TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

# Grafika zobrazovacího panelu elektrického motocyklu

## Bakalářská práce

*Studijní program:* B2612 Elektrotechnika a informatika
*Studijní obor:* Elektronické informační a řídicí systémy

*Autor práce:* **Vojtěch Hartan**
*Vedoucí práce:* Ing. Lukáš Krčmář
Ústav mechatroniky a technické informatiky

*Konzultant práce:* Ing. Petr Bílek, Ph.D.
Ústav mechatroniky a technické informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

**Zadání bakalářské práce**

# Grafika zobrazovacího panelu elektrického motocyklu

*Jméno a příjmení:* **Vojtěch Hartan**
*Osobní číslo:* M18000027
*Studijní program:* B2612 Elektrotechnika a informatika
*Studijní obor:* Elektronické informační a řídicí systémy
*Zadávající katedra:* Ústav mechatroniky a technické informatiky
*Akademický rok:* **2021/2022**

**Zásady pro vypracování:**

1. Proveďte důkladnou rešerši zobrazovacích panelů a jejich grafiky pro palivové i elektrické motocykly.
2. Vytvořte grafický design hlavní obrazovky včetně uživatelského a servisního menu. Zohledněte denní a noční režim. Použijte pro návrh grafiky některý z pokročilých nástrojů např. TouchGFX a některou vývojovou desku s mikrořadičem řady STM32H7.
3. Vytvořte firmware podle předem navrženého vývojového diagramu v programovacím jazyce C tak, aby grafika displeje reagovala na CAN zprávy a digitální/analogové signály. Navrhněte a sestavte shield na nepájivém kontaktním poli pro připojení CAN sběrnice a digitálních/analogových signálů k vývojové desce.
4. Otestujte zobrazovací panel v simulovaných podmínkách a zhodnoťte dosažené parametry.

| | |
|---|---|
| *Rozsah grafických prací:* | dle potřeby dokumentace |
| *Rozsah pracovní zprávy:* | 30–40 stran |
| *Forma zpracování práce:* | tištěná/elektronická |
| *Jazyk práce:* | Čeština |

**Seznam odborné literatury:**


[1] User manual STM32H747I-DISCO [online]. [cit. 2020-10-7]. Dostupné z: https://www.st.com/resource/en/user_manual/dm00504240-discovery-kit-with-stm32h747xi-mcu-stmicroelectronics.pdf

[2] TouchGFX [online]. [cit. 2020-10-7]. Dostupné z: https://www.touchgfx.com/

[3] Muhammad Ali Mazidi, Shujen Chen, Eshragh Ghaemi. STM32 Arm Programming for Embedded Systems (Volume 6). 2018. MicroDigitalEd. p. 378. ISBN: 978-0997925944.


| | |
|---|---|
| *Vedoucí práce:* | Ing. Lukáš Krčmář<br>Ústav mechatroniky a technické informatiky |
| *Konzultant práce:* | Ing. Petr Bílek, Ph.D.<br>Ústav mechatroniky a technické informatiky |
| *Datum zadání práce:* | 12. října 2021 |
| *Předpokládaný termín odevzdání:* | 16. května 2022 |

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2021

# Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

23. dubna 2022 Vojtěch Hartan

## Poděkování:

Rád bych poděkoval těm, kteří mi byli nápomocni při vytváření této práce a díky nimž jsem byl schopen dovést práci do zdárného konce.

# Abstrakt:

Práce se zaměřuje na vlastní návrh grafiky displeje do elektrických motocyklů. Cílem je vytvoření vhodné grafiky pro displej kontroleru, který bude komunikovat přes rozhraní CAN bus a bude možné přes příslušný software podle potřeb a dle přijatých dat ovládat grafické rozhraní. Práce je rozdělená do dvou částí – komunikace a grafika. Konečným řešením je spojení těchto dvou částí ve funkční celek a otestování na elektrickém motocyklu.



**Klíčová slova:** STMicroelectronics, CAN, grafika, motocykl, přístrojová deska, kontrolér

# Abstract:

The work focuses on the design of display graphics for electric motorcycles. The goal is to create a suitable graphic for display of controller that will communicate via the CAN bus interface and it will be possible to control the graphic interface via the appropriate software according to a received data. The work is divided into several parts – communication and graphics. The final solution is to combine these two parts into a functional unit and test it on electric motorcycle.



**Key words:** STMicroelectronics, CAN, graphic, motorcycle, dashboard, controller

# Obsah

# Seznam obrázků

# Seznam zdrojových kódů



# Seznam tabulek

# Seznam zkratek

| | |
|---|---|
| USB | univerzální sériová sběrnice, připojení periferií |
| CAN | Controller Area Network, sběrnice |
| ST, STM | STMicroelectronics, firma |
| TFT | Thin Film Transistor, způsob vykreslení obrazu |
| LCD | Liquid Crystal Display, zobrazovací zařízení |
| RAM | Random Access Memory, druh paměti zařízení |
| RGB | Red Green Blue, barevný model |
| Tx | Transmit, CAN signál odesílající data |
| Rx | Receive, CAN signál přijímající data |
| CANH | CAN High, diferenciální signál CAN sběrnice |
| CANL | CAN Low, diferenciální signál CAN sběrnice |
| RTOS | Real Time Operating Systém, operační systém reálného času |

# Úvod

Společně s vývojem techniky se posouvá i automobilový průmysl. Mění se normy a jistoty, na které jsme zvyklí. Výkon motorů automobilů i motocyklů je neúměrně vyšší, než by si průkopníci tohoto odvětví mohli představit. Nejde jen o výkon, ale také i o využití nových a daleko více pokročilých materiálů, postupů a v dnešní době i typu pohonu. Společně s tímto vývojem rostou i nároky zákazníků na vybrané produkty. Automobilové společnosti jsou si toho vědomy, a tak kladou důraz na neustálý vývoj od motorové části až po tu komfortní. Tohoto si lze všimnout na infotainmentu vozidel a motocyklů. Bluetooth je v současné době ve vozech nižší cenové relace základ. Stejně tak i GPS navigace, která byla dříve drahý nadstandard, se v dnešní době přidává do vozů a motocyklů téměř základně. A k tomuto patří i palubní deska. Nový trend udává, že klasické budíky jsou spíše historie či přežitek, a tíhne se k nahrazením displeji.

Doba se posouvá a svět se pokouší nahradit analogovou techniku tou digitální. Tachometr není výjimkou. Počátky jsou u menších displejů mezi tachometry a pokračuje to až k dnešnímu stavu, kdy velká část automobilů a motocyklů nemá klasickou analogovou ručičku, nýbrž vše je zobrazováno na obrazovce digitálně. Tato práce je snaha vytvořit a najít nejlepší řešení na toto téma. Vytvořit pomocí displeje a jednoho zařízení zobrazovač stavů a rychlosti motocyklu.

# 1. Rešerše používaných zařízení, metod a postupů pro vývoj grafiky displeje

## 1.1. HMI – Human-Machine Interface Displej

Jak název napovídá, jedná se o rozhraní mezi člověkem a strojem či systémem. Představuje prostředky pro předávání a zobrazování informace o stavu konkrétního zařízení. Mohou to být hodnoty nebo stavy, ve kterých se systém nachází. Mezi výhody HMI patří široká škála využití a provedení. Od jednoduchých a odolných zpracování například pro těžební průmysl, až po složitá a grafická provedení pro zobrazení stavů výrobních hal. Tato zařízení disponují širokými komunikačními rozhraními jako například Profi-bus, RS232, USB, CAN a mnohá další.

Jak již bylo výše zmíněno, díky širokému výběru provedení si lze volit HMI displeje například pro práci s arduinem, což je obdoba řešení této práce, nebo přímo pro spojení se zařízením. Ceny se pohybují od několika stovek korun českých za jednoduchý displej až po desítky tisíc za složité zařízení. Výhodou tohoto řešení by mohla být možnost vytváření grafického rozhraní a programu přímo v zařízení. Bohužel velkou nevýhodou je málo vyskytující se možnost připojení pomocí CAN rozhraní, což je pro automobilový průmysl, respektive cíl této práce, nevyhovující a jeho všeobecně známá vyšší cena. Jako příklad mohu uvést displej Nextion USART HMI TFT LCD – 2.4“, rozlišení 320 × 240 pixelů, 65 tisíc barev, cena 650 Kč, viz obrázek 2.

Další možnost je například robustní IDEC HG5G – 15“ viz obrázek 1, jehož rozlišení je 1024 × 768 pixelů, minimální výdrž 100000 hodin, cena 57500 Kč.

*Obrázek 2: 2.4" displej Nextion USART HMI [2]*


*Obrázek 1: Dotykový displej IDEC HG5G [1]*

Bohužel ani jeden z těchto displejů nenahradí programovatelnou desku od společnosti STMicroeletronics. Problémem může být ať už velikost displeje, kvalita, rozhraní nebo cena.

## 1.2. Univerzální displeje

Další možností pro uživatele je koupit již přednastavený displej. V tomto případě není nutné nic programovat ani nastavovat, pouze se připojí přes dané rozhraní a může fungovat. Toto řešení je přívětivé pro jedince, kteří buď nejsou technicky zdatní nebo chtějí mít rychlé a především jednoduché řešení. Tyto displeje se dají koupit již od několika stovek korun českých.

Nevýhodou je nemožnost si jakkoli displej upravit a překonfigurovat, takže grafika i provedení jsou velice jednoduché a předem pevně dané. Jako příklad je možné uvést Palubní displej 2,8“ STU SE-163 viz obrázek 4, cena 1646 Kč, napájecí napětí 12 V, fyzické rozměry 83 × 54 × 18 mm. Displej se připojí do diagnostického konektoru OBD a veškeré údaje si vyčítá sám z datové sběrnice vozidla.


*Obrázek 4: Palubní displej STU SE-163 [3]*


*Obrázek 3: Univerzální displej [4]*

## 1.3. Analogový tachometr

Samozřejmostí je stále prodávající se analogová technika. Nejenže je to již dlouhými lety ověřená technologie, ale vznikají i tendence vracet se k užívání starých věcí z důvodu svého osobního stylu a mnohdy i nostalgického pocitu. Měl bych samozřejmě ještě rozlišit tyto mechanické budíky, kdy v dávné historii docházelo zobrazováním hodnot, například tlaku, opravdovým průchodem oleje zařízením. V nedávné, ale i v dnešní době jsou to spíše již jen mechanické zobrazovače, které data dostávají z řídící jednotky. Ačkoli mají tyto budíky svoji duševní hodnotu a ukazují nezkreslenou realitu, svých nevýhod oproti budíkům digitálním mají nemálo. Nelze si informace a jejich zobrazování jakkoli upravit, posunout, změnit barvu či pozadí a všeobecně vlastní nastavení nepřichází v úvahu. Jako příklad bych uvedl Analogový tachometr Biker’s choice viz obrázek 5, který je právě prodáván na online portálu Amazon za cenu 108 $, což může odpovídat ceně samotného mikrokontroleru, který byl použit k vypracování této bakalářské práce.


*Obrázek 5: Analogový tachometr Biker's choice [5]*

## 1.4. Vlastní vývoj

Samozřejmostí zůstává vlastní vývoj a využití společností třetích stran. Nelze více nahlédnout do jejich interních záležitostí, avšak je známo, že často využívají TFT LCD displejů, které využívají tenkého tranzistorového filmu. To řídí jednotlivé pixely a nese s sebou spoustu výhod jako kvalitní obraz, vysoký kontrast, nižší spotřebu energie a malé rozměry. Například tuto technologii užívá společnost Ducati, která tento displej často vkládá i do svých motocyklů. Spolu s Ducati i společnosti jako BMW, KTM nebo například Aprilia využívají služeb softwarových společností, takže jsou jejich systémy velice podobné, ne-li často identické [6].

Za zmínku stojí virtuální kokpit společnosti Audi, která se hlásí k vlastnímu vývoji a použití čipu Tegra 30 od společnosti NVIDIA, se kterou má Audi dlouholetý partnerský vztah. Základ tvoří TFT displej s úhlopříčkou 12,3 palce. Rozlišení displeje je 1440 × 540 pixelů, díky němuž dosáhnou ostrého obrazu s výrazným a kontrastním zobrazením. Mezi jeho přednosti patří animace a světelné efekty. Virtual kokpit od společnosti Audi viz obrázek 6 disponuje dvěma uživatelskými rozhraními, mezi kterými je uživatel schopen přepínat pomocí multifunkčního volantu. V režimu infotainment jsou upřednostněny informace a nabídky jako telefon, ovládání audiosystému nebo například mapa. Otáčkoměr a rychloměr jsou zobrazeny jako malé kruhové budíky po stranách displeje. V režimu Classic je středové pole menší a rychloměr s otáčkoměrem jsou upřednostněny jako u tradičních přístrojových desek. Výhodou tohoto řešení je rozmanitost dostupných informací pro uživatele, dynamické animace o stavu vozidla, obrazy zachycené parkovacími kamerami a dále [7].


*Obrázek 6: Audi Virtual Cocpit*

## 1.5. Vývoj grafiky

Pro vývoj grafické části pro obrazovky motocyklů nebylo nalezeno mnoho dedikovaných softwarů. Za zmínku stojí určitě program, který se daným tématem zabývá – Rightware Kanzi. Je to editor uživatelského rozhraní, ve kterém se dá jednoduše vytvořit prostředí pro různé aplikace [8].


*Obrázek 7: Rightware Kanzi [8]*

Velká nevýhoda tohoto programu je jeho nedostupnost. Jednak se přístup k softwaru uděluje a zároveň cena je pro mé řešení astronomická. Pro Android aplikace je cena 999 $ a plná cena programu je 10999 $.

Další možností je LVGL – Light and Versatile Graphic Library. Je to knihovna obsahující vše potřebné pro vytvoření plně vybavených graficko – uživatelských rozhraní. Lze díky ní vytvořit vizuálně atraktivní grafiku na vestavěných zařízeních, pro která je především určena. Vyčnívá svým využitím pro mikrokontrolery s omezenými zdroji díky jednoduchým grafickým elementům a nízkého využití paměti. Mezi její vlastnosti patří předpřipravené objekty jako tlačítka, grafy a tak dále. V jejím repertoáru můžeme najít i pokročilá grafická jádra podporující základní funkce. Můžeme využít podporu pro velkou škálu periferií a v neposlední řadě podporu pro vícero displejů [9]. Grafické prvky jsou členěny do stromové struktury stylem rodič-potomek. Rodič může obsahovat dynamický počet potomků, a každý potomek se může stát rodičem pro další podřazenou strukturu. Celý projekt má pak hierarchický charakter, kde je prvek bez rodiče nazýván stránkou.

```
/* Vytvoření tlačítka na danou obrazovku */
 lv_obj_t * btn = lv_btn_create (lv_scr_act ());
 /* Nastavení pozice */
 lv_obj_set_pos (btn,15,15);
 /* Nastavení velikosti */
 lv_obj_set_size (btn,150,70);
 /* Assign a callback to the button */
 lv_obj_add_event_cb(btn,btn_event_cb,LV_EVENT_CLICKED,NULL);

 /* Přidání popisku tlačítku */
 lv_obj_t * label = lv_label_create(btn);
 /* Přidání popisku tlačítku */
 lv_label_set_text(label," Button ");
 /* Nastavení pozice popisku */
 lv_obj_center(label);
```

*Kód 1: LVGL knihovna – přidání tlačítka a vytvoření metody stisku*

V neposlední řadě by se měl zmínit QT pro mikrokontrolery. Je to kompletní grafický framework a soubor nástrojů se vším, co by mohlo být třeba pro návrh, vývoj a nasazení grafického rozhraní do mikrokontrolerů. Hlásí se již k 25 letům vývoje designu na milionech zařízení [10].

Další možnosti, jak si své zařízení více oživit, jsou již známé placené nebo volně přístupné grafické editory, jejichž nevýhodou je však nemožnost čerpat z již vytvořených grafických prvků, nemožnost je programově editovat a jejich využití by bylo čistě pro vytvoření objektu, který se následně vloží do grafiky. Mezi příklady lze uvést Adobe Photoshop, Inkscape, Gimp a mnoho dalších.

# 2. STMicroelectronics

## 2.1. Úvod

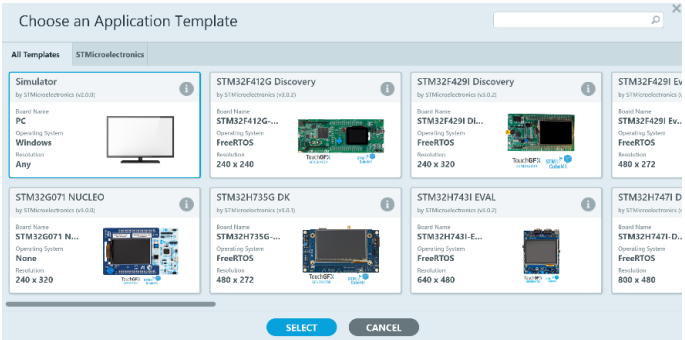
STMicroelectronics, je francouzsko-italská firma se sídlem v Ženevě, Švýcarsku. Vzhledem k příjmům je to největší výrobce polovodičových čipů v Evropě. Vyrábí široké pole desek a mikročipů. Nekončí jen u tohoto, snaží se i o inovace ve světě chytré mobility, energie a internetu věcí [11]. Právě na této společnosti a jejich softwaru stojí tato bakalářská práce, při které bylo využito třech různých typů desek a dva různé softwary, které budou popsány níže v této práci.

## 2.2. TouchGFX

TouchGFX je pokročilý grafický software optimalizovaný pro STM32 mikrokontrolery. Pomocí tohoto nástroje je možné vytvořit a zároveň částečně naprogramovat grafické rozhraní pro jejich přístroje. Jeho přední výhodou je bezplatná licence, předpřipravené objekty, hierarchická strukturalizace projektu, ovládání grafiky, textů, přidávání vícero „obrazovek“ a dále. Některé jeho prvky budou k nahlédnutí dále v této práci.

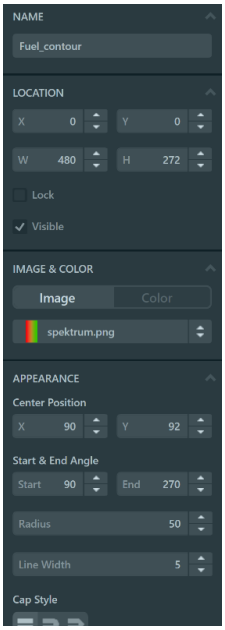

*Obrázek 8: TouchGFX [12]*

Jako první při otevření aplikace lze vidět okno, kde je možné založit nový projekt, otevřít již vytvořený ba dokonce se nechat inspirovat online projekty. Při vytváření nového projektu si z intuitivního výběru vyberete desku, se kterou bude software pracovat, a TouchGFX si již doplní nutné informace jako například rozlišení displeje. Po napsání názvu je projekt vytvořen.




*Obrázek 9: TouchGFX – výběr kontroleru*

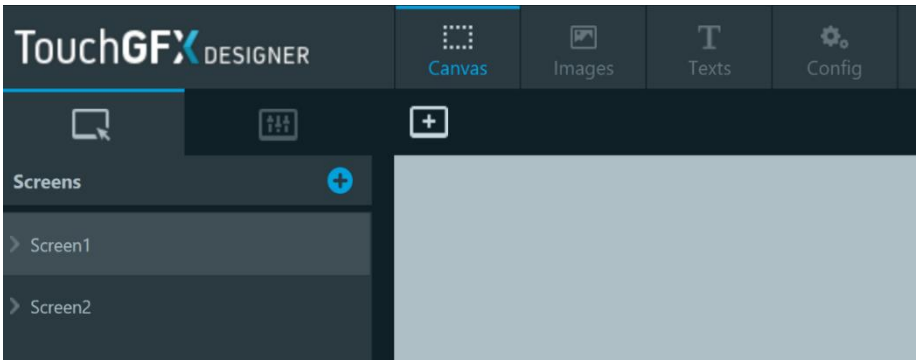
Ve chvíli, kdy je projekt vytvořený, je možné přidávat objekty z již předpřipraveného výběru. V této práci bylo často využíváno objektů v podkategorii tvarů, jako jsou line, což by bylo možné volně přeložit jako čára, circle neboli kruh, shape jakožto tvar. Zmíněné objekty sloužily jako taková stavební jednotka displeje a je s nimi vytvořena většina částí grafického rozhraní. Samozřejmostí je jejich široká nabídka úprav, kdy například kruhu lze nastavit průměr, tloušťku okraje, počátek a také konec, výplň, barvu a tak dále. Ještě bych vyzdvihl objekt Image jakožto obrázek, který byl použit při vytvoření složitějších obrazců.




*Obrázek 10: TouchGFX – nastavení objektu*


*Obrázek 11: TouchGFX – objekty*

Vložení objektu na pracovní plochu dojde vybráním nebo přetažením. V tu chvíli program již ví, že byl přidán ovládací prvek a dovolí s ním pracovat. Sám mu přiřadí jméno, které lze následně změnit.




*Obrázek 12: TouchGFX – výběr obrazovky a nastavení*

Jak je možné vidět, viz obrázek 10, daný objekt má několik možností nastavení. V tomto konkrétním případě je to objekt kruh, který má název, umístění, jako barva je vložen obrázek a tloušťku obvodu kruhu. Nastavením začátku a konce z něj lze vytvořit půlkruh.

Jak ukazuje obrázek 12, lze si všimnout přepínání obrazovek, které byly již vytvořeny nebo přidávat nové. V horní liště je přepínání mezi pracovní plochou, nastavením nebo vkládáním a uložením obrázků, které je důležité pro využití objektu image. Zde se ukládají veškeré obrázky, které je třeba v projektu použít, a nelze je využít, pokud nebyly uloženy právě zde. Poslední výběr je nastavení textového formátu, kde si lze uložit definované styly textu mezi které patří font, velikost a tak dále.

Daným objektům lze přiřadit jednoduché, předem definované interakce mezi sebou. Pro ty složité máte možnost otevření kódu ve Visual Studiu, kde je možnost naprogramování složitějších a komplexnějších metod. Je zde i možnost simulace připraveného grafického prostředí, aby šlo vyzkoušet, jak se kód chová, což bylo pro mě využitelné pouze zpočátku, jelikož celé konečné řešení využívá komunikace a ta se zde nasimulovat nedá.

Jeho výhodou je spolupráce s dalším softwarem z rodiny ST – CubeIDE, který je popsán níže. Při propojení těchto dvou softwarů a vygenerování kódu v TouchGFX se automaticky veškeré úpravy v grafickém rozhraní aktualizují v CubeIDE.

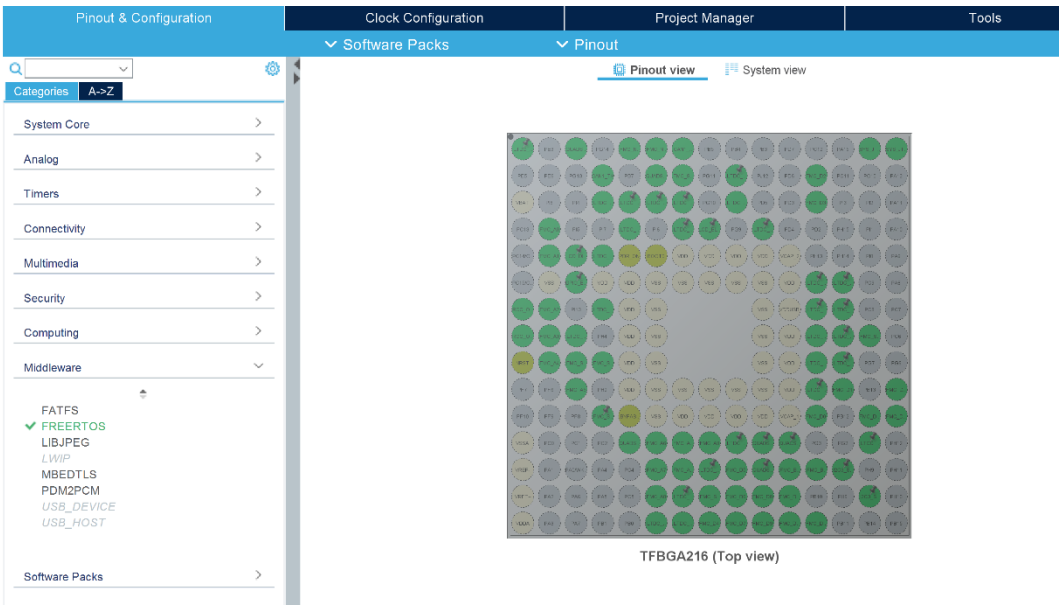
Velkou nevýhodou tohoto softwaru je nemožnost pokročilejších a propracovanějších grafických zobrazení, jaké můžeme vidět v moderních digitálních přístrojových deskách. Jako konkrétní příklad lze uvést, že jsem v řešení chtěl přidat stíny, které by aplikaci dodaly hloubku a lepší vzhled, avšak pro toto není nástroj. Výběr je omezen pouze na předpřipravené objekty, které lze nakonfigurovat, ale nedá se s nimi zpracovat úplně vše. Možností by bylo vytvořit ve vektorové grafice obrázek stínu a ten následně vložit, jenže to by ve větším množství mohlo zatěžovat procesor. Tento software tedy není vhodný pro extrémně náročné grafické aplikace, což se projevilo i na výsledku.

## 2.3. CubeIDE

Jako v předchozím případě je tento software určen pro ST mikrokontrolery. V tomto programu je možné vyvíjet programy pro desky, nastavovat je, zapínat a vypínat jednotlivé piny a celkově zajistit chůzi daného kontroleru s možností sjednotit ovládání systému s grafickým rozhraním. Jeho předností je jeho přehlednost a možnost nastavení jednotlivých detailů chodu mikrokontroleru.


*Obrázek 14: CubeIDE - nastavení CAN*

Jakmile je mikrokontroler vybrán, dostane se uživatel do menu viz obrázek 15. V něm se skrývá nastavení celé desky. V této části si uživatel může nastavit veškeré fungování mikrokontroleru a jednotlivé piny, s kterými software CubeIDE a deska má pracovat. Jsou zde kategorie, ve kterých se jako příklad může uvést FREERTOS, který je aktuálně zeleně označen. To je znamení, že je tato část aktivní. V části connectivity, neboli konektivita, je například samotné nastavení CAN rozhraní viz obrázek 14. Je důležité si zde nastavit veškeré maličkosti. Chyba zde způsobí nefunkčnost kódu, potažmo nefunkčnost celého projektu.




*Obrázek 15: CubeIDE*

# 3. Řešení

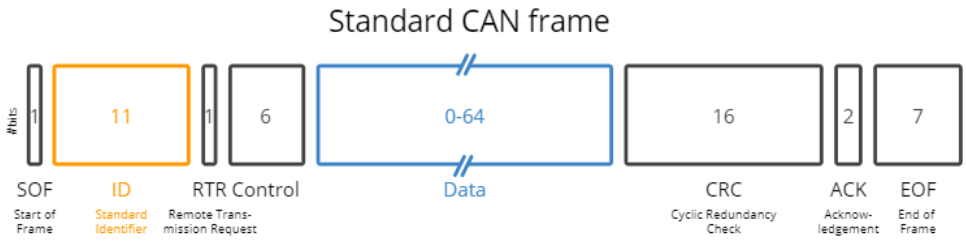
## 3.1. Úvod

Při vyhotovení této práce došlo na užití rovnou tří mikrokontrolerů od STMicroelectronics. Jsou to STM32H747I-DISCO, STM32F769I-DISC1 a STM32F746G-DISCO. Důvodem je nemožnost vytvoření funkčního řešení u prvních dvou zmíněných desek. Důvody jsou popsány dále v této práci. Důležité bylo navrhnout grafickou část a spojit ji s tou komunikační. Obě části se píšou a navrhují v rozdílných softwarech, a právě v tom byla tato práce náročná.

## 3.2. CANbus

CAN bus je sériová datová sběrnice využívána v průmyslu ale i automobilním světě. Získala svou popularitu díky mnoha výhodám a byla vyvinuta firmou Bosch v roce 1986. Vyznačuje se dvěma vodiči – CANH a CANL, jejichž význam je právě v přenosu dat. Při přenosu dat se napětí na CANH zvedne na 3.75 V a u CANL klesne na 1.25 V, čímž vytvoří rozdíl napětí o velikost 2.5 V, a tím dojde ke změně stavu [13].

CAN definuje takzvaný CAN rámec zprávy, které má z pravidla 7 polí.




*Obrázek 16: CAN rámec [14]*

První bit je Start of Frame, neboli počátek zprávy, pak následuje takzvané rozhodovací pole, které obsahuje identifikační číslo, které označuje prioritu zprávy, čím menší tím důležitější. Následuje RTR neboli Remote Transmition Request – žádost o vzdálený přístup, který ukazuje, zda jde o rámec dat nebo o požadavek rámce bez bitů. Následuje kontrolní pole, které označuje, zda se jedná o standartní rámec nebo o prodloužený formát. Dále je rezervovaný bit r0 pro budoucí prodloužení. V tuto chvíli přichází na řadu data, jelikož další část je Data Length Code, který udává, kolik bytů zpráva obsahuje a samozřejmě, která data nese. Po datech přichází na řadu CRC, což je bezpečnostní kontrola na detekci bitových chyb. Na konci každé zprávy bývá ještě ACK, které udává, zda příjemce přijal zprávu v pořádku [15].

Výhodou užívání CAN bus je například možnost využití standartního rámce o velikosti ID 11 bitů nebo prodlouženou verzi, kdy je délka označení zprávy 29 bitů. Mezi další výhody může být uvedena robustnost, což z této sběrnice dělá ideální volbu v automobilovém průmyslu. Snižuje počet nutných drátů, její rychlost a účinnost z důvodu označení důležitosti zpráv a další.

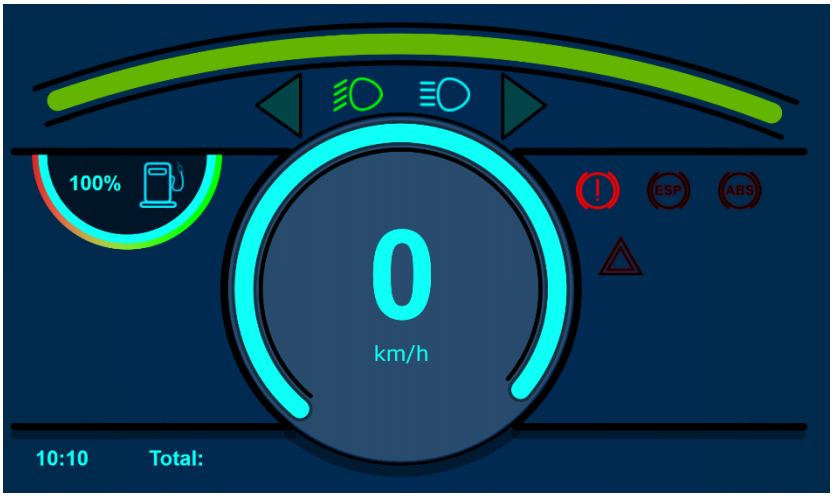
## 3.3. STM32H747I-DISCO

První kontroler, se kterým byla práce vypracovávaná. Dvoujádrový procesor s 2 megabytovou flash pamětí a 1 megabytovou pamětí RAM. Tato deska je osazena čtyř palcovým dotykovým LCD displejem s rozlišením 800 × 480 pixelů. Důvodem, proč by mikrokontroler nemusel přijímat CAN data, mohl být chybějící CAN konektor. Tento problém se dá vyřešit FANOUT deskou od stejné společnosti, která těmito piny osazena byla a mohla být připojena k mikrokontroleru přes UART rozhraní.

*Obrázek 18: STM32H747I-DISCO [16]*

*Obrázek 17: FANOUT board [17]*

Při práci s touto deskou bylo dosaženo téměř hotového grafického rozhraní. Výhodou této desky byl relativně ostrý displej s vyšším rozlišením, než má mikrokontroler, se kterým byla práce vyhotovena a dokončena. Také byla osazena ovládacími prvky, například tlačítky, které mohly být použity při ovládání grafiky. Právě zde jsem navrhl a vyhotovil svou představu o zobrazovacím rozhraní. Navrhl jsem velký tachometr umístěný ve střední části obrazovky. Levá část byla vyhrazena pro ukazatele stavu nabití motocyklu. Do pravé části byli umístěny informační ikony. Horní část měla sloužit pro zobrazení využití výkonu motocyklu, spolu s ikonami světel a blinkrů. Dolní okraj displeje byl vyhrazen pro informace například času a ujetých kilometrů.




*Obrázek 19: H747 – návrh grafiky*

Bohužel problém nastal ve chvíli, kdy jsem se pokusil o naprogramování komunikace. Nevýhodou byl chybějící podrobnější návod. Nastavení desky bylo úspěšné, ale nedošlo k propojení TouchGFX s CubeIDE, neboli propojení komunikace s grafikou.

Tento problém nešlo vyřešit. Software neustále odkazoval na chyby, které nebyly validní. Důvodem byla chybějící podpora ze strany ST pro tuto desku, a tak nebylo možné dosáhnout propojení, což znamenalo, že by práce nebyla dovedena do zdárného konce. V případě, že by společnost opravila nebo dodala podporu pro tuto desku, mohl by se tento projekt vyhotovit právě s tímto přístrojem.

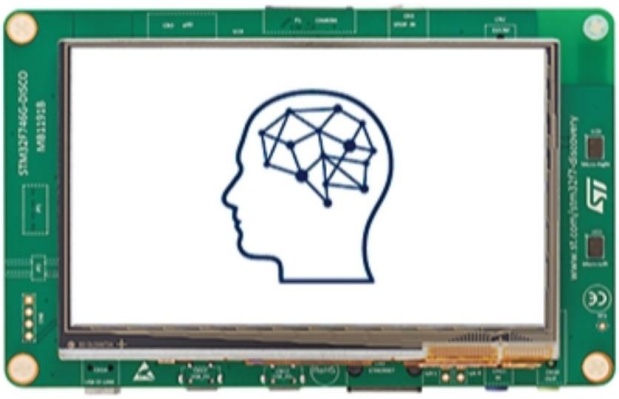
## 3.4. STM32F769I-DISC1

Jakmile bylo jasné, že s předchozí deskou nebylo možné vyhotovit moji bakalářskou práci, bylo nutné pokusit se využít jiný mikrokontroler. Volba přišla právě na STM32F769I-DISC1. Ačkoli produktový název napovídá, že deska displej nemá, tak byla osazena LCD displejem s rozlišením 800 × 480 pixelů. Avšak již od začátku mikrokontroler vykazoval abnormální chování, jako například vypínání a nemožnost aplikaci spustit, což se nakonec vysvětlilo poničeným čipem, který se přehříval. Z tohoto důvodu muselo dojít k výměně přístroje.


*Obrázek 20: STM32F769I-DISC1 [18]*

## 3.5. STM32F746G-DISCO

Nezdary, uvedené v předchozích odstavcích, projekt přivedly k už druhé změně a třetímu mikrokontroleru. Třetí, již konečnou volbou byla deska STMF746G-DISCO, která využívá procesor s jedním jádrem. Obsahuje 1 megabytovou flash paměť a 340 kilobytovou paměť RAM. Je osazena 4.3“ RGB LCD displejem s rozlišením 480 × 272 pixelů. Důležitou výhodou bylo osazení CAN rozhraním.



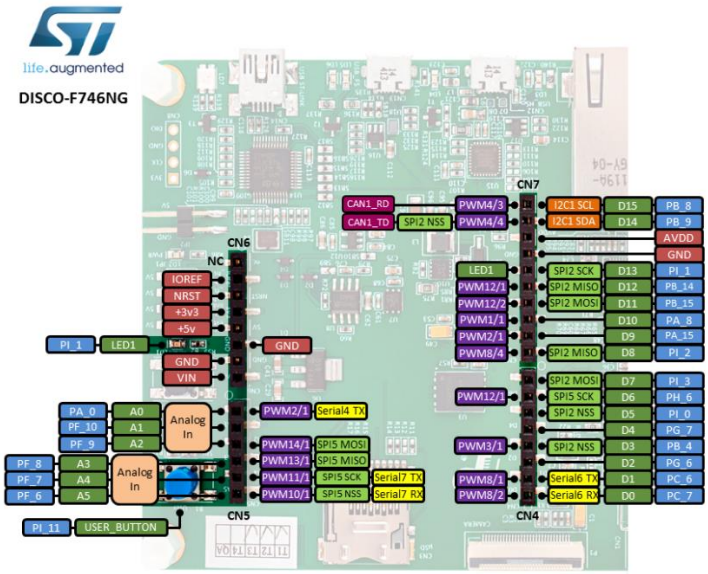
*Obrázek 21: STM32F746G-DISCO [19]*

Oproti předchozím mikrokontrolerům F746G zaostává jak v pamětích, tak rozlišením displeje, avšak pro tuto desku existuje plná podpora od ST, takže bylo možné bakalářskou práci s tímto zařízením vyhotovit.

## 3.6. Zapojení

Většina desek má svoje piny, které lze chápat jako vývody kontroleru či součástky, obvykle nějakého integrované obvodu. Význam jednotlivých pinů se dá najít v datovém listu daného obvodu a můžeme je obecně nazvat pinout. Tyto informace jsou pro uživatele důležité, jelikož díky nim je schopen desku správně připojit a používat.

V mém konkrétním případě má STM32F746 piny na spodní straně desky. A jejich význam lze vidět na obrázku níže. Obzvláště důležité jsou piny PB_8 a PB_9, na které byly přivedeny komunikační vodiče. Z důvodu užití CAN budiče byly ještě využity piny GND, neboli země, a také vývodu napětí na 3.3 V a 5 V.




*Obrázek 22: Pinout STM32F746 [20]*

Pro správné fungování nebylo možné jednoduchého propojení kontroler – motocykl. Bylo třeba přidat mezistupeň. Tímto mezistupněm je takzvaný CAN budič, konkrétněji CJMCU – 1051. Obvod CJMCU-1051 je vysokorychlostní vysílač a rozhraní mezi CAN řadičem a fyzickou sběrnicí, která zajišťuje diferenciální vysílací a přijímací funkce pro CAN řadič.


*Obrázek 23: CJMCU 1051 [21]*

Propojení bylo následovné:

*Tabulka 1: Zapojení*

| CJMCU1051 | STM32F746 |
|---|---|
| Vcc | 5V |
| GND | GND |
| CTX | PB9 |
| CRX | PB8 |
| CANH | - |
| CANL | - |
| VIO | 3.3V |

CTX a CRX funguje jako vstup a výstup CAN rozhraní do mikrokontroleru a CANH a CANL jsou vstupy a výstupy pro motocykl.

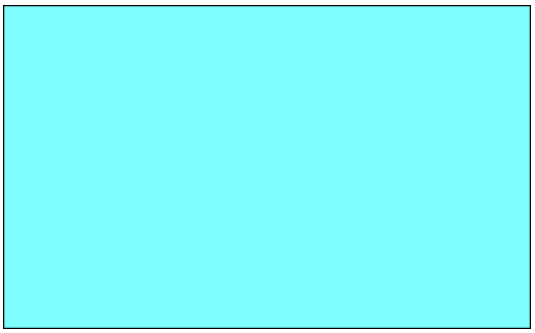
## 3.7. Grafická část

Jak bylo zmíněno v předchozích kapitolách a odstavcích, grafická část a celý její návrh se odehrává v TouchGFX. Proto bylo zapotřebí nejdříve vytvořit projekt, který byl nahrán do CubeIDE, aby došlo k propojení těchto dvou platforem. Ve chvíli, kdy došlo k propojení, tak bylo možné začít navrhovat grafickou část displeje.

Jelikož představa, jak bude displej vypadat, byla již navržena, využil jsem původní návrh a pouze jsem ho upravil pro rozlišení této desky. Avšak projevila se nedokonalost softwaru TouchGFX, čímž je nemožnost kopírovat objekty mezi projekty, což znamenalo, že cokoliv bylo již připraveno z prvního projektu, tak bylo třeba vytvořit znovu.

Jako základní kámen práce bylo nutné připravit vhodné pozadí. Jelikož TouchGFX nepodporuje komplexnější grafické úpravy, vytvořil jsem ve vektorové grafice pozadí celé palubní desky. K tomuto úkolu byl užit Inkscape, což je otevřený editor vektorové grafiky. Došlo k úpravám původního návrhu z důvodu rozdílného rozlišení LCD displeje a tím pádem i velikosti rámu. Aby došlo alespoň k mírné hloubce displeje, návrh dostal decentní stínování, které není na první pohled vidět, ale ve výsledku je to pohlednější volba.

*Obrázek 25: Tmavé pozadí*



*Obrázek 24: Světlé pozadí*

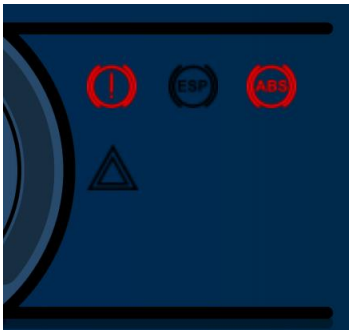
Již při tomto návrhu byl respektován denní a noční režim, který se právě odvíjí od světlosti použitých barev. Vektorová grafika byla vyexportována jako grafický formát png, který byl v TouchGFX vložen jako podklad pro tachometr pomocí objektu Image.

Po vytvoření podkladu ve formě pozadí bylo nutné navrhnout zobrazení rychlosti. Toho bylo docíleno využitím objektu Circle, který na pracovní plochu vloží kruh. Na další straně si můžeme všimnout rovnou 9 objektů kruhů. Nejvíce viditelný je ukazatel rychlosti a světlý podklad tachometru, který byl navržen pro zvýraznění celého kruhu a pokusu o prohloubení, respektive vytažení. Vlevo a vpravo jsou dva kruhy, které mají zadaný pevný začátek, konec a tloušťku, to znamená, že je to spíše čtvrtkruh, který má za úkol ohraničení tachometru. To samé se odehrává v horní a dolní části tachometru, jediný rozdíl je v tloušťce kruhu. K dolní části byl přidán ještě jeden, kterému byla nastavena průhlednost, aby došlo k efektu stínování. Poslední tři objekty kruhu umístěné v centru obrazovky jsou vyhrazeny na vnitřní ohraničení tachometru, podklad, který má opět nastavenou průhlednost a samotný ukazovač rychlosti, který se pohybuje ve směru hodinových ručiček v závislosti na rychlosti. Zmíněný objekt v návrhu zastupuje klasickou analogovou ručičku. Ve středu kruhů je velkými čísly zobrazena rychlost a jednotky.


*Obrázek 26: Zobrazení rychlosti*

Další nedílnou součástí grafického návrhu displeje jsou informační ikonky, kterým je vyhrazena celá pravá část displeje. V době návrhu nebylo týmem, který zhotovuje motocykl, řečeno, které ikony mají být zobrazovány, a tak bylo navrženo několik ikon, které budou fungovat jako názorná ukázka, jak by konečný výsledek vypadal. Z důvodu nedostatku kvalitních obrázků ikon pro automobilový průmysl jsem byl nucen opět zvolit program Inkscape a nakreslit je ve vektorové grafice. Obrázky, potažmo ikony, byly vždy vyhotoveny v páru – červená a černá. Červená samozřejmě značí aktivaci a černá byla využita jako překrytí, aby o nich uživatel věděl. Jako v předchozím případě, vektorová grafika je exportována do png formátu, který byl na pracovní plochu vložen jako objekt Image. Vždy je vidět buď červená nebo černá verze.

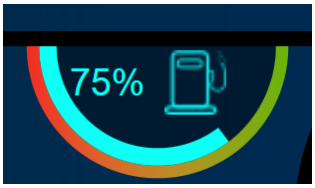
Navzájem si v případě nutnosti vymění průhlednost na 100 procent nebo na 0 procent. Výsledek můžeme vidět níže.




*Obrázek 27: Ikony*

V levé části obrazovky se nachází místo pro stav baterie – zobrazené tradičně jako nádrž a zároveň zde lze zobrazit mód motocyklu, a sice ECO/Normal/Sport. Zajímavostí může být ukazatel stavu nabití baterie, který je vyobrazen zároveň jako číselná hodnota v procentech, ale také jako ubývající barevný půlkruh. Také je zde vidět, viz obrázek 28, dalšího půlkruhu, a to barevné kombinace červené se zelenou. Zelená plynule přechází do červené. To znamená, že ve chvíli, kdy půlkruhový ukazatel ustupuje, značí tím nutnost dobití baterie.

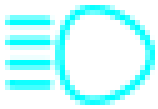
Toho bylo opět docíleno návrhem vektorové grafiky v programu Inkscape, pomocí kterého byl vytvořen grafický objekt, který plynule přechází z červené na zelenou, respektive ze zelené na červenou. Výsledek byl vyexportován a vložen do objektu kruh.




*Obrázek 28: Stav baterie*

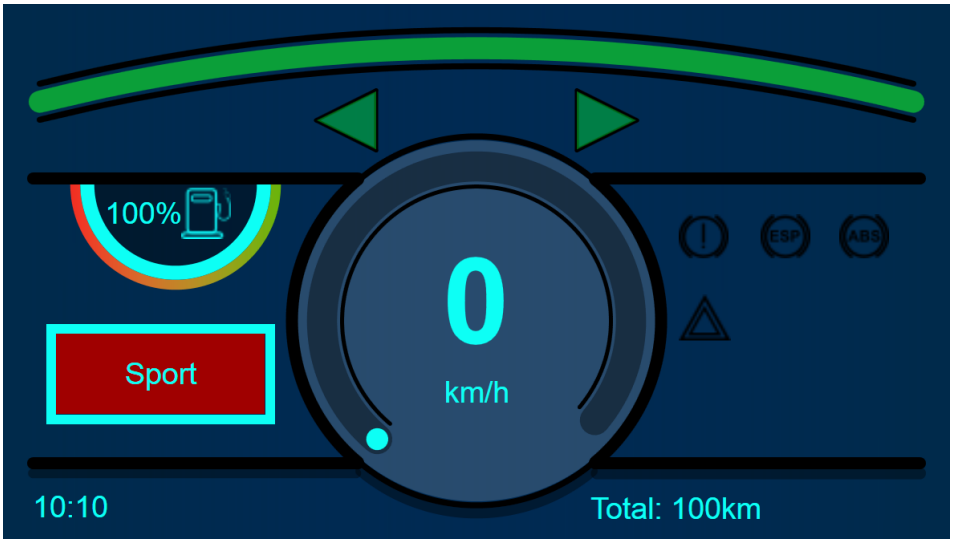
Horní část obrazovky zabírá ukazatel využití síly pohonné jednotky, v klasickém motocyklu by to byl ukazatel otáček. Pod ním se nachází blinkry a ikony světel. Blinkry byly vyhotoveny jako trojúhelníky, které jsou za normálního stavu částečně průhledné a ve chvíli, kdy jsou aktivovány, tak se rozsvítí křiklavě zelenou. Ikony světel jsou vytvořeny stejně jako v předchozím případě, a sice pomocí vektorové grafiky.

*Obrázek 30: Horní část obrazovky*



*Obrázek 29: Ikona dálkových světel*

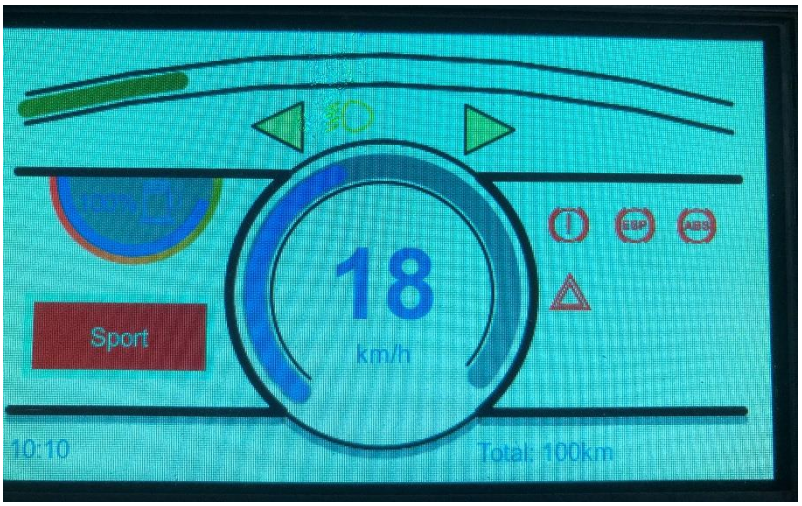
Při spojení všech bloků, které jsou uvedeny v předchozích kapitolách této práce, vznikne hotový grafický základ pro displej motocyklu, který se v zásadě neliší od prvotního návrhu. Můžeme si všimnout, viz obrázek 31, konečného návrhu grafiky pro elektrický motocykl. Toto je tmavá verze, tedy noční režim návrhu. Jednotlivé prvky jsou navrženy tak, aby nebyly rušivé pro uživatele za předpokladu jízdy za snížených světelných podmínek.




*Obrázek 31: Grafika displeje – tmavý režim*

Světlá verze, tedy denní režim, je naprosto totožný s režimem tmavým. Změny se objevují v barvách a jejich užití. Podklad je světlý viz obrázek 32. Užité zobrazovací barvy – světle modrá či tyrkysová se zamění za tmavě modrou, aby bylo docíleno jednoduchého rozlišení zobrazovaných prvků.

Jak jsem již zmínil, software TouchGFX obsahuje i simulátor, který by teoreticky mohl být využit na kvalitní zobrazení světlého režimu, avšak projekt a program reagují pouze na CAN zprávy, proto bylo nutné vložit fotografii desky během chodu. Bohužel vyfocený displej nevynikne tak, jako vyniká v realitě. Přiložená fotografie má za účel ukázat, že nedošlo ke změně rozložení objektů, nýbrž ke změně využití barev.




*Obrázek 32: Grafika displeje – světlý režim*

Poté, co byla vytvořena hlavní část grafiky a ověřena její funkčnost, začal jsem se věnovat také návrhu a vytvoření menu. Bylo nezbytné přidat další obrazovky, které budou schopny mezi sebou přepínat a reagovat na CAN zprávy.

Stejně jako v hlavní obrazovce je zde totožné barevné pozadí, je respektováno využití denního či nočního režimu. A právě zde je možné mezi režimy přepínat. Zároveň si zde lze vybrat mezi jízdními režimy motocyklu. Při vytváření tohoto menu jsem narazil na problém. TouchGFX potažmo CubeIDE není schopno pracovat s objekty obecně. To znamená nemožnost použít obecnou programovou smyčku, pomocí které by bylo možné ovládat všechny objekty v grafickém rozhraní. V hlavní obrazovce zobrazovače rychlosti toto problém nebyl, jelikož jednotlivé objekty bylo v mém zájmu ovládat ručně. Avšak pro hladký chod menu a v zájmu srozumitelného programu by v této části bylo této funkce ovládání třeba. Z tohoto důvodu bylo nutné ovládat a tím pádem psát instrukce pro každý objekt zvlášť. Což může teoreticky být pro mikrokontroler náročné a mohlo by to vést k neoptimalizované aplikaci.


*Obrázek 33: Menu*

## 3.8. Nastavení

Jakmile byla vyhotovena grafická část práce, tak přišel čas na nastavení mikrokontroleru a naprogramování komunikace. Jak bylo zmíněno výše, tato část se odehrává v CubeIDE. Využil jsem tedy již vytvořený projekt z TouchGFX, aby došlo k propojení těchto dvou oddílů a zároveň jsem využil výhody, že se zde již některé informace nastavily. Avšak bylo nezbytné se věnovat i dalším částím, které jsou v krátkosti rozepsány níže. Toto vše se odehrává v.ioc souboru, který je pro CubeIDE něco jako odrazový můstek.

**Connectivity:**
CAN1 – CAN1 RX interrupt

V záložce connectivity lze najít možná nejdůležitější část a tou je aktivace CAN rozhraní. Před aktivací se musí ještě nastavit. V záložce „Parameter Setting“ neboli nastavení parametrů upravit PRESCALE na 4, TIME QUANTA IN BIT SEGMENT 1 na 16 Times a TIME QUANTA IN BIT SEGMENT 2 na 8 Times. Výsledkem je nastavení časové známky, takzvaný baudrate, pro správný chod komunikace. Samozřejmě, jestliže zdroj CAN zpráv má jiné časování, či je domluvená rozdílná konvence, zde se může upravit. Záložka OPERATING MODE nastavit na Normal. Mikrokontroler umí takzvaný Loopback, který jsem využíval v začátcích své bakalářské práce, kdy deska odesílá data a sama tato data přijímá, což může být užito jako testování ovládání.

V záložce Multimedia bylo třeba vyplnit LTDC, což znamená LCD – TFT Display Controller. Díky němuž je grafika schopna reagovat bez užití procesoru.

**Multimedia:**
LTDC – Layer 0 – Blending factor1 – Alpha constant
LTDC – Layer 0 – Blending factor2 – Alpha constant
LTDC – Color frame buffer start adress 0xC0000000

Záložka Middleware obsahuje možnost nastavení RTOS. Jeho hlavní význam je, že obsluhuje vícero úkolů najednou. V jeho nabídce může uživatel vytvářet úlohy a přiřadit jim důležitost. To znamená, že

ve chvíli, kdy procesor vykonává nedůležitý úkol a přijde na řadu důležitější s vyšší prioritou, tak přestane obsluhovat původní zadání a obslouží to důležitější. Po vykonání se opět vrátí k původní práci. Výsledkem je, že přepíná mezi úlohami a postupně je ukládá.

**Middleware:**
FREERTOS
TOTAL_HEAP_SIZE 65536 Bytes

V neposlední řadě bylo třeba se věnovat i obsahu záložky Softwarepacks. Ačkoli by k aktivaci mělo propojením TouchGFX a CubeIDE dojít automaticky, mohlo by se stát, že se tak nestane. Pokud by k aktivaci využití TouchGFX nedošlo, tak v horní záložce „Software packs“ stačí otevřít záložku Manage software packs, kde je možnost vybrat danou verzi TouchGFX. V levé části obrazovky se software přidá do záložky a stačí jej pouze potvrdit.

## 3.9. Komunikace a ovládání grafiky dle přijmutích dat

Pro ujištění, že se navržená a vytvořená grafika chová dle očekávání, bylo třeba mít zdroj nebo generátor CAN zpráv, na které byl program situován. Tímto zdrojem bylo Arduino Due, které pomocí CAN budiče, vodičů a programu bylo schopno zastoupit elektrický motocykl. To práci značně zjednodušilo, jelikož nebylo potřeba být neustále připojen k motocyklu. Ve chvíli, kdy byly ověřeny veškeré grafické prvky práce, začal jsem pracovat na programování komunikace ušité na míru propojení motocykl – mikrokontroler.

Po nastavení desky, které bylo možné vidět v předchozích řádcích této bakalářské práce, se vygeneruje a otevře main.c soubor. Z názvu jde poznat, že právě zde se odehrává hlavní část kódu mikrokontroleru a je to místo, kde lze přijímat data. Rád bych vysvětlil a upozornil na nejdůležitější části programu.

Níže uvedená funkce nazvaná HAL_CAN_GetRxMessage viz kód 2, má za úkol přijímání informací na CAN rozhraní. Po zpracování dat a kontrole validity se informace ihned odešle pomocí „fronty“ pojmenované xQueueSendFromISR dále deskou. Využití fronty je pro řešení esenciální, jelikož využitím RTOSU by mohlo dojít k přepnutí úlohy desky a data by mohla být ztracena. Toto je možný způsob, kterým se ztracení dat zamezí. Jak si lze všimnout, kontroluje se zde i identifikační číslo zprávy, které je uloženo v proměnné pRxHeader.StdId. Toto je rozhodování, zda přijatá data jsou pro zobrazení důležitá či ne. Viz tabulka 2, která je přiložena níže, kde je uvedeno, že jediná data, která má deska přijmout, jsou 111 a 113 hexadecimálně. Vzhledem k užívání datového typu uint8_t, neboli celočíselné hodnotě o délce osmi bitů, na devátou pozici proměnné rxMessage je přiřazena hodnota 1 nebo 2. Pomocí takto přiřazené hodnoty je program schopen v grafické části rozhodnout, který ovládací prvek má na přijatá data reagovat.

```
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan) {
	HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &pRxHeader,
&rxMessage[0]);
	if(pRxHeader.StdId == filter1)
		{
			rxMessage[8] = 1;
			xQueueSendFromISR(messageQ,&rxMessage[0],0);
		}
```

*Kód 2: Main – Přijmutí zprávy, filtrace a odeslání dat ke zpracování*

Následně jsou data odeslána do model.cpp, který by se dal přirovnat mezi-rozhraní mezi GUI, neboli graficko-uživatelským rozhraním a komunikační částí. TouchGFX využívá architektury nazývanou Model – View – Presenter. Model je rozhraní, které definuje data, která mají být zobrazena nebo na která se má jakkoli jinak reagovat. Zároveň se využívá pro komunikaci s vnějším světem. Rozhraní View má za úkol data zobrazovat nebo předávat Presenteru události, na která má reagovat. Presenter je společné rozhraní pro Model a pro View a jedná na základě informacích obdržených od dvou výše zmíněných struktur. V přiložených zdrojových kódech a v odstavcích, které budou následovat, je možné si tohoto propojení všimnout.

```
Model::Model() : modelListener(0)
{
	messageQ = xQueueCreate(10, sizeof(uint8_t));
}

void Model::tick()
{
	if(xQueueReceive(messageQ, &rxMess[0], 0)==pdTRUE)
		{
			for(int i=0;i<9;i++)
			{
				modelListener->setNewValue(rxMess[i]);
			}
		}
}
```

*Kód 3: Model – Odeslání dat ke zpracování*

Výše uvedený kód je inicializace a přijmutí dat z main.c. Je zde deklarována messageQ, což by mohlo být chápáno jako nositel zprávy. Přijatou zprávu neboli hodnotu uloží program do proměnné rxMess. Data se odtud předají dál do modeListeneru, který obsahuje deklarovanou metodu setNetValue. ModelListener.hpp je propojen s hlavičkovým souborem s názvem Screen1Presenter.hpp. Screen1Presenter.cpp si data převezme a odešle do konečné destinace.

```
void Screen1Presenter::setNewValue(unsigned int value)
{
	view.updateVal(value);
}
```

*Kód 4: Screen Presenter – odeslání dat ke zpracování*

V Screen1View.hpp je deklarována funkce updateVal, která ve Screen1View.cpp ovládá grafické rozhraní.

```
void Screen1View::updateVal(unsigned int newValue)
```

*Kód 5: Screen1View – deklarace metody pro přijmutí dat*

S přijatými daty pracuji v Screen1View.cpp a Screen2View.cpp, což jsou vytvořené obrazovky v TouchGFX. Pro pohodlné využívání grafických příkazů a pro přehlednost programu jsou v programu vytvořené bloky funkcí, které jsou volané dle příchozích dat. To znamená, že veškeré grafické operace měly svoji funkci, což značně zpříjemnilo obsluhu programu. A ať už přišla jakákoli data, program mohl datům pouze přiřadit funkci, která se provedla a pokračoval v obsluze kódu.

Vzhledem k tomu, že se data přijímají jako osmice 8 bitových hodnot s identifikačním číslem, tak se musí data ještě rozdělit. Po rozdělení dat dle identifikačního čísla se musí zmíněná hodnota v tomto konkrétním případě pojmenovaná jako promena0 zamaskovat. To znamená, že se provede bitový součet s maskou daného bitu. Pro vysvětlení uvedu příklad. Dejme tomu, že chceme zjistit hodnotu na nultém bitu, tedy na bitu s hodnotou 1. Celá osmice, pokud je každý bit 1, má hodnotu 255 dekadických. Provede se bitový součet proměnné, v které jsou uložená data s hodnotou 254 a výsledek může být buď 255 – to znamená, že nultý bit je v jedničce, nebo 254, což znamená, že nultý bit je v nule. Pro sedmý bit by maskou byla hodnota 127. Stejným způsobem je vytvořena maska s každým bitem a tím je tedy program schopen zjistit stavy, které daná proměnná nese.

```
if ((promena0 | 247) == 255)
            {
                  mainlightsON();
            }
if ((promena0 | 223) == 255) //set button
```

*Kód 6: Maskování*

Jak je v předchozích odstavcích avizováno, níže přiložená tabulka obsahuje význam dat, která se vyčítají po sběrnici. Konkrétně tedy identifikační číslo 111 hexadecimálně obsahuje stisk tlačítek na řídítkách motocyklu, světelné blinkry, světla a rychlost. Identifikační číslo 113 hexadecimálně nese hodnoty o celkových ujetých kilometrech a ujetých kilometrech po resetování uživatelem.

*Tabulka 2: Význam dat*

| **0b76543210** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|---|---|---|---|---|---|---|---|---|
| **CAN2 - Front ECU - ID 111** | | | | | | | | |
| **Byte 0** | up | down | set | return | lights | brakes | r. winker | l. winker |
| **Byte 2** | speed | speed | speed | speed | speed | speed | speed | speed |
| **CAN2 - Front ECU - ID 113** | | | | | | | | |
| **Byte 0** | trip | trip | trip | trip | trip | trip | trip | trip |
| **Byte 1** | trip | trip | trip | trip | trip | trip | trip | trip |
| **Byte 2** | trip | trip | trip | trip | trip | trip | trip | trip |
| **Byte 3** | trip | trip | trip | trip | trip | trip | trip | trip |
| **Byte 4** | ODO | ODO | ODO | ODO | ODO | ODO | ODO | ODO |
| **Byte 5** | ODO | ODO | ODO | ODO | ODO | ODO | ODO | ODO |
| **Byte 6** | ODO | ODO | ODO | ODO | ODO | ODO | ODO | ODO |
| **Byte 7** | ODO | ODO | ODO | ODO | ODO | ODO | ODO | ODO |

Samotný tachometr opět využívá proměnnou newValue. V případě navržené obdoby analogové ručičky je zde grafický prvek kruh, který dle hodnoty přidává nebo ubírá ve vyplnění. Neboli dochází ke změně úhlu.

```
Speed_speed_line.setArc(220,320+newValue);
Speed_speed_line.invalidate();
```

*Kód 7: Metoda pro zobrazení rychlosti*

Hodnota rychlosti, neboli text vložený do středu displeje, je ovládána funkcí, kterou v sobě TouchGFX potažmo CubeIDE má již vytvořenou.

```
Unicode::snprintf(SpeedBuffer, SPEED_SIZE, "%i", newValue);
Speed.setWildcard(SpeedBuffer);
Speed.invalidate();
```

*Kód 8: Metoda pro zobrazení rychlosti – text*

Daleko složitější je přepnutí denního, respektive nočního režimu. Musí při tom dojít k přenastavení téměř každého objektu. Pokud objekt, který má být ovlivněn, má vlastnost barvy, tak se využívá již připravené funkce setColor. Pomocí knihovny, která je do projektu přidána, se využívá RGB metody pro ovládání barev. Do zmíněné funkce se vkládají dekadické hodnoty v pořadí, jak může název napovídat. Písmeno R – Red, G – Green, B – Blue. Každá z těchto proměnných má rozsah 0 až 255, což udává sytost dané barvy v jejich kombinaci. Tímto způsobem je možné využít téměř jakékoli barvy, která je v projektu využita. Stejně to platí i pro příkaz setBorderColor, pouze v tomto případě se nastavuje barva okraje. Obdobným způsobem pracuje metoda změny barvy textu.

```
Speed_speed_linePainter.setColor(touchgfx::Color::getColorFrom24BitRGB(1
3, 115, 255));
```

*Kód 9: Změna barvy objektu*

Mezi další příkazy, které jsou v projektu často využívány, patří příkaz setAlpha. Tato hodnota udává průhlednost daného objektu. Pokud je nastavena na 0, objekt bude plně transparentní a nebude tedy viděn. Naopak pokud je nastavena na 255, objekt, pokud nebude překrýván, bude plně viditelný.

```
Background_light.setAlpha(255);
```

*Kód 10: Změna viditelnosti objektu*

Výše si můžeme všimnout část metody pro změnu na světlý režim obrazovky. Konkrétně tedy změna objektu image, ve kterém je uložené pozadí, na viditelné.

Funkce invalidate byla jednou z nejčastěji používaných příkazů v této práci, jelikož má za úkol zrušení platnosti dané oblasti, což má za význam překreslení dané oblasti při další operaci malování.

```
Fuel_1.setAlpha(255);
Fuel_1.invalidate();
```

*Kód 11: Funkce invalidate*

Náročnou částí kódu bylo přepnutí obrazovky, jelikož metoda, která má právě toto za úkol, byla vytvořena softwarem TouchGFX. Avšak vývojáři mají v programu chybu a ve chvíli, kdy je vytvořená další obrazovka a je nutné se na ní programově přepnout, tak metoda, respektive metody, které by přepnutí obsluhovaly, neexistují a je nutné je dopisovat v hlavičkovém souboru ručně. Toto se musí dělat pokaždé, kdy je kód vygenerován v TouchGFX kvůli jakékoli změně grafického rozhraní. Navíc hlavičkový soubor je nastaven na „read only“, to znamená, že nelze přepsat v CubeIDE a je nutné k přepsání využít Visual Studio, které dovolí tuto ochranu obejít. Toho se docílí napsáním níže uvedeného kódu do FrontedApplicationBase.cpp. Samozřejmě za předpokladu, že je v hlavičkovém souboru napsaná definice pro tuto metodu.

```
void FrontendApplicationBase::gotoScreen2ScreenNoTransition()
{
    transitionCallback =
touchgfx::Callback<FrontendApplicationBase>(this,
&FrontendApplication::gotoScreen2ScreenNoTransitionImpl);
    pendingScreenTransitionCallback = &transitionCallback;
}
void FrontendApplicationBase::gotoScreen2ScreenNoTransitionImpl()
{
    touchgfx::makeTransition<Screen2View, Screen2Presenter,
touchgfx::NoTransition, Model >(&currentScreen, &currentPresenter,
frontendHeap, &currentTransition, &model);
    }
```

*Kód 12: Deklarace metody pro přepnutí obrazovky*

Následně je možné využít tuto funkci v kódu.

```
static_cast<FrontendApplication*>(Application::getInstance())-
>gotoScreen2ScreenNoTransition();
```

*Kód 13: Přepnutí obrazovky*

Právě proto byla práce se softwarem firmy ST náročná. Není ještě plně odladěný a je plný chyb, které se pracně opravují. A vzhledem k tomu, že se neustále na grafice mění přinejmenším maličkosti, tak bylo nutné neustále kopírovat a vkládat stejné řádky kódu.

Jak bylo zmíněno v předchozích odstavcích, na všechna data a ovládání jsou vytvořené funkce, které je možné kdykoli programově zavolat, což zjednoduší práci programu. Níže se můžeme podívat na funkci rozsvícení blinkru, kde si lze všimnout změny v průhlednosti objektu a tím pádem i zobrazení plné sytosti barvy a vizuální aktivace.

```
void Screen1View::leftwinkerON()
{
	Left_winker.setAlpha(255);
	Left_winker.invalidate();

}
```

*Kód 14: Metoda aktivace blinkru*

K tématu práce s průhledností by se mělo ještě dodat následující. Navrhl jsem metodu, které si můžete všimnout níže. Standby metoda má za úkol zobrazovat režim nebo mód jako méně viditelný, pokud ho zrovna uživatel nechce měnit. Tím bylo docíleno moderního vzhledu menu. Pokud uživatel nebyl zrovna postavený na výběru denního/nočního režimu, tak tyto objekty, respektive aktuálně aktivní objekt měl nastavenou průhlednost na hodnotu 100, což na škále 0 až 255 se dá interpretovat jako napůl průhledný. Díky tomu byl uživatel vizuálně schopen pochopit, že má možnost tento objekt nastavit, ale momentálně ho nemá vybrán – viz obrázek 33.

```
void Screen2View::ModeLightStandby()
{
	regime__light.setAlpha(100);
	Light.setAlpha(100);
	regime__light.invalidate();
	Light.invalidate();
}

void Screen2View::ModeOFF()
{
	if(regime__light.getAlpha() == 255)
	{
		ModeLightStandby();
	}
}
```

*Kód 15: Metoda Standby režimu*

# 4. Závěr

V počátku této práce proběhlo seznámení s možnými způsoby řešení na téma displeje pro elektrické motocykly, rešerše již používaných zařízení a inspirace z již používaných grafik od renomovaných výrobců.

Následně se práce zaměřila převážně na design, rozpoložení a zobrazovací prvky displeje. V této části byly navrženy veškeré prvky, které jsou třeba pro ovládání motocyklu. Byl definován světlý a tmavý režim, které respektují zobrazení dle světelných podmínek a bylo navrženo zobrazování jízdních režimů motocyklu. Konečná podoba grafiky byla odsouhlasena a bylo možné pokračovat v návrhu ovládání dle přijatých dat.

Třetí krok ve vypracování konečného řešení bylo navržení komunikace, která by respektovala formát dat, která se používají v automobilovém průmyslu. Pro testování bylo užito externí periferie, která sloužila jako zdroj CAN zpráv. V tomto kroku byla vytvořená a vyladěná komunikační a ovládací část práce.

Jakmile byl projekt dokončený, přišel čas připojit desku k elektrickému motocyklu nacházejícímu se na půdě univerzity. Komunikační vodiče CANH i CANL byly připojeny na CAN budič u mikrokontroleru. Řešení bylo funkční, pomocí tlačítek na levé části řídítek motocyklu je uživatel schopen se pohybovat v menu a měnit nastavení denního či nočního režimu, měnit módy motocyklu a byly vyzkoušeny i další funkce jako například blinkry nebo dálková světla, což splnilo zadání bakalářské práce.

Pro vývoj displeje do elektrického motocyklu by deska teoreticky mohla být využitá. Její možnou nevýhodou by byla relativně menší velikost a rozlišení displeje, ale když byl kontroler připojen, dalo se objektivně říci, že by se displej dal obstojně využít. Avšak využití TouchGFX pro vývoj grafiky displeje je méně optimistický. Software není schopen pohodlně překreslovat a měnit objekty dle představ. Při vývoji a dokončování práce bylo nalezeno nemalé množství slepých uliček, které se musely nakonec vyřešit jiným způsobem. Pro jemnější ikony nebo detaily v obrazovce je téměř nepoužitelný, a to je například důvod, proč bylo častokrát nutné zvolit vektorovou grafiku a výsledek využívat jako objekt obrázek.

# Literatura


[1] Displej Nextion. *Dratek.cz* [online]. Liberec: dratek.cz, 2022 [cit. 2022-04-03]. Dostupné z: https://dratek.cz/arduino/3214-2.4-displej-nextion-usart-hmi-tft-lcd-dotykovy.html

[2] IDEC HG5G. *Rem-technik.cz* [online]. [cit. 2022-04-11]. Dostupné z: https://www.rem-technik.cz/displeje-hmi/dotykove-displeje/dotykove-displeje-hg/dotykovy-displej-idec-hg5g-v-1494.html

[3] STU SE-163. *Emerx.cz* [online]. [cit. 2022-04-11]. Dostupné z: https://www.emerx.cz/palubni-displej-2-8-lcd-obdii-full.html

[4] Univerzální displej. *AliExpress.com* [online]. [cit. 2022-04-11]. Dostupné z: https://www.aliexpress.com/item/4001254801282.html?spm=a2g0o.productlist.0.0.58dc7f2ezu7ZSC&algo_pvid=26e27718-7bf2-4834-b020-a7d4ad0b3ea9&algo_exp_id=26e27718-7bf2-4834-b020-a7d4ad0b3ea9-3&pdp_ext_f=%7B%22sku_id%22%3A%2210000015483870010%22%7D&pdp_pi=-1%3B1049.89%3B-1%3B-1%40salePrice%3BCZK%3Bsearch-mainSearch

[5] Tachometr Biker's choice. *Amazon.cz* [online]. [cit. 2022-04-11]. Dostupné z: https://www.amazon.com/85-HARLEY-FXEF-Bikers-Choice/dp/B00HXC41UI

[6] TFT displeje. *Gearpatrol.com* [online]. 1 [cit. 2022-04-11]. Dostupné z: https://www.gearpatrol.com/cars/motorcycles/a403115/tft-displays-are-the-next-motorcycle-technology-trend/

[7] Audi virtuální kokpit. *Audi.cz* [online]. 1 [cit. 2022-04-11]. Dostupné z: https://www.audi.cz/servis-a-prislusenstvi/konektivita-a-technologie/audi-inovativni-technologie/audi-virtual-cockpit

[8] Rightware Kanzi. *Rightware.com* [online]. [cit. 2022-04-11]. Dostupné z: https://rightware.com/product/kanzi-studio/

[9] LVGL knihovna. *Wiki.seedstudio.com* [online]. [cit. 2022-04-11]. Dostupné z: https://wiki.seeedstudio.com/reTerminal-build-UI-using-LVGL/

[10] QT. *Qt.io* [online]. [cit. 2022-04-11]. Dostupné z: https://www.qt.io/product/develop-software-microcontrollers-mcu

[11] STMicroelectronics. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-04-11]. Dostupné z: https://en.wikipedia.org/wiki/STMicroelectronics

[12] TouchGFX [online]. In: *avnet.com*. [cit. 2022-04-11]. Dostupné z: https://www.avnet.com/wps/portal/silica/manufacturers/m/stmicroelectronics/stmicroelectronics-touchgfx-designer/

[13] CANbus. *Autoditex.com* [online]. 1 [cit. 2022-04-11]. Dostupné z: https://autoditex.com/page/can-bus--controller-area-network-34-1.html

[14] CANbus rámec. In: *Csselectronics.com* [online]. [cit. 2022-04-11]. Dostupné z: https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial

[15] CANbus. *Canlab.cz* [online]. 1 [cit. 2022-04-11]. Dostupné z: https://www.canlab.cz/cs/can_bus

[16] STM32H747I-DISCO. In: *St.com* [online]. [cit. 2022-04-11]. Dostupné z: https://www.st.com/en/evaluation-tools/stm32h747i-disco.html

[17] FANout Board. In: *Manualslib.com* [online]. [cit. 2022-04-11]. Dostupné z: https://www.manualslib.com/manual/2032487/St-StmodPlus.html

[18] STMF769I-DISC1. In: *St.com* [online]. [cit. 2022-04-11]. Dostupné z: https://www.st.com/en/evaluation-tools/32f769idiscovery.html

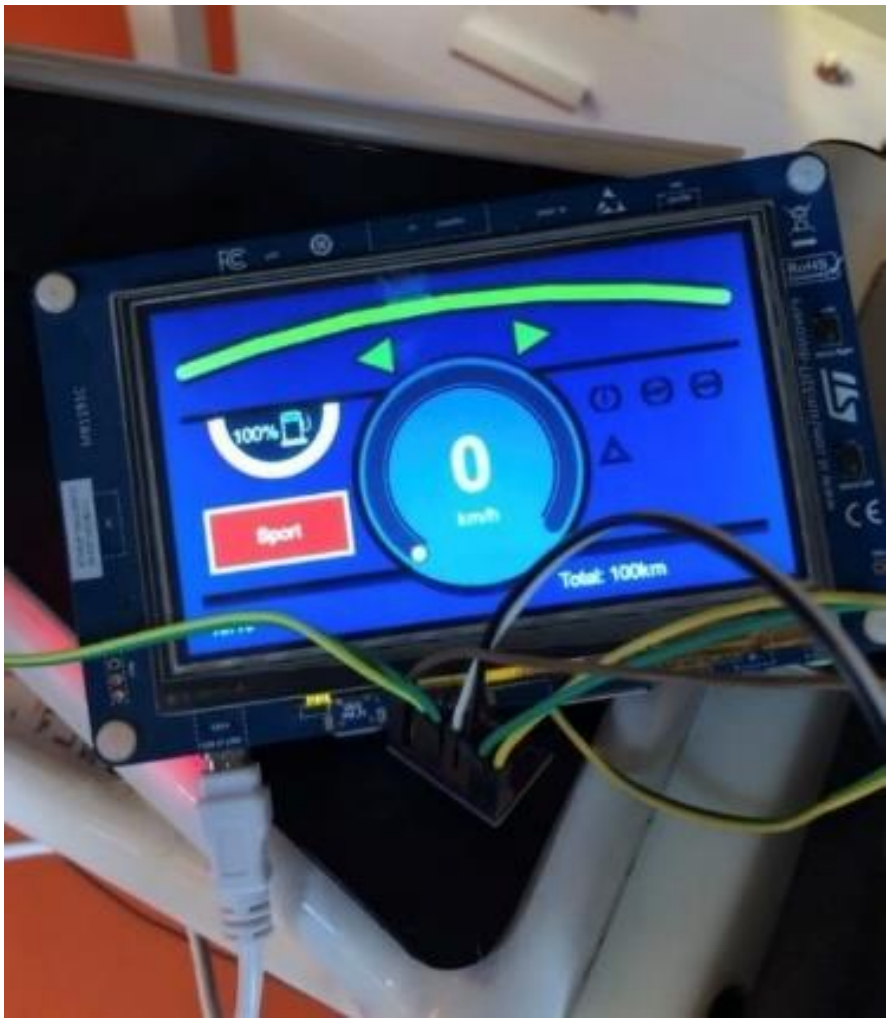
[19] STM32F746G-DISCO. In: *St.com* [online]. [cit. 2022-04-11]. Dostupné z: https://www.st.com/en/evaluation-tools/32f746gdiscovery.html

[20] PINout STM32F746. In: *Os.mbed.com* [online]. [cit. 2022-04-11]. Dostupné z: https://os.mbed.com/platforms/ST-Discovery-F746NG/

[21] CJMCU 1051. In: *Electropeak.com* [online]. [cit. 2022-04-11]. Dostupné z: https://electropeak.com/cjmcu-tja1051-can-transiever-module

# Přílohy

## A Hotové řešení



# B Hotové řešení menu