# Laser Gate Timer 2020-2021

4 FEVRIER

**Association Alliance Glisse de Franche Comté**

**Créé par : Dorian VOYDIE & Tanguy KESSELY**

# Table des matières

# Rapport Technique

## Introduction

L'objectif premier de ce projet est de concevoir un prototype d'un produit afin de satisfaire les besoins de l'association Alliance Glisse Franche Comté. Un prototype a déjà été mis en place mais il ne remplit pas toutes les attentes du **CdCF**. A partir du travail qui a déjà été effectué, nous allons dans un premier temps cerner l'environnement des besoins, définir le périmètre de notre projet, et essayer de voir ce qui se fait déjà sur le marché.

### Description du projet

**OBJECTIF :**

Développer des cellules de chronométrage sans fil pour le speed slalom en roller et afficher le résultat sur PC/Smartphone. Le système doit être à moindre coût.

| | |
|---|---|
| Réalisateurs : | Dorian VOYDIE : 06.27.86.02.06<br>Tanguy KESSELY : 07.81.89.84.90 |
| Descriptif du projet : | - *A **QUI, QUOI** sert le produit ?*<br>  - *Le produit sert à chronométrer les participants sur un parcours de roller slalom*<br>  - *Le produit sert à mesurer le temps de parcours d'un participant sur une piste*<br>- *Sur **QUI, QUOI** agit le produit ?*<br>  - *Le produit agit sur l'entraîneur et lui permet de visualiser les performances de ses athlètes, mais aussi les arbitres et les organisateurs par la prise en main de l'outil et son installation*<br>- ***POURQUOI** dans quel but le produit existe ?*<br>  - Avoir accès facilement à un outil de chronométrage fiable et peu cher. |

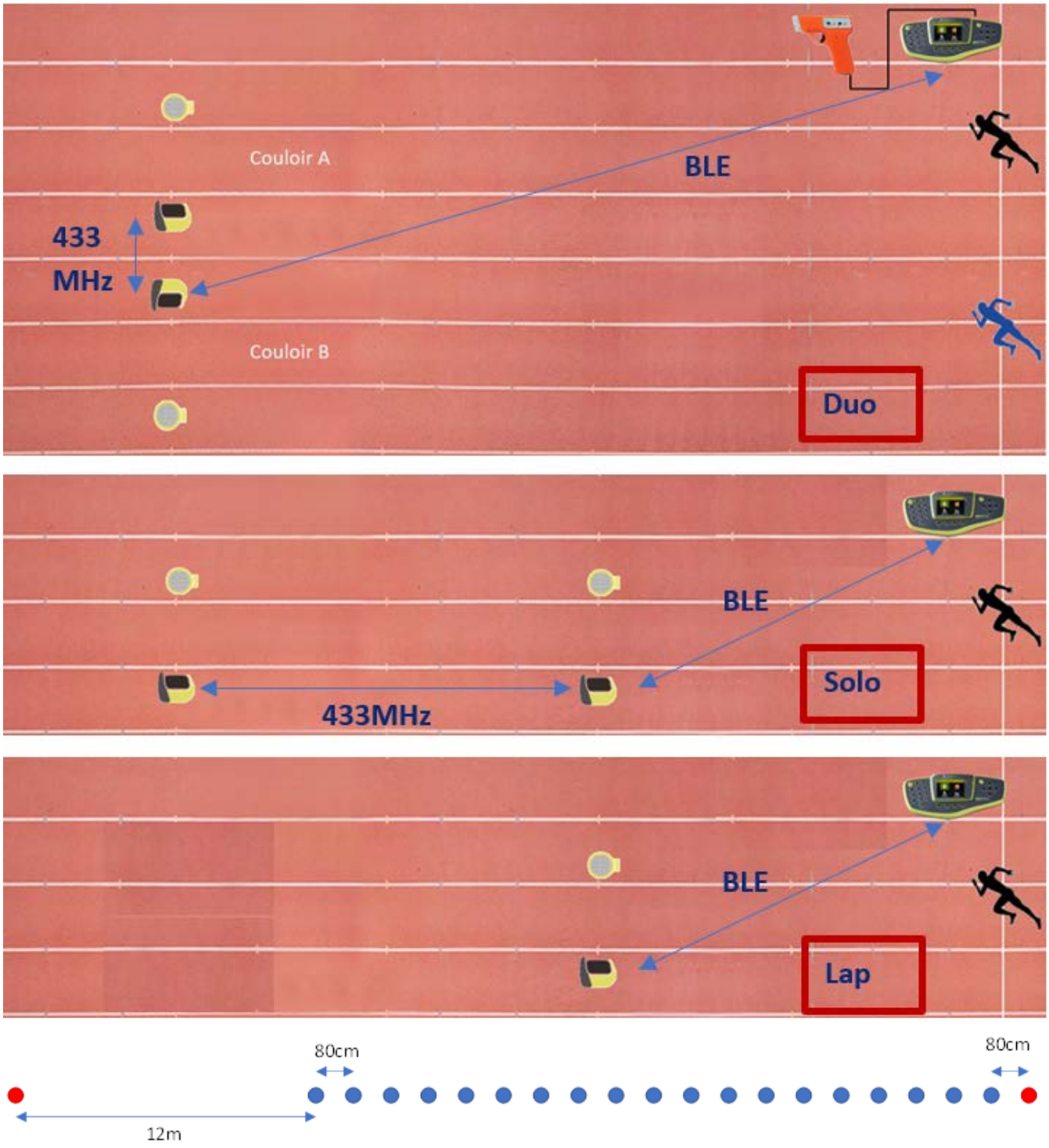



*Figure 1 : Configurations d'utilisation et spécifications de distances*

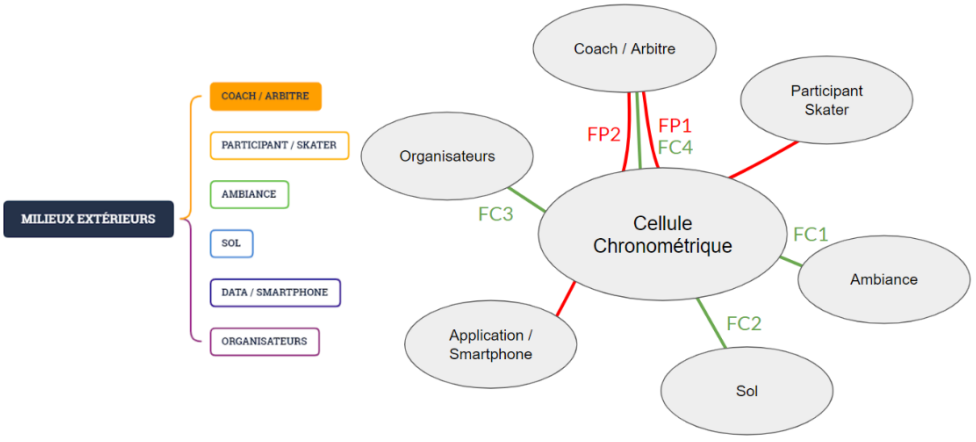
L'objectif ici est d'atteindre une précision de l'ordre du centième de seconde (éventuellement millième), le tout en minimisant le prix du produit final (des choses existent sur le marché mais à plus de 500€).

Plusieurs modes de fonctionnement sont demandés (illustration 1), ainsi qu'un système de suivi par téléphone (app). Il faut également que le système soit portatif, autonome en énergie, résistant et facile d'installation.

Nous développerons chacun des points dans le **CdCF**.

# Cahier des Charges Fonctionnel :

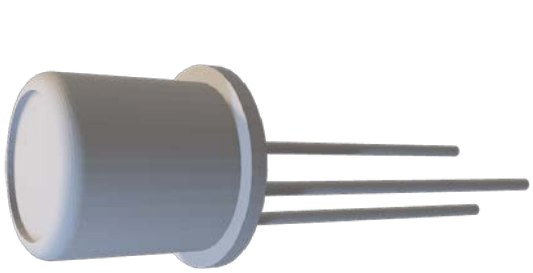
| CAHIER DES CHARGES FONCTIONNEL | | Réalisé par Tanguy KESSELY & Dorian VOYDIE | | | | |
|---|---|---|---|---|---|---|
| Date | *29/9/2020* | | | | | |
| **Fnc** | **Fonctions et contraintes** | **Level** | **Critères** | **Les conditions de performance idéales** | **Flex** | **Commentaires divers** |
| | | ***D**ésiré* ***E**xigé* | | Niveaux optimaux des caractéristiques pour chaque fonction | | |
| | | | | | | |
| *FP1* | *La cellule doit permettre au COACH de CHRONOMÉTRER le SKATER* | E | *Précision*<br>*Précision*<br>*Autonomie* | *1/100 s*<br>*1/1000 s*<br>*3h* | *0*<br>*2*<br>*0* | Le millième inutile |
| *FP2* | La cellule doit ENVOYER et RECEVOIR des informations du COACH via une APPLICATION | E | *Protocole*<br>*Support de réception*<br>*Support d'envoi* | *BLE*<br>*PC*<br>*Pistolet / Déclencheur* | 1<br>1<br>0 | Ou wifi<br>Sinon API mobile |
| *FC1* | La cellule doit RÉSISTER à l'AMBIANCE | D | *Protection aux chocs*<br>*Température*<br>*Humidité*<br>*UV*<br>*Matériaux* | *Carter complet*<br>*< 40°C*<br>*< 90%*<br>*Pas d'exposition*<br>*Inoxydables, résistants, tenace* | 0<br>1<br>1<br>2<br>1 | <br>Gymnase OK<br>Gymnase OK<br>En intérieur<br>En fonction du prix |
| *FC2* | La cellule doit S'ADAPTER à la forme du SOL | D | *Poids (répartition)*<br>*Stabilité au sol*<br>*Hauteur de mesure*<br>*Angle de mesure* | *250 g < X < 750 g*<br>*Adhésif, grippe, plat*<br>*Environ 20 cm du sol*<br>*Parallèle au sol* | 1<br>0<br>0<br>0 | Transportable<br>Entièrement Stable |
| *FC3* | La cellule doit permettre aux ORGANISATEURS de l'INSTALLER facilement | D | *Ergonomie*<br>*Vitesse de mise en place* | *Poignée*<br>*< 1mn* | 1<br>0 | Jointe au carter |
| *FC4* | *La cellule ne doit pas COÛTER trop cher* | E | *Prix* | *< 500€* | 0 | |

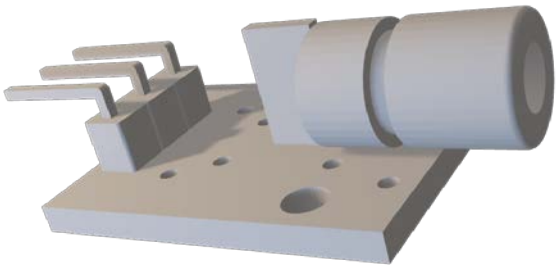


# Expérimentations

## Laser (KY-008) et photodiode (SD5600-10)

Pour pouvoir créer une porte laser, nous avons utilisé les deux modèles susnommés ainsi qu'un miroir.





Leur documentions sont trouvables en ligne. Nous avons réalisé des tests d'acquisition de la photodiode et avons relevé son Rising Time d'environ 6µs (figure 2). Cela nous permettait de nous assurer un retard négligeable sur le déclenchement ou l'arrêt du chronomètre (souhaitant une précision à 1/100 secondes, ce retard de 6µs est bel et bien négligeable)

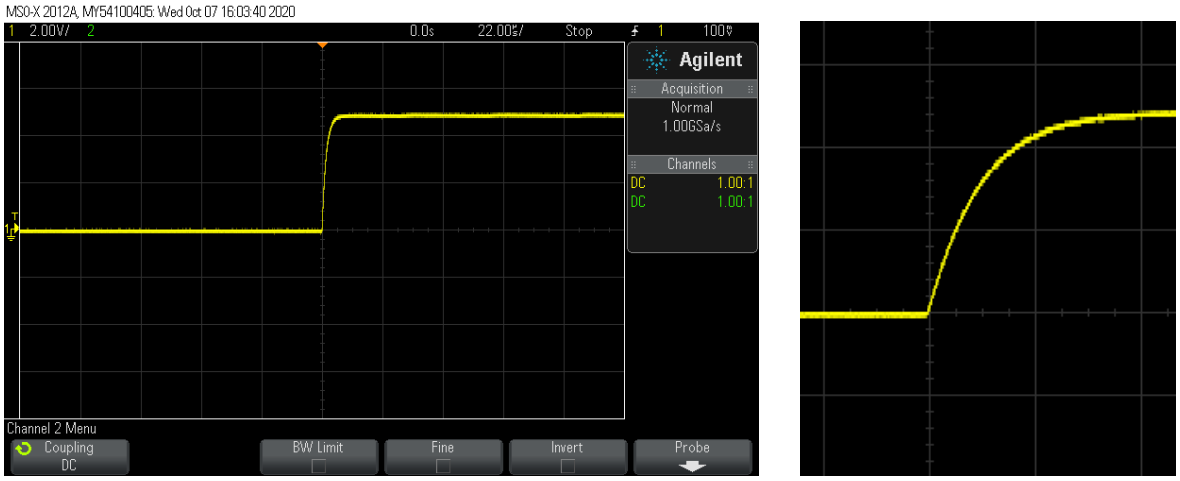


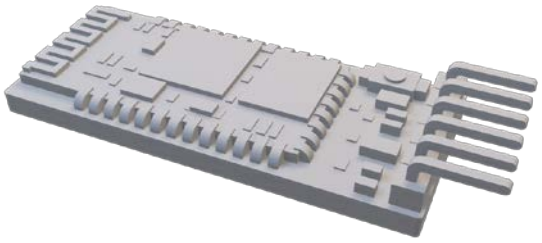

*Figure 2 : Mesure du Rising Time de la Photodiode SD5600*

Pour des raisons d'économie d'énergie, nous avons également décidé de piloter notre laser grâce à un transistor 2N2222.

# Communication entre les deux cellules

## Modules de communication Bluetooth HC-05

Pour assurer la communication entre les deux cellules, nous avons commencé par utiliser des modules Bluetooth maitre/esclave HC05 :



L'inconvénient de ces modules est que leur temps de transmission d'une information est très élevé, mais surtout n'est pas fixe. Nous avons réalisé des mesures à l'aide d'un compteur incrémental. Grâce à ce dernier, on mesure le temps réel entre deux appuis des boutons des cellules. On le compare ensuite au temps que nous retourne la cellule de départ, après que la cellule d'arrivée lui ait transmis l'ordre par Bluetooth d'arrêter le Timer.

Nous en avons tiré un tableau et une courbe. Le **délai moyen** entre le compteur et la transmission Bluetooth est **174.6ms**. Le plus gros écart à la moyenne en **dispersion** est de **15ms**

| Compteur (µs) | Bluetooth (µs) | Délai (ms) | Ecart à la moyenne (ms) |
|---|---|---|---|
| 1065795 | 1236148 | 170,353 | 4,246133333 |
| 1110636 | 1289500 | 178,864 | 4,264866667 |
| 2988052 | 3155366 | 167,314 | 7,285133333 |
| 3142866 | 3331101 | 188,235 | 13,63586667 |
| 3206598 | 3384769 | 178,171 | 3,571866667 |
| 3867165 | 4031803 | 164,638 | 9,961133333 |
| 5254229 | 5435432 | 181,203 | 6,603866667 |
| 6901390 | 7073501 | 172,111 | 2,488133333 |
| 8265705 | 8436586 | 170,881 | 3,718133333 |
| 9890622 | 10074956 | 184,334 | 9,734866667 |
| 33192833 | 33352423 | 159,59 | -15,00913333 |
| 80756059 | 80932863 | 176,804 | 2,204866667 |
| 121847537 | 122018335 | 170,798 | -3,801133333 |
| 181195067 | 181384424 | 189,357 | 14,75786667 |
| 238674820 | 238841154 | 166,334 | 8,265133333 |

*Tableau 1 : Comparaison HC-05 & Compteur incrémental*

C'est donc, pour notre utilisation, un module non juste et non fiable.

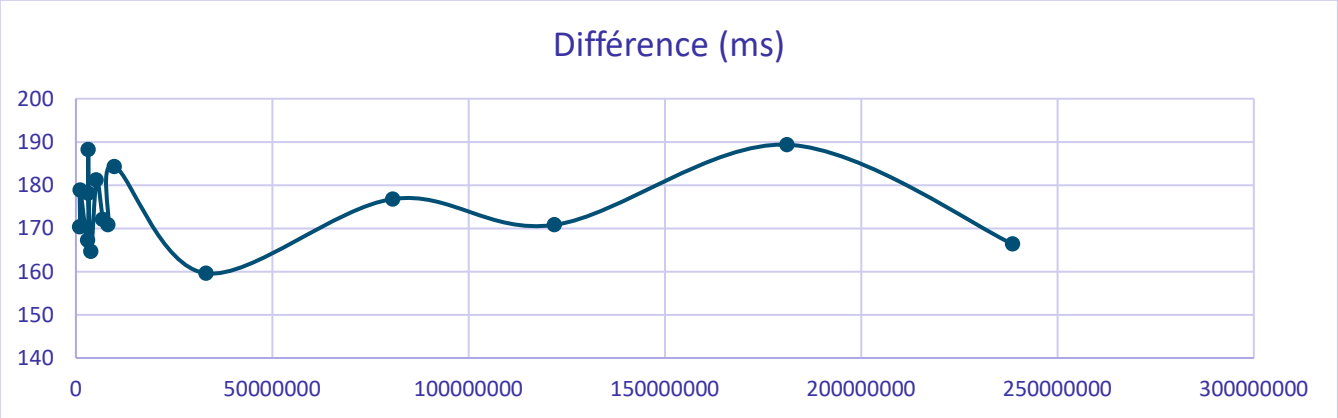



*Figure 3 : Dispersion du délai de réception par les modules HC-05*

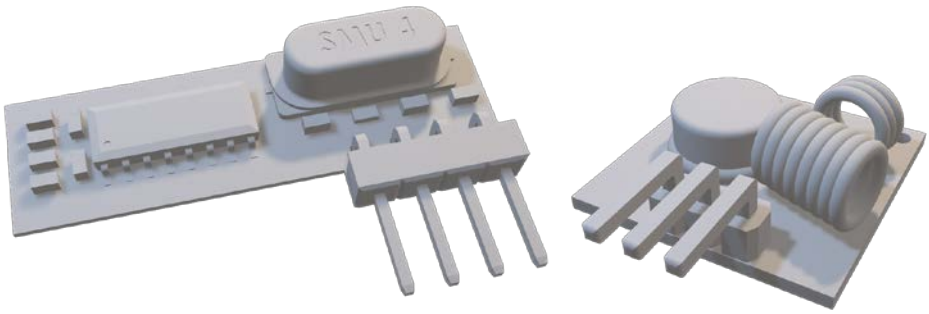
Notre cahier des charges spécifiant une **précision** du **centième de seconde**, le simple fait d'utiliser ces modules ne nous permettait pas de respecter ce critère.

### Modules de communication RF (QAM-RX10-433 & FS1000A)

Nous nous sommes donc rabattus sur une technologie plus simple : les modules RF 433MHz.

Récepteur QAM-RX10-433 de 3.3 à 6V **(Actuellement alimenté en 5V)**

Emetteur FS1000A de 3.3 à 12V **(Actuellement alimenté en 5V)**



En effet, ces modules utilisant les ondes radio, le temps de réception de l'information est quasiment instantané !

Suite à l'implémentation de ces modules, nous avons réalisé le même test que pour le module HC-05 à l'aide du compteur, et voici les résultats :

| Compteur (s) | Modules RF (s) | Différence (ms) |
|---|---|---|
| 11,233 | 11,25 | 17 |
| 4,295 | 4,32 | 25 |
| 11,646 | 11,66 | 14 |
| 37,773 | 37,79 | 17 |
| 69,831 | 69,85 | 19 |
| 3,239 | 3,26 | 21 |
| 18,423 | 18,44 | 17 |
| 22,894 | 22,91 | 16 |
| 4,707 | 4,72 | 13 |
| 2,593 | 2,61 | 17 |

*Tableau 2 : Comparaison RF & Compteur incrémental*

On obtient cette fois-ci un **délai moyen de 17.6ms** et une **dispersion moyenne de 2.44ms**.

Voici l'aspect de la courbe obtenue :

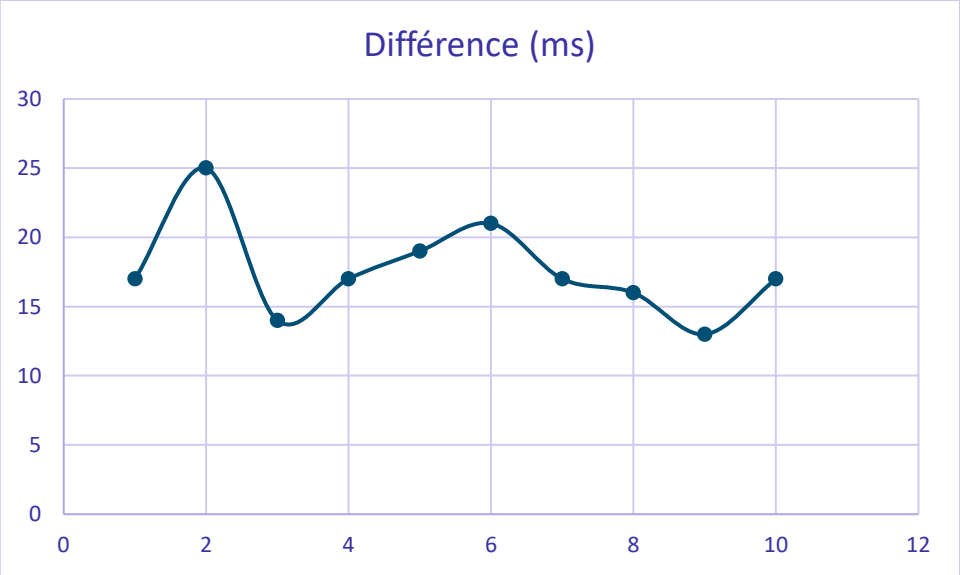
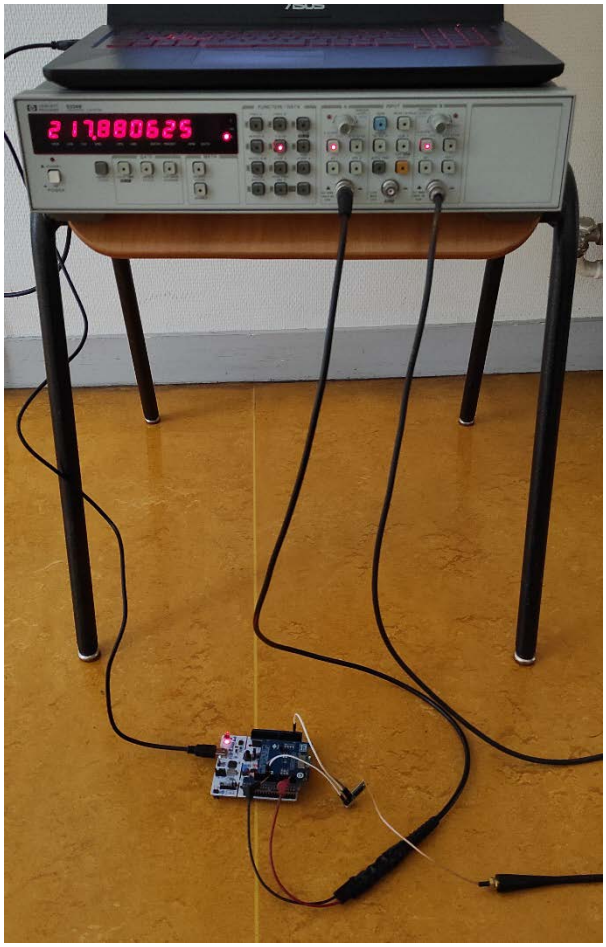
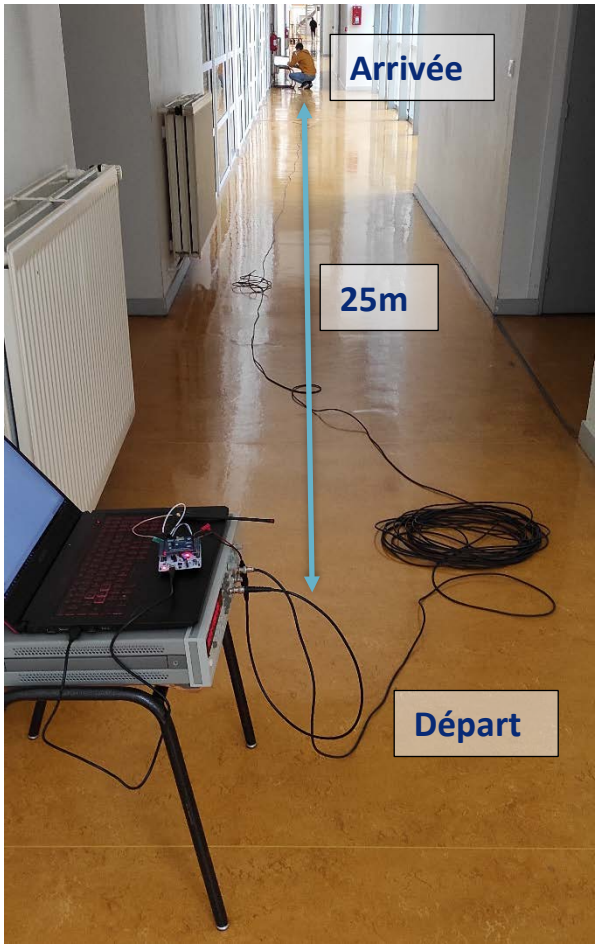


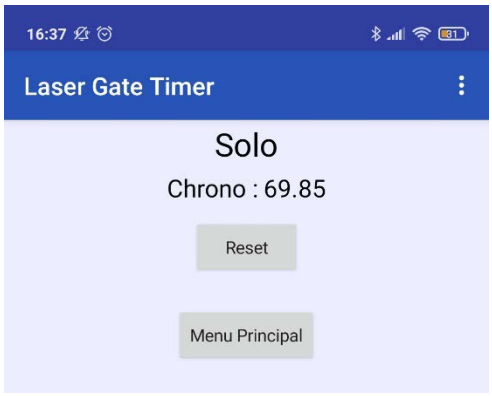

*Figure 4 : Dispersion du délai de réception par les modules RF*

Ces données montrent la fiabilité et la justesse des modules RF. L'information étant quasi-instantanément perçue, le **délai moyen** de **17.6ms** serait alors dû au process du microcontrôleur avant le calcul du chrono. Pour obtenir une précision au centième près, on se permettra donc **d'enlever 20ms à la valeur du chronomètre** afin de se rapprocher le plus possible de la valeur réelle (obtenant ainsi une précision à **±5ms**).

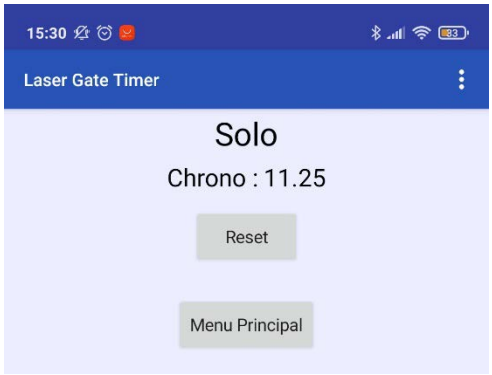
## Photos de l'expérience







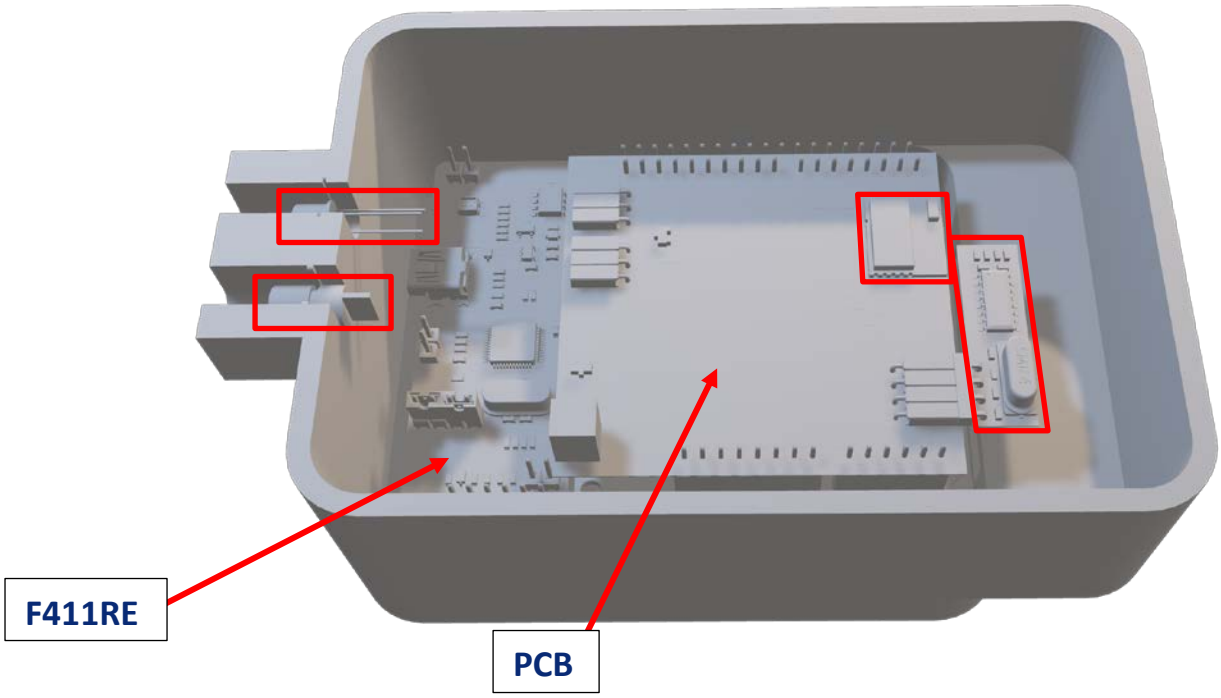

## Shield BLE IDB05A1 / Module SPBTLE

Ce module présent sur la cellule de départ permet à l'utilisateur de communiquer directement avec le matériel via l'application fournie.

Nous utilisons notamment les canaux UART du module pour recevoir et transmettre des informations à la cellule de départ.

## Modèle final

Nous avons ensuite repris la coque de nos prédécesseurs en y apportant quelques modifications afin de pouvoir accueillir tous nos composants (voici un extrait de la cellule de départ).




Le PCB a pour objectif d'embarquer le module BLE, le module RF, le laser et la photodiode afin d'éviter un câblage trop important. Vous retrouverez le détail du PCB dans la Technologie des circuits.

# Conception électronique

## Choix des composants (voir les documentations dans le dossier)

Nous avons choisi le microcontrôleur STM32-F411RE de telle manière à pouvoir créer un Shield qui contiendrait tout le hardware embarqué nécessaire au projet. Le modèle final est assez similaire à l'ObCP !

Nous avons choisi des composants à faible consommation d'énergie, notamment en utilisant la version Low Energy du Bluetooth, ou encore des modules 433MHz moins énergivores que d'autres systèmes de communication (comme les modules HC-05).

Le laser a été choisi pour sa couleur, sa tension d'alimentation et son intensité suffisante en intérieur permettant d'éclairer sur une très longue distance.

Nous avons choisi la photodiode en conséquence, celle-ci permet de capter la lumière visible (400 – 800 nm) au-dessus d'une certaine intensité. De cette façon, le seul moyen de l'activer et de croiser le faisceau laser rouge suffisamment intense (alimenté en 5V).

## Calcul de la consommation & Dimensionnement batterie

| | Départ | | | | Arrivée | | |
|---|---|---|---|---|---|---|---|
| | **Composants** | **Alimentation choisie** | **Courant estimé continu (mA)** | | **Composants** | **Alimentation choisie** | **Courant estimé continu (mA)** |
| F411 | | 5V | 23 | F411 | | 5V | 23 |
| Laser | 5V | 5V | 30 | Laser | 5V | 5V | 30 |
| Photodiode | 4,5 – 16 | 5V | 12 | Photodiode | 4,5 – 16 | 5V | 12 |
| Récepteur RF | 3,3 – 6V | 5V | 28 | Émetteur RF | 3-12V | 5V | 22 |
| ... | | | | ... | | | |
| PWM | | | 40 | PWM | | | 40 |
| Module BLE | 1.7-3.6 | 3.3V | 0,2 | | | | |
| **Départ =** | 133,2 | mA | | **Durée :** | 3 | h | |
| **Arrivée =** | 127 | mA | | | | | |
| Capacité > | 399,6 | mAh | | | | | |

*Tableau 3 : Tableau de consommation des deux cellules*

Pour chaque composant, nous avons relevé les courants consommés soit dans la documentation, soit mesurés directement sur le système. Nous les avons répartis dans un tableau pour chaque cellule.
En somme, les composants d'une même cellule consomment environ **150 mA de courant**. Nous souhaitons faire fonctionner notre système au moins 3h sans avoir à le recharger. Il nous faudra donc une **batterie de près de 450 mAh**. En prenant une batterie de 1000mAh comme on a pu en avoir sur l'ObCP, notre système tiendrait deux sessions de 3h.

### Cartographie des alimentations

Voir fichier Excel « Cartographie des alimentations » en pièce jointe.

# Programmation Microcontrôleur et Interface

### Architecture simplifiée du programme

Pour une meilleure compréhension du fonctionnement des différents modes, reportez-vous à la Figure 1 : Configurations d'utilisation et spécifications de distances.

Les logigrammes qui suivent visent à simplifier le développement du programme qui va suivre. Pour plus de renseignements sur le cœur du programme, n'hésitez pas à contacter Dorian VOYDIE.

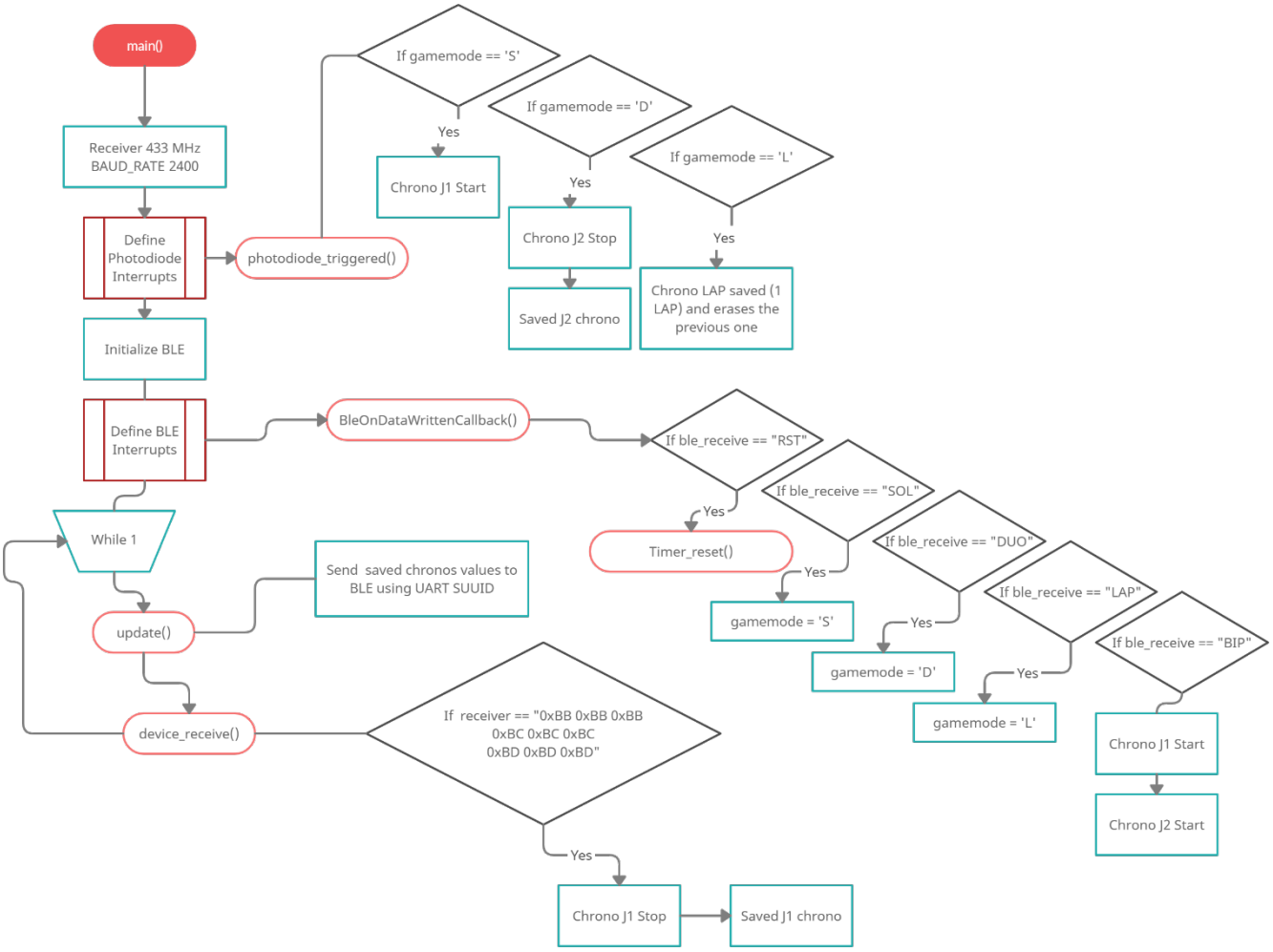



*Figure 5 : Architecture du programme de la cellule de départ*

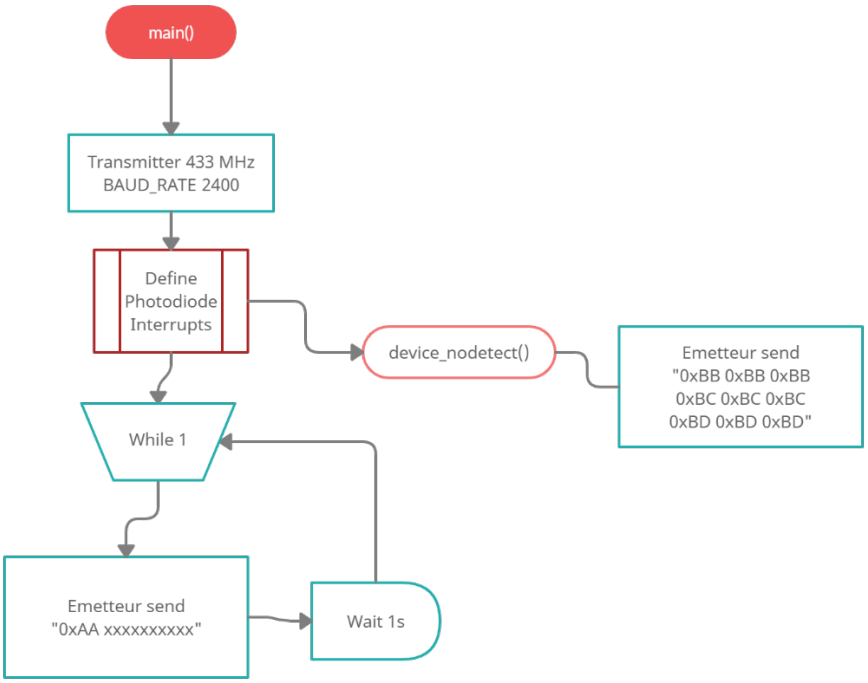



*Figure 6 : Architecture du programme de la cellule d'arrivée*

# Développement du programme

Ici sont explicités les blocs principaux des programmes. Il existe différents programmes selon l'équipement utilisé, notamment des programmes de débogage remplaçant le laser et la photodiode par l'USER_BUTTON (c'est plus pratique à utiliser…).

**Programme de l'Arrivée :**

```
int main(int, char**)
{
    device.baud(2400);
    my_button.fall(&device_nodetect);
    photodiode.rise(&device_detect);
    photodiode.fall(&device_nodetect);
    laser = 1;

    while (1) {
        //while RF device is receiving data
        //RF Transmit Code
        //Send 10101010 pattern when idle to keep receiver in sync and locked to transmitter
        //When receiver loses the sync lock (Around 30MS with no data change seen) it starts sending out noise
        device.putc(0xAA);
        for( int h = 0; h < 10; h++ ) {

            device.putc('x');
        }
        wait(1);
    }
}
```

*Figure 7 : Fonction main() de l'arrivée*

Sur la cellule d'arrivée, on retrouve le module émetteur RF 433MHz en liaison série device(D2,D8). On règle son BAUD_RATE à 2400 bauds et une boucle infinie se lance. Toute les secondes, le programme envoie le caractère '0xAA' suivi de 10 'x' afin « **d'accrocher** » le signal du récepteur de la cellule de départ. Cet accrochage survit jusqu'à 60m avec des antennes et sans trop grosses interférences autour. Si on souhaite augmenter cette limite, il faut alors augmenter la tension de l'émetteur (pour l'instant alimenté en 5V, on peut aller jusqu'à 12V). Cela aurait cependant pour effet d'augmenter de manière conséquente la consommation de la cellule (voir documentation de l'émetteur).

**Il y a deux interruptions différentes :**

**device_detect** :

```
void device_detect()
{
    compteur = 1;
    wait(0.001);
    compteur = 0;
    for( int h = 0; h < 10; h++ ) {
        device.putc(0xCC);
    }
}
```

*Figure 8 : Interruption device_detect()*

Cette interruption survient à l'instant où le laser traverse la photodiode : l'émetteur envoie alors 10 fois le caractère '0xCC' au récepteur de la cellule de départ pour signaler que la porte est active.

**device_nodetect** :

```
void device_nodetect()
{
    compteur = 1;
    wait(0.001);
    compteur = 0;
    for( int h = 0; h < 2; h++ ) {
        device.putc(0xBB);
        device.putc(0xBB);
        device.putc(0xBB);
        device.putc(0xBC);
        device.putc(0xBC);
        device.putc(0xBC);
        device.putc(0xBD);
        device.putc(0xBD);
        device.putc(0xBD);
    }

}
```

*Figure 9 : Interruption device_nodetect()*

Cette interruption survient à l'instant où le laser ne traverse plus la photodiode : l'émetteur envoie alors 2 fois les caractères '0xBB' '0xBC' '0xBD' au récepteur de la cellule de départ pour signaler que la porte a été traversée. Ces différents caractères serviront, dans le bon ordre, de clef de vérification.

**Programme du départ :**

```
int main(void)
{
    device.baud(2400);
    myButton.fall(&Timer_triggered);
    photodiode.fall(&Timer_triggered);


    /****** START Initialiser BLE **********/
    BLE &ble = BLE::Instance();
    ble.init(bleInitComplete);
    /******* FIN initialise BLE ***********/
}
```

*Figure 10 : Fonction main(void) du départ*

Comme pour l'arrivée, on règle le BAUD_RATE sur la même valeur, et on initialise le BLE.

```
// Attend les événements sur BLE
while (true) {

    ble.waitForEvent();
    update();
    //RF Receive Code
    device_receive();
    wait(0.01);
}
```

*Figure 11 : Initialisation du BLE*

Une fois que le **BLE est initialisé**, on entame la boucle infinie. Cette boucle s'itère toutes les 0.01 secondes et active deux fonctions : **update()** (permet la mise à jour des chronos sur l'application) et **device_receive()** (permet la réception des caractères sur la bande 433MHz).

```
/////////COMMUNICATION///////////////
void device_receive()
{
    temp=device.getc();
    //Ignore Sync pattern and do not pass on to PC
    if (temp!=0xAA) {
        //Mode solo, la cellule arrivée envoie u à la cellule départ
        if(temp == 0xBB and temp2!=0xBB) {
            temp2=0xBB;
        }
        if(temp == 0xBC and temp2==0xBB) {
            temp3 = 0xBC;
        }
        if(temp == 0xBD and temp3==0xBC) {
            end1 = timer_player_1.read_ms();
            timer_player_1.stop();
            chrono1 = end1-begin1-0.02;
            photodiode2 = 0;
            pc.printf("C'est bon");
        }
        if(temp == 0xCC) {
            photodiode2 = 1;
        }
        pc.putc(temp);
    }


}
//////////////////////////////////////
```

*Figure 12 : Fonction device_receive()*

Cette fonction a pour but de lire les caractères qu'elle reçoit, et de les comparer avec les clefs diffusées par l'émetteur (soit '0xCC', soit '0xBB' '0xBC' '0xBD').

Effectivement, si on appelle **« photodiode2 »** la photodiode de la cellule d'arrivée, lorsque celle-ci vaut 1, la fonction **device_detect**() s'activera alors et enverra '0xCC'.

De même si la photodiode2 vaut 0, alors la fonction **device_nodetect**() est appelée, ainsi on s'attend à récupérer '0xBB' puis '0xBC' et enfin '0xBD'.

On a donc une triple sécurité, cette combinaison doit arriver dans le bon ordre. Il y a donc de très faibles chances que le chronomètre se STOP à cause d'interférences.

```
void update(void)
{

    if (gamemode != 'A') {
        char train1[5];
        char train2[5];
        float chrono_value_J1 = float(chrono1)* 0.001F;
        float chrono_value_J2 = float(chrono2)* 0.001F;

        //Transformation des valeurs numeriques en chaine de caracteres
        sprintf(train1,"%5.2f",chrono_value_J1);
        sprintf(train2,"%5.2f",chrono_value_J2);
        //Integre les trois chaines de caractere contenant les ts dans la chaine uartBuff
        sprintf(uartBuff, "%s %s", train1, train2) ;
        //Envoie la chaine uartBuff sur le sevice TX UART BLE
        uartServicePtr->write(uartBuff, UARTService::BLE_UART_SERVICE_MAX_DATA_LEN);
        // Réinitialiser la chaîne uartBuff en entrant 0 dans les premiers caractères UART_BUFFER
        memset(uartBuff, 0, UART_BUFFER);
    } else {
        char laser1[1];
        char laser2[1];

        //Transformation des valeurs numeriques en chaine de caracteres
        float meas = photodiode1.read();
        sprintf(laser1,"%1.0f",meas);
        sprintf(laser2,"%1.0f",photodiode2);
        //Integre les trois chaines de caractere contenant les ts dans la chaine uartBuff
        sprintf(uartBuff, "%s %s", laser1, laser2) ;
        //Envoie la chaine uartBuff sur le sevice TX UART BLE
        uartServicePtr->write(uartBuff, UARTService::BLE_UART_SERVICE_MAX_DATA_LEN);
        // Réinitialiser la chaîne uartBuff en entrant 0 dans les premiers caractères UART_BUFFER
        memset(uartBuff, 0, UART_BUFFER);
    }


}
```

*Figure 13 : Fonction update() sur le BLE en lecture*

Cette fonction a **deux modes** : le mode d'alignement (gamemode = A) et les modes de jeux (Solo, Duo ou Lap : gamemode = S, D ou L).

**En mode alignement**, la fonction **enverra sur l'UUID UART la valeur des photodiodes** des deux cellules (0 ou 1), permettant ainsi de savoir si les lasers sont bien alignés avec leurs photodiodes

**En mode jeu**, la fonction enverra sur le même UUID **la valeur des deux chronos** mis à jour lors des interruptions (voir ci-après)

```
void photodiode_triggered()
{
    if (gamemode == 'S') {
        compteur = 1;
        wait(0.001);
        compteur = 0;
        Timer_Reset();
        timer_player_1.start();
        begin1 = timer_player_1.read_ms();
        myled=1;
        wait(0.5);
        myled=0;

    }
    if (gamemode == 'D') {
        wait(0.001);
        compteur = 0;
        end2 = timer_player_2.read_ms();
        timer_player_2.stop();
        chrono2 = end2-begin2;
    }
    if (gamemode == 'L') {
        wait(0.001);
        compteur = 0;
        timer_player_1.start();
        chrono1 = timer_player_1.read_ms();
        myled=1;
        wait(0.5);
        myled=0;
    }
}
```

*Figure 14 : Interruption sur la photodiode*

Selon le mode de jeu dans lequel on se trouve, l'interruption qui a lieu lors d'un passage par la photodiode (« fall ») n'aura pas le même effet.

Etant la cellule de départ, en mode **SOLO** elle aura pour effet de reset le dernier temps acquis, puis de **redémarrer le chronomètre du « Joueur 1 »** pour être prêt à mesurer un nouveau temps (chrono1 pour « Joueur 1 »).

En mode **LAP**, à chaque passage devant la photodiode **la variable chrono1 prend la valeur du chronomètre continuant de tourner**. Celle-ci s'update par ailleurs sur le BLE à chaque fois après 0.01s grâce à la fonction update() dans la boucle infinie (voir plus haut).

En mode **DUO**, le départ des deux chronomètres étant donné à partir de l'application, couper la photodiode aura pour effet de **couper le chronomètre du « Joueur2 ».**

```
void BleOnDataWrittenCallback(const GattWriteCallbackParams *params)
{
    char reception[UART_BUFFER];
    char commande[3];
    if (params->handle == uartServicePtr->getTXCharacteristicHandle()) {
        // Copie de la chaine reçue dans reception
        sprintf(reception,"%s", params->data);
        // Copie dans la chaine commande des deux premier caracteres de la chaine reception
        sprintf(commande,"%c%c%c", reception[0], reception[1], reception[2]);

        if( strcmp(commande, "RST" )==0 ) {
            Timer_Reset();
            update();
        }
        if( strcmp(commande, "SOL" )==0 ) {
            gamemode = 'S';
            laser=1;
            update();
        }
        if( strcmp(commande, "DUO" )==0 ) {
            gamemode = 'D';
            laser=1;
            update();
        }
        if( strcmp(commande, "LAP" )==0 ) {
            gamemode = 'L';
            laser=1;
            update();
        }
        if( strcmp(commande, "MEN" )==0 ) {
            gamemode = 'M';
            laser=0;
            update();
        }
        if( strcmp(commande, "LAS" )==0 ) {
            gamemode = 'A';
            laser=1;
            update();
        }
        if( strcmp(commande, "BIP" )==0 ) {
            Timer_Reset();
            timer_player_1.start();
            timer_player_2.start();
            begin1 = timer_player_1.read_ms();
            begin1 = timer_player_2.read_ms();
            myled=1;
            wait(0.5);
            myled=0;
        }
    }
```

*Figure 15 : Interruption Callback sur le BLE en écriture*

La communication BLE marche dans les deux sens, aussi pour être encore plus en interaction avec les cellules, naviguer à travers l'application enverra des instructions aux cellules. Par exemple, le laser est étant quand on est dans le menu, et il est allumé quand on est dans n'importe quel mode de jeu. Le « BIP » est le signal reçu lorsqu'on appuie sur le bouton de départ du mode DUO (sur l'application). Les chronomètres des deux joueurs sont alors remis à 0 puis relancés.

## Bibliothèques

https://os.mbed.com/teams/ENSMM/code/Laser_Gate_Timer_Depart/
https://os.mbed.com/teams/ENSMM/code/Laser_Gate_Timer_Arrivee/

# Application développée sous MIT AppInventor

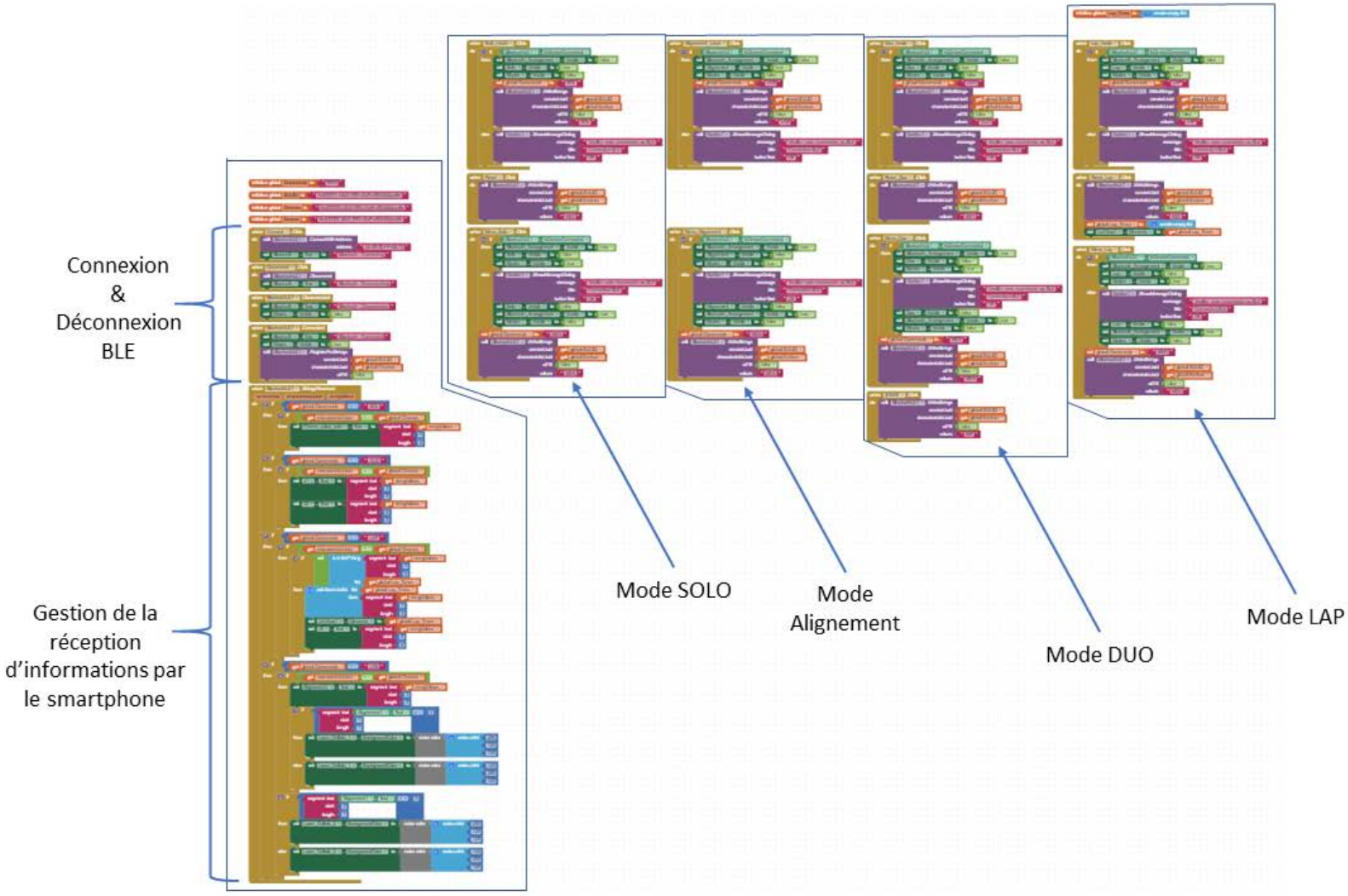



*Figure 16 : Architecture de l'application pour Smartphone développée sous MIT AppInventor*

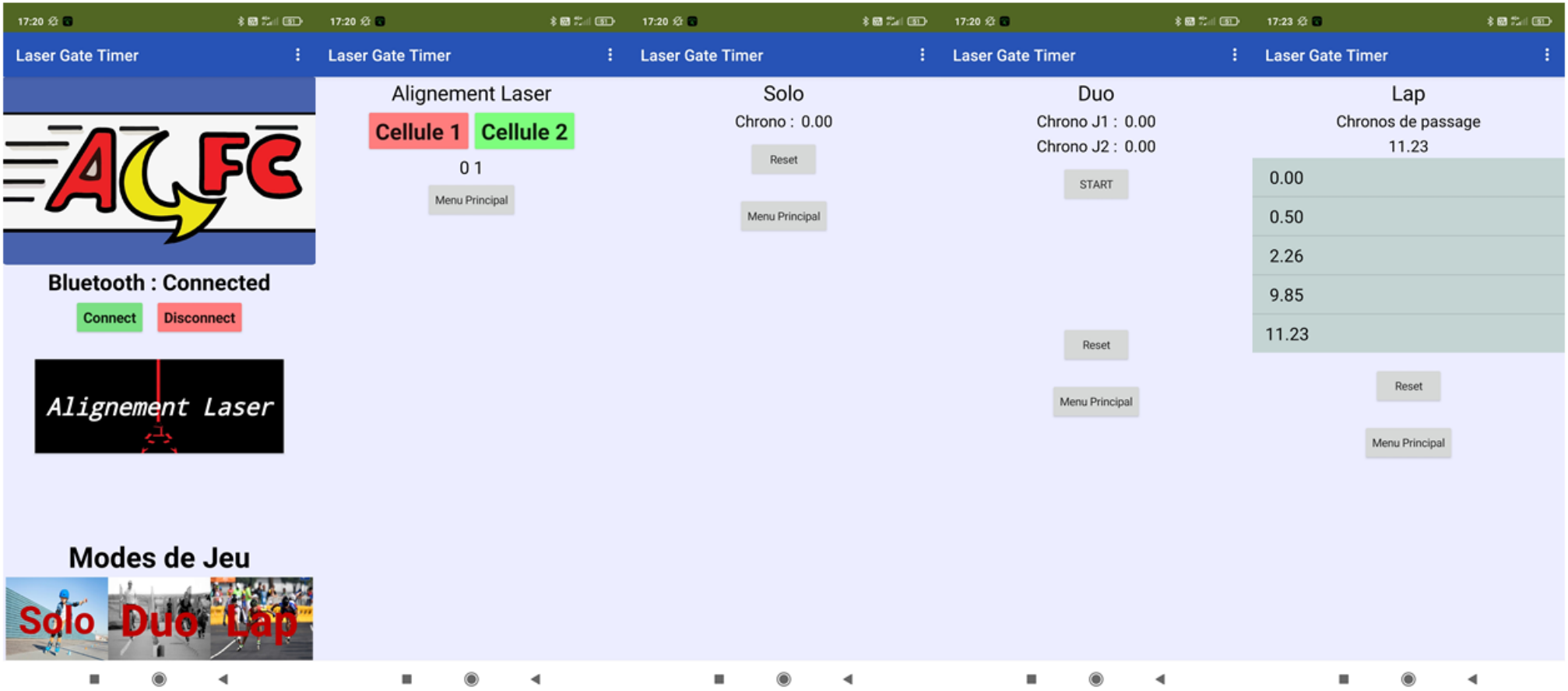
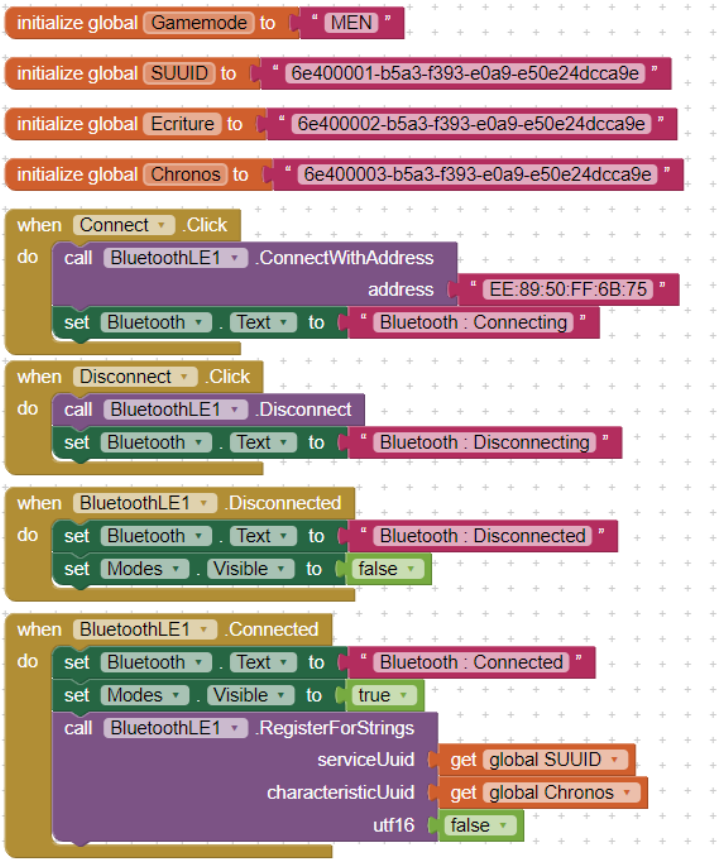



*Figure 17 : Différents menus de l'application*




**Initialisation du mode de jeu à « Menu », et définition des UUID Service & Caractéristiques pour l'écriture et la lecture**

**Boutons de connexion et déconnexion à l'adresse mac spécifique de la cellule de départ. Affichage du statut de la connexion.**
**Mise à jour à 0 de l'affichage des chronos.**

***Figure 18 : Application - Interface de connexion***

Rappel : Le programme développé sous Mbed envoie les chronomètres sous la forme :

« AA.AA_BB.BB »

Pour les chronos des joueurs A et B en mode Solo, Duo ou Lap

```
char train1[5];
char train2[5];
float chrono_value_J1 = float(chrono1)* 0.001F;
float chrono_value_J2 = float(chrono2)* 0.001F;

//Transformation des valeurs numeriques en chaine de caracteres
sprintf(train1,"%5.2f",chrono_value_J1);
sprintf(train2,"%5.2f",chrono_value_J2);
//Integre les trois chaines de caractere contenant les ts dans la chaine uartBuff
sprintf(uartBuff, "%s %s", train1, train2) ;
```

En mode Alignement cependant, on a la valeur de la photodiode (0 ou 1) sous la forme :

« X Y »

```
char laser1[1];
char laser2[1];

//Transformation des valeurs numeriques en chaine de caracteres
float meas = photodiode1.read();
sprintf(laser1,"%1.0f",meas);
sprintf(laser2,"%1.0f",photodiode2);
//Integre les trois chaines de caractere contenant les ts dans la chaine uartBuff
sprintf(uartBuff, "%s %s", laser1, laser2) ;
```

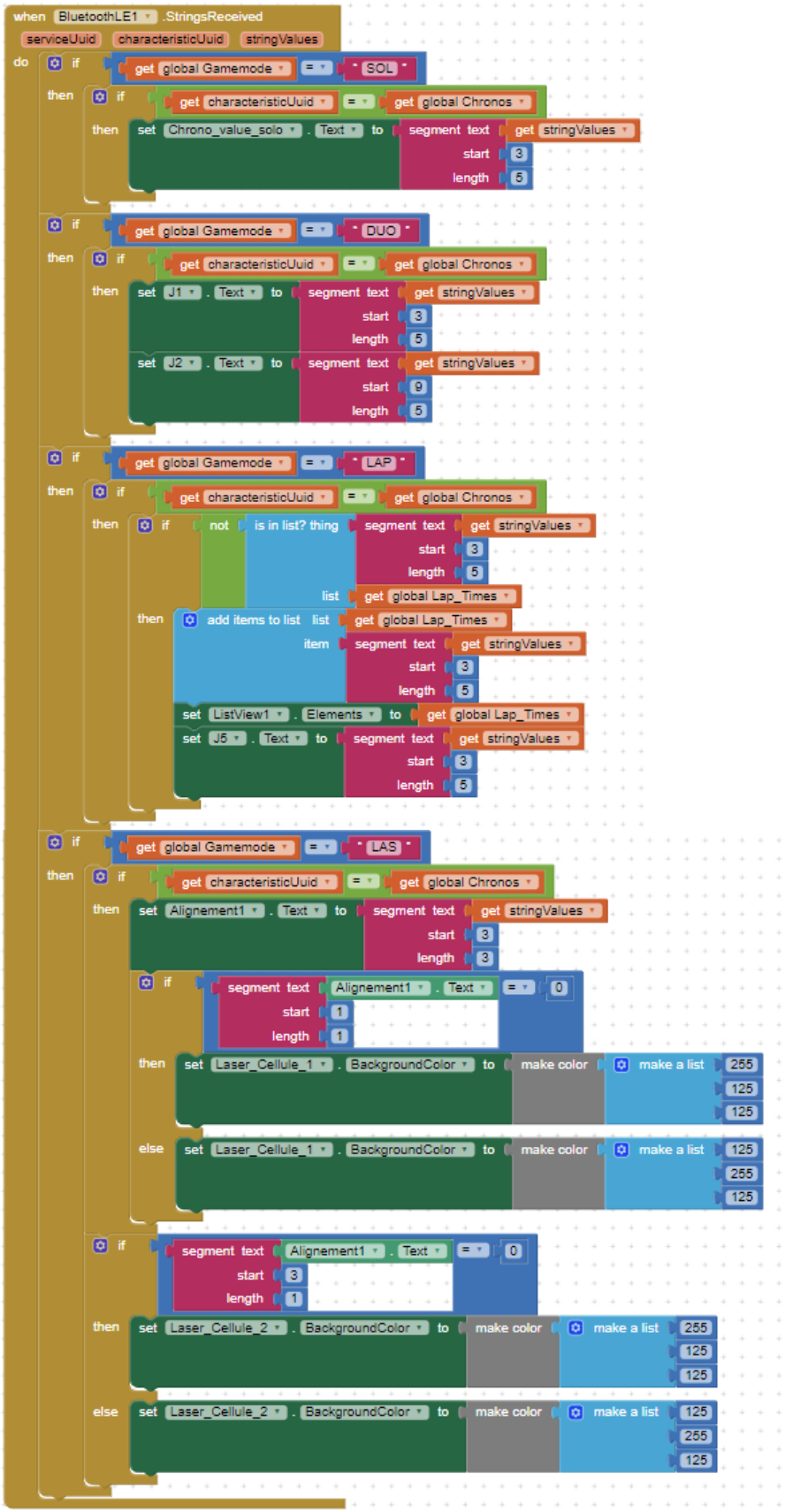



*Figure 19 : Application - BLE lecture*

**En mode solo, on récupère uniquement le premier chrono (caractères 3 à 7)**

**En mode duo, on récupère les deux chronos (caractères 3 à 7 puis caractères de 9 à 14)**

**En mode Lap, on ajoute tout nouveau chronomètre (le premier) à la liste déroulante. Le dernier chrono est tout de même affiché en haut de l'interface**

**En mode alignement selon si la valeur de la photodiode est 0 ou autre (1), la couleur du bouton correspondant s'allumera soit en rouge (photodiode coupée), soit en vert (photodiode passante)**

Globalement, le reste du programme sert à donner des « ordres » à la cellule de départ ; surtout lors des changements de mode de jeu dans l'application (en appuyant sur les boutons), on active différentes fonctionnalités sur la cellule de départ :

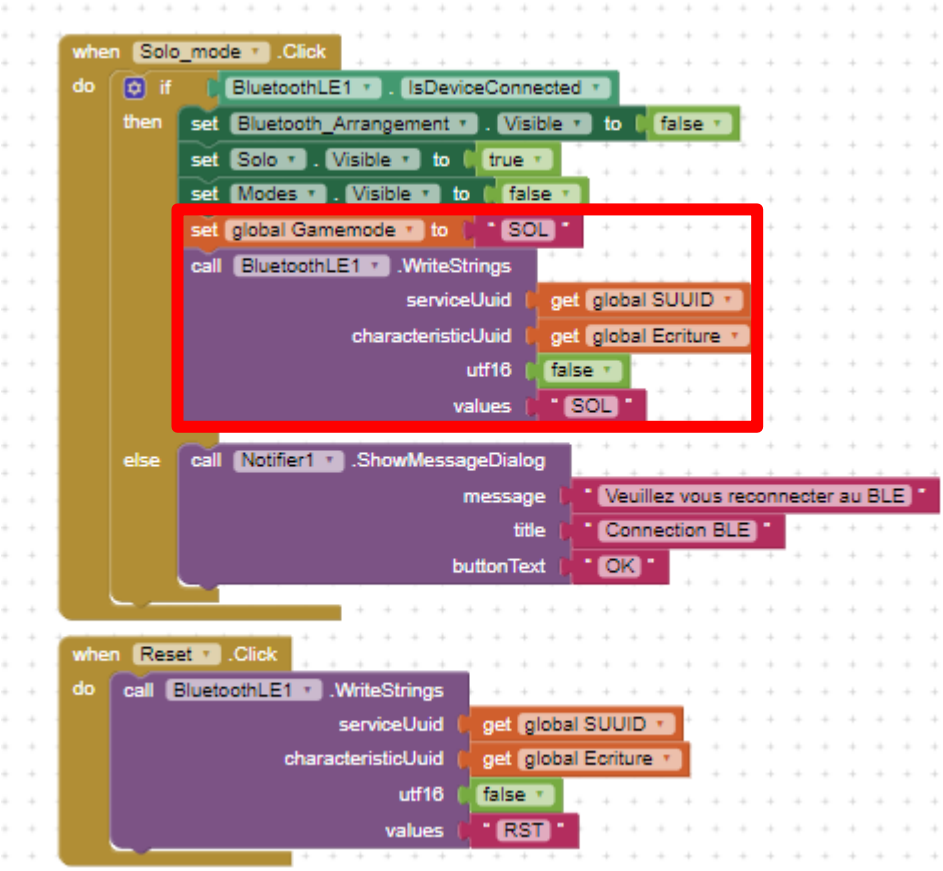



*Figure 20 : Application - BLE écriture*

Par exemple, ouvrir le mode de jeu « Solo » change le mode de jeu à « S », puis appelle la fonction « update() » qui – en fonction du mode de jeu – a elle-même des effets différents. On peut également demander au programme de remettre à 0 les compteurs avec la commande « RST » :

```
void BleOnDataWrittenCallback(const GattWriteCallbackParams *params)
{
    char reception[UART_BUFFER];
    char commande[3];
    if (params->handle == uartServicePtr->getTXCharacteristicHandle()) {
        // Copie de la chaine reçue dans reception
        sprintf(reception,"%s", params->data);
        // Copie dans la chaine commande des deux premier caracteres de la chaine reception
        sprintf(commande,"%c%c%c", reception[0], reception[1], reception[2]);

        if( strcmp(commande, "RST" )==0 ) {
            Timer_Reset();
            update();
        }
        if( strcmp(commande, "SOL" )==0 ) {
            gamemode = 'S';
            laser=1;
            update();
        }
```

*Figure 21 : Commandes en écriture Mbed*

# Technologie des circuits

## Design de la carte

Pour mener à bien notre projet, nous avons décidé dans un premier temps d'utiliser comme base un ObCP pour la cellule de départ ainsi qu'une carte Nucleo F411 pour l'arrivée. Il fallait donc rajouter les fonctionnalités dont nous avions besoin et qui n'étaient pas encore présentes sur chacune de ces cartes. Nous devions ajouter à notre prototype initial les composants suivants :

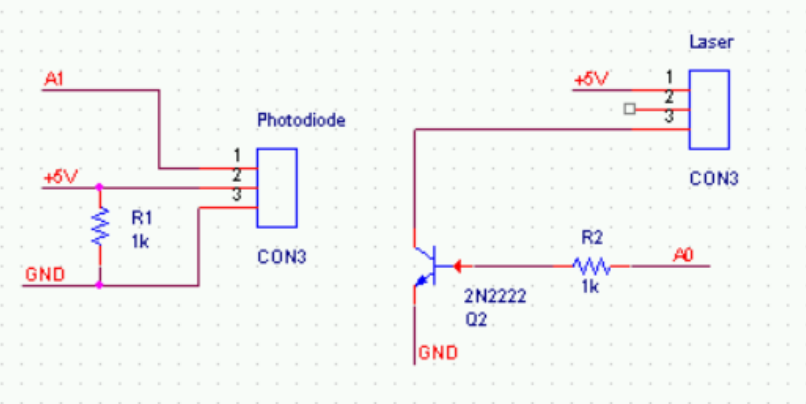
| Shield pour le départ | Shield pour l'arrivée |
|---|---|
| Un laser (1) | Un laser (1) |
| Une photodiode (2) | Une photodiode (2) |
| Un module HC-05 (3) | Un module HC-05 (3) |
| Un module Radio - Fréquences ALPHA - TRX433S (4) | Un module Radio - Fréquences ALPHA - TRX433S (4) |
| Un module récepteur Radio - Fréquences (5) | Un module émetteur Radio - Fréquences (6) |

Vous pouvez vous référer aux chiffres Figure 22 p29

Pour ce faire, nous avons conçu deux shields (un pour le départ et un pour l'arrivée), incluant les composants précédemment listés. Nous avons décidé de disposer différents types de communication ( un module HC-05, un module RF ALPHA - TRX433S, ainsi qu'un récepteur RF pour le départ et émetteur pour l'arrivée), afin de réaliser nos différents tests et de trouver le moyen optimal de transmettre les données.
Concernant la mise en place de ces shields, nous avons listé les fonctions auxquelles les shields doivent répondre, chaque composant est le centre d'une fonction. Ensuite, nous avons recherché dans les documentations. Ainsi nous disposions des données concernant le voltage en entrée à respecter, mais également leur montage. On se propose d'étudier les montages de chaque composant dans la suite.

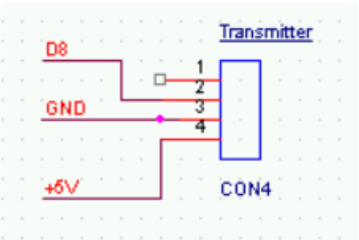
### Laser et photodiode

Nous nous sommes contentés de reproduire ces montages pour la photodiode, et pour le laser. Nous souhaitions rajouter sur le laser une fonction d'interrupteur pour pouvoir contrôler son allumage. Nous avons simplement rajouté un transistor 2N2222.




### Connecteur pour l'émetteur RF

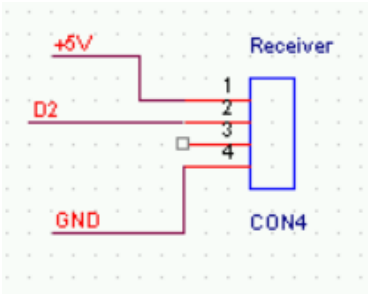
Pour ce connecteur, nous avons regardé les documentations, sur lesquelles il nous était simplement indiqué de brancher une broche sur VCC, une broche data sur D8 pour recevoir le signal à transmettre, et une broche sur le ground. La dernière correspondait à l'antenne. Nous avons soudé un fil car, après divers tests d'antenne, le fil s'avérait être la solution la meilleure. Le schéma de branchement est simplement le suivant :




### Connecteur pour le récepteur RF

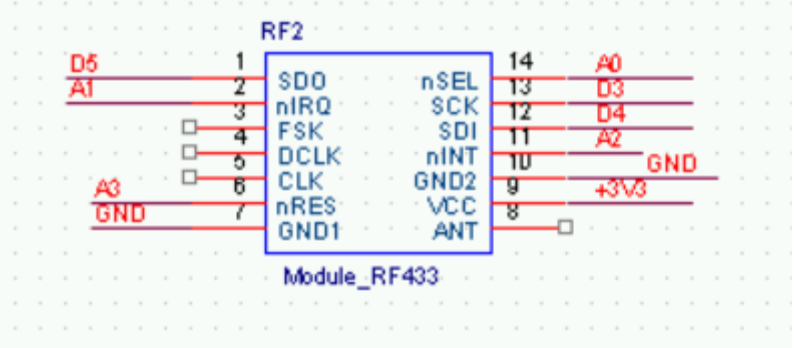
Nous avons procédé de la même manière pour le connecteur destiné à accueillir le récepteur : nous avons étudié la documentation. Celle-ci prescrivait de la même manière que précédemment, de simplement brancher les broches sur les bons pins :
Une broche sur VCC, une broche pour transmettre les datas reçues par le récepteur à la carte sur D2, une broche sur le ground.

Le récepteur compte deux broches pour les datas, cependant ces deux pins sont identiques et interchangeables. Il nous en suffisait d'un seul que nous avons connecté à D2 pour transmettre les données à la carte.  Le schéma est le suivant :



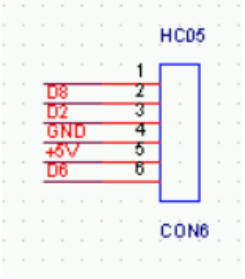

**Un module Radio - Fréquences ALPHA - TRX433S**

De même, les branchements étaient décrits dans la documentation de ce module, il suffisait de vérifier la compatibilité des pins par rapport aux fonctions exigées par le composant.




**Connecteur pour les modules HC-05**

De même, les branchements étaient décrits dans la documentation de ce module. Aucun autre composant n'était nécessaire pour faire fonctionner correctement les modules HC-05. Il suffisait de les brancher de la manière suivante :



## Choix du placement

Nous avons décidé de placer les différents composants de la manière suivante sur la carte :

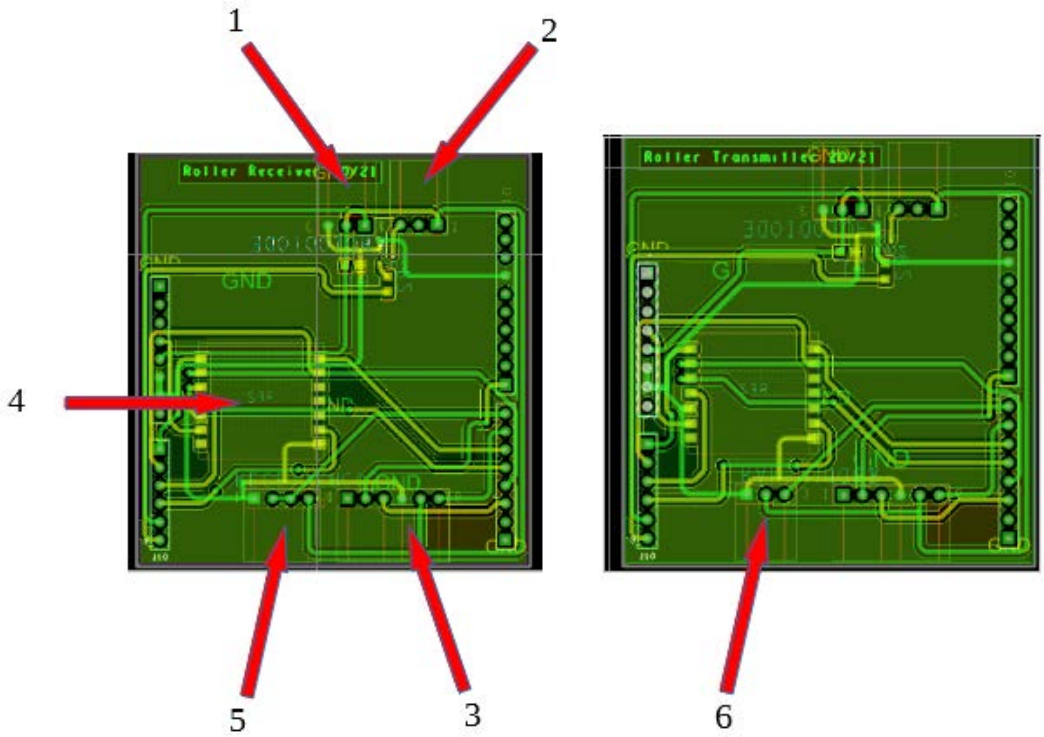



*Figure 22 : Premiers shields pour le départ (gauche) et pour l'arrivée (droite)*

Par rapport aux placements des cartes dans la cellule finale, nous avons placé côte à côte les connecteurs destinés à accueillir le laser et la photodiode côte-à-côte afin de faciliter la détection laser. De l'autre côté de la carte, nous avons décidé de placer les connecteurs destinés à accueillir les composants pour les différents types de communication (émetteur RF, récepteur RF, modules HC-05). Par ailleurs, étant donné qu'aucune autre condition sur la position des composants n'était imposée, nous les avons placés de manière à simplifier les connexions.

## Evolution entre première et seconde carte & améliorations à apporter

Etant donné que nous souhaitions disposer d'un système autonome, nous avons décidé après réflexion de ne pas inclure de port de batterie, mais plutôt d'utiliser une power Bank pour remplir cette fonction. Celle-ci se branche directement sur le port d'alimentation de la F411.
Voici le récapitulatif des composants présents sur les shields :

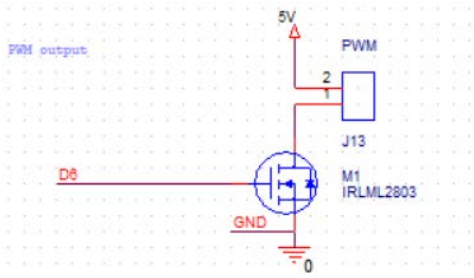
| Shield pour le départ avec BLE | Shield pour le départ sans BLE | Shield pour l'arrivée |
|---|---|---|
| Un laser (1) | Un laser | Un laser |
| Une photodiode (2) | Une photodiode | Une photodiode |
| Un module récepteur Radio - Fréquences (3) | Un module récepteur Radiofréquences | Un module émetteur Radio - Fréquences (6) |
| Une sortie PWM et un haut-parleur (4) | Une sortie PWM et un haut-parleur | Une sortie PWM et un haut-parleur |
| Un module BLE (5) | | |

Vous pouvez vous référer aux chiffres **Figure 23** (ci-après).

Les différents composants communs aux premiers PCB sont connectés de la même manière sur ces PCB. Nous avons simplement rajouté des sorties PWM, ainsi qu'un module BLE sur l'un des shields. Nous les avons connectés de la manière suivante :

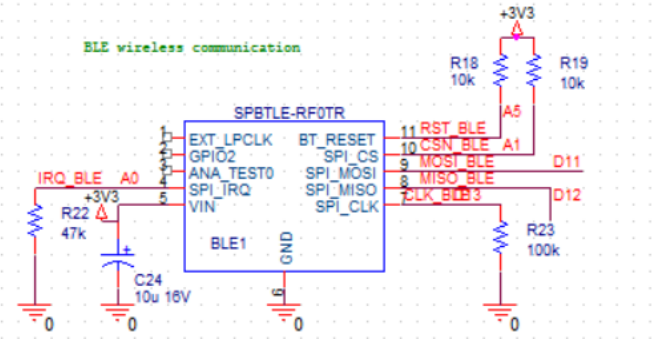
**Sortie PWM**

Pour la sortie PWM, nous voulions être capable de l'activer quand nous le voulions. Pour cela, nous avons ajouté un transistor IRLML2803.
Les branchements du bornier sont très simples : un pin sur VCC et l'autre relié au transistor.
Le transistor est relié en entrée à D6. Lorsque le signal de D6 est nul, la sortie du transistor est sur GND, le PWM est éteint. Inversement lorsque le signal D6 passe à 1.




## Module BLE

Concernant ce module, nous nous sommes une fois de plus basés sur les branchements indiqués dans la documentation du BLE pour connecter le BLE sur le shield en vérifiant qu'il n'y ait pas de conflits au niveau des données arrivait sur les pins. Nous n'avions cependant pas le besoin d'ajouter la mémoire comme sur l'ObCP.




Ce travail a abouti aux trois PCB ci-dessous :

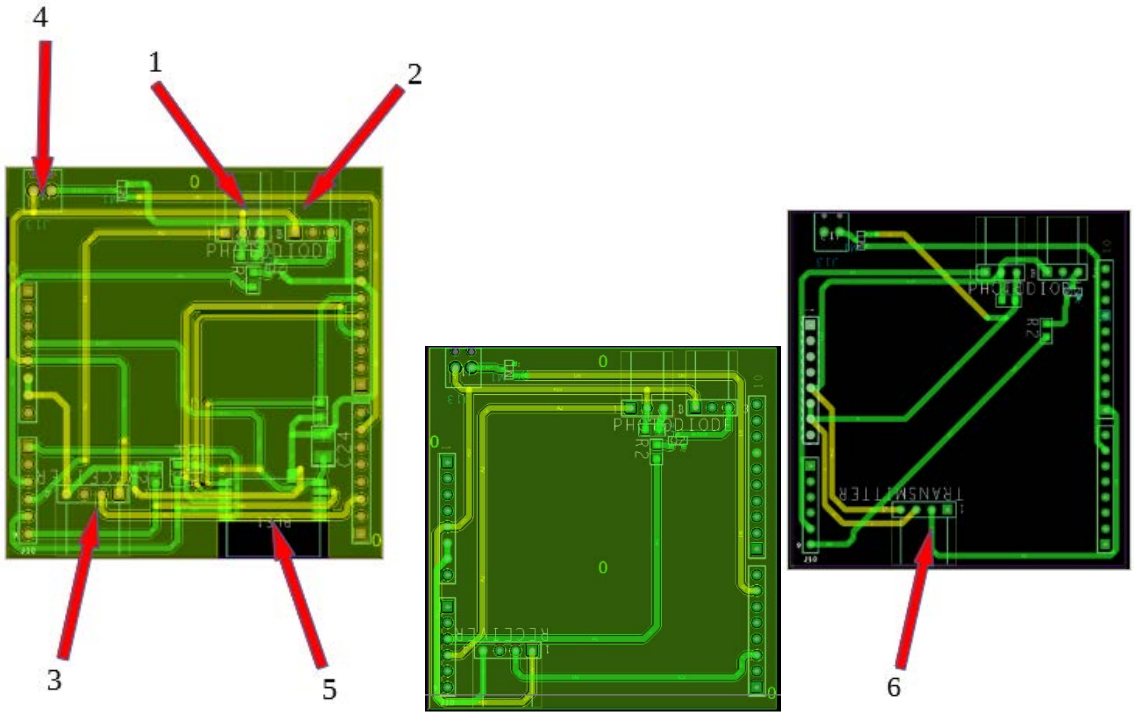



*Figure 23 : Shields de départ avec BLE (en haut à gauche), de départ sans BLE (au milieu) et d'arrivée (à droite)*

## Difficultés de réalisation & Recherche d'erreurs et correction

Concernant nos shields, nous avons rencontré plusieurs difficultés. D'abord, lors de la conception, nous avons fait plusieurs erreurs. Certains connecteurs ont été inversés, ce qui provoque une différence entre le schéma électrique de la carte et le PCB. Ensuite, nous avons utilisé des pistes nécessitant l'utilisation de fonctions interrupt. Cependant, nous n'avons pas vérifié que les pins de la carte supportent ces fonctions. Nous avons donc eu un problème dans le traitement des données pour allumer ou éteindre le laser. La solution a été d'inverser des pistes.
Concernant le module BLE, nous avons rencontré le même problème que précédemment : nous avons connecté la piste CLK du module sur une piste qui ne supportait pas les fonctions nécessaires. Cependant, après ces modifications, le module BLE fonctionnait bien (détectable par un téléphone).
Le dernier module que nous n'avons pas su faire fonctionner est le module émetteur - récepteur RF des premiers shields.

Pour les améliorations futures, l'intégration d'un port pour une batterie serait intéressante, bien que nous ayons trouvé la solution des power Bank pour alimenter les cartes.

# Nouveautés et améliorations à apporter

Nous avons réussi au cours de ce projet à utiliser deux types de communication totalement dissociées mais marchant indépendamment. La grande difficulté reste la juste utilisation des bibliothèques pour le BLE. En effet, nous utilisons ici la méthode « ble.waitForEvent() » afin de mettre en veille le BLE, mais cela nous empêche d'envoyer les informations quand nous le souhaitons (par exemple à chaque fois qu'une porte laser est coupée). En effet nous sommes ici limités à une actualisation toutes les 0.01 seconds car augmenter la fréquence a pour effet de faire crash la connexion avec le téléphone. Si on vaut avoir une précision inférieure au 1/100 secondes, il faudra donc trouver une solution à ce problème
Nous pourrions également augmenter la tension pour certains éléments, comme l'émetteur RF ou le PWM pour avoir de meilleures performances (le dimensionnement de la batterie serait alors à rectifier).

# Normes applicables & Certifications requises pour la mise sur le marché

| Sujet | Directive européenne | Justification | Normes harmonisées |
|---|---|---|---|
| Electro Magnétique | Directive CEM (compatibilité électromagnétiques) 2014/30/UE | Applicable. Appareil susceptible d'émettre ou d'être affecté par des perturbations électromagnétiques | EN 55014-1 et 55014-2 Compatibilité électromagnétique : exigences pour les appareils électroménagers, outillages électriques et appareils analogues |
| BLE, modules RF | Directive RED (Equipement Radioélectriques) 2014/53/UE | Applicable. Appareil contient des modules Radiofréquences, un module BLE | |
| Enveloppes Carter Protection | Directive de protection procurée par les enveloppes pour le matériel électrique | Applicable. Indice IP / IK visé, avec une tension <72.5kV | Norme CEI 60529 et 62262 : Protection enveloppes indice IP / IK |
| Laser | Directive 2006/25/CE (Protection contre l'exposition aux rayonnements optiques artificiels) | Applicable. Appareil possédant un laser de longueur d'onde comprise entre 180 nm et 1mm. Étiquetage selon la classe du laser. | Norme IEC 60825-1 : Classification laser. |
| Fin du cycle de vie du produit | ROHS (2011/65/UE) (Réduction des substances dangereuses ;<br><br>DEEE (2012/19/UE) (déchets de composants électroniques) | Applicable. Comporte des composants électroniques.<br><br>Nécessité d'un étiquetage adapté. | |
| Batterie | Directive 2013/56/UE relative aux piles et accumulateurs ainsi qu'aux déchets de piles et d'accumulateurs | Applicable. Sera doté d'une batterie externe ~1000mAh<br><br>Nécessité d'une juste utilisation de la batterie pour un système portable. | IEC 60086-1 et suivants : piles électriques.<br><br>- IEC 62133-2 : accumulateurs alcalins destinés à l'utilisation dans des applications portables.<br><br>- IEC 61960 : accumulateurs alcalins pour applications portables. |

Il y a une multitude de normes locales associées et respectées par chacun des composants utilisés. Ils sont présents dans les documentations.

# Remerciements

Nous souhaitons enfin remercier Joël IMBAUD et David VERNIER pour leur aide et l'attention qu'ils nous ont procuré durant ces 6 mois. C'est en grande partie grâce à eux si nous avons pu trouver des solutions à nos problèmes !
Merci également à Frédéric MOULIN pour le temps court mais agréable que nous avons pu passer à discuter de ce projet.

Au-delà du projet, la quasi-totalité des cours était en cohérence avec le projet mené tout au long du semestre, c'était en somme un véritable plaisir de suivre ce cursus !

Tanguy et Dorian

# Table des illustrations