



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



BTS CIEL

Épreuve E6.2 – Session 2025

Dossier de revue

Projet Biathlon

Kevin Scherrer (IR-2)
Issa Bâ (IR-1)
Tomy Torrecillas (ER-4)
Benoît Gouley (ER-3)

Sommaire

Partie commune.....	3
1 - Présentation Générale.....	5
2 - Répartition des tâches.....	12
3 - Compte rendu de réunion.....	15
4 - Fiches recettes.....	17
Kévin Scherrer.....	1
Issa Bâ.....	1
Tomy Torrecillas.....	1
Benoît Gouley.....	1

Partie commune
Projet Biathlon

Partie commune..... 3

1 - Présentation Générale.....	5
1.1 - Contexte.....	5
1.2 - Cahier des charges.....	6
2 - Répartition des tâches.....	12
2.1 Annonce globale de la disposition du projet.....	12
2.2 Diagramme de Gantt prévisionnel.....	13
3 - Compte rendu de réunion.....	15
3.1 - Réunion du 14 janvier 2025 :.....	15
3.2 - Réunion du 12 mars 2025.....	16
4 - Fiches recettes.....	17

1 - Présentation Générale

1.1 - Contexte

Le Lycée Benoit (surtout le professeur de sport Mr.LeDuc) souhaite organiser une événement sportif qui serait une option au BAC inspiré du biathlon olympique (qui combine ski de fond et tir à la carabine), l'établissement adapte cette discipline pour un cadre scolaire, en remplaçant les éléments originaux par un pistolet laser et une cible, suite au désir de Mr.LeDuc on remplacera prochainement le pistolet par une carabine .

L'objectif est de proposer une activité jamais faite auparavant dans les lycées et alliant course d'endurance et épreuve de précision afin de favoriser le dépassement de soi. Ce projet s'inscrit dans une démarche éducative qui vise à sensibiliser les élèves au bienfaits du sport et tout en développant des compétences telles que la gestion du stress et la concentration

– La section EPS du lycée BENOIT propose pour le bac (à partir de 2025/2026) un biathlon composé de deux épreuves : Course de résistance, tirs sur cible à l'aide d'un pistolet laser.

L'équipement commercialisé étant trop onéreux (~4000€ pour 2 tirs simultanés), nous avons accepté de travailler à la réalisation technique de cette épreuve de tirs.--

1.2 - Cahier des charges

Il pourra y avoir entre 1 et 8 (paramétrable) coureurs simultanément.

Avant l'épreuve, le professeur (arbitre) paramètre le déroulement de la course et démarre la session (chronomètre).

L'épreuve commence par une course autour du stade (pas loin de 2 tours).

Arrivé au poste de tir, le coureur se munit d'un pistolet laser.

En cas de faute de la part du coureur pendant la session (virage coupé par exemple), le juge qui le suit peut lui attribuer une pénalité de 30 secondes (valeur paramétrée par l'arbitre avant le début de session).

Après concentration, cinq tirs (paramétrable) sont exécutés sur une cible personnelle.

Un signal lumineux permettra au coureur de savoir si la cible est atteinte.

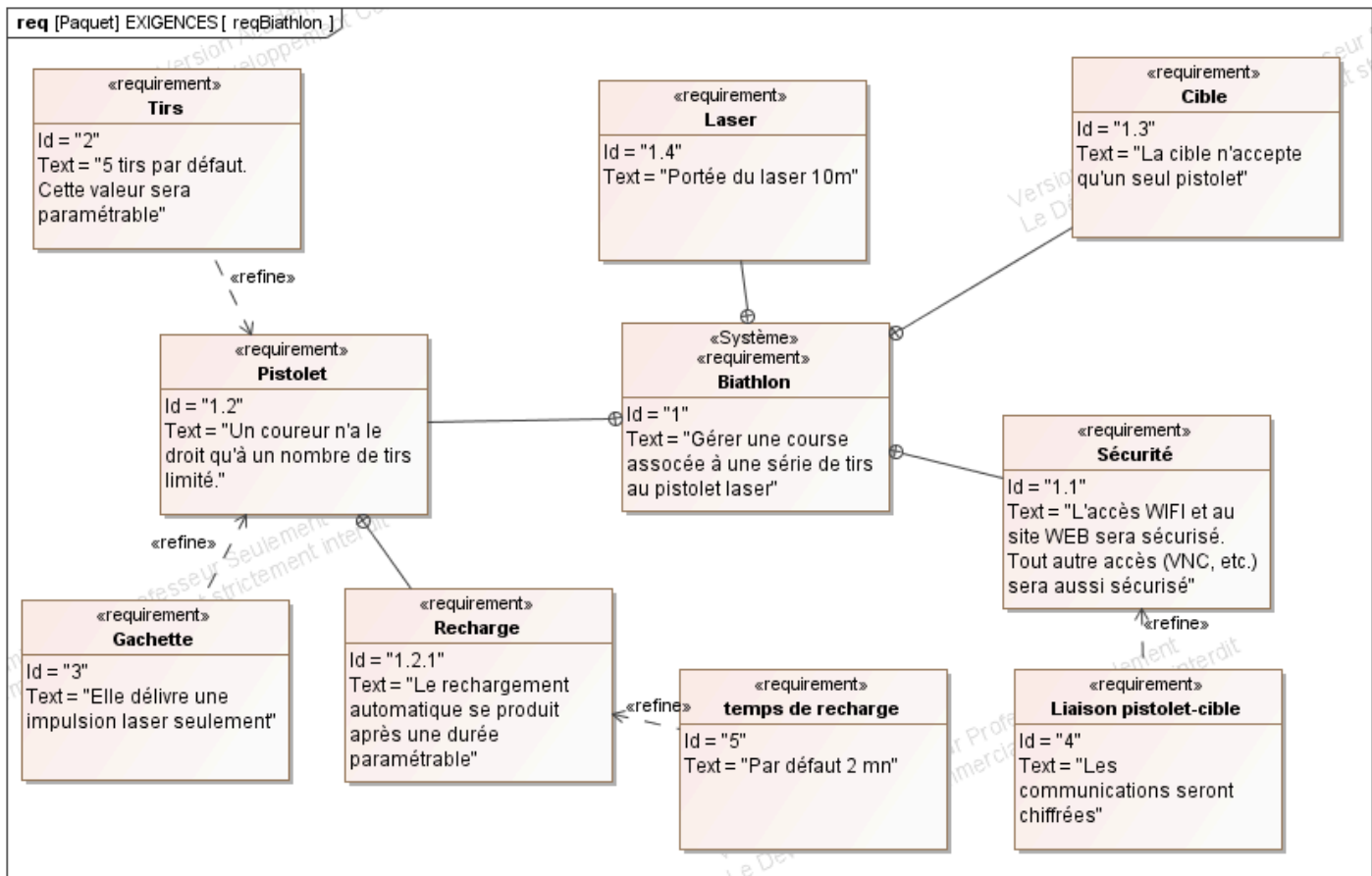
Le système affiche les temps aux tours avant passage au stand de tir, après, avec et sans pénalité et rend disponible le résultat pour le public (Page web public).

Lorsque tous les coureurs ont terminé, l'arbitre met fin à la course, et a la possibilité de sauver les résultats sur un support externe.

Initialement, nous étions soumis à différents types de contraintes :

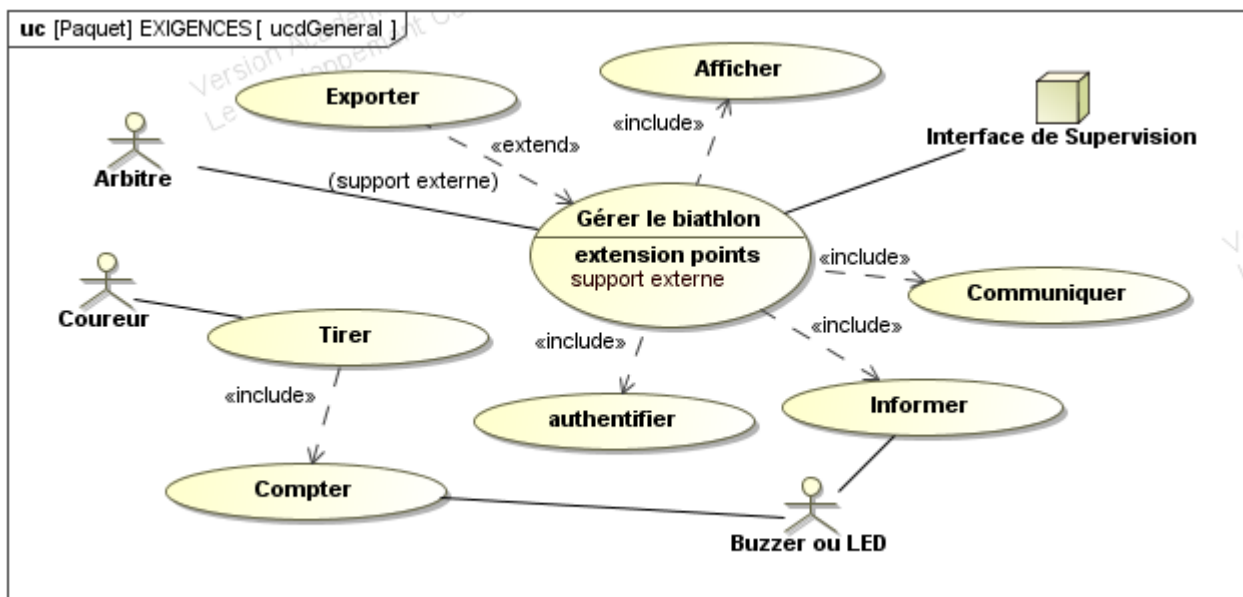
- Contraintes financières (budget alloué) : 500€ max
- Contraintes de développement (matériel et/ou logiciel imposé / technologies utilisées) :
- Contraintes qualité (conformité, délais, ...) : Les matériels pistolet et cible doivent ressembler à la réalité.
- Contraintes de fiabilité, cyber sécurité : Le système doit être protégé contre les attaques (WIFI), les données doivent être transportées avec sécurité et fiabilité en utilisant des protocoles chiffrés. Un pare-feu doit être paramétré pour sécuriser les communications.

Ci-dessous figure le diagramme SysML d'exigence :



Définissant ainsi les exigences du système et ses connexions, selon les états et transitions entre ces derniers.

Ci-dessous figure le diagramme SysML des cas d'utilisation :



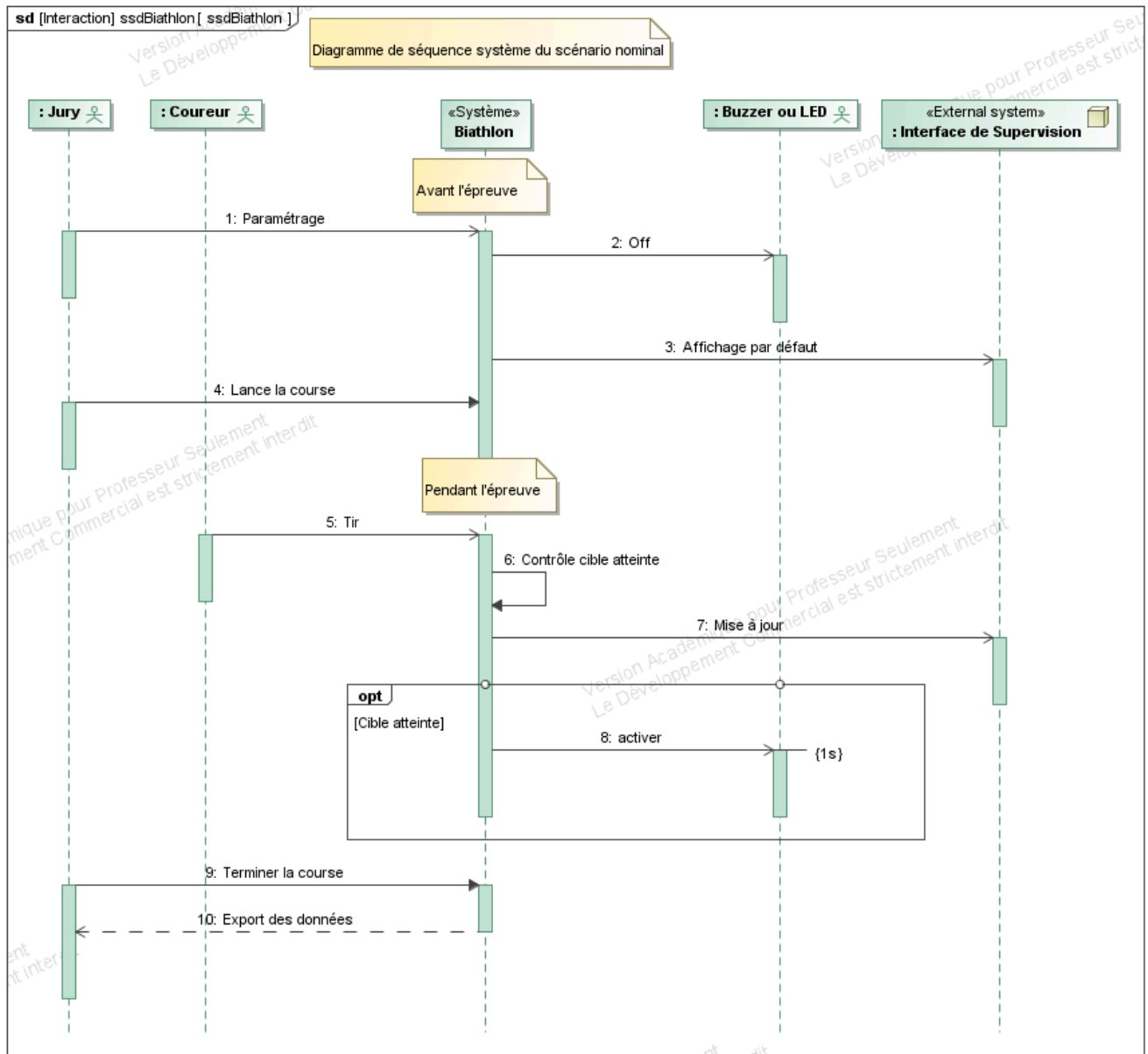
Montrant ainsi les interactions entre les acteurs externes (utilisateurs, capteurs, composants système) et le système.

Scénario des principaux cas d'utilisation :

Gérer le biathlon : Permet au professeur de s'authentifier pour paramétrer la session de tir, les coureurs, pour lancer le départ de la course. Il aura la possibilité d'exporter les résultats sur un support amovible. Pendant la course, le système communique avec les cibles pour afficher et mémoriser les tirs ayant atteint l'objectif.

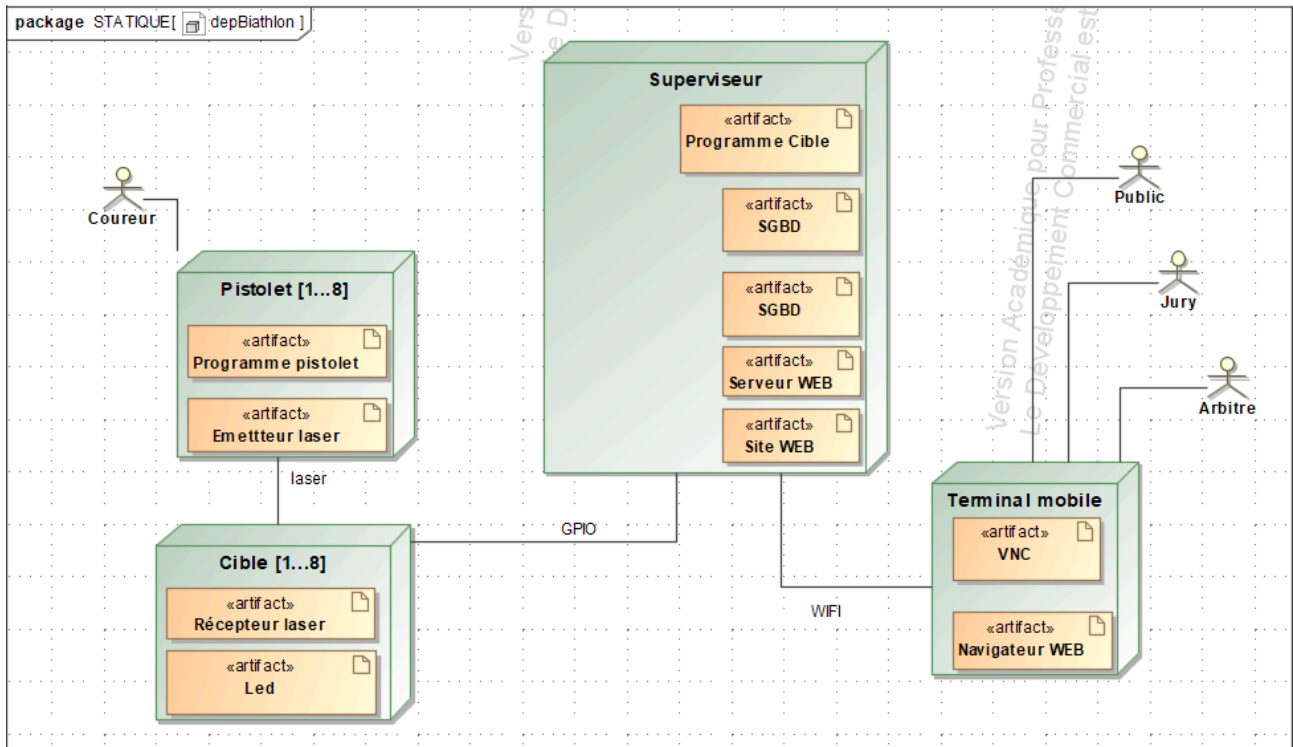
Tirer : Compte le nombre de tirs effectués par le coureur. Lorsque le nombre de tirs maximum est atteint, la gâchette se bloque pendant un certain temps, paramétré en amont (en fonction du mode entraînement ou épreuve). Passé ce temps, la LED alors rouge devient verte, les tirs peuvent continuer.

Ci-dessous figure le diagramme SysML de séquence :



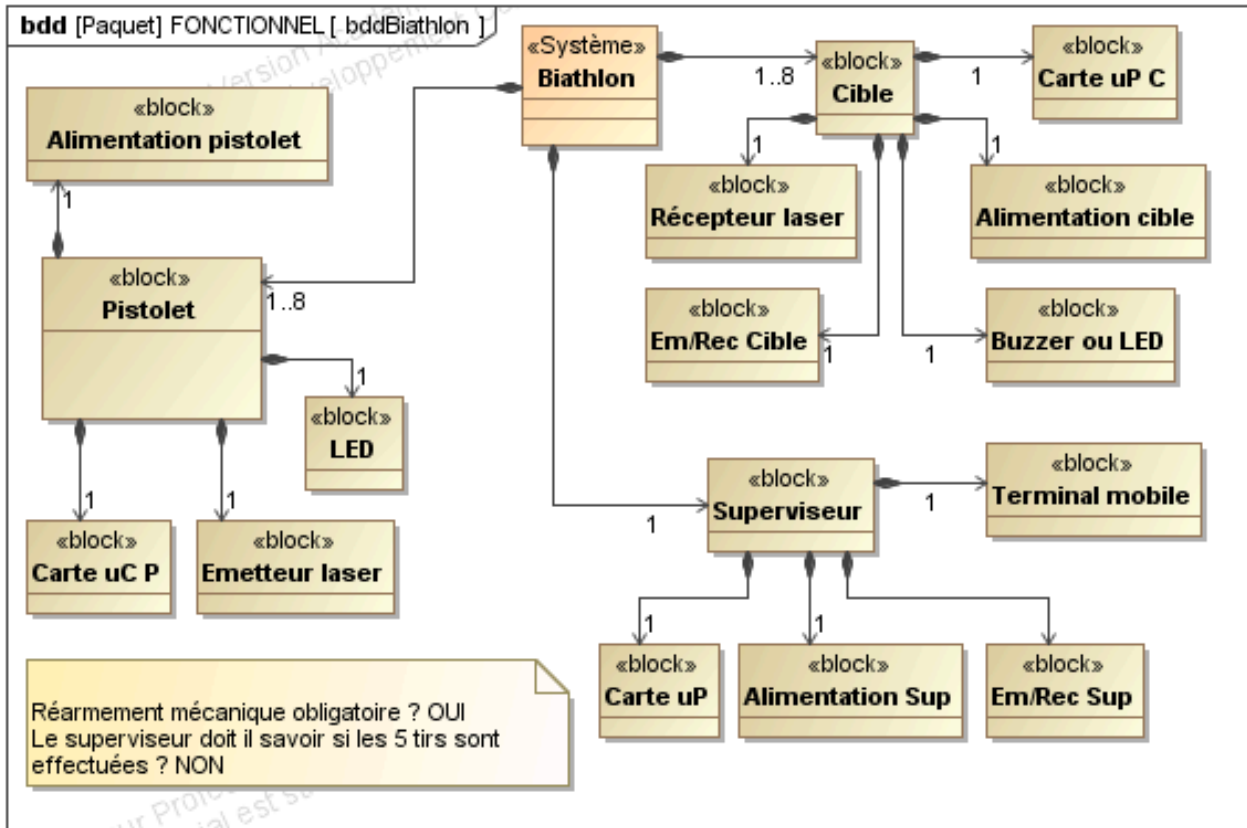
Montrant ainsi le déroulement des interactions entre les différentes parties du système de façon chronologique.

Ci-dessous figure le diagramme SysML de déploiement :



Montrant ainsi la répartition physique des différents éléments composant le système.

Ci-dessous figure le diagramme SysML des blocs :



Modélisant ainsi la structure des composants du système sous forme de blocs. Lesdits blocs peuvent inclure des sous-systèmes, des composants matériels ou logiciels et leurs relations.

2 - Répartition des tâches

2.1 Annonce globale de la disposition du projet

La partie électronique et informatique sont réparties en 2 parties, la structure des tâches a été déterminée de sorte que :

Issa Bâ ait pour tâche:

- D'installer l'environnement dédié au serveur, donc Raspberry pi.
- La communication pistolet/cible vers superviseur
- L'installation de l'AP WIFI et d'y connecter les appareils
- La fiche de recettes
- De réaliser le code source du superviseur (hormis la classe Bdd)
- De choisir le protocole et installer un serveur adapté, ici serveur TCP

Kévin Scherrer ait pour tâche :

- La partie Web permettant la visualisation prévu pour le public
- La partie Web permettant l'interaction avec la bdd depuis le site web (Partie Jury, donc 4 personnes et celle du prof qui a un accès administrateur)
- De mettre en place la base de données
- La réalisation de la classe Bdd, qui définira les méthodes concernées

Benoît Gouley ait pour tâche :

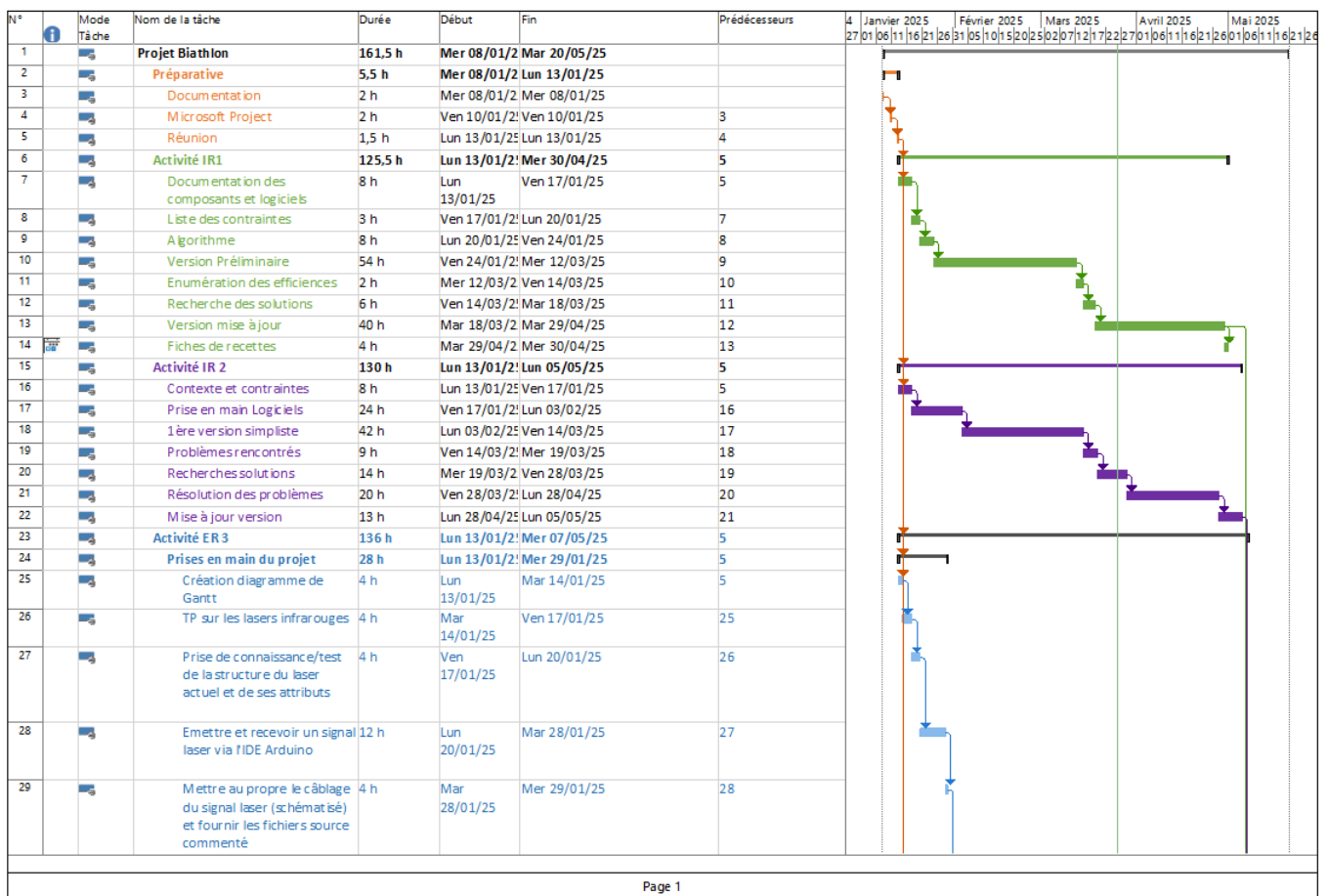
- Responsable technique de l'électronique embarquée du pistolet
- L'installation de l'environnement de développement compatible avec le microcontrôleur ESP-32 du pistolet
- Concevoir une structure permettant de générer un signal laser avec un codage sélectionnable
- De faire les essais sur la carte par câblage rapide avec le responsable de la cible
- De choisir la source et d'assurer l'alimentation autonome du pistolet
- Proposer un schéma structurel de l'ensemble du pistolet
- Concevoir un circuit imprimé compatible avec le boîtier du pistolet

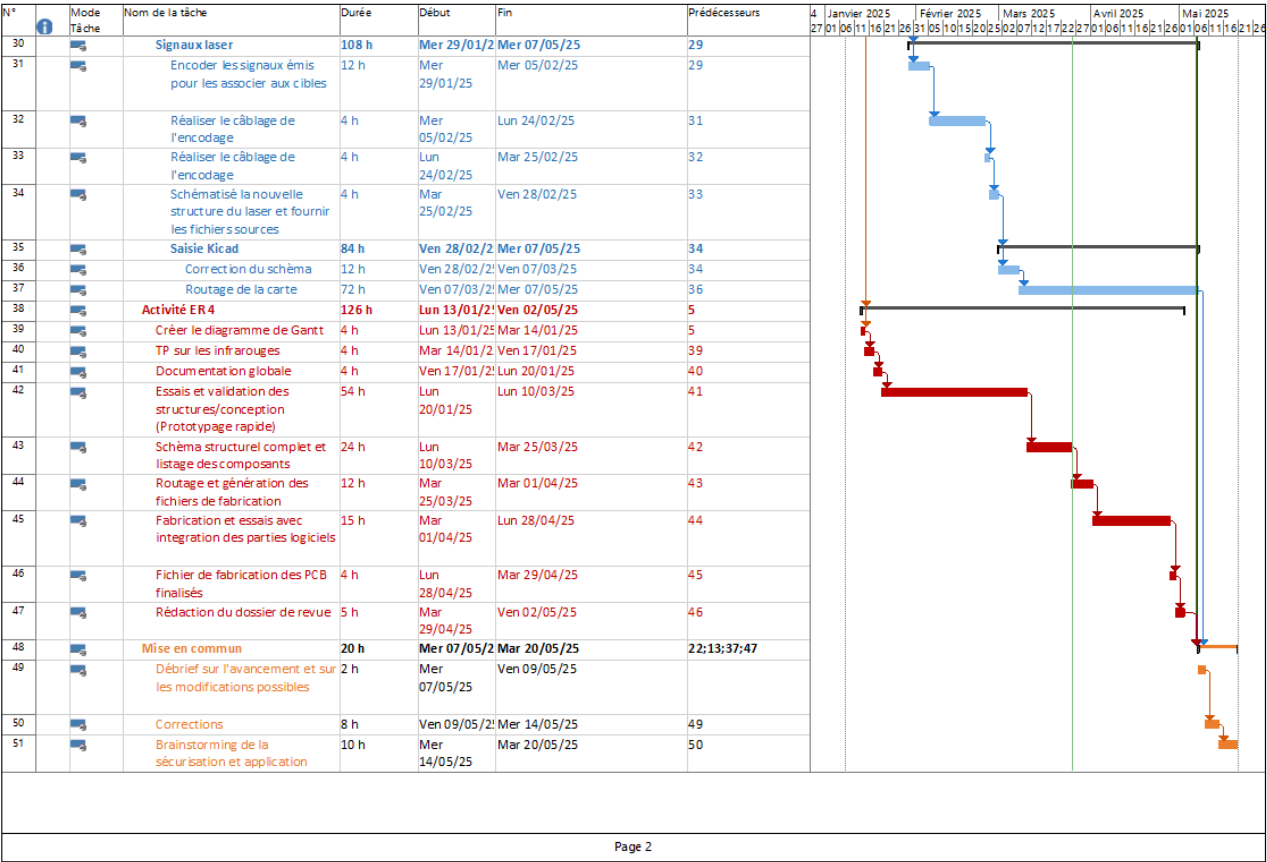
Tomy Torrecillas ait pour tâche :

- Responsable technique de l'électronique embarquée de la cible Concevoir une structure permettant de détecter l'impact d'un signal laser avec un codage sélectionnable
- De faire les essais sur la carte par câblage rapide avec le responsable du pistolet
- Concevoir un circuit imprimé ayant un format identique à celui des cibles commercialisées

2.2 Diagramme de Gantt prévisionnel

Ci-joint sous forme de diagramme, les tâches et le temps que l'on prévoit.
Dans les annexes, nous y avons inséré un diagramme mis à jour de sorte à avoir un contraste entre les attentes, d'avant le projet entamé et la réalité des imprévus rencontrés.





Peu de temps après le début du projet nous nous sommes aperçu de la vision biaisée que l'on avait des tâches à faire, nous les précisons dans nos parties personnelles.

3 - Compte rendu de réunion

3.1 - Réunion du 14 janvier 2025 :

Règles instaurées :

L'épreuve est faite pour 5 candidats simultanés.

Le participant prend place à la zone de départ et arrivé, lorsque le feu vert est donné, il part et complète deux tours de terrain avant de s'insérer sur le parcours de tir indiqué par des plots. Cette course est chronométrée sous le nom de T1, il effectue 5 tirs, chaque tir raté vaut pénalité. La pénalité est une course supplémentaire sur un autre parcours (voir figure 1).

Le temps de tir est enregistré sous T2, celui du tir+pénalités sous T3. On peut donc calculer le temps total avec l'addition de T1 et T3. L'élève répète cette épreuve une deuxième fois.

L'adresse du site se termine soit par /biathlon/ juge , /arbitre ou /public pour une recherche simplifiée.

Modèles Logique / Conceptuel de Données

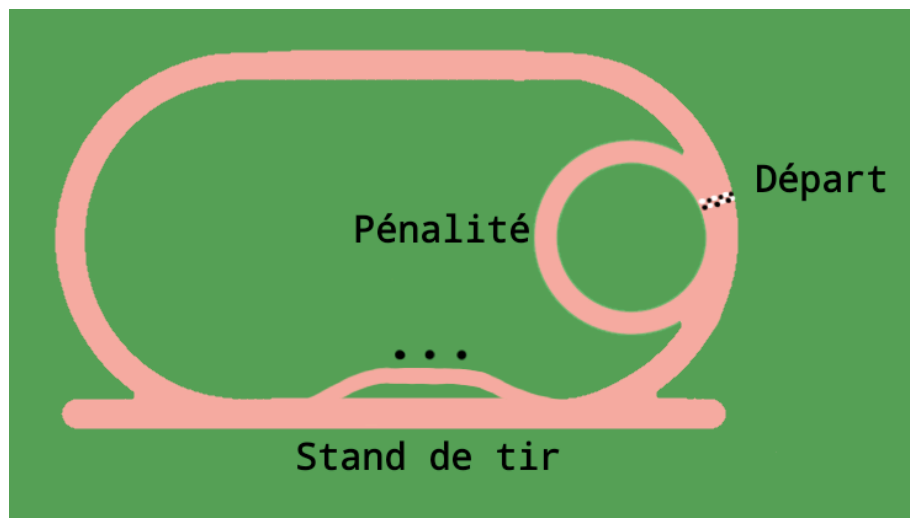


Figure 1

3.2 - Réunion du 12 mars 2025

Pour l'interface IHM on gardera trois classes, Public, Juge et Arbitre.

L'arbitre configure la course et la lance après avoir attendu que les juges rejoignent la session, il pourra aussi générer un tableau sous format .CSV en fin de course, qui lui permettra un affichage des résultats sous forme simplifiée.

Le juge se connectera à la session de l'arbitre, il peut ne pas y avoir de juge, c'est pour cela que la configuration "juge" ne sera activée seulement si un système se connecte en tant que tel. Les juges ont accès au tableau des scores mais ils peuvent surtout rajouter des pénalités de temps sur les participants si souhaité.

La classe public ne possède qu'un accès au tableau global.

Le design devra être simple d'utilisation et réactif aux manipulations des utilisateurs pour permettre une bonne expérience.

4 - Fiches recettes

4.1 - Fiches Kévin Scherrer (IR)

Fiche recette 1 : Arbitre

FICHE RECETTE, QUALIFICATION DU SYSTÈME

Nom du système :	Projet Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur fiche :	Kévin Scherrer	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de la partie « Arbitre »
Objectif du scénario :	Vérifier que l'arbitre a bien capacité à s'authentifier, paramétrer la course et la gérer (démarrer, interrompre, terminer).

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

S'assurer la connectivité au point d'accès Wi-Fi du système à partir d'un terminal mobile (tablette ou téléphone portable). À l'aide d'un navigateur dédié, se connecter via l'URL « www.biathlon.lab/arbitre/ »

EXÉCUTION DU TEST

Description courte :	L'arbitre doit pouvoir : - S'authentifier - Paramétrer le nombre de coureurs - Lancer la course une fois les juges connectés - Suivre l'évolution de la course - Terminer la course (générer un fichier .csv) - Interrompre la course à tout moment (remise à zéro)
Résultats attendus :	La base de données est mise à jour en conséquence. Le fichier .csv se trouve dans le dossier « /home/pi/Downloads/ »

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

Fiche recette 2 : Juge

FICHE RECETTE, QUALIFICATION DU SYSTÈME

Nom du système :	Projet Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur fiche :	Kévin Scherrer	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de la partie « Juge »
Objectif du scénario :	Vérifier que le juge a bien capacité à s'authentifier, gérer les temps et les pénalités.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

S'assurer la connectivité au point d'accès Wi-Fi du système à partir d'un terminal mobile (tablette ou téléphone portable). À l'aide d'un navigateur dédié, se connecter via l'URL « www.biathlon.lab/juge/ »

EXÉCUTION DU TEST

Description courte :	Le juge doit pouvoir : - S'authentifier - Entrer le nom du coureur suivi - Chronométrer les temps au tour - Gérer les pénalités en fonction des tirs ratés du coureur
Résultats attendus :	La base de données est mise à jour en conséquence.

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

Fiche recette 3 : Public

FICHE RECETTE, QUALIFICATION DU SYSTÈME

Nom du système :	Projet Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur fiche :	Kévin Scherrer	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de la partie « Public »
Objectif du scénario :	Vérifier que le tableau s'actualise dynamiquement.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

S'assurer la connectivité au point d'accès Wi-Fi du système à partir d'un terminal mobile (tablette ou téléphone portable). À l'aide d'un navigateur dédié, se connecter via l'URL « www.biathlon.lab/public/ »

EXÉCUTION DU TEST

Description courte :	Modification de la base de données manuellement via phpMyAdmin.
Résultats attendus :	La page réagit, actualisant le tableau.

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

4.2 - Fiches Issa Bâ (IR)

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur de la fiche :	Issa Bâ	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de tir réussi en mode entraînement.
Objectif du scénario :	Que la cible allume la led correspondante au tir en vert.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

Il faudra que la connectivité entre les appareils soit opérationnelle et que la course soit initialisée.
--

EXÉCUTION DU TEST

Description courte :	Après la mise en place de tous les dispositifs, une séquence de tir réussie est effectuée.
Résultats attendus :	La base de données n'a pas été changée et la cible a mis la bonne led en vert.

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

Fiche recette 1 : tir réussi en mode entraînement

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur de la fiche :	Issa Bâ	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de tir réussi en mode examen.
Objectif du scénario :	Que la cible allume la led correspondante au tir en vert et que le suivi des tirs soit enregistré dans la base de données.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

Il faudra que la connectivité entre les appareils soit opérationnelle et que la course soit initialisée.
--

EXÉCUTION DU TEST

Description courte :	Après la mise en place de tous les dispositifs, une séquence de tir réussie est effectuée.
Résultats attendus :	La base de données a bien été changée et la cible a mis la bonne led en vert.

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

Fiche recette 2 : tir réussi en mode examen

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur de la fiche :	Issa Bâ	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de tir raté en mode entraînement.
Objectif du scénario :	Que la cible allume la led correspondante au tir en rouge.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

Il faudra que la connectivité entre les appareils soit opérationnelle et que la course soit initialisée.

EXÉCUTION DU TEST

Description courte :	Après la mise en place de tous les dispositifs, une séquence de tir raté est effectuée.
Résultats attendus :	La base de données n'a pas été changée et la cible a mis la bonne led en rouge.

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

Fiche recette 3 : tir raté en mode entraînement

FICHE RECETTE, QUALIFICATION DU SYSTEME

Nom du système :	Biathlon	Code de la campagne de test :	Biathlon_001
Concepteur de la fiche :	Issa Bâ	Date :	10/06/25

IDENTIFICATION DU SCENARIO

Identification du scénario de recette	
Titre :	Test de tir raté en mode examen
Objectif du scénario :	Que la cible allume la led correspondante au tir en rouge et que le suivi des tirs soit enregistré dans la base de données.

CONDITIONS INITIALES NÉCESSAIRES POUR EFFECTUER LA RECETTE

Il faudra que la connectivité entre les appareils soit opérationnelle et que la course soit initialisée.

EXÉCUTION DU TEST

Description courte :	Après la mise en place de tous les dispositifs, une séquence de tir ratée est effectuée.
Résultats attendus :	La base de données a bien été changée et la cible a mis la bonne led en rouge.

BILAN

--

REMARQUES

--

CONCLUSION

VALIDE		NON VALIDE	
--------	--	------------	--

(Mettre une croix dans la case correspondante)

Fiche recette 4 : tir raté en mode examen

5 - Audit de sécurité

Fiche d'audit de sécurité.		Réf. 2025-1
Projet	Auteur	Date
Projet Biathlon	Kévin Scherrer, Issa Bâ, Tomy Torrecillas et Benoît Gouley	28/05/25
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 Mot de passe modifiable
SGBD	Attaque pour trouver le mot de passe admin ou autre Piratage des données	Mot de passe complexe du SGBD. Chiffrage complexe du mot de passe.
WIFI	Déchiffrer le mot de passe	Utilisation de Wi-Fi Protected Access 2 (WPA2) Chiffrement AES
Carabine et cible	Possibilité de saboter le score d'un participant avec un autre laser	Laser contenant une trame spécifique
Synthèse		
L'audit de sécurité du projet Biathlon met en évidence plusieurs risques sur les plans web, réseau, base de données et matériel. Des solutions adaptées ont été proposées pour chaque point sensible : sécurisation HTTPS, protection des formulaires, mots de passe robustes, chiffrement AES, et authentification des dispositifs de tir. Ces mesures permettent de renforcer la fiabilité et l'intégrité du système dans son ensemble.		

Kévin Scherrer

Partie Personnelle - Kevin Scherrer

Kévin Scherrer	1
1 - Rôle et objectifs	3
2 - Diagrammes	4
2.1 - Diagramme de Gantt	4
2.2 - Diagrammes SysML	6
2.2.1 - Cas d'utilisation	6
2.2.2 - Exigences	7
2.2.3 - Blocs	8
2.2.4 - Déploiement	8
2.2.5 - Séquence	9
3 - Interface WEB Utilisateur	10
3.1 - Espace Public	10
3.2 - Espace Juge	12
3.3 - Espace Master	16
4 - Scripts	20
4.1 - Partie Master	20
4.2 - Partie Juges	21
4.3 - Partie Public	21
5 - Constitution BDD	22
5.1 - Modélisation	22
5.2 - Code Source BDD	23
5.3 - Interface Graphique	27
6 - Sécurisation	29
6.1 - Passer du http au https	29
6.2 - Installation pare-feu	29

1 - Rôle et objectifs

Ici je vais expliquer mes missions principales plus en détail. Ces dernières se scindent en 2, à savoir :

- Mise en place de la Base De Données (souvent appelée BDD)
- Conception de l’affichage WEB (public, utilisateur, administrateur)

Il y figure également une tâche transversale de gestion de la communication entre les 2 entités susnommées.

J’occupe également un rôle d’assistance au codage, conjointement à la partie de mon confrère IR1 (Issa) dont la partie personnelle se trouve plus bas.

L’objectif étant de pouvoir accéder aux résultats de la session en cours tant qu’elle est active, la Base de Données joue donc un rôle administratif aussi bien que public. Le code se trouve donc ici être le lien sécurisé entre la BDD et l’affichage public des informations de session en cours.

Le code et la base de données sont hébergés sur une Raspberry Pi 3, comme exigé par le cahier des charges.

La Raspberry a, depuis, rendu l’âme (Carte SD hors service), emportant 2 mois de travail avec elle.

La carte SD a donc été refaite depuis zéro, ce qui a pris 2 semaines entre réflexion et exécution de notre plan d’action.

Initialement tournait sur la Raspberry un serveur LAMP (Linux, Apache, MariaDB, PHP) pour pouvoir y installer la Base de Données et y accéder depuis n’importe quel terminal distant ; Ordinateur pour continuer le développement, ou Téléphone portable pour s’y connecter dans le cadre d’une situation réelle du projet.

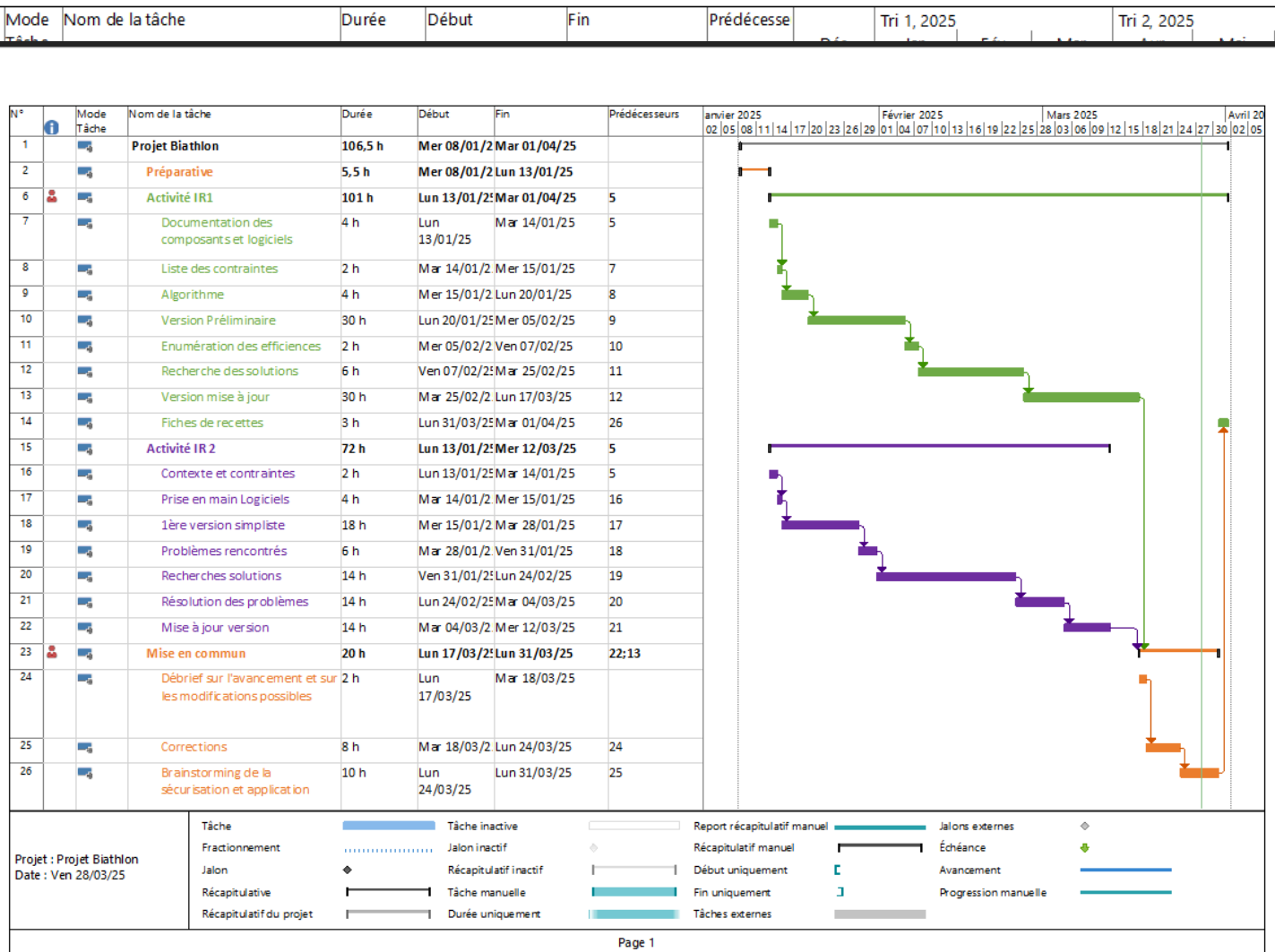
Le code est donc logiquement écrit en php, pour assurer les appels inter-fichier et la communication avec la Base de Données.

2 - Diagrammes

2.1 - Diagramme de Gantt

À l'origine, le diagramme pour ma partie était planifié ainsi :

À mesure que le projet avance, le diagramme de Gantt ressemble désormais à ceci :

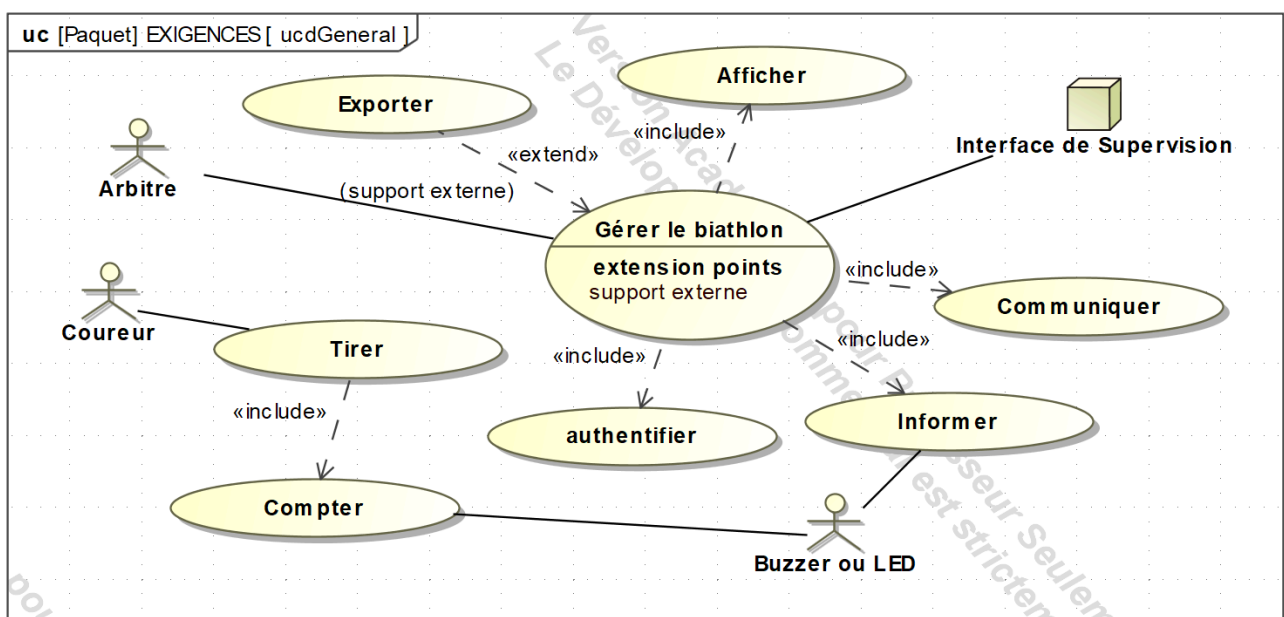


Les différentes tâches composant ce diagramme de Gantt vont être explorées successivement et de façon détaillée tout au long de la lecture du dossier.

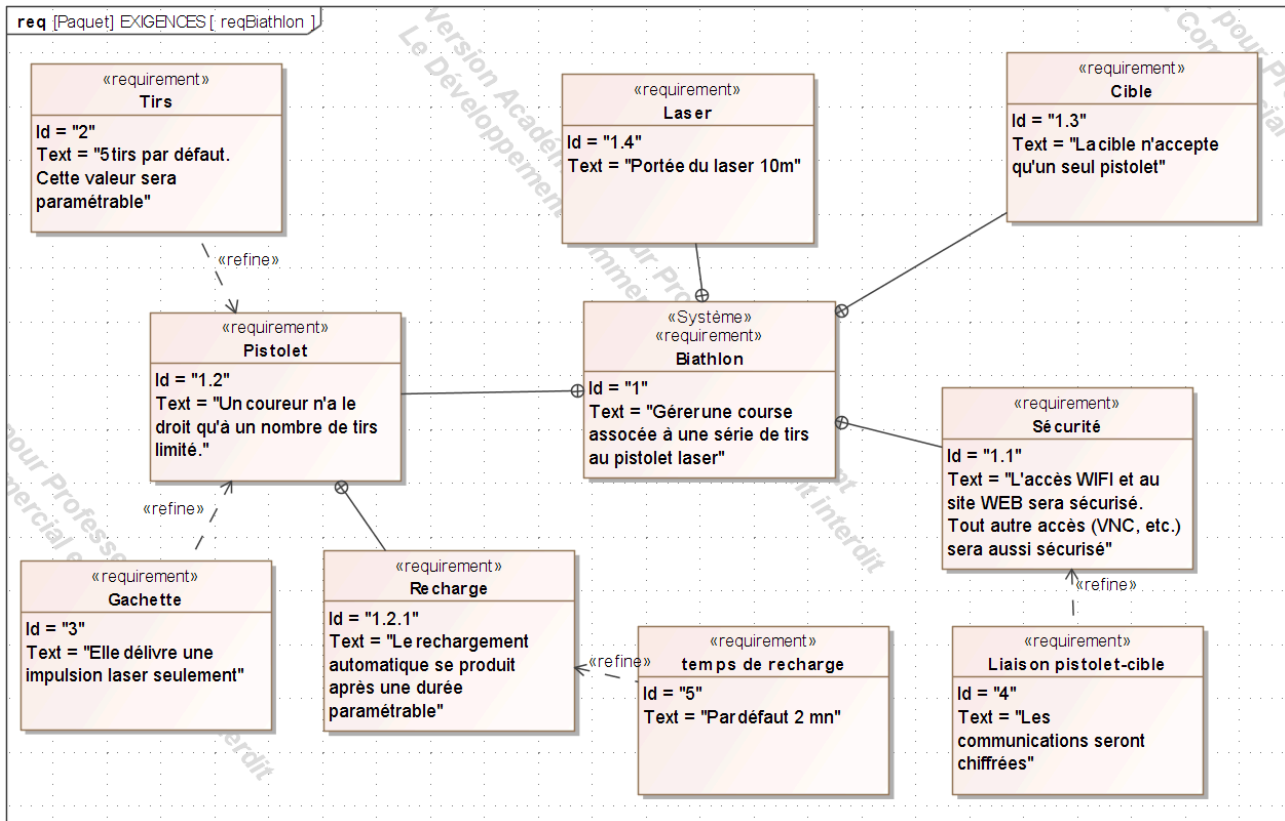
2.2 - Diagrammes SysML

Ici-bas figurent les différents diagrammes SysML nous permettant de définir les cas d'utilisation, exigences, interactions entre les blocs du système, ainsi que le déploiement de la séquence d'épreuve.

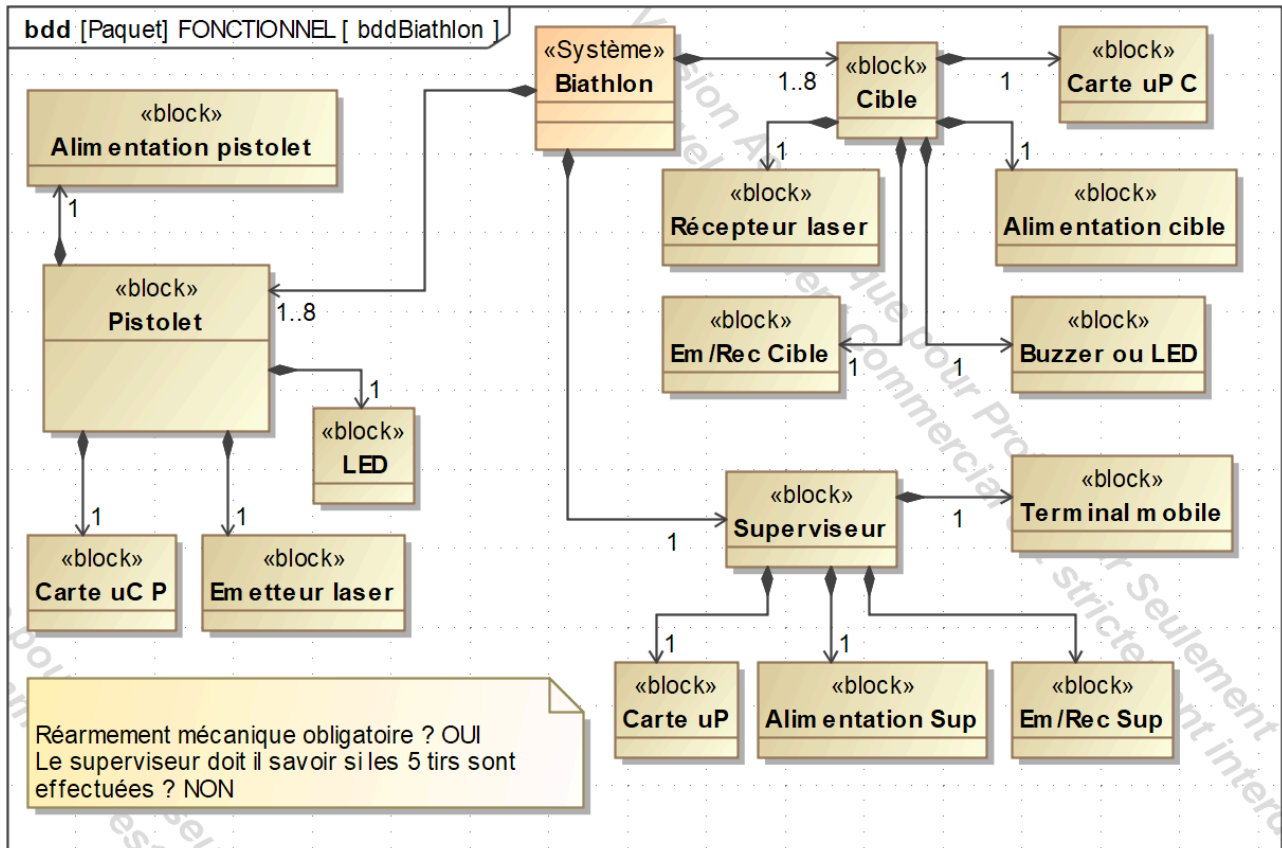
2.2.1 - Cas d'utilisation



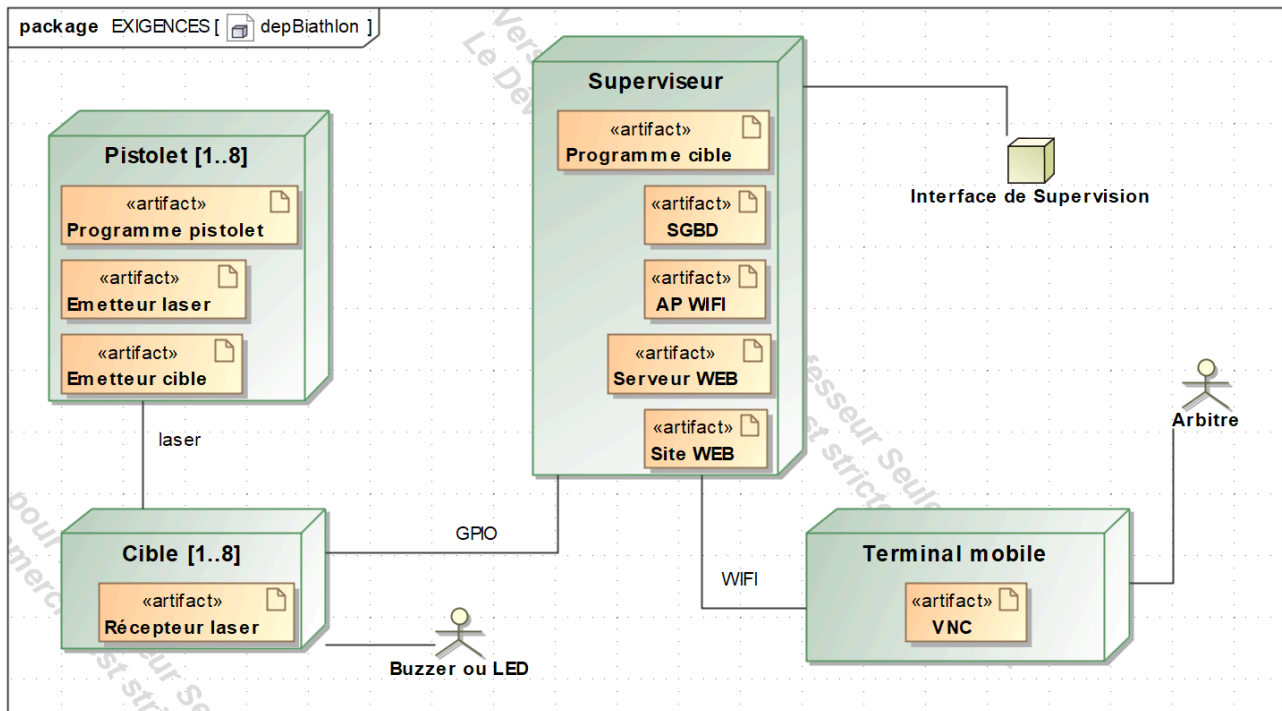
2.2.2 - Exigences



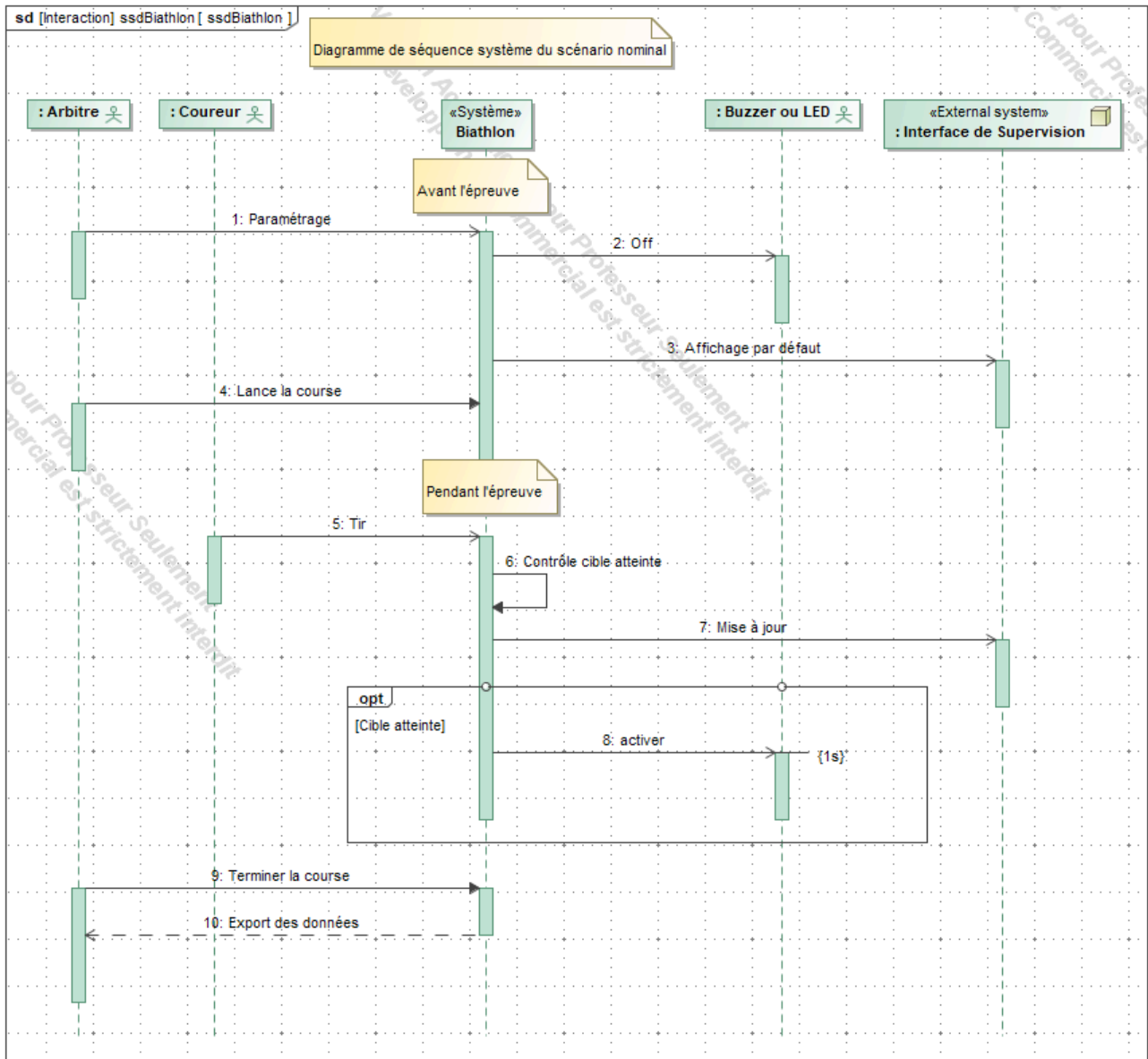
2.2.3 - Blocs



2.2.4 - Déploiement



2.2.5 - Séquence



Afin de constituer le Modèle Logique de Données (MLD), il a fallu créer des maquettes d'interface utilisateur graphique. C'est à cette fin là que j'ai utilisé Mockplus.

J'ai apporté depuis, quelques modifications aux maquettes initialement fournies par le professeur (M. ANTOINE).

Ci-dessous, l'interface publique lorsqu'une course n'est pas active, accessible via l'adresse : www.biathlon.fr/public

BIATHLON SUIVI DE COURSE PUBLIC

Num	Coureur	2T1	ST1	TP1	2T2	ST2	TP2	2T3	ST3	TP3	Total
Pas de course en cours !											
Temps au format min:sec:cent. 2T : 2 tours de stade. ST : Séquence de 5 tirs. TP : Tours de pénalité.											

<

Il s'agit d'un simple tableau affichant les données de la base, il est actuellement vide car il n'y a pas de course en cours.

BIATHLON SUIVI DE COURSE PUBLIC											
Num	Coureur	2T1	ST1	TP1	2T2	ST2	TP2	2T3	ST3	TP3	Total
1	Bianca	00:26:36 (dt=00:26:36)	00:36:79 (dt=00:10:43) 0/0	00:44:91 (dt=00:08:11)	01:08:83 (dt=00:23:92)	01:21:23 (dt=00:12:39) 0/0	01:23:45 (dt=00:02:22)	01:25:58 (dt=00:02:12)	01:30:26 (dt=00:04:68) 0/0	01:40:87 (dt=00:10:61)	01:40:87
Temps au format min:sec:cent. 2T : 2 tours de stade. ST : Séquence de 5 tirs. TP : Tours de pénalité.											
© 2025 - Biathlon Supervision System											

Il s’agit là d’un simple tableau affichant les temps et les vitesses des étapes de course en cours.

3.2 - Espace Juge

Ci-dessous, l'écran auquel ont accès les juges à l'adresse : www.biathlon.fr/juge



Ici figure l'interface des juges, qui supervisent les coureurs, lorsqu'aucune course n'est en cours.

BIATHLON PARAMETRES JUGE

Votre nom (juge) :

Nom du coureur :

Sauver et attendre le départ

Définissez les paramètres...

© 2025 - Biathlon Supervision System

Voici la page de login pour les juges, toujours à la même adresse.
Après avoir entré ces informations, ils se retrouvent en état d'attente du master.

BIATHLON PARAMETRES JUGE

Votre nom (juge) :

Bernard

Nom du coureur :

Bianca



Attente du démarrage de la course
Vous êtes le juge 1 !

© 2025 - Biathlon Supervision System

Le juge ci-dessus attend le départ de la course, signal envoyé par le master via son interface.

Une fois la course démarrée, la supervision des juges commence, ils font donc leur relevés de temps chronométrés.

BIATHLON SUIVI COURSE JUGE

Nom de la course : TG4_mardi
Nombre de coureurs/juges : 1

Bernard est le juge 1 de Bianca

TEMPS

Course terminée !

Num	Coureur	2T1	ST1	TP1	2T2	ST2	TP2	2T3	ST3	TP3	Total
1	Bianca	00:26:36 (dt=00:26:36)	00:36:79 (dt=00:10:43) 0/0	00:44:91 (dt=00:08:11)	01:08:83 (dt=00:23:92)	01:21:23 (dt=00:12:39) 0/0	01:23:45 (dt=00:02:22)	01:25:58 (dt=00:02:12)	01:30:26 (dt=00:04:68) 0/0	01:40:87 (dt=00:10:61)	01:40:87

Temps au format min:sec:cent. 2T : 2 tours de stade. ST : Séquence de 5 tirs. TP : Tours de pénalité.

© 2025 - Biathlon Supervision System

Une fois la course terminée, on revient à la page d’attente du master.

3.3 - Espace Master

Enfin, l'interface d'authentification pour le professeur d'EPS (M. LEDUC) :

www.biathlon.fr/master

BIATHLON AUTHENTICATION MASTER v1.6 by PhA (2025)

Vérification du Code

Entrez un code à 6 chiffres :

Commencer

© 2025 - Biathlon Supervision System

Après authentification du master, il arrive sur sa page de configuration de la course.

BIATHLON PARAMETRES MASTER

Nom course :

Nombre de juge :

Sauver et attendre les juges

RAZ

Définissez les paramètres...

Fichiers CSV disponibles

[biathlon_2025-05-20_21-40-11.csv](#)
[biathlon_2025-05-20_23-46-31.csv](#)
[biathlon_2025-05-21_16-23-58.csv](#)
[biathlon_2025-05-21_16-36-31.csv](#)
[biathlon_2025-05-21_21-57-00.csv](#)
[biathlon_2025-05-22_10-29-05.csv](#)

Purger les anciens fichiers (>1an)

© 2025 - Biathlon Supervision System

Sur l'interface ci-dessus, le master peut entrer bon nombre d'informations, telles que le nom de la course, le nombre de juges, la distance du terrain, ou avoir accès aux archives (fichiers .csv) des dernières courses effectuées. Une fois toutes les données nécessaires au lancement de la course communiquées à l'application, il ne reste plus qu'à la lancer.

BIATHLON LANCEMENT COURSE MASTER

Nom de la course : TG4_mardi
Nombre de coureurs/juges : 1

Bernard est le juge 1 de Bianca

GO RAZ

Cliquez pour démarrer la course !

© 2025 - Biathlon Supervision System

Cet écran est un récapitulatif afin de s'assurer que tout est bon avant le départ.

BIATHLON SUIVI COURSE MASTER

Nom de la course : **TG4_mardi**
 Nombre de coureurs/juges : **1**

 Bernard est le juge 1 de Bianca

FIN RAZ

Course en cours !

Num	Coureur	2T1	ST1	TP1	2T2	ST2	TP2	2T3	ST3	TP3	Total
1	Bianca	00:26:36 (dt=00:26:36)	00:36:79 (dt=00:10:43) 0/0	00:44:91 (dt=00:08:11)	01:08:83 (dt=00:23:92)	01:21:23 (dt=00:12:39) 0/0	01:23:45 (dt=00:02:22)	01:25:58 (dt=00:02:12)	01:30:26 (dt=00:04:68) 0/0	01:40:87 (dt=00:10:61)	01:40:87

Temps au format min.sec.cent. 2T : 2 tours de stade. ST : Séquence de 5 tirs. TP : Tours de pénalité.

© 2025 - Biathlon Supervision System

Sur cet écran, le master surveille que les juges font bien leur travail de supervision et relevés de chronomètres.

192.168.6.35 indique

Es-tu sûr de vouloir terminer la course ?
 (un fichier csv sera généré dans /biathlon/res)

OK
Annuler

Pop-up qui demande confirmation de fin de course au master, et lui rappelle qu'un fichier .csv sera créé à l'issue de la terminaison de cette course (fichiers accessibles à tout moment, dès lors que la Raspberry Pi 3 est alimentée).

Voici donc le dispositif par lequel le professeur d'EPS chapeaute le déroulement de son cours.

4 - Scripts

Ici figurent les fichiers (codes php) qui généreront les liens et communications entre les différentes pages de l'interface utilisateur présentées en détail dans la partie précédente.

Certains scripts php ne sont dédiés qu'à une fonction spécifique, ou ne sont là que pour interagir avec la base de données en arrière-plan, ici, raz.php :

```
<?php
    require 'cbdd.php';
    session_start();
    $db->setState(0);
    $db->setNbJuges(0);
    $db->viderTableRace();
    $db->setTokenMaster("0");
    header("Location: logout.php");
?>
```

Ce script a pour intérêt principal de remettre à zéro l'état de la base de données.

Par exemple, le fichier cbdd.php contient la classe Cbdd chargée d'interagir directement sur la base de données avec des commandes et fonctions privées ou publiques, parmi lesquelles on retrouve une façon d'exécuter les commandes SQL basiques telles que update, select ou delete.

4.1 - Partie Master

Les scripts qui iront interagir avec la base de données modifient bon nombre de valeurs pour renvoyer le script correspondant à l'état de la base de données (soit masterX.php avec le X prenant les valeurs 0, 1, 2 ou 3).

Cet état est un paramètre de la base de données à utiliser, en tant qu'administrateur, pour que l'application renvoie la page correspondante à l'état de la base.

Si l'état de la base affiche '0', c'est que l'arbitre n'est pas encore connecté. Juges et public ne peuvent donc pas s'authentifier, pas avant l'arbitre, ainsi que sa configuration de la course.

Si l'état de la base affiche '1', l'arbitre s'est authentifié et est en attente de l'authentification des juges, le master attend la connexion de tous les juges, le nombre de juges est configuré par l'arbitre jusqu'à 8 maximum (avec leurs paramètres associés : nom et prénom du juge, et nom prénom du coureur suivi).

Si l'état de la base affiche '2', les juges sont tous connectés, le master peut alors démarrer la session de course.

Si l'état de la base affiche '3', la course est lancée par le master, les juges suivent alors leurs coureurs associés et le public peut suivre le tableau des scores et temps.

4.2 - Partie Juges

Même schéma ici, les pages juges sont soumises aux mêmes changements que ceux de la partie master.

Si la base affiche '1', les juges ne peuvent pas encore s'authentifier, la page reste en état de chargement jusqu'à connexion du master.

Si la base affiche '2', les juges peuvent s'authentifier et entrer ensuite les noms des coureurs qu'ils supervisent.

Si la base affiche '3', la course est lancée, les juges n'ont plus qu'à occuper leur fonction de prise de mesure de scores et temps.

4.3 - Partie Public

Le fichier public.php quant à lui, renvoie simplement le tableau des scores et temps, ce même tableau qui est affiché sur les états finaux de pages master (3) et juge (3).

5 - Constitution BDD

5.1 - Modélisation

Tout d'abord, avant de constituer la Base de Données, il est important de dresser un plan d'attaque, j'ai nommé : le Modèle Conceptuel de Données. Le MCD doit répondre aux critères suivants :

Master (le professeur)

- Authentification
- Nom Course
- Nombre de Juges
- Autoriser la connexion des juges
- Lancer la course

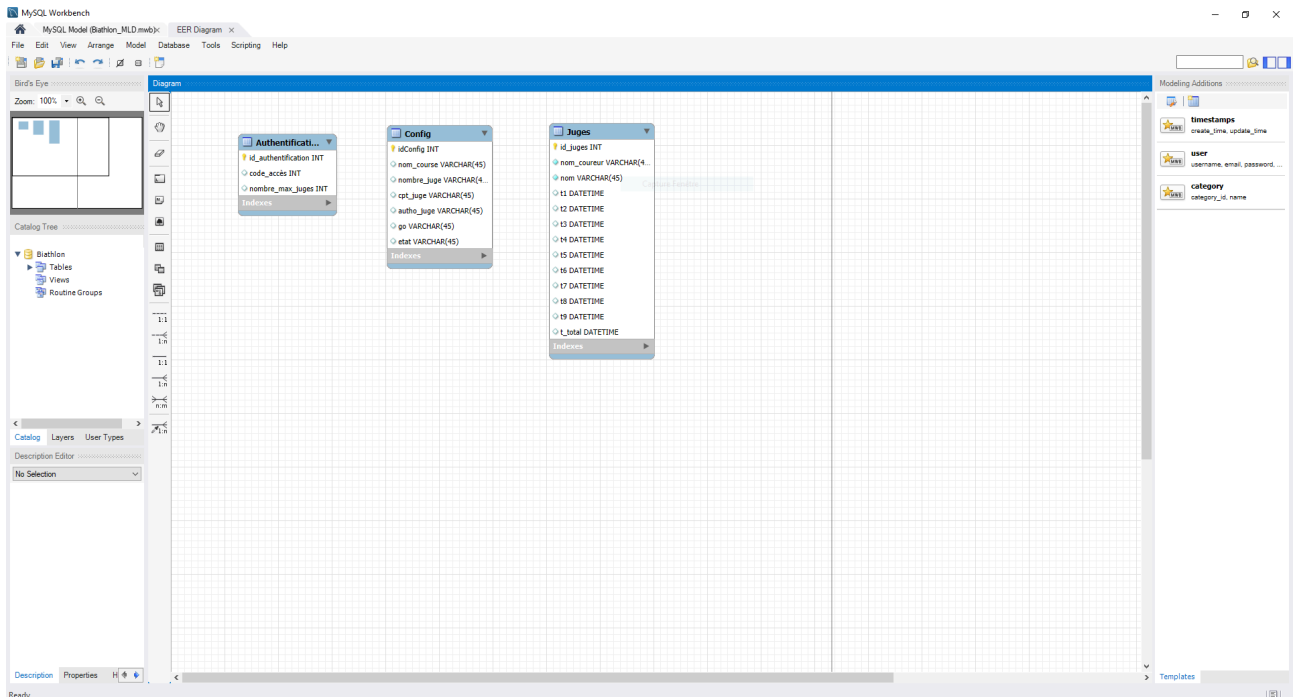
Juges (les élèves qui suivent les coureurs)

- Étapes de course de chaque coureur (9 temps)
 - Nom du juge
 - Nom du coureur
- N° du juge (automatique)
- État course

Public (les élèves qui courent pas)

- Temps (intermédiaires, avant et après pénalité éventuelle)
 - Scores (tirs réussis, sur 5)
- Noms coureurs et juges associés

Tout en suivant cette petite liste de critères, il faut modéliser cette topologie, ce que j'ai fait avec MySQL Workbench.



Le but ici est de tracer la topologie et les liens et communications entre les tables.

5.2 - Code Source BDD

Le fichier setBdd.inc.php contient les informations de login ci-dessous :

```
<?php
    $host = 'localhost';
    $bddname = 'biathlon';
    $username = 'biathlon';
    $password = 'biathlon';
?>
```

Le fichier biathlon.sql ci-dessous contient l'entièreté du code source de la base de données.

```
-- phpMyAdmin SQL Dump
-- version 5.2.2
-- https://www.phpmyadmin.net/
--
-- Host: localhost
-- Generation Time: May 06, 2025 at 01:11 PM
-- Server version: 10.11.9-MariaDB
-- PHP Version: 7.4.33
```

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `biathlon`
--

--
-- -----
--
-- Table structure for table `activity`
--

CREATE TABLE `activity` (
  `id_activity` int(11) NOT NULL,
  `state` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_520_ci;

--
-- Dumping data for table `activity`
--

INSERT INTO `activity` (`id_activity`, `state`) VALUES
(1, 0);

--
-- -----
--
-- Table structure for table `config`
--

CREATE TABLE `config` (
```

```
`id_config` int(11) NOT NULL,  
`code` int(11) NOT NULL,  
`token` varchar(32) NOT NULL,  
`nom_course` varchar(50) NOT NULL,  
`nb_juges` int(11) NOT NULL DEFAULT 0,  
`max_juges` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_520_ci;  
  
--  
-- Dumping data for table `config`  
--  
  
INSERT INTO `config` (`id_config`, `code`, `token`, `nom_course`,  
`nb_juges`, `max_juges`) VALUES  
(1, 111111, '037afc1154ebe16d409459b17fb2c6f6', 'Course', 0, 0);  
  
-----  
  
--  
-- Table structure for table `race`  
--  
  
CREATE TABLE `race` (  
  `id_race` int(11) NOT NULL,  
  `num` int(11) NOT NULL,  
  `runnerName` varchar(50) NOT NULL,  
  `judgeName` varchar(50) NOT NULL,  
  `token` varchar(32) NOT NULL,  
  `t0` double DEFAULT NULL,  
  `t1` double DEFAULT NULL,  
  `t2` double DEFAULT NULL,  
  `t3` double DEFAULT NULL,  
  `t4` double DEFAULT NULL,  
  `t5` double DEFAULT NULL,  
  `t6` double DEFAULT NULL,  
  `t7` double DEFAULT NULL,  
  `t8` double DEFAULT NULL,  
  `t9` double DEFAULT NULL,  
  `totalTime` double DEFAULT 0,  
  `seqTirs1` varchar(5) DEFAULT '0/0',
```

```
`seqTirs2` varchar(5) DEFAULT '0/0',
`seqTirs3` varchar(5) DEFAULT '0/0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_520_ci;

--
-- Indexes for dumped tables
--

--
-- Indexes for table `activity`
--
ALTER TABLE `activity`
  ADD PRIMARY KEY (`id_activity`);

--
-- Indexes for table `config`
--
ALTER TABLE `config`
  ADD PRIMARY KEY (`id_config`);

--
-- Indexes for table `race`
--
ALTER TABLE `race`
  ADD PRIMARY KEY (`id_race`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `activity`
--
ALTER TABLE `activity`
  MODIFY `id_activity` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `config`
--
ALTER TABLE `config`
```

```

MODIFY `id_config` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT for table `race`
--
ALTER TABLE `race`
  MODIFY `id_race` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

5.3 - Interface Graphique

Après authentification par phpMyAdmin :

The screenshot shows the phpMyAdmin interface for a database named 'biathlon'. The left sidebar shows the database structure with tables 'activity', 'config', and 'race'. The main panel displays a table list with columns: Table, Action, Lignes, Type, Interclassement, Taille, and Perte. The tables listed are 'activity', 'config', and 'race', all using InnoDB engine and utf8mb4_unicode_520_ci collation.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
activity	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_unicode_520_ci	16,0 kio	-
config	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_unicode_520_ci	16,0 kio	-
race	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_unicode_520_ci	16,0 kio	-
3 tables	Somme	3	InnoDB	utf8mb4_general_ci	48,0 kio	0 o

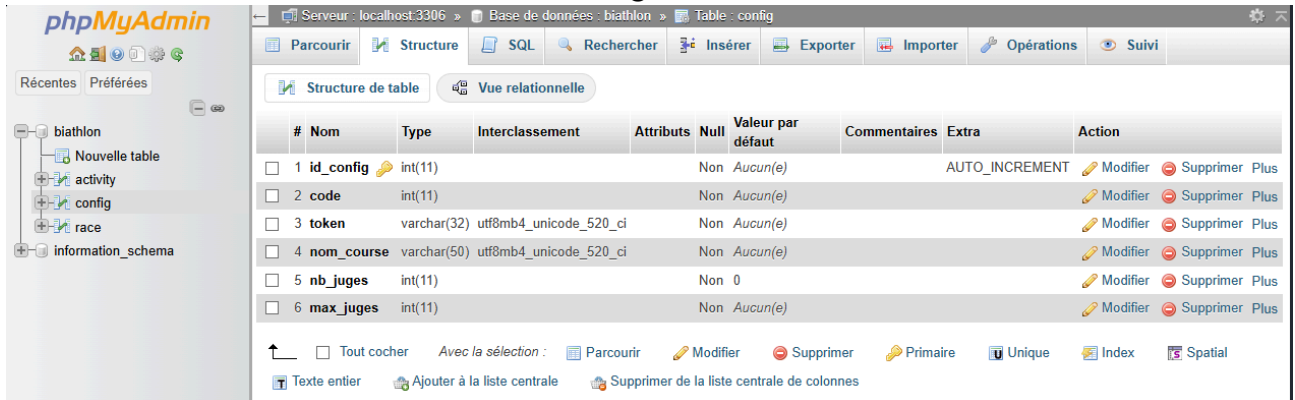
On arrive sur cette interface, qui est la base de données du projet biathlon, qui admet 3 tables nommées 'activity', 'config' et 'race', dont les structures sont détaillées plus bas.

Voici le détail de la structure de la table activity :

The screenshot shows the phpMyAdmin interface for the 'activity' table. The left sidebar shows the database structure with tables 'activity', 'config', and 'race'. The main panel displays the table structure with columns: #, Nom, Type, Interclassement, Attributs, Null, Valeur par défaut, Commentaires, Extra, and Action. The table has two columns: 'id_activity' (int(11), primary key, AUTO_INCREMENT) and 'state' (int(11)).

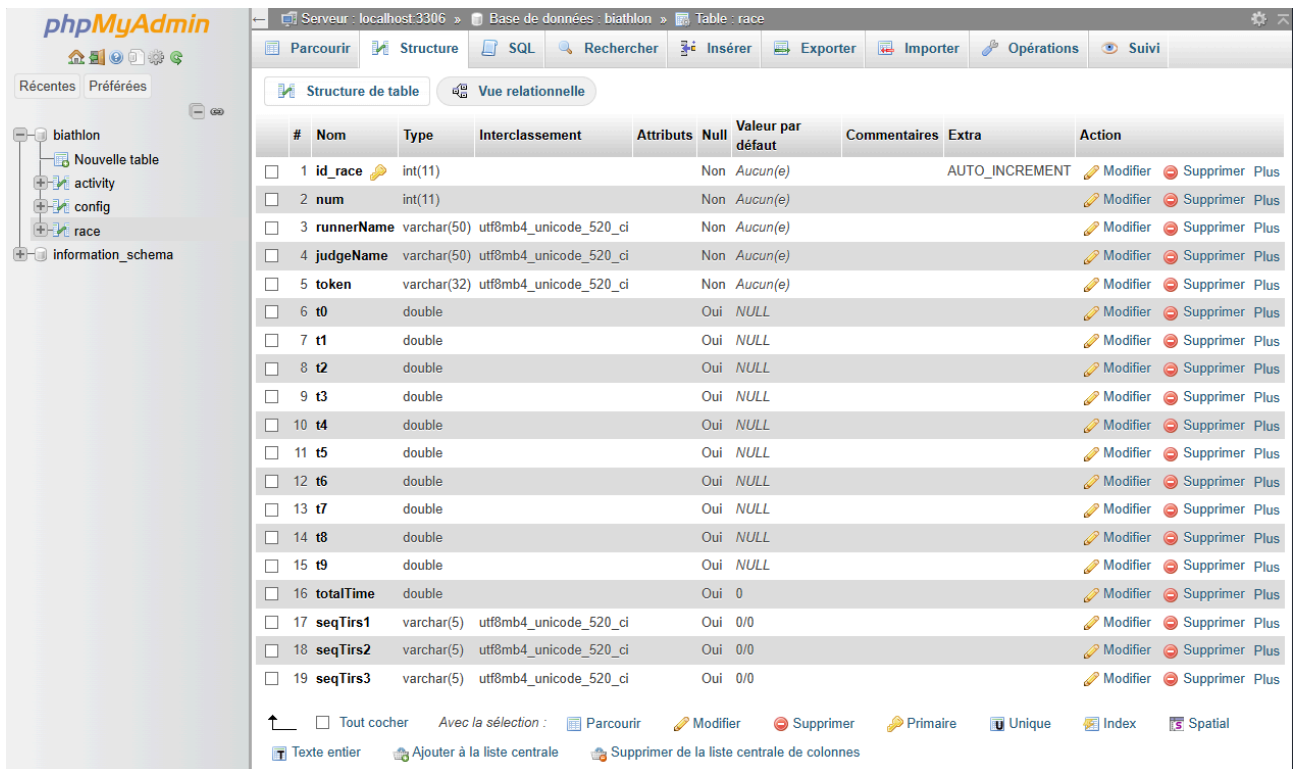
#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id_activity	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
2	state	int(11)			Non	Aucun(e)			Modifier Supprimer Plus

Voici le détail de la structure de la table config :



#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐ 1	id_config	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
☐ 2	code	int(11)			Non	Aucun(e)			Modifier Supprimer Plus
☐ 3	token	varchar(32)	utf8mb4_unicode_520_ci		Non	Aucun(e)			Modifier Supprimer Plus
☐ 4	nom_course	varchar(50)	utf8mb4_unicode_520_ci		Non	Aucun(e)			Modifier Supprimer Plus
☐ 5	nb_juges	int(11)			Non	0			Modifier Supprimer Plus
☐ 6	max_juges	int(11)			Non	Aucun(e)			Modifier Supprimer Plus

Voici le détail de la structure de la table race :



#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
☐ 1	id_race	int(11)			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
☐ 2	num	int(11)			Non	Aucun(e)			Modifier Supprimer Plus
☐ 3	runnerName	varchar(50)	utf8mb4_unicode_520_ci		Non	Aucun(e)			Modifier Supprimer Plus
☐ 4	judgeName	varchar(50)	utf8mb4_unicode_520_ci		Non	Aucun(e)			Modifier Supprimer Plus
☐ 5	token	varchar(32)	utf8mb4_unicode_520_ci		Non	Aucun(e)			Modifier Supprimer Plus
☐ 6	t0	double			Oui	NULL			Modifier Supprimer Plus
☐ 7	t1	double			Oui	NULL			Modifier Supprimer Plus
☐ 8	t2	double			Oui	NULL			Modifier Supprimer Plus
☐ 9	t3	double			Oui	NULL			Modifier Supprimer Plus
☐ 10	t4	double			Oui	NULL			Modifier Supprimer Plus
☐ 11	t5	double			Oui	NULL			Modifier Supprimer Plus
☐ 12	t6	double			Oui	NULL			Modifier Supprimer Plus
☐ 13	t7	double			Oui	NULL			Modifier Supprimer Plus
☐ 14	t8	double			Oui	NULL			Modifier Supprimer Plus
☐ 15	t9	double			Oui	NULL			Modifier Supprimer Plus
☐ 16	totalTime	double			Oui	0			Modifier Supprimer Plus
☐ 17	seqTirs1	varchar(5)	utf8mb4_unicode_520_ci		Oui	0/0			Modifier Supprimer Plus
☐ 18	seqTirs2	varchar(5)	utf8mb4_unicode_520_ci		Oui	0/0			Modifier Supprimer Plus
☐ 19	seqTirs3	varchar(5)	utf8mb4_unicode_520_ci		Oui	0/0			Modifier Supprimer Plus

6 - Sécurisation

Afin de s'assurer de la sécurité des échanges et du bon fonctionnement de l'application, ainsi que du déroulé des courses, il fallait évidemment sécuriser l'application, avec un système de cryptage des échanges (https) et installer un firewall.

6.1 - Passer du http au https

Par défaut, le système Linux Apache MySQL PHP (LAMP) prévoit que le port serveur Apache en http soit 80, et 443 en https. Seulement, il y a quelques modifications à effectuer en plus de cela pour s'assurer du bon fonctionnement, notamment rediriger toutes les connexions http en https, donc du port 80 au port 443.

D'abord, ouvrir avec nano le fichier de configuration accessible via la commande suivante :
-> /etc/apache2/sites-available/000-default.conf

Puis redémarrer le serveur Apache :
-> sudo systemctl restart apache2

6.2 - Installation pare-feu

La prochaine étape de sécurisation consiste alors en l'installation d'un firewall, ce dernier avait essentiellement besoin d'être léger, nous n'avons pas besoin d'une grande complexité d'exécution, nous avons opté pour le pare-feu Uncomplicated FireWall (UFW).

Ci-dessous les seules commandes que j'ai eu besoin d'utiliser
sudo apt install ufw
sudo ufw enable
sudo ufw allow ssh
sudo ufw status verbose

Issa Bâ

Partie personnelle - Issa Bâ

Sommaire

1. Présentation rôle et objectifs.....	3
2. Disposition des tâches.....	4
2.1. Ordre.....	4
2.2 Cahier des charges.....	4
2.3 Diagramme de Gantt.....	5
3. Mise en application des tâches.....	6
3.1 Mise en place de l'AP Wifi.....	6
3.2 Conception du code source côté serveur.....	7
3.3 Développement du code source intégré dans les micro-contrôleurs (cible/pistolet).....	11
4. Partie physique.....	13
5. ANNEXES.....	14

1. Présentation rôle et objectifs

Mon rôle dans ce projet est de faire un code source capable de traiter les trames reçues par le pistolet et la cible, un autre permettant l'envoi de ces trames depuis les micro-contrôleurs intégrés, pouvoir assurer leurs intégrités de sorte à ce qu'une trame avec des données manquantes ne soit pas traitée, faire en sorte que les données circulent correctement du pistolet à la cible (qu'un tir active une led spécifique avec une couleur qui l'est également se basant sur la réussite du tir).

Je dois faire également la connectivité des appareils grâce à un AP Wifi (Access Point Wifi), point d'accès wifi, qui sera installé sur la raspberry et où les appareils wifi tenteront constamment de se connecter, en cas de réussite le "mode examen" sera activé, mode où les résultats seront envoyés sur la bdd, où les tirs réussis/ratés seront signalés à l'aide des leds lumineuses placés sur la cible, le mode examen est désactivé lorsque la connexion est perdue.

Dans le cas échéant, le mode "entraînement" sera activé, mode qui ne nécessite aucune connexion entre les appareils, la cible compte uniquement les tirs réussis, le mode est désactivé si la connexion devient à nouveau apte à l'ensemble des appareils.

Une fiche recette approuvant ou non l'efficacité d'une partie de mes tâches est également exigée.

Mon objectif sera de proposer une version fonctionnelle et au plus proche de son paroxysme. Conclure sur la version proposée, développer une analyse sur les complications rencontrées et un bilan introspectif.

2. Disposition des tâches

2.1. *Ordre*

L'arrangement que j'ai choisi est celle où je commence par l'installation du micro-ordinateur (Raspberry pi), puis j'y mettrai en place l'AP Wifi, ensuite par la conception du code source contenu dans la Raspberry, dans celui de la cible et du pistolet.

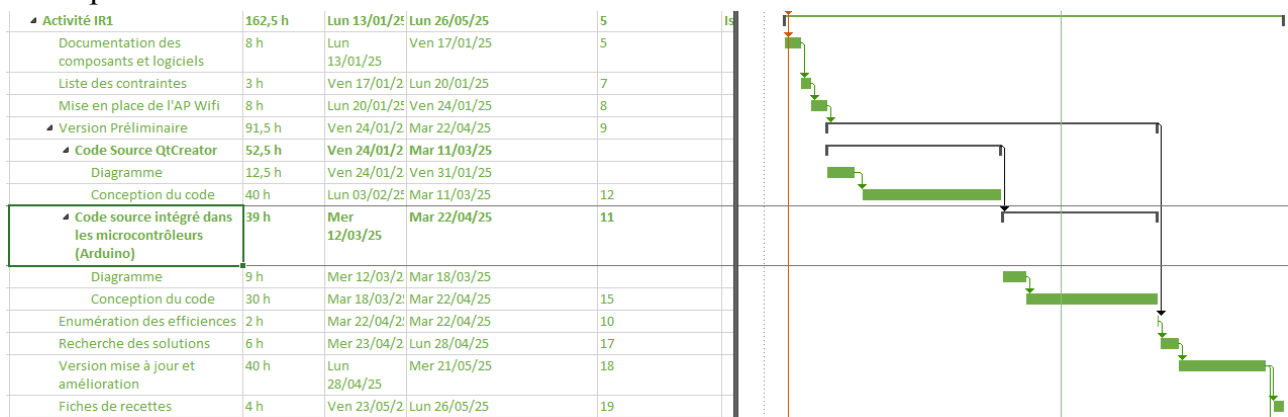
2.2 *Cahier des charges*

Mes tâches à réaliser sont :

- Installation de l'AP WIFI
- Réalisation le code source du superviseur (hormis la classe Bdd)
- Code source intégré dans les microcontrôleurs (Pistolet/Cible)
- Fiche de recettes

2.3 Diagramme de Gantt

Ci-dessous, ma partie du diagramme de Gantt avec temps et tâches réels (entier dans Annexe, document-annexe 1). On peut d'ores et déjà souligner que la planification Microsoft Project n'est pas assez pointilleuse et aborde les tâches de manières trop globales, en effet l'AP Wifi, l'installation des systèmes d'exploitation, des logiciels et les codes sources, n'y sont pas énumérés et faisant office de plan principal le manque de ces étapes nécessaires nuit fortement à l'avancement du projet. Donc pour continuer plus sereinement il m'a fallu revisiter le diagramme, en en faisant un plus dense mais qui m'a nettement plus éclairé sur les étapes à suivre.



3. Mise en application des tâches

3.1 Mise en place de l'AP Wifi

Pour la mise en place du point d'accès wifi, j'ai décidé d'utiliser RaspAP, un logiciel open source capable de plusieurs autres fonctionnalités comme le routage, Nat ou par exemple l'installation d'un vpn.

L'AP Wifi servira à la connectivité entre les appareils et doit être capable d'attribuer une adresse IP à chacune d'entre elles.

Pour cela il faudra donc un adressage DHCP (Dynamic Host Configuration Protocol), la possibilité choisi pour parvenir à ce résultat était "Dnsmasq", un serveur faisant office d'adressage DHCP et de DNS (Domain Name Service) essentiel pour traduire les noms de domaine en adresse IP, qui est compris et installé en même temps que le logiciel RaspAP.

Ce logiciel contient également hostapd, un logiciel, daemon car c'est un programme qui tourne en arrière plan, permettant de créer un point d'accès via un serveur installé simultanément, qui plus est avec différents protocoles de sécurités notamment celui qu'on utilisera le WPA2 (Wi-Fi Protected Access).

Le protocole WPA2 est bien plus sécurisé que son prédécesseur le WPA, qui utilisait TKIP (Temporal Key Integrity Protocol) cryptage désormais obsolète, là où WPA2 utilise un cryptage AES(Advanced Encryption Standard) qui reste faillible face à une forte puissance de calcul mais bonifie tout de même la sécurité du point d'accès. Le mieux aurait été d'utiliser WPA3 qui est nettement meilleur mais non desservi par hostapd.

Donc après l'installation de dnsmasq et de hostapd, dans le fichier hostapd.conf, je configure le cryptage WPA souhaité, le SSID (Service Set Identifier), indiquant le nom du point d'accès wifi et le mot de passe.

Après avoir installé le point d'accès, il ne fonctionnait toujours pas cela était dû à la configuration de dnsmasq.d, fichier de configuration de Dnsmasq, qui ne renvoyé vers aucun dossier, ce problème n'empêchait pas la connexion à l'AP Wifi mais aurait pu poser problème au niveau de l'adressage IP.

Après rectification le point d'accès était opérationnel, permettait de s'y connecter et également l'adressage d'IP et l'aspect sécurité était rempli, connexion nécessitant un mot de passe et avec un protocole WPA2 avec un cryptage AES.

3.2 Conception du code source côté serveur

La suite est la conception d'un code côté serveur, code conçu sur Qt Creator en C++ et installé sur la raspberry, ce code doit être capable de :

- Servir de serveur TCP
- Pouvoir gérer les clients du serveur (les lister, recevoir/envoyer des trames)
- Décrypter les trames pour pouvoir déduire quel appareil a envoyé la trame et son numéro
- Mettre les données dans les tables de bdd

Voici ci-joint les diagrammes SysML/UML les différents comportements des classes selon les situations.

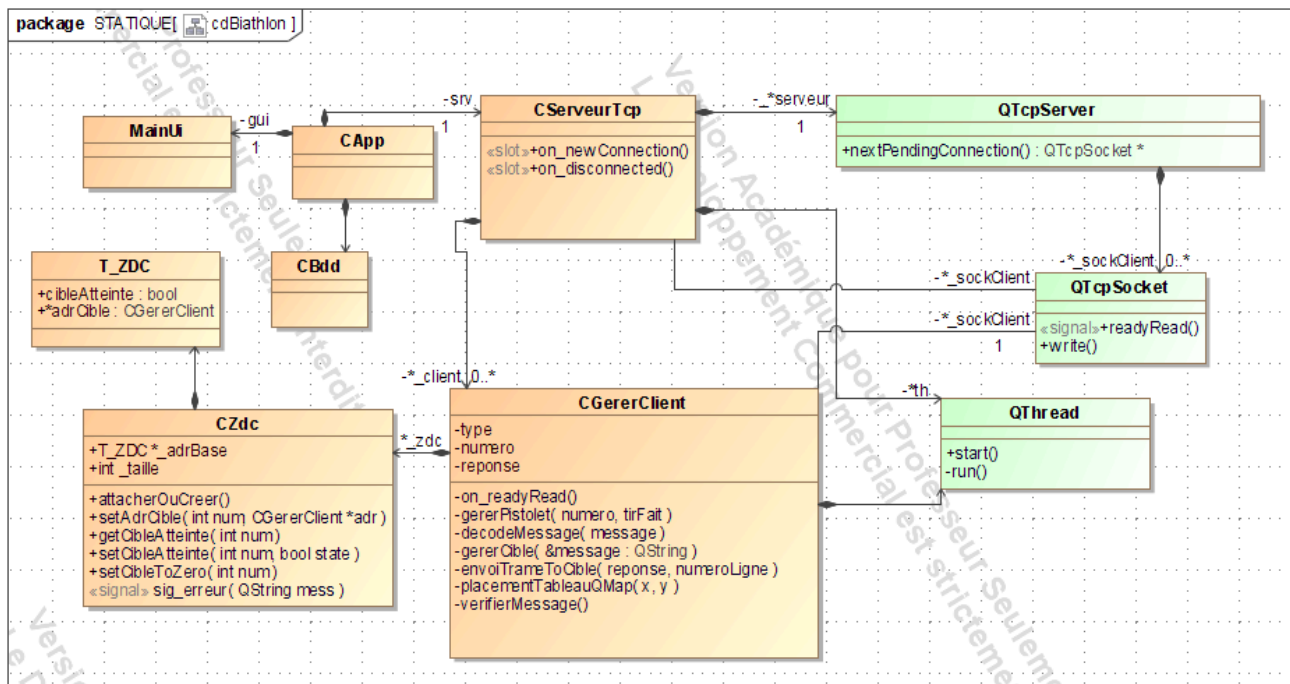


Diagramme de classe : Schématisation du code source

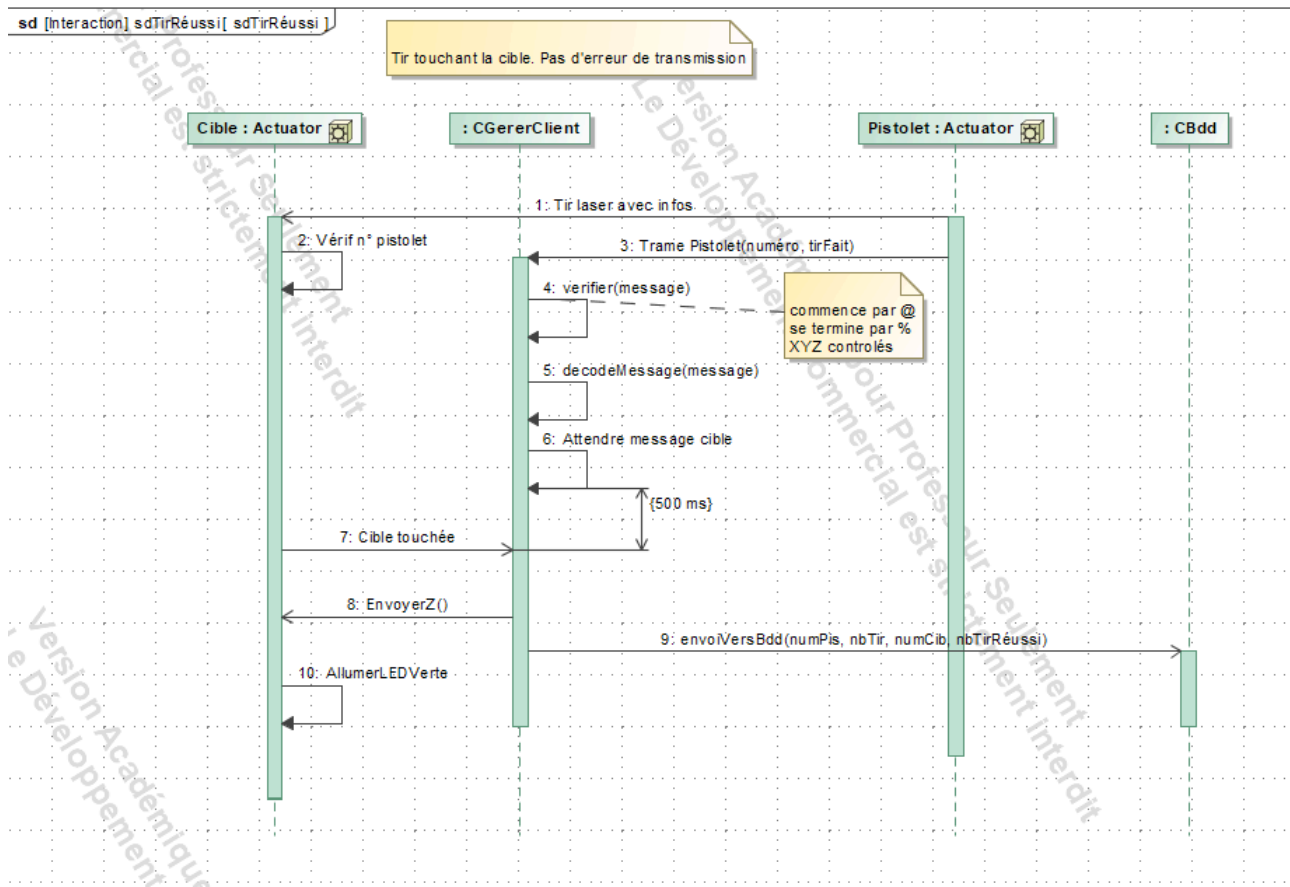


Diagramme de séquence côté serveur (nominal) : Tir réussi par le coureur

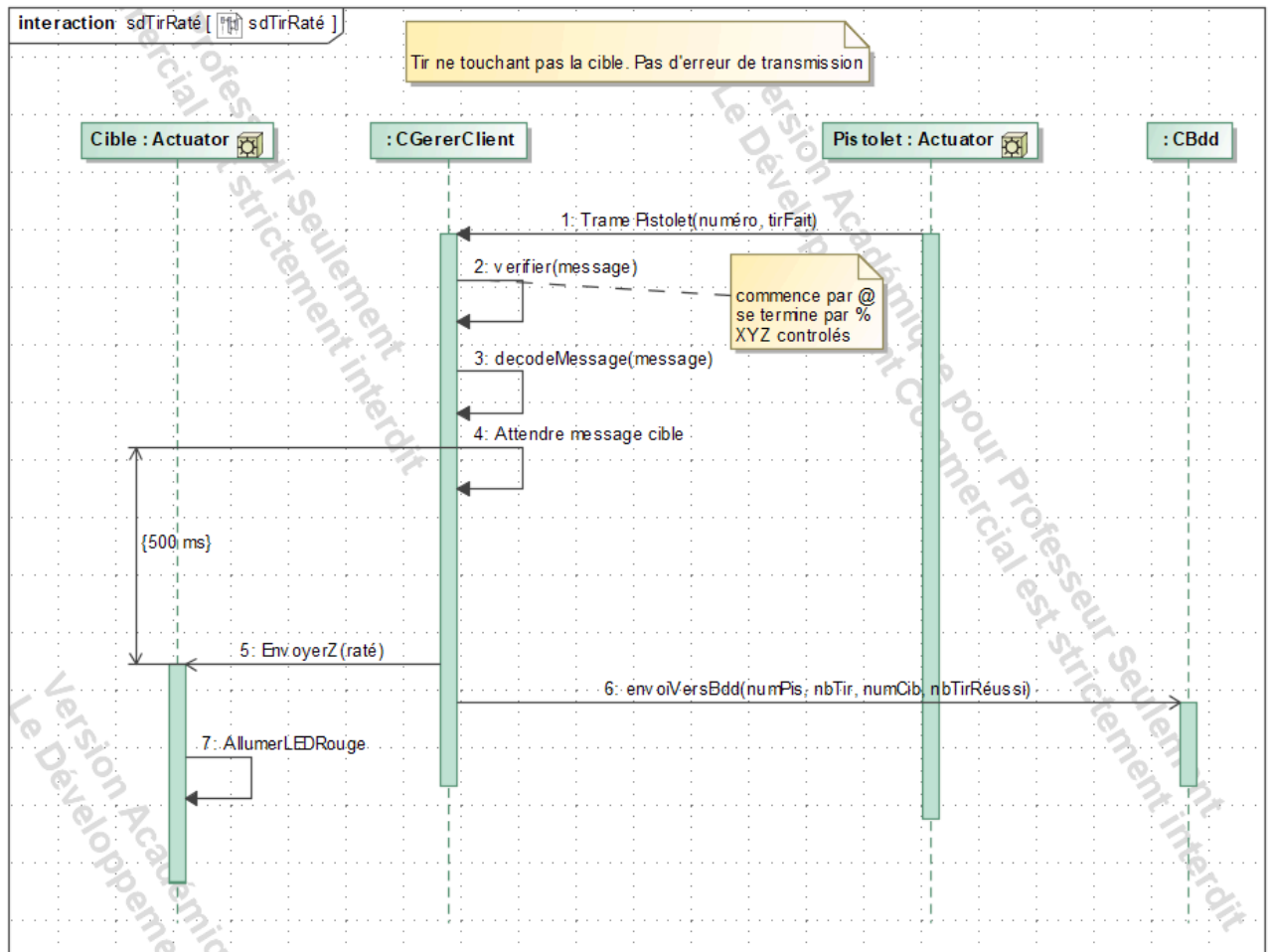


Diagramme de séquence côté serveur (alternatif) : Tir raté par le coureur

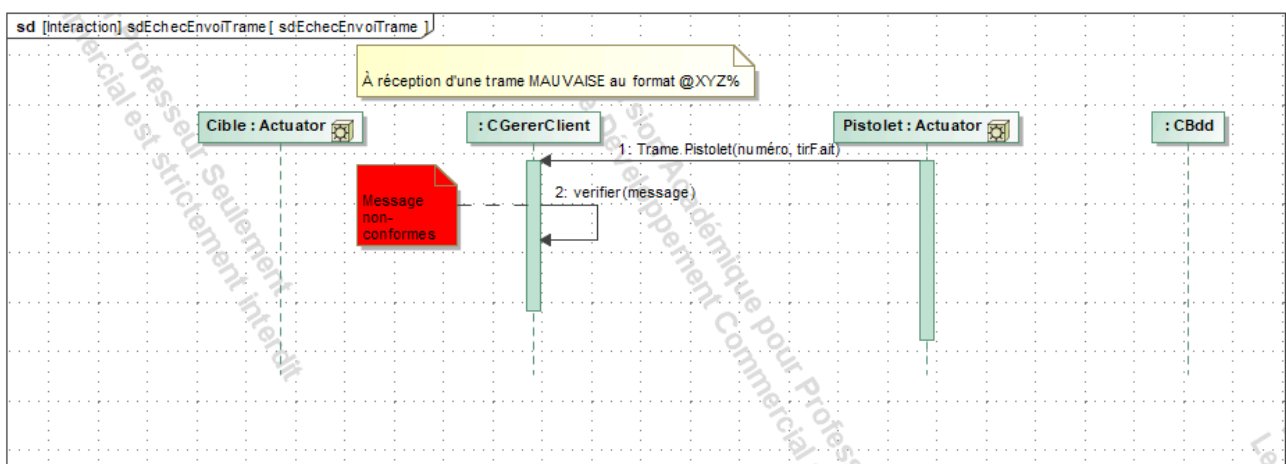


Diagramme de séquence(échec) : Réception d'une mauvaise trame

Pour pouvoir déduire qu'elle appareil a envoyé la trame, son numéro et l'information qu'elle contient, j'ai imaginé un format typique, tel que :

@XYZ%

Les caractères "@" et "%" servent à vérifier l'intégralité de la trame, bien que faillible cela apportera une sécurité supplémentaire contre les pertes de données.

Le caractère "X" désigne l'appareil concerné où "0" équivaut à une cible et "1" à un pistolet, "Y" correspond au numéro et "Z" équivaut à, dans le cas où c'est un pistolet, "0" si le tir est réussi, dans le cas où c'est une cible "1", dans le cas général si c'est égal à "0" cela signifie que c'est une trame d'identification et que la classe CGererclient se chargera de rajouter l'appareil dans la structure qui servira à renvoyer une trame si nécessaire.

L'ensemble du code fonctionne ainsi, la classe CServeurTCP contient et lance le serveur TCP si une connexion est établie l'adresse de la socket est conservé et envoyé à la classe CGererClient, après l'envoi d'une trame depuis l'un des clients au serveur TCP, si c'est une trame d'identification cette adresse est listée dans _adrBase de la structure T_ZDC appartenant à la classe CZdc (Classe Zone de données commune).

Lors de la réception d'une trame disant qu'un pistolet a tiré, le code source attend pendant 10 secondes, pendant ce temps si une cible a été touchée elle enverra une trame où "Z" égal à "1", la classe CGererClient se chargera de mettre en "true" "cibleAtteinte" situé dans la structure T_ZDC.

Une fois les 10 secondes passées, le code source vérifie l'état de "cibleAtteinte" s'il est "true" pour la cible ayant le même numéro cela signifie que le tir est réussi sinon c'est un tir raté, chaque tir est prévu d'être envoyé sur la base de données avec le numéro du tir.

Si le tir est dit réussi la méthode "envoiTrameToCible" envoi une trame qui détient le résultat du tir et le numéro du tireur, cette trame sera décrypté par la cible et mettra le numéro de la led correspondant au numéro du tir dans l'état renvoyé par la méthode, soit si le tir est réussi mettra la led en vert sinon en rouge, quel que soit le résultat du tir la variable _nbTirs comptera les tirs réalisés.

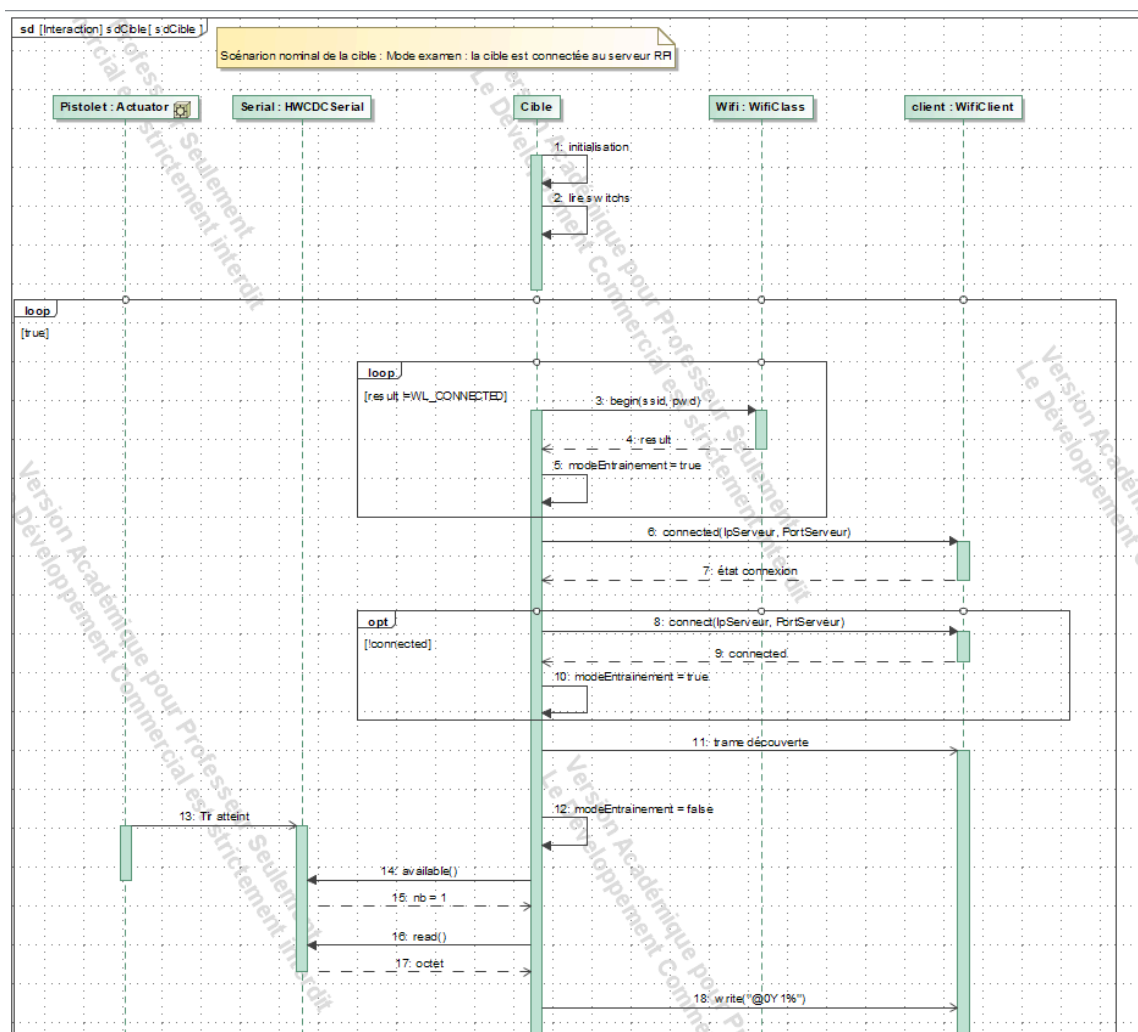
Les tirs ratés et les tirs réalisés sont mis sur la base de données seulement si la course a été lancée depuis le site web, ces données seront utilisées pour informer les juges et le public.

À mon premier essai, après avoir simulé la connexion et l'envoi de trame par une console et la commande telnet, je pensais que le code source était complet sauf que lorsque le code source devait gérer plusieurs cibles et pistolets à la fois, donc il a fallu penser à mettre un sorte de liste qui listera l'adresse des sockets pour pouvoir communiquer avec les appareils sans risque de l'envoyer à un autre.

3.3 Développement du code source intégré dans les micro-contrôleurs (cible/pistolet)

Le but de ce code est de se connecter à l'AP Wifi, au serveur TCP, de retenter les connexions dans le cas échéants, si celle-ci est faite une frame d'identification est envoyée au code source, dans le cas où c'est un pistolet chaque tir enverra la frame typique avec le numéro de tir adéquat et pour une cible une frame de confirmation en cas d'impact, une écoute constante est faite pour pouvoir lire une frame d'état disant de changer une led (tel que @NY%, "N" équivaut au numéro de tir et "Y" désigne l'état à mettre).

Voici ci-joint les diagrammes SysML/UML expliquant les différents comportements des microcontrôleurs selon les situations.



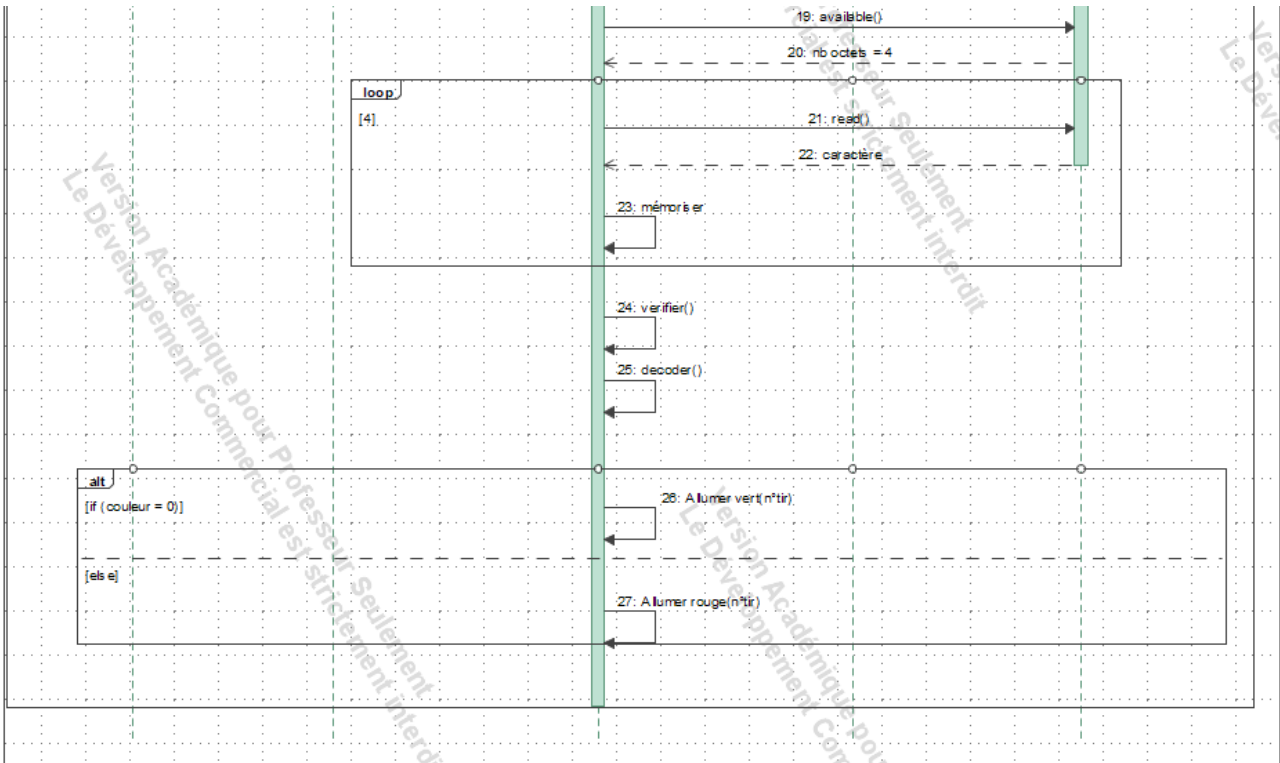


Diagramme de séquence (nominal) : Cible

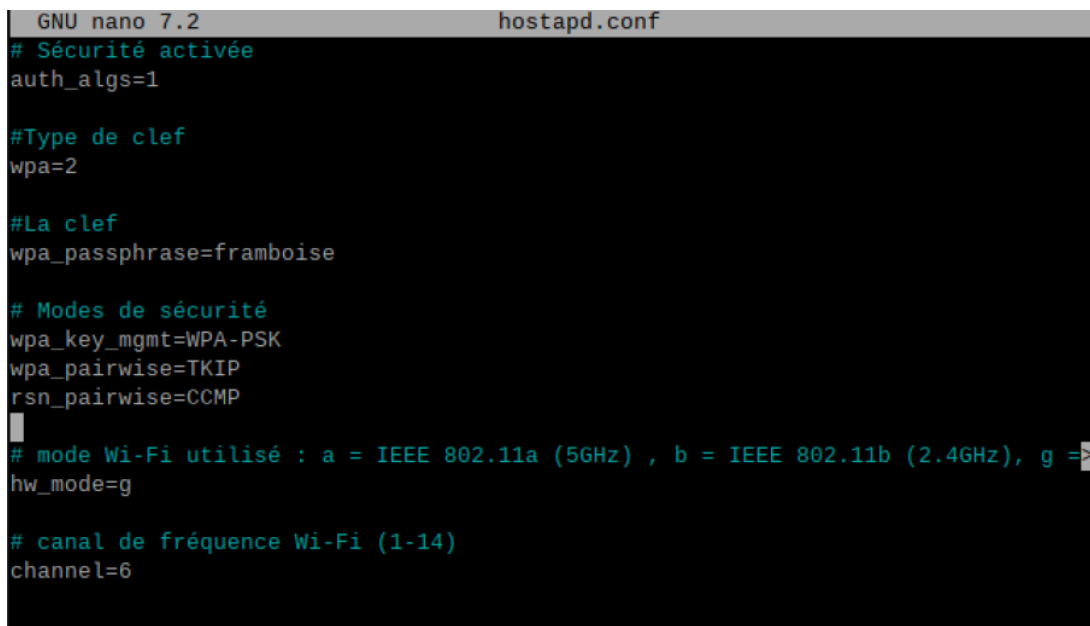
4.Aspect physique

Pour ma partie physique, j'ai décidé de présenter la Wi-fi, qui est l'abréviation de Wireless Fidelity qui en français signifie fiabilité du sans fil combinant un ensemble de protocoles de communication sans fil.

Son but est de transmettre des données numériques sans fil en utilisant les ondes radios, et est régi par les normes de l'association professionnelle IEEE (Institute of Electrical and Electronics Engineers), plus précisément par la norme IEEE 802.11 concernant les connexions sans fil qui définit les caractéristique d'un réseau.

Les caractéristiques d'un wifi sont définies par sa portée (de 20-70 mètres), son débit maximal définissant le nombre de bits transférable par seconde, la fréquence d'émission qui détermine à quelle vitesse les ondes Wi-fi sont envoyées et le type de modulation, donc la manière dont les données numériques sont transformées pour être envoyées.

Selon la configuration du point d'accès *Wi-fi*, figurant ci-dessous, la norme utilisée serait la 802.11g.



```
GNU nano 7.2 hostapd.conf
# Sécurité activée
auth_algs=1

#Type de clef
wpa=2

#La clef
wpa_passphrase=framboise

# Modes de sécurité
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

# mode Wi-Fi utilisé : a = IEEE 802.11a (5GHz) , b = IEEE 802.11b (2.4GHz), g =>
hw_mode=g

# canal de fréquence Wi-Fi (1-14)
channel=6
```

Capture d'écran d'un des fichiers de configurations (hostapd.conf)

Et voici l'ensemble des caractéristiques correspondant à chaque version des différents Wi-fi :

Version Wi-Fi	Norme	Portée (en intérieur)	Débit théorique maximal	Fréquences utilisées
Wi-Fi 1	802.11b	35 mètres	11 Mbps	2,4 GHz
Wi-Fi 2	802.11a	20 mètres	54 Mbps	5 GHz
Wi-Fi 3	802.11g	38 mètres	54 Mbps	2,4 GHz
Wi-Fi 4	802.11n	70 mètres	600 Mbps	2,4 GHz et 5 GHz
Wi-Fi 5	802.11ac	35 mètres	3,5 Gbps	5 GHz
Wi-Fi 6	802.11ax	50 mètres	9,6 Gbps	2,4 GHz et 5 GHz
Wi-Fi 6E	802.11ax	50 mètres	9,6 Gbps	2,4 GHz, 5 GHz et 6 GHz
Wi-Fi 7	802.11be	50 mètres (estimé)	46 Gbps (théorique)	2,4 GHz, 5 GHz et 6 GHz

Donc notre disposition correspond à la version Wi-Fi 3 donc une portée de 38 mètres, un débit théorique maximal de 54 Mbps et une fréquence porteuse, fréquence à laquelle le signal est transmis, à 2.4 GHz.

En vue des ressources demandées par notre projet, cette version nous satisfait totalement que ce soit du point de vue de la portée ou de la fréquence utilisée.

5. ANNEXES

Lien Framagit vers le code source Qt Creator :

<https://framagit.org/GeorgeMusashi/qprojetbiathlon>

Lien Framagit vers les code source Arduino :

https://framagit.org/GeorgeMusashi/arduino-cible_pistolet

Sources :

Partie physique :

[fr.wikipedia.org/wiki/IEEE_802.11](fr.wikipedia.org/wiki/IEEE_802.11)

[fr.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers](fr.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers)

www.zoneadsl.com

<fr.wikipedia.org/wiki/Wi-Fi>

Access point Wi-Fi

<https://wiki-fablab.grandbesancon.fr>

Tomy Torrecillas
Projet Biathlon – Tomy Torrecillas

Sommaire

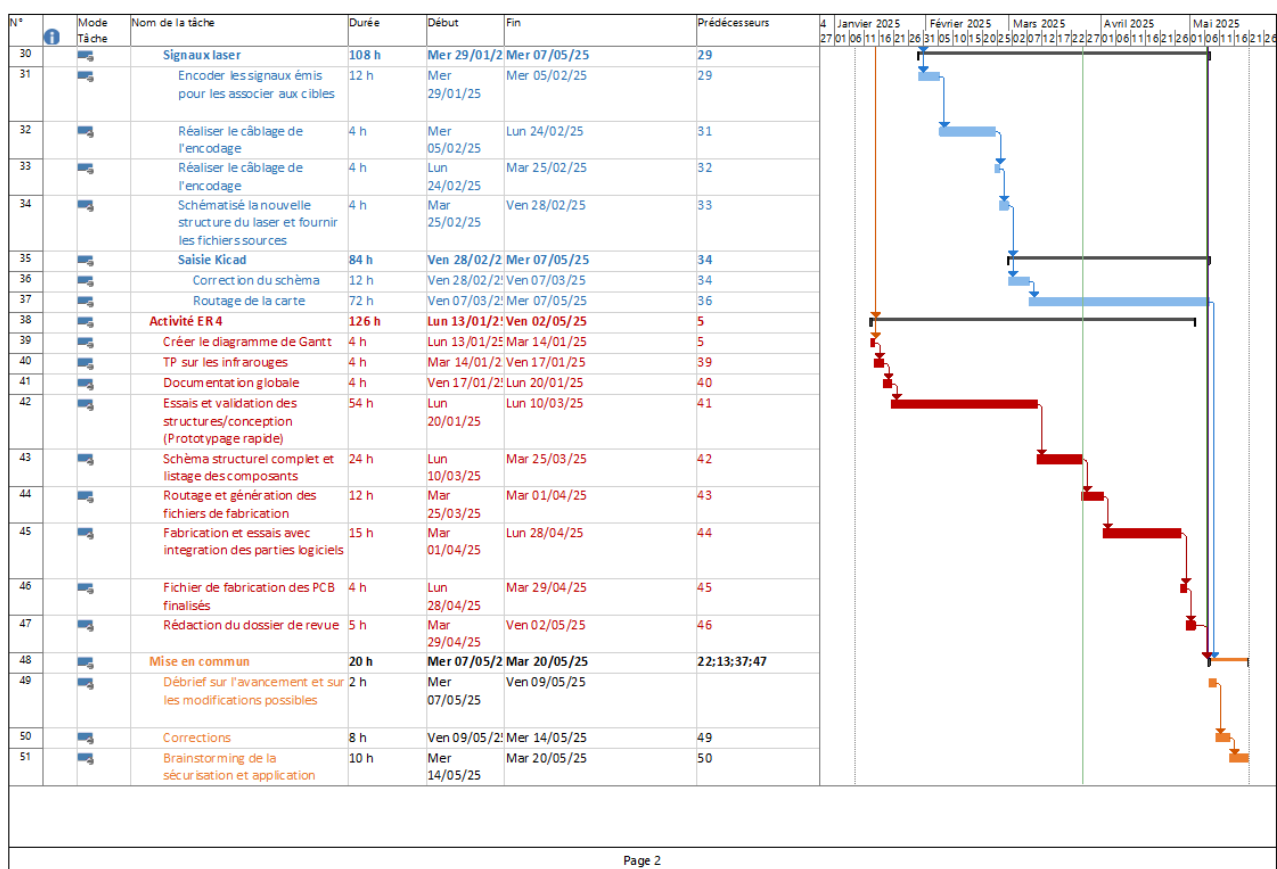
1 - Introduction.....	3
2 - Diagramme de Gantt.....	4
2.1. Diagramme de Gantt temp réel.....	5
3 - Diagramme d'exigences.....	6
4 - Schéma structurel réalisé avec Kicad.....	7
4.1. Indicateurs LED et Transistors (Partie Haute du Schéma).....	8
4.2. Détection des tirs (Capteur SFH4201).....	9
4.3. Microcontrôleur et Traitement (ESP32-S3).....	10
4.4. Communication et Contrôle (ATtiny85).....	11
4.5. Pont Diviseur.....	12
5 - Carte de la Cible.....	13
5.1. Routage final de la Carte.....	16
5.2. Carte avant soudure.....	17
5.3. Carte après application de la pâte à braser.....	18
5.4. Application de la pâte à braser.....	19
5.5. Carte après soudure.....	20
6 - Analyse trame Essais ESP32-S3.....	21
7 - Suivi du Projet.....	22
8 - Nomenclature de la carte.....	24
9 - Fiche de mise en service Cible Biathlon.....	26

1 - Introduction

Dans le cadre de la création d'un biathlon au sein de mon lycée suite à la demande du Professeur de sport Mr. Le Duc , j'ai eu l'opportunité de jouer un rôle central dans l'organisation et la mise en place de ce sport qui mélange endurance et précision . Plus particulièrement, ma mission s'est concentrée sur la réalisation de la cible .

En collaboration avec une équipe d'élèves et de professeurs motivée , j'ai contribué à concevoir un système adapté à l'environnement scolaire mais aussi au budget. Mon objectif principal était de créer une cible interactive et fiable, capable de détecter avec précision les tirs et d'afficher les résultats en temps réel. Nous sommes en train de développer un prototype fonctionnel, testé et amélioré au fil des essais.

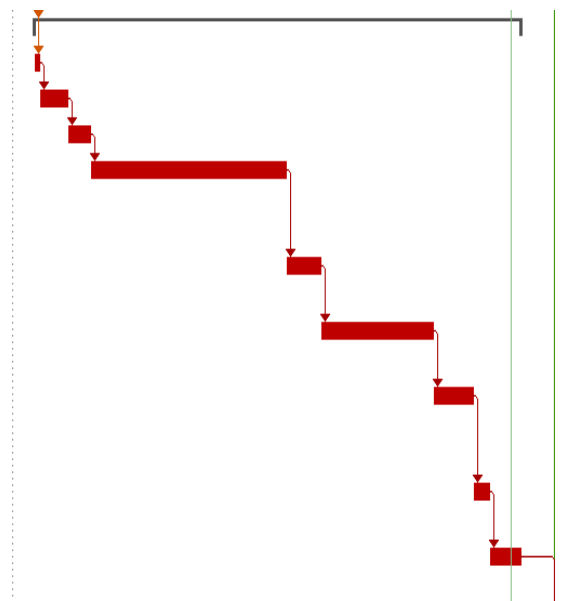
2 - Diagramme de Gantt



Ce diagramme de Gantt prévisionnel représente l'organisation du travail et le temps consacré à chaque étape de mon projet de cible laser en rouge. Il montre que le projet s'est déroulé sur plusieurs semaines, en partant de la planification (création du diagramme, documentation) jusqu'à la fabrication du PCB final et la rédaction du dossier de revue. Chaque tâche est planifiée avec une certaine durée. L'utilisation de ce diagramme m'a aidé à structurer efficacement les différentes phases du projet, comme la conception, la soudure et les tests.

2.1. Diagramme de Gantt temp réel

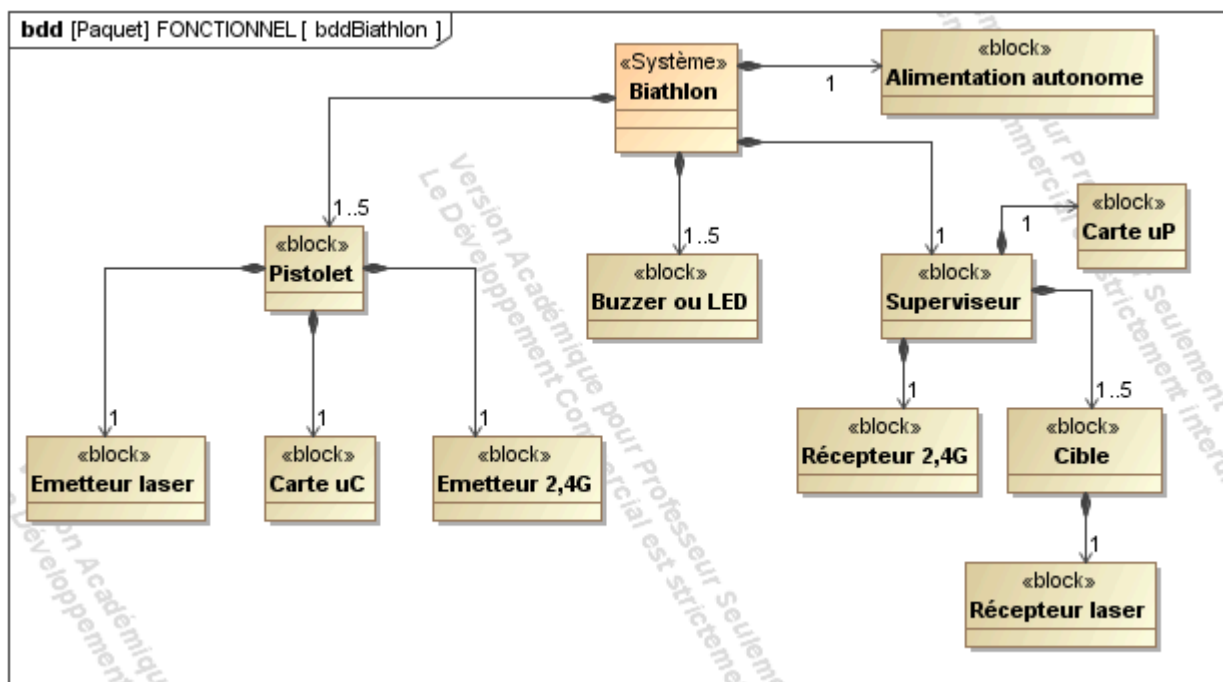
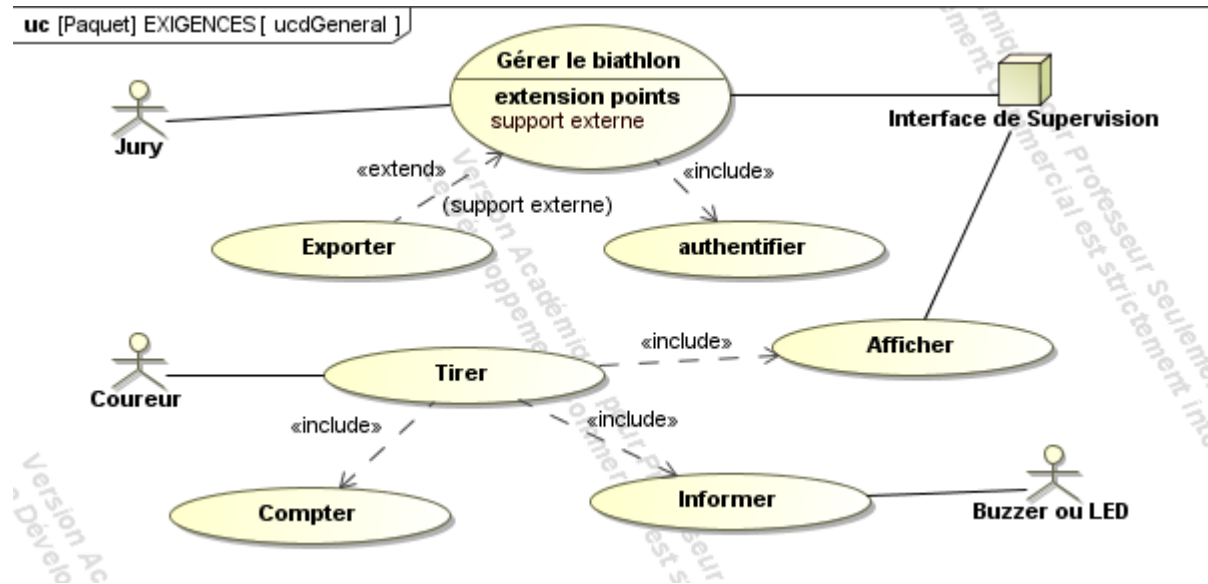
▲ Activité ER 4	146 h	Lun 13/01/25	Mer 14/05/25	5
Créer le diagramme de Gantt	4 h	Lun 13/01/25	Mar 14/01/25	5
TP sur les infrarouges	12 h	Mar 14/01/25	Mar 21/01/25	45
Documentation globale	6 h	Mar 21/01/25	Lun 27/01/25	46
Essais et validation des structures/conception (Prototypage rapide)	54 h	Lun 27/01/25	Lun 17/03/25	47
Schéma structurel complet et listage des composants	16 h	Lun 17/03/25	Mar 25/03/25	48
Routage et génération des fichiers de fabrication	18 h	Mar 25/03/25	Mar 22/04/25	49
Fabrication et essais avec intégration des parties logiciels	16 h	Mar 22/04/25	Ven 02/05/25	50
Fichier de fabrication des PCB finalisés	8 h	Ven 02/05/25	Mar 06/05/25	51
Rédaction du dossier de revue	12 h	Mar 06/05/25	Mer 14/05/25	52



On constate que la grande majorité des tâches prévues m'ont demandé beaucoup plus de temps que prévu, notamment la préparation des fichiers de fabrication, la rédaction du dossier de revue, ainsi que le routage, qui s'est avéré particulièrement long.

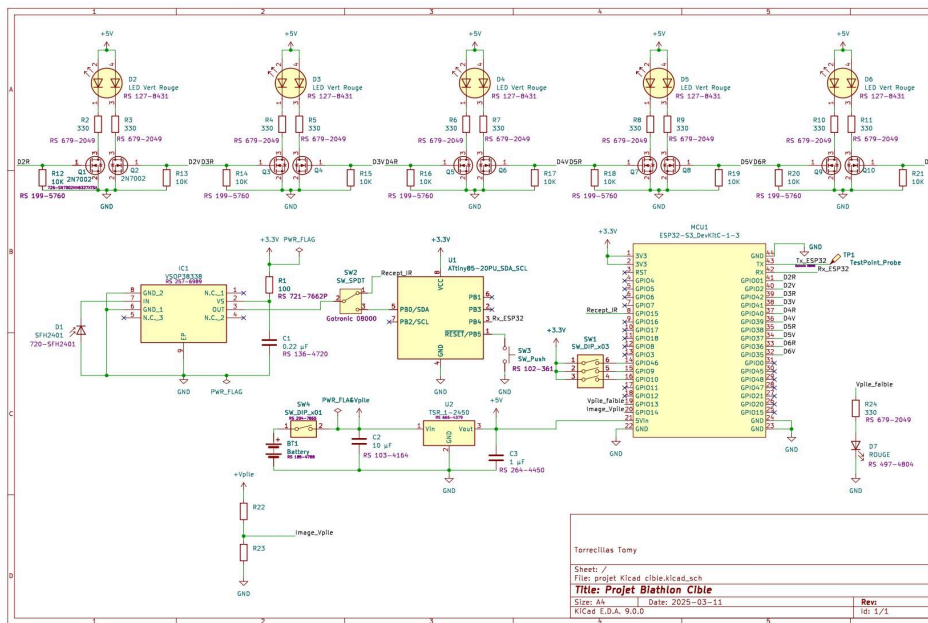
De plus, le TP sur les infrarouges nous a également pris plus de temps que prévu, en raison de plusieurs essais réalisés en dehors des heures de TP.

3 - Diagramme d'exigences



4 - Schéma structurel réalisé avec Kicad

Schéma de la cible.

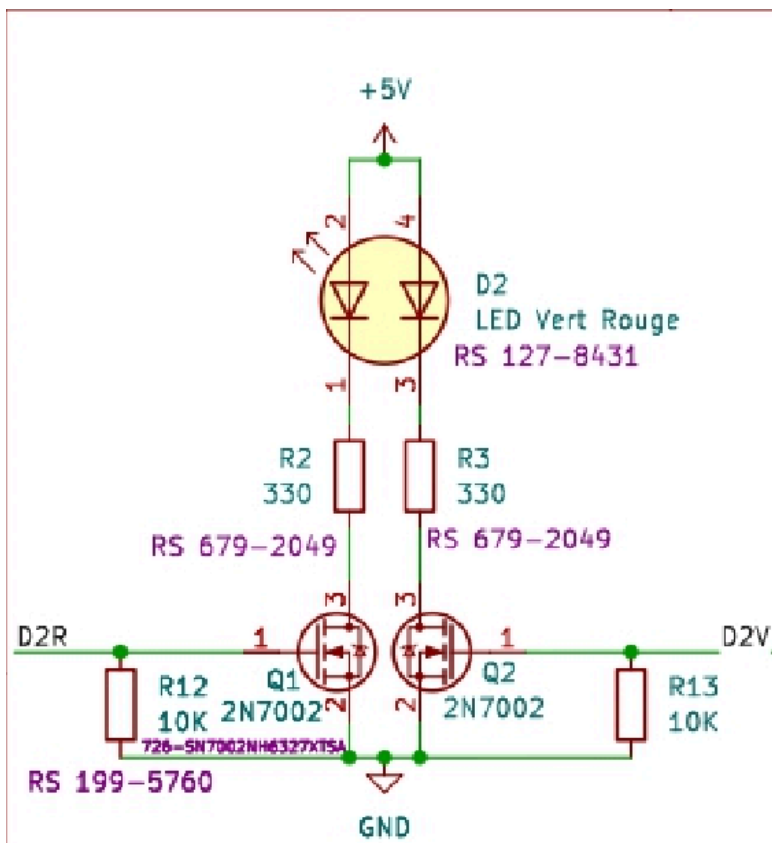


La Cible utilise pour le retro-engineering

<https://www.ecoaims.com/products/laser-pistols-and-optical-pistols/laser-pistols-and-targets/pp320exp-4-pack-club-cases-pro-smartphone-compatible-copy/>

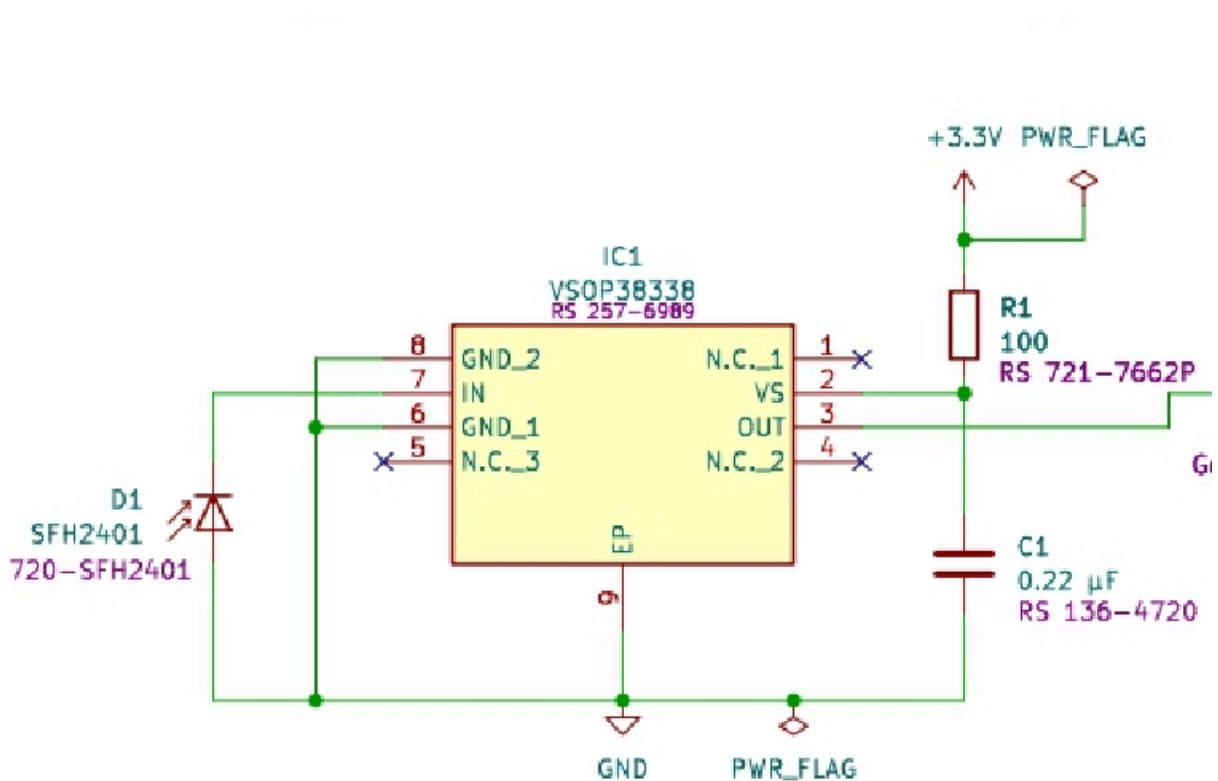
4.1. Indicateurs LED et Transistors (Partie Haute du Schéma)

- **LEDs rouges et vertes (D2 à D6) :** Ces LEDs indiquent visuellement si un tir est réussi ou non.
- **Transistors MOS Canal N (Q2 à Q6 - 2N7002) :** Ils agissent comme des interrupteurs électroniques permettant d'allumer ou d'éteindre les LEDs selon le signal de contrôle venant de l'ESP32.
- **Résistances associées (R12 à R21) :** Elles limitent le courant pour protéger les LEDs et assurer un fonctionnement correct des transistors.



4.2. Détection des tirs (Capteur SFH4201)

- **Photodiode infrarouge (D1 - SFH4201)** : Cette photodiode détecte la présence d'un faisceau laser (tir du biathlète).
- **Circuit amplificateur IC1 (VOS617A-3X008T)** : un démodulateur pour photodiode, il permet de supprimer la porteuse à 38KHz du signal infrarouge reçu.
- **Résistances et condensateurs associés (R1, C1)** : Ils stabilisent le signal et ajustent la sensibilité du capteur.

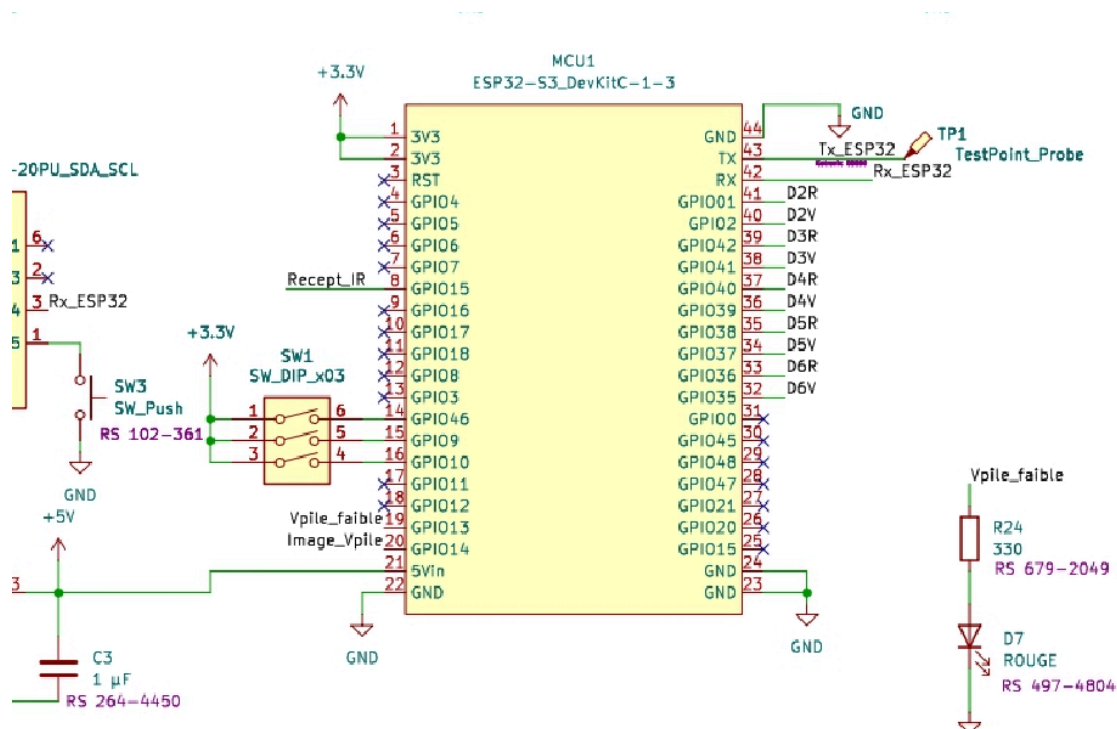


4.3. Microcontrôleur et Traitement (ESP32-S3)

- **ESP32-S3 (MCU1) :** C'est le cerveau du système, il reçoit les signaux du capteur, traite les informations et commande les LEDs.

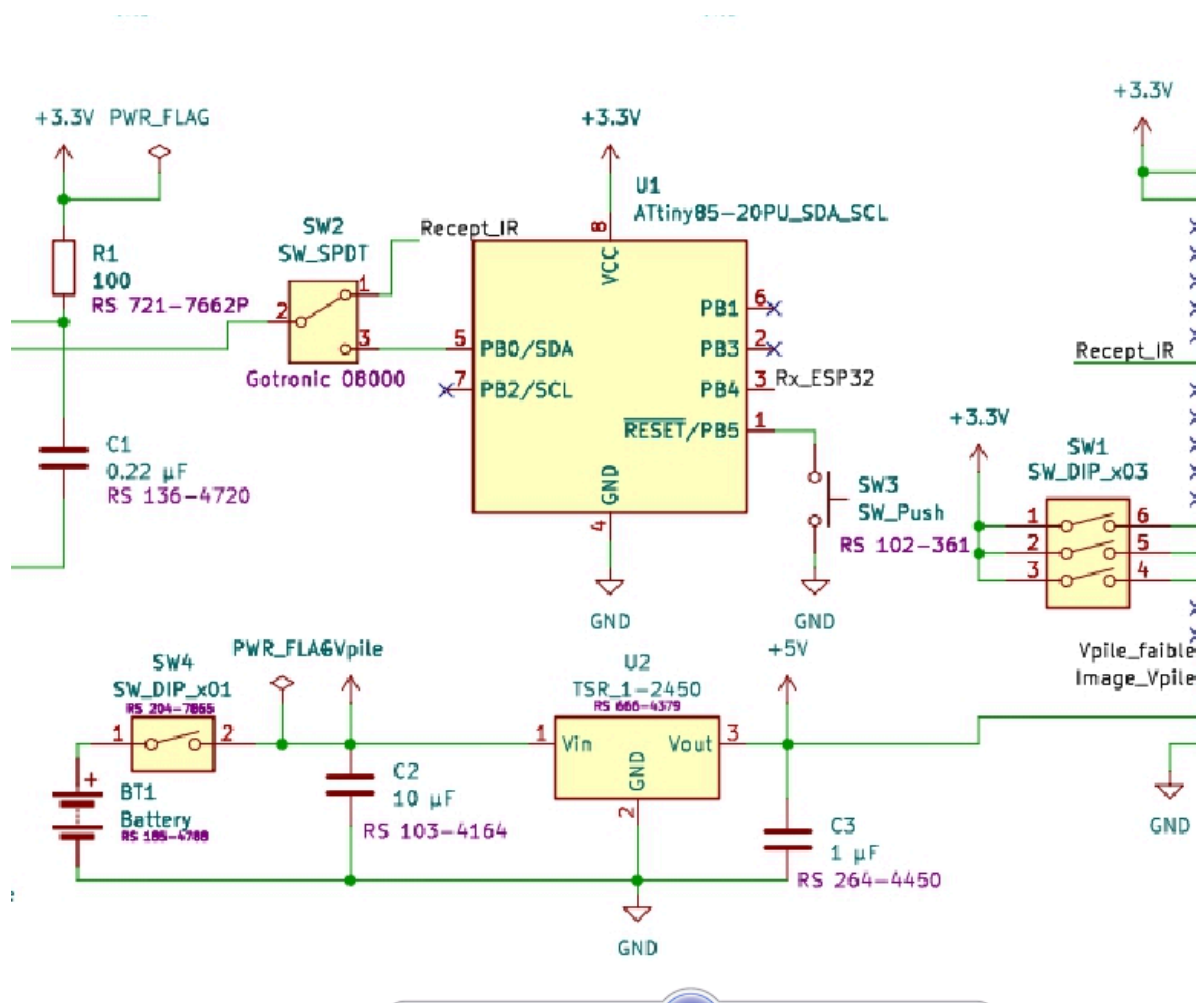
Le choix de l'ESP32 est du :

- sa programmation peut se faire dans l'environnement de programmation Arduino, qui propose plein de bibliothèques,
- la carte DEV Kit C1 donne accès à de nombreux GPIO du microcontrôleur,
- cette carte est facilement disponible chez les distributeurs.
- son prix est abordable,
- **Connexions GPIO :** Utilisés pour interagir avec les LEDs et les capteurs.
- **Alimentation 3.3V et GND :** L'ESP32 fonctionne sous 3.3V, fourni par un régulateur.
- La carte de développement doit être alimentée sous +5V, elle génère du 3.3V, entre autres pour son microcontrôleur.
- **Connecteurs d'entrée/sortie (Image_Vpile, Vpile_faible, etc.) :** Ils permettent de connecter des éléments externes au circuit.

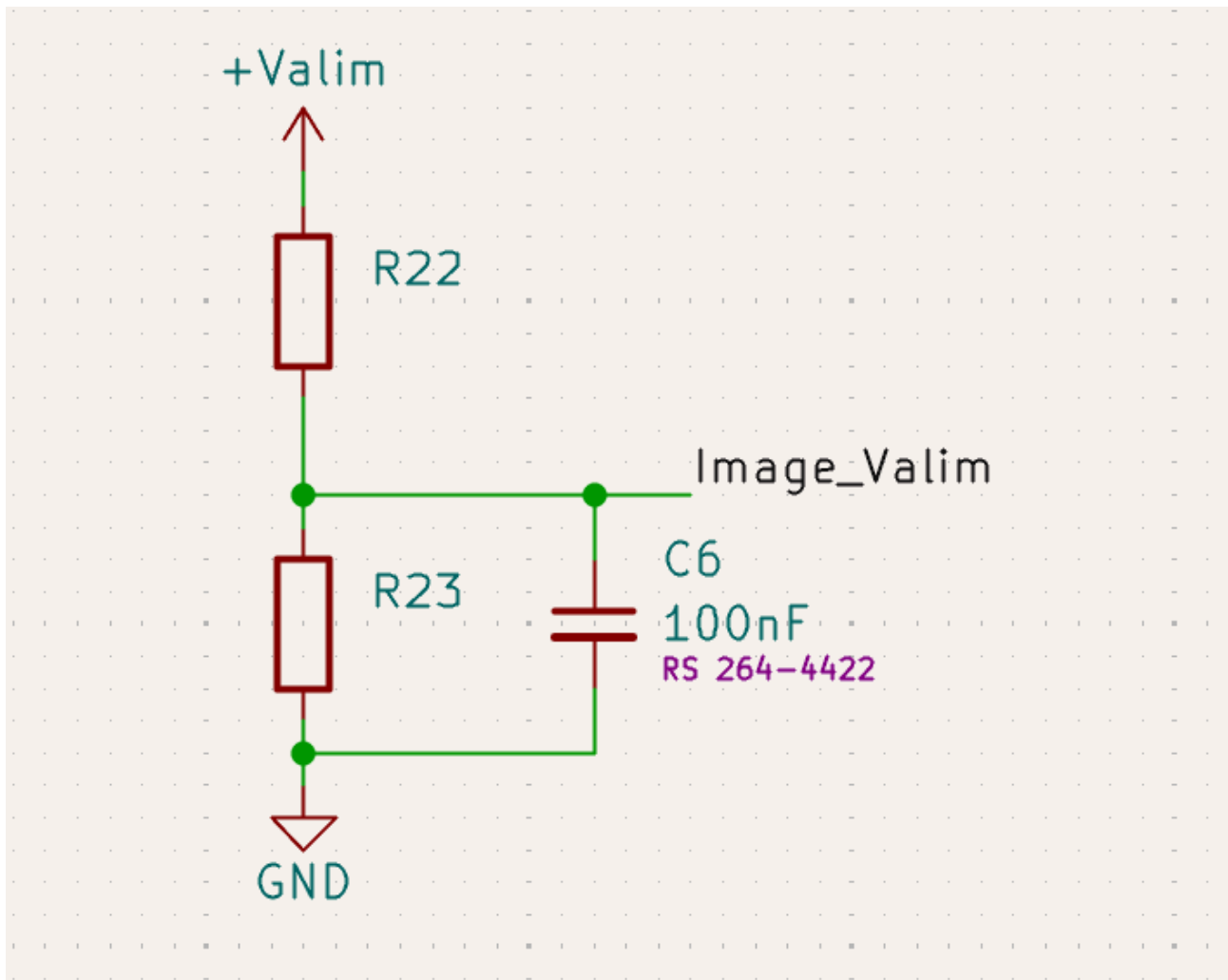


4.4. Communication et Contrôle (ATtiny85)

- **Microcontrôleur ATtiny85 (U1)** : Sur la carte cible, l'ATtiny85 utilise la librairie IRremote pour lire les commandes envoyées en IR via le protocole NEC par la carabine laser. Il transmet ensuite la donnée en UART à l'ESP32-S3, qui, bien que capable de le faire, est déchargé de cette tâche pour se concentrer sur la gestion Wi-Fi en temps réel.
- **Connexions SDA/SCL** : Elles permettent à l'ATtiny85 de communiquer avec l'ESP32 via I2C.
- **Bouton Reset (SW1)** : Permet de réinitialiser le microcontrôleur si nécessaire.
- **Batterie (BT1)** : Source d'énergie principale du système.
- **Régulateur de tension (C2, C3, SW4, etc.)** : Permet de stabiliser la tension d'alimentation pour les différents composants (ESP32 = 5V, LEDs = 5V , ATtiny = 3.3V).

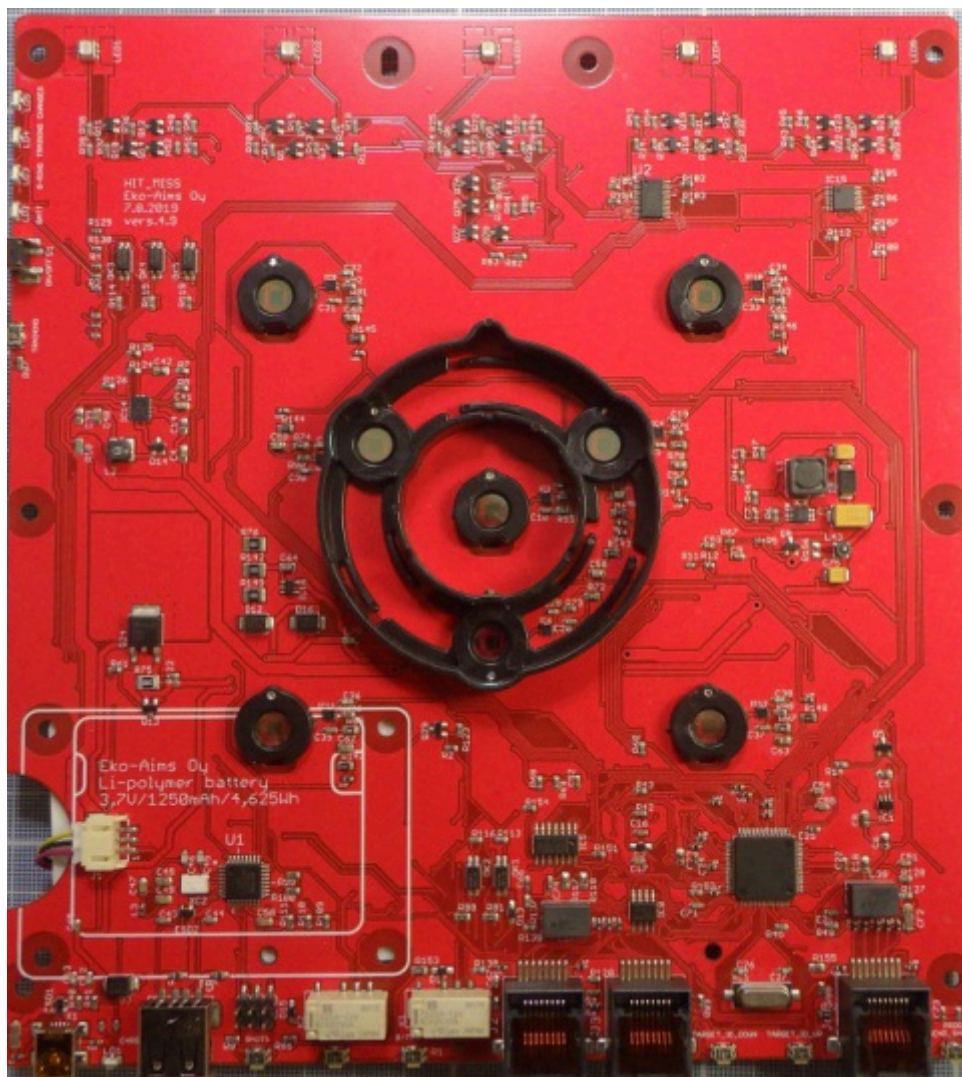


4.5. Pont Diviseur



Un pont diviseur de tension permet à l'ESP32-S3 de surveiller le niveau de la batterie via une entrée analogique. Sur la deuxième version corrigée du PCB, cette mesure est faite sur l'entrée GPIO03, ce qui évite d'utiliser l'ADC2 et permet ainsi de conserver l'usage du Wi-Fi, indispensable au fonctionnement du système.

5 - Carte de la Cible



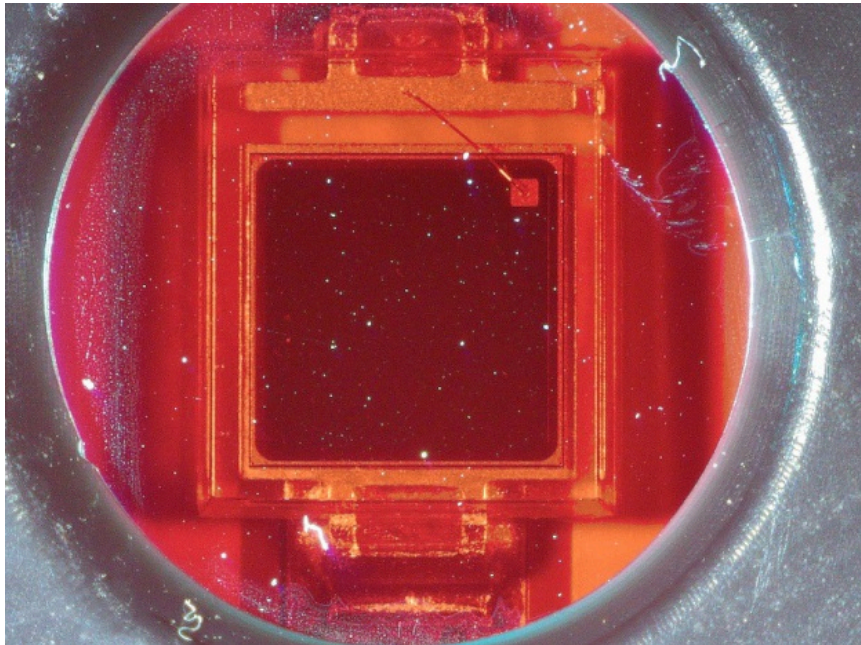
Voici la carte qui se trouve à l'intérieur de la cible qui est la raison pour laquelle la cible fonctionne.

<https://www.ecoaims.com/products/laser-pistols-and-optical-pistols/laser-pistols-and-targets/pp320exp-4-pack-club-cases-pro-smart-phone-compatible-copy/>

Les petits cercles qui se trouvent vers le centre sont les récepteurs infrarouge qui permettent de dire si oui ou non on a touché la cible . La carte possède un fond rouge car cela permet de mieux réfléchir la lumière rouge , vers le revêtement réfléchissant de la face avant .



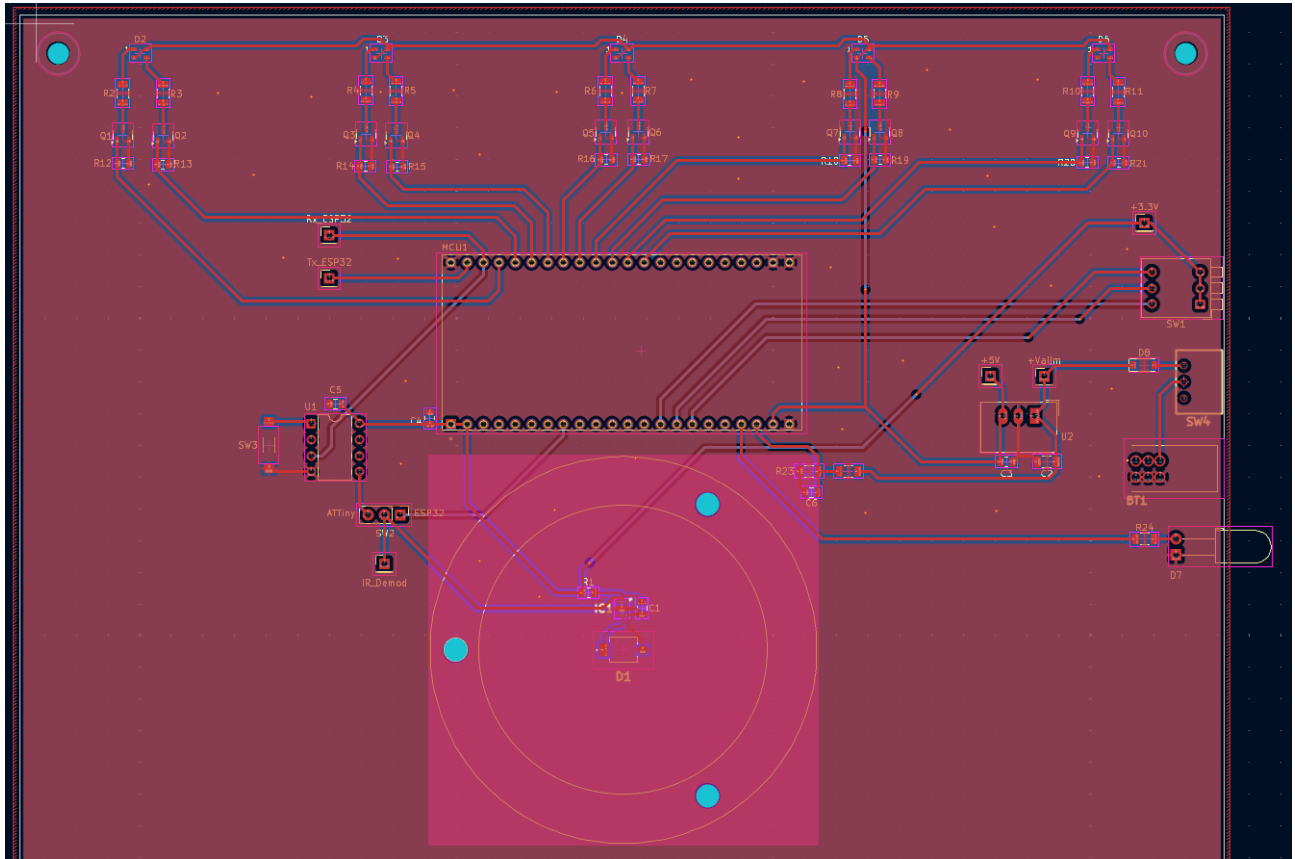
*Le code **3338** est un **code de marquage abrégé** utilisé par plusieurs fabricants pour identifier des composants. Grâce à notre rétro-ingénierie de la carte précédente, nous avons identifié et réutilisé le même démodulateur pour photodiode. Celui-ci permet de supprimer la porteuse à 38 kHz du signal infrarouge reçu — d'où le "38" dans la référence "3338".*



Cette image montre une photodiode vue en gros plan, via la loupe électronique du lycée. La structure centrale carrée, entourée de contacts métalliques, est la zone sensible à la lumière. L'éclat rouge et les reflets lumineux indiquent un filtrage spectral (probablement un filtre rouge).

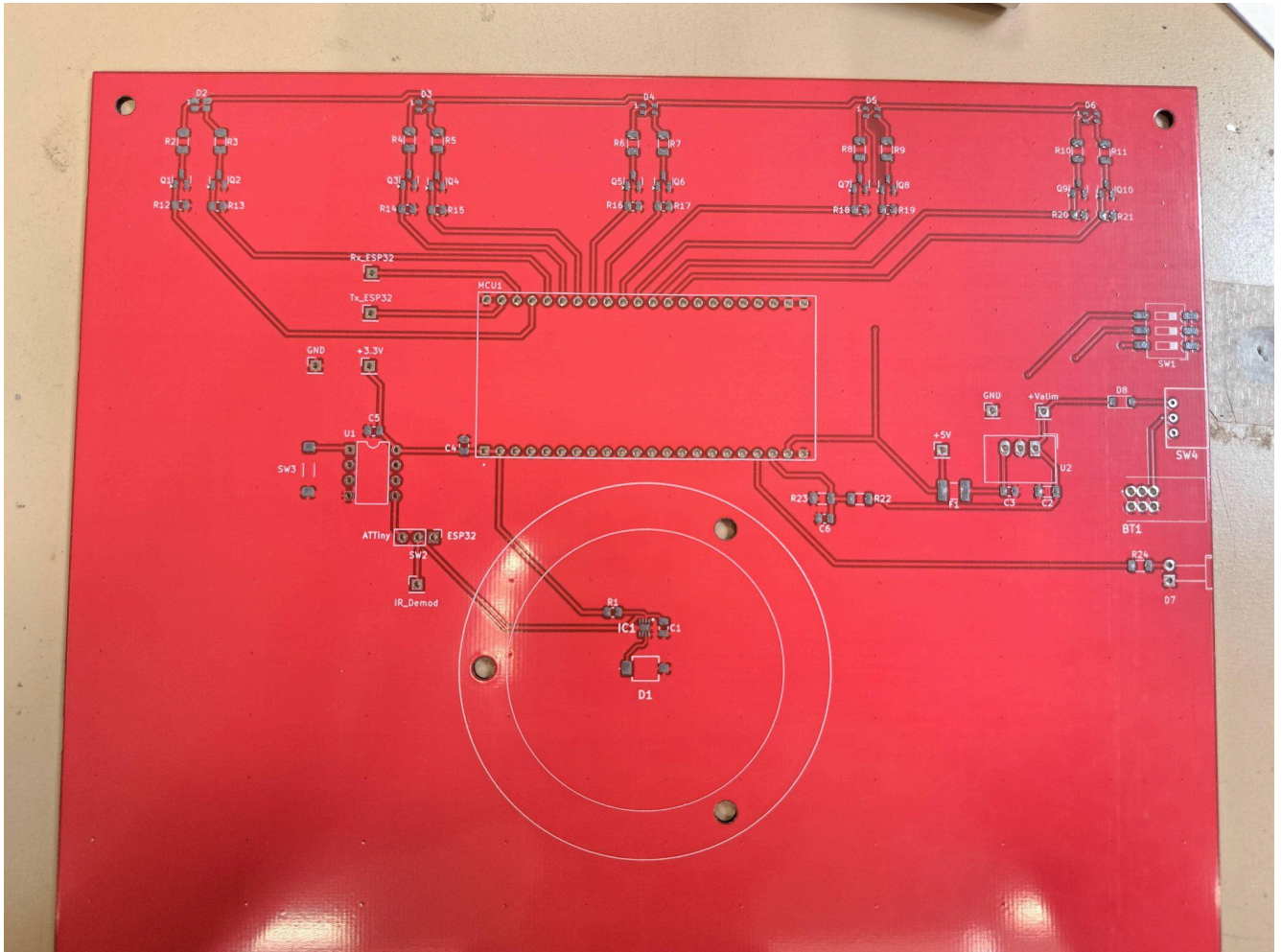
Le problème, cependant, est que nous n'avons pas trouvé le même verre rouge. Nous avons donc opté pour une lentille qui permet de réfléchir le faisceau laser vers le centre, peu importe l'endroit où il touche la surface. Ayant réduit le nombre de capteurs à un seul, placé au centre, au lieu des huit initialement répartis sur toute la carte, cela facilite donc la réception du faisceau laser.

5.1. Routage final de la Carte



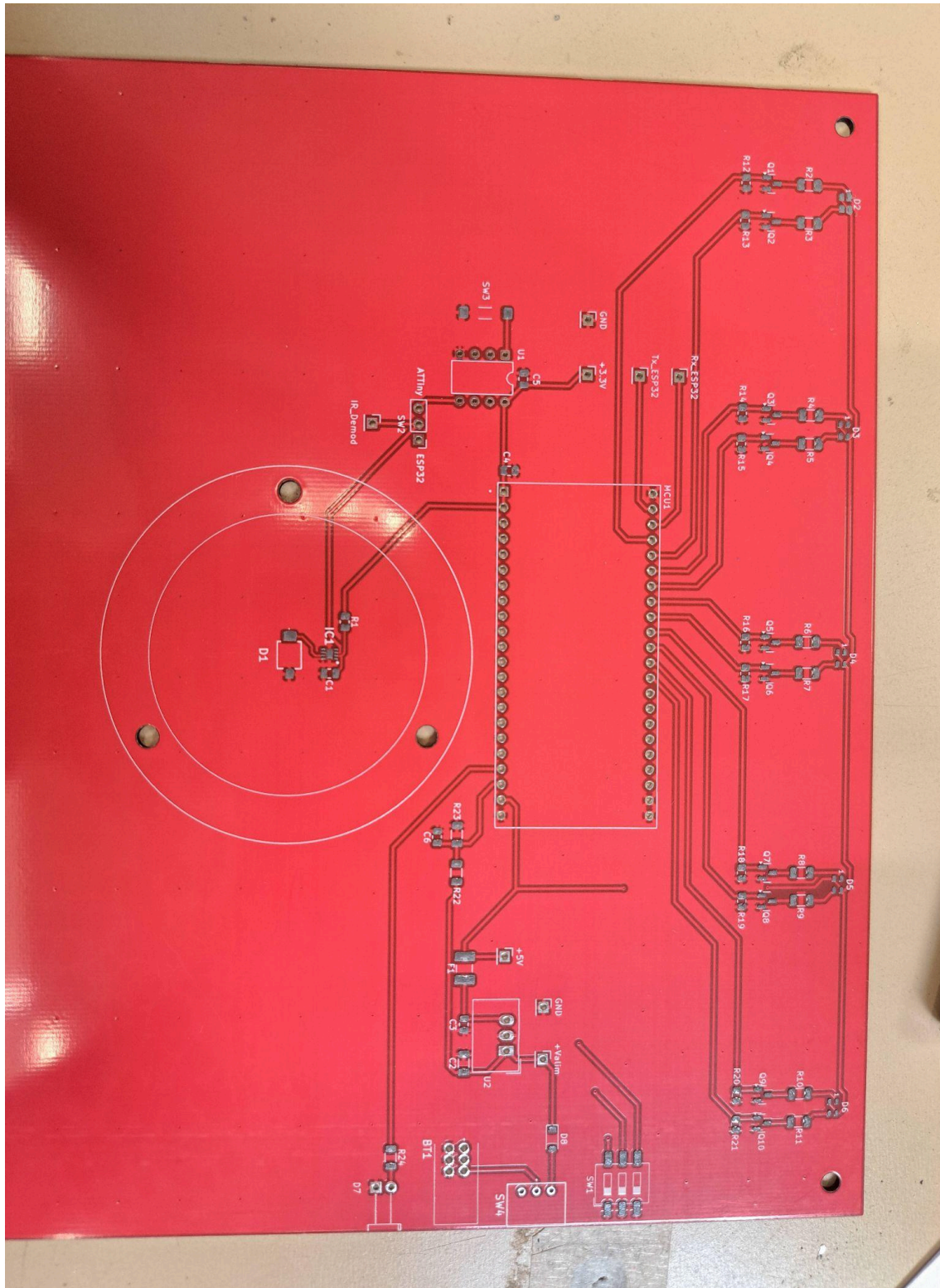
Le routage de la carte a été effectué de manière à organiser clairement les pistes entre les composants. Les connexions sont bien réparties et les pistes ne se croisent pas inutilement. Les signaux de puissance et de commande sont séparés pour éviter les perturbations. Le placement des composants facilite aussi le cheminement des pistes et réduit les risques d'erreurs lors de la soudure. Même si je possède une carte très grosse qui facilite aussi la vie niveaux soudure .

5.2. Carte avant soudure



La carte a été correctement fabriquée, avec des sérigraphies claires. Les zones sont bien définies, ce qui facilitera la soudure et les futurs tests. Aucun défaut visuel n'est présent sur les pistes ou les pads. Cette carte servira de base pour le montage final et les essais fonctionnels.

5.3. Carte après application de la pâte à braser



5.4. Application de la pâte à braser

1.Préparation :

On nettoie la carte PCB avec de l'alcool isopropylique puis on la fixe sur un support avec du ruban double face pour éviter que la carte ne bouge .

2.Positionner le pochoir :

On aligne soigneusement un pochoir en inox sur le PCB on s'assure que les ouvertures du pochoir correspondent exactement aux pads CMS .

3.Application de la pâte à braser :

On met une petite quantité de pâte à braser sur une raclette puis on étale uniformément la pâte tout en appuyant fermement , faire un seul passage si possible pour éviter que la pâte ne déborde sur la carte , puis retirer le pochoir doucement verticalement .

4.Placement des composants CMS :

Utilisation d'une pince fine pour déposer les composants sur la pâte , on aligne les composants le plus précisément possible même si la pâte corrigera légèrement au moment du chauffage

5.Chauffage des composants (Reflow/refusion) :

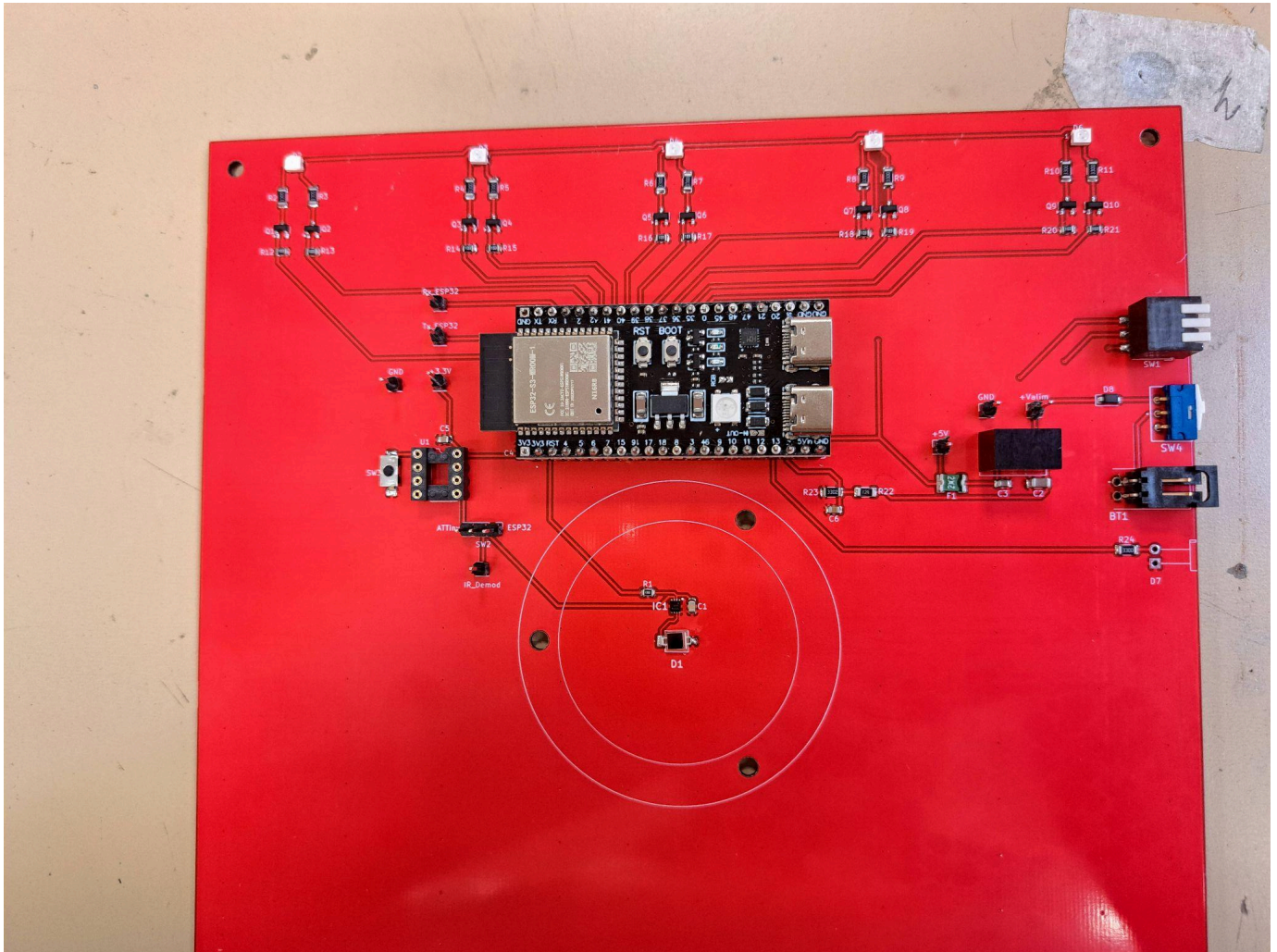
On utilise normalement un four à refusion qui chauffe entre 220-240°C max selon la pâte mais ici ma carte est bien trop grande nous avons alors utilisé une station à air chaud et nous avons chauffé progressivement en balayant chaque zone .

6.Inspection :

On vérifie ensuite à l'œil nu ou à la loupe les soudures et reprend manuellement à la panne si nécessaire .

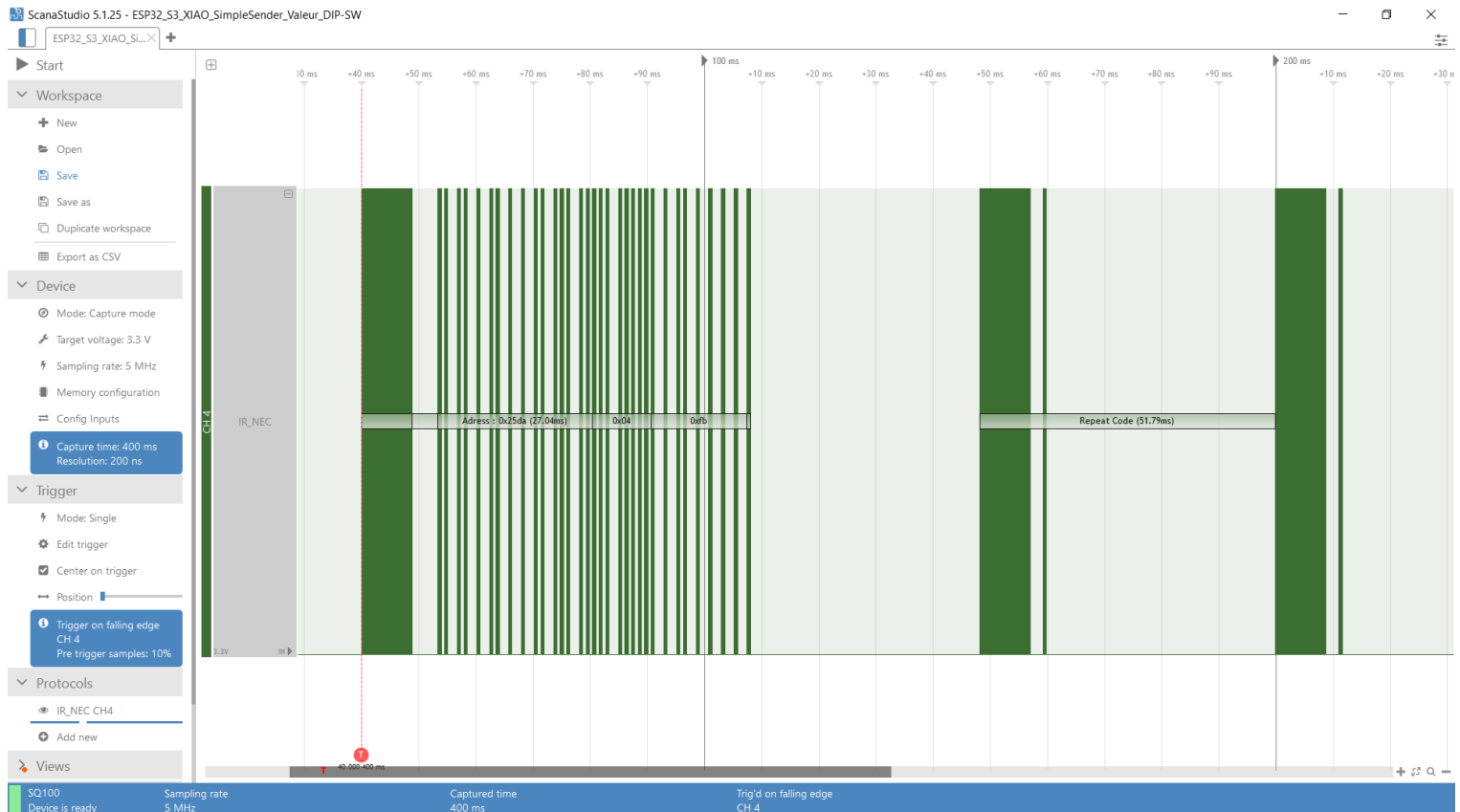
FIN

5.5. Carte après soudure



J'ai espacé un maximum les composants pour faciliter le dépannage et la lecture du schéma , il manque la LED qui s'allume en rouge lorsque la cible atteint un certain pourcentage de batterie .

6 - Analyse trame Essais ESP32-S3



Lors des tests de transmission infrarouge entre l'ESP32-S3 et un récepteur IR, une trame a été capturée à l'aide du logiciel ScanaStudio. La première séquence détectée correspond à une trame complète, contenant :

Adresse : 0x25DA

Commande : 0x04

Commande inversée : 0xFB (complément logique de 0x04)

La durée totale de cette trame est d'environ 27,04 ms.

Par la suite, plusieurs codes de répétition ont été détectés, avec une durée approximative de 51,79 ms chacun. Ces codes sont automatiquement émis par l'émetteur infrarouge tant que le bouton reste appuyé. Ce mécanisme permet d'éviter de réémettre la trame complète à chaque répétition, ce qui est une caractéristique standard du protocole NEC

7 - Suivi du Projet

6 janvier 2025

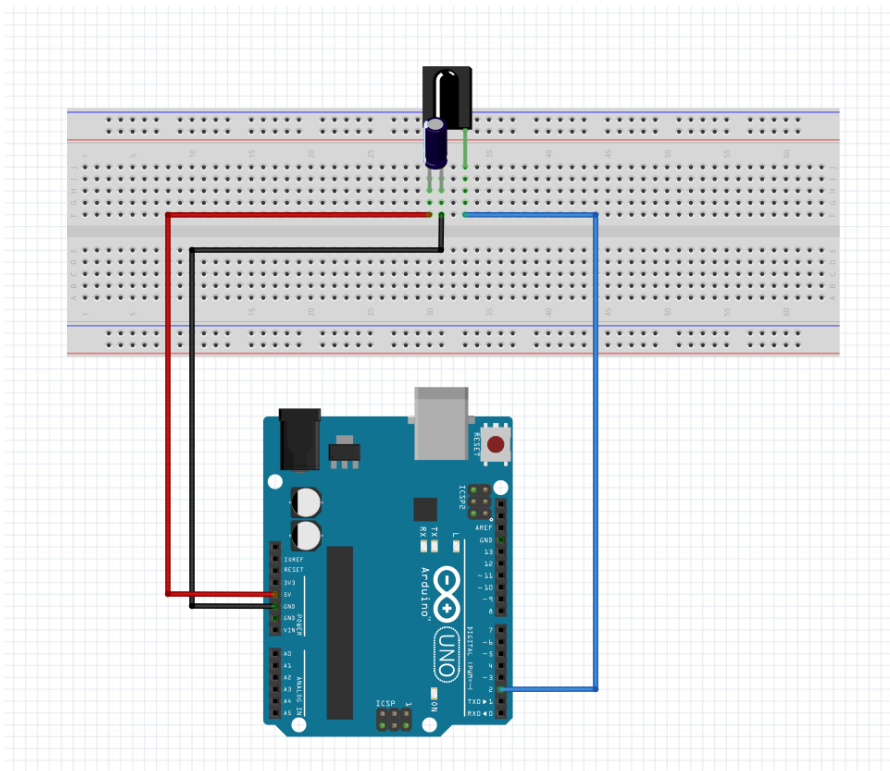
Réunion de lancement, prise de notes sur ma partie. Configuration de GANTT.

7, 13 et 14 janvier

TP sur la communication infrarouge entre cible et pistolet (38kHz, portée 10 m).

27 janvier

Premier câblage sur table puis sur Fritzing. Ajustements à prévoir (inversion de broches bleu/rouge).



28 janvier

Rencontre avec le prof de sport pour mieux cerner les attentes.

Problème identifié : risque de comptage multiple du signal.

Tests avec deux Arduino (émetteur/récepteur) et deux bibliothèques :

- **IRremote** (émetteur) : peu fiable en réception.
- **IRMP** (récepteur) : plus précis.
Prochaine étape : tester IRMP sur les deux cartes.

3 février

Changement de carte pour un **ESP32-S3-WROOM-1**, l'ancienne étant incompatible avec la bibliothèque.

10 mars

Réalisation du schéma de la carte.

Décision prise en réunion de conserver l'ATtiny pour soulager l'ESP32.

Test de réception du signal avec Badr.

11 mars

Tests du capteur à 10 m. Réception stable.

Début du **routage** malgré des incertitudes sur l'alimentation et la taille de la carte.

Mise à jour des symboles et empreintes.

18 mars

Mesures prises sur une carte de référence (biathlon).

Reprise du routage avec correction des empreintes.

24 mars

Empreintes finalisées. Recherche de solutions pour concentrer le faisceau laser sur le capteur : test d'une loupe et d'un autre composant.

1er avril

Routage terminé et envoyé pour validation.

Deuxième **oral de projet** effectué.

29 avril

Nomenclature complétée (références, prix, codes).

Début de l'assemblage de la carte :

- Nettoyage à l'alcool isopropylique
- Application de pâte à braser
- Positionnement des composants
- Fusion avec pompe à air chaud
- Soudure finale (LEDs, etc.)

8 - Nomenclature de la carte

Repère	Désignation du matériel	Valeur	Tol.±	Equivalent	Fournisseur	Boîtier	Remarques	Qté	Condition	ref rapide	Prix U HT	Prix T HT
R1	Résistance CMS Panasonic série ERJ2PC ±1% 0.5W	100 Ohms			Radiospares	CMS 0805		1	Par 5	721-7662P	0.090	0.090
R2 à R14, R24	Résistance CMS Vishay série CRC07 ±1% 0.25W	330 Ohms			Radiospares	CMS 1206		11	Par 50	679-2049	0.026	0.286
R12 à R21	Résistance CMS Yageo ±5% 0.25 W0.125 W	10K Ohms			Radiospares	CMS 0805		10	Par 100	199-5760	0.008	0.080
R22, R23	Résistance CMS Vishay série CRC07 ±1% 0.25W	~ Ohms			Radiospares	CMS 1206		2	Par 50	679-2049	0.026	0.062
C1	Condensateur céramique multicouche 25V c.c.	0.22 µF			Radiospares	CMS 0805		1	Par 50	136-4720	0.132	0.132
C2	Condensateur céramique multicouche 50V ±10%	10 µF			Radiospares	CMS		1	Par 20	103-4164	0.235	0.235
C3	Condensateur céramique multicouche MLCC, 10V c.c., -20 – +80 1 µF				Radiospares	CMS		1	Par 25	264-4450	0.042	0.042
C4	Condensateur céramique multicouche MLCC, 10V c.c., -20 – +80 1µF				Radiospares	CMS		1	Par 25	264-4450	0.042	0.042
C5, C6	Condensateur céramique multicouche MLCC, 50V ±20%	100nF			Radiospares	CMS		2	Par 25	264-4422	4.170	8.340
D1	Photodiodes Si PIN Photodiode	SFH2401			Mouser			1	Par 1	720-SFH2401	1.590	1.590
D2 à D6	LED Vert, Rouge, CMS, PLCC4, 2 LEDs, 24V	LED Vert Rouge			Radiospares	CMS		5	Par 50	127-8431	0.260	1.300
D7	LED Rouge, Traversant, 5 mm (T-1 3/4), 2.25 V	ROUGE			Radiospares			1	Par 5	228-5988	0.436	0.436
D8	Diode CMS Nexperia, 1A, 30V, SOD-123	PMEG3010ER			Radiospares	CMS		1	Par 50	816-6855	0.145	0.145
F1	Fusible réarmable Littelfuse, 30V c.c.	400mA 0.2A			Radiospares			1	Par 10	517-7105	0.369	0.369
IC1	Récepteur IR, VSOP38338, 38kHz	VSOP38338			Radiospares			1	Par 1	257-6989	0.680	0.680
MCU1	Carte ESP32-S3-DevKitC-1 S3-32MB	ESP32-S3 DevKitC-1-3			GoTronic			1	Par 1	37729	26.900	26.900
SW1	Commutateur DIP, DIP, 100 mA @ 5V c.c., 26 mA @ 24V	SW_DIP_X03			Radiospares	CMS		1	Par 2	155-8925	1.935	1.935
SW2	Connecteur HE14 MH100	SW_SPDT			GoTronic			1	Par 1	GoTronic 08000	0.600	0.600
SW3	Interrupteur tactile C & K CMS, SPST, 6 x 3.8 x 2.5mm,	SW_Push			Radiospares	K CMS		1	Par 10	102-361	0.631	0.631
SW4	Commutateur à glissement, SPST, Verrouillable 0.3 A	SW_DIP_X01			Radiospares			1	Par 5	913-9069	1.002	1.002
Q1 à Q10	MOSFETs N-CH 60V 200mA SOT-23	2N702			Mouser			10	Par 1	SN7002NHH6327X	0.228	2.28
TOTAL HT FEUILLE											47.167	
TOTAL HT GLOBAL											47.167	
COEFFICIENT TVA										1.2	56.600	
Crée :	Hortoland C.		01/19/2018		112							
Mis à jour :	Torreillas T		04/28/2025									

[illegible]

9 - Fiche de mise en service Cible Biathlon

Matériel nécessaire :

carte cible câblée	Carte pistolet/carabine fonctionnelle
ATtiny85 + programmeur SparkFun	ESP32-S3 Dev Kit C1

Important : lorsqu'on enlève la carte ESP32 de son support, ne pas faire levier sur l'antenne du module ESP32-S3 !

Procédure à suivre :

- ☐ Le circuit imprimé étant intégralement câblé, mais les circuits U1 (ATtiny85) et MCU1 (ESP32-S3 Dev board) non disposés sur leur support.
- ☐ Utiliser un multimètre en position test diode pour vérifier que les LED Rouge et Verte de D2 s'allument lorsqu'on présente le + du multimètre sur le point test +5 ; et le – sur l'extrémité basse de R2 puis de R3.
- ☐ Faire de même pour D3, D4, D5 et D6.
- ☐ Utiliser une batterie 12V chargée ou une alimentation de laboratoire réglée entre +12V et +13,5V raccordée sur le connecteur BT1.
- ☐ Mettre SW4 sur ON.
- ☐ Vérifier entre les points de test +5V et GND qu'une tension de +5V est présente. De même entre les broches 21 et 22 du support de MCU1.
- ☐ Mettre SW4 sur OFF.
- ☐ Si l'ATTiny85 est neuf, graver la séquence d'initialisation en utilisant le programmeur SparkFun avec l'IDE Arduino.
- ☐ Puis programmer l'ATTiny85 avec le sketch contenu dans le dossier :
"ATTiny85_IR_NEC_UART_projets_2025.ino"
- ☐ Disposer l'ATTiny85 sur son support en veillant à le mettre dans le bon sens.
- ☐ Programmer la carte ESP32-S3 Dev Board par l'intermédiaire du connecteur USB avec le sketch
"ESP32_Dev_Recep_Code_ATTiny85_Detect_Batt_Faible_GPIO03.ino"
- ☐ Puis débrancher le câble USB. Insérer la carte ESP32-S3 sur le PCB de la carte cible, en veillant à la mettre dans le bon sens.
- ☐ Mettre la carte cible sous tension.
- ☐ Les LED doivent être testées automatiquement, et une sous alimentation signalée brièvement (toutes les LED en orange) si elle est détectée.
- ☐ Disposer les interrupteurs de SW1 à l'identique de ceux de la carte de la carabine.
- ☐ Alimenter la carte carabine, viser la cible et appuyer sur la détente. Une nouvelle LED verte doit s'allumer pour chaque cible réussie.

Benoit Gouley

Partie personnelle – Benoît Gouley

Sommaire

1. Partie commune.....	
1.1. Présentation générale.....	
1.2. Répartition des tâches.....	
1.3. Compte rendu de réunion.....	
1.4. Fiches recettes.....	
2. Introduction au projet.....	
3. Présentation général.....	
3.1. Fonctionnement du système.....	
3.2. Cahier des charge personnel.....	
4. Analyse du projet.....	
4.1. Diagramme SysML.....	
4.2. Diagramme de Gantt.....	
5. Mise en oeuvre du projet	
5.1. Recherche de librairies en C++.....	
5.2. Prise en main.....	
5.2.1. Protocole NEC.....	
5.2.2. Premiers essais.....	
5.2.3. Choix du microcontrôleur.....	
5.3. Xiao ESP32-S3.....	
5.4. Développement de la carte.....	
5.5. Interactions avec la carte.....	
5.5.1. Réarmement, gâchette et mire.....	
5.5.2. Insertion du switch.....	
5.6. Schéma structurel simplifié.....	
5.7. Relevé ensemble des trames.....	
5.8. Résultat des tests en condition.....	
5.9. Prix de la carte et fiche de dépannage.....	
6. Annexes.....	

2.Introduction au projet

Le projet Biathlon crée un équipement moderne en réponse à la demande du professeur M.Le Duc afin de déployer une activité biathlon qui pourrait être proposée en tant qu'épreuve du baccalauréat. Notre but est donc d'introduire un matériel professionnel à bas coût et facile d'utilisation avec une possibilité de production (fichier gerber). Cela comprend une carabine laser, une cible et un environnement logicielle avec interface de gestion.

3.Présentation Général

3.1. Fonctionnement du système

Lors de la phase de tir (5 tirs), chaque déclenchement de la gâchette est envoyé à la base par la carabine via Wi-Fi, la cible enverra son nombre de tirs reçus de la même manière. Un signal lumineux doit indiquer la réussite ou l'échec sur la cible, la base doit informer la cible si un tir est effectué qu'elle n'a pas détecté pour que celle-ci allume une des 5 LED en rouge. Au bout de 5 essais, la carabine se bloque logiciellement.

Afin d'éviter tout tir sur la cible d'un autre, le signal infrarouge doit être paramétrable sur 3 bits par le gestionnaire de la session. Cette option offre 8 places de participants et se fera via un switch accessible à l'extérieur des deux systèmes électroniques.

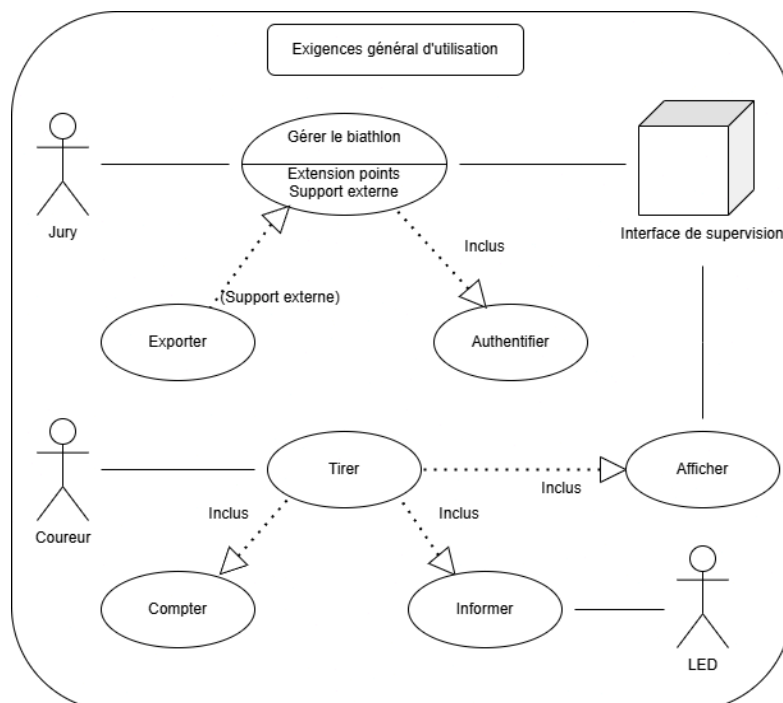
Les résultats sont disponibles au public, juge et arbitre sur une page web en .php (hypertexte processor, langage de programmation pour des pages web dynamiques). L'arbitre peut mettre fin à la course, ajouter des pénalités et sauvegarder les résultats des participants sous format.CVS (Comma-Separated Values ou en français valeur séparées par des point-virgules), ce fichier est en temps réel.

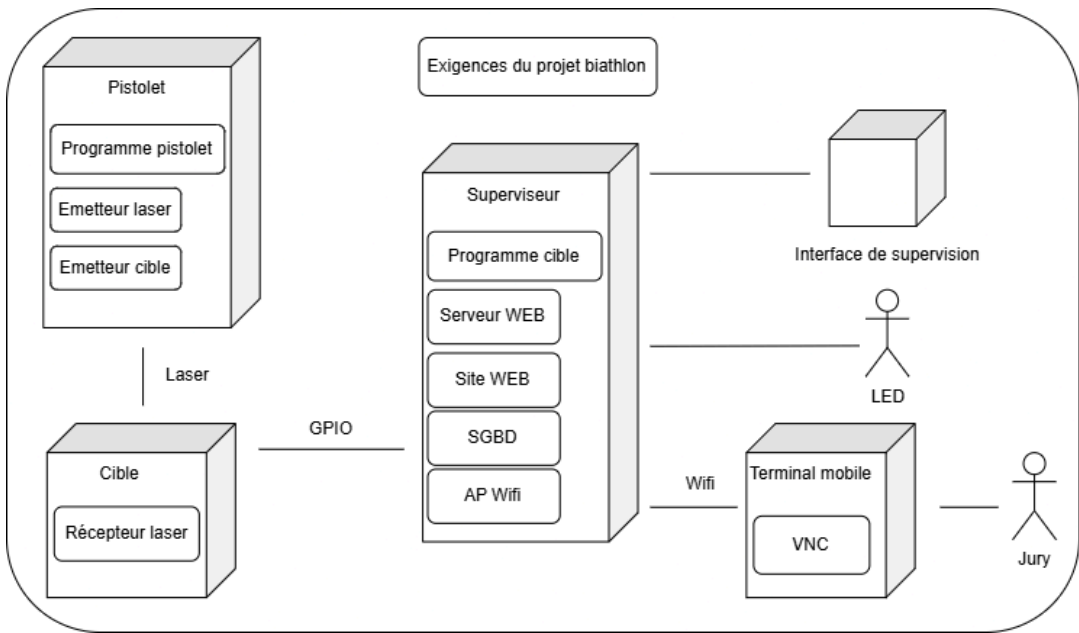
3.2. Cahier des charges personnel

- Concevoir un schéma générant un signal laser de classe 1, appairé à une cible qui ne pourra réagir qu'à ce signal comme réussite, le signal est personnalisable sur 3 bits.
- Effectuer le routage d'une carte adaptée au format de la carabine.
- Produire les fichiers gerber pour que le PCB soit sous-traité.
- Câbler et tester la carte.
- Documenter la mise en service, les éventuels problèmes ou pistes d'amélioration.
- Programmation de la carabine sur l'IDE arduino avec un microcontrôleur ESP32 intégrant une liaison Wi-Fi

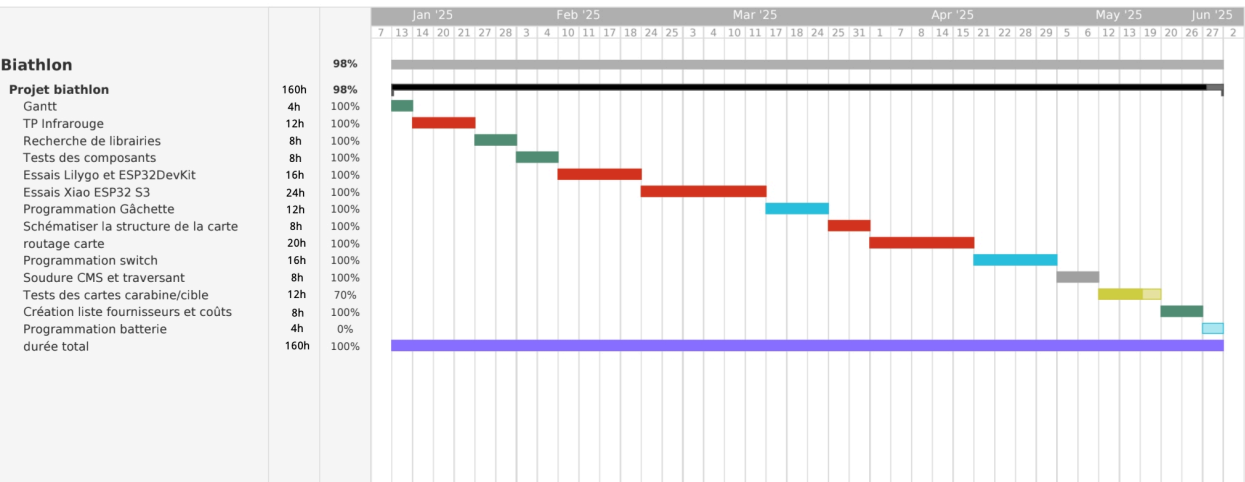
4. Analyse du projet

4.1. Diagramme SysML





4.2. Diagramme de Gantt



5. Mise en oeuvre du projet

5.1. Recherche de librairies en C++

La première librairie utilisée était **IRRemote** pour les exemples “SimpleSender” et “SimpleReceiver” contenant 18 protocoles acceptés pour la partie receveur, le code, bien commenté, permet une première prise en main sans modification.

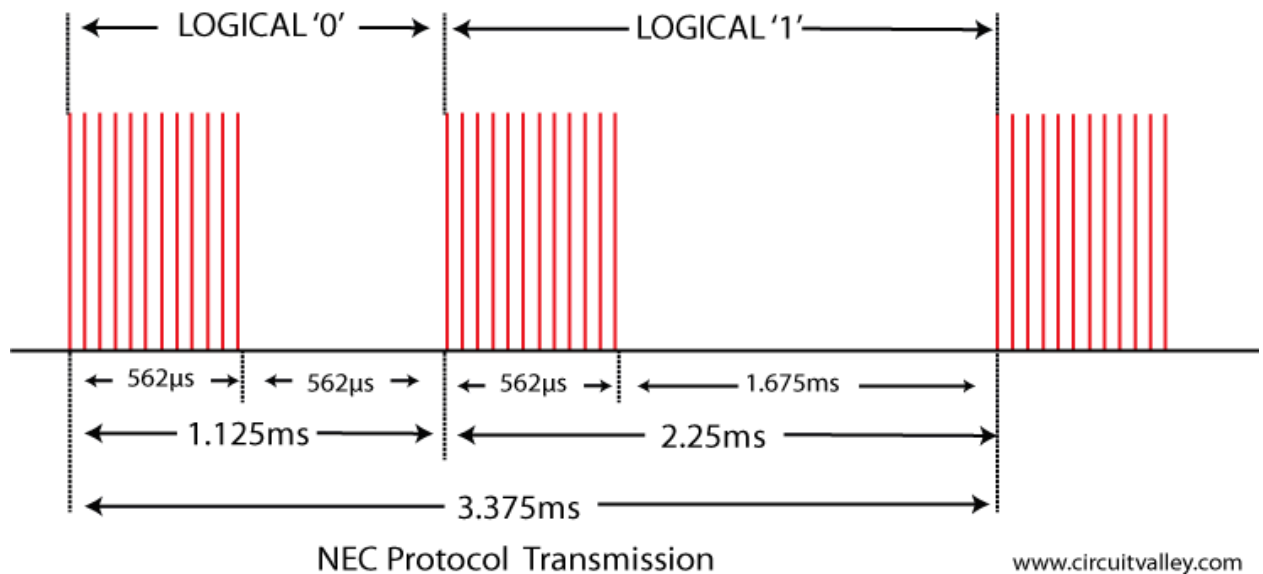
La librairie **IRMP** a été choisie pour la suite de nos travaux, plus claire à la lecture et à la modification. Contrairement à IRRemote, elle permet de modifier la broche de sortie du signal et contient un tableau des broches défini par défauts suivants les microcontrôleurs.

5.2. Prise en main

5.2.1. Protocole NEC

(NEC = Nippon Electric Company)

Afin de transmettre un signal infrarouge clair et qui ne puisse pas être erroné par les néons de l’atelier ou le soleil en extérieur lors de l’activité biathlon, nous utilisons le protocole de transmission infrarouge NEC qui comprend une porteuse de 38kHz. Il est simple, fiable et consomme moins que d’autres protocoles car il est adapté pour des systèmes à pile tel que les télécommandes ou manettes de contrôles sans fil. Le protocole NEC marche avec le codage par distances d’impulsions. L’espace entre les pulsions varie pour représenter la donnée binaire.

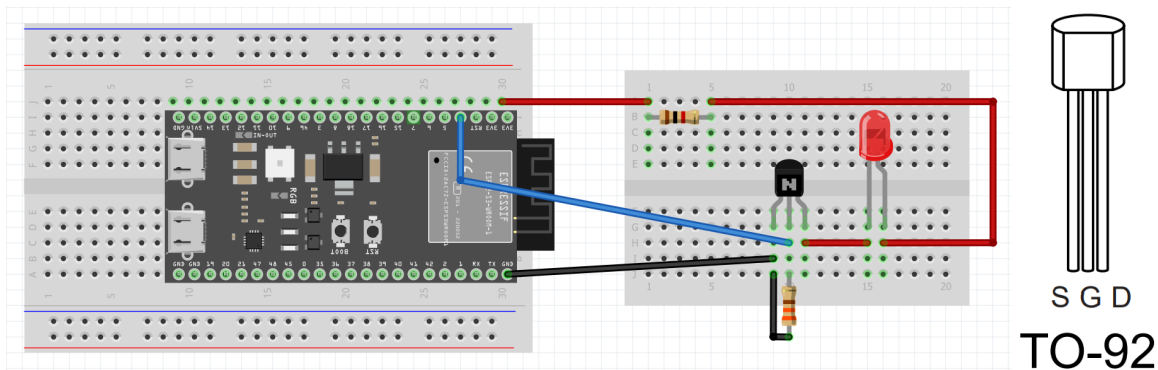


Dans notre cas, le protocole NEC a une adresse et une commande avec leurs complément inverse (4 x 8 bits). Nous n'avons pas à nous soucier de partager les plages d'adresses avec d'autres constructeurs puisque l'usage de notre projet se fait en extérieur loin d'autres systèmes et qu'il ne sera pas commercialisé.

5.2.2. Premiers essais

Le premier choix de carte s'est porté sur l'ESP32S3 T-Display Lilygo mais certaines de ses broches étaient réservées à son écran intégré donc nous sommes passé sur un ESP32S3 DevKit pour la carabine ainsi que la cible, les deux étant disponibles nous avons prit la meilleur pour conduire nos tests, en l'occurrence celle avec le plus de liberté.

Pour le premier montage, on câble le laser sur l'alimentation 3.3V de la carte avec une résistance de 1kΩ en amont pour le protéger, ensuite, on envoie le signal sur le gate du transistor. Ce dernier va venir relier la sortie du laser et le GND de l'arduino selon le code injecté dans l'ESP. On met la Source à la terre et le Drain en sortie du laser. La résistance entre Source et Gate sert à ramener un état bas lors du démarrage.



Le premier programme (simple sender,IRMP) sélectionne la trame en hexadécimal grâce à la ligne : **irsnd_data.command = 0xFB;**

ce signal se répète toutes les 5 secondes **delay(5000);** avec une incrémentation de 1 à chaque itération de void loop() **irsnd_data.command++;**

Le programme originel contient le protocole infrarouge de Samsung, la seule modification apportée au premier code est la suppression des lignes qui y sont directement liées.

PinDefinitionsAndMore.h n'est pas modifié, il sert notamment à définir la broche de génération de la trame suivant les cartes.

Le moniteur série de l'émetteur affiche donc :

START

Send IR signals at pin 3

Protocol=NEC Address=0x707 Command=0xA2 Repeats=2

Protocol=NEC Address=0x707 Command=0xA3 Repeats=2

Protocol=NEC Address=0x707 Command=0xA4 Repeats=2

Au niveau du récepteur pour la cible, l'exemple "SimpleReceiver" est utilisé, les trames sont bien reçues et décodées par le programme suivant l'incrémentation des valeurs hexadécimal.

Afin d'envoyer une valeur hexadécimal de notre choix, on modifie donc

irsnd_data.command = 0x32;

On désactive l'incrémentation et on réduit le délai à 500 pour faciliter les tests et ne plus viser avec une données envoyée toutes les 5 secondes.

Après ces changements le moniteur série affichera **Command=0x32** et le programme ne fera plus évoluer cette valeur.

5.2.4. Choix du microcontrôleur

	ESP32-S3 Lilygo	ESP32-S3 Dev-Kit	XIAO ESP32 S3	Souhaiter
Taille	51 x 25.5 mm	64 x 21 mm	21 x 17.5 mm	Compacte
GPIO	13	36	11	6
Prix	24	27	8	Minimal
consommation	120–220 mA	100-230 mA	100-200 mA	Minimal
wifi	oui	oui	oui	Présent
Langage c++	oui	oui	oui	Présent

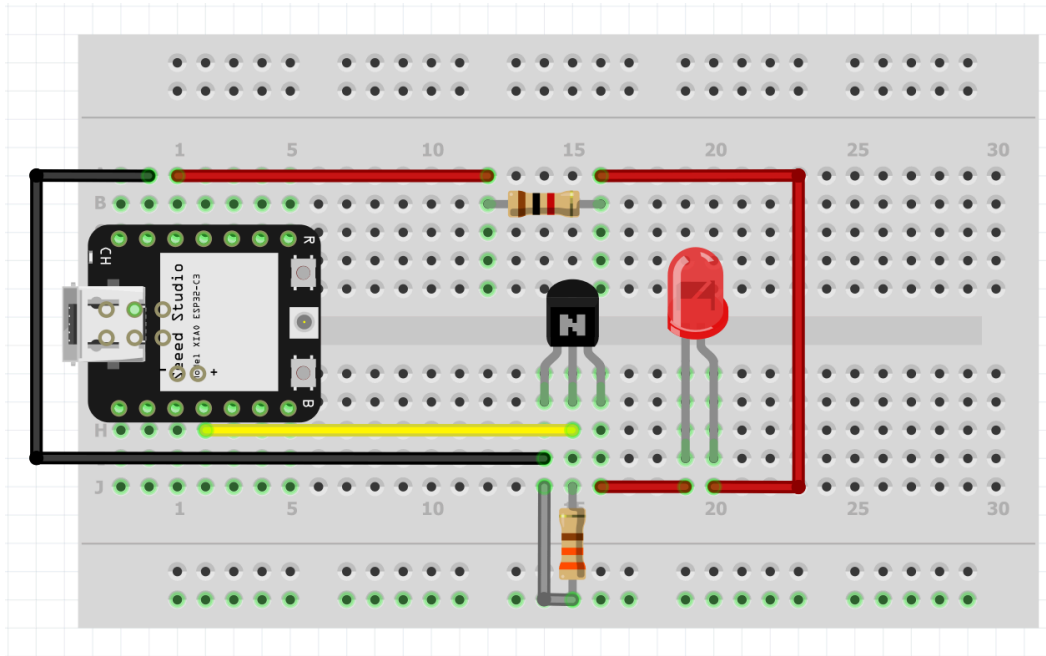
Les trois cartes ont le même processeur ESP32-S3 Xtensa LX7 dual-core qui leur permet d'accepter le langage C++, on peut donc continuer à utiliser la librairies IRMP sans problèmes de compatibilités

5.3. XIAO ESP32C3

Pourquoi changer?

Le Xiao S3 est bien plus petit que le précédent ESP32, il répond toujours à nos besoin : 3 GPIO pour le switch, 2 pour la gâchette et la mire et un pour le contrôle du laser.. Tension de 5V et 3.3V ainsi que les pins RT/TX si besoin pour compléter des tests/vérifications. Le plus important étant sa taille réduite de 21 x 17.5mm contre 62.74 x 25.4mm, soit une réduction de surface de ~76%, ce qui n'est pas négligeable pour la facilité d'insertion de la carte dans la carabine.

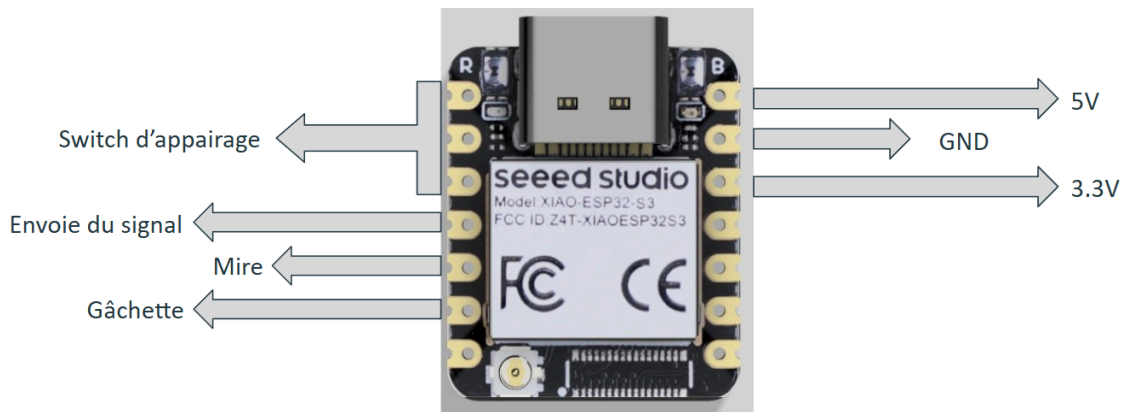
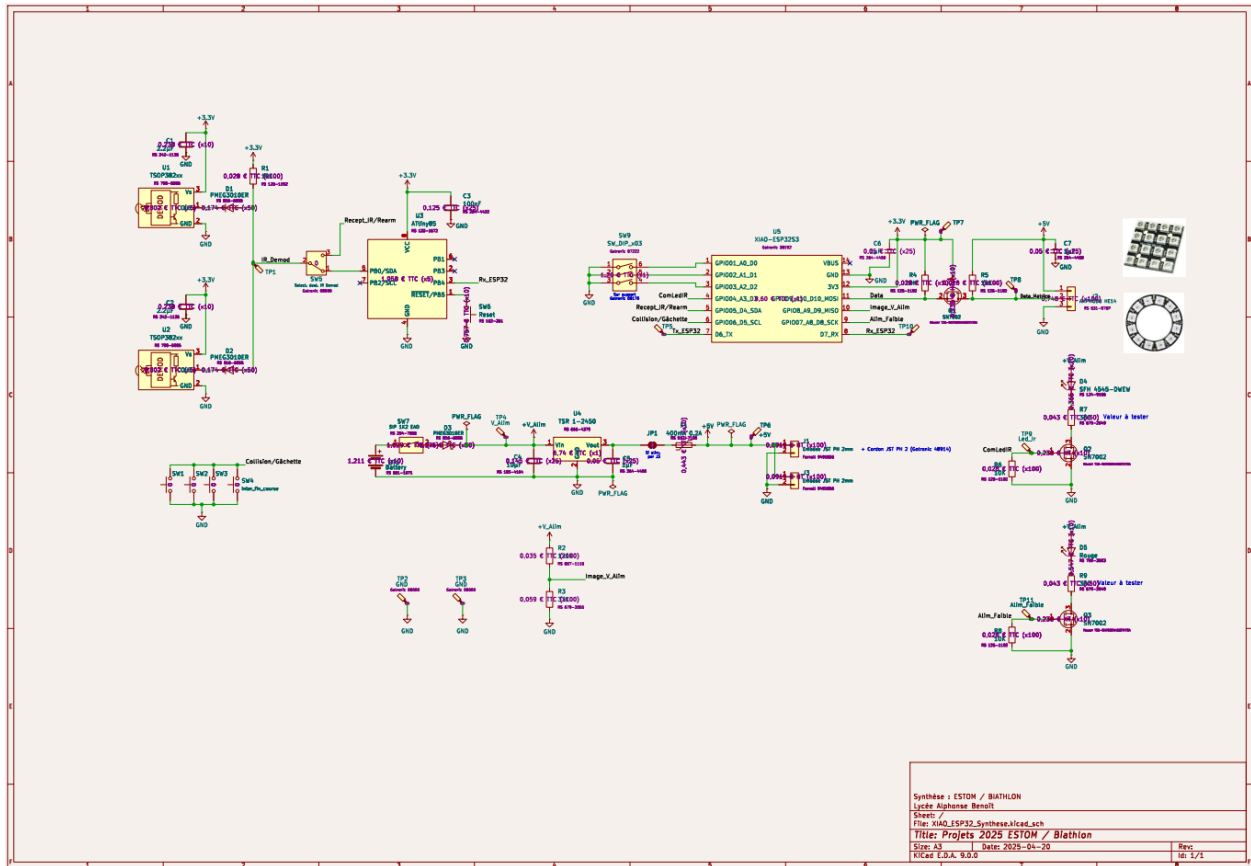
Le montage utilisé est semblable au premier, le GPIO 4 du XIAO S3 est utilisé comme sortie du signal IR.



5.4. Développement de la carte

La carte actuellement utilisée permet aussi à deux autres élèves de travailler sur leur projets basé sur la communication par signaux infrarouges, c'est pourquoi une partie des composants du schéma structurel suivant ne sont attribués à aucun rôles dans mon utilisation personnelle, ce schéma me permet d'aborder le switch, la gâchette et la mire.

Un schéma structurel simplifié est disponible plus bas, avec les composants utiles à mon projet.

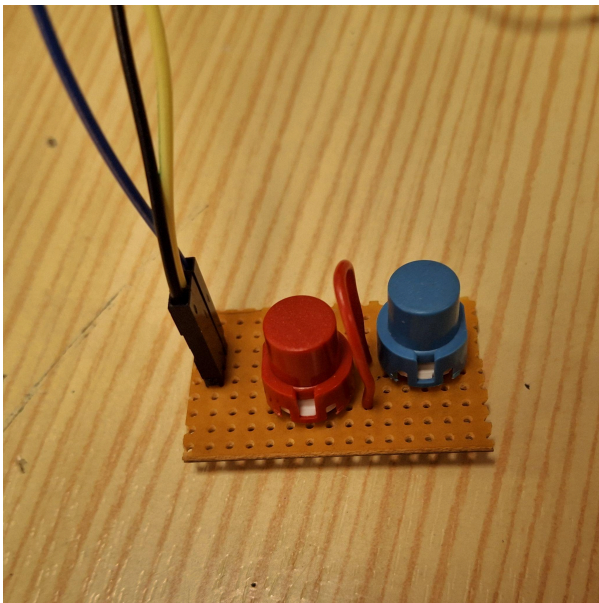


Utilisation des broches du XIAO

5.5. Interactions manuelle de la carte

5.5.1. Réarmement, gâchette et mire

Le réarmement sera mécanique et ne nécessitera pas de programmation ou d'éléments électroniques. La gâchette ainsi que la mire fonctionnent avec deux boutons poussoir simple sur une plaque d'essai à souder.

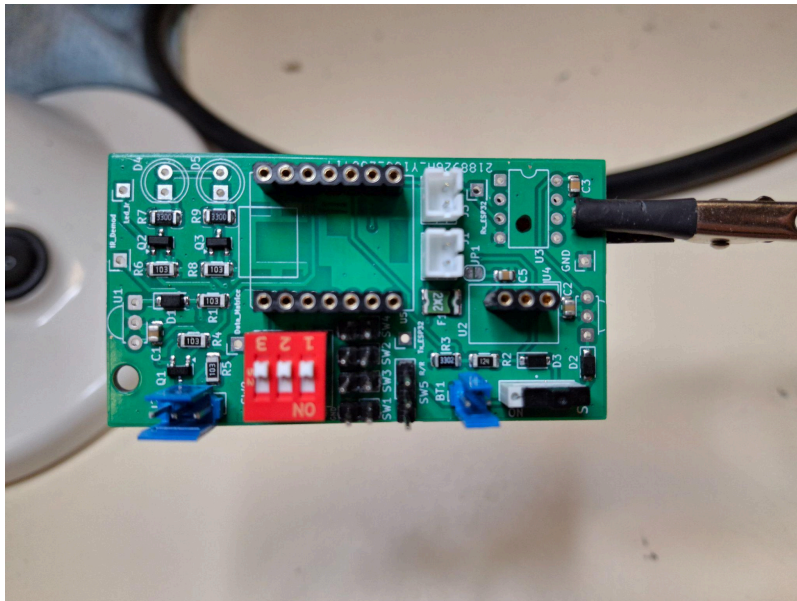


Le bouton rouge, relié à la broche 6 gère la gâchette

Le bouton bleu, relié à la broche 5 permet l'usage de la mire (laser en continue)

```
void setup() {  
[...]  
pinMode(6,INPUT_PULLUP);  
attachInterrupt(6, debounceInterrupt, FALLING); }  
void debounceInterrupt() {  
static unsigned long lastInterrupt = 0;  
unsigned long interruptTime = millis();  
if (interruptTime - lastInterrupt > 200){  
    Serial.print("Gachette utilisée");  
    gachette=true;  
}  
lastInterrupt = interruptTime;  
}
```

5.5.2. Insertion du switch



Extrait du code pour le switch :

```
void loop() {
    delay(50);
    if (mire)
    {
        digitalWrite(4,HIGH);
    }
    else digitalWrite(4,LOW);

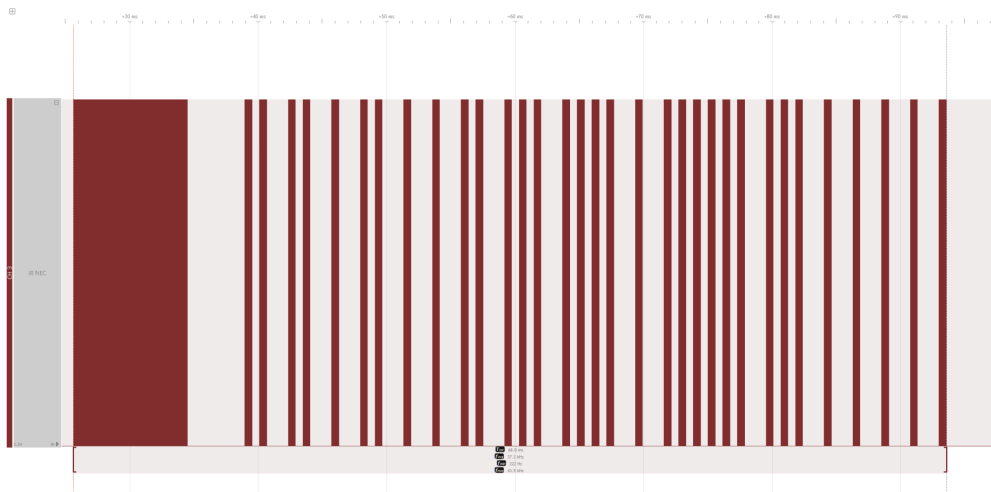
    if (gachette)
    {
        gachette = false;
        DIP_SW = 0;
        if (!digitalRead(1)) DIP_SW +=1;
        if (!digitalRead(2)) DIP_SW +=2;
        if (!digitalRead(3)) DIP_SW +=4;
        Serial.print(" Valeur DIP_SW = ");
        Serial.println(DIP_SW);
        irsnd_data.command=DIP_SW;
        irsnd_send_data(&irsnd_data, true);
        irsnd_data_print(&Serial, &irsnd_data);    }    }
```

Le début du code sert à activer/désactiver le laser en mode continue, il change l'état actuel de la mire. La mire aide lors des tests de distance et servira aux participants du biathlon.

Lorsque la gâchette est pressée, la variable correspondante est remise à `false`, associée au `delay(50)`; cela permet d'éviter tout problèmes de rebonds mécaniques. On vient ensuite lire les trois entrées du switch et on les assigne aux valeurs 1, 2 et 4 pour les additionner a `DIP_SW` ce qui nous donne des valeurs de 0 à 7. On transmet cette valeur avec la ligne `irsnd_data.command=DIP_SW`; qui comme dit précédemment sert à définir la donnée à envoyer en hexadécimal.

On utilise un système Ikalogic pour lire la trame envoyée par le code à l'aide du logiciel ScanaStudio. On vient brancher une des 4 broches d'entrée sur la pin `Led_Ir` que l'on a inséré sur la carte et on relie les deux terres. Pour la fréquence d'échantillonnage, on utilise 1MHz, le déclenchement se fera en ascendant et le protocole utilisé "IR_NEC".

Après ces étapes, on obtient donc une trame binaire comme ceci :



$$T_{\text{total}} = 68\text{ms} ; f_{\text{avg}} = 37.3\text{kHz}$$

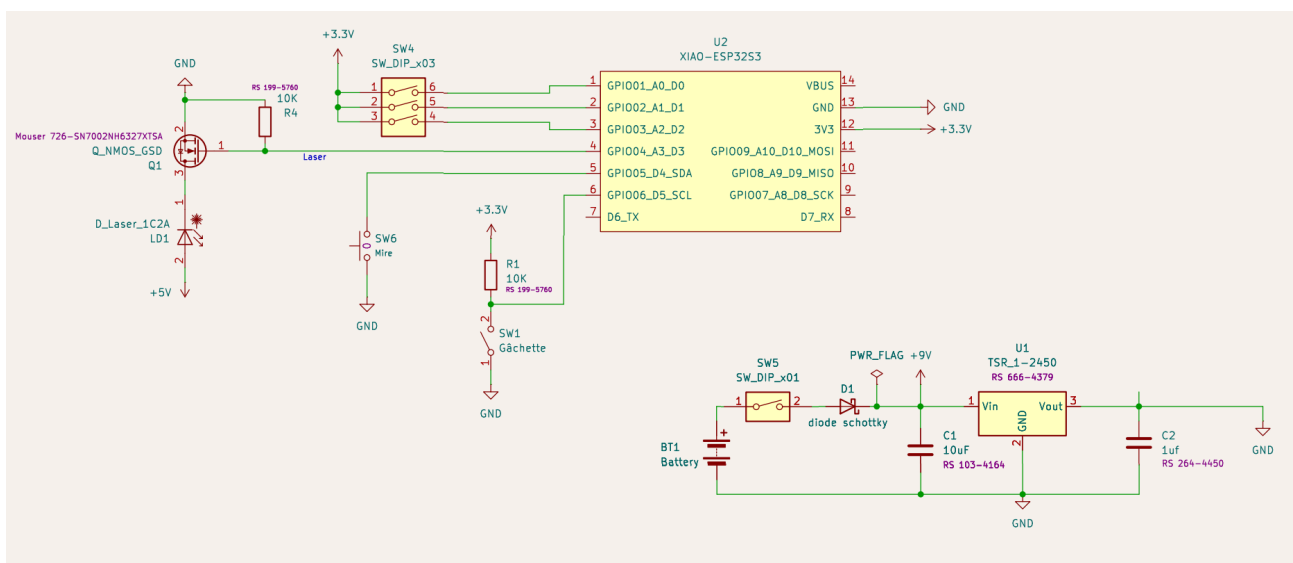
On peut lire :

01011011 10100100 01100000 10011111

	Binaire	Hexadécimal (inversé)
Adresse	0101 1011	1101 1010 = 0xda
Complément d'adresse	1010 0100	0x25
commande	0110 0000	0x06 (valeur switch)
complément de commande	1001 1111	0xF9

5.6. Schéma structurel simplifiée

Les parties de la cartes réellement utilisé pour la carabine :



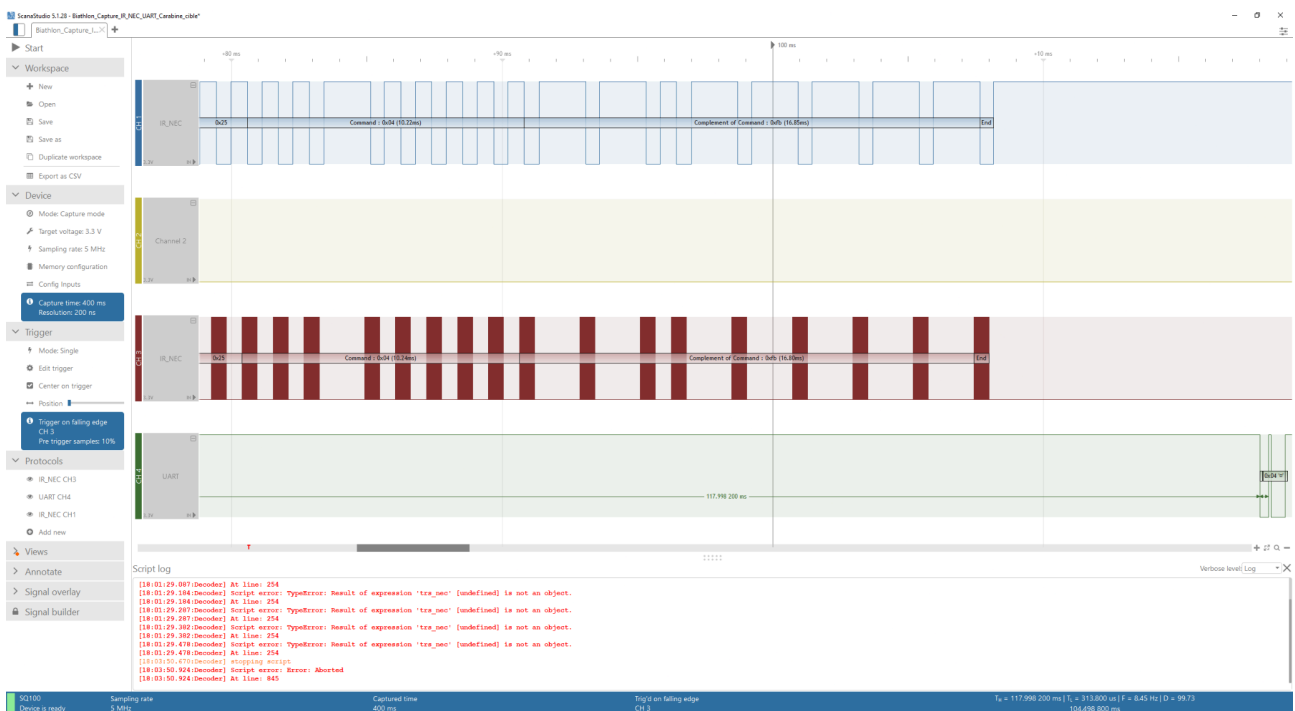
Sur cette deuxième version du schéma, on attribue :

GPIO 1, 2 et 3 au switch ; GPIO 4 au laser

GPIO 5 a la mire ; GPIO 6 a la gâchette

L'alimentation contient une diode schottky pour empêcher le courant de traverser les composants à l'envers si la pile est inversée lors de l'utilisation. U1 est une alimentation à découpage pour adapter la batterie de 9V à la tension de fonctionnement qui est de 5V.

5.7. Relevés des trames émetteur, récepteur et ESP32 de la cible :



Il s'agit d'une capture de trame de l'émetteur (Channel 3) du récepteur (channel 1) et de l'UART de la carte (Channel 4).

On peut remarquer que les états entre le récepteur et la cible sont inversés et décalés d'environ 0,1 ms, le relevé de trame est effectué avec les deux systèmes proches l'un de l'autre dû à l'utilisation de l'ikalogic et ses fils relativement court. On peut ignorer le temps que met le signal à arriver jusqu'au capteur, le délai de 0,1 ms correspond à la démodulation du signal et la suppression de la porteuse de 38kHz.

L'UART représente la réaction du processeur de la cible au signal, il va ensuite allumer une LED en vert ou ne pas réagir si les deux valeurs des switches de la carabine et de la cible ne sont pas identiques.

5.8. Résultats des tests en condition :

En tirant sur...	longue lentille	petite lentille
test intérieur, tir sur capteur à 5 m	déecté	déecté
int capteur 10m	déecté	déecté
int pcb 5m	non déecté	non déecté
int pcb 10m	déecté	non déecté
ext capteur 5m	n'a pas fonctionner en intérieur	déecté
ext capteur 10m	X	déecté
ext pcb 5m	X	non déecté
ext pcb 10m	X	non déecté

5.9. Prix de la carte et fiche de dépannage

5.9.1. Prix de la carte :

Prix des composants total : 27,44 €

Prix de fabrication du PCB (sur JLCPCB) : 3,51 €

Coût total : 30,95 € TTC

	A	B	C	D	E
1	<u>Reference</u>	<u>Qty</u>	<u>Value</u>	<u>Code Commande</u>	<u>Prix Unité</u>
2	JP1	1	SolderJumper_2_Open		0
3	SW1,SW2,SW3,SW4	4	<u>Inter_fin_course</u>	<u>Gotronic 08000</u>	0
4	SW5	1	Select. dest. IR Demod	<u>Gotronic 08000</u>	0
5	TP1	1	<u>IR_Demod</u>	<u>Gotronic 08000</u>	0
6	TP2,TP3	2	GND	<u>Gotronic 08000</u>	0
7	TP4	1	<u>V_Alim</u>	<u>Gotronic 08000</u>	0
8	TP5	1	Tx_ESP32	<u>Gotronic 08000</u>	0
9	TP6	1	+5V	<u>Gotronic 08000</u>	0
10	TP7	1	+3.3V	<u>Gotronic 08000</u>	0
11	TP8	1	<u>Data_Matrice</u>	<u>Gotronic 08000</u>	0
12	TP9	1	<u>Led_Ir</u>	<u>Gotronic 08000</u>	0
13	TP10	1	Rx_ESP32	<u>Gotronic 08000</u>	0
14	TP11	1	<u>Alim_Faible</u>	<u>Gotronic 08000</u>	0
15	R1,R4,R5,R6,R8	5	10K	RS 125-1192	0,028 € TTC (x100)
16	R2	1	120K	RS 807-1110	0,035 € TTC (x100)
17	R7,R9	2	330	RS 679-2049	0,043 € TTC (x50)
18	C5,C6,C7	3	1µF	RS 264-4450	0,05 € TTC (x25)
19	R3	1	33K	RS 679-2055	0,059 € TTC (x100)
20	J1,J3	2	Embase JST PH 2mm	Farnell 9491856	0,0919 € HT (x100)
21	C3	1	100nF	RS 264-4422	0,125 € TTC (x25)
22	C4	1	10µF	RS 103-4164	0,145 € TTC (x25)
23	D1,D2,D3	3	PMEG3010ER	RS 816-6855	0,174 € TTC (x50)
24	Q1,Q2,Q3	3	SN7002	<u>Mouser 726-SN7002NH6327XTSA</u>	0,238 € HT (x10)
25	C1,C2	2	2.2µF	RS 242-1135	0,238 € TTC (x10)
26	D4	1	<u>SFH 4545-DWEW</u>	RS 124-9106	0,365 € TTC (x20)
27	F1	1	400mA 0.2A	RS 517-7105	0,443 € TTC (x10)
28	D5	1	Rouge	RS 769-3583	0,547 € TTC (x10)
29	J2	1	<u>AMPMODU HE14</u>	RS 531-970P	0,748 € TTC (x100)
30	SW6	1	Reset	RS 102-361	0,757 € TTC (x10)
31	U1,U2	2	TSOP382xx	RS 708-5086	0,802 € TTC (x5)
32	BT1	1	<u>Battery</u>	RS 681-1871	1,211 € TTC (x10)
33	SW7	1	<u>SIP 1K2 EAO</u>	RS 204-7865	1,829 € TTC (x5)
34	U3	1	ATtiny85	RS 133-1672	1,958 € TTC (x5)
35	SW9	1	SW_DIP_x03	<u>Gotronic 07222</u>	1.20 € TTC (x1)
36	U4	1	TSR 1-2450	RS 666-4379	6,74 € TTC (x1)
37	U5	1	XIAO-ESP32S3	<u>Gotronic 38192</u>	8,60 € TTC (x1)

5.9.2. Fiche de dépannage préventive

Bloc Alimentation :

BT1 ; C1 ; C2 ; SW5 ; U1 ; Diode Schottky D1

Le principal problème du bloc d'alimentation est lié aux surtensions possibles lors de la mise en place de la batterie ou de sa manipulation.

- Tester la tension délivrée par la batterie.
- Si bonne, tester la tension de la diode, une mauvaise manipulation avec une batterie mise à l'envers peut avoir endommagé ce composant, coupant la liaison interne.
- Les condensateurs peuvent, avec mauvaise manipulation, surchauffer et ne plus fonctionner correctement, voir brûler.
- Tester les routes de la carte en mesurant les broches de deux composants.

Bloc Laser :

LD1 ; R4 ; transistor MOSFET

- Si le laser ne marche pas, le tester à l'aide d'une autre carte arduino ou ESP, une simple résistance de 330 Ohm avant le laser suffit à le tester en continu sans l'abîmer.
- Si le composant ne marche pas mais est en bon état lors du test, vérifier le programme injecter

Bloc Switch :

5W4

- Vérifier l'état physique des 3 switchs (partie soudure + plastique).
- étant donné que ce bloc n'est fait que d'un composant, si le switch est physiquement en bon état alors il faut vérifier le code injecter. Les pins du switch ne prennent que le pull-down, il faut vérifier si c'est bien dans les lignes de code et non pas pull-up. Si non, alors le problème vient des calculs en décimal qui servent à attribuer les codes.

Bloc gâchette & mire :

R1 ; SW1 ; SW6

- Vérifier la conductivité lorsque SW1 et SW6 sont pressés, les deux sont des composants mécaniques fragiles qui peuvent se casser.
- Vérifier que R1 n'est pas endommagé et que SW1 ne soit pas forcé à l'état haut
- Vérifier le programme

6. Annexes

Code complet de la carabine :

```
/*
 * ESP32_S3_XIAO_SimpleSender_Valeur_DIP-SW_Gachette.ino
 * Envoie des trames IR au format NEC en fonction de la valeur des interrupteurs
 * Déclenché par une gâchette sur GPIO6
 */

#include <Arduino.h>
#include "PinDefinitionsAndMore.h"
#include <irsndSelectMain15Protocols.h>
#include <irsnd.hpp>
#include <WiFi.h>
#include <WiFiClient.h>
// Configuration Wi-Fi
const char ssid[] = "Biathlon";
const char password[] = "biathlon";
IPAddress IpServeur(192, 168, 6, 35);
int PortServeur = 2222;
// Variables globales
int y = 0;
bool gachette = false;
WiFiClient client;
IRMP_DATA irsnd_data;
// Prototypes
void debounceInterrupt();
void setup() {
    Serial.begin(115200);
    #if defined(__AVR_ATmega32U4__) || defined(SERIAL_PORT_USBVIRTUAL) ||
defined(SERIAL_USB) || defined(USBCON) || defined(SERIALUSB_PID) ||
defined(ARDUINO_ARCH_RP2040) || defined(ARDUINO_attiny3217)
    delay(4000); // Laisse le temps au moniteur série de se connecter
#endif
    // Init IR
    irsnd_init();
    irmp_irsnd_LEDFeedback(true); // Active le feedback LED
    Serial.println(F("Send IR signals at pin " STR(IRSND_OUTPUT_PIN)));

    // Configuration du protocole NEC
```

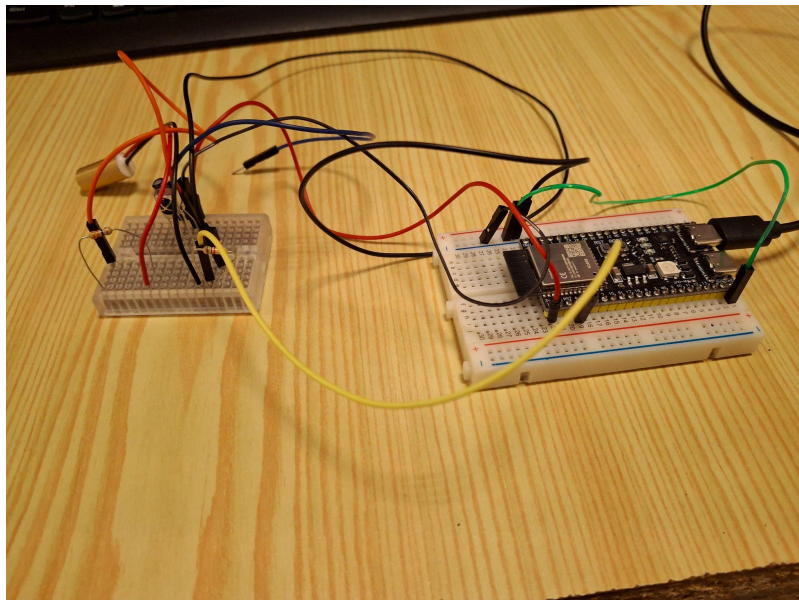
```
irsnd_data.protocol = IRMP_NEC_PROTOCOL;
irsnd_data.address = 0x25DA; // Adresse avec complément
irsnd_data.command = 0x01;   // Commande par défaut
irsnd_data.flags = 2;        // Répéter la trame 2 fois
// Envoi de test
if (!irsnd_send_data(&irsnd_data, true)) {
    Serial.println(F("Protocole non trouvé")); }
irsnd_data_print(&Serial, &irsnd_data);
// Configuration des entrées
pinMode(1, INPUT_PULLUP);
pinMode(2, INPUT_PULLUP);
pinMode(3, INPUT_PULLUP);
pinMode(6, INPUT_PULLUP); // Gâchette
attachInterrupt(6, debounceInterrupt, FALLING);
// Connexion Wi-Fi
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password); }
void loop() {
    delay(1000);
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("Pas de WIFI, prochain essai...");
        WiFi.begin(ssid, password);
        return; }
    Serial.println("Connecté au WIFI");
    // Tentative de connexion serveur
    if (!client.connected()) {
        client.stop();
        Serial.println("Connexion serveur perdue, tentative de
reconnexion...");
        client.connect(IpServeur, PortServeur);
        if (!client.connected()) {
            Serial.println("Mode entraînement : pas de serveur"); } }
    // Si gâchette activée, lire valeur et envoyer trame IR
    if (gachette) {
        gachette = false;
        y = 0;
        if (!digitalRead(1)) y += 1;
        if (!digitalRead(2)) y += 2;
        if (!digitalRead(3)) y += 4;

        Serial.print("Numéro de la cible lu : ");
        Serial.println(y);
    }
```

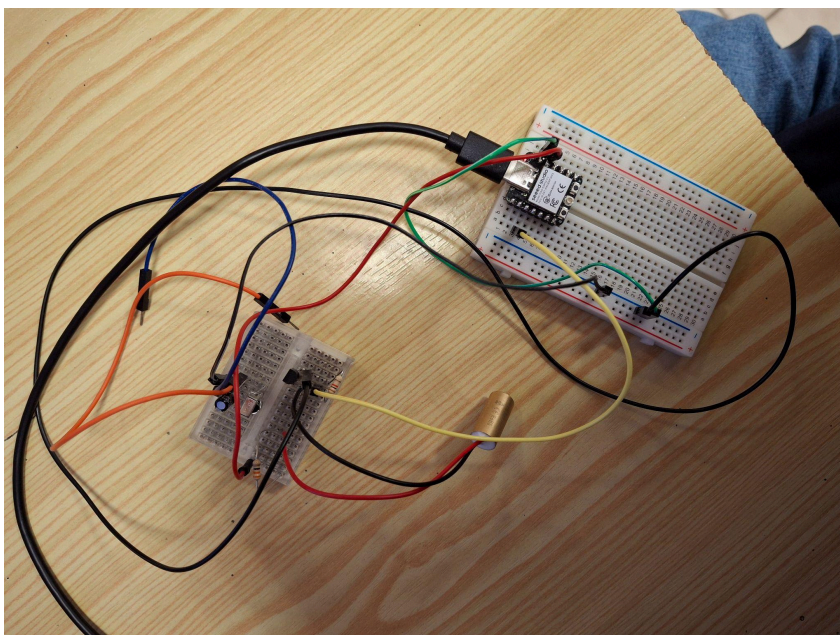
```
    irsnd_data.command = y;
    irsnd_send_data(&irsnd_data, true);
    irsnd_data_print(&Serial, &irsnd_data); } }

// Interruption de la gâchette (GPIO6)
void debounceInterrupt() {
    static unsigned long lastInterrupt = 0;
    unsigned long interruptTime = millis();
    if (interruptTime - lastInterrupt > 200) {
        Serial.println("Gâchette utilisée");
        gachette = true;
        String trameDecouverte = "@0" + String(y) + "1%";
        if (client.connected()) {
            client.write(trameDecouverte.c_str()); // Trame de découverte
        } else {
            Serial.println("Client non connecté, trame non envoyée"); } }
    lastInterrupt = interruptTime;
}
```

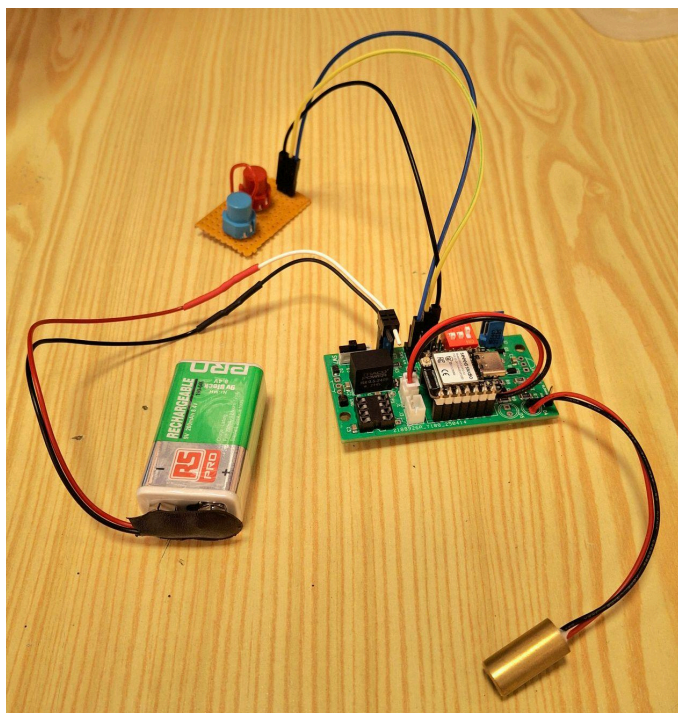
Montage de l'ESP DevKit sur plaque (avec récepteur inclus dans le montage mais non utilisé)



Montage du XIAOS3 sur plaque (avec récepteur inclus dans le montage mais non utilisé)



Montage final de la carabine



Montage final de la cible (5 tirs réussis)

