Ovladač s LCD dotykovým displejem

Marek Makový

Bakalářská práce
2015

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Marek Makový**
Osobní číslo:	**A13883**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Ovladač s LCD dotykovým displejem**
Téma anglicky:	**A Controller with an LCD Touch Screen**

Zásady pro vypracování:

1. **Prostudujte architekturu procesoru ARM Cortex M.**
2. **Analyzujte použité řadiče a SW ovladače LCD displeje.**
3. **Implementujte snímání souřadnic dotyků pomocí AD převodníku procesoru.**
4. **Navrhněte grafické uživatelské prostředí ovladače.**
5. **Implementujte komunikační protokol ovladače.**

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **YIU, J. The Definitive Guide to the ARM Cortex-M3. Elesevier, 2007. ISBN 978-0-7506-8534-4.**
2. **SLOSS, A., D. SYMES a Ch. WRIGHT. ARM System Developers Guide. Elesevier, 2004. ISBN 1-55860-874-5.**
3. **VALVANO, J. W. Embedded Systems: Real-Time Interfacing to Arm Cortex(TM)-M Microcontrollers. CreateSpace Independent Publishing Platform, 2011. ISBN 978-1463590154.**
4. **VALVANO, J. W. Embedded Systems: Introduction to ARM Cortex-M Microcontrollers. CreateSpace Independent Publishing Platform, 2013. ISBN 978-1477508992.**
5. **VALVANO, J. W. Embedded Systems: Real-Time Operating Systems for the ARM Cortex-M Microcontrollers. CreateSpace Independent Publishing Platform, 2012. ISBN 978-1466468863.**
6. **SEAL, D. ARM Architecture Reference Manual. Addison-Wesley, 2001. ISBN 978-0201737196.**
7. **SLOSS, A., D. SYMES a C. WRIGHT. ARM System Developers Guide. Morgan Kaufmann, 2004. ISBN 978-1558608740.**

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **6. března 2015**

Termín odevzdání bakalářské práce: **22. května 2015**

Ve Zlíně dne 6. března 2015

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

16.5.2015

.....
podpis diplomanta

ABSTRAKT

Cílem této práce je vytvořit systém s grafickým dotykovým LCD displejem a procesorem ARM Cortex M4. Ovladač má být zabudován do masážní vany a bude sloužit pro uživatelsky přívětivé řízení agregátů masážní vany, tj. vzduchových kompresorů, vodního čerpadla, servo-ventilů a RGB světla. Nejdříve jsem prostudoval použité HW řadiče a implementoval ovladač LCD displeje a jeho dotykové vrstvy. Následoval návrh grafického uživatelského prostředí a jeho softwarová realizace. Dále jsem se podílel na návrhu protokolu pro komunikaci se silovou jednotkou a následně jej implementoval do ovladače.

Klíčová slova: LCD ovladač, ARM Cortex-M4, masážní vana, STM32, STM32F4, HX8352, dotykový displej

ABSTRACT

Main goal of this thesis is to realize a control system with LCD display and ARM Cortex-M4 MCU. Controller is supposed to be embedded into massage bathtub and his main purpose is user friendly controlling of appliance – air compressor, water pump, servo valves and RGB LED light. I started by getting familiar with ARM Cortex-M4 core. Then I implemented a driver for LCD display and his touch screen. My next goal was to design and implement a graphical user interface. I was also participating on designing a protocol for communication with power unit and then I implemented the protocol on the controller side.

Keywords: LCD controller, ARM Cortex-M4, massage bathtub, STM32, STM32F4, HX8352, touchscreen

Poděkování patří Ing. Dulíkovi Ph.D. a týmu doktorandů z laboratoře D206. Byli vždy ochotní poradit.

OBSAH

ÚVOD

Tato práce vznikla v rámci programu Inovační vouchery Zlínského kraje z roku 2013, který řešila Fakulta aplikované informatiky Univerzity Tomáše Bati ve Zlíně ve spolupráci s firmou Santech v roce 2013-2014. Firma Santech je výrobcem masážních van a chce rozšířit své portfolio o vany ovládané pomocí dotykového LCD displeje. Zákazníkova představa je LCD dotykový displej v rámečku umístěný do stěny masážní vany, na kterém bude možno zobrazit především barevné ikony, v omezeném rozsahu i alfanumerické symboly. Pomocí těchto prvků bude možné ovládat agregáty, které jsou ve vaně nainstalovány a sledovat měřené údaje, např. teplotu vody . Z důvodu použití ve vaně byla zvolena rezistivní technologie dotykové vrstvy. Důležitým kritériem byla cena, proto byl použit 3,2 palcový displej s rozlišením 240x400 pixelů. Ten je dodáván s rezistivní dotykovou vrstvou, která je vyvedena na konektor displeje přímo, bez řadiče. Bylo proto potřeba ji připojit na A/D převodník procesoru a napsat pro ni ovladač. Ovladač vany neřídí agregáty přímo, ty jsou řízeny silovou jednotkou, která je s ovladačem propojena sběrnicí RS422 a komunikují mezi sebou protokolem, jehož návrh a implementace byla součástí práce. Vana disponuje funkcí automatického čištění, která, pokud je aktivována, spustí samočistící proceduru po vypuštění vany. Masážní vana disponuje také automatickými programy, které podle předem definovaného rozpisu zapínají a vypínají konkrétní agregáty ve stanoveném čase.

Implementace ovladače proběhla ve 3 fázích:

- konstrukce elektroniky ovladače, tj. desky plošných spojů s CPU typu ARM a s dotykovým LCD displejem,
- návrh grafického uživatelského rozhraní,
- implementace firmware ovladače, tj. aplikace s grafickým uživatelským rozhraním, která uživateli masážní vany umožní ovládat všechny její funkce.

Realizace celého projektu obnášela spolupráci několika týmů:

- plošný spoj elektroniky LCD ovladače vytvořili Ing. Peter Janků, Tomáš Harik a Petr Vítek,
- návrhy grafického uživatelského rozhraní LCD ovladače zpracovali Bc. Václav Peredarjuk v rámci své diplomové práce a zlínská firma eDesign

- firmware implementoval Marek Makový s využitím knihoven STMicroelectronics, implementace RLE Ing. Tomáše Dulíka, knihoven Bc. Tomáše Juřeny pro zápis do flash paměti a pro inicializaci boot loaderu,
- silová elektronika byla implementována týmem FEKT VUT v Brně v rámci projektu Inovačního voucheru Jihomoravského kraje.

Mým úkolem bylo napsat ovladač pro dotykovou vrstvu a LCD displej, vytvořit a implementovat protokol pro komunikaci po RS422, vytvořit knihovnu pro uložení a zobrazení grafických prvků, jejich interakci v závislosti na doteku a toto vše spojit do výsledné aplikace ovladače vany.

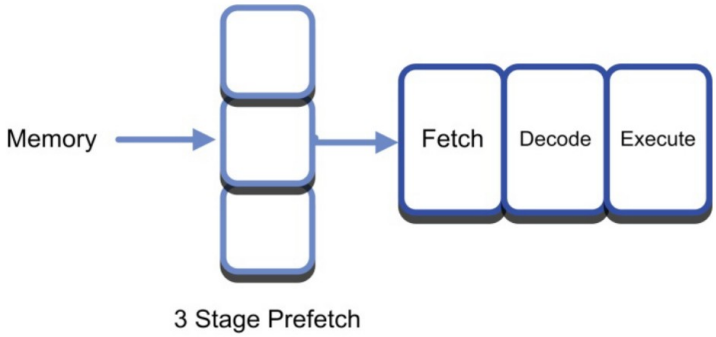
I. TEORETICKÁ ČÁST

1 PROCESOR ARM CORTEX M

Procesory architektury ARM Cortex-M jsou cíleny pro aplikace citlivé na cenu a spotřebu energie. Jako typické použití uvádí výrobce řízení motorů, zařízení spadající do oblasti Internet of Things, chytré senzory a rozhraní člověk-stroj. Procesory této řady implementují instrukční sadu Thumb-2, která je směsicí 16bitových a 32bitových instrukcí. Cortex-M implementuje architekturu ARMv7, která vychází z harvardské architektury. [1]

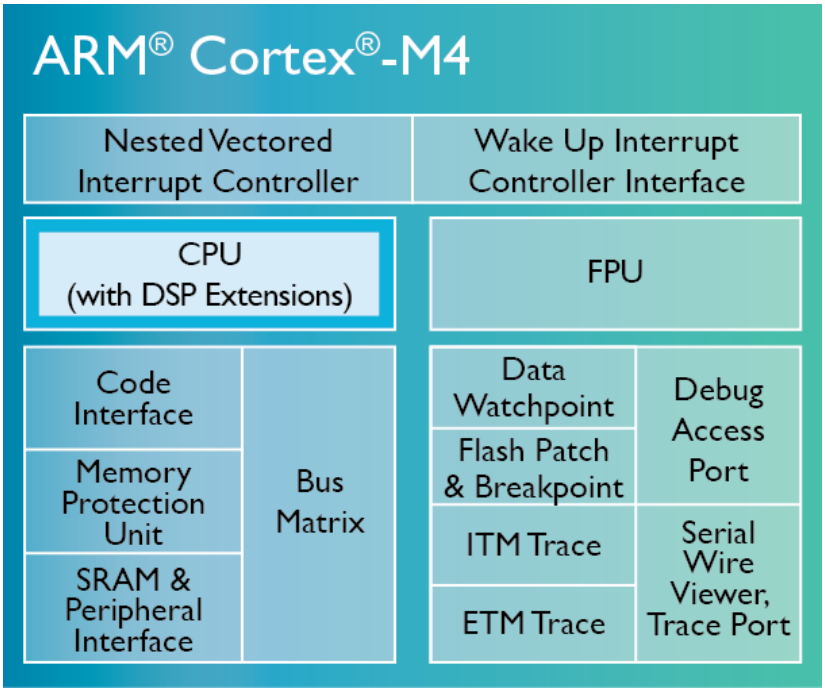
1.1 Architektura Cortex-M4

Architektura Cortex-M4 dokáže provést většinu instrukcí v jednom cyklu. Její pipeline má 3 fáze. Disponuje FPU s jednoduchou přesností, až 240 fyzickými přerušeními a až 256 úrovněmi přerušení. [8]



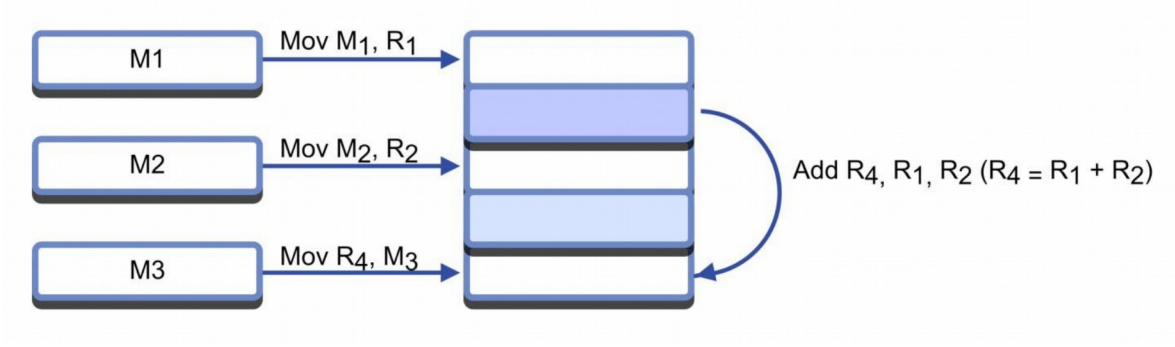
Obr. 1. Pipeline architektury Cortex-M4 [8]

Zatímco jedna instrukce je prováděna, druhá je dekódována a třetí je načítána. Cortex M-4 disponuje také branch prediction, která při zjištění instrukce větvení provádí speculative fetch, díky kterému jsou zpracovávány obě možnosti větvení.[1]



Obr. 2. Blokové schéma architektury ARM Cortex-M4 [10]

Cortex-M4 má load & store architekturu, data proto musí být načtena do centrálního registru než s nimi může být pracováno.[8]



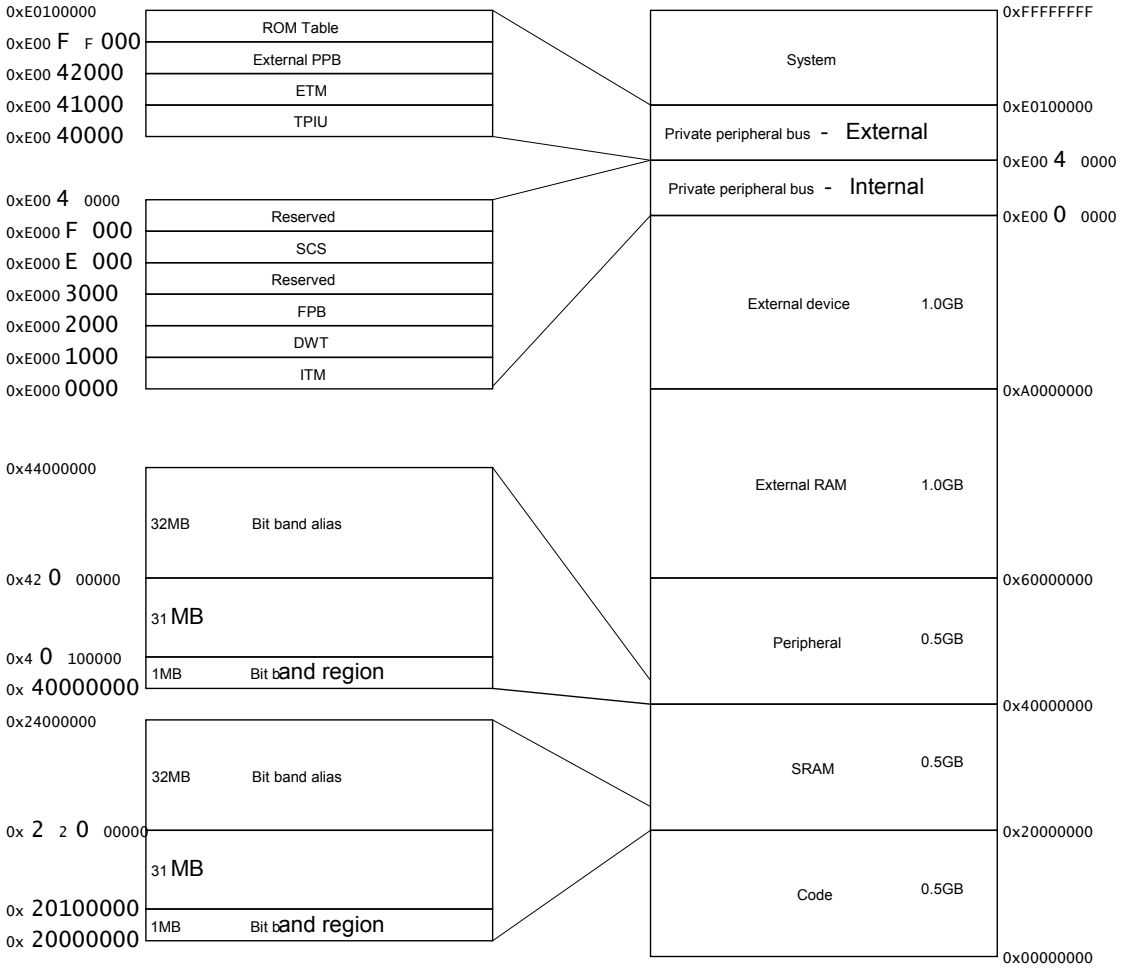
Obr. 3. Princip fungování load & store architektury [8]

Register file procesoru se skládá z 16 32bitových registrů. R0 až R12 jsou registry obecného použití, R13-R15 jsou speciální registry. Registr R13 je používán jako stack pointer a je zdvojen, což umožňuje procesoru provozovat 2 operační módy současně, využití typicky při běhu RTOS. Registr R14 je tzv. link register, ukládá se do něj návratová adresa při volání funkce. R15 slouží jako program counter. [14]

Obr. 4. Soubor registrů procesoru Cortex-M4 [8]

1.2 Mapa paměti

Architektura Cortex-M4 má definovanou strukturu paměti jak je zobrazeno na obrázku 5. Adresovatelný prostor má velikost 4 GB. Program může být umístěn do oblastí Code, SRAM nebo External RAM. Region Code se nachází na prvním 0,5 GB adresovatelného prostoru. Za ním následuje 0,5 GB SRAM blok, ve kterém je vyhrazen prostor o velikosti 32 MB pro bit-band oblast. Ta je specifická tím, že operace read-modify-write v této oblasti je provedena atomicky. Následující 0,5 GB je vyhrazen periferiím vestavěným do čipu. Další úseky jsou pro externí RAM a externí zařízení. Poslední blok paměti je vyhrazen pro komponenty systémové úrovně, vnitřní a externí sběrnice zařízení a periferie specifické pro konkrétního výrobce čipu. [1]



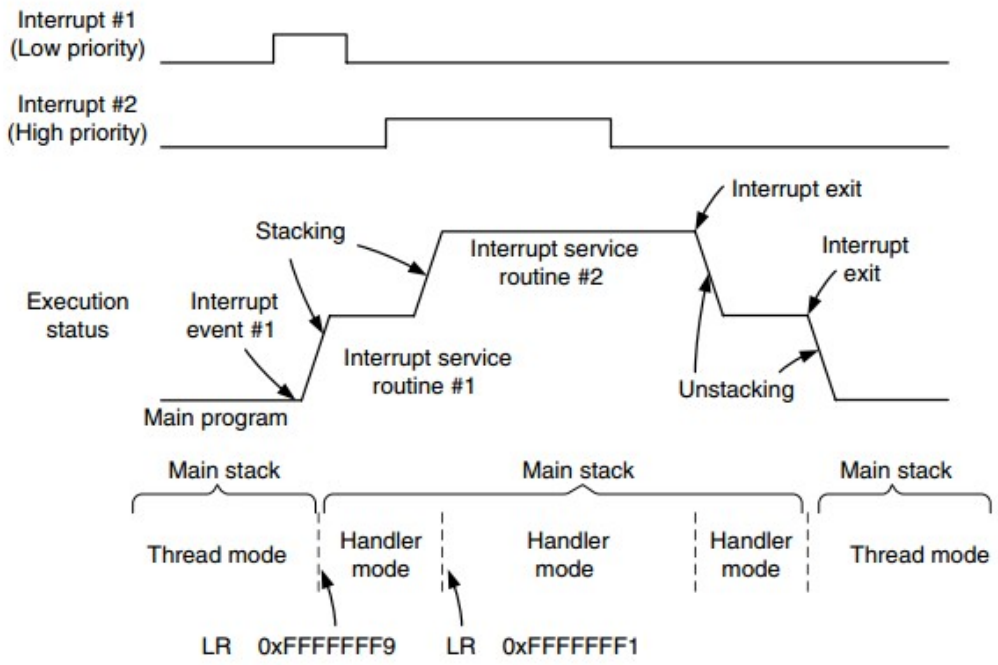
Obr. 5. Mapa paměti architektury Cortex-M4 [13]

1.3 Systém přerušení

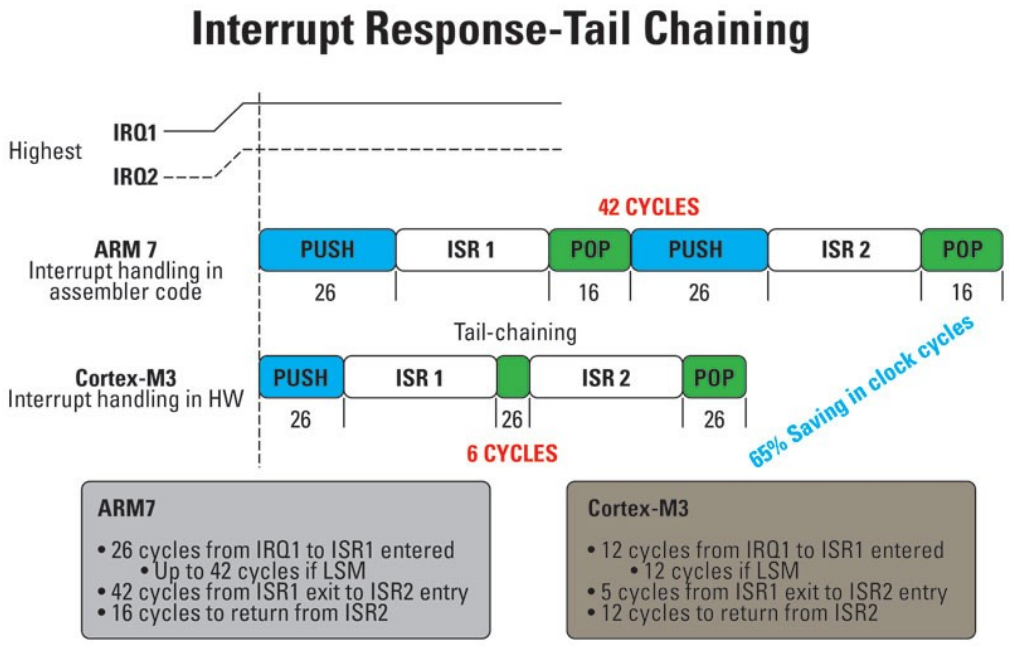
Cortex-M4 umožňuje použití více úrovní priorit přerušení. Také je možno nastavit preemptivní a nepreemtivní chování. Pokud je procesor nastaven na preemptivní chování a právě obsluhuje přerušení, během jehož obsluhy se vyskytne jiné přerušení s vyšší prioritou, je původně obsluhované přerušení pozastaveno a řízení přejde do obsluhy prioritnějšího přerušení. Po jeho dokončení se přejde do obsluhy původního přerušení, naváže se na místo, ve kterém bylo přerušeno a po dokončení se provede návrat do hlavního programu viz obrázek 6. [1]

Pokud se řízení nachází mimo obsluhu přerušení a dojde k příchodu přerušení, je jeho obsluha započata 12 cyklů po jeho příchodu. Pokud dorazí požadavek na přerušení během obsluhy jiného, méně prioritního přerušení, trvá přechod z obsluhy původního přerušení do obsluhy nově příchozího přerušení 6 cyklů.

Při obsluze přerušení a výskytu dalšího přerušení stejné nebo nižší priority se využívá tzv. tail chaining, kdy procesor po dokončení první obsluhy neprovádí tzv. unstacking(načtení obsahu registrů ze zásobníku, kam byl při přechodu do obsluhy přerušení uložen), ale přejde do obsluhy druhého přerušení, viz obrázek 7. [1]



Obr. 6. Znázornění zanoření obsluhy přerušení [1]



Obr. 7. Ukázka Tail chaining [8]

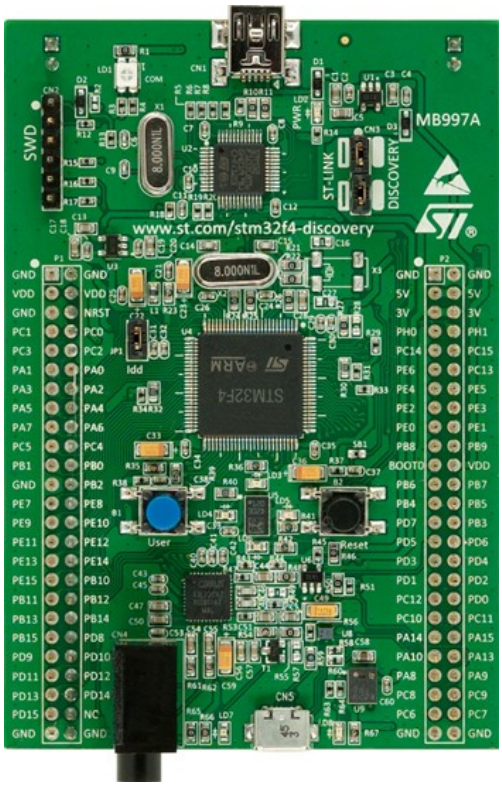
1.4 Procesory STM32F4

Řada procesorů STM32F4 je implementací architektury ARM Cortex-M4 firmou STMicroelectronics. Je vybavena jednotkou pro výpočty s desetinou čárkou, jednotkou pro zpracování digitálně reprezentovaných signálů. Maximální frekvence procesoru je 180 MHz. K dostání jsou varianty s velikostí flash paměti 256 kB až 2 MB, velikost operační paměti 96 až 384 kB. Procesory disponují USB 2.0, SDIO, USART, SPI a I2C rozhraním, 2 D/A převodníky, 3 12bitovými A/D převodníky a až 17 časovači.[11]

Při tvorbě této práce byly použity postupně procesory STM32F407VE, STM32F407VG, STM32F427VI. Důvodem byly rostoucí nároky na flash paměť procesoru.

Discovery kit je deska osazená procesorem STM32F407VGT6 a dalšími periferiemi sloužící pro základní seznámení s procesory této řady. Deska je osazena ST-Link programátorem, pomocí kterého je možné procesor programovat a ladit. Dále se na desce nachází USB konektor, audio jack, 4 LED diody, 2 tlačítka, externí krystal a konektory s vyvedenými piny procesoru. [11]

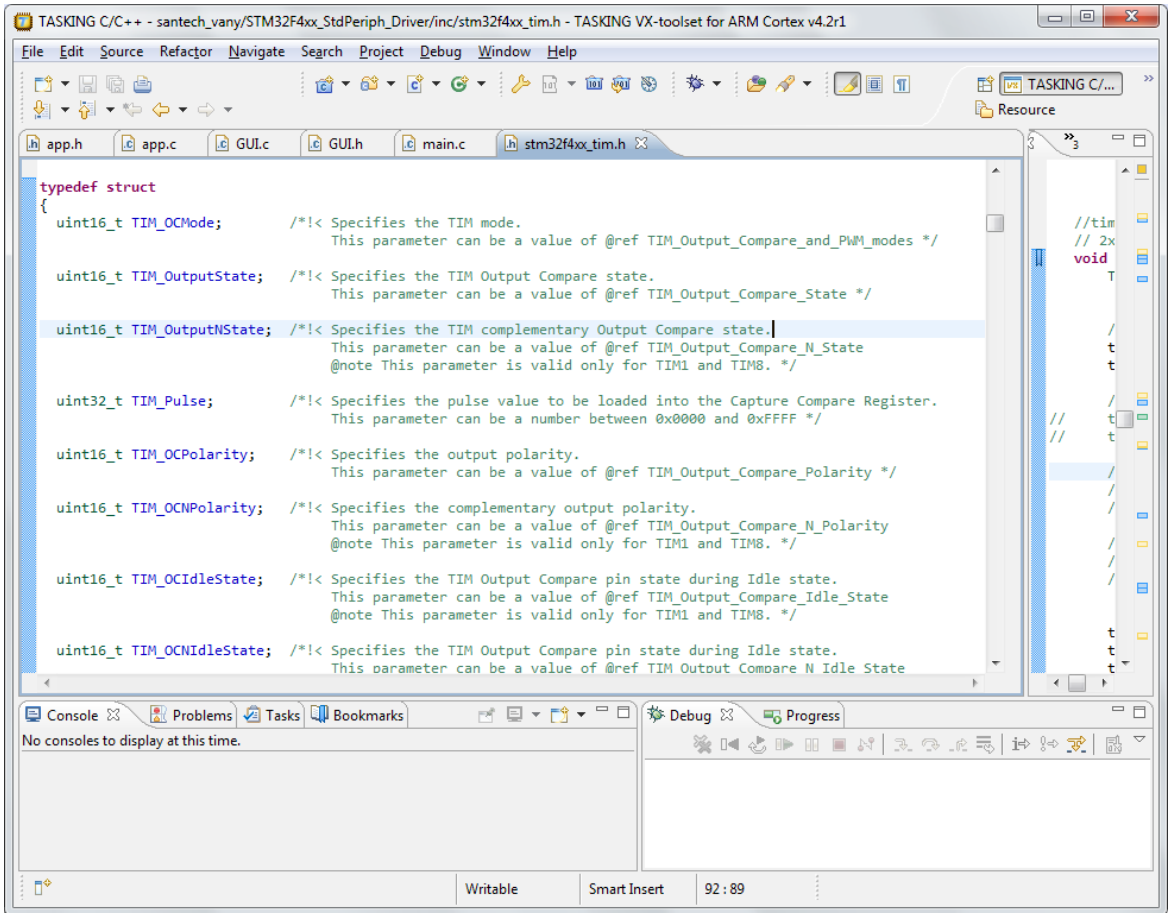
Tento kit byl použit při raném prototypování ovladače.



Obr. 8. Discovery kit s použitým procesorem STM32F4, měřítko 1:1 [11]

1.5 Programování procesoru

K programování a ladění bylo použito vývojové prostředí TASKING VX-toolset for ARM. Jde o integrované vývojové prostředí postavené na Eclipse. Jeho součástí je kompilátor pro řady procesorů Cortex-M0, M3, M4, debugger a linker. Programování je možné přes rozhraní JTAG nebo SWD. [15]



Obr. 9. ukázka vývojového prostředí Altium TASKING VX-toolset for ARM

Joint Test Action Group je standard definovaný normou IEEE 1149.1, tzv. Standard Test Access Port (TAP). Jedná se o architekturu Boundary-Scan pro testování plošných spojů, programování FLASH pamětí apod. JTAG je možné použít kromě primárního účelu, kterým je testování plošných spojů a interní funkce obvodů, také k programování flash pamětí, procesorů, FPGA, CPLD a dalších. K tomuto bylo vytvořeno několik standardů, např. IEEE 1532, JEDEC STAPL, nebo nestandardizovaný, ale hodně používaný Serial Vector Format [17].

Obr. 10. JTAG adaptér s připojeným prototypem ovladače

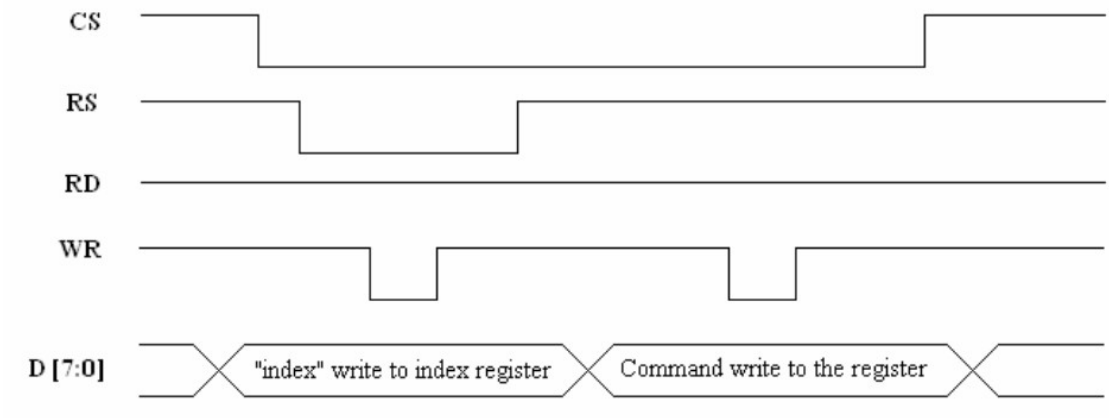
Serial Wire Debug(SWD) je debugovací rozhraní pro zařízení s omezeným počtem vývodů, což je případ některých provedení mikrokontrolérů nebo zákaznických integrovaných obvodů, kde počet pinů může mít velký vliv na cenu. [16]

SWD nahrazuje 5pinový JTAG port pomocí hodinového a datového pinu. Umožňuje přitom veškerou ladící a testovací funkcionalitu, kterou nabízí JTAG, navíc nabízí přístup k systémové paměti bez nutnosti zastavit procesor nebo nahrát speciální kód. SWD používá standardní obousměrný ARM protokol definovaný v ARM Debug Interface v5 k poslání dat z a do debuggeru a cílového systému. [16]

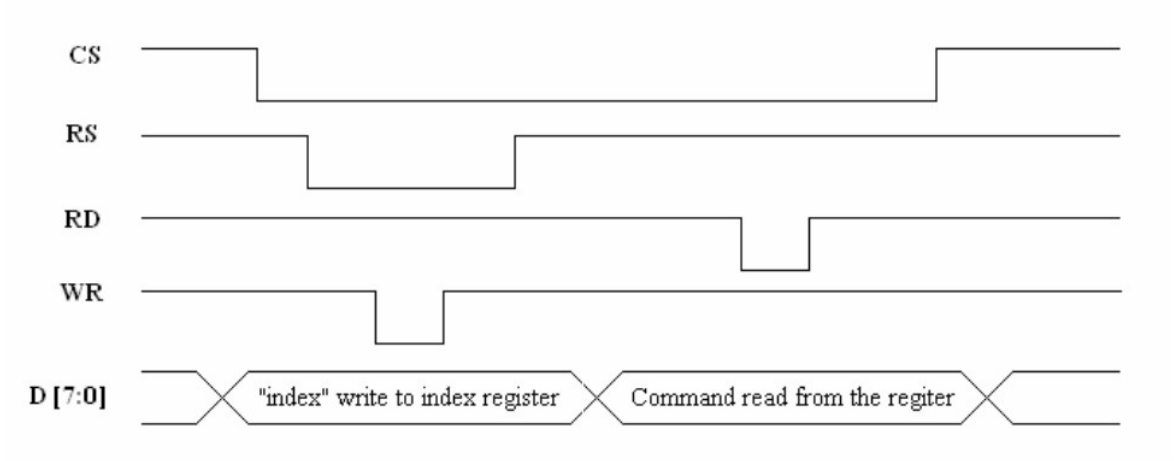
2 POUŽITÉ ŘADIČE A SW OVLADAČE LCD DISPLEJE

Použit byl TFT displej s rozlišením 400x240 pixelů. Displej je řízen řadičem HX8352-A. Ten dokáže pracovat s barevnými režimy 18 bitů na pixel(6-6-6) a 16 bitů na pixel (5-6-5). Je možno použít 8/16/18 bitové paralelní MPU rozhraní, 16/18 bitové paralelní RGB rozhraní a sériové rozhraní. Řadič je schopen pracovat v režimu Intel-80 a Motorola-68, výběr režimu je proveden pomocí signálu P68, ten ovšem není na konektor displeje vyveden a je tak možné použít pouze režim Intel-80.

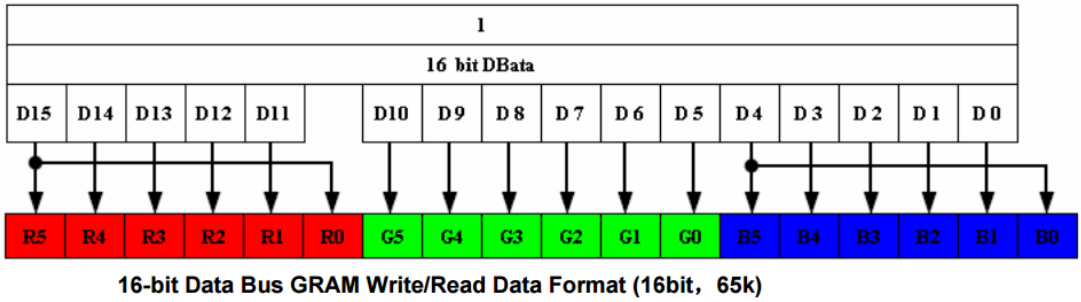
Obr. 11. Použitý displej [14]



Obr. 12. Časovací diagram displeje pro zápis dat [12]



Obr. 13. Časovací diagram displeje pro čtení dat [12]



Obr. 14. Formát dat pro zápis a čtení obrazových dat displeje [12]

2.1 Ovladač displeje a knihovna grafického uživatelského prostředí

K displeji a řadiči není dodáván žádný software. V době implementace nebyl k nalezení žádný hotový ovladač displeje pro naši platformu, bylo proto potřeba jej napsat.

Knihovny grafického uživatelského prostředí se najít dají, přímo STMicroelectronics nabízí 2 knihovny. První je STM32 embedded GUI library , kterou by bylo potřeba upravit pro použití na STM32F407, druhou je STemWin. Je možno dohledat další, ovšem bylo by potřeba je portovat na naši platformu, některé jsou navíc placené. Dalším problémem jsou požadavky na funkcionalitu. Zadavatel má specifické požadavky, např. konkrétní velikost fontů, widgety se třemi (výjimečně i více) stavy, specifický tvar widgetu.

- **STemWin -** v době realizace nebyla dostupná, k nalezení na http://www.st.com/web/catalog/tools/FM147/CL1794/SC961/SS1743/PF259225?s_searchtype=partnumber

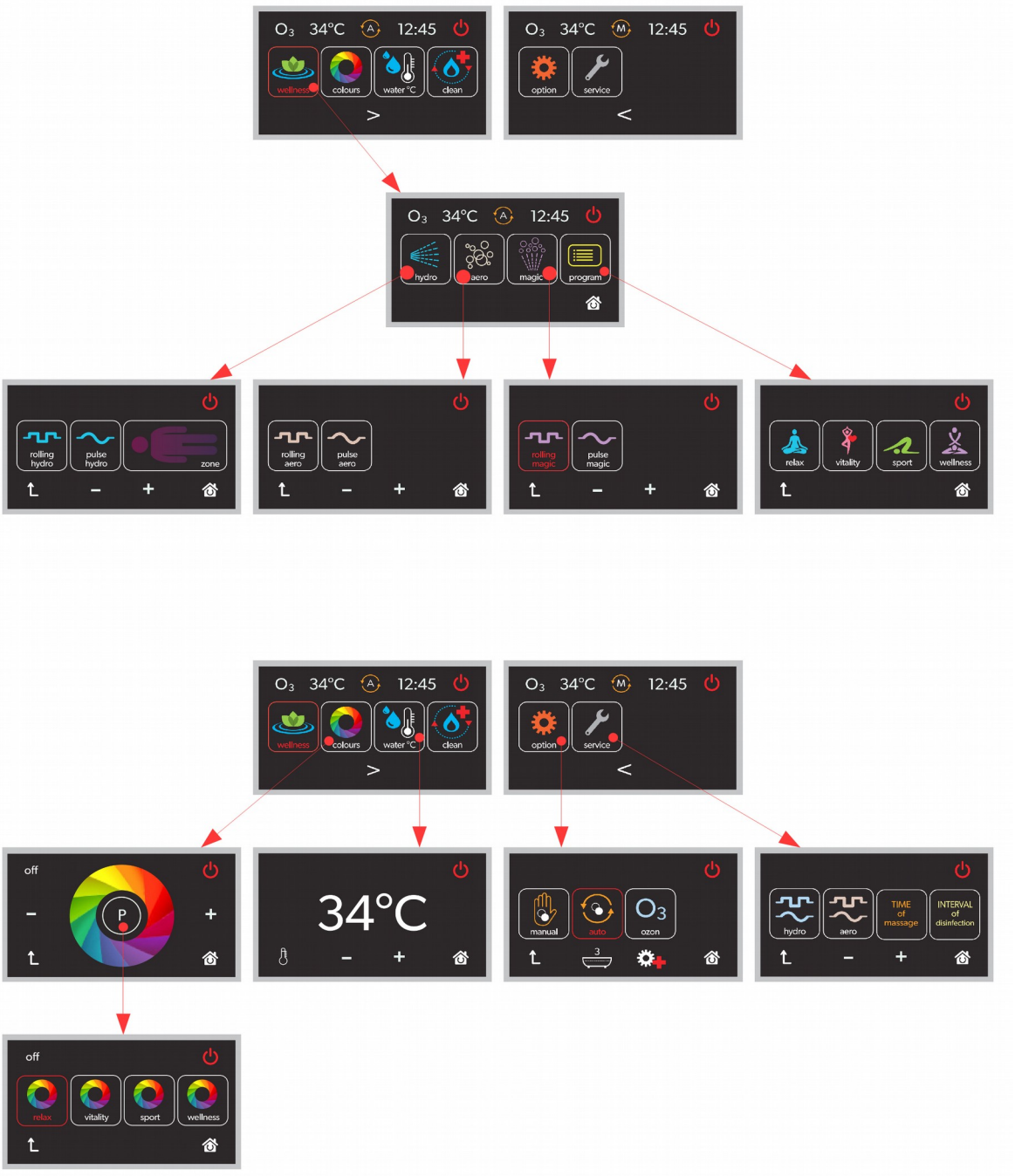
- **STM32 embedded GUI library** - není portována na STM32F4, omezený rejstřík velikostí fontů, dostupná z http://www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1743/LN1734/PF257934
- **Graphics Color Dot Matrix RGB driver library** - placená, dostupná na http://www.ramtex.dk/products/lcdd0129.htm

Součástí displeje je rezistivní dotyková vrstva. Její kontakty jsou vyvedeny přímo na konektor displeje, bylo proto potřeba je připojit na A/D převodník a napsat pro ni ovladač.

II. PRAKTICKÁ ČÁST

3 GRAFICKÉ UŽIVATELSKÉ PROSTŘEDÍ OVLADAČE

Grafický návrh vznikl ve spolupráci s externí firmou a zadavatelem. Nabídky jsou rozděleny do 3 úrovní. Mezi nimi je možno se pohybovat pomocí widgetů v prostředním a spodním řádku každé obrazovky. Ve většině obrazovek je widget pro přechod do hlavního menu a widget pro přechod zpět v historii procházení.



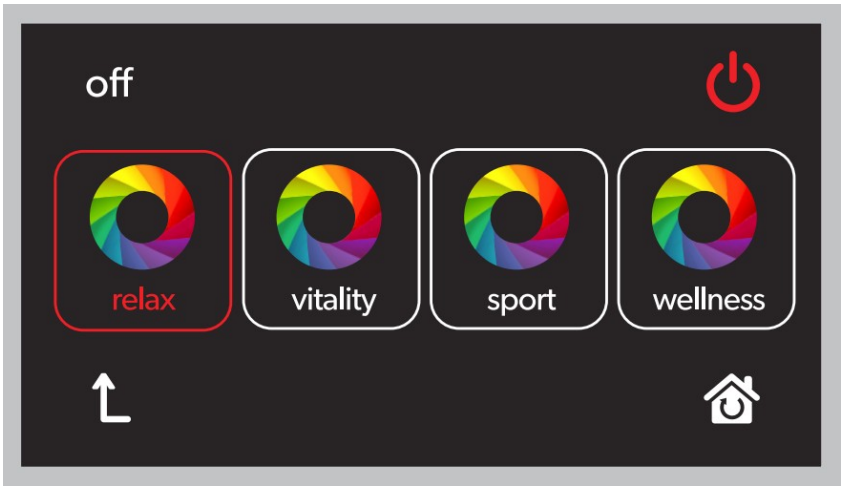
Obr. 15. Hierarchie uživatelských nabídek

Hlavní nabídka, která je zobrazena po startu ovladače nebo po stisku tlačítka home, má dvě části. Mezi nimi je možno přecházet šipkou ve spodní části viz obr. 16. V záhlaví hlavní nabídky (a některých dalších nabídek) jsou zobrazeny údaje o stavu ozonizátoru, o aktuální teplotě vody, nastaveném režimu automatické hygieny, čase zbývajícím do konce automatického programu.

Hlavním prvkem uživatelského prostředí je widget formátu jako na obr. 16 v prostředním řádku. Může být použit pro přecházení mezi nabídkami nebo pro změnu stavu funkce/parametru. Pokud je funkcí widgetu přechod do jiné nabídky, provede se po stisknutí přechod. Může nastat situace, že přechod není možný, např. při vypuštěné vaně nám ovladač nezpřístupní nastavení příhřevu, protože tento nemůže fungovat při vypuštěné vaně. V takovém případě je widget zašedlý jako 3. ikona v obr. 16.

Obr. 16. Ukázka aktivované funkce a neaktivovatelné funkce

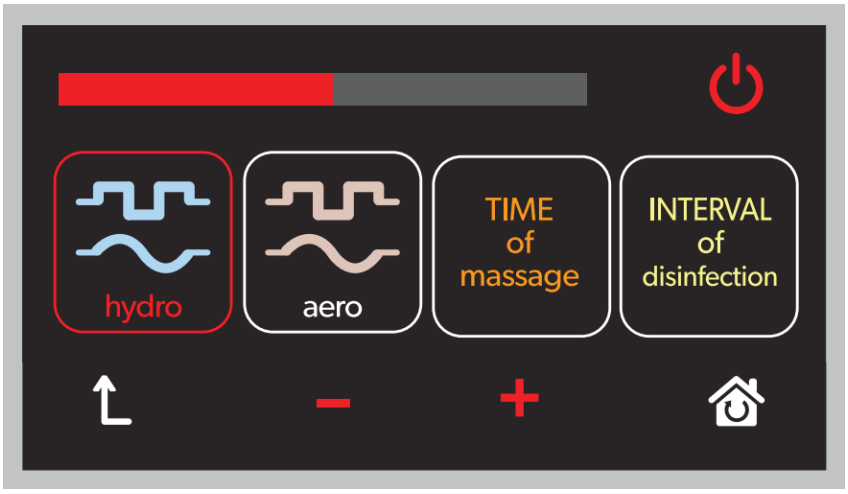
Některá tlačítka pro přechod do jiné nabídky slouží zároveň jako indikátory, jestli běží některá z funkcí, která pod tuto nabídku logicky spadá, např. pokud zapneme automatický program chromoterapie (který se nalézá v colours → program), bude widget colours v hlavní nabídce označen červeně, viz druhý widget na obr. 16. Takovýto widget zároveň umožňuje všechny jeho aktivované podfunkce vypnout pomocí dlouhého stisku. Můžeme tak například zmíněný aktivovaný automatický program chromoterapie vypnout dlouhým stiskem widgetu colours v hlavním menu nebo jej vypnout jeho zapínacím widgetem v nabídce chromo programů(levý widget na obr. 17.)



Obr. 17. Ukázka widgetu v aktivním stavu

Pokud je widget přepínačem nějaké funkce/parametru, je zároveň i indikátorem tohoto stavu, červené orámování značí aktivovanou funkci, bílý rámeček značí aktivovatelnou funkci a zašedlá ikona a zašedlý rámeček naznačuje, že danou funkci není možno provést, např. dříve uvedené zapnutí příhřevu při vypuštěné vaně.

Pokud je v některé nabídce potřeba nastavit parametr, který má více než 2 stavy, je toto zpravidla realizováno pomocí ukazatele intenzity v záhlaví a widgetů + a - v zápatí, viz obr. 18. Nastavení teploty příhřevu je provedeno mírně jinak, ukazatelem je číselná hodnota, nastavení se provádí pomocí widgetu + a - jako v předchozím případě, viz obr. 19.



Obr. 18. Ukázka prvku pro nastavení periody vln hydromasáže

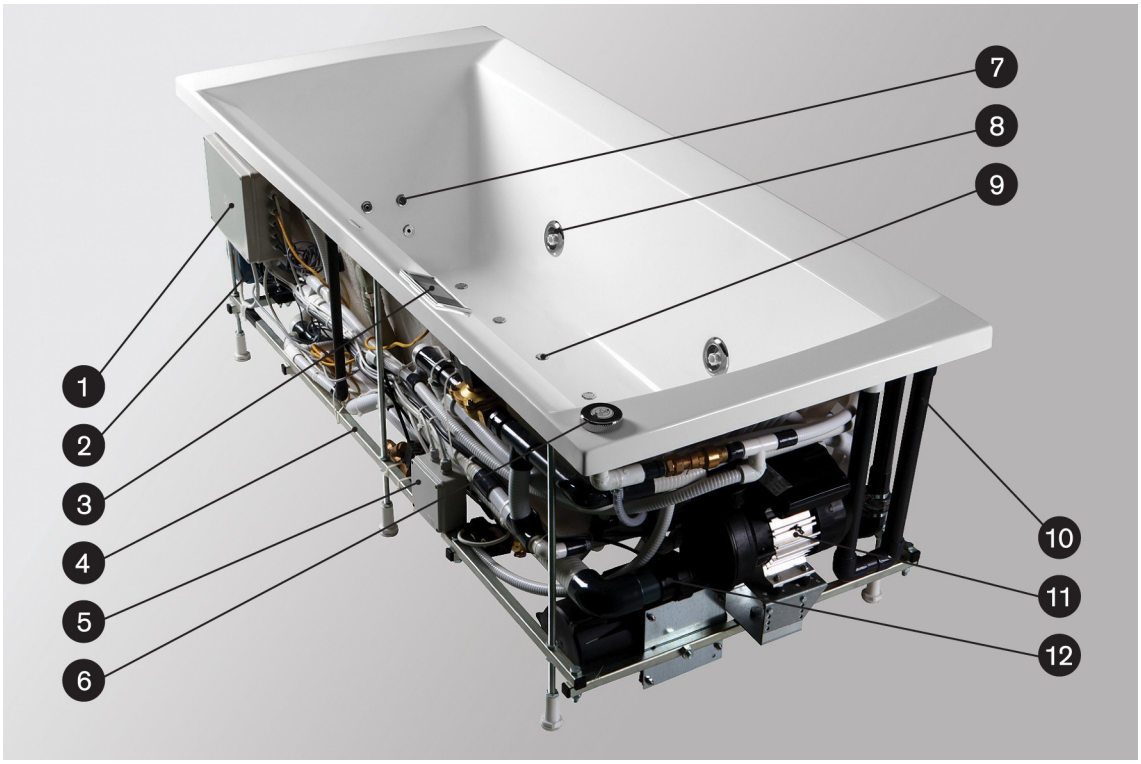
Obr. 19. Nabídka nastavení příhřevu

4 FUNKCIONALITA VANY, PŘÍTOMNÉ AGREGÁTY

Vana je vybavena následujícími agregáty:

- čerpadlo
- turbokompresor
- kompresor
- 4 servoventily
- elektroventil
- topné těleso
- RGB LED světlo

Masážní vana má ve svém dně a po stranách umístěny masážní trysky. Do vany je umístěno také čerpadlo a 2 kompresory, kterými je vytvářen proud vody a vzduchu. Ten je poté pomocí servoventilů usměrňován potrubím do trysek a tím plní hlavní účel vany – vodní masáž uživatele. Vana dále disponuje osvětlením pomocí RGB LED, příhřevem vody, samočistící funkcí a automatickým vypnutím.

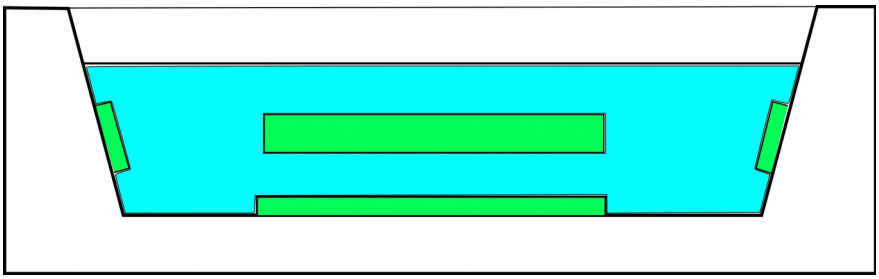


Obr. 20. Pohled na vanu s viditelnými agregáty [18]

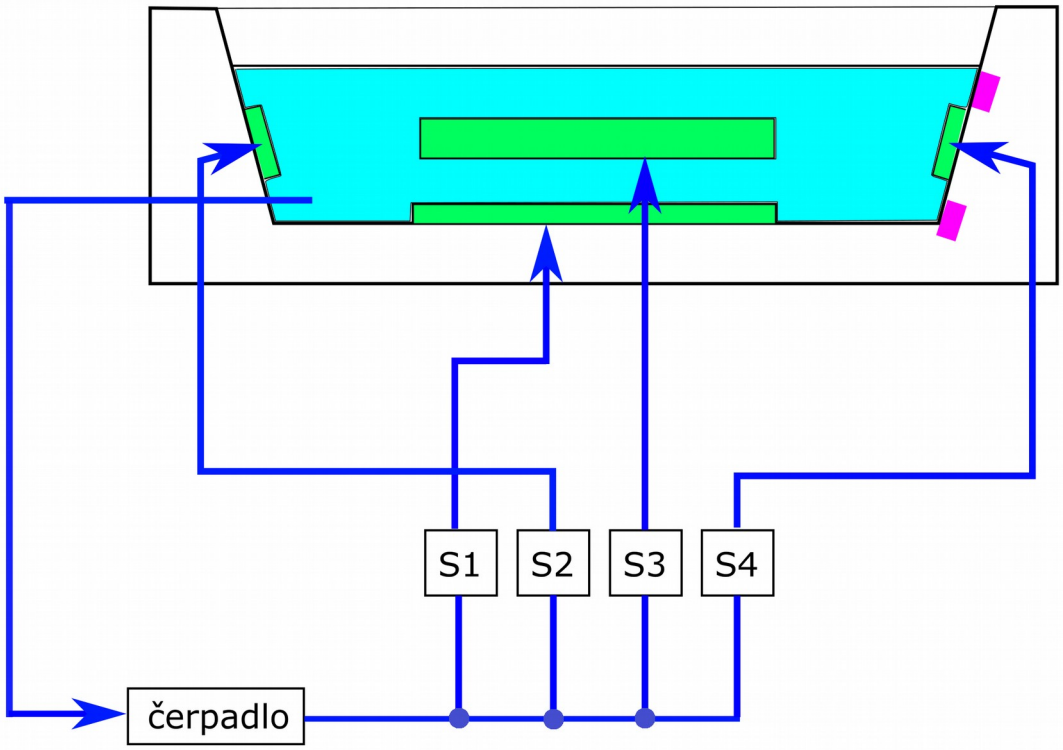
4.1 Trysky

Vana obsahuje 4 skupiny trysek jak je vidět na obrázku 21. Schématické znázornění umístění trysek je na obrázku 22. První skupina je určena pro masáž zad, trysky po stranách jsou určeny k masáži boků, třetí skupina k masáži nohou, trysky na dně umožňují tzv. perličkovou masáž.

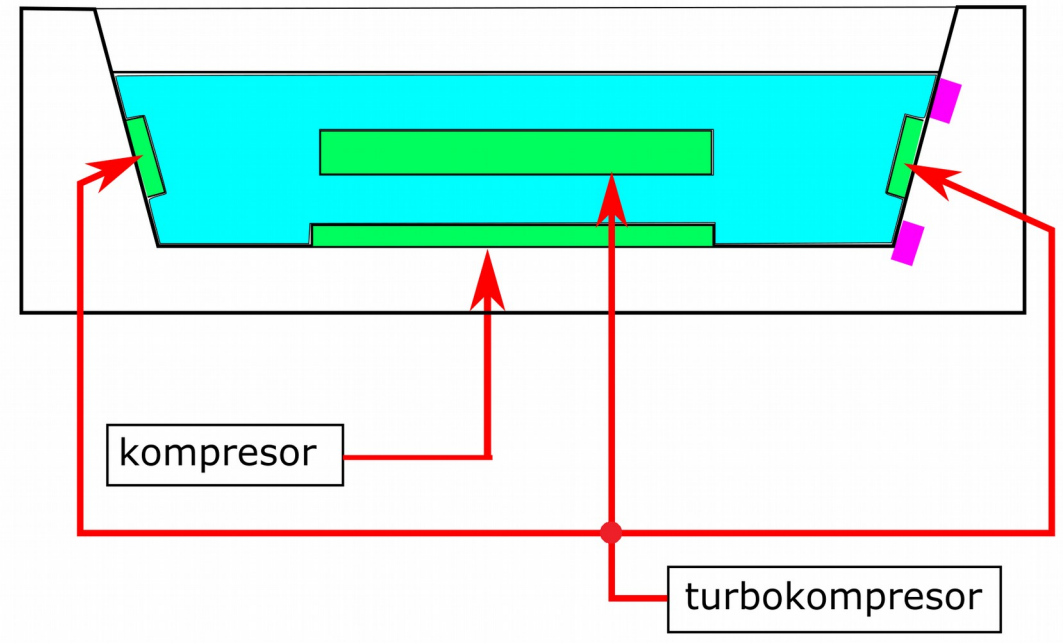
Obr. 21. Vana s viditelnými tryskami [18]



Obr. 22. Nákres umístění trysek v řezu vanou



Obr. 23. Schéma znázorňující vodní okruh



Obr. 24. Schéma vzduchového okruhu

4.2 Automatické masážní programy

Ovladač nabízí 4 automatické programy (wellness, relax, sport, vitality), které samy řídí čerpadlo, servoventily, kompresor, turbokompresor a LED světlo podle výrobcem daného schématu. Tyto programy mají nastavitelné trvání 10, 15, 20 minut. Aktivní může být vždy jen jeden automatický program. Pokud běží automatický program, není možno souběžně spustit masážní funkce aero, hydro, magic nebo samostatné ovládání LED světla. Po spuštění programu se v záhlaví obrazovky ovladače zobrazí čas zbývající do konce programu.

Obr. 25. Dialog výběru automatického programu

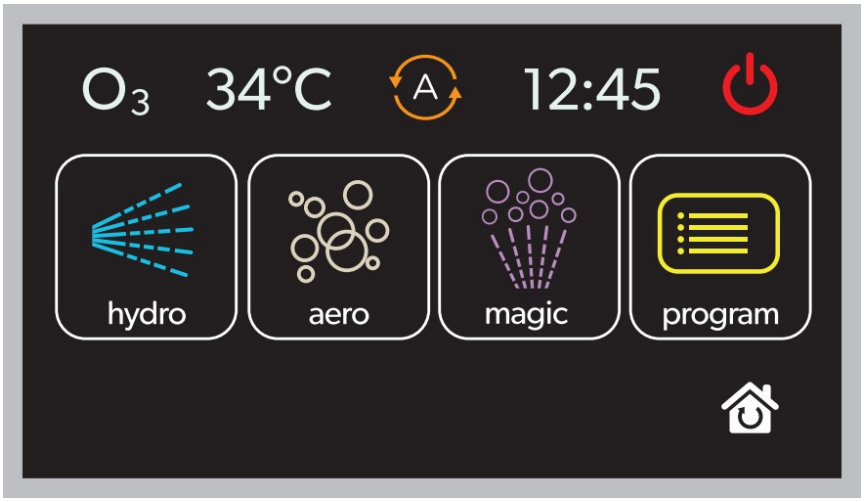
V tabulce 1 je ukázka rozpisu automatického programu „vitality“ jak jej dodal zadavatel. První řádek slouží jako časová osa. Velké písmeno P v celém řádku znamená, že čerpadlo má být po dobu 10 min zapnuto. Turbokompresor má být prvních 5 minut vypnutý, v 6. minutě má mít 20 % výkon, poté minutu měnit výkon harmonicky od minimální úrovně do maximální, v 8. minutě 80 % výkon, poté 30 %, poté opět sinusový průběh jako v 7. minutě. Z posledních 4 řádků můžeme vyčíst polohy servoventilů v jednotlivých časech programu. Velké písmeno S udává otevřený ventil, malé písmeno zavřený ventil.

Čas [min]	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
čerpadlo	P	P	P	P	P	P	P	P	P	P
turbokompresor	t	t	t	t	t	T2	Tval	T8	T3	Tval
kompresor	k1	K1	K3	k	k	k	k	k	K3	Kval
servoventil 1	S1	S1	S1	s1	s1	s1	s1	s1	S1	S1
servoventil 2	s2	s2	s2	s2	s2	S2	S2	S2	S2	S2
servoventil 3	s3	s3	s3	s3	S3	S3	S3	S3	S3	S3
servoventil 4	s4	s4	s4	S4	S4	S4	S4	S4	S4	S4

Tab. 1. Ukázka rozpisu prvních 10 minut automatického programu "Vitality"

4.3 Další masážní funkce

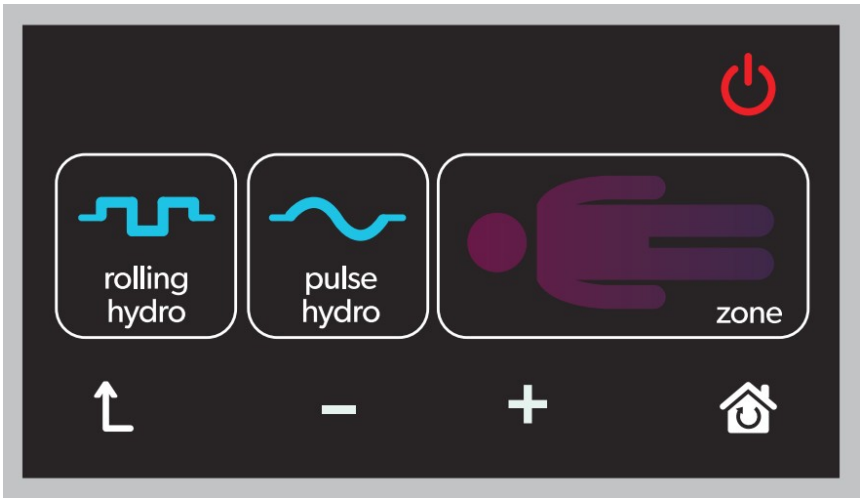
Ovladač nabízí kromě automatických programů ještě další masážní funkce, u kterých si může některé parametry nastavit sám uživatel. U těchto funkcí je možno si spustit nezávisle program chromoterapie(LED světla). Vzhledem k tomu, že tyto funkce používají stejné agregáty jako automatické programy, spuštěním některé z těchto masážních funkcí se vypne právě běžící automatický program. Souběh funkce hydro a funkce aero je možný, protože používají nezávislé větve a agregáty systému, ovšem při aktivované funkci magic není možno mít aktivní funkci aero nebo hydro.



Obr. 26. Hlavní nabídka masážních funkcí

4.3.1 Funkce hydro

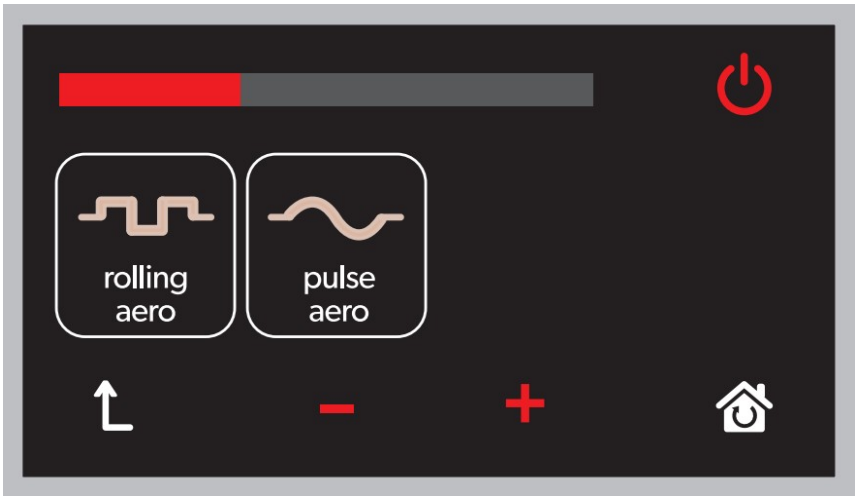
Funkce hydro umožňuje valivou a pulzní hydromasáž, kdy směs vody a vzduchu proudí do zádových, nožních a bočních trysek. Přitom se mění výkon turbokompresoru buď skokově – pulzní hydromasáž, nebo plynule – valivá hydromasáž. Další možností je hydromasáž s nastavitelným konstantním výkonem turbokompresoru. Pomocí tlačítek plus a mínus nastavíme požadovanou intenzitu masáže, opět proudí směs vody a vzduchu do všech trysek kromě trysek na dně vany. Poslední možná varianta hydromasáže je zónová hydromasáž. Pomocí symbolu postavy v dialogu hydromasáže (obr. 27) si můžeme vybrat skupinu trysek, která má být použita, trysky ve dně vany nelze při tomto typu masáže zvolit, výkon turbokompresoru je přednastaven a nelze jej měnit.



Obr. 27. Dialog nastavení funkce hydro

4.3.2 Funkce aero

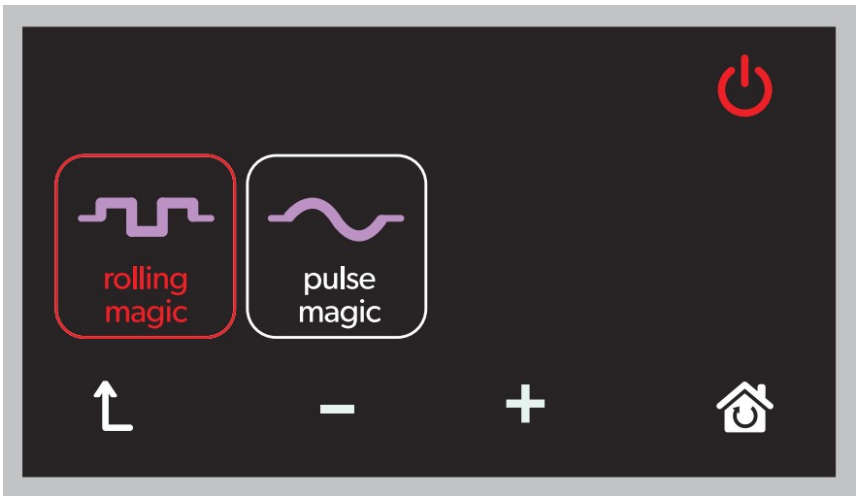
Při této funkci je kompresor zapnutý a tlačí vzduch přes trysky na dně vany. Ostatní trysky jsou neaktivní, čerpadlo je vypnuté. Podobně jako u hydromasáže je možno vybrat pulzní, valivou nebo nastavitelnou konstantní úroveň výkonu kompresoru.



Obr. 28. Dialog nastavení funkce aero, nastaven konstantní výkon kompresoru

4.3.3 Funkce magic

Funkce magic je podobná funkci aero, rozdíl je v zapnutém čerpadle. Jedinou otevřenou vodní větví je servoventil 1, ze spodních trysek vychází tedy směs vody a vzduchu. Výrobce tuto funkci označuje také jako „perlička“ nebo „Champagne“.



Obr. 29. Dialog nastavení funkce magic, zapnut valivý režim masáže

4.4 Ovládání RGB světla

Některé varianty vany mají ve svém těle zabudované RGB LED světlo, viz obrázek 30.

Obr. 30. Vana se zabudovanou RGB LED [18]

Ovladač dává uživateli možnost nastavit vybranou barvu pomocí barevného kruhu (viz obr. 31), zapnout jeden ze čtyř automatických programů, které periodicky mění barvy podle výrobcem daného schématu (obr. 32) nebo světlo vypnout. K nabídce automatických programů se dostaneme přes tlačítko uprostřed barevného kruhu. Ovládání světla je možné i při vypuštěné vaně.

Obr. 31. Požadovaný vzhled widgetu pro výběr barvy LED světla

Obr. 32. Dialog s nabídkou automatických programů RGB LED

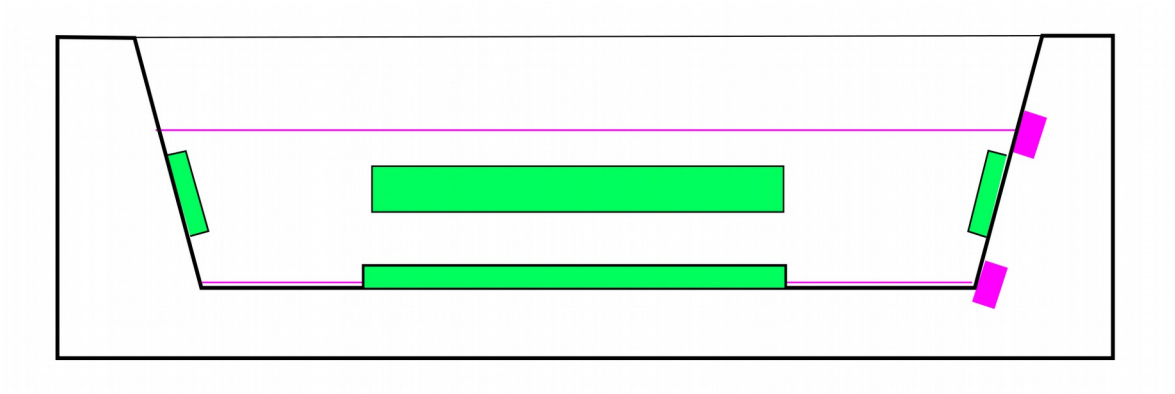
Automatické programy jsou výrobcem určené posloupnosti barev, k přepnutí barvy dochází vždy po minutě. Tyto automatické programy nemají danou dobu trvání, běží dokola, ukončit je může uživatel nebo automatické vypnutí ovladače. Pokud je spuštěn automatický wellness program(relax, vitality, sport, wellness), je automaticky spuštěn i odpovídající chromoprogram.

4.5 Samočistící program

Vana disponuje funkcí automatického čištění. Tato procedura spočívá ve spuštění turbokompresoru a kompresoru na 5 min po vypuštění vany, tím dojde k vyfouknutí zbytku vody z potrubí a jeho vysušení. Funkce se spouští automaticky po vypuštění vany a je možno ji deaktivovat.

4.6 Kontrola hladiny

Ovladač musí periodicky kontrolovat hladinu vody. Pokud klesne hladina pod horní mezní úroveň, dojde k vypnutí běžícího masážního programu a v uživatelském rozhraní se znepřístupní widgety pro výběr masážních funkcí. Pokud klesne hladina pod dolní mezní úroveň, dojde ke spuštění samočistícího procesu. Hladina je kontrolována každých 5 s.



Obr. 33. Schéma znázorňující umístění hladinových snímačů

5 KNIHOVNA GRAFICKÉHO UŽIVATELSKÉHO PROSTŘEDÍ

V rámci vývoje ovladače vznikla knihovna pro interakci mezi uživatelem a procesorem.

5.1 Hlavní požadavky

Hlavní požadavky, které musí ovladač(knihovna) pseudoparalelně splňovat:

- vykreslování obrazu na displej,
- čtení údajů z dotykové vrstvy,
- odesílání a příjem dat pomocí UART (posílání příkazů silové jednotce),
- aktulizace času a na něm závislých akcí (odpočet běžícího programu, posílání příkazů silové jednotce).

Hlavní požadavky na vykreslování:

- zobrazení widgetů,
- jednoduché přidávání a přepínání jazykových verzí widgetů,
- možnost zobrazit dekadické číslice ve dvou velikostech a dvou barvách.

Hlavní požadavky na logiku GUI:

- možnost přepínání jazykových verzí všech widgetů najednou pomocí jednoho widgetu,
- procházení mezi nabídkami stiskem widgetu(přepínání obrazovek),
- funkce zpět pro přechod do předchozí nabídky,
- widget pro přechod do hlavního menu,
- změna stavu funkce/proměnné stiskem widgetu(zapnutí automatického programu apod.).

5.2 Realizace pseudoparalelních procesů

Ovladač musí být schopen souběžně zpracovávat události ze svých vstupů. To je dosaženo použitím pseudoprocesů, které fungují obdobně jako procesy v běžném operačním systému počítače. Periferie procesoru umožňují při dosažení konkrétního stavu(např. přetečení

časovače) generovat přerušení. Tomuto přerušení je možno nastavit prioritu a tím ovlivnit chování při příchodu více přerušení najednou nebo při příchodu přerušení při již vykonávané obsluze přerušení viz kapitola 1.3: Systém přerušení. Procedury pro obsluhu těchto přerušení můžeme použít pro realizaci pseudoprocesů.

5.2.1 Proces obsluhy UART

Tento pseudoproces je časově nejvíce kritický, protože události na tomto vstupu jsou vůči ostatním procesům nejkratší. Například příjem jednoho byte trvá při zde použitých parametrech přenosu přibližně 400 μs, což je podstatně méně než např. dotyk prstu na dotykové vrstvě. Pokud by protistrana ovladače posílala data a proces obsluhy UART by byl blokován jiným bežícím procesem, buffer pro příjem dat by byl naplněn po ukončení příjmu prvního byte a všechny následující, které do ovladače dorazily před vyprázdněním vstupního bufferu, by byly zahozeny. Tento proces proto musí mít nejvyšší prioritu ze všech procesů. Obsluha tohoto přerušení je spuštěna při dokončení příjmu byte nebo při dokončení odesílání byte. Po dokončení příjmu zkopíruje obsluha přerušení nově přijatý byte do pole za dříve přijaté a pokud je přijatý byte ukončovacím znakem zprávy, je přijatá zpráva předána k provedení.

5.2.2 Vykreslovací proces

Vykreslovací proces je spouštěn periodicky každých 20 ms jako obsluha přerušení časovače. Vykreslení obrazu se skládá z vykreslení barvy pozadí na celém displeji, vykreslení widgetů a vykreslení dalších prvků - ukazatel času, teploty a další. Při nejjednodušším možném přístupu, kdy se při každém průchodu vykreslovací funkcí vykreslily znova všechny prvky, docházelo k viditelnému problikávání obrazu. Proto bylo potřeba tento proces zefektivnit.

Překreslovací proces má statickou proměnnou, do které si vždy na konci funkce uloží ukazatel na vykreslenou nabídku(řádek 2 v ukázce níže). Při dalším průchodu se porovná tento ukazatel a ukazatel na nabídku, která se má vykreslit. Pokud došlo ke změně(tedy uživatel přešel do jiné nabídky mezi předchozím a tímto vykreslením), dojde k překreslení celého displeje barvou pozadí(řádek 8).

Dále se pokračuje nezávisle na předchozím porovnání, je zavolána funkce GUI_view_draw, která vykreslí všechny widgety dané nabídky a poté další funkce vykreslující prvky uživatelského rozhraní, které nepatří mezi widgety. Jde o vykreslení

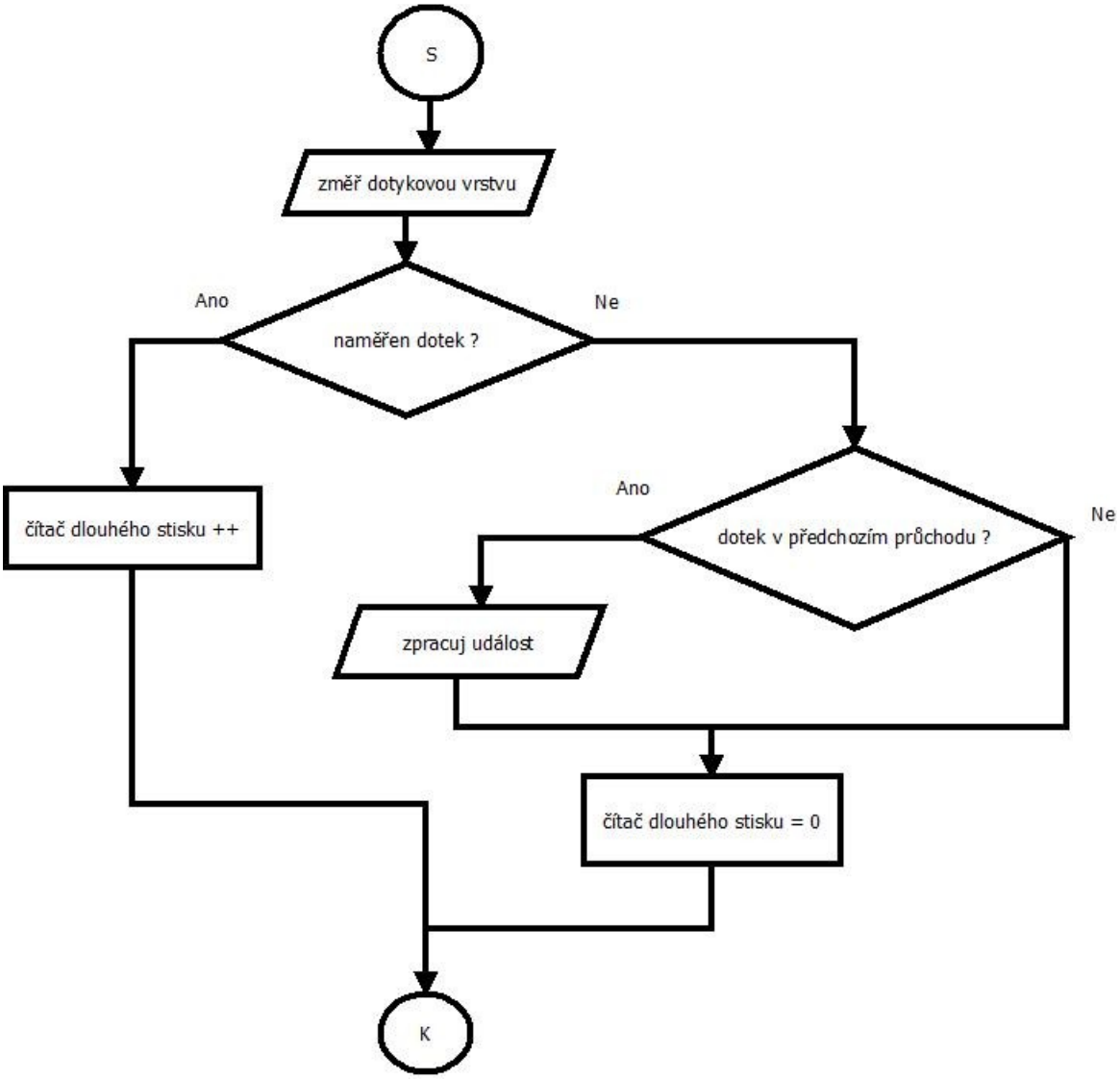
nastaveného režimu automatické hygieny(řádek 12), vykreslení zbývajícího času běžícího programu, vykreslení nastavené teploty příhřevu, vykreslení ikony informující o nízké hladině vody, vykreslení nastaveného režimu ozonizátoru.

Nakonec je aktualizován ukazatel na nabídku, která byla vykreslena v tomto průchodu vykreslovací funkcí.

```
1  void TIM5_IRQHandler(void){
2      static GUI_view_t *previousState_actualView;
3
4      if (TIM_GetITStatus(TIM5, TIM_IT_Update) != RESET){
5          TIM_ClearITPendingBit(TIM5, TIM_IT_Update);
6
7          if((actualView != previousState_actualView)){
8                LCD_Driver_Paint16(BACKGROUND_COLOR);
9
10         GUI_view_draw(actualView);
11
12         app_drawHygieneState();
13         app_drawRemainingTime2();
14         app_drawDesiredTempInHeading();
15         app_drawWLIndicator();
16         app_drawOzonState();
17
18         previousState_actualView = currentView;
19
20     }
21 }
```

5.2.3 Proces čtení dotykové vrstvy

Proces je tvořen obsluhou přerušení časovače TIM3. Je spouštěn každých 10 ms. Při každém spuštění je nejdříve provedeno měření, jestli je na vrstvě dotyk, pokud ano, je změřen v obou osách. Proces dotyk zpracuje až při puštění displeje, zároveň předává informaci, jestli jde o dlouhý stisk.



Obr. 34. Diagram procesu čtení dotykové vrstvy

5.2.4 Proces aktualizace času

Tento proces má na starost především řízení automatických programů. Spouští se každých 500 ms jako obsluha přerušení časovače TIM2.

V tomto procesu jsou volány funkce provádějící následující činnosti:

- kontrola hladiny vody
- kontrola, zda má být spuštěna samočistící procedura
- poslání příkazů silové jednotce podle času, ve kterém se nachází masážní program
- poslání příkazů silové jednotce podle času, ve kterém se nachází chromo program(automatický program LED světla)
- aktualizace času zbývajícího do konce automatického programu

- aktualizace času zbývajícího do automatického vypnutí

5.2.5 Hlavní programová smyčka

Hlavní programová smyčka je tvořena nekonečným cyklem, ve kterém je monitorována fronta příkazů určených k poslání silové jednotce. Silová jednotka nedokáže přijmout víc než jeden příkaz najednou. Další příkaz může ovladač poslat až obdrží odpověď o úspěšném provedení od silové jednotky. Toto značně komplikuje návrh. Provedení jednoho příkazu může trvat až 3 s. Pokud ovladač potřebuje poslat sadu příkazů, musí je posílat samostatně, po vyslání příkazu počkat na odpověď a až poté může poslat další příkaz. Z toho důvodu je v ovladači implementována fronta, která je obsluhována z hlavní smyčky programu. Pokud smyčka zjistí přítomnost zprávy ve frontě, inicializuje její poslání přes UART, dál už se o poslání stará obsluha přerušení jednotky UART.

5.3 Datový typ GUI_widget_t

```
1 typedef struct  GUI_widget_t{
2     uint8_t status;
3     int value;
4     int x;
5     int y;
6     GUI_image_t *image_on[NUMBEROFLANGUAGES];
7     GUI_image_t *image_off[NUMBEROFLANGUAGES];
8     GUI_image_t *image_selected[NUMBEROFLANGUAGES];
9     int multiLanguage;
10    struct GUI_widget_t ** pGroup;
11    enum GUI_eventCode_t eventCode;
12};
```

Složka status udává v jakém stavu se widget nachází. Typicky slouží pro přepínání stavů nemožno nastavovat, vypnuto a zapnuto, viz obr. 35.

Obr. 35. 3 stavy widgetu

Složka value není GUI knihovnou použita, může ji ale použít uživatelská aplikace. Slouží k uložení hodnoty logicky spjaté s widgetem, např. intenzita funkce hydro viz obr. 18 na straně 26.

Složky x a y jsou souřadnice, na kterých má být widget vykreslen. Jde o souřadnice levého horního rohu.

Image_on, *image_off, *image_selected jsou ukazatele na proměnné typu GUI_image_t. Podle stavu widgetu, ve kterém se nachází, tedy podle složky status, je vybrán k vykreslení odpovídající obrázek. Pokud je funkce reprezentovaná widgetem vypnuta, tedy atribut status je 0, vykreslí se obrázek odkazovaný ukazatelem image_off. Jde o pole ukazatelů, každý prvek v poli je jednou jazykovou verzí obrázku.

Prvek multiLanguage informuje GUI knihovnu o tom, zda je daný widget jazykově univerzální. Mnohé widgety nepotřebují rozdílné obrázky, např. vypínací tlačítko. Prvek multiLanguage je v takovém případě nastaven na 1 a GUI knihovna poté používá pouze první prvek v polích image_on, image_off, image_selected.

Ukazatel pGroup slouží pro zařazení widgetu do skupiny widgetů, které jsou nějak logicky spjaty. Pokud např. máme 3 widgety, kdy aktivní může být pouze jeden z nich, vytvoříme skupinu, na kterou odkazujeme přes složku pGroup a GUI knihovna při stisku některého widgetu sama obstará vypnutí ostatních widgetů. Pokud je potřeba implementovat jinou logickou souvislost widgetů mezi sebou, je potřeba ji implementovat až v uživatelské aplikaci.

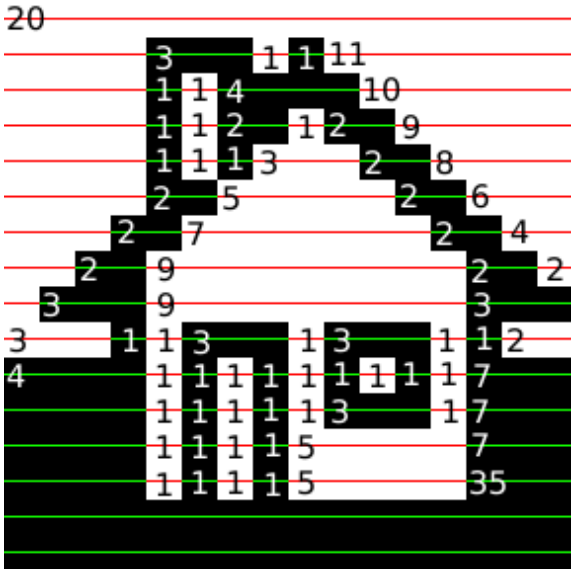
EventCode je kód události. Jde o výčtový typ. Jeho hodnota je při stisku widgetu předána funkci zpracovávající události z dotykové vrstvy.

5.4 Uložení bitmap ve flash paměti

Bylo potřeba najít kompromis mezi úrovní komprese a rychlostí vykreslování. Řadič displeje přijímá data pomocí 16 bitové paralelní sběrnice ve formátu 5-6-5, tedy 5 bitů pro červenou složku přenášeného pixelu, 6 bitů pro zelenou složku a zbylých 5 bitů pro modrou složku. Uložení takovéhoto formátu v nekomprimované podobě je problém z hlediska kapacity flash paměti procesoru. Jeden snímek přes celou obrazovku použitého displeje by zabral 240x400x2 = 188 kB, takže by se do interní paměti procesoru vešlo takovýchto obrázků jen pět. Běžné formáty, např. JPEG, používají vyspělé kompresní techniky jako diskrétní cosinová transformace. Dekódování takového formátu není vzhledem k výkonu procesoru možné. S ohledem na to, že ovládací tlačítka zobrazované na displeji jsou vždy jednoduché symboly, bez barevných gradientů, s jednolitým pozadím, nabízí se použití RLE kódování.

5.4.1 Jak funguje RLE

RLE ukládá obrázek jako posloupnost počtu opakování stejného pixelu. Pokud budeme chtít zakódovat obrázek 36, podívá se kódovací algoritmus na první pixel, poznačí si jeho barvu nebo index do barevné palety, prochází obrázek dále po řádcích a počítá přitom pixely, které prošel. Až narazí na pixel jiné barvy(černý pixel na 2. řádku), dodá k zapsané barvě/indexu do palety počet opakování tohoto předchozího pixelu a začne počítat počet opakování nového pixelu. Obrázek 33 by tedy byl zakódován: 20x bílá, 3x černá, 1x bílá, 1x černá...



Obr. 36. Naznačení principu RLE komprese, rozměr obrázku je 16 x 16 pixelů[36]

5.4.2 Použití RLE v projektu

Barvy v obrázcích jsou uloženy pomocí indexované palety, obrázek tedy není popsán posloupností počet pixelů * jejich barva, namísto barvy pixelu je použit odkaz do palety barev. Takováto posloupnost je zapsána jako 2 bajty, první udává počet opakování, druhý je indexem do palety, viz 1., 2. a 5. řádek úryvku kódu níže. Pokud je posloupnost tvořena jen jedním pixelem, je tento uložen jako 1 bajt, který slouží jako index do palety, viz 3, a 4. řádek v ukázce. Potřebujeme nějak rozlišit jednobajtové zápisy samostatných pixelů od dvoubajtových zápisů posloupností. To je dáno nejvyšším bitem bajtu, pokud při dekódování obrázku narazíme na bajt např. 0x94(první řádek ukázky), je jeho nejvyšší bit 1, víme tedy, že jde o počet opakování, a jeho hodnota je hodnota bajtu bez jeho MSB, tedy v rozsahu 0 až 127. Druhý bajt je potom indexem do barevné palety. Pokud má první načtený bajt MSB nulový, jde o zápis samostatného pixelu formou indexu do palety.

```
1 0x94,0x01,    // 20 opakování barvy na indexu 0x01:
2 0x83,0x00,    // 3 opakování barvy na indexu 0x00:
3 0x01,         // 1 pixel barvy na indexu 0x01
4 0x00,         // 1 pixel barvy na indexu 0x00
5 0x8b,0x01,    // 11 opakování barvy na indexu 0x01:
```

Výše zmíněné rozlišení pomocí MSB omezuje počet užitečných hodnot v bajtu na 128, barevná paleta tak může mít maximálně 128 barev. Každý obrázek má ovšem paletu vlastní, celkově počet barev v uživatelském prostředí není omezen. Omezení barevné palety na 128 barev na obrázek se při použitém grafickém návrhu ukázalo jako neomezující.

5.4.3 Výsledky použití RLE

Každý obrázek má svou paletu (v obrázku 37 pro vyšší přehlednost zkrácena), následuje pole, jehož prvky jsou indexy do barevné palety nebo (pokud je nejvyšší bit prvku 1) udává počet iterací prvku následujícího.

```
#include "GUI.h"
// Hicolor (16bits) 565 model used in palette colors...
// Palette size reported by Java=256
const unsigned short palette_program_selected[]={
    0x2104,0x2104,0x28e4,0x2904,0x28e4,0x2904,0x30e4,0x30e4,
    0x48e4,0x48e4,0x4904,0x48e4,0x4904,0x50e4,0x50e4,0x5104,
    0x70e4,0x70e4,0x70e5,0x7104,0x78e4,0x78e4,0x7104,0x88c4,
    0x90e4,0x90e4,0x88e4,0x8104,0x8104,0x90e4,0x98c4,0x8104,
    0x98e4,0xa8c5,0x98e4,0xa8c4,0xa0e4,0xa104,0xb0c4,0xb0c4,
    0xb8e4,0xc8a4,0xb0e4,0xc0c4,0xc8c4,0xc0c5,0xd884,0xb8e4,
    0xc8e5,0xd8c4,0xd8c4,0xe0c4,0xe8a4,0xd8e4,0xd0e4,0xe0c4,
    0x5284,0x6b04,0x7b84,0x8405,0x9465,0xa4c5,0xa524,0xb564,
};  // 127 items generated in the palette
const unsigned char pixels_program_selected[]={
// 11 repetitions of 0x01:
    0x8b,0x01,0x0e,0x2c,0x3f,
// 68 repetitions of 0x54:
    0xc4,0x54,0x50,0x36,0x17,
// 21 repetitions of 0x01:
    0x95,0x01,0x23,0x55,
```

Obr. 37. Ukázka začátku zdrojového kódu obrázku 38

Obr. 38. Ukázka widgetu

```
// 2 repetitions of 0x23:
    0x82,0x23,
// 14 repetitions of 0x01:
    0x8e,0x01,

};  // size of this pixel array = 2152 bytes, original size=9603.
    // Resulting size = 22% of the original.
#define program_selected_WIDTH 97
#define program_selected_HEIGHT 99
#define program_selected_RLE 1
#define program_selected_USEDITEMS 127
#define program_selected_TEMP 2152

GUI_image_t program_selected = {program_selected_WIDTH, program_sel
```

Obr. 39. Ukázka konce zdrojového kódu obrázku 38

Jak můžeme vidět v ukázce kódu, velikost widgetu na obrázku 38 se po použití implementované komprese snížila na 22 % původní velikosti.

5.5 Snímání souřadnic dotyků

Použitý displej nemá vlastní řadič pro dotykovou vrstvu, bylo proto potřeba připojit vývody fólie na vstupy A/D převodníků. Vrstvy jsou dvě, obě jsou připojeny na piny procesoru, které mohou fungovat jako Push-Pull výstup i jako vstup A/D převodníku. Pro zjištění souřadnic dotyku nastavíme piny, na které je první vrstva připojena jako GPIO výstupy, jeden na logickou jedničku, druhý na logickou nulu. Tím nám mezi piny jedné fólie vznikne napětí 3,3 V. Fólie mezi piny je rezistor. Jeden z pinů, na který je připojená druhá vrstva nastavíme jako vstup A/D převodníku a druhý pin nastavíme do stavu vysoké impedance. Pokud nyní dojde k doteku, vrstvy se v místě doteku prstu vodivě propojí a na převodník se dostane napětí úměrné souřadnici v měřené ose. Po změření souřadnice dojde k záměně nastavení pinů. Napájecí pár pinů bude nyní měřícím a naopak.

Pokud došlo k doteku, je tato událost předána smyčce vykonávající zpracování událostí a je proveden odpovídající kód.

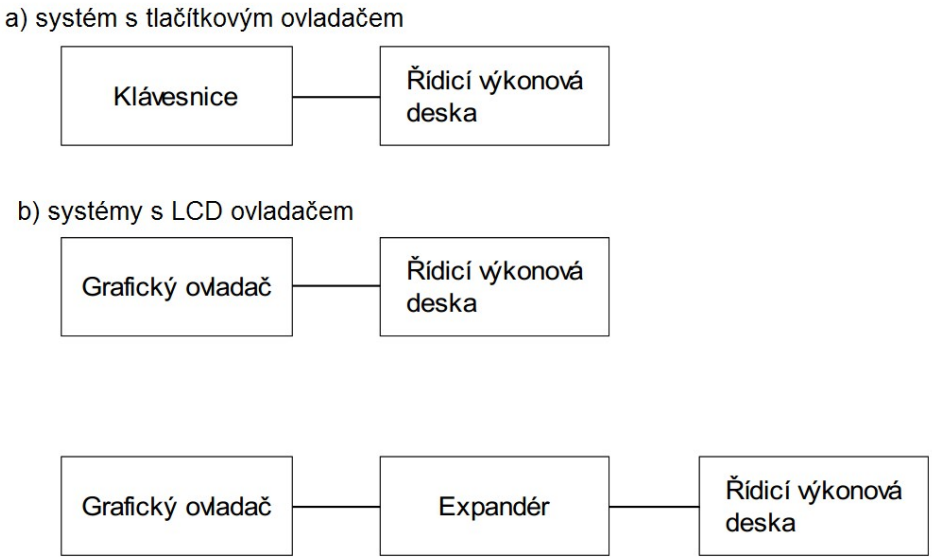
U rezistivních fólií existuje problém, že každá má mezi póly mírně odlišný odpor, nemůžeme proto zjistit koeficienty z jednoho displeje a aplikovat jiný, protože ovladač by nám vracel dotyk nepřesně. Při prvotním spuštění ovladače je proto automaticky spuštěna kalibrace, která si potřebné údaje naměří. Prvním krokem je naměření mezní hodnoty pro rozhodování, zdali došlo k doteku nebo ne. Poté jsou změřeny 4 body v rozích displeje, pro každý bod jsou změřena napětí v obou osách.

Jelikož dotyková vrstva je jediným ovládacím prvkem masážní vany, je potřeba zajistit a ověřit její plnou funkčnost. Po změření předchozích 4 bodů je proto potřeba ověřit, jestli byla kalibrace provedena správně. Na displeji se postupně zobrazí 5 bodů, jejich souřadnice jsou generovány náhodně. Obsluha se musí postupně těchto bodů dotknout. Ovladač po každém doteku podle dříve naměřených koeficientů porovná právě naměřené hodnoty s hodnotami vypočítanými podle koeficientů zjištěných při kalibraci. Pokud si neodpovídají v mezích tolerance, je počítadlo úspěšných doteků vynulováno a je generováno dalších 5 bodů. Pokud bylo 5 bodů po sobě provedeno správně, naměřené koeficienty jsou uloženy do flash paměti jako platné hodnoty a jsou načítány při spuštění ovladače. Pokud byl v úvodní fázi dotek proveden nesprávně, stačí ovladač odpojit od napájení, koeficienty nebudou uloženy do flash paměti, příznak úspěšně provedené

kalibrace nebude nastaven a při dalším spuštění se kalibrace spustí znova.

6 KOMUNIKACE SE SILOVOU JEDNOTKOU

Jelikož ovladač neovládá agregáty přímo, nýbrž pomocí silové jednotky, bylo potřeba navrhnout protokol pro komunikaci těchto dvou systémů. Elektronika silové jednotky je rozdělena na dvě části – řídicí výkonovou desku a expandér. V levnějších variantách vany, které nabízejí méně funkcí a nejsou osazeny všemi agregáty, není potřeba veškerá řídicí silová elektronika a expandér je proto vynechán, funkce vany jsou ovládány pomocí jednoduchého tlačítkového ovladače. Vyšší varianta má řídicí výkonovou desku, grafický LCD ovladač a volitelně expandér. Předmětem této práce je varianta s LCD ovladačem, řídicí výkonovou deskou a expandérem.



Obr. 40. Možné konfigurace systému

6.1 RS422

RS422 je standard, který definuje napěťové úrovně na sběrnici, ostatní parametry přenosu jsou dány vyššími vrstvami komunikačních protokolů.

6.2 Připojení zařízení na společnou sběrnici

Jak bylo uvedeno dříve, napěťové úrovně jsou dány standardem RS422, jinak jde o sériovou komunikaci UART s parametry: Data length 8 bit, 19200 bps, 1 stop bit.

6.3 Příkazy pro silovou jednotku

Obě části tvořící silovou jednotku nikdy komunikaci neinicializují, vždy pouze odpovídají na příkazy ovladače.

Příkazy pro řídicí výkonovou desku:

• **SOUT,<P, Z, H, L>,<1;0>%CRC** – pomocí tohoto příkazu ovládáme zařízení jako je čerpadlo (P), ionizátor (Z), topení (H) nebo světlo (L); výběr zařízení, které se má ovládat, provádíme prvním parametrem. Hodnota 1 resp. 0 ve druhém parametru znamená zapnuté zařízení resp. vypnuté. Dále je tímto příkazem možné ovládat i jednotlivé ventily, případně servoventily, viz následující příkazy.

SOUT,H,1%--- Tímto příkazem zapneme topné těleso.

• **SPWR,<T,K>,<1;0>,<0–100>%CRC** – tímto příkazem ovládáme kompresor nebo turbokompresor. Znak T v prvním parametru znamená turbokompresor, znak K je vyhrazen pro kompresor a ve druhém parametru rozsah 0–100 pak znamená výkon na který má být zařízení nastaveno. Příklad použití:

SPWR,T,1,50%--- Odesláním tohoto příkazu spustíme turbokompresor na poloviční výkon, CRC součet není odesílán a je nahrazen znaky ---

• **GVAL,<P, Z, H, L, S, V>,<n>** – pokud je potřeba zjistit v jakém stavu se nachází jednotlivá zařízení, které řídicí deska umožňuje ovládat, použijeme tento příkaz. Prvním parametrem nastavíme zařízení, na které se budeme dotazovat (P=čerpadlo, Z=ionizátor, H=topení, L=světlo, S=servoventil a V=ventil). Druhý parametr „n“ znamená, u kolikátého zařízení čtené periferie se má vyčíst parametr. Pro periferie, které nemají více stejných zařízení, se případné číslo v parametru příkazu ignoruje a provede se vyčtení stavu. Odesláním příkazu pod tímto odstavcem zjistíme stav topení:

GVAL,H,0%---

Příkazy pro expandér:

• **SOUT,S,<1–4>,<1,0>%CRC** – parametr S je vyhrazené písmeno pro ovládání servoventilů, druhý parametr má rozsah 1–4 a určuje, který servoventil je ovládán. Následující parametr má dvě hodnoty: 1 znamená otevřený servoventil a číslice 0 pak uzavření servoventilu. Následujícím příkazem uzavřeme servoventil číslo 4:

SOUT,S,4, 0%---

• **SOUT,V,<1–2>,<1,0>%CRC** – k ovládání elektroventilů (písmeno V v prvním parametru) použijeme tento příkaz. Číslo 1 nebo 2 ve druhém parametru určuje, který ventil je ovládán a stejně jako u předchozího příkazu, následující číslice 1–0 určuje, zda je ventil otevřen (1) nebo uzavřen (0). Na dalším řádku je příklad použití příkazu, který otevírá elektroventil číslo 1:

SOUT,V,1,1%---

• **SLED,<R, G, B>,<0–9>%CRC** – tento příkaz slouží k ovládání LED, kde R, G, B určuje barvu LED a 0–9 intenzitu svícení. Intenzitu svícení LED je možné regulovat dvojím způsobem. První způsob je plynulá změna intenzity v deseti krocích (0–9) s plynulým přechodem z jedné úrovně jasu na druhou. Druhou možností je přímé řízení intenzity svícení LED v 255 krocích – v tomto případě je přechod skokový. Místo rozsahu 0–9 je možné dosadit číslo v rozsahu 100–355. Přičemž hodnota 100 je vypnutí LED a 355 je maximální intenzita. Výhodou použití tohoto rozsahu je možnost jemného ovládání intenzity svícení LED. Příklady příkazů pro ovládání LED:

SLED,R,0%--- Znamená plynulé vypnutí červené LED bez použití CRC součtu.

SLED,G,128%--- Tento příkaz znamená nastavení zelené LED na poloviční intenzitu svícení bez použití CRC součtu.

• **GVAL,<R,G,B>,<r>** – vyčtení jasu ovládané LED diody, první parametr určuje barvu (R, G, B), druhý parametr určuje vrácený rozsah hodnot: pro r=0 je vrácen rozsah 0 – 9, pro r=1 rozsah 0 – 255. V obou případech rozsah odpovídá rozsahu úplného zhasnutí po plný svit.

GVAL,R,1%---

Společné příkazy:

• **GVAL,X,<n>** – zjištění stavu hladinového snímače připojeného k desce řídicí výkonové jednotky (n=0) nebo k desce expandéru (n=1, n=2 podle čísla snímače) :

GVAL,X,0%--- nebo GVAL,X,1%--- (na první variantu odpovídá deska řídicí výkonové jednotky, na druhou odpovídá deska expandéru)

• **PING,<M,E>** – jestliže potřebujeme ověřit, zda mikrokontrolér na desce expandéru nebo řídicí desce je schopen přijímat data. Parametrem M (Master = řídicí deska) nebo E (Expandér) provádíme výběr desky, kterou budeme volat (viz. příklad pod odstavcem).

PING,E%--- Odesláním tohoto příkazu zjistíme připravenost expandéru ke komunikaci.

ZÁVĚR

V první kapitole je popsána architektura ARM Cortex-M4 a její implementace v podobě procesoru STM32F4. Druhá kapitola popisuje použitý displej, jeho řadič, možnosti komunikace a jeho dotykovou vrstvu. Kapitola se dále zabývá nalezením již existujícího ovladače displeje pro náš procesor, vhodné knihovny grafického uživatelského prostředí a zhodnocením, zda existující SW portovat pro naši aplikaci nebo implementovat vlastní.

Další část práce seznamuje čtenáře s realizací požadavků zadání. Třetí kapitola popisuje navržené uživatelské prostředí, hlavní uživatelské prvky a hierarchii nabídek. Čtvrtá kapitola zmiňuje umístění a funkci trysek, přítomné agregáty a jejich vzájemné propojení, aby mohly být vysvětleny hlavní funkce ovladače vany. Následuje popis funkce nabízených automatických masážních programů, manuálních masážních funkcí, možností řízení RGB LED světla a samočistící procedury. Pátá kapitola shrnuje požadavky na vytvořenou knihovnu grafického uživatelského prostředí, aby bylo možné implementovat dříve uvedenou funkcionalitu vany. Vysvětluje použití pseudoparalelně běžících procesů, u každého vysvětluje jeho hlavní účel, periodu spouštění nebo událost vedoucí k jeho spuštění. Důležitou částí kapitoly je také vysvětlení funkce, popis implementace a výsledek použití RLE. Poslední kapitola popisuje navržený a implementovaný protokol pro komunikaci se silovou jednotkou a způsob jejího propojení s ovladačem.

Výsledkem práce je ovladač implementující 4 automatické masážní programy s nastavitelnou dobou trvání, 3 masážní funkce, automatické a manuální řízení LED světla a samočistící proceduru. Součástí řešení je realizace navrženého uživatelského prostředí. Ovladač je připraven na přidání dalších jazykových verzí, které je potom možno jednoduše přepínat. Aktuální využití paměti jednou jazykovou verzí a kódem aplikace je přibližně 25 % interní 2 MB flash paměti procesoru, v záloze je ještě externí 4 MB flash paměť připojená přes SPI rozhraní. Při realizaci firmware ovladače byl kromě aplikace samotné implementován ovladač displeje, jeho dotykové vrstvy, ovladač jednotky UART, algoritmus pro dekódování RLE a knihovna grafického uživatelského prostředí. Práce na projektu bude pravděpodobně pokračovat přidáním jazykových verzí a přidáním možnosti aktualizace firmware pomocí bluetooth.

SEZNAM POUŽITÉ LITERATURY

[1] YIU, J. *The Definitive Guide to the ARM Cortex-M3*. Elesevier, 2007. ISBN 978-0-7506-8534-4.

[2] SLOSS, A., D. SYMES a Ch. WRIGHT. *ARM System Developer's Guide.* Elesevier, 2004. ISBN 1-55860-874-5.

[3] VALVANO, J. W. *Embedded Systems: Real-Time Interfacing to Arm® Cortex(TM)-M Microcontrollers.* CreateSpace Independent Publishing Platform, 2011. ISBN 978-1463590154.

[4] VALVANO, J. W. *Embedded Systems: Introduction to ARM® Cortex™-M Microcontrollers*. CreateSpace Independent Publishing Platform, 2013. ISBN 978-1477508992.

[5] VALVANO, J. W. *Embedded Systems: Real-Time Operating Systems for the ARM® Cortex™-M Microcontrollers.* CreateSpace Independent Publishing Platform, 2012. ISBN 978-1466468863.

[6] SEAL, D. *ARM Architecture Reference Manual.* Addison-Wesley, 2001. ISBN 978-0201737196.

[7] SLOSS, A., D. SYMES a C. WRIGHT. *ARM System Developer's Guide.* Morgan Kaufmann, 2004. ISBN 978-1558608740.

[8] MARTIN, T. *The Insider's Guide To The STM32 ARM Based Microcontroller.* Hitex (UK) Ltd., 2008. ISBN 0-9549988 8.

[9] KOLÅS , Ø. Chapter 1: Digital image representation [online]. [cit. 2015-05-12]. Dostupné z: http://pippin.gimp.org/image_processing/images/rle.png.

[10] ARM Ltd. Cortex-M Series. *ARM*. [online]. © 2014 [cit. 2015-05-12]. Dostupné z: http://www.arm.com/products/processors/cortex-m/.

[11] STMicroelectronics. STM32F4 Series - STMicroelectronics. *STMicroelectronics*. [online]. © 2015 [cit. 2015-05-12]. Dostupné z: http://www.st.com/web/en/catalog/mmc/FM141/SC1169/SS1577?s_searchtype=keyword.

[12] Himax Technologies, Inc. DATASHEET: HX8352-A (T).*Display Future*. [online]. Version 05. 12-2008 [cit. 2015-05-12]. Dostupné z: http://www.displayfuture.com/Display/datasheet/controller/HX8352.pdf

[13] ARM Ltd. ARM Infocenter. *ARM*. [online]. © 2014 [cit. 2015-05-12]. Dostupné z: http://infocenter.arm.com/help/index.jsp.

[14] ITEAD Intelligent Systems Co.Ltd. 3.2 Inch TFT Widescreen Panel with Resolution 400×240. *iMall*. [online]. © 2015 [cit. 2015-05-12]. Dostupné z: http://imall.iteadstudio.com/im120906009.html

[15] Altium BV. Tasking - C compiler for ARM Cortex-M series (VX-toolset). *Tasking*. [online]. © 1977-2015 [cit. 2015-05-13]. Dostupné z: http://www.tasking.com/products/arm/.

[16] ARM Ltd.. Serial Wire Debug – ARM. *ARM.* [online]. © 2014 [cit. 2015-05-13]. Dostupné z: *http://www.arm.com/products/serial-wire-debug.php*

[17] Přispěvatelé Wikipedie. *Joint Test Action Group* [online]. Wikipedie: Otevřená encyklopedie. © 2013, Datum poslední revize 9. 03. 2013, 20:12 UTC, [cit. 13. 05. 2015]. Dostupné z: http://cs.wikipedia.org/w/index.php?title=Joint_Test_Action_Group&oldid=9874929

[18] Santech, s.r.o. Vany | Santech. *Santech.* [online]. © 2011 [cit. 2015-05-13]. Dostupné z: *http://www.santech.cz/*

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

A/D	Analog/Digital
CPLD	Complex Programmable Logic Device
CRC	Cyclic redundancy check
D/A	Digital/Analog
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
GUI	Graphical user interface
HW	Hardware
I2C	Inter-Integrated Circuit
IRQ	Interrupt request
JEDEC	Joint Electron Devices Engineering Council
JPEG	Joint Photographic Experts Group
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least significant bit
MSB	Most significant bit
RAM	Random Access Memory
RGB	Red Green Blue
RLE	Run-length encoding
RTOS	Real-Time Operating System
SDIO	Secure Digital Input Output
SPI	Serial Peripheral Interface
SRAM	Static Random Aces Memory
SWD	Serial Wire Debug
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous/Asynchronous Receiver and Transmitter
USB	Universal Serial Bus

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH