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

<titre DU DOCUMENT>

par

gervais, Françcois

ST-onge, Rémi

présenté à

Jean-françois boland

montréal, le <date>

**REMERCIEMENTS <S’il y a lieu>**

<Texte interligne et demi>

**<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**

USB – Universal Serial bus

XBOX – Marque déposé de Microsoft pour sa console de jeux vidéo

DE2 – Carte de développement produite par Terasic

HID – Humain Interface Device

SPI – Serial Peripheral Interface

PID – Packet Identification

CRC – Cyclic Redondancy Check

**LISTE DES SYMBOLES ET UNITÉS DE MESURE**

<Texte interligne simple>

INTRODUCTION

<Texte interligne 1 1/2>

# Analyse de la problématique

## Division en sous-problèmes

<Texte>

## Étude de la complexité

<Texte>

# Choix d’une solution

## Jet d’idées

<Texte>

## Définition des concepts

<Texte>

# 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.
* 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 potentionellement être utilisé.

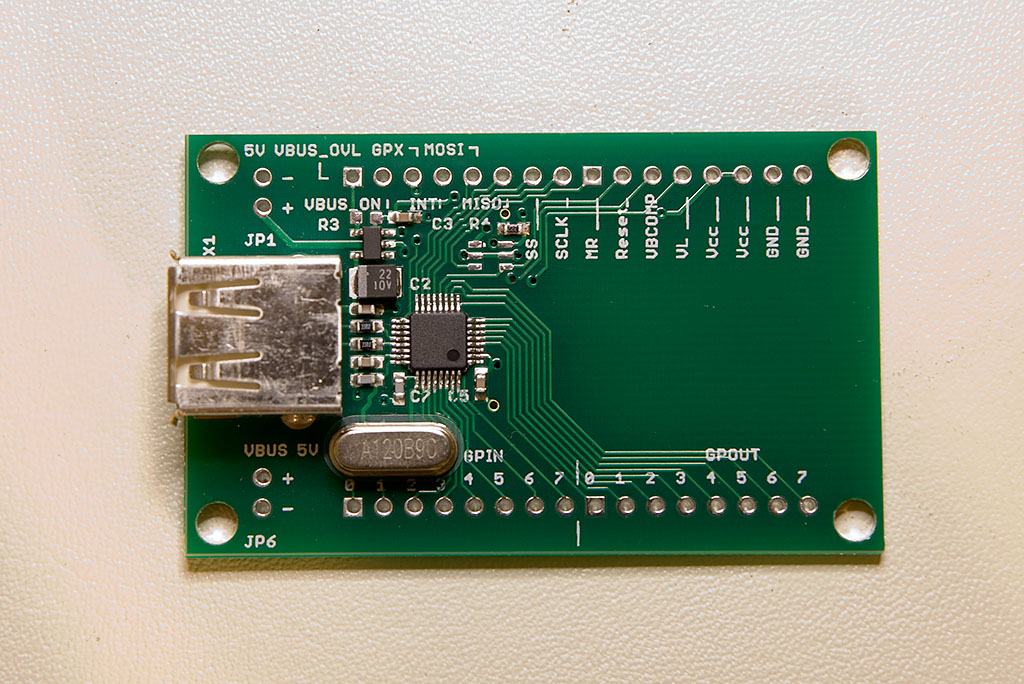
# Réalisation

## 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

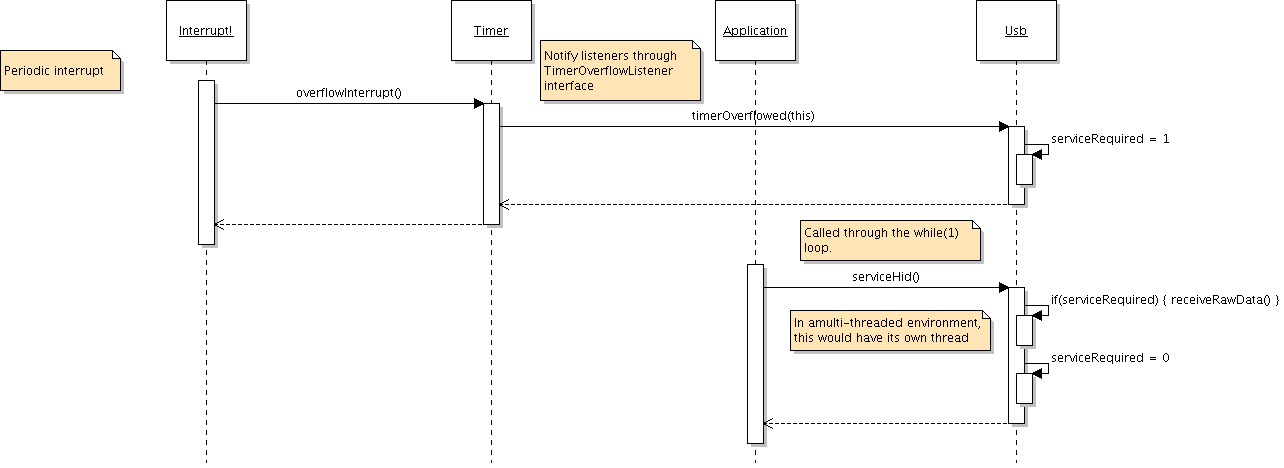
### Énumération du périphérique

### Requêtes HID

Dans cette section nous allons étudier plus en détail ce qui se passe lorsque nous interrogeons le périphérique USB.

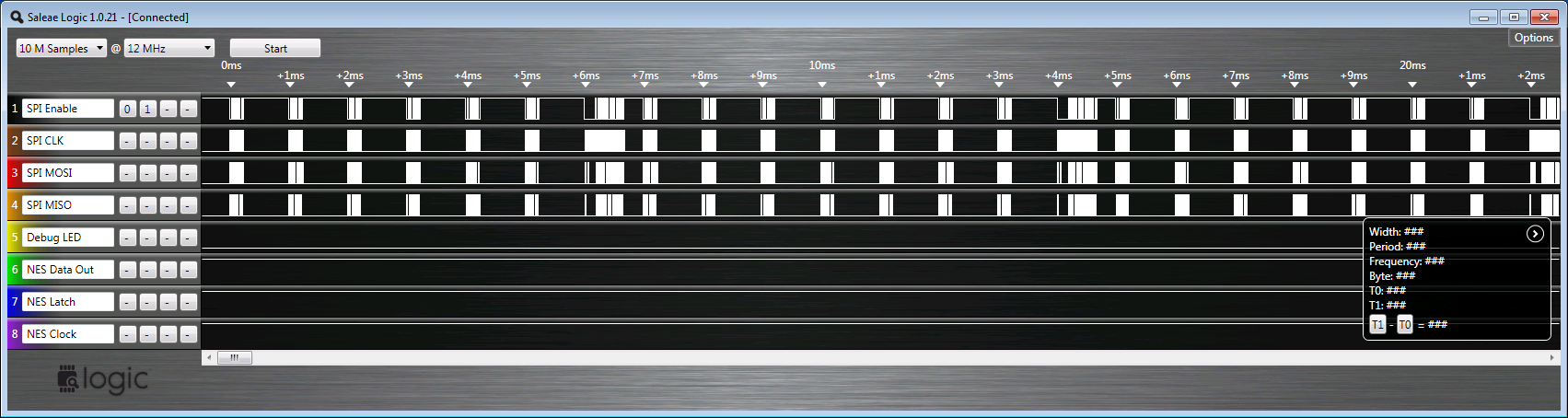
### Temps entre les requêtes HID

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 ses communications mais ceci est une facette que nous traiterons dans les prochaines sections.

### Temps alloué à la communication

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 sur 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 la communication la plus longue dure 0.9065 millisecondes. Voyons si tout ceci fait a du sens. Cette communication est composée de 56 octets. La fréquence d’opération du SPI pour cet exemple est la fréquence d’opération des périphériques 2 divisée par 128. Dans notre cas, cette fréquence d’opération 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.

Pourquoi les résultats pratique et théorique ne concorde pas? Nous avons oubliez de tenir compte du temps mort entre les transactions. En effet, lors d’une transaction, l’esclave de la communication est sélectionné et nous opérons à pleine vitesse. Par contre, entre les transactions, nous devons désélectionner l’esclave, préparé la seconde transaction et resélectionner l’esclave ce qui prend en moyenne 6 microsecondes. Ce délai est invariable en fonction de la fréquence du SPI et ce facteur nous empêche de faire une relation directe entre la fréquence du SPI et le temps passé en communication.

Il est tout de même possible de faire une étude pratique du phénomène. Il est important de comprendre que la fréquence d’opération du SPI dépend du diviseur choisi. La table suivante fait le parallèle entre le diviseur et la fréquence d’opération toujours en considérant une fréquence de périphérique de 72 MHz ce qui est notre cas ici.

|  |  |
| --- | --- |
| Diviseur | Fréquence d’opération (MHz) |
| 2 | 36 |
| 4 | 18 |
| 8 | 9 |
| 16 | 4.5 |
| 32 | 2.25 |
| 64 | 1.125 |
| 128 | 0.5625 |
| 256 | 0.28125 |

Avec notre montage du prototype actuel, nous ne pouvons pas nous permettre de monter trop haut en fréquence. En fait la fréquence maximale que nous pouvons atteindre est 1.125 Mhz. Ceci nous permet de faire une étude avec trois mesures. Il nous sera tout de même possible de faire une extrapolation mais celle-ci sera probablement imprécise pour les hautes fréquences. Il est à noté également que les communications dépendent d’un autre facteur soit le temps de réponse du périphérique. Pour chaque requête, il faut attendre que le périphérique ait répondu et il faut transmettre 2 octets au contrôleur USB pour savoir si la transaction entre le périphérique et le contrôleur est terminé.

|  |  |
| --- | --- |
| Fréquence d’opération (Mhz) | Temps de la plus longue transaction (ms) |
| 0.28125 | 1.412 |
| 0.5625 | 0.907 |
| 1.125 | 0.467 |

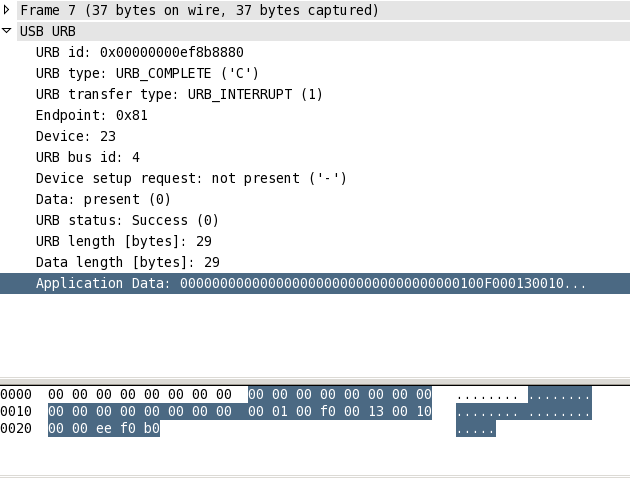
### Décodage des rapports HID

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.

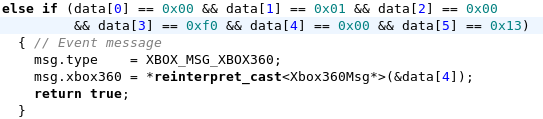
Afin de bien définir le format du rapport, j’ai utilisé le site officiel de l’ingénierie inverse du XBOX 360, free60.org. Celui-ci ne dispose d’information que pour la télécommande avec fil mais ceci est néanmoins en très bon point de départ. Ce site définit le rapport comme suit :

|  |  |  |  |
| --- | --- | --- | --- |
| **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 :

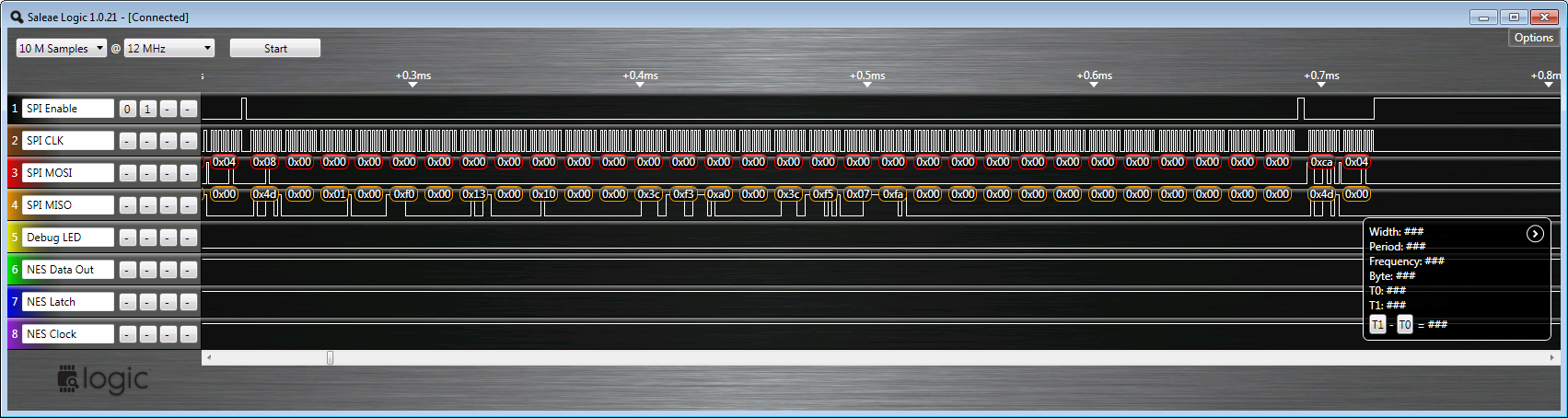


Lors que nous enfonçons le bouton A de la télécommande, nous recevons le message de la capture précédente. Il et important de mentionner que le rapport commence à l’adresse 0x18 sur les captures. Ce rapport en particulier commence donc par les octets 0x00, 0x01, 0x00, 0xF0, 0x00, 0x13. Selon le tableau 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

text

### Aparté – L’interface avec le Nintendo

text

# <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

http://free60.org/wiki/GamePad

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

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

d

BIBLIOGRAPHIE

<Texte – Style Références bibliographiques déjà activé>