Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Softwarové vybavení realizující most Modbus - HART

Plzeň 2018 Zdeněk Bečvář

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 28. června 2018

Zdeněk Bečvář

Abstract

The purpose of this bachelor thesis is to create bridge between two industrial communication protocols, Modbus - HART. In the first part of this thesis there is theoretical information about industry communication protocols and their properties. The second part describes a software implementation in C programming language and testing functionality of the Modbus - HART bridge.

Abstrakt

Cílem této bakalářské práce je vytvořit programový nástroj pro spojení dvou průmyslových komunikačních protokolů, a to protokolu Modbus a protokolu HART. V první části práce jsou teoreticky popsány průmyslové komunikační sítě a jejich vlastnosti. Druhá část práce popisuje implementaci naprogramovaného mostu Modbus -HART v jazyku C a otestování jeho funkčnosti.

Obsah

1 Úvod

V minulém století zažila automatizace velký vzestup a vývoj. S minimalizací elektroniky a vyšším výpočetním výkonem integrovaných obvodů a procesorů přišel požadavek na distribuované řízení procesů. Pro tento účel byly vyvinuty průmyslové sběrnice. Avšak vývoj byl v relativně krátkém čase značně překotný a vzniklo několik desítek navzájem nekompatibilních sběrnic. Tato práce se zabývá spojením dvou takových sběrnic.

V kapitole 2 si nejprve přiblížíme stručný vývoj průmyslových komunikačních sběrnic, následně se zaměříme na všeobecné znalosti o sítích a komunikacích a podrobněji si popíšeme sběrnice Modbus a HART.

V kapitole 3 si popíšeme způsob implementace mostu Modbus - HART, návrh programového vybavení a následně si tento program otestujeme. Závěr kapitoly bude věnován uživatelské příručce, kde bude vysvětleno, jak s programem pracovat.

2 Průmyslové komunikace

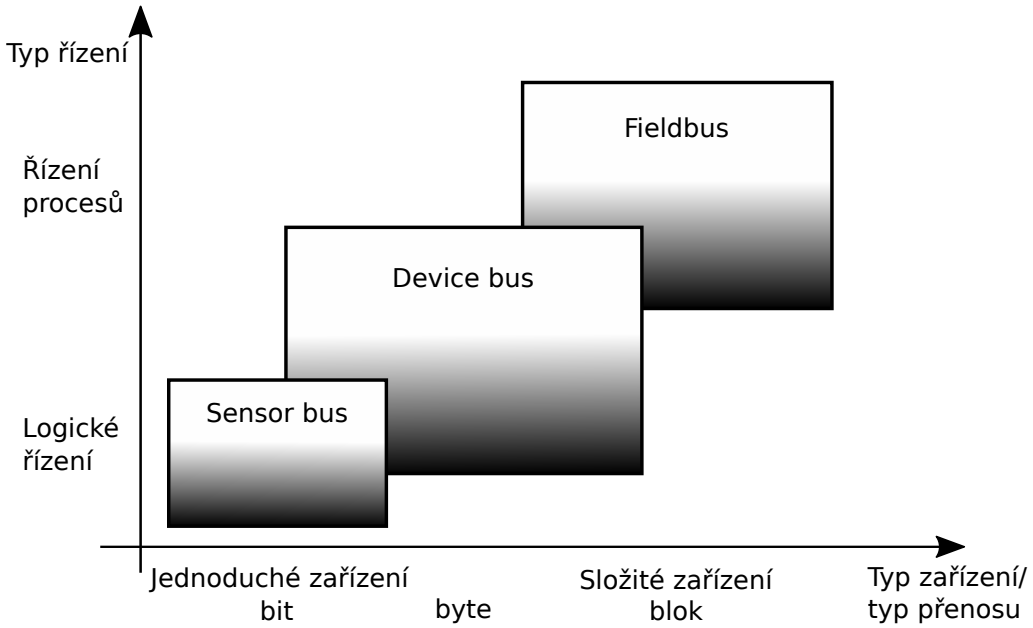
Jak již to tak bývá, s válkou jde ruku v ruce technologický vývoj. A po 2. světové válce se začala výrazně vyvíjet průmyslová automatizace. Do jisté míry v těsném závěsu za výpočetní technikou. V 50. a 60. letech zažívá automatizace bouřlivý rozvoj. V 50. letech má hlavní slovo releová technika a v 60. letech s rozvojem polovodičové elektroniky se začíná centralizovaně řídit počítačem. S dalším rozvojem polovodičové elektroniky se integrované obvody stále zmenšují a jsou výkonnější. V 80. letech přicházejí integrované obvody s vysokou integrací a začíná rozvoj mikroprocesorové techniky. Mikroprocesory jsou již tak výkonné, že začínají pronikat do nejnižších pater úrovní řízení. Uskutečňují tak přímé logické řízení i přímé řízení regulačních smyček (ventilů, klapek, pohonů, os obráběcích strojů atd.) reprezentované zejména programovatelnými automaty PLC. O řízení se přestává starat centrální řídící počítač a řídící systémy se stávají distribuované. [10]

Před příchodem distribuovaných systémů byla převažujícím způsobem zapojení proudová smyčka. Předností proudové smyčky byla odolnost proti rušení při malých rychlostech a vzdálenostech až stovky metrů. Pouze dvoubodové propojení (ke každému čidlu a akčnímu členu vlastní dvoudrátový spoj) a malé rychlosti ale byly pro nastupující distribuované systémy nedostatečné. Proto se v 80. letech vyvíjí speciální sériové sběrnice. [8]

2.1 Rozdělení průmyslových sběrnic

Sériové průmyslové sběrnice jsou přizpůsobené potřebám první úrovně řízení technologických procesů (programovatelné automaty, mikrořadiče, distribuované řídící stanice, inteligentní čidla, akční členy, distribuované inteligentní svorkovnice, odloučené karty vstupů a výstupů atd.), pro komunikaci mezi řídícími členy navzájem a také pro komunikaci s vyšší, operátorskou úrovní. Sběrnice jsou rozděleny do několika typů, a to Sensor/aktor bus, Device bus a Fieldbus. Sběrnice nejsou striktně oddělené a prolínají se, což je vidět na obrázku 2.1. Daný standard lze tedy zařadit do obou sousedních skupin. Např. sběrnice Interbus je někdy charakterizována jako typický Sensor bus, jindy pak jako Device bus. Na druhou stranu sběrnici HART nelze považovat jako Fieldbus. [8]

Sensor bus jsou sběrnice pro komunikaci v reálném čase se senzory a jednoduchými akčními členy. Z referenčního modelu ISO/OSI obvykle definují pouze dvě nejnižší úrovně, tedy fyzickou a linkovou úroveň. Referenční model



Obrázek 2.1: Rozdělení průmyslových sběrnic [8]

ISO/OSI bude podrobněji rozebrán v kapitole 2.3. Sensor bus posílá krátké rámce a přenos rámce je velmi rychlý. [9]

Pro komunikaci na úrovni programovatelných automatů se používají sběrnice Device bus. Opět definují dvě nejnižší vrstvy modelu ISO/OSI, ale ještě přidávají vrstvu nejvyšší, aplikační. Device bus používá delší rámce než Sensor bus umožnující konfiguraci akčních členů a senzorů. Při zachování relativně nízké ceny lze také účinně řídit komplexní procesy. [9]

Poslední a hierarchicky nejvyšší sběrnicí je Fieldbus. Z modelu ISO/OSI definují také tři stejné vrstvy jako Device bus, ale například Foundation Fieldbus navíc přidávají osmou, uživatelskou vrstvu. Sběrnice jsou typu multimaster s redundancí (zdvojení) přenosových linek pro větší spolehlivost a bezpečnost přenosu. Umožňují událostmi řízené služby, objektově orientované přenosy dat a proměnných, funkce pro zprávu sítě atd. [9]

2.2 IEC 61158

V průběhu 80. a 90. let došlo k rozvoji několika desítek různorodých průmyslových sběrnic. Některé sběrnice vznikly jako národní, některé jako firemní standardy s různým světovým rozšířením. Například Profibus (Proces Fieldbus) byl vyvinut s podporou několika významných německých firem pod koordinací DBFT (Německé federální ministerstvo výzkumu a technologie), sběrnici FIP (Factory Instrumentation Protocol) vyvinula skupina

francouzských, německých a italských firem a stala se standardem především ve Francii, sběrnice CAN (Controller Area Network) byla vyvinuta firmou Bosch na technologii Intel a je velmi populární v automobilovém průmyslu. [8]

Pozice těchto sériových sběrnic byla velkým množstvím poněkud oslabována, byť konkurenční prostředí je vždy výhodné pro všechny strany. Zde ale bylo konkurenční až moc a organizace IEC (Mezinárodní elektrotechnická komise) nastavila podmínky a pravidla pro vytvoření celosvětového standardu. Úkolu se chopila americká společnost FieldComm Group, která se snažila splnit požadavky společnosti IEC. Ovšem se sběrnicí Foundation Fieldbus přišla až v roce 1996 a to bylo již pozdě. Sběrnice se sice uchytila zejména, v Severní Americe, ale ne celosvětově.

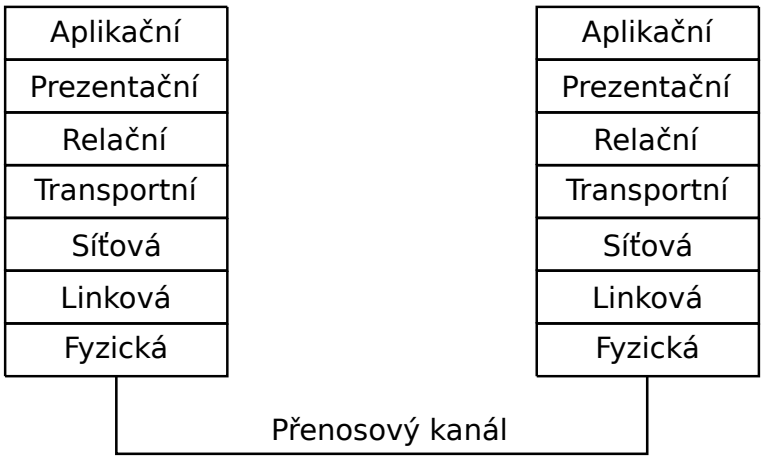
V roce 1999 IEC ukončila období volného vývoje průmyslových sériových sběrnic tím, že připustila do mezinárodního standardu IEC 61158 většinu existujících národních a proprietárních standardů. Jde o návrh označovaný jako „Multi-system-standard“, který zahrnuje osm průmyslových sběrnic. [10] Některé sběrnice byly již v roce 1997 začleněny do evropské normy EN 50170 organizace CENELEC (Evropský výbor pro normalizaci), avšak po deseti letech byla norma zrušena a i v Evropě byla převzata zmíněná norma IEC 61158 jako norma EN 61158. Tato norma byla rozšířena na konečných šestnáct průmyslových sběrnic, kdy některé sběrnice z původní normy byly nahrazeny jinými. V konečné šestnáctce se nacházejí i sběrnice Modbus a HART.

2.3 Referenční model ISO/OSI

Referenční model ISO/OSI vypracovala na začátku 80. let minulého století mezinárodní standardizační organizace ISO jako normu ISO 7498 pro spojení různorodých počítačových systémů. Model byl vyvíjen bez závislosti na nějakém firemním řešení, jedná se tedy o model otevřený. Při dodržení podmínek stanovených modelem mohou různí účastníci mezi sebou navzájem spolehlivě komunikovat. [8, 11]

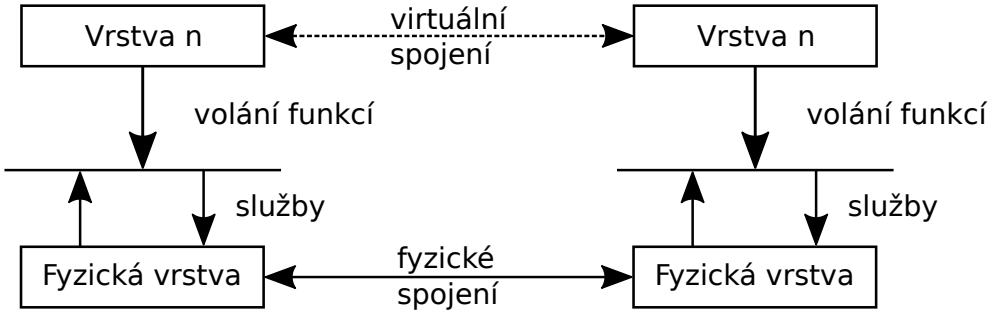
Model je sedmivrstvý a jeho strukturu lze vidět na obrázku 2.2. Každá vrstva má svojí funkci a služby. Zpráva od odesílatele jde od aplikační vrstvy směrem dolů přes ostatní vrstvy a přenese se přenosovým médiem. U příjemce jde od nejnižší vrstvy po vrstvu aplikační, kde jsou již data tak, jak je odesílatel poslal. Každá vrstva si ke zprávě z vyšší vrstvy přidá svá data, takže přenosovým médiem jde delší zpráva, než jakou odesílatel poslal. Každá vrstva na straně odesílatele při komunikaci naváže virtuální spojení

(obrázek 2.3) se stejnou vrstvou na straně příjemce a o dalších vrstvách neví. Příslušná vrstva pracuje s daty jen té vrstvě určené. Výsledná zpráva se přenáší fyzickým médiem ať pevným, nebo bezdrátovým.



Obrázek 2.2: Referenční model ISO/OSI

Každá vrstva má dvě základní funkce. První jsou služby a druhá je protokol vrstvy. Protokol je soubor pravidel, podle kterých probíhá komunikace mezi účastníky přenosu. Dosah každé vrstvy je omezen na vrstvu nejblíže vyšší a nižší na vlastní straně a na stejnou vrstvu jiného účastníka. Při odesílání zprávy vyšší vrstva volá funkci vrstvy nižší, a naopak při příjmu zprávy poskytuje nižší vrstva svoje služby vrstvě vyšší, jak lze vidět na obrázku 2.3. Uvnitř vrstvy se pak vykonává funkce příslušející dané vrstvě a vlastní protokol. [8]



Obrázek 2.3: Přenos zprávy mezi vrstvami OSI modelu [11]

Obecně lze říci, že nejnižší vrstvy 1 až 3 jsou vrstvy spojené s komunikačním procesem. Nejvyšší vrstvy 5 až 7 pak úzce souvisí s aplikačním programem. Poslední vrstva, tedy vrstva transportní, uskutečňuje přechod mezi těmito podsystémy.[8] Například nejpoužívanější internetový protokol,

Ethernetový model TCP/IP, vyvinutý o rok dříve než model ISO/OSI, používá jen 4 vrstvy. V podstatě nejvyšší tři spojuje do jedné - aplikační a nejnižší dvě do síťového rozhraní. Síťovou a transportní vrstvu již nelze s ničím spojit.

Průmyslové sítě většinou také nepoužívají všech sedm vrstev modelu ISO/OSI, ale jen dvě až tři, jak bylo popsáno v kapitole 2.1. A to dvě nejnižší vrstvy, fyzické a transportní, a vrstvu nejvyšší, vrstvu aplikační. Výjimkou v použití modelu ISO/OSI je Foundation Fieldbus, který si ke stávajícím vrstvám přidává osmou, uživatelskou vrstvu.

2.3.1 Fyzická vrstva

Nejnižší vrstva modelu ISO/OSI zajišťuje komunikujícím účastníkům sítě fyzický přenos dat a definuje elektrické a mechanické parametry přenosového kanálu. Fyzická vrstva přenáší zprávy formou elektrického signálu (optický, rádiový, napěťový, proudový) bit po bitu a představuje rozhraní mezi fyzickým spojem a linkovou vrstvou. [8]

Mechanické a elektrické vlastnosti fyzické vrstvy [8]:

- rozměry a vlastnosti konektorů
- kabeláž a topologie (provedení kabelu, materiál, průřez vodičů, stínění, kapacita, indukčnost, odpor atd.)
- kódování bitů (RZ, NRZ, RZI, NRZI, Manchester atd.)
- určení logických úrovní (napěťové, proudové, frekvenční atd.)

2.3.2 Linková vrstva

Je-li na síti více účastníků, může snadno dojít ke kolizi v případě, že budou všichni naráz vysílat. Na sběrnici jsou pak neplatné údaje a žádnému účastníkovi se jeho zprávu nepodaří odvysílat. Linková vrstva tedy stanoví přístupovou metodu, která účastníkovi umožní získat oprávnění vysílat zprávu.[8]

Na přístupové metodě závisí způsob přenosu informací mezi vysílacími stanicemi. Přenos dat se provádí podle způsobu adresování [8]:

- zdroj/cíl
- master/slave
- peer to peer
- producent/konzument

Služby linkové vrstvy [8]:

- přístup k médiu
- přenos ucelených rámců
- základní zabezpečení proti chybám přenosu
- detekce základních chyb
- potvrzení správně přijatých rámců
- opětovné poslání poškozených rámců

2.3.3 Síťová vrstva

Síťová vrstva zajišťuje adresování a specifikuje formát adres. V některých sítích lze jednu stanici adresovat více způsoby, lze definovat skupiny adres (multicast) nebo společnou adresu pro všechny účastníky dané sítě (broadcast). V průmyslových sítích jsou topologie celkem jednoduché, proto je síťová vrstva vynechána a její služby poskytuje aplikační vrstva. [8]

2.3.4 Transportní vrstva

Transportní vrstva zajišťuje bezchybný a spolehlivý přenos dat, fragmentaci a následnou defragmentaci dlouhých zpráv, detekci pokročilejších chyb než v linkové vrstvě a opakované poslání poškozeného rámce. V průmyslových sítích jsou protokoly a přenášené zprávy celkem jednoduché, proto je transportní vrstva vynechána a její služby poskytuje aplikační vrstva. [8]

2.3.5 Relační vrstva

Relační vrstva vyjednává, navazuje, udržuje a ukončuje spojení mezi účastníky. Využívá služeb transportní vrstvy pro bezpečný přenos dat a zajišťuje požadované funkce na spojení pro prezentační vrstvu. V průmyslových sítích jsou přenášené zprávy celkem jednoduché a pro přenos každého rámce je navázáno samostatné spojení, proto je relační vrstva vynechána a její služby poskytuje aplikační vrstva. [8]

2.3.6 Prezentační vrstva

Různí účastníci v síti mají různý způsob zpracování a posílání dat (např. big-endian vs. little-endian), proto prezentační vrstva převádí posílané zprávy

do vhodného formátu pro zpracování aplikační vrstvou. Tato vrstva se také stará o zabezpečení komunikace šifrováním a dešifrováním zpráv. V průmyslových sítích je prezentační vrstva vynechána a její služby poskytuje aplikační vrstva. [8]

2.3.7 Aplikační vrstva

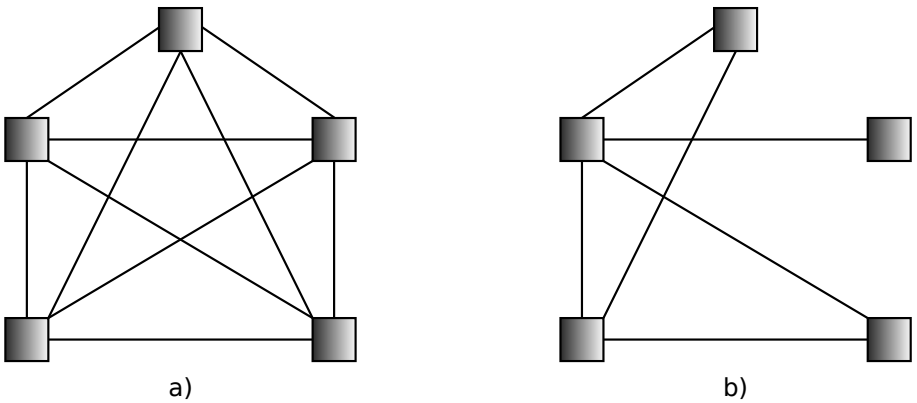
Aplikační vrstva poskytuje komunikační rozhraní aplikaci. Zajišťuje přenos dat mezi účastníky, definuje význam přenášených zpráv a datové typy. Tato vrstva zajišťuje služby všech nižších vrstev, které nebyly protokolem implementovány, jelikož je vrstvou poslední, nejvyšší. [8]

2.4 Architektury sítí

Propojení různých prvků do jednoho celku se nazývá architektura sítě. Struktura a tvar sítě mají zásadní vliv na rozšiřitelnost sítě o další prvek, odolnost sítě proti výpadku a její případnou obnovu.

Polygonální síť

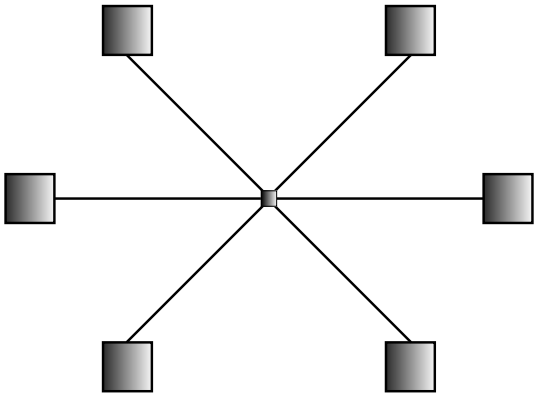
Polygonální síť lze dělit na úplnou a neúplnou. Oba případy jsou znázorněny na obrázku2.4. V úplné polygonální síti je každý prvek propojen s každým. Počet spojení v síti je $\frac{N(N-1)}{2}$, kde N je počet prvků v síti. Taková síť je nejnákladnější a nejsložitější, na druhou stranu nejrychlejší a nejodolnější proti výpadku. Každý prvek může komunikovat se svým adresátem a při výpadku spoje zprávu poslat jinudy. Pokud jsou z úplné polygonální sítě odebrány některé spoje, je tato síť neúplná.



Obrázek 2.4: Polygonální síť: a) úplná, b) neúplná

Hvězdicová síť

Ve hvězdicové síti je každý prvek spojen s centrální prvkem samostatným spojem, jak je zobrazeno na obrázku 2.5. Jako centrální prvek může sloužit hub nebo přepínač, přes který proudí veškerý provoz a který se stará o komunikaci mezi jednotlivými prvky. Tato síť je jednoduchá na realizaci a má dobrou odolnost při výpadku některého prvku. Při výpadku centrálního prvku ovšem zkolabuje celá síť.



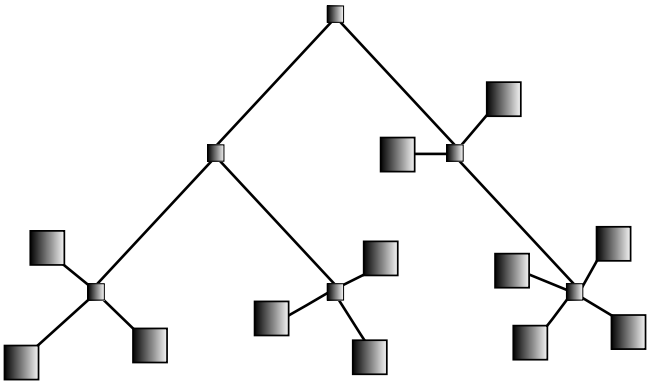
Obrázek 2.5: Hvězdicová síť

Stromová síť

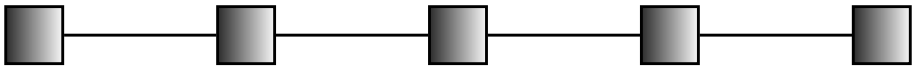
Stromová síť vzniká z hvězdicové sítě spojením několika centrálních řídících prvků. Skládá se tak z několika spojených hvězd, možná struktura je na obrázku 2.6. Tato síť je stále jednoduchá na realizaci s možností udělat síť velmi rozsáhlou. V síti je vždy jen jedna možná cesta od odesílatele k příjemci, nevznikají smyčky. Komunikace probíhá přes nadřazené centrální prvky. Pokud v síti selže nějaký prvek, ostatní v síti stále mohou komunikovat. Ovšem selže-li selže vysoce postavený centrální prvek, nemůže spolu komunikovat již značné množství prvků.

Lineární síť

Lineární síť je nejjednodušší sítí, v níž platí, že každý prvek má maximálně dvě spojení, což je patrné z obrázku 2.7. Komunikace probíhá přes sousední prvky a zpráva může jít jen na jednu nebo druhou stranu. Pokud prvek není platným příjemcem, pouze ji přepošle dál. Při výpadku jednoho prvku se síť rozdělí na právě dvě části. Pro každou část platí, že prvky jsou spolu ve spojení.



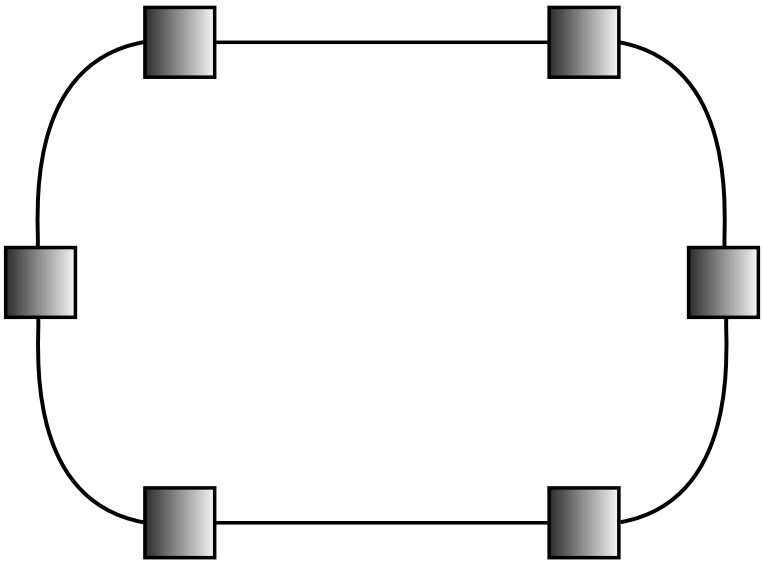
Obrázek 2.6: Stromová síť



Obrázek 2.7: Lineární síť

Kruhová síť

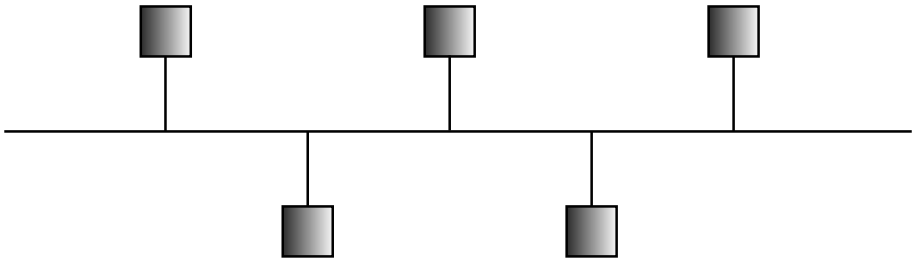
Každý prvek v kruhové síti má právě dva spoje, jak je zobrazeno na obrázku 2.8. Komunikace je však ještě jednodušší než u lineární sítě, jelikož zprávu prostě stačí poslat po kruhu a adresáta si již najde. S rostoucím počtem prvků je přenos zprávy pomalejší, jelikož musí projít přes všechny prvky mezi dvěma komunikujícími. Při výpadku jakéhokoliv prvku dojde ke zhroucení celé sítě. V takovém případě se z kruhové sítě stává síť lineární, s čímž ovšem není počítáno.



Obrázek 2.8: Kruhová síť

Sběrnicová síť

Sběrnicová síť (Bus topology), obrázek 2.9, se snaží o propojení prvků, jako má úplná polygonální síť, tedy spojení každého s každým, ale s minimálními náklady a podstatným zjednodušením. Každý prvek je připojený na jediné společné spojení, sběrnici. Poslanou zprávu po sběrnici slyší všichni, přijme ji však pouze adresát. Při komunikaci je však nutné sběrnici řídit či se jinak starat o kolize, jelikož při vysílání dvou a více prvků naráz vznikne na sběrnici neidentifikovatelný stav, kdy nikdo nic nepošle a nikdo nic nepřijme.



Obrázek 2.9: Sběrnice

Pro přístup na sběrnici existuje několik přístupových metod.

Metody s náhodným přístupem se pro automatizaci příliš nehodí, jelikož je nutné zajistit řízení v reálném čase a tyto metody nezaručují doručení zprávy ve stanoveném čase. Takovým přístupem je např. **CSMA** (Carrier Sense Multiple Access), kdy ve verzi **CSMA/CD** (detekce kolize) vysílací prvek nejdříve sleduje sběrnici, zda je volná a může začít vysílat. Avšak ve stejnou chvíli může zahájit přenos více stanic. Proto stále sledují sběrnici, a pokud zjistí, že nastala kolize, všechny přestanou vysílat a za náhodný časový interval zkusí vysílat znovu.

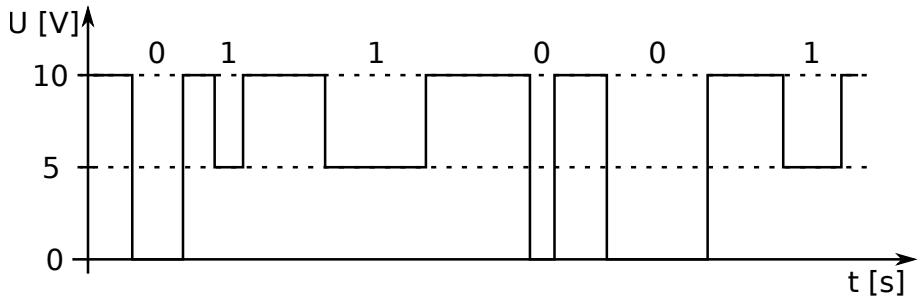
V automatizaci se spíše používají *metody s definovaným přístupem*, kdy je daný vysílací čas, a tedy nehrozí kolize. U **metody bitové mapy** je na sběrnici synchronizátor, který vyšle krátký paket a stanice, které chtějí vysílat, mu odpoví v čase, který mají na odpověď přidělený. Následně vysílají v čase, který jim opět vymezí synchronizátor. U **metody přidělování pověření** je na sběrnici zaveden logický kruh. V kruhu má vždy pouze jedna stanice pověření (token) a ta jediná může vysílat. Po ukončení vysílání, pokud stanice vůbec vysílat chtěla, posílá stanice pověření dál. Pro případ, že by se pověření ztratilo, existuje na sběrnici monitorovací stanice, která po určitém čase zjistí, že v kruhu chybí pověření a vygeneruje nové. **Prioritní metoda** používá statickou a dynamickou prioritu. Statická priorita je hodnota stanici přidělená systematicky a dynamická priorita roste s časem, pokud stanice právě nevysílala. Vysílá ta stanice, která má nejvyšší prioritu

a po vysílání se dynamická priorita vynuluje. V automatizaci nejpoužívanější metodou je **Master/slave**. V síti je stanice master, která má jediná právo vysílat požadavky na zbylé stanice slave. Slave může vysílat pouze tehdy, pokud ho k tomu master vyzve. Většinou má master požadavek na slave, který mu obratem poskytne odpověď. Při více slavů na sběrnici je možno tuto metodu nazvat **Klient/server**, kdy master se jako klient ptá různých serverů, slavů, na dotazy, a dostává odpovědi.

2.5 Přenos a kódování zprávy

Přenos zprávy může probíhat **synchronně**, **asynchronně** a **arytmicky**.

Asynchronní přenos, jak již název napovídá, nemá žádné časování a synchronizaci přenosu. To, že jde o platná data, se pozná podle další, třetí úrovně, která odděluje jednotlivé bity. Přenos jednoho bitu může trvat různou dobu, stejně tak i mezera mezi jednotlivými bity může být různá. Na obrázku 2.10 je možný tvar přenosu, kdy hladina 10 V znamená mezeru mezi bity, hladina 5 V logickou jedničku a 0 V logickou nulu.

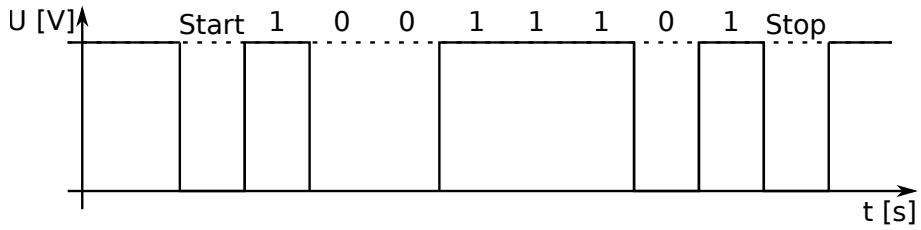


Obrázek 2.10: Asynchronní přenos

Při **arytmickém** přenosu dat také dochází k přenosu v libovolnou dobu, ale zpráva je již synchronizována na svém začátku a konci. Uprostřed zprávy, v platných datech, synchronizace není a počítá se s udržením hodin po dobu přijetí zprávy. S další přijatou zprávou se hodiny opět sesynchronizují na počátečním prvku.

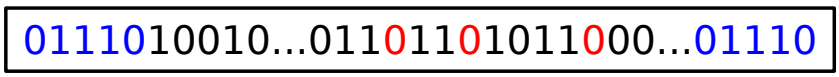
Zprávy mohou být bitově, bytově nebo znakově orientované. Příklad **bytově** orientované zprávy lze vidět na obrázku 2.11, kdy přenos vždy začíná jedním *Start bitem* a končí alespoň jedním, maximálně dvěma *Stop bity*. Bytově orientované zprávy jsou vždy stejně dlouhé.

Možný příklad **bitově** orientovaného přenosu je na obrázku 2.12. Jako tzv. *křídlová značka* je zde zvolena kombinace pěti bitů, s logickými nulami na krajích a třemi jedničkami uprostřed, na obrázku vyznačena modře.



Obrázek 2.11: Bytově orientovaný přenos

Tato kombinace vždy zahajuje a končí přenos. Pokud se ve zprávě objeví posloupnost dvou a více jedniček (obecně o jednu jedničku méně než v křídlové značce), odesílatel vždy přidá za každou druhou jedničku logickou nulu a adresát automaticky po každé druhé jedničce logickou nulu odebere, vyjma křídlové značky. Tyto přidané nuly jsou na obrázku zvýrazněny červeně. Pokud se v přenosu objeví více jedniček za sebou, znamená to chybu přenosu. Velikost dat může být libovolná.



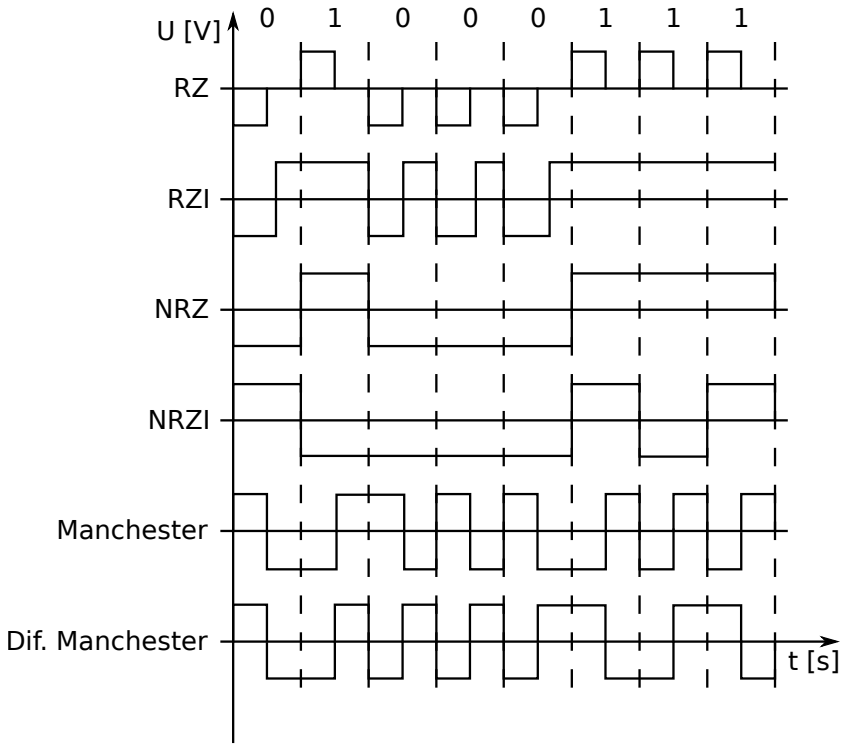
Obrázek 2.12: Bitově orientovaný přenos

Znakově orientovaný přenos je podobný přenosu bitovému, avšak posílají se zde znaky. Na obrázku 2.13 jsou modře vyznačeny řídící znaky a to SYN jako synchronizační značka, STX jako začátek textu, ETX jako konec textu a ESC jako únikový znak. Pokud se v datové části chce poslat jakákoliv řídící značka jako data, odesílatel před ni vloží únikový znak a adresát ví, že ji má přečíst jako data. Řídící znak poslaný jako data je na obrázku vyznačený červeně.

Obrázek 2.13: Znakově orientovaný přenos

Synchronní přenos je jako jediný synchronizovaný pravidelně bez dalších řídících znaků či bitů. Charakteristické okamžiky datového signálu jsou od sebe vždy vzdáleny stejně. Avšak synchronizace se musí stále posílat přenosovým kanálem, nelze se spolehnout ani na sebelepší hodiny. Hodiny se tedy mohou posílat zvlášť svým kanálem, to je ale celkem nákladné a stejně může dojít ke zpoždění hodinového signálu oproti datům, která se pak korektně nepřijmou. Druhou možností, jak přenášet hodinový signál, je tento

signál zakódovat přímo do vysílaných dat. Různé způsoby kódování jsou zobrazeny na obrázku 2.14. Na zakódování hodin do dat se však hodí pouze RZ a oba Menchestery, popsané dále.



Obrázek 2.14: Kódování zprávy

První možností kódování zprávy je **Return to zero** (RZ), návrat k nule, kdy se po každém poslaném bitu kanál nastaví na výchozí hodnotu, např. 0 V. Po každém bitu tedy nastane změna a při každé změně pak dochází k synchronizaci hodin. Avšak stejně jako u asynchronního přenosu potřebuje toto kódování jako jediné tři napěťové stavy a je velmi neefektivní. Další možností je **Return to zero inverted** (RZI), návrat k nule s inverzí, kterému stačí pouze dva stavy. Logická nula je kódována pulsem kratším, než je délka jednoho hodinového signálu a jednička je při přenosu beze změny, tedy ve výchozí pozici. Pokud se bude vysílat sekvence několika jedniček, hrozí rozladění hodin. Kódování **Non return to zero** (NRZ), bez návratu k nule, je přímý přepis logického stavu na přenosový kanál. Logická nula má jednu napěťovou hodnotu a jednička druhou. Při dlouhé sekvenci nul nebo jedniček však může dojít k rozladění hodin. **Non return to zero inverted** (NRZI), bez návratu k nule s inverzí, při logické jedničce změní napěťový stav přenosového kanálu s hodinovým signálem (invertuje ho), při logické nule se stav nemění. Z toho plyne, že pokud bude sekvence logických nul, hrozí roz-

ladění hodin. Kódování **Manchester** synchronizuje hodiny v každém bitu stejně jako RZ, avšak stačí mu na to pouze dva stavy. V polovině hodinového taktu dochází vždy ke změně signálu. Pokud jde shora dolů, kódovala se logická nula, pokud zdola nahoru, kódovala se logická jednička. **Diferenciální Manchester** má stejné synchronizační vlastnosti jako Manchester, ovšem není zde důležitý směr, ale změna. V každém taktu uprostřed opět dochází ke změně signálu. Pokud na začátku došlo ke změně, kódovala se logická nula, pokud změna nebyla, kódovala se logická jednička.

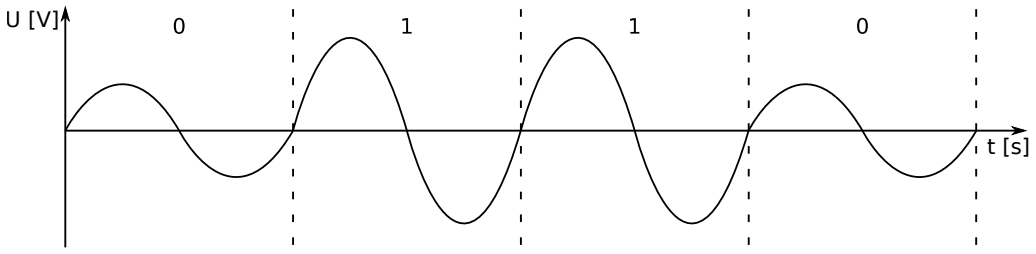
Kódování zprávy se nepoužívá pouze u synchronizovaného přenosu, zprávy lze kódovat při přenosu libovolném. Zpráva arytmického přenosu zobrazena na obrázku 2.11 je kódována binárně na přímo, tedy kódem NRZ.

Přenosy výše popsané jsou označované jako přenosy v základním pásmu. Jsou vodičem přenášeny dvěma (v případě kódování RZ třemi) napěťovými úrovněmi. To se ale nehodí pro všechny typy přenosových kanálů. Pro bezdrátové spojení je takový přenos nepoužitelný. Také náhlé skokové změny napěťové úrovně mohou způsobit problém.

2.6 Modulace zpráv

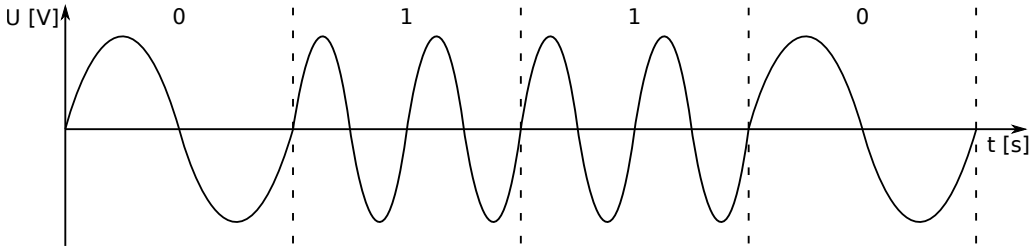
Na základě problémů uvedených na konci předchozí kapitoly je zaveden tzv. přenos v přeloženém pásmu. Užitečný signál je namodulován na signál, který se daným kanálem šíří nejlépe. Většinou je tímto nosným signálem sinusoida. Sinusový harmonický signál má tři základní složky a to amplitudu (A), frekvenci (ω) a fázi (ψ) jak je patrné ze vzorce popisující jeho průběh: $y = A \cdot \sin(\omega \cdot t + \psi)$. Změny těchto hodnot se pak používají v modulování přenosu. Signál je vždy spojitý a nenastává zde skoková změna, vyjma „špatné“ změny u fázové modulace např. o 45°, kdy se průběh sinusovky změní skokově z nulové hodnoty na hodnotu maximální.

Amplitudová modulace používá změnu amplitudy nosného signálu pro vyjádření různých logických hodnot. Na obrázku 2.15 je logická nula vyjádřena poloviční amplitudou než logická jednička.



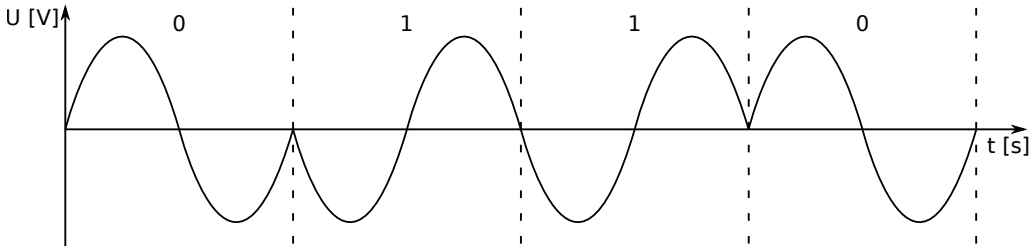
Obrázek 2.15: Amplitudová modulace

Frekvenční modulace používá změnu frekvence nosného signálu pro vyjádření různých logických hodnot. Na obrázku 2.16 je logická nula vyjádřena poloviční frekvencí než logická jednička.



Obrázek 2.16: Frekvenční modulace

Fázová modulace používá změnu fáze nosného signálu pro vyjádření různých logických hodnot. Na obrázku 2.17 má logická jednička obrácenou fázi než logická nula a je tedy oproti nule posunuta o 180°.



Obrázek 2.17: Fázová modulace

Ve všech předchozích příkladech modulací byly použity dvě změny modulovaného signálu. Modulace je tedy dvoustavová a jeden stav přenáší jeden bit. Lze přenášet i více bitů. V takovém případě ovšem musí dojít k navýšení počtu stavů a tento růst je exponenciální. Pro přenos n bitů je potřeba 2^n stavů. Tedy při čtyřech stavech lze přenášet dva bity naráz, při osmi stavech pak tři bity atd. Čím více bitů se přenáší v jednom stavu, tím je přenos náročnější na kvalitu a přesnost přenosových a koncových obvodů. Všechny modulace jdou také zkombinovat dohromady (vzniká např. frekvenčně-amplitudová modulace), čímž se opět zvýší počet přenesených bitů v jednom stavu.

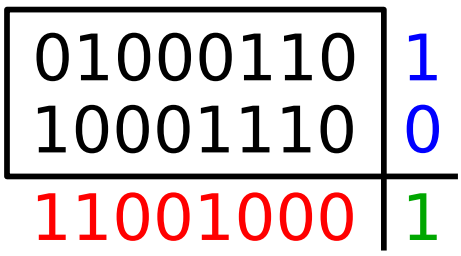
2.7 Zabezpečení dat

Při přenosu dat může dojít k chybě, tj. jeden nebo více bitů dorazí k příjemci s opačnou logickou hodnotou. Data tak mohou být neplatná, v horším případě mohou být přijata jako jiná platná data. Proto se při přenosu

většinou zabezpečují. Zabezpečení rozšiřuje původní data o několik dalších bitů, je redundantní a snižuje efektivitu přenosu. Zabezpečovací kódy dokáží buď chybu pouze detekovat, nebo ji po detekci rovnou opravit. Zabezpečení zprávy není nikdy stoprocentní, s navyšováním počtu zabezpečovacích bitů ale účinnost roste.

Nejjednodušším a nejméně efektivním zabezpečením je přenášený bit prostě zopakovat. **Opakovací** kód opakuje každý bit a **koktavý** kód opakuje skupinu bitů. Opakování lze použít nesčetněkrát. S každým opakováním lze detekovat další chybu, při sudém počtu opakování lze chybu i opravit. Tedy při jednom opakování lze chybu detekovat, při dvou opakováních lze dvě detekovat a jednu opravit, při třech opakováních lze tři detekovat a jednu opravit atd. Oprava zprávy však nemusí být nikdy správná, pokud při přenosu nastane více chyb.

Dalším jednoduchým, ale již mnohem používanějším zabezpečením je **parita**. Parita může být lichá nebo sudá a za přenášenou zprávu přidává paritní bit. Takové paritě se říká příčná parita. Sudá parita kontroluje počet jedniček ve zprávě a pokud je jich sudý počet, paritní bit má hodnotu logické nuly, pokud je počet lichý, paritní bit je logická jednička. U liché parity je to přesně naopak. Parita dokáže odhalit lichý počet chyb, sudý počet chyb odhalit nedokáže, jelikož počet jedniček bude s paritou souhlasit. Pokud se zabezpečuje skupina zpráv, kontrolují se stejnolehlé bity ve všech zprávách a této paritě se říká podélná. Kombinaci těchto parit lze vidět na obrázku 2.18. Parita je sudá, příčné paritní bity jsou vyznačeny modře, podélné paritní bity červeně a zelený bit je dopočítaná parita parit. Takováto parita dokáže již jednu chybu opravit.



Obrázek 2.18: Příčná a podélná parita

Kontrolní součet (checksum) sčítá hodnoty jednotlivých přenášených znaků v aritmetice modulo. Znaky jsou brány jako celá dvojková čísla bez znaménka. Zpráva libovolné délky se rozdělí na n-tice a každá n-tice nul a jedniček je převedena na číslo v desítkové soustavě. Označíme-li součet těchto čísel (v desítkové soustavě) jako S, pak výsledná hodnota zabezpečení je $S\ mod(2^n)$. Tedy například pro $n = 8$ je zpráva rozdělena po bytech

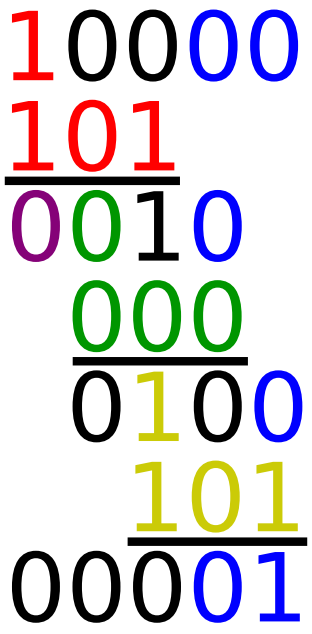
a výsledné zabezpečení je zbytek po dělení součtu S číslem 2^8.

Nejúčinnější formou zabezpečení je **cyklický redundantní součet CRC (Cyclic Redundancy Check)**. Stojí na silných matematických základech a je velmi spolehlivý, avšak za cenu, že chyby pouze detekuje. Na druhou stranu výpočet a HW implementace (XOR hradla a posuvný registr) jsou vcelku jednoduché. CRC je hašovací funkce, takže pro libovolně dlouho posílanou zprávu se délka zabezpečení pro dané CRC nemění. Na základě jednotlivých bitů ve zprávě se provede operace XOR s průběžným výsledkem a generujícím polynomem pro dané CRC. Po odeslání dat se zabezpečením, příjemce také vypočte CRC z přijatých dat a porovná ho se zaslaným zabezpečením. Pokud se schodují, zpráva s nejvyšší pravděpodobností přišla v pořádku.

Na obrázku 2.19 si ukážeme výpočet zabezpečení. Generující, nebo též charakteristický polynom byl zvolen $x^2 + 1$, binárně 101. Vypočítaný součet je tedy CRC-2 podle nejvyššího stupně polynomu, zabezpečení je dvou bitové. Chceme zabezpečit zprávu 100. V první řádce jsou modře vyznačené nuly pro další výpočet a při odeslání zprávy jsou nahrazeny vypočítaným zabezpečením. Základní princip spočívá v tom, že pokud je ve zprávě (nebo posléze v průběžných výsledcích) na pozici nejvyššího bitu jednička, provede se operace XOR se zabezpečovacím polynomem a následně posuv o jeden bit doprava, pokud se objeví nula, provede se pouze posuv. Při posuvu by na nejvyšším místě neměla být nikdy jednička, vždy by měla přetékat nula. Zprávu máme zarovnanou na jedničku, kterou vezmeme jako první (na obrázku vyznačena červeně). Provedeme operaci XOR a vznikne průběžný výsledek 001, který posuneme o jeden bit doprava. Fialová nula nám tedy přetéká a modrou nulu si vezmeme z předpřipravené pozice. Jelikož je v současném výsledku na nejvyšší pozici nula, označená zeleně, provedeme pouze posuv. Na obrázku je znázorněna operace XOR s nulami místo generujícího polynomu, což je ve výsledku stejné číslo. Po posuvu máme hodnotu 100, na nejvyšším místě je jednička, označená žlutě, provedeme tedy opět operaci XOR s generujícím polynomem. Na poslední řádce můžeme vidět, že jsme obsadili všechny předpřipravené modré pozice a všechny přetečené hodnoty jsou nulové. Výpočet tedy ukončíme. Zabezpečení vstupních dat je 01.

Obvykle se používají polynomy s mnohem větším stupněm, např. CRC-16 nebo CRC-32.

V následující kapitole se podrobně teoreticky seznámíme s prvním ze dvou protokolů, s nimiž pak budeme následně pracovat praktické části.



Obrázek 2.19: Cyklický redundantní součet

2.8 Modbus

Modbus je sběrnice typu Device bus vyvinutá společností Modicon v roce 1979. V průmyslovém výrobních prostředí je Modbus, s více jak 7 miliony použitými zařízeními zejména v Severní Americe a Evropě, velmi používanou sběrnicí. Je to otevřený a jednoduchý komunikační protokol. Z modelu ISO/OSI (viz. kapitola 2.3) používá tři vrstvy - fyzickou, linkovou a aplikační. Existuje i verze Modbus TCP pro síť Ethernet, ta pak z tohoto modelu používá více vrstev nutných pro přenos daný protokolem TCP/IP.

Nyní se budeme věnovat popisu jednotlivých vrstev modelu ISO/OSI pro sběrnici Modbus.

2.8.1 Fyzická vrstva

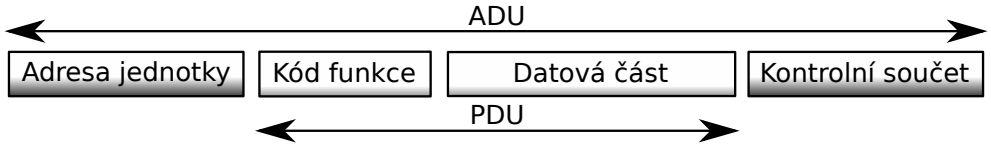
K přenosu dat je možné použít celou řadu komunikačních médií jako sériové linky typu RS-232, RS-422 a RS-485, optické a rádiové sítě, nebo již zmíněnou síť Ethernet.

Standardně je využíván sériový port **RS-232C**. Tato verze z roku 1969 je verzí poslední a v oblasti osobních počítačů se již nepoužívá. V automatizaci je však toto rozhraní stále velmi oblíbené. Jedná se o dvoubodové datové spojení, kdy jeden datový vodič je pro příjem (RxD), druhý pro vysílání (TxD) a společnou zem (GND). Data jsou přenášena opačnými napěťovými úrovněmi, obvykle v rozmezí 5 V až 15 V. Nejčastěji je používáno napětí 12

V, kde napětím -12 V je přenášena logická jednička a napětím 12 V logická nula. Přenos dat je arytmický a bytově orientovaný. Přenos jednoho bytu je na obrázku 2.11. Každá jednotka musí podporovat paritu, ta je pak vložena mezi datové bity a Stop bit. Pokud parita není využita, používají se dva Stop bity.

2.8.2 Linková vrstva

Obecný formát zprávy je na obrázku 2.20. Základní část zprávy - PDU (Protokol Data Unit) se posílá vždy a je nezávislá na komunikační vrstvě. K základní části se v aplikační úrovni přidá adresa a zabezpečení, a vznikne celá Modbus zpráva - ADU (Aplication Data Unit).



Obrázek 2.20: Základní tvar Modbus zprávy [7]

Na sériové lince se používají dva vysílací režimy, které určují formát přenášených dat. První režim, kterým je **RTU** (Remote Terminal Unit), musí podporovat každá jednotka. V tomto režimu každý byte obsahuje dva 4-bitové hexadecimální znaky. Velikost jednotlivých částí je na obrázku 2.21, kde N je počet přenesených bytů. Začátek a konec zprávy je rozpoznán pomlkou na sběrnici delší než 3,5 znaku. Vysílání zprávy musí být souvislé, mezi jednotlivými byty nesmí být mezera delší než 1,5 znaku. Maximální délka zprávy může být 256 bytů podle implementace na sériové lince, datová část může mít tedy maximálně 252 bytů. Druhý režim je **ASCII**, který není povinný. Jak název napovídá, zde se posílají ASCII znaky. Jedním bytem lze poslat maximálně jeden číselný znak. Režim ASCII je tedy vždy minimálně dvakrát tak datově náročný než RTU režim. Na druhou stranu, začátek zprávy indikuje znak „:“ a konec zprávy dvojice znaků „CR, LF“ (Carriage Return, Line Feed), to dovoluje posílat znaky až s vteřinovou mezerou. [6, 7]

Začátek	Adresa	Funkce	Data	CRC	Konec
> 3,5 znaku	8 bitů	8 bitů	N * 8 bitů	16 bitů	> 3,5 znaku

Obrázek 2.21: Rámec RTU zprávy [7]

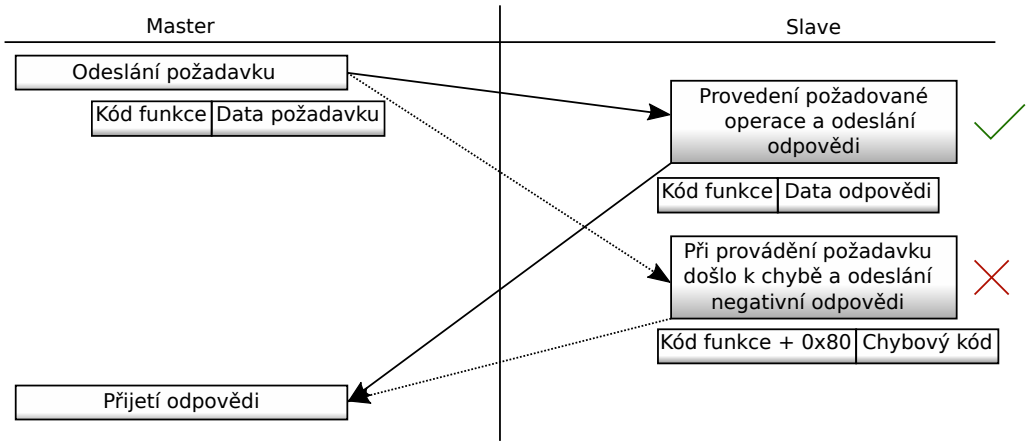
Pro přístup ke sběrnici je použita metoda Master/slave (popsaná v kapitole 2.4). Jak ukazuje obrázek 2.21, jednotky se adresují 8 bity. Ke sběrnici

může být připojen maximálně jeden master a až 247 slave jednotek, jejichž adresa je jedinečná. Adresy od 248 do 255 jsou rezervované. Pokud master odešle zprávu s adresou 0, zpráva je poslána v *Broadcast* režimu všem jednotkám na sběrnici a žádná z nich neodpoví. Master nemá žádnou specifickou adresu. [7]

V režimu RTU jsou odesílané zprávy zabezpečené CRC-16. Generující polynom má tvar $x^{16} + x^{15} + x^{13} + 1$ a zabezpečení se vypočítá ze všech dat ve zprávě předcházejících, tedy z adresy, kódu funkce a dat. [6]

2.8.3 Aplikační vrstva

Protokol Modbus definuje tři základní typy zpráv: *požadavek*, *odpověď* a *záporná odpověď*. Výměna zpráv je zobrazena na obrázku 2.22. Kód funkce udává jednotce slave, jakou operaci má provést. Datová část, poslaná masterem, slouží k uskutečnění operace, určené kódem funkce. Obsah může být například adresa a počet vstupů, které má slave číst a odeslat, nebo hodnota registrů, které má zapsat. Pro provedení některých funkcí nejsou potřeba žádná data a v tom případě nemusí datová část ve zprávě vůbec být. Funkce je dána 8 bity, může nabývat hodnot 1 až 255. Polovina tohoto rozsahu je věnována chybné odpovědi. Pokud slave přijme bezchybně požadavek, ale není schopen ho normálně zpracovat, přičte ke kódu funkce hodnotu 128, hexadecimálně 0x80, což je nejvýznamnější bit v kódu funkce, a společně s chybovým stavem odešle odpověď jako zápornou. Nejčastější chybové stavy jsou znázorněny v tabulce 2.1. Při situaci, kdy nastane komunikační chyba nebo zpráva dojde, ale má neplatné zabezpečení (parita, CRC), se záporná odpověď neposílá.



Obrázek 2.22: Modbus transakce [7]

Tabulka 2.1: Chybové kódy [7]

Kód	**Jméno**	**Význam**
0x01	Ilegální funkce	Požadovaná funkce není slavem podporována
0x02	Ilegální adresa dat	Zadaná adresa je mimo podporovaný rozsah
0x03	Ilegální hodnota dat	Předávaná data jsou neplatná
0x04	Selhání zařízení	Při provádění požadavku došlo k neodstranitelné chybě

Platné funkce mají kódy 1 až 127 a jsou rozděleny do tří skupin. První a hlavní skupinou jsou **veřejné** funkce. Tyto funkce jsou jasně definovany, schváleny společností MODBUS.org, je garantována jejich unikátnost a jsou veřejně zdokumentovany např. v [6]. Definice nejvýznamnějších funkčních kódů je v tabulce 2.2. Další skupinou jsou funkce **uživatelské**. Pro tyto funkce jsou vymezeny dva rozsahy: 65 až 72 a 100 až 110. Všude jinde jsou funkce veřejné. Jak název napovídá, tyto funkce umožňují uživateli implementovat funkce, které nejsou definované ve specifikaci a není tak zaručena jejich unikátnost. Po projednání se schvalovací organizací je lze zařadit do funkcí veřejných. Do poslední skupiny se řadí **rezervované** funkce, které jsou používány některými firmami, tyto funkce nejsou veřejně dostupné.

Tabulka 2.2: Definice funkčních kódů [7]

Popis funkcí				Kódy funkcí
Přístup k datům	**Bitový přístup**	Fyzické diskrtétní vstupy	Čti diskrétní vstupy	02; 0x02
		Interní bity nebo fyzické cívky	Čti stavové registry	01; 0x01
			Zapiš stavový registr	05; 0x05
			Zapiš více stavových registrů	15; 0x0F
	16 - bitový přístup	Fyzické vstupní registry	Čti vstupní registr	04; 0x04
		Interní registry nebo fyzické výstupní registry	Čti uchovávací registry	03; 0x03
			Zapiš jeden registr	06; 0x06
			Zapiš více registrů	16; 0x10
			Čti/zapiš více registrů	23; 0x17
			Zapiš registr s maskováním	22; 0x16
			Čti FIFO frontu	24; 0x18
	Přístup k záznamům v souborech		Čti záznam ze souboru	20; 0x14
			Zapiš záznam do souboru	21; 0x15
Diagnostika			Čti stav	07; 0x07
			Diagnostika	08; 0x08
			Čti čítač komun. událostí	11; 0x0B
			Čti záznam komun. událostí	12; 0x0C
			Sděl identifikaci	17; 0x11
			Čti identifikaci zařízení	43; 0x2B

V rámci aplikační vrstvy se nyní podíváme na datový model Modbusu.

Datový model

Datový model Modbusu je založen na sadě tabulek s charakteristickým významem. Adresní prostor se adresuje 16 bity s maximální hodnotou 65536. Mapování tabulek do adresního prostoru je závislé na konkrétním zařízením. Tabulky se mohou částečně nebo úplně překrývat, nebo může být jedna tabulka namapovaná do celého adresního prostoru. Obvykle se však z důvodu kompatibility rozděluje adresní prostor na čtyři stejně velké části o velikosti 10000 položek, přesně jak popisuje tabulka 2.3. Přístupná je každá položka jednotlivě nebo po skupině, skupina je omezena maximální délkou datové části zprávy (253 bytů). K jednotlivým datovým sekcím se přistupuje přes dané funkce, typicky je to pro stavové registry funkce 01, pro diskrétní vstupy funkce 02, pro vstupní registry funkce 04 a pro uchovávací registry funkce 03. [7]

Tabulka 2.3: Datový model [7]

Tabulka	Typ položky	Přístup	Popis	Adresa (MODICON)
Stavové registry *(Coils)*	1-bit	Čtení/zápis	Data modifikovaná aplikačním programem	0 až 9999
Diskrétní vstupy *(Dicrete Inputs)*	1-bit	Čtení	Data poskytovaná I/O systémem	10000 až 19999
Vstupní registry *(Input Registers)*	16-bitové slovo	Čtení	Data poskytovaná I/O systémem	30000 až 39999
Uchovávací registry *(Holding Registers)*	16-bitové slovo	Čtení/zápis	Data modifikovaná aplikačním programem	40000 až 49999

Dále si v rámci aplikační vrstvy podrobně rozebereme funkci použitou v realizovaném mostu.

Funkce „čti uchovávací registry“

Na funkci „čti uchovávací registry“ uvedené v tabulce 2.2 si ukážeme jednotlivé rozložení bitů a obsah posílané zprávy. Funkce slouží ke čtení obsahu souvislého bloku a dokáže přečíst až 125 uchovávacích registrů. V požadavku, který je vypsán v tabulc 2.4, je specifikována adresa prvního registru a počet registrů, které se mají číst. V odpovědi je počet přenášených bytů a hodnoty registrů, popsané v tabulce 2.5, kde N je počet registrů. Registr je 16 bitový, proto se přenáší 2 byty. Pokud nastane chyba, ke kódu funkce se přičte hodnota 0x80 a s daným chybovým stavem se odešle zpět, jak ukazuje tabulka 2.6. [7]

Tabulka 2.4: Požadavek mastera [7]

Kód funkce	1 byte	0x03
Počáteční adresa	2 byty	0x0000 až 0xFFFF
Počet registrů	2 byty	0x01 až 0x7D (125)

Tabulka 2.5: Odpověď stanice slave [7]

Kód funkce	1 byte	0x03
Počet bytů	2 byty	2 * N
Hodnoty registrů	2 * N bytů	Data

Tabulka 2.6: Odpověď s chybou [7]

Kód funkce	1 byte	0x83
Chybový kód	1 byty	0x01, 0x02, 0x03 nebo 0x04

V následující kapitole se podrobně teoreticky seznámíme s druhým protokolem.

2.9 HART

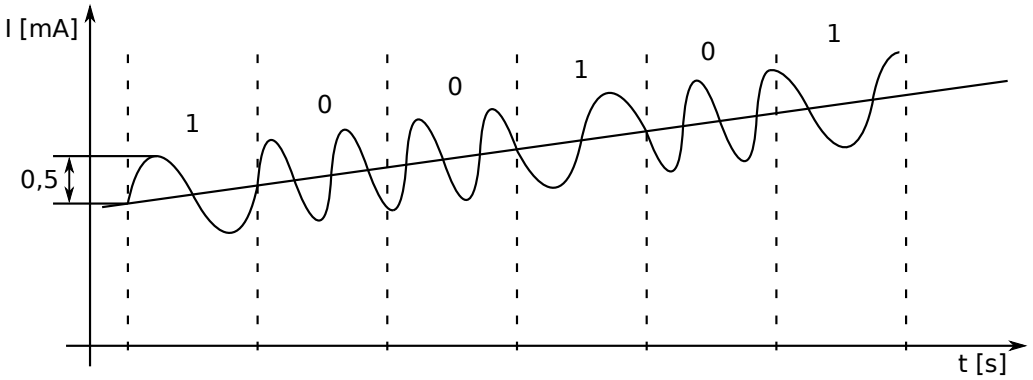
Protokol HART (*Highway Addressable Remote Transducer*) je, stejně jako Modbus, sběrnicí typu Device bus. Byla vyvinuta společností Rosemont Inc. v roce 1986. Od prvního vydání se protokol stále vylepšuje. Od prvního vydání se protokol stále vylepšuje, ovšem všechny nové revize jsou zpětně kompatibilní. Současné vydání je revize 7 z roku 2007. S více jak 30 miliony podporovaných nainstalovaných zařízení po celém světě, jde o jednu z nejrozšířenějších průmyslových sběrnic. Protokol je vhodný zejména pro připojení inteligentních analogových čidel. Přenos dat z čidla do nadřazeného systému probíhá v digitální podobě. Kromě zjištění aktuální hodnoty lze čidlu měnit parametry, kalibrovat jej či zjišťova diagnostické informace. Z modelu ISO/OSI používá tři vrstvy. Stejně jako Modbus (verze TCP), i HART přišel v poslední revizi s bezdrátovou verzí, WirelessHART. Ta však přišla o hlavní výhodu původního HARTu, a to komunikaci po tradiční proudové smyčce 4-20 mA, se kterou je HART kompatibilní. [12]

Nyní se budeme věnovat popisu jednotlivých vrstev HARTu.

2.9.1 Fyzická vrstva

Jak bylo zmíněno, HART pro přenos dat používá standardní proudovou smyčku 4-20 mA. Zavedení do stávající infrastruktury je velmi výhodné, protože není potřeba počítat s dalšími náklady na rozvod kabelů. HART je s proudovou smyčkou kompatibilní, což znamená, že lze po proudové smyčce posílat jak analogové hodnoty, tak i digitální data. Toho je docíleno namodulováním digitálních dat na data analogová modulací FSK (*Frequency Shift Keying*) podle standardu Bell 202. V podstatě se jedná o spojení frekvenční

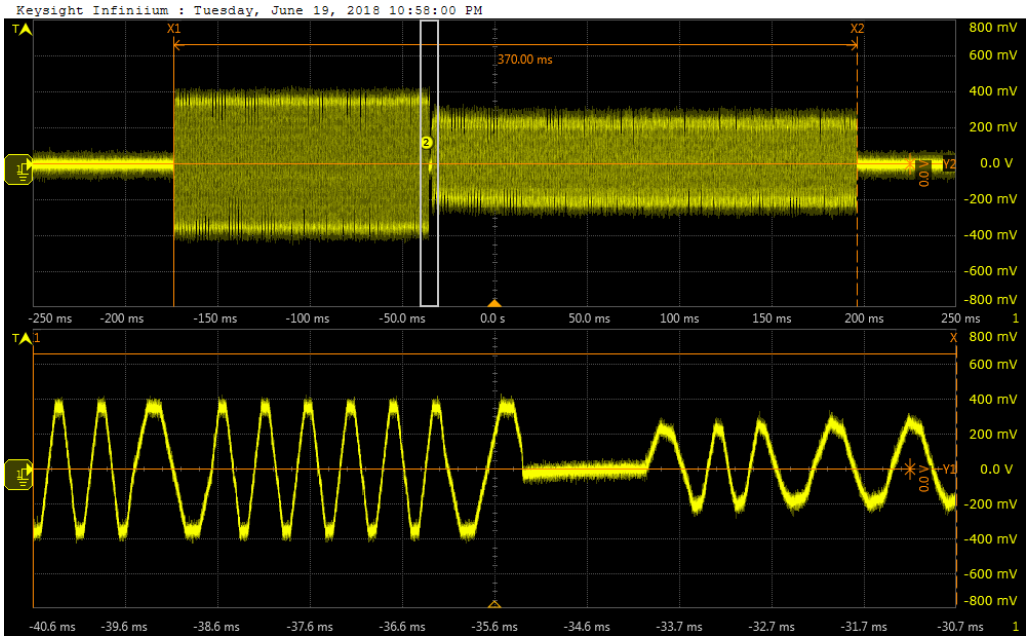
a fázové modulace, popsané v kapitole 2.6, kde hlavní roli hraje modulace frekvenční - logická jednička je namodulována frekvencí 1200 Hz a logická nula frekvencí 2200 Hz, jak je vidět na obrázku 2.23. Amplituda signálu je 0,5 mA. Fázová modulace je odvozena od pevného intervalu jednoho bitu pro logickou jedničku - délka jednoho intervalu je 1/1200. Pro logickou jedničku se tedy na interval vejde přesně jedna perioda sinusovky. Pro logickou nulu je, jak víme, frekvence vyšší (2200 Hz), ovšem je to méně než dvojnásobek hodnoty 1200. To způsobí, že na jeden interval se vejdou necelé dvě periody sinusového signálu. Protože signál je spojitý, musí nutně dojít k fázovému posunu sinusoidy pro každou další logickou hodnotu. Při jiných kombinacích nul a jedniček je možné provést fázový posun o 90°, 180° nebo 270°. Takto namodulovaný signál má nulovou střední hodnotu a nedochází ke zkreslení či poškození původního analogového signálu přenášeného proudovou smyčkou. Možný přenos dat je vidět na obrázku 2.23. Analogová hodnota může být i ze staršího zařízení nepodporující HART. Dosažená komunikační rychlost je 1200 Bd (Baud - modulační rychlost udávající počet změn stavu za jednu vteřinu). Každou vteřinu lze standardně provést dvě výměny zpráv, což je poměrně pomalé, HART tedy není příliš vhodný pro rychlé procesy. Jelikož jde o proudovou smyčku, hodí se do výbušného prostředí. [4, 12]



Obrázek 2.23: Digitální a analogová data na proudové smyčce

Na obrázku 2.24 je vidět reálná komunikace na sběrnici, kdy master poslal požadavek na čtení hodnoty a slave mu po krátké pauze odpověděl. Celá výměna v tomto případě trvala přibližně 370 ms. V dolní části obrázku je pak přiblížení signálu a je vidět změny frekvencí na základě modulace. Hlavní nosnou informací je frekvence, která je ze signálu jasně rozpoznatelná a nevadí, že signál není ideálně sinusový.

Stejně jako u Modbusu se jedná o arytmický a bytově orientovaný přenos. Každý poslaný byte je zabezpečen lichou paritou a ukončen jedním Stop bitem.



Obrázek 2.24: Reálný průběh signálu na sběrnici

Jako kabeláž je doporučený stíněný kroucený pár o maximální délce 1500 m a časové konstantně RC (násobek kapacity a impedance vodiče) menší nebo rovno 65 μs. Při použití více zařízení na jednom páru vodičů klesá jeho použitelná délka. Všechna připojená zařízení mají předepsané rozmezí možné impedance a doporučenou hodnotu kapacity. [12]

2.9.2 Linková vrstva

Pro přístup na sběrnici je opět použita metoda Master/slave. Maximální počet slave zařízení na sběrnici je 16. Pokud je ale na sběrnici více než jedno zařízení slave, nelze použít analogovou komunikaci po proudové smyčce. Netypicky lze však na sběrnici zapojit dvě zařízení master. Důvodem je usnadnění konfigurace sítě, kdy primární master je pevný řídící člen sítě sbírající naměřené hodnoty a sekundární master ruční konfigurátor.

V předchozí kapitole jsme si uvedli, že standardně lze poslat maximálně dvě zprávy. To platí pro stav, kdy master vznese požadavek a slave mu odpoví. Existuje však **burst** režim, při kterém master zařízení slave pověří odesíláním odpovědi bez požadavku. V tomto režimu lze přenést až tři zprávy za jednu vteřinu. Tento režim není povinný, má však vyhrazené místo v posílané zprávě.

Obecný formát zprávy je na obrázku 2.25. Všechny pojmy uvedené v obrázku budou vysvětleny a popsány níže.

Obrázek 2.25: Struktura zprávy protokolu HART [12]

Úvodní sekvence slouží přijímací straně k synchronizaci s vysílaným signálem. Každý byte obsahuje hodnotu 0xFF. Synchronizace se tedy skládá ze samých jedniček. Minimální počet poslaných synchronizačních bytů je 5 a maximální 20.

Startovací znak je velikosti jednoho bytu a rozlišuje typ posílané zprávy. Ta může být různě dlouhá a startovací znak tak ukazuje na položku počet bytů, jelikož ostatní položky jsou fixní. Obsah startovacího znaku udává tabulka 2.7.

Tabulka 2.7: Obsah startovacího znaku[12]

	Krátký formát rámce	**Dlouhý formát rámce**
Master → slave	0x02	0x82
Slave → master	0x06	0x86
Burst mode slave → master	0x01	0x81

Od verze 5 rozeznává HART dva druhy **adres**, dlouhou (novější) a krátkou (starší). Adresa je zvolena velikostí rámce. Pokud je zvolen krátký rámec, má adresa velikost 1 bytu. Čtyři nejméně významné byty určují krátkou

tzv. polling adresu. Při zvolení dlouhého rámce má adresa velikost 5 bytů a 38 nejméně významných bitů tvoří adresu. Tři nejméně významné byty obsahují identifikační číslo zařízení, další byte obsahuje kód typu zařízení a šest nejméně významných bitů z nejvýznamnějšího bytu nese identifikační číslo výrobce. U obou formátů rámce platí, že nejvýznamnější bit adresy určuje mastera, kdy logickou jedničkou je označen primární master, a další bit značí, zda je zařízení v burst režimu, pokud ano, pak je použita opět logická jednička.

Počet bytů udává počet přenášených datových bytů a je velikosti 1 byte. Maximální možný počet přenášených dat je 255 bytů. Tento byte také ukazuje na adresu kontrolního součtu.

Status je zahrnut pouze ve zprávách, které odesílá slave a je velký 2 byty. První byte obsahuje informace o chybách při komunikaci a o zpracování přijatého příkazu. Druhý byte informuje o stavu zařízení a výsledku poslední operace. Oba byty jsou v případě bezchybného stavu nulové.

Největší možné množství **dat** je tedy pro slave 253 bytů a pro mastera 255 bytů. Formát dat ve zprávě je součástí specifikace jednotlivých příkazů. Dovolené typy jsou celá čísla bez znamének (8, 16 nebo 24 bitů), čísla s plovoucí řádovou čárkou podle IEEE 754 s jednoduchou přesností, speciální ASCII znaky určené 6 bity a jednotlivé bity určující stav zařízení.

Kontrolní součet je vypočten jako XOR všech předchozích bytů počínaje startovacím znakem. Takto vypočtené zabezpečení je vlastně podélná parita a spolu s příčnou paritou posílanou za každým bytem tvoří již kvalitní zabezpečení. [3, 12]

Nejmenší možná délka zprávy je 10 bytů odesílaná masterem s minimální délkou úvodní sekvence, krátkým rámcem a bez dat. Naopak nejdelší možná délka zprávy je 284 bytů při použití maximální délky úvodní sekvence, dlouhého rámce a obsazení všech dat.

2.9.3 Aplikační vrstva

Zařízení HART obdobně jako Modbusu odešle *požadavek* a zpět dostane *odpověď* nebo *odpověď s chybou*. Chybu však nelze určit z hodnoty přijaté funkce, ale z přijatého pole status. Pro funkci je vyhrazen jeden byt může být tak implementováno až 255 funkcí.

Obdobně i funkce HARTu jsou rozděleny do tří skupin. **Univerzální** příkazy (*Universal Commands*) jsou povinné pro všechny zařízení a v současné revizi jich je dvacet. Výpis všech v tabulce 2.8 a podrobně v [5]. Další příkazy jsou **obvykle používané** (*Common-Practice Commands*) a **specifické** (*Device Specific Commands*), jsou již nepovinné a mj. umožňují ka-

libraci (nula a zesílení), nastavení fyzikální jednotky, zápis sériového čísla, volbu čtyř měřených fyzikálních veličin, reset čidla atd.

Tabulka 2.8: Univerzální funkce [5]

Kódy funkcí	Názvy funkcí
00; 0x00	Čtení unikátního identifikátoru
01; 0x01	Čtení primární proměnné
02; 0x02	Hodnota proudové smyčky a procenta rozsahu
03; 0x03	Čtení dynamických proměnných proměnných a procent rozsahu
06; 0x06	Zápis krátké adresy
07; 0x07	Čtení konfigurace proudové smyčky
08; 0x08	Čtení klasifikace dynamických proměnných
09; 0x09	Čtení proměnných se statusem
11; 0x0B	Čtení unikátního identifikátoru s tagem
12; 0x0C	Čtení zprávy
13; 0x0D	Čtení tagu, popisku a data
14; 0x0E	Čtení informací o převodníku
15; 0x0F	Čtení informací o zařízení
16; 0x10	Čtení konečného výrobního čísla
17; 0x11	Zápis zprávy
18; 0x12	Zápis tagu, popisku a data
19; 0x13	Zápis konečného výrobního čísla
20; 0x14	Čtení dlouhého tagu
21; 0x15	Čtení unikátního identifikátoru spojeného s dlouhým tagem
22; 0x16	Zápis dlouhého tagu

V další kapitole se zaměříme na praktickou část bakalářské práce.

3 Programová realizace mostu

V této kapitole bude popsáno programové vybavení mostu Modbus - HART, které je výsledkem praktické části této práce. Most byl naprogramován v jazyce C ve vývojovém prostředí Dev-C++. K simulaci strany Modbus byl použit program Modbus Slave od firmy Witte Software a k odposlouchávání sériových portů program Device Monitoring Studio od společnosti HHD Software Ltd. Všechen použitý software je legální (placené verze programů jsou volně dostupné ve zkušební verzi).

3.1 Specifikace požadavků

Na trhu existuje několik hotových mostů Modbus - HART (např. od firem Fint, Papouch). Ty jsou však pro uživatele tzv. black boxy, černými krabičkami, do kterých není vidět. Takové mosty jsou implementovány na jeden daný procesor a jsou zcela uzavřené. Jejich zdrojový kód není znám.

Hlavním požadavkem na programové vybavení je jeho snadná přenositelnost a rozšiřitelnost. Most by měl umět základní funkce pro přenos naměřených údajů z měřící jednotky pracující na sběrnici Modbus k řídící stanici na sběrnici HART.

Podle specifikace HARTu [2] je takovýto slave zařazen do první třídy z šesti možných, kdy nejmenší třída je 0. Každá třída obstarává složitější funkce a náročnější komunikaci s řídící jednotkou než třída předchozí. Třída 1 je určena k cyklickému posílání procesních dat. Každá taková jednotka by měla implementovat univerzální funkce 1, 2, 3 a 9 (viz. tabulka 2.8).

3.2 Popis implementace

Pro snadnou přenositelnost naprogramovaného mostu byla vybrána platforma PC a jazyk C jako programovací jazyk. Počítače jsou velmi rozšířené v osobní sféře. Jako průmyslové verze však značně pronikají do automatizace. Jazyk C je stále jeden z nejrozšířenějších jazyků a překladač tohoto jazyka existuje na nespočet procesorů.

Ke správné funkčnosti mostu je potřeba mít dva volné sériové porty. Jeden pro Modbus master a druhý pro HART slave. Programové vybavení je vyvíjeno pro operační systém Windows. Posledním nutným požadavkem je implementace vláken. Jelikož na sběrnici HART náš most dělá stanici

slave, musí neustále na portu poslouchat, zda nedostane požadavek. Pokud by most byl řešen jednovláknově, mohla by nastat situace, kdy by vlákno obstarávalo komunikaci na straně Modbusu a zpozdilo by se čekajíc na odpověď. V takovém případě by most nemusel vůbec postřehnout požadavek od mastera na HARTu a korektně mu odpovědět. K našemu účelu je použita knihovna `pthread.h`.

3.3 Programové vybavení

Volně dostupný zdrojový kód na [1] se snaží o implementaci Modbus mastera. Z takto implementovaného kódu jsme se inspirovali prací se sériovým portem.

Po spuštění mostu je potřeba nastavit sériové porty pro jednotlivé protokoly. Informace o sériových portech jsou uloženy v textovém souboru (podrobněji v kapitole 3.5), který si pro tento účel načteme a porty s těmito daty nastavíme pro komunikaci. Po nastavení portů se program rozdělí na dvě vlákna, kdy jedno obsluhuje protokol Modbus a druhé HART.

Na Modbusu most dělá mastera. Použijeme funkci „čti uchovávací registry“ (viz. kapitola 2.8.3) pro zjištění uložených hodnot. Podle informací z nastavovacího textového souboru se čte od jedné do čtyř proměnných z daného rozsahu adres. Tento rozsah adres se nejdříve ověří čtením dat z krajních mezí zadaného intervalu. Pokud zadané meze nejsou k dispozici, požadavek na zadaný interval se neustále opakuje. Pokud dostaneme kladnou odpověď, víme, že můžeme bezchybně číst proměnné ze zadaných adres uvnitř intervalu. Tyto proměnné přečteme a uložíme do sdílené proměnné, ke které přistupujeme přes mutex. Při odesílání požadavku složíme zprávu z příslušných dat a na konec přiložíme zabezpečení, které si dopočítáme. Po přijetí odpovědi si zabezpečení překontrolujeme. Pokud je v pořádku, přijatá data můžeme uložit.

Na straně HARTu most dělá jednotku slave. V samostatném vláknu tedy most stále čeká, než po lince přijde jakýkoliv požadavek. Pokud zachytí úvodní sekvenci, začne číst zbylá data požadavku. Pokud při čtení dojde k chybě přenosu, všechna přijatá data se zahodí a začne znovu poslouchat na portu. Po přijetí celého požadavku se nejdříve vypočte zabezpečení. Pokud není správné, most odešle masterovi zprávu s nastavením příslušného bitu ve statusu s informací o chybě při přenosu. Při přijetí platného požadavku se ověří, zda je zpráva adresována nám. Když ne, na zprávu se neodpovídá. V případě požadavku na naší adresu se provede příslušná funkce. Složí se vhodná data, dopočte se zabezpečení a odpověď se odešle po portu zpět.

3.3.1 Použité soubory

Soubor `main.c` je soubor spouštěcí. Tento soubor čte inicializační textový soubor a data uloží do příslušných struktur. Po načtení textových souborů dojde k inicializaci sériových portů podle požadavků uložených v souborech. Následně se spustí obě vlákna a běh programů je rozdělen dle protokolů.

O komunikaci na sběrnici Modbus se stará soubor `modbusProtocol.c`, v hlavičkovém souboru `modbusProtocol.h` jsou pak deklarace jednotlivých modulů. Tento soubor dává dohromady jednotlivé byty požadavku a celou zprávu předá modulu `rs232_msgSend` v souboru `rs232API.c` pro odeslání po otevřené sériové lince. Podle počtu proměnných uložených v inicializačním souboru se (s předchozí kontrolou přístupné datové sekce Modbusu) ptá jednotlivě na všechny proměnné. Po každém požadavku čeká na odpověď využívaje modulu `rs232_msgRecieve` stejného souboru. Přijatá data pak neustále ukládá do sdílené paměti a tím je pro HART aktualizuje. Soubor implementuje pouze funkci „čti uchovávací registry“, jelikož HART se v základní verzi ptá jen na více bytové proměnné.

O komunikaci na sběrnici HART se stará soubor `hartProtocol.c`, v hlavičkovém souboru `hartProtocol.h` jsou pak deklarace jednotlivých modulů. Ta funguje opačně než třída Modbus. Využívá stejných modulů souboru `rs232API.c` pro čtení a zápis dat po sériové lince. Neustále čeká na lince, dokud nedojde nějaký požadavek. Příchozí požadavek následně ověří a pokud ho lze zpracovat, sestaví se odpověď po bytech a po sériové lince se odešle. Třída implementuje několik funkcí HARTu. Funkce 1, 2, 3 a 9 (viz. tabulka 2.8) jsou povinné pro splnění třídy 1. Nejdůležitější je funkce 1 požadující hodnotu primární proměnné. Funkce 2 a 9 jsou svázané s daným čidlem a proudovou smyčkou. Odpovědí jsou generické údaje nebo hodnoty podle specifikace pro nepodporovanou funkci. Funkce 0 byla implementována, aby mohl master zjistit dlouhou adresu a následně ji použít pro komunikaci. Nad rámec třídy 1 byla implementována funkce 13 pro čtení tagu, popisku a data. Tyto informace jsou uloženy v inicializačním textovém souboru.

Soubor `rs232API.c` se stará o komunikaci po sériové lince. V hlavičkovém souboru `modbusProtocol.h` jsou pak deklarace jednotlivých funkcí. Modul `rs232Init` inicializuje a spouští sériový port se zadanými parametry. Modul `re232Close` ho pak zavírá. Zbylé moduly zmíněné v předchozích odstavcích se starají o odeslání a příjem dat na otevřené lince.

Nakonec je použito několik hlavičkových souborů. Soubor `macros.h` zajišťuje makra pro posun jednotlivých bytů na správnou pozici, jelikož se více bytová slova odesílají po jednom bytu. Oba protokoly používají pro re-

prezentaci dat tzv. „Big-endian“. Další soubor, `defines.h`, definuje některé konstanty použité u obou protokolů. Všechny konstanty jsou popsány ve specifikaci HARTu. Soubor `struct.h` obsahuje struktury použité u obou protokolů. Poslední soubor `threads.h` se stará o mezivláknovou komunikaci. Obsahuje mutex a společná data.

3.3.2 Použité procedury

V programu je použito několik procedur. Všechny jsou okomentovány v přiloženém zdrojovém kódu. Některé z nich si nyní popíšeme.

Modul `modbusCRC`

Prvním takovým modulem (viz. ukázka kódu 3.1) je výpočet dvou bytového CRC pro Modbus. Do modulu se předají požadovaná data k zabezpečení a jejich délka. Následně se podle principu popsaného v kapitole 2.7 vypočte požadované zabezpečení.

```
unsigned int modbusCRC(unsigned char* data, unsigned char
    dataLen) {
  unsigned int crc = 0xFFFF;
  int pos, i;
  for (pos = 0; pos < dataLen; pos++) {
    crc ^= (unsigned int)data[pos];

    for (i = 8; i != 0; i--) {
      if ((crc & 0x0001) != 0) {
        crc >>= 1;
        crc ^= 0xA001;
      }
      else
        crc >>= 1;
    }
  }
  return crc;
}
```

Ukázka kódu 3.1: Procedura výpočtu zabezpečení zprávy na Modbusu

Modul `hartProcessReadPV`

Tento modul (viz. ukázka kódu 3.2) se stará o složení odpovědi HARTu na požadavek čtení primární proměnné. Nastaví kód funkce, velikost datové části, která bude vždy 7 bytů a status na bezchybný stav. Následně přes mu-

tex načte sdílená data a zavolá modul na připojení ostatních nezbytných položek jako preambule a zabezpečení.

```
void hartProcessReadPV(st_protocolQuery* protocolQuery,
    pthread_mutex_t* lockm, unsigned char* address, unsigned char
     command, unsigned char* buffer, char* msg, int pos) {
  msg[pos++] = command;

  msg[pos++] = 7;

  msg[pos++] = 0x00;
  msg[pos++] = 0x00;

  msg[pos++] = HART_UNTI_GRAMS;

  pthread_mutex_lock(lockm);
  hartAppendMsgFloat(msg, &pos, (float)shared_value.values[0]);
  pthread_mutex_unlock(lockm);

  hartSendMsg(protocolQuery->hHart, msg, pos);
}
```

Ukázka kódu 3.2: Procedura složení datové části zpracované funkce

Modul `hartAppendMsgFloat`

Modul `hartAppendMsgFloat` (viz. ukázka kódu 3.3) se stará o správné zařazení jednotlivých bytů do odesílané zprávy podle endianity. Nejvýznamnější byte bude v přenášené zprávě jako první, nejméně významný pak jako poslední přenášený.

```
void hartAppendMsgFloat(unsigned char* msg, int* pos, float
    value) {
  int intVal = *(int*)(float*)&value;
  msg[(*pos)++] = VAL_4_1(intVal);
  msg[(*pos)++] = VAL_4_2(intVal);
  msg[(*pos)++] = VAL_4_3(intVal);
  msg[(*pos)++] = VAL_4_4(intVal);
}
```

Ukázka kódu 3.3: Procedura řazení bytů do odesílané zprávy

3.4 Testování programového vybavení

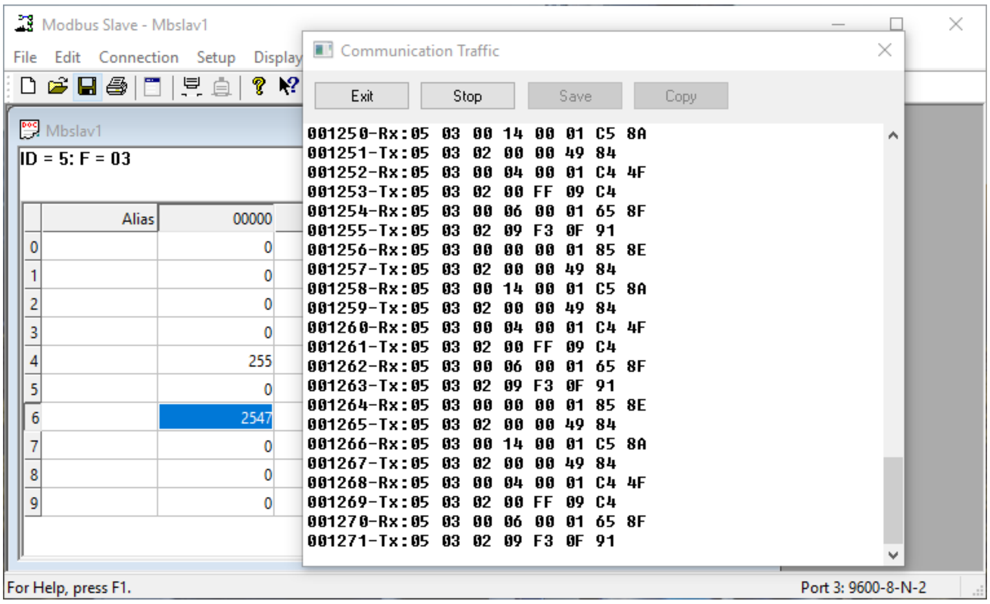
Pro otestování funkčnosti byl na straně Modbusu použit program Modbus Slave. Na straně HARTu byl použit vlastní testovací program vycházející

již z implementovaného HART slave. Byl upraven tak, aby na sběrnici HART dělal jednotku master.

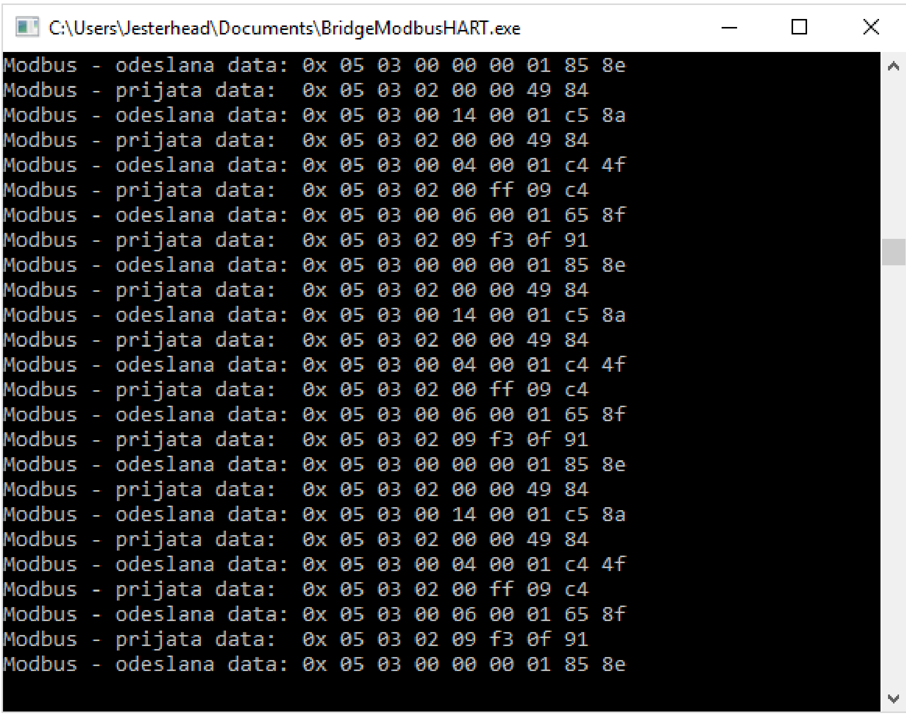
Tento program si nastavení sériové linky načte z přiloženého souboru. Po úspěšné inicializaci portu vyšle požadavek na krátkou adresu stanice, která v síti není. Slave by na požadavek, který není adresovaný jemu, neměl odpovídat. Následně pošle master požadavek na čtení unikátního identifikátoru na připojenou slave jednotku. Ta mu obratem pošle požadovaná data. Z unikátního identifikátoru si přečte dlouhou adresu jednotky a bude komunikovat přes tuto adresu. Nakonec master čte v deseti vteřinovém intervalu hodnotu primární proměnné, kterou most získá z programu Modbus Slave.

Testování probíhalo na PC s přidanými dvěma PCI kartami Axagon PCIA-S2. Dohromady měly tyto karty čtyři sériové porty. Jedna karta zajišťovala spojení vždy jednomu protokolu.

Nejdříve byl spuštěn program Modbus Slave a byl nastaven podle parametrů našeho Modbus mastera. V testovaném případě byla adresa zařízení nastavena na hodnotu 5, rozsah datové sekce byl zvolen od nuly do třiceti a vybrána funkce 4 (čtení uchovávacích registrů). Port byl nastaven na rychlost 9600 Bd, bez parity a byly použity dva Stop bity. Následně se spustil náš naprogramovaný most. Sériový port pro Modbus byl nastaven stejně jako v programu Modbus Slave. Sériový port pro HART byl z testovacích důvodů nastaven na rychlost 9600 Bd, lichou paritu a byl použit jeden Stop bit. Obě zařízení, slave i master, musí mít vždy stejně nastavené sériové porty. Po nastavení portů se okamžitě spustila výměna zpráv, jak je vidět v okně s programem Modbus Slave na obrázku 3.1 i v okně s konzolí našeho mostu na obrázku 3.2. Oba výpisy předávaných zpráv jsou stejné, kromě směru dat. Ty jsou logicky opačné. Na obrázku 3.1 je po levé straně vidět datová sekce s nastavenými hodnotami na adresách 4 a 6. Na tyto adresy se ptal náš most. V pravé části je vidět okno s komunikací. Vpravo dole je pak nastavení sériového portu. Na obrázku 3.2 se na začátku vypíše stav inicializací sériových portů a pak kromě chybových výpisů jen komunikace pro Modbus. Jelikož je most řešen dvouvláknově, jednotný výpis z obou vláken do jedné konzole není možný, a proto komunikace po HARTu se nevypisuje.

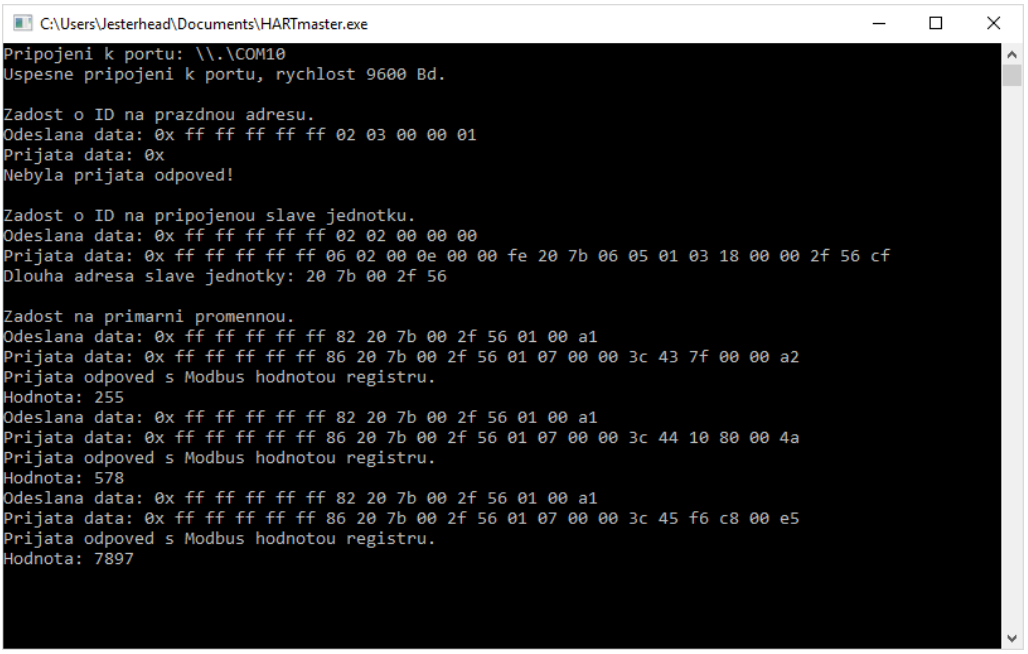


Obrázek 3.1: Okno programu Modbus Slave

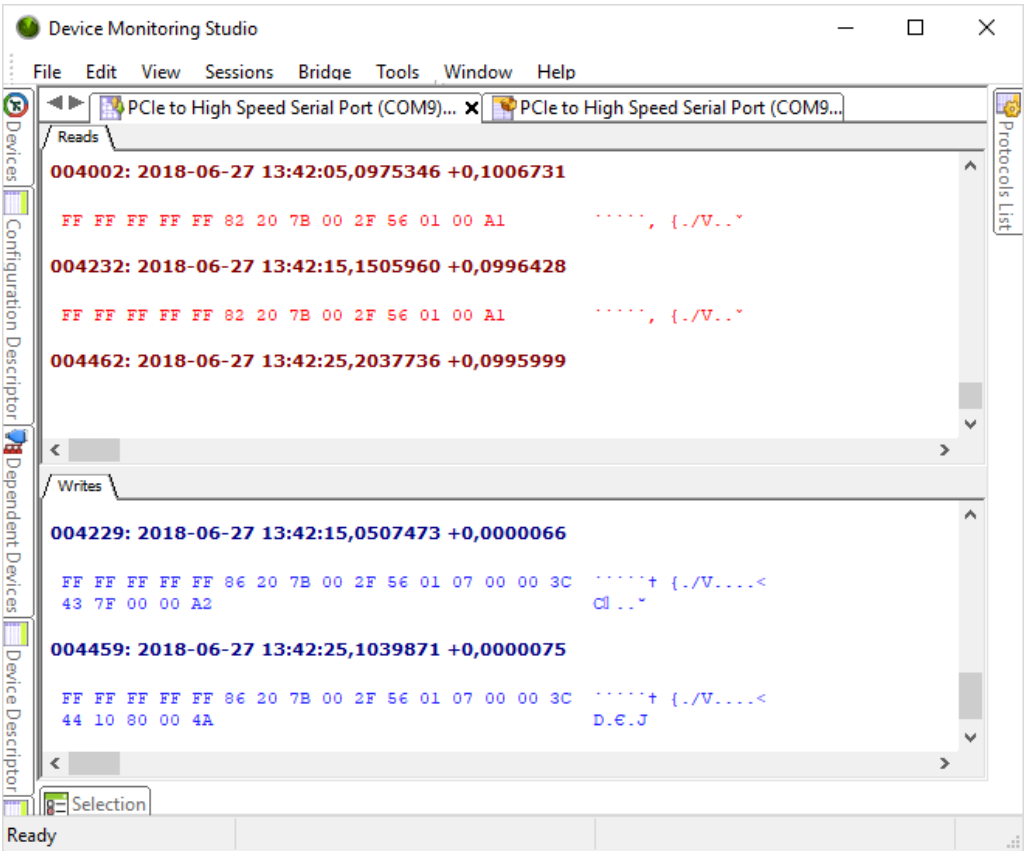


Obrázek 3.2: Konzole naprogramovaného mostu

Tuto komunikaci lze jednoduše sledovat programem Device Monitor Studio. Na obrázku 3.3 je vidět odesílání zpráv tak, jak bylo uvedeno na začátku této kapitoly. Master vyšle požadavek na čtení unikátního identifikátoru stanici s krátkou adresou 3. Tato stanice na sběrnici však není a žádný jiný slave odpověď nevyšle. Master tedy pošle stejný požadavek na stanici s krátkou adresou 2. Tuto adresu má nastaven náš most a řádně odpoví. Tím si master zjistil dlouhou adresu a následně již posílá požadavky na primární proměnou s dlouhou adresou našeho mostu. Stanice master se třikrát zeptala po deseti vteřinách na primární hodnotu slave jednotky. Most jí vrátil aktuální hodnotu požadované proměnné tak, jak byla měněna v programu Modbus Slave. Na obrázku 3.4 je výpis z programu Device Monitor Studio. Ten sledoval port s naším připojeným mostem na straně HARTu a komunikaci vypsal. V horní části obrázku lze červeně vidět to, co náš most přečetl z portu. Je to požadavek na primární proměnnou. V dolní části je modře označena odpověď mostu. První odpovědí se posílá hodnota 255 přenášena podle specifikace HARTu ve formátu float, tedy jako hodnota 0x 43 7F 00 00. Na další požadavek pošle hodnotu 578, tak, jak je vidět na obrázku 3.3. Můžeme si všimnout, že nesouhlasí časy příjmu požadavku a odpovědi. Jako kdyby náš most odpovídal na požadavek až po deseti vteřinách. To je dáno tím, že Device Monitor je nastaven na skrytí „prázdných“ zpráv, které na portu neustále čteme. Daný čas tak patří prvnímu odposlechu po odeslání poslední zprávy a při příjmu reálného požadavku po deseti vteřinách se již čas nevypíše.



Obrázek 3.3: Konzole HART masteru



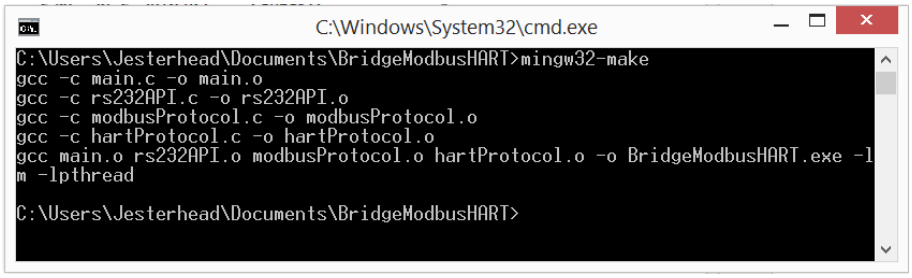
Obrázek 3.4: Sledování komunikace na portu s HART slave

3.5 Uživatelská příručka

Hlavní program `BridgeModbusHART.exe` a testovací program `HARTmaster.exe` jsou konzolovými aplikacemi. Aplikace jsou přeložitelné a spustitelné na systémech **Windows**. Pokud aplikace není přeložena (nemáme k dispozici `BridgeModbusHART.exe` nebo `HARTmaster.exe`), překládá se pomocí souboru `makefile` a je vyžadováno mít nainstalovaný překladač, např. **GCC** se správně nastavenou cestou *Path*.

3.5.1 Sestavení a spuštění programu

- otevřeme příkazovou řádku v adresáři se zdrojovými soubory
- spustíme překlad a sestavení programu příkazem `mingw32-make` (obrázek 3.5)



Obrázek 3.5: Překlad a sestavení programu v příkazovém řádku

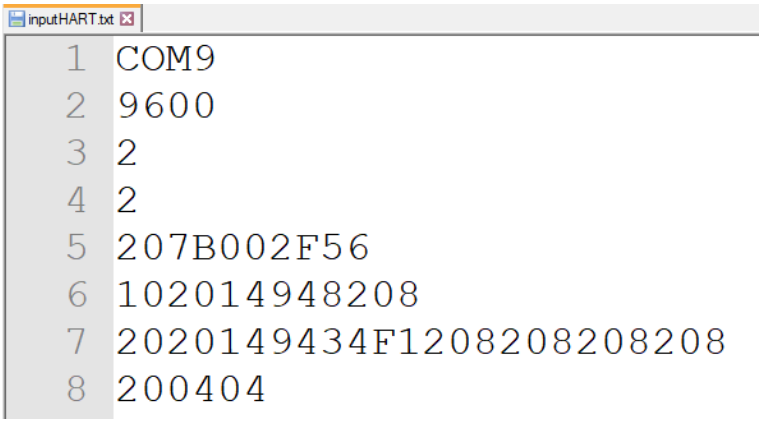
- program spustíme buď v příkazovém řádku příkazem `BridgeModbusHART.exe`, nebo přes souborového správce klasicky dvojklikem

Oba programy se překládají a spouštějí identicky. Samozřejmě každý ve svém adresáři.

3.5.2 Vstupní data

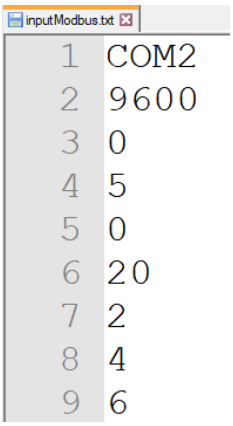
Pro správné spuštění programu je nutné mít spolu se spouštěcím souborem v adresáři textové inicializační soubory se správnými daty pro nastavení komunikace a mostu. Pro `BridgeModbusHART.exe` jsou to soubory `inputHART.txt` a `inputModbus.txt`. Pro `HARTmaster.exe` je to soubor `inputTesting.txt`.

Příklad `inputHART.txt` je na obrázku 3.6. Na prvním řádku je zvolen sériový port. Musí být uveden znaky COM a následuje číslo příslušného portu, ke kterému chceme, aby byl HART slave připojen. Na dalším řádku je baudová rychlost, pro testovací účely zvolena 9600 Bd. Na třetím řádku je parita. Číslo je bráno podle konstant knihovny starající se o sériové porty. Konstanta 0 znamená přenos bez parity, konstanta 1 přenos se sudou paritou a konstanta 2 s lichou paritou. Na čtvrtém řádku je naše krátká adresa. Na pátém řádku jsou identifikační údaje, které znamenají dlouhou adresu. Na dalších řádcích jsou uložena data, např. pro funkci 13 je to čtení tagu uloženého na řádku šest, popisku na řádku sedm a data na řádku osm. Tag a popisek jsou uloženy jako 6-bitové ASCII, takže na třech bytech jsou uloženy čtyři znaky.



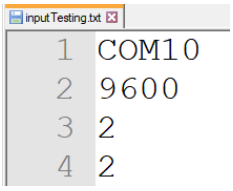
Obrázek 3.6: Textový soubor pro nastavení HART části mostu

Příklad `inputModbus.txt` je na obrázku 3.7. První čtyři řádky jsou stejné jako u předchozího souboru. U Modbusu však není krátká adresa, ale jen adresa slave zařízení. Na dalších dvou řádcích je datové rozmezí, ze kterého se budou číst proměnné. V našem případě od nultého do dvacátého registru. Tyto meze pak most kontroluje. Na sedmé řádce je počet proměnných, které se mají číst. HART může číst maximálně čtyři proměnné, proto je také tato hodnota omezena na čtyři. Na dalších řádcích jsou adresy, ze kterých se tyto proměnné mají číst. Zadané adresy musí být v intervalu zadaném na předchozích řádcích.



Obrázek 3.7: Textový soubor pro nastavení Modbus části mostu

Posledním souborem je `inputTesting.txt`. Struktura souboru je na obrázku 3.8. Na prvních třech řádcích je opět nastavení sériového portu a na čtvrtém řádku je krátká adresa našeho mostu.



Obrázek 3.8: Textový soubor pro nastavení HART mastera

3.6 Obsah CD

Přiložené CD obsahuje v **kořenovém adresáři** tuto dokumentaci v PDF formátu. V adresáři **BridgeModbusHART** se nachází *zdrojové kódy* naprogramovaného mostu Modbus - HART spolu se souborem *makefile* pro překlad těchto zdrojových kódů. Ve složce se také nachází již *přeložený spustitelný soubor*. V adresáři **HARTmaster** se nachází *zdrojové kódy* naprogramovaného testovacího programu HART master. Také zde je soubor *makefile* a již *přeložený spustitelný soubor*. V adresáři **LaTeX** jsou *zdrojové LaTeXsoubory*. V podadresáři **Obr** jsou všechny použité *obrázky*.

3.7 Eventuální budoucí rozšíření

Pro základní přenos naměřených dat mezi oběma protokoly bylo implementováno několik základních funkcí. Zde se nabízí asi největší možnost pro rozšíření této práce. Zejména na straně HARTu lze implementovat až několik desítek funkcí ze všech tří skupin.

S implementací dalších funkcí by bylo nutné data (např. tag, popisek, datum atd.) ukládat do připravených textových souborů.

Možným rozšířením by bylo zvolení lepších datových struktur pro uchování dat a nastavení mostu, než jakým je textový soubor.

Naprogramovaný most by bylo také možné vyzkoušet v nějaké reálné aplikaci. Na jedné straně mít např. čidlo komunikující přes protokol Modbus a na druhé straně řídící stanici komunikující přes protokol HART.

4 Závěr

Bakalářská práce se zabývala průmyslovými komunikačními sběrnicemi a spojením dvou navzájem nekompatibilních sběrnic, Modbusu a HARTu. Hlavním úkolem práce bylo:

- prostudování metod průmyslových komunikací po sériových sběrnicích
- navrhnutí základní struktury programového vybavení mostu Modbus - HART
- naprogramování softwarového vybavení mostu Modbus - HART
- provedení základních testů navrženého softwaru

V kapitole 2 jsme se obecně věnovali průmyslovým komunikacím a sítím. Rozebrali jsme si stručně historii a vývoj průmyslových sběrnic. Popsali jsme si referenční model ISO/OSI s jednotlivými vrstvami, architektury sítí, přenosy a kódování zpráv, modulaci zpráv a zabezpečení přenášených dat proti poškození. Poté jsme se podrobněji seznámili se sběrnicemi Modbus a HART a jejich implementovanými vrstvami.

V kapitole 3 jsme se podrobněji seznámili s požadavky na realizaci mostu Modbus - HART, popsali si implementaci mostu a následně podrobně rozebrali jednotlivé funkce a soubory naprogramovaného mostu.

Navržený software jsme podrobili testům funkčnosti. Na straně Modbusu byl zvolen zavedený program Modbus Slave, který má na základní použití dostatek možností. Na straně HARTu nebyl k dispozici vhodný program v roli řídící jednotky na sběrnici. Z toho důvodu byla naprogramována vlastní testovací řídící jednotka.

Testování probíhalo na jednom PC s přídavnými PCI kartami rozšiřující množství sériových portů. Tyto karty však nebyly příliš spolehlivé a ztěžovaly testování. Nebylo např. možné využít nižší baudovou rychlost přesně podle specifikace HARTu. Také velmi často padaly jejich ovladače a do restartu nebyly schopny komunikovat po sériové lince.

Programové vybavení se ale povedlo úspěšně otestovat a výsledky jsou vidět v příslušné kapitole. Testováním jsme ověřili základní funkčnost naprogramovaného mostu. Řídící jednotka odeslala požadavek na proměnnou uloženou v datové sekci Modbusu a most mu okamžitě odpověděl správnou hodnotu, jelikož si hodnoty registrů na straně Modbusu neustále kontroluje.

Literatura

[1] Cemin, D. Modbus C, 2013. Dostupné z: `https://sourceforge.net/projects/modbusc/`.

[2] *HART Command Sumamry Specification.* HART Communication Foundation, 2001. Revision 8.0.

[3] *HART Data Link Layer Specification.* HART Communication Foundation, 2001. Revision 8.1.

[4] *HART FSK Physical Layer Specification.* HART Communication Foundation, 1999. Revision 8.0.

[5] *HART Universal Command Specification.* HART Communication Foundation, 2001. Revision 6.0.

[6] Pefhany, S. *MODBUS PROTOCOL.* Trexon Inc., 2000.

[7] Ronešová, A. Přehled protokolu MODBUS. 2005.

[8] Zezulka, F. *Prostředky průmyslové automatizace.* VUTIUM, 2004. ISBN 80-214-2610- 1.

[9] Zezulka, F. Mezinárodní standardizace v oblasti průmyslových komunikačních sběrnic. *Automatizace.* 1998, 41, 7, s. 393. ISSN 0005- 125X.

[10] Zezulka, F. *Průmyslová automatizace.* Profesor thesis, Brno, 2000.

[11] Zezulka, F. – Hynčica, O. Průmyslový Ethernet II: Referenční model ISO/OSI. *Automa.* 2007, 13, 3, s. 86–90. ISSN 0001-0782.

[12] Zezulka, F. – Fiedler, P. – Bradáč, Z. Prostředky průmyslové automatizace. elektronické texty, 2003.