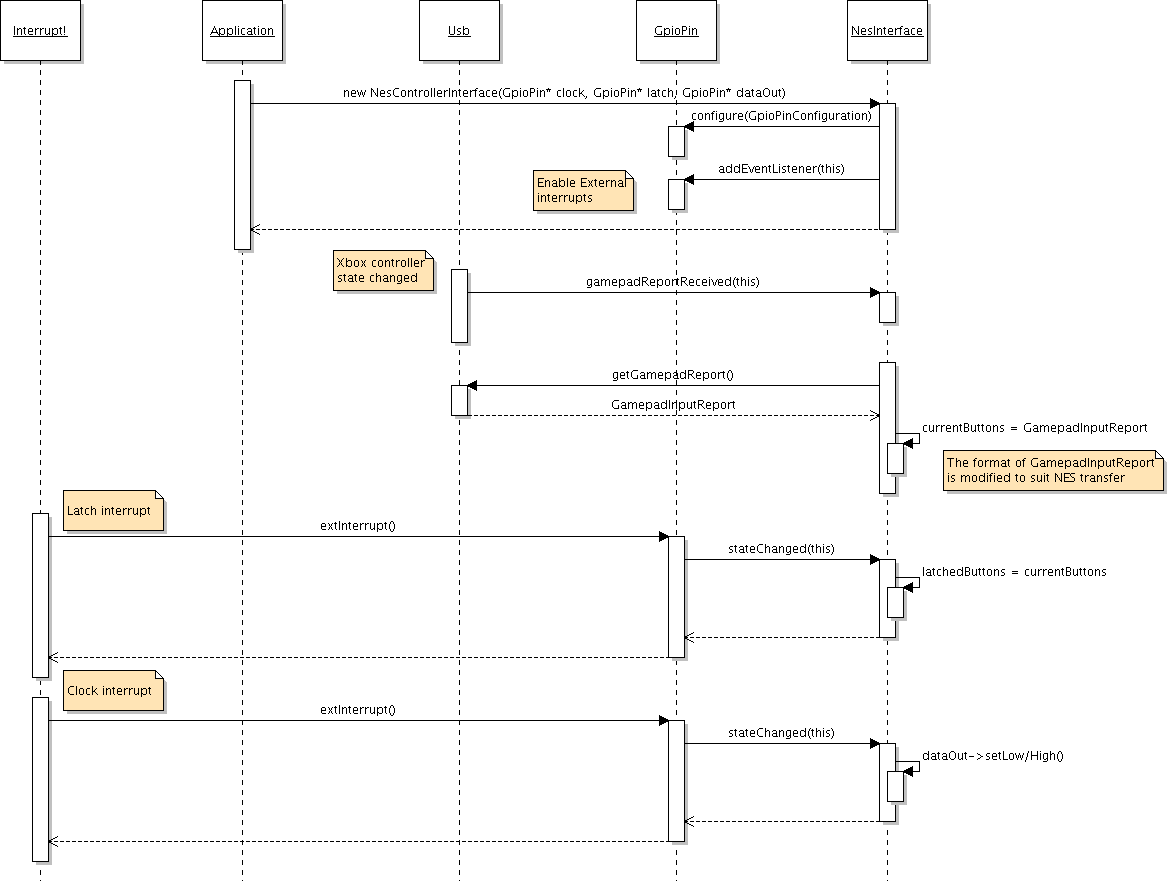


Comme vous pouvez le constaté, l’horloge est à l’était haut par défaut. Il est important de noté que selon plusieurs source sur internet, celle-ci est à l’état bas par défaut. Ceci est assez bizarre mais laisse croire que le Nintendo à subie quelques révisions durant son existance. De toute façon ceci ne change pas l’implémentation de l’interface.

Une autre chose importante à remarqué c’est quand bien que le Nintendo soit assez vieux, la communication entre la manette et celui-ci est très rapide. Le signal Latch à une largeur de 3.3 microsecondes et les impulsions d’horloge 0.5 microsecondes. D’autre part, la fréquence de l’horloge est de 38 kHz ce qui est assez facile à suivre. Le Nintendo demande l’état de la manette 60 fois par secondes.

Comme la télécommande de Xbox doit être vérifié 1000 fois par secondes et que le Nintendo demande l’état de la manette seulement 60 fois par seconde, nous sommes assuré d’envoyé des donnés fraiche au Nintendo. Cependant il est impossible de faire la synchronisation. Ceci ne cause pas de problème car il est très facile de faire les deux choses en même temps à l’aide d’interruptions.

Nous utilisons donc 2 interruptions externes. Une pour le Latch et l’autre pour l’horloge. Le principe est fort simple. À toutes les fois que la télécommande Xbox change d’état, elle envoie cette information au microcontrôleur par le biais du pilote USB. Le pilote décode ensuite ce rapport et l’envoie à l’interface du Nintendo qui s’est enregistré préalablement. L’interface conserve cette information sous un format pouvant être envoyé facile au Nintendo. D’un autre coté, lorsque l’interface reçoit une interruption venant du Latch, elle prend la dernière information reçue et en fait une copie pour le prochain transfère. Lors d’une interruption venant du signal de l’horloge, l’interface met l’état d’un bouton sur la ligne de sortie et prépare l’état du prochain bouton. Le diagramme de séquence suivant met en perspective cette communication.

Cette implémentation fût testée sur une vraie plateforme Nintendo et permet un contrôle aussi fluide qu’avec la vraie manette d’origine.

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

# <titre>

## <Titre>

<Texte>

CONCLUSION

<Texte interligne 1 1/2>

RECOMMANDATIONS <S’il y a lieu>

<Texte interligne 1 1/2>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

1. <titre>

<Texte>

APPENDICES <S’il y a lieu>

<Texte>

LISTE DE RÉFÉRENCES

USB Complete Fourth Edition

http://free60.org/wiki/GamePad

xboxdrv-linux - http://pingus.seul.org/~grumbel/xboxdrv/

Wireshark - http://www.wireshark.org/

Xpad

BIBLIOGRAPHIE

<Texte – Style Références bibliographiques déjà activé>