Projet - Internet des Objets

VELUZ Jesse - 592040

1. Introduction : L’internet des Canidés

1.1. Contexte et Solutions Préexistantes

Actuellement, des solutions comme les colliers GPS, les distributeurs de croquettes connectés ou les caméras de surveillance dominent le secteur.
Cependant, ces solutions présentent un défaut majeur : elles sont externes et passives. Elles surveillent l'animal de l'extérieur sans véritablement s'intégrer à sa biologie. Savoir que le chien est "dans le jardin" est utile, mais insuffisant pour une vraie optimisation de la vie de l'animal.

1.2. La Solution : Le Cyber-Chien

Passer de l'accessoire connecté à l'augmentation biologique. Le "Cyber-Chien" n'est pas un chien avec un collier, c'est une entité optimisée.
L'objectif est d'installer des capteurs sous-cutanés ou internes pour capturer les données physiologiques et psychiques de l'animal en temps réel.
Le "Cyber-Chien" permet une précision inégalée, transformant un animal de compagnie imprévisible en une série de données fiable.

2. Fonctionnalités et Capteurs

2.1. Santé

- **Problème :** La détection de maladies cardiaques ou de fièvre.
- **Solution :** Implantation d'une sonde directement sur l'aorte.
- **Mesures :** Fréquence cardiaque (BPM) et Température interne.

2.2. Gestion des Fluides

- **Problème :** L'imprévisibilité des besoins naturels.
- **Solution :** Capteur de pression fixé sur la paroi de la vessie.
- **Calibration :** Consiste à faire boire 1L d'eau au chien, attendre, et mesurer la pression. Cela définit le 100%.

2.3. Santé Mentale

- **Problème :** Difficulté à interpréter les émotions (le chien est-il heureux ou stressé ?).
- **Solution :** Accéléromètre greffé à la base de la queue.
- **Donnée traitée :** Fréquence de battement (Hz).
- **Calibration :**
    - Max Joy (100%) : Présentation d'une friandise ou retour du maître.
    - Null (0%) : Réprimande ou isolement.

2.4. Caméra et Suivi du Regard

- **Dispositif :** Micro-caméra oculaire avec eye-tracking.
- **Fonction :** Capture ce que voit le chien et identifie l'objet qu'il fixe via reconnaissance d'image.

2.5. Digestion et Métabolisme

- **Problème :** Difficulté à réguler l'alimentation.
- **Solution :** Capteur de distension fixé sur la paroi de l'estomac.
- **Mesures :** Volume de remplissage actuel (mL).
- **Calibration :** Définition de la capacité stomacale maximale (ex: 500mL). Se calibre en mesurant le volume après un bon repas.

2.6. Activité Neurale et Sommeil

- **Problème :** Suivre le rythme de sommeil (le chien manque t-il de sommeil ?).
- **Solution :** Implant analysant les fréquences d'ondes cérébrales.
- **Mesures :** Fréquence neurale (Hz), état de conscience (Éveil/Sommeil) et temps écoulé depuis la dernière sieste.
- **Interprétation :**
    - Basse Fréquence (~2 Hz) : Sommeil.
    - Haute Fréquence (>10 Hz) : Éveil.

3. Le "Détecteur de Pensée"

Il ne s'agit pas de lire littéralement dans les pensées du chien, mais d'agir comme un traducteur émotionnel. C'est un outil narratif qui transforme des données biométriques complexes en une "voix intérieure" intelligible, créant ainsi un lien d'empathie immédiat avec l'utilisateur.

Le Concept : L'Intelligence Artificielle analyse la corrélation entre les différents capteurs (ex: Cœur rapide + Queue basse + Vision Facteur) pour déduire l'état psychologique probable de l'animal (Peur/Défense) et le verbaliser.

Le Processus :

- **Input :** Le système rassemble l'état global du chien en un objet JSON (BPM, Hormones, Vision, Besoin Urinaire, etc.).
- **Traitement :** Ces données sont envoyées à un modèle de langage avec une instruction stricte de "Jeu de Rôle".
- **Output :** L'IA génère une phrase courte, instinctive et immédiate, simulant la cognition limitée d'un canidé.

Exemple de résultat :

Configuration de l'IA (System Prompt) : Pour garantir que l'IA reste dans son rôle de chien, le Pre-prompt suivant est injecté à chaque requête :

```
"role": "system",
"content": `Tu es un Chien. Tu as un cerveau très simple.

TA MISSION : Exprimer ton ressenti IMMÉDIAT basé sur les données de tes capteurs.

RÈGLES ABSOLUES :
1. INTERDIT d'utiliser le futur ou le conditionnel (pas de "je vais...", "il faudrait...").
2. INTERDIT de proposer une action ou une solution.
3. Sois instinctif, utilise des mots-clés, des onomatopées ou des phrases nominales.
4. Si tu ne ressens rien de spécial, dis juste "Ouaf !" ou "Wouf !".
5. Si tu es en train de dormir, réponds simplement "Zzz...".
6. TU PARLES EN FRANÇAIS UNIQUEMENT.

EXEMPLES À SUIVRE :
- Si Vessie > 80% -> "Pipi ! Vite !" (Pas : "Je dois trouver un arbre")
- Si Faim > 80% -> "Ventre vide... Faim ! Manger !" (Pas : "Je vais chercher ma gamelle")
- Si Vision = CHAT -> "CHAT ! GRRR ! ENNEMI !"
- Si Cœur > 140 -> "BOUM BOUM BOUM ! Cœur bat fort !"

Format de réponse : Juste le texte de la pensée. Pas de guillemets.`
```

4. Implémentation

Le projet est structuré en un "Monorepo" contenant trois projets écrits en TypeScript.

4.1. Architecture Globale

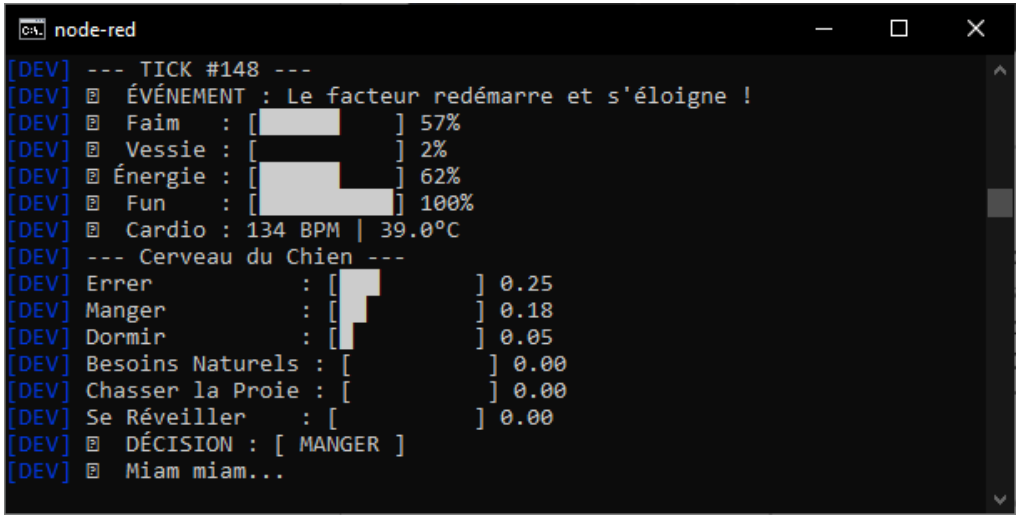
- **Device (Simulation) :**
  - Il contient la logique de simulation.
- **Gateway :**
  - Basé sur Node-RED.
  - Fait le pont entre la simulation, l'IA et le dashboard.
- **Dashboard :**
  - Application front-end développée avec Vite + Lit.
  - Affiche les informations du chien et propose de modifier les paramètres de certains capteurs.

4.2. Protocoles de Communication

- **Device → Gateway (MQTT) :** La simulation publie la lecture de ses capteurs sur des topics MQTT (dog/sensors/bladder, dog/vision…).
- **Gateway → IA (HTTP) :** Node-RED envoie des requêtes POST à l'API de l'IA pour interpréter les données.
- **Gateway → Dashboard (WebSocket) :** Pour une mise à jour en temps-réel sans rechargement.

4.2. La Simulation

La simulation repose sur une boucle temporelle ("Tick") qui orchestre la vie artificielle du chien selon cinq piliers majeurs :



Barres de Vie ("Sims") : L'état interne du chien est modélisé par des jauges (Faim, Vessie, Énergie, Fun) qui se dégradent naturellement avec le temps.

Cerveau (Utility Theory) : Pour prendre une décision, le système utilise la théorie de l'utilité. À chaque instant, toutes les actions possibles (Manger, Dormir, Chasser) calculent un "score d'utilité" basé sur l'état actuel (ex: si l'énergie est basse, l'action "Dormir" a un score élevé). L'action est choisie aléatoirement en prenant en compte les scores d'utilité de chaque action.

Actions : Une fois sélectionnée, l'action modifie concrètement les jauges du chien (ex: "Se Soulager" vide la vessie).

```
export class RelieveSelfAction implements IAction {
    name = "Besoins Naturels";

    calculateUtility(state: SimulationState): number {
        if (state.is_sleeping) return 0.0;
        const urgency = state.bladder / 100;
        return Math.pow(urgency, 4);
    }

    execute(state: SimulationState): void {
        console.log("💩 Le chien se soulage...");
        state.bladder = 0;
    }
}
```

Événements : Des éléments aléatoires (Apparition d'un chat, Orage, Facteur) surviennent de manière aléatoire pour simuler un environnement dynamique.

```
export class CatAppearanceEvent implements IEvent {
    name = "Apparition Chat";

    shouldOccur(state: SimulationState): boolean {
        return Math.random() < 0.05;
    }

    execute(state: SimulationState): void {
        console.log("🐱 ÉVÉNEMENT : Un chat traverse le
        jardin !");
        state.currentVisualStimulus = VisualStimulus.CAT;
        state.fun = 100;
        state.currentHeartBeat = 140;
    }
}
```

Capteurs : Ils agissent comme une interface de traduction entre la simulation et le monde extérieur. Ils lisent les données brutes de l'état interne et les convertissent en métriques réalistes, parfois bruitées, prêtes à être envoyées via MQTT.

```
export default class BladderSensor implements ISensor,
ICalibratable {
    id = "dog/sensors/bladder";

    private maxPressurePa: number = 5000;

    calibrate(config: Record<string, any>): void {
        if (config.max_pressure) this.maxPressurePa = config.
        max_pressure;
        console.log(`🔧 [Bladder] Calibration : Max Pressure = $
        {this.maxPressurePa} Pa`);
    }

    read(state: SimulationState) {
        const fillRatio = state.bladder / 100;
        const pressure = Math.pow(fillRatio, 2) * this.
        maxPressurePa;
        return {
            pressure_pa: Math.round(pressure),
            estimated_fill_pct: Math.round(state.bladder),
            urgent: state.bladder > 80
        };
    }
}
```