18/11/2022

FERDJI Elias 201500010148 DJILLALI Mazigh 171731057209 CHEMLOUL Mounir 171731099587

TINYML: RECONNAISANCE DE GESTES AVEC LA CARTE ARDUINO NANO 33 BLE

1. MATERIEL ET PROTOCOL CHOISI

Pour commencer nous allons faire la liste du matériel et des codes que nous avons utilisés pour ensuite expliquer la méthode que nous avons choisie afin de faire communiquer les deux cartes Arduino

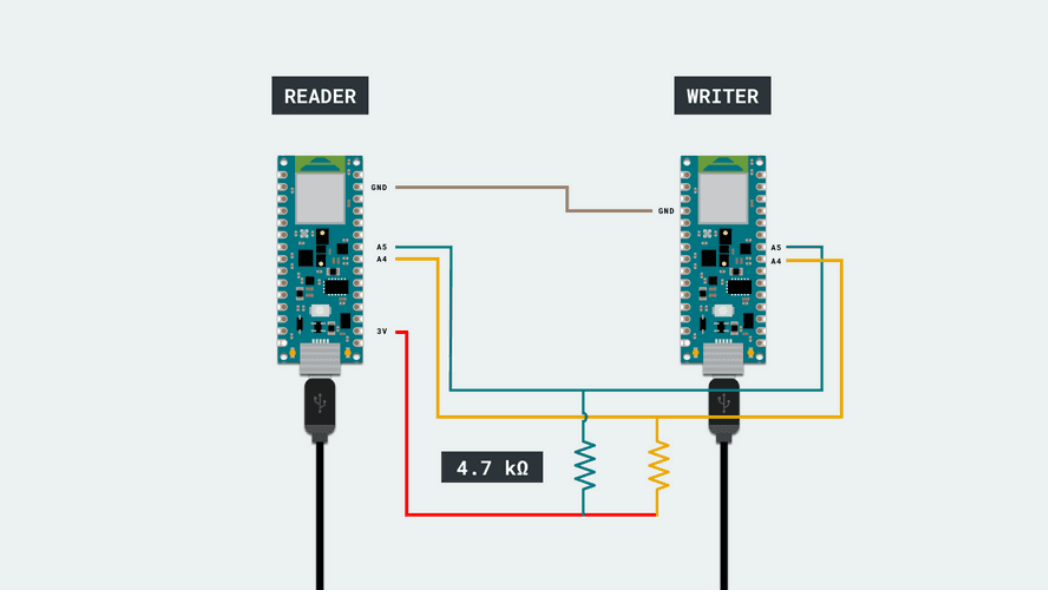
A. Liste du matériel utilisé

- 2 x Cartes Arduino Nano 33 BLE
- 15 x câbles de liaison
- Arduino IDE version 2.0.3

B. Technique de communication entre les cartes

Nous avons décidé de choisir de faire communiquer les deux cartes en utilisant la méthode Inter-Integrated Circuit «I2C».

La communication entre les deux cartes se fait à travers les ports A4 et A5, le port 3V sert à l'alimentation (qui est facultative dans notre cas) . Le montage se fait comme dans le schéma suivant :



2. CAPTURE DES DONNEES, CONSTITUTION DES DATASETS ET APPRENTISSAGE

Après avoir réalisé le montage nous aurons donc deux cartes, une étant la carte émettrice et la seconde sera celle qui va recevoir les données.

Aussi nous nous sommes basés sur la bibliothèque « Wire.h » afin de coder la communication.

A. Capture des données

Nous rappelons que dans ce projet nous essayons de capturer et de reconnaitre un mouvement exécuté avec les deux bras, c'est donc pour cela que nous utiliserons une carte pour capturer le mouvement droit et la seconde pour capturer le mouvement gauche.

Afin de capturer les données de l'accéléromètre et du gyroscope, nous avons utilisé le code vu en TP « IMU_Capture.io » . Dans la loop du code, nous allons au fur et à mesure lire les valeurs de l'accéléromètre dans aX, aY, aZ et celles du gyroscope dans gX, gY, gZ, et ces valeurs là vont être affichées dans le Serial Monitor, ainsi nous pouvons donc récupérer nos données et les mettre dans un fichier « .csv »

B. Constitution des datasets

Les deux mouvements qu'on a choisi sont le « Clap » qui est un applaudissement et le second est un mouvement qui consiste en : avancer la main droite et reculer la main gauche « RightForwardLeftBackward »

Pour chacun des mouvements, nous les avons répétés 15 fois afin de constituer un dataset fiable.

Lors de la constitution des fichiers de données, nous avons alterné entre le mouvement capturé par la droite et celui capturé par la gauche.

C. Apprentissage des mouvements

Une fois nos données rassemblées et nos datasets constitués, la prochaine étape est d'utiliser le machine-learning afin de générer un modèle capable de reconnaitre les mouvements récupérés

Pour cela, nous avons utilisé le fichier « arduino_tinyml_workshop,ipynb », dans lequel nous avons uploadé nos datasets, précisé le nom de nos gestes et procédé à l'apprentissage.

Le programme en python nous donne en sortie un fichier « model.h » que nous allons mettre dans notre dossier de travail pour pouvoir avancer vers la prochaine étape : La classification

3. COMMUNICATION ET CLASSIFICATION

A. Communication

Nous avons parlé plus haut de la communication entre les cartes et la méthode utilisée, ici nous allons expliquer au niveau du code notre manière d'utiliser la bibliothèque « Wire.h ».

Notre but est de reconnaitre des gestes effectués avec les deux mains en utilisant UN SEUL modèle . Pour cela, il nous faut programmer les deux cartes tel que :

- Une carte A va capturer ses propres coordonnées et les envoyer à la seconde carte
- Une seconde carte B va capturer ses propres données, réceptionner les données envoyées par la première et les envoyer pour une classification afin de reconnaitre le mouvement effectué.

1. Carte A :

Dans la boucle loop() :

```
// print the data in CSV format
Serial.print(aX, 3);
Serial.print(',');
Serial.print(aY, 3);
Serial.print(',');
Serial.print(aZ, 3);
Serial.print(',');
Serial.print(gX, 3);
Serial.print(',');
Serial.print(gY, 3);
Serial.print(',');
Serial.print(gZ, 3);
Serial.println();
//data est un string declare plus haut , qui contiendra les informations d'acceleration et du gyroscope
//la transimition se fait ligne par ligne
data += String(aX,3);
data += ",";
data += String(aY,3);
data += ",";
data += String(aZ,3);
data += ",";
data += String(gX,3);
data += ",";
data += String(gY,3);
data += ",";
data += String(gZ,3);
data += " \n";
Wire.beginTransmission(9); // debut de la transmition
Wire.write(data.c_str());  //formatage des donnes envoyees
Wire.endTransmission();    // fin de la transmition
```

La carte A aura pour code, le code « IMU_Capture.io » auquel nous ajoutons ce code dans le loop()

2. *Carte B :*

Dans la fonction setup() :

```
Wire.onReceive(receiveEvent); // nous attachons une fonction receiveEvent qui s'executera en boucle en attendant une information de la part
//de larduino Master
```

Fonction receiveEvent() :

```
void receiveEvent(int bytes) {

  while(Wire.available()){
  response += (char) Wire.read();

  }
  dataReceived = response;

  if (cpt<119){
     cpt++;
  }
  if (cpt >= 119){
    response = "";
    cpt =0;}
}
```

Tant que la communication est ouverte, on met dans notre response les données qu'on a reçu depuis la carte A (on les récupère ligne par ligne) .

On compte 119 lignes et on vide la réponse pour délimiter avec le prochain mouvement.

B. CLASSIFICATION

La dernière étape de ce projet est la classification des données recueillies par les deux cartes en se basant sur modèle créé plus haut.

Pour y parvenir, nous nous sommes basés sur le code « IMU_Classifier.io » fourni en TP, auquel nous avons ajouté les codes présentés précédemment pour la carte B .