

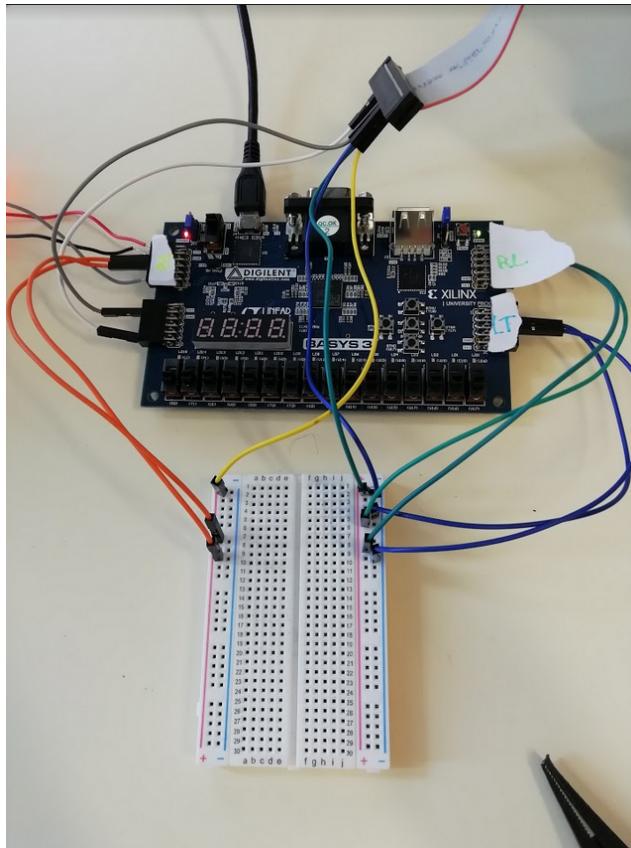


Professeur encadrant :
M. Teboul

Réalisé par :
*VIAL Valentin,
MICHALET Quentin,
1G2TP1*

Année : 2020-2021

Projet d'électronique : Effets lumineux par commande tactile



Sommaire :

- I. Présentation du projet
- II. Implémentation de la neopixel
- III. Intégration de l'écran tactile.
- IV. Affichage des couleurs sur la neopixel suivant l'axe X
- V. Mode Y
- VI. Pour aller plus loin
- VII. Annexes

I. Présentation du projet

1. Objectif(s)

L'objectif de ce projet est de commander la couleur d'une ou plusieurs leds en fonction de la position d'un doigt sur une dalle tactile.

2. Composants utilisés

Nous avons à disposition un bandeau de plusieurs leds neopixels ainsi qu'un boîtier tactile comprenant une surface tactile de type résistif, un afficheur et un CAN. Nous disposons aussi d'une carte basys 3.



II. Implémentation de la neopixel



1. Branchements

La neopixel se distingue des LEDs classiques par l'intégration d'une “puce pilote” interne à chaque cellule lumineuse. Ceci simplifie le travail du microcontrôleur car le changement de couleur est commandé par un seul signal pour tout le bandeau.

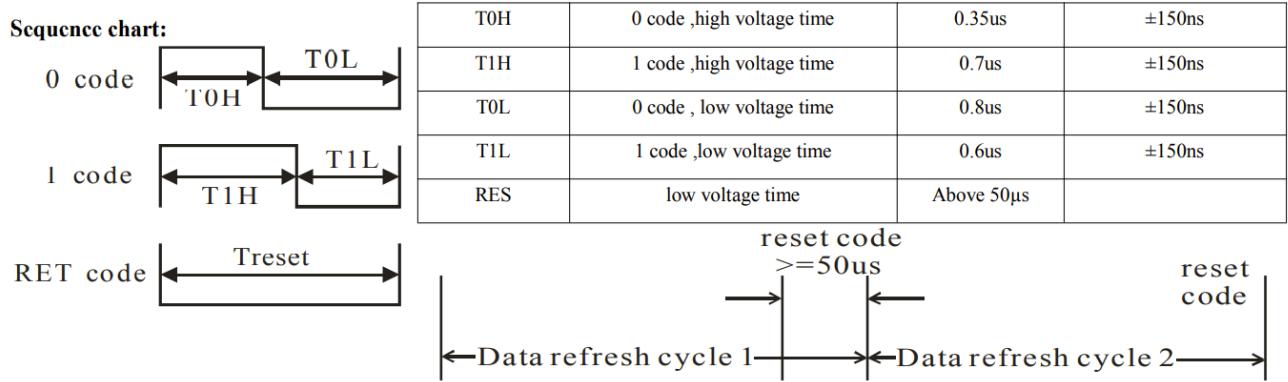
Notre bandeau possède donc seulement 3 sorties différentes (GND, 5Vdc, DIN). Afin de lier ces 3 sorties à la Basys, nous avons donc soudé des câbles Arduino.



Pour piloter la led, il nous faut transmettre au port Din une succession de bits correspondant aux codages sur 8 bits de chaque couleur du mélange Vert Rouge Bleu.

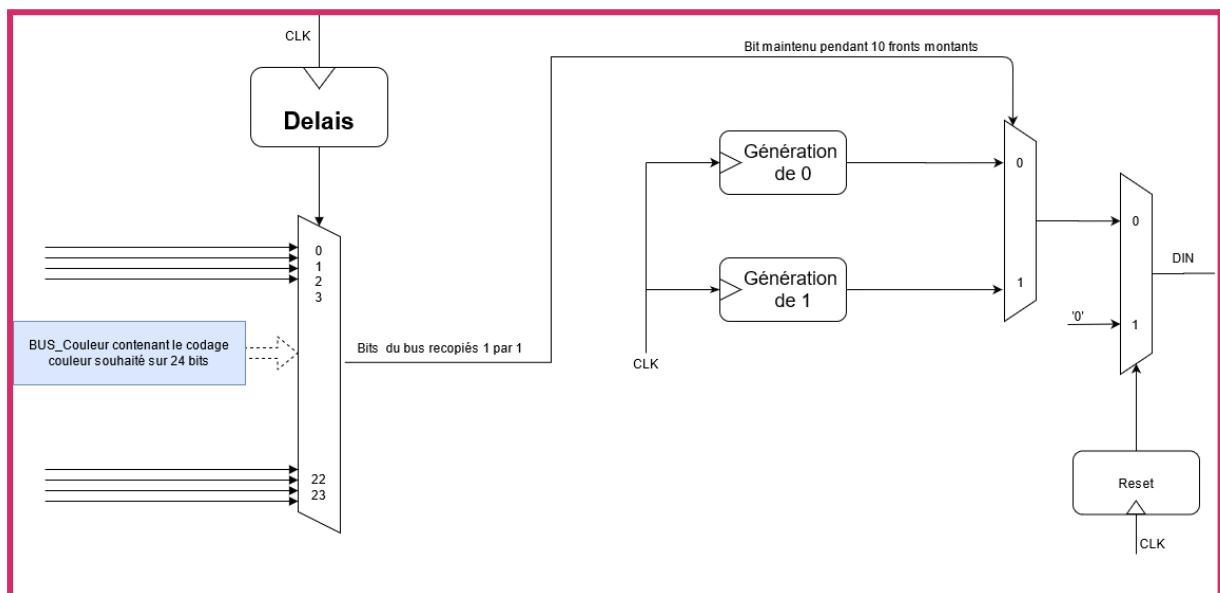
2. Spécificité

La communication avec la néopixel par le fil DIN doit respecter un format spécial permettant à la puce pilote de distinguer l'arrivée d'un 1 ou d'un 0.



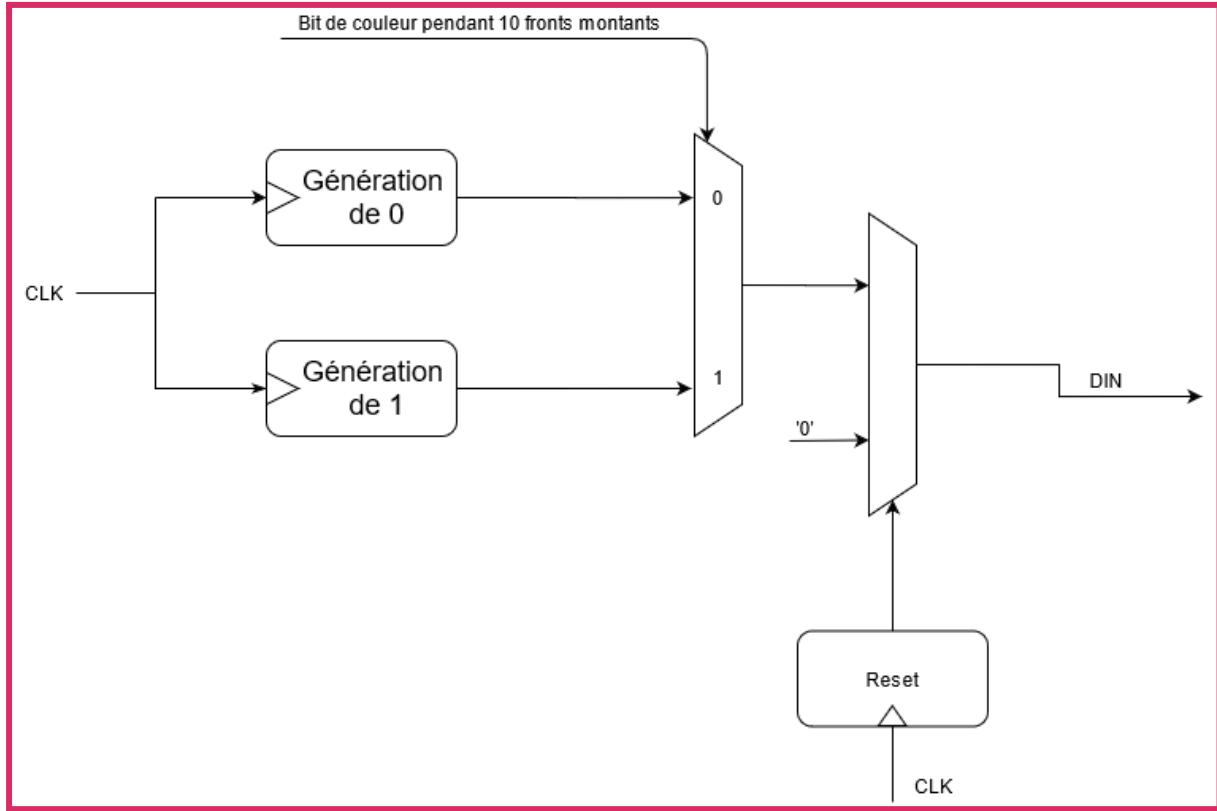
Le cahier des charges est donc le suivant :

- Une nouvelle actualisation de la couleur doit être effectuée au moins 50 us après la précédente (RES = reset time). Pendant ce temps Treset, seuls des '0' doivent être envoyés. Après, un nouveau cycle de données démarre correspondant à une nouvelle couleur.
- L'envoi d'un '0' ou d'un '1' correspond à l'envoi d'un signal créneau de rapport cyclique variable donné dans le tableau ci-dessus. Chaque bit envoyé correspondant au codage d'une couleur est donc converti en signal créneau.



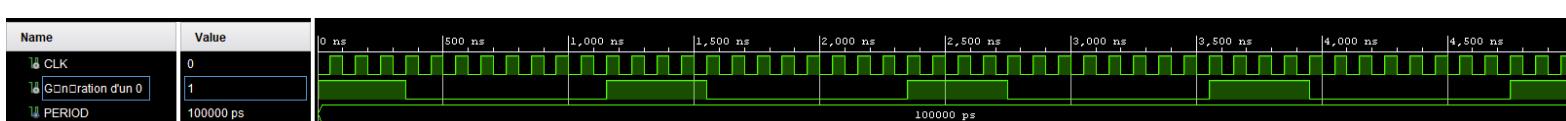
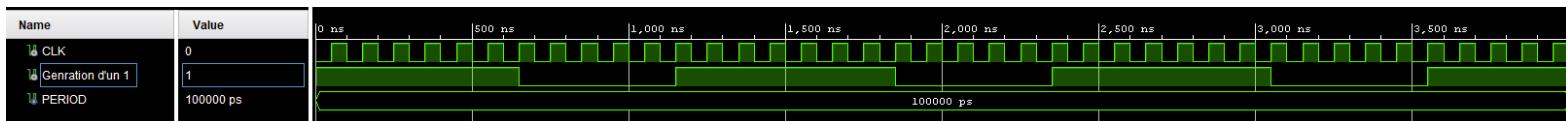
3. Application

- Sys.vhd



Gene_0 et Gene_1 :

Notre premier objectif a donc été la génération conforme de 0 et 1. Les rapports cycliques imposés ont défini notre nouvelle fréquence de fonctionnement de 10MHz. Ainsi, nous avons pu créer les différents créneaux.

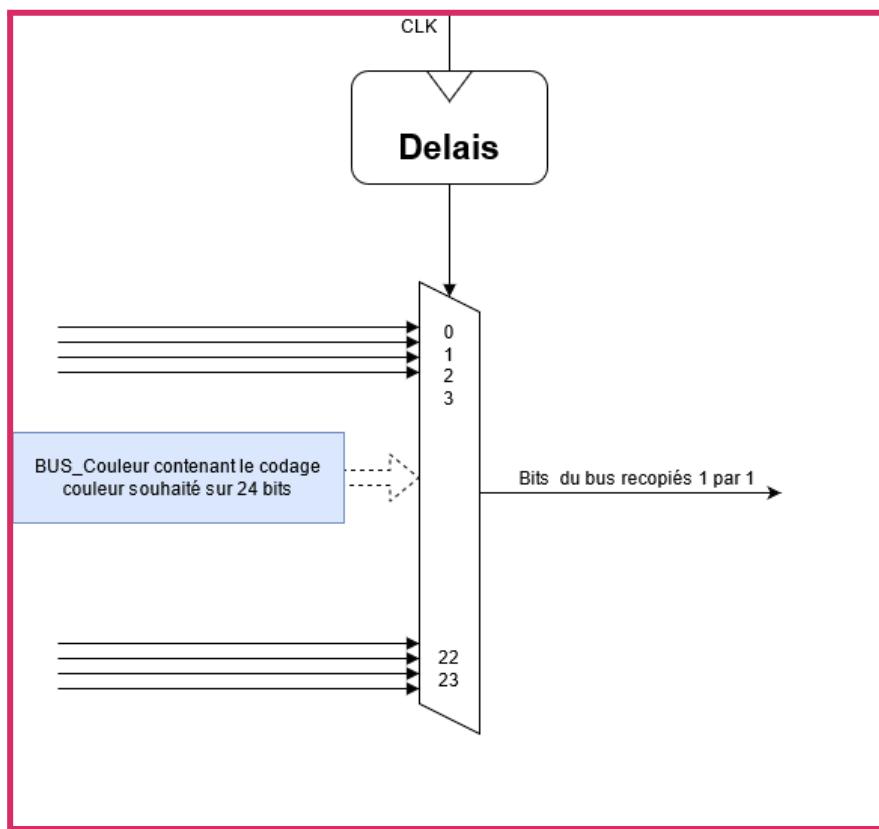


Reset_Block :

Nous avons géré la problématique du temps Treset entre chaque rafraîchissement de couleur en ajoutant le bloc Reset. Ce dernier contrôle un MUX et ainsi sélectionne alternativement pendant une durée conforme :

- Le signal issue de la génération créneau
- Un signal constant à '0' formant ainsi le signal de Reset.

- Delaïs.vhd



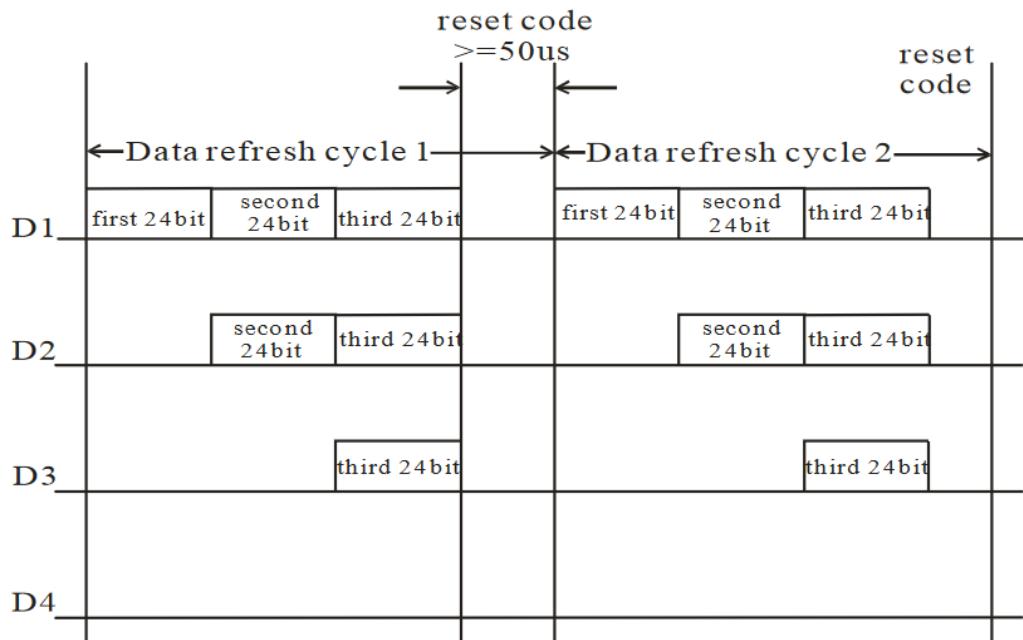
MUX24x1v1x1 :

Ce bloc prend un entrée un bus de 24 bits correspondant au codage de la couleur en VRB. (3x 8bits). Sa sortie est reliée à un MUX permettant de faire correspondre un bit avec son créneau correspondant.

Reset :

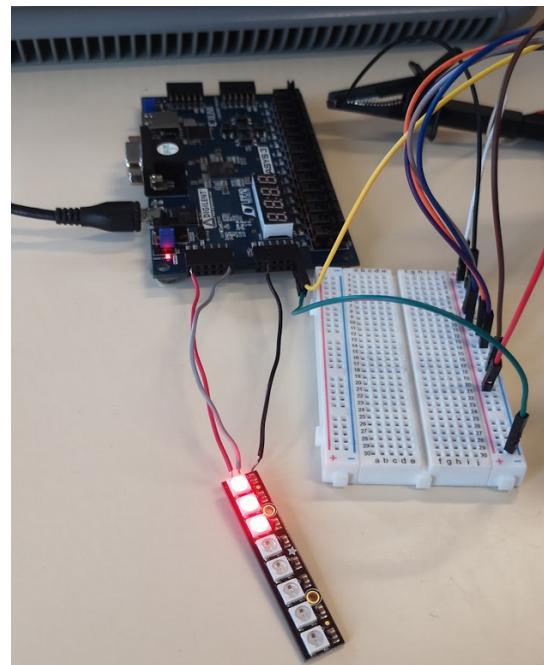
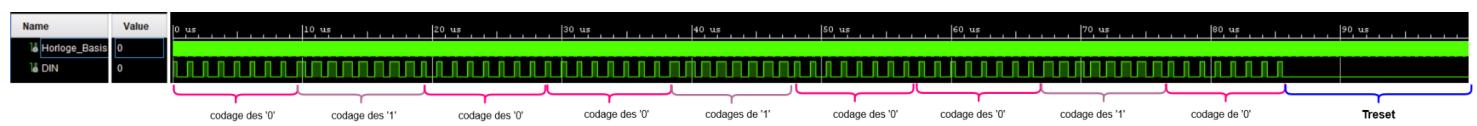
Ce bloc pilote le MUX 24x1v1x1 en attribuant à successivement à sa sortie un bit du bus de 24 bits. Mais attention, il ne faut pas faire varier la sortie du MUX à tout les fronts montant d'horloge puisque chaque bit de couleur doit être traduit en un signal créneau. Ainsi le bloc Reset veille à faire évoluer la sortie tous les 10 fronts montants d'horloge pour que l'ensemble du signal créneau correspondant puisse être envoyé sur DIN.

Data transmission method:



4. Exemple pour la couleur rouge

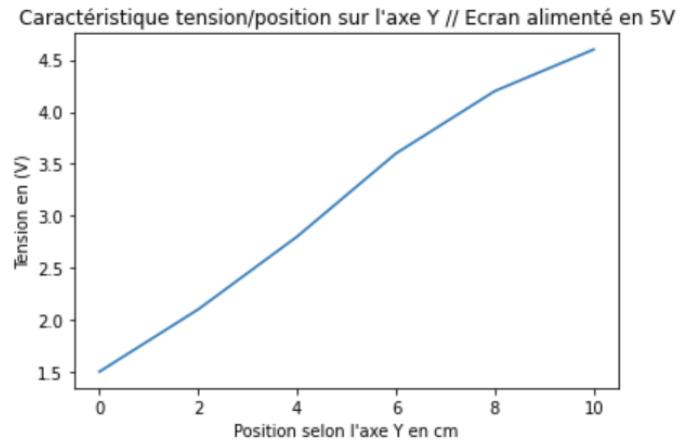
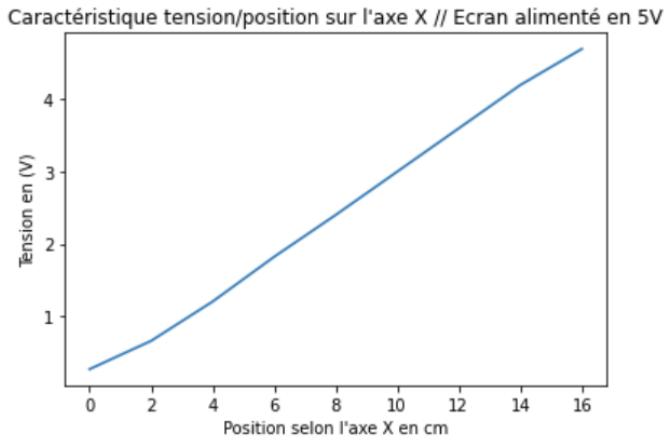
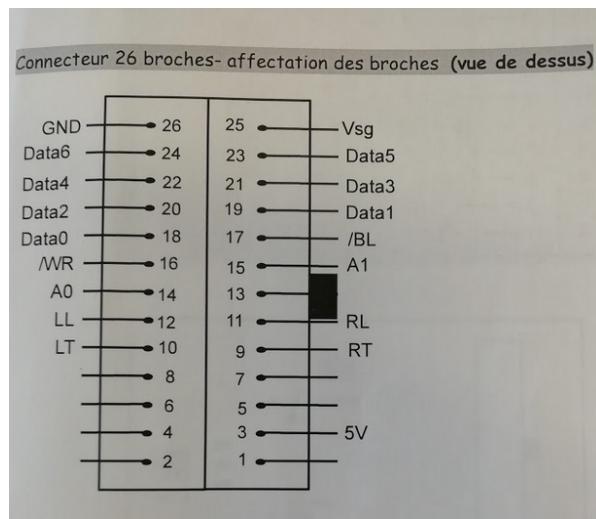
La couleur rouge correspond au codage “00000000 11111111 00000000” en VRB. On souhaite allumer 3 leds de notre bandeau led. Cela implique donc d’envoyer 3 fois le codage correspondant sous forme de créneaux puis un signal de Reset.



III. Intégration de l'écran tactile.

1. Mesure des tensions en fonction de l'appui sur la dalle tactile

Afin de connaître la tension envoyée par appui sur la dalle tactile, nous avons branché les broches du connecteur de la dalle à un oscilloscope. En analysant l'affectation des broches et en branchant les broches correctes (RT, RL, LT et LL), nous avons pu relever la tension correspondant à un minimum de tension lorsque l'appui se situe à gauche de l'écran (donc sur l'axe LT LL) et à un maximum de tension, égal à la tension d'alimentation de l'écran, lorsque l'appui se situe à droite (donc sur l'axe RL RT).



La tension délivrée par l'écran tactile est donc proportionnelle à la position du doigt selon chacun des axes X et Y.

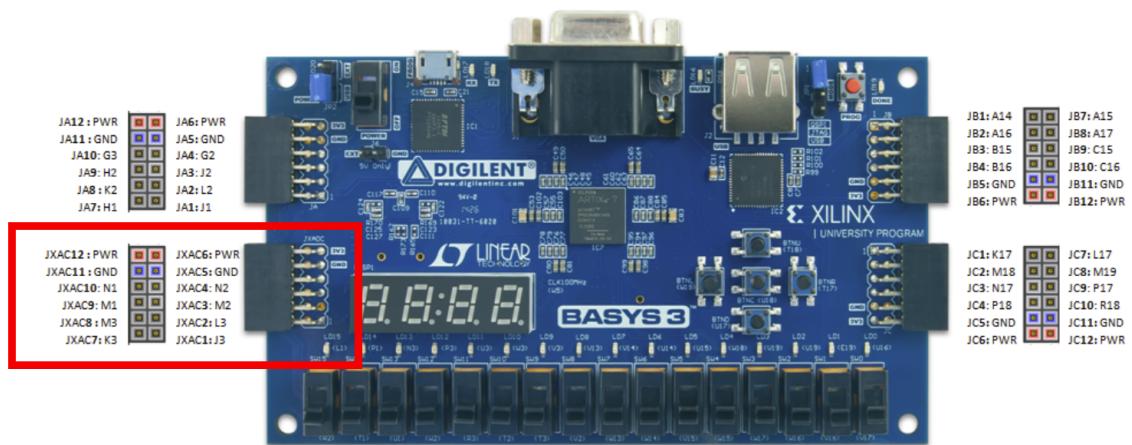
Montage :

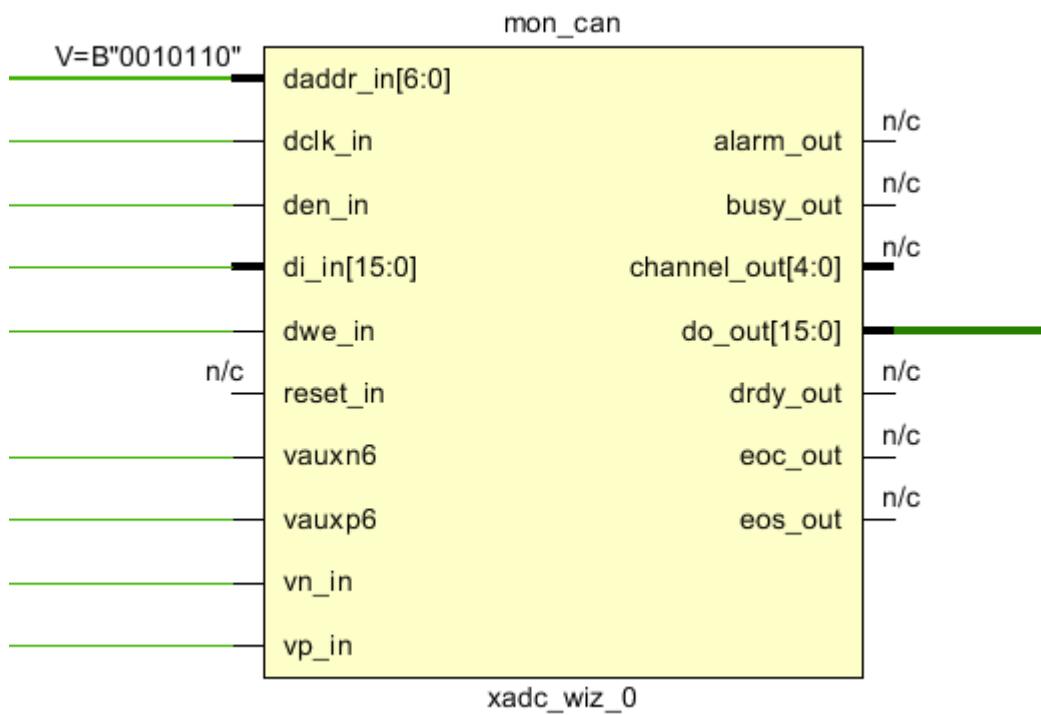


2. Liaison avec la Basys.

Pour traiter numériquement la tension de issue de l'écran tactile, nous avons utilisé le CAN interne de la basys. Attention, le CAN de la basys 3 admet une tension d'entrée maximale de 1V.

Basys3: Pmod Pin-Out Diagram



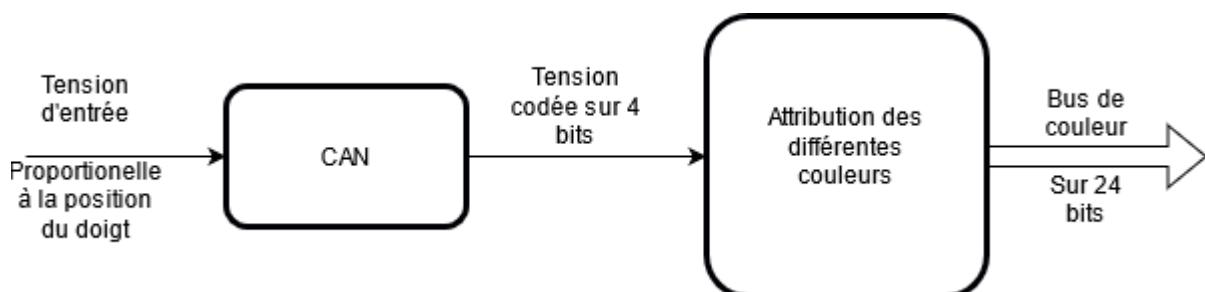


Le CAN convertit la tension donnée entre vauxp6 et vauxn6 en un mot de 16 bits. Puisqu'une telle précision n'était pas nécessaire, nous nous sommes limités aux 4 bits de poids fort. Ainsi, nous avons une plage de 16 mots différents.

Enfin, nous avons relié la valeur de ces 4 bits aux leds présentes sur la Basys afin de visualiser en direct la conversion.

3. Convertir la tension en une couleur.

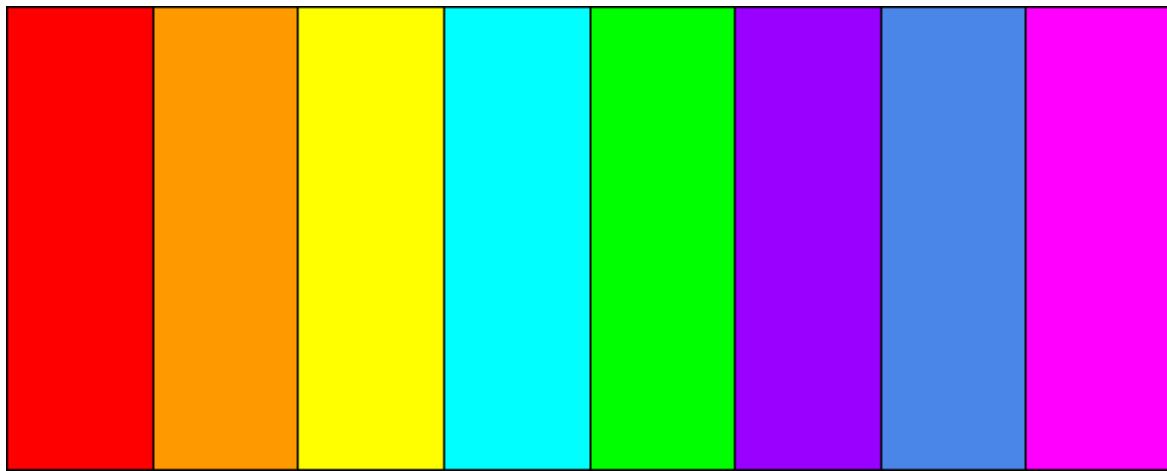
L'étape finale a été de faire correspondre les différentes tensions numériques sur 4 bits à des couleurs codées sur 24 bits.



IV. Affichage des couleurs sur la neopixel suivant l'axe X

1. Division de la dalle en 8 sections correspondant aux couleurs

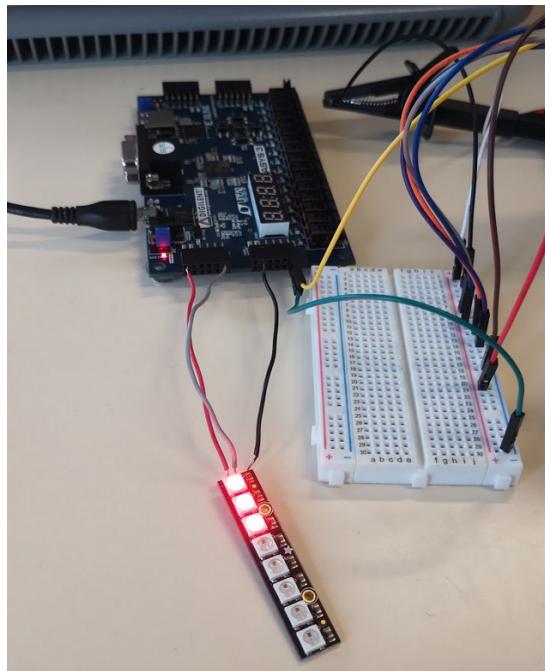
Rappelons l'objectif principal de ce projet : afficher une couleur en fonction de la position du doigt ! Pour réaliser cela, il est donc nécessaire de subdiviser l'écran en plusieurs parties, et que chaque section corresponde à une couleur selon l'appui. Nous avons donc décidé de créer 8 couleurs codées sur 24 bits, allant du rouge au rose, et évoluant suivant le passage du doigt dans chacune des sections de l'écran. Ainsi, un appui sur la droite de l'écran renvoie une tension de 1V codée en 1111 correspondant à la couleur rose. Visuellement, cela donne :



2. Test avec alim stabilisée

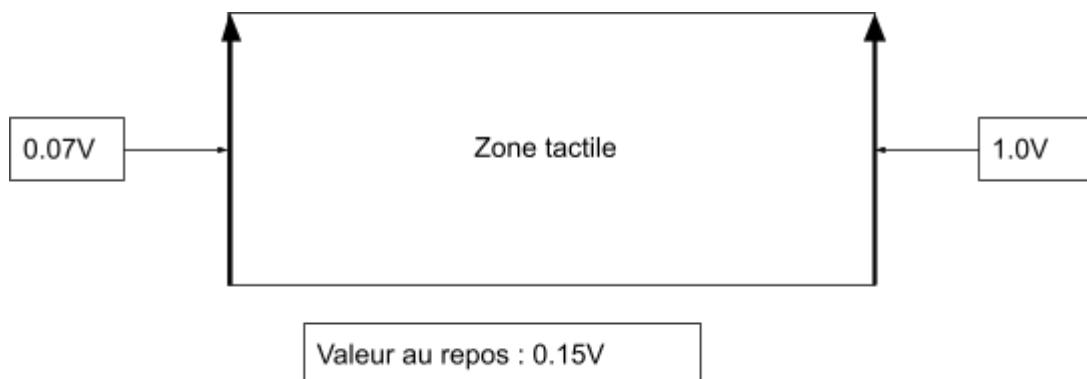
Pour tester si la neopixel fonctionnait et évoluait suivant l'axe X, nous avons utilisé une alimentation continue de 5V. Le résultat fut alors concluant mais l'écran tactile n'était pas utilisé en entier, puisque l'alimentation était 5 fois supérieure à la tension max que le CAN peut convertir qui est de 1V. En effet, seul 1/5ème de l'écran évolue dans ce cas-là, et ensuite la couleur reste rose.





3. Branchements uniquement sur la Basys

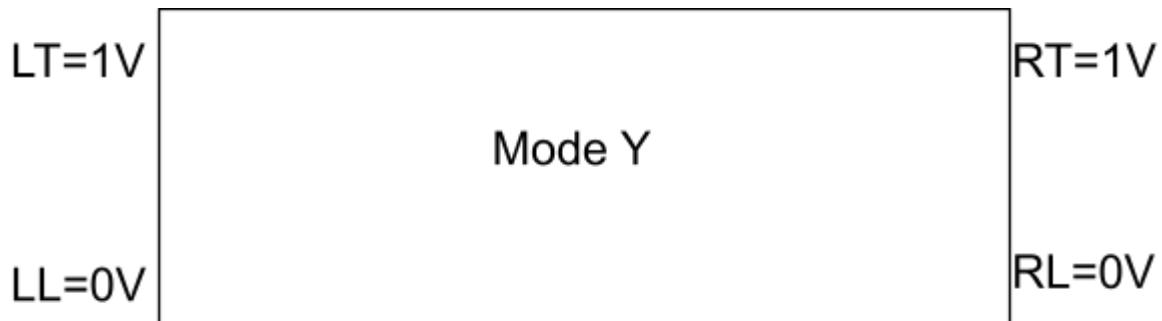
L'objectif étant de brancher le côté droit de la dalle tactile à 1V et le côté gauche à 0V, nous avons branché les broches RT, RL, LT et LL directement sur les ports PMOD. En commandant la valeur de ces derniers, l'alimentation devient inutile et la basys devient suffisante afin de paramétriser la valeur des tensions suivant l'axe X.



V. Mode Y

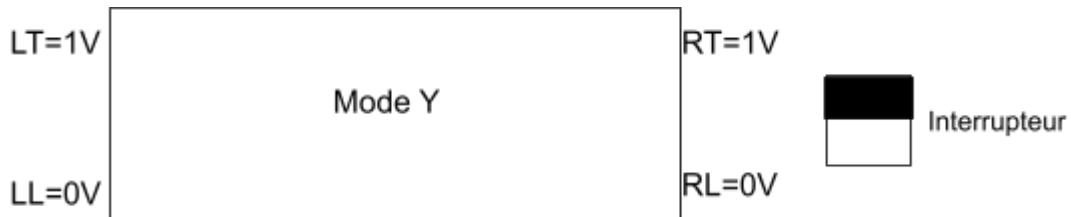
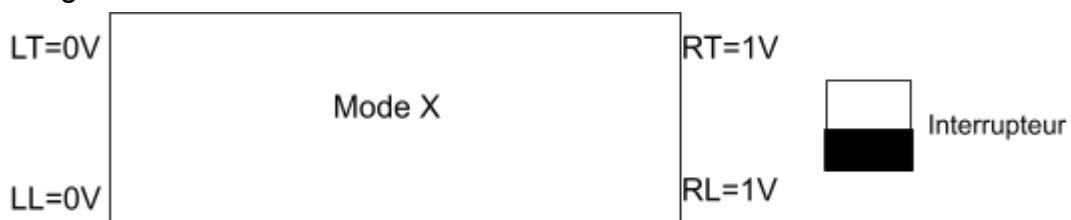
1. Principe

Le mode Y est une simple transposition du mode X. Il suffit d'alimenter à +Vcc la partie haute de l'écran et maintenir à 0V la partie basse. Pour cela, on alimente avec des '1' les bords RT et LT, alors que les angles LT et LL sont alimentés avec des '0'.



2. Ajout d'un interrupteur pour passer du mode X au mode Y.

Avant d'implémenter un mode X&Y, nous avons ajouté un interrupteur permettant de passer du mode X au mode Y à n'importe quel moment. Nous avons ainsi vérifié les changements d'alimentation successifs de l'écran tactile.



VI. Pour aller plus loin

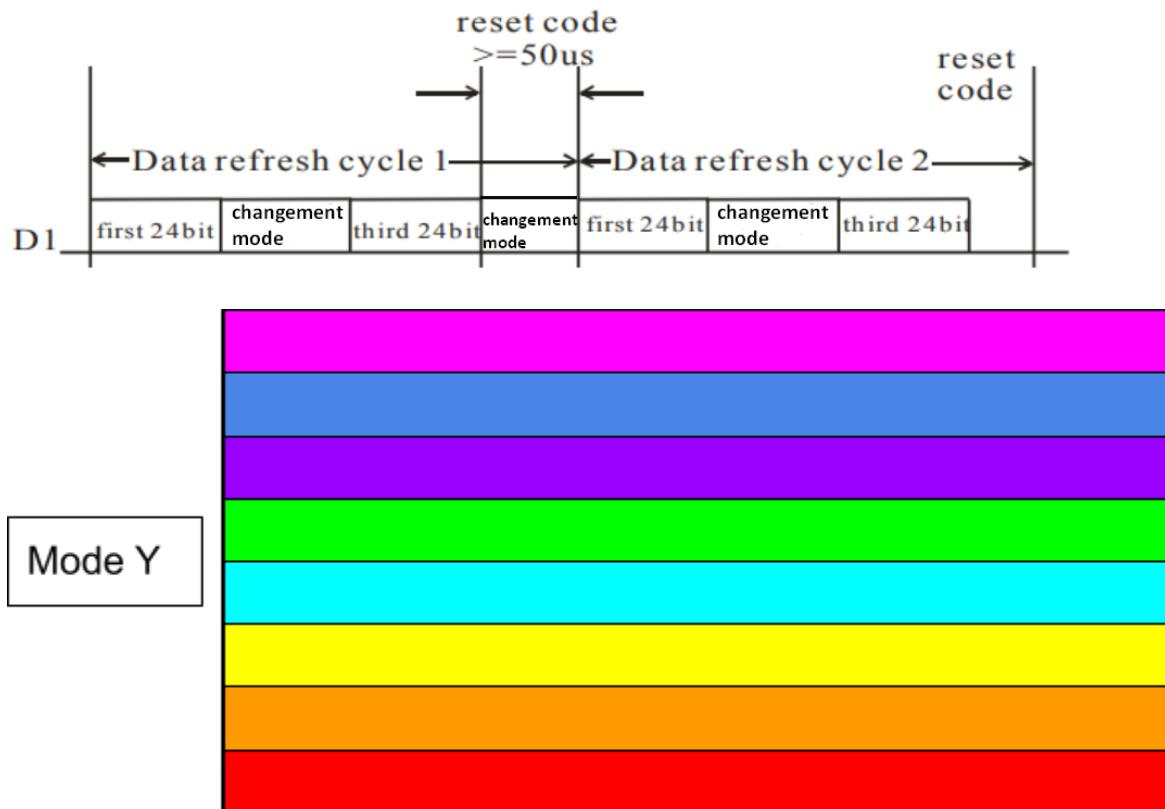
Nous avons étudié ces améliorations sans pouvoir les concrétiser par manque de temps.

1. Modification de la luminosité

La modification de la luminosité s'effectue en modifiant la tension de fonctionnement de la neopixel. Pour cela, on reprend le principe d'alimentation de l'écran tactile : l'envoi continu de '1' sur une broche pmod de la Basys3. Cela permet d'imposer une tension de 3.3V entre une broche et la masse. Pour faire varier cette tension, nous avons généré un signal créneau pour faire varier la tension moyenne de sortie entre la broche pmod en question et la masse.

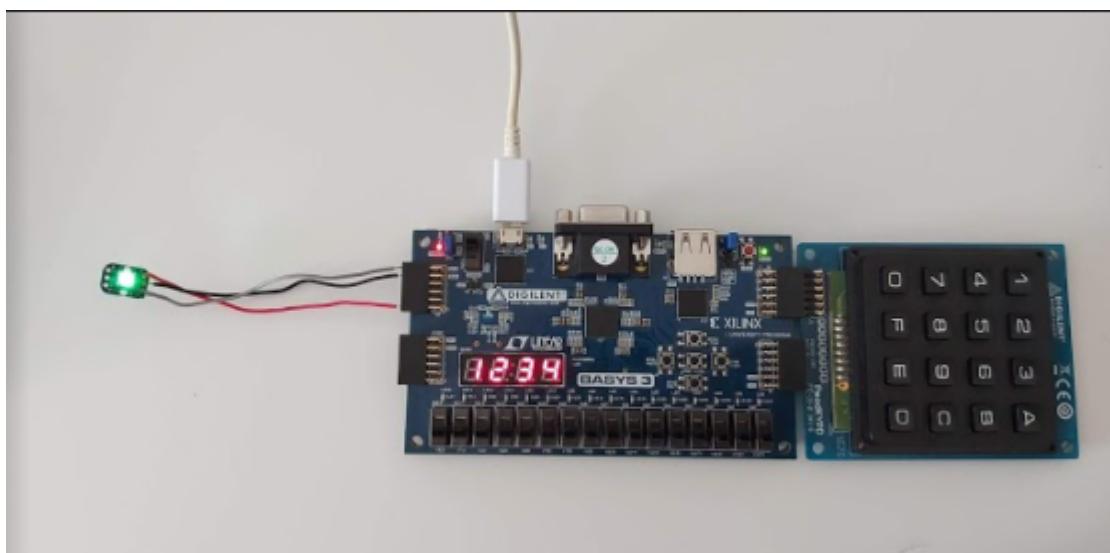
2. Mode x & y 'simultanés'

Finalement, nous avons tenté d'attribuer une led à la composante X et une seconde led pour la composante Y. Le changement d'alimentation de l'écran tactile pour alterner les modes de fonctionnement doit s'effectuer après chaque envoi d'une couleur.



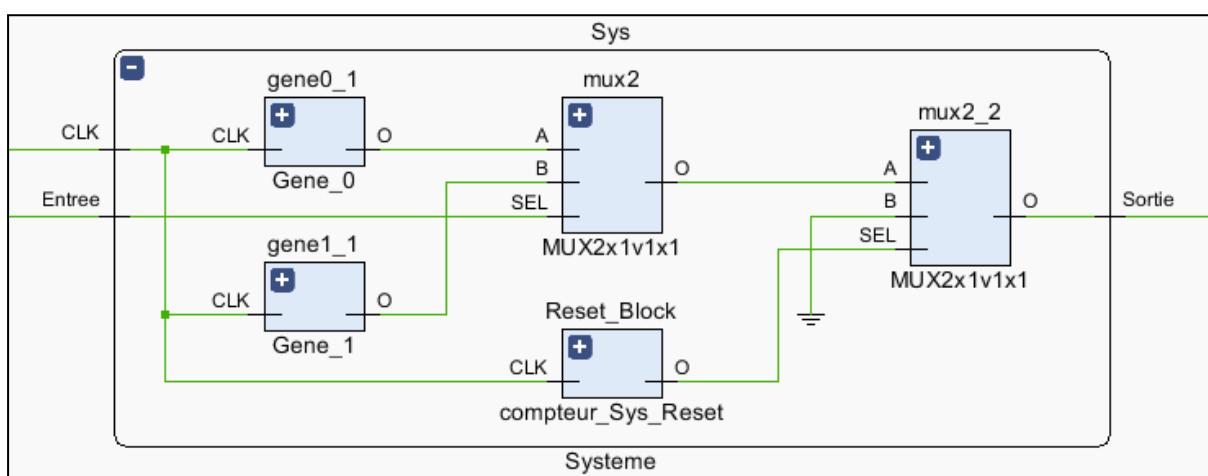
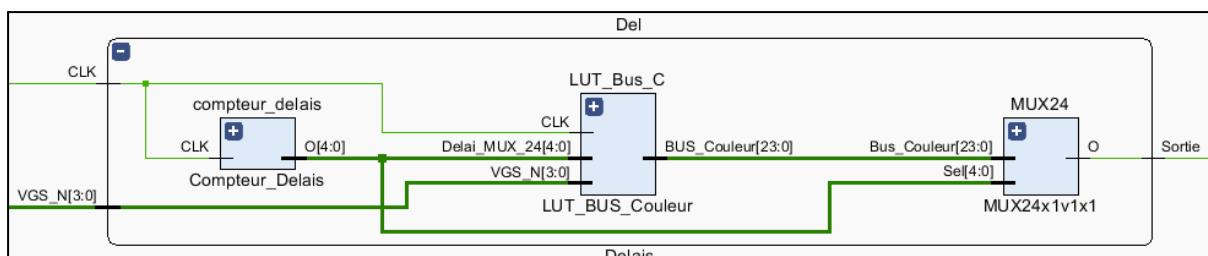
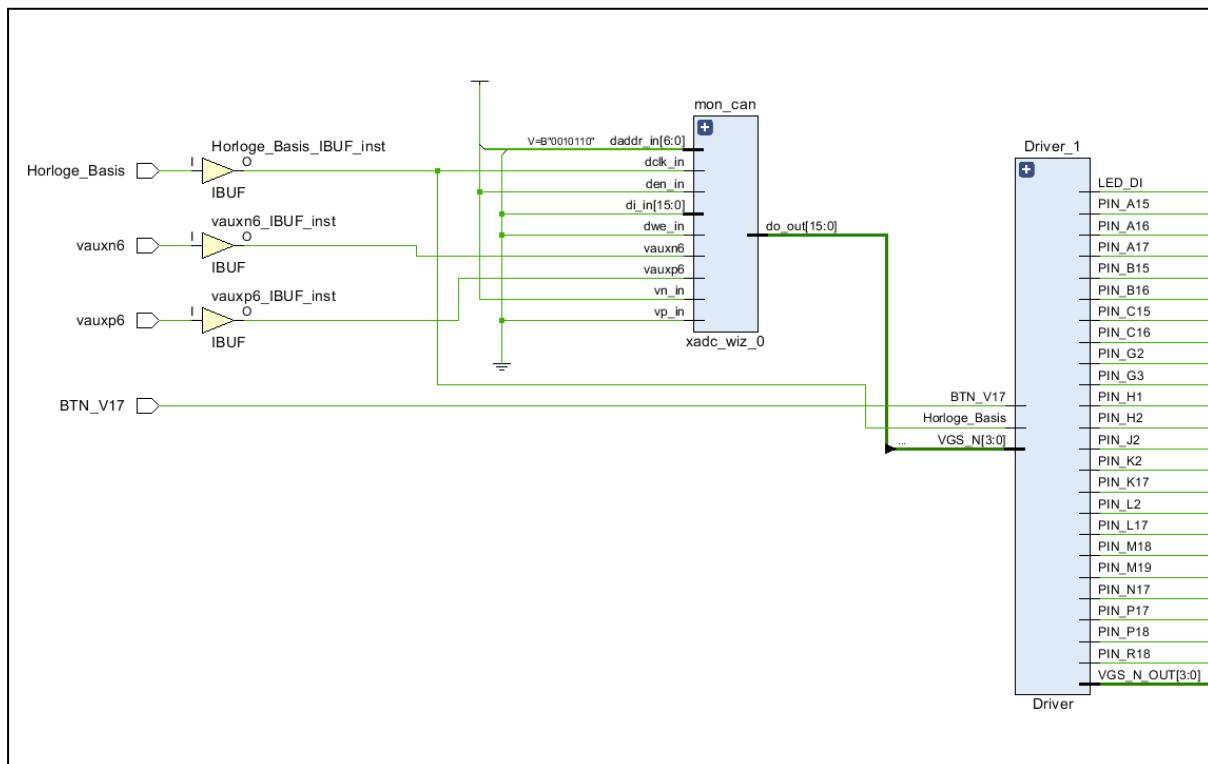
Dernière précision :

Il est à noter que notre projet a été un peu ralenti lors des dernières séances par un autre groupe ayant besoin du driver de notre neopixel. En effet, ce projet consistait à créer un digicode. A partir d'un clavier hexa, l'utilisateur tape un code à 4 chiffres et si le code correspond au code en mémoire, la led émet un double flash. Cela nous a tout de même permis de reprendre une partie de notre code VHDL afin de le rendre compréhensible et utilisable par d'autres groupes.



Annexes

Schematics détaillé :



Broches PMOD Basys 3 :

Basys3: Pmod Pin-Out Diagram

