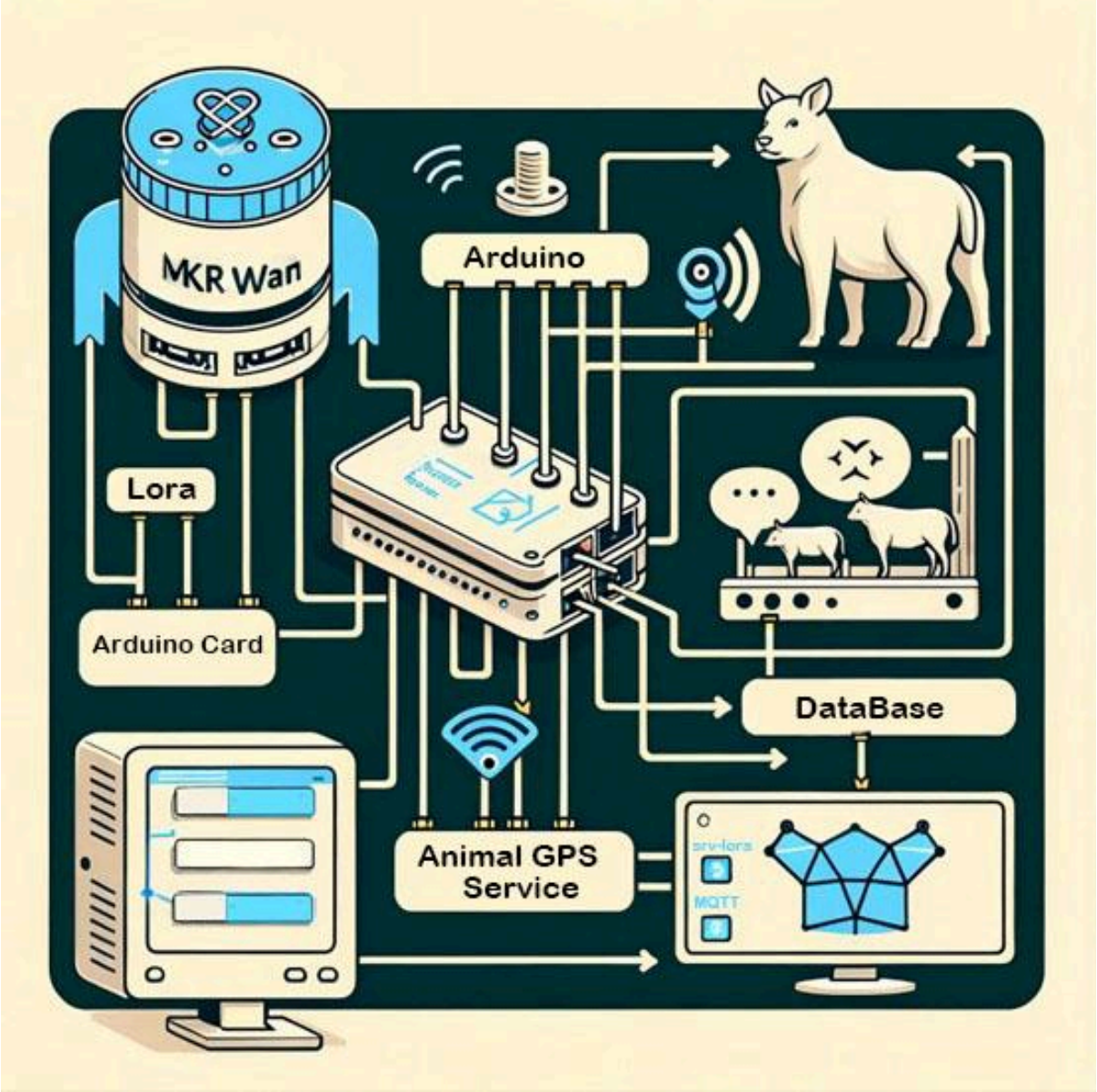
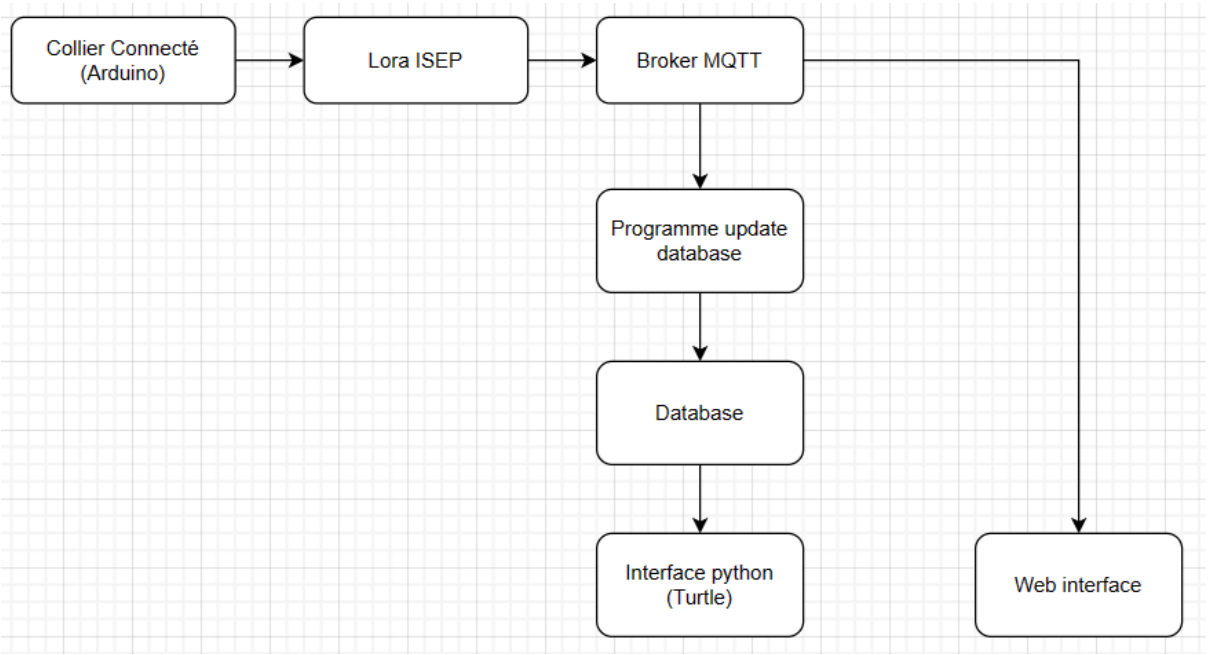
Projet - (Année 2023-2024)
Spécification_info_S4

Mouzheng LI
Lianghong LI
Guokuang DENG
Zhengdong XU



Description du Projet



Le but de ce projet est de développer un prototype pour l'identification, le positionnement et, éventuellement, le suivi de l'état de santé des animaux tels que les chats, les chiens, les chevaux, et les vaches. Ce prototype comprendra un collier connecté, utilisant une carte Arduino MKRWAN, et un tableau de bord pour afficher en temps réel les données reçues. Le système alertera également lorsque les animaux sortent d'une zone prédéfinie ou, potentiellement, lorsqu'ils subissent un niveau de stress élevé.

En détail, notre projet utilise un système de suivi pour les animaux nommé "Panda", qui intègre un collier connecté avec une carte Arduino MKRWAN pour monitorer des animaux tels que les chats, chiens, et chevaux. Ce système enregistre et transmet les coordonnées X et Y de l'animal, ainsi que sa température, à travers le réseau LoRa d'ISEP vers un serveur MQTT nommé "srv-lora".MQTT est un protocole de messagerie léger de publication/abonnement pour les applications IoT et de capteurs qui transfère efficacement les messages entre les appareils. Les données sont stockées dans une base de données SQLite animal_tracking.db et affichées via un tableau de bord développé en Python. Ce dashboard utilise les bibliothèques turtle pour l'affichage graphique et tkinter pour les alertes si un animal quitte une zone prédéfinie. Il existe deux versions du tableau de bord : une version de base qui affiche les informations dans une interface graphique simple et une version avancée sous forme de page web pour une visualisation plus riche et interactive.

Architecture Technique

Collier Connecté : Chaque animal porte un collier équipé d'une carte Arduino MKRWAN qui recueille les données de position (coordonnées X et Y) et de température.

Réseau LoRa : Le collier transmet les données collectées via le réseau LoRa, optimisé pour les communications à longue portée et à faible consommation d'énergie. On utilise réseau LoRa d'ISEP : "srv-lora.isep.fr"

Serveur MQTT (srv-lora) : Le serveur reçoit les données du collier. Ce serveur utilise le protocole MQTT pour la gestion efficace des messages entre les dispositifs IoT.
MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie léger de publication/abonnement conçu pour les applications IoT dans des environnements de réseau à faible bande passante, à latence élevée ou instables. Il s'agit d'un protocole basé sur un modèle client-serveur, où le client peut être un capteur, un appareil ou une application, et le serveur est l'intermédiaire responsable de la réception et de la distribution des messages.

Base de données SQLite (animal_tracking.db) : Les données reçues sont stockées dans une base de données SQLite. Cette base contient des informations telles que le nom de l'animal, ses coordonnées, et sa température.

Tableau de bord :

Version de base : Utilise la bibliothèque turtle de Python pour afficher les positions des animaux sur un canevas graphique et génère des alertes via tkinter si un animal sort d'une zone prédéfinie.
Version avancée : Présente les données sur une page web interactive pour une visualisation plus détaillée et des fonctionnalités enrichies.

Structure des codes du Projet

```
Projet/
│
├── PythonCode/
│   ├── data_processing.py # Script pour le traitement des données
│   └── dashboard_basic.py # Implémentation d'un tableau de bord basique (utilise turtle, etc.)
│
├── ArduinoCode/
│   └── collar_device.ino # Code pour le dispositif de collier Arduino, incluant la logique d'envoi des données
│
└── WebInterface/ # Développé avec le framework web Python Flask
    ├── app.py # Application principale, définit les routes et les vues
    ├── templates/ # Dossier contenant les fichiers HTML
    │   └── index.html # Structure HTML du tableau de bord web
    └── static/ # Dossier pour les fichiers CSS et JavaScript
        ├── styles.css # Styles CSS pour le tableau de bord web
        └── script.js # Fichier JavaScript pour améliorer l'interaction côté client
```

Vous pouvez vérifier sur github avec le lien :
[petitlang/I2-Gr1-Porjet_mqtt_arduino_web (github.com)]

Spécifications Fonctionnelles

Collier Connecté :

Programme Arduino : Développement d'un programme pour simuler le mouvement aléatoire de l'animal à partir de la position initiale (0,0).
Transmission des Données : Envoie des coordonnées X et Y, et éventuellement des données de santé (ex : température) toutes les 10 secondes.
Format des Données : Les données sont formatées comme suit : Nom:Coordonnée X:Coordonnée Y[:Données Supplémentaires] (ex: Marguerite:34:-12:T=37).
Infrastructure de Communication :

Réseau et Serveur :

Antenne LoRa :
Utilisée pour amplifier la portée et la qualité du signal des données transmises via le réseau LoRa.
Acceptez les données générées par la carte Arduino et convertissez-les en informations mqtt
Broker MQTT (srv-lora) :
Centralise et gère les messages entrants via le protocole MQTT, assurant une communication efficace entre les colliers et le système central.Grâce à sa conception simple et à ses petits en-têtes de message, le protocole MQTT est bien adapté aux appareils à ressources limitées et aux réseaux à faible bande passante.

Tableau de Bord :

Réception des Données : Écoute et traite les messages du broker MQTT srv-lora.
Affichage : Présente les positions des animaux avec des couleurs distinctes pour chaque animal.
Alertes : Notification en cas de sortie d'un animal de la zone prédéfinie (-200, -200 à 200, 200).
Versions :

- Version Basique : Utilise turtle pour l'affichage graphique et tkinter pour les notifications d'alerte.
- Version Avancée : Utilise une interface web développée avec Flask pour afficher les positions et les alertes en temps réel.

En bref, la carte Arduino génère et transmet des données, l'antenne réseau permet la transmission de données sur le réseau LoRa, l'agent MQTT communique entre les composants du système et le tableau de bord écoute et analyse les données entrantes, affiche les données sous forme graphique et, dans la version avancée, indique l'emplacement de l'animal et déclenche des alarmes.

Planning

- Développement du projet : Réparti sur 6 séances encadrées et 3 séances en autonomie, avec l'utilisation de Git et GitHub pour la gestion de version et le partage du code.

Les séances : vendredi 05 avril 10:45-12:45 à NDL, mardi 16 avril 10:45-12:45 à NDC, mercredi 17 avril 16:00-18:00 à NDC(autonomie), vendredi 20 avril 10:45-12:45 à NDL, mercredi 24 avril 16:00-18:00 (autonomie), jeudi 25 avril 13:45-15:45(autonomie),vendredi 26 avril 10:45-12;45 à NDL(soutenance)

	Lianghong LI	Mouzheng LI	Guokuang DENG	Zhendong XU (être en charge, faire ensemblement)
Séance 1et 2 Vendredi 05 avril (NDL) Mardi 16 avril (NDC)	Développer le programme pour la carte Arduino MKRWAN.	Configurer le réseau LoRa et le serveur MQTT.	Commencer le développement du tableau de bord. C : Back-end et interface "turtle".	Conception de l'interface frontale de Web.
Séance 3 et 4 Mercredi 17 avril (autonomie) Vendredi 20 avril (NDL)	Tests en situation réelle, ajustement et optimisation de la communication et du traitement par le serveur.	Tests en situation réelle, ajustement et optimisation de la communication et du traitement par le serveur.	Continuer le développement et l'amélioration du tableau de bord, débuter l'intégration et les tests.	Continuer le développement et l'amélioration du tableau de bord, débuter l'intégration et les tests.
Séance 5 et 6 Mercredi 24 avril et Jeudi 25 avril (autonomie)	Intégration du système, assurer la connexion fluide entre les parties. Tests parallèles et préparation de la soutenance.	Intégration du système, assurer la connexion fluide entre les parties. Tests parallèles et préparation de la soutenance.	Intégration du système, assurer la connexion fluide entre les parties. Tests parallèles et préparation de la soutenance.	Intégration du système, assurer la connexion fluide entre les parties. Tests parallèles et préparation de la soutenance.