ILBER GOKAL

Séance 1: 16/12/2025 : Découverte de l'impression 3D à partir d'un modèle existant - Ghastling Articulated Desk Toy

Objectifs :

- Se familiariser avec les principes de base de l’impression 3D à partir d’un fichier numérique déjà conçu.
- Identifier le rôle du logiciel de slicing et analyser l’impact des principaux paramètres sur le résultat final.
- Comprendre la gestion de densités de remplissage différentes sur un même plateau pour optimiser la légèreté et la solidité.

Présentation du projet : Dans le cadre de ce projet, l’objectif était de réaliser un "Ghastling" (bébé Ghast), un mob issu de la mise à jour Minecraft "Chase the Skies". Contrairement aux modèles précédents, ce projet introduit une dimension mécanique avec des pattes amovibles à clipser ("snap-on"). Ce choix permet de tester la précision dimensionnelle de l'imprimante pour l'assemblage de pièces séparées, ainsi que l'utilisation de réglages avancés comme l'étirage (ironing) pour améliorer la finition de surface.

Matériel et logiciel utilisés :

- **Logiciel de slicing :** PrusaSlicer
- **Imprimante 3D :** Original Prusa MK4S
- **Technologie :** FFF
- **Matériel :** PLA

Processus d'impression : Le modèle du Ghastling a été importé dans PrusaSlicer. Pour garantir un assemblage parfait et une esthétique optimale, les paramètres suivants ont été appliqués :

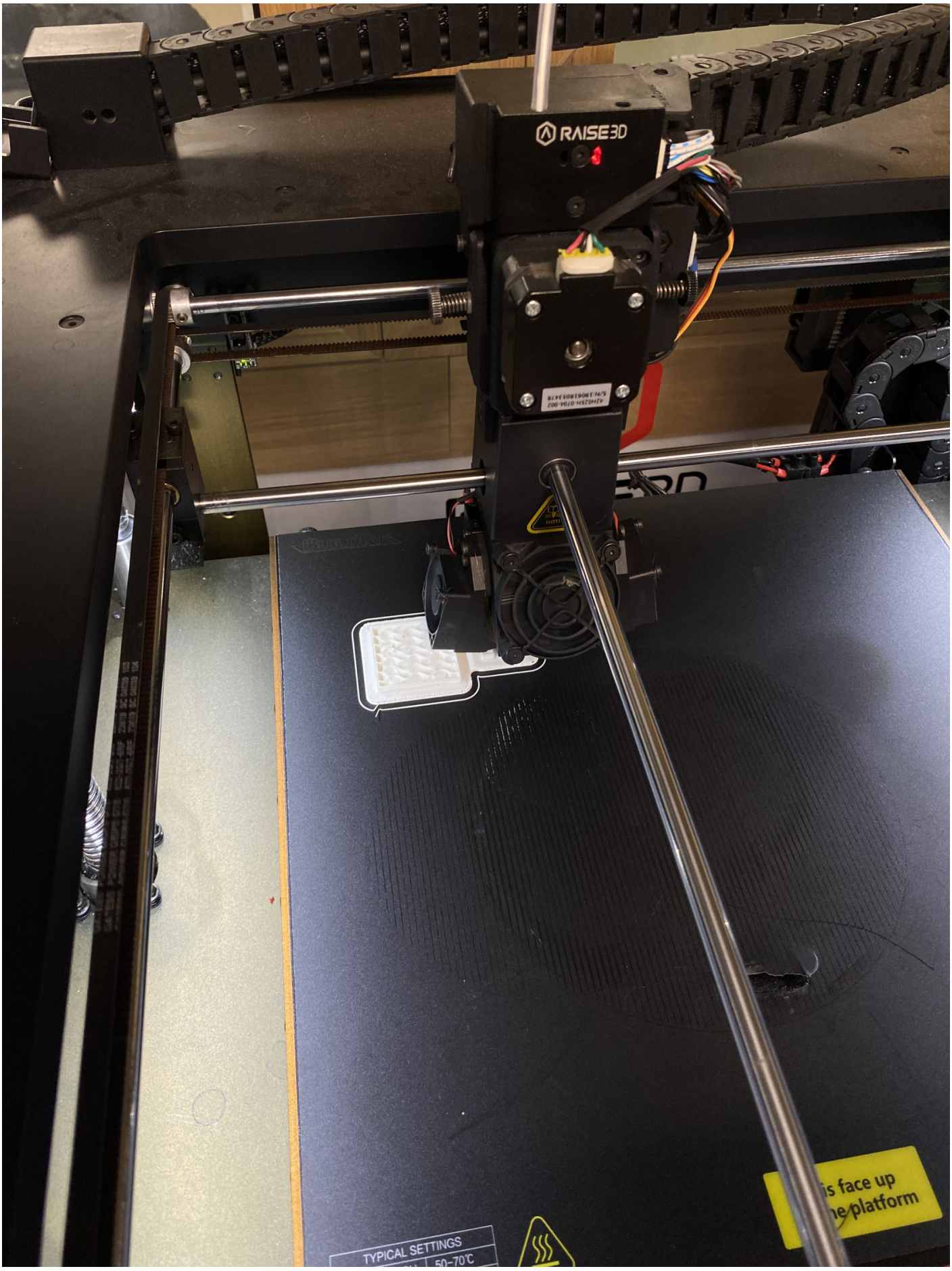
- **Orientation :** Face vers le haut (pour la netteté des détails du visage)
- **Température de la buse :** 200 °C
- **Température du plateau :** 60 °C
- **Taux de remplissage (infill) :** 5 % pour le corps (légèreté) et 30 % pour les pattes (solidité)
- **Adhérence :** Bordure (brim) de 5 mm
- **Supports :** Aucun
- **Option de finition :** Étirage (ironing) activé sur la surface supérieure du visage.
- **Temps d’impression estimé :** environ 1 heure

Résultats et observations : L'impression s'est déroulée sans encombre. L'utilisation de la bordure (brim) a été déterminante pour maintenir les pattes en place durant toute l'opération.

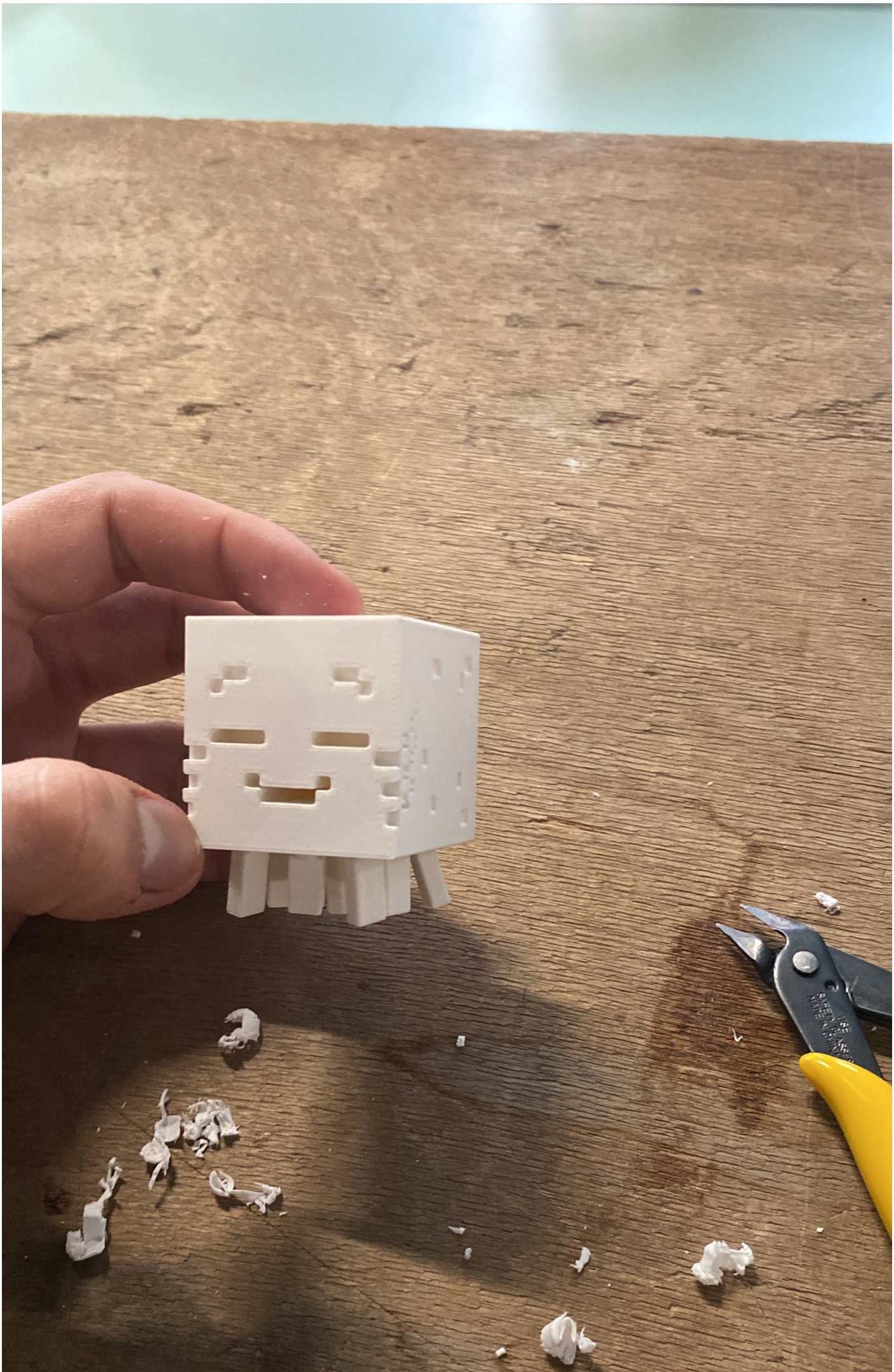
L'option d'étirage (ironing) a permis d'obtenir une surface très lisse sur le dessus de la tête, masquant efficacement les lignes de couches visibles sur les projets précédents. L'assemblage des pattes par clipsage s'est fait sans forcer, validant la précision des réglages et de la machine.

Conclusion : Ce projet a permis d'approfondir la maîtrise de PrusaSlicer en manipulant des densités de remplissage variables. Le Ghastling démontre que l'impression 3D permet non seulement de créer des objets décoratifs, mais aussi des jouets articulés fonctionnels avec une finition de surface proche d'un produit industriel.

Première tentative infructueuse :



Objet terminé :



Exercice 2 : Création d’un objet en mobilisant un maximum de fonctions OpenSCAD
Sifflet à bille (Survival Whistle) – Objet Print-in-Place

Objectifs :

- Utiliser la modélisation paramétrique pour créer une chambre de résonance acoustique.
- Maîtriser les fonctions de soustraction pour créer un conduit d'air (fente) et une sortie (fenêtre).
- Concevoir un élément mobile interne (bille) imprimé directement en position libre (Print-in-Place).
- Ajuster les tolérances pour garantir que la bille se détache après l'impression.

Contexte du projet : Pour cet exercice, j'ai conçu un sifflet monobloc. La difficulté réside dans la partie supérieure : la "lèvre" (labium) doit être assez fine pour diviser le flux d'air et créer un son, tandis que la bille intérieure doit rester parfaitement sphérique sans coller aux parois. Le paramètre clé est la **clearance = 0.4 mm** autour de la bille centrale.

```
// --- PARAMÈTRES (MODIFIABLES) ---
$fn = 100;              // Haute résolution pour l'acoustique
d_exterieur = 20;       // Diamètre du sifflet
largeur_bec = 15;       // Largeur de l'embout buccal
longueur_bec = 20;      // Longueur de l'embout
epaisseur = 1.5;        // Épaisseur des parois
d_bille = 8;            // Diamètre de la bille interne (la "pea")
clearance = 0.4;        // Jeu pour que la bille ne colle pas

union() {
   difference() {
      union() {
         cylinder(d = d_exterieur, h = d_exterieur, center = true);
         translate([longueur_bec/2 + 5, 0, 0])
            cube([longueur_bec, largeur_bec, d_exterieur], center = true);
      }       sphere(d = d_exterieur - (epaisseur * 2));

      translate([longueur_bec/2 + 5, 0, 0])
         cube([longueur_bec + 2, largeur_bec - 4, 3], center = true);



      translate([8, 0, 5])
         rotate([0, -45, 0])
            cube([10, largeur_bec - 2, 10], center = true);
   }    color("Red")
      translate([0, 0, - (d_exterieur/2 - epaisseur - d_bille/2) + clearance])
         sphere(d = d_bille);
}
```

Paramètres principaux :

- Diamètre du sifflet : 20 mm
- Diamètre de la bille interne : 8 mm
- Jeu mécanique (Clearance) : 0.4 mm
- Épaisseur de paroi : 1.5 mm
- Résolution ($fn) : 100 (pour une étanchéité à l'air maximale)

Matériaux / Outils / Logiciels :

- **Modélisation :** OpenSCAD
- **Slicer :** PrusaSlicer
- **Imprimante :** Original Prusa MK4S
- **Filament :** PLA (Orange pour la visibilité)
- **Température buse :** 210 °C (Température réduite pour éviter la fusion de la bille)
- **Infill :** 15%
- **Supports :** Aucun (le sifflet est conçu pour s'auto-supporter)

Étapes de fabrication :

1. Écriture du code OpenSCAD avec une attention particulière sur l'angle de la lèvre (45°).
2. Exportation STL et vérification de l'étanchéité du maillage.
3. Placement vertical sur le plateau dans PrusaSlicer.
4. Lancement de l'impression.
5. Post-traitement : Utilisation d'un petit tournevis pour "libérer" la bille interne collée par les micro-supports de la première couche.

Problèmes rencontrés lors de l’impression : Le principal défi a été le **"Bridging"** (pontage) sur le dessus de la chambre interne. Si le filament s'affaisse trop, il peut tomber sur la bille et la bloquer définitivement.

- **Cause probable :** Vitesse de refroidissement insuffisante sur les couches de fermeture.
- **Solution :** Augmenter la ventilation (fan speed) à 100% lors du passage sur le vide de la chambre.

Améliorations envisagées :

- Ajouter un anneau d'attache (keychain loop) via une fonction `torus` ou `difference` de cylindres.
- Réduire la hauteur de couche à 0.10 mm pour rendre la lèvre du sifflet plus tranchante et le son plus strident.

Conclusion : Ce projet démontre qu'OpenSCAD permet de créer des objets dont la complexité est interne. Le sifflet est un excellent test de **précision pneumatique** : une simple fuite d'air due à une mauvaise fusion des couches rendrait l'objet inutile. C'est l'équilibre parfait entre géométrie simple et fonction mécanique complexe.

Objet terminé :

Voici une réécriture harmonisée de vos exercices Arduino, en conservant la structure rigoureuse de vos comptes-rendus précédents (Objectifs, Matériel, Montage, Code, Analyse).

Exercice 4 : Pilotage de sorties numériques – Clignotement alterné de deux LEDs

Exercice 4 : Pilotage de sorties numériques – Clignotement alterné de deux LEDs

Objectif

L'objectif de cet exercice est de maîtriser le contrôle de plusieurs sorties numériques indépendantes sur une carte Arduino. Contrairement à un clignotement synchrone, ce programme impose un cycle alterné entre deux LEDs, simulant un système de balisage simple.

Matériel utilisé

- **Microcontrôleur :** Arduino Uno
- **Composants :** 2 LEDs, 2 résistances de 220 Ω
- **Interface :** Breadboard et fils de connexion
- **Logiciels :** Arduino IDE & TinkerCAD (simulation)

Montage du circuit

1. **Câblage des anodes :** La LED 1 est connectée à la broche numérique **13** et la LED 2 à la broche **12**.
2. **Protection :** Une résistance de 220 Ω est placée en série sur chaque LED pour limiter l'intensité du courant sous 5 V.
3. **Masse commune :** Les cathodes sont reliées au rail **GND** de l'Arduino.

Code source

C++

```
void setup() {
  pinMode(13, OUTPUT); // Configuration de la LED 1
  pinMode(12, OUTPUT); // Configuration de la LED 2
}

void loop() {
  digitalWrite(13, HIGH); // Allume LED 1
  digitalWrite(12, LOW);  // Éteint LED 2
  delay(1000);            // Pause 1s

  digitalWrite(13, LOW);  // Éteint LED 1
  digitalWrite(12, HIGH); // Allume LED 2
  delay(1000);            // Pause 1s
}
```

Analyse et Résultats

Le cycle de boucle (`loop`) permet un basculement d'état binaire entre les deux broches. L'utilisation de la fonction `delay()` bloque le processeur pendant 1000 ms, ce qui est suffisant pour ce projet mais limite la réactivité du système pour des tâches plus complexes. Le résultat obtenu est une alternance parfaite et régulière des deux sources lumineuses.

Exercice 5 : Acquisition de données et signalisation – Capteur ultrasonique HC-SR04

Exercice 5 : Acquisition de données et signalisation – Capteur ultrasonique HC-SR04

Objectif

Cet exercice vise à intégrer un capteur d'entrée (Input) pour piloter des actionneurs de sortie (Output). Il s'agit de convertir un signal physique (onde sonore) en une donnée numérique (distance en cm) afin de créer un système de signalisation visuelle par paliers.

Matériel utilisé

- **Capteur :** Ultrasonique HC-SR04
- **Signalisation :** 3 LEDs (Verte, Jaune, Rouge) + 3 résistances 220 Ω
- **Microcontrôleur :** Arduino Uno

Montage du circuit

- **Capteur :** VCC (5V), GND (GND), **Trig (A0)**, **Echo (A1)**.
- **LEDs :** Connectées aux broches **11, 12 et 13**.
- **Principe :** Le capteur utilise les broches analogiques A0/A1 configurées ici en mode numérique pour déclencher et lire l'écho.

Algorithme et Calcul

Le système repose sur la vitesse du son dans l'air ($v \approx 343 \text{ m/s}$ ou $0,0343 \text{ cm/µs}$). La distance est obtenue par la formule :

$$d = \frac{t \times 0,0343}{2}$$

(Le diviseur 2 correspond au trajet aller-retour de l'onde).

Code source

C++

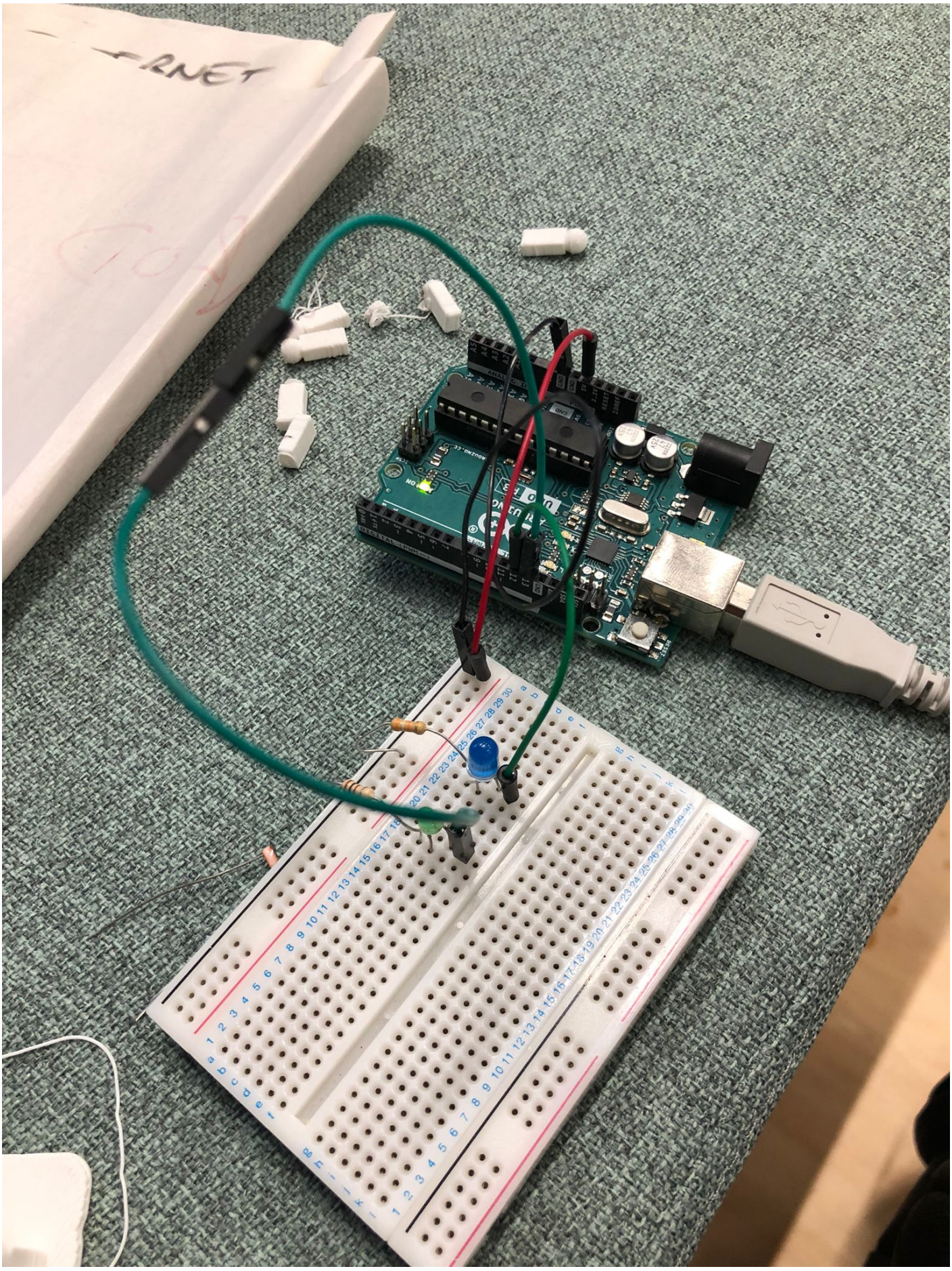
```
long readUltrasonicDistance(int triggerPin, int echoPin) {
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);
}

void setup() {
  pinMode(11, OUTPUT); pinMode(12, OUTPUT); pinMode(13, OUTPUT);
}
```

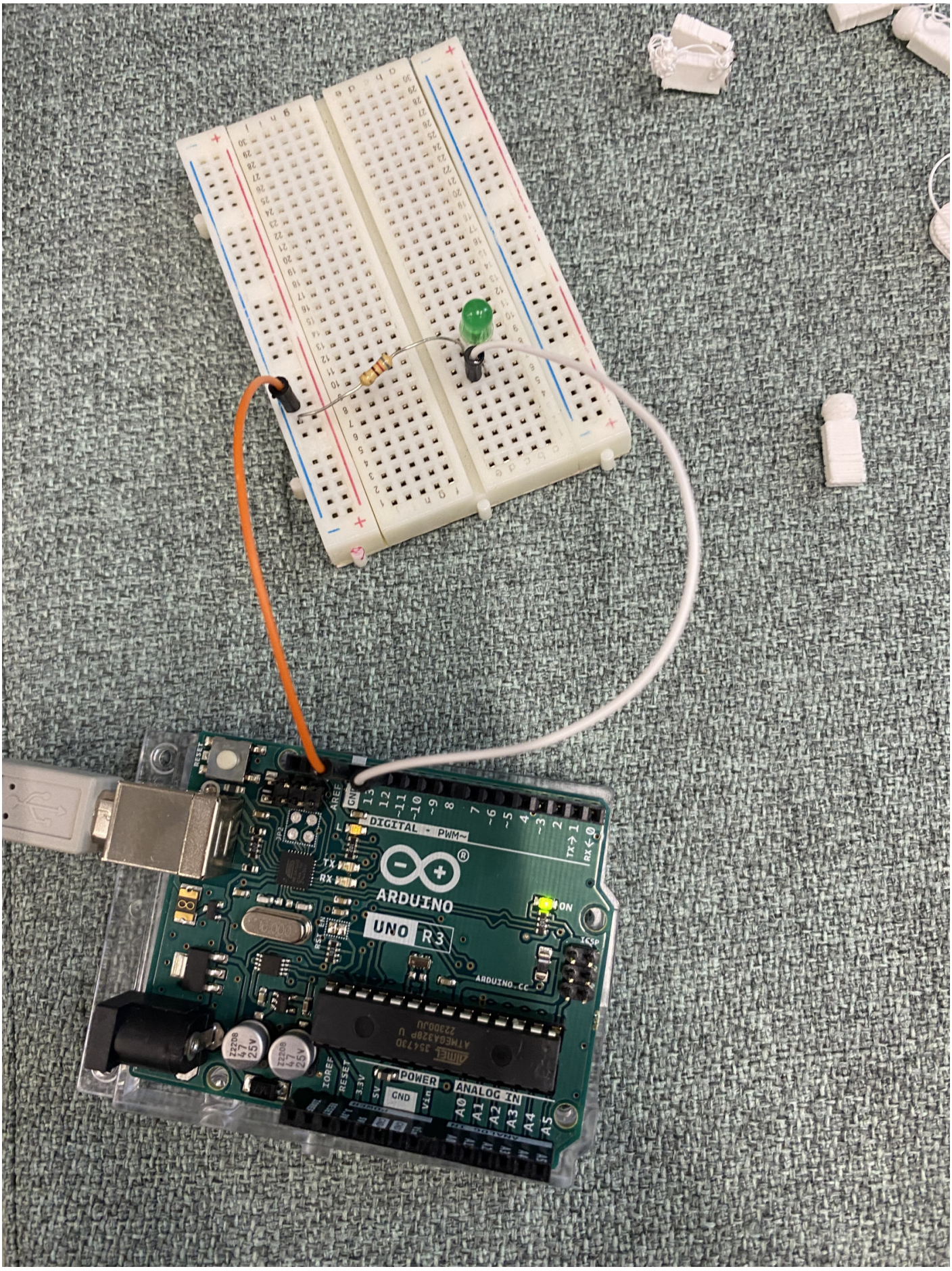
```
void loop() {
  float distance = 0.01723 * readUltrasonicDistance(A0, A1);

  // Logique de signalisation
  digitalWrite(11, (distance > 100) ? HIGH : LOW);
  digitalWrite(12, (distance > 50 && distance <= 100) ? HIGH : LOW);
  digitalWrite(13, (distance > 20 && distance <= 50) ? HIGH : LOW);

  delay(10); // Stabilité de lecture
}
```

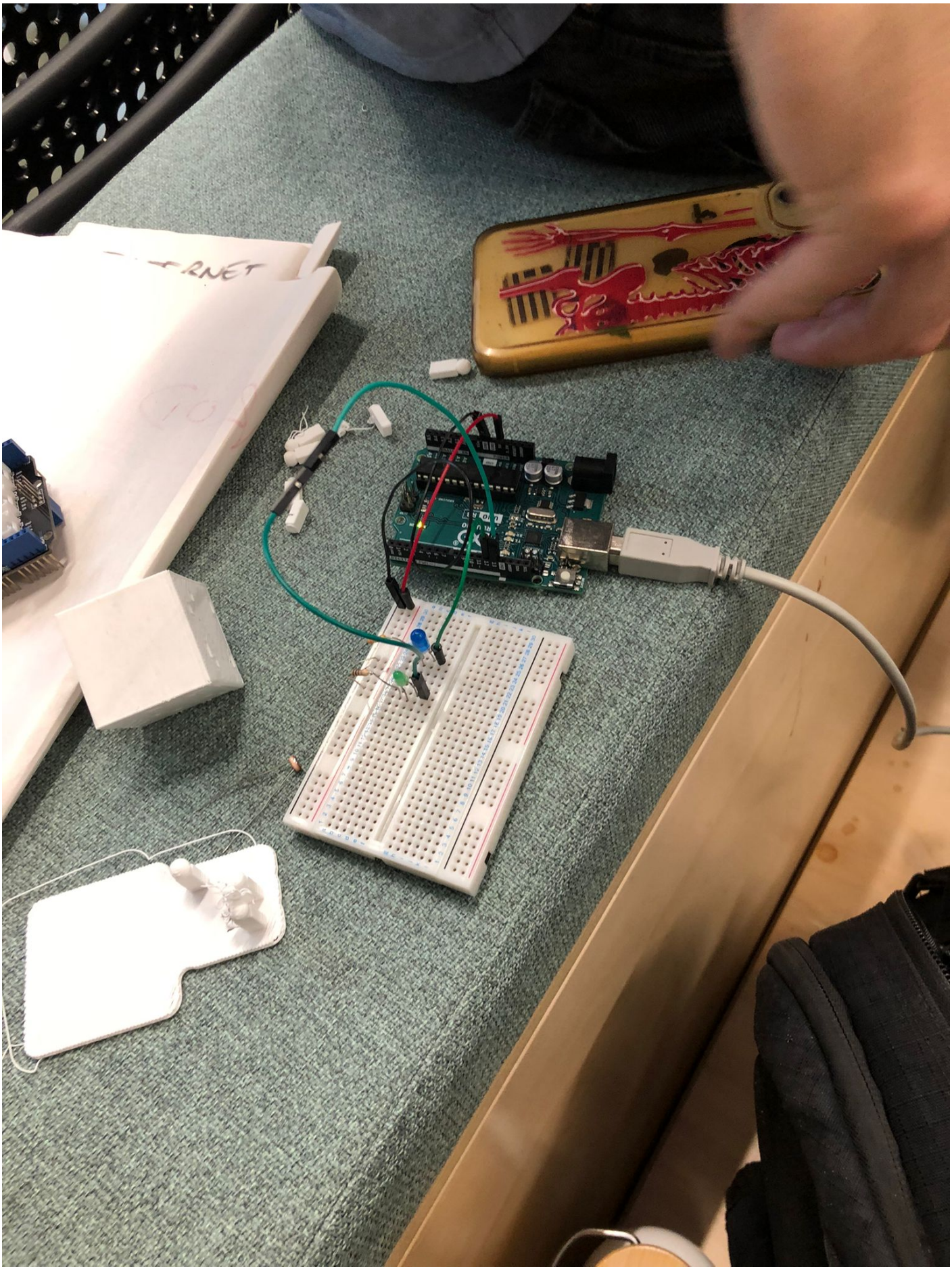
Analyse et Conclusion

La simulation sur TinkerCAD a validé la logique des seuils avant le passage au réel. Le système réagit en temps réel :

1. **Zone de sécurité (> 100 cm) :** LED 1 active.
2. **Zone d'approche (50-100 cm) :** LED 2 active.
3. **Zone critique (20-50 cm) :** LED 3 active.
   Ce projet démontre l'efficacité d'Arduino pour transformer une mesure invisible (ultrasons) en une information visuelle exploitable pour un utilisateur.




ARDUINO
UNO R3
DIGITAL (PWM~)
POWER
ANALOG IN
ARDUINO.CC



Revision #1
Created 1 March 2026 18:50:32 by Gokal Ilber
Updated 1 March 2026 19:12:30 by Gokal Ilber