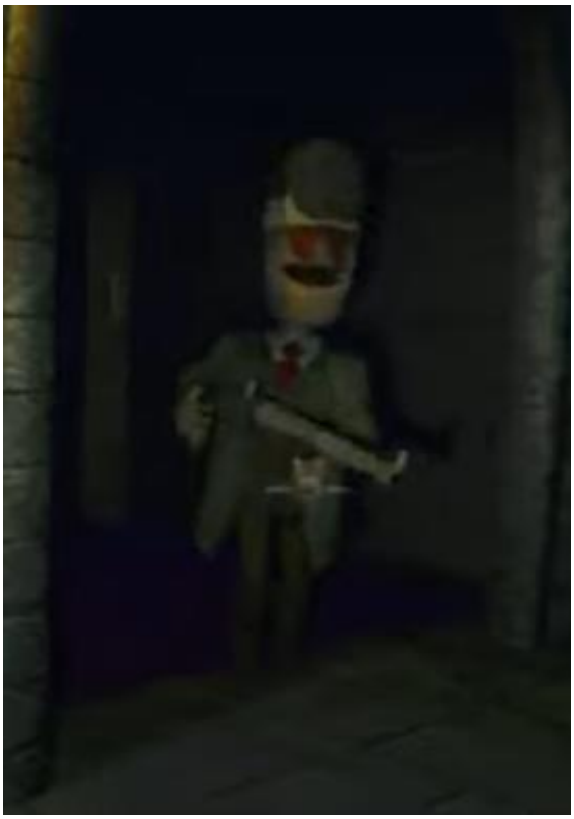
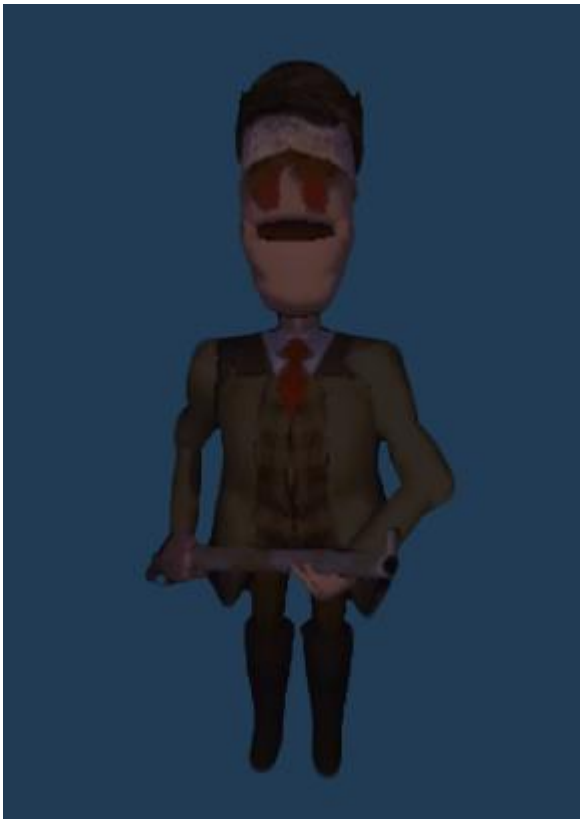
Projet Robotique n°3 – Robot Chasseur R.E.P.O

🎯 Objectif du Projet

L’objectif de ce projet est la conception et la réalisation d’une plateforme robotique inspirée d’un "chasseur" issu de l’univers du jeu *R.E.P.O*. Le système doit être capable de détecter un bruit fort dans l’environnement, d’en estimer la direction, de s’orienter automatiquement vers cette source sonore, puis de déclencher une réponse audiovisuelle. L’architecture matérielle et logicielle est pensée pour être évolutive, permettant l’intégration future de capteurs ou fonctionnalités supplémentaires.

📷 Photo de référence

⚙️ Fonctionnalités Techniques

Le robot intègre les fonctions suivantes :

- **Détection acoustique** : Capacité à détecter un pic sonore dépassant un seuil préétabli à l’aide de quatre microphones analogiques orientés selon les axes cardinal (avant, droite, arrière, gauche).

- **Analyse directionnelle** : Traitement des signaux audio pour estimer la direction dominante d’émission sonore.
- **Réaction motorisée** : Rotation mécanique contrôlée via servomoteur pour s’orienter vers la source détectée.
- **Retour audio** : Lecture contextuelle d’un fichier audio via un module DFPlayer Mini.
- **Réponse visuelle** : Allumage de LEDs directionnelles pour une représentation visuelle de la provenance du son.
- **Architecture modulaire** : Conception facilitant l’ajout d’actuateurs ou de capteurs (ultrasons, infrarouge, Bluetooth...).

🧩 Architecture Matérielle

Composant	Rôle
Arduino Mega 2560	Microcontrôleur principal
4x Microphones MAX4466	Acquisition sonore directionnelle
Servomoteur DS04-NFC	Rotation du châssis ou de la "tête"
DFPlayer Mini + carte SD	Système audio embarqué
Powerbank 5V / 3A	Alimentation autonome pour les modules
Breadboard & câblage Dupont	Prototypage sans soudure

🔌 Schéma de Câblage – Vue Générale

- **Microphones MAX4466**
  - Branchés sur A0 à A3 (avant, droite, arrière, gauche)
  - Alimentés en 5V, masse commune
- **Servomoteur DS04-NFC**
  - Signal : D9 (PWM)
  - Alimentation directe via powerbank (bypass du régulateur Arduino pour éviter les chutes de tension)
- **DFPlayer Mini**
  - Communication série logicielle (D10 RX, D11 TX)
  - Alimentation directe depuis la powerbank
- **LEDs directionnelles**
  - Connectées sur D4 à D7 (sortie digitale), protégées par résistances de 220–330Ω
- **Alimentation**
  - Utilisation d’un câble adaptateur JZK (USB-A → Dupont) pour extraire du 5V propre depuis une powerbank (courant max 3A)

🧠 Architecture Logicielle

Le système repose sur une logique événementielle simple mais robuste :

```
scss
CopierModifier
[Entrée analogique microphones]
        ↓
Analyse de pic sonore (30 échantillons)
        ↓
Détermination de la direction principale
        ↓
Contrôle du servomoteur vers la direction détectée
        ↓
Déclenchement audio via DFPlayer Mini
        ↓
Retour à la position initiale (réinitialisation mécanique)
```

Le code est développé en C++ via l'environnement Arduino IDE. Il inclut une calibration des niveaux de repos pour chaque microphone afin d'améliorer la robustesse face au bruit ambiant.

🧪 Algorithme de Détection Directionnelle

1. Acquisition de 30 échantillons par microphone
2. Détermination du pic maximal par micro
3. Calcul de la différence par rapport à un niveau de repos pré-calibré
4. Détection du micro avec le delta le plus élevé (source sonore dominante)
5. Contrôle moteur basé sur un mapping simple direction ↔ durée PWM (calibrée manuellement)

L'approche, bien que rudimentaire, permet une reconnaissance directionnelle grossière (avant/droite/arrière/gauche) avec une latence faible.

💻 Code Source – Détection Directionnelle et Contrôle Servo

```
#include <Servo.h>

// === Broches des microphones ===
#define micFront  A0
#define micRight  A1
#define micBack   A2
#define micLeft   A3

// === Niveau de repos mesuré pour chaque micro ===
#define baseFront 336
#define baseRight 336
```

```
#define baseBack  336
#define baseLeft  350

// === Seuil de déclenchement ===
#define seuilRelatif 90

// === Durées calibrées pour rotation servo (en ms) ===
#define ROTATION_90_GAUCHE 560
#define ROTATION_90_DROITE 580
#define ROTATION_180        1140

// === Servo rotatif 360° ===
Servo headServo;
const int servoPin = 9;

void setup() {
  Serial.begin(9600);
  headServo.attach(servoPin);
  arreterServo(); // position initiale = arrêt
  Serial.println("Servo prêt (face AVANT contre butée gauche)");
  delay(1000);
}

void loop() {
  // Lire les valeurs des 4 micros
  int valFront = lirePic(micFront);
  int valRight = lirePic(micRight);
  int valBack  = lirePic(micBack);
  int valLeft  = lirePic(micLeft);

  int ecartFront = valFront - baseFront;
  int ecartRight = valRight - baseRight;
  int ecartBack  = valBack  - baseBack;
  int ecartLeft  = valLeft  - baseLeft;

  // === Affichage des mesures micro ===
  Serial.print("📊 Micros - AVANT: "); Serial.print(valFront);
  Serial.print(" | DROITE: "); Serial.print(valRight);
  Serial.print(" | ARRIERE: "); Serial.print(valBack);
  Serial.print(" | GAUCHE: "); Serial.println(valLeft);

  // Détection d'un son fort
  if (ecartFront > seuilRelatif || ecartRight > seuilRelatif ||
      ecartBack > seuilRelatif  || ecartLeft  > seuilRelatif) {

    int deltas[4] = {ecartFront, ecartRight, ecartBack, ecartLeft};
    const char* directions[4] = {"AVANT", "DROITE", "ARRIERE", "GAUCHE"};

    // Identifier la direction dominante
```

```
int maxIdx = 0;
for (int i = 1; i < 4; i++) {
  if (deltas[i] > deltas[maxIdx]) {
    maxIdx = i;
  }
}

Serial.print("🔊 Son fort détecté depuis : ");
Serial.println(directions[maxIdx]);

if (maxIdx == 0) {
  Serial.println("🎯 Déjà orienté vers l'avant. Pas de mouvement.");
} else {
  int angle = 0;
  const char* dir = "";

  // === Tourner vers la direction du bruit ===
  if (maxIdx == 1) { // DROITE
    angle = 90;
    dir = "DROITE";
    Serial.print("🔁 Rotation : ");
    Serial.print(angle);
    Serial.print("° vers ");
    Serial.println(dir);
    tournerDroite(ROTATION_90_DROITE);
  } else if (maxIdx == 2) { // ARRIERE
    angle = 180;
    dir = "ARRIERE";
    Serial.print("🔁 Rotation : ");
    Serial.print(angle);
    Serial.print("° vers ");
    Serial.println(dir);
    tournerDroite(ROTATION_180);
  } else if (maxIdx == 3) { // GAUCHE → contourner par 3×90° droite
    angle = 270;
    dir = "GAUCHE (via droite)";
    Serial.print("🔁 Rotation : ");
    Serial.print(angle);
    Serial.print("° vers ");
    Serial.println(dir);
    tournerDroite(ROTATION_90_DROITE * 3);
  }

  Serial.println("⏳ Attente 3 sec dans cette direction...");
  delay(3000);

  // === Retour à la position initiale contre la butée gauche ===
  Serial.println("🔙 Retour à l'avant (butée gauche)");
```

```
      if (maxIdx == 1) {
        tournerGauche(ROTATION_90_GAUCHE + 400);
      } else if (maxIdx == 2) {
        tournerGauche(ROTATION_180 + 400);
      } else if (maxIdx == 3) {
        tournerGauche((ROTATION_90_GAUCHE * 3) + 400);
      }

      Serial.println("✔ De nouveau face AVANT (contre butée)");
      arreterServo();
      Serial.println("⏸ Pause de 5 secondes avant de réécouter...");
      delay(5000);
    }
  }

  delay(10); // boucle rapide
}

// === Lecture du pic max sur 30 échantillons ===
int lirePic(int pin) {
  int maxVal = 0;
  for (int i = 0; i < 30; i++) {
    int val = analogRead(pin);
    if (val > maxVal) maxVal = val;
    delayMicroseconds(300);
  }
  return maxVal;
}

// === Contrôle du servo rotatif ===
void tournerDroite(int duree) {
  headServo.write(60);     // sens horaire
  delay(duree);
  arreterServo();
}

void tournerGauche(int duree) {
  headServo.write(120);      // sens antihoraire
  delay(duree);
  arreterServo();
}

void arreterServo() {
  headServo.write(90);      // arrêt
  delay(50);
}
```

🧭 Réglages et Calibration

- **Seuil de détection** : fixé à un écart de +90 unités analogiques
- **Durée de rotation servo** : mesurée empiriquement (90° → ~580ms)
- **Retour à zéro** : système prévu pour revenir contre une butée mécanique à chaque cycle

---

🌱 Améliorations Possibles

- **Filtrage numérique** : Ajout d'un passe-haut numérique ou d'une moyenne mobile pour filtrer les bruits parasites
- **Interpolation angulaire** : Passage à une estimation continue de l'angle (au lieu de 4 directions fixes)
- **Communication sans fil** : Ajout d'un module Bluetooth HC-05 ou ESP8266 pour contrôle distant ou télémétrie

---

💰 Budget Prévisionnel

Composant	Estimé	Réel	Source
Arduino Mega 2560	20€	18€	Amazon
Microphones x4	12€	12€	Amazon
DFPlayer Mini	5€	0€	En stock
Servomoteur DS04	9€	13€	Amazon
Powerbank	8€	8€	Action
Câble JZK USB → Dupont	5€	6,50€	Amazon
Breadboard, câbles	5€	0€	En stock
Matériaux châssis	15€	13€	Castorama
Total	**~80€**	**70€**	—

✅ Conclusion

Le projet R.E.P.O démontre une intégration réussie de modules électroniques grand public dans un système embarqué réactif et évolutif. L'utilisation d'un microcontrôleur performant, d'une alimentation adaptée et d'un agencement modulaire permet d'envisager diverses applications pédagogiques ou interactives. Ce prototype peut servir de base à un robot sentinelle, un assistant domotique réactif, ou un système de démonstration pour la perception acoustique directionnelle