Mendelova univerzita v Brně
Provozně ekonomická fakulta

Využití ARM procesorů v řídícím systému vytápění

Diplomová práce

Vedoucí práce:

Ing. Vít Ondroušek, Ph.D. **Bc. Václav Steiger**

Brno 2016

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Vítu Ondrouškovi, Ph.D. za profesionální přístup a značnou trpělivost při vedení této diplomové práce. Dále také děkuji zaměstnancům výzkumu a vývoje sekce Regulace a BMS firmy Faster.cz, s.r.o z Brna - Maloměřic, především Ing. Adamu Škorpíkovi a Ing. Miroslavu Vondrovi, jejichž přístup, ochota a nasazení byly pro zdárné dokončení této diplomové práce zcela nepostradatelné. Poděkování patří také Ing. Robertu Roušovi a Ing. Janu Kolomazníkovi, Ph.D., bez nichž bych se k řešení této práce vůbec nedostal.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Využití ARM procesorů v řídícím systému vytápění**
vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 25. prosince 2016 _____

Abstract

Steiger, V. Usage of ARM processors in heating control system. Diploma thesis. Brno: Mendel University, 2016.
This thesis aims to simplify the process of controlling heating and cooling systems with standardised, widely accessible computers on purpose of integration into „smart home“ systems. In theoretical part, heating systems available on the market are reviewed, mainly with focus on communication interfaces and protocols in use. Then the specifications of one of the most widely used protocol in heating systems communication – OpenTherm – and one of the most widely used protocols in industrial computers communication – RS485 Modbus – are examined. Theoretical part also contains a description of integrated development environment and platform-specific development properties for STM32 ARM platform. In desing and Implementation is described RS485 Modbus – OpenTherm converter firmware written in C language. Behaviour of the converter is then tested in practise.

Keywords

OpenTherm, Modbus, RS485, ARM, STM32, converter

Abstrakt

Steiger, V. Využití ARM procesorů v řídícím systému vytápění. Diplomová práce. Brno: Mendelova univerzita v Brně, 2016.
Práce si klade za cíl usnadnit řízení topných a chladících systémů běžně dostupnými počítači za účelem integrace do systémů „chytré domácnosti“. V teoretické části jsou zkoumány systémy vytápění dostupné na trhu, hlavně z hlediska typů použitých komunikačních rozhraní a protokolů. Dále je detailně rozebrána specifikace jednoho z široce používaných protokolů na straně systémů vytápění – OpenTherm a jednoho z protokolů hojně využívaných v komunikaci mezi průmyslovými počítači – RS485 Modbus. Teoretická část ještě obsahuje popis vývojového prostředí a specifik vývoje na platformě STM32 ARM, zvolené pro implementaci převodníku RS485 Modbus – OpenTherm. V částech Návrh a Implementace je popsáno konkrétní řešení firmware převodníku RS485 Modbus - Opentherm v jazyce C. Funkce převodníku je poté prakticky ověřena.

Klíčová slova

OpenTherm, Modbus, RS485, ARM, STM32, převodník.

Obsah

1 Úvod **10**

2 Cíl práce a motivace **12**

2.1 Motivace 12

2.2 Cíl práce 12

3 Teoretická část **14**

3.1 Rozbor existujících zařízení pro vytápění 14

3.2 Popis komunikačních protokolů 25

3.2.1 RS485 Modbus 28

3.2.2 Opentherm 36

3.3 Vývojové prostředí CooCox CoIDE 45

3.4 Vývojový kit STM32 Discovery 46

4 Požadavky **48**

5 Metodika **50**

6 Návrh **51**

7 Implementace **56**

7.1 Implementace nízkoúrovnových funkcí 56

7.2 Hlavní programová smyčka 64

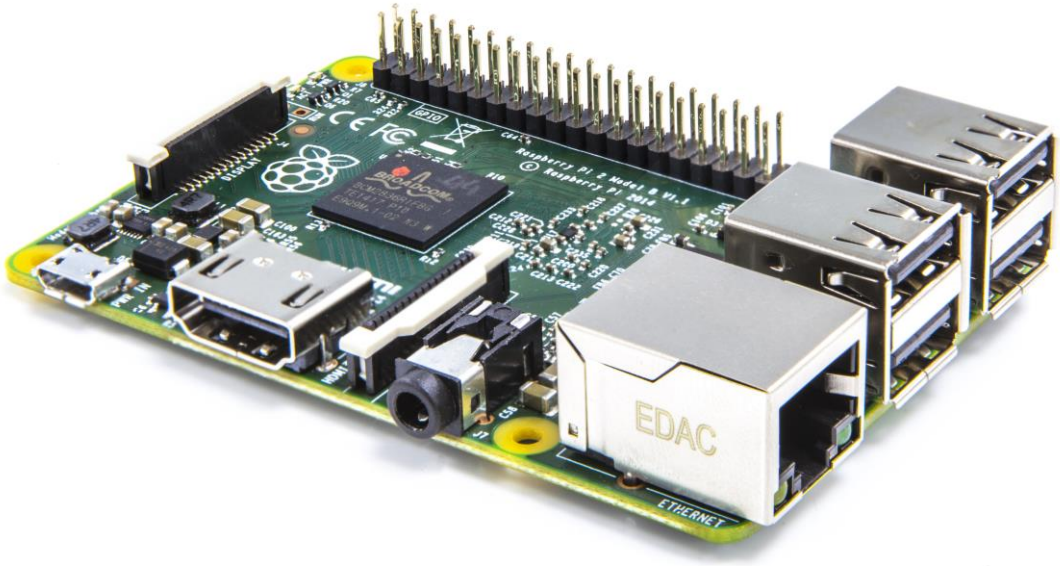
8 Testování a zhodnocení **68**

9 Závěr **69**

10 Literatura **70**

1 Úvod

V poslední době se stále častěji setkáváme se snahami automatizovat co nejvíce problémů, úkolů a činností lidského života, nebo alespoň při jejich řešení stále více využívat výpočetní techniku. Příchod nových počítačových platforem jako Raspberry Pi (obr. 1) , které nabízejí velmi lákavou kombinaci malých rozměrů, nízké ceny, výborné podpory komunikačních rozhraní a protokolů spolu s možností provozovat standartní desktopový operační systém (například typu GNU LINUX) znamená, že prototypování a vývoj na míru stavěných zařízení řízených počítačem nikdy nebyl jednodušší ani levnější. Výkon těchto počítačů je přitom dostačující i pro poměrně náročné úlohy, jako např. rozpoznávání obrazu pro strojové vidění.



Obr. 1 Raspberry PI 2. Zdroj: http://www.raspberrypi.org

Platformy jako Arduino [1], které za ještě nižší cenu nabízejí především výbornou softwarovou podporu ve formě velmi rozsáhlé palety knihoven s již implementovanou funkcionalitou v kombinaci s možností snadného připojení hardwaru jako jsou displaye, klávesnice, drátové i bezdrátové komunikační moduly, moduly pro příjem signálů (GPS, DCF 77, atd.) znamenají výrazné zjednodušení a zrychlení vývoje zařízení ovládáných integrovaným mikrokontrolérem. Navíc se v oboru programování mikrokontrolérů díky integraci s vývojovými prostředími a nativní podpoře vyšších programovacích jazyků (C, C++) začínají tyto jazyky více objevovat a vytlačují tradiční způsob programování přímo v jazycích symbolických adres – což z hlediska programátora znamená především výrazné urychlení vývoje aplikací na těchto platformách.

Díky tomu vznikají zcela nové obory a způsoby využití počítače jako řídícího prvku. Ožívají vize výrobců výpočetní techniky staré desítky let o zahrnutí počítačů a počítačového řízení do širokého množství činností v domácnosti. Přichází pojem „Internet of Things“ [2] – tedy myšlenka dálkové ovladatelnosti a přizpůsobitelnosti všemožných spotřebičů od žárovky přes ledničku až například po centrální vytápění a klimatizaci. Tato myšlenka rozšiřuje i vnímání toho, co si můžeme představit pod pojmem zařízení připojené do internetu – na rozdíl od minulosti, kdy koncová zařízení byly prostě stolní počítače, laptopy nebo servery se přes rozmach chytrých mobilních telefonů a inteligentních infotainmentů pro automobily dostáváme k současnosti - levnému, malému počítači v roli řídícímu systému, který může v domácnosti vykonávat jednu specifickou činnost ovládáním žárovky namísto klasického vypínače počínaje a řízením vytápění místo klasického termostatu konče. Možnosti dálkového a výrazně inteligentnějšího řízení, které se tímto otevírají, uvádějí do praktického nasazení dlouho pouze teoretický pojem „Chytrá domácnost“ (Smart home) [3].

2 Cíl práce a motivace

2.1 Motivace

Vývoj specializovaných počítačových řídících systémů nebyl nikdy jednodušší. Problém s tímto poměrně náhlým a prudkým vzestupem počítačového řízení většinou spočívá hlavně v hardwarové nekompatibilitě rozhraní řídících počítačů s rozhraními řízených systémů. Chceme počítačem často řídit systém nebo zařízení, které na to původně vůbec nebylo stavěné a náš řídící systém musí nahrazovat určitou formu analogového či elektromechanického řízení. Systém případně podporuje digitální řízení, využívá ale obskurní rozhraní či protokol mezi systémy daného oboru běžný, ale s běžně dostupnými počítači zcela nekompatibilní. Dobrou ukázkou takového pro běžné počítače obskurního protokolu je například protokol CAN [4] na sběrnici CAN-BUS, který se běžně používá v automobilovém průmyslu pro komunikaci mezi řídícími jednotkami ve vozidlech, ale i v širší průmyslové praxi pro komunikaci například v rámci produkčních výrobních linek. Běžné počítače si ale s protokolem CAN většinou neporadí a chceme-li vyvíjet zařízení pro sběrnici CAN-BUS, je obvykle nutné data nejprve nějakým způsobem převádět na rozhraní počítačem podporované.

Na rozdíl od sběrnice CAN, pro kterou existuje poměrně omezený, ale přece výběr zařízení – počítačů, které ji podporují, rozhraní OpenTherm používané pro komunikaci v systémech vytápění a klimatizace je běžnými počítačovými platformami přehlíženo dokonale. Díky specifickému použití a technologickému pozadí, ze kterého vychází, je rozhraní OpenTherm naprosto nekompatibilní s jakýmkoli rozhraním, které se vyskytuje na běžně dostupných počítačích. Chceme-li do „Chytré domácnosti“ zahrnout například i inteligentní počítačové řízení kotle pro vytápění, vzniká problém. Nejsme totiž schopni nijak snadno konvertovat rozhraní OpenTherm, po kterém kotel komunikuje s řídícím termostatem na jakékoliv rozhraní vlastní našemu řídícímu počítači.

Tato práce řeší návrh, implementaci a testování převodníku, který je takové konverze schopen převodem dat mezi rozhraními OpenTherm a RS 485 Modbus, které je široce podporované napříč počítačovými platformami.

2.2 Cíl práce

Cílem práce je vyvinout funkční firmware pro převodník mezi rozhraními OpenTherm a RS485 Modbus. Situace na trhu ukazuje poměrně hojnou podporu pro využívání sběrnice OpenTherm pro komunikaci kotlů a klimatizačních systémů s řídícími termostaty, stejně jako výbornou podporu rozhraní RS 485 miniaturními jednodeskovými embedded počítači, což bude popsáno v teoretické části práce. Rozborem těchto dvou protokolů také bude demonstrována jejich značná vzájemná nekompatibilita, která předznamená nutnost poměrně komplexního řešení firmware převodníku ve snaze tuto nekompatibilitu překonat. Dále bude představena platforma, která byla vybrána pro vývoj, a se kterou je třeba se seznámit z hlediska vnitřní architektury a obsluhy integrovaných periferií. Z

ohledem na specifika a vlastnosti platformy bude proveden návrh a implementace firmware převodníku, jehož funkce poté bude otestována.

3 Teoretická část

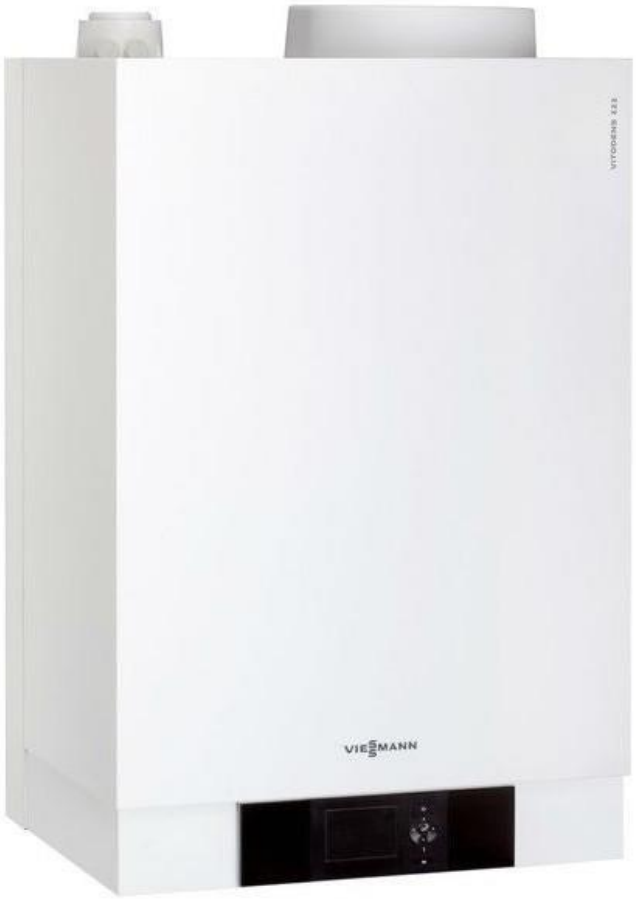
3.1 Rozbor existujících zařízení pro vytápění

Základní charakteristika převodníku, jehož firmware je hlavním předmětem vývoje v rámci této práce, byla direktivně stanovena firmou Faster.cz, s.r.o. [23] coby produkt, který vyřeší problém řízení značné části systémů vytápění a klimatizace nabízených v současnosti (2015) na trhu a který bude možné ovládat širokou paletou různých nadřazených řídících systémů. Následující rozbor existujících zařízení pro vytápění ukáže, zda je sběrnice OpenTherm opravdu podporována v dostatečně velkém rozsahu napříč kategoriemi topných systémů, na jaké podtypy těchto systémů můžeme převodník nasadit běžně a u jakých podtypů těchto systémů je rozhraní Opentherm spíše vzácností.

Snadnou cestou, jak vyhledat topná a chladící zařízení podporující protokol OpenTherm, by měla být návštěva webu The OpenTherm asociation [5]. Asociace, která stojí za vývojem a standardizací rozhraní OpenTherm po stránce hardwaru i specifikace komunikačního protokolu totiž na svém webu kromě jiného také uvádí výčet firem, které členstvím v této asociaci projevili zájem o nasazení podpory komunikace přes rozhraní OpenTherm do svých produktů. Jsou zde zastoupeni mezi jinými i velcí výrobci topných systémů a kotlů, jako například Viessmann, Vaillant, Siemens, Bosch, Danfoss, ale i výrobci centrálních tepelných čerpadel a klimatizací, jako např Hitachi.

Pro účely rozboru je ale třeba nalézt konkrétní zařízení, která bude možné řídit přes rozhraní OpenTherm, nejlépe z kategorie menších zdrojů horké vody určených pro domácnosti, byty či rodinné domy, využívající jako palivo zemní plyn. Taková zařízení budou dobře odpovídat modelové situaci příkladu nejpravděpodobnějšího využití vyvíjeného převodníku: zákazník se snaží vystavět řízení vytápění ve stylu „chytré domácnosti“ v měřítku jednoho bytu či rodinného domku.

Hledání začneme u výrobků společnosti Viessmann. Firma Viessman [6] je předním výrobcem systémů vytápění a klimatizace, jako i dalších zařízení souvisejících s výrobou energií a tepla, jako např tepelná čerpadla, vyvíječe a úpravny bioplynu, fotovoltaické a solární termické systémy atd. Firma se výrobně nezaměřuje na konkrétní výkonové pásmo topných systémů, ale nabízí kompletní portfolio výrobků s výkony od 35kW do 116MW tepelných. Při pohledu na dokumentaci k základní modelovou řadě plynových kondenzačních kotlů firmy Viessmann určenou pro domácnosti, tedy modely Vitodens 100 [7] a 111 ale zjišťujeme, že v žádné produktové dokumentaci se podpora ovládání pomocí protokolu OpenTherm neuvádí. Dokumentace se sice zminuje o řízení kotle externím termostatem, digitální regulaci, či o podpoře proprietálního řídícího systému Vitotronic, protokol OpenTherm ale jakoby byl všem zařízením řady Vitodens neznámý. To je znepokojivý fakt, uvážíme-li členství firmy Viessmann v asociaci OpenTherm.



Obr. 2 Plynový kondenzační kotel Viessmann Vitodens 111 určený pro domácnosti. Zdroj: [6]

Nezbývá, než pokusit se využít při hledání opačný přístup. Rozhraní OpenTherm má ve své typické aplikaci spojovat kotel s řídícím termostatem. Nabízí se tedy otázka: Vyrábí firma Viessmann termostat, komunikující přes rozhraní OpenTherm? A pokud ano, jaká topná zařízení jsou s tento termostatem uvedena jako kompatibilní?

Odpověd na tuto otázku existuje v podobě termostatu s podporou ekvitermní regulace Viessmann Vitotrol 100 OT [8]. Pohled do provozní dokumentace a návodu k použití tohoto termostatu ale příliš mnoho nového světla do celé problematiky nevnáší. Z dokumentace vyplývá, že Vitotrol 100 OT je pouze jednou ze tří verzí termostatu, které se od sebe liší právě druhem použitého komunikačního rozhraní, tedy dostupná ke koupi je také například ještě verze RT, která používá jednoduché dvoustavové ovládání kotle se stavy zapnuto/vypnuto. V dokumentaci verze OT navíc také není uvedeno které konkrétní modely topných systémů jsou s termostatem kompatibilní.

Obr. 3 Ekvitermní termostat Viessmann Vitotrol 100 ve verzi OT. Zdroj: [6]

Po dlouhotrvajícím prohledávání dokumentací které nikam nevedlo a mnoha telefonátech s českým zastoupením firmy Viessmann se podařilo vnést do celé problematiky přece jen trochu světla. Podpora protokolu OpenTherm v základní a nejprodávanější řadě plynových kondenzačních kotlů Viessmann Vitodens 100 existuje, kotle podporují komunikaci po protokolu Opentherm vždy v podtypu -W. Toto platí beze změny i pro vyšší řadu Vitodens 111, tedy kotle Vitodens 100-W a 111-W ve všech modifikacích můžeme řídit přes rozhraní OpenTherm, ovšem je ještě nutné do kotle doinstalovat speciální kabelový svazek X-21, který je běžně dostupný na trhu jako příslušenství. Toto dokonce platí i pro nejvyšší řady domácích kotlů Vitodens 200 a 222, i když jen v omezeném měřítku – podle českého zastoupení firmy Viessmann je možné kotle Vitodens 200 a 222 v rámci objednávky naspecifikovat pro podporu řízení přes protokol OpenTherm. Takový kotel je pak také označen jako Vitodens 200-W, respektive 222-W Dlužno dodat, že tato informace mi byla předána pouze telefonicky a zastoupení firmy Viessman odmítlo poskytnout relevantní písemný zdroj s uvedením důvodu, že takový zdroj neexistuje.

Tyto informace znamenají z pohledu vývoje převodníku pracujícího s rozhraním OpenTherm dobré zprávy. Znamenají totiž, že většina plynových kondenzačních kotlů určených pro domácnosti firmy Viessmann podporuje řízení pomocí protokolu OpenTherm. Zvláštní na celé věci je ale znatelná a značná neochota firmy Viessmann přiznávat u svých kotlů podporu protokolu OpenTherm. Namísto toho firma protěžuje vlastní, uzavřený a proprietární inteligentní řídící systém Vitotronic a podpora otevřeného protokolu OpenTherm je s písemně nedohledatelná u všech modelů. U větších topných systémů (Vitodens 200, Vitodens 222) jsem se informaci o existující podpoře dozvěděl až na přímý telefonický dotaz a v běžně dostupných informačních pramenech o ní není sebemenší zmínka.

Takové chování firmy Viessmann je totiž v přímém rozporu s původní myšlenkou rozhraní OpenTherm a cílem, který se snaží naplnovat asociace OpenTherm. Tato původní myšlenka totiž spočívá v boji s proprietárními a

uzavřenými rozhraními, protokoly a způsoby komunikace v rámci systémů vytápění převážně v měřítku domácností a rodinných domů. Specifikace komunikačního protokolu stejně jako rozhraní OpenTherm včetně jeho hardwarové stránky je otevřená a veřejně přístupná pro všechny, nejen pro členy asociace. Je-li společnost členem asociace (což například společnost Viessmann je), měla by se aktivním způsobem podílet na podpoře rozhraní OpenTherm nasazováním tohoto rozhraní do svých produktů, případně by měla využívat poznatky z vlastního výzkumu a vývoje pro další rozšiřování a vylepšování specifikace rozhraní OpenTherm. Jinými slovy - současná strategie společnosti Viessmann zaměřená prakticky pouze na podporu a protěžování proprietálního, uzavřeného protokolu vlastní konstrukce jde přesně proti záměrům OpenTherm asociace, které by společnost Viessmann kvůli svému členství v asociaci měla určitě plnit.

Společnost Viessmann je sice předním dodavatelem systémů vytápění pro domácnosti a rodinné domy, není ale jediným. Lepší představu o situaci na trhu lze získat pohledem na portfolio výrobků jiného výrobce. V rešerši budeme pokračovat u výrobků společnosti Vaillant.

Společnost Vaillant [9] je výrobcem topné a klimatizační techniky s více než 140letou tradicí. V současnosti vyrábí v továrnách ve více než 13 státech světa, ačkoliv původní sídlo má stejně jako firma Viessmann v Německu. Narozdíl od firmy Viessmann se firma Vaillant zabývá spíše výrobou tepelných čerpadel, systémů pro využití sluneční energie a kogeneračních systémů.

Obr. 4 Plynový kondenzační kotel Vaillant ecoTEC plus určený pro vytápění domácností. Zdroj: [9]

Pro účely této práce je důležité, že firma Vaillant není také žádným nováčkem ve výrobě systémů pro vytápění domácností a rodinných domů – právě pro tento účel nabízí kompletní portfolio závěsných i stacionárních plynových kotlů s provozem na zemní plyn.

V portfoliu firmy Vaillant představují řešení vytápění v měřítku domácností a a rodinných domů kotle modelové řady ecoTEC plus a ecoTEC pro. Na první pohled se při prostudování produktové dokumentace a návodu k použití zdá, že strategie firmy Vaillant se nápadně podobá strategii firmy Viessmann. V žádném návodu k použití nebo produktové dokumentaci není o protokolu OpenTherm ani zmínka, natož o jeho podpoře. Co ale je zmíněno v dokumentaci ke všem modelům kotlů ecoTEC [10], včetně toho nejméně výkonného a nejlevnějšího, je podpora inteligentního externího ekvitermního řízení pomocí proprietárního komunikačního rozhraní s názvem e-bus.

Nabízí se otázka podobná, jako v případě výrobků společnosti Viessmann. Vyrábí společnost Vaillant termostat, komunikující prostřednictvím rozhraní OpenTherm? A pokud ano, která zařízení pro vytápění jsou v dokumentaci uvedena jako s tímto termostatem kompatibilní?

Odpověd na tuto otázku je odlišná, než v případě výrobků společnosti Viessmann. Firma Vaillant nevyrábí žádný termostat, ekvitermní regulátor ani jiný řídící člen, který by podporoval komunikaci pomocí protokolu OpenTherm.

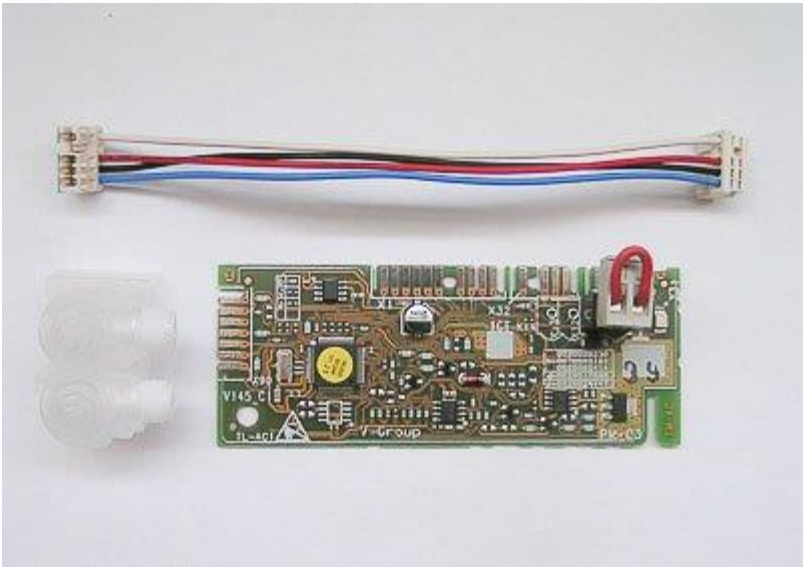
Obr. 5 Základní, nejlevnější model prostorového termostatu Vaillant calorMATIC 350 – podporuje pouze rozhraní e-bus. Zdroj: [9]

Naopak, všechna regulační technika a termostaty vyráběné firmou Vaillant komunikují výhradně prostřednictvím uzavřeného, proprietárního rozhraní e-bus, které z výroby podporují všechny plynové kondenzační kotle vyráběné firmou Vaillant a dokonce i ostatní zařízení z firemního portfolia, jako například rekuperační jednotky a systémy vytápění a klimatizace pracující na principu tepelného čerpadla.

To jsou velmi znepokojivé informace, uvážíme-li členství firmy Vaillant v The OpenTherm association. Narozdíl od společnosti Viessmann, která sice podporu komunikaci prostřednictvím rozhraní OpenTherm u svých zařízení nikde přímo neuvádí, nicméně rozhraní přece jen u významné části svých zařízení OpenTherm podporuje, společnost Vaillant jakoby podporu rozhraní OpenTherm i přes své členství v OpenTherm asociaci zcela zavrhla.

Další průzkum a telefonování ukázalo, že alespon v případě plynových kondenzačních kotlů řady ecoTEC situace není zcela beznadějná. Coby dokoupitelné příslušenství totiž pro řadu kotlů ecoTEC existuje přídavný modul s názvem VR-33 OpenTherm module. Tento modul po doinstalování do kotle a propojení s jeho hlavní řídící deskou rozšiřuje komunikační schopnosti kotle o rozhraní OpenTherm. Standartně přítomné rozhraní e-bus je ale stále aktivní, použitelné a navíc má prioritu – je-li ke kotli připojen termostat, nebo nadřazený

řídící systém komunikující pomocí rozhraní e-bus, kotel vůbec nekomunikuje ani nereaguje na příkazy přijímané přes rozhraní OpenTherm.



Obr. 6 Vaillant VR33 OpenTherm module – způsob, jakým lze kotle Vaillant ecoTEC dovybavit OpenTherm rozhraním. Zdroj: http://www.klima-parts.nl

V rámci získání komplexní představy o situaci na trhu se systémy pro vytápění řiditelnými pomocí rozhraní OpenTherm se seznámíme ještě s nabídkou tuzemského výrobce značky Thermona.

Firma Thermona [11], sídlící v současnosti v Zastávce u Brna je jediným výrobcem systémů pro vytápění, který uskutečnuje výrobu svých zařízení v České republice. Byla založena v roce 1990 a její portfolio obsahuje kompletní nabídku systémů pro vytápění a ohřev teplé vody v měřítku pro domácnosti a rodinné domy. Specialitou ve výrobním programu společnosti Thermona jsou tzv. kaskádová topná zařízení, kdy řízení jednotlivých zdrojů tepla a teplé vody umožnuje zapojení do kaskády a samostatné sekvenční spínání, stejně jako nezávislou regulaci každého zdroje. Tento přístup se promítá do schopnosti dosahovat mnohem širší modulace výkonu v porovnání s konkurenčními řešeními, a tím i vyšší úspory energie a paliv.

Námi zkoumané kategorii systémů pro vytápění v měřítku domácnosti nebo rodinného domu odpovídají z portfolia společnosti Thermona kotle řady Therm KD.A. Jedná se o plynové kondenzační kotle, určené pro ohřev pouze teplé užitkové vody určené pro vytápění.

Obr. 7 Základní model základní řady plynových kondenzačních kotlů Thermona 14 KD.A – plně podporuje řízení pomocí rozhraní OpenTherm, stejně jako drtivá většina systémů pro vytápění firmy Thermona. Zdroj: [11]

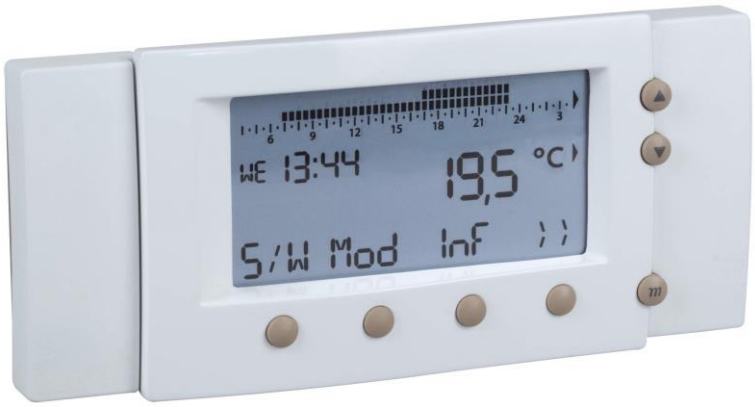
První pohled do produktové dokumentace [12] výrobků firmy Thermona ihned ukazuje nápadnou odlišnost přístupu firmy Thermona k poskytování informací. Skutečnost, že všechny výrobky v portfoliu firmy jsou přehledně seřazeny v jediném dokumentu - Katalogu produktů, výrazně usnadnuje a ulehčuje proces vyhledávání informací. Po skušenostech s hledáním informací o výrobcích firem Vaillant a Viessmann je opravdovým překvapením odpověd na otázku podpory řízení pomocí rozhraní OpenTherm. Podle katalogu produktů totiž z vyjímkou elektrických kotlů ekonomické řady Therm ELN podporují řízení pomocí protokolu OpenTherm všechny plynové a dokonce i všechny elektrické kotle pro ohřev topné vody, teplé vody a kotle kombinované, což je vzhledem k

faktu, že společnost Thermona není členem asociace OpenTherm opravdu překvapivé. Informace o podpoře řídících protokolů je také na rozdíl od dokumentace systémů pro vytápění firem Vaillant a Viessman velmi jasně v Katalogu produktů uvedena pro každý produkt. Vyhledávání a telefonické dotazování se na informace tedy v případě výrobků společnosti Thermona zcela odpadá.

Obr. 8 Elektrický kotel ekonomické řady Thermona ELN – kotle této řady jako jediné z celého portfolia výrobků firmy Thermona nepodporují řízení pomocí protokolu OpenTherm. Zdroj: [11]

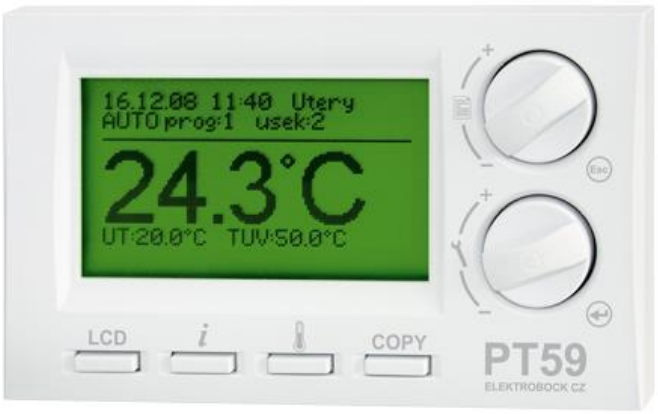
Společnost Thermona se tedy nesnaží po vzoru svých německých konkurentů, firem Viessmann a Vaillant, vyvíjet a protěžovat vlastní řídící systémy využívající proprietární komunikační protokoly. Namísto toho se rozhodla svoji podporu pro číslicové řídící systémy otevřeně implementovat využitím otevřeného protokolu OpenTherm, a to i přesto, že sama není členem asociace OpenTherm. V souladu s touto politikou také společnost Thermona nabízí sortiment inteligentních

termostatů s modely PT59 a CR04, který umožnuje řízení kaskádového zapojení topných jednotek. Při bližším zkoumání produktové dokumentace se ovšem ukazuje, že oba modely inteligentních termostatů jsou ve skutečnosti výrobky jiných firem. CR04 [13] je originálním výrobkem firmy Honeywell, zatímco PT59 [14] je výrobkem české společnosti Elektrobock, s.r.o.



Obr. 9 Inteligentní termostat Honeywell CR04 s podporou protokolu OpenTherm. Zdroj: http://www.honeywell.com

Uvážíme-li zásady asociace OpenTherm, kdy protokol OpenTherm by měl být univerzální mezi všemi zařízeními které jej podporují, výrobcem kotlů přímo podporovaná komunikace s inteligentními termostaty vyráběnými jiným výrobcem se jeví jako ideální. Narozdíl od situace s výrobky značek Vaillant a Viessmann, které protokol pouze tiše podporují se zdá, že společnost Thermona se otevřeně snaží o zavedení protokolu OpenTherm do běžného užívání, o jeho faktickou standartizaci a tím o naplnění původní myšlenky asociace OpenTherm, tedy situace, kdy konečný zákazník má jistotu, že topné zařízení společnosti Thermona může regulovat bez obav o kompatibilitu různými regulátory různých výrobců. Pohledem do produktové dokumentace inteligentních termostatů PT59 a CR04 je jasné, že podobná snaha udělat z protokolu OpenTherm opravdový standart napříč celým svým portfoliem je mezi výrobci topných zařízení a systémů pro vytápění a produkci teplé vody vzácná a ojedinělá. Produktová dokumentace inteligentního termostatu Elektrobock PT59 totiž uvádí nutnost výběru typu (značky) kotle, se kterým má termostat spolupracovat.



Obr. 10 Inteligentní termostat Elektrobock PT59 s podporou protokolu OpenTherm. Zdroj: http://www.elektrobock.cz

Tato funkcionalita je také uváděna jako nová funkce podporovaná od verze firmwaru 11.14. Narozdíl od ideální situace pro kterou byl protokol OpenTherm navržen, tedy jakýkoliv kotel s podporou protokolu OpenTherm by mělo být bez potíží možné řídit jakýmkoliv inteligentním termostatem podporujícím protokol OpenTherm, vypadá reálná situace spíše tak, že kotel a řídící inteligentní termostat by měly být stejné značky (pokud vůbec výrobce kotle vyrábí i inteligentní termostat s podporou protokolu OpenTherm), nebo je alespon nutné nastavit řídící termostat na správnou značku řízeného kotle a ani pak není zaručeno, že zařízení budou komunikovat bezvadně. Společnost Honeywell v produktové dokumentaci ke svému inteligentnímu termostatu CR04 řeší tento problém s kompatibilitou jiným způsobem – v dokumentaci je přímo uvedeno, která OpenTherm data ID termostat při řízení kotle používá, a která tedy musí být na straně kotle implementovány.

Tento přístup k řešení problémů s kompatibilitou OpenTherm zařízení je poněkud neštastný. Sice v souladu s původní myšlenkou asociace OpenTherm otevřeně uvádí všechny informace o podpoře protokolu, v současné situaci kdy je ale nesnadné zjistit, zda systémy pro vytápění vůbec nějak podporují protokol OpenTherm, natož přesný seznam podporovaných příkazů je tato informace pro koncového uživatele poněkud bezvýznamná. V konečném důsledku se také snižuje počet podporovaných zařízení v reálném provozním nasazení, protože narozdíl od možosti nastavení značky řízeného kotle, kdy má uživatel určitou jistotu že komunikace bude probíhat bez vad a výrobce termostatu komunikaci s kotlem této

značky vyskoušel a odladil, zde musí schopnost komunikace kotel-termostat ověřit až zákazník systémem pokus omyl.

Řešení použité v inteligentním termostatu Honeywell CR04 ale demonstruje a naráží na hlavní a největší problém protokolu OpenTherm a důvod, proč dosud protokol OpenTherm není v rámci systémů pro vytápění a jejich číslicového inteligentního řízení v praxi standardem.

Problém spočívá v samotném návrhu komunikace v rámci protokolu. Komunikace prostřednictvím protokolu OpenTherm probíhá vysíláním zpráv opatřených ID, kde hodnota ID definuje význam dat ve zprávě. Definice aplikační vrstvy protokolu OpenTherm obsahuje seznam ID, které musí zařízení pracující s protokolem OpenTherm bezpodmínečně podporovat. Pro implementaci inteligentního řízení je většinou nutné využívat více informací, tedy více a jiná ID než ta, obsažená v minimálním seznamu podporovaných ID. Každá společnost vyrábějící kotle implementuje inteligentní řízení jinak a používá jinou sadu komunikačních ID. Propojíme-li inteligentní termostat a kotel různých značek, existuje velká šance, že zpracování některých ID zpráv vysílaných termostatem nebude na straně kotle podporováno a obráceně některá ID odpovědí kotle nebudou zpracovány termostatem, komunikace se rozpadne a řízení nebude funkční. Tato situace je přitom plně v souladu s definicí aplikační vrstvy protokolu OpenTherm podle oficiální dokumentace. A přestože nutně podporované ID zpráv pro inteligentní řízení většinou zdaleka nestačí, implementace podpory všech ID zpráv není ve specifikaci protokolu vyžadována. Dokonce je polovina ID zpráv přímo ve specifikaci protokolu rezevována pro testovací a diagnostické zprávy, jejichž významy si může každý výrobce navrhnout sám. Je proto velmi složité, až prakticky nemožné navrhnout inteligentní termostat, který by bezvadně pracoval s jakýmkoliv zařízením pro vytápění jakéhokoliv výrobce prostřednictvím protokolu OpenTherm a byl s nimi plně kompatibilní.

V této situaci se jako jedno z řešení nabízí řízení pomocí běžného počítače. Řídíme-li kotel inteligentním termostatem, který je po stránce hardwaru zpravidla řešen jako jednočipový mikropočítač, narážíme neustále díky situaci s protokolem OpenTherm a jeho podporou v systémech pro vytápění na nutnost úprav palety vysílaných ID zpráv a vytváření různých verzí této palety pro zajištění podpory různých značek a typů systémů pro vytápění. Toto lze u jednočipových mikropočítačů zajistit zpravidla pouze upgrady firmwaru, což nebývá uživatelsky snadno proveditelný proces.

Přesuneme-li algoritmus řízení na nadřazený, běžný počítač, připojený v rámci „Chytré domácnosti“ do internetu, lze proces dalšího vývoje funkcí firmwaru a změn v podobě komunikace kotlů zautomatizovat jednoduše updatem softwaru řídícího běžného počítače. Zmizí problém s případným nákupem modernějšího kotle, se kterým si termostat náhle neporadí – jednoduše stáhneme novou verzi softwaru do řídícího počítače, software se může aktualizovat automaticky, nebo je možné komunikaci mnohem více do hloubky konfigurovat a tím odladit a odstranit případné problémové situace. V situaci, kdy na trhu vlastně neexistuje jednotná implementace komunikace na protokolu OpenTherm je výrazně snažší implementovat podporu na platformě běžného počítače. Běžný počítač může také

implementovat výrazně složitější algoritmy řízení vytápění, například algoritmy, které pro zajištění funkčnosti potřebují externí data (předpověď počasí apod.).

Problém běžných počítačů je ale v nulové podpoře rozhraní OpenTherm. Protože specifikace protokolu OpenTherm velmi jasně definuje fyzickou a linkovou vrstvu protokolu, která činí svojí exotičností největší a hlavní problém. Převodník, který převede fyzickou a linkovou vrstvu protokolu OpenTherm na protokol bližší běžným počítačům (RS485 Modbus) a přitom umožní na aplikační vrstvě protokolu vysílat a přijímat zprávy s jakýmkoliv data ID znamená v kombinaci s nadřazeným běžným řídícím počítačem, v jehož softwaru můžeme plně konfigurovat vysílaná a přijímaná ID zpráv z hlediska kompatibility se systémy pro vytápění jedno z nejlepších možných řešení.

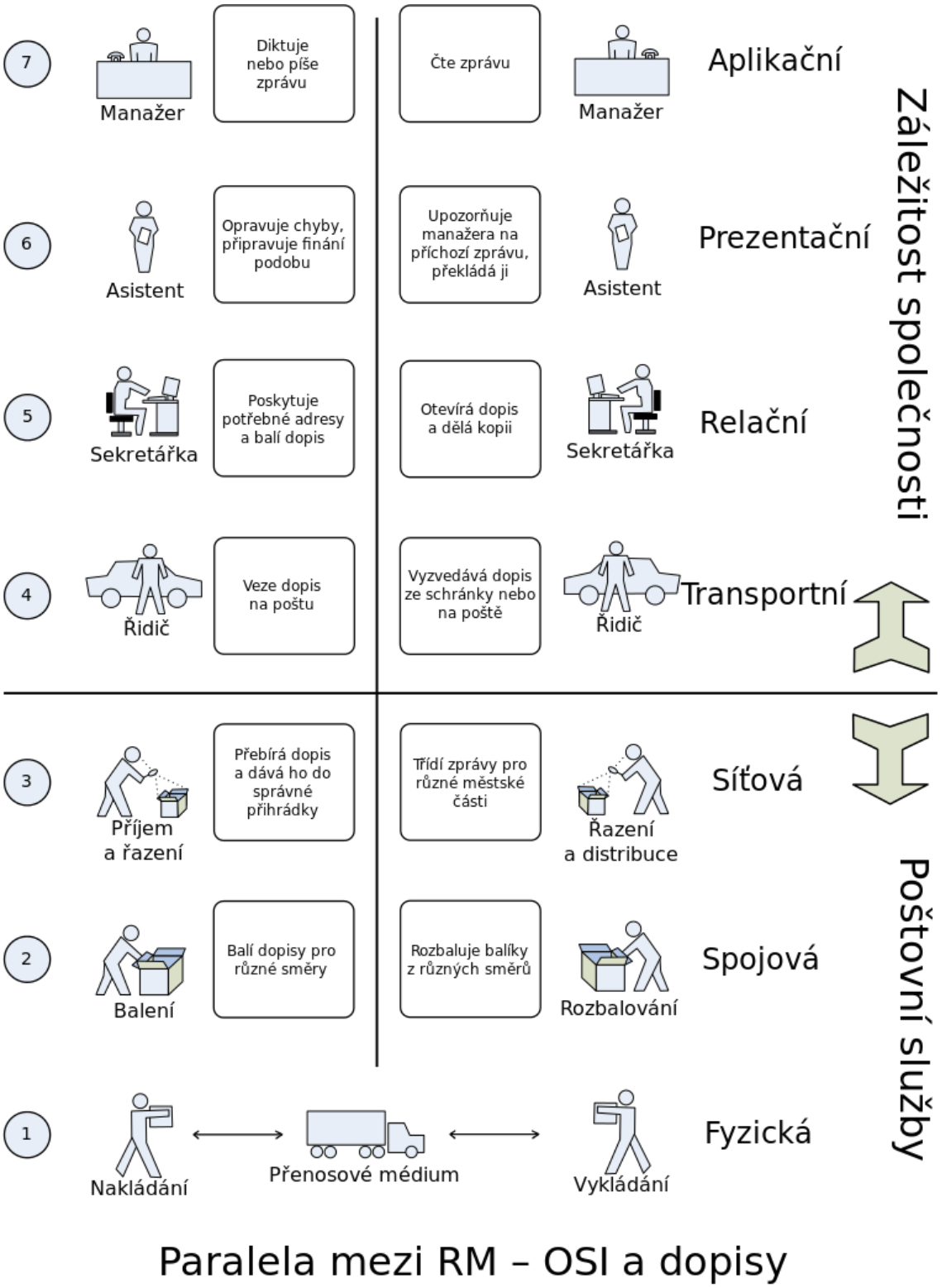
3.2 Popis komunikačních protokolů

Abychm mohli relevantně popsat komunikační protokoly a srozumitelně poukázat na jejich odlišnosti, musíme nejprve zavést vhodný model pro popis komunikačních rozhraní mezi počítači. Jedním z takovýchto vhodných modelů je referenční model ISO/OSI [15].

Referenční model ISO/OSI byl navržen za primárním účelem abstrakce sítové komunikace mezi počítači, přičemž hlavní myšlenkou bylo usnadnění standardizace norem definujících tuto komunikaci. Abstrakce modelu popisuje dělení komunikace do tzv. vrstev, které celý proces komunikace rozdělují na určité logicky ohraničené mezikroky. Každá vrstva může využívat služeb jiné, podřízené vrstvy v rámci zařízení – taková komunikace probíhá v rámci tzv. rozhraní. Vrstvy stejného druhu také mohou komunikovat mezi sebou v rámci různých spolu komunikujících zařízení. Taková komunikace probíhá pomocí tzv. protokolů. Protože nerozdělitelnou součástí modelu je mechanismus abstrakce a fyzické uskutečnování komunikace vyšších vrstev pomocí a prostřednictvím vrstev nižších, disponuje řešení implementované pomocí takového modelu značnou modularitou – umožnuje výměnu vrstvy nebo několika vrstev (a tím například přenosové technologie nebo celé části komunikace, která má co dočinění s fyzickou stránkou přenosu) bez nutnosti jakkoliv upravovat implementaci ostatních vrstev.

Převážná většina komunikačních protokolů, používaných v současné praxi pro komunikaci počítačů je navržena s ohledem na tento nebo podobné modely. Jednou s respektovaných klíčových vlastností modelu je například přísný důraz na modularitu a snadnost záměny jednotlivých vrstev. Jelikož v praxi se ukázalo, že pro dobře použitelnou a rozumně robustní implementaci by byla implementace všech vrstev modelu zbytečně komplikovaná, model počítá i s tzv. nulovými vrstvami, které informaci nijak nemění a v procesu komunikace se nijak neprojevují. Praktická implementace stejného efektu se zpravidla dosahuje vícenásobnými vrstvami, které vyhovují několika teoretickým modelovým vrstvám najednou.

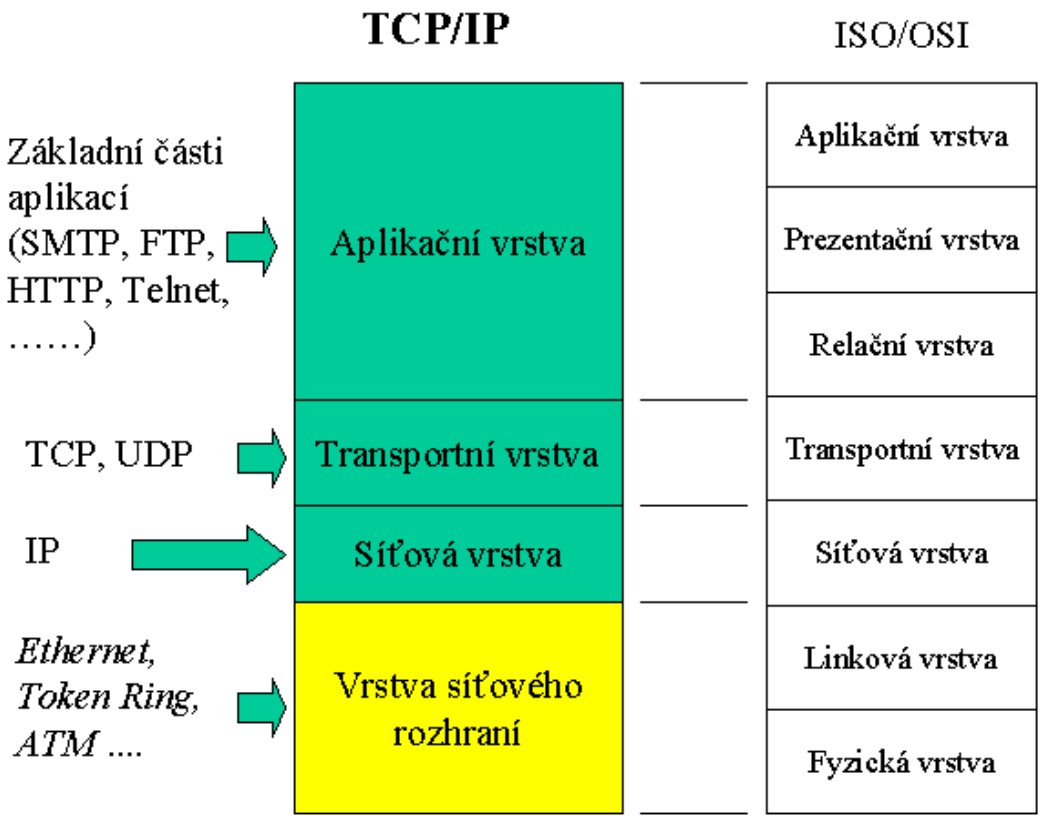
Základní dělení komunikace na vrstvy dle protokolu ISO/OSI spolu s výstižnou paralelou poštovní komunikace představuje obrázek 9:



Obr. 11 Popis protokolu ISO/OSI – paralela s poštovní komunikací. Zdroj: https://cs.wikipedia.org/wiki/Referen%C4%8Dn%C3%AD_model_ISO/OSI

Praktickou implementaci ISO/OSI protokolových vrstev do reálně implementovaných protokolů a rozhraní si můžeme přibížit na příkladu, se kterým se setkal každý člověk, který se kdy připojil na internet – TCP/IP stacku [16].

Kombinace IP a TCP protokolu, označovaná souhrnně jako TCP/IP stack, tvoří nejpoužívanější protokoly, používané pro přenos informací přes sít Internet. Obrázek 12 zobrazuje zařazení těchto protokolů podle ISO/OSI modelu:



Obr. 12 :TCP/IP stack dle modelu ISO/OSI. Zdroj: http://www.earchiv.cz

Protokol IP podle modelu ISO/OSI provádí a poskytuje funkce sítové vrstvy. Obsahuje tedy prostředky, jak může odesílatel dat v počítačové síti určit jejich příjemce a cestu k němu. Tato funkcionalita splnuje definici sítové vrstvy bez výrazných přesahů do jiných vrstev.

V rámci TCP/IP stacku běží nad IP protokolem protokol TCP. Tento protokol provádí a poskytuje funkci transportní vrstvy, tedy s pohledu aplikace je schopen určitý ucelený datový přenos přenést přes sít tak, aby jej bylo možné na druhé straně znovu zrekonstruovat, v ideálním případě bezvadně. Tato funkcionalita splnujě definici transportní vrstvy bez výrazných přesahů do jiných vrstev.

Co se ale ostatních vrstev modelu týče, přesné definice mizí. Nad TCP protokolem může běžet komunikace různými protokoly, které více či méně

obsahují funkcinalitu definovanou ve všech třech vrstvách nad transportní vrstvou ISO/OSI modelu. Zatím co například XML protokol [17] neobsahuje vůbec definici podoby prezentace dat, a tedy lze diskutovat o implementaci funkcionality prezentační vrstvy, protokol Telnet [18] pracuje zcela jistě v aplikační vrstvě.

Toto praktické zjednodušení teoretického modelu je jevem, se kterým se setkáme dále při popisu používaných komunikačních protokolů. Zatímco protokol OpenTherm je díky svému poměrně jednoznačnému zaměření poněkud monolitický a obsahuje definici funkcí ekvivalentních od fyzické až po aplikační vrstu, RS485 Modbus je ve skutečnosti stackem podobně jako TCP/IP, tedy kombinací dvou protokolů.

3.2.1 RS485 Modbus

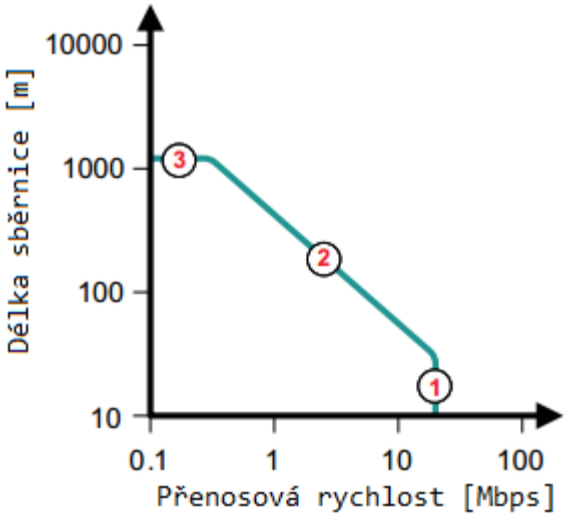
Jelikož RS485 Modbus není jediný monolitický protokol, ale stack, neboli spojení dvou protokolů v jednu kombinaci, je třeba nejprve popsat protokol nižší vrstvy, který je protokolem vyšší vrstvy využíván. Tímto protokolem nižší vrstvy je RS485.

RS485 [19] je asynchronní sériový protokol s diferenciálním elektrickým přenosem dat. Je to v definici dle ISO/OSI modelu protokol fyzické vrstvy, který nemá žádný přesah do vrstvy datové – jeho definice tedy neobsahuje ani předpis pro podobu vysílaných dat, pouze definici logických úrovní a dále pouze určitá doporučení ohledně spojové vrstvy.

Protokol RS485 patří do rodiny sériových protokolů spolu s protokoly RS422 a z prostředí osobních počítačů známějším RS232. Komunikace probíhá diferenciálně po dvou vodičích označovaných jako A a B, log. 1 nastává, je-li A kladnější než B, log. 0 nastává v opačném případě, tedy je-li B kladnější, než A. Nutné rozdílové napětí pro bezpečné rozpoznání log. úrovní je minimálně 200mV, nebo 1,5V na zátěži 54Ω.

Obr. 13 Minimální nutné diferenciální úrovně RS 485. Zdroj: [19]

Komunikace probíhá po kroucených párech vodičů, což narozdíl od sériových protokolů RS232 a RS422 výrazně zvyšuje maximální teoretickou délkou sběrnice. Vztah mezi maximální přenosovou rychlostí a délkou sběrnice popisuje obrázek 13:



Obr. 14 Vztah mezi maximální přenosovou rychlostí a délkou sběrnice protokolu RS485. Zdroj: [19]

V případě nejkratších délek sběrnice (jednotky metrů), kde nejužším hrdlem pro přenos dat je vlastní reakční doba budičů sběrnice, doporučuje specifikace přenosovu rychlost 10Mbps. Reálným obvodům se na tyto krátké vzdálenosti daří dosahovat přenosových rychlostí až 40Mbps. Naopak při využití nejnižších přenosových rychlostí v řádu desítek kbps lze dosáhnout úspěšného přenosu s délkami sběrnice dosahujícími až 1200m.

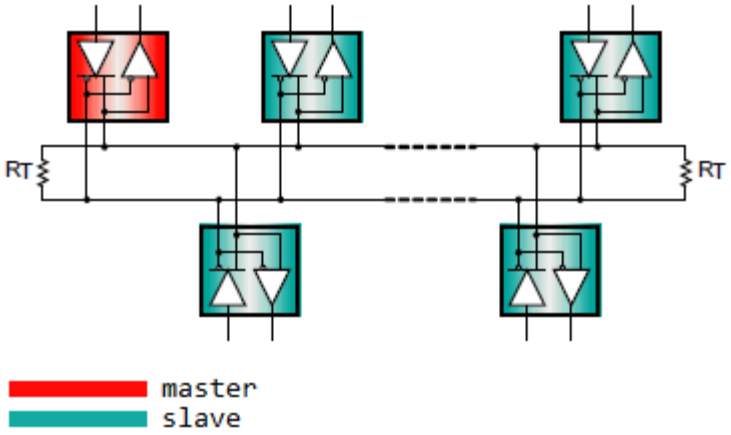
Zařízení na sběrnici se dělí na dvě kategorie. Master, hlavní a řídící zařízení, které se může vyskytovat na sběrnici pouze jednou a jedno nebo více podřízených Slave zařízení. Protokol RS485 podporuje point-to-multipoint komunikaci, její řešení, problém adresování zařízení apod. ale neřeší a přenechává jej na protokolu vyšší vrstvy.

Komunikace probíhá ve dvou variantách. Jednodušší, dvouvodičová, činí z protokolu RS485 poloduplexní komunikační prostředek. V tomto režimu jsou všechna zařízení připojena na jeden diferenciální kroucený pár vodičů, který tvoří komunikační sběrnici.
Komunikace v tomto režimu probíhá následovně:

- Pokud nikdo nevysílá, master vysílá na sběrnici požadavek (může obsahovat adresu slave, který má odpovědět)
- Slave přijímá požadavek a zpracovává odpověd
- Pokud nikdo nevysílá, slave vysílá odpoved

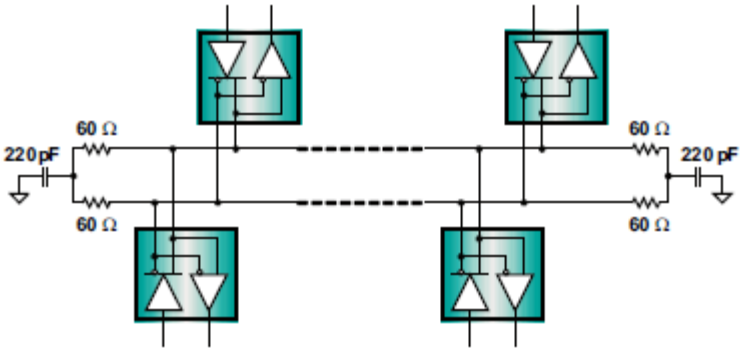
Pořadí kroků je v dokumentaci pevně definováno. Slave zařízení tedy nesmí vysílat bez předchozího dotazu od Master zařízení. Master zařízení vždy iniciuje komunikaci a tedy neodpovídá nikomu na požadavky. Je také doporučeno řešení případné kolize současně odpovídajících Slave zařízení pomocí náhodného

prodlužování času, po který Slave zařízení poslouchá sběrnici před započením vysílání odpovědi. Zapojení dvouvodičové varianty sběrnice popisuje obrázek 15:



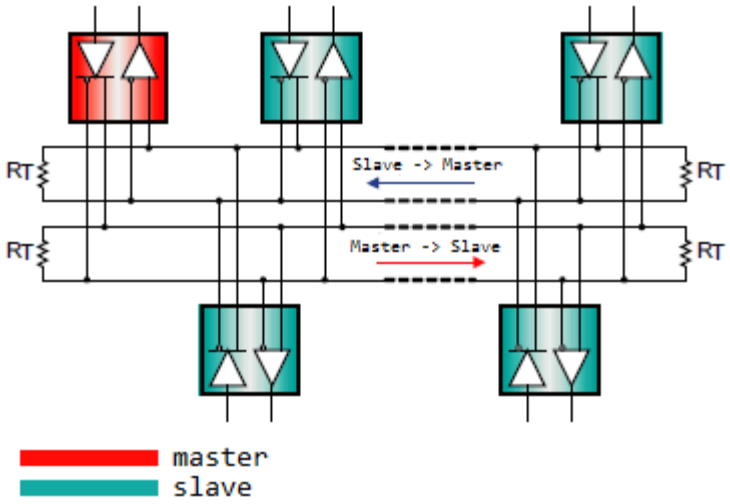
Obr. 15 Dvouvodičová varianta RS485. Zdroj: [19]

Z popisu sběrnice a z obrázku 15 je patrná nutnost kroucený pár vodičů ukončovat na obou koncích terminátorem. Díky diferenciální povaze napětové komunikace, při které napětové úrovně nejsou vztaženy k žádnému vnějšímu napětí, poměrně dlouhé sběrnici a možnosti relativně vysokých komunikačních rychlostí vzniká nezanedbatelný problém s odrazy na vedení. Tato situace je ještě umocněna tím, že v praxi prakticky nelze nikdy zajistit stejné vzdálenosti zařízení na sběrnici, ani stejné délky přívodů ze sběrnice k budiči. Doporučení ve specifikaci protokolu uvádí vhodnou impendanci kabelů a tedy i terminátorů R_T = 120Ω. Ve velmi zarušených prostředích dokonce specifikace doporučuje terminaci vedení jednoduchým filtrem typu dolní propust s impendancí rovněž 120Ω, viz obrázek 16:



Obr. 16 Dolnopropustné filtry ve funkci terminátoru sběrnice RS485 pro velmi zarušená prostředí. Zdroj: [19]

Druhá z používaných možných variant komunikace pomocí rozhraní RS485 je tzv. čtyřvodíčová. Každé zařízení v rámci této varianty má dva nezávislé budiče připojené každý na vlastní kroucený pár vodičů. Sběrnice je v této variantě plně duplexní. Zapojení čtyřvodičové varianty sběrnice zobrazuje obrázek 17:



Obr. 17 Čtyřvodičová varianta RS485. Zdroj: [19]

Z obrázku 17 je patrné, že jedna sběrnice se využívá pro komunikaci směrem od zařízení Master k zařízením Slave, druhá pro komunikaci opačným směrem. Komunikace tedy probíhá následovně:

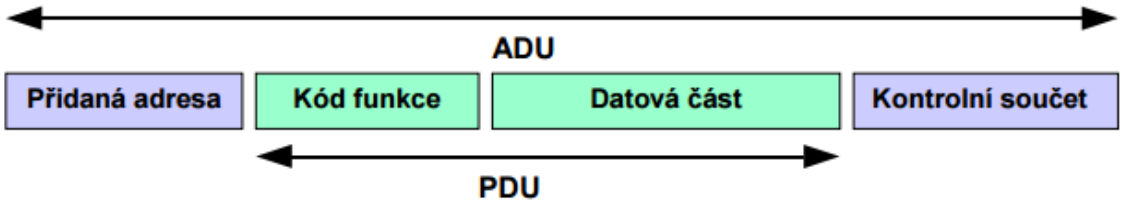
- Pokud nikdo nevysílá na I. sběrnici (na obr. 17 červená šipka), master vysílá na I. sběrnici požadavek (může obsahovat adresu slave, který má odpovědět)
- Slave přijímá požadavek přes rozhraní na I. sběrnici a zpracovává odpověd

- Pokud nikdo nevysílá na II. sběrnici (na obr.17 modrá šipka), slave vysílá na II. sběrnici odpoved

Čtyřvodičové zapojení rozhraní RS485 představuje i přes větší složitost lepší variantu hlavně při delších délkách sběrnice, kde pomáhá redukovat prodlevy a zvyšovat přenosovou rychlost. Naopak dvouvodičová varianta představuje variantu sice jednodušší a méně odolnou proti nepříznivým vnějším vlivům, ale mezi ostatními sériovými rozhraními typu RS232 a RS485 jde stále o variantu nejrobustnější, svými vlastnostmi velmi vhodnou do průmyslového prostředí. V rámci řešení převodníku bude využito dvouvodičové varianty sběrnice.

Protokol Modbus [20] tvoří s rozhraním RS485 takzvaný stack – je tedy navržen tak, že jeho specifická varianta přímo počítá s provozem nad rozhraním RS485. Protokol Modbus je protokolem aplikační vrstvy, i když v jeho definičních dokumentech nalezneme i prvky ostatních vrstev ISO/OSI modelu – definice protokolu například zahrnuje koncept adres a adresace zařízení na sběrnici, což je funkčním znakem vrstvy sítové. Lze tedy říci, že protokol Modbus implementuje funkce všech vrstev ISO/OSI modelu kromě vrstvy nejnižší, fyzické, kterou v našem případě implementuje rozhraní RS485. Naopak, samotné použití protokolu Modbus v rámci úlohy aplikační vrstvy může být poněkud diskutabilní, protože aplikační použití není v definičních dokumentech rozhraní Modbus pevně definováno. Definice v tomto ohledu končí zavedením datových struktur uvnitř jednotlivých zařízení, doporučení využití těchto struktur a zavedením typů vstupně výstupních operací nad těmito strukturami, ale na pevnou definici vlastního aplikačního použití ve stylu definic obsažených v dokumentaci protokolu OpenTherm v definičních dokumentech nenarazíme. Protokol je místo toho navržen tak, aby pomocí něj bylo možné univerzálně ovládat prakticky jakékoliv zařízení.

Datová zpráva protokolu Modbus se komunikuje ve formátu Big endian (tedy nejprve MSB, poté LSB) a dělí se na vnitřní a vnější část. Vnitřní část, tzv. Protocol Data Unit (PDU), obsahuje samotnou datovou komunikaci mezi zařízeními a její podoba je vždy stejná bez ohledu na rozhraní, po kterém komunikace probíhá. Vnější část, tzv. Application Data Unit (ADU), se liší v závislosti na rozhraní a obsluhuje spíše funkce sítové vrstvy. Grafické znázornění viz obrázek 18 :



Obr. 18 Datová zpráva protokolu Modbus. Zdroj: [20]

Datový model protokolu Modbus funguje na principu abstrahované datové struktury, kterou každé zařízení musí povinně implementovat. Tato datová

struktura se dělí na podoblasti, určené pro různá logická aplikační využití. Počítá se například s tím, že se bude komunikovat hodnota, která je na zařízení reálnou měřenou hodnotou – tedy bude z fyzikální podstaty věci jen pro čtení. Naopak, budou také jistě existovat hodnoty, které bude potřeba zapisovat i číst. Dále se počítá s tím, že se budou komunikovat delší datová slova, například ve významu určité číselné měřené hodnoty, také bude ale určitě v jistých situacích potřeba komunikovat jednobitovou diskrétní informaci, příznaky ve smyslu zapnout/vypnout, ano/ne. Na všechny tyto varianty je datový model Modbus připraven, viz obr.ázek 19:

Tabulka	Typ položky	Přístup	Popis
Diskrétní vstupy *(Discrete Inputs)*	1-bit	Pouze čtení	Data poskytovaná I/O systémem
Cívky *(Coils)*	1-bit	Čtení/zápis	Data modifikovatelná aplikačním programem
Vstupní registry *(Input Registers)*	16-bitové slovo	Pouze čtení	Data poskytovaná I/O systémem
Uchovávací registry *(Holding Registers)*	16-bitové slovo	Čtení/zápis	Data modifikovatelná aplikačním programem

Obr. 19 Datový model Modbus. Zdroj: [20]

S tímto datovým modelem je úzce spjata podoba PDU, jehož druhou část tvoří právě datová položka podle definice datového modelu. První část PDU obsahuje takzvaný kód funkce, což je část datové zprávy popisující požadovanou vstupně výstupní operaci nad datovou položkou. Jednotlivé kódy funkcí jsou jednoznačně určeny a kód funkce se konkrétně váže k typu dat podle datového modelu. Seznam a význam kódů funkcí viz obrázek 20:

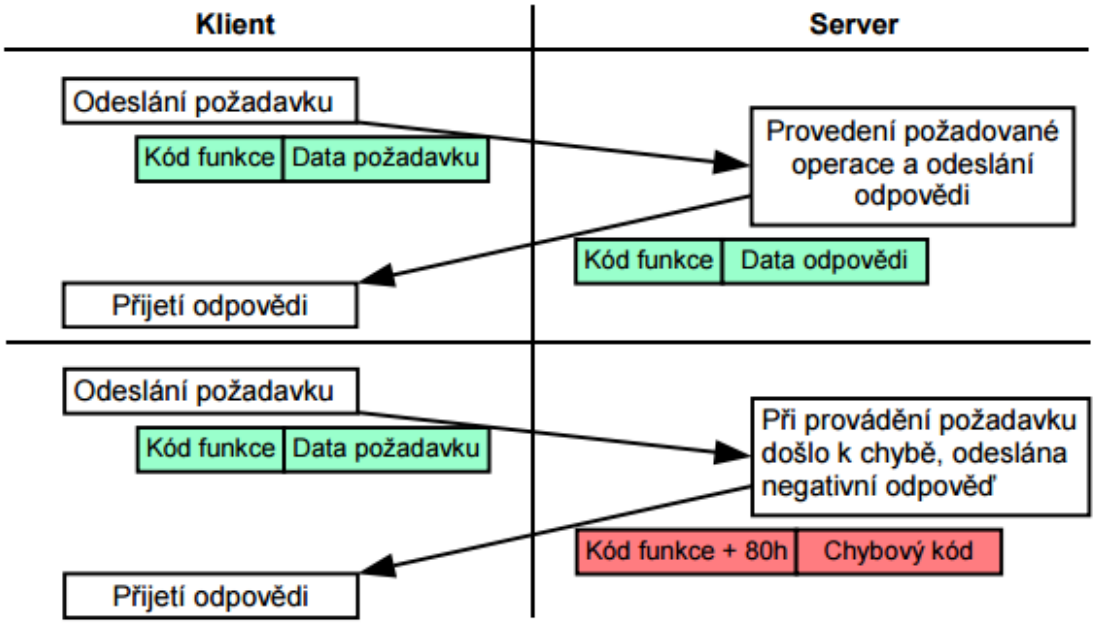
				Kódy funkcí			
				Kód	Podfunkce	hex	
Přístup k datům	Bitový přístup	Fyzické diskrétní vstupy	Čti diskrétní vstupy	02		02	
		Interní bity nebo fyzické cívky	Čti cívky	01		01	
			Zapiš jednu cívku	05		05	
			Zapiš více cívek	15		0F	
	16-bitový přístup	Fyzické vstupní registry	Čti vstupní registr	04		04	
		Interní registry nebo fyzické výstupní registry	Čti uchovávací registry	03		03	
			Zapiš jeden registr	06		06	
			Zapiš více registrů	16		10	
			Čti/zapiš více registrů	23		17	
			Zapiš registr s maskováním	22		16	
			Čti FIFO frontu	24		18	
	Přístup k záznamům v souborech		Čti záznam ze souboru	20	6	14	
			Zapiš záznam do souboru	21	6	15	
Diagnostika			Čti stav	07		07	
			Diagnostika	08	00-18, 20	08	
			Čti čítač kom. událostí	11		0B	
			Čti záznam kom. událostí	12		0C	
			Sděl identifikaci	17		11	
			Čti identifikaci zařízení	43	14	2B	
Ostatní			Zapouzdřený přenos	43	13, 14	2B	
			CANOpen základní odkaz	43	13	2B	

Obr. 20 Seznam kódů funkcí Modbus. Zdroj: [20]

Komunikace (viz obrázek 21) probíhá v módu klient (master) – server (slave). V dotazu klienta na server je tedy v PDU kód funkce podle obrázku 20 a datová část obsahuje data v souladu s kódem funkce – tedy například v případě kódu funkce pro čtení dat adresu segmentu datového modelu, který žádáme číst. V odpovědi serveru je v PDU kódu funkce stejný kód funkce jako byl v dotazu a v datové části samotná přečtená data ze segmentu datového modelu. V případě chybové odpovědi obsahuje PDU hodnotu kódu funkce +80h a v datové části chybový kód, který naznačuje povahu chyby. Adresa segmentu datového modelu může nabývat hodnot v rozsahu 0 – 65535, a to pro každou součást datového modelu zvlášť, můžeme tedy mít Civku s adresou 0 a vstupní registr s adresou 0 a tyto dvě hodnoty se nebudou v paměti překrývat. Celkem je tedy možné mít 65536 * 4 různých unikátních segmentů – datových položek datového modelu. Způsob adresování segmentu datového modelu podléhá zvyklosti, že adresový prostor by měl být rozdělen na části podle typu dat, tedy je doporučeno adresovat segmenty datového modelu tak, aby se adresy nepřekrývaly. Toto je ale pouze doporučení, autor implementace může zvolit libovolné adresování segmentů podle toho, jak uzná za vhodné.

Datová část zprávy má proměnnou délku která je s ohledem na zpětnou kompatibilitu s protokoly fyzické vrstvy maximálně 252 bytů nebo 504 znaků na rozhraní RS485. Reprezentace dat se liší podle kódu funkce. Některé kódy funkcí

operují ještě s takzvanými podfunkcemi, jejichž kódy se umistují za kód funkce a o délku kódu podfunkce je potom zkrácena délka vlastních dat.



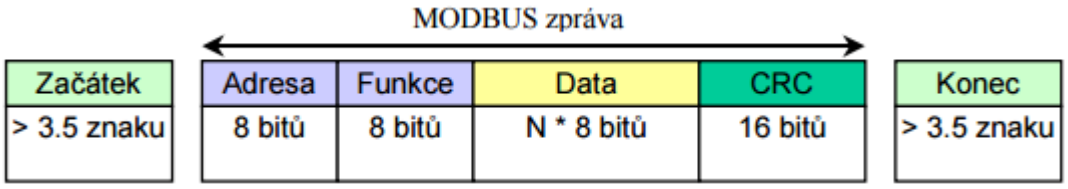
Obr. 21 Komunikace Klent-Server pomocí protokolu Modbus. Zdroj: [20]

Vnější vrstva, ADU, obsahuje vnitřní PDU jako svůj payload a její podoba se liší v závislosti na protokolu fyzické vrstvy, který je použit pro přenos na fyzické vrstvě. V našem případě, kdy je použit protokol RS485, se ADU skládá z adresy zařízení, payloadu (PDU) a kontrolního součtu.

První část ADU, adresa zařízení, může nabývat hodnot 1 až 247 pro jednotlivá Server (Slave) zařízení, což se používá v point-to-point komunikaci. Protokol počítá i s point-to-multipoint komunikací, proto je adresa 0 vyhrazena pro broadcast. Na této adrese poslouchají všechna Server (Slave) zařízení. Adresy 247 až 255 jsou rezervovány pro pozdější využití. Master (Klient) zařízení může být na sběrnici jenom jedno a vždy zahajuje komunikaci, proto vůbec nepotřebuje a nemá vlastní adresu.

Protokol Modbus podporuje v souladu s tradicí ostatních sériových protokolů RS232 a RS422 dva různé režimy přenosu. První režim, Modbus RTU, je režimem nativním, každé zařízení na sběrnici Modbus musí tento režim podporovat. V tomto režimu probíhá komunikace jako čistě datová – délky položek v datové zprávě se počítají na byty, každý byte vnitřně obsahuje dva nibbly dat vyjádřitelné ve formě dvou hexadecimálních znaků s pořadím MSB, LSB. Přenášený znak se standartně komunikuje ve tvaru 1 start bit + 8 datových bitů + 1 paritní bit (standartně je užita sudá parita) + 1 stop bit, celkem tedy 11 bitů i s režií. Mezery mezi těmito jedenáctibitovými znaky nesmí být delší než 1,5 vysílací délky znaku, jinak je komunikace zahozena jako neplatná. Začátek a konec datové zprávy se

rozpoznává jako přerušení komunikace na dobu delší, než 3,5 vysílací délky znaku, viz obrázku 22 :



Obr. 22 Datová zpráva MODBUS RTU. Zdroj: [20]

Poslední část ADU, kontrolní součet, se v případě Modbus RTU počítá pomocí algoritmu CRC s generujícím polynomem $x^{16} + x^{15} + x^2 + 1$. Cyklický redundantní součet se vypočítá z celé ADU (tedy adresa+kód funkce+PDU), ovšem počítá se s dat bez komunikační režie, tedy ne z 11tibitového, ale vždy z 8mibitového znaku.

Druhý komunikační režim, Modbus ASCII, implementuje komunikaci jako znakovou, což je v oblasti sériových komunikačních protokolů běžný a tradičně používaný režim. Zde se délky položek v datové zprávě počítají na ASCII znaky, každých 8 bitů dat se vysílá jako dva ASCII znaky. Formát jednoho přenášeného znaku je 1 start bit + 7 datových bitů + 1 paritní bit (standartně sudá parita) + 1 stop bit, celkem tedy 10bitů. Mezi jednotlivými přenášenými znaky může být časová mezera až 1s, začátek a konec přenosu zprávy se nerozpoznává na základě relativního časování. Začátek přenosu zprávy je vždy uvozen ASCII znakem „:“ (dvojtečka, ASCII 3Ah), přenos zprávy se ukončuje sekvencí ASCII znaků CR+LF (0Dh, ASCII 0Ah). Maximální délka datové části zprávy činí 504 (2*252) znaků, viz obrázek 23:

Začátek	Adresa	Funkce	Data	LRC	Konec
znak „:“	2 znaky	2 znaky	0 až 2*252 znaků	2 znaky	2 znaky CR, LF

Obr. 23 Datová zpráva Modbus ASCII. Zdroj: [20]

Kontrolní součet a mechanizmus jeho výpočtu se radikálně liší od řešení použitého v Modbus RTU. Používá se mnohem implementačně snažší LRC, který v podstatě vrací kontrolní součet rovný počtu bitů v log. 1. LRC se podobně jako CRC v Modbus RTU počítá z kompletní ADU, ovšem bez začátečního znaku „:“ (dvojtečka).

V implementaci firmware převodníku bude použita nativní varianta komunikace Modbus RTU.

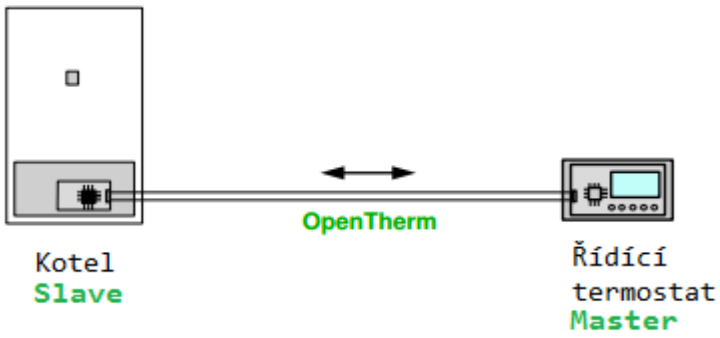
3.2.2 Opentherm

OpenTherm je otevřený protokol [21] určený pro komunikaci v prostředí digitálně řízených systémů pro vytápění a klimatizaci. Podobu protokolu určuje a jeho použití zastřešuje The OpenTherm Association [5] se sídlem v Nizozemském Zoetermeeru.

Obr. 24 Logo asociace OpenTherm. Zdroj: [5]

Asociace OpenTherm v sobě sdružuje výrobce topné a klimatizační techniky, kterým členství v asociaci přináší možnost podílet se na úpravách specifikace protokolu podle poznatků získaných jeho praktickým nasazením. Protokol je ale otevřený – může ho do svého zařízení nasadit kdokoliv, tedy i nečlen asociace OpenTherm.

Protokol je navržený na komunikaci point-to-point s jedním Master a jedním Slave zařízením. Roli Master zařízení vždy zastává řídící systém, inteligentní termostat apod. Roli Slave zařízení vždy zastává ovládáný systém pro vytápění nebo klimatizaci.

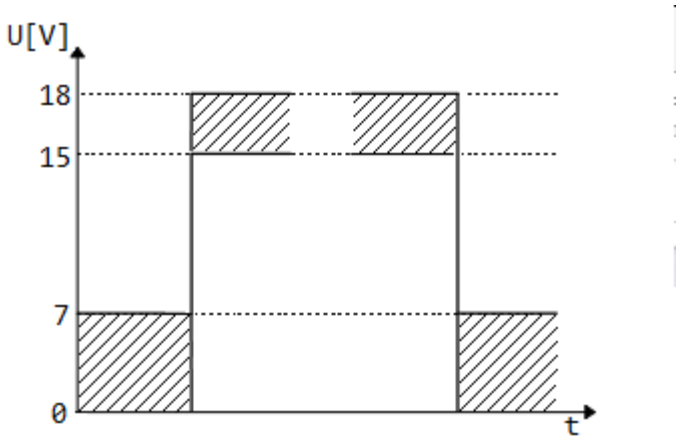


Obr. 25 Typické nasazení protokolu OpenTherm. Zdroj: [21]

Protokol OpenTherm je monolitický ve smyslu definice obsluhy vrstev přes celý rozsah ISO/OSI modelu. Protokol obsahuje definici fyzické vrstvy, dále definici datalinkové vrstvy, kterou lze v rámci ISO/OSI modelu chápat jako kombinaci vrstvy sítové s vrstvou spojovou a konečně definici vrstvy aplikační.

Fyzická vrstva protokolu OpenTherm je navržena jako zpětně kompatibilní se staršími systémy ovládání systémů pro vytápění, tento starý systém je dokonce v protokolové specifikaci dodefinován jako varianta protokolu OpenTherm/Lite (OT/-). Komunikace probíhá po jednom nekrouceném páru vodičů. Jednodušší varianta protokolu OpenTherm/Lite počítá s jednosměrnou komunikací signálem s pulzně šířkovou modulací, kde se celé řízení redukuje na přenos jediné informace: požadovaný topný (chladící) výkon: 0 pro signál se střídou 0%, požadovaný topný (chladící) výkon: maximální pro signál se střídou 100%. Varianta rozhraní OpenTherm/Lite je zaměřena spíše na využití v jednodušších systémech, které vůbec nemusí obsahovat digitální řídící prvky.

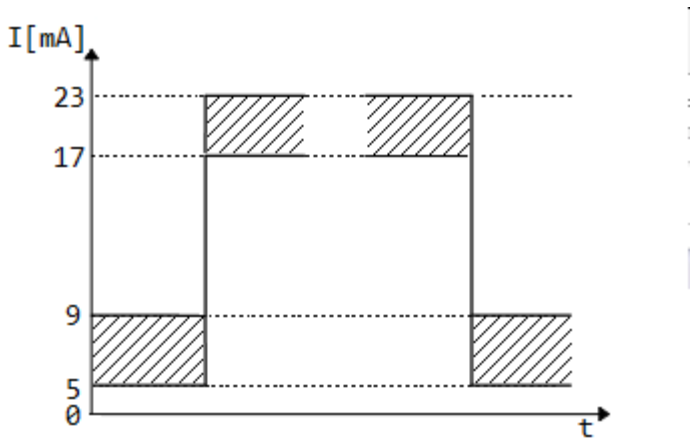
Plnohodnotná variata OpenTherm protokolu, OpenTherm/Plus (OT/+), pracuje na fyzické vrstvě z důvodu zpětné kompatibility také s jediným nekrouceným párem vodičů. Komunikace je poloduplexní, z důvodu nutnosti vystačit si s jediným párem vodičů je použito netradiční řešení: komunikace směrem Master → Slave probíhá napětově, log. 0 je definována jako napětí 0 – 7V, log. 1 je definována jako napětí 15 – 18V.



Obr. 26 Napětové úrovně komunikace Master → Slave na rozhraní OpenTherm.. Zdroj: [21]

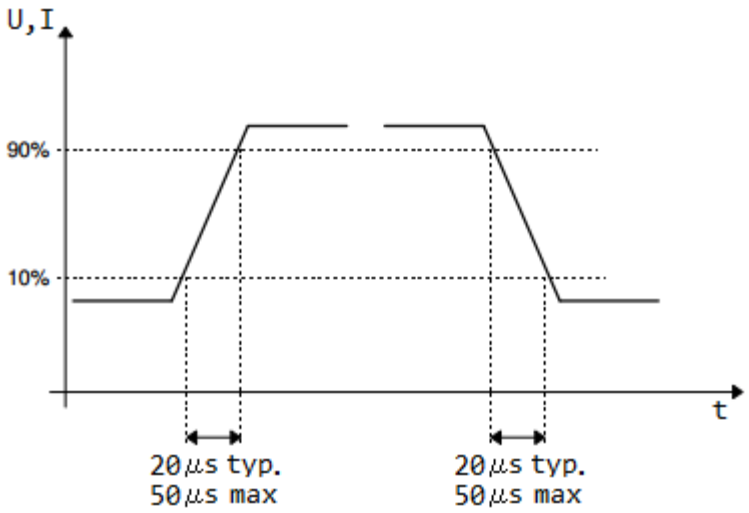
Specifikace rozhraní OpenTherm také počítá s napájením inteligentního řídícího systému (tedy Master strany) přímo po komunikační lince. S tím souvisí i definice klidového stavu, tedy maximálních povolených napětí a proudů v log.0 tak, aby bylo v tomto stavu možné inteligentní řídící člen napájet.

Komunikace Slave → Master probíhá v režimu proudové smyčky (viz obrázek 27), log. 0 je tedy definována jako 5 – 9mA, log. 1 je definována jako 17 – 23mA. Situace je ještě komplikovanější díky tomu, že ve specifikaci je uvedeno, že na polaritě zapojení vodičů nezáleží - vstupní a výstupní obvody rozhraní tedy musí reagovat na napětí a proudy v kladném i záporném směru. Specifikace uvádí maximální doporučenou délku vedení 50m, maximální odpor páru vodičů 2*5Ω a pro velmi elektromagneticky zarušená prostředí doporučuje nahradit nekroucený pár vodičů párem krouceným nebo vodiče stínit.



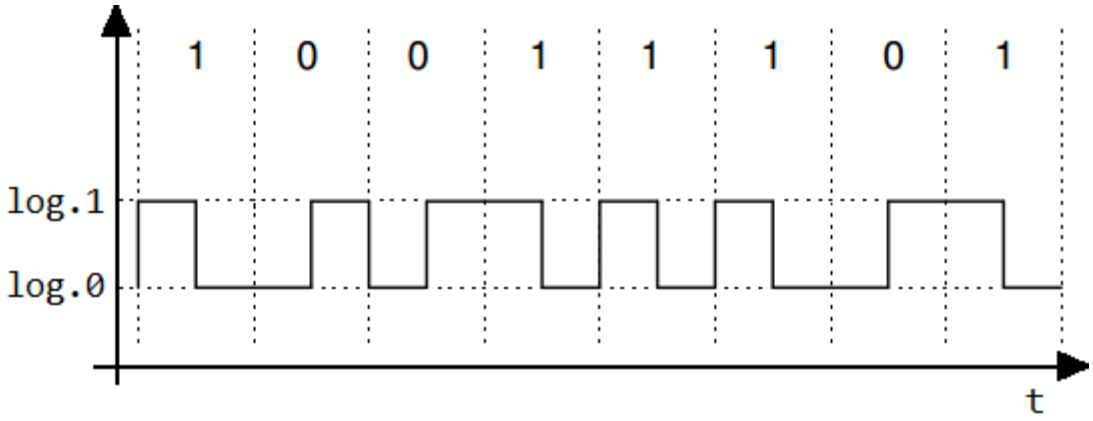
Obr. 27 Proudové úrovně komunikace Slave → Master rozhraní OpenTherm. Zdroj: [21]

Narozdíl od RS485 je v definici fyzické vrstvy protokolu OpenTherm pevně definováno i časování. U proudové i napětové komunikace je rise time i fall time definován shodně jako 20µs typicky, 50µs maximálně, viz obrázek 28.:



Obr. 28 Časování komunikace OpenTherm. Zdroj: [21]

Komunikace po rozhraní OpenTherm/Plus využívá na fyzické vrstě kódování typu Manchester. V tomto kódování jsou bity na vedení definovány ne jako samotné logické úrovně, ale jako přechody mezi úrovněmi. V případě protokolu OpenTherm je bit s hodnotou 0 definován jako přechod z log. 0 do log. 1 na vedení. Analogicky, bit s hodnotou 1 je definován jako přechod z log. 1 do log. 0 na vedení.



Obr. 29 Příklad datového přenosu OpenTherm v kódování Manchester. Zdroj: [21]

Kódování Manchester zajišťuje několik důležitých funkcí. Při přenosu bez využití kódování nastává problém při přenosu velkých shluků dat stejné logické hodnoty ke strátě synchronizace, protože logická úroven se nemění a časování není z čeho korigovat. Kódování Manchester tento problém obchází, protože na jeden bit signálu vždy připadá minimálně jeden přechod mezi logickými úrovněmi. Nejhorší možný případ nastane když přenášíme alternující sekvenci bitů (...1-0-1-0-1-0…) a i tehdy stále odpovídá jeden přechod mezi logickými úrovněmi jednomu bitu – synchronizace spojení tedy nemůže být narušena nevhodnou podobou přenášených dat.

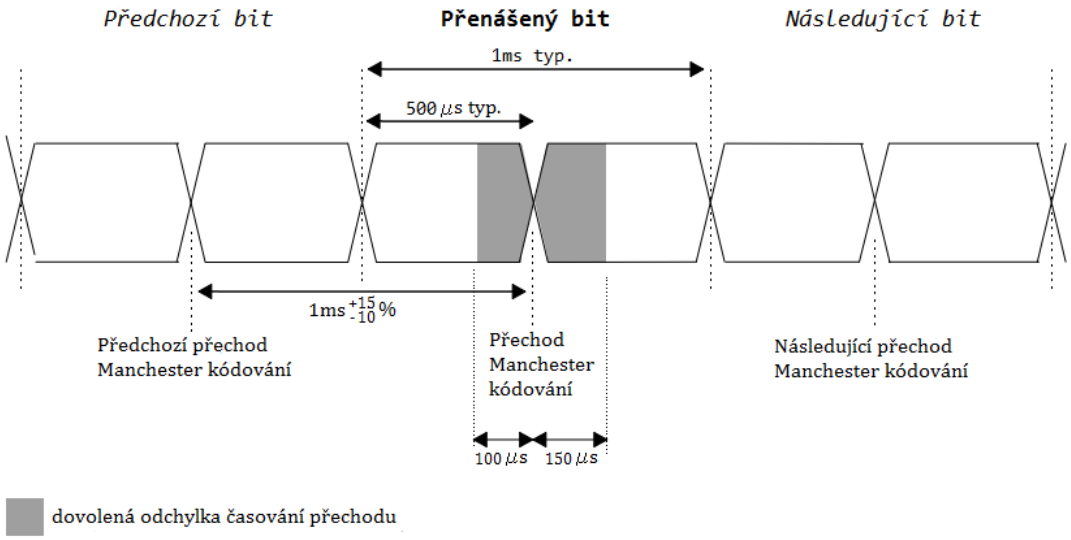
Další důležitou funkcí je detekce přerušení přenosu. Při přenosu bez využití kódování není přijímací strana schopna určit, zda déletrvající úroven log.0 znamená opravdu přenos shluku bitů s log.0, nebo přerušení přenosu a výpadek komunikace. Díky neustálým přechodům mezi logickými úrovněmi na aktivním přenosovém kanále s využitím kódování Manchester může přijímací strana detekovat výpadek velmi spolehlivě už po době, rovnající se době trvání přenosu jednoho bitu.

Z obrázku 29 je také zřejmé, že poměr fyzicky přítomných úrovní log. 0 a log. 1 na vedení je v limitě stále stejný bez ohledu na data, což v porovnání s vysíláním bez využití kódování efektivně zvyšuje množství energie přenášené po vedení. Tato vlastnost usnadnuje připadné napájení Master zařízení přímo pomocí rozhraní OpenTherm.

Určitou nevýhodu spojenou s nasazením kódování Manchester představuje skutečnost, že signál na vedení běží z vyšší taktovací frekvencí (frekvenci fyzických změn logických úrovní na vedení), než přenosovou (frekvencí přenosu dat). V nejhorším případě přenosu shluku bitů stejné hodnoty je taktovací frekvence dvojnásobkem frekvence přenosové. V nejlepším případě přenosu alternující sekvence bitů (...1-0-1-0-1-0-1…) je taktovací frekvence rovna frekvenci přenosové. V případě reálných dat je tedy taktovací frekvence rovna nebo vyšší, než frekvence přenosová, což sebou nese negativní dopady na potřebu kvalitnějšího přijímacího a vysílacího hardwaru a kabeláže, která je schopna si s vyšší taktovací frekvencí poradit. Vzhledem k poměrně nízké přenosové rychlosti

rozhraní OpenTherm lze ale říci, že tento problém se na rozhraní OpenTherm v praxi příliš neprojevuje

Vlastní časování přenosu datové zprávy je zobrazeno na obrázku 30.

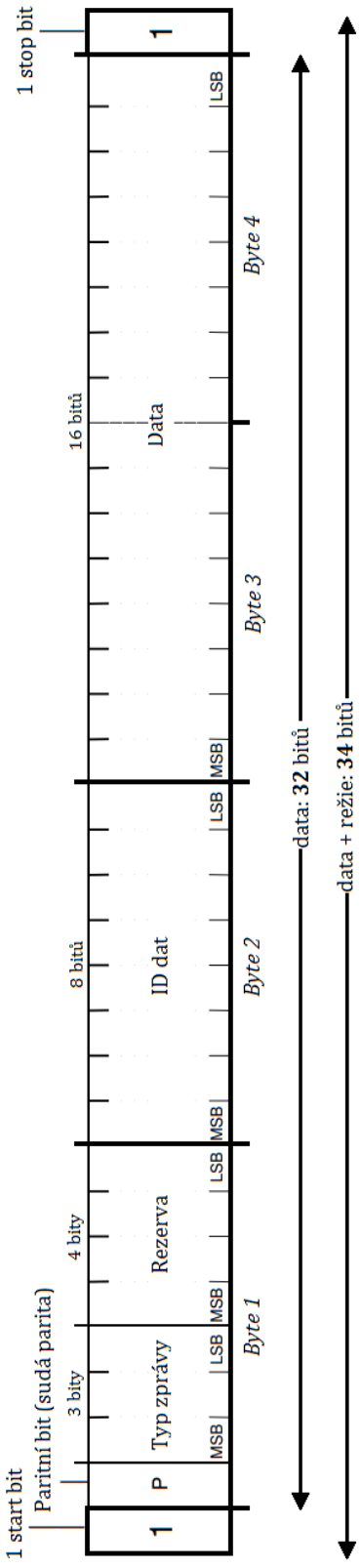


Obr. 30 Časování fyzické vrstvy protokolu OpenTherm/Plus. Zdroj: [21]

Přenosová rychlost rozhraní OpenTherm/Plus je stanovena na 1000bps. Z toho lze odvodit, že přenos jednoho bitu trvá typicky 1 ms. Kvůli využití Manchester kódování je tato doba rozdělena v ideálním případě přesně na poloviny přechodem Manchester kódování, každá z polovin trvá v ideálním případě 500µs. Specifikace protokolu OpentTherm/Plus určuje maximální ještě přípustnou chybu časování Manchester přechodu jako -100µs až +150µs. Každá z polovin přenášeného bitu tedy může v praxi trvat od 400 do 650µs. Specifikace protokolu OpenTherm/Plus doporučuje při implementaci dekodéru na přijímací straně rozhraní neudržovat pokud možno vůbec žádné interní nezávislé časování ke kterému by se intervaly úrovní na vedení měly vztahovat, ale při každé detekované změně stavu linky vynulovat časovač a počítat zmíněných 400 až 650µs, případně dvojnásobných 800 až 1300µs vždy od začátku s tím, že změny mimo tato časová okna se ignorují. Tímto spůsobem nedojde tak snadno k výpadku časové synchronizace komunikace v případě, že je časování nestabilní.

Protokol OpenTherm narozdíl od RS485 Modbus neimplementuje žádné pokročilé kontrolní mechanizmy pro ověření správnosti přijatých dat. Dokumentace k tomuto uvádí, že základním kontrolním mechanizmem správnosti přijatých dat na bitové úrovni je právě nasazení Manchester kódování.

Specifikace protokolu OpenTherm zahrnuje i součásti definicí podle sítové vrstvy protokolu ISO/OSI. Veškerá komunikace protokolem OpenTherm probíhá z hlediska sítové vrstvy prostřednictvím datových zpráv ve tvaru viz obrázek 31:



Obr. 31 Datová zpráva protokolu OpenTherm. Zdroj: [21]

Datová zpráva protokolu OpenTherm začíná start bitem, který má vždy hodnotu log. 1. Start bit se spolu se stop bitem nezapočítává do celkové délky zprávy, která činí 32 bitů. Celkový objem fyzicky přenášených dat po vedení v rámci jedné datové zprávy je vždy 34 bitů.

První část datové zprávy, která se započítává do délky 32 bitů, je paritní bit. Protokol OpenTherm používá sudou paritu, parita se počítá z celé délky datové zprávy mimo start a stop bity.

Druhou částí datové zptávy je tříbitové pole Typ zprávy. Možných typů zprávy je celkem 8, viz tab. 1:

Tab.1: Typy datové zprávy protokolu OpenTherm:

Hodnota pole Typ zprávy	Směr komunikace	Význam
000	Master→ Slave	Čti data
001	Master→ Slave	Zapiš data
010	Master→ Slave	Neplatná data
011	rezerva	rezerva
100	Slave → Master	Potvrzuji čtení dat
101	Slave → Master	Potvrzuji zápis dat
110	Slave → Master	Neplatná data
111	Slave → Master	Neznámé ID dat

Význam hodnot v poli Typ zprávy úzce souvisí s formátem konverzace podle specifikace sítové vrstvy protokolu. V prvním kroku konverzace posílá zařízení Master zprávu zařízení Slave s jedním ze tří možných typů zprávy:

První typ, „Čti data“, je použit pokud jsou dále ve zprávě data s ID které je ve specifikaci aplikační vrstvy uvedeno jako pro čtení, tedy data která má zařízení Slave poskytnout zařízení Master. Na tuto zprávu posílá zařízení Slave odpověd s typem zprávy „Potvrzuji čtení dat“ pokud vše na straně zařízení Slave proběhlo bez chyb, Pokud je na zařízení Slave známo, že vracená data nejsou platná, zařízení slave odpovídá typem zprávy „Neplatná data“ (hodnota 110b). Pokud zařízení Slave vůbec nepodporuje ID dat v dotazu od zařízení Master, odpovídá typem zprávy „Neznámé ID dat“.

Druhý typ, „Zapiš data“, je použit pokud jsou dále ve zprávě data s ID, které je ve specifikaci aplikační vrstvy uvedeno jako pro zápis, tedy data, která zařízení Master zapisuje do zařízení Slave. Následné odpovědi zařízení Slave jsou analogické s typem zprávy „Čti data“ s tím rozdílem, že v případě kdy nenastane chybový stav, zařízení Slave odpovídá typem zprávy „Potvrzuji zápis dat“.

Třetí a poslední typ, „Neplatná data“ (s hodnotou 010b), se použije, pokud je na zařízení Master známo, že data odesílaná ve zprávě jsou neplatná. Podle specifikace OpenTherm/Plus má být použití tohoto typu zprávy upozorněním pro

zařízení Slave, že má data ve zprávě ignorovat. Zařízení Slave na tento typ zprávy odpovídá typem zprávy „Neplatná data“ (s hodnotou 110), nebo v případě nepodpory ID dat ve zprávě typem zprávy „Neznámé ID dat“.

Třetí částí datové zprávy je čtyřbitová položka, která se v současné verzi protokolu OpenTherm/Plus nepoužívá. Specifikace doporučuje, aby hodnota této položky byla 0000b za všech okolností.

Čtvrtá část datové zprávy je osmibitové pole ID dat. Toto pole určuje význam i datový typ následujícího pole Data. Pole ID dat může nabývat hodnot 0 až 255, přičemž významy hodnot 0 až 127 jsou pevně určeny tabulkou v definici aplikační vrstvy protokolu OpenTherm. Hodnoty 128 až 255 mohou být využity volně podle vlastních definic výrobců zařízení (členů The OpenTherm Association), ovšem pouze pro diagnostické a testovací účely, nesmí jich být v žádném případě využito pro provozní ovládání. Definiční tabulka kromě významu každého ID definuje i příslušný datový typ pole Data a to, jestli jsou data pro čtení, nebo pro zápis a tedy jaký typ zprávy je při komunikaci těchto dat třeba využít.

Pro splnění formální podpory protokolu OpenTherm/Plus stačí výrobci zařízení implementovat pouze šest unikátních ID, v případě Master zařízení stačí dokonce pouze tři, viz tab. 2:

Tab. 2: Minimální implementace protokolu OpenTherm:

ID dat	Název	Popis	Čtení/zápis
0	Master a Slave - stav	Dvě osmibitová pole stavových příznaků, jedno pro Master a jedno pro Slave zařízení	Čtení
1	Požadovaná hodnota	Požadovaná teplota ve stupních Celsia (typicky 0 – 100 °C)	Zápis
3	Slave - konfigurace	Osmibitové pole konfiguračních příznaků, osmibitová hodnota MemberID (indikace výrobce)	Čtení
14	Maximální relativní modulace	Pro systémy více spřažených Slave zařízení – nastavení stropu citlivost na ovládání (0 – 100%). Povinné pouze pro zařízení Slave.	Zápis
17	Relativní modulace	Aktuální relativní výkon zařízení (0 – 100%). Povinné pouze pro zařízení Slave.	Čtení
25	Teplota topné vody	Teplota topné vody na výstupním vedení z kotle (-40 – 127 °C). Povinné pouze pro zařízení Slave.	Čtení

Pátá část datové zprávy je šestnáctibitové pole Data. Zde se v datové zprávě přenáší samotná užitečná informace, jejíž význam a použitý datový typ udává pole ID dat. V závislosti na ID dat může být v poli Data informace následujících datových typů:

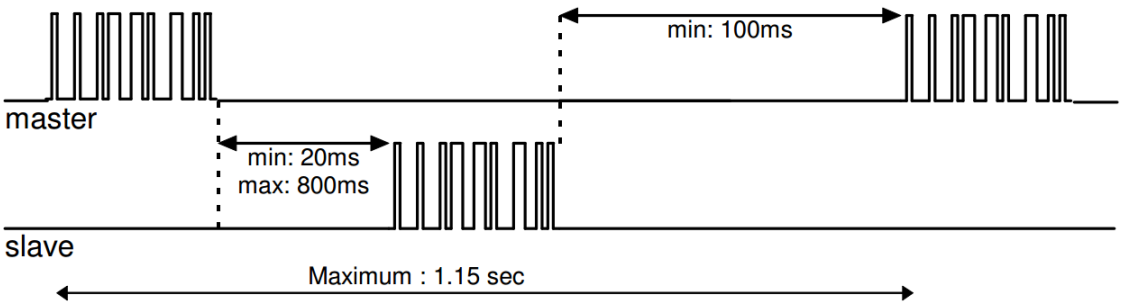
- flag8 - osmibitový datový typ, sestávající z osmi jednobitových příznaků.
- u8 – osmibitový datový typ, celé číslo bez znaménka. Nabývá hodnot 0 – 255.
- s8 – osmibitový datový typ, celé číslo se znaménkem. Nabývá hodnot -127 až 127.

Všechny osmibitové datové typy mohou být v jedné datové zprávě doplněny stejným, nebo jiným osmibitovým datovým typem.

Datové typy šestnáctibitové:

- u16 – šestnáctibitový datový typ, celé číslo bez znaménka. Nabývá hodnot 0 – 65535.
- s16 – šestnáctibitový datový typ, celé číslo se znaménkem. Nabývá hodnot -32768 až 32767.
- f8.8 – šestnáctibitový datový typ, desetinné číslo s pevnou desetinnou čárkou ve formátu 1bit znaménko, 7 bitů celého čísla, 8 bitů desetinného čísla (rozlišení 1/256). Povolený rozsah hodnot dle specifikace činí -40,0 až 127,0.

Protokol OpenTherm ve své specifikaci jasně definuje i časování konverzací (viz obrázek 32). Konverzace se vždy skládá ze dvou částí. V první části je odeslána jedna datová zpráva zařízením Master. Na tuto zprávu odpovídá v druhé části konverzace svojí datovou zprávou zařízení Slave. Odpověd od zařízení Slave musí přijít v rozmezí 20 až 800ms od ukončení vysílání zprávy zařízením Master. Mezi koncem jedné konverzace a začátkem další musí uplynout minimálně 100ms, maximálně 1,15 sekundy.

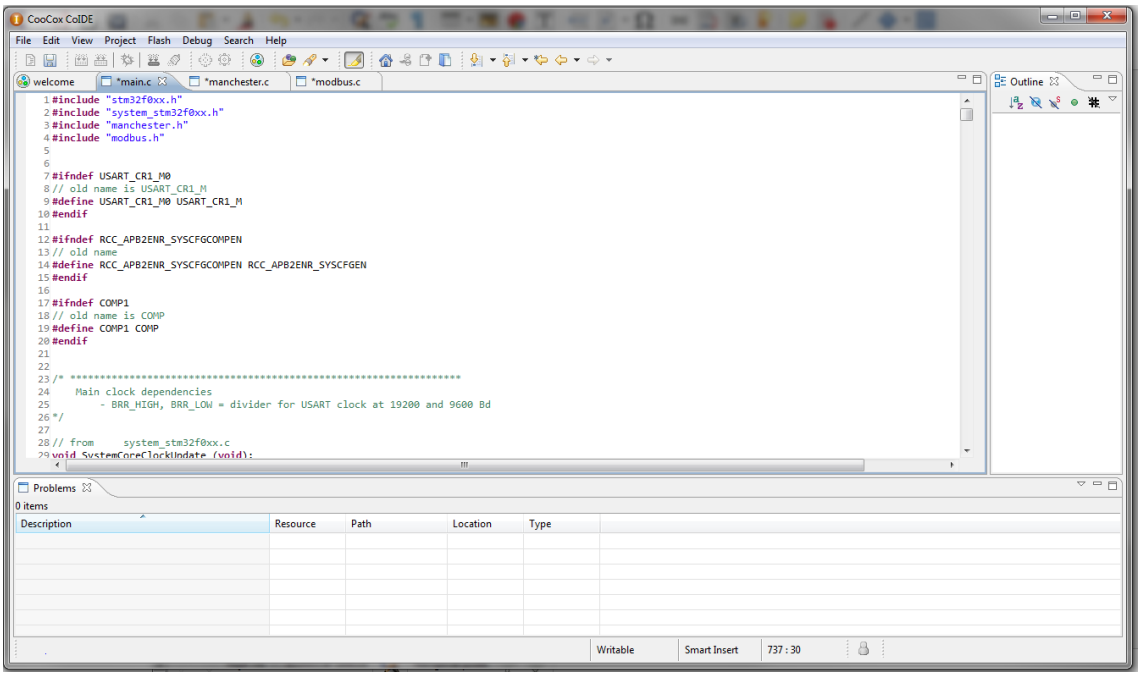


Obr. 32 Časování konverzace protokolu OpenTherm. Zdroj: [21]

Pokud zařízení Master nekomunikuje po delší dobu než 1,15 sekundy, nebo pokud zařízení Slave neodpovídá po delší dobu než 800ms, stav se dle dokumentace posuzuje jako výpadek komunikace.

3.3 Vývojové prostředí CooCox CoIDE

Pro naprogramování firmwaru bylo využito vývojové prostředí CooCox CoIDE [22]. Je to vývojové prostředí určené pro jazyk C a primárně zaměřené na vývoj na platformě ARM.



Obr. 33 Vývojové prostředí CooCox CoIDE.

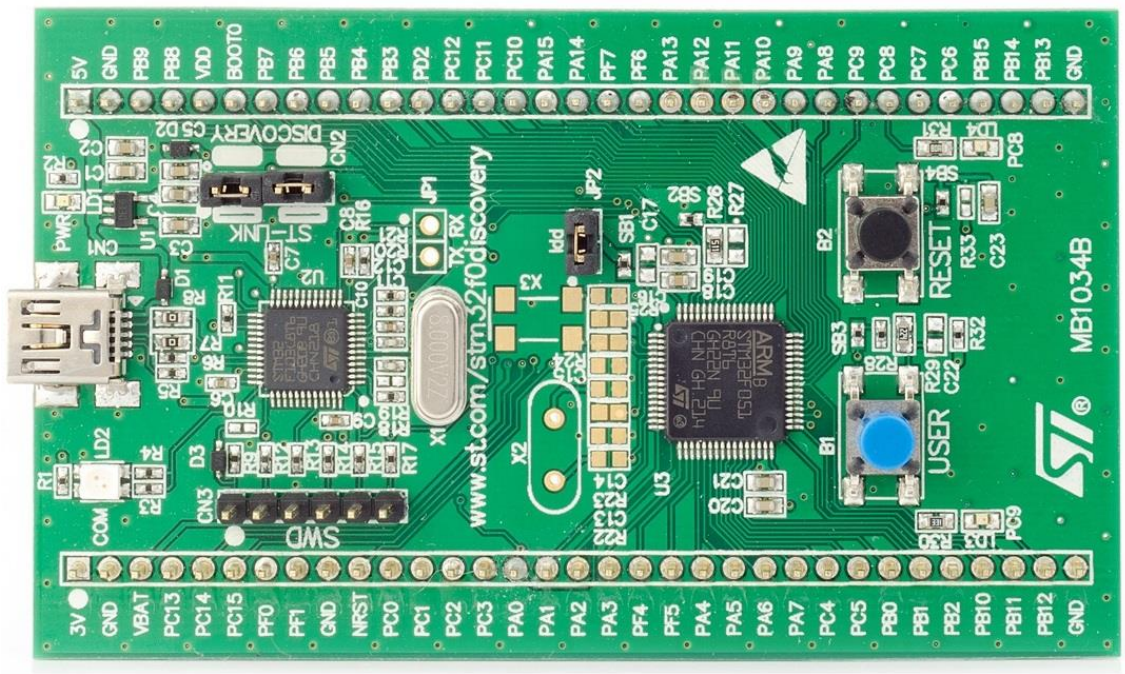
Vývojové prostředí CooCox CoIDE vychází z populárního vývojového prostředí Eclipse a podporuje celou řadu běžných funkcí vývojových prostředí jako např. našeptávač, pokročilé vyhledávání, nastavitelné barvení syntaxe apod. Jedna z nejdůležitějších vlastnosti prostředí CooCox CoIDE spočívá ve výborné podpoře funkcí určených pro odhalování a odstranování chyb v programu. Ve spolupráci s ovladačem ST Link, který dodává výrobce jednočipů ST Microelectronics prostředí podporuje překlad programu, nahrání na cílovou platformu ve vývojovém kitu pomocí USB rozhraní a následné spuštění programu na cílové platformě, to vše na jedno kliknutí. Zcela pak odpadá tradiční přístup ve formě zdlouhavého flashování programu do chipu přes specializované programátory, což velmi výrazně zrychluje vývoj na platformě.

Možnosti této kombinace vývojového prostředí a ovladače pokračují podporou tzv. breakpointů a watchpointů. Pomocí breakpointů je možné na libovolný řádek v programu umístit značku, na které se provádění programu zastaví. Tato funkce vývojového prostředí je ještě rozšířena usnadnění umistování breakpointů na metody, tedy lze snadno vybrat ze seznamu metodu (funkci), která je-li zavolána, chod programu se zastaví. Užitečnost této funkce spočívá hlavně v tom, že nachází-li se program v zastaveném stavu, je možné z vývojového prostředí

vyčítat hodnoty proměnných programu běžícího na cílové platformě. Na podobném principu funguje funkce Watchpoint, která na vybraném místě chod programu na okamžik zastaví, aktualizuje hodnoty vyčítaných proměnných programu běžícího na cílové platformě a program znovu aktivuje. S využitím těchto funkcí je tedy možné analyzovat práci programu na cílové platformě na velmi vysoké úrovni, která často nebývá zvykem ani při programování na počítačích typu PC.

3.4 Vývojový kit STM32 Discovery

Pro vývoj firmware převodníku byl firmou Faster.cz, s.r.o [23], která vývoj zařízení iniciovala, navržen vývojový kit STM32 Discovery firmy ST Microelectronics (obrázek 33). Použitá platforma, STM32 ARM [24], vyniká hlavně kombinací nízké ceny jednočipů a vysokého výkonu díky integrovanému RISC jádru ARM Cortex-M0 s maximální taktovací frekvencí 48MHz, velkém počtem osazenných periferií (dva ADC, DAC, dva nastavitelné analogové komparátory, dva USART obvody, čtyři šestnáctibitové GPIO, externí přerušení) a cenově dostupným vývojovým kitem, který je navržen pro podporu co nejsnažšího a nejrychlejšího vývoje ve spolupráci s PC.



Obr. 34 Vývojový kit STM32 Discovery. Zdroj: [25]

Vývojový kit [25] ve své výbavě zahrnuje několik z hlediska vývojáře velmi příjemných funkcí. Kontakty vstupně výstupních pinů jednočipu jsou spolu s napájením vyvedeny na dvě řady kontaktového pole se standartní roztečí kontaktů 2,54mm, neboť fyzická konstrukce kitu počítá se zasunutím modulu do

nepájivého pole, což umožnuje jeho propojení s prakticky libovolným dalším hardwarem. Modul ale lze provozovat i bez dalšího hardwaru nebo signálů přítomných na kontaktovém poli. Modul neobsahuje externí oscilátor, jeho připojení je ale umožněno pomocí vývodů kontaktového pole a modul obsahuje nezbytné externí komponenty pro využití interního 8MHz oscilátoru a propojku, která umožnuje jednoduše přepínat mezi interním a případným externím oscilátorem. Pro samostatné použití je kromě tlačítka RESET vybaven ještě uživatelským tlačítkem, které lze snadno využít pro testovací účely. Kromě toho je také modul osazen jednou modrou a jednou zelenou uživatelskou LED – je tedy možné provádět základní vstupně výstupní operace bez osazování dalších komponent. Modul je primárně navržen pro spolupráci s počítačem typu PC přes USB rozhraní, přes které probíhá i napájení modulu. Na straně PC je vývojový kit doplněn softwarovým ovladačem ST Link, který umožnuje při využití kompatibilního vývojového prostředí poměrně pokročilé debugovací funkce v reálném čase. Tyto umožnuje topologie modulu – kromě vlastního STM32 jednočipu v roli cílové platformy totiž ještě obsahuje zákaznický čip firmy ST Microelectronics s dalším pro programátora jednoduše nepřístupným ARM jádrem, které je k hlavnímu jednočipu připojeno rozhraním JTAG a provádí komunikaci s PC přes rozhraní USB, ovládání hlavního jednočipu, řízení debugovacích funkcí a obsluhu flash paměti hlavního jednočipu. Navíc ovládá červenozelenou LED, která pracuje v roli signalizace provozního stavu vývojového kitu – zelená barva značí funkci zákaznického čipu, červená pak přístup k cílovému jednočipu pomocí rozhraní JTAG. Zmíněné JTAG rozhraní je dostupné na samostatném výstupu a dvěma propojkami je možné zákaznický čip vyřadit a přemostit, takže v případě, že není žádoucí použití zákaznického čipu a USB rozhraní například z důvodu kolizí (když je například potřeba využít JTAG rozhraní cílového jednočipu přímo v programované aplikaci), modul je na to připraven. Případně lze JTAG komunikaci mezi aktivním zákaznickým čipem a cílovým jednočipem na samostatném výstupu monitorovat.

4 Požadavky

Požadavky na vlastnosti a funkce zařízení jsou díky poměrně jasně definovanému prostředí, do kterého bude převodník zasazen, jednoznačně určeny. Narozdíl od procesu navrhování komplexních softwarových systémů lze všechny požadavky redukovat na požadavky funkční a jednoznačně a konkrétně je určit. Protože navrhované zařízení je primárně převodníkem mezi rozhraními OpenTherm a RS485 Modbus, můžeme požadavky rozdělit na kategorie podle pohledu na chování na jednotlivých rozhraních.

Zařízení musí zastávat funkce:

- Převodník OpenTherm ↔ RS485 Modbus.
- Sniffer OpenTherm s ovládáním po RS485 Modbus

V režimu převodníku se zařízení musí na rozhraní OpenTherm chovat jako Master – tedy musí emulovat inteligentní řídící systém. Pro zajištění této funkcionality je třeba:

- Udržovat komunikaci. Na rozhraní OpenTherm platí, že zařízení Master musí minimálně každou 1,15s poslat datovou zprávu na zařízení Slave.
- Pracovat samostatně. Zatímco na straně RS485 Modbus není potřeba provádět udržovací komunikaci, na straně OpenTherm je to nutná podmínka. Převodník tedy nemůže pouze přímo přeposílat data z RS485 Modbus na OpenTherm a zpět, ale musí generovat vlastní komunikaci po rozhraní OpenTherm nezávisle na komunikaci po rozhraní RS485 Modbus

V režimu převodníku se zařízení musí na rozhraní RS485 Modbus chovat jako Slave. Přes toto rozhraní také bude kromě komunikace s nadřazeným řídícím systémem probíhat konfigurace. Pro zajištění této funkcionality je třeba:

- Reagovat na dotazy nadřazeného řídícího systému na nastavené adrese a na broadcast komunikaci
- Být schopen odpovědět relevantními daty na všechny formy dotazu nadřazeného zařízení. Specifikace Modbus dovoluje vícenásobné dotazy, kdy se v rámci jedné datové zprávy komunikuje více segmentů datového modelu, tedy i více ID OpenTherm dat. Protokol OpenTherm toto neumožnuje a komunikace n ID dat může v nejhorším možném případě trvat až n*1,15s, což je nepřijatelně dlouho. Bude tedy třeba zavést určitou formu vnitřní paměti dat a vícenásobné dotazy obsluhovat z ní.
- Indikovat provozní stav rozhraní OpenTherm, například výpadek komunikace z důvodu nedodržení časování.
- Podporovat konfiguraci. Na rozhraní RS485 Modbus není pevně definovaná přenosová rychlost, význam paritního bitu ani Modbus Slave adresa. Je nutné umožnit uživateli tyto položky nastavit, aby bylo možné například zapojit více převodníků na jednu RS485 sběrnici bezkonfliktně.

- Komunikovat OpenTherm data co nejpřesněji, tedy pokud možno beze změn. Úpravy, přepočty a interpretace dat by mohly generovat další problémy s kompatibilitou OpenTherm zařízení.

V režimu snifferu se má zařízení chovat jako odposlouchávací prostředek, který je schopen nahrát komunikaci mezi zapojenými zařízeními Master a Slave na rozhraní OpenTherm. Provozní režim snifferu je zaveden za účelem získávání dat pro reverzní inženýrství podoby komunikace na rozhraní OpenTherm, aby se tato data dala následně využít například pro úpravu komunikačního profilu v aplikaci v nadřazeném řídícím systému. Pro snažší integraci převodníku do již existujícího systému je převodník vybaven dalším rozhraním OpenTherm, určeným pro připojení inteligentního termostatu. .

V režimu snifferu je na rozhraní OpenTherm pro zajištění této funkcionality třeba:

- Na rozhraní k termostatu působit jako Slave. Termostat ovládá převodník, jako by to bylo zařízení Slave.
- Na rozhraní ke kotli (nebo jinému zařízení pro vytápění nebo klimatizaci) působit jako Master. Převodník v režimu snifferu vlastně musí pracovat jako bridge mezi dvěma OpenTherm rozhraními.
- Zaznamenávat veškerou komunikaci mezi zařízeními OpenTherm. Vzhledem k podobě fyzické vrstvy rozhraní OpenTherm není možné s jedním rozhraním jednoduše zaznamenávat oba směry komunikace a zároveň stejný hardware využít pro emulaci Master zařízení. Díky řešení se dvěma rozhraními se ale vnitřní paměť převodníku může chovat jako odraz stavu obou OpenTherm zařízení, protože čtecí a zápisové OpenTherm příkazy převodník kromě retranslace také zaznamená.

V režimu snifferu je na rozhraní RS485 třeba chovat se podobně jako v režimu převodníku, ovšem s jedním výrazným rozdílem – převodník musí ignorovat Modbus příkazy pro zápis, neboť v režimu převodníku není dovoleno zvenčí zasahovat do OpenTherm komunikace.

5 Metodika

V rešeršní části práce je objasněna role rozhraní OpenTherm v prostředí číslicově řízených systémů pro chlazení a vytápění a přístup výrobců k jeho praktickému nasazování. Rozhraní OpenTherm je popsáno z hlediska funkčních vlastností a je jsou diskutovány jeho slabiny, vedoucí k problémům s jeho praktickým nasazením. Následně je prezentováno jedno z možných řešení těchto problémů, které vyžaduje konstrukci zařízení - převodníku mezi rozhraním OpenTherm a rozhraním běžnějším, jehož firmware si tato práce klade za cíl vyvinout.

Pro dosažení cíle bylo třeba provést tyto kroky:

- Důkladně se seznámit s funkcí protokolů OpenTherm s RS485 Modbus
- Důkladně se seznámit s ovládáním hardwarových mechanizmů a integrovaných periferií v rámci zvolené hardwarové platformě
- Navrhnout princip funkce firmwaru tak, aby plnil všechny požadavky na něj kladené a zajišťoval požadovanou funkcionalitu
- Seznámit se ze specifiky programování a vývoje na určené cílové hardwarové platformě
- V souladu s návrhem firmware implementovat v jazyce C.
- Implementaci otestovat z hlediska funkčnosti v reálných podmínkách

6 Návrh

První zásadní problém, který je třeba při návrhu firmware vyřešit, je role jednotlivých rozhraní a zařízení v jednotlivých režimech. S tím souvisí uspořádání toku dat mezi rozhraními zařízení a jednoznačné určení toho, kdo bude s kým kdy co komunikovat.

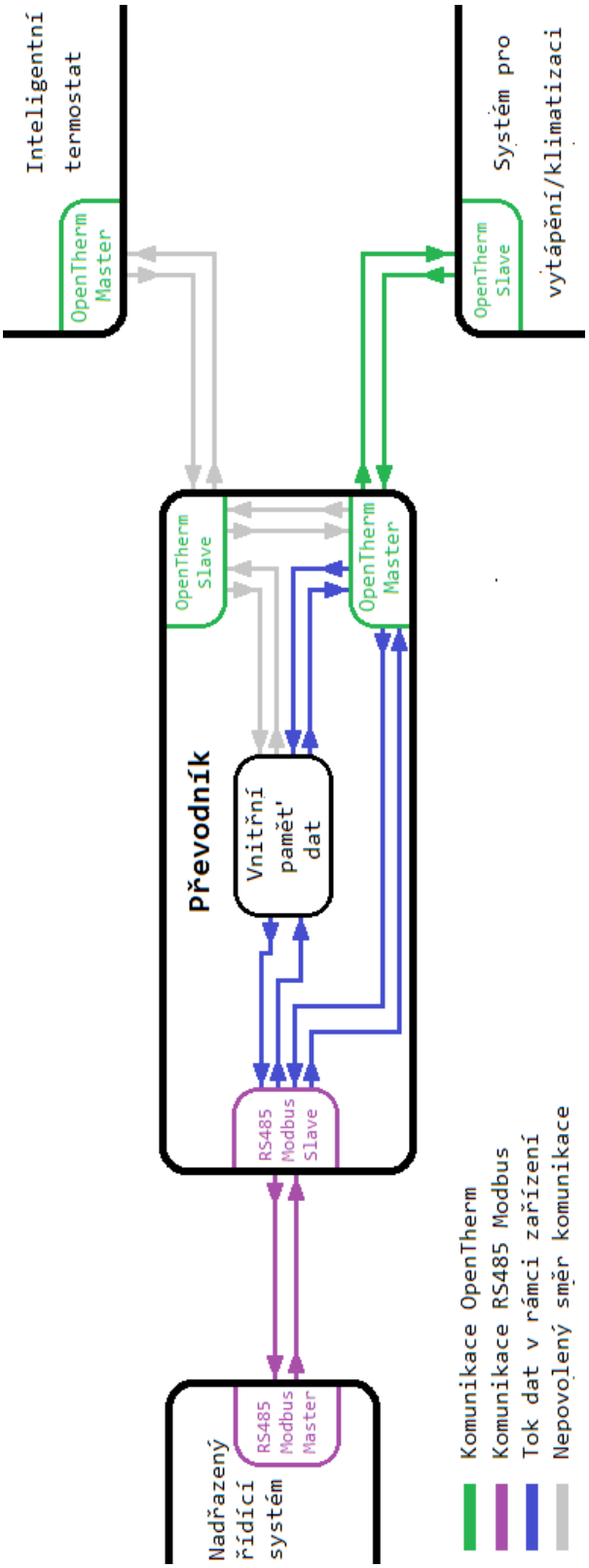
V režimu převodníku (viz obrázek 34) dochází v situaci provozní komunikace na rozhraní RS485 (tedy bez příkazů konfigurace převodníku) ke dvěma situacím.

V první situaci požaduje nadřazený řídící systém hodnotu jediného data ID. V takovém případě není z hlediska času potřebného k dokončení problém dotázat se na data dotazu pocházející z RS485 Modbus přímo na rozhraní OpenTherm Master a odpověd přijatou po rozhraní OpenTherm Master od systému pro vytápění/klimatizaci obratem odeslat jako odpověd na RS485 Modbus. Tuto situaci popisuje pár modrých šipek, vedoucích v obrázku 34 přímo z rozhraní RS485 Modbus do OpenTherm Master a zpět. Také v tomto případě je třeba přenášenými daty aktualizovat vnitřní paměť dat, to obrázek pro větší přehlednost nezachycuje.

V druhé situaci požaduje nadřazený řídící systém hodnotu více data ID v jediném dotazu. V takovém případě začíná vadit specifikace časování konverzací protokolu OpenTherm, která připouští maximální dobu mezi dvěma dotazy zařízení Master 1,15s. Vzhledem k tomu, že v jediném Modbus dotazu může být až 252 bytů čistých dat a OpenTherm protokol přenáší 16tibitová data, je v nejhorším případě možné, že bude potřeba v jedné Modbus odpovědi vrátit 252/2, tedy 126 hodnot OpenTherm data ID. Protokol OpenTherm neumožnuje vícenásobné dotazování, takže by v takovém případě bylo třeba provést 126 OpenTherm dotazů, což by v nejhorším případě mohlo trvat až 126*1,15 = 144,9s, tedy více než dvě minuty. Na rozhraní RS485 Modbus sice není definován čas pro timeout, ale v praktickém nasazení je čekání přes dvě minuty nepřijatelně dlouhé. Proto je nutné převodník vybavit vnitřní pamětí, která bude ukládat hodnoty všech dle specifikace možných OpenTherm data ID a bude sloužit jako buffer pro obsluhování těchto vícenásobných dotazů. V případě vícenásobného dotazu jsou tedy hodnoty vraceny ne přímo z OpenTherm rozhraní, ale z vnitřní paměti převodníku, což popisují v obrázku 34 modré šipky propojující RS485 Modbus rozhraní s vnitřní pamětí dat.

Ukládání dat do vnitřní paměti ale přináší jiný problém. Jakým způsobem zajistit, aby byly v paměti vždy čerstvé informace? Tento problém a zároven problém nutnosti udržovací komunikace na OpenTherm rozhraní řeší cyklovací funkce která ve chvílich, kdy na převodník nepřichází dotazy z nadřazeného řídícího systému po rozhraní RS485 Modbus provádí cyklicky dotazy po rozhraní OpenTherm na všechny data ID. Odpovědi na tyto dotazy jsou ukádány do vnitřní paměti dat, což zajištuje její aktuálnost. Tento mechanizmus je v obrázku 34 ilustrován modrými šipkami propojujícími vnitřní paměť a OpenTherm rozhraní.

Tato vlastnost zavádí určité problémy a problémové stavy do návrhu. Bude-li se například nadřazený řídící systém neustále nepřetržitě dotazovat, nebude mít převodník volný čas potřebný pro aktualizaci celé vnitřní paměti. Ta data ID, která



Obr. 35 Tok dat a komunikace v režimu převodník

jsou nadřazeným řídícím systémem dotazována budou aktuální i v paměti, ale kompletně aktuální paměť z principu věci nebude. Elegantnější řešení vzhledem k vlastnostem protokolu OpenTherm ale není podle mě možné, omezení funkčnosti převodníku není podle mého názoru nijak zásadní. Je ale třeba s tímto problémem jako s vlastností návrhu počítat.

Komunikace s inteligentním termostatem na OpenTherm Slave rozhraní není povolena. V režimu převodníku má v roli řídícího prvku pracovat nadřazený řídící systém prostřednictvím převodníku. Protokol OpenTherm je navržen jako point-to-point, nepočítá s existencí více zařízení typu Master na vedení. Připustili-li bychom komunikaci s případným zařízením typu Master na druhém OpenTherm rozhraní, mohlo by to vést ke kolizím. Druhé rozhraní OpenTherm není v režimu převodníku vypnuto fyzicky z důvodu případného napájení zařízení Master, ale neprobíhá na něm komunikace žádným směrem.

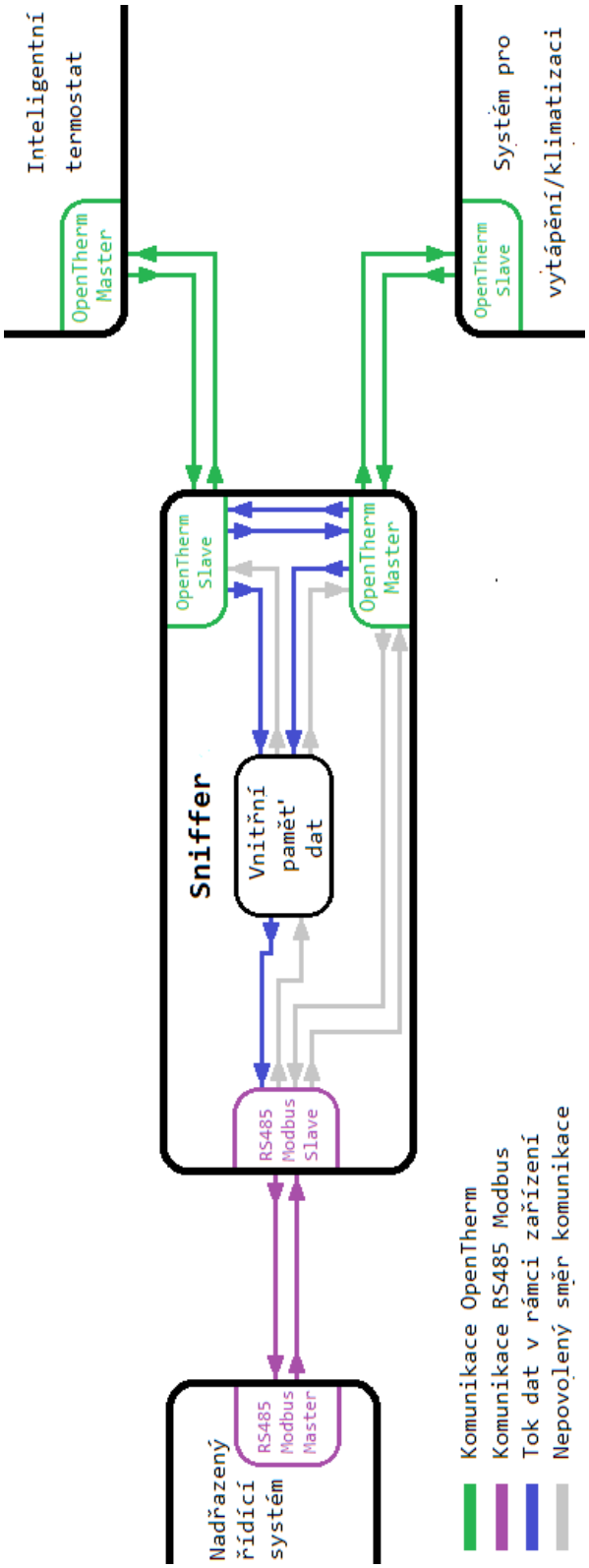
Stejný model toku dat a komunikace, tentokrát pro zařízení v režimu snifferu, je zobrazen na obrázku 35.

Funkce zařízení v režimu snifferu je od funkce v režimu převodníku odlišná. Hlavním směrem komunikace je zde komunikace mezi dvěma OpenTherm rozhraními. Zařízení v režimu sniffer primárně provádí retranslaci zpráv mezi OpenTherm rozhraními tak, aby se z pohledu inteligentního termostatu i systému pro vytápění/klimatizaci jevilo jako neviditelné. Jelikož datová zpráva OpenTherm má vždy stejnou délku, je možné provádět přímou retranslaci bez nutnosti provádět čtení z vnitřní paměti. Do vnitřní paměti je ale zapisována každá datová zpráva OpenTherm ve směru Master → Slave, která má typ zprávy Zapiš data a každá zpráva v opačném směru, která má typ zprávy Potvrzuji čtení dat. Toto chování popisují modré šipky propojující v obrázku obě OpenTherm rozhraní a vnitřní paměť.

Komunikace na rozhraní RS485 Modbus se také liší. Dovolené jsou všechny operace zahrnující čtení dat, tedy čtení jednotlivé i vícenásobné. Zakázány jsou operace zápisu, které v režimu snifferu nemají smysl, neboť zápisem by mohlo docházet ke kolizi s daty, která přichází po OpenTherm rozhraní směrem od inteligentního termostatu. Toto chování popisuje modrá šipka, spojující vnitřní paměť s rozhraním RS485 Modbus.

Z tohoto návrhu komunikace vychází návrh konkrétního řešení vnitřní topologie převodníku. Centrálním prvkem návrhu je vnitřní pamět. Tato pamět bude ukládat data ve tvaru odpovídajícím protokolu OpenTherm, a to všechna povolená data ID, tedy celkem 256 datových položek. Toto jsem zvolil toho důvodu, že na OpenTherm rozhraní jsou tvary, významy a datové typy dat definovány jasně, kdežto na rozhraní RS485 Modbus nikoliv. Na rozhraní RS485 Modbus je v datovém modelu definován jeden adresovatelný segment jako položka v délce 16 bitů. Na rozhraní OpenTherm má datová část datové zprávy velikost také 16 bitů. Z důvodu udržení co nejširší kompatibility na rozhraní OpenTherm je žádoucí pokud možno vůbec data nealternovat. Vnitřní pamět tedy bude obsahovat 256 16tibitových datových položek ve tvaru, jako by byly přímo získané z datových zpráv protokolu OpenTherm.

Ve specifikaci protokolu OpenTherm ale některá data ID podléhají režimu pro zápis, kdežto jiná, kterých je drtivá většina, podláhají režimu pro čtení. Vnitřní



Obr. 36 Tok dat a komunikace v režimu Sniffer

pamět tedy ještě musí ukládat o každém data ID informaci, zda pro jeho komunikaci používat zápisový, nebo čtecí OpenTherm příkaz.

Centrální pamět budou obsluhovat nezávislé rutiny ve formě programových smyček. V případě vícenásobného dotazu ze strany nadřazeného řídícího systému po rozhraní RS485 Modbus bude jednou obslužnou rutinou vyhledány všechny potřebné informace z vnitřní paměti, sestavena patřičná odpověd a odeslána.

Další obslužná rutina bude v režimu převodníku a v případě, kdy ze strany nadřazeného řídícího systému nepřichází nebo nečeká na vyřízení žádný příkaz, cyklicky provádět dotazování po rozhraní OpenTherm na všechna podle specifikace možná data ID a odpovědi bude zapisovat do vnitřní paměti.

Další obslužná rutina bude v režimu snifferu v případě dotazu ze strany nadřazeného řídícího systému na hodnotu jediného data ID tuto hodnotu vyčítat z vnitřní paměti, konstruovat z ní RS485 Modbus odpověd, kterou odešle.

V zařízení budou potřeba ještě další rutiny, které svou funkcí na vnitřní paměti zcela nezávisí. V režimu převodníku v případě dotazu ze strany nadřazeného řídícího systému na hodnotu jediného data ID bude obslužná rutina formulovat z tohoto RS485 Modbus dotazu OpenTherm dotaz, který odešle co nejdříve to bude možné. Z OpenTherm odpovědi ze strany systému pro vytápění/klimatizaci na tento dotaz poté rutina zkonstruuje RS485 Modbus odpověd, kterou odešle. Kromě toho také aktualizuje informaci ve vnitřní paměti daty z OpenTherm odpovědi.

V režimu snifferu poběží také dvě rutiny, které budou zajišťovat retranslaci dat mezi oběma OpenTherm rozhraními. Jeden přijatý OpenTherm dotaz na jednom rozhraní ve funkci Slave se obratem přepošle jako dotaz na druhé OpenTherm rozhraní ve funkci Master, coz bude zajišťovat první z rutin. Tato rutina také aktualizuje přijatými daty vnitřní pamět, pokud datová zpráva bude typu „Zapiš data“. Odpověd na druhém OpenTherm rozhraní se obratem přepošle jako odpověd na první OpenTherm rozhraní, což zajistí druhá rutina. Tato rutina také aktualizuje přijatými daty vnitřní pamět, pokud bude typu „Potvrzuji čtení dat“.

7 Implementace

Veškerá programová podpora hardwarové platformy se pro jazyk C omezuje pouze na hlavičkový s definicemi adres registrů a ovládání řadiče přerušení a zdroje taktovacího kmitočtu. Vzhledem k této faktické neexistenci programových knihoven je v rámci implementace nezbytné naprogramovat nejprve funkce nejnižší úrovně a vhodně nastavit konfigurační registry procesoru. Proto se nejprve zaměříme na implementaci těchto základů, na kterých lze posléze vystavět funkcionalitu odvozenou v návrhu funkce firmware.

7.1 Implementace nízkoúrovnových funkcí

Pro celý další popis platí, že všechen programový kód je uložen na kompaktrním disku, přiloženém k této práci a není-li uvedeno jinak, diskutovaná funkcionalita je implementována v hlavním souboru *main.c*. Prvním a základním krokem v rámci implementace je definice proměnných, konstant a datových typů.

Základní proměnnou celého firmware je vnitřní pamět dat. Tato je definovaná 16tibitovým polem *data_map* s délkou 256 položek K této paměti se váže pole příznaků *wset*, které obsahuje 256bitů jednobitových příznaků toho, že data ID je pro zápis. Z důvodů pamětové úspornosti je implementováno jako pole 32bitových integerů. Proto je nutné implementovat funkci *is_in_wset*, která s použitím bitových posuvů a logických funkcí vrací pro vstupní data ID logickou hodnotu *true*, pokud dataID je pro zápis.

Další nezbytně nutnou proměnnou je indexovací proměnná *data_comm_index*, která představuje index pro cyklické dotazování na OpenTherm v rámci udržovací komunikace se systémem pro vytápění/klimatizaci v režimu převodníku. K této proměnné se váže funkce *next_datacomm_index*, která, je-li zavolána, inkrementuje proměnnou *data_comm_index*, případně ji při překročení povoleného rozsahu nuluje.

Firmware obsahuje ještě další proměnné, jejichž účel bude objasněna dále v kontextu funkčních celků programu, ke kterým přísluší.

Po definici proměnných je potřeba provést konfiguraci hardware. První a nejdůležitější konfigurace se provádí v hlavičkovém souboru a určuje zdroj a frekvenci taktovacího kmitočtu. Je nastavena na použití interního oscilátoru HSI o frekvenci 8Mhz, ze kterého následně fázový závěs vyrábí všechny potřebné vnitřní taktovací signály včetně hlavního taktovacího signálu jádra o frekvenci 48MHz.

Z hardwarových prostředků jednočipu jsou ve firmwaru využity 16tibitové vstupně výstupní porty A, B, které je třeba explicitně aktivovat a do vnitřních podpůrných obvodů přivést taktovací signál. Jednotlivé piny jsou pak nastaveny podle požadovaných funkcí.

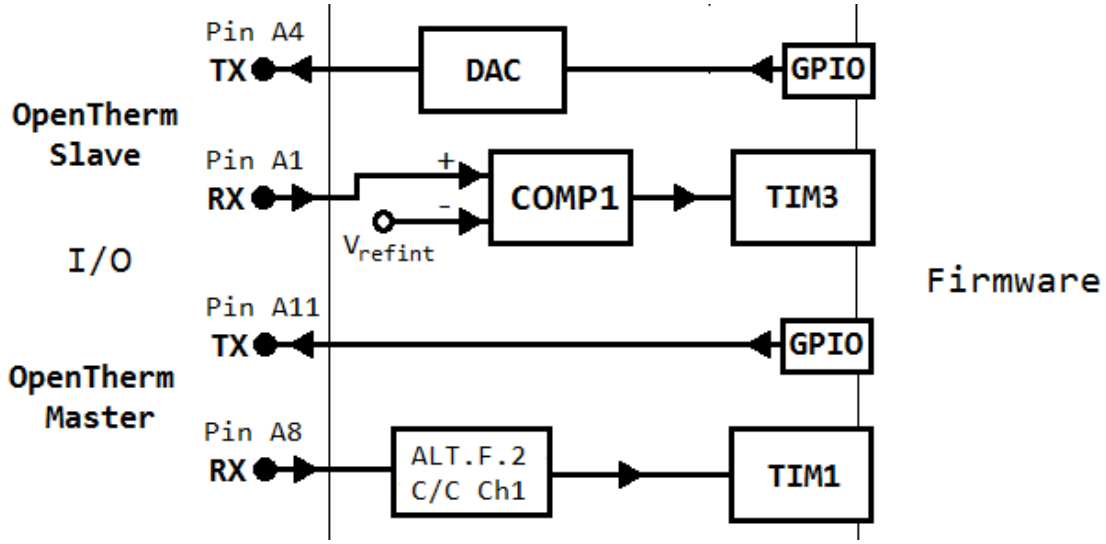
Piny A6 a A7 jsou nastaveny jako výstupní – jsou k nim připojeny indikační LED pro indikaci komunikace na OpenTherm rozhraních.

Pin A11 je nastaven jako digitální výstupní – je určen pro napětové vysílání na OpenTherm Master rozhraní.

Pin A8 je nastaven pro alternativní funkci – první vstupní kanál časovače TIM1, funguje jako přijímací vstup na OpenTherm Master rozhraní.

Pin A1 je nastaven jako vstupní s alternativní funkcí – dále je zapojen jako neinvertující vstup komparátoru COMP1, což je využito pro příjem na OpenTherm Slave rozhraní.

Pin A4 je nastaven jako výstupní analogový výstup z digitálně analogového převodníku DAC1 pro vysílání na OpenTherm Slave rozhraní, které probíhá proudově, tedy v režimu proudové smyčky. Výstupní napětí, které mění externí budič OpenTherm na vhodné hodnoty proudu je nastaveno v nastavení DAC1. Celé zapojení hardwarových vstupů a výstupů rozhraní OpenTherm s využitím integrovaných periferií jednočipu je zobrazeno na obrázku 37:



Obr. 37 Zapojení I/O pinů a využití integrovaných periferií

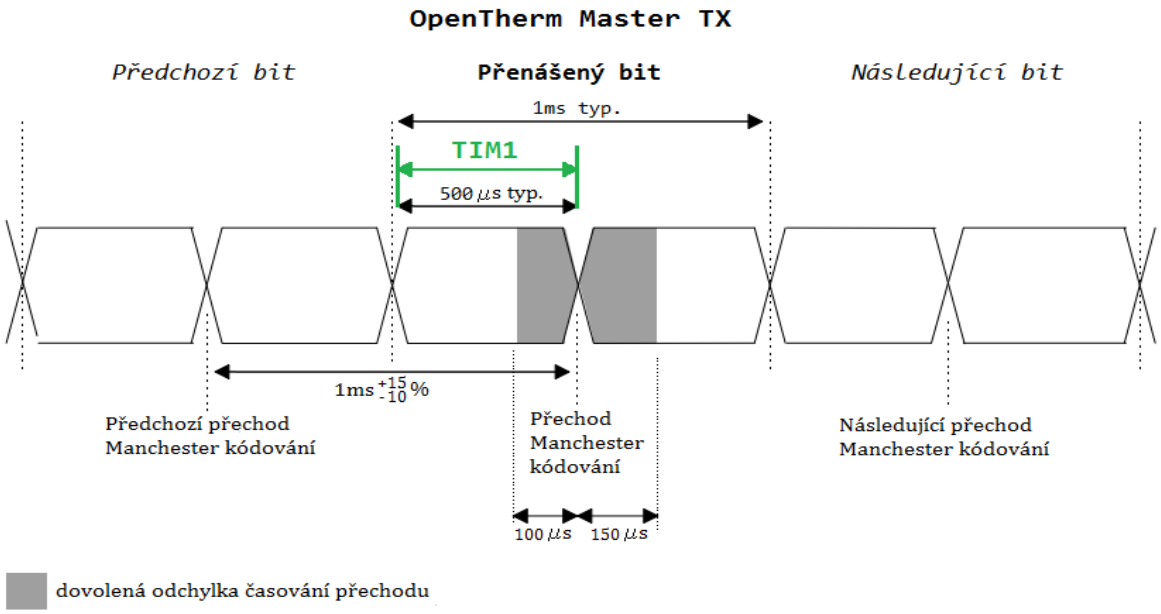
Situace na straně RS485 Modbus je výrazně méně komplikovaná, protože o vysílání a příjem se stará integrovaný USART1 obvod, který je nakonfigurován pro rozhraní RS485 a režim Modbus RTU. Integrovaný Universal Synchronous - Asynchronous Receiver – Transmitter, neboli USART obvod ovládá externí budič RS485 linky pomocí třech signálů. Pin A9 je nastaven jako USART1 RX pro příjem, pin A10 jako USART1 TX pro vysílání a pin A12 jako USART RTS – výstup indikující připravenost vysílat data, který přepne budič do režimu vysílání.

Piny B5, B4, A13, A14, A15, B3, B1, B0 jsou využity pro zamýšlené lokální nastavení Modbus adresy, parity (sudá/lichá) a přenosové rychlosti (9600/19200 baud). Tato funkce ale zatím není podporována v návrhu hardwaru zařízení a počítá se s ní pro pozdější revize.

Nízkoúrovnové funkce spočívají hlavně ve fyzickém zajištění příjmu a vysílání na OpenTherm a RS485 Modbus rozhraních. Tyto jsou řešeny hlavně s pomocí využití časovačů a přerušení jimi vyvolaných. Z časovačů jsou využity vnitřní časovače TIM1, TIM3, TIM15, TIM16 a to takto:

Časovač TIM1 je využit ve dvou situacích. Jednak při vysílání na OpenTherm Master rozhraní směrem k systému pro vytápění/klimatizaci, kdy je s jeho pomocí odměřována polovina bitu Manchester kódu v trvání 500µs (viz obrázek 38).

Druhak je využit při přijmu na OpenTherm Master rozhraní, kde se s jeho pomocí provádí detekce přechodu mezi logickými úrovněmi.

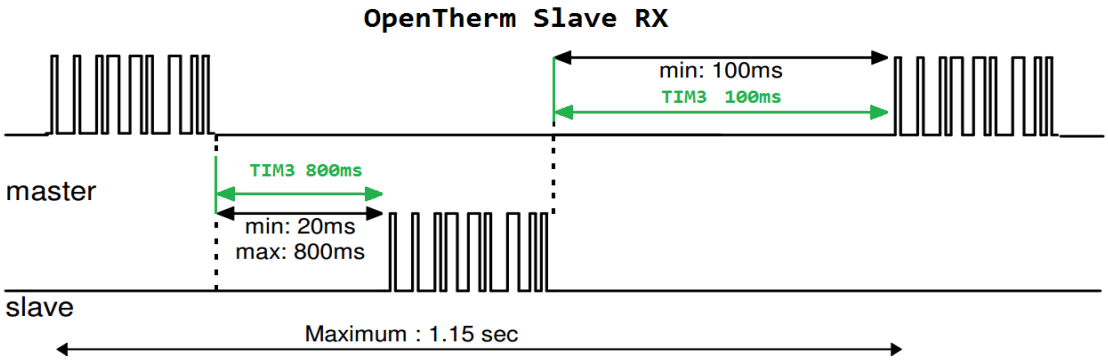


Obr. 38 Jedno z využití časovače TIM1 – TX OpenTherm Master. . Zdroj: [21]

Časovač TIM15 se používá jako pomocný časovač pro timeouty konverzace při přijmu na OpenTherm Master rozhraní. Jeho funkce se mění podle momentální fáze konverzace. Pokud je konverzace ve stavu čekání na odpověd systému pro vytápění/klimatizaci, časovač TIM15 je využit pro odměření maximálního timeoutu pro odpověd Slave zařízení. Je-li tato doba (800ms) dosažena před přijetím odpovědi, rozhraní opouští režim přijímače za účelem provedení dalšího vysílání.

Pokud je konverzace ve stavu čekání na možnost nového vysílání (protokol OpenTherm definuje minimální čekací dobu mezi ukončením příjmu odpovědi Slave zařízení na předchozí dotaz a odesláním dotazu nového), je pomocí časovače TIM15 odměřováno uplynutí této doby (100ms) a až po jejím uplynutí je povoleno vysílání nového dotazu.

Časovač TIM3 je využit pro příjem i vysílání na OpenTherm Slave rozhraní (obrázek 39). Jeho prostřednictvím se při přijmu měří prodleva mezi přechody mezi logickými úrovněmi. Časovač TIM3 pracuje v kooperaci s komparátorem COMP1, který porovnává vstup RX OpenTherm Slave (pin A1) s vnitřní referencí a je nastaven tak, aby detekoval náběžnou i sestupnou hranu signálu, tedy překročení i pokles pod napětí vnitřní reference. Při vysílání na OpenTherm Slave se jeho prostřednictvím odměřují 500µs poloviny bitu v Manchester kódování.



Obr. 39 Jedno z využití časovače TIM3 – OpenTherm Slave RX. Zdroj: [21]

Kromě toho je také použit na měření 100ms timeoutu mezi odesláním odpovědi v režimu Snifferu pro inteligentní termostat a přijmem dotazu začínajícího další konverzaci a na měření 800ms timeoutu mezi koncem vysílání odpovědi a začátkem přijmu dalšího dotazu podobně jako časovač TIM15 na rozhraní OpenTherm Master.

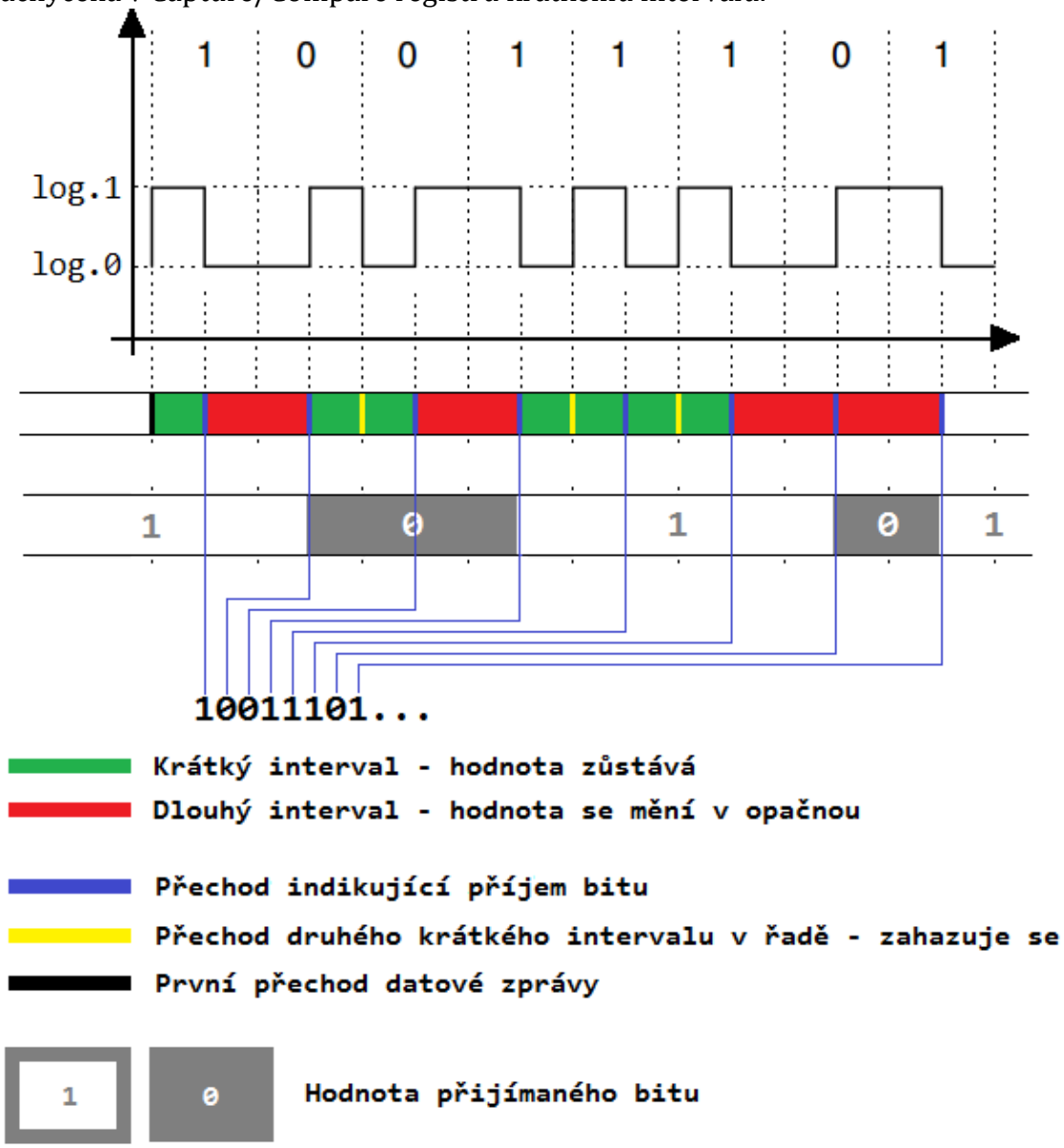
Časovač TIM16 je využit jako počítadlo timeoutu na straně RS485 Modbus komunikace. Tento timeout není ve specifikaci protokolu RS485 Modbus vůbec definován, ale v tomto konkrétním případě byla určena jeho defaultní hodnota jako 1s. Dojde-li tedy k přijmu Modbus datové zprávy a data je nutné obstarat čerstvá prostřednictvím OpenTherm Master rozhraní, je odstartován časovač TIM16. Po uplynutí 1s timeoutu se transakce vzdává, RS485 Modbus rozhraní přechází zpět do režimu přijímání a očekává nový dotaz od nadřazeného řídícího systému.

Na funkci časovačů navazuje obsluha přerušení, která jsou ve většině případů časovači spouštěna. Jednočipové mikrokontroléry rodiny STM32F0 jsou vybaveny pokročilým systémem vrstvených přerušení s nastavitelnou prioritou. Tu lze využít v případě, kdy obsluha jednoho přerušení je více kritická, než obsluha přerušení jiného. Řadič přerušení navíc podporuje vrstvená přerušení. Může tedy nastat situace, kdy z hlavního programu procesor skočí do obsluhy přerušení s nižší prioritou a v době obsluhy tohoto přerušení dojde k přerušení vyšší priority, do jehož obsluhy se skočí přímo z obslužné rutiny přerušení nižší priority. Takto lze přerušení vrstvit až do čtyřech vrstev, přičemž čím vyšší číslo priority, tím nižší je priorita.

Časovač TIM1 je nastaven na vyvolání dvou různých přerušení. První z nich, *TIM1_CC_IRQ*, je vyvoláno, pokud nastane tzv. Compare/Capture událost. Spolu s nastavením hardwaru to znamená, že přerušení je vyvoláno v případě, že na pinu A8 (OpenTherm Master RX) dojde ke změně logické úrovně. Chod časovače při vyvolání přerušení není zastaven, pouze je do Capture/Compare registru uložena vnitřní hodnota časovače, při které k přerušení došlo. Této funkcionality je využito v přijímači a dekodéru Manchester kódu ,který je v externím přilinkovaném souboru *manchester.c*, takto:

V obslužné rutině *TIM_CC_IRQHandler* je v první řadě časovač vynulován. Pokud jde o úplný začátek přenosu, jiná akce se nedějě. Dále se postupuje podle

algoritmu dekodéru Manchester: V každé obsluze přerušení je z Capture/Compare registru zjištěno, jednalo-li se od předchozího vyvolání přerušení o krátký, nebo dlouhý časový interval. Manchester dekodér je pouze s využitím této informace schopen dekódovat a vyjádřit původní datovou zprávu. OpenTherm datová zpráva začíná vždy bitem v log.1. V prvním přerušení v rámci přijmu datové zprávy (černě označeném) je časovač pouze vynulován. Při dalším přerušení odpovídá hodnota zachycená v Capture/Compare registru krátkému intervalu.



Obr. 40 Princip Manchester dekodéru. Zdroj: [21]

Krátká nebo dlouhá hodnota v případě Manchester kódování neodpovídá skutečné logické hodnotě bitu, ale v případě krátké hodnoty znamená, že přijímaná logická hodnota bitu se nemění a zůstává ve stavu, ve kterém byla před vyvoláním přerušení, v případě dlouhé hodnoty časového intervalu se přijímaná logická hodnota bitu mění v opačnou, a to již pro současný přijímaný bit. Se dvěma výjimkami znamená každé přerušení přijetí jednoho bitu. První výjimka je první

přerušení v rámci datové zprávy. Druhá výjímka je přerušení s krátkým časovým intervalem, pokud bezprostředně před ním došlo také k přerušení s krátkým časovým intervalem. Aplikací těchto pravidel je algoritmus schopen dekódovat přijímanou datovou zprávu v kódování Manchester, viz obrázek 40.

S funkcí tohoto přerušení je spojeno několik proměnných a externích funkcí. Samotné dekódování impulzů v bity a kontrola počtu přijatých bitů se provádí voláním funkce *manchester_decode v* externím přilinkovaném souboru *manchester.c*.

Stav během přijímu se ukládá do stavové proměnné *OT_boil_comm.status*, podle které se řídí i ostatní části programu pro komunikaci na OpenTherm Master rozhraní. Přijatá datová zpráva je uložena do struktury *OT_boil_in,* která je vnitřně rozdělena podle specifikace OpenTherm na část *data_value*, *data_id* a *msg_type*. V případě chyby je nastaven chybový příznak *OT_boil_comm.rx_err* Přerušení je nastaveno s prioritou 1.

Přerušení *TIM1_BRK_UP_TRG_COM_IRQ* je generováno přetečením časovače TIM1. Používá se v rámci vysílání na OpenTherm Master rozhraní k odměřování 500μs polovin bitu pro Manchester kódování. V jeho obsluze *TIM1_BRK_UP_TRG_COM _IRQHandler* se provádí vysílání s využitím externí funkce *manchester_encode* přilinkované z externího souboru *manchester.c*, která pracuje analogicky k Manchester dekodéru a s využitím opačného postupu kóduje odesílanou datovou zprávu. V rámci obsluhy se pak podle výsledku kódování změní, nebo nezmění logická výstupní úrověn na TX pinu OpenTherm Master rozhraní. Přerušení je nastaveno s prioritou 1.

Přerušení *TIM3_IRQ* je generováno třemi různými událostmi. Která událost přerušení vyvolala se zjišťuje až v obsluze přerušení. Kvůli tomu obsahuje obslužná rutina *TIM3_IRQHandler* tři různé větve kódu, které se spouští každá v případě jiného důvodu vyvolání přerušení.

První dva důvody pro generování přerušení jsou přetečení časovače. K tomu jednak dochází, je-li časovač TIM3 použit k odměření 100ms timeoutu mezi odesláním odpovědi na OpenTherm Slave rozhraní a příjmem dalšího dotazu. V tomto případě je v obsluze přerušení zkontrolováno, jestli se OpenTherm Slave rozhraní nachází ve stavu aktivního příjmu, tedy jestli na rozhraní proudí data. Pokud ano, znamená to, že k timeoutu nedošlo. Pokud ne, je patřičně nastaven příznak *OT_thermo_comm.status*.

Druhak, je-li OpenTherm Slave rozhraní už ve stavu aktivního vysílání, časovač odměřuje 500μs poloviny bity Manchester kódování a řídí vysílání odpovědi na OpenTherm Slave rozhraní analogicky s algoritmem v *TIM1_BRK_UP_TRG_COM _IRQHandler.*

Třetí důvod pro generování přerušení je Compare/Capture událost spuštěná komparátorem COMP1. K té dochází v rámci příjmu na OpenTherm Slave rozhraní. Mechanizmus je naprosto stejný s mechanizmem příjmu na OpenTherm Master rozhraní v obsluze přerušení *TIM1_CC_IRQ*. Používá se stejný algoritmus Manchester dekodéru a pracuje se s analogickými proměnnými *OT_thermo_comm.status*, *OT_thermo_comm.rx.err* a *OT_thermo_in.* Přerušení je nastaveno s prioritou 1.

Přerušení *TIM15_IRQ* je vyvoláno pouze přetečením časovače. Používá se k odměření obou timeoutů mezi datovými zprávami na OpenTherm Master rozhraní, tedy k zjištění, zda zařízení Slave odpovědělo na námi odeslaný dotaz dříve, než za 800ms a také k odměření povinné 100ms pauzy mezi ukončením přijmu odpovědi od systému pro vytápění/klimatizaci a začátkem vysílání dotazu počínajícího následující OpenTherm konverzaci. V obou případech dochází v rámci obslužné rutiny *TIM15_IRQHandler* k nastavení patřičného stavu v proměnné *OT_boil_comm.status* Přerušení je nastaveno s prioritou 2.

Přerušení *TIM16_IRQ* je také vyvoláno výlučně přetečením časovače. Používá se pro detekci timeoutu na straně RS485 Modbus komunikace. Vyvoláním přerušení je zrušena transakce nepokračující z důvodu selhání OpenTherm komunikace a USART je přepnut zpět do režimu přijímače.

Na straně RS485 Modbus komunkace probíhá obsluha vysílání i přijmu přes USART pomocí přerušení *USART1_IRQ*. Narozdíl od nutnosti konstruovat kódovací a dekódovací algoritmy pro OpenTherm rozhraní je USART schopen se postarat o výrazně více funkcí automaticky a práce s ním je méně náročná a příjemnější, než s rozhraním OpenTherm. USART je třeba správně nastavit z hlediska přenosové rychlosti, parity, timeoutu pro příjem, tvaru znaku (počet start bitů, počet stop bitů, použití a typ parity, délka datové části), typu komunikačního rozhraní, budiče. Všechno toto nastavení je přejato z proměnných, které jsou dostupné pro čtení i zápis přes rozhraní RS485 Modbus – je tedy možné dálkově konfigurovat nastavení USART.

Je-li USART vhodně nastaven, vyvolává přerušení *USART1_IRQ* v několika různých případech. V obslužné rutině *USART1_IRQHandler* je tedy třeba řešit následující situace:

Bylo spuštěno vysílání a USART potřebuje další znak. Přenos přes USART pracuje po jednotlivých znacích a znak, který chceme odeslat je třeba umístit do vysílacího registru USART a vynulovat příslušný chybový příznak.

Bylo dosaženo konce vysílání datové zprávy. Před každým vysíláním nastavíme délku vysílané zprávy ve znacích a USART tuto hodnotu automaticky dekrementuje. Dosáhne-li nuly, je vyvoláno přerušení. V tomto případě ukončujeme režim vysílání a přecházíme do režimu přenosu.

Byl přijat nový znak. V případě, že jsme v režimu příjmu vyčteme znak z přijímacího registru USART a uložíme. Přečtením znaku se automaticky nuluje příznak přijetí znaku.

Během přijmu bylo dosaženo nastaveného timeoutu. Reagujeme vyčtením nekompletních dat a restartem USART, aby mohlo dojít k započetím nového přijmu.

USART také poskytuje ve svých registrech řadu příznaků, indikujících chybu v přijmu – chybu parity, přetečení, šumu na lince, nekorektní tvar znaku. Pokud je některý z těchto příznaků aktivován, indikujeme možnost, že data jsou poškozena, nastavením délky přijaté zprávy na 0. USART pracuje se dvěma páry proměnných. Činí je řetězec *Modbus_in* a celočíselná proměnná *Modbus_length_in* pro příjem a analogicky *Modbus_out* s *Modbus_length_out* pro vysílání. V proměnných jsou uloženy vysílaný a přijímaný řetězec – datová zpráva a její délka, důležitá pro informování USART obvodu pro korektní odvysílání.

Určitou mezivrstvu mezi obsluhami přerušení a hlavní programovou smyčkou tvoří řada funkcí s názvy ve tvaru *<rozhraní>_<akce>.* Tyto funkce jsou volány z hlavní programové smyčky, provádí přípravy, spouštění a ukončování režimů rozhraní, zapínají a vypínají potřebná přerušení, inicializují, spouštějí a zastavují časovače.

Prvním kvartetem takových funkcí jsou funkce pro práci s rozhraním OpenTherm Master *OT_boiler_TX*, *OT_boiler_RX*, *OT_boiler_WAIT* a *OT_boiler_IDLE*.

Funkce *OT_boiler_TX* provede inicializaci časovače TIM1 pro odměřování polovin bitu Manchester kódu, nastaví stavovou proměnnou *OT_boil_comm.status*, inicializuje Manchester kodér (v externím přilinkovaném souboru *manchester.c*) a zapne časovač. Dále už vysílání řídí přerušení při přetékání časovače TIM1.

Funkce *OT_boiler_RX* provede inicializaci časovače TIM15 pro měření přijímacího timeoutu a inicializaci časovače TIM1 pro Capture/Compare, tedy vyvolání přerušení při zachycení změny logické úrovně na přijímacím pinu OpenTherm Master rozhraní (Pin A8). Přetečení časovače je nastaveno na nejdelší možnou délku jednoho bitu, aby v průběhu příjmu v situaci, kdy se protistrana odmlčí nedošlo k nekonečnému čekání. Nakonec se nastaví stavová proměnná a oba časovače se zapnou. Dále už příjem řídí Capture/Compare přerušení časovače TIM1.

Funkce *OT_boiler_WAIT* používá časovač TIM15 k vyčkání po čas stanovený parametrem funkce. Během tohoto času je stavová proměnná nastavená na hodnotu OT_STATUS_WAITING a na rozhraní OpenTherm Master neprobíhá vysílání ani není schopno přijmu. Používá se pro odměření 100ms timeoutu mezi koncem přijmu odpovědi protistrany a začátkem vysílání nového dotazu. Funkce časovač inicializuje, nastaví stavovou proměnnou a zapne časovač. Zrušení stavu potom proběhne při přetečení časovače v obsluze příslušného přerušení.

Funkce *OT_boiler_IDLE* je pomocná funkcem která vypíná časovače TIM1 a TIM15 a nastavuje stavovou proměnnou pro indikaci IDLE stavu – tedy stavu, ve kterém probíhá cyklické dotazování na všechny povolené hodnoty data ID na OpenTherm Master rozhraní a aktualizování vnitřní paměti. Funkce samotná toto dotazování neprovádí, to je záležitostí hlavní smyčky programu.

Další dvě funkce jsou podobným ovládacím rozhraním pro OpenTherm Slave. Jsou to funkce *OT_thermo_TX* a *OT_thermo_WAIT.* Na rozhraní OpenTherm Slave není třeba funkce pro příjem, ten totiž v režimu snifferu probíhá neustále a jen po úspěšném provedení přijmu se rozhraní může přepnout do stavu vysílání.

Funkce *OT_thermo_TX* tedy provádí dokončení retranslace komunikace mezi rozhraními OpenTherm Master a Slave. Nejprve přichází datová zprává na rozhraní OpenTherm Slave, tato je předána na rozhraní OpenTherm Master. Odpověd z rozhraní OpenTherm Master je zpět na rozhraní OpenTherm Slave odvysílána právě funkcí *OT_thermo_TX*, tedy voláním této funkce je vysílání započato. Vysílání začíná inicializací časovače TIM3 na odměřování 500µs polovin bitu Manchester kódování, je vynulován stavový příznak *OT_thermo_comm.boiler_req*, čímž je mezitransakce na OpenTherm Master rozhraní označena za ukončenou a vysílání je započato inicializací Manchester kodéru z externího přilinkovaného souboru *manchester.c* a zapnutím časovače. Počátek vysílání může být opožděn o hodnotu vstupního parametru *time*. Tato

vlastnost funkce bude potřeba hlavně v situaci, kdy se nepodaří obdržet validní data z OpenTherm Master rozhraní a převodník bude nucen odpovědět daty z vnitřní paměti. V tomto případě bude nutné před vysílání odpovědi na OpenTherm Slave rozhraní vložit umělou minimální mezeru, podle specifikace OpenTherm 20ms.

Funkce *OT_thermo_WAIT* je použita pro implementaci čekání při retranslaci dat v režimu snifferu a to hlavně v situaci, kdy se čeká na dokončení mezitransakce na OpenTherm Master rozhraní. Funkce pracuje analogicky s funkcí *OT_boil_WAIT*, jen využívá časovač TIM3.

Poslední tři funkce této vrstvy rozhraní jsou určeny pro práci s rozhraním RS485 Modbus. Jsou to funkce *MB_iface_TX*, *MB_iface_RX* a *MB_iface_WAIT*.

Funkce *MB_iface_TX* provádi přípravu a odstartování vysílání. Před zavoláním funkce musí být výstupní datová zpráva již umístěna v řetězci *Modbus_out* a její délka v *Modbus_out_length*. Funkce zavolá funkci pro výpočet CRC z přilinkovaného externího souboru *modbus.c*, vypočítané CRC přidá ke zprávě, aktualizuje její délku, nastaví stavovou proměnnou rozhraní Mod_comm.status a započne vysílání. Další průbeh vysílání se řeší v obsluze přerušení USART1_IRQ.

Funkce *MB_iface_RX* provádí přípravu a přechod do režimu příjmu. Funkce vlastně provede restart USART, vynuluje délkové a indexační proměnné Modbus, a zapne USART v režmu přijímače.

Funkce *MB_iface_WAIT* má poněkud jiný účel než funkce *WAIT na OpenTherm rozhraní. Používá se pro nastavení a spuštění časovače TIM16, který odměřuje timeout v případě, že zařízení v režimu převodníku a data, která žádá nadřazený řídící systém je třeba obstarat čerstvě prostřednictvím OpenTherm Master rozhraní od systému pro vytápění/klimatizaci. Protože po celou tuto dobu je USART v režimu vysílání a je připraven okamžitě odvysílat odpověd, došlo by v případě ztráty OpenTherm komunikace k nekonečnému čekání a USART by se nikdy nedostal zpět do režimu přijímače. Po přetečení čítače TIM16 je v jeho obsluze přerušení transakce zrušena a USART přechází do přijímacího režimu.

7.2 Hlavní programová smyčka

Hlavní programová smyčka implementuje funkcionalitu, odvozenou v rámci návrhu s využitím nízkoúrovnových funkcí.

Program ještě před hlavní programovou smyčkou začíná zavoláním funkcí pro inicializaci hardware, které nastaví integrované periferie, výstupní porty, časovače, přerušení, USART a řadič přerušení. Dále nastaví zařízení do režimu snifferu (monitor mode), resetuje USART do režimu přijmu, nastaví OpenTherm Slave rozhraní do režimu přijmu a na OpenTherm Master zařízení nastaví IDLE režim. V implementaci režimu snifferu je tedy určitý rozdíl oproti návrhu – cyklické dotazování probíhá i v tomto režimu V praktickém použití se ukázalo, že je toto řešení lepší z důvodu aktuality dat ve vnitřní paměti.

První část hlavní programové smyčky je řízena hodnotou stavové proměnné *Mod_comm.status*. V případě, že bylo právě dokončeno vysílání Modbus odpovědi, je cyklus funkce zařízení započat od začátku nastavením USART znovu do přijímacího režimu. V případě, že došlo k timeoutu z důvodu selhání komunikace

na OpenTherm Master rozhraní, je funkční cyklus zařízení restartován stejným způsobem. V případě, že USART právě přijal datovou zprávu, je tato zpráva analyzována zavoláním funkce *mb_analyze_packet*.

Funkce *mb_analyze_packet* je základní rozhodovací funkce pro činnost zařízení iniciovanou prostřednictvím dotazů na RS485 Modbus rozhraní. Funkce také alternuje hodnoty v poli diagnostických hodnot, které definice rozhraní Modbus přikazuje podporovat. Toto pole obsahuje počítadlo zpráv na rozhraní *bus_messages*, počítadlo poškozených zpráv *bus_comm_errors*, počítadlo korektních zpráv *messages* a další počítadla, jejichž inkrementace není implementována, ale jejich přítomnost a přítomnost funkcí pro jejich čtení je nutná pro splnění specifikace Modbus. Funkce po jejím zavolání nejprve inkrementuje počítadlo všech zpráv na sběrnici *bus_messages*. Poté kontroluje správnost délky a platnost CRC zprávy. V případě, že je zpráva poškozena, inkrementuje počítadlo *bus_comm_errors*. V případě, že zpráva nemá shodnou Modbus Slave adresu s adresou nastavenou v konfiguraci rozhraní (tedy je určena pro jiné Modbus zařízení), funkce končí. Pokud je zpráva validní, funkce inkrementuje počítadlo *messages* a rozhoduje o další činnosti na základě typu Modbus příkazu. Implementovány jsou tyto příkazy:

Diagnostika Modbus komunikace (08h). V rámci diagnostiky jsou implementovány tyto podfunkce:

Vrat data požadavku (00h). V tomto případě se odesílá jako odpověd přijatá zpráva beze změny. Samotné sestavení odpovědi je provedno funkcí *mb_writecoil_response* z externího souboru *modbus.c*, která do výstupního *Modbus_out* řetězce vloží Modbus odpověd ve tvaru odpovědi s jedním polem příznaků (po bytech: adresa, kód Modbus funkce, délka dat v bytech = 1, osmibitové pole jednobitových příznaků). Funkce se volá bez parametru, v tomto případě vrací přijatou zprávu.

Vymaž počítadla (0Ah). V tomto případě se nulují všechna diagnostická počítadla Modbus komunikace a vrací se přijatá zpráva jako odpověd ve funkcí *mb_writecoil_response*.

Vrat počet zpráv na sběrnici (0Bh). V tomto případě se vrací hodnota počítadla *bus_messages* pomocí metody *mb_set_reg_response*, tedy ve tvaru odpovědi s jedním registrem (po bytech: adresa, kód Modbus funkce, délka dat v bytech = 2, MSB dat, LSB dat).

Vrat počet chybně přijatých zpráv (0Ch). tomto případě se vrací hodnota počítadla *bad_comm_errors* pomocí metody *mb_set_reg_response*.

Vrat‘ počet bezvadně přijatých a zpracovaných zpráv (0Eh). V tomto případě se vrací hodnota počítadla *messages* pomocí metody *mb_set_reg_response*.

Dále implementace diagnostických funkcí obsahuje implementace vrácení hodnot dalších počítadel, inkrementace těchto počítadel ale nikde v programu neprobíhá, tyto funkce tedy vrací vždy 0, a to vždy pomocí metody *mb_set_reg_response*.

Neimplementované funkce vrací chybovou zprávu (kód funkce + 80h) s typem chyby Nepodporovaná funkce (01h) pomocí funkce *mb_set_error* (po bytech: adresa, kód funkce +80h, typ chyby).

Další implementovaný příkaz Modbus je Čti vstupní regist (04h). V rámci této funkce se provádí více různých operací. Je-li adresa požadovaného registru v intervalu 0 až 256, funkce vrací hodnotu data ID OpenTherm odpovídající hodnotě požadovaného registru pomocí volání funkce *mb_set_reg_response*. V režimu převodníku jsou tato data obstarána čerstvě mezitransakcí na OpenTherm Master rozhraní.

Je-li adresa požadovaného registru v intervalu 1000 až 1009, je vracena hodnota z pole hodnot nastavení časovačů prostřednictvím funkce rozhraní *timing_read*. Tímto spůsobem lze ovládat časování OpenTherm komunikace i nastavení RS485 Modbus komunikace a tím dosáhnout funkce zařízení i s nestandartně časovanými zařízeními.

Je-li adresa mimo tyto rozsahy, je vrácena chybová zpráva (kód funkce +80h) s typem chyby Neplatná adresa (02h) pomocí funkce *mb_set_error*.

Další implementovaný příkaz Modbus je Čti uchovávací registry (03h). Pomocí tohoto příkazu je implementována funkce vracení více hodnot registrů v jedné odpovědi. Odpověd je generována funkcí *mb_set_big_response* (po bytech: Adresa, kód funkce, délka dat = počet odesílaných registrů*2, odesílané registry vždy v pořadí MSB, LSB).

Další implementovaný příkaz Modbus je Zapiš jeden registr (06h). Funkce funguje analogicky s funkcí Čti uchovávací registry, podporuje ale pouze zápis jednoho registru během obsluhy jedné Modbus zprávy. Pomocí této funkce lze zapisovat OpenTherm data v rozsahu adres 0 – 256 i proměnné pro časování s adresami 1000 – 1009. Odpověd je generována funkcí *mb_set_reg_response*. V režimu snifferu je zápis zakázán a funkce vrací chybovou zprávu s typem chyby Zařízení zaneprázdněno (06h) pomocí funkce *mb_set_error*.

Další implementovanou funkcí Modbus je Zapiš vstupní registry (10h). Vzhledem možným problémům s kolizemi při jednorázovém zápisu více registrů (nadřazený řídící systém by mohl být informován o zapsání před tím, než by se zápis stihl fyzicky vykomunikovat na OpenTherm Master rozhraní) je v rámci tohoto registru povolen zápis pouze jednoho registru. Obsluha této funkce tedy pracuje naprosto shodně s obsluhou funkce Zapiš vstupní registr (06h).

Posledním párem implementovaných Modbus funkcí jsou funkce Čti cívky (01h) a Zapiš jednu cívku (05h) pro obsluhu bitových příznaků. Tyto dvě funkce jsou použity pro přepínání provozního režimu zařízení. Příkazem Zapiš jednu cívku se s adresou 0 přepne zařízení do režimu převodník (Controller mode), zápisem na adresu 1 se zařízení přepne do režimu Passive monitor (rezervováno a prozatím nepoužito, příprava na další funkcionalitu), zápisem na adresu 2 se zařízení přepne do režimu snifferu (Monitor mode). Odpověd se generuje funkcí *mb_writecoil_response,* která kopíruje přijatou datovou zprávu. V případě pokusu o zápis na jinou adresu je generována chybová zpráva funkcí *mb_set_error* s typem chyby Neplatná adresa (02h). Funkce Čti cívky vrací při čtení na adrese 0 hodnoty prvních osmi bitových příznaků, které obsahují tři příznaky využité v přepínání režimů, a to pomocí funkce *mb_set_coil_response*. Při pokusu o čtení jíné adresy vrací chybovou zprávu s typem chyby Neplatná adresa (02h).

Na všechny ostatní Modbus funkce zařízení odpovídá chybovou zprávou s typem chyby Nepodporovaná funkce (01h).

Funkce *mb_analyze_packet* má návratovou hodnotu podle akce, kterou je třeba po analýze provést. Funkce vrací hodnotu 1, pokud je možné ihned odpovědět Modbus odpovědí. Funkce vrací hodnotu 2, pokud je třeba provést OpenTherm Master mezitransakci pro získání čerstvých dat. V případech kdy dojde k chybě, Modbus dotaz je určen pro jiné zařízení nebo není třeba na rozhraní Modbus odpovídat, funkce vrací 0. Algoritmus hlavní programové smyčky se podle této návratové hodnoty příslušně zachová nastavením rozhraní USART nebo nastavením požadavku na OpenTherm Master komunikaci.

Druhá část programové smyčky je řízena podle hodnoty stavové proměnné OpenTherm Slave rozhraní *OT_thermo_comm.status*. Pokud přijde příkaz ze strany OpenTherm Slave, a zařízení je v režimu snifferu, je dotaz odeslán na OpenTherm Master rozhraní, je aktivována mezitransakce a OpenTherm Slave rozhraní přechází do režimu čekání. Zbytek transakce se pak řídí v závislosti na doběhnutí komunikace na OpenTherm Master rozhraní, případě timeouty OpenTherm Master. Je-li typ OpenTherm zprávy Zapiš data, je daty ze zprávy aktualizována vnitřní pamět. Pokud je přijatá OpenTherm Slave zpráva poškozená nebo neplatná, OpenTherm Slave komunikace je restartována přepnutím rozhraní do stavu WAIT.

Poslední část programové smyčky je řízena podle hodnoty stavové proměnné OpenTherm Master rozhraní *OT_boil_comm.status*.

Pokud je OpenTherm Master zařízení ve stavu IDLE a existují požadavky na mezitransakci z OpenTherm Slave rozhraní (režim snifferu), nebo RS485 Modbus rozhraní (režim převodníku), je příslušná mezitransakce odstartována. V opačném případě je zahájen dotaz na OpenTherm Master rozhraní v rámcí cyklického dotazování v IDLE režimu.

Bylo-li právě dokončeno vysílání požadavku, je OpenTherm Master rozhraní přepnuto do režimu příjmu. Do vnitřní paměti dat je do volného okna mezi data ID 58 až 100 na adresu 99 uložena hodnota časovače TIM3 z diagnostických důvodů.

Byl-li právě dokončen příjem odpovědi, z diagnostických důvodů se na adresy 98 a 97 ukládají hodnoty časovačů TIM15 a TIM3 a přijatá data se spracovávají. Pokud je datová zpráva typu Potvrzuji čtení dat, je daty ze zprávy aktualizována vnitřní pamět. Pokud byla přijatá odpověd součástí mezitransakce pro požadavek z RS485 Modbus rozhraní a je platná, je z přijaté odpovědi zkonstruována Modbus odpověd a zahájeno její odesílání. Pokud je odpověd poškozená nebo neplatná, odesílá se na RS485 Modbus rozhraní chybová zpráva voláním funkce *mb_set_error* s typem chyby Selhání zařízení (04h). Pokud byla přijatá odpověd součástí mezitransakce pro OpenTherm Slave požadavek (režim snifferu), je započato odesílání odpovědi na OpenTherm Slave rozhraní. Pokud byla přijatá odpověd odpovědí na cyklický dotaz v IDLE režimu, je daty aktualizována vnitřní pamět.

Pokud došlo k timeoutu v průběhu přijmu a přijatá odpověd měla být součástí odpovědi pro RS485 Modbus rozhraní, odesílá se na RS485 Modbus chybová zpráva pomocí volání funkce *mb_set_error* s typem chyhy Selhání zařízení (04h).

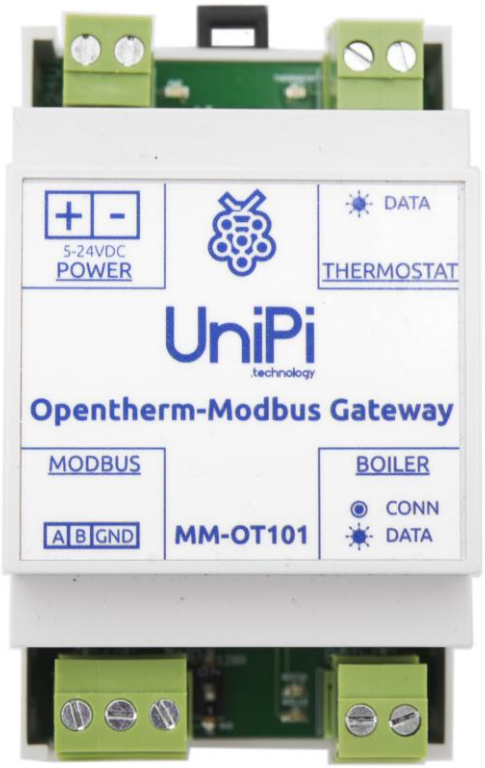
Pokud dojde k naplnění času timeoutu ve stavu WAIT, je rozhraní OpenTherm Master přepnuto do stavu IDLE, a je zahájeno cyklické dotazování.

Tímto je kompletně naplněna implementace funkcionality navržené v rámci návrhu.

8 Testování a zhodnocení

Původní verze firmwaru vznikla před více než rokem a půl, tedy v květnu roku 2015. Vývoj probíhal poněkud nestardantním způsobem, kdy já jsem byl ve svém bydlišti v Olomouci vybaven pouze vývojovým kitem bez dalšího hardwaru a bez možnosti kód prakticky testovat a veškeré testy probíhaly v Brněnské vývojové laboratoři firmy Faster.cz, s.r.o [23], která projekt iniciovala a kam jsem emailem odesílal zdrojové kódy firmwaru, které tam překládali, nahrávali do svého vývojového kitu s připojeným extermím hardware a prováděli testování v reálných podmínkách. Z tohoto důvodu se v kódu objevuje nezanedbatelné množství částí pro umožnění nejrůznějšího testování.

Verze kódu prezentovaná v této práci je verzí pozdní, která je odladěná v reálných podmínkách v interakci se skutečným hardwarem a je funkční. Neustále však probíhalo a probíhá pozměnování funkčnosti a doimplementovávání nových vlastností. Zařízení je nyní (listopad 2016) v předprodukční fázi a je uvolněno pro testování u prvních zákazníků. Z výstupů testování lze konstatovat, že zařízení je schopné zajistit funkcionalitu, pro jejíž provádění bylo navrženo a v praktickém použití se osvědčilo.



Obr. 41 Zařízení v předproduční fázi. Zdroj: http://unipi.technology

9 Závěr

Tato práce si klade za cíl provést rešerší existujících technologií, komunikačních protokolů a způsobů komunikace v oblasti dálkových číslicově řízených systémů vytápění, navrhnout řešení převodníku mezi protokoly Opentherm a RS485 Modbus, implementovat návrh řešení na zvolené hardwarové platformě s ARM procesorem, navržené řešení otestovat a zhodnotit.

Rešerše existujících technologií, protokolů a způsobů komunikace byla provedena a ukázala nezáviděníhodnou situaci na poli dálkově řízených číslicových systémů vytápění a klimatizace. Ukázala, že jedno z případných řešení vyžaduje mimo jiné konstrukci převodníku mezi protokoly OpenTherm a RS485 Modbus. Dále je podrobně rozebrán princip funkce protokolů OpenTherm a RS485 Modbus a prezentována platforma, vývojový kit a vývojové prostředí v rámci projektu použité. S využitím těchto informací je vypracován návrh řešení firmware převodníku mezi OpenTherm a RS485 Modbus rozhraními, tento návrh je implementován a implementace je podrobně vysvětlena. Následně je diskutován způsob testování převodníku, a výsledky tohoto testování.

Lze konstatovat, že cílů vytýčených v zadání této práce bylo dosaženo. Výstupem této práce je funkční firmware převodníku a snifferu pro OpenTherm a RS485 Modbus, který je momentálně (prosinec 2016) komerčně dostupný zákazníkům.

10 Literatura

1.VODA, ZBYŠEK. *Průvodce světem Arduina.* Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

2.GREENGARD, SAMUEL. *The internet of things.* Cambridge, Massachusetts: MIT Press, 2015. ISBN 9780262527736.

3.HARPER, RICHARD. *Inside the smart home.* New York: Springer, c2003. ISBN 1852336889.

4.LAWRENZ, WOLFHARD. *CAN system engineering: from theory to practical applications.* New York: Springer, c1997. ISBN 0387949399.

5.*The OpenTherm association* [online]. [cit. 2016-12-27]. Dostupné z: https://www.opentherm.eu/association/

6.*Topné, průmyslové a chladící systémy | Viessmann Česká republika* [online]. [cit. 2016-12-27]. Dostupné z: http://www.viessmann.cz/

7.*VITODENS 100, 200: Návod k obsluze pro provozovatele zařízení* [online]. 5595 398 CZ. 2007 [cit. 2016-12-27]. Dostupné z: http://public.vitoservis.cz/ManualDownload2.aspx

8.*VITOTROL 100 OT: Návod k obsluze pro provozovatele zařízení* [online]. 5619 445 CZ. 2016 [cit. 2016-12-27]. Dostupné z: http://public.vitoservis.cz/ManualDownload2.aspx

9.*Kotle a tepelná čerpadla Vaillant* [online]. [cit. 2016-12-27]. Dostupné z: https://www.vaillant.cz/pro-zakazniky/

10.*Vaillant ecoTEC: Instructions for installation and servicing* [online]. 839592_03 GB, 2006 [cit. 2016-12-27].

11.*Thermona: plynové kondenzační kotle, elektrokotle a kaskádové kotelny* [online]. [cit. 2016-12-27]. Dostupné z: http://www.thermona.cz/

12.*Katalog produktů: Thermona* [online]. KP. 2016 [cit. 2016-12-27]. Dostupné z: http://www.thermona.cz/Thermona/media/content/Dokumentace/Katalog%20produktu/Katalog_produktu_2016-10.pdf

13.*CR04 - POKOJOVÁ JEDNOTKA S MODULAČNÍM PROGRAMOVATELNÝM REGULÁTOREM: Uživatelská příručka* [online]. 2007 [cit. 2016-12-27]. Dostupné z: viadrus.cz/doc/cms_library/cr-11006-722.pdf

14.*Prostorový termostat pro regulaci kotlů s komunikací OpenTherm PT59: Návod* [online]. V1114. 2014 [cit. 2016-12-27]. Dostupné z: https://www.elektrobock.cz/pt59-navod/f280

15.BEZPALEC, PAVEL. *Vrstvové protokoly* [online]. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická [cit. 2016-12-27]. Dostupné z: http://data.cedupoint.cz/oppa_e-learning/2_KME/146.pdf

16.KOZIEROK, CHARLES M. *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. San Francisco: No Starch Press, c2005. ISBN 15-932-7047-X.

17.WALMSLEY, PRISCILLA. *Definitive XML Schema: a comprehensive, illustrated Internet protocols reference*. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, c2013. Charles F. Goldfarb definitive XML series. ISBN 978-013-2886-727.

18.WALMSLEY, PRISCILLA. *Understanding and using telnet*. 2nd ed. S.l.: Delmar, 1996. Charles F. Goldfarb definitive XML series. ISBN 978-082-7380-202.

19.KUGELSTADT, THOMAS. *The RS-485 design guide* [online]. SLLA272C. Texas Instruments, 2008 [cit. 2016-12-27]. Dostupné z: http://www.ti.com/lit/an/slla272c/slla272c.pdf

20.RONEŠOVÁ, ANDREA. *Přehled protokolu MODBUS* [online]. Západočeská univerzita v Plzni, 2005 [cit. 2016-12-27]. Dostupné z: http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf

21.*The OpenTherm Communications Protocol - Protocol Specification: A Point-to-Point Communications System for HVAC Controls* [online]. V2.2. The OpenTherm Association, 2003 [cit. 2016-12-27]. Dostupné z: https://www.domoticaforum.eu/uploaded/Ard%20M/Opentherm%20Protocol%20v2-2.pdf

22.*CooCox: CoIDE - Free IDE for ARM Cortex-M Desing* [online]. [cit. 2016-12-27]. Dostupné z: http://www.coocox.org/software/coide.php

23.*Faster.cz, s.r.o.: be faster* [online]. [cit. 2016-12-27]. Dostupné z: http://faster.cz/cs/

24.*RM0091 Reference manual: STM32F0x1/STM32F0x2/STM32F0x8 advanced ARM®-based 32-bit MCUs* [online]. 018940-Rev.8. ST Microelectronics, 2015 [cit. 2016-12-27]. Dostupné z: http://www.st.com/content/ccc/resource/technical/document/reference_manual/c2/f8/8a/f2/18/e6/43/96/DM00031936.pdf/files/DM00031936.pdf/jcr:content/translations/en.DM00031936.pdf

25.*STM32F0DISCOVERY - Databrief: Discovery kit for STM32F051xx microcontrollers* [online]. 022931-Rev.2. ST Microelectronics, 2014 [cit. 2016-12-27]. Dostupné z: http://www.st.com/content/ccc/resource/technical/document/data_brief/e2/98/be/f9/3c/6a/4e/91/DM00050631.pdf/files/DM00050631.pdf/jcr:content/translations/en.DM00050631.pdf