VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Josef Kořínek

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

HMI APLIKACE PRO PLC
HMI APPLICATION FOR PLC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Josef Kořínek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jakub Arm
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Josef Kořínek ***ID:*** 164311
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

HMI aplikace pro PLC

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit rozhraní člověk-stroj (HMI) pro ovládání laboratorního modelu „Kmeny“.

1. Realizujte připojení laboratorního modelu „Kmeny“ na PLC PFC200 a v prostředí Codesys vytvořte vzorovou úlohu pro řízení modelu.
2. Proveďte rešerši HW a SW prostředků pro rozhraní člověk-stroj daného PLC.
3. Navrhněte koncepci systému PLC-HMI.
4. Navrhněte a realizujte HMI aplikaci, kterou implementujte do zvoleného zařízení.
5. Demonstrujte funkčnost řešení a vyhodnoťte jeho vlastnosti (komunikace, systémové zdroje).

DOPORUČENÁ LITERATURA:

Hollifield, B., Eddie, H., Dana, O., TotalBoox,, & TBX,. (2013). The High Performance HMI Handbook. Plant Automation Services, Inc.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Jakub Arm
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Cílem této práce je vytvořit rozhraní člověk-stroj. V teoretické části se práce zabývá pravidly a doporučeními pro návrh vysoce výkonného HMI. Dále vyhledáním vhodných hardwarových prostředků v podobě jednodeskových počítačů a dotykových obrazovek. Následně vyhledáním softwarových prostředků v podobě knihoven pro komunikaci s PLC a knihoven pro tvorbu a zobrazení grafického prostředí. Na konci teoretické části je navržen koncept řešení pro realizaci rozhraní člověk-stroj.
V praktické části se práce zabývá připojením PLC PFC200 od firmy Wago k laboratornímu modelu Kmeny. Dále vytvořením programu v prostředí Codesys pro řízení modelu Kmeny. Následně vytvořením HMI aplikace v jazyce C++ s použitím vybraného frameworku GLG Toolkit a knihovny Libmodbus a její implementací do vybraného jednodeskového počítače Raspberry Pi 2. Nakonec jsou zhodnoceny vlastnosti vytvořeného HMI.

KLÍČOVÁ SLOVA

Rozhraní člověk-stroj, PLC PFC200, Jednodeskové počítače, Raspberry Pi, Banana Pi, Orange Pi, BeagleBord, Modbus, Libmodbus, GLG Toolkit

ABSTRACT

The aim of this work is to create a human-machine interface. In the theoretical part, the thesis deals with the rules and recommendations for the design of high user-friendly HMI. Next, the suitable hardware in the form of single-board computers and touch screens is addressed. Furthermore, the possible software like libraries for communication with PLCs and libraries for creating a graphical environment are covered. Finally, the concept of the human-machine interface is presented.
In the practical part, the thesis deals with the connection of PLC PFC200 from Wago to the laboratory model Kmeny. Additionally, the program in the Codesys for control the model is created. Then, the created HMI application is described that is built using the chosen GLG Toolkit framework and the Libmodbus library. Furthermore, the implementation into the chosen signel-board computer Raspberry Pi 2 is outlined. Finally, some of the key parameters of the created system are evaluated.

KEYWORDS

Human–Machine Interface, PLC PFC200, Single-board computers , Raspberry Pi, Banana Pi, Orange Pi, BeagleBord, Modbus, Libmodbus, GLG Toolkit

KOŘÍNEK, Josef. *HMI aplikace pro PLC*. Brno, Rok 2018, 49 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Jakub Arm

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „HMI aplikace pro PLC“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé bakalářské práce panu Ing. Jakubu Armovi za ochotu, odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem této práce je vytvořit rozhraní člověk-stroj. Rozhraní člověk-stroj anglicky human-machine interface zkráceně HMI je zařízení, jenž má za úkol poskytnout člověku pohled do technologického procesu a informovat ho o procesních parametrech a to v reálném čase. Dále by mělo umožnit člověku pohodlnou změnu procesních parametrů.

V teoretické části se práce zabývá pravidly a doporučeními pro návrh vysoce výkonného HMI. Dále vyhledáním vhodných hardwarových prostředků v podobě jednodeskových počítačů a dotykových obrazovek. Následně vyhledáním softwarových prostředků v podobě knihoven pro komunikaci s PLC a knihoven pro tvorbu a zobrazení grafického prostředí. Na konci teoretické části je navržen koncept řešení pro realizaci rozhraní člověk-stroj.

V praktické části bude provedeno připojení PLC PFC200 od firmy Wago k laboratornímu modelu Kmeny. Dále vytvoření programu v prostředí Codesys pro řízení modelu Kmeny. Poté návrh a implementace HMI aplikace do zvoleného zařízení. Nakonec budou zhodnoceny vlastnosti vytvořeného HMI.

2 Teorie

2.1 Základní principy pro návrh vysoce výkoného HMI

Nejdůležitějším prvkem pro návrh HMI je uvědomit si pro koho je vytvářeno. Mnohdy je požadavek, aby návrh HMI splňoval potřeby větší skupiny uživatelů například: manažerů, vedoucích, inženýrů a operátorů. Tyto potřeby však nejsou shodné a v takovém případě je nutno sáhnout ke kompromisu, což ale není žádoucí. Hlavním, nejdůležitějším a také nejčastějším uživatelem HMI je operátor. Tudíž je žádoucí HMI navrhnout pro něho a informace ostatním skupinám uživatelů poskytovat jinými způsoby.[1]

Cílem HMI je poskytnou operátorovi potřebné informace v jasné a intuitivní formě a takovým způsobem, aby byla minimalizována možnost chyb. K dosažení těchto požadavků je potřeba řídit se třemi základními principy, které jsou:
Přehlednost

- HMI by mělo být po grafické stránce přehledné, srozumitelné a intuitivní.
- Stav procesu a provozní podmínky by měly být jasně patrné.
- Grafické prvky pro řízení procesu mají být jednoduše a jasně rozeznatelné.
- Grafika by neměla obsahovat nepotřebné detaily.
- Grafika by měla sdělovat relevatní informace a ne zobrazovat jen data.
- Informace mají být upřednostňovány podle důležitosti.
- Alarmy a upozornění na vyjimečné stavy mají být jasně viditelné a rozlišitelné podle důležitosti.

Konzistivita

- HMI by mělo být hiearchicky a logicky uspořádané.
- Grafické prvky by měly být intuitivní, standartizované a ovládání by mělo vyžadovat minimum úkonů.

Zpětná vazba

- HMI by mělo být navrženo tak, aby co nejméně unavovalo obsluhu.
- Zobrazovací i ovládací prvky by se měly chovat předvídatelně a stejně za všech okolností.
- Akce s možnými negativními důsledky by měly obsahovat potvrzovací mechanismy, aby se předešlo nechtěné aktivaci.

Použitím těchto tří základních principů dosáhneme následujících výsledků. Pozornost operátora je nasměrována k nejdůležitějším informacím. Reakční doba operátorátora je minimalizována, protože je HMI navrženo tak, aby se v něm mohl snadno orientovat a zásahy vyžadovaly co nejméně úkonů. Chybám a zmatení operátora je zabráněno díky předvídatelnému chování, jednoduchému ovládání a zpětné vazbě.[1]

2.1.1 Znamení dobré a špatné grafiky

Známení nevhodné grafiky

- Neobsahuje grafy.
- Velké blikající plameny zobrazující hořáky.
- Jasně barevné procesní prvky zobrazené ve 3D.
- Mnoho křížících se linií.
- Barvy určené pro alarmy se vyskytují na běžných objektech.
- Detailní vyobrazení neměnících se objektů.
- Animované dopravníky, pumpy.
- Animace tekutin, postřikovačů a podobné.
- Jednotky zobrazené velkým jasným písmem.[1]

Znamení dobré grafiky

- Animace slouží pouze pro zdůraznění zvláštních situací.
- Pozadí je šedé z důvodu nízkého oslnění a je použit nízký kontrast.
- Rozvržení grafických prvků je totožné s tím, jak operátor vnímá technologický proces.
- Zobrazení procesních stavů je realizováno v kontextu s informací.
- Nejdůležitější procesní parametry mají graf.
- Omezení použití barev a barvy určené pro alarmy nejsou použity jinde.
- Objekty určené pro spouštění důležitých akcí obsahují potvrzovací dialog.
- Jednotky jsou zobrazeny s nízkým kontrastem nebo nejsou vůbec zobrazeny.[1]

2.1.2 Data a informace

Dalším důležitým prvkem při tvorbě HMI je uvědomit si fakt, že data nejsou vždy informace. Mezi daty a informacemi je rozdíl. Pokud HMI obsahuje velké množství dat stává se nepřehledným. Navíc vyhodnocení dat může vyžadovat podrobné znalosti o funkci technologického procesu, které operátor nemusí mít. Proto je důležité uvádět číselná data s jejich mezemi, aby měl operátor informaci, jestli je stav technologického procesu v mezích nebo ne. Nejlepší je když je poloha v přípustných mezích zobrazováná pouze graficky a při vybočení se dodatečně zobrazí i číselný údaj pro přesnější informaci.[1]

2.1.3 Analogové ukazatele

Čtení a porozumění analogovým ukazatelům je pro člověka podstatně rychlejší a vyžaduje podstatně méně soustředění. Použití analogových ukazatelů je proto v mnoha případech vhodnější, protože zajišťují rychlejší idetifikaci vybočení z mezních hodnot

a rychlejší korekci tohoto stavu operátorem. Navíc díky potřebě menšího soustředění méně psychicky unavují operátora a ten tak zůstává pozornější při monitorování technologického procesu.[1]

2.1.4 Použití barev

Vnímání barev je obsáhlé téma, o kterém už bylo napsáno mnoho publikací. Při návrhu HMI jsou však barvy často užívány nadbytečně a nesprávně. Při tvorbě vysoce výkoného HMI je nutné barvy používat omezeně pouze ve zvláštních případech pro zdůraznění důležitých situací. Výběr a použití barev při tvorbě HMI se musí řídit stejnými pravidly pro celé HMI. Barva by však nikdy neměla být jediným zdůrazňujícím faktorem, ale měla by být doplněna dalšími odlišovacími technikami například změnou tvaru atd.[1]

Správné užívání barev je prvek dělající grafiku snadněji pochopitelnou, přehlednější. Přehnané využívání barev při návrhu HMI dělá naopak grafiku nepřehlednou. Operátor se v ní orientuje pomaleji a s větším psychickým úsilím, což způsobuje jeho pomalejší reakce, zvýšenou míru únavy a větší pravděpodobnost přehlédnutí nebezpečného stavu nebo nesprávné reakce.[1]

Barvy pozadí

Dlouhou dobu se předpokládalo, že vhodného kontrastu se dosáhne světlým nebo tmavým pozadím ne však něčím mezi. Toto doporučení vycházelo z omezených zobrazovacích schopností dřívějších displejů. Konstrast ale není vše. Vysoký konstrast, obzvláště světlé prvky na černém pozadí, je při delším sledování obrazovky unavující a nepříjemný pro oči. Ideálním pozadím při návrhu HMI je světle šedá barva, která je pro oči mnohem méně unavující a méně nepříjemná. Samozřejmostí je dobré osvětlení kontrolní místnosti. Nejlepších výsledků dosáhneme experimentováním s různými odstíny šedé na pozadí a objektů v popředí. Obecně doporučovanými barvami vhodnými pro pozadí jsou odstíny šedé s kódy RGB 221, 221, 221 a RGB 192, 192, 192.[1]

Vnímání barev lidským mozkem

Barvy jsou lidským mozkem vnímány předběžně. Pozornost člověka je proto přitahována k barevným objektům. Obzvláště pokud je barevný objekt ve vysokém konstrastu s pozadím je mu věnována velká pozornost. Tohoto poznatku je vhodné při návrhu HMI důsledně využívat a barvy a konstrast použít pro zdůraznění objektů a mimořádných stavů.[1]

Barvy objektů v popředí

Při návrhu grafiky HMI je vhodné používat barvy co nejméně a nejstřídměji. HMI grafika se výrazně liší od ostaní grafiky používané na PC, která má za úkol navnadit, zaujmout a upoutat uživatele. Časté využívání barev je tam proto žádoucí. Není tam vyžadována okamžitá akce s možnými závažnými důsledky. Při návrhu HMI je ale důležitá především rychlá orientace a porozumění stavu operátorem, a tudíž je střídmé užívání barev na místě. Procesní linie a obrysy tanků by měly být tmavě hnědé nebo černé ne však barevné. Zdůraznění by mělo být realizováno tloušťkou čar a ne barvami. Barvy by měly sloužit pouze pro zobrazení mimořádných stavů a zdůraznění důležitých objektů.[1]

Barvy jsou však špatně vnímány periferním viděním, takže by neměly být jediným způsobem pro upozornění. Vhodné je doplnění o blikání, které je periferním viděním dobře rozeznatelné. Ani blikání by nemělo být používáno nadměrně, ale pouze za účelem zdůraznění mimořádného stavu.[1]

Barvoslepost

Rovněž při návrhu HMI je třeba uvážit, že operátor může trpět barvoslepostí a barva by tedy nikdy neměla být jediným informačním prvkem.[1]

2.1.5 Zobrazování textu

Při zobrazováni textu je pořeba doržovat několik principů. Za prvé množství textu by mělo být omezeno. Obrazovka by neměla připomínat manuál. Text by měl být použit pro popis prvků, u kterých není jejich název zřejmý z tvaru nebo pozice na obrazovce. Za druhé text by měl být z důvodu kontrastu světle šedý ne černý. Za třetí text zobrazovaný na obrazovce by měl využívat vždy bezpatkový font. Tištěné knihy a dokumenty sice používají fonty patkové, protože to činí čtení snadnější a rychlejší. Ale ty však disponují větším rozlišením než zobrazovací displeje, na kterých jsou patkové fonty hůře čitelné. Dalším principem, kterým je potřeba se řídit je použití velkého dobře viditelného textu pro rozeznání stejně vypadajících zařízení vyskytující se na obrazovce vícekrát, pokud je není možné rozeznat polohou. Obzvláště v případě, že je na podrobné informační obrazovce zobrazen pouze jeden objekt z několika stejných v technologickém procesu se vyskytujících objektů. Mezi další principy patří použití pouze velkých písmen v krátkých popiscích jednoslovných názvech. V ostatních případech je čitelnější použití velkého písmena na začátku popisu a zbytek písmen malým písmem. Dalším principem je udržovat slovník použitých zkratek a vyhnout se tak pozdějším problémů při nejasném významu zkratky. Ohledně velikosti písma existuje mnoho doporučení. Pro typickou vzdálenost 0,6

metrů od obrazovky je minimální velikost písma 2,8 mm, doporučená velikost je 2,5mm a mximální 4,1mm. Nejlepší způsobem je však nechat velikost textu vyzkoušet operátorem a upravit ji podle jeho požadavků. Pro operátora je nejvhodnější, když má jasně čitelný text přímo před sebou a není tedy vhodné umisťovat na obrazovku velké množství dat a textu, díky čemuž je následně nutné použít malé a hůře čitelné písmo a displej se tak stává nepřehledným.[1]

2.1.6 Zobrazování hodnot

Pro zobrazování měnících se hodnot platí jiná pravidla než pro zobrazení statického textu. Vhodnou barvou pro zobrazení měnících se hodnot při použití světle šedého pozadí je tučná modrá barva, díky níž dojde k jednoznačnému oddělení od statického textu. Vedoucí nuly nejsou zobrazovány s výjímkou vedoucí nuly u desetinných čísel. Hodnoty jsou zobrazovány pouze s tolika desetinnými místy, kolik je potřebné pro operátora a ne více. V případě, že je potřeba zobrazit vedle hodnot i jednotky. Použijeme pro jejich zobrazení text s nízkým kontrastem a umístíme je blízko hodnoty. Pokud jsou hodnoty vybrány k editaci, měl by být takový stav indikován. Ideálním způsobem, jak toho dosáhnout, je ohraničit text bílým rámečkem. Zprávy o akcích by měla být krátká, jasná a jednoznačne přiřaditelná provedené akci.[1]

2.1.7 Alarmy

Každá hodnota překračující meze alarmu musí být jasným a nepřehlédnutelným způsobem zdůrazněna. Existuje několik metod jak toho dosáhnout. Zpravidla se řídí těmito základními principy.

- S prioritou alarmu je pevně spjata barva. Každá priorita má svou barvu, která se v HMI nikde jinde nevyskytuje.
- Nepotvrzený alarm by měl být odlišený od potvrzeného. Nejčastějším způsobem je to realizováno blikáním nepotvrzeného alarmu.
- V případě, že je aktivních více alarmů naráz je zobrazován ten s největší prioritou.[1]

Priority alarmů

Je dobrým zvykem rozdělovat alarmy na tři priority a odělený diagnostický alarm. Diagnostický alarm je alarm detekující problém operátorem neřešitelný a vyžadující zásah údržby. Pro barvy přiřazené jednotlivým prioritám alarmu platí následující doporučení.

- Pro nejvyšší prioritu je doporučená barva červená.
- Pro druhou nejvyšší prioritu je doporučená barva žlutá.

- Pro třetí nejvyšší prioritu se doporučuje oranžová barva.
- Pro diagnostiký alarm s nejnižší prioritou je doporučená barva purpurová.

Důležité je, aby barvy byly na displeji snadno a neomylně rozeznatelné od ostatních použitých barev.[1]

Zobrazování alarmů

Pro zobrazování alarmů lze použít několik způsobů. Prvním je celistvý barevný blok za zobrazovanou hodnotou. Tento způsob má své výhody mezi něž patří. Barva přitahuje pozornost na hodnotu mimo meze. Pro zobrazení nepotvrzených alarmů bliká pozadí a ne text, který je tím pádem stále viditelný a tudíž dobře čitelný. Rovněmž má tento způsob i své nevýhody. Kombinace barev může být problematická obzvláště modrý text na červeném pozadí není dobře čitelný. Priorita alarmů je zobrazována jen barvou, což není vhodné pro barvoslepé.[1]

Druhým způsobem zobrazení alarmu je barevný rámeček kolem hodnoty mimo meze. Stejně jako u předcházejícího způsobu barva přitahuje pozornost operátora, i když v trochu menší míře. Pro zobrazení nepotvrzených alarmů platí stejná výhoda a to blikání pozadí a ne textu, který zůstává viditelný a tudíž stále dobře čitelný. Výhodou oproti předchozímu způsobu je však odstranění problému s čitelností při použití modré a červené. Mezi nevýhodami rovněž zůstáva indikace priority alarmu pouze barvou.[1]

Třetím způsobem je zobrazení nového grafického prvku poblíž hodnoty, který se objeví při aktivitě alarmu a při neaktivitě zmizí. Mezi výhody tohoto řešení patří výhody předchozích způsobů. Barva přitahuje pozornost k hodnotě mimo meze. U nepotvrzených alarmů bliká pouze grafický prvek a ne text, takže není omezena jeho čitelnost. Nevyskytuje se problém s kombinací barev. Grafický prvek alarmu může být umístěn kdekoliv poblíž hodnoty. Nicméně pozice by měla být u všech hodnot stejná. Novou výhodou je zobrazení priority i jinými způsoby než jen barvou. Například tvarem nebo číslem uvnitř prvku. Všechny předchozí způsoby mají ale jednu nevýhodu. Žádný způsob neukazuje přímo typ alarmu.[1]

Čtvrtý způsob se tedy liší od třetího pouze tím, že je uvnitř grafického prvku zobrazena navíc zkratka typu alarmu. Ač se to může zdát jako výhoda, nemusí tomu tak být vždy. Například je možné, aby měla jedna hodnota současně aktivních více typů alarmů, přičem se současně dovnitř grafického prvku nemusí vejít nebo musí být prvek zbytečně zvětšen. Větší indikační prvek alarmu zabere více místa, které se navíc násobí počtem grafických prvků indikujících alarm. Chybějící místo nás potom při návrhu nutí zmenšit ostatní grafické objekty nebo text a tím se zbytečně snižuje přehlednost displeje. Rovněž není doporučováno, aby byl typ alarmu signalizován barvou zobrazované hodnoty. Vzhledem k tomu, že výhoda tohoto způsobu

sebou nese problémy pro praktickou realizaci, není tento způsob realizace alarmu doporučován a nejlepším způsobem je tedy způsob třetí.[1]

Přístup k alarmům

Z předchozí podkapitoly o zobrazování alarmů zůstává však ještě nedořešen přístup k typu alarmu, dalším informacím a podrobnostem o nastalé situaci. V předcházející podkapitole jsme už došli k rozhodnutí, že není vhodné přidávat typ alarmu do grafického prvku alarm indikující. Musíme to tedy řešit jinak. Rovněž z předchozích doporučení víme, že efektivní HMI by mělo dosahovat požadovaných cílu s co nejmenším počtem stisknutí kláves. Proto je nejvhodnějším způsobem aby, na každé obrazovce kde může dojít k zobrazení alarmu byla možnost přejít jedním stiskem tlačítka na podobrazovku, kde se budou zobrazovat typy alarmů a jejich podrobnosti seřazené podle priority. Podobrazovka s aktivními alarmy by měla rovněž umožnovat potvrzení alarmu, a to pouze jednou. Jednou potvrzený alarm by už neměl vyžadovat další potvrzení při nezměněných podmínkách.[1]

2.1.8 Zobrazování objektů

Pro zobrazování objektů by měly být vytvořeny jednotné vzory tak, aby všechny objekty stejného typu vypadaly stejně a tím byl dán jednotný vzhled. Popisy by měly mít jednotnou strukturu. Pokud je identifikace objektu zřejmá, popis není potřeba. Není ani vhodné, aby měl každý objekt u sebe popis, protože nadměrné množství popisků HMI znepřehledňuje a zhoršuje v něm orientaci.[1]

2.2 Rešerše HW a SW prostředků pro rozhraní člověk stroj PLC-PFC200

Pro realizaci HMI komunikujícím s PLC PFC200 se bude v první řadě nutné seznámit s PLC samotným, především s podporou komunikačních protokolů, kterými disponuje. Následně bude třeba vybrat ten nejvhodnější a podrobněji ho rozebrat. Poté přijde na řadu vyhlednání hardwaru, na kterém bude aplikce běžet a který bude podporovat vybraný komunikační protokol. Nakonec bude vyhledán software ve formě knihoven, jenž budou použity pro zobrazování grafiky HMI aplikace a rovněž komunikaci s PLC.

2.2.1 PLC-PFC200

Jedná se o modulární PLC s procesorem ARM cortex A8 600 MHz s operačním systémem Linux podporujícím provoz v reálném čase. Disponuje 256 MB paměti RAM a 256 MB paměti flash. PLC je kompatibilní se všemi moduly ze systému WAGO-I/O-SYSTEM 750. Pro komunikaci je vybaveno dvěmi porty Ethernet, jedním portem RS232/485 a podporou protokolů DHCP, DNS, NTP, FTP, FTPS, SNMP, HTTP, HTTPS, SSH, MODBUS (TCP, UDP, RTU).

Modul PFC200 lze konfigurovat prostřednictvím webového administračního rozhraní a programovat pomocí programovacího prostředí Codesys. PLC podporuje programovací jazyky IL, LD, FBD, ST a FC podle normy IEC 61131-3.[8]

2.2.2 Modbus

Modbus je komunikační protokol vytvořený firmou Modicon. V komunikaci vystupují dvě strany klient-master a server-slave. Komunikace probíhá stylem požadavek a následná odpověď. Protokol je definován na úrovni aplikační vrstvy ISO/OSI modelu díky čemuž lze provozovat na řadě komunikačních medií např. RS-232, RS-422, RS-485, Ethernet TCP/IP.[2]

Obsah zprávy

Protokol Modbus definuje obsah zprávy na úrovni protokolu (PDU - Protocol Data Unit). Obsah zprávy na úrovni protokolu je tvořen kódem funkce a daty a je nezávislý na použitém komunikačním médiu. Dále je zpráva rozšířena o další části podle použitého komunikačního média především o adresu a kontrolní součet a tvoří tak zprávu na aplikační úrovni (ADU - Application Data Unit).[2]

Způsob komunikace

Komunikaci vždy zahajuje klient. Kód funkce unvitř zprávy oznamuje serveru, jaký typ operace má provést. Kód funkce může nabývat hodnot z rozsahu 1-255. Funkční kódy 128-255 jsou ale vyhrazeny pro oznámení chyby. Datová část pak tvoří parametry pro operaci a u některých operací není potřeba. Pokud proběhne operace v pořádku, server vrací zprávu, jenž má v kódu funkce kód provedené operace pro kontrolu úspěšného provedení. Dále pak v datové části požadovaná data pokud má nějaké poslat. V případě chyby vrací server v kódu funkce kód požadované operace s nastaveným nejvyšším bitem. A v datové oblasti pak chybový kód oznamující důvod chyby. [2]

Maximální velikost zprávy PDU vychází z první implementace na sériové lince a je 253 bytů. Protokolem Modbus jsou definovány tři základní typy zpráv PDU.

1. Požadavek tvořený jedním bytem pro kód funkce a zbytkem pro data.
2. Odpověď tvořená jedním bytem vráceného kódu funkce a zbytek je tvořen vrácenými daty.
3. Oznámení chyby tvořené jedním bytem pro oznámení selhání a jedním bytem pro chybový kód oznamující důvod chyby.

Modbus je typu Big-endian, tedy pokud posílá čísla větší než jeden byte pošle nejdříve nejvyšší byte, poté druhý nejvyšší a tak dále až po nejnižší byte.[2]

Rozdělení dat

Protokol Modbus rozděluje data podle typu do čtyř tabulek. Data z tabulek lze číst po jednom nebo po větších skupinách omezených pouze maximální velikostí datové části zprávy. Prvním typem jsou Diskrétní vstupy. Každá položka je tvořena jedním bitem. Data jsou přístupná v režimu pouze pro čtení. Zpravidla se jedná o vstupy PLC.

Druhým typem jsou cívky. Každá položka je tvořena jedním bitem, ale mohou být čteny a zapisovány.

Třetím typem jsou vstupní registry. Každá položka je tvořena slovem tedy 16 bity. Vstupní registry jsou přístupné pouze pro čtení.

Čtvrtým a posledním typem jsou uchovávací registry. Uchovávací registry jsou také tvořeny slovem tedy 16 bity a lze je jak číst tak zapisovat.

Dle specifikáci protokolu může mít každá tabulka až 65536 položek.[2]

Kódy funkcí

Kódy funkcí se dle specifikace protokolu Modbus dělí na tři skupiny.

1. Veřejné - Unikátní kódy definované a schválené společností MODBUS-IDA.org.
2. Uživatelsky definované - Umožňující uživateli implementovat funkci, která není definována specifikací.
3. Rezervované - Kódy funkcí, které jsou v současnosti používány některými firmami a které nejsou dostupné pro veřejné použití.[2]

Popis vybraných kódů funkcí

- *0x01 Čti cívky* - Přečte stav jedné cívky nebo až 2000 cívek na jednou. V požadavku musí být uvedena adresa první cívky a počet čtených cívek. Data jsou vracena po bytech, přičemž v každém je obsažen stav 8 cívek.
- *0x02 Čti diskrétní vstupy* - Tato funkce se chová podobně jako funkce předchozí s tím rozdílem, že čte z jiné tabulky dat. Přečte stav diskrétních vstupů v počtu od 1 do 2000 na jednou. V požadavku musí být uvedena adresa prvního

diskrétního vstupu a počet diskrétních vstupů. Data jsou vracena po bytech, přičemž v každém je obsažen stav 8 diskrétních vstupů.

- *0x03 Čti uchovávací registry* - Slouží ke čtení obsahu uchovávacích registrů od jednoho až do 125. V požadavku je potřeba uvést adresu prvního registru a počet registrů, které chceme přečíst. V odpovědi odpovídá každému registru dvojice bytů.
- *0x04 Čti vstupní registry* - Stejně jako funkce předchozí čte registry tentokrát ovšem místo uchovávacích registry vstupní. Rovněž může číst v počtu od jednoho do 125 registrů. Požadavem musí obsahovat adresu prvního registru a počet registrů, které chceme přečíst.
- *0x05 Zapiš jednu cívku* - Tato funkce nastavuje jednu cívku do stavu log 1 nebo 0. V požadavku je specifikována adresa výstupu, který se má nastavit a hodnota, na kterou se má nastavit. Pokud se má cívka nastavit na hodnotu log 0 musí být v požadavku uvedena hodnota 0x0000, pokud se má nastavit do log 1 tak musí být v požadavku uvedena hodnota 0xFF00. V případě úspěchu vrací server kopii požadavku.
- *0x06 Zapiš jeden registr* - Tato funkce zapíše data do jednoho uchovávacího registru. V požadavku je třeba specifikovat adresu registru, do kterého se má zapsat a hodnotu, kterou tam chceme zapsat. Pokud nenastane chyba, server vrátí, po té co je register zapsán jako odpověď kopii požadavku.
- *0x0F Zapiš více cívek* - Pomocí této funkce dojde k nastavení hodnot až 1968 cívek do stavů log 1 nebo log 0. V požadavku se uvádí adresa prvního výstupu, který se má nastavit a hodnoty, na které se mají výstupy nastavit. Odpověď značící bezchybné zapsání obsahuje počáteční adresu a počet nastavených cívek.
- *0x10 Zapiš více registrů* - Zapíše blok až 120 registrů. V požadavku se uvádí adresa prvního registru, který se má zapsat, počet registrů a hodnoty, které se mají zapsat. Odpověď značící úspěch obsahuje počáteční adresu a počet zapsaných registrů.
- *0x17 Čti/Zapiš více registrů* - Pomocí této funkce je provedena kombinace čtení a zápisu registrů v jedné transakci. Nejdříve jsou data zapsána a poté přečtena. Požadavek musí obsahovat adresu prvního registru a počet registrů, které se mají číst. Dále pak adresu, počet registrů a hodnoty, které se mají zapsat. V případě úspěchu server opoví kódem funkce a přečtenými daty.[2]

Zpracování chyby

Po vyslání požadavku od klienta může nastat několik možných situací. Standartně server požadavek přijme v pořádku provede žádanou akci a odpoví zprávou, jenž

Kód	Název	Význam
01	Ilegální funkce	Server požadovanou funkci nepodporuje
02	Ilegální adresa dat	Zadaná adresa je neplatná
03	Ilegální hodnota dat	Data přijatá serverem jsou chybná
04	Selhání zařízení	Při provádění operace se vyskytla neodstranitelná chyba

Tab. 2.1: Nejčastější chybové kódy protokolu Modbus [2]

obsahuje kód provedené akce a případně data, pokud je to požadováno například při čtení. Pokud však server požadavek nepřijme z důvodu poruchy v komunikaci, není o tom klient žádným způsobem obeznámen a je proto nutné, aby měl implementovaný časový limit během, kterého čeká na odpověď a po jeho překročení uzná požadavek jako chybu. Pokud server požadavek přijme, ale je po kontrole parity nebo kontrolního součtu detekován jako chybný tak server neposílá zpět žádnou zprávu a klient tuto chybu musí detekovat opět za pomocí vypršení časového limitu na odpověd. Pokud server požadavek v pořádku přijme, ale není ho schopný vykonat, vrací klientovi oznámení chyby obsahující kód funkce požadavku, ve kterém nastaví nejvyšší bit a v datové oblasti kód chyby. Nejčastější chybové kódy protokolu Modbus jsou uvedeny v tabulce 2.1.[2]

Implementace Modbus TCP/IP

Modbus standart definuje nejen aplikační vrstvu ISO/OSI modelu a definuje i některé implementace protokolu Modbus. Například Modbus TCP nebo Modbus na sériové lince. Modbus TCP definuje zprávu ADU jako MBAP hlavičku, kód funkce a datovou část.[2]

Hlavička MBAP je dlouhá 7 bytů a je tvořena identifikátorem transakce (2 byty), identifikátorem protokolu (2 byty), délkou (2 byty) a identifikátorem zařízení (1 byte). Identifikátor transakce nastavuje klient, server ho pouze kopíruje. Identifikátor transakce využívá klient pro párování požadavku a odpovědi. Identifikátor protokolu slouží pro speciální účely. Pro Modbus protokol je nastaven na 0. Délka nese informaci o délce zbytku zprávy a zahrnuje tedy délku identifikátoru zařízení a délku dat. Identifikátor zařízení slouží pro účely komunikace sériové linky kdy data jdou část cesty po TCP-IP síti. Pro komunikaci Modbus TCP je vyhrazen port 502.[3]

2.2.3 HW prosředky

Raspberry Pi

Je malý jednodeskový počítač o velikosti platební karty. Byl vyvinut v roce 2012 bristskou nadací Raspberry Pi Foundantion. Dnes se vyskytuje na trhu ve třech verzích Raspberry Pi čtyři modely A, A+, B, B+, Raspberry Pi 2 model B a Raspberry Pi 3 model B.[10]

V základní verzi Raspberry Pi obsahuje

- procesor ARM Cortex A6 s taktem 700 MHz
- grafický procesor VideoCore IV
- 256-512 MB RAM podle modelu
- obrazový výstup HDMI
- zvukový výstup HDMI a 3,5 mm jack konektor
- 12 vstupně-výstupních pinů
- Slot pro SD nebo micro SD nebo MMC kartu
- UART
- sběrnice I2C a SPI
- JTAG Debug
- watchdog timer
- 1-4 USB porty verze 2.0 podle modelu
- Modely B a B+ obsahují ethernetový adaptér 10/100 s konektorem RJ45

Cena modelu B+ se pohybuje kolem 850 Kč.

Verze Raspberry Pi 2 obsahuje nověji

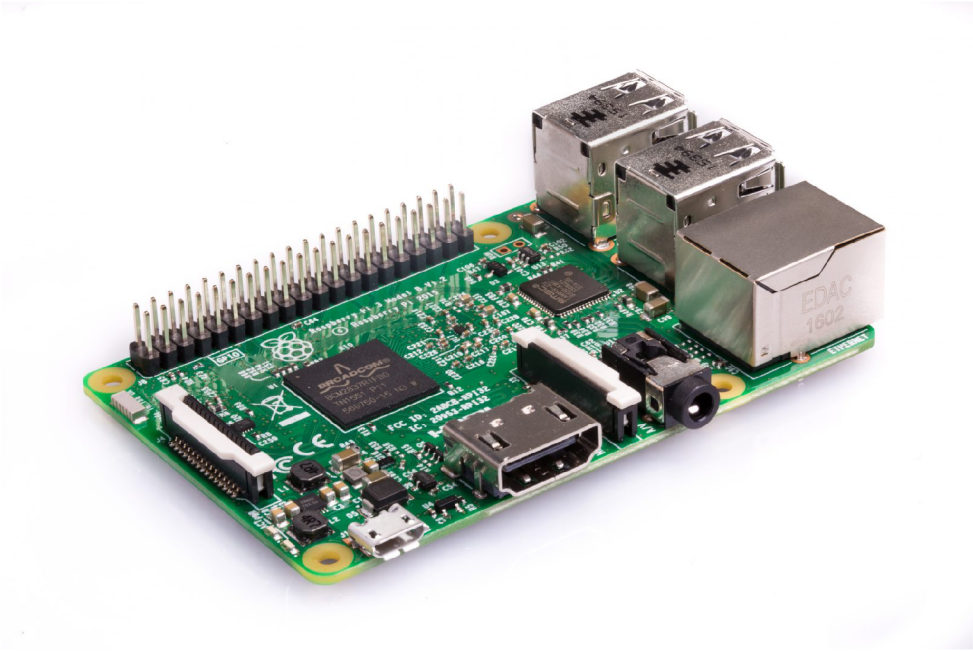
- čtyřjádrový procesor ARM Cortex A7 s taktem 900 MHz
- 1 GB paměti RAM

S cenou kolem 1000 Kč.

Nejnovější verze Raspberry Pi 3 obsahuje

- 64 bitový ARM Cortex A53 s taktem 1,2 GHz
- 1 GB paměti RAM
- Wi-Fi modul
- Bluetooth modul
- 40 vstupně-výstupních pinů

A stojí kolem 1000 Kč.



Obr. 2.1: Raspberry Pi 3 [9]

Hlavním operačním systémem pro Raspberry Pi je Raspbian. Jsou však podporovány i další Linuxové operační systémy např. Ubuntu Mate, Snappy Ubuntu Core, Risc OS. A dále i Windows 10 IOT core.[10]

Banana Pi

Jedná se jednodeskový počítač vyráběný firmou Lemaker podobný Raspberry Pi a je rovněž kompatibilní s většinou podporovaných příslušenství pro Raspberry Pi.[12]

Model BP-A20 M1 je tvořen

- dvoujádrovým procesorem ARM cortex A7 s taktem 1 GHz
- grafickým jádrem Mali400MP2
- 1 GB paměti RAM typu DDR3
- obrazovým výstupem HDMI
- zvukovými výstupy HDMI a 3,5mm jack konektor
- 26 vstupně-výstupními piny
- Slotem pro připojení SD karty
- Sata rozhraním
- dvěmi USB proty verze 2.0
- Ethernetovým adaptérem 10/100/1000 s konektorem RJ45[12]

Cena za model BP-A20 M1 je zhruba 1000 Kč.

Model BPI-M2 Berry nověji disponuje

- čtyřjádrovým procesorem ARM cortex A7
- slotem pro micro SD kartu
- WiFi
- Bluetooth
- 40 vstupně-výstupními piny
- čtyřmi USB porty verze 2.0

Cena modelu BPI-M2 Berry je 1400 Kč.

Model BPI-M3 má oproti předchozímu modelu BPI-M2 Berry

- osmijádrový procesor ARM cortex A7 s taktem 1.8 GHz
- grafické jádro PowerVR SGX544MP1
- 2 GB paměti RAM typu DDR3
- dva USB porty verze 2.0

Cena modelu BPI-M3 je 2500 Kč.

Banana Pi podporuje mnohé Linuxové operační sytémy např. Bananian, Raspbian, Ubuntu, Debian a operační systém Android.[11]

Obr. 2.2: Banana PI M1 [11]

Orange Pi

Stejně jako Raspberry Pi nebo Banana Pi je i Orange Pi jednodeskový počítač. Orange Pi vyrábí čínská firma Shenzhen Xunlong Software CO.,Limited. Postupně se objevilo mnoho modelů Orange Pi Plus, Orange Pi One či Orange Pi One Plus a Orange Pi Plus 2.[13]

Orange Pi One obsahuje

- čtyřjádrový procesor ARM cortex A7
- grafické jádro Mali400MP2
- video výstup HDMI
- 512MB paměti RAM typu DDR3
- dva USB porty verze 2.0
- 10/100M Ethernet RJ45
- jeden port USB 2.0
- 40 vstupně-výstupních pinů

Cena Orange Pi One se poohybuje kolem 800 Kč.

Orange Pi Plus 2 obsahuje

- čtyřjádrový procesor ARM cortex A7
- grafické jádro Mali400MP2
- video výstupem HDMI
- 2GB paměti RAM typu DDR3
- čtyři porty USB 2.0
- 10/100M/1000M Ethernetem RJ45
- 40 vstupně-výstupních pinů

Orange Pi Plus 2 stojí 1800 Kč.

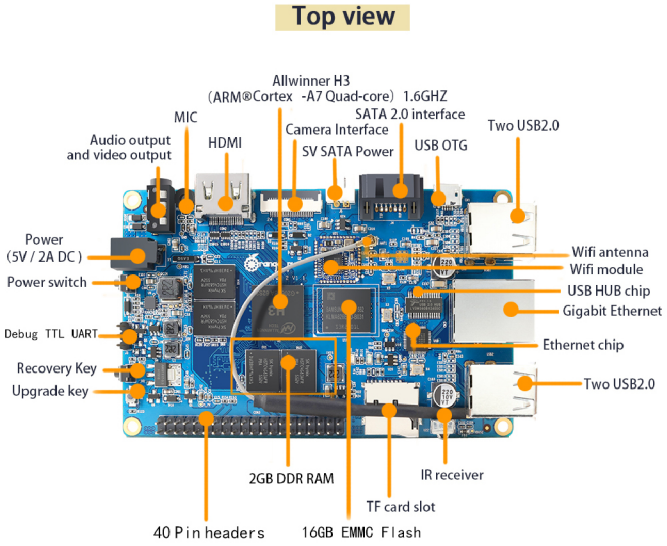
Orange Pi One Plus disponuje

- čtyřjádrovým 64 bitovým procesorem ARM cortex A-53
- grafickým jádrem Mali T720
- video výstupem HDMI
- 1GB paměti RAM typu DDR3
- jedním portem USB 2.0 a jedním micro USB portem
- 10/100M/1000M Ethernetem RJ45

Na Orange Pi lze stejně jako na Raspberry Pi a Banana Pi nainstalovat různé Linuxové distribuce jako např. Ubuntu, Debian, Raspbian a Android.[14]

BeagleBord

Další v řadě jednodeskových počítačů je BeagleBord vyvinutý firmou Texas Instruments. BeagleBord existuje v několika verzích BeagleBoard, BeagleBone a BeagleBone Black.[16]



Obr. 2.3: Orange Pi Plus 2 [14]

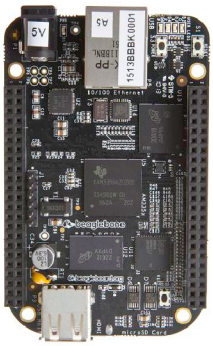
BeagleBone obsahuje

- procesor ARM cortex A-8 s taktem 720 MHz
- 256MB DDR2 RAM
- jeden port USB 2.0
- Ethernet
- 2x46 vstupně-výstupních pinů

BeagleBone Black disponuje

- procesorem ARM cortex A-8 s taktem 1 GHz
- 512MB DDR3 RAM
- grafickým výstupem HDMI
- jedním portem USB 2.0
- Ethernetem
- 2x46 vstupně-výstupními piny BeagleBone Black stojí 1000 Kč.

BeagleBone umožňuje běh řady Linuxových operačních systémů Angstrom Distribution, Ubuntu, Debian, ArchLinux, Sabayon, Buildroot, Erlang, Fedora.[15]

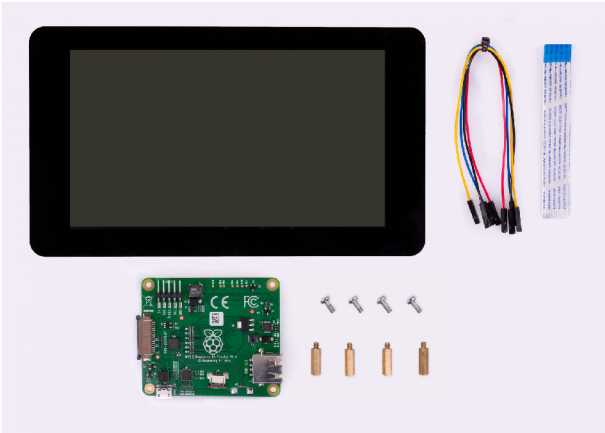


Obr. 2.4: BeagleBone Black [15]

Raspberry Pi Touchscreen

Jedná se o sedmipalcový kapacitní dotykový displej, který se připojuje k Raspberry Pi pomocí DSI kabelu pro přenos signálů a dvou drátových propojek pro napájení. Displej disponuje rozlišením 800 x 480 a dotyková plocha detekuje až 10 dotyků na jednou. Displej podporuje modely Raspberry Pi A, B, A+, B+, Raspberry Pi 2 model B a Raspberry Pi 3 model B.[9]

Raspberry Pi Touchscreen stojí okolo 1900 Kč.



Obr. 2.5: Dotykový displej pro Raspberry Pi [9]

2.2.4 SW prostředky

Libmodbus

Je knihovna napsaná v jazyce C umožňující komunikaci pomocí modbus protokolu jak po sériové lince tak po ethernetu. Je distribuová pod licencí LGPL v2.1+. Do-

stupná je pro platformy Linux, Mac OS X, FreeBSD, QNX a Win32. Podporuje přímo Raspberry Pi a pro standard RS485 i Beaglebone.[17]

FreeMODBUS

Jedná se o bezplatnou implementaci Modbus protokolu zaměřenou na embedded systémy. FreeMODBUS podporuje komunikaci pro embedded Linux a další platformy především po sériové lince, komunikaci protokolem TCP po ethernetu umožňuje pouze pro platformy Win32, lwIP/MCF5235, a lwIP/STR71X.[18]

Qt

Je multiplafromní knihovna, pro tvorbu uživatelského grafického rozhraní. Byla vyvinuta v roce 1999 společností Trolltech, v roce 2008 poté prodána firmě Nokia. Od roku 2014 paří firmě The Qt Company. Disponuje podporou pro platformy Windows, Linux, macOS a také pro Linuxové operační systémy běžící na architektuře ARM. Qt je primárně vyvíjena jako C++ knihovna ale podporuje i další jazyky jako `C#`, Java, Phyton, Ruby. Knihovna je dostupná pod GPL, LGPL licencí. Případně je možné zakoupit i komereční licenci.[19]

GTK+

GTK+ je multiplatformní knihovna pro tvorbu grafického rozhraní. Napsaná je v jazyce C ale podporuje i další jazyky jako Perl a Python. Podporuje platformy Windows, Linux a Mac OS X. Je šířena pod svobodnou licencí GNU LGPL takže lze použít jak pro vývoj svobodného softwaru tak i komerečního a to zcela bez poplatků. [20]

GLG Toolkit

Je grafický framework pro tvorbu grafických prostředí pracujících v reálném času. Je dostupný jak pod komereční licencí tak ve verzi Free Community Edition. Free Community Edition nabízí knihovny pro C/C++ a Javu obsahující předvytvořené komponenty a také interaktivní grafický editor pro tvorbu nových komponent. Knihovna C/C++ nabízí přímou podporu pro jednodeskové počítače Raspberry Pi a BeagleBone s procesory ARM pro architektury ARM6 a ARM7.[21]

µGFX

Jedná se o malou knihovnu pro tvorbu grafických aplikací pro embedded zařízení umožnující použití na malých diplejích a dotykových obrazovkách. Knihovna je dostupná zdarma bez omezení pro nekomerční a studijní účely. Pro komerční účely

je nutné zakoupit licenci. Podporvané jsou operační systémy Linux, Mac OS X a Win32 a další. Knihovna je napsána v jazyce C a může být použita i v programech napsaných v jazyce C++.[22]

2.3 Koncepce systému PLC-HMI

Rozhraní člověk-stroj se bude skládát ze softwarové části tvořené aplikací, která bude využívat knihovnu Libmodbus pro komunikaci s PLC pomocí protokolu Modbus TCP. Tuto knihovnu jsem vybral, protože disponuje svobodnou licencí, kvalitní dokumentací, je dostupná pro operační systém Linux a procesory architektury ARM. Konkrétně podporuje přímo jednodeskový počítač Raspberry Pi. Další částí softwarového řešení bude grafický framework GLG Toolkit ve verzi Free Community Edition pro vytvoření grafického rozhraní aplikace. Tento framework jsem vybral pro jeho připravenost pro práci v reálném času, podporu operačního systému Linux a procesorů architektur ARM6 a ARM7, konkrétně pak pro podporu jednodeskového počítače Raspberry Pi. Dalšími důvody pro výběr byly bezplatná licence pro nekomerční použití, množství předpřipravených grafických objektů využitelných pro realizaci grafické části rozhraní-člověk stroj a nakonec grafický editor, kterým lze dotvořit další grafické prvky podle potřeby. Aplikace poběží pod operačním systémem Linux, jenž je široce podporován řadou jednodeskových počítačů a výše vybranými knihovnami. Úkolem aplikace napsané v jazyce C++ bude provádět načítání dat z PLC, zobrazování načtených dat v grafickém režimu na displeji a odesílání zásahů operátora do PLC. Návrh grafické části aplikace bude vycházet z doporučení pro návrh vysoce výkoného HMI kapitola 2.1.

Hardwarová část bude realizována pomocí PC, na kterém poběží vytvořená aplikace. Klasické PC by však bylo pro tento účel příliš drahé, rozměrné a zbytečně výkonné. Ideálním řešením bude použít jednodeskový PC. V rámci této práce jsem vyhledal nejvíce prodávané jednodeskové PC a jejich nejdůležitější parametry jsou shrnuty ve srovnávací tabulce 2.2. Po zvážení parametrů z tabulky 2.2 jsem se rozhodl, že HMI aplikace poběží na jednodeskovém minipočítači Raspberry Pi 2. Důvodů pro výběr Raspberry Pi 2 bylo několik. Konkurenční minipočítače za stejnou cenu disponují zhruba stejným hardwarovým vybavením, Raspberry Pi 2 však disponuje největší komunitou uživatelů a díky tomu kvalitní dokumentací, množstvím návodů a podporou řadou softwarových a hardwarových prostředů. V tabulce 2.2 se vyskytují ještě výkonnější modely, ty jsou však pro danou HMI aplikaci zbytečně výkonné a rovněž drahé. Raspberry Pi 3 jsem nevybral i přesto, že jeho cena je stejná jako cena Raspberry Pi 2. Důvodem je novější architektura procesoru ARM Cortex A53, kterou nepodporuje výše vybraný grafický framework. Raspberry Pi 2

PC	Procesor	Operační paměť	Ethernet	Cena
Raspberry Pi	ARM Cortex A6 700 MHz	256-512 MB	10/100	850 Kč
Raspberry Pi 2	ARM Cortex A7 900 MHz	1 GB	10/100	1000 Kč
Raspberry Pi 3	64 bitový ARM Cortex A53 1,2 GHz	1 GB	10/100	1000 Kč
BP-A20 M1	ARM cortex A7 1 GHz	1 GB	10/100/1000	1000 Kč
BPI-M2 Berry	ARM cortex A7	1 GB	10/100/1000	1400 Kč
BPI-M3	ARM cortex A7 1.8 GHz	2 GB	10/100/1000	2500 Kč
Orange Pi One	ARM cortex A7	512MB	10/100	800 Kč
Orange Pi Plus 2	ARM cortex A7	2GB	10/100/1000	1800 Kč
BeagleBone Black	ARM cortex A-8 1 GHz	512MB	10/100	1000 Kč

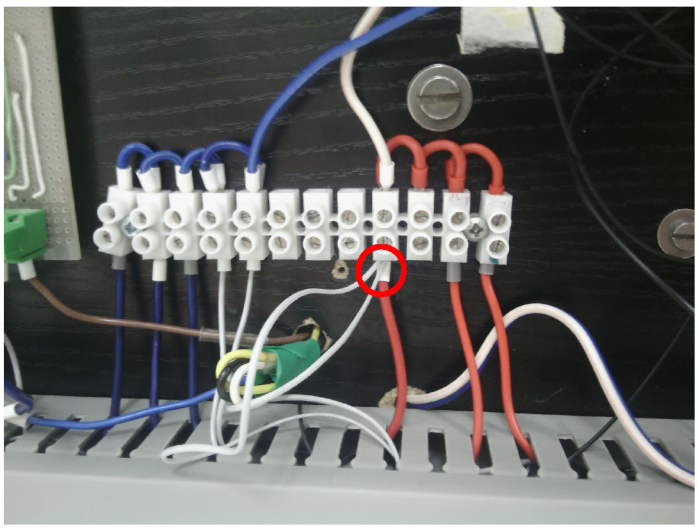
Tab. 2.2: Srovnávací tabulka jednodeskových PC

bude doplněn dotykovým displejem Raspberry Pi Touchscreen zajišťující interakci s uživatelem.

3 Řešení

3.1 Napojení modelu Kmeny na PLC PFC200

Napojení jsem provedl dle výstupního LPT konektoru modelu Kmeny, pouze bylo potřeba vyvést navíc ještě napájení pro PLC. To jsem realizoval za použití dvou nevyužitých rezervních pinů 6 a 19 jenž jsem vodiči připojil na svorkovnici napájení podle obrázku 3.1.



Obr. 3.1: Vyvedení napájení z modelu Kmeny

3.2 Vytvoření vzorové úlohy pro řízení modelu Kmeny

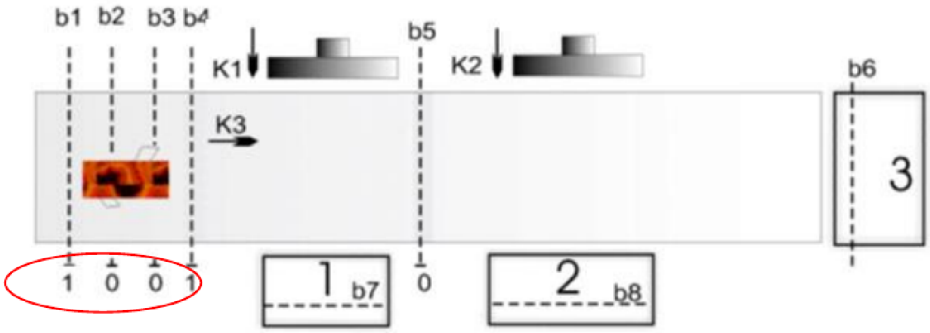
3.2.1 Zadání

1.Klády tří různých velikostí přijíždějí po dopravníku. Velikost klád se rozlišuje soustavou čtyř čidel b1, b2, b3 a b4. Krátké klády se zatlačí manipulátorem K1 do boxu 1. Manipulátor K1 se ovládá 2 s dlouhým impulsem tak, aby kládu zachytil ve správné době. Po spadnutí klády do boxu 1 (indikováno b7) je možné očekávat další kládu. Střední klády se zatlačí manipulátorem K2 do boxu.

2. Manipulátor se ovládá 2 s dlouhým impulsem, se zpožděním od čidla b5 tak, aby kládu zachytil ve správné době. Po spadnutí klády do boxu 2 (indikováno b8) je možné očekávat další kládu. Dlouhé klády se nechají dojet na konec pásu, po odjetí z čidla b6 jsou v boxu a lze očekávat další kládu.

3. Pomocí čítačů počítejte klady jednotlivých boxech.

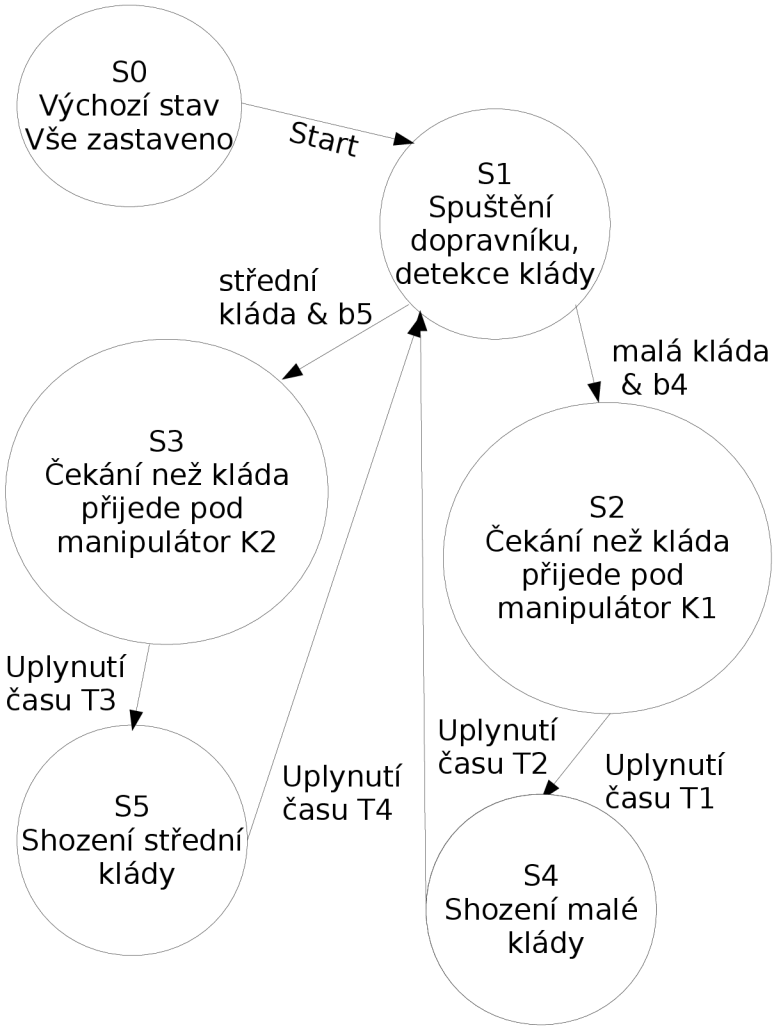
4. Systém se uvede do chodu stisknutím tlačítka START, zastavit lze v libovolném okamžiku tlačítkem STOP.Na indikačních světlech zobrazujte možnost položení další klády (červená - kláda je na pásu, zelená - je možné položit další kládu).[7]s. 34



Obr. 3.2: Schéma modelu Kmeny [7]

3.2.2 Stavový diagram

Program jsem vytvořil podle stavového diagramu na obrázku 3.3. Program má ještě další funkce, které nejsou z důvodu přehlednosti v diagramu obsažené. Po stisknutí tlačítka stop přechází program z jakéhokoliv stavu do stavu S0, kde zastavuje dopravník a uvádí oba manipulátory do výchozího stavu. Dále program počítá klády padající do jednotlivých boxů pomocí sensozrů B6, B7 a B8 pokud se nenachází ve stavu S0.



Obr. 3.3: Stavový diagram

3.3 Propojení Raspberry Pi 2 a Raspberry Pi Touchscreen

Vedení signálů mezi Raspberry Pi 2 a dotykovým displejem je zajištěno pomocí DSI kabelu. Napájení lze provést několika různými způsoby.

1. Napájet samostaně Raspberry Pi 2 ze zdroje pomocí micro USB konektoru a zároveň napájet samostaně dotykový displej z vlastního zdroje rovněž pomocí micro USB konektoru.
2. Napájet Raspberry Pi 2 ze zdroje připojeného pomocí micro USB konektoru a displej napájet z Raspberry Pi 2 pomocí drátových propojek.
3. Napájet displej ze zdroje pomocí micro USB konektoru a Raspberry Pi 2 napájet z displeje pomocí drátových propojek.

Protože je k napájení Raspberry Pi 2 s dotykovým displejem doporučován zdroj schopný dodat až 2,5 A a já jsem měl k dispozi dva zdroje s maximálním proudem 1 A tak jsem zvolil možnost první.

3.4 Instalace softwaru do Raspberry Pi

Po spojení Raspberry Pi 2 s dotykovým displejem jsem se přesunul k softwarové stránce věci. V první řadě je nutno do Raspberry Pi nahrát operační systém. Po instalaci operačního systému budou nainstalovány vybrané softwarové nástroje. Než se pustím do samotného vývoje aplikace je třeba ještě vyřešit, jak a čím bude aplikace překládána.

3.4.1 Nahrání operačního systému do Raspberry Pi 2

Jako operační sytém jsem vybral výchozí systém pro Raspberry Pi 2 a to Raspbian ve verzi Stretch. Z oficiálních stránek `https://www.raspberrypi.org/downloads/raspbian/` jsem stáhl obraz systému a ten následně nahrál na paměťovou kartu, kterou jsem umístil do Raspberry Pi. Následně po nabootování jsem provedl upgrade celého systému na nejnovější verzi.

3.4.2 Instalace GLG Toolkit

Instalace GLG Toolkitu je velmi jednoduchá. Z webových stránek `http://www.genlogic.com/select_platform.html` jsem stáhl příslušnou verzi Linux ARM7. A rozbalil do požadovaného umístění.

3.4.3 Instalace knihoven libmodbus

Instalaci knihoven libmodbus lze provést dvěmi způsoby buď stažením zdrojových kódů z webových stránek `http://libmodbus.org/download/` a následnou kompilací. Nebo instalací balíčků libmodbus-dev a libmodbus5 dostupných pro Raspbian pomocí nástroje aptitude, kterou jsem použil já.

3.4.4 Překlad a spuštění HMI aplikace

K překladu zdrojových kódů aplikace jsem využil generický makefile umístěný ve složce src nainstalovaného GLG Toolkitu a upravil ho pro své potřeby zejména přidáním knihoven, se kterými se má aplikace linkovat a přidáním cest, kde hledat knihovny a hlavičkové soubory. Dále jsem do projektu přidal soubor GlgClass.cpp nacházející se rovněž ve složce src nainstalovaného GLG Toolkitu, jenž obsahuje třídy a metody pro inicializaci a manipulaci s grafikou. Překlad jsem realizoval programem make ve verzi 4.1 a překladačem g++ ve verzi 6.3.0.

Vzhledem k tomu, že se GLG Toolkit snaží pro zobrazení použít primárně OpenGL. Zatímco ovladač pro OpenGL je pro Raspberry Pi pouze v experimentální verzi. Tak jsem aplikaci spouštěl s parametrem -glg-disable-opengl, který OpenGL deaktivuje a pro vykreslení grafiky použije standartní X Window.

3.5 Návrh HMI

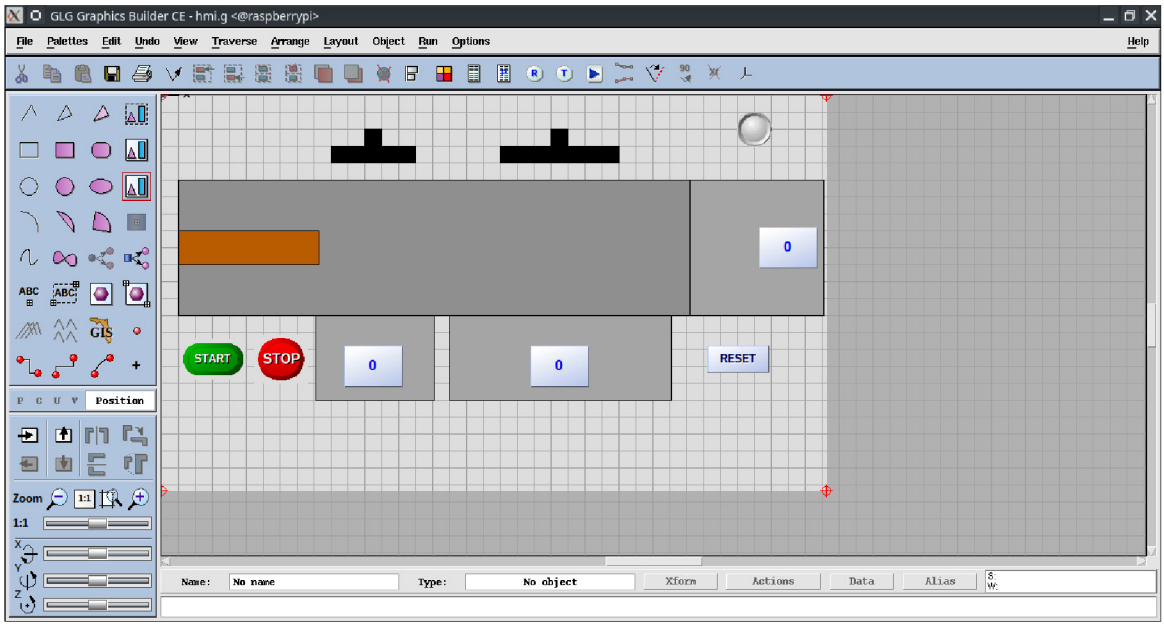
HMI jsem vytvořil ze dvou částí. První částí je program napsaný jazyce C++, který má za úkol aplikaci řídit. Druhou částí je grafická část, kterou jsem vytvořil pomocí programu GLG Graphics Builder, jenž je součástí použitého frameworku GLG Toolkit.

3.5.1 Grafická část

GLG Graphics Builder

Program GLG Graphics Builder slouží pro tvorbu vlastních widgetů a nalézá se uvnitř složky bin, která je součástí nainstalovaného GLG Toolkitu. Základem pro zobrazení je vždy objekt typu viewport s názvem $Widget, který je automaticky vytvořen při volbě File->New->Widget. Každý objekt má své základní parametry, které lze nastavit v okně Properties, jenž lze vyvolat kliknutím pravým tlačítkem na objekt a zvolením volby Properties nebo tlačítkem Properties na horní nástrojové liště při označeném objektu. Další parametry objektů lze nalézt v okně Resources dostupné po kliknutí pravého tlačítka na objekt a volby Resources nebo kliknutím na tlačítko Resources v horní nástrojové liště po označení objektu levým tlačítkem myši. Parametry dostupné v okně Resources se od parametrů dostupných v okně Properties liší tím, že parametry dostupné v okně Resources lze měnit za běhu programu a tím docílit požadovaných grafických změn. Objekty, které se mají zobrazovat musí být umístěny uvnitř objektu typu viewport s názvem $Widget. Pokud chceme umisťovat objekty do objektu $Widget je nutné do něho vstoupit. Do objektu lze vstoupit

pomocí tlačítka se šipkou dolů a vystoupit tlačítkem se šipkou nahoru. Okno programu GLG Graphics Builder je možno vidět na obrázku 3.4, který zobrazuje situaci uvnitř objektu $Widget a je možno do něho přidávat další objekty a tyto objekty upravovat. V levé části programu se nachází nástroje pro tvorbu vlastních objektů. Rovněž je možné přidávat objekty už hotové z menu Palettes mezi něž patří tlačítka, zobrazovače hodnot, grafy, signalizace, přepínače, tanky, ventily, motory a řadu dalších. Výstupem z programu je poté soubor s příponou g jenž obsahuje kompletní grafiku aplikace s možností měnit její parametry z běžícího programu. Tutorial pro práci s programem GLG Graphics Builder lze nalézt v [4] a podrobné informace lze nalézt v [5].



Obr. 3.4: GLG Graphics Builder

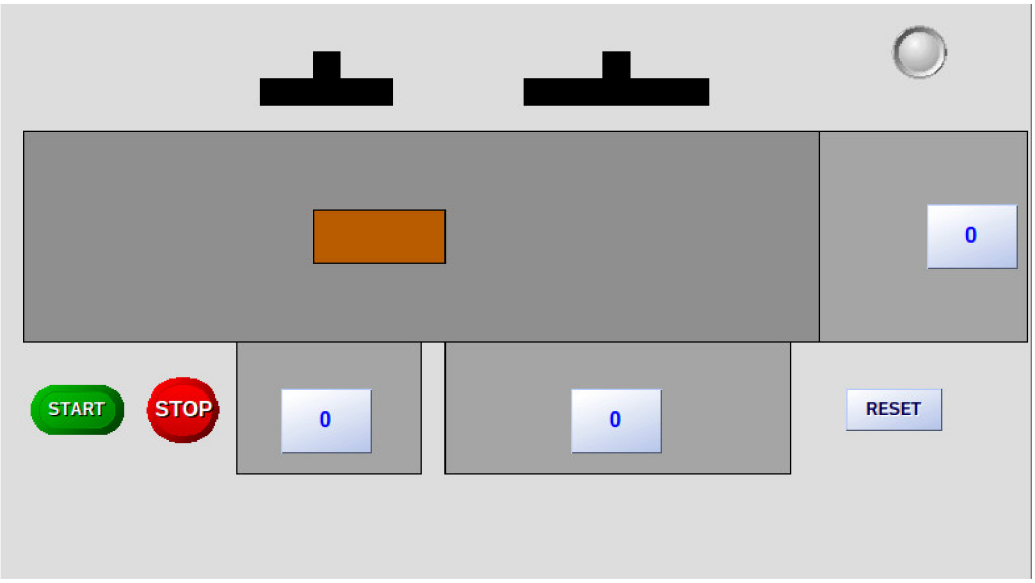
Navržená grafika

Navrženou grafickou část lze vidět na obrázku 3.5. Při návrhu grafické části HMI jsem se držel pravidel a doporučení uvedených v kapitole 2.1. Mým základním cílem bylo, aby výsledná grafická stránka byla co nejpřehlednější a nejjednoduší. Pro pozadí jsem zvolil světle šedou barvu, jak je doporučováno v kapitole 2.1.4. Většina ostatních prvků jako dopravník a boxy pro klády jsou rovněž v odstínech šedé jak radí kapitola 2.1.4.

Pro ukazatele počtu kusů klád v boxech jsem rovněž zvolil barvu šedou. Pro zobrazení hodnot jsem pak použil barvu modrou jak je doporučeno v kapitole 2.1.6.

Pro manipulátory jsem zvolil barvu černou, abych je nepatrně zdůraznil a lépe zviditelnil vzhledem k tomu, že se jedná o prvek, který se po obrazovce rychle pohybuje. Pro klády jsem zvolil barvu hnědou. Není to však proto, že by hnědá barva měla symbolizovat, že je kláda ze dřeva, což není ani doporučováno. Ale hnědou barvu jsem zvolil proto, abych kládu jako ústřední objekt celého HMI zdůraznil a lépe zviditelnil. Nechtěl jsem však zase, aby byla zviditelněna příliš, protože dle kapitoly 2.1.4 by se měly barvy využívat jen pro zdůraznění mimořádných jevů, zatímco kláda pohybující se po dopravníku je pro tuto úlohu jevem zcela běžným. Proto jsem zvolil barvu méně nápadnou tedy hnědou.

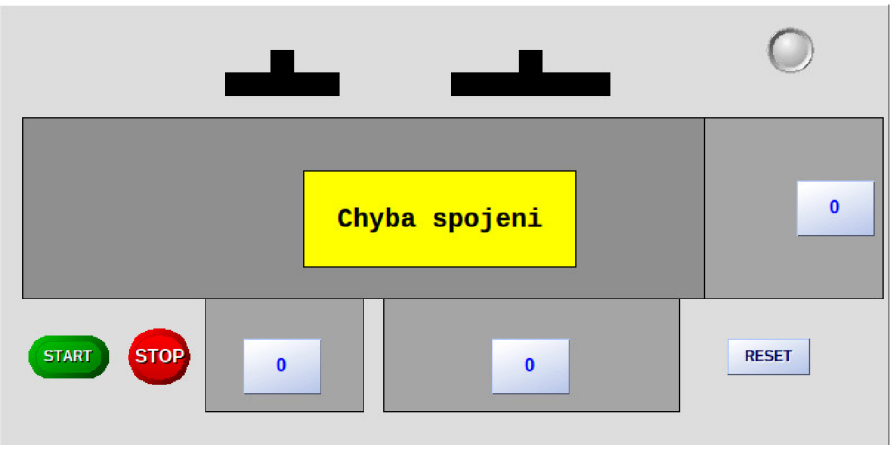
Dále je v HMI aplikaci přítomna signalizace. Pokud je proces zastaven má signalizace barvu šedou. V případě, že je proces spuštěn, změní se barva na zelenou, pokud je možno na pás položit kládu nebo na barvu červenou, pokud je už kláda zpracovávána a je nutno s položením další klády počkat. Pro tlačítko Reset jsem opět zvolil barvu z odstínu šedé vzhledem k tomu, že se jedná o tlačítko bez mimořádné funkce. Pro tlačítko Start jsem zvolil barvu zelenou hlavně proto, že je to barva pro toto tlačítko pevně vžitá. Použití vžité a standartizované barvy tak činí HMI přehlednější, intuitivnější a zmenšuje šanci, že operátor udělá v rozhodujícím okamžiku chybu. Rovněž pro tlačítko Stop jsem ze stejných důvodů zvolil barvu červenou. Navíc díky barevnému ladění HMI aplikace tvoří tlačítko Stop se svou jasně červenou barvou nejvýraznější prvek. V případě problému je tedy velmi snadné ho najít a použít, což zvyšuje bezpečnost a ovladatelnost.



Obr. 3.5: HMI aplikace

V případě, že při spuštění nelze navázat spojení s PLC nebo dojde při běhu ke

ztrátě spojení objeví se uprostřed displeje oznámení s blikajícím žlutým pozadím jak lze vidět na obrázku 3.6 a dojde k zastavení klády a vrácení manipulátorů do výchozích pozic. Žlutou barvu pro pozadí jsem vybral, protože není jinak v aplikaci jinde přítomna a rovněž je doporučována jako barva vhodná pro alarmy dle kapitoly 2.1.7. Komunikaci lze restartovat tlačítkem Reset. Po stisknutí tlačítka Reset aplikace zavře spojení a pokusí se otevřít nové. V případě úspěchu hláška zmizí a aplikace pokračuje v činnosti. Vzhledem k tomu, že kláda není detekována neustále, ale pouze když projíždí kolem senzorů tak nelze ihned zobrazit její správnou pozici. Správná pozice klády je zobrazována až od nové klády na dopravníku.



Obr. 3.6: HMI aplikace - Chyba spojeni

3.5.2 Programová část

Činost HMI aplikace je znázorněna na diagramu 3.7. Pro vykreslování grafiky program používá GLG Toolkit jak bylo navrženo v konceptu v kapitole 2.3. Nejprve dojde k inicializaci vytvořením objektu GlgSessionC. Následně je vytvořena třída hmi, jenž v sobě obsahuje metody a data pro manipulaci s grafikou i čtení dat pomocí Modbus protokolu. Některé získává děděním třídy GlgObjectC, některé využívají funkce z knihovny Libmodbus, jenž jsem pouze obalil kódem pro detekci chyb a zapouzdřil do třídy a zbytek jsem doprogramoval sám. První použitou metodou z GLG Toolkitu je LoadWidget, jenž načte widget vytvořený v programu GLG Graphics Builder. Dále jsou hojně využívány metody GetResource a SetResource, které zajišťují čtení parametrů z grafické části a zapisování parametrů do grafické části. Metoda InitialDraw provede první vykreslení grafiky na displej. Následně je spuštěn časovač, jenž spouští funkci načítající data z PLC a překreslování grafiky na displeji. Nakonci funkce je znovu spuštěn časovač, který funkci po 10 ms zavolá znovu. Doba

10 ms je poměrně malá, byla však nutná, protože manipulátory se pohybují velice rychle a s vyšší hodnotou nebylo možné jejich pohyb kvalitně zobrazit. Podrobnější popis použitých metod GLG Toolkitu včetně parametrů je umístěn v příloze A.

Z knihoven Libmodbus jsem v programu použil funkce

- *modbus_t *modbus_new_tcp(const char *ip, int port)* - Alokuje strukturu modbus_t a inicializuje v ní parametry pro komunikaci.
- *modbus_connect(modbus_t *ctx)* - Vytvoří spojení s parametry ze struktury modbus_t.
- *int modbus_read_bits(modbus_t *ctx, int addr, int nb, uint8_t *dest)* - Čte cívky pomocí Modbus kódu 0x01 a ukládá je do pole prvků uint8_t. Každý bit zabere jeden uint8_t.
- *int modbus_read_input_bits(modbus_t *ctx, int addr, int nb, uint8_t *dest)* - Čte diskrétní vstupy pomocí Modbus kódu 0x02 a ukládá je do pole prvků uint8_t.
- *int modbus_read_registers(modbus_t *ctx, int addr, int nb, uint16_t *dest)* - Čte uchovávací registry pomocí Modbus kódu 0x03 a ukládá je do pole prvků uint16_t.
- *int modbus_write_bit(modbus_t *ctx, int addr, int status)* - Zapíše hodnotu do jedné cívky pomocí Modbus kódu 0x05.
- *void modbus_close(modbus_t *ctx)* - Ukončí spojení.
- *void modbus_free(modbus_t *ctx)* - Dealokuje strukturu modbus_t.[17]

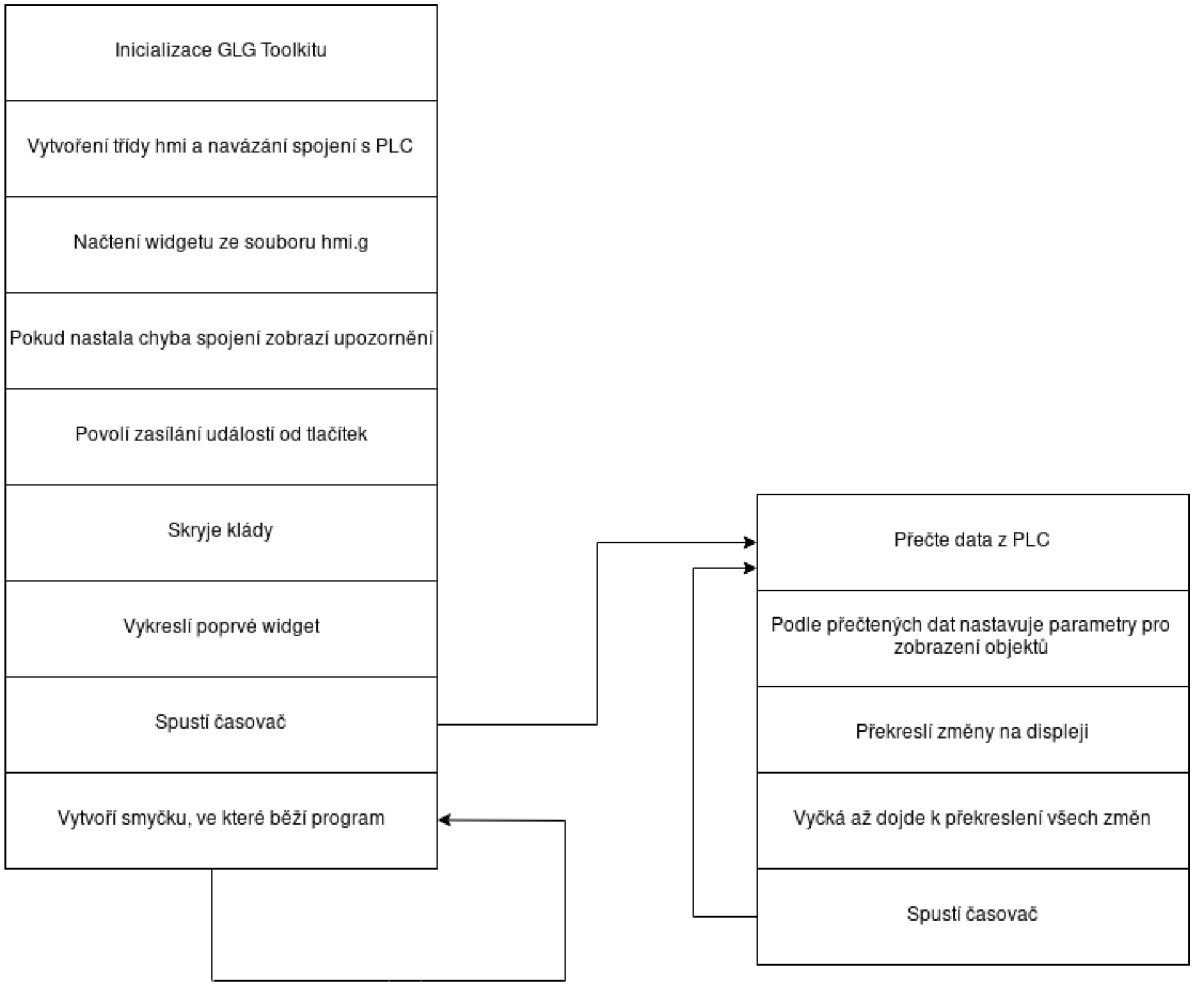
3.6 Zhodnocení vlastností

Tato kapitola se zaměřuje na vyhodnocení vlastností HMI aplikace. Především se zabývá, jestli má vybrané Raspberry Pi 2 dostatek systémových prostředků pro běh HMI aplikace. Dále jsem měřil přesnost časovače a jeho jitter. Nakonec jsem změřil dobu programové smyčky HMI aplikace.

3.6.1 Systémové prostředky

Procesor

Vytížení procesoru jsem měřil standartním linuxovým nástrojem top. Vytížení procesoru HMI aplikací bylo dáno procesy hmi a Xorg. Bez spuštěné HMI aplikace byl procesor vytížen celkově 8,7% a proces Xorg vytěžoval procesor 1%. Při běžící HMI aplikaci byl procesor celkově vytížen 68,2%, hmi aplikace vytěžovala procesor 9,6% a proces Xorg 53,8%. Z výsledků vyplývá, že z hlediska vytížení procesoru zvládá Raspberry Pi 2 chod aplikace dobře a zůstáva ještě zhruba 30% rezerva. Navíc pro



Obr. 3.7: Diagram programu

aplikaci byla použita nízká perioda překreslování. V případě vyšší periody překreslování by bylo vytížení procesoru menší.

Operační paměť

K měření paměti spotřebované procesy hmi a Xorg jsem opět použil nástroj top. Bez spuštěné aplikace byla celková spotřeba paměti 75 MiB proces Xorg zabíral 38 MiB. Po spuštění aplikace vzrostla celková spotřeba paměti na 82 MiB, hmi aplikace využívala 10 MiB a proces Xorg 40 MiB. Vzhlem ktomu, že Raspberry Pi 2 disponuje 1GiB operační paměti je využití paměti HMI aplikací zanedbatelné.

3.6.2 Měření přestnosti a jitteru časovače

Dobu běhu časovače jsem určil měřením času mezi voláním funkce spouštěné časovačem s nastaveným časem 10 ms. Funkce zjištuje čas voláním funkce `clock_gettime`,

která využívá časovač `CLOCK_MONOTONIC`. Ten je následně odečten od času získaného při minulém volání a výsledkem je tedy uplynulý čas mezi voláním těchto dvou funkcí a tedy čas časovače. Čas je měřen v nanosekundách. Pro výpočty jsem změřil 1000 hodnot. Z těchto hodnot jsem vypočetl nejistotu typu A, následně nejistotu rozšířenou a nakonec jitter.

Výpočet nejistot

Nejistoty jsem vypočítal podle standartních vzorců

$$u_A = k_s \cdot s_{\overline{x}} \tag{3.1}$$

$$s_{\overline{x}} = \sqrt{\frac{1}{n \cdot (n-1)} \sum_{i=1}^{n} (x_i - \overline{x})^2} \tag{3.2}$$

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} \tag{3.3}$$

$$u_C = \sqrt{u_A^2 + u_B^2} \tag{3.4}$$

$$U = k_r \cdot u_C \tag{3.5}$$

Průměrná hodnota doby časovače vypočtené podle vzorce 3.3 je 10,102 ms.
Nejistotu typu A jsem vypočetl podle vzorce 3.1 kde jsem k_s volil 10 a vyšla 31,263 μs.
Rozšířená nejistota vypočtená dle vzorce 3.5 kde jsem za k_r dosadil 2 abych získal interval s 95% pravděpodobností vyšla 62,526 μs.
Změřená doba časovače je tedy 10,102±0,063 ms.

Výpočet jitteru

$$jitter = |změřená\ doba\ časovače - nastavená\ doba\ časovače| \tag{3.6}$$

Jitter jsem vypočítal dle vzorce 3.6 a vyšel 102 μs.

3.6.3 Měření doby programové smyčky

Programovou smyčkou je myšlena funkce zajišťují čtení dat z PLC a překreslování grafiky na displeji graficky znázorněná vpravo na diagramu na obrázku 3.7. Dobu celé programové smyčky jsem měřil obdobným způsobem jako dobu běhu časovače. Na začátku jsem vždy získal čas pomocí funkce `clock_gettime` využívající časovač

`CLOCK_MONOTONIC`. A náledně opět odečetl od času získaného při minulém volání a výsledkem byl čas celé programové smyčky tedy čas časovače nastaveného na 10 ms, doba čtení dat z modbusu, doba programového zpracovaní podmínek a čas překreslení grafiky. Čas jsem opět měřil v nanosekundách. Pro výpočety jsem změřil 1000 hodnot. Z těchto hodnot jsem vypočetl nejistotu typu A. A následně nejistotu rozšířenou.

Výpočet nejistot

Průměrná hodnota programové smyčky vypočtená dle vzorce 3.3 činí 37,762 ms.
Nejistotu typu A jsem vypočetl podle vzorce 3.1 kde jsem k_s volil 10 a vyšla 114,195 μs.
Rozšířená nejistota vypočtená dle vzorce 3.5 kde jsem za k_r dosadil 2 abych získal interval s 95% pravděpodobností vyšla 228,389 μs.
Změřená doba programové smyčky tedy vyšla 37,762±0,228 ms.

4 Závěr

V této práci jsem vytvořil HMI aplikaci. Začal jsem vyhledáním pravidel a doporučení pro tvorbu vysoce výkoného HMI, jenž jsem se snažil uplatnit při tvorbě grafického vzhledu HMI aplikace. Následně jsem po prozkoumání komunikačních možností PLC PFC200 zvolil pro komunikaci protokol Modbus TCP.

Dále jsem vyhledal několik nejdostupnějších jednodeskovým počítačů, z nichž jsem vybral ten nejvhodnější, jimž byl Raspberry Pi 2 model B. Pro interakci s uživatelem jsem Raspberry Pi 2 doplnil dotykovým displejem Raspberry Pi Touchscreen. Následně jsem provedl vyhledání knihoven dostupných s bezplatnou licencí pro nekomerční použití pro účely vykreslování grafiky na displej a komunikaci pomocí vybraného protokolu Modbus TCP. Pro vykreslování grafiky na displej jsem vybral framework GLG Toolkit, jenž je přímo pro vytváření HMI navržen. GLG Toolkit disponuje obsáhlou dokumentací, v níž jsou některé informace hned jasné a jiné je potřeba složitě dohledávat a dovozovat. Nicméně tvorba HMI v něm jde dobře hlavně díky tomu, že je pro konstrukci HMI přímo určen. Rovněž velice pomáhá přiložený program GLG Graphics Builder, s nímž jde pohodlně vytvořit požadovaný vzhled aplikace. Pro komunikaci pomocí protokolu Modbus TCP jse vybral volně dostupnou knihovnu Libmodbus. Knihovna Libmodbus má dobrou dokumentaci a celkově je knihovna navržena velice intuitivně, její použití je snadné a funguje velmi spolehlivě.

V praktické části jsem poté připojil laboratorní model Kmeny k PLC PFC200 a vytvořil program do PLC v prostředí Codesys. Následně jsem propojil Raspberry Pi s dotykovým displejem a nahrál do něho operační systém a nainstaloval vybrané softwarové prostředky. Poté jsem vytvořil grafickou část aplikace pomocí programu GLG Graphics Builder. Při návrhu jsem se snažil řídit pravidly a doporučeními sepsanými v teoretické části práce. Následně jsem vytvořil aplikaci napsanou v jazyce C++, která načte grafiku vytvořenou v programu GLG Graphics Builder a podle dat přijatých z PLC ji zobrazuje. Dále aplikace odchytává události od tlačítek a odesílá do PLC potřebné zásahy. V závěru práce jsem vyhodnotil využití systémových zdrojů aplikací a dospěl jsem k tomu, že Raspberry Pi 2 má dostatek systémových zdrojů pro provozování této aplikace.

Literatura

[1] BILL R. HOLLIFIELD ...[ET AL.]. –. The High performance HMI handbook: a comprehensive guide to designing, implementing and maintaining effective HMIs for industrial plant operations. Houston, TX: Plant Automation Services, 2008. ISBN 978-097-7896-912.

[2] *Přehled protokolu MODBUS* [online]. RONEŠOVÁ, Andrea. [cit. 2017-11-20]. Dostupné z: `http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf`

[3] *MODBUS Messaging on TCP/IP Implementation Guide* [online]. MODBUS-IDA. [cit. 2017-11-20]. Dostupné z: `http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf`

[4] *GLG Builder and Animation Tutorial* [online]. GENERIC LOGIC, INC. [cit. 2018-5-5]. Dostupné z:

[5] *GLG User's Guide and Builder Reference Manual* [online]. Generic Logic [cit. 2018-05-19]. Dostupné z: `http://www.genlogic.com/doc_html/glg.pdf`

[6] *GLG Programming Reference Manual* [online]. Generic Logic [cit. 2018-05-19]. Dostupné z: `http://www.genlogic.com/doc_html/glgpr.pdf`

[7] *Laboratorní cvičení BPPA - II* [online]. JIRGL, Miroslav a Jakub ARM. [cit. 2017-10-25]. Dostupné z: `https://www.vutbr.cz/www_base/priloha.php?dpid=149369`

[8] *PLC – PFC200 Controller* [online]. [cit. 2017-11-3]. Dostupné z: `https://www.wago.com/wagoweb/documentation/750/eng_dat/d07508202_00000000_0en.pdf`

[9] *Raspberry Pi* raspberrypi.org [online]. [cit. 2017-12-04]. Dostupné z: `https://www.raspberrypi.org/`

[10] *Raspberry Pi.* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2017-12-03]. Dostupné z: `https://cs.wikipedia.org/wiki/Raspberry_Pi#Porovn%C3%A1n%C3%AD_jednotliv%C3%BDch_model%C5%AF`

[11] *Banana Pi* [online]. [cit. 2017-12-04]. Dostupné z: `http://www.banana-pi.org`

[12] *Banana Pi* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2017-12-04]. Dostupné z: `https://cs.wikipedia.org/wiki/Banana_Pi#Banana_Pi_-_model_BP-A20_M1`

[13] *Orange Pi* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2017-12-13]. Dostupné z: https://cs.wikipedia.org/wiki/Orange_Pi#Orange_Pi_One

[14] *Orange Pi* : orangepi.org [online]. [cit. 2017-12-15]. Dostupné z: http://www.orangepi.org/

[15] *BeagleBone* : beagleboard.org [online]. [cit. 2017-12-21]. Dostupné z: https://beagleboard.org

[16] *BeagleBoard* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2017-12-22]. Dostupné z: https://en.wikipedia.org/wiki/BeagleBoard

[17] *Libmodbus* Libmodbus.org [online]. [cit. 2017-12-22]. Dostupné z: http://libmodbus.org/documentation/

[18] *FreeMODBUS* Freemodbus.org [online]. [cit. 2017-12-22]. Dostupné z: https://www.freemodbus.org/index.php

[19] *Knihovna Qt* In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2017-12-17]. Dostupné z: https://cs.wikipedia.org/wiki/Qt_(knihovna)

[20] *GTK+* Gtk.org [online]. [cit. 2017-12-17]. Dostupné z: https://www.gtk.org/

[21] *Generic Logic* Genlogic.com [online]. [cit. 2018-01-02]. Dostupné z: http://www.genlogic.com/free_embedded_graphics.html

[22] *µGFX* Ugfx.io [online]. [cit. 2018-01-02]. Dostupné z: https://ugfx.io

Seznam příloh

A Použité prostředky frameworku glg toolkit

- *#include "GlgClass.h"* - Hlavičkový soubor s deklarací používaných tříd.

- *GlgAppContext* - Aplikační kontext získaný při inicializaci.

- *#include "GlgMain.h"* - Definuje počátek programu. Musí být umístěn před funkcí GlgMain.

- *int GlgMain(int argc, char *argv[], GlgAppContext InitAppContext)* - Fuknce, která se vykonává po spuštění programu. První dva parametry jsou jako u standartní funkce main. Třetí parametr je objet typu GlgAppContext, který musí být následně předán funkci provádějící inicializaci.

- *GlgSessionC(GlgBoolean initialized, GlgAppContext application_context,int argc = 0, char ** argv = NULL)* - Třída poskytující rozhraní pro incializaci. Konstruktor provede inicializaci GLG Toolkitu. Prvním parametrem udává jestli už byl GLG Toolkit inicializován. Druhý parametr musí být objekt GlgAppContext z metody GlgMain. Třetí a čtvrtý parametr předávají parametry se kterými byl program spuštěn z funkce GlgMain.

- *GlgObjectC* - Hlavní třída pro práci s grafickými objekty.

- *GlgBoolean LoadWidget(char * filename)* - Metoda třídy GlgObjectC, která načte objekt `$Widget` typu viewport ze souboru předaného parametrem.

- *void EnableCallback(GlgCallbackType callback_type, GlgObject callback_viewport = NULL)* - Povoluje zasílání zpráv od určitých objektů například tlačítek. Prvním parametrem je typ objektů, které mohou zprávy zasílat. Druhý parametr je objekt, kterému mohou být zprávy zasílány. V případě, že je NULL povoluje se zasílání zpráv do widgetu viewport.

- *virtual void Input(GlgObjectC& callback_viewport, GlgObjectC& message)* - Metoda zachytávající zprávy od objektů například tlačítek. Může být ve zděděné třídě přepsána.

- *GlgBoolean SetResource(char * resource_name, double value)* - Metoda třídy

GlgObjectC. Nastaví parametr grafickému objektu. Prvním parametrem funkce je jméno parametru grafického objektu, který se má nastavit. Řetězec je ve tvaru název objektu/název parametru. Druhým parametrem funkce je hodnota, která se má nastavit.

- *GlgBoolean SetResource(char * resource_name, char * s_value)* - Obdobná metoda s tím rozdílem, že nastavovaný parametr je typu řetezec.

- *GlgBoolean GetResource(char * resource_name, double * value)* - Metoda třídy GlgObjectC. Získa parametr grafického objektu a uloží ho do proměnné.

- *GlgBoolean GetResource(char * resource_name, char ** s_value)* - Stejná metoda jako předchozí s tím rozdílem, že proměnná je řetězec.

- *void InitialDraw(void)* - Metoda třídy GlgObjectC. Vykreslí prvně grafiku na displej.

- *GlgLong GlgAddTimeOut(GlgAppContext app_context, GlgLong interval, GlgTimerProc timer_callback, GlgAnyType client_data)* - Spustí časovač. Prvním parametrem je kontext aplikace. Druhým parametrem je čas časovače. Třetím parametrem je funkce, která se má zavolat až časovač vyprší. Čtvrtým parametrem jsou data, která jsou předána funkci ze třetího parametru jako parametr.

- *GlgLong GlgMainLoop(GlgAppContext app_context)* - Vytvoří nekonečnou smyčku. Parametrem je kontext aplikace.

- *GlgBoolean Update(void)* - Metoda třídy GlgObjectC. Překreslí změny v grafice.

- *GlgBoolean Sync(void)* - Metoda třídy GlgObjectC. Přinutí vykreslit všechny požadavky pro překreslení z funkce Update a počká než jsou dokončeny.[6]

B Obsah přiloženého CD

- /
  - `Bakalarska-prace.pdf` ..... Elektronická verze Bakalářské práce
  - `program-plc` ..... Program pro řízení modelu Kmeny
  - `hmi` ..... HMI aplikace
    - `main.cpp` ..... Hlavní část aplikace
    - `hmi.cpp` ..... Obsahuje metody třídy hmi
    - `hmi.h` ... Obsahuje definici třídy hmi a nastavují se v něm adresy pro Modbus
    - `hmi.g` .... Grafická část aplikace vytvořená v programu GLG Graphics Builder
    - `makefile` ..... Soubor make pro překlad aplikace
  - `mereni-casovac.txt` ..... Naměřená data časovače
  - `mereni-program.txt` ..... Naměřená data programové smyčky