Séance 2 : chauffage d'appoint pour les abeilles "BeeWarm"

Introduction :

Lors de cette séance, nous avons poursuivi le développement du système de chauffage d'appoint pour les ruches, **BeeWarm**. J'ai principalement travaillé sur la partie communication et interaction entre l'utilisateur et le système embarqué via une application mobile. Dans un premier temps, nous avons conceptualisé l'identité visuelle du projet. Ensuite, nous avons programmé l'application afin de permettre la connexion en Bluetooth avec l'ESP32 placé dans la ruche, et le contrôle de la température du couvain. Enfin, nous avons commencé à tester le fonctionnement du Bluetooth et du Wi-Fi sur l'ESP32 à travers un programme de commande d'une LED, dans le but d'évaluer et de valider la technologie la plus adaptée à notre projet.

1) Conception du Logo de l'identité visuelle :

Nous avons également conçu l'identité visuelle de notre projet. J'ai donc conceptualisé le logo à l'aide de "Canva", en m'inspirant de l'univers graphique des abeilles et de l'apiculture. Les couleurs chaudes ainsi que le motif en alvéoles rappellent l'environnement du couvain, tandis que l'abeille stylisée renforce l'association avec la ruche.

2) Développement de l'application BeeWarm :

- **Fonctionnement général**

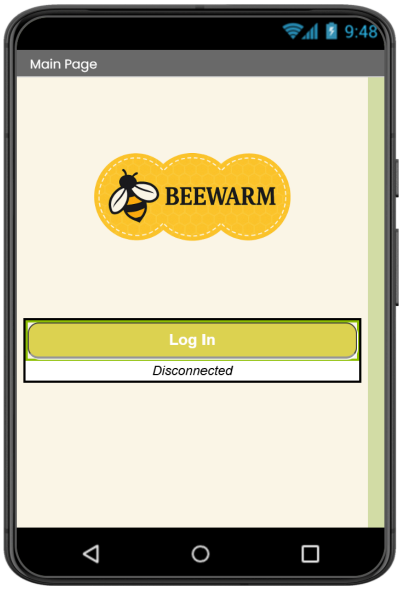
Afin de pouvoir communiquer avec l'ESP32 à l'intérieur de la ruche, on a opté pour le développement d'une application, qui permet de se connecter à un appareil Bluetooth afin

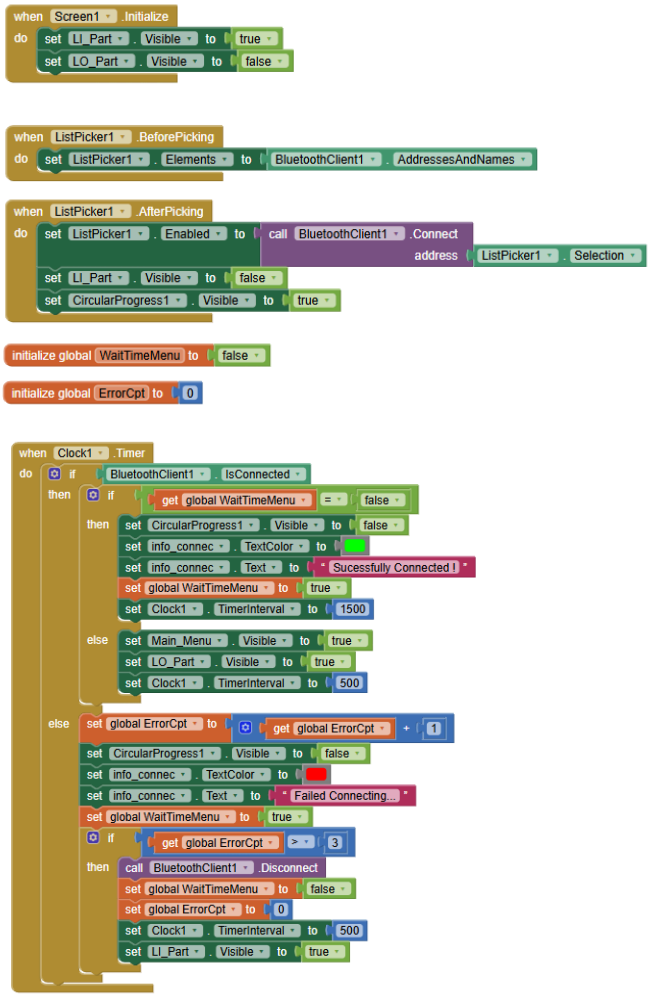
de contrôler la température destinée au bon développement des larves d'abeilles. Pour cela, on a décidé de la concevoir via le logiciel "Mit App Inventor".

En effet, l'application fonctionne avec un seul écran, mais plusieurs parties d'interface sont rendues visibles ou invisibles selon l'état de l'application (connexion, menu, contrôle), ce qui garde la connexion Bluetooth active en continu.

- ## Gestion de la connexion Bluetooth

L'utilisateur sélectionne un appareil dans une liste d'équipements Bluetooth appairés au téléphone "ListPicker1". Pendant la tentative de connexion, un indicateur de chargement apparaît "CircularProgress1" et des messages d'état informent l'utilisateur du succès ou de l'échec de la connexion. Après une connexion réussie, l'application attend environ 1,5s avant d'afficher le menu.





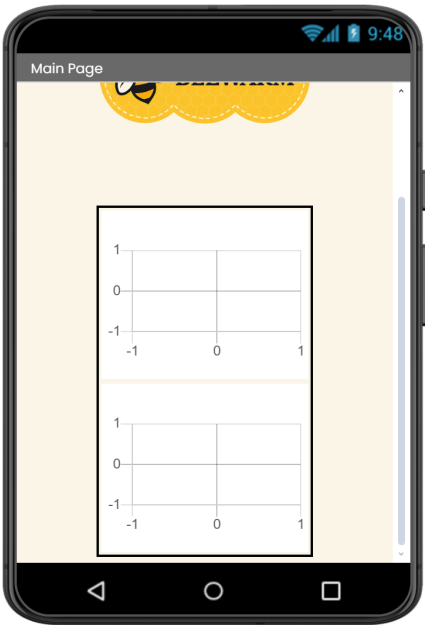
- ## Gestion de l'échec de connexion

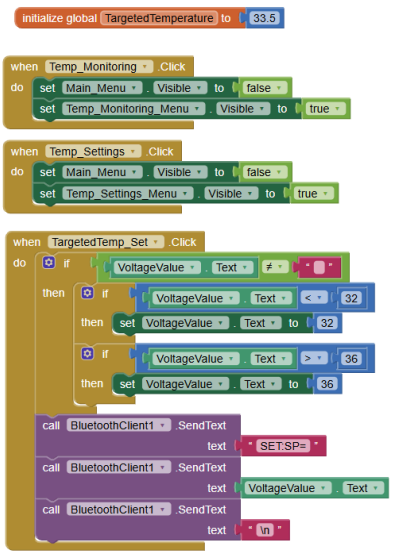
Une variable nommée "ErrorCpt" compte les tentatives échouées afin de détecter rapidement un échec persistant. Si la connexion échoue, un message rouge « Failed Connecting… » s'affiche pendant un court temps (≈1,5s). Une déconnexion forcée est

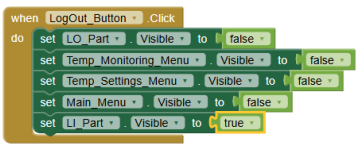
ensuite effectuée, puis l'interface revient automatiquement à la page de connexion afin de permettre une nouvelle tentative.

- **Saisie et contrôle de la température**

Dans la partie contrôle, l'utilisateur peut entrer une température cible pour les larves d'abeilles. Cette valeur est stockée dans la variable "TargetedTemperature**"**. L'application corrige automatiquement cette valeur pour qu'elle reste dans la plage optimale (32 °C à 36 °C). La consigne validée est envoyée après au système embarqué sous forme d'une commande texte.







3) Tests Arduino sur ESP32 :

Nous avons également réalisé un test pratique sur une ESP32 afin d'évaluer le fonctionnement des communications Wifi et Bluetooth du microcontrôleur. Pour cela, nous avons écrit un programme permettant de contrôler l'allumage d'une LED à distance à l'aide de ces deux technologies :

```
#include <WiFi.h>
#include <BluetoothSerial.h>

BluetoothSerial SerialBT;

// A changer
const char* ssid = "Votre_SSID";
const char* password = "Votre_MDP";

void setup() {

  pinMode(13, OUTPUT);

  Serial.begin(115200);
  delay(1000);

  Serial.println("\n===== TEST WiFi & Bluetooth ESP32 =====");

  // Test wifi : Affiche MAC si connecté
  Serial.print("Adresse MAC WiFi : ");
  Serial.println(WiFi.macAddress());

  Serial.print("Connexion à : ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  unsigned long startAttempt = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - startAttempt < 5000) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("WiFi connecté !");
    Serial.print("Adresse IP : ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("Impossible de se connecter au WiFi");
  }

  // Test bluetooth
  if (SerialBT.begin("ESP32_Test")) {
    Serial.println("Bluetooth activé !");
    Serial.println("Recherchez l'appareil : ESP32_Test");
  } else {
    Serial.println("Erreur Bluetooth");
  }
}


void loop() {

  // Test LED
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);

  // Envoi/lecture Bluetooth
  if (SerialBT.available()) {
    String msg = SerialBT.readString();
    Serial.print("Reçu en BT : ");
    Serial.println(msg);
  }

  // Envoi automatique pour test
  SerialBT.println("Ping BT depuis ESP32");
  delay(1000);
}

```

Conclusion :

Au cours de cette séance, j'ai réussi à finaliser le logo de BeeWarm, développé l'application mobile avec gestion du Bluetooth et validé une petite partie de la communication avec l'ESP32 par un test LED.

Lors des prochaines séances, j'ai pour objectif de :

- intégrer et tester le code Arduino avec le système de chauffage
- ajouter une base de données pour attribuer identifiant et mot de passe aux appareils connectés
- participer au code Stepper pour la partie mécanique liée au panneau et moteur
- ajouter le retour de température mesurée directement dans l'application