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

I Zarządzanie projektem **2**

1 Założenia projektowe **2**
1.1 Wymagania projektowe . . . 2

2 Ryzyka projektowe **2**
2.1 Odejście członków zespołu . . . 2
2.2 Ograniczony budżet . . . 3
2.3 Opóźnienia w dostawie komponentów . . . 3
2.4 Opóźnienia w montażu układu (PCB) . . . 3
2.5 Niewystarczające umiejętności . . . 3
2.6 Skomplikowana dokumentacja techniczna komponentów . . . 3

3 Harmonogram projektu **3**

4 Zarządzanie zespołem **4**
4.1 Podział zadań . . . 4
4.2 Planowanie pracy . . . 4

II Implementacja **5**

5 Projekt urządzenia **5**
5.1 Wybrane moduły elektroniczne . . . 5
5.2 Kosztorys prototypu . . . 5
5.3 Napotkane problemy . . . 5

6 Firmware **5**
6.1 Modularna struktura kodu . . . 5
6.2 Architektura wielowątkowa . . . 6
6.2.1 Zmienne współdzielone . . . 6
6.3 Protokół komunikacji Bluetooth Low Energy (BLE) . . . 6
6.3.1 Obsługa po stronie urządzenia . . . 6
6.4 Napotkane problemy . . . 7
6.4.1 Konflikty między bibliotekami . . . 7
6.4.2 Błąd w implementacji protokołu I2C w bibliotece ESP-IDF . . . 7

7 Aplikacja mobilna **7**
7.1 Napotkane problemy . . . 7

III Efekty **8**

8 Prototyp urządzenia **8**

9 Aplikacja mobilna **8**

10 Możliwości rozwoju **8**

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 [6], 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

5.1 Wybrane moduły elektroniczne

Do budowy prototypu wybrano następujące moduły elektroniczne:

- **MCU - Espressif DevKitV4 ESP-WROOM-32E**
- **Magnetometr cyfrowy GY-273 3-osiowy I2C 3.3V / 5V - HMC5883L**
- **Moduł GPS GY-NEO6MV2 NEO-6M z Anteną**
- **Wyświetlacz LCD Waveshare 1.8inch 128x160**

5.2 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

5.3 Napotkane problemy

Tu można opisać problemy napotkane podczas projektowania urządzenia.

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. [8] Zawiera konfigurację i funkcje pomocnicze do obsługi wyświetlacza.
- **lvgl_esp32_drivers** - moduł zawierający sterowniki dla ESP32 do biblioteki LVGL. [7]

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,[8] która jest prostym silnikiem GUI
- Biblioteką LVGL ESP32 Drivers, [7] 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, blad 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

7.1 Napotkane problemy

Tu można opisać problemy napotkane podczas pisania oprogramowania.

Część III

Efekty

8 Prototyp urządzenia

9 Aplikacja mobilna

10 Możliwości rozwoju

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

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

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