UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2022 Martin Obolecký

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

MODEL REÁLNÉHO ZAŘÍZENÍ REALIZOVANÝ S VYUŽITÍM MIKROKONTROLÉRU

Martin Obolecký

Bakalářská práce

2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Obolecký**
Osobní číslo:	**I19035**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Téma práce:	**Model reálného zařízení realizovaný s využitím mikrokontroléru**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je navrhnout laboratorní modul, který bude možné použít k simulaci chování reálného zařízení – technologického procesu. Jádrem modulu bude vybraný mikrokontrolér (např. ATmega od spol. Atmel). Modul bude připojitelný k PLC automatu, který bude použit k jeho řízení.
Teoretická část: Stručná rešerše problematiky týkající se řízení technologických procesů s využitím PLC, popis problematiky řízení s využitím mikrokontroléru, vč. možností jeho programování.
Implementační část: Realizace modulu simulujícího reálný proces s využitím mikrokontroléru, návrh podpůrných obvodů umožňujících připojení tlačítek, LED diod a displeje. Tvorba ukázkové laboratorní úlohy, programování řídicí aplikace pro PLC, zpracování technické dokumentace.

Rozsah pracovní zprávy: **cca 40 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

MALÝ, M. 2019. Porty, bajty, osmibity: počítače na koleni. Nakladatelství CZ.NIC. ISBN 978-80-88168-39-3.
BRTNÍK, B.; MATOUŠEK, D. 2011. Mikroprocesorová technika. BEN – technická literatura. ISBN 978-80-7300-406-4.
ŠMEJKAL, L.; MARTINÁSKOVÁ, M. 2002. *PLC a automatizace 1: základní pojmy, úvod do programování*. Praha: BEN – technická literatura. ISBN 80-86056-58-9.
ŠMEJKAL, L. 2005. *PLC a automatizace 2: sekvenční logické systémy a základy fuzzy logiky*. Praha: BEN – technická literatura. ISBN 80-7300-087-3.

Vedoucí bakalářské práce: **Ing. Libor Kupka, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **17. prosince 2021**
Termín odevzdání bakalářské práce: **13. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Daniel Honc, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 7. ledna 2022

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne

Martin Obolecký

Poděkování

Chtěl bych poděkovat vedoucímu mé práce, Ing. Liborovi Kupkovi, Ph.D. za pomoc a trpělivost při řešení mé práce. Rád bych poděkoval i Ing. Zdeňkovi Oboleckému za poskytnutí modulů a za podporu při řešení problémů a také děkuji své rodině a přátelům za podporu po celou dobu mého studia.

V Pardubicích dne

Martin Obolecký

ANOTACE

Cílem této práce je vytvořit komunikaci mezi více moduly pro ovládání reálného zařízení. Jako komunikační modul může být využit PLC nebo jiný mikrokontrolér. Bylo třeba naprogramovat tyto moduly tak, aby výstupem bylo možné ovládat a kontrolovat stav reálného zařízení.

KLÍČOVÁ SLOVA

PLC, AMiT, mikrokontrolér, komunikace, RS485, DetStudio, Modbus RTU

TITLE

MICROCONTROLLER BASED REAL DEVICE MODEL

ANNOTATION

The goal of this work is to create communication between multiple modules for controlling a real device. A PLC or other microcontroller can be used as a communication module. Load the program into these modules so that the output can control and monitor the state of the real device.

KEYWORDS

PLC, AMiT, microcontroller, communication, RS485, DetStudio, Modbus RTU

OBSAH

Seznam ilustrací 10

Seznam tabulek 12

Seznam zkratek a značek 13

Úvod 14

1 Mikrokontrolér 15

1.1 Architektura mikrokontroléru 15

1.1.1 Rozdělení 15

1.1.2 Mikrokontroléry AMiT 16

1.2 Periferie mikrokontroléru 17

1.2.1 Komunikace RS232 17

1.2.2 Ethernet 18

1.2.3 ADC převodník 20

1.2.3 Přerušení (Interrupt) 21

1.2.4 Časovače a čítače 22

1.2.5 Watchdog časovač 23

1.3 Historie a vlastnosti PLC 24

2 Komunikace a protokoly 27

2.1 Komunikační protokol RS485 28

2.2 Výhody RS485 a RS422 proti RS232 28

2.3 Modbus protokol 29

2.3.1 Modbus ASCII 29

2.3.2 Modbus RTU 30

2.3.3 Modbus rámcování 30

2.4 Zakončovací rezistory 32

2.5 CAN protokol 33

3 Hardware 35

3.1 Napájecí zdroj AD1048-24FS 35
3.2 Jistič 35
3.3 Spínače 37
4 DetStudio202 37
4.1 Aplikační prostředí 38
4.2 Identifikace modulu 40
4.3 Test programu pro blikání LED 41
4.3.1 IO Komunikace 42
4.3.2 Proměnné 42
4.3.3 Aliasy 42
4.3.4 Program blikání 43
4.3.5 Návrh zapojení testu pro blikání LED 44
4.3.6 Inspektor 45
5 Protokol modbus RTU 46
5.1 Stanice Master 46
5.2 Stanice Slave 47
5.3 Sdílení hodnot proměnných Master stanice 47
5.4 Sdílení hodnot proměnných Slave stanice 49
6 Realizace 50
6.1 Konfigurace IO 50
6.2 Proměnné 51
6.3 Aliasy 52
6.4 Obrazovka 54
6.5 Návrh připojení PLC modulů 56
6.6 Kompletace 57
7 Program 57
7.1 Proc00 58

7.2 Podprogram Proc01 59

Závěr 63

Použitá literatura 64

Přílohy 65

SEZNAM ILUSTRACÍ

Obrázek 1.1 – Mikrokontrolér AMiT 15
Obrázek 1.2 – Komunikace RS232 18
Obrázek 1.3 – Ethernet 19
Obrázek 1.4 – 2bitový analogově-digitální převodník 20
Obrázek 1.5 – Čítač 22
Obrázek 1.6 – Watchdog časovač 24
Obrázek 1.7 – PLC Architektura 26
Obrázek 2.1 – komunikace RS485 27
Obrázek 2.2 – CAN uzel (node) 33
Obrázek 3.1 – Zdroj AD1048-24FS 35
Obrázek 3.2 – Jistič Schneider B16 36
Obrázek 3.3 – Tlačítka a přepínače 37
Obrázek 4.1 – Vývojové prostředí DetStudio202 38
Obrázek 4.3 – Výběr modulu AMiNi4DS 39
Obrázek 4.2 – Výběr modulu AMiNi4DW2/G 39
Obrázek 4.4 – Parametry projektu 40
Obrázek 4.5 – Identifikace zařízení 41
Obrázek 4.6 – IO Konfigurace 42
Obrázek 4.7 – Proměnné 42
Obrázek 4.8 – Aliasy 43
Obrázek 4.9 – Program pro blikání LED 43
Obrázek 4.11 – Schéma zapojení pro blikání LED 44
Obrázek 4.10 – Inspektor DetStudio 45
Obrázek 5.1 – Vlastnosti Master stanice 46
Obrázek 5.2 – Vlastnosti Slave stanice 47
Obrázek 5.3 – Nastavení master komunikace 48
Obrázek 5.4 – Inspektor funkční komunikace 48
Obrázek 5.5 – Komunikace holding registru 48
Obrázek 5.7 – Holding registr Slave stanice 49
Obrázek 5.6 – Proměnné Slave stanice 49
Obrázek 6.1 – IO Konfigurace dopravníku (master) 50
Obrázek 6.2 – IO Konfigurace dopravníku (slave) 51

Obrázek 6.3 – Proměnné programu (slave) 51
Obrázek 6.4 – Proměnné programu (master) 52
Obrázek 6.5 – Aliasy stanice master 53
Obrázek 6.6 – Aliasy stanice slave 54
Obrázek 6.7 – Obrazovka master stanice 55
Obrázek 6.8 – Nečinný (Idle) proces pro obrazovku 55
Obrázek 6.9 – Schéma dopravníku s PLC moduly 56
Obrázek 6.10 – Vývoj zapojení 57
Obrázek 6.11 – Kompletace rozvaděče 57
Obrázek 7.2 – Program Proc00 58
Obrázek 7.1 – Komunikace programu 58
Obrázek 7.3 – Klidové stavy Proc01 59
Obrázek 7.4 – Start/Stop tlačítko Proc01 59
Obrázek 7.5 – Rychlost pohybu Proc01 60
Obrázek 7.6 – Dojezdy dopravníku 60
Obrázek 7.7 – Směr pohybu a pohyb dopravníku 61
Obrázek 7.8 – Alarm texty obrazovky 61
Obrázek 7.9 – Změna směru pohybu dopravníku 62

SEZNAM TABULEK

Tabulka 1.1 – Řídicí terminály a kompaktní systémy firmy AMiT (AMiT, 2022) 16

Tabulka 1.1 – Řídicí terminály a kompaktní systémy firmy AMiT – pokračování 17

Tabulka 1.2 – 2bitový A/D převodník 21

Tabulka 2.1 – ¨Rámec zprávy Modbus ASCII, (Modicon, 1996) 31

Tabulka 2.2 – Rámec zprávy Modbus RTU, (Modicon, 1996) 32

Tabulka 4.1 – Seznam součástek pro blikání LED 44

Tabulka 6.1 – Seznam součástek pro pásový dopravník 56

SEZNAM ZKRATEK A ZNAČEK

CPU	centrální procesorová jednotka
EEPROM	elektricky mazatelná paměť pouze pro čtení
IO	vstup/výstup
LAN	místní síť
LED	elektroluminiscenční dioda
PLC	programovatelný logický automat
PC	osobní počítač
RAM	paměť s náhodným přístupem
ROM	paměť pouze pro čtení
TCP	protokol kontroly přenosu
UDP	uživatelský datagramový protokol

ÚVOD

Toto téma se týká oblasti IP konfigurace, komunikace PLC s PC a synchronizace sdílených dat mezi moduly s následným využitím. Pro řešení bude nutné najít IP adresu mikrokontroléru, abychom mohli navázat spojení a případně upravit IP adresu mikrokontroléru. Po navázání spojení se spojí komunikace prostřednictvím protokolu Modbus RTU.

Jednočipové mikropočítače dnes najdeme v mnoha zařízeních a přístrojích, se kterými se denně běžně setkáváme. Propojení více mikrokontrolérů se v praxi využívá stále častěji, ale je nedostatek lidí, kteří tomuto tématu rozumí nebo se touto oblastí živí. Problematika spočívá už v první komunikaci mikrokontroléru připojeného k počítači, kde je vyžadována znalost v oblasti počítačových sítí.

Mikropočítače řídí kancelářská zařízení, jako jsou digitální telefony, faxy, telefonní ústředny, kopírky a tiskárny. Čtenář této práce se může dozvědět obecné informace o mikrokontrolerech a jak tyto mikrokontroléry spojit a sdílet data.

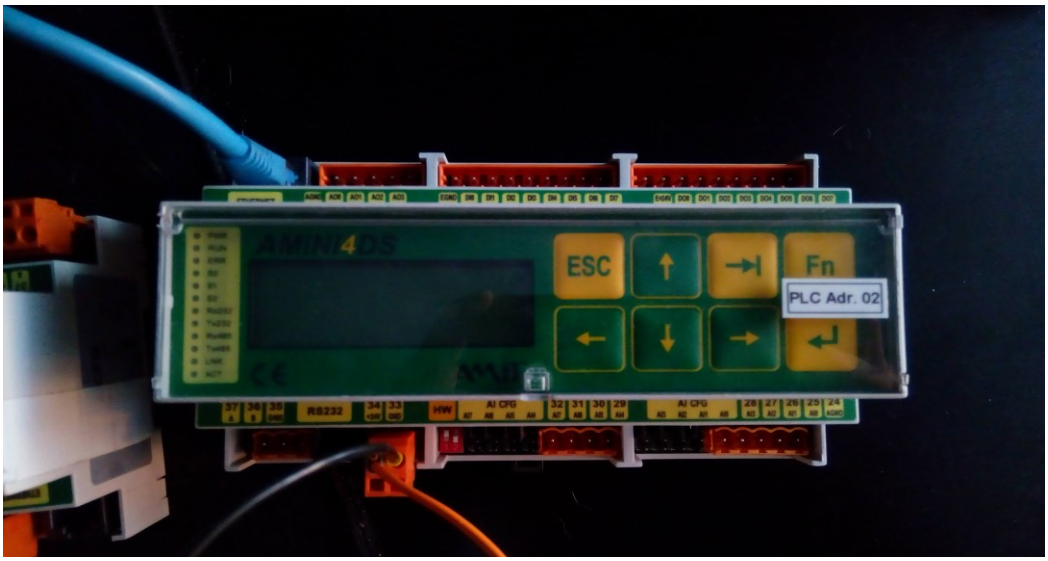
Na závěr je dokázána schopnost komunikace pomocí charakterového zobrazovače na jednom z využitých PLC modulů a ovládání digitálních výstupů a vstupů mezi moduly pomocí komunikace RS485 s využitím protokolu Modbus RTU.

1 MIKROKONTROLÉR

Jedná se o jednočipový počítač. Technologie mikrokontroléru byla vyvinuta po mikroprocesoru a odstraňuje nedostatky mikroprocesoru (Šíl, 2012).

1.1 ARCHITEKTURA MIKROKONTROLÉRU

Mikrokontrolérový čip je úzce spojen s CPU, registry, řídicími jednotkami přerušení, vyhrazenými IO porty a pamětí (RAM a ROM). Paměť programu je typu EPROM nebo Flash. Specifické mikrokontroléry pak využívají paměť typu ROM nebo doplněné o nevolatilní paměť EEPROM. Mikrokontrolér je samostatná jednotka a nevyžaduje další bloky pro jeho funkčnost. Mikrokontrolér je obecně užíván pro operace určené na aplikace. Obvykle mikrokontroléry obsahují generátor hodinového signálu nebo kontrolní obvody správné funkčnosti. V porovnání s mikroprocesorem je mikrokontrolér energicky výhodnější i cenově dostupnější. Vyrábí se v různých velikostech od 3×3 mm s osmi vývody až po typy se 100 až 200 vývody (Gadget-info.com, 2019).



Obrázek 1.1 – Mikrokontrolér AMiT

1.1.1 Rozdělení

Z hlediska rozdělení podle paměti, lze mikrokontroléry rozdělit na 2 skupiny:

Von Neumannova architektura

Pro tuto architekturu je charakteristické využití jednoho adresního prostoru. V tomto prostoru je mapována paměť programu, data a registry pro řízení IO obvodů. Takové registry jsou označovány u mikrokontrolérů jako SFR (Special Function Registers). Tuto architekturu využívají například mikrokontroléry Intel 80196, Hitachi, Motorola 68HC11 a další (Dudáček, 2021).

Harwardská koncepce

Koncepce s odděleným adresním prostorem pro paměť programu a zvlášť pro data. Tuto koncepci využívají například mikrokontroléry Intel 8051, 8052, PIC firmy Microchip, AVR firmy Atmel a další (Dudáček, 2021).

1.1.2 Mikrokontroléry AMiT

Firma AMiT vyrábí volně programovatelné řídicí systémy, které jsou vhodné pro aplikace tzv. malé a střední automatizace. Nejčastěji se tyto řídicí systémy využívají v oblasti automatizace budov, řízení technologických celků, energetice nebo vytápění měst a obcí. Z hlediska poměru cena/výkon jsou tyto řídicí systémy ideální volbou. Firma AMiT disponuje vysokou spolehlivostí svých produktů. Důkazem toho jsou tisíce realizovaných aplikací po celém světě (AMiT, 2022).

Všechny řídicí systémy jsou kompatibilní a snadno připojitelné do informačního systému DB-Net/IP od firmy AMiT. Pro propojení systémů jiných výrobců je možné výhodně využít komunikaci Modbus RTU nebo Modbus TCP. Použití řady podporovaných komunikačních protokolů nebo využití možnosti parametrizace uživatelského protokolu lze připojit další zařízení, regulátory či snímače k systému. Ve vývojovém prostředí DetStudio je jednotná parametrizace řídicích systémů a programování (AMiT, 2022).

Tabulka 1.1 – Řídicí terminály a kompaktní systémy firmy AMiT (AMiT, 2022)

	Di	DO	AI	AO	Rozhraní	Displej
ADiR	6	8	6	-	RS232, RS485	2×8 znaků, 6 kláves
AMiNi-ES	8	8	4	-	RS232, RS485, Ethernet	-
AMiNi4W2	8	8	8	4	RS232, RS485, Ethernet, SD, webserver	-

Tabulka 1.1 – Řídicí terminály a kompaktní systémy firmy AMiT – pokračování

	Di	DO	AI	AO	Rozhraní	Displej
AMiNi4DW2	8	8	8	4	RS232, RS485, Ethernet, SD, webserver	122×32 bodů, 8 kláves
AMR-CP4x	-	-	-	-	RS232, RS485, Ethernet, Poseidon 868 MHz, SD, webserver	-
ART4000W3	8	8	8	2	RS232, RS485, Ethernet, SD, webserver	4×20 znaků, 27 kláves
AMiRiS99W3	16	16	8	4	RS232, Ethernet, (RS485 / CAN / M-Bus), SD, webserver	-
AMAP99W3	24	23	15	6	RS232, Ethernet, (RS485 / CAN / M-Bus), SD, webserver	-
ACOS200	32	32	16	8	RS232, RS485, Ethernet, (RS232 / RS485 / CAN), SD, webserver	-

1.2 PERIFERIE MIKROKONTROLÉRU

Kromě základních součástí, jako jsou procesor, operační paměť, paměť programu, oscilátor a vstupně/výstupní rozhraní, může mikrokontrolér obsahovat i další periferie.

1.2.1 Komunikace RS232

Tato komunikace umožňuje duplexní komunikaci. Komunikuje pomocí sériového rozhraní USART (Universal Synchronous and Asynchronous seriál Receiver and Transmitter – univerzální synchronní a asynchronní sériový přijímač a vysílač). Komunikace RS232 obsahuje filtraci šumu, detekci falešného start bitu, detekci chybného znaku a přetečení datového registru. Přijímací kanál má přidělen přijímací registr, kde se ukládají přijatá data. Uložená data musí být zpracována dříve, než se přijmou nová data k uložení, tedy k přepsání předchozích dat. Při komunikaci může dojít ke třem různým typům přerušení (ukončení

příjmu, vyprázdnění vysílacího registru a ukončení vysílání). Je kompatibilní pro práci v multiprocesorovém režimu (Šíl, 2012).

Pro dosažení vyšší rychlosti přenosu se využívá v asynchronním režimu. Pro správnou funkčnost komunikace musí dojít na začátku přenosu k synchronizaci. Pro sériovou komunikaci se současně s datovým vodičem vede tzv. synchronizační vodič, který dává informaci o začátku přenosu. Pro asynchronní komunikaci se synchronizační vodič nevyužívá a k synchronizaci dochází až po přijetí datového bloku. Pro synchronizaci se užívá jeden start bit, který se přepíná v logické 0 a logické 1 mezi dvěma sekvencemi datových bloků. Pro datové signály (RXD a TXD) logická 0 dosahuje od +3 V do +15 V, logická 1 od -3 V až -15 V. Řídicí signály (RTS, DTR, DSR, CRS, …) logická 0 dosahuje od -3 V do -15 V a logická 1 od +3 V do +15 V (Olmr, 2005).

Obrázek 1.2 – Komunikace RS232

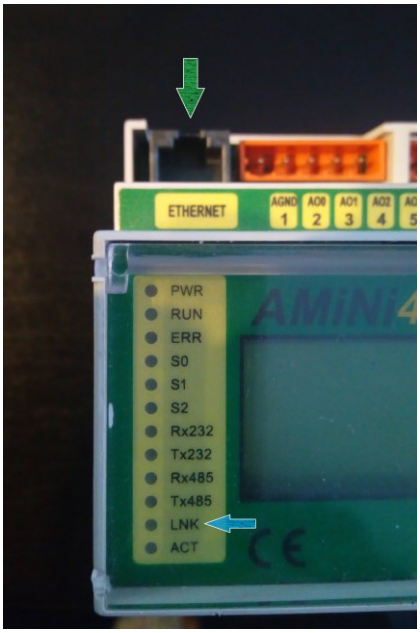
1.2.2 Ethernet

Ethernet je jedním z informačních síťových standardů. Se zvyšující se potřebou informačních propojení mezi firmami a kancelářemi v průběhu posledních let získal Ethernet popularitu jako síťový standard pro posílání instrukcí a získávání produkčních stavových hlášení. Vývojem jednočipových ethernetových řadičů se dosáhlo nižších cen ethernetových karet a většina počítačových motherboardů je má integrované. V síti Ethernetu jsou všechny

pracovní stanice propojeny a vysílají signály přes stejný kabel. Jedná se tedy o sdílenou síťovou technologii. Ethernet funguje na typu protokolu TCP (Transmission Control Protocol) nebo UDP (User Datagram Protocol). Výhodou Ethernetu je jeho spolehlivost, zabezpečení pomocí brány firewall, vysoká rychlost přenosu dat a jeho snadné používání (Mitsubishi Electric Corporation, 2014).

Data vyslaná prostřednictvím TCP mohou být přijata na TCP portu. TCP je vysoce spolehlivý na komunikačním formátu 1:1. Před tím, než odešle jakákoli data, musí být stanoveno propojení mezi ostatními zařízeními. Tento protokol je vhodný pro aplikace, které vyžadují spolehlivý přenos dat. V případě chyby přenosu se automaticky neodeslaná část vyšle znovu (Mitsubishi Electric Corporation, 2014).

Data vyslaná z aplikace prostřednictvím UDP jsou jednoduše odeslána do specifického cíle. Přenos funguje ve vysoké rychlosti přenosu, z důvodu jeho jednoduchého protokolu. Tento protokol je vhodný pro aplikace, které se používají k monitoringu v reálném čase. V případě chyby přenosu se ztrácí celý balíček dat (Mitsubishi Electric Corporation, 2014).



Obrázek 1.3 – Ethernet

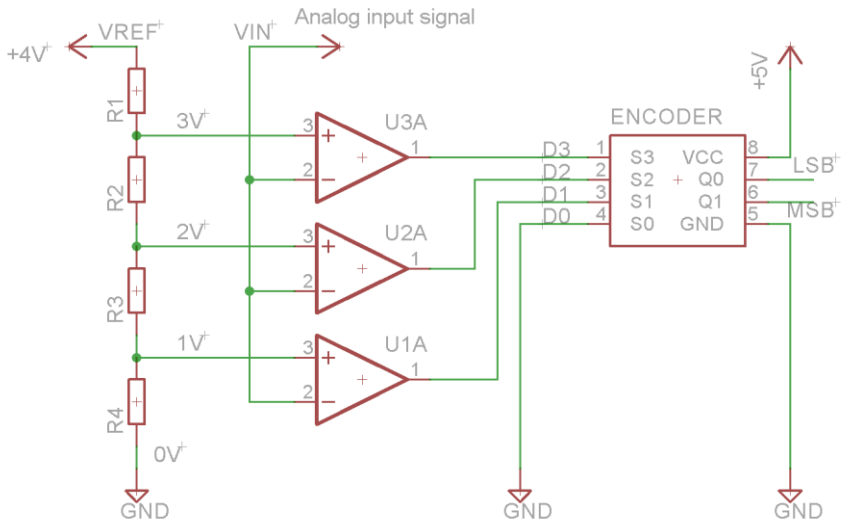
Signalizace propojení mezi mikrokontrolérem a počítačem je znázorněna rozsvícením LED diody s označením LNK na mikrokontroléru. Pro připojení ethernetu se běžně využívá konektor RJ45 (Mitsubishi Electric Corporation, 2014).

1.2.3 ADC převodník

Analogově digitální převodník umožňuje mikroprocesorem řízeným obvodům Arduinu, Raspberry Pi a dalším podobným digitálním logickým obvodům komunikovat se skutečným světem. Ve skutečném světě mají analogové signály neustále se měnící hodnoty. Tyto hodnoty pocházejí z různých zdrojů a senzorů, které mohou měřit zvuk, světlo, teplotu nebo pohyb, a mnoho digitálních systémů interaguje se svým prostředím pomocí měření analogových signálů z takových převodníků. Zatímco analogové signály mohou být spojité a poskytovat nekonečné množství různých hodnot napětí, digitální obvody na druhé straně pracují s binárním signálem, který má pouze dva diskrétní stavy, logickou „1“ (HIGH) nebo logickou „0“ (LOW). Je tedy nutné mít elektronický obvod, který dokáže převádět mezi dvěma různými stavy plynule se měnících analogových signálů a diskrétních digitálních signálů. Takovým obvodem je právě analogově-digitální převodník (AspenCore, 2022).

Analogově digitální převodník pořídí snímek analogového napětí v jednom okamžiku a vytvoří digitální výstupní kód, který toto analogové napětí představuje. Počet binárních číslic nebo bitů použitých k reprezentaci této analogové hodnoty napětí závisí na rozlišení A/D převodníku. Například 4bitový ADC bude mít rozlišení jedné části na 15, zatímco 8bitový ADC bude mít rozlišení jedné části na 255. Analogově digitální převodník tedy přijímá neznámý spojitý analogový signál a převádí jej na „n“-bitový binární počet 2n bitů (AspenCore, 2022).

Obvod 2bitového analogově-digitálního převodníku



Obrázek 1.4 – 2bitový analogově-digitální převodník

Výstup 2bitového A/D převodníku

Tabulka 1.2 – 2bitový A/D převodník

Analogové vstupní napětí	Výstup převodníku				Digitální výstup	
	D_3	D_2	D_1	D_0	Q_1	Q_0
0 až 1 V	0	0	0	0	0	0
1 až 2 V	0	0	1	X	0	1
2 až 3 V	0	1	X	X	1	0
3 až 4 V	1	X	X	X	1	1

Hodnota „X“ může nabývat logické 1 i logické 0. Aby byly A/D převodníky užitečné, musí produkovat smysluplnou digitální reprezentaci analogového vstupního signálu. Když je vstupní napětí mezi 0 V a 1 V, bude vstup na všech třech komparátorech nižší než referenční napětí, takže jejich výstupní stavy budou LOW a kodér vydá binární nulu (00) na kolících Q_0 a Q_1. Když se vstupní napětí zvýší a překročí 1 V, ale je menší než 2 V, komparátor, který má referenční napěťový vstup nastavený na 1 V, detekuje tento rozdíl napětí a vytvoří výstupní stav HIGH. Prioritní kodér, který se používá jako 4 až 2bitové kódování, detekuje změnu na vstupu D_1 a vytváří binární výstup „1“ (01) (AspenCore, 2022).

1.2.3 Přerušení (Interrupt)

Přerušení je taková událost, která pomocí řadiče pozastaví úlohu programu, kterou aktuálně provádí, provede jiný úkol a poté se vrátí k pozastavenému úkolu od toho místa, kde došlo k přerušení. Aby bylo možné provést přerušení, musí být nakonfigurováno a povoleno. Když kterékoli z přerušení je nakonfigurováno a povoleno, následně dojde ke spuštění uživatelského programu, který může vykonat následující:

a) Pozastaví její činnost,

b) provádí definovaný úkol, při kterém došlo k přerušení,

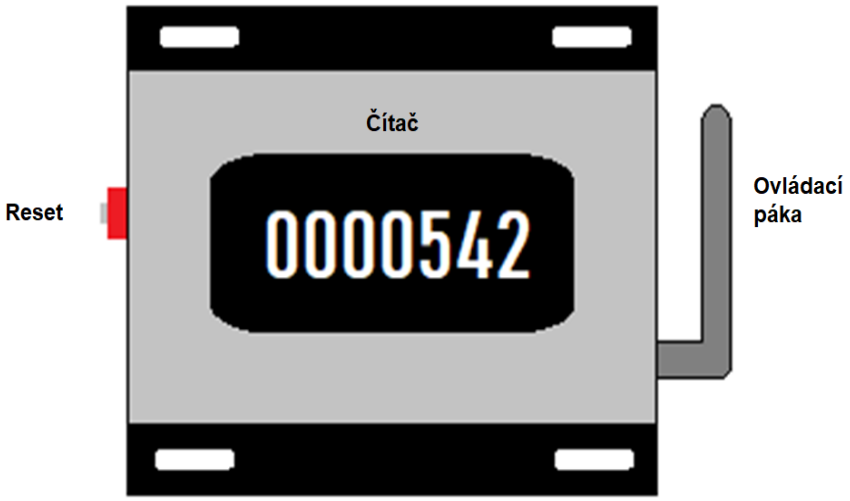
c) vrátí se do pozastaveného provozu.

Pro některé aplikace, které vyžadují vysokorychlostní odezvu (jako je řízení polohy), zpoždění v době skenování bude jistě znamenat zvýšení chybovosti. Za daných okolností lze požadavek přesnosti dosáhnout pouze použitím funkce „Přerušení“.

Hlavní rozdíl mezi přerušením a podprogramem CALL je však ten, že přerušení může nastat kdykoli během provádění hlavního programu, zatímco podprogram CALL lze provést pouze tehdy, když během skenování hlavního programu se provede instrukce CALL (Jackson, 2016).

1.2.4 Časovače a čítače

Časovače a čítače existují stejně dlouho jako relé a poskytují důležitou součást ve vývoji logiky. Časovače byly v minulosti konstruovány jako přídavné zařízení k relé pro zpomalení přechodu pístu z okamžitého otevření nebo zavření. Typické příklady čítačů PLC zahrnují přímé počítání v procesu, dva čítače používané k získání součtu dvou hodnot a dva čítače používané k určení rozdílu mezi dvěma hodnotami. Tyto klíčové instrukce jsou široce používány v průmyslových aplikacích. Mohou být například použity ke sledování počtu lahví s vodou vycházející z čerpací stanice, pro ovládání brány se změnou směru pohybu předmětů (knih, lahví atd.), aby se počítala doba na dokončení procesu, měření lineární vzdálenosti, po které se objekt pohybuje, měření rychlosti otáček motorů, sledování nežádoucích předmětů na dopravníku atd. Instrukce pro počítání PLC jsou pokročilou verzí softwaru nebo náhradou elektronických nebo mechanických čítačů (Sanamrao123, 2015).



Obrázek 1.5 – Čítač

Programové čítače mohou plnit stejnou funkci jako mechanické čítače. Vždy, když je ovládací páka posunuta, počítadlo přidá jedno číslo a ovládání páky se poté automaticky vrátí do původní polohy. Resetování na nulu se provádí tlačítkem umístěným na straně jednotky (Sanamrao123, 2015).

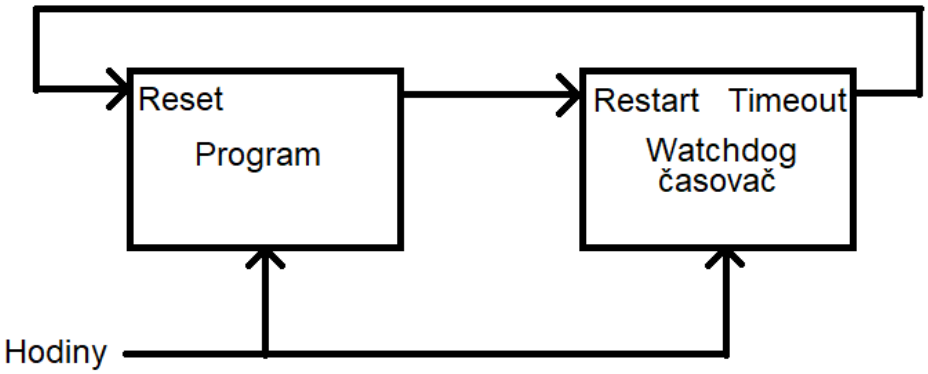
Elektronické čítače umí přičítat, odčítat nebo jejich kombinaci, aby počítaly nahoru nebo dolů. Přestože většina čítačů používaných v průmyslu jsou čítače sčítací, četné aplikace vyžadují implementaci odčítacích čítačů nebo kombinaci čítačů sčítacích a odčítacích. Čítače aktivují nebo deaktivují zařízení po určité době, kdy vypršel interval nebo počet dosáhl přednastavenou hodnotu. Všichni výrobci PLC nabízejí nějakou formu čítače instrukcí jako součást jejich instrukční sady. Jeden společný aplikační čítač sleduje počet instrukcí, které zpracoval. Čítače jsou podobné časovačům s tím rozdílem, že nefungují jako interní hodiny, ale jsou závislé na externích nebo programových zdrojích pro počítání (Sanamrao123, 2015).

1.2.5 Watchdog časovač

V PLC, když CPU zpracovává logiku, trvá to určitou dobu. Doba, kterou CPU potřebuje ke zpracování logiky, závisí na velikosti programu. Takže skenování od první instrukce do konce programu se nazývá jeden skenovací cyklus. Pro jeden cyklus skenování nesmí čas překročit definovanou dobu cyklu a dobu potřebnou ke zpracování logiky. Na základě doby skenování, kdy CPU provádí logiku, začne tento elektronický časovač počítat čas na pozadí. Pokud doba cyklu překročí stanovenou dobu, tento elektronický časovač způsobí chybu. Aby se této funkce dalo využít, časovač začne měřit čas od první instrukce a na konci resetuje své časování. Z důvodu chyby programování nebo jakékoli hardwarové chyby, pokud se CPU nepodaří resetovat časovač, dojde k chybě časového limitu. Watchdog časovač pomáhá resetovat CPU, když se vyskytnou tyto typy chyb, takže PLC začne znovu pracovat bez jakéhokoli ručního resetování. Pokud se chyby vyskytují opakovaně, pak nemusí být hlídací časovač užitečný a k vyřešení problému je nutný lidský zásah. Watchdog časovač funguje na pozadí při provádění programu (Patel, 2022).

Funkcí hlídacího časovače je v zásadě zajistit, aby nějaká porucha zpracování nevyřadila celý systém. Pokud watchdog časovač překročí předem nastavený limit, časovač vyšle signál do procesoru PLC, aby zastavil program, dokud nebude problém odstraněn (Budimir, 2017).

Nejčastěji je nutné, aby hlídací časovač byl deaktivován během spouštění mikroprocesoru. Některé mikroprocesory připojují jejich WDT k internímu oscilátoru oddělenému od systémových hodin. Externí hlídací časovače mohou také monitorovat napájení VCC a při dosažení tohoto napětí provést reset systému při poklesu napětí pod stanovenou hranici. Některé časovače poskytují odskočený a ESD chráněný vstup resetovacího spínače. Monitoring napětí v takových integrovaných obvodech je obecně přesnější v rozsahu provozních teplot, než je vestavěný brownout obvod obsažený v některých mikroprocesorech (Schlaepfer, 2008).



Obrázek 1.6 – Watchdog časovač

1.3 HISTORIE A VLASTNOSTI PLC

První programovatelné logické automaty (PLC) byly představeny koncem 60. let 20. století. Byly vyvinuty tak, aby nabízely stejnou funkčnost jako stávající reléové logické systémy. Jedná se o programovatelný, opakovaně použitelný a spolehlivý systém. Odolá drsnému průmyslovému prostředí. První PLC neměly pevný disk, měli záložní baterii. Start probíhal během několika sekund. Pro programování byla použita Ladder logika (Ieec.uned.es, 2006).

Programmable logic controller (PLC) je specializovaný počítač používaný k řízení strojů a procesů. Využívá programovatelnou paměť k ukládání instrukcí a specifických funkcí, které zahrnují ovládání zapnutí/vypnutí, časování, počítání, sekvenci, aritmetiku a zpracování dat (Ieec.uned.es, 2006).

Výhody řídicích systémů PLC

• Flexibilní,

• rychlejší doba odezvy,

• menší a jednodušší rozvod kabelů,

• Solid State – žádné pohyblivé mechanické komponenty,

• modulární design – snadno se opravuje a rozšiřuje,

• zvládá mnohem složitější systémy,

• k dispozici jsou sofistikované instrukční sady,

• umožňuje diagnostiku „snadno řešitelné problémy“,

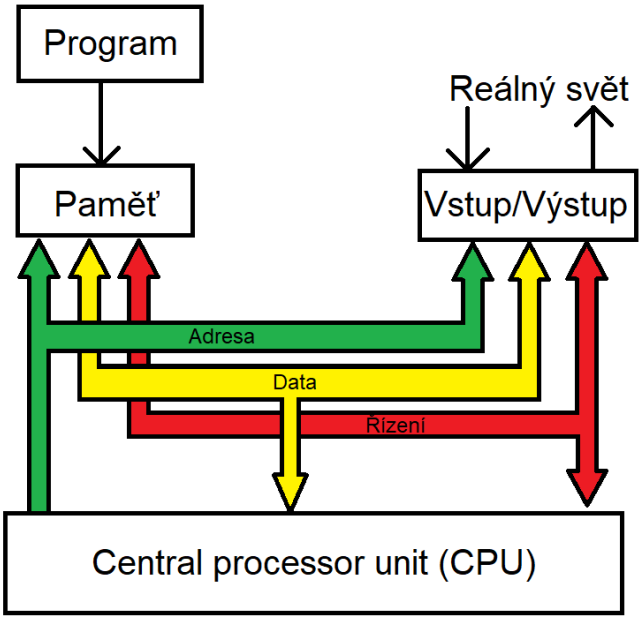
• levnější.

Program nahrazuje většinu externích kabelů, které by byly potřebné pro řízení procesu. Jakmile je program napsán a otestován, lze jej stáhnout do jiných PLC. Protože je veškerá logika obsažena v paměti PLC, není šance, že by došlo k chybě logického zapojení. Výrobci originálního vybavení (OEM) mohou poskytnout aktualizace systému pro proces pouhým odesláním nového programu. Je snazší vytvořit a změnit program v PLC než zapojovat a přepojovat obvod. Koncoví uživatelé mohou program upravovat v terénu (Ieec.uned.es, 2006).

Původně byly PLC navrženy tak, aby nahradily logiku řízení relé. Úspora nákladů pomocí PLC byla tak významná, že ovládání relé se stává zastaralým, s výjimkou aplikací pro napájení. Obecně platí, že pokud aplikace vyžaduje více než přibližně 6 řídicích relé, bude obvykle levnější nainstalovat PLC (Ieec.uned.es, 2006).

PLC může komunikovat s jinými řídicími jednotkami nebo počítačovým vybavením. Mohou být propojeny do sítě, aby mohly provádět takové funkce, jako je: dohledové řízení, shromažďování dat, monitorování zařízení a parametrů procesu a stahování a nahrávání programů (Ieec.uned.es, 2006).

PLC pracují v reálném čase, což znamená, že událost, ke které dojde v poli, povede k nějaké operaci nebo výstupu. Stroje, které zpracovávají tisíce položek za sekundu a objekty, které před senzorem stráví jen zlomek sekundy, vyžadují schopnost rychlé reakce PLC (Ieec.uned.es, 2006).



Obrázek 1.7 – PLC Architektura

Návrh otevřené architektury umožňuje snadné připojení systému k zařízením a programům od jiných výrobců. Uzavřená architektura nebo proprietární systém je systém, jehož konstrukce ztěžuje připojení zařízení a programů vyrobených jinými výrobci. Při práci se systémy PLC, které jsou svou povahou proprietární, se musí dbát na to, že jakýkoli generický hardware nebo software, který se použije, je kompatibilní s vaším konkrétním PLC (Ieec.uned.es, 2006).

PLC procesor (CPU) se skládá z mikroprocesoru pro implementaci logiky a řízení komunikace mezi moduly. Procesor přijímá vstupní data z různých snímacích zařízení, provádí uložený uživatelský program a odesílá příslušné výstupní příkazy do řídicích zařízení (Ieec.uned.es, 2006).

2 KOMUNIKACE A PROTOKOLY

Elektronická komunikace je celá o propojení obvodů (procesorů nebo jiných integrovaných obvodů) za účelem vytvoření symbiotického systému. Aby si tyto jednotlivé okruhy vyměňovaly informace, musí sdílet společný standardní komunikační protokol. K dosažení výměny dat bylo navrženo mnoho komunikačních protokolů (Sonnenberg, 2019).

Nejběžnější sériové komunikační protokoly jsou RS232, RS485, RS422, USB a Ethernet. Ale protože USB a Ethernet vyžadují výkonná rozhraní se složitými protokoly, mnoho účinných zařízení využívalo RS232, RS485 a RS422 (Sonnenberg, 2019).

Sériové protokoly RS232, RS485, RS422 se týkají pouze hardwarového rozhraní, nikoli softwarových protokolů používaných ke komunikaci zařízení. Na trhu existuje mnoho protokolů, takže nelze předpokládat interoperabilitu mezi různými výrobci portů "RS232".

Mikrokontroléry AMiNi4W2 a AMiNi4DS umožňují komunikaci na rozhraní RS232, RS485 a Ethernet (AMiT, 2022).

Obrázek 2.1 – komunikace RS485

2.1 KOMUNIKAČNÍ PROTOKOL RS485

Standard komunikačního protokolu RS485 byl definován v roce 1983 a využívá se až do dnešní doby, protože má dobré vlastnosti pro instalaci v budovách. Sběrnice může pracovat až na délce 1200 metrů a díky nízké rychlosti přenosu je vysoce odolná proti rušení. Proto má i stále vysokou podporu výrobců. Pro propojení dvou zařízení je nutné využít pro komunikaci stejného protokolu. Další podmínkou pro správnou funkčnost komunikace je určení jednoho master zařízení, které tuto komunikaci bude řídit (Vidim, 2011).

Základní typ sběrnice RS485 je tvořen třemi vodiči. Vodiče A a B představují logický stav, který je dán potenciálovým rozdílem těchto vodičů. Třetí vodič je společný a používá se jako uzemnění. V praxi se na vodič A připojuje plus a na vodič B mínus. Připojení může být v dokumentaci zakresleno opačně, a proto je lepší si ověřit polaritu multimetrem (Vidim, 2011).

Pro realizaci trasy signálů rozhraní RS485 se využívá jeden kroucený pár vodičů pro komunikaci s impedancí 120 Ω. Na krátké vzdálenosti není nutné využívat stínění. Způsob přidělování sběrnice závisí na využité kategorii protokolu, tzn. jednoduché struktury „master-slave“ nebo složitější „multimaster“ (Tedia spol. s.r.o., 2018).

2.2 VÝHODY RS485 A RS422 PROTI RS232

RS485 a RS422 využívají rozlišení. Pro každý signál jsou potřeba dva vodiče. Pro přenos logické 1 je linka B ve stavu „High“ a linka A ve stavu „Low“. Pro přenos logické 0 je linka B „Low“ a linka A „High“. Výhodou tohoto uspořádání je, že signály mohou být přenášeny rychleji a na větší vzdálenosti, než je možné s jediným vodičem (Sonnenberg, 2019).

Hlavní rozdíl mezi RS422 a RS485 je v typech povolených komunikací. RS422 umožňuje pouze jednosměrnou (simplexní) komunikaci mezi jedním ovladačem a až deseti přijímacími zařízeními. Pro ovládání zařízení bez potřeby zpětné vazby, bude dobře fungovat síť RS-422 multidrop. Rozhraní RS485 bylo navrženo tak, aby řešilo omezení RS422 na vícenásobné poklesy a umožnilo komunikaci s až 32 zařízením (Sonnenberg, 2019).

2.3 MODBUS PROTOKOL

Komunikační protokol Modbus se dělí na Modbus TCP a Modbus RTU. Modbus protokol je založen na struktuře „master-slave“. Výhoda spočívá v jednoduchosti, rychlosti a nezávislosti na výrobci. Běžně se využívá pro komunikaci mezi automaty, termostaty, čidly, frekvenčními měniči a další (Domat Control System s.r.o., 2020).

Regulátory lze nastavit tak, aby komunikovaly ve standardních sítích Modbus jeden ze dvou režimů přenosu: ASCII nebo RTU. Během konfigurace každého ovladače si uživatel vybere požadovaný režim spolu s komunikačními parametry sériového portu (přenosová rychlost, režim parity atd.). Režim a sériové parametry musí být stejné pro všechna zařízení v síti Modbus. Výběr režimu ASCII nebo RTU se týká pouze standardních sítí Modbus. Definuje bitový obsah polí zpráv přenášených sériově v těchto sítích. Určuje, jak budou informace zabaleny do polí zpráv a dekódovány (Modicon, 1996).

Na jiných sítích, jako je například Modbus Plus, jsou zprávy Modbus umístěny do rámce, které nesouvisejí se sériovým přenosem. Například žádost o přečtení přídržného registru lze zpracovávat mezi dvěma ovladači na Modbus Plus bez ohledu na aktuálním nastavení sériového portu Modbus obou ovladačů (Modicon, 1996).

2.3.1 Modbus ASCII

Když jsou ovladače nastaveny tak, aby komunikovaly v síti Modbus pomocí ASCII režimu (American Standard Code for Information Interchange), každý 8bitový bajt ve zprávě je odeslán jako dva znaky ASCII. Hlavní výhodou tohoto režimu je, že umožňuje, aby se mezi znaky objevovaly časové intervaly až jedné sekundy, aniž by došlo k chybě (Modicon, 1996).

Formát pro každý bajt v režimu ASCII je:

Systém kódování: hexadecimální, znaky ASCII 0–9, A–F,
každý obsahuje jeden hexadecimální znak,
ASCII znak zprávy.

Bitů na bajt: jeden start bit,
7 datových bitů, nejméně významný bit se odesílá jako první,
1 bit pro sudou/lichou paritu; žádný bit pro žádnou paritu,

1 stop bit při použití parity; 2 bity, pokud není parita.

Kontrolní pole chyb: kontrola podélné redundance.

2.3.2 Modbus RTU

Když jsou ovladače nastaveny tak, aby komunikovaly v síti Modbus pomocí RTU režimu (Remote Terminal Unit), každý 8bitový bajt ve zprávě obsahuje dva 4bitové hexadecimální znaky. Hlavní výhodou tohoto režimu je, že větší hustota znaků umožňuje lepší datovou propustnost než ASCII při stejné přenosové rychlosti. Každá zpráva musí být přenášena v nepřetržitém proudu (Modicon, 1996).

Formát pro každý bajt v režimu RTU je:

Systém kódování: 8bitový binární, hexadecimální 0–9, A–F,

každý bajt obsahuje dva hexadecimální znaky,

8bitové pole zprávy.

Bitů na bajt: jeden start bit,

8 datových bitů, nejméně významný bit se odesílá jako první,

1 bit pro sudou/lichou paritu; žádný bit pro žádnou paritu,

1 stop bit při použití parity; 2 bity, pokud není parita.

Kontrolní pole chyb: kontrola cyklické redundance.

2.3.3 Modbus rámcování

V jednom ze dvou režimů sériového přenosu (ASCII nebo RTU) je zpráva Modbus umístěna vysílacím zařízením do rámce, který má známý počáteční a koncový bod. To umožňuje přijímacím zařízením začít na začátku zprávy, přečíst část adresy a určit, které zařízení je adresováno (nebo všechna zařízení, zda je zpráva vysílána) a určit, kdy je zpráva dokončena. Lze detekovat dílčí zprávy a v důsledku toho nastavit chyby (Modicon, 1996).

ASCII rámcování

V režimu ASCII začínají zprávy znakem „dvojtečka“ (hexadecimální znak ASCII 3A) a končí dvojicí „carriage return – line feed“ (CRLF) (ASCII 0D a 0A hex). Povolené znaky přenášené pro všechna ostatní pole jsou hexadecimální 0–9, A–F. Síťová zařízení nepřetržitě monitorují síťovou sběrnici, zda neobsahují znak „dvojtečka“. Když je přijata, každé zařízení dekóduje další pole (adresové pole) a zjistí, zda se jedná o adresované zařízení. (Modicon, 1996).

Mezi znaky ve zprávě mohou uplynout intervaly dlouhé až po dobu jedné sekundy. Pokud dojde k delšímu intervalu, přijímající zařízení předpokládá, že došlo k chybě. Níže je uveden typický rámec zprávy (Modicon, 1996).

Tabulka 2.1 – Rámec zprávy Modbus ASCII, (Modicon, 1996)

Start	Adresa	Funkce	Data	LRC ověření	Konec
1 znak	2 znaky	2 znaky	n znaků	2 znaky	2 znaky CRLF

RTU rámcování

V režimu RTU zprávy začínají tichým intervalem o délce alespoň 3,5 znaku. To se nejsnáze implementuje jako násobek četnosti znaků při přenosové rychlosti (baud rate), který se používá v síti (v tabulce 2.2 zobrazeno jako T1–T2–T3–T4). První pole, které se pak odešle, je adresa zařízení (Modicon, 1996).

Povolené znaky přenášené pro všechna pole jsou hexadecimální 0–9, A–F. Síťová zařízení monitorují síťovou sběrnici nepřetržitě, a to i během „tichých“ intervalů. Když je přijato první pole (pole adresy), každé zařízení jej dekóduje, aby určil, zda se jedná o adresované zařízení (Modicon, 1996).

Po posledním vysílaném znaku následuje podobný interval, minimálně 3,5 znaku, který označuje konec zprávy. Po tomto intervalu může začít nová zpráva. Celý rámec zprávy musí být přenášen jako nepřetržitý celek. Pokud se před dokončením rámce vyskytne „tichý“ interval delší než 1,5 znaku, přijímající zařízení vyprázdní neúplnou zprávu a předpokládá, že další bajt bude polem adresy nové zprávy (Modicon, 1996).

Podobně, pokud nová zpráva začíná dříve než 3,5 znaku po předchozí zprávě, přijímající zařízení ji bude považovat za pokračování předchozí zprávy. Tím se způsobí chyba, protože hodnota v posledním poli CRC nebude pro kombinované zprávy platná. Níže je uveden typický rámec zprávy (Modicon, 1996).

Tabulka 2.2 – Rámec zprávy Modbus RTU, (Modicon, 1996)

Start	Adresa	Funkce	Data	CRC ověření	Konec
T1-T2-T3-T4	8 bitů	8 bitů	N x 8 bitů	16 bitů	T1-T2-T3-T4

Když se pro rámcování znaků používá režim RTU, pole kontroly chyb obsahuje 16bitovou hodnotu implementovanou jako dva 8bitové bajty. Hodnota kontroly chyb je výsledkem výpočtu kontroly cyklické redundance provedeného u obsahu zprávy. Pole CRC je připojeno ke zprávě jako poslední pole ve zprávě. Když se tak stane, nejprve se připojí bajt nižšího řádu, za nímž následuje bajt vyššího řádu. Bajt nejvyššího řádu CRC je poslední bajt, který má být ve zprávě odeslán (Modicon, 1996).

2.4 ZAKONČOVACÍ REZISTORY

Zakončovací odpory by měly být umístěny na obou koncích krouceného páru vodičů. Hodnota impedance by měla být stejná jako charakteristické impedance kroucené dvoulinky a měla by být umístěna na vzdálenějších koncích kabelu. U kabelů o délce 610 m nebo méně není při přenosové rychlosti 9 600 bps nebo méně potřeba zakončovací odpor (Sonnenberg, 2019).

Pokud je vyžadován zakončovací odpor, měl by být použit 120 Ω nebo vyšší. Neměly by být použity více než 2 ukončovací odpory, jeden na každém konci přenosové linky RS485 (Sonnenberg, 2019).

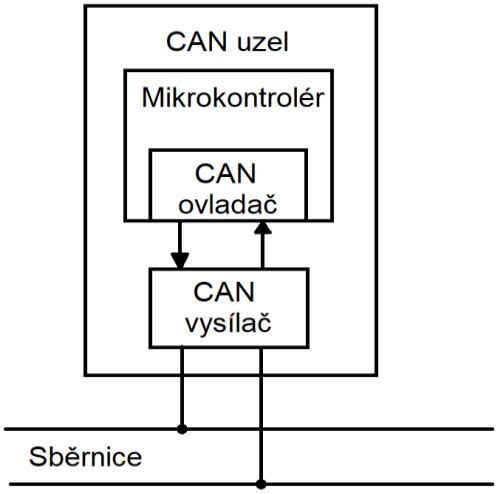
Vodiče A B by měly být vždy krouceny dohromady. Smyčka vodičů pomáhá snižovat šum, a když se elektrický, magnetický a RF šum spojí s kabelem A B, smyčkou kabelu, dostane se šum rovnoměrně na vodiče A i B, takže rozdílová data nemají žádný rozdílový šum (Sonnenberg, 2019).

2.5 CAN PROTOKOL

Sběrnici CAN vyvinula společnost BOSCH jako multi-master systém vysílání zpráv, který specifikuje maximální rychlost signalizace 1 megabit za sekundu (bps). Na rozdíl od tradiční sítě, jako je USB nebo Ethernet, CAN neposílá velké bloky dat point-to-point z uzlu A do uzlu B pod dohledem centrální sběrnice master. V síti CAN se mnoho krátkých zpráv, jako je teplota nebo otáčky za minutu, vysílá do celé sítě, což zajišťuje konzistenci dat v každém uzlu systému (Corrigan, 2002).

CAN je sériová komunikační sběrnice definovaná Mezinárodní normalizační organizací (ISO), původně vyvinutá pro automobilový průmysl, aby nahradila složitý kabelový svazek dvouvodičovou sběrnicí. Specifikace vyžaduje vysokou odolnost vůči elektrickému rušení a schopnost vlastní diagnostiky a opravy datových chyb. Tyto funkce vedly k popularitě CAN v různých průmyslových odvětvích, včetně automatizace budov, lékařství a výroby (Corrigan, 2002).

Komunikační protokol CAN popisuje, jak jsou informace předávány mezi zařízeními v síti, a odpovídá modelu OSI (Open Systems Interconnection). Skutečná komunikace mezi zařízeními propojenými fyzickým médiem je definována fyzickou vrstvou modelu OSI (Corrigan, 2002).



Obrázek 2.2 – CAN uzel (node)

Komunikační protokol CAN je protokol s vícenásobným přístupem s rozpoznáváním nosného signálu s detekcí kolize a rozhodováním o prioritě zprávy (CSMA/CD+AMP). CSMA znamená, že každý uzel na sběrnici musí čekat po předepsanou dobu nečinnosti, než se pokusí odeslat zprávu. CD+AMP znamená, že kolize jsou řešeny pomocí bitové arbitráže

na základě předem naprogramované priority každé zprávy v poli identifikátoru zprávy. Přístup na sběrnici vždy vyhrává identifikátor s vyšší prioritou. To znamená, že nejvyšší hodnota v identifikátoru pokračuje ve vysílání, protože má nejvyšší prioritu. Vzhledem k tomu, že každý uzel na sběrnici se podílí na zapisování každého bitu „tak, jak je zapsán“, rozhodovací uzel ví, zda umístil na sběrnici logický bit (Corrigan, 2002).

3 HARDWARE

Rozvaděč by měl být osazen potřebnými jističi a může obsahovat chrániče, přepěťovou ochranu a jiné přístroje (Pojar, 2018).

3.1 NAPÁJECÍ ZDROJ AD1048-24FS

Tento napájecí zdroj je určen pro osazení na DIN lištu. Má jeden výstup a to 24 V stejnosměrného napětí. Výstupní výkon je 48 W. Jedná se o zdroj s plastovým pouzdrem. Zdroj je vybaven indikační LED diodou pro zobrazení stavu výstupního napětí na zdroji. Vstupní rozsah napětí je v plném rozsahu, tedy od 100 až do 230 V střídavého napětí při frekvenci 47 až 63 Hz. Stupeň ochrany IP20 (Acro Engineering Inc., 2017).

Obrázek 3.1 – Zdroj AD1048-24FS

3.2 JISTIČ

Je to takový elektrický přístroj, který je určen ke spínání elektrických obvodů a chrání je proti nadproudům. Generování a dodávání elektrické energie uživatelům je cílem systému elektrické energie. Aby bylo možné zkonstruovat a umožnit odeslání energie tam, kde bude

využita, musí mít spolehlivost a hospodárnost. Vzhledem k tomu, že energetický systém je využit k výrobě, přenosu, distribuci a bydlení, potřebuje ochranu, aby nedošlo k poškození elektrických spotřebičů nebo ke ztrátě života (Goh, 2017).

V elektrotechnice je energetický systém jednou z exkluzivních oblastí, které se zabývají co nejkratším a nejrychlejším časem pro vypnutí a izolaci vadné oblasti v energetickém systému tak, aby porucha přímo neovlivnila systém. Výsledkem je, že systému zůstává jeho stabilita a spolehlivost. Základními prvky ochrany je snímač pro identifikaci poruchového stavu a zařízení pro iniciaci vypínacího signálu do jističe (Goh, 2017).

Ochranný systém musí splňovat požadavky na rychlé a automatické odpojení vadné části elektrické sítě a minimalizaci odpojení nebo přerušení dodávky energie ke spotřebiteli. Schopnost vypnout, pokud došlo k poruše v napájecím systému, by měla mít ochrana napájecího systému, což je jistič. Schopnost vypnutí u jističů je spolehlivost, selektivita, citlivost, rychlost a stabilita (Goh, 2017).

Jistič, známý také jako automaticky ovládaný elektrický spínač, při zjištění poruchy zareaguje přerušením toku proudu. Existují různé velikosti jističů, od malých zařízení až po velké rozváděče, které se používají k ochraně nízkého proudu v obvodu až po vysokonapěťový obvod (Goh, 2017).

Obrázek 3.2 – Jistič Schneider B16

3.3 SPÍNAČE

Spínač je součástka, která ovládá otevřenost nebo uzavřenost elektrického obvodu. Umožňují kontrolu nad tokem proudu v obvodu (aniž by se tam muselo skutečně dostat a ručně řezat nebo spojovat dráty). Spínače jsou kritickými součástmi v jakémkoli obvodu, který vyžaduje interakci nebo ovládání uživatele. Spínač může existovat pouze v jednom ze dvou stavů: otevřený nebo zavřený. Ve vypnutém stavu vypadá spínač jako otevřená mezera v obvodu. To ve skutečnosti vypadá jako otevřený obvod, který brání toku proudu. V zapnutém stavu se spínač chová jako kus dokonale vodivého drátu. Tím se okruh uzavře, systém se „zapne“ a proud bude nerušeně protékat zbytkem systému (Jimblom, 2013).

Tlačítkové spínače jsou klasickým mžikovým spínačem. Obvykle mají tyto spínače opravdu pěknou, hmatovou, „cvakavou“ zpětnou vazbu při stisknutí. Jsou vyráběny v nejrůznějších typech: velké, malé, barevné, osvětlené (když tlačítkem prosvítá LED). Mohou být uzpůsobeny pro povrchovou montáž, s průchozím otvorem nebo dokonce montáž na panel (Jimblom, 2013).

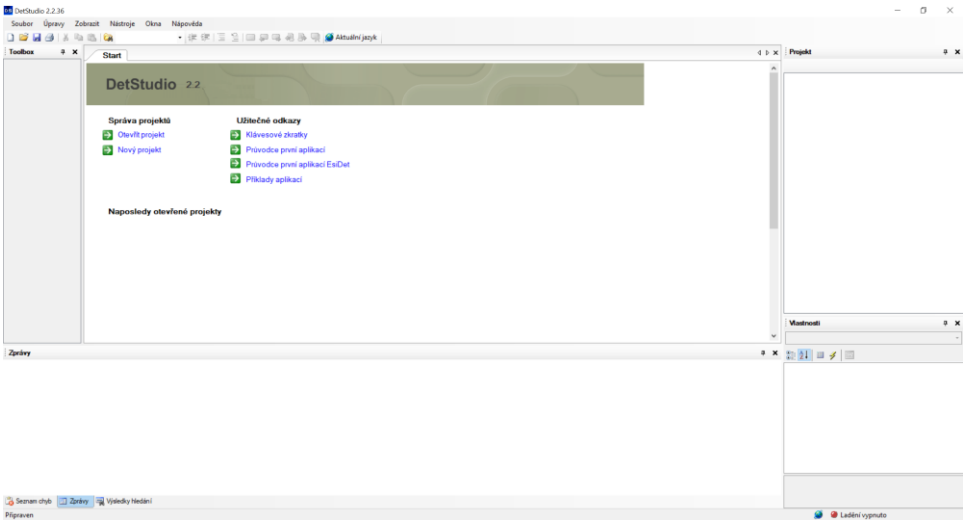
Obrázek 3.3 – Tlačítka a přepínače

4 DETSTUDIO202

Pro komunikaci a programování jsem využil program DetStudio202, který doporučuje česká firma AMiT na svých webových stránkách pro PC komunikaci s moduly. Výhoda tohoto programu je kompatibilita i pro českou verzi.

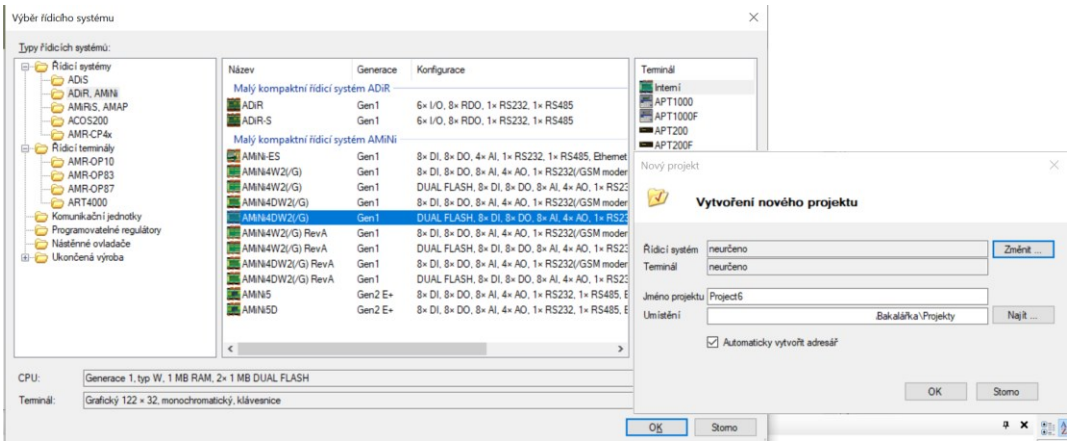
4.1 APLIKAČNÍ PROSTŘEDÍ

Po instalaci programu DetStudio202 se při jeho spuštění otevře hlavní okno s výběrem projektu, na kterém jsem vytvořil nový projekt. Po uložení prvního projektu se v tomto okně zobrazuje nabídka posledních otevřených projektů. Pro ilustraci prvního spuštění programu jsou poslední otevřené projekty zakryty (viz obrázek 4.1).



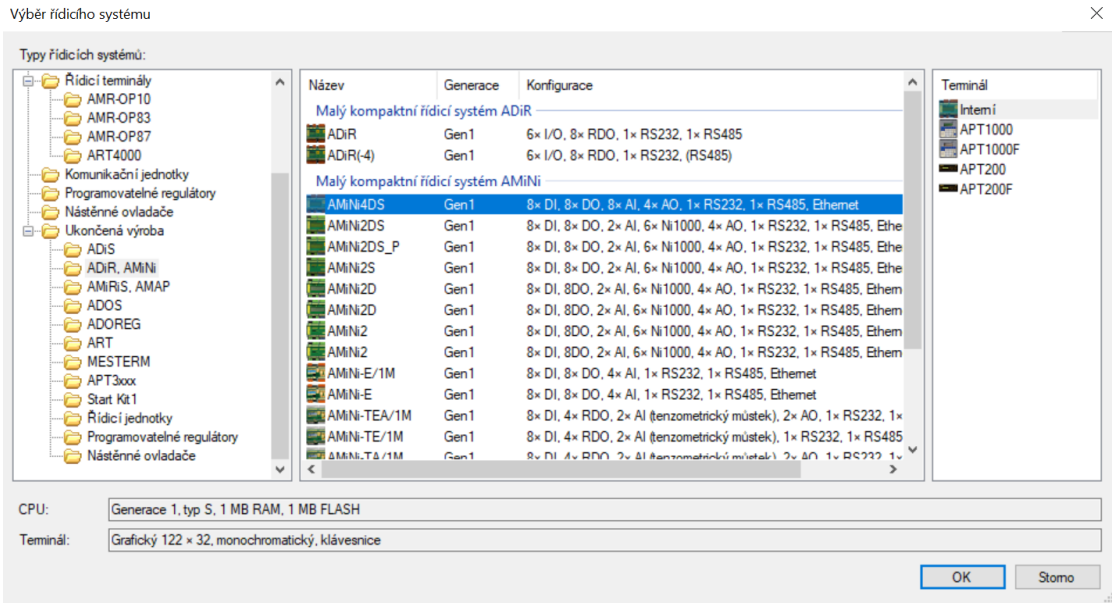
Obrázek 4.1 – Vývojové prostředí DetStudio202

Při vytváření projektu jsem musel zadat správné parametry projektu, aby komunikace mezi PC a modulem fungovala správně. Při zadání špatného typu řídicího modulu by mi nemusel fungovat systém správně. Prvním modulem, který jsem se snažil spojit, byl typ AMiNi4DW2/G s dvojitou flash pamětí. Po výběru řídicího systému jsem si vhodně zadal název projektu, abych se lépe orientoval mezi ostatními projekty. Umístění projektu není podstatné, jen je vhodné mít všechny na jedné adrese.



Obrázek 4.2 – Výběr modulu AMiNi4DW2/G

Druhý modul AMiNi4DS se v aplikaci DetStudio nenachází mezi novějšími typy řídicích systémů a našel jsem ho v kategorii *Ukončená výroba*. Ačkoli je tento modul staršího typu, společnost AMiT dbá na kompatibilitu všech svých produktů, a proto jsem mohl tento modul využít pro komunikaci. Tento modul jsem řešil až později, dokud jsem nevyřešil identifikaci prvního modulu. Řešení problémů v identifikaci jsem popsal v kapitole 4.2.



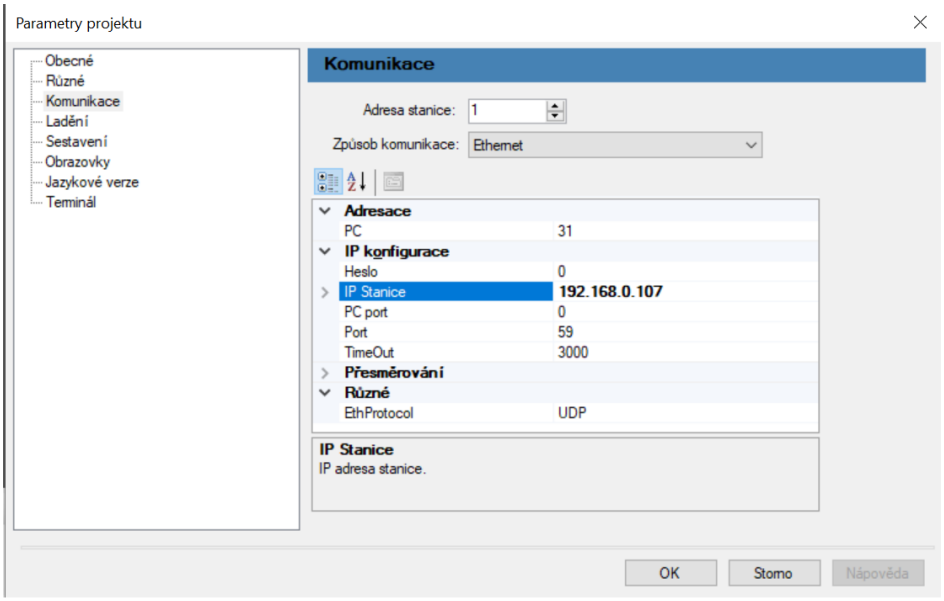
Obrázek 4.3 – Výběr modulu AMiNi4DS

4.2 IDENTIFIKACE MODULU

Poté, co jsem zadal typ řídicího modulu, jsem musel nastavit komunikační parametry (viz obrázek 4.4). Komunikaci lze navázat použitím jednoho ze čtyř způsobů: sériová linka, Ethernet, servisní mód nebo GPRS. Pro tuto práci jsem použil způsob komunikace přes Ethernet. Pro propojení přes Ethernet jsem využil switch ES-3205P, přes který jsem jedním vstupem napojil router, druhým vstupem první PLC a třetím vstupem druhé PLC.

V nastavení komunikace je adresa stanice klíčová pro definici master-slave zařízení a zároveň pro komunikaci mezi PC a PLC. Pokud bych zadal správnou IP adresu a komunikační port se špatnou adresou stanice, nefungovala by identifikace a nešlo by se se stanicí spojit a nahrávat do ní potřebná data.

Pro Ethernet komunikaci je nutné znát IP adresu modulu, kterou běžně při zakoupení lze najít v příbalovém letáku nebo získat od prodejce. Znát IP adresu modulu je potřeba k tomu, aby se nastavila statická adresa počítače na stejnou podsíť a umožnilo se tak přepsání IP adresy modulu na IP adresu takovou, na které bude celý projekt pracovat.

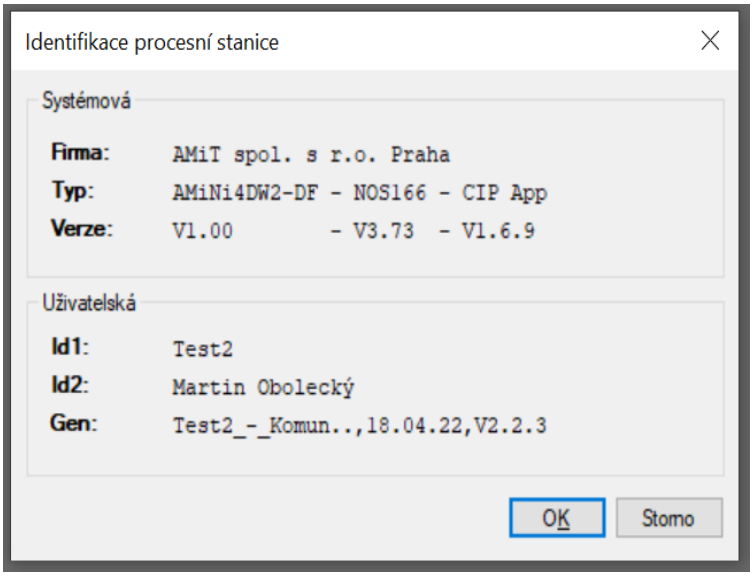


Obrázek 4.4 – Parametry projektu

Jelikož mé moduly už někdy byly v provozu a IP adresu někdo změnil, je u některých starších modulů obtížnější identifikovat IP adresu. Pro jednodušší identifikaci IP adresy připojeného zařízení lze vyhledat zařízení přes hlavní lištu v sekci *Nástroje – Najít stanici na*

ethernetu. Pokud se připojený modul nezobrazí v seznamu, jako například AMiNi4DS, bylo nutné najít IP adresu jinak. Pro nalezení IP adresy jsem vyzkoušel různé programy, jako je například Advanced IP Scanner, který ale bohužel také nenašel IP adresu připojeného modulu. IP adresu se mi povedlo najít až s programem WireShark, který umí sledovat každý odeslaný a přijatý packet prostřednictvím Ethernet protokolu. Pro komunikaci jsem využil adresy 192.168.0.106 pro svůj počítač, 192.168.0.107 pro modul AMiNi4DW2 a 192.168.0.108 pro modul AMiNi4DS.

Pro ověření propojení jsem využil z hlavního panelu *Přenos – Identifikace*. Dokud se mi nezobrazila tabulka (viz obrázek 4.5), je nastavená komunikace špatně. Nejčastější problém byl v nastavení adresy stanice, protože je obtížné zjistit, na které adrese stanice komunikuje. Nakonec jsem si zjistil hodnotu adresy z dokumentací z použitých lokalit, kde se tyto PLC využívaly. Po nastavení identifikace modulu lze nahrát NOS, data a celý projekt přímo do stanice nebo změnit její konfiguraci.



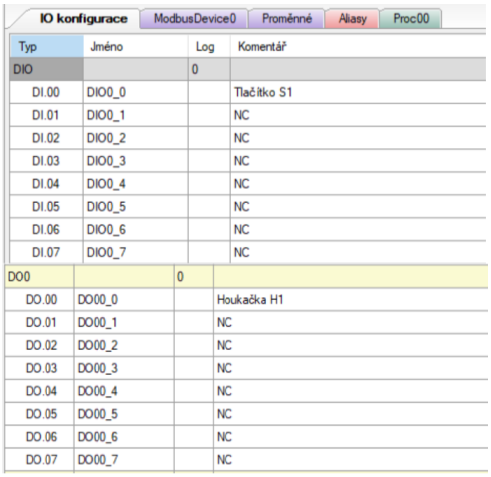
Obrázek 4.5 – Identifikace zařízení

4.3 TEST PROGRAMU PRO BLIKÁNÍ LED

Po vytvoření projektu a nastavení identifikace jsem teprve mohl začít tvořit nějaká data projektu. Pro správnou strukturu a přehlednost projektu jsem přidal komentáře pro všechny využité vstupy a výstupy modulu v IO konfiguraci projektu.

4.3.1 IO Komunikace

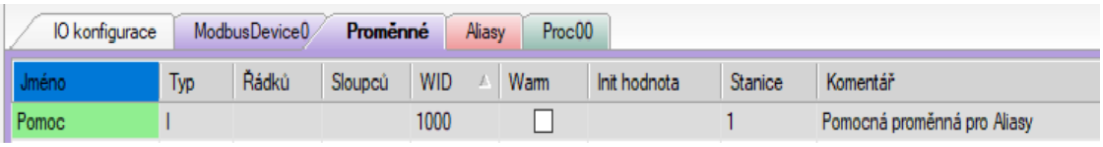
V sekci I/O komunikace jsem si pro první test programu zapsal, že vstup typu *DI.00* s výchozím pojmenováním *DIO0.0* bude mít komentář „Tlačítko S1“. Pro výstup jsem si typ *DO.00* s výchozím pojmenováním *DO00.0* předepsal komentář jako „Houkačka H1“.



Obrázek 4.6 – IO Konfigurace

4.3.2 Proměnné

Po nastavení komentářů pro orientaci vstupních a výstupních kanálů jsem si připravil jednu proměnnou nazvanou jako „Pomoc“. Vytvořil jsem ji jako pomocnou proměnnou k vytvoření dvou aliasů, které budou odkazovat na binární stavy této proměnné. Inicializační hodnotu nechávám na hodnotě 0, abych poznal načtení hodnoty ze vstupního kanálu *DIO0.0*.

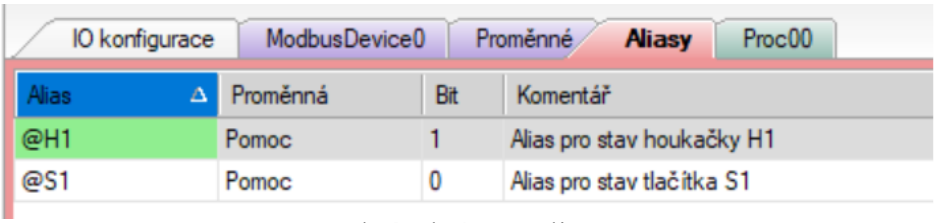


Obrázek 4.7 – Proměnné

4.3.3 Aliasy

Pro načítání hodnot ze vstupního signálu jsem použil funkci *BinIn*, která čte binární hodnotu a zapisuje ji do proměnné nebo přímo do aliasu. Pro jednoduché ovládání houkačky

jsem si vytvořil alias „@S1“, který odkazuje na bit 0 proměnné *Pomoc*, kterou jsem si vytvořil. Do komentáře pro tento alias jsem si zapsal informaci, že se jedná o „Alias pro stav tlačítka S1“, abych při pozdějším studování věděl, k čemu slouží tento alias. Druhý alias jsem pojmenoval jako „@H1“, který odkazuje na bit 1 proměnné *Pomoc*. Do komentáře jsem si vypsal „Alias pro stav houkačky H1“. Komentáře jsou podstatné pro pochopení nejen pro mne, ale i pro budoucího vývojáře.



Obrázek 4.8 – Aliasy

4.3.4 Program blikání

Po nastavení všech potřebných proměnných, aliasů a komentářů ke vstupním a výstupním svorkám jsem si sepsal jednoduchý program do *normal* procesu „Proc00“, který funguje opakovaně s periodou 1000 milisekund. Program lze psát ve strukturovaném textu, v žebříkové logice nebo v sekvenčním funkčním schématu. Mně osobně vyhovuje více strukturovaný text.

```
BinIn #DIO0_0, 0x0000, @S1

If @S1
        Let @H1 = not @H1
Else
        Let @H1 = False
EndIf

BinOut @S1, 0x0000, #DO00_0
```

Obrázek 4.9 – Program pro blikání LED

Pro získání binární hodnoty ze vstupu *DIO.0* jsem použil funkci *BinIn*. Pro tuto funkci jsem zadal parametry funkce s odkazem na čtení vstupního signálu *DIO0.0*, negaci čtení bitového stavu jsem neměnil a čtenou hodnotu jsem nastavil zápis na alias *@S1*.

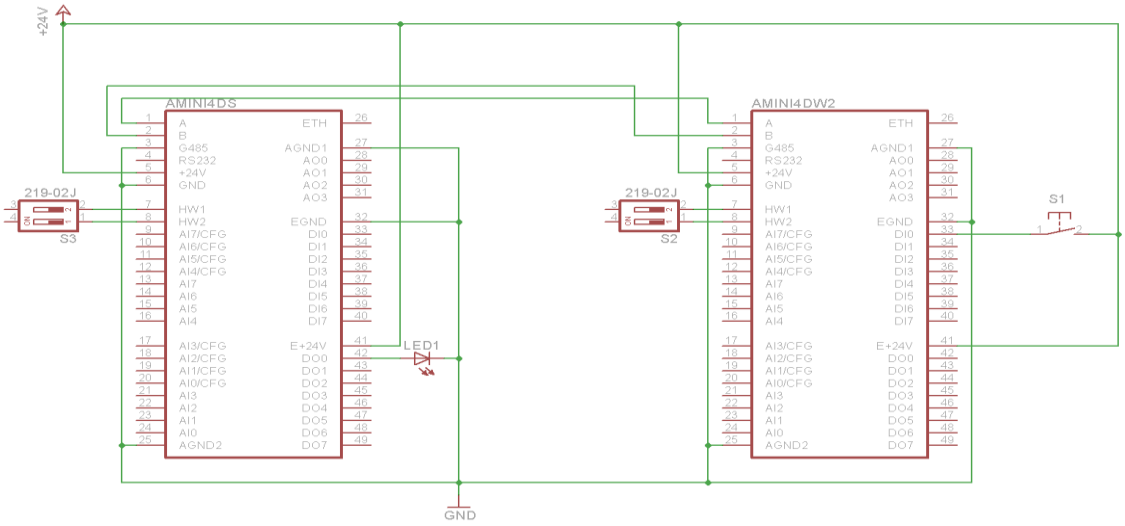
Pro ověření stavu a následné operace na základě stavu hodnoty aliasu *@S1* jsem vytvořil podmínku *If @S1*, která nastane tehdy, kdy hodnota zapsaná v alias proměnné *@S1* je logická 1. Pro tento stav se alias *@H1* zneguje, a tím se dosáhne přepínání stavu, tedy k periodickému blikání výstupu. Příkaz pro toto blikání jsem nastavil jako *Let @H1 = not @H1*. V případě že na vstupu bude logická 0, nastavil jsem alias *@H1 tak*, aby se nastavil na logickou 0, tedy vypnutí výstupního stavu. Příkaz pro vypnutí jsem nastavil jako *Let @H1 = False*. Takto jednoduchý zápis podmínky mi stačil k rozpoznávání stavu na vstupu *DIO0.0*, tedy jsem ho ukončil příkazem *EndIf*.

Pro to, abych mohl reálně kontrolovat stav výstupu *DO00.0,* jsem využil funkci *BinOut*, které jsem nastavil parametry pro zápis hodnoty do výstupního kanálu pomocí aliasu *@S1*. Negaci jsem nevyužil a *channel* zapisovaného signálu jsem zadal jako *DO00.0*.

4.3.5 Návrh zapojení testu pro blikání LED

Tabulka 4.1 – Seznam součástek pro blikání LED

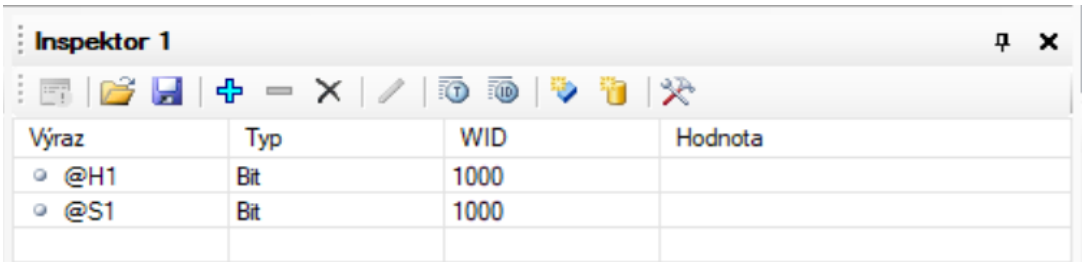
Označení	Typ	Hodnota	Funkce
LED1	LED dioda	LED3MM	indikátor blikání
S1	tlačítko	31-XX	zapnout / vypnout blikání



Obrázek 4.11 – Schéma zapojení pro blikání LED

4.3.6 Inspektor

Nahrál jsem program do stanice pomocí *Generace – Generuj vše*. Při nahrávání programu nezadávám heslo, protože to zatím nebyl program určený k reálnému použití a nebylo třeba ho zabezpečovat. Především každé nahrávání programu by stanice vyžadovala heslo a zdržovalo by to při práci. Bez připojeného zařízení jsem sledoval funkčnost pomocí *Ladění – Inspektor 1*.



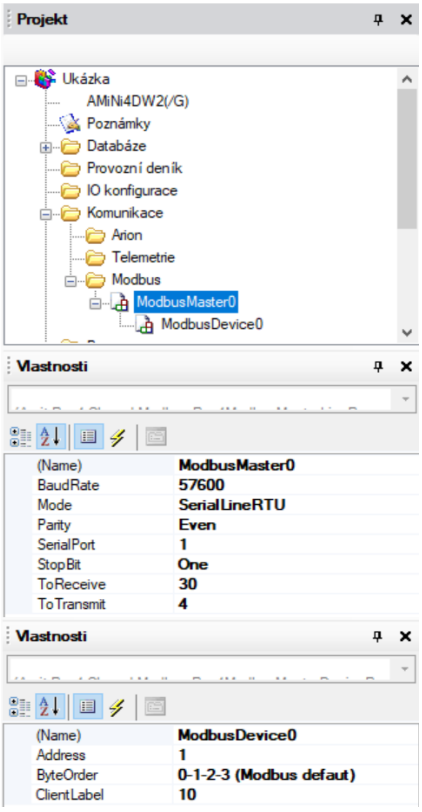
Obrázek 4.10 – Inspektor DetStudio

5 PROTOKOL MODBUS RTU

Pro svou bakalářskou práci jsem využil komunikační protokol Modbus RTU, protože jsem pro realizaci využil starší modul AMITI4DS, který nepodporuje TCP propojení. Více se užívá Modbus/TCP, Profibus či Profinet a další.

5.1 STANICE MASTER

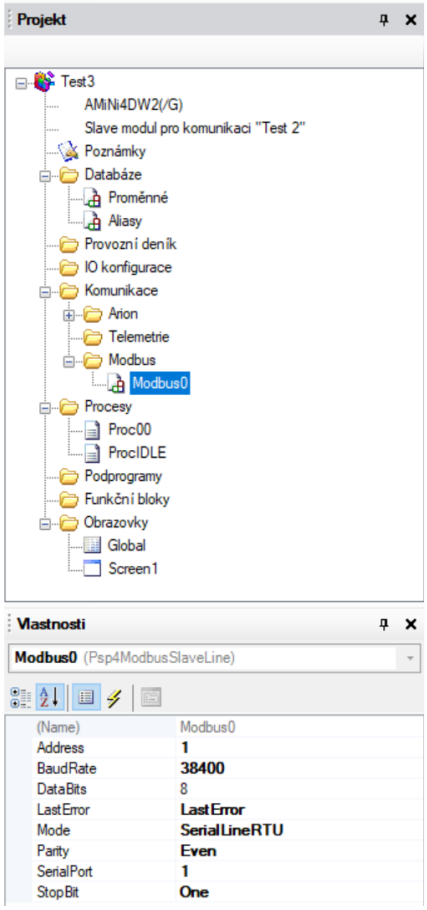
Pro propojení dvou PLC modulů jsem jeden z modulů nastavil jako „master“ zařízení a ostatní moduly jako slave v komunikačním protokolu Modbus RTU. Pro přidání takového master zařízení slouží v DetStudiu sekce *Komunikace – Modbus – Přidat Master*. Po vybrání jsem přidal zařízení s označením ModbusMasterX a pro propojení se zařízením slave. Pro komunikaci prostřednictvím RS485 slouží SerialPort o hodnotě 1. Pro definici slave zařízení se pod ModbusMasterX přidávají zařízení pojmenované ve výchozím nastavení ModbusDeviceX. U každého slave zařízení jsem nastavil unikátní adresu pro každé zařízení, na kterou se pak odkazuje komunikační protokol. Pokud bych využil manuální komunikaci, musel bych nastavit hodnotu *ClientLabel* pro definici návěští slave stanice.



Obrázek 5.1 – Vlastnosti Master stanice

5.2 STANICE SLAVE

Vytvořil jsem druhý projekt, který později nahraji do stanice. Pro přidání takového slave zařízení slouží v DetStudiu sekce *Komunikace – Modbus – Přidat Slave*. Po vybrání se přidá zařízení s označením ModbusX. Jelikož nemám více zařízení, které bych potřeboval spojit, ponechávám název nezměněn. Stejně jako u stanice master jsem pro komunikaci prostřednictvím RS485 nastavil SerialPort na hodnotu 1. U tohoto slave zařízení jsem nastavil unikátní adresu stejnou, jako jsem ji nastavoval kapitole 5.1 pro slave zařízení master stanice.

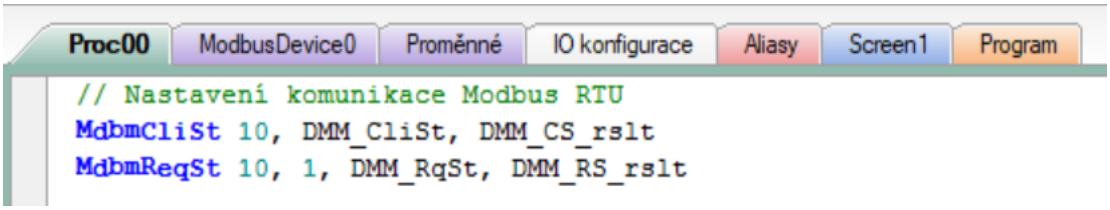


Obrázek 5.2 – Vlastnosti Slave stanice

5.3 SDÍLENÍ HODNOT PROMĚNNÝCH MASTER STANICE

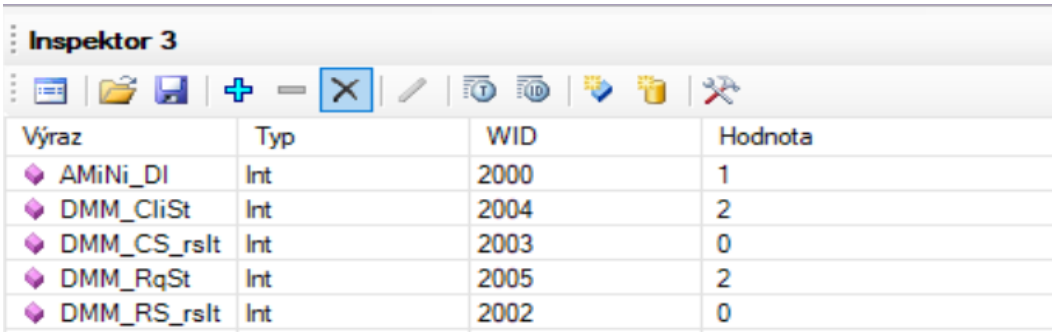
Pro automatickou komunikaci jsem využil knihovnu *MODBUS* a z ní použil funkce *MdbmCliSt* a *MdbmReqSt*. Funkce *MdbmCliSt* slouží pro zjištění stavu klienta. Funkce *MdbmReqSt* slouží pro zjištění stavu komunikace dat. Abych mohl v případě špatné komunikace analyzovat problémy, zadal jsem si proměnné pro zjištění stavu komunikačního

rozhraní *DMM_CliSt*, pro výsledek provedení modulu klienta *DMM_CS_rslt*, pro stav požadavku na data *DMM_RqSt* a pro výsledek provedení modulu požadavku *DMM_RS_rslt*. Pro každou proměnnou, kterou je potřeba komunikovat, jsem zadal návěští definice klienta. Pro identifikaci proměnné jsem zadal návěští požadovaného komunikačního řádku.



Obrázek 5.3 – Nastavení master komunikace

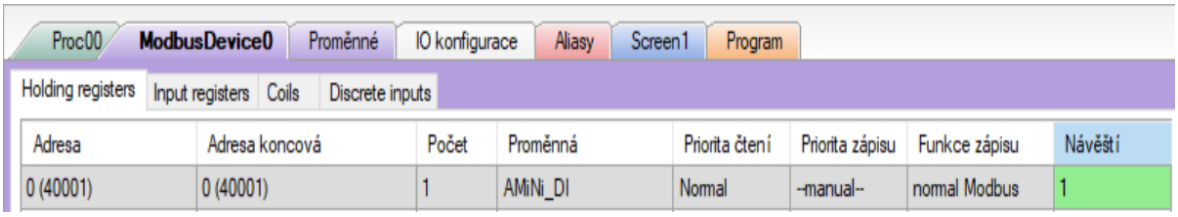
Komunikace je funkční právě tehdy, kdy proměnné, které jsem si zadal, mají hodnoty dle obrázku 5.4.



Výraz	Typ	WID	Hodnota
AMiNi_DI	Int	2000	1
DMM_CliSt	Int	2004	2
DMM_CS_rslt	Int	2003	0
DMM_RqSt	Int	2005	2
DMM_RS_rslt	Int	2002	0

Obrázek 5.4 – Inspektor funkční komunikace

Pro sdílení dat jsem zadal holding registr s adresou 0, který odkazuje na proměnnou *AMiNi_DI*, které je přidělena hodnota z programu slave zařízení. Prioritu pro čtení této hodnoty jsem zadal 1000 milisekund, tedy čtení s periodou jedné vteřiny. Podrobnější popis o hodnotě proměnné *AMiNi_DI* v kapitole 5.2.

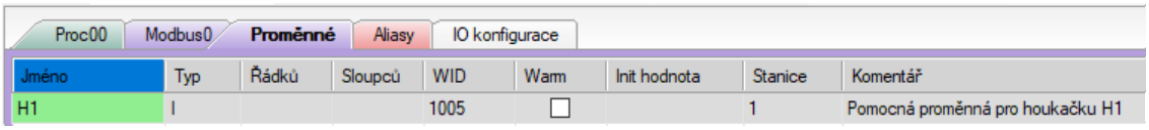


Adresa	Adresa koncová	Počet	Proměnná	Priorita čtení	Priorita zápisu	Funkce zápisu	Návěští
0 (40001)	0 (40001)	1	AMiNi_DI	Normal	--manual--	normal Modbus	1

Obrázek 5.5 – Komunikace holding registru

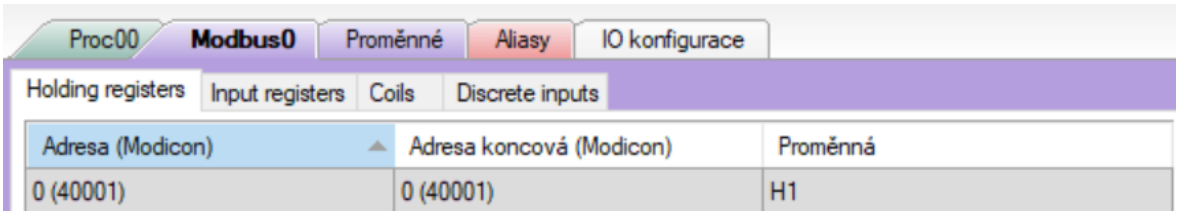
5.4 SDÍLENÍ HODNOT PROMĚNNÝCH SLAVE STANICE

Slave zařízení definuje, jaké proměnné se budou komunikovat. Pro příklad komunikace jsem vytvořil proměnnou H1, kterou budu odesílat do master stanice.



Obrázek 5.6 – Proměnné Slave stanice

Aby došlo ke správné komunikaci, zadal jsem i na slave stanici adresu proměnné 0 s odkazem na vytvořenou proměnnou H1.



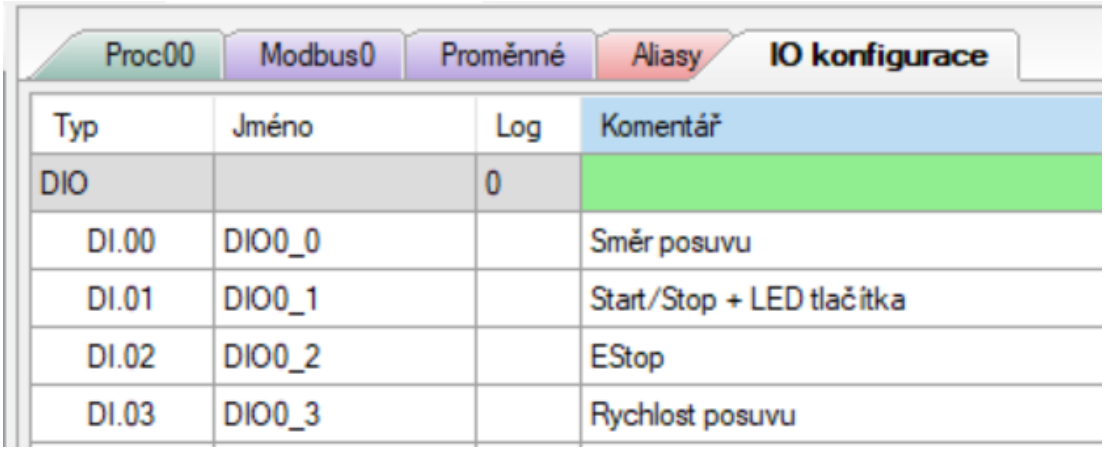
Obrázek 5.7 – Holding registr Slave stanice

6 REALIZACE

Pro tuto práci jsem si jako předlohu vybral pásový dopravník, který bude ovládán sérií přepínačů, tlačítek a houkačkou pro hlášení alarmu. Hlavní, tedy master, stanicí zobrazuji průběh celého programu dopravníku. Slave stanicí určuji její parametry a vše je komunikováno do master stanice pro kontrolu stavu dopravníku. Pro připojení napájení je použito běžné domácí připojení 230 V o frekvenci 50 Hz. To je navedeno na napájecí zdroj AD1048-24FS, který je určen na DIN lištu. Tímto napájecím zdrojem je pak napájen veškerý hardware v rozvaděčové skříni.

6.1 KONFIGURACE IO

Pro slave zařízení jsem využil 4 vstupní hodnoty *DIO0_0* až *DIO0_3*. Na vstup *DIO0_0* jsem připojil dvoupolohový přepínač, kterým ovládám směr vpřed nebo vzad. Pro vstup DIO0_1 jsem využil pro spuštění nebo zastavení pásu. Pro signalizaci běhu jsem využil zelené tlačítko a k tomu současně připojená LED dioda pro signalizaci spuštěného procesu. Vstup DIO0_2 slouží pro vyhodnocení stavu nouzového tlačítka. Pro rychlost posuvu pak



Obrázek 6.1 – IO Konfigurace dopravníku (master)

slouží vstup *DIO0_3*.

Pro master zařízení jsem využil výstupy *DO00_0* a *DO00_1*. Výstupem *DO00_0* ovládám připojenou houkačku, abych signalizoval zamáčknuté nouzové tlačítko (EStop). Výstupem *DO00_1* signalizuji spuštěný program, tedy pohyb dopravníku. Vstupy jsem

nevyužil, protože čtu ze „slave“ stanice potřebná data prostřednictvím Modbus protokolu připojeného komunikací RS485. Podrobnější popis o komunikaci jsem již popsal v kapitole 5.

DO0		0	
DO.00	DO00_0		Alarm houkačka H1
DO.01	DO00_1		Světlo na tlačítku S2 (Start/Stop)

Obrázek 6.2 – IO Konfigurace dopravníku (slave)

6.2 PROMĚNNÉ

Na slave stanici jsem si vystačil s jedinou proměnnou, která obsahuje logické ovládání na binárním stavu holding registru. Tedy pro nultý bit proměnné *H1* odpovídá hodnota pro směr pohybu dopravníku, bit 1 jsem nadefinoval jako *Start/Stop* tlačítko, bit 2 jako nouzové zastavení a bit 3 jako rychlost pobyhu dopravníku.

Proc00 | Modbus0 | **Proměnné** | Aliasy | IO konfigurace

Jméno	Typ	Řádků	Sloupců	WID	Warm	Init hodnota	Stanice	Komentář
H1	I			1005	☐		1	Registr pro komunikaci pomocí RS485

Obrázek 6.3 – Proměnné programu (slave)

Na master stanici bylo zapotřebí více proměnných. Pro kontrolu komunikace RS485 jsem využil proměnné *DMM_CliSt* pro zjištění stavu komunikačního rozhraní, *DMM_CS_rslt* pro výsledek provedení modulu klienta, *DMM_RqSt* pro stav požadavku na data a *DMM_RS_rslt* pro výsledek provedení modulu požadavku.

Pro načítání vstupů slave zařízení využívám proměnnou *AMiNi_DI*. Pro výstupní hodnoty z master zařízení používám proměnnou *AMiNi_DO*. V této proměnné odesílám stav spuštěného tlačítka *Start/Stop* a tím se mi toto tlačítko dle stavu rozsvítí nebo zhasne.

Další využitou proměnnou je *Alarm*, která slouží pro zobrazení chybové hlášky na obrazovce. Proměnnou *Error* využívám pro spínání houkačky při spuštěném nouzovém tlačítku.

Proměnná *S1_Stav* mi slouží pro směr dopravníku. Tato hodnota se načítá ze vstupu slave stanice a podle stavu určuje směr vpřed, nebo vzad. Proměnná *Speed* obsahuje informaci o rychlosti dopravníku a mění se dle otočeného přepínače směru. Proměnná *Speed* je blokována pro vyšší rychlosti dopravníku, pakliže se jeho pozice blíží ke koncovému bodu (vpřed i vzad). Nejdůležitější proměnnou je proměnná *Pozice*. Tato proměnná je závislá téměř na všech proměnných a její hodnotou je definována pozice dopravníku.

Proc00 | ModbusDevice0 | **Proměnné** | IO konfigurace | Aliasy | Screen1 | Proc01

Jméno	Typ	Řádků	Sloupců	WID	Warm	Init hodnota	Stanice	Komentář
Alarm	I			2010	☐		2	Alarm kod
AMiNi_DI	I			2000	☐		2	Digital vstup
AMiNi_DO	I			2006	☐		2	Digital Výstup
DMM_CliSt	I			2004	☐		2	Stav klienta / rozhraní
DMM_CS_rslt	I			2003	☐		2	Výsledek provedení modulu
DMM_RqSt	I			2005	☐		2	Stav požadavku
DMM_RS_rslt	I			2002	☐		2	Výsledek provedení modulu
Error	I			2007	☐		2	Houkačka error
Item_Alarm	I			2001	☐		2	Pomocná proměnná pro Alarm text
Pozice	I			2008	☐		2	Pozice pohybové části
S1_Stav	I			2009	☐		2	Stav S1
Speed	I			2011	☐	1	2	Rychlost čítače

Obrázek 6.4 – Proměnné programu (master)

6.3 ALIASY

Aliasy jsou nedílnou součástí při práci s proměnnými. Pomocí aliasů se lze bitově odkazovat na proměnné a jednodušeji tak pracovat s logickými hodnotami 0 a 1. Každý alias se váže k některé z vytvořených proměnných.

Aliasy stanice master

Pro master stanici jsem využil šesti aliasů, kde každý alias má svou specifickou funkci.

Alias *@Err*, který odkazuje na bit 0 proměnné *Error*, jsem využil pro signalizaci přerušovaného pískání houkačky. Jelikož použitá houkačka přerušovaně houká při stálém signálu, program pro spuštění houkání jen rozhoduje, kdy má být spuštěno houkání a kdy se má vypnout. Pro hodnotu 0 se vypne houkání, pro hodnotu 1 se houkání spustí a houká tak dlouho, dokud se nezmění zpět na hodnotu 0 (například odblokování nouzového tlačítka).

Alias *@H1* odkazuje bit 0 proměnné *AMiNi_DI*. Tento alias jsem využil pro určení směru vpřed nebo vzad, tedy logická 0 vzad, logická 1 vpřed. Logická hodnota je určena přepínačem S1, který je připojený na slave stanici na vstup *DIO0_0*.

Alias *@Run* slouží k určení stavu tlačítka *Start/Stop*. Tedy definuje, kdy je tlačítko stisknuto. Na stavu tohoto aliasu je závislý alias *@RunDO*, který podle stavu stisknutého tlačítka rozsvítí nebo zhasne světlo tlačítka *Start/Stop*. Hodnotu stavu rozsvíceného nebo zhasnutého světla přenáší proměnnou *AMiNi_DO* na bitu 1.

Každá rozvodná skříň by měla obsahovat nouzové tlačítko pro rychlé odpojení. Není tomu jinak ani v mém projektu, a tudíž jsem si vytvořil alias *@EStop*, který signalizuje stav zamáčknutého nouzového tlačítka. Alias odkazuje na proměnnou *AMiNi_DI*, tedy je sdílen slave stanicí. Pokud je nouzové tlačítko stisknuté, zastaví se hlavní program a je signalizován alarm kód na obrazovce a současně je spuštěna zvuková signalizace stopky.

Alias *@Speed* jsem určil pro stav rychlosti pohybu. Pokud je tento alias ve stavu logické 1, přepne se program na rychlost 2 a v opačném případě se přepne na rychlost 1.

Proc00 | ModbusDevice0 | Proměnné | IO konfigurace | **Aliasy** | Screen1 | Program

Alias	Proměnná	Bit	Komentář
@Err	Error	0	Alias pro přerušovaný alarm
@EStop	AMiNi_DI	2	Alias pro EStop
@H1	AMiNi_DI	0	Alias pro stav směru pohybu dopravníku
@Run	AMiNi_DI	1	Alias pro Start/Stop
@RunDO	AMiNi_DO	1	Alias pro světlo Run
@Speed	AMiNi_DI	3	Rychlost dopravníku

Obrázek 6.5 – Aliasy stanice master

Aliasy stanice slave

Pro slave stanici jsem využil čtyř aliasů pro bitovou signalizaci stavů na vstupech slave stanice. Jelikož se funkcí *BinIn* dá přidělit vždy jen jeden binární stav (alias), rozdělil jsem tuto funkci na 4 příkazy, kde každý příkaz obsahuje jeden z aliasů.

Všechny aliasy na slave stanici odkazují na proměnnou *H1*. Každý alias je sdílen master stanici a ta s daty dále pracuje a vyhodnocuje.

Alias *@S1* odkazuje na nultý bit a mění se v závislosti na stavu přepínače S1. Tento alias určuje směr posunu dopravníku. Alias *@S2* odkazuje na bit 1 a jeho stav je určen podle stisknutého tlačítka *Start/Stop*. Alias *@S3* odkazuje na bit 2 a jeho úkolem je odesílání stavu nouzového tlačítka. Alias *@S4* odkazuje na bit 3 proměnné *H1* a mění se podle pozice přepínače pro rychlost.

Proc00 | Modbus0 | Proměnné | **Aliasy** | IO konfigurace

Alias	Proměnná	Bit	Komentář
@S1	H1	0	Alias pro směr posunu dopravníku
@S2	H1	1	Světlo S2 Start/Stop
@S3	H1	2	Alias pro EStop
@S4	H1	3	Alias pro rychlost pohybu

Obrázek 6.6 – Aliasy stanice slave

6.4 OBRAZOVKA

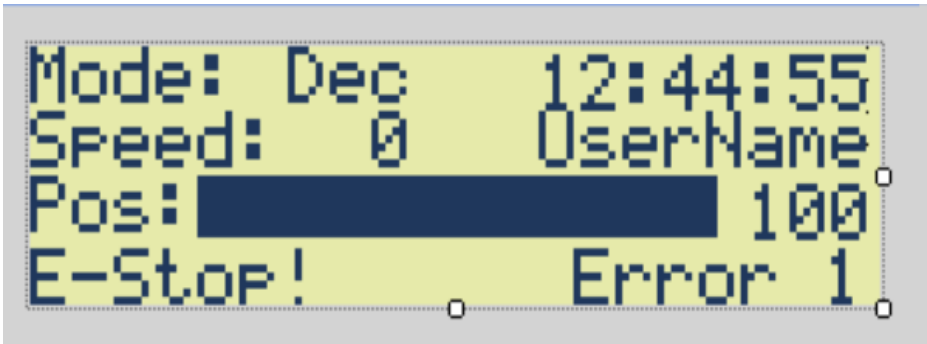
Obrazovku využívám jako výstup své práce a slouží jako kontrola stavu procesu. PLC s obrazovkou funguje jako master stanice, tedy je hlavní řídicí stanicí.

Na prvním řádku zleva mám umístěný *Label* s textem „Mode“. Vedle tohoto textu mám textový řetězec *Alarm2*, který se mění v závislosti na hodnotě proměnné *S1_Stav*. Přepíná mezi textem „Fwd“ a „Bwd“ (Forward and Backward), tedy odčítání pozice pro směr vzad a přičítání pozice pro směr vpřed. Na pravou stranu tohoto řádku jsem nastavil datum, který je synchronizovaný s programem.

Druhý řádek obrazovky z levé strany umístěn *Label* s textem „Speed“. Vedle tohoto textu mám odkaz na hodnotu proměnné *Speed*, tedy se zobrazuje rychlost, která je aktuálně nastavena pro pohyb. V pravé části pod datem jsem zadal jméno autora programu.

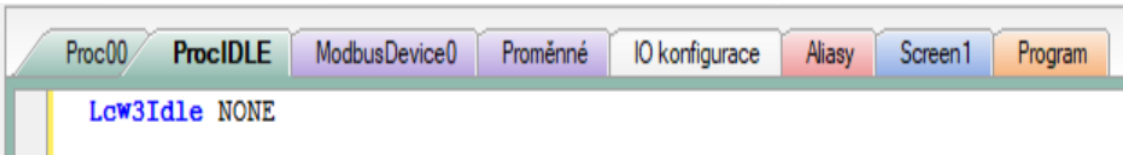
Třetí řádek zobrazuje pozici, na které se aktuálně nachází dopravník. Tedy vlevo jsem zadal text „Pos“ jako pozice v anglickém jazyce. Za tento text jsem vložil ukazatel průběhu pro proměnnou *Pozice*. Na pravé straně kromě grafického znázornění pozice jsem přidal i hodnotu proměnné pro lepší přesnost údaje.

Na posledním řádku zobrazuji alarmové hlášení. Pro zobrazení alarmu využívám proměnnou *Alarm*. Podle hodnoty této proměnné se určuje text, který se má zobrazit na obrazovce. Pro hodnotu 0 se nezobrazí nic. Jedná se tedy o klidový stav a není co hlásit. Pro hodnotu 1, tedy bit 0 ve stavu logické 1, se zobrazí alarmový text „E-Stop!“ v levé části a „Error 1“ v pravé části řádku, který se zobrazuje při stisku nouzového tlačítka. Pro hodnotu 2 jsem nastavil text „Předmět vyložen“, který se zobrazí tehdy, kdy hodnota Pozice dosáhne hodnoty 50 a signalizuje povolení vykládání předmětu. Pro hodnotu 4 se zobrazí text „Předmět naložen“, který signalizuje dojezd na pozici 0 a je povoleno nakládání předmětu. Pro Alarm hodnotu 8 jsem zadal text „Naloženo“, které se zobrazí po naložení předmětu a je zobrazováno po celou dobu pohybu po dopravníku, dokud se nezmění status. Alarm hodnota 16 obsahuje text „Prázdné“ a funguje na stejném principu jako Alarm hodnota 8, jen s rozdílem prázdného dopravníku. Alarm hodnota 32 s textem „Už je naloženo“ je ochranný prvek pro zablokování opakovaného naložení a hlídá, aby se nenaložilo více předmětů najednou a došlo tak k nějaké kolizi. Alarm hodnota 64 s textem „Už je vyloženo“ je stejný případ s rozdílem, že je to pro vykládání.



Obrázek 6.7 – Obrazovka master stanice

Pro přenos dat mezi softwarovou částí řídicí stanice a softwarovou částí pro monitor jsem použil funkci *Lcw3Idle*. Tato funkce se používá pro připojení grafického terminálu parametrizovaného v editoru obrazovek. Do této funkce jsem nezadával žádný parametr. Tento proces slouží pouze pro obsluhu obrazovky.

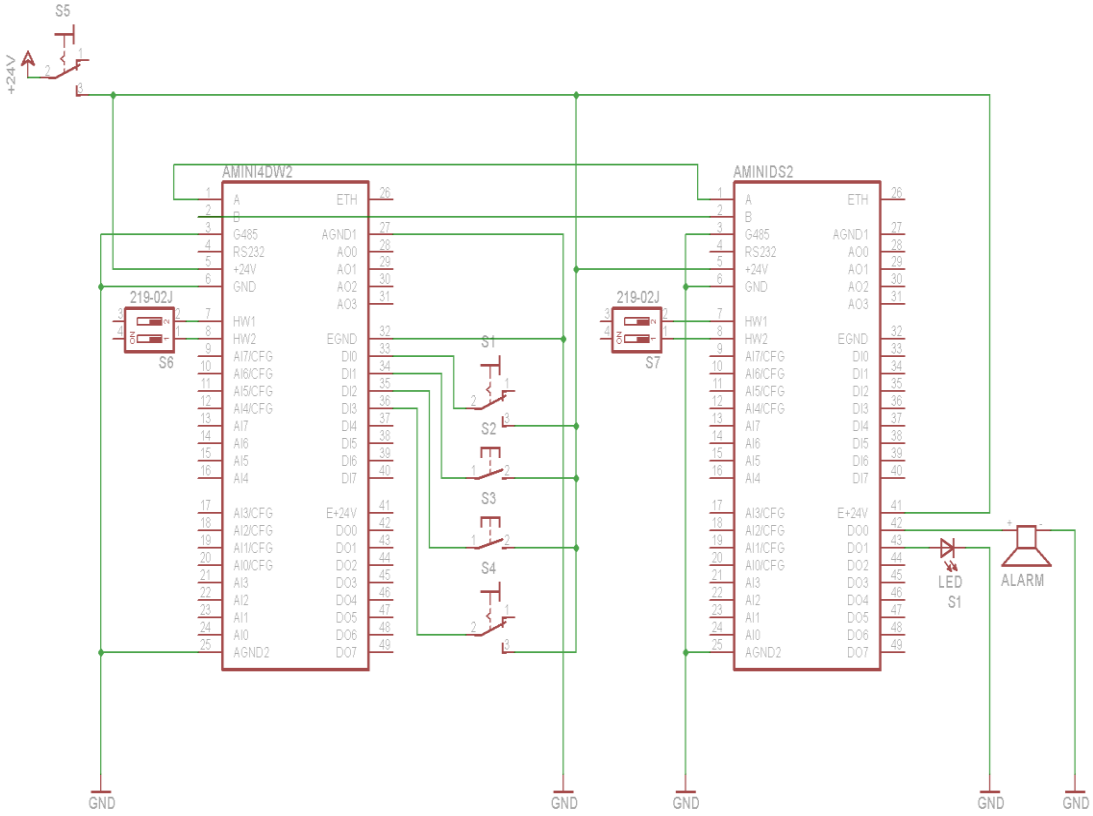


Obrázek 6.8 – Nečinný (Idle) proces pro obrazovku

6.5 NÁVRH PŘIPOJENÍ PLC MODULŮ

Návrh zapojení jsem vytvořil v programu Eagle. PLC jsou napájeny stejnosměrným zdrojem napětí +24 V.

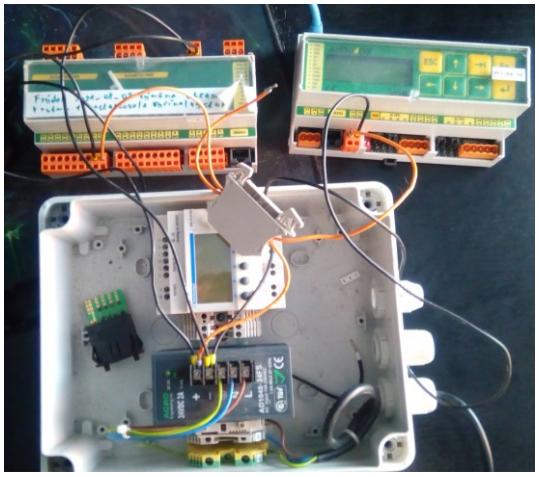
Schéma zapojení pásového dopravníku s komunikací RS485 a možností jeho ovládání na obrázku 6.9.



Obrázek 6.9 – Schéma dopravníku s PLC moduly

Tabulka 6.1 – Seznam součástek pro pásový dopravník

Označení	Typ	Hodnota	Funkce
S1	přepínač	320-916	směr pohybu
S2	tlačítko	31-XX	start / stop
S3	červené tlačítko	31-XX	nouzové zastavení
S4	přepínač	320-916	rychlost pohybu
S5	přepínač	320-916	start PLC
LED S1	LED dioda	LED3MM (LED)	indikátor spuštěného programu

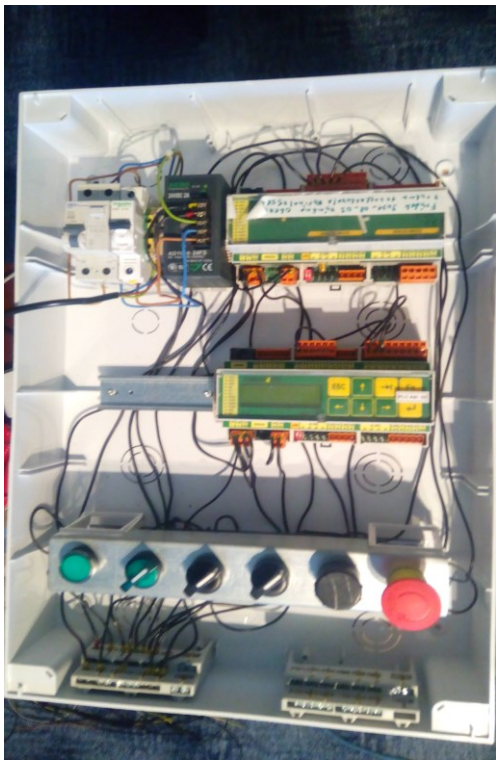


Obrázek 6.10 – Vývoj zapojení

6.6 KOMPLETACE

Na obrázku 6.10 je zobrazen prvotní vývoj mé práce pro otestování funkčnosti PLC modulů.

Pro konstrukci reálného modelu jsem využil plastovou skříň, která se užívá pro běžné rozvaděče, se třemi DIN lištami pro upevnění komponentů. Hlavní elektrický přívod 230 V s frekvencí 50 Hz je naveden na proudový chránič pro ochranu osob. Na proudový chránič je připojen 10A jistič, který chrání připojený obvod. Přes tento 10A jistič je připojen zdroj +24 V, který za pomocí svorkovnic oživuje celý elektrický obvod rozvaděče.



Obrázek 6.11 – Kompletace

7 PROGRAM

Pro přehlednost a kvalitu každého programování je nutné psát komentáře. Začátek každého programu začíná inicializací proměnných nebo nastavení komunikace.

7.1 PROC00

Proměnné mám vytvořené, a tedy na začátek programu zadávám komunikaci mezi stanicemi. Princip této komunikace je popsána v kapitole 5.3.

```
// Nastavení komunikace Modbus RTU
MdbmCliSt 10, DMM_CliSt, DMM_CS_rslt
MdbmReqSt 10, 1, DMM_RqSt, DMM_RS_rslt
```

Obrázek 7.1 – Komunikace programu

Po nastavení komunikace začíná vlastní program. V první části mám vypsaný program pro kontrolu stavu nouzového tlačítka. Začínám podmínkou takovou, kdy je nouzové tlačítko aktivní. Pokud je aktivní, spustí se aliasem @Err zvukový alarm. Zvukový alarm je aktivní tak dlouho, dokud je nouzové tlačítko aktivní. Současně s tímto zvukovým alarmem se v cyklu zobrazuje text alarmu na obrazovce. Hodnota tohoto alarmu pro zobrazení je 1 a hodnotou 0 se zaručuje jeho vypnutí a tím při každém cyklu se vypne nebo zapne a způsobí tak blikání textu.

```
// Kontrola EStop
If @EStop                                   // EStop aktivní
  Let @Err = True                           // Spuštění zvukového alarmu
  If Alarm == 1                             // Blikání Error kódu
    Let Alarm = 0
  Else
    Let Alarm = 1
  EndIf
Else
  Call Program                              // Běh programu
EndIf
BinOut @Err, 0x0000, #DO00_0                // Zvukový výstup Alarmové EStopky
```

Obrázek 7.2 – Program Proc00

Pokud není nouzové tlačítko aktivní, volá se podprogram Proc01. Po dokončení běhu Proc01 se odešle funkcí *BinOut* hodnota pro zvukový výstup alarmu na výstupní kontakt DO00_0. Toto způsobí houkání v případě aktivního nouzového tlačítka, jinak je výstupní hodnota nulová a zvukový výstup se neprojeví.

7.2 PODPROGRAM PROC01

V tomto podprogramu jsem sepsal celou funkci dopravníku. Jelikož aktivní nouzové tlačítko mění některé proměnné, je nutné tyto proměnné vrátit zpět do původní polohy, pokud má běžet program. Pro začátek tohoto podprogramu jsem zadal příkaz pro vynulování hodnoty aliasu pro houkačku @Err. Tím zajistím, aby nehoukala houkačka při spuštěném běžném režimu dopravníku.

```
// Klidové stavy
Let @Err = False                                    // Houkačka
```

Obrázek 7.3 – Klidové stavy Proc01

Další částí podprogramu je řešení ovládání tlačítka *Start/Stop*. Vytvořil jsem podmínku pro stav aktivního tlačítka. Pokud je tlačítko stisknuté, zneguje se alias @RunDO a tím se převrátí výstupní logická hodnota tohoto aliasu. Po této podmínce, ať splněné nebo nesplněné, odešle se stav aliasu @RunDO výstupem DO00_1 a docílí se tak rozsvícení LED na tlačítku *Start/Stop*.

```
// Start-Stop tlačítko S1
If @Run                                 // Stav tlačítka Start/Stop
  Let @RunDO = not @RunDO               // Světlo tlačítka Start/Stop
EndIf
BinOut @RunDO, 0x0000, #DO00_1          // Odeslání stavu Start/Stop tlačítka na světlo tlačítka
```

Obrázek 7.4 – Start/Stop tlačítko Proc01

Změnu rychlosti dopravníku jsem umožnil pomocí přepínače S3 a to tak, že pokud bude přepínač v zapnuté poloze, odešle se binární logický stav 1 do master stanice a na alias @Speed se zapíše logická 1. Pokud má alias @Speed logickou hodnotu 1 nastaví se rychlost 2, jinak se nastaví na rychlost 1.

```
// Rychlost pohybu
If @Speed                                   // Rychlost 2 při S3 = True, jinak rychlost 1
  Let Speed = 2
Else
  Let Speed = 1
EndIf
```

Obrázek 7.5 – Rychlost pohybu Proc01

Pro zpomalení u konců dráhy jsem nastavil rozmezí pro dojezdy a pomalé starty od krajů. Od pozice 0 do pozice 5 je rychlost pohybu dopravníku nastavena na 1 i při přepínači rychlosti v pozici logické 1. To samé pak pro pozici 45 až 50. Pro oba směry je tato rychlost omezena pro bezpečnost předmětu proti rychlému zastavení nebo poškození dojezdových brzd.

Funkce zpomalení pro dojezdy je aktivní pouze při spuštěném programu. Pokud je program spuštěn pro pohyb a rychlost je nastavena přepínačem na rychlost 2, pak se posuzuje, zda je pozice větší než 44, nebo menší než 6, aby se zpomalil dojezd.

```
// Dojezdy
If @RunDO                                   // Stav světla Start/Stop tlačítka, tedy běžící program
  If Speed == 2                             // Pro rychlý posuv
    If Pozice > 44                          // Nad 44 se zpomalí na rychlost 1
      Let Speed = 1                         // Rychlost 1
    EndIf
    If Pozice < 6                           // Pod 6 se zpomalí na rychlost 1
      Let Speed = 1                         // Rychlost 1
    EndIf
  EndIf
```

Obrázek 7.6 – Dojezdy dopravníku

Směr pohybu a pohyb dopravníku je určen podle stavu proměnné S1_Stav, kterou mohu ovládat přepínačem S1. Při směru vpřed se posouvá pozice o velikost proměnné Speed. Pro reálné zařízení pak tuto hodnotu pozice je nutné načítat snímačem. V případě, že hodnota pozice přesáhne hodnotu 49, nastaví se proměnná Item_Alarm na hodnotu 0, která symbolizuje vyložení předmětu z dopravníku a zároveň určuje text na obrazovce. Po dosažení hodnoty 50 se zastaví pohyb dopravníku a čeká na manuální zásah. Zastavení dopravníku lze zautomatizovat podle získané informace o přijatém předmětu či odebraném předmětu z dopravníku a docílit tak automatického chodu pro přesun jednoho předmětu z bodu 0 do bodu 50.

Pokud je proměnná S1_Stav na hodnotě 0, pozice dopravníku je odčítána o velikost proměnné Speed. Když pozice dosáhne hodnoty 0, proměnná Item_Alarm se nastaví na hodnotu 1 a dojde tak k naložení předmětu na dopravník. Program se na této pozici zastaví stejně jako v bodě 50.

```
// Směr pohybu a pohyb
  If S1_Stav == 1                    // Stav směru posuvu
    If Pozice > 49                   // Stop na konci
      Let Item_Alarm = 0             // Odebrání předmětu z dopravníku
      Let @RunDO = False             // Vypnutí běhu programu, čeká na další Start/Stop impulz
    Else
      Let Pozice = Pozice + Speed    // Posuv pozice pozice vpřed
    EndIf
  Else
    If Pozice < 1                    // Stop na konci
      Let Item_Alarm = 1             // Naložení předmětu na dopravník
      Let @RunDO = False             // Vypnutí běhu programu, čeká na další Start/Stop impulz
    Else
      Let Pozice = Pozice - Speed    // Posuv pozice pozice vzad
    EndIf
  EndIf
```

Obrázek 7.7 – Směr pohybu a pohyb dopravníku

Alarm texty mám definované podle pozičních podmínek a zároveň podle naloženého či prázdného dopravníku. Pokud je pozice v rozmezí 1 až 49 a zároveň je naložen dopravník, vypíše se text hodnoty proměnné Alarm bit 8 na obrazovce. Alarm bit 8 obsahuje text „Naloženo“. Pokud není naložen dopravník, vypíše se text proměnné Alarm bit 16. Text bitu 16 proměnné Alarm jsem zadal „Prázdné“.

```
// Alarm texty
  If (Pozice > 0) And (Pozice < 50)      // Při pohybu zobrazení stavu
    If Item_Alarm == 1                   // Pokud je naložen
      Let Alarm = 8                      // Napiš je naložen
    EndIf
    If Item_Alarm == 0                   // Pokud je prázdný
      Let Alarm = 16                     // Napiš je prázdný
    EndIf
  EndIf
  If Pozice == 0                         // Dojezd vlevo
    If Item_Alarm == 1                   // Pokud je naložen
      Let Alarm = 32                     // Už je naložený
    Else
      Let Alarm = 4                      // Nakládání
    EndIf
  EndIf
  If Pozice == 50                        // Dojezd vpravo
    If Item_Alarm == 0                   // Pokud je prázdný
      Let Alarm = 64                     // Už je prázdný
    Else
      Let Alarm = 2                      // Vyprázdnit
    EndIf
  EndIf
```

Obrázek 7.8 – Alarm texty

Alarm bit 32 s textem „Už je naloženo“ se vypíše v případě proměnné Pozice hodnoty 0 při naloženém dopravníku. Při prázdném dopravníku se při hodnotě 0 vypíše alarm bit 4, který obsahuje text „Předmět naložen“.

Pokud dopravník dosáhne hodnotu Pozice 50 a je naložený, vypíše se Alarm bit 2 obsahující text „Předmět vyložen“. Pokud je prázdný, vypíše se Alarm bit 64 s textem „Už je vyloženo“.

Všechny tyto části programu fungují a mění parametry pouze při spuštěném běhu programu. To znamená, že směr pohybu dopravníku se nedá změnit, dokud se dopravník pohybuje. Musí se nejprve zastavit běh programu a teprve poté lze změnit směr pohybu. Směr pohybu se mění přepínačem S1, tudíž podle stavu proměnné AMiNi_DI bit 0 se rozhoduje o změně směru.

```
// Pouze při zastaveném běhu lze měnit směr
Else
  If @H1
    Let S1_Stav = 1                    // Směr vpřed (Fwd)
  Else
    Let S1_Stav = 0                    // Směr vzad (Bwd)
  EndIf
EndIf
```

Obrázek 7.9 – Změna směru pohybu dopravníku

ZÁVĚR

Cílem této práce bylo provedení laboratorního modulu pro simulaci chování reálného zařízení. Vytvořit komunikaci mezi PLC moduly obnášelo dlouhou přípravu po teoretické stránce. Není dostatek veřejných informací ohledně propojení mezi moduly nebo jsou cenově příliš drahé pro studijní poměry.

První problém spočíval v obstarání dvou PLC modulů ke studijním či pracovním účelům. Nejedná se o levné produkty, které by měla doma každá domácnost a zároveň takové produkty, které jsou navzájem kompatibilní v komunikaci.

Druhým hlavním problémem bylo samotné propojení PLC modulu s počítačem, neboť pro komunikaci je zapotřebí znát nastavení PLC od výrobce, nebo v mém případě získat tyto informace od osoby, která tento modul již užívala a změnila původní nastavení. Zapotřebí bylo nutné zjistit IP adresu stanice a připojit se na stejnou podsíť pro možnost její konfigurace. Propojení pomocí protokolu modbus RTU bylo po sérii několika neúspěšných testů provedeno v pořádku.

Výsledkem této práce je funkční laboratorní model komunikace dvou PLC modulů, které mohou ovládat pásový dopravník. Dopravník lze zastavit v jakékoli pozici, měnit směr při zastaveném stavu, měnit rychlost pohybu, zastavit program pomocí nouzového tlačítka a obsahuje pomalé dojezdy. Stav je možné sledovat na obrazovce a je vybaven alarmovým hlášením se zvukovým signálem. Program lze libovolně upravit a přizpůsobit jinému reálnému zařízení nebo vylepšit stávající program pro připojení dalších periferií.

POUŽITÁ LITERATURA

ACRO Engineering Inc. 2017. *AD1048FS Series* [online]. Acro Engineering Incorporation [online]. [cit. 2022-04-20]. Dostupné z https://www.dialcomp.hu/letoltes/Acro/AD1048FS%20Series.pdf

AMiT. 2022. *AMiNi4DW2* [online]. AMiT Automation [online]. [cit. 2022-04-14]. Dostupné z https://amitomation.cz/produkt/ridici-systemy/amini4dw2g/

AMiT. 2022. *Řídicí systémy* [online]. AMiT Automation [online]. [cit. 2022-04-14]. Dostupné z https://amitomation.cz/produkt/ridici-systemy/

AspenCore. 2022. *Analogue to Digital Converter* [online]. Electronics Tutorials [online]. [cit. 2022-04-15]. Dostupné z https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html

BUDIMIR, Miles. 2017. *What do PLC watchdog timers do?* [online]. Motion control tips [online]. [cit. 2022-04-17]. Dostupné z https://www.motioncontroltips.com/plc-watchdog-timers/

CORRIGAN, Steve. 2002. *Introduction to the Controller Area Network (CAN)* [online]. Texas Instruments [online]. [cit. 2022-04-19]. Dostupné z https://www.ti.com/lit/an/sloa101b/sloa101b.pdf

Domat Control System s.r.o. 2020. *Komunikační protokol Modbus v kostce* [online]. Tzbinfo [online]. [cit. 2022-04-17]. Dostupné z https://elektro.tzb-info.cz/126361-komunikacni-protokol-modbus-v-kostce

DUDÁČEK, K. 2001. *Mikrokontroléry* [online]. Západočeská univerzita v Plzni [online]. [cit. 2022-04-13]. Dostupné z http://home.zcu.cz/~dudacek/Pot/mikrokontrolery.pdf

Gadget-info.com. 2019. *Rozdíl mezi mikroprocesorem a mikrokontrolérem* [online]. Gadget-info.com [online]. [cit. 2022-04-12]. Dostupné z https://cs.gadget-info.com/difference-between-microprocessor

GOH, Hui Hwang. 2017. *Types of Circuit Breaker and its Application in Substation Protection* [online]. ResearchGate [online]. [cit. 2022-04-20]. Dostupné z https://www.researchgate.net/publication/322029193_Types_of_Circuit_Breaker_and_its_Application_in_Substation_Protection

Ieec.uned.es. 2006. *Introduction to Programmable Logic Controllers (PLC's)* [online]. UNED [online]. [cit. 2022-04-22]. Dostupné z http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/introtoplcs_SUPER.pdf

JACKSON, D. J. 2016. *Programmable Logic Controllers* [online]. Electrical and Computer Engineering [online]. [cit. 2022-04-16]. Dostupné z http://jjackson.eng.ua.edu/courses/ece485/lectures/LECT13.pdf

JIMBLOM. 2013. *Button and Switch Basics* [online]. Sparkfun [online]. [cit. 2022-04-21]. Dostupné z https://learn.sparkfun.com/tutorials/button-and-switch-basics

Mitsubishi Electric Corporation. 2014. *PLC Ethernet* [online]. Mitsubishi Electric [online]. [cit. 2022-04-15]. Dostupné z https://www.mitsubishielectric.com/fa/assist/e-learning/pdf/eng/1-Ethernet_na_eng.pdf

Modicon Inc. 1996. *Modicon Modbus Protocol Reference Guide* [online]. Modbus, North Andover (Massachusetts) [online]. [cit. 2022-04-18]. Dostupné z https://modbus.org/docs/PI_MBUS_300.pdf

OLMR, Vít. 2005. *HW server představuje – Sériová linka RS-232* [online]. Vyvoj.hw.cz [online]. [cit. 2022-04-12]. Dostupné z https://vyvoj.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html

PATEL, Suhel. 2022. *Watchdog Timer in PLC* [online]. Inst Tools [online]. [cit. 2022-04-16]. Dostupné z https://instrumentationtools.com/watchdog-timer-in-plc/

POJAR, Petr. 2018. *Jak si sestavit domovní rozvaděč snadno a bez starostí* [online]. ČESKÉSTAVBY.cz [online]. [cit. 2022-04-20]. Dostupné z https://www.ceskestavby.cz/clanky/jak-si-sestavit-domovni-rozvadec-snadno-a-bez-starosti-25653.html

SANAMRAO123. 2015. *Programming Counters* [online]. Msn [online]. [cit. 2022-04-16]. Dostupné z https://sanamrao123.files.wordpress.com/2015/04/ch-8.pdf

SCHLAEPFER, Eric. 2008. *Comparison of Internal and External Watchdog Timers* [online]. Maxim Integrated [online]. [cit. 2022-04-19]. Dostupné z https://pdfserv.maximintegrated.com/en/an/AN4229.pdf

SONNENBERG, John. 2019. *Serial Communications RS232, RS485, RS422* [online]. Raveon [online]. [cit. 2022-04-21]. Dostupné z https://www.raveon.com/wp-content/uploads/2019/01/AN236SerialComm.pdf

ŠÍL, Petr. 2012. *Řízení manipulátoru pomocí mikrokontroléru* [online]. Brno. [cit. 2022-04-13]. Dostupné z https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=55172. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce DANIEL ZUTH.

Tedia spol. s.r.o. 2018. *Obecné vlastnosti komunikačních prostředků* [online]. TEDIA [online]. [cit. 2022-04-22]. Dostupné z https://www.tedia.cz/podpora/komunikace-obecne-vlastnosti.html

VIDIM, Jan. 2011. *Instalace komunikační sběrnice RS485 v budovách* [online]. Komunikačné systémy [online]. [cit. 2022-04-17]. Dostupné z https://domat.blob.core.windows.net/cms/UserFiles/wp-content/uploads/2011_01_IDB_RS485_SK.pdf

PŘÍLOHY

Příloha A – CD

Příloha A

Příloha k bakalářské práci

Model reálného zařízení realizovaný s využitím mikrokontroléru

Martin Obolecký

CD

PŘÍLOHA A – CD

Obsah

1 Text bakalářské práce ve formátu PDF.

2 Řídicí programy PLC modulů

3 Schémata zapojení