**Avertissement sonore :**

Doc github : <https://github.com/RalphBacon/MP3-Player-DFPlayer/blob/master/DFPlayer%20Mini%20Manual.pdf>

Fonctionne en UART / 9600 bauds (réglable)

On peut mettre tous les sons qu’on veut, il faut les injecter sur la mini carte SD

ATTENTION : Le Tx et Rx de l’arduino sont en 5V tandis que sur le MP3 ils sont en 3.3V, il faut donc faire un pont diviseur de tension comme sur le schéma vu sur le github

(Edit : en fait le DFPlayer peut prendre de 3.2V à 5V)

Choses à faire :

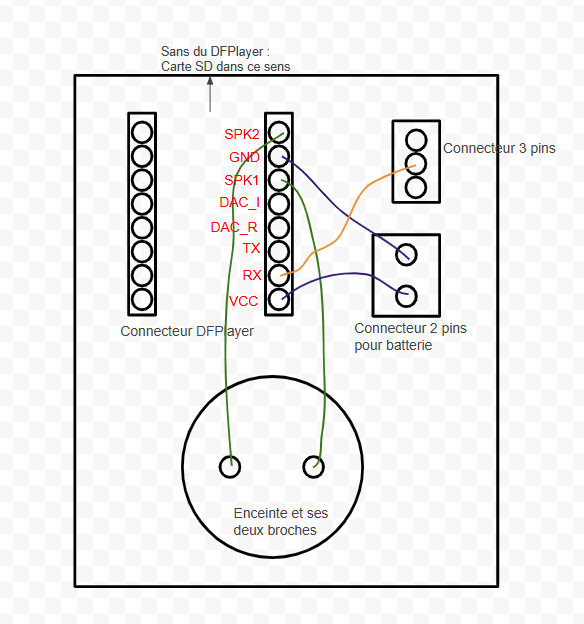
Trouver comment faire du texte sur github

Trouver avec quel microcontroleur on utilisera le DFPlayer : il faut regarder les ports (UART).

Faire mon propre schéma en incluant la carte déjà présente

Séance 2 :

Schéma de la carte électronique du DFPlayer et son enceinte :



(voir gg drawing pour modifier)

Ceci est le matériel donné, on remarque le connecteur 2 pins relié au VCC et au GND, on viendra donc brancher la batterie sur ce connecteur servant d’alimentation. De plus, le connecteur 3 pins est branché au RX, ce qui signifie que c’est le port UART afin d’envoyer les données au DFPlayer. Le speaker est quant à lui déjà bien branché.

Il faut alors :

* Trouver un moyen d’alimenter la carte
* Trouver un moyen de gérer l’envoi de données (microcontrôleur adapté avec un port UART)
* Choisir les sons à jouer

Trouver un moyen d’alimenter la carte :

Voir avec les alimentations des autres composants de la voiture. Nécessaire : 3.3V.

Pour nos tests, on se branche à des pins VCC et GND de la carte qui fournissent du 4.2V.

Trouver un moyen de gérer l’envoi de données (microcontrôleur adapté avec port UART) :

On utilisera la carte KEIL. Nous utiliserons son port 2.8 en TX et 2.9 en RX (UART2 dans le code). Nous reprenons le code “baseUART” de l’IUT. Ainsi, toute la configuration est déjà faite.

Choisir les sons à jouer :

Les sons qui étaient déjà sur la carte sont sur mon cloud, il faut maintenant choisir nos propres sons et les mettre sur la carte SD (Adam a un lecteur de carte SD contrairement aux pc de l’IUT)

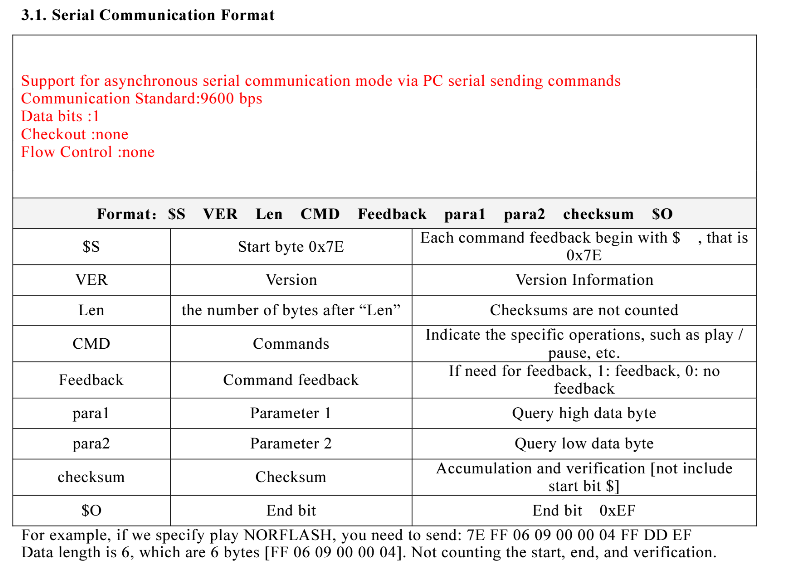
Maintenant que cela est fait, nous avons notre connecteur UART mais ce dernier est en mode asynchrone, il faut donc trouver comment gérer les horloges. De plus, il faut trouver quelles informations envoyer au DFPlayer pour jouer les différents sons. Pour cela, il faut trouver comment le DFPlayer gère les fichiers mp3 sur sa mémoire, parce que par exemple, j’ai vu que il faut renommer les mp3 avec “000x”

Trouver comment gérer les horloges :

Les horloges sont réglées elles-mêmes, le programme KEIL gère l’envoie à 9600 bauds, et le DFPlayer est déjà configuré en 9600 bauds

Quelles informations envoyer au DFPlayer pour jouer les différents sons :

Nommer les dossiers de 01 à 99. Nommer les fichiers de 001 à 255.



Constitution d’une trame donc environ 10 octets:

| 1er octet | 0x7E |
| --- | --- |
| 2e octet | Version : 0xFF |
| 3e octet | Lenght : 0x06 |
| 4e octet | Commande |
| 5e octet | Feedback : 0x01 ou 0x00 |
| 6e octet | Data sur 2 octets, ici c’est le premier octet (high) |
| 7e octet | deuxième octets de la data (low) |
| 8e octet | Checksum premier octet |
| 9e octet | Checksum deuxième octet |
| 10e octet | 0xEF |

On écrit une fonction pour envoyer ce qu’on veut jouer :

|  |
| --- |

On a également créé une fonction pour attendre afin de réaliser des tests :

|  |
| --- |

Puis, dans le main, on initialise l’UART, on initialise le DFPlayer, et on peut enfin jouer des sons ! :

|  |
| --- |

Voir programme KEIL sur la session IUT de Nathan.

Egalement, on a testé l’envoie UART des trames, par exemples, envoie d’une trame d’initialisation :

|  |
| --- |