



**A G H**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**Wydział Inżynierii Mechanicznej i Robotyki**

**KATEDRA ROBOTYKI I MECHATRONIKI**

## Projekt dyplomowy

*Projekt i wykonanie systemu lokalizacji pojazdu  
Design and Implementation of a Vehicle Localization  
System*

Autor:

*Mirosz Stec*

Kierunek studiów:

Inżynieria Mechatroniczna

Opiekun pracy:

*dr inż. Grzegorz Góra*

Kraków, 2025/2026

## Spis treści

1.	Wstęp .....	3
2.	Cel i zakres pracy .....	4
3.	Przegląd technologii i porównanie konkurencyjnych rozwiązań .....	5
3.1.	Przegląd technologii wykorzystywanych w urządzeniu .....	5
3.1.1.	Mikrokontroler ESP32 i architektura SoC .....	5
3.1.2.	System Operacyjny Czasu Rzeczywistego (FreeRTOS) .....	6
3.1.3.	Technologia GNSS i protokół NMEA 0183 .....	6
3.1.4.	Protokół MQTT (Message Queuing Telemetry Transport) .....	7
3.1.5.	Serializacja danych JSON .....	7
3.1.6.	Magistrala 1-Wire .....	7
3.1.7.	Platforma ThingsBoard.....	8
3.1.8.	Biblioteki programistyczne .....	8
3.2.	Analiza wybranych rozwiązań rynkowych .....	9
3.2.1.	Lokalizator GPS BearPoint C18 .....	9
3.2.2.	Lokalizator GPS OBD MK08 .....	10
3.2.3.	Lokalizator GPS MK6B 4G (Magnetyczny).....	10
3.3.	Kategoryzacja dostępnych rozwiązań .....	11
3.4.	Analiza porównawcza.....	12
3.5.	Wnioski z przeglądu i uzasadnienie wyboru technologii.....	13
4.	Projekt .....	14
4.1.	Założenia systemowe i architektura sprzętowa.....	14
4.2.	Architektura oprogramowania .....	14
4.3.	Bezpieczeństwo i testy .....	15
4.4.	Wykonanie.....	15
5.	Podsumowanie i wnioski .....	18
6.	Wykaz Publikacji .....	19
7.	Załączniki .....	20

## **1. Wstęp**

Dynamiczny rozwój technologii Internetu Rzeczy (IoT) oraz miniaturyzacja układów elektronicznych umożliwiają tworzenie coraz bardziej zaawansowanych, a jednocześnie kompaktowych systemów monitorujących i sterujących. Jednym z praktycznych zastosowań tej technologii jest lokalizacja i nadzorowanie pojazdów, co znajduje szerokie zastosowanie zarówno w logistyce, jak i w codziennym użytkowaniu samochodów prywatnych.

Współczesne systemy lokalizacyjne pozwalają nie tylko na określanie pozycji pojazdu, ale także na zbieranie i analizę danych środowiskowych oraz diagnostycznych, co znacząco zwiększa ich funkcjonalność i wartość użytkową.

Niniejsza praca inżynierska dotyczy projektu i wykonania systemu lokalizacji pojazdu opartego na mikrokontrolerze ESP32 oraz module GNSS. Opracowany układ ma za zadanie rejestrować dane o położeniu pojazdu i zapisywać je lokalnie na karcie SD, a po uzyskaniu dostępu do znanej sieci bezprzewodowej przesyłać je na serwer z wykorzystaniem komunikacji Wi-Fi. Dzięki temu możliwe będzie późniejsze analizowanie lub wizualizowanie zgromadzonych informacji. Projekt zakłada również opcjonalną integrację z interfejsem OBD2, co pozwoli na rejestrację podstawowych danych diagnostycznych pojazdu i ich synchronizację z danymi lokalizacyjnymi. Ponadto system może zostać rozbudowany o czujniki środowiskowe, takie jak pomiar temperatury czy wilgotności.

W ramach pracy wykonano prototyp urządzenia oraz zaprojektowano obudowę umożliwiającą testy w warunkach zbliżonych do rzeczywistych. Motywacją do podjęcia tematu była chęć praktycznego zastosowania wiedzy z zakresu elektroniki i programowania mikrokontrolerów, a także zainteresowanie technologiami IoT. Dodatkowym impulsem była możliwość wykorzystania rozwiązania w środowisku rzeczywistym, związanym z eksploatacją pojazdów w ramach rodzinnej wypożyczalni. Realizacja projektu pozwoliła na zdobycie doświadczenia w projektowaniu układów elektronicznych, integracji sprzętu i oprogramowania oraz w przetwarzaniu danych w systemach rozproszonych.

## **2. Cel i zakres pracy**

Celem pracy jest zaprojektowanie, wykonanie, uruchomienie oraz przetestowanie prototypu urządzenia umożliwiającego lokalizację pojazdu przy wykorzystaniu modułu GNSS (np. GPS).

### Cel główny

- Zaprojektować i zrealizować prototyp systemu lokalizacji pojazdu opartego na ESP32 i module GNSS.

### Cele szczegółowe

- Zaimplementować oprogramowanie mikrokontrolera do odczytu GNSS, zapisu na SD oraz synchronizacji danych z serwerem.
- Wykonać prototyp płytki (lub układ testowy) i obudowy.
- Przeprowadzić testy.
- Ocenić jakość danych GNSS i skuteczność synchronizacji.

### Zakres pracy

- Analiza wymagań i przegląd dostępnych rozwiązań.
- Stworzenie prototypu urządzenia
- Projekt schematu elektronicznego i PCB lub układu testowego.
- Implementacja firmware w Arduino IDE (ESP32).
- Przygotowanie wniosków i propozycji dalszych prac.

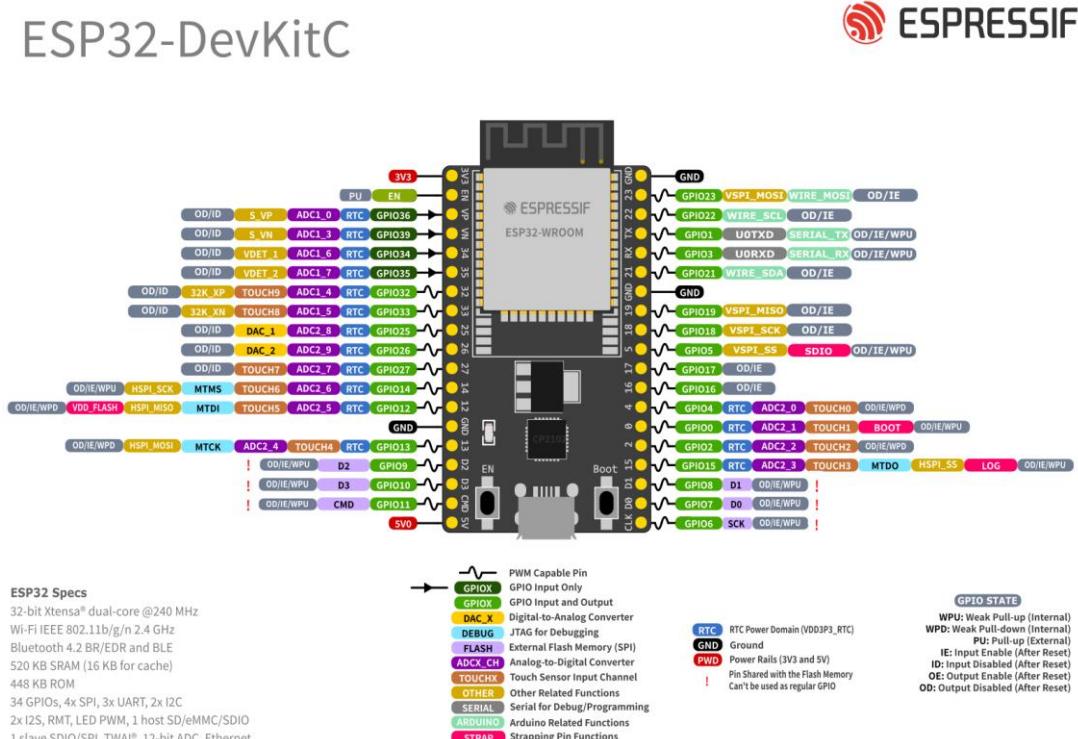
### **3. Przegląd technologii i porównanie konkurencyjnych rozwiązań**

### **3.1. Przegląd technologii wykorzystywanych w urządzeniu**

Realizacja projektu systemu lokalizacji pojazdu wymagała integracji szeregu technologii sprzętowych oraz programowych. W niniejszym rozdziale omówiono kluczowe standardy, protokoły oraz środowiska uruchomieniowe, które stanowiły fundament dla opracowanego rozwiązania.

### 3.1.1. Mikrokontroler ESP32 i architektura SoC

Sercem zaprojektowanego urządzenia jest układ ESP32 firmy Espressif Systems. Jest to tani, energooszczędny układ typu SoC (System on Chip) z zintegrowaną obsługą sieci Wi-Fi oraz Bluetooth. Wybór tej platformy podyktowany był jej wysoką wydajnością (dwurdzeniowy procesor Xtensa® 32-bit LX6), dużą ilością pamięci SRAM (520kB) oraz bogatym zestawem interfejsów periferyjnych (UART, SPI, I2C), co jest kluczowe dla jednoczesnej obsługi modułu GNSS, karty SD oraz transmisji sieciowej.



### **3.1.2. System Operacyjny Czasu Rzeczywistego (FreeRTOS)**

W projekcie, zamiast klasycznej pętli głównej (super-loop), zastosowano system operacyjny czasu rzeczywistego FreeRTOS. Jest to otwarty system operacyjny przeznaczony dla systemów wbudowanych, który umożliwia wielozadaniowość z wywłaszczeniem.

Wykorzystanie FreeRTOS pozwoliło na podział oprogramowania na niezależne zadania (Tasks), takie jak:

- Obsługa modułu GPS,
- Odczyt temperatury,
- Zarządzanie połączeniem Wi-Fi,
- Logowanie danych na kartę SD.

Dzięki mechanizmom synchronizacji, takim jak semafory (Semaphores) i mutexy (Mutexes), możliwe było bezpieczne współdzielenie zasobów sprzętowych pomiędzy wątkami bez ryzyka wyścigów danych (race conditions).

### **3.1.3. Technologia GNSS i protokół NMEA 0183**

Do określania pozycji geograficznej wykorzystano technologię GNSS (Global Navigation Satellite System). Moduł odbiornika komunikuje się z mikrokontrolerem za pomocą interfejsu UART, przesyłając dane w formacie tekstowym NMEA 0183.

Standard NMEA definiuje "zdania" (sentences) zawierające informacje o pozycji, czasie i statusie satelitów. W projekcie kluczowe są ramki:

- **\$GPRMC** (Recommended Minimum Specific GNSS Data) – zawierająca czas, datę, szerokość i długość geograficzną oraz prędkość.
- **\$GPGGA** (Global Positioning System Fix Data) – dostarczająca informacji o wysokości n.p.m. oraz jakości fixa (liczba satelitów, HDOP).

### **3.1.4. Protokół MQTT (Message Queuing Telemetry Transport)**

Jako warstwę transportową do przesyłania danych telemetrycznych do chmury wybrano protokół MQTT. Jest to lekki protokół typu Publish/Subscribe, działający na szczeblu stosu TCP/IP.

Jego główne zalety w kontekście lokalizatora pojazdu to:

- **Niski narzut danych:** Nagłówek pakietu jest minimalny, co oszczędza transfer.
- **Odporność na niestabilne łącza:** Mechanizmy QoS (Quality of Service) i Keep-Alive pozwalają na utrzymanie sesji w trudnych warunkach sieciowych.
- **Model Pub/Sub:** Urządzenie (Publisher) wysyła dane do Brokera (serwer ThingsBoard), nie musząc wiedzieć, kto te dane odbiera.

### **3.1.5. Serializacja danych JSON**

Do formatowania danych przesyłanych do serwera oraz zapisywanych na karcie SD wykorzystano format JSON (JavaScript Object Notation). Mimo że jest on bardziej obszerny niż formaty binarne, zapewnia czytelność dla człowieka i łatwą integrację z nowoczesnymi platformami webowymi oraz IoT (ThingsBoard natywnie obsługuje JSON). W projekcie zastosowano wariant JSON Lines (.jsonl) dla logów na karcie SD, co pozwala na dopisywanie rekordów linia po linii bez konieczności wczytywania całego pliku do pamięci RAM.

### **3.1.6. Magistrala 1-Wire**

Do komunikacji z cyfrowym czujnikiem temperatury DS18B20 wykorzystano magistralę 1-Wire. Jest to szeregowy interfejs komunikacyjny wymagający tylko jednego przewodu danych (oraz masy). Pozwala to na łatwą rozbudowę systemu o kolejne czujniki bez zajmowania dodatkowych pinów mikrokontrolera, gdyż każde urządzenie 1-Wire posiada unikalny 64-bitowy adres fabryczny.

### **3.1.7. Platforma ThingsBoard**

Jako system backendowy do wizualizacji i gromadzenia danych wybrano platformę IoT ThingsBoard. Umożliwia ona tworzenie interaktywnych dashboardów, zarządzanie urządzeniami oraz analizę danych telemetrycznych w czasie rzeczywistym. Dzięki obsłudze protokołu MQTT, integracja z układem ESP32 jest natywna i nie wymaga pośredniczących bramek.

### **3.1.8. Biblioteki programistyczne**

Oprogramowanie urządzenia (Firmware) zostało napisane w języku C++ z wykorzystaniem frameworka Arduino dla ESP32. Kluczowe wykorzystane biblioteki to:

- **TinyGPSPlus:** Zaawansowany parser ramek NMEA, obsługujący wyodrębnianie współrzędnych i czasu.
- **ArduinoJson:** Biblioteka do wydajnej serializacji i deserializacji obiektów JSON w systemach wbudowanych.
- **PubSubClient:** Klient protokołu MQTT.
- **SD / FS:** Biblioteki do obsługi systemu plików FAT32 na kartach pamięci.

## **3.2. Analiza wybranych rozwiązań rynkowych**

Rynek systemów lokalizacji pojazdów (AVL – Automatic Vehicle Location) jest obecnie nasycony rozwiązaniami o zróżnicowanej funkcjonalności, cenie oraz przeznaczeniu. Od prostych rejestratorów tras, przez systemy antykradzieżowe, aż po zaawansowaną telematykę flotową. W celu osadzenia realizowanego projektu w kontekście rynkowym, dokonano analizy trzech reprezentatywnych urządzeń dostępnych w sprzedaży oraz porównano je z projektowanym rozwiązaniem opartym na mikrokontrolerze ESP32.

Do przeglądu wybrano urządzenia reprezentujące różne segmenty rynku:

- klasyczny lokalizator montowany na stałe,
- rozwiązanie typu Plug&Play pod złącze OBD;
- przenośny lokalizator z własnym zasilaniem.

### **3.2.1. Lokalizator GPS BearPoint C18**

<https://bearpoint.pl/produkt/lokalizator-gps-bearpoint/> (odwiedzona 03.12.2025)

BearPoint C18 to przykład klasycznego lokalizatora montowanego na stałe w instalacji elektrycznej pojazdu. Urządzenie dedykowane jest zarówno klientom indywidualnym, jak i flotowym.

- **Charakterystyka:** Urządzenie korzysta z technologii 4G/LTE (z fallbackiem do 2G), co zapewnia szerokie pokrycie zasięgiem. Posiada wbudowaną baterię podtrzymującą pracę po odłączeniu zasilania głównego.
- **Funkcjonalność:** Oferuje śledzenie w czasie rzeczywistym, funkcję zdalnego odcięcia zapłonu (przez przekaźnik), alarmy o przekroczeniu prędkości oraz geostrefy.
- **Model biznesowy:** Producent oferuje dostęp do dedykowanej platformy śledzącej (często w modelu abonentowym lub z opłatą za serwer), co generuje stałe koszty eksploatacji, ale zapewnia gotowe środowisko wizualizacyjne.

### **3.2.2. Lokalizator GPS OBD MK08**

<https://szpiegujemy.com/product-pol-16904> (odwiedzona 03.12.2025)

MK08 reprezentuje kategorię urządzeń "Plug & Play", wykorzystujących złącze diagnostyczne OBDII jako źródło zasilania oraz punkt montażowy.

- **Charakterystyka:** Największą zaletą tego rozwiązania jest brak konieczności ingerencji w instalację elektryczną pojazdu – montaż sprowadza się do wpięcia urządzenia w gniazdo diagnostyczne.
- **Funkcjonalność:** Urządzenie zapewnia podstawową lokalizację, historię tras oraz alarmy wstrząsowe. Mimo podłączenia do gniazda OBD, tanie wersje tego typu urządzeń często wykorzystują je wyłącznie do poboru energii, nie oferując pełnej telemetrii parametrów silnika (jak RPM czy kody błędów), co odróżnia je od zaawansowanych systemów telematycznych.
- **Zasilanie:** Ciągłe zasilanie z akumulatora pojazdu eliminuje problem ładowania, ale w przypadku długiego postoju może obciążać akumulator rozruchowy.

### **3.2.3. Lokalizator GPS MK6B 4G (Magnetyczny)**

<https://szpiegujemy.com/product-pol-16963> (odwiedzona 03.12.2025)

Model MK6B to przedstawiciel lokalizatorów mobilnych, niewymagających stałego podłączenia do instalacji pojazdu.

- **Charakterystyka:** Urządzenie wyposażone jest w bardzo pojemny akumulator (np. 3000 mAh lub większy), co pozwala na pracę od kilku tygodni do kilku miesięcy bez ładowania (w zależności od częstotliwości raportowania). Silny magnes umożliwia szybki montaż na zewnątrz pojazdu lub w kontenerach.
- **Komunikacja:** Wykorzystuje modem 4G LTE do przesyłu danych w czasie rzeczywistym.
- **Zastosowanie:** Idealne do doraźnego śledzenia, zabezpieczania ładunków lub maszyn budowlanych, gdzie nie ma dostępu do stałego zasilania.

### **3.3. Kategoryzacja dostępnych rozwiązań**

Na podstawie przeglądu rynku można wyróżnić trzy główne kategorie systemów lokalizacji, różniące się architekturą i grupą docelową:

1. **Gotowe komercyjne trackery GPS (np. BearPoint, MK08, MK6B):** Są to kompletne produkty, często zamknięte (proprietary), zintegrowane z kartą SIM i dedykowaną chmurą producenta. Zapewniają wysoką niezawodność i łatwość obsługi, ale wiążą się z kosztami abonentowymi i brakiem możliwości modyfikacji oprogramowania czy sposobu przetwarzania danych.
2. **Rozwiązania OEM (Original Equipment Manufacturer):** Systemy fabrycznie montowane w nowoczesnych pojazdach, głęboko zintegrowane z magistralą CAN i systemami infotainment. Oferują najwyższą jakość danych diagnostycznych, ale są zamknięte dla użytkownika i trudne do przeniesienia do starszych pojazdów.
3. **Modułowe rozwiązania DIY (Projekt inżynierski):** Systemy oparte na uniwersalnych mikrokontrolerach (ESP32, Arduino) i oddzielnych modułach GNSS/GSM. Cechują się wysoką elastycznością, niskim kosztem początkowym i pełną kontrolą nad danymi, wymagają jednak samodzielnej integracji sprzętowej i programowej.

### **3.4. Analiza porównawcza**

Poniższa tabela przedstawia porównanie analizowanych rozwiązań komercyjnych z prototypem realizowanym w ramach niniejszej pracy inżynierskiej.

Kryterium	Komercyjne	OEM	DIY
Dokładność lokalizacji	Wysoka	Bardzo wysoka	Zależna od modułu
Ciągłość przesyłu	Wysoka / Ciągła	Wysoka / Ciągła	Zależna
Zasilanie	Złącze pojazdu (OBD) + bateria	Instalacja pojazdu	Bateryjne (Li-Ion 18650) + możliwość OBD
Koszt	Średni + Abonament	Wysoki	Niski
Otwartość systemu	Zamknięty	Całkowicie zamknięty	Otwarty

### **3.5. Wnioski z przeglądu i uzasadnienie wyboru technologii**

Analiza konkurencyjnych rozwiązań prowadzi do następujących wniosków, które ukształtowały założenia projektowe niniejszej pracy:

1. **Zasadność podejścia modułowego:** Dla celów edukacyjnych i prototypowych, wykorzystanie platformy ESP32 jest optymalne. Pozwala na drastyczne obniżenie kosztów budowy urządzenia oraz umożliwia szybkie iteracje (zmiany w kodzie i sprzęcie), co jest niemożliwe w zamkniętych produktach komercyjnych.
2. **Akceptacja ograniczeń komunikacyjnych:** Wybór Wi-Fi jako głównego kanału transmisji jest świadomym kompromisem. Choć komercyjne trackery (MK6B, BearPoint) oferują ciągły monitoring przez LTE, w zastosowaniach takich jak analiza tras po fakcie, logistyka wewnętrzna czy "czarne skrzynki", synchronizacja okresowa (Batch Upload) po powrocie do zasięgu znanej sieci jest wystarczająca i tańsza w eksploatacji (brak karty SIM).
3. **Potencjał integracji OBD2:** Przegląd urządzenia MK08 wskazuje na duży potencjał złącza diagnostycznego. Jednakże, aby w projekcie DIY uzyskać funkcjonalność wykraczającą poza samo zasilanie (tj. odczyt parametrów silnika), konieczne byłoby zastosowanie dodatkowego transceivera CAN (np. TJA1050) oraz implementacja stosu protokołów ISO 15765-4 / SAE J1979, co stanowi potencjalny kierunek rozwoju prototypu w przyszłości.

Podsumowując, realizowany projekt nie stanowi bezpośredniej konkurencji dla systemów antykradzieżowych (wymagających LTE), lecz jest elastyczną platformą telemetryczną (Data Logger), oferującą funkcjonalności niedostępne w tanich trackerach, takie jak pełny dostęp do surowych danych i możliwość dowolnej konfiguracji parametrów zapisu.

## 4. Projekt

### 4.1. Założenia systemowe i architektura sprzętowa

System został zaprojektowany jako modułowy rejestrator parametrów jazdy. Główną jednostką obliczeniową jest platforma ESP32. Lokalizacja realizowana jest przez zewnętrzny moduł GNSS kompatybilny ze standardem NMEA (komunikacja UART).

- **Zasilanie:** Urządzenie zasilane jest ogniwem litowo-jonowym typu 18650. Układ zasilania wyposażono w moduł ładowania (TP4056) oraz stabilizator napięcia (LDO) dostarczający napięcia 3.3V dla mikrokontrolera.
- **Pamięć masowa:** Do lokalnego buforowania danych zastosowano kartę microSD. Zapis odbywa się w formacie tekstowym (JSON), co umożliwia łatwy odczyt danych na komputerze PC.
- **Łączność:** Podstawowym kanałem komunikacji jest Wi-Fi. Moduł Bluetooth może służyć do wstępnej konfiguracji urządzenia.
- **Rozszerzenia:** Przewidziano miejsce na transceiver CAN (np. TJA1050) do obsługi OBD2 oraz złącza dla czujników środowiskowych (I2C/1-Wire).

### 4.2. Architektura oprogramowania

Oprogramowanie zostało podzielone na warstwy logiczne:

1. **Warstwa sprzętowa (HAL):** Sterowniki obsługujące moduł GNSS, kartę SD oraz czujniki.
2. **Warstwa logiki biznesowej:** Odpowiada za agregację danych. Dane z GPS (współrzędne, prędkość, kurs) są łączone z odczytami czujników i opatrywane znacznikiem czasu (Timestamp UTC).

3. **Warstwa komunikacji i synchronizacji:** Odpowiada za wykrywanie znanej sieci Wi-Fi. Po połączeniu system przechodzi w tryb "Burst transfer", wysyłając zgromadzone na karcie SD dane do serwera (metodą POST lub przez MQTT). Po otrzymaniu potwierdzenia od serwera, lokalne kopie danych są oznaczane jako zsynchronizowane lub usuwane.
4. **Zarządzanie energią:** Zaimplementowano harmonogram pracy, w którym ESP32 przechodzi w stan *Deep Sleep* pomiędzy kolejnymi cyklami pomiarowymi, co znacząco wydłuża czas pracy na baterii.

#### **4.3. Bezpieczeństwo i testy**

Weryfikacja tożsamości urządzenia odbywa się poprzez token API zaszyty w oprogramowaniu układowym. System jest odporny na błędy sieciowe dzięki zastosowaniu kolejki zapisu oraz mechanizmu ponawiania połączeń (*retry logic*).

Przeprowadzono następujące testy:

- **Funkcjonalne:** Potwierdzono poprawność parsowania ramek NMEA i zapisu plików na karcie SD.
- **Polowe:** Wykonano jazdy próbne, porównując zarejestrowaną trasę z trasą referencyjną (Google Maps).
- **Trwałości:** Zbadano czas pracy na baterii w różnych interwałach raportowania.

#### **4.4. Wykonanie**

##### **Założenia systemowe**

- Platforma: ESP32.
- GNSS: zewnętrzny moduł kompatybilny z NMEA (np. UART).
- Pamięć lokalna: karta microSD z zapisem JSON/CSV.
- Łączność: Wi-Fi.

- Zasilanie: ogniwo 18650 z układem ładowania i stabilizacją napięcia.
- Opcjonalnie: CAN transceiver do OBD2 oraz czujniki (temp., wilgotność).

#### Architektura sprzętowa (skrót)

- Zasilanie: gniazdo baterii 18650, układ ładowania/zarządzania baterią, regulator napięcia do 5 V/3.3 V. Filtracja i zabezpieczenia.
- Moduły peryferyjne: GNSS (UART), SD (SPI), CAN transceiver (np. MCP/TJA) na interfejsie CAN, czujniki I<sup>C</sup>/SPI.
- Interfejsy programowe: biblioteki TinyGPSPlus do parsowania NMEA, SD.h do zapisu, ArduinoJson do serializacji, WiFi/HTTPClient do synchronizacji.

#### Architektura oprogramowania (skrót)

- Warstwa sprzętowa: sterowniki GNSS, SD, CAN, czujników.
- Warstwa logiki: agregacja danych, buforowanie, format zapisu (JSON z timestamp, lat, lon, alt, speed, OBD data).
- Warstwa komunikacji: procedury łączenia z Wi-Fi, retry logic, POST do endpointu REST. Alternatywnie MQTT.
- Zarządzanie energią: tryby głębokiego uśpienia ESP32 między odczytami, harmonogram odpytywania GNSS.

#### Format danych i synchronizacja

- Proponowany format: JSON per rekord z polami: timestamp (UTC), lat, lon, hdop, sats, speed, heading, source, optional obd{rpm,dtc,...}, sensors{temp,hum}.
- Zapis lokalny: plik na SD.
- Synchronizacja: przy wykryciu znanej sieci Wi-Fi system wysyła nieprzesłane rekordy do serwera. Potwierdzenie serwera usuwa lokalne kopie lub oznacza je jako zsynchronizowane.

## Bezpieczeństwo i niezawodność

- Uwierzytelnianie do serwera: token API w firmware.
- Mechanizmy odporne na błędy: kolejka zapisu, limit rozmiaru pliku, rotacja plików, retry połączeń.

## Testy i metody oceny

- Testy funkcjonalne: poprawność odczytu GNSS, zapisu na SD, przesyłu do serwera.
- Testy polowe: testy z jazdą próbna, porównanie trasy z referencją.
- Testy trwałości: czas pracy na baterii, odporność na restart.
- Metryki: czas opóźnienia przesyłu, współczynnik utraconych rekordów, średni błąd pozycji (jeśli dostępne dane referencyjne), czas pracy na baterii.

## 5. Podsumowanie i wnioski

W ramach pracy dyplomowej zaprojektowano i zrealizowano prototyp systemu lokalizacji pojazdu oparty na układzie ESP32. Prototyp spełnia założone cele funkcjonalne: poprawnie rejestruje dane GNSS, buforuje je na karcie pamięci i synchronizuje z serwerem po wykryciu sieci Wi-Fi.

### Wnioski:

1. Układ ESP32 posiada wystarczającą moc obliczeniową do obsługi zadań lokalizacyjnych i kryptograficznych.
2. Wykorzystanie Wi-Fi jako medium transmisyjnego drastycznie obniża koszty eksploracji, ale ogranicza zastosowanie urządzenia do monitoringu *post-factum* (brak podglądu w czasie rzeczywistym poza zasięgiem bazy).
3. Integracja z OBD2 jest możliwa, ale wymaga precyzyjnego dostosowania protokołów do konkretnego modelu pojazdu.

**Rekomendacje dalszych prac:** Rozwój projektu powinien kierować się w stronę dodania modułu łączności komórkowej (NB-IoT/LTE-M) dla zapewnienia ciągłej transmisji oraz implementacji zaawansowanych mechanizmów oszczędzania energii. Warto również rozważyć szyfrowanie danych zapisywanych na karcie SD w celu ochrony prywatności użytkownika.

## 6. Wykaz Publikacji

- Dokumentacja ESP32-DevKitC V4

Dostępny: [https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp32-devkitc/user\\_guide.html](https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32/esp32-devkitc/user_guide.html) (odwiedzona 03.12.2025).

- Dokumentacja moduły GNSS

Dostępny: [https://www.waveshare.com/wiki/LC76G\\_GNSS\\_Module](https://www.waveshare.com/wiki/LC76G_GNSS_Module) (odwiedzona 03.12.2025)

- Dokumentacja ThingsBoard

Dostępny: <https://thingsboard.io/docs/guides/> (odwiedzona 03.12.2025)

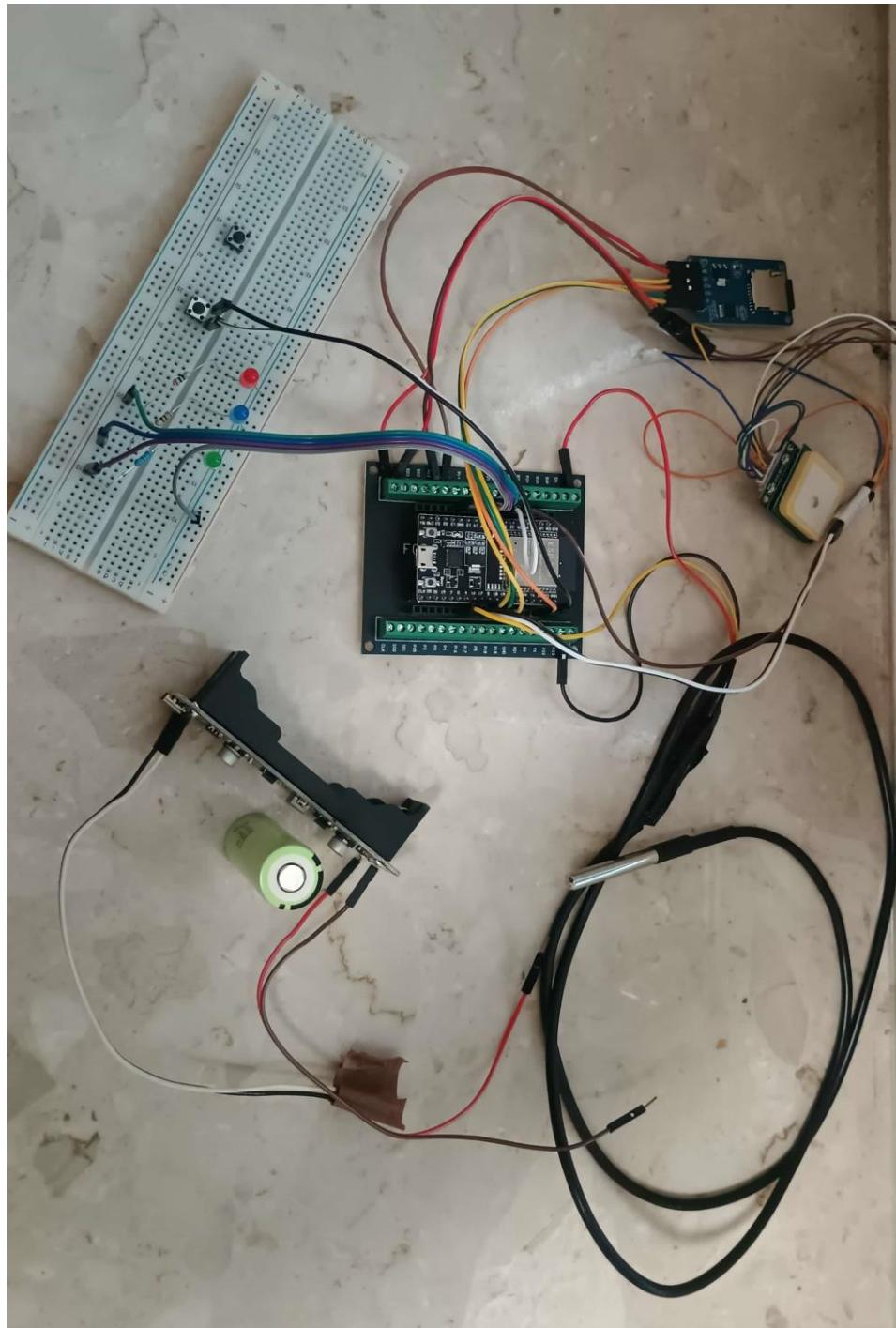
- Biblioteka TinyGPSPlus

Dostępny: <https://github.com/mikalhart/TinyGPSPlus> (odwiedzona 03.12.2025)

- Biblioteka ArduinoJson

Dostępny: <https://arduinojson.org/v7/> (odwiedzona 03.12.2025)

## 7. Załączniki



Rysunek 1 Prototyp urządzenia