

Rapport de projet de 1ere année 2022 :

Animation programmable LED

Objectif :

Mettre en place une carte avec des LEDs, que l'on pourrait alors programmer en envoyant une commande depuis un ordinateur grâce à un programme en C/Python. L'idée est alors de réaliser un petit écran de LEDs 4x4 qui nous permettra ensuite de réaliser de petites animations.

Matériel :

- STM32F410
- Driver de LEDs
- LEDs de toutes les couleurs
- Oscilloscope / Multimètre
- Résistances

Sommaire:

1) Présentation du Driver

- a) *Utilisations*
- b) *Spécificité du STP04CM05XTTR*

2) Programmations des signaux

- a) *Utilisation d'un Code en mode "Timer"*
- b) *Programmation sous STM32CUBE IDE*

3) Réalisation d'un circuit imprimé

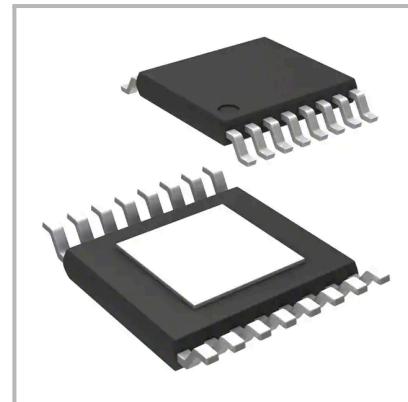
4) Tests et Conclusion

Présentation du Driver

Le composant central de notre projet est le driver de LED commandable STP04CM05XTTR

C'est pourquoi il est important de bien comprendre sa fonction et comment il fonctionne pour la suite du projet.

STP04CM05XTTR



Utilisations

Un driver de LED classique possède deux fonctions principales :

- Ajuster la tension que peut recevoir les LEDs.
- Réguler le flux d'électricité .

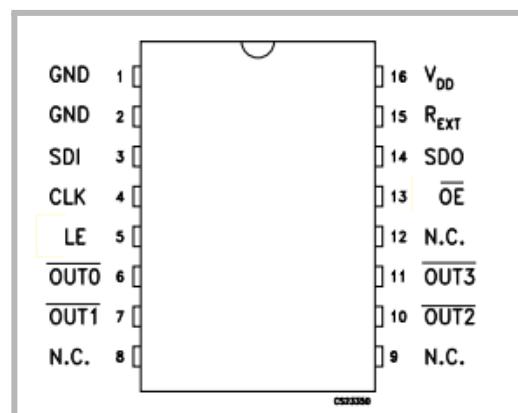
Tout ça dans le but de pouvoir allumer de multiples LEDs avec une seule tension sans mettre en danger celles ci.

C'est un composant très utilisé pour gérer de multiples LEDs, notamment pour les réalisations d'écrans, de guirlande ou de bandes de LEDs.

Spécificité du STP04CM05XTTR

Notre driver ne possède que 4 sorties pour allumer des LEDs :
OUT1 OUT2 OUT3 et OUT4

Mais il possède un système de commande grâce aux entrées SDI CLK et LE qui vont permettre de choisir quelle sortie allumer



il nous reste maintenant à décrypter les entrées sorties

CLK (IN)

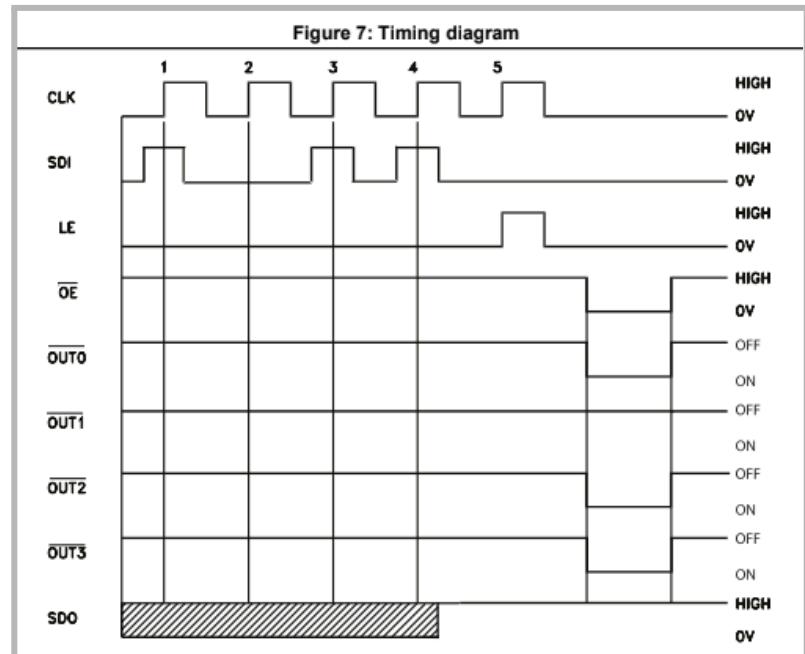
la clock à envoyer dans le driver, elle sera utilisé uniquement par SDI (1 - 20 MHZ de fréquences)

SDI (IN)

Serial Data Input c'est le signal qui va permettre de coder les informations (uniquement sur le front montant de l'horloge) High → ON, 0V → OFF

LE (IN)

Latch Enable est le signal qui va indiquer à notre composant de garder en mémoire la séquence.



OE (IN)

Output Enable active les sorties.

SDO (OUT)

Serial Data Output est le même signal que SDI mais avec l'information qui a été latch en moins, il permet de mettre plusieurs drivers en chaîne et de tous les gérer avec un seul signal de commande

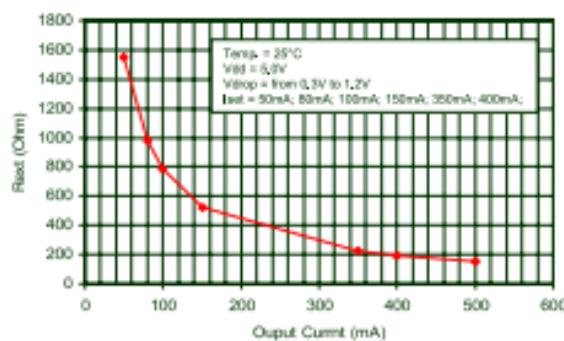
OUT0 / OUT1 / OUT2 / OUT3 (OUT)

Les sorties de notre driver qui dépendent donc de SDI, du LE et surtout de si OE est activé

REXT (IN)

l'entrée Reset va permettre de moduler l'intensité en sorties en variant la résistance branchée selon cette courbe.

Figure 13: Output current-REXT resistor



Programmations des signaux (*dans le code gn est le compteur*)

On a utiliser un code en mode “TIMER” qui se répétait à une certaine fréquence pour reconstruire chaque signal a la main.

1. Sortie OE (Output Enable)

```
while (1)
{
    // OE (Bouton 1 PC_13)
    BoutonLu=HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
    if(BoutonLu==GPIO_PIN_RESET){
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_RESET);
    }
    else{
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);
    }
}
```

Cette sortie enverra un signal à la broche PB_3 (D3) Pour pouvoir controller l'allumage ou non des LEDs avec le Bouton.

2. Sortie LE (Latch Enable)

```
// GENERATION de LE broche PA_10 (LE)
if(gn<20){
    if((gn+1)%20==19 || (gn+1)%20==0){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_SET);
    }
    else{
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_10, GPIO_PIN_RESET);
    }
}
else
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_10, GPIO_PIN_RESET);
```

Cette sortie enverra un signal à la broche PA_10 (D2).

3. Sortie CLK (clock) :

```
// GENERATION DE CLK broche PB_4
if(gn<16){
    if(((gn%4)/2)==0){
        // HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);
        // HAL_GPIO_WritePin(GPIOA, CLK_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,CLK_Pin, GPIO_PIN_RESET);
    }
    if(((gn%4)/2)==1){
        // HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET);
        // HAL_GPIO_WritePin(GPIOA, CLK_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOB, CLK_Pin, GPIO_PIN_SET);
    }
}
else
    HAL_GPIO_WritePin(GPIOB,CLK_Pin, GPIO_PIN_RESET);
```

composé de 4 temps (le signal est nul pour 2 temps et égale à 1 pour

les 2 autres) qui se rèpètent en boucle, ceci est fait pour pouvoir synchroniser notre système

Visualisation du signal clk à l'aide d'un oscilloscope :



(résultat cohérent avec le code)

4. Sortie SDI :

```
// GENERATION DE SDI broche PB_5

int Ordre[4] = {0,1,0,1};
if(gn<20){
    // Premier bit
    if(gn%20 ==1 || gn%20 ==2 || gn%20 ==3 ){
        if (Ordre[0] == 1){
            HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_SET);
        }
        else{
            HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_RESET);
        }
    }

    //Deuxieme
    if(gn%20 ==5 || gn%20 ==6 || gn%20 ==7 ){
        if (Ordre[1] == 1){
            HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_SET);
        }
        else{
            HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_RESET);
        }
    }
    //...
}

//Troisieme
if(gn%20 ==9 || gn%20 ==10 || gn%20 ==11 ){
    if (Ordre[2] == 1){
        HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_SET);
    }
    else{
        HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_RESET);
    }
}

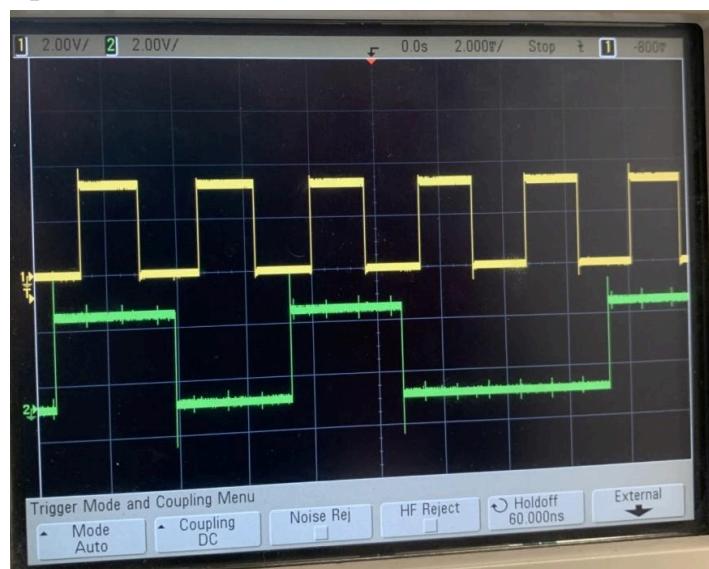
//Quatrieme
if(gn%20 ==13 || gn%20 ==14 || gn%20 ==15 ){
    if (Ordre[3] == 1){
        HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_SET);
    }
    else{
        HAL_GPIO_WritePin(GPIOB, SDI_Pin, GPIO_PIN_RESET);
    }
}

}
else
    HAL_GPIO_WritePin(GPIOB,SDI_Pin, GPIO_PIN_RESET);
```

Cette sortie enverra un signal à la broche PB_5 (D4), pour pouvoir contrôler (et animer) Les LEDs

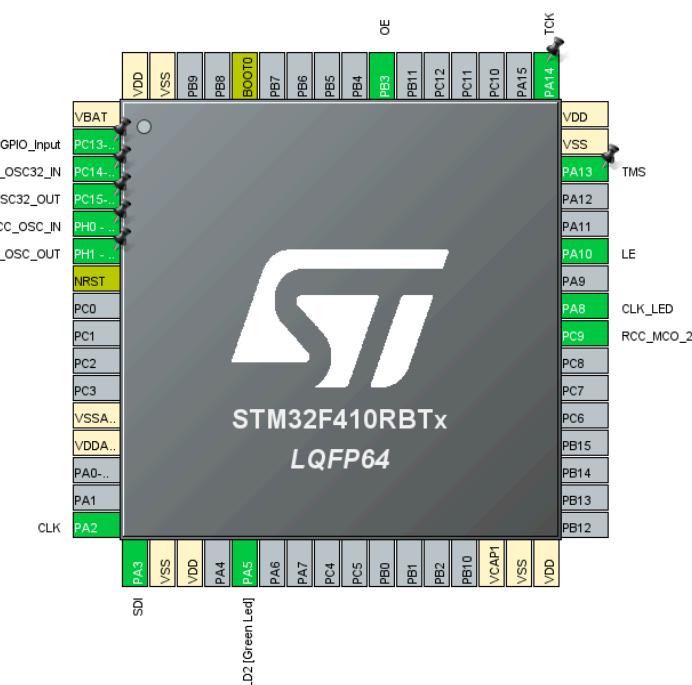
Par exemple : Dans ce cas on a pris comme entrée [0, 1, 0, 1], donc on voit bien que seulement les LED 2 et 4 qui s'allume (les LEDs 1 et 3 restent éteints)

Visualisation du signal SDI (en vert) à l'aide d'un oscilloscope :
(Dans ce cas on a pris [1, 0, 1, 0] comme entrée)



On voit bien que SDI vaut 1 pour le premier est le troisième front montant et il est nul pour le 2ème et 3ème

Pinout view :



Ici on a déclarer tous les sorties (outputs) du système

Réalisation d'un circuit imprimé

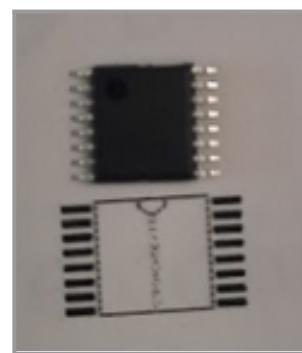
Petit circuit pour breadboard

Il nous faut rapidement avoir un composant disponible pour réaliser des tests avant de mettre en place le circuit final. Il nous faut donc réaliser un petit circuit pour mettre notre composant sur breadboard.

Néanmoins la trace n'était pas disponible nous avons alors dû la réaliser.

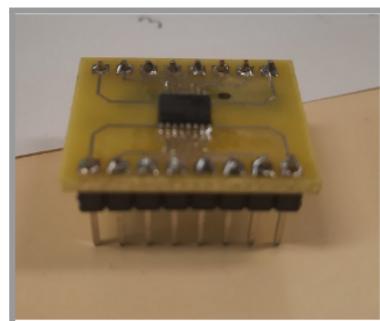


Réalisation sur eagle



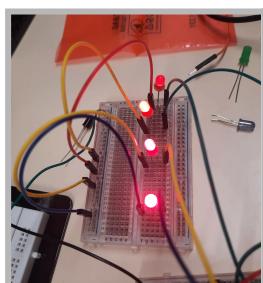
Et vérification de l'empreinte

Mais ce n'était qu'une perte de temps puisque ce genre de circuit est assez classique et les techniciens en avaient déjà en stock, il nous a alors suffit de souder et notre composant était prêt pour les tests !

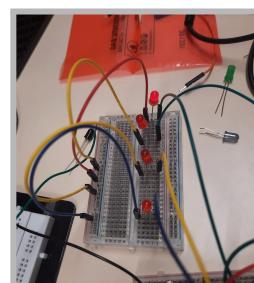


Néanmoins lors des tests la tension sur la sortie 4 était très basse et nous n'avons pas pu trouver le problème (la soudure était bonne)

ON



OFF



Circuit imprimé pour un Driver

A cause d'un manque de temps, nous avons préféré réaliser notre circuit imprimé pour un seul driver plutôt que plusieurs, pour au moins finaliser notre projet avec un driver.

Nous avons donc réalisé notre PCB sur eagle

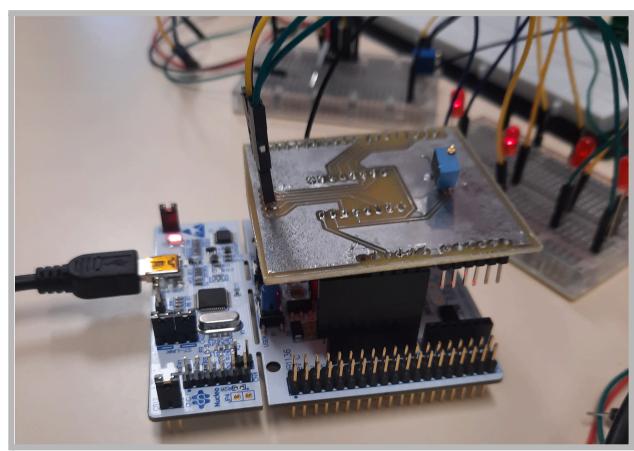
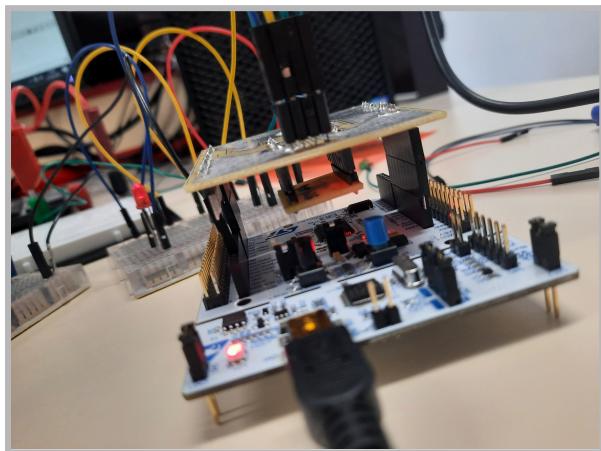
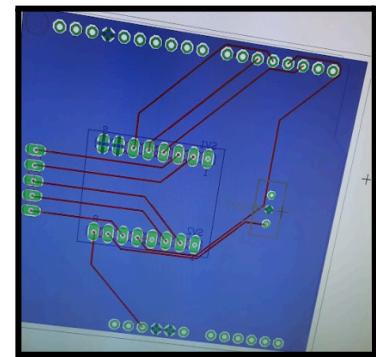
Celui-ci doit donc pouvoir se brancher directement à la STM32 et rester assez flexibles si jamais on veut effectuer des changements dans notre projet.

On a donc essayé de simplifier au maximum notre pcb en mettant tout sur une face avec des plans de masses et en mettant le composant a l'envers.

Néanmoins sur le premier jet nous avons oublié de mettre une épaisseur d'isolation et on a donc dû le refaire.

lors de la réception du PCB nous nous sommes rendu compte que nous avions oublié de faire l'effet miroir sur la trace du stm32 et donc notre composant ne pouvait pas s'imbriquer comme prévu.

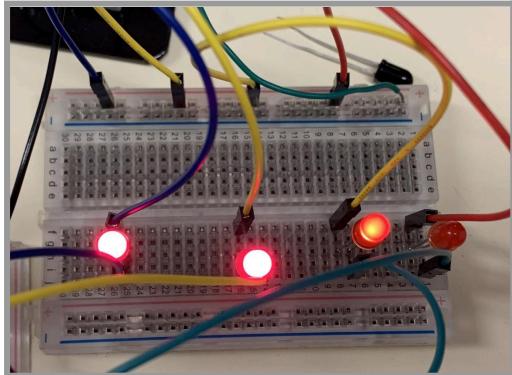
on a réussi à trouver un moyen de le brancher quand même mais la connectique n'était pas bonne et donc rien ne marchait.



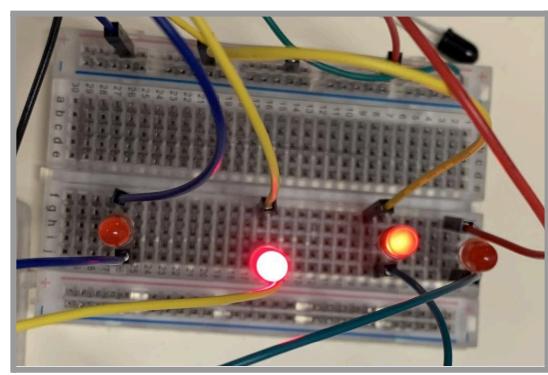
Tests et Conclusion

Ainsi finalement notre projet a pu aboutir sur un certain nombres de points :

- Compréhension de notre composant.
- Encodage des signaux.
- Réalisation d'un circuit imprimé.
- Commande de nos LEDs.



code 1110



code 0110

Néanmoins il nous restait de nombreuses pistes a abordé :

- Utiliser plusieurs drivers en chaîne.
- Refaire un PCB fonctionnel.
- finalisé le projet en réalisant proprement un écran.

Nous avons tout de même appris à utiliser eagle et STM32CUBE IDE pour nos futurs projets et ne perdrons pas autant de temps sur nos futurs projets.