How does the Flic Button work

Project 78: Integratie beacons with Social-Buddy

Algemene informatie

Project Code:	**CMI-TI-23-TINPRJ0478**
Project Goal:	'Enrich the Buddy App with a Flic button a Tile tag'
Team Members:	Ahmet Oral (1023107) - Author Khizer Butt (1052313) Nguyen Do (1057048) Terrence Zhong (1028516)
Skills docent:	Sandra Hekkelman
Technical Coach:	Alexander Slaa
Document Version:	1

Introduction

In order to integrate the Flic button with the buddy app we need a few packages, namely:

1. flic_button.dart
2. provider.dart

Permissions

Before we can work with the button we need to ask the users for certain permissions. You can find these in /android/src/main/AndroidManifest.xml. We need the following permissions:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
    <uses-permission android:name="android.permission.BLUETOOTH"
android:maxSdkVersion="30" />
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
android:maxSdkVersion="30" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

if these permissions are not accepted by the user the app can not connect to the app.

Flic_button.dart

Flic_button.dart is the package which we use to connect to the flic button. It has a few functions we use not only to connect to the button but also for example: maintain the connection, detect a button press etc. in the file flic_state.dart you will find the functions we use for the flic button.
ButtonState Class

The ButtonState class manages the state and functionality of Flic buttons.

Private Variables:

- **int _no**: A counter for the number of Flic buttons.
- **bool _isScanning**: Indicates if the app is currently scanning for Flic buttons.
- **FlicButtonPlugin? _flicButtonManager**: Manages Flic button operations.
- **Map<String, Flic2Button> _buttonsFound**: Stores found Flic buttons.
- **Flic2ButtonClick? _lastClick**: Stores the last Flic button click event.

Getters:

- **int get no**: Returns the counter for the number of Flic buttons.
- **bool get isScanning**: Returns whether the app is currently scanning for Flic buttons.
- **Map<String, Flic2Button> get buttonsFound**: Returns the map of found Flic buttons.
- **Flic2ButtonClick? get lastClick**: Returns the last Flic button click event.
- **FlicButtonPlugin? get flicButtonManager**: Returns the Flic button manager.

Constructor:

- **ButtonState()**: Initializes the class and starts/stops Flic button manager.

Public Methods:

- **void startStopFlic2()**: Initializes or disposes the Flic button manager.
- **void startStopScanningForFlic2()**: Starts or stops scanning for Flic buttons based on the current scanning state.
- **void getButtons()**: Retrieves all known Flic buttons and starts listening to them.
- **void connectDisconnectButton(Flic2Button button)**: Connects or disconnects a Flic button based on its current connection state.
- **void forgetButton(Flic2Button button)**: Forgets a Flic button and removes it from the list.

Private Methods:

- **void _startStopFlic2()**: Initializes or disposes the Flic button manager.
- **Future<void> _startStopScanningForFlic2() async**: Starts or stops scanning for Flic buttons based on the current scanning state.
- **void _getButtons()**: Retrieves all known Flic buttons and starts listening to them.
- **void _addButtonAndListen(Flic2Button button)**: Adds a Flic button to the list and starts listening to it.
- **Future<void> _connectDisconnectButton(Flic2Button button) async**: Connects or disconnects a Flic button based on its current connection state.
- **void _forgetButton(Flic2Button button)**: Forgets a Flic button and removes it from the list.
- **void _handleButtonClick()**: Handles button click events and logs them to Firestore.

Override Methods:

- **void dispose()**: Disposes the Flic button manager and cleans up resources.

_FlicListener Class:

The _FlicListener class listens for Flic button events.

Constructor:

- **_FlicListener(this._buttonState)**: Initializes the listener with the given ButtonState instance.

Override Methods:

- **void onButtonClicked(Flic2ButtonClick buttonClick)**: Handles button click events by calling _handleButtonClick in ButtonState.

Provider.dart

In our Flutter application, we use the Provider package to manage the state of the ButtonState class. The Provider package is a popular state management solution in Flutter that allows you to efficiently manage and update your application state.

Setting Up the Provider:

To set up the Provider for ButtonState, in your main.dart you wrap your root widget with ChangeNotifierProvider. This ensures that the ButtonState is available throughout the widget tree and can be accessed by any descendant widget.

```
runApp(
    ChangeNotifierProvider(
      create: (_) => ButtonState(),
      child: const MyApp(),
    ),
  );
```

ChangeNotifierProvider:

This is a special type of Provider that listens to changes in a ChangeNotifier and rebuilds any widgets that depend on it when notified.

- **create**: This parameter is a function that returns an instance of ButtonState. It initializes ButtonState and provides it to the widget tree.
- **child**: The child parameter is the root widget of your application, MyApp in this case. This widget and its descendants can now access the ButtonState.

Accessing ButtonState in FlicConnect Widget:

In the Flic_connect.dart widget, we use the Provider package to access the ButtonState instance. This allows the widget to interact with and display information about the Flic buttons.

In this code:

- **final buttonState** = Provider.of<ButtonState>(context);: This line accesses the ButtonState instance from the nearest ChangeNotifierProvider in the widget tree. This allows the widget to interact with the ButtonState and respond to changes.
- **buttonState.startStopFlic2()**, **buttonState.getButtons()**, **buttonState.startStopScanningForFlic2()**, **buttonState.connectDisconnectButton(e)**, and **buttonState.forgetButton(e)**: These methods are called on the buttonState instance to perform various actions related to Flic button management.
- The widget tree, including the **FutureBuilder**, **ElevatedButton**, **Text**, and **ListView** widgets, uses properties and methods from the buttonState to display and interact with Flic buttons dynamically.

Change Log

Version	Date	Changes
1	29-5-24	Version one
2	31-5-24	Changed the way the code is displayed