Rapport projet synthétiseur

Table des matières

[Introduction 2](#_Toc1)

[Scénario d’usage du synthétiseur 2](#_Toc2)

[Bilan d’entrée/sortie 3](#_Toc3)

[Bilan des périphériques 4](#_Toc4)

[Bilan de la mémoire 4](#_Toc5)

[Génération des notes de musiques 5](#_Toc6)

[Liste des tâches de l’OS avec priorité 7](#_Toc7)

[Schémas bloc des périphériques 9](#_Toc8)

[Schémas électriques 10](#_Toc9)

[État d’avancement du projet 12](#_Toc10)

[Captures d’oscilloscopes (SPI) 13](#_Toc11)

[Capture d’oscilloscope (Bouton Poussoir) 14](#_Toc12)

[Capture d’oscilloscope (Encodeur incrémental) 15](#_Toc13)

[Capture d’oscilloscope (Potentiomètre) 16](#_Toc14)

Jérémy COLZY

Mathieu GRESSET

Mamadou BARRY BOY

Léo DELAS

## Introduction

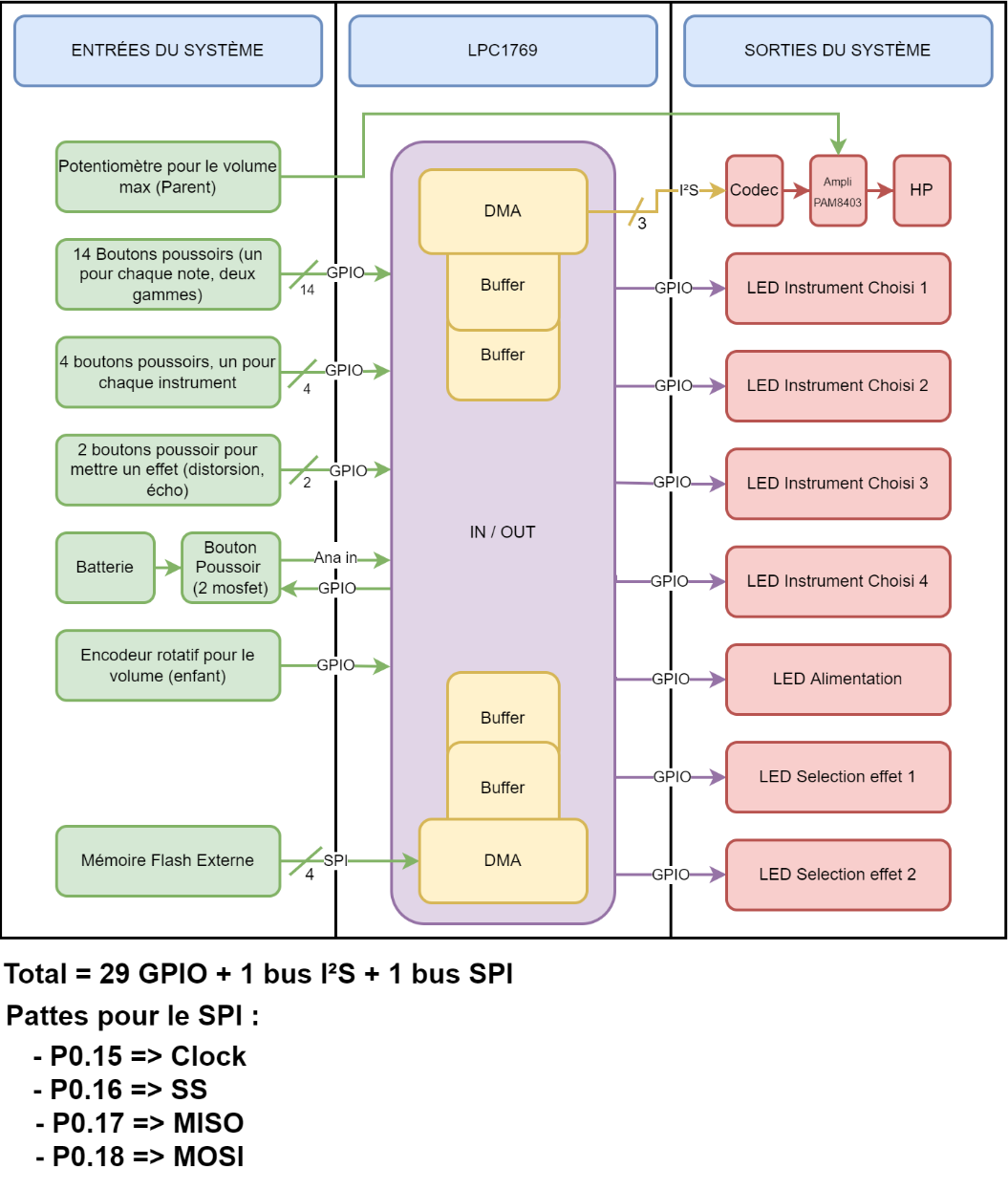
Le but de ce document est de détailler les solutions choisies concernant la configuration du LPC1769 ainsi que les périphériques extérieurs pour la réalisation du synthétiseur, à savoir le scénario d’usage, le bilan d’entrée/sortie, le bilan de la mémoire, la liste des tâches de l'OS avec priorité, schéma bloc des périphériques, les schémas électriques, une liste des interruptions, et une liste d’avancement du projet.

## Scénario d’usage du synthétiseur

* L’adulte ouvre le boîtier d’alimentation afin d’accéder au bouton permettant de régler le volume maximum de l’appareil puis referme le boîtier d’alimentation.
* L’enfant voit le synthétiseur, il essaie de jouer un son en appuyant sur une touche du synthétiseur, mais cela ne fonctionne pas, il appui donc sur le bouton ON/OFF.
* Au bout de quelques millisecondes, le produit est allumé et fonctionnel, prêt à être utilisé
* L’enfant aperçoit 4 LEDs avec une image d’instrument et un bouton associer à chaque LEDs, par défaut la LED du piano est allumée, en appuyant sur un bouton pour changer l’instrument, la LED correspondante s’allume et celle déjà allumée s’éteint.
* L’enfant appuie donc sur un bouton pour jouer un son, une fois appuyer le son correspondant est joué.
* Dès que la touche est relâchée, le son disparait.
* Lorsque l’enfant appuie sur plusieurs touches à la fois, les sons s’additionnes. Au maximum, 4 sons peuvent être joué simultanément. Si une touche supplémentaire est appuyée, le son de la première touche appuyé disparait.
* Il peut aussi ajouter des effets de sons tel qu’un écho ou de la distorsion.
* Le volume maximum du synthétiseur peut être défini par un adulte à l’aide d’un potentiomètre protégé par un cache pour pas que les enfants y touche.
* Les enfants ont la possibilité de régler le volume du synthétiseur via un encodeur rotatif. Le volume varie entre 0 dB et le volume défini par les parents par pas de 3 dB.

## Bilan d’entrée/sortie

Voici ci-dessous un schéma de notre bilan d’entrée/sortie pour le projet du synthétiseur pour enfant.



Notre système comporte 23 entrées dont une non reliée au microcontrôleur (Potentiomètre pour le volume max) qui est directement relié à l’amplificateur pour régler le gain de l’amplificateur. Le système comporte également 7 LED de sortie afin de communiquer des informations sur l’état du système.

Les données de la mémoire flash sont chargé dans des buffers de la mémoire interne de la carte LPC1769 à l’aide d’un DMA qui est un périphérique interne à la carte.

Les échantillons audio traités sont stocké dans les buffers de la carte LPC1769 qui sont ensuite envoyé au codec audio.

Une fois le signal reçu sur le codec, le signal est traduit et envoyé à l’amplificateur qui adapte le niveau de sons avant que le haut-parleur ne joue le son.

## Bilan des périphériques

Les périphériques que nous allons utiliser pour ce projet sont les suivants :

* 2 DMA
  + 1 pour récupérer les données de la mémoire flash sur la carte LPC1769
  + 1 pour envoyer les données de la carte LPC1769 au circuit qui permet de jouer le son
* 1 Timer
  + 1 Timer pour gérer le système de veille automatique.

## Bilan de la mémoire

Après avoir testé plusieurs méthodes d’échantillonnage/stockages nous avons décidé de stocker les sons de tous les instruments pour chaque note dans la mémoire. Pour les sons non-évanescent nous avons choisie de ne stocker dans la mémoire qu’une seul période du son à jouer et de re-boucler cette échantillons pendant 5 secondes. Pour les sons évanescent nous avons choisie de stocker la note entière. Le seul traitement du signal qui sera à opérer concernera les effets appliqués aux notes.

Monsieur Bras nous a fourni une mémoire de 16 Mo. Suite à nos essais nous avons déterminé que pour stocker toutes les 14 notes (soit 2 octaves) pour les sons évanescents il nous fallait 6 Mo d’espace mémoire et pour les sons non-évanescent il nous fallait 10 Ko d’espace mémoire.

Nous avons, sur la carte, une SRAM de 16 ou 32 Ko. Le but pour les buffers est de ne pas dépasser la moitié de la capacité de la SRAM. Le système doit pouvoir jouer 4 notes en simultanés ce qui veut dire que nous avons besoins de 4 \* 2 buffers pour récupérer les données de la mémoire vers le LPC. Nous avons également 2 buffers pour envoyer le son sur les haut parleurs.

On utilise donc au total 10 buffers de 256 octets soit un total de 2560 octets. Nous pensons que la tailles des buffers est à ajuster en pratique. Mais n’ayant pas finie le projet actuellement nous ne pouvons pas bien régler la taille des buffers.

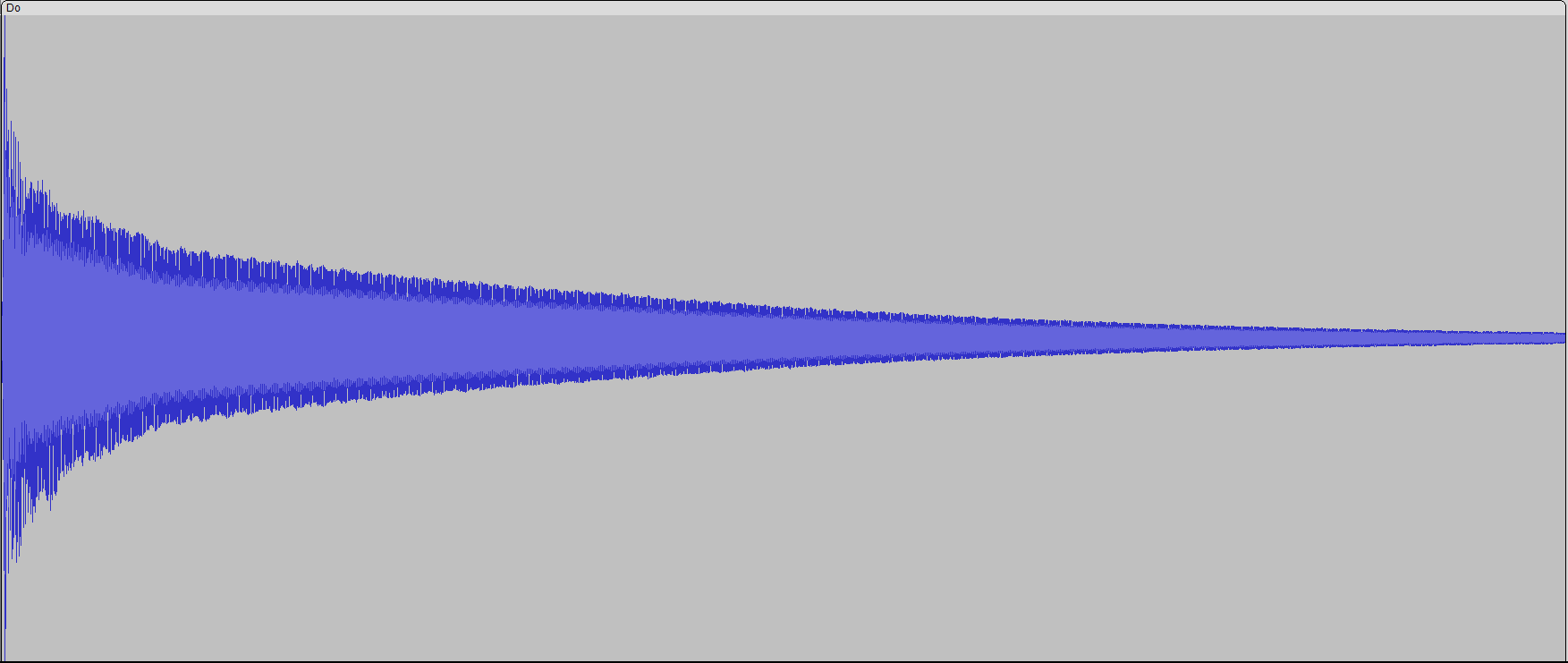
## Génération des notes de musiques

Afin de générer les sons depuis le synthétiseur, nous avons choisi d’enregistrer les différentes notes des 4 instruments choisis afin de les stockés dans la mémoire. Pour rappel, les instruments choisis sont :

* Le piano
* La flute de pan
* Le carillon
* Le saxophone

Ces différents instruments sont classé en deux catégories, les instruments évanescents (piano et carillon) et les instruments non évanescent (flute et saxophone). Pour chacun, nous avons enregistrer les notes des octaves 3 et 4. Nous n’avons pas voulu enregistrer les notes modifiées par des altérations (dièse ou bémol). Cela représente 14 notes en tout.

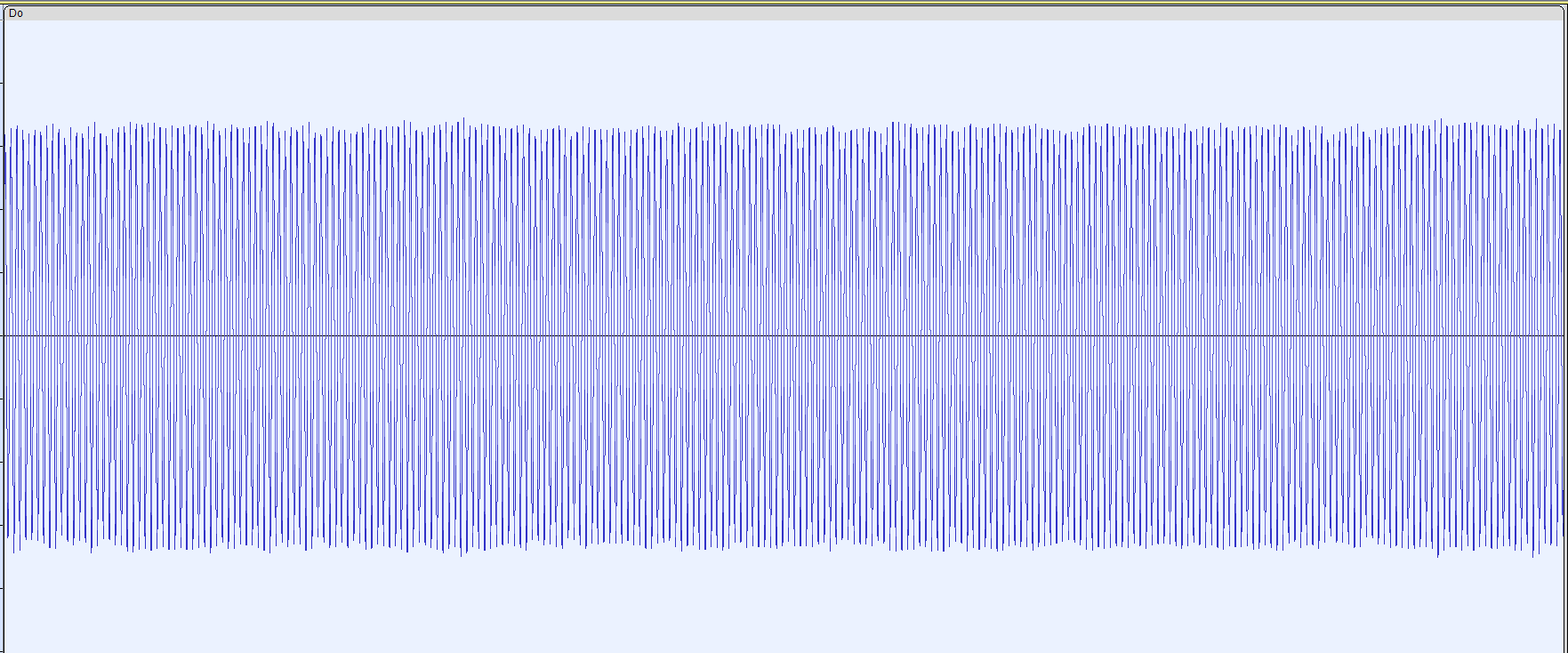
Pour les instruments évanescent, nous avons choisis d’enregistrer le son de chaque note entre le temps d’appuie sur une touche et le temps où le son disparait.



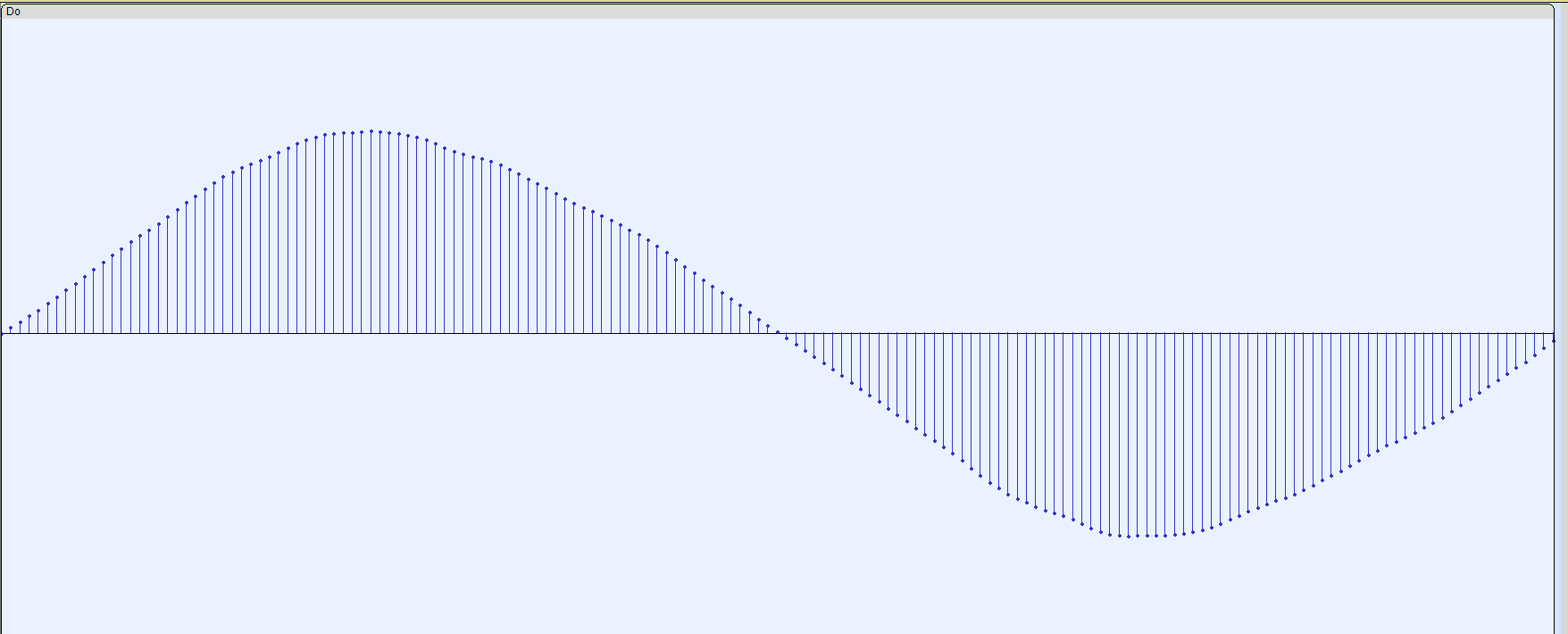
Chronogramme de la note Do (octave 3) du piano.

Un tel son sera alors stocké ainsi dans la mémoire.

Pour les instruments non évanescents, nous avons enregistré deux types de son, une note pendant une durée de 1 seconde environ et une seule période de son. Ces deux enregistrements peuvent être utile selon la taille de la mémoire. Si la mémoire est trop petite, nous stockerons seulement une période de son qui sera répété tant que la touche est maintenue appuyée. Cela demande une grande quantité de ressource au processeur. Si la mémoire est suffisamment grande, nous stockerons le son d’une durée de 1 seconde et celui-ci sera lu en boucle tant que la touche est appuyée. Ce système permet d’économiser des ressources processeur mais nécessite plus de place mémoire.



Chronogramme d’une note Do (octave 3) de flute de pan sur une durée de 1s.



Chronogramme d’une période de note Do (octave 3) de flute de pan.

## Liste des tâches de l’OS avec priorité

Variables globales :

- "unsigned uint32\_t encodeur" contient la valeur de l'encodeur

- "unsigned uint32\_t potentiometre" contient la valeur du potentiomètre

- "unsigned uint32\_t bt\_distortion" contient la valeur du bouton de distortion

- "unsigned uint32\_t bt\_echo" contient la valeur du bouton de echo

- "unsigned uint32\_t bt\_flute" contient la valeur du bouton de la flute

- "unsigned uint32\_t bt\_saxophone" contient la valeur du bouton du saxophone

- "unsigned uint32\_t bt\_carrilon" contient la valeur du bouton du carrilon

- "unsigned uint32\_t bt\_piano" contient la valeur du bouton du piano

tableaux en variables globale :

- "uint8\_t buffer\_SPI\_1\_1[256]" buffer 1 de la liaison SPI associé au lecteur 1

- "uint8\_t buffer\_SPI\_1\_2[256]" buffer 2 de la liaison SPI associé au lecteur 1

- "uint8\_t buffer\_SPI\_2\_1[256]" buffer 1 de la liaison SPI associé au lecteur 2

- "uint8\_t buffer\_SPI\_2\_2[256]" buffer 2 de la liaison SPI associé au lecteur 2

- "uint8\_t buffer\_SPI\_3\_1[256]" buffer 1 de la liaison SPI associé au lecteur 3

- "uint8\_t buffer\_SPI\_3\_2[256]" buffer 2 de la liaison SPI associé au lecteur 3

- "uint8\_t buffer\_SPI\_4\_1[256]" buffer 1 de la liaison SPI associé au lecteur 4

- "uint8\_t buffer\_SPI\_4\_2[256]" buffer 2 de la liaison SPI associé au lecteur 4

- "uint8\_t buffer\_I2S\_1\_1[256]" buffer 1 de la liaison I2S

- "uint8\_t buffer\_I2S\_1\_2[256]" buffer 2 de la liaison I2S

- "char select\_player[4]" tableau permettant de savoir quel est le plus ancien lecteur qui joue et donc savoir quel lecteur attribuer à quel note

\*\* Les différents lecteurs correspondes à la gestion de la polyphonie. Chaque lecteur correspond à une sortie audio simultané. Il y a donc 4 lecteurs pour 4 notes joué en même temps.

La fonction main de notre code a pour rôle d'initialiser tous les périphériques avant de lancer l'OS.

**Tâche 1 :** "Input\_selector", priorité importante, déclenchement périodique toutes les 100ms

Cette tâches à pour but de scruter toutes les entrées du système (tous les boutons + la valeur du potentiomètre et de l'encodeur) et de mettre à jour la valeurs des variables d'entrée. Cette tâche permet également de choisir un lecteur pour chaque note. Si aucune touches n’est enfoncé au bout de 30 secondes on active la tâche "veille"

**Tâche 2 :** "Chargement\_des\_donnees", priorité normal, déclenchement suite à la sélection d'un lecteur (choix d'une note)

Cette tâche à pour but de charger dans la carte LPC les données stocké dans la mémoire SPI à l'aide d'un DMA. Les données charger prioritairement sont celles du lecteurs le plus ancien. Cette tâches utilise tous les buffers de stockage SPI. On lève un sémaphore à chaque fois qu'un buffer SPI est remplie.

**Tâche 3 :** "traitement\_du\_son", priorité normal, déclenchement suite à un signal déclenché lorsque tous les buffer sont chargées

Cette tâche à pour but de mixer les buffers des différents lecteur (buffer SPI) et d'appliquer les effets d'echo et de distortion puis de charger le buffer lié à l'I2S. On lève un sémaphore à chaque fois qu'un buffer I2S est remplie.

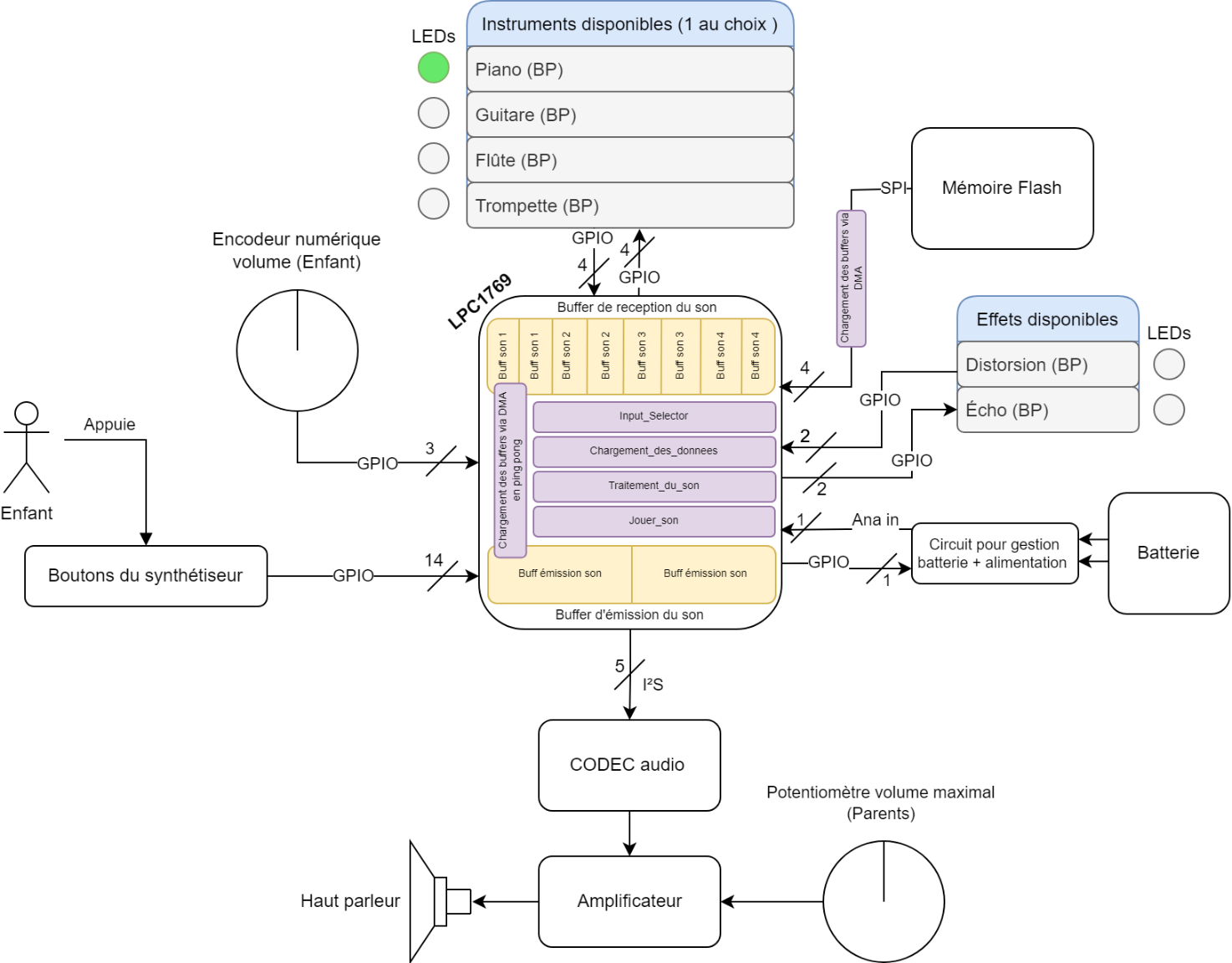
**Tâche 4 :** "jouer\_son", priorité normal, déclenchement suite à un signal déclenché lorsqu'un buffer I2S est remplie

Cette tâche à pour but d'envoyer les données du buffers I2S au circuit qui joue la note (ampli + haut parleur)

**Tâche 5 :** "veille", priorité maximum, déclenchement sur une interruption provenant du Match Register d’un timer

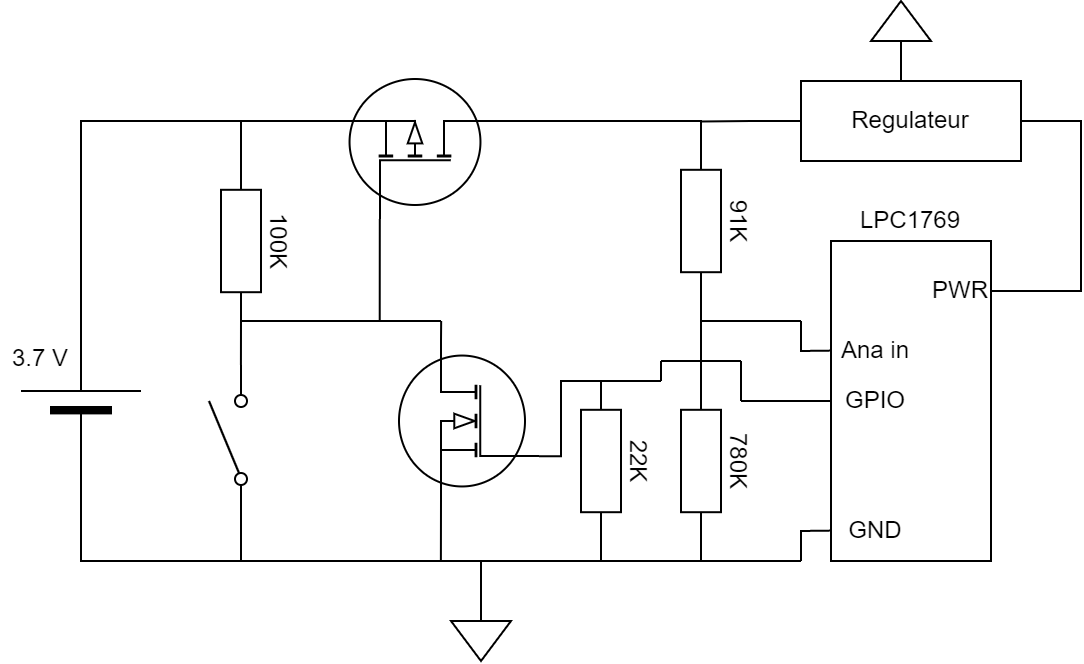
Cette tâche permet de mettre en veille et de réveiller le système.

## Schémas bloc des périphériques



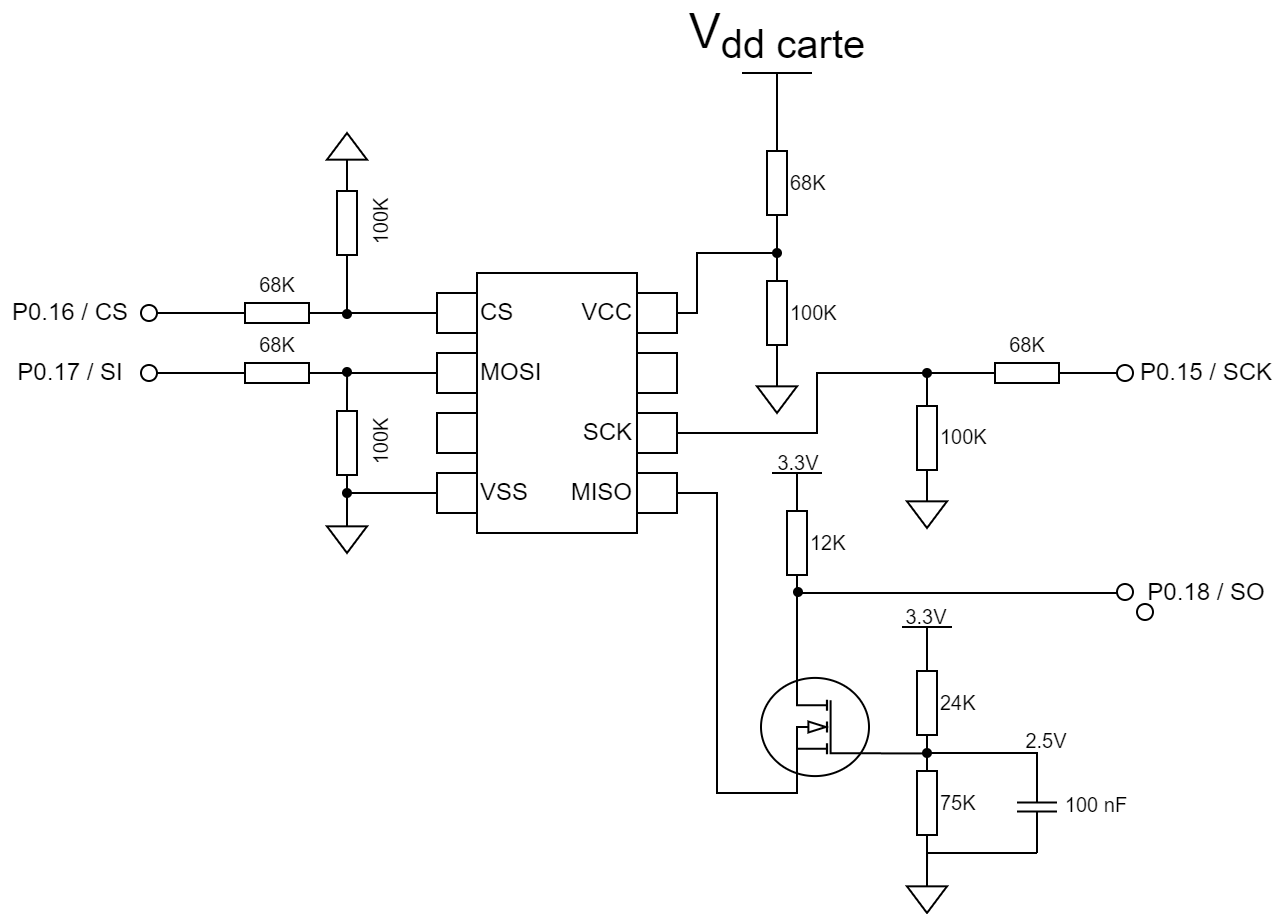
## Schémas électriques

Pour la gestion de l’alimentation nous utiliserons le schéma électrique suivant.



Le but de ce schéma électrique est d’alimenter la carte du LPC1769 lors d’un appuie sur le bouton d’alimentation, puis d’avoir un maintien de l’alimentation sans avoir à rester appuyé sur le bouton. Lorsque la batterie descend en dessous d’un certain seuil, la carte coupe son alimentation en bloquant le transistor de maintien de l’alimentation. La carte LPC1769 implémentera également une veille automatique au bout d’un certain temps.

Pour alimenter et utilisé la mémoire flash nous utiliserons le montage électrique suivant :



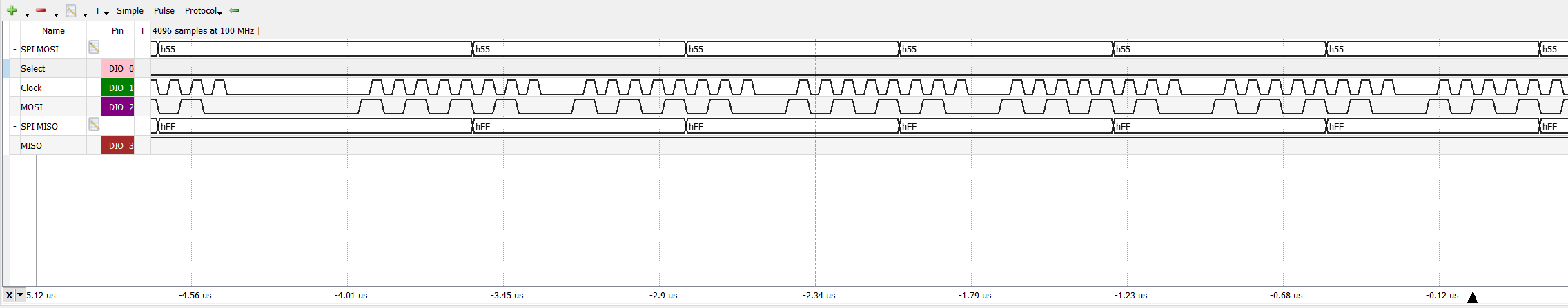
Nous avons besoin de mettre un pont diviseur de tension aux entrées / sorties P0.15 / P0.16 / P0.17 car la mémoire fonctionnant en 2V, nous avons besoin d’abaisser le niveau de tension des signaux de commande / alimentation, pour pouvoir récupérer en sortie un signal compris entre 0V et 3.3V, nous avons recourt au montage ci-dessus.

## État d’avancement du projet

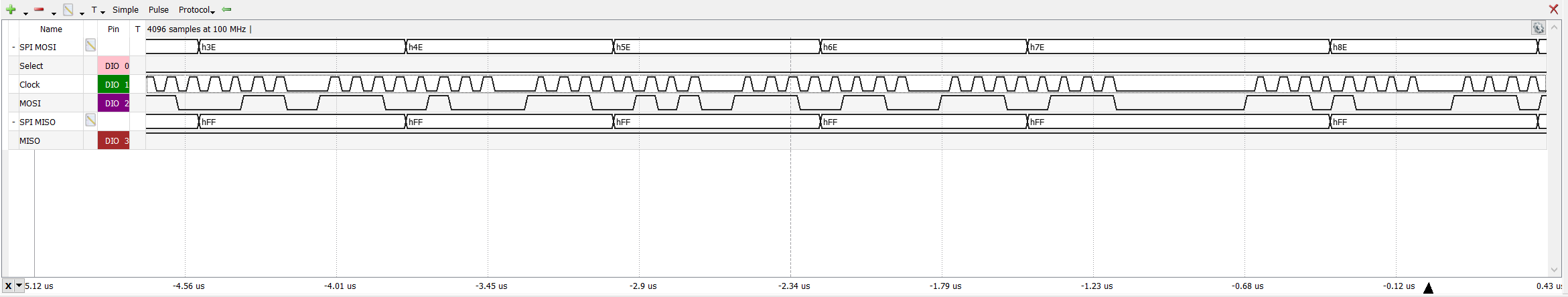
Voici ci-dessous un tableau qui représente notre état d’avancement dans le projet :

|  |  |
| --- | --- |
| **Étapes du projet** | **État d’avancement** |
| Cahier des charges Fonctionnelles | Réalisé et validé |
| Cahier des charges techniques | Réalisé et validé |
| Commande des composants | Réalisé et validé |
| Planifications (temps + technique) | Réalisé et validé |
| Réalisation d’une maquette (hardware + software) permettant de valider le fonctionnement des boutons, potentiomètres, encodeur rotatif et LED | Réalisé et validé |
| Réalisation d’une maquette (hardware + software) permettant de valider la communication SPI entre la mémoire et la carte LPC | 90%  La liaison est fonctionnelle mais nous n’avons pas encore initialisé la mémoire pour que cette dernière nous réponde sur le bus |
| Réalisation d’une maquette (hardware + software) permettant à la carte LPC de jouer un son. | 0%, la maquette prototype n’a pas été faite |
| Assemblage final (software + hardware) | 0% |

## Captures d’oscilloscopes (SPI)



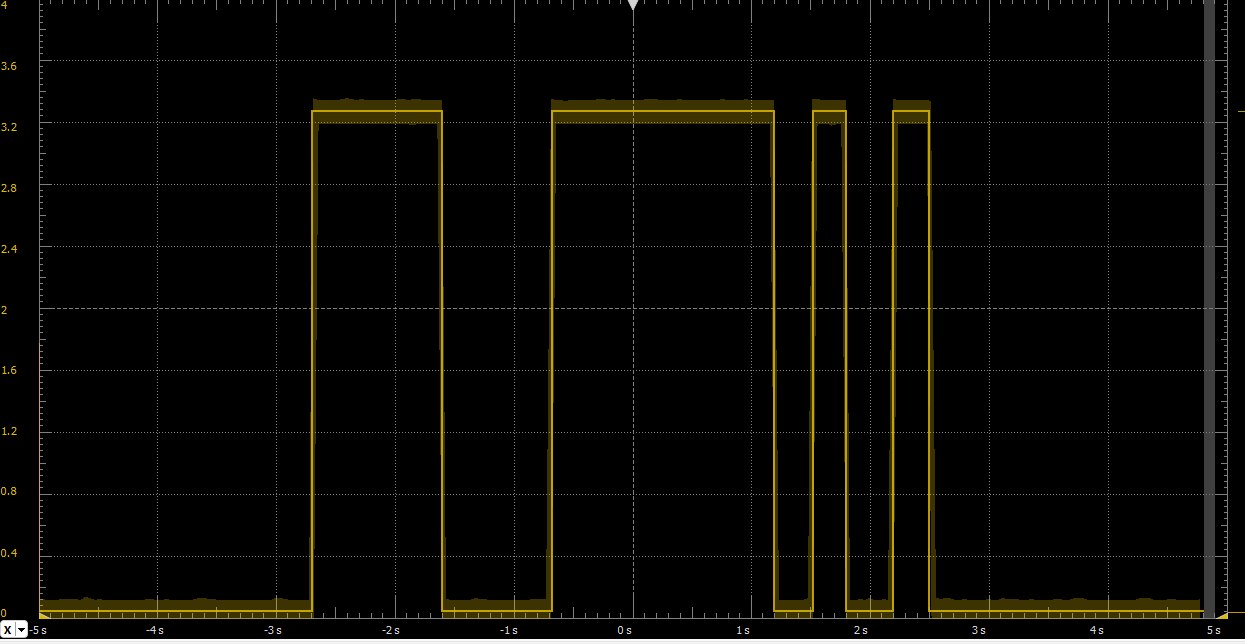
Capture d’oscilloscope d’une transmission spi d’une même valeur



capture d’oscilloscope d’une transmission spi d’un buffer incrémentale

## Capture d’oscilloscope (Bouton Poussoir)

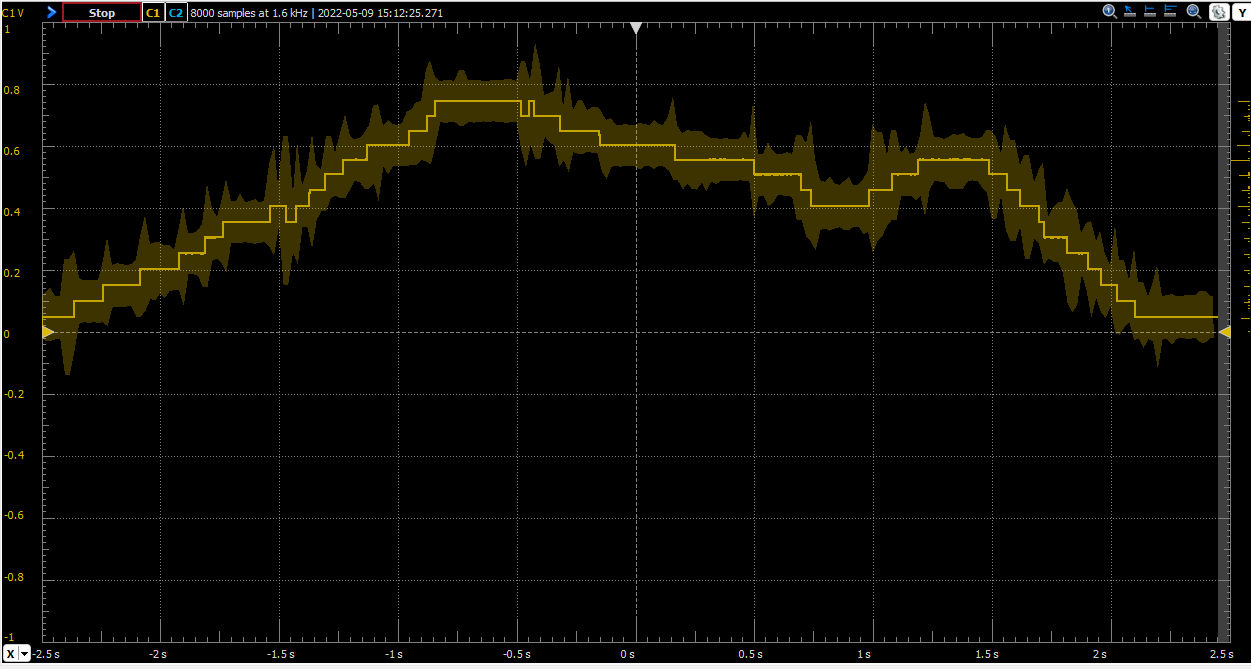
Appui du bouton poussoir. Allumage de la Led rouge de la carte.



## Capture d’oscilloscope (Encodeur incrémental)

Décrémentation du codeur par pas de 0x10

Incrémentation du codeur par pas de 0x10



## Capture d’oscilloscope (Potentiomètre)

Le potentiomètre utilisé n’est pas linéaire d’où la forme logarithmique de la courbe lors des phase de montée ou de descente.

Diminution du potentiomètre

Augmentation du potentiomètre

