SmartCompass

Inteligentne urządzenie do nawigacji

Dominik Cybulski - 255520@student.pwr.edu.pl
Michał Durkalec - 263917@student.pwr.edu.pl
Stanisław Kurzyp - 264477@student.pwr.edu.pl

Prowadzący: prof. Robert Burduk
Politechnika Wrocławska
Semestr letni 2024

Spis treści

Część I

Zarządzanie projektem

1 Założenia projektowe

Celem projektu będzie zaprojektowanie i zbudowanie urządzenia służącego do nawigacji pieszej. Urządzenie będzie składało się z mikroprocesora z modułem Bluetooth, modułu GPS, wyświetlacza LCD oraz układu zasilającego. Konfiguracja będzie odbywała się za pomocą aplikacji mobilnej, w której możliwe będzie zaplanowanie trasy i wgranie jej do urządzenia. Po skonfigurowaniu trasy, urządzenie będzie wskazywało kierunek i odległość do następnego punktu (na wyświetlaczu LCD).

1.1 Wymagania projektowe

Wymagania projektowe zostały określone za pomocą tablicy MoSCoW:

- **Must have:**
  - Urządzenie poprawnie wskazuje kierunek i odległość do kolejnych punktów trasy.
  - Aplikacja pozwala na zaplanowanie trasy na mapie i wgranie jej do pamięci urządzenia.
  - Po konfiguracji urządzenie działa autonomicznie, bez potrzeby komunikacji z aplikacją.
- **Should have:**
  - Bateria pozwala na całodzienne korzystanie z urządzenia (około 8 godzin).
  - Urządzenie implementuje mechanizmy zmniejszające zużycie energii (np. wygaszanie ekranu).
  - Aktualny stan nawigacji jest zapisywany w pamięci nieulotnej, co umożliwia wznowienie korzystania z urządzenia po utracie zasilania.
- **Could have:**
  - Urządzenie zapisuje aktualne położenie i pozwala na zgranie przebiegu trasy do aplikacji.
  - Aplikacja wyświetla historię przebytych tras razem z czasami przejścia.
  - Realizacja urządzenia na samodzielnie zaprojektowanej płytce PCB.
- **Won't have:**
  - Urządzenie nie posiada interaktywnego interfejsu użytkownika, wyświetla jedynie aktualny stan trasy.

2 Ryzyka projektowe

Realizacja projektu wiąże się z kilkoma ryzykami, które zostały zidentyfikowane i ocenione pod względem prawdopodobieństwa wystąpienia oraz wpływu na projekt. Poniżej przedstawiono główne ryzyka oraz planowane działania w celu ich minimalizacji.

2.1 Odejście członków zespołu

- **Prawdopodobieństwo:** Bardzo niskie
- **Wpływ:** Bardzo niski
- **Plan działania:** Akceptacja ryzyka ze względu na niskie prawdopodobieństwo i wpływ.

2.2 Ograniczony budżet

- **Prawdopodobieństwo:** Średnie
- **Wpływ:** Wysokie
- **Plan działania:** Opracowanie szczegółowego kosztorysu oraz pozyskanie dodatkowych środków finansowych od Politechniki.

2.3 Opóźnienia w dostawie komponentów

- **Prawdopodobieństwo:** Średnie
- **Wpływ:** Średnie
- **Plan działania:** Uwzględnienie potencjalnych opóźnień w harmonogramie realizacji oraz opracowanie alternatywnych planów montażu.

2.4 Opóźnienia w montażu układu (PCB)

- **Prawdopodobieństwo:** Średnie
- **Wpływ:** Średnie
- **Plan działania:** Uwzględnienie potencjalnych opóźnień w harmonogramie realizacji oraz opracowanie alternatywnego planu montażu.

2.5 Niewystarczające umiejętności

- **Prawdopodobieństwo:** Średnie
- **Wpływ:** Średnie
- **Plan działania:** Akceptacja ryzyka oraz regularne szkolenia i konsultacje w zespole.

2.6 Skomplikowana dokumentacja techniczna komponentów

- **Prawdopodobieństwo:** Średnie
- **Wpływ:** Niskie
- **Plan działania:** Wybór innych komponentów lub korzystanie z nieoficjalnych dokumentacji i wsparcia społeczności online.

3 Harmonogram projektu

Projekt został podzielony na trzy główne etapy:

- **Etap I - Projektowanie** (do 8 kwietnia 2024)
  - Opracowanie listy komponentów elektronicznych.
  - Analiza i wybór komponentów.
  - Projekt układu elektronicznego.
  - Wybór technologii programowania aplikacji mobilnej.
  - Projektowanie widoków aplikacji mobilnej.
  - Opracowanie protokołu komunikacji między urządzeniem a aplikacją.
- **Etap II - Prototypowanie i testowanie** (do 6 maja 2024)
  - Zakup komponentów.
  - Montaż układu na płytce prototypowej.
  - Programowanie urządzenia i aplikacji mobilnej.

  - Testy komunikacji między aplikacją a urządzeniem.
- **Etap III - Montaż** (do 27 maja 2024)
  - Montaż układu w wybranej technologii.
  - Budowa obudowy urządzenia.
  - Przygotowanie wersji produkcyjnej aplikacji mobilnej.

4 Zarządzanie zespołem

4.1 Podział zadań

Przed rozpoczęciem projektu został ustalony następujący podział zadań:

- **Dominik Cybulski, Michał Durkalec:**
  Projektowanie układu elektronicznego, programowanie urządzenia.
- **Stanisław Kurzyp:**
  Projektowanie aplikacji mobilnej.

4.2 Planowanie pracy

Do planowania poszczególnych zadań oraz monitorowania postępów wykorzystane zostało narzędzie Github Projects [7], które pozwala na

- Tworzenie zadań i przypisywanie ich do członków zespołu.
- Planowanie terminów realizacji zadań.
- Monitorowanie postępów prac.
- Łączenie zadań z konkretnymi gałęziami kodu w systemie kontroli wersji.

Część II

Implementacja

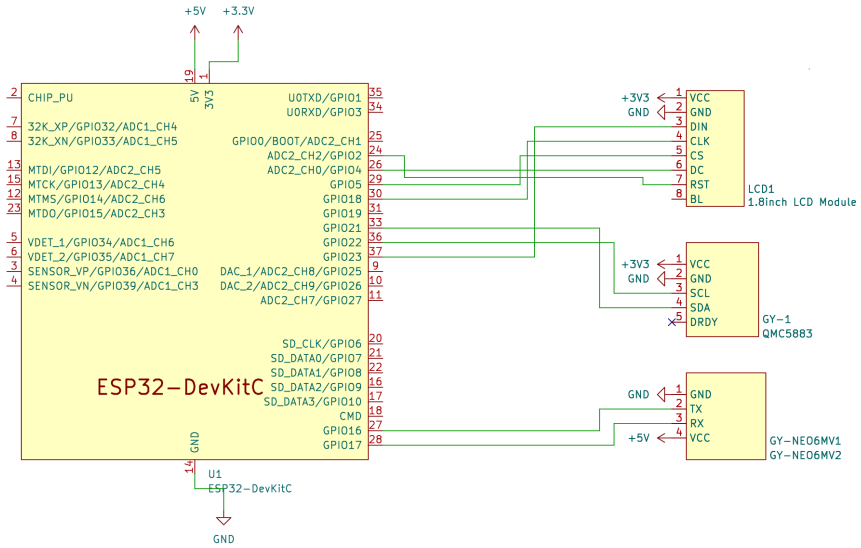
5 Projekt urządzenia

Urządzenie zostało zaprojektowane z myślą o maksymalnej prostocie, w związku z czym nie występuje żadna magistrala, a wszystkie moduły podłączone są wedle architektury punkt-punkt. Głównym elementem urządzenia jest płytka prototypowa **DevKitV4 ESP-WROOM-32E**, zawierająca zaprogramowaną logikę sterowania oraz wbudowany moduł Bluetooth/WiFi.

Do urządzeń peryferyjnych należą:

- Wyświetlacz LCD Waveshare 1.8inch 128x160
- Magnetometr cyfrowy 3-osiowy HMC5883L
- GPS GY-NEO6MV2 z anteną

Schemat podłączenia wszystkich elementów zaprezentowano na rysunku 1.



Rysunek 1: Schemat prototypu urządzenia

5.1 Kosztorys prototypu

Kosztorys uwzględnia ceny głównych modułów urządzenia. Dotychczasowe koszty zostały pokryte przez członków grupy projektowej. Posiadamy również inne niezbędne części podstawowe, takie jak rezystory, kondensatory czy przyciski, które będą niezbędne przy testowaniu prototypu.

Nazwa części	**Typ**	**Cena detaliczna**	**Faktyczny koszt**
Espressif DevKitV4	ESP-WROOM-32E	35,99 zł	35,99 zł
GY-273	Magnetometr cyfrowy 3-osiowy	17,00 zł	13,71 zł
GY-NEO6MV2	Moduł GPS z anteną	27,99 zł	27,99 zł
Waveshare 13892	Wyświetlacz LCD 128x160px	33,90 zł	0,00 zł
Razem		**114,88 zł**	**77,69 zł**

Tabela 1: Kosztorys prototypu

5.2 DevKitV4 ESP-WROOM-32E

DevKitV4 ESP-WROOM-32E to zaawansowany moduł mikroprocesorowy z serii ESP32. Łączy w sobie wysoką wydajność, niskie zużycie energii i wszechstronność. Wyposażony jest w podwójny rdzeń, obsługujący taktowanie do 240 MHz, oraz wbudowane Wi-Fi i Bluetooth. Posiada szeroki zakres interfejsów, zapewniając elastyczność w przypadku podłączania z modułami peryferyjnymi.

5.3 LCD Waveshare 1.8inch 128x160

Waveshare 1.8inch 128x160 to kolorowy wyświetlacz TFT o rozdzielczości 128x160 pikseli. Oferuje jasny i wyraźny obraz, umożliwiający wyświetlanie tekstu, grafiki i prostych animacji. Dane przesyłane są poprzez interfejs SPI.

5.4 Mangetometr HMC5883L

HMC5883L to trójosiowy magnetometr cyfrowy, który może mierzyć pole magnetyczne ziemi, co pozwala na określenie orientacji przestrzennej i wykorzystanie w roli kompasu. Dane odczytywane są poprzez interfejs I2C.

5.5 GPS GY-NEO6MV2

GY-NEO6MV2 to moduł GPS oparty na układzie NEO-6M, który umożliwia precyzyjne śledzenie pozycji geograficznej poprzez komunikację z satelitą za pomocą podłączonej anteny. Dane przezkazywane są poprzez standardowy interfejs UART.

5.6 Napotkane problemy

Podczas programowania oraz testowania komunikacji mikroprocesora z urządzeniami peryferyjnymi napotkano problemy z obsługą modułu GPS oraz magnetometru.

W przypadku połączenia z modułem GPS, wykryto wadliwość 2 z 4 zamówionych urządzeń - po wyznaczeniu oraz podłączeniu działających komponentów, nie wystąpiły już dalsze problemy.

W przypadku obsługi magnetometru pojawiały się problemy wydające się losowe w chwili próby nawiązania połączenia master-slave z modułem. Problemy te niekiedy zakłócały działanie całego programu załadowanego do mikroprocesora, niekiedy nie występowały, w innych przypadkach zaś powodowały błąd watchdoga w linii inicjalizacji sterownika I2C. Sterownik zawieszał się, oczekując transmisji na linii danych, które nigdy się nie pojawiały. Problem zidentyfikowano w błędzie implementacji sterownika I2C przez ESP-IDF. Użycie starszej wersji sterownika trwale usunęło problem.

6 Firmware

Firmware aplikacji został napisany w języku C z wykorzystaniem platformy ESP-IDF [5]. ESP-IDF to oficjalne środowisko programistyczne dla platformy ESP32, które zawiera narzędzia do budowania, debugowania i wgrywania oprogramowania na urządzenie.

6.1 Modularna struktura kodu

Framework ESP-IDF pozwala na tworzenie projektów w sposób modułowy, co pozwala na łatwe dodawanie nowych funkcjonalności i niezależne testowanie poszczególnych komponentów. W projekcie wyróżniono następujące moduły (komponenty):

- **main** - moduł główny, inicjalizuje pozostałe moduły i definiuje zmienne współdzielone.
- **sc_ble** - moduł odpowiedzialny za obsługę protokołu BLE. Zawiera definicje usług i charakterystyk GATT.
- **sc_gps** - moduł obsługujący moduł GPS. Zawiera funkcje do odczytu współrzędnych geograficznych.

- **sc_compass** - moduł obsługujący magnetometr. Zawiera funkcje do odczytu kierunku kompasu.
- **sc_display** - moduł obsługujący wyświetlacz LCD. Zawiera funkcje do wyświetlania informacji na ekranie z wykorzystaniem biblioteki LVGL.
- **sc_logic** - moduł odpowiedzialny za logikę aplikacji. Zawiera funkcje do obliczania kierunku i dystansu do celu.
- **lvgl** - moduł zawierający bibliotekę LVGL. [9] Zawiera konfigurację i funkcje pomocnicze do obsługi wyświetlacza.
- **lvgl_esp32_drivers** - moduł zawierający sterowniki dla ESP32 do biblioteki LVGL. [8]

6.2 Architektura wielowątkowa

Framework ESP-IDF jest oparty na architekturze FreeRTOS [1], co pozwala na tworzenie wielowątkowych aplikacji. Zawarta we frameworku wersja FreeRTOS pozwala na wykorzystanie dwóch rdzeni procesora ESP32, zapewniając równoległe wykonywanie zadań w dwóch wątkach. Każdy z modułów aplikacji opisanych w sekcji 6.1 działa w osobnym wątku.

6.2.1 Zmienne współdzielone

Program zawiera dwie zmienne współdzielone odpowiedzialne za przechowywanie aktualnych danych na temat odczytów z sensorów oraz przechowywanie aktualnych danych do wyświetlenia na ekranie.

Listing 1: Definicja zmiennej dla odczytów sensorów

```
typedef struct {
    SemaphoreHandle_t mutex;
    compass_position_t position;
    float bearing; // in radians
    float bearing_deg; // in degrees
    compass_path_t path;
    bool position_updated;
} compass_data_t;
```

Listing 2: Definicja zmiennej dla danych wynikowych

```
typedef struct {
    SemaphoreHandle_t mutex;
    // Angle in 0.1 degrees - between 0 and 3600
    int16_t angle;
    // Next waypoint id
    uint16_t next_wp;
    // Distance to next waypoint in meters
    uint32_t distance;
    bool finished;
} display_data_t;
```

Każda zmienna posiada semafor, który zapewnia dostęp do zasobu w sposób atomowy, co zapobiega konfliktom między wątkami.

6.3 Protokół komunikacji Bluetooth Low Energy (BLE)

Protokół komunikacji BLE został zaimplementowany zgodnie ze standardem Bluetooth Core 4.2 [3] na platformie ESP32. Użyto stosu i ESP-NimBLE [4], który zapewni odpowiednie zarządzanie pamięcią oraz obsługę komunikacji BLE. Stos ESP-NimBLE jest alternatywną wersją stosu Apache MyNewt Nimble [2], przystosowaną do pracy z mikrokontrolerami producenta Espressif.

6.3.1 Obsługa po stronie urządzenia

Urządzenie wykorzystuje stos ESP-NimBLE do implementacji protokołu BLE, co pozwala na zmniejszenie zużycia pamięci. Komunikacja z aplikacją mobilną opiera się na deklaracji usług i charakterystyk GATT, umożliwiając przesyłanie danych trasy w postaci tablic par współrzędnych geograficznych.

Listing 3: Definicja charakterystyki GATT

```
static uint8_t gatt_svr_chr_val[MAX_CHAR_LEN];
static uint16_t gatt_svr_chr_val_handle;
static const ble_uuid128_t gatt_svr_chr_uuid =
        BLE_UUID128_INIT(0x00, 0x00, 0x00, 0x00, 0x11, 0x11, 0x11, 0x11,
                         0x22, 0x22, 0x22, 0x22, 0x33, 0x33, 0x33, 0x33);
```

6.4 Napotkane problemy

W trakcie rozwoju oprogramowania pojawiły się dwa główne problemy opisane w poniższych podrozdziałach.

6.4.1 Konflikty między bibliotekami

W trakcie rozwoju modułu odpowiedzialnego za komunikację z wyświetlaczem LCD pojawił się konflikt między:

- Frameworkiem ESP-IDF
- Biblioteką LVGL,[9] która jest prostym silnikiem GUI
- Biblioteką LVGL ESP32 Drivers, [8] która zawiera sterowniki dla ESP32 do biblioteki LVGL

Początkowe próby integracji wszystkich trzech bibliotek w najnowszej wersji nie powiodły się, co skutkowało koniecznością doboru kompatybilnych wersji bibliotek oraz dostosowaniem konfiguracji. Proces naprawy konfliktów trwał kilka dni.

6.4.2 Błąd w implementacji protokołu I2C w bibliotece ESP-IDF

Do komunikacji z magnetometrem QMC5883 wykorzystano interfejs I2C *(Inter Integrated Circut)*, który jest standardowym interfejsem komunikacyjnym dla urządzeń peryferyjnych. Producent mikrokontrolera ESP32, firma Espressif, dostarcza bibliotekę do obsługi interfejsu I2C w ramach frameworku ESP-IDF.

W trakcie implementacji okazało się, że biblioteka ESP-IDF zawiera błąd w implementacji protokołu I2C, który uniemożliwiał poprawną komunikację z magnetometrem. Dodatkowo, błąd ten był niewykrywalny w procesie debugowania, co nie pozwalało na jednoznaczne zlokalizowanie problemu (mógł wynikać z błędu w kodzie, konfiguracji, podłęczeniu lub z uszkodzenia magnetometru).

Błąd naprawiono poprzez zastosowanie alternatywnej biblioteki do obsługi interfejsu I2C, co pozwoliło na poprawną komunikację z magnetometrem.

7 Aplikacja mobilna

Aplikacja mobilna spełnia następujące funkcjonalności:

- rysowanie tras
- zapisywanie tras
- zarządzanie trasami
- nawiązywanie połączenia z urządzeniem SmartCompass
- przesyłanie tras do urządzenia SmartCompass

7.1 Wykorzystane technologie

Aplikacja została zrealizowana przy użyciu frameworku **React Native** [10] pozwalającego tworzyć jeden projekt, który następnie kompilowany jest do kodu aplikacji wykonywalnej na urządzeniach z Androidem jak i iOS. Ze względu na wymagania powiązane z uzyskaniem dostępu do modułu Bluetooth Low Energy urządzenia, zadecydowaliśmy się rozwijać naszą aplikację jedynie w kierunku systemu Android. Ponadto wykorzystano platformę i zestaw narzędzi **Expo** [6], które upraszczają proces tworzenia, testowania i wdrażania aplikacji mobilnych.

7.2 Wykorzystane biblioteki

W aplikacji wykorzystano szereg bibliotek, które ułatwiły realizację poszczególnych funkcjonalności:

- **react-native-async-storage** - Służy do przechowywania danych lokalnych na urządzeniu. Umożliwia łatwe zapisywanie i odczytywanie danych, takich jak ustawienia użytkownika czy trasy.
- **react-navigation** - Biblioteka do nawigacji w aplikacjach React Native. Umożliwia łatwe zarządzanie stosami ekranów, nawigację pomiędzy nimi oraz implementację różnorodnych wzorców nawigacji.
- **react-native-ble-plx** - Biblioteka do obsługi Bluetooth Low Energy (BLE). Umożliwia skanowanie, łączenie i komunikację z urządzeniami BLE, co jest kluczowe dla integracji z urządzeniem SmartCompass.
- **react-native-quick-base64** - Umożliwia szybkie kodowanie i dekodowanie danych w formacie base64, co jest przydatne przy przesyłaniu danych przez BLE.
- **react-native-vector-icons** - Biblioteka dostarczająca zestaw ikon do aplikacji. Umożliwia łatwe dodawanie ikon do przycisków, menu i innych elementów interfejsu użytkownika.
- **react-native-maps** - Biblioteka do integracji map w aplikacji. Umożliwia wyświetlanie map, rysowanie tras oraz interakcję z punktami na mapie.
- **react-native-reanimated** - Biblioteka do tworzenia płynnych i wydajnych animacji w aplikacji. Umożliwia tworzenie zaawansowanych animacji i interakcji, co poprawia ogólne wrażenia użytkownika.

7.3 Tworzenie wersji *release* aplikacji

Pierwszym krokiem jest dodanie informacji takich jak nazwa, ikona, splash screen oraz konfiguracje specyficzne dla platformy Android do pliku `app.json`.

Listing 4: Plik app.json

```
"expo": {
    "name": "app",
    "slug": "app",
    "version": "1.0.0",
    "orientation": "portrait",
    "icon": "./assets/icon.png",
    "plugins": [
      "react-native-ble-plx"
    ],
    "userInterfaceStyle": "light",
    "splash": {
      "image": "./assets/splash.png",
      "resizeMode": "contain",
      "backgroundColor": "#ffffff"
    },
}
```

należy również utworzyć konfigurację dla usługi **Expo Application Services (EAS)**, który jest częścią procesu budowania aplikacji. Konfigurację tą umieszcza się w pliku `eas.json`.

Listing 5: Plik eas.json

```
{
  "cli": {
    "version": ">= 8.0.0"
  },
  "build": {
    "development": {
      "developmentClient": true,
```

```
      "distribution": "internal"
    },
    "preview": {
      "android": {
        "buildType": "apk"
      }
    },
    "production": {}
  },
  "submit": {
    "production": {}
  }
};
```

Dla klucza `preview` określamy typ budowania aplikacji jako `"apk"` dla platformy Android. Następnie należy zalogować się do konta *Expo*, a później, korzystając z komendy `'npx build -p android {profile preview'`, wygenerować plik `.apk`, który dostępny będzie do pobrania z konta użytkownika w platformie *Expo*. Proces generowania pliku `.apk` jest zautomatyzowany i przeprowadzany przez narzędzie **Expo Build Service**. W przypadku naszej aplikacji rozwiązanie to znacząco ułatwia cały proces względem *Bare Workflow*, czyli podejścia w którym nie wykorzystuje się frameworka *Expo*.

7.3.1 Różnice pomiędzy wersją testową, a *release*

Wersje testowe i release różnią się znacząco pod względem celu, konfiguracji, procesu budowania oraz dystrybucji. Wersje testowe są przeznaczone do szybkich iteracji i debugowania, zawierając narzędzia deweloperskie i umożliwiając natychmiastowe aktualizacje. Wersje release są zoptymalizowane pod kątem wydajności, stabilności i bezpieczeństwa, przeznaczone do dystrybucji do szerokiego grona użytkowników końcowych poprzez oficjalne sklepy aplikacji.

7.4 Napotkane problemy

Obsługa połączenia BLE z urządzeniem okazała się być trudniejsza niż zakładaliśmy, w szczególności monitorowanie sesji z poziomu aplikacji mobilnej. Cały proces wymaga również uzyskania odpowiednich pozwoleń od urządzenia na którym działa aplikacja, a proces testowania był utrudniony ponieważ posiadamy tylko jeden egzemplarz urządzenia SmartCompass. Ze względów technicznych UUID serwisu BLE naszego urządzenia SmartCompass został na sztywno ustawiony w aplikacji.

Część III

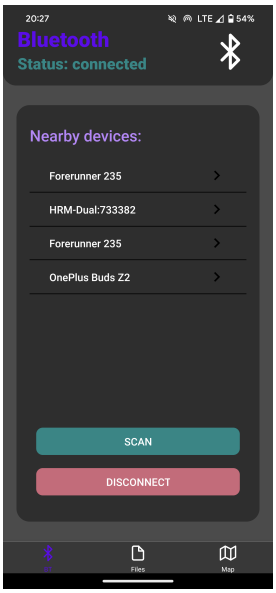
Efekty

8 Prototyp urządzenia

9 Aplikacja mobilna

9.1 Widoki aplikacji

9.1.1 BLE connection

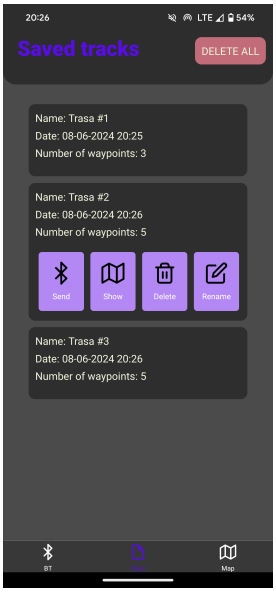


Rysunek 2: Ekran BLE connection

Widok ten pozwala użytkownikowi rozpocząć skanowanie pobliskich urządzeń BLE, a następnie wybrać jedno z nich i połączyć się z nim. Jeśli jest to urządzenie SmartCompass, to znaleziony zostanie udostępniany przez nie serwis BLE, którego UUID zapisane jest w aplikacji. Nawiązanie z nim połączenia umożliwi przesyłanie do tego urządzenia danych. Na ekranie tym znajdują się również dwa przyciski:

- **Przycisk Scan** - rozpoczyna skanowanie urządzeń
- **Przycisk Disconnect** - rozłącza z urządzeniem.

9.1.2 Files



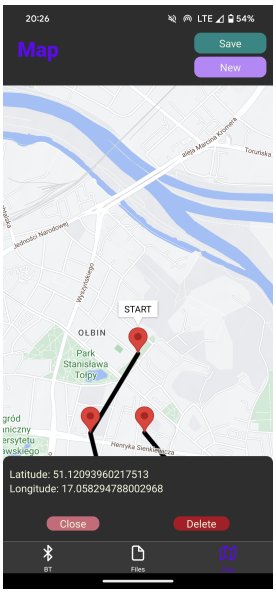
Rysunek 3: Ekran plików

Widok ten oferuje możliwość przeglądania zapisanych w urządzeniu tras

Po kliknięciu w trasę, rozwinie się menu z 4 przyciskami:

- **delete** – usuwa wybraną trasę.
- **edit** – przenosi do widoku mapy i wyświetla w nim wybraną trasę pozwalając na jej edycję.
- **send** – pozwala przesłać wybraną trasę do urządzenia SmartCompass jeśli jest ono połączone.

9.1.3 Map



Rysunek 4: Ekran Mapy

Widok mapy oferuje możliwość rysowania trasy, dodawania nowej trasy oraz zapisu obecnie narysowanej trasy. W tym celu wykorzystywane są poniższe przyciski:

- **NEW** – wyczyszczenie punktów na mapie, jeśli trasa jest zapisana nie zostanie ona utracona. Pozwala na rysowanie nowej trasy
- **SAVE** – służy do zapisu aktualnie narysowanej trasy. Po kliknięciu pojawi się okienko z miejscem na wpisanie nazwy trasy. Trasa zapisana zostanie do wewnętrznej, statycznej pamięci urządzenia i widoczna będzie w widoku **Files** po restarcie aplikacji.

Ponadto do interakcji z mapą użytkownik wykorzystuje następujące gesty:

- **przytrzymanie palcem w pustym miejscu na mapie** – dodanie nowego punktu
- **dłuższe przytrzymanie znacznika punktu** – przejście w tryb przenoszenia punktu poprzez przesuwanie palcem po ekranie. Znacznik zostanie upuszczony po oderwaniu palca od ekranu
- **kliknięcie w znacznik** – nad znacznikiem wyświetli się małe pole z jego numerem na trasie lub napisem *START/META*. Ponadto na dole ekranu wyświetli się okno z dokładną lokalizacją punktu. Kliknięcie w puste miejsce na mapie spowoduje zamknięcie tego okna i zniknięcie pola nad znacznikiem.

10 Możliwości rozwoju

10.1 Rozwój aplikacji mobilnej

Aplikacja realizuje zdefiniowane dla niej założenia. Potencjał na rozwój znajduje się w interfejsie użytkownika i, pomimo że jest kompletny, można wprowadzić zmiany, które poprawią ogólne wrażenia oraz intuicyjność korzystania z aplikacji dla użytkownika końcowego.

Literatura

[1] Amazon Web Services, Inc. Freertos documentation. https://freertos.org/index.html.

[2] Apache Software Foundation. Apache mynewt documentation. https://mynewt.apache.org/.

[3] Bluetooth SIG, Inc. Bluetooth core specification 4.2. https://www.bluetooth.com/specifications/specs/core-specification-4-2/.

[4] Espressif Systems. Esp-idf bluetooth nimble api reference. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/bluetooth/nimble/index.html.

[5] Espressif Systems. Esp-idf getting started guide v5.2.1 (stable). https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html.

[6] Expo Team. Expo documentation. https://expo.dev/.

[7] GitHub, Inc. About projects. https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects.

[8] LVGL Project. Lvgl esp32 drivers. https://github.com/lvgl/lvgl_esp32_drivers.

[9] LVGL Project. Lvgl esp32 port. https://github.com/lvgl/lv_port_esp32.

[10] Meta Platforms, Inc. React native documentation. https://reactnative.dev/.