Séance 3 : Chauffage d'appoint pour les abeilles « BeeWarm »

Introduction :

Pendant cette séance, l'objectif était de mettre en service la PCB du système de chauffage et de la rendre autonome, sans dépendre du port USB du PC. J'ai d'abord configuré la communication entre l'ESP32 et le PC en installant le driver FTDI232, puis écrit un programme de test Bluetooth pour envoyer des températures simulées vers l'application. Ensuite, nous avons travaillé sur l'alimentation externe en soudant de nouveaux fils. Enfin, nous avons alimenté le système et vérifié la bonne transmission des données.

1) Mise en place de la communication Bluetooth :

La première étape que j'ai réalisée a été l'installation du driver FTDI. Ce driver est essentiel pour assurer la communication entre l'ordinateur et le module ESP32 via le port USB. Une fois le driver correctement installé, j'ai pu établir une liaison série stable avec le microcontrôleur.

Pour vérifier que l'ESP32 répond bien, j'ai écrit un code de test qui envoie régulièrement des données simulant les températures dans la ruche et dans le corps de chauffe.

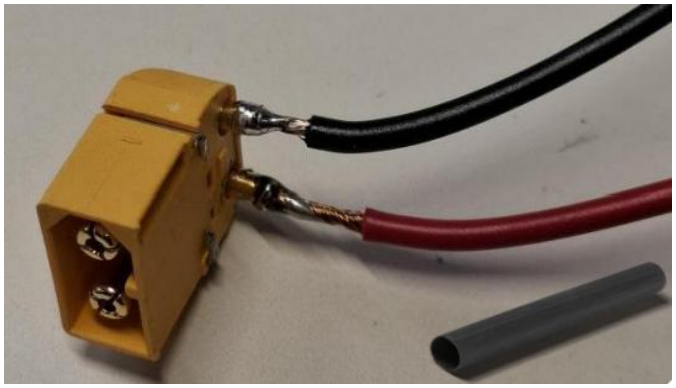
```
//#include <OneWire.h>
//#include <DallasTemperature.h>
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;

float tVeg = 30.0;
float tRuche = 30.0;

void setup() {
    Serial.begin(115200);
    delay(1000);
    Serial.println("test");
    delay(1000);

  SerialBT.begin("BeeWarmESP32");  // nom de l'esp32 visible sur le smartphone

}

void loop() {
  tVeg += random(-5, 6)*0.1; // [-5;6[ x 0,1 => simuler des variations entre [-0,5°C;0,5°C]
  tRuche += random(-5, 6)*0.1;
  if (tVeg < 25) tVeg = 25; // T min végétaline
  if (tVeg > 45) tVeg = 45; // T max végétaline
  if (tRuche < 25) tRuche = 25; // T min ruche
  if (tRuche > 35) tRuche = 35; // T max ruche

  String message = "DATA;" + String(tVeg, 1) + "°C;" + String(tRuche, 1) + "°C"; // conversion float à string avec ##,#
  SerialBT.println(message); // envoie du message
  Serial.println(message);
  delay(3000);
}
```

2) Modification de l'alimentation du circuit :

La prise normalement prévue sur la carte pour l'alimentation par panneau solaire devait initialement servir à brancher la source à une tension entre 0V et 25V.

Dans un premier temps, le système était alimenté uniquement par le port USB, ce qui n'est pas tout le temps adapté à une utilisation sur le terrain.

Et donc pour autonomiser l'alimentation du circuit, on a soudé deux fils directement sur la PCB, l'un pour l'alimentation et l'autre pour la masse. J'ai ensuite ajouté des connecteurs Wago. Et finalement, pour protéger l'ensemble, j'ai utilisé de la gaine thermorétractable, ce qui assure l'isolation thermique et électrique des contacts.



3) Programme de validation du fonctionnement global de l'ESP32 (à vérifier) :

J'ai écrit un programme de test permettant de simuler le comportement du système avant l'intégration des capteurs réels :

```
#include <BluetoothSerial.h>
BluetoothSerial SerialBT;

float tVeg = 30.0;
float tRuche = 30.0;
float consigne = 33.0;

String macStr;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("BeeWarmESP32"); // Nom Bluetooth

  // Récupérer l'adresse MAC
  uint8_t mac[6];
  esp_efuse_mac_get_default(mac);
  char buffer[18];
  sprintf(buffer, "%02X:%02X:%02X:%02X:%02X:%02X",
  mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
  macStr = String(buffer);

  Serial.println("MAC = " + macStr);
}

void loop() {
  tVeg   += random(-5, 6) * 0.1;
  tRuche += random(-5, 6) * 0.1;

  tVeg   = constrain(tVeg,   25, 45);
  tRuche = constrain(tRuche, 25, 45);

  int chauffage = (tRuche < consigne) ? 1 : 0;

  // données enviyées
  String msg = "DATA;" + String(tVeg,1) + ";" + String(tRuche,1) + ";" + String(consigne,1) + ";" + String(chauffage) + ";" + macStr;
  SerialBT.println(msg);
  Serial.println(msg);

  // Lire commande Bluetooth
  if (SerialBT.available()) {
    String cmd = SerialBT.readStringUntil('\n');
    cmd.trim();
    if (cmd.startsWith("SET:")) {
      consigne = cmd.substring(4).toFloat();
      SerialBT.println("OK;Consigne=" + String(consigne,1));
    }
  }

  delay(3000);
}
```

Ici, ce code sert principalement de version de test. Il simule deux températures : celle de la ruche et celle de l'élément chauffant, en leur faisant subir de petites variations pour imiter un fonctionnement réel. L'ESP32 active le Bluetooth et envoie régulièrement ces valeurs vers l'application, sous la forme d'un message simple à interpréter. L'adresse MAC du module est également transmise, ce qui facilite son identification lors du couplage.

Une commande envoyée depuis l'application permet aussi de modifier la température de consigne, ce qui prépare déjà le contrôle du chauffage dans la version finale.

Conclusion :

Cette séance a permis de rendre le système autonome et communicant : l'ESP32 est maintenant alimenté en tension réglable, fonctionne en Bluetooth et envoie correctement les données vers l'application. Les bases matérielles et logicielles sont donc en place.

La suite consistera à remplacer les valeurs simulées par de vrais capteurs et à intégrer le chauffage réel pour tester le système directement en conditions de ruche.