ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA APLIKOVANÉ ELEKTRONIKY A TELEKOMUNIKACÍ

DIPLOMOVÁ PRÁCE

Distribuovaná ovládací jednotka pro ozvučovací systém automobilu

Stanislav Brtna **2017**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Stanislav BRTNA**
Osobní číslo:	**E15N0065P**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Dopravní elektroinženýrství a autoelektronika**
Název tématu:	**Distribuovaná ovládací jednotka pro ozvučovací systém automobilu**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

Navrhněte koncepci ovládací jednotky pro ovládání ozvučovacího systému v automobilu. Ovládací jednotka bude využívat dotykového displeje a komunikaci pomocí rozhraní CAN.

1. Seznamte se s funkcemi ovládací jednotky a specifikujte potřebné signály hardwarového rozhraní a základní strukturu uživatelského ovládacího rozhraní s využitím dotykového displeje.
2. Na základě předchozí analýzy navrhněte blokovou strukturu zařízení a specifikujte vhodný řídicí mikrokontrolér a dotykový displej.
3. Proveďte návrh obvodového řešení zařízení. Zařízení realizujte a oživte.
4. Navrhněte a realizujte knihovnu funkcí pro zobrazení vhodných ovládacích prvků s volbou možnosti vysílání obecně definovaných zpráv pomocí rozhraní CAN. Do použitého mikrokontroléru implementujte odpovídající firmware.
5. Diskutujte možnost uživatelské konfigurace a propojení zařízení s nadřazeným systémem.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí diplomové práce: **Ing. Petr Krist, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání diplomové práce: **14. října 2016**

Termín odevzdání diplomové práce: **19. května 2017**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 14. října 2016

Abstrakt

Práce popisuje návrh a výrobu řídící jednotky ozvučovacího systému s využitím kitu STM32F769I-DISCO.

Klíčová slova

STM32, CAN

Abstract

This master's thesis describes development of control unit for car audio system. Main component of that control unit is STM32F769I-DISCO development board from STM.

Keywords

STM32, CAN

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

……

podpis

V Plzni dne 19.5. 2017 Stanislav Brtna

Seznam symbolů a zkratek

CAN - Controller Area Network, datová sběrnice

LCD - liquid crystal display, display z tekutých krystalů

SD - Secure Digital, standard v oblasti paměťových karet

px - pixel, obrazový bod

1 Úvod

Cílem této diplomové práce je vyvinout distribuovanou řídicí jednotku pro ozvučovací systém v automobilu. Autor si není přesně jist tím, co všechno má být na řídící jednotce distribuované, tak se předpokládá, že vyvíjená řídící jednotka má být spíše součástí distribuovaného ozvučovacího systému, jen takto totiž zadání dává nějaký smysl.

V dalších kapitolách této práce jsou blíže specifikovány požadavky na výsledné zařízení, navrženo blokové schéma hardware, popsán návrh schématu. Dále je rámcově popsán řídící software. V závěru jsou zhodnoceny výsledky práce.

2 Bližší specifikace

Zadání této diplomové práce specifikuje požadavky na návrh a ovládací jednotky pro distribuovaný ozvučovací systém automobilu. Tím se rozumí, že tato ovládací jednotka bude po rozhraní CAN ovládat zatím blíže nespecifikované audio zařízení. Uživatel bude s ovládací jednotkou pracovat pomocí jejího dotykového LCD displeje. Na displeji bude pro uživatelovo potěšení zobrazeno grafické rozhraní. Grafické rozhraní bude sestávat z několika obrazovek, na každou z nich bude možnost umístit ovládací prvky a přiřadit jim jistou funkcionalitu, spočívající například ve změně aktuální obrazovky nebo odeslání zprávy po rozhraní CAN. Ve zbytku této kapitoly budou upřesněny požadavky na hardware a software.

2.1 Požadované grafické prvky

Grafické rozhraní řídící jednotky by mělo umožňovat vykreslení následujících grafických prvků. Tento seznam byl vypracován společně se zadavatelem práce.

2.1.1 Textové pole

Textové pole je jeden ze základních prvků jakéhokoli grafického uživatelského rozhraní, umožňuje programátorovi snadné vykreslení textu, který může informovat uživatele.

2.1.2 Tlačítko

Tlačítko umožňuje uživateli ovlivnit chod programu a zároveň může (svým popiskem) uživatele informovat o způsobu jakým bude chod programu ovlivněn. Tlačítko umožňuje vložit uživateli pouze logickou hodnotu.

2.1.3 Posuvník

Posuvník umožňuje uživateli předat programu numerickou hodnotu v určitém, předem daném, rozsahu. V této konkrétní aplikaci je potřeba implementovat jak jednorozměrný, tak dvojrozměrný posuvník.

2.2 Požadavky na konfigurovatelnost

Dalším požadavkem je snadná konfigurovatelnost uživatelského rozhraní. Toho bude dosaženo umístěním konfigurace prvků uživatelského rozhraní na SD kartu do snadno upravitelného textového souboru. Pro většinu úprav software řídící jednotky tedy nebude potřeba nic jiného, než počítač se čtečkou SD karet a textovým editorem.

2.3 Požadavky na hardware a volba discovery kitu od STM

Základním požadavkem na hardware je použití rozumného dotykového LCD. Dále pak mikrokontrolér o dostatečném výkonu aby dokázal plynule ovládat displej a dotykovou vrstvu. Posledním hardwarovým požadavkem je implementace CAN rozhraní a možnost napájení celé řídící jednotky z palubní sítě automobilu.

2.3.1 Displej

V ideálním případě by bylo využito dotykového displeje přímo pro automobilový průmysl. Bohužel tyto displeje si výrobci autoelektroniky nechávají vyrábět na zakázku a není možné je běžně zakoupit. Další problém by byl s pořízením dokumentace k takovémuto displeji.

Na trhu dostupné dotykové displeje jsou zase příliš drahé, málo dostupné, jejich rozhraní je často špatně dokumentované. Proto bylo užito kitu STM32F769I-DISCO.

2.3.2 STM32F769I-DISCO

Tento vývojový kit od firmy STM je osazen poměrně kvalitním kapacitním dotykovým displejem, obsahuje procesor řady F7 a jedná se o špičku mezi vývojovými kity STM. Přes špičkové parametry je cena kitu poměrně nízká (2395.55 Kč na TME.cz). Cena displeje srovnatelného s displejem tohoto kitu by byla oproti ceně kitu dvojnásobná.

Další pozitivum užití tohoto kitu je možnost jeho budoucí znovupoužitelnosti pro jinou bakalářskou nebo diplomovou práci.

Užití tohoto kitu velmi zjednodušuje návrh hardware i software. Kit obsahuje většinu potřebných obvodů, je skvěle dokumentován a jsou k němu dostupné mnohé příklady ukázkových zdrojových kódů. Potřebné obvody které kit neobsahuje (zdroj a CAN budič) budou umístěny na rozšiřující modul, který bude s kitem propojen skrz rozšiřující konektor.

Obr. 1: Kit STM32F769I-DISCO

3 Návrh blokové struktury

Vzhledem k vyšší komplikovanosti užitého softwarového řešení, je části software věnována celá kapitola 6.

3.1 Hardware

Bloková struktura hardware systému je z velké míry podřízena realizaci převážné většiny hardware pomocí kitu STM32F769I-DISCO. Důvody pro volbu právě tohoto kitu jsou popsány v předchozí kapitole.

Zbytek hardware tvoří rozšiřující modul, jehož hlavním účelem bude implementace CAN budiče a napájecího zdroje.

4 Návrh obvodového řešení

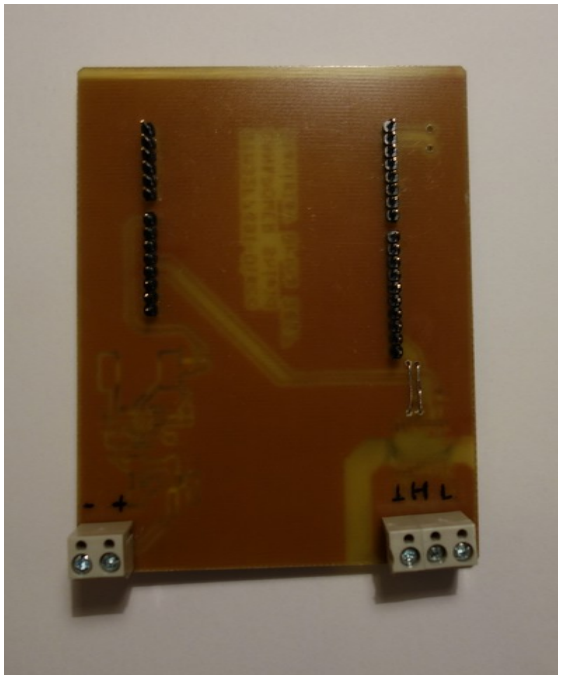
Návrh obvodového řešení se užitím kitu STM32F769I-DISCO velmi zjednodušil, dohromady není moc co navrhovat, většinu už implementuje užitý kit.

4.1 Rozšiřující modul

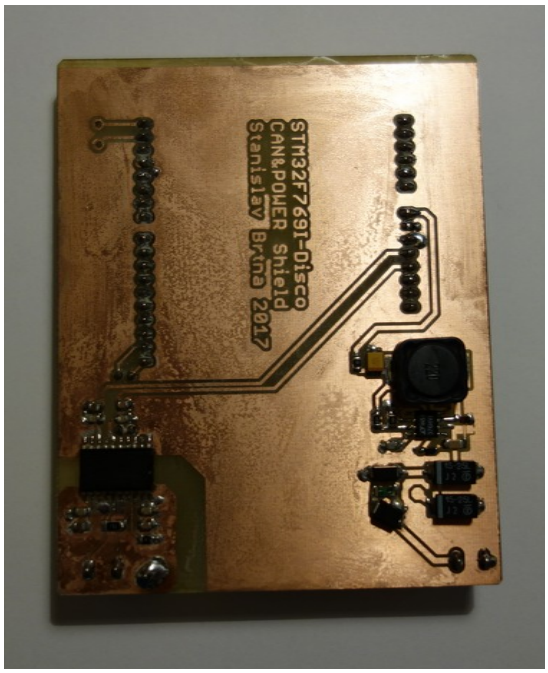
Ani tady se toho moc světoborného neděje, na rozšiřujícím modulu jsou dva integrované obvody, zapojené přesně podle katalogu. Prvním je integrovaný CAN budič s galvanickým oddělením, ADM3053. Druhým je spínaný zdroj realizovaný pomocí LT1376, s výstupním napětím 8 V. Kompletní schéma se nachází v příloze, v další kapitole věnované osazení a oživení se můžete podívat na obrázky osazené desky.

5 Osazení a oživení

DPS rozšiřovacího modulu byla vyrobena foto-cestou (filmová šablona je součástí příloh). Ručně odvrtána a osazena. Zdroj 8 V funguje dle předpokladů. CAN budič bohužel z neobjasněných důvodů nedrží na CAN_HI a CAN_LO vodičích správná napětí. To pravděpodobně způsobuje, že se zprávy sice odešlou, ale vysílač (řídící jednotka) neobdrží potvrzení o přijetí. Tento problém CANu může být způsoben i na straně software, případně na straně CAN analyzátoru, se kterým byla jednotka testována. Vzhledem k tomu, že užitý CAN analyzátor odeslaná data přijme správně, tak závažnost problému bude možno posoudit až podle toho, zda bude řídící jednotka komunikovat s dalšími zařízeními distribuovaného ozvučovacího systému.



Obr. 3: Rozšiřující modul, vrchní strana.



Obr. 2: Rozšiřující modul, spodní strana.

6 Řídící software

6.1 Grafická knihovna

Nejnižší vrstvu grafické knihovny tvoří moduly *lcd_basics* a *lcd_io*. První modul, *lcd_io* řeší napojení grafické knihovny na ovladač zobrazovací jednotky. Druhý modul, *lcd_basics* umožňuje vykreslení elementárních grafických prvků, jako například: bodů, čar, obdélníků, kruhů a kružnic. Dále umožňuje vykreslit textové řetězce předpřipraveným fontem s variabilní šířkou znaku. Poslední a asi nejdůležitější vlastností je možnost omezení vykreslovací plochy pomocí funkcí *LCD_setDrawArea* a *LCD_setSubDrawArea*. Při omezené vykreslovací oblasti se vykreslí pouze ty části grafických prvků, spadající do této

oblasti. *LCD_setDrawArea* pevně omezí oblast vykreslování. *LCD_setSubDrawArea* umožňuje omezit vykreslovací oblast pouze uvnitř už omezení vykreslovací oblasti.

Na základních funkcích této grafické knihovny stojí samotná knihovna grafického rozhraní, rozdělená do souborů *pscg.h*, *pscg.c*, *pscg_draw.c* a *pscg_elements.c*. Toto rozdělení je čistě z důvodů přehlednosti *pscg_elements.c* obshahuje konstruktory elementů grafického rozhraní, *pscg_draw.c* obsahuje vykreslovací funkce příslušných elementů a v *pscg.c* je zbytek funkčního kódu a jsou zde definovány globální proměnné.

Prvky grafického rozhraní jsou v paměti uloženy jako pole struktur. Je tedy omezen maximální počet zobrazitelných prvků. Ve struktuře prvku je uloženo o jaký typ prvku se jedná a jsou zde i další potřebné parametry, viz dále.

Prvky grafického rozhraní

- obrazovka
- tlačítko
- posuvník
- textové pole

Hlavní parametry prvků grafického rozhraní

- pozice prvku
- obrazovka na které se prvek nachází
- hodnota
- poslední událost prvku
- validnost prvku

Základním prvkem je prvek obrazovky (screen). Na obrazovku můžeme vkládat další prvky grafického rozhraní, obrazovku je možné vykreslit, obrazovce je možné předávat události, jako například kliknutí myší nebo pozici dotyku na dotykovém panelu.

Každý z prvků grafického rozhraní má své parametry, jako například na jaké obrazovce se nachází a nebo pozici prvku vzhledem k počátku dané obrazovky. Prvek má svou hodnotu, význam hodnoty se liší v závislosti na konkrétním prvku. Prvek si dále pamatuje poslední jemu předanou událost. Posledním z důležitých parametrů prvku je jeho validita, při vytvoření prvku se nejprve prochází pole prvků, po nalezení prvního ne-validního prvku se tento prvek naplní novými daty a označí za validní. V případě potřeby smazat prvek se prvek jednoduše označí za ne-validní.

Od těchto vnitřních procesů knihovny je však programátor účinně oddělen. Vytváření prvků se provádí voláním konstruktoru, konstruktor vrátí identifikátor (*id*) nově vytvořeného prvku. Pomocí tohoto *id* je poté možné měnit nebo číst parametry konkrétního prvku.

Ukázka kódu: (v jazyku C)

```
//inicializace proměnných do kterých budou uloženy identifikátory
uint16_t screen;
uint16_t button;
uint16_t text;
//volání konstruktoru vrátí identifikátor
screen = pscg_add_screen();
//s identifikátorem poté můžeme dále pracovat
text = pscg_add_text(2, 0, 22, 3, "hello world", screen);
button = pscg_add_button(2, 3, 16, 6, "button", screen);
//vykreslení
LCD_setDrawArea(0, 0, LCD_W, LCD_H);
pscg_draw_screen(0, 0, 800, 479, screen, 1);
```

Obr. 4: Takto se zobrazí předchozí kód.

Předání události obrazovce je velmi podobné jejímu vykreslení. Poté můžeme zkusit přečíst poslední událost tlačítka a případně událost obsloužit.

```
//předání souřadnic dotyku a příslušné události
pscg_touch_input(0, 0,800,479,touchX,touchY,ev,screen);
if (pscg_get_event(button)==EV_RELEASED){
  //obslužný kód
  printf("button pressed\n");
  //vymazání události
  pscg_set_event(button,EV_NONE);
}
```

Jak jste si už mohli povšimnout, obrazovka je také jen grafický prvek a díky tomu je možno na jednu obrazovku přidat další pod-obrazovky.

Autor nepředpokládá, že by někoho doopravdy zajímalo, jak to funguje. Proto bude popsáno pouze rozhraní knihovny.

```
uint16_t pscg_add_screen();
```

Přidá novou obrazovku, vrátí její id.

```
void pscg_destroy_screen(uint16_t id);
```

Odstraní obrazovku a všechny prvky na ní obsažené

Obrazovky mají speciální parametry xScroll, yScroll, x_cell a y_cell. Scroll parametr umožňuje posunout obsah uvnitř vykreslované obrazovky, cell parametry mění rozlišení mřížky, do které se zarovnávají prvky v obrazovce. Oba parametry jsou v px. Níže popsanými funkcemi je možno tyto parametry číst nebo měnit.

```
uint16_t pscg_get_xscroll(uint16_t id);
uint16_t pscg_get_yscroll(uint16_t id);
void pscg_set_xscroll(uint16_t id, uint16_t val);
void pscg_set_yscroll(uint16_t id, uint16_t val);
void pscg_set_x_cell(uint16_t id, uint16_t val);
void pscg_set_y_cell(uint16_t id, uint16_t val);
uint16_t pscg_get_x_cell(uint16_t id);
uint16_t pscg_get_y_cell(uint16_t id);
```

Nové prvky je možno vytvářet konstruktory těchto prvků. Některé parametry těchto konstruktorů jsou obecné, společné pro všechny konstruktory. Návratovou hodnotou konstruktoru je id nově vytvořeného prvku.

x1 – x souřadnice levého horního rohu vykreslovaného prvku v mřížce dané hodnotou x_cell obrazovky
y1 – y souřadnice levého horního rohu vykreslovaného prvku v mřížce dané hodnotou x_cell obrazovky
x2 – x souřadnice pravého dolního rohu vykreslovaného prvku v mřížce dané hodnotou x_cell obrazovky
y2 – y souřadnice pravého dolního rohu vykreslovaného prvku v mřížce dané hodnotou y_cell obrazovky
screen – obrazovka na kterou má být nový prvek přidán

```
uint16_t pscg_add_button(uint16_t x1, uint16_t y1, uint16_t
  x2, uint16_t y2, uint8_t *str, uint16_t screen);
```

str – popisek tlačítka

```
uint16_t pscg_add_slider_v(uint16_t x1, uint16_t y1, uint16_t
  x2, uint16_t y2, uint16_t param, uint16_t value, uint16_t
  screen);
```

param – maxinální hodnota posuvníku
value – aktuální hodnota posuvníku

```
uint16_t pscg_add_text(uint16_t x1, uint16_t y1, uint16_t x2,
  uint16_t y2, uint8_t *str, uint16_t screen);
```

str – obsah textového pole

```
uint16_t pscg_add_progbar_v(uint16_t x1, uint16_t y1, uint16_t
  x2, uint16_t y2, uint16_t param, uint16_t value, uint16_t
  screen);
```

param – maxinální hodnota posuvníku
value – aktuální hodnota posuvníku

Všeobecné funkce pro práci s prvky:

```
void pscg_set_screen(uint16_t id, uint16_t val);
```

Nastaví prvku s *id* `id` obrazovku s *id* `val`.

Následující funkce asi nepotřebují příliš dalšího vysvětlování.

```
uint16_t pscg_get_value(uint16_t id);
void pscg_set_value(uint16_t id, uint16_t val);

uint16_t pscg_get_param(uint16_t id);
void pscg_set_param(uint16_t id, uint16_t val);

uint16_t pscg_get_x1(uint16_t id);
uint16_t pscg_get_x2(uint16_t id);
uint16_t pscg_get_y1(uint16_t id);
uint16_t pscg_get_y2(uint16_t id);

void pscg_set_x1(uint16_t id, uint16_t val);
void pscg_set_x2(uint16_t id, uint16_t val);
void pscg_set_y1(uint16_t id, uint16_t val);
void pscg_set_y2(uint16_t id, uint16_t val);

void pscg_set_str(uint16_t id, uint8_t *str);

void  pscg_set_x1y1x2y2(uint16_t  id,  uint16_t  x1,uint16_t  y1,
  uint16_t x2,uint16_t y2);
```

```
uint16_t pscg_get_event(uint16_t id);
```

Vrací poslední událost prvku.

```
void pscg_set_event(uint16_t id, gr2EventType val);
```

Umožňuje nastavit event prvku, využitelné hlavně pro mazání už zpracované události.

```
void pscg_draw_screen(uint16_t x1, uint16_t y1,uint16_t x2,
  uint16_t y2, uint16_t screen, uint8_t all );
```

Vykreslí screen daného id.
x1 – x souřadnice levého horního rohu vykreslované obrazovky v px
y1 – y souřadnice levého horního rohu vykreslované obrazovky v px
x2 – x souřadnice pravého dolního rohu vykreslované obrazovky v px
y2 – y souřadnice pravého dolního rohu vykreslované obrazovky v px
screen – id screenu k vykreslení
all – 1 pokud se mají překreslit všechny prvky, 0 pokud se mají překreslit jen modifikované prvky

```
uint8_t pscg_touch_input(uint16_t x1, uint16_t y1,uint16_t x2,
  uint16_t y2,uint16_t touch_x,uint16_t touch_y,gr2EventType
  event, uint16_t screen );
```

Funkce pro předání souřadnic dotyku a typu události do vykreslené obrazovky.
touch_x – x souřadnice dotyku, v px
touch_y – y souřadnice dotyku, v px
event - událost typu gr2EventType

pozn: gr2EventType je enumerační typ sestávající z hodnot:

- EV_NONE – žádná událost
- EV_PRESSED – právě stisknuto
- EV_HOLD – právě drženo
- EV_LONGHOLD – delší dobu drženo
- EV_RELEASED – právě puštěno

Napojení grafické knihovny na konkrétní zobrazovací jednotku

Modul *lcd_io* pro vykreslení bodu volá funkci:

```
void ExtDrawPoint(int x, int y, uint16_t color);
```

Tuto funkci už je možno implementovat podle konkrétního zobrazovacího rozhraní, nezávisle na zbytku knihovny. Autor ji implementoval jak pro knihovnu SDL2 aby bylo možno použít grafickou knihovnu GR2 na osobních počítačích, tak pro vývojový kit STM32F769I-DISCO. U vývojového kitu je použito ještě několika dalších optimalizací, jako napojení některých funkcí z *lcd_basics* přímo na vykreslovací funkce dodávané od STM, to za účelem zvýšení rychlosti vykreslování.

6.2 Konfigurační soubor

Zadání práce specifikuje pouze tvorbu knihovny grafických funkcí, nechává však volnost v konkrétní implementaci. Dodávaná knihovna v jazyce C by byla pravděpodobně postačující a požadavek na snadnou konfigurovatelnost rozhraní snadno naplnitelný jednoduše strojově zpracovatelným textovým souborem, přesto se autor rozhodl pro použití svého skriptovacího jazyka SVS. SVS je ve vývoji od září roku 2016, vývoj je převážně výukového charakteru. Oblast nasazení SVS je segment mikrokontrolérů s hodnotami paměti RAM 20 kB a více. Dále je kladen důraz na jednoduchost použití, škálovatelnost a jednoduchost napojení běhového prostředí na zbytek aplikace v jazyce C.

Jednotlivé funkce grafické knihovny a funkce STM32 HAL knihovny pro práci s CAN sběrnicí jsou obaleny shodnými funkcemi skriptu SVS. Řídicí jednotka po startu a inicializaci

potřebného hardware načte z SD karty konfigurační soubor, který zavede do běhového prostředí SVS, v tom se poté konfigurační soubor vykonává. Výhoda tohoto řešení je v rozdělení funkcionality mezi „natvrdo“ naprogramovaný nativní firmware řídící jednotky a skript na kartě. Idea je taková, mít tu část funkcionality, o které předpokládáme, že bude častěji modifikována implementovanou ve skriptu, protože pro modifikaci této části funkcionality bude v ideálním případě postačovat počítač s poznámkovým blokem a čtečkou karet. Na druhou stranu skript umožní poměrně rozsáhlé možnosti konfigurace funkcionality. Takto minimalizujeme nutnost instalace jakéhokoli vývojového prostředí pro daný mikrokontrolér, které by bylo potřeba, pokud bychom chtěli často modifikovat funkcionalitu nativního kódu.

6.2.1 Detailní popis konfigurace

Konfigurační soubor je rozdělen na dvě sekce, sekci *init*, jež se vykoná při prvním spuštění jednotky a sekci *update*, jež se vykoná periodicky, vyvolá li uživatel nějakou událost.

Příklad konfiguračního souboru: (jazyk SVS)

```
function init {
  #komentář
  screen = sys pAddScreen();
  sys pSetYcell(screen,42);
  button = sys pAddButton( 2, 1, 16, 3, "Button", screen);

  sys pSetMainScr(screen);
  sys pSetScrName("Hello world");

  sys canSetId(600 , 0 );
  sys canSetRTR(0);
}

function update {
  if (sys pGetEvent(button)==3){
    sys canSend(3 ,1,2,3,0,0,0,0,0);
    sys pSetEvent(button,0);
  }
}
```

Jak je patrné z předchozí ukázky, užití skriptu je podobné předchozímu příkladu použití grafické knihovny pomocí jazyka C. Obalové funkce jsou uvozeny klíčovým slovem ***sys***, komentáře jsou uvozeny znakem ***#***. Sekce *init* a *update* jsou uvozeny klíčovým slovem function. V sekci update můžeme vidět obsluhu tlačítka, které po stisknutí odešle zprávu po sběrnici CAN. Zpráva bude mít délku 3 byty a hodnoty bajtů budou 1, 2 a 3.

Jednoduchou modifikací obslužného kódu můžeme například implementovat čítač stisknutí tlačítka a počet stisknutí odeslat po sběrnici CAN. A to bez rekompilace firmware.

```
if (sys pGetEvent(button)==3){
  sys canSend(1 ,counter,0,0,0,0,0,0,0);
  counter=counter+1;
  sys pSetEvent(button,0);
}
```

Obr. 5: Takto se zobrazí předchozí ukázka kódu

Mezi grafickou knihovnou GR2 a interpretem konfiguračního souboru je navíc mezivrstva, jež řeší názvy obrazovek a tlačítko zpět, umístěné v pravém horním rohu. Tato mezivrstva umožňuje snadné přecházení mezi pod-obrazovkami. Další část kódu řeší vstup událostí z dotykového displeje.

Pokud bychom chtěli modifikovat obsluhu tlačítka tak, aby po jeho stisknutí přešel uživatel do podobrazovky id *subscr.* (viz příklad 2 na CD)

```
if (sys pGetEvent(button)==3){
      sys pSetMainScr(subscr);
      sys pSetScrName("Subscreen");
      sys pSetEvent(button,0);
    }
```

Jak obsluha prvků všech obrazovek se musí nacházet v sekci update, inicializace nových obrazovek, případně nových grafických prvků se může nacházet v jakékoli sekci.

6.2.2 Přehled SVS

Tab. 1 Tabulka klíčových slov

Klíčové slovo	Význam
function	Uvozuje funkci.
if	Podmíněný příkaz.
else	Druhá větev podmíněného příkazu.
while	Smyčka typu while.
break	Přerušení vykonávané smyčky.
return	Opuštění právě vykonávané funkce.
end	Ukončí načítání souboru.
sys	Uvozuje volání systémové funkce.

Systémové funkce

```
sys pAddScreen();
```

Přidá novou obrazovku, vrací id.

```
sys pSetMainScr(id);
```

Nastaví hlavní obrazovku na obrazovku s daným id.

```
pSetScrName(name);
```

Nastaví popisek hlavní obrazovky.

Následující funkce jsou velmi podobné funkcím popsaným v kapitole věnované grafické knihovně.

```
pAddButton(x1,y1,x2,y2, str, scrId );
pAddSlider(x1,y1,x2,y2,z_kolika, kolik,scrId);
pAddVBar(x1,y1,x2,y2,z_kolika, kolik,scrId);
pAddText(x1,y1,x2,y2,str,scrId);

pGetValue(Id);
pSetValue(Id, val);

pGetEvent(Id);
pSetEvent(Id, val);
```

Poznámka: Hodnota eventu předávaná funkcemi pGetEvent a pSetEvent nabývá hodnot:

0 - EV_NONE
1 - EV_PRESSED
2 - EV_HOLD
3 - EV_RELEASED

```
pSetScreen(Id, scr);
pSetStr(Id,str);
pSetXYXY(Id,x1,y1,x2,y2);
pGetXscroll(Id);
pGetXcell(Id);
pGetYcell(Id);
pSetXcell(Id,val);
pSetYcell(Id,val);
pGetYscroll(Id);
pSetXscroll(Id,val);
pSetYscroll(Id,val);
```

Následující funkce jsou určeny pro práci s rozhraním CAN.

```
canSetId(id, ext_id);
```

Nastaví ID odesílaných CAN zpráv.

```
canSetRTR(rtr);
```

Pokud rtr bude 1, nastaví odesílaným zprávám příznak RTR.

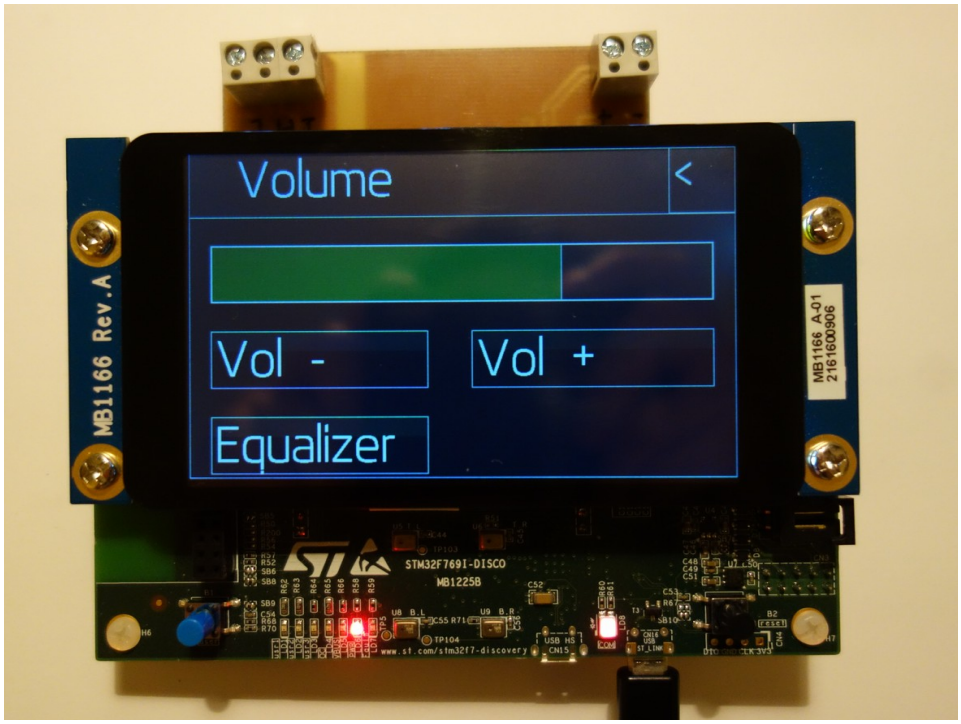
```
canSend(len, d0, d1, d2, d3, d4, d5, d6, d7);
```

Odešle danou zprávu po sběrnici CAN.

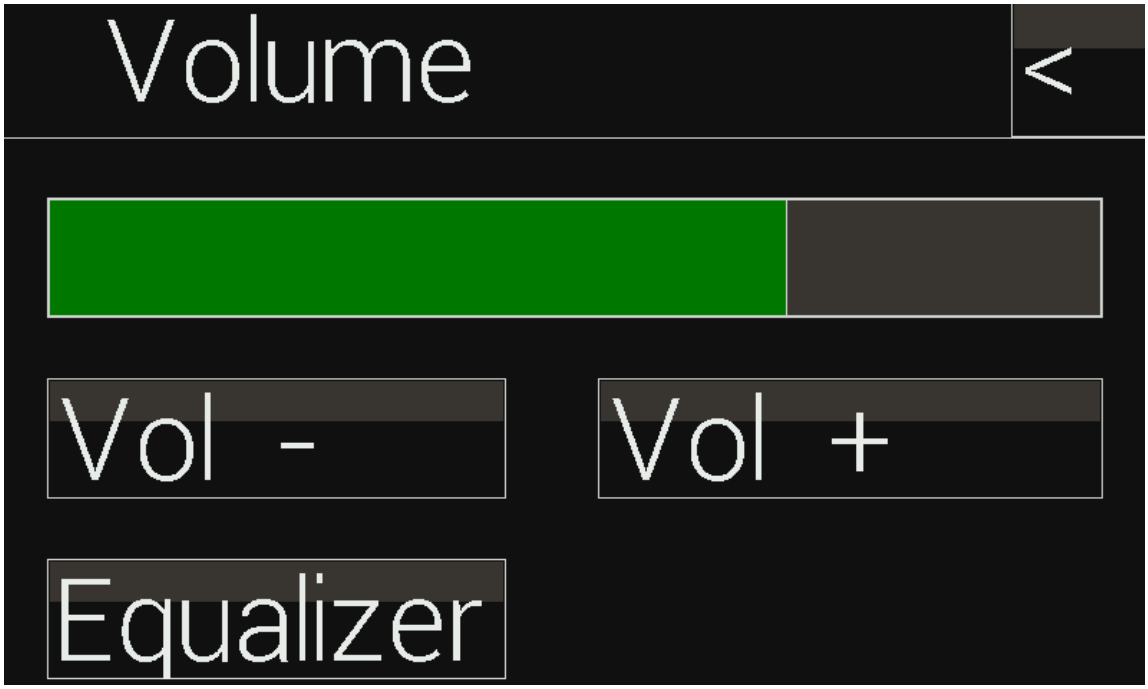
len - počet bytů zprávy (max 8)
d0 … d7 - datové byty odeslané zprávy

6.2.3 Simulátor pro PC

Čtenářovu pozornému oku jistě neušlo, že snímky obrazovky na uvedených příkladech nejsou fotografiemi displeje řídící jednotky. Jedná se o snímky obrazovky ze simulátoru, kde programové vybavení řídící jednotky neběží nad knihovnami mikrokontroléru, ale nad knihovnou SDL2 na PC. Tento simulátor umožňuje otestovat konfiguraci řídící jednotky v pohodlí PC, bez nutnosti nahrávání konfigurace do zařízení. Kompatibilita není stoprocentní, ale je dostatečná pro průběžné testování aplikace.



Obr. 6: Ukázka běhu zařízení



Obr. 7: Ukázka té samé obrazovky v simulátoru.

7 Závěr

Využití vývojového kitu se vyplatilo zrychlením vývoje hardware i software. Bohužel aktuální stav zařízení není dokonalý. Je zde problém s odesíláním zpráv po sběrnici CAN, který se sice podařilo ošetřit tak, že se při použití s užitým CAN analyzátorem neprojevuje, avšak se může snadno projevit až bude řídící jednotka finálně nasazena. Grafická knihovna GR2 neobsahuje kompletně všechny požadované grafické prvky. Konfigurační rozhraní skriptu je sice velmi flexibilní a umožňuje rozsáhlé změny funkcionality bez požadavků na specializovaný software, ale má své problémy, jako jsou například horší možnosti ladění, problematické chybové výpisy a nedostatek dokumentace. Není neobvyklá situace, při které dojde výskytu těžko odhalitelné chyby v některé části programu. Proto je v aktuálním stavu zařízení jen těžce použitelné pro cílovou aplikaci.

8 Seznam literatury a informačních zdrojů

[1] *GRIES, David. Compiler Construction for Digital Computers.* John Wiley and Sons, New York, 1971. ISBN-13: 978-0471327769.

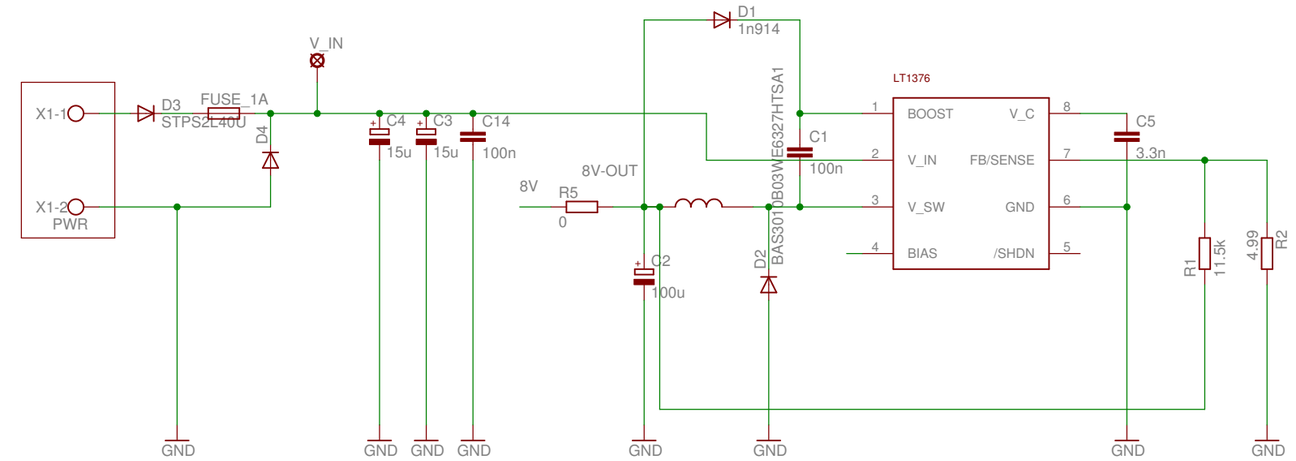
Přílohy

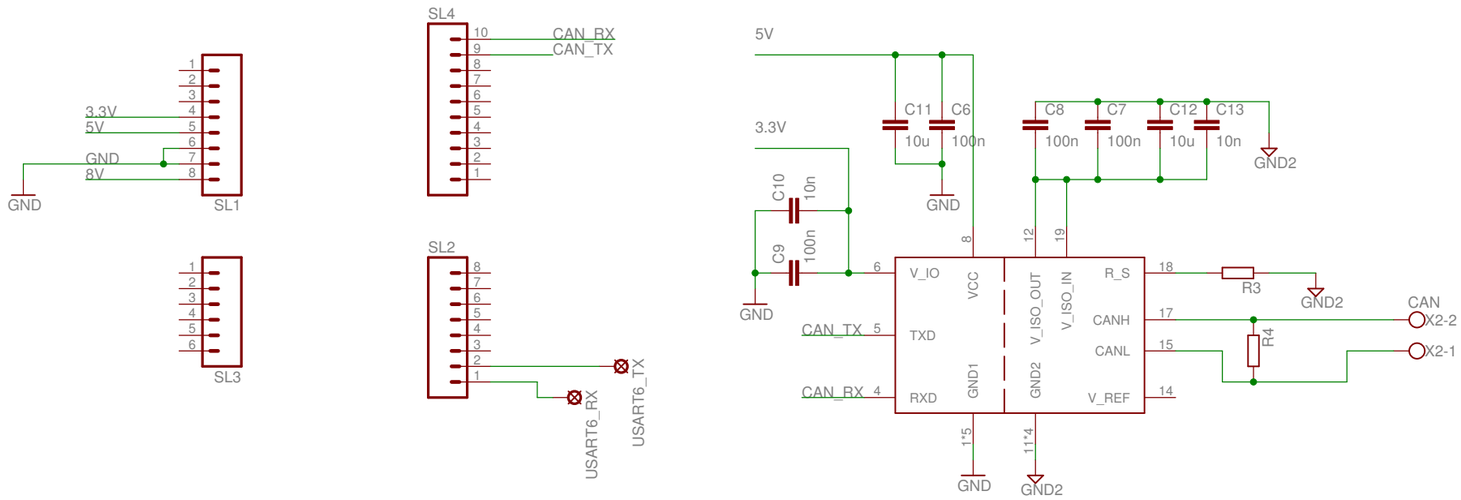
Obsah přiloženého CD:

- Zdrojové kódy
  - zdrojový kód aplikace
  - zdrojový kód simulátoru pro PC
  - příklady použití konfiguračního souboru
- Podklady k výrobě DPS, ve formátu EAGLE
- Tato práce ve formátu PDF
- Katalogové listy použitých součástek ve formátu PDF

Příloha 1:
Filmová předloha použitá k osvitu DPS.

Příloha 2:
Kompletní schéma zapojení





STM32F769I-DISCO CAN & POWER Expansion

Stanislav Brtna 2017