Cabulator - Taxi Meter System

Generated by Doxygen 1.15.0

Chapter 1

Directory Hierarchy

1.1 Directories

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Directory Documentation

5.1 include Directory Reference

Files

- file background.h

  Obsługa tła aplikacji - rysowanie PNG na ekranie TFT.
- file gps_reader.h

  Obsługa modułu GPS - pobieranie pozycji i danych satelitarnych.
- file gui_elements.h

  Komponenty interfejsu użytkownika - elementy GUI dla TFT.
- file obd_reader.h

  Zminimalizowana implementacja czytnika OBDII - komunikacja z ELM327.
- file screen_about.h

  Ekran "About" - informacje o aplikacji.
- file screen_brightness.h

  Ustawienia jasności ekranu - kontrola podświetlenia TFT.
- file screen_connection.h

  Ekran połączeń - interfejs konfiguracji połączeń
- file screen_gps-debug.h

  Ekran diagnostyki GPS - szczegółowe dane satelitarne.
- file screen_gps.h

  Ekran GPS - interfejs zarządzania modułem GPS.
- file screen_home.h

  Główny ekran aplikacji - domyślny widok.
- file screen_manager.h

  Zarządzanie stanem ekranów aplikacji.
- file screen_obd-debug.h

  Ekran diagnostyki OBD - szczegółowe dane z pojazdu.
- file screen_obd.h

  Ekran OBD - interfejs zarządzania modułem OBD.
- file screen_settings.h

  Ekran ustawień aplikacji.
- file screen_tariff.h

  Ekran ustawień taryfy - konfiguracja rozliczeń
- file screen_trip.h

Ekran trasy - monitorowanie aktywnej podróży.

- file sd_manager.h

  Obsługa karty SD - logowanie tras i danych GPS.

- file tft_display.h

  Niskopoziomowa obsługa wyświetlacza TFT.

5.2 src Directory Reference

Files

- file background.cpp
- file gps_reader.cpp
- file gui_elements.cpp
- file main.cpp
- file obd_reader.cpp
- file screen_about.cpp
- file screen_brightness.cpp
- file screen_connection.cpp
- file screen_gps-debug.cpp
- file screen_gps.cpp
- file screen_home.cpp
- file screen_manager.cpp
- file screen_obd-debug.cpp
- file screen_obd.cpp
- file screen_settings.cpp
- file screen_tariff.cpp
- file screen_trip.cpp
- file sd_manager.cpp
- file tft_display.cpp

Chapter 6

Namespace Documentation

6.1 GPS Namespace Reference

Przestrzeń nazw zawierająca całą obsługę modułu GPS.

Classes

- struct Fix

  Struktura przechowująca dane z odczytu GPS (fix).

Functions

- void begin ()

  Inicjalizacja modułu GPS.
- bool poll (Fix &out)

  Pobiera najnowszą próbkę GPS.
- bool hasLiveFix ()

  Sprawdza, czy mamy aktualny fix GPS.
- void debugStatus ()

  Wypisuje status GPS na Serial (debug).
- bool setSystemTimeFromGPS (const Fix &fix)

  Zwraca surowe linie NMEA z modułu GPS.
- const char ∗const ∗ getLastRawLines ()
- static uint32_t msSince (uint32_t t0)

Variables

- Fix lastFix

  Ostatnia pobrana próbka GPS.
- static TinyGPSPlus gps
- static HardwareSerial ∗ port = &Serial2
- static uint32_t lastSample = 0
- static char rawLogLines [5][128] = {{0}}
- static uint8_t rawLinePos = 0
- static char currentRawLine [128] = {0}
- static uint8_t currentRawLinePos = 0

6.1.1 Detailed Description

Przestrzeń nazw zawierająca całą obsługę modułu GPS.

Zapewnia interfejs do komunikacji z modułem GPS przez UART, parsowanie danych NMEA oraz udostępnianie aktualnej pozycji.

6.1.2 Function Documentation

6.1.2.1 begin()

```
void GPS::begin ()
```

Inicjalizacja modułu GPS.

Konfiguruje UART i rozpoczyna nasłuchiwanie danych NMEA. Należy wywołać raz w setup().

Note

Piny i baudrate konfigurowane w cabulator_settings.h

Definition at line 29 of file gps_reader.cpp.

6.1.2.2 debugStatus()

```
void GPS::debugStatus ()
```

Wypisuje status GPS na Serial (debug).

Drukuje informacje o:

- Statusie połączenia
- Liczbie satelitów
- Ostatniej pozycji
- Wartości HDOP

Definition at line 186 of file gps_reader.cpp.

6.1.2.3 getLastRawLines()

```
const char *const * GPS::getLastRawLines ()
```

Definition at line 130 of file gps_reader.cpp.

6.1.2.4 hasLiveFix()

```
bool GPS::hasLiveFix ()
```

Sprawdza, czy mamy aktualny fix GPS.

Returns

true jeśli ostatni fix jest ważny i aktualny, false w przeciwnym razie

Przydatne do szybkiego sprawdzenia stanu GPS bez pobierania pełnych danych.

Definition at line 181 of file gps_reader.cpp.

6.1.2.5 msSince()

```
uint32_t GPS::msSince (
            uint32_t t0)  [inline], [static]
```

Definition at line 24 of file gps_reader.cpp.

6.1.2.6 poll()

```
bool GPS::poll (
            Fix & out)
```

Pobiera najnowszą próbkę GPS.

Parameters

out	*out*	Referencja do struktury Fix, która zostanie wypełniona danymi

Returns

true jeśli udało się pobrać nowe dane, false jeśli brak nowych danych

Warning

Sprawdź pole `out.valid` - może być false nawet przy return true

Definition at line 48 of file gps_reader.cpp.

6.1.2.7 setSystemTimeFromGPS()

```
bool GPS::setSystemTimeFromGPS (
            const Fix & fix)
```

Zwraca surowe linie NMEA z modułu GPS.

Returns

Wskaźnik do tablicy wskaźników na C-stringi z liniami NMEA

Przydatne do debugowania i analizy surowych danych z GPS.

Note

Dane są nadpisywane przy każdym wywołaniu poll()

Ustaw systemowy zegar ESP32 na podstawie danych GPS

Parameters

fix	Struktura Fix zawierająca dane daty/czasu z GPS

Returns

true jeśli udało się ustawić zegar, false jeśli data/czas były nieważne

Synchronizuje wewnętrzny zegar ESP32 z czasem z modułu GPS. Powinno być wywoływane gdy fix.dateTimeValid == true

Definition at line 138 of file gps_reader.cpp.

6.1.3 Variable Documentation

6.1.3.1 currentRawLine

```
char GPS::currentRawLine[128] = {0}  [static]
```

Definition at line 21 of file gps_reader.cpp.

6.1.3.2 currentRawLinePos

```
uint8_t GPS::currentRawLinePos = 0  [static]
```

Definition at line 22 of file gps_reader.cpp.

6.1.3.3 gps

```
TinyGPSPlus GPS::gps  [static]
```

Definition at line 13 of file gps_reader.cpp.

6.1.3.4 lastFix

```
Fix GPS::lastFix
```

Ostatnia pobrana próbka GPS.

Globalna zmienna przechowująca ostatni odczyt. Aktualizowana automatycznie przez poll().

See also

poll() do pobrania świeżych danych

Definition at line 16 of file gps_reader.cpp.

6.1.3.5 lastSample

```
uint32_t GPS::lastSample = 0  [static]
```

Definition at line 15 of file gps_reader.cpp.

6.1.3.6 port

```
HardwareSerial* GPS::port = &Serial2  [static]
```

Definition at line 14 of file gps_reader.cpp.

6.1.3.7 rawLinePos

```
uint8_t GPS::rawLinePos = 0  [static]
```

Definition at line 20 of file gps_reader.cpp.

6.1.3.8 rawLogLines

```
char GPS::rawLogLines[5][128] = {{0}}  [static]
```

Definition at line 19 of file gps_reader.cpp.

6.2 OBD Namespace Reference

Przestrzeń nazw zawierająca całą obsługę modułu OBDII.

Functions

- bool init ()

  Inicjalizuje i łączy się z modułem OBD przez Bluetooth.
- long readOdometer ()

  Odczytuje wartość odometru z pojazdu.
- float readFuelRate ()

  Odczytuje bieżące zużycie paliwa na podstawie MAF.
- float calculateCost ()

  Oblicza koszt przejazdu na podstawie spalania i taryfy.
- void task (void *param)

  Task FreeRTOS obsługujący komunikację OBD w tle.
- bool sendCmd (const char *cmd, char *response, int maxLen, int timeout=1000)

Variables

- bool btConnected = false

  Status połączenia Bluetooth z modułem OBD.
- bool elmReady = false

  Status gotowości modułu ELM327.
- BluetoothSerial SerialBT

6.2.1 Detailed Description

Przestrzeń nazw zawierająca całą obsługę modułu OBDII.

6.2.2 Function Documentation

6.2.2.1 calculateCost()

```
float OBD::calculateCost ()
```

Oblicza koszt przejazdu na podstawie spalania i taryfy.

Wykorzystuje bieżące spalanie i aktywny tryb taryfy do wyliczenia kosztu na jednostkę czasu.

Returns

Koszt w jednostce walutowej na godzinę

See also

readFuelRate() dla źródła danych spalania

Definition at line 208 of file obd_reader.cpp.

6.2.2.2 init()

```
bool OBD::init ()
```

Inicjalizuje i łączy się z modułem OBD przez Bluetooth.

Konfiguruje połączenie Bluetooth, wysyła komendy inicjalizacyjne AT i sprawdza komunikację z pojazdem.

Returns

true jeśli inicjalizacja zakończyła się sukcesem, false w przypadku błędu

Note

Wymaga wcześniejszej konfiguracji adresu MAC w cabulator_settings.h

Definition at line 53 of file obd_reader.cpp.

6.2.2.3 readFuelRate()

```
float OBD::readFuelRate ()
```

Odczytuje bieżące zużycie paliwa na podstawie MAF.

Oblicza spalanie wykorzystując przepływ powietrza (MAF - Mass Air Flow).

Returns

Spalanie w litrach na godzinę [L/h], lub -1 przy błędzie

Note

Dokładność zależy od kalibracji współczynnika AFR

Definition at line 167 of file obd_reader.cpp.

6.2.2.4 readOdometer()

```
long OBD::readOdometer ()
```

Odczytuje wartość odometru z pojazdu.

Wysyła zapytanie PID do ECU i parsuje odpowiedź.

Returns

Wartość odometru w kilometrach, lub -1 przy błędzie odczytu

Warning

Nie wszystkie pojazdy obsługują ten PID

Definition at line 123 of file obd_reader.cpp.

6.2.2.5 sendCmd()

```
bool OBD::sendCmd (
            const char * cmd,
            char * response,
            int maxLen,
            int timeout = 1000)
```

Definition at line 20 of file obd_reader.cpp.

6.2.2.6 task()

```
void OBD::task (
            void * param)
```

Task FreeRTOS obsługujący komunikację OBD w tle.

Cyklicznie odpytuje ECU o dane i aktualizuje zmienne globalne. Powinien być uruchomiony przez xTaskCreate().

Parameters

param	Parametr przekazywany do tasku (nieużywany)

Note

Task działa w nieskończonej pętli z delay

Definition at line 218 of file obd_reader.cpp.

6.2.3 Variable Documentation

6.2.3.1 btConnected

```
bool OBD::btConnected = false
```

Status połączenia Bluetooth z modułem OBD.

true jeśli połączenie Bluetooth jest aktywne

Definition at line 16 of file obd_reader.cpp.

6.2.3.2 elmReady

```
bool OBD::elmReady = false
```

Status gotowości modułu ELM327.

true jeśli moduł odpowiada na komendy AT i jest gotowy do pracy

Definition at line 17 of file obd_reader.cpp.

6.2.3.3 SerialBT

```
BluetoothSerial OBD::SerialBT
```

Definition at line 13 of file obd_reader.cpp.

6.3 SDManager Namespace Reference

Classes

- struct TripData

  Struktura danych trasy do zapisu.
- struct GPSData

  Struktura danych GPS do zapisu.

Functions

- bool init ()
  Inicjalizuje kartę SD na HSPI (Secondary SPI).
- bool isReady ()
  Sprawdza czy karta SD jest gotowa.
- String createTripSession ()
  Tworzy nową sesję trasy z timestamp'em.
- bool saveTripData (const String &tripPath, const TripData &data)
  Zapisuje dane trasy do pliku trip_data.csv.
- bool saveGPSData (const String &tripPath, const GPSData &data)
  Zapisuje wpis danych GPS do pliku gps_log.csv.
- void onGPSFix (const GPSData &data)
  Callback wywoływany przez GPS gdy pojawi się nowy fix.
- void finalizeTrip (const TripData &data)
  Finalizuje trasę - zapisuje ostateczne dane podsumowania na SD.
- void listTrips ()
  Listuje wszystkie dostępne logi tras.
- bool getLastTrip (String &tripPath)
  Pobiera dane z ostatniej trasy.
- static String getTimestamp ()
- static String getDateOnly ()

Variables

- static bool sdReady = false
- static String currentTripPath = ""

6.3.1 Function Documentation

6.3.1.1 createTripSession()

```
String SDManager::createTripSession ()
```

Tworzy nową sesję trasy z timestamp'em.

Tworzy folder w formacie "YYYY-MM-DD_HH-MM-SS" w katalogu /logs/trips/

Returns

Ścieżka utworzonego folderu, lub pusty string jeśli błąd

Definition at line 73 of file sd_manager.cpp.

6.3.1.2 finalizeTrip()

```
void SDManager::finalizeTrip (
            const TripData & data)
```

Finalizuje trasę - zapisuje ostateczne dane podsumowania na SD.

Ta funkcja powinna być wywoływana gdy użytkownik kończy trasę (np. wciśnie przycisk "Stop" w screen_trip.cpp).

Funkcja zapisuje dane końcowe trasy (distanceKm, fuelUsedLiters, cost, etc.) do pliku trip_data.csv w folderze bieżącej trasy.

Po zapisaniu, zmienną globalną currentTripPath powinna być wyczyszczona.

Parameters

data	Struktura TripData zawierająca podsumowanie przejazdu

Note

Po wyeliminowaniu tej funkcji trip jest "zamknięty" na SD

See also

screen_trip.cpp - tam gdzie jest wywoływana

TripData - struktura danych trasy

Definition at line 247 of file sd_manager.cpp.

6.3.1.3 getDateOnly()

```
String SDManager::getDateOnly ()  [static]
```

Definition at line 35 of file sd_manager.cpp.

6.3.1.4 getLastTrip()

```
bool SDManager::getLastTrip (
            String & tripPath)
```

Pobiera dane z ostatniej trasy.

Parameters

tripPath[out]	Ścieżka do folderu ostatniej trasy

Returns

true jeśli znaleziono trasę, false jeśli brak tras

Definition at line 205 of file sd_manager.cpp.

6.3.1.5 getTimestamp()

```
String SDManager::getTimestamp ()  [static]
```

Definition at line 25 of file sd_manager.cpp.

6.3.1.6 init()

```
bool SDManager::init ()
```

Inicjalizuje kartę SD na HSPI (Secondary SPI).

Funkcja konfiguruje HSPI bus (Secondary SPI)

Używanie osobnego bus (HSPI) zamiast domyślnego SPI (VSP) pozwala karcie SD pracować równolegle z TFT (wyświetlacz) i touch inputem, bez konfliktów na linii komunikacyjnej.

Funkcja tworzy również strukturę katalogów /logs/trips/ na karcie SD.

Returns

true jeśli SD została poprawnie zainicjalizowana i zmontowana, false w przypadku błędu (np. brak karty, błąd zapisu)

Note

Powinna być wywołana w setup() po inicjalizacji TFT, aby upewnić się że touch będzie poprawnie skalibrowany po incjalizacji SD.

See also

SDManager::onGPSFix(), SDManager::finalizeTrip()

Definition at line 44 of file sd_manager.cpp.

6.3.1.7 isReady()

```
bool SDManager::isReady ()
```

Sprawdza czy karta SD jest gotowa.

Returns

true jeśli karta SD jest dostępna

Definition at line 69 of file sd_manager.cpp.

6.3.1.8 listTrips()

```
void SDManager::listTrips ()
```

Listuje wszystkie dostępne logi tras.

Wypisuje na Serial listę folderów z trasami.

Definition at line 171 of file sd_manager.cpp.

6.3.1.9 onGPSFix()

```
void SDManager::onGPSFix (
            const GPSData & data)
```

Callback wywoływany przez GPS gdy pojawi się nowy fix.

Ta funkcja jest automatycznie wywoływana przez moduł GPS (z gps_reader.cpp) każdorazowo gdy nowy fix jest dostępny (∼co 1 sekundę podczas normalnej pracy).

Jeśli trip jest aktywny (currentTripPath != ""), dane GPS są zapisywane do pliku gps_log.csv w folderze bieżącej trasy.

Zapis odbywa się na HSPI bus, więc nie blokuje operacji na TFT/touch.

Parameters

data	Struktura GPSData zawierająca: szerokość, długość, satelity, HDOP, valid, timestamp

Note

Ta funkcja jest non-blocking i powinna być szybka (wpis do CSV)

See also

gps_reader.cpp - tam gdzie jest wywoływana
GPSData - struktura danych GPS

Definition at line 239 of file sd_manager.cpp.

6.3.1.10 saveGPSData()

```
bool SDManager::saveGPSData (
            const String & tripPath,
            const GPSData & data)
```

Zapisuje wpis danych GPS do pliku gps_log.csv.

Parameters

tripPath	Ścieżka folderu trasy
data	Struktura z danymi GPS

Returns

true jeśli zapis się powiódł, false w przypadku błędu

Definition at line 141 of file sd_manager.cpp.

6.3.1.11 saveTripData()

```
bool SDManager::saveTripData (
            const String & tripPath,
            const TripData & data)
```

Zapisuje dane trasy do pliku trip_data.csv.

Parameters

tripPath	Ścieżka folderu trasy (zwrócona z createTripSession)
data	Struktura z danymi trasy

Returns

true jeśli zapis się powiódł, false w przypadku błędu

Definition at line 111 of file sd_manager.cpp.

6.3.2 Variable Documentation

6.3.2.1 currentTripPath

```
String SDManager::currentTripPath = ""  [static]
```

Definition at line 22 of file sd_manager.cpp.

6.3.2.2 sdReady

```
bool SDManager::sdReady = false  [static]
```

Definition at line 21 of file sd_manager.cpp.

Chapter 7

Class Documentation

7.1 Background Class Reference

Klasa do zarządzania tłem ekranu TFT.

`#include <background.h>`

Public Member Functions

- Background (const String &path)
  Konstruktor klasy Background.
- void setPath (const String &path)
  Ustawia nową ścieżkę do pliku PNG.
- String path () const
  Zwraca aktualną ścieżkę do pliku PNG.
- bool draw (TFT_eSPI &tft, PNG &png, bool center=true)
  Rysuje tło na ekranie TFT.

Static Public Member Functions

- static void listFS (const char *dir="/")
  Listuje pliki w systemie plików LittleFS.
- static int pngDraw (PNGDRAW *p)
  Callback wywoływany podczas dekodowania każdej linii PNG.
- static bool drawPngFullScreen (const char *path, bool center)
  Rysuje pełnoekranowy obraz PNG z podanej ścieżki.

Callbacki dla dekodera PNG

- static void * pngOpen (const char *filename, int32_t *pFileSize)
  Callback otwierający plik PNG.
- static void pngClose (void *handle)
  Callback zamykający plik PNG.
- static int32_t pngRead (PNGFILE *pFile, uint8_t *pBuff, int32_t iLen)
  Callback czytający dane z pliku PNG.
- static int32_t pngSeek (PNGFILE *pFile, int32_t iPosition)
  Callback ustawiający pozycję w pliku PNG.

Static Public Attributes

Statyczne zmienne konfiguracyjne

- static int s_offX = 0
  Offset poziomy przy rysowaniu PNG.
- static int s_offY = 0
  Offset pionowy przy rysowaniu PNG.
- static uint16_t s_lineBuf [320] = {0}
  Bufor linii do rysowania PNG (szerokość ekranu).
- static File s_pngFile
  Uchwyt do otwartego pliku PNG.
- static TFT_eSPI * s_tft = nullptr
  Wskaźnik do obiektu wyświetlacza TFT.
- static PNG * s_png = nullptr
  Wskaźnik do dekodera PNG.

7.1.1 Detailed Description

Klasa do zarządzania tłem ekranu TFT.

Umożliwia ładowanie i wyświetlanie obrazów PNG jako tła aplikacji. Obsługuje system plików LittleFS oraz dekodowanie PNG w locie.

Note

Wymaga zainicjalizowanego LittleFS przed użyciem

Definition at line 35 of file background.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Background()

```
Background::Background (
            const String & path)  [explicit]
```

Konstruktor klasy Background.

Parameters

path	Ścieżka do pliku PNG w systemie plików LittleFS

Definition at line 12 of file background.cpp.

7.1.3 Member Function Documentation

7.1.3.1 draw()

```
bool Background::draw (
            TFT_eSPI & tft,
            PNG & png,
            bool center = true)
```

Rysuje tło na ekranie TFT.

Parameters

tft	Referencja do obiektu wyświetlacza TFT
png	Referencja do obiektu dekodera PNG
center	Czy wyśrodkować obraz na ekranie (domyślnie true)

Returns

true jeśli rysowanie się powiodło, false w przypadku błędu

Definition at line 87 of file background.cpp.

7.1.3.2 drawPngFullScreen()

```
bool Background::drawPngFullScreen (
            const char * path,
            bool center)  [static]
```

Rysuje pełnoekranowy obraz PNG z podanej ścieżki.

Parameters

path	Ścieżka do pliku PNG
center	Czy wyśrodkować obraz

Returns

true jeśli sukces, false w przypadku błędu

Note

Funkcja statyczna - nie wymaga instancji klasy

Definition at line 60 of file background.cpp.

7.1.3.3 listFS()

```
void Background::listFS (
            const char * dir = "/")  [static]
```

Listuje pliki w systemie plików LittleFS.

Parameters

dir	Katalog do listowania (domyślnie "/")

Wypisuje listę plików na Serial w celach debugowania.

Definition at line 97 of file background.cpp.

7.1.3.4 path()

```
String Background::path () const
```

Zwraca aktualną ścieżkę do pliku PNG.

Returns

Ścieżka do pliku PNG jako String

Definition at line 14 of file background.cpp.

7.1.3.5 pngClose()

```
void Background::pngClose (
            void * handle)  [static]
```

Callback zamykający plik PNG.

Parameters

handle	Uchwyt do pliku zwrócony przez pngOpen()

Definition at line 27 of file background.cpp.

7.1.3.6 pngDraw()

```
int Background::pngDraw (
            PNGDRAW * p)  [static]
```

Callback wywoływany podczas dekodowania każdej linii PNG.

Parameters

p	Wskaźnik do struktury PNGDRAW z danymi linii

Returns

1 jeśli sukces, 0 w przypadku błędu

Funkcja renderuje zdekodowaną linię bezpośrednio na wyświetlacz TFT.

Definition at line 50 of file background.cpp.

7.1.3.7 pngOpen()

```
void * Background::pngOpen (
            const char * filename,
            int32_t * pFileSize)  [static]
```

Callback otwierający plik PNG.

Parameters

filename	Nazwa pliku do otwarcia
pFileSize[out]	Wskaźnik do zmiennej przechowującej rozmiar pliku

Returns

Uchwyt do pliku lub nullptr w przypadku błędu

Definition at line 17 of file background.cpp.

7.1.3.8 pngRead()

```
int32_t Background::pngRead (
            PNGFILE * pFile,
            uint8_t * pBuff,
            int32_t iLen)  [static]
```

Callback czytający dane z pliku PNG.

Parameters

pFile	Wskaźnik do struktury PNGFILE
pBuff	Bufor docelowy na dane
iLen	Liczba bajtów do odczytania

Returns

Liczba faktycznie odczytanych bajtów

Definition at line 35 of file background.cpp.

7.1.3.9 pngSeek()

```
int32_t Background::pngSeek (
            PNGFILE * pFile,
            int32_t iPosition)  [static]
```

Callback ustawiający pozycję w pliku PNG.

Parameters

pFile	Wskaźnik do struktury PNGFILE
iPosition	Nowa pozycja w pliku

Returns

Nowa pozycja lub kod błędu

Definition at line 42 of file background.cpp.

7.1.3.10 setPath()

```
void Background::setPath (
            const String & path)
```

Ustawia nową ścieżkę do pliku PNG.

Parameters

path	Nowa ścieżka do pliku PNG

Definition at line 13 of file background.cpp.

7.1.4 Member Data Documentation

7.1.4.1 s_lineBuf

```
uint16_t Background::s_lineBuf = {0}  [static]
```

Bufor linii do rysowania PNG (szerokość ekranu).

Definition at line 6 of file background.h.

7.1.4.2 s_offX

```
int Background::s_offX = 0  [static]
```

Offset poziomy przy rysowaniu PNG.

Definition at line 41 of file background.h.

7.1.4.3 s_offY

```
int Background::s_offY = 0  [static]
```

Offset pionowy przy rysowaniu PNG.

Definition at line 42 of file background.h.

7.1.4.4 s_png

```
PNG * Background::s_png = nullptr  [static]
```

Wskaźnik do dekodera PNG.

Definition at line 46 of file background.h.

7.1.4.5 s_pngFile

```
File Background::s_pngFile  [static]
```

Uchwyt do otwartego pliku PNG.

Definition at line 44 of file background.h.

7.1.4.6 s_tft

```
TFT_eSPI * Background::s_tft = nullptr  [static]
```

Wskaźnik do obiektu wyświetlacza TFT.

Definition at line 45 of file background.h.

The documentation for this class was generated from the following files:

- include/background.h
- src/background.cpp

7.2 GPS::Fix Struct Reference

Struktura przechowująca dane z odczytu GPS (fix).

```
#include <gps_reader.h>
```

Public Attributes

- bool valid
  Czy pozycja jest ważna (fix uzyskany).
- uint32_t takenAtMs
  Timestamp pobrania próbki [ms od startu].
- uint8_t sats
  Liczba widocznych satelitów.
- uint16_t hdop
  Precyzja pozioma HDOP (x100, np. 120 = 1.20).
- double lat
  Szerokość geograficzna [stopnie].
- double lng
  Długość geograficzna [stopnie].
- uint16_t year
  Rok (np. 2025).
- uint8_t month
  Miesiąc (1-12).
- uint8_t day
  Dzień (1-31).
- uint8_t hour
  Godzina (0-23).
- uint8_t minute
  Minuta (0-59).
- uint8_t second
  Sekunda (0-59).
- bool dateTimeValid
  Czy data/czas są ważne.

7.2.1 Detailed Description

Struktura przechowująca dane z odczytu GPS (fix).

Zawiera wszystkie istotne informacje o aktualnej pozycji oraz metadane o jakości sygnału.

Note

Pole `valid` należy sprawdzić przed użyciem współrzędnych

Definition at line 38 of file gps_reader.h.

7.2.2 Member Data Documentation

7.2.2.1 dateTimeValid

```
bool GPS::Fix::dateTimeValid
```

Czy data/czas są ważne.

Definition at line 52 of file gps_reader.h.

7.2.2.2 day

```
uint8_t GPS::Fix::day
```

Dzień (1-31).

Definition at line 48 of file gps_reader.h.

7.2.2.3 hdop

```
uint16_t GPS::Fix::hdop
```

Precyzja pozioma HDOP (x100, np. 120 = 1.20).

Definition at line 42 of file gps_reader.h.

7.2.2.4 hour

```
uint8_t GPS::Fix::hour
```

Godzina (0-23).

Definition at line 49 of file gps_reader.h.

7.2.2.5 lat

```
double GPS::Fix::lat
```

Szerokość geograficzna [stopnie].

Definition at line 43 of file gps_reader.h.

7.2.2.6 lng

```
double GPS::Fix::lng
```

Długość geograficzna [stopnie].

Definition at line 44 of file gps_reader.h.

7.2.2.7 minute

```
uint8_t GPS::Fix::minute
```

Minuta (0-59).

Definition at line 50 of file gps_reader.h.

7.2.2.8 month

```
uint8_t GPS::Fix::month
```

Miesiąc (1-12).

Definition at line 47 of file gps_reader.h.

7.2.2.9 sats

```
uint8_t GPS::Fix::sats
```

Liczba widocznych satelitów.

Definition at line 41 of file gps_reader.h.

7.2.2.10 second

```
uint8_t GPS::Fix::second
```

Sekunda (0-59).

Definition at line 51 of file gps_reader.h.

7.2.2.11 takenAtMs

```
uint32_t GPS::Fix::takenAtMs
```

Timestamp pobrania próbki [ms od startu].

Definition at line 40 of file gps_reader.h.

7.2.2.12 valid

```
bool GPS::Fix::valid
```

Czy pozycja jest ważna (fix uzyskany).

Definition at line 39 of file gps_reader.h.

7.2.2.13 year

```
uint16_t GPS::Fix::year
```

Rok (np. 2025).

Definition at line 46 of file gps_reader.h.

The documentation for this struct was generated from the following file:

- include/gps_reader.h

7.3 SDManager::GPSData Struct Reference

Struktura danych GPS do zapisu.

```
#include <sd_manager.h>
```

Public Attributes

- double latitude
  Szerokość geograficzna.
- double longitude
  Długość geograficzna.
- uint8_t satellites
  Liczba satelitów.
- uint16_t hdop
  Precyzja HDOP.
- bool valid
  Czy fix jest ważny.
- unsigned long timestamp
  Timestamp w ms od startu.

7.3.1 Detailed Description

Struktura danych GPS do zapisu.

Definition at line 71 of file sd_manager.h.

7.3.2 Member Data Documentation

7.3.2.1 hdop

```
uint16_t SDManager::GPSData::hdop
```

Precyzja HDOP.

Definition at line 75 of file sd_manager.h.

7.3.2.2 latitude

```
double SDManager::GPSData::latitude
```

Szerokość geograficzna.

Definition at line 72 of file sd_manager.h.

7.3.2.3 longitude

```
double SDManager::GPSData::longitude
```

Długość geograficzna.

Definition at line 73 of file sd_manager.h.

7.3.2.4 satellites

```
uint8_t SDManager::GPSData::satellites
```

Liczba satelitów.

Definition at line 74 of file sd_manager.h.

7.3.2.5 timestamp

```
unsigned long SDManager::GPSData::timestamp
```

Timestamp w ms od startu.

Definition at line 77 of file sd_manager.h.

7.3.2.6 valid

```
bool SDManager::GPSData::valid
```

Czy fix jest ważny.

Definition at line 76 of file sd_manager.h.

The documentation for this struct was generated from the following file:

- include/sd_manager.h

7.4 TextStyle Struct Reference

Struktura definiująca styl wyświetlania tekstu.

```
#include <gui_elements.h>
```

Public Attributes

- uint8_t font
  Numer czcionki TFT_eSPI (1-8 + custom).
- uint16_t color
  Kolor tekstu w formacie RGB565.
- uint8_t datum
  Punkt wyrównania tekstu (TL_DATUM, MC_DATUM, BR_DATUM, etc.).

7.4.1 Detailed Description

Struktura definiująca styl wyświetlania tekstu.

Używana do konfiguracji wyglądu tekstu w aplikacji. Pozwala na jednolite zarządzanie stylami w różnych ekranach.

Definition at line 24 of file gui_elements.h.

7.4.2 Member Data Documentation

7.4.2.1 color

```
uint16_t TextStyle::color
```

Kolor tekstu w formacie RGB565.

Definition at line 26 of file gui_elements.h.

7.4.2.2 datum

```
uint8_t TextStyle::datum
```

Punkt wyrównania tekstu (TL_DATUM, MC_DATUM, BR_DATUM, etc.).

Definition at line 27 of file gui_elements.h.

7.4.2.3 font

```
uint8_t TextStyle::font
```

Numer czcionki TFT_eSPI (1-8 + custom).

Definition at line 25 of file gui_elements.h.

The documentation for this struct was generated from the following file:

- include/gui_elements.h

7.5 SDManager::TripData Struct Reference

Struktura danych trasy do zapisu.

```
#include <sd_manager.h>
```

Public Attributes

- float distanceKm
  Przejechany dystans w km.
- float fuelUsedLiters
  Zużyte paliwo w litrach.
- int tariffMode
  Tryb taryfy (0=km, 1=paliwo).
- float tariffValue
  Wartość taryfy.
- float totalCost
  Całkowity koszt.

7.5.1 Detailed Description

Struktura danych trasy do zapisu.

Definition at line 59 of file sd_manager.h.

7.5.2 Member Data Documentation

7.5.2.1 distanceKm

```
float SDManager::TripData::distanceKm
```

Przejechany dystans w km.

Definition at line 60 of file sd_manager.h.

7.5.2.2 fuelUsedLiters

```
float SDManager::TripData::fuelUsedLiters
```

Zużyte paliwo w litrach.

Definition at line 61 of file sd_manager.h.

7.5.2.3 tariffMode

```
int SDManager::TripData::tariffMode
```

Tryb taryfy (0=km, 1=paliwo).

Definition at line 62 of file sd_manager.h.

7.5.2.4 tariffValue

```
float SDManager::TripData::tariffValue
```

Wartość taryfy.

Definition at line 63 of file sd_manager.h.

7.5.2.5 totalCost

```
float SDManager::TripData::totalCost
```

Całkowity koszt.

Definition at line 64 of file sd_manager.h.

The documentation for this struct was generated from the following file:

- include/sd_manager.h

Chapter 8

File Documentation

8.1 include/background.h File Reference

Obsługa tła aplikacji - rysowanie PNG na ekranie TFT.

```
#include <Arduino.h>
#include <FS.h>
#include <PNGdec.h>
#include <LittleFS.h>
#include <TFT_eSPI.h>
```

Classes

- class Background

  Klasa do zarządzania tłem ekranu TFT.

8.1.1 Detailed Description

Obsługa tła aplikacji - rysowanie PNG na ekranie TFT.

Version

1.0

Date

2025-01-20

Plik zawiera klasę i funkcje do obsługi tła ekranu. Pozwala na rysowanie tła z pliku PNG, zarządzanie ścieżką pliku oraz obsługę systemu plików LittleFS.

Definition in file background.h.

8.2 background.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef BACKGROUND_H
00013 #define BACKGROUND_H
00014
00015 #include <Arduino.h>
00016 #include <FS.h>
00017 #include <PNGdec.h>
00018 #include <LittleFS.h>
00019 #include <TFT_eSPI.h>
00020
00021 using namespace fs;
00022
00023 // Forward declaration
00024 class TFT_eSPI;
00025
00035 class Background {
00036
00037 public:
00038
00041     static int s_offX;
00042     static int s_offY;
00043     static uint16_t s_lineBuf[320];
00044     static File s_pngFile;
00045     static TFT_eSPI *s_tft;
00046     static PNG *s_png;
00048
00053     explicit Background(const String &path);
00054
00059     void setPath(const String &path);
00060
00065     String path() const;
00066
00074     bool draw(TFT_eSPI &tft, PNG &png, bool center = true);
00075
00082     static void listFS(const char *dir = "/");
00083
00086
00093     static void *pngOpen(const char *filename, int32_t *pFileSize);
00094
00099     static void pngClose(void *handle);
00100
00108     static int32_t pngRead(PNGFILE *pFile, uint8_t *pBuff, int32_t iLen);
00109
00116     static int32_t pngSeek(PNGFILE *pFile, int32_t iPosition);
00117
00119
00127     static int pngDraw(PNGDRAW *p);
00128
00137     static bool drawPngFullScreen(const char *path, bool center);
00138
00139 private:
00140
00141     String _path;
00142 };
00143
00144 #endif // BACKGROUND_H
```

8.3 include/gps_reader.h File Reference

Obsługa modułu GPS - pobieranie pozycji i danych satelitarnych.

```
#include <Arduino.h>
#include "../cabulator_settings.h"
```

Classes

- struct GPS::Fix

  Struktura przechowująca dane z odczytu GPS (fix).

Namespaces

- namespace GPS

  Przestrzeń nazw zawierająca całą obsługę modułu GPS.

Functions

- void GPS::begin ()

  Inicjalizacja modułu GPS.
- bool GPS::poll (Fix &out)

  Pobiera najnowszą próbkę GPS.
- bool GPS::hasLiveFix ()

  Sprawdza, czy mamy aktualny fix GPS.
- void GPS::debugStatus ()

  Wypisuje status GPS na Serial (debug).
- bool GPS::setSystemTimeFromGPS (const Fix &fix)

  Zwraca surowe linie NMEA z modułu GPS.
- const char ∗const ∗ GPS::getLastRawLines ()

Variables

- Fix GPS::lastFix

  Ostatnia pobrana próbka GPS.

8.3.1 Detailed Description

Obsługa modułu GPS - pobieranie pozycji i danych satelitarnych.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje i struktury do obsługi GPS. Pozwala na inicjalizację, pobieranie danych, sprawdzanie statusu oraz debugowanie połączenia z GPS.

See also

cabulator_settings.h Konfiguracja pinów i parametrów GPS

Definition in file gps_reader.h.

8.4 gps_reader.h

Go to the documentation of this file.

```
00001
00013
00014 #ifndef GPS_READER_H
00015 #define GPS_READER_H
00016
00017 #include <Arduino.h>
00018 #include "../cabulator_settings.h"
00019
00027 namespace GPS {
00028
00038     struct Fix {
00039         bool valid;
00040         uint32_t takenAtMs;
00041         uint8_t sats;
00042         uint16_t hdop;
00043         double lat;
00044         double lng;
00045
00046         uint16_t year;
00047         uint8_t month;
00048         uint8_t day;
00049         uint8_t hour;
00050         uint8_t minute;
00051         uint8_t second;
00052         bool dateTimeValid;
00053     };
00054
00063     void begin();
00064
00072     bool poll(Fix& out);
00073
00080     bool hasLiveFix();
00081
00091     void debugStatus();
00092
00109     bool setSystemTimeFromGPS(const Fix& fix);
00110
00111     const char* const* getLastRawLines();
00112
00121     extern Fix lastFix;
00122
00123 }
00124
00125 #endif // GPS_READER_H
```

8.5 include/gui_elements.h File Reference

Komponenty interfejsu użytkownika - elementy GUI dla TFT.

```
#include <TFT_eSPI.h>
```

Classes

- struct TextStyle

  Struktura definiująca styl wyświetlania tekstu.

Functions

- void drawText (TFT_eSPI *tft, const char *text, int x, int y, const TextStyle &style, uint16_t overrideColor=0)

  Rysuje tekst na ekranie TFT z określonym stylem.
- void drawTextWithBackground (TFT_eSPI *tft, const char *text, int x, int y, uint8_t datum, uint8_t font, uint16_t textColor, uint16_t bgColor, int16_t paddingWidth=0)

  Rysuje tekst z kolorowym tłem.
- void resetTextBuffer (char *buf, size_t size)

  Resetuje bufor tekstowy wypełniając go zerami.

8.5.1 Detailed Description

Komponenty interfejsu użytkownika - elementy GUI dla TFT.

Version

1.0

Date

2025-01-20

Plik zawiera struktury i funkcje do tworzenia elementów GUI. Definiuje style przycisków, tekstu oraz uniwersalne funkcje rysowania używane we wszystkich ekranach aplikacji.

Definition in file gui_elements.h.

8.5.2 Function Documentation

8.5.2.1 drawText()

```
void drawText (
            TFT_eSPI * tft,
            const char * text,
            int x,
            int y,
            const TextStyle & style,
            uint16_t overrideColor = 0)
```

Rysuje tekst na ekranie TFT z określonym stylem.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT
text	Tekst do wyświetlenia (C-string)
x	Współrzędna X pozycji tekstu
y	Współrzędna Y pozycji tekstu
style	Referencja do struktury TextStyle definiującej styl
overrideColor	Opcjonalny kolor nadpisujący styl (0 = użyj koloru ze stylu)

Note

Punkt odniesienia tekstu (datum) określa jak x,y interpretowane są względem tekstu

See also

TextStyle

Definition at line 8 of file gui_elements.cpp.

8.5.2.2 drawTextWithBackground()

```
void drawTextWithBackground (
            TFT_eSPI * tft,
            const char * text,
            int x,
            int y,
            uint8_t datum,
            uint8_t font,
            uint16_t textColor,
            uint16_t bgColor,
            int16_t paddingWidth = 0)
```

Rysuje tekst z kolorowym tłem.

Renderuje prostokątne tło pod tekstem, przydatne do tworzenia etykiet, przycisków lub wyróżnionych napisów.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT
text	Tekst do wyświetlenia
x	Współrzędna X pozycji tekstu
y	Współrzędna Y pozycji tekstu
datum	Punkt odniesienia tekstu (np. MC_DATUM dla środka)
font	Numer czcionki TFT_eSPI
textColor	Kolor tekstu (RGB565)
bgColor	Kolor tła (RGB565)
paddingWidth	Dodatkowa szerokość tła po bokach [px] (domyślnie 0)

Warning

Tło jest rysowane przed tekstem, może nadpisać istniejącą grafikę

Note

PaddingWidth pozwala na dodanie przestrzeni po bokach tekstu w tle

Definition at line 18 of file gui_elements.cpp.

8.5.2.3 resetTextBuffer()

```
void resetTextBuffer (
            char * buf,
            size_t size)  [inline]
```

Resetuje bufor tekstowy wypełniając go zerami.

Funkcja pomocnicza do czyszczenia buforów char[] przed formatowaniem nowego tekstu.

Parameters

buf	Wskaźnik do bufora do wyczyszczenia
size	Rozmiar bufora w bajtach

Note

Funkcja inline dla wydajności

Definition at line 87 of file gui_elements.h.

8.6 gui_elements.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef GUI_ELEMENTS_H
00013 #define GUI_ELEMENTS_H
00014
00015 #include <TFT_eSPI.h>
00016
00024 struct TextStyle {
00025     uint8_t  font;
00026     uint16_t color;
00027     uint8_t  datum;
00028 };
00029
00044 void drawText(TFT_eSPI* tft, const char* text, int x, int y, const TextStyle& style, uint16_t
      overrideColor = 0);
00045
00065 void drawTextWithBackground(
00066     TFT_eSPI* tft,
00067     const char* text,
00068     int x, int y,
00069     uint8_t datum,
00070     uint8_t font,
00071     uint16_t textColor,
00072     uint16_t bgColor,
00073     int16_t paddingWidth = 0
00074 );
00075
00087 inline void resetTextBuffer(char* buf, size_t size) {
00088     memset(buf, 0, size);
00089 }
00090
00091 #endif // GUI_ELEMENTS_H
```

8.7 include/obd_reader.h File Reference

Zminimalizowana implementacja czytnika OBDII - komunikacja z ELM327.

```
#include <Arduino.h>
#include "../cabulator_settings.h"
```

Namespaces

- namespace OBD

  Przestrzeń nazw zawierająca całą obsługę modułu OBDII.

Functions

- bool OBD::init ()

  Inicjalizuje i łączy się z modułem OBD przez Bluetooth.
- long OBD::readOdometer ()

  Odczytuje wartość odometru z pojazdu.
- float OBD::readFuelRate ()

  Odczytuje bieżące zużycie paliwa na podstawie MAF.
- float OBD::calculateCost ()

  Oblicza koszt przejazdu na podstawie spalania i taryfy.
- void OBD::task (void *param)

  Task FreeRTOS obsługujący komunikację OBD w tle.

Variables

- bool OBD::btConnected = false

  Status połączenia Bluetooth z modułem OBD.
- bool OBD::elmReady = false

  Status gotowości modułu ELM327.

8.7.1 Detailed Description

Zminimalizowana implementacja czytnika OBDII - komunikacja z ELM327.

Version

1.0

Date

2025-01-20

Plik zawiera deklaracje funkcji i zmiennych do obsługi modułu OBDII. Obsługuje inicjalizację, odczyt odometru i spalania oraz obliczanie kosztów.

See also

cabulator_settings.h Konfiguracja pinów i parametrów OBD

Definition in file obd_reader.h.

8.8 obd_reader.h

Go to the documentation of this file.

```
00001
00015
00016 #pragma once
00017 #include <Arduino.h>
00018 #include "../cabulator_settings.h"
00019
00020 namespace OBD {
00021
00027     extern bool btConnected;
00028
00034     extern bool elmReady;
00035
00046     bool init();
00047
00057     long readOdometer();
00058
00068     float readFuelRate();
00069
00080     float calculateCost();
00081
00092     void task(void* param);
00093
00094 } // namespace OBD
```

8.9 include/screen_about.h File Reference

Ekran "About" - informacje o aplikacji.

```
#include <TFT_eSPI.h>
```

Functions

- void initAboutScreen (TFT_eSPI *tft)

  Inicjalizuje ekran "About" Rysuje tło i elementy interfejsu na ekranie informacji o aplikacji.
- void handleAboutTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie "About".

8.9.1 Detailed Description

Ekran "About" - informacje o aplikacji.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do inicjalizacji i obsługi ekranu informacyjnego "About".

Definition in file screen_about.h.

8.9.2 Function Documentation

8.9.2.1 handleAboutTouch()

```
void handleAboutTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie "About".

Przetwarza interakcje użytkownika, np. powrót do ekranu głównego.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 28 of file screen_about.cpp.

8.9.2.2 initAboutScreen()

```
void initAboutScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran "About" Rysuje tło i elementy interfejsu na ekranie informacji o aplikacji.

Definition at line 12 of file screen_about.cpp.

8.10 screen_about.h

Go to the documentation of this file.

```
00001
00009
00010 #ifndef SCREEN_ABOUT_H
00011 #define SCREEN_ABOUT_H
00012
00013 #include <TFT_eSPI.h>
00014
00019 void initAboutScreen(TFT_eSPI* tft);
00020
00029 void handleAboutTouch(uint16_t x, uint16_t y);
00030
00031 #endif
```

8.11 include/screen_brightness.h File Reference

Ustawienia jasności ekranu - kontrola podświetlenia TFT.

```
#include <TFT_eSPI.h>
#include <Arduino.h>
```

Functions

- void initBrightnessModule ()

  Inicjalizuje moduł sterowania jasnością
- void initBrightnessScreen (TFT_eSPI ∗tft)

  Inicjalizuje ekran ustawień jasności.
- void handleBrightnessTouch (TFT_eSPI ∗tft, uint16_t x, uint16_t y, uint8_t ∗levelOut)

  Obsługuje dotyk na ekranie ustawień jasności.

Variables

- uint8_t brightnessLevel

  Aktualny poziom jasności podświetlenia.

8.11.1 Detailed Description

Ustawienia jasności ekranu - kontrola podświetlenia TFT.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje i typy do obsługi ekranu jasności. Pozwala na inicjalizację modułu, obsługę dotyku oraz zmianę poziomu jasności podświetlenia wyświetlacza.

Definition in file screen_brightness.h.

8.11.2 Function Documentation

8.11.2.1 handleBrightnessTouch()

```
void handleBrightnessTouch (
            TFT_eSPI * tft,
            uint16_t x,
            uint16_t y,
            uint8_t * levelOut)
```

Obsługuje dotyk na ekranie ustawień jasności.

Przetwarza interakcje z suwakiem i przyciskami, aktualizuje jasność w czasie rzeczywistym.

Parameters

	tft	Wskaźnik do obiektu wyświetlacza TFT
	x	Współrzędna X punktu dotyku
	y	Współrzędna Y punktu dotyku
out	*levelOut*	Wskaźnik do zmiennej gdzie zostanie zapisany nowy poziom jasności

Definition at line 69 of file screen_brightness.cpp.

8.11.2.2 initBrightnessModule()

```
void initBrightnessModule ()
```

Inicjalizuje moduł sterowania jasnością

Konfiguruje PWM do kontroli podświetlenia i wczytuje zapisaną wartość z EEPROM. Należy wywołać raz podczas startu aplikacji.

Definition at line 20 of file screen_brightness.cpp.

8.11.2.3 initBrightnessScreen()

```
void initBrightnessScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran ustawień jasności.

Rysuje interfejs z suwakiem jasności i przyciskami kontrolnymi.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 47 of file screen_brightness.cpp.

8.11.3 Variable Documentation

8.11.3.1 brightnessLevel

```
uint8_t brightnessLevel  [extern]
```

Aktualny poziom jasności podświetlenia.

Wartość w zakresie 0-100 określająca procent maksymalnej jasności.

- 0 = podświetlenie wyłączone
- 100 = maksymalna jasność

Definition at line 17 of file screen_brightness.cpp.

8.12 screen_brightness.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef SCREEN_BRIGHTNESS_H
00013 #define SCREEN_BRIGHTNESS_H
00014
00015 #include <TFT_eSPI.h>
00016 #include <Arduino.h>
00017
00025 extern uint8_t brightnessLevel;
00026
00033 void initBrightnessModule();
00034
00042 void initBrightnessScreen(TFT_eSPI* tft);
00043
00054 void handleBrightnessTouch(TFT_eSPI* tft, uint16_t x, uint16_t y, uint8_t* levelOut);
00055
00056 #endif // SCREEN_BRIGHTNESS_H
```

8.13 include/screen_connection.h File Reference

Ekran połączeń - interfejs konfiguracji połączeń

```
#include <TFT_eSPI.h>
```

Functions

- void initConnectionScreen (TFT_eSPI *tft)

  Inicjalizuje i wyświetla ekran połączeń
- void handleConnectionTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie połączeń

8.13.1 Detailed Description

Ekran połączeń - interfejs konfiguracji połączeń

Version

1.0

Date

2025-01-20

Minimalistyczny ekran pośredni z dwoma przyciskami i powrotem do ustawień.

Definition in file screen_connection.h.

8.13.2 Function Documentation

8.13.2.1 handleConnectionTouch()

```
void handleConnectionTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie połączeń

Przetwarza wybór przycisków i nawigację między ekranami.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 31 of file screen_connection.cpp.

8.13.2.2 initConnectionScreen()

```
void initConnectionScreen (
            TFT_eSPI * tft)
```

Inicjalizuje i wyświetla ekran połączeń

Rysuje przyciski do konfiguracji połączeń oraz opcję powrotu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 15 of file screen_connection.cpp.

8.14 screen_connection.h

Go to the documentation of this file.

```
00001
00009
00010 #ifndef SCREEN_CONNECTION_H
00011 #define SCREEN_CONNECTION_H
00012
00013 #include <TFT_eSPI.h>
00014
00022 void initConnectionScreen(TFT_eSPI* tft);
00023
00032 void handleConnectionTouch(uint16_t x, uint16_t y);
00033
00034 #endif // SCREEN_CONNECTION_H
```

8.15 include/screen_gps-debug.h File Reference

Ekran diagnostyki GPS - szczegółowe dane satelitarne.

```
#include <TFT_eSPI.h>
```

Functions

- void initGpsDebugScreen (TFT_eSPI *tft)

  Inicjalizuje ekran debugowania GPS.
- void updateGpsDebugScreen (TFT_eSPI *tft)

  Aktualizuje dane na ekranie debugowania GPS.
- void handleGpsDebugTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie debugowania GPS.

8.15.1 Detailed Description

Ekran diagnostyki GPS - szczegółowe dane satelitarne.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do inicjalizacji i aktualizacji ekranu debugowania GPS.

Definition in file screen_gps-debug.h.

8.15.2 Function Documentation

8.15.2.1 handleGpsDebugTouch()

```
void handleGpsDebugTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie debugowania GPS.

Przetwarza interakcje, np. powrót do poprzedniego ekranu.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 83 of file screen_gps-debug.cpp.

8.15.2.2 initGpsDebugScreen()

```
void initGpsDebugScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran debugowania GPS.

Rysuje layout ekranu z polami na dane GPS i status połączenia.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 13 of file screen_gps-debug.cpp.

8.15.2.3 updateGpsDebugScreen()

```
void updateGpsDebugScreen (
            TFT_eSPI * tft)
```

Aktualizuje dane na ekranie debugowania GPS.

Odświeża wyświetlane wartości pozycji, liczby satelitów, HDOP oraz inne parametry diagnostyczne.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywoływać cyklicznie dla aktualnych danych

Definition at line 38 of file screen_gps-debug.cpp.

8.16 screen_gps-debug.h

Go to the documentation of this file.

```
00001
00009
00010 #ifndef SCREEN_GPS_DEBUG_H
00011 #define SCREEN_GPS_DEBUG_H
00012
00013 #include <TFT_eSPI.h>
00014
00022 void initGpsDebugScreen(TFT_eSPI* tft);
00023
00034 void updateGpsDebugScreen(TFT_eSPI* tft);
00035
00044 void handleGpsDebugTouch(uint16_t x, uint16_t y);
00045
00046 #endif // SCREEN_GPS_DEBUG_H
```

8.17 include/screen_gps.h File Reference

Ekran GPS - interfejs zarządzania modułem GPS.

```
#include <TFT_eSPI.h>
```

Functions

- void initGpsScreen (TFT_eSPI *tft)

  Inicjalizuje ekran GPS.
- void handleGpsTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie GPS.

8.17.1 Detailed Description

Ekran GPS - interfejs zarządzania modułem GPS.

Version

1.0

Date

2025-01-20

Minimalistyczny ekran z dwoma przyciskami (powrót, reset GPS) oraz miejscem na logi GPS.

Definition in file screen_gps.h.

8.17.2 Function Documentation

8.17.2.1 handleGpsTouch()

```
void handleGpsTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie GPS.

Przetwarza naciśnięcia przycisków powrotu i resetu GPS.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 29 of file screen_gps.cpp.

8.17.2.2 initGpsScreen()

```
void initGpsScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran GPS.

Rysuje przyciski sterujące oraz obszar wyświetlania logów GPS.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 16 of file screen_gps.cpp.

8.18 screen_gps.h

Go to the documentation of this file.

```
00001
00010
00011 #ifndef SCREEN_GPS_H
00012 #define SCREEN_GPS_H
00013
00014 #include <TFT_eSPI.h>
00015
00023 void initGpsScreen(TFT_eSPI* tft);
00024
00033 void handleGpsTouch(uint16_t x, uint16_t y);
00034
00035 #endif // SCREEN_GPS_H
```

8.19 include/screen_home.h File Reference

Główny ekran aplikacji - domyślny widok.

```
#include <TFT_eSPI.h>
```

Functions

- void initHomeScreen (TFT_eSPI *tft)
  Inicjalizuje ekran główny aplikacji.
- void handleHomeTouch (uint16_t x, uint16_t y)
  Obsługuje dotyk na ekranie głównym.
- void updateGPSStatus (TFT_eSPI *tft)
  Aktualizuje widget statusu GPS na ekranie głównym.

8.19.1 Detailed Description

Główny ekran aplikacji - domyślny widok.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do obsługi ekranu głównego (domyślnego widoku). Odpowiada za wyświetlanie interfejsu użytkownika i obsługę interakcji z przyciskami na ekranie głównym.

Definition in file screen_home.h.

8.19.2 Function Documentation

8.19.2.1 handleHomeTouch()

```
void handleHomeTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie głównym.

Przetwarza naciśnięcia przycisków i uruchamia odpowiednie akcje.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 139 of file screen_home.cpp.

8.19.2.2 initHomeScreen()

```
void initHomeScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran główny aplikacji.

Rysuje główny interfejs z przyciskami menu i widgetami statusu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 32 of file screen_home.cpp.

8.19.2.3 updateGPSStatus()

```
void updateGPSStatus (
            TFT_eSPI * tft)
```

Aktualizuje widget statusu GPS na ekranie głównym.

Odświeża ikony i informacje o połączeniu GPS bez przerysowywania całego ekranu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywoływać przy zmianie statusu GPS

Definition at line 72 of file screen_home.cpp.

8.20 screen_home.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef SCREEN_HOME_H
00013 #define SCREEN_HOME_H
00014
00015 #include <TFT_eSPI.h>
00016
00024 void initHomeScreen(TFT_eSPI* tft);
00025
00034 void handleHomeTouch(uint16_t x, uint16_t y);
00035
00045 void updateGPSStatus(TFT_eSPI* tft);
00046
00047 #endif // SCREEN_HOME_H
```

8.21 include/screen_manager.h File Reference

Zarządzanie stanem ekranów aplikacji.

Enumerations

- enum ScreenState {
  SCREEN_WELCOME , SCREEN_HOME , SCREEN_SETTINGS , SCREEN_TARIFF ,
  SCREEN_BRIGHTNESS , SCREEN_CONNECTION , SCREEN_GPS , SCREEN_OBD ,
  SCREEN_OBD_DEBUG , SCREEN_GPS_DEBUG , SCREEN_ABOUT , SCREEN_TRIP }

  Wszystkie dostępne ekrany w systemie.

Variables

- ScreenState currentScreen

  Aktualny stan ekranu aplikacji.

8.21.1 Detailed Description

Zarządzanie stanem ekranów aplikacji.

Version

1.0

Date

2025-01-20

Plik odpowiada za globalny stan aktualnie wyświetlanego ekranu. Dzięki temu w main.cpp wiemy, który handler dotyku wywołać.

Definition in file screen_manager.h.

8.21.2 Enumeration Type Documentation

8.21.2.1 ScreenState

```
enum ScreenState
```

Wszystkie dostępne ekrany w systemie.

Definiuje stany aplikacji używane do nawigacji między ekranami.

Enumerator

SCREEN_WELCOME	Ekran powitalny (wyświetlany przy starcie).
SCREEN_HOME	Ekran główny (domyślny widok).
SCREEN_SETTINGS	Ekran ustawień głównych.
SCREEN_TARIFF	Ekran konfiguracji taryfy.
SCREEN_BRIGHTNESS	Ustawienia jasności i kalibracji dotyku.
SCREEN_CONNECTION	Ekran ustawień połączeń (GPS/OBD).
SCREEN_GPS	Ekran informacyjny GPS.
SCREEN_OBD	Ekran informacyjny OBD.
SCREEN_OBD_DEBUG	Ekran diagnostyki OBD (szczegółowe dane).
SCREEN_GPS_DEBUG	Ekran diagnostyki GPS (satelity, HDOP).
SCREEN_ABOUT	Ekran informacji o autorze i systemie.
SCREEN_TRIP	Ekran aktualnej trasy.

Definition at line 20 of file screen_manager.h.

8.21.3 Variable Documentation

8.21.3.1 currentScreen

```
ScreenState currentScreen  [extern]
```

Aktualny stan ekranu aplikacji.

Zmienna globalna określająca który ekran jest obecnie wyświetlany. Używana w głównej pętli do routowania zdarzeń dotyku.

Note

Modyfikowana przez funkcje handleXxxTouch() przy zmianie ekranu

Definition at line 9 of file screen_manager.cpp.

8.22 screen_manager.h

Go to the documentation of this file.

```
00001
00010
00011 #ifndef SCREEN_MANAGER_H
00012 #define SCREEN_MANAGER_H
00013
00020 enum ScreenState {
00021     SCREEN_WELCOME,
00022     SCREEN_HOME,
00023     SCREEN_SETTINGS,
00024     SCREEN_TARIFF,
00025     SCREEN_BRIGHTNESS,
00026     SCREEN_CONNECTION,
00027     SCREEN_GPS,
00028     SCREEN_OBD,
00029     SCREEN_OBD_DEBUG,
00030     SCREEN_GPS_DEBUG,
00031     SCREEN_ABOUT,
00032     SCREEN_TRIP
00033 };
00034
00043 extern ScreenState currentScreen;
00044
00045 #endif // SCREEN_MANAGER_H
```

8.23 include/screen_obd-debug.h File Reference

Ekran diagnostyki OBD - szczegółowe dane z pojazdu.

```
#include <TFT_eSPI.h>
```

Functions

- void initObdDebugScreen (TFT_eSPI *tft)
  Inicjalizuje ekran debugowania OBD.
- void updateObdDebugScreen (TFT_eSPI *tft)
  Aktualizuje dane na ekranie debugowania OBD.
- void handleObdDebugTouch (uint16_t x, uint16_t y)
  Obsługuje dotyk na ekranie debugowania OBD.

8.23.1 Detailed Description

Ekran diagnostyki OBD - szczegółowe dane z pojazdu.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do inicjalizacji i aktualizacji ekranu debugowania OBD.

Definition in file screen_obd-debug.h.

8.23.2 Function Documentation

8.23.2.1 handleObdDebugTouch()

```
void handleObdDebugTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie debugowania OBD.

Przetwarza interakcje, np. powrót do poprzedniego ekranu.

Parameters

x	Współrzędna X punktu dotyku

y	Współrzędna Y punktu dotyku

Definition at line 82 of file screen_obd-debug.cpp.

8.23.2.2 initObdDebugScreen()

```
void initObdDebugScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran debugowania OBD.

Rysuje layout z polami na dane diagnostyczne pojazdu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 12 of file screen_obd-debug.cpp.

8.23.2.3 updateObdDebugScreen()

```
void updateObdDebugScreen (
            TFT_eSPI * tft)
```

Aktualizuje dane na ekranie debugowania OBD.

Odświeża wyświetlane parametry pojazdu (obroty, temperatura, przepływ powietrza).

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywoływać cyklicznie dla aktualnych danych

Definition at line 41 of file screen_obd-debug.cpp.

8.24 screen_obd-debug.h

Go to the documentation of this file.

```
00001
00009
00010 #ifndef SCREEN_OBD_DEBUG_H
00011 #define SCREEN_OBD_DEBUG_H
00012
00013 #include <TFT_eSPI.h>
00014
00022 void initObdDebugScreen(TFT_eSPI* tft);
00023
00033 void updateObdDebugScreen(TFT_eSPI* tft);
00034
00043 void handleObdDebugTouch(uint16_t x, uint16_t y);
00044
00045 #endif // SCREEN_OBD_DEBUG_H
```

8.25 include/screen_obd.h File Reference

Ekran OBD - interfejs zarządzania modułem OBD.

```
#include <TFT_eSPI.h>
```

Functions

- void initObdScreen (TFT_eSPI *tft)
  Inicjalizuje ekran OBD.
- void handleObdTouch (uint16_t x, uint16_t y)
  Obsługuje dotyk na ekranie OBD.

8.25.1 Detailed Description

Ekran OBD - interfejs zarządzania modułem OBD.

Version

1.0

Date

2025-01-20

Minimalistyczny ekran z przyciskami sterującymi i obszarem logów OBD.

Note

Komentarz w kodzie mówi "SCREEN GPS" - prawdopodobnie błąd copy-paste

Definition in file screen_obd.h.

8.25.2 Function Documentation

8.25.2.1 handleObdTouch()

```
void handleObdTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie OBD.

Przetwarza naciśnięcia przycisków i nawigację.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 30 of file screen_obd.cpp.

8.25.2.2 initObdScreen()

```
void initObdScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran OBD.

Rysuje przyciski sterujące oraz obszar wyświetlania statusu OBD.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 15 of file screen_obd.cpp.

8.26 screen_obd.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef SCREEN_OBD_H
00013 #define SCREEN_OBD_H
00014
00015 #include <TFT_eSPI.h>
00016
00024 void initObdScreen(TFT_eSPI* tft);
00025
00034 void handleObdTouch(uint16_t x, uint16_t y);
00035
00036 #endif // SCREEN_OBD_H
```

8.27 include/screen_settings.h File Reference

Ekran ustawień aplikacji.

```
#include <TFT_eSPI.h>
```

Functions

- void initSettingsScreen (TFT_eSPI *tft)

  Inicjalizuje ekran ustawień
- void handleSettingsTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie ustawień

8.27.1 Detailed Description

Ekran ustawień aplikacji.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do obsługi ekranu ustawień. Umożliwia użytkownikowi dostęp do konfiguracji taryfy, podświetlenia oraz ustawień połączenia z siecią.

Definition in file screen_settings.h.

8.27.2 Function Documentation

8.27.2.1 handleSettingsTouch()

```
void handleSettingsTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie ustawień

Przetwarza wybór opcji i przejścia do ekranów szczegółowych ustawień.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 38 of file screen_settings.cpp.

8.27.2.2 initSettingsScreen()

```
void initSettingsScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran ustawień

Rysuje menu z opcjami konfiguracji taryfy, jasności i połączeń.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 23 of file screen_settings.cpp.

8.28 screen_settings.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef SCREEN_SETTINGS_H
00013 #define SCREEN_SETTINGS_H
00014
00015 #include <TFT_eSPI.h>
00016
00024 void initSettingsScreen(TFT_eSPI* tft);
00025
00034 void handleSettingsTouch(uint16_t x, uint16_t y);
00035
00036 #endif // SCREEN_SETTINGS_H
```

8.29 include/screen_tariff.h File Reference

Ekran ustawień taryfy - konfiguracja rozliczeń

```
#include <TFT_eSPI.h>
```

Enumerations

- enum TariffMode { TARIFF_PER_KM = 0 , TARIFF_PER_LITRE = 1 }

  Tryby obliczania taryfy.

Functions

- void initTariffScreen (TFT_eSPI *tft)

  Inicjalizuje ekran konfiguracji taryfy.
- void handleTariffTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie taryfy.
- void drawTariffValue (TFT_eSPI *tft)

  Rysuje aktualną wartość taryfy na ekranie.
- void loadTariffFromEEPROM ()

  Ładuje ustawienia taryfy z EEPROM.
- void saveTariffToEEPROM ()

  Zapisuje ustawienia taryfy do EEPROM.

Variables

- float tariffValue

  Wartość taryfy w aktualnym trybie.
- TariffMode tariffMode

  Aktualny tryb taryfy.

8.29.1 Detailed Description

Ekran ustawień taryfy - konfiguracja rozliczeń

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do obsługi ekranu taryfy. Umożliwia użytkownikowi ustawienie wartości taryfy za km lub litr oraz zarządzanie trybem rozliczeń.

Definition in file screen_tariff.h.

8.29.2 Enumeration Type Documentation

8.29.2.1 TariffMode

```
enum TariffMode
```

Tryby obliczania taryfy.

Określa sposób rozliczania kosztów przejazdu.

Enumerator

TARIFF_PER_KM	Stała stawka za kilometr.
TARIFF_PER_LITRE	Stała stawka za spalony litr paliwa.

Definition at line 23 of file screen_tariff.h.

8.29.3 Function Documentation

8.29.3.1 drawTariffValue()

```
void drawTariffValue (
            TFT_eSPI * tft)
```

Rysuje aktualną wartość taryfy na ekranie.

Odświeża wyświetlaną liczbę po zmianie wartości przez użytkownika.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 67 of file screen_tariff.cpp.

8.29.3.2 handleTariffTouch()

```
void handleTariffTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie taryfy.

Przetwarza zmianę wartości, przełączanie trybów i zapis ustawień.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 83 of file screen_tariff.cpp.

8.29.3.3 initTariffScreen()

```
void initTariffScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran konfiguracji taryfy.

Rysuje kontrolki do zmiany wartości i trybu taryfy.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 52 of file screen_tariff.cpp.

8.29.3.4 loadTariffFromEEPROM()

```
void loadTariffFromEEPROM ()
```

Ładuje ustawienia taryfy z EEPROM.

Odczytuje zapisaną wartość i tryb taryfy z pamięci nieulotnej. Wywoływane przy starcie aplikacji.

Definition at line 20 of file screen_tariff.cpp.

8.29.3.5 saveTariffToEEPROM()

```
void saveTariffToEEPROM ()
```

Zapisuje ustawienia taryfy do EEPROM.

Zachowuje aktualną wartość i tryb taryfy w pamięci nieulotnej.

Definition at line 40 of file screen_tariff.cpp.

8.29.4 Variable Documentation

8.29.4.1 tariffMode

```
TariffMode tariffMode  [extern]
```

Aktualny tryb taryfy.

Określa czy rozliczamy za km czy za litr paliwa.

Definition at line 18 of file screen_tariff.cpp.

8.29.4.2 tariffValue

```
float tariffValue  [extern]
```

Wartość taryfy w aktualnym trybie.

Przechowuje cenę za jednostkę (km lub litr) w zależności od trybu.

Definition at line 17 of file screen_tariff.cpp.

8.30 screen_tariff.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef SCREEN_TARIFF_H
00013 #define SCREEN_TARIFF_H
00014
00015 #include <TFT_eSPI.h>
00016
00023 enum TariffMode {
00024     TARIFF_PER_KM = 0,
00025     TARIFF_PER_LITRE = 1
00026 };
00027
00033 extern float tariffValue;
00034
00040 extern TariffMode tariffMode;
00041
00049 void initTariffScreen(TFT_eSPI* tft);
00050
00059 void handleTariffTouch(uint16_t x, uint16_t y);
00060
00068 void drawTariffValue(TFT_eSPI* tft);
00069
00076 void loadTariffFromEEPROM();
00077
00083 void saveTariffToEEPROM();
00084
00085 #endif // SCREEN_TARIFF_H
```

8.31 include/screen_trip.h File Reference

Ekran trasy - monitorowanie aktywnej podróży.

```
#include <TFT_eSPI.h>
```

Functions

- void initTripScreen (TFT_eSPI *tft)
  Inicjalizuje ekran trasy.
- void updateTripStatus (TFT_eSPI *tft)
  Aktualizuje wyświetlany status trasy.
- void handleTripTouch (uint16_t x, uint16_t y)
  Obsługuje dotyk na ekranie trasy.
- void resetTripData ()
  Resetuje wszystkie dane trasy do wartości początkowych.
- void saveTripDataToEEPROM ()
  Zapisuje dane trasy do EEPROM.
- bool loadTripDataFromEEPROM ()
  Wczytuje dane trasy z EEPROM.
- void clearTripDataFromEEPROM ()
  Czyści zapisane dane trasy z EEPROM.

Variables

Zmienne stanu trasy

- float distanceTraveled
  Przejechany dystans w kilometrach.
- float fuelUsed
  Zużyte paliwo w litrach.
- bool tripActive
  Czy trasa jest aktywna.
- bool tripPaused
  Czy trasa jest zapauzowana.
- bool obdErrorPending
  Flaga błędu OBD do obsługi w pętli głównej.
- bool resetTripLogicFlag
  Flaga resetu logiki tripa.

8.31.1 Detailed Description

Ekran trasy - monitorowanie aktywnej podróży.

Version

1.0

Date

2025-01-20

Plik nagłówkowy zawiera deklaracje funkcji do obsługi ekranu trasy. Wyświetla przebieg trasy, koszt, zużycie paliwa oraz obsługuje pauzowanie.

Definition in file screen_trip.h.

8.31.2 Function Documentation

8.31.2.1 clearTripDataFromEEPROM()

```
void clearTripDataFromEEPROM ()
```

Czyści zapisane dane trasy z EEPROM.

Usuwa dane z EEPROM gdy trasa zostanie prawidłowo zakończona.

Definition at line 81 of file screen_trip.cpp.

8.31.2.2 handleTripTouch()

```
void handleTripTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie trasy.

Przetwarza przyciski start/pauza/stop oraz nawigację.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 186 of file screen_trip.cpp.

8.31.2.3 initTripScreen()

```
void initTripScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran trasy.

Rysuje interfejs z danymi trasy (dystans, spalanie, koszt) oraz przyciski sterujące.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 102 of file screen_trip.cpp.

8.31.2.4 loadTripDataFromEEPROM()

```
bool loadTripDataFromEEPROM ()
```

Wczytuje dane trasy z EEPROM.

Odtwarza poprzednią sesję w przypadku utraty zasilania.

Returns

true jeśli dane zostały pomyślnie wczytane, false jeśli EEPROM jest pusty

Definition at line 59 of file screen_trip.cpp.

8.31.2.5 resetTripData()

```
void resetTripData ()
```

Resetuje wszystkie dane trasy do wartości początkowych.

Zeruje liczniki dystansu, spalania i kosztu. Wywoływane przy rozpoczynaniu nowej trasy.

Definition at line 89 of file screen_trip.cpp.

8.31.2.6 saveTripDataToEEPROM()

```
void saveTripDataToEEPROM ()
```

Zapisuje dane trasy do EEPROM.

Przechowuje dystans, zużycie paliwa i stan trasy. Wywoływane co 30 sekund podczas aktywnej trasy.

Definition at line 38 of file screen_trip.cpp.

8.31.2.7 updateTripStatus()

```
void updateTripStatus (
            TFT_eSPI * tft)
```

Aktualizuje wyświetlany status trasy.

Odświeża informacje o dystansie, spalaniu i koszcie bez przerysowywania całego ekranu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 147 of file screen_trip.cpp.

8.31.3 Variable Documentation

8.31.3.1 distanceTraveled

```
float distanceTraveled  [extern]
```

Przejechany dystans w kilometrach.

Wartość akumulowana od rozpoczęcia trasy.

Definition at line 32 of file screen_trip.cpp.

8.31.3.2 fuelUsed

```
float fuelUsed  [extern]
```

Zużyte paliwo w litrach.

Suma spalania od rozpoczęcia trasy.

Definition at line 33 of file screen_trip.cpp.

8.31.3.3 obdErrorPending

```
bool obdErrorPending  [extern]
```

Flaga błędu OBD do obsługi w pętli głównej.

Sygnalizuje utratę połączenia z modułem OBD podczas trasy.

Definition at line 34 of file screen_trip.cpp.

8.31.3.4 resetTripLogicFlag

```
bool resetTripLogicFlag  [extern]
```

Flaga resetu logiki tripa.

Używana do synchronizacji operacji resetowania w głównej pętli.

Definition at line 35 of file screen_trip.cpp.

8.31.3.5 tripActive

```
bool tripActive  [extern]
```

Czy trasa jest aktywna.

true = trip rozpoczęty, false = trip nie rozpoczęty

Definition at line 23 of file screen_trip.cpp.

8.31.3.6 tripPaused

```
bool tripPaused  [extern]
```

Czy trasa jest zapauzowana.

true = trip wstrzymany, false = trip w trakcie

Definition at line 24 of file screen_trip.cpp.

8.32 screen_trip.h

Go to the documentation of this file.

```
00001
00010
00011 #ifndef SCREEN_TRIP_H
00012 #define SCREEN_TRIP_H
00013
00014 #include <TFT_eSPI.h>
00015
00018
00024 extern float distanceTraveled;
00025
00031 extern float fuelUsed;
00032
00038 extern bool tripActive;
00039
00045 extern bool tripPaused;
00046
00052 extern bool obdErrorPending;
00053
00059 extern bool resetTripLogicFlag;
00060
00062
00070 void initTripScreen(TFT_eSPI* tft);
00071
00079 void updateTripStatus(TFT_eSPI* tft);
00080
00089 void handleTripTouch(uint16_t x, uint16_t y);
00090
00097 void resetTripData();
00098
00105 void saveTripDataToEEPROM();
00106
00113 bool loadTripDataFromEEPROM();
00114
00120 void clearTripDataFromEEPROM();
00121
00122 #endif // SCREEN_TRIP_H
```

8.33 include/sd_manager.h File Reference

Obsługa karty SD - logowanie tras i danych GPS.

```
#include <Arduino.h>
#include <SD.h>
#include <FS.h>
```

Classes

- struct SDManager::TripData

  Struktura danych trasy do zapisu.
- struct SDManager::GPSData

  Struktura danych GPS do zapisu.

Namespaces

- namespace SDManager

Functions

- bool SDManager::init ()

  Inicjalizuje kartę SD na HSPI (Secondary SPI).
- bool SDManager::isReady ()

  Sprawdza czy karta SD jest gotowa.
- String SDManager::createTripSession ()

  Tworzy nową sesję trasy z timestamp'em.
- bool SDManager::saveTripData (const String &tripPath, const TripData &data)

  Zapisuje dane trasy do pliku trip_data.csv.
- bool SDManager::saveGPSData (const String &tripPath, const GPSData &data)

  Zapisuje wpis danych GPS do pliku gps_log.csv.
- void SDManager::onGPSFix (const GPSData &data)

  Callback wywoływany przez GPS gdy pojawi się nowy fix.
- void SDManager::finalizeTrip (const TripData &data)

  Finalizuje trasę - zapisuje ostateczne dane podsumowania na SD.
- void SDManager::listTrips ()

  Listuje wszystkie dostępne logi tras.
- bool SDManager::getLastTrip (String &tripPath)

  Pobiera dane z ostatniej trasy.

8.33.1 Detailed Description

Obsługa karty SD - logowanie tras i danych GPS.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do obsługi karty SD, tworzenia folderów tras i zapisywania danych z przejazdu.

8.33.1.1 Architektura SPI

Projekt używa **dwóch niezależnych bus SPI** aby uniknąć konfliktów między urządzeniami:

8.33.1.1.1 VSP (Primary SPI) - dla wyświetlacza TFT i touch input

Obsługiwany przez bibliotekę TFT_eSPI:

- **CLK** = GPIO18 (Serial Clock)
- **MOSI** = GPIO23 (Master Out, Slave In - dane do TFT)
- **MISO** = GPIO19 (Master In, Slave Out - dane z touch)

8.33.1.1.2 HSPI (Secondary SPI) - dla karty SD

Inicjalizowany przez SDManager::init() w sd_manager.cpp:

- **CLK** = GPIO27 (Serial Clock)
- **MOSI** = GPIO14 (Master Out, Slave In - dane do karty SD)
- **MISO** = GPIO12 (Master In, Slave Out - dane z karty SD)
- **CS** = GPIO13 (Chip Select - aktywacja karty SD)

Zaleta: Dzięki oddzielnym bus SPI, wyświetlacz i karta SD mogą pracować równocześnie bez konfliktów, ponieważ każde urządzenie ma swoje dedykowane linie komunikacyjne.

8.33.1.2 Struktura danych

Każda trasa jest przechowywana w strukturze:

```
/logs/trips/YYYY-MM-DD_HH-MM-SS/
gps_log.csv        (dane GPS, ~1 entry/sek)
trip_data.csv      (dane końcowe trasy)
```

8.33.1.3 Przepływ danych

- GPS podaje nowy fix → SDManager::onGPSFix() zapisuje do gps_log.csv
- Użytkownik kończy trasę → SDManager::finalizeTrip() zapisuje trip_data.csv

Definition in file sd_manager.h.

8.34 sd_manager.h

Go to the documentation of this file.

```
00001
00045
00046 #ifndef SD_MANAGER_H
00047 #define SD_MANAGER_H
00048
00049 #include <Arduino.h>
00050 #include <SD.h>
00051 #include <FS.h>
00052
00053 namespace SDManager {
00054
00059     struct TripData {
00060         float distanceKm;
00061         float fuelUsedLiters;
00062         int tariffMode;
00063         float tariffValue;
00064         float totalCost;
00065     };
00066
00071     struct GPSData {
00072         double latitude;
00073         double longitude;
00074         uint8_t satellites;
00075         uint16_t hdop;
00076         bool valid;
00077         unsigned long timestamp;
00078     };
00079
```

```
00100     bool init();
00101
00107     bool isReady();
00108
00115     String createTripSession();
00116
00124     bool saveTripData(const String& tripPath, const TripData& data);
00125
00133     bool saveGPSData(const String& tripPath, const GPSData& data);
00134
00153     void onGPSFix(const GPSData& data);
00154
00173     void finalizeTrip(const TripData& data);
00174
00180     void listTrips();
00181
00188     bool getLastTrip(String& tripPath);
00189
00190 }  // namespace SDManager
00191
00192 #endif  // SD_MANAGER_H
```

8.35 include/tft_display.h File Reference

Niskopoziomowa obsługa wyświetlacza TFT.

```
#include <TFT_eSPI.h>
```

Functions

- void initTFT (TFT_eSPI *tft)
  Inicjalizuje wyświetlacz TFT.
- void setBacklight (uint8_t brightness)
  Ustawia jasność podświetlenia wyświetlacza.

8.35.1 Detailed Description

Niskopoziomowa obsługa wyświetlacza TFT.

Version

1.0

Date

2025-01-20

Plik zawiera funkcje do konfiguracji wyświetlacza TFT. Odpowiada za inicjalizację wyświetlacza (rotacja, kalibracja) oraz sterowanie jasnością podświetlenia.

Definition in file tft_display.h.

8.35.2 Function Documentation

8.35.2.1 initTFT()

```
void initTFT (
            TFT_eSPI * tft)
```

Inicjalizuje wyświetlacz TFT.

Konfiguruje rotację ekranu, kalibrację dotyku oraz ustawienia początkowe wyświetlacza.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywołać przed jakąkolwiek operacją rysowania

Definition at line 19 of file tft_display.cpp.

8.35.2.2 setBacklight()

```
void setBacklight (
            uint8_t brightness)
```

Ustawia jasność podświetlenia wyświetlacza.

Steruje jasnością przez PWM na pinie backlight.

Parameters

brightness	Poziom jasności (0-255) 0 = podświetlenie wyłączone 255 = maksymalna jasność

Definition at line 33 of file tft_display.cpp.

8.36 tft_display.h

Go to the documentation of this file.

```
00001
00011
00012 #ifndef TFT_DISPLAY_H
00013 #define TFT_DISPLAY_H
00014
00015 #include <TFT_eSPI.h>
00016
00027 void initTFT(TFT_eSPI* tft);
00028
00038 void setBacklight(uint8_t brightness);
00039
00040 #endif // TFT_DISPLAY_H
```

8.37 src/background.cpp File Reference

```
#include "Background.h"
#include <FS.h>
```

8.38 background.cpp

Go to the documentation of this file.

```
00001 #include "Background.h"
00002 #include <FS.h>
00003
00004 int Background::s_offX = 0;
00005 int Background::s_offY = 0;
00006 uint16_t Background::s_lineBuf[320] = {0};
00007 File Background::s_pngFile;
00008 TFT_eSPI *Background::s_tft = nullptr;
00009 PNG *Background::s_png = nullptr;
00010
00011 // Ustawienie ścieżki do pliku PNG
00012 Background::Background(const String &path) : _path(path) {}
00013 void Background::setPath(const String &path) { _path = path; }
00014 String Background::path() const { return _path; }
00015
00016 // Otworzenie PNG - callbacki
00017 void *Background::pngOpen(const char *filename, int32_t *pFileSize) {
00018
00019   s_pngFile = LittleFS.open(filename, "r");
00020   if (!s_pngFile)
00021     return nullptr;
00022   *pFileSize = s_pngFile.size();
00023   return (void *)&s_pngFile; // handle, wraca w pFile->fHandle
00024 }
00025
00026 // Zamknięcie PNG
00027 void Background::pngClose(void *handle) {
00028
00029   File *f = (File *)handle;
00030   if (f)
00031     f->close();
00032 }
00033
00034 // Odczyt danych PNG
00035 int32_t Background::pngRead(PNGFILE *pFile, uint8_t *pBuff, int32_t iLen) {
00036
00037   File *f = (File *)pFile->fHandle;
00038   return f->read(pBuff, iLen);
00039 }
00040
00041 // Ustawienie pozycji w pliku PNG
00042 int32_t Background::pngSeek(PNGFILE *pFile, int32_t iPosition) {
00043
00044   File *f = (File *)pFile->fHandle;
00045   f->seek(iPosition);
00046   return iPosition;
00047 }
00048
00049 // DRAW jednej linii (używa s_png i s_tft)
00050 int Background::pngDraw(PNGDRAW *p) {
00051
00052   // konwersja linii PNG -> RGB565
00053   s_png->getLineAsRGB565(p, s_lineBuf, PNG_RGB565_BIG_ENDIAN, 0xFFFFFFFF);
00054   // rysowanie od x = s_offX, na wysokości s_offY + p->y, szerokość = p->iWidth
00055   s_tft->pushImage(s_offX, s_offY + p->y, p->iWidth, 1, s_lineBuf);
00056   return 1;
00057 }
00058
00059 // Rysowanie PNG na pełnym ekranie
00060 bool Background::drawPngFullScreen(const char *path, bool center) {
00061
00062   // otwarcie pliku PNG
00063   if (s_png->open(path, pngOpen, pngClose, pngRead, pngSeek, pngDraw) != PNG_SUCCESS) {
00064
00065     Serial.println("[ERROR] PNG: open/init FAIL");
00066     return false;
00067   }
00068
00069   // przeliczenie offsetów do wyśrodkowania
00070   s_offX = s_offY = 0;
00071   if (center) {
00072
00073     int iw = s_png->getWidth();
00074     int ih = s_png->getHeight();
00075     if (iw < s_tft->width())
00076       s_offX = (s_tft->width() - iw) / 2;
00077     if (ih < s_tft->height())
00078       s_offY = (s_tft->height() - ih) / 2;
00079   }
00080
00081   // Dekodowanie i rysowanie PNG
00082   s_png->decode(NULL, 0);
```

```
00083   s_png->close();
00084   return true;
00085 }
00086
00087 bool Background::draw(TFT_eSPI &tft, PNG &png, bool center) {
00088   // Ustawienie statycznych wskaźników
00089   s_tft = &tft;
00090   s_png = &png;
00091
00092   // Rysowanie PNG
00093   return drawPngFullScreen(_path.c_str(), center);
00094 }
00095
00096 // Listowanie plików w LittleFS
00097 void Background::listFS(const char *dir) {
00098
00099   File root = LittleFS.open("/");
00100   if (!root || !root.isDirectory()) {
00101
00102     Serial.println("[ERROR][FS] root is not a directory or failed to open");
00103     return;
00104   }
00105
00106   int fileCount = 0;
00107   File file = root.openNextFile();
00108
00109   while (file) {
00110
00111     Serial.printf("%s (%uB)\n", file.name(), (unsigned)file.size());
00112     fileCount++;
00113     file = root.openNextFile();
00114   }
00115   Serial.printf("[FS] Found %d files in FS root\n", fileCount);
00116 }
```

8.39 src/gps_reader.cpp File Reference

```
#include "gps_reader.h"
#include "sd_manager.h"
#include <TinyGPSPlus.h>
#include <sys/time.h>
#include <time.h>
```

Namespaces

- namespace GPS

  Przestrzeń nazw zawierająca całą obsługę modułu GPS.

Macros

- #define GPS_DEBUG_RAW 0

Functions

- static uint32_t GPS::msSince (uint32_t t0)
- void GPS::begin ()

  Inicjalizacja modułu GPS.
- bool GPS::poll (Fix &out)

  Pobiera najnowszą próbkę GPS.
- const char *const * GPS::getLastRawLines ()
- bool GPS::setSystemTimeFromGPS (const Fix &fix)

  Zwraca surowe linie NMEA z modułu GPS.
- bool GPS::hasLiveFix ()

  Sprawdza, czy mamy aktualny fix GPS.
- void GPS::debugStatus ()

  Wypisuje status GPS na Serial (debug).

Variables

- static TinyGPSPlus GPS::gps
- static HardwareSerial * GPS::port = &Serial2
- static uint32_t GPS::lastSample = 0
- static char GPS::rawLogLines [5][128] = {{0}}
- static uint8_t GPS::rawLinePos = 0
- static char GPS::currentRawLine [128] = {0}
- static uint8_t GPS::currentRawLinePos = 0

8.39.1 Macro Definition Documentation

8.39.1.1 GPS_DEBUG_RAW

```
#define GPS_DEBUG_RAW 0
```

Definition at line 8 of file gps_reader.cpp.

8.40 gps_reader.cpp

Go to the documentation of this file.

```
00001 #include "gps_reader.h"
00002 #include "sd_manager.h"
00003 #include <TinyGPSPlus.h>
00004 #include <sys/time.h>
00005 #include <time.h>
00006
00007 #ifndef GPS_DEBUG_RAW
00008 #define GPS_DEBUG_RAW 0
00009 #endif
00010
00011 namespace GPS {
00012
00013 static TinyGPSPlus gps;                     // Obiekt TinyGPSPlus do parsowania NMEA
00014 static HardwareSerial* port = &Serial2;     // Używamy Serial2 dla ESP32
00015 static uint32_t lastSample = 0;             // Timestamp ostatniej próbki
00016 Fix lastFix;                                // Ostatni fix GPS
00017
00018 // Bufor na ostatnią linię NMEA
00019 static char rawLogLines[5][128] = {{0}};    // 5 ostatnich linii NMEA
00020 static uint8_t rawLinePos = 0;              // Pozycja w buforze
00021 static char currentRawLine[128] = {0};      // Bieżąca linia NMEA
00022 static uint8_t currentRawLinePos = 0;       // Pozycja w bieżącej linii
00023
00024 static inline uint32_t msSince(uint32_t t0) {
00025     return millis() - t0;
00026 }
00027
00028 // Inicjalizacja GPS
00029 void begin() {
00030
00031     Serial.println("\n[GPS] Initializing GPS Module...");
00032
00033     port->begin(BAUD_RATE, SERIAL_8N1, PIN_RX, PIN_TX);
00034     lastSample = millis();
00035     lastFix = Fix{};
00036
00037     Serial.println("[GPS] ========================");
00038     Serial.printf("[GPS] Port: Serial2\n");
00039     Serial.printf("[GPS] Baud: %d\n", BAUD_RATE);
00040     Serial.printf("[GPS] RX Pin: %d\n", PIN_RX);
00041     Serial.printf("[GPS] TX Pin: %d\n", PIN_TX);
00042     Serial.println("[GPS] ========================");
00043
00044     Serial.println("[GPS] GPS Module initialized");
00045 }
00046
```

```
00047 // Polling GPS - zwraca true jeśli jest nowy fix
00048 bool poll(Fix& out) {
00049
00050     if (!port) return false;
00051
00052     while (port->available() > 0) {
00053
00054         char c = static_cast<char>(port->read());
00055         // Zebranie surowej linii NMEA dla debugu
00056         if (c == '\n' || c == '\r') {
00057
00058             if (currentRawLinePos > 0) {
00059
00060                 // Zapisz linię do bufora ostatnich 5 linii
00061                 currentRawLine[currentRawLinePos] = '\0';
00062                 for (int i = 0; i < 4; ++i) {
00063                     strcpy(rawLogLines[i], rawLogLines[i+1]);
00064                 }
00065
00066                 strncpy(rawLogLines[4], currentRawLine, sizeof(rawLogLines[4])-1);
00067                 rawLogLines[4][sizeof(rawLogLines[4])-1] = '\0';
00068                 currentRawLinePos = 0;
00069             }
00070
00071         } else { // Normalny znak
00072
00073             if (currentRawLinePos < sizeof(currentRawLine) - 2)
00074                 currentRawLine[currentRawLinePos++] = c;
00075         }
00076         gps.encode(c);
00077     }
00078
00079     // Sprawdzenie czy minął czas próbkowania
00080     if (msSince(lastSample) >= SAMPLE_MS) {
00081
00082         lastSample = millis();
00083         bool locValid = gps.location.isValid() && gps.location.age() < 2000;    // Valid jeśli <2s
00084         lastFix.valid = locValid;                                               // Ustaw valid
00085         lastFix.takenAtMs = lastSample;                                         // Timestamp próbki

00086         lastFix.sats = gps.satellites.value();                                  // Liczba satelitów
00087         uint32_t hd = gps.hdop.value();                                         // HDOP x100
00088         lastFix.hdop = (hd <= 0xFFFF) ? (uint16_t)hd : (uint16_t)65535;         // HDOP x100 z
      klipowaniem
00089
00090         // Parsuj pozycję
00091         if (locValid) {
00092
00093             lastFix.lat = gps.location.lat();   // Szerokość
00094             lastFix.lng = gps.location.lng();   // Długość
00095         }
00096
00097         // Parsuj datę i czas z GPS
00098         if (gps.date.isValid() && gps.time.isValid()) {
00099
00100             lastFix.year = gps.date.year();
00101             lastFix.month = gps.date.month();
00102             lastFix.day = gps.date.day();
00103             lastFix.hour = gps.time.hour();
00104             lastFix.minute = gps.time.minute();
00105             lastFix.second = gps.time.second();
00106             lastFix.dateTimeValid = true;
00107
00108         } else
00109             lastFix.dateTimeValid = false;
00110
00111         // Przekaż dane wyjściowe
00112         out = lastFix;
00113
00114         // Parsowanie danych na zapis do SD
00115         SDManager::GPSData gpsData;
00116         gpsData.latitude = lastFix.lat;
00117         gpsData.longitude = lastFix.lng;
00118         gpsData.satellites = lastFix.sats;
00119         gpsData.hdop = lastFix.hdop;
00120         gpsData.valid = lastFix.valid;
00121         gpsData.timestamp = lastFix.takenAtMs;
00122         SDManager::onGPSFix(gpsData);
00123
00124         return true;
00125     }
00126     return false;
00127 }
00128
00129 // Funkcja zwracająca wskaźnik do tablicy 5 ostatnich linii NMEA
00130 const char* const* getLastRawLines() {
00131
```

```
00132     static const char* ptrs[5];
00133     for (int i = 0; i < 5; ++i) ptrs[i] = rawLogLines[i];
00134     return ptrs;
00135 }
00136
00137 // Synchronizacja czasu systemowego z GPS
00138 bool setSystemTimeFromGPS(const Fix& fix) {
00139
00140     if (!fix.dateTimeValid) {
00141         return false;
00142     }
00143
00144     // Konwertowanie daty/czasu z GPS na structure timeval
00145     struct tm timeinfo = {};
00146     timeinfo.tm_year = fix.year - 1900;      // tm_year to lata od 1900
00147     timeinfo.tm_mon = fix.month - 1;         // tm_mon to 0-11
00148     timeinfo.tm_mday = fix.day;
00149     timeinfo.tm_hour = fix.hour;
00150     timeinfo.tm_min = fix.minute;
00151     timeinfo.tm_sec = fix.second;
00152     timeinfo.tm_isdst = -1;                  // Nieznane informacje o DST
00153
00154     // Konwersja na timestamp
00155     time_t timestamp = mktime(&timeinfo);
00156     if (timestamp == -1) {
00157
00158         Serial.println("[GPS] ERROR: Failed to convert GPS time to timestamp");
00159         return false;
00160     }
00161
00162     // Ustawienie czasu systemowego
00163     struct timeval tv = {};
00164     tv.tv_sec = timestamp;
00165     tv.tv_usec = 0;
00166
00167     if (settimeofday(&tv, nullptr) == 0) {
00168
00169         Serial.printf("[GPS] System time synchronized: %04d-%02d-%02d %02d:%02d:%02d\n",
00170             fix.year, fix.month, fix.day, fix.hour, fix.minute, fix.second);
00171         return true;
00172
00173     } else {
00174
00175         Serial.println("[GPS] ERROR: Failed to set system time");
00176         return false;
00177     }
00178 }
00179
00180     // Sprawdzenie czy jest aktualny fix GPS
00181     bool hasLiveFix() {
00182         return gps.location.isValid() && gps.location.age() < 2000;
00183     }
00184
00185     // Debugowanie statusu GPS
00186     void debugStatus() {
00187
00188         Fix fix;
00189         if (poll(fix)) {
00190
00191             if (fix.valid) {
00192
00193                 Serial.printf("[GPS] Lat: %.6f, Lng: %.6f, Sats: %d\n",
00194                     fix.lat, fix.lng, fix.sats);
00195             }
00196         }
00197     }
00198 }
```

8.41 src/gui_elements.cpp File Reference

```
#include "gui_elements.h"
```

Functions

- void drawText (TFT_eSPI *tft, const char *text, int x, int y, const TextStyle &style, uint16_t overrideColor)

Rysuje tekst na ekranie TFT z określonym stylem.

- void drawTextWithBackground (TFT_eSPI *tft, const char *text, int x, int y, uint8_t datum, uint8_t font, uint16_t textColor, uint16_t bgColor, int16_t paddingWidth)

  Rysuje tekst z kolorowym tłem.

8.41.1 Function Documentation

8.41.1.1 drawText()

```
void drawText (
            TFT_eSPI * tft,
            const char * text,
            int x,
            int y,
            const TextStyle & style,
            uint16_t overrideColor = 0)
```

Rysuje tekst na ekranie TFT z określonym stylem.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT
text	Tekst do wyświetlenia (C-string)
x	Współrzędna X pozycji tekstu
y	Współrzędna Y pozycji tekstu
style	Referencja do struktury TextStyle definiującej styl
overrideColor	Opcjonalny kolor nadpisujący styl (0 = użyj koloru ze stylu)

Note

Punkt odniesienia tekstu (datum) określa jak x,y interpretowane są względem tekstu

See also

TextStyle

Definition at line 8 of file gui_elements.cpp.

8.41.1.2 drawTextWithBackground()

```
void drawTextWithBackground (
            TFT_eSPI * tft,
            const char * text,
            int x,
            int y,
            uint8_t datum,
            uint8_t font,
            uint16_t textColor,
            uint16_t bgColor,
            int16_t paddingWidth = 0)
```

Rysuje tekst z kolorowym tłem.

Renderuje prostokątne tło pod tekstem, przydatne do tworzenia etykiet, przycisków lub wyróżnionych napisów.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT
text	Tekst do wyświetlenia
x	Współrzędna X pozycji tekstu
y	Współrzędna Y pozycji tekstu
datum	Punkt odniesienia tekstu (np. MC_DATUM dla środka)
font	Numer czcionki TFT_eSPI
textColor	Kolor tekstu (RGB565)
bgColor	Kolor tła (RGB565)
paddingWidth	Dodatkowa szerokość tła po bokach [px] (domyślnie 0)

Warning

Tło jest rysowane przed tekstem, może nadpisać istniejącą grafikę

Note

PaddingWidth pozwala na dodanie przestrzeni po bokach tekstu w tle

Definition at line 18 of file gui_elements.cpp.

8.42 gui_elements.cpp

Go to the documentation of this file.

```
00001 #include "gui_elements.h"
00002
00003 // ================================================================================
00004 // Funkcje pomocnicze do rysowania tekstów
00005 // ================================================================================
00006
00007 // Funkcja rysująca tekst
00008 void drawText(TFT_eSPI* tft, const char* text, int x, int y, const TextStyle& style, uint16_t
      overrideColor)
00009 {
00010   tft->setTextDatum(style.datum);  // Ustawienie punktu odniesienia dla tekstu
00011   tft->setTextFont(style.font);    // Ustawienie czcionki
00012   uint16_t colorToUse = (overrideColor != 0) ? overrideColor : style.color; // Ustalamy kolor tekstu
00013   tft->setTextColor(colorToUse, TFT_BLACK);
00014   tft->drawString(text, x, y);  // Rysowanie tekstu
00015 }
00016
00017 // Funkcja rysująca tekst z tłem
00018 void drawTextWithBackground(
00019     TFT_eSPI* tft,
00020     const char* text,
00021     int x, int y,
00022     uint8_t datum,
00023     uint8_t font,
00024     uint16_t textColor,
00025     uint16_t bgColor,
00026     int16_t paddingWidth
00027 ) {
00028     tft->setTextDatum(datum);
00029     tft->setTextFont(font);
00030     int16_t w = tft->textWidth(text);
00031     int16_t h = tft->fontHeight();
00032     int16_t vMargin = h / 5;
00033     int16_t h_bg = h + 2 * vMargin;
00034     int16_t w_bg = w;
00035     int bg_x = x, bg_y = y;
00036     switch (datum) {
00037       case TR_DATUM:
00038         w_bg = w + paddingWidth;
```

```
00039         bg_x = x - w_bg;
00040         bg_y = y - vMargin;
00041         break;
00042       case TL_DATUM:
00043         w_bg = w + paddingWidth;
00044         bg_x = x;
00045         bg_y = y - vMargin;
00046         break;
00047       case TC_DATUM:
00048         w_bg = w + 2 * paddingWidth;
00049         bg_x = x - w_bg / 2;
00050         bg_y = y - vMargin;
00051         break;
00052       default:
00053         w_bg = w + 2 * paddingWidth;
00054         bg_x = x - w_bg / 2;
00055         bg_y = y - vMargin;
00056         break;
00057     }
00058     tft->fillRect(bg_x, bg_y, w_bg, h_bg, bgColor);
00059     tft->setTextColor(textColor, bgColor);
00060     tft->drawString(text, x, y);
00061 }
```

8.43 src/main.cpp File Reference

```
#include <Arduino.h>
#include <TFT_eSPI.h>
#include <EEPROM.h>
#include "esp_task_wdt.h"
#include "tft_display.h"
#include "gui_elements.h"
#include "gps_reader.h"
#include "obd_reader.h"
#include "background.h"
#include "screen_manager.h"
#include "sd_manager.h"
#include "screen_home.h"
#include "screen_settings.h"
#include "screen_brightness.h"
#include "screen_connection.h"
#include "screen_about.h"
#include "screen_gps.h"
#include "screen_gps-debug.h"
#include "screen_tariff.h"
#include "screen_trip.h"
#include "screen_obd.h"
#include "screen_obd-debug.h"
```

Functions

- void taskOBD (void *param)
- void taskGPS (void *param)
- void setup ()
- void loop ()

Variables

- TFT_eSPI tft
- PNG png
- String currentTripPath = ""

8.43.1 Function Documentation

8.43.1.1 loop()

```
void loop ()
```

Definition at line 136 of file main.cpp.

8.43.1.2 setup()

```
void setup ()
```

Definition at line 74 of file main.cpp.

8.43.1.3 taskGPS()

```
void taskGPS (
            void * param)
```

Definition at line 45 of file main.cpp.

8.43.1.4 taskOBD()

```
void taskOBD (
            void * param)
```

Definition at line 40 of file main.cpp.

8.43.2 Variable Documentation

8.43.2.1 currentTripPath

```
String currentTripPath = ""
```

Definition at line 32 of file main.cpp.

8.43.2.2 png

```
PNG png
```

Definition at line 31 of file main.cpp.

8.43.2.3 tft

```
TFT_eSPI tft
```

Definition at line 30 of file main.cpp.

8.44 main.cpp

Go to the documentation of this file.

```
00001 #include <Arduino.h>
00002 #include <TFT_eSPI.h>
00003 #include <EEPROM.h>
00004 #include "esp_task_wdt.h"
00005
00006 #include "tft_display.h"
00007 #include "gui_elements.h"
00008 #include "gps_reader.h"
00009 #include "obd_reader.h"
00010 #include "background.h"
00011 #include "screen_manager.h"
00012 #include "sd_manager.h"
00013
00014 #include "screen_home.h"
00015 #include "screen_settings.h"
00016 #include "screen_brightness.h"
00017 #include "screen_connection.h"
00018 #include "screen_about.h"
00019 #include "screen_gps.h"
00020 #include "screen_gps-debug.h"
00021 #include "screen_tariff.h"
00022 #include "screen_trip.h"
00023 #include "screen_obd.h"
00024 #include "screen_obd-debug.h"
00025
00026 // ================================================================================
00027 // GLOBALNE ZMIENNE
00028 // ================================================================================
00029
00030 TFT_eSPI tft;
00031 PNG png;
00032 String currentTripPath = "";  // Ścieżka do folderu obecnej trasy
00033
00034
00035 // ================================================================================
00036 // TASKI RTOS - Wielowątkowe wykonanie
00037 // ================================================================================
00038
00039 // Task OBD - czytanie danych z modułu OBD co 2 sekundy
00040 void taskOBD(void* param) {
00041   OBD::task(param);
00042 }
00043
00044 // Task GPS - odczyt i debug GPS co 10 sekund
00045 void taskGPS(void* param) {
00046   static bool timeSync = false;     // Flaga synchronizacji czasu systemowego z GPS
00047
00048   while (true) {
00049
00050       GPS::Fix fix;
00051
00052       if (GPS::poll(fix)) {
00053
00054           GPS::debugStatus();
00055           // Synchronizacja czasu systemowego z GPS
00056           if (!timeSync && fix.dateTimeValid) {
00057
00058               if (GPS::setSystemTimeFromGPS(fix)) {
00059
00060                   timeSync = true;
00061                   Serial.println("[GPS] System time synchronized from GPS");
00062               }
00063           }
00064
00065       }
00066       vTaskDelay(10000 / portTICK_PERIOD_MS); // Opóźnienie 10 sekund
00067   }
00068 }
00069
00070 // ================================================================================
00071 // SETUP - Inicjalizacja systemu
00072 // ================================================================================
00073
00074 void setup() {
00075
00076   Serial.begin(115200);
00077
00078   // Wyłączenie WDT dla rdzenia 0 (uniknięcie resetów podczas długich operacji)
00079   disableCore0WDT();
00080
00081   // Całkowite wyciszenie logów (przynajmniej próbowałem bo obd nadal pluje faktami)
00082   esp_log_level_set("*", ESP_LOG_NONE);
```

```
00083
00084   Serial.println("\n\n\n[SYSTEM] ========== INITIALIZING ==========\n");
00085
00086   // ========== FILESYSTEM ==========
00087   LittleFS.begin(true);
00088   Serial.println("[FS] ========================");
00089   Background::listFS("/");
00090   Serial.println("[FS] ========================\n");
00091
00092   // ========== WYŚWIETLACZ ==========
00093   initTFT(&tft);
00094   Background bg("/loading.png");
00095   bg.draw(tft, png, true);
00096   currentScreen = SCREEN_WELCOME;
00097
00098   // ========== EEPROM & JASNOŚĆ & TARYFA ==========
00099   EEPROM.begin(100);            // Zarezerowanie 100 bajtów w EEPROM
00100   initBrightnessModule();
00101   loadTariffFromEEPROM();
00102
00103   // ========== INICJALIZACJA KARTY SD ==========
00104   if (!SDManager::init()) {
00105       Serial.println("[WARNING] SD Card initialization failed, continuing anyway\n");
00106   }
00107
00108   // Rekalibracja dotyku po inicjalizacji SD
00109   uint16_t calData_recal[5] = { 243, 3566, 356, 3415, 1 };
00110   tft.setTouch(calData_recal);
00111   Serial.println("[TOUCH] Re-calibrated after SD init");
00112
00113   // ========== INICJALIZACJA GPS I OBD ==========
00114   GPS::begin();
00115
00116   if (!OBD::init()) {
00117       Serial.println("[WARNING] OBD initialization failed, continuing anyway\n");
00118   }
00119
00120   Serial.println("[SYSTEM] ========== INIT COMPLETE ==========\n");
00121
00122   // ========== INICJALIZACJA EKRANU GŁÓWNEGO ==========
00123   initHomeScreen(&tft);
00124   currentScreen = SCREEN_HOME;
00125
00126   // ========== FREERTOS TASKS ==========
00127   xTaskCreate(taskOBD, "OBD", 8192, (void*)&tft, 2, NULL);
00128   xTaskCreate(taskGPS, "GPS", 4096, (void*)&tft, 1, NULL);
00129 }
00130
00131
00132 // ================================================================================
00133 // LOOP - Główna pętla obsługi UI (blokujący dotyk i zmiana ekranów)
00134 // ================================================================================
00135
00136 void loop() {
00137   uint16_t x, y;
00138
00139   // ========== OBSŁUGA DOTYKU ==========
00140   static unsigned long lastTouchTime = 0;
00141   const unsigned long debounceDelay = 250;
00142   static bool touchReleased = true;
00143
00144   if (tft.getTouch(&x, &y)) {
00145       x = tft.width() - x;
00146       y = tft.height() - y;
00147       Serial.printf("[TOUCH] x=%d, y=%d, screen=%d\n", x, y, currentScreen);
00148
00149       if (currentScreen == SCREEN_BRIGHTNESS) {
00150           // Jasność ma specjalną obsługę bez debounce
00151           uint8_t newLevel = 0;
00152           handleBrightnessTouch(&tft, x, y, &newLevel);
00153       } else if (touchReleased) {
00154           touchReleased = false;
00155           unsigned long now = millis();
00156
00157           if (now - lastTouchTime > debounceDelay) {
00158               lastTouchTime = now;
00159
00160               // DELEGOWANIE OBSŁUGI DOTYKU DO AKTYWNEGO EKRANU
00161               switch (currentScreen) {
00162                   case SCREEN_HOME:
00163                       handleHomeTouch(x, y);
00164                       break;
00165                   case SCREEN_SETTINGS:
00166                       handleSettingsTouch(x, y);
00167                       break;
00168                   case SCREEN_TARIFF:
00169                       handleTariffTouch(x, y);
```

```
00170                     break;
00171                 case SCREEN_CONNECTION:
00172                     handleConnectionTouch(x, y);
00173                     break;
00174                 case SCREEN_ABOUT:
00175                     handleAboutTouch(x, y);
00176                     break;
00177                 case SCREEN_GPS:
00178                     handleGpsTouch(x, y);
00179                     break;
00180                 case SCREEN_GPS_DEBUG:
00181                     handleGpsDebugTouch(x, y);
00182                     break;
00183                 case SCREEN_TRIP:
00184                     handleTripTouch(x, y);
00185                     break;
00186                 case SCREEN_OBD:
00187                     handleObdTouch(x, y);
00188                     break;
00189                 case SCREEN_OBD_DEBUG:
00190                     handleObdDebugTouch(x, y);
00191                     break;
00192                 default:
00193                     break;
00194             }
00195         }
00196     }
00197 } else {
00198     touchReleased = true;
00199 }
00200
00201 // ========== AKTUALIZACJA DANYCH W ZALEŻNOŚCI OD AKTYWNEGO EKRANU ==========
00202 static unsigned long lastScreenUpdate = 0;
00203 if (millis() - lastScreenUpdate > 1000) {
00204
00205     if (currentScreen == SCREEN_HOME)
00206         updateGPSStatus(&tft);
00207     else if (currentScreen == SCREEN_TRIP)
00208         updateTripStatus(&tft);
00209     else if (currentScreen == SCREEN_OBD_DEBUG)
00210         updateObdDebugScreen(&tft);
00211
00212
00213     lastScreenUpdate = millis();
00214 }
00215
00216 // ========== ZAPIS DANYCH TRASY DO EEPROM CO 30 SEKUND ==========
00217 static unsigned long lastTripSave = 0;
00218 if (tripActive && millis() - lastTripSave > 30000) {
00219
00220     saveTripDataToEEPROM();
00221     lastTripSave = millis();
00222 }
00223 }
```

8.45 src/obd_reader.cpp File Reference

```
#include "obd_reader.h"
#include "screen_trip.h"
#include "screen_tariff.h"
#include "../cabulator_settings.h"
#include <BluetoothSerial.h>
```

Namespaces

- namespace OBD

  Przestrzeń nazw zawierająca całą obsługę modułu OBDII.

Functions

- bool OBD::sendCmd (const char *cmd, char *response, int maxLen, int timeout=1000)
- bool OBD::init ()

  Inicjalizuje i łączy się z modułem OBD przez Bluetooth.
- long OBD::readOdometer ()

  Odczytuje wartość odometru z pojazdu.
- float OBD::readFuelRate ()

  Odczytuje bieżące zużycie paliwa na podstawie MAF.
- float OBD::calculateCost ()

  Oblicza koszt przejazdu na podstawie spalania i taryfy.
- void OBD::task (void *param)

  Task FreeRTOS obsługujący komunikację OBD w tle.

Variables

- BluetoothSerial OBD::SerialBT

8.46 obd_reader.cpp

Go to the documentation of this file.

```
00001 #include "obd_reader.h"
00002 #include "screen_trip.h"
00003 #include "screen_tariff.h"
00004 #include "../cabulator_settings.h"
00005 #include <BluetoothSerial.h>
00006
00007 // ============================================================================
00008 // MINIMALNA IMPLEMENTACJA OBD
00009 // ============================================================================
00010
00011 namespace OBD {
00012
00013 BluetoothSerial SerialBT;
00014
00015 // Status połączenia OBD
00016 bool btConnected = false;
00017 bool elmReady = false;
00018
00019 // Wysyłanie komendy do OBD - zwraca odpowiedź w buforze
00020 bool sendCmd(const char* cmd, char* response, int maxLen, int timeout = 1000) {
00021
00022     while (SerialBT.available()) SerialBT.read();
00023     SerialBT.print(cmd);
00024     SerialBT.print("\r");
00025
00026     int idx = 0;
00027     unsigned long start = millis();
00028
00029     while (millis() - start < timeout && idx < maxLen - 1) {
00030
00031         if (SerialBT.available()) {
00032
00033             char c = SerialBT.read();
00034             if (c == '>') break;
00035             if (c != '\r' && c != '\n') {  // Ignoruj znaki nowej linii i powrotu
00036                 response[idx++] = c;
00037             }
00038         }
00039         vTaskDelay(10 / portTICK_PERIOD_MS);
00040     }
00041
00042     response[idx] = '\0';  // Eliminacja pustych znaków na końcu
00043
00044     // Usunięcie trailing spaces (spacji na końcu odpowiedzi)
00045     while (idx > 0 && response[idx-1] == ' ') {
00046         response[--idx] = '\0';
00047     }
00048
```

```
00049     return (idx > 0);
00050 }
00051
00052 // Inicjalizacja połączenia OBD
00053 bool init() {
00054
00055 #if OBD_SIMULATION_MODE
00056     Serial.println("\n[OBD] ===== SIMULATION MODE ENABLED =====");
00057     Serial.println("[OBD] Module status: SIMULATED");
00058     Serial.println("[OBD] Simulating: 100 km/h, 0.8 - 10 L/H\n");
00059     Serial.println("\n[OBD] ====================================");
00060     btConnected = true;
00061     elmReady = true;
00062     return true;
00063
00064 #else
00065     // Połączenie Bluetooth
00066     SerialBT.begin(OBD_CONFIG::DEVICE_NAME, true);
00067     uint8_t addr[6];
00068     sscanf(OBD_CONFIG::DEVICE_MAC_STR, "%hhx:%hhx:%hhx:%hhx:%hhx:%hhx",
00069            &addr[0], &addr[1], &addr[2], &addr[3], &addr[4], &addr[5]);
00070
00071     bool connected = false;
00072     for (int i = 1; i <= 3; i++) {
00073
00074         Serial.printf("[OBD] Module CONNECTING (TRY %d/3)\n", i);
00075         if (SerialBT.connect(addr)) {
00076
00077             connected = true;
00078             break;
00079         }
00080
00081         vTaskDelay(1000 / portTICK_PERIOD_MS);
00082     }
00083
00084     if (!connected) {
00085
00086         Serial.println("[OBD] Module NOT CONNECTED");
00087         btConnected = false;
00088         elmReady = false;
00089         return false;
00090     }
00091
00092     // Inicjalizacja ELM327
00093     char resp[128];
00094     vTaskDelay(500 / portTICK_PERIOD_MS);
00095     sendCmd("ATZ", resp, sizeof(resp), 2000);
00096     vTaskDelay(500 / portTICK_PERIOD_MS);
00097     sendCmd("ATE0", resp, sizeof(resp), 500);
00098     sendCmd("ATL0", resp, sizeof(resp), 500);
00099     sendCmd("ATS0", resp, sizeof(resp), 500);
00100    sendCmd("ATH0", resp, sizeof(resp), 500);
00101    sendCmd("ATSP6", resp, sizeof(resp), 1000);
00102    sendCmd("0100", resp, sizeof(resp), 3000);
00103
00104    if (strstr(resp, "41") != NULL) {
00105
00106        Serial.println("[OBD] Module CONNECTED BLUETOOTH + ELM");
00107        btConnected = true;
00108        elmReady = true;
00109        return true;
00110
00111    } else {
00112
00113        Serial.println("[OBD] Module CONNECTED ONLY BLUETOOTH");
00114        btConnected = true;
00115        elmReady = false;
00116        return false;
00117    }
00118
00119 #endif
00120 }
00121
00122 // Odczyt odometru - zwraca KM
00123 long readOdometer() {
00124
00125 #if OBD_SIMULATION_MODE
00126     // Symulacja opóźnienia (jak rzeczywisty ELM327)
00127     vTaskDelay(600 / portTICK_PERIOD_MS);
00128     // Symulacja: prędkość 50 km/h = 28 metrów co 2 sekundy
00129         static long simulatedOdoKiloMeters = 50000; // 50tys przebiegu startowego
00130         static unsigned long lastIncMs = 0;
00131         unsigned long now = millis();
00132         if (lastIncMs == 0) lastIncMs = now;
00133         if (now - lastIncMs >= 36000) {
00134             simulatedOdoKiloMeters += 1; // +1 km co 36 sekundy (100 km/h)
00135             lastIncMs += 36000;
```

```
00136         }
00137         return simulatedOdoKiloMeters;
00138 #else
00139     char resp[64];
00140     if (!sendCmd(CarPID::ODOMETER, resp, sizeof(resp))) {
00141         return -1;
00142     }
00143
00144     // Wyszukanie prefiksu odpowiedzi
00145     char* p = strstr(resp, CarPID::ODOMETER_RESP_PREFIX);
00146     if (p != NULL && strlen(p) >= 12) {
00147
00148         // Wyciągnięcie 6 bajtów hex odpowiedzi
00149         char hexStr[7];
00150         strncpy(hexStr, p + 6, 6);
00151         hexStr[6] = '\0';
00152
00153         long kilometers = strtol(hexStr, NULL, 16);
00154
00155         if (kilometers >= 1000 && kilometers < 999999999) {
00156
00157             Serial.printf("[ODO] Odometer read: %ld KM\n", kilometers);
00158             return kilometers;
00159         }
00160     }
00161     return -1; // Błąd odczytu
00162
00163 #endif
00164 }
00165
00166 // Odczyt spalania przez MAF - zwraca L/h
00167 float readFuelRate() {
00168
00169 #if OBD_SIMULATION_MODE
00170     // Symulacja opóźnienia (jak rzeczywisty ELM327)
00171     vTaskDelay(500 / portTICK_PERIOD_MS);
00172     // Symulacja spalania: losowo od 0.8 do 10 L/h
00173     float minL = 0.8f, maxL = 10.0f;
00174     float rand01 = (float)rand() / (float)RAND_MAX;
00175     float fuel = minL + (maxL - minL) * rand01;
00176     return fuel;
00177
00178 #else
00179     char resp[64];
00180     if (!sendCmd(CarPID::MAF, resp, sizeof(resp))) {
00181         return -1.0f;
00182     }
00183
00184     // Szukanie prefiksu odpowiedzi
00185     char* p = strstr(resp, CarPID::MAF_RESP_PREFIX);
00186     if (p != NULL && strlen(p) >= 8) {
00187         // Wyciągnij 2 bajty hex
00188         char byteA[3], byteB[3];
00189         strncpy(byteA, p + 4, 2); byteA[2] = '\0';
00190         strncpy(byteB, p + 6, 2); byteB[2] = '\0';
00191
00192         int a = strtol(byteA, NULL, 16);
00193         int b = strtol(byteB, NULL, 16);
00194         float maf = ((a * 256) + b) / 100.0f;
00195
00196         float afr = 14.7f;
00197         float dens = 0.755f;
00198         float fuelRate = (maf / afr / dens) * 3.6f;
00199         Serial.printf("[FUEL] Fuel rate: %.2f L/H\n", fuelRate);
00200         return fuelRate;
00201     }
00202     return -1.0f; // Błąd odczytu
00203
00204 #endif
00205 }
00206
00207 // Obliczenie kosztu przejazdu
00208 float calculateCost() {
00209
00210     if (tariffMode == TARIFF_PER_KM)
00211         return distanceTraveled * tariffValue;
00212     else
00213         return fuelUsed * tariffValue;
00214
00215 }
00216
00217 // Task OBD uruchomiony w tle (FreeRTOS)
00218 void task(void* param) {
00219
00220     static float lastOdo = -1.0f;
00221     static unsigned long lastMillis = 0;
00222
```

```
00223     while (true) {
00224         if (tripActive) {
00225 
00226             // Odczyt odometru
00227             long odoRaw = readOdometer();
00228             float dist = (odoRaw >= 0) ? (float)odoRaw : -1.0f;
00229             Serial.printf("[OBD_TASK] odoRaw=%ld, dist=%.2f\n", odoRaw, dist);
00230 
00231             // Odczyt paliwa
00232             float fuel = readFuelRate();
00233             Serial.printf("[OBD_TASK] fuel=%.2f\n", fuel);
00234 
00235             // LICZENIE tylko gdy nie zapauzowany
00236             if (!tripPaused) {
00237 
00238                 unsigned long now = millis();
00239 
00240                 // Inicjalizacja przy pierwszym odczycie
00241                 if (lastMillis == 0 && dist >= 0 && fuel >= 0) {
00242 
00243                     lastMillis = now;
00244                     lastOdo = dist;
00245                 }
00246 
00247                 // Obliczanie przyrostów
00248                 else if (lastMillis > 0 && dist >= 0) {
00249 
00250                     float hours = (now - lastMillis) / 3600000.0f;
00251 
00252                     // Dystans
00253                     float deltaDist = max(0.0f, dist - lastOdo);
00254                     if (deltaDist > 0.0001f) {
00255                         distanceTraveled += deltaDist;
00256                     }
00257                     lastOdo = dist;
00258 
00259                     // Paliwo
00260                     if (fuel >= 0 && hours > 0) {
00261                         float fuelDelta = fuel * hours;
00262                         fuelUsed += fuelDelta;
00263                     }
00264 
00265                     lastMillis = now;
00266                 }
00267 
00268             } else {
00269 
00270                 // Gdy zapauzowany - reset timera żeby po wznowieniu nie było skoku
00271                 lastMillis = 0;
00272             }
00273 
00274         } else {
00275 
00276             // Reset przy nieaktywnym tripie
00277             lastOdo = -1.0f;
00278             lastMillis = 0;
00279         }
00280 
00281         vTaskDelay(2000 / portTICK_PERIOD_MS);  // Opóźnienie 2 sekundy
00282     }
00283 }
00284 
00285 } // namespace OBD
```

8.47 src/screen_about.cpp File Reference

```
#include "screen_about.h"
#include "screen_home.h"
#include "screen_manager.h"
#include "tft_display.h"
#include "background.h"
#include <Arduino.h>
```

Functions

- void initAboutScreen (TFT_eSPI *tft)

Inicjalizuje ekran "About" Rysuje tło i elementy interfejsu na ekranie informacji o aplikacji.

- void handleAboutTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie "About".

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgAbout = nullptr

8.47.1 Function Documentation

8.47.1.1 handleAboutTouch()

```
void handleAboutTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie "About".

Przetwarza interakcje użytkownika, np. powrót do ekranu głównego.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 28 of file screen_about.cpp.

8.47.1.2 initAboutScreen()

```
void initAboutScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran "About" Rysuje tło i elementy interfejsu na ekranie informacji o aplikacji.

Definition at line 12 of file screen_about.cpp.

8.47.2 Variable Documentation

8.47.2.1 bgAbout

```
Background* bgAbout = nullptr  [static]
```

Definition at line 9 of file screen_about.cpp.

8.47.2.2 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 8 of file screen_about.cpp.

8.48 screen_about.cpp

Go to the documentation of this file.

```
00001 #include "screen_about.h"
00002 #include "screen_home.h"
00003 #include "screen_manager.h"
00004 #include "tft_display.h"
00005 #include "background.h"
00006 #include <Arduino.h>
00007
00008 static TFT_eSPI* tftPtr = nullptr;
00009 static Background* bgAbout = nullptr;
00010
00011 // Inicjalizacja ekranu "About"
00012 void initAboutScreen(TFT_eSPI* tft) {
00013
00014     tftPtr = tft;
00015
00016     if (bgAbout) {
00017
00018         delete bgAbout;
00019         bgAbout = nullptr;
00020     }
00021
00022     bgAbout = new Background("/about.png");
00023     bgAbout->draw(*tft, *bgAbout->s_png, true);
00024     Serial.println("[SYSTEM] About screen initialized");
00025 }
00026
00027 // Obsługa dotyku na ekranie "About"
00028 void handleAboutTouch(uint16_t x, uint16_t y) {
00029
00030     // Przycisk powrotu (lewy górny róg)
00031     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00032
00033         initHomeScreen(tftPtr);
00034         currentScreen = SCREEN_HOME;
00035         return;
00036     }
00037 }
```

8.49 src/screen_brightness.cpp File Reference

```
#include "screen_brightness.h"
#include "gui_elements.h"
#include "tft_display.h"
#include "screen_settings.h"
#include "screen_manager.h"
#include "screen_home.h"
#include "background.h"
#include <Arduino.h>
#include <EEPROM.h>
```

Macros

- #define BRIGHTNESS_ADDR 0

Functions

- void initBrightnessModule ()

  Inicjalizuje moduł sterowania jasnością
- void drawBrightnessPercent (TFT_eSPI *tft, uint8_t brightness)
- void initBrightnessScreen (TFT_eSPI *tft)

  Inicjalizuje ekran ustawień jasności.
- void handleBrightnessTouch (TFT_eSPI *tft, uint16_t x, uint16_t y, uint8_t *levelOut)

  Obsługuje dotyk na ekranie ustawień jasności.

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgBrightness = nullptr
- static char lastText [8] = ""
- uint8_t brightnessLevel = 250

  Aktualny poziom jasności podświetlenia.

8.49.1 Macro Definition Documentation

8.49.1.1 BRIGHTNESS_ADDR

```
#define BRIGHTNESS_ADDR 0
```

Definition at line 11 of file screen_brightness.cpp.

8.49.2 Function Documentation

8.49.2.1 drawBrightnessPercent()

```
void drawBrightnessPercent (
            TFT_eSPI * tft,
            uint8_t brightness)
```

Definition at line 33 of file screen_brightness.cpp.

8.49.2.2 handleBrightnessTouch()

```
void handleBrightnessTouch (
            TFT_eSPI * tft,
            uint16_t x,
            uint16_t y,
            uint8_t * levelOut)
```

Obsługuje dotyk na ekranie ustawień jasności.

Przetwarza interakcje z suwakiem i przyciskami, aktualizuje jasność w czasie rzeczywistym.

Parameters

	tft	Wskaźnik do obiektu wyświetlacza TFT
	x	Współrzędna X punktu dotyku
	y	Współrzędna Y punktu dotyku
out	*levelOut*	Wskaźnik do zmiennej gdzie zostanie zapisany nowy poziom jasności

Definition at line 69 of file screen_brightness.cpp.

8.49.2.3 initBrightnessModule()

```
void initBrightnessModule ()
```

Inicjalizuje moduł sterowania jasnością

Konfiguruje PWM do kontroli podświetlenia i wczytuje zapisaną wartość z EEPROM. Należy wywołać raz podczas startu aplikacji.

Definition at line 20 of file screen_brightness.cpp.

8.49.2.4 initBrightnessScreen()

```
void initBrightnessScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran ustawień jasności.

Rysuje interfejs z suwakiem jasności i przyciskami kontrolnymi.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 47 of file screen_brightness.cpp.

8.49.3 Variable Documentation

8.49.3.1 bgBrightness

```
Background* bgBrightness = nullptr  [static]
```

Definition at line 14 of file screen_brightness.cpp.

8.49.3.2 brightnessLevel

```
uint8_t brightnessLevel = 250
```

Aktualny poziom jasności podświetlenia.

Wartość w zakresie 0-100 określająca procent maksymalnej jasności.

- 0 = podświetlenie wyłączone
- 100 = maksymalna jasność

Definition at line 17 of file screen_brightness.cpp.

8.49.3.3 lastText

```
char lastText[8] = ""  [static]
```

Definition at line 15 of file screen_brightness.cpp.

8.49.3.4 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 13 of file screen_brightness.cpp.

8.50 screen_brightness.cpp

Go to the documentation of this file.

```
00001 #include "screen_brightness.h"
00002 #include "gui_elements.h"
00003 #include "tft_display.h"
00004 #include "screen_settings.h"
00005 #include "screen_manager.h"
00006 #include "screen_home.h"
00007 #include "background.h"
00008 #include <Arduino.h>
00009 #include <EEPROM.h>
00010
00011 #define BRIGHTNESS_ADDR 0 // Adres w EEPROM do przechowywania poziomu jasności
00012
00013 static TFT_eSPI* tftPtr = nullptr;
00014 static Background* bgBrightness = nullptr;
00015 static char lastText[8] = "";
00016
00017 uint8_t brightnessLevel = 250;
00018
00019 // Funkcja ustawiająca jasność podświetlenia ekranu
00020 void initBrightnessModule() {
00021
00022     uint8_t saved = EEPROM.read(BRIGHTNESS_ADDR);
00023     brightnessLevel = saved;
00024
00025     if (brightnessLevel < 5) brightnessLevel = 255;
00026
00027     setBacklight(brightnessLevel);
00028     Serial.print("[BRIGHTNESS] Brightness module initialized with level: ");
00029     Serial.println(brightnessLevel);
00030 }
00031
00032 // Funkcja rysująca procentową jasność na ekranie
00033 void drawBrightnessPercent(TFT_eSPI* tft, uint8_t brightness) {
```

```
00034
00035     char buf[8];
00036     uint8_t percent = (uint8_t)((brightness * 100) / 255);
00037     sprintf(buf, "%d%%", percent);
00038
00039     if (strcmp(buf, lastText) != 0) {
00040
00041         drawTextWithBackground(tft, buf, 300, 75, TR_DATUM, 2, TFT_WHITE, TFT_BLACK, 0);
00042         strcpy(lastText, buf);
00043     }
00044 }
00045
00046 // Funkcja inicjalizująca ekran jasności
00047 void initBrightnessScreen(TFT_eSPI* tft) {
00048
00049     tftPtr = tft;
00050     if (bgBrightness) {
00051
00052         delete bgBrightness;
00053         bgBrightness = nullptr;
00054     }
00055
00056     bgBrightness = new Background("/brightness.png");
00057     bgBrightness->draw(*tft, *bgBrightness->s_png, true);
00058     setBacklight(brightnessLevel);
00059
00060     // Reset bufora tekstu jasności przy wejściu na ekran
00061     resetTextBuffer(lastText, sizeof(lastText));
00062     // Wyświetlanie aktualnej jasności w procentach
00063     drawBrightnessPercent(tftPtr, brightnessLevel);
00064
00065     Serial.println("[SYSTEM] Brightness screen initialized");
00066 }
00067
00068 // Funkcja główna obsługująca dotyk na ekranie jasności
00069 void handleBrightnessTouch(TFT_eSPI* tft, uint16_t x, uint16_t y, uint8_t* levelOut) {
00070
00071     // Przycisk -10
00072     if (x >= 10 && x < 150 && y >= 130 && y < 170) {
00073
00074         brightnessLevel = constrain(brightnessLevel - 10, 10, 255);
00075         setBacklight(brightnessLevel);
00076         drawBrightnessPercent(tftPtr, brightnessLevel);
00077         return;
00078     }
00079
00080     // Przycisk +10
00081     if (x >= 170 && x < 310 && y >= 130 && y < 170) {
00082
00083         brightnessLevel = constrain(brightnessLevel + 10, 10, 255);
00084         setBacklight(brightnessLevel);
00085         drawBrightnessPercent(tftPtr, brightnessLevel);
00086         return;
00087     }
00088
00089     // Powrót do ekranu głównego (górny prawy róg)
00090     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00091
00092         EEPROM.write(BRIGHTNESS_ADDR, brightnessLevel);
00093         EEPROM.commit();
00094         Serial.print("[BRIGHTNESS] Brightness saved to EEPROM: ");
00095         Serial.println(brightnessLevel);
00096         initHomeScreen(tftPtr);
00097         currentScreen = SCREEN_HOME;
00098         return;
00099     }
00100 }
```

8.51 src/screen_connection.cpp File Reference

```
#include "screen_connection.h"
#include "screen_manager.h"
#include "background.h"
#include "screen_home.h"
#include "screen_gps.h"
#include "screen_about.h"
#include "screen_obd.h"
#include "screen_obd-debug.h"
```

```
#include <Arduino.h>
```

Functions

- void initConnectionScreen (TFT_eSPI *tft)

  Inicjalizuje i wyświetla ekran połączeń
- void handleConnectionTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie połączeń

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgConnection = nullptr

8.51.1 Function Documentation

8.51.1.1 handleConnectionTouch()

```
void handleConnectionTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie połączeń

Przetwarza wybór przycisków i nawigację między ekranami.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 31 of file screen_connection.cpp.

8.51.1.2 initConnectionScreen()

```
void initConnectionScreen (
            TFT_eSPI * tft)
```

Inicjalizuje i wyświetla ekran połączeń

Rysuje przyciski do konfiguracji połączeń oraz opcję powrotu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 15 of file screen_connection.cpp.

8.51.2 Variable Documentation

8.51.2.1 bgConnection

```
Background* bgConnection = nullptr  [static]
```

Definition at line 12 of file screen_connection.cpp.

8.51.2.2 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 11 of file screen_connection.cpp.

8.52 screen_connection.cpp

Go to the documentation of this file.

```
00001 #include "screen_connection.h"
00002 #include "screen_manager.h"
00003 #include "background.h"
00004 #include "screen_home.h"
00005 #include "screen_gps.h"
00006 #include "screen_about.h"
00007 #include "screen_obd.h"
00008 #include "screen_obd-debug.h"
00009 #include <Arduino.h>
00010
00011 static TFT_eSPI* tftPtr = nullptr;
00012 static Background* bgConnection = nullptr;
00013
00014 // Funkcja inicjalizująca ekran połączeń
00015 void initConnectionScreen(TFT_eSPI* tft) {
00016
00017     tftPtr = tft;
00018     if (bgConnection) {
00019
00020         delete bgConnection;
00021         bgConnection = nullptr;
00022     }
00023
00024     bgConnection = new Background("/connection.png");
00025     bgConnection->draw(*tft, *bgConnection->s_png, true);
00026
00027     Serial.println("[SYSTEM] Connection screen initialized");
00028 }
00029
00030 // Funkcja główna obsługująca dotyk na ekranie połączeń
00031 void handleConnectionTouch(uint16_t x, uint16_t y) {
00032
00033     // Przycisk 1: GPS INFO
00034     if (x >= 10 && x < 310 && y >= 60 && y < 100) {
00035
00036         initGpsScreen(tftPtr);
00037         currentScreen = SCREEN_GPS;
00038         return;
00039     }
00040
00041     // Przycisk 2: OBDI INFO
00042     if (x >= 10 && x < 310 && y >= 120 && y < 160) {
00043
00044         initObdScreen(tftPtr);
00045         currentScreen = SCREEN_OBD;
00046         return;
00047     }
00048
00049     // Przycisk 3: ABOUT
00050     if (x >= 10 && x < 310 && y >= 180 && y < 220) {
00051
00052         initAboutScreen(tftPtr);
00053         currentScreen = SCREEN_ABOUT;
```

```
00054         return;
00055     }
00056
00057     // Powrót do ekranu głównego (górny prawy róg)
00058     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00059
00060         initHomeScreen(tftPtr);
00061         currentScreen = SCREEN_HOME;
00062     }
00063 }
```

8.53 src/screen_gps-debug.cpp File Reference

```
#include "screen_gps-debug.h"
#include "gps_reader.h"
#include "screen_manager.h"
#include "screen_home.h"
#include "screen_gps.h"
#include <Arduino.h>
```

Functions

- void initGpsDebugScreen (TFT_eSPI *tft)

  Inicjalizuje ekran debugowania GPS.
- void updateGpsDebugScreen (TFT_eSPI *tft)

  Aktualizuje dane na ekranie debugowania GPS.
- void handleGpsDebugTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie debugowania GPS.

Variables

- static TFT_eSPI * tftPtr = nullptr
- static char lastLatText [32] = ""
- static char lastLonText [32] = ""
- static char lastSatText [32] = ""

8.53.1 Function Documentation

8.53.1.1 handleGpsDebugTouch()

```
void handleGpsDebugTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie debugowania GPS.

Przetwarza interakcje, np. powrót do poprzedniego ekranu.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 83 of file screen_gps-debug.cpp.

8.53.1.2 initGpsDebugScreen()

```
void initGpsDebugScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran debugowania GPS.

Rysuje layout ekranu z polami na dane GPS i status połączenia.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 13 of file screen_gps-debug.cpp.

8.53.1.3 updateGpsDebugScreen()

```
void updateGpsDebugScreen (
            TFT_eSPI * tft)
```

Aktualizuje dane na ekranie debugowania GPS.

Odświeża wyświetlane wartości pozycji, liczby satelitów, HDOP oraz inne parametry diagnostyczne.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywoływać cyklicznie dla aktualnych danych

Definition at line 38 of file screen_gps-debug.cpp.

8.53.2 Variable Documentation

8.53.2.1 lastLatText

```
char lastLatText[32] = ""  [static]
```

Definition at line 9 of file screen_gps-debug.cpp.

8.53.2.2 lastLonText

```
char lastLonText[32] = ""  [static]
```

Definition at line 10 of file screen_gps-debug.cpp.

8.53.2.3 lastSatText

```
char lastSatText[32] = ""  [static]
```

Definition at line 11 of file screen_gps-debug.cpp.

8.53.2.4 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 8 of file screen_gps-debug.cpp.

8.54 screen_gps-debug.cpp

Go to the documentation of this file.

```
00001 #include "screen_gps-debug.h"
00002 #include "gps_reader.h"
00003 #include "screen_manager.h"
00004 #include "screen_home.h"
00005 #include "screen_gps.h"
00006 #include <Arduino.h>
00007
00008 static TFT_eSPI* tftPtr = nullptr;
00009 static char lastLatText[32] = "";
00010 static char lastLonText[32] = "";
00011 static char lastSatText[32] = "";
00012
00013 void initGpsDebugScreen(TFT_eSPI* tft) {
00014
00015     tftPtr = tft;
00016     tft->fillScreen(TFT_BLACK);
00017     // Tytuł
00018     tft->setTextColor(TFT_CYAN, TFT_BLACK);
00019     tft->setTextDatum(TC_DATUM);
00020     tft->drawString("GPS DIAGNOSTICS", 160, 10, 4);
00021     // Opisy pól
00022     tft->setTextColor(TFT_GREEN, TFT_BLACK);
00023     tft->setTextDatum(TL_DATUM);
00024     tft->drawString("Latitude:", 10, 60, 2);
00025     tft->drawString("Longitude:", 10, 100, 2);
00026     tft->drawString("Satellites:", 10, 140, 2);
00027     // Przycisk powrotu
00028     tft->fillRect(10, 200, 300, 40, TFT_DARKGREY);
00029     tft->setTextColor(TFT_WHITE, TFT_DARKGREY);
00030     tft->setTextDatum(MC_DATUM);
00031     tft->drawString("BACK", 160, 220, 2);
00032     strcpy(lastLatText, "");
00033     strcpy(lastLonText, "");
00034     strcpy(lastSatText, "");
00035     Serial.println("[SYSTEM] GPS-DEBUG screen initialized");
00036 }
00037
00038 void updateGpsDebugScreen(TFT_eSPI* tft) {
00039
00040     if (!tft) return;
00041
00042     // Aktualizacja danych GPS
00043     GPS::poll(GPS::lastFix);
00044     char latText[32];
00045     char lonText[32];
00046     char satText[32];
```

```
00047
00048     if (GPS::lastFix.valid) {
00049
00050         snprintf(latText, sizeof(latText), "%f", GPS::lastFix.lat);
00051         snprintf(lonText, sizeof(lonText), "%f", GPS::lastFix.lng);
00052         snprintf(satText, sizeof(satText), "%d", GPS::lastFix.sats);
00053     } else {
00054
00055         strcpy(latText, "N/A");
00056         strcpy(lonText, "N/A");
00057         strcpy(satText, "N/A");
00058     }
00059
00060     if (strcmp(latText, lastLatText) != 0) {
00061
00062         tft->setTextColor(TFT_WHITE, TFT_BLACK);
00063         tft->fillRect(120, 60, 180, 24, TFT_BLACK);
00064         tft->drawString(latText, 120, 65, 2);
00065         strcpy(lastLatText, latText);
00066     }
00067     if (strcmp(lonText, lastLonText) != 0) {
00068
00069         tft->setTextColor(TFT_WHITE, TFT_BLACK);
00070         tft->fillRect(120, 100, 180, 24, TFT_BLACK);
00071         tft->drawString(lonText, 120, 105, 2);
00072         strcpy(lastLonText, lonText);
00073     }
00074     if (strcmp(satText, lastSatText) != 0) {
00075
00076         tft->setTextColor(TFT_WHITE, TFT_BLACK);
00077         tft->fillRect(120, 140, 180, 24, TFT_BLACK);
00078         tft->drawString(satText, 120, 145, 2);
00079         strcpy(lastSatText, satText);
00080     }
00081 }
00082
00083 void handleGpsDebugTouch(uint16_t x, uint16_t y) {
00084
00085     if (x >= 10 && x <= 310 && y >= 200 && y <= 240) {
00086
00087         initGpsScreen(tftPtr);
00088         currentScreen = SCREEN_GPS;
00089         return;
00090     }
00091 }
```

8.55 src/screen_gps.cpp File Reference

```
#include "screen_gps.h"
#include "screen_manager.h"
#include "tft_display.h"
#include "background.h"
#include "screen_home.h"
#include "gui_elements.h"
#include "gps_reader.h"
#include "screen_gps-debug.h"
#include <Arduino.h>
```

Functions

- void initGpsScreen (TFT_eSPI *tft)

  Inicjalizuje ekran GPS.
- void handleGpsTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie GPS.

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgGps = nullptr

8.55.1 Function Documentation

8.55.1.1 handleGpsTouch()

```
void handleGpsTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie GPS.

Przetwarza naciśnięcia przycisków powrotu i resetu GPS.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 29 of file screen_gps.cpp.

8.55.1.2 initGpsScreen()

```
void initGpsScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran GPS.

Rysuje przyciski sterujące oraz obszar wyświetlania logów GPS.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 16 of file screen_gps.cpp.

8.55.2 Variable Documentation

8.55.2.1 bgGps

```
Background* bgGps = nullptr  [static]
```

Definition at line 13 of file screen_gps.cpp.

8.55.2.2 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 12 of file screen_gps.cpp.

8.56 screen_gps.cpp

Go to the documentation of this file.

```
00001
00002 #include "screen_gps.h"
00003 #include "screen_manager.h"
00004 #include "tft_display.h"
00005 #include "background.h"
00006 #include "screen_home.h"
00007 #include "gui_elements.h"
00008 #include "gps_reader.h"
00009 #include "screen_gps-debug.h"
00010 #include <Arduino.h>
00011
00012 static TFT_eSPI* tftPtr = nullptr;
00013 static Background* bgGps = nullptr;
00014
00015 // Inicjalizacja ekranu GPS
00016 void initGpsScreen(TFT_eSPI* tft) {
00017
00018     tftPtr = tft;
00019     if (bgGps) {
00020         delete bgGps;
00021         bgGps = nullptr;
00022     }
00023     bgGps = new Background("/gps.png");
00024     bgGps->draw(*tft, *bgGps->s_png, true);
00025     Serial.println("[SYSTEM] GPS screen initialized");
00026 }
00027
00028 // Obsługa dotyku na ekranie GPS
00029 void handleGpsTouch(uint16_t x, uint16_t y) {
00030
00031     // Debugowanie modułu GPS
00032     if (x >= 10 && x < 310 && y >= 180 && y < 220) {
00033
00034         initGpsDebugScreen(tftPtr);
00035         currentScreen = SCREEN_GPS_DEBUG;
00036         return;
00037     }
00038
00039     // Powrót do ekranu głównego (górny prawy róg)
00040     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00041
00042         initHomeScreen(tftPtr);
00043         currentScreen = SCREEN_HOME;
00044         return;
00045     }
00046 }
```

8.57 src/screen_home.cpp File Reference

```
#include "screen_home.h"
#include "gps_reader.h"
#include "obd_reader.h"
#include "screen_settings.h"
#include "screen_manager.h"
#include "screen_gps-debug.h"
#include "tft_display.h"
#include "background.h"
#include "gui_elements.h"
#include "screen_tariff.h"
#include "screen_trip.h"
#include <Arduino.h>
```

Functions

- void initHomeScreen (TFT_eSPI *tft)

Inicjalizuje ekran główny aplikacji.

- void updateGPSStatus (TFT_eSPI *tft)

  Aktualizuje widget statusu GPS na ekranie głównym.

- void handleHomeTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie głównym.

Variables

- static char lastGpsText [64] = ""
- static char lastObdText [32] = ""
- static TFT_eSPI * tftPtr = nullptr
- static Background * bgHome = nullptr

8.57.1 Function Documentation

8.57.1.1 handleHomeTouch()

```
void handleHomeTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie głównym.

Przetwarza naciśnięcia przycisków i uruchamia odpowiednie akcje.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 139 of file screen_home.cpp.

8.57.1.2 initHomeScreen()

```
void initHomeScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran główny aplikacji.

Rysuje główny interfejs z przyciskami menu i widgetami statusu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 32 of file screen_home.cpp.

8.57.1.3 updateGPSStatus()

```
void updateGPSStatus (
            TFT_eSPI * tft)
```

Aktualizuje widget statusu GPS na ekranie głównym.

Odświeża ikony i informacje o połączeniu GPS bez przerysowywania całego ekranu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywoływać przy zmianie statusu GPS

Definition at line 72 of file screen_home.cpp.

8.57.2 Variable Documentation

8.57.2.1 bgHome

```
Background* bgHome = nullptr  [static]
```

Definition at line 24 of file screen_home.cpp.

8.57.2.2 lastGpsText

```
char lastGpsText[64] = ""  [static]
```

Definition at line 18 of file screen_home.cpp.

8.57.2.3 lastObdText

```
char lastObdText[32] = ""  [static]
```

Definition at line 19 of file screen_home.cpp.

8.57.2.4 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 23 of file screen_home.cpp.

8.58 screen_home.cpp

Go to the documentation of this file.

```
00001 #include "screen_home.h"
00002 #include "gps_reader.h"
00003 #include "obd_reader.h"
00004 #include "screen_settings.h"
00005 #include "screen_manager.h"
00006 #include "screen_gps-debug.h"
00007 #include "tft_display.h"
00008 #include "background.h"
00009 #include "gui_elements.h"
00010 #include "screen_tariff.h"
00011 #include "screen_trip.h"
00012 #include <Arduino.h>
00013
00014 // ============================================================================
00015 // GLOBALNE ZMIENNE - Deklaracje zmiennych globalnych używanych na ekranie głównym
00016
00017 // Bufor tekstu GPS i OBD statusu
00018 static char lastGpsText[64] = "";
00019 static char lastObdText[32] = "";
00020
00021 // ============================================================================
00022
00023 static TFT_eSPI* tftPtr = nullptr;
00024 static Background* bgHome = nullptr;
00025
00026 // ============================================================================
00027 // FUNKCJE EKRANU GŁÓWNEGO - Inicjalizacja, aktualizacja GPS i obsługa dotyku
00028 // Timer do ograniczania odświeżania statusu GPS
00029 // ============================================================================
00030
00031 // Funkcja inicjalizująca ekran główny
00032 void initHomeScreen(TFT_eSPI* tft) {
00033
00034     tftPtr = tft;
00035     if (bgHome) {
00036
00037         delete bgHome;
00038         bgHome = nullptr;
00039     }
00040
00041     if(tripActive)
00042         bgHome = new Background("/home_resume.png");
00043     else if(OBD::btConnected && OBD::elmReady)
00044         bgHome = new Background("/home_active.png");
00045     else
00046         bgHome = new Background("/home.png");
00047
00048     // Rysowanie tła ekranu głównego
00049     bgHome->draw(*tft, *bgHome->s_png, true);
00050
00051     // Reset bufora tekstu GPS statusu przy wejściu na ekran
00052     resetTextBuffer(lastGpsText, sizeof(lastGpsText));
00053     resetTextBuffer(lastObdText, sizeof(lastObdText));
00054
00055     // Wyświetlanie taryfy w górnej części ekranu
00056     char tariffBuf[32];
00057     if (tariffMode == TARIFF_PER_KM) {
00058
00059         sprintf(tariffBuf, "%.2f ZL/KM", tariffValue);
00060         drawTextWithBackground(tft, tariffBuf, 300, 70, TR_DATUM, 2, TFT_YELLOW, TFT_BLACK, 60);
00061
00062     } else {
00063
00064         sprintf(tariffBuf, "%.2f ZL/L", tariffValue);
00065         drawTextWithBackground(tft, tariffBuf, 300, 70, TR_DATUM, 2, TFT_CYAN, TFT_BLACK, 60);
00066     }
00067
00068     Serial.println("[SYSTEM] Home screen initialized");
00069 }
00070
00071 // Wyświetlanie statusu GPS i OBD na ekranie głównym
00072 void updateGPSStatus(TFT_eSPI* tft) {
00073
00074     if (!tft) return;
00075     GPS::poll(GPS::lastFix); // Pobierz najnowszą próbkę GPS
00076
00077     // Wyświetlanie uproszczonego statusu GPS
00078     char gpsBuf[16];
00079     uint16_t gpsColor = TFT_RED;
00080     if (!GPS::lastFix.valid) {
00081
00082         strcpy(gpsBuf, "NO FIX");
```

```
00083         gpsColor = TFT_RED;
00084
00085     } else if (GPS::lastFix.sats < 5) {
00086
00087         strcpy(gpsBuf, "WEAK");
00088         gpsColor = TFT_ORANGE;
00089
00090     } else {
00091
00092         strcpy(gpsBuf, "STRONG");
00093         gpsColor = TFT_GREEN;
00094     }
00095
00096     // Wyświetlanie statusu OBD
00097     char obdBuf[32];
00098     bool forceRedraw = false;
00099
00100     if (OBD::btConnected && OBD::elmReady)
00101         strcpy(obdBuf, "CONNECTED");
00102     else if (OBD::btConnected)
00103         strcpy(obdBuf, "BT ONLY");
00104     else
00105         strcpy(obdBuf, "NO LINK");
00106
00107     // Mechanizm podmiany tła przy zmianie statusu OBD
00108     static bool lastObdWasConnected = false;
00109     bool nowConnected = (OBD::btConnected && OBD::elmReady);
00110     static String lastBgPath = "";
00111     const char* desiredBg = nowConnected ? "/home_active.png" : "/home.png";
00112
00113     if (lastBgPath != desiredBg) {
00114
00115         if (bgHome) { delete bgHome; bgHome = nullptr; }
00116         bgHome = new Background(desiredBg);
00117         bgHome->draw(*tft, *bgHome->s_png, true);
00118         lastBgPath = desiredBg;
00119         forceRedraw = true;
00120     }
00121     lastObdWasConnected = nowConnected;
00122
00123     // Rysowanie napisów tylko jeśli się zmieniły lub wymuszone GPS
00124     if (strcmp(gpsBuf, lastGpsText) != 0 || forceRedraw) {
00125
00126         drawTextWithBackground(tft, gpsBuf, 300, 110, TR_DATUM, 2, gpsColor, TFT_BLACK, 0);
00127         strcpy(lastGpsText, gpsBuf);
00128     }
00129
00130     // Rysowanie napisów tylko jeśli się zmieniły lub wymuszone OBD
00131     if (strcmp(obdBuf, lastObdText) != 0 || forceRedraw) {
00132
00133         drawTextWithBackground(tft, obdBuf, 300, 130, TR_DATUM, 2, OBD::btConnected ? TFT_GREEN :
      TFT_RED, TFT_BLACK, 0);
00134         strcpy(lastObdText, obdBuf);
00135     }
00136 }
00137
00138 // Obsługa dotyku na ekranie głównym
00139 void handleHomeTouch(uint16_t x, uint16_t y) {
00140
00141   // Przejście do ustawień (górny prawy róg)
00142   if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00143
00144     initSettingsScreen(tftPtr);
00145     currentScreen = SCREEN_SETTINGS;
00146     return;
00147   }
00148
00149   // Rozpoczęcie jazdy (dolny prostokąt) tylko jeśli OBD połączone
00150   if (x >= 20 && x < 220 && y >= 200 && y < 240) {
00151
00152     if (OBD::btConnected && OBD::elmReady) {
00153
00154         initTripScreen(tftPtr);
00155         currentScreen = SCREEN_TRIP;
00156
00157     } else {
00158         Serial.println("[ERROR] Cannot start trip if OBD is not connected");
00159     }
00160     return;
00161   }
00162 }
```

8.59 src/screen_manager.cpp File Reference

```
#include "screen_manager.h"
```

Variables

- ScreenState currentScreen = SCREEN_WELCOME

  Aktualny stan ekranu aplikacji.

8.59.1 Variable Documentation

8.59.1.1 currentScreen

```
ScreenState currentScreen = SCREEN_WELCOME
```

Aktualny stan ekranu aplikacji.

Zmienna globalna określająca który ekran jest obecnie wyświetlany. Używana w głównej pętli do routowania zdarzeń dotyku.

Note

Modyfikowana przez funkcje handleXxxTouch() przy zmianie ekranu

Definition at line 9 of file screen_manager.cpp.

8.60 screen_manager.cpp

Go to the documentation of this file.

```
00001 #include "screen_manager.h"
00002
00003 // ================================================================================
00004 // SCREEN MANAGER - Implementacja
00005 // ================================================================================
00006
00007 // Definicja globalnej zmiennej stanu ekranu
00008 // Domyślnie ustawiony na ekran powitalny (zmieniane w setup() w main.cpp)
00009 ScreenState currentScreen = SCREEN_WELCOME;
```

8.61 src/screen_obd-debug.cpp File Reference

```
#include "screen_obd-debug.h"
#include "screen_obd.h"
#include "screen_manager.h"
#include "obd_reader.h"
#include "screen_home.h"
#include <Arduino.h>
```

Functions

- void initObdDebugScreen (TFT_eSPI *tft)

  Inicjalizuje ekran debugowania OBD.
- void updateObdDebugScreen (TFT_eSPI *tft)

  Aktualizuje dane na ekranie debugowania OBD.
- void handleObdDebugTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie debugowania OBD.

Variables

- static TFT_eSPI * tftPtr = nullptr
- static char lastOdoText [32] = ""
- static char lastFuelText [32] = ""

8.61.1 Function Documentation

8.61.1.1 handleObdDebugTouch()

```
void handleObdDebugTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie debugowania OBD.

Przetwarza interakcje, np. powrót do poprzedniego ekranu.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 82 of file screen_obd-debug.cpp.

8.61.1.2 initObdDebugScreen()

```
void initObdDebugScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran debugowania OBD.

Rysuje layout z polami na dane diagnostyczne pojazdu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 12 of file screen_obd-debug.cpp.

8.61.1.3 updateObdDebugScreen()

```
void updateObdDebugScreen (
            TFT_eSPI * tft)
```

Aktualizuje dane na ekranie debugowania OBD.

Odświeża wyświetlane parametry pojazdu (obroty, temperatura, przepływ powietrza).

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywoływać cyklicznie dla aktualnych danych

Definition at line 41 of file screen_obd-debug.cpp.

8.61.2 Variable Documentation

8.61.2.1 lastFuelText

```
char lastFuelText[32] = ""  [static]
```

Definition at line 10 of file screen_obd-debug.cpp.

8.61.2.2 lastOdoText

```
char lastOdoText[32] = ""  [static]
```

Definition at line 9 of file screen_obd-debug.cpp.

8.61.2.3 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 8 of file screen_obd-debug.cpp.

8.62 screen_obd-debug.cpp

Go to the documentation of this file.

```
00001 #include "screen_obd-debug.h"
00002 #include "screen_obd.h"
00003 #include "screen_manager.h"
00004 #include "obd_reader.h"
00005 #include "screen_home.h"
00006 #include <Arduino.h>
00007
00008 static TFT_eSPI* tftPtr = nullptr;
00009 static char lastOdoText[32] = "";
00010 static char lastFuelText[32] = "";
00011
00012 void initObdDebugScreen(TFT_eSPI* tft) {
00013
00014     tftPtr = tft;
00015     tft->fillScreen(TFT_BLACK);
00016
00017     // Tytuł
00018     tft->setTextColor(TFT_CYAN, TFT_BLACK);
00019     tft->setTextDatum(TC_DATUM);
00020     tft->drawString("OBD DIAGNOSTICS", 160, 10, 4);
00021
00022     // Opisy pól
00023     tft->setTextColor(TFT_GREEN, TFT_BLACK);
00024     tft->setTextDatum(TL_DATUM);
```

```
00025     tft->drawString("Odometer:", 10, 60, 2);
00026     tft->setTextColor(TFT_YELLOW, TFT_BLACK);
00027     tft->drawString("Fuel Rate:", 10, 120, 2);
00028
00029     // Przycisk powrotu
00030     tft->fillRect(10, 200, 300, 40, TFT_DARKGREY);
00031     tft->setTextColor(TFT_WHITE, TFT_DARKGREY);
00032     tft->setTextDatum(MC_DATUM);
00033     tft->drawString("BACK", 160, 220, 2);
00034
00035     strcpy(lastOdoText, "");
00036     strcpy(lastFuelText, "");
00037
00038     Serial.println("[SYSTEM] OBD-DEBUG screen initialized");
00039 }
00040
00041 void updateObdDebugScreen(TFT_eSPI* tft) {
00042
00043     if (!tft) return;
00044
00045     // Odczyt danych
00046     long odo = OBD::readOdometer();
00047     float fuel = OBD::readFuelRate();
00048
00049     char odoText[32];
00050     char fuelText[32];
00051
00052     if (odo > 0)
00053         sprintf(odoText, "%ld KM", odo);
00054     else
00055         sprintf(odoText, "N/A");
00056
00057     if (fuel >= 0)
00058         sprintf(fuelText, "%.2f L/H", fuel);
00059     else
00060         sprintf(fuelText, "N/A");
00061
00062     // Update tylko przy zmianie
00063     if (strcmp(odoText, lastOdoText) != 0) {
00064
00065         tft->fillRect(10, 85, 300, 30, TFT_BLACK);
00066         tft->setTextColor(TFT_WHITE, TFT_BLACK);
00067         tft->setTextDatum(TL_DATUM);
00068         tft->drawString(odoText, 10, 85, 4);
00069         strcpy(lastOdoText, odoText);
00070     }
00071
00072     if (strcmp(fuelText, lastFuelText) != 0) {
00073
00074         tft->fillRect(10, 145, 300, 30, TFT_BLACK);
00075         tft->setTextColor(TFT_WHITE, TFT_BLACK);
00076         tft->setTextDatum(TL_DATUM);
00077         tft->drawString(fuelText, 10, 145, 4);
00078         strcpy(lastFuelText, fuelText);
00079     }
00080 }
00081
00082 void handleObdDebugTouch(uint16_t x, uint16_t y) {
00083
00084     // Przycisk powrotu
00085     if (x >= 10 && x < 310 && y >= 200 && y < 240) {
00086
00087         initObdScreen(tftPtr);
00088         currentScreen = SCREEN_OBD;
00089         return;
00090     }
00091 }
```

8.63 src/screen_obd.cpp File Reference

```
#include "screen_obd.h"
#include "screen_obd-debug.h"
#include "screen_manager.h"
#include "tft_display.h"
#include "background.h"
#include "screen_home.h"
#include "gui_elements.h"
#include "gps_reader.h"
```

```
#include <Arduino.h>
```

Functions

- void initObdScreen (TFT_eSPI *tft)

  Inicjalizuje ekran OBD.
- void handleObdTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie OBD.

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgGps = nullptr

8.63.1 Function Documentation

8.63.1.1 handleObdTouch()

```
void handleObdTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie OBD.

Przetwarza naciśnięcia przycisków i nawigację.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 30 of file screen_obd.cpp.

8.63.1.2 initObdScreen()

```
void initObdScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran OBD.

Rysuje przyciski sterujące oraz obszar wyświetlania statusu OBD.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 15 of file screen_obd.cpp.

8.63.2 Variable Documentation

8.63.2.1 bgGps

```
Background* bgGps = nullptr  [static]
```

Definition at line 13 of file screen_obd.cpp.

8.63.2.2 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 12 of file screen_obd.cpp.

8.64 screen_obd.cpp

Go to the documentation of this file.

```
00001
00002 #include "screen_obd.h"
00003 #include "screen_obd-debug.h"
00004 #include "screen_manager.h"
00005 #include "tft_display.h"
00006 #include "background.h"
00007 #include "screen_home.h"
00008 #include "gui_elements.h"
00009 #include "gps_reader.h"
00010 #include <Arduino.h>
00011
00012 static TFT_eSPI* tftPtr = nullptr;
00013 static Background* bgGps = nullptr;
00014
00015 void initObdScreen(TFT_eSPI* tft) {
00016
00017     tftPtr = tft;
00018
00019     if (bgGps) {
00020
00021         delete bgGps;
00022         bgGps = nullptr;
00023     }
00024
00025     bgGps = new Background("/obd.png");
00026     bgGps->draw(*tft, *bgGps->s_png, true);
00027     Serial.println("[SYSTEM] OBD screen initialized");
00028 }
00029
00030 void handleObdTouch(uint16_t x, uint16_t y) {
00031
00032     // Włączenie ekranu debugowania OBD
00033     if (x >= 10 && x < 310 && y >= 180 && y < 220) {
00034
00035         initObdDebugScreen(tftPtr);
00036         currentScreen = SCREEN_OBD_DEBUG;
00037         return;
00038     }
00039
00040     // Powrót do ekranu głównego
00041     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00042
00043         initHomeScreen(tftPtr);
00044         currentScreen = SCREEN_HOME;
00045         return;
00046     }
00047 }
00048
```

8.65 src/screen_settings.cpp File Reference

```
#include "screen_settings.h"
#include "screen_manager.h"
#include "screen_home.h"
#include "screen_brightness.h"
#include "screen_connection.h"
#include "background.h"
#include "gui_elements.h"
#include "screen_tariff.h"
#include <Arduino.h>
```

Functions

- void initSettingsScreen (TFT_eSPI *tft)
  Inicjalizuje ekran ustawień
- void handleSettingsTouch (uint16_t x, uint16_t y)
  Obsługuje dotyk na ekranie ustawień

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgSettings = nullptr

8.65.1 Function Documentation

8.65.1.1 handleSettingsTouch()

```
void handleSettingsTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie ustawień

Przetwarza wybór opcji i przejścia do ekranów szczegółowych ustawień.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 38 of file screen_settings.cpp.

8.65.1.2 initSettingsScreen()

```
void initSettingsScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran ustawień

Rysuje menu z opcjami konfiguracji taryfy, jasności i połączeń.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 23 of file screen_settings.cpp.

8.65.2 Variable Documentation

8.65.2.1 bgSettings

```
Background* bgSettings = nullptr  [static]
```

Definition at line 16 of file screen_settings.cpp.

8.65.2.2 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 15 of file screen_settings.cpp.

8.66 screen_settings.cpp

Go to the documentation of this file.

```
00001 #include "screen_settings.h"
00002 #include "screen_manager.h"
00003 #include "screen_home.h"
00004 #include "screen_brightness.h"
00005 #include "screen_connection.h"
00006 #include "background.h"
00007 #include "gui_elements.h"
00008 #include "screen_tariff.h"
00009 #include <Arduino.h>
00010
00011 // ============================================================================
00012 // GLOBALNE ZMIENNE - Przyciski na ekranie ustawień
00013 // ============================================================================
00014
00015 static TFT_eSPI* tftPtr = nullptr;
00016 static Background* bgSettings = nullptr;
00017
00018 // ============================================================================
00019 // FUNKCJE EKRANU USTAWIEŃ - Inicjalizacja i obsługa dotyku
00020 // ============================================================================
00021
00022 // Funkcja inicjalizująca ekran ustawień
00023 void initSettingsScreen(TFT_eSPI* tft) {
00024
00025   tftPtr = tft;
00026   if (bgSettings) {
00027       delete bgSettings;
00028       bgSettings = nullptr;
00029   }
00030
00031   bgSettings = new Background("/settings.png");
00032   bgSettings->draw(*tft, *bgSettings->s_png, true);
00033
00034   Serial.println("[SYSTEM] Settings screen initialized");
00035 }
00036
00037 // Funkcja główna obsługująca dotyk na ekranie ustawień
00038 void handleSettingsTouch(uint16_t x, uint16_t y) {
00039
00040   // Powrót do ekranu głównego (górny prawy róg)
00041   if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00042
00043       initHomeScreen(tftPtr);
```

```
00044       currentScreen = SCREEN_HOME;
00045       return;
00046   }
00047
00048   //przejście do ekranu taryfy (górny przycisk)
00049     if (x >= 10 && x < 310 && y >= 60 && y < 100) {
00050
00051       initTariffScreen(tftPtr);
00052       currentScreen = SCREEN_TARIFF;
00053       return;
00054   }
00055
00056   // Przejście do ekranu jasności (środkowy przycisk)
00057     if (x >= 10 && x < 310 && y >= 120 && y < 160) {
00058
00059       initBrightnessScreen(tftPtr);
00060       currentScreen = SCREEN_BRIGHTNESS;
00061       return;
00062   }
00063
00064   // Przejście do ekranu połączeń (dolny przycisk)
00065     if (x >= 10 && x < 310 && y >= 180 && y < 220) {
00066
00067       initConnectionScreen(tftPtr);
00068       currentScreen = SCREEN_CONNECTION;
00069       return;
00070   }
00071
00072 }
```

8.67 src/screen_tariff.cpp File Reference

```
#include "screen_tariff.h"
#include "background.h"
#include "gui_elements.h"
#include "screen_manager.h"
#include "screen_home.h"
#include <EEPROM.h>
#include <Arduino.h>
```

Macros

- #define EEPROM_TARIFF_VALUE_ADDR 10
- #define EEPROM_TARIFF_MODE_ADDR 14

Functions

- void loadTariffFromEEPROM ()

  Ładuje ustawienia taryfy z EEPROM.
- void saveTariffToEEPROM ()

  Zapisuje ustawienia taryfy do EEPROM.
- void initTariffScreen (TFT_eSPI *tft)

  Inicjalizuje ekran konfiguracji taryfy.
- void drawTariffValue (TFT_eSPI *tft)

  Rysuje aktualną wartość taryfy na ekranie.
- void handleTariffTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie taryfy.

Variables

- static TFT_eSPI * tftPtr = nullptr
- static Background * bgTariff = nullptr
- float tariffValue = 3.00f

  Wartość taryfy w aktualnym trybie.
- TariffMode tariffMode = TARIFF_PER_KM

  Aktualny tryb taryfy.

8.67.1 Macro Definition Documentation

8.67.1.1 EEPROM_TARIFF_MODE_ADDR

```
#define EEPROM_TARIFF_MODE_ADDR 14
```

Definition at line 14 of file screen_tariff.cpp.

8.67.1.2 EEPROM_TARIFF_VALUE_ADDR

```
#define EEPROM_TARIFF_VALUE_ADDR 10
```

Definition at line 13 of file screen_tariff.cpp.

8.67.2 Function Documentation

8.67.2.1 drawTariffValue()

```
void drawTariffValue (
            TFT_eSPI * tft)
```

Rysuje aktualną wartość taryfy na ekranie.

Odświeża wyświetlaną liczbę po zmianie wartości przez użytkownika.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 67 of file screen_tariff.cpp.

8.67.2.2 handleTariffTouch()

```
void handleTariffTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie taryfy.

Przetwarza zmianę wartości, przełączanie trybów i zapis ustawień.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 83 of file screen_tariff.cpp.

8.67.2.3 initTariffScreen()

```
void initTariffScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran konfiguracji taryfy.

Rysuje kontrolki do zmiany wartości i trybu taryfy.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 52 of file screen_tariff.cpp.

8.67.2.4 loadTariffFromEEPROM()

```
void loadTariffFromEEPROM ()
```

Ładuje ustawienia taryfy z EEPROM.

Odczytuje zapisaną wartość i tryb taryfy z pamięci nieulotnej. Wywoływane przy starcie aplikacji.

Definition at line 20 of file screen_tariff.cpp.

8.67.2.5 saveTariffToEEPROM()

```
void saveTariffToEEPROM ()
```

Zapisuje ustawienia taryfy do EEPROM.

Zachowuje aktualną wartość i tryb taryfy w pamięci nieulotnej.

Definition at line 40 of file screen_tariff.cpp.

8.67.3 Variable Documentation

8.67.3.1 bgTariff

```
Background* bgTariff = nullptr  [static]
```

Definition at line 10 of file screen_tariff.cpp.

8.67.3.2 tariffMode

```
TariffMode tariffMode = TARIFF_PER_KM
```

Aktualny tryb taryfy.

Określa czy rozliczamy za km czy za litr paliwa.

Definition at line 18 of file screen_tariff.cpp.

8.67.3.3 tariffValue

```
float tariffValue = 3.00f
```

Wartość taryfy w aktualnym trybie.

Przechowuje cenę za jednostkę (km lub litr) w zależności od trybu.

Definition at line 17 of file screen_tariff.cpp.

8.67.3.4 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 9 of file screen_tariff.cpp.

8.68 screen_tariff.cpp

Go to the documentation of this file.

```
00001 #include "screen_tariff.h"
00002 #include "background.h"
00003 #include "gui_elements.h"
00004 #include "screen_manager.h"
00005 #include "screen_home.h"
00006 #include <EEPROM.h>
00007 #include <Arduino.h>
00008
00009 static TFT_eSPI* tftPtr = nullptr;
00010 static Background* bgTariff = nullptr;
00011
00012 // Adresy w EEPROM
00013 #define EEPROM_TARIFF_VALUE_ADDR 10
00014 #define EEPROM_TARIFF_MODE_ADDR 14
00015
00016 // Globalne zmienne taryfy
00017 float tariffValue = 3.00f;
00018 TariffMode tariffMode = TARIFF_PER_KM;
00019
00020 void loadTariffFromEEPROM() {
00021
00022     Serial.print("[TARIFF] tariff loaded from EEPROM: \n");
00023     EEPROM.get(EEPROM_TARIFF_VALUE_ADDR, tariffValue);
00024     Serial.print("[TARIFF] Loaded value: ");
00025     Serial.println(tariffValue, 4);
00026
00027     if (isnan(tariffValue) || tariffValue < 0.01f || tariffValue > 100.0f) {
00028
00029         Serial.println("[TARIFF] Invalid value, setting default 3.00");
00030         tariffValue = 3.00f; // default
00031     }
00032
00033     uint8_t mode = EEPROM.read(EEPROM_TARIFF_MODE_ADDR);
00034     Serial.print("[TARIFF] Loaded mode: ");
```

```
00035     Serial.println(mode);
00036     if (mode > 1) mode = 0;
00037     tariffMode = (TariffMode)mode;
00038 }
00039
00040 void saveTariffToEEPROM() {
00041
00042     EEPROM.put(EEPROM_TARIFF_VALUE_ADDR, tariffValue);
00043     EEPROM.write(EEPROM_TARIFF_MODE_ADDR, (uint8_t)tariffMode);
00044     EEPROM.commit();
00045     Serial.print("[TARIFF] Tariff saved to EEPROM: \n");
00046     Serial.print("[TARIFF] Saved value: ");
00047     Serial.println(tariffValue, 4);
00048     Serial.print("[TARIFF] Saved mode: ");
00049     Serial.println((uint8_t)tariffMode);
00050 }
00051
00052 void initTariffScreen(TFT_eSPI* tft) {
00053
00054     tftPtr = tft;
00055
00056     if (bgTariff) {
00057
00058         delete bgTariff;
00059         bgTariff = nullptr;
00060     }
00061
00062     bgTariff = new Background("/tariff.png");
00063     bgTariff->draw(*tft, *bgTariff->s_png, true);
00064     drawTariffValue(tft);
00065 }
00066
00067 void drawTariffValue(TFT_eSPI* tft) {
00068
00069     char buf[32];
00070
00071     if (tariffMode == TARIFF_PER_KM) {
00072
00073         sprintf(buf, "%.2f ZL/K", tariffValue);
00074         drawTextWithBackground(tft, buf, 300, 75, TR_DATUM, 2, TFT_YELLOW, TFT_BLACK, 60);
00075
00076     } else {
00077
00078         sprintf(buf, "%.2f ZL/L", tariffValue);
00079         drawTextWithBackground(tft, buf, 300, 75, TR_DATUM, 2, TFT_CYAN, TFT_BLACK, 60);
00080     }
00081 }
00082
00083 void handleTariffTouch(uint16_t x, uint16_t y) {
00084
00085     // -1zł
00086     if (x >= 10 && x < 150 && y >= 120 && y < 155) {
00087
00088         tariffValue = max(0.0f, tariffValue - 1.0f);
00089         drawTariffValue(tftPtr);
00090         return;
00091     }
00092     // +1zł
00093     if (x >= 170 && x < 310 && y >= 120 && y < 155) {
00094
00095         tariffValue += 1.0f;
00096         drawTariffValue(tftPtr);
00097         return;
00098     }
00099     // -0,5zł
00100     if (x >= 10 && x < 150 && y >= 160 && y < 175) {
00101
00102         tariffValue = max(0.0f, tariffValue - 0.5f);
00103         drawTariffValue(tftPtr);
00104         return;
00105     }
00106     // -0,5zł
00107     if (x >= 170 && x < 310 && y >= 160 && y < 175) {
00108
00109         tariffValue += 0.5f;
00110         drawTariffValue(tftPtr);
00111         return;
00112     }
00113     // Switch mode
00114     if (x >= 10 && x < 310 && y >= 195 && y < 230) {
00115
00116         tariffMode = (tariffMode == TARIFF_PER_KM) ? TARIFF_PER_LITRE : TARIFF_PER_KM;
00117         drawTariffValue(tftPtr);
00118         return;
00119     }
00120     // Back
00121     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
```

```
00122
00123         saveTariffToEEPROM();
00124         initHomeScreen(tftPtr);
00125         currentScreen = SCREEN_HOME;
00126         return;
00127     }
00128 }
```

8.69 src/screen_trip.cpp File Reference

```
#include "screen_trip.h"
#include "screen_obd.h"
#include "tft_display.h"
#include "background.h"
#include "gui_elements.h"
#include "obd_reader.h"
#include "screen_tariff.h"
#include "screen_manager.h"
#include "screen_home.h"
#include "sd_manager.h"
#include <Arduino.h>
#include <EEPROM.h>
#include "esp_task_wdt.h"
```

Macros

- #define TRIP_EEPROM_VALID_FLAG 15
- #define TRIP_EEPROM_DISTANCE 16
- #define TRIP_EEPROM_FUEL 20
- #define TRIP_EEPROM_PAUSED 24

Functions

- void saveTripDataToEEPROM ()

  Zapisuje dane trasy do EEPROM.
- bool loadTripDataFromEEPROM ()

  Wczytuje dane trasy z EEPROM.
- void clearTripDataFromEEPROM ()

  Czyści zapisane dane trasy z EEPROM.
- void resetTripData ()

  Resetuje wszystkie dane trasy do wartości początkowych.
- void initTripScreen (TFT_eSPI *tft)

  Inicjalizuje ekran trasy.
- void updateTripStatus (TFT_eSPI *tft)

  Aktualizuje wyświetlany status trasy.
- void handleTripTouch (uint16_t x, uint16_t y)

  Obsługuje dotyk na ekranie trasy.

Variables

- bool tripActive = false

  Czy trasa jest aktywna.
- bool tripPaused = false

  Czy trasa jest zapauzowana.
- static char lastDueText [32] = ""
- static char lastDistText [32] = ""
- static char lastFuelText [32] = ""
- static TFT_eSPI * tftPtr = nullptr
- static Background * bgTrip = nullptr
- float distanceTraveled = 0.0f

  Przejechany dystans w kilometrach.
- float fuelUsed = 0.0f

  Zużyte paliwo w litrach.
- bool obdErrorPending = false

  Flaga błędu OBD do obsługi w pętli głównej.
- bool resetTripLogicFlag = false

  Flaga resetu logiki tripa.

8.69.1 Macro Definition Documentation

8.69.1.1 TRIP_EEPROM_DISTANCE

```
#define TRIP_EEPROM_DISTANCE 16
```

Definition at line 18 of file screen_trip.cpp.

8.69.1.2 TRIP_EEPROM_FUEL

```
#define TRIP_EEPROM_FUEL 20
```

Definition at line 19 of file screen_trip.cpp.

8.69.1.3 TRIP_EEPROM_PAUSED

```
#define TRIP_EEPROM_PAUSED 24
```

Definition at line 20 of file screen_trip.cpp.

8.69.1.4 TRIP_EEPROM_VALID_FLAG

```
#define TRIP_EEPROM_VALID_FLAG 15
```

Definition at line 17 of file screen_trip.cpp.

8.69.2 Function Documentation

8.69.2.1 clearTripDataFromEEPROM()

```
void clearTripDataFromEEPROM ()
```

Czyści zapisane dane trasy z EEPROM.

Usuwa dane z EEPROM gdy trasa zostanie prawidłowo zakończona.

Definition at line 81 of file screen_trip.cpp.

8.69.2.2 handleTripTouch()

```
void handleTripTouch (
            uint16_t x,
            uint16_t y)
```

Obsługuje dotyk na ekranie trasy.

Przetwarza przyciski start/pauza/stop oraz nawigację.

Parameters

x	Współrzędna X punktu dotyku
y	Współrzędna Y punktu dotyku

Definition at line 186 of file screen_trip.cpp.

8.69.2.3 initTripScreen()

```
void initTripScreen (
            TFT_eSPI * tft)
```

Inicjalizuje ekran trasy.

Rysuje interfejs z danymi trasy (dystans, spalanie, koszt) oraz przyciski sterujące.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 102 of file screen_trip.cpp.

8.69.2.4 loadTripDataFromEEPROM()

```
bool loadTripDataFromEEPROM ()
```

Wczytuje dane trasy z EEPROM.

Odtwarza poprzednią sesję w przypadku utraty zasilania.

Returns

true jeśli dane zostały pomyślnie wczytane, false jeśli EEPROM jest pusty

Definition at line 59 of file screen_trip.cpp.

8.69.2.5 resetTripData()

```
void resetTripData ()
```

Resetuje wszystkie dane trasy do wartości początkowych.

Zeruje liczniki dystansu, spalania i kosztu. Wywoływane przy rozpoczynaniu nowej trasy.

Definition at line 89 of file screen_trip.cpp.

8.69.2.6 saveTripDataToEEPROM()

```
void saveTripDataToEEPROM ()
```

Zapisuje dane trasy do EEPROM.

Przechowuje dystans, zużycie paliwa i stan trasy. Wywoływane co 30 sekund podczas aktywnej trasy.

Definition at line 38 of file screen_trip.cpp.

8.69.2.7 updateTripStatus()

```
void updateTripStatus (
            TFT_eSPI * tft)
```

Aktualizuje wyświetlany status trasy.

Odświeża informacje o dystansie, spalaniu i koszcie bez przerysowywania całego ekranu.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Definition at line 147 of file screen_trip.cpp.

8.69.3 Variable Documentation

8.69.3.1 bgTrip

```
Background* bgTrip = nullptr  [static]
```

Definition at line 29 of file screen_trip.cpp.

8.69.3.2 distanceTraveled

```
float distanceTraveled = 0.0f
```

Przejechany dystans w kilometrach.

Wartość akumulowana od rozpoczęcia trasy.

Definition at line 32 of file screen_trip.cpp.

8.69.3.3 fuelUsed

```
float fuelUsed = 0.0f
```

Zużyte paliwo w litrach.

Suma spalania od rozpoczęcia trasy.

Definition at line 33 of file screen_trip.cpp.

8.69.3.4 lastDistText

```
char lastDistText[32] = ""  [static]
```

Definition at line 26 of file screen_trip.cpp.

8.69.3.5 lastDueText

```
char lastDueText[32] = ""  [static]
```

Definition at line 25 of file screen_trip.cpp.

8.69.3.6 lastFuelText

```
char lastFuelText[32] = ""  [static]
```

Definition at line 27 of file screen_trip.cpp.

8.69.3.7 obdErrorPending

```
bool obdErrorPending = false
```

Flaga błędu OBD do obsługi w pętli głównej.

Sygnalizuje utratę połączenia z modułem OBD podczas trasy.

Definition at line 34 of file screen_trip.cpp.

8.69.3.8 resetTripLogicFlag

```
bool resetTripLogicFlag = false
```

Flaga resetu logiki tripa.

Używana do synchronizacji operacji resetowania w głównej pętli.

Definition at line 35 of file screen_trip.cpp.

8.69.3.9 tftPtr

```
TFT_eSPI* tftPtr = nullptr  [static]
```

Definition at line 28 of file screen_trip.cpp.

8.69.3.10 tripActive

```
bool tripActive = false
```

Czy trasa jest aktywna.

true = trip rozpoczęty, false = trip nie rozpoczęty

Definition at line 23 of file screen_trip.cpp.

8.69.3.11 tripPaused

```
bool tripPaused = false
```

Czy trasa jest zapauzowana.

true = trip wstrzymany, false = trip w trakcie

Definition at line 24 of file screen_trip.cpp.

8.70 screen_trip.cpp

Go to the documentation of this file.

```
00001 #include "screen_trip.h"
00002 #include "screen_obd.h"
00003 #include "tft_display.h"
00004 #include "background.h"
00005 #include "gui_elements.h"
00006 #include "obd_reader.h"
00007 #include "screen_tariff.h"
00008 #include "screen_manager.h"
00009 #include "screen_home.h"
00010 #include "sd_manager.h"
00011
00012 #include <Arduino.h>
00013 #include <EEPROM.h>
00014 #include "esp_task_wdt.h"
00015
00016 // EEPROM adresy dla danych trasy (po tariff: bajt 15+)
00017 #define TRIP_EEPROM_VALID_FLAG 15    // 1 bajt - flaga ważności danych
00018 #define TRIP_EEPROM_DISTANCE 16      // 4 bajty - dystans (float)
00019 #define TRIP_EEPROM_FUEL 20          // 4 bajty - paliwo (float)
00020 #define TRIP_EEPROM_PAUSED 24        // 1 bajt - flaga pauzowania
00021
00022 // Bufory tekstów do aktualizacji tylko przy zmianie
00023 bool tripActive = false;
00024 bool tripPaused = false;
00025 static char lastDueText[32] = "";
00026 static char lastDistText[32] = "";
00027 static char lastFuelText[32] = "";
00028 static TFT_eSPI* tftPtr = nullptr;
00029 static Background* bgTrip = nullptr;
00030
00031 // Dane trasy (globalne, liczone w tle)
00032 float distanceTraveled = 0.0f; // w km
00033 float fuelUsed = 0.0f;         // w litrach
00034 bool obdErrorPending = false;  // Flaga błędu OBD do obsługi w pętli głównej
00035 bool resetTripLogicFlag = false;  // Flaga resetu logiki tripa
00036
00037 // Zapisanie danych trasy do EEPROM
00038 void saveTripDataToEEPROM() {
00039
00040     // Flaga ważności - 0xAB oznacza że dane są ważne
00041     EEPROM.write(TRIP_EEPROM_VALID_FLAG, 0xAB);
00042
00043     // Zapisz dystans (float = 4 bajty)
00044     EEPROM.writeFloat(TRIP_EEPROM_DISTANCE, distanceTraveled);
00045
00046     // Zapisz zużycie paliwa (float = 4 bajty)
00047     EEPROM.writeFloat(TRIP_EEPROM_FUEL, fuelUsed);
00048
00049     // Zapisz stan pauzy (1 bajt)
00050     EEPROM.write(TRIP_EEPROM_PAUSED, tripPaused ? 1 : 0);
00051
00052     // Zatwierdź zapisy
00053     EEPROM.commit();
00054
00055     Serial.printf("[TRIP] Saved to EEPROM: dist=%.2f km, fuel=%.2f L, paused=%d\n", distanceTraveled,
      fuelUsed, tripPaused);
00056 }
00057
00058 // Wczytanie danych trasy z EEPROM
00059 bool loadTripDataFromEEPROM() {
00060
00061     // Sprawdź flagę ważności
00062     uint8_t validFlag = EEPROM.read(TRIP_EEPROM_VALID_FLAG);
00063
00064     if (validFlag != 0xAB) {
00065
00066         Serial.println("[TRIP] No valid trip data in EEPROM");
00067         return false;
00068     }
00069
00070     // Wczytaj dane
00071     distanceTraveled = EEPROM.readFloat(TRIP_EEPROM_DISTANCE);
00072     fuelUsed = EEPROM.readFloat(TRIP_EEPROM_FUEL);
00073     tripPaused = (EEPROM.read(TRIP_EEPROM_PAUSED) == 1);
00074
00075     Serial.printf("[TRIP] Loaded from EEPROM: dist=%.2f km, fuel=%.2f L,
      paused=%d\n",distanceTraveled, fuelUsed, tripPaused);
00076
00077     return true;
00078 }
00079
00080 // Czyszczenie danych trasy z EEPROM
```

```
00081 void clearTripDataFromEEPROM() {
00082
00083     EEPROM.write(TRIP_EEPROM_VALID_FLAG, 0xFF); // Flaga nieważności
00084     EEPROM.commit();
00085     Serial.println("[TRIP] Trip data cleared from EEPROM");
00086 }
00087
00088 // Reset danych trasy
00089 void resetTripData() {
00090
00091     distanceTraveled = 0.0f;
00092     fuelUsed = 0.0f;
00093     tripPaused = false;
00094     tripActive = false;
00095     resetTripLogicFlag = true;  // Zresetuj także logikę liczenia
00096     resetTextBuffer(lastDueText, sizeof(lastDueText));
00097     resetTextBuffer(lastDistText, sizeof(lastDistText));
00098     resetTextBuffer(lastFuelText, sizeof(lastFuelText));
00099 }
00100
00101 // Inicjalizacja ekranu trasy
00102 void initTripScreen(TFT_eSPI* tft) {
00103
00104     tftPtr = tft;
00105     if (bgTrip) { delete bgTrip; bgTrip = nullptr; }
00106
00107     // Próba wczytania poprzedniej sesji z EEPROM
00108     if (!loadTripDataFromEEPROM()) {
00109
00110         // Jeśli nie ma danych w EEPROM, RESET
00111         distanceTraveled = 0.0f;
00112         fuelUsed = 0.0f;
00113         tripPaused = false;
00114     }
00115
00116     // Wznowienie istniejącego stanu tripa (nie resetuj)
00117     if (tripPaused)
00118         bgTrip = new Background("/trip_paused.png");
00119     else
00120         bgTrip = new Background("/trip.png");
00121
00122     // Rysowanie tła
00123     bgTrip->draw(*tft, *bgTrip->s_png, true);
00124     tripActive = true;
00125     Serial.println("[TRIP] Trip screen initialized - STARTING TRIP");
00126     Serial.printf("[TRIP] OBD::btConnected=%d, OBD::elmReady=%d\n", OBD::btConnected, OBD::elmReady);
00127
00128
00129     // ========== Tworzenie nowej sesji SD ==========
00130     if (SDManager::isReady()) {
00131
00132         extern String currentTripPath;
00133         currentTripPath = SDManager::createTripSession();
00134
00135         if (!currentTripPath.isEmpty())
00136             Serial.printf("[TRIP] SD session created: %s\n", currentTripPath.c_str());
00137     }
00138
00139     // Reset buforów tekstów do aktualizacji
00140     resetTextBuffer(lastDueText, sizeof(lastDueText));
00141     resetTextBuffer(lastDistText, sizeof(lastDistText));
00142     resetTextBuffer(lastFuelText, sizeof(lastFuelText));
00143     updateTripStatus(tftPtr);
00144 }
00145
00146 // Aktualizacja statusu trasy
00147 void updateTripStatus(TFT_eSPI* tft) {
00148
00149     if (!tft) return;
00150     // Obliczenie należności
00151     float currentDue = 0.0f;
00152     int startedKm = (int)distanceTraveled + 1; // Każdy rozpoczęty km, minimum 1
00153
00154     if (tariffMode == TARIFF_PER_KM)
00155         currentDue = startedKm * tariffValue;
00156     else
00157         currentDue = fuelUsed * tariffValue;
00158
00159     // Bufory tekstów
00160     char dueBuf[32], distBuf[32], fuelBuf[32];
00161     sprintf(dueBuf, "%.2f ZL", currentDue);
00162     sprintf(distBuf, "%d km", startedKm);
00163     sprintf(fuelBuf, "%.2f L", fuelUsed);
00164
00165     // Rysowanie tekstu tylko jeśli zmienne zmieniły wartość od ostatniego rysowania
00166     if (strcmp(dueBuf, lastDueText) != 0) {
00167
```

```
00168         drawTextWithBackground(tft, dueBuf, 300, 70, TR_DATUM, 2, TFT_YELLOW, TFT_BLACK, 0);
00169         strcpy(lastDueText, dueBuf);
00170     }
00171
00172     if (strcmp(distBuf, lastDistText) != 0) {
00173
00174         drawTextWithBackground(tft, distBuf, 300, 110, TR_DATUM, 2, TFT_GREEN, TFT_BLACK, 0);
00175         strcpy(lastDistText, distBuf);
00176     }
00177
00178     if (strcmp(fuelBuf, lastFuelText) != 0) {
00179
00180         drawTextWithBackground(tft, fuelBuf, 300, 130, TR_DATUM, 2, TFT_CYAN, TFT_BLACK, 0);
00181         strcpy(lastFuelText, fuelBuf);
00182     }
00183 }
00184
00185 // Obsługa dotyku na ekranie trasy
00186 void handleTripTouch(uint16_t x, uint16_t y) {
00187
00188     // Przycisk pauzy/odpauzowania
00189     if (x >= 20 && x < 220 && y >= 200 && y < 240) {
00190
00191         if (!tripPaused) {
00192
00193             tripPaused = true;
00194             if (bgTrip) { delete bgTrip; bgTrip = nullptr; }
00195             bgTrip = new Background("/trip_paused.png");
00196             bgTrip->draw(*tftPtr, *bgTrip->s_png, true);
00197             resetTextBuffer(lastDueText, sizeof(lastDueText));
00198             resetTextBuffer(lastDistText, sizeof(lastDistText));
00199             resetTextBuffer(lastFuelText, sizeof(lastFuelText));
00200             updateTripStatus(tftPtr);
00201             Serial.println("[TRIP] Trip paused");
00202             delay(50);
00203
00204         } else {
00205
00206             tripPaused = false;
00207             if (bgTrip) { delete bgTrip; bgTrip = nullptr; }
00208             bgTrip = new Background("/trip.png");
00209             bgTrip->draw(*tftPtr, *bgTrip->s_png, true);
00210             resetTextBuffer(lastDueText, sizeof(lastDueText));
00211             resetTextBuffer(lastDistText, sizeof(lastDistText));
00212             resetTextBuffer(lastFuelText, sizeof(lastFuelText));
00213             updateTripStatus(tftPtr);
00214             Serial.println("[TRIP] Trip resumed");
00215             delay(50);
00216         }
00217         return;
00218     }
00219
00220     // Przycisk powrotu: jeśli trip jest zapauzowany, kasuje tripa, jeśli nie to tylko wraca do ekranu
      głównego
00221     if (x >= 260 && x < 310 && y >= 10 && y < 50) {
00222
00223         if (tripPaused) {
00224
00225             // ========== FINALIZACJA TRASY NA KARCIE SD ==========
00226             extern String currentTripPath;
00227             if (SDManager::isReady() && !currentTripPath.isEmpty()) {
00228
00229                 SDManager::TripData finalData;
00230                 finalData.distanceKm = distanceTraveled;
00231                 finalData.fuelUsedLiters = fuelUsed;
00232                 finalData.tariffMode = tariffMode;
00233                 finalData.tariffValue = tariffValue;
00234
00235                 // Wiliczenie całkowietego kosztu
00236                 int startedKm = (int)distanceTraveled + 1;
00237                 if (tariffMode == TARIFF_PER_KM)
00238                     finalData.totalCost = startedKm * tariffValue;
00239                 else
00240                     finalData.totalCost = fuelUsed * tariffValue;
00241
00242                 SDManager::finalizeTrip(finalData);
00243                 currentTripPath = "";  // Wyczyszczenie ścieżki bieżącej trasy
00244             }
00245
00246             resetTripData();
00247             clearTripDataFromEEPROM();
00248             Serial.println("[TRIP] Trip ended and reset (exit from pause)");
00249         }
00250
00251         // Powrót do ekranu głównego
00252         initHomeScreen(tftPtr);
00253         currentScreen = SCREEN_HOME;
```

```
00254         tftPtr = nullptr;
00255         return;
00256     }
00257 }
```

8.71 src/sd_manager.cpp File Reference

```
#include "sd_manager.h"
#include "../cabulator_settings.h"
#include <time.h>
#include <SPI.h>
#include <SD.h>
```

Namespaces

- namespace SDManager

Functions

- static SPIClass sdSPI (HSPI)
- static String SDManager::getTimestamp ()
- static String SDManager::getDateOnly ()
- bool SDManager::init ()

  Inicjalizuje kartę SD na HSPI (Secondary SPI).
- bool SDManager::isReady ()

  Sprawdza czy karta SD jest gotowa.
- String SDManager::createTripSession ()

  Tworzy nową sesję trasy z timestamp'em.
- bool SDManager::saveTripData (const String &tripPath, const TripData &data)

  Zapisuje dane trasy do pliku trip_data.csv.
- bool SDManager::saveGPSData (const String &tripPath, const GPSData &data)

  Zapisuje wpis danych GPS do pliku gps_log.csv.
- void SDManager::listTrips ()

  Listuje wszystkie dostępne logi tras.
- bool SDManager::getLastTrip (String &tripPath)

  Pobiera dane z ostatniej trasy.
- void SDManager::onGPSFix (const GPSData &data)

  Callback wywoływany przez GPS gdy pojawi się nowy fix.
- void SDManager::finalizeTrip (const TripData &data)

  Finalizuje trasę - zapisuje ostateczne dane podsumowania na SD.

Variables

- bool tripActive

  Czy trasa jest aktywna.
- String currentTripPath
- static bool SDManager::sdReady = false
- static String SDManager::currentTripPath = ""

8.71.1 Function Documentation

8.71.1.1 sdSPI()

```
SPIClass sdSPI (
            HSPI )  [static]
```

8.71.2 Variable Documentation

8.71.2.1 currentTripPath

```
String currentTripPath  [extern]
```

Definition at line 32 of file main.cpp.

8.71.2.2 tripActive

```
bool tripActive  [extern]
```

Czy trasa jest aktywna.

true = trip rozpoczęty, false = trip nie rozpoczęty

Definition at line 23 of file screen_trip.cpp.

8.72 sd_manager.cpp

Go to the documentation of this file.

```
00001 #include "sd_manager.h"
00002 #include "../cabulator_settings.h"
00003 #include <time.h>
00004 #include <SPI.h>
00005 #include <SD.h>
00006
00007 // Czytnik SD używa HSPI (Secondary SPI) aby nie kolidował z TFT (Primary SPI)
00008 // TFT używa VSP (Primary SPI) - SD zostało przeniesione na osobny kanał
00009
00010 using namespace SDCARD;
00011
00012 // Instancja SPI dla karty SD (HSPI)
00013 static SPIClass sdSPI(HSPI);
00014
00015 extern bool tripActive;
00016 extern String currentTripPath;
00017
00018 namespace SDManager {
00019
00020     // Zmienne globalne
00021     static bool sdReady = false;
00022     static String currentTripPath = "";
00023
00024     // Funkcja pomocnicza: zwraca aktualną datę i czas w formacie YYYY-MM-DD_HH-MM-SS
00025     static String getTimestamp() {
00026         time_t now = time(nullptr);
00027         struct tm* timeinfo = localtime(&now);
00028
00029         char buffer[20];
00030         strftime(buffer, sizeof(buffer), "%Y-%m-%d_%H-%M-%S", timeinfo);
00031         return String(buffer);
00032     }
00033
```

```
00034     // Funkcja pomocnicza: zwraca aktualną datę w formacie YYYY-MM-DD
00035     static String getDateOnly() {
00036         time_t now = time(nullptr);
00037         struct tm* timeinfo = localtime(&now);
00038 
00039         char buffer[11];
00040         strftime(buffer, sizeof(buffer), "%Y-%m-%d", timeinfo);
00041         return String(buffer);
00042     }
00043 
00044     bool init() {
00045         Serial.println("[SD] SD card initializing...");
00046 
00047         // Konfiguracja SPI dla karty SD - używamy HSPI
00048         sdSPI.begin(SDCARD::PIN_CLK, SDCARD::PIN_MISO_DATA, SDCARD::PIN_MOSI_DATA, SDCARD::PIN_CS_SD);
00049 
00050         // Inicjalizacja karty SD z instancją HSPI
00051         if (!SD.begin(SDCARD::PIN_CS_SD, sdSPI, 4000000)) {  // 4MHz speed for SD
00052             Serial.println("[SD] ERROR: Failed to initialize SD card!");
00053             sdReady = false;
00054             return false;
00055         }
00056 
00057         Serial.println("[SD] SD card initialized successfully!");
00058         uint64_t cardSize = SD.cardSize() / (1024 * 1024);
00059         Serial.printf("[SD] Card size: %lld MB\n", cardSize);
00060         sdReady = true;
00061 
00062         // Tworzenie głównych katalogów jeśli nie istnieją
00063         SD.mkdir("/logs");
00064         SD.mkdir("/logs/trips");
00065 
00066         return true;
00067     }
00068 
00069     bool isReady() {
00070         return sdReady;
00071     }
00072 
00073     String createTripSession() {
00074         if (!sdReady) {
00075             Serial.println("[SD] ERROR: SD card is not ready!");
00076             return "";
00077         }
00078 
00079         // Tworzenie ścieżki z datą i czasem
00080         String timestamp = getTimestamp();
00081         String tripPath = "/logs/trips/" + timestamp;
00082 
00083         Serial.println("[SD] Creating trip session: " + tripPath);
00084 
00085         // Tworzenie folderu
00086         if (!SD.mkdir(tripPath)) {
00087             Serial.println("[SD] WARNING: Folder already exists or could not be created: " +
    tripPath);
00088         }
00089 
00090         currentTripPath = tripPath;
00091 
00092         // Tworzenie nagłówka pliku trip_data.csv
00093         File tripFile = SD.open(tripPath + "/trip_data.csv", FILE_WRITE);
00094         if (tripFile) {
00095             tripFile.println("Timestamp,DistanceKm,FuelLiters,TariffMode,TariffValue,TotalCost");
00096             tripFile.close();
00097             Serial.println("[SD] File trip_data.csv created");
00098         }
00099 
00100         // Tworzenie nagłówka pliku gps_log.csv
00101         File gpsFile = SD.open(tripPath + "/gps_log.csv", FILE_WRITE);
00102         if (gpsFile) {
00103             gpsFile.println("Timestamp,Latitude,Longitude,Satellites,HDOP,Valid");
00104             gpsFile.close();
00105             Serial.println("[SD] File gps_log.csv created");
00106         }
00107 
00108         return tripPath;
00109     }
00110 
00111     bool saveTripData(const String& tripPath, const TripData& data) {
00112 
00113         if (!sdReady) {
00114 
00115             Serial.println("[SD] ERROR: SD card is not ready!");
00116             return false;
00117         }
00118 
00119         File tripFile = SD.open(tripPath + "/trip_data.csv", FILE_APPEND);
```

```
00120         if (!tripFile) {
00121 
00122             Serial.println("[SD] ERROR: Failed to open trip_data.csv file!");
00123             return false;
00124         }
00125 
00126         // FORMAT DANYCH TRIPU
00127         // Format: Timestamp,DistanceKm,FuelLiters,TariffMode,TariffValue,TotalCost
00128         String line = String(millis()) + ",";
00129         line += String(data.distanceKm, 2) + ",";
00130         line += String(data.fuelUsedLiters, 3) + ",";
00131         line += String(data.tariffMode) + ",";
00132         line += String(data.tariffValue, 2) + ",";
00133         line += String(data.totalCost, 2);
00134 
00135         tripFile.println(line);
00136         tripFile.close();
00137 
00138         return true;
00139     }
00140 
00141     bool saveGPSData(const String& tripPath, const GPSData& data) {
00142 
00143         if (!sdReady) {
00144 
00145             Serial.println("[SD] ERROR: SD card is not ready!");
00146             return false;
00147         }
00148 
00149         File gpsFile = SD.open(tripPath + "/gps_log.csv", FILE_APPEND);
00150         if (!gpsFile) {
00151 
00152             Serial.println("[SD] ERROR: Failed to open gps_log.csv file!");
00153             return false;
00154         }
00155 
00156         // FORMAT DANYCH GPS
00157         // Format: Timestamp,Latitude,Longitude,Satellites,HDOP,Valid
00158         String line = String(data.timestamp) + ",";
00159         line += String(data.latitude, 6) + ",";
00160         line += String(data.longitude, 6) + ",";
00161         line += String(data.satellites) + ",";
00162         line += String(data.hdop) + ",";
00163         line += (data.valid ? "1" : "0");
00164 
00165         gpsFile.println(line);
00166         gpsFile.close();
00167 
00168         return true;
00169     }
00170 
00171     void listTrips() {
00172 
00173         if (!sdReady) {
00174 
00175             Serial.println("[SD] ERROR: SD card is not ready!");
00176             return;
00177         }
00178 
00179         Serial.println("[SD] === AVAILABLE TRIPS ===");
00180         File root = SD.open("/logs/trips");
00181 
00182         if (!root || !root.isDirectory()) {
00183 
00184             Serial.println("[SD] Trips folder is missing or is a file!");
00185             return;
00186         }
00187 
00188         File file = root.openNextFile();
00189         int tripCount = 0;
00190 
00191         while (file) {
00192 
00193             if (file.isDirectory()) {
00194 
00195                 Serial.println("  -> " + String(file.name()));
00196                 tripCount++;
00197             }
00198             file = root.openNextFile();
00199         }
00200 
00201         Serial.println("[SD] Total trips: " + String(tripCount));
00202         root.close();
00203     }
00204 
00205     bool getLastTrip(String& tripPath) {
00206 
```

```
00207         if (!sdReady) {
00208
00209             Serial.println("[SD] ERROR: SD card is not ready!");
00210             return false;
00211         }
00212
00213         File root = SD.open("/logs/trips");
00214         if (!root || !root.isDirectory())
00215             return false;
00216
00217         String lastTripName = "";
00218         File file = root.openNextFile();
00219
00220         while (file) {
00221
00222             if (file.isDirectory()) {
00223
00224                 String fileName = String(file.name());
00225                 if (fileName > lastTripName)
00226                     lastTripName = fileName;
00227             }
00228             file = root.openNextFile();
00229         }
00230         root.close();
00231
00232         if (lastTripName.isEmpty())
00233             return false;
00234
00235         tripPath = "/logs/trips/" + lastTripName;
00236         return true;
00237     }
00238
00239     void onGPSFix(const GPSData& data) {
00240
00241         // Callback wywoływany przez GPS gdy pojawi się nowy fix
00242         // Sprawdzenie czy sesja tripu jest aktywna
00243         if (tripActive && isReady() && !currentTripPath.isEmpty())
00244             saveGPSData(currentTripPath, data);
00245     }
00246
00247     void finalizeTrip(const TripData& data) {
00248         // Finalizacja sesji tripu - zapisanie podsumowania
00249         if (!isReady() || currentTripPath.isEmpty()) {
00250             Serial.println("[SD] ERROR: Cannot finalize - SD path is missing!");
00251             return;
00252         }
00253
00254         Serial.println("[SD] Finalizing trip session...");
00255         saveTripData(currentTripPath, data);
00256     }
00257
00258 }  // namespace SDManager
```

8.73 src/tft_display.cpp File Reference

```
#include "tft_display.h"
```

Functions

- void initTFT (TFT_eSPI *tft)

  Inicjalizuje wyświetlacz TFT.
- void setBacklight (uint8_t brightness)

  Ustawia jasność podświetlenia wyświetlacza.

Variables

- constexpr int BL_PIN = 17
- constexpr int BL_CH = 0
- uint16_t calData [5] = { 243, 3566, 356, 3415, 1 }

8.73.1 Function Documentation

8.73.1.1 initTFT()

```
void initTFT (
            TFT_eSPI * tft)
```

Inicjalizuje wyświetlacz TFT.

Konfiguruje rotację ekranu, kalibrację dotyku oraz ustawienia początkowe wyświetlacza.

Parameters

tft	Wskaźnik do obiektu wyświetlacza TFT

Note

Należy wywołać przed jakąkolwiek operacją rysowania

Definition at line 19 of file tft_display.cpp.

8.73.1.2 setBacklight()

```
void setBacklight (
            uint8_t brightness)
```

Ustawia jasność podświetlenia wyświetlacza.

Steruje jasnością przez PWM na pinie backlight.

Parameters

brightness	Poziom jasności (0-255) 0 = podświetlenie wyłączone 255 = maksymalna jasność

Definition at line 33 of file tft_display.cpp.

8.73.2 Variable Documentation

8.73.2.1 BL_CH

```
int BL_CH = 0  [constexpr]
```

Definition at line 9 of file tft_display.cpp.

8.73.2.2 BL_PIN

```
int BL_PIN = 17  [constexpr]
```

Definition at line 8 of file tft_display.cpp.

8.73.2.3 calData

```
uint16_t calData[5] = { 243, 3566, 356, 3415, 1 }
```

Definition at line 12 of file tft_display.cpp.

8.74 tft_display.cpp

Go to the documentation of this file.

```
00001 #include "tft_display.h"
00002
00003 // ================================================================================
00004 // KONFIGURACJA TFT - Ustawienia dla ekranu TFT
00005 // ================================================================================
00006
00007 // Ustawienia pinu PWM dla podświetlenia
00008 constexpr int BL_PIN = 17;    // Pin do sterowania podświetleniem
00009 constexpr int BL_CH = 0;      // Kanał PWM
00010
00011 // Przechowywanie danych kalibracji ekranu
00012 uint16_t calData[5] = { 243, 3566, 356, 3415, 1 }; // Przykładowe dane kalibracji
00013
00014 // ================================================================================
00015 // FUNKCJE EKRANU TFT - Inicjalizacja i sterowanie podświetleniem
00016 // ================================================================================
00017
00018 // Funkcja do ustawienia konfiguracji TFT (w tym kalibracja)
00019 void initTFT(TFT_eSPI* tft) {
00020   tft->init();                  // Inicjalizacja wyświetlacza
00021   tft->setRotation(3);          // Ustawienie rotacji wyświetlacza
00022   tft->fillScreen(TFT_BLACK);   // Wypełnienie ekranu kolorem czarnym
00023
00024   // Ustawienie danych kalibracji ekranu
00025   tft->setTouch(calData);       // Kalibracja ekranu (na podstawie wartości calData)
00026
00027   // Inicjalizacja PWM dla podświetlenia
00028   ledcSetup(BL_CH, 5000, 8);    // Ustawienie PWM: częstotliwość 5kHz, rozdzielczość 8-bitowa
00029   ledcAttachPin(BL_PIN, BL_CH); // Przypisanie pinu do kanału PWM
00030 }
00031
00032 // Funkcja do ustawienia jasności podświetlenia
00033 void setBacklight(uint8_t brightness) {
00034   ledcWrite(BL_CH, brightness); // Ustawienie jasności (0-255)
00035 }
```