Lycée Alphonse BENOÎT
Cours Victor HUGO
84800 L'ISLE SUR LA SORGUE
Téléphone : 04-90-20-64-20

ACADÉMIE
D'AIX-MARSEILLE
Liberté
Égalité
Fraternité

BTS CIEL

Épreuve E6.2 – Session 2025

Dossier de revue final

Projet : **ESTOM**

Espace de **S**ensibilisation au **T**ri des **O**rdures **M**énagères



Étudiants:

- Yannis ZAPOLOWSKI (IR 1)
- Guilhem BUSSI (IR 2)
- Clément VOLLE (IR 3)
- Badr ERRAFI (ER 1)
- Nathan PASCAL (ER 2)

Table des matières

Table des matières....2
Partie commune....6
I - Expression du besoin....7
I.1 - Contexte....7
I.2 - Problématique....7
II - Répartition des Tâches....8
III - Acteurs et Utilisations Général....9
IV - Compte rendu de Réunion....11
IV.1 - Réunion du 08/01/2025....11
IV.2 - Réunion du 14/01/2025....13
IV.3 - Réunion du 17/01/2025....15
IV.4 - Réunion du 21/01/2025....17
IV.5 - Réunion du 05/02/2025....19
V - Fiches Recettes....20
VI - Fiche d'audit de sécurité....26
Partie Personnelle (Yannis ZAPOLOWSKI - IR 1) :....27
I - Objectifs....28
II - Planification....28
II.1 - Outils et méthode....28
II.2 - Horaire de travail :....28
II.3 - Séparation du travail en plusieurs parties :....29
II.4 - Résultats attendus....29
III - Déroulement du travail....29
III.1 - configuration de l'ESP-32....29
III.2 - Connexion et communication avec la SuperVision....29
III.3 - Réalisation des fonctions de la BoM....30
IV - Difficultés rencontrées et Solutions appliquées....36
V - Compétences mobilisées....36
VI - Bilan personnel....36
Partie Personnelle (Guilhem BUSSI - IR 2) :....38
I - Introduction personnelle....39
II - Planification....39
II.1 - Outils utilisés et méthodes....39
II.2 - Horaires de travail....40
III - Résultats attendus....42
IV - Déroulement du travail....43
IV.1 - Mise en place d'un environnement de développement....43
IV.2 - Création du premier script pour envoyer un message UDP au SuperViseur....45
IV.3 - Réception et interprétation des "ordres" du SuperViseur....50

IV.4 - Nouvelle carte et intégration du protocole du TCP....54
IV.5 - Fonctionnement de l'antenne de l'ESP32....60
V - Compétences mobilisées....62
VI - Difficultés rencontrées et surmontées....62
VII - Tâches externes à ma partie....63
IV - Bilan personnel....66
Partie Personnelle (Clément VOLLE - IR 3) :....67
I - Introduction personnelle....68
II - Objectifs et Exigences....68
II.1 - Objectifs fonctionnels....68
II.2 - Exigences techniques....69
III - Planification....69
III.1 - Outils et méthode....69
III.2 - Tâches planifiées....70
IV - Présentation de ma partie Applicative & Web....72
IV.1 - Généralités....72
IV.2 - Architecture technique....72
IV.3 - Sécurité et fiabilité....72
V - Déroulement du travail....73
V.0 - Le Wi-Fi : Approche Physique des Points d'Accès....73
V.1 - Itération 1 : Mise en place de l'environnement....75
V.2 - Itération 2 : Développement de l'interface Web dynamique....77
V.3 - Itération 3 : Développement de l'application et interaction avec l'interface Web....89
VI - Compétences Mobilisées....94
VII - Bilan personnel....95
Partie Personnelle (Badr ERRAFI - ER 1) :....96
I - Introduction....97
II - OBJECTIF....97
III - PLANIFICATION....97
III.1 - MATÉRIEL UTILISÉ....97
III.2 - Planification des Tâches....99
IV - RESULTATS ATTENDUS....101
V - TRAVAUX EFFECTUÉS....102
V.1 - ESSAIE EFFECTUÉS....102
V.2 - DEUXIÈME ESSAI EFFECTUÉ....103
V.3 - TROISIÈME ESSAIE EFFECTUÉS....1
V.4 - AUTRE ESSAI....1
VI - ANALYSE....1
VI.1 - AUTRE ANALYSE....1
VII - FABRICATION....1
Partie Personnelle (Nathan PASCAL - ER 2) :....1
I -Présentation simple de ma partie....1
I.1 - Présentation TOM....1

I.2 - Diagramme de Bloc Interne....1
I.3 - Diagramme de Gantt....1
I.4 - Présentation des différents outils et matériaux utilisés....1
II -Objectifs généraux....1
II.1 -Besoins et exigences fonctionnelles....1
II.2 -Contraintes techniques et matérielles....1
III -Les début du projet....1
III.1 -Tests matrice led....1
III.2 -Tests capteur de lumière....1
III.3 -Problématique posé par le spectromètre....1
IV -Les premier test après la remise en question du système de communication....1
IV.1 -Tests communication infrarouge....1
IV.2 -Tests du régulateur de tension et mesure d'autonomie....1
V -Schéma et Routage....1
V.1 -Création du schéma pour la carte Esp-32-S3-WROOM-1....1
V.2 -Création du schéma pour la carte XIAO-Esp-32-S3....1
VI -De la fabrication à la mise en service....1
VII -Conclusion....1
ANNEXES....1
Annexes Yannis ZAPOLOWSKI - IR 1....1
Annexe 1 - Script actuel des BOM:....1
Annexes Guilhem BUSSI - IR 2....1
Annexe 1 - Script actuel des TOM:....1
Annexes Clément VOLLE - IR 3....1
Annexe 1 - Fichier : gestion-de-la-partie.php....1
Annexe 2 - Fichier : parametre-du-jeu.php....1
Annexe 3 - Fichier : update_status.php....1
Annexe 4 - Fichier : update_joueurs.php....1
Annexe 5 - Fichier : update_collisions.php....1
Annexe 6 - Fichier : accueil.php....1
Annexe 7 - Fichier : get_infoBOM.php....1
Annexe 8 - Fichier : get_connexionIP.php....1
Annexe 9 - Fichier : get_timer_state.php....1
Annexe 10 - Fichier : db_connect.php....1
Annexe 11 - Fichier : updateTimer.js....1
Annexe 12 - Fichier : updateCamion.js....1
Annexe 13 - Fichier : updateConnexion.js....1
Annexe 14 - Fichier : main.cpp....1
Annexe 15 - Fichier : CApp.cpp....1
Annexe 16 - Fichier : CApp.h....1
Annexe 17 - Fichier : CDatabase.cpp....1
Annexe 18 - Fichier : CDatabase.h....1
Annexe 19 - Fichier : CConnectChecker.cpp....1

Annexe 20 - Fichier : CConnectChecker.h....1
Annexe 21 - Fichier : CTcpServer.cpp....1
Annexe 22 - Fichier : CTcpServer.h....1
Annexe 23 - Fichier : CTcpSender.cpp....1
Annexe 23 - Fichier : CTcpSender.h....1
Annexes Badr ERRAFI - ER 1....1
Annexes Nathan PASCALE - ER 2....1

Partie commune

I - Expression du besoin

I.1 - Contexte

L'agglomération de Cavaillon (LMV) nous a commandé un projet qui consiste à sensibiliser les enfants à l'importance du tri et de la récolte des déchets. On nous a donc demandé de réaliser un jeu qui consiste , grâce à une Benne à Ordure Ménagère (BOM), à la récolte des déchets situés dans différentes Tours à Ordure Ménagère (TOM), placées de façon aléatoire sur un parcours de 3mx3m, délimité par des cordes. Il y a un total de 4 types de déchets, représentés dans notre jeu par 4 couleurs différentes.

1. **Déchets plastiques** **(TOM et BOM de couleur JAUNE)**
2. **Déchets verre** **(TOM et BOM de couleur VERT)**
3. **Déchets ménager** **(TOM et BOM de couleur NOIRE)**
4. **Déchets cartons** **(TOM et BOM de couleur BLEU)**

Il y a un total de 10 points de collecte par couleur, soit un total de 40 TOM. Il doit également y avoir un affichage Web sur Tablette pour afficher l'état de la course, l'avancement des joueurs, et configurer les parties si besoin. Les BOM seront équipées de différents capteurs qui vont permettre de détecter la présence d'une TOM sur son côté (deux côtés), mais doit également pouvoir détecter les collisions réalisées sur la corde, ou entre joueurs (BOM). La partie doit durer environ 5 minutes au maximum.

Pour réaliser notre projet, nous avons donc besoin d'une Raspberry Pi, qui va servir de centre de SuperVision, le site Web, la base de données, ainsi que l'application de SuperVision qui sera hébergée dessus. Chaque BOM & TOM seront également équipées d'un micro-contrôleur équipé d'une carte Wifi pour la communication avec l'application de SuperVision. Des émetteurs infrarouges seront installés sur les TOM pour envoyer des signaux aux BOM concernant leur identifiant et couleur, celà permettra aux BOM de vérifier si la couleur correspond, et si la TOM est pleine ou vide.

Le Site Web devra afficher les informations de la course en cours, disposer un bouton de lancement, et d'une page de paramètres pour régler les parties.

Le budget alloué à notre projet est de 4 000€, et notre équipe est composée de 3 étudiants de l'option Informatique & Réseau (IR) et deux étudiants de l'option Électronique et Réseaux (ER).

I.2 - Problématique

Suite à cela, notre problématique pour ce projet sera la suivante :

Comment concevoir et réaliser un jeu physique pour apprendre aux jeunes générations le tri des ordures ?

II - Répartition des Tâches

Les tâches seront réparties comme suit pour la **partie IR** :

- Guilhem BUSSI sera chargé du programme des **TOM**, connexion au Wifi, ainsi que la communication avec l'application.

- Yannis ZAPOLOWSKI sera responsable du programme des **BOM**, connexion au Wifi, ainsi que la communication avec l'application.

- Clément VOLLE sera responsable de la création du site Web (sécurisé), ainsi que de l'application de **SuperVision** qui permet la communication entre les BOM/TOM & l'application Web et également chargé de mettre en place un point d'accès Wifi sécurisé.

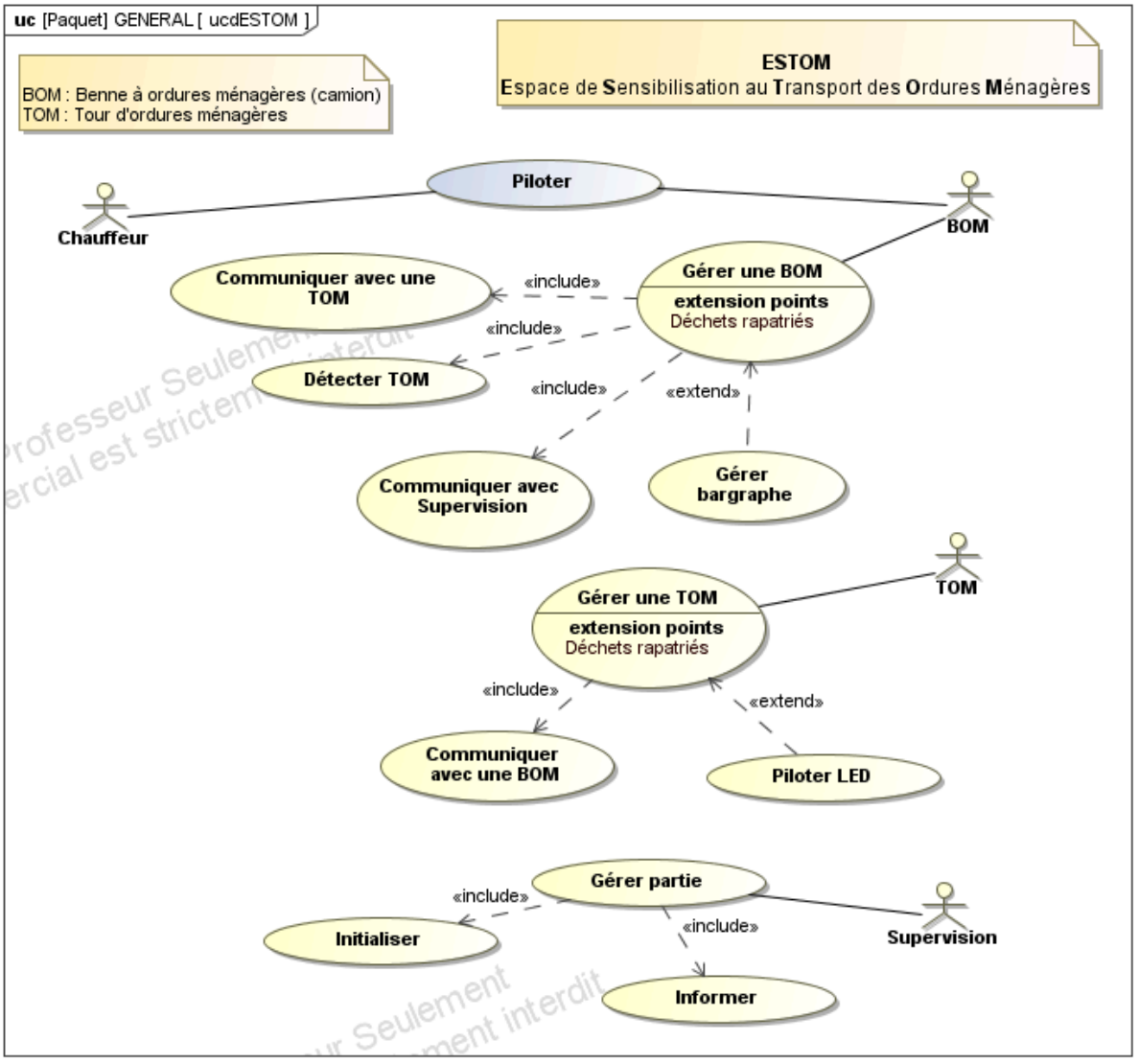
Les tâches seront réparties comme suit pour la **partie ER** :

- Badr ERRAFI sera chargé de la partie matériel et électronique des **BOM**.

- Nathan PASCAL sera lui chargé de l'électronique des **TOM**.

III - Acteurs et Utilisations Générales

Le diagramme suivant permet de mettre en évidence les acteurs du projet, ainsi que les utilisations nécessaires :



Piloter : Compte-tenu des contraintes dimensionnelles imposées par le donneur d'ordre, et dans un souci de disponibilité et de pérennité des BOM, des véhicules radiocommandés du commerce n'ont pas été retenus. Un châssis 4 roues motrices intégrant un récepteur de télécommande BLE constituera la base de la BOM, la télécommande pouvant être constituée soit d'un téléphone disposant de l'application Totem, soit d'une télécommande manuelle issue d'un projet open source. La carrosserie du véhicule sera réalisée en impression 3D.

Gérer une BOM : Logiciel intégré à la BOM permettant la détection d'une TOM compatible en fonction de la couleur de cette dernière, de communiquer avec le logiciel de superVision pour vérifier si les déchets sont présents, et de mettre à jour la supervision ainsi que gérer le degré de remplissage de la BOM si un transfert a bien eu lieu. Informer la SuperVision en cas de collision subie lors du jeu. Une barre est disposée sur la BOM pour informer le Joueur du niveau de sa progression au cours de la partie.

Gérer une TOM : Script intégré à la TOM permettant d'envoyer et recevoir des trames UDP au SuperViseur. Elle devra ensuite attendre un ordre du logiciel de SuperVision pour l'avertir d'un remplissage en cours (Clignotement de la LED) et d'un ordre d'arrêt ou de vidage de la TOM qui selon l'ordre, éteindra ou laissera allumer la TOM (Si la BOM bouge durant les 5s d'attente, alors la durée de remplissage est remise à zéro et la TOM n'est pas vidée. Si la BOM patiente bien pendant 5 secondes, alors la TOM éteindra ces Led).

Gérer la partie : Logiciel qui communique avec les BOM pour les initialiser, suivre leur évolution au cours du jeu. Un signal de fin est envoyé à toutes les BOM & les TOM pour signifier l'arrêt du jeu et le retour aux réglages initiaux. Les résultats obtenus sont disponibles en temps réel à partir d'une interface WEB. Différents paramètres pourront également être sélectionnés pour configurer la partie.

IV - Compte rendu de Réunion

IV.1 - Réunion du 08/01/2025

Compte-rendu de réunion

Détails de la réunion : *08 janvier 2025 à 8h30 jusqu'à 9h30*

Participant(s) : *VOLLE Clément, IR3, ZAPOLOWSKI Yannis, IR1, BUSSI Guilhem, IR2, ANTOINE Philippe, Professeur responsable projet*

Absent(s) : *Étudiants ER*

Ordre du jour / objectifs : *Transmettre des informations à propos du projet, répondre au question et différentes problématiques.*

Prochaine réunion : *Pas programmé*

---

1. ***Visuel de l'interface WEB***

Discussion

- *Liberté sur le visuel du site WEB tout en respectant les différentes informations « publique » nécessaire pour l'affichage des information du jeux.*

Conclusion / Action

- *Liberté sur le visuel de l'application, on va également mettre en place un onglet « administrateur » sur le site pour avoir accès au paramétrage du jeu. Cette onglet admin sera sécurisé par un mot de passe.*

2. ***Quel Type de communication?***

Discussion

- *On as discuté de la communication entre les BOM et les TOM, on a également discuter et réfléchi sur la possibilité de « vérifier » que la benne est vide ou pleine lors de la charge du camion de la benne, il serait donc d'après nous intéressant d'y mettre une connexion Wifi pour qu'elle soit contrôlé également via le SuperViseur, il est normalement prévu une connexion moins gourmande en énergie comme par exemple une connexion Bluetooth LE. Cette partie est pour le moment en cours de réflexion dans le groupe des ER.*

Conclusion / Action

- *Nous allons rediscuter du la communication entre BOM et TOM lors d'une prochaine réunion avec le groupe des ER. Les autres communications (DOM & SuperVision, WEB & SuperVison se ferons normalement par Wifi (chiffré selon ce qui existe de mieux à définir plus tard).*

3. *Quel Type de capteur ?*

Discussion

- *On as discuté des capteurs qui seront inclue dans le camion pour détecter la TOM mais également la/les collision (avec un systèm de plaque de pression), nous n'avons pas encore d'information précise sur les capteurs pour le moment.*

Conclusion / Action

- *En attente d'une prochaine réunion avec l'équipe des ER pour davantage d'information.*

4. *Configuration et paramétrage*

Discussion

- *Nous avons discuter de l'ensemble et du fonctionnement global du jeux. Nous avons donc suggérer un paramétrage préalable du jeux avant sont lancement par le Superviseur. La configuration des points, et du temps potentiellement envisageable.*

Conclusion / Action

- *Possibilité de paramétrer le temps de la partie (parcours), temps par défault :3 min.*
- *Possibilité de paramétrer le temps de réinitialisation des TOM (électroniquement ou via le site WEB), temps par défault : 5 min.*
- *Nombre de points à gagner et à perdre paramétrable, pour les points à gagner, nous avons pour idée de proposer deux options :*

➢ *Option A : La récolte de Chaque TOM donne un nombre de point FIXE et identique.*

➢ *Option B : La récolte des TOM rapporte un nombre de points progressif (1 pts pour la 1ère TOM récolter, 2pts pour la seconde, 3 pour la troisième etc.)*

Les pénalités seront paramétrable avec une valeur « FIXE ».

IV.2 - Réunion du 14/01/2025

Compte-rendu de réunion

Détails de la réunion : *14 janvier 2025 à 14h30 jusqu'à 16h*

Participant(s) : *VOLLE Clément - IR3, ZAPOLOWSKI Yannis - IR1, BUSSI Guilhem - IR2, ANTOINE Philippe - Professeur responsable projet IR, ERRAFI Badr – ER 4, PASCAL Nathan – ER 5, HORTOLLAND Christian – Professeur responsable ER*

Absent(s) : *Personnes*

Ordre du jour / objectifs : *Réunion général*

Prochaine réunion : *Pas programmé*

1. ***Discussion sur la création du dossier de Projet***

Discussion

- *Comment la mise en forme et l'organisation du dossier de projet seront constitués et créé à la fin de l'année*

Conclusion / Action

- *Le Dossier de Projet se constituera d'une page de présentation avec 5 prénoms/noms, logo du Projet, en résumé, l'essentiel. Ensuite la seconde page sera un sommaire numéroté (pour faciliter l'accès aux différentes parties). La troisième partie sera la partie commune du projet et la quatrième seront les parties individuelles de chaque élèves (le pied de page de chaque partie devra contenir le nom et prénom des personnes concerné par la partie). Et enfin les annexes*

2. ***Quel est le matériel utilisé ?***

Discussion

- *Nous avons demander quel Arduino allez être utilisé, si il possédé du Wifi intégré, quel sont les modules qui vont être utilisé pour sur la Arduino des TOM et des BOM.*

Conclusion / Action

- *La Arduino ESP 32 disposera du Wifi intégré mais aurait des problèmes à se connecter à un point d'accès Wifi d'une Raspberry Pi (à vérifier), l'environnement de développement sera à choisir (éviter Visual Studio car trop volumineux), nous avons pu voir les modules qui seront utilisé pour les LED des TOM et l'affichage LED à couleur des BOM. Les TOM disposeront de batteries. Et les connexions devront se faire si possible par Wifi*

3. *Comment se dérouleront les revues ?*

Discussion

- *On a discuté des revues de projet qui aurons lieu et la notion de l'évaluation final.*

Conclusion / Action

- *A chaque revue de projet, il faudra montrer l'avancement du projet dans sa globalité avec le dossier de projet par une présentation oral individuel de 10 minutes avec diaporama préparé puis ensuite 5 minutes de questions/réponses. La première revue étant le 21 Janvier dans peu de temps, la présence d'un diaporama ne sera pas nécessaire. Lors de la 3ème revue, la note sera secrète jusqu'à la revue de projet avec l'entreprise, car les notes seront combinées pour former la note finale de l'examen du BTS. Il faudra aussi penser à faire un diagramme SysML de classe.*

4. *Accès aux ressources*

Discussion

- *Nous avons discuter de de la gestion des ressources au seins de l'équipe.*

Conclusion / Action

- *L'équipe de projet disposera d'une ESP 32 Arduino pour effectuer les différents tests à venir sur simulateur. Les élèves d'IR et ER devront se la passer à tour de rôle selon les besoins de chacun et jusqu'à ce que nous disposions de plus d'Arduino. Nous pourrons aussi trouver des ressources dans le disque partagé appelé « Ma boîte à archives ».*

IV.3 - Réunion du 17/01/2025

Compte-rendu de réunion

Détails de la réunion : *17 janvier 2025 à 15h40 jusqu'à 16h00*

Participant(s) : *VOLLE Clément, IR3, ZAPOLOWSKI Yannis, IR1, BUSSI Guilhem, IR2*

Absent(s) : *Étudiants ER, ANTOINE Philippe, Professeur responsable projet*

Ordre du jour / objectifs : *Transmettre des informations à propos de la Base de Données, répondre au question et différentes problématiques.*

Prochaine réunion : *Pas programmé*

1. **Les *Collision***

Discussion

- *Déterminer sur la page Web a la fin de la partie le nombre de collision faite pour chaque joueur*

Conclusion / Action

- *on va également mettre en place une catégorie «Collision» dans les données sur le site pour afficher le nombre de collision qu'a effectué chaque joueur lors de la partie.*

2. **Les *Scores***

Discussion

- *On as discuté de la manière de calculé les scores et la manière de les afficher*

Conclusion / Action

- *Nous allons établie de faire les calcules du scores dans les camions et de transmettre les scores une fois calculé au Web pour les afficher dans un classement mit a jour a la fin de la partie. Nous avons émit la supposition de les afficher en temps réel mais nous remettons ce sujet à plus tard lorsque les scores seront réalisés*

3. Un *Chrono ?*

Discussion

- *On a discuté et émit l'idée de faire un chrono pour afficher le temps qu'a mit un joueur a finir si il y arrive.*

Conclusion / Action

- *Vu que l'idée d'un chrono est optionnel, nous avons décider de remettre ce sujet à plus tard pour l'avis du client a ce sujet.*

4. *Tables ToM et BoM*

Discussion

- *Nous avons mis au clair les éléments que nous souhaitions avoir dans les données du Web et les données qu'utiliseront les ToM et les BoM.*

Conclusion / Action

- *Les BoM devront être capable de changer les couleurs dans un paramétrage manuel*
- *Les ToM devront être capable de changer les couleurs dans un paramétrage manuel*
- *Les ToM devront informer le Web de leurs Statuts :*
  - *1 pour signaler qu'elles sont pleine et 0 pour signaler qu'elles sont vides.*

IV.4 - Réunion du 21/01/2025

Compte-rendu de réunion

Détails de la réunion : *21 janvier 2025 à 14h50 jusqu'à 15h20*

Participant(s) : *VOLLE Clément, IR3, ZAPOLOWSKI Yannis, IR1, BUSSI Guilhem, IR2, ANTOINE Philippe, Professeur responsable projet*

Absent(s) : *Étudiants ER*

Ordre du jour / objectifs : *Communication entre les BOM et TOM indispensable.*

Prochaine réunion : *Pas programmé*

---

1. ***Communication BOM & TOM***

Discussion

- *Problème de communication entre BOM et TOM, comment est-ce que la TOM est « identifier » par le camion ? Comment c'est-elle qu'elle est vidé ? Problème d'identification de la TOM par la BOM.*

Conclusion / Action

- *Sujet à discuter lors d'une prochaine réunion avec la partie Électronique pour mettre en place une solution au problème d'identification de la TOM par la BOM.*

2. ***Communication UDP***

Discussion

- *Est-ce que la communication UDP est validé ? Est-ce suffisant et est-ce que ça correspond au exigence ?*

Conclusion / Action

- *Confirmation de l'utilisation de l'UDP comme moyen de communication général.*

3. **Réalisation d'un model MCD pour la BDD**

Discussion

- *On nous as conseillé de réaliser un model MCD pour la base de donné suite à l'écriture des besoin réaliser dans la réunion précédente.*

Conclusion / Action

- *Réalisation d'un model MCD pour bien visualiser les besoins de la base de donné.*

IV.5 - Réunion du 05/02/2025

Compte-rendu de réunion

Détails de la réunion : *5 Février 2025 à 16h jusqu'à 17h30*

Participant(s) : *VOLLE Clément - IR3, ZAPOLOWSKI Yannis - IR1, BUSSI Guilhem - IR2, ANTOINE Philippe - Professeur responsable projet IR, ERRAFI Badr – ER 4, PASCAL Nathan – ER 5, HORTOLLAND Christian – Professeur responsable ER*

Absent(s) : *Personnes*

Ordre du jour / objectifs : *Déterminer et fixer les moyens de communications entre les TOM et BOM*

Prochaine réunion : *Pas programmé*

1. ***Discussion sur la création du dossier de Projet***

Discussion

- *Y-a t'il une communications entre le micro-contrôleur BOM et son système moteur ?*

Conclusion / Action

- *Non le micro-contrôleur BOM et le système moteur ne dispose pas de moyens de communications entre eux.*

2. ***Comment la communication entre le BOM et la TOM se fera ?***

Discussion

- *Quelle technologie sera utilisé afin que le BOM puisse reconnaître la TOM et qu'ils puissent communiquer ?*

Conclusion / Action

- *Le BOM disposera de capteurs Infra Rouge qui permettrons de capter l'émetteur Infra Rouge de la TOM.*

3. **Quel comportement la TOM aura *?***

Discussion

- *Comment faire en sorte que l'état de la TOM soit visible pour l'utilisateur ?*

V - Fiches Recettes

FICHE RECETTE, QUALIFICATION DU SYSTÈME

Nom du système :	**Projet ESTOM**	Code de la campagne de test :	001
Rédacteur de fiche recette :	Guilhem BUSSI	Date :	20/05/2025

IDENTIFICATION DU SCÉNARIO

Identification du scénario de recette	
Titre :	Fonctionnement global de la TOM (Tour à Ordures Ménagère).
Objectif du scénario :	S'assurer que la connexion TCP entre la TOM et le SuperViseur soit fonctionnelle, et que les ordres soient bien reçus, interprétés, et exécutés correctement avec un retour utilisateur.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

Microcontrôleur XIAO-ESP32S3 connecté et alimenté par liaison filaire USB-C à un **ordinateur**.
Logiciel Arduino IDE lancé et connecté à l'ESP32 avec **Moniteur série** ouvert.
Raspberry et **logiciel SuperViseur** lancés. **Tablette** en main et connectée au **site web de ESTOM**.

EXÉCUTION DU TEST

Description courte :	Vérifier la connexion automatique de la TOM au SuperViseur dans le Moniteur Série. Ensuite prendre la tablette connectée au site internet et lancer une partie. Vérifier que les ordres soient bien reçus et interprétés.
Résultats attendus :	La TOM a réussi à se connecter à la Raspberry SuperViseur et reçoit les différents ordres liés au déroulement de la partie et exécute les différentes actions nécessaires au bon déroulement du jeu.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	**Projet ESTOM**	Code de la campagne de test :	001
Rédacteur de fiche recette	Clément VOLLE	Date :	20/05/2025

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Contrôle de l'onglet « gestion de la partie »
Objectif du scénario :	Vérifier le bon fonctionnement et l'opérationnalité des différentes fonctionnalités de la page « **Gestion de la parties** » : lancer une partie, mettre en pause, reprendre, réinitialiser la partie, et ajouter des joueurs.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

S'assurer que la **Raspberry Pi** est branchée et allumée, disposer d'une **tablette** en format paysage, et avoir un accès au site web « **estom.fr** ».

EXECUTION DU TEST

Description courte :	Accéder au site web depuis le nom de domaine **estom.fr**, puis se rendre dans l'onglet « **Gestion de la partie** ». Cliquer sur « **Lancer la partie** », attendre quelques secondes, puis cliquer sur « **Mettre en pause** ». Cliquer ensuite sur « **Reprendre la partie** » puis sur « **Réinitialiser la partie** ». Cliquer sur « **Ajouter Joueurs** » pour afficher la fenêtre d'attribution des joueurs et affecter un pseudo/nom à une couleur de BOM.
Résultats attendus :	Le timer de l'UI doit rester fonctionnel et fluide lors des différentes manipulations. Vérifier sa bonne mise en pause, son démarrage/redémarrage correct, ainsi que sa bonne réinitialisation. L'ajout et l'affichage du ou des joueurs doivent correspondre à la couleur attribuée.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	**Projet ESTOM**	Code de la campagne de test :	001
Rédacteur de fiche recette	Clément VOLLE	Date :	20/05/2025

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Contrôle du bon fonctionnement de la page « Paramétrage du jeu»
Objectif du scénario :	Tester le bon fonctionnement et l'opérationnalité de la page « Paramétrage du jeu » en changeant les paramètres du jeu.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

Raspberry branchée et allumée, avoir une **tablette** (format paysage) ainsi qu'un accès au site Web « **estom.fr** ».

EXECUTION DU TEST

Description courte :	Ce connecter au point d'accès Wifi « **AP-ESTOM** » et accéder au site web sur un navigateur avec le nom de domaine **estom.fr**, rendez vous ensuite dans l'onglet « **Paramétrage du jeu** ». Modifier un des paramètres (ou chaque paramètre), puis **valider** les modifications pour les sauvegarder.
Résultats attendus :	Après rafraichissement de la page (glisser votre doigt du haut vers le bas), les paramètres modifiés ont bien été sauvegardés et sont correctement affichés dans « Paramétrage du jeu».

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	**Projet ESTOM**	Code de la campagne de test :	001
Rédacteur de fiche recette	Clément VOLLE	Date :	20/05/2025

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Vérification de l'accès au site WEB
Objectif du scénario :	Tester le bon fonctionnement et l'opérationnalité du site Web sur une Tablette au format paysage.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

Raspberry branchée et allumée, avoir une **tablette** (format paysage) connectée au **point d'accès** « **AP-ESTOM** » avec un **navigateur internet** installé.

EXECUTION DU TEST

Description courte :	S'assurer que la tablette est bien connectée au point d'accès Wi-Fi « **AP-ESTOM** » avec le mot de passe : **Projet-ESTOM_2025**. Vérifier l'accessibilité du site web sur un navigateur internet avec le domaine **estom.fr**.
Résultats attendus :	Vérifier l'affichage sécurisé (HTTPS) de la page d'accueil ESTOM ainsi que l'accessibilité des différents onglets du site web.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	**Projet ESTOM**	Code de la campagne de test :	001
Rédacteur de fiche recette	Clément VOLLE	Date :	20/05/2025

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Tester l'UI lors du vidage de la TOM
Objectif du scénario :	Vérifier que l'interface visuelle de la page "**gestion-de-la-partie**" se met bien à jour lorsqu'une TOM est **vidée** par une BOM.

CONDITIONS INITIALES NECESSAIRES POUR EFFECTUER LA RECETTE

S'assurer que la **Raspberry Pi** est branchée et allumée, disposer d'une **tablette** en format paysage, et avoir accès au site web « **estom.fr** » avec un affichage sur l'onglet « **gestion-de-la-partie** ». La **BOM** et la **TOM** doivent également être branchées et fonctionnelles, avec l'application ouverte pour la BOM.

EXECUTION DU TEST

Description courte :	Accéder au site web depuis le nom de domaine **estom.fr**, puis se rendre dans l'onglet « **Gestion de la partie** ». Brancher la BOM et la ou les TOM. Cliquer sur « **Lancer la partie** ». Lancer le vidage de la TOM via l'application de la BOM.
Résultats attendus :	Avancement de la **BOM** sur l'interface (onglet « **gestion-de-la-partie** »), **grisage** de la **TOM** et augmentation du **score** associé.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

FICHE RECETTE, QUALIFICATION DU SYSTÈME

Qualification du système :	**Projet ESTOM**	Code de la campagne de test :	001
Rédacteur de fiche recette :	Zapolowski Yannis	Date :	20/05/2025

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Fonctionnement global de la BOM (Benne à ordures Ménagère).
Objectif du scénario :	S'assurer que la connexion TCP entre la BOM et le Superviseur soit fonctionnelle, et que les ordres soit bien reçus, interprétés, et exécuté correctement avec un retour utilisateur.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

- **Microcontrôleur ESP-32 modèle LiliGo**
- **Logiciel Arduino IDE** lancé et connecté à l'ESP32 avec le Moniteur Série ouvert.
- **Raspberry et logiciel Superviseur** lancés. Tablette en main et connectée au site web de ESTOM.

EXÉCUTION DU TEST

Description courte :	Vérifier la connexion automatique de la BOM au SuperViseur dans le moniteur Série. Ensuite prendre la tablette connectée au site internet et lancer une partie. Vérifier que les ordres soient bien reçus et interprétés. Vérifier que les fonctions soient bien réalisées au cours de la partie.
Résultats attendus :	La BOM a réussi à se connecter à la Raspberry (SuperViseur) et reçoit les différents ordres liés au déroulement de la partie. La BOM exécute les différents action nécessaires au bon déroulement du jeu. Les différentes fonctions son correctement réalisées durant la partie.

BILAN

REMARQUES

CONCLUSION

VALIDE		NON VALIDE	

(Mettre une croix dans la case correspondante)

VI - Fiche d'audit de sécurité

Projet	*Auteur*	*Date*	*Référence*
Projet ESTOM	Groupe ESTOM	27/05/2025	FASxxxxxx

Désignation	*Risque*	*Solution*
Accès WEB	Intrusion, injection de requête SQL, lecture de trame	Passer le site en HTTPS Sécuriser les formulaires Sécuriser les droits des fichiers sur le serveur
Base de données	Attaque pour trouver le mot de passe admin ou autre	Mot de passe complexe de la base de données. Chiffrage complexe du mot de passe. Droit restreint pour les utilisateurs.
WIFI	Accès non autorisé au réseau, interception de trafic, attaque MITM (Man-In-The-Middle), contournement du réseau local	Utiliser WPA2 (ou WPA3 si possible) avec chiffrement AES (CCMP) Utiliser un mot de passe fort et unique Désactiver le WPS (Wi-Fi Protected Setup) pour éviter les vulnérabilités de type brute-force sur le code PIN.

Synthèse

Le site a bien été mis en HTTPS, les formulaires sont correctement sécurisés pour éviter les injections SQL.

La base de données a un mot de passe complet et sécurisé, et chaque utilisateur qui doit accéder à la base de données a des droits spécifiques.

Le point d'accès Wifi est bien configuré selon les capacités de la Raspberry Pi 3, le WPA2 avec chiffrement AES est privilégié dans notre cas (le WPA3 n'est pas disponible avec cette raspberry pi), le mot de passe Wifi est assez complexe et non d'origine, et le WPS est bien désactivé.

Partie Personnelle (Yannis ZAPOLOWSKI - IR 1) :

BOM

I - Objectifs

- Réaliser une connexion avec la SuperVision
- Réaliser les communications entre les BoM et la SuperVision
- Réaliser les différentes fonctions de la par de la BoM

II - Planification

II.1 - Outils et méthode

- outils :
  - ESP-32 modèle LiliGo
  - Le logiciel Arduino IDE 2.3.4
  - La librairie Arduinojson

II.2 - Horaire de travail :

Projet ESTOM	**162 heures?**	**Mar 07/01/25**	**Mar 20/05/25**
Découverte & analyse du projet	2 heures	Mar 07/01/25	Mar 07/01/25
Première réunion d'information	1 heure	Mer 08/01/25	Mer 08/01/25
Rédaction compte Rendu Réunion	1 heure	Mer 08/01/25	Mer 08/01/25
Réalisation du Project	1 heure	Mer 08/01/25	Mer 08/01/25
Recherche et teste simulateur ESP32 (led bar)	20 heures	Ven 10/01/25	Ven 24/01/25
réunion Ir Er 1	1 heure	Mar 14/01/25	Mar 14/01/25
réunion IR 2	0,5 heure	Ven 17/01/25	Ven 17/01/25
programation de l'ESP32-S3 avec le WEB	13 heures	Ven 24/01/25	Ven 31/01/25
programation de l'ESP32 pour la communication d'ordre	12 heures	Lun 03/02/25	Lun 24/02/25
Revue 1	1 heure	Mer 05/02/25	Mer 05/02/25
Logiciel BOM	**51 heures?**	**Lun 03/03/25**	**Mar 01/04/25**
Interface de com. Avec BOM	13 heures	Lun 10/03/25	Mar 18/03/25
réalisation des communications BOM-Supervision	13 heures	Lun 24/02/25	Lun 03/03/25
Codage du logiciel (général)	12 heures	Lun 03/03/25	Lun 10/03/25
Documentaton de rapport de revu	27 heures?	Lun 17/03/25	Mar 01/04/25
Revue 2	1 heure	Mer 02/04/25	Mer 02/04/25
Finalisation de la BOM, mise au propre	22 heures	Mer 02/04/25	Mer 30/04/25
Création Fiche de recette	3 heures	Ven 02/05/25	Ven 02/05/25
Test et synchronisation global et général avec l'équipe	5 heures	Lun 12/05/25	Mar 13/05/25
adaptation des messages de UDP en TCP	10 heures	Mer 14/05/25	Mar 20/05/25
Finalisation Application SuperVision	10 heures	Lun 05/05/25	Ven 09/05/25

II.3 - Séparation du travail en plusieurs parties :

- configuration et préparation de l'ESP-32
- La connexion à la SuperVision
- Communication avec la SuperVision
  - communication via UDP puis en TCP
- Réalisation des différentes fonctions :
  - Le remplissage
  - La gestion des collisions
  - réinitialisation de la progression à la fin de la partie
  - Interaction avec la ToM
  - Gestion des fonctions durant partie et entre deux parties

II.4 - Résultats attendus

- L'ESP-32 parvient à se connecter à la SuperVision et communique avec elle.
- La BoM parvient à réaliser les diverses fonctions prévues.
- Être capable de réaliser des tests de diverses situations tel que le vidage d'une tour, le redémarrage d'une partie ou l'envoi des informations à jour de la BoM à la Supervision

III - Déroulement du travail

III.1 - configuration de l'ESP-32

La carte utilisée pour les BoM du projet est la carte ESP-32 modèle LiliGo T-Display-S3 que la partie ER nous a offert afin de coder, programmer et réaliser notre partie de projet en attendant une décision finale sur la carte qu'il y aura dans les BoM.

J'ai installé le logiciel Arduino IDE 2.3.4 afin de pouvoir coder et téléverser des programmes sur l'ESP-32 avec la librairie Arduinojson, j'ai configuré le logiciel afin d'avoir les paramètres adéquats à la carte, pour cela j'ai sélectionné le port "Com7" et sélectionné la carte "ESP-32 modèle LiliGo T-Display-S3"

On règle la fréquence du moniteur série à 115200 afin de bien voir les résultats que l'on souhaite voir sur le Moniteur Série.

III.2 - Connexion et communication avec la SuperVision

Grâce à l'aide de Guilhem j'ai pu connecté l'ESP-32 grâce à la libraire WiFi.h et de la fonction "connectWiFi", le programme dans Setup fait que le programme est en boucle tant qu'il ne se connecte pas à la SuperVision.

Suite à cela, je me suis accordé avec Clément pour le système de communication qui sera des messages Json envoyé par TCP afin de récupérer efficacement les informations qui nous intéressent en fonction de la situation.

L'envoi de message se fait donc grâce à la fonction "SendTCPMessage"

Je me suis arrangé avec Clément pour mettre un système de statut et d'ordre :

Les statuts sont envoyé par la Bom à la SuperVision pour lui communiquer les différents états de la BoM au cours de la partie qui sont :

- Statut 1 pour l'état par défaut, rien de spécial
- Statut 2 pour informer la SuperVision du remplissage de la BoM avec une tour compatible
- Statut 3 pour informer la SuperVision que le remplissage a été réalisé
- Statut 4 pour informer la SuperVision que la BoM n'est plus devant la tour durant le vidage, le vidage de cette est donc annulé
- Statut 5 pour informer la SuperVision que la BoM a subit un choc

Les Ordres sont envoyés par la SuperVision à la BoM pour lui communiquer les différentes actions à réaliser au cours de la partie qui sont :

- Ordre 1 qui est une demande d'informations, la BoM envoie donc un message informant sa progression, son nombre de choc, son état actuel
- Ordre 2 qui est que la partie redémarre à zéro, la BoM remet ces informations de parties a 0
- Ordre 3 qui est que la tour est pleine, la BoM peut donc commencer le vidage de cette tour
- Ordre 4 est un ordre dédié uniquement à la tour
- Ordre 5 qui est que la tour est déjà vide, il n'y a donc pas de vidage
- Ordre 6 qui est que le jeu a été mis en pause, la BoM doit geler certaines fonctions
- Ordre 7 qui est que le jeu reprend doit remettre en marche certaines fonctions

III.3 - Réalisation des fonctions de la BoM

Puisque que l'ESP-32 est correctement configurée et que nous sommes capables de nous connecter à la SuperVision tout en communiquant avec elle, j'ai pu réaliser les diverses fonctions nécessaires au bon déroulement de la partie du côté des BoM.

connectWiFi :

La Fonction connectWiFi est la fonction réalisée à l'aide de Guilhem Bussi de se connecter à la SuperVision

```
void connectWiFi() {
  Serial.print("Connexion au WiFi");
  WiFi.begin(ssid, password);
  unsigned long startAttemptTime = millis();


  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 10000) {
    delay(500);
    Serial.print(".");
  }


  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnecté au WiFi");
    Serial.print("Adresse IP locale : ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nÉchec de connexion WiFi.");
```

```
  }
}
```

SendTCPMessage :

La fonction “SendTCPMessage” pour envoyer des messages à la SuperVision, elle s'exécute toutes les minutes ou lorsque la SuperVision envoie l’ordre 1.

La Fonction “ListenOrder” qui s'exécute toutes les secondes, et permet à la BoM de recevoir les ordres envoyés par la SuperVision et de réagir en fonction de quel ordre a été reçu.

CheckTour :

La fonction CheckTour est la fonction qui s'exécute en présence d’une tour à l'arrêt après avoir reçu sa couleur via infra rouge, on commence par vérifier si la couleur de tour correspond à la celle de la BoM, si ce n’est pas le cas, rien ne se passe de plus, tandis que si leurs couleurs correspondent, alors on informe la SuperVision de l’IP de la tour en question pour savoir si elle est pleine ou vide, si la Bom reçoit l’ordre 5 alors on arrête la fonction ici, si la BoM reçoit l’ordre 3 alors la tour est pleine, on lance la séquence de vidage, si la BoM n’a pas bougé pendant 5 secondes qui est la durée du vidage alors le vidage est complet, on informe la SuperVision avec le statut 3 et le joueur peut continuer la partie, si en revanche le joueur a déplacé la BoM durant le processus de vidage, alors on arrête le processus et informe la SuperVision de l'arrêt du vidage avec le statut 4.

```
void listenOrder() {
  Serial.println("début de l'écoute");
  String data = clientRecept.readStringUntil('\n');
  if (data.length() == 0) return;


  Serial.print("Trame reçue : ");
  Serial.println(data);


  StaticJsonDocument<512> doc;
  DeserializationError error = deserializeJson(doc, data);


  if (error) {
    Serial.print("Erreur JSON : ");
    Serial.println(error.f_str());
    return;
  }


  if (!doc.containsKey("Ordre")) {
    Serial.println("Trame invalide (pas de clé 'Ordre').");
```

```
    return;
  }


  int ordre = doc["Ordre"];
  Serial.print("Ordre reçu : ");
  Serial.println(ordre);


  switch (ordre) {
    case 1:
      Serial.println("Ordre 1 : Envoi des infos");
      sendTCPMessageToServer();
      break;
    case 2:
      Serial.println("Ordre 2 : partie redémarrée");
      //remise a zero de la progression et des collision
      nbrTOM = doc["nbrTOM"];
      Progress = 0.0;
      collision = 0;
      Status = 1;
      Game = 0;
      Serial.print("Nombre de tour : ");
      Serial.println(nbrTOM);
      calcul();
      sendTCPMessageToServer();
      break;
    case 3:
      Serial.println("Ordre 3 : TOM pleine, vidage !");
      stop = 0;
      Tom = 1;
      break;
    case 5:
      Serial.println("Ordre 5 : TOM déjà vide");
      Tom = 0;
      stop = 1;
      break;
    case 6:
      Serial.println("Ordre 6");
      Game = 1;
      break;
    case 7:
```

```
      Serial.println("Ordre 7");
      Game = 0;
      break;
    default:
      Serial.println("Ordre inconnu.");
      break;
  }
}
```

GetHit :

La fonction GetHit est faite actuellement pour des tests, elle est appelée de manière manuelle dans le programme pour simuler une collision, cette fonction sera davantage développée et automatisée lorsque je pourrais avoir des capteurs pour déclencher la fonction

```
void Gethit() {
  int trigger = 1;
  if (trigger == 1) {
    Serial.println("Collision subit !");
    collision = collision + 1;
    Status = 5;

    DynamicJsonDocument doc(256);

    doc["Type"] = "BOM";
    doc["Couleur"] = String(Couleur);
    doc["Progression"] = String(Progress);
    doc["NbrCollision"] = String(collision);
    doc["Status"] = String(Status);

    serializeJson(doc, clientRecept);
    clientRecept.println();

    Serial.print("Infos envoyées : ");
    serializeJson(doc, Serial);
    Serial.println();
    Status = 1;
  }
  else {
    Serial.println("aucune collision subit.");
  }
}
```

Dans la fonction, la variable trigger simule l'activation du capteur de choc ce qui fait continuer la fonction, la BoM augmente de 1 le nombre de collision et modifie le statut pour être en statut 5 (qui informe la SuperVision d'un choc comme dit précédemment). Suite à cela, la BoM envoie un message à la SuperVision au sujet de son identité (une BoM et sa couleur), ces informations de la partie (sa progression et le nombre de chocs) ainsi que son statut ici à 5.

J'ai réalisé deux fonctions, functionEverySecond et functionEveryMinute qui sont deux fonctions qui s'exécute respectivement toutes les secondes pour checker si la SuperVision a envoyé un ordre avec ListenOrder et toutes les minutes pour envoyer à la SuperVision les informations de la BoM

Simulation :

La Fonction Simulation est une fonction temporaire, elle permet de manière manuelle d'effectuer des tests au cours de la partie, sans nécessité de capteur pour exécuter les fonctions au bon moment

```
void Simulation() {
  if (Game == 1){
    Serial.println("La partie est soit en pause soit terminé !");
    return;
  }

  if (Game == 0){
      if (Serial.available() > 0) {
      char commande = Serial.read();
      switch (commande) {
        case '1':
          Gethit();
          break;
        case 'a':
          Serial.println("Tour numéro 1");
          ipToM = "192.168.0.39";
          CheckTour();
          break;
        case 'z':
          Serial.println("Tour numéro 2");
          ipToM = "192.168.0.40";
          CheckTour();
          break;
        case 'e':
          Serial.println("Tour numéro 3");
          ipToM = "192.168.0.41";
          CheckTour();
          break;
        case 'r':
```

```
            Serial.println("Tour numéro 4");
            ipToM = "192.168.0.42";
            CheckTour();
            break;
          case 't':
            Serial.println("Tour numéro 5");
            ipToM = "192.168.0.43";
            CheckTour();
            break;
          case 'y':
            Serial.println("Tour numéro 6");
            ipToM = "192.168.0.44";
            CheckTour();
            break;
          case 'u':
            Serial.println("Tour numéro 7");
            ipToM = "192.168.0.45";
            CheckTour();
            break;
          case 'i':
            Serial.println("Tour numéro 8");
            ipToM = "192.168.0.46";
            CheckTour();
            break;
          case 'o':
            Serial.println("Tour numéro 9");
            ipToM = "192.168.0.47";
            CheckTour();
            break;
          case 'p':
            Serial.println("Tour numéro 10");
            ipToM = "192.168.0.37"; //IP de la TOM gérée par Guilhem
            CheckTour();
            break;
        }
      }
    }
}
```

Pour réaliser un test en cours de partie, il faut envoyer au moniteur série les commandes représentant les touches d'un clavier azerty allant de "a" jusqu'à "p" pour tenter un vidage dans diverses tours ayant les IP allant de "192.168.0.39" jusqu'à "192.168.0.47" ainsi que l'IP "192.168.0.37" qui représante l'IP de

la TOM de Guilhem. La fonction permet aussi de simuler un choc durant la partie.

calcul :

La fonction calcul est simple, au démarrage de la partie, avec l'ordre 2 la BoM est informée du nombre de tours par couleur, cette fonction calcule le pourcentage de progression que rapporte chaque tour et évite que la Bom ne dépasse les 100% de progression.

IV - Difficultés rencontrées et Solutions appliquées

Problème : la BoM n'écoutait pas la SuperVision pour déterminer si la tour interagissait avec était pleine ou vide, elle continuait le programme de remplissage comme si la tour était pleine systématiquement.

Solution : modifier la fonction afin d'avoir une variable locale qui change si la tour est vide et qui ferme donc la fonction de remplissage (la variable change si l'ordre indiquant que la tour est vide est reçu)

Problème : La fonction Delay gèle le code durant la durée programmée (ici pendant une seconde suivit d'une minute de gel) BoM prend une minute pour informer la SuperVision de son état et pour l'écouter en cas d'ordre transmis -> Les Ordres sont donc reçus et effectués en retard sur la partie !

Solution : Remplacer les lignes de code "delay(temps en millisecondes)" par une fonction baptisée "Timer" qui permet d'exécuter l'envoi de l'état de la BoM à la SuperVision toutes les minutes et d'écouter la supervision chaque seconde tout en permettant au code de s'exécuter en arrière plan sans le geler.

Problème : Comment simuler une collision ou la détection de tour sans capteur physique pour déclencher le programme adéquat ?

Solution : Pour effectuer les simulations, j'ai réalisé une fonction temporaire nommée "simulation", cette fonction détecte si la commande "1" a été envoyé dans le Moniteur Série afin d'appeler la fonction de collision afin de simuler un choc au cours de la partie, puis si les commandes représentant les touches d'un clavier azerty allant de "a" jusqu'à "p" sont envoyé au moniteur série, alors la fonction simulation appellera la fonction de check de tour avec des IP allant respectivement de "192.168.0.39" jusqu'à "192.168.0.8"

V - Compétences mobilisées

Lors de ce projet, j'ai eu l'occasion de mobiliser les compétences suivantes :

- communication en équipe
- programmation
- gestion du temps
- gestion des tâches à réaliser

VI - Bilan personnel

Ce projet m'a permis de réaliser avec mon équipe un projet en partie autonome, sur les choix à faire pour progresser, faire face aux différents problèmes que l'on rencontre et réfléchir à comment les régler. Mais aussi le fait de communiquer en équipe pour concevoir un élément qui prend forme avec le temps, avec un planning et des exigences à remplir et respecter.

Je remercie mes enseignants de m’avoir donné l'occasion de travailler sur le projet ESTOM, je suis motivé à continuer le projet dans le but de le finaliser et de voir le produit final qu’est ce jeu au thème écologique.

Je remercie aussi mon équipe qui a rendu ce projet motivant et m’a donné envie de le continuer dans le futur, ils m’ont aidé tout au long du projet jusqu’au stade où nous sommes. Le travail a été agréable à faire en leur compagnie.

Partie Personnelle (Guilhem BUSSI - IR 2) :

TOM

I - Introduction personnelle

Durant le second semestre de ma deuxième année de BTS CIEL, nous avons dû nous mettre en équipe afin de réaliser un projet complet qui nous a été désigné. Pour cet année, nous avons dû réaliser le projet ESTOM (Espace de Sensibilisation au Tri des Ordures Ménagères) qui a était commandée par la commune de Cavaillon, dont l'objectif est de concevoir un système ludique de tri des déchets basé sur des véhicules radio télécommandés (BOM) et des tour à ordure Ménagère (TOM).

Mon rôle au sein de mon équipe est de créer un script qui sera téléversé dans les 40 cartes Arduino ESP32-S3 qui aura pour bût la réception et l'envoie de trames TCP contenant des "ordres" du SuperViseur par Wifi. Cela permettra le contrôle de la LED de la TOM afin d'avoir un retour utilisateur lumineux pour le joueur quand la TOM est allumé, éteint, en cours de connexion, ou bien en train de se faire vider.

II - Planification

II.1 - Outils utilisés et méthodes

Pour mener à bien mon rôle, j'ai décidé d'utiliser différents outils techniques afin d'arriver à mes objectifs.

- Pour le développement du script de la TOM, j'ai utilisé l'environnement de développement officiel de Arduino appelé "**Arduino IDE**", cela va me permettre de pouvoir connecter ma ESP32-S3 à mon ordinateur en connection filaire USB-C afin de pouvoir écrire, compiler et téléverser des script dans le micro-contrôleur directement. Et j'ai aussi utiliser **RealVNC Viewer** qui m'as offert avoir un accès distant à la Raspberry du SuperViseur, afin de pouvoir créer et tester les différents premiers scripts d'envoi et de réception UDP via Wifi du SuperViseur dans un environnement Linux. Mes scripts furent ensuite récupérés par l'IR 3 pour sa création d'un script d'écoute de messages UDP.

- Durant le projet j'ai dû aussi aider l'équipe en créant des ressources graphiques, que ça soit pour le site web ou bien des logos à éditer pour la documentation. Pour cela j'ai utilisé **Paint.net**, un logiciel de retouches photos gratuit. Étant un parfait compromis entre la facilité d'utilisation de Paint normal et du professionnalisme de PhotoShop, Paint.net offre une alternative gratuite permettant la création de différents calques pour l'édition d'images de toutes sortes au pixel près.

- Pour tout ce qui est documentation j'ai décidé d'utiliser **Word** de Microsoft 365 sur mon ordinateur personnel et **Microsoft Project** pour la planification des tâches sur le temps.

En ce qui concerne ma méthode de travail, j'ai opté pour une approche étape par étape.

Avançant d'objectifs en objectifs, cela me permet d'identifier les différents problèmes qui peuvent survenir au cours de la réalisation du projet, sans s'encombrer de devoir effectuer toute les tâches en simultanées et se perdre, mais cela permet aussi de garder un fil directeur avec une vision global du projet et un vision plus approfondis du fonctionnement de la machine.

II.2 - Horaires de travail

Pour la planification de mes tâches, j'ai créé sur Microsoft Project une estimation de la planification du projet, que j'ai ensuite mis à jour de semaines en semaines avec l'avancement de mon travail.

Planification prévisionnel (10 Janvier 2025):

Nom de la tâche	Durée	Début	Fin	Prédéces	Noms ressources	Calendrier des tâches
Projet ESTOM	**175,33 heures**	**Mer 08/01/25**	**Lun 02/06/25**		EC1;EC2;IR1;IR2;I	**Projet E62 Session 2025**
Réception du projet	**1 heure**	**Mer 08/01/25**	**Mer 08/01/25**			**Aucun**
Première réunion d'informatio	1 heure	Mer 08/01/25	Mer 08/01/25			Aucun
1 - Première mise en place	**23,5 heures**	**Mer 08/01/25**	**Ven 24/01/25**	**2**	**IR2**	**Aucun**
Création du logo	20 minutes	Mer 08/01/25	Mer 08/01/25	3		Aucun
Création du google drive	10 minutes	Mer 08/01/25	Mer 08/01/25	5		Aucun
Création du Microsoft Project	1 heure	Mer 08/01/25	Mer 08/01/25	6		Aucun
Mise en oeuvre de l'environne	15 heures	Mer 08/01/25	Lun 20/01/25	7		Aucun
Mise en place de la communica	7 heures	Lun 20/01/25	Ven 24/01/25	8		Aucun
Revue 1	1 heure	Ven 24/01/25	Lun 27/01/25	9		Aucun
2 - Configuration physique	**20 heures**	**Lun 27/01/25**	**Ven 07/02/25**	**4**	**IR2**	**Aucun**
Configuration de la carte micro	20 heures	Lun 27/01/25	Ven 07/02/25	10		Aucun
Revue 2	1 heure	Mer 05/03/25	Mer 05/03/25	12		Aucun
3 - Configuration logiciel	**40 heures**	**Mer 05/03/25**	**Lun 31/03/25**	**11**	**IR2**	**Aucun**
Codage du logiciel de la TOM	40 heures	Mer 05/03/25	Lun 31/03/25	13		Aucun
Revue 3	1 heure	Mer 23/04/25	Mer 23/04/25	15		Aucun
4 - Finalisation du projet	**5 heures**	**Mer 23/04/25**	**Ven 25/04/25**	**14**	**IR2**	**Aucun**
Fiches de recette de la TOM	2 heures	Mer 23/04/25	Mer 23/04/25	16		Aucun
Création du manuel utilisateur	3 heures	Ven 25/04/25	Ven 25/04/25	18		Aucun
Remise du projet	20 minutes	Lun 02/06/25	Lun 02/06/25	19		Aucun
Revue finale	**1 heure**	**Lun 02/06/25**	**Lun 02/06/25**	**20**		**Aucun**

Planification actuelle (28 Mai 2025):

Projet ESTOM	203 heures?	Mar 07/01/25	Jeu 12/06/25		EC1;EC2;IR1;IR2;I
Réception du projet	**1 heure**	**Mer 08/01/25**	**Mer 08/01/25**		
Première réunion d'information	1 heure	Mer 08/01/25	Mer 08/01/25		
1 - Première mise en place	**53 heures**	**Mer 08/01/25**	**Ven 07/02/25**	**2**	**IR2**
Création du logo	20 minutes	Mer 08/01/25	Mer 08/01/25	3	
Création du google drive	10 minutes	Mer 08/01/25	Mer 08/01/25	5	
Création du Microsoft Project	1 heure	Mer 08/01/25	Mer 08/01/25	6	
Mise en oeuvre de l'environnement de développement	**7,67 heures**	**Lun 13/01/25**	**Ven 17/01/25**	**7**	
Recherche d'un logiciel pour développer le programme	1 heure	Lun 13/01/25	Lun 13/01/25	7	
Recherche sur les connection UDP via Wifi et recherche d'un scri	30 minutes	Mar 14/01/25	Mar 14/01/25	9	
Brainsorming pour le design du site internet	15 minutes	Mar 14/01/25	Mar 14/01/25	10	
Création de différents logo pour le site	30 minutes	Mar 14/01/25	Mar 14/01/25	11	
Réunion entre IR et ER pour parler des termes du projet	1,5 heures	Mar 14/01/25	Mar 14/01/25	12	EC1;EC2;IR1;IR2;I
Rédaction du rapport de réunion	30 minutes	Mer 15/01/25	Mer 15/01/25	13	
Finalisation de la mise en place de l'environement de développ	40 minutes	Ven 17/01/25	Ven 17/01/25	14	
Mise en place de la communication avec la BOM	**31,83 heures**	**Ven 17/01/25**	**Mar 04/02/25**	**8**	
Création d'un scrip de communication UDP sur la Arduino pour n	1 heure	Ven 17/01/25	Ven 17/01/25	15	
Réunion entre IR pour parler de la base de donnée du jeu	10 minutes	Ven 17/01/25	Ven 17/01/25	17	IR1;IR2;IR3
Création de script pour écoute de messages UDP sur la Raspberr	1 heure	Ven 17/01/25	Ven 17/01/25	18	
Amélioration des envoies de données UDP vers Raspberry	4 heures	Lun 20/01/25	Mar 21/01/25	19	
Formatage des données UDP reçues sur la Raspberry et envoye	6 heures	Mar 21/01/25	Mer 22/01/25	20	
Finalisation des scripts de communication entre Rapsberry et Ar	5 heures	Mer 22/01/25	Ven 24/01/25	21	
Différentes améliorations esthétiques du site internet	5 heures	Lun 27/01/25	Mar 28/01/25	22	
Implémentation de "Grid" sur la page de paramètre et création/	3 heures	Mer 29/01/25	Mer 29/01/25	23	
Remaniment graphique de la page "Gestion des parties" en emp	5 heures	Ven 31/01/25	Lun 03/02/25	24	
Réunion sur les moyens de communications entre la BOM, TOM	1,5 heures	Mar 04/02/25	Mar 04/02/25	25	EC1;EC2;IR1;IR2;I
Revue 1	1 heure	Mar 04/02/25	Mar 04/02/25	16	
Rédaction du rapport de réunion	2 heures	Mer 05/02/25	Mer 05/02/25	27	
Création d'éléments de design pour le site	3 heures	Ven 07/02/25	Ven 07/02/25	28	
2 - Configuration physique	**20 heures**	**Ven 07/03/25**	**Mer 19/03/25**	**4**	**IR2**
Configuration de la carte microcontrôleur	**20 heures**	**Ven 07/03/25**	**Mer 19/03/25**	**29**	
Obtention d'une nouvelle Carte et installation du script pour réception d'ordres	20 heures	Ven 07/03/25	Mer 19/03/25	29	
Création du dossier de projet pour Revue 2	8 heures	Mar 25/03/25	Ven 28/03/25	32	
Préparation et passage à l'oral de revue de projet 2	1 heure	Mer 02/04/25	Mer 02/04/25	33	
Revue 2	0,5 heure	Ven 04/04/25	Ven 04/04/25	34	
4 - Finalisation du projet	**44 heures?**	**Mar 22/04/25**	**Lun 19/05/25**	**35**	**IR2**
Application des conseils professeur pour la revue de projet suivante	9 heures?	Mar 22/04/25	Ven 25/04/25	35	
Adaptation du script de TOM de UDP à TCP	22 heures?	Lun 28/04/25	Ven 09/05/25	37	
Installation du script sur la nouvelle carte et correction de problème Wifi	5 heures?	Lun 12/05/25	Mar 13/05/25	38	
Fiches de recette de la TOM	1 heure	Mer 14/05/25	Mer 14/05/25	39	
Création du dossier de projet pour Revue 3	6 heures?	Mer 14/05/25	Lun 19/05/25	40	
Préparation du passage à la Revue 3	1 heure	Lun 19/05/25	Lun 19/05/25	41	
Revue 3	0,5 heure	Mar 20/05/25	Mar 20/05/25	42	
Création du manuel utilisateur	8 heures	Mer 21/05/25	Lun 26/05/25	43	
Finalisation du dossier de projet	8 heures?	Mar 27/05/25	Lun 02/06/25	44	
Remise du projet	1 heure	Lun 02/06/25	Lun 02/06/25	44	
Peaufinement des derniers détails	9 heures?	Mar 07/01/25	Ven 10/01/25		
Revue finale	**16 heures**	**Mar 10/06/25**	**Jeu 12/06/25**		

III - Résultats attendus

Pour pouvoir affirmer le bon fonctionnement de la ESP32-S3, différents critère doivent être atteints:

- **Communication entre la TOM et le SuperViseur,** via connexion Wifi et envoi/réceptions de trames UDP pour la réception d'ordres provenant du SuperViseur et l'envoie des différentes données stockées localement sur la ESP32.

- **Contrôle de la LED** externe connectée au microcontrôleur via le script afin d'avoir un retour utilisateur au moment des différents état de la tour (éteinte quand vide, clignotante quand se fait vider par la BOM ou bien au moment de la connexion, allumée quand pleine).

  Cet objectif est en étroite collaboration avec les membres ER 1 et ER 2 de l'équipe, afin de savoir comment implémenter cette fonctionnalité au sein du script.

- **Interprétation des ordres reçus** du SuperViseur et les utiliser dans le script afin qu'ils actionnent les différentes actions de la TOM (allumer/éteindre la LED).

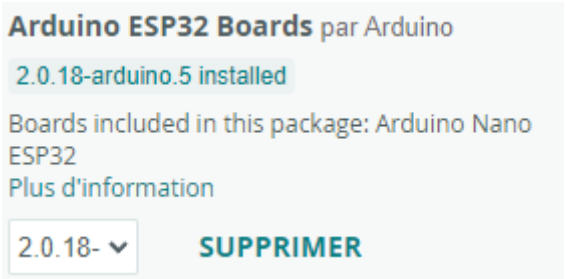
- **Changement de couleur** via le switch sur le TOM qui permettra que la TOM puissent devnir n'importe quelle couleur au sein du jeu durant la phase de préparation.

IV - Déroulement du travail

IV.1 - Mise en place d'un environnement de développement

Au tout début du projet, on m'a donné un microcontrôleur de la marque Arduino de modèle ESP32-S3. On utilisera précisément ce microcontrôleur car il est doté d'un module Wifi intégré qui permettra aux TOM de pouvoir se connecter en Wifi au SuperViseur.

Après avoir branché l'ESP en filaire avec un câble USB-C sur mon ordinateur, j'ai dû chercher un moyen pour téléverser des scripts à l'intérieur de la carte. Suite à plusieurs lectures de documentation, j'ai trouvé qu'on pouvait utiliser un logiciel d'environnement de développement appelé Arduino IDE (version 2.3.4). Après l'avoir installé, j'ai mis en place dans le Gestionnaire de Cartes, le paquet de cartes "Arduino ESP32 Boards" afin que le logiciel puisse communiquer avec la carte.



Ensuite, j'ai sélectionné le module "**ESP32-S3 Dev Module**" pour commencer à coder le script qui sera téléversé sur la carte.

Au départ pour tester toute les fonctionnalités de la carte, j'ai créé un script très simple:

```
void setup() {
  delay(500);
  Serial.begin(115200);
  delay(500);
  Serial.println("Test de communication série !");
}


void loop() {
  Serial.println("Bonjour depuis l'ESP32-S3 !");
  delay(500);
}
```

Voici comment cela fonctionne:

La fonction **setup()** est la première fonction du script à s'exécuter au lancement de la carte:

- Un délai initial de **500 ms** est ajouté pour éviter des problèmes liés à l'initialisation rapide de l'ESP32-S3.
- La communication est initiée avec une vitesse de **115200 bauds** (vitesse de transmission des données qui sera sélectionnée dans le Moniteur série).
- Une autre pause de **500 ms** est appliquée après l'initialisation.
- Un premier message "**Test de communication série !**" est envoyé sur le moniteur série.

Ensuite, la fonction **loop()** va s'exécuter en boucle :

- À chaque itération, la carte envoie le message "**Bonjour depuis l'ESP32-S3 !**" sur le port série.
- Ensuite, un **délai de 500 ms** est ajouté avant la prochaine itération, évitant ainsi une surcharge du port série.

Ce petit script permet de tester le bon fonctionnement de la communication série et que le script s'exécute bien une fois téléversé dans la carte.

IV.2 - Création du premier script pour envoyer un message UDP au SuperViseur

Ensuite, j'ai discuté avec l'IR 3 afin de convenir à un mode de communication. Nous avons donc décidé d'utiliser la Point Wifi créé par la Raspberry afin de communiquer en messages UDP.

J'ai donc créé ce script afin de tester les envoie de messages UDP:

```
#include <WiFi.h>
#include <WiFiUdp.h>

// Configuration du réseau WiFi
const char* ssid = "AP-ESTOM";          // Nom du réseau WiFi
const char* password = "Projet-ESTOM_2025"; // Mot de passe du réseau

// Configuration UDP
const char* udpAddress = "192.168.0.10"; // Adresse IP de destination (Raspberry Pi)
const int udpPort = 5005;                // Port cible
WiFiUDP udp;                             // Objet UDP

void setup() {
  // Initialisation de la communication série
  Serial.begin(115200);
  delay(500);

  // Connexion au réseau WiFi
  Serial.print("Connexion au réseau WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnecté au WiFi");

  // Affichage des informations réseau
  Serial.print("Adresse IP de l'ESP32 : ");
  Serial.println(WiFi.localIP());
  Serial.print("Adresse MAC de l'ESP32 : ");
  Serial.println(WiFi.macAddress());


  // Préparation et envoi du message UDP
  sendMACAddress();
}

void loop() {
  // Boucle vide, car tout est effectué dans le setup.
}


// Fonction pour récupérer et envoyer l'adresse MAC via UDP
void sendMACAddress() {
  // Récupération de l'adresse MAC
  String macAddress = WiFi.macAddress();

  // Création du message à envoyer
  String message = "MAC: " + macAddress;

  // Envoi du message via UDP
  udp.beginPacket(udpAddress, udpPort);
  udp.write((uint8_t*)message.c_str(), message.length());  // Conversion en tableau d'octets
  udp.endPacket();

  // Affichage dans le moniteur série pour confirmation
  Serial.println("Message UDP envoyé : " + message);
}
```

Ce code me permet d'envoyer un message UDP depuis ma ESP32 jusqu'à la Raspberry SuperViseur via le point Wifi de cette dernière avec la fonction "**sendMACAddress()**".

Inclusion des bibliothèques et configuration WiFi

Le programme commence par inclure les bibliothèques WiFi.h et WiFiUdp.h, qui permettent respectivement de gérer la connexion au réseau WiFi et d'envoyer des paquets via le protocole UDP. Ensuite, il définit le SSID et le mot de passe du réseau auquel l'ESP32 doit se connecter. Cela lui permet de rejoindre le réseau local afin de communiquer avec la Raspberry.

Configuration de la communication UDP

L'ESP32 doit envoyer son adresse MAC à une machine distante. Pour cela, le programme définit l'adresse IP de destination (192.168.0.10), qui correspond au Raspberry Pi, ainsi que le port UDP 5005 sur lequel les données seront envoyées. L'objet WiFiUDP udp est ensuite créé pour gérer l'envoi des paquets UDP.

Connexion au WiFi

Dans la fonction setup(), l'ESP32 tente de se connecter au réseau WiFi en utilisant WiFi.begin(ssid, password). Tant que la connexion n'est pas établie, le programme affiche un "." toutes les 500 millisecondes. Une fois connecté, l'adresse IP et l'adresse MAC de l'ESP32 sont affichées dans le moniteur série.

Envoi de l'adresse MAC via UDP

Après la connexion, l'ESP32 appelle la fonction sendMACAddress(), qui récupère l'adresse MAC de la carte via WiFi.macAddress(), puis construit un message au format "MAC: XX:XX:XX:XX:XX:XX". Ce message est ensuite envoyé à l'adresse IP du Raspberry Pi sur le port 5005 à l'aide des fonctions udp.beginPacket(), udp.write(), et udp.endPacket(). Une confirmation de l'envoi est également affichée dans le moniteur série.

Mais même si ce script se lance et m'affiche confirmation de l'envoie du message UDP, je dois toujours pouvoir m'assurer si la Raspberry reçoit bien efficacement les messages UDP qui lui sont envoyés.

Pour cela je me suis connecté sur la Raspberry SuperViseur de l'IR 3 avec le logiciel Real VNC Viewer qui permet un connexion à distance d'un poste via le réseau locale, l'IP du poste auquel se connecter, l'identifiant de l'utilisateur de la Raspberry (root dans notre cas) et le mot de passe de l'utilisateur (ESTOM).

Ensuite avec la commande "Nano", j'ai pû créer un script en langage C sur le système Linux de la Raspberry qui "écouterais" et affichera tous les messages UDP qu'elle reçoit sur le port 5005.

Voici le programme :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>  // Pour les structures et fonctions liées aux sockets réseau

#define PORT 5005          // Définition du port d'écoute UDP
#define BUFFER_SIZE 1024   // Taille du buffer pour stocker les messages reçus

int main() {
    int sockfd;
    char buffer[BUFFER_SIZE];
    struct sockaddr_in servaddr, cliaddr;
    socklen_t len = sizeof(cliaddr);

    // Création du socket UDP
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0) {
        perror("Erreur lors de la création du socket"); // Affiche l'erreur en cas d'échec
        exit(EXIT_FAILURE); // Quitte le programme avec un code d'erreur
    }

    // Initialisation de la structure du serveur
    memset(&servaddr, 0, sizeof(servaddr));
    servaddr.sin_family = AF_INET;  // Utilisation d'IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    // Liaison du socket à l'adresse et au port spécifiés
    if (bind(sockfd, (const struct sockaddr *)&servaddr, sizeof(servaddr)) < 0) {
        perror("Erreur lors de la liaison"); // Affiche l'erreur en cas d'échec
        close(sockfd); // Ferme le socket
        exit(EXIT_FAILURE); // Quitte le programme avec un code d'erreur
    }

    while (1) {
        // Réception d'un message UDP
        int n = recvfrom(sockfd, buffer, BUFFER_SIZE, 0, (struct sockaddr *)&cliaddr, &len);
        if (n > 0) {
            buffer[n] = '\0'; // Ajout du terminateur de fin de chaîne pour éviter la corruption de
données
            printf("%s\n", buffer); // Affichage du message reçu
        }
    }

    close(sockfd);
    return 0;
}
```

1. Inclusion des bibliothèques

Le script commence par inclure plusieurs bibliothèques nécessaires à son fonctionnement. La bibliothèque <stdio.h> est utilisée pour la gestion des entrées et sorties, notamment printf. <stdlib.h> permet d'utiliser des fonctions standard comme exit. <string.h> est utile pour manipuler des chaînes de caractères avec des fonctions comme memset. La bibliothèque <unistd.h> contient des fonctions système UNIX telles que close, et <arpa/inet.h> est indispensable pour la gestion des sockets réseau.

2. Définition des constantes

Deux constantes sont définies :

- PORT est fixé à 5005, ce qui signifie que le script écoutera ce port pour recevoir les messages de l'ESP32.
- BUFFER_SIZE est défini à 1024, ce qui correspond à la taille maximale d'un message pouvant être stocké dans le buffer avant d'être affiché.

3. Déclaration des variables

Le programme déclare plusieurs variables essentielles. sockfd est l'identifiant du socket UDP. buffer est un tableau de caractères qui stockera temporairement les messages reçus. servaddr et cliaddr sont des structures sockaddr_in qui contiennent respectivement les informations du serveur et du client (adresse IP et port). La variable len stocke la taille de cliaddr et sera utilisée lors de la réception des messages.

4. Création du socket UDP

Le socket UDP est créé à l'aide de la fonction socket(AF_INET, SOCK_DGRAM, 0). Si la création échoue (sockfd < 0), le programme affiche un message d'erreur et quitte immédiatement avec exit(EXIT_FAILURE). Un socket est un point de communication permettant d'envoyer et de recevoir des données via le réseau.

5. Configuration de l'adresse du serveur

Avant de pouvoir utiliser le socket, il faut configurer l'adresse du serveur. Cela est fait en initialisant servaddr à zéro avec memset, puis en définissant :

- servaddr.sin_family = AF_INET pour indiquer que le socket utilise IPv4.
- servaddr.sin_addr.s_addr = INADDR_ANY afin d'écouter sur toutes les interfaces réseau disponibles.
- servaddr.sin_port = htons(PORT) pour définir le port d'écoute, en prenant en compte l'ordre des octets utilisé par le réseau (big-endian).

6. Liaison du socket au port

La fonction bind() permet d'associer le socket à l'adresse et au port définis. Si cette liaison échoue, le programme affiche une erreur, ferme le socket avec close(sockfd), et quitte immédiatement. Cette étape est cruciale, car sans elle, le socket ne pourra pas recevoir de messages.

7. Boucle de réception des messages UDP

Une boucle infinie while (1) est ensuite lancée pour écouter et traiter en continu les messages UDP reçus.
La fonction recvfrom() attend qu'un message soit reçu et le stocke dans buffer. Si un message est effectivement reçu (n > 0), un \0 (caractère de fin de chaîne) est ajouté pour éviter des erreurs d'affichage, et le message est imprimé avec printf.

8. Fermeture du socket

Le socket est techniquement fermé avec close(sockfd), mais cette ligne ne sera jamais atteinte car la boucle est infinie. Cependant, si une condition d'arrêt était ajoutée, la fermeture du socket permettrait de libérer les ressources système.

Donc voici comment fonctionnent les communications entre l'ESP32 des tours et la Raspberry.

IV.3 - Réception et interprétation des “ordres” du SuperViseur

Ensuite, une fois que les communication entre les deux appareillages étaient établies, j’ai commencé à développer le script actuelle de l’ESP32 afin qu’il puissent :

- **Stocker** les informations locales de la TOM dans le script (Couleur TOM, État de la LED, État du stockage de la TOM).

- **Envoyer** ces informations sous format JSON pour la Raspberry SuperViseur via la Wifi (il faut maintenant envoyer les informations en JSON car l’IR 3 a récupéré le script “d’écoute” que j’avais créé sur la Raspberry pour l’utiliser dans son logiciel qui va gérer le site Web du Projet mais aussi la base de données du site, et donc il a besoin que les messages soient sous la forme JSON).

- **Écouter** pour des “ordres” provenant du logiciel de la Raspberry et les interpréter. Ceci permettront à la TOM d’exécuter des méthodes selon l’avancement du jeu (Allumer la LED au début du jeu et quand la TOM est pleine, la faire clignoter quand elle se fait vider par une BOM, et éteindre la LED quand elle est vide).

Pour réaliser cela, il faut réaliser plusieurs modifications, rendant le script bien plus conséquent.

Pour éviter qu’il devient trop rapidement brouillons, il faut l’organiser dans différentes méthodes qui seront appelées quand cela sera nécessaire.

(Voir “Annexe N°1” pour script complet)

Bibliothèques et Configuration

Le script commence par inclure les bibliothèques nécessaires : WiFi.h, WiFiUdp.h, et ArduinoJson.h. Ces bibliothèques permettent respectivement de gérer la connexion WiFi, d'établir une communication UDP, et de manipuler des objets JSON. Les paramètres de connexion au réseau WiFi, comme le SSID et le mot de passe, ainsi que les configurations pour la communication UDP, y compris l'adresse IP et les ports, sont définis au début du script.

```
#include <WiFi.h>
#include <WiFiUdp.h>
#include <ArduinoJson.h>

// Configuration du réseau WiFi
const char* ssid = "AP-ESTOM";
const char* password = "Projet-ESTOM_2025";

// Configuration UDP
const char* udpAddress = "192.168.0.10";  // Adresse IP du PC/Serveur
const int udpPort = 5005;
const int localUdpPort = 6000;
WiFiUDP udp;
```

Variables et Timers

Deux timers sont déclarés : previousMillis1s et previousMillis1min, qui sont utilisés pour exécuter des actions répétées toutes les secondes et toutes les minutes. De plus, le code définit plusieurs variables pour stocker l'état de la tour, telles que le stockage, l'index de couleur et l'état de la LED. Elles seront ensuite utilisées dans le script pour être utilisées et puis envoyées au SuperViseur.

```
//on prepare un timer
unsigned long previousMillis1s = 0;
unsigned long previousMillis1min = 0;

const unsigned long interval1s = 1000;
const unsigned long interval1min = 60000;

// Configuration des facteurs de la tour
const char* stockageTour = "1";    // Stockage de la Tour
const char* IndexCouleur = "1";    // Index utilisé pour identifier la couleur de la tour  1=vert,
2=jaune, 3=noir, 4=bleu
const char* EtatLED = "1";    // Etat d'allumage de la LED  0=éteint, 1=allumée, 2=clignotte
```

Fonctions Principales

Le code contient plusieurs fonctions clés. La fonction connectWiFi() gère la connexion au réseau WiFi, effectuant des tentatives de reconnexion en cas d'échec.

```
void connectWiFi() {
  Serial.print("Connexion au réseau WiFi");

  WiFi.begin(ssid, password);
  unsigned long startAttemptTime = millis();

  // Tentative de connexion pendant 5 secondes
  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 5000) {
    delay(500);
    Serial.print(".");
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnecté au WiFi");
    Serial.print("Adresse IP ESP32 : ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nÉchec de connexion WiFi, nouvelle tentative dans 5 secondes...");
  }
}
```

La fonction listenOrder() est responsable de l'écoute des ordres envoyés via UDP. Elle attend jusqu'à 5 secondes pour recevoir un paquet, puis essaie de le parser en tant qu'objet JSON. Selon l'ordre reçu, le code peut effectuer différentes actions, comme demander des informations, initialiser la TOM, vider la TOM, éteindre la LED, ou mettre en pause et reprendre une partie.

Boucle Principale

La boucle principale, définie dans la fonction loop(), vérifie si une seconde ou une minute s'est écoulée pour exécuter les fonctions appropriées.

```
void loop() {
  unsigned long currentMillis = millis();


  if (currentMillis - previousMillis1s >= interval1s) {
    previousMillis1s = currentMillis;
    functionEvery1Seconds();
  }


  if (currentMillis - previousMillis1min >= interval1min) {
    previousMillis1min = currentMillis;
    functionEveryMinute();
  }
}
```

La fonction functionEvery1Seconds() appelle listenOrder() pour traiter les ordres entrants, tandis que functionEveryMinute() envoie un message JSON contenant des informations sur l'état de la tour. Le fait qu'elle écoute les ordres toute les secondes assure que l'ESP32 reste réactif et communicatif, et l'enovie des informations toutes les minutes permet au SuperViseur de mettre à jour sa base de données si jamais une TOM se déconnecte.

```
void functionEvery1Seconds() {
  listenOrder();
}


void functionEveryMinute() {
  Serial.println("Execution toutes les 1minute !");
  sendUDPMessage();
}
```

Actions Définies

Le code définit plusieurs “ordres” spécifiques. La fonction remiseAZero() réinitialise les paramètres de la tour et allume la LED, tandis que clignotementLED() change l'état de la LED pour qu'elle clignote.

```
void remiseAZero(){
  stockageTour = "1";
  EtatLED = "1";
  Serial.println("La LED est allumé");
  sendUDPMessage();
}


void clignotementLED(){
  EtatLED = "2";
  Serial.println("La LED clignotte");
}
```

La fonction eteintLED() éteint la LED, et les fonctions pausePartie() et reprise() indiquent respectivement si la partie est mise en pause ou reprise. Ces actions permettent un contrôle flexible de l'appareil.

```
void eteintLED(){
  EtatLED = "0";
  Serial.println("La LED est éteinte");
}


void pausePartie(){
  Serial.println("Partie en pause");
}


void reprise(){
  Serial.println("Reprise de la partie");
}
```

Exécution

Pour résumer, lors de l'allumage, le code essaie d'abord de se connecter au WiFi.

Une fois la connexion établie, il démarre le socket UDP et envoie un message initial. Ensuite, l'ESP32 entre dans sa boucle principale, écoutant les ordres UDP tout en exécutant les actions programmées toutes les secondes et toutes les minutes. Ce fonctionnement permet à l'ESP32 de recevoir des commandes à distance, d'exécuter des actions en fonction de ses ordres, et de communiquer des informations sur son état à la Raspberry du SuperViseur.

IV.4 - Nouvelle carte et intégration du protocole du TCP

L'équipe ER a décidé de changer la carte ESP32 en une carte Z4T-XIAOESP32S3.

D'un point de vue informatique, cela ne change rien pour la conception du script. Juste il faut aller dans le Gestionnaire de carte dans le logicielle Arduino et sélectionner le modèle XIAO_ESP32S3 afin que le transfert du script marche.

Après plusieurs délibérations au sein du groupe, nous avions décidé de ne plus utiliser les **protocoles UDP** dans les communications entre pareils , et de le remplacer par l'utilisation du **protocole TCP**.

La raison de ce changement est dû au fait que le protocole UDP offre de **meilleurs fiabilité** et dispose de plusieurs **qualités**, tel que :

- **Fiabilité de transmission :**
  TCP garantit que les données arrivent correctement et dans le bon ordre, grâce à :
  - L'accusé de réception (ACK)
  - La retransmission automatique des paquets perdus

- **Connexion orientée :**
  TCP établit une connexion stable avant d'envoyer des données, assurant un lien cohérent entre les deux appareils.

- **Moins de perte de données :**
  Grâce à ses mécanismes de contrôle et de correction d'erreurs, TCP minimise fortement les pertes de données.

- **Simplicité côté script:**
  Avec TCP, l'application n'a pas besoin de gérer la perte, la réémission ou l'ordre des paquets — tout cela est géré par la couche TCP, ce qui simplifie énormément le script du côté SuperViseur.

- **Meilleure sécurité de transmission :**
  TCP étant plus structuré, il est plus adapté à l'intégration avec des couches de sécurité comme TLS (HTTPS, par exemple).

Pour réaliser ces modifications, j'ai dû toucher plusieurs parties du script et les modifier afin d'implémenter de façon fluide le protocole TCP.

Voici les parties modifiées du code :

Fonction **establishTCPConnection()** :

Maintenant, l'ESP32 doit jouer le rôle d'un "**client**" au serveur TCP de la Raspberry.

Cela signifie qu'après avoir réussi sa connexion au Wifi du SuperViseur, il va tenter de se connecter au serveur TCP de la Raspberry en boucle jusqu'à ce qu'il établisse une connexion.

Il utilise la méthode **clientRecpet.connect()** pour se connecter au serveur TCP à l'adresse **tcpAddress** par le port **tcpPort**.

```
void establishTCPConnection() {
  Serial.println("Connexion au serveur TCP...");
  if (clientRecept.connect(tcpAddress, tcpPort)) {
    Serial.println("Connexion TCP réussie.");
    sendTCPMessageToServer();  // Envoi initial
  } else {
    Serial.println("Échec de connexion TCP.");
  }
}
```

Fonction handleIncomingOrder() :

Ici, **listenOrder()** deviens **handleIncomingOrder()**. Le changement ici est l'utilisation de la commande **clientRecpt.readStringUntil('\n')** qui va lire toute nouvelle réception de communications provenant du serveur TCP et va les lire jusqu'à un retour à la ligne **\n**.

Il va ensuite poursuivre son fonctionnement comme précédemment en transformant les **données JSON** qu'il reçoit en **Ordre** qu'il pourra interpréter.

```
void handleIncomingOrder() {
  String data = clientRecept.readStringUntil('\n');
  if (data.length() == 0) return;

  StaticJsonDocument<512> doc;
  DeserializationError error = deserializeJson(doc, data);
  if (error) return;

  int ordre = doc["Ordre"];
  switch (ordre) {
    case 1: sendTCPMessageToServer(); break;
    case 2: remiseAZero(); sendTCPMessageToServer(); break;
    ...
  }
}
```

Fonction sendTCPMessageToServer() :

La fonction sendTCPMessageToServer() a pour rôle d'envoyer un message JSON formaté au serveur du SuperViseur via la connexion TCP, afin de lui transmettre l'état actuel de la TOM.Elle commence par créer un document JSON contenant des données clés comme l'adresse IP locale, l'adresse MAC, le niveau du signal WiFi (RSSI), et l'état d'une LED.

Ce document est ensuite sérialisé dans une chaîne de caractères, qui est envoyée au serveur via clientRecept.print(), suivie d'un saut de ligne pour marquer la fin de la trame. Un message est également affiché sur le port série pour confirmer l'envoi.

Cette fonction est cruciale car cela sera la fonction qui sera appelée par l'Ordre 1, qui permettra au SuperViseur de demander sur commande les différentes données de toute les TOM pour qu'il puisse mettre à jour sa base de données.

Cette fonction est appelée au Setup(), toutes les minutes, et à l'exécution de chaque Ordre quasiment.

```
void sendTCPMessageToServer() {
  if (!clientRecept.connected()) {
    if (!clientRecept.connect(tcpAddress, tcpPort)) {
      Serial.println("Échec de reconnexion");
      return;
    }
  }

  DynamicJsonDocument doc(256);
  doc["Type"] = "TOM";
  doc["Couleur"] = IndexCouleur;
  doc["Stockage"] = stockageTour;

  serializeJson(doc, clientRecept);
  clientRecept.println();  // Important : fin du message

  Serial.print("Infos envoyées : ");
  serializeJson(doc, Serial);
  Serial.println();
}
```

Fonction loop() :

La fonction **loop()** a été remanié afin qu'il n'appelle pas à chaque seconde la fonction **listenOrder()** (car cela n'est pas nécessaire avec la fonction **clientRecept.avaible()**.

La fonction **loop()** assure en continu la gestion de la communication avec le serveur. Elle commence par vérifier si la connexion TCP avec le SuperViseur est toujours active ; si ce n'est pas le cas, elle tente de la rétablir immédiatement et, en cas de succès, elle renvoie les informations actuelles au serveur.

Ensuite, elle surveille la présence de données entrantes sur la connexion TCP. Si des données sont reçues, la fonction **handleIncomingOrder()** est lancée.

Enfin, la fonction vérifie si une minute s'est écoulée depuis le dernier envoi par le biais d'un timer. Si c'est le cas, elle envoie à nouveau les informations d'état de la TOM au serveur pour maintenir une synchronisation de la base de données de la Raspberry régulière. Cette boucle permet donc de garder la connexion active, de traiter les ordres reçus et de transmettre l'état des TOM à intervalles réguliers.

Cette fonction est appelée après le Setup(), et sert au fonctionnement général du script.

```
void loop() {
  unsigned long currentMillis = millis();

  if (!clientRecept.connected()) {
    if (clientRecept.connect(tcpAddress, tcpPort)) {
      sendTCPMessageToServer();
    } else {
      delay(1000);
      return;
    }
  }

  if (clientRecept.available()) {
    handleIncomingOrder();
  }

  if (currentMillis - previousMillis1min >= interval1min) {
    previousMillis1min = currentMillis;
    sendTCPMessageToServer();
  }
}
```

Récapitulative

En conclusion, ce script est une nette amélioration comparé au script précédents.

Il permet de stocker toutes les fonctionnalités nécessaires au bon fonctionnement de la TOM durant la partie de jeu, permettant un contrôle à distance par le SuperViseur via des ordres et un retour utilisateur via le contrôle de sa LED.

Cependant, il manque encore une implémentation “physique” du contrôle de LED, qui sera en collaboration avec les étudiants d’ER afin de savoir comment utiliser ces LED à travers le script des TOM (quelle librairie utiliser, à quel GPIO accéder, etc…), ou bien l’utilisation d’un switch qui sera utilisé pour choisir physiquement la couleur de la TOM durant la phase de préparation.

IV.5 - Fonctionnement de l'antenne de l'ESP32

1. Rôle de l'antenne dans une communication Wi-Fi

L'antenne est un composant essentiel qui permet à l'ESP32 de **convertir des signaux électriques en ondes électromagnétiques** (en émission), et inversement (en réception).
Dans notre projet, chaque module ESP32 est équipé d'une **antenne externe**, connectée par un port u.FL. Cela permet une meilleure portée et une meilleure réception qu'une antenne intégrée.

Lorsqu'un signal électrique alternatif traverse l'antenne, il crée un **champ électrique et magnétique oscillant**, ce qui génère une onde électromagnétique rayonnante. Cette onde se propage dans l'air et transporte les informations numériques (données du protocole TCP/IP).

2. Fréquence, longueur d'onde et adaptation

L'antenne utilisée dans le projet fonctionne à la **fréquence de 2,4 GHz**, ce qui est standard pour les communications Wi-Fi (802.11 b/g/n).
Cette fréquence correspond à une longueur d'onde d'environ :

$\lambda = c / f = (3{,}0 \times 10^8 \text{ m/s}) / (2{,}4 \times 10^9 \text{ Hz}) \approx$ **12,5 cm**

Les antennes sont souvent dimensionnées en fonction de cette longueur d'onde.
Par exemple, une **antenne quart d'onde** à 2,4 GHz mesure environ **3,1 cm**, ce qui correspond à $\lambda/4$. Cette dimension maximise l'efficacité du rayonnement.

3. Adaptation d'impédance et efficacité

Pour qu'un maximum d'énergie soit transféré entre le circuit de l'ESP32 et l'antenne, il faut que leurs **impédances soient adaptées** (typiquement 50 ohms).
Une mauvaise adaptation entraîne des **réflexions d'onde** dans le câble (phénomène d'onde stationnaire), ce qui réduit considérablement la puissance rayonnée.

La qualité de cette adaptation est mesurée par le **taux d'onde stationnaire** (ROS ou VSWR – Voltage Standing Wave Ratio). Idéalement, ce rapport est proche de 1:1.

4. Polarisation et directivité

L'onde électromagnétique rayonnée par une antenne est **polarisée**, généralement de façon linéaire.
Pour une transmission optimale, il faut que la polarisation de l'antenne de l'émetteur corresponde à celle du récepteur.

La **directivité** de l'antenne externe influence également la portée :

- Une antenne **omnidirectionnelle** rayonne dans toutes les directions à l'horizontale (comme une tige verticale).
- Une antenne **directionnelle** concentre l'énergie dans une direction précise, augmentant la portée dans cet axe.

Dans le projet, l'antenne utilisée est **omnidirectionnelle**, ce qui permet une couverture uniforme autour du robot.

5. Atténuation et environnement

L'efficacité de l'antenne peut être affectée par l'environnement :

- **Réflexion** sur les surfaces métalliques,
- **Absorption** par l'eau ou le corps humain,
- **Interférences** avec d'autres sources électromagnétiques.

Cependant, dans notre cas, les distances sont faibles (quelques mètres) et les obstacles rares.
La **perte de puissance** liée à la distance peut être modélisée avec la **formule de Friis** :

$P_r = (Pe \times G_e \times G_r \times \lambda^2) / ((4\pi \times R)^2)$

où :

- P_r = puissance reçue,
- Pe = puissance émise,
- G_e et G_r = gains des antennes,
- λ = longueur d'onde,
- R = distance.

Conclusion

L'utilisation d'une antenne externe améliore fortement la qualité de la connexion Wi-Fi, en garantissant une meilleure sensibilité en réception et un rayonnement plus efficace.
Ce choix est particulièrement pertinent dans un projet embarqué comme le nôtre, où la fiabilité de la communication sans fil est essentielle pour recevoir des instructions et transmettre des données en temps réel.

V - Compétences mobilisées

➢ Compétences Techniques :

- **Environnement Arduino :** Configuration et utilisation des microcontrôleurs Arduino ESP32-S3.
- **Conception de script en C :** Création de scripts en langage C pour le fonctionnement de la carte Arduino et l'écoute des messages UDP sur la Raspberry.
- **Développement Web :** Réarrangement de différents éléments sur le site web du projet via des "Grid" en PHP/CSS.

➢ Compétences Méthodologique :

- **Organisation d'un projet :** Utilisation de Microsoft Project et prévisions des tâches à venir.
- **Répartition du travail :** Se répartir les tâches entre coéquipiers en différentes sections pour travailler plus efficacement.

➢ Compétences Transversales :

- **Analyse du code et erreurs :** Scruter pour des erreurs au sein du code et mise en place de correctifs simples et efficaces.
- **Création de ressources graphiques :** Créer différentes ressources 2D pour le site internet du Projet.
- **Communication professionnelle :** Rédaction de fiches de recettes, documentation utilisateur, compte rendu de réunion, communication avec l'ensemble de l'équipe.
- **Travail en autonomie et avec rigueur :** Travail bien effectué et respect des délais.

VI - Difficultés rencontrées et surmontées

Problème N°1: Le premier problème rencontré sur ma partie fût le fait de connecté la ESP32 à mon ordinateur, je ne savais pas quel logiciel utiliser et puis après l'installation j'ai eu du mal à faire fonctionner l'environnement de développement Arduino pour le rendre compatible avec mon microcontrôleur.

Solution N°1: Suite à des recherches en ligne et la lecture des documentations officielles de la carte Arduino ESP32-S3, j'ai su qu'il fallait utiliser le logiciel d'environnement de développement Arduino IDE sur Windows 10. Après avoir installé et lancé le logiciel, j'ai branché le microcontrôleur en filaire avec un câble USB-C et j'ai dû installer différents plugins dans le logiciel afin que la carte et l'IDE puissent communiquer ensemble. Dans un premier temps, j'ai installé la bibliothèque "WiFiEsp" afin que le script puisse utiliser des commandes pour se connecter au Wifi du SuperViseur. Ensuite j'ai installé dans le "Gestionnaire de carte" le paquet "Ardino ESP2 Boards" afin de pouvoir ensuite sélectionner dans "Outils>Cartes>esp32" le module de développement "ESP32S3 Dev Module" pour enfin pourvoir comment le développement de mon script.

Problème N°2: Le second problème fût comment vérifier que les messages envoyés à la Raspberry du SuperViseur soit bien exacte ?

Solution N°2: Pour cela j'ai utiliser RealVNC Viewer pour me connecter à la Raspberry de l'IR 3 qui sert de SuperViseur et j'ai ensuite créé un script d'écoute de messages UDP en langage C afin de vérifier que les messages envoyés depuis la ESP32 soient bien reçus et interprétés par la Raspberry.

Problème N°3: Suite à un jour de maladie m'ayant empêché de poursuivre l'avancée de l'écriture du script, à mon retour mon script ne recevait plus les ordres émis par le SuperViseur.

Solution N°3: Après plusieurs vérifications avec l'IR 3, nous avons remarqué que le SuperViseur envoyait bien les ordres en UDP à la bonne adresse IP, mais la TOM ne semblait pas récupérer et afficher les messages qu'elle recevait. Après plusieurs modifications de mon script afin de régler le problème, rien ne change et j'ai donc décidé de me baser sur le script de la BOM, afin de réécrire un script pour la TOM à neuf afin de retrouver l'état originel de fonctionnement. J'ai donc copié le script de la BOM que j'ai ensuite modifié avec des morceaux de mon script afin que cela marche à nouveau. Après avoir téléversé mon nouveau script et quelques tests de communications d'ordres, la TOM remarché enfin et capté bien les ordres !

Problème N°4: Après être passé sur un nouveau modèle de carte Arduino (la XIAO-ESP32S3), la fonction qui gère la connexion Wifi au SuperViseur ne fonctionnait plus et affichait le message d'erreur "sta is connecting, return error".

Solution N°4: Après plusieurs vérifications, il s'avérait que le nouveau micro-contrôleur ne disposait pas d'antenne intégrée. Cela expliquait le message d'erreur qui s'affichait car il ne pouvait pas accéder à une antenne pour capter un signal. Donc le problème fût réglé dès que l'on brancha une antenne sur le micro-contrôleur. Il a pû capter le signal Wifi.

VII - Tâches externes à ma partie

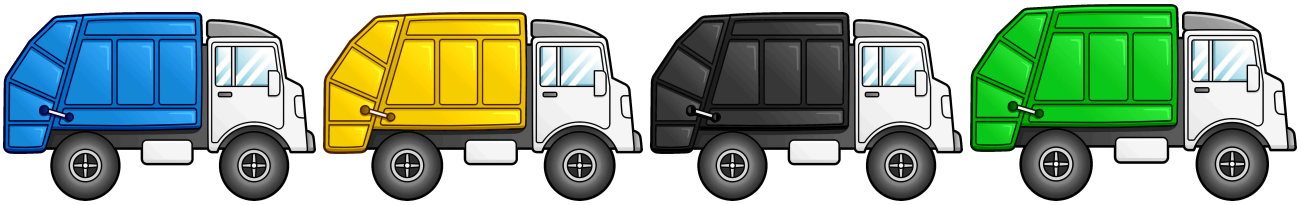
Durant mes activités sur le projet, j'ai eu quelques temps morts durant lesquels j'ai aidé mes coéquipiers à créer et éditer différentes images pour l'aspect esthétique du projet (la plupart créés sur Paint.Net) .

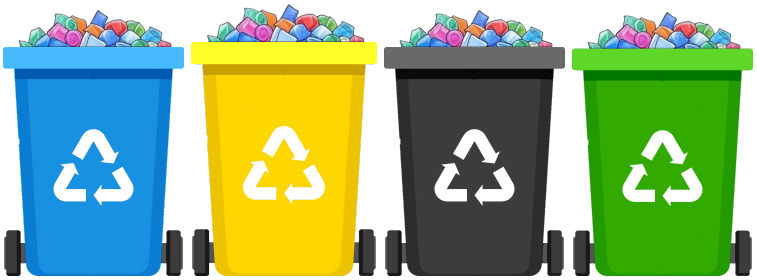
Parmis mes différentes contributions, voici quelque unes d'entre elles:

- Le logo du projet:

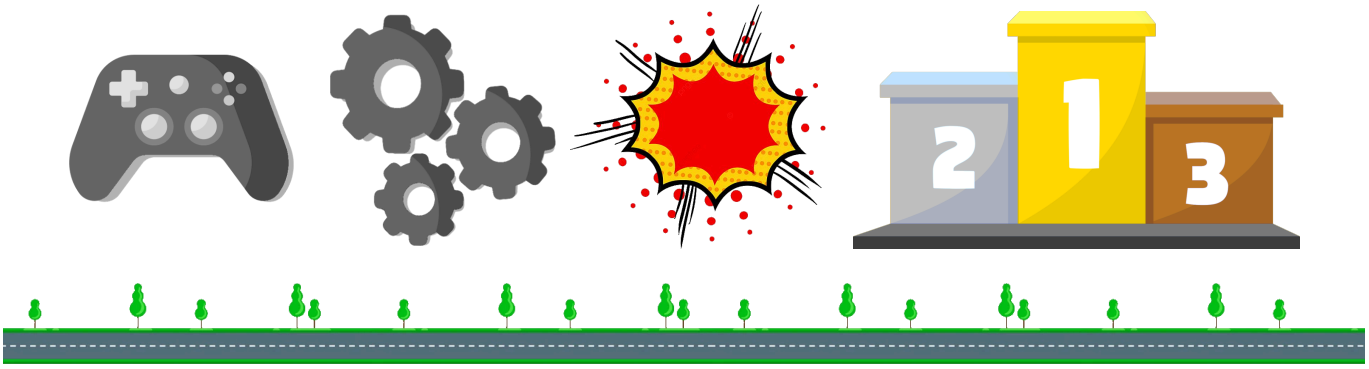


- Icônes utilisés sur le site Web :









- Images de fonds pour le site internet (généré en partie par IA):





IV - Bilan personnel

Suite au développement des TOM dans le cadre du projet ESTOM, je suis confiant sur le fait de dire que le script pour les tours TOM est quasiment terminé, sur le plan informatique.

Il manquera encore l'implémentation physique du contrôle de la LED, pour avoir un retour utilisateur pour le joueur, et aussi l'implémentation d'un switch qui sera utilisé pour choisir physiquement la couleur de la TOM durant la phase de préparation.

Mais à part cela, tout marche comme il le faut. Je remercie nos enseignants qui nous ont guidés par des conseils judicieux et une ligne directrice à suivre.

Et je remercie mon équipe, avec qui j'ai pû travailler dans un environnement propice au travail et à la collaboration.

- Guilhem Bussi (IR 2)

Partie Personnelle (Clément VOLLE - IR 3) :

SuperViseur

I - Introduction personnelle

Dans le cadre de ma deuxième année de BTS CIEL, j'ai intégré l'équipe du projet ESTOM (Espace de Sensibilisation au Tri des Ordures Ménagères) dont l'objectif est de concevoir un système ludique de tri des déchets basé sur des véhicules radio télécommandés (BOM) et des tours à ordures Ménagères (TOM).

Ma mission, relevant de la sous-épreuve IR-3, consiste à développer l'interface applicative et Web de supervision du jeu. Cette interface doit permettre à la fois de configurer les parties (nombre de TOM, durée, pénalités…), de suivre en temps réel l'état des BOM & TOM via une base de données centralisée, et d'assurer la communication sécurisée entre le serveur Raspberry Pi et les objets connectés du système (micro-contrôleur BOM & TOM).

II - Objectifs et Exigences

II.1 - Objectifs fonctionnels

➢ Interface de configuration :

L'interface de configuration doit offrir au superviseur les possibilités suivantes :

- **Définir le nombre de TOM par couleur**, et, en cas de défaillance d'une TOM, ajuster automatiquement le nombre total de TOM pour garantir l'équité entre les joueurs.
- **Fixer le nombre de BOM** selon le même principe, afin de compenser toute panne technique.
- **Paramétrer la durée de la partie** sur une plage comprise entre 30 secondes et 600 secondes (10 minutes).
- **Configurer le malus de collision**, ajustable de 0,2 à 5 points selon le niveau de pénalité souhaitée.

Pendant le développement, j'ai également ajouté deux règles optionnelles pour enrichir le gameplay :

- **Gain de points fixe** : attribution d'un nombre de points constant à chaque récolte de TOM.
- **Gain de points progressif** : attribution de points croissants à mesure que le joueur collecte plusieurs TOM consécutives.

➢ Contrôle du jeu :

Le contrôle du jeu doit se faire sur la page principale du site Web, j'ai donc implémenté 4 options possibles pour le contrôle du jeu.

- **Lancer la partie** permet de définir le statut du jeu à 1 (partie en cours)
- **Mettre en pause** permet de définir le statut du jeu à 2
- **A l'arrêt** (stop), une fois le Timer terminé, le statut est défini à 0
- **Lors du compte à rebours**, qui dure 5 secondes (pour avertir que la partie va bientôt commencer) le statut du jeu est défini à 3

➢ Visualisation des résultats en temps réel :

Besoin essentiel d'un affichage simple et clair des résultats de la partie en Temps réel, mise à jour de l'affichage toutes les secondes.

II.2 - Exigences techniques

➢ **Sécurité réseau :**
Accès Wi-Fi protégé par la solution WPA2-PSK dans notre cas, pare-feu configuré pour interdire les ports inutiles.

➢ **Performance :**
Temps de réponse et communication des informations suffisamment rapide.

➢ **Fiabilité :**
Affichage continu durant tout le temps d'une partie, pas de bugs visuels. Aucun problème visible, réception certaine des données de BOM & TOM.

➢ **Compatibilité :**
Doit pouvoir fonctionner sur Chrome, Firefox et testé sur Tablette.

➢ **Documentation :**
Manuel utilisateur et fiches recettes complètes.

III - Planification

III.1 - Outils et méthode

Pour mener à bien ma mission, j'ai fait appel à un ensemble **d'outils techniques** et méthodologiques afin de structurer chaque étape de mon travail.

- Pour le développement, j'ai pris l'initiative de coder sur Qt-Creator pour créer mon application en C++, cet outil de développement est intéressant pour son système de debug, elle est également compatible avec l'os de la Raspberry Pi.
- Pour la partie Web, je vais simplement utiliser l'application VS Code sur Windows, liée en ssh à la raspberry Pi.
- Au niveau de la création de la documentation, je vais utiliser MagicDraw, MySQL WorkBench et Microsoft Project.

N'ayant aucune contrainte imposant une méthode formelle, j'ai adopté une approche **itérative et incrémentale** qui m'a permis de progresser fonctionnalité par fonctionnalité selon le schéma suivant :

- **Réflexion** : définition claire de l'objectif et choix de la solution technique
- **Développement** : implémentation du code correspondant
- **Tests** : validation unitaire puis intégrée avant passage à la fonctionnalité suivante

Cette démarche cyclique assure l'avancement et une correction rapide des anomalies de chaque fonctionnalité. Le souci de cette méthodologie d'après moi est le manque de vision "globale", je suis donc revenu à de nombreuses reprises sur les fonctionnalités déjà présentes pour les modifier et les rendre cohérentes avec l'ensemble de mes fonctionnalités et de mon objectif.

Je vais également ordonner ma partie en 3 itérations, voici le nom des itérations ainsi que leur contenu respectif :

Itération I – Mise en place de l'environnement

- Installation et configuration du serveur Raspberry Pi
- Mise en place du hotspot Wi-Fi sécurisé (hostapd + dnsmasq)

Itération II – Développement de l'interface Web

- Conception du modèle relationnel dans MySQL Workbench
- Création des pages PHP : paramétrage du jeu, gestion du timer, contrôle des commandes
- Implémentation du rafraîchissement dynamique via JavaScript (AJAX)
- Tests unitaires et recette fonctionnelle de l'interface

Itération III – Développement de l'application et interaction avec l'interface Web.

- Développement des classes C++ dans Qt Creator
- Lecture continue du statut du jeu et réception des messages TCP JSON (système d'envoi d'ordre et de réception de statuts)
- Synchronisation bidirectionnelle entre l'application C++ et la base MySQL
- Validation de la communication front-end/back-end et tests d'intégration

III.2 - Tâches planifiées

Pour planifier les tâches, j'ai élaboré une ébauche de la planification de mon projet, que je mets à jour régulièrement avec une version réelle au fur et à mesure de l'avancement.

Planification prévisionnel (08/01/2025):

Nom de la tâche	Durée	Début	Fin	Prédé	Noms ressources	Calendrier des tâches
Projet ESTOM	**188 heures**	**Mar 07/01/25**	**Lun 09/06/25**		ER1;ER2;IR1;IR2;IR3	**Projet E62 Session 2025**
Découverte & analyse du projet	2 heures	Mar 07/01/25	Mar 07/01/25			Aucun
Première réunion d'information	1 heure	Mer 08/01/25	Mer 08/01/25	2	IR1;IR2;IR3	Aucun
Rédaction compte Rendu Réunion	30 minutes	Mer 08/01/25	Mer 08/01/25	3	IR3	Aucun
Réalisation du Project	1 heure	Mer 08/01/25	Mer 08/01/25	4	IR3	Aucun
Réalisation global du site Internet avec BDD sur RPI 3	18 heures	Mer 08/01/25	Mar 21/01/25	5	IR3	Aucun
Revue 1	1 heure	Mar 21/01/25	Mar 21/01/25	6	IR3	Aucun
Configuration Wifi (Mise en place)	2 heures	Mar 21/01/25	Ven 24/01/25	7	IR3	Aucun
Configuration Firewall et sécurisation du site	2 heures	Ven 24/01/25	Lun 27/01/25	8	IR3	Aucun
Logiciel SuperVision	**39 heures**	**Lun 27/01/25**	**Mer 05/03/25**			**Aucun**
Codage du logiciel (général)	20 heures	Lun 27/01/25	Ven 07/02/25	9	IR3	Aucun
Interface de com. Avec BOM	19 heures	Ven 07/02/25	Mer 05/03/25	11	IR3;IR1	Aucun
Revue 2	1 heure	Mer 05/03/25	Mer 05/03/25	12	IR3	Aucun
Finalisation du site WEB, mise au propre	10 heures	Mer 05/03/25	Mer 12/03/25	13		Aucun
Création Fiche de recette	3 heures	Mer 12/03/25	Ven 14/03/25	14		Aucun
Finalisation Application SuperVision	30 heures	Ven 14/03/25	Mer 02/04/25	15		Aucun
Test et synchronisation global et général avec l'équipe	20 heures	Mer 02/04/25	Mer 30/04/25	16		Aucun
Revue 3	1 heure	Mer 30/04/25	Mer 30/04/25	17		Aucun
Revue finale	**1 heure**	**Lun 09/06/25**	**Lun 09/06/25**			**Aucun**

Planification réelle (26/03/2025):

Nom de la tâche	Durée	Début	Fin	Prédé	Noms ressources
Projet ESTOM	**188 heures**	**Mar 07/01/25**	**Lun 09/06/25**		ER1;ER2;IR1;IR2;IR3
Découverte & analyse du projet	2 heures	Mar 07/01/25	Mar 07/01/25		
Première réunion d'information	1 heure	Mer 08/01/25	Mer 08/01/25	2	IR1;IR2;IR3
Rédaction compte Rendu Réunion	30 minutes	Mer 08/01/25	Mer 08/01/25	3	IR3
Réalisation du Project	1 heure	Mer 08/01/25	Mer 08/01/25	4	IR3
Mise en place du point d'accès Wifi sécurisé (WPA2) sur la raspberry	3 heures	Mer 08/01/25	Ven 10/01/25	5	IR3
Réalisation global du site Internet sur RPI 3	**18,67 heures**	**Ven 10/01/25**	**Ven 24/01/25**		**IR3**
Page d'accueil et Header du site (Design)	6 heures	Ven 10/01/25	Mer 15/01/25	6	
Réunion n°2 (avec ER + IR + Prof) Général	1,5 heures	Mar 14/01/25	Mar 14/01/25		ER1;ER2;IR1;IR2;IR3
Réunion n°3 (IR SANS PROF) BDD	30 minutes	Ven 17/01/25	Ven 17/01/25		IR1;IR2;IR3
Mise en place de la base de donnée	3 heures	Mer 15/01/25	Ven 17/01/25	8	
Page "Gestion des parties"	**4 heures**	**Ven 17/01/25**	**Lun 20/01/25**		
Création d'un "Timer" en temps réel	4 heures	Ven 17/01/25	Lun 20/01/25	11	
Page "parametre du jeu"	**4 heures**	**Mar 21/01/25**	**Ven 24/01/25**		
Création d'un formulaire pour les différents paramètre du jeu	4 heures	Mar 21/01/25	Ven 24/01/25	17	
Réunion n°4 - Comunication TOM-BOM	40 minutes	Mar 21/01/25	Mar 21/01/25	13	IR1;IR2;IR3
Rédaction CR Réunion n°4	30 minutes	Mar 21/01/25	Mar 21/01/25	16	
Logiciel SuperVision 1	**15 heures**	**Ven 24/01/25**	**Mar 04/02/25**		
Début du Logiciel	11 heures	Ven 24/01/25	Ven 31/01/25	15	
Model MCD (base de donné)	3 heures	Ven 31/01/25	Mar 04/02/25	19	
Eboche papier d'un diagramme de Classe	1 heure	Mar 04/02/25	Mar 04/02/25	20	
Réunion n°5 - Communication TOM-BOM FullTeam	1,5 heures	Mar 04/02/25	Mar 04/02/25	21	ER1;ER2;IR1;IR2;IR3
Revue 1	1 heure	Mer 05/02/25	Mer 05/02/25	22	IR3
Diagramme de Séquence com BOM & TOM	2 heures	Mer 05/02/25	Mer 05/02/25	23	
Logiciel SuperVision 2	**28 heures**	**Ven 07/02/25**	**Mar 11/03/25**		
Création Classe base de donné	5 heures	Ven 07/02/25	Lun 24/02/25	24	IR3
Création Classe CheckConnect	4 heures	Mar 25/02/25	Mer 26/02/25	26	
Simulation COM entre BOM & TOM avec Team IR	5 heures	Mer 26/02/25	Ven 28/02/25	27	IR1;IR2;IR3
Mise en place du système d'ordre & de status	6 heures	Lun 03/03/25	Mer 05/03/25	28	
Multiple test de communication avec l'API & TOM/BOM	8 heures	Mer 05/03/25	Mar 11/03/25	29	IR1;IR2;IR3
Mise à jour du Site web et finalisation des paramètres	15 heures	Mar 11/03/25	Mer 19/03/25	30	
Grande simulation d'une partie complète (virtuellement) & multiple réglage	6 heures	Mer 19/03/25	Lun 24/03/25	31	IR1;IR2;IR3
Réalisation du Dossier de Projet & préparation Revue	15 heures	Mar 25/03/25	Mer 02/04/25	32	
Revue 2	1 heure	Mer 02/04/25	Mer 02/04/25	33	IR3

Revue 2	1 heure	Mer 02/04/25	Mer 02/04/25	33	IR3
Finalisation du Diagramme de Classe de l'application	3 heures	Ven 04/04/25	Ven 04/04/25	34	
Application des conseils professeur pour la revue de projet suivante	8 heures	Mar 22/04/25	Ven 25/04/25	35	
Adaptation du code UDP en TCP	16 heures	Ven 25/04/25	Mar 06/05/25	36	
Sécurisation du site Web	2 heures	Mar 06/05/25	Mer 07/05/25	37	
Création des fiches recette	5 heures	Mer 07/05/25	Ven 09/05/25	38	
Rédaction du Dossier de Projet Revue 3 (ajout partie Physique)	10 heures	Lun 12/05/25	Ven 16/05/25	39	
Préparation Oral Revue 3	3 heures	Ven 16/05/25	Lun 19/05/25	40	
Revue 3	1 heure	Mar 20/05/25	Mar 20/05/25	41	

IV - Présentation de ma partie Applicative & Web

IV.1 - Généralités

Ma mission au sein du projet ESTOM consiste à concevoir, développer et déployer **l'interface applicative et Web de supervision**, qui assure la configuration du jeu, la collecte des données en temps réel et l'affichage dynamique des résultats pour les utilisateurs (superviseurs). Cette couche logicielle fait le lien entre :

- **Le système embarqué C++** (classes CDBReader et TCPListener) qui récupère en continu le statut des BOM et TOM depuis la base de données et via les messages TCP au format JSON
- **La base de données MySQL "ESTOM"**, pour l'état du jeu (paramètres, scores, statuts, ip, etc…), ainsi que les informations des BOM & TOM.
- **Le site Web PHP/Mysql**, accessible depuis le hotspot Wi-Fi hébergé sur la Raspberry Pi, qui fournit une interface graphique sécurisée pour configurer et piloter la partie

IV.2 - Architecture technique

- **Front-end** : PHP procédural avec MySql, JavaScript (AJAX) pour le rafraîchissement automatique
- **Back-end** : Raspberry Pi (Raspbian + Apache2 + PHP + MySQL), point d'accès WPA2 isolé (hostapd + dnsmasq)
- **Base de données** : schéma relationnel simple (tables Config, BOM, TOM) conçu dans MySQL Workbench
- **Intégration temps réel** : classe CDBReader (Qt Creator) effectuant un polling SQL toutes les secondes pour mettre à jour la variable *status* (Statut de la partie)

IV.3 - Sécurité et fiabilité

- Authentification WPA2-PSK pour l'accès Wi-Fi
- Pare-feu iptables configuré pour n'autoriser que le trafic HTTP(S) et MySQL interne
- Validation côté serveur des requêtes PHP pour éviter les injections SQL

Cette partie applicative & Web constitue le **pivot central du projet ESTOM**, garantissant la cohérence entre les composants matériels (BOM/TOM) et l'expérience utilisateur lors du jeu.

V - Déroulement du travail

V.0 - Le Wi-Fi : Approche Physique des Points d'Accès

1. Introduction

Le Wi-Fi (Wireless Fidelity) est une technologie de communication sans fil utilisée pour connecter des appareils à un réseau local (LAN) via des ondes radio. Dans un réseau Wi-Fi, le **point d'accès (ou access point)** est un élément central qui joue le rôle de relais entre les appareils et le réseau câblé. Au-delà des aspects logiciels et réseaux, le fonctionnement du Wi-Fi repose sur des principes **physiques fondamentaux** liés à la **propagation des ondes électromagnétiques**.

2. Nature Physique du Signal Wi-Fi

Le Wi-Fi utilise des **ondes électromagnétiques**, plus précisément des **ondes radio**. Ces ondes sont caractérisées par leur fréquence, leur longueur d'onde et leur puissance.

Les principales bandes de fréquences utilisées sont :

- **2,4 GHz** (longue portée mais plus de perturbations)
- **5 GHz** (portée plus courte mais moins d'interférences)
- **6 GHz** (Wi-Fi 6E, encore plus rapide mais portée encore plus réduite)

Ces ondes sont émises par l'antenne du point d'accès et se propagent dans l'environnement. Elles peuvent :

- **être absorbées** par certains matériaux (murs en béton, eau, etc.),
- **subir des interférences** avec d'autres signaux (Bluetooth, micro-ondes…).

3. Étude du Point d'Accès : Raspberry Pi 3 Model B

Notre point d'accès est une **Raspberry Pi 3 modèle B**, qui intègre une carte Wi-Fi fonctionnant uniquement sur la bande **2,4 GHz**. Elle utilise la norme **IEEE 802.11n**, qui permet d'atteindre des débits théoriques allant jusqu'à 150 Mbps dans des conditions optimales.

<u>Caractéristiques matérielles :</u>

- **Fréquence utilisée** : 2,4 GHz
- **Longueur d'onde associée** : $\lambda = \frac{c}{f} = \frac{3\times10^{8}}{2{,}4\times10^{9}} \approx 0{,}125$ m(12,5 cm)

Cela signifie que l'onde radio générée a une longueur d'onde d'environ 12,5 cm. C'est un paramètre important dans le dimensionnement d'antennes.

- **Antenne** : Intégrée sur le circuit imprimé (antenne PCB)
- **Puissance d'émission** : 16 dBm, soit **40 mW** ($P=10^{\frac{16}{10}} \approx 39.81$mW)
- **Gain estimé de l'antenne** : s'exprime en dbi, j'ai pris une valeur typique de **2 dBi**

4. Calcul de la Puissance Isotrope Rayonnée Équivalente (PIRE)

La **PIRE** représente la puissance que le système émettra si l'antenne était parfaitement isotrope (émettant dans toutes les directions de manière égale). Elle est calculée comme suit :

PIRE (dBm)= Puissance d'émission + Gain de l'antenne

PIRE= 16 dBm + 2 dBi = 18 dBm ⇒ ≈ 63 mW

C'est bien **en dessous de la limite réglementaire européenne** (20 dBm, soit 100 mW maximum pour la bande 2,4 GHz).

5. Estimation de la Portée du Signal

La distance maximale de communication peut être estimée avec le **modèle de propagation en espace libre** (équation de Friis). Pour une puissance de réception minimale acceptable de **-80 dBm** (valeur typique pour un signal Wi-Fi exploitable), et en prenant en compte :

On peut estimer la portée maximal du signal avec l'équation de Friis (FSL) :

Pr = Pt + Gt + Gr − 20log(d) − 20log(f) − 32,45

Où :

Pr : puissance reçue (en dBm, ex. seuil de -80 dBm pour bon signal)
Pt : puissance transmise (16 dBm)
Gt, Gr : gains des antennes (2 dBi)
f : fréquence en MHz (2400)
d : distance en km

On a donc :

20*log(d) = -Pr + Pt + Gt + Gr - 20log(f) - 32,45

$$d = 10^{\frac{-Pr + Pt + Gt + Gr - 20log(f) - 32{,}45}{20}} \approx 0.994\ km$$

La distance maximale de la portée du Wifi serait donc d'environ 1 km en **espace totalement Libre** *(sans réflexion, sans obstacle, sans mur, sans interférence…)* la distance "normale" serait d'environ **50 m** dans une situation classique.

V.1 - Itération 1 : Mise en place de l'environnement

➢ Installation et configuration du serveur Raspberry Pi

Avant toute chose, il est nécessaire de mettre en place l'environnement de travail sur la Raspberry Pi. Pour cela, j'ai installé un système d'exploitation via **Raspberry Pi Imager**, compatible avec ma version (Raspberry Pi 3 Model B+).

1. Activation du SSH

Une fois l'OS installé, j'effectue les premières configurations : activation du SSH pour le développement, modification de l'utilisateur et du mot de passe. J'active également la connexion SSH pour l'utilisateur root via le fichier **/etc/ssh/sshd_config** en ajoutant la ligne : **PermitRootLogin yes**.

2. Installation de l'IDE C++

J'installe ensuite l'IDE **Qt Creator** (version 4.8.2), que j'utiliserai pour le développement de l'application en C++. Une vérification rapide confirme son bon fonctionnement.

3. Installation et configuration de LAMP

Je procède à l'installation de la stack **LAMP** (Linux, Apache, MySQL, PHP) via la ligne de commande :
sudo apt install apache2 php mysql-server -y
Cela permet de disposer de tout le nécessaire pour le développement de l'interface web : Apache pour le serveur, MariaDB (MySQL) pour la base de données, et PHP pour le traitement côté serveur.

Je configure ensuite le serveur Apache pour qu'il interprète correctement les fichiers **.php**, en modifiant **/etc/apache2/mods-enabled/dir.conf** (ajout de **index.php** en début de ligne). J'active aussi l'affichage des erreurs PHP dans le fichier **php.ini** (**display_errors** et **log_errors** sur **On**), puis je redémarre Apache.

4. Préparation de la base de données

Je crée la base de données **ESTOM**, ainsi que deux utilisateurs :

- Un utilisateur autorisé à se connecter depuis l'extérieur (autre que **localhost**)
- Un second, dédié à l'application, avec uniquement les droits nécessaires sur les tables utiles

➢ Mise en place du hotspot Wi-Fi sécurisé (hostapd + dnsmasq)

Afin de permettre aux BOM, TOM et aux postes clients de se connecter au serveur Raspberry Pi **sans connexion Internet**, j'ai mis en place un point d'accès Wi-Fi sécurisé (**WPA2**) en utilisant **hostapd** et **dnsmasq**.
Pour cela, je me suis appuyé sur diverses documentations et tutoriels trouvés sur des sites spécialisés et des forums afin de bien comprendre chaque étape.

J'ai commencé par installer **hostapd**, qui permet de créer le point d'accès Wi-Fi, ainsi que **dnsmasq**, utilisé pour la configuration du serveur **DHCP**.

1. Configuration adresse statique :

Je commence par configurer l'adresse statique de la raspberry, que je vais définir de façon arbitraire comme suit dans /etc/dhcpcd.conf :

```
interface wlan0
static ip_address=192.168.0.10/26
nohook wpa_supplicant
```

J'ai choisi le masque 26 car ça me permet d'avoir au total 2^6=64 -2=62 adresse réseau, sachant qu'il m'en faut au minimum 45, car nous avons 40 TOM & 4 BOM + 1 tablette.

Comme j'utilise ici le service hostapd, ET que ma raspberry sera connecté en filaire (donc pas en Wifi) alors il peut y avoir des interférences avec wpa_supplicant, un outil utilisé pour gérer les connexions Wifi (en général pour ce connecter à des réseaux Wifi)
On utilise donc l'option nohook pour le désactiver, comme suit : nohook wpa_supplicant

2. Configuration du service DHCP :

Je souhaite ensuite configurer la plage dhcp suivante : 192.168.0.11 à 192.168.0.73 (62 adresses), pour se faire on va ensuite créer ou modifier le fichier de configuration ***/etc/dnsmasq.conf*** avec ceci :

```
interface=wlan0
dhcp-range=192.168.0.11,192.168.0.73,255.255.255.192,24h
```

Maintenant, les nouvelles demandes de connexion auront une ip attribuée dans la plage dhcp. Sous conseil de mes professeurs, on m'a recommandé de ne pas mettre une plage dhcp qui fait exactement 45 ip, mais un peu plus si besoin.

3. Configuration du point d'accès :

Le point d'accès Wifi peut maintenant être configuré, pour cela, comme prévu, je vais utiliser hostapd.

Je commence donc à créer le fichier ***/etc/hostapd/hostapd.conf*** et à y inclure les lignes suivantes recommandées sur la documentation de hostapd (option du point d'accès) :

```
interface=wlan0    # Nom de l'interface Wi-Fi
driver=nl80211     # Pilote utilisé (standard pour la plupart des interfaces modernes)
ssid=AP-ESTOM      # Nom du réseau Wi-Fi (SSID)
hw_mode=g          # Mode réseau (g = 802.11g)
channel=7          # Canal utilisé par le Wi-Fi (de 1 à 11)
wmm_enabled=0      # QoS (Qualité de service) désactivée
macaddr_acl=0      # Contrôle d'accès MAC désactivé (aucune whitelist)
auth_algs=1        # Algorithme d'authentification (mode ouvert avec WPA)
ignore_broadcast_ssid=0           # Rend le SSID visible (0 = visible, 1 = caché)
wpa=2                             # Utilisation du protocole WPA2
wpa_passphrase=Projet-ESTOM_2025  # Mot de passe du réseau Wi-Fi
wpa_key_mgmt=WPA-PSK              # Méthode de gestion des clés (clé prépartagée)
wpa_pairwise=TKIP    # Chiffrement WPA (optionnel, compatible mais moins sécurisé)
rsn_pairwise=CCMP    # Chiffrement WPA2 recommandé (plus sécurisé)
```

Ensuite bien préciser à hostapd d'utiliser le fichier de conf précédent, pour cela on va ajouter/modifier la ligne ***DAEMON_CONF="/etc/hostapd/hostapd.conf"*** dans le fichier ***/etc/default/hostapd***

On active ensuite les services avec les commandes suivantes :

sudo systemctl unmask hostapd
sudo systemctl enable hostapd dnsmasq
sudo systemctl start hostapd dnsmasq

Après vérification et test, je peux confirmer que mon point d'accès Wifi est bien opérationnel. Mon serveur sur la raspberry est donc prêt pour la suite de mes itérations.

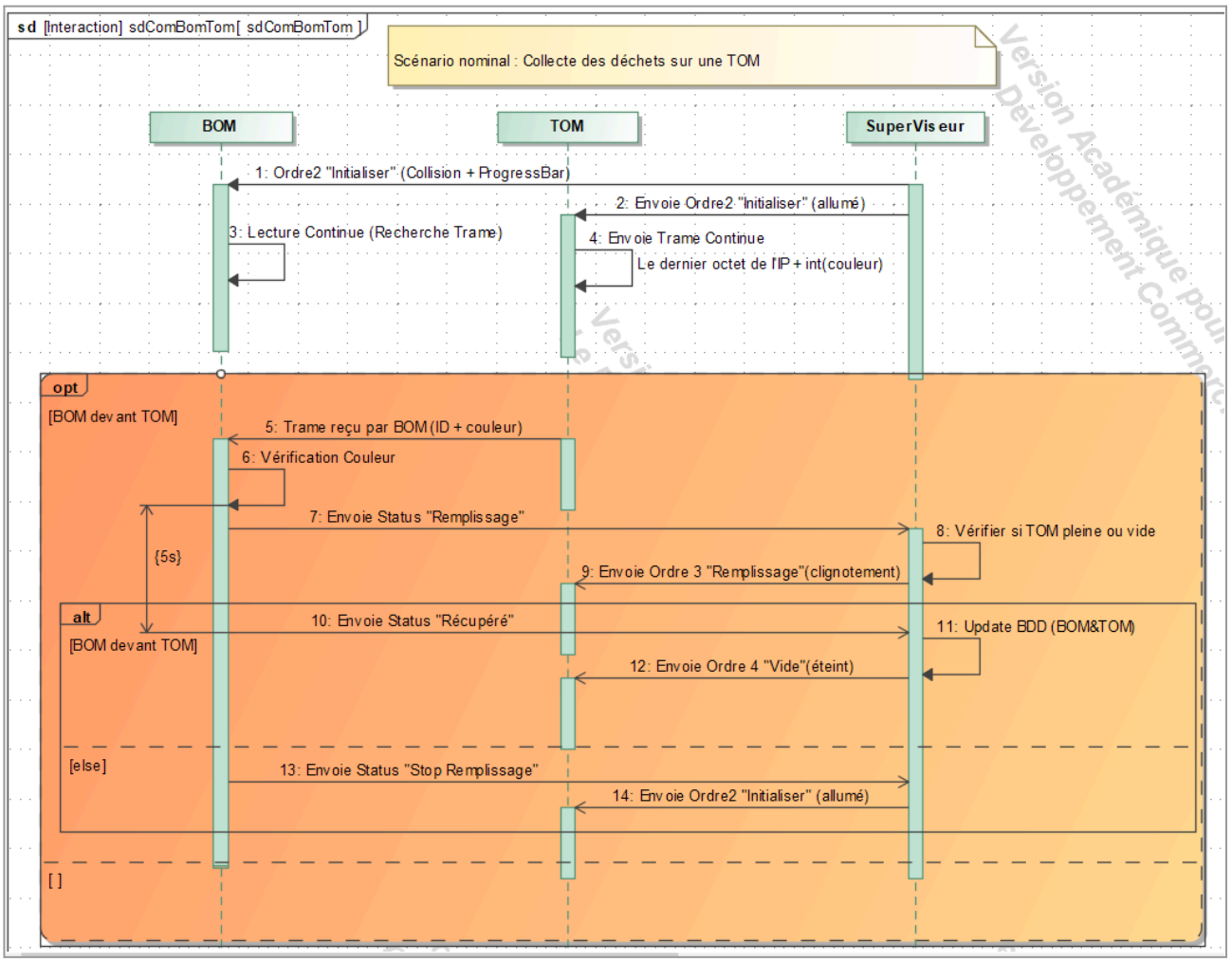
V.2 - Itération 2 : Développement de l'interface Web dynamique

➢ Conception du modèle relationnel dans MySQL Workbench

Pour commencer la conception d'une base de données, il y a plusieurs points à discuter, à savoir les besoins et les usages de celle-ci. Après Réflexion, voici ce que je peux dire sur les besoins de la base de données et ses usages.

1. Diagramme de Séquence BOM & TOM

Pour définir les besoins et les usages, j'ai tout d'abord réalisé un diagramme de séquence nominal qui décrit la communication entre la BOM et la TOM lors d'une collecte.



2. Table “Config” :

Ma base de données doit donc tout d’abord, contenir une Table “Config” avec les colonnes qui permettront de prendre en compte les différents paramètres de la partie. Nous devons donc faire en sorte que le **Malus de Collision** soit bien pris en compte, également la **durée du jeu** et les **deux Options** que j’ai décidé d'implémenter en bonus, et le **nombre de points gagné par récolte**.

Cette table doit également contenir le **Statut** du jeu, pour que l’application et l’ensemble des appareils connectés soient au courant de l’état du jeu (En cours, à l’arrêt, en pause), il doit également y avoir le **nombre de TOM par couleur,** ainsi que le **nombre de BOM de chaque couleur**.
Pour réaliser un “Timer” qui garde en tête le temps restant, j’ajoute aussi une colonne **Temps_restant** et **Heure_depart** dans ma table Config.

3. Table “BOM” :

Pour la table “BOM”, elle doit pouvoir contenir la **progression** de celle-ci, ainsi que sa **couleur**, elle enverra également un **statut** qui lui est propre, et que j’ai défini à l’aide de mon coéquipier :

1 = Information (la BOM envoie ces informations)
2 = Remplissage (la BOM commence un remplissage)
3 = Récupéré (la BOM vient de confirmer la récolte)
4 = StopRemplissage (la BOM viens de bouger, délais non respectés (5s))
5 = Collision/Choc (Nouvelle collision enregistrée)

Donc, d’après ces informations, il me faut également, une colonne qui indique le **nombre de Collisions**, ainsi que son **adresse IP**. Pour des raisons de maintenance et gestion, j’ai aussi pris l’initiative de rajouter une colonne **Connected** qui indique si l’appareil est connecté ou pas. Et il me faut en plus une colonne **Joueur** pour l’affichage de chaque joueur.

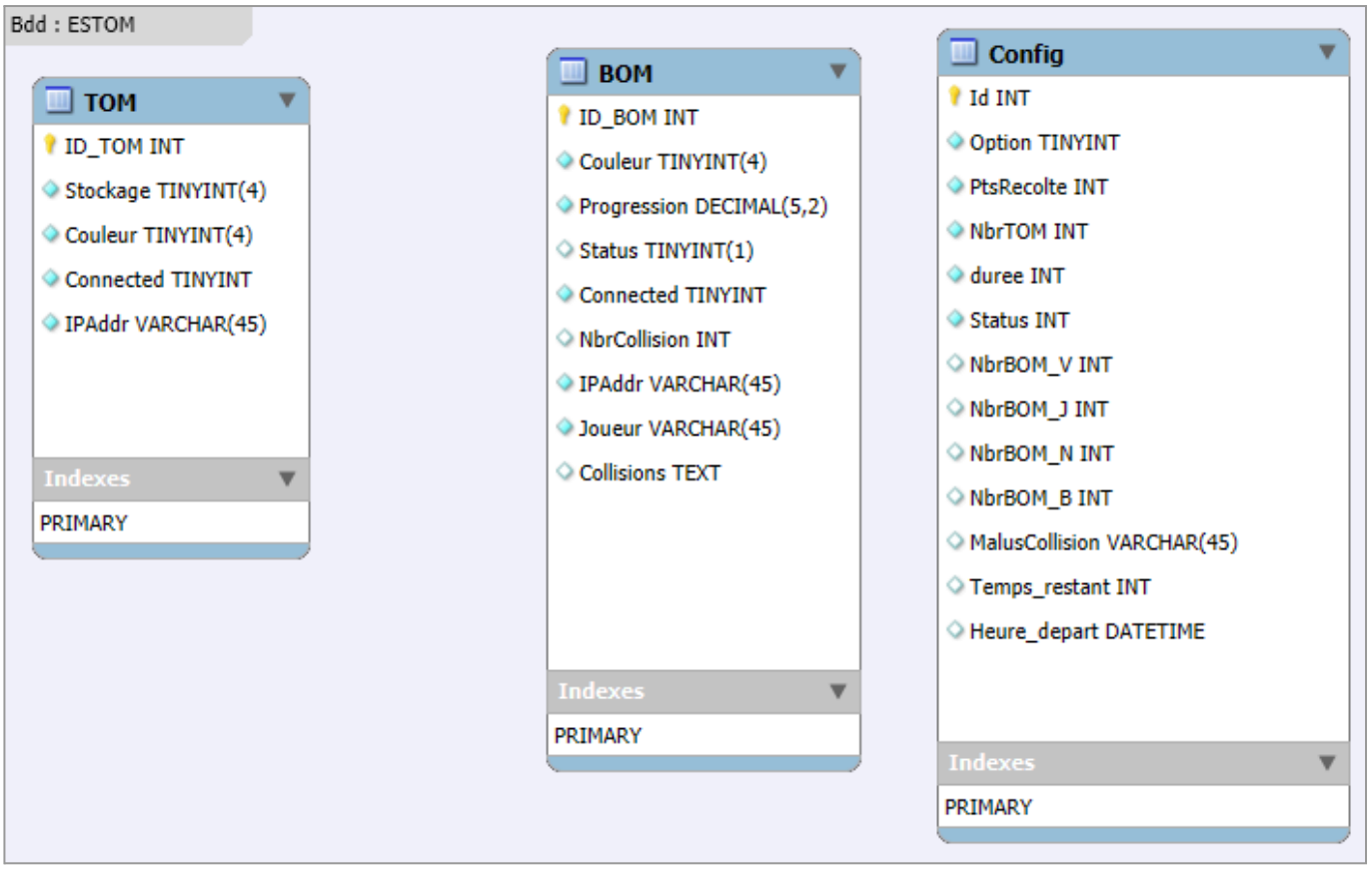
4. Table “TOM” :

La Table TOM doit contenir bien moins d'informations que les deux autres, il me faut juste leur **adresse IP**, leur **couleur**, et je dois pour l’affichage dynamique, avoir également l'information sur leur **stockage** et leur **connectivité.**

5. Modèles MCD & MLD

Après avoir déterminé les besoins et les usages, j'ai réalisé un Modèle MCD assez primitif sur papier. J'ai également déterminé qu'il n'y avait aucun "lien" entre chaque Table. Je devrais ne pas en avoir besoin pour mon site Web et mon application. Je modifierai cela si nécessaire dans le futur.

Avec toutes ces informations réunies, je peux donc réaliser le Modèle MLD.



➢ Création des pages PHP nécessaires

Pour le site Web, nous avons décidé (IR) de réaliser un page **d'accueil**, une page "**gestion de la partie**" pour l'affichage en temps réel de la partie (avec résultat) ainsi que le bouton de démarrage de la partie. Une page "**Paramétrage du jeu**" pour que le SuperViseur puisse paramétrer comme bon lui semble les parties est également nécessaire. Je décide aussi de faire les pages de mon site en PHP, le HTML n'est pas pratique dans mon cas, car je souhaite faire un site dynamique, et que les codes PHP sont difficilement implémentables dans une page HTML alors que le HTML, lui, est facilement utilisable dans une page PHP.

Grâce à l'IR 2 (Guilhem BUSSI), nous avons pu avoir quelques images à utiliser pour le style de notre site Web.
La totalité de mes fichiers PHP se trouve en Annexe de notre Projet (le CSS ne sera pas compris dans l'annexe..). Certaines parties de mes fichiers que je n'estime pas très intéressantes ne seront pas présentées ci-dessous.

1. CSS

Tout d'abord, pour le "Style" global de nos pages et de notre site, nous avons décidé de faire un petit effet visuel de vague en 3D… Voici une brique de CSS intéressante qui a nécessité quelques heures de recherche et de test.

```css
header {
    background: #00d77d;
    background-image: url("../../img/banniere.png");
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center;
    color: white;
    padding: 1rem;
    padding-top: 0;
    height: 5rem;
    text-align: center;
    position: relative;
    overflow: hidden;
    clip-path: polygon(100% 0%, 0% 0% , 0.00% 97.03%, 1.33% 96.32%, 2.67% 95.55%, 4.00% 94.74%, 5.33%
93.91%, 6.67% 93.08%, 8.00% 92.27%, 9.33% 91.49%, 10.67% 90.77%, 12.00% 90.12%, 13.33% 89.55%, 14.67%
89.08%, 16.00% 88.72%, 17.33% 88.48%, 18.67% 88.36%, 20.00% 88.36%, 21.33% 88.49%, 22.67% 88.74%,
24.00% 89.11%, 25.33% 89.58%, 26.67% 90.16%, 28.00% 90.81%, 29.33% 91.54%, 30.67% 92.32%, 32.00%
93.13%, 33.33% 93.96%, 34.67% 94.79%, 36.00% 95.60%, 37.33% 96.36%, 38.67% 97.08%, 40.00% 97.71%,
41.33% 98.27%, 42.67% 98.72%, 44.00% 99.06%, 45.33% 99.28%, 46.67% 99.38%, 48.00% 99.36%, 49.33%
99.21%, 50.67% 98.94%, 52.00% 98.55%, 53.33% 98.06%, 54.67% 97.47%, 56.00% 96.80%, 57.33% 96.06%,
58.67% 95.28%, 60.00% 94.46%, 61.33% 93.63%, 62.67% 92.80%, 64.00% 92.00%, 65.33% 91.24%, 66.67%
90.54%, 68.00% 89.92%, 69.33% 89.38%, 70.67% 88.95%, 72.00% 88.63%, 73.33% 88.42%, 74.67% 88.34%,
76.00% 88.39%, 77.33% 88.56%, 78.67% 88.85%, 80.00% 89.26%, 81.33% 89.76%, 82.67% 90.37%, 84.00%
91.05%, 85.33% 91.79%, 86.67% 92.59%, 88.00% 93.41%, 89.33% 94.24%, 90.67% 95.06%, 92.00% 95.86%,
93.33% 96.61%, 94.67% 97.30%, 96.00% 97.91%, 97.33% 98.43%, 98.67% 98.84%, 100.00% 99.15%);
    position: relative;
}
```

La fonctionnalité ***clip-path: polygon*** permet de découper un polygone et de l'appliquer par la suite. J'ai eu beaucoup de mal à réaliser ce que je souhaite, après quelques recherches, j'ai pu utiliser un simulateur que vous pouvez aller visiter au lien suivant : ***https://www.wavegenerator.app/***

Linear-gradient permet d'avoir l'effet 3D souhaité, également pris d'un simulateur CSS basique.

2. Fichier db_connect.php

Voici le fichier php que j'utiliserais dans mes prochains fichiers pour accéder à ma base de données. Celà m'éviterait à chaque fois d'écrire les vérifications et les informations.

```php
<?php
// Configuration de la base de données
$host = 'localhost';
$username = 'SuperViseur';
$password = 'SuperViseur-estom_2025';
$dbname = 'ESTOM';

// Création d'un objet MySQLi
$conn = new mysqli($host, $username, $password, $dbname);

// Vérification de la connexion
if ($conn->connect_error) {
    die("Erreur de connexion : " . $conn->connect_error);
}
?>
```

3. Fichier parametre-du-jeu.php

Je passe ensuite à la création de ma page paramètre, cela me sera plus pratique car comme ça je pourrais utiliser et valider les informations avant de passer à la fenêtre d'affichage (gestion), dans mon fichier je crée donc la récupération des différents paramètres utile de ma base de (table Config) en premier lieu.

```php
<?php
// Inclusion du fichier de connexion
include '/var/www/html/private-estom/db_connect.php';
// Initialisation des paramètres
$tempsPartie = 180;
$options = 1;
$ptsRecolte = 10;
$nbrTOM = 10;
$malusCollision = 2;
$nbrBOM_V = 1;
$nbrBOM_J = 1;
$nbrBOM_N = 1;
$nbrBOM_B = 1;
// Récupération des paramètres existants
$sql = "SELECT Options, PtsRecolte, NbrTOM, MalusCollision, NbrBOM_V, NbrBOM_J, NbrBOM_N, NbrBOM_B,
duree FROM Config WHERE Id = 1";
$result = $conn->query($sql);
if ($result && $result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $options = $row['Options'];
    $ptsRecolte = $row['PtsRecolte'];
    $nbrTOM = $row['NbrTOM'];
    $malusCollision = $row['MalusCollision'];
    $nbrBOM_V = $row['NbrBOM_V'];
    $nbrBOM_J = $row['NbrBOM_J'];
    $nbrBOM_N = $row['NbrBOM_N'];
    $nbrBOM_B = $row['NbrBOM_B'];
    $tempsPartie = $row['duree'];
}
```

Ensuite, je crée le formulaire nécessaire pour présenter les différents paramètres.

Voici ci-dessous un bref exemple, j'utilise ici la méthode "POST" qui me permettra via php de "récupérer" les valeurs de mon formulaire une fois envoyé/validé.

```php
<div id="Principale">
    <form method="post" action="parametre-du-jeu.php" id="BoiteRegles">
        <fieldset id="fieldset">
            <legend id="titre">Règles du Jeu</legend>

            <div class="Regle-Container">

                <label for="temps_partie" id="temps_partie">Temps de la partie (en secondes) :</label>
                <input type="number" id="temps_partie_input" name="temps_partie" value="<?php echo
htmlspecialchars($tempsPartie); ?>" step="10" min="30" max="300" required>

                <label for="nombre_tours" id="nombre_tours">Nombre de TOM par couleur :</label>
                <input type="number" id="nombre_tours_input" name="nombre_tours" value="<?php echo
htmlspecialchars($nbrTOM); ?>" min="5" max="10" required>

                <label for="malus_collision" id="malus_collision">Malus de collision :</label>
                <input type="number" id="malus_collision_input" name="malus_collision" value="<?php
echo htmlspecialchars($malusCollision); ?>" step="0.2" min="0.2" max="5" required>
```

Je mets également les conditions d'insertion dans mon formulaire, pour éviter tout problème de saisie, donc limitation de texte si besoin, "plage" d'insertion de nombre, pour éviter les débordements, et le "required" qui permet de rendre obligatoire la saisie de valeur.

Voici ci-dessous une partie PHP qui permet de récupérer les valeurs de POST :

```php
// Si le formulaire est soumis
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
// Mise à jour du temps de la partie
// Validation des entrées utilisateur avec double vérification des valeurs!
$nouveauTempsPartie = filter_var($_POST['temps_partie'], FILTER_VALIDATE_INT, [
    'options' => ['min_range' => 30, 'max_range' => 300]
]) ?? $tempsPartie;

$nouveauNbrTOM = filter_var($_POST['nombre_tours'], FILTER_VALIDATE_INT, [
    'options' => ['min_range' => 5, 'max_range' => 10]
]) ?? $nbrTOM;
$nouveauMalusCollision = filter_var($_POST['malus_collision'], FILTER_VALIDATE_FLOAT);
if ($nouveauMalusCollision === false || $nouveauMalusCollision < 0.2 || $nouveauMalusCollision > 5) {
    $nouveauMalusCollision = $malusCollision;
}
. . .

// Mise à jour dans la base de données
$sqlUpdate = "UPDATE Config SET Options = ?, PtsRecolte = ?, NbrTOM = ?, MalusCollision = ?, NbrBOM_V =
?, NbrBOM_J = ?, NbrBOM_N = ?, NbrBOM_B = ?, duree = ? WHERE Id = 1";
$stmt = $conn->prepare($sqlUpdate);
$stmt->bind_param("iiidiiiii", $nouveauOptions, $nouveauPtsRecolte, $nouveauNbrTOM,
$nouveauMalusCollision, $nouveauNbrBOM_V, $nouveauNbrBOM_J, $nouveauNbrBOM_N, $nouveauNbrBOM_B,
$nouveauTempsPartie);
$stmt->execute();

header("Location: parametre-du-jeu.php?success=1");
exit;
}
```

J'ai également fait en sorte de sécuriser mes requêtes sql en les préparants avant l'injection. Cela permet d'éviter les failles de sécurité de type Injection SQL.

J'ai également mis dans ce formulaire en place un script (JavaScript) qui permet de faire "disparaître" la saisie de ptsRecolte si l'option B est choisie.

```javascript
<script>
    // Fonction pour gérer la visibilité du champ "Pts par Tour"
    function togglePointsField() {
        const optionAInput = document.querySelector('input[name="regle"][value="option_a"]');
        const pointsField = document.getElementById("optionA_settings");

        // Afficher ou masquer le champ selon l'option sélectionnée
        if (optionAInput.checked) {
            pointsField.style.display = "block";
        } else {
            pointsField.style.display = "none";
        }
    }

    // Ajout d'événement
    document.addEventListener("DOMContentLoaded", () => {
        // Exécuter une première fois pour régler l'affichage initial
        togglePointsField();

        // Ajouter des "écouteurs" sur les options de règle
        document.querySelectorAll('input[name="regle"]').forEach(input => {
            input.addEventListener("change", togglePointsField);
        });
    });
</script>
```

J'ai aussi fait à côté de mon formulaire de paramètre, une fenêtre qui affiche les BOM et les TOM connectés, avec leur ip respective, les images se "grisent" une fois qu'elles ne sont plus connectées. C'est grâce à un ping réalisé toutes les X secondes avec mon application que je mets à jour le connected de ma base de données. Ce qui permet d'avoir une mise à jour toutes les 2,5 secondes de "l'état" de connexion de mes appareils.

J'ai décidé de faire cela pour une meilleure gestion des appareils connectés.

Voici la partie php (affichage) :

```
<div id="IconesParametre">
    <?php
    $couleurs = [
        "Vert" => $nbrBOM_V,
        "Jaune" => $nbrBOM_J,
        "Noir" => $nbrBOM_N,
        "Bleu" => $nbrBOM_B
    ];
    $maxBOM = max($nbrBOM_V, $nbrBOM_J, $nbrBOM_N, $nbrBOM_B);
    $position = 1; // Position de chaque colonne dans la grille

    foreach ($couleurs as $couleur => $nombre) {
        echo '<div id="ColoneBOM' . $couleur . '" style="grid-column: ' . $position . '; grid-row:
1;">';

        // Afficher 1 ou 2 BOM par couleur
        for ($i = 0; $i < $maxBOM; $i++) {
            echo '<div id="BOM-' . $couleur . '-' . $i . '" class="bom-container">
                    <img src="./img/BOM-' . $couleur . '.png" alt="Camion ' . $couleur . '" id="BOM-' .
$couleur . '-' . $i . '" class="grise" width="75" height="45">
                    <span id="BOM-IP-' . $couleur . '-' . $i . '" class="ip-text"></span>
                    </div>';
        }
        echo '</div>';
        echo '<div id="ColoneTOM' . $couleur . '" style="grid-column: ' . $position . '; grid-row:
2;">';
        for ($i = 0; $i < 10; $i++) {
            echo '<div class="LigneTOM">
                    <img src="./img/TOM-' . $couleur . '.png" alt="TOM ' . $couleur . '" id="TOM-' .
$couleur . '-' . $i . '" class="grise imageTOM" width="29" height="45">
                    <span id="TOM-IP-' . $couleur . '-' . $i . '" class="ip-text"></span>
                </div>';
        }
        echo '</div>';
        $position++; // Passe à la colonne suivante
    }
    ?>
</div>
```

Voici une partie du script Java associé à cette partie (AJAX) :

```
function updateConnexion() {
    fetch('get_connexionIP.php')
        .then(response => response.json())
        .then(data => {
            console.log("Données reçues :", data); // DEBUG

            let currentState = {}; // Stocke l'état actuel
            let tomUsedIPs = {}; // Suivi des IPs déjà attribuées pour éviter les doublons
            let tomCount = { "Vert": 0, "Jaune": 0, "Noir": 0, "Bleu": 0 };

            // Réinitialiser les IP affichées
            document.querySelectorAll('.ip-text').forEach(span => {
                span.textContent = 'N/A';
            });

            data.forEach(entry => {
                let couleur = entry.couleur;
                let ip = entry.ip;
                let type = entry.source; // "BOM" ou "TOM"
                let isConnected = entry.connected === 1;
```

L'utilisation **d'AJAX** permet de réaliser une action de façon asynchrone par rapport au code, ce qui permet son exécution sans attendre le reste du code. C'est la méthode la plus **répandue** et utilisée pour éviter de rafraichir une page manuellement (pas besoin de rafraîchissement pour actualiser la page). C'est ce qui rend un site Web **dynamique**.

Dans l'exemple ci-dessus, je fait appelle à la fonction **fetch** et la méthode .**then** qui permet d'attendre le résultat asynchrone de la requête avant d'effectuer une action.

Cette méthode prend une fonction en paramètre, dont le seul paramètre est le résultat de la promesse précédente. Ici je récupère donc les données (DATA) du JSON précédemment demandé. Ces données récupérées permettent d'avoir les informations suffisantes envoyées depuis un autre fichier php.

J'envoie par exemple en JSON le type/la table (BOM ou TOM), la couleur, l'ip (important pour rendre unique chaque image) et le "connected" (pour griser/mettre en couleur l'image).

Voici le fichier get_connexionIP.php (partie qui stocker les données utiles) :

```
foreach ($tables as $table) {
    $sql = "SELECT IPAddr, Connected, Couleur FROM $table"; // On boucle sur chaque table
    $result = $conn->query($sql);

    if ($result && $result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            // Extraire les deux derniers octets de l'adresse IP
            $ip_parts = explode('.', $row['IPAddr']);
            $short_ip = isset($ip_parts[2]) && isset($ip_parts[3]) ? $ip_parts[2] . '.' . $ip_parts[3]
: '??.??';
            //$short_ip = isset($ip_parts[3]) ? $ip_parts[3] : '??';


            $couleur = isset($couleurs[$row['Couleur']]) ? $couleurs[$row['Couleur']] : "Inconnu";

            $data[] = [
                'source' => $table, // Indique si l'IP vient de BOM ou TOM
                'couleur' => $couleur,
                'ip' => $short_ip,
                'connected' => (int)$row['Connected']
            ];
        }
    }
}
```

Une fois l'ensemble de ces informations fournies, j'ai plus qu'à attribuer de façon "automatique" un id uniquement à chaque image pour l'identifier avec précision, et ensuite enlever le filtre .grise (CSS) qui grise automatiquement toutes les images.

A la fin de mon script, après la fonction qui met à jour l'affichage (d'AJAX), je refais appel à mon script après un certain temps grâce à un setInterval :

```
// Rafraîchir les données toutes les 2.5 secondes
setInterval(updateConnexion, 2500);
```

4. Fichier gestion-de-la-partie.php

Le Fichier de ma page d'affichage des résultats, et de démarrage de la partie est essentiel au projet, c'est la partie immergée de l'iceberg pour les joueurs, elle est donc très importante.

Sur cette page, il faut un moyen de lancer la partie de façon simple et efficace, un moyen d'ajouter/configurer les joueurs présents, et surtout, un moyen d'afficher de façon sympathique les résultats et l'avancée globale du parcours avec les points.
Après réflexion, et petite réunion entre IR, nous avons discuté sur ces points importants. J'ai donc émis l'idée de mettre une représentation visuelle de l'avance de chaque joueur. Une sorte de barre de progression.

L'affichage consiste à afficher le scoreboard (Tableau des Score individuel) à gauche de chaque "parcours" par joueur. L'affichage est basé en "ligne", chaque joueur à sa "ligne", au centre/milieu, il y a une image de route, pour un effet visuel plus sympathique, ensuite j'ai disposé de manière automatique, les TOM en fonction de leurs nombres. Il y a également une animation qui permet de faire avancer la BOM en fonction de sa progression, et les TOM sont grisées au moment où la BOM la "récolte", en se basant sur sa progression.

La **gestion des joueurs via une fenêtre modale** a été choisie, j'ai trouvé cette solution simple et efficace, sans nécessité de grand changement dans ma page. Lorsqu'un utilisateur clique sur le bouton « Ajouter Joueurs », une modale s'affiche pour saisir les pseudos. Voici un extrait qui montre l'ouverture de la modale et l'insertion dynamique des champs via PHP :

```
<!-- Fenêtre modale -->
<div id="playerModal" class="modal">
    <div class="modal-content">
        <span class="close-btn">&times;</span>
        <h2>Ajoutez vos joueurs</h2>
        <form id="playerForm">
            <div id="playerInputs">
                <?php

                [ . . . ]

                // Récupération des BOM avec leur ID, couleur et joueur
                $sql = "SELECT ID, Couleur, Joueur FROM BOM ORDER BY Couleur, ID";
                $result = $conn->query($sql);

                // Conversion explicite en entiers
                $nbrBOMConfig2 = (int) $nbrBOMConfig;
                $nbrBOMEnregistrees2 = (int) $nbrBOMEnregistrees;

                if ($result && $result->num_rows > 0) {
                    while ($row = $result->fetch_assoc()) {
                        $bomID = $row['ID'];
                        $couleur = $colorMap[$row['Couleur']];
                        $joueur = $row['Joueur'] ?? "";

                        echo "<div class='player-input'>
                                <span>BOM#$bomID - $couleur</span>
                                <input type='text' name='joueurs[$bomID]' value='$joueur'
maxlength='20' placeholder='Pseudo joueur' required>
                            </div>";
                    }
                }
                // Affichage du message d'alerte si le nombre de BOM est incorrect
                if ($nbrBOMConfig2 !== $nbrBOMEnregistrees2) {
                    echo "<p style='color: red; font-weight: bold; margin-top: 10px;'>
                            ⚠ Attention : La configuration prévoit <strong>$nbrBOMConfig2
BOM</strong>,
                            mais <strong>$nbrBOMEnregistrees2 BOM</strong> sont enregistrées !
                        </p>";
                }
                ?>
            </div>
            <button type="submit">Valider</button>
        </form>
    </div>
</div>
```

Voici le script Associé (AJAX) pour la fenêtre Modale :

```
<script>
    document.addEventListener("DOMContentLoaded", function () {
        // Récupération des éléments important
        const modal = document.getElementById("playerModal");
        const openModalButton = document.getElementById("openModalButton");
        const closeModalButton = document.querySelector(".close-btn");
        const playerForm = document.getElementById("playerForm");

        // Ajouter un écouteur d'événement sur le bouton d'ouverture pour afficher la modale
        openModalButton.addEventListener("click", function () {
            modal.style.display = "flex";
        });

        // Ajouter un écouteur d'événement sur le bouton de fermeture pour cacher la modale
        closeModalButton.addEventListener("click", function () {
            modal.style.display = "none";
        });

        // Intercepter la soumission du formulaire pour envoyer les données en AJAX
        playerForm.addEventListener("submit", function (event) {
            event.preventDefault();

            // Envoyer les données du formulaire en POST vers update_joueurs.php
            const formData = new FormData(playerForm);
            fetch('update_joueurs.php', {
                method: 'POST',
                body: formData
            })
            .then(response => response.text()) // Récupération de la réponse.
            .then(() => {
                modal.style.display = "none";
                location.reload(); // Recharger la page pour maj.
            })
            .catch(error => console.error("Erreur :", error));
        });
    });
</script>
```

Et voici mon fichier **update_joueurs.php** :

```
include '/var/www/html/private-estom/db_connect.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    foreach ($_POST['joueurs'] as $bomID => $pseudo) {
        $pseudo = substr(trim($pseudo), 0, 20); // Sécurisation (max 20 caractères)
        $sql = "UPDATE BOM SET Joueur = ? WHERE ID = ?";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("si", $pseudo, $bomID);
        $stmt->execute();
    }
}

echo "Mise à jour réussie !";
```

Ce mécanisme permet d'actualiser les pseudos en envoyant les données via une requête **fetch** à **update_joueurs.php**, ce qui rend la gestion des joueurs fluide et interactive.

Un autre aspect intéressant est l'**affichage dynamique du timer**. La page dispose d'un élément dédié qui affiche un message de démarrage, puis se transforme en compte à rebours grâce à JavaScript. Les fonctions suivantes illustrent comment le timer est mis à jour chaque seconde, je n'ai pas mis l'ensemble des fonctions liées par souci de place, je vous renvoie en Annexe pour le code complet :

```
function startCountdown(duration) {
    const timerDiv = document.getElementById('timer');
    remainingTime = duration; // Mise à jour de la valeur de remainingTime

    function updateTimer() {
        const minutes = Math.floor(remainingTime / 60);
        const seconds = remainingTime % 60;
        timerDiv.textContent = `Temps restant : ${minutes}m ${seconds}s`;

        if (remainingTime > 0) {
            remainingTime--; // Met à jour la variable restante
        } else {
            timerDiv.textContent = "Temps écoulé !";
            clearInterval(timerInterval);
            isTimerRunning = false;
            updateDatabaseStatus('stopped');
            updateButton(3); // "Lancer le timer"
        }
    }
    updateTimer();
    timerInterval = setInterval(updateTimer, 1000);
}
// Fonction AJAX pour récupérer l'état actuel
window.addEventListener('DOMContentLoaded', () => {
    fetch('get_timer_state.php')
    .then(response => response.json())
    .then(data => {
        if (data.error) {
            console.error(data.error);
            return;
        }
        const status = data.status;
        remainingTime = parseInt(data.temps_restant) || timerDuration;

        switch (status) {
        case 'running':
            isTimerRunning = true;
            startCountdown(remainingTime); // Appel avec la valeur correcte
            updateButton(1);
        break;
        case 'paused':
            isTimerRunning = false;
            const minutes = Math.floor(remainingTime / 60);
            const seconds = remainingTime % 60;
            document.getElementById('timer').textContent = `Temps restant : ${minutes}m ${seconds}s`;
            updateButton(2);
        break;
        case 'loading':
            document.getElementById('timer').textContent = `Chargement de la partie...`;
            updateButton(1);
        break;
        case 'stopped':
        default:
            isTimerRunning = false;
            document.getElementById('timer').textContent = "Cliquez sur le bouton pour lancer la Partie
!";
            updateButton(3);
        break;
        }
    })
    .catch(error => {
        console.error("Erreur lors de la récupération du timer :", error);
    });
});
```

Ce script assure que le temps restant est affiché en temps réel, avec des appels à la base de données pour mettre à jour l'état du jeu, la gestion de l'affichage des différents boutons est également gérée dans ce script.

Enfin, le fichier gère de plus **l'animation des éléments de jeu**. Par exemple, pour chaque BOM, le script récupère sa progression et ajuste sa position sur l'écran afin de refléter visuellement l'avancement sur la route. Un **extrait** de code montre comment la position est animée.

```
data.forEach((camionData, index) => {
    let ip = camionData.ip;
    let totalTOMs = nbrTOM; // Nombre total de tours (TOM) sur la route
    let tomsPasses = Math.round(camionData.progression / (100 / (totalTOMs))); // Nombre de TOM
atteints (dépassé)
    let progressionPourcentage = Math.min(100, camionData.progression); //Progress de la BOM (%)
    let nbrCollision = camionData.nbrCollision ?? 0; //Nbr Collision par BOM
    let collisions = camionData.collisions; //Coordonée des collisions
    let camion = document.querySelector(`.imageCamion[data-couleur="${couleur}"][data-ip="${ip}"]`);

    if (!camion) return; // Si le camion n'existe pas, on ignore
    let distance = Math.abs(progressionPourcentage - camion.dataset.prevProgression || 0);
    // Définir la durée en fonction de la distance
    let animationDuration;
    if (distance <= 10) {
        animationDuration = 2000 + (distance * 100); // Entre 2 et 3 secondes pour les petites distances
    } else if (distance >= 50) {
        animationDuration = 500 + (distance * 10); // Rapide si distance grande
    } else {
        animationDuration = 1500 + (distance * 20); // Progression moyenne
    }
    // Appliquer la transition fluide
    camion.style.transition = `left ${animationDuration / 1000}s linear`;
    camion.style.left = `${progressionPourcentage}%`;
    // Sauvegarder la dernière progression
    camion.dataset.prevProgression = progressionPourcentage;
```

Ce mécanisme permet une animation **fluide** qui rend l'interface plus **immersive**, tout en mettant à jour **dynamiquement** les scores et en gérant des aspects comme le « grisage » des TOM en fonction de la progression. La partie complète du code est disponible en Annexe !

Voici également un extrait de la partie qui calcule le score en fonction de l'option choisie :

```
// Mise à jour du score
let scoreText = document.getElementById(`score-${ip}`);
if (optionMode === 0) {
    // OPTION A : Score fixe par TOM
    scoreBrut = tomsPasses * ptsRecolte;
    maxPossibleScore = totalTOMs * ptsRecolte; // Score max basé sur le nombre de TOM et ptsRecolte
} else if (optionMode === 1) {
    // OPTION B : Score progressif
    let basePts = ptsRecolte * 0.6; // La première TOM rapporte 60% du score de base
    let increment = (ptsRecolte * 1.4 - basePts) / (totalTOMs - 1); // Augmente progressivement

    for (let i = 0; i < tomsPasses; i++) {
        scoreBrut += basePts + (i * increment); // Ajoute des points de manière progressive
    }

    // Calcul du score max théorique en Option B
    for (let i = 0; i < totalTOMs; i++) {
        maxPossibleScore += basePts + (i * increment);
    }
}
// Appliquer le malus des collisions
scoreBrut -= nbrCollision * malusCollision;
scoreBrut = Math.max(0, scoreBrut); // Empêcher un score négatif
// Affichage sans normalisation, mais avec une limite dynamique
scoreBrut = Math.min(maxPossibleScore, Math.max(0, scoreBrut));
scoreText.textContent = `${Math.round(scoreBrut)}/${Math.round(maxPossibleScore)}`;
```

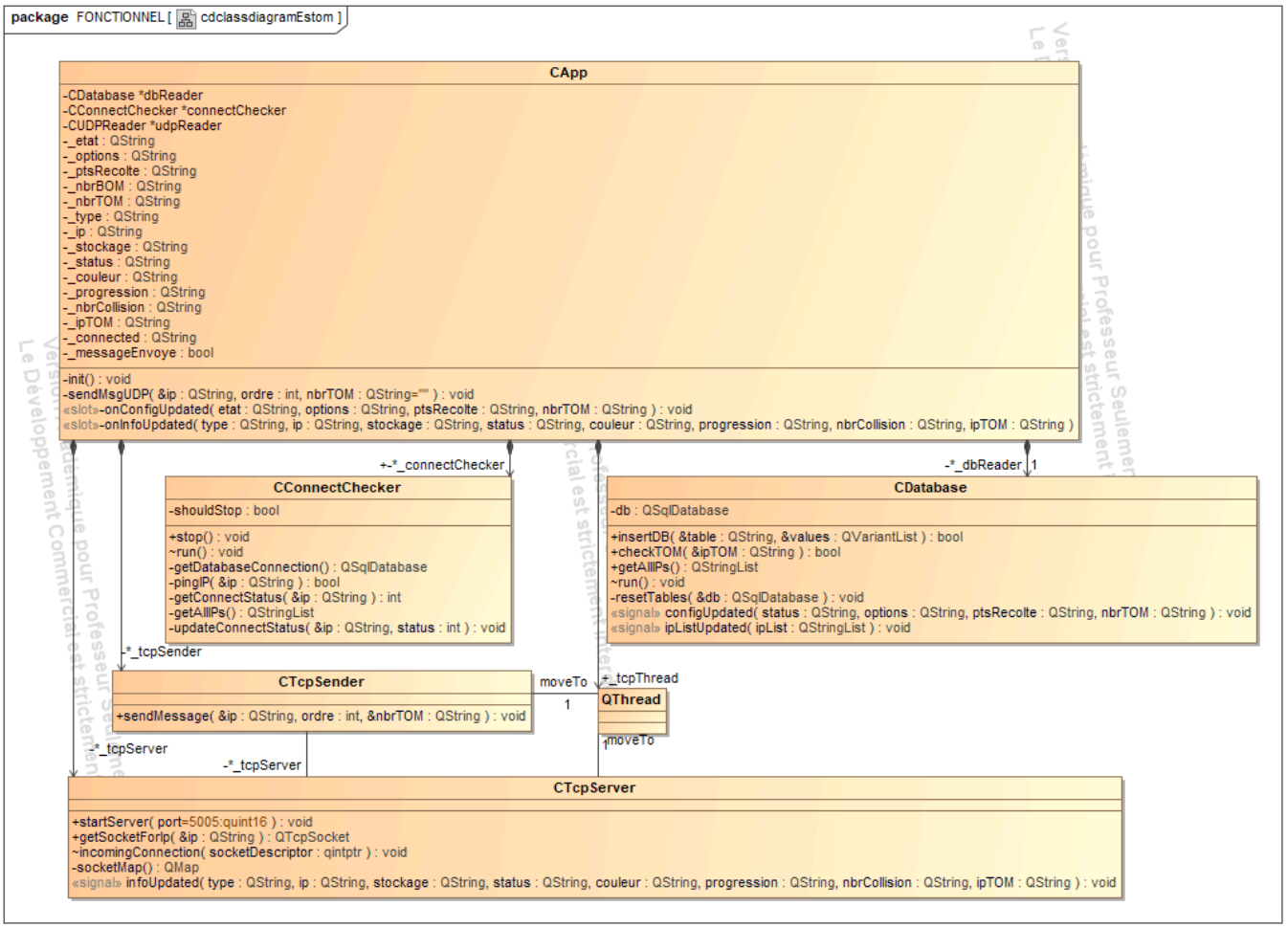
Ensuite, la page gère également l'affichage de petite image de "**Choc**" lorsqu'une **collision** a lieu, elles sont placées de façon **aléatoire** autour de la BOM. Les coordonnées sont ensuite enregistrées au format JSON dans la base de données pour les récupérer facilement en cas de problème sur le site Web (S'il faut actualiser la page par exemple).

Le reste de mon site Web et l'ensemble des fichiers est disponible en Annexe si besoin, après des vérifications à différents moments de mon développement, le site est maintenant fonctionnel (tests réalisés avec succès).

V.3 - Itération 3 : Développement de l'application et interaction avec l'interface Web.

➢ Conception du diagramme de Classe de l'application

La création d'un diagramme de classes s'est révélée essentielle pour bien comprendre la structure et le fonctionnement de mon application. Après plusieurs ébauches réalisées sur papier, j'ai abouti au diagramme suivant, qui présente les différentes classes, leurs méthodes, ainsi que leurs relations.



➢ Réalisation des différentes Classes

La création de l'application, destinée à assurer la liaison entre les différents appareils connectés du jeu (BOM & TOM) et le site web, nécessite d'abord une classe principale. Celle-ci fait office de point central entre les différentes classes du projet. Je vais donc présenter quelques extraits de code issus de certaines pages, que je juge particulièrement pertinents.

Le développement de l'application a commencé par la création de la classe **Cdatabase.cpp,** responsable de la communication avec la base de données.

1. Classe CDatabase.cpp :

Le code permettant la connexion à la base de données est assez simple, et elle doit surtout permettre la récupération de toutes les données de la Table Config, pour faire ensuite fonctionner l'ensemble de mon code. Je dois donc récupérer les valeurs de cette Table.

```
void CDatabase::run()
{
        [ . . . ]
    while (!isInterruptionRequested()) {
        QSqlQuery query(threadDb);
        if (!query.exec("SELECT * FROM Config")) {
            qDebug() << "Erreur SELECT Config :" << query.lastError().text();
            QThread::sleep(1);
            continue;
        }

        if (!query.next()) {
            qDebug() << "La table Config est vide ou aucune ligne trouvée.";
            QThread::sleep(1);
            continue;
        }
        QString Status = query.value("Status").toString();
        QString Options = query.value("Options").toString();
        QString PtsRecolte = query.value("PtsRecolte").toString();
        QString NbrBOM_V = query.value("NbrBOM_V").toString();
        QString NbrBOM_J = query.value("NbrBOM_J").toString();
        QString NbrBOM_N = query.value("NbrBOM_N").toString();
        QString NbrBOM_B = query.value("NbrBOM_B").toString();
        QString NbrTOM = query.value("NbrTOM").toString();
        emit configUpdated(Status, Options, PtsRecolte, NbrTOM);

        QThread::sleep(1);  // Pause de 5 secondes entre chaque lecture
    }
    threadDb.close();
    QSqlDatabase::removeDatabase("CDatabaseConnection");
}
```

Afin que mon application puisse connaître l'état du jeu, j'ai mis en place une méthode de récupération du statut, exécutée toutes les secondes. Cela permet de maintenir une **synchronisation** régulière des informations.
Ma classe de gestion de base de données permet l'insertion ainsi que la mise à jour des tables et colonnes. Ces opérations étant assez classiques, je vous invite à consulter le code en annexe si vous souhaitez en savoir plus.
Pour garantir un bon fonctionnement au démarrage de l'application, un nettoyage de la base est effectué : les tables **BOM** et **TOM** sont vidées afin de réinitialiser proprement les adresses IP des appareils connectés.
Avant d'aborder la classe **TCP**, il est important de noter qu'au début du projet, nous avions opté pour une communication via le protocole **UDP**. Cependant, nous avons finalement choisi **TCP** pour sa fiabilité : ce protocole garantit la bonne réception des messages, contrairement à UDP où cette vérification devait être faite manuellement.

2. Communication TCP :

La classe **TCPServer** est responsable de la création du serveur au lancement de l'application. Elle permet ensuite aux différents appareils de s'y connecter, ce qui en fait l'une des classes centrales de notre projet. Elle gère également la réception des messages, que nous avons choisi de structurer au format JSON. Ce format, basé sur un système de clés/valeurs, facilite l'extraction des données utiles, l'envoi des ordres, ainsi que la réception des statuts ou informations des différents appareils.

Je vous renvoie au diagramme de séquence réalisé et expliqué dans l'itération 2 pour mieux comprendre notre choix.

Voici donc un extrait du code qui permet de "décoder" les valeurs reçues par les appareils :

```
connect(clientSocket, &QTcpSocket::readyRead, this, [=]() {
    buffer->append(clientSocket->readAll());

    if (!buffer->contains('\n')) return;
    QList<QByteArray> messages = buffer->split('\n');

    for (int i = 0; i < messages.size() - 1; ++i) {
        QByteArray message = messages[i].trimmed();
        if (message.isEmpty()) continue;

        QJsonParseError parseError;
        QJsonDocument doc = QJsonDocument::fromJson(message, &parseError);

        QJsonObject jsonObj = doc.object();

        QString stockage = jsonObj.value("Stockage").toString();
        QString couleur = jsonObj.value("Couleur").toString();
        QString progression = jsonObj.value("Progression").toString();
        QString nbrCollision = jsonObj.value("NbrCollision").toString();
        QString status = jsonObj.value("Status").toString();
        QString ipTOM = jsonObj.value("ipTOM").toString();
        QString type = jsonObj.value("Type").toString();

        QString senderIP = clientSocket->peerAddress().toString();
        if (senderIP.startsWith("::ffff:")) {
            senderIP = senderIP.right(senderIP.length() - 7);
        }
```

Nous avons rencontré un problème important lors de nos tests : bien que les appareils parvenaient à se connecter au serveur, l'envoi des messages posait problème de mon côté. J'ai mis un certain temps à comprendre le fonctionnement des **sockets TCP**. Pour envoyer un message à un client spécifique, il fallait utiliser le même socket que celui utilisé lors de la connexion. J'ai donc dû **stocker** ces sockets dans une map (socket map), en les associant à l'adresse IP du client, afin de pouvoir gérer correctement le système d'ordres.

```
        // Adresse IP du client connecté
        QString senderIP = clientSocket->peerAddress().toString();
        if (senderIP.startsWith("::ffff:")) {
            senderIP = senderIP.right(senderIP.length() - 7);
        }
        qDebug() << "Nouvelle connexion TCP de:" << senderIP;

        // Stockage temporaire uniquement si socket est en ConnectedState
        if (clientSocket->state() == QAbstractSocket::ConnectedState) {
            socketMap[senderIP] = clientSocket;
        } else {
            connect(clientSocket, &QTcpSocket::connected, this, [=]() {
                qDebug() << "Connexion confirmée pour:" << senderIP;
                socketMap[senderIP] = clientSocket;
            });
        }
```

Une fois le socket stocké et associé à l'adresse IP correspondante, j'ai pu créer ma classe **TCPSender**, chargée de l'envoi des ordres. Ci-dessous, un extrait de code illustrant ce fonctionnement :

```
void CTcpSender::sendMessage(const QString &ip, int ordre, const QString &nbrTOM)
{
    qDebug() << "<=====|TCP-SENDER-" << ip <<"|=====>";
    QTcpSocket *socket = m_tcpServer->getSocketForIP(ip);

    if (!socket || socket->state() != QAbstractSocket::ConnectedState) {
        qWarning() << "Socket non disponible ou déconnecté pour l'IP:" << ip;
        return;
    }

    QJsonObject jsonObj;
    jsonObj["Ordre"] = ordre;
    if (ordre == 2) {
        qDebug() << "Ordre = 2 donc envoie nbrTOM :" << nbrTOM;
        jsonObj["nbrTOM"] = nbrTOM;
    }
    QJsonDocument doc(jsonObj);
    QByteArray message = doc.toJson(QJsonDocument::Compact) + '\n';

    socket->write(message);
    socket->flush();
    qDebug() << "Message TCP envoyé à" << ip;
    qDebug() << "Données:" << message;
}
```

La méthode sendMessage est appelée dans la classe principal CApp.cpp qui suit.

3. Classe CApp.cpp :

Dans la classe principale, je gère l'ensemble de la logique du système, notamment la réception des statuts des BOM. Ces statuts permettent à l'application de déterminer les actions à effectuer : quel ordre envoyer, quelles données mettre à jour dans la base, etc.
Voici la méthode chargée d'envoyer les ordres en fonction du statut reçu par la BOM :

```
void CApp::onInfoUpdated(QString type, QString ip, QString stockage, QString status, QString couleur,
QString progression, QString nbrCollision, QString ipTOM)
{
    if (_type == "BOM") {
        _connected = "1";
        switch (_status.toInt()) {
        case 1: //Information
            dbReader->insertDB("BOM", QVariantList{_progression, _couleur, _status, _connected,
_nbrCollision, _ip});
            break;
        case 2: //Remplissage (vérification TOM stockage)
            if (dbReader->checkTOM(_ipTOM)) {
                sendMsgTCP(_ipTOM, 3); //Remplissage en cours de BOM
            } else {
                sendMsgTCP(_ip, 5); //Remplissage en cours intérronpu
            }
            break;
        case 3: //Récupéré
            _stockage = "0";
            dbReader->insertDB("BOM", QVariantList{_progression, _couleur, _status, _connected,
_nbrCollision, _ip});
            dbReader->insertDB("TOM", QVariantList{_stockage, _couleur, _connected, _ipTOM});
            sendMsgTCP(_ipTOM, 4);
            break;
        case 4: //Stop Remplissage
            sendMsgTCP(_ipTOM, 2);
            break;
        case 5:  //Collision/choc
            dbReader->insertDB("BOM", QVariantList{_progression, _couleur, _status, _connected,
_nbrCollision, _ip});
            break;
        }
    }
    if (_type == "TOM") {
        int _connected = 1;
        dbReader->insertDB("TOM", QVariantList{_stockage, _couleur, QString::number(_connected), _ip});
    }
}
```

J'ai choisi d'utiliser une instruction **switch** pour faciliter le décodage des statuts et la prise de décision sur la démarche à suivre. Cette méthode constitue l'un des éléments clés du code, car elle assure le lien entre la partie visuelle (interface ou site web) et la partie applicative, moins visible mais essentielle au fonctionnement global.

Une autre méthode importante de cette classe concerne la transmission des informations de la partie aux différents appareils, en particulier aux BOM. Cela repose sur le **statut de la partie**, défini plus tôt dans le projet, qui permet à l'application et aux appareils connectés de savoir dans quel état se trouve le jeu (en cours, en pause, terminé, etc.).

```
void CApp::onConfigUpdated(QString etat, QString options, QString ptsRecolte, QString nbrTOM)
{
    static int _etatPrecedent = -1; // Stocke l'état précédent (-1 pour garantir un premier envoi)
    this->_etat = etat;  //0=stopped ; 1=running ; 2=paused ; 3=loading (countdown)
    this->_options = options;
    this->_ptsRecolte = ptsRecolte;
    this->_nbrTOM = nbrTOM;

    //Ordre à envoyé selon Etat?
    //1=Envoie demande info,
    //2=Initialiser, (loading = 3)
    //6=Pause, (stopped & paused = 0 & 2)
    //7=Continue (running = 1)
    int currentEtat = _etat.toInt();
    QStringList ipList = dbReader->getAllIPs(); // Récupère toutes les IPs

    if (currentEtat != _etatPrecedent) {
        for (const QString &ip : ipList) {
            switch (currentEtat) {
                case 0: //Envoie ordre 6 pour mise en pause
                case 2:
                    sendMsgTCP(ip, 6); // Envoie l'ordre 6 à chaque IP
                    qDebug() << "Ordre 6 envoyé à l'IP :" << ip;
                    break;
                case 1:  //Envoie ordre 7 pour continuer/commencer le jeu
                    sendMsgTCP(ip, 7); // Envoie l'ordre 7 à chaque IP
                    qDebug() << "Ordre 7 envoyé à l'IP :" << ip;
                    break;
                case 3: //Envoie Ordre 2 pour initialisation
                    sendMsgTCP(ip, 2, _nbrTOM); // Envoie l'ordre 2 à chaque IP
                    qDebug() << "Ordre 2 envoyé à l'IP :" << ip;
                    break;
            }
        }
        _etatPrecedent = currentEtat;
    }
}
```

J'ai globalement fait le tour de la partie applicative de mon projet. Ce travail m'a permis de découvrir le **protocole UDP** en début de développement, puis de comprendre pourquoi le **TCP** était plus adapté pour assurer la **fiabilité** des échanges. La mise en place des **statuts** et des **ordres** a également demandé une bonne réflexion pour structurer la **logique** du système et assurer une communication cohérente entre l'application, les appareils connectés et la base de données.

VI - Compétences Mobilisées

➢ Compétences Techniques :

- **Administration Linux & Réseaux :** configuration de la Raspberry Pi, hotspot WPA2, service DHCP
- **Sécurité :** chiffrement WPA2, validation des entrées PHP (injection SQL), mise en place d'un pare-feu
- **Conception de base de données :** besoin et usage de la base de données, diagramme MCD (papier), et modèle MLD (Workbench)
- **Développement Web :** PHP/Mysql pour le back-end, JavaScript (AJAX) pour le front-end
- **Programmation C++ :** implémentation des classes nécessaires pour l'écoute TCP (JSON) et le sondage de la base de données toutes les secondes

➢ Compétences Méthodologiques :

- **Gestion de projet :** planification via Microsoft Project, suivi des tâches prévues
- **Approche itérative et incrémentale :** découpage en trois itérations : réflexion, tests unitaires et d'intégration à chaque phase
- **Modélisation SysML :** formalisation des cas d'utilisation, diagrammes de séquence et de déploiement dans MagicDraw

➢ Compétences Transversales :

- **Résolution de problèmes :** identification rapide des bugs, mise en place de correction simple et stable.
- **Communication professionnelle :** rédaction de fiches de recette, documentation utilisateur, compte rendu de réunion, communication avec l'ensemble de l'équipe.
- **Autonomie et rigueur :** respect des délais et organisation de mon travail
- **Coordination d'équipe :** répartition des tâches entre IR, motivation des membres et suivi de leur travail régulier (IR)

VII - Bilan personnel

Durant la partie du projet actuellement réalisée, je me suis beaucoup amusé à concevoir et à améliorer continuellement l'interface Web. La découverte de nouvelles méthodes m'a permis d'apprendre énormément, tant sur le plan technique que sur l'organisation du travail. La réflexion nécessaire pour gérer les ordres et les statuts m'a particulièrement plu, car elle m'a permis de mieux comprendre la logique derrière ce type d'architecture.

J'ai également été satisfait de travailler avec l'équipe à laquelle j'ai été assigné. La répartition des tâches s'est faite de manière fluide et la communication au sein de l'équipe IR a été très bonne. Je tiens à remercier mes professeurs pour leur aide tout au long du projet. L'élaboration des documents a parfois été fastidieuse, mais loin d'être insurmontable : cela m'a permis de me familiariser avec des outils que je connaissais peu jusqu'à présent.

Les objectifs liés à l'affichage du site Web, ainsi que ceux concernant l'ensemble du jeu, ont été atteints conformément aux exigences du client, je l'espère. J'ai même pris l'initiative d'ajouter quelques options et idées supplémentaires, dans l'espoir qu'elles soient appréciées et validées. Ce projet a été une expérience enrichissante, tant sur le plan technique que personnel.

Partie Personnelle (Badr ERRAFI - ER 1) :

BOM

I - Introduction

Lors de ma deuxième année de BTS, j'ai choisi d'intégrer l'équipe du projet **ESTOM** (Espace de Sensibilisation au Tri des Ordures Ménagères).
Le but de ce projet est de créer un espace de sensibilisation ludique pour les enfants, basé sur le tri des déchets.

Ma partie du projet consiste à réaliser une carte électronique intégrée à la **Benne à Ordures Ménagères (BOM)**, capable de détecter des chocs et de communiquer avec la **Tour à Ordures Ménagères (TOM)** via un récepteur infrarouge.
Toutes les données générées seront transmises par Wi-Fi.

II - OBJECTIF

L'objectif de ma partie est de concevoir une carte électronique permettant de gérer plusieurs fonctions, notamment :

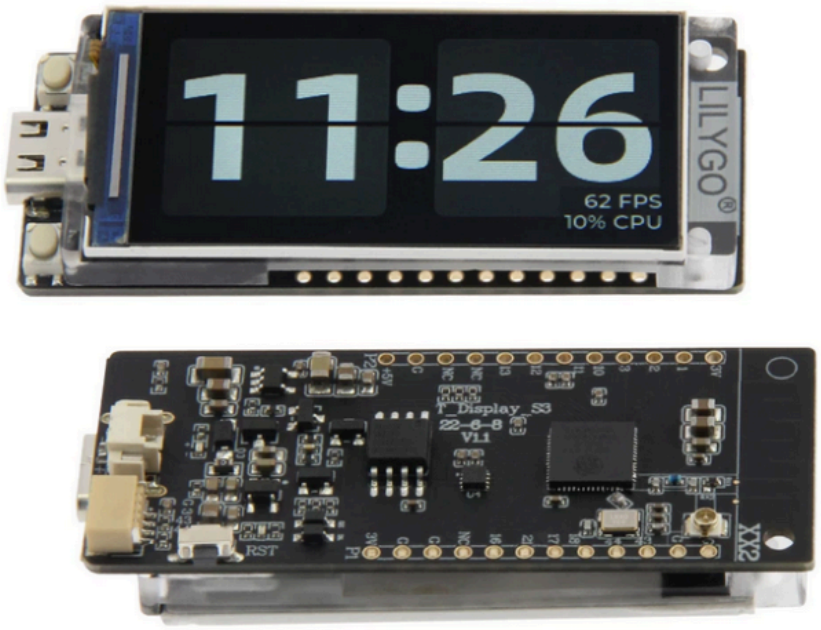
- Intégrer deux démodulateurs infrarouges, placés de chaque côté de la benne, afin de permettre la communication avec la TOM.
- Ajouter des boutons poussoirs permettant de compter le nombre de chocs subis par la benne.
- Mettre en place une communication avec la TOM.

III - PLANIFICATION

III.1 - MATÉRIEL UTILISÉ

Pour le matériel utilisé pour faire les tests pour l'instant j'ai utilisé,

Un ESP32-S3 T-Display LILYGO qui m'a permis de tester au début les différentes fonctions, mais comme nous n'avons pas besoin d'écran et il y a également un problème avec le protocole de communication qui est l'Infrarouge alors nous avons changé de micro contrôleur.



Nous sommes passé sur une autre version de l'esp32 qui est **ESP32-S3 Dev-Kit C1** qui ressemble à ceci:

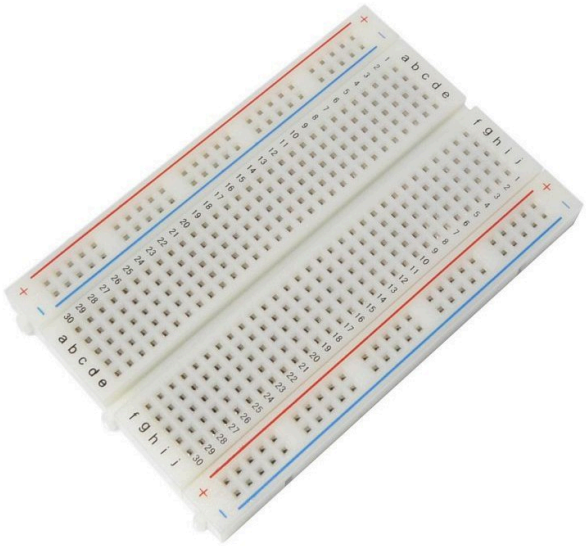


Après avoir fait plusieurs tests avec cet carte de développement on s'est rendu compte que cette carte prenait beaucoup de place en plus d'avoir beaucoup de broche qui ne servent a rien donc nous avons changés pour un modèle plus petit qui est le :**XIAO ESP32S3**



Ce modèle est plus petit et comporte moins de broche donc idéal pour notre cas.

Une plaque Labdec:

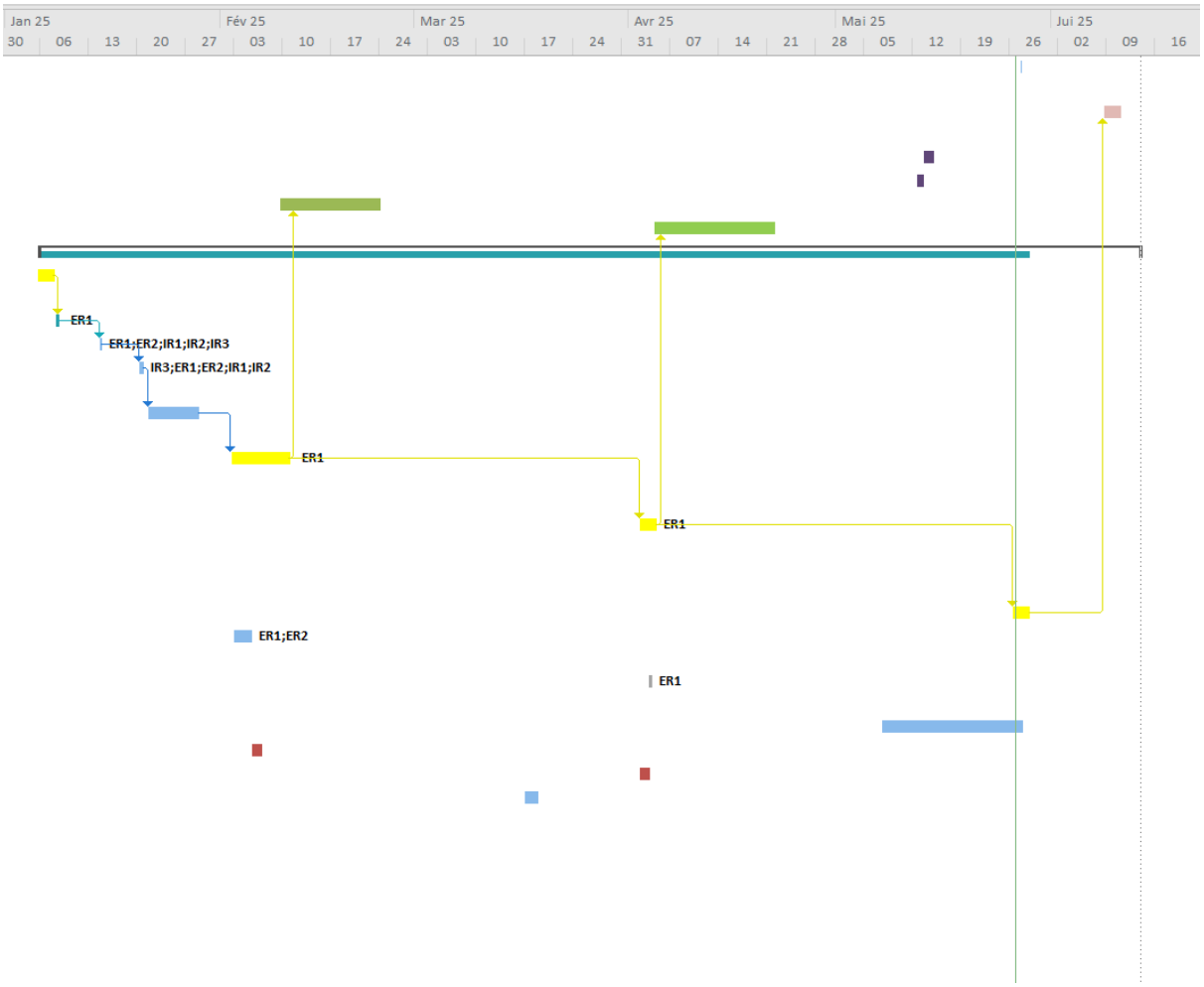


J’ai aussi utiliser des sites comme **Fritzing** ou alors **Tinkercad**

III.2 - Planification des Tâches

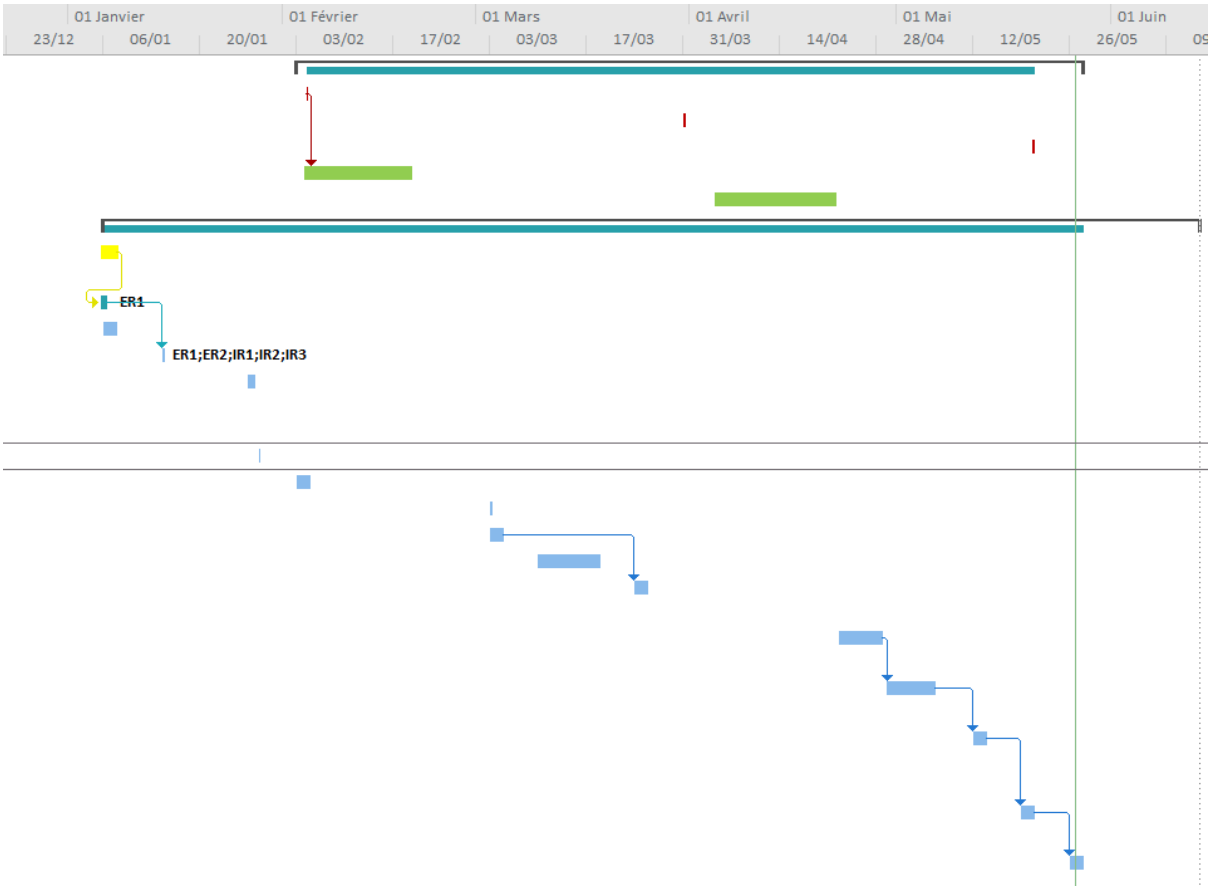
Ceci est une planification prévisionnelle

	Moc Tâcl	Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources
1		**Evaluations du projet**	**174 h**	**Lun 03/02/25**	**Mar 27/05/25**		
2		Revue 1	2 h	Lun 03/02/25	Lun 03/02/25		
3		Revue 2	2 h	Lun 31/03/25	Lun 31/03/25		
4		Dépôt dossier technique	3 h	Mar 27/05/25	Mar 27/05/25		
5		Soutenance finale E62 (Revue 3)	7 h	Lun 09/06/25	Mar 10/06/25	18	
6		**Epreuve E4**	9 h	Mer 14/05/25	Mer 14/05/25		
7		**Epreuve E1**	4 h	Mar 13/05/25	Mar 13/05/25		
8		Vacances d'hiver	3,43 jrs	Lun 10/02/25	Dim 23/02/25	16	
9		Vacances printemps	5,43 jrs	Sam 05/04/25	Lun 21/04/25	17	
10		**Projet**	**241 h?**	**Lun 06/01/25**	**Ven 13/06/25**		
11		Specifications générales	7 h	Lun 06/01/25	Mar 07/01/25		
12		decouverte du projet	2 h	Mer 08/01/25	Mer 08/01/25	11	ER1
13		reunion de projet	2 h	Mar 14/01/25	Mar 14/01/25	12	ER1;ER2;IR1;IR2;IR3
14		definir le protocole de communication	2 h	Lun 20/01/25	Lun 20/01/25	13	IR3;ER1;ER2;IR1; IR2
15		se documenter sur les LED et faire des essais	14 h	Mar 21/01/25	Mar 28/01/25	14	
16		Essais et validation des structures (prototypage rapide)	19 h	Lun 03/02/25	Lun 10/02/25	15	ER1
17		Prototypage rapide et routage en fonction des dimensions du camion	10 h	Mer 02/04/25	Ven 04/04/25	16	ER1
18		Fabrication et essais	8 h	Lun 26/05/25	Mer 28/05/25	17	
19		tester le capteur avec l'Arduino	8 h	Lun 03/02/25	Mer 05/02/25		ER1;ER2
20		Fichier de fabrication PCB finalisés	3 h	Ven 04/04/25	Ven 04/04/25		ER1
21		Rédaction du dossier	30 h	Mer 07/05/25	Mar 27/05/25		
22		1er BTS Blanc	9 h	Jeu 06/02/25	Jeu 06/02/25		
23		2nd BTS Blanc	9 h	Jeu 03/04/25	Jeu 03/04/25		
24		Soutenance E61	1 jr?	Lun 17/03/25	Mar 18/03/25		


Jan 25
Fév 25
Mar 25
Avr 25
Mai 25
Jui 25
30 06 13 20 27 03 10 17 24 03 10 17 24 31 07 14 21 28 05 12 19 26 02 09 16
ER1
ER1;ER2;IR1;IR2;IR3
IR3;ER1;ER2;IR1;IR2
ER1
ER1
ER1;ER2
ER1

Ceci est une planification réelle

	Moc Tâch	Nom de la tâche	Durée	Début	Fin	Prédécesseurs	Noms ressources
1		**Evaluations du projet**	**174 h?**	**Lun 03/02/25**	**Mar 27/05/25**		
2		Revue 1	1 h	Mar 04/02/25	Mar 04/02/25		
3		Revue 2	1 h	Lun 31/03/25	Lun 31/03/25		
4		Revue 3	1 h	Mar 20/05/25	Mar 20/05/25		
5		Vacances d'hiver	3,43 jrs	Mar 04/02/25	Mer 19/02/25	2	
6		Vacances printemps	5,43 jrs	Sam 05/04/25	Lun 21/04/25		
7		**Projet**	**241 h?**	**Lun 06/01/25**	**Ven 13/06/25**		
8		Specifications générales	7 h	Lun 06/01/25	Mar 07/01/25		
9		decouverte du projet	3 h	Lun 06/01/25	Lun 06/01/25	8	ER1
10		Gant previsionnelle	7 h	Lun 06/01/25	Mar 07/01/25		
11		reunion de projet	1 h	Mar 14/01/25	Mar 14/01/25	9	ER1;ER2;IR1;IR2;IR3
12		se documenter sur la bargraphe et faire des essais	3 h	Lun 27/01/25	Lun 27/01/25		
13		essais des BP	3 h	Mar 28/01/25	Mar 28/01/25		
14		TP Infra-rouge	7 h	Lun 03/02/25	Mar 04/02/25		
15		association des codes	2 h	Lun 03/03/25	Lun 03/03/25		
16		essaie ATTiny	7 h	Lun 03/03/25	Mar 04/03/25		
17		shéma structurelle	17 h	Lun 10/03/25	Mar 18/03/25		
18		fin des essaies de l'ATTiny	7 h	Lun 24/03/25	Mar 25/03/25	16	
19		cablage de la carte finale	8 h	Mar 22/04/25	Lun 28/04/25		
20		mise en service de la carte	12 h	Mar 29/04/25	Mar 06/05/25	19	
21		essaie de la communication avec la TOM	7 h	Lun 12/05/25	Mar 13/05/25	20	
22		préparation du dossier pour revue 3	7 h	Lun 19/05/25	Mar 20/05/25	21	
23		preparation du dossier finale	7 h	Lun 26/05/25	Mar 27/05/25	22	



IV - RESULTATS ATTENDUS

Dans le cadre de ce projet, les travaux attendus de ma part sont les suivants :

- Gérer un bargraphe à LED programmable, permettant d'indiquer la quantité d'ordures récupérées par le camion.
- Gérer un interrupteur de fin de course, afin de compter les chocs subis par le camion et ainsi appliquer d'éventuelles pénalités.
- Gérer un récepteur infrarouge pour recevoir les données envoyées par la TOM.
- Utiliser un microcontrôleur ATTiny85 pour démoduler les données reçues par le récepteur infrarouge.
- Piloter l'ensemble de ces composants à l'aide d'un ESP32.

V - TRAVAUX EFFECTUÉS

V.1 - ESSAIE EFFECTUÉS

Dans un premiers temp j'ai manipulé le bargraphe à Led , J'ai cherché un moyen de faire fonctionner ce composant et je suis tombé sur une librairie qui permet de mettre en fonctionnement voilà la librairie trouver :

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h>
#endif
#define PIN        43
#define NUMPIXELS 16
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500
void setup() {
 #if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif

  pixels.begin();
}

void loop() {
  pixels.clear();
  for(int i=0; i<NUMPIXELS; i++) {
    pixels.setPixelColor(i, pixels.Color(0, 150, 0));

    pixels.show();
    delay(DELAYVAL);
  }
}
```

La ligne de code "#define PIN 43" permet de définir la broche du microcontrôleur
La ligne de code " pixels.setPixelColor(i, pixels.Color(0, 150, 0)); " permet de définir la couleur du bargraphe
Ce composant je l'ai testé sur arduino puis sur ESP32

V.2 - DEUXIÈME ESSAI EFFECTUÉ

Par la suite, j'ai testé l'interrupteur de fin de course. J'ai créé un code permettant de compter le nombre de fois où l'interrupteur est actionné.
Cependant, je me suis rapidement rendu compte qu'un seul appui pouvait parfois être comptabilisé plusieurs fois.
Après quelques recherches, j'ai découvert qu'il s'agissait d'un phénomène appelé **rebond**.
Pour résoudre ce problème, je me suis aidé d'une vidéo explicative qui présentait un code permettant d'éliminer les rebonds.

Voici le premier code que j'ai utilisé, qui permet simplement d'allumer la LED lorsqu'on appuie sur le bouton.

```
const int ledPin = 7;
const int buttonPin = 2;
int buttonState = 0;
int lastButtonState = 0;
int pressCount = 0;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW && lastButtonState == HIGH) {
    delay(50);
    pressCount++;
    if (pressCount % 2 == 1) {
      digitalWrite(ledPin, HIGH);
    } else {
      digitalWrite(ledPin, LOW);
    }
    Serial.print("Nombre de pressions : ");
    Serial.println(pressCount);
  }
  lastButtonState = buttonState;
  delay(10);
}
```

résultats de ce code:

```
Nombre de pressions : 15
Nombre de pressions : 16
Nombre de pressions : 17
Nombre de pressions : 18
Nombre de pressions : 19
Nombre de pressions : 20
Nombre de pressions : 21
Nombre de pressions : 22
Nombre de pressions : 23
Nombre de pressions : 24
Nombre de pressions : 25
Nombre de pressions : 26
Nombre de pressions : 27
Nombre de pressions : 28
Nombre de pressions : 29
Nombre de pressions : 30
Nombre de pressions : 31
Nombre de pressions : 32
```

Lors de la pressions numéro 18 le bouton en a compté 3

Code avec un anti rebond:

```
#define PIN_BP 2
#define PIN_LED_ROUGE 3
#define DELAI_SANS_REBOND 10000

unsigned long compteur = 0; // Variable pour le compteur

void setup() {
  pinMode(PIN_LED_ROUGE, OUTPUT);
  pinMode(PIN_BP, INPUT_PULLUP);
  Serial.begin(9600); // Initialise la communication série pour afficher le compteur
}

void loop() {
  static uint8_t dernierBP = HIGH;
  uint8_t BP = digitalRead(PIN_BP);
  static unsigned long derniereTransition = 0;
  unsigned long _micros;

  if (dernierBP != BP) {
    _micros = micros();// NeoPixel Ring simple sketch (c) 2013 Shae Erisson
```

resultat du code :

```
Compteur: 1
Compteur: 2
Compteur: 3
Compteur: 4
Compteur: 5
Compteur: 6
Compteur: 7
Compteur: 8
Compteur: 9
Compteur: 10
Compteur: 11
```

Ici je n'ai pas de rebond lorsque j'appuie sur le bouton une fois sa compte qu'une seule fois.

V.3 - TROISIÈME ESSAIE EFFECTUÉS

Pour la suite j'ai mis en oeuvre un ATTiny85 pour le faire fonctionner l'ATTiny doit être programmé au préalable pour le programmer j'ai utilisé un **Programmateur SparkFun**



J'ai programmé l'ATTiny de sorte qu'il puisse démoduler les données que reçoit le modulateur IR (Infra Rouge)

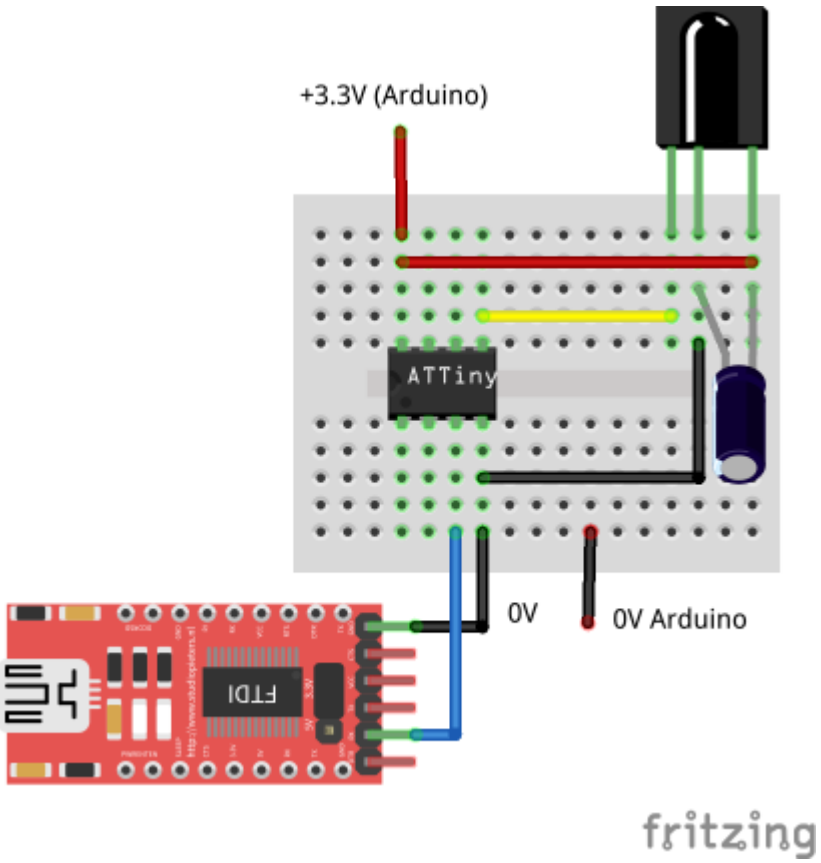
Voila le code :

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3,4);  //rx, tx
// Specify protocol(s) - Must be listed before IRremote.hpp
#define DECODE_NEC
// Include required libraries
#include <Arduino.h>
#include <IRremote.hpp>
// IR Receiver pin
#define IR_RECEIVE_PIN 0
// LED Pin
//#define LED_PIN 5
// Data received variable
volatile bool irDataReceived = false;
// variables
volatile int touche = 0;
uint16_t flecheHaut = 0x718;
uint16_t flecheBas = 0x719;
// Receive Callback
void ReceiveCallbackHandler() {
  // Decode IR data
  IrReceiver.decode();
touche= IrReceiver.decodedIRData.command;
 /* if (IrReceiver.decodedIRData.command == flecheHaut) {
  touche = 10;
  }
   else if (IrReceiver.decodedIRData.command == flecheBas) {
    touche = 11;
  }
  else
  {
    touche = 0;
  }*
  // Set data received flag true
  irDataReceived = true;

  // Resume receiving
  IrReceiver.resume();
}
void setup() {
  pinMode(5,INPUT_PULLUP); // PB5= RESET avec PullUp
   mySerial.begin(9600);
   mySerial.println("Reception touche IR NEC");
  // Start IR Receiver with LED feedback on BUILTIN LED
  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);
  // Attach Callback Handler
  IrReceiver.registerReceiveCompleteCallback(ReceiveCallbackHandler);
}
void loop() {
  // See if new data is available
  if (irDataReceived) {
  mySerial.print("Touche : ");
  mySerial.println(touche);
```

```
    delay(1000);
    // Set data received flag false
    irDataReceived = false;
  }
}
```

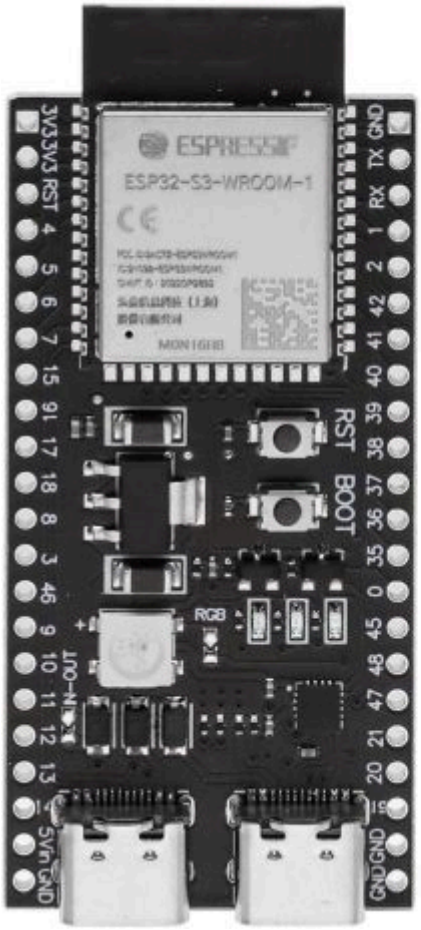
Après avoir programmé l'ATTiny j'ai du cablé le démodulateur IR qui est un **TSOP38238,** le 38 signifie la fréquence de la porteuse
voila le câblage utiliser :



Pour visualiser les données infrarouge j'ai utilisé un appareil qui permet de visualiser la communication UART, j'ai visualisé les données infrarouge avec l'appli UTF-8 TeraTerm Pro
Les données infrarouges ont été envoyés par une télécommande programmé avec le protocole NEC

Après avoir fais ceci j'ai décidé de faire la même chose mais sur l'ESP32-S3 T-Display LILYGO, mais j'ai eu un problème je n'arrivais pas à faire visualiser ce qu'envoyait l'ATTiny a l'ESP
Donc pour resoudre ce probleme j'ai changer l'ESP32-S3 T-Display par un **ESP32-S3 Dev-Kit C1** qui lui a une broche RX , TX qui est déjà dédié au microcontrôleur

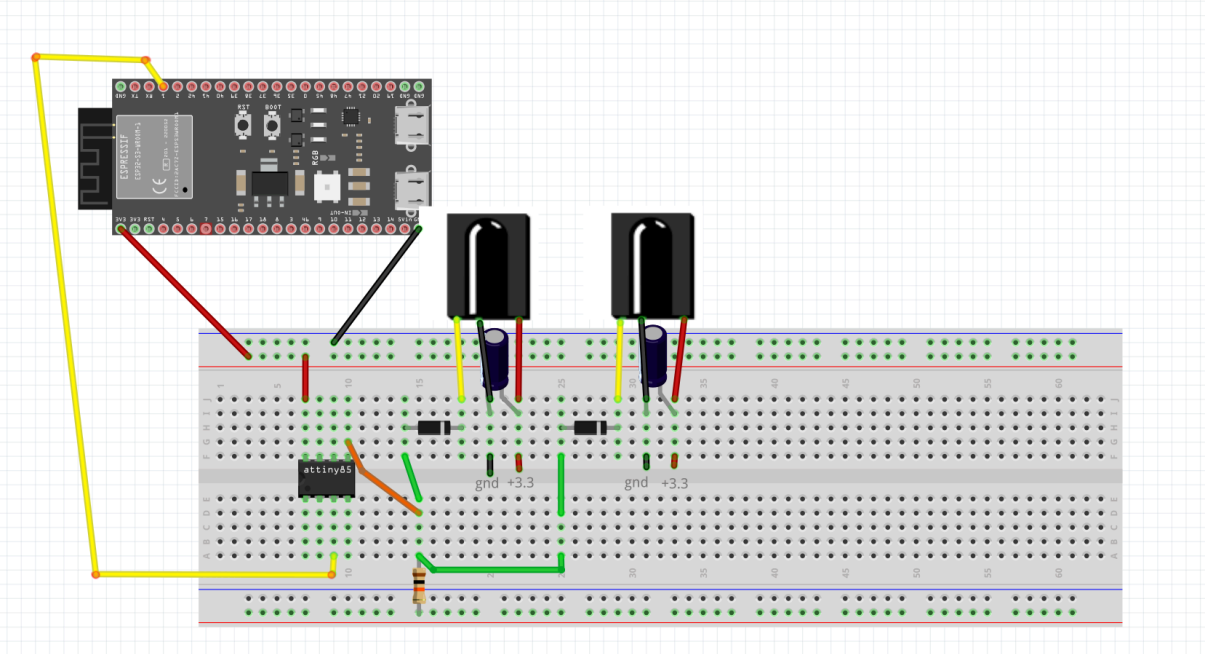
Front



Back



Le test suivant a été de mettre en œuvre deux démodulateurs InfraRouge car dans le camion il faut deux démodulateurs un de chaque côté pour cela j'ai câbler un ET logique avec deux diodes et une résistance de 10K Ohms . Le ET logique câblée permet d'avoir un seul ATTiny pour deux démodulateur



Ce câblage a été effectué sur Fritzing
câblage sur plaque labdec:

Dans l’ESP32 j’ai mis ce code :

```
char incomingByte = 0; // for incoming serial data
int valInt;

    void setup() {

      Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

    }


    void loop() {

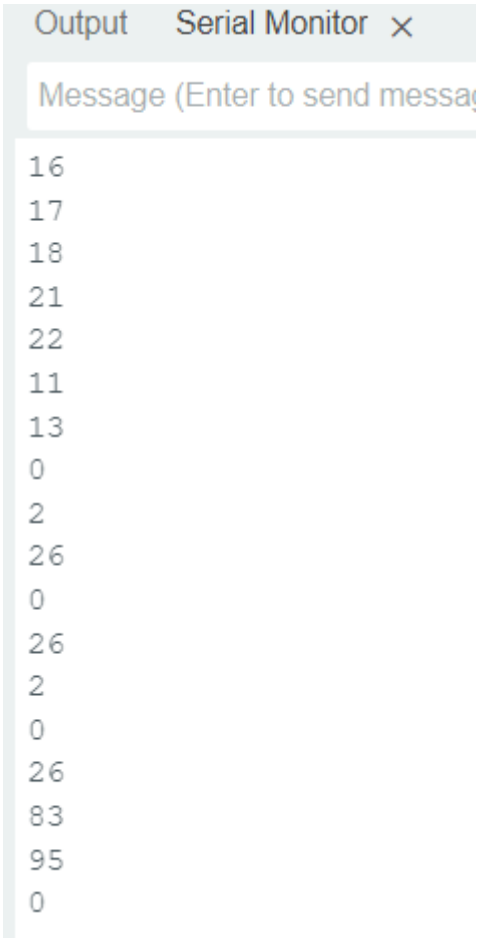
      // reply only when you receive data:

      if (Serial.available() > 0) {
        //read the incoming byte:
        incomingByte = Serial.read();
        valInt= incomingByte;
        // say what you got:
       // Serial.print("I received: ");
      //Serial.println(incomingByte);
      Serial.println(valInt);
      }

    }
```

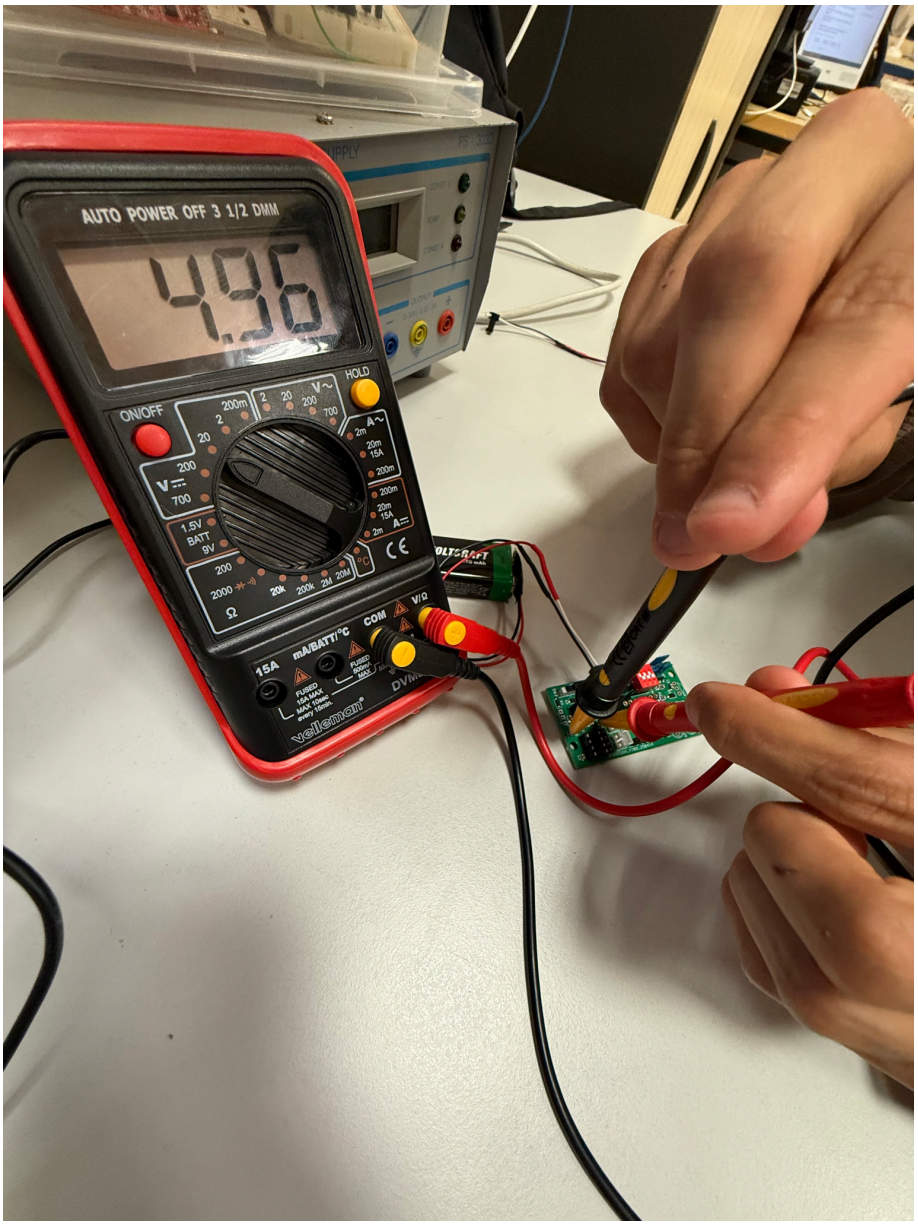
Et dans l'ATTiny je ne l’ai pas changer

Voilà ce qu’on obtient, a noter qu' avec différente télécommande on peut avoir un décalage ici j’ai eu un décalage de 16 donc quand j’appuie sur la touche 1 sa m’affiche 16.


Output
Serial Monitor
Message (Enter to send messa
16
17
18
21
22
11
13
0
2
26
0
26
2
0
26
83
95
0

V.4 - AUTRE ESSAI

Pour la suite des essais j'ai testé le régulateur de tension, le but du régulateur de tension dans notre cas est d'abaisser la tension d'entrée de la batterie pour alimenter la carte en +5V.

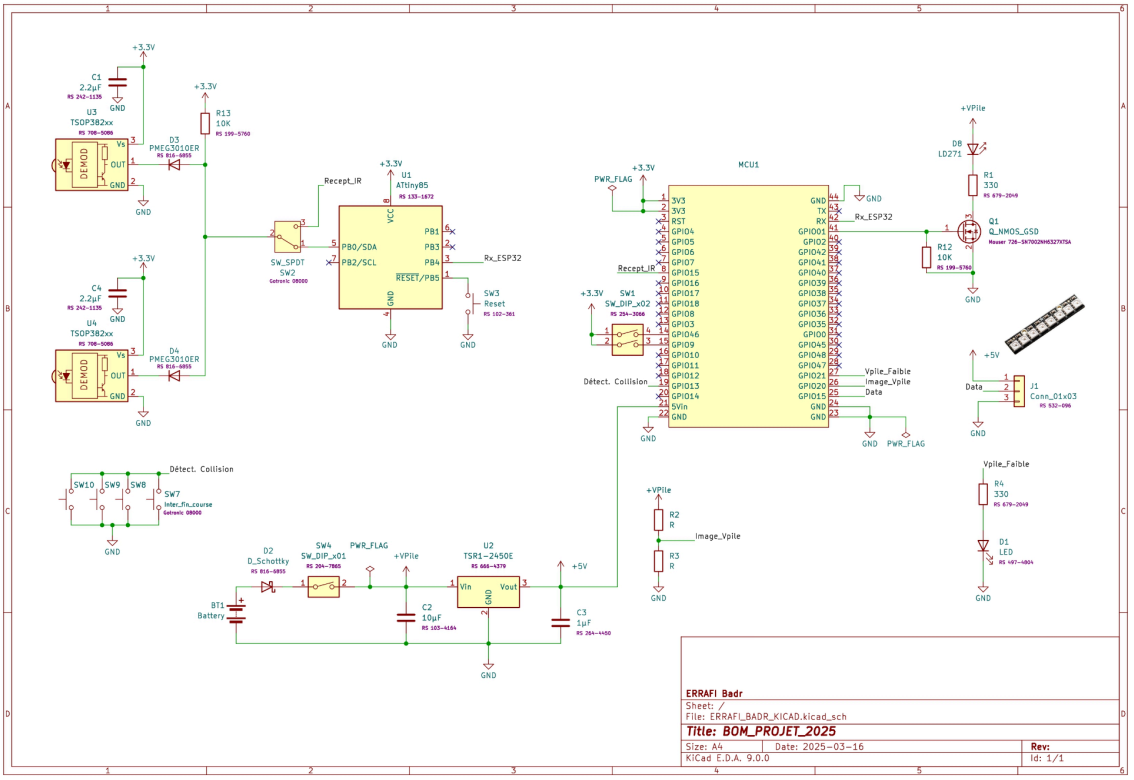


Pour effectuer ce test, j'ai utilisé un multimètre réglé en voltmètre, une batterie de 12V, ainsi qu'une carte équipée d'un régulateur de tension.
En entrée du régulateur, nous avons appliqué la tension de la batterie, et en sortie, nous avons mesuré 4,95 V. On peut donc en conclure que le régulateur fonctionne correctement.

Par la suite, nous avons alimenté le régulateur à l'aide d'une alimentation extérieure, dans le but de déterminer à partir de quelle tension d'entrée le régulateur cessait de fournir la tension attendue.
Nous avons constaté qu'à partir de 5,4 V en entrée, le régulateur ne fonctionnait plus correctement.

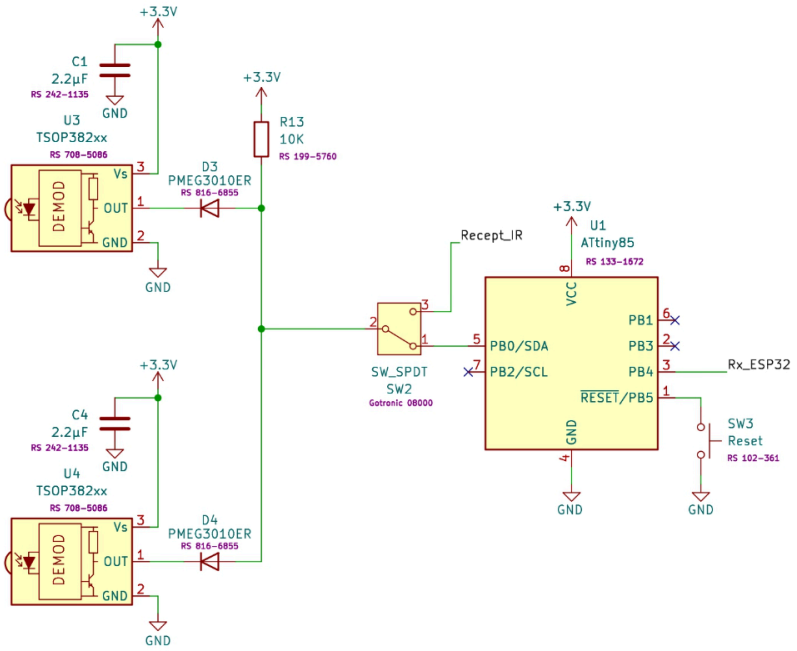
VI - ANALYSE

Je vais maintenant analyser le document structurel de la carte

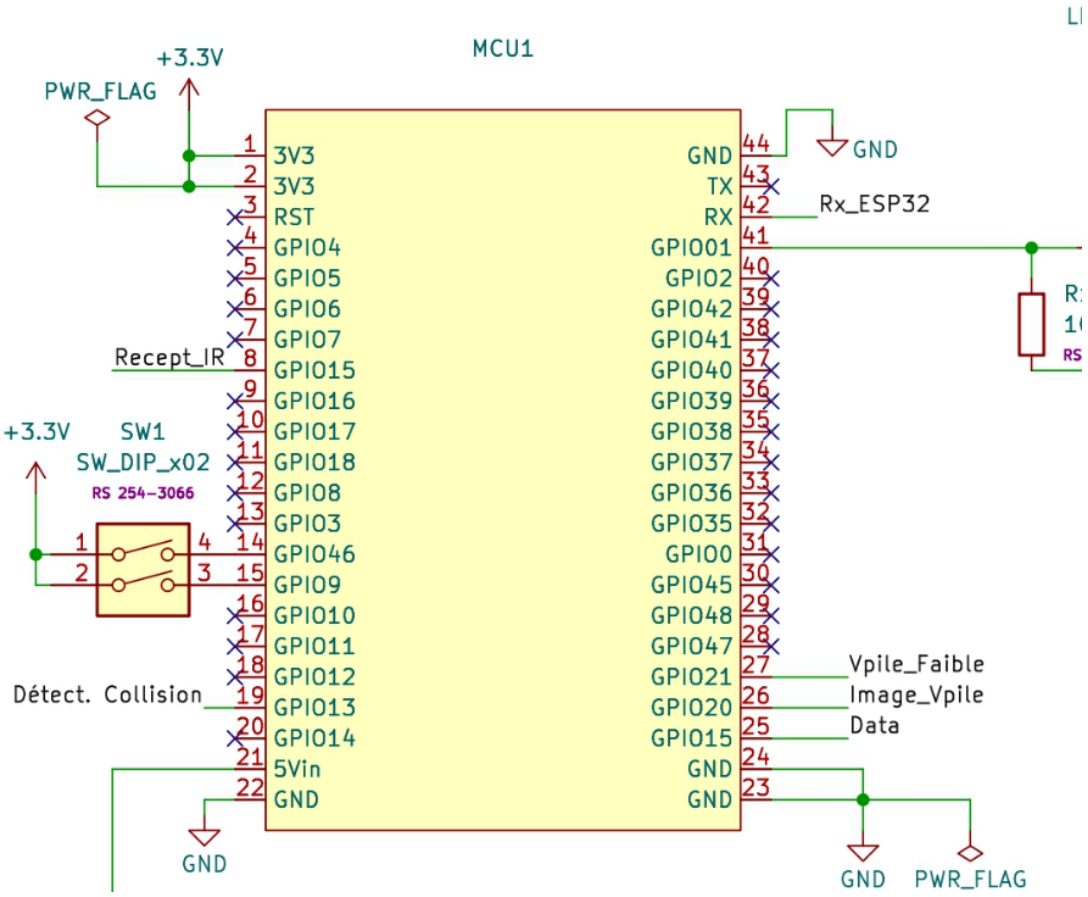


On observe ici la porte **ET logique** câblée avec les deux démodulateurs infrarouges, deux diodes, ainsi qu'une résistance de **pull-up**.
Le circuit comprend également l'**ATTiny85**, avec la possibilité de choisir entre l'utilisation de l'ATTiny ou de l'ESP32 à l'aide de l'interrupteur **SW2**.

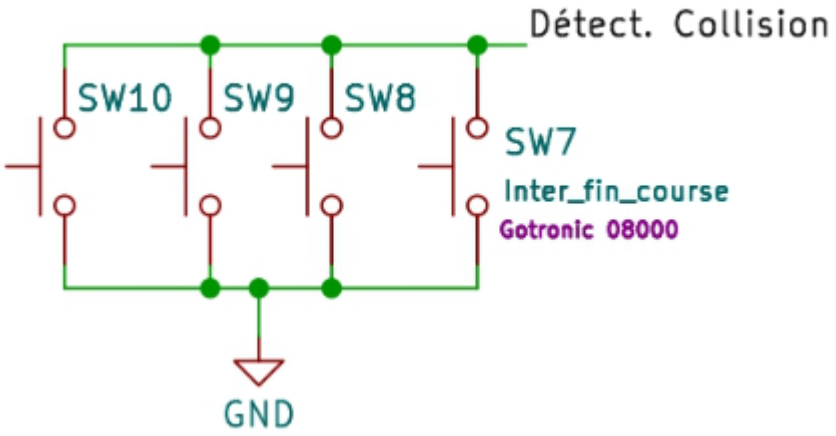
L'objectif de l'ATTiny est de **délester** l'ESP32 de certaines tâches, afin que celui-ci n'ait pas à gérer trop de traitements simultanément.



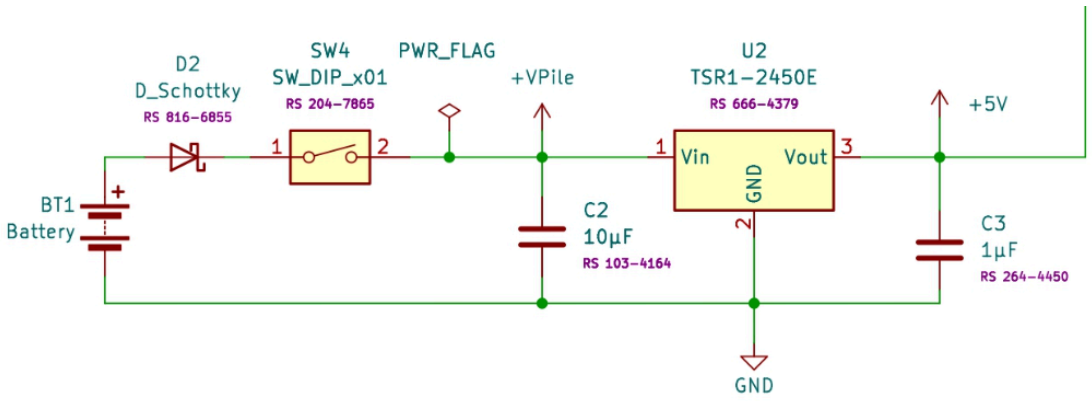
On voit ici l'ESP32 et un DIP Switchs qui permet de pouvoir changer la couleur de la benne sans a avoir a changer tout le code à l'intérieur du microcontrôleur.



on voit ici qu'on a les interrupteurs de fin de courses qui vont permettre de détecter les chocs

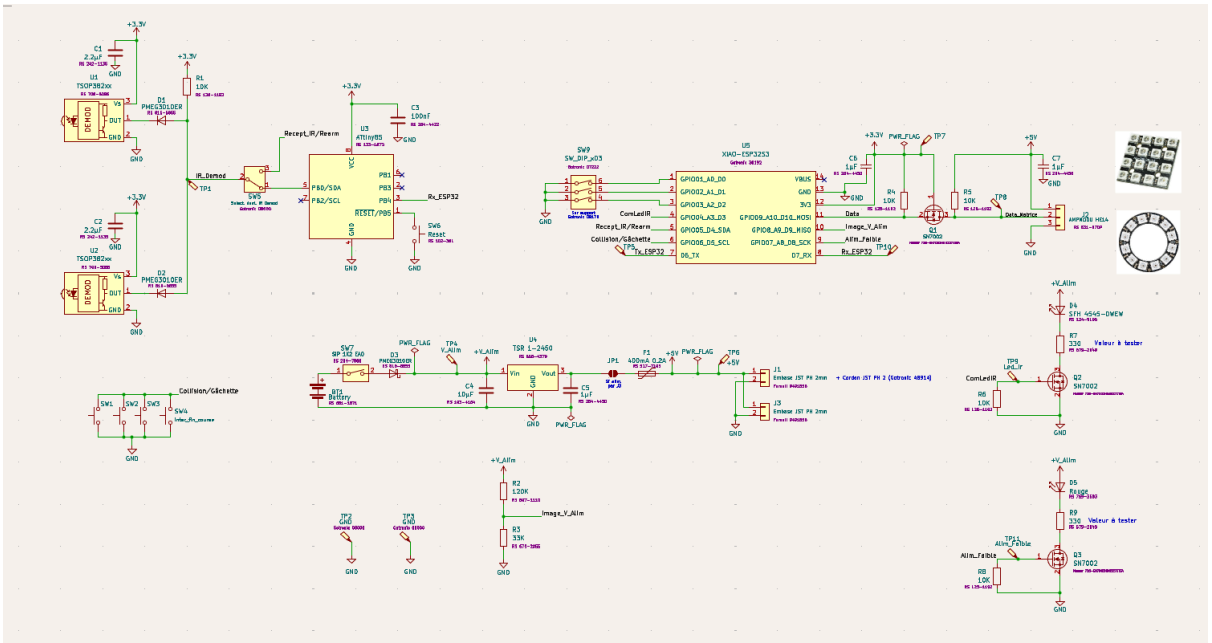


Ici on a une batterie, un interrupteur qui permet d'utiliser la batterie ou pas et un régulateur de tension qui permet d'alimenter en +5V en sortie.

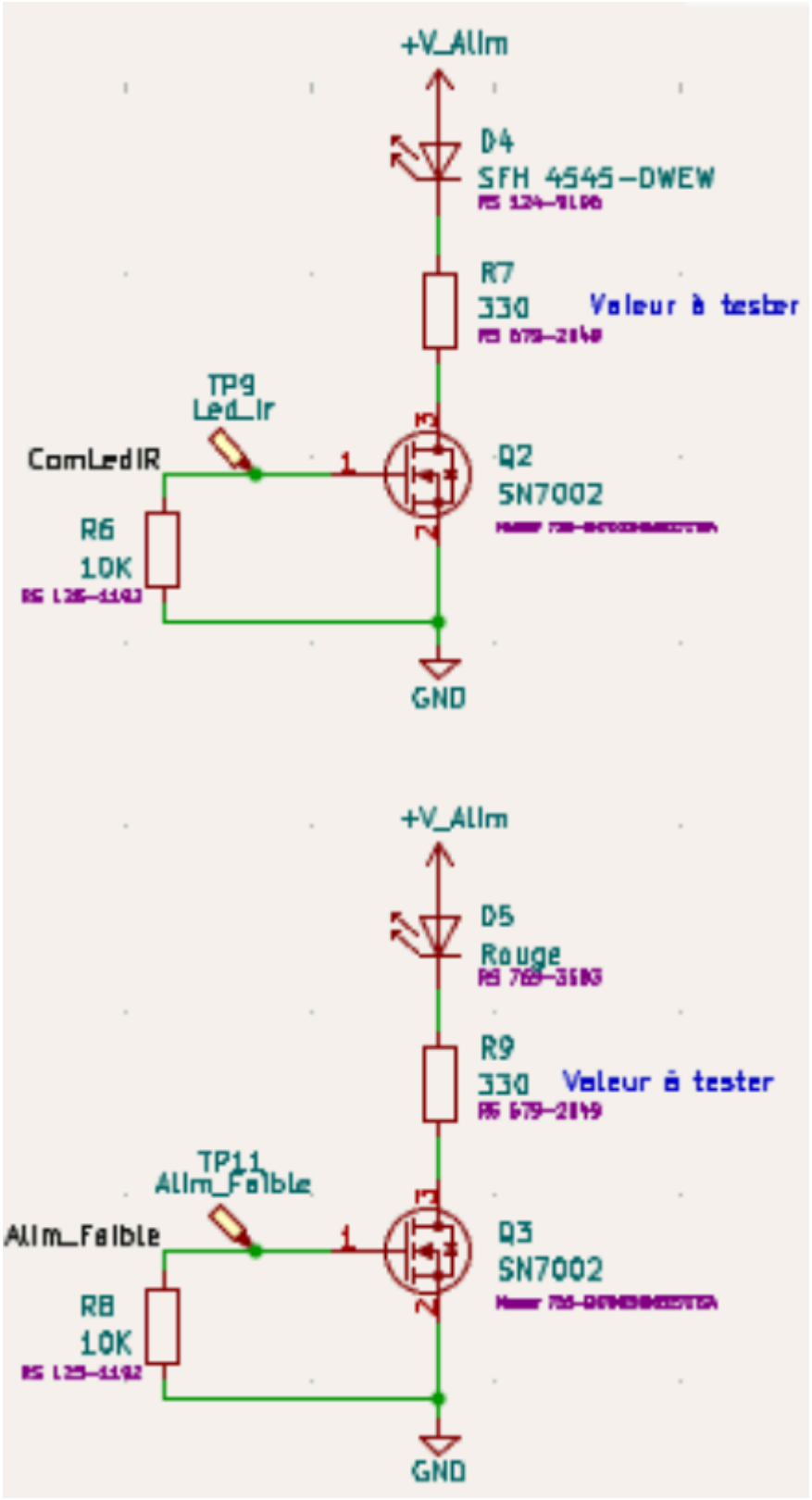


VI.1 - AUTRE ANALYSE

Après avoir changé la taille du microcontrôleur, nous avons décidé de créer une carte qui réunit les deux parties du projet ESTOM donc la TOM et la BOM la carte aura les mêmes fonctions que la précédente.



Sur cette version du schéma on a une partie en plus que sur l'ancienne cette partie est l'émission InfraRouge qui fait partie de la TOM


+V_Allm
D4
SFH 4545-DWEW
R7
330
Valeur à tester
TP9
Led_Ir
ComLedIR
Q2
SN7002
R6
10K
GND
+V_Allm
D5
Rouge
R9
330
Valeur à tester
TP11
Allm_Faible
Allm_Faible
Q3
SN7002
R8
10K
GND

VII - FABRICATION

VII.1 - NOMENCLATURE

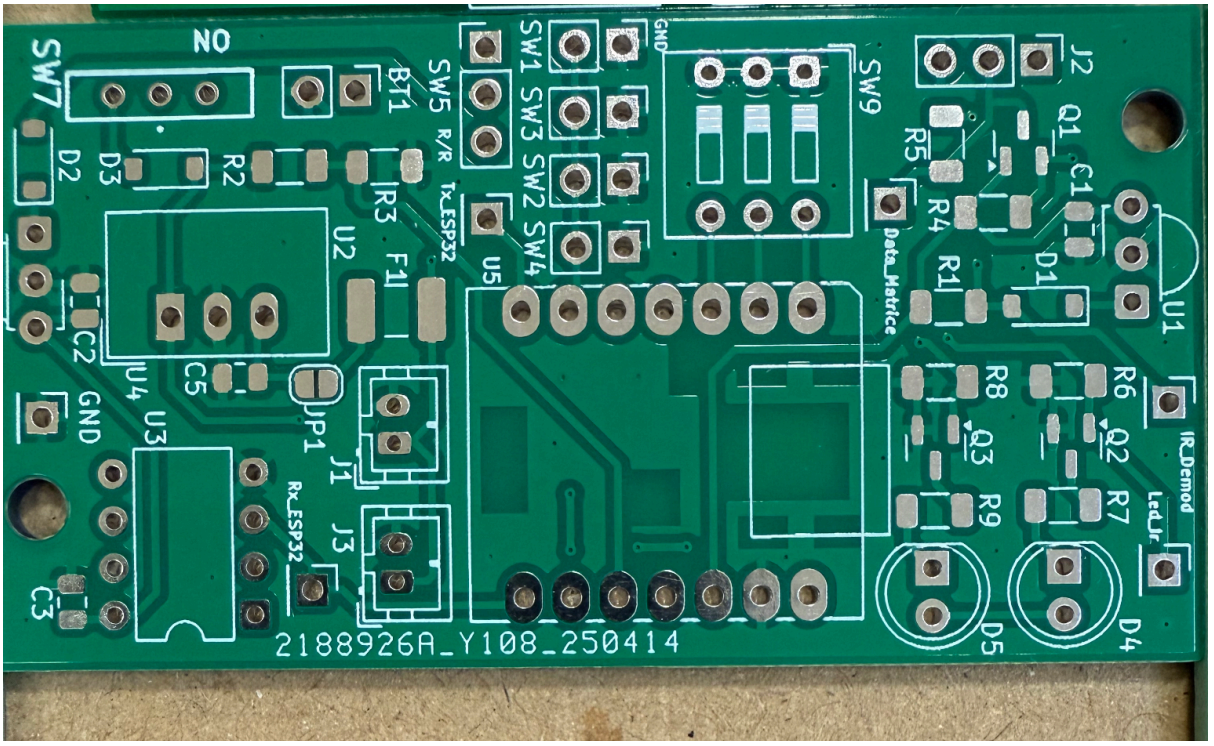
Repère	nation du m	Valeur (Référence)	Tol.±	Equivalent	Fournisseur	Boitier	Remarques	Qté	Qté Stock	Condition	ref rapide	Prix U HT	Prix T HT
BT1					Radiospares						681-1871		
C1,C2	Condensateur CMS MLCC 16V	2,2 µF			Radiospares	CMS 0805		2		Par 10	242-1135	0,198	0,396
C3	Condensateur CMS MLCC 50V	100nF	±20%		Radiospares	CMS 0805		1		Par 25	264-4422	0,104	0,104
C4	Condensateur CMS MLCC 35V	10 µF	±10%		Radiospares	CMS 1206		1		Par 20	103-4164	0,235	0,235
C5,C6,C7	Condensateur CMS MLCC 16V	1 µF			Radiospares	CMS 0805		3		Par 25	264-4450	0,042	0,126
D1,D2,D3	Diode Schottky CMS	1 A			Radiospares	SOD-123		3		Par 50	816-6855	0,145	0,435
D4	Led infrarouge traversant				Radiospares	5 mm		1		Par 20	124-9106	0,304	0,304
D5	LED rouge traversant				Radiospares	5 mm		1		Par 10	769-3583	0,456	0,456
F1	Fusible réarmable CMS	400 mA			Radiospares			1		Par 10	517-7105	0,369	0,369
J1,J3					Farnell			2			9491856		
J2	Embases de circuit imprimé traversant				Radiospares			1		Par 200	531-970P	0,623	0,623
Q1,Q2,Q3	Transistors MOSFET CMS				Mouser	SOT-23-3		3		Par 1	726-SN7002NH6327XTSA	0,228	0,684
R1,R4,R5,R6,R8	Résistances CMS	10 kΩ	±1%		Radiospares	CMS 1206		5		Par 100	125-1192	0,023	0,115
R2	Résistances CMS	120kΩ	±5%		Radiospares	CMS 1206		1		Par 100	807-1110	0,029	0,115
R3	Résistances CMS	33kΩ	±1%		Radiospares	CMS 1206		1		Par 50	679-2055	0,026	0,026
R7 ,R9	Résistances CMS	330Ω	±1%		Radiospares	CMS 1206		2		Par 50	679-2049	0,026	0,026
	Connecteurs (point de test)				gotronic			17		Par 1	8000	0,600	0,600
SW6	boutton poussoir CMS				Radiospares			1		Par 10	102-361	0,631	0,631
SW7	Commutateur traversant				Radiospares			1		Par 50	204-7865	1,524	1,524
U1,U2	Recepteur IR traversant				Radiospares			2		Par 50	708-5086	0,624	1,248
U3	ATtiny 85				Radiospares	133-1672		1		Par 50	133-1672	1,572	1,572
U4	Régulateur de tension				Radiospares	SIP 3		1		Par 1	666-4379	5,620	5,620
U5	ESP 32				gotronic			1		Par 1	38192	8,600	8,600

	Nom	Date	Folio
0	ERRAFI	29/04/2025	1/1
0		29/04/2025	

TOTAL HT FEUILLE		23,783
TOTAL HT GLOBAL		23,783
COEFFICIENT TVA	1,2	28,540

VII.2 - CÂBLAGE

Après avoir passé la commande et reçu la carte, le stencil et les composants.

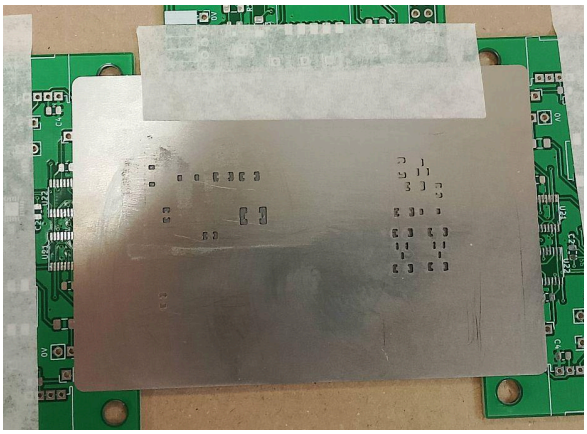


Nous avons désormais une procédure à suivre afin de câbler la carte dans de bonnes conditions.

Tout d'abord, il est nécessaire de **nettoyer la carte** avec de l'**alcool isopropylique**. Ensuite, les composants doivent être soudés **du plus petit au plus grand**, en commençant par les **composants CMS** (composants montés en surface).

Pour cela, on positionne le **stencil** (pochoir) de manière à ce qu'il ne bouge pas. Il est important de noter que le stencil doit également être **nettoyé** avant utilisation.

On applique ensuite de la **pâte à braser** à l'aide d'une **spatule**, en l'étalant uniquement sur les zones où les composants doivent être placés.

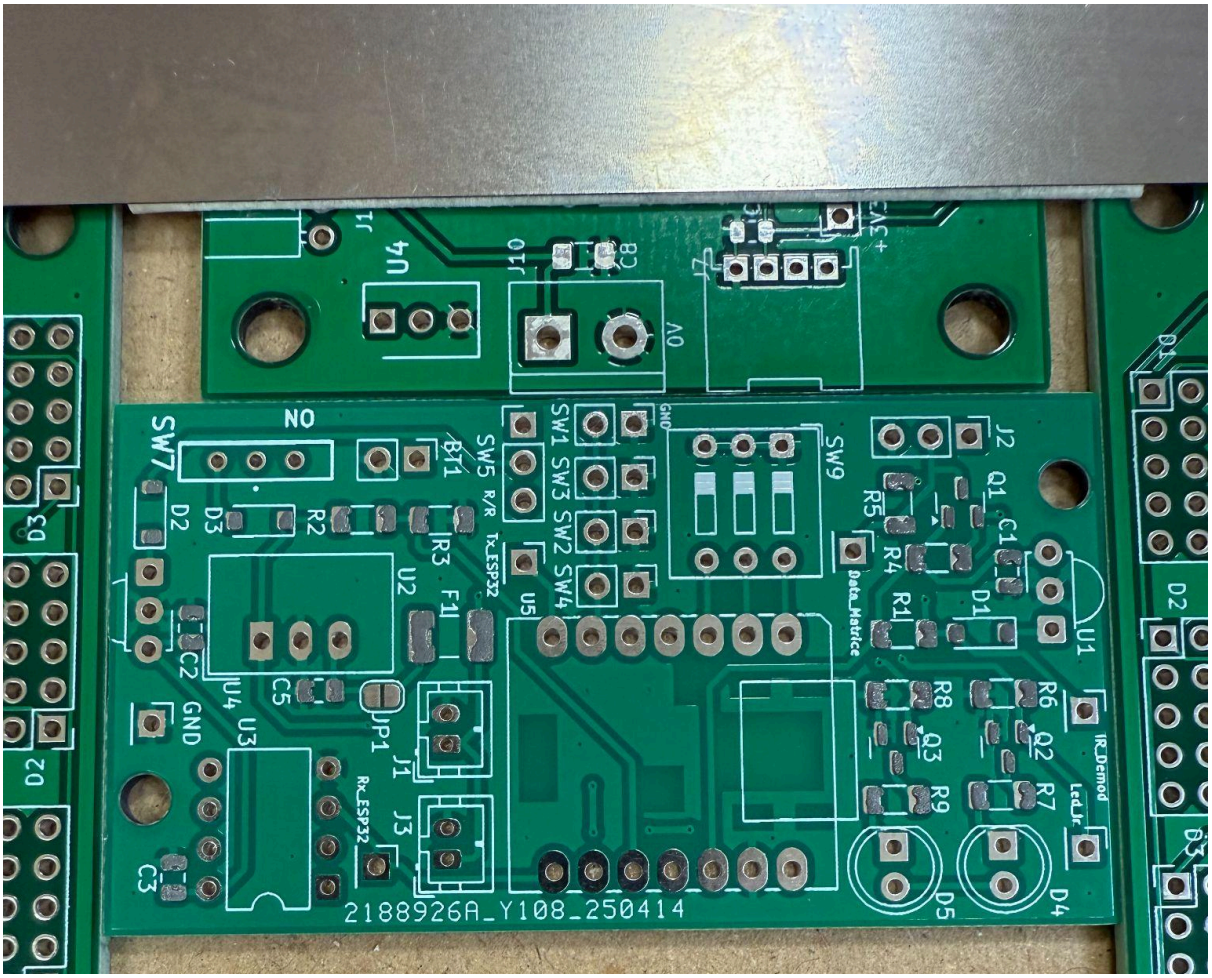


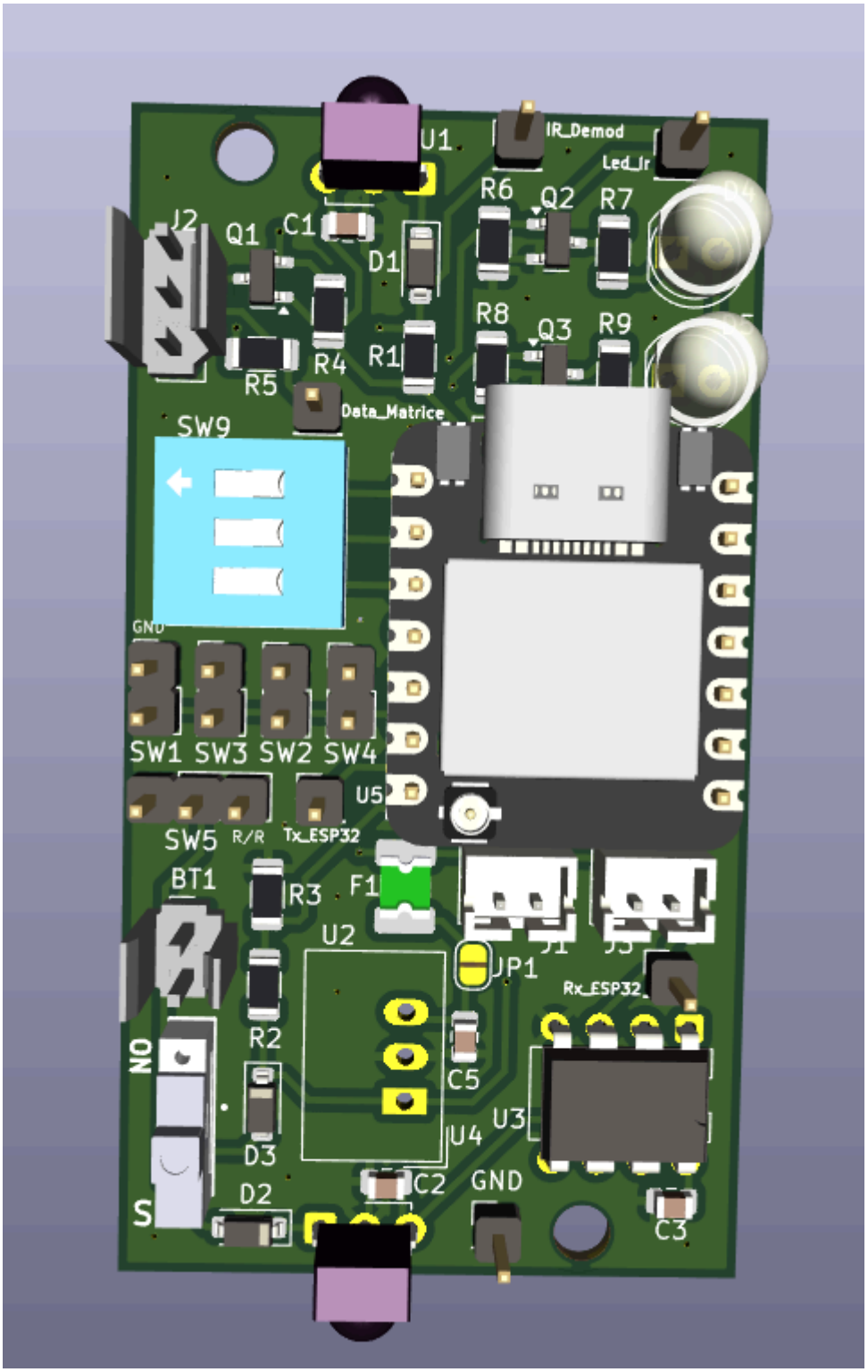


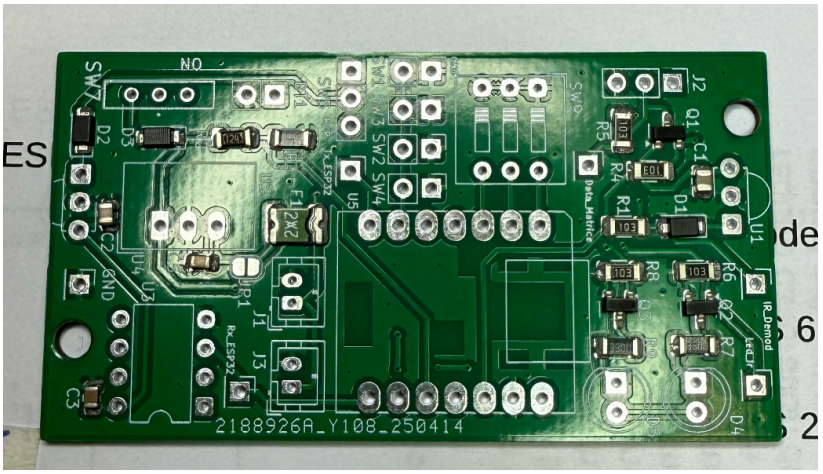
Après avoir appliqué la pâte à braser, on passe à une étape minutieuse : le **placement des composants** dans le bon sens et au bon endroit.

Pour cela, j'ai utilisé :

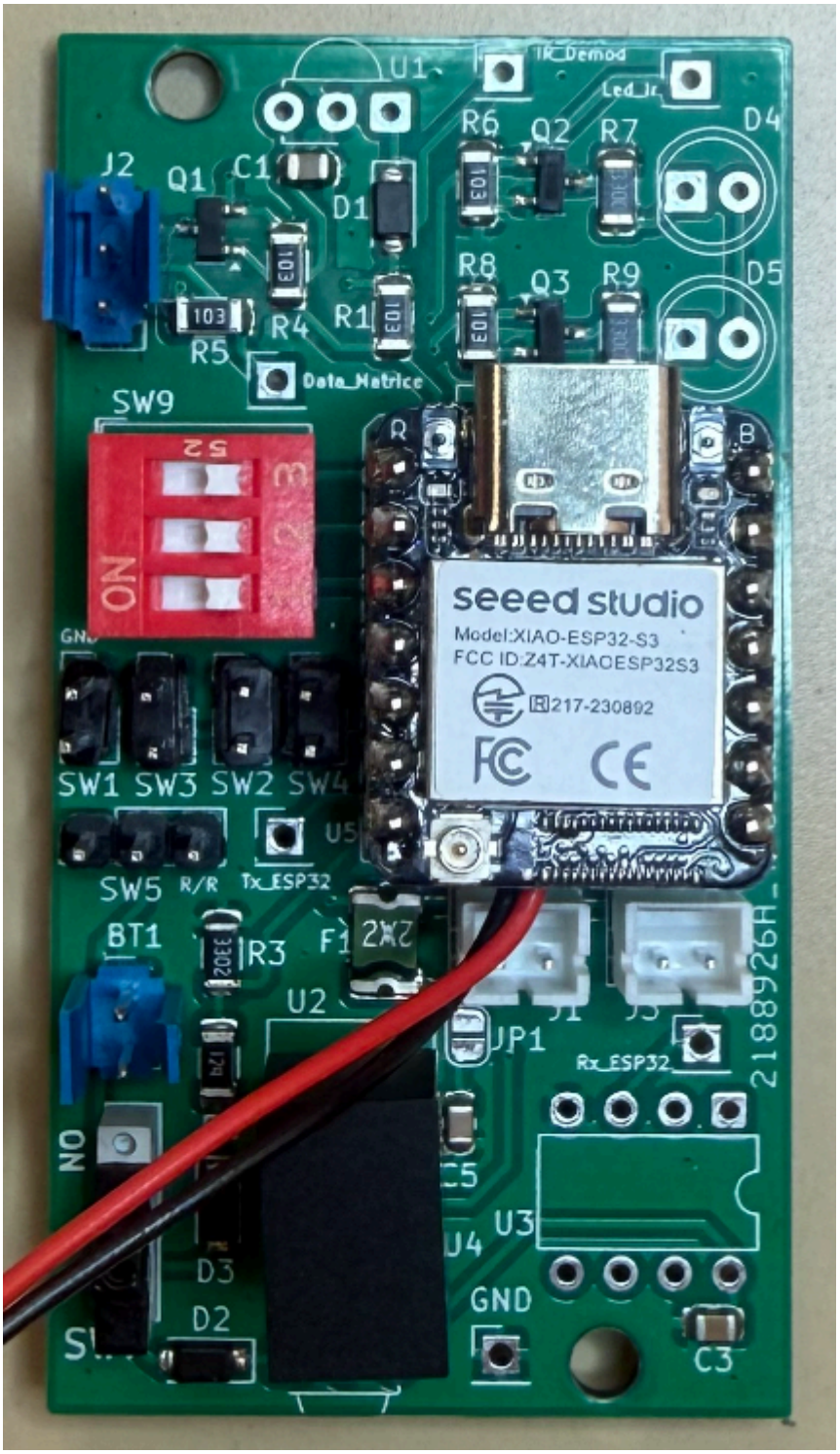
- une **pince brucelle** suffisamment fine pour manipuler les composants avec précision,
- un **multimètre** afin de vérifier le sens des LEDs,
- ainsi que le **schéma en 3D** ou le **routage de la carte** pour positionner correctement les composants.


J1
U4
J10
C8
A0
+3V3
NO
SW7
BT1
SW5
R/R
Tx_ESP32
SW1
SW3
SW2
SW4
GND
SW9
J2
Q1
C1
R5
R4
R1
D1
U1
D2
D3
R2
R3
F1
U2
U5
Data_Matrice
C2
C5
U4
U3
GND
JP1
J1
J3
Rx_ESP32
C3
R8
R6
Q3
Q2
R9
R7
IR_Demod
Led_Jr
D5
D4
2188926A_Y108_250414

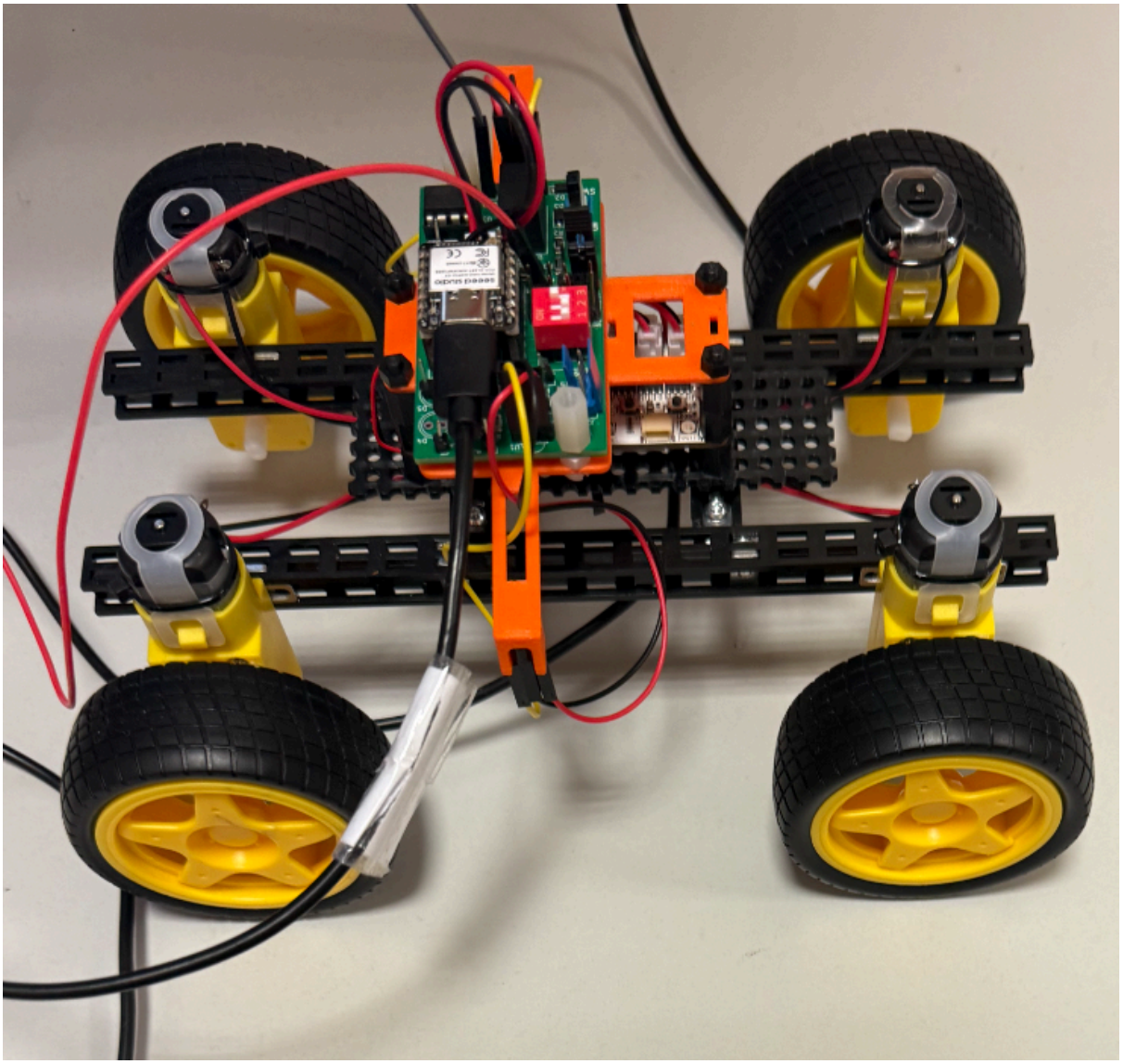

U1
IR_Demod
Led_Ir
R6
Q2
R7
J2
Q1
C1
D1
R8
Q3
R9
R4
R1
R5
Data_Matrice
SW9
GND
SW1
SW3
SW2
SW4
U5
SW5
R/R
Tx_ESP32
BT1
R3
F1
U2
JP1
Rx_ESP32
R2
C5
U3
U4
D3
C2
GND
D2
C3
NO
S



Ensuite on place notre carte dans un four, après ça les composants CMS sont soudés il me reste à souder les composants traversants , voilà ce que ça donne après le soudage



Voilà on peut dire que la fabrication est terminée il me reste à poser la carte sur le support comme ceci.



VIII - FICHE MAINTENANCE

Avant de commencer toute opération de maintenance sur une carte électronique, **il est impératif de la déconnecter de toute source d'alimentation**.
L'utilisation d'outils adaptés est essentielle : **multimètre, analyseur logique, oscilloscope, capacimètre**, etc.

Microcontrôleur principal : ESP32

On peut vérifier que le module ne surchauffe pas en fonctionnement (température excessive = défaillance possible).
Il est aussi important de tester la **connexion USB** pour s'assurer qu'il est reconnu par l'ordinateur ou l'environnement de développement.

Microcontrôleur secondaire : ATtiny85

Ce microcontrôleur doit être **alimenté en 3,3 V** (à mesurer avec un multimètre).
On peut aussi tester le fonctionnement de la broche **PB0**, utilisée pour la communication UART, à l'aide d'un **analyseur logique**.

Démodulateurs IR : TSOP382xx

Pour tester les démodulateurs infrarouges, on peut utiliser une **télécommande infrarouge** et vérifier la réception des signaux à l'aide d'un **analyseur logique** ou d'un oscilloscope.

Transistors : SN7002

Un **multimètre en mode test de diode** permet de vérifier qu'il n'y a pas de court-circuit entre les bornes (Drain, Source, Gate). On vérifie également qu'ils commutent correctement.

Régulateur de tension : TSR 1-2450

Le régulateur doit fournir une **tension de sortie stable de 5V**. On vérifie à la fois la tension d'entrée et la tension de sortie avec un **multimètre**.

Condensateurs

Les condensateurs peuvent être testés à l'aide d'un **capacimètre** ou d'un testeur ESR. On vérifie aussi visuellement qu'ils ne sont ni **bombés ni fuyants**, ce qui indiquerait un vieillissement ou un défaut.

Batterie

La tension de la batterie doit être mesurée régulièrement, est ne doit pas être en dessous de 9V.

IX - FICHE DE MISE EN SERVICE

Après avoir câblé la carte, nous passons à sa **mise en service**.

Dans un premier temps, j'ai testé le **bargraphe de LEDs programmable** en utilisant la librairie **"Adafruit NeoPixel"**.
Dans un second temps, j'ai vérifié le fonctionnement du **régulateur de tension** à l'aide d'une pile rechargeable de 9 V.
Ce régulateur a pour rôle d'abaisser la tension afin de pouvoir alimenter correctement le microcontrôleur ou d'autres composants sensibles.

Ensuite, nous avons testé la **communication infrarouge avec la TOM**.
L'objectif est d'incrémenter le bargraphe à chaque fois que la TOM est vidée. Pour cela, nous utilisons un **démodulateur infrarouge (IR)**, qui nécessite la programmation de l'**ATTiny** pour fonctionner correctement.

Concernant la **détection des chocs**, un code spécifique a été mis en place pour supprimer les **rebonds**. Ce code permet également d'éteindre une LED à chaque détection de **collision**.

L'ensemble de ces fonctions est **géré par un ESP32**, qui centralise les différentes interactions entre les composants.

X - ESSAI COMMUNICATION

Après avoir câblé la carte et l'avoir fixée sur un support, nous devons maintenant établir la communication avec la **TOM**.
Pour cela, nous programmons l'**ESP32** avec un code qui remplit plusieurs fonctions, notamment :

- allumer une LED au moment où un signal infrarouge, émis par la TOM, est reçu.

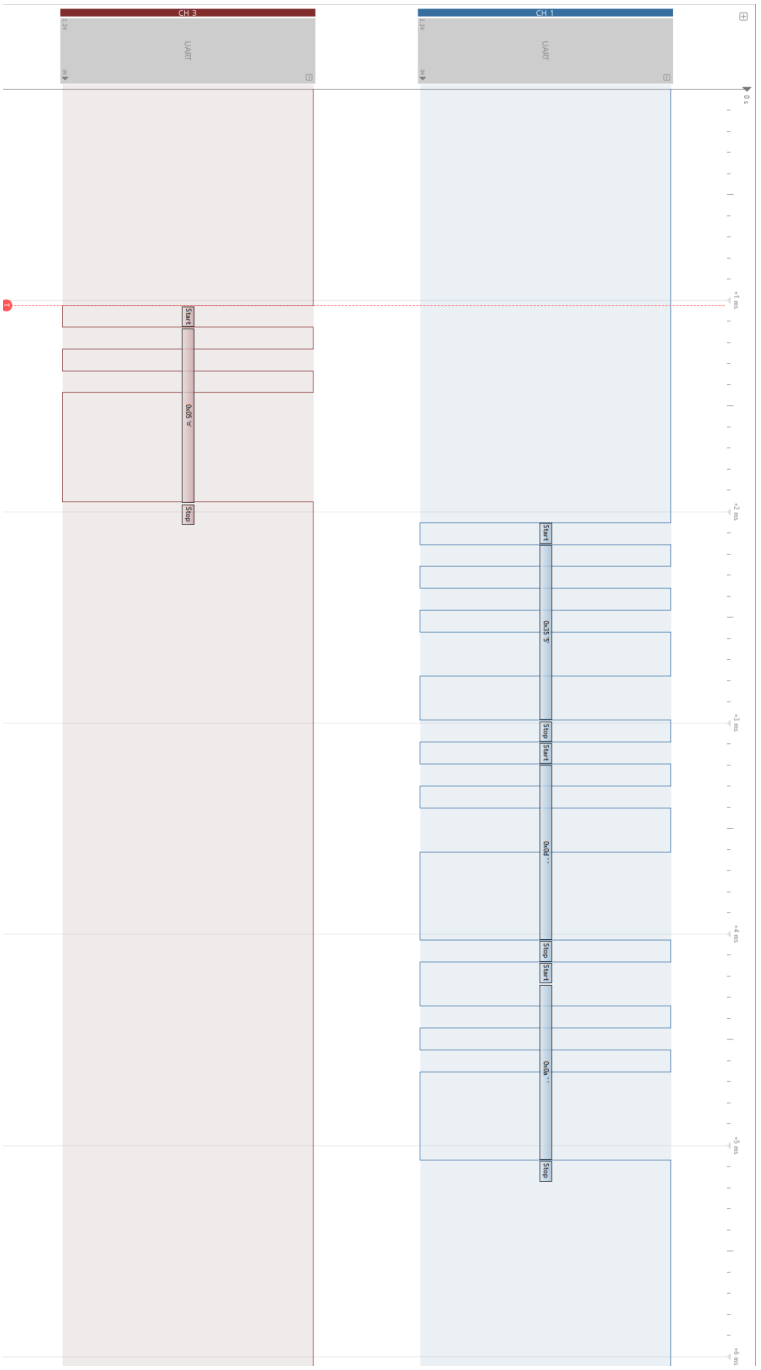
Afin de s'assurer que les bonnes données infrarouges sont réceptionnées, des **switches** ont été installés. Il est essentiel que le switch présent sur la **TOM** et celui sur la **BOM** soient réglés sur le **même numéro**. Ce numéro permet d'identifier la couleur associée à la TOM et à la BOM.

Ensuite, nous visualisons les données reçues à la **sortie de l'ATTiny** ainsi qu'à la **sortie de l'ESP32**.

- En **rouge**, nous observons les données issues de l'ATTiny.
- En **bleu**, celles issues de l'ESP32.

Dans le code de l'émetteur, un **"5"** est envoyé.
On constate qu'en rouge, le **"5"** est bien reçu à la sortie de l'ATTiny. En revanche, à la sortie de l'ESP32 (en bleu), plusieurs éléments apparaissent :

- Le **"0x35"**, qui correspond à la valeur ASCII du chiffre **"5"**,
- Les autres données proviennent des instructions **println** présentes dans le code, qui ajoutent des retours à la ligne ou des textes complémentaires dans le flux série.


CH 3
UART
CH 1
UART
Start
Stop
Start
Stop
Start
Stop

XI - CONCLUSION

Ce projet, mené dans le cadre de ma deuxième année de BTS, m'a permis de participer activement au développement de l'**ESTOM** (Espace de Sensibilisation au Tri des Ordures Ménagères). L'objectif principal était de créer un espace interactif et pédagogique à destination des enfants, en s'appuyant sur des éléments concrets comme la **BOM** (Benne à Ordures Ménagères) et la **TOM** (Tour à Ordures Ménagères), afin de sensibiliser au tri des déchets.

Tout au long du projet, j'ai été confronté à plusieurs défis techniques : gestion des **rebonds** sur les interrupteurs, communication infrarouge fiable, intégration de composants CMS, et programmation embarquée sur différents microcontrôleurs. Ces expériences m'ont permis de renforcer mes compétences en **électronique**, **programmation** (notamment sur l'ESP32 et l'ATTiny), et en **diagnostic de systèmes embarqués**.

Ce projet m'a non seulement permis d'appliquer mes connaissances théoriques, mais aussi de développer une méthode de travail rigoureuse, allant du câblage jusqu'aux tests fonctionnels. Je ressors de cette expérience plus autonome, plus précis dans mon travail, et avec une meilleure compréhension des enjeux liés à la conception de systèmes électroniques communicants.

Partie Personnelle (Nathan PASCAL - ER 2) :

TOM

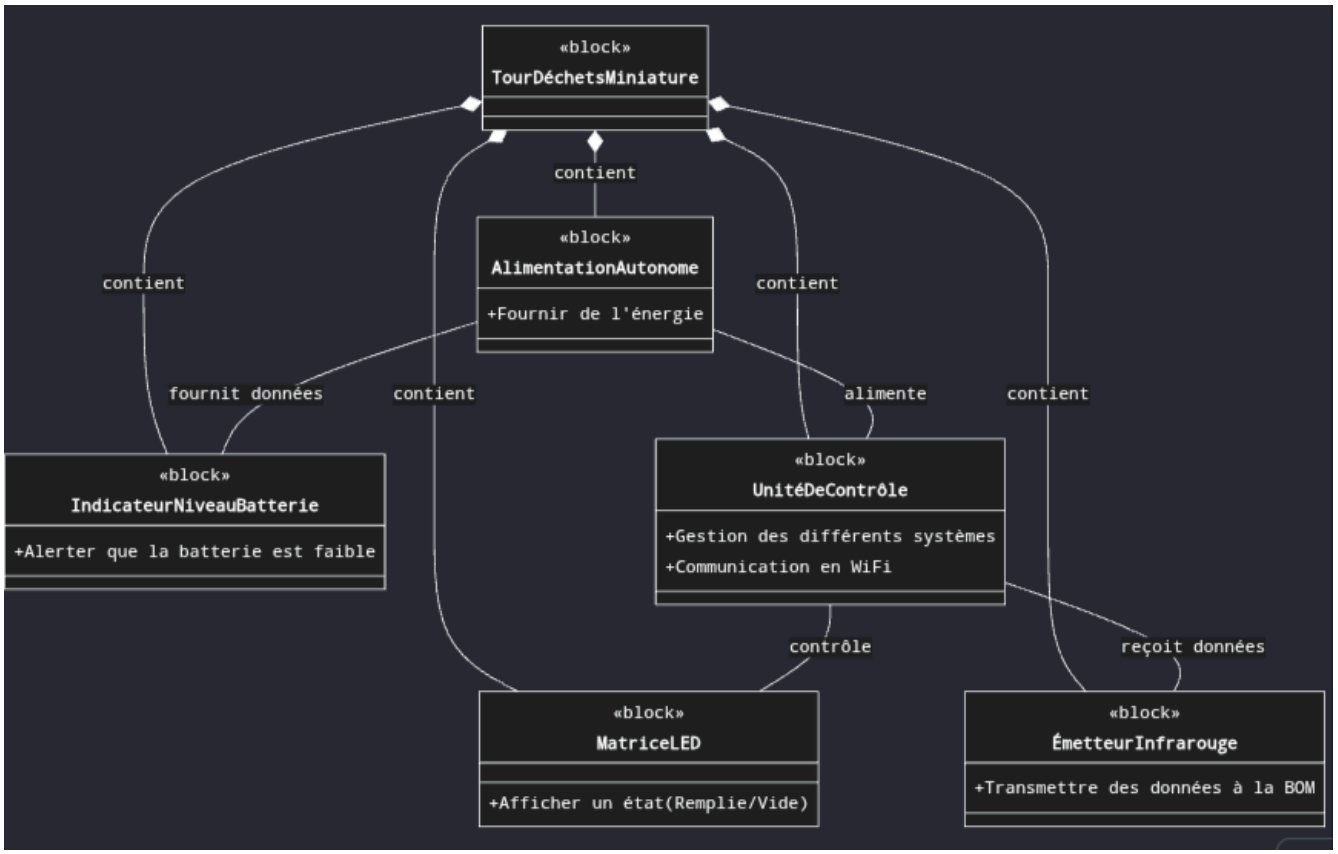
I -Présentation simple de ma partie

I.1 - Présentation TOM

Les **TOM (Tours à Ordures Ménagères)** sont des bacs de collecte, chacun pour un type de déchet : emballage, verre, carton ou déchets ménagers. Pendant l'activité, des camions bennes télécommandés (BOM) viennent récupérer les déchets des TOM dans un temps limité, transformant ainsi le tri en un **jeu éducatif.**

Chaque TOM est muni d'un émetteur infrarouge qui lui permet de communiquer avec la BOM qui possède elle-même deux récepteurs infrarouge . De plus, ils communiquent via Wi-Fi avec un contrôleur principal (Raspberry Pi), qui gère tout le système. Grâce à cela, le transfert des déchets est simulé.

I.2 - Diagramme de Bloc Interne



I.3 - Diagramme de Gantt

Diagramme de gantt prévisionnelle :

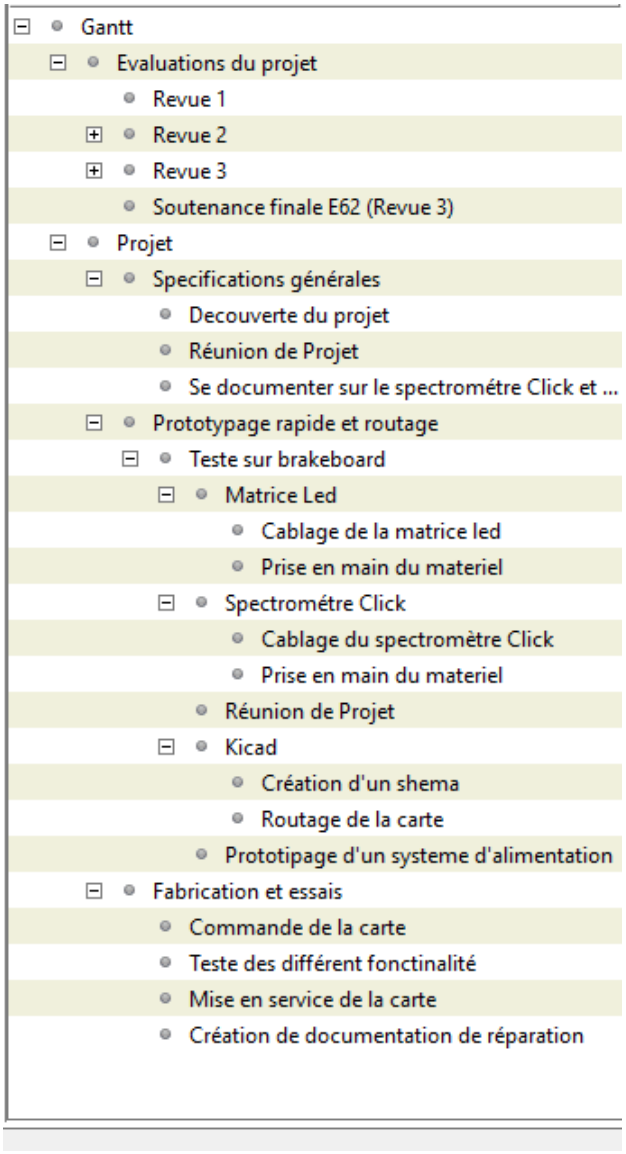


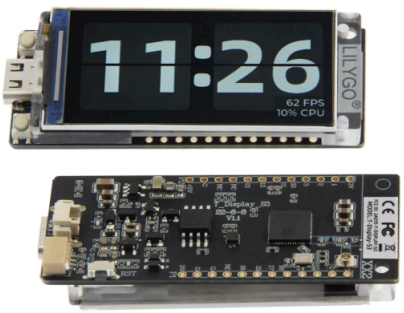
Diagramme de gantt finale :

- Projet
  - Specifications générales
    - Decouverte du projet
    - Réunion de Projet
    - Se documenter sur le spectrométre Click e...
  - Prototypage rapide et routage
    - Teste sur brakeboard
      - Matrice Led
        - Cablage de la matrice led
        - Prise en main du materiel
      - Spectrométre Click
        - Cablage du spectromètre Click
        - Prise en main du materiel
      - Réunion de Projet
      - Teste sur la comunication Infrarouge
        - Se documenter sur les différent co...
        - Cablage Comunication infrarouge
        - Teste de la comunication Infrarouge
      - Teste de consomation et cration d'un ...
        - Se documenter sur les différent co...
        - Cablage De l'alimentation
  - Fabrication et essais
    - Kicad
      - Création d'un shema
      - Routage de la carte
    - Commande de la carte
    - Teste des différent fonctinalité
    - Mise en service de la carte
    - Création de documentation de réparation

I.4 - Présentation des différents outils et matériaux utilisés

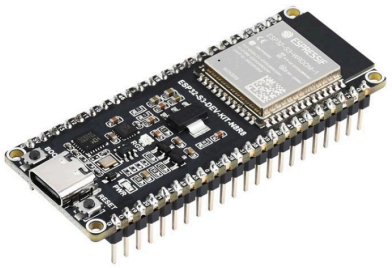
Durant ce projet j'ai eu l'occasion d'utiliser différentes cartes de développement, cela est dû au fait que nous devions choisir la carte la plus optimisée pour mon projet tout en gardant toutes les fonctionnalités importantes (Suffisamment D'entrée/Sortie, Communication Wifi, Possibilité d'installer une alimentation externe, Compatible avec les librairies arduino, Carte compacte). Voici la liste exhaustive de toute les cartes utilisées durant ce projet avec les différents tests effectués :

- Esp-32-S3R8 T-Display Lilygo



- Note : La carte ESP32-S3R8 a été utilisée car nous la possédions déjà. Elle nous a permis de vérifier le bon fonctionnement des différentes fonctionnalités dont nous avions besoin. Cependant, nous l'avons remplacée car elle comporte des fonctionnalités superflues, et des cartes moins onéreuses intégrant ce dont nous avions besoin étaient disponibles.
- Prix : 26 € TTC

- Esp-32-S3-WROOM-1



- Note : Cette carte était la plus avantageuse en termes de prix, mais après avoir réévalué la taille souhaitée, elle ne s'est pas révélée optimale. Malgré cela, j'ai eu le temps de réaliser plusieurs tests montrant qu'elle aurait pu convenir à notre projet. De plus, avant que nous changions de carte, j'ai pu réaliser un **schéma structurel** de celle-ci.
- Prix : 3,90 € TT

- XIAO-Esp-32-C3



- Note : Cette carte a été sélectionnée pour sa taille compacte et son prix attractif, mais son utilisation a été interrompue dès que nos tests sur l'infrarouge se sont révélés infructueux
- Prix : 6,5 € TTC

- XIAO-Esp-32-S3



  - Note : Nous avons opté pour cette carte car elle permet la communication infrarouge tout en conservant un gabarit identique à celui de la XIAO ESP32-C3, tout en restant à un prix relativement accessible.
  - Prix : 9 € TTC

II -Objectifs généraux

II.1 -Besoins et exigences fonctionnelles

Dans le cadre du projet ESTOM, les **Tours d'Ordures Ménagères (TOM)** sont super importantes pour simuler le tri sélectif. Chaque TOM est un point de collecte où les joueurs, à bord des camions BOM, doivent récupérer les déchets adaptés. Pour que l'expérience soit interactive et immersive, il faut concevoir les TOM de façon réaliste, avec un code couleur facile à reconnaître : **jaune les embalages recyclables, vert pour le verre, bleu pour le carton et noir pour les déchets ménagers**.

Le fonctionnement des TOM repose sur un **système de détection infrarouge** qui permet d'interagir avec les BOM. Chaque tour a un **émetteur infrarouge** qui envoie des données en continue, quand un camion s'approche et qui détecte une TOM compatible un timer de **cinq secondes est lancé.** matrice led positionné sur la TOM va clignoter pour simuler le transfert des déchets. À ce moment-là, la TOM s'éteint temporairement pour montrer qu'elle a été vidée. **Toutes les cinq minutes**, à la fin de la session de jeu, l'ensemble des TOM se réinitialise automatiquement et donc se rallume permettent au utilisateur de voir les différentes tours allumées.

Finalement, les TOM cherchent à offrir une interaction avec les BOM tout en garantissant que le processus de tri des déchets est bien représenté. Elles doivent être assez **solides** pour

résister à une utilisation répétée lors des événements de sensibilisation de plus il faudra que le jeu soit facile à comprendre, pour permettre à petits et grands d'y jouer.

II.2 -Contraintes techniques et matérielles

Pour commencer, chaque TOM est dotée d'un **système type ESP32** qui gère la communication avec les BOM et le superviseur (raspberry pi) pour pouvoir contrôler les différents capteurs. Grâce à l'émetteur infrarouge installé sur chaque TOM, on peut détecter la présence d'un camion à proximité et déclencher l'animation de collecte des déchets.

Le système électronique doit être **énergétiquement efficace** afin de fonctionner longtemps sans avoir besoin de le recharger. En plus, les TOM doivent être conçues pour être facilement configurables via un programme intégré, ce qui permet d'adapter certains paramètres du jeu, comme la fréquence de réinitialisation ou la gestion des interactions avec les BOM.

En ce qui concerne la fabrication, on va utiliser **l'impression 3D** pour réaliser les TOM, ce qui nous donne la flexibilité de peaufiner leur design selon les besoins du projet. Elles devraient ressembler à de vrais conteneurs de tri sélectif tout en étant assez légères pour être facilement transportées et installées lors des événements. Il est essentiel que leur structure soit robuste pour résister à une manipulation fréquente, en particulier par les joueurs qui interagiront avec elles de manière répétée.

Pour terminer, la **sécurisation des communications** entre les TOM et les autres éléments du système est cruciale. Toutes les données doivent être **chiffrées** pour éviter toute tentative d'intrusion ou de perturbation extérieure mais cela reste facultatif.

III -Les début du projet

III.1 -Tests matrice led

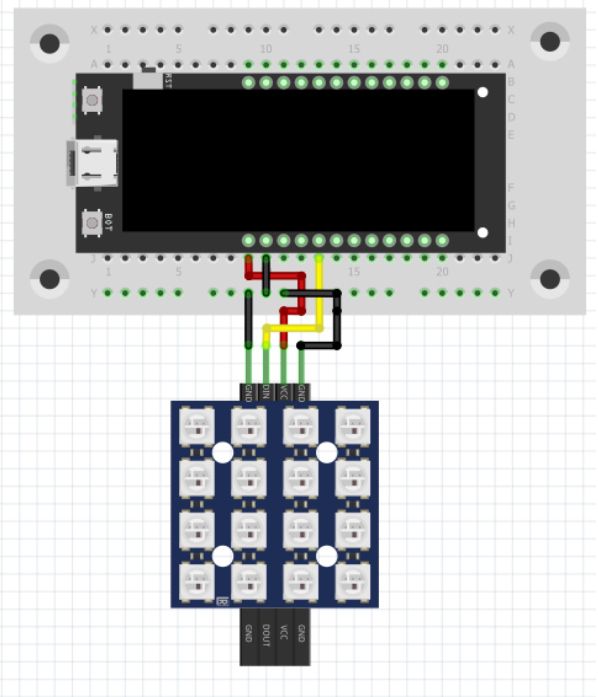
J'ai débuté le projet par une série de tests, les tout premier on était sur la matrice led pour mes tests j'ai pu utiliser la carte **ESP32-S3R8 (Carte de teste) et** la matrice **LED NeoPixel 4x4 (WS2812)**





Les premiers tests ont montré que la matrice **LED NeoPixel 4x4** fonctionnait bien avec **ESP32-S3R8** en utilisant la bibliothèque AdaFruit NeoPixel.

voici du montage :



et voici le code utilisé :

```
// NeoPixel Ring simple sketch (c) 2013 Shae Erisson

// Released under the GPLv3 license to match the rest of the
// Adafruit NeoPixel library

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

#define PIN       13 // On Trinket or Gemma, suggest changing this to 1

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 16 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels

void setup() {
  // These lines are specifically to support the Adafruit Trinket 5V 16 MHz.
  // Any other board, you can remove this part (but no harm leaving it):
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);
#endif
  // END of Trinket-specific code.

  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}

void loop() {

  // The first NeoPixel in a strand is #0, second is 1, all the way up
  // to the count of pixels minus one.
  for(int i=0; i<NUMPIXELS; i++) { // For each pixel...

    // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
    // Here we're using a moderately bright green color:
    pixels.setPixelColor(i, pixels.Color(0, 50, 0));

    pixels.show();   // Send the updated pixel colors to the hardware.

    delay(DELAYVAL); // Pause before next pass through loop
  }
}
```

Au début, notre principal souci était la différence de tension entre les bus de communication de **l'ESP 32 (3,3V)** et les **LEDs (5V)**. Mais malgré ça aucun problème de fonctionnement à première vue. Même si le système fonctionne sans level shifter pour le moment,en ajouter un pourrait vraiment renforcer sa fiabilité à long terme.

De plus j'ai eu l'occasion de tester sa consommation pour nous permettre de prévoir le futur système d'alimentation voici les mesures que j'ai pu faire avec mon ampèremètre relié en série.voici mes différentes mesures :

Pour tester toutes mes valeurs j'ai utilisé la fonction :
(pixels.setPixelColor(i, pixels.Color(0, 0, 0));)
c'est fonction viens de la libs adafruit NeoPixel qui permet de bougé la valeur de l'intensité lumineuse en fonction de la valeur de couleur

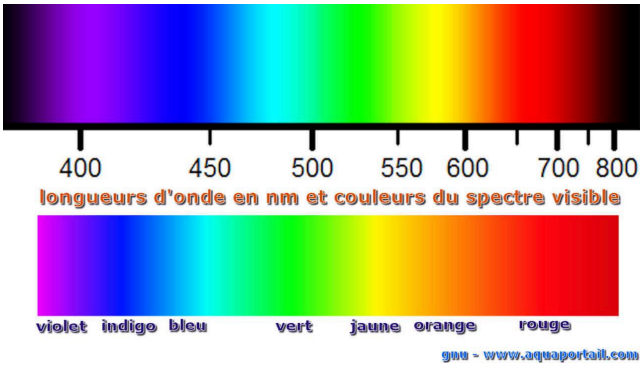
0,50,0 = 48 mA
0,100,0 = 87 mA
0,150,0 = 125 mA
0,200,0 = 162 mA
0,255,0 = 200 mA
255,255,0 = 328 mA
255,255,255 = 385 mA

III.2 -Tests capteur de lumière

Pour mon deuxième teste, j'ai travaillé sur le spectromètre, vu qu'on avait choisi une technique qui repose sur la détection de la lumière. Ce capteur (**AS7341-DLGT**) devait nous aider à mesurer la lumière émise par la TOM.

Mon travail, c'était de comprendre comment bien utiliser ce spectromètre : comment il fonctionnait et comment l'adapter à nos besoins. Ensuite, il fallait aussi choisir les bonnes couleurs pour chaque TOM afin d'éviter les soucis de détection, surtout si les teintes étaient trop proches dans le spectre.

C'était vraiment essentiel, parce que si deux couleurs était trop proche dans le spectre, on aurait pu les confondre en pensant qu'elles provenaient de deux TOM différents alors qu'elles étaient identiques ou bien l'inverse



Voici les différente couleurs que j'ai pu récupérer grâce à mes mesures :
Définition du rouge par le Spectromètre à 15 Cm :

```
F1 415nm : 0.02
F2 445nm : 0.03
F3 480nm : 0.03
F4 515nm : 0.04
F5 555nm : 0.07
F6 590nm : 0.45
F7 630nm : 0.91
F8 680nm : 0.06
Clear    : 0.54
NIR      : 0.09
```

Définition du rouge par le Spectromètre à 5 Cm :

```
F1 415nm : 0.10
F2 445nm : 0.07
F3 480nm : 0.11
F4 515nm : 0.14
F5 555nm : 0.11
F6 590nm : 0.91
F7 630nm : 0.91
F8 680nm : 0.20
Clear    : 0.91
NIR      : 0.42
```

on peut remarquer que le rouge né pas définie rouge part le spectrométrie mais plus rouge orangé voir jaune orangé.

Ce qui m'a paru le plus intéressant, c'est qu'il ne voit pas la lumière de la même manière avec seulement **10 cm de différence** et du coup je trouve ça plutôt intéressant de le noter car si on voulait définir une distance "max" entre le camion et la TOM grâce à ses valeurs on aurait pu le faire.

III.3 -Problématique posé par le spectromètre

Le premier défi, c'était les lumières parasites, des lumières venant de l'extérieur, cela risquait de fausser la détection et d'envoyer des informations incorrectes au camion.

Le deuxième souci identifié était les limites du spectromètre pour une identification précise. le spectromètre peut voir la couleur jaune, mais ne peux pas dire s'il s'agissait de la TOM Jaune 1, Jaune 2, etc. Ce manque de précision pour distinguer les TOM qui ont la même teinte

empêchait le système d'identifier exactement quelle TOM a été récupérée. En conséquence, cela bloquait l'extinction du bon panneau LED. C'est cette limitation qui nous a poussés à revoir complètement notre approche technique. Après cette remise en question du système qui possède beaucoup de limites, nous avons décidé d'utiliser un autre système plus simple qui serait plus permissif, la communication infrarouge.

IV -Les premier test après la remise en question du système de communication

IV.1 -Tests communication infrarouge

La première étape après la réunion a consisté à explorer la communication infrarouge avec **ESP32-S3R8 (Carte de test)** afin de bien comprendre son fonctionnement et ses applications potentielles. Pour cela, nous avons commencé par suivre un tutoriel détaillé sur l'utilisation des modules infrarouges et leur interaction avec un microcontrôleur. Une fois les bases assimilées, nous avons procédé par l'assemblage sur brakeboard de tous en vérifiant que le **Récepteur ir (TSOP38238)** puisse bien être alimenté par l'**ESP32-S3R8** ce qui était bien le cas :

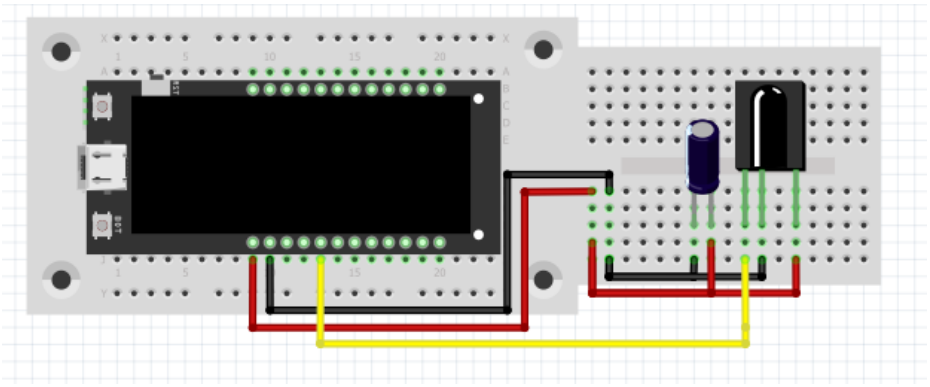
- Alimentation possible sur **ESP32-S3R8**
  - +5V
  - +3.3V
- Alimentation possible que peux recevoire le **TSOP38238**
  - 2.0 V to 5.5 V

Mais de vérifier que le **TSOP38238** puisse bien démoduler la porteuse de 38 kHz qui est un des points clef du protocole de communication Nec qui était aussi le cas :

Carrier frequency	30 kHz	TSOP38230
	33 kHz	TSOP38233
	36 kHz	TSOP38236
	38 kHz	TSOP38238
	40 kHz	TSOP38240
	56 kHz	TSOP38256

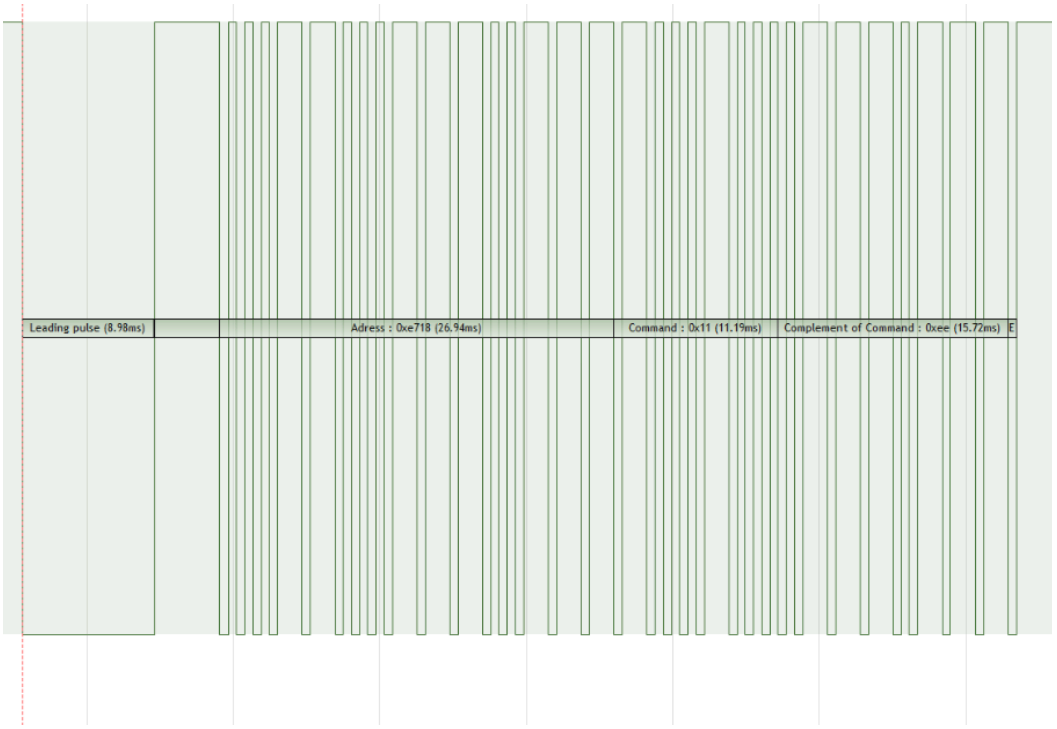
Puis nous avons l'installation de la bibliothèque **IRMP**, qui permet de gérer à la fois l'émission et la réception des signaux infrarouges.

Nous avons ensuite effectué des tests préliminaires pour vérifier le bon fonctionnement du système. Pour ce faire, nous avons utilisé une télécommande multimarque afin d'envoyer des signaux vers un montage comprenant un **ESP32-S3R8 (Carte de test)** et un récepteur IR.



Ces essais ont rapidement confirmé que notre récepteur était bien capable de capter et de décoder les codes NEC transmis par la télécommande sans difficulté.

Pour approfondir la compréhension des signaux infrarouges, un analyseur logique **iKaLogic** couplé à la photodiode a été utilisé pour capturer des trames IR. Cette manipulation a permis d'étudier en détail la structure des signaux émis par les télécommandes.

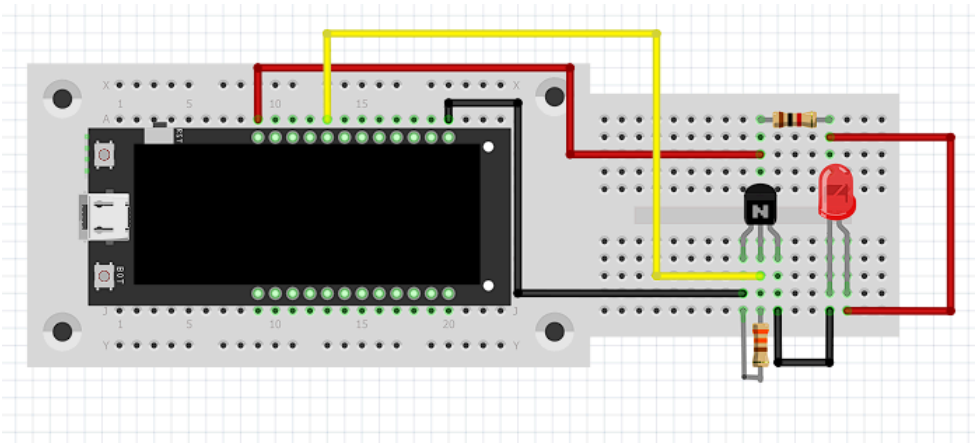


Explication de la trame ici présente :

- Le **lending pulse** et la période d'initiation
- L'**adresse** et composé de 8 bits d'adresse (souvent l'identifiant du fabricant) et 8 bits d'adresse inverse (complément à 1)
- La **commande** et composé de 8 bits qui et le code de la touche

- Le **complément de commande** composé de 8 bits et qui est la commande inversée (complément à 1)

Dans un second temps, nous avons cherché à établir une communication entre deux **ESP32.** Pour cela, nous avons câblé un second montage sur une autre breadboard, en utilisant une autre **ESP32-S3R8(Carte de test)** qui était reliée à un VN 2222 comme relais pour nous permettre d'émettre avec une led infrarouge (cette manoeuvre nous permet d'éviter de tirer trop de courant sur le microcontrôleur).

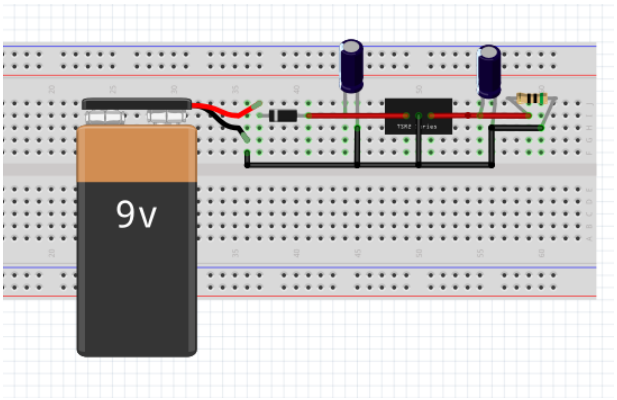


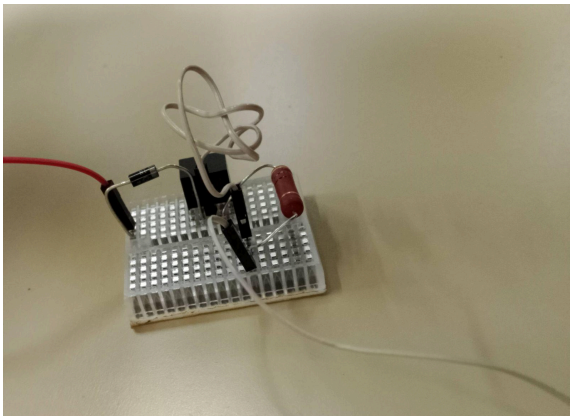
Une fois le circuit correctement assemblé, nous avons pu mettre en œuvre la librairie **IRMP** observer que les deux appareils communiquent ensemble , ce qui démontre le bon fonctionnement de la transmission des signaux entre l'émetteur et le récepteur.

Ces premiers tests nous ont permis de confirmer la configuration initiale de notre système, ainsi que d'identifier certains problèmes que nous devrons résoudre ultérieurement (communication à plus de 5 mètre plus rebond des la lumière infrarouge)

IV.2 -Tests du régulateur de tension et mesure d'autonomie

Dans un second temps, des tests ont été menés pour évaluer les performances d'un régulateur de tension et mesurer l'autonomie d'une batterie 9V 250mA.
voici les schéma plus le montage réel





Un programme Arduino spécifique a été utilisé pour enregistrer la durée de décharge jusqu'à ce que la tension descende en dessous d'un seuil prédéfini (5V) aux bornes du régulateur.

Voici le code :

```
// Mesure durée
unsigned long startTime;
unsigned long totalTime[4];
const int PIN = A0; /*Déclaration de la broche analogique qui servira à faire la mesure*/
int value;
int value1;
unsigned long stopTime;
unsigned long total;

void setup() {
  Serial.begin(9600);
  startTime = millis();        /*définition de l'heure de début*/
}

void loop() {
  value = analogRead(PIN);
  Serial.print("reg 5V : ");
  Serial.println (value);
  if((value < 103)|(value<410)){ /*si la tension est très faible alors*/
                  stopTime = millis(); /*définition de l'heure de fin*/
                  total = stopTime - startTime; /*calcul de la durée en ms*/

                  totalTime[0] = total/3600000; /*coversion des ms en h*/
                  totalTime[1] = (total % 3600000) / 60000; /*conversion des ms en m*/
                  totalTime[2] = ((total % 3600000) % 60000) / 1000; /*conversion des ms en s*/
                  totalTime[3] = ((total % 3600000) % 60000) % 1000;

                  Serial.print("Durée totale :");
                  Serial.print(totalTime[0]);
                  Serial.print("h");
                  Serial.print(totalTime[1]);
                  Serial.print("min");
                  Serial.print(totalTime[2]);
```

```
                Serial.print("s");
                Serial.print(totalTime[3]);
                Serial.println("ms");

                while(1);
  }
}
```

et voici le résultat :

```
reg 5V : 466
reg 5V : 409
Durée totale :0h38min28s793ms
```

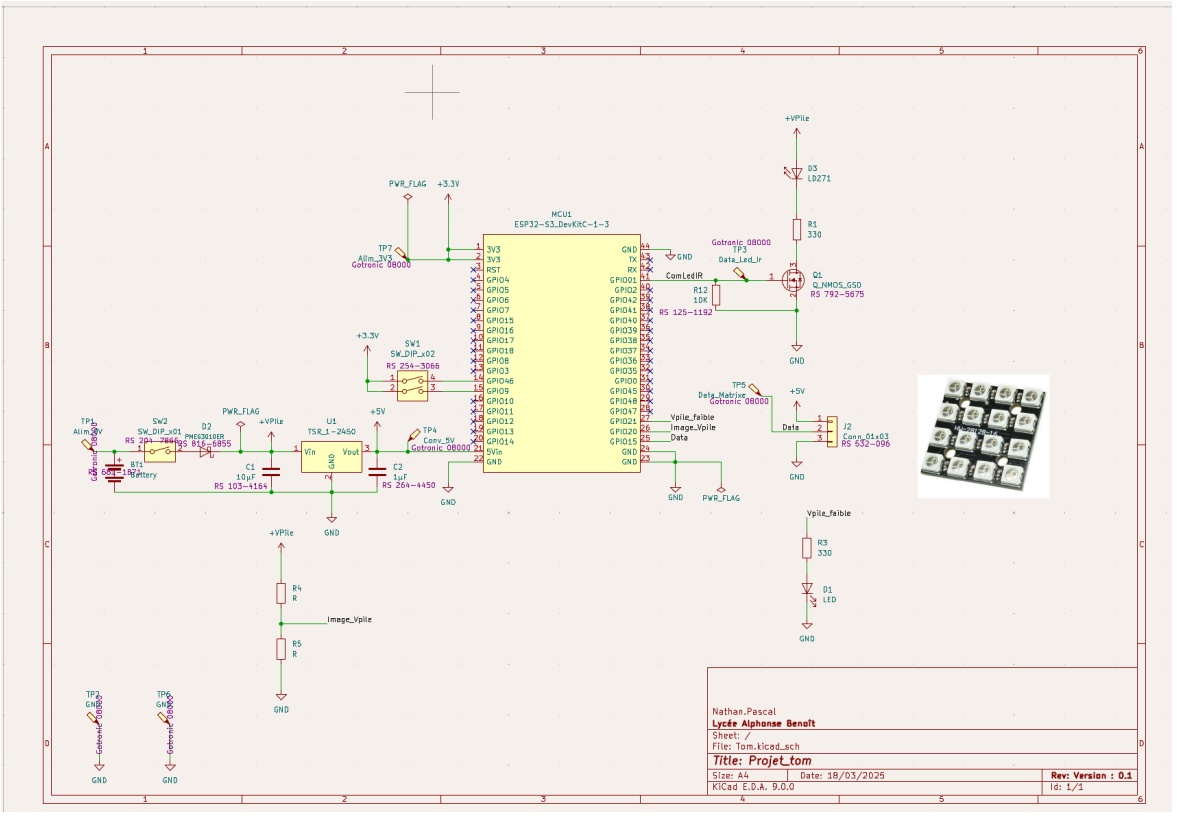
Voici une photo attestant que la valeur est bien inférieur à 5 V



Grâce à ce test, nous avons pu nous rendre compte que la batterie était sous-dimensionnée. Cela nous permettra de la changer pour une batterie de 12V 1.2A aux plon ce qui théoriquement devrait augmenter de X4.8 l'autonomie de la TOM pour atteindre 3h et 4 minutes

V -Schéma et Routage

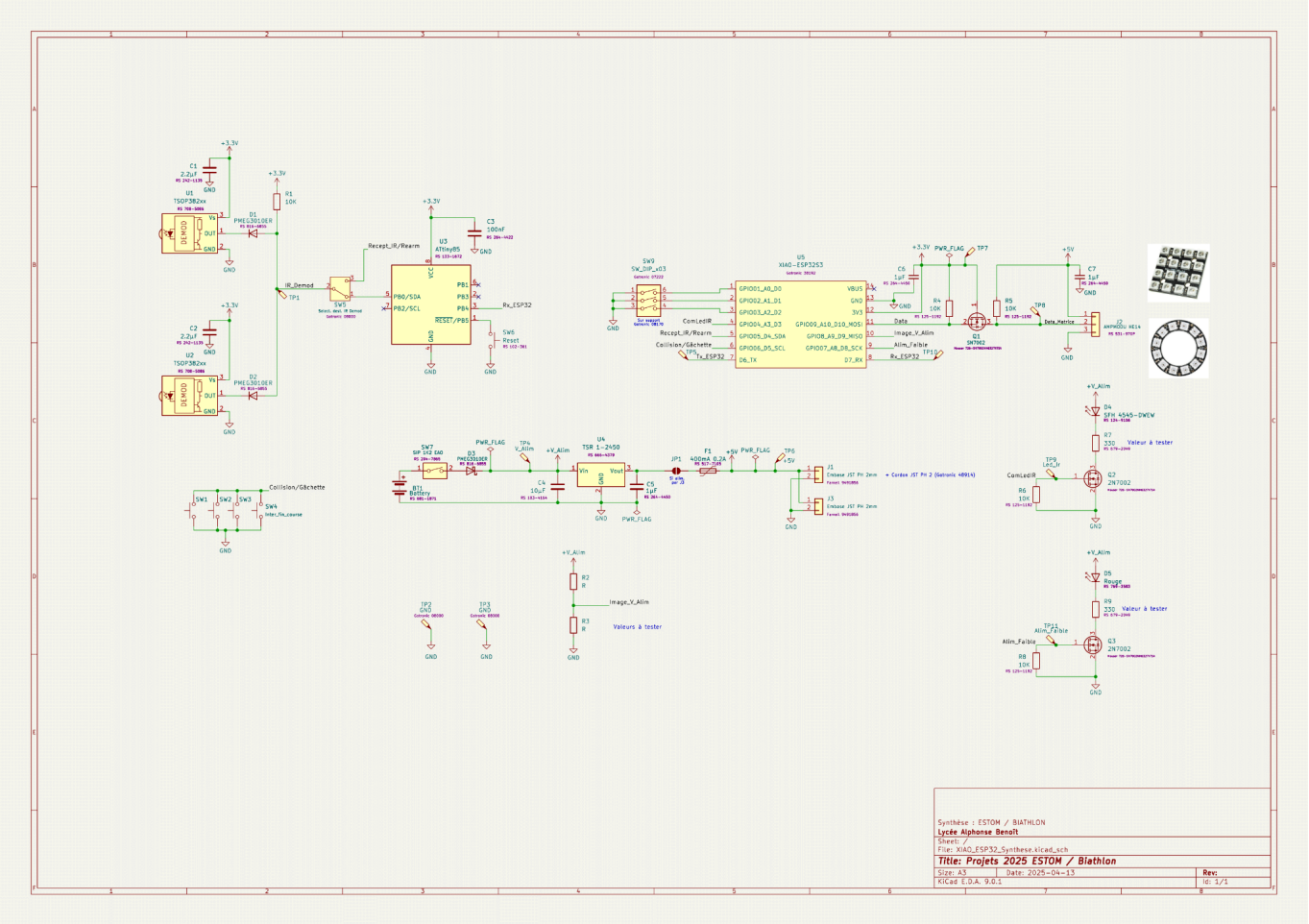
V.1 -Création du schéma pour la carte Esp-32-S3-WROOM-1



Pendant la création de ce schéma, on a incorporé plusieurs fonctionnalités, comme une LED qui indique quand la batterie est faible, Il y a aussi un système d'interrupteurs qui permet aux IR de créer un code adaptatif en fonction de leur configuration, permettant ainsi de choisir la couleur souhaitée.

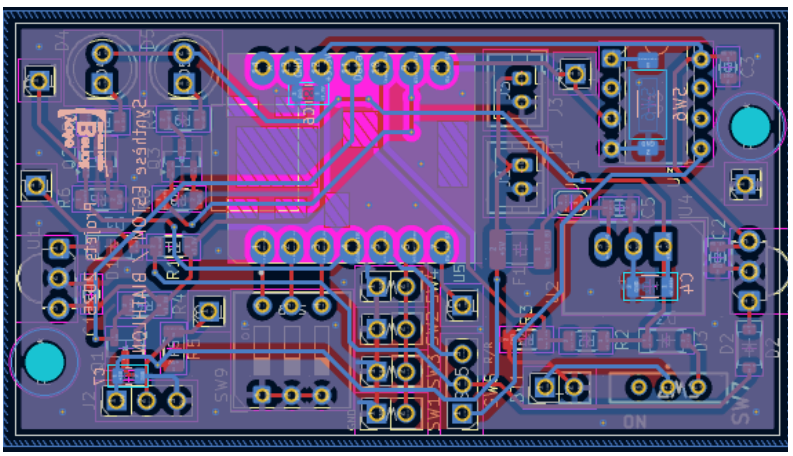
Mais entre-temps nous avons changé de carte de développement pour une plateforme vachement plus compacte le XIAO-ESP32-S3 permettant de réduire la taille finale de notre carte.

V.2 -Création du schéma pour la carte XIAO-Esp-32-S3



Pour ce schéma on peut remarquer que c'est la fusion de mon shema et celui de badr errafi, Il a était réalisé par notre professeur pour permettre d'abaisser le coût de fabrication des carte et proposé un produit finale vachement plus permissible de plus des ajustement on était effectué la led témoins de batterie faible qui utilise le même système que la led Infrarouge et un level shifter pour la matrice led qui vient fiabiliser le montage .

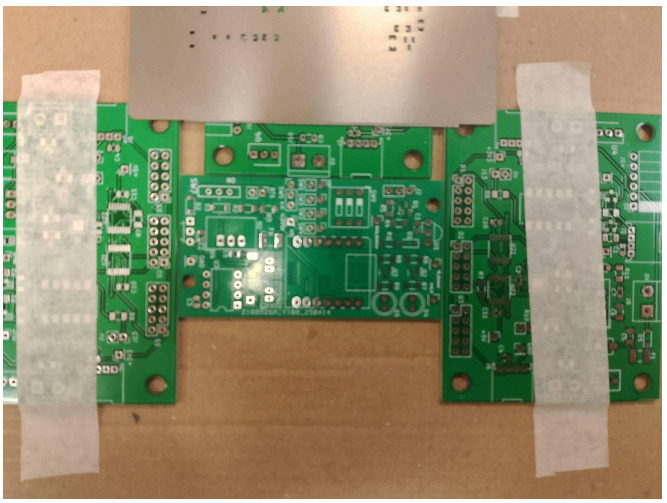
Voici le routage de la carte :

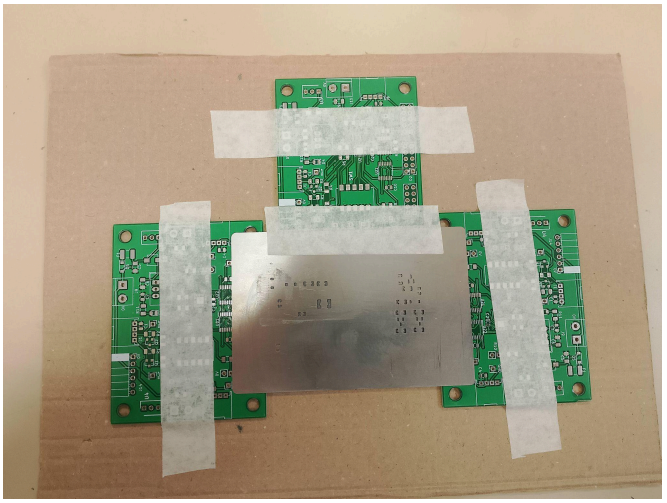


Après ceci nous avons commandé la carte et son stencil chez JLPCB.

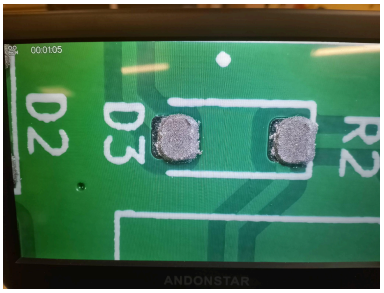
VI -De la fabrication à la mise en service

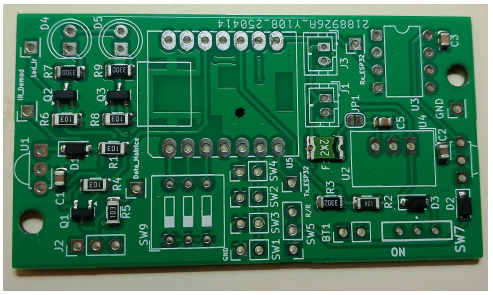
Après avoir reçu la carte je l'ai apprêtée pour appliquer la pâte à braser et on l'a bloquée entre 3 circuits imprimés.



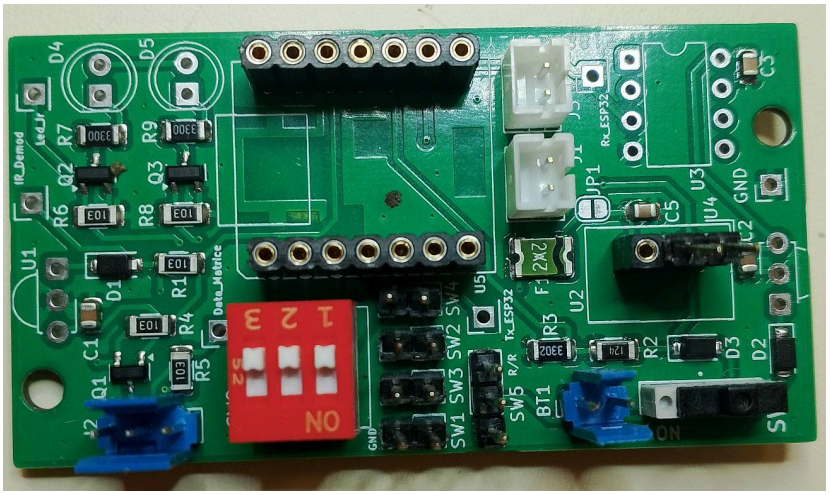


Ensuite, j'ai appliqué la pâte à braser sur le stencil en vérifiant bien que toute la pâte était correctement déposée. Après cela, j'ai délicatement placé les composants pour enfin les mettre au four pendant 2 minute et 50 seconde à 250 C





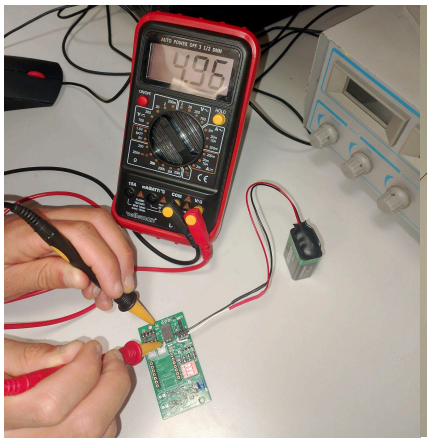
Puis j'ai soudé les différents composants traversant que nous avions à notre disposition.

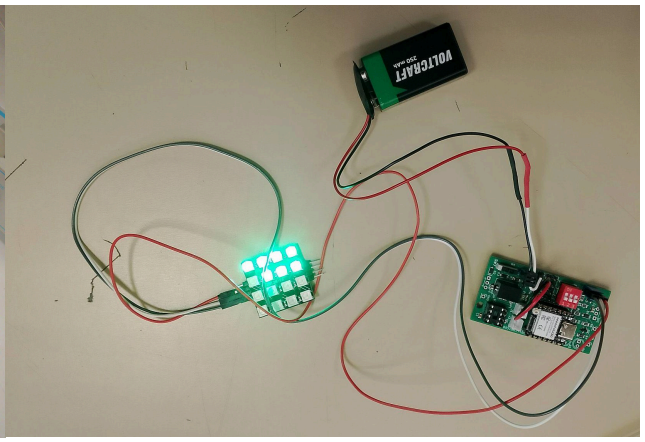


Après avoir fini, on a pu estimer le prix moyen de la fabrication pour une carte ce qui est plutôt intéressant pour évaluer le prix finale pour la fabrication des 40 TOM

Repère	Valeur (Référence)	Tol.±	Equivalent	Fournisseur	Boitier	Remarques	Qté	Condition	ref rapide	Prix U HT	Prix T HT
BT1	Battery			Radiospares			1	Par 10	RS 681-1871	1,009	1,009
C1,C2	2.2µF			Radiospares			2	Par 10	RS 242-1135	0,198	0,396
C3	100nF			Radiospares			1	Par	RS 264-4422	0,023	0,023
C4	10µF			Radiospares			1	Par	RS 103-4164	0,023	0,023
C5,C6,C7	1µF			Radiospares			3	Par 25	RS 264-4450	0,04	0,126
D1,D2,D3	PMEG3010ER			Radiospares			3	Par 50	RS 816-6855	0,145	0,435
D4	SFH			Radiospares			1	Par 20	RS 124-9106	0,30	0,304
D5				Radiospares			1	Par 10	RS 769-3583	0,456	0,456
F1	Rouge			Radiospares			1	Par 10	RS 517-7105	0,369	0,369
J1,J3	400mA 0.2A			Farnell			2	Par 100	Farnell 9491856	0,0919	0,184
J2	AMPMODU HE14			Radiospares			1	Par 200	RS 531-970P	0,62	0,623
JP1							1	Par			0,000
Q1,Q2,Q3	2N7002,SN7002			Radiospares			3	Par	SN7002NH6327X	0,02	0,069
R1, R4-R6, R8	10K			Radiospares			5	Par	RS 125-1192	0,023	0,115
R2	120K			Radiospares			1	Par	RS 807-1110	0,03	0,029
R3	33K			Radiospares			1	Par	RS 679-2055	0,026	0,026
R7, R9	330			Radiospares			2	Par	RS 679-2049	0,026	0,052
SW1-SW9				Gotronic			9	Par	Gotronic 08000	0,5	4,500
TP1-TP11				Gotronic			11	Par	Gotronic 08000	0,50	5,500
U1, U2	TSOP382xx			Radiospares			2	Par	RS 708-5086	0,624	1,248
U3	ATtiny85			Radiospares			1	Par	RS 133-1672	1,572	1,572
U4	TSR 1-2450			Radiospares			1	Par	RS 666-4379	5,62	5,620
U5	XIAO-ESP32S3			Gotronic			1	Par	Gotronic 38192	7,170	7,170
Solder Paste				JLCPCB			1	Par 1	7016920	6,160	6,160
2-layer Bare Ringid			Farnell 2472693	JLCPCB			1	Par 5	2472692	0,700	0,700

TOTAL FEUILLES PRECENTES		29,849
TOTAL HT FEUILLE		6,860
TOTAL HT GLOBAL		36,709
COEFFICIENT TVA	1,2	44,051



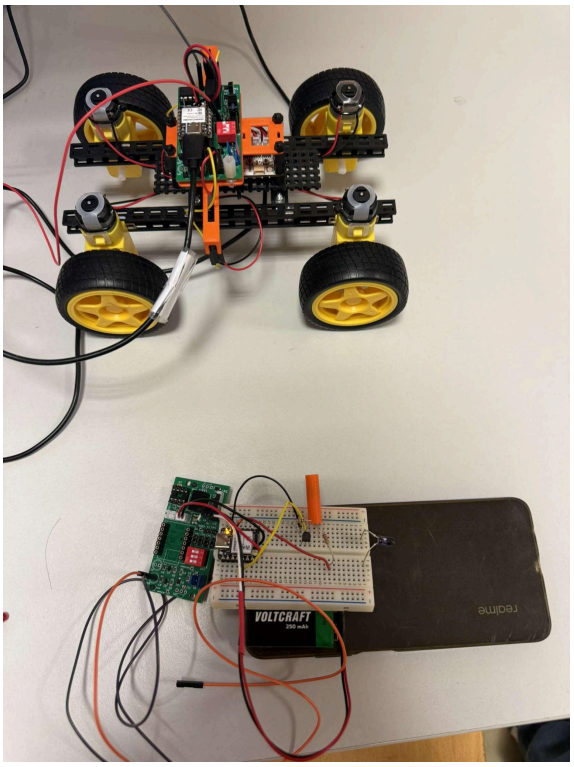


Puis j'ai pu enfin commencé à tester la carte.

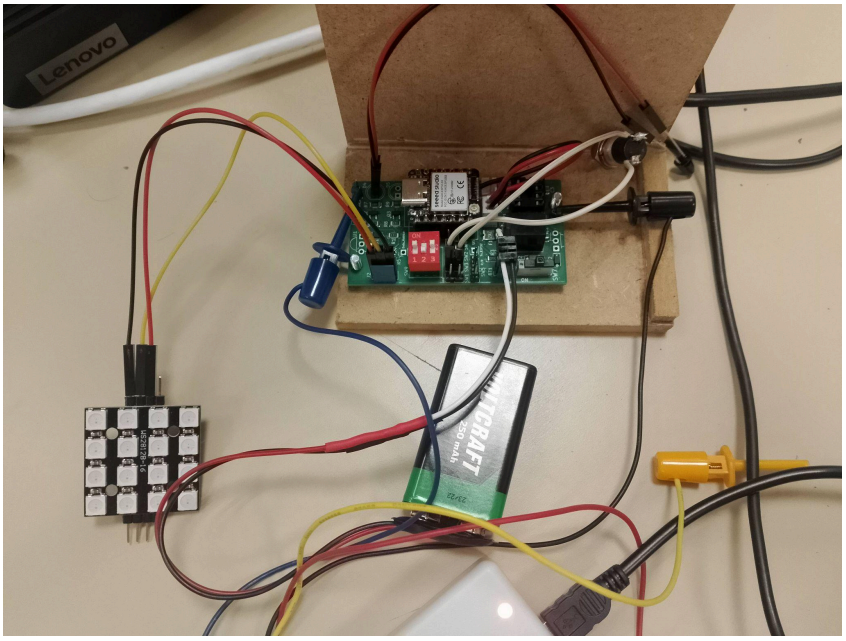
les premiers test se sont portés sur l'alimentation pour voir si notre montage était bien fonctionnel et voir les éventuel défauts de conception ou problème pouvant découler de notre montage.Une question nous a mis en doute quand on s'est demandé que se passerait t'il si on alimentait la carte avec une batterie externe et qu' en plus on se connecte sur usb C qui lui même alimenté la carte en 5V ce qui pouvait créer des problèmes d'alimentation,

mais après quelques recherches, on s'est rendu compte que quand VUSB et VBatterie sont tous les deux à 5 V, la sortie se charge automatiquement avec VUSB. C'est parce que la diode Schottky de VUSB a une chute de tension plus faible (~0,3 V) que celle de VBatterie (~0,7 V). Résultat, seule l'alimentation USB prend le relais, tandis que l'alimentation VRAW est ignorée.

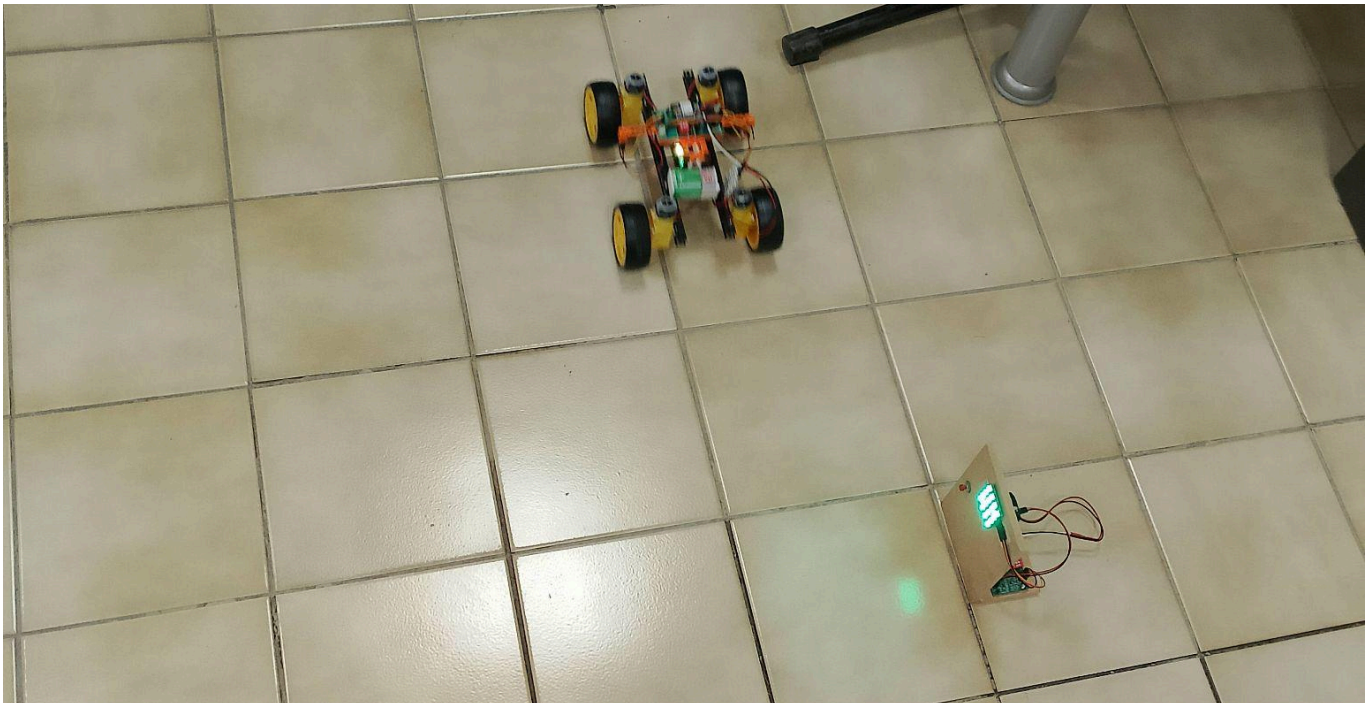
Malgré tout ça pour sécuriser un maximum le montage l'ajoute d'une diode côté VBatterie serait une bonne solution permettant d'abaisser la tension de plus ou moins 0.6V et donc d'avoir un montage plus fiable .



Nous avons ensuite testé la communication IR, et là, on s'est rendu compte que la détection IR fonctionne bien, On a réussi à envoyer des codes à une distance de jusqu'à 4 mètres,ce qui nous a posé un problème car le camion ne devait pas non plus être trop loin de la borne. Du coup, on a décidé de modifier la résistance pour limiter le courant et ainsi réduire la portée de la LED. Après d'autres tests, on a constaté que la résistance de 100kΩ était la plus adaptée pour notre projet. permettant de réduire la portée et aux les rebonds pour atteindre un distance max entre 0 et 20 cm.



Après avoir modifié la résistance, on a eu la possibilité de fabriquer un support en carton compressé. J'ai testé toutes les fonctionnalités avant de l'installer sur le châssis. De plus, j'ai installé un bouton poussoir pour faire mes différents tests de simulation.



Grâce à cette simulation de jeux, on a pu se rendre compte que tout fonctionnait correctement, et qu'il fallait être minutieux pour se placer à la bonne distance de la BOM.

VII -Conclusion

Actuellement, le projet TOM arrive presque à son terme sur certains points. Il nous reste encore quelques étapes nous permettant d'officialiser la fin du projet telles que l'installation et l'ajustement des cartes dans les boîtiers imprimés en 3D, voir si la communication avec le superviseur fonctionne sans problème, mais aussi de tester le code de Guilhem pour voir si les différentes fonctionnalités fonctionnent correctement. De plus, il faudrait faire des tests grandeur nature pour vérifier que le jeu n'est pas trop dur et de permettre d'ajuster d'éventuels problèmes. Cela pourrait reconduire le projet à l'année prochaine pour permettre d'avoir un produit final mieux conçu et totalement fonctionnel.

ANNEXES

Annexes Yannis ZAPOLOWSKI - IR 1

Annexe 1 - Script actuel des BOM:

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>


// WiFi
const char* ssid = "AP-ESTOM";
const char* password = "Projet-ESTOM_2025";


// Adresse IP et port du serveur TCP
const char* tcpAddress = "192.168.0.10";
const uint16_t tcpPort = 5005;


// Client TCP unique (réception + envoi)
WiFiClient clientRecept;


//on prepare un timer
unsigned long previousMillis1s = 0;
unsigned long previousMillis5s = 0;
unsigned long previousMillis1min = 0;

const unsigned long interval1s = 1000;
const unsigned long interval5s = 5000;
const unsigned long interval1min = 60000;


// Variables de simulation
const char* stockageTour = "1";
const char* IndexCouleur = "1";


// numérisation des valeurs :
int Game = 0; //0 = ON et 1 = OFF
float Progress = 0.0; //Progression dans la partie en pourcentage
int collision = 0; //nombre de collision actuel
int Couleur = 1; // Couleur de l'appareil : (1=vert, 2=jaune, 3=noir, 4=bleu)
int Status = 1; //Status pour la supervision
int Tom = 0;
int stop = 0;
const char* ipToM = "192.168.0.48";
float points = 10.0;
int nbrTOM = 10;

void connectWiFi() {
  Serial.print("Connexion au WiFi");
  WiFi.begin(ssid, password);
  unsigned long startAttemptTime = millis();


  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 10000) {
    delay(500);
    Serial.print(".");
  }


  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConnecté au WiFi");
    Serial.print("Adresse IP locale : ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\nÉchec de connexion WiFi.");
  }
}
```

```
void establishTCPConnection() {
  Serial.println("Connexion au serveur TCP...");
  if (clientRecept.connect(tcpAddress, tcpPort)) {
    Serial.println("Connexion TCP réussie.");
    sendTCPMessageToServer();  // Envoi initial après connexion
  } else {
    Serial.println("Échec de connexion TCP.");
  }
}


void setup() {
  Serial.begin(115200);
  connectWiFi();
  establishTCPConnection();
}


void loop() {
  unsigned long currentMillis = millis();


  if (!clientRecept.connected()) {
    Serial.println("Connexion perdue, tentative de reconnexion...");
    if (clientRecept.connect(tcpAddress, tcpPort)) {
      Serial.println("Reconnecté au serveur.");
      sendTCPMessageToServer();  // Ré-envoi des infos à la reconnexion
    } else {
      Serial.println("Échec de la reconnexion.");
      delay(1000);
      return;
    }
  }

  if (clientRecept.available()) {
    listenOrder();
  }

  if (currentMillis - previousMillis1min >= interval1min) {
    previousMillis1min = currentMillis;
    sendTCPMessageToServer();
  }

  if (currentMillis - previousMillis1s >= interval1s) {
    previousMillis1s = currentMillis;
    functionEvery1Seconds();
  }
}


void functionEvery1Seconds() {
  Simulation();
}


void listenOrder() {
  Serial.println("début de l'écoute");
  String data = clientRecept.readStringUntil('\n');
  if (data.length() == 0) return;


  Serial.print("Trame reçue : ");
  Serial.println(data);


  StaticJsonDocument<512> doc;
  DeserializationError error = deserializeJson(doc, data);
```

```
  if (error) {
    Serial.print("Erreur JSON : ");
    Serial.println(error.f_str());
    return;
  }


  if (!doc.containsKey("Ordre")) {
    Serial.println("Trame invalide (pas de clé 'Ordre').");
    return;
  }


  int ordre = doc["Ordre"];
  Serial.print("Ordre reçu : ");
  Serial.println(ordre);


  switch (ordre) {
    case 1:
      Serial.println("Ordre 1 : Envoi des infos");
      sendTCPMessageToServer();
      break;
    case 2:
      Serial.println("Ordre 2 : partie redémarrée");
      //remise a zero de la progression et des collision
      nbrTOM = doc["nbrTOM"];
      Progress = 0.0;
      collision = 0;
      Status = 1;
      Game = 0;
      Serial.print("Nombre de tour : ");
      Serial.println(nbrTOM);
      calcul();
      sendTCPMessageToServer();
      break;
    case 3:
      Serial.println("Ordre 3 : TOM pleine, vidage !");
      stop = 0;
      Tom = 1;
      break;
    case 5:
      Serial.println("Ordre 5 : TOM déjà vide");
      Tom = 0;
      stop = 1;
      break;
    case 6:
      Serial.println("Ordre 6");
      Game = 1;
      break;
    case 7:
      Serial.println("Ordre 7");
      Game = 0;
      break;
    default:
      Serial.println("Ordre inconnu.");
      break;
  }
}


void sendTCPMessageToServer() {
  if (!clientRecept.connected()) {
    Serial.println("Connexion perdue pour envoi, tentative de reconnexion...");
    if (!clientRecept.connect(tcpAddress, tcpPort)) {
      Serial.println("Échec de reconnexion pour envoi.");
      return;
    }
  }
```

```
  DynamicJsonDocument doc(256);
  doc["Type"] = "BOM";
  doc["Couleur"] = IndexCouleur;
  doc["Progression"] = String(Progress);
  doc["NbrCollision"] = String(collision);
  doc["Status"] = String(Status);

  serializeJson(doc, clientRecept);
  clientRecept.println();

  Serial.print("Infos envoyées : ");
  serializeJson(doc, Serial);
  Serial.println();
}


void CheckTour() {
  stop = 0;
  int couleurTOM = 1;
  if (Couleur != couleurTOM) {
    Serial.println("Pas la bonne couleur !");
  }
  if (Couleur == couleurTOM) {
    Serial.println("la bonne couleur");
    Status = 2;
    //envoie au web de l'info
    DynamicJsonDocument doc(256);
    doc["Type"] = "BOM";
    doc["Couleur"] = String(Couleur);
    doc["Status"] = String(Status);
    doc["ipTOM"] = String(ipToM);

    serializeJson(doc, clientRecept);
    clientRecept.println();

    Serial.print("Infos envoyées : ");
    serializeJson(doc, Serial);
    Serial.println();
    Serial.println("verrification de la tour");
    listenOrder();
    Serial.println(stop);

    if (stop == 1){
      Serial.println("Tour vide !");
      Status = 1;
      return;
    }
    else {
      Serial.println("Tour pleine !");
      Tom = 1;
    }

    if (Tom == 1){
      Serial.println("TOM vidée !");
      Progress = Progress + points;
      Status = 3;

      DynamicJsonDocument doc(256);

      doc["Type"] = "BOM";
      doc["Couleur"] = String(Couleur);
      doc["Progression"] = String(Progress);
      doc["NbrCollision"] = String(collision);
      doc["Status"] = String(Status);
      doc["ipTOM"] = String(ipToM);
      serializeJson(doc, clientRecept);
      clientRecept.println();

      Serial.print("Infos envoyées : ");
```

```
      serializeJson(doc, Serial);
      Serial.println();
      Status = 1;
    }
    if (Tom == 0) {
      Serial.println("Le joueur n'est plus devant la tour !");
      Status = 4;

      DynamicJsonDocument doc(256);

      doc["Type"] = "BOM";
      doc["Couleur"] = String(Couleur);
      doc["Status"] = String(Status);
      doc["ipTOM"] = String(ipToM);

      serializeJson(doc, clientRecept);
      clientRecept.println();

      Serial.print("Infos envoyées : ");
      serializeJson(doc, Serial);
      Serial.println();
      Status = 1;
    }
  }
}
void Gethit() {
  int trigger = 1;
  if (trigger == 1) {
    Serial.println("Collision subit !");
    collision = collision + 1;
    Status = 5;
    DynamicJsonDocument doc(256);
    doc["Type"] = "BOM";
    doc["Couleur"] = String(Couleur);
    doc["Progression"] = String(Progress);
    doc["NbrCollision"] = String(collision);
    doc["Status"] = String(Status);

    serializeJson(doc, clientRecept);
    clientRecept.println();

    Serial.print("Infos envoyées : ");
    serializeJson(doc, Serial);
    Serial.println();
    Status = 1;
  }
  else {
    Serial.println("aucune collision subit.");
  }
}
void Simulation() {
  if (Game == 1){
    Serial.println("La partie est soit en pause soit terminé !");
    return;
  }

  if (Game == 0){
      if (Serial.available() > 0) {
      char commande = Serial.read();
      switch (commande) {
        case '1':
          Gethit();
          break;
        case 'a':
          Serial.println("Tour numéro 1");
          ipToM = "192.168.0.39";
          CheckTour();
          break;
        case 'z':
          Serial.println("Tour numéro 2");
```

```
          ipToM = "192.168.0.40";
          CheckTour();
          break;
        case 'e':
          Serial.println("Tour numéro 3");
          ipToM = "192.168.0.41";
          CheckTour();
          break;
        case 'r':
          Serial.println("Tour numéro 4");
          ipToM = "192.168.0.42";
          CheckTour();
          break;
        case 't':
          Serial.println("Tour numéro 5");
          ipToM = "192.168.0.43";
          CheckTour();
          break;
        case 'y':
          Serial.println("Tour numéro 6");
          ipToM = "192.168.0.44";
          CheckTour();
          break;
        case 'u':
          Serial.println("Tour numéro 7");
          ipToM = "192.168.0.45";
          CheckTour();
          break;
        case 'i':
          Serial.println("Tour numéro 8");
          ipToM = "192.168.0.46";
          CheckTour();
          break;
        case 'o':
          Serial.println("Tour numéro 9");
          ipToM = "192.168.0.47";
          CheckTour();
          break;
        case 'p':
          Serial.println("Tour numéro 10");
          ipToM = "192.168.0.37";
          CheckTour();
          break;
      }
    }
  }
}
// Fonctions système
void remiseAZero() {
  collision = 0;
  Progress = 0;
  Serial.println("Remise à zéro");
}
void pausePartie() {
  Serial.println("Partie en pause");
}
void reprise() {
  Serial.println("Partie reprise");
}
void calcul() {
  Serial.println(nbrTOM);
  points = (float)100/(nbrTOM);
  Serial.println(points);
  if (points >= 100) {
    points = 100;
  }
  Serial.println("Dans cette partie, les tours rapportent : ");
  Serial.println(points);
}
```

Annexes Guilhem BUSSI - IR 2

Annexe 1 - Script actuel des TOM:

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>

// WiFi
const char* ssid = "AP-ESTOM";
const char* password = "Projet-ESTOM_2025";

// Adresse IP et port du serveur TCP
const char* tcpAddress = "192.168.0.10";
const uint16_t tcpPort = 5005;

// Client TCP unique (réception + envoi)
WiFiClient clientRecept;

// Timers
unsigned long previousMillis1min = 0;
const unsigned long interval1min = 60000;

// Variables de simulation
const char* stockageTour = "1";
const char* IndexCouleur = "1";
const char* EtatLED = "1";

void connectWiFi() {
  Serial.print("Connexion au WiFi");
  WiFi.begin(ssid, password);
  unsigned long startAttemptTime = millis();

  // Attente initiale de 5 secondes
  while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < 5000) {
    delay(500);
    Serial.print(".");
  }

  // Si non connecté après 5 secondes, entrer dans une boucle de reconnexion
  while (WiFi.status() != WL_CONNECTED) {
    Serial.println("\nÉchec de connexion WiFi. Nouvelle tentative...");
    WiFi.disconnect();
    delay(1000);
    WiFi.begin(ssid, password);

    unsigned long retryStartTime = millis();
    while (WiFi.status() != WL_CONNECTED && millis() - retryStartTime < 5000) {
      delay(500);
      Serial.print(".");
    }
  }
```

```
  Serial.println("\nConnecté au WiFi");
  Serial.print("Adresse IP locale : ");
  Serial.println(WiFi.localIP());
}

void establishTCPConnection() {
  Serial.println("Connexion au serveur TCP...");
  if (clientRecept.connect(tcpAddress, tcpPort)) {
    Serial.println("Connexion TCP réussie.");
    sendTCPMessageToServer();  // Envoi initial après connexion
  } else {
    Serial.println("Échec de connexion TCP.");
  }
}

void setup() {
  Serial.begin(115200);
  connectWiFi();
  establishTCPConnection();
}

void loop() {
  unsigned long currentMillis = millis();

  if (!clientRecept.connected()) {
    Serial.println("Connexion perdue, tentative de reconnexion...");
    if (clientRecept.connect(tcpAddress, tcpPort)) {
      Serial.println("Reconnecté au serveur.");
      sendTCPMessageToServer();  // Ré-envoi des infos à la reconnexion
    } else {
      Serial.println("Échec de la reconnexion.");
      delay(1000);
      return;
    }
  }

  if (clientRecept.available()) {
    handleIncomingOrder();
  }

  if (currentMillis - previousMillis1min >= interval1min) {
    previousMillis1min = currentMillis;
    sendTCPMessageToServer();
  }
}

void handleIncomingOrder() {
  String data = clientRecept.readStringUntil('\n');
  if (data.length() == 0) return;

  Serial.print("Trame reçue : ");
```

```
  Serial.println(data);

  StaticJsonDocument<512> doc;
  DeserializationError error = deserializeJson(doc, data);

  if (error) {
    Serial.print("Erreur JSON : ");
    Serial.println(error.f_str());
    return;
  }

  if (!doc.containsKey("Ordre")) {
    Serial.println("Trame invalide (pas de clé 'Ordre').");
    return;
  }

  int ordre = doc["Ordre"];
  Serial.print("Ordre reçu : ");
  Serial.println(ordre);

  switch (ordre) {
    case 1:
      Serial.println("Ordre 1 : Envoi des infos");
      sendTCPMessageToServer();
      break;
    case 2:
      remiseAZero();
      sendTCPMessageToServer();
      break;
    case 3:
      clignotementLED();
      break;
    case 4:
      eteintLED();
      sendTCPMessageToServer();
      break;
    case 6:
      pausePartie();
      sendTCPMessageToServer();
      break;
    case 7:
      reprise();
      sendTCPMessageToServer();
      break;
    default:
      Serial.println("Ordre inconnu.");
      break;
  }
}

void sendTCPMessageToServer() {
  if (!clientRecept.connected()) {
```

```
    Serial.println("Connexion perdue pour envoi, tentative de reconnexion...");
    if (!clientRecept.connect(tcpAddress, tcpPort)) {
      Serial.println("Échec de reconnexion pour envoi.");
      return;
    }
  }

  DynamicJsonDocument doc(256);
  doc["Type"] = "TOM";
  doc["Couleur"] = IndexCouleur;
  doc["Stockage"] = stockageTour;

  serializeJson(doc, clientRecept);
  clientRecept.println();

  Serial.print("Infos envoyées : ");
  serializeJson(doc, Serial);
  Serial.println();
}

// Fonctions système
void remiseAZero() {
  stockageTour = "1";
  EtatLED = "1";
  Serial.println("Remise à zéro (LED allumée)");
}

void clignotementLED() {
  EtatLED = "2";
  Serial.println("LED clignotante");
}

void eteintLED() {
  stockageTour = "0";
  EtatLED = "0";
  Serial.println("LED éteinte");
}

void pausePartie() {
  Serial.println("Partie en pause");
}

void reprise() {
  Serial.println("Partie reprise");
}
```

Annexes Clément VOLLE - IR 3

Annexe 1 - Fichier : gestion-de-la-partie.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ESTOM</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="icon" type="image/x-icon" href="/img/estom-logo.ico">
</head>
<body>
    <header>
        <h1 class="title">ESTOM</h1>
        <nav class="navbar">
            <ul class="menu">
            <a class="btn " id="btn1" href="./gestion-de-la-partie.php">Gestion de la partie</a>
            <a class="btn " id="btn2" href="./parametre-du-jeu.php">Paramétrage du Jeu</a>
            </ul>
        </nav>
        <div class="logo" id="logo">
            <a href="./accueil.php">
                <img src="./img/estom-logo.png" alt="" id="estom-logo">
            </a>
        </div>
    </header>
    <main>
        <section>
            <h2>Gestion de la partie</h2>
            <div id="timer" class="timer">Cliquez sur le bouton pour lancer la Partie !</div>

            <div id="TableauGestion" class="GTableauGestion">
                <!-- Bouton pour ouvrir la fenêtre de saisie des joueurs -->
                <button id="openModalButton">Ajouter Joueurs</button>

                <!-- Fenêtre modale Joueur -->
                <div id="playerModal" class="modal">
                    <div class="modal-content">
                        <span class="close-btn">&times;</span>
                        <h2>Ajoutez vos joueurs</h2>
                        <form id="playerForm">
                            <div id="playerInputs">
                                <?php
                                include '/var/www/html/private-estom/db_connect.php';

                                // Récupération du nombre de BOM configurées
                                $sql = "SELECT NbrBOM_V, NbrBOM_J, NbrBOM_N, NbrBOM_B FROM Config WHERE
Id = 1";
                                $result = $conn->query($sql);
                                $nbrBOMConfig = 0;

                                if ($result && $result->num_rows > 0) {
                                    $row = $result->fetch_assoc();
                                    $nbrBOMConfig = $row['NbrBOM_V'] + $row['NbrBOM_J'] +
$row['NbrBOM_N'] + $row['NbrBOM_B'];
                                }

                                // Récupération du nombre de BOM enregistrées
                                $sql = "SELECT COUNT(*) AS totalBOM FROM BOM";
                                $result = $conn->query($sql);
                                $nbrBOMEnregistrees = 0;

                                if ($result && $result->num_rows > 0) {
                                    $row = $result->fetch_assoc();
                                    $nbrBOMEnregistrees = $row['totalBOM'];
                                }
                                $colorMap = [
                                    1=> "Vert",
                                    2=> "Jaune",
                                    3=> "Noir",
                                    4=> "Bleu"
                                ];
```

```
                                // Récupération des BOM avec leur ID, couleur et joueur
                                $sql = "SELECT ID, Couleur, Joueur FROM BOM ORDER BY Couleur, ID";
                                $result = $conn->query($sql);

                                // Conversion explicite en entiers
                                $nbrBOMConfig2 = (int) $nbrBOMConfig;
                                $nbrBOMEnregistrees2 = (int) $nbrBOMEnregistrees;

                                if ($result && $result->num_rows > 0) {
                                    while ($row = $result->fetch_assoc()) {
                                        $bomID = $row['ID'];
                                        $couleur = $colorMap[$row['Couleur']];
                                        $joueur = $row['Joueur'] ?? "";

                                        echo "<div class='player-input'>
                                                <span>BOM#$bomID - $couleur</span>
                                                <input type='text' name='joueurs[$bomID]'
value='$joueur' maxlength='20' placeholder='Pseudo joueur' required>
                                            </div>";
                                    }
                                }
                                // Affichage du message d'alerte si le nombre de BOM est incorrect
                                if ($nbrBOMConfig2 !== $nbrBOMEnregistrees2) {
                                    echo "<p style='color: red; font-weight: bold; margin-top: 10px;'>
                                            ⚠ Attention : La configuration prévoit
<strong>$nbrBOMConfig2 BOM</strong>,
                                            mais <strong>$nbrBOMEnregistrees2 BOM</strong> sont
enregistrées !
                                        </p>";
                                }
                                ?>
                            </div>
                            <button type="submit">Valider</button>
                        </form>
                    </div>
                </div>

                <?php
                // Inclusion du fichier de connexion
                include '/var/www/html/private-estom/db_connect.php';

                // Récupérer la durée du timer (en secondes) depuis la base de données
                $timerDuration = 180; // Valeur par défaut : 3 minutes (180 secondes)
                $sql = "SELECT duree FROM Config WHERE id = 1"; // Exemple de table de config
                $result = $conn->query($sql);

                if ($result && $result->num_rows > 0) {
                    $row = $result->fetch_assoc();
                    $timerDuration = (int)$row['duree'];
                }

                // Injecter la durée du timer dans le script JS
                echo "<script>const timerDuration = $timerDuration;</script>";

                // Récupération des paramètres existants
                $sql = "SELECT Options, PtsRecolte, NbrTOM, MalusCollision, NbrBOM_V, NbrBOM_J,
NbrBOM_N, NbrBOM_B, duree FROM Config WHERE Id = 1";
                $result = $conn->query($sql);

                if ($result && $result->num_rows > 0) {
                    $row = $result->fetch_assoc();
                    $options = $row['Options'];
                    $ptsRecolte = $row['PtsRecolte'];
                    $nbrTOM = $row['NbrTOM'];
                    $malusCollision = $row['MalusCollision'];
                    $nbrBOM_V = $row['NbrBOM_V'];
                    $nbrBOM_J = $row['NbrBOM_J'];
                    $nbrBOM_N = $row['NbrBOM_N'];
                    $nbrBOM_B = $row['NbrBOM_B'];
                    $tempsPartie = $row['duree'];
                }
                echo "<script>
                            const malusCollision = $malusCollision;
                            const ptsRecolte = $ptsRecolte;
```

```
                        const optionMode = $options;
                        const nbrTOM = $nbrTOM;
                    </script>";

                $colorMap = [
                    "Vert" => 1,
                    "Jaune" => 2,
                    "Noir" => 3,
                    "Bleu" => 4
                ];

                ?>


                <button id="startGameButton">Lancer la partie</button>
                <button id="resetGameButton" style="display:none;">Réinitialiser la partie</button>
            </div> <!-- in div "TableauGestion" -->


            <!-- Modal "podiumModal" -->
<!--             <div id="podiumModal" class="modal" style="display:none;">
                <div class="modal-content">
                    <div class="img">
                        <img src="img/podium.png" alt="Podium" style="width:100%;">
                    </div>
                    <div id="classementContent">
                    </div>
                </div>
            </div> -->

            <div id="tableauBars">
                <?php
                $couleurs = [
                    "Vert" => $nbrBOM_V,
                    "Jaune" => $nbrBOM_J,
                    "Noir" => $nbrBOM_N,
                    "Bleu" => $nbrBOM_B
                ];
                $rgbaColors = [ //rgb correspondant aux couleurs BOM & TOM
                    "Vert" => "rgba(14, 202, 26, 0.6)",
                    "Jaune" => "rgba(250, 206, 0, 0.6)",
                    "Noir" => "rgba(59, 59, 59, 0.6)",
                    "Bleu" => "rgba(23, 140, 221, 0.6)"
                ];

                foreach ($couleurs as $couleur => $nombreBOM) {
                    // Récupérer les IP des BOM de cette couleur
                    $sql = "SELECT IPAddr FROM BOM WHERE Couleur = ?";
                    $stmt = $conn->prepare($sql);
                    $stmt->bind_param("i", $colorMap[$couleur]);
                    $stmt->execute();
                    $result = $stmt->get_result();

                    $ipList = [];
                    while ($row = $result->fetch_assoc()) {
                        $ipList[] = $row['IPAddr'];
                    }
                    $stmt->close();

                    for ($i = 0; $i < $nombreBOM; $i++) {
                        $ip = $ipList[$i] ?? "NULL"; // Assigne une IP si disponible, sinon NULL

                        echo "<div class='route-container' data-couleur='$couleur' data-ip='$ip'>";

                        // Scoreboard à gauche du parcours
                        echo "<div class='scoreboard-container'>
                        <img src='./img/CadreScore-$couleur.png' class='scoreboard'
data-couleur='$couleur' data-ip='$ip'>
                        <span class='score-text' id='score-$ip'>0</span>
                        </div>";

                        echo "<img src='./img/route.png' alt='Route' class='imageRoute'>";

                        // BOM avec IP ou sans IP
```

```
                        echo "<img src='./img/BOM-$couleur.png' alt='Camion $couleur'
                            class='imageCamion' data-couleur='$couleur' data-ip='$ip'
                            style='position: absolute; left: 0%; top: 50%; width: 60px; opacity: 0.9;
z-index: 3;'>";
                            echo "<span class='pseudo' data-ip='$ip'
                            style='
                                display:none;
                                background-color: {$rgbaColors[$couleur]};
                            '></span>";

                        echo "<img src='./img/LigneArrive.gif' alt='Ligne d'arrivée'
class='imageArrive'>";

                        for ($j = 0; $j < $nbrTOM; $j++) {
                            $espacement = 100 / ($nbrTOM + 1);
                            $positionLeft = ($j + 1) * $espacement;

                            echo "<img src='./img/TOM-$couleur.png' alt='Tour $couleur'
                                class='imageTour' data-couleur='$couleur' data-ip='$ip' data-index='$j'
                                style='
                                    position: absolute;
                                    top: 50%;
                                    left: {$positionLeft}%;
                                    transform: translate(-50%, -50%);
                                    width: 20px;
                                    opacity: 1;
                                    z-index: 4;
                                '>";
                        }
                        echo "</div>";
                    }
                }
                ?>
            </div>
        </section>
    </main>
    <script>
        document.addEventListener("DOMContentLoaded", function () {
            const modal = document.getElementById("playerModal");
            const openModalButton = document.getElementById("openModalButton");
            const closeModalButton = document.querySelector(".close-btn");
            const playerForm = document.getElementById("playerForm");

            openModalButton.addEventListener("click", function () {
                modal.style.display = "flex";
            });

            closeModalButton.addEventListener("click", function () {
                modal.style.display = "none";
            });

            playerForm.addEventListener("submit", function (event) {
                event.preventDefault();

                const formData = new FormData(playerForm);
                fetch('update_joueurs.php', {
                    method: 'POST',
                    body: formData
                })
                .then(response => response.text())
                .then(() => {
                    modal.style.display = "none";
                    location.reload();
                })
                .catch(error => console.error("Erreur :", error));
            });
        });
    </script>
    <script>
        function showPodium() {
            document.getElementById("podiumModal").style.display = "block";
        }
        function hidePodium() {
            document.getElementById("podiumModal").style.display = "none";
```

```
        }
        // Fermeture quand on clique en dehors de la zone modale (hors .modal-content)
        document.getElementById("podiumModal").addEventListener("click", function (event) {
            const modalContent = this.querySelector(".modal-content");
            if (modalContent && !modalContent.contains(event.target)) {
                hidePodium();
            }
        });

        // Fermeture avec la touche Échap
        document.addEventListener("keydown", function (event) {
            if (event.key === "Escape") {
                hidePodium();
            }
        });
    </script>
    <script src="js/gestion-de-la-partie/updateCamion.js"></script>
    <script src="js/gestion-de-la-partie/updateTimer.js"></script>
    <footer>
        <p>&copy; 2025 ESTOM</p>
    </footer>
</body>
</html>
```

Annexe 2 - Fichier : parametre-du-jeu.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ESTOM</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="icon" type="image/x-icon" href="/img/estom-logo.ico">
</head>
<body>
    <header>
        <h1 class="title">ESTOM</h1>
        <nav class="navbar">
            <ul class="menu">
            <a class="btn " id="btn1" href="./gestion-de-la-partie.php">Gestion de la partie</a>
            <a class="btn " id="btn2" href="./parametre-du-jeu.php">Paramétrage du Jeu</a>
            </ul>
        </nav>
        <div class="logo" id="logo">
            <a href="./accueil.php">
                <img src="./img/estom-logo.png" alt="" id="estom-logo">
            </a>
        </div>
    </header>
    <main>
        <section>
            <h2>Paramétrage du Jeu</h2>

            <?php
            // Inclusion du fichier de connexion
            include '/var/www/html/private-estom/db_connect.php';

            // Initialisation des paramètres
            $tempsPartie = 180;
            $options = 1;
            $ptsRecolte = 10;
            $nbrTOM = 10;
            $malusCollision = 2;
            $nbrBOM_V = 1;
            $nbrBOM_J = 1;
            $nbrBOM_N = 1;
            $nbrBOM_B = 1;
            // Récupération des paramètres existants
            $sql = "SELECT Options, PtsRecolte, NbrTOM, MalusCollision, NbrBOM_V, NbrBOM_J, NbrBOM_N,
NbrBOM_B, duree FROM Config WHERE Id = 1";
            $result = $conn->query($sql);
```

```
            if ($result && $result->num_rows > 0) {
                $row = $result->fetch_assoc();
                $options = $row['Options'];
                $ptsRecolte = $row['PtsRecolte'];
                $nbrTOM = $row['NbrTOM'];
                $malusCollision = $row['MalusCollision'];
                $nbrBOM_V = $row['NbrBOM_V'];
                $nbrBOM_J = $row['NbrBOM_J'];
                $nbrBOM_N = $row['NbrBOM_N'];
                $nbrBOM_B = $row['NbrBOM_B'];
                $tempsPartie = $row['duree'];
            }

            // Si le formulaire est soumis
            if ($_SERVER['REQUEST_METHOD'] === 'POST') {
                // Mise à jour du temps de la partie
                // Validation des entrées utilisateur avec double vérification des valeurs!
                $nouveauTempsPartie = filter_var($_POST['temps_partie'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 30, 'max_range' => 600]
                ]) ?? $tempsPartie;

                $nouveauNbrTOM = filter_var($_POST['nombre_tours'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 5, 'max_range' => 10]
                ]) ?? $nbrTOM;
                $nouveauMalusCollision = filter_var($_POST['malus_collision'], FILTER_VALIDATE_FLOAT);
                if ($nouveauMalusCollision === false || $nouveauMalusCollision < 0.2 ||
$nouveauMalusCollision > 5) {
                    $nouveauMalusCollision = $malusCollision;
                }


                $nouveauNbrBOM_V = filter_var($_POST['nombre_bom_v'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 1, 'max_range' => 2]
                ]) ?? $nbrBOM_V;
                $nouveauNbrBOM_J = filter_var($_POST['nombre_bom_j'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 1, 'max_range' => 2]
                ]) ?? $nbrBOM_J;
                $nouveauNbrBOM_N = filter_var($_POST['nombre_bom_n'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 1, 'max_range' => 2]
                ]) ?? $nbrBOM_N;
                $nouveauNbrBOM_B = filter_var($_POST['nombre_bom_b'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 1, 'max_range' => 2]
                ]) ?? $nbrBOM_B;

                $nouveauPtsRecolte = filter_var($_POST['pts_recolte'], FILTER_VALIDATE_INT, [
                    'options' => ['min_range' => 1, 'max_range' => 100]
                ]) ?? $ptsRecolte;

                //Option, si Option = 0 alors Option A, si Option = 1 Alors option B !
                if (isset($_POST['regle']) && $_POST['regle'] === 'option_a') {
                    $nouveauOptions = 0;
                } else {
                    $nouveauOptions = 1;
                }

                // Mise à jour dans la base de données
                $sqlUpdate = "UPDATE Config SET Options = ?, PtsRecolte = ?, NbrTOM = ?, MalusCollision
= ?, NbrBOM_V = ?, NbrBOM_J = ?, NbrBOM_N = ?, NbrBOM_B = ?, duree = ? WHERE Id = 1";
                $stmt = $conn->prepare($sqlUpdate);
                $stmt->bind_param("iiidiiiii", $nouveauOptions, $nouveauPtsRecolte, $nouveauNbrTOM,
$nouveauMalusCollision, $nouveauNbrBOM_V, $nouveauNbrBOM_J, $nouveauNbrBOM_N, $nouveauNbrBOM_B,
$nouveauTempsPartie);
                $stmt->execute();

                header("Location: parametre-du-jeu.php?success=1");
                exit;
            }

            // Fermeture de la connexion
            $conn->close();
            ?>

            <div id="Principale">
```

```
                <form method="post" action="parametre-du-jeu.php" id="BoiteRegles">

                    <fieldset id="fieldset">
                        <legend id="titre">Règles du Jeu</legend>

                        <div class="Regle-Container">

                            <label for="temps_partie" id="temps_partie">Temps de la partie (en
secondes) :</label>
                            <input type="number" id="temps_partie_input" name="temps_partie"
value="<?php echo htmlspecialchars($tempsPartie); ?>" step="10" min="30" max="600" required>

                            <label for="nombre_tours" id="nombre_tours">Nombre de TOM par couleur
:</label>
                            <input type="number" id="nombre_tours_input" name="nombre_tours"
value="<?php echo htmlspecialchars($nbrTOM); ?>" min="5" max="10" required>

                            <label for="malus_collision" id="malus_collision">Malus de collision
:</label>
                            <input type="number" id="malus_collision_input" name="malus_collision"
value="<?php echo htmlspecialchars($malusCollision); ?>" step="0.2" min="0.2" max="5" required>

                            <label id="optionA_label" for="optionA">
                                <input type="radio" name="regle" id="optionA" value="option_a" <?php
echo $options == 0 ? 'checked' : ''; ?>>
                                <span id="optionA_text">Option A : Points identiques pour chaque
récolte</span>
                            </label>
                            <div id="optionA_settings">
                                <label for="pts_recolte">Points par TOM récolté :</label>
                                <input type="number" id="pts_recolte_input" name="pts_recolte"
value="<?php echo htmlspecialchars($ptsRecolte); ?>" min="1" max="100">
                            </div>

                            <label  id="optionB_label" for="optionB">
                                <input type="radio" name="regle" id="optionB" value="option_b" <?php
echo $options == 1 ? 'checked' : ''; ?>>
                                <span id="optionB_text">Option B : Points progressifs selon
l'avancement de la récolte</span>
                            </label>

                            <div id="tableau_Nombres">
                                <label for="nombre_bom_v" id="nombre_vert">BOM Verte:</label>
                                <input type="number" id="nombre_vert_input" name="nombre_bom_v"
value="<?php echo htmlspecialchars($nbrBOM_V); ?>" min="0" max="2" required>

                                <label for="nombre_bom_j" id="nombre_jaune">BOM Jaune:</label>
                                <input type="number" id="nombre_jaune_input" name="nombre_bom_j"
value="<?php echo htmlspecialchars($nbrBOM_J); ?>" min="0" max="2" required>

                                <label for="nombre_bom_n" id="nombre_noir">BOM Noir:</label>
                                <input type="number" id="nombre_noir_input" name="nombre_bom_n"
value="<?php echo htmlspecialchars($nbrBOM_N); ?>" min="0" max="2" required>

                                <label for="nombre_bom_b" id="nombre_bleu">BOM Bleu:</label>
                                <input type="number" id="nombre_bleu_input" name="nombre_bom_b"
value="<?php echo htmlspecialchars($nbrBOM_B); ?>" min="0" max="2" required>
                            </div>
                        </div>
                    </fieldset>

                    <button type="submit">Sauvegarder les paramètres</button>
                </form>

                <div id="IconesParametre">
                    <?php
                    $couleurs = [
                        "Vert" => $nbrBOM_V,
                        "Jaune" => $nbrBOM_J,
                        "Noir" => $nbrBOM_N,
                        "Bleu" => $nbrBOM_B
                    ];
                    $maxBOM = max($nbrBOM_V, $nbrBOM_J, $nbrBOM_N, $nbrBOM_B);
                    $position = 1; // Position de chaque colonne dans la grille
```

```
                    foreach ($couleurs as $couleur => $nombre) {
                        echo '<div id="ColoneBOM' . $couleur . '" style="grid-column: ' . $position .
'; grid-row: 1;">';

                        // Afficher 1 ou 2 BOM par couleur
                        for ($i = 0; $i < $maxBOM; $i++) {
                            echo '<div id="BOM-' . $couleur . '-' . $i . '" class="bom-container">
                                    <img src="./img/BOM-' . $couleur . '.png" alt="Camion ' . $couleur
. '" id="BOM-' . $couleur . '-' . $i . '" class="grise" width="75" height="45">
                                    <span id="BOM-IP-' . $couleur . '-' . $i . '"
class="ip-text"></span>
                                </div>';
                        }
                        echo '</div>';
                        echo '<div id="ColoneTOM' . $couleur . '" style="grid-column: ' . $position .
'; grid-row: 2;">';
                        for ($i = 0; $i < 10; $i++) {
                            echo '<div class="LigneTOM">
                                    <img src="./img/TOM-' . $couleur . '.png" alt="TOM ' . $couleur .
'" id="TOM-' . $couleur . '-' . $i . '" class="grise imageTOM" width="29" height="45">
                                    <span id="TOM-IP-' . $couleur . '-' . $i . '"
class="ip-text"></span>
                                </div>';
                        }
                        echo '</div>';
                        $position++; // Passe à la colonne suivante
                    }
                    ?>
                </div>

            </div>
        </section>
    </main>
    <footer>
        <p>&copy; 2025 ESTOM</p>
    </footer>
</body>
    <script src="js/parametre-du-jeu/updateConnexion.js"></script>
    <script>
        // Fonction pour gérer la visibilité du champ "Pts par Tour"
        function togglePointsField() {
            const optionAInput = document.querySelector('input[name="regle"][value="option_a"]');
            const pointsField = document.getElementById("optionA_settings");

            // Afficher ou masquer le champ selon l'option sélectionnée
            if (optionAInput.checked) {
                pointsField.style.display = "block";
            } else {
                pointsField.style.display = "none";
            }
        }

        // Ajout d'événement
        document.addEventListener("DOMContentLoaded", () => {
            // Exécuter une première fois pour régler l'affichage initial
            togglePointsField();

            // Ajouter des "écouteurs" sur les options de règle
            document.querySelectorAll('input[name="regle"]').forEach(input => {
                input.addEventListener("change", togglePointsField);
            });
        });
    </script>
</html>
```

Annexe 3 - Fichier : update_status.php

```php
<?php
include '/var/www/html/private-estom/db_connect.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['status'])) {
    $status = $conn->real_escape_string($_POST['status']);

    // Conversion du statut en valeur numérique
    switch ($status) {
        case 'stopped':
            $statusValue = 0;
            break;
        case 'running':
            $statusValue = 1;
            break;
        case 'paused':
            $statusValue = 2;
            break;
        case 'loading':
            $statusValue = 3;
            break;
        default:
            echo "Statut invalide.";
            $conn->close();
            exit();
    }

    $updateSql = "";

    if ($_POST['status'] === 'running' && isset($_POST['temps_restant'])) {
        $temps_restant = (int)$_POST['temps_restant'];

        // ➤ Récupérer la durée totale
        $result = $conn->query("SELECT Duree FROM Config WHERE id = 1");
        if ($result && $row = $result->fetch_assoc()) {
            $duree = (int)$row['Duree'];
            $temps_ecoule = $duree - $temps_restant;
            $nouvelle_heure_depart = date('Y-m-d H:i:s', time() - $temps_ecoule);

            $stmt = $conn->prepare("UPDATE Config SET Status = 1, Heure_depart = ?, Temps_restant =
NULL WHERE id = 1");
            $stmt->bind_param("s", $nouvelle_heure_depart);
            $stmt->execute();
            $stmt->close();
            echo 'Reprise réussie';
            exit;
        } else {
            echo "Erreur lors de la récupération de la durée.";
            $conn->close();
            exit;
        }
    }

    if ($status === 'running') {
        // Vérifie si on a déjà une heure_depart
        $result = $conn->query("SELECT Heure_depart, Temps_restant FROM Config WHERE id = 1");
        if ($result && $row = $result->fetch_assoc()) {
            if (empty($row['Heure_depart']) || $row['Heure_depart'] === '0000-00-00 00:00:00') {
                // On redémarre à partir de Temps_restant (s'il y en a), sinon durée complète
                $now = date("Y-m-d H:i:s");
                $updateSql = "UPDATE Config SET Status = $statusValue, Heure_depart = '$now',
Temps_restant = NULL WHERE id = 1";
            } else {
                // Ne pas réinitialiser heure_depart
                $updateSql = "UPDATE Config SET Status = $statusValue WHERE id = 1";
            }
        } else {
            echo "Erreur lors de la récupération des données.";
            $conn->close();
            exit();
        }
```

```
    } elseif ($status === 'paused') {
        // Calcule le temps restant
        $result = $conn->query("SELECT Heure_depart, Duree FROM Config WHERE id = 1");
        if ($result && $row = $result->fetch_assoc()) {
            $heureDepart = strtotime($row['Heure_depart']);
            $duree = (int)$row['Duree'];
            $tempsEcoule = time() - $heureDepart;
            $tempsRestant = max(0, $duree - $tempsEcoule);

            $updateSql = "UPDATE Config SET Status = $statusValue, Temps_restant = $tempsRestant,
Heure_depart = NULL WHERE id = 1";
        } else {
            echo "Erreur lors de la récupération des données.";
            $conn->close();
            exit();
        }

    } elseif ($status === 'stopped') {
        // Reset total : plus d'heure de départ ni de temps restant
        $updateSql = "UPDATE Config SET Status = $statusValue, Temps_restant = NULL, Heure_depart =
NULL WHERE id = 1";

    } elseif ($status === 'loading') {
        // Loading : juste MAJ du statut
        $updateSql = "UPDATE Config SET Status = $statusValue WHERE id = 1";
    }

    if (!empty($updateSql) && $conn->query($updateSql) === TRUE) {
        echo "Mise à jour réussie. Nouveau statut : $status ($statusValue)";
    } else {
        echo "Erreur SQL : " . $conn->error;
    }

    $conn->close();
}
?>
```

Annexe 4 - Fichier : update_joueurs.php

```
<?php
include '/var/www/html/private-estom/db_connect.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    foreach ($_POST['joueurs'] as $bomID => $pseudo) {
        $pseudo = substr(trim($pseudo), 0, 20); // Sécurisation (max 20 caractères)
        $sql = "UPDATE BOM SET Joueur = ? WHERE ID = ?";
        $stmt = $conn->prepare($sql);
        $stmt->bind_param("si", $pseudo, $bomID);
        $stmt->execute();
    }
}

echo "Mise à jour réussie !";
```

Annexe 5 - Fichier : update_collisions.php

```php
<?php
include '/var/www/html/private-estom/db_connect.php';

header('Content-Type: application/json');

// Récupérer les données reçues
$data = json_decode(file_get_contents("php://input"), true);

if (!isset($data["ip"]) || !isset($data["collisions"])) {
    echo json_encode(["error" => "Données invalides"]);
    exit;
}

$ip = $data["ip"];
$newCollisions = $data["collisions"];

// **Récupérer les collisions actuelles**
$sqlSelect = "SELECT Collisions FROM BOM WHERE IPAddr = ?";
$stmtSelect = $conn->prepare($sqlSelect);
$stmtSelect->bind_param("s", $ip);
$stmtSelect->execute();
$result = $stmtSelect->get_result();
$row = $result->fetch_assoc();
$stmtSelect->close();

$existingCollisions = json_decode($row['Collisions'], true) ?? [];

// **Ne pas ajouter des collisions en double**
foreach ($newCollisions as $collision) {
    $alreadyExists = false;
    foreach ($existingCollisions as $existingCollision) {
        if ($existingCollision['x'] == $collision['x'] && $existingCollision['y'] == $collision['y']) {
            $alreadyExists = true;
            break;
        }
    }
    if (!$alreadyExists) {
        $existingCollisions[] = $collision;
    }
}

// Mise à jour des collisions dans la base
$collisionsJson = json_encode($existingCollisions);
$sqlUpdate = "UPDATE BOM SET Collisions = ? WHERE IPAddr = ?";
$stmtUpdate = $conn->prepare($sqlUpdate);
$stmtUpdate->bind_param("ss", $collisionsJson, $ip);
$success = $stmtUpdate->execute();

if ($success) {
    echo json_encode(["success" => true]);
} else {
    echo json_encode(["error" => "Échec de l'enregistrement"]);
}

$stmtUpdate->close();
$conn->close();
?>
```

Annexe 6 - Fichier : accueil.php

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>ESTOM</title>
        <link rel="stylesheet" href="styles.css">
        <link rel="icon" type="image/x-icon" href="/img/estom-logo.ico">
    </head>
<body>
    <header>
        <h1 class="title">ESTOM</h1>
        <nav class="navbar">
            <ul class="menu">
            <a class="btn " id="btn1" href="./gestion-de-la-partie.php">Gestion de la partie</a>
            <a class="btn " id="btn2" href="./parametre-du-jeu.php">Paramétrage du Jeu</a>
            </ul>
        </nav>
        <div class="logo" id="logo">
            <a href="./accueil.php">
                <img src="./img/estom-logo.png" alt="" id="estom-logo">
            </a>
        </div>
        <div class="wave"></div>
    </header>
    <main>
        <div class="image-container">
            <div class="manetteImg">
                <a href="./gestion-de-la-partie.php">
                    <img src="./img/manette.png" alt="Manette" id="manette" class="manette-image">
                </a>
            </div>
            <div class="engrenagesImg">
                <a href="./parametre-du-jeu.php">
                    <img src="./img/engrenages.png" alt="Engrenages" id="engrenages"
class="engrenages-image">
                </a>
            </div>
        </div>

    </main>
    <footer>
        <p>&copy; 2025 ESTOM</p>
    </footer>
</body>
</html>
```

Annexe 7 - Fichier : get_infoBOM.php

```
<?php
include '/var/www/html/private-estom/db_connect.php';

header('Content-Type: application/json');

// Vérification de connexion
if (!$conn || $conn->connect_error) {
    die(json_encode(["error" => "Erreur de connexion à la base de données"]));
}

// Correspondance entre le nom de la couleur et l'entier dans la base de données
$colorMap = [
    "Vert" => 1,
    "Jaune" => 2,
    "Noir" => 3,
    "Bleu" => 4
];

$couleur = $_GET['couleur'] ?? '';  // Récupère la couleur du camion demandée
if (!isset($colorMap[$couleur])) {
```

```
    echo json_encode(["error" => "Couleur invalide"]);
    exit;
}

$couleurDB = $colorMap[$couleur]; // Convertit en entier pour la requête SQL

$sql = "SELECT IPAddr, Progression, NbrCollision, Joueur, Collisions  FROM BOM WHERE Couleur = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $couleurDB);
$stmt->execute();
$result = $stmt->get_result();

$data = [];
while ($row = $result->fetch_assoc()) {
    $collisions = json_decode($row['Collisions'], true) ?? [];

    if ((int)$row['NbrCollision'] === 0) {
        $collisions = [];
        $sqlUpdate = "UPDATE BOM SET Collisions = NULL WHERE IPAddr = ?";
        $stmtUpdate = $conn->prepare($sqlUpdate);
        $stmtUpdate->bind_param("s", $row['IPAddr']);
        $stmtUpdate->execute();
        $stmtUpdate->close();
    }
    // Ajouter un index unique pour chaque collision (evite les doublons)
    foreach ($collisions as $key => $collision) {
        $collisions[$key]['index'] = $key;  // L'index est basé sur la clé dans le tableau JSON
    }
    $data[] = [
        "ip" => $row['IPAddr'],
        "progression" => (float)$row['Progression'],
        "nbrCollision" => (int)$row['NbrCollision'],
        "joueur" => $row['Joueur'] ?? "",
        "collisions" => $collisions
    ];
}

echo json_encode($data);

$stmt->close();
$conn->close();
?>
```

Annexe 8 - Fichier : get_connexionIP.php

```
<?php
include '/var/www/html/private-estom/db_connect.php';

header('Content-Type: application/json');

$tables = ['BOM', 'TOM']; // Ajout des deux tables
$couleurs = [
    1 => "Vert",
    2 => "Jaune",
    3 => "Noir",
    4 => "Bleu"
];

$data = [];

foreach ($tables as $table) {
    $sql = "SELECT IPAddr, Connected, Couleur FROM $table"; // On boucle sur chaque table
    $result = $conn->query($sql);

    if ($result && $result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            // Extraire les deux derniers octets de l'adresse IP
            $ip_parts = explode('.', $row['IPAddr']);
            $short_ip = isset($ip_parts[2]) && isset($ip_parts[3]) ? $ip_parts[2] . '.' . $ip_parts[3]
: '??.??';
            //$short_ip = isset($ip_parts[3]) ? $ip_parts[3] : '??';
```

```
            $couleur = isset($couleurs[$row['Couleur']]) ? $couleurs[$row['Couleur']] : "Inconnu";

            $data[] = [
                'source' => $table, // Indique si l'IP vient de BOM ou TOM
                'couleur' => $couleur,
                'ip' => $short_ip,
                'connected' => (int)$row['Connected']
            ];
        }
    }
}

echo json_encode($data);
$conn->close();
?>
```

Annexe 9 - Fichier : get_timer_state.php

```
<?php
header('Content-Type: application/json');
include '/var/www/html/private-estom/db_connect.php';

$response = [];

$result = $conn->query("SELECT Status, duree, Temps_restant, Heure_depart FROM Config WHERE id = 1");

if ($result && $row = $result->fetch_assoc()) {
    $statusValue = (int)$row['Status'];
    $statusText = 'stopped';
    switch ($statusValue) {
        case 0:
            $statusText = 'stopped';
            break;
        case 1:
            $statusText = 'running';
            break;
        case 2:
            $statusText = 'paused';
            break;
        case 3:
            $statusText = 'loading';
            break;
    }

    // Par défaut, on prend Temps_restant
    $tempsRestant = (int)$row['Temps_restant'];

    if ($statusText === 'running') {
        // Ne PAS utiliser Temps_restant ici — recalcul dynamique à partir de Heure_depart
        $heureDepart = strtotime($row['Heure_depart']);
        $tempsEcoule = time() - $heureDepart;
        $tempsRestant = max(0, (int)$row['duree'] - $tempsEcoule);
    }

    $response = [
        'status' => $statusText,
        'temps_restant' => $tempsRestant,
        'duree' => (int)$row['duree']
    ];
} else {
    $response = ['error' => 'Impossible de récupérer l'état du timer.'];
}

$conn->close();
echo json_encode($response);
?>
```

Annexe 10 - Fichier : db_connect.php

```php
<?php
// Configuration de la base de données
$host = 'localhost';
$username = 'SuperViseur';
$password = 'SuperViseur-estom_2025';
$dbname = 'ESTOM';

// Création d'un objet MySQLi
$conn = new mysqli($host, $username, $password, $dbname);

// Vérification de la connexion
if ($conn->connect_error) {
    die("Erreur de connexion : " . $conn->connect_error);
}
?>
```

Annexe 11 - Fichier : updateTimer.js

```javascript
let timerInterval;
let remainingTime = timerDuration; // Valeur initiale du temps restant
let isTimerRunning = false; // État du timer
let isStarting = false;

function startCountdown(duration) {
    const timerDiv = document.getElementById('timer');
    remainingTime = duration; // Mise à jour de la valeur de remainingTime

    function updateTimer() {
        const minutes = Math.floor(remainingTime / 60);
        const seconds = remainingTime % 60;
        timerDiv.textContent = `Temps restant : ${minutes}m ${seconds}s`;

        if (remainingTime > 0) {
            remainingTime--; // Met à jour la variable restante
        } else {
            timerDiv.textContent = "Temps écoulé !";
            clearInterval(timerInterval);
            isTimerRunning = false;
            updateDatabaseStatus('stopped');
            updateButton(3); // "Lancer le timer"

            // Gestion de l'affichage du Podium
            showPodium();
            //updateDatabaseStatus('podium'); //optionnel

            // Masquer la modale après 20 secondes
            setTimeout(() => {
                hidePodium();
                //updateDatabaseStatus('stopped'); // Revenir à "stopped" après podium
            }, 60000);
        }
    }

    updateTimer();
    timerInterval = setInterval(updateTimer, 1000);
}

function updateButton(state) {
    const startButton = document.getElementById('startGameButton');
    const resetButton = document.getElementById('resetGameButton');

    switch (state) {
        case 1: // Timer en cours
            startButton.textContent = "Mettre en pause";
            resetButton.style.display = 'inline';
            break;
        case 2: // Timer en pause
            startButton.textContent = "Reprendre la partie";
            resetButton.style.display = 'inline';
```

```
            break;
        case 3: // Timer terminé ou non démarré
            startButton.textContent = "Lancer une nouvelle partie";
            resetButton.style.display = 'none';
            break;
    }
}

// Fonction AJAX pour récupérer l'état actuel
window.addEventListener('DOMContentLoaded', () => {
    fetch('get_timer_state.php')
        .then(response => response.json())
        .then(data => {
            if (data.error) {
                console.error(data.error);
                return;
            }

            const status = data.status;
            remainingTime = parseInt(data.temps_restant) || timerDuration;

            switch (status) {
                case 'running':
                    isTimerRunning = true;
                    startCountdown(remainingTime); // Appel avec la valeur correcte
                    updateButton(1);
                    break;
                case 'paused':
                    isTimerRunning = false;
                    const minutes = Math.floor(remainingTime / 60);
                    const seconds = remainingTime % 60;
                    document.getElementById('timer').textContent = `Temps restant : ${minutes}m
${seconds}s`;
                    updateButton(2);
                    break;
                case 'loading':
                    document.getElementById('timer').textContent = `Chargement de la partie...`;
                    updateButton(1);
                    break;
                case 'stopped':
                default:
                    isTimerRunning = false;
                    document.getElementById('timer').textContent = "Cliquez sur le bouton pour lancer
la Partie !";
                    updateButton(3);
                    break;
            }
        })
        .catch(error => {
            console.error("Erreur lors de la récupération du timer :", error);
        });
});

function updateDatabaseStatus(status, tempsRestant = null) {
    const xhr = new XMLHttpRequest();
    xhr.open("POST", "./update_status.php", true);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

    let params = `status=${status}`;
    if (tempsRestant !== null) {
        params += `&temps_restant=${tempsRestant}`;
    }

    xhr.onload = function () {
        if (xhr.status === 200) {
            console.log(`Statut mis à jour : ${status}`);
        } else {
            console.error("Erreur lors de la mise à jour du statut.");
        }
    };

    xhr.send(params);
}
```

```
document.getElementById('startGameButton').addEventListener('click', function () {
    if (isStarting) return;

    if (isTimerRunning) {
        clearInterval(timerInterval); // Met en pause le timer
        isTimerRunning = false;
        updateDatabaseStatus('paused');
        updateButton(2); // "Reprendre la partie"
    } else if (remainingTime === timerDuration || remainingTime === 0) {
        isStarting = true;
        remainingTime = timerDuration; // Réinitialiser à la durée initiale
        updateDatabaseStatus('stopped');
        // Compte à rebours de 5 secondes avant le démarrage
        let countdown = 5;
        const timerDiv = document.getElementById('timer');
        timerDiv.textContent = `Démarrage dans : ${countdown}s`;
        const countdownInterval = setInterval(() => {
            countdown--;
            timerDiv.textContent = `Démarrage dans : ${countdown}s`;
            updateDatabaseStatus('loading'); // Chargement et mise en place du jeu (5s)
            if (countdown <= 0) {
                clearInterval(countdownInterval);
                isStarting = false;
                updateDatabaseStatus('running'); // Mise à jour de la base de données
                startCountdown(remainingTime); // Appel avec la bonne valeur
                isTimerRunning = true;
                updateButton(1);
            }
        }, 1000);
    } else {
        // === CAS DE LA REPRISE D'UNE PARTIE EN PAUSE ===
        startCountdown(remainingTime); // Reprendre à partir du temps restant
        isTimerRunning = true;
        updateDatabaseStatus('running', remainingTime); // On passe aussi le temps restant
        updateButton(1); // "Mettre en pause"
    }
});

document.getElementById('resetGameButton').addEventListener('click', function () {
    clearInterval(timerInterval); // Annuler le timer en cours
    remainingTime = timerDuration; // Réinitialiser le temps
    document.getElementById('timer').textContent = "Cliquez sur le bouton pour lancer la partie";
    isTimerRunning = false;
    updateDatabaseStatus('stopped');
    updateButton(3); // "Lancer le timer"
});
```

Annexe 12 - Fichier : updateCamion.js

```
document.addEventListener("DOMContentLoaded", function () {
    let collisionPositions = {}; // Stocke les positions des collisions pour chaque camion
    //let lastProgression = {}; // Stocke la dernière position du camion pour un déplacement fluide

    function updateCamionPosition(couleur) {
        fetch(`get_InfoBOM.php?couleur=${couleur}`)
            .then(response => {
                    if (!response.ok) throw new Error(`Status HTTP : ${response.status}`);
                    return response.json();
                })
            .then(data => {
                if (data.error) {
                    console.error(`Erreur : ${data.error}`);
                    return;
                }

                data.forEach((camionData, index) => {
                    let ip = camionData.ip;
                    let totalTOMs = nbrTOM; // Nombre total de tours (TOM) sur la route
                    let tomsPasses = Math.round(camionData.progression / (100 / (totalTOMs))); //
Nombre de TOM atteints (dépassé)
                    let progressionPourcentage = Math.min(100, camionData.progression); //Progress de
la BOM (%)
```

```
                    let nbrCollision = camionData.nbrCollision ?? 0; //Nbr Collision par BOM
                    let collisions = camionData.collisions; //Coordonée des collisions
                    let camion =
document.querySelector(`.imageCamion[data-couleur="${couleur}"][data-ip="${ip}"]`);

                    if (!camion) return; // Si le camion n'existe pas, on ignore

                    let distance = Math.abs(progressionPourcentage - camion.dataset.prevProgression ||
0);

                    // Définir la durée en fonction de la distance
                    let animationDuration;
                    if (distance <= 10) {
                        animationDuration = 2000 + (distance * 100); // Entre 2 et 3 secondes pour les
petites distances
                    } else if (distance >= 50) {
                        animationDuration = 500 + (distance * 10); // Rapide si distance grande
                    } else {
                        animationDuration = 1500 + (distance * 20); // Progression moyenne
                    }

                    // Appliquer la transition fluide
                    camion.style.transition = `left ${animationDuration / 1000}s linear`;
                    camion.style.left = `${progressionPourcentage}%`;

                    // Sauvegarder la dernière progression
                    camion.dataset.prevProgression = progressionPourcentage;

                    // Mise à jour du score
                    let scoreText = document.getElementById(`score-${ip}`);


                    console.log(`progression = ${progressionPourcentage}, tomsPasses = ${tomsPasses}`);
                    let maxPossibleScore = 0; // Nouveau score max dynamique
                    let scoreBrut = 0;

                    if (optionMode === 0) {
                        // OPTION A : Score fixe par TOM
                        scoreBrut = tomsPasses * ptsRecolte;
                        maxPossibleScore = totalTOMs * ptsRecolte; // Score max basé sur le nombre de
TOM et ptsRecolte
                    } else if (optionMode === 1) {
                        // OPTION B : Score progressif
                        let basePts = ptsRecolte * 0.6; // La première TOM rapporte 60% du score de base
                        let increment = (ptsRecolte * 1.4 - basePts) / (totalTOMs - 1); // Augmente
progressivement

                        for (let i = 0; i < tomsPasses; i++) {
                            scoreBrut += basePts + (i * increment); // Ajoute des points de manière
progressive
                        }

                        // Calcul du score max théorique en Option B
                        for (let i = 0; i < totalTOMs; i++) {
                            maxPossibleScore += basePts + (i * increment);
                        }
                    }

                    // Appliquer le malus des collisions
                    scoreBrut -= nbrCollision * malusCollision;
                    scoreBrut = Math.max(0, scoreBrut); // Empêcher un score négatif

                    // Affichage sans normalisation, mais avec une limite dynamique
                    scoreBrut = Math.min(maxPossibleScore, Math.max(0, scoreBrut));

                    scoreText.textContent = `${Math.round(scoreBrut)}/${Math.round(maxPossibleScore)}`;

                    // Ajout ou mise à jour du pseudo au-dessus du camion
                    let pseudoSpan = camion.parentElement.querySelector(`.pseudo[data-ip="${ip}"]`);

                    if (!pseudoSpan) {
                        pseudoSpan = document.createElement("span");
                        pseudoSpan.className = "pseudo";
                        pseudoSpan.dataset.ip = ip;
```

```
                        pseudoSpan.style.position = "absolute";
                        pseudoSpan.style.fontWeight = "bold";
                        pseudoSpan.style.background = "rgba(255, 255, 255, 0.8)";
                        pseudoSpan.style.padding = "3px 5px";
                        pseudoSpan.style.borderRadius = "5px";
                        pseudoSpan.style.whiteSpace = "nowrap";
                        pseudoSpan.style.display = "none"; // Caché par défaut
                        pseudoSpan.style.transition = "left 0.5s linear"; // Ajout de transition
                        camion.parentElement.appendChild(pseudoSpan);
                    }

                    // Met à jour le pseudo du joueur
                    let joueurPseudo = camionData.joueur ?? ""; // Récupère le pseudo du joueur

                    if (joueurPseudo !== "") {
                        pseudoSpan.textContent = joueurPseudo;
                        pseudoSpan.style.display = "block"; // Affiche le pseudo si un joueur est défini
                    } else {
                        pseudoSpan.style.display = "none"; // Cache le pseudo s'il n'y a pas de joueur
                    }

                    // Appliquer la transition
                    pseudoSpan.style.transition = `left ${animationDuration / 1000}s linear`;

                    // Positionner le pseudo au-dessus du camion avec transition fluide
                    pseudoSpan.style.left = `${progressionPourcentage}%`;
                    pseudoSpan.style.top = `${camion.offsetTop - 20}px`; // Décalage au-dessus du camion


                    // Grisage des TOM en fonction de la progression du camion
                    let toms =
document.querySelectorAll(`.imageTour[data-couleur="${couleur}"][data-ip="${ip}"]`);
                    //console.log(`TOMs trouvés pour ${ip} :`, toms.length);
                    //console.log(`Progression du camion (${ip}) :`, progressionPourcentage);
                    toms.forEach((tom, i) => {
                        let espacement = (100 / (toms.length + 1)).toFixed(1);
                        espacement = parseFloat(espacement);
                        let tomPosition = (i + 1) * espacement;

                        console.log(`TOM ${i + 1}: position=${tomPosition.toFixed(2)},
progression=${progressionPourcentage.toFixed(2)}`);

                        if (progressionPourcentage >= tomPosition) {
                            tom.style.opacity = "0.3";
                            tom.style.filter = "grayscale(100%)";
                        } else {
                            tom.style.opacity = "1";
                            tom.style.filter = "none";
                        }
                    });

                    // ========= Gestion des collisions =========
                    // Suppression des collisions si leur nombre diminue
                    if (!collisionPositions[ip]) {
                        collisionPositions[ip] = [];
                    }

                    let existingIndexes = Object.keys(collisionPositions[ip]); // Index existants en
mémoire
                    let collisionIndexes = collisions.map(c => c.index); // Index en base

                    // Supprimer les collisions qui ne sont plus en base
                    existingIndexes.forEach(index => {
                        if (!collisionIndexes.includes(parseInt(index))) {
                            let el =
document.querySelector(`.collision[data-ip="${ip}"][data-index="${index}"]`);
                            if (el) el.remove();
                            delete collisionPositions[ip][index]; // Supprime aussi du cache JS
                        }
                    });

                    // Ajouter les collisions existantes en base si elles ne sont pas encore affichées
                    collisions.forEach(collision => {
                        let index = collision.index;
```

```
                        let x = collision.x;
                        let y = collision.y;

                        if (!collisionPositions[ip][index]) {
                            let imgCollision = document.createElement("img");
                            imgCollision.src = "./img/Crash.png";
                            imgCollision.className = "collision";
                            imgCollision.dataset.ip = ip;
                            imgCollision.dataset.index = index;
                            imgCollision.dataset.x = x;
                            imgCollision.dataset.y = y;
                            imgCollision.style = `
                                position: absolute;
                                top: ${y}%;
                                left: ${x}%;
                                transform: translate(-50%, -50%);
                                width: 40px;
                                opacity: 0.7;
                                z-index: 5;
                            `;
                            camion.parentElement.appendChild(imgCollision);
                            collisionPositions[ip][index] = { x, y };
                        }
                    });

                    // Ajouter de nouvelles collisions si le nombre a augmenté
                    if (collisionIndexes.length < nbrCollision) {
                        let newIndex = collisionIndexes.length; // Nouvel index
                        let collisionX = progressionPourcentage + (Math.random() * 10 - 5);
                        let collisionY = 50 + (Math.random() * 10 - 5);

                        let newCollision = { x: collisionX, y: collisionY, index: newIndex };

                        collisionPositions[ip][newIndex] = newCollision;

                        let imgCollision = document.createElement("img");
                        imgCollision.src = "./img/Crash.png";
                        imgCollision.className = "collision";
                        imgCollision.dataset.ip = ip;
                        imgCollision.dataset.index = newIndex;
                        imgCollision.dataset.x = collisionX;
                        imgCollision.dataset.y = collisionY;
                        imgCollision.style = `
                            position: absolute;
                            top: ${collisionY}%;
                            left: ${collisionX}%;
                            transform: translate(-50%, -50%);
                            width: 40px;
                            opacity: 0.7;
                            z-index: 5;
                        `;
                        camion.parentElement.appendChild(imgCollision);

                        // Enregistrer les nouvelles collisions dans la base
                        fetch('update_collisions.php', {
                            method: 'POST',
                            headers: { "Content-Type": "application/json" },
                            body: JSON.stringify({ ip, collisions:
Object.values(collisionPositions[ip]) })
                        });
                    }
                });
            })
        .catch(error => console.error("Erreur AJAX :", error));
    }

    function updateAllCamions() {
        let couleurs = ["Vert", "Jaune", "Noir", "Bleu"];
        couleurs.forEach(couleur => updateCamionPosition(couleur));
    }

    setInterval(updateAllCamions, 1000);
});
```

Annexe 13 - Fichier : updateConnexion.js

```
// Objet pour stocker l'état précédent des connexions
let previousState = {};

function updateConnexion() {
    fetch('get_connexionIP.php')
        .then(response => response.json())
        .then(data => {
            console.log("Données reçues :", data); // DEBUG

            let currentState = {}; // Stocke l'état actuel
            let tomUsedIPs = {}; // Suivi des IPs déjà attribuées pour éviter les doublons
            let tomCount = { "Vert": 0, "Jaune": 0, "Noir": 0, "Bleu": 0 };

            // Réinitialiser les IP affichées
            document.querySelectorAll('.ip-text').forEach(span => {
                span.textContent = 'N/A';
            });

            data.forEach(entry => {
                let couleur = entry.couleur;
                let ip = entry.ip;
                let type = entry.source; // "BOM" ou "TOM"
                let isConnected = entry.connected === 1;

                console.log(`Vérification ${type} - Couleur: ${couleur}, IP: ${ip}, Connecté:
${isConnected}`);

                if (type === "BOM") {
                    let bomElements = document.querySelectorAll(`[id^="BOM-${couleur}-"]`); // Récupère
tous les BOM de cette couleur

                    bomElements.forEach((bomElement, index) => {
                        let ipElement = document.getElementById(`BOM-IP-${couleur}-${index}`);

                        currentState[`BOM-${couleur}-${index}`] = isConnected;

                        if (isConnected) {
                            // Vérifier si l'état était différent avant d'ajouter l'effet
                            if (!previousState[`BOM-${couleur}-${index}`]) {
                                bomElement.classList.add('connecte');
                                bomElement.classList.remove('grise', 'deconnecte');
                            }
                            if (ipElement && ipElement.textContent !== ip) {
                                ipElement.textContent = ip;
                            }
                        }
                    });
                } else if (type === "TOM" && tomCount[couleur] < 10) {
                    let tomElements = document.querySelectorAll(`[id^="TOM-${couleur}-"]`);

                    // Vérifier qu'on n'a pas déjà attribué cette IP
                    if (!tomUsedIPs[ip]) {
                        for (let i = 0; i < tomElements.length; i++) {
                            let tomElement = tomElements[i];
                            let ipElement = document.getElementById(`TOM-IP-${couleur}-${i}`);
                            let key = `TOM-${couleur}-${i}`;

                            // Vérifier si ce TOM n'est pas déjà pris
                            if (!currentState[key]) {
                                currentState[key] = isConnected;

                                if (isConnected) {
                                    if (!previousState[key]) {
                                        tomElement.classList.add('connecte');
                                        tomElement.classList.remove('grise', 'deconnecte');
                                    }
                                    if (ipElement && ipElement.textContent !== ip) {
                                        ipElement.textContent = ip;
                                    }

                                    // Marquer cette IP comme utilisée
```

```
                                    tomUsedIPs[ip] = true;
                                    break; // Sortir de la boucle pour ne pas attribuer la même IP à
plusieurs TOM
                                }
                            }
                        }
                    }
                }
            });

            // Gestion des déconnexions
            Object.keys(previousState).forEach(id => {
                if (previousState[id] && !currentState[id]) {
                    let element = document.getElementById(id);
                    if (element) {
                        element.classList.add('deconnecte');
                        element.classList.remove('connecte');

                        // Après 5 secondes, le remet en gris
                        setTimeout(() => {
                            element.classList.add('grise');
                            element.classList.remove('deconnecte');
                        }, 2000);
                    }
                }
            });

            // Mettre à jour l'état précédent
            previousState = currentState;
        })
        .catch(error => console.error("Erreur lors de la récupération des connexions:", error));
}

// Rafraîchir les données toutes les 2.5 secondes
setInterval(updateConnexion, 2500);

// Lancer une mise à jour immédiate au chargement
updateConnexion();
```

Annexe 14 - Fichier : main.cpp

```
#include "capp.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    CApp w;
    //w.show();

    return a.exec();
}
```

Annexe 15 - Fichier : CApp.cpp

```
#include <QTcpSocket>
#include <QJsonDocument>
#include <QJsonObject>
#include "capp.h"

CApp::CApp(QObject *parent)
    : QObject(parent)
{
    init();
}

CApp::~CApp()
{
    if (_dbReader) {
        _dbReader->requestInterruption();
        _dbReader->wait();
        delete _dbReader;
    }
    if (_tcpThread) {
        _tcpThread->quit();
        _tcpThread->wait();
        delete _tcpThread;
    }
    delete _tcpSender;
    if (_connectThread) {
        _connectThread->quit();
        _connectThread->wait();
        delete _connectThread;
    }
    delete _connectChecker;
}

void CApp::onConfigUpdated(QString etat, QString options, QString ptsRecolte, QString nbrTOM)
{
    static int _etatPrecedent = -1; // Stocke l'état précédent (-1 pour garantir un premier envoi)
    this->_etat = etat;  //0=stopped ; 1=running ; 2=paused ; 3=loading (countdown)
    this->_options = options;
    this->_ptsRecolte = ptsRecolte;
    this->_nbrTOM = nbrTOM;

    //Ordre à envoyé selon Etat?
    //1=Envoie demande info,
    //2=Initialiser, (loading = 3)
    //6=Pause, (stopped & paused = 0 & 2)
    //7=Continue (running = 1)
    int currentEtat = _etat.toInt();
    QStringList ipList = _dbReader->getAllIPs(); // Récupère toutes les IPs

    if (currentEtat != _etatPrecedent) {
        for (const QString &ip : ipList) {
            switch (currentEtat) {
                case 0: //Envoie ordre 6 pour mise en pause
                case 2:
                    sendMsgTCP(ip, 6); // Envoie l'ordre 6 à chaque IP
                    qDebug() << "Ordre 6 envoyé à l'IP :" << ip;
                    break;
                case 1:  //Envoie ordre 7 pour continuer/commencer le jeu
                    sendMsgTCP(ip, 7); // Envoie l'ordre 7 à chaque IP
                    qDebug() << "Ordre 7 envoyé à l'IP :" << ip;
                    break;
                case 3: //Envoie Ordre 2 pour initialisation
                    sendMsgTCP(ip, 2, _nbrTOM); // Envoie l'ordre 2 à chaque IP
                    qDebug() << "Ordre 2 envoyé à l'IP :" << ip;
                    break;
            }
        }
        _etatPrecedent = currentEtat;
    }
}
```

```
void CApp::onInfoUpdated(QString type, QString ip, QString stockage, QString status, QString couleur,
QString progression, QString nbrCollision, QString ipTOM)
{
    this->_type = type;
    this->_ip = ip;
    this->_stockage = stockage;
    this->_status = status; //Reception par SuperViseur : 1 = Information, 2= Remplissage, 3= récupéré,
4=StopRemplissage,5= Collision/Choc
    this->_couleur = couleur; //1=vert, 2=jaune, 3=noir, 4=bleu
    this->_progression = progression;
    this->_nbrCollision = nbrCollision;
    this->_ipTOM = ipTOM;

    qDebug() << "<=====|MAJ-INFO-" << _ip <<"|=====>";


    if (_type == "BOM") {
        _connected = "1";
        //qDebug() << "IP envoyé à bdd:" << _ip;
        switch (_status.toInt()) {
            case 1: //Information
                _dbReader->insertDB("BOM", QVariantList{_progression, _couleur, _status, _connected,
_nbrCollision, _ip});
                break;
            case 2: //Remplissage (vérification TOM stockage)
                if (_dbReader->checkTOM(_ipTOM)) {
                    qDebug() << "Remplissage en cours de BOM (" << _ip << ") qui vide TOM (" << _ipTOM
<< ")";
                    sendMsgTCP(_ipTOM, 3);
                } else {
                    qDebug() << "Remplissage en cours intérronpu : TOM (" << _ipTOM << ") déjà vide.";
                    sendMsgTCP(_ip, 5);
                }
                break;
            case 3: //Récupéré
                _stockage = "0";
                _dbReader->insertDB("BOM", QVariantList{_progression, _couleur, _status, _connected,
_nbrCollision, _ip});
                _dbReader->insertDB("TOM", QVariantList{_stockage, _couleur, _connected, _ipTOM});
                sendMsgTCP(_ipTOM, 4);
                break;
            case 4: //Stop Remplissage
                sendMsgTCP(_ipTOM, 2);
                break;
            case 5:  //Collision/choc
                _dbReader->insertDB("BOM", QVariantList{_progression, _couleur, _status, _connected,
_nbrCollision, _ip});
                break;
            default:
                qDebug() << "Status BOM inconnu :" << _status;
                break;
        }
    }

    if (_type == "TOM") {
        int _connected = 1;
        _dbReader->insertDB("TOM", QVariantList{_stockage, _couleur, QString::number(_connected),
_ip});
    }

}

void CApp::init() {
    // Initialisation de la base de données
    _dbReader = new CDatabase(this);
    //udpReader = new CUDPReader;

    // Démarrer le thread UDP
    //udpThread = new QThread(this);
    //CUDPReader *udpReader = new CUDPReader();
    //udpReader->moveToThread(udpThread);

    // Démarrage du serveur TCP dans un thread
    _tcpServer = new CTcpServer;
```

```
    _tcpThread = new QThread(this);
    _tcpServer->moveToThread(_tcpThread);

    // Connexion du signal de démarrage
    connect(_tcpThread, &QThread::started, [=]() {
        _tcpServer->startServer(5005);
    });

    connect(_tcpServer, &CTcpServer::infoUpdated, this, &CApp::onInfoUpdated);
    // Nettoyage du thread quand il se termine
    connect(_tcpThread, &QThread::finished, _tcpServer, &QObject::deleteLater);
    _tcpThread->start();
    // Création du sender TCP
    _tcpSender = new CTcpSender(_tcpServer, this);
    _tcpSender->moveToThread(_tcpThread);
    // connecte
    connect(this, &CApp::sendTcpMessageRequested, _tcpSender, &CTcpSender::sendMessage);


    // Connexion pour les mises à jour de configuration
    connect(_dbReader, &CDatabase::configUpdated, this, &CApp::onConfigUpdated);
    _dbReader->start();

    _connectChecker = new CConnectChecker(_dbReader);
    _connectChecker->start();

}

void CApp::sendMsgTCP(const QString &ip, int ordre, const QString &extraData)
{
    //Ordre info :
    //1=Envoie demande info,
    //2=Initialiser,
    //3=Remplissage en cours (TOM),
    //4=Vide-toi (TOM),
    //5=TOM déjà vidé (BOM),
    //6=Pause,
    //7=Continue
    emit sendTcpMessageRequested(ip, ordre, extraData);
}
```

Annexe 16 - Fichier : CApp.h

```
#ifndef CAPP_H
#define CAPP_H

#include <QObject>
#include <QDebug>
#include <QThread>

#include "CDatabase.h"
#include "CTcpServer.h"
#include "CTcpSender.h"
#include "CConnectChecker.h"

class CApp : public QObject
{
    Q_OBJECT

public:
    CApp(QObject *parent = nullptr);
    ~CApp();
private slots:
    void onConfigUpdated(QString etat, QString options, QString ptsRecolte, QString nbrTOM);
    void onInfoUpdated(QString type, QString ip, QString stockage, QString status, QString couleur,
QString progression, QString nbrCollision, QString ipTOM);

private:
    void init();
    CDatabase *_dbReader;
    QThread *_connectThread;
    CConnectChecker *_connectChecker;
```

```
    QThread *_tcpThread;

    CTcpServer *_tcpServer;
    CTcpSender *_tcpSender;
    void sendMsgTCP(const QString &ip, int ordre, const QString &extraData ="");

    //Config variable
    QString _etat;
    QString _options;
    QString _ptsRecolte;
    QString _nbrBOM;
    QString _nbrTOM;

    //Info variable
    QString _type;
    QString _ip;
    QString _stockage;
    QString _status;
    QString _couleur;
    QString _progression;
    QString _nbrCollision;
    QString _ipTOM;

    QString _connected;

    bool _messageEnvoye = false;
signals:
    void sendTcpMessageRequested(const QString &ip, int ordre, const QString &extraData ="");
};


#endif // CAPP_H
```

Annexe 17 - Fichier : CDatabase.cpp

```
#include "CDatabase.h"

CDatabase::CDatabase(QObject *parent) : QThread(parent)
{
    // Connexion temporaire pour reset
    QSqlDatabase resetDb = QSqlDatabase::addDatabase("QMYSQL", "ResetConnection");
    resetDb.setHostName("localhost");
    resetDb.setPort(3306);
    resetDb.setDatabaseName("ESTOM");
    resetDb.setUserName("SuperViseur");
    resetDb.setPassword("SuperViseur-estom_2025");

    if (!resetDb.open()) {
        qDebug() << "Erreur : Impossible d'ouvrir la BDD pour le reset.";
        qDebug() << resetDb.lastError().text();
    } else {
        qDebug() << "Connexion temporaire pour reset réussie.";
        resetTables(resetDb);
    }

    // Supprimer la connexion temporaire pour éviter les conflits
    resetDb.close();
    QSqlDatabase::removeDatabase("ResetConnection");

    // La vraie connexion du thread sera ouverte dans run()
}

CDatabase::~CDatabase()
{
    requestInterruption();
    wait();
}

void CDatabase::resetTables(QSqlDatabase &db)
{
    QSqlQuery query(db);
```

```
    QStringList tables = {"BOM", "TOM"};
    for (const QString &table : tables) {
        //qDebug() << "Réinitialisation de la table :" << table;

        if (!query.exec("DELETE FROM " + table)) {
            qDebug() << "Erreur lors de la suppression des données de" << table << ":" <<
query.lastError().text();
        }

        if (!query.exec("ALTER TABLE " + table + " AUTO_INCREMENT = 1")) {
            qDebug() << "Erreur lors de la réinitialisation de l'ID de" << table << ":" <<
query.lastError().text();
        }
    }

    qDebug() << "Tables BOM et TOM réinitialisées avec succès.";
}

void CDatabase::run()
{
    threadDb = QSqlDatabase::addDatabase("QMYSQL", "CDatabaseConnection");
    threadDb.setHostName("localhost");
    threadDb.setPort(3306);
    threadDb.setDatabaseName("ESTOM");
    threadDb.setUserName("SuperViseur");
    threadDb.setPassword("SuperViseur-estom_2025");

    if (!threadDb.open()) {
        qDebug() << "Erreur : Impossible d'ouvrir la base de données depuis le thread.";
        qDebug() << threadDb.lastError().text();
        return;
    }
    while (!isInterruptionRequested()) {
        QSqlQuery query(threadDb);
        if (!query.exec("SELECT * FROM Config")) {
            qDebug() << "Erreur SELECT Config :" << query.lastError().text();
            QThread::sleep(1);
            continue;
        }
        if (!query.next()) {
            qDebug() << "La table Config est vide ou aucune ligne trouvée.";
            QThread::sleep(1);
            continue;
        }
        //qDebug() << "===== BDD =====";
        QString Status = query.value("Status").toString();
        //qDebug() << "Status trouvé :" << Status;
        QString Options = query.value("Options").toString();
        //qDebug() << "Options trouvé :" << Options;
        QString PtsRecolte = query.value("PtsRecolte").toString();
        //qDebug() << "Pts par Récolte :" << PtsRecolte;
        QString NbrBOM_V = query.value("NbrBOM_V").toString();
        QString NbrBOM_J = query.value("NbrBOM_J").toString();
        QString NbrBOM_N = query.value("NbrBOM_N").toString();
        QString NbrBOM_B = query.value("NbrBOM_B").toString();
        //qDebug() << "Nombre de BOM :" << NbrBOM;
        QString NbrTOM = query.value("NbrTOM").toString();
        //qDebug() << "Nombre de TOM :" << NbrTOM;
        //qDebug() << "Lecture BDD";
        emit configUpdated(Status, Options, PtsRecolte, NbrTOM);

        QThread::sleep(1);  // Pause de 5 secondes entre chaque lecture
    }
    threadDb.close();
    QSqlDatabase::removeDatabase("CDatabaseConnection");
}
bool CDatabase::insertDB(const QString &table, const QVariantList &values)
{
    if (values.isEmpty()) {
        qDebug() << "Erreur : aucune valeur fournie pour l'insertion.";
        return false;
    }

    // Construire la liste des placeholders "?" pour la requête préparée
```

```
    QStringList placeholders;
    for (int i = 0; i < values.size(); ++i) {
        placeholders.append("?");
    }

    QStringList columns;
    columns.clear();
    if (table == "BOM") {
       QStringList arg = {"Progression", "Couleur", "Status", "Connected", "NbrCollision", "IPAddr"};
       columns.append(arg);
       qDebug() << "Table BOM";
    }else if (table == "TOM") {
       QStringList arg = {"Stockage", "Couleur", "Connected", "IPAddr"};
       columns.append(arg);
       qDebug() << "Table TOM";
    } else {
        qDebug() << "Table non reconue";
        return false;
    }
    if (columns.size() != values.size()) {
        qDebug() << "Erreur : Le nombre de valeurs ne correspond pas aux colonnes. Colum :" <<
columns.size() << "valeur:" << values.size();
        return false;
    }
    // Construire la requête "ON DUPLICATE KEY UPDATE"
    //permet de mettre à jour une ligne existante lorsqu'une contrainte d'unicité est violée lors d'un
INSERT
    QStringList updateClauses;
    for (const QString &col : columns) {
        if (col != "IPAddr") { // Ne pas mettre à jour l'IPAddr car c'est la clé
            updateClauses.append(QString("%1 = VALUES(%1)").arg(col));
        }
    }

    QString queryString = QString(
        "INSERT INTO %1 (%2) VALUES (%3) "
        "ON DUPLICATE KEY UPDATE %4"
    ).arg(table)
     .arg(columns.join(", "))
     .arg(placeholders.join(", "))
     .arg(updateClauses.join(", "));

    QString ip;

    if (table == "TOM" && values.size() > 3) {
        ip = values[3].toString();
    } else if (table == "BOM" && values.size() > 5) {
        ip = values[5].toString();
    } else {
        ip = "Unknown";
    }

    qDebug() << "Commande suite ip :" << ip << " " << queryString;

    QSqlQuery query(threadDb);
    query.prepare(queryString);

    for (const QVariant &value : values) {
        query.addBindValue(value);
    }
    if (!query.exec()) {
        qDebug() << "Erreur lors de l'insertion :" << query.lastError().text();
        return false;
    }
    return true;
}
bool CDatabase::checkTOM(const QString &ipTOM)
{
    QSqlQuery query(threadDb);
    query.prepare("SELECT Stockage FROM TOM WHERE IPAddr = :ipTOM LIMIT 1");
    query.bindValue(":ipTOM", ipTOM);

    if (!query.exec()) {
```

```
        qDebug() << "Erreur lors de la vérification du stockage de la TOM pour l'IP" << ipTOM << ":" <<
query.lastError().text();
        return false; // En cas d'erreur, on considère que c'est vide pour éviter des blocages
    }

    if (query.next()) {
        int stockage = query.value("Stockage").toInt();
        if (stockage == 1) {
            qDebug() << "TOM avec IP" << ipTOM << "est pleine.";
            return true;
        } else {
            qDebug() << "TOM avec IP" << ipTOM << "est vide.";
            return false;
        }
    }

    qDebug() << "Aucune TOM trouvée avec l'IP" << ipTOM << ".";
    return false; // Si aucune donnée trouvée, on considère comme vide
}

QStringList CDatabase::getAllIPs()
{
    QStringList ipList;
    QSqlQuery query(threadDb);

    if (!query.exec("SELECT IPAddr FROM BOM UNION SELECT IPAddr FROM TOM")) {
        qDebug() << "Erreur lors de la récupération des IPs:" << query.lastError().text();
        return ipList;
    }

    while (query.next()) {
        ipList.append(query.value(0).toString());
    }

    //qDebug() << "Liste des IPs : " << ipList;
    return ipList;
}
```

Annexe 18 - Fichier : CDatabase.h

```
#ifndef CDATABASE_H
#define CDATABASE_H

#include <QThread>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include <QDebug>

class CDatabase : public QThread
{
    Q_OBJECT

public:
    explicit CDatabase(QObject *parent = nullptr);
    ~CDatabase();

    bool insertDB(const QString &table, const QVariantList &values);
    bool checkTOM(const QString &ipTOM);

    QStringList getAllIPs();
protected:
    void run() override;  // Méthode exécutée en boucle dans le thread

private:
    QSqlDatabase threadDb;
    void resetTables(QSqlDatabase &db);
signals:
    void configUpdated(QString status, QString options, QString ptsRecolte, QString nbrTOM);
    void ipListUpdated(QStringList ipList);
};
```

```
#endif // CDBREADER_H
```

Annexe 19 - Fichier : CConnectChecker.cpp

```
#include "CConnectChecker.h"
#include <QDebug>
#include <QSqlError>

CConnectChecker::CConnectChecker(QObject *parent)
    : QThread(parent), shouldStop(false) {}

CConnectChecker::~CConnectChecker() {
    stop();
    wait();
}

void CConnectChecker::stop() {
    shouldStop = true;
}

QSqlDatabase CConnectChecker::getDatabaseConnection()
{
    QString connectionName =
QString("checkerConnection_%1").arg(reinterpret_cast<quintptr>(QThread::currentThread()));

    if (QSqlDatabase::contains(connectionName)) {
        return QSqlDatabase::database(connectionName);
    }

    QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL", connectionName);
    db.setHostName("localhost");
    db.setPort(3306);
    db.setDatabaseName("ESTOM");
    db.setUserName("SuperViseur");
    db.setPassword("SuperViseur-estom_2025");

    if (!db.open()) {
        qDebug() << "Erreur : Impossible d'ouvrir la base de données.";
        qDebug() << db.lastError().text();
    }
    return db;
}

void CConnectChecker::run()
{
    while (!shouldStop) {
        QStringList ipList = getAllIPs();  // Met à jour la liste à chaque cycle

        if (ipList.isEmpty()) {
            qDebug() << "Aucune IP trouvée, attente de 5 secondes...";
        }

        for (const QString &ip : ipList) {
            if (shouldStop) break;  // Vérifie l'arrêt en cours d'exécution

            bool isConnected = pingIP(ip);
            updateConnectStatus(ip, isConnected ? 1 : 0);
        }

        for (int i = 0; i < 50 && !shouldStop; ++i) {
            QThread::msleep(100);  // Pause plus réactive (5s au total)
        }
    }

    // Nettoyage de la connexion après utilisation
    QSqlDatabase::removeDatabase(QSqlDatabase::database().connectionName());
}

bool CConnectChecker::pingIP(const QString &ip) {
    QProcess process;
    QString command = QString("ping -c 1 -W 1 %1").arg(ip);  // Teste une seule fois avec un timeout de
1s

    process.start(command);
```

```
    process.waitForFinished(2000);  // Timeout général de 2s

    return process.exitCode() == 0; // Retourne true si l'IP répond au ping
}

int CConnectChecker::getConnectStatus(const QString &ip)
{
    QSqlDatabase db = getDatabaseConnection();
    QSqlQuery query(db);
    query.prepare("SELECT Connected FROM BOM WHERE IPAddr = :ip UNION SELECT Connected FROM TOM WHERE
IPAddr = :ip");
    query.bindValue(":ip", ip);

    if (!query.exec()) {
        qDebug() << "Erreur lors de la récupération du statut de connexion pour" << ip << ":" <<
query.lastError().text();
        return -1;
    }

    if (query.next()) {
        return query.value(0).toInt();
    }

    return -1;
}

QStringList CConnectChecker::getAllIPs()
{
    QStringList ipList;
    QSqlDatabase db = getDatabaseConnection();
    QSqlQuery query(db);

    if (!query.exec("SELECT IPAddr FROM BOM UNION SELECT IPAddr FROM TOM")) {
        qDebug() << "Erreur lors de la récupération des IPs:" << query.lastError().text();
        return ipList;
    }

    while (query.next()) {
        ipList.append(query.value(0).toString());
    }

    //qDebug() << "Liste des IPs : " << ipList;
    return ipList;
}

void CConnectChecker::updateConnectStatus(const QString &ip, int status)
{
    QSqlDatabase db = getDatabaseConnection();
    QSqlQuery query(db);

    query.prepare("UPDATE BOM SET Connected = :status WHERE IPAddr = :ip");
    query.bindValue(":status", status);
    query.bindValue(":ip", ip);
    if (!query.exec()) {
        qDebug() << "Erreur lors de la mise à jour de BOM:" << query.lastError().text();
    }

    query.prepare("UPDATE TOM SET Connected = :status WHERE IPAddr = :ip");
    query.bindValue(":status", status);
    query.bindValue(":ip", ip);
    if (!query.exec()) {
        qDebug() << "Erreur lors de la mise à jour de TOM:" << query.lastError().text();
    }
}
```

Annexe 20 - Fichier : CConnectChecker.h

```
#ifndef CCONNECTCHECKER_H
#define CCONNECTCHECKER_H

#include <QObject>
#include <QThread>
#include <QStringList>
#include <QProcess>
#include <QtConcurrent/QtConcurrent>
#include "CDatabase.h"

class CConnectChecker : public QThread
{
    Q_OBJECT
public:
    explicit CConnectChecker(QObject *parent = nullptr);
    ~CConnectChecker();


    void stop();
protected:
    void run() override;

private:
    bool shouldStop;

    QSqlDatabase getDatabaseConnection();
    bool pingIP(const QString &ip);
    int getConnectStatus(const QString &ip);

    QStringList getAllIPs();
    void updateConnectStatus(const QString &ip, int status);

};

#endif // CCONNECTCHECKER_H
```

Annexe 21 - Fichier : CTcpServer.cpp

```
// CTcpServer.cpp
#include "CTcpServer.h"

CTcpServer::CTcpServer(QObject *parent) : QTcpServer(parent) {}

void CTcpServer::startServer(quint16 port) {
    if (!this->listen(QHostAddress::Any, port)) {
        qDebug() << "Server TCP n'a pas démarré!";
    } else {
        qDebug() << "Server TCP démarré sur le port" << port;
    }
}

QTcpSocket *CTcpServer::getSocketForIP(const QString &ip) const
{
    //qDebug() << "Recherche de socket pour l'ip: " << ip;
    //qDebug() << "Ips disponibles dans socketMap: " << socketMap.keys();
    return socketMap.value(ip, nullptr);
}

void CTcpServer::incomingConnection(qintptr socketDescriptor) {
    QMetaObject::invokeMethod(this, [=]() {
        QTcpSocket *clientSocket = new QTcpSocket(); // Pas de parent ici !

        if (!clientSocket->setSocketDescriptor(socketDescriptor)) {
            qDebug() << "Erreur de socket TCP !";
            clientSocket->deleteLater();
            return;
```

```
        }

        // Adresse IP du client connecté
        QString senderIP = clientSocket->peerAddress().toString();
        if (senderIP.startsWith("::ffff:")) {
            senderIP = senderIP.right(senderIP.length() - 7);
        }
        qDebug() << "Nouvelle connexion TCP de:" << senderIP;

        // Stockage temporaire uniquement si socket est en ConnectedState
        if (clientSocket->state() == QAbstractSocket::ConnectedState) {
            socketMap[senderIP] = clientSocket;
        } else {
            connect(clientSocket, &QTcpSocket::connected, this, [=]() {
                qDebug() << "Connexion confirmée pour:" << senderIP;
                socketMap[senderIP] = clientSocket;
            });
        }

        QByteArray *buffer = new QByteArray;

        connect(clientSocket, &QTcpSocket::readyRead, this, [=]() {
            buffer->append(clientSocket->readAll());

            if (!buffer->contains('\n')) return;

            QList<QByteArray> messages = buffer->split('\n');
            for (int i = 0; i < messages.size() - 1; ++i) {
                QByteArray message = messages[i].trimmed();
                if (message.isEmpty()) continue;

                qDebug() << "<=====|TCP-REQUEST-" << senderIP <<"|=====>";
                qDebug() << "Message TCP reçu:" << message;

                QJsonParseError parseError;
                QJsonDocument doc = QJsonDocument::fromJson(message, &parseError);

                if (parseError.error != QJsonParseError::NoError || !doc.isObject()) {
                    qWarning() << "Erreur JSON TCP:" << parseError.errorString();
                    continue;
                }

                QJsonObject jsonObj = doc.object();

                QString stockage = jsonObj.value("Stockage").toString();
                QString couleur = jsonObj.value("Couleur").toString();
                QString progression = jsonObj.value("Progression").toString();
                QString nbrCollision = jsonObj.value("NbrCollision").toString();
                QString status = jsonObj.value("Status").toString();
                QString ipTOM = jsonObj.value("ipTOM").toString();
                QString type = jsonObj.value("Type").toString();

                QString senderIP = clientSocket->peerAddress().toString();
                if (senderIP.startsWith("::ffff:")) {
                    senderIP = senderIP.right(senderIP.length() - 7);
                }

                qDebug() << "Stockage" << stockage;
                qDebug() << "Couleur" << couleur;
                qDebug() << "Progression" << progression;
                qDebug() << "NbrCollision" << nbrCollision;
                qDebug() << "Status" << status;
                qDebug() << "ipTOM" << ipTOM;
                qDebug() << "Type" << type;

                emit infoUpdated(type, senderIP, stockage, status, couleur, progression, nbrCollision,
ipTOM);
            }

            *buffer = messages.last();
        });

        connect(clientSocket, &QTcpSocket::disconnected, clientSocket, &QTcpSocket::deleteLater);
    }, Qt::QueuedConnection);
```

```
}
```

Annexe 22 - Fichier : CTcpServer.h

```
// CTcpServer.h

#ifndef CTCP_SERVER_H
#define CTCP_SERVER_H

#include <QTcpServer>
#include <QTcpSocket>
#include <QJsonDocument>
#include <QJsonObject>
#include <QDebug>

class CTcpServer : public QTcpServer {
    Q_OBJECT
public:
    explicit CTcpServer(QObject *parent = nullptr);
    void startServer(quint16 port = 5005);
    QTcpSocket* getSocketForIP(const QString &ip) const;

signals:
    void infoUpdated(QString type, QString ip, QString stockage, QString status, QString couleur,
QString progression, QString nbrCollision, QString ipTOM);

protected:
    void incomingConnection(qintptr socketDescriptor) override;
private:
    QMap<QString, QTcpSocket*> socketMap;
};

#endif // CTCP_SERVER_H
```

Annexe 23 - Fichier : CTcpSender.cpp

```
// CTcpSender.cpp
#include "CTcpSender.h"

CTcpSender::CTcpSender(CTcpServer *tcpServer, QObject *parent) : QObject(parent), _tcpServer(tcpServer)
{}

void CTcpSender::sendMessage(const QString &ip, int ordre, const QString &nbrTOM)
{
    qDebug() << "<=====|TCP-SENDER-" << ip <<"|=====>";
    QTcpSocket *socket = _tcpServer->getSocketForIP(ip);

    if (!socket || socket->state() != QAbstractSocket::ConnectedState) {
        qWarning() << "Socket non disponible ou déconnecté pour l'IP:" << ip;
        return;
    }

    QJsonObject jsonObj;
    jsonObj["Ordre"] = ordre;
    if (ordre == 2) {
        qDebug() << "Ordre = 2 donc envoie nbrTOM :" << nbrTOM;
        jsonObj["nbrTOM"] = nbrTOM;
    }

    QJsonDocument doc(jsonObj);
    QByteArray message = doc.toJson(QJsonDocument::Compact) + '\n';

    socket->write(message);
    socket->flush();
    qDebug() << "Message TCP envoyé à" << ip;
    qDebug() << "Données:" << message;
}
```

Annexe 23 - Fichier : CTcpSender.h

```
// CTcpSender.h
#pragma once

#include <QTcpSocket>
#include "CTcpServer.h"
#include <QObject>
#include <QJsonObject>
#include <QJsonDocument>
#include <QHostAddress>
#include <QDebug>

class CTcpSender : public QObject
{
    Q_OBJECT

public:
    explicit CTcpSender(CTcpServer *tcpServer, QObject *parent = nullptr);
    void sendMessage(const QString &ip, int ordre, const QString &nbrTOM = QString());

private:
    CTcpServer *_tcpServer = nullptr;
};
```

Annexes Badr ERRAFI - ER 1

Annexe 1 - Fiche de suivie

Janvier 2025

- **Documentation sur le projet**
- **Création du Gantt prévisionnel**
- **Début tests bouton poussoir** avec suppression des rebonds via code

---

Février 2025

- **TP sur télécommande infrarouge**
- **Réunion et revue de projet**
- **Tests IR et début essais avec ATtiny**

Mars 2025

- **Association de codes (Neopixels + bouton)**
- **Visualisation données IR avec TeraTerm**
- **Schémas sur KiCad**
- **Tests ATTiny / Arduino / ESP32**
- **Câblage sur Fritzing + tests communication série**

Avril 2025

- **Début et fin du câblage de la carte**
- **Préparation des livrables : dossier, diapo, nomenclature, fiche de maintenance**

Mai 2025

- **Essais de la carte (LEDs, communication avec TOM)**
- **Préparation à la revue 3**
- **Présentation de la revue finale**

Annexes Nathan PASCALE - ER 2