Projet-ESP32-Mobile

Alexandre Cauty M2 IASD

January 2024

Pour le code, voici le lien du github : `https://github.com/alexandre-cauty08/iotproject/tree/master`.

Contents

1 Introduction

L'objectif de ce projet est de développer une interface utilisateur en Flutter pour interagir avec l'API RESTful implémentée pour le projet du capteur TTGO T-Display. Cette interface servira de pont entre l'utilisateur et le capteur, permettant une interaction visuelle et intuitive avec les données et commandes du capteur.

2 Récapitulatif Projet Arduino

Le but du projet était de créer une API REST sur une carte ESP32. Cette API doit être capable de récupérer les valeurs des capteurs connectés à l'ESP32. Les capteurs utilisés seront une thermo-résistance et une photo-résistance. Il doit être possible de contrôler l'allumage et l'extinction d'une LED.

2.1 Routes web server

La bibliothèque WebServer a permis de mettre en place un webServer sur le micro contrôleur. Grâce à cela,des appareils externes ont la possibilité d'envoyer des requêtes à notre serveur. Une multitude de route a été mis en place pour pouvoir réaliser les différentes fonctionnalités proposées.

```
1  server.on("/on", allume);
2  server.on("/off", eteindre);
3  server.on("/light", getLight);
4  server.on("/temp", getTemp);
5  server.on("/AugmenteSeuil", AugmenteSeuil);
6  server.on("/DiminueSeuil", DiminueSeuil);
7  server.on("/getCapteurs", getCapteurs);
8  server.onNotFound(handleNotFound);
9  server.begin();
```

Listing 1: Code des routes web

Pour avoir un retour sous format JSON des informations sur la luminosité, température . Deux fonctions ont été conçues :

- L'une permet la création d'un fichier JSON
- L'autre permet de remplir les champs dans le fichier JSON existant

```
void create_json(char *tag, float value, char *unit)
{
  jsonDocument.clear();
  jsonDocument["type"] = tag;
  jsonDocument["value"] = value;
  jsonDocument["unit"] = unit;
  serializeJson(jsonDocument, buffer);
}
```

Listing 2: Code création d'un JSON

```
void add_json_object(char *tag, float value, char *unit)
{
  JsonObject obj = jsonDocument.createNestedObject();
  obj["type"] = tag;
  obj["value"] = value;
  obj["unit"] = unit;
}
```

Listing 3: Code ajouter des champs au format

Ce récapitulatif permet de visualiser la suite du projet. Chaque route possède une sortie sous format JSON qu'on récupère sous Flutter.

2.2 Fonctionnalités de l'API

Voici les fonctionnalités disponible du projet à partir de chaque requête :

- Allumer la led
- Eteindre la led
- Récupérer les données de la photo résistance
- Récupérer les données de la thermo résistance
- La lumière s'allume ou s'éteint en fonction du seuil
- Augmenter le seuil
- Diminuer le seuil
- Afficher les capteurs utilisés

Pour chaque requête, un fichier JSON est créé dans le but de savoir ce qu'il y a été récupéré ou produit par la requête.

3 UML

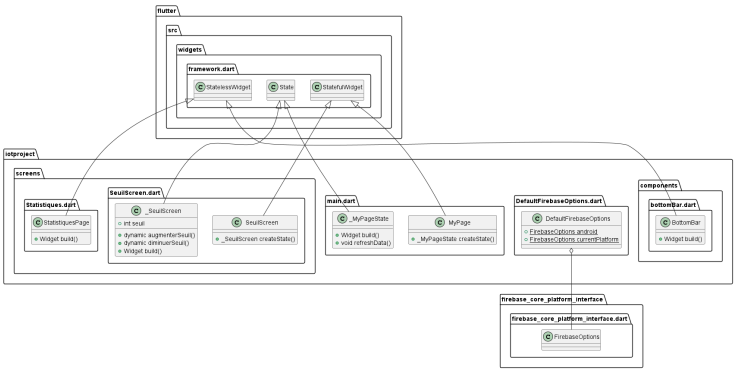

Figure 1: UML final

J'ai ajouté l'UML dans le dépôt car trop grand pour le rapport.

4 Workflow

1. Créer le montage de la figure 2
2. Lancer le programme Arduino de l'ancien projet. (Serveur Wifi activé)
3. Lancer l'application sur Android Studio.
4. Affichage de la Page d'accueil. La page d'acceuil est affiché, présentant les données récupérées par leurs urls et les boutons qui ont chacun une fonctionnalité.
5. Bouton refresh : affiche les nouveaux résultats tirés par la photo et thermo résistance.
6. Bouton Seuil : navigation vers la page Seuil. L'utilisateur clique le bouton seuil.
7. L'utilisateur a la possibilité de soit augmenter ou diminuer le seuil de luminosité.
8. Bouton Statistiques : navigation vers la page Statistiques
9. L'utilisateur peut voir les résultats de toutes les températures et luminosités qui ont été enregistré dans la page principale.
10. Fin de l'application : l'utilisateur peut quitter l'application

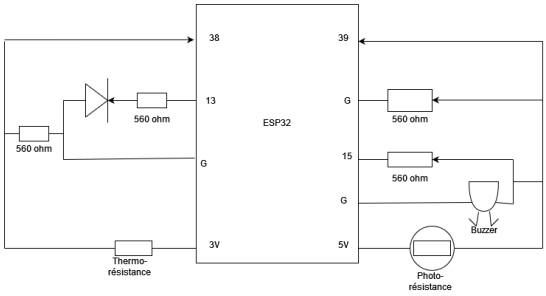

Figure 2: Montage

5 Architecture Flutter

5.1 main.dart

La classe main qui définit une classe MyPage, s'étend sur un StatefulWidget et a un état représenté par la classe MyPageState. Elle est la page principale dans lequel on trouvera les informations sur la température, luminosité et l'état de la led. Elle permet la naviguation sur les plusieurs pages disponibles et l'actualisation des données. On y retrouve :

- AppBar : barre d'application
- Body : Column qui contient plusieurs Widgets
- FutureBuilder : Récupération de données. Utilisation de plusieurs Future Builder pour gérer les opérations asynchrones de récupération de données.
- Container : Affichage des données dans des conteneurs stylisés
- Boutons : bouton pour allumer ou éteindre la led
- FloatActionButton pour actualiser les données, ce bouton appelle une fonction qui permet de relancer les 2 fonctions : TempData et LightData2.
- BottomNavigationBar : barre de naviguation

On retrouve donc 5 dossiers : components, routes, screens,service et utils.

5.2 Utils

Le dossier utils possède la classe utils.dart, les 4 fonctions représentent des appels de requêtes Url. On retrouve 2 fonctions qui agissent sur l'état de la lampe : Allume() et Eteindre(). Exemple : lorsqu'on clique sur le bouton 'Allumer la led'. Cela créer une requête qui appelle l'url suivant : 'http://192.168.1.41/on'.

```
void allume()
{
  mode = 0;
  create_json("Lumiere allumee",mode,".");
  server.send(200, "application/json", buffer);
}
```

Listing 4: Code condition de la led

```
 if(mode == 0)
  {
    tft.fillScreen(TFT_BLACK);
    digitalWrite(ledPin, HIGH); // Allume la led
  }
  else if(mode == 1)
  {
    tft.fillScreen(TFT_BLACK); // Eteint la led
    digitalWrite(ledPin, LOW);
  }
  else
  {
    if(lux > seuilAlarme)
    {
    tft.fillScreen(TFT_RED);
    digitalWrite(ledPin, LOW); // Eteint la LED
    myTone(BUZZER_PIN,100);
    }
    else
    {
    tft.fillScreen(TFT_GREEN);
    digitalWrite(ledPin, HIGH); // Allume la LED
    myTone(BUZZER_PIN,0);
    }
}
```

Listing 5: Code condition de la led

Ce bouton va permettre de changer le mode de la led à 0 . La led s'allumera après avoir appuyer sur le bouton 'Allumer la led'. Les 2 autres fonctions évoquent la récupération de données. Par exemple :

```
Future<Map<String, dynamic>> TempData() async {

  const String apiUrl = "http://192.168.1.41/temp";

    final response = await http.get(Uri.parse(apiUrl));

    if (response.statusCode == 200)
    {
      Map<String, dynamic> jsonData = json.decode(response.
          body);
      return jsonData;
    }
    else
    {
      throw Exception('Echec de la requete HTTP');
    }
```

Listing 6: Code fonction récupération de données temperature

Cette fonction récupère la donnée issue du format json pouvoir ensuite l'afficher dans le main.dart.

```
Map<String, dynamic>? jsonData = snapshot.
    data;
dynamic value = jsonData?["value"];
return Container(
  padding: const EdgeInsets.all(16.0),
  margin: const EdgeInsets.all(8.0),
  decoration: BoxDecoration(
    border: Border.all(color: Colors.
        blueAccent),
    borderRadius: BorderRadius.circular
        (10.0),
  ),
  child: Column(
    children: [
      const Text('Type : Temperature'),
      Text('Valeur : $value C'),
```

Listing 7: Code Affichage Temperature (main.dart)

5.3 Components

Le dossier compenents possède la classe BottomBar.dart. Comme son nom l'indique elle représente la bottom bar qui sera rappelé dans d'autres classes.

5.4 Routes

Le dossier routes contient la classe navigation.dart. Cette classe permet la naviguation de la HomePage aux pages Seuil et Statistiques à l'aide de Float-ActionButton.

5.5 Screens

Le dossier screens représente toutes les vues disponibles dans l'application. On retrouve 3 vues possibles : HompePage.dart, Seuil.dart et Statistiques.dart.

Dans la vue Seuil.dart, on retrouve l'a possibilité d'augmenter ou diminuer le seuil. Dans le code arduino précédent, une condition a été créé sur le seuil de luminosité. Si la luminosité dépasse 1500 lux la led s'éteint sinon elle s'allume. Grâce à cette page nous pouvons faire varier ce seuil à l'aide 2 boutons qui appelleront chacun leur url respectif.

Dans la vue Statistiques.dart, on retrouvera toutes les collections sur la luminosité et température issue de Firebase. Une liste de données est utilisée pour afficher toutes les informations que l'on a pu enregistrer sur le main.

5.6 Service

Le dossier service contient la classe database.dart, elle possède 2 fonctions: l'une permet d'enregistrer les données qui seront récupérées du format JSON et l'autre permet de créer une liste des données pour les afficher sur l'interface Statistiques.dart.

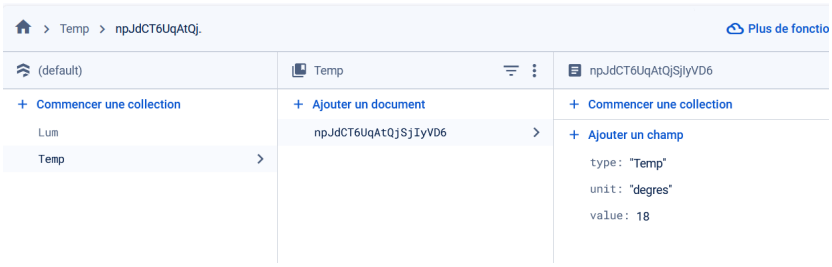

Figure 3: Firestore collection Temp

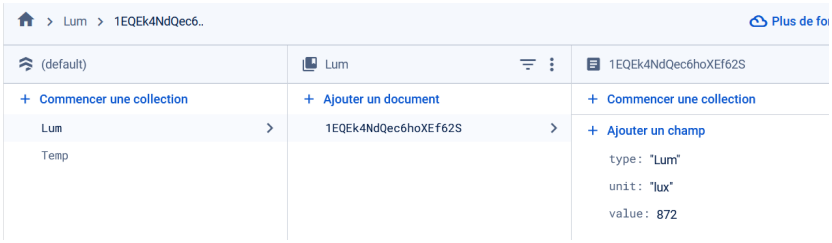

Figure 4: Firestore collection Lum

6 Interfaces

6.1 Main

Figure 5: Interface main

6.2 Seuil

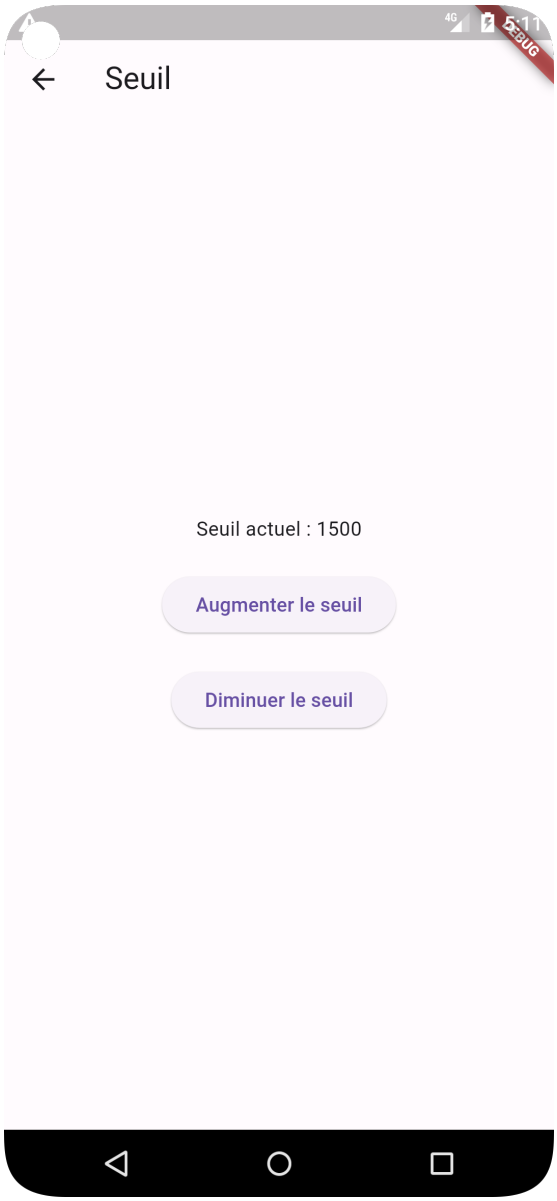

Figure 6: Interface seuil

6.3 Statistiques

Figure 7: Interface statistiques

7 Difficultés

Mauvaise organisation de ma part, j'ai pris un peu trop de temps à finir le TP3. La page statistique peut être améliorer niveau design. Le fait d'être tout seul sur ces projets et travaux pratiques m'a pris beaucoup du temps.

8 Conclusion

Pour conclure, ce projet m'a permis la mise en place et le développement de système embarqué et mobile. Toutes les fonctionnalités demandées ont été développé et fonctionnelles. Après avoir programmé sur le micro-contrôleur ESP32,l'utilisation de la photo résistance, thermo résistance. Il serait intéressant de se pencher vers de nouveaux outils comme les caméras.