Dokumentacja oprogramowania MW-TFT v1.1

Do karty SD zastosowano system plików FATFS natomiast do pamięci NOR Flash system litlefs, ze względu na bezpieczeństwo danych oraz zoptymalizowanie zużycia pamięci.

Funkcje:

* **void** **eveMsgMultiLine**(**int16\_t** startX, **int16\_t** startY, **int16\_t** font, **int16\_t** lineSpacing, **int16\_t** lineCount, ...)

Wyświetla tekst w podanej ilości wierszy na wyświetlaczu. Parametry *startX*, *startY* odpowiadają za współrzędne początkowe napisu, *font* za wielkość czcionki, *lineSpacing* za odstęp pomiędzy wierszami, *lineCount* za ilość wierszy.

* **void** **dataCopyFn**(**void**)

Jeśli wciśnięty jest przycisk górny(*UP\_BUTTON*) urządzenie przechodzi w tryb kopiowania danych z karty SD do pamięci FLASH. Po włożeniu karty SD i wciśnięciu przycisku enter(*ENT\_BUTTON*) wywoływana jest funkcja:

* **int** **copyFileSdToFlash**(**const** **char** \*sd\_path, **const** **char** \*flash\_path)

Wykonuje ona operację skopiowania plików czcionek: *font.raw, font\_pl.raw, numbers.raw* z karty SD do pamięci FLASH. Na ekranie wyświetlany jest komunikat o powodzeniu lub niepowodzeniu operacji kopiowania plików. Po wywołaniu funkcji konieczne jest zresetowanie urządzenia.

W przypadku braku wciśnięcia przycisku górnego wywoływana jest funkcja

* **void** **\_\_init\_storage**()

Która za pomocą funkcji ***void******\_\_init\_nor****()* inicjalizuje pamięć typu NOR FLASH, następnie za pomocą funkcji ***void******\_\_init\_littlefs****()* inicjuje system plików typu LittleFS.

Jeśli wciśnięty jest przycisk dolny(DN\_BUTTON) wywoływana jest procedura czyszczenia zawartości pamięci FLASH. W tym celu należy potwierdzić wykonanie procedury wciskając jednocześnie przyciski lewy, prawy oraz enter. Po wykonaniu procedury konieczne jest zresetowanie urządzenia.

* **uint32\_t readParameters (void)**

Funkcja odczytuje parametry zapisane w pamięci EEPROM, sprawdza ich poprawność oraz oblicza i zwraca sumę kontrolną(CRC).

* **uint32\_t** **readCRCEEPROM**(**void**)

Odczytuje parametry CRC z pamięci EEPROM.

* **int32\_t** **\_\_attribute\_\_**((optimize("-Ofast"))) **loadAssetFromFlash**(**uint8\_t** fileNo, **char** \*path, **uint32\_t** offset)

Ładuje plik z pamięci FLASH do pamięci RAM, zwraca status operacji 0 – błąd lub 1 – powodzenie. ***\_\_attribute\_\_****((optimize("-Ofast")))* zapewnia optymalizację kompilatora dla jak największej wydajności. Funkcja wyświetla progres operacji ładowania na wyświetlaczu. Parametr *path* odpowiada za ścieżkę do pliku, który ma zostać załadowany, *offset* przesunięcie w pamięci RAM, z jakim zostanie zapisany plik, natomiast *fileNo* numer ładowanego pliku.

* **void** **updateParamValue**(**void**)

Umożliwia zmianę parametrów takich jak jasność wyświetlacza oraz język poprzez liftcan. Parametry te zapisywane są poprzez funkcję ***saveParameters****();*. Jeśli komórka wartości jasności wyświetlacza jest pusta, lub jej wartość ma 0 lub więcej niż wartość maksymalna to jasność wyświetlacza ustawiana jest na pozom maksymalny. Podobnie z kolorem wyświetlacza, który domyślnie ustawiony jest na czarny.

* **uint32\_t** **sendCanFrame** (**uint8\_t** workMode, **uint8\_t** canDLC, **uint8\_t** canByte0, **uint8\_t** canByte1, **uint8\_t** canByte2, **uint8\_t** canByte3)

Wysyła ramkę CAN z 4 bajtami danych. Zmienna *workMode* odpowiada za typ wysyłanej wiadomości(jej CANid), *canDLC* za długość danych, natomiast *canByte1-3* za dane do wysłania. W przypadku wystąpienia błędu transmisji próba wysłania ramki ponawiana jest 5 krotnie.

* **void** **playCallBuzzerFn**(**uint8\_t** \_callDir)

Włącza buzzer na czas 2,5s. Parametr *callDir* oznacza ilość powtórzeń włączenia buzzera możliwe przyjmowane wartości to 1 lub 2. Funkcja ustawia flagę *playGong* na wartość false(0).

* **void** **setOutputs**(**uint8\_t** outputs)

Funkcja ustawia stan 6 wyjść(piny *EU2\_OUT1, EU\_OUT2, EO\_OUT3, EO2\_OUT4, RU\_OUT5, RO\_OUT6*) na podstawie bitów w bajcie outputs. Wartość 1 ustawia stan wysoki, natomiast wartość 0 stan niski. Bit 5 odpowiada wyjściu *EU2\_OUT1*, natomiast bit 0 wyjściu *RO\_OUT6*.

* **void** **setBuzzer**(**void**)

Zarządza stanem buzzera, jeśli *global.buzzerRunState == 1* buzzer ma być aktywny, jeśli *global.buzzerRunState == 2* buzzer ma zostać wyłączony. Parametr zapisywany jest do pamięci EEPROM oraz RAM, jednak tylko w przypadku gdy jej stan uległ zmianie(jest inny niż wartość *nodeParameters[BUZZER].wartosc*).

* **void** **tftDisp3** (**uint8\_t** floorCode, **uint8\_t** displayArrow, **uint8\_t** arrowDir) ??

Wyświetla na ekranie informacje takie jak: opis i numer piętra, strzałki jazdy, aktywuje gong.

* **void** **draw\_arrow**(**int** color\_r, **int** color\_g, **int** color\_b, **int** alpha, **int** line\_width, **int** x, **int** y, **int** arrow\_height, **int** arrow\_width, **int** arrow)

Rysuje strzałki na wyświetlaczu. Parametry: *color\_r/g/b* odpowiadają za kolor strzałki, *alpha* za przezroczystość, *line\_width* grubość linii, *x,y* współrzędne strzałki, *arrow\_height* wysokość strzałki, *arrow\_width* szerokość strzałki, *arrow* kierunek strzałki(góra/dół/obie).

* **void** **checkButtonState**(**void**)

Obsługa menu(wejście do menu odbywa się poprzez ustawienie flagi *menuScreen = true* w przerwaniu od przycisku enter(*ENT\_BUTTON*)) – sprawdza kolejno czy któryś z przycisków został wciśnięty, po 20s bezczynności wychodzi z menu.   
Wciśnięcie przycisku górnego(*UP\_BUTTON*) powoduje przejście do pozycji wyżej, wciśnięcie przycisku dolnego(*DN\_BUTTON*) powoduje przejście do pozycji niżej, przewijanie działa w pętli tj. jeśli znajdujemy się na pozycji skrajnej i wciśnięty zostanie przycisk przeniesieni zostaniemy na samą górę lub dół listy.   
Wciśnięcie przycisku lewego(*LEFT\_BUTTON*) powoduje zmniejszenie wartości aktualnie wybranego parametru, wciśnięcie przycisku prawego(*RIGHT\_BUTTON*) powoduje zwiększenie wartości aktualnie wybranego parametru. W celu uniknięcia przekroczenia dopuszczalnej wartości parametru w przypadku osiągnięcia wartości minimalnej lub maksymalnej wciśnięcie przycisku nie spowoduje żadnej akcji. Podczas zmiany parametru GONG\_VOL wywoływana jest automatycznie funkcja

* **void** **checkGong**(**void**)

która powoduje jednorazowe odegranie gongu.

W przypadku przyciśnięcia przycisku prawego(*RIGHT\_BUTTON*) na pozycji BUZZER, wywołana zostanie funkcja

* **void** **checkBuzzerSound**(**void**)

która odpowiedzialna jest za test działania buzzera.

Wciśnięcie przycisku enter(*ENT\_BUTTON*) wywoła działanie jedynie podczas wybrania 3 pozycji w menu:   
ZAPISZ PARAMETRY – spowoduje zapisanie wprowadzonych parametrów oraz reset systemu(UWAGA! W przypadku wyjścia z menu bez wybrania tej opcji parametry pozostaną niezmienione),   
TEST TFT COLOR – wywoła test działania kanałów kolorów za pomocą funkcji,

* **void** **tftMatrixTest** (**void**)

która kolejno inkrementuje wartości współczynników R/G/B od 0 do wartości maksymalnej 255.

WYJŚCIE – wyjście z menu przy pomocy funkcji

* **void** **outOfMenu**(**void**)

która czyści ekran oraz ustawia kursor na pozycji 40 i zwraca status 5.

Obsługa menu odbywa się przy pomocy funkcji(odświeżanie co 100ms)

* **void** **tftDispMenu** (**uint8\_t** status)

która wyświetla menu tekstowe w zależności od wybranej orientacji wyświetlacza oraz strzałkę „>” przy aktualnie wybranej pozycji.

* **uint8\_t** **playGongFn**(**uint8\_t** step)

Funkcja ustawia głośność i odtwarza gong, w zależności od kierunku jazdy dźwigu – w górę – jeden gong, w dół – 2 gongi, przystanek – 3 gongi.

* **int dispStr(const char\* str, int16\_t y, uint8\_t reset)**

Wyświetla napis i zależnie od długości napisu(mniej niż 13) decyduje czy ma być on wyświetlany statycznie czy przewijany przy pomocy funkcji:

* **int scrollText (const char \*text, uint16\_t startX, uint16\_t startY, uint16\_t endX, uint16\_t speed, uint16\_t gap, uint8\_t reset)**

realizuje przewijanie tekstu poziomo z prawej strony ekranu na lewą, w określonym zakresie i z ustaloną prędkością. Parametr *text* – wyświetlany tekst, *startX/Y* – położenie początku tekstu, *endX* – prawa krawędź tekstu, *speed* – prędkość przewijania, *gap* – dystans między końcem tekstu a ekranu, *reset* – wartość true powoduje powrót na początek przewijanego tekstu.

* **void** **decodeCodeFloor**(**uint8\_t**\* canFloorCode, **uint8\_t**\* flashing, **uint8\_t** \*errorDetect, **uint8\_t** lang, **uint8\_t** lastCode)

Dekoduje i wyświetla informacje o piętrze na podstawie zmiennej *canFloorCode* z magistrali CAN. Za pomocą funkcji

* **void** **displayLargeFont**(**uint8\_t** code)

która wyświetla piętro lub poziom na którym znajduje się dźwig

* **int** **showExtraFunctionCode**(**uint8\_t** Code, **uint8\_t** landscape, **int16\_t** y, **int16\_t** yL1, **int16\_t** yL2, **uint8\_t** reset, **uint8\_t** lang)

która wyświetla symbole I nazwy specjalne pięter.

Jeśli w magistrali wystąpi informacja o wystąpieniu błędu – wskaźnik *errorDetect* na ekranie wyświetla komunikat JAZDA ORIENTUJĄCA w przypadku jej wystąpienia lub za pomocą funkcji

* **int** **showErrorName**(**uint8\_t** lastCode, **uint8\_t** landscape, **int16\_t** y, **int16\_t** yL1, **uint8\_t** reset, **uint8\_t** lang)

komunikat „Exx”, gdzie xx to numer kodu usterki jaki wystąpił.

Jeśli zmienna lastCode, która zawiera ostatni odebrany kod piętra lub status ma wartość 100(KOD\_DZWIG\_NIECZYNNY) lub 200(KOD\_BRAK\_LIFTCAN) funkcja

* **int** **outOforder**(**uint8\_t** lastCode, **int16\_t** yL1, **int16\_t** yL2, **uint8\_t** landscape, **uint8\_t** lang)

wyświetla komunikaty WINDA NIECZYNNA lub LIFTCAN NIE WYKRYTY.