Raport końcowy

Rodzaj zajęć
Projekt

Przedmiot
Systemy Mikroprocesorowe i Wbudowane

Rok akademicki	Miasto	Kierunek	Semestr	Prowadzący	Grupa	Sekcja
2025/2026	G	INF	5		2	3

Planowany termin wykonywania ćwiczenia		Faktyczny termin wykonania ćwiczenia	
Data	Godzina	Data	Godzina

Numer ćwiczenia	Temat ćwiczenia
	Taksometr OBDII + GPS

Skład sekcji		
	Imię i nazwisko	Uwagi
1	Jakub Bodzioch	
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Spis treści

Opis zrealizowanego systemu i jego funkcji 3
Wykorzystane elementy elektroniczne 4
ESP32-WROOM-32 (dev-kit w wersji pierwszej): 4
Moduł GPS GY-NEO6M V2: 4
Wyświetlacz LCD 2.8" 240x320 ST7789: 4
Interfejs diagnostyczny VGATE ICAR2: 4
Narzędzia wykorzystane podczas realizacji projektu 5
Specyfikacja wewnętrzna 6
Schemat blokowy urządzenia: 6
Schemat ideowy systemu: 6
Schemat montażowy: 7
Koncept wyglądu płytki PCB: 7
Opis funkcji poszczególnych bloków programowych układu: 8
Algorytm oprogramowania urządzenia: 9
Opis interakcji oprogramowania z układem elektronicznym: 11
Warstwa sprzętowa: 11
Warstwa Oprogramowania: 11
Szczegółowy opis działania ważniejszych procedur oprogramowania: 12
Procedura taskOBD() – Zadanie OBD (FreeRTOS): 12
Procedura taskGPS() – Zadanie GPS (FreeRTOS): 13
Specyfikacja zewnętrzna 14
Opis funkcji elementów sterujących urządzeniem: 14
Opis funkcji elementów wykonawczych: 14
Opis reakcji oprogramowania na zdarzenia zewnętrzne: 14
Skrócona instrukcja obsługi urządzenia: 15
1. Uruchomienie i przygotowanie: 15
2. Konfiguracja (opcjonalnie): 15
3. Obsługa trasy: 15
4. Zakończenie podróży: 15
Szybkie rozwiązywanie problemów: 16

Opis montażu i uruchamiania systemu 17
Problemy napotkane podczas projektowania systemu: 17
Problem z parowaniem urządzenia Bluetooth i obsługą kodu PIN: 17
Brak możliwości połączenia ESP32 z interfejsem OBD-II: 17
Nieprawidłowa kolejność inicjalizacji Bluetooth na ESP32: 17
Brak standardowego PID przebiegu w OBD-II: 17
Problemy z watchdogiem (WDT) i przekraczaniem czasu wykonania zadań: 18
Testowanie urządzenia: 19
Wnioski z uruchamiania i testowania systemu: 21
Kosztorys projektu 22
Linki do ważniejszych elementów i narzędzi wymienionych w dokumencie 23
Załączniki do dokumentu: 24

Opis zrealizowanego systemu i jego funkcji

Zaprojektowane urządzenie o pseudonimie roboczym Cabulator to system telematyki taksówkarskiej, przeznaczony dla aut prywatnych, zbudowany na platformie ESP32 z wyświetlaczem dotykowym TFT, łączący dane z modułu GPS i diagnostyki pojazdu OBD-II. System realizuje funkcję cyfrowego taksometru z rozbudowanymi możliwościami monitorowania i rejestracji przejazdu.

Główne komponenty systemu obejmują:

- **Czytnik GPS** odpowiedzialny za śledzenie pozycji pojazdu i synchronizację czasu systemowego z danymi satelitarnymi,
- **Moduł OBD-II** umożliwiający odczyt przebiegu i konsumpcji paliwa z pojazdu,
- **Kartę SD** do archiwizacji tras w formacie CSV,
- **Wyświetlacz TFT 3,2"** z interfejsem dotykowym do interakcji użytkownika.

Architektura systemu opiera się na wielowątkowym modelu RTOS z dedykowanymi taskami dla GPS i OBD, podczas gdy główna pętla aplikacji obsługuje logikę interfejsu użytkownika i zmianę ekranów. Interfejs posiada 12 ekranów funkcjonalnych w tym: ekran główny jako centralne menu, ekran aktywnej trasy, monitorujący dystans i koszt przejazdu, ekran ustawień taryfy (rozliczenie za km lub litr paliwa), panel konfiguracji jasności, ekrany diagnostyczne GPS i OBD z szczegółowymi danymi satelitów i parametrów pojazdu, ekran ustawień połączeń, oraz ekran informacyjny. Dane przejazdu przechowywane są na karcie SD w strukturze katalogów czasowych z logami i końcowymi statystykami trasy, a parametry taryfy są trwale zapisywane w pamięci mikroprocesora.

Wykorzystane elementy elektroniczne

ESP32-WROOM-32 (dev-kit w wersji pierwszej):

Wybrany jako mikrokontroler główny ze względu na zaawansowane możliwości komunikacyjne: obsługę dwóch niezależnych magistrali SPI (dla wyświetlacza TFT i karty SD bez konfliktów), UART dla modułu GPS, Bluetooth dla komunikacji z modułem OBD-II, a także dużej ilości portów GPIO niezbędnych do sterowania podświetleniem, czytnikiem SD i obsługi dotyku.

Taktowanie procesora 240 MHz zapewnia płynne działanie interfejsu dotykowego w rzeczywistym czasie, a wbudowana obsługa RTOS (FreeRTOS) umożliwia wielowątkowe wykonywanie zadań (tasków dla GPS i OBD równocześnie). Dodatkowo wybrany model umożliwia nawiązywanie połączenie Bluetooth w wersji SPP, które jest jedynym wspieranym standardem w tańszych interfejsach OBD-II.

Moduł GPS GY-NEO6M V2:

Zastosowany ze względu na niezawodność komunikacji przez UART, dokładność pozycjonowania (około 2,5 m), wsparcie dla NMEA 0183 (parsowanie przez bibliotekę TinyGPSPlus), możliwość synchronizacji czasu systemowego z danymi satelitarnymi oraz niskie zapotrzebowanie energetyczne (w okolicy 60 mA). Jest to również jeden z tańszych modułów dostępnych na rynku.

Wyświetlacz LCD 2.8" 240x320 ST7789:

Zapewnia wystarczającą rozdzielczość do wyświetlenia 12 ekranów funkcjonalnych z przejrzystym interfejsem dotykowym, obsługę SPI (szybkiej komunikacji), odporność na działanie słoneczne w warunkach pracy na desce rozdzielczej samochodu, oraz kompatybilność z biblioteką TFT_eSPI zoptymalizowaną dla ESP32. Dodatkowo moduł posiada w sobie wbudowany interfejs kart SD. Umożliwia on obsługę zarówno wyświetlania, jak i odczytu za pomocą jednej magistrali SPI, jednak jak już wcześniej wspomniałem, zdecydowałem się wykorzystać do tego celu osobny interfejs HSPI.

Interfejs diagnostyczny VGATE ICAR2:

Umożliwia odczyt danych pojazdu przez Bluetooth. Posiada on krótszy czas łączenia ze sterownikiem silnika ECU samochodu od większości interfejsów oferowanych na rynku. Wykorzystuje on ELM327 w wersji 2.2, który charakteryzuje się szeroką kompatybilnością z pojazdami od 1996 r. oraz intuicyjnym sterowaniem komendami AT. Ten konkretny model

obsługuje połączenie bluetooth w standardzie wspieranym przez mikrokontroler, czyli w standardzie SPP.

Narzędzia wykorzystane podczas realizacji projektu

IDE: VS Code + PlatformIO – rozwój na ESP32, kompilacja, upload, zarządzanie bibliotekami projektu.

Dokumentacja kodu: Doxygen – generowanie HTML/LaTeX z komentarzy w kodzie.

Oprogramowanie biurowe: Microsoft Office (Word/PowerPoint) – dokumentacja i prezentacje postępów.

Projekt CAD/PCB: KiCad – schemat, PCB, pliki produkcyjne.

Modelowanie 3D: Shapr3D – projekt obudowy pod druk 3D, szybkie iteracje.

Wytwarzanie mechaniczne: Drukarka 3D Bambulab A1 mini – prototypowanie obudowy systemu.

Realizacja płytki projektu: Lutownica + materiały – montaż przewodów, złącz i modułów.

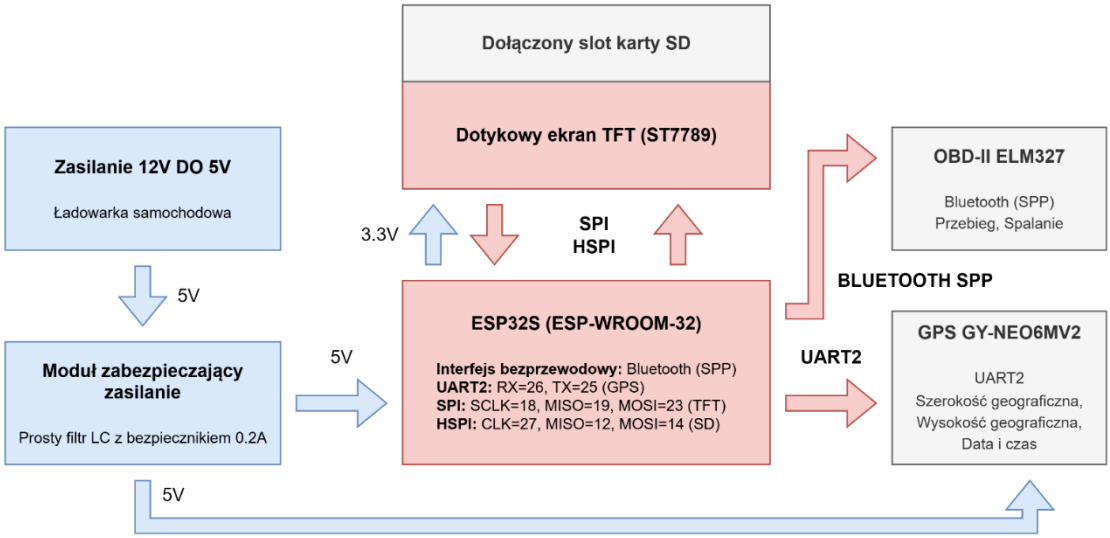
Diagnostyka pojazdu: Vgate iCar2 (ELM327) – OBD-II przez Bluetooth, komendy AT, test PID.

Pomiary: Multimetr – weryfikacja zasilania i sygnałów.

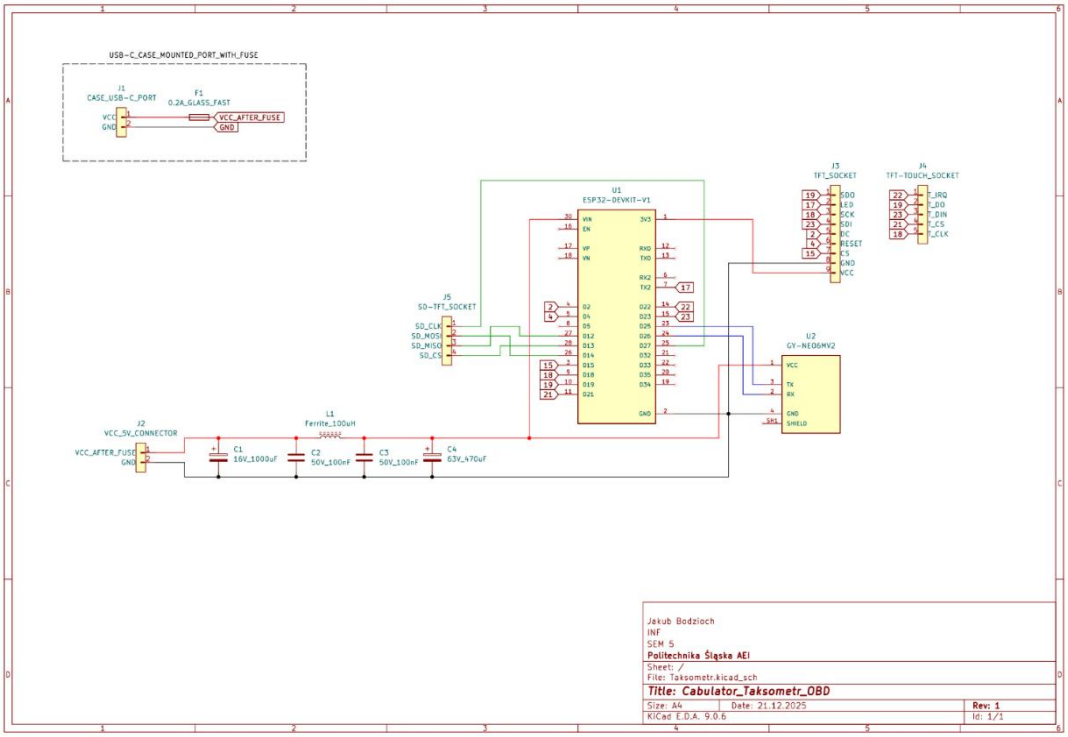
Symulacje obwodów: LTSpice – analiza i walidacja projektu filtra przed montażem. Pozwala sprawdzić charakterystykę częstotliwościową, odpowiedź impulsową i wpływ tolerancji elementów, co zmniejsza ryzyko wystąpienia błędów.

Specyfikacja wewnętrzna

Schemat blokowy urządzenia:

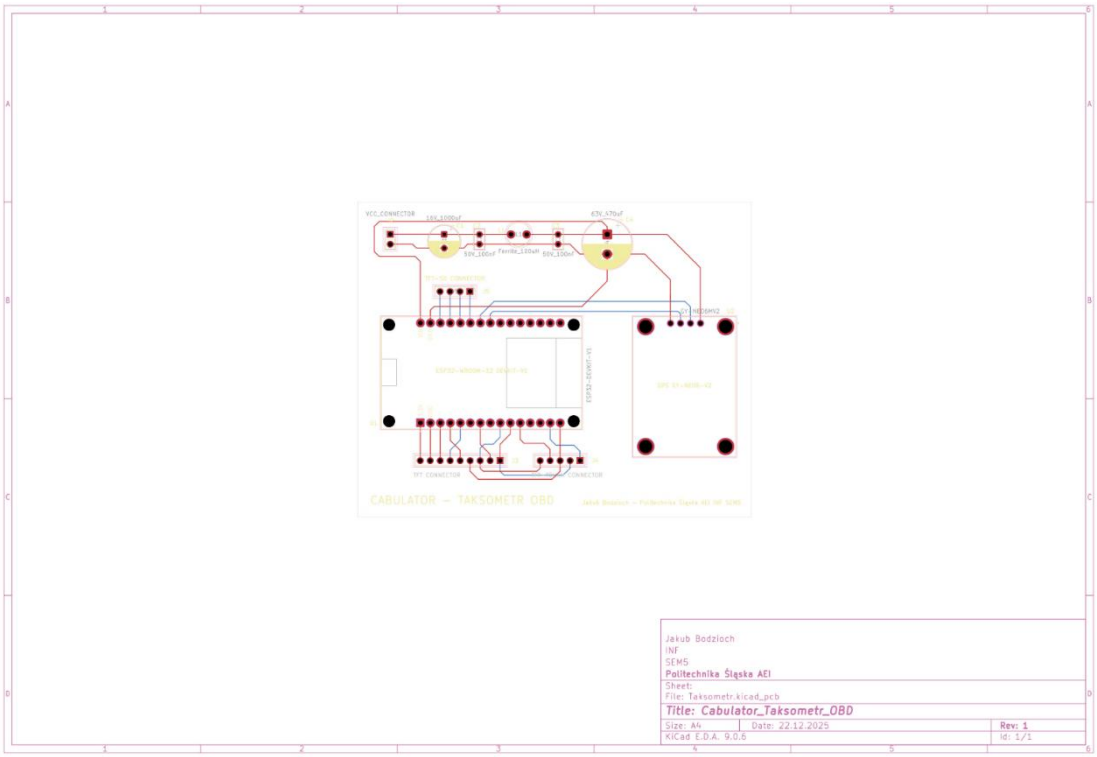


Schemat ideowy systemu:



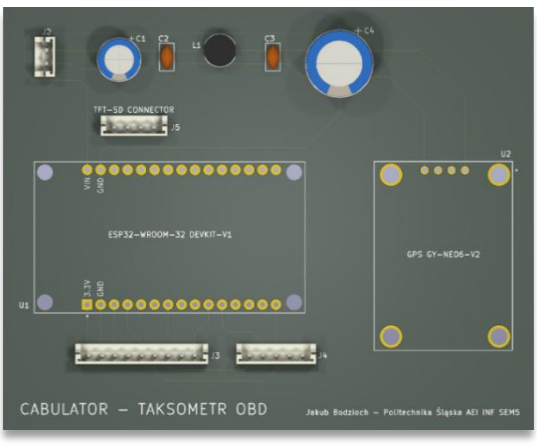
Dostępny również w pełnym formacie PDF.

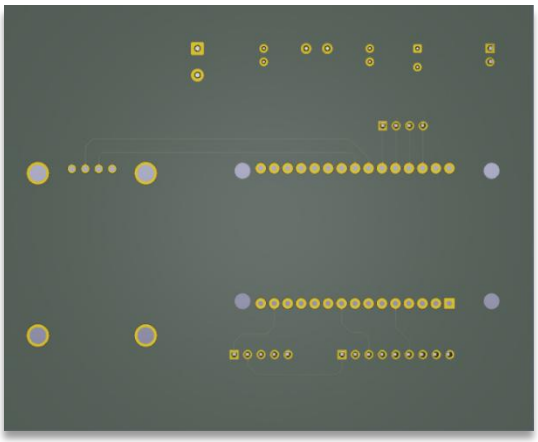
Schemat montażowy:



Dostępny również w pełnym formacie PDF.

Koncept wyglądu płytki PCB:





Opis funkcji poszczególnych bloków programowych układu:

Aplikacja główna: inicjalizacja podzespołów, start zadań RTOS, pętla UI i routing dotyku.

main.cpp

Moduł GPS: inicjalizacja UART, parsowanie NMEA (TinyGPSPlus), publikacja GPS::Fix, debug statusu, synchronizacja czasu, bufor surowych ramek.

gps_reader.h, gps_reader.cpp

Moduł OBD: połączenie przez BluetoothSerial, inicjalizacja ELM327, wysyłanie komend AT, odczyt PIDs, okresowe zadanie OBD::task().

obd_reader.h, obd_reader.cpp

Logger SD: inicjalizacja HSPI, tworzenie folderu trasy, zapis gps_log.csv i trip_data.csv, finalizacja trasy.

sd_manager.h, sd_manager.cpp

Wyświetlacz/Backlight: konfiguracja TFT, kalibracja dotyku, sterowanie jasnością (PWM).

tft_display.cpp

Zarządzanie ekranami: definicja ScreenState i globalnego currentScreen do routingu UI.

screen_manager.h

Ekrany UI: widoki i ich handlery (Home, Settings, Brightness, Connection, About, GPS, GPS-Debug, OBD, OBD-Debug, Tariff, Trip).

Przykłady: screen_home.h, screen_tariff.h, screen_trip.h

Konfiguracja systemu: piny, parametry GPS/SD, adres MAC OBD, tryb symulacji OBD, PIDs.

cabulator_settings.h

Zachęcam do zapoznania się z dokumentacją w doxygenie, która w bardzo przejrzysty sposób opisuje relację poszczególnych modułów kodu.

Algorytm oprogramowania urządzenia:

Ze względu na obszerność i modularną strukturę projektu (wiele plików w katalogach include/ i src/ oraz wygenerowanej dokumentacji Doxygen), w raporcie końcowym zamieszczam jedynie reprezentatywny fragment kodu, który pokazuje architekturę uruchomienia systemu, podział na zadania RTOS.

Pełny kod źródłowy wraz z komentarzami, plikami nagłówkowymi i implementacjami modułów (GPS, OBD, SD, TFT) udostępniam w repozytorium GitHub. Taki podział pozwala zachować czytelność dokumentu i jednocześnie zapewnia pełną możliwość weryfikacji oraz uruchomienia projektu.

```
// ============================================================================
// GLOBALNE ZMIENNE
// ============================================================================
TFT_eSPI tft;
PNG;
String currentTripPath = ""; // Ścieżka do folderu obecnej trasy

// ============================================================================
// TASKI RTOS - Wielowątkowe wykonanie
// ============================================================================
// Task OBD - czytanie danych z modułu OBD co 2 sekundy
void taskOBD(void* param) {
  OBD::task(param);
}

// Task GPS - odczyt i debug GPS co 10 sekund
void taskGPS(void* param) {
  static bool timeSync = false;    // Flaga synchronizacji czasu systemowego z GPS

  while (true) {

    GPS::Fix;

    if (GPS::poll(fix)) {

      GPS::debugStatus();
      // Synchronizacja czasu systemowego z GPS
      if (!timeSync && fix.dateTimeValid) {

        if (GPS::setSystemTimeFromGPS(fix)) {

          timeSync = true;
          Serial.println("[GPS] System time synchronized from GPS");
        }
      }

    }
    vTaskDelay(10000 / portTICK_PERIOD_MS); // Opóźnienie 10 sekund
  }
}
```

```
// ============================================================================
// SETUP - Inicjalizacja systemu
// ============================================================================
void setup() {

  Serial.begin(115200);

  // Wyłączenie WDT dla rdzenia 0 (uniknięcie resetów podczas długich operacji)
  disableCore0WDT();
  // Całkowite wyciszenie logów (przynajmniej próbowałem bo obd nadal pluje faktami)
  esp_log_level_set("*", ESP_LOG_NONE);

  Serial.println("\n\n\n[SYSTEM] ========== INITIALIZING ==========\n");

  // ========== FILESYSTEM ==========
  LittleFS.begin(true);
  Serial.println("[FS] ========================");
  Background::listFS("/");
  Serial.println("[FS] ========================\n");

  // ========== WYŚWIETLACZ ==========
  initTFT(&tft);
  Background bg("/loading.png");
  bg.draw(tft, png, true);
  currentScreen = SCREEN_WELCOME;

  // ========== EEPROM & JASNOŚĆ & TARYFA ==========
  EEPROM.begin(100);          // Zarezerowanie 100 bajtów w EEPROM
  initBrightnessModule();
  loadTariffFromEEPROM();

  // ========== INICJALIZACJA KARTY SD ==========
  if (!SDManager::init()) {
    Serial.println("[WARNING] SD Card initialization failed, continuing anyway\n");
  }

  // Rekalibracja dotyku po inicjalizacji SD
  uint16_t calData_recal[5] = { 243, 3566, 356, 3415, 1 };
  tft.setTouch(calData_recal);
  Serial.println("[TOUCH] Re-calibrated after SD init");

  // ========== INICJALIZACJA GPS I OBD ==========
  GPS::begin();

  if (!OBD::init()) {
    Serial.println("[WARNING] OBD initialization failed, continuing anyway\n");
  }

  Serial.println("[SYSTEM] ========== INIT COMPLETE ==========\n");
```

Opis interakcji oprogramowania z układem elektronicznym:

Warstwa sprzętowa:

ESP32-WROOM-32: jednostka centralna; UART (GPS), Bluetooth SPP (OBD), SPI (TFT/touch), HSPI (SD),

GPS GY-NEO6M: strumień NMEA przez UART2 (9600 bps), źródło pozycji i czasu.

OBD-II Vgate iCar2 (ELM327): łączność Bluetooth, komendy AT i PID do odczytu parametrów pojazdu.

TFT 2.8" ST7789 + dotyk: interfejs użytkownika, transfer po SPI, sterowanie jasnością przez PWM.

Karta SD (HSPI): niezależna magistrala SPI do logowania danych, struktura katalogów per trasa.

Zasilanie + filtr: filtracja zakłóceń z instalacji 5V z ładowarki samochodowej, stabilizacja dla modułów.

Warstwa Oprogramowania:

Start systemu: inicjalizacja FS, TFT, EEPROM, SD, GPS, OBD oraz ekranu startowego w main.cpp

Zadania RTOS: taskGPS (co około 10 s) oraz taskOBD (co około 2 s) uruchamiane przez xTaskCreate, pętla loop() obsługuje dotyk i odświeżanie ekranów.

GPS: GPS::begin() konfiguruje UART2, GPS::poll() parsuje NMEA (TinyGPSPlus) i zwraca GPS::Fix, GPS::setSystemTimeFromGPS() synchronizuje czas RTC z GPS.

OBD: OBD::init() zestawia Bluetooth SPP i inicjuje ELM327, OBD::task() cyklicznie odczytuje PID aktualizując metryki przejazdu.

TFT i dotyk: initTFT() konfiguruje wyświetlacz i kalibrację; w loop() wywołania tft.getTouch() są mapowane na handlery ekranów (np. handleHomeTouch()).

Podświetlenie: sterowanie jasnością przez PWM (funkcje ledcSetup/ledcWrite) oraz dedykowana obsługa na ekranie Brightness.

SD/FS: SDManager::init() uruchamia HSPI dla karty SD, dane GPS i podsumowania tras zapisywane do CSV, LittleFS służy do zasobów (np. grafika ekranu powitalnego).

EEPROM: EEPROM.begin() i funkcje taryfy (np. loadTariffFromEEPROM()) oraz okresowy zapis stanu trasy.

Szczegółowy opis działania ważniejszych procedur oprogramowania:

Procedura taskOBD() – Zadanie OBD (FreeRTOS):

Zadanie wykonywane w pętli, gdy tripActive == true, z opóźnieniami między odczytami:

```
if (tripActive) {
   long odoRaw = readOdometer();     // PID 0x22DD01 dla Volvo V40
   float fuel = readFuelRate();      // PID 0x0110 (MAF) → przeliczenie na L/H

   if (!tripPaused && lastMillis > 0) {
      float hours = (now - lastMillis) / 3600000.0f;
      float deltaDist = max(0.0f, dist - lastOdo);
      distanceTraveled += deltaDist;
      fuelUsed += fuel * hours;

      // Zapis na SD co 10 sekund
      if (now - lastSDUpdate >= 10000) {
         SDManager::TripUpdateData updateData { distanceTraveled, fuelUsed };
         SDManager::updateTripFile(currentTripPath, updateData);
      }
   }
}
```

readOdometer(): wysyła komendę AT ATSH, ATCRA, 2210 do modułu ELM327 przez Bluetooth, parsuje odpowiedź 62DD01 i konwertuje heksadecymalnie na wartość odometru (przebiegu) [km].

readFuelRate(): odczytuje MAF (Mass Air Flow), przelicza na L/H: fuelRate = (MAF / 14.7 / 0.755) * 3.6.

Liczenie przyrostów: dystans i paliwo akumulują się tylko gdy trasa nie jest zapauzowana (!tripPaused), różnice obliczane są iteracyjnie z każdym odczytem.

Zapis na SD co 10 sekund: SDManager::updateTripFile() aktualizuje trip_data.csv z bieżącymi wartościami.

Obliczanie kosztu:
calculateCost() = distanceTraveled * tariffValue dla trybu TARIFF_PER_KM,
fuelUsed * tariffValue dla trybu TARIFF_PER_LITRE.

Procedura taskGPS() – Zadanie GPS (FreeRTOS):

Zadanie wykonywane co ~10 sekund (opóźnienie vTaskDelay(10000 / portTICK_PERIOD_MS)):

```
GPS::Fix;
if (GPS::poll(fix)) {  // Nieblokujący polling UART, parsowanie NMEA (TinyGPSPlus)
   GPS::debugStatus();
   if (!timeSync && fix.dateTimeValid) {
      if (GPS::setSystemTimeFromGPS(fix)) {
         timeSync = true;  // Flaga synchronizacji czasu systemowego z GPS
      }
   }
}
```

GPS::poll(fix): czyta dostępne znaki z UART2, parsuje ramki NMEA (pozycja, czas, satelity, HDOP), zwraca true jeśli upłynął czas próbkowania (SAMPLE_MS = 1000 ms).

Struktura GPS::Fix zawiera: valid (czy fix ważny), lat/lng (współrzędne), sats (liczba satelitów), hdop (precyzja pozioma), year/month/day/hour/minute/second (data/czas), dateTimeValid.

Synchronizacja czasu systemowego: GPS::setSystemTimeFromGPS() ustawia RTC ESP32 z daty/czasu GPS (jednokrotnie, gdy flaga timeSync == false), zostaje użyty do stemplowania logów SD i EEPROM.

Callback SDManager::onGPSFix(): wywoływany wewnątrz GPS::poll(), aby zapisać dane GPS do gps_log.csv na karcie SD.

Specyfikacja zewnętrzna

Opis funkcji elementów sterujących urządzeniem:

Ekran dotykowy TFT: główne UI. Gesty dotyku wybierają przyciski ekranowe (start/stop trasy, pauza, zmiana taryfy, podgląd GPS/OBD, ustawienia jasności i połączeń).

Moduł OBD-II (Vgate iCar2) przez Bluetooth: zewnętrzne źródło danych pojazdu. Jego dostępność traktowana jest jako zdarzenie sterujące. Gdy nie ma z nim połączenia urządzenie blokuje możliwość rozpoczęcia trasy.

GPS (GNSS) przez UART: dostarcza fix pozycji/czasu. Brak fixu lub niski HDOP wpływa na status i logowanie trasy.

Karta SD: obecność/nośnik gotowości jest sprawdzana przy starcie. Decyduje, czy logi i pliki CSV będą zapisywane.

Zasilanie (5 V) + filtr: stan zasilania wpływa na stabilność. Przy zaniku użytkownik bazuje na ostatnich zapisach (EEPROM/logi SD).

Opis funkcji elementów wykonawczych:

Wyświetlacz TFT 2.8" (ST7789) z podświetleniem PWM: prezentuje ekrany (Home, Trip, GPS/OBD info, Settings, Tariff, Brightness). Podświetlenie ekranu sterowane poprzez jeden z dostępnych ekranów umożliwia dopasowanie jasności do warunków w kabinie.

Sygnalizacja statusu na ekranie: dane przedstawiane są w postaci napisów bezpośrednio na odpowiednich ekranach systemu. GPS (fix/HDOP/satelity), OBD (połączony/rozłączony), SD (gotowość), taryfa (tryb per km / per litr), stan trasy (aktywna/pauza/zakończona).

Opis reakcji oprogramowania na zdarzenia zewnętrzne:

Dotyk ekranu: natychmiastowe przejście do odpowiednich ekranów/akcji (start/pauza/stop trasy, zmiana taryfy, podgląd diagnostyczny).

Utrata fixu GPS lub słaby sygnał (wysokie HDOP): status GPS na ekranie przechodzi w ostrzeżenie. Logi GPS mogą mieć flagę nieważności, lub brak synchronizacji czasu.

Brak połączenia OBD/Bluetooth lub błąd PID: status OBD na ekranie wskazuje rozłączenie. Metryki spalania/dystansu mogą być wstrzymane lub oznaczone jako niedostępne.

Niedostępna karta SD: system pracuje dalej, ale informuje o braku logowania na nośnik. Dane krytyczne częściowo zabezpieczone przez EEPROM (np. stan trasy).

Zanik/niestabilność zasilania: dzięki okresowemu zapisowi do EEPROM i SD minimalizowana jest utrata danych, po restarcie użytkownik może wznowić nową trasę, mając wcześniejsze logi.

Synchronizacja czasu z GPS: pierwszy ważny fix ustawia zegar systemowy, jeśli brak daty/czasu z satelitów, używany jest czas systemowy.

Skrócona instrukcja obsługi urządzenia:

1. Uruchomienie i przygotowanie:

Zasilanie: Podłącz kabel do portu USB-C umiejscowionego z boku urządzenia. W samochodzie użyj ładowarki o mocy min. 10W.

Start systemu: Poczekaj na ekranie powitalnym, aż urządzenie połączy się z modułem OBD-II.

Gotowość do pracy: Na ekranie głównym sprawdź ikony statusu: GPS, OBD oraz SD. Powinny być aktywne przed ruszeniem w trasę.

2. Konfiguracja (opcjonalnie):

Jasność: Dostosuj podświetlenie w menu Brightness.

Koszty: W menu Tariff wybierz sposób naliczania (zł/km lub zł/litr) i wpisz swoją stawkę.

3. Obsługa trasy:

Start: Aby zacząć pomiar, wejdź w zakładkę Home/Trip i naciśnij Start.

Podgląd: W trakcie jazdy możesz śledzić dystans, zużycie paliwa i aktualny koszt.

Pauza i wychodzenie: Jeśli chcesz na chwilę wyjść do menu głównego bez przerywania pomiarów, naciśnij X w prawym górnym rogu.

UWAGA: Jeśli naciśniesz X, gdy trasa jest zapauzowana, urządzenie uzna podróż za zakończoną i zresetuje wszystkie parametry.

4. Zakończenie podróży:

Aby zakończyć trasę, najpierw użyj przycisku pauzy, a następnie wyjdź z ekranu trasy (przycisk X). Podsumowanie zostanie automatycznie zapisane na karcie SD.

Szybkie rozwiązywanie problemów:

Brak zapisu na SD: Sprawdź, czy karta jest dobrze włożona. W razie potrzeby zrestartuj urządzenie. Bez karty urządzenie działa normalnie, ale nie zapisuje historii (logów).

Brak sygnału GPS: Upewnij się, że urządzenie ma bezpośrednią widoczność nieba. W trudnych warunkach, złapanie pierwszego sygnału może zająć nawet do 2 minut.

Brak danych z auta (OBD): Sprawdź, czy interfejs Vgate iCar2 jest włączony i sparowany. Dane odczytywane będą automatycznie po odzyskaniu połączenia.

Nagły restart/Zanik prądu: Urządzenie co 30 sekund zapisuje postęp podróży. Po restarcie po prostu naciśnij Start. Stare dane zostaną automatycznie przywrócone i będziesz mógł kontynuować trasę.

Urządzenie nie włącza się: Sprawdź bezpiecznik urządzenia. Z boku obudowy koło portu zasilania znajduje się puszka bezpiecznika. Należy ją odkręcić i sprawdzić czy zamontowany w obudowie bezpiecznik nie jest przepalony. W przypadku wymaganej wymiany, należy zastosować bezpiecznik szklany tupu Fast o progu minimalnie 0.2A a maksymalnie 0.3A.

Opis montażu i uruchamiania systemu

Problemy napotkane podczas projektowania systemu:

Problem z parowaniem urządzenia Bluetooth i obsługą kodu PIN:

Problem: Podczas próby parowania ESP32 z interfejsem OBD-II wymagane było podanie kodu PIN „1234”. Początkowo wywołanie funkcji setPin() powodowało błąd kompilacji lub zwracało wartość false.

Rozwiązanie: Zidentyfikowano zmianę sygnatury funkcji setPin() w nowszych wersjach Arduino Core dla ESP32. Zastosowano poprawne wywołanie funkcji z podaniem długości kodu.

Brak możliwości połączenia ESP32 z interfejsem OBD-II:

Problem: Interfejs OBD-II poprawnie łączył się z aplikacją mobilną, jednak ESP32 nie było w stanie nawiązać połączenia i wielokrotnie zgłaszało nieudane próby połączenia.

Rozwiązanie: Ustalono, że aplikacja mobilna nie może być jednocześnie połączona z interfejsem OBD-II w momencie próby połączenia przez ESP32. Dodatkowo wyjaśniono różnicę pomiędzy stanem „sparowane” a „połączone” urządzenia Bluetooth i upewniono się, że interfejs nie jest aktywnie połączony z innym urządzeniem.

Nieprawidłowa kolejność inicjalizacji Bluetooth na ESP32:

Problem: Różne wersje kodu inicjalizowały Bluetooth w odmiennej kolejności, co prowadziło do niejednoznacznych rezultatów i problemów z połączeniem.

Rozwiązanie: Zastosowano stabilną i sprawdzoną sekwencję inicjalizacji.

Brak standardowego PID przebiegu w OBD-II:

Problem: Standard OBD-II nie gwarantuje dostępności przebiegu pojazdu. Standardowy PID 01 A6 okazał się niewspierany przez testowany pojazd.

Rozwiązanie: Zastosowano specyficzny PID dla pojazdów Volvo V40 (2014+)

Problemy z watchdogiem (WDT) i przekraczaniem czasu wykonania zadań:

Problem: Podczas realizacji projektu występowały częste resety mikrokontrolera spowodowane przez mechanizm watchdog (WDT). Problem pojawiał się głównie podczas działania zadania odpowiedzialnego za komunikację z interfejsem OBD-II.

Zadanie OBD wykonywało operacje blokujące (oczekiwanie na odpowiedź Bluetooth / OBD) i w niektórych przypadkach nie kończyło się w założonym maksymalnym czasie wykonania, co powodowało wyzwolenie watchdoga i restart systemu.

Rozwiązanie: Kod zadania OBD został uproszczony i zrefaktoryzowany w taki sposób, aby:

- ograniczyć liczbę operacji blokujących,
- skrócić czas pojedynczego cyklu zadania,
- unikać długotrwałego oczekiwania na odpowiedź z interfejsu OBD-II,

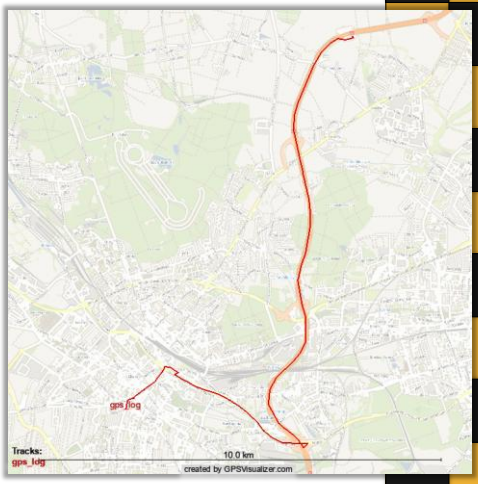
Testowanie urządzenia:

Gliwice – Dąbowa Górnicza

Jazda autostradą A1 poza miastem z dostępem do dobrego sygnału GPS. Poruszanie się prędkościami powyżej 100 km/h.

Dane pozyskane z sytemu:
Długość trasy: 19.00 km,
Spalone paliwo: 1.528 l,
Tryb taryfy: 1 (zł/l),
Wartość taryfy: 6.50 zł/l,
Należność: 9.93 zł

Dane pozyskane z systemu samochodu:
Długość trasy: 18.4 km
Średnie spalanie na trasie: 7.3 l/100km
(1,34 l/18.4km)



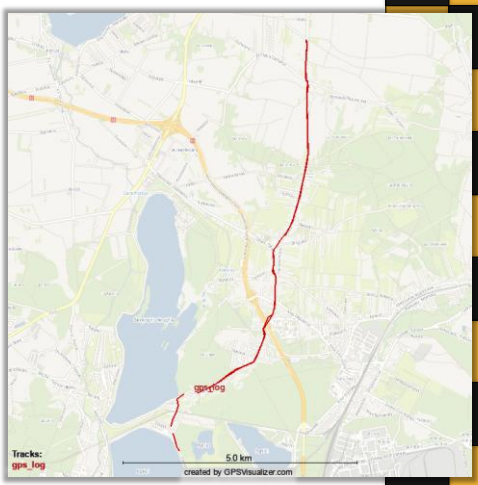
Fragment prezentacji końcowej projektu.

Dąbrowa Górnicza – Poza miastem

Jazda poza miastem z dostępem do, momentami słabego sygnału GPS. Poruszanie się prędkościami do 90 km/h.

Dane pozyskane z sytemu:
Długość trasy: 19.00 km,
Spalone paliwo: 1.625 l,
Tryb taryfy: 1 (zł/l),
Wartość taryfy: 5.50 zł/l,
Należność: 8.93 zł

Dane pozyskane z systemu samochodu:
Długość trasy: 18.9 km
Średnie spalanie na trasie: 8.1 l/100km
(1.53 l/18.9km)



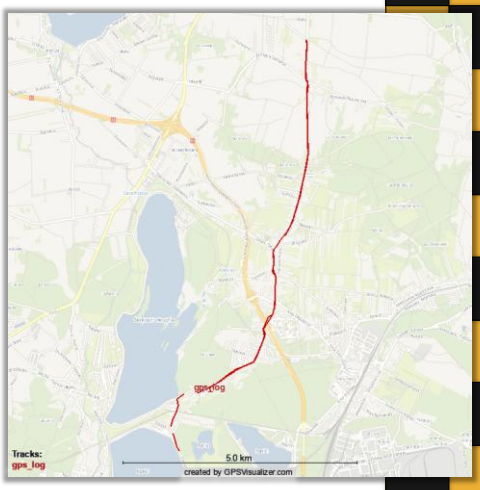
Fragment prezentacji końcowej projektu.

Dąbrowa Górnicza – Poza miastem

Jazda poza miastem z dostępem do, momentami słabego sygnału GPS. Poruszanie się prędkościami do 90 km/h.

Dane pozyskane z sytemu:
Długość trasy: 19.00 km,
Spalone paliwo: 1.625 l,
Tryb taryfy: 1 (zł/l),
Wartość taryfy: 5.50 zł/l,
Należność: 8.93 zł

Dane pozyskane z systemu samochodu:
Długość trasy: 18.9 km
Średnie spalanie na trasie: 8.1 l/100km (1.53 l/18.9km)



Fragment prezentacji końcowej projektu.

Wnioski z uruchamiania i testowania systemu:

Przeprowadzone testy w ruchu miejskim, pozamiejskim oraz autostradowym potwierdziły poprawne działanie systemu w zróżnicowanych warunkach eksploatacyjnych. System był w stanie rejestrować dane zarówno przy niskich prędkościach i częstych zmianach tempa jazdy, jak i przy długotrwałej jeździe ze stałą, wysoką prędkością.

Porównanie długości tras oraz zużycia paliwa obliczonego przez system z danymi pochodzącymi z systemu samochodu wykazało: niewielkie różnice w długości tras, wynikające głównie z niedokładności sygnału GPS, zbliżone wartości zużycia paliwa, mieszczące się w akceptowalnym zakresie błędu pomiarowego. Największe rozbieżności występowały w warunkach miejskich, co można przypisać gorszemu sygnałowi GPS oraz częstym zatrzymaniom pojazdu.

Po uproszczeniu zadania odpowiedzialnego za komunikację z OBD-II i skróceniu czasu jego wykonywania, system działał stabilnie bez występowania resetów spowodowanych przez watchdog. Potwierdza to poprawność przyjętych założeń dotyczących zarządzania czasem wykonywania zadań oraz konieczność unikania operacji blokujących w systemach czasu rzeczywistego.

Na podstawie przeprowadzonych testów można stwierdzić, że zaprojektowany system: umożliwia poprawny odczyt parametrów eksploatacyjnych pojazdu, pozwala na wiarygodne szacowanie zużycia paliwa oraz kosztów przejazdu, działa stabilnie w długotrwałych testach drogowych.

System spełnia założenia projektowe i może stanowić podstawę do dalszej rozbudowy, np. o transmisję sieciową lub wizualizację wyników.

Kosztorys projektu

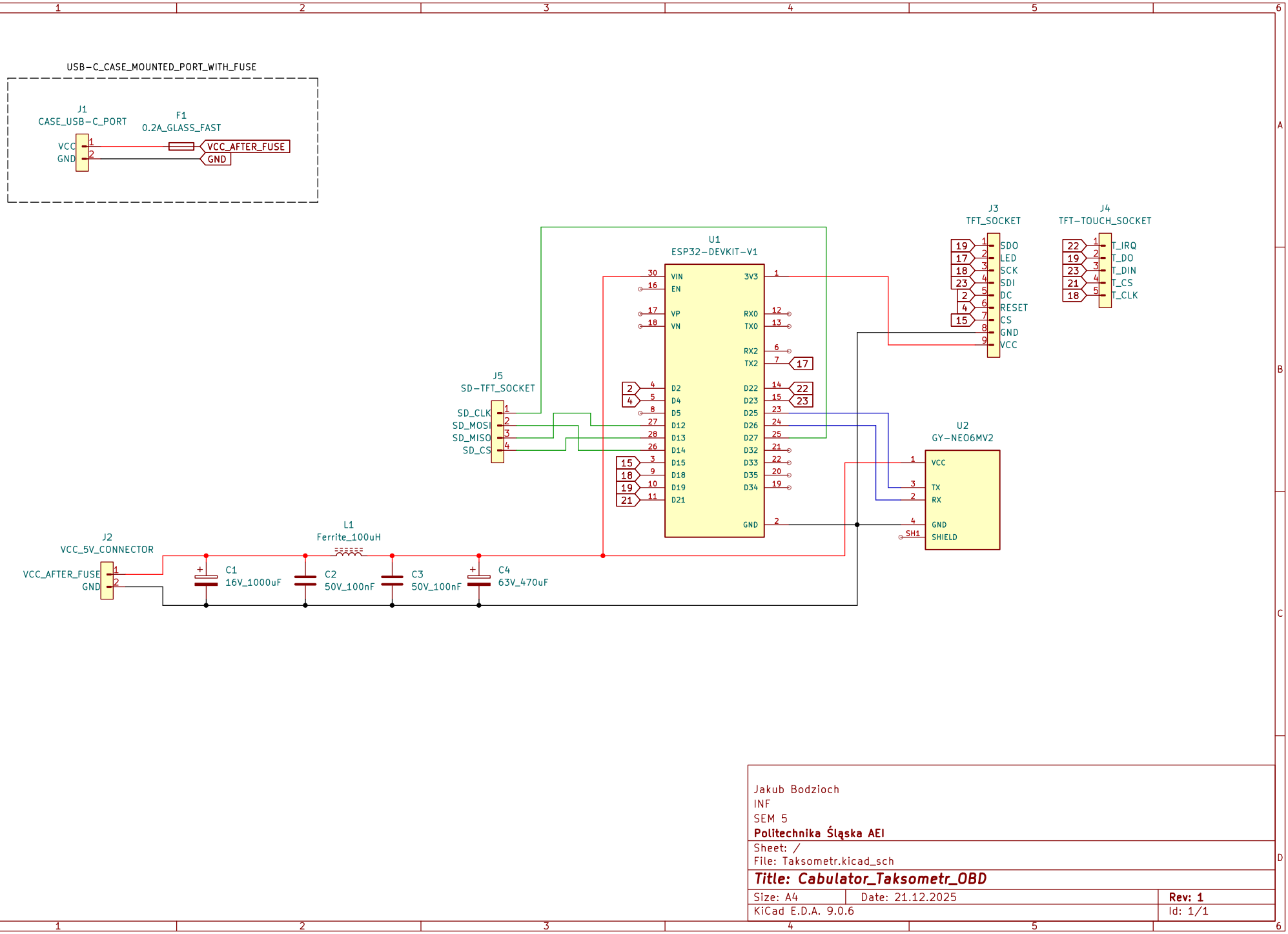
KOSZTORYS URZĄDZENIA					
Element	**Ilość (w zestawie)**	**Ilość (w urządzeniu)**	**Cena (zestawu)**	**Cena (za sztukę)**	**SUMA**
ESP-32S ESP-WROOM-32 WiFi+Bluetooth ESP32 NodeMCU	1	1	26,90 zł	26,90 zł	**26,90 zł**
Wyświetlacz LCD 2.8" 240x320 ST7789 SPI dotykowy slot microSD	1	1	46,80 zł	46,80 zł	**46,80 zł**
Moduł GPS GY-NEO6MV2 NEO-6M z anteną	1	1	23,75 zł	23,75 zł	**23,75 zł**
Zestaw gniazda + wtyczki XH2.54 2Pin 3Pin 4Pin 5Pin	230	2	10,56 zł	0,05 zł	**0,09 zł**
Zestaw Bezpieczniki szklane 5x20mm	100	1	9,14 zł	0,09 zł	**0,09 zł**
Zestaw gniazd goldpin 2.54mm 40Pin	10	1	6,99 zł	0,70 zł	**0,70 zł**
Zestaw goldpin 2.54mm 40Pin	10	1	5,29 zł	0,53 zł	**0,53 zł**
Kondensator ceramiczny 50V 100nF	100	2	5,01 zł	0,05 zł	**0,10 zł**
Złącze USB 3.1 Typ-C 2-pinowe do montażu w obudowie	5	1	5,62 zł	1,12 zł	**1,12 zł**
Cewka indukcyjna miedziana ferryt 100uH	10	1	3,89 zł	0,39 zł	**0,39 zł**
Aluminiowe kondensatory elektrolityczne 460uF	10	1	6,72 zł	0,67 zł	**0,67 zł**
Aluminiowe kondensatory elektrolityczne 1000uF	10	1	4,39 zł	0,44 zł	**0,44 zł**
Protoboard PCB 6x8cm jednostronna	1	1	5,94 zł	5,94 zł	**5,94 zł**
Uchwyty szklanych bezpieczników montowane w obudowie 5x20mm	5	1	5,89 zł	1,18 zł	**1,18 zł**
Przewód połączeniowy 26WG różne kolory	12	3	33,00 zł	2,75 zł	**8,25 zł**
Oudowa projektu (własny druk 3D) z materiału PLA	1000	97,88	50,00 zł	0,05 zł	**4,89 zł**
				SUMA DLA URZĄDZENIA:	**121,85 zł**
		SUMA DLA ZAMÓWIENIA			**249,89 zł**

POZOSTAŁE ELEMENTY (UŻYWANE PRZY PROTOTYPIE)					
Element	**Ilość (w zestawie)**	**Ilość (w urządzeniu)**	**Cena (zestawu)**	**Cena (za sztukę)**	**SUMA**
Płytka stykowa uniwersalna prototypowa MB-102 830pin	1	3	6,75 zł	6,75 zł	**20,25 zł**
Zworki i inne przewody połączeniowe	100	100	20,00 zł	0,20 zł	**20,00 zł**
Zestaw przewodów do płytki stykowej 560 szt.	320	320	20,60 zł	0,06 zł	**20,60 zł**
Płytka złącza USB Typ C 3.1 Serial Basic	5	1	7,84 zł	1,57 zł	**1,57 zł**
				SUMA CAŁKOWITA:	**62,42 zł**
		SUMA DLA ZAMÓWIENIA			**55,19 zł**

Linki do ważniejszych elementów i narzędzi wymienionych w dokumencie

Nazwa elementu / narzędzia	Pełny adres URL
ESP32-WROOM-32	https://elektroweb.pl/pl/esp32/197-esp-32s-esp-wroom-32-wifibluetooth-esp32-nodemcu.html
Moduł GPS GY-NEO6M V2	https://elektroweb.pl/pl/moduly-gps/951-modul-gps-neo-6m-gy-neo6mv2-z-antena.html
Wyświetlacz LCD 2.8" ST7789	https://elektroweb.pl/pl/wyswietlacze-lcd/1285-wyswietlacz-lcd-28-240x320-st7789-spi-dotykowy-slot-microsd.html
Interfejs Vgate iCar2	https://vgate.pl
Bambu Lab A1 mini	https://eu.store.bambulab.com/pl/products/a1-mini
LTspice	https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html
KiCad EDA	https://www.kicad.org/
Shapr3D	https://www.shapr3d.com/
Visual Studio Code	https://code.visualstudio.com/
PlatformIO	https://platformio.org/
Doxygen	https://www.doxygen.nl/

Załączniki do dokumentu:

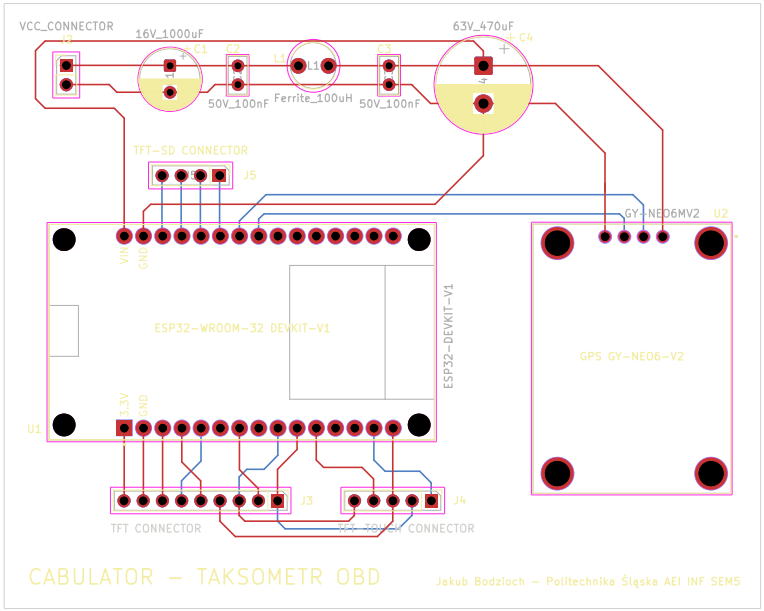


Jakub Bodzioch
INF
SEM 5
Politechnika Śląska AEI

Sheet: /
File: Taksometr.kicad_sch

Title: Cabulator_Taksometr_OBD

Size: A4	Date: 21.12.2025	**Rev: 1**
KiCad E.D.A. 9.0.6		Id: 1/1



Jakub Bodzioch INF SEM5 **Politechnika Śląska AEI**		
Sheet: File: Taksometr.kicad_pcb		
Title: Cabulator_Taksometr_OBD		
Size: A4	Date: 22.12.2025	**Rev: 1**
KiCad E.D.A. 9.0.6		Id: 1/1