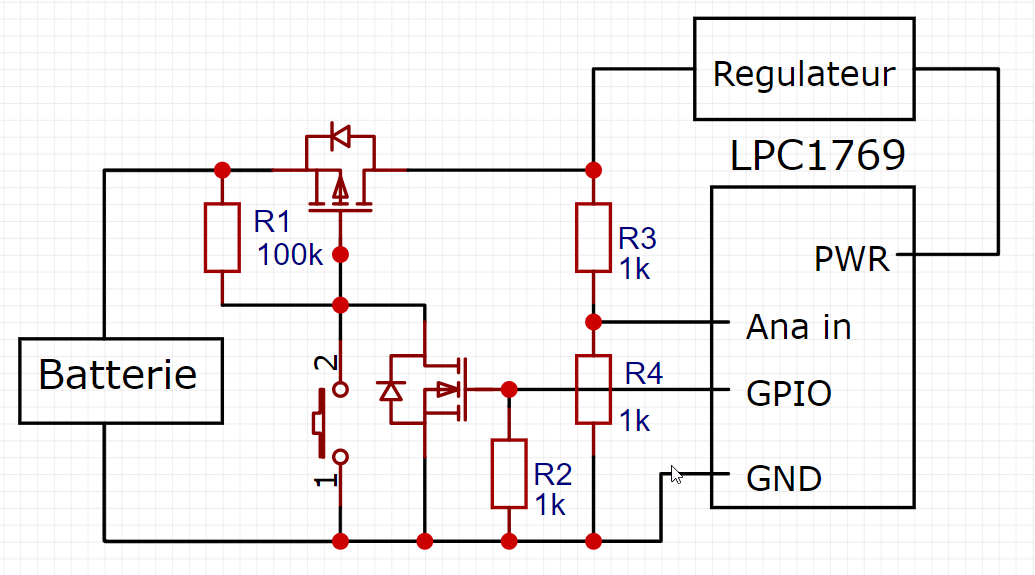
# Configuration du LPC1769

Le but de ce document est de détailler les solutions choisie concernant la configuration du LPC1769, à savoir le bilan d’entrée/sortie, le bilan des périphériques, la configuration des périphérique et le bilan de la mémoire.

## Gestion de l’alimentation

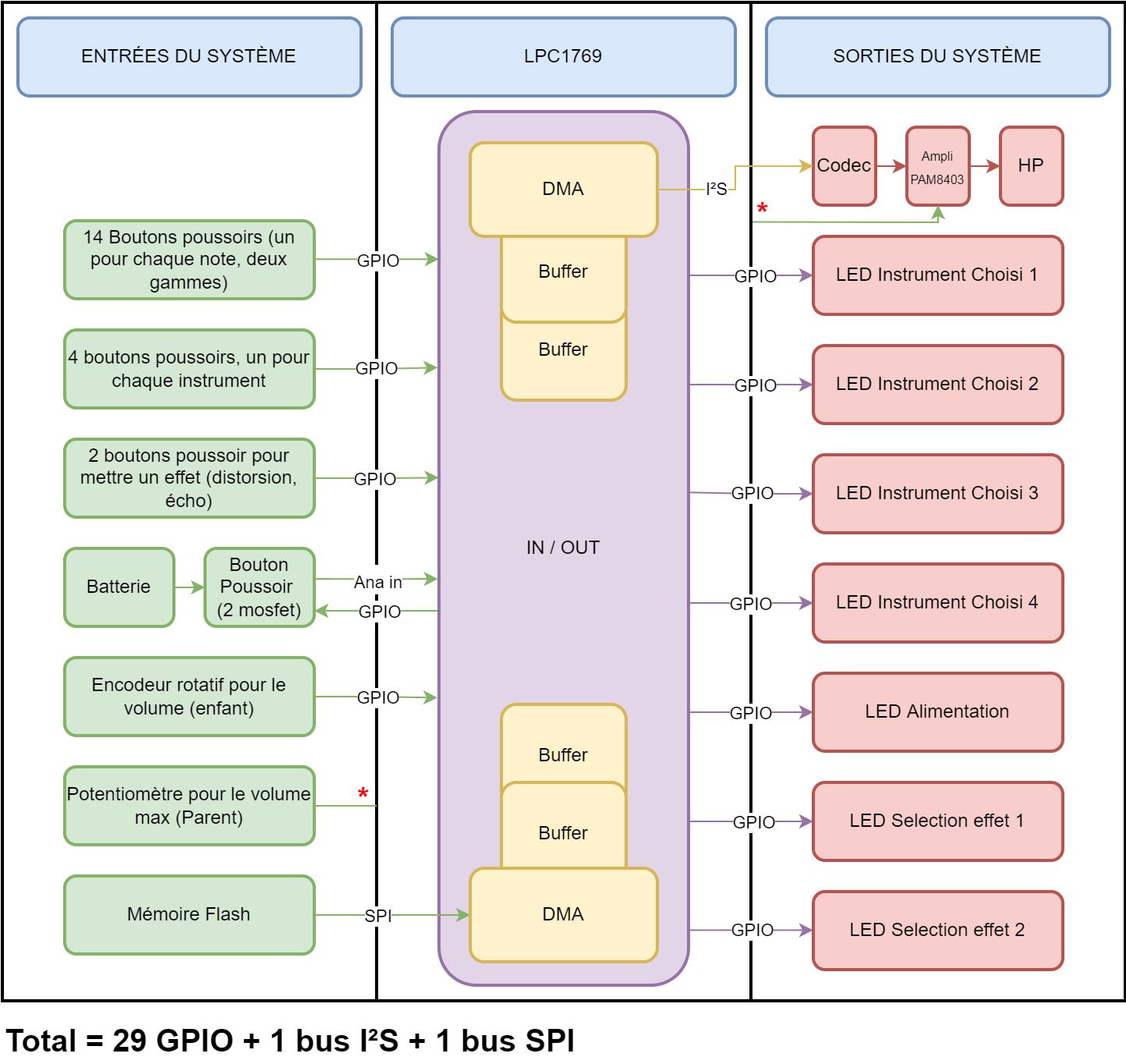
Pour la gestion de l’alimentation nous utiliserons le schéma électrique suivant.



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 maintient 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 maintient de l’alimentation. La carte LPC1769 implémentera également une veille automatique au bout d’un certain temps.

## 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.

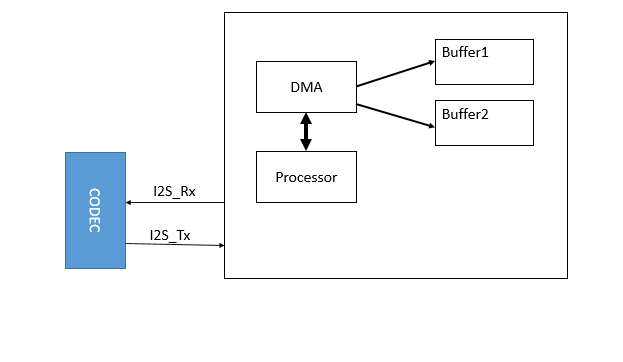
Une fois le signal reçu sur le codec, le signal est traduit et envoyé à l’amplificateur qui adapte le niveau de sons avant que le haut-parleur ne joue le sons.

## 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 envoyé les données de la carte LPC1769 au circuit qui permet de jouer le son
* 2 Timer
  + 1 Timer pour envoyé les échantillons à une fréquence régulière pour respecter le signal.
  + 1 Timer pour gérer le système de veille automatique.
* I2S

|  |  |  |
| --- | --- | --- |
| Nom | Port | Pin |
| I2STX\_CLK | P.7 | 10 |
| I2STX\_WS | P.8 | 8 |
| I2STX\_SDA | P.9 | 10 |
| I2SRX\_CLK | P.23 | 5 |
| I2SRX\_WS | P.24 | 1 |
| I2SRX\_SDA | P.25 | 2 |
|  |  |  |



## 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 en brute dans la mémoire. Le seul traitement du signal qui sera à opérer concernera les effets appliqué au notes.

Monsieur Bras nous a fournit une mémoire de 16 Mo. Suite à nos essais nous avons déterminé que pour stocker toutes les 14 notes (soit 2 octaves) pour chacun des 4 instruments il nous faut 11 Mo 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 moitier de la capacité de la SRAM.

Moitier = 8 Ko,

4 buffers = 2 Ko par buffers