ID-Fab

Juin 2024

Projet Prototypage

Capacitron Duel (Jeu de rapidité et de reflexe)

Baptiste MONDION

RESPONSABLES ACADÉMIQUES

François BERNIER

MINES Saint-Étienne | Institut Mines-Télécom

Table des matières

Remerciements

Nous tenons à exprimer notre sincère gratitude envers les personnes suivantes :

François BERNIER pour sa réactivité, sa pédagogie et son assistance tout au long de notre projet.

Cyril CALMES pour nous avoir fait découvrir ce beau logiciel qu'est Kicad.

Roger DELATTRE pour ses conseils.

Introduction

Capacitron Duel est un jeu interactif de rapidité et de réflexe, où deux joueurs s'affrontent, qui s'inspire d'un jeu d'arcade. Le jeu est simplifié avec seulement 1 capteur par joueur. Lorsque la LED rouge est allumée, il ne faut pas se faire détecter sous peine de perdre un point. Lorsqu'un délai aléatoire déclenche les LEDs vertes, le premier joueur à être détecté gagne un point. Le jeu continue jusqu'à ce qu'un joueur atteigne 9 points, moment auquel le jeu se termine.

Ressources

Dans ce projet, nous allons avoir besoin d'outils de modélisation à travers Comsol pour les simulations des forces physiques et Partquest pour la simulation des schémas électriques, ainsi que Draw.io pour les schémas et tout ce qui entoure le conceptuel.

Nous allons aussi avoir besoin d'outils pour la conception avec KiCad pour les PCB et Autodesk Inventor pour l'impression des pièces. STM32 CubeIDE pour coder sur les cartes ST et Visual Studio pour mettre en forme le code.

Figure 1. Logos des différentes applications utilisées

Concept

Le concept de notre jeu peut être résumé par le schéma suivant :

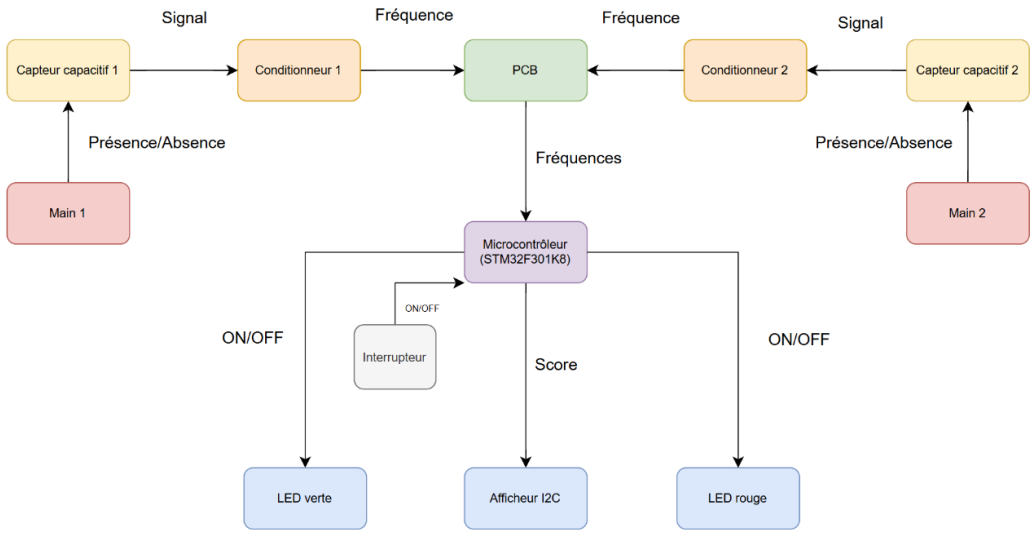

Figure 2. Schématisation de notre modèle sur Draw.io

On y voit la nécessité de mettre en place un PCB pour faire fonctionner les deux capteurs et conditionneurs sur le même microcontrôleur.

Description des étapes du projet

Faire du prototypage, c'est aussi un mode de fonctionnement. C'est savoir conceptualiser, puis modéliser avant de concevoir et tester. C'est pourquoi, dans cette logique qui est aussi celle de l'ingénieur, nous allons premièrement modéliser notre conditionneur sur Partquest et analyser nos électrodes sur Comsol pour déterminer comment optimiser leur fonctionnement. Nous testerons ensuite le modèle avec une seule électrode sur une carte, puis nous concevrons les éléments de notre projet avec KiCad et Autodesk Inventor.

Conception

Pour réaliser notre récepteur d'information, nous allons avoir besoin de mesurer les variations de la capacité du capteur. Celle-ci est modifiée par la permittivité relative de la Terre, par la surface du capteur et la distance au capteur (C = ϵ×S/d).

Conditionneur

Partie théorique

La loi de base pour un condensateur est :

$$I = C\frac{dV}{dt}$$

En intégrant cette équation sur un intervalle de temps ΔT avec une variation de tension ΔV, nous obtenons :

$$\Delta V = \frac{I}{C}\Delta T$$

Ce qui peut être réarrangé pour trouver ΔT :

$$\Delta T = \frac{\Delta V C}{I}$$

Selon l'énoncé, la tension aux bornes de la capacité est la moitié de V_{out} :

$$\Delta T = \frac{\Delta V_{out}}{2} \frac{C}{I}$$

Les tensions de basculement selon le théorème de Millman sont :

$V_{seuil+} = V_{max} \frac{R_6}{R_5+R_6}$

$V_{seuil-} = -V_{max} \frac{R_6}{R_5+R_6}$

Les tensions de basculement V_+ et V_- sont spécifiées comme étant un quart de la tension d'alimentation V_{max} :

$V_+ = \frac{V_{max}}{4}$
$V_- = -\frac{V_{max}}{4}$

Donc la variation de tension ΔV est :

$$\Delta V = V_+ - V_- = \frac{V_{max}}{4} - \left(-\frac{V_{max}}{4}\right) = \frac{V_{max}}{2}$$

Substituons ΔV dans l'équation pour ΔT :

$$\Delta T = \frac{\frac{V_{max}}{2}}{2} \frac{C}{I} = \frac{V_{max} C}{4I}$$

Le courant I est donné par :

$$I = \frac{V_{in}}{R_4}$$

Avec $V_{in} = V_{max}$, donc :

$$I = \frac{V_{max}}{R_4}$$

En substituant I dans l'expression de ΔT :

$$\Delta T = \frac{V_{max} C}{4 \frac{V_{max}}{R_4}} = \frac{R_4 C}{4}$$

La période T est deux fois ΔT :

$$T = 2\Delta T = 2\frac{R_4 C}{4} = \frac{R_4 C}{2}$$

La fréquence F est l'inverse de la période T :

$$F = \frac{1}{T} = \frac{2}{R_4 C}$$

Donc, nous avons démontré que :

$$F = \frac{2}{R_4 C}$$

Ce qui conclut la démonstration de la relation entre la fréquence du conditionneur et la capacité du capteur.

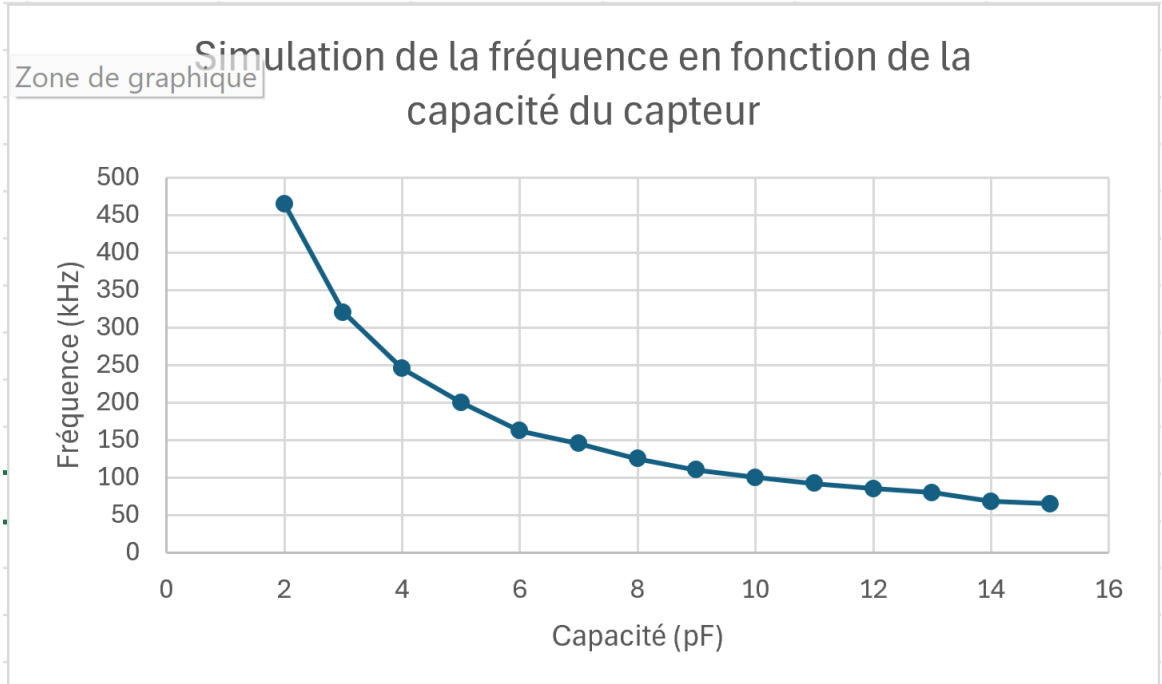

Figure 3. Simulation de la variation de la fréquence en fonction de la capacité du capteur

Simulation électrique

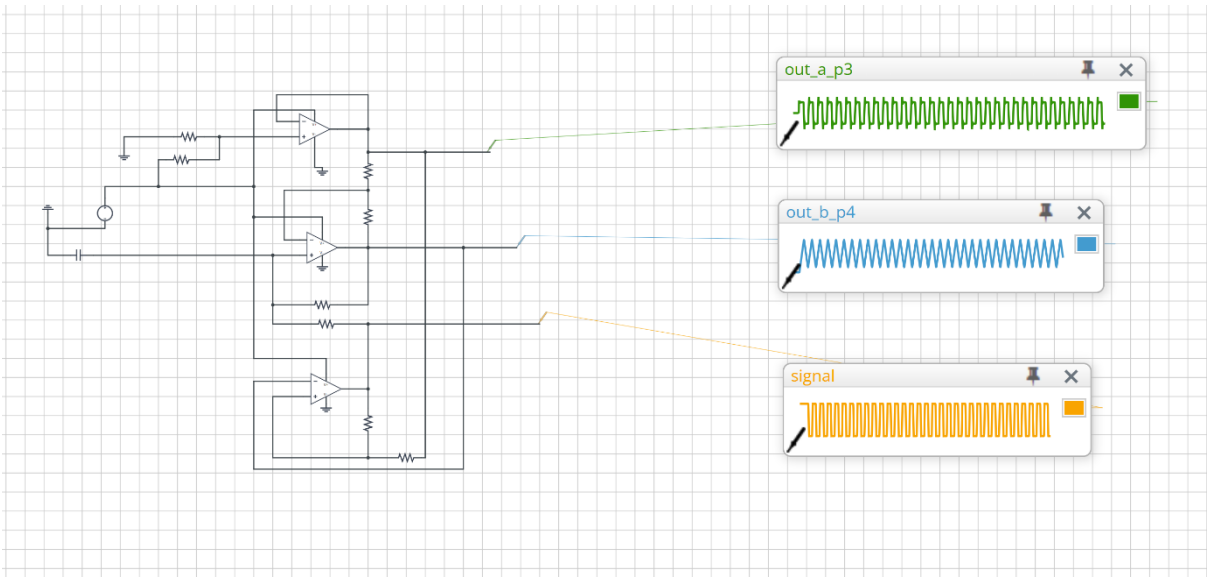

Figure 4. Montage complet réalisé sur Partquest

La première étape dans la conception du conditionneur est de mettre en place un convertisseur tension-courant, qui permet d'envoyer un courant constant I dans la capacité.

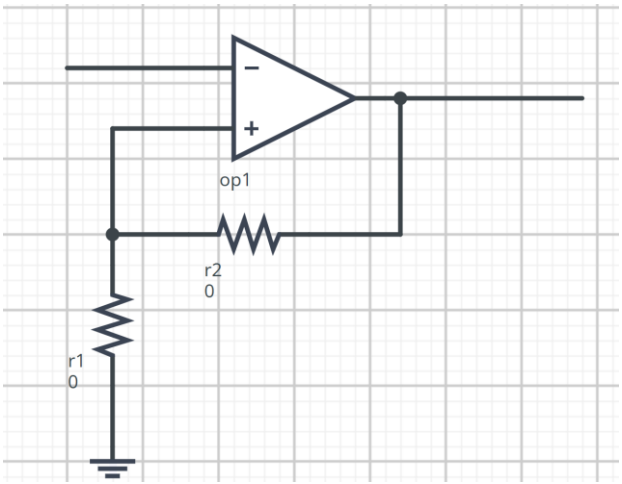

Figure 5. Convertisseur tension-courant

Dans ce montage, nous analysons un amplificateur opérationnel idéal en régime linéaire avec $V+ = V-$. En utilisant le théorème de Millman aux nœuds $V+$ et $V-$, nous obtenons les équations suivantes :

Pour $V-$:
$V- = \frac{V_{out}R_2}{R_1+R_2}$

Pour $V+$:
$V+ = \frac{V_{in}}{R_4} + \frac{V_{out}}{R_3}\left(\frac{1}{R_4} + \frac{1}{Z_c} + \frac{1}{R_3}\right)$

Étant donné que $V+ = V-$, en égalisant les deux expressions, nous avons :
$\frac{V_{out}R_2}{R_1+R_2} = \frac{V_{in}}{R_4}\left(1 + \frac{R_4}{Z_c} + \frac{R_4}{R_3}\right) - \frac{R_4(R_1+R_2)}{R_1R_3}$

Pour obtenir un courant constant (indépendant de Z_c), il faut que $R_1R_3 = R_4R_2$. En posant $R_1 = R_2$ et $R_3 = R_4$, nous simplifions l'expression pour le courant :

$I = \frac{V+}{Z_c} = \frac{V_{in}}{Z_c+R_4+\frac{R_4Z_c}{R_3} - \frac{Z_cR_4(R_1+R_2)}{R_1R_3}}$

Avec les conditions $R_1 = R_2$ et $R_3 = R_4$, nous avons :
$R_1R_3 = R_4R_2$
$R_1R_4 = R_4R_2$
$R_1 = R_2$
$R_3 = R_4$

Donc, $I = \frac{V_{in}}{R_4}$, ce qui est indépendant de Z_c.

En résumé, en choisissant $R_1 = R_2$ et $R_3 = R_4$, le courant devient :
$I = \frac{V_{in}}{R_4}$

Cela garantit que le courant est constant et indépendant de l'impédance Z_c.

On observe bien la charge et la décharge sur la figure 7 en cyan.

D'après les conditions données, si $R1 = R2$ et $R3 = R4$, on peut observer la charge et la décharge de la capacité avec les relations suivantes :

1. $V_{capa} = V+$
2. $V- = \frac{V_{out}}{2} = V+ = V_{capa}$

D'où :

$$V_{out} = 2 \cdot V_{capa}$$

Ce qui signifie que la tension de sortie (V_{out}) est deux fois la tension de la capacité (V_{capa}).

Le signal triangulaire de charge et de décharge de notre capacité, que l'on peut observer en cyan sur la figure 7, ne peut pas être interprété par un microcontrôleur. Il est donc nécessaire d'utiliser un comparateur à hystérésis pour le transformer en un signal créneau de même fréquence, mais interprétable par un microcontrôleur.

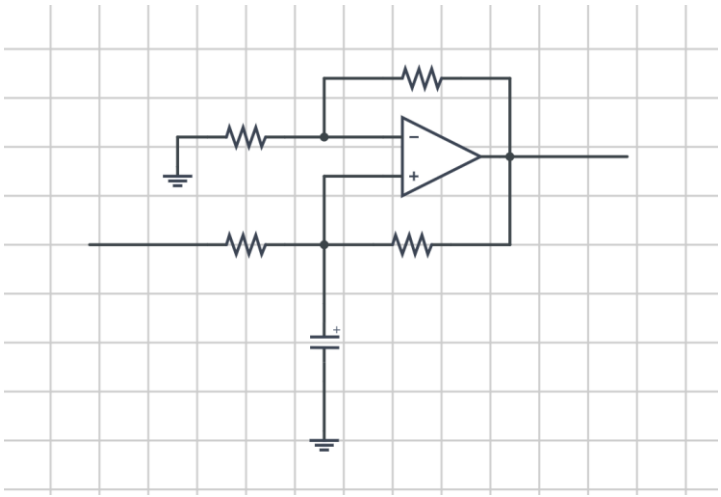

Figure 6. Comparateur à hystérésis

Les tensions de basculement données par le théorème de Millman sont alors les suivantes :

$$V_{\text{seuil}+} = V_{\max}\frac{R_6}{R_5+R_6}$$

et

$$V_{\text{seuil}-} = -V_{\max}\frac{R_6}{R_5+R_6}$$

On obtient donc un signal créneau en [-5 ; 5] Volts.

La prochaine étape consiste à boucler la sortie du comparateur à hystérésis sur l'entrée du convertisseur tension-courant. Cela crée un oscillateur, avec un signal créneau de sortie qui charge et décharge la capacité. Ainsi, nous obtenons un signal créneau à la même fréquence que le cycle de charge et de décharge de la capacité.

Il reste à rendre la simulation plus réaliste avec une alimentation unique et à centrer le signal sur 2.5V, car le microcontrôleur fonctionne entre 0V et 5V. Pour cela, on met en place une alimentation unique et on centre le signal sur 2.5V à l'aide d'un amplificateur opérationnel en montage suiveur. La simulation Partquest nous montre bien un signal créneau de 0-5V en jaune sur la figure 7.

En se branchant aux pins P1 à 04 du conditionneur, on peut avoir accès au signal triangulaire, créneau et au signal traité de l'électrode (P2) et les observer sur l'oscilloscope.

Sur la figure 6 (bis), on peut voir à gauche, le signal devenir triangulaire à cause d'un condensateur. Lorsqu'on ajoute un deuxième condensateur, un phénomène de charge et de décharge apparait lorsqu'on rapproche la main. Le système est alors plus lent.

Figure 6. (bis) Signal triangulaire à gauche et signal triangulaire en « décharge » à droite

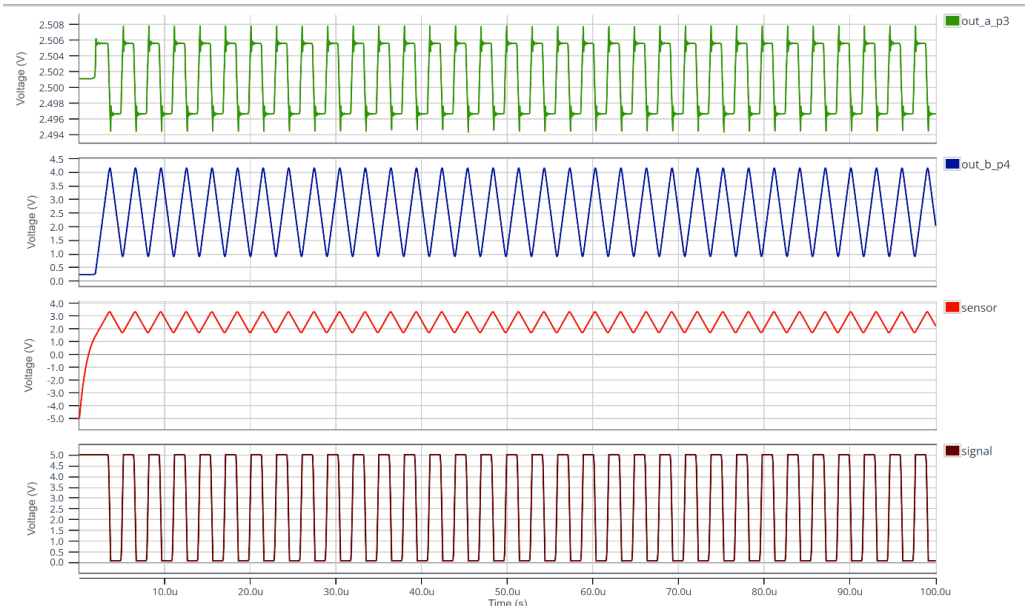

Figure 7. Résultat extrait de Partquest

Masse interne et externe

Nous allons utiliser un système de masse interne en branchant une face de l'électrode à l'alimentation et l'autre face sur la masse du conditionneur. On crée alors un champs électriques entre les deux armatures. Ce dispositif est assez limité en portée car l'objet doit rentrer dans le champs électrique pour être détecté ce qui limite son utilisation. En effet Il faut toucher une arrête du condensateur pour détecter une variation de fréquence.

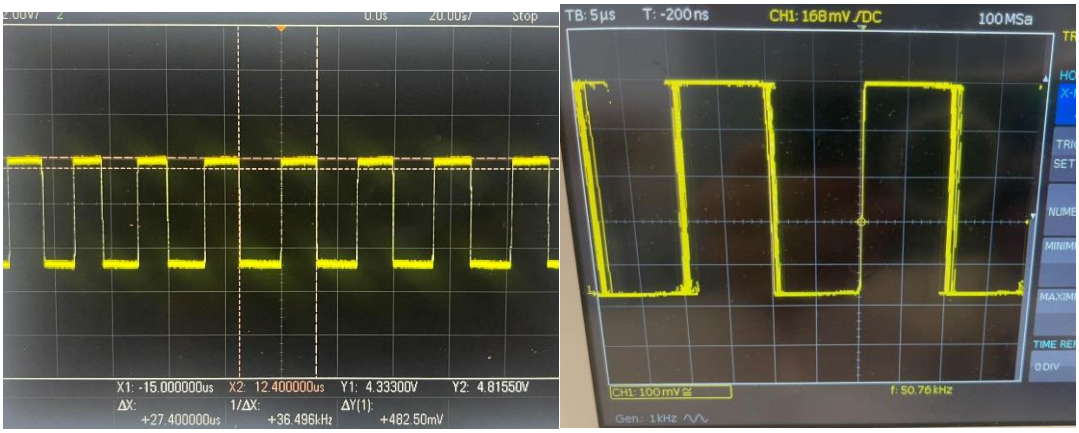

Figure 8. Masse interne : Fréquence de référence à gauche et fréquence lors du toucher de l'électrode à droite

On remarque un faible changement de 0,5 kHz lors du toucher.

Pour améliorer la détection, nous allons mettre en place une masse externe. L'électrode n'est alors connectée qu'à l'alimentation du conditionneur. La main de l'utilisateur jouera la masse (l'humain est connecté à la terre).

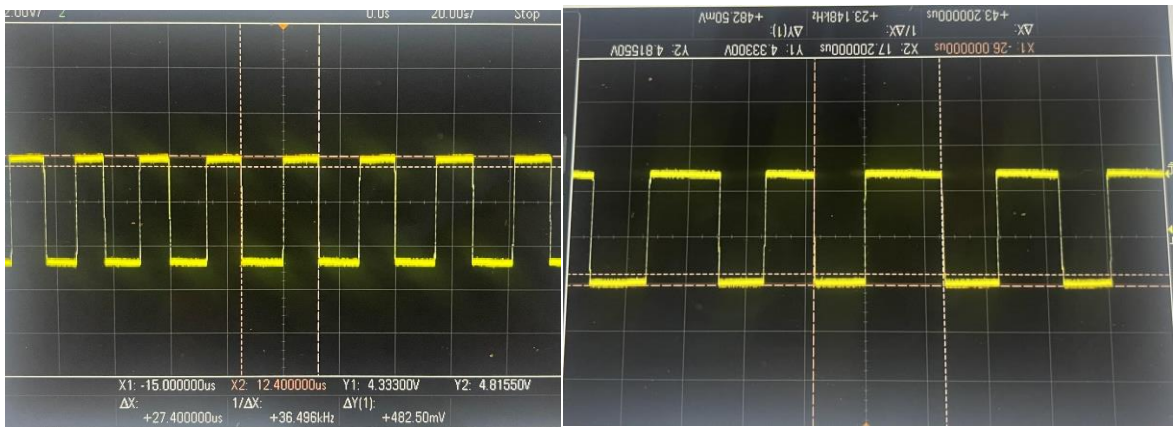

Figure 9. Masse externe : Fréquence de référence à gauche et fréquence lors du toucher de l'électrode à droite

Pour une main le changement est de 30 kHz, la précision et la portée du dispositif sont donc meilleures. On utilisera par la suite une masse externe

Design Kicab

Maintenant que nous avons modélisé notre capteur avec l'électrode reste à désigner le conditionneur. Pour cela, nous allons utiliser le logiciel Kicad.

Dans un premier temps, nous allons réaliser le schéma électrique du conditionneur et vérifier que tous les branchements sont correctes pour ensuite attribuer les empreintes des composants.

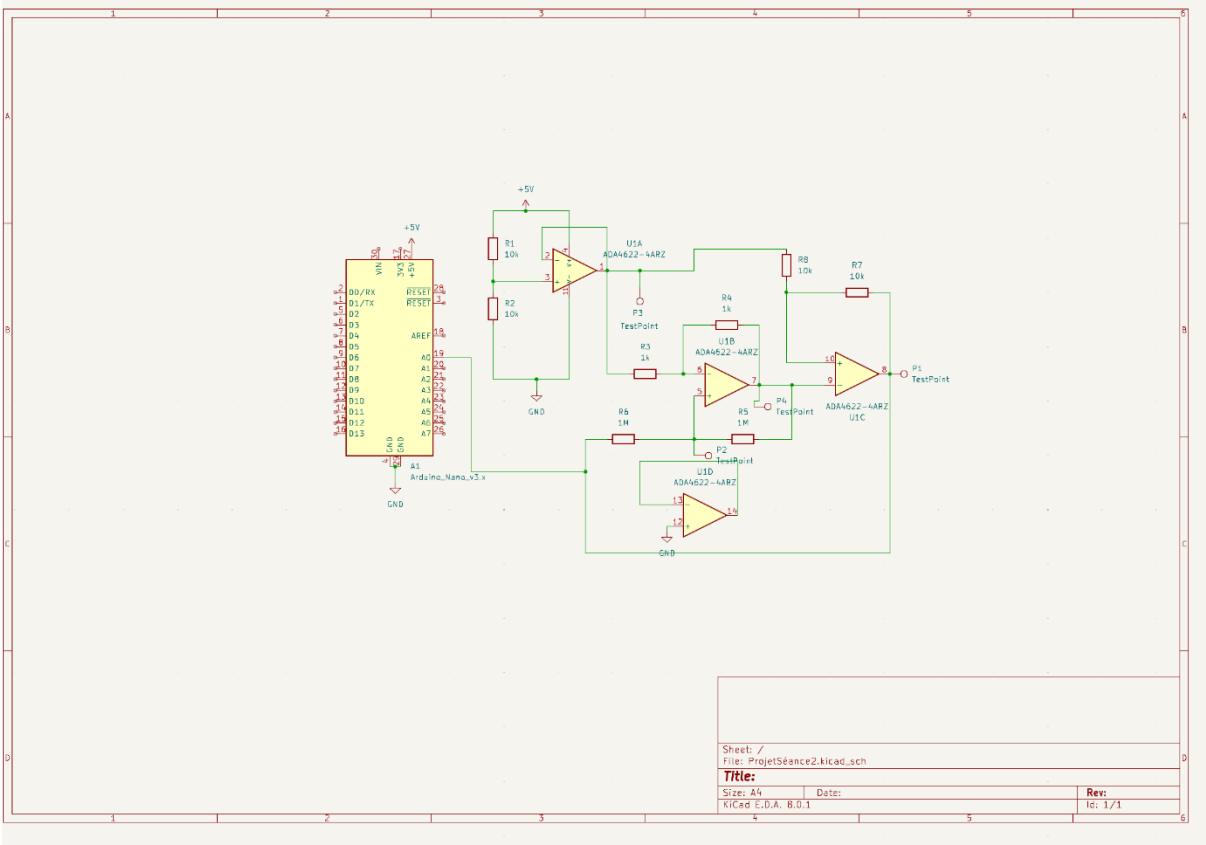

Figure 10. Schéma électrique du conditionneur extrait de Kicad

Une fois que l'étape électrique est terminée, il faut faire le routage. On débugge ensuite jusqu'à avoir 0 erreur pour ensuite générer le fichier Gerber et envoyer le conditionneur en impression. Un outil utile est le logiciel de routage automatique Freerouting qui se met en plugin externe sur Kicad. Nous l'avons surtout utilisé pour le routage du PCB.

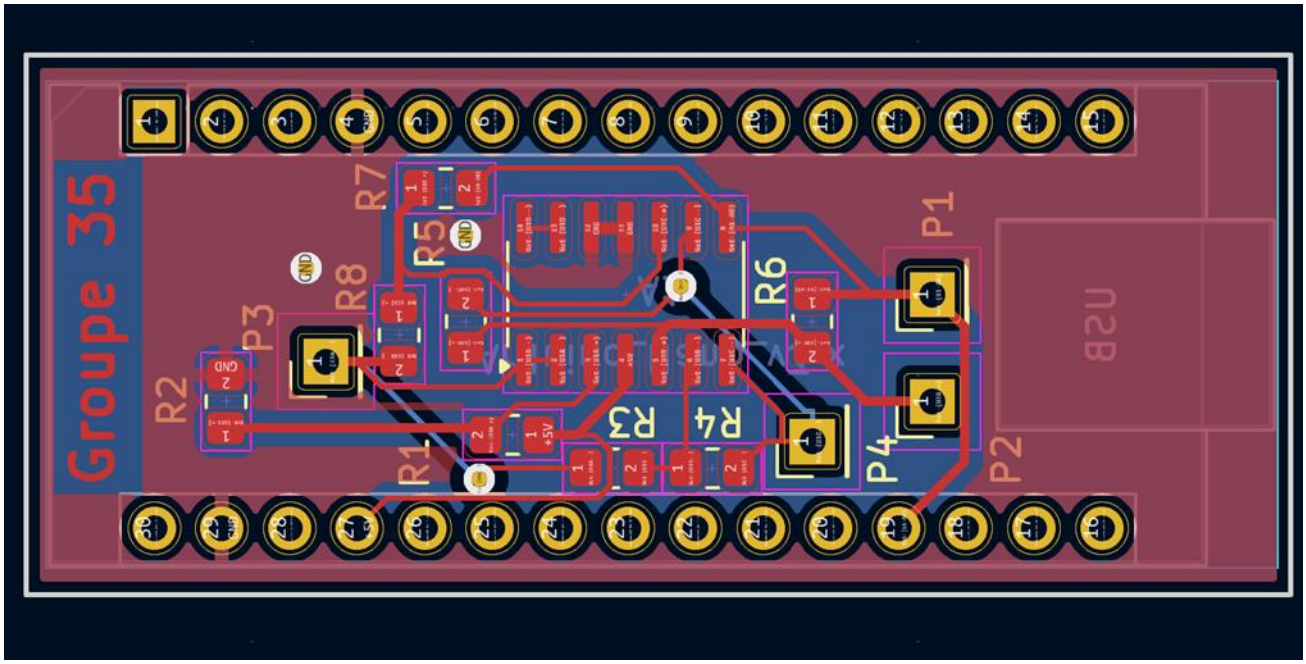

Figure 11. Représentation de l'étape de routage sous Kicad

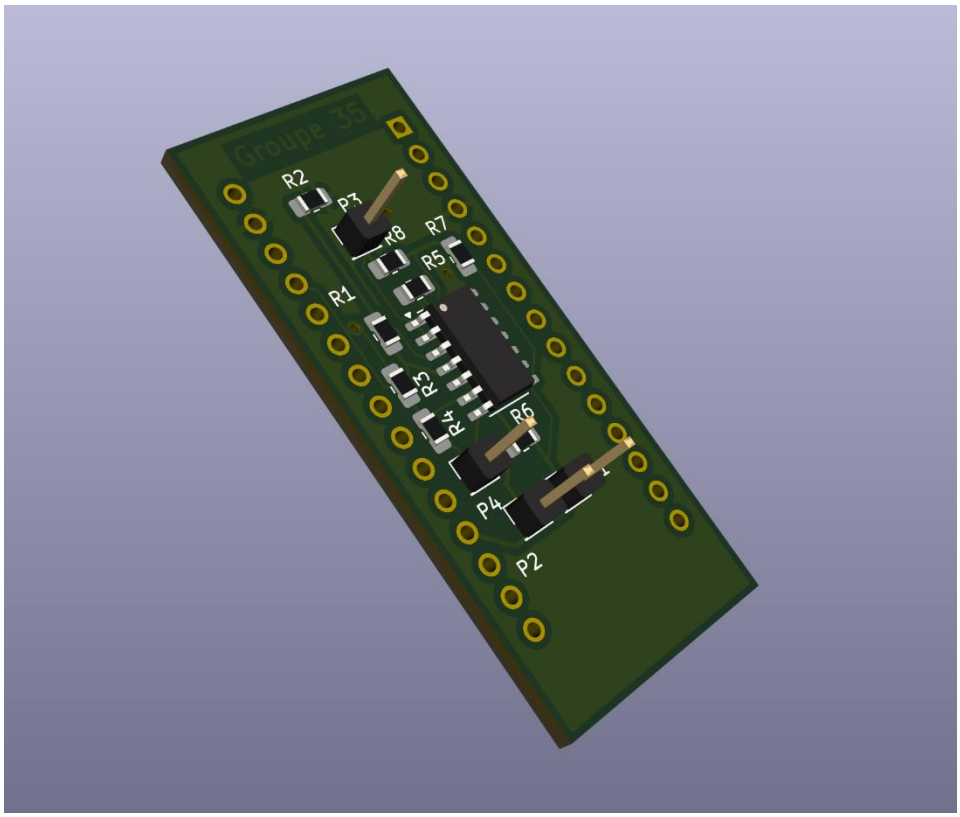

Figure 12. Modélisation 3D du conditionneur sous Kicad

Il ne reste alors plus qu'à l'imprimer et à souder des branche dessus et les résistances pour qu'il soit opérationnel.

Capteur capacitif

Nous allons avoir besoin de deux capteurs pour notre jeu. Les deux capteurs sont sensibles au même environnement, il est donc cohérent de se demander quelle distance minimale entre les capteurs permet qu'un capteur capte une présence et l'autre non ou encore quelle taille minimale du capteur permet de bien capter la présence d'une main.

Simulation de l'environnement

Dans un premier temps nous allons utiliser Comsol Multiphysics pour modéliser la situation en utilisant du FR4 pour les électrode et un milieu classique composé d'air (permittivité de 1). Nous avons fait varier la taille et la distance des électrodes, modélisées par un carré. La distance de 7,1 pouces est la plus petite distance permettant l'indépendance de la détection de présence selon le seuil fixé dans le programme (voir partie Stm32). Les capteurs font 3,2 pouces de côtés car, pour une taille plus petite, la présence d'une main n'engendrerait pas une variation capacitive assez importante pour dépasser le seuil.

A noter que les formes rondes et triangulaires ont été testées mais que la forme des électrodes n'impactait pas la variation capacitive produite.

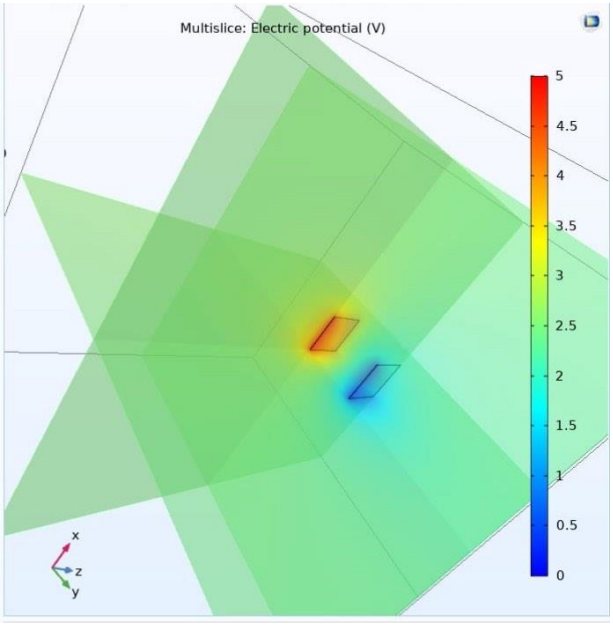

Figure 13 Modélisation des électrodes sur Comsol Multiphysics

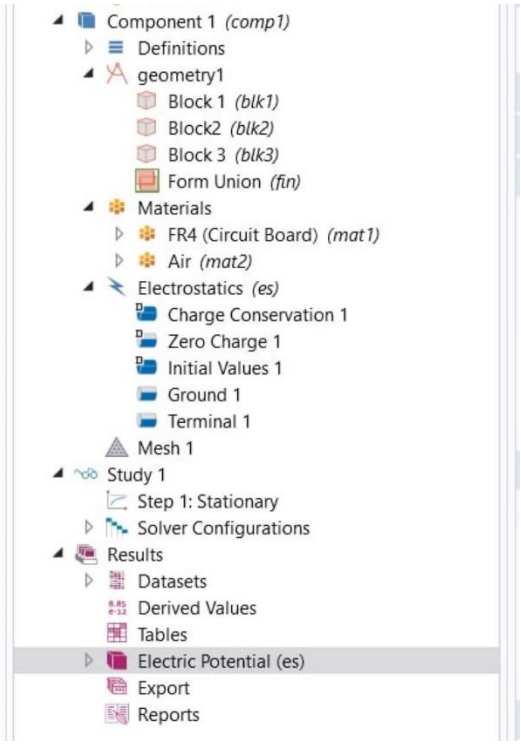

Figure 14. Choix pour la simulation sur Comsol Multiphysics

Réalisation et test des électrodes

Il reste maintenant à tester les choix découlant de la simulation sur les électrodes. On a fabriqué en FR4 les électrodes et on a soudé un fil qu'on a isolé avec du papier adhésif cuivré. On utilisera deux microcontrôleur pour le test car le PCB n'était pas encore produit à l'époque. Chaque capteur est branché sur une voie d'un oscilloscope.

Dans un premier temps, nous allons tester que lorsque les deux plaques sont collés, une perturbation commune apparaît lorsqu'une main se rapproche. C'est le cas.

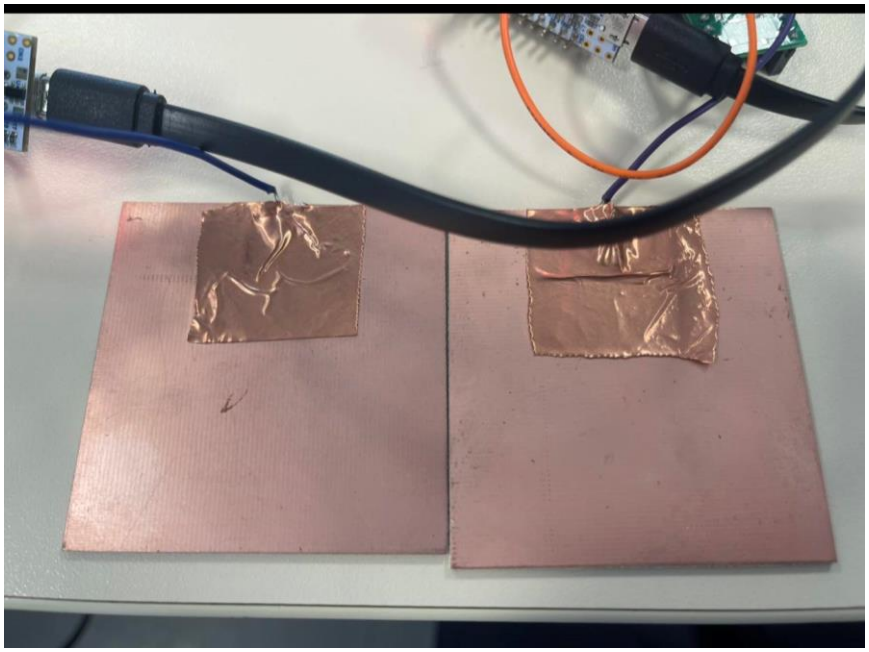

Figure 15. Positionnement 1 des électrodes

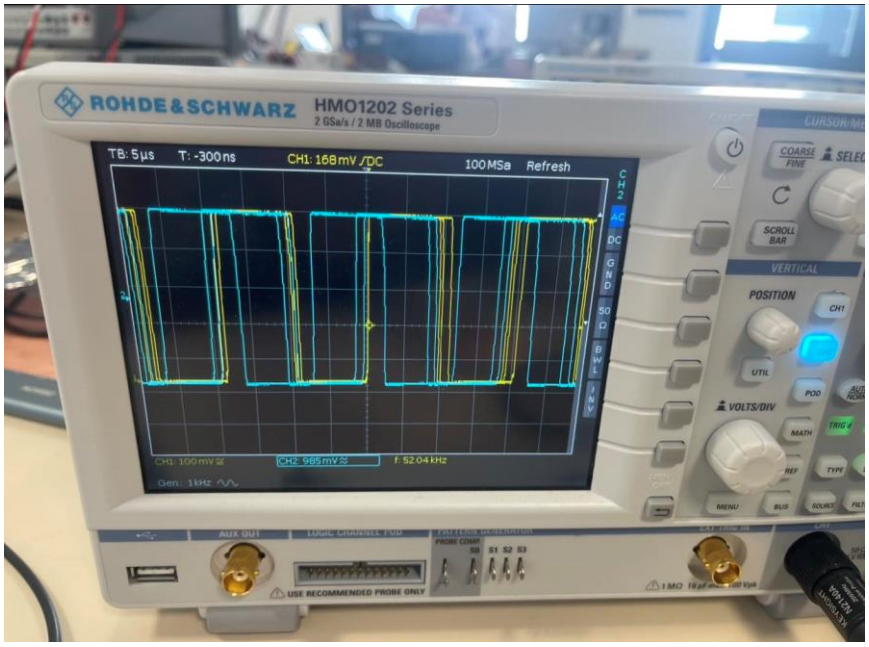

Figure 16. Résultat sur l'oscilloscope du positionnement 1

Maintenant, on éloigne les deux plaques. On se rend compte que l'indépendance des résultats des deux plaques est meilleure lorsqu'elles sont positionnées à la verticale et non à l'horizontale comme sur la figure 15. Positionnées à la verticale, on observe bien sur l'oscilloscope que lorsque l'on touche l'électrode 1, le signal de l'électrode 2 n'est pas impacté.

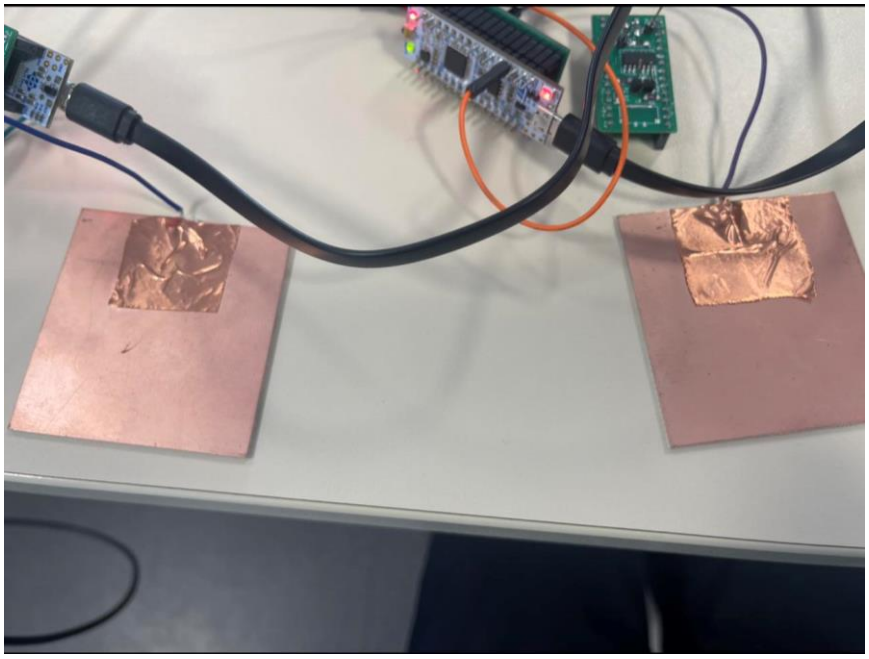

Figure 17. Positionnement horizontal des électrodes

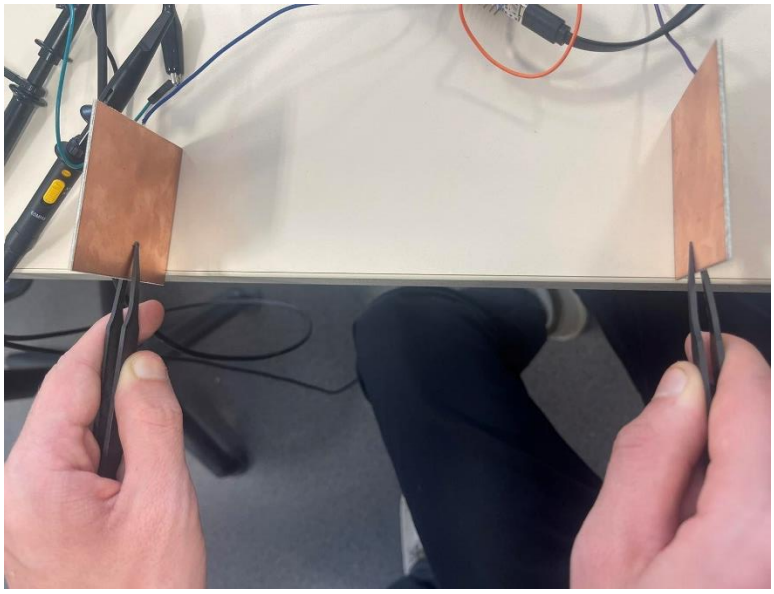

Figure 18. Positionnement vertical des électrodes (avec des pinces isolantes)

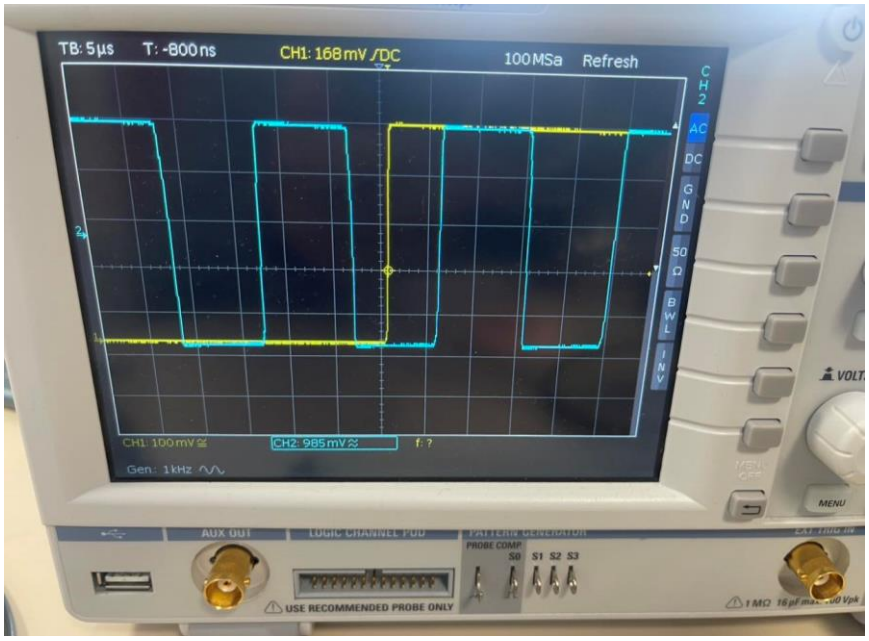

Figure 19. Résultat sur l'oscilloscope du positionnement vertical

On a donc les tailles et distances idéales de nos deux électrodes.

Réalisation du prototype

Il reste maintenant à créer le modèle de notre jeu. Il faut donc choisir les composants les plus adaptés au fonctionnement de notre jeu mais aussi aux particularité du microcontrôleur.

Choix du matériel utilisé

L'afficheur des scores sera un afficheur i2c car son fonctionnement sur 4 branches et en 3,5 V est particulièrement adapté aux pins du microcontrôleur.

Figure 20. Afficheur i2c

De même des résistances (qui seront intégrés sur le PCB) sont nécessaires pour protéger les deux diodes.

Figure 21. Diode verte

Voici un tableau plus détaillé des composants de notre projet :

Description	**Lien RS ou autre**
LED Rouge	L-53HD | LED Rouge, Traversant, 5 mm (T-1 3/4), 2,25 V | RS (rs-online.com)
LED Verte	L-53GD | LED Vert, Traversant, 5 mm (T-1 3/4), 2 V | RS (rs-online.com)
Interrupteur	https://fr.rs-online.com/web/p/interrupteurs-a-bascule/7932519
Afficheur led pixel i2c	FC1602G01-RNNYBW-66SE | Afficheur graphique LCD Fordata, LCD 2 x 16 caractères | RS (rs-online.com)
Pile 9V	

Embout pile 9V	
Conditionneur	
Résistance 250 ohm	

Une fois le code téléversé, le montage est indépendant et alimenté par une pile en 9V.

Programmation

La programmation du code du jeu a été réalisée en C avec les bibliothèques intégrées au logiciel Stm32 CubeIDE.

Schéma des pins

Dans un premier temps, le schéma de la figure 20 résume les différents branchements de la carte : 2 GPIO_OUTPUT pour les LEDS, un SDA et SCL pour l'afficheur i2c et deux TIMER pour les capteurs.

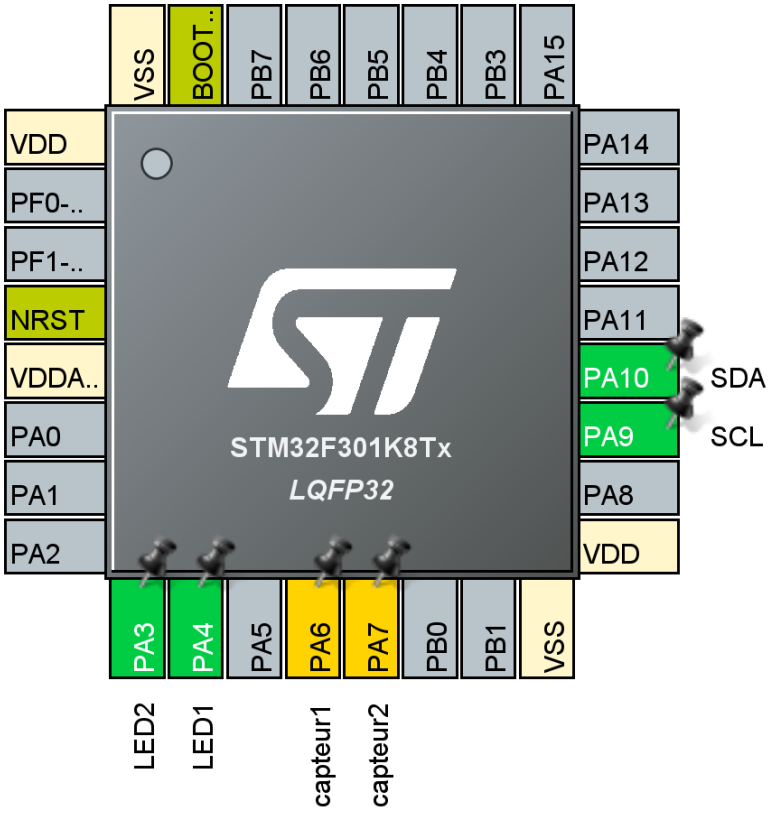

Figure 22. Schéma des pins utilisés du Microcontrôleurs

Description des étapes

Le code peut se décomposer en deux grosses parties : la réception de la fréquence des capteurs et la modélisation du jeu. Le code est fourni en annexe (Annexe 1).

La fonction HAL_TIM_IC_CaptureCallback est exécutée toutes les millisecondes. Cette fréquence de déclenchement à été établie grâce au prescaleur et au counter period . Elle vérifie d'abord si l'événement provient du timer TIM16 ou TIM17 (capteur 1 ou 2). Ensuite, elle lit les valeurs capturées par les canaux 1 des timers et calcule la différence de temps entre les deux captures pour déterminer la fréquence du mouvement détecté. Si la fréquence dépasse un seuil prédéfini, elle met à jour l'état de détection pour indiquer quel capteur a détecté un mouvement.

```c
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef* htim) {
    // Vérifie si le timer est TIM16
    if(htim->Instance == TIM16) {
        // Vérifie si c'est la première capture
        if(premier_input_1 == 0) {
            ic1_a = HAL_TIM_ReadCapturedValue(&htim16, TIM_CHANNEL_1); // Lire la première valeur capturée
            premier_input_1 = 1; // Indique que la première capture est faite
        } else {
            ic1_b = HAL_TIM_ReadCapturedValue(&htim16, TIM_CHANNEL_1); // Lire la deuxième valeur capturée
            diff1 = ic1_b - ic1_a; // Calculer la différence de temps entre les deux captures
            if (diff1 != 0) {
                frequence1 = () / diff1; // Calculer la fréquence
                if (frequence1 > seuil_detection) { // Vérifier si la fréquence dépasse le seuil de détection
                    etat_detection = 1; // Mise à jour de l'état de détection pour indiquer que TIM16 a détecté un mouvement
                }
            }
            premier_input_1 = 0; // Réinitialiser pour la prochaine capture
        }
    }
    // Vérifie si le timer est TIM17
    else if(htim->Instance == TIM17) {
        // Vérifie si c'est la première capture
        if(premier_input_2 == 0) {
            ic2_a = HAL_TIM_ReadCapturedValue(&htim17, TIM_CHANNEL_1); // Lire la première valeur capturée
            premier_input_2 = 1; // Indique que la première capture est faite
        } else {
            ic2_b = HAL_TIM_ReadCapturedValue(&htim17, TIM_CHANNEL_1); // Lire la deuxième valeur capturée
            diff2 = ic2_b - ic2_a; // Calculer la différence de temps entre les deux captures
            if (diff2 != 0) {
```

Figure 23. Code de la fonction HAL_TIM_IC_CaptureCallback

La fonction jeu gère le déroulement du jeu. Elle initialise les scores des joueurs et génère une séquence aléatoire de jeu.

```c
void jeu(void) {
    srand(time(NULL)); // Initialisation de la graine pour la génération de nombres aléatoires
    score_joueur1 = 0;
    score_joueur2 = 0;
    uint32_t start_time;
    while (score_joueur1 < 10 && score_joueur2 < 10) {
        // Allumer la LED rouge et eteinre la LED verte
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);

        // Attendre un temps aléatoire entre 3 et 6 secondes
        // et vérifier que personne ne met sa main
        random_delay = (3 + (rand() % 4))*1000;
        start_time = HAL_GetTick();
        while ((HAL_GetTick() - start_time) < random_delay) {
            if (etat_detection == 1 && deja_detecte_1 == 0) {
                score_joueur1--;
                deja_detecte_1 = 1;
            } else if (etat_detection == 2 && deja_detecte_2 == 0) {
                score_joueur2--;
                deja_detecte_2 = 1;
            }
        }

        // Eteindre la LED rouge et allumer la LED verte
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
```

Figure 24. Code de la fonction jeu

Elle allume une LED rouge et éteint une LED verte pour indiquer le début d'un tour. Elle attend un temps aléatoire entre 3 et 6 secondes, tout en vérifiant que les joueurs n'ont pas passé leur main pendant cette période. Si c'est le cas elle enlève un point. Puis la LED verte s'allume pour indiquer aux joueurs qu'ils peuvent passer leur main. Elle attend ensuite un maximum de 2 secondes pour détecter un mouvement de l'un des joueurs. Si l'un des joueurs réussit à passer sa main, son score est incrémenté et on quitte la boucle while. Enfin, la fonction vérifie s'il y a un joueur a atteint 10 points et réinitialise les scores si nécessaire.

```
// Eteindre la LED rouge et allumer la LED verte
HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);


// attendre au max 2 secondes pour détecter un mouvement
start_time = HAL_GetTick();
while ((HAL_GetTick() - start_time) < 2000) {
    if (etat_detection == 1) {
        score_joueur1++;
        break; // Sortir de la boucle si un joueur a marqué
    } else if (etat_detection == 2) {
        score_joueur2++;
        break; // Sortir de la boucle si un joueur a marqué
    }
}
```

Figure 25. Code de l'allumage des LED et de l'attente

On réalise et test le montage dans un premier temps avec une seule électrode (le joueur 2 ne fait rien) sur une board pour vérifier les branchement le fonctionnement générale avant d'imprimer le PCB.

Board de test

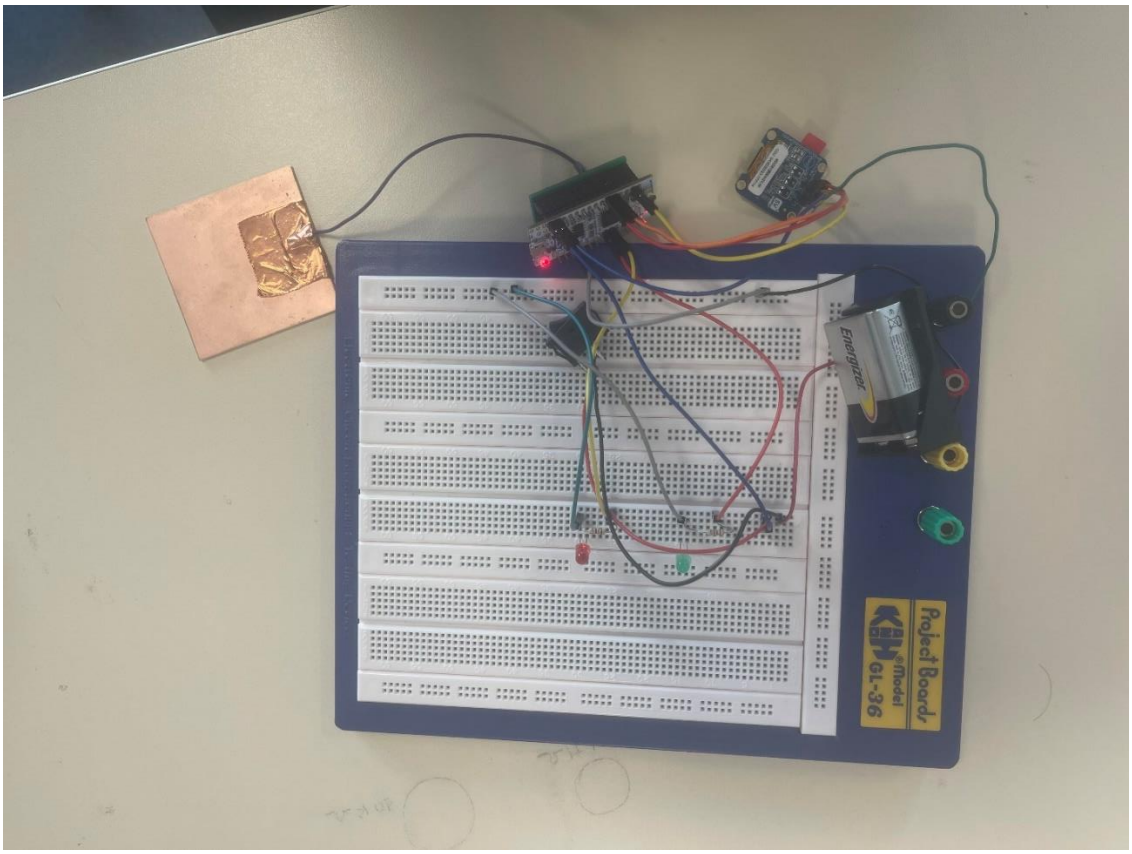

Figure 26. Montage réalisé sur le board pour 1 électrode

Le projet marche pour une électrode. L'affichage de l'afficheur doit être retravaillé et le seuil de détection du capteur abaissé (on doit presque coller la main).

Le PCB

Pour pouvoir utiliser deux conditionneur sur le même microcontrôleur sans avoir un montage brouillon, un PCB a été réalisé pour structuré la partie branchement du projet. Sa conception suit le même fonctionnement que le conditionneur sur Kicad avec la réalisation du schéma électrique, le routage et le débogage. Le logiciel Freerouting nous a aidé pour le routage.

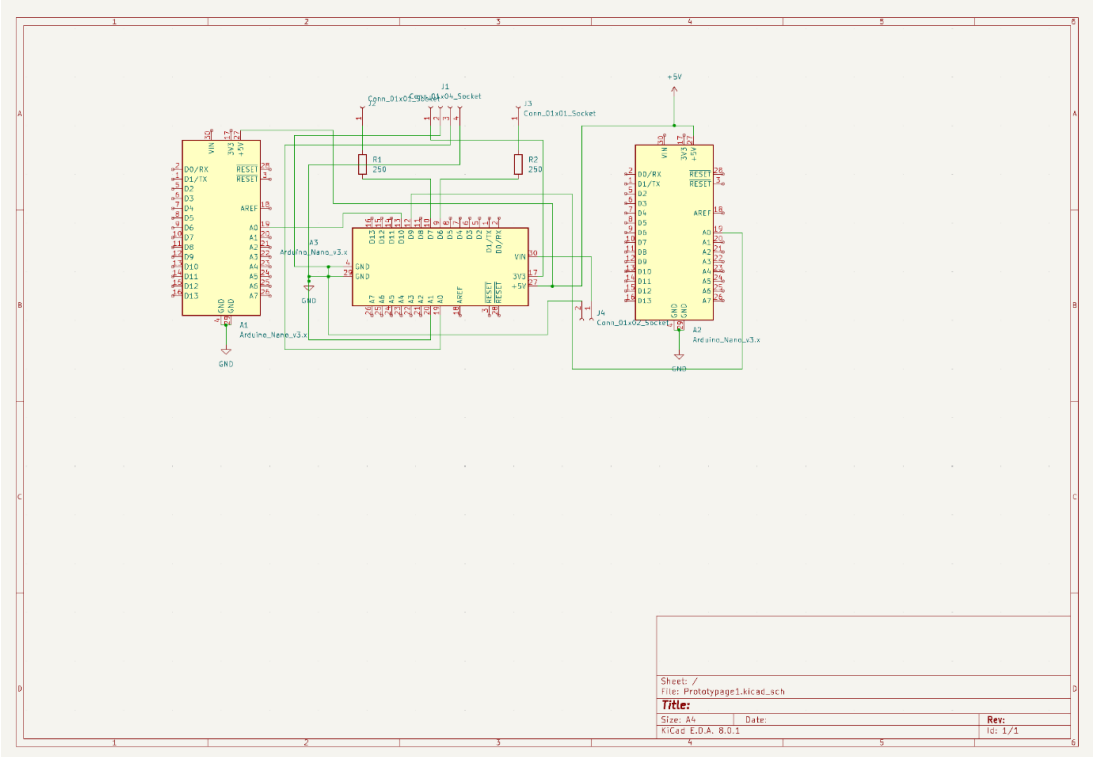

Figure 27. Schéma électrique du PCB sur Kicad

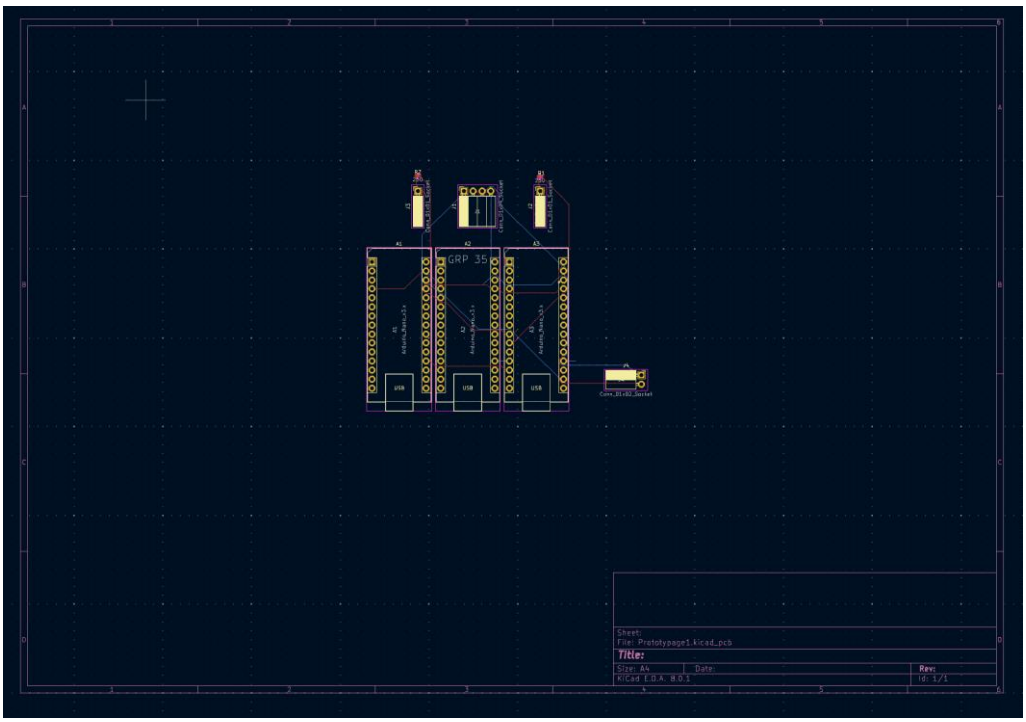

Figure 28. Modélisation de l'étape de routage du PCB sur Kicad

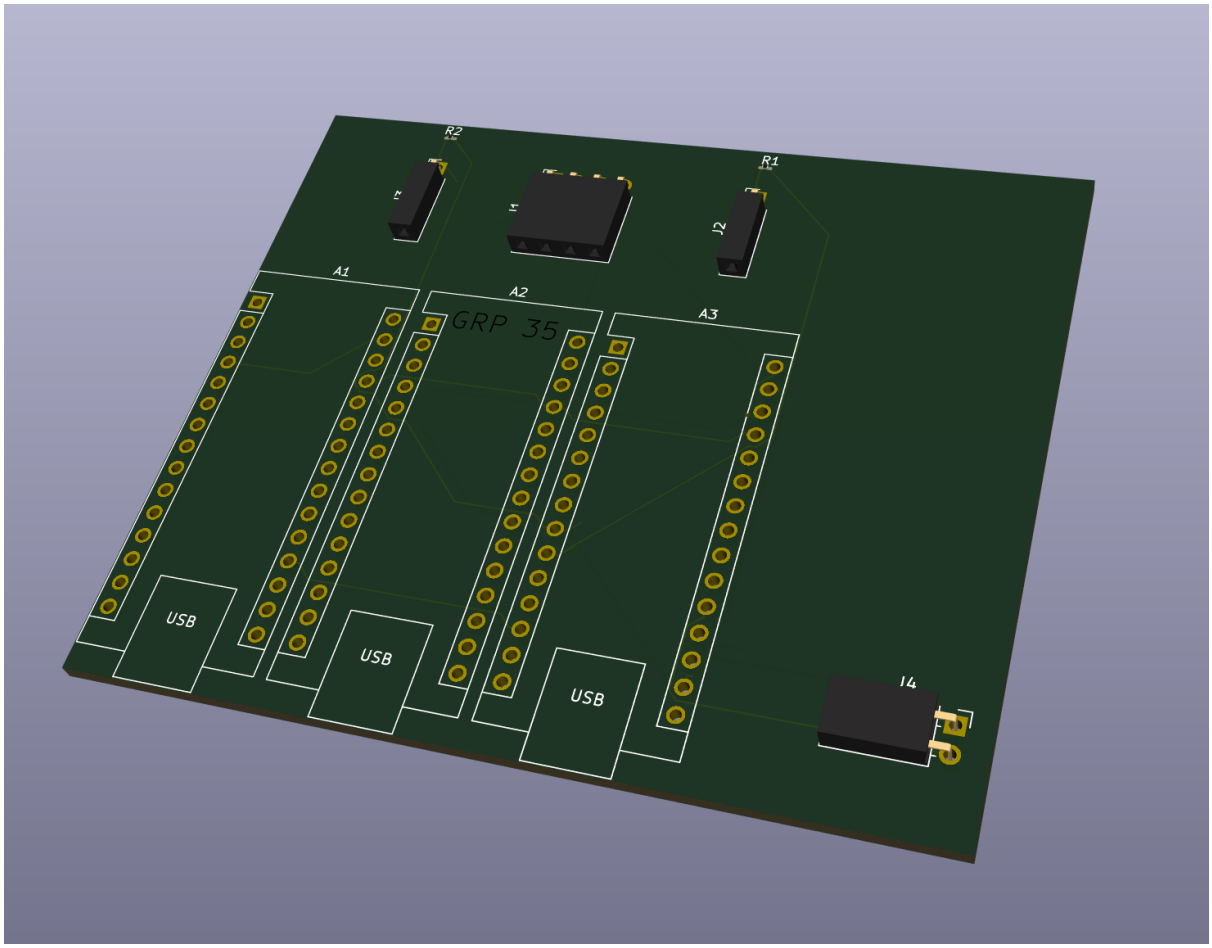

Figure 29. Modélisation 3D du PCB sur Kicad

On imprime ensuite le PCB pour ensuite souder les résistances et les branches.

Design 3D

Il reste maintenant l'étape de la modélisation de la boîte sur Autodesk Inventor. On réalise la boîte en .ipt qu'on convertit en .stl pour l'imprimer sur l'imprimante 3D.

On peut voir les emplacements pour les deux diodes, l'afficheur, l'interrupteur et les deux électrodes.

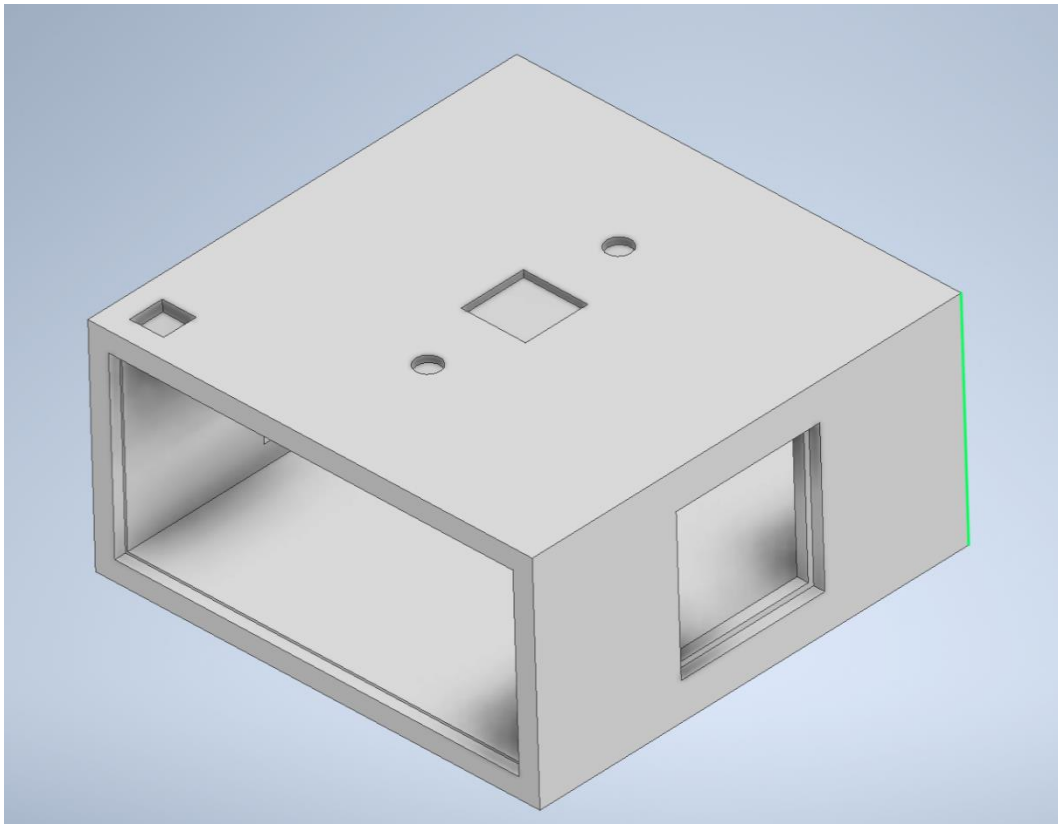

Figure 30. Simulation 3D de la boîte sur Autodesk Inventor

Cependant la boîte nécessitait 55h d'impression et 650 grammes de PLA ce qui était beaucoup trop important. Nous avons donc extrait les faces en .svg pour les imprimer en découpe laser sur du contre plaqué.

Pour un meilleur rendu, deux emplacements rectangulaires ont été intégrés pour y placer des plaques en PVC des dimensions indiquées sur la figure 31.

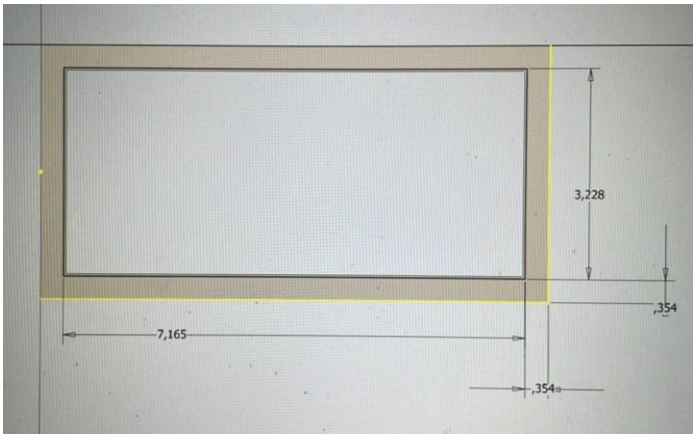

Figure 31. Dimension des vitres de la boîte

On effectue le dimensionnement sur une plaques.

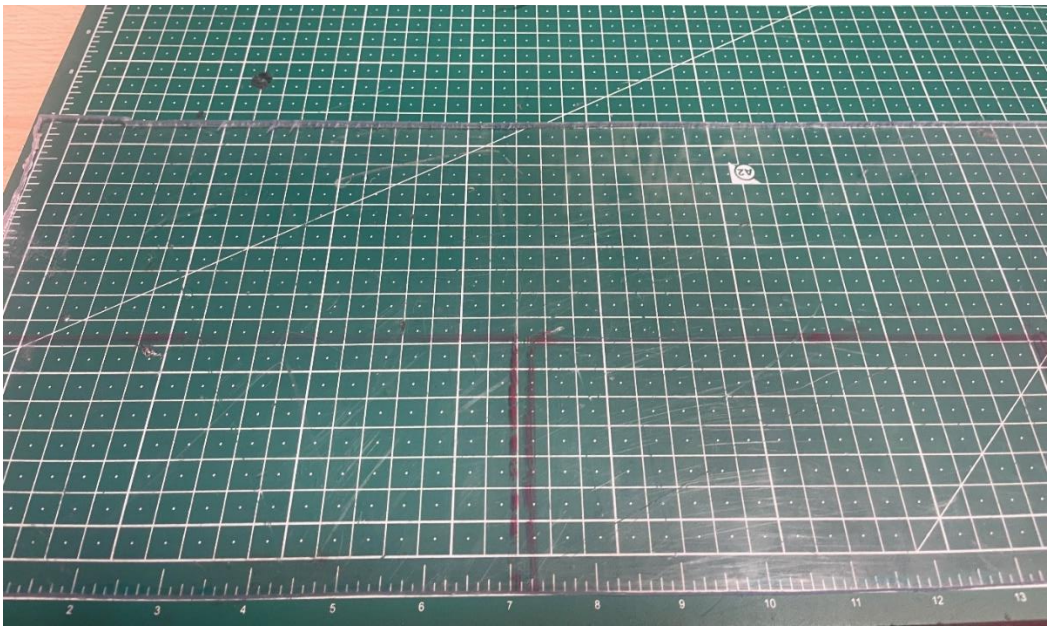

Figure 32. Mesure des dimensions des vitres

On effectue la découpe à la scie à métaux comme on peut le voir sur la figure 31.

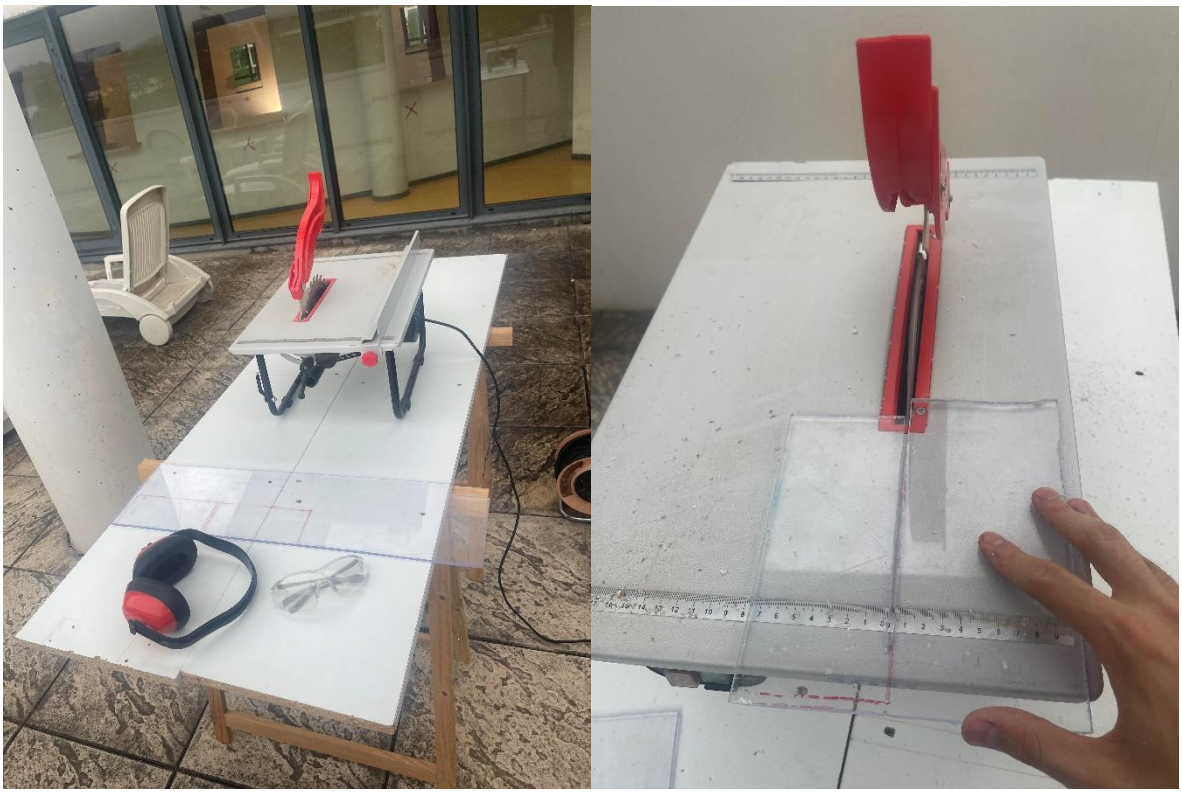

Figure 33. Découpe des vitres sur la plaque de PVC

Erreur et problème rencontré

Les erreurs sur Kicad nous ont beaucoup retardées notamment pour le PCB. Nous avons aussi soudé les mauvaises branches et abimé une piste en dessoudant. Tous ces incidents nous ont fait perdre beaucoup de temps.

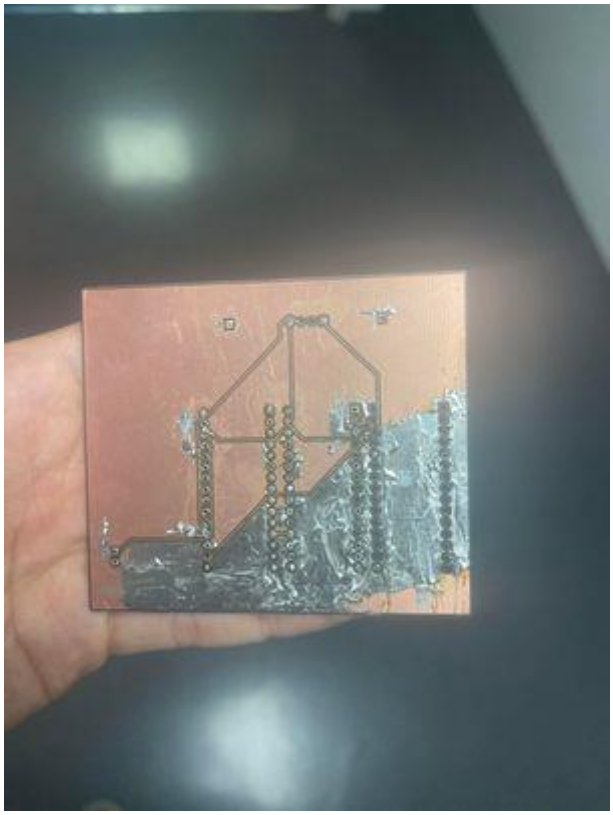

Figure 34. PCB abimé

Nous avons réalisé également un jumper avec du fil conducteur pour corriger un pin abîmé sur le deuxième PCB. La couche de cuivre est en effet très fine et donc fragile.

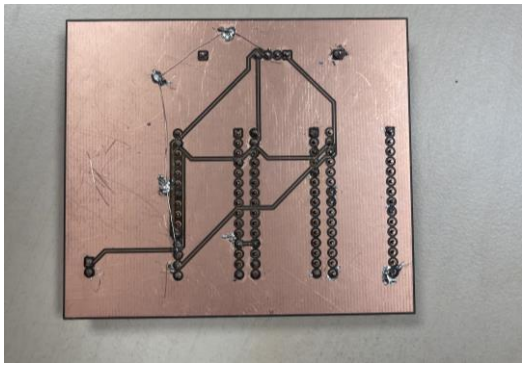

Figure 35. jumper sur le deuxième PCB

L'étape d'impression 3D a été aussi longue car le modèle était trop volumineux et donc il fallait trouver le meilleur matériau pour le réaliser.

Le débogage du code a été long pour l'afficheur et le retour des capteurs.

Durabilité

La boîte a été imprimé en bois et non en PVC dans une volonté de réduire l'impact de notre projet sur la planète. Prendre en compte l'aspect environnementale est indispensable dans notre monde actuel en tant qu'ingénieur. Ce projet tente donc de s'inscrire dans la logique des objectifs de développement durable (O.D.D.).

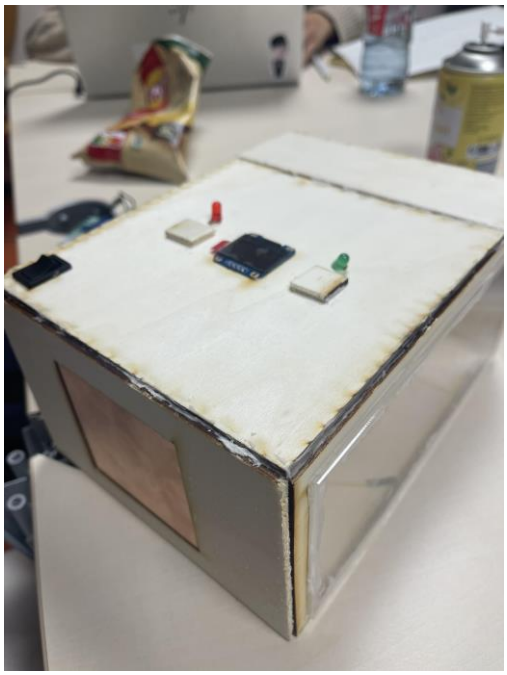

Figure 36. Projet final

Conclusion

Le projet est globalement fonctionnel et permet à deux utilisateurs de jouer sur le board. Ainsi, l'objectif de notre projet est atteint. Toutefois, notre système présente encore des limites qui offrent des pistes d'amélioration. L'affichage sur l'I2C n'est pas entièrement opérationnel et le projet fonctionne uniquement sur le board et pas sur le PCB.

De plus, la masse et le terminal pourraient être mieux isolés sur le PCB. Il serait également utile d'afficher des messages sur l'écran pour indiquer le gagnant ou lorsque l'un des joueurs enchaîne des points.

Ce projet nous a permis d'apprendre énormément sur la conception, en passant par la simulation et la modélisation avec Comsol Multiphysics, jusqu'à la réalisation concrète sur Kicad ou AutoDesk Inventor. J'ai vraiment apprécié découvrir ce mode de pensée orienté vers le prototypage et toute la logique qui en découle.

Bibliographie

Pour les timers

Getting Started with STM32G0 and STM32CubeIDE: Timer in Output Compare – EmbeddedExpertIO

Pour le code

Input capture in STM32 - CONTROLLERSTECH

How to generate a one second interrupt using an ST... - STMicroelectronics Community

What is Switch Bounce and How to Debounce – ATM | Digi-Key Electronics (youtube.com)

Durabilité

https://www.bois.com/bois/ecologie/transition/materiau-durable

Annexe

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "stdio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
```

```
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------
*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------
*/

/* USER CODE BEGIN PV */
// Déclaration des variables pour stocker les valeurs capturées, les
fréquences et l'état de détection
uint32_t ic1_a = 0, ic1_b = 0, ic2_a = 0, ic2_b = 0;
uint32_t diff1 = 0, diff2 = 0;
uint32_t frequence1 = 0, frequence2 = 0;
uint8_t premier_input_1 = 0, premier_input_2 = 0;
uint32_t seuil_detection = 1; // Seuil de détection
uint8_t etat_detection = 0; // 0: Aucun mouvement, 1: TIM16, 2: TIM17
uint8_t score_joueur1 = 0, score_joueur2 = 0;
uint8_t deja_detecte_1 = 0, deja_detecte_2 = 0; //permet de ne pas marquer
pleins de points en restant la main sur le capteur.
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------
*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------
*/
/* USER CODE BEGIN 0 */


/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */

void jeu();
int main(void)
{
  /* USER CODE BEGIN 1 */
```

```
  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------
*/

  /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM16_Init();
  MX_TIM17_Init();
  HAL_Init();

  /* USER CODE BEGIN 2 */
  HAL_TIM_Base_Start_IT(&htim6);
  HAL_TIM_IC_Start_IT(&htim16, TIM_CHANNEL_1);
  HAL_TIM_IC_Start_IT(&htim17, TIM_CHANNEL_1);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */
    jeu();
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
```

```
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL8;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef* htim) {
    // Vérifie si le timer est TIM16
    if(htim->Instance == TIM16) {
        // Vérifie si c'est la première capture
        if(premier_input_1 == 0) {
            ic1_a = HAL_TIM_ReadCapturedValue(&htim16, TIM_CHANNEL_1); // Lire
la première valeur capturée
            premier_input_1 = 1; // Indique que la première capture est faite
        } else {
            ic1_b = HAL_TIM_ReadCapturedValue(&htim16, TIM_CHANNEL_1); // Lire
la deuxième valeur capturée
            diff1 = ic1_b - ic1_a; // Calculer la différence de temps entre
les deux captures
```

```
            if (diff1 != 0) {
                frequence1 = () / diff1; // Calculer la fréquence
                if (frequence1 > seuil_detection) { // Vérifier si la
fréquence dépasse le seuil de détection
                    etat_detection = 1; // Mise à jour de l'état de détection
pour indiquer que TIM16 a détecté un mouvement
                }
            }
            premier_input_1 = 0; // Réinitialiser pour la prochaine capture
        }
    }
    // Vérifie si le timer est TIM17
    else if(htim->Instance == TIM17) {
        // Vérifie si c'est la première capture
        if(premier_input_2 == 0) {
            ic2_a = HAL_TIM_ReadCapturedValue(&htim17, TIM_CHANNEL_1); // Lire
la première valeur capturée
            premier_input_2 = 1; // Indique que la première capture est faite
        } else {
            ic2_b = HAL_TIM_ReadCapturedValue(&htim17, TIM_CHANNEL_1); // Lire
la deuxième valeur capturée
            diff2 = ic2_b - ic2_a; // Calculer la différence de temps entre
les deux captures
            if (diff2 != 0) {
                frequence2 = HAL_RCC_GetPCLK1Freq() / diff2; // Calculer la
fréquence
                if (frequence2 > seuil_detection) { // Vérifier si la
fréquence dépasse le seuil de détection
                    etat_detection = 2; // Mise à jour de l'état de détection
pour indiquer que TIM17 a détecté un mouvement
                }
            }
            premier_input_2 = 0; // Réinitialiser pour la prochaine capture
        }
    }
}


void jeu(void) {
    srand(time(NULL)); // Initialisation de la graine pour la génération de
nombres aléatoires
    score_joueur1 = 0;
    score_joueur2 = 0;
    uint32_t start_time;
    while (score_joueur1 < 10 && score_joueur2 < 10) {
        // Allumer la LED rouge et eteinre la LED verte
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
        HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
```

```
        // Attendre un temps aléatoire entre 3 et 6 secondes
        // et vérifier que personne ne met sa main
        random_delay = (3 + (rand() % 4))*1000;
        start_time = HAL_GetTick();
        while ((HAL_GetTick() - start_time) < random_delay) {
            if (etat_detection == 1 && deja_detecte_1 == 0) {
                score_joueur1--;
                deja_detecte_1 = 1;
            } else if (etat_detection == 2 && deja_detecte_2 == 0) {
                score_joueur2--;
                deja_detecte_2 = 1;
            }
        }

        // Eteindre la LED rouge et allumer la LED verte
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);


        // attendre au max 2 secondes pour détecter un mouvement
        start_time = HAL_GetTick();
        while ((HAL_GetTick() - start_time) < 2000) {
            if (etat_detection == 1) {
                score_joueur1++;
                break; // Sortir de la boucle si un joueur a marqué
            } else if (etat_detection == 2) {
                score_joueur2++;
                break; // Sortir de la boucle si un joueur a marqué
            }
        }

        // Indiquer le gagnant
        if (score_joueur1 >= 10) {
            // Indiquer que le joueur 1 a gagné (implémentation non faite)
            //reset
            score_joueur1 = 0;
            score_joueur2 = 0;
        } else if (score_joueur2 >= 10) {
            // Indiquer que le joueur 2 a gagné (implémentation non faite)
            score_joueur1 = 0;
            score_joueur2 = 0;
        }


    }
}

/* USER CODE END 4 */
/**
```

```
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state
*/
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line
number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line)
*/
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Annexe 1. Code du jeu