TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Komunikační protokoly pro chytré sítě

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Jakub Pecháček**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2019

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

Communication protocols for smart grids

Bachelor thesis

Study programme: B2612 – Electrical Engineering and Informatics
Study branch: 2612R011 – Electronic Information and Control Systems

Author: **Jakub Pecháček**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2018/2019**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení:	**Jakub Pecháček**
Název práce:	**Komunikační protokoly pro chytré sítě**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**
Vedoucí práce:	**Ing. Jan Kraus Ph.D.**
Rozsah práce:	**30—40 stran**

Zásady pro vypracování:

1. Seznamte se s protokoly pro přenos dat v energetice zejména v prostředí tzv. chytrých budov respektive chytrých sítí.
2. Ve vlastní aplikaci na konkrétních jednoduchých příkladech a s využitím existujících knihoven demonstrujte obvyklé funkce jednotlivých protokolů.
3. Prozkoumejte, implementujte, popište a otestujte odlišnosti jednotlivých řešení a v závěru práce shrňte přehlednou formou zjištěná fakta.

Seznam odborné literatury:

[1] FAN, Zhong, et al. Smart grid communications: Overview of research challenges, solutions, and standardization activities. IEEE Communications Surveys & Tutorials, 2013, 15.1: 21-38.

[2] APPASANI, Bhargav; MADDIKARA, Jaya Bharata Reddy; MOHANTA, Dusmanta Kumar. Standards and Communication Systems in Smart Grid. In: Smart Grids and Their Communication Systems. Springer, Singapore, 2019. p. 283-327.

[3] SEMBROIZ, David; RICCIARDI, Sergio; CAREGLIO, Davide. A Novel Cloud-Based IoT Architecture for Smart Building Automation. In: Security and Resilience in Intelligent Data-Centric Systems and Communication Networks. 2018. p. 215-233.

[4] MINOLI, Daniel; SOHRABY, Kazem; OCCHIOGROSSO, Benedict. IoT considerations, requirements, and architectures for smart buildings—Energy optimization and next-generation building management systems. IEEE Internet of Things Journal, 2017, 4.1: 269-283.

V Liberci dne .....

.....
Ing. Jan Kraus Ph.D.

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

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

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

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

Datum:

Podpis:

Poděkování

Tímto bych rád poděkoval vedoucímu mé Bakalářské práce, Ing. Janu Krausovi, Ph.D., za užitečné odborné rady a cenné připomínky, které přispěly k úspěšnému dokončení této práce.

Abstrakt

Tento dokument se zabývá Bakalářskou prací Fakulty mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci. Jejím cílem je popsat ročníkovou práci. Bakalářská práce obsahuje obecnou rešerši komunikačních protokolů pro chytré sítě a budovy. Následně byly pro vybrané protokoly naprogramovány jednoduché příklady pro demonstrování jejich funkcí. Ukázky byly programovány ve vývojovém prostředí Microsoft Visual Studio pomocí jazyka C#. Jedná se o konzolové aplikace, které jsou v závěru práce porovnány. Výsledkem této práce je rešerše komunikačních protokolů pro chytré sítě a budovy a jednoduché příklady, které se dají v praxi použít pro základ aplikací těchto protokolů.

Klíčová slova:

komunikační protokoly, chytré budovy, chytré sítě, C#, konzolové aplikace

Abstract

This document deals with the Bachelor thesis of the Faculty of Mechatronics, Informatics and Interdisciplinary Studies of the Technical University in Liberec. Its aim is to describe the coursework. The bachelor thesis contains general research of communication protocols for smart networks and buildings. Subsequently, simple examples for demonstrating their functions were programmed for selected protocols. The samples were programmed in the Microsoft Visual Studio development environment using C#. These are console applications that are compared at the end of the thesis. The result of this work is a review of communication protocols for smart networks and buildings and simple examples that can be used in practice for the basis of application these protocols.

Key words:

communication protocols, smart buildings, smart grids, C#, console application

Obsah

Seznam obrázků

1 Úvod

Hlavním cílem této bakalářské práce je udělat obecnou rešerši nejvíce používaných komunikačních protokolů jak v chytrých sítích, tak i v chytrých budovách. Následně jsou v této práci vybrány čtyři z těchto protokolů. Pro tyto protokoly jsou následně naprogramovány jednoduché aplikace, na kterých jsou zobrazeny jejich funkce přenosu dat a jejich odlišnosti, případně jejich specifické funkce.

Práce je rozvržena do více částí. První část této práce se zabývá obecnou rešerší. Pro výběr těchto protokolů byly nastaveny určitá kritéria. Hlavním kritériem pro výběr komunikačních protokolů je použití v energetice, případně v průmyslové automatizaci. Dalším kritériem pro výběr protokolů do rešerše bylo širší použití daného protokolu. Protokoly využívané jednou nebo dvěma firmami nejsou tolik zajímavé. Druhá část této práce se zabývá výběrem protokolů pro následné programování jednotlivých aplikací. Třetí část je samotný popis jednotlivých aplikací a ukázka funkcí jednotlivých protokolů v daných aplikacích.

V závěru práce jsou shrnuta jednotlivá zjištěná fakta. Následně jsou zde popsány rozdíly jednotlivých aplikací daných protokolů, rozdíly, jak jsou data posílány, jaké funkce tyto protokoly odlišují a zda jdou v protokolech posílat veškerá data. Dále jsou v závěru popsány osobní poznatky k výběru protokolů pro programování aplikací, výběru knihoven pro aplikace, i jednotlivé poznatky k samotnému programování s knihovnami.

2 Přehled komunikačních protokolů

2.1 Chytré sítě

Chytrá síť je silová elektrická a komunikační síť, která umožňuje regulovat výkon a spotřebu elektrické energie v reálném čase. Chytré sítě mít jak malé i globální rozměry.

Principem těchto sítí je obousměrná interaktivní komunikace mezi všemi prvky této sítě. Prvky mohou být spotřebiče, výrobní stroje nebo klidně i zadávací panel požadavků na danou síť. Nevýhodou těchto sítí je bezpečnostní riziko odposlouchávání citlivých údajů.

Inteligentní sítě mohou dosáhnout stádia plné automatizace. To znamená, že síť obsahuje kontrolní a řídicí systém. Senzory monitorují chování dané sítě a dávají informaci řídicímu systému. Tento systém data vyhodnotí a provede zásahy do sítě tak, aby nenastala porucha, nebo se dostatečně rychle odstranila a síť běžela a fungovala podle původního zadání.

Mezi protokoly pro chytré sítě se mohou zařadit tyto protokoly:

1. DLMS / COSEM
2. IEC 61850
3. IEC 60870-5-104
4. Z-wave
5. ZigBee
6. 6LoWPAN

2.1.1 DLMS / COSEM

Komunikační protokol DLMS (Device Language Message Specification) se v literatuře velmi často objevuje pod názvem DLMS/COSEM. Nazývá se tak, jelikož to jsou dvě rozdílné věci, které se však používají zásadně spolu. DLMS/COSEM používá DLMS komunikační protokol a COSEM rozhraní, které definuje třídy v

aplikační a transportní vrstvě protokolu DLMS. Dá se tedy psát oboje, jak DLMS tak i DLMS/COSEM protokol.

Je navržen pro podporu zasílání dat mezi distribučními zařízeními, služby vzdáleného měření dat, vzdálené ovládání zařízení. DLMS protokol má vlastní řadu norem pouze k využití v energetice, tato řada se nazývá IEC 62056.

Komunikace

Data při komunikaci v DLMS jsou rozřazeny do tříd. Data jsou rozřazeny podle hodnot nebo informací, které obsahují. Třídy obsahují objekty. Objekt je kolekce atributů a metod. Atributy obsahují samotná data. Atributy obsahují názvy a hodnoty. Metody jsou určité funkce, které mohou data poskytovat nebo měnit hodnoty atributů. Samotné třídy jsou kolekce objektů.

Komunikace DLMS protokolu funguje na hierarchickém principu. DLMS protokol obsahuje hlavní a několik nižších vrstev (uživatelská, objektová, aplikační a nižší třídy zpracovávající data). Uživatel má přímý přístup k objektům neboli třídám, takže i samotným datům. K těmto datům se dostává pomocí jmen a samotných kódů. Uživatel zadává i parametr co se s daty bude dít. Tento parametr bude zpracován podle příslušné třídy a následně předán do aplikační vrstvy. Aplikační vrstva vloží výstup z objektové vrstvy do samotného zařízení, kterému data přísluší a celý tento požadavek, co se má dít, předá do nižších vrstev, kde se data již zpracovávají v jednotlivých zařízeních.

Klient a server mezi sebou mohou komunikovat ve všech vrstvách. Je však potřeba nejdříve provést asociace. Asociace se provádí při navazování spojení v aplikační vrstvě a při asociaci si klient a server stanoví určité komunikační parametry, které při komunikaci musí dodržovat. Po navázání spojení se může klient dotazovat přímo na data v serveru a naopak.

Protokol DLMS je nadstavba UDP/IP protokolu. Komunikace v DLMS je binární, ale při použití IEC 62056 se provádí komunikace v ASCII.

Použití

DLMS protokol se může použít jak v energetice tak i v běžném domácím použití. Má nevýhodu, že je třeba udržovat stálé a stabilní spojení komunikace. Další nevýhodou je, že DLMS pracuje jako nadstavba UDP/IP, protože potřebuje větší šířku pásma pro komunikaci a je pomalejší, jelikož při navazování komunikace je složitější.

DLMS protokol bych tedy použil v sítích kde je zaručená stabilní komunikace, nezáleží na rychlosti, zařízení jsou napájeny síťovými zdroji.

Nejpoužívanější knihovnou pro tento protokol je open source knihonva Gurux. Tato knihovna slouží k implementaci funkcí pro server i clienta.

2.1.2 IEC 61850

IEC 61850 je soubor norem pro komunikaci mezi zařízeními především v energetických rozvodnách a elektrických soustavách. IEC 61850 dále stanovuje požadavky, které jsou potřeba zajistit při komunikaci v rozvodnách. Obsahuje komunikační protokoly, ale také i určité standardy pro řízení zařízení.

Cíle protokolu

Hlavním důvodem vytvoření tohoto protokolu bylo, že ve světě bylo nespočet různých komunikačních protokolů pro komunikaci v energetice a každý měl svoje vlastní pravidla. Tudíž při tvorbě chytré sítě, pro rozvodnu, bylo potřeba brát ohled na výrobce a jaké zařízení dokáže komunikovat s jakým protokolem.

Cílem tohoto protokolu bylo tedy sjednotit pravidla komunikace a umožnit vytvoření sítí, v kterých budou komunikovat mezi sebou zařízení bez ohledu na výrobce. Tyto zařízení, neboli IED (Intelligent Electronic Device) zajišťují ochranu a provoz rozvodny, tudíž v žádném případě nemohlo nastat, že se přerušila komunikace mezi dvěma zařízeními z důvodu, že každý výrobce si jeho komunikaci udělal po svém.

Komunikace

Architektura komunikačních protokolů pro normu IEC 61850 je typu klient a server. Avšak má výhodu, že se snaží odstraňovat hlavní nevýhody této architektury tím, že umožňuje i klientům řídit přenos dat a komunikaci. Tato výhoda dovoluje přesunout řídící a komunikační funkce blíže k potřebným zařízením a umožňuje zvýšit komunikační rychlosti a stabilitu přenosu, jelikož přenos může být navázán na menší vzdálenosti.

Komunikace probíhá formou publish/subscribe. Data od klientů jdou do určitého serveru a ten následně rozesílá těm, co si data objednají. Může zde být i režim multicasting. Server posílá data všem účastníkům sítě, ale data čtou pouze ti, co si je objednali (subscribers).

Dále umožňuje integrovat všechny komunikační, řídicí, ochranné a měřící funkce pro chod a ochranu rozvoden.

Data se posílají v tzv. objektech. Tyto objekty spojují data a programy, takže veškeré informace a funkce se nacházejí na jednom místě. Díky tomu je pro uživatele i jednotlivá zařízení mnohem snazší přístup k datům a funkcím s nimi spojených.

IEC 61850 kombinuje ethernet s vysokým výkonem a zabezpečením. Tím poskytuje vysokorychlostní ochranu, uzamčení a přepínání. Toto je zajištěno vysokorychlostním přenosem kritických dat.

Použití

IEC 61850 se používá pouze pro komunikaci chytrých sítí v energetice. Největší výhodou je vzájemná kompatibilita zařízení v energetice. Využívá se pro rozvodny, komunikaci mezi rozvodnami a elektrárnami, vysokorychlostní komunikaci všude v energetice.

Pro tento protkol se nejčastěji používá open source knihovna OpenIEC61850. Tato knihovna poskytuje veškeré funkce protkolu v jazyce Java.

2.1.3 IEC 60870-5-104

Standard IEC 60870-5-104 definuje funkci systémů a protokolů používaných pro vzdálenou správu dat, jejich prohlížení a řízení. Tato norma se zabývá pouze daty v automatizačních systémech a energetických automatizačních systémech. Popisuje komunikační profil pro zasílání dat mezi dvěma zařízeními.

IEC 60870-5-104 je rozšíření IEC 60870-5-101 o několik transportních funkcí protokolu TCP/IP. Jedná se tedy o další nadstavbu protokolu. Lze tedy využívat téměř veškeré funkce protokolu TCP/IP.

Komunikace

Komunikace v protokolech vycházejících z normy IEC 60870-5-104 pracují na modelu master-slave. To znamená, že určité zařízení (master), při komunikaci řídí přenos dat a pracuje s nimi a další (slave) data poskytují a pracují tak, jak master určí. Toto se provádí především při komunikaci na sběrnici. Master tyto požadavky na další zařízení rozesílá postupně, nikoli najednou. Každé zařízení slave reaguje pouze na data, která mu jsou určeny, ostatní ignoruje.

Tento typ komunikace má velkou nevýhodu, jelikož data na sběrnici může odposlouchávat jakékoli připojené zařízení. Při komunikaci pouze mezi stroji, však na toto bezpečností riziko nemusíme brát příliš velký ohled. V rozvodnách je tato bezpečnost důležitější.

U komunikace mezi stroji většinou přenos dat řídí PLC. Ostatní akční členy, regulátory a podobné zařízení, jsou slave jednotky. V energetických soustavách zpravidla soustavu řídí počítač. Řídí zbylé jednotky, které se starají o chod elektráren a rozvoden.

IEC 60870-5-104 obsahuje implementované funkce vylepšující architekturu master-slave. Jedna z funkcí je například, že slave může obsahovat kritické informace, které jsou potřeba zpracovávat přednostně. Tyto informace pak mohou mít vyšší prioritu a mohou být doručeny masteru i bez toho aniž, by si o ně řekl. Tato funkce výrazně zlepšuje bezpečnostní chod energetických sítí.

Použití

Komunikační protokoly normy IEC 60870-5-104 se mohou použít nejen v energetice. Mnohem více, než v energetice, se využívají u automatizačních systémů, ale použití v energetice může mít také své výhody. Dále se také mohou hodit všude, kde je potřeba zajistit chod událostí po sobě, jelikož data (události) mohou obsahovat i čas, kdy k nim došlo.

Mohou být vhodné pro výrobní linky, energetické rozvodny, sítě, kde je potřeba kontrolovat sekvenční chod a sítě, kde je potřeba zajistit rychlé odpovědi v případě poruchy. K použití tohoto protkolu je vejvíce rozšířená knihovna IEC 60870-5-101/104 C#/.NET v jazyce C#. Tato knihovna poskytuje funkce serveru i clienta.

2.1.4 Z-wave

Z-wave je komunikační protokol vytvořený primárně pro domácí automatizace a IoT. Byl vyvinut pro ovládání, monitoring a zajištění určitých stavů v budovách určených pro bydlení a v menších budovách pro soukromé použití. Z-wave je určen k bezdrátové komunikaci. Protokol je standardizován a lze propojit mezi produkty od různých výrobců. Byl navržen pro zařízení, které nemají vysoké energetické požadavky a pracují pouze s malými objemy dat.

Komunikace

Pro komunikaci mezi jednotlivými zařízeními používá vlastní vyhrazenou šířku pásma. Tato šířka pásma neprotíná a ani se neblíží šířkám pásma Wi-Fi nebo Bluetooth. Využívá 800 – 900 MHz v závislosti na kontinentu, kde je protokol použit (v Evropě se tato frekvence pohybuje v okolí 869 MHz).

Protokol Z-wave má smíšenou topologii sítě. To znamená, že některá zařízení v sítí mohou být propojeny s více než jedním dalším zařízením v síti. Tato topologie se někdy také nazývá síťová topologie.

V síti se nachází centrální prvek, který rozdává příkazy jednotlivým zařízením. Toto má za důsledek zpomalení chodu sítě a tak Z-wave protokol podporuje rozdávání příkazů každého zařízení v síti. Dokáže sledovat a řídit průběh jiných zařízení. Centrální zařízení tak slouži pouze ke kontrole nebo k novému nastavení sítě uživatelem.

Použití

Použitím protokolu Z-wave získáme velmi rychlou komunikaci s malou odezvou. Nevýhodou této rychlé komunikace s malou odezvou je dosah a objem dat. Lze posílat pouze velmi malé packety. Vzdálenost, na kterou jsou schopna dvě zařízení

spolehlivě komunikovat, se udává do 30 až 40 metrů. Pokud je využit na volném prostranství, ne uvnitř budovy, zvedne se dosah až na 100 metrů.

Výhodou použití toho protokolu je, že podporuje takzvanou schopnost "plug & play". To znamená, že zařízení stačí připojit do sítě a okamžitě se samo nastaví a začne komunikovat s dalším zařízeními. Další výhodou je, že stačí připojit pouze jedno zařízení k internetu a následně lze celou síť kontrolovat a nastavovat pomocí vzdáleného webového prohlížeče nebo chytrého telefonu. Použití toho protokolu je vymezeno výhradně pro domácí účely chytrých domů a nelze tento protokol použít v energetice ani žádném dalším průmyslovém odvětví. Neobsahuje žádné nadstavby nebo vlastní normy pro použití v energetice. Tento protokol se čistě soustředí na chytré domácnosti.

Nejčastější použití Z-wave protokolu probíhá přes knihovnu OpenZWave. Jak už nadpis napovídá, jedná se o open source knihovnu. Knihovna poskytuje třídy pro přijímaní a vysílání dat a mnoho dalších.

2.1.5 ZigBee

ZigBee je bezdrátový komunikační protokol postavený na základech protokolu Bluetooth, proto také spadá do stejné skupiny norem. Stejně jako Bluetooth je určen pro zařízení s nízkým výkonem. ZigBee podporuje komunikaci na delší vzdálenosti bez, přímé radiové viditelnosti jednotlivých zařízení. ZigBee byl primárně navržen pro průmyslové použití. Nejedná se tedy o pokus vytvoření konkurenčního protokolu pro Bluetooth, ale o jeho doplněk pro průmysl.

ZigBee byl navržen pro aplikace, kde není Bluetooth vhodný. Jedná se tedy o hodně podobný protokol, který však neslouží pro osobní využití, ale slouží primárně pro průmyslové aplikace.

Komunikace

Komunikace tohoto protokolu je prováděna v úzkém komunikačním pásmu. Toto pásmo se pohybuje téměř totožně s pásmem protokolu Z-wave, nenarušuje tedy pásma Wi-Fi ani Bluetooth.

ZigBee může mít 3 druhy topologie. Prvním a nejvíce používanou je hvězdicová topologie s centrálním zařízením. Druhým typem topologie je hvězdicová topologie. Tato topologie umožňuje delší komunikační vzdálenosti. Pokud u této topologie využijeme redundantní spojení, vznikne stejná topologie jako u Z-wave, tedy smíšenou neboli síťovou topologií. Redundantní spojení znamená, že se vytvoří spojení mezi zařízeními, kde to není nutné, ale v případě výpadku se tato spojení mohou použít. Zařízení je spojeno s každým dalším zařízením v dosahu. Díky tomu není třeba dbát na pořadí umístění prvků, ale stačí pouze kontrolovat, zda se nachází v dosahu alespoň jednoho dalšího zařízení, které je připojeno do sítě.

ZigBee si dělí zařízení v síti na dva druhy. První se nazývá FFD – Full Functional Device, neboli zařízení s plnou funkčností. Druhý typ je zařízení s omezenou funkčností a ten se nazývá RFD – Reduced Functionality Device. Zařízení s plnou funkčností obsahují kompletní protokol a mohou využívat veškeré služby, které ZigBee nabízí. Zařízení s omezenou funkčností obsahují pouze nezbytné rámce protokolu a slouží pouze jako koncová. Tyto zařízení jsou omezena z důvodu omezeného hardwaru.

Protokol dovoluje i implementaci zabezpečení. Toto zabezpečení využívá takzvané klíče, které mohou být již přiřazeny k MAC adrese zařízení, nebo vytvořeny při instalaci zařízení do sítě. Vytváření klíčů řídí centrální zařízení. Toto zařízení také kontroluje veškerou komunikaci a tím může kontrolovat i klíče. Tímto způsobem protokol zajišťuje určitý stupeň bezpečnosti narušení.

Jednotlivá zařízení jsou při komunikaci adresovány pomocí binárních adres. Tyto adresy mohou mít dvě podoby, dlouhou nebo krátkou verzi. Použití krátké verze je určeno především pro RFD zařízení. Použití této zkrácené verze ušetří 48 bitů paměti zařízení. Je to tedy použito u extrémně jednoduchých zařízení. Při komunikaci dvou sítí mezi sebou je k adresaci použito ID sítě.

Použití

Při použití toho protokolu získáme síť, která má velice pomalé přenosové rychlosti, avšak přenos dat je poměrně stabilní. Přenos je dále odolný oproti rušení jiným bezdrátovým přenosem, jelikož přenos probíhá ve vzdáleném pásmu od běžných přenosovým pásem.

Použití je také velice vhodné pro sítě, kde jsou použity zařízení s bateriemi. Protokol má velice nízkou spotřebu energie. Není vhodný pro aplikace, kde je potřeba přenášet velký objem dat s vysokou rychlostí. Pro řízení energetiky je tedy nevhodný, protože v energetice je rychlost přenosu kritických dat téměř nutností.

Díky těmto vlastnostem se s protokolem setkáme v průmyslové automatizaci, zdravotnictví (monitorování pacientů), chytré domácnosti (zabezpečení, osvětlení, topení), počítačovém průmyslu (bezdrátové periferie).

Nejčastěi se tento protokol používá s knihovnou SiplmeZigBee. Tato knihovna je určena pro Arduino a poskytuje funkce pro komunikaci v síti ZigBee.

2.1.6 6LoWPAN

Protokol 6LoWPAN spadá pod úplně stejnou normu jako ZigBee. Každý z nich si však postavil jinou nadstavbu nad touto normou a každý pracuje trošku jinak. Základ však mají stejný a pracují ve stejném frekvenčním pásmu. Stejně jako ZigBee a Z-wave byl navržen pro nízko výkonové sítě. Byl navržen pro sítě, kde je rychlost přenosu dat relevantní a vyžadují především spolehlivost přenosu dat. 6LoWPAN propojuje Internet protokolu (IP), konktrétně IPv6, a bezdrátové sítě definované

normou IEEE 802.15.4. Tato norma definuje bezdrátové komunikační sítě. Hlavním cílem tohoto protokolu bylo vytvořit protokol, který není náročný na spotřebu energie a zároveň zařízení mohou být připojeny k internetu a využívat Internet Protokol verze 6. Tím získáme síť, která se nemusí připojovat do internetu, protože sama je část internetu a lze se na každá zařízení připojit celkem snadno z jakéhokoli zařízení, které je také připojeno na internet.

Komunikace

Architektura sítě má stejnou topologii jako ZigBee nebo Z-wave, tedy síťovou. Každé zařízení je propojeno se všemi v dosahu. Správném rozmístěním zařízení se dá zvětšovat či zmenšovat dosah sítě. V síti se však nachází další 2 zařízení navíc oproti ZigBee nebo Z-wave. Nachází se zde webový server a takzvaná brána. Brána odděluje internet od naší vnitřní sítě. Webový server zase publikuje data na internet a zpracovává data z něj. V síti se může nacházet centrální zařízení, pomocí kterého uživatel rozdává příkazy, jak se síť má chovat, nutné to ale není. Jako zařízení, pomocí kterých uživatel bude rozdávat příkazy, může sloužit jakékoli zařízení připojeno k internetu a přes bránu se uživatel může vzdáleně připojit do sítě. Každé zařízení v sítí může číst data z ostatních a podle nich přizpůsobit svůj chod. Zařízení mohou být aktivní, mohou nastavovat ostatní zařízení, nebo pasivní, ty mohou jen prezentovat svoje data.

Ke komunikaci může 6LoWPAN využívat jak IEEE 802.15.4, tak i Ethernet nebo Wi-Fi. 6LoWPAN využívá veškeré funkce IPv6 a pomocí toho protokolu se i adresuje. Může využívat i systém zabezpečení "Acces Control List". Toto zabezpečení slouží k obraně proti neoprávněnému přístupu z internetu do vnitřní sítě.

Použití

6LoWPAN nemá žádné přímé určení. Je však navrhnut pro pomalé přenosové rychlosti, protože přenos dat probíhá po malých rámcích. Jeho výhodou je snadná adresace a také využívání veškerých funkcí IPv6. Nevýhodou je nutnost znalosti IP protokolu, pro schopnost naprogramovat síť s protokolem 6LoWPAN. Dále je nutnost připojení do internetu, což může způsobit jisté bezpečnostní riziko. Při soukromém použití nás toto riziko nemusí příliš zajímat, ale při použití v energetice už se toto riziko zvyšuje. Z těchto důvodů se tento protokol z mého pohledu hodí především pro domácí aplikace. Dále je možnost použití pro průmyslové aplikace, kde není potřeba velká přenosová rychlost, ale je zde potřeba určitá spolehlivost s možností existence určitého bezpečnostního rizika narušení nebo odposlouchávání dat.

Nejvíce rozšířenou knihovnou pro tento protokol je Contiki. Tato knihovna poskytuje komunikaci protokolů 6lowpan, RPL a CoAP.

2.2 Chytré budovy

Chytrá budova je taková budova, která využívá technologii ke sdílení informací o tom, co se děje v jednotlivých systémech budovy tak, aby optimalizovala její účinnost. Toto sdílení informací se používá především k automatizaci různých procesů, například topení, klimatizace, bezpečnost a další.

Vybudování chytrých budov je velice nákladné. V případě rozhodnutí o vybudování, jsou tyto náklady akceptovatelné, protože po vybudování chytrých budov se zvýší výkonnost budovy a tím se ušetří peníze - ať už jen na topení nebo klimatizování. Často jsou tyto systémy naistalovány špatně, takzvaně neinteligentně, a tudíž budova neslouží jako chytrá budova, ale pouze jako budova v chytrými aplikacemi.

Hlavním cílem inteligentních budov je zabránit plýtvání energiemi a zdroji. Dalším cílem je snížení nákladů, zvýšení výkonnosti a energetické účinnosti budov.

Následujcí protokoly pro chytré budovy se řadí mezi nejpoužívanější :

1. MQTT
2. Modbus
3. BACnet
4. KNX
5. CANopen
6. Profinet
7. OPC UA
8. Další méně rozšířené protkoly (DNP3, LonTalk, Ego-n, Synco living a Elko EP)

2.2.1 MQTT

MQTT, neboli MQ Telemetry Transport, je M2M (machine to machine) nebo IoT komunikační protokol. Byl navržen pro extrémně nenáročné přenášení zpráv a dat. Jedná se o jednoduchý a nenáročný komunikační protokol.

Cíle protokolu

Hlavním cílem protokolu je snížení zatížení sítě a omezení požadavků na zdroje jednotlivých zařízení, když je potřeba určitá komunikace mezi jednotlivými zařízeními.

Navrženo pro jednoduchá zařízení, sítě s velkou latencí (pingem), sítě s úzkou šířkou komunikačního pásma, nespolehlivá nebo poruchová zařízení i celé sítě.

Jedná se tedy o protokol, jenž nezatěžuje síť při komunikaci. MQTT se snaží zajišťovat určitý stupeň spolehlivosti, který udává zajištění doručení zprávy mezi zařízeními v síti.

Komunikace

MQTT protokol je založen na principu publish/subscribe (zveřejnit/odebírat). To znamená, že síť má vlastní centrum a klienty. Centrum se nazývá MQTT Broker. Klienti mohou být různé aplikace, senzory, webové služby nebo samotná zařízení v síti, které data vytváří nebo je pouze zpracovávají.

Funguje na principu, kdy každý klient (například teploměr) produkuje data, které následně publikuje do daného centra. Dané centrum následně tyto data rozešle všem klientům, kteří si o tento protokol zažádali (subscribe). Data obsahují vlastní název téma, nazývaný topic, podle kterého si klienti data objednávají, nebo Broker data rozesílá.

Protokol MQTT byl vyvinut jako nadstavba protokolu TCP/IP.

MQTT také podporuje určitý stupeň zabezpečení komunikace. Od určité verze lze do dat vložit požadavky na uživatelské údaje. Lze do packetů hlavičky vložit jméno a heslo. Toto zabezpečení využívá šifrovanou komunikaci.

Nevýhodou tohoto zabezpečení je výrazné zvýšení zatížení sítě, které se celý protokol snaží co nejvíce snížit. Pokud tedy není šifrování nutné, využívá se.

Použití

MQTT protokol se využívá a hodí se pro sítě jak v energetice, tak i v domácím použití. Dá se použít všude, kde jsou sítě s obousměrným přenosem dat, sítě s úzkou šířkou komunikačního pásma, sítě s bezdrátovými zařízeními.

Je výhodné v tom, že nezatěžuje síť a tím snižuje požadavky na zdroje zařízení, tak se velice hodí do sítí, kde jsou použity bezdrátové zařízení. Hodí se do takových sítí, protože prodlužuje výdrž baterie oproti výdrži při použití jiného komunikačního protokolu. Další výhodou je, že nepotřebuje velkou šířku pásma, tudíž se může použít i v případě, že se v okolí vyskytuje nějaká další komunikace nebo bezdrátový přenos dat.

Nejrozšířenější knihovnou pro MQTT je Mosquitto. Tato knihovna poskytuje veškeré funkce protokolu.

2.2.2 Modbus

Protokol Modbus je otevřený komunikační protokol, který umožňuje vzájemnou komunikaci mezi různými zařízeními. Umožňuje přenášet data po různých typech sítí i sběrnic.

Původně byl využíván pouze jednou firmou pro průmyslovou komunikaci, ale díky jeho jednoduchosti se rozšířil mezi veřejnost a v dnešní době je využíván poměrně hojně. Toto rozšíření způsobilo to, že není standardizován ani objektově orientovaný. Každá jeho aplikace může být jiná a samotný protokol neumí nic speciálního, ale jelikož je otevřený, tak v každá aplikaci prakticky může umět vše. Jeho výhodou je to, že jsou definovány pouze vstupy a výstupy. Samotná struktura je pak na uživateli.

V dnešní době je zde pokus o standardizaci protokolu. Je zde nadstavba tohoto protokolu s myšlenkou, že každé zařízení bude obsahovat bloky. Každý blok, který bude mít určitá data o daném zařízení, bude na stejné adrese na všech zařízeních. Například ID zařízení bude na adrese 1 na všech zařízeních. Pozor, tato adresa není stejná adresa celého zařízení. Tato adresa slouží pouze k vnitřní adresaci na dané bloky. Tento pokus by řešil problém se sjednocením protokolu mezi firmami, kteří Modbus využívají.

Komunikace

Modbus je založen na architektuře "master-slave". Na sběrnici je jedno zařízení master a několik zařízení slave. Master posílá dotazy a slave pouze odpovídá. Master posílá dotazy pouze na zařízení, od kterých žádá odpovědi. Toho docílí adresováním, avšak adresování se liší v závislosti na použití sítě nebo sběrnice. Pokud použijeme Ethernet, adresujeme pomocí IP adres. Ale pokud použijeme sériové rozhraní, adresa je hexadecimální číslo 0 až 255. Při použití sériové linky může Modbus pracovat v režimu ASCII. Tím se data i adresace převede do ASCII znaků.

Komunikace probíhá stylem požadavek - odpověď. Master vyšle data s adresou zařízení, požadavkem co má zařízení udělat a případně daty. Adresu zpracují všechny zařízení na sběrnici, ale další data pouze to zařízení, které má požadovanou adresu. Pokud máme Modbus s vnitřními bloky v zařízení o kterých jsem již mluvil, adresace bloku je vložena mezi adresu zařízení a vlastní požadavek funkce.

Velikost packetů není předem určena a čistě závisí na uživateli, jak si velikost určí v aplikaci.

Použití

Modbus je vhodný pro použití především u průmyslových zařízení a systémech automatizace budov. Je vhodný pro použití při komunikaci mezi rozdílnými sítěmi

nebo sběrnicemi, jelikož tento protokol je otevřený a může být upraven na jakoukoli aplikaci.

Při použití tohoto protokolu je zde jedna velká nevýhoda, která může být občas považována na výhodu. Je to právě to, že samotný protokol není standardizován a jsou předem definovány pouze vstupy a výstupy. Zda jsou digitální nebo analogové a zda se mají chovat jako vstup, výstup nebo oboje. Toto je výhoda, když potřebujeme specifickou aplikaci, ale je to také nevýhoda při snaze o spojení dvou aplikací od dvou vývojářů. Při použití vnitřních bloků zařízení tuto nevýhodu eliminujeme a je poměrně jednoduché a přehledné sjednotit více aplikací v jednu.

Tento protokol se tedy dá využít v jakékoli průmyslové aplikaci. Jeho jediná nevýhoda spočívá ve snaze o přenesení kritických dat. Tyto data nemůže samotné slave zařízení "vnutit"masteru. Když už se tato data dostanou na řadu ke komunikaci, musí projít přes samotný master k zařízení ke kterému patří.

Nejpoužívanější knihovna tohoto protokolu je knihovna libmodbus. Jedná se o knihovnu pro jazyk C a podporuje jak přijímání, tak vysílání data přes Ethernet nebo sériovou linku.

2.2.3 BACnet

Protokol BACnet slouží k celosvětové normalizaci a standardizaci automatizace budov. Jedná se o snahu zaručit univerzálnost a zaměnitelnost jednotlivých prvků tohoto protokolu. Snaha o zaručení fungujícího systému při použití zařízení od různých výrobců. Výsledkem tohoto protokolu je tedy univerzální předpis různých zařízení a funkcí.

BACnet byl navržen především k využití k automatizaci a řízení budov. Byl navržen pro komunikaci pomocí internetového připojení a zajištění přístupu pomocí IP adres.

Komunikace

Každé zařízení je reprezentováno jedním či více objekty. Tyto objekty jsou pak charakterizovány vlastnostmi, které popisují jejích chování a řídí jejich provoz. Některá data v objektech mohou být pouze ke čtení, některé zase umožnují zápis. BACnet definuje 23 různých standardních typů objektů, do kterých si pak jednotlivá zařízení můžeme přiřadit.

Každé zařízení může komunikovat s každým dalším objektem, nebo zařízením v síti. Není zde žádný centrální prvek sítě. Architektura systému je tedy síťová už jen z principu, že protokol může využívat jíž vybudovaných internetových sítí.

Protokol využívá nejen stávající internetové sítě, ale využívá i adresaci pomocí IP adres. Při připojení systému do internetu přes IP router, můžeme využít takzvaný

"IP tunneling"a projit přes internet 2 sítě mezi sebou. Tyto sítě se pak chovají jako jedna a můžeme propojit více budov mezi sebou.

BACnet může využívat i komunikaci přes sériovou sběrnici, ale to se nepoužívá, jelikož je zbytečně složitá adresace a navíc nelze využívat výhod TCP/IP protokolu a nelze systém připojit to internetu. Ztratíme tedy možnost kontrolovat systém zvenčí. Celkově je mnohem lepší využít Ethernet, protože v dnešní době se Ethernetová sít nachází téměř v každé budově.

Použití

BACnet je v dnešní době jeden z nejpoužívanějších protokolů pro automatizaci budov. Je tomu tak jelikož je normován a jeho nasazení je poměrně jednoduché, jelikož využívá již existujících internetových sítí. Umožňuje provozovat komunikaci současně i s jinými stávajícími sítěmi.

Při použití pro automatizaci budov, na kterou je primárně zaměřen, je použití IP komunikace Ethernet velmi vhodná. Při takovémto použití můžeme jednak realizovat komunikaci více budov mezi sebou, ale můžeme i kontrolovat a řídit budovy nebo celé systémy z jakéholiv přístupu na internet a to je v dnešní době jedno kritérií při výběru.

BACnet díky své rychlosti a možnosti komunikace "každý s každým"je vhodný pro použití do budov, kde chceme detekovat a hlásit požár, řídit osvětlení, topení, ventilaci, zabezpečení a hlídání budov, automatizovat chod budovy.

Pro tento protokol se používá BACnet Stack. Tato knihovna je pro jazyk C, ale podporuje mnoho dalších jazyků a kompilátorů.

2.2.4 M-BUS

M-BUS je evropský standard, který byl původně určený především pro vzdálené měření spotřeby. Postupně se rozšířil a v dnešní době se používá pro vzdálené měření dat a regulaci systémů. Při odečítání těchto hodnot nezáleží na rychlosti přenosu dat, ale hodně záleží na přesnosti těchto dat. Takže tento protokol neklade důraz na rychlost, ale důraz na kvalitu přenosu a zabezpečení proti rušení.

V dnešní době je tento standard využíván mnoha výrobci, tudíž podporuje možnost vzájemné komunikace zařízení různých výrobců.

M-BUS podporuje a zajišťuje spolehlivé propojení mnoha zařízení až na vzdálenosti několika kilometrů. Počet připojení zařízení na tyto vzdálenosti se udává řádově ve stovkách.

Pokud chceme použít bezdrátovou komunikaci, M-BUS poskytuje vlastní řešení tohoto problému. M-BUS vyvíjí vlastní komunikační standard pro bezdrátovou komunikaci M-BUS Wireless. Tento standard zatím není příliš rozšířen a probíhá prozatím vývoj.

Komunikace

Struktura sítě je sběrnicového typu, protože je to pro průmyslový sběr dat to nejefektivnější řešení. Využitím této sběrnice získáme jednoduché připojování a odpojování zařízení bez jakéhokoli narušení komunikace mezi zbytkem zařízení. Získáme možnost jednoduše vysílat data více zařízením najednou, ale je zde problém, že pokud vysílá jedno zařízení, nemůže komunikovat další. To má za následek snížení rychlosti systému, ale na rychlost M-BUS nehledí. Vzdálenost mezi jednotlivými zařízeními musí být maximálně 1km. Samotná komunikace probíhá způsobem Master-slave. Komunikace mezi zařízeními se provádí binárně. Probíhá zde sériová 8bitová asynchronní komunikace. Adresy stanic jsou primárně jednobytové. Je zde však možnost, při vyšším počtu zařízení, použít 8bytovou adresaci.

Samotná tato sběrnice je poměrně atypická. Využívá dvoudrátové spojení. Je atypická tím, že master vysílá data pomocí dvou hodnot napětí – 36 a 24 voltů. Při komunikaci k jednotlivým zařízením od Masteru znamená 36 voltů logickou 1 a 24 voltů logickou 0. Tímto způsobem je možnost i napájení zařízení ze sběrnice.

Při opačné komunikaci, tedy slave zařízení odpovídají na master dotazy, se neprovádí změna napětí, ale změna proudu. Slave zařízení odebírá 1,5mA při logické 1 a při logické 0 je tato hodnota o 11 až 20mA vyšší.

Použití

Při použití této sběrnice nedochází k narušení ostatních komunikací, jelikož nevyužívá stávajících sítí, ale je potřeba vytvořit dvoudrátovou sběrnici. Tato sběrnice je určena výhradně pro přenos dat z měřících zařízení do centrálního zařízení, které pak provede zásah do systému.

Vzhledem k přenosovým rychlostem není sběrnice M-BUS vhodná pro systém, kde je potřeba rychlého přenosu kritických zařízení. Není tedy vhodná na zabezpečení budov nebo přímo do energetiky, do automatického chodu budovy (například rozvoden).

Je však vhodná pro měření hodnot a díky zajištění bezpečnosti doručení dat a odolnosti vůči rušení, by se tato sběrnice mohla používat v energetice pouze pro měření spotřeby jednotlivých zařízení. Dále by mohl být určen pro měření dat v budově a následnou regulaci veličin. M-BUS je v podstatě univerzální řešení pomalého regulování veličin na velké vzdálenosti s určitou mírou zabezpečení. Pro regulování například teploty v budově je více než dostačující.

Knihovna libmbus slouží ke čtení dat z měřících přístrojů, podmínkou je podpora tohoto protokolu. Knihovna může číst data po RS232 nebo Ethernetu.

2.2.5 KNX

Při vytváření sběrnice KNX, neboli Konnex Bus, byla jako základ použita sběrnice EIB. Důvodem tohoto využití byl komerční úspěch EIB. Jelikož EIB je základ pro KNX, tak všechny přístroje, které vyhovují EIB, automaticky vyhovují i KNX. KNX má oproti EIB mnohem více funkcí, nabízí možnost připojení více zařízení a také podporuje využití různých přenosových médií. KNX je prakticky rozšíření EIB. Díky tomuto propojení EIB prakticky zanikla a často v literatuře KNX objevuje pod názvem KNX/EIB.

Stejně jako předešlé sběrnice i tato sběrnice podporuje komunikaci mezi mnoha přístroji od různých výrobců. Také podporuje napájení zařízení ze sběrnice.

Komunikace

Topologie je k překvapení libovolná, pouze s jedním omezením, že na sběrnici nesmí být uzavřená smyčka. Můžeme tedy použít téměř veškeré topologie, až na kruhovou a síťovou.

Typická sběrnice je stromová, která napodobuje samotou budovu. Jedna takzvaná linie (představme si jako místnost) může obsahovat až 256 zařízení. Takových linií může být na hlavní linii (patro budovy) připojeno až 15. Ty jsou následně připojeny na páteřní linii, kterou si už můžeme představit jako celou budovu, a zde je možnost zase připojení 15 hlavních linii. Pokud mezi sebou tyto čísla vynásobíme, získáme počet kolik je možných zařízení na jedné sběrnici, dostaneme až 57600 jednotlivých zařízení. Je zde však další omezení - jeden napájecí zdroj může napájet pouze 64 zařízení.

Poloha zařízení v síti je definována individuální adresou. Tato adresa je složena ze tří částí, které jsou mezi sebou odděleny tečkami. První část obsahuje číslo 0 až 15 a udává oblast (představili jsme si patro), přičemž 0 je vyhrazeno pro samotnou páteřní linii. Další část obsahuje opět 0 až 15 a určuje samotnou linii (místnost). Zde je 0 vyhrazena pro hlavní linii. Poslední část adresy určuje pořadí zařízení na dané linii a může nabývat hodnot 0 až 255. Tato individuální adresa má vyhrazeno 16 bitů. KNX umožňuje i vytváření skupin pro jednodušší komunikaci a ovládání několik zařízení najednou.

Samotná komunikace probíhá binárně a po sběrnici jsou přenášeny takzvané datové telegramy. Pro přenos dat může být využita kroucená dvojlinka, optické vedení nebo silové vedení. Nejčastěji se používá kroucená dvojlinka. Samotný signál je brán jako rozdíl napětí mezi kladným a záporným vodičem. Logická 0 je vyjádřena pulsem o velikosti -5V na každý vodič, tudíž napětí mezi vodiči je rovno 14V. Logická 1 je

vyjádřena jako nulový puls, tedy napěťový rozdíl je 24V. Časová jednotka těchto pulzů je také normalizovaná.

Použití

Jelikož KNX sběrnice může využívat obrovský počet jednotlivých zařízení, je vhodná pro průmyslové výrobní haly, tomuto nasvědčuje i vzdálenost mezi jednotlivými zařízeními, která se udává na 700metrů. Může dosahovat i vyšších přenosových rychlostí, záleží pouze na použitém médiu. Hodí se tedy i do zabezpečování chodu budov a celkové automatizaci budov. Díky své rychlosti a bezpečné komunikaci může být použit i do protipožárních systémů. Díky svým vlastnostem se hodí také například pro měření regulace, detekci požáru, plnou automatizaci budov a výrobní průmyslové haly.

Pro KNX protokol se najčastěji využívá funkcí knihovny xknx. Tato knihovna slouží pro jazyk Python. Knihovna poskytuje funkce pro komunikaci mezi dvěma zařízeními.

Pokud by se použil jako přenosové médium optický kabel, myslím si, že ani tak by se tato sběrnice nemohla použít v energetice. Vzhledem k tomu, že nejrychlejší přenos se udává jako 32kbps, nemyslím si, že by tato rychlost byla dostačující pro přenos kritické informace. Vycházím z toho, že pouze adresa zabírá 16 bitů.

2.2.6 CANopen

CANopen je standardizovaný komunikační protokol, který je nadstavba pro sběrnici CAN. Tento protokol byl navržen pro tuto sběrnici s tím, že má za úkol zjednodušovat návrh řídicího systému. Byl navržen pro rychlý přenos dat po sériové sběrnici s důrazem na rychlost a zaručení přenesení kritických dat. Cílem tohoto protokolu bylo výrazně zjednodušit návrh systému a vyhnout se problematice spojené se sběrnicí CAN, z tohoto důvodu tento protokol funguje pouze na této sběrnici. Hlavním z těchto problémů bylo správné časování zpráv, CANopen tento problém eliminuje standardizací komunikačních objektů a zavedením speciálních funkcí (například synchronizace).

Komunikace

Komunikace protokolu CANopen spočívá na principu master-slave. Princip komunikace je však vylepšen dvěma typy přenosu dat. Přenos procesních dat a přenos služebních dat. Každý tento přenos funguje trošku odlišně. Zatímco přenos procesních dat funguje jako producent a konzument, přenos služebních dat pracuje podle modelu klient a server. Při přenosu procesních dat dochází k přenosu kritických dat a tento přenos může každé zařízení vyslat bez požadavku na tyto data. To znamená, že producent může být například nějaký senzor a vyslat data, která nejsou vyžádaná, ale jsou kritická a následně je konzument přijme a vyhodnotí.

Při přenosu těchto kritických dat, je přijme pouze ten konzument, kterému data náleží. Tento přenos dat nevyužívá potvrzení a tyto data jsou vysílány cyklicky dokud nedojde ke změně v systému, která změní proces, který způsobil jejich vysílání. Při přenosu služebních dat je zde typický přenos stylem, že server vyšle dotaz na data a klient mu je poskytne. Klient sám od sebe data na sběrnici neposkytuje. Tento přenos už není cyklický a klient přestane vysílat data po přijetí potvrzení od serveru.

Data jsou vysílány v objektech. Samotné objekty obsahují adresu, typ objektu, data a parametr, zda je povoleno data měnit nebo pouze číst. Adresování v tomto protokolu je poměrně jednoduché. Každé zařízení na sběrnici má vlastní ID a všechny zařízení, které potřebují toto ID znát, ho mají uložené. Při komunikaci pak zařízení čistě zapíše do objektu ID zařízení, komu data náleží. Při přenosu kritických dat je zde pouze ID a data.

Použití

Hlavní výhodou při použití tohoto protokolu je rychlý přenos kritických dat. CANopen se dá použít pouze ve spojení se sběrnici CAN. Při použití toho spojení získáme odolný systém, který zaručuje rychlost a určitý stupeň zaručení doručení dat. Jeho nevýhodou je jeho jednoduchost. Každý, kdo tomuto protokolu rozumí, může velmi snadno data číst. Z toho vyplývá, že protokol není moc bezpečný a jeho použití se nedoporučuje do sítí, kde je určité bezpečnostní riziko. Hodí se tedy do výrobních průmyslových hal, kde je potřeba rychle data přenášet, ale nezaleží na stupni zabezpečení.

Kvaser CANopen stack je knihovna poskytující funkce pro master i slave zařízení. Použití této knihovny je nejrozšířenější. Poskytuje funkce pro komunikaci těchto zařízení v síti.

2.2.7 Profinet

Profinet je průmyslová sběrnice, která byla navržena jako nadstavba TCP/IP. Její hlavní zaměření je v oblasti řídicích systémů v průmyslové automatizaci. Tento protokol je otevřený a podléhá určité standardizaci. Patří mezi jeden z nejrozšířenějších protokolů v průmyslové automatizaci, který je založený na TCP/IP. Profinet umožňuje využití již existujících internetových tras bez jakéhokoli jejich narušení. Díky tomuto se hojně rozšiřuje, protože tímto krokem razantně snižuje pořizovací cenu. Díky základům v protokolu TCP/IP může být jako přenosové médium použit i bezdrátový přenos přes Wi-Fi nebo optický Ethernet. Jedná se prakticky o rozšíření výhod protokolu TCP/IP do průmyslové automatizace.

Komunikace

Topologie tohoto protokolu není předem stanovená, uživatel si může vytvořit jakoukoliv síť, stejně jak tomu je u internetu. Jak bylo jednou zmíněno, lze použít už stávající síť a podle potřeby ji třeba rozšířit. Profinet dokáže rozlišovat 3 typy komunikace. První je pro přenos standartních informací, u kterých není důležitý čas doručení, a Profinet se chová jako TCP/IP. Druhý je komunikace v reálném čase (RT), tento typ je určen pro časově závislé kritické informace.

Pro tento typ komunikace slouží vlastní komunikační kanál. Tento kanál je část standartního komunikační kanálu. Zařízení se schopností této komunikace mají snížené nároky na procesor, jedná se většinou o snímače nebo čidla. Třetí typ je komunikace v Izochronním reálném čase (IRT). Tento typ je určen pro aplikace, kde je potřeba velmi malé odezvy a přesnou synchronizaci. Toto je docíleno tím, že IRT má vlastní uzavřený komunikační kanál. Tímto stylem může být standardní komunikace, která obsahuje RT, existovat zároveň s IRT komunikací. Tyto dva komunikační kanály pracují nezávisle na sobě. Ať se použije jakákoliv komunikace, přenos dat probíhá stejně jako u TCP/IP ve tvaru HTML nebo XML. Adresace je prováděna pomocí IP adres zařízení, při nutnosti však lze využít MAC adres.

Použití

Použití protokolu Profinet se nabízí jako nejlepší řešení tam, kde již existuje vybudovaná komunikační síť. Toto se využívá ve velkém rozsahu, jelikož tímto způsobem lze ušetřit velký obnos peněz, který by byl vynaložen na vybudování komunikační sítě. Navíc díky použití stávajícího Ethernetu lze přistupovat i přes webové rozhraní. Díky tomuto přístupu lze dohlížet na chod a zároveň upravovat, nebo servisovat, systém v reálném čase z jakéhokoli místa s přístupem k webu. Díky těmto vlastnostem se hodí Profinet do velkých výrobních hal, kde je potřeba určitá data přenášet velmi rychle a s velmi malou odezvou (menší než 1ms). Dále se tento protokol hodí do výrobních hal, kde je potřeba určitý stupeň utajení dat, ale s možností kontroly dat z místa mimo halu.

Pro tento komunikační protokol jsem nenalezl žádnou opec source knihovnu. Existuje pouze knihovna od firmy Siemens, která podporuje čtení dat pomocí tohoto protokolu.

2.2.8 OPC UA

OPC UA je nová verze komunikačního protokolu pro průmyslovou automatizaci OPC. Na rozdíl od OPC tento protokol není funkční pouze pro operační systém Windows, ale je založen na již fungujících komunikačních protokolech TCP/IP a HTTP a jedná se o jejich nadstavbu. Tím je zaručeno, že protokol OPC UA může fungovat na jakémkoli zařízení, které nemusí obsahovat OS Windows a může být

zabudováno například do PLC automatů. Na rozdíl od OPC, které definuje přenos každého typu dat zvlášť, nové OPC UA definuje pouze formát přenosu dat a neřeší, o jaké data se jedná.

Komunikace

OPC UA podporuje dvojí typ komunikace. První podporuje binární komunikaci pomocí TCP/IP protokolu a druhá je přes web služby pomocí HTTP protokolu. Přenos samotných dat probíhá ve formě přenosu takzvaných zpráv mezi klienty a serverem. Jako síť může být použit již stávající Ethernet nebo Wi-Fi síť. Každé zařízení v síti má vlastní třídu, kterou je definováno. Tato třída obsahuje jméno zařízení, použití, datový typ a popis. Ve jméně je jméno samotného zařízení. V použití je zapsáno, zda má zařízení nějaká omezení. Popis definuje charakteristiku zařízení nebo samotného použití tohoto zařízení. K adresování je použito jméno zařízení. Přístup k datům je pomocí objektů.

Server vyšle požadavek, který obsahuje jméno zařízení, id objektu, metodu a zda chce číst nebo zapisovat. Přes jméno zařízení se dostane k několika blokům. Pomocí ID bloku vybere samotný blok, který obsahuje určitá data. Do bloku vyšle požadavek, zda chce data číst nebo jen zapisovat a metoda určuje, co se s daty bude dít. Každé zařízení v síti musí mít ke komunikaci aplikační certifikát, pomocí kterého se provádí zabezpečení systému. OPC AU definuje 4 úrovně zabezpečení dat komunikace – bez autentizace, serverová autentizace, klientská autentizace a oboustranná autentizace. Bez autentizace znamená, že všechny certifikáty jsou považovány za důvěryhodné a pokud jakékoli zařízení obsahuje certifikát, může být připojeno do systému a komunikovat. Serverová autentizace spočívá v tom, že server se připojí k jakémukoli klientu a ověření klienta proběhne přes jméno a heslo. Klientská autentizace dovoluje klientovi se připojit na jakýkoliv server, ale pouze pomocí administrátorských dat. Oboustranná znamená, že klient i server se může připojit pouze pokud obsahuje důvěryhodný certifikát.

Použití

Vzhledem k tomu, že protokol UPC UA využívá přenosu dat pomocí využití internetových protokolů, je přenos dat poměrně jednoduchý. To je způsobeno tím, že v dnešní době je internetová síť součástí téměř každé průmyslové budovy. UPC UA je tedy jednoduché implementovat do jakékoli budovy. V největší míře se používá pro systémy částečné automatizace. Pro systémy, které kontrolují energie v budově, řízení teplot, ale také hlavně u systémů, které potřebují rychlý přenos kritických dat a u systémů, které potřebují ke svému chodu určitý pojem o čase (například počítání cyklů, nebo určit čas, kdy se jaká událost stala).

Pro uživatelské použití tohoto protokolu se nejčastěji využívá knihovna OuickOPC. Tato knihovna poskytuje funkce clienta. Je možné ji implementovat do mnoha programovacích jakzyků (například C, C++, C#, Python a další).

2.2.9 Další méně rozšířené protokoly

DNP3

Jedná se komunikační protokol, který se využívá výhradně v Americe. Je používán jako protokol ke komunikaci mezi dvěma zařízeními v průmyslové automatizaci. Jedná se o poměrně dobře zabezpečenou komunikaci, která zaručuje určitý stupeň doručení zprávy. Protokol podporuje i zabezpečení proti odposlouchávání dat. Tento protokol se v Evropských zemí téměř nevyužívá.

LonTalk

LonTalk je komunikační protokol, který se používá pro přenos dat po sběrnici LON. Tento protokol využívá přenos dat ve zprávách a přenos je realizován po sběrnici, která podporuje velké množství přenosových médií. Tento protokol využívají pouze 3 světové firmy (Echelon, Motorola a Toshiba).

Ego-n

Ego-n je komunikační protokol pro inteligentní elektroinstalaci. Používá se jak pro průmyslové, tak pro domácí využití. Přenos dat je opět přes sběrnici a je zde jeden modul, který řídí celý systém. Tento protokol je využit pouze firmou ABB.

Synco living

Synco living je systém inteligentních komunikační prvků, které dohromady tvoří inteligentní domácnosti a budovy. Jedná se o plnou automatizaci. Tento systém vychází ze sběrnice KNX. Pouze tuto sběrnici vylepšuje, ale funguje téměř stejně. Tento systém využívá a nabízí pouze firma Siemens.

Elko EP

Elko EP je česká obdoba systému Synco living. Tento produkt nabízí firma Elko EP. Oproti Synco living podporuje ELKO EP automatizacepro neprůmyslové použití. Podporuje pouze automatizaci malých domů, maximálně hotelů.

3 Výběr protokolů

Po hotové rešerši protokolů přišlo na řadu vybrat z těchto protokolů tři až čtyři pro ukázku jejich funkcí. Při vybírání protokolů jsem vzal v potaz nejpoužívanější knihovny, které jsem zmiňoval v rešerši. Pokud však knihovna nebyla veřejně přístupná, hledal jsem jinou. Jako programovací jazyk jsem si, v závislosti na konzultaci s vedoucím práce, zvolil C#. Pokud tedy nejpoužívanější knihovna daného protokolu nebyla .Net hledal jsem .Net knihovnu pro daný protokol. Při hledání .Net knihoven jsem hledal knihovny nedávno a často aktualizované. To bylo jedno z hlavním kritérií pro výběr knihovny. Knihovny, které pro přenos dat vyžadují sběrnici, jsem z výběru odebral. Dále bylo potřeba, aby daný protokol měl knihovny jak pro server, tak pro clienta, případně pro obojí v jedné.

Poté, když už bylo nalezeno několik knihoven pro jednotlivé protokoly, přišlo na řadu vybrat, pro jaké se budou programovat aplikace. Po konzultaci s vedoucím jsme vybrali čtyři protokoly, které mají odlišný způsob přenosu dat. Bylo by zbytečné použít dva sobě velice podobné protokoly, jejichž přenos dat je téměř totožný.
Pro tuto práci tedy byla vybrána tato čtveřice protokolů:

- DLMS/COSEM
- IEC 60870-5-104
- MQTT
- OPC UA

Pro každý z protokolů jsem si pro jednoduší implementaci knihoven zvolil pouze knihovny podporující funkce pro obě zařízení v síti.

Pro MQTT jsem nepoužil knihovnu z rešerše (viz strana 19), protože tato knihovna byla pouze pro vývojové prostředí Eclipse. Pro DLMS jsem knihovnu z rešerše využil, jelikož obsahovala část .Net. Pro IEC 60870-5-104 jsem využil knihovnu z rešerše. Pro OPC UA jsem knihovnu z rešerše nevyužil, jelikož obsahovala funkce pouze pro clienta.

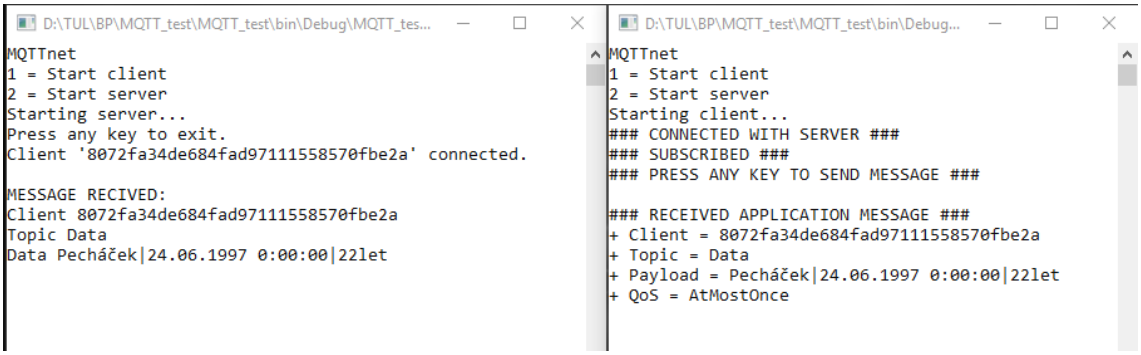
4 Jednotlivé aplikace

Pro programování jednotlivých aplikací jsem si zvolil vývojové prostředí Microsoft Visual Studio. Toto prostředí jsem si zvolil, protože se v tomto prostředí pohybuji již od základní školy. Na vysoké škole jsme v něm programovali a také nám ho zdarma poskytuje škola ke studijním účelům.

4.1 MQTT

Jako první protokol pro ukázku aplikace jsem zvolil MQTT. Pro tento protokol existuje velké množství knihoven, obsahujících funkce pouze pro clienta nebo server. Knihovny MQTTnet [52] a M2Mqtt [54], které obsahují funkce pro clienta i server, jsem vyzkoušel. M2Mqtt nemohla najít cestu ke knihovně, přestože byla ručně nastavena, zvolil jsem tedy programovat a ukazovat funkce v knihovně MQTTnet.

Knihovna podporovala balíček NuGet. Po nainstalování a vložení do projektu bylo velice snadné využívat jeho funkce. Následně jsem hledal jak využívat danou knihovnu. Bohužel, veškeré návody byly pro jiné programovací jazyky. Knihovnu jsem se tedy učil používat pomocí příkladů, které složka s knihovnou obsahovala. Nejprve jsem se musel naučit, jak vlastní kód funguje a co dělá. Poté jsem začal psát vlastní části kódu a komunikaci protokolu začal zprovozňovat. Po propojení serveru a clienta, jsem mezi nimi začal posílat předem určená data.

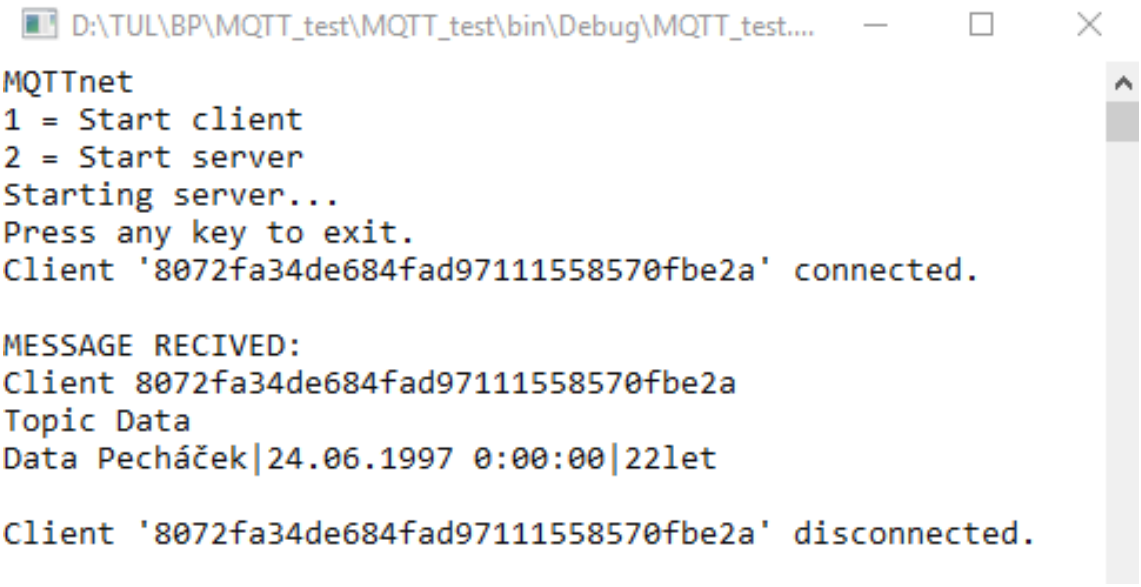


Obrázek 4.1: MQTT - poslání zprávy a připojení

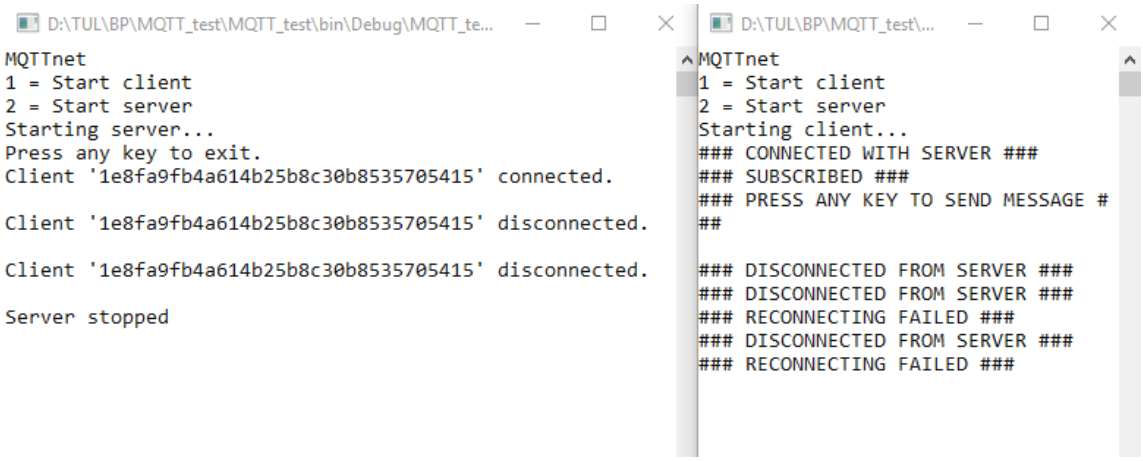
Na obrázku 4.1 lze vidět, jak samotná aplikace vypadá. Nejprve určíme, zda spustí server nebo clienta. Client je nastaven tak, aby hledal server na adrese 127.0.0.1, tedy localhost. Pokud na této adrese server nenajde, vypíše chybovou hlášku a začne s opětovným připojováním. Pokud je však server zapnut, client ho najde, oba vypíší, že připojení proběhlo, a client může začít vysílat data. Po zmáčknutí libovolné klávesy se odešlou specifická, předem určená data. V aplikaci lze vidět, že u protokolu MQTT nefunguje klasické posílání dat, ale takzvaný "publish data". To znamená, že dané zařízení vyšle data všem zařízením v sítí. Je to patrné i z toho, že client, který data vyslal, tyto data také přijal.

Data se posílají v předem daném tvaru. Obsahují ID zařízení, které data vyslalo, téma dat, samotná data a v tomto případě i kvalitu přenosu. Datům lze v této knihovně nastavit ještě "retain flag", což znamená, že je server uschová a všem clientům, kteří se připojí, je bude poskytovat. Server tyto "retain" data zapisuje do seznamu.

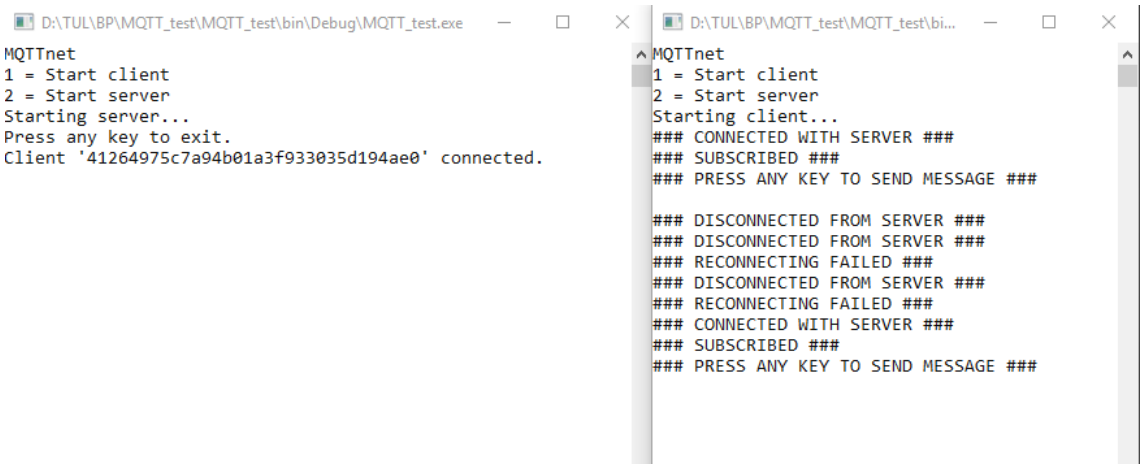
Po odpojení clienta server vypíše hlášku o odpojení (viz Obrázek 4.2). Pokud je client připojen a server se vypne, začne opětovné připojování (viz Obrázek 4.3) a v případě, že se server opět zapne, client se automaticky připojí. Poté může znovu probíhat komunikace (viz Obrázek 4.4).



Obrázek 4.2: MQTT - odpojení clienta



Obrázek 4.3: MQTT- recconecting



Obrázek 4.4: MQTT - reconnected

V ukázkách lze vidět výhody protokolu MQTT. První výhodou je zasílání dat bez ohledu na jejich obsah nebo tvar. Veškerá data se posílají ve tvaru zprávy, do které lze zapsat jakýkoliv datový typ. Mnou vytvořená aplikace posílá datum narození jako typ DateTime (viz Obrázek 4.5).

```
var applicationMessage = new MqttApplicationMessageBuilder()
    .WithTopic("Data")
    .WithPayload("Pecháček|"+ DateTime.Parse("24.6.1997")+"|22let")
    .WithExactlyOnceQoS()
    .Build();
```

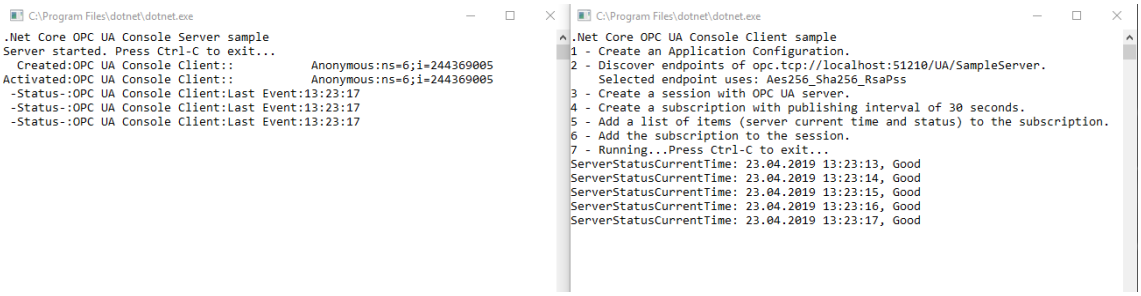
Obrázek 4.5: MQTT - tvar dat

Další výhodou je takzvaný "data pushing". To znamená, že client vyšle data směrem k serveru bez jeho iniciativy. Nevýhodou je "data publish". Jakmile kdokoliv v síti data pošle, může je kdokoliv přečíst.

Zvolení MQTTnet bylo výhodné, protože server i client využívá stejnou knihovnu, avšak tato knihovna nepodporuje funkci odpovědi serveru na přijatá data. Toto by se dalo vyřešit použitím alternativní knihovny, která se zaměřuje čistě na server.

4.2 OPC UA

Další protokol pro zprovoznění ukázky byl OPC UA. Hledání vyhovujících knihoven pro tento protokol už nebylo tak snadné, jako pro protokol MQTT. Protokol OPC UA není rozšířen do takové míry. Knihovna OPCFoundation.NetStandard.OPC.UA [59] podporovala funkce pro clienta i pro server. Zvolil jsem tedy tuto knihovnu a snažil se ji zprovoznit. Dokumentaci a jednoduché ukázky, které knihovna obsahovala, jsem využil k pochopení této knihovny. Dokumentace bohužel obsahovala pouze informace k instalaci samotné knihovny a spuštění ukázek. Musel jsem tedy naprogramovaným ukázkám porozumět sám. Jakmile jsem kód pochopil, upravil a nastavil jsem ho tak, aby server i client komunikovali na jednom počítači. Ukázka však obsahovala pouze dotaz clienta na stav připojení k serveru a datum serveru. Nejdříve bylo zapotřebí synchronizovat čas, protože client správně zobrazoval čas serveru, tedy koordinovaný světový čas, avšak server zobrazoval čas počítače.

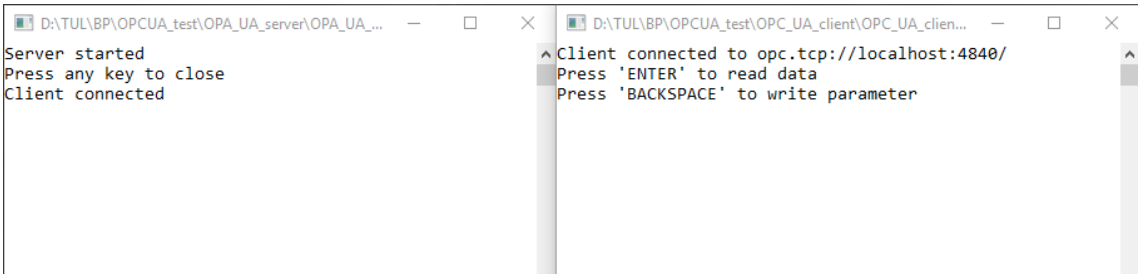


Obrázek 4.6: OPC UA - ukázka nepoužité knihovny

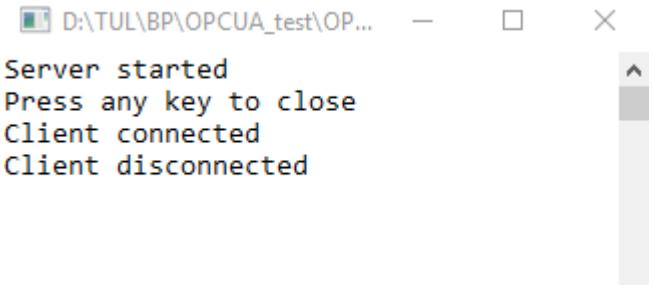
Následovalo zprovoznění posílání dat. Vytvoření komunikace v této knihovně bylo velice složité a bohužel jsem nedohledal žádnou dokumentaci, která by popisovala jaký tvar mají data mít, ani žádnou zmínku o této funkci. Metodou "pokus omyl"jsem se snažil data poslat, ale k zavolání této funkce bylo zapotřebí vyplnit velké množství parametrů, u kterých jsem nevěděl jakou mají nabývat hodnotu, nebo jaký mají mít typ.

Rozhodl jsme se tak zvolit jinou knihovnu, a to Opc.UaFc.Advadced [56]. Tato knihovna také podporovala funkce jak pro server, tak pro clienta. Neobsahovala však

již jednotlivé ukázky, protože jsem k ní našel pouze NuGet balíček, nikoli složku ke stažení. Našel jsem však stránku [58] s částmi kódu k využití této knihovny a také PDF [10] soubor s jejími funkcemi, pomocí nichž jsem začal psát vlastní program. Snažil jsem se vytvořit server na vlastním počítači. Jeho vytvoření bylo poměrně jednoduché. Jako první krok bylo potřeba vytvořit samotné vnitřní bloky tohoto serveru, takzvané "Nodes". Tyto bloky slouží ke čtení clientem nebo k samotnému zápisu dat. Následně jsem vytvořil server s těmito bloky, přičemž bylo třeba určit na jaké adrese se server spustí. Zadal jsem opět localhost. Spouštění pak již bylo jednoduché. Pokud je server spuštěn, detekuje vytvoření a zrušení spojení s clientem.

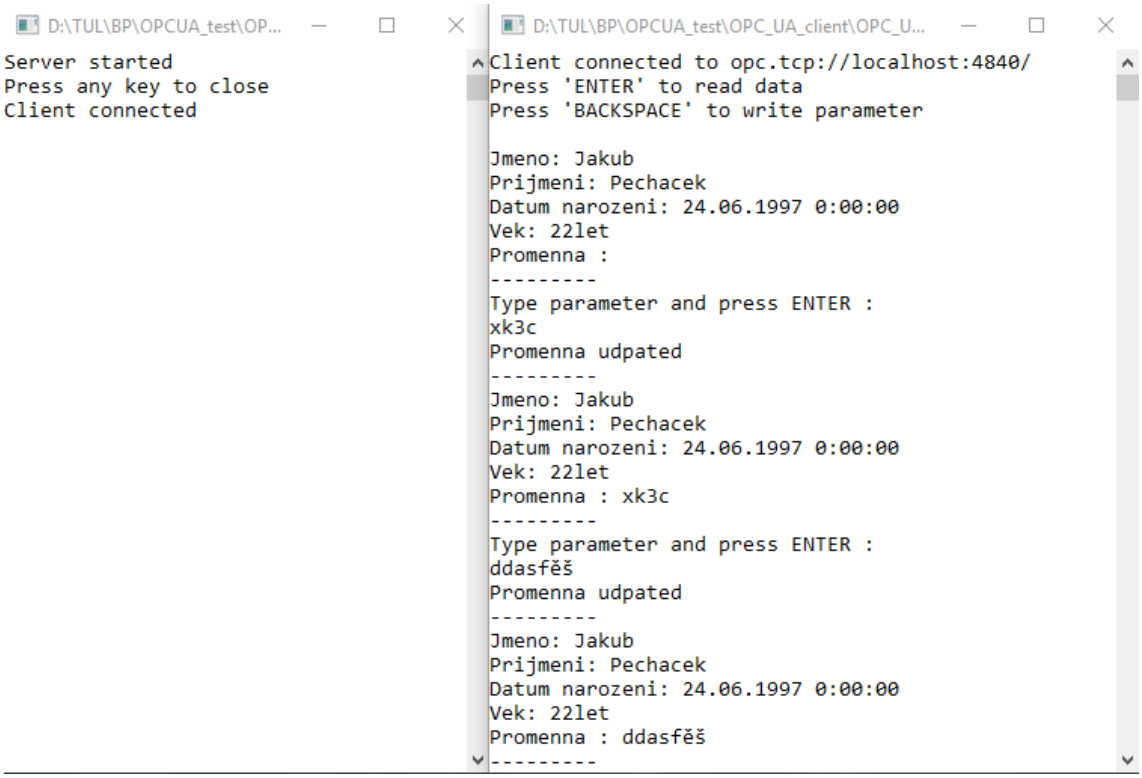


Obrázek 4.7: OPC UA - připojení



Obrázek 4.8: OPC UA - odpojení

Na již zmiňované stránce [58] jsem našel podobu clienta a upravil ji pro své podmínky. Pokud client nenajde server na dané adrese, vypíše chybovou hlášku. Prvně jsem pracoval pouze se čtením dat ze serveru a po jejich úspěšném čtení jsem hledal funkce, jak data zapisovat. Knihovna umí měnit data bloků vytvořených na serveru. Bohužel neumí bloky vytvářet. Vytvořil jsem tedy jeden blok navíc, do kterého data zapisuji, a při dalším čtení jsou tato data změněna. Client se umí rozhodovat, zda chce data číst, nebo zapisovat.



Obrázek 4.9: OPC UA - funkce

Mnou vytvořená aplikace simuluje jednoduchost čtení a zápisu dat v protokolu OPC UA. Bohužel zde není klasická funkce posílání dat. Umožňuje pouze zápis a čtení bloků na daném serveru. Bloky na serveru však mohou mít jakýkoliv typ, který jim předem nadefinujeme. V mém případě datum narození opět typ DateTime.

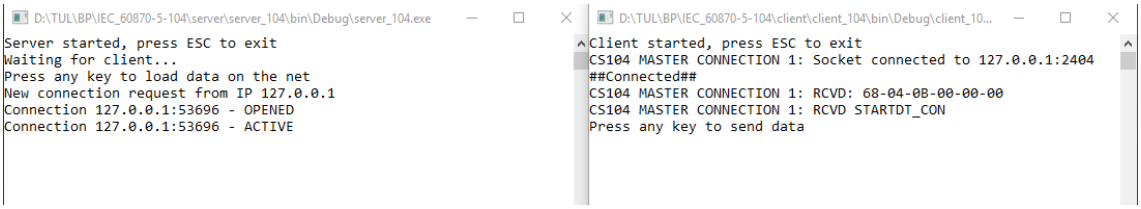
```
var testNode1 = new OpcDataVariableNode<string>("jmeno", "Jakub");
var testNode2= new OpcDataVariableNode<string>("prijmeni", "Pechacek");
var testNode3 = new OpcDataVariableNode<DateTime>("datum", DateTime.Parse("24.6.1997"));
var testNode4 = new OpcDataVariableNode<int >("let", 22);
var testNode5 = new OpcDataVariableNode<string>("value", "");
```

Obrázek 4.10: OPC UA - tvar bloků serveru

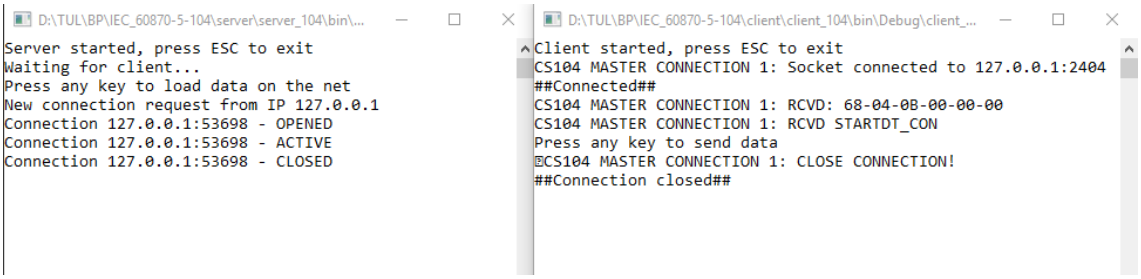
4.3 IEC 60870-5-104

Třetím programovaným protokolem byl IEC 60870-5-104. Pro tento protokol jsem našel knihovnu lib60870-5 [63]. Tato knihovna podporovala .NET, ale byla ve spojení s protokolem IEC 60870-5-101. To však ničemu nevadilo, protože, jak jsem již zmiňoval v teoretické části, IEC 60870-5-104 je pouze rozšíření IEC 60870-5-101. Dokumentace k této knihovně nebyla bohužel žádná, protože na stránkách této

knihovny je ještě knihovna pro IEC 61850 a veškerá dokumentace je jí přizpůsobena. Byl jsem tedy nucen se vše naučit sám jen z příkladů, které knihovna poskytuje. U příkladů na server a clienta jsem se snažil porozumět kódu a poté opět oživit server. Používal jsem části kódů z ukázek knihovny a pomocí kopírování funkcí se snažil server spustit. Nastavení a spuštění serveru bylo poměrně jednoduché přes pár příkazů. Následovalo nastavení potvrzení připojení. Povolil jsem pouze jednu adresu, která se mohla na server připojit – localhost. Po naprogramování žádosti clienta o připojení jsem vytvořil události s vypisováním připojení pro server i clienta.



Obrázek 4.11: IEC 60870-5-104 - připojení



Obrázek 4.12: IEC 60870-5-104 - odpojení

Po jejich propojení jsem se snažil posílat data. Prvně bylo nutné zjistit, jak se data posílají. Děje se tak přes takzvané "asdu". Tyto "asdu"obsahují knihovnou předem předdefinované bloky. Seznam těchto bloků jsem našel v souborech samotné knihovny. Obsahuje bloky především na posílání naměřených dat. Posílal jsem mezi sebou nejdříve čistě číselná data a snažil se je přijmout na server. Na serveru jsem vytvořil událost na přijetí "asdu". Podobnou událost jsem vytvořil i pro clienta, protože server umí data posílat také. Jakmile mi fungovalo posílání a přijímaní číselných dat z obou stran, snažil jsem se poslat specifická data, která jsem posílal i v předešlých aplikacích. Byl jsem tedy nucen najít blok na posílání textu, který však v seznamu chyběl. Využil jsem blok, který posílal sekvenci bitů a s jehož pomocí bych mohl data posílat binárně. Ve chvíli kdy jsem se snažil převést typ String na typ UInt32, ve kterém tento blok data posílá, narazil jsem na problém. Vývojové prostředí zobrazovalo chybu - špatný vstupní řetězec. Zkusil jsem tedy posílat data po jednom písmenku.

```
ASDU msg = new ASDU(con.GetApplicationLayerParameters(), CauseOfTransmission.INTERROGATED_BY_STATION, false, false, 2, 1, false);


msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('P'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('E'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('C'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('H'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('A'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('C'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('E'), new QualityDescriptor()));
msg.AddInformationObject(new Bitstring32(1, Convert.ToUInt32('K'), new QualityDescriptor()));


con.SendASDU(msg);
ASDU msg2 = new ASDU(con.GetApplicationLayerParameters(), CauseOfTransmission.INTERROGATED_BY_STATION, false, false, 2, 1, false);


msg2.AddInformationObject(new MeasuredValueScaled(2, 22, new QualityDescriptor()));

con.SendASDU(msg2);
```

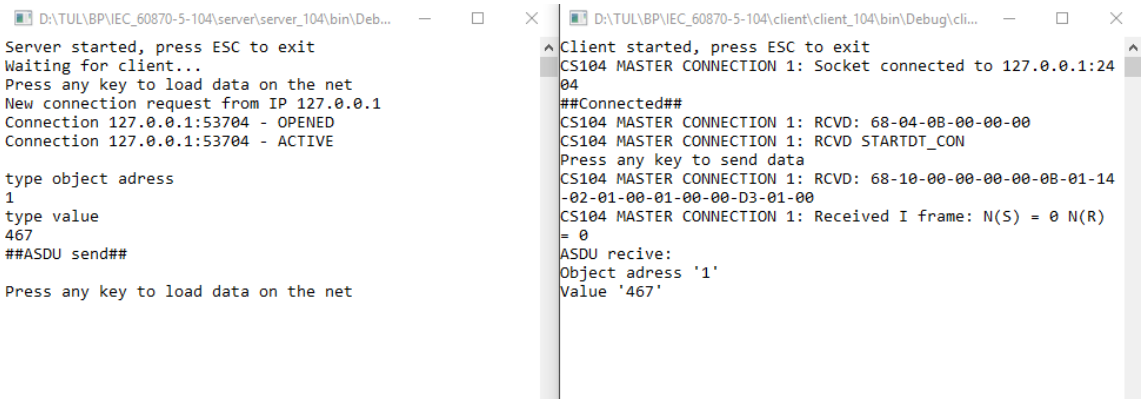
Obrázek 4.13: IEC 60870-5-104 - tvar posílaných dat

Tento pokus byl již úspěšný. Posílal jsem nejdříve "asdu", která obsahovala několik bloků. Každý tento blok obsahoval jedno písmeno. Jakmile jsem chtěl přidat další zprávu s číselnými údaji, nastala chyba s čtením dat. Při přijetí dat bylo nutné nejprve rozpoznat jaké bloky daná zpráva obsahuje a následně je odlišně zpracovávat.

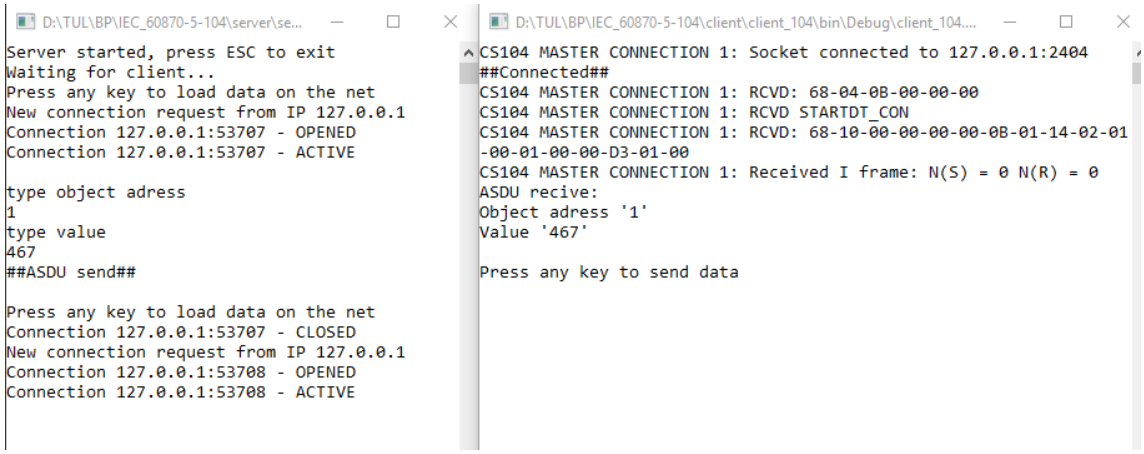
```
D:\TUL\BP\IEC_60870-5-104\server\...
Server started, press ESC to exit
Waiting for client...
Press any key to load data on the net
New connection request from IP 127.0.0.1
Connection 127.0.0.1:53699 - OPENED
Connection 127.0.0.1:53699 - ACTIVE
ASDU recive:
Object adress '1'
Jmeno 'PECHACEK'

ASDU recive:
Object adress '2'
Let '22'
```

```
D:\TUL\BP\IEC_60870-5-104\client\client_104\bin\Debug\client_1...
Client started, press ESC to exit
CS104 MASTER CONNECTION 1: Socket connected to 127.0.0.1:2404
##Connected##
CS104 MASTER CONNECTION 1: RCVD: 68-04-0B-00-00-00
CS104 MASTER CONNECTION 1: RCVD STARTDT_CON
Press any key to send data

CS104 MASTER CONNECTION 1: ------k-buffer------
CS104 MASTER CONNECTION 1: 0 : S 1 : time 1556019781557
CS104 MASTER CONNECTION 1: --------------------
CS104 MASTER CONNECTION 1: ------k-buffer------
CS104 MASTER CONNECTION 1: 0 : S 1 : time 1556019781557
CS104 MASTER CONNECTION 1: 1 : S 2 : time 1556019781558
CS104 MASTER CONNECTION 1: --------------------
##ASDU send##

Press any key to send data
```

Obrázek 4.14: IEC 60870-5-104 - tvar posílaných dat



Obrázek 4.15: IEC 60870-5-104 - tvar posílaných dat



Obrázek 4.16: IEC 60870-5-104 - tvar posílaných dat

Na obrázku 4.14 je znázorněna komunikace, kdy client posílá zprávy na server. Na obrázcích 4.15 a 4.16 je vidět druhý směr komunikace. Zde se zasílání dat liší. Protože client využívá funkce poslat, data se pošlou jen jednou. Server ale používá funkci zvanou zařadit do fronty, neboli enqueue. Server data pošle na síť a clienti je přijmou. Výhodou je, že pokud se client připojí na server, kde tyto data už nahrány jsou, automaticky je přijme hned po připojení na server. Na obrázcích je ukázka, kdy client data přijal, následně se vypnul a znovu zapnul a data opět přijal ihned po připojení.

V aplikaci mi u clienta bohužel vypisuje nějaké události i samotná knihovna. Tyto události jsem se snažil v knihovně zakomentovat, ale neúspěšně. Události jsem sice zakomentoval, ale když jsem celý projekt i se samotnou knihovnou chtěl přeložit, přeložila se pouze má část projektu. Knihovna je nějakým způsobem zamčena proti překládání a její přeložení bylo přeskočeno, takže moje úpravy se v projektu neprojevily.

4.4 DLMS/COSEM

Poslední přišel na řadu protokol DLMS/COSEM. Při hledání .Net knihoven pro tento protokol jsem měl nejmenší štěstí. Nalezl jsem pouze jedinou knihovnu, a to Gurux.DLMS.Net [40]. Samotná knihovna Gurux je poměrně rozšířená a dokumentace k ní je také dost, avšak .Net část této knihovny už tak rozšířená není. Dokumentace k ní je omezená na pár souborů. Byl jsem tedy odsouzen pouze k samotným příkladům, které tato knihovna obsahovala.

Nejprve jsem se snažil zprovoznit server. Ten pravděpodobně fungoval, server vypisoval čas od poslední události. Následovala ukázka clienta. Nejdříve jsem se snažil porozumět kódu, jelikož po jeho spuštění se automaticky aplikace okamžitě vypnula. Zjistil jsem, že client pracuje s funkcí, která čte parametry z příkazového řádku při spuštění. Snažil jsem se na internetu zjistit, jaký mají mít tyto parametry tvar, bohužel neúspěšně. Zadání parametrů se po několika neúspěšných pokusech podařilo. Zadal jsem do nich IP adresu localhost a port, na kterém server běžel. Aplikace se přestala vypínat, nastal však druhý problém a aplikace začala zobrazovat další chybu. Aplikace se nyní zastavovala na řádku, kde se vytvářel reader pro čtení dat z měřáků. Dle mého názoru pracuje ukázka knihovny s měřáky, které podporují tento protokol. Tyto měřáky jsem neměl k dispozici, z toho důvodu jsem to nemohl otestovat.

Snažil jsem se zprovoznit vlastní program, kde použiji části dokumentace k serveru a clinetovi. Server se choval jako v samotné ukázce v knihovně. Při programování clienta jsem narazil na problém. Našel jsem zde funkci write a na serveru jsem vytvořil blok, do kterého by se data mohli zapisovat. Při snaze o použití této funkce jsem zjistil, že její parametry se odkazují na další vnitřní bloky této knihovny. Snažil jsem dostat přes samotné odkazy co nejdále, ale bohužel jsem nenašel jaký tvar by data měly obsahovat. Jeden parametr odkazuje na index bloku na serveru, tento index jsem bloku na serveru nastavil při jeho vytváření. Po vytvoření dat se třídou tohoto objektu se mi nepodařilo nahrát jejich hodnoty. Třída dat mě bohužel odkazovala na další objekty knihovny. I když se mi podařilo samotným datům nastavit typ, nepřišel jsem na způsob, jak těmto datům přiřadit hodnotu. Přes klasické přiřazování hodnot mi vývojové prostředí zobrazovalo chybu a funkci jsem na to nenašel.

Knihovnu Gurux.DLMS.Net se mi tedy přes veškeré snahy nepodařilo zprovoznit. Pokusil jsem se zprovoznit vlastní program, ke kterému jsem čerpal z dokumentace k ukázkám knihovny, i samotné ukázky knihovny. V obou případech jsem byl bohužel neúspěšný.

5 Shrnutí

Při programování protokolu MQTT nastal pouze jeden problém s první knihovnou, která nefungovala, protože údajně nemohla najít cestu na danou knihovnu. Tento problém jsem vyřešil využitím jiné knihovny, která podporovala balíček NuGet, díky kterému bylo použití druhé knihovny značně jednodušší. Posílání dat pomocí této knihovny bylo velice jednoduché. V první řadě bylo potřeba vytvořit samotnou zprávu. Tvar této zprávy je předem stanovený. Je ale možné v něm měnit kvalitu přenosu dat a zachování zprávy na serveru. Zpráva obsahuje téma, samotná data, kvalitu přenosu a následně je možné přidat udržovací bit. Do samotných dat zprávy lze zapsat jakýkoliv datový typ. V ukázce zapisuji string a DateTime. Na omezení velikosti samotné zprávy jsem nenarazil. V aplikaci je i ukázka největší výhody tohoto protokolu, a to "data pushing". To znamená, že client pošle data na server bez jakékoliv iniciativy daného serveru. Server data bez problémů zpracuje.

V protokolu OPC UA je server takzvané "hloupé"zařízení. Server má jen vytvořené bloky, které client čte, nebo upravuje. Sám server posílat data neumí. Client má velice jednoduchý přístup k těmto datům. Jaké data chce číst, nebo zapisovat, určuje pouze pomocí názvu bloku na serveru. Tento název byl definován při vytváření bloku. Tato knihovna nepodporuje vytváření bloků clientem, může pracovat pouze s již vytvořenými. Tyto bloky mohou mít jakýkoliv typ dat. V mé aplikaci ukazuji string, int i DateTime. Protokol OPC UA nepodporuje klasický tvar posílání dat. Client se pouze ptá na bloky na serveru a server mu je poskytuje, nebo client upravuje jejich hodnoty.

Server v protokolu IEC 60870-5-104 není jen zařízení pro poskytování dat. Umí také posílat data, ale od posílání dat clientem se to liší. Když server pošle data, nahraje si je do fronty a každému clientovi, který se připojí do sítě, tyto data poskytne. Když client pošle data na server, je to klasické posílání dat. Tato knihovna bohužel také neumí odpovídat na přijatá data. Samotný tvar posílání dat musí splňovat předdefinované bloky. Bylo tedy nutné na serveru rozpoznávat jaký typ dat byl přijat a následně je odlišně zpracovávat, jelikož každý blok může obsahovat jiná data. Pro posílání stejných dat jako u předešlých dvou protokolů bylo potřeba vytvořit více zpráv. Jedna obsahovala binární bloky, do kterých jsem nahrál jednotlivé znaky písemné zprávy, druhá obsahovala číselné údaje. Posílání číselných údajů bylo jednodušší. Přijetí písemné zprávy bylo těžší, jelikož každý blok zprávy musel být zpracován a následně vypsán zvlášť, aby opět tvořili slovo.

Knihovna k protokolu DLMS měla rozsáhlou dokumentaci, která však neobsahuje část pro .Net. Snažil jsem se tedy protokol zprovoznit pomocí příkladů, které knihovna obsahovala. Bohužel bez úspěchu. Server se spustil. Nejdříve se mi u clienta nedařilo nastavit ip adresu a port serveru, protože si tyto parametry ukázky načítaly z příkazového řádku a dokumentace neuváděla, v jakém tvaru by parametry měly být. Po několika pokusech jsem tyto parametry nastavil, avšak client se k serveru nepřipojil. Příklady obsahovali readme.txt, kde bylo popsáno vytvoření serveru i clienta. Pomocí těchto souborů jsem se snažil programovat. Server se opět spustil, client se vytvořil, ale posílání dat bylo zmatenější než u první knihovny v protokolu OPC UA. Dokumentace k posílání dat nebyla žádná, takže se mi funkce nepovedla zprovoznit. Z tohoto důvodu se mi nepodařilo DLMS/COSEM zprovoznit a nemohu jej tak porovnat s předešlými.

Tyto čtyři protokoly byly vybrány, protože každý z nich posílá data jiným způsobem. Jejich programování bylo naprosto odlišné. Mají odlišný tvar dat i jiný styl posílání dat. Jeden má "data publishing", druhý data čte ze serveru a třetí posílá data klasicky. Nejjednodušší posílání dat bylo u MQTT, jelikož nebylo potřeba definovat typ těchto dat. Velice jednoduchá komunikace byla také u OPC UA. Protokol IEC 60870-5-104 měl posílání dat nejsložitější. Bylo potřeba definovat typ pomocí předdefinovaných bloků a bylo složité data zpracovávat, protože každý typ bylo potřeba zpracovat jinak. Připojení clienta na server měl nejjednodušší OPC UA. Stačilo vytvořit clienta s adresou serveru a připojit se. MQTT měl připojení složitější, protože client se musel vytvořit, následně ho bylo potřeba nastavit a pak teprve připojit. Nejsložitější připojování měl IEC 60870-5-104. Zde bylo potřeba vytvořit clienta s adresou, následně vyslat požadavek o připojení a teprve po jeho potvrzení se připojit.

6 Závěr

Výsledkem mé práce je rešerše komunikačních protokolů pro chytré sítě a chytré budovy v českém jazyce a trojice jednoduchých aplikací, které znázorňují použití protokolů a jejich komunikaci. Tyto aplikace se dají použít v praxi jako základ pro programování tohoto protokolu.

Zvolením knihovny MQTTnet jsem bohužel přišel o funkci odpovědi. Tato knihovna nepodporuje funkce, kdy server po přijetí dat na tyto data odpoví. Podporuje pouze stejnou funkci jako u clienta a to "data publish". Abychom mohli odpovídat na přijatá data, museli bychom použít alternativní knihovnu, která tyto funkce podporuje. Velká část alternativních knihoven podporuje funkce jen pro clienta nebo pro server. Volbou využít jednu knihovnu pro obě zařízení jsem přišel o nějaké funkce tohoto protokolu.

Při programování OPC UA jsem nejprve zprovoznil první knihovnu. Tato knihovna však měla velice složité posílání dat a dokumentace k tomu jsem nedohledal. Nalezl jsem tedy více uživatelsky přívětivou knihovnu, která opět podporovala funkce jak pro server, tak pro clienta.

Programování IEC 60870-5-104 bylo složitější než předešlé dva protokoly. Problémem bylo, že vývojáři vyvíjí knihovnu i pro IEC 61850 a veškerou dokumentaci přizpůsobují této knihovně. Výhodou však byla podpora funkce pro server i clienta.

Protokol DLMS se mi bohužel nepodařilo zprovoznit, protože ukázky, které byly součástí knihovny, pravděpodobně pracovaly se zařízeními podporující tento protokol a ty jsem k dispozici neměl. Knihovna zároveň neobsahovala dostatečnou dokumentaci pro její .Net část. Uvnitř samotného kódu knihovny jsem nedohledal paremetry funkce pro posílání dat. Z těchto důvodů jsem nebyl úspěšný ani při realizaci vlastního programu.

Z mého pohledu jsou protokoly MQTT a OPC UA jednoduché pro použití. Jejich knihovny jsou rozšířené. Po pochopení knihoven je jejich využívání velice jednoduché. Protokol IEC 60870-5-104 je na použití složitější, jelikož knihovna není tak jednoduchá a dokumentace k ní je téměř nulová. Dokonce i po pochopení této knihovny je složitější práce s ní, než u předešlých dvou protokolů. Protokol DLMS je nejsložitější. Jeho knihovna je velice složitá a dokumentace k ní je téměř nulová.

V práci jsem splnil všechny body zadání.

Literatura

[1] JIRKA, Matěj. *Framework DLMS/COSEM pro sběr dat v AMM systémech.* Praha, 2016. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Doc. Ing. Jiří Novák Ph.D.

[2] KOZÁK, Jiří. *Instalační sběrnice v inteligetních budovách.* Brno, 2011. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Tomáš Marada, Ph.D.

[3] ÚLEHA, Tomáš. *Komunikační prostředky inteligentních domů.* Zlín, 2013. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně. Vedoucí práce Doc. Ing. Fratišek Hruška, Ph.D.

[4] HORACKOVA, Maria. *Linkové a síťové protokoly IoT.* Praha, 2018. Bakalářská práce. Vysoká škola ekonomická v Praze. Vedoucí práce Ing. Felix Espinoza.

[5] AUSBERGER, Tomáš. *OPC UA servery a jejich použití pro přenos dat řídicích systémů.* Plzeň, 2015. Bakalářská práce. Západočeská univerzita v Plzni. Vedoucí práce Ing. Pavel Balda, Ph.D.

[6] PEKÁREK, Dominik. *Popis a testování komunikačních protokolů normy IEC 60870-5-103 a 60870-5-104.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Stanislav Sumec, Ph.D.

[7] POLAK, Martin. *PROTOKOL IPV6 V BEZDRÁTOVÝCH SENZOROVÝCH SÍTÍCH.* Brno, 2011. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Vladimír Červenka.

[8] MATOUŠEK, Petr. *Analysis of DLMS Protocol* [online]. 2017, , 38 [cit. 2019-04-24]. Dostupné z: http://www.fit.vutbr.cz/research/pubs/tr.php.en?file=%2Fpub%2F11616%2FTR-DLMS.pdf&id=11616

[9] *Komunikační protokol CANopen: Základní informace a zadání úloh* [online]. , 12 [cit. 2019-04-24]. Dostupné z: http://www.mcu.cz/images/newspost_images/18/22/popis_canopen.pdf

[10] *OPC UA Client SDK .NET: Tutorial WorkshopConsoleClient and WorkshopClientSample* [online]. In: . s. 40 [cit. 2019-04-24]. Dostupné z: https://technosoftware.com/documents/OPC_UA_Client_Development_with_NET.pdf

[11] *Principy komunikace a diagnostika sítí Profinet* [online]. In: . 2013, s. 4 [cit. 2019-04-24]. Dostupné z: http://www.automa.cz/Aton/FileRepository/pdf_articles/10377.pdf

[12] *Profinet - Standard pro průmyslový Ethernet v automatizaci* [online]. In: . 2005, s. 15 [cit. 2019-04-24]. Dostupné z: http://stest1.etnetera.cz/ad/current/content/data_files/automatizacni_systemy/prumyslova_komunikace/profinet/profinet_04_2005_cz.pdf

[13] RONEŠOVÁ, Andrea. *Přehled protokolu MODBUS* [online]. In: . 2005, s. 19 [cit. 2019-04-24]. Dostupné z: http://home.zcu.cz/ ronesova/bastl/files/modbus.pdf

[14] BACnet. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/BACnet

[15] CANopen. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/CANopen#Communication

[16] DNP3. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/DNP3

[17] ELKO EP. In: *Wikipedia: Otevřená encyklopedie* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2017 [cit. 2019-04-24]. Dostupné z: https://cs.wikipedia.org/wiki/ELKO_EP

[18] IEC 62056. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2018 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/IEC_62056

[19] IEEE 802.15.4. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/IEEE_802.15.4

[20] LonTalk. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2016 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/LonTalk

[21] M-Bus. In: *Wikipedia: Otevřená encyklopedie* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2018 [cit. 2019-04-24]. Dostupné z: https://cs.wikipedia.org/wiki/M-Bus

[22] Modbus. In: *Wikipedia: Otevřená encyklopedie* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2017 [cit. 2019-04-24]. Dostupné z: https://cs.wikipedia.org/wiki/Modbus

[23] MQTT. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/MQTT

[24] OPC Unified Architecture. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/OPC_Unified_Architecture#Protocols

[25] PROFINET. In: *Wikipedia: Otevřená encyklopedie* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2018 [cit. 2019-04-24]. Dostupné z: https://cs.wikipedia.org/wiki/PROFINET

[26] Zigbee. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/Zigbee

[27] ZigBee. In: *Wikipedia: Otevřená encyklopedie* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2017 [cit. 2019-04-24]. Dostupné z: https://cs.wikipedia.org/wiki/ZigBee

[28] Z-Wave. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/Z-Wave

[29] 6LoWPAN. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2019 [cit. 2019-04-24]. Dostupné z: https://en.wikipedia.org/wiki/6LoWPAN

[30] Libraries. *Libraries* [online]. 2019, 2019 [cit. 2019-04-24]. Dostupné z: https://github.com/mqtt/mqtt.github.io/wiki/libraries

[31] *Arduino Library for XBee S2 ZigBee* [online]. 2017 [cit. 2019-04-24]. Dostupné z: https://github.com/ericburger/simple-zigbee

[32] *Asynchronous Python Library for KNX* [online]. [cit. 2019-04-24]. Dostupné z: https://xknx.io/

[33] *BACnet* [online]. [cit. 2019-04-24]. Dostupné z: https://www.cimetrics.com/collections/bacnet?gclid=CjwKCAjw7_rlBRBaEiwAc23rhih7Ymh30vdArkwzpaozQwNVcpNjneRoCHH0R9zkVsowo7w3H7MfwxoCO8kQAvD_BwE

[34] *BACnet Stack* [online]. 2019 [cit. 2019-04-24]. Dostupné z: http://bacnet.sourceforge.net/

[35] *BACstac* [online]. [cit. 2019-04-24]. Dostupné z: https://www.cimetrics.com/collections/bacstac

[36] *CANopen — vyšší komunikační protokol pro vestavné sítě* [online]. [cit. 2019-04-24]. Dostupné z: http://automa.cz/cz/casopis-clanky/canopen-vyssi-komunikacni-protokol-pro-vestavne-site-2004_04_32279_854/

[37] *Co je CANopen a jak na něj* [online]. 2006 [cit. 2019-04-24]. Dostupné z: https://vyvoj.hw.cz/produkty/co-je-canopen-a-jak-na-nej.html

[38] *Co je Synco living?* [online]. [cit. 2019-04-24]. Dostupné z: https://w5.siemens.com/web/cz/cz/corporate/portal/home/produkty_a_sluzby/ibt/synco_living/o_systemu/pages/co_je_synco_living.aspx

[39] *Gurux* [online]. [cit. 2019-04-24]. Dostupné z: https://github.com/Gurux

[40] *Gurux DLMS library for C#* [online]. 2019 [cit. 2019-04-24]. Dostupné z: https://github.com/Gurux/Gurux.DLMS.Net

[41] *IEC 60870-5-101/104 C#/.NET* [online]. [cit. 2019-04-24]. Dostupné z: https://www.mz-automation.de/communication-protocols/iec-60870-5-101-104-c-net-source-code-library/

[42] *IEC 61850: soubor norem pro komunikaci v energetice s velkým potenciálem výhod* [online]. [cit. 2019-04-24]. Dostupné z: http://automa.cz/cz/casopis-clanky/iec-61850-soubor-norem-pro-komunikaci-v-energetice-s-velkym-potencialem-vyhod-2010_03_40771_5154/

[43] *Komunikace přes rozhraní OPC UA* [online]. [cit. 2019-04-24]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Subsystems/Comm/OPC/OPCUA.htm

[44] *Kvaser CANopen Stack* [online]. [cit. 2019-04-24]. Dostupné z: https://tke.fi/products/can-bus/software/kvaser-canopen-stack/?gclid=CjwKCAjw7_rlBRBaEiwAc23rhpzUVEoaMljnHIk_YgXoGre7tbHdbJ-kaJsMT_59T2LvoOMsnvBfYBoC93gQAvD_BwE

[45] *Libmbus: M-bus Library from Raditex Control* [online]. [cit. 2019-04-24]. Dostupné z: http://www.rscada.se/libmbus/

[46] *Libmodbus* [online]. [cit. 2019-04-24]. Dostupné z: https://libmodbus.org/documentation/

[47] *Libraries* [online]. [cit. 2019-04-24]. Dostupné z: https://www.iot-lab.info/libraries/

[48] *Library* [online]. 2011 [cit. 2019-04-24]. Dostupné z: http://mqtt.org/tag/library

[49] *Library for PROFINET data records* [online]. 2018 [cit. 2019-04-24]. Dostupné z: https://support.industry.siemens.com/cs/document/109753067/library-for-profinet-data-records?dti=0&lc=en-AE

[50] *M-BUS (Meter-Bus) - základní popis komunikačního protokolu* [online]. 2010 [cit. 2019-04-24]. Dostupné z: https://automatizace.hw.cz/mbus-meterbus-zakladni-popis-komunikacniho-modelu

[51] *MQTT* [online]. 2016 [cit. 2019-04-24]. Dostupné z: https://www.iot-portal.cz/2016/05/24/mqtt/

[52] *MQTTnet* [online]. 2019 [cit. 2019-04-24]. Dostupné z: https://github.com/chkr1011/MQTTnet

[53] *MQTT 101 - How to Get Started with the lightweight IoT Protocol* [online]. 2014 [cit. 2019-04-24]. Dostupné z: https://www.hivemq.com/blog/how-to-get-started-with-mqtt/

[54] *M2Mqtt* [online]. [cit. 2019-04-24]. Dostupné z: https://m2mqtt.wordpress.com/

[55] *Norma IEC 61850* [online]. [cit. 2019-04-24]. Dostupné z: https://www.se.com/cz/cs/product-range-presentation/60793-norma-iec-61850/

[56] *OPC Advanced SDK for .NET Framework and .NET Standard* [online]. 2019 [cit. 2019-04-24]. Dostupné z: https://nugetmusthaves.com/Package/Opc.UaFx.Advanced

[57] *OPC Client Toolkit Made Right* [online]. [cit. 2019-04-24]. Dostupné z: http://www.opclabs.com/products/quickopc

[58] *OPC UA Framework Advanced* [online]. [cit. 2019-04-24]. Dostupné z: https://wiki.traeger.de/en/software/sdk/opc-ua/opc-ua.framework.advanced/start

[59] *OPC Unified Architecture .NET Standard* [online]. 2019 [cit. 2019-04-24]. Dostupné z: https://github.com/OPCFoundation/UA-.NETStandard

[60] *OpenIEC61850* [online]. [cit. 2019-04-24]. Dostupné z: https://www.beanit.com/iec-61850/

[61] *OpenZWave Library* [online]. [cit. 2019-04-24]. Dostupné z: http://www.openzwave.com/dev/index.html

[62] *Protokolu Modbus RTU v kostce* [online]. [cit. 2019-04-24]. Dostupné z: https://ipc2u.cz/articles/simple-decisions/protokolu-modbus-rtu-v-kostce/

[63] *Using the C# API* [online]. [cit. 2019-04-24]. Dostupné z: http://libiec61850.com/libiec61850/documentation/using-the-c-api/

[64] *Úvod do BACnetu - Building Automation and Controls Network* [online]. 2012 [cit. 2019-04-24]. Dostupné z: https://automatizace.hw.cz/uvod-do-bacnetu-building-automation-and-controls-network

[65] *Úvod do KNX* [online]. 2013 [cit. 2019-04-24]. Dostupné z: https://automatizace.hw.cz/teorie-a-praxe/knx.html

[66] *What is a Smart Building* [online]. 2019 [cit. 2019-04-24]. Dostupné z: https://www.trueoccupancy.com/blog/what-is-a-smart-building

[67] *ZigBee a Bluetooth LE — silná kombinace pro chytré domy i průmysl* [online]. 2017 [cit. 2019-04-24]. Dostupné z: https://www.iot-portal.cz/2017/11/13/zigbee-a-bluetooth-le-silna-kombinace-pro-chytre-domy-i-prumysl/

[68] *Z-Wave* [online]. [cit. 2019-04-24]. Dostupné z: https://www.mojefibaro.cz/system/technologie-z-wawe/

A Obsah přiloženého CD

- Textová práce v .pdf
- Zdrojový kód LaTeXu ve formě .txt
- Aplikace se zdrojovými kódy