Date : 10/03/2025

Compte-rendu séance 10 Téo Baillot d'Estivaux :

Objectifs de la séance :

L'objectif de ma séance d'aujourd'hui est de finaliser l'amélioration des mouvements que j'avais commencé lors de la dernière séance de façon à ce que l'on puisse attraper des objets se situant à différentes distances de la base du bras robotique. Ceci va donc améliorer grandement la mobilité du bras robotique qui est pour le moment capable d'uniquement attraper des objets qui se situent dans un rayon précis.

Mise en place du nouveau code :

L'idée est donc de modifier le code de la carte réceptrice pour modifier les mouvements du bras robotique en fonction des valeurs renvoyées par l'accéléromètre en utilisant les mesures faites lors de la séance précédente. L'idée est donc maintenant d'utiliser l'axe Y pour déplacer le bras robotique suivant cette idée : au départ quand l'accéléromètre est vers le haut, le bras robotique est en position initiale vers le haut. Quand on va commencer à baisser l'accéléromètre, la pince va se pencher de façon à pouvoir attraper un objet le plus loin possible de la pince. Puis, lorsqu'on continuera de baisser l'accéléromètre, le bras robotique se pliera de façon à pouvoir attraper des objets de plus en plus proches de sa base.

Pour ce faire, j'ai donc modifié le code de la carte réceptrice en utilisant les mesures de la séance précédente et en faisant des ajustements au fur et à mesure des tests. Je suis finalement arrivé à un résultat plutôt satisfaisant qui permet vraiment d'améliorer énormément la mobilité de la pince et qui rend même son contrôle plus simple en faisant le code suivant :

```
#include <ESP32Servo.h>

#include <esp_now.h>

#include <WiFi.h>


#define PIN_SERVO 16          // Définition de la broche pour le servo SG90

#define PIN_SERVO2 13         // Définition de la broche pour le servo SG902

#define PIN_SERVO3 14         // Définition de la broche pour le servo SG903

#define PIN_SERVO4 15         // Définition de la broche pour le servo SG904

#define SERVO_PIN_5 4         // Définition de la broche pour le servo SG905


Servo sg90;                   // Déclaration de l'objet servo pour SG90 (Servo contrôlé par l'axe X de l'accéléromètre)
```

```
Servo sg902;              // Déclaration de l'objet servo pour SG902 (Servo du bas contrôlé par l'axe Y de l'accéléromètre)

Servo sg903;              // Déclaration de l'objet servo pour SG903 (Servo du milieu contrôlé par l'axe Y de l'accéléromètre)

Servo sg904;              // Déclaration de l'objet servo pour SG904 (Servo du haut contrôlé par l'axe Y de l'accéléromètre)

Servo sg905;              // Déclaration de l'objet servo pour SG905 (Servo pour l'ouverture et la fermeture de la pince contrôlé par le capteur flex)


const int FLEX_MIN = 3400;     // Définition de la valeur minimale du capteur flex

const int FLEX_MAX = 2700;     // Définition de la valeur maximale du capteur flex

const int SERVO_MIN_ANGLE = 0; // Définition de l'angle minimum pour les servos

const int SERVO_MAX_ANGLE = 90;// Définition de l'angle maximum pour les servos

const int SERVO_MIN_ANGLE_FLEX = 20;// Définition de l'angle maximum pour le servo du capteur flex

const int ANGLE_CHANGE_THRESHOLD = 1; // Seuil à dépasser pour changer l'angle du servo dans certains cas


float currentAngleSg90 = 0;    // Variable pour suivre l'angle actuel du servo SG90

float currentAngleSg902 = 0;   // Variable pour suivre l'angle actuel du servo SG902

float currentAngleSg903 = 0;   // Variable pour suivre l'angle actuel du servo SG903

float currentAngleSg904 = 0;   // Variable pour suivre l'angle actuel du servo SG904

float currentAngleSg905 = 0;   // Variable pour suivre l'angle actuel du servo SG905


struct SensorData {         // Définition d'une structure pour contenir les données du capteur

  float accelX;           // Accéléromètre sur l'axe X

  float accelY;           // Accéléromètre sur l'axe Y

  int flex;              // Valeur du capteur flex

};

SensorData receivedData;      // Déclaration de l'objet `receivedData` pour stocker les données reçues


void OnDataRecv(const esp_now_recv_info_t *recv_info, const uint8_t *incomingData, int len) {
```

```
  SensorData receivedData;  // Déclaration d'une variable locale pour stocker les données reçues
  memcpy(&receivedData, incomingData, sizeof(receivedData)); // Copie les données reçues dans receivedData

  // Calcul de l'angle du servo pour le capteur flex
  int servoAngle = map(receivedData.flex, FLEX_MIN, FLEX_MAX, SERVO_MAX_ANGLE, SERVO_MIN_ANGLE_FLEX); // Mapping de la valeur flex à un angle de servo
  servoAngle = constrain(servoAngle, SERVO_MIN_ANGLE_FLEX, SERVO_MAX_ANGLE); // On vérifie que le résultat est dans la plage de données valide
  if (abs(servoAngle - currentAngleSg905) >= ANGLE_CHANGE_THRESHOLD) { // Vérification si l'angle a changé d'au moins 5 degré
    sg905.write(servoAngle); // Envoie la nouvelle valeur d'angle au servo SG905
    currentAngleSg905 = servoAngle; // Met à jour l'angle actuel de SG905
  }

  // Calcul de l'angle pour l'accéléromètre X
  int mappedAngleSg90 = map(receivedData.accelX, 12000, -12000, 0, 90); // Mapping de la valeur de l'accéléromètre X à un angle de servo.
  mappedAngleSg90 = constrain(mappedAngleSg90, SERVO_MIN_ANGLE, SERVO_MAX_ANGLE); // On vérifie que le résultat est dans la plage de données valide
  if (abs(mappedAngleSg90 - currentAngleSg90) >= ANGLE_CHANGE_THRESHOLD) { // Vérification si l'angle a changé d'au moins 5 degré
    sg90.write(mappedAngleSg90); // Envoie la nouvelle valeur d'angle au servo SG90
    currentAngleSg90 = mappedAngleSg90; // Met à jour l'angle actuel de SG90
  }

  // Autres mouvements en fonction de Y
  if (receivedData.accelY < -10000) { // Si la valeur de Y est inférieure à 700
    sg902.write(0); // Met le servo SG902 à 0°.
    sg903.write(0); // Met le servo SG903 à 0°.
    sg904.write(0); // Met le servo SG904 à 0°.
    currentAngleSg902 = currentAngleSg903 = currentAngleSg904 = 0; // Met à jour les angles actuels
```

```
  } else if (receivedData.accelY < -7700) { // Si la valeur de Y est inférieure à 800
    int angle1 = map(receivedData.accelY, -10000, -7700, 0, 50);
    int angle2 = map(receivedData.accelY, -10000, -7700, 0, 35); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Envoie l'angle au servo SG902
    sg903.write(0); // Met le servo SG903 à 0°.
    sg904.write(angle2); // Met le servo SG904 à 0°.
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903  = 0; // Met à jour les angles actuels de SG903 et SG904
    currentAngleSg904=angle2;
  } else if (receivedData.accelY < -5400) { // Si la valeur de Y est inférieure à 900
    int angle1 = map(receivedData.accelY, -7700, -5400, 50, 35); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, -7700, -5400, 0, 15); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = map(receivedData.accelY, -7700, -5400, 35, 45); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  } else if (receivedData.accelY < -3100) { // Si la valeur de Y est inférieure à 1050
    int angle1 = map(receivedData.accelY, -5400, -3100, 35, 30); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, -5400, -3100, 15, 20); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = map(receivedData.accelY, -5400, -3100, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
```

```
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  }else if (receivedData.accelY < -800) { // Si la valeur de Y est inférieure à 1050
    int angle1 = map(receivedData.accelY, -3100, -800, 30, 25); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, -3100, -800, 20, 35); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = map(receivedData.accelY, -3100, -800, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  } else if (receivedData.accelY < 1500) { // Si la valeur de Y est inférieure à 1050
    int angle1 = map(receivedData.accelY, -800, 1500, 25, 20); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, -800, 1500, 35, 40); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = map(receivedData.accelY, -800, 1500, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  } else if (receivedData.accelY < 3800) { // Si la valeur de Y est inférieure à 1050
    int angle1 = map(receivedData.accelY, 1500, 3800, 20, 15); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, 1500, 3800, 40, 50); // Mappe la valeur de Y à un angle entre 0 et 30°
```

```
    int angle3 = map(receivedData.accelY, 1500, 3800, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°

    sg902.write(angle1); // Met le servo SG902 à 30°

    sg903.write(angle2); // Envoie l'angle au servo SG903

    sg904.write(angle3); // Met le servo SG904 à 0°

    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902

    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903

    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904

  } else if (receivedData.accelY < 6100) { // Si la valeur de Y est inférieure à 1050

    int angle1 = map(receivedData.accelY, 3800, 6100, 15, 10); // Mappe la valeur de Y à un angle entre 0 et 30°

    int angle2 = map(receivedData.accelY, 3800, 6100, 50, 60); // Mappe la valeur de Y à un angle entre 0 et 30°

    int angle3 = map(receivedData.accelY, 3800, 6100, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°

    sg902.write(angle1); // Met le servo SG902 à 30°

    sg903.write(angle2); // Envoie l'angle au servo SG903

    sg904.write(angle3); // Met le servo SG904 à 0°

    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902

    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903

    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904

  }

  else if (receivedData.accelY < 8400) { // Si la valeur de Y est inférieure à 1050

    int angle1 = map(receivedData.accelY, 6100, 8400, 10, 5); // Mappe la valeur de Y à un angle entre 0 et 30°

    int angle2 = map(receivedData.accelY, 6100, 8400, 60, 65); // Mappe la valeur de Y à un angle entre 0 et 30°

    int angle3 = map(receivedData.accelY, 6100, 8400, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°

    sg902.write(angle1); // Met le servo SG902 à 30°

    sg903.write(angle2); // Envoie l'angle au servo SG903

    sg904.write(angle3); // Met le servo SG904 à 0°

    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902

    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
```

```
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  }
  else if (receivedData.accelY < 10700) { // Si la valeur de Y est inférieure à 1050
    int angle1 = map(receivedData.accelY, 8400, 10700, 5, 0); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, 8400, 10700, 65, 70); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = map(receivedData.accelY, 8400, 10700, 45, 45); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  } else if (receivedData.accelY < 13000) { // Si la valeur de Y est inférieure à 1050
    int angle1 = map(receivedData.accelY, 10700, 13000, 0, 0); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = map(receivedData.accelY, 10700, 13000, 70, 80); // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = map(receivedData.accelY, 10700, 13000, 45, 50); // Mappe la valeur de Y à un angle entre 0 et 30°
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  }
  else { // Si la valeur de Y est supérieure ou égale à 1050
    int angle1 = 0; // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle2 = 75; // Mappe la valeur de Y à un angle entre 0 et 30°
    int angle3 = 65; // Mappe la valeur de Y à un angle entre 0 et 30°
```

```
    sg902.write(angle1); // Met le servo SG902 à 30°
    sg903.write(angle2); // Envoie l'angle au servo SG903
    sg904.write(angle3); // Met le servo SG904 à 0°
    currentAngleSg902 = angle1; // Met à jour l'angle actuel de SG902
    currentAngleSg903 = angle2; // Met à jour l'angle actuel de SG903
    currentAngleSg904 = angle3; // Met à jour l'angle actuel de SG904
  }
  Serial.println(receivedData.accelY);
}

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA); // Configure l'ESP32 en mode WiFi

  // Initialisation d'ESP-NOW
  if (esp_now_init() != ESP_OK) { // Initialise ESP-NOW pour la communication sans fil
    Serial.println("Erreur d'initialisation ESP-NOW"); // Si l'initialisation échoue, afficher un message d'erreur
    return;
  }

  esp_now_register_recv_cb(OnDataRecv); // Enregistre la fonction qui sera appelée à chaque tentative d'envoie de données via ESP-NOW

  sg90.attach(PIN_SERVO);       // Attache le servo SG90 à la broche PIN_SERVO
  sg902.attach(PIN_SERVO2);     // Attache le servo SG902 à la broche PIN_SERVO2
  sg903.attach(PIN_SERVO3);     // Attache le servo SG903 à la broche PIN_SERVO3
  sg904.attach(PIN_SERVO4);     // Attache le servo SG904 à la broche PIN_SERVO4
  sg905.attach(SERVO_PIN_5);    // Attache le servo SG905 à la broche SERVO_PIN_5

  sg90.write(0);              // Initialise le servo SG90 à 0°
  sg902.write(0);             // Initialise le servo SG902 à 0°
```

```
  sg903.write(0);          // Initialise le servo SG903 à 0°
  sg904.write(0);          // Initialise le servo SG904 à 0°
  sg905.write(0);          // Initialise le servo SG905 à 0°
}

void loop() {
  // Boucle principale vide car l'action est déclenchée par la réception de données via ESP-NOW
}
```

Tests :

Tout au long du développement du code précédent, j'ai fait des tests de façon à ajuster les mouvements du bras robotiques pour qu'ils soient le plus fluides possibles et aussi le plus précis possible. Cependant, au bout d'un moment, je n'ai pas compris pourquoi mais l'accéléromètre s'est mis à ne plus fonctionner.

Mon premier réflexe a été de retéléverser le code, mais j'obtenais soit un message d'erreur me disant que ma carte n'arrive pas à lire les données de l'accéléromètre, soit l'accéléromètre ne renvoyait que des 0. J'ai donc ensuite vérifié le câblage mais sans trouver de problème. J'ai par la suite essayé de tester avec un autre accéléromètre mais j'ai eu de nouveau le même problème ce qui m'a permis de comprendre que le problème ne vient pas de mon composant. Je me suis donc penché sur le code de la carte émettrice qui permet de lire les données de l'accéléromètre sans vraiment comprendre d'où pourrait venir le problème par rapport au début de la séance car je n'avais pas touché à ce code.

Après un long moment à chercher le problème sans le trouver, j'ai décidé de tester le tout premier code que j'avais fait pour tester l'accéléromètre numérique qui permet uniquement de lire les données du capteur. A mon grand étonnement, ce code permettait à l'accéléromètre de fonctionner correctement, j'ai donc compris que le problème venait probablement du code de ma carte émettrice et après un long moment à essayer de trouver le problème de ce code je me suis résolu à le refaire d'une façon différente pour essayer de résoudre le problème.

J'ai finalement réussi à développer une nouvelle version du code de la carte émettrice :

```
#include <esp_now.h>

#include <WiFi.h>

#include <Wire.h>


#define MPU6050_ADDR 0x68  // Adresse I²C du MPU-6050

#define FLEX_SENSOR_PIN 39 // GPIO pour le capteur flex

#define FILTER_SAMPLES 30  // Nombre d'échantillons pour le filtrage
```

```
uint8_t broadcastAddress[] = {0x30, 0xAE, 0xA4, 0x6F, 0x06, 0x84};

// Historique pour filtrer les données des capteurs
float flexHistory[FILTER_SAMPLES];
int filterIndex = 0;

// Structure pour envoyer les données des capteurs
struct SensorData {
  float accelX;
  float accelY;
  int flex;
};
SensorData dataToSend;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Données envoyées avec succès" :
"Échec de l'envoi des données");
}

void setup() {
  Serial.begin(115200);
  Wire.begin(); // Initialisation I²C

  // Vérifier la connexion avec le MPU-6050
  Wire.beginTransmission(MPU6050_ADDR);
  if (Wire.endTransmission() != 0) {
    Serial.println("MPU-6050 non détecté !");
    while (1);
  }
  Serial.println("MPU-6050 détecté !");
```

```
  // Activer le MPU-6050
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();

  // Initialisation ESP-NOW
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("Erreur d'initialisation ESP-NOW");
    return;
  }
  esp_now_register_send_cb(OnDataSent);

  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Échec de l'ajout du pair");
    return;
  }

  analogReadResolution(12);
}

void readAccelData(int16_t *ax, int16_t *ay) {
  Wire.beginTransmission(MPU6050_ADDR);
  Wire.write(0x3B);
```

```
  Wire.endTransmission(false);
  Wire.requestFrom(MPU6050_ADDR, 4, true);

  if (Wire.available() == 4) {
    *ax = Wire.read() << 8 | Wire.read();
    *ay = Wire.read() << 8 | Wire.read();
  } else {
    Serial.println("Erreur de lecture MPU-6050 !");
  }
}

float getFilteredValue(float *history, int pin) {
  float current = analogRead(pin);
  history[filterIndex] = current;
  float sum = 0;
  for (int i = 0; i < FILTER_SAMPLES; i++) {
    sum += history[i];
  }
  filterIndex = (filterIndex + 1) % FILTER_SAMPLES;
  return sum / FILTER_SAMPLES;
}

void loop() {
  int16_t ax, ay;
  readAccelData(&ax, &ay);

  dataToSend.accelX = ax;
  dataToSend.accelY = ay;
  dataToSend.flex = getFilteredValue(flexHistory, FLEX_SENSOR_PIN);

  Serial.print("X: "); Serial.print(dataToSend.accelX);
```

```
  Serial.print(" Y: "); Serial.print(dataToSend.accelY);
  Serial.print(" Flex: "); Serial.println(dataToSend.flex);

  esp_now_send(broadcastAddress, (uint8_t *)&dataToSend, sizeof(dataToSend));

  delay(10);
}
```

Ce code m'a permis de pouvoir lire les données de l'accéléromètre correctement sans avoir le problème expliqué précédemment mais malheureusement je n'ai pas réussi à établir la transmission de ces données jusqu'à la carte réceptrice et n'ai pas eu le temps de régler ce problème lors de cette séance.

Cette séance a donc été riche en problèmes inattendus m'ayant fait perdre beaucoup de temps dans l'avancée du projet sans vraiment comprendre de raison apparente à ce problème alors que j'aurais pu travailler sur la résolution du problème de pertes de paquets lors de la transmission des données de la carte émettrice à la carte réceptrice comme j'aurais voulu le faire au départ.

Objectifs de la prochaine séance :

L'objectif de la prochaine séance est de réussir à rétablir un code pour la carte émettrice fonctionnel car apparemment le code que j'avais fait et que j'avais mis dans les précédents rapports ne fonctionne plus (peu être à cause d'une mise à jour côté software) et j'espère également pouvoir avoir enfin le temps de travailler sur le problème de pertes de paquets décris précédemment.