TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Využití platformy STM32 F4/F7 pro návrh HMI panelů

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Stanislav Horváth**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Development of Human-Machine Interfaces on STM32 F4/F7 platform

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Stanislav Horváth**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2015/2016

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Stanislav Horváth**
Osobní číslo:	**M13000103**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Využití platformy STM32 F4/F7 pro návrh HMI panelů**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s možnostmi vývojových kitů STM32 discovery a způsobem vývoje aplikací pro zvolený RTOS či bez OS.
2. Navrhněte a vytvořte grafickou aplikaci pro vizualizaci a podpůrné knihovny pro vyčítání a archivací dat z dostupných periferií zvoleného kitu - zejména Modbus TCP anebo RTU.
3. Ověřte a demonstrujte správnou, spolehlivou a efektivní funkci Vámi vytvořené aplikace.
4. Shrňte dosažené výsledky a diskutujte výhody a nevýhody zvoleného řešení.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30–40 stran**

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

Seznam odborné literatury:

[1] **ST MICROELECTRONICS. UM1907: Discovery kit for STM32F7 Series with STM32F746NG MCU [online]. 2015 [cit. 2015-10-01]. Dostupné z: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/user_manual/DM00190424.pdf**

[2] **ST MICROELECTRONICS. DB2582: Discovery kit with STM32F746NG MCU [online]. 2015 [cit. 2015-10-01]. Dostupné z: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00179227.pdf**

[3] **KMB SYSTEMS. PA 144, SMC 144, SMY 133, SMZ 133, ARTIQ 144 Multifunctional Panel Meters & Power Quality Analyzers Protocol description for Modbus TCP and Modbus RTU protocol: For device firmware revision 2.0 [online]. 2015 [cit. 2015-10-01]. Dostupné z: http://www.kmb.cz/index.php/cs/component/phocadownload/category/14-dokumenty-komunikace?download=320:komunikacni-protokol-modbus-tcp-a-rtu-en-fw-v2-0**

Vedoucí bakalářské práce: **Ing. Jan Kraus, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Viktor Bubla**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, 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 (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

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

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16.5. 2016

Podpis: 

Poděkování

Chtěl bych tímto poděkovat vedoucímu práce Ing. Janu Krausovi, Ph. D. za odborné konzultace, užitečné rady a za poskytnutí veškeré potřebné techniky.

Abstrakt

V této práci se seznamuji s vyvíjením embedded softwaru především na platformě STM32. Popisuji zde vývojové a ladící nástroje od společnosti STMicroelectronics, knihovny pro práci s periferiemi jako je karta SD, Ethernet či LCD. Výsledkem práce je aplikace s grafickým uživatelským rozhraním, ve kterém jsou uživateli zobrazována data, načtená z protokolu modbus TCP či RTU. Aplikace využívá vlastní knihovny pro generování dotazu a zpracování odpovědi z výše uvedeného protokolu, které jsou zde též popsány.

Klíčová slova

Vestavěné systémy, Modbus, STM32, HMI, Procesory ARM

Abstract

In this work, I get acquainte with developing embedded software platform primarily to STM32. Here I describe the development and debugging tools from STMicroelectronics, library for working with peripherals such as an SD card, Ethernet or LCD. The result is an application with a graphical user interface in which the user displays the data retrieved from Modbus TCP or RTU. The app uses a custom library for query generation and processing responses from the Modbus protocol, which are also described.

Keywords

Embedded systems, Modbus, STM32, HMI, ARM processors

Obsah

Seznam symbolů, zkratek a termínů

- OS – operační systém
- RTOS – operační systém reálného času
- HMI – humanmachine interface
- LwIP – lightweight TCP/IP
- IDE – vývojové prostředí
- DHCP – Dynamic Host Configuration Protocol
- DMA – Direct Memory Access
- Modbus – průmyslový protokol
- Vlákno/Task – jedno či několik paraelně běžících úloh

Úvod

Procesory ARM nás obklopují ze všech stran, nalezneme je v moderních mobilních telefonech, tabletech, routerech či v tiskárnách. Důvodem tak obrovského rozšíření je skvělý poměr cena/výkon. Pokud bychom si chtěli ARM procesor pořídit, tak máme na výběr z široké škály modelů, lišící se výkonem a uplatněním. V případě stavby jednoduchého převodníku signálu nebo transceiveru zvolíme procesor na bázi Cortex-M0 (např.: STM32F0). Pokud chceme ovládat grafický display a vyžadujeme vyšší výkon, sáhneme po procesoru alespoň s Cortex-M4 (např.: STM32F4).

Zajímavým uplatněním těchto procesorů je v průmyslovém odvětví, protože nám umožňují programování realtime aplikací. V prvním vlákně nám může běžet například kontrola stavu vozidla (soft-deadline), ve druhém vlákně bezpečnost pasažéra (hard-deadline) a třetí vlákno může například zobrazovat na displeji chybové hlášení uživateli (firm-deadline). Získáváme tak multifunkční systém s výhodami systému reálného času za příznivou cenu. Dalšími výhodami mohou být například malé rozměry výsledného produktu a nízká spotřeba, čehož je využíváno kupříkladu v mobilních zařízeních.

Tato práce pojednává o problematice vývoje procesorů ARM především na platformě STM32 a její použití v HMI panelech. V teoretické části se zmiňuji o aktuálních procesorech na trhu, jenž se hodí pro vývoj grafických aplikací, uvádím jejich výhody a nevýhody. Dále se zmiňuji o některých operačních systémech, které je možno na tyto platformy nasadit. Zbývající část je již zaměřena na platformu STM32, popisuji discovery kity F4 a F7 a porovnávám je mezi sebou. Představím knihovny a nejrůznější programy pro vývoj embedded softwaru. Nalezneme zde ladící nástroje, ale i souhrn vývojových prostředí (komerční i nekomerční). Ve druhé polovině se věnuji vývoji konkrétní aplikace, která má, mimo jiné, za úkol zobrazovat data získaná z měřících přístrojů pomocí protokolu modbus TCP/RTU v grafickém uživatelském rozhraní (GUI). Aplikace pracuje s dotykovým displejem a poskytuje uživateli grafické ovládací prvky pro snazší ovládání aplikace. Navrhnu strukturu aplikace, podpůrné knihovny pro vyčítání a ukládání dat na kartu a strukturu konfiguračních souborů. Na konci praktické části práce nalezneme testování klíčových funkcionalit aplikace. Konkrétně testování latence webserveru, rychlost operací s paměťovou kartou a práci s modbus TCP.

2 Dostupné platformy a systémy

Platforma **STM32** je vyvíjena společností STMicroelectronics, která se zabývá průmyslovými řešeními. Jedná se o rodinu 32bitových procesorů architektury ARM, která nabízí jak modely s velmi nízkou spotřebou tak i relativně výkonné [1]. Výhodou volby této platformy je dobrá podpora a nízká cena. STMicroelectronics zdarma poskytuje k vývoji na svých produktech mnoho různých programů a utilit, které ušetří čas se psaním a laděním kódu.

SAM9X je řada procesorů od společnosti Atmel na platformě SMART ARM. Atmel poskytuje tuto řadu procesorů právě pro vývoj HMI panelů. [2] Atmel poskytuje vývojářům zdarma své IDE, jenž pojmenovali Atmel Studio, se kterým lze naprogramovat více jak 300 různých procesorů Atmel AVR či Atmel SMART [3]. Nevýhodou tohoto řešení je vysoká cena.

i.MX je název rodiny procesorů postavené na architektuře ARM od společnosti NXP, určené pro vývoj GUI aplikací, protože poskytuje skvělou podporu grafického zobrazení. Díky hardwarovému 3D akcelerátoru můžeme na některých modelech zobrazovat 3D objekty, většina modelů má v sobě dokonce více procesorových jader a je schopna zobrazit HD rozlišení. [4]

FreeRTOS je realtime operační systém, který podporuje více jak 30 architektur vestavěných systémů. [5] FreeRTOS umí preemptivní multitasking, fronty, mutexy, softwarové časovače a další viz [6]. Jedná se o systém s minimálními hardwarovými nároky. Plánovač úloh zabírá pouze 236 bajtů RAM, režie jednoho vlákna stojí 64 bajtů RAM a režie každé fronty stojí 76 bajtů RAM. Při základním nastavení vyžaduje operační systém okolo 5-10KB paměti ROM v závislosti na architektuře a použitého kompilátoru. [7] FreeRTOS je distribuován pod licencí opensource, je tak možný použít pro komerční účely bez nutnosti poskytovat zdrojový kód aplikace též jako opensource.

SafeRTOS je certifikovaný realtime operační systém. Zatímco FreeRTOS byl navržen pro využití maxima výkonu, minimální režie a zdrojů, tak SafeRTOS klade důraz na bezpečnost a determinovanost na úkor zdrojů (například statické alokování paměti, jenž má definovanou dobu odezvy namísto dynamické).

NETMF for STM32 je open-source micro framework napsaný z větší části v jazyce C/C++ pro platformu STM32. Aplikace je možné psát ve Visual Studio a to v jazyku C#. Framework obaluje hardware (hardware abstract layer) a aplikace je tak přenositelná mezi více procesory. NETMF poskytuje možnost vytvářet vlákna, mutexy, vyjímky a má implementovaný garbage collector napsaný v C/C++. K běhu NETMF je potřeba hardware s minimálně 32 KB RAM a 256 KB Flash [8], což odpovídá řadě STM32F1 (Cortex-M3) a výše. Výhodou tohoto systému je jednoduchost vývoje aplikací na úkor optimalizace.

3 Vývoj na platformě STM32

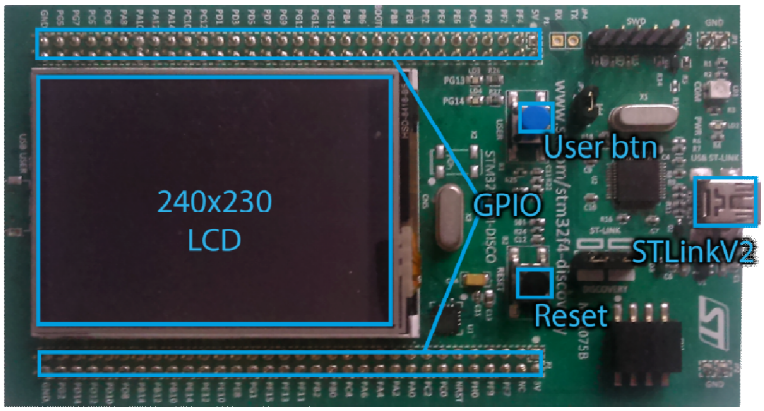
3.1 Embedded systémy

Embedded lze do češtiny přeložit jako vestavěný, jedná se tedy o vestavěné systémy, které nejsou zpravidla vidět. Mikrovlnná trouba je klasickým příkladem vestavěného systému z domácnosti nebo to může být automat pro zakoupení parkovacího lístku. Kritérií, která se kladou na embedded systémy je hned několik. Jednak pro nás může být důležitá malá velikost a nízká spotřeba v případě mobilních/wearable zařízení jako jsou různé měřiče tepu, krokoměry a další… Pokud by se jednalo o techniku do terénu tak nás bude zajímat velká robustnost či odolnost extrémním teplotám.

3.2 Možnosti a srovnání vývojových kitů F4 a F7

3.2.1 STM32F429I-Discovery

Kit F4 discovery je relativně malá deska tištěných spojů o velikosti 120x61 milimetrů osazena mnoha komunikačními sběrnicemi a moduly. Na první pohled si všimneme displeje na přední straně s úhlopříčkou 2,4 palců a rozlišením 240x320 bodů do níž je zapisováno pomocí ovladače s podporou protokolu RGB, což je standardní rozhraní pro komunikaci s LCD displeji. Součástí displeje je rezistivní dotyková vrstva s vývody pro SPI.



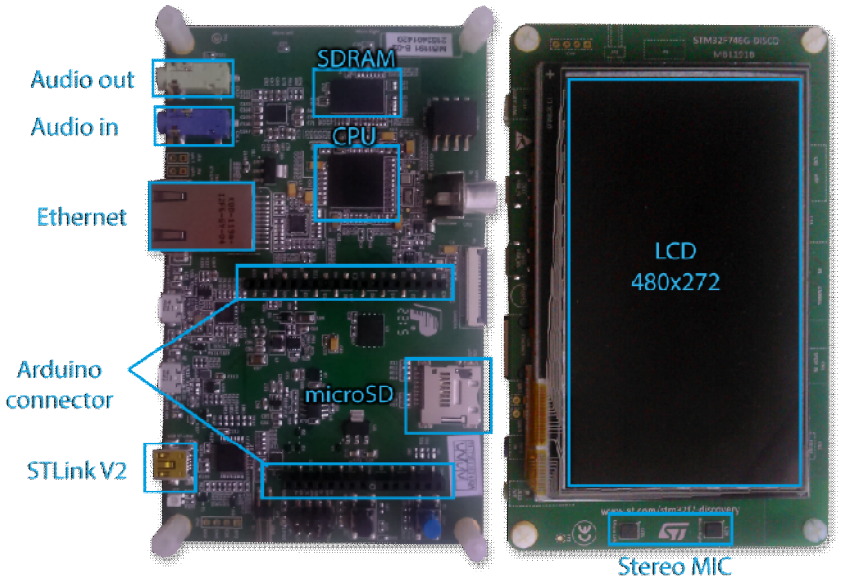
Obrázek 1 STM32F429I-Discovery

Kit obsahuje čtyři řady po třiceti dvou vstupně-výstupních pinů, z nichž většina může mít speciální funkce – například jej můžeme použít pro komunikaci s UART, I2C, SPI či CAN protokolem. Kit pohání ARM Cortex-M4 s taktovací frekvencí až 180 MHz a 256KB SRAM paměti. [9] tento procesor má jedno DMA pro obsloužení USB a druhé pro možnost komunikace po Ethernetu. Dále paměť SDRAM s automatickou obnovou paměťových buněk,

která tak rozšiřuje volatilní paměťový prostor o 64 Mbitů. [10] Důležitou součástí kitu je též programátor, který nám umožňuje skrze technologii JTAG nahrát vytvořený software do procesoru. Též jsme díky němu schopni debugovat program na úrovni zdrojového kódu a stává se tak silným ladícím nástrojem. Nakonec na desce nalezneme dvě tlačítka - z nichž jedno slouží pro rychlý reset procesoru a druhé je uživatelsky programovatelné. Dva různé USB konektory, které je možné přepnout mezi komunikací s programátorem či procesorem. Několik svítivých diod, kterými například můžeme indikovat běh programu a akcelerometr pro detekci změn pohybu přístroje.

3.2.2 STM32F746G-Discovery

Model F7 je výkonnější [Tabulka 1] než výše zmiňovaný model F4. Přední plochu tištěného spoje zabírá 4,3 palcový LCD display s technologií TFT a zobrazitelnou plochou 480x272 pixelů. Na panelu je nalepena dotyková kapacitní vrstva umožňující snazší práci s uživatelským rozhraním bez nutnosti tlačit na vrstvu. Na druhé straně je patice pouze s 21 GPIO piny rozdělenou do dvou řad, jelikož ostatní piny jsou použity pro několik dalších konektorů a modulů.



Obrázek 2 STM32F746G-Discovery

Například si můžeme všimnout audio vstupu (modrý konektor) a výstupu (zelený konektor), slotu pro SD kartu, ethernetu, tří USB portů (jeden je použit pro JTAG) a několika dalších viz Obrázek 2.

Kit pohání procesor na bázi Cortex-M7 s taktovací frekvencí až 216 MHz, 320KB paměti SRAM a L1 cache o velikosti 8KB (4KB pro data a 4KB pro instrukce). [11] Aby mohl procesor rychleji zobrazovat grafická data, obsahuje DMA2D, jenž se stará o přenos a konverzi dat z paměti do displeje, aniž by procesor tato činnost jakkoliv brzdila. Další DMA se starají o rychlou komunikaci přes Ethernet či USB. Procesor podporuje všechny komunikační protokoly jako STM32F429, navíc však přibyl například SAI (serial audio interface) pro podporu audio přenosů. [12]

Tabulka 1 Porovnání kitu F4 a F7

Název \ Hodnota	**STM32F746G-Discovery**	**STM32F429I-Discovery**
Maximální taktovací frekvence [MHz]	216	180
Velikost SRAM paměti [KB]	320	256
DMIPS/MHz	462/2,14	225/1,25
Velikost SDRAM paměti [Mbit]	128	64
Úhlopříčka displeje [“]	2,4“	4,3“
Rozlišení displeje [px]	240x320	480x272
Velikost kitu [mm]	66x119,3	80x130

Zdroje: [11] [10]

3.3 Programování procesorů STM32

Programování spočívá ve zkompilování kódu v jazyce dobře čitelném pro programátora (C/C++, Java, C#) do binárního kódu v podobě instrukcí zpracovatelných procesorem. U procesorů STM32 se binární soubor zapisuje do flash paměti procesoru skrze ST-link V2, což je obvod který zprostředkovává komunikaci mezi počítačem a procesorem jako je například již zmíněné programování, debugování [13], ale dovoluje nám nahrát i jinou verzi firmwaru samotného programátoru. ST-link V2 je součástí vývojových kitů a dovoluje nám snadno pracovat s procesorem, jelikož po připojení kitu k počítači ST-link V2 simuluje flash disk
a po nahrání binárního souboru do tohoto disku se přeprogramuje flash paměť procesoru.

3.3.1 IDE

Základní možností vývoje softwaru pro procesory STM32 je v offline vývojovým prostředím na lokálním počítači. Takové prostředí poskytuje rychlou kompilaci, různorodé nastavení a většina z nich i ladící nástroje. Pouze vývojové prostřední nám však nestačí, musíme k němu

připojit i takzvaný „toolchain“, což je soubor funkcí a nástrojů pro práci s cílovým hardwarem. A jelikož ST-link V2 není plug-and-play, musíme nainstalovat ovladače k programátoru ze stránek STMicroelectronics, které pak vývojové prostředí využívá k programování a ladění aplikace. Vývojových prostřední je hned několik, můžeme je rozdělit na freeware a komerční.

Freeware:

- **SW4STM32** – IDE od společnosti ST založené na Eclipse, má podporu ST-Link a je multiplatformní.
- **CooCox** – Zjednodušené prostředí Eclipse, podpora ladění.
- **emIDE** – Přehledné a velmi jednoduché IDE.
- **TrueSTUDIO** – od firmy atollic, zdarma verze bez omezení, placená verze navíc disponuje možností vytvářet grafy ze zaznamenaných veličin a uživatelskou podporou.

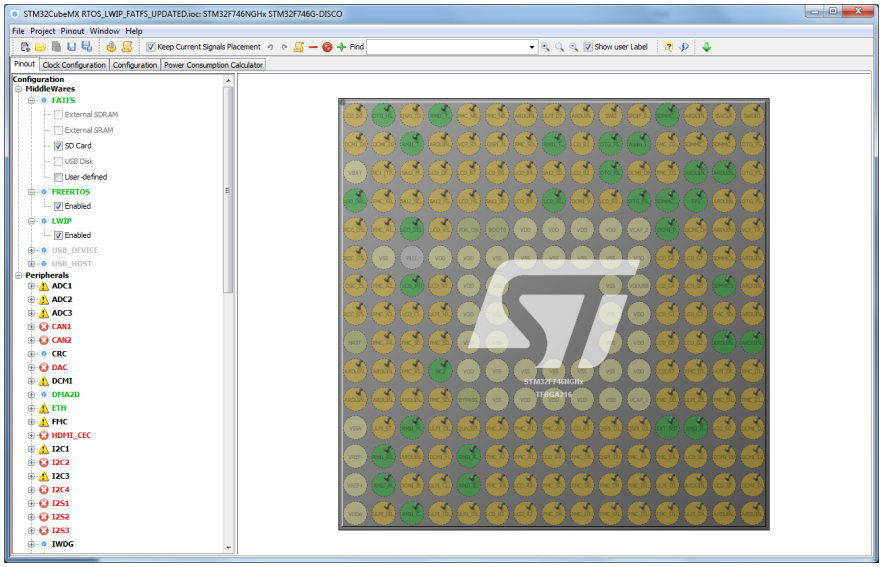
Komerční:

- **IAR** – Vysoce optimalizovaný překladač, podpora ST-Link. Trial verze buď na 30 dní zdarma anebo do velikosti kódu 32KB.
- **µVision** – IDE od společnosti KEIL, lze zdarma vyvíjet do velikosti kódu 32KB
- **CrossWorks** – Multiplaformní, podporuje celou řadu procesorů.
- **Code::Blocks + EPS** – IDE je zdarma, ale je nutno přikoupit rozšíření zvané EPS, které přidává mimo jiné možnost ladit výsledný program.

3.3.2 Konfigurace hw s CubeMX

CubeMX je software od společnosti ST pro generování inicializačních zdrojových kódů a nastavení vnitřních funkcí procesoru pro desktopové vývojové prostředí jako je například IAR, KEIL nebo CooCox. Pracovní plocha programu se po spuštění a vytvoření projektu rozděluje do čtyř záložek a zároveň hlavních funkcí programu. V první záložce můžeme nastavit funkci jednotlivým pinům procesoru jako je například analogově-digitální převodník, přerušení, vstupy, výstupy, I2C a další… Také zde můžeme aktivovat jakýkoliv middleware, pro který máme volné piny. V druhé záložce se nám nabízí nastavení hodin a děličů. Můžeme tak zde nastavit frekvenci procesoru, anebo rychlost jednotlivých periferií jako je ethernet, karta SD nebo UART. V následující záložce máme možnost blíže upravovat periferie

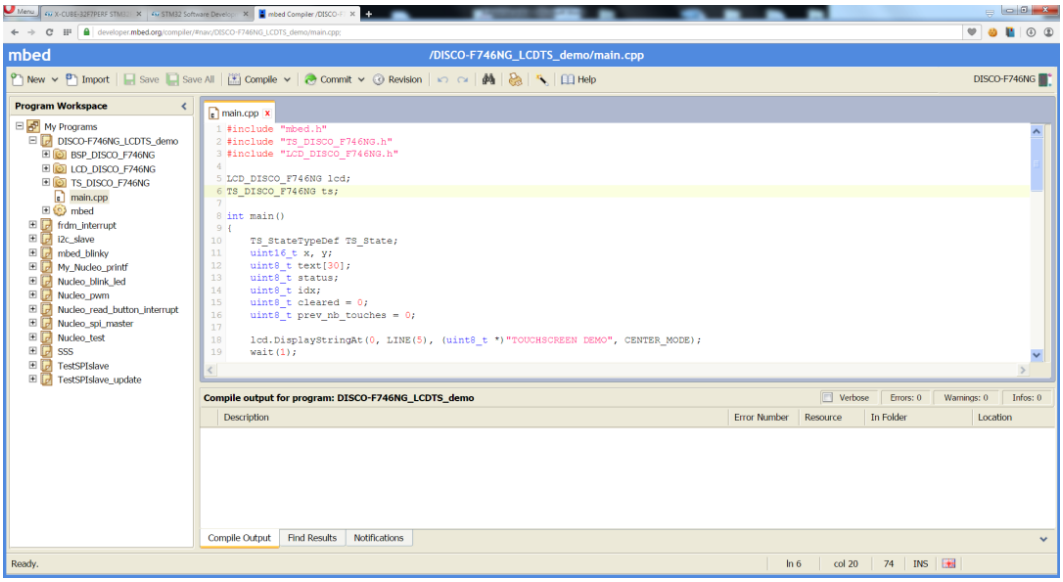
a middlewary než tomu bylo v prvním listu. Poslední list nám pomáhá spočíst energickou náročnost aplikace, což se hodí v případě vyvíjení baterií napájených (lowpower) zařízení. Po zadání využití obvodů a frekvenci procesoru za určitý čas vykreslí software graf spotřeby aplikace.



Obrázek 3 Prostředí aplikace CubeMX

3.3.3 mbed

Zajímavým nápadem je online vývojové prostředí mbed, které vyvinula společnost ST a kde po registrování a zvolení své platformy je možné napsat a zkompilovat program v prohlížeči. Výhodou tohoto řešení je jednoduchost, knihovny jsou napsané v objektovém jazyce C++ a jsou navrženy pro snadné používání. Portál skýtá též mnoho příkladů, kde je ukázáno jak dané periferie použít a je možnost si je do svého profilu naimportovat jedním stisknutím. Kompilace překladu se provádí na serveru, což pro nás znamená, že nemusíme mít nainstalovaný žádný toolchain a vývojové prostředí. Procesor se naprogramuje pouhým přesunutím vygenerovaného binárního souboru do simulovaného flash disku ST-Linku. Mbed však nemůže být použit pro vývoj větších projektů zahrnující operační systém, či middleware. Mbed neumožňuje tak rozsáhlá nastavení, jaká máme k dispozici při použití CubeMX s IDE a toolchain. Též v něm nelze přeložit vkládaný assembler tzv. „inline assembler“, což je součást mnoha rychlých knihoven, protože dopředu víme, jak bude přeložen a bývá použit v kritických částech programu.

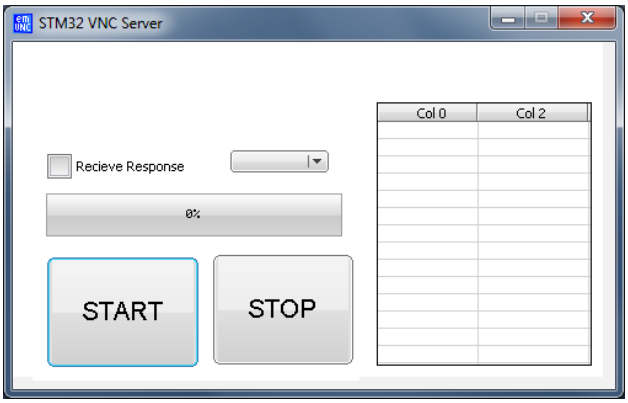


Obrázek 4 Online vývojové prostředí mbed

Portál mbed je vhodný pro tvorbu malých projektů, kde jsou od procesoru očekávány základní funkce, a není kladen důraz na vysokou efektivitu kódu. U složitějších grafických aplikací a aplikací s operačním systémem, bych mbed nevyužil.

3.3.4 emVNC

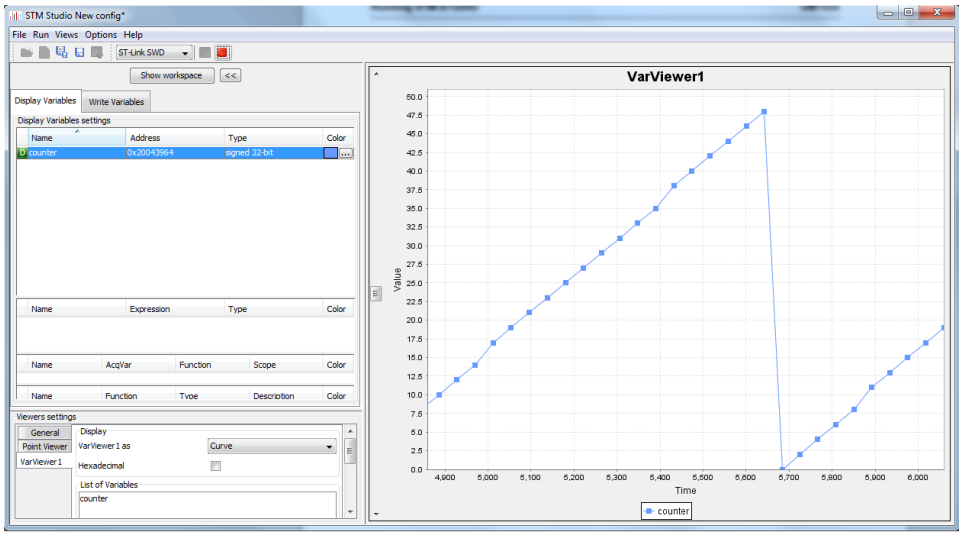
VNC slouží k posílání snímek obrazovky z jednoho zařízení na druhý, přičemž zařízení, ze kterého jsou snímky pořizovány, je server a zařízení na kterém jsou snímky zobrazeny, je klient. VNC protokol poskytuje klientovi také ovládat serverový počítač myší či klávesnicí. emVNC je program, běžící pod Windows, který zastupuje VNC klienta a je koncipován na maximální jednoduchost. Kit na druhé straně implementuje VNC server, pomocí API knihovny STemWin a díky tomu lze ovládat aplikaci na dálku z počítače.



Obrázek 5 Ukázka programu emVNC

3.3.5 STMStudio

STMStudio je samostatný, lehce ovladatelný software určený pro čtení a zápis do paměti běžícího procesoru. Pokud totiž známe, jak jsou proměnné v paměti namapované, můžeme s nimi pracovat v cílovém kitu. STMStudio je dokáže zobrazovat do grafů, popřípadě ukládat do souboru a my s nimi můžeme dále pracovat dle potřeby. Na následujícím obrázku je příklad čtení čítače a jeho zobrazení do grafu.



Obrázek 6 Zobrazení čítače v programu STMStudio

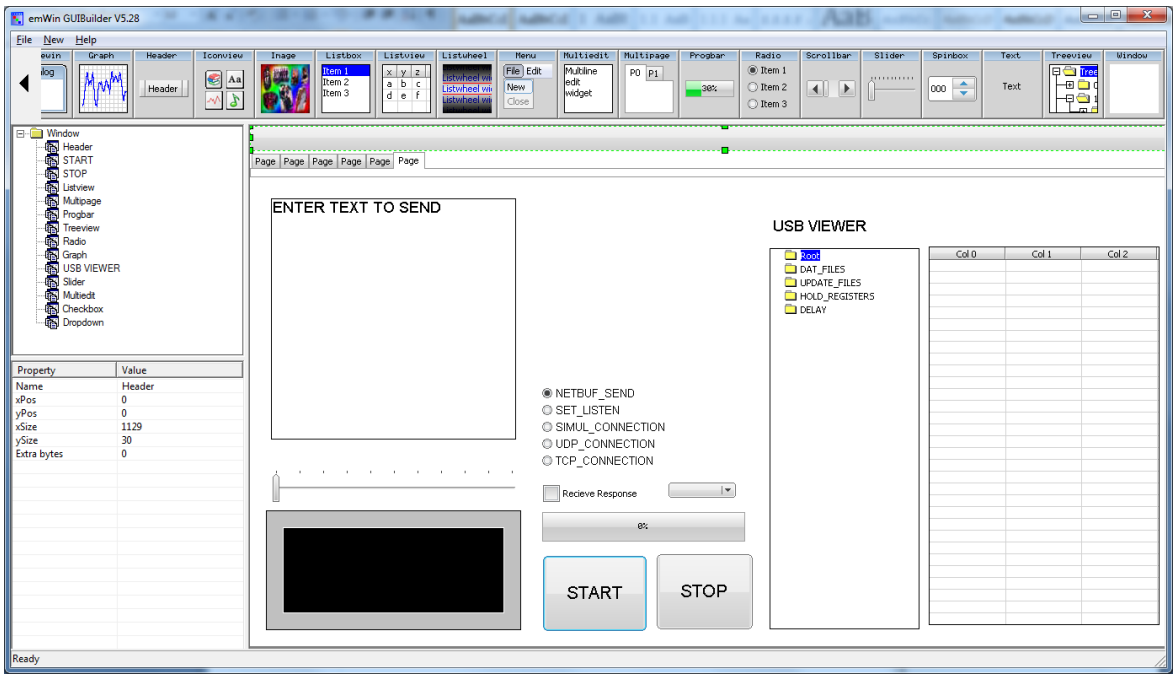
3.4 Knihovny

3.4.1 HAL Drivers

Hardware Abstract Layer je základní knihovna pro práci s procesorem. Programátor tak pomocí metod může nastavovat periferie procesoru a procesor samotný aniž by znal přesné mapování registrů v paměti. K dispozici tak máme balíky metod spravující i2c komunikaci, časovače, DMA, UART, vstupně výstupní piny a jiné… Před HAL ovladači se více používaly SPL (Standard Peripheral Library) ovladače, které jsou založeny na makrech jazyka C. Výhodou maker je, že jsou vykonávány linkerem, ještě před sestavením, takže výsledný kód je rychlý. Výsledná aplikace však není lehce přenositelná mezi více procesory a práce s SPL je složitější, než s HAL [14]. HAL metody jsou využívány softwarem CubeMX pro inicializaci, což ještě více usnadňuje práci. K HAL ovladačům je dále nadstavbou například knihovna BSP, se kterými lze například kreslit po displeji geometrické tvary, číst stav dotykové vrstvy po I2C a mnoho dalšího.

3.4.2 STemWin

STemWin je knihovna pro tvorbu grafického uživatelského rozhraní. Knihovna umožňuje vykreslovat tabulky, tlačítka, texty, posuvníky, obrázky či například jednoduché menu. STemWin poskytuje API pro kreslení základních geometrických tvarů v případě, že je potřeba napsat vlastní metodu pro překreslení libovolné komponenty. Viz manuál [15].



Obrázek 7 Ukázka programu GUIBuilder

S knihovnou jsou dodávané také podpůrné programy:

GUIBuilder – tvorba oken aplikace (viz Obrázek 7)

FontCvt – konvertor písma, který z křivek vytvoří bitmapu (v C) čitelnou pro knihovnu

BmpCvt – konvertor obrázků ve formátech bmp, gif, png do zdrojového souboru

JPEG2Movie – konvertor obrázku jpg do pohyblivé animace zapsané v C

3.4.3 LwIP

Lightweight TCP/IP je knihovna implementující celou řadu protokolů po TCP/IP viz oficiální web [16]. Prvním přístupem jak pracovat s LwIP knihovnou je skrze raw/native API. Jelikož se jedná o low level API, tak se hodí tam, kde nemáme více vlákennou aplikaci, protože nemá ošetřený přístup ke zdrojům z více vláken najednou. Na druhou stranu tu máme Netconn Api, které je vhodné pro operační systém, pracuje se s ním snáze a efektivněji, je ale pomalejší, než předchozí přístup.

3.4.4 FatFS

FatFS je opensource knihovna, implementující souborový systém FAT. K tomu, aby mohla knihovna pracovat s nižší vrstvou, potřebuje driver. Driver si můžeme napsat sami podle šablony [17], ale v případě STM32 jej lze získat zdarma od výrobce. FatFS poskytuje lowlevel operace (inicializace, čtení sektorů, zápis do sektorů), tak operace nad soubory a složkami (vytvoření složky, otevření souboru, zápis textově či binárně, zjištění volného místa na disku a mnoho dalšího). FatFS může pracovat až s deseti diskovými oddíly, musíme ale specifikovat toto číslo v konfiguraci. Je možné zapnout funkci LFN, jenž dovolí vytvářet soubory s délkou názvu až 255 znaků (opět je nutné deklarovat v konfiguračním souboru). Tuto funkci má patentovanou Microsoft, pokud by byla použita pro komerční projekt, je nutné požádat o licenci.

4 Cíl práce

- Seznámit se s kity F4 a F7 a ověřit funkce pomocí komplexní aplikace, která by pomocí TCP/IP či RS485 vyčítala data z měřících přístrojů protokolem Modbus TCP a RTU dle [18]. Tyto načtená data se dále zpracují a zobrazí na displeji. Cíl práce můžeme rozčlenit do několika dílčích úkolů.
- Nastavit všechny potřebné periferie a naimportovat knihovny nutné pro klíčové funkce aplikace.
- Navrhnout vnitřní strukturu programu a komunikaci mezi jednotlivými vlákny operačního systému. Aplikace by měla být schopna načítat nastavení z paměťové karty připojené k zařízení, proto je nutné vymyslet, jak budou konfigurační soubory vypadat.
- Naprogramovat vlákna aplikace a pomocné knihovny pro zpracování dat.
- Poté je potřeba zjistit jaké vizuální prvky lze použít a následně navrhnout grafické uživatelské rozhraní, které by bylo přehledné a poskytovalo uživateli snadnou kontrolu nad zobrazovanými daty.

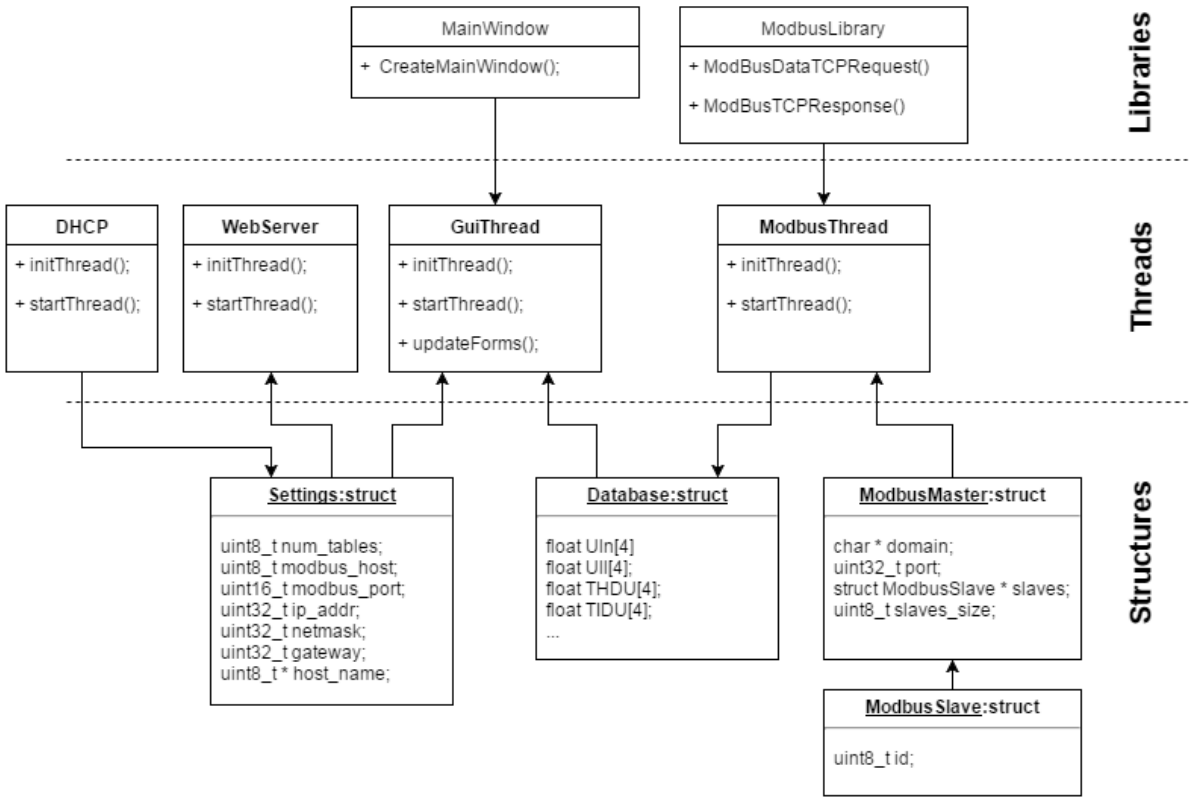
Součástí práce by mělo být také otestování funkčnosti řešení.

5 Návrh řešení

Aplikaci budu vyvíjet na kitu STM32F7 Discovery, protože má oproti verzi F4 větší displej, což je podstatné pokud chceme zobrazit větší soubor hodnot najednou. Též má ethernetový port a slot na SD kartu přímo na desce a není potřeba pořizovat žádné přídavné moduly. Operační systém jsem zvolil FreeRTOS, protože je možné jej snadno přidat do projektu již při zakládání. Vyžaduje minimální zdroje a již je vyřešená vzájemná kompatibilita. Pro práci s kartou SD využiji knihovnu FatFS, která pro účely konfigurace postačuje.

5.1 Struktura aplikace

Správná struktura aplikace je důležitá pro snadný vývoj a ladění softwaru. Rozhodl jsem se aplikaci dělit do logických celků, kde každá část má svůj jednoznačný úkol, což nám umožňuje přehlednost projektu. Struktura aplikace se skládá ze tří logických celků – knihovny, vláken a struktur jazyka C. Na Obrázek 8 Struktura aplikaceObrázek 8 můžeme vidět schéma struktury aplikace.



Obrázek 8 Struktura aplikace

5.1.1 Struktury a Mutexy

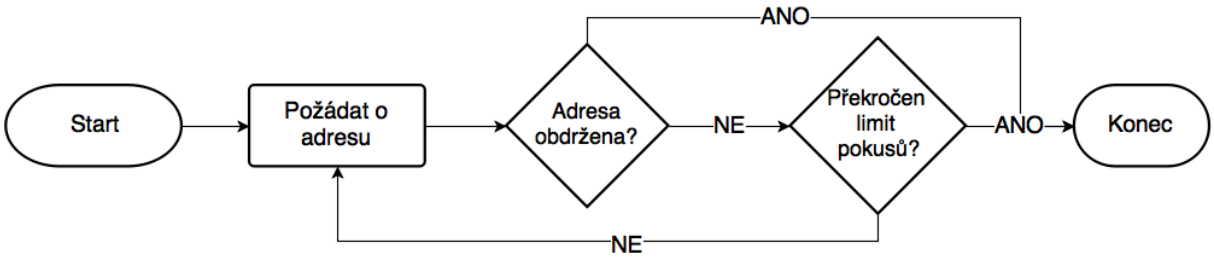
Struktura v programovacím jazyce C sdružuje několik datových typů, které mají k sobě logický vztah. Aby bylo možné k těmto strukturám přistupovat z několika vláken paraelně, vytvořil jsem nad každou strukturou mutex, který tento problém řeší.

Do struktury `Settings` jsou ukládány hodnoty z konfiguračního souboru a při inicializaci naopak vlákna z této struktury čtou a upravují tak svůj běh. Obsah je tak skoro identický s konfiguračním souborem. `Database` je struktura uchovávající konkrétního měřicího přístroje. Periodicky jsou tyto data obnovována a překreslována uživateli v tabulkách či grafech. `ModbusSlave` je struktura reprezentující koncový přístroj, ze kterého jsou čtená data. S tím souvisí struktura `ModbusMaster`, která představuje nadřazený přístroj, kterého se na data dotazuji.

5.1.2 Vlákna

Jelikož aplikace musí zvládat několik úloh najednou (komunikace, zobrazení…), musí být pro každou úlohu vymezeno vlákno/task. Preemtivní operační systém pak přiděluje časová kvanta jednotlivým úlohám podle jejich priorit. FreeRTOS v základním nastavení přiděluje úlohám systémové zdroje o velikosti 1ms. V případě, že žádná z úloh nepracuje a čeká na další cyklus, pak běží základní úloha Idle, která má nejnižší prioritu.

Vlákno `DHCPThread` řeší získávání adresy z DHCP serveru pomocí jednoduchého stavového automatu. Pokud je adresa obdržena v limitu pokusů, je nastavena, pokud byly pokusy neúspěšné, nastaví se statická IP adresa. V obou případech se výsledek zapíše do struktury `Settings`, která je popsána níže.



Obrázek 9 Stavový automat - získání adresy DHCP

`ModbusThread` je vlákno, které obsluhuje komunikaci s Modbus protokolem po TCP. Nejprve načte ze struktury `ModbusMaster` doménu aktuálně zvoleného zařízení, zjistí její

IP adresu a začne se dotazovat na data. Dotazování se řídí podle statické tabulky, která obsahuje data o jednotlivých dotazech. Konkrétně v ní najdeme identifikační číslo dotazu, číslo funkce, bázovou adresu dat a velikost požadovaných dat. Odpověď ze serveru se dále předá metodě `modbus_serve_rutine()`, která zjistí, ke kterému dotazu odpověď patří a data patřičně roztřídí do struktury `Database`.

`GUIThread` je vlákno, jenž se stará o grafický výstup aplikace. Po inicializaci grafické knihovny provádí v cyklu tři operace – přečtení stavu dotykové vrstvy, zápis dat ze struktur do grafických oken a překreslení scény. V tomto vlákně, jsou uchována identifikační čísla jednotlivých oken, aby se k nim mohlo rychle přistupovat a upravovat data.

5.1.3 Konfigurační soubory

Konfigurace je způsob, jak nastavit zařízení bez zásahu do zdrojového kódu. Konfigurační soubory jsou ukládány na SD kartu ve formátu CSV, protože se snadno upravují na jakékoliv desktopové platformě a též se snadno zpracovávají. Jako oddělovač sloupců je použit středník, protože čárka by se mohla vyskytovat v desetinném čísle. Všechny konfigurační soubory jsou umístěny přímo v kořenovém adresáři karty a dělíme je na globální konfiguraci a konfiguraci modbus zařízení (Modbus Master a Modbus Slave).

5.1.4 Globální konfigurace

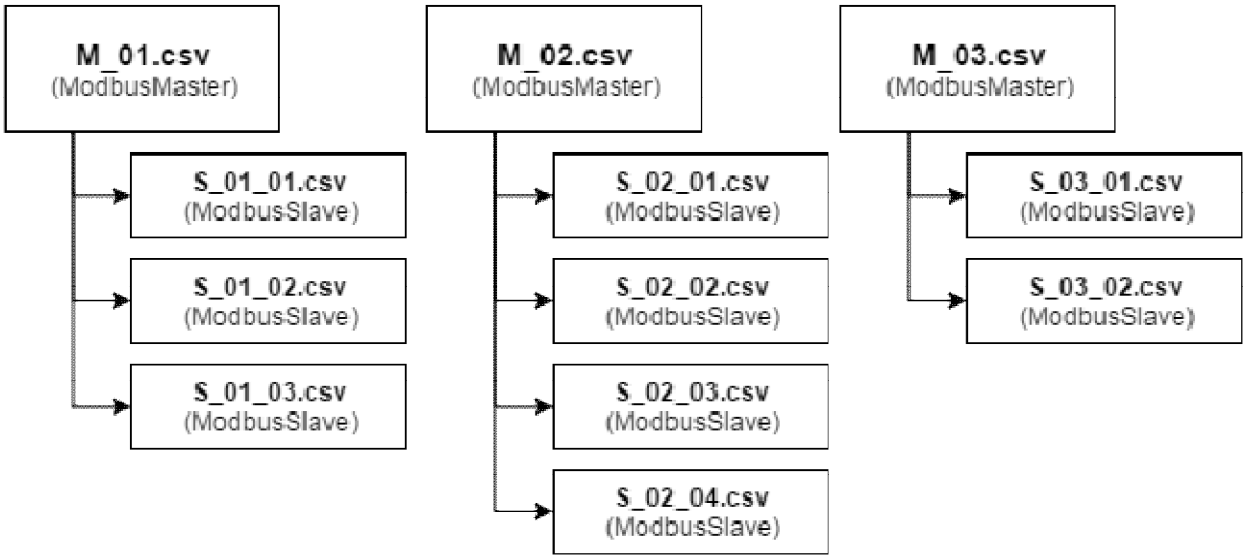
Souhrnné nastavení je obsaženo v souboru Global.csv, povoluje se v něm DHCP, webserver a upozornění emailem. Také se zde nastavují parametry k jednotlivým funkcím jako je statická IP adresa, maska podsítě, adresa mailserveru a nebo přístupové údaje k emailu. Aby bylo možné parametrizovat rychlost dotazování, je v konfiguraci konstanta delay, která určuje dobu mezi sadou dotazů (mezi načtením všech registrů z měřícího přístroje). Všechny konstanty jsou vypsány níže v Tabulka 2. Při startu aplikace se inicializuje periferie karty SD, poté se načte konfigurace, zapíše do vnitřní struktury aplikace, zamkne se pomocí mutexů a teprve potom se spouští všechny vlákna aplikace, které si sami zažádají o mutex a načtou příslušné hodnoty z těch struktur, které jsou pro ně relevantní. V Tabulka 2 můžeme vidět nastavení hodnot v souboru Global.csv.

Tabulka 2 Konstanty v konfiguraci Global.csv

Název	Hodnota
DHCP_ENABLE	1
STATIC_IP	10.0.0.40
MASK	0.0.0.255
GATE	0.0.0.0
WEBSERVER_ENABLE	1
MAIL_ENABLE	1
MAILSERVER	SNMP.SEZNAM.CZ
MAIL_USER	STANDA
MAIL_PW	1234
MAIL_ALERT_MIN_LEVEL	2
REQUEST_DELAY	300

5.1.5 Konfigurace modbus zařízeních

Abychom mohli určit, ze kterých měřících zařízení bude aplikace číst data, musíme vytvořit konfigurační soubory modbus zařízeních. Navrhnul jsem strukturu souborů, která poskytuje vazbu 1:N. Každé Master zařízení je definováno souborem M_<Master_ID>.csv, kde <Master__ID> je pořadí master zařízení číslované od jedné. Slave zařízení má následující pattern S_<Master_ID>_<Slave_ID>.csv, <kde Master_ID> je pořadí nadřazeného zařízení a <Slave_ID> je pořadí slave zařízení číslované od jedné. Příklad takové struktury můžeme vidět na Obrázek 10. Výhodou tohoto řešení je snadná implementace.



Obrázek 10 Konfigurační struktura modbus zařízeních

6 Realizace řešení

6.1 Generování projektu

Pro vygenerování projektu a inicializaci základních funkcí jsem využil program CubeMX. Po vybrání cílového procesoru (STM32F746NGHx) jsem nastavil periferii ethernetu, SD karty a UARTu. Pro UART jsem zvolil piny PF7 (TX) a PF6 (RX), které jsou součástí arduino konektoru viz Obrázek 2. UART je plnoduplexní, ale RS485 poloduplexní protokol, proto bylo potřeba zvolit další pin, který by transceiveru sděloval směr komunikace, pro tento účel jsem vybral pin PF8. Aktivoval jsem LwIP, FreeRTOS a FatFS middleware. A nastavil konstanty podle tabulky níže.

Tabulka 3 Konstanty projektu

Název	Hodnota
UART7_BAUD_RATE	1920
UART7_WORD_LENGTH	8
UART7_STOP_BITS	1
UART7_PARITY	Even
USE_PREEMTION	Enabled
LWIP_TCP	Enabled
LWIP_DNS	Enabled
LWIP_NETCONN	Enabled
LWIP_DHCP	Enabled
TOTAL_HEAP_SIZE	30720 Bytes

6.2 Knihovna pro modbus

Vytvořil jsem knihovnu pro generování modbus dotazu a zpracování odpovědi do vlastní struktury. Metody jsou ve verzi pro modbus TCP i RTU. Knihovna využívá datové typy a metody platformy STM32 jako je například uint8_t a alokování paměti pomocí metody `pvPortMalloc()`, čímž není přenositelná na jiné platformy bez drobných úprav.

`ModBusDataRTURequest()` generuje dotaz pro modbus RTU. Argumentem do ní musíme vložit ID slave zařízení, číslo funkce, adresu a počet buněk, které chceme od dané adresy přečíst. Výstupem funkce je pointer na pole bajtů o velikosti konstanty `MODBUS_RTU_REQUEST_LENGTH`, které tvoří modbus dotaz s obsaženou kontrolní sumou CRC16.

ModBusDataTCPRequest() generuje dotaz pro modbus TCP. Na rozdíl od předchozí metody je zapotřebí uvést ještě transfer ID, kde si můžeme definovat, o který konkrétní dotaz se jedná. V aplikaci jsem si určil transfer ID jako index mapovacího vektoru, který odpovídá dotazu, díky tomu vím přesně jaká data v odpovědi očekávat. Výstupem funkce je pointer na pole bajtů o velikosti konstanty `MODBUS_TCP_REQUEST_LENGTH`. O kontrolní součet se v tomto případě stará fyzická vrstva TCP/IP, takže není obsažen v dotazu.

ModBusRTUResponse() zpracovává odpověď od slave zařízení z modbus RTU. Vstupem metody je pointer na pole dat a velikost dat. Po zpracování odpovědi metoda vrací pointer na strukturu `MB_Response`.

ModBusTCPResponse() metoda zpracovávající odpověď z modbus TCP. Od předešlé se liší pouze tím, že ve výstupní struktuře bude vyplněné transfer ID.

MB_Response součástí knihovny je definice struktury modbus odpovědi. V ní je obsaženo ID slave zařízení, ze kterého odpověď přišla, transfer ID pokud se jedná o modbus TCP, funkce, počet vrácených buněk a vrácená data.

6.3 Knihovna pro práci s csv

Po snaze ukládat data a nastavení do textového souboru, kde byly hodnoty odděleny znakem pro nový řádek, se ukázalo být výhodné jej raději uložit do formátu csv. Protože software s tabulkovým procesorem má dnes v počítači skoro každý, data jsou přehledná a kdykoliv chceme, můžeme si z nich jednoduše například vygenerovat graf. Též je tento formát relativně snadný na implementaci. Aplikace používá pro nový řádek formát windows, tedy „CR+LF“.

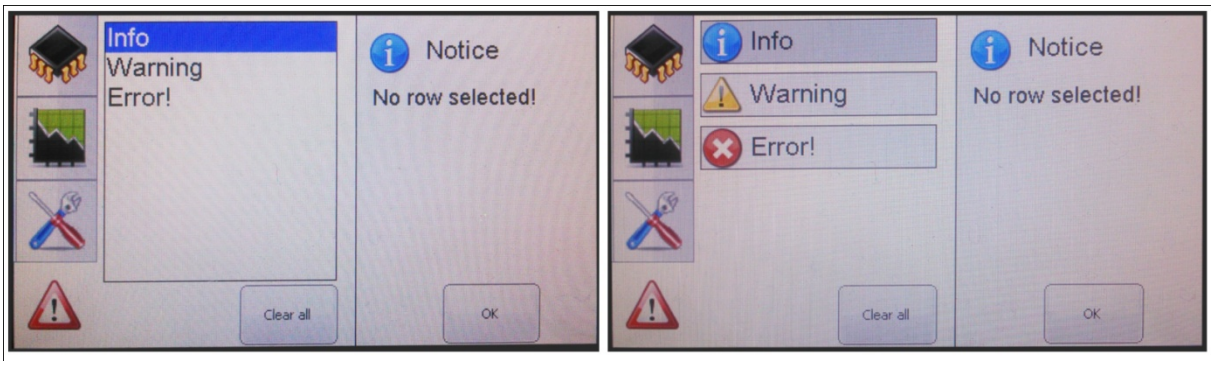
csv_to_table() metoda určená pro parsování souboru csv zadaného parametrem (pole bajtů) do struktury Table. Metoda rozpoznává středník jako oddělovací znak mezi sloupci.

table_to_csv() metoda určená k opačnému procesu, tedy z tabulky vložené argumentem, vrátí pole bajtů ve formátu csv, které lze snadno uložit na kartu SD.

Table je struktura knihovny, která reprezentuje tabulku hodnot. Obsahuje informaci o počtu řádků, sloupců a je v ní uložena matice stringů (respektive polí charů).

6.4 Tvorba vizualizace

Grafickou podobu aplikace jsem nejprve načrtnul a poté navrhnul v programu GUIBuilder, jenž vygeneroval zdrojové soubory oken s vnitřní strukturou a inicializovanými komponenty (widgety). GUIBuilder nemá tolik možností, kolik jich knihovna STemWin poskytuje, například v něm nelze nastavit orientaci `MultiPage`. Proto bylo nutné inicializaci některých komponent dopsat ručně.



Obrázek 11 Ukázka překreslovací funkce

Aby byla aplikace uživatelsky přívětivá, napsal jsem nové překreslovací funkce některým komponentám a nastavil jim je pomocí metody `<Widget>_SetOwnerDraw()`. Na obrázku Obrázek 11 můžeme vidět výsledek vlastní metody pro překreslení seznamu.

6.5 Řízení komunikace

Komunikace je klíčovou funkcí aplikace, proto je nutné rozmyslet, jakým způsobem bude probíhat. Jelikož otevření socketu a následný „handshake“ mezi dvěma zařízeními stojí mnoho času, uvažoval jsem následující řešení:

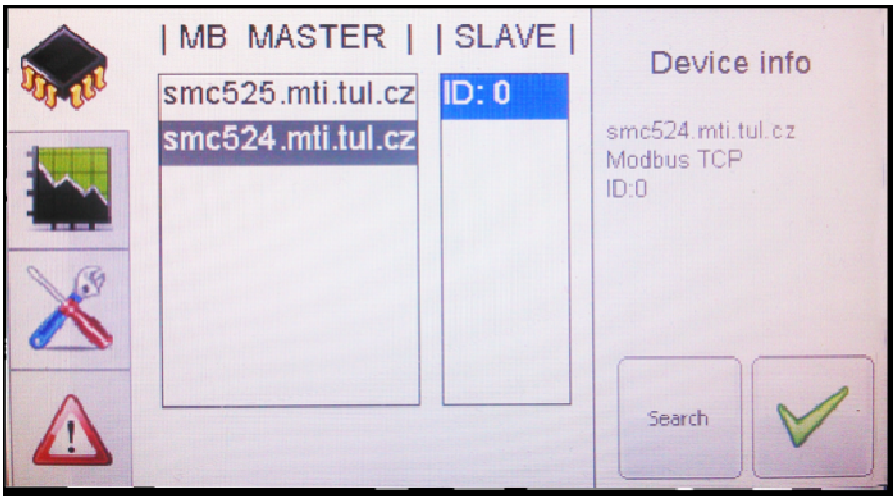
- Otevřít socket při každém obnovení dat a po načtení dat opět spojení uzavřít.
- Otevřít socket pouze na začátku a uzavřít až při změně serveru uživatelem.
- Otevřít několik socketů pro několik zařízení současně a načítat data paralelně.

První možnost je nejjednodušší na implementaci a lze využít tam, kde nám nezáleží na rychlé obnově dat. Třetí možnost využívá mnoho zdrojů a je efektivní pokud chceme rychle obnovovat data z několika přístrojů najednou. Rozhodl jsem pro druhou možnost, protože zobrazuji data pouze o jednom zařízení, které si uživatel vybere, a chtěl bych uživateli

poskytnout možnost nastavit rychlou obnovovací frekvenci. Nevýhodou zvoleného řešení je, že pokud bychom chtěli obnovovat data jednou za větší časový úsek (30 minut, hodina) tak blokujeme cílovému zařízení socket a většinu času jej nevyužíváme. V případě takto dlouhých intervalů by se vyplatil první přístup. Pokud bych zamýšlel vytvořit aplikaci i pro tak dlouhé intervaly, zkombinoval bych první a druhý přístup a podmínil je podmínkou. Pokud by pak perioda překročila danou konstantu (například 3 minuty) aplikace při zvolila první metodu, pokud by ale byla perioda kratší, aplikace by socket nechala otevřený, dle druhé metody.

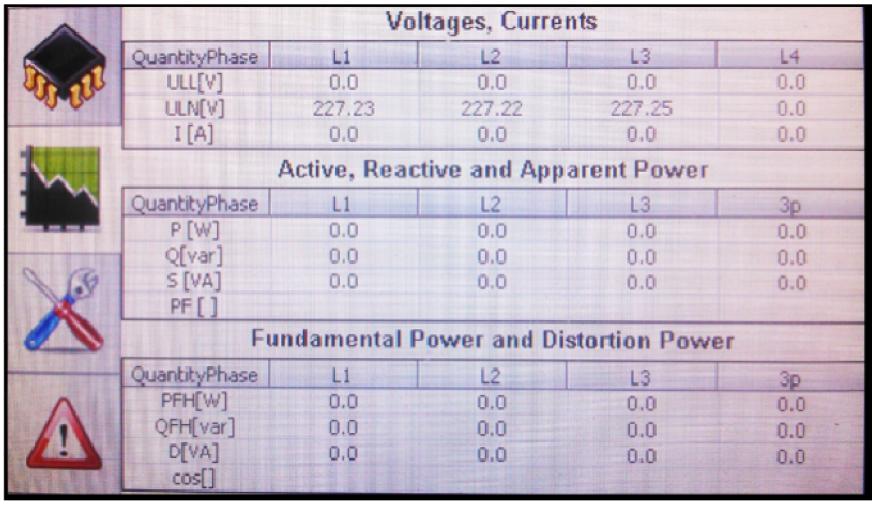
6.6 Shrnutí

Aplikace si umí získat adresu z DHCP serveru, pokud server neodpovídá, nastaví adresu statickou. V aplikaci běží webserver ukazující základní informace. Aplikace vyčítá data z aktuálně zvoleného zařízení, buď přes modbus TCP nebo RTU. V případě kritického stavu aplikace odešle email o aktuálním stavu na přednastavený email. Výše uvedené funkce jsou konfigurovatelné pomocí souborů na SD kartě.



Obrázek 12 Výběr cílového zařízení

Komunikace probíhá s právě zvoleným zařízením, zavírá a otevírá se nový socket právě při aktivního zařízení. Na Obrázek 12 můžeme vidět výběr aktuálního zařízení z listu vygenerovaného pomocí konfiguračních souborů z SD karty. Po odškrtnutí výběru se aplikace odpojí od aktuálního zařízení a připojí se k zařízení právě vybranému. Okamžitě po připojení se začne dotazovat. Tlačítko „Search“ načítá nastavení z SD karty a aktualizuje list zařízení v grafickým uživatelským rozhraní.



Obrázek 13 Zobrazení dat do tabulek

Na Obrázek 13 lze vidět zobrazení dat ze struktury `Database` do GUI aplikace. Bylo zapotřebí napsat metodu, jenž z datového typu float vytvoří pointer na pole charů, které by reprezentovali textovou podobu desetinného čísla, protože výchozí metoda `sprintf()` nevracela správný výstup.

7 Vyhodnocení řešení

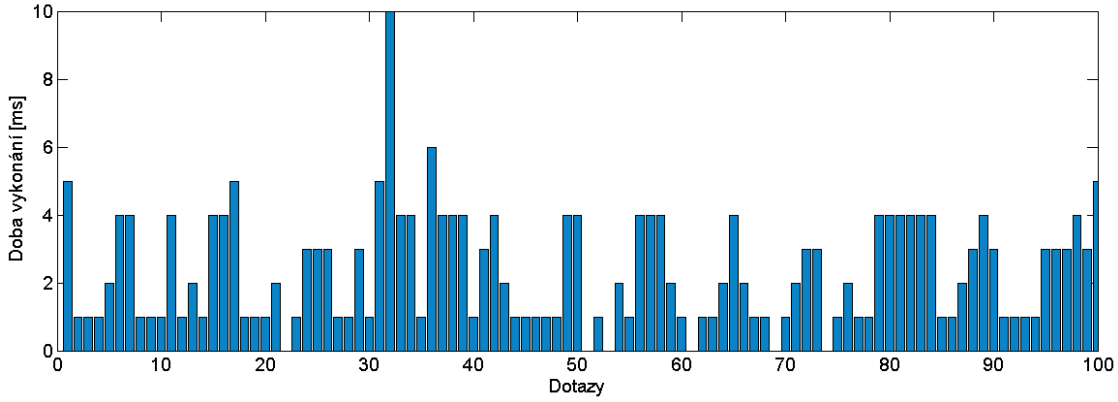
Měření probíhala v operačním systému FreeRTOS, kde jsem pro získání počtu taktů od startu systému využil metodu `osKernelSysTick()`. Metoda vrací stav čítače operačního systému s frekvencí 1KHz, což znamená, že jednotky odpovídají milisekundám. Pro získání naměřených dat, jsem využil zápisu na SD kartu po skončení měření, takže zápis neovlivnil výsledky měření.

7.1 Měření latence webserveru

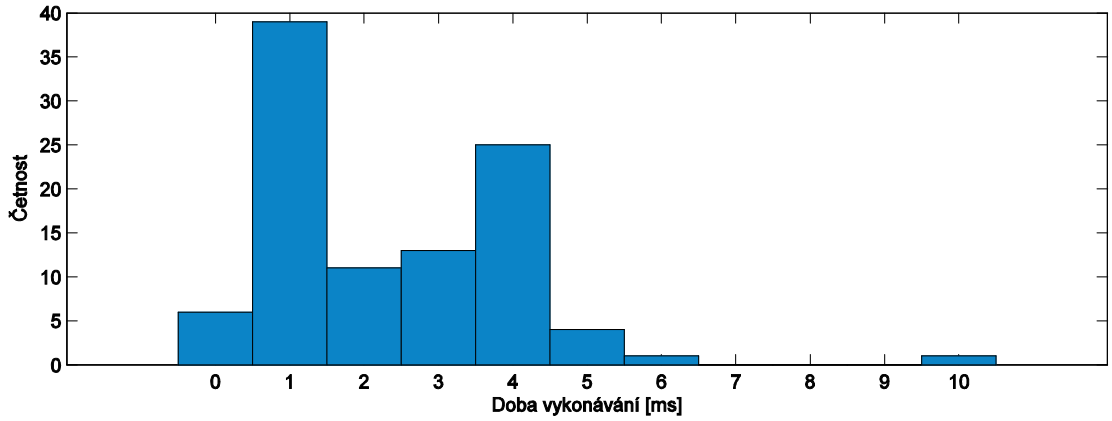
Měření jsem prováděl ze strany serveru (doba od otevření socketu až po jeho uzavření) a ze strany klienta (doba odezvy serveru). Měření probíhalo v lokální síti za pomoci počítače (klienta), switche a aplikace (serveru). Na straně klienta jsem spustil následující skript v příkazové řádce:

```
%time% > Start.txt
for /l %x in (1, 1, 100) do (
wget http://10.0.0.3/index.html
)
%time% > End.txt
```

Jak si můžeme všimnout, skript nám nejprve uloží čas spuštění do souboru start.txt, poté provede sto dotazů na webserver a nakonec uloží čas ukončení do souboru end.txt.



Obrázek 14 Graf doby vykonávání odpovědi serveru

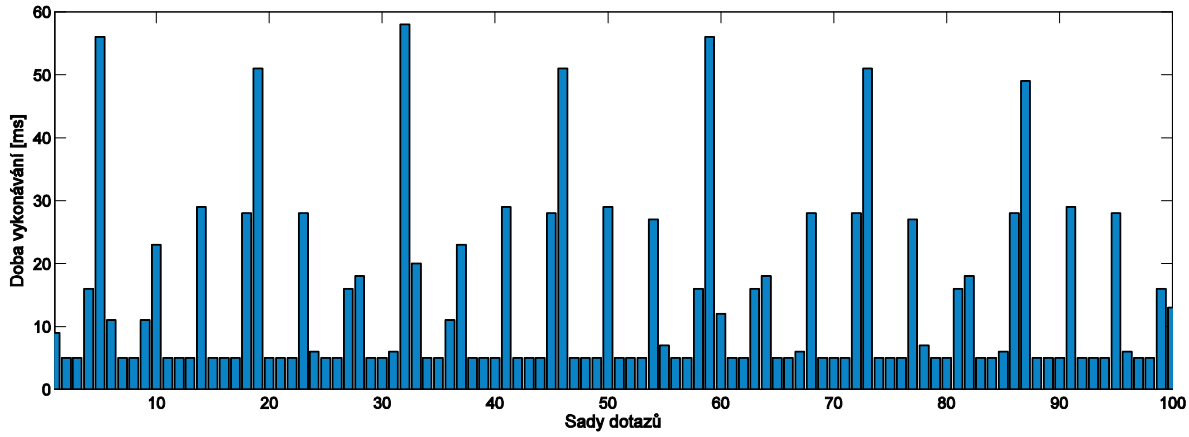


Obrázek 15 Histogram četností doby vykonávání dotazů

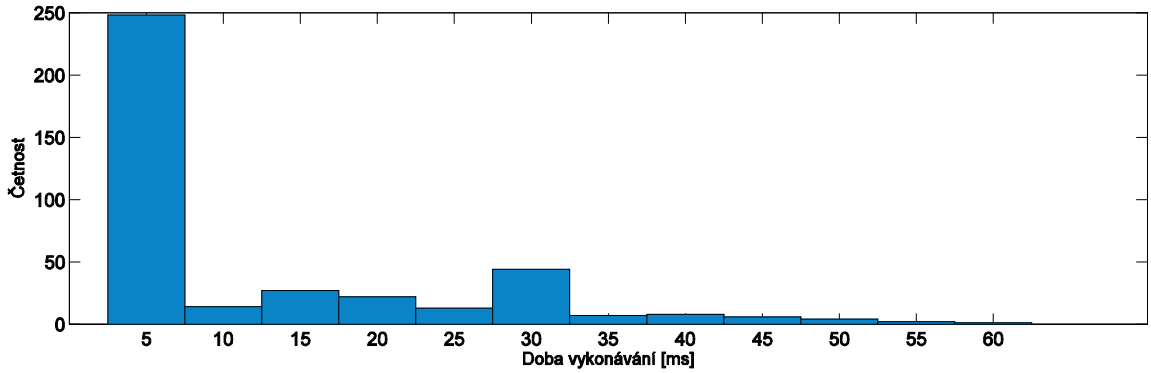
Zpracování dotazu a odeslání odpovědi o velikosti 1KB na straně serveru trvalo v rozmezí od 1ms do 10ms, zatímco odezva serveru z pohledu klienta trvala v průměru 45,4ms.

7.2 Měření rychlosti modbus TCP

Měření rychlosti dotazů modbus TCP probíhalo v lokální síti, vytvořil jsem konfigurační soubor pro měřicí přístroj a nastavil v globální konfiguraci dobu čekání mezi dotazy na nulu. Tím jsem dosáhl nepřetržitého provozu a měřil jsem počet dotazů za vteřinu a dobu jedné sady dotazů (včetně parsování dat do vnitřní struktury programu). Sada dotazů se skládá celkem z jedenácti dotazů pro konkrétní registry. Celkový počet dotázaných registrů v jedné sadě je 522 (1044 bajtů). Celkem proběhlo 4400 dotazů za 5418ms, což dává v průměru 812 dotazů za vteřinu.



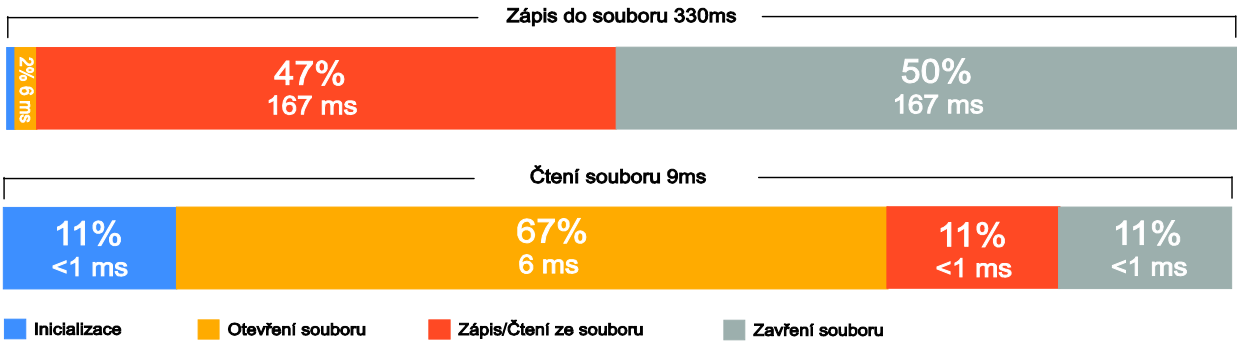
Obrázek 16 Výřez 1-100 dob vykonávání jedné sady



Obrázek 17 Histogram doby vykonávání sady dotazů

7.3 Měření práce s SD kartou

Měřil jsem dobu potřebnou pro práci s SD kartou. Na Obrázek 18 můžeme vidět procentuální rozdělení doby zápisu a čtení z SD karty. Velikost souboru jsem zvolil 1KB dat, staticky uložených, do paměti procesoru. Na Obrázek 18 můžeme vidět rozdělení dob jednotlivých částí práce s kartou. Modře označena je doba potřebná pro inicializaci, žlutá představuje otevření souboru, oranžová je doba zápisu (horní část) nebo doba čtení (spodní část) a nakonec uzavření souboru je znázorněno šedou barvou.



Obrázek 18 Rozdělení doby zápisu/čtení do/z SD karty

Závěr

Cílem práce bylo vytvořit aplikaci s grafickým uživatelským rozhraním, jenž by přes protokol modbus TCP anebo RTU získávala data a zobrazovala je na displeji. Nejprve bylo potřeba navrhnout strukturu aplikace. Každá z hlavních funkcionalit má své vlákno a prostředky, které vlákna mezi sebou sdílí, mají mutex. Aby byla aplikace konfigurovatelná, navrhnula se struktura konfiguračních souborů ukládaných na kartu SD ve formátu CSV. Tvorba samotné aplikace začala vygenerováním projektu pomocí programu CubeMX. Byla navržena grafická podoba aplikace v programu GUIBuilder a následně upravena a odladěna na cílovém hardwaru. Byla vytvořena knihovna pro vyčítání dat z protokolu modbus TCP a RTU, jenž umožňuje generovat dotaz a zpracovávat odpověď. Pro archivaci a načtení dat byla napsána knihovna pracující se soubory CSV.

Výsledkem práce je aplikace zobrazující data z aktuálně zvoleného měřicího přístroje, které získává pomocí protokolu modbus. Aplikace si umí získat IP adresu z DHCP serveru, pokud není dostupný, nastaví si adresu statickou. V aplikaci je implementovaný webserver, poskytující základní informace po připojení se internetovým prohlížečem. V případě kritické situace aplikace odešle email. Virtuální zařízení v aplikaci i funkcionality aplikace jsou konfigurovatelné. Cíl práce byl tedy splněn.

Jednou z výhod mého řešení je modularita celé aplikace. Pokud je potřeba připojit k aplikaci novou funkcionalitu, vytvoří se soubor s vláknem, nainicializuje se a úloha bude při startu aplikace běžet. Pokud je potřeba načítat z měřících přístrojů více dat, stačí přidat řádek do tabulky mapovacích vektorů a modbus vlákno již bude dotazovat i nové registry. Jelikož jsem dodržel strukturu projektu v souladu s externími knihovnami, je též jednoduché aktualizovat na nové verze. Nevýhodou zvoleného řešení je, že aplikace komunikuje pouze s jedním měřícím zařízením, dokud uživatel nepřepne aktivní zařízení. Pokud bych chtěl zobrazovat veličiny z několika zařízení najednou, tak by to s tímto řešením nebylo možné bez úprav algoritmu dotazování.

Myslím, že platforma STM32 je perspektivní a má pro embedded aplikace smysl a chtěl bych ve vývoji i nadále pokračovat. Aplikaci by bylo možné rozšířit o dynamicky vytvářené okna, podle konfigurovatelných mapovacích vektorů. Též by bylo možné aplikaci vybavit tzv. lokátorem, který zjistí dostupné modbus zařízení v síti a uloží si je do paměti a SD karty, pro příští spuštění.

Seznam tabulek

Seznam obrázků

Seznam použité literatury

1. **STMicroelectronics.** STM32 32-bit ARM Cortex MCUs - STMicroelectronics. [Online] [Cited: Duben 3, 2016.] http://www2.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus.html?querycriteria=productId=SC1169.

2. **Atmel Corporation.** *Atmel SMART ARM Processor Based MCUs.* [Online] [Citace: 5. Únor 2016.] http://www.atmel.com/products/microcontrollers/arm/default.aspx.

3. **Atmel.** Atmel Studio. *Atmel Corporation.* [Online] [Citace: 4. Únor 2016.] http://www.atmel.com/tools/atmelstudio.aspx.

4. **NXP.** i.MX 6 Series Comparison Table. *NXP Semiconductors.* [Online] 2016. [Citace: 25. Únor 2016.] http://cache.nxp.com/files/32bit/doc/brochure/FLYRIMXPRDCMPR.pdf.

5. **FreeRTOS.** *Microcontrollers and compiler tool chains supported by FreeRTOS.* [Online] [Citace: 8. Březen 2016.] http://www.freertos.org/RTOS_ports.html.

6. —. A Free real time operating system (RTOS) for small embedded systems - list of RTOS features. *FreeRTOS.* [Online] [Citace: 8. Březen 2016.] http://www.freertos.org/FreeRTOS_Features.html.

7. —. FreeRTOS FAQ relating to FreeRTOS memory management and usage. FreeRTOS is an Open Source RTOS Kernel for small embedded systems. *FreeRTOS.* [Online] [Citace: 8. Březen 2016.] http://www.freertos.org/FAQMem.html.

8. **Heeb, Cuno Pfister & Beat.** NETMF for STM32. *Oberon microsystems AG.* [Online] [Citace: 20. Duben 2016.] http://www.mountaineer.org/app/download/6069093475/NETMF+for+STM32+-+Tour+d%27Horizon+V20140122.pdf.

9. **STMicroelectronics.** 32F429IDISCOVERY - STMicroelectronics. *STMicroelectronics.* [Online] [Citace: 2016. Února 2.] http://www2.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/32f429idiscovery.html.

10. —. Discovery kit with STM32F429ZI MCU. *STMicroelectronics.* [Online] [Citace: 5. Únor 2016.] http://www2.st.com/content/ccc/resource/technical/document/user_manual/6b/25/05/23/a9/45/4d/6a/DM00093903.pdf/files/DM00093903.pdf/jcr:content/translations/en.DM00093903.pdf.

11. —. STM32F746NG - STMicroelectronics. *STMicroelectronics.* [Online] [Citace: 2. Únor 2016.] http://www2.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f7-series/stm32f7x6/stm32f746ng.html.

12. —. Discovery kit for STM32F7 Series with STM32F746NG MCU. *STMicroelectronics.* [Online] [Citace: 2. Únor 2016.] http://www2.st.com/content/ccc/resource/technical/document/user_manual/f0/14/c1/b9/95/6d/40/4d/DM00190424.pdf/files/DM00190424.pdf/jcr:content/translations/en.DM00190424.pdf.

13. —. ST-LINK/V2 - STMicroelectronics. *STMicroelectronics.* [Online] [Citace: 5. Březen 2016.] http://www2.st.com/content/st_com/en/products/development-tools/hardware-development-tools/development-tool-hardware-for-mcus/debug-hardware-for-mcus/debug-hardware-for-stm32-mcus/st-link-v2.html.

14. —. STM32 embedded software. *STMicroelectronics.* [Online] [Citace: 6. Březen 2016.] http://www.st.com/st-web-ui/static/active/cn/resource/sales_and_marketing/presentation/product_presentation/stm32_embedded_software_offering.pdf.

15. —. Getting started with STemWin Library. *STMicroelectronics.* [Online] [Citace: 6. Březen 2016.] http://www2.st.com/content/ccc/resource/technical/document/application_note/54/c9/95/42/8c/0b/43/69/DM00089670.pdf/files/DM00089670.pdf/jcr:content/translations/en.DM00089670.pdf.

16. lwIP - A Lightweight TCP/IP stack - Summary [Savannah]. *lwIP - A Lightweight TCP/IP stack.* [Online] [Citace: 20. Březen 2016.] http://savannah.nongnu.org/projects/lwip/.

17. **STMicroelectronics.** Developing Applications on STM32Cube with FatFs. *STMicroelectronics.* [Online] [Citace: 1. Duben 2016.] http://www.st.com/st-web-ui/static/active/jp/resource/technical/document/user_manual/DM00105259.pdf.

18. MODBUS APPLICATION PROTOCOL SPECIFICATION. *Modbus Organization.* [Online] [Citace: 20. Březen 2016.] http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.

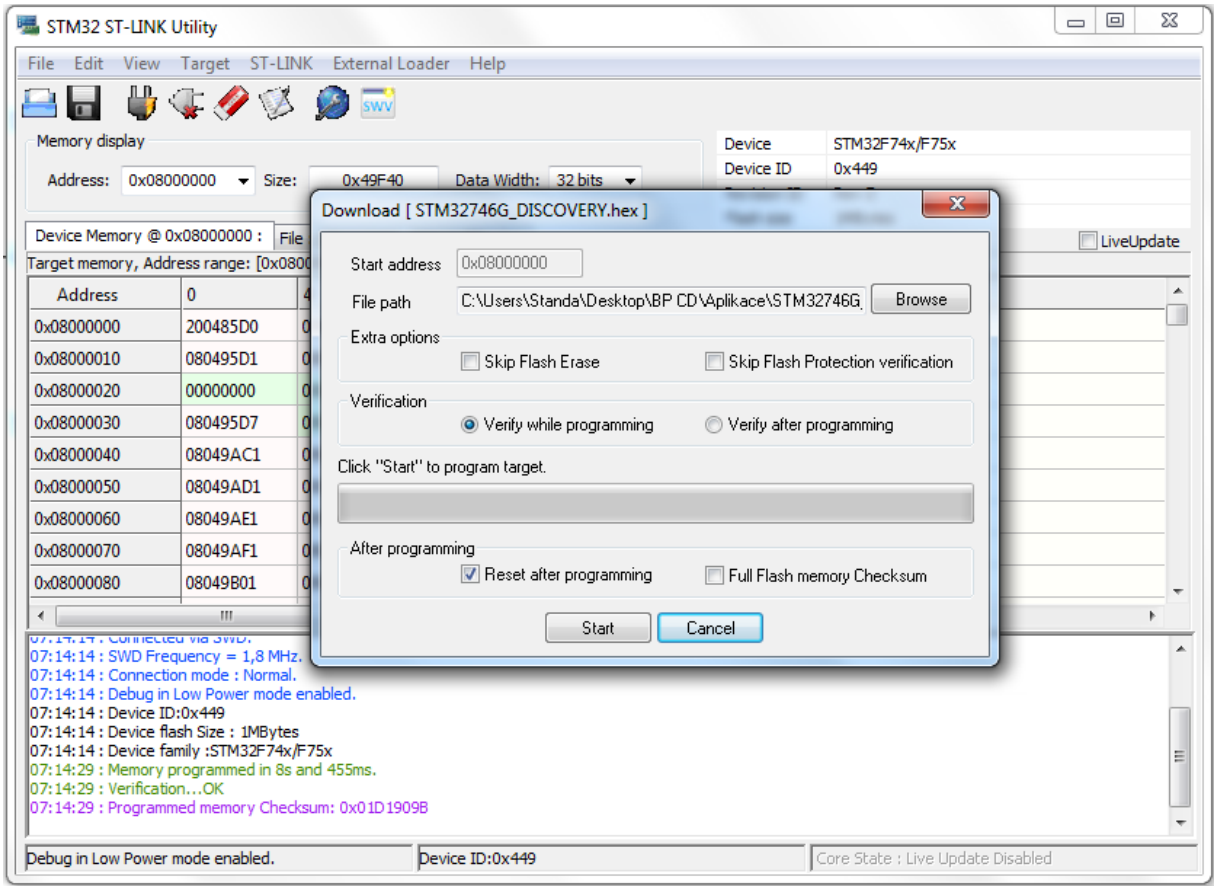
Přílohy

Obsah přiloženého CD

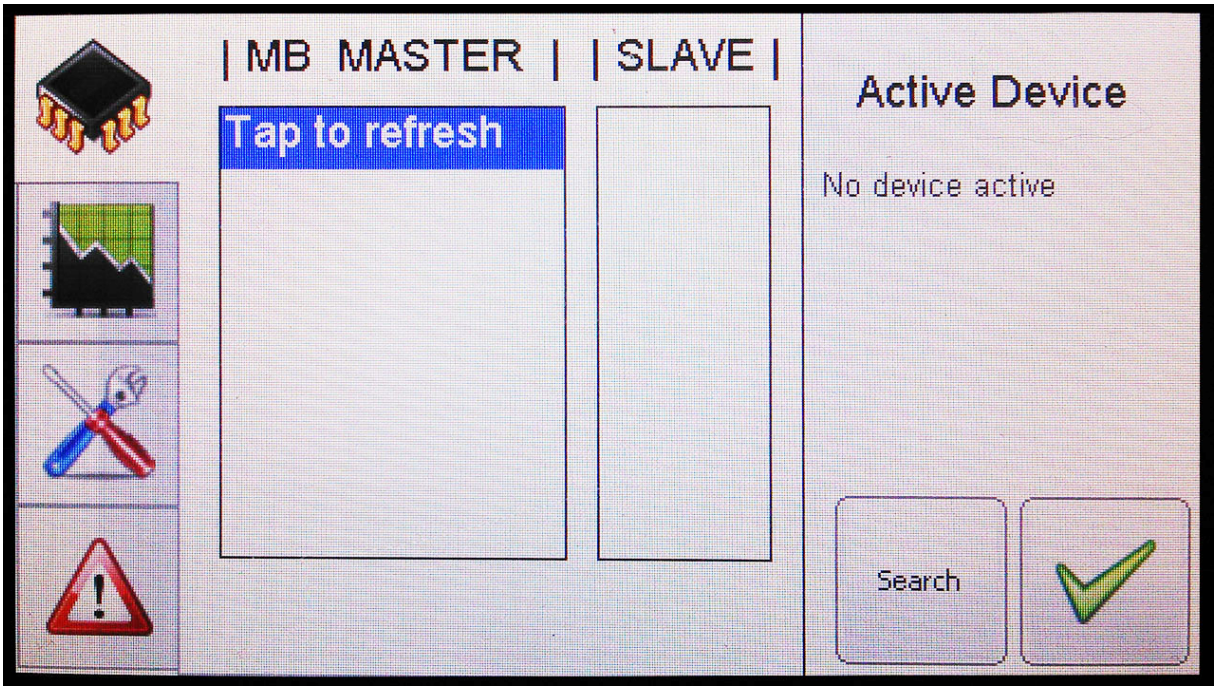
- Bakalářská_práce.pdf – Tato práce ve formátu PDF
- Bakalářská_práce.docx – Tato práce ve formátu Word 2007
- Aplikace/
    - STM32746G_DISCOVERY.hex – Výsledná aplikace
- Data/
    - Modbus_tcp.txt – Doby vykonávání sady dotazů v měření modbus tcp ve formátu [start-stop] v zápisu pole pro Matlab
    - wgetTimeDiff.txt – Doby vykonávání dotazů na webserver v „syrové“ podobě.
    - wgetTimeDiff_matlab.txt – Doby vykonávání dotazů na webserver v poli pro Matlab
    - Start.txt – Doba spuštění skriptu pro testování webserveru (klient)
    - End.txt – Doba ukončení skriptu pro testování webserveru (klient)
    - 1KB.txt – Testovací soubor o velikosti 1KB
- SDCARD/ – Obsah karty SD pro konfiguraci aplikace
    - GLOBAL.csv – Globální konfigurace
    - M_01.csv – Konfigurační soubor Modbus Master č. 01
    - S_01.csv – Konfigurační soubor Slave zařízení patřící pod master č. 01

Naprogramování kitu HEX souborem z CD

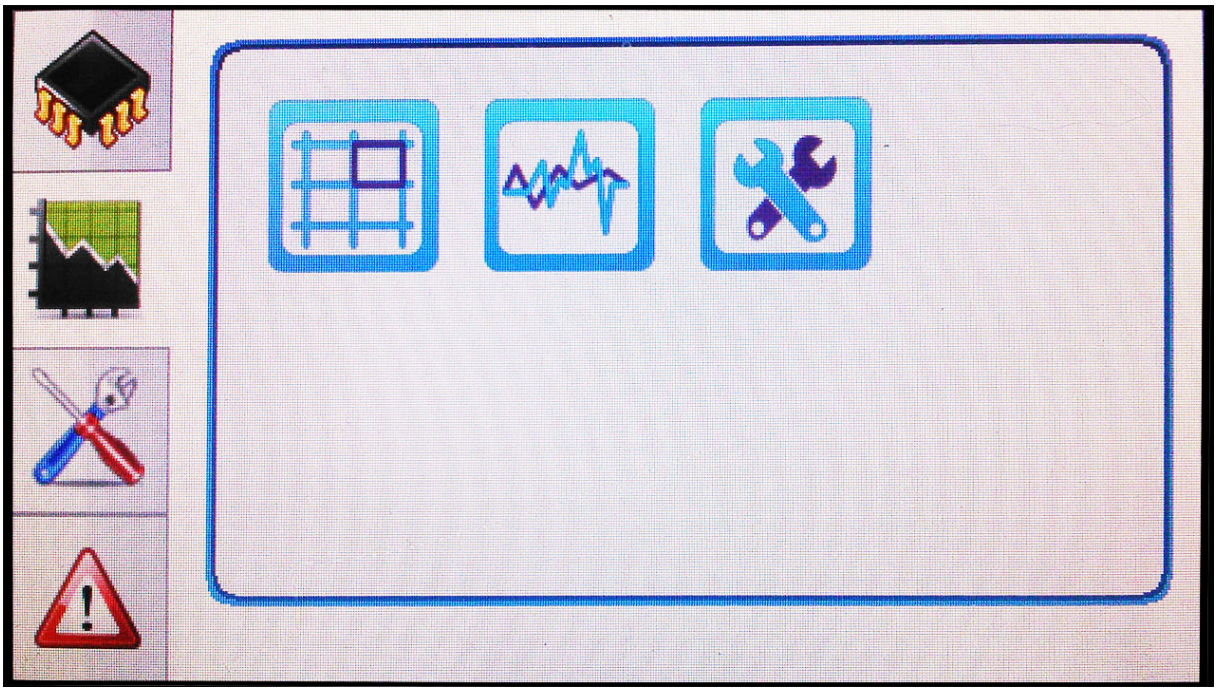
1. Nejprve je potřeba stáhnout program ST-Link Utility ze stránek STMicroelectronics (goo.gl/Rm6T0V)
2. Dále je potřeba program rozbalit a spustit
3. Instalační průvodce nás provede instalací
4. Po dokončení instalace spustíme nainstalovaný program a v horním menu vybereme položky External Louder -> Add External Louder
5. V listu najdeme a zaškrtneme řádek s textem „N25Q128A_STM32F746G-DISCO“
6. a potvrdíme.
7. Otevřeme HEX soubor pomocí File -> Open File…
8. V menu vybereme Target -> Program & Verify a naprogramujeme kit stisknutím tlačítka Start.
9. Naprogramování a výsledná aplikace je na následujících screenshotech.



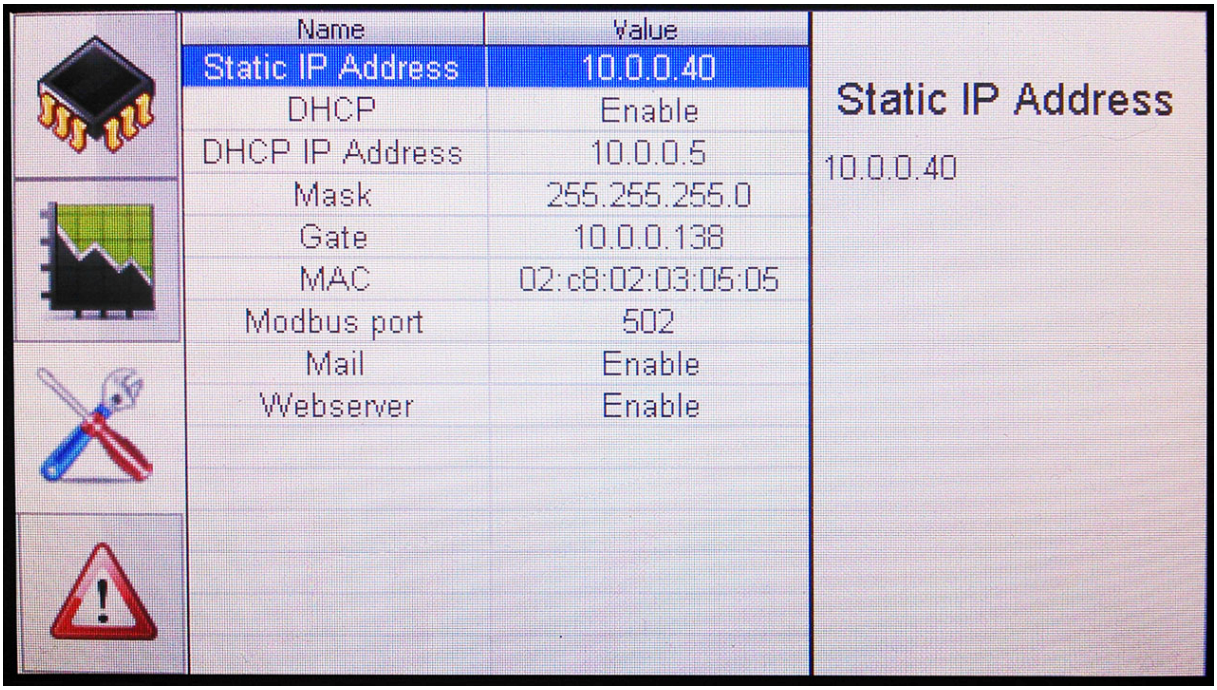
Obrázek 19 Naprogramování kitu aplikací ST-Link Utility



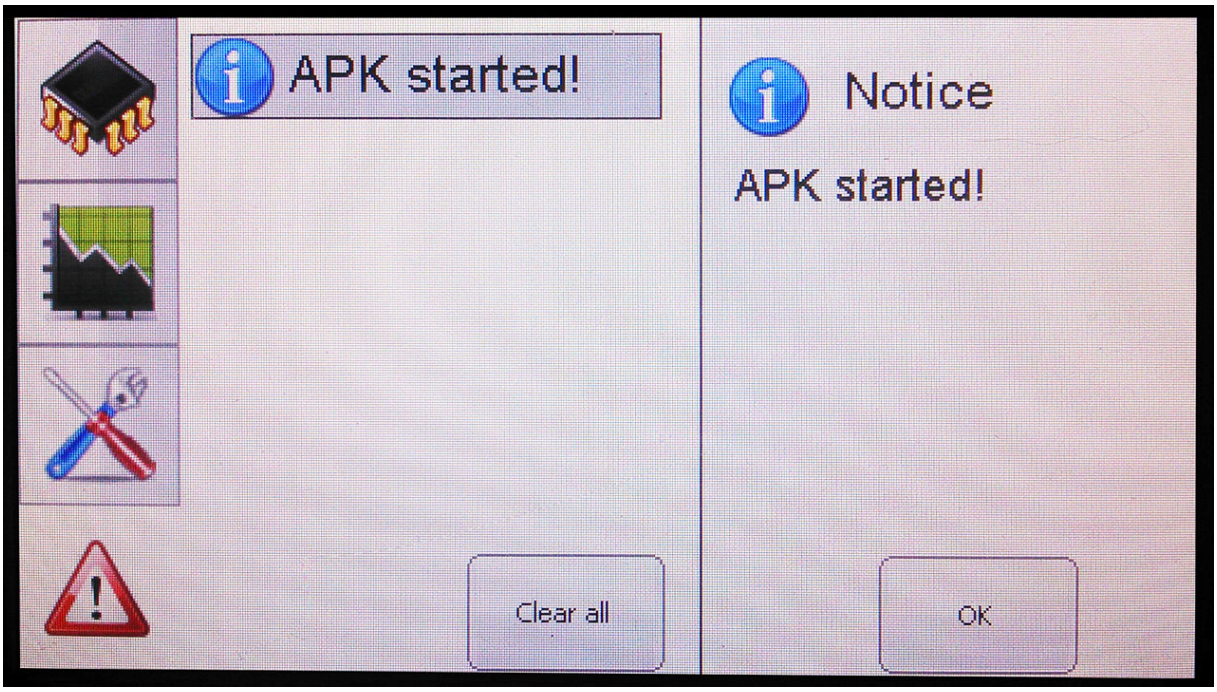
Obrázek 20 Výsledná aplikace - manažer modbus zařízení



Obrázek 21 Výsledná aplikace - Výběr zobrazení veličin



Obrázek 22 Výsledná aplikace - pohled do zobrazení nastavení



Obrázek 23 Výsledná aplikace - zobrazení hlášení