L’AIRSER

BEDIA Julien, CAPPIELLO Raphaël,
KHAMMA Houssameddine, N’GUYEN VAN KY Nicolas

SOMMAIRE

Introduction

Le développement de la technologie des drones a révolutionné la manière dont les militaires mènent des opérations sur le terrain. L'utilisation de drones permet aux forces armées de surveiller des zones dangereuses ou inaccessibles, de recueillir des informations sur l'ennemi et de mener des attaques ciblées. Dans le cadre de ce sujet, l'armée dispose de deux drones, l'un en l'air et l'autre au sol, pour effectuer des tirs sur des cibles désignées. Le défi consiste à déterminer quel drone a effectué les tirs, afin de pouvoir attribuer la responsabilité de l'attaque. Dans cet environnement militaire, il est crucial de pouvoir détecter et identifier avec précision les acteurs impliqués dans une attaque, afin de garantir la sécurité des troupes et de minimiser les dommages collatéraux.

Définition du besoin

Nous avons besoin de distinguer deux types de drones, les drones aériens et les drones terrestres, en fonction de la trame de données qu'ils envoient à nos récepteurs. Pour cela, nous aurons besoin d'un émetteur prototype équipé d'un laser pour envoyer la trame de données et d'un récepteur unique prototype qui décodera la trame de données reçue pour identifier le drone..

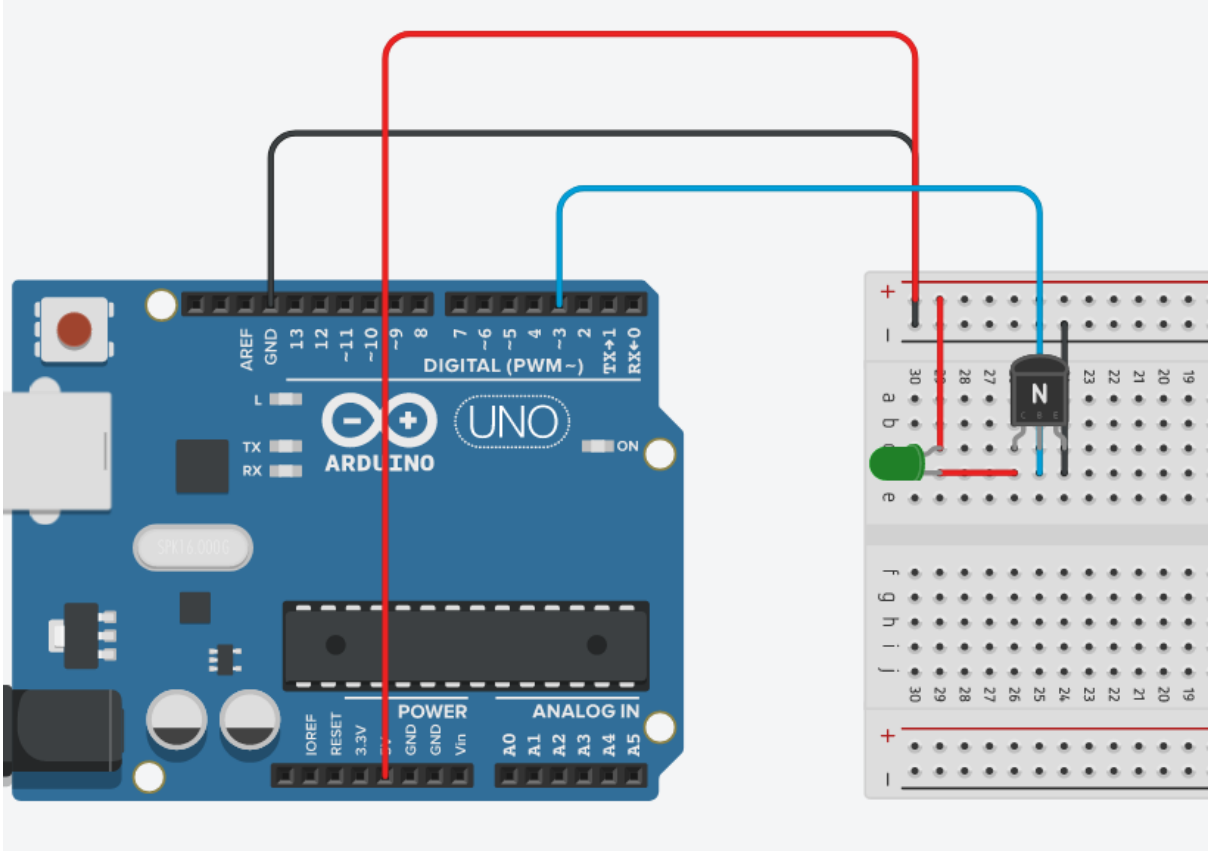
Définition de la solution technique :

Le système de détection de drones consiste en un émetteur équipé d'un laser pour envoyer une trame de données et un récepteur unique qui décode la trame de données reçue pour identifier le drone qui a tiré. La trame de données envoyée par le drone aérien commence par les bits (110) et est suivie par les bits (10), tandis que la trame de données envoyée par les drones terrestres commence par les bits (110) et est suivie par les bits (01).

Le récepteur décodera la trame de données reçue et identifiera le type de drone à cibler en fonction des bits suivants les bits (110). Le système sera conçu pour une utilisation en extérieur et devra être capable de fonctionner dans des conditions météorologiques favorables.

Emetteur

Prototype



Prototype de l'émetteur laser qui va permettre d'émettre une trame différente en fonction du type de drone (aérien ou au sol) à notre cible.

Composants

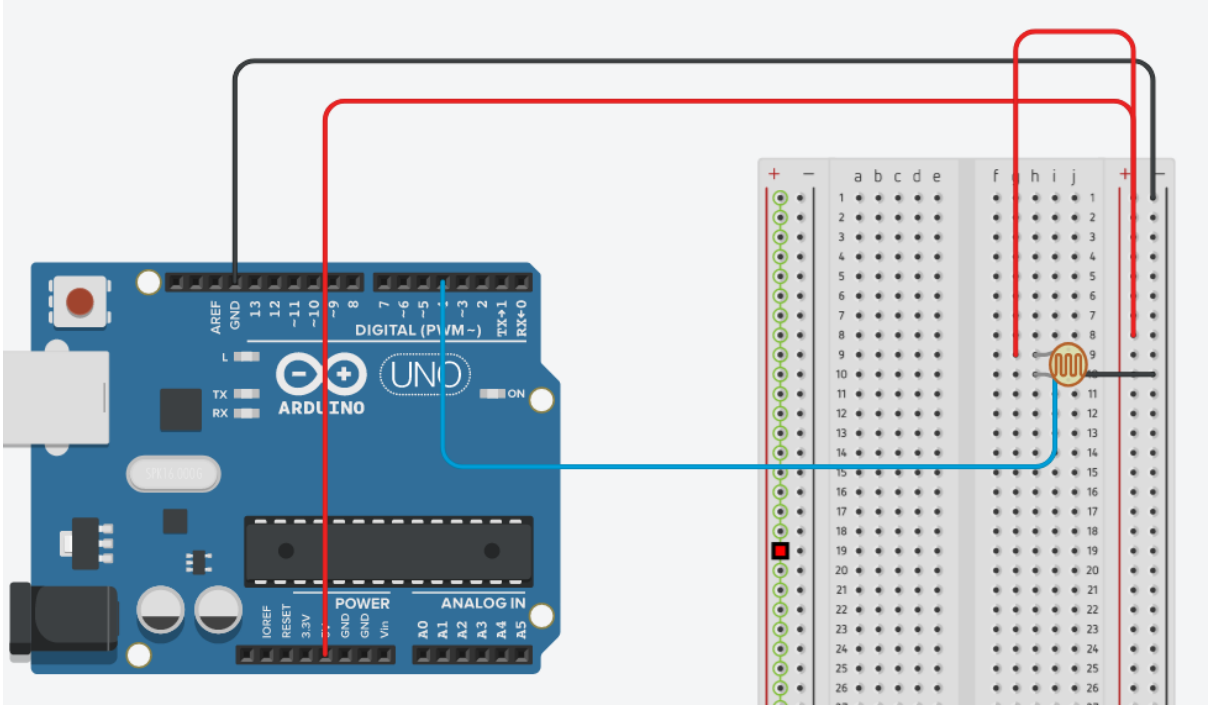
Arduino ESP 32
Émettre la trame en fonction du type de drône
Drone aérien : 11001
Drone au sol : 11010

Laser
Longueur d'onde : environ 520nm (vert)

Transistor
Référence : NPN Bipolaire
Le transistor va permettre de faire passer une tension de 5V et il nous permet de faire la variation de la tension via un PIN sur l'arduino et ainsi émettre la donnée (1 ou 0).

Récepteur

Prototype



Prototype du récepteur qui va permettre de récupérer la trame envoyé par l'émetteur et ainsi déterminer quel est le type de drône qui a tiré sur la cible.

Composants

Composants - référence chaque composant + utilité

Arduino ESP 32
Réceptionner la trame émise par l'émetteur laser.

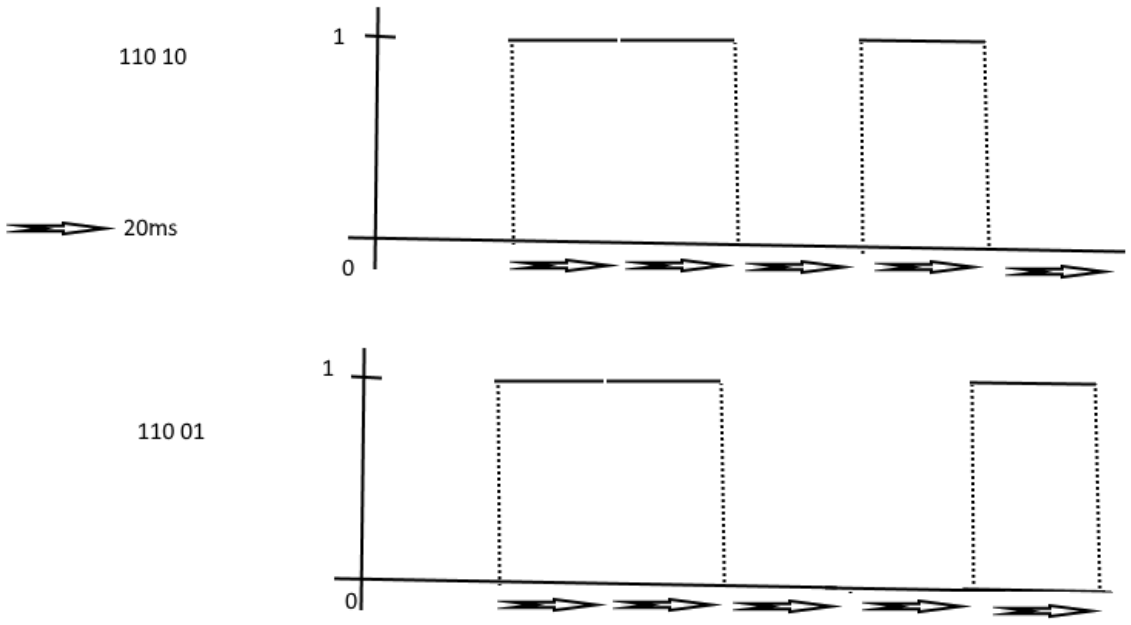
Photorésistance
Référence : LDR CDS 5mm - B07F9NNCZ5
Pic spectral : 540 nm
Résistance à la lumière (10 lux) : 5-10 Kohm
Temps de réponse: 20ms
La photorésistance va permettre de faire varier la valeur reçu dans un PIN sur l'arduino en fonction de la lumière reçu et ainsi déterminer si le laser a émis un bit ou pas.

Filtre
Filtre de couleur pour filtrer la lumière du soleil (bleu + jaune)

Lentille de Fresnel
La lentille va nous permettre de concentrer le rayon du signal laser sur une longue distance dans un point plus précis de notre récepteur, notamment sur la photoresistance.

Programmation

La programmation des 2 arduinos va nous permettre de configurer l'émetteur pour envoyer 2 trames différentes dans un espace de temps égal à : …
Et configurer le récepteur pour réceptionner la trame émise en 100ms (20ms par bit).



Arduino émetteur

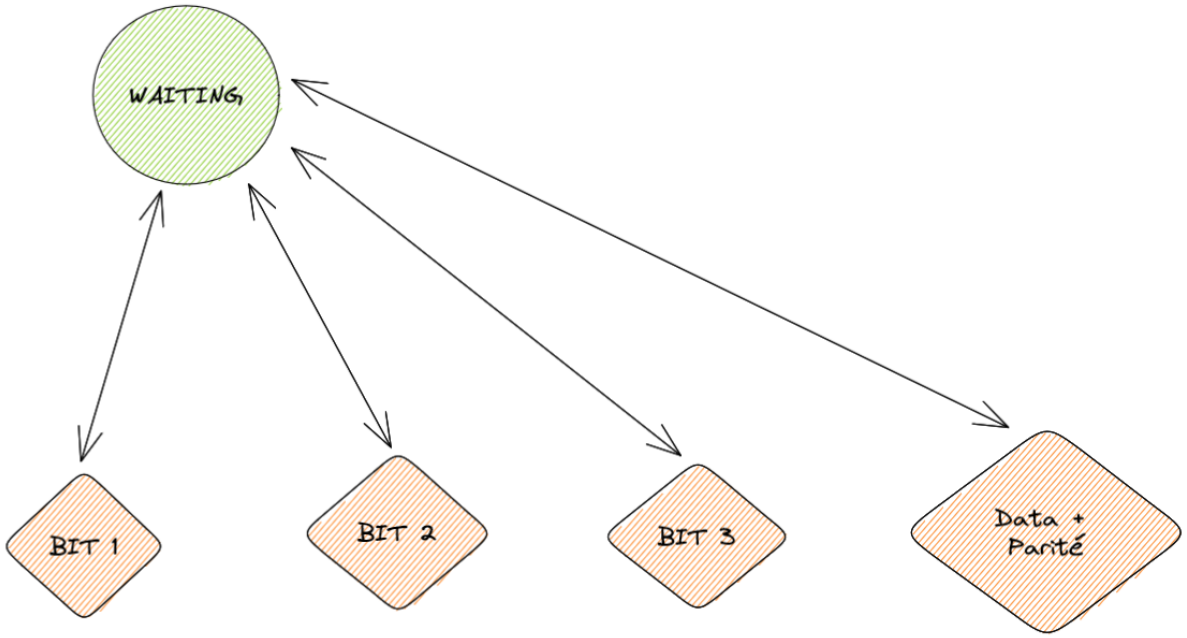
Code : Voir le fichier emetteur.cpp

Pour émettre une valeur 1 l'émetteur va allumer le laser pendant 4ms puis s'éteindre pendant 18ms
Pour émettre une valeur 0 l'émetteur va s'éteindre pendant 19ms

Arduino récepteur

Code : Voir le fichier recepteur.cpp

Pour réceptionner les valeurs le récepteur va écouter pendant 20ms et déterminer si il y aeu une variation de tension perçu avec la photoresistance et ainsi déterminer si l'émetteur à envoyer une valeur 1.



La Finite State Machine (FSM) est composé de 5 états :

- State WAITING : Écoute les bits d'entrées toutes les 20ms et détermine dans quel état entrer en fonction de l'index dans la trame
- State BIT 1 : Vérifie si le bit de départ est égal à 1, prépare le premier bit de la trame et change l'index dans la trame pour en déduire l'état suivant
- State BIT 2 : Vérifie si le bit de départ est égal à 1, prépare le premier bit de la trame et change l'index dans la trame pour en déduire l'état suivant
- State BIT 3 : Vérifie si le bit de départ est égal à 0, prépare le premier bit de la trame et change l'index dans la trame pour en déduire l'état suivant
- State DATA_PARITE : Si on est dans le bit 4 (data) on enregistre la data et on prépare l'index pour passer à la parité
  Si on est dans le bit 5 (parité) on vérifie qu'il soit différent au bit data, si c'est le cas on enregistre la trame et on l'affiche sinon

Résultat final

Ci-dessous, nous pouvons retrouver la FSM ainsi que les résultats obtenus dans la console (la trame) lorsque le laser balaye le récepteur.
Nous remarquons que l'on récupère bien la trame que l'on souhaite transmettre avec le laser.
(Ici une distance de 35m environ à l'ombre en utilisant filtre et lentille de fresnel)
D'autres tests sont nécessaires pour espérer toucher une distance de 100m.

```
108     case STATE_BIT_1:
109       //On vérifie le premier bit de la trame depart
110       if(currentBit == '1') {
111         dataTrame[0] = currentBit;
112         dataTrameIndex++;
113       } else {
114         dataTrameIndex = 0;
115       }
116       // Serial.println("############STATE_BIT_1############");
117       // Serial.println(currentBit);
118       // Serial.println("########################");
119       currentState = STATE_WAITING;
120       break;
121     case STATE_BIT_2:
122       //On vérifie le deuxième bit de la trame depart
123       if(currentBit == '1') {
124         dataTrame[1] = currentBit;
125         dataTrameIndex++;
126       } else {
127         dataTrameIndex =0;
128       }
129       // Serial.println("############STATE_BIT_2#############");
130       // Serial.println(currentBit);
131       // Serial.println("########################");
132       currentState = STATE_WAITING;
133       break;
134     case STATE_BIT_3:
135       //On vérifie le troisième bit de la trame depart
136       if(currentBit == '0') {
137         dataTrame[2] = currentBit;
138         dataTrameIndex++;
139       } else {
140         dataTrameIndex =0;
141       }
142       // Serial.println("#############STATE_BIT_3###########");
143       // Serial.println(currentBit);
```

PROBLÈMES 44 SORTIE TERMINAL CONSOLE DE DÉBOGAGE

```
########################
STATE_DATA_PARITE
STATE_DATA_PARITE
########################
11001
########################
STATE_DATA_PARITE
STATE_DATA_PARITE
```