

SYP

Schuljahr 2024/2025

Dokumentation

Flutter Indoor-Navigation

Ausgeführt von:

Jasmin Mayer, 5AHIF
Simon Reiter, 5AHIF

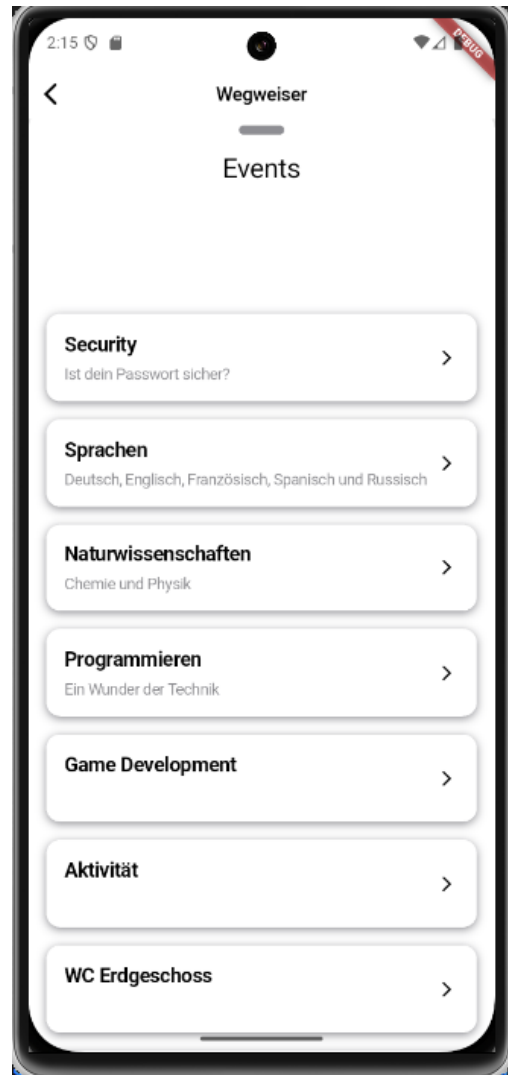
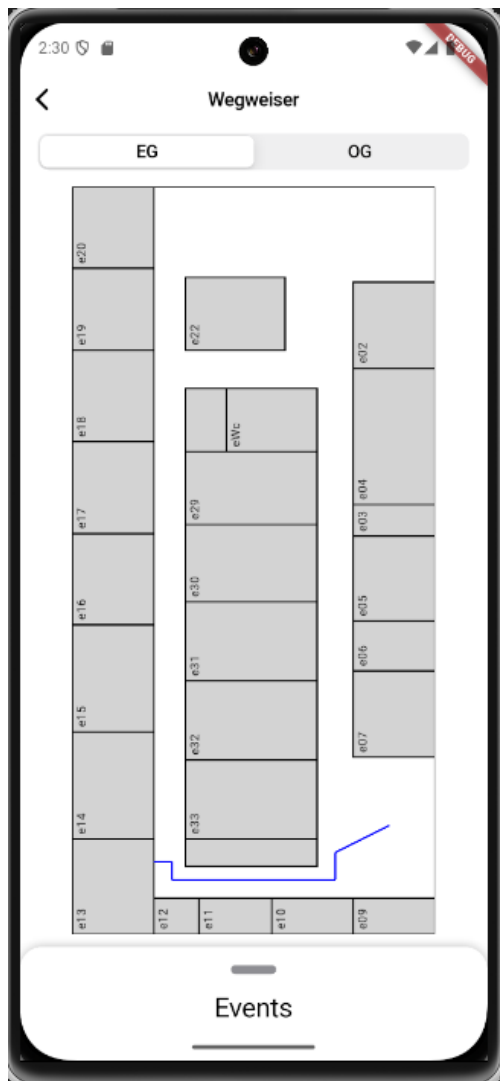
Grieskirchen, am 17.03.2025

Inhaltsverzeichnis

1	Ziel.....	3
2	Lösungsansätze.....	4
2.1	WLAN.....	4
2.2	GPS	4
2.3	AR	4
2.4	Bluetooth.....	4
3	Umsetzung und Ideen	5
3.1	Beacon.....	5
3.2	Positionierung	5
3.3	Wegpunkte Generator	5
4	Quellcode-Ausschnitte	6
4.1	Empfang von IBeacon-Paketen	6
4.2	Beacon konfigurieren	8
5	Probleme	9
5.1	WLAN.....	9
5.2	BLE Beacons.....	9
5.3	Beacon-Scanner.....	9
6	Ideen	10
6.1	Kombination	10
6.2	Kalibrierung.....	10

1 Ziel

Das Ziel der Applikation ist am Tag der offenen Tür eine Indoor-Navigation zu implementieren, welche Besucher einen Weg zu ausgewählten Events, wie Präsentationen und Vorstellungen, navigiert. Ebenfalls sollten die User eine Möglichkeit haben eine Übersicht der Schule zu haben und auf einen Blick auf den Grundriss der Schule mit den Klassen aktuelle Events zu sehen.



2 Lösungsansätze

Für die Navigation ist die Positionserkennung unerlässlich und kann durch folgende Methoden erfolgen.

2.1 WLAN

Diese Methode wäre gut geeignet für die Schule, da die Infrastruktur bereits besteht. Jedoch wurde sich aufgrund von 5.1 gegen die Umsetzung mit WLAN entschieden.

2.2 GPS

Eine weitere Möglichkeit wäre GPS. Unsere Sorge war, dass diese zu ungenau wäre. Eventuell könnte die Genauigkeit ausreichen, jedoch würde GPS für eine Kombination mit einer anderen Methode eher passen, als für eine eigenstehende Positionserkennung.

2.3 AR

Die Umsetzung der Navigation könnte auch auf AR basieren. Dazu gibt es Online ein paar Anleitungen. Dabei fand ich folgendes Video interessant:

<https://www.youtube.com/watch?v=G3khfWQUa78>

2.4 Bluetooth

Über Bluetooth können iBeacon-Pakete empfangen werden. Diese werden von Beacons ausgesendet. Dabei kann über Trilateration festgestellt werden, wo sich der Empfänger befindet, da die Sender (Beacons) eine bekannte Position haben.

3 Umsetzung und Ideen

3.1 Beacon

Als Beacon wurde ein ESP-32 verwendet. Um den Chip mit den Funktionen eines Beacons auszustatten muss eine Software geladen werden. Diese Software wird bereits von ESP-IDF bereitgestellt.

3.2 Positionierung

Für die Positionierung wurde das Paket flutter_beacon verwendet. Dieses funktioniert aber seit der Version **3.29.0** (12. Februar 2025) nicht mehr.
https://pub.dev/packages/flutter_beacon

3.3 Wegpunkte Generator

Da die Positionierung stark schwankt war die Idee, einen dynamischen Wegpunkt-Generator zu entwickeln, welcher automatisch in einer vorgegebenen Map, fixe Punkte einzeichnet. Diese Punkte haben eine Art Checkpoint-Funktion. Der Sinn darin ist, dass am Anfang der Navigation in einem gut kalibrierten Zeitintervall der Durchschnitt der schwankenden Position auf eine fixe Positon abzubilden. Das ist der Startpunkt der Navigation.

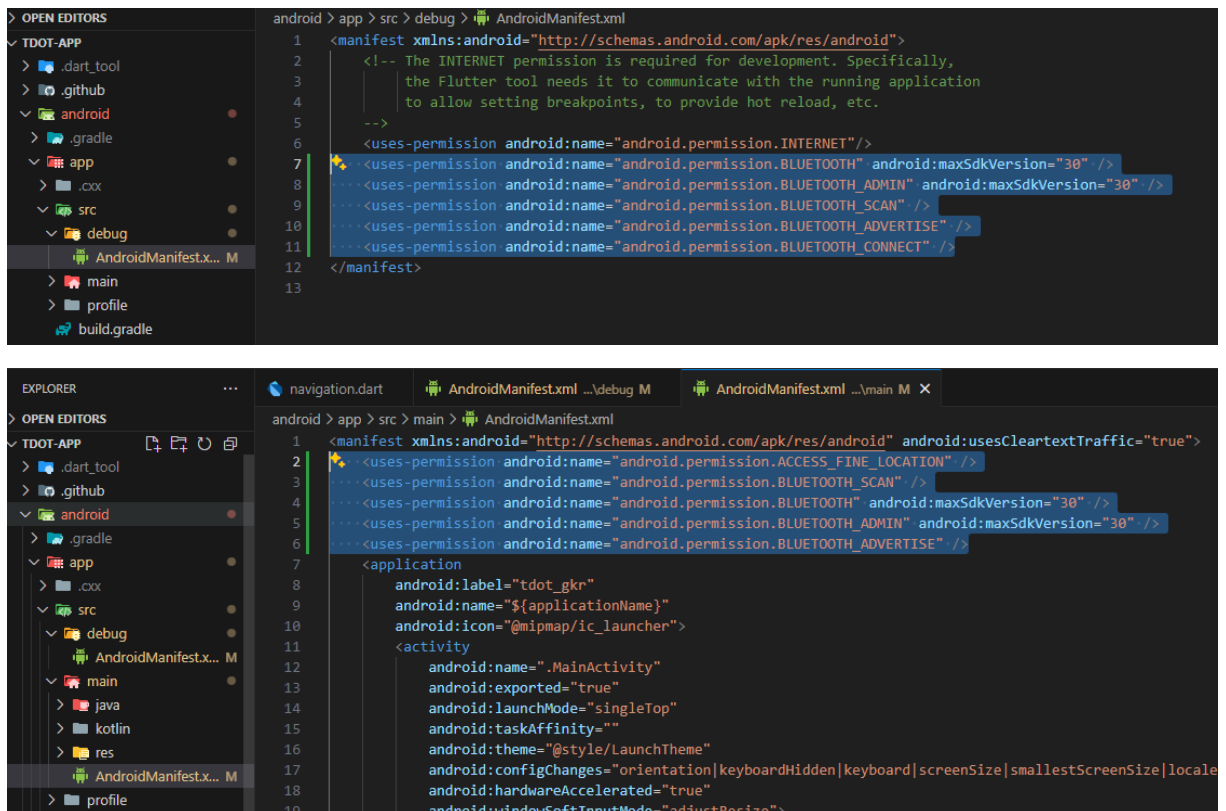
Damit der Navigationsweg nach vorne und wieder zurück springt, wird geschaut, ab wann eine gewisse Anzahl an Checkpoints erreicht wurden. Wenn das der Fall ist, wird der Navigationsweg ab den Checkpoint neugezeichnet.

4 Quellcode-Ausschnitte

4.1 Empfang von IBeacon-Paketen

Am Anfang wird der Beacon initialisiert und die Berechtigungen abgefragt, um auf die benötigten Funktionen zugreifen zu können. Für das wurde das Paket `permission_handler: ^11.3.1` verwendet.

Diese Permissions müssen, wie bereits aus AndroidStudio bekannt, im `AndroidManifest.xml` eingetragen werden. Weitere Berechtigungen, oder welche für OS-Betriebssysteme benötigt werden stehen jeweils in den README-Dateien der Flutter Beacon-Bibliotheken.



```
android > app > src > debug > AndroidManifest.xml
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android">
2   <!-- The INTERNET permission is required for development. Specifically,
3        the Flutter tool needs it to communicate with the running application
4        to allow setting breakpoints, to provide hot reload, etc.
5   -->
6   <uses-permission android:name="android.permission.INTERNET"/>
7   <uses-permission android:name="android.permission.BLUETOOTH" android:maxSdkVersion="30" />
8   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" android:maxSdkVersion="30" />
9   <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
10  <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
11  <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
12 </manifest>
13
```

```
android > app > src > main > AndroidManifest.xml
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:usesCleartextTraffic="true">
2   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
3   <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
4   <uses-permission android:name="android.permission.BLUETOOTH" android:maxSdkVersion="30" />
5   <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" android:maxSdkVersion="30" />
6   <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
7   <application
8     android:label="tdot_gkr"
9     android:name="${applicationName}"
10    android:icon="@mipmap/ic_launcher">
11     <activity
12       android:name=".MainActivity"
13       android:exported="true"
14       android:launchMode="singleTop"
15       android:taskAffinity=""
16       android:theme="@style/LaunchTheme"
17       android:configChanges="orientation|keyboardHidden|keyboard|screenSize|smallestScreenSize|locale
18       android:hardwareAccelerated="true"
19       android:windowSoftInputMode="adjustResize">
```

Um iBeacons zu empfangen wird die Methode „ranging“ verwendet und mit „listen“ geschaut, ob ein iBeacon-Paket empfangen wird. Diese Method benötigt „Regions“. Diese sind vom Typ Region und haben einen Identifikator. Dieser ist in unserem Fall com.beacon, da wir nur iBeacons empfangen möchten.

```
final regions = <Region>[  
  Region(identifizier: 'com.beacon'),  
]; // <Region>[]
```

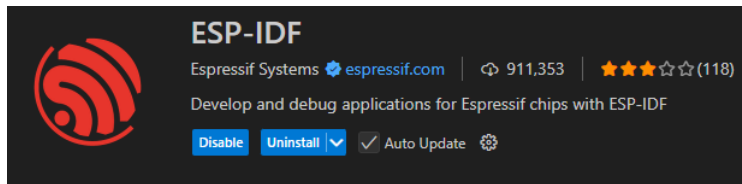
Wenn nun ein iBeacon empfangen wird kann man aus dem „result“ die Beacons auslesen und daraus Informationen wie die UUID, Distanz (accuracy), usw. auslesen. In diesem Codeausschnitt wird geprüft, von welchen Beacons ein Paket empfangen worden ist und die Distanz aktualisiert. Dafür wurde zuvor eine Liste von UUID's der vorhandenen Beacons erstellt.

Von der HTL wurden sechs ESP32 zur Verfügung gestellt. Diese wurden mit folgenden UUID's ausgestattet.

```
// final Map<String, Coordinate> beaconCoordinates = {  
//   '54C30B7D-6B88-4BB6-AEDA-A157C07476BA': Coordinate(x: 0, y: 150),  
//   'B1291BC1-5CC2-402E-854C-CBC111F67295': Coordinate(x: 0, y: 0),  
//   'A8A7DC3D-5EB1-4DE4-B020-62BD11DAC3FE': Coordinate(x: 200, y: 0),  
//   'BDB02063-BC84-4A7A-B9B4-A6FCD8691EC9': Coordinate(x: 0, y: 150),  
//   '9D90F6C9-CF5E-4222-80F7-177C080A0D4D': Coordinate(x: 0, y: 0),  
//   '4041FCF1-3C8B-4948-B096-0804CAF87341': Coordinate(x: 200, y: 0),  
// };  
  
void startBLE() async {  
  try {  
    await flutterBeacon.initializeAndCheckScanning;  
    await Permission.bluetoothScan.request();  
    await Permission.bluetoothAdvertise.request();  
    await Permission.location.request();  
    await Permission.bluetoothConnect.request();  
    flutterBeacon.ranging(regions).listen((RangingResult result) {  
      setState(() {  
        text = '';  
  
        text = result.beacons.toString();  
        text = result.region.toString();  
        text += '\n';  
        text += result.beacons.toString();  
        text += '\n';  
  
        for (var x in result.beacons) {  
          for (int i = 0; i < beaconCoordinates.keys.length; i++) {  
            if (x.proximityUUID.toString() ==  
                beaconCoordinates.keys.elementAt(i)) {  
              distances[i] = x.accuracy * 100;  
              text += '\n d${i + 1}: ${x.accuracy * 100}';  
            }  
          }  
        }  
  
        calcPosLine();  
      });  
    });  
  } on PlatformException catch (e) {  
    print('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ error: $e');    Don't invoke  
  }  
}
```

4.2 Beacon konfigurieren

Folgende Extension wird in Visual Studio Code hinzugefügt.



Danach wird die Anleitung befolgt und benötigte Inhalte heruntergeladen.

Anschließend wird das Beispiel-Projekt für einen IBeacon verwendet, welches von Espressif auf Github bereitgestellt wird.

https://github.com/espressif/esp-idf/tree/master/examples/bluetooth/bluedroid/ble/ble_ibeacon

Wenn man einen Beacon angeschlossen hat wird ganz links (unserem Fall „COM6“) der Port ausgewählt, an welchem der Beacon am PC angeschlossen wurde. Wenn man nun ein Programm auf den Beacon schreiben möchte wird das ganz rechte Symbol (Flamme) geklickt.



Im kopierten Projekt von Espressif werden im File „esp_ibeacon_api.h“ die Parameter eingestellt. Dabei wird 1 beim Sender eingetragen und 0 beim Receiver, da ein Beaconsender benötigt wird. Des Weiteren wird die gewünschte UUID des Beacons bei „ESP_UUID“ eingetragen. Nachdem alle Einstellungen vorgenommen wurden kann das Programm über „Build, Flash and Monitor“ auf den Beacon übertragen werden (ganz rechtes Symbol: „Flamme“).

```
#define IBEACON_SENDER 1
#define IBEACON_RECEIVER 0
#define IBEACON_MODE 0

/* Major and Minor part are stored in big endian mode in iBeacon packet,
 * need to use this macro to transfer while creating or processing
 * iBeacon data */
#define ENDIAN_CHANGE_U16(x) (((x)&0xFF00)>>8) + (((x)&0xFF)<<8)

/* Espressif WeChat official account can be found using WeChat "Yao Yi Yao Zhou Bian",
 * if device advertises using ESP defined UUID.
 * Please refer to http://zb.weixin.qq.com for further information. */
#define ESP_UUID {0xFD, 0xA5, 0x06, 0x93, 0xA4, 0xE2, 0x4F, 0xB1, 0xAF, 0xCF, 0xC6, 0xEB, 0x07, 0x64, 0x78, 0x25}
#define ESP_MAJOR 10167
#define ESP_MINOR 61958
```


5 Probleme

5.1 WLAN

Es besteht bereits eine Diplomarbeit, welche sich mit dieser Methode beschäftigt hat. Dabei wurde herausgefunden, dass der Scan-Intervall zu lange dauert.

Bei weiterer Recherche und Programmierung konnte dies bestätigt werden.

Dabei konnten folgende Drosselungen von Android gefunden werden.

- Android 8.0 and Android 8.1:
Jede Background-App kann einmal pro 30 Minuten scannen.
- Android 9 and later:
Jede Foreground-App kann 4 mal pro 2 Minuten scannen.
Alle Background-Apps zusammen können einmal pro 30 Minuten scannen.

Informationen zur Drosselung und weitere Bedingungen, wie welche Berechtigungen benötigt werden können unter folgendem Link nachgelesen werden:

<https://developer.android.com/develop/connectivity/wifi/wifi-scan?hl=de>

5.2 BLE Beacons

Bei klarer Sicht auf die Beacons sind Distanzen relativ akkurat. Dabei war es jedoch wichtig, dass richtige Flutter-Paket zu verwenden, welche Distanzberechnungen bereits zur Verfügung stellt, oder eine genügend gute Kalkulation zu finden.

Sobal aber etwas die Sicht auf den Beacon blockiert, ob Mensch oder Wand, wurde die Distanz steigend inakkurat.

5.3 Beacon-Scanner

Das Paket welches für die Entwicklung verwendet wurde, hat ab der neuesten Flutter-Version nicht mehr funktioniert, da es veraltete Funktionen benutzt.

https://pub.dev/packages/flutter_beacon

Eigentlich war dieses Paket gut geeignet und hatte die Distanzberechnung bereits integriert und war sehr einfach zu implementieren. Eventuell bieten die aktuelleren Pakete wie `flutter_blue_plus` oder `beacon_flutter` eine gute Alternative. `Flutter_blue_plus` ist zwar nicht spezifisch für Beacons ausgelegt, aber man bekommt trotzdem genügend Werte, um eine akkuratere Positionierung zu ermöglichen.

https://pub.dev/packages/flutter_blue_plus

https://pub.dev/packages/beacon_flutter

6 Ideen

6.1 Kombination

Um die Genauigkeit und das Benutzerempfinden zu verbessern könnte eine Kombination von mehreren Methoden entwickelt werden. Eine solche Kombination könnte WLAN oder GPS mit BLE-Beacons sein, um die Genauigkeit zu erhöhen.

Die Verwendung von Handysensoren wie der Beschleunigungsmesser oder der Gyrosensor kann die Entwicklung erleichtern. Dabei könnten vergangene Distanzen über den Beschleunigungsmesser gemessen und mit den Entfernungen der Positionierungsmethoden verglichen werden, um den Fehler zu minimieren.

Mit dem Gyrosensor können Handyausrichtungen gemessen werden. Dies hat den Vorteil, dass man weiß, in welche Richtung der Besucher geht. Ebenfalls kann die Handyausrichtung mit der Ausrichtung der Schulmap abgeglichen werden und die Map sogar nach der Ausrichtung des Handy's gereht werden.

6.2 Kalibrierung

Eine Kalibrierung der Beacons wäre eine eventuelle Lösung oder eine Minimierung des Risikos von 4.2. Eine Idee wäre, eine App zu entwickeln, mit welcher man die durchschnittlichen Distanzen zu einem Beacon abspeichern kann.

Der Ablauf könnte wie folgt aussehen. Ein Teammitglied stellt sich in die Mitte des Ganges auf Höhe eines Beacons (z.B.: neben einer Klassentür, wenn Beacon an Wlan-AP angeschlossen ist). Danach wird nach Signalen von diesem Beacon gescannt und über eine gewisse Zeit die Distanzen als Durchschnitt zusammengefasst abgespeichert. Diese kalibrierten Distanzen könnten über eine Schnittstelle in der Hauptapp integriert werden, sodass auch nachher noch die Möglichkeit besteht die Kalibrierung anzupassen.

Deshalb ist es nicht empfehlenswert die Kalibrierung direkt an der Beacon-Software selbst durchzuführen.