Compte rendu séance 5 Téo Baillot d'Estivaux

Objectifs de la séance :

Lors de la dernière séance, j'ai fini le fonctionnement du bras robotique pour qu'il fonctionne parfaitement sans bluetooth. L'objectif de cette séance sera donc d'établir une communication entre les deux cartes afin de faire en sorte par la suite qu'une carte renvoie les données de l'accéléromètre et du capteur flex à l'autre carte qui elle contrôlera les différents servos moteurs.

Communication buetooth :

Premier essai :

Le but de cette tentative est d'utiliser les modules bluetooth intégré aux deux cartes ESP32 pour envoyer un message simple d'une carte à une autre.
Pour ce faire nous allons établir deux codes, un pour la carte qui enverra le code, et un pour la carte qui recevra le code.
La difficulté est qu'il y a très peu de documentation pour établir une communication bluetooth entre deux cartes par contre j'ai trouvé beaucoup de documentation pour envoyer des données d'une carte à un téléphone donc je vais essayer de me débrouiller en utilisant la bibliothèque BluetoothSerial en faisant un appairage automatique des deux cartes en bluetooth.

Code de la carte émettrice :

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  Serial.println("Test Bluetooth émetteur...");

// On teste si la connexion s'est bien établie, tant que ce n'est pas le cas on attend
  if (!SerialBT.begin("TTGO-Emitter")) {
    Serial.println("Erreur d'initialisation Bluetooth !");
    while (1);
  }
  Serial.println("Bluetooth prêt !");
}

void loop() {
  SerialBT.println("Test Bluetooth !");
  Serial.println("Message envoyé via Bluetooth.");
  delay(1000);
}
```

carte réceptrice :

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  Serial.println("Test Bluetooth récepteur...");
```

```
// On teste si la connexion s'est bien établie, tant que ce n'est pas le cas on attend
  if (!SerialBT.begin("TTGO-Receiver")) {
    Serial.println("Erreur d'initialisation Bluetooth !");
    while (1);
  }
  Serial.println("Bluetooth prêt !");
}

void loop() {
  if (SerialBT.available()) {
    String data = SerialBT.readStringUntil('\n');
    Serial.println("Données reçues : " + data); // On affiche les données reçues
  }
}
```

En testant le code, je me suis rendu compte que l'appairage automatique n'a pas l'air de fonctionner puisque aucun message n'apparaît dans les consoles des deux cartes.

Deuxième essai :

Pour ce deuxième essai, je vais essayer de trouver une autre façon d'appairer les deux cartes puisque l'appairage automatique ne fonctionne pas.
L'objectif est donc de récupérer l'adresse MAC de la carte réceptrice pour spécifier dans le code de la carte émettrice l'adresse à laquelle les données devront être envoyées.

J'ai donc trouvé un programme pour trouver l'adresse MAC d'une carte:

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

void readMacAddress(){
  uint8_t baseMac[6];
  esp_err_t ret = esp_wifi_get_mac(WIFI_IF_STA, baseMac);
  if (ret == ESP_OK) {
    Serial.printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                  baseMac[0], baseMac[1], baseMac[2],
                  baseMac[3], baseMac[4], baseMac[5]);
  } else {
    Serial.println("Failed to read MAC address");
  }
}

void setup(){
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);
  WiFi.STA.begin();

  Serial.print("[DEFAULT] ESP32 Board MAC Address: ");
  readMacAddress();
}
void loop(){

}
```

On obtient l’adresse MAC de la carte émettrice: 30:ae:a4:74:06:ac
Et l’adresse MAC de la carte réceptrice : 30:ae:a4:6f:06:84

En utilisant cette adresse MAC de la carte réceptrice j’ai fait les codes suivants :
Code de la carte émettrice :

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  Serial.println("Démarrage du programme");

  // Connexion Bluetooth à la carte réceptrice en utilisant son adresse MAC
  String receiverMAC = "30:AE:A4:6F:06:84";  // Adresse MAC de la carte réceptrice
  SerialBT.begin("ESP32-Sender");  // Nom de l'appareil Bluetooth (émetteur)

  // Connexion à la carte réceptrice
  if (SerialBT.connect(receiverMAC)) {
    Serial.println("Connecté à la carte réceptrice !");
  } else {
    Serial.println("Échec de la connexion Bluetooth !");
  }
}

void loop() {
  if (SerialBT.connected()) {
    // Si la connexion est établie, envoyer des données
    SerialBT.println("Hello");
    delay(1000);
  } else {
// Si la connexion n’est pas établie, attendre
    Serial.println("En attente de connexion...");
    delay(1000);
  }
}
```

Code de la carte réceptrice :

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

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

  Serial.println("Démarrage du programme");

  // Initialisation du Bluetooth
  if (!SerialBT.begin("ESP32-Receiver")) {
    Serial.println("Erreur d'initialisation du Bluetooth !");
    return;
  }
```

```
  Serial.println("En attente de connexion Bluetooth...");
}

void loop() {
// Si la carte est prête à recevoir des données on récupère les données envoyées
  if (SerialBT.available()) {
    String receivedData = SerialBT.readString();
    Serial.println("Données reçues : " + receivedData);
  }
}
```

Finalement ça ne fonctionnait toujours pas, la carte réceptrice affichait dans le moniteur série qu'elle était prête à recevoir mais la carte émetrice affichait dans le moniteur série qu'elle n'arrivait pas à établir la connexion bluetooth.

<u>Troisième essai :</u>

En essayant de récupérer le code de quelqu'un qui contrôle un drone par bluetooth et en l'adaptant pour simplement envoyer un message, j'ai essayé d'établir la connexion bluetooth entre les deux cartes.
Le code d'origine renvoyait la position de deux joystick et j'ai juste modifié le code pour renvoyer le message « HELLO » à la place.
Les codes modifiés sont les suivants :

Code de la carte réceptrice :

```
#include "BluetoothSerial.h"

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32test"); //Bluetooth device name
  Serial.println("The device started, now you can pair it with bluetooth!");
}

uint8_t calculate_checksum(uint8_t *data) {
  uint8_t checksum = 0;
  checksum |= 0b11000000 & data[1];
  checksum |= 0b00110000 & data[2];
  checksum |= 0b00001100 & data[3];
  checksum |= 0b00000011 & data[4];
  return checksum;
}

void loop() {
  uint8_t recv_data[6];
  if (SerialBT.available()) {
    SerialBT.readBytes(recv_data, 6);

    if (recv_data[0] != 'T') {
      Serial.print("Receive error!");
```

```
    return;
   }

   if (recv_data[5] != calculate_checksum(recv_data)) {
    Serial.print("Decode error!");
    return;
   }
   Serial.printf(recv_data[1], recv_data[2], recv_data[3], recv_data[4]);
  }
  delay(20);
}
```

Code de la carte émettrice :

```
#include "BluetoothSerial.h"
#define Right_VRX_PIN 12
#define Right_VRY_PIN 13
#define Left_VRX_PIN 25
#define Left_VRY_PIN 26

BluetoothSerial SerialBT;

String MACadd = "30:AE:A4:6F:06:84";//adresse MAC de notre carte réceptrice
uint8_t address[6]  = {0x30, 0xAE, 0xA4, 0x6F, 0x06, 0x84};//adresse MAC de notre carte réceptrice en
Hexadécimal
bool connected;

void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32test", true);
  Serial.println("The device started in master mode, make sure remote BT device is on!");

  // connect(address) is fast (upto 10 secs max), connect(name) is slow (upto 30 secs max) as it needs
  // to resolve name to address first, but it allows to connect to different devices with the same name.
  // Set CoreDebugLevel to Info to view devices bluetooth address and device names
  connected = SerialBT.connect(address);

  if(connected) {
   Serial.println("Connected Succesfully!");
  } else {
   while(!SerialBT.connected(10000)) {
    Serial.println("Failed to connect. Make sure remote device is available and in range, then restart
app.");
   }
  }
  // disconnect() may take upto 10 secs max
  if (SerialBT.disconnect()) {
   Serial.println("Disconnected Succesfully!");
  }
  // this would reconnect to the name(will use address, if resolved) or address used with
connect(name/address).
  SerialBT.connect();

}

uint8_t calculate_checksum(uint8_t *data) {
```

```
  uint8_t checksum = 0;
  checksum |= 0b11000000 & data[1];
  checksum |= 0b00110000 & data[2];
  checksum |= 0b00001100 & data[3];
  checksum |= 0b00000011 & data[4];
  return checksum;
}

void loop() {

  uint8_t send_data[6];

  send_data[0] = 'H';
  send_data[1] = 'E';
  send_data[2] = 'L';
  send_data[3] = 'L';
  send_data[4] = '0';
  send_data[5] = calculate_checksum(send_data);
  SerialBT.write(send_data, 6);

  delay(20);
}
```

Cet essai qui utilise également l'adresse MAC de notre carte réceptrice n'est toujours pas concluant car il affiche constamment le message « Failed to connect » dans les moniteurs séries des deux cartes donc la connexion avec les deux cartes n'a toujours pas été établie§

<u>Quatrième essai :</u>
L'essai suivant consiste à essayer avec ESPNow qui est censé être une solution performante et simple d'utilisation pour établir une communication entre deux cartes. Cependant cette connexion se fera par Wifi et non par Bluetooth. ESP-NOW permet de faire une communication dans les deux sens.

J'ai finalement réussi à faire le code suivant qui envoie le message « Hello » d'une carte à l'autre :

Code de la carte émettrice:

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

// On spécifie l'adresse MAX en hexadécimal de notre carte réceptrice
uint8_t broadcastAddress[] = {0x30, 0xAE, 0xA4, 0x6F, 0x06, 0x84};

// Variable pour vérifier si l'envoie de données est réussi
String success;

// Fonction appelée quand la donnée est envoyée
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Etat du dernier paquet envoyé:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envoyé avec succès" : "envoie raté");
  success = (status == ESP_NOW_SEND_SUCCESS) ? "Envoie réussi :)" : "Envoie raté :(";
}

// Fonction appelée quand la donnée est reçue
void OnDataRecv(const esp_now_recv_info_t *recv_info, const uint8_t *incomingData, int len) {
  Serial.print("Message reçu: ");
```

```
  char msg[len + 1]; // On converti la donnée en chaîne de caractères
  memcpy(msg, incomingData, len);
  msg[len] = '\0'; // ajout du caractère de fin de chaîne de caractère
  Serial.println(msg);

  // Affiche l'adresse MAC de la carte émettrice
  Serial.print("From MAC: ");
  for (int i = 0; i < 6; i++) {
    Serial.printf("%02X", recv_info->src_addr[i]);
    if (i < 5) Serial.print(":");
  }
  Serial.println();
}

void setup() {
  Serial.begin(115200);

  // on active le wifi
  WiFi.mode(WIFI_STA);

  // on active ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Erreur d'initialisation ESP-NOW");
    return;
  }

  // Appelle des fonctions pour vérifier si les données sont bien envoyées et bien reçues
  esp_now_register_send_cb(OnDataSent);
  esp_now_register_recv_cb(OnDataRecv);

  // Stockage des informations liées à la communication
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0; //Définition du canal de communication
  peerInfo.encrypt = false;
```

//Ajout de l'appareil appairé

```
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Echec de l'appairage");
    return;
  }
}

void loop() {
  // Message à envoyer
  const char* message = "Hello";

  // Envoie du message avec ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *)message, strlen(message));

  if (result == ESP_OK) {
    Serial.println("Envoie réussi");
  } else {
```

```
  Serial.println("Erreur lors de l'envoie");
 }
 delay(2000);
}
```

Code de la carte réceptrice:

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

// Fonction appelée quand la donnée est reçue
void OnDataRecv(const esp_now_recv_info_t *recv_info, const uint8_t *incomingData, int len) {
 // Converti la donnée reçu en chaîne de caractères
 char msg[len + 1];
 memcpy(msg, incomingData, len);
 msg[len] = '\0'; // ajout du caractère de fin de chaîne de caractère

 // Affichage du message
  Serial.print("Message envoyé par: ");
 for (int i = 0; i < 6; i++) {
  Serial.printf("%02X", recv_info->src_addr[i]);
  if (i < 5) Serial.print(":");
 }
 Serial.println();
 Serial.print("Message: ");
 Serial.println(msg);
}

void setup() {
 Serial.begin(115200);
 Serial.println("Initialisation");

 // Activation du wifi
 WiFi.mode(WIFI_STA);

 // Activation de ESP-NOW
 if (esp_now_init() != ESP_OK) {
  Serial.println("Erreur d'inialisation ESP-NOW");
  return;
 }

 // On appelle la fonction pour vérifier que le message est bien envoyé
 esp_now_register_recv_cb(OnDataRecv);

 Serial.println("Pret à recevoir des données");
}

void loop() {
 // Rien à écrire car les messages récupérés sont gérés avec l'appel de la fonction OnDataRecv
}
```

Objectif de la prochaine séance :

Maintenant que j'ai réussi à établir une communication simple entre les deux cartes, l'objectif de la prochaine séance sera d'adapter le code pour envoyer les données de l'accéléromètre et du capteur flex d'une carte à l'autre carte de façon à pouvoir contrôler le bras robotique à distance.