特別実験 最終報告書

報告者：電子・情報システム工学専攻

10 番　　山﨑　敦史

共同実験者：

11 番　　弓削　林太郎

8 番　　福田　有紗

提出日：令和 4 年 2 月 14 日

1. 目的

本実験では、教員が学内のどこにいるかを突き止める位置情報システムの開発を目標とした。作成に至った背景として、実際に私たち学生がレポートの提出や教員への質問を行う際に研究室に訪問することがある。その際、教員が外出や会議などで研究室から離れており教員と学生の間ですれ違いが度々あるため教員が研究室に滞在しているかを確認できるアプリケーションを作成した。また、シラバスを踏まえて特別実験を通じて達成するべき目標は以下のとおりである。

・Flutter の基本的な知識を身につける。
・Firebase の基本的な知識を身につける。
・Slack の活用方法を習得する。
・Miro の活用方法を習得する。
・GitHub の基本的な使い方を習得する。
・スケジュールを設定し、計画的に実験を進める能力を身につける。
・問題を明確にとらえ、最も適切な解決策や方法を見つける能力を身につける。
・チームワーク力を発揮して問題解決を行う能力を身につける。

2. 方法

本実験で計画したスケジュールは以下の通りである。

表 1　スケジュール表

作業＼日付	9/24	10/1	10/8	10/15	10/22	11/5	11/12	11/19	11/26	12/23	12/10	12/17	12/23	1/14	1/21	1/28
制作物の検討																
コンテストの検討																
Flutterを学ぶ																
バックグラウンドの整備																
画面の作成																
APIの作成																
デザインの改良																
アプリアイコンの作成																
システム全体の結合																
報告書の作成																

本実験で使用したハードウェアは以下の通りである。

・ 開発用パソコン（Windows・Linux）
・ Bluetooth ビーコン 「FeasyBeacon FSC-BP-104」

また使用したソフトウェアは以下の通りである。

・　スマートフォンアプリ開発フレームワーク「Flutter & Dart」
・　Google Firebase　（クラウドサービス）
・　Git, GitHub　（ソースコード管理）
・　Miro　（デザイン・アイデア・作業管理）

3.　結果と考察

3.1.　システム全体の概要

本研究では Bluetooth ビーコンにより教員の位置を特定し、他ユーザーにインターネットを通して共有を行う。Bluetooth ビーコンを学校内の研究室などに配置することで高度の取得が不可能な GPS の代わりに位置情報の特定を可能とする。

本実験で作成したスマートフォンアプリ（以下アプリと呼ぶ）は位置情報を提供する教員とそれを閲覧する教員以外のユーザーに分かれる。

まず、システムを教員として利用する場合はユーザー登録を必要とする。閲覧のみおこなうユーザーは登録不要である。

以下アプリの使用手順を説明する。

図 1 はアプリ起動時の画面である。

左上の三本線のアイコンを押下すると、図 2 のように表示される。されに「教員登録」を押下すると図 3 の登録画面が表示される。

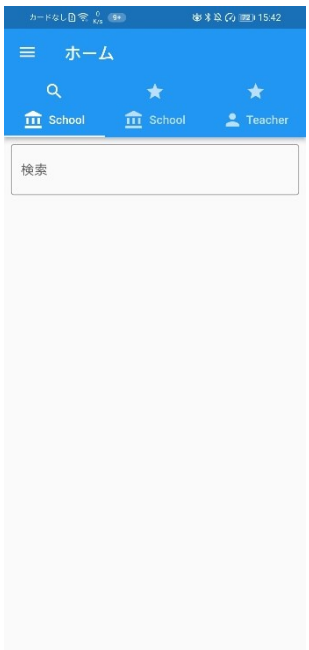

図 1　ホーム画面

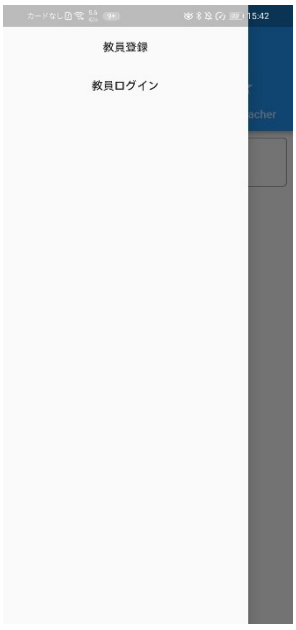

図 2　ドロワーの表示

図 3　教員登録画面

ここでメールアドレスとパスワードを入力し、認証システムにデータを送る。

認証システムとして FirebaseAuth を使用して行っている。

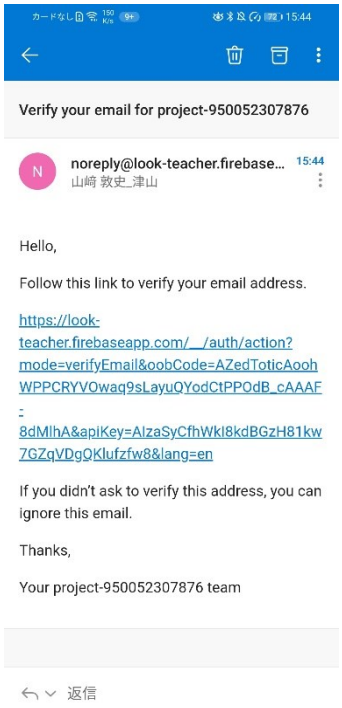

図４　登録後の画面　　図５　認証メール　　図６　教員専用ドロワー

正しく認証システムにデータが送られた場合、図４の画面が表示され、登録したメールアドレスにメールアドレスが正しいかどうかを判別するための URL リンクが図５のように送られる。このリンクにアクセスすると図６のように教員専用のドロワーが表示される。

現段階では教員名が登録されていないため、「ユーザープロファイル設定」ボタンを押下し設定する。押下すると図７に示す画面が表示され、教員名がないことがわ

図７　ユーザープロファイル画面　　図８　プロファイル編集画面　　図９　教員専用ドロワー

かる。図７右上の「編集」ボタンを押下すると図８に示す編集画面が表示される。ここで教員名と位置情報を取得するための Bluetooth スキャンの編集を行える。図８の決定ボタンを押下すると図９に示す図７と同じ画面に戻り、教員名の設定が反映されていることがわかる。

Bluetooth ビーコンと研究室情報を結びつける登録作業は教員ユーザーが行う必要がある。また、ビーコン情報と研究室情報は学校と結びつける必要があるため、教員ユーザーはシステム上にある学校に登録されなければならない。

現段階ではシステム上に学校が一つも存在しないため、図 6 の「学校を作る」ボタンにより学校をシステム上に作成する。ボタンを押下すると、図 10 のような画面が出現する。ここに学校名を入力し「登録」ボタンを押下するとシステムに学校が作成され、その学校に教員兼管理者として登録（所属）される。図 11 に示すホーム画面に、学校が新たに作成されたことが確認できる。

図 13　Bluetooth 検出結果

図 14　デバイス登録画面

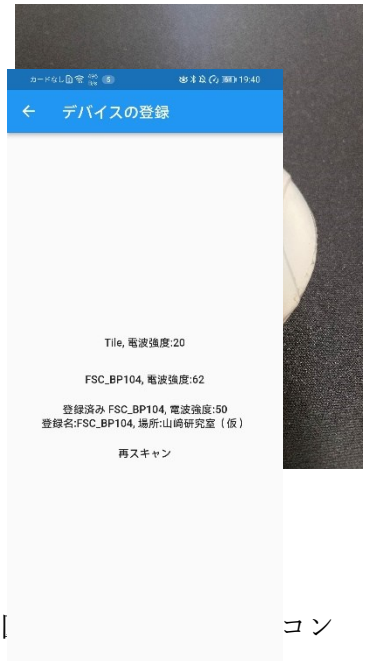

図 15　登録後 Bluetooth 検出結果

次に Bluetooth ビーコンを「津山工業高等専門学校」のデバイスとして登録する。図 6 の「Bluetooth デバイスの登録」ボタンを押下すると周囲の Bluetooth デバイスの検出処理が開始する。図 12 に実験で使用した Bluetooth ビーコンを示す。実験ではこのビーコンを 2 つ用意した。処理が終了すると検出したデバイスを表示する。図 13 に示すように 2 つのビーコンが正しく検出したことが確認できた。ビーコンの表示はボタンになっているため、押下すると図 14 に示す登録画面が表示される。ビーコンが設置される部屋名「山崎研究室（仮）」を入力し、「登録」ボタンを押下すると登録が完了する。再び Bluetooth の検出を行うと図 15 に示すようにビーコンが登録済みとなっていることが確認できる。同様にして 2 つ目のビーコンを「弓削研究室（仮）」と登録する。

図 16 のように教員がいる部屋を特定することができる。

図 16　教員情報の表示

ここで、教員ユーザーの位置情報を取得する方法を説明する。

いずれかの学校に登録されていて Bluetooth スキャンを有効にしている教員ユーザーは 40 秒に 1 度、周囲の Bluetooth デバイスを検出（以下定期スキャンとする）するプログラムが動作する。Bluetooth デバイスには固有の ID が割り当てられていて、デバイスの登録ではその ID とデバイス名、デバイスのある場所を紐づけて教員ユーザーが所属する学校のデバイスリストに追加する。

定期スキャンで検出した Bluetooth デバイスを学校のデバイスリストにあるものだけを残し、その中で最も近い（電波強度が強い）ものを教員の位置情報を示すデバイスとして紐づけ定期スキャンした時間とともに FirebaseFirestore に格納する。

これにより、教員の位置情報を示すデバイスから学校のデバイスリストを参照することで、部屋の名前がわかるようになっている。

3.2.　メンバーの担当分担

本実験を進める前の段階で計画していた役割については以下の通りである。

- 山崎：実験の計画を立て、Flutter の説明の役割を担当
- 弓削：プログラム作成の補助や改善案の提案の役割を担当
- 福田：画面の作成を行う役割

3.3.　作成過程

ここでは作成したシステムの作成過程を表 2 に記述する。

表 2 システムの作成過程

日付	実施内容
9/24	制作物の検討を行い、該当するコンテストについて調べる。
10/1	出場コンテストの決定、開発環境の構築と GitHub の操作を学ぶ。
10/8	Flutter の操作を学ぶ。
10/15	作成する UI の決定、Flutter の Widget について学ぶ。
10/22	BLE beacon の動作確認を行う。

11/5	Firebase の設定を行う。
11/12	報告書に書く内容を決定する。
11/19	アカウント登録画面の作成、Firebase とのコントローラーの作成、Bluetooth のバックグラウンド処理の確認。
11/26	教員一覧画面の作成、バックグラウンドでアプリの通知を送る処理の作成。
12/3	作成する画面の検討、画面の作成。
12/10	画面の作成、処理簡単化のための API の作成。
12/17	12/10 の作業の継続。
12/23	画面の作成、UI の改良。
1/14	休校のため、報告書の作成
1/21	アプリアイコンの作成、システム全体を結合。
1/28	報告書の作成

3.4. 工夫点などその他

アプリを製作は、情報のリアルタイム性を重視して行った。教員の情報を逐一更新し表示することで閲覧ユーザーの不安を解消できると考えたためである。FirebaseFirestore の機能と Flutter のパッケージライブラリである riverpod や hooks でリアルタイム性を保つ容易にできた。

また、ユーザーが困ることがないようにチームで議論した。「学校を探す手間が減れば良いね」と私が発言すると、「お気に入り機能があれば楽そう」と弓削の発言と、「検索機能があれば探す手間が省かれる」と福田の発言より次の機能を作成した。

- 学校検索機能
- 学校お気に入り機能
- 教員お気に入り機能

自分の役割として実験の計画や工期を見積もることと Flutter 開発を指導した。班員の能力からどれくらいの作業を割り当てられるかを考えることが難しかった。これは、実験全体のタスクを定義し、タスクを処理するごとに新しいタスクを割り当てる方法で対処した。Flutter を教えるときには目的を先に説明したり、実装例を先に教えることやどこまで理解しているかを確認することを意識した。

開発経験に差のあるメンバーでの短期間開発だったため、API 作成や画面作成の分担を図 17 のように Miro を使って行ったため完成させることができたと思った。

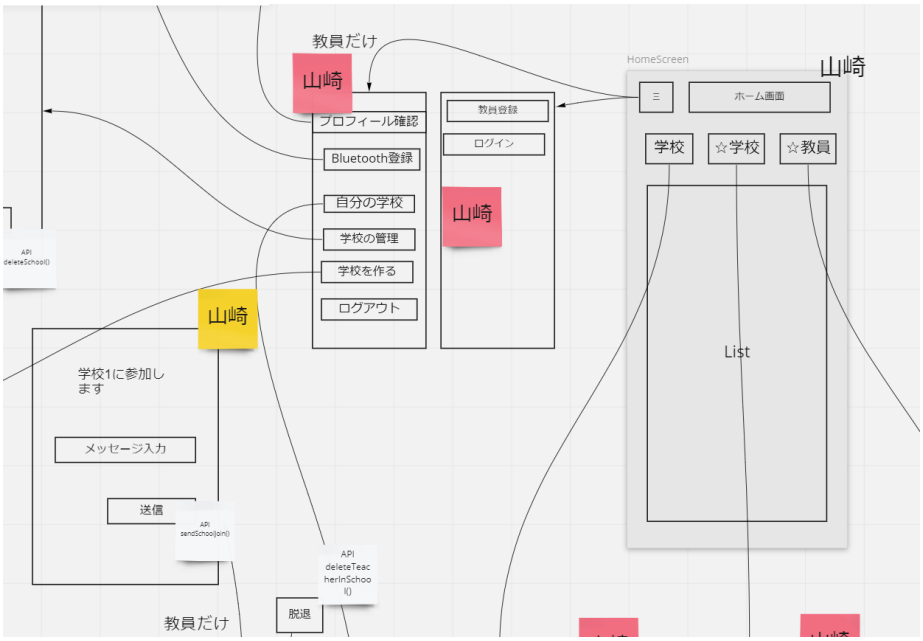

図 17　Miro による画面と API の管理図

4.　まとめ

本実験ではスマートフォンアプリを用いた教員の位置情報を閲覧するシステムの開発を達成できた。ビーコン設置やアプリが普及すれば、教員を探す時間を短くすることができる。開発経験に差のあるチームで上手く完成まで上手く進められたと思った。班全員が Flutter でアプリの画面が作成できるようになった。

アプリの懸念点として、テスト運用しか行っていないことがあげられる。セキュリティやプライバシーの観点をもう少し検討できたらより良いアプリになるだろうと思った。また、AppStore や Google Play Store にアップロードできるようにしたいと思った。作成したソースコードは Teams の B チームフォルダーの look_teacher 内の lib に保存されている。

5.　付録

本実験で作成したプログラムを一部記述する。

付録 1　Bluetooth デバイスを検出
フィルタリングするプログラム

```
import 'dart:developer';

import 'package:flutter_blue/flutter_blue.dart';
```

```
Future<Map<BluetoothDevice, int>> bluetoothScan() async {
  final deviceMap = <BluetoothDevice, int>{};
  final FlutterBlue flutterBlue = FlutterBlue.instance;

  // 現在スキャン中かチェック
  if (await flutterBlue.isScanning.first) {
    await flutterBlue.stopScan();
  }
  log('start scan');
  await flutterBlue.startScan(timeout: const Duration(seconds: 5));

  // スキャン結果を使う
  flutterBlue.scanResults.listen((scanList) {
    for (final result in scanList) {
      // 名前のあるものだけ deviceMap に追加する
      if (result.device.name != '') {
        deviceMap.addEntries({MapEntry(result.device, result.rssi)});
      }
    }
  });
  await flutterBlue.stopScan();
  log('stop scan');

  // スキャンが成功したとき deviceMap を返却する
  return deviceMap;
}

// 最も近くにある device を取得する
Future<String?> getNearestDeviceId(List<String> filterIdList) async {
  // スキャン結果を取得する
  final Map<BluetoothDevice, int> deviceMap = await bluetoothScan();

  // スキャンが成功しているとき。成功しているときは deviceMap が null でない
  if (deviceMap != null) {
    //filterIdList(school.deviceList の deviceId)に含まれない device を削除する
    deviceMap.removeWhere((device, rssi) {
```

```
                return !filterIdList.contains(device.id.toString());
            });

            log('filtered device');
            log(deviceMap.toString());

            // deviceMap が空出ないとき
            if (deviceMap.isNotEmpty) {
              // 最も近いデバイスを保持する変数の作成
              // deviceMap の最初の device の id を初期値とする
              String nearestDeviceId = deviceMap.keys.first.id.toString();

              // rssi（電波強度）の最弱値=-100 を初期値とする
              int nearestRssi = -100;

              // deviceMap すべての要素を引き出し、処理を行う
              deviceMap.forEach((device, rssi) {
                // 最も近いデバイスと deviceMap の要素のどっちが電波強度が強いか比較
                if (nearestRssi <= rssi) {
                  // 要素のほうが強いとき、その要素を最も近いデバイスとする
                  nearestRssi = rssi;
                  nearestDeviceId = device.id.toString();
                }
              });
              return nearestDeviceId;
            }
          }
}
```