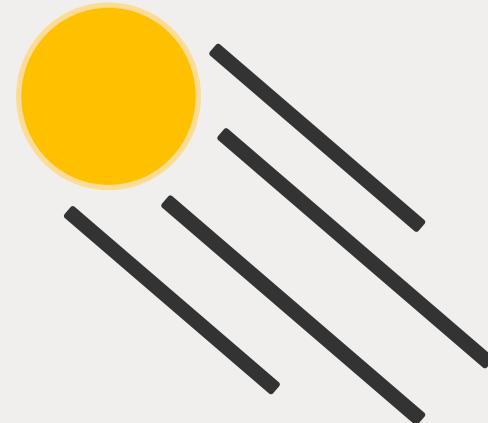


TRAVAIL D'INITIATIVE PERSONNELLE ENCADRÉ



CIBLE AUTOMATIQUE

ZACHARIE BUI
JACOB YVAN

Nb SCEI : 35164



SPORT : AIRSOFT

C'EST QUOI ?



Sc : Adobe stock





SPORT : AIRSOFT

POURQUOI, RAPPORT AU THÈME ?



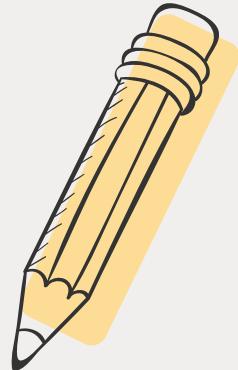
| | |
|--------------|--|
| Sport | Méconnu, en plein développement |
| Santé | Permet de travailler l'endurance |
| Stratégie | Besoin d'établir des stratégies pour la victoire |
| Entraînement | Nécessité d'infrastructures fiables et durables |





Problématique

Comment concevoir, développer et optimiser une cible d'airsoft automatique afin d'améliorer la précision des tirs, les compétences des joueurs, et l'aspect écologique ?



BUT

Première cible

Grande cible

Conclusion

SOMMAIRE

O1**BUT**

Nos idées, ce que l'on a voulu mettre en place

O2**Première cible**

Partie en majorité faite par Zacharie

O3**Grande cible**

Partie faite en majorité par Yvan

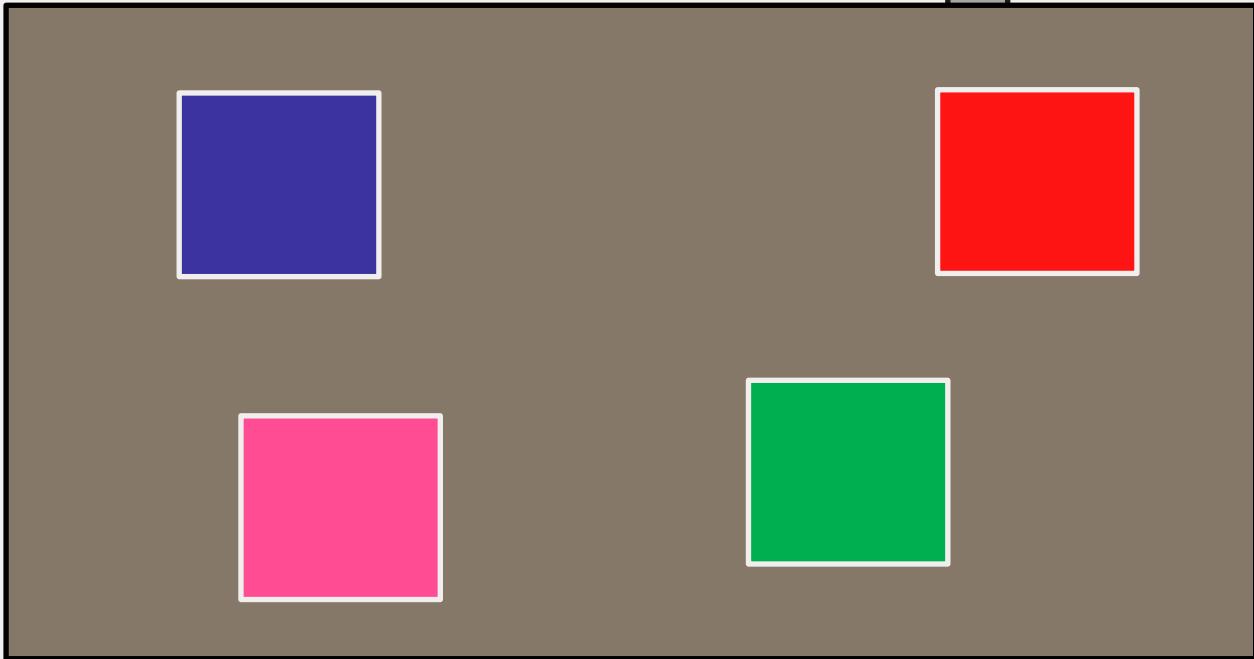
O4**Conclusion**

Sommes-nous satisfaits ?

O5**Annexe**

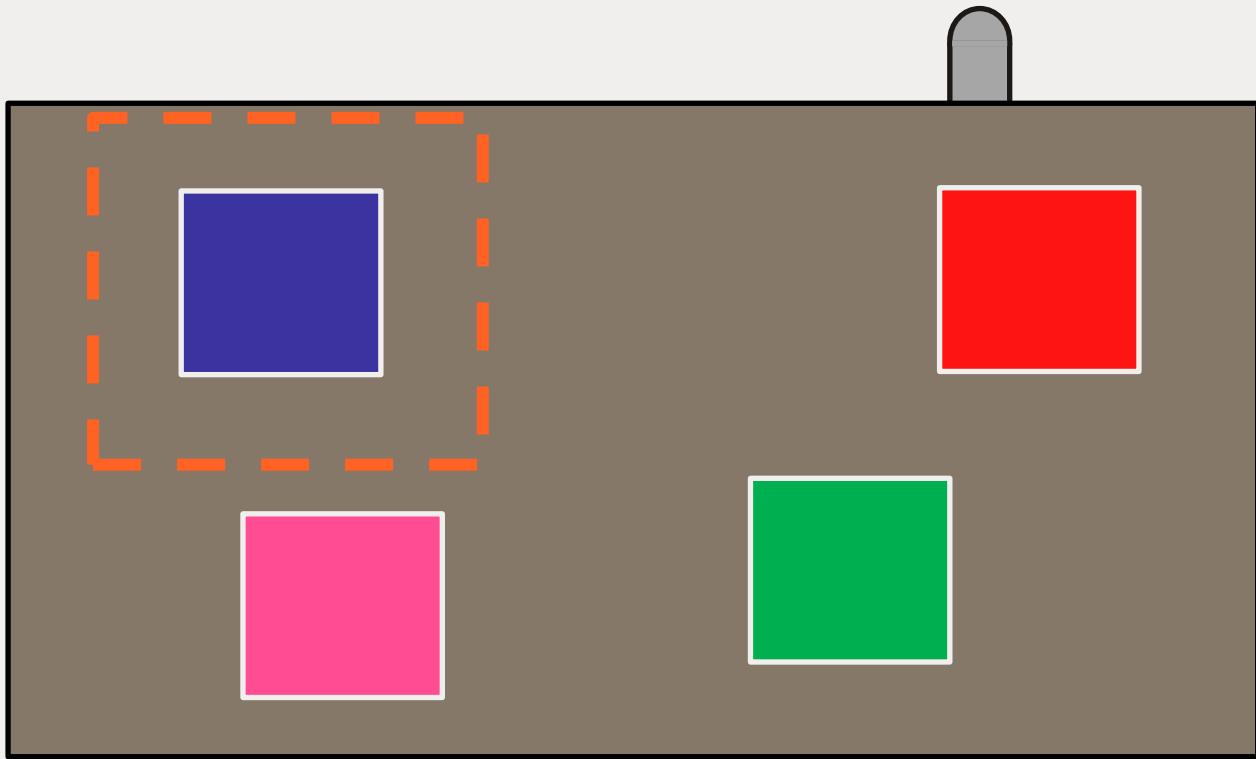
Pour plus de détails



BUT**Première cible****Grande cible****Conclusion****O1****But**

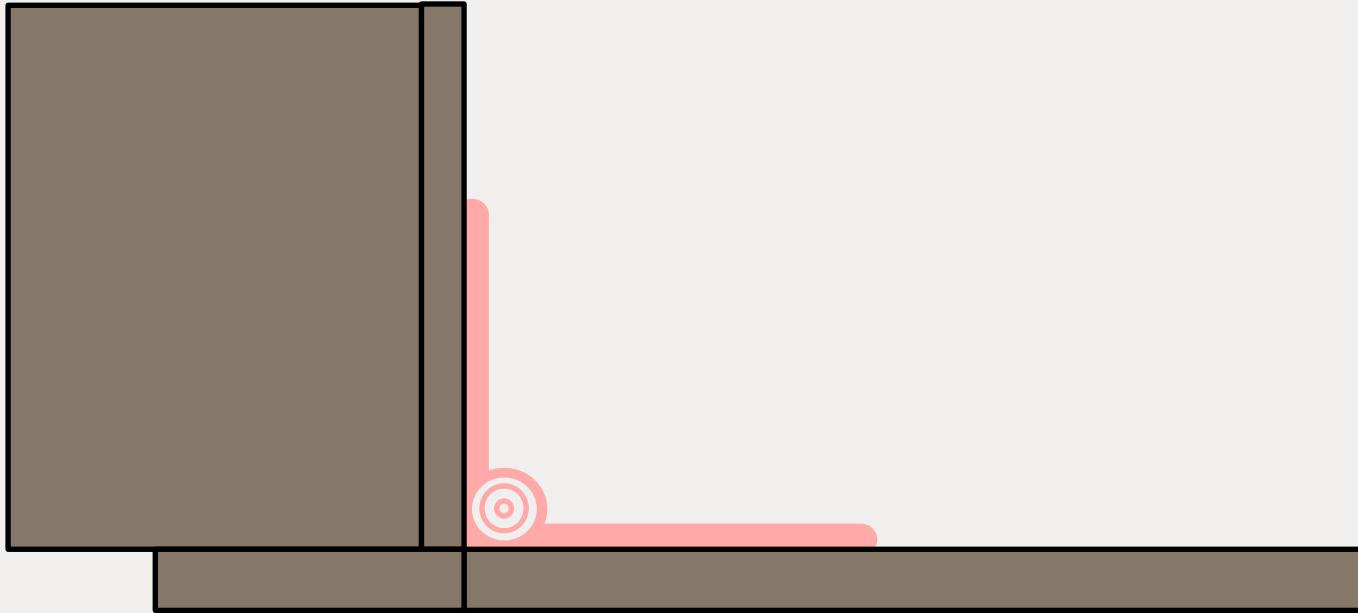
Création d'une cible d'airsoft
automatique



BUT**Première cible****Grande cible****Conclusion****O1****BUT**

Création d'une cible d'airsoft
automatique



BUT**Première cible****Grande cible****Conclusion****O1****BUT**

Création d'une cible d'airsoft
automatique



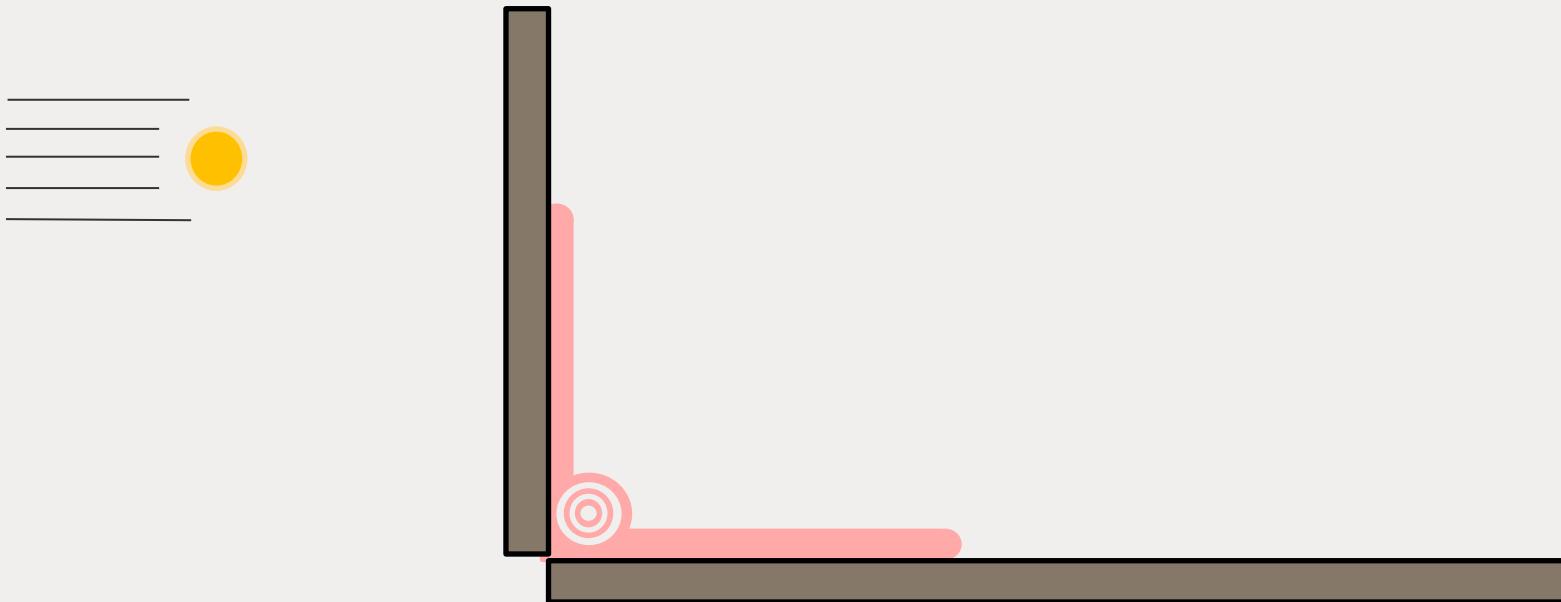
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



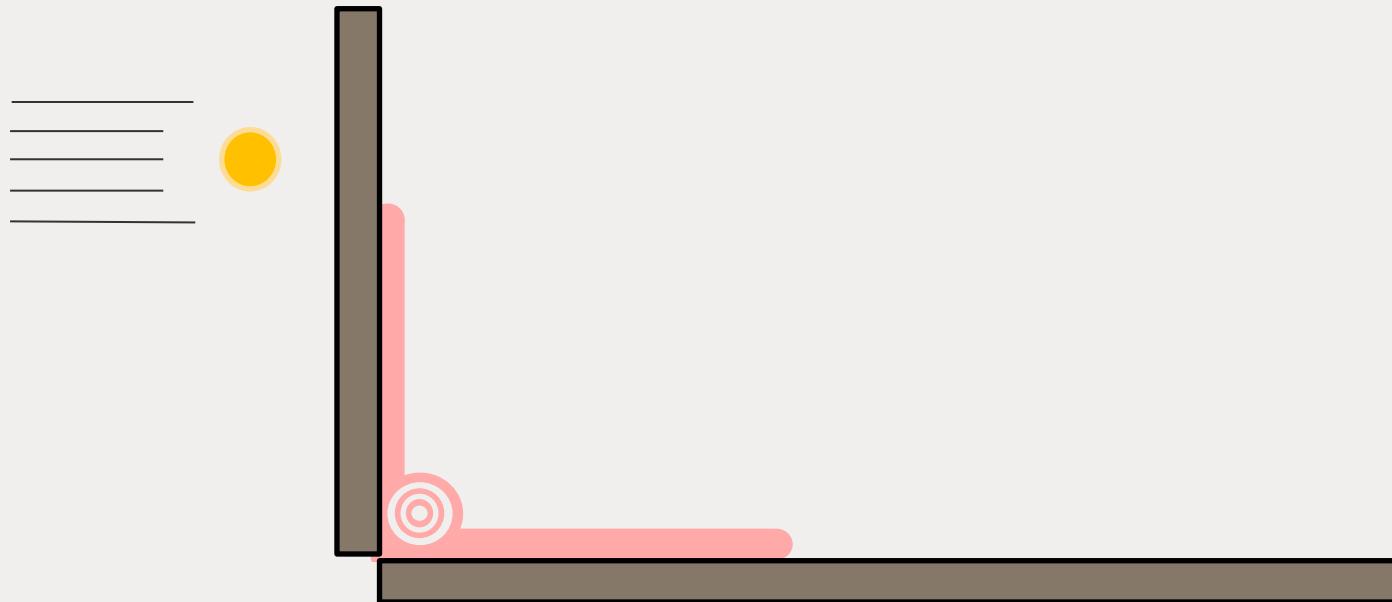
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



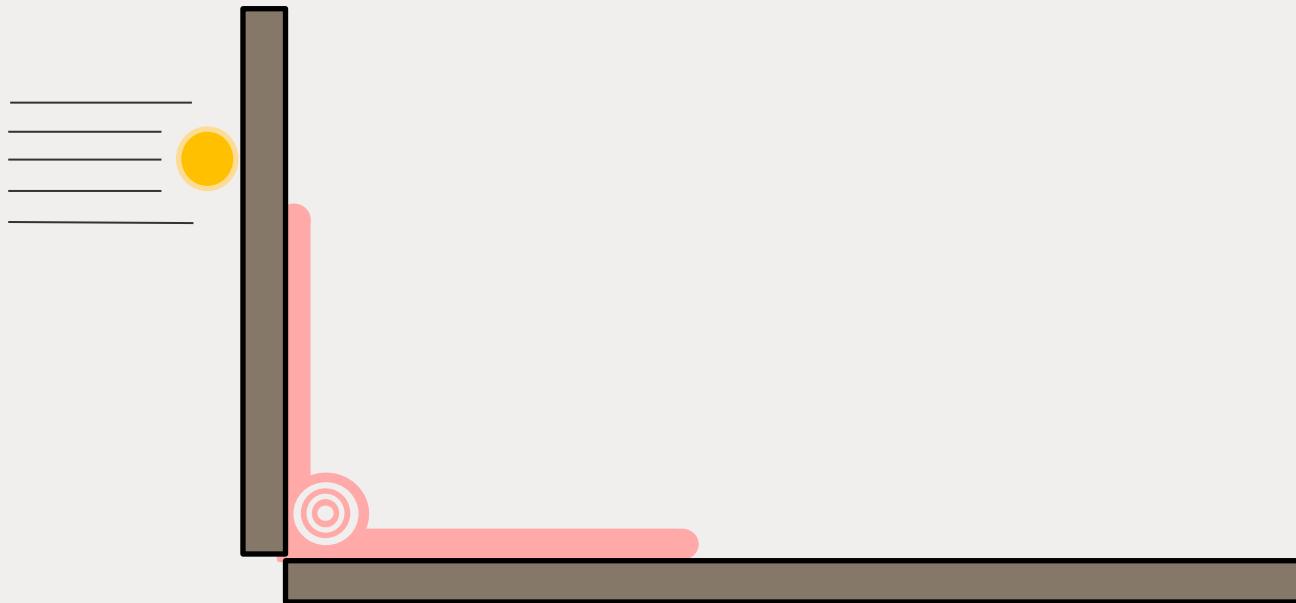
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



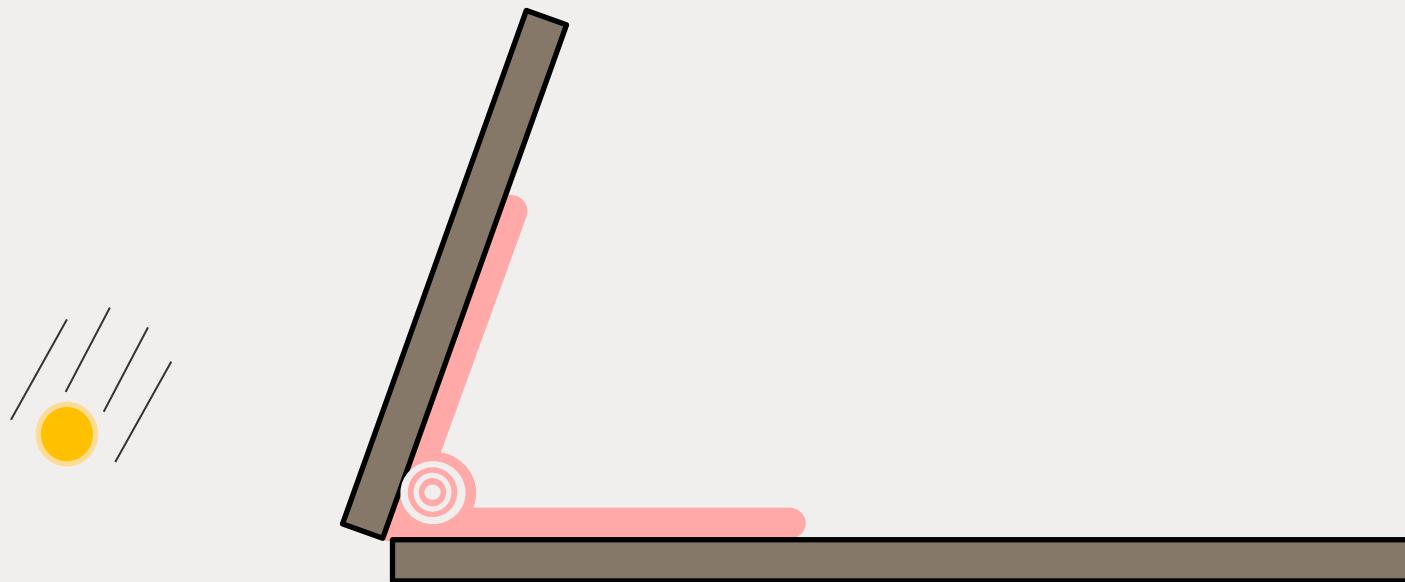
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



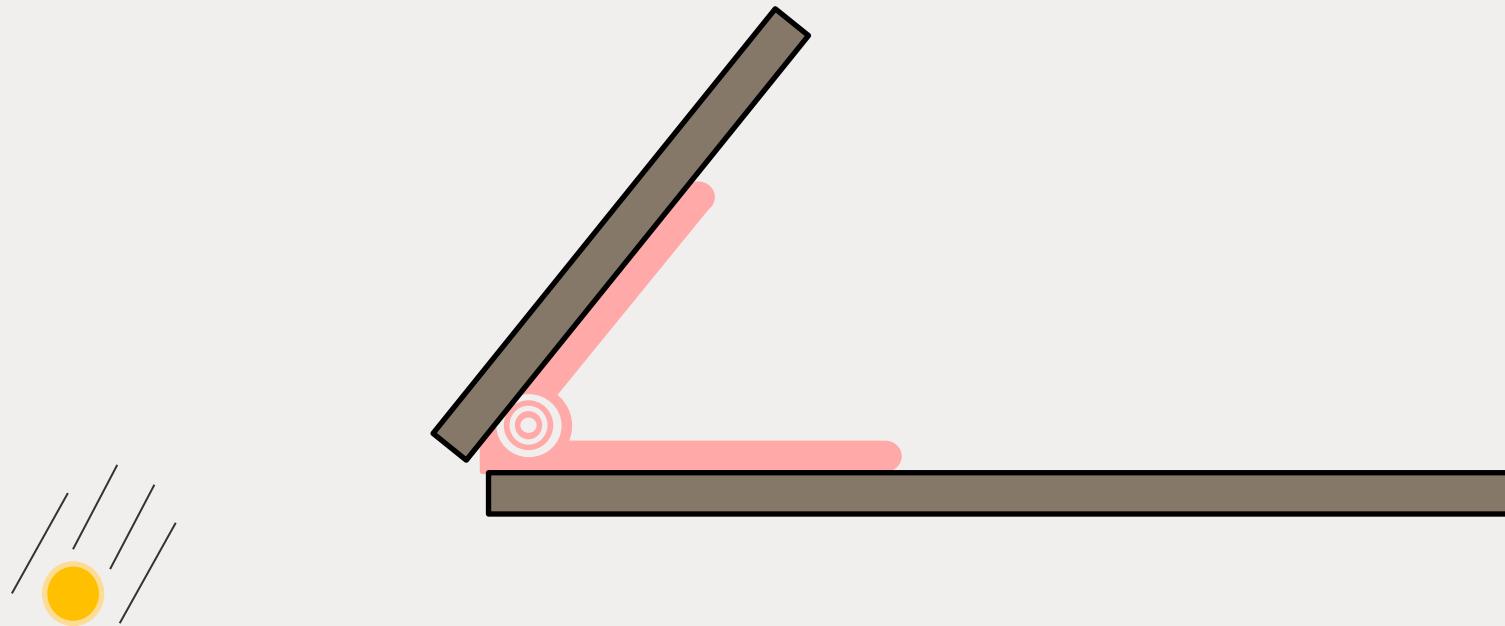
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



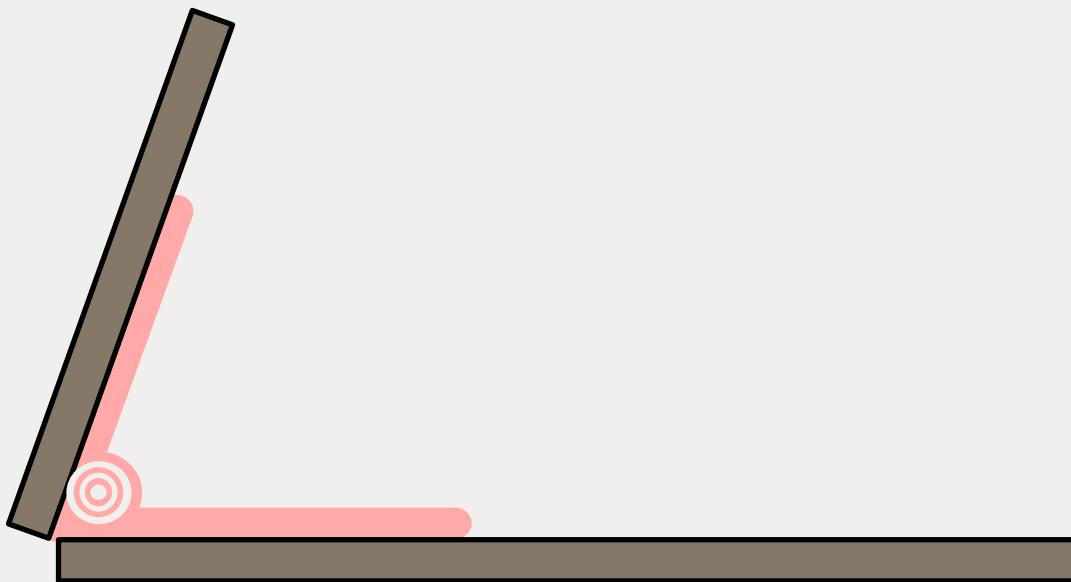
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



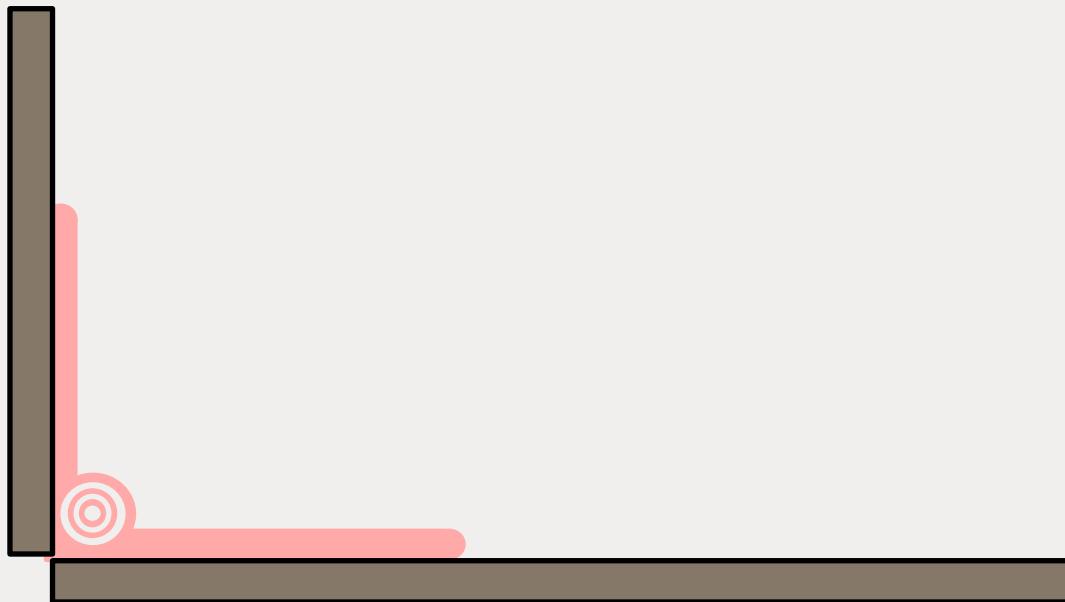
BUT

Première
cible

Grande
cible

Conclusion

Mise en mouvement



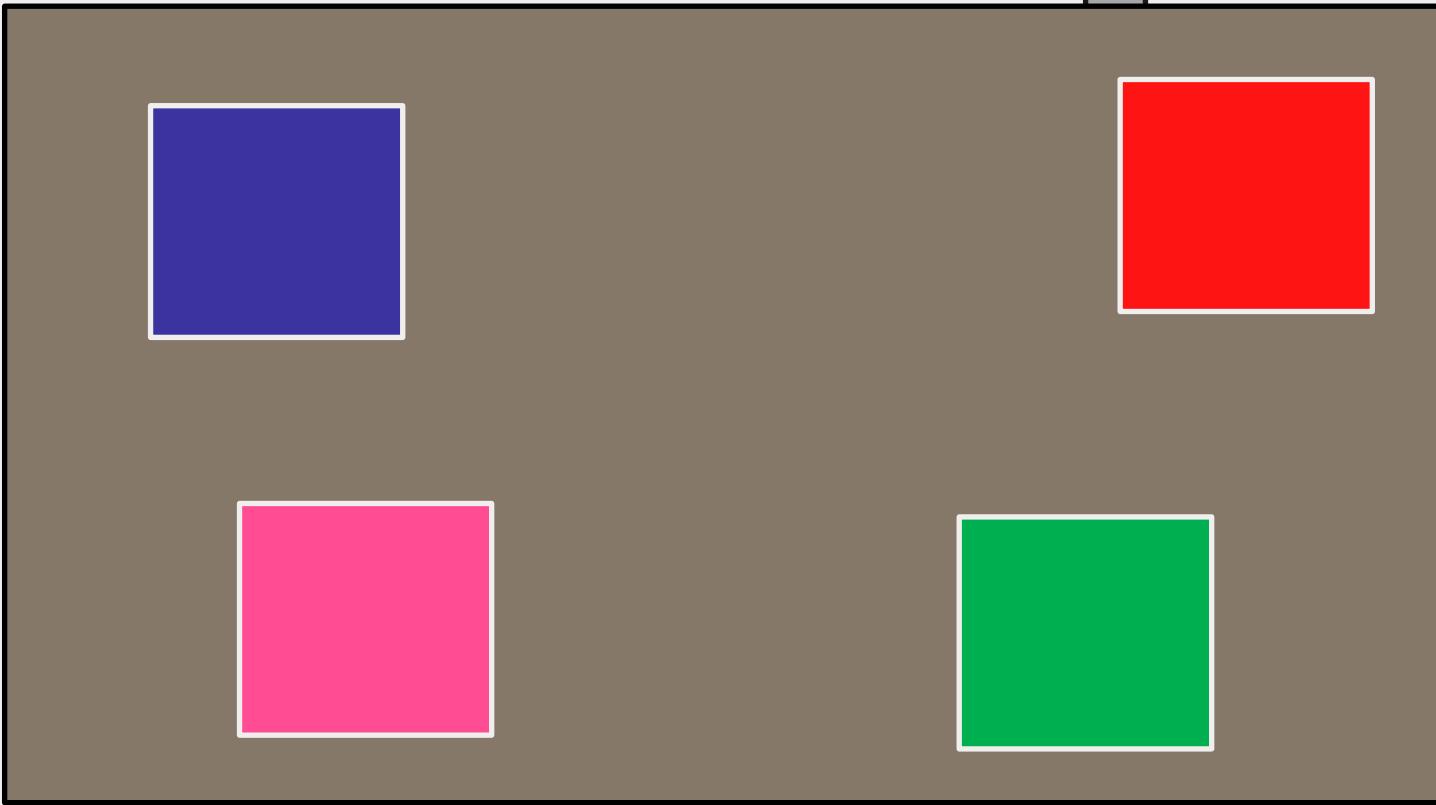
BUT

Première
cible

Grande
cible

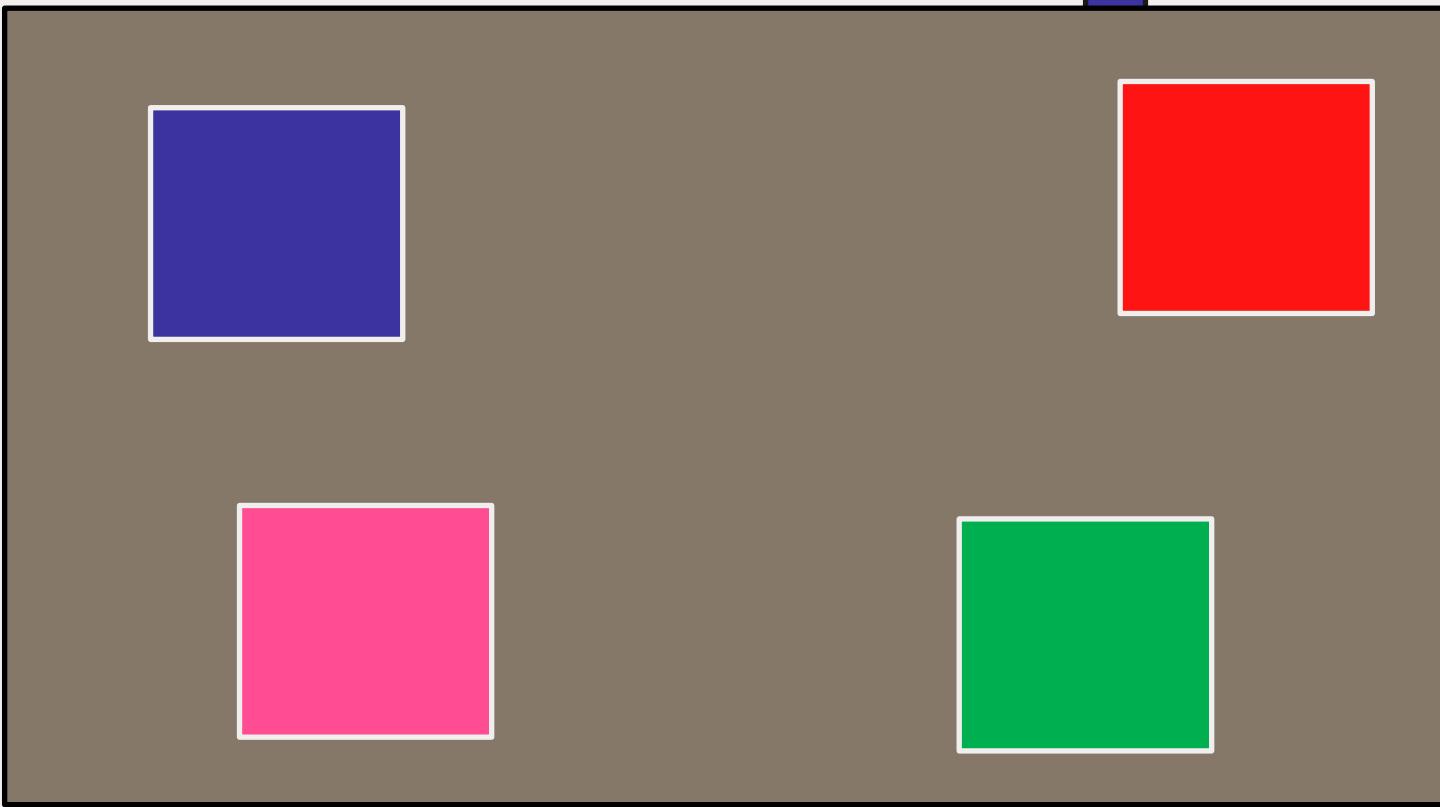
Conclusion

Projet final



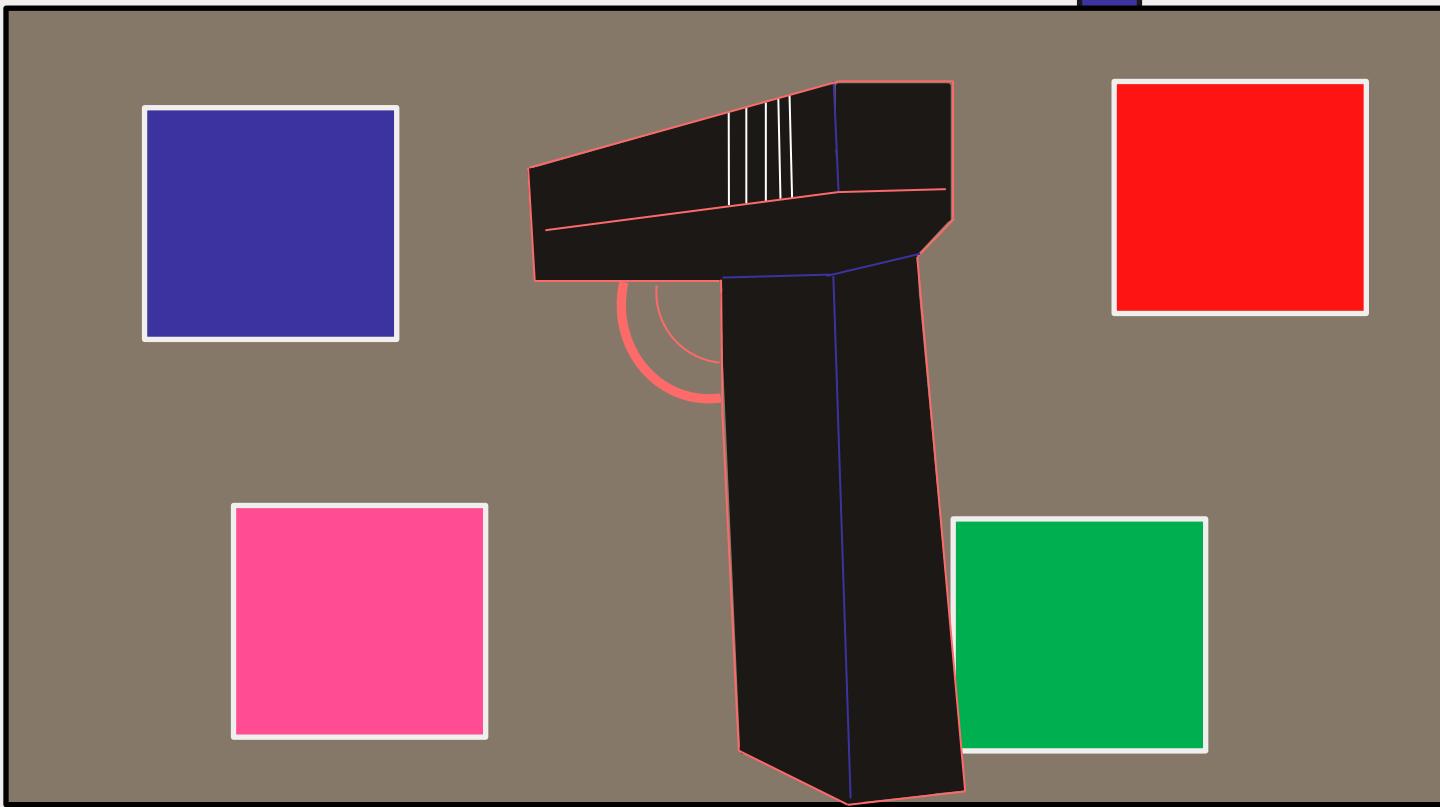
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



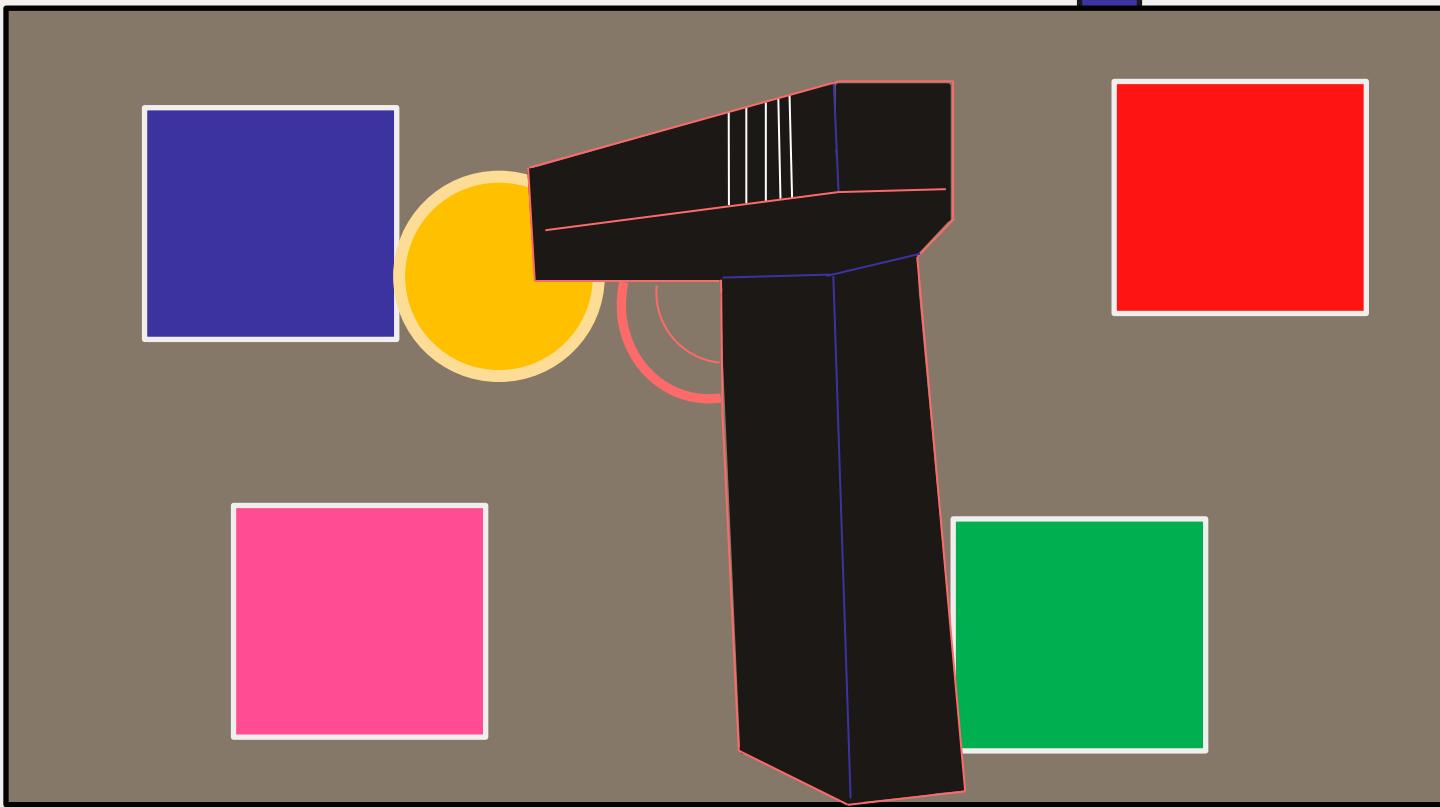
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



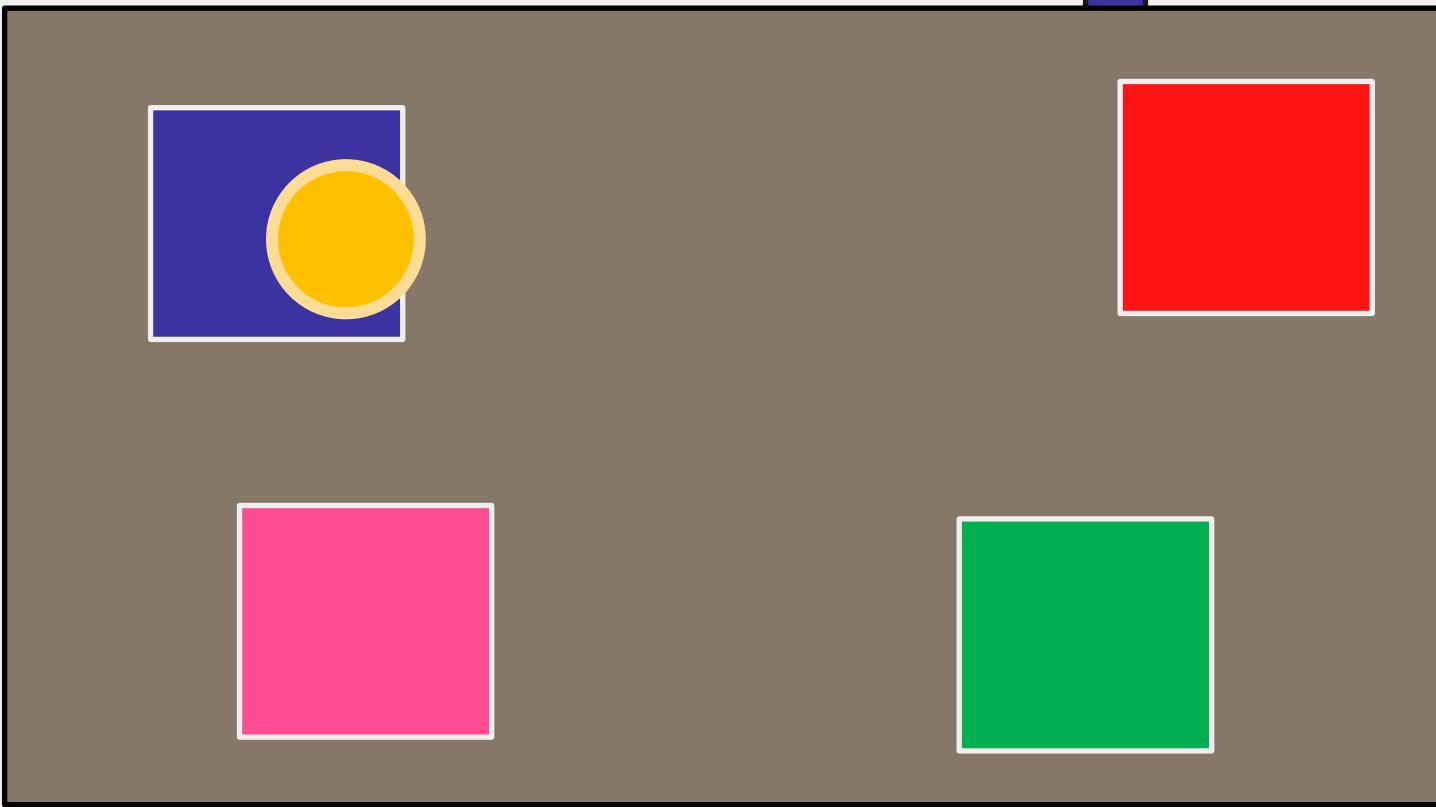
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



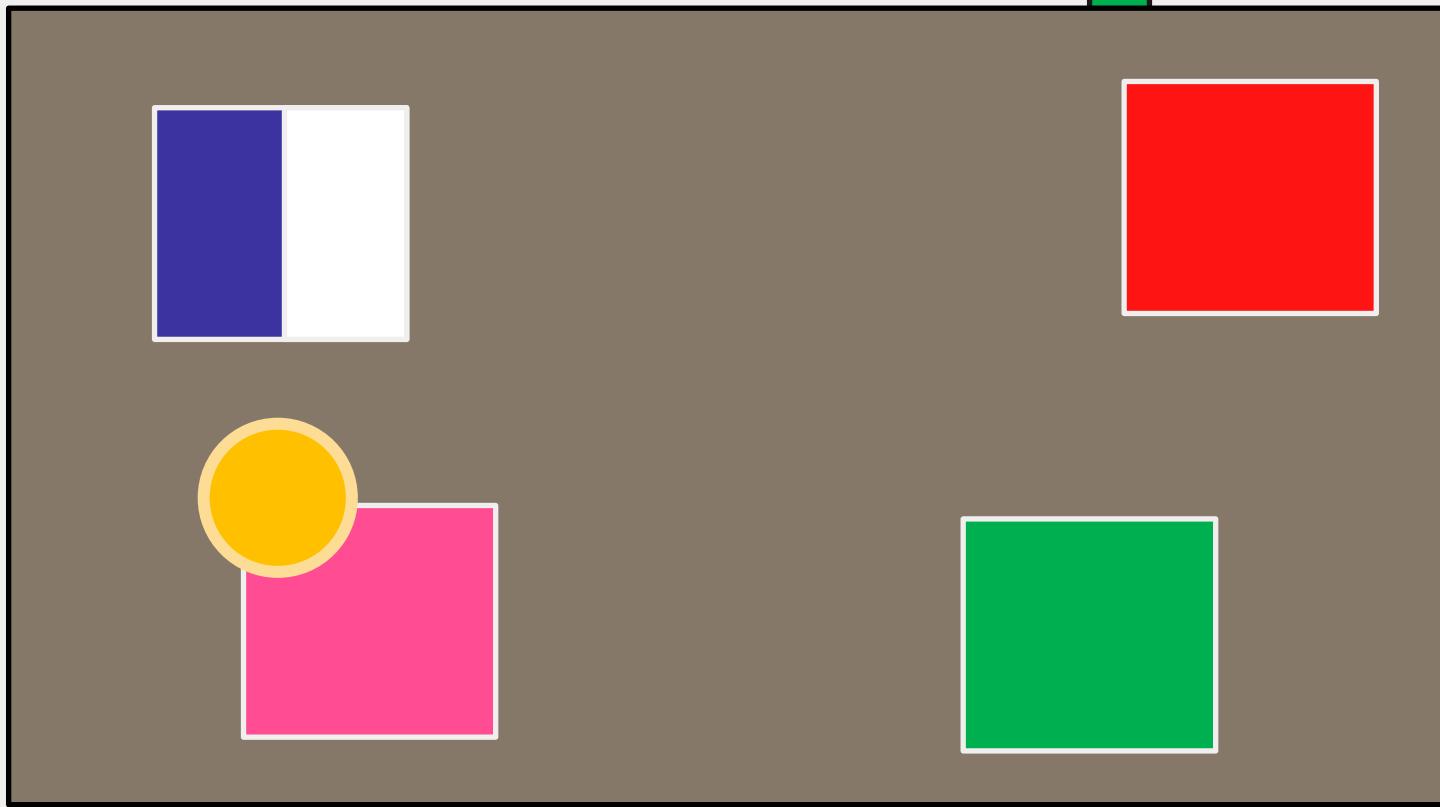
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



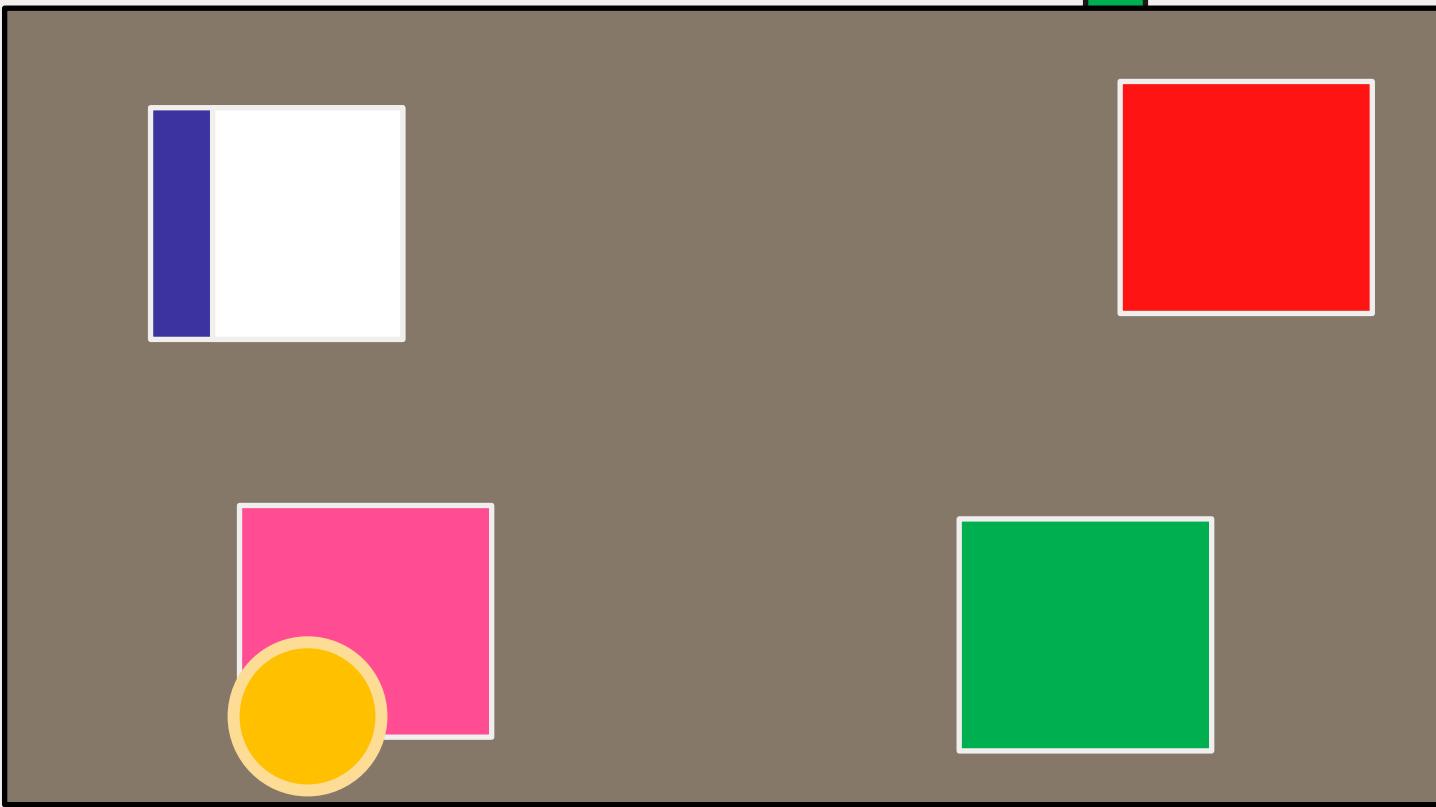
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



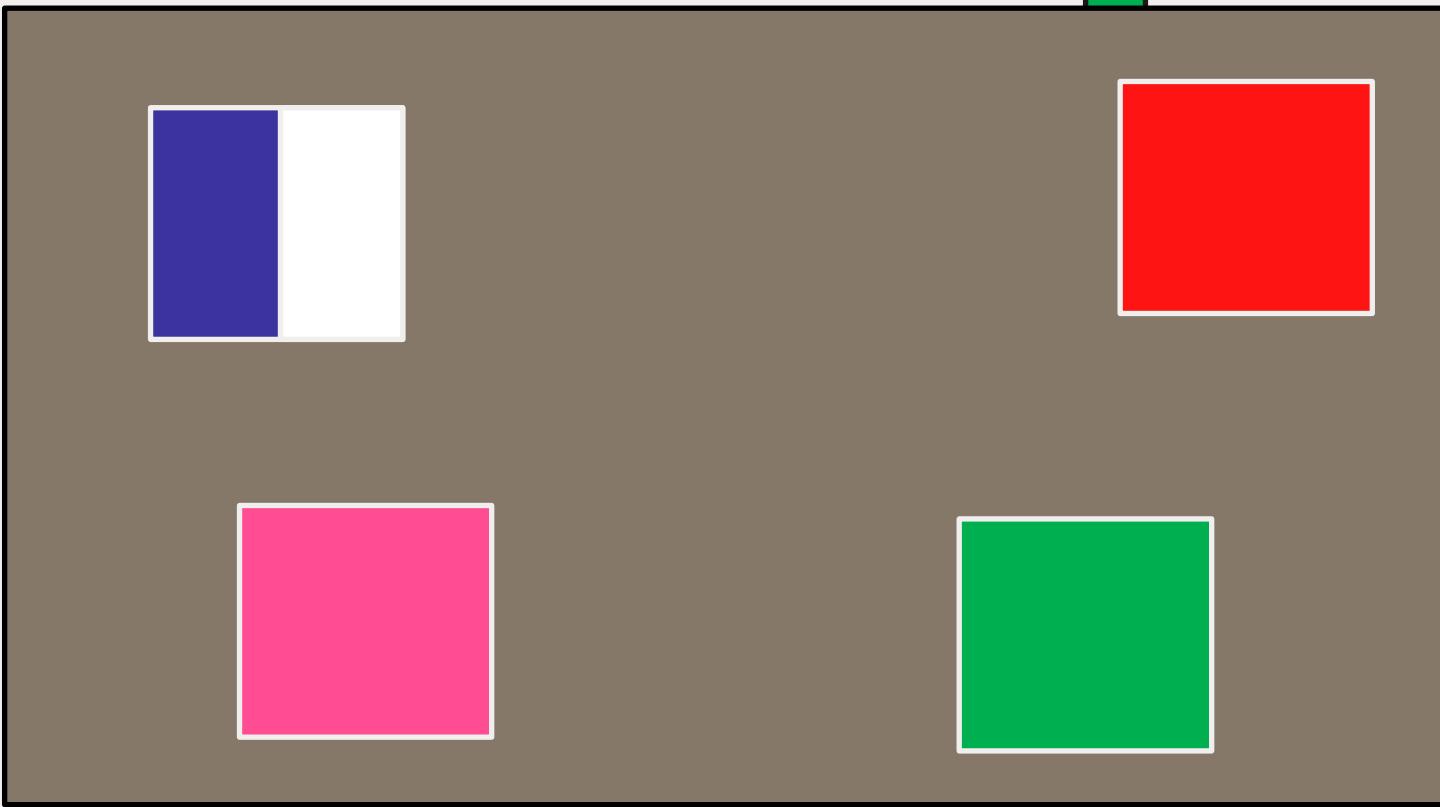
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



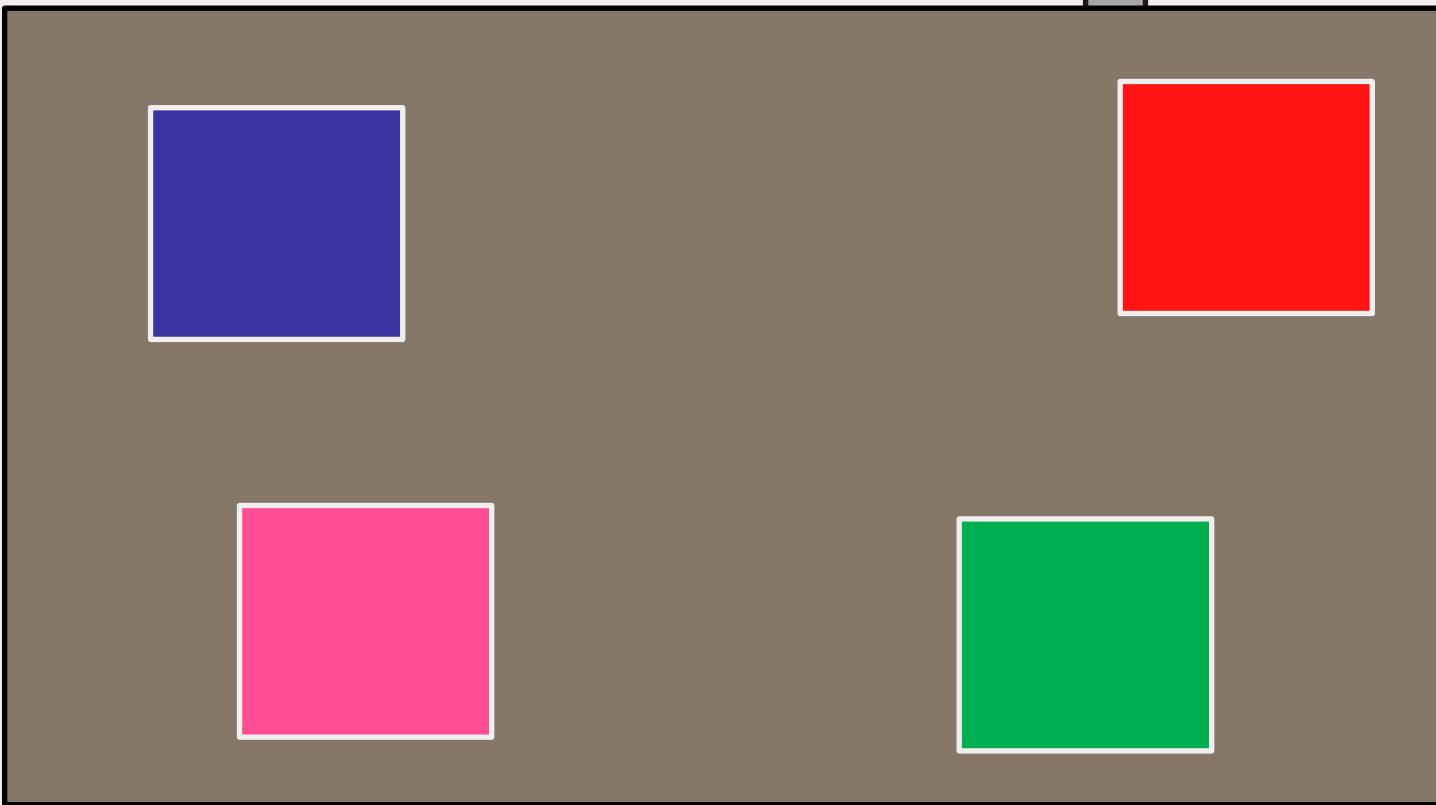
BUT**Première
cible****Grande
cible****Conclusion**

Projet final



BUT**Première
cible****Grande
cible****Conclusion**

Projet final



BUT

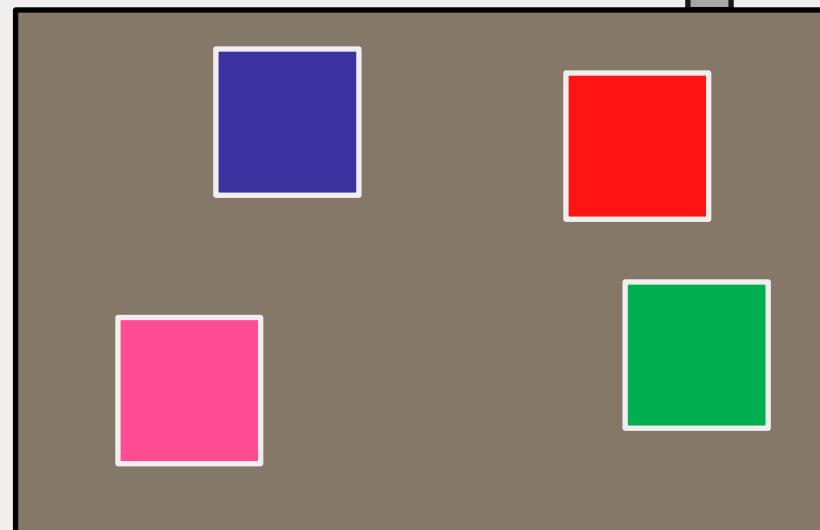
Première
cibleGrande
cible

Conclusion

ACHÈVEMENT DU PROJET

Création d'un jeu

Augmentant : -la compétitivité
-l'envie d'entraînement



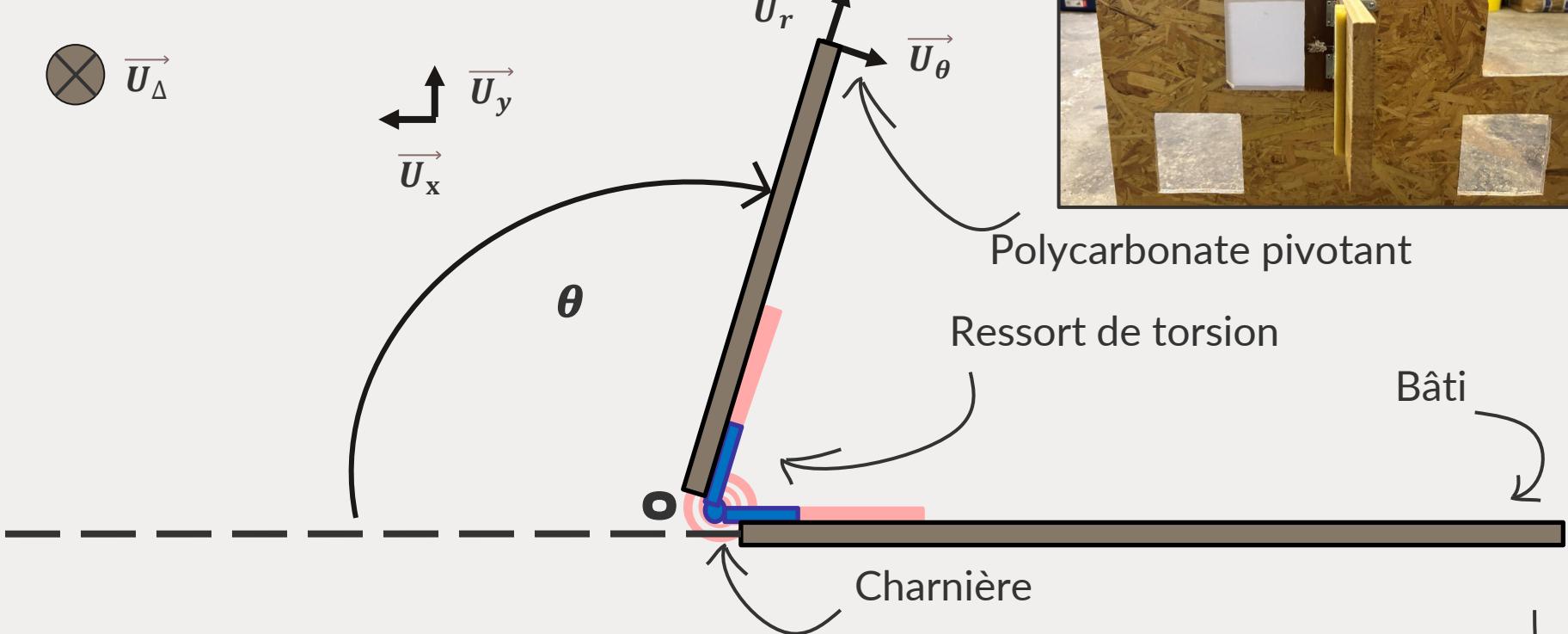
BUT**Première cible****Grande cible****Conclusion****02**

PREMIÈRE CIBLE

Partie présentée par mon camarade

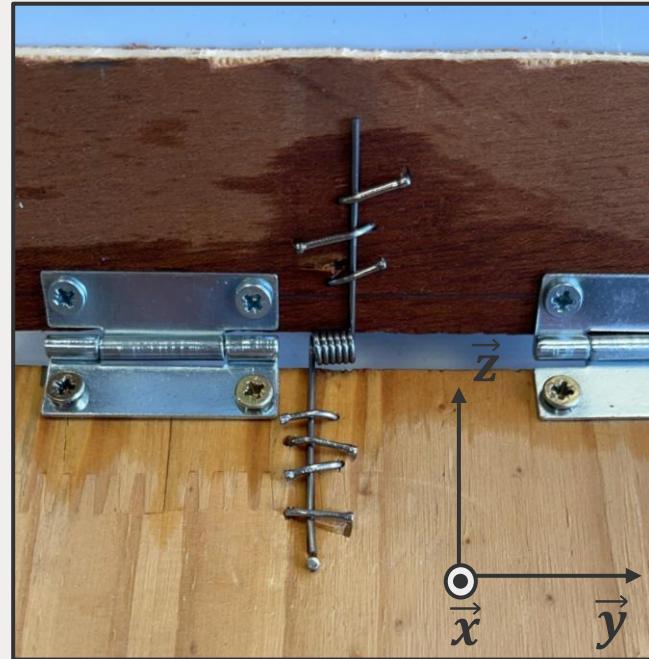
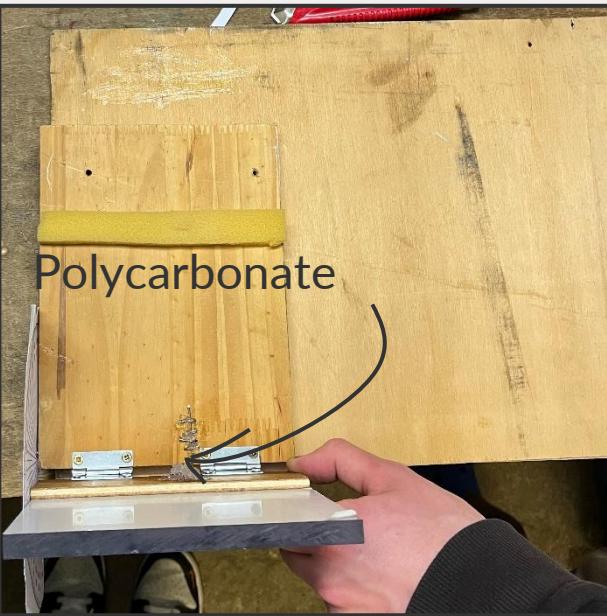
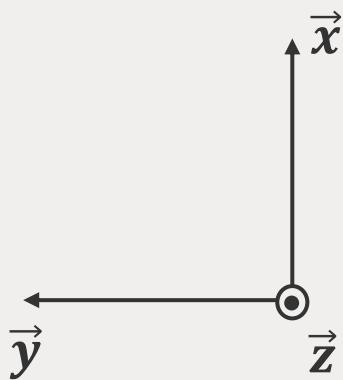


Notre prototype



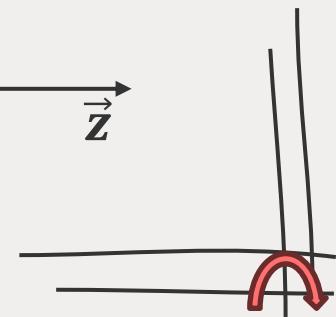
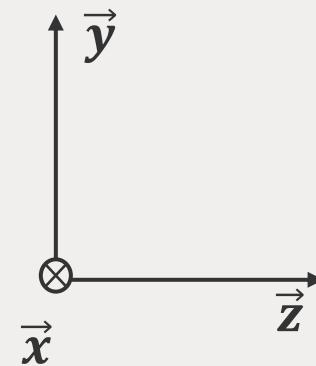
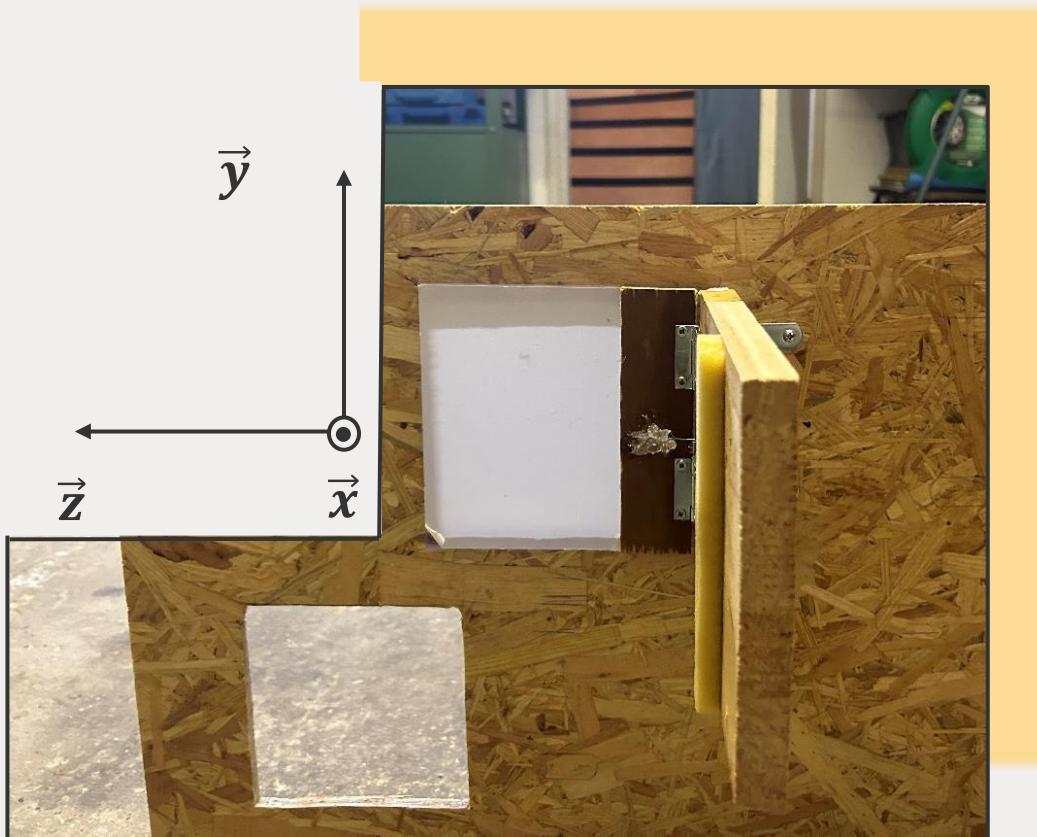
- Réalisation petites cibles indépendante de la réalisation de la grande

Notre prototype: fabrication



- Cible tirée : polycarbonate
- Fixation des ressorts avec des clous et de la colle
- Charnière lubrifiée

Notre prototype: fixation

 \vec{y} \vec{x} \vec{z} 

BUT

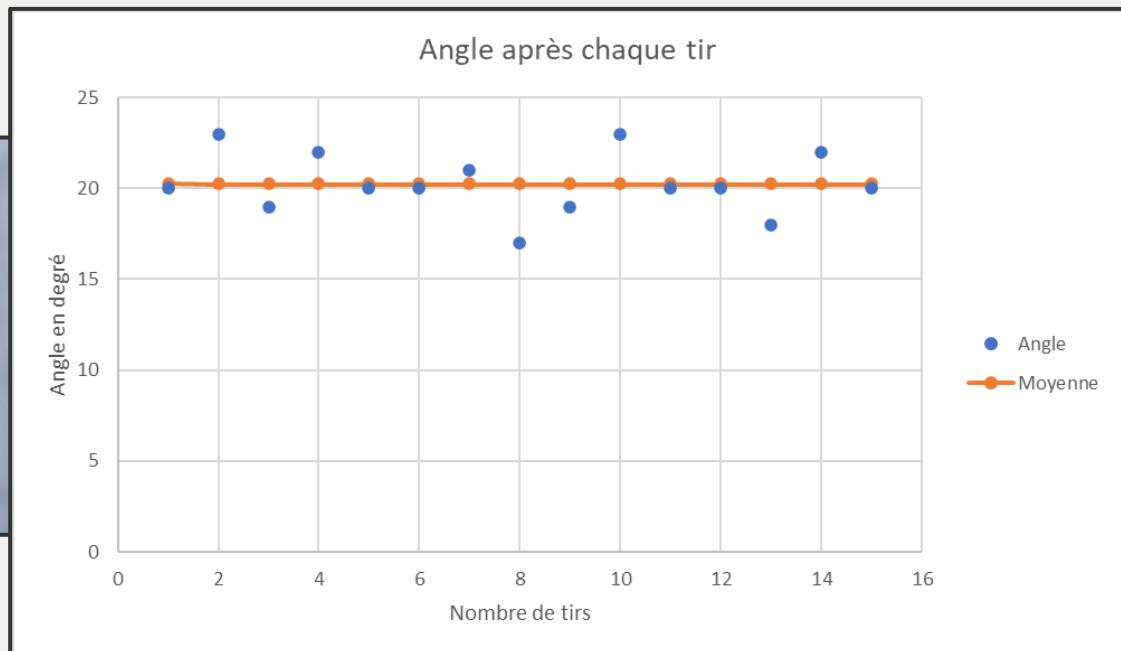
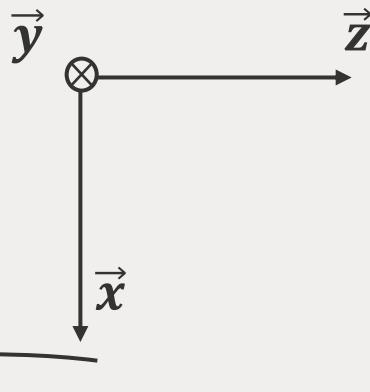
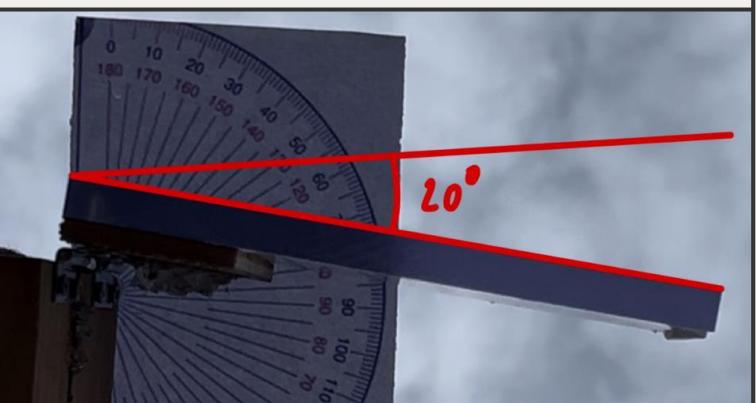
Première
cibleGrande
cible

Conclusion

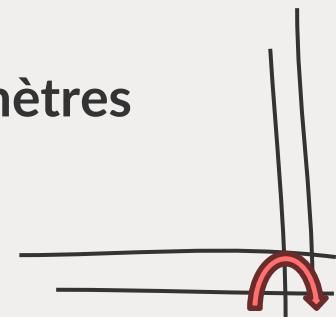
Notre prototype: fixation



Notre prototype: tests



- Distance du tir : 7 mètres
- Nombre de tirs : 15



BUT

Première
cibleGrande
cible

Conclusion

Résultats de la première cible

Exigence**Critère****Déplacement angulaire suffisant****Détection du mouvement par les capteurs****Retour à l'état initial****Ecart: nul****Oscillations****Minimum**

BUT**Première cible****Grande cible****Conclusion****03**

GRANDE CIBLE

Mon projet principal

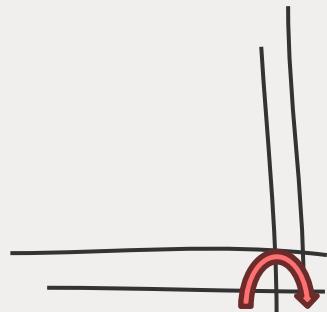
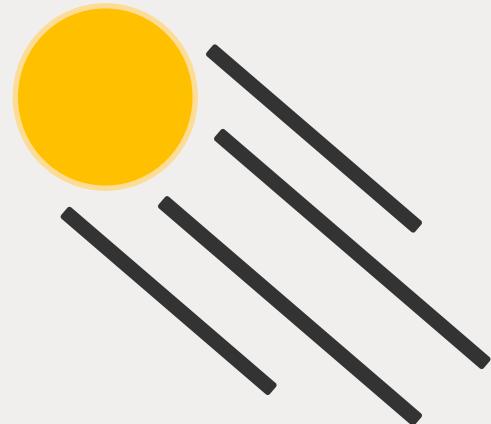




Comment automatiser la grande cible ?

Plusieurs critères:

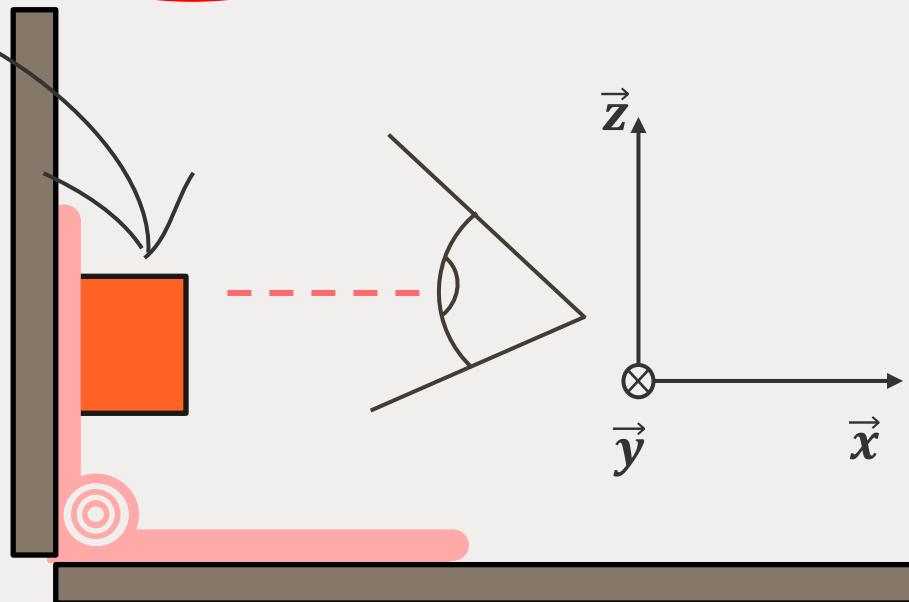
- Déetecter qu'une petite cible est touchée dans un temps court
- Activer des algorithmes (allumer une led)



Le capteur

Les différents capteurs possibles

- Cellule photoélectrique
- Capteur ultra-son



Sc : Semageek

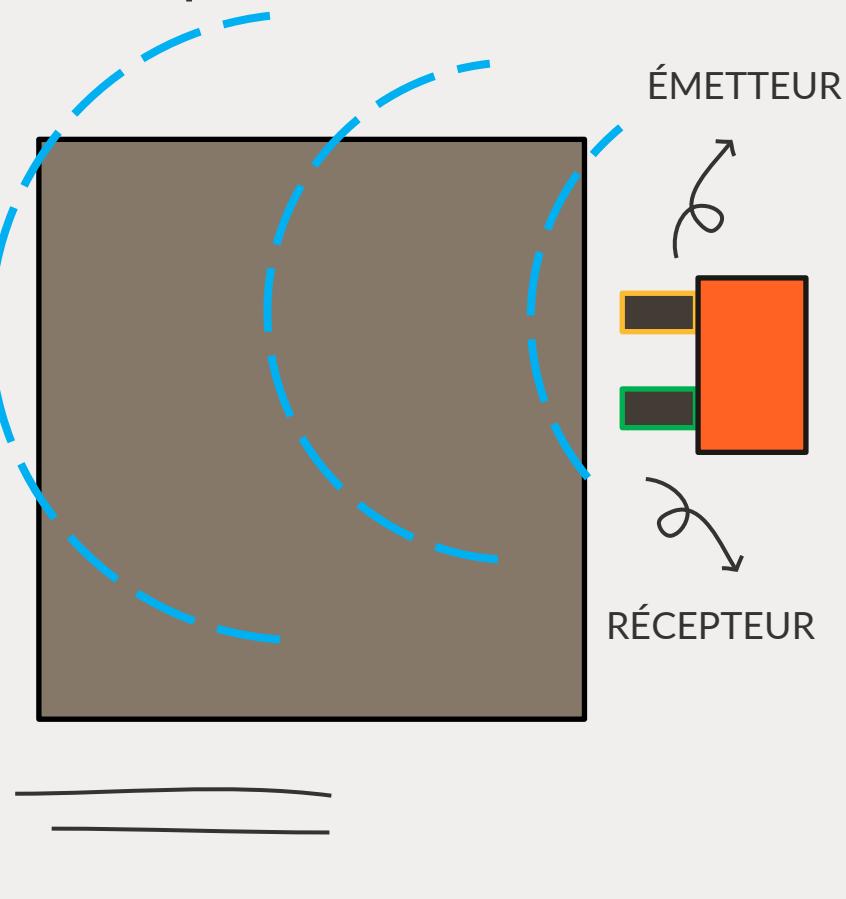


Sc : Semageek

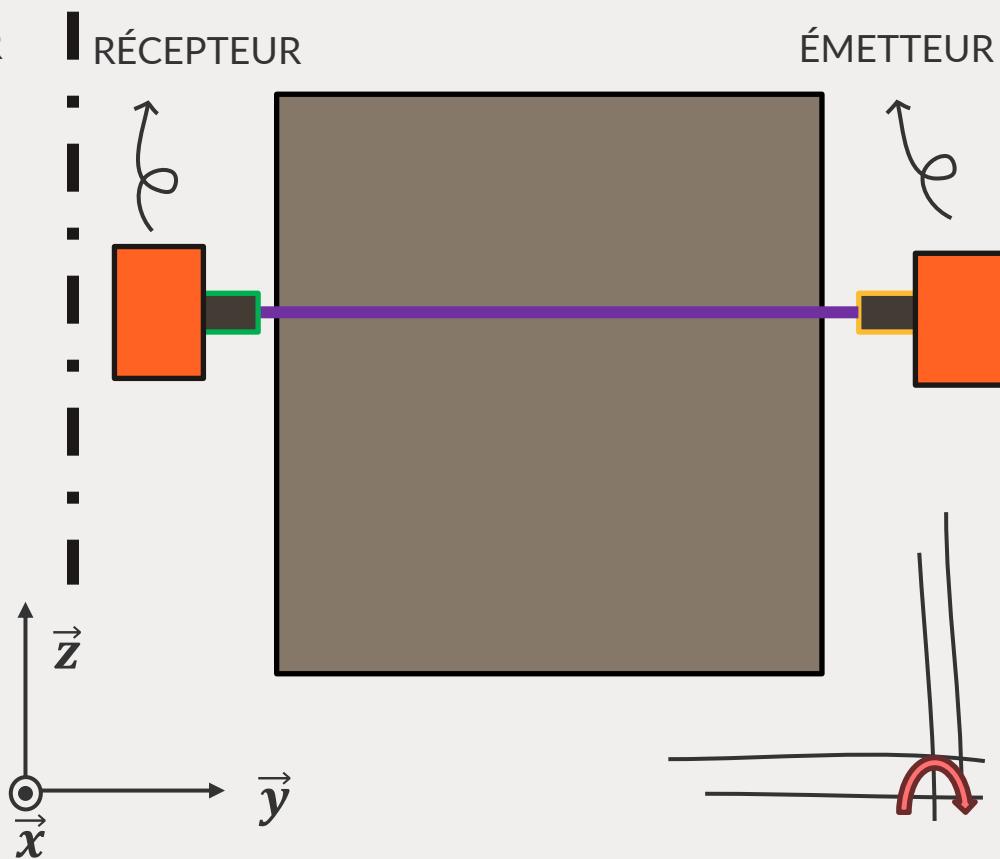


Le capteur : fonctionnement

- Capteur ultra-son



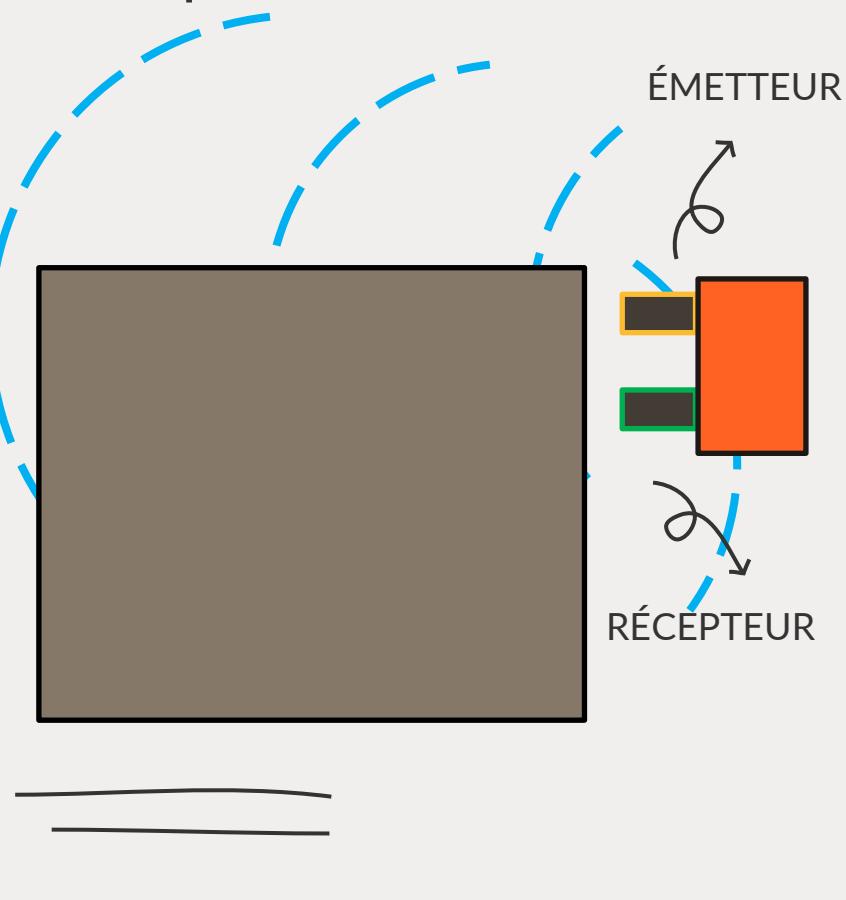
- Cellule photoélectrique



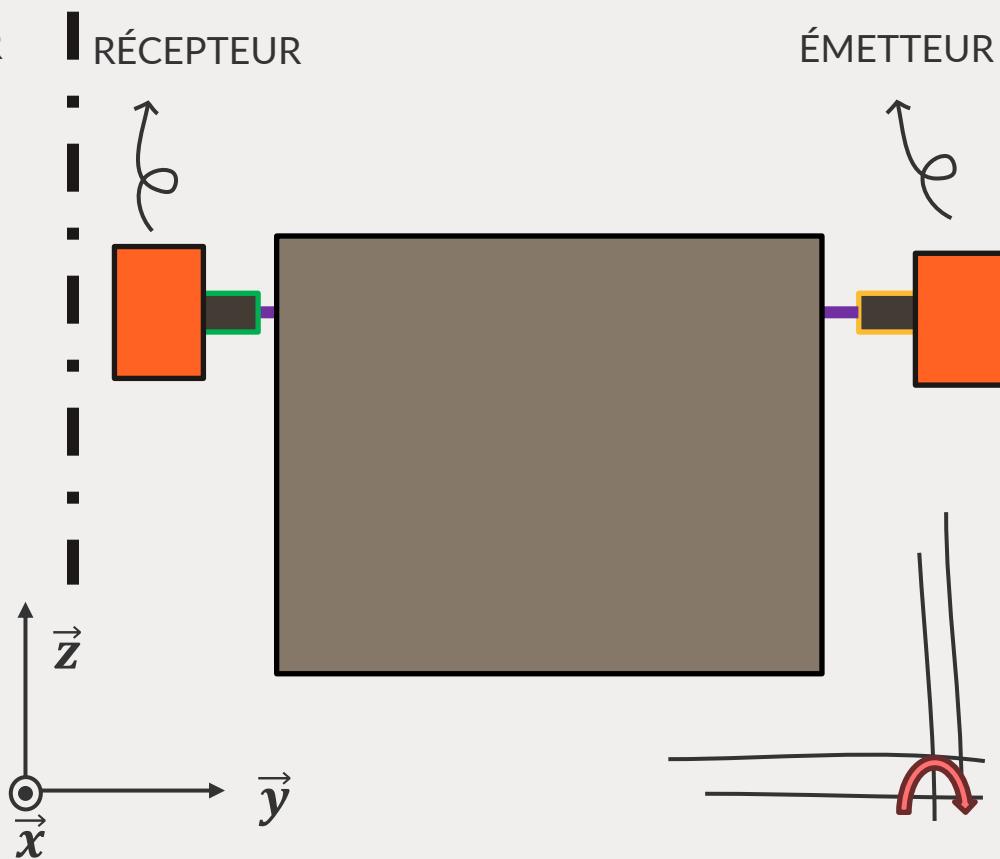


Le capteur : fonctionnement

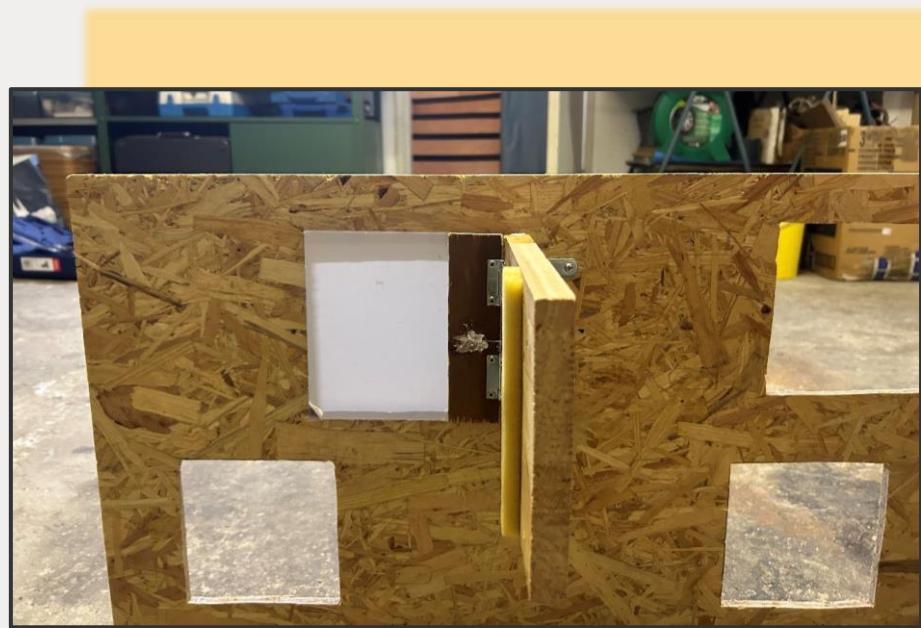
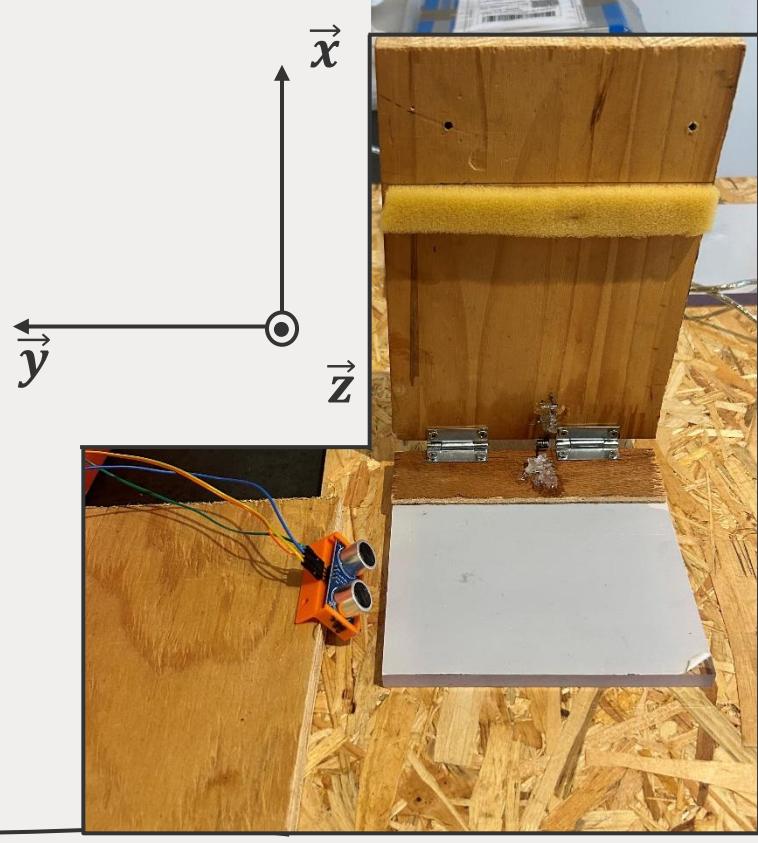
- Capteur ultra-son



- Cellule photoélectrique



Le capteur : fonctionnement



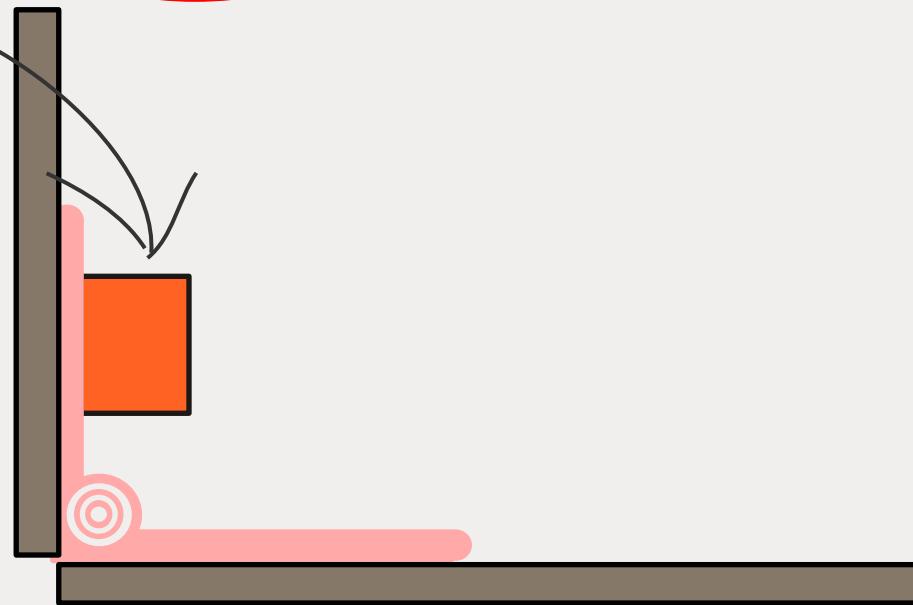
Le capteur

Choix : capteur ultra-son

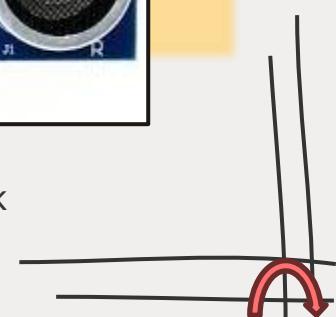
- Plus petit encombrement
- Capteur créé pour Arduino
- Distance entre le capteur et la cible est faible (donc temps de parcours aussi faible)

$$D \approx 1 \text{ cm}$$

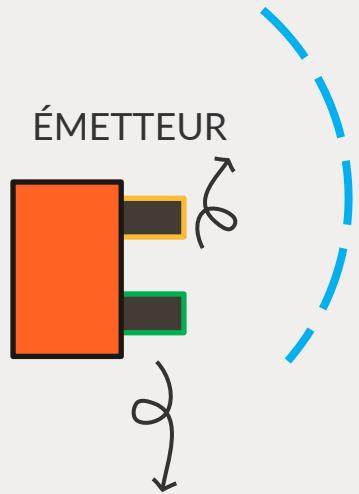
$$\Delta t = 2t = \frac{D}{v} \approx 6,5 \times 10^{-4} \text{ s}$$



Sc : Semageek

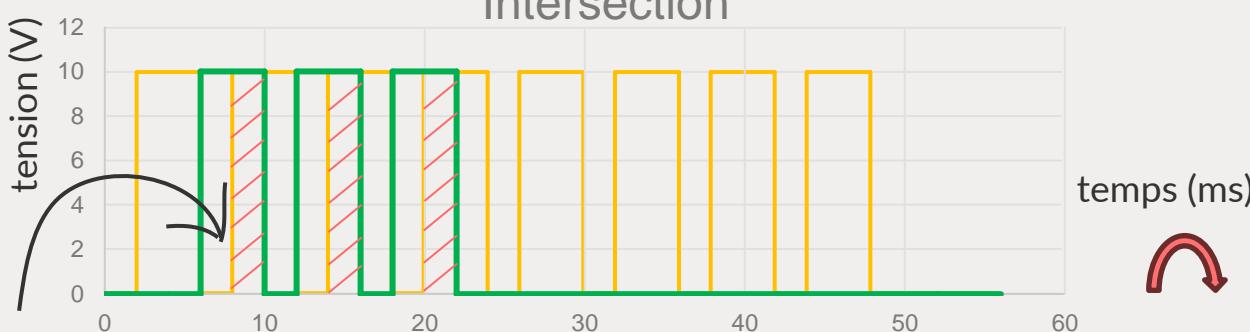
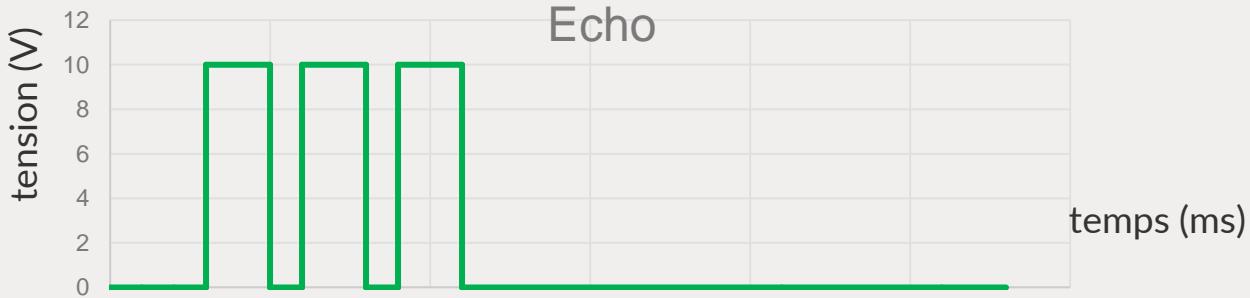
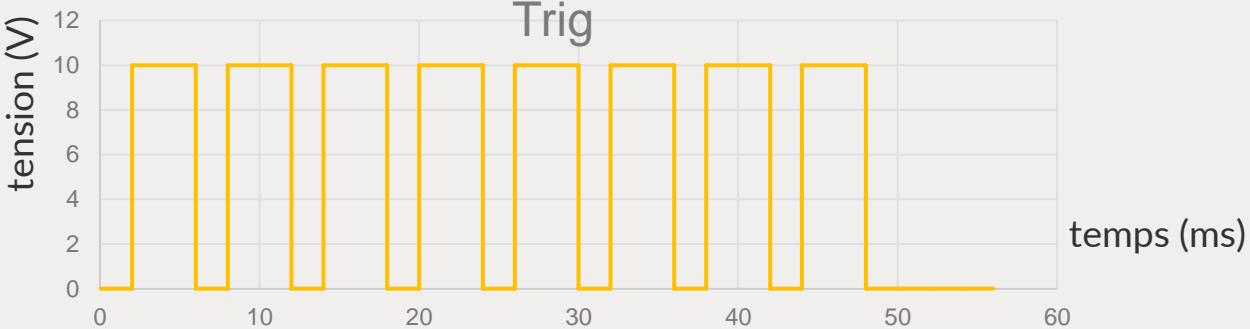


Capteur ultra-son : le principe



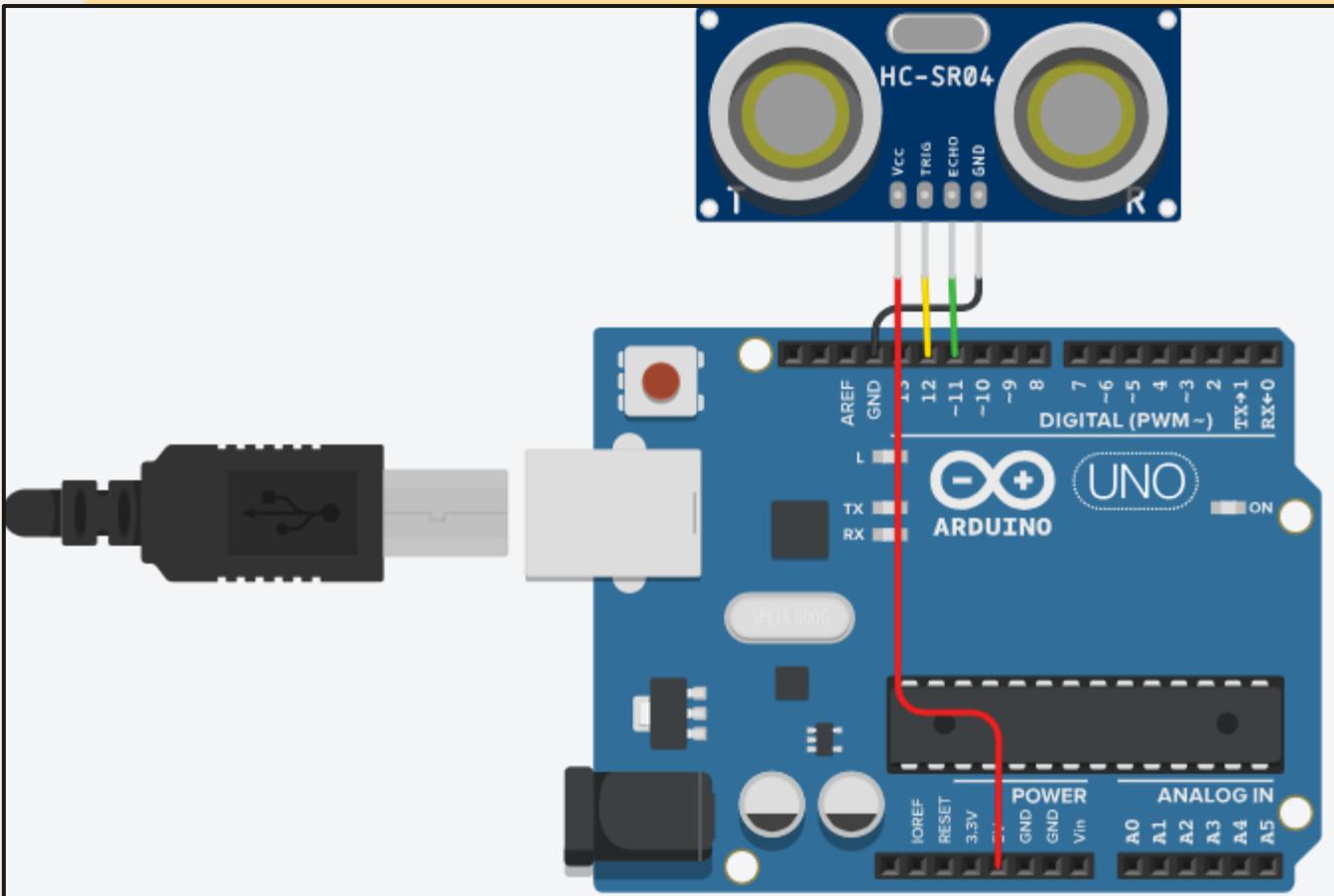
Fronts montants
et descendants

Moments de détections



Capteur ultra-son : le branchement

- Masse
- Alimentation
- Trig
- Echo

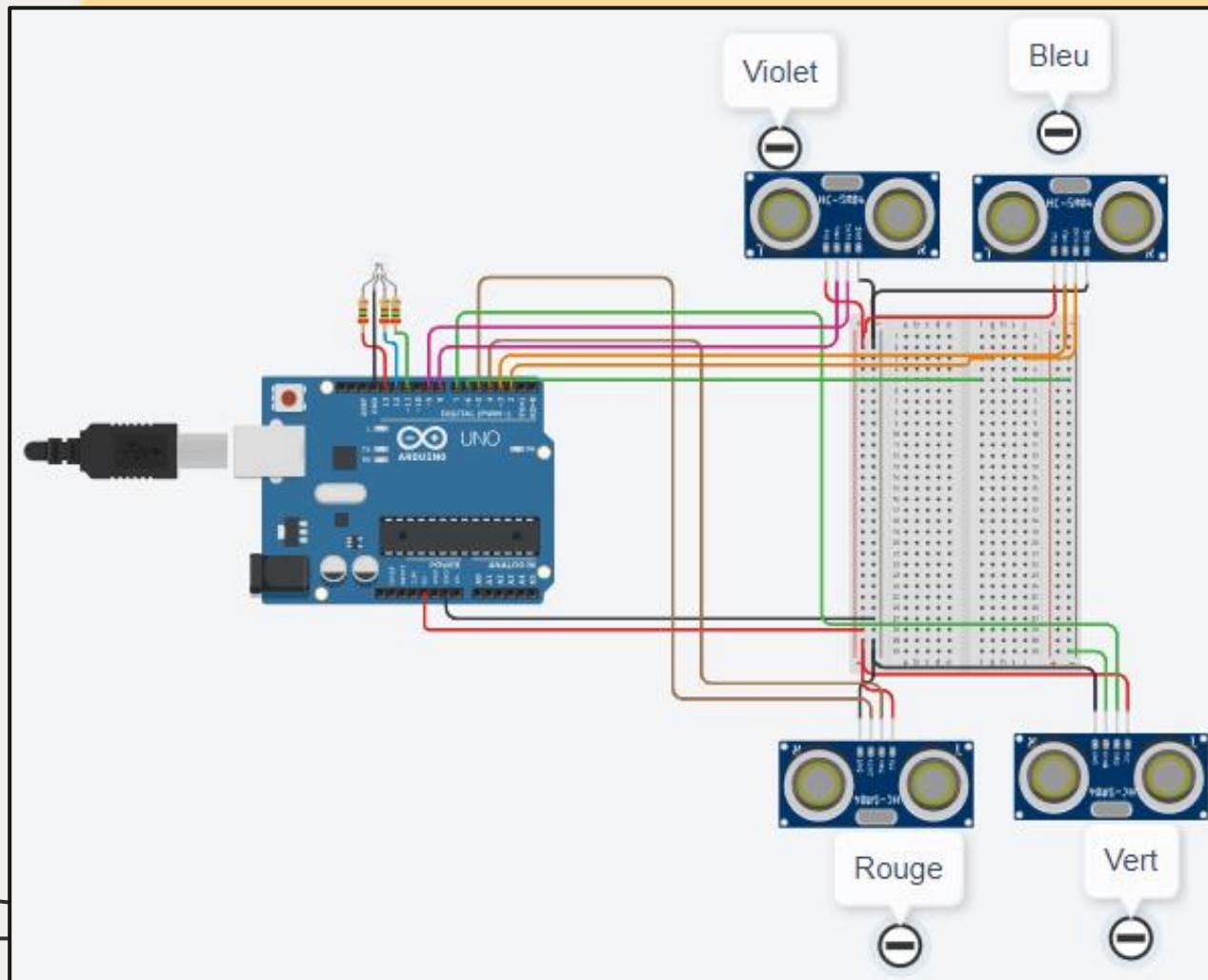


BUT

Première
cibleGrande
cible

Conclusion

Les branchements : initialisation



BUT

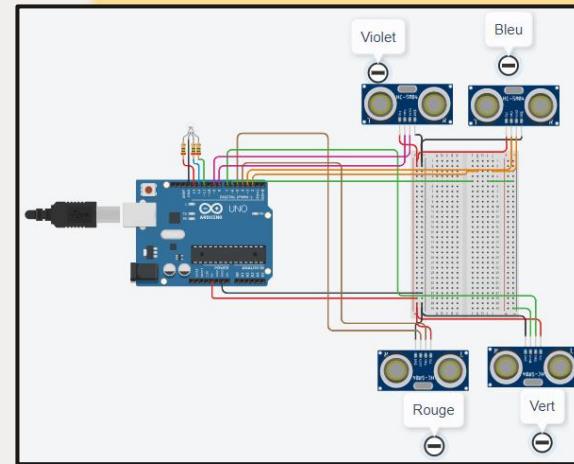
Première
cibleGrande
cible

Conclusion

Le développement : initialisation

Vérification et mise en place du programme

Définition des broches des capteurs et de la LED RGB



BUT

Première
cibleGrande
cible

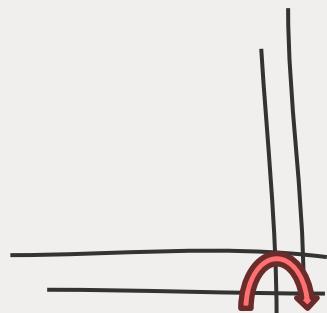
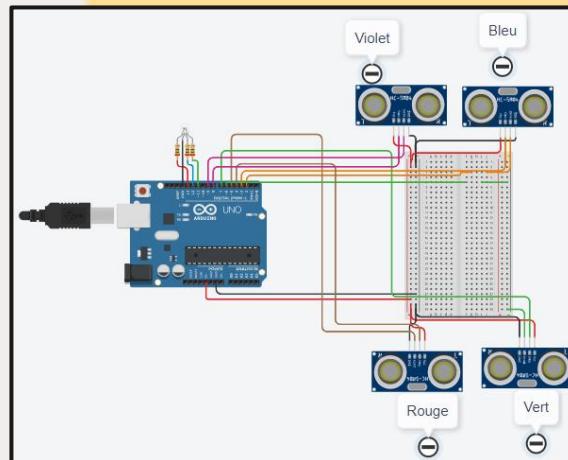
Conclusion

Le développement : initialisation

Vérification et mise en place du programme

Définition des broches des capteurs et de la LED RGB

→
Impulsion d'onde pour chaque capteur



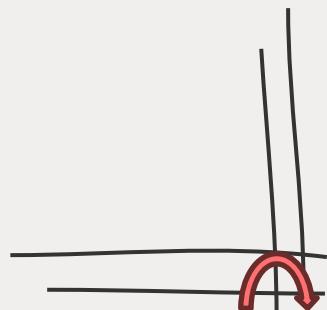
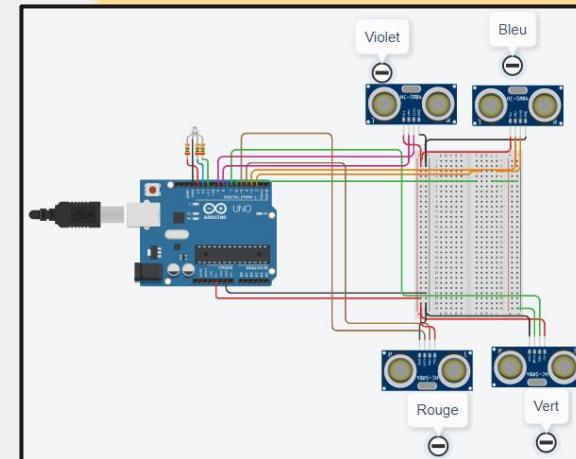
Le développement : initialisation

Vérification et mise en place du programme

Définition des broches des capteurs et de la LED RGB

Impulsion d'onde pour chaque capteur

Détection de la cible



Le développement : initialisation

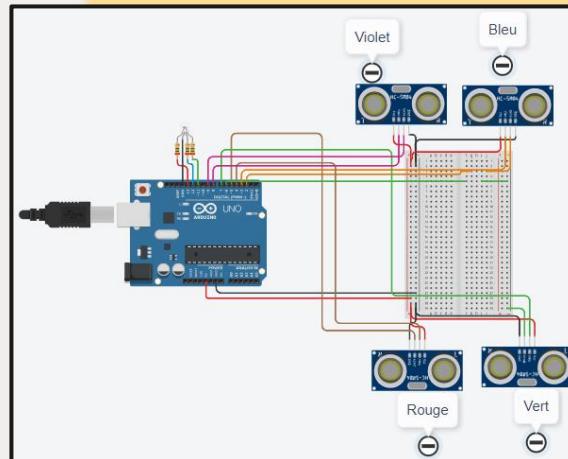
Vérification et mise en place du programme

Définition des broches des capteurs et de la LED RGB

Impulsion d'onde pour chaque capteur

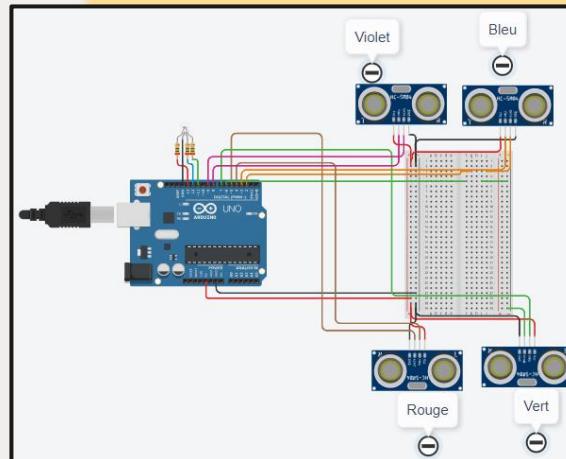
Détection de la cible

Activation de la couleur de LED RGB associée



Le développement : initialisation

Vérification et mise en place du programme



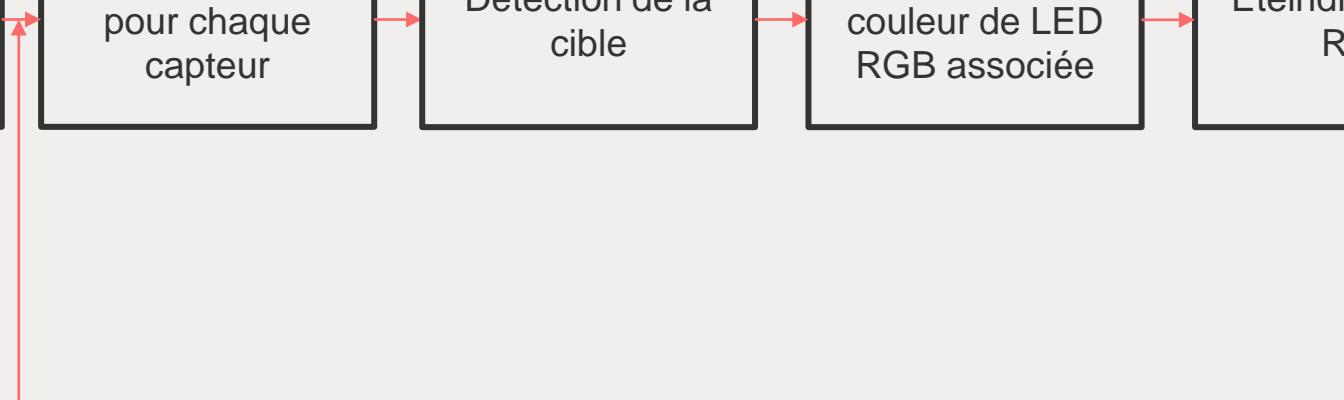
Définition des broches des capteurs et de la LED RGB

Impulsion d'onde pour chaque capteur

Détection de la cible

Activation de la couleur de LED RGB associée

Eteindre la LED RGB



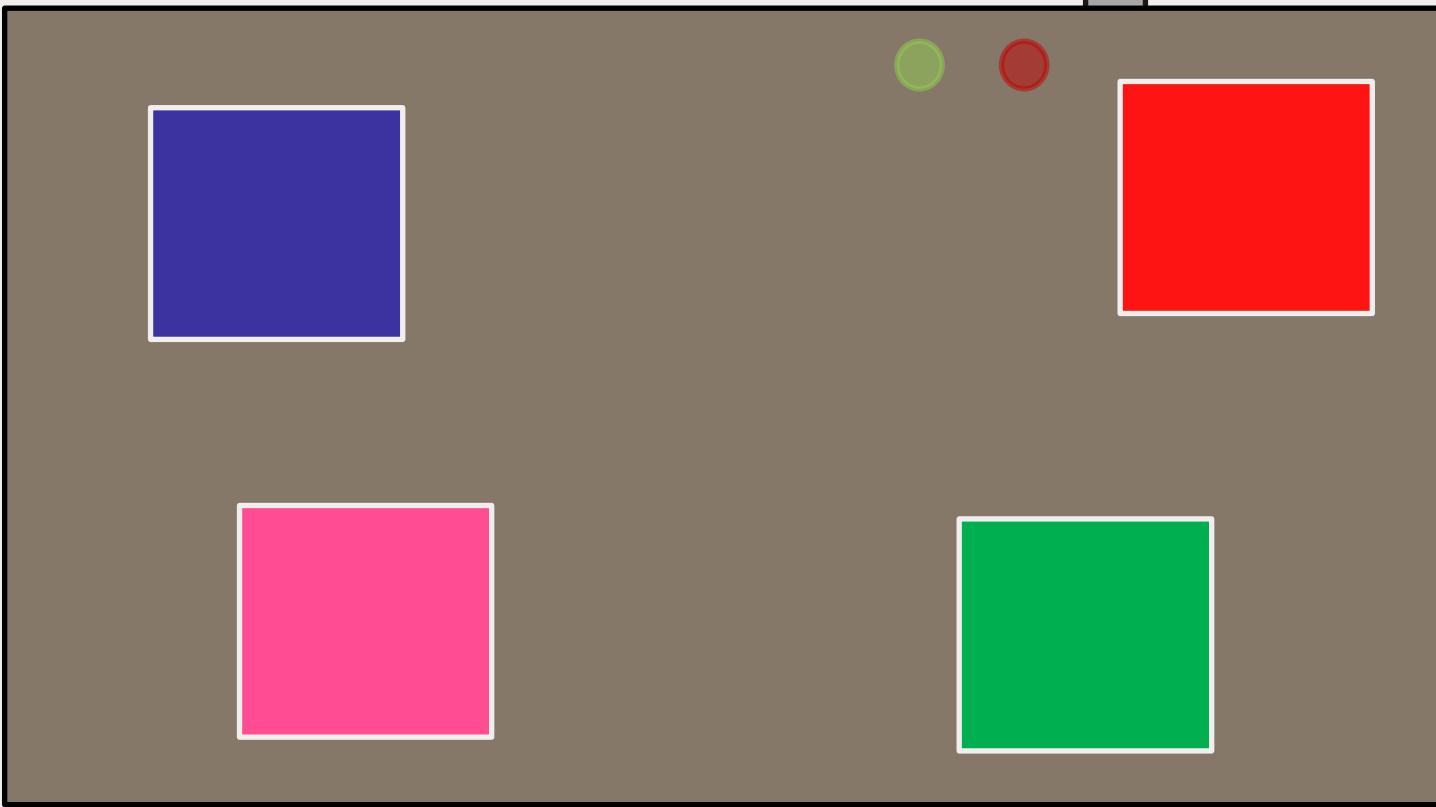
BUT

Première
cible

Grande
cible

Conclusion

Projet final



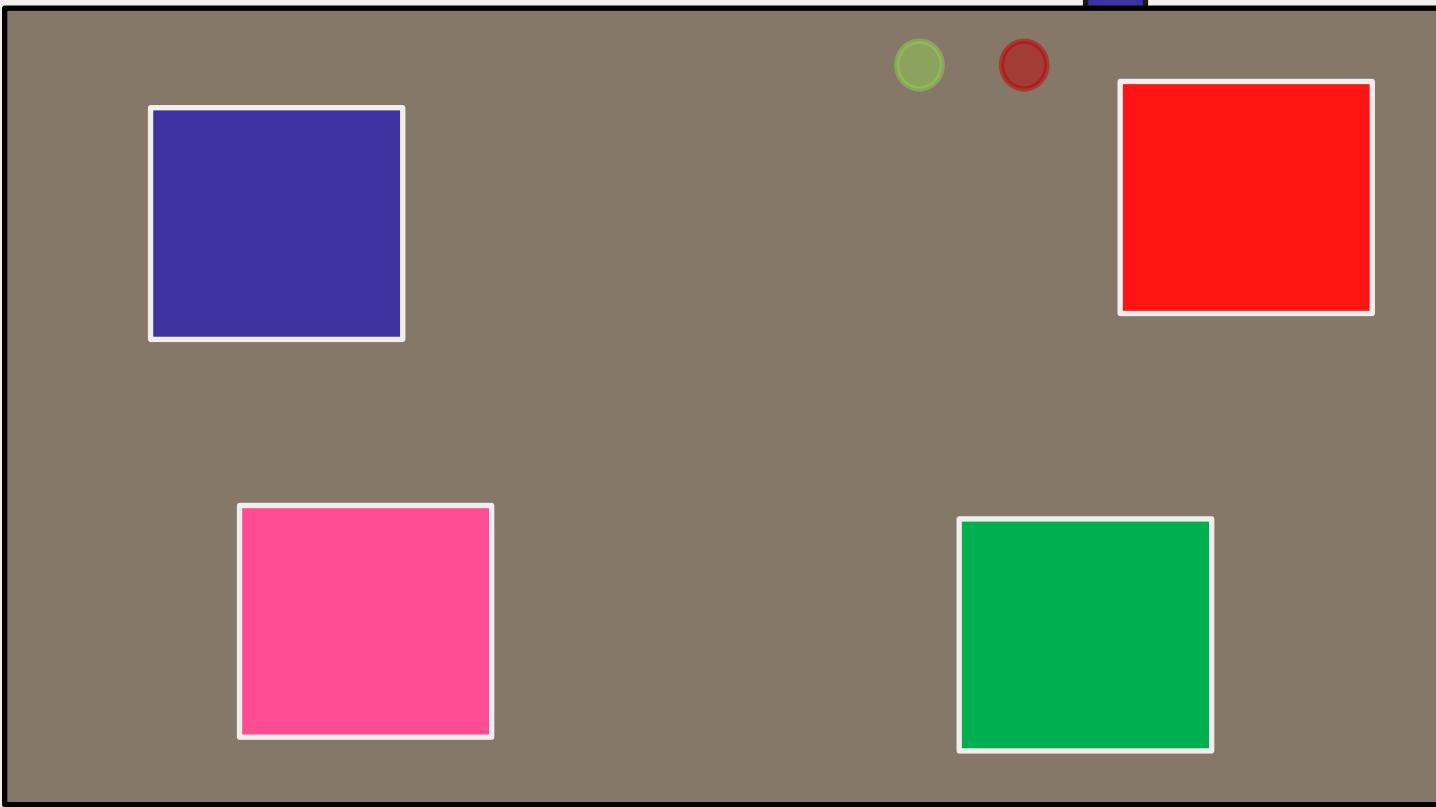
BUT

Première
cible

Grande
cible

Conclusion

Projet final

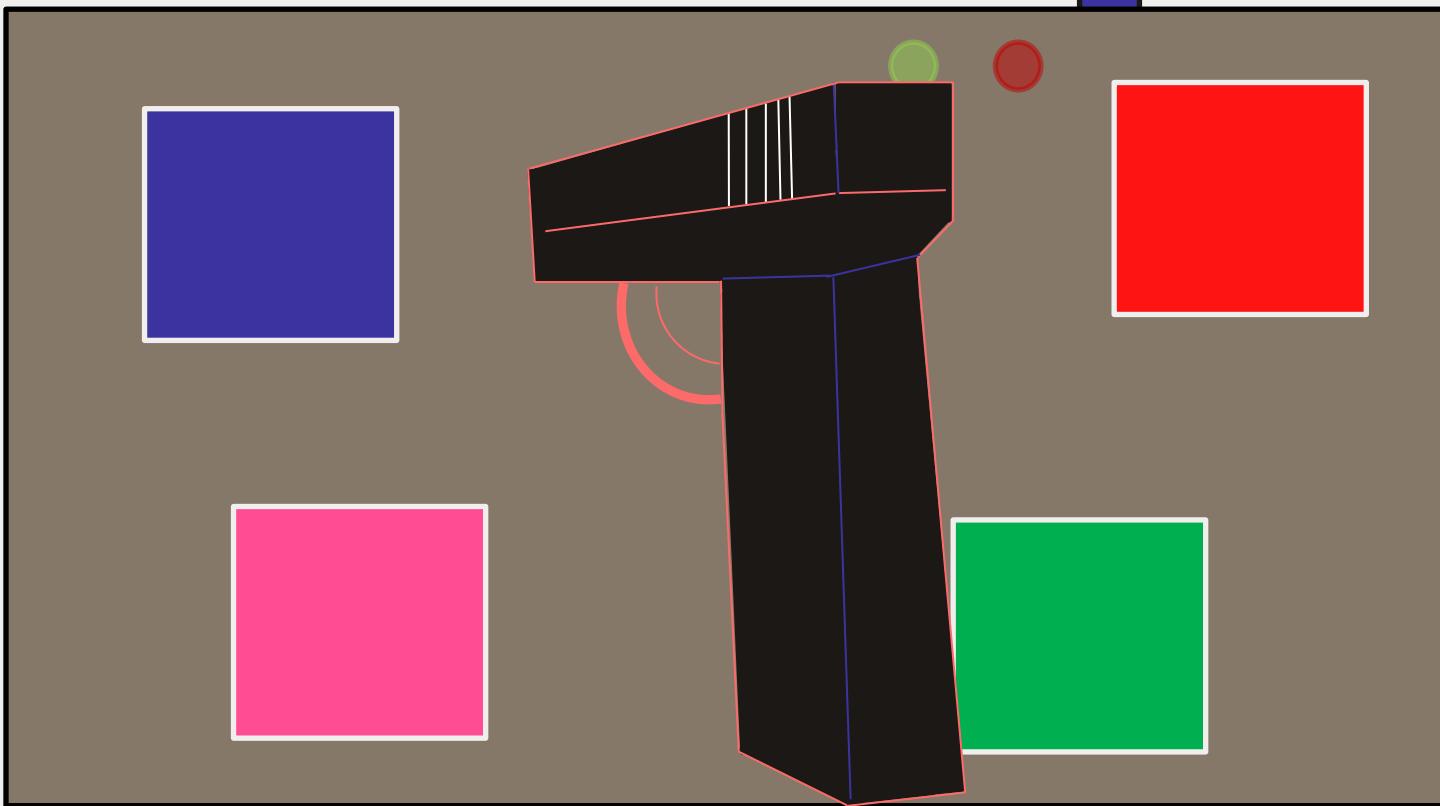


BUT

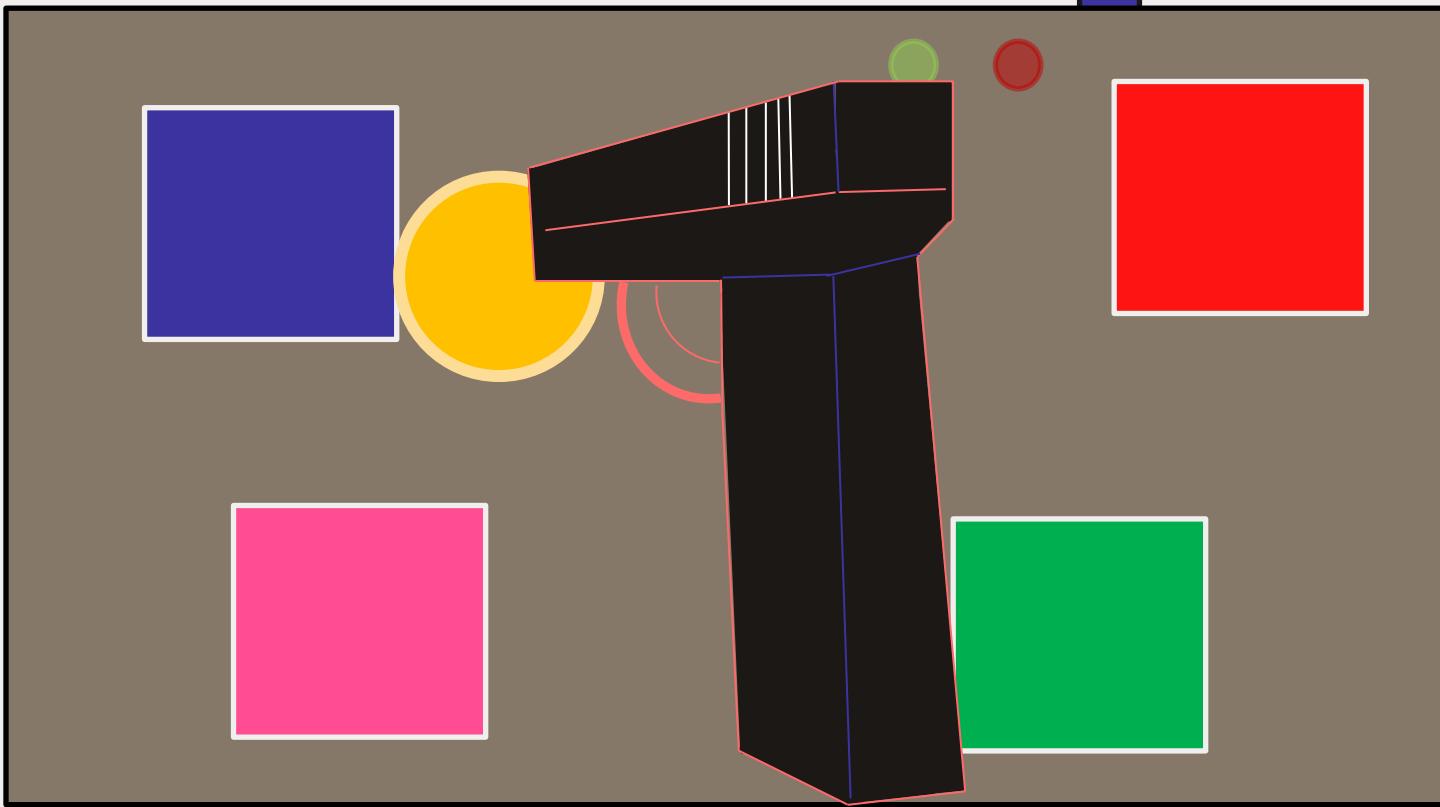
Première
cibleGrande
cible

Conclusion

Projet final



Projet final



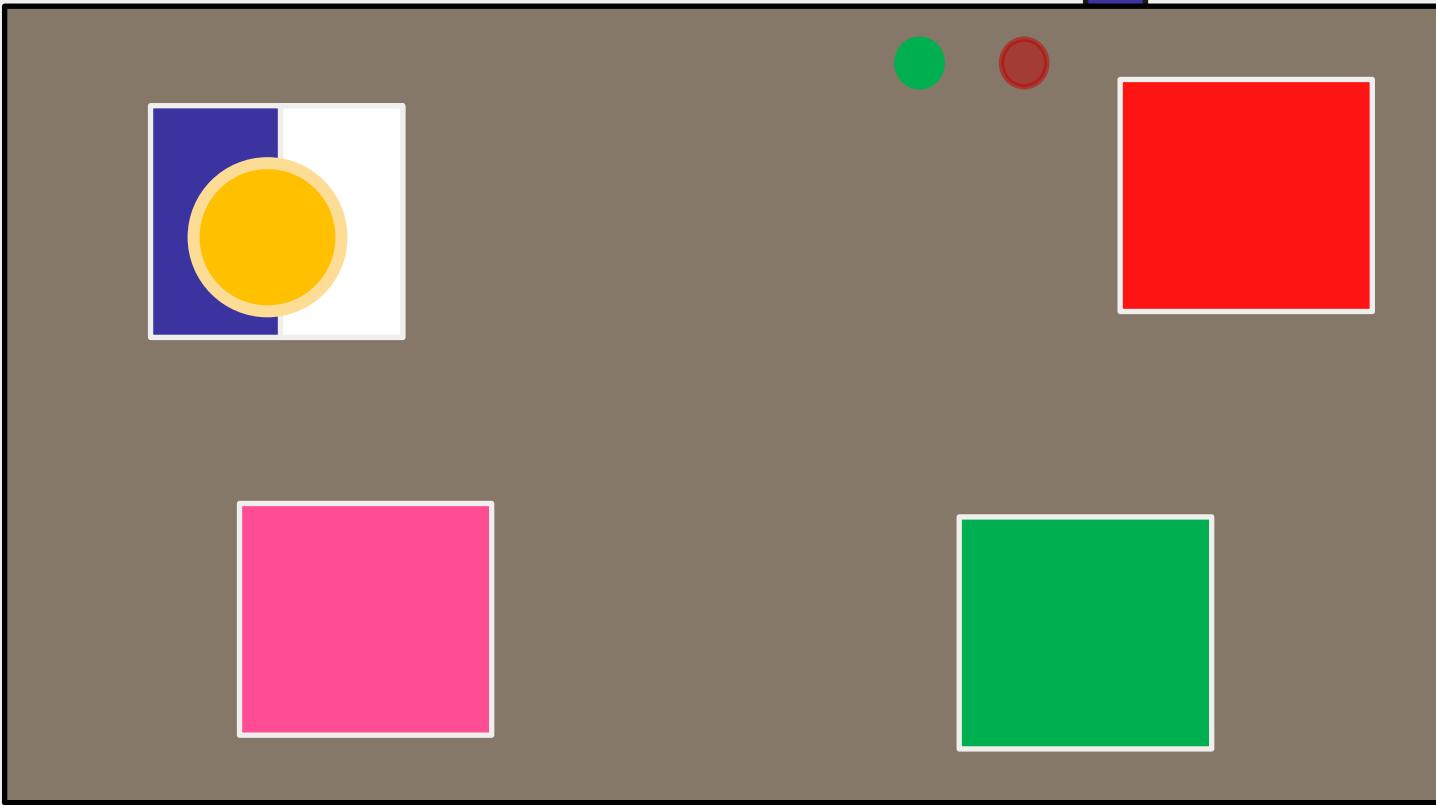
BUT

Première
cible

Grande
cible

Conclusion

Projet final

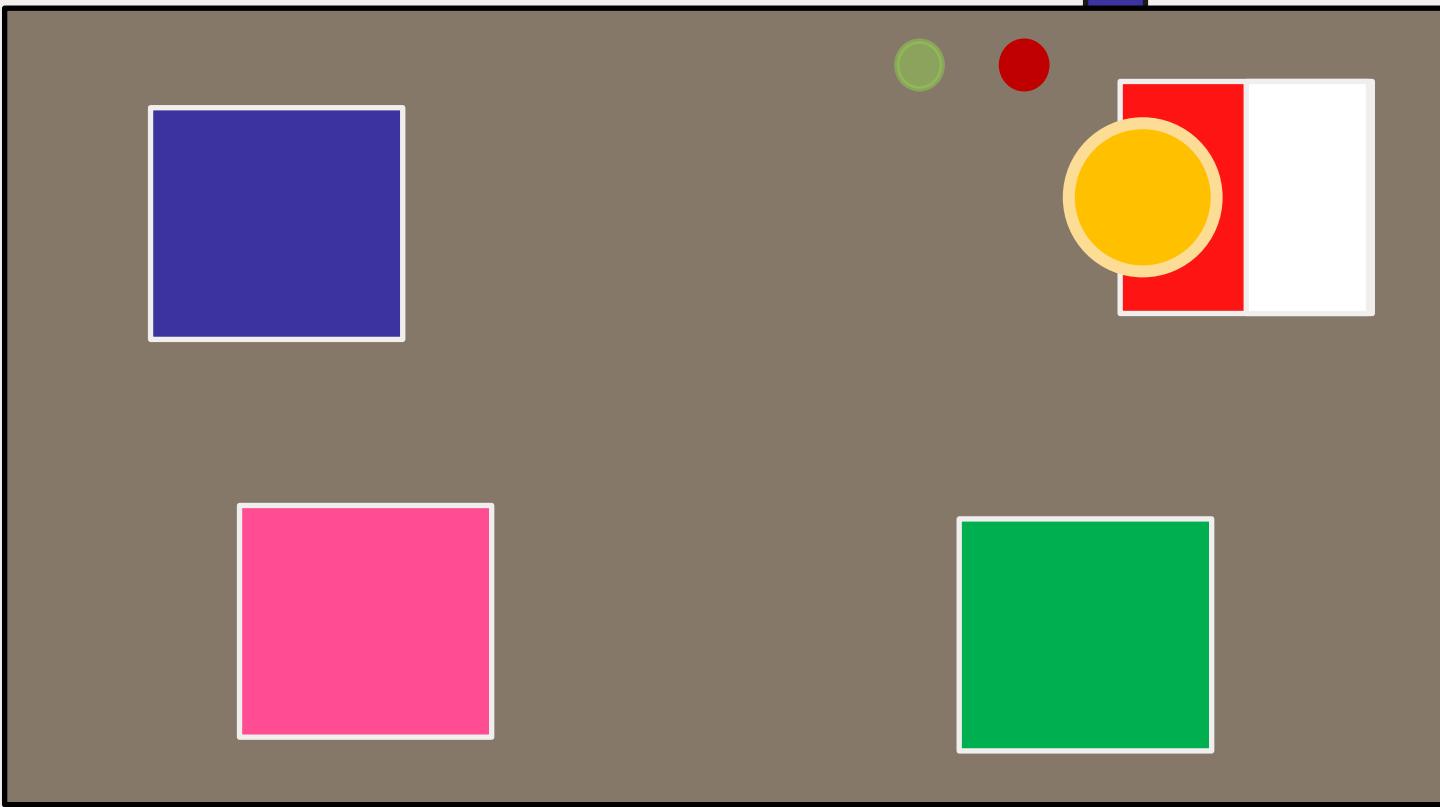


BUT

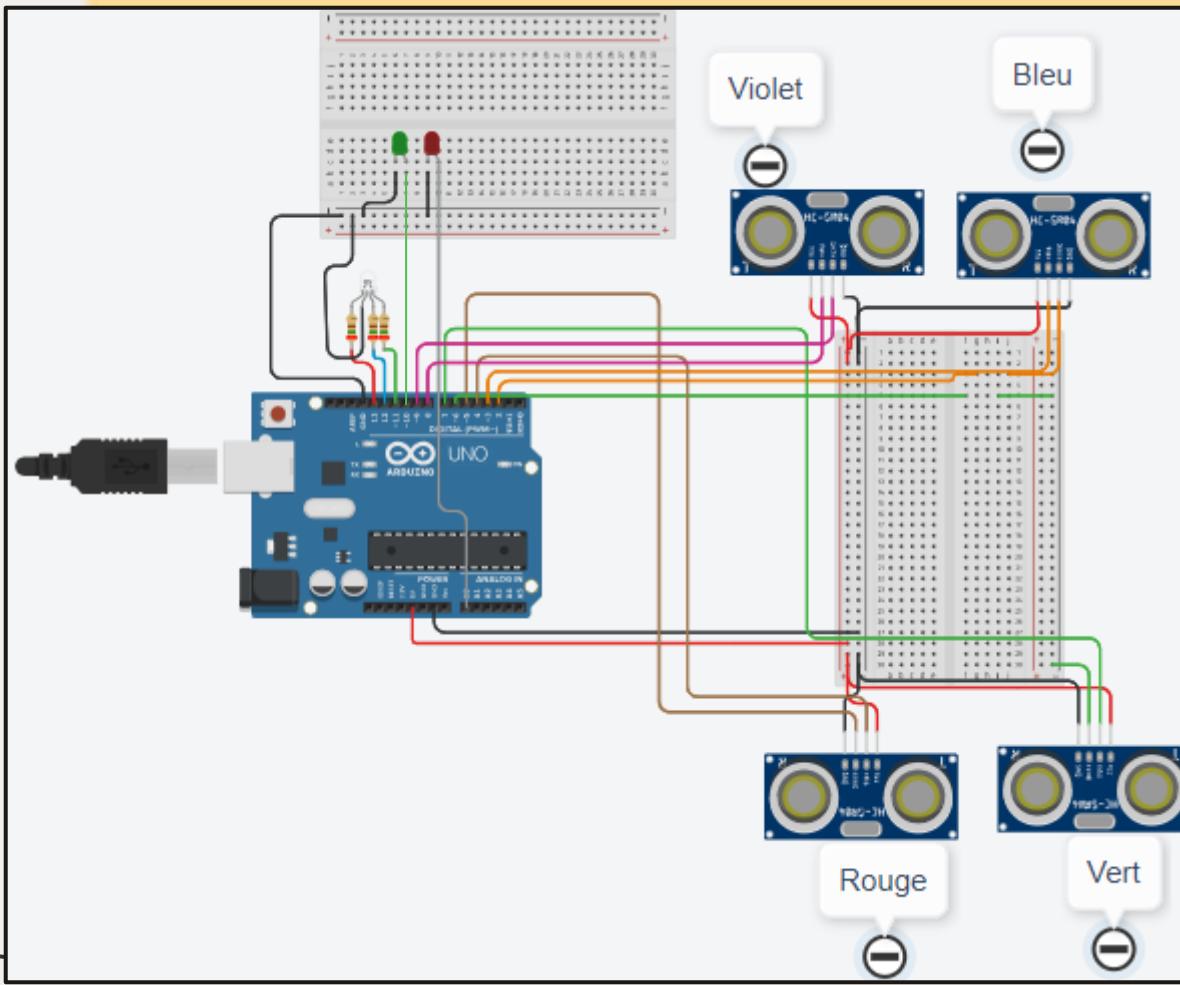
Première
cibleGrande
cible

Conclusion

Projet final



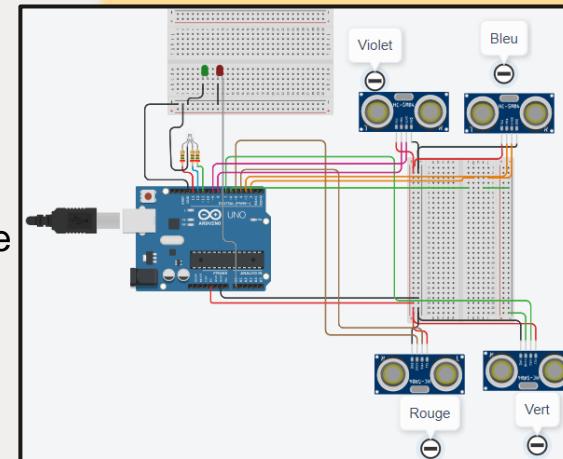
Les branchements : projet final



Le développement : projet final

Définition des broches des capteurs et des LEDs

Vérification et mise en place du programme

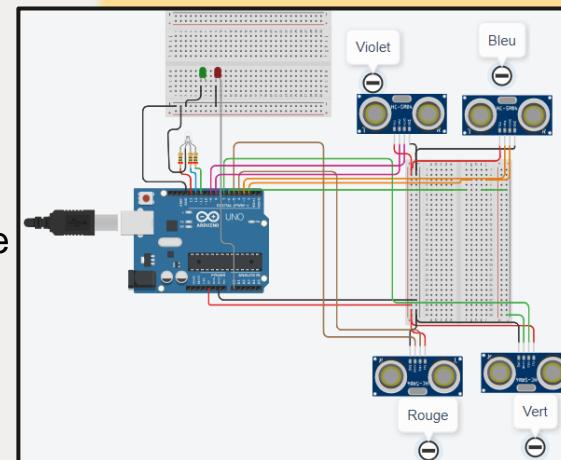


Le développement : projet final

Définition des broches des capteurs et des LEDs

Vérification et mise en place du programme

Impulsion d'onde pour chaque capteur



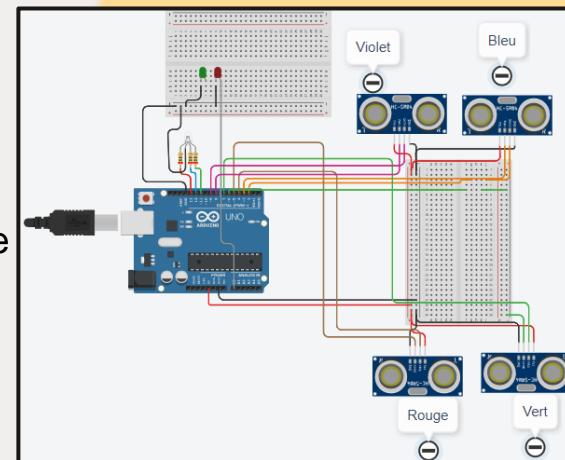
Le développement : projet final

Définition des broches des capteurs et des LEDs

Vérification et mise en place du programme

Impulsion d'onde pour chaque capteur

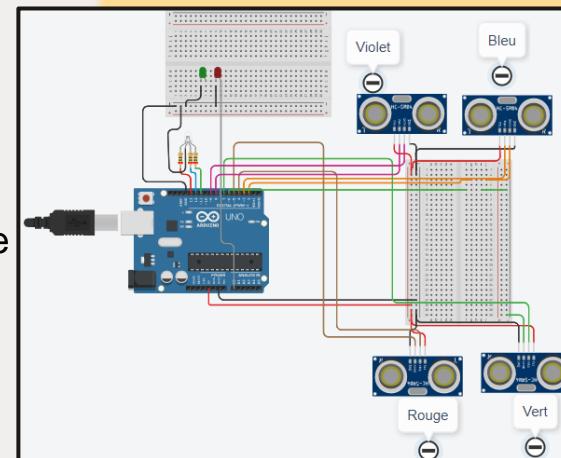
LED RGB s'allume aléatoirement



Le développement : projet final

Définition des broches des capteurs et des LEDs

Vérification et mise en place du programme



Impulsion d'onde pour chaque capteur

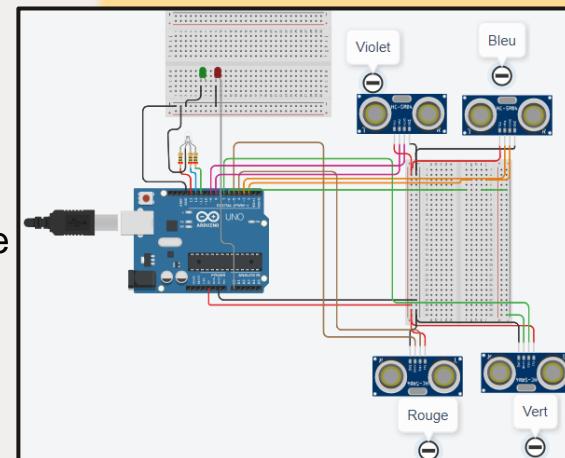
LED RGB s'allume aléatoirement

Détection de la cible

Le développement : projet final

Définition des broches des capteurs et des LEDs

Vérification et mise en place du programme



Impulsion d'onde pour chaque capteur

LED RGB s'allume aléatoirement

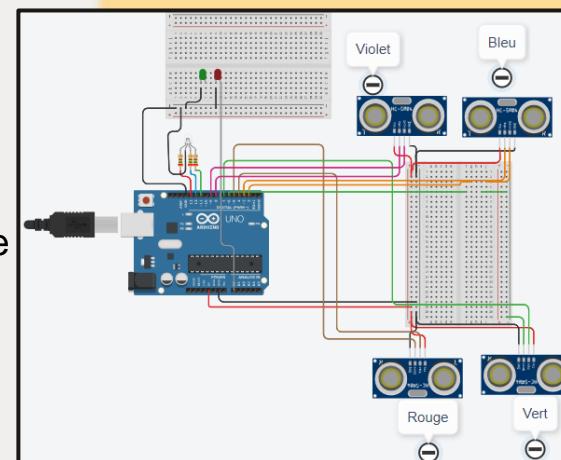
Détection de la cible

LED Vert/Rouge s'allume en fonction du succès ou de l'échec

Le développement : projet final

Définition des broches des capteurs et des LEDs

Vérification et mise en place du programme



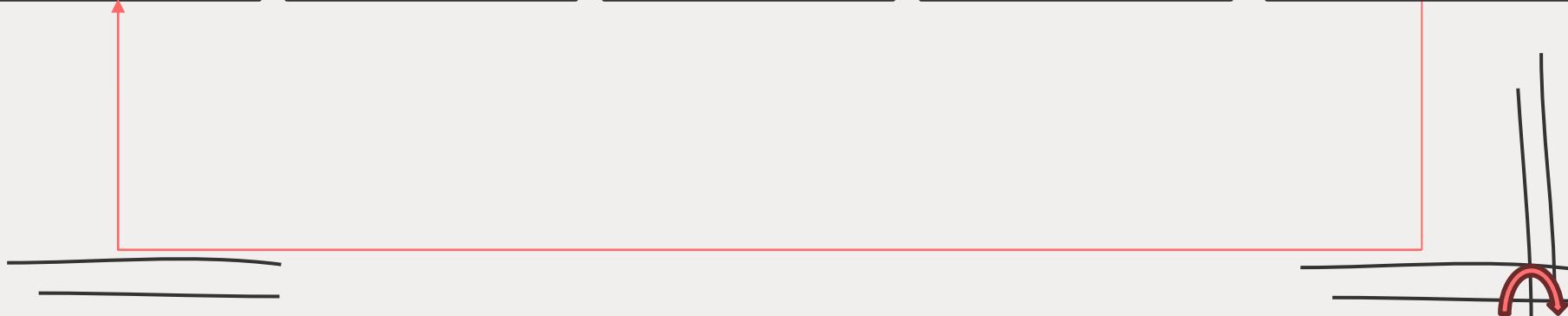
Impulsion d'onde pour chaque capteur

LED RGB s'allume aléatoirement

Détection de la cible

LED Vert/Rouge s'allume en fonction du succès ou de l'échec

LED RGB, LED Vert et Rouge s'éteignent



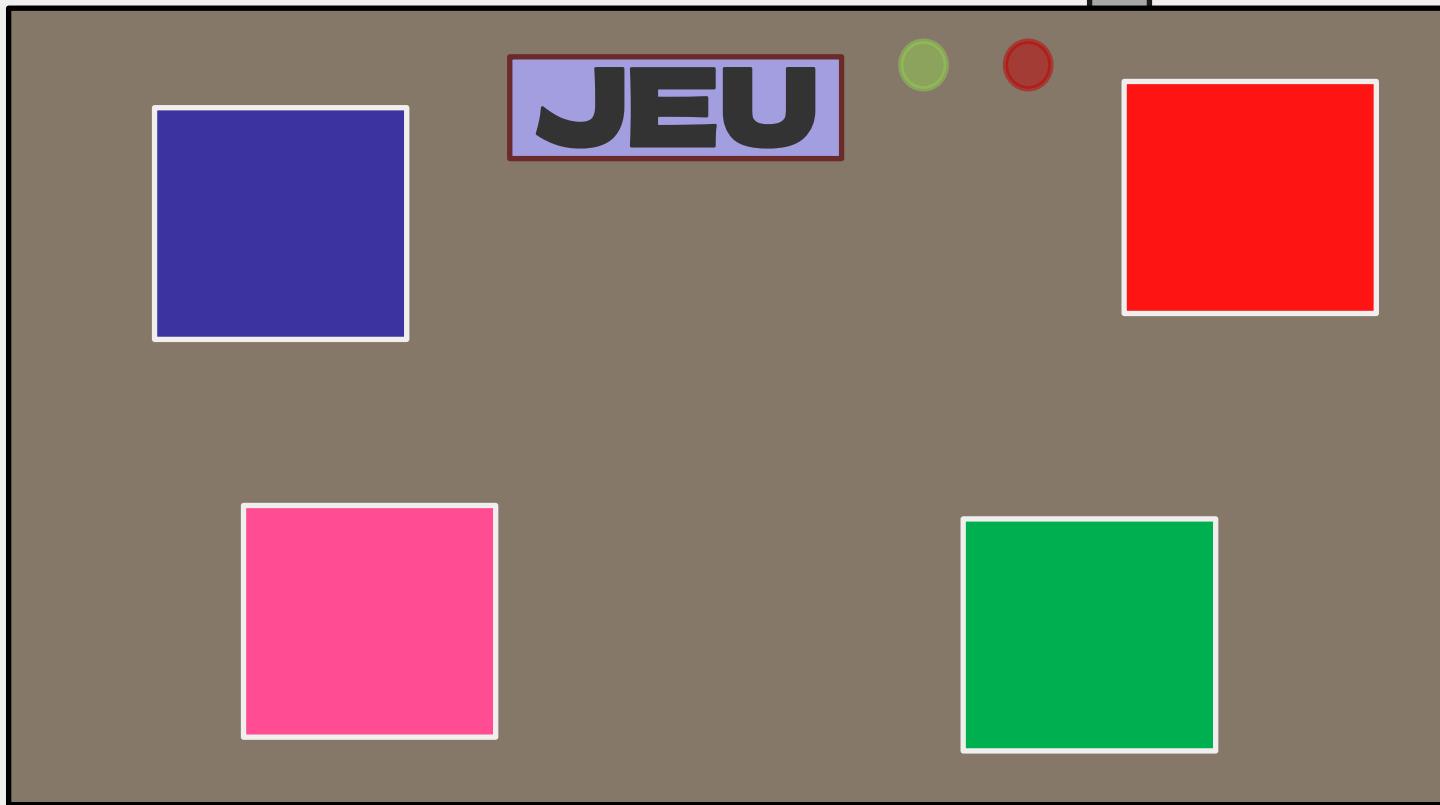
BUT

Première
cible

Grande
cible

Conclusion

Aboutissement



BUT

Première
cibleGrande
cible

Conclusion



Aboutissement : les modes de jeu

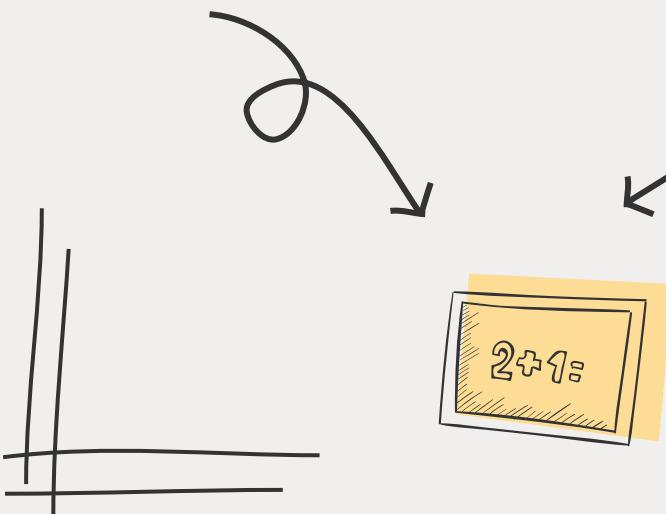
Importance de toucher

Bonne cible touchée = 15 points
(ENNEMI ÉLIMINÉ)



Importance des coéquipiers

Mauvaise cible touchée = -15 points
(ALLIÉ ÉLIMINÉ)
Absence de tir = -5 points



Et finalement ?



Mise en place du jeu

Étant donné les modes de jeu:

- Boucle de 7 itérations
- 7 ou 15 mètres de distance à la cible



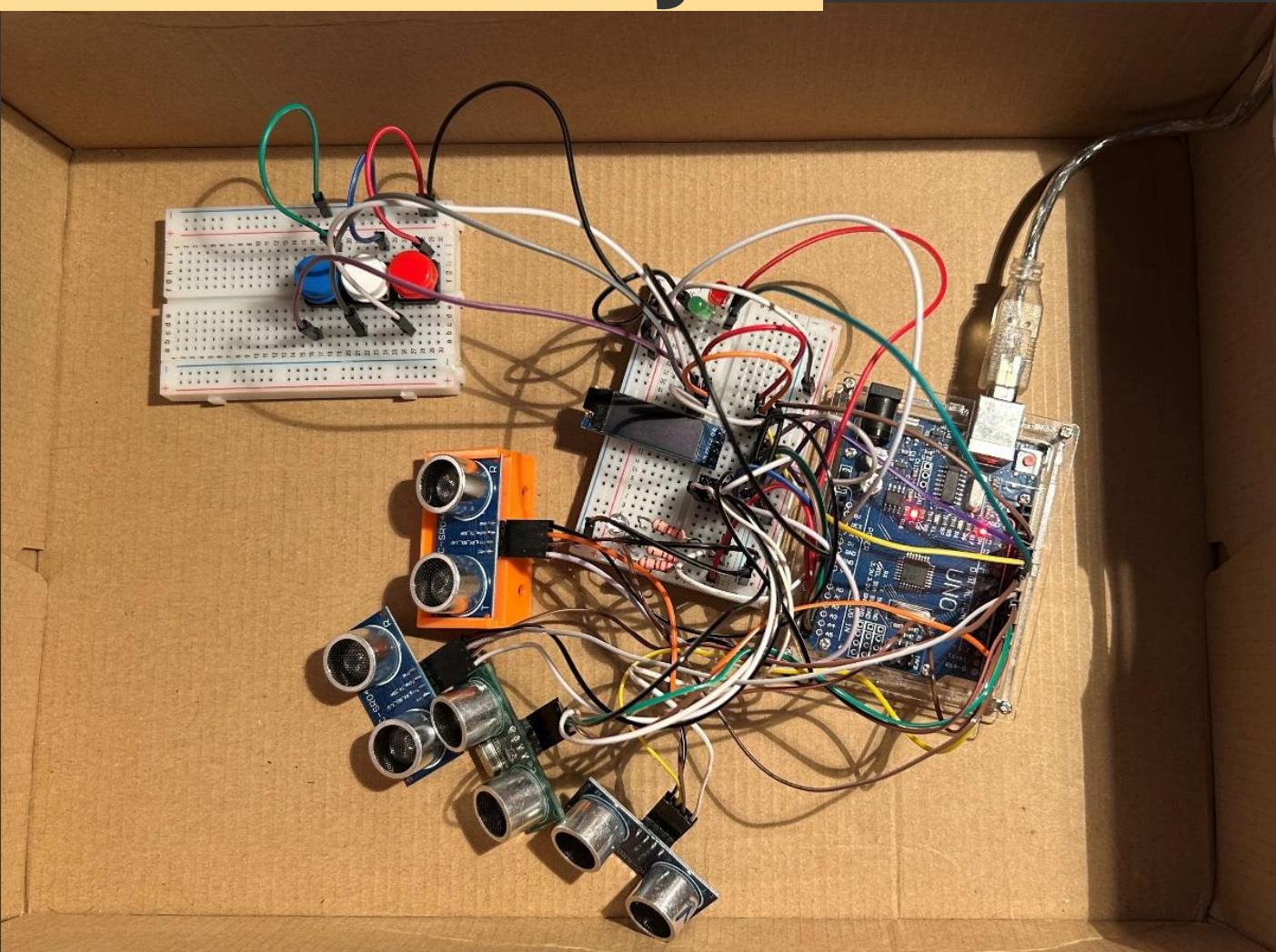
Critères du score final :

- Points (établis avant)
- Temps de d'accomplissement du « défi »

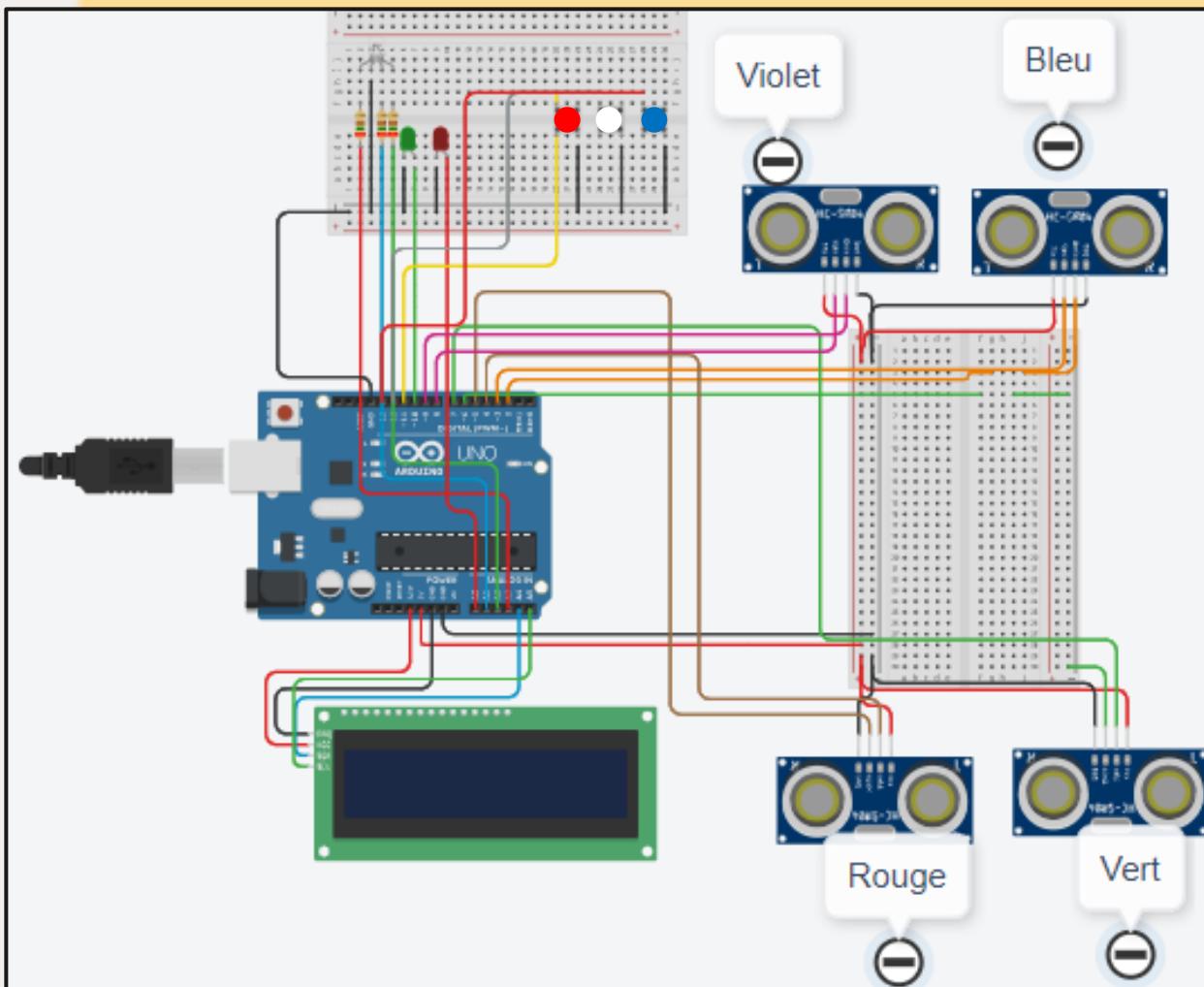


BUT**Première cible****Grande cible****Conclusion**

Branchement du jeu

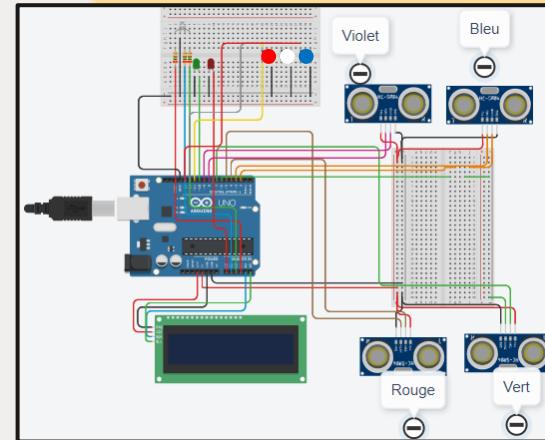


Branchement du jeu



Algorithme du jeu

Vérification et mise en place du programme

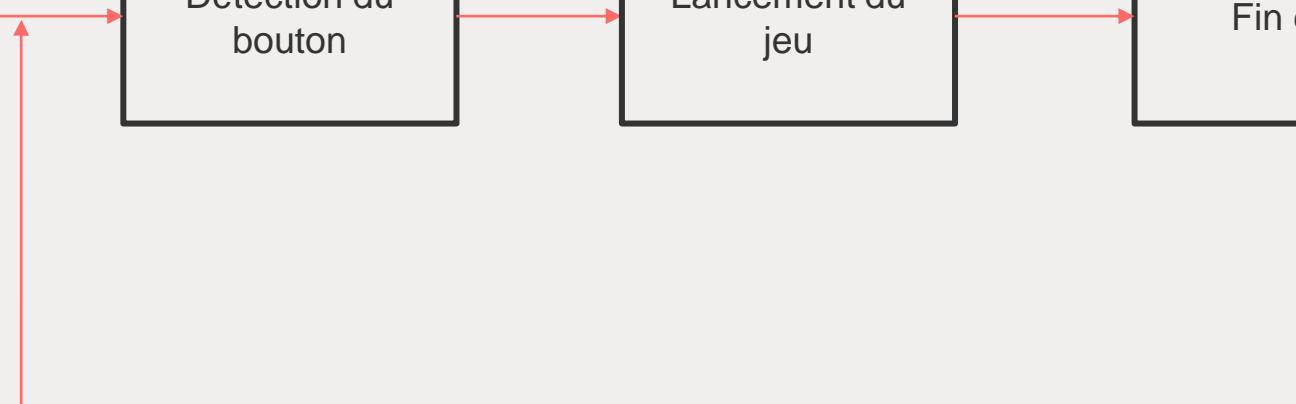


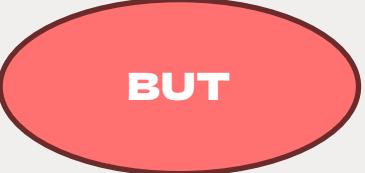
Définition des broches
des capteurs, des
LEDs, des boutons et
de l'écran

Détection du
bouton

Lancement du
jeu

Fin du jeu



 BUT Première
cible Grande
cible Conclusion

Algorithme du jeu

Vérification et mise en place du programme



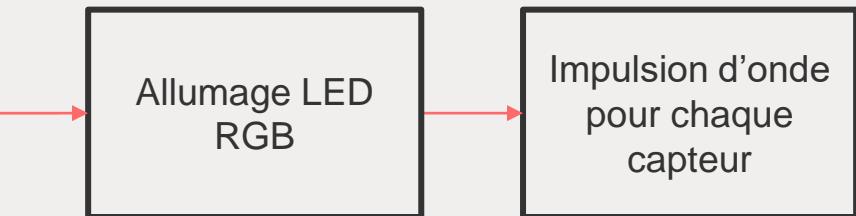
```
graph TD; A[Allumage LED RGB] --> B[Initialisation des variables]; B --> C[Reception des données]; C --> D[Décodage des données]; D --> E[Calcul des coordonnées]; E --> F[Commande à la LED]; F --> G[Retour à l'initialisation]; G --> H[Fin]
```

Allumage LED
RGB



Algorithme du jeu

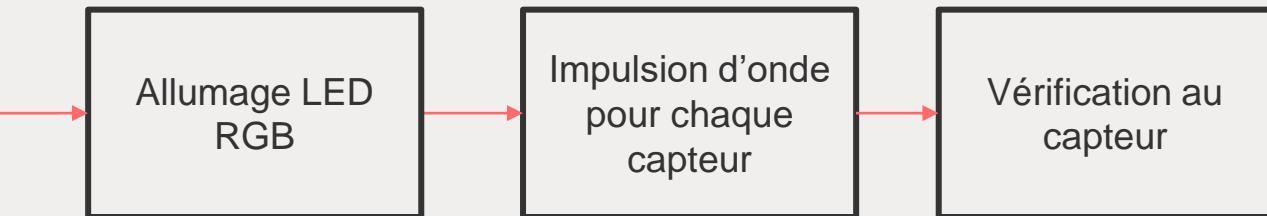
Vérification et mise en place du programme





Algorithme du jeu

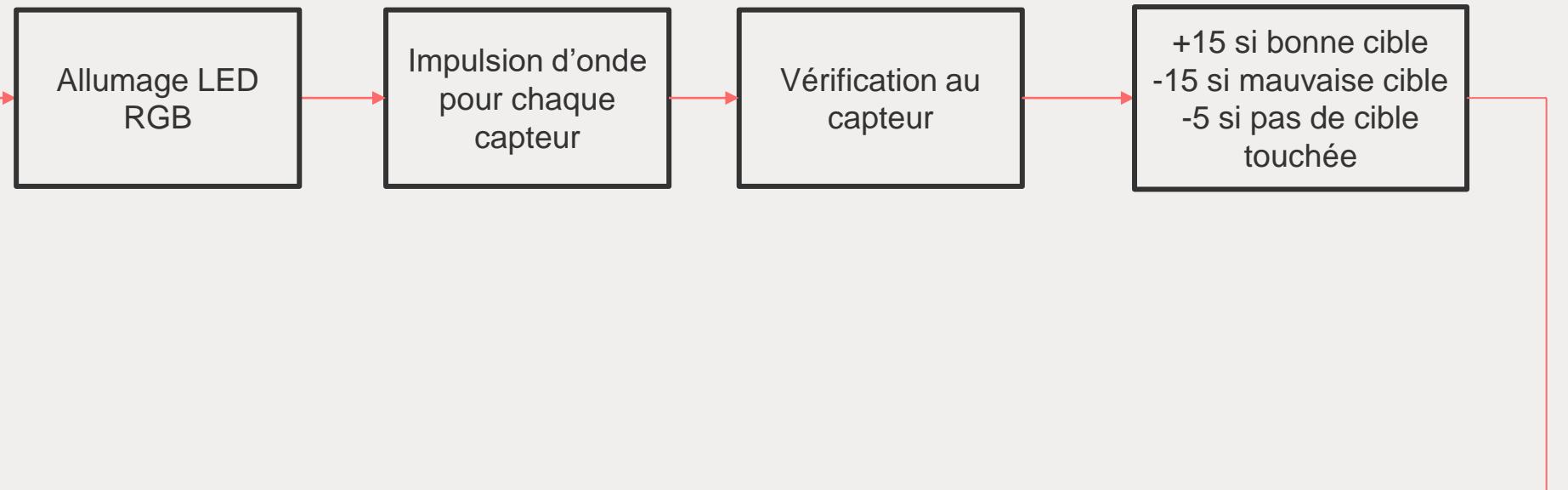
Vérification et mise en place du programme





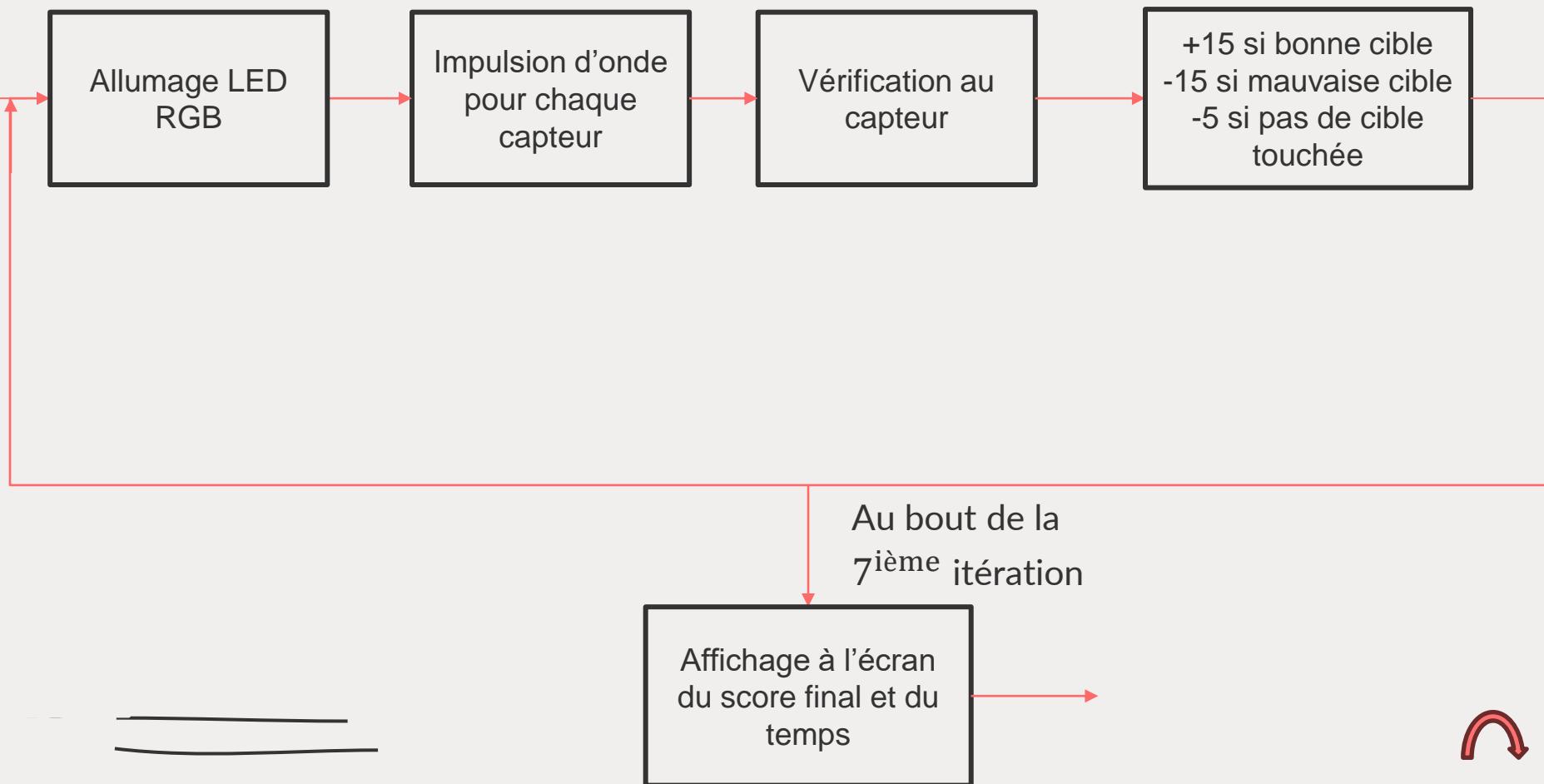
Algorithme du jeu

Vérification et mise en place du programme

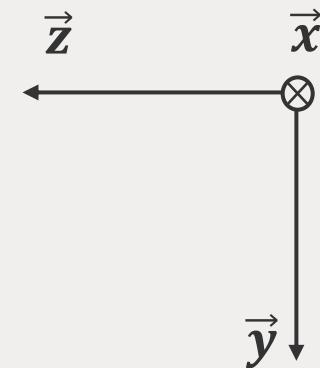
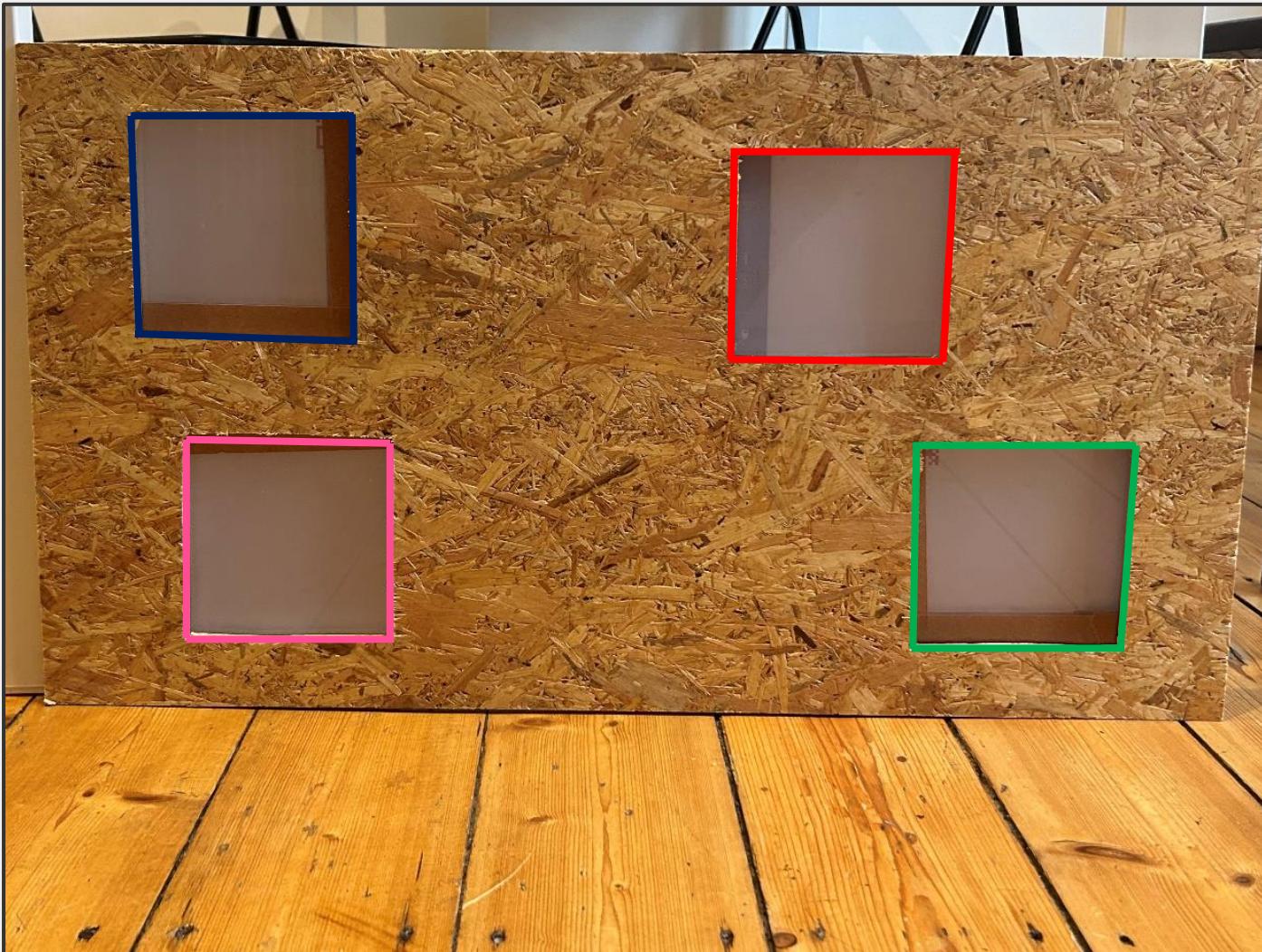


Algorithme du jeu

Vérification et mise en place du programme



Aspect final

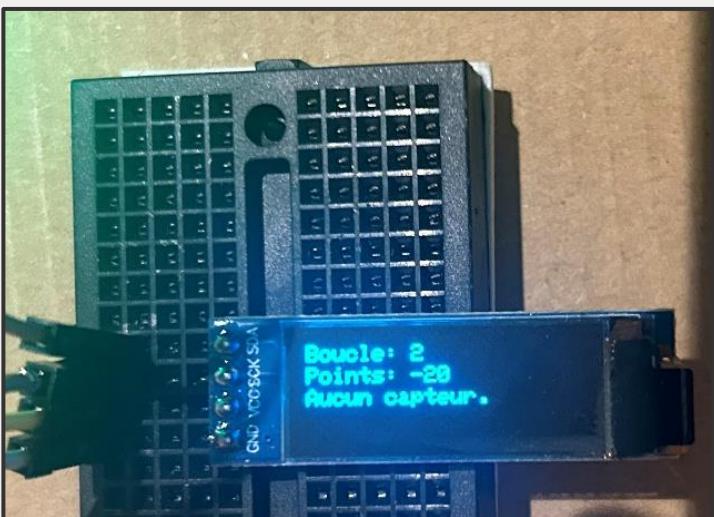


BUT

Première
cibleGrande
cible

Conclusion

Aspect final



BUT**Première
cible****Grande
cible****Conclusion****04**

CONCLUSION

Création d'une cible d'airsoft
automatique



BUT

Première
cibleGrande
cible

Conclusion

Rappel de Problématique

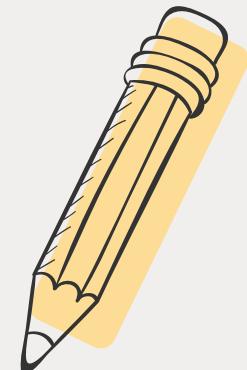
Comment concevoir, développer et

optimiser une cible d'airsoft

automatique afin d'améliorer la

précision des tirs, les compétences

des joueurs, et l'aspect écologique ?



BUT

Première
cibleGrande
cible

Conclusion

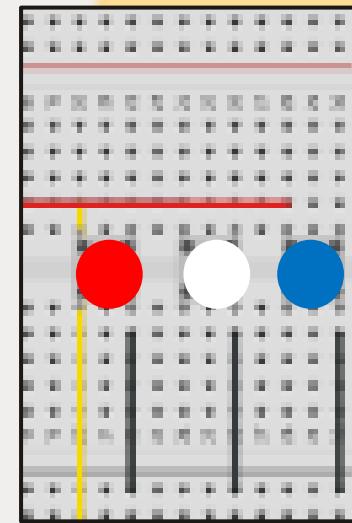
LES POINTS D'AMÉLIORATION

Les boutons

- Démarrage du jeu : ok
- Arrêt (pause) et réinitialisation : échec

Accessibilité

- Ecran derrière la cible
- LEDs sur le côté



BUT

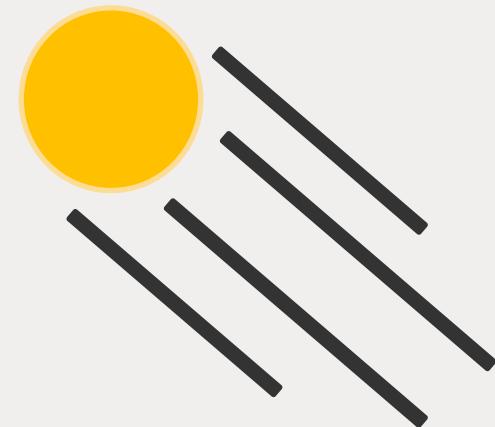
Première
cibleGrande
cible

Conclusion

LES RÉUSSITES

La viabilité :

- Les petites et la grande cibles résistent
- Le jeu est réactif



Fierté :

- La création et l'aboutissement du projet
- Les fonctionnalités apportées en cours de route



MERCI POUR VOTRE

ATTENTION

ZACHARIE BUI
JACOB YVAN

Nb SCEI : 35164

BUT

Première
cible

Grande
cible

Conclusion

Annexe

05

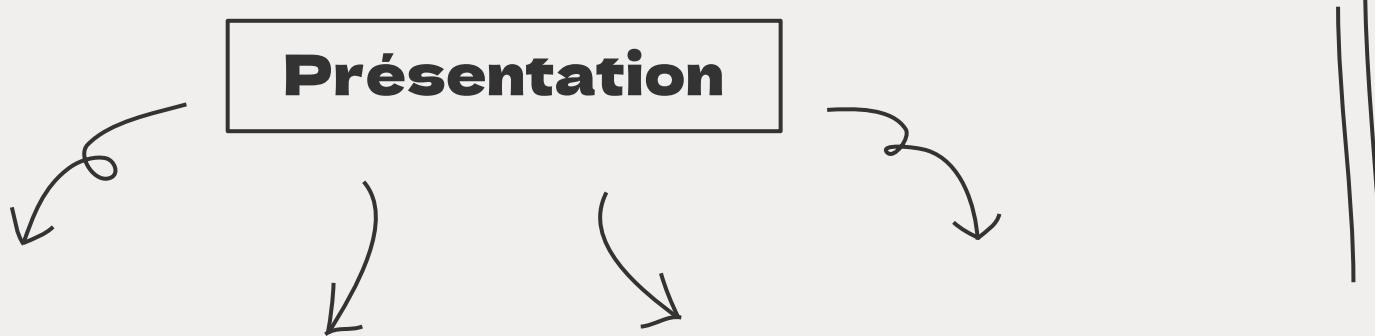
ANNEXE



BUT**Première cible****Grande cible****Conclusion****Annexe**

Retour sur la présentation

Présentation



But

PAGES :

Animation PC : [8](#)
Animation GC : [10](#)

Première cible

PAGES :

Présentation PC : [13](#)
Photos PC : [14->16](#)
Expériences PC : [17](#)
Conclusion PC : [18](#)

Grande cible

PAGES :

Choix capteur : [21->23](#)
Ultrasons détail : [24->27](#)
Initialisation : [28/29](#)
Projet final : [30->32](#)
Aboutissement : [33->41](#)

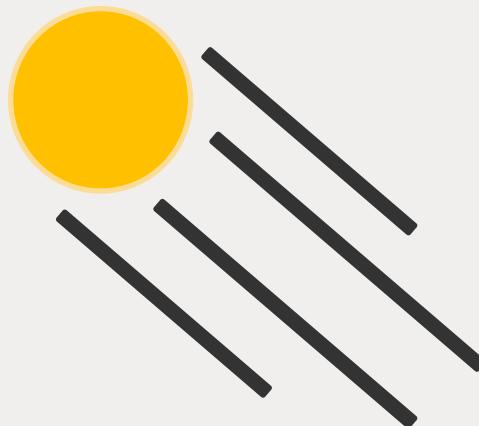
Conclusion

PAGES :

Points d'amélioration : [44](#)
Réussites : [45](#)

BUT**Première
cible****Grande
cible****Conclusion****Annexe**

LES QUESTIONS:



BUT

Première
cibleGrande
cible

Conclusion

Annexe

SOMMAIRE : ANNEXE



Code ARDUINO

PAGES :

Constantes : [51](#)Void setup : [52](#)Initialisation LEDs : [53](#)Initialisation capteurs : [54/55](#)Vérification boutons: [56->58](#)Grandes lignes jeu : [59](#)Jeu final : [60->66](#)

Calculs première cible

PAGES :

Etudes matériaux : [67](#)Premier prototype : [68](#)Deuxième prototype : [69](#)Constante de raideur théo : [70](#)Energie et vitesse bille : [71->75](#)Problèmes de la cible : [76](#)Constante de raideur exp : [77->79](#)Evolution angle : [80->83](#)

Autres

PAGE :

Fonctionnement cellule photoélectrique : [84](#)Modèle d'IZOD : [85](#)Fonctionnement LED : [86](#)Règles airsoft : [87](#)

CONSTANTES

```
1 #include <SPI.h>
2 #include <Wire.h>
3 #include <Adafruit_GFX.h>
4 #include <Adafruit_SSD1306.h>
5
6 const int trigPins[] = {3, 7, 5, 9}; // Broches de déclenchement des capteurs ultrasoniques
7 const int echoPins[] = {2, 6, 4, 8}; // Broches de réception des capteurs ultrasoniques
8 const int RGBrouge = A0;
9 const int RGBbleu = A1;
10 const int RGBvert = A2;
11 const int LEDR = A3;
12 const int LEDV = 10;
13 const int pin_Ibleu = 13; // Pin bouton bleu
14 long duration[4];
15 int distance[4];
16 bool obstacleDetecte = false;
17 Adafruit_SSD1306 display = Adafruit_SSD1306(128, 32, &Wire);
18 byte compteur = 0;
19 unsigned long startTime;
20 const unsigned long loopDuration = 5000; // Durée en millisecondes (5 secondes)
21 const int maxRepetitions = 7;
22 unsigned long totalLoopTime = 0;
23 int points = 0;
24 int bonnesCibles = 0;
25 bool gameStarted = false;
```

VOID SETUP

```
27 void setup() {  
28     Serial.begin(9600);  
29     for (int i = 0; i < 4; i++) {  
30         pinMode(trigPins[i], OUTPUT); // Configuration des capteurs ultrasons  
31         pinMode(echoPins[i], INPUT);  
32     }  
33     pinMode(RGBrouge, OUTPUT);  
34     pinMode(RGBbleu, OUTPUT);  
35     pinMode(RGBvert, OUTPUT);  
36     pinMode(LED_R, OUTPUT);  
37     pinMode(LED_V, OUTPUT);  
38     pinMode(pin_Ibleu, INPUT_PULLUP); // Bouton bleu  
39     display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Initialisation de l'écran OLED  
40     delay(1000);  
41     display.setTextSize(1);  
42     display.setTextColor(WHITE);  
43     display.clearDisplay();  
44     display.display();  
45 }  
46 }
```



```
202 void test_rouge(){
203     digitalWrite(RGBrouge, HIGH);
204     delay(1000);
205     digitalWrite(RGBrouge, LOW);
206 }
207 /////////////////
208 void test_vert(){
209     digitalWrite(RGBvert, HIGH);
210     delay(1000);
211     digitalWrite(RGBvert, LOW);
212 }
213 /////////////////
214 void test_bleu(){
215     digitalWrite(RGBbleu, HIGH);
216     delay(1000);
217     digitalWrite(RGBbleu, LOW);
218 }
```

Initialisation LED

```
void test_ledr(){
    digitalWrite(LED_R, HIGH);
    delay(1000);
    digitalWrite(LED_R, LOW);
}

///////////////
void test_ledv(){
    digitalWrite(LED_V, HIGH);
    delay(1000);
    digitalWrite(LED_V, LOW);
}
```



```
52 void loop(){  
53     for (int i = 0; i < 4; i++) {  
54         digitalWrite(trigPins[i], LOW);  
55         delayMicroseconds(2);  
56         digitalWrite(trigPins[i], HIGH);  
57         delayMicroseconds(10);  
58         digitalWrite(trigPins[i], LOW);  
59         duration[i] = pulseIn(echoPins[i], HIGH);  
60         distance[i] = duration[i] * 0.034 / 2;  
61  
62         // Vérifier si un obstacle a été détecté par un des capteurs  
63         if (distance[i] < 8) {  
64             obstacleDetecte = true;  
65         }  
66     }  
67     if (obstacleDetecte) {  
68         for (int i = 0; i < 4; i++) {  
69             if (distance[i] < 8) {  
70                 // Allumer la LED avec la couleur appropriée en fonction de i  
71                 if (i == 0) { // Si i est égal à 0, allumer en bleu  
72                     digitalWrite(RGBbleu, HIGH);  
73                     delay(300); // Attendre pendant 1 seconde  
74                 }  
75                 if (i == 1) { // Si i est égal à 1, allumer en rouge  
76                     digitalWrite(RGBrouge, HIGH);  
77                     delay(300); // Attendre pendant 1 seconde  
78             }
```

Initialisation capteurs



```
79         if (i == 2) { // Si i est égal à 2, allumer en vert
80             digitalWrite(RGBvert, HIGH);
81             delay(300); // Attendre pendant 1 seconde
82         }
83         if (i == 3) { // Si i est égal à 3, allumer en violet (rouge + bleu)
84             digitalWrite(RGBrouge, HIGH);
85             digitalWrite(RGBbleu, HIGH);
86             delay(300); // Attendre pendant 1 seconde
87         }
88     }
89 }
90 }
91 if (!obstacleDetecte) {
92     digitalWrite(RGBrouge, LOW);
93     digitalWrite(RGBvert, LOW);
94     digitalWrite(RGBbleu, LOW);
95 }
96 digitalWrite(RGBrouge, LOW);
97 digitalWrite(RGBvert, LOW);
98 digitalWrite(RGBbleu, LOW);
99 }
```

Initialisation capteurs



Vérification boutons

```
27 boolean ledRAllumer = 0;  
28 boolean ledVAllumer=0;  
29 boolean RGBBallumer = 0;  
30 boolean RGBRallumer = 0;  
31 boolean RGBVallumer = 0;  
32 boolean boutonbleu = 0;  
33 boolean boutonrouge = 0;  
34 boolean boutonblanc = 0;  
35 byte boutonActif = 0;
```



Vérification boutons

```
63 √ void loop() {  
64     boolean etatBoutonrouge = digitalRead(pin_Irouge); // Récupère l'état du bouton  
65     boolean etatBoutonblanc = digitalRead(pin_Iblanc); // Récupère l'état du bouton  
66     boolean etatBoutonbleu = digitalRead(pin_Ibleu); // Récupère l'état du bouton  
67  
68     if (etatBoutonbleu == 0 && boutonbleu == 0 && (boutonActif == 0 || boutonActif == 3)) { // On appuie sur le bouton bleu  
69         boutonbleu = 1;  
70         boutonActif = 3;  
71         if (RGBBallumer == 0) { // Si la led est éteinte on l'allume  
72             digitalWrite(RGBbleu, HIGH);  
73             RGBBallumer = 1;  
74         }  
75         if (RGBVallumer == 1) { // Si la led verte est allumée, on l'éteint  
76             digitalWrite(RGBvert, LOW);  
77             RGBVallumer = 0;  
78         }  
79     }  
80  
81     if (etatBoutonblanc == 0 && boutonblanc == 0 && (boutonActif == 0 || boutonActif == 2)) { // On appuie sur le bouton blanc  
82         boutonblanc = 1;  
83         boutonActif = 2;  
84         if (RGBVallumer == 0) { // Si la led est éteinte on l'allume  
85             digitalWrite(RGBvert, HIGH);  
86             RGBVallumer = 1;  
87         }  
88         if (RGBBallumer == 1) { // Si la led bleue est allumée, on l'éteint  
89             digitalWrite(RGBbleu, LOW);  
90             RGBBallumer = 0;
```



Vérification boutons

```
91 }  
92 }  
93  
94 if (etatBoutonrouge == 0 && boutonrouge == 0 && boutonActif != 0) { // On appuie sur le bouton rouge pour éteindre le bouton actif  
95     boutonrouge = 1;  
96     boutonActif = 0;  
97     if (RGBVallumer == 1) { // Si la led verte est allumée, on l'éteint  
98         digitalWrite(RGBvert, LOW);  
99         RGBVallumer = 0;  
100    }  
101    if (RGBBallumer == 1) { // Si la led bleue est allumée, on l'éteint  
102        digitalWrite(RGBbleu, LOW);  
103        RGBBallumer = 0;  
104    }  
105 }  
106  
107 if (etatBoutonrouge == 1 && boutonrouge == 1) { //On arrête d'appuyer sur le bouton rouge  
108     boutonrouge = 0;  
109 }  
110  
111 if (etatBoutonbleu == 1 && boutonbleu == 1) { //On arrête d'appuyer sur le bouton bleu  
112     boutonbleu = 0;  
113 }  
114  
115 if (etatBoutonblanc == 1 && boutonblanc == 1) { //On arrête d'appuyer sur le bouton blanc  
116     boutonblanc = 0;  
117 }  
118  
119 delay(10); // Attente de 10 ms  
120 }
```



Acheminement de pensée pour le jeu

- Faire clignoter la LED RGB aléatoirement V1
- Associer un capteur à une couleur V2
- Déetecter si la bonne cible est touchée V3
- Allumer la LED verte, rouge ou clignotement en conséquence V4
- Implémentation du système de point V5
- Ajout de l'écran V6
- Commencer une partie grâce au bouton V7
- Utiliser plusieurs boutons pour mettre sur pause, réinitialiser le jeu, commencer une partie V8

| |
|-----------------------------------|
| adafruit |
| libraries |
| TIPE_FONCTIONS |
| TIPE_FONCTIONS-V2 |
| TIPE_init |
| TIPE_initialisation_final |
| TIPE_interrupteurs |
| TIPE_interrupteurs_replacer_final |
| TIPE_JEU_V1 |
| TIPE_JEU_V2 |
| TIPE_JEU_V3 |
| TIPE_JEU_V4 |
| TIPE_JEU_V5 |
| TIPE_JEU_V6 |
| TIPE_JEU_V7 |
| TIPE_JEU_V8 |
| TIPE_TEST_CIBLE |
| TIPEV1 |



```
47 void loop() {  
48     while (digitalRead(pin_Ibleu) == HIGH) {  
49         // Attendre que le bouton bleu soit enfoncé pour démarrer le jeu  
50         delay(100);  
51     }  
52  
53     gameStarted = true; // Démarrer le jeu  
54     startTime = millis(); // Réinitialiser le timer de départ  
55  
56     while (gameStarted) {  
57         unsigned long boucleStartTime = millis(); // Début du temps de boucle  
58         int aleatoire = random(0, 4);  
59         digitalWrite(RGBbleu, LOW);  
60         digitalWrite(RGBvert, LOW);  
61         digitalWrite(RGBrouge, LOW);  
62  
63         if (aleatoire == 0) {  
64             digitalWrite(RGBbleu, HIGH);  
65             Serial.println("LED RGB: Bleu");  
66         } else if (aleatoire == 1) {  
67             digitalWrite(RGBvert, HIGH);  
68             Serial.println("LED RGB: Vert");  
69         } else if (aleatoire == 2) {  
70             digitalWrite(RGBrouge, HIGH);  
71             Serial.println("LED RGB: Rouge");  
72     }  
73 }
```

Jeu final



```
72 } else if (aleatoire == 3) {  
73     digitalWrite(RGBbleu, HIGH);  
74     digitalWrite(RGBrouge, HIGH);  
75     Serial.println("LED RGB: Violet");  
76 }  
77  
78     Serial.println("Début boucle.");  
79     obstacleDetecte = false;  
80     bool capteurCorrect = false;  
81  
82     while (millis() - boucleStartTime < loopDuration) {  
83         for (int i = 0; i < 4; i++) {  
84             digitalWrite(trigPins[i], LOW);  
85             delayMicroseconds(2);  
86             digitalWrite(trigPins[i], HIGH);  
87             delayMicroseconds(10);  
88             digitalWrite(trigPins[i], LOW);  
89  
90             duration[i] = pulseIn(echoPins[i], HIGH);  
91             distance[i] = duration[i] * 0.034 / 2;  
92  
93             if (distance[i] < 8) {  
94                 Serial.print("Capteur ");  
95                 Serial.print(i);  
96                 Serial.println(" : Obstacle détecté !");
```

Jeu final



```
97         obstacleDetecte = true;  
98  
99     }  
100    if (i == aleatoire) {  
101        capteurCorrect = true;  
102    } else {  
103        points -= 15;  
104        Serial.println("Mauvais capteur touché. -15 points");  
105        digitalWrite(LED_R, HIGH);  
106        delay(1000);  
107        digitalWrite(LED_R, LOW);  
108    }  
109    } else {  
110        Serial.print("Capteur ");  
111        Serial.print(i);  
112        Serial.println(" : Aucun obstacle.");  
113    }  
114  
115    if (obstacleDetecte) {  
116        if (capteurCorrect) {  
117            points += 15;  
118            bonnesCibles++;  
119            Serial.println("Bon capteur touché. +15 points");  
120            digitalWrite(LED_V, HIGH);  
121            delay(1000);  
122        }  
123    }  
124  
125    if (points >= 100) {  
126        Serial.println("Félicitations ! Vous avez atteint 100 points.");  
127        Serial.println("Le jeu est terminé.");  
128        Serial.println("Merci d'avoir joué.");  
129        Serial.println("Au revoir.");  
130        Serial.println("Fin du programme.");  
131    }  
132  
133    // Ajoutez ici la logique pour gérer les autres cibles et les erreurs.  
134  
135    // Vérifiez si une autre cible a été atteinte ou pas.  
136    // Si oui, ajoutez 15 points et incrémentez le compteur de bonnes cibles.  
137  
138    // Si non, vérifiez si le capteur est correct ou pas.  
139    // Si oui, ajoutez 15 points et incrémentez le compteur de bonnes cibles.  
140  
141    // Si non, soustrayez 15 points et décrémentez le compteur de bonnes cibles.  
142  
143    // Continuez jusqu'à ce que le joueur atteigne 100 points.  
144  
145    // Si le joueur atteint 100 points, affichez un message de victoire et terminez le programme.  
146  
147    // Sinon, continuez à afficher les messages et à vérifier les capteurs jusqu'à ce que le joueur atteigne 100 points.  
148  
149    // Utilisez la fonction delay() pour faire attendre l'Arduino entre les vérifications.  
150  
151    // N'oubliez pas de mettre à jour les variables appropriées à chaque vérification.  
152  
153    // Bonne chance et à bientôt !
```

Jeu final



```
122         digitalWrite(LEDV, LOW);
123     }
124     break;
125 }
126 }
127
128 if (!obstacleDetecte) {
129     points -= 5;
130     Serial.println("Aucun capteur touché. -5 points");
131     for (int i = 0; i < 10; i++) {
132         digitalWrite(LEDR, HIGH);
133         delay(50);
134         digitalWrite(LEDR, LOW);
135         delay(50);
136     }
137 }
138
139 display.clearDisplay();
140 display.setCursor(0, 0);
141 display.print("Boucle: ");
142 display.print(compteur + 1);
143 display.setCursor(0, 10);
144 display.print("Points: ");
145 display.print(points);
146 if (obstacleDetecte) {
```

Jeu final



```
147     if (capteurCorrect) {  
148         display.setCursor(0, 20);  
149         display.print("Bon capteur!");  
150     } else {  
151         display.setCursor(0, 20);  
152         display.print("Mauvais capteur!");  
153     }  
154 } else {  
155     display.setCursor(0, 20);  
156     display.print("Aucun capteur.");  
157 }  
158 display.display();  
159  
160 unsigned long boucleEndTime = millis();  
161 unsigned long boucleDuration = boucleEndTime - boucleStartTime;  
162 totalLoopTime += boucleDuration;  
163  
164 Serial.print("Temps de la boucle: ");  
165 Serial.print(boucleDuration);  
166 Serial.println(" ms");  
167  
168 compteur++;  
169 Serial.print("Boucle terminée. Compteur: ");  
170 Serial.println(compteur);  
171 Serial.print("Points actuels: ")
```

Jeu final



```
172     Serial.println(points);  
173  
174     digitalWrite(RGBbleu, LOW);  
175     digitalWrite(RGBvert, LOW);  
176     digitalWrite(RGBrouge, LOW);  
177  
178     delay(2000); // Délai entre chaque itération de la boucle  
179  
180     if (compteur >= maxRepetitions) {  
181         gameStarted = false; // Arrêter le jeu après un certain nombre de répétitions  
182     }  
183 }  
184  
185 // Afficher les résultats finaux sur l'écran OLED  
186 display.clearDisplay();  
187 display.setCursor(0, 0);  
188 display.print("Temps total: ");  
189 display.print(totalLoopTime);  
190 display.print(" ms");  
191 display.setCursor(0, 10);  
192 display.print("Points totaux: ");  
193 display.print(points);  
194 display.setCursor(0, 20);  
195 display.print("Bonnes cibles: ");  
196 display.print(bonnesCibles);
```

Jeu final

```
197     display.display();  
198  
199     // Afficher les résultats finaux dans la console série  
200     Serial.print("Temps total des boucles: ");  
201     Serial.print(totalLoopTime);  
202     Serial.println(" ms");  
203  
204     Serial.print("Points totaux: ");  
205     Serial.println(points);  
206  
207     Serial.print("Bonnes cibles touchées: ");  
208     Serial.println(bonnesCibles);  
209  
210     Serial.println("Fin du programme après 7 répétitions.");  
211  
212     // Réinitialiser les variables pour une nouvelle partie  
213     compteur = 0;  
214     points = 0;  
215     bonnesCibles = 0;  
216     totalLoopTime = 0;  
217     digitalWrite(LED_R, LOW);  
218     digitalWrite(LED_V, LOW);  
219     digitalWrite(RGBbleu, LOW);  
220     digitalWrite(RGBvert, LOW);  
221     digitalWrite(RGBrouge, LOW);  
222 }
```

Jeu final



Première cible : choix des matériaux

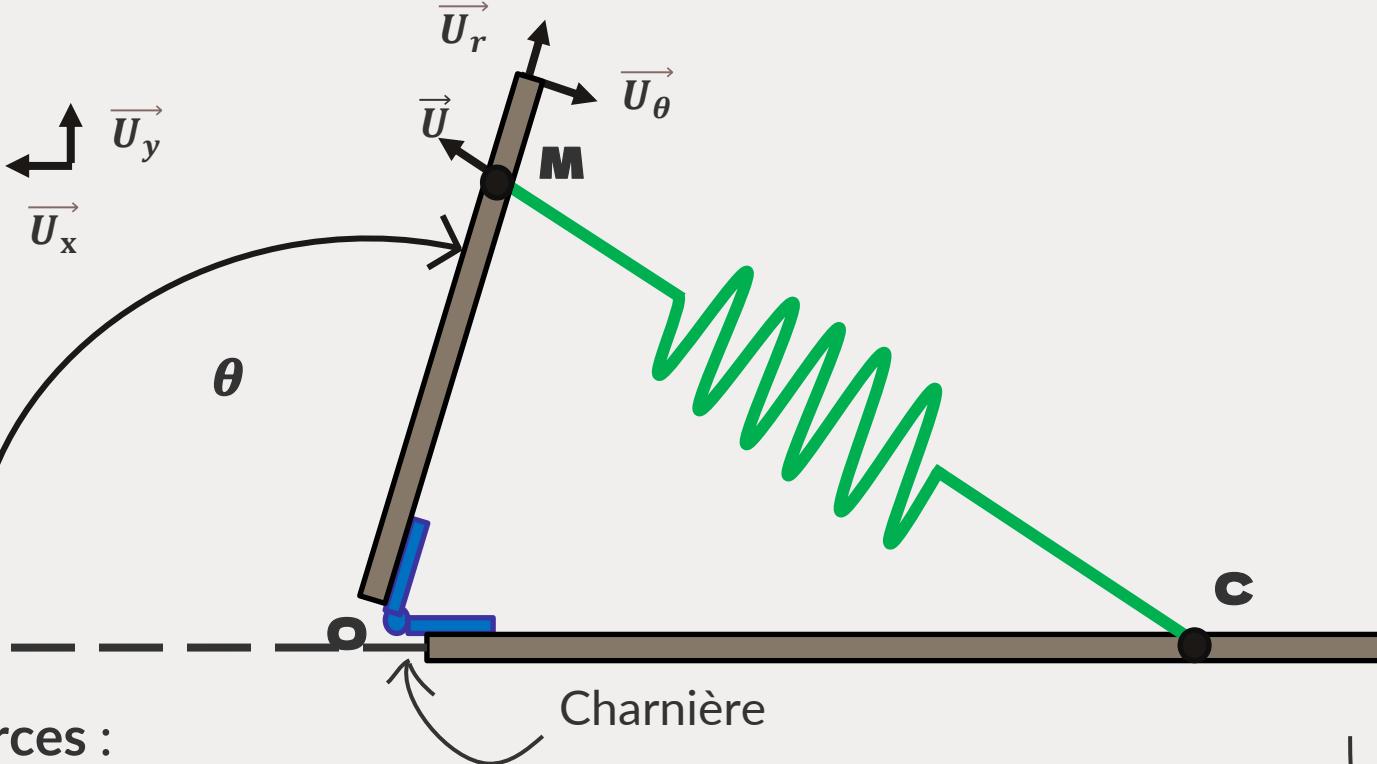
| Matériaux | Planche de chêne | Planche de sapin | Polycarbonate | PVC |
|---|--------------------|---------------------|----------------|----------------|
| Résistance aux chocs (IZOD) kJ/m ² | 0,5 à 0,74* | 0,35 à 0,65* | 10 | 4 |
| MASSE VOLUMIQUE g/cm ³ | 0,71 | 0,46 | 1,20 | 1,38 |
| COÛT € au m ² | 80 à 100 | 45 | 10 à 55 | 14 à 39 |



Le premier prototype



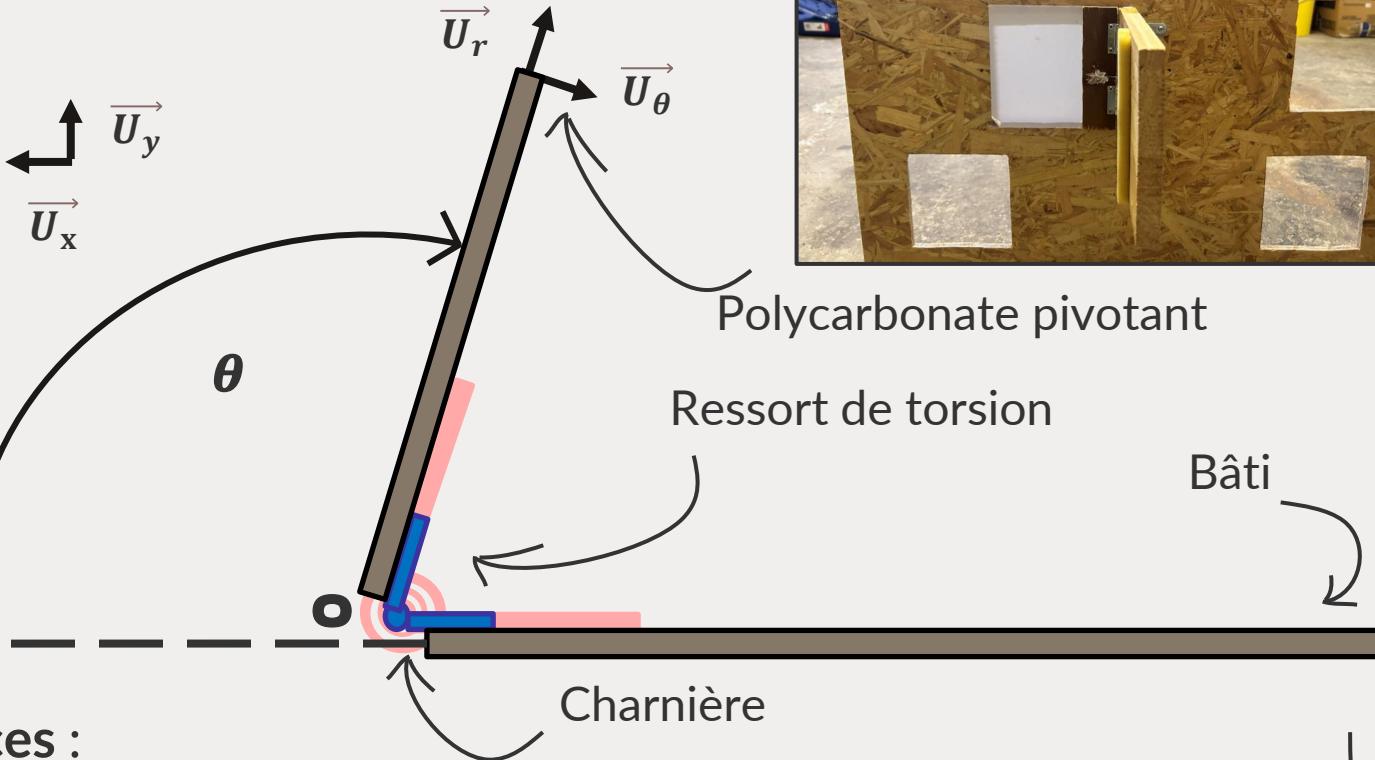
$$\vec{U} = \frac{\overrightarrow{CM}}{\|\overrightarrow{CM}\|}$$



Inventaire des forces :

- Ressort
- Poids (pas pris en compte car la réaction compense)
- Frottement (charnière)

Deuxième prototype

 \vec{U}_Δ 

Inventaire des forces :

- Ressort de torsion
- Poids (pas pris en compte car la réaction compense)
- Frottement (négligé car charnière lubrifiée)

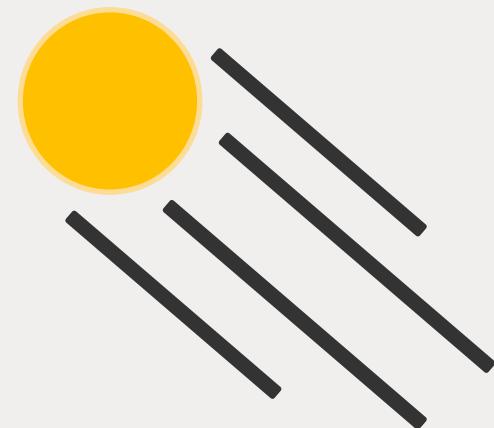
Première cible : théorie

Hypothèses : Forces dérivant d'énergie potentielle /
pas d'énergie dissipée, 100% transmise

Energie potentielle du ressort : $\frac{1}{2} C \theta_{max}^2$

Energie cinétique de la bille : $\frac{1}{2} mV^2$

Théorème de la conservation de l'énergie mécanique :

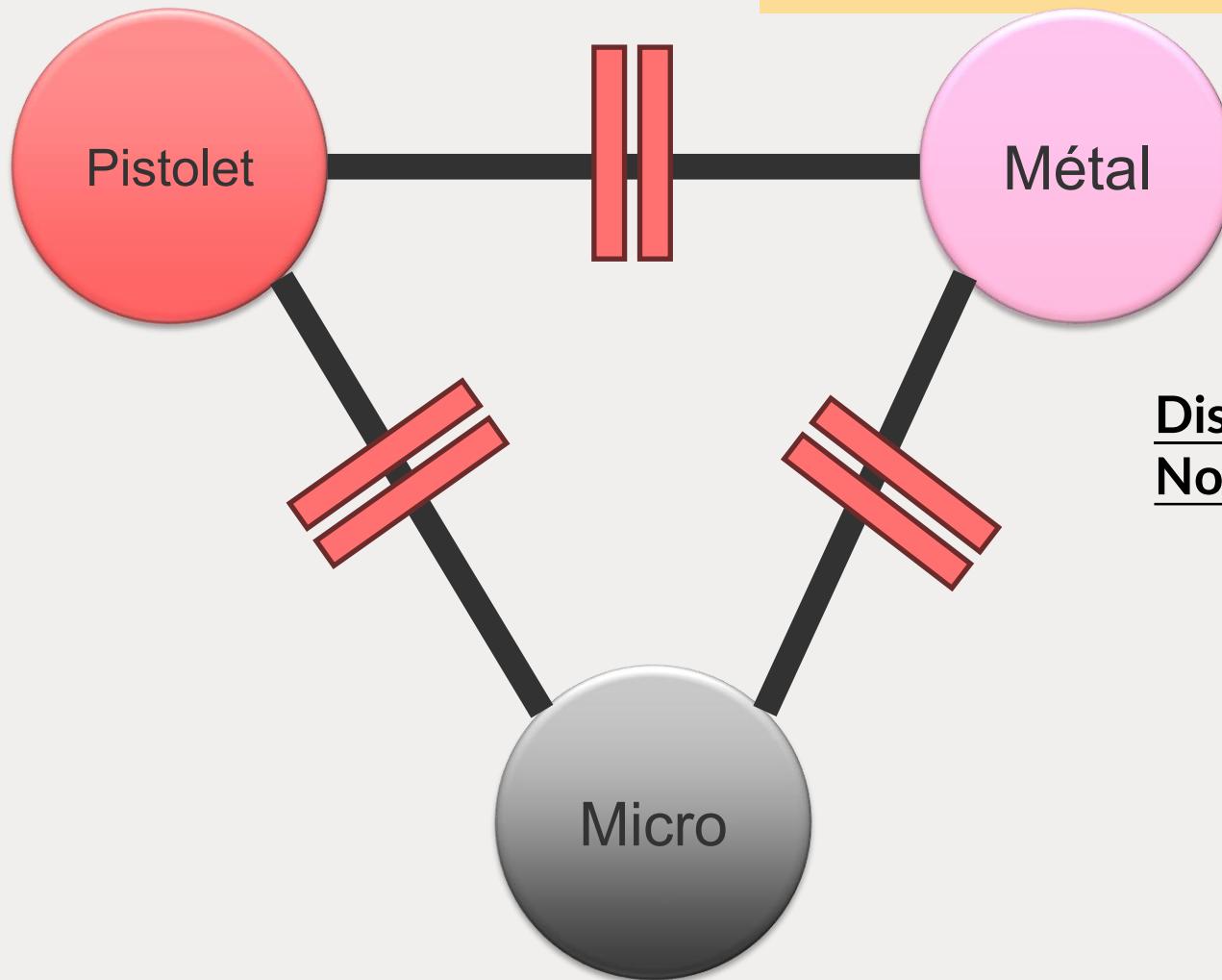


$$\Delta E_m = 0$$

$$E_{c1} = E_{p2} \Leftrightarrow \frac{1}{2} mV^2 = \frac{1}{2} C \theta_{max}^2$$



Première cible : vitesse de la bille

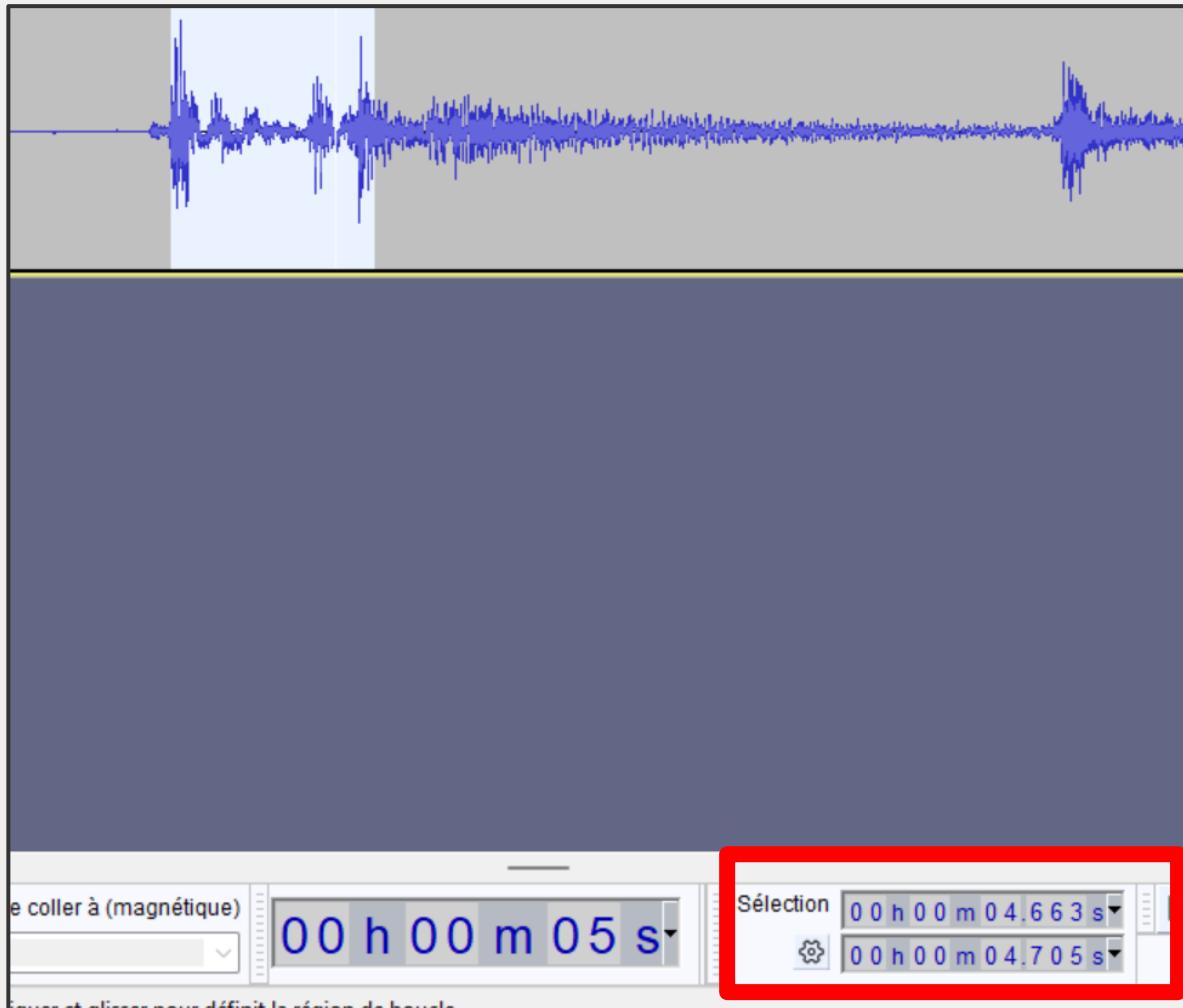


Distance : 5 mètres

Nombre de tirs : 10



Première cible : vitesse de la bille

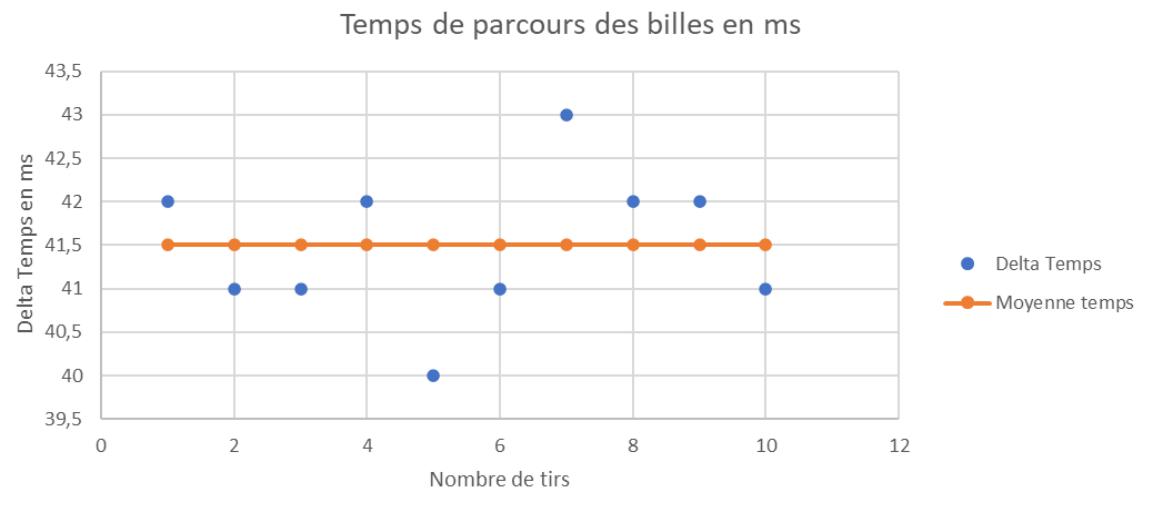


$$\Delta t = 42 \text{ ms}$$
$$V = \frac{d}{\Delta t} = 119 \text{ m.s}^{-1}$$

Expériences répétées
plusieurs fois



Vitesse de la bille : incertitudes



$$U_A = \frac{\sigma_{n-1}}{\sqrt{n}}$$

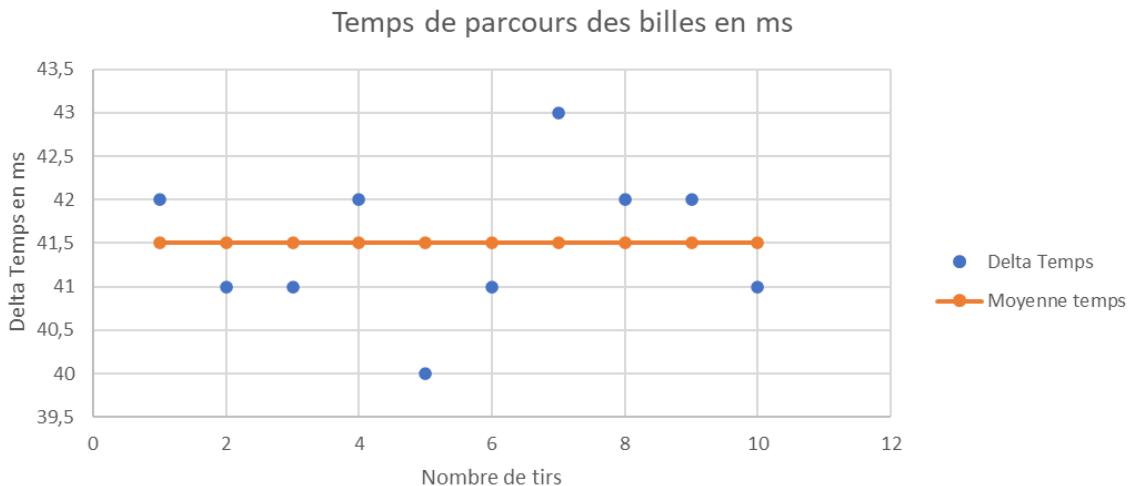
$$\sigma_{n-1} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (m_i - \bar{m})^2}$$

$$\sum_{i=1}^n (m_i - \bar{m})^2 = 6.5$$

$$\sigma_{n-1} = \sqrt{\frac{6.5}{10-1}} = 0.849$$



Vitesse de la bille : incertitudes



$$U_A = \frac{\sigma_{n-1}}{\sqrt{n}}$$

$$\sigma_{n-1} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (m_i - \bar{m})^2}$$

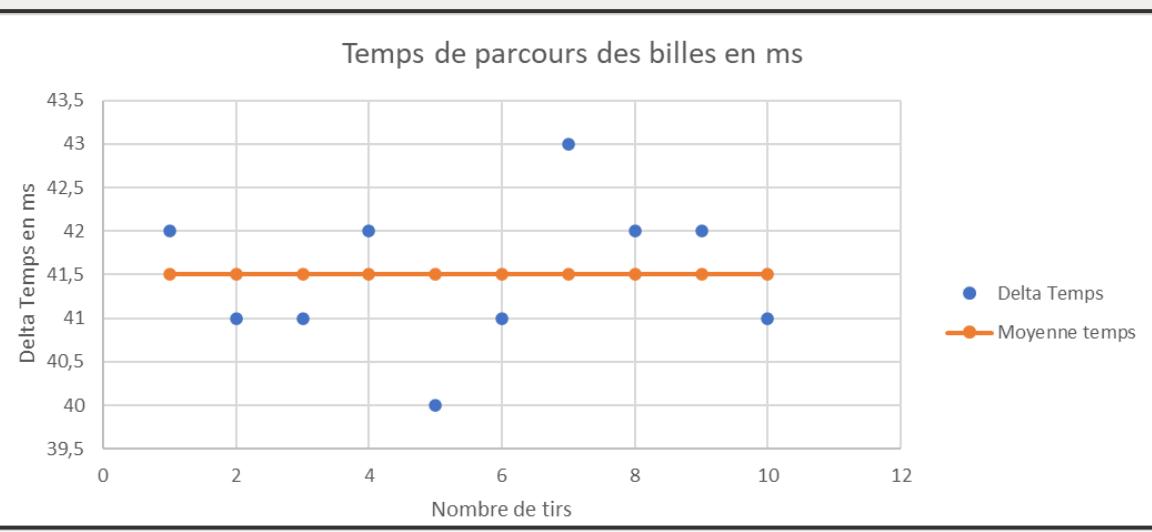
$$= 0.849$$

$$U_A = \frac{\sigma_{n-1}}{\sqrt{n}} = 0,268$$

$$T = 41.5 \pm 0.3 \text{ ms}$$



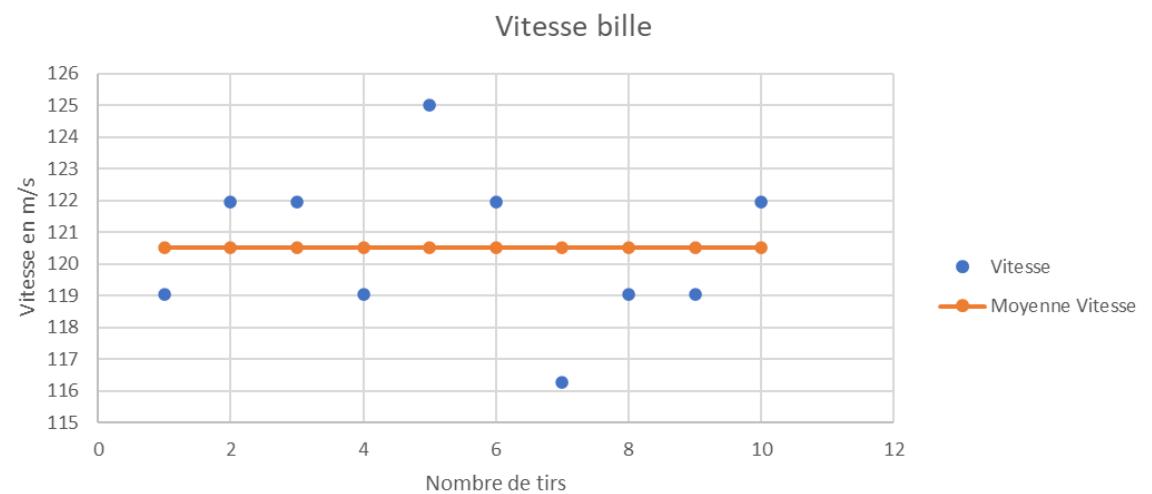
Vitesse de la bille : énergie cinétique



Vitesse moyenne :
 $120,5 \text{ m.s}^{-1}$

Masse d'une bille : $0,20\text{g}$

Energie cinétique de la bille :
 $\text{Ec} = \frac{1}{2} mV^2 = 1,45 \text{ J}$



BUT

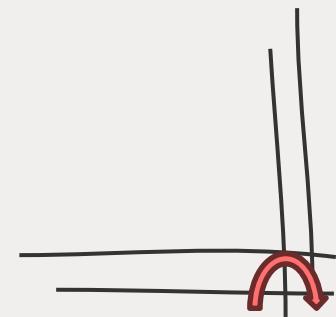
Première
cibleGrande
cible

Conclusion

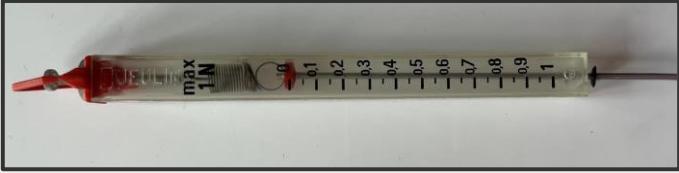
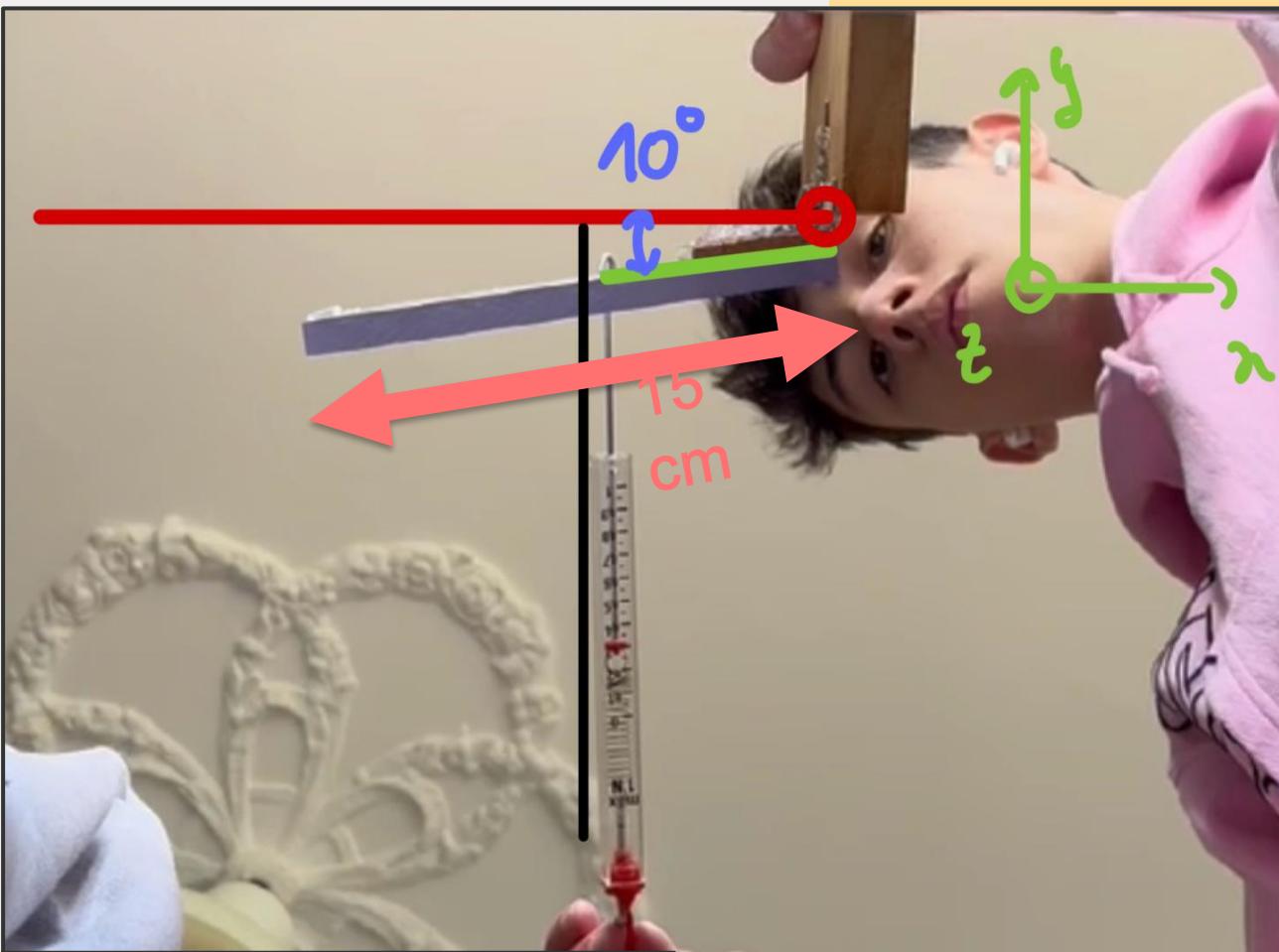
Annexe

Problèmes de la première cible

| Exigence | Energie cinétique de la bille | Energie potentielle ressort |
|------------------|-------------------------------|-----------------------------|
| Valeurs (Joules) | 1,45 | 0,007 |



Première cible : constante de raideur



$$\vec{M}_{/o}^t = \overrightarrow{OM} \wedge \vec{F}$$

$$M_{axe}^t = \text{bras de levier} \times F$$

$$M_{axe}^t = C \times \theta$$

$$\text{bras de levier} \times F = C \times \theta$$

$$\text{Bras de levier} = 6.66\text{cm}$$

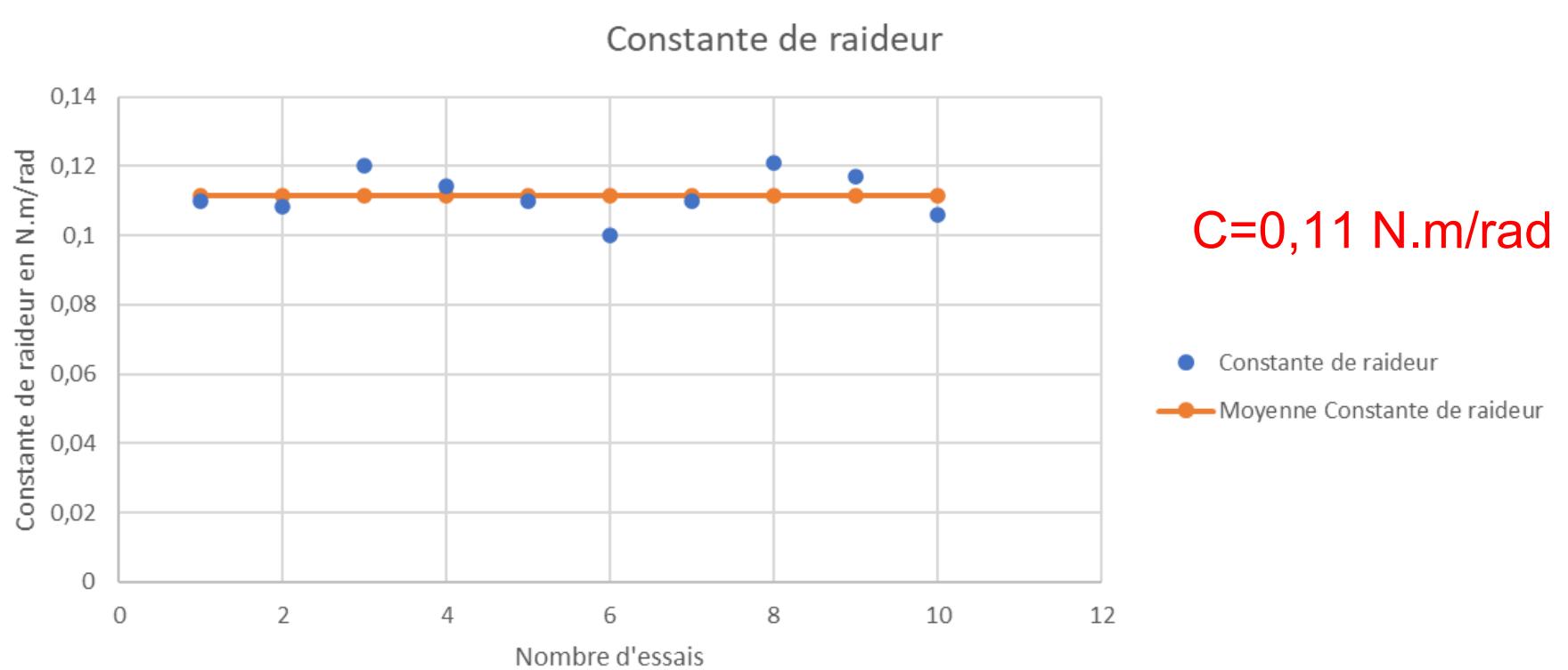
$$\theta = 10^\circ = 0,17 \text{ radians}$$

$$F = 0.30 \text{ N}$$

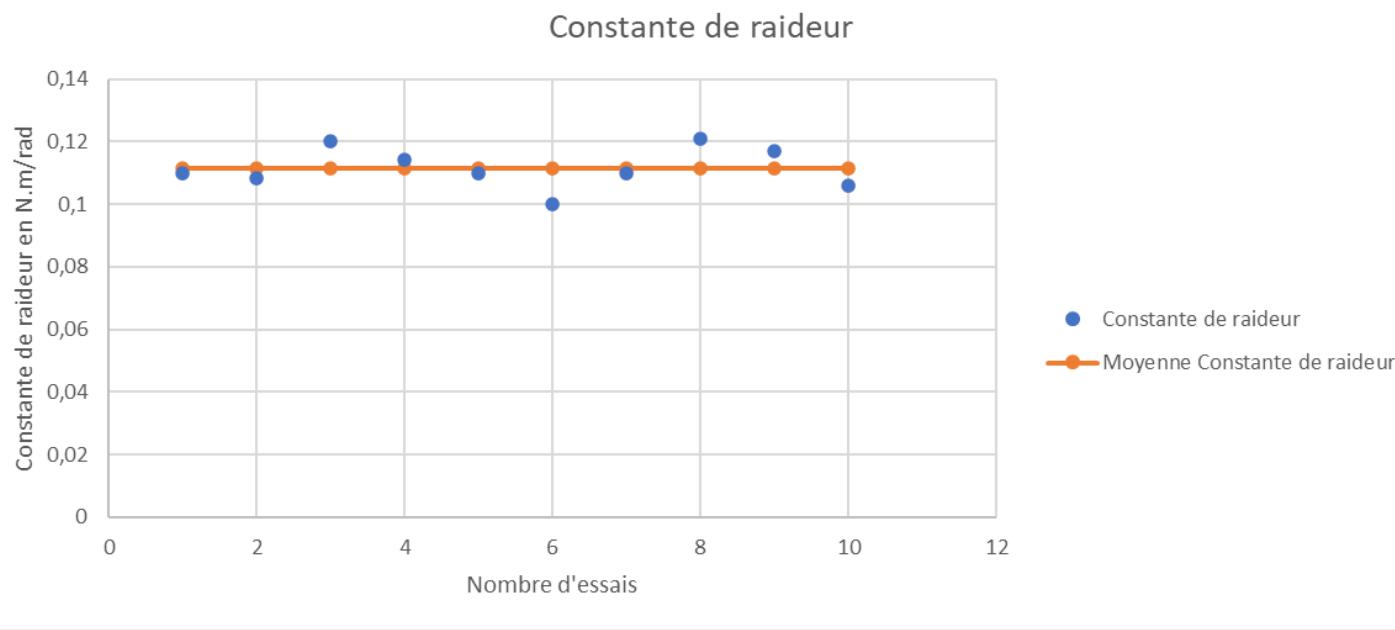
$$C = 0.11 \text{ N.m.rad}^{-1}$$



Première cible : constante de raideur



Première cible : constante de raideur



$$U_A = \frac{\sigma_{n-1}}{\sqrt{n}} = 0,0022$$

$$C = 0.11 \pm 0.0022 \text{ N.m/rad}$$



Première cible : vérifications tir à 7 mètres



BUT**Première cible****Grande cible****Conclusion****Annexe**

Première cible : vérifications tir à 7 mètres



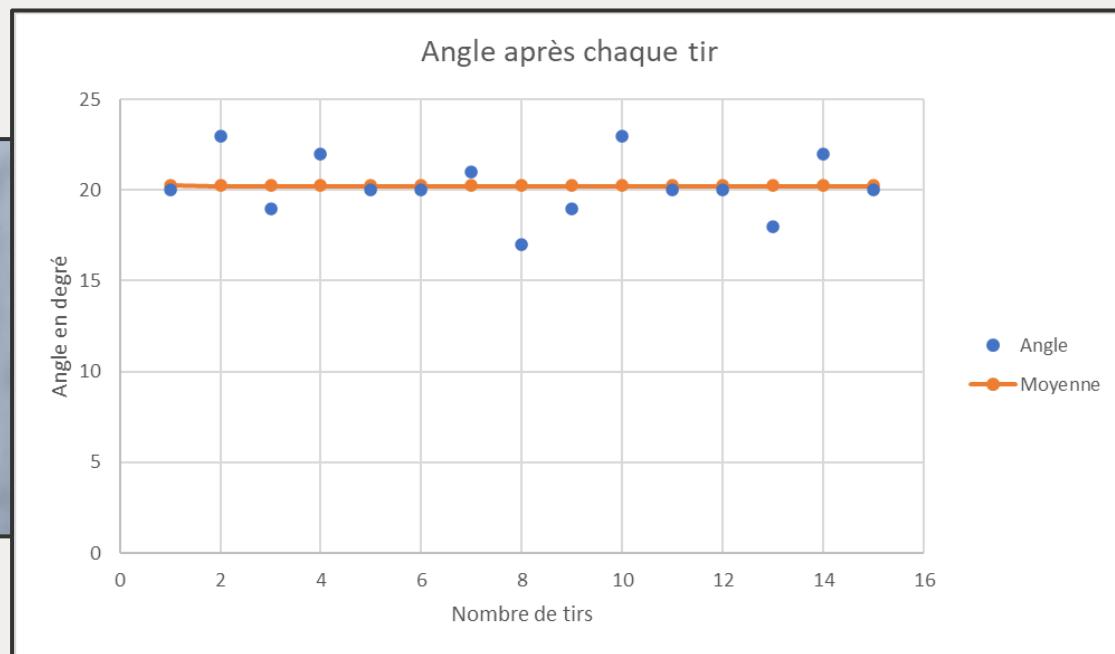
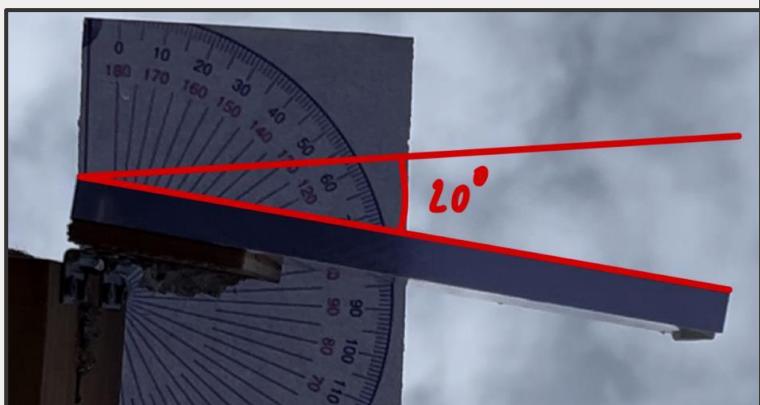
Première cible : vérifications tir à 7 mètres



Première cible : vérifications des résultats

Distance du tir : 7 mètres

Nombre de tirs : 15

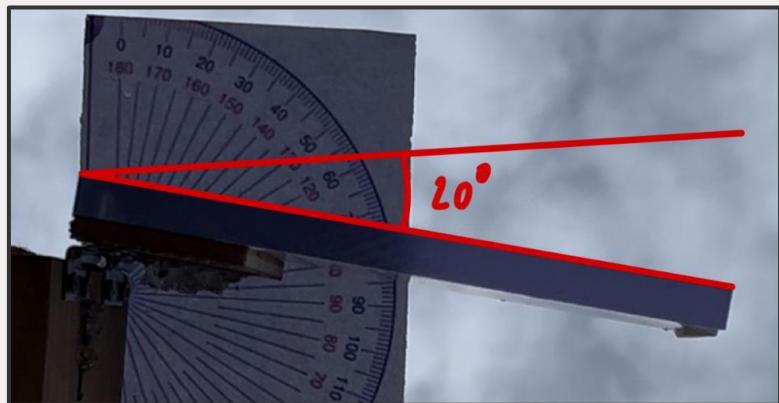


Première cible : vérifications des résultats

Distance du tir : 7 mètres

Nombre de tirs : 15

99,5% d'énergie
non transmise



$$U_A = \frac{\sigma_{n-1}}{\sqrt{n}} = 0,49$$

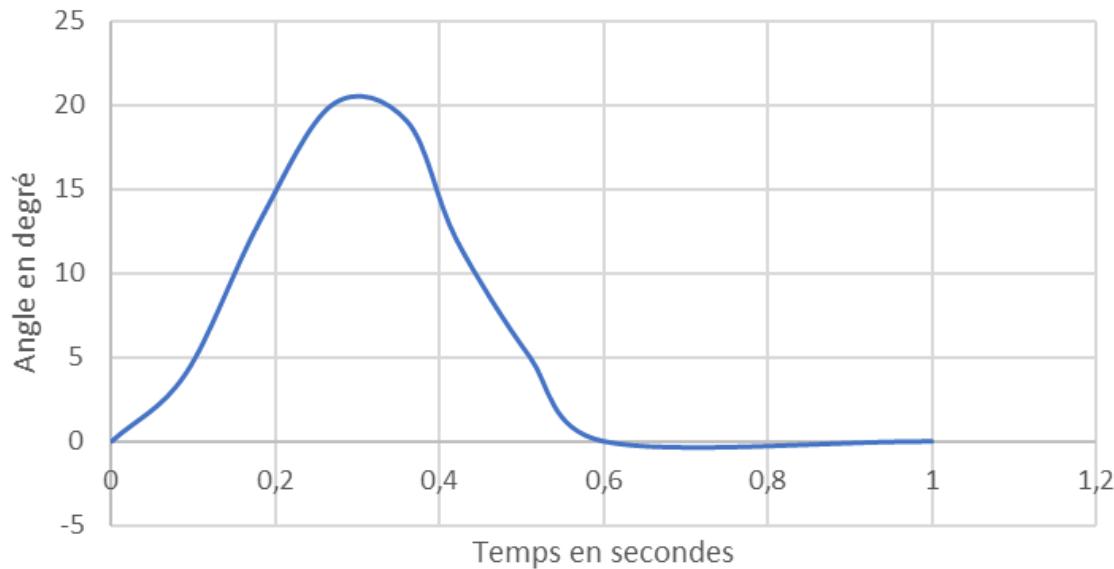
$\theta=20 \pm 0.5$ degrés



Première cible : vérifications des résultats

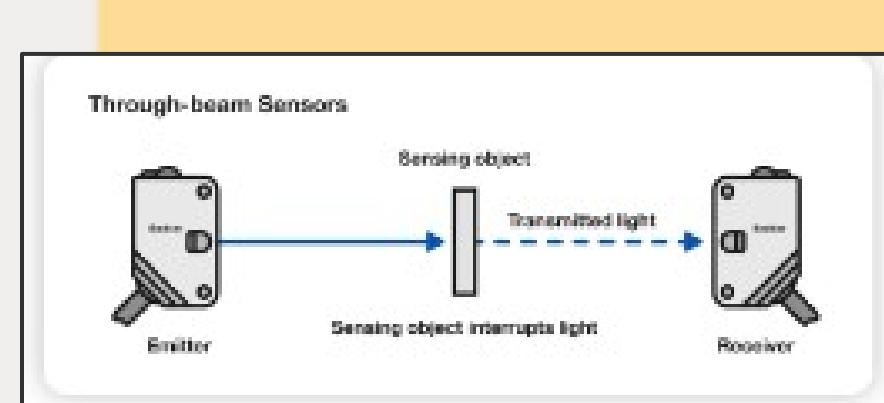
Retour sans oscillations / Négligeables

Position angulaire en fonction du temps



Cellule photoélectrique fonctionnement détaillé

- souvent utilisé en « tout ou rien »
- propriétés varient en fonction de l'intensité lumineuse
- par exemple: la résistance diminue lorsque la lumière passe sur la photorésistance

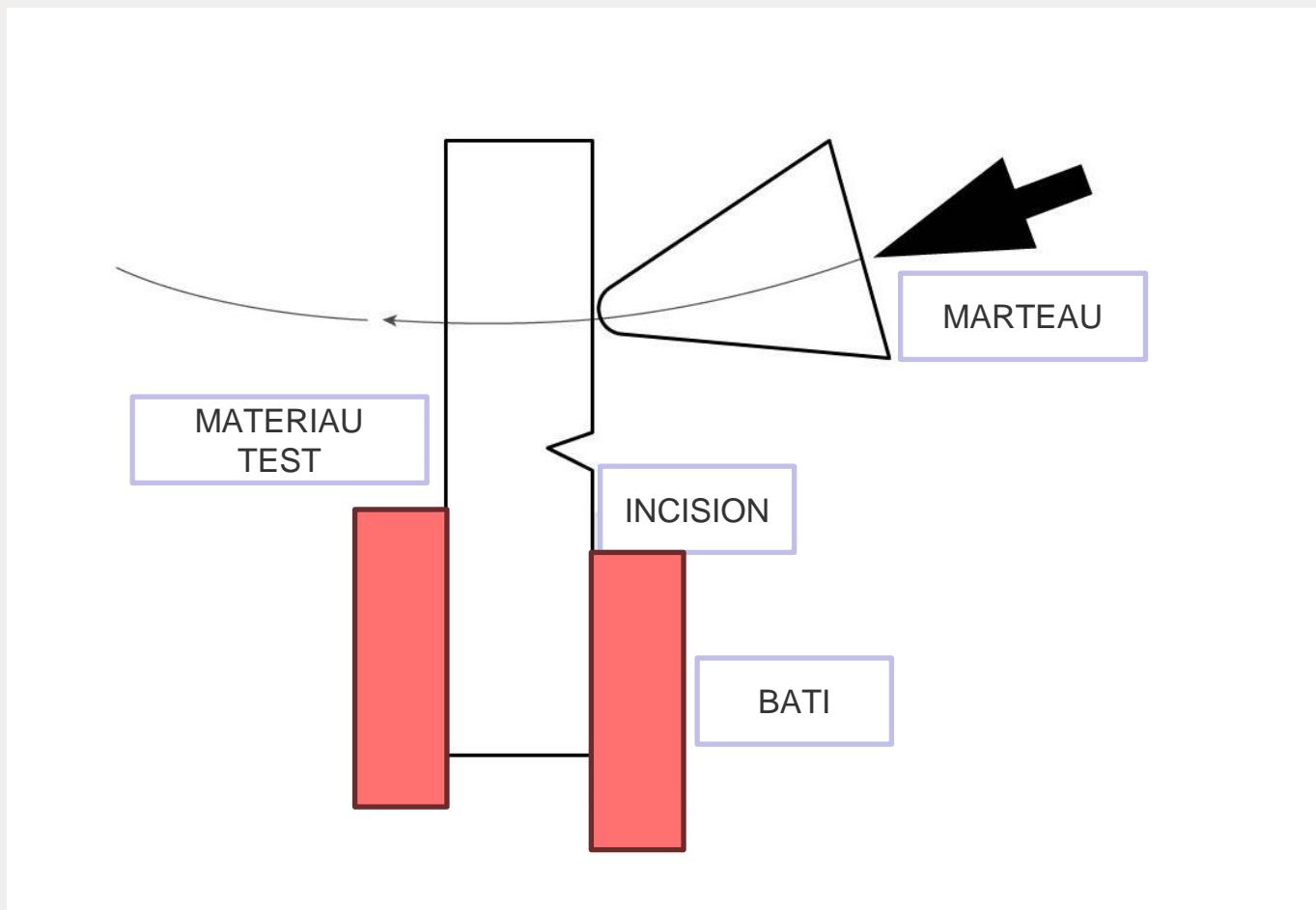


Sc : GSD-automatisme



Résistance aux chocs : modèle d'IZOD

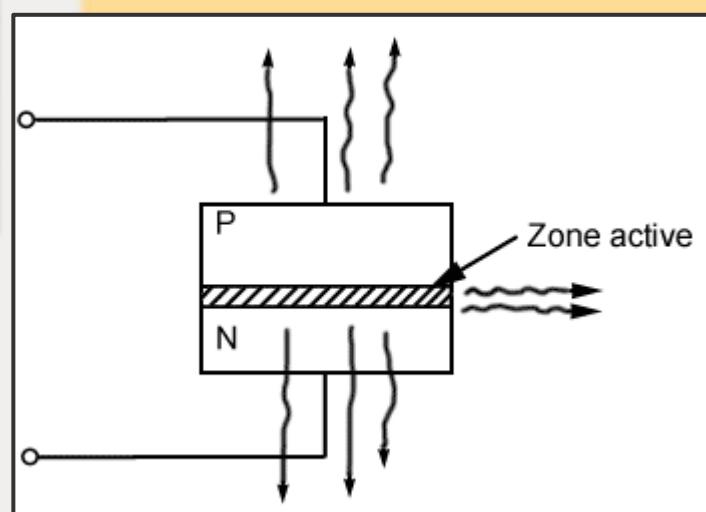
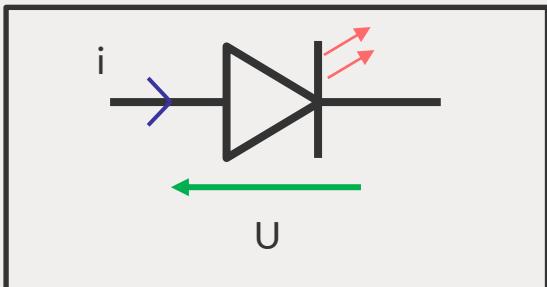
Mesure la quantité totale d'énergie qu'un matériau est capable d'absorber



LED : fonctionnement



| Couleur | Longueur d'onde (nm) | Energie des photons (eV) |
|-------------|----------------------|--------------------------|
| UltraViolet | < 390 | > 3,18 |
| Violet | 390-455 | 2,72-3,18 |
| Bleu | 455-490 | 2,53-2,72 |
| Cyan | 490-515 | 2,41-2,53 |
| Vert | 515-570 | 2,18-2,41 |
| Jaune | 570-600 | 2,06-2,18 |
| Orange | 600-625 | 1,98-2,06 |
| Rouge | 625-720 | 1,72-1,98 |
| InfraRouge | > 720 | < 1,72 |



Règle airsoft

Pourquoi 2 joules max :

- Réduction des risques de blessures
- Différenciation avec les armes à feu

En limitant l'énergie à 2 joules, le risque de blessures graves est significativement réduit.

on assure que les répliques d'airsoft restent distinctes des armes à feu en termes de puissance, facilitant ainsi leur régulation et leur utilisation dans des contextes de loisir.



