Masarykova univerzita
Fakulta informatiky


# Měření a sběr elektrických veličin s využitím mesh sítě

Diplomová práce

**Vít Holásek**

Brno, jaro 2018

# Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vít Holásek

**Vedoucí práce:** RNDr. Zdeněk Matěj, Ph.D.

# Poděkování

Rád bych touto cestou poděkoval RNDr. Zdeňku Matějovi, Ph.D za vstřícnost, odborný dohled a zkušenosti, které mi pomohly k vypracování této práce. Dále děkuji vedení a kolegům z firmy TESCO SW a.s. za poskytnutí podpory, nástrojů a prostředků, nezbytných pro realizaci diplomové práce.

# Shrnutí

Tato práce popisuje návrh a realizaci systému pro monitoring spotřebičů v budovách. Základem je zařízení umožňující měření elektrických veličin na spotřebiči a jejich odesílání k dalšímu zpracování. Jednotlivá zařízení tvoří bezdrátovou síť, která umožňuje komunikaci s již existující cloudovou platformou. Část textu se věnuje analýze nejpoužívanějších bezdrátových sítí pro Internet věcí a výběrem vhodné technologie pro realizaci zadání.

# Klíčová slova

Internet věcí, embedded systémy, měření energií, bezdrátové sítě, mesh sítě, Thread, KW41Z, Microsoft Azure

# Obsah

# Úvod

V rámci své bakalářské práce v roce 2016 [27] jsem se zabýval návrhem zařízení pro měření elektrických veličin na spotřebiči, které komunikuje prostřednictvím Wi-Fi sítě. Výhodou takového řešení je skutečnost, že Wi-Fi síť dnes pokrývá většinu domácností a budov. Ke svému provozu tak zařízení nepotřebuje žádné další síťové prvky. Je vhodné pro případy užití, kdy uživatel potřebuje sledovat a ovládat malý počet spotřebičů. Tato zařízení však mají vyšší nároky na spotřebu a výpočetní výkon pro zabezpečení komunikace. Tím se zvyšuje konstrukční složitost, velikost i cena.

Řešení s použitím Wi-Fi sítě však není úplně vhodné pro systémy tvořené větším počtem komunikujících uzlů. Pro ty je dnes v oblasti Internetu věcí k dispozici mnoho specializovaných protokolů. Ty často umožňují vytváření tzv. mesh sítí. Zařízení je tak schopné komunikovat s uzly, které nejsou fyzicky v dosahu jeho radiového vysílání, skrze jiná zařízení v dosahu. Tato práce se zabývá použitím právě takových sítí.

Existuje několik rozšířených protokolů které mají své výhody a nevýhody. Spousta velkých výrobců v oblasti elektroniky si vyvíjí vlastní řešení a poskytuje vlastní hardware pro jejich použití. Sítě pro Internet věcí jak v oblasti LAN[1] i WAN[2] jsou zatím nejednotné a neexistuje jediný široce používaný standard. Kvůli této skutečnosti je poměrně složité propojovat různorodá řešení do celků. Takové sítě potřebují pro komunikaci s globální internetovou sítí jeden nebo více přístupových bodů, které poslouží jako výchozí brána do internetu. Taková zařízení se označují pojmem gateway nebo také hraniční zařízení. Jednotliví výrobci často poskytují pro své systémy vlastní hardware. Brána může být také využita k integraci a automatizaci připojených zařízení a zároveň může lokálně uchovávat a zpracovávat data ještě předtím, než je odešle na server. Řešení pak může snížit nároky na serverovou infrastrukturu.

Hlavním přínosem této práce by mělo být použití moderních a perspektivních technologií, které mohou poskytnout konkurenční vý-

[1] LAN – Local Area Network
[2] WAN – Wide Area Network

hodu oproti překonaným a složitým řešením. Cílem je tyto technologie prozkoumat a poznatky využít pro návrh a realizaci prototypu unikátního systému pro monitoring velkého množství elektrických spotřebičů v budovách. Systém by mělo být možné využívat jak v novostavbách, tak v již stojících budovách bez větších zásahů do elektroinstalace. Základními atributy výsledku by měly být otevřenost, přenositelnost a především jednoduchost z pohledu používání. Systém navíc musí být rozšiřitelný o další typy zařízení (například pro sledovaní teploty). Nakonec by mělo být řešení otestováno a vyhodnoceno, zda je použitelné v požadovaných podmínkách

Dalším důležitým aspektem je zabezpečení komunikace. Zanedbaná bezpečnost je stále velkou slabinou Internetu věcí. S exponenciálně rostoucím počtem různých zařízení v síti roste také motivace útočníků je napadat. V poslední době jsou evidovány stále masivnější útoky a na korektní zabezpečení je tedy potřeba myslet již při návrhu systému [2].

Práce je realizována ve spolupráci se společností TESCO SW a.s. v rámci projektu Hexade zaměřeného na chytré domácnosti a monitoring energií [13].

# 1 Požadavky na řešení

Základem celého systému budou zařízení umožňující měření elektrických veličin na spotřebiči. Pro odlišení stávající produktové řady Smart Plug je zařízení označeno pracovním názvem *Smart Socket*. Pro účely této práce budeme dále používat pojem *zásuvka*. Požadované měřené veličiny jsou: napětí v elektrické síti, odběr proudu, příkon spotřebiče a spotřeba energie. Zařízení bude schopno bezdrátově přenášet měřená data s periodou v řádu několika vteřin. Požadavkem je také možnost co největší miniaturizace, aby zařízení bylo možné zabudovat přímo do zásuvky elektrické sítě. Předpokládá se, že zařízení budou umístěna v budově rovnoměrně v každé místnosti. Cílovou skupinou jsou malé a střední podniky, domácnosti nebo bytové domy. Cena výsledného produktu by neměla výrazně navýšit náklady na elektroinstalaci. Systém musí být možné instalovat i ve starších budovách bez nutnosti přestavby a větších zásahů do elektroinstalace.

Komunikační síť musí zajistit spolehlivý přenos dat mezi zařízeními. Vhodným řešením je síť umožňující decentralizovanou topologii mesh. Síť by měla v budoucnu umožnit i napojení dalších typů zařízení včetně takových, které budou napájeny pouze z baterie. Počet připojených uzlů by se měl pohybovat nejvýše v řádu stovek.

Součástí systému bude centrální stanice, která bude dále označována pojmem *gateway*. Zařízení gateway umožní zprostředkovat komunikaci s internetovou sítí. Bude navrženo tak, aby jej bylo možné rozšiřovat o další komunikační protokoly a funkcionality, jako je propojení s domácími hlasovými asistenty apod. Bude zároveň sloužit jako centrum domácí automatizace. Musí být také schopné zprostředkovat komunikaci mezi jednotlivými zařízeními v síti a existující cloudovou platformou založenou na službě Azure IoT Hub [44].

# 2 Analýza komunikačních sítí

V oblasti Internetu věcí vniklo v posledních letech několik různých sítí. Obecně je rozlišujeme dle dosahu a použití. Ze skupiny WAN bezdrátových sítí s dosahem v řádu kilometrů od vysílače jsou v České republice většinově pokryté technologie LoRaWAN a Sigfox [25]. Ty se hodí především pro bateriově napájená zařízení mimo dosah lokálních komunikačních sítí. Neumožňují efektivní přenos objemnějších dat a komunikaci v reálném čase s nízkou odezvou. Je to však dobrá alternativa ke klasickým GSM modulům.

Ze sítí v rozsahu MAN[1] lze uvést českou technologii IQRF [18]. Ta umožňuje efektivní přenos dat na vzdálenosti řádově stovek metrů s minimální spotřebou. Použití IQRF se osvědčilo například ve městech v aplikacích, jako jsou senzory pro parkování.

Pro účely této práce jsou vhodnější bezdrátové sítě ze skupiny LAN nebo PAN[2], kterým se bude tato kapitola dále věnovat. Jedná se o způsoby bezdrátové komunikace na krátkou vzdálenost. Mezi nejznámější patří Z-Wave [26], Zigbee [43], Bluetooth a Thread [63]. Důležitým aspektem těchto sítí jsou nízké nároky na spotřebu energie a výpočetní zdroje nezbytné pro zajištění zabezpečené komunikace.

V následujícím textu se bude často vyskytovat pojem mesh sítí. Někdy se také nazývají bezdrátové ad hoc sítě. Ve zkratce se jedná o decentralizovanou síť, jejíž topologie se ustavuje dynamicky v závislosti na rozmístění komunikujících uzlů. Většinou jsou navrženy tak, aby neměly jediný bod selhání, tedy fungovaly i při výpadku některého z uzlů. Důležitou vlastností je, že uzel může komunikovat s jiným uzlem, který není v dosahu jeho rádiového vysílače, prostřednictvím uzlů ležících mezi nimi.

Pro úplnost je potřeba zmínit také přenos dat voděným médiem. V případě chytrých zásuvek se jeví jako nejjednodušší řešení komunikace Power Line (PLC) [48]. Nebylo by potřeba budování dodatečné komunikační infrastruktury. Takové řešení by však nebylo použitelné pro další typy zařízení, které nebudou přímo připojeny do elektrické sítě. V novostavbách, kde se s automatizací počítá od začátku, je obvykle upřednostňována komunikace po vodičích místo bezdrátové.

---

[1] MAN – Metropolitan Area Network

[2] PAN – Personal Area Network

## 2.1 Dostupné technologie

### 2.1.1 Z-Wave

Jednou z nejpoužívanějších technologií WPAN[3] je síť Z-Wave vyvinutá dánskou společností Zensys. Primárně cílí na domácí automatizaci. Z-Wave standard je spravován uskupením Z-Wave Alliance, do kterého se zapojilo více než 300 společností. Mezi přední členy patří například čínská firma Huawei, LG nebo Honeywell. Vlastníkem licence k technologii Z-Wave je společnost Sigma Designs [26].

Síť komunikuje v bezlicenčním pásmu v oblasti pod 1 GHz. Konkrétní frekvence se liší podle legislativy dané země. Pro přenos dat se používá FSK[4] modulace. Uváděný dosah je až 36 metrů, ve volném prostoru i 100 metrů. V budovách se doporučuje dodržovat vzdálenosti mezi uzly do 10 metrů. Síť umožňuje velikost toku dat nejvýše 100 kb/s.

Síť podporuje topologii typu mesh a lze do ní zapojit nejvýše 232 zařízení. K topologii sítě se vztahují dva typy identifikátorů. Home ID je 32bitový identifikátor sítě jako celku. Node ID je 8bitový identifikátor zařízení v síti. Uzly sítě s různým Home ID spolu nemohou za normálních podmínek komunikovat, protože jednotlivé sítě jsou vzájemně izolované. Každý uzel si udržuje tabulku sousedních uzlů, kterou dynamicky přizpůsobuje dle topologie sítě.

Komunikující uzly mohou mít tři různé role – *Controller*, *Slave* nebo *Routing Slave*. Zařízení typu Controller má předdefinované Home ID, Slave jej dostane přiděleno v momentě, kdy se přidá do dané sítě během procesu párování. Controller také přidělí Slave uzlu jeho Node ID. Tento proces se nazývá inkluze. Při odebrání zařízení ze sítě se smaže jeho Home ID i Node ID, dojde k tzv. exkluzi. Uzel typu Controller má informaci o sousedních uzlech a uchovává si kompletní směrovací tabulku. Může komunikovat se všemi uzly, které jsou dosažitelné v rámci topologie sítě. Slave má informaci o všech sousedních uzlech a neuchovává si směrovací tabulku. Může pouze odpovídat uzlu, ze kterého přijal zprávu. Nemůže posílat nevyžádané zprávy. Routing Slave si uchovává částečnou směrovací tabulku, může odpovědět uzlu od kterého přijal zprávu a může poslat nevyžádanou zprávu určitým

[3] WPAN – Wireless Personal Area Network
[4] FSK – Frequency-shift Keying

předdefinovaným uzlům. Typická aplikace zařízení typu Slave je například ovladač světla nebo topení. Různé senzory, které potřebují odesílat data, musí být typu Routing Slave [67].

Od roku 2016 je definován nový standard zabezpečení Security 2, který plně zabezpečuje proces párování a komunikaci mezi uzly. Zabezpečení dle předchozí specifikace se podařilo prolomit. Při procesu párování dojde k výměně klíče pro symetrické šifrování AES. Bezpečná výměna klíče je zajištěna pomocí Diffieho-Hellmanova protokolu s využitím eliptických křivek (ECDH) [24].

Jednotlivé sítě se dají propojit (*bridge*) do větších celků. Největší výhodou tohoto protokolu je komunikace v nezarušeném pásmu a vzájemná interoperabilita mezi produkty různých výrobců. Signál s delší vlnovou délkou lépe proniká přes překážky. Existuje řešení podporující adresaci IPv6 (Z/IP) a software pro výchozí bránu umožňující komunikaci s externí sítí. Samotná síť však IP protokol nevyužívá. Specifikace protokolu je veřejně dostupná a k dispozici jsou otevřené nástroje pro vývoj jednotlivých zařízení i samotné brány [23].

### 2.1.2 Zigbee

Zigbee je další komunikační technologie WPAN. První revize standardu je platná od roku 2004. V roce 2002 vzniklo uskupení Zigbee Alliance, do kterého je nyní zapojeno více než 60 významných firem. Mezi ty nejznámější patří Huawei, NXP, Legrand, Philips, Honeywell a další [43].

Síť pracuje v bezlicenčním pásmu 868 MHz, 915 MHz nebo 2,4 GHz. Rychlost přenosu dat je 20-250 kb/s. Uváděný dosah radiového vysílání je 10 až 100 metrů v závislosti na prostředí. Fyzická vrstva používá kódování DSSS [5] s modulací BPSK[6] nebo OQPSK[7] (pro 2.4 GHz). Fyzická vrstva a MAC[8] vychází ze standardu IEEE 802.15.4 [64]. Síťová a transportní vrstva je definována Zigbee Alliance a existuje několik různých verzí protokolů. Specifikace dále rozšiřuje funkcionalitu o síťovou a aplikační vrstvu. Síťová vrstva řeší směrování, bezpečnost, samoopravnost, samoorganizovatelnost a poskytuje rozhraní pro apli-

[5] DSSS – Direct Sequence Spread Spectrum
[6] BPSK – Binary Phase-shift Keying
[7] OQPSK – Offset Quadrature Phase-shift Keying
[8] MAC – Medium Access Control

kace s topologií hvězda, strom nebo mesh. Specifikace aplikační vrstvy definuje takzvané aplikační profily a Zigbee device objekty (ZDO). Pro adresaci zařízení se používá 64bitová MAC adresa. Ke směrování slouží zkrácená 16bitová síťová adresa. Využívají se algoritmy AODV[9] a DSR[10]. Teoreticky je počet uzlů v síti omezen počtem možných adres, tedy 65 536. V praxi ale závisí reálný počet zařízení na topologii sítě a použitém směrovacím protokolu.

Z pohledu funkčnosti se zařízení klasifikují jako plně funkční zařízení (FDD) a zařízení s redukovanou funkčností (RFD). RFD jsou obvykle většinu času v režimu spánku. Jsou definovány 3 typy komunikačních uzlů – *Coordinator*, *Router* a *End Device*. Coordinator je jedinečný správce sítě, který směruje toky dat sítí. V sítích typu hvězda nebo strom je možný bezkolizní přístup v časových dílech přidělených jednotlivým koncovým zařízením. V síti s topologií mesh je umožněn přístup k médiu pouze pomocí soupeření a nedochází k synchronizaci zařízení. Router musí kdykoli přijímat data, tedy je neustále aktivní. Zařízení s rolí End Device mohou spát a vysílat pouze v případě potřeby. Router uchovává data pro sousední zařízení typu End Device, ta si je po probuzení vyžádají. Samotná komunikace je zabezpečena pomocí symetrické šifry AES-128.

Zigbee je technologie používaná převážně v průmyslu [7]. Specifikací jsou definovány základní aplikační profily pro časté případy užití jako automatizace budovy, zdravotní péče, domácí automatizace vzdálená kontrola a další. Profily umožňují interoperabilitu zařízení stejného typu. Kompletní specifikace je otevřená a veřejně dostupná. Opět existují softwarová řešení pro výchozí brány a zajištění adresace IPv6. Poslední revize Zigbee 3.0 zásadním způsobem mění aplikační profily za účelem lepší interoperability [66].

### 2.1.3 Thread

Další síť navržená primárně pro Internet věcí se nazývá Thread. Specifikace byla poprvé představena v roce 2014 uskupením Thread Group. Do vývoje nové technologie se zapojily velké společnosti jako Nest

[9] AODV – Ad hoc On-Demand Distance Vector
[10] DSR – Dynamic Source Routing

(patří do holdingu společnosti Google), Microsoft, Arm, Samsung, Philips, NXP a desítky dalších [42].

Síť Thread je také postavena na standardu pro bezdrátový přenos dat IEEE 802.15.4 [63]. Zároveň implementuje technologii 6LoWPAN[11] umožňující transformaci paketů IPv6 pro přenos v rámci IEEE 802.15.4. Narozdíl od ostatních zmíněných je Thread síť už ze samotného návrhu založena na protokolu IPv6. Thread pracuje v bezlicenčním pásmu 2,4 GHz a používá modulaci OQPSK. Maximální rychlost přenosu dat je 250 kb/s. Udávaný dosah je 30 metrů, na volném prostranství může být i více než 100 metrů. Oproti Zigbee však není specifikací striktně definována aplikační vrstva. Lze tak použít libovolnou aplikaci nad UDP protokolem. Příkladem může být protokol CoAP [57].

Největšími přednostmi sítě Thread jsou bezpečnost, jednoduchost a implicitní podpora IPv6 a protokolu UDP. Existuje otevřená portovatelná knihovna OpenThread [36] vyvíjená společností Nest, která implementuje protokol Thread. Součástí je také implementace hraničního zařízení tvořícího bránu do Internetu. Umožňuje obousměrné propojení sítě Thread s lokální IP sítí a Internetem. Díky podpoře mDNS[12] je možné navázat komunikaci s libovolnými servery v Internetu.

### 2.1.4 Bluetooth mesh

Pro úplnost jsou v tomto textu zmíněny také technologie Bluetooth mesh [29] a Wi-Fi HaLow [7]. Problémem starších verzí protokolu Bluetooth je krátký dosah radiového vysílání a vyšší nároky na spotřebu energie. Je také limitován maximální počet *slave* uzlů připojených do sítě. To se změnilo s příchodem standardu Bluetooth 4.0 neboli BLE (Bluetooth Low Energy). BLE zásadním způsobem snižuje nároky na výkon a umožňuje efektivní přenos menšího objemu dat. Technologie není zpětně kompatibilní. S verzí 4.2 se přidala také podpora standardu 6LoWPAN v rámci nového profilu *Internet Protocol Support*. K nové verzi Bluetooth je specifikací definován i nový profil pro mesh sítě [29].

[11] 6LoWPAN – IPv6 over Low-Power Wireless Personal Area Networks
[12] mDNS – Multicast Domain Name System

Technologie BLE komunikuje v bezlicenčním pásmu 2,4 GHz. Udávaný dosah je více než 100 metrů v otevřeném prostoru. Reálně však do 30 metrů, v budovách kolem 10 metrů. Výstupní výkon vysílání se snížil z původně až 1 W u starších standardů na 1 mW, čímž se vyrovnává s ostatními technologiemi zmíněnými v této práci. Profil umožňuje přenos dat rychlostí 125 kb/s.

Bluetooth mesh se obecně hodí pro menší počet zařízení rovnoměrně rozmístěných na menším prostoru. Využívá se spíše v průmyslu. Lze očekávat další zlepšení vlastností komunikace v rámci nové specifikace Bluetooth 5.

### 2.1.5 Wi-Fi HaLow

V oblasti sítí LAN pro komunikaci mezi počítači jsou dnes masově rozšířené standardy ze skupiny IEEE 802.11 neboli Wi-Fi. Umožňují vysokou propustnost pro přenos dat, to však přináší vysoké nároky na výkon komunikujících uzlů. Z toho důvodu je Wi-Fi nevhodné pro aplikace v zařízeních z oblasti Internetu věcí napájená především z baterií.

Uskupení firem Wi-Fi Alliance v roce 2017 představilo nový standard IEEE 802.11ah nazvaný Wi-Fi HaLow. Pracuje v bezlicenčním pásmu v oblasti pod 1 GHz. Opět byl navržen pro snížení energetických požadavků a jeho ambicí je konkurovat technologii BLE. Protokol se však zatím neprosadil a na trhu ještě není žádné zařízení, které by jej používalo [7].

## 2.2 Preferované řešení

Z pohledu nároků na výkon a hardwarové zdroje jsou všechny analyzované způsoby komunikace úsporné. Toto kritérium je důvodem jejich vzniku, proto není pro výběr určující. Z-Wave a volitelně Zigbee mohou komunikovat na frekvencích pod 1 GHz, Thread a Bluetooth využívají pásmo 2,4 GHz. Signál s větší vlnovou délkou (tedy nižší frekvencí) má menší útlum a lépe prochází přes překážky. Pro vysílání je však potřeba větší anténa a s nižší frekvencí klesá také rychlost přenosu dat. Nevýhodou pásma pod 1 GHz je, že frekvence, šířka pásma a legislativa není jednotná a liší se v různých částech světa. Volné

pásmo ISM 2,4 GHz je naopak jednotné. Bývá však sdílené dalšími sítěmi jako Wi-Fi a tedy obsahuje více potencionálního rušení. V závislosti na útlumu signálu je dosah přibližně dvakrát menší. Při vhodném rozmístění komunikujících uzlů v budově a využití sítě s topologií mesh nehraje frekvence zásadní roli a všechna řešení jsou schopna zajistit dostatečné pokrytí.

Technologie Z-Wave a Zigbee jsou z pohledu funkcionality podobné. Jsou na trhu už několik let a bylo certifikováno mnoho zařízení využívající zmíněné způsoby komunikace. Zigbee cílí více na průmysl a definuje vyspělejší ale komplikovanější aplikační vrstvu. Z-Wave je navržen spíše pro chytré domácnosti. Bluetooth komunikace se vyskytuje u mnoha produktů, použití profilu pro mesh sítě však není běžné a neposkytuje žádné zásadní výhody. WiFi HaLow se zatím na trhu u žádných zařízení nevyskytuje, ani nejsou dostupné žádné širší informace, proto tuto technologii zatím nemá smysl uvažovat.

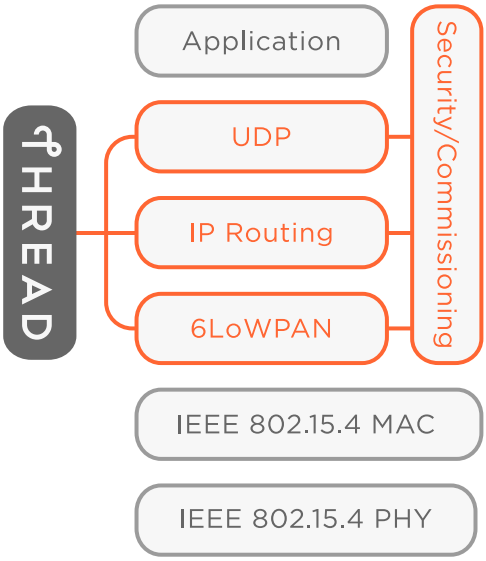
Síť Thread zatím není využívána masově. Jejího vývoje se však účastní významné společnosti a je použita v produktech Nest. Na trhu už je dostupné dostatečné spektrum modulů různých výrobců podporujících komunikaci Thread. Díky shodné fyzické vrstvě byl protokol implementován dodatečně i na zařízení původně používaná pro Zigbee. Zásadním rozdílem oproti ostatním technologiím je implicitní použití 6LoWPAN. Lze tedy snadno propojit síť s jinými sítěmi jako Wi-Fi, Ethernet nebo 4G LTE. Zařízení také mohou snadno komunikovat se službami v Internetu. Do budoucna lze předpokládat větší rozšíření protokolu IPv6, tudíž je tato technologie perspektivní. Přínosem je dále jednoduchost celého návrhu, hodí se tak i pro chytré domácnosti a aplikace cílené na koncové uživatele. Nevýhodou i výhodou zároveň může být skutečnost, že není striktně definována aplikační vrstva. Nelze tedy garantovat interoperabilitu zařízení různých výrobců. Thread ale poskytuje větší volnost při vývoji aplikací a přibližuje jejich návrh síťovým aplikacím na běžných počítačích. Tato technologie se jeví pro účely práce jako nejvhodnější.

Nevýhodou všech zmíněných technologií je skutečnost, že nelze tyto sítě jednoduše vzájemně propojovat do jednoho celku. Pro každou síť je potřeba samostatná výstupní brána. Probíhá však jistá spolupráce mezi Zigbee a Thread [65]. Aplikační vrstva Zigbee a její aplikační rozhraní bylo přeneseno na síťový model protokolu Thread.

# 3 Thread síť

## 3.1 Síťový model

Obrázek 3.1 znázorňuje síťový model protokolu Thread. Návrh je přímočarý a využívá přístupy osvědčené u jiných sítí. Následující text se věnuje podrobnějšímu popisu fungování jednotlivých vrstev modelu.




Obrázek 3.1: Síťový model protokolu Thread (zdroj: [63])

### Fyzická vrstva a vrstva datového spoje

Fyzická vrstva a vrstva datového spoje jsou definovány dle standardu IEEE 802.15.4, který byl vyvinut pro bezdrátovou komunikaci pomalejších zařízení s nízkou spotřebou. Fyzická vrstva komunikuje v bezlicenčním pásmu ISM 2,4 GHz. Pro přenos dat se používá modulace OQPSK, výstupní výkon vysílání je do 1 mW. Umožňuje komunikaci mezi dvěma uzly do vzdálenosti 30 metrů (v otevřeném prostoru i více

než 100 metrů) s maximální rychlostí přenosu dat 250 kb/s. Pro řízení přístupu k médiu v topologii mesh se používá protokol CSMA/CA. K adresaci mezi dvěma uzly slouží 64bitový jednoznačný identifikátor EUI-64. Dle IEEE 802.15.4 rozlišujeme 2 typy uzlů na základě jejich funkcionality - plně funkční zařízení (FFD) a zařízení s redukovanou funkčností (RFD), které na rozdíl od FFD může většinu času trávit ve spánku a probouzí se pouze když potřebuje komunikovat.

Vrstva datového spoje je rozšířena standardem 6LoWPAN definovaným RFC 6282, který poskytuje mechanismus pro transformaci paketů IPv6 pro přenos pomocí IEEE 802.15.4 [4]. Standardem je definována komprese 40bytových hlaviček IPv6 do 8bytových, které mohou být přenášeny fyzickou vrstvou. Dále popisuje fragmentaci a spojování datových paketů. IEEE 802.15.4 umožňuje přenos rámců velikosti 127 bytů, zatímco maximální přenosová jednotka dle IPv6 je 1280 bytů. Definuje také bezstavovou automatickou konfiguraci, která generuje IPv6 adresy pro jednotlivá zařízení. Důležitou součástí technologie jsou hraniční zařízení, která umožňují propojení WPAN sítě s jinými sítěmi používajícími protokol IPv6. Případně poskytují NAT[1] překlad adres pro komunikaci se sítěmi, které podporují pouze IPv4.

#### Síťová vrstva

Síťová vrstva implementuje směrování paketů v rámci protokolu IPv6. Jednotlivé typy adres budou blíže popsány v další sekci.

#### Transportní vrstva

Transportní vrstva sítě Thread implementuje protokol UDP. Jedná se o tzv. best effort protokol, který nezaručuje doručení zprávy, ale díky své jednoduchosti je vhodný pro aplikace s nízkými nároky na hardwarové zdroje a objem přenášených dat [61].

#### Aplikační vrstva

Síťový model Thread nespecifikuje aplikační vrstvu. Pro síťovou komunikaci mnoha aplikací se používá například protokol CoAP definovaný

[1] NAT – Network Address Translation

standardem RFC 7252 [57]. Ten je implementován ve většině SDK[2] pro vývoj aplikací s využitím sítě Thread. Jedná se o specializovaný protokol pro zařízení s omezenou funkcionalitou. Je navržen tak, aby jej bylo snadné používat spolu s protokolem HTTP. Umožňuje komunikaci na principu výzva odpověď. Analogicky k HTTP jsou standardem definovány metody GET, POST, PUT a DELETE. V odpovědi je pak obsažen stavový kód, který je číselně definován podobně jako u HTTP. CoAP dále přidává podporu multicastu. Na straně serveru jsou data organizována do tzv. *resources* s unikátními URI[3] identifikátory. Protokol umožňuje parametrizované URI a dodatečné hlavičky, které se podobají nejpoužívanějším hlavičkám protokolu HTTP. CoAP může fungovat nad TCP i UDP protokolem.

## 3.2 Směrování a role zařízení

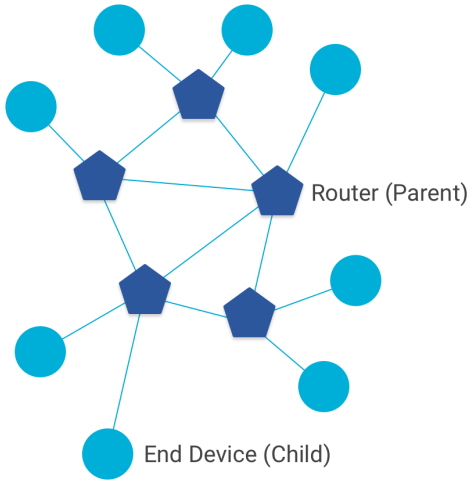
Z pohledu směrování jsou definovány 2 základní typy uzlů. Prvním typem je *Router*. Ten umožňuje směrování paketů jiným síťovým zařízení a poskytuje zabezpečené přidávání nových uzlů do sítě (tzv. *commissioning*). Router musí být neustále aktivní. Druhým typem je *End Device*. Přímo komunikuje pouze s Router uzlem a nemůže přeposílat žádné pakety dalším uzlům. End Device může mít radiový přijímač většinu času vypnutý pro větší úsporu energie. Router umožňuje agregaci nedoručených zpráv pro zařízení, která jsou ve spícím stavu. Z hlediska topologie grafu je mezi uzly typu Router a End Device vztah rodič – potomek.

Dle funkcionality rozlišujeme 2 základní typy zařízení – Full Thread Device (FTD) a Minimal Thread Device (MTD). FTD může mít následující funkce:

- Router
- Router Eligible End Device (REED) – Zařízení může být povýšeno na Router, pokud je jediným uzlem v dosahu End Device. Pokud nemá žádné potomky, může změnit svou roli zpět na End Device.

[2] Software Development Kit – sada vývojových nástrojů
[3] URI – Uniform Resource Identifier

- Full End Device (FED) – Může mít pouze roli End Device.



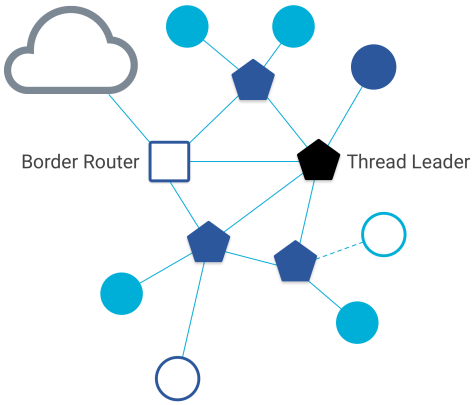

Obrázek 3.2: Základní role uzlů sítě (zdroj: [33])

FTD může v síti fungovat jako Router i End Device. MTD zařízení není zahrnuto do multicastové komunikace a posílá všechny zprávy svému rodiči v topologii. Dále se rozlišují 2 typy MTD zařízení:

- Minimal End Device (MED) – Je neustále na příjmu a nepotřebuje získávat zprávy od svého rodiče.
- Sleepy End Device (SED) – Většinu času je v režimu spánku. Občas se probudí a získá zprávy od svého rodiče.

Síť Thread definuje další 2 důležité typy zařízení – *Leader* a *Border Router*. Uzel typu Leader má stejnou funkci jako Router a navíc je zodpovědný za správu ostatních Router uzlů. Navíc uchovává a propaguje konfigurační data sítě. V síti je vždy jediný Leader. Kvůli odolnosti proti výpadkům probíhá jeho výběr samozvolením. Border Router umožňuje přesměrování komunikace ze sítě Thread do jiných sítí jako Wi-Fi, LTE 4G apod. Poskytuje také konektivitu zvenčí. Síť může obsahovat více uzlů typu Border Router.

V samostatné síti se vždy vyskytuje právě jeden Leader. Síť může obsahovat nejvýše 32 uzlů typu Router. Ke každému Router uzlu pak může být připojeno nejvýše 511 End Device. V závislosti na topologii si síť Thread snaží udržovat mezi 16 a 23 Router uzly.




Obrázek 3.3: Speciální role uzlů sítě (zdroj: [33])

## 3.3 Adresování

Síť Thread používá k adresování zařízení v síti IPv6 adresy. Tato část práce popisuje, jakým způsobem lze jednotlivá zařízení adresovat a jaké typy adres jsou v síti definovány. Pro komunikaci se zařízeními lze využít adresy následujících rozsahů:

- Link-Local – Slouží k adresaci rozhraní dosažitelných v rámci jednoho datového spoje. Adresy mají prefix `fe80::/16`.
- Mesh-Local – Identifikují rozhraní v rámci jedné sítě. Adresy mají prefix `fd00::/8`.
- Global — Umožňují adresaci z externí sítě.

### Routing Locator

Každý uzel sítě má přidělený 16bitový identifikátor Router ID a Child ID. Router zařízení si udržuje tabulku svých potomků v topologii sítě. Child ID uzlu typu Router má hodnotu 0. Kombinace těchto identifikátorů jednoznačně identifikuje každé zařízení v síti. Tento údaj se nazývá RLOC16.

RLOC16 je součástí identifikátoru rozhraní (IID), který odpovídá posledním 64 bitům IPv6 adresy. IID je tvaru `0000:00ff:fe00:RLOC16`. IID v kombinaci s Mesh-Local prefixem tvoří tzv. *RLOC* adresu (Routing Locator). RLOC závisí na topologii sítě, může se tedy během její existence měnit [17].

### Unicastové adresy

Další kategorií unicastových adres jsou EID identifikátory (Endpoint Identifier). Umožňují identifikovat rozhraní Thread v rámci sítě nezávisle na topologii. Existují tyto základní typy adres:

- Link-Local Address (LLA) – Identifikuje rozhraní dosažitelné v rámci datového spoje. IID část je založena na IEEE 802.15.4 Extended Address. Používá se k objevování sousedních uzlů a konfiguraci spojení. Neslouží ke směrování. Má prefix ve tvaru `fe80::/16`.

- Mesh-Local EID (ML-EID) – Jedná se o EID identifikátor nezávislý na topologii sítě. Slouží k dosažení rozhraní v rámci stejné Thread sítě. Je náhodně zvolen po přidání zařízení do sítě. Používá prefix ve tvaru `fd00::/8`. Tento typ adresy by měl být používán v aplikacích.

- Anycast Locator (ALOC) – Identifikuje rozhraní prostřednictvím průzkumu RLOC bez znalosti konkrétní RLOC adresy. IID je ve tvaru `0000:00ff:fe00:ALOC16`, kde ALOC16 identifikuje skupiny zařízení podle rolí (bude zmíněno později). Tyto adresy nejsou převážně aplikacemi používány.

- Global Unicast Address (GUA) – Identifikuje rozhraní v globálním rozsahu z vnější sítě. IID je založeno na IEEE 802.15.4 Extended Address. Jedná se o veřejnou IPv6 adresu s prefixem `2000::/3`.

**Multicastové adresy**

V síti jsou vyhrazeny následující adresy pro identifikaci různých skupin zařízení:

- `ff02::1` – Link-Local adresa sloužící k adresaci všech zařízení typu FTD a MED

- `ff02::2` – Link-Local adresa sloužící k adresaci všech zařízení typu FTD a Border Router

- `ff03::1` – Mesh-Local adresa sloužící k adresaci všech zařízení typu FTD a MED

- `ff03::2` – Mesh-Local adresa sloužící k adresaci všech zařízení typu FTD a Border Router

Tyto multicastové adresy nepodporují adresaci spících zařízení. Součástí Mesh-Local prefixu je ale multicastová adresa umožňující adresaci všech zařízení v síti.

#### Anycastové adresy

Poslední skupinou jsou anycastové adresy, které slouží k adresaci zařízení s určitou rolí. Jedná se o již zmíněné ALOC adresy. V Thread síti jsou vyhrazeny následující hodnoty ALOC16:

- `fc00` – Leader
- `fc01-fc0f` – DHCPv6 Agent
- `fc10-fc2f` – Služba
- `fc30-fc37` – Commissioner
- `fc40-fc4e` – Neighbor Discovery Agent

## 3.4 Formování sítě

Pro každou síť Thread se používají 3 různé unikátní identifikátory: 16bitový identifikátor Personal Area Network ID (*PAN ID*), 32bitový identifikátor Extended Personal Area Network ID (*XPAN ID*) a textový název sítě.

Když se zařízení pokouší vytvořit novou síť nebo připojit do již existující, provádí skenování okolních sítí. To probíhá tak, že zařízení zašle rámec *IEEE 802.15.4 Beacon Request* na všechny kanály. Každý uzel typu Router nebo REED v dosahu zašle odpověď s identifikátory sítě.

Pro konfiguraci datových spojů a šíření informací o síti Thread je použit protokol MLE[4] pro vytváření mesh sítí v rámci IEEE 802.15.4 [28]. MLE pokrývá objevování, zjišťování kvality a ustavení spojů se sousedními uzly a šíření informací jako *Leader RLOC*, *Partition ID*, *on-mesh prefix* a síťová data. Směrování je založeno na protokolu RIP[5] [52].

#### Vytváření sítě

Pokud se zařízení rozhodne vytvořit novou síť, dojde ke zvolení nejméně vytíženého kanálu a PAN ID, které není použito v žádné okolní síti.

[4] MLE – Mesh Link Establishment
[5] RIP – Routing Information Protocol

Poté se zařízení stane uzlem typu Router a následně se samo zvolí jako Leader. Poté zašle *MLE Advertisement* zprávu okolním zařízením s informací o svém stavu.

### Připojení k existující síti

Pokud se zařízení pokusí připojit existující k síti, musí mít korektně nakonfigurované základní údaje jako kanál, PAN ID, XPAN ID, název sítě a síťový klíč pro zabezpečení komunikace. Ty získá v rámci procesu nazvaného *commissioning*, který je popsán v následující sekci. Zařízení provede *MLE Attach* proces a přidá se do sítě jako potomek. Proces probíhá tak, že zařízení zašle multicastovou zprávu *Parent Request* všem sousedním zařízením typu Router nebo REED v cílové síti. Daná zařízení odpoví zprávou *Parent Response*. Zařízení si poté zvolí vhodný rodičovský uzel a pošle mu zprávu *Child ID Request*. Dotyčný odpoví zprávou *Child ID Response* a dojde k ustavení spojení.

### Zvolení Router uzlu

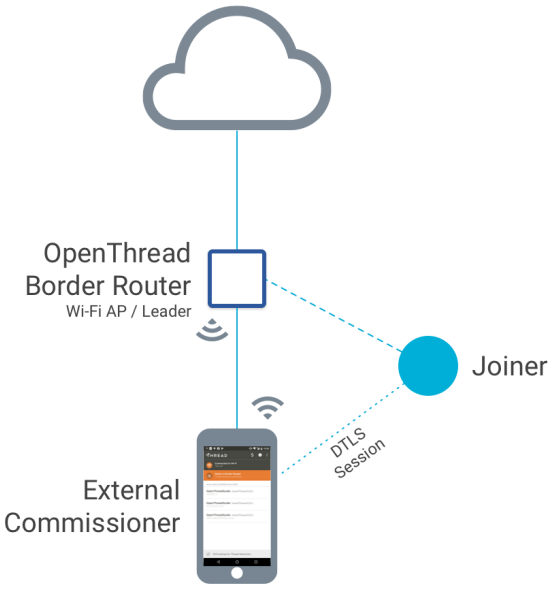
Router uzly musí tvořit v grafu topologie sítě souvislou dominující množinu. To znamená že tvoří souvislý podgraf a všechna zařízení End Device jsou připojena přímo k nějakému Router uzlu. Každé zařízení se do sítě přidá nejprve s rolí End Device a může být následně povýšeno na Router.

Síť Thread zvyšuje počet Router uzlů, pokud jejich počet klesne pod hranici 16, je třeba zvýšení počtu cest mezi uzly nebo je potřeba zvýšit počet End Device. Jejich stav se naopak snižuje, pokud je počet Router uzlů vyšší než 32 nebo je potřeba vytvořit Router v jiné části sítě.

Po přidání do sítě se End Device pokouší povýšit na Router. Nejprve se dotáže Leader uzlu na přidělení *Router ID*. Pokud Leader akceptuje požadavek a zašle identifikátor, dojde k povýšení role uzlu na Router. Pro ustavení spoje mezi dvěma Router uzly se používá proces *MLE Link Request*. Pokud je zařízení degradováno zpět na End Device, zruší se spoj a zařízení se opět přidá do sítě jako potomek pomocí procesu *MLE Attach*.

## 3.5 Přidání zařízení do sítě a zabezpečení

Proces přidání nového zařízení do sítě se nazývá *commissioning*. Zařízení které se pokouší přidat do sítě se nazývá *Joiner*. Dále v síti existuje minimálně jedno zařízení s funkcí *Commissioner*, které zajišťuje autentizaci. Roli Commissioner může mít zařízení komunikující v rámci sítě Thread, ale také externí zařízení komunikující prostřednictvím Border Router. Proces přidání je tak možný například pomocí chytrého telefonu nebo počítače.




Obrázek 3.4: Přidání do sítě pomocí externího zařízení (zdroj: [8])

Joiner má přidělené tajemství ve formě klíče, který je potřeba pro proces přidání. Commissioner tento klíč může získat ručním zadáním nebo prostřednictvím mobilní aplikace naskenováním QR kódu [59]. Joiner během procesu skenuje existující sítě. Po nalezení cílové sítě dojde ke konfiguraci kanálu, PAN ID, XPAN ID a názvu sítě. Pomocí sdíleného tajemství se vytvoří DTLS [5] spojení mezi zařízením Joiner

a Commissioner. Pomocí protokolu J-PAKE[6] [21] dojde k předání síťového klíče a přidání joiner zařízení do sítě. Samotná komunikace mezi uzly v síti je zabezpečená pomocí blokové šifry AES.

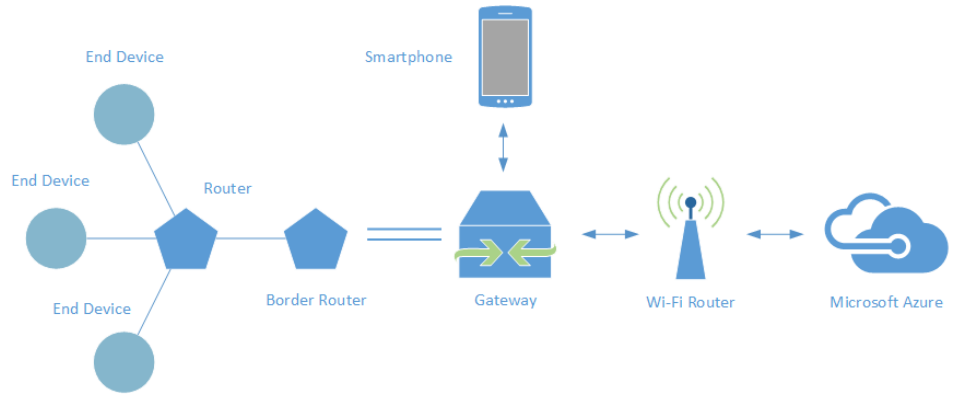
[6] Password-Authenticated Key Exchange by Juggling

# 4 Návrh systému

Kapitola popisuje návrh a implementaci řešení vytvořeného v rámci této práce. Cílem je vytvořit prototyp systému pro měření spotřeby a sběru dat o spotřebičích v domácnosti. Měřená data se předávají do existující platformy založené na službě Azure IoT Hub. Pro komunikaci mezi senzory je použita bezdrátová síť Thread.

## 4.1 Architektura systému

Obrázek 4.1 graficky znázorňuje zjednodušenou architekturu systému. Základem jsou moduly do elektrických zásuvek, které měří elektrické veličiny na připojeném spotřebiči. Tyto moduly komunikují prostřednictvím sítě Thread. Do sítě je možné dále připojit i další senzory a aktuátory libovolné funkcionality. Vzhledem k tomu, že zásuvky budou přímo napájeny z elektrické sítě, mohou mít v síti roli Router a budou tvořit páteřní topologii pro připojení zařízení typu Sleepy End Device.




Obrázek 4.1: Konceptuální architektura systému (zdroj: autor)

Síť obsahuje jedno zařízení typu Border Router, které tvoří výstupní bránu do Internetu. V kontextu této práce se jedná o zařízení nazvané *gateway*. Gateway jako celek slouží k propojení sítě Thread s domácí

LAN sítí a zároveň zajišťuje konektivitu se službou Azure IoT Hub. Má výpočetní kapacitu dostačující pro zpracování dat a automatizované ovládání připojených zařízení. Prostřednictvím Wi-Fi přístupového je možné připojení pomocí mobilní aplikace, která zajistí pohodlné přidávání jednotlivých zásuvek a konfiguraci sítě. Zařízení gateway může komunikovat s lokální sítí pomocí Wi-Fi, případně Ethernetu.

## 4.2 Azure IoT Hub

Existující platforma Hexade, kterou nyní využívají chytré Wi-Fi zásuvky, je založena na službě Azure IoT Hub [44]. Jedná se o škálovatelnou službu poskytovanou v rámci platformy Microsoft Azure. Služba zajišťuje správu identit jednotlivých zařízení, jejich konfiguračních dat a tajemství používaných pro autentizaci a zabezpečenou komunikaci. Dále poskytuje obousměrnou komunikaci mezi zařízeními a cílovou aplikací. Konektivita je zajištěna pomocí aplikačního protokolu MQTT nebo AMQP. Službu je možné rozšiřovat o další protokoly. Azure IoT Hub obsahuje také podporu aplikací hraničních zařízení nazvaných Azure IoT Edge. Tato funkcionalita je použita ve výsledném řešení a je podrobněji popsána v následujících kapitolách.

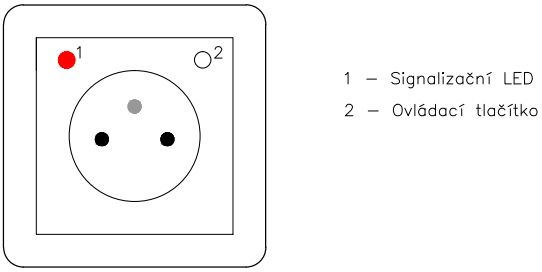
Součástí platformy Hexade je mimo jiné služba pro analýzu proudu dat, služba pro dávkové zpracování velkých dat, databáze a webové rozhraní pro klientské aplikace.

## 4.3 Zařízení pro měření spotřeby

Tato část textu obsahuje návrh modulu zásuvky. Jedná se o miniaturní zařízení s nízkou spotřebou, které umožňuje měření elektrických veličin a sledování chování spotřebiče. Je schopné komunikovat pomocí bezdrátové sítě Thread. Konstrukce je navržena modulárně takovým způsobem, aby byly části použitelné pro konstrukci dalších zařízení. Jedním z cílů je přenositelnost zdrojových kódů na jiné platformy, aby nebyl vývoj omezen pouze na jednoho výrobce čipů.

Zařízení bude možné zabudovat do standardizovaných zásuvkových rámečků v elektroinstalaci v interiéru budovy. Zásuvky by tak neměly vyčnívat a výrazně se lišit od běžných elektrických zásuvek. Bude je také možné snadno vyměnit. Součástí konstrukce zásuvky

bude tlačítko umožňující základní ovládání a signalizační dioda pro signalizovaní stavu zařízení. Vizualizace s rozmístěním ovládacích komponent zásuvky je na obrázku 4.2. Moduly komunikují pomocí sítě Thread. Předpokládá se rovnoměrné rozmístění zásuvek v místnostech takovým způsobem, aby bylo možné zformovat souvislou síť. Návrh zařízení se skládá ze dvou propojených částí – komunikační a měřicí.




Obrázek 4.2: Vizualizace elektrické zásuvky (zdroj: autor)

### 4.3.1 Komunikační modul

Na trhu je momentálně dostupných několik mikročipů, které podporují protokol Thread. Často se jedná o moduly běžně používané pro sítě nad standardem IEEE 802.15.4, jako je Zigbee [46]. Pro zásuvkový modul vyvinutý v rámci této práce byl vybrán komunikační modul R41Z od společnosti Rigado, který je založen na integrovaném obvodu KW41Z vyráběném společností NXP [53]. Výhodou je schopnost komunikace pomocí Thread, Zigbee i Bluetooth 4.2.

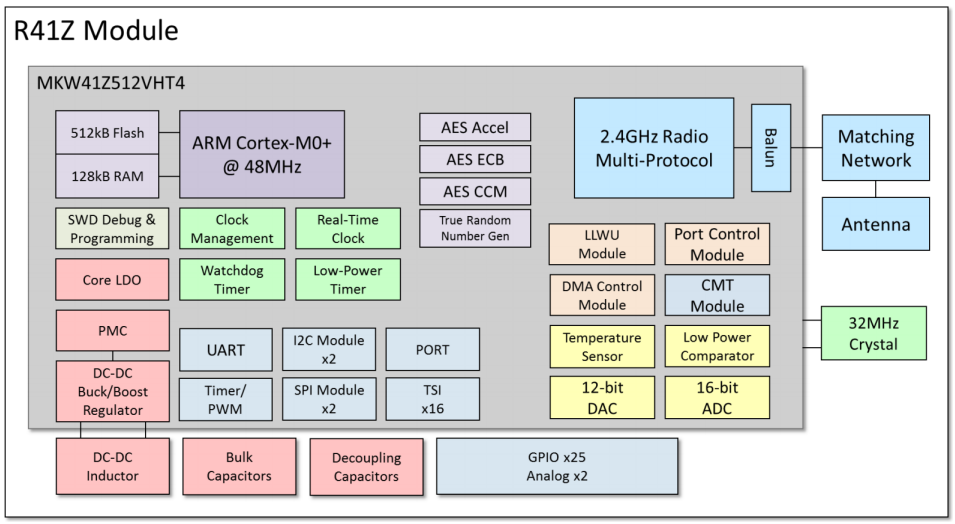
#### Modul R41Z

Modul je založen na SoC[1] KW41Z řady Kinetis společnosti NXP. Diagram 4.3 znázorňuje blokové zapojení modulu R41Z a čipu KW41Z. Jedná se o jednočipový počítač s integrovaným rádiovým modulem,

[1] System on a Chip – počítač integrovaný v jediném integrovaném obvodu

který podporuje komunikaci v pásmu ISM 2,4 GHz. Obvod obsahuje jádro ARM Cortex-M0+, které lze taktovat až na frekvenci 48 MHz. Je vybaven pamětí flash velikosti 512 KB a operační pamětí typu SRAM velikosti 128 KB. Dále obsahuje běžné periferie jako GPIO, LPUART[2], SPI, I2C, A/D převodník, modul reálného času, několik typů časovačů a další. Obvod se vyznačuje nízkou spotřebou a různými režimy pro maximální úsporu energie při běžných aplikacích. Z pohledu zabezpečení komunikace je výhodou také generátor náhodných čísel a hardwarový akcelerátor pro šifru AES-128. Je také vybaven rozhraním SWD pro nahrávání a ladění firmware [30].

Samotný modul R41Z obsahuje základní zapojení obvodu KW41Z se stíněním, anténou na desce plošného spoje a krystalem s frekvencí 32 MHz. Rozměry desky modulu jsou 10,6 x 12,2 mm. Piny jsou vyvedené na kontakty ze spodní strany desky, osazení je tedy možné pouze v peci. Moduly jsou distribuovány s přidělenou unikátní MAC adresou.




Obrázek 4.3: Blokový diagram modulu R41Z (zdroj: [54])

[2] LPUART – Low Power Universal Asynchronous Receiver and Transmitter

### 4.3.2 Obvod měření elektrických veličin

Měřicí obvod slouží k měření elektrických charakteristik silnoproudé části síťové zásuvky s připojeným spotřebičem. Základními sledovanými veličinami jsou napětí v elektrické síti, elektrický proud, příkon připojeného spotřebiče a celková spotřeba energie. Uvažujeme elektrickou síť 230 V, 50 Hz. Zapojení by mělo bezpečně vydržet velikost protékajícího proudu až 16 A.

#### Obvod STPM32

Základem měřicího zapojení je integrovaný obvod STPM32 vyráběný firmou STMicroelectronics [1]. Jedná se o obvod typu ASSP[3] určený k přesnému měření pomocí proudového transformátoru, rogowského cívky nebo bočníkového odporu. Obvod umožňuje měření hodnot RMS[4] napětí, RMS proudu, činného a jalového výkonu, fázového posuvu a příslušných energií. Udávaná chyba měření činného výkonu je menší než 0,1%. Dále umožňuje detekovat přepětí a podpětí v síti a mnoho dalších událostí. Obsahuje také 2 programovatelné výstupy pro LED diody umožňující optický přenos měřených dat. Obvod je napájen napětím 3,3 V a udávaný odběr proudu činí 4,3 mA.

Obvod STPM32 obsahuje jeden kanál pro měření napětí a jeden pro měření proudu. Obsahuje 24bitové sigma-delta A/D převodníky, programovatelné zesilovače a filtry horní propusti. Takt obvodu lze řídit interním krystalem s frekvencí 16 MHz, přesným vnějším krystalem nebo oscilátorem. Vyrábí se v miniaturním pouzdře QFN24L. Komunikace s obvodem je možná pomocí standardního sériového rozhraní UART[5] nebo SPI[6].

Konfigurace, ovládání a čtení měření hodnot je prováděno pomocí zápisu a čtení vnitřních registrů obvodu. Jsou použité 32bitové registry, které jsou podrobně popsány v dokumentaci [1]. Obecně se jedná o konfigurační přepisovatelné registry, stavové registry pro indikaci událostí a čítače měřených hodnot.

---

[3] ASSP – Application-specific Standard Part

[4] Zkratka RMS (Root Mean Square) označuje střední kvadratickou hodnotu – v elektrotechnice tzv. efektivní hodnotu

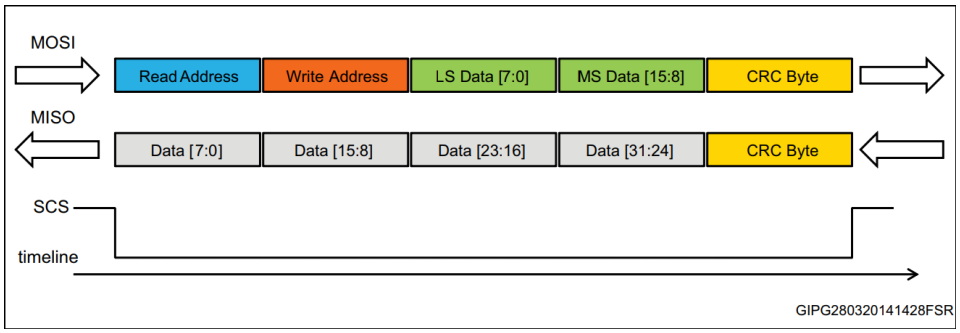
[5] UART – Universal Asynchronous Receiver and Transmitter

[6] SPI – Serial Peripheral Interface

### Popis komunikace

K základnímu ovládání obvodu slouží 3 piny – *EN* (aktivace čipu), *SYN* (synchronizační pin) a *SCS* (chip-select). Logickou hodnotou na pinu SCS při startu čipu lze mimo jiné zvolit mezi režimy komunikace UART (logická 1) nebo SPI (logická 0).

Komunikační rozhraní podporuje operace zápisu a čtení registrů. Komunikace probíhá v plně duplexním režimu a je rozdělena do datových rámců po 5 bytech. Master zařízení posílá rámec s žádostí čtení a zápisu registrů, kde první byte obsahuje adresu pro čtení, druhý adresu pro zápis, další 2 byty obsahují data pro zápis a poslední byte obsahuje CRC[7] pro verifikaci integrity posílaných dat. Rámec s odpovědí obsahuje 4 byty s daty požadovaného registru a v posledním bytu je opět CRC kód. Požadavek zápisu dat je proveden okamžitě po odeslání rámce s žádostí. Při čtení dat je nezbytné přečíst minimálně 2 příchozí rámce. Kontrola CRC je volitelná a je možné ji aktivovat hodnotou *CRCEnable* v registru *US_REG1*. Polynom pro výpočet CRC je možné explicitně nastavit hodnotou *CRCPolynomial*.



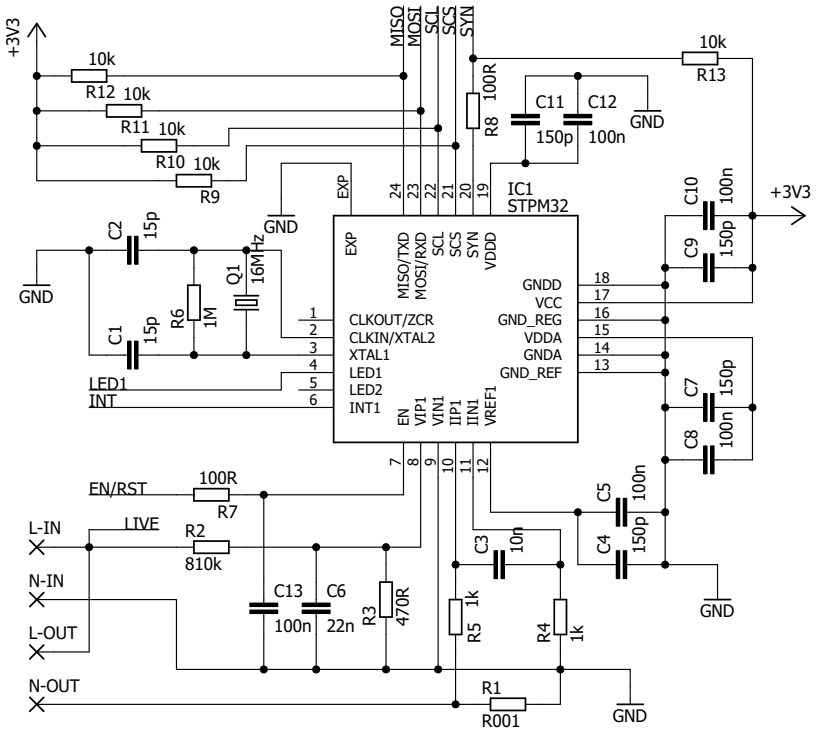

Obrázek 4.4: Komunikační rámec STPM32 (zdroj: [1])

Komunikace rozhraní UART probíhá prostřednictvím 2 datových vodičů RXD a TXD. Rozhraní komunikuje v plně duplexním asynchronním režimu s konfigurací – 1 start bit, 8 datových bitů a 1 stop bit. Navíc při použití UART musí být při přenosu dat pin SCS na logické hodnotě 1.

[7] Cyclic Redundancy Check – cyklický redundantní součet

Ke komunikace prostřednictvím rozhraní SPI jsou použity 3 obvyklé vodiče MISO, MOSI a SCL. Jako *chip-select* je použit pin SCS, který musí být pro aktivaci komunikace na logické hodnotě 0. Rozhraní SPI je nakonfigurováno v módu 3, tedy: *cpol=1, cpha=1*.

**Popis zapojení měřicího obvodu**




Obrázek 4.5: Schéma zapojení měřicí části (zdroj: autor)

Na obrázku 4.5 je schéma zapojení měřicí části zařízení. Kompletní schéma je v příloze B. Základem zapojení je měřicí čip IC1, který je napájen z napájecího zdroje napětím velikosti 3,3 V. Zdrojem hodinového signálu integrovaného obvodu je krystal Q1. Na napěťový kanál na pinech VIP1 a VIN1 je přivedeno síťové napětí přes dělič tvořený rezistory R2 a R3, který sníží velikost napětí na měřitelnou

hodnotu. K měření proudu slouží bočníkový rezistor R1 zapojený za spotřebič. Velikost odporu je 1 mΩ a je připojen k proudovému kanálu tvořenému piny IIP1 a IIN1. Vývody EN/RST, INT, MISO, MOSI, SCL, SCS a SYN slouží ke komunikaci a ovládání měřicího čipu.

### Měření proudu pomocí bočníkového odporu

Měření v navržené aplikaci je realizováno pomocí bočníkového odporu. Princip vychází z ohmova zákona – při průchodu elektrického proudu odporem na něm lze naměřit určité napětí dané vztahem (4.1). Hodnota napětí je převedena pomocí A/D převodníku do digitální interpretace a je použita k výpočtům dalších veličin.

$$U = I \cdot R \tag{4.1}$$

Pro běžné aplikace je vhodné použít SMD rezistory s nízkou odchylkou pod 1%, maximálním tepelným výkonem alespoň 2W a velikostí odporu 1-3 mΩ. Větší hodnoty odporu se mohou při procházejícím proudu v řádu jednotek ampér projevit negativně na zahřívání součástky.

### Kalibrace

V následujících odstavcích je zmíněno několik vztahů, které obsahují proměnné závislé na hodnotách součástek použitých v zapojení. Tabulka 4.1 obsahuje seznam proměnných s jejich významem a hodnotami pro dosazení.

Vzhledem k odchylkám hodnot použitých součástek, odchylce napěťové reference pro A/D převodník a variabilní kvalitě osazení DPS[8] je nezbytné provést kalibraci obvodu pro dosažení co nejpřesnějšího měření. Kalibrace je prováděna vůči referenční zátěži (tzv. *etalonu*) nastavením kalibrační hodnoty odpovídajícího registru čipu STPM32. Ke kalibraci slouží hodnota *CHV1* registru *DSP_CR5* pro napěťový kanál a *CHC1* registru *DSP_CR6* pro proudový kanál. Dále je možné kalibrovat fázový posun pro proudový a napěťový kanál hodnotami *PHV1* a *PHC1* registru *DSP_CR4*.

[8] Deska plošného spoje

| Proměnná | Dosazená hodnota | Popis |
|---|---|---|
| $V_{ref}$ | 1,18 V | Referenční napětí (výchozí hodnota platí při použití interní reference — registr *DSP_CR*, hodnota *ENVREF1*) |
| $R_1$ | 810 000 Ω | Velikost odporu prvního rezistoru (*R2*) děliče napětí pro napěťový kanál |
| $R_2$ | 470 Ω | Velikost odporu druhého rezistoru (*R3*) děliče napětí pro napěťový kanál |
| $A_V$ | 2 | Velikost násobku zesílení napěťového kanálu |
| $A_I$ | 16 | Velikost násobku zesílení proudového kanálu kanálu (registr *DFE_CR*, hodnota *GAIN*) |
| $k_{si}$ | 0,001 Ω | Senzitivita — při použití bočníkového rezistoru je rovna velikosti jeho odporu |
| $cal_V$ | 0,875 (výchozí) | Hodnota korekčního faktoru kalibrace napěťového kanálu (registr *DSP_CR5*, hodnota *CHV1*) |
| $cal_I$ | 0,875 (výchozí) | Hodnota korekčního faktoru kalibrace proudového kanálu (registr *DSP_CR5*, hodnota *CHI1*) |
| *DClk* | 16 000 000 Hz | Frekvence krystalu *Q1* |

Tabulka 4.1: Proměnné pro výpočet vztahů obvodu STPM32

Střední hodnota kalibrace odpovídá bitové hodnotě $0x800$. Rozsah kalibrace je pak $\pm$ 12,5%. Vstupem výpočtu je nominální hodnota registru pro proud $I_N$ a napětí $V_N$. Cílové hodnoty registrů napětí $X_V$ a $X_I$ jsou dané vztahy (4.2) a (4.3).

$$X_V = \frac{V_N \cdot A_V \cdot cal_V \cdot 2^{15}}{V_{ref} \cdot (1 + \frac{R_1}{R_2})} \tag{4.2}$$

$$X_I = \frac{I_N \cdot A_I \cdot cal_I \cdot k_s i \cdot 2^{17}}{V_{ref}} \tag{4.3}$$

Po aplikaci daných nominálních hodnot je potřeba provést několik nezávislých měření RMS napětí a proudu a spočítat průměr přečtených hodnot. $V_{AV}$ pak nazveme průměrnou hodnotu napětí a $I_{AV}$ průměrnou hodnotu proudu. Z těchto spočtených proměnných pak lze spočítat hodnoty kalibračních registrů dle vztahů (4.4) a (4.5).

$$CHV = 14336 \cdot \frac{X_V}{V_{AV}} \cdot 12288 \tag{4.4}$$

$$CHC = 14336 \cdot \frac{X_I}{I_{AV}} \cdot 12288 \tag{4.5}$$

Pomocí následujících rovnic se pak spočítají odpovídající korekční faktory pro další výpočty:

$$K_V = 0,125 \cdot \frac{CHV}{2048} + 0,75 \tag{4.6}$$

$$K_I = 0,125 \cdot \frac{CHI}{2048} + 0,75 \tag{4.7}$$

Při měření bočníkovým rezistorem bez použití proudového a napěťového transformátoru nemusíme uvažovat zásadní odchylky fázového posuvu. Hodnoty *PHV1* a *PHC1* tak nastavíme na výchozí hodnotu $0x0$.

### Čítače a přepočet na skutečné veličiny

Měřené hodnoty jsou zapisovány do interních registrů, které nejsou přístupné pro čtení komunikačním rozhraním. Hodnoty se neustále mění a proto je potřeba při čtení provézt tzv. operaci *latch*, která okamžité hodnoty zkopíruje do registrů pro čtení. Latch lze vyvolat hardwarovou cestou a to jedním pulzem na pinu SYN čipu. Další možností je zapsat hodnotu $0x1$ do konfiguračního bitu *S/W latch1* registru *DSP_CR3*.

Aplikace vytvořené v rámci této práce používá následující veličiny a jim odpovídající hodnoty registrů:

- RMS napětí – *V1 RMS Data*, registr *DSP_REG14*
- RMS proud – *C1 RMS Data*, registr *DSP_REG14*
- Činný výkon – *PH1 Active Power*, registr *PH1_REG5*
- Celková spotřeba – *PH1 Active Energy*, registr *PH1_REG1*
- Fázový posun – *C1_PHA*, registr *DSP_REG17*

Hodnoty jsou uloženy v registrech v binární podobě, kterou je potřeba převést na reálné veličiny vynásobením reálnou hodnotou jednoho bitu (značena zkratkou LSB) dle následujících vztahů:

$$LSB_{VRMS} = \frac{V_{ref} \cdot (1 + \frac{R_1}{R_2})}{cal_V \cdot A_V \cdot 2^{15}}[V] \tag{4.8}$$

$$LSB_{IRMS} = \frac{V_{ref}}{cal_I \cdot A_I \cdot 2^{17} \cdot k_{si}}[A] \tag{4.9}$$

$$LSB_P = \frac{V_{ref}^2 \cdot (1 + \frac{R_1}{R_2})}{cal_V \cdot cal_I \cdot A_V \cdot A_I \cdot k_{si} \cdot 2^{28}}[W] \tag{4.10}$$

$$LSB_E = \frac{V_{ref}^2 \cdot (1 + \frac{R_1}{R_2})}{3600 \cdot DClk \cdot cal_V \cdot cal_I \cdot A_V \cdot A_I \cdot k_{si} \cdot 2^{17}}[Wh] \tag{4.11}$$

### 4.3.3 Popis zapojení zařízení

Kompletní schéma zapojení je v příloze B. Zapojení měřicí části bylo již popsáno v předchozí části. Napájení je realizováno jednoduchým obvodem s kondenzátorem C15 a zenerovou diodou D2 stabilizující vstupní napětí na hodnotu 3,3 V. Obvod je proti zkratu chráněn pojistným odporem R14. Komunikační modul U1 je opatřen krystalem Q2, který slouží jako zdroj přesného hodinového signálu pro modul reálného času. Tlačítko S1 je hlavním ovládacím prvkem zapojení. LED

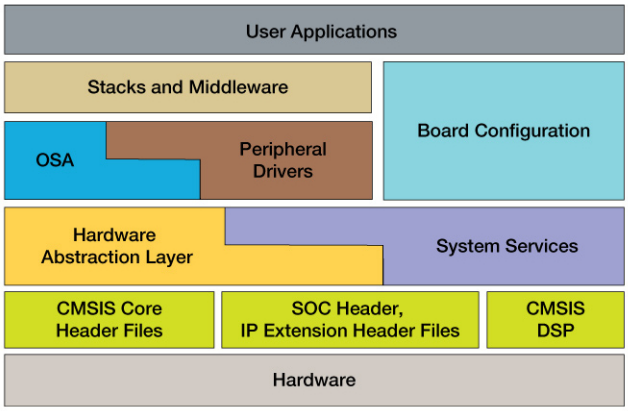
dioda LED1 slouží k signalizaci stavu zařízení. Obvod U1 komunikuje s měřicím čipem prostřednictvím rozhraní SPI na pinech PTA16, PTA17 a PTA18 (alternativní funkce ALT2 [30]). Na programovací a testovací konektor JP1 jsou vyvedena rozhraní UART a SWD.

### 4.3.4 Popis firmware

Firmware zásuvky využívá několik knihoven třetích stran. Všechny části jsou napsané v jazyce C nebo C++. Veškeré zdrojové kódy jsou dostupné jako *open-source*. Kód je naspán takovým způsobem, aby se dalo celé řešení snadno portovat na jiné platformy.

#### Kinetis SDK

Základem funkcí závislých na platformě KW41Z je sada nástrojů Kinetis SDK v2 . Jedná se o implementaci abstraktního rozhraní a ovladačů pro ovládání čipu a jeho periferií mikrokontrolérů řady Kinetis [22]. Součástí SDK jsou také názorné příklady práce s konkrétními funkcemi čipu. Na webu výrobce [22] se dále nachází nástroj pro sestavení SDK konkrétního čipu, nástroj na konfiguraci pinů a generátor projektů.




Obrázek 4.6: Blokový diagram architektury Kinetis SDK (zdroj: [22])

### OpenThread

OpenThread je knihovna implementující protokol Thread. Je vyvinutá společností Nest [41]. Původně byla určená pro zjednodušení vývoje software produktů Nest. Vyznačuje se dobrou přenositelností a obsahuje porty na několik platforem certifikovaných pro Thread. Knihovna je implementována v jazyce C++, rozhraní je tvořeno hlavičkovými soubory v jazyce C. Implementovány jsou všechny vrstvy síťového modelu a všechny role uzlů, včetně Border Router. Knihovna dále obsahuje implementaci některých aplikací jako UDP socket, CoAP klient a server, DHCPv6 klient a server a DNSv6 klient.

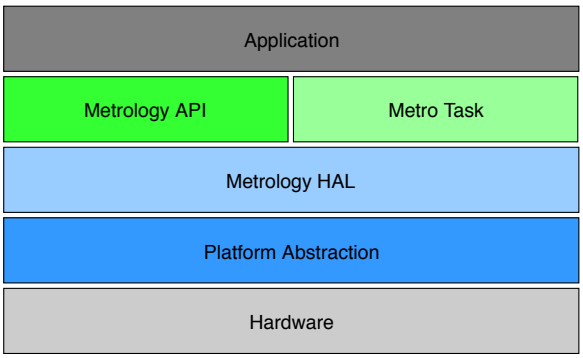
Projekt mimo jiné obsahuje implementaci aplikace ovládanou pomocí příkazového řádku, nebo aplikaci, která umožňuje používat čip jako síťový koprocesor (NCP[9]) jiných zařízení. Součástí je také aplikace pro Border Router běžící na linuxových systémech. OpenThread využívá pro kryptografické operace a implementaci DTLS knihovnu MbedTLS [55]. Port pro platformu KW41Z používá nástroje Kinetis SDK.

### Knihovna ovladače pro měřicí čip

Součástí aplikace je knihovna pro ovládání měřicího čipu STPM32. Je naprogramována s využitím ukázkové implementace firmware pro čipy řady STPM3x vydané společností STMicroelectronics [56]. Program byl upraven takovým způsobem, aby se dal snadno přenášet na jiné platformy. Součástí knihovny je její port na platformu KW41Z.

Knihovna implementuje komunikaci pomocí rozhraní UART i SPI, zápis a čtení registrů čipu a základní funkce nezbytné pro plnohodnotné používání měřicího obvodu. Součástí je také implementace funkcionality měření a přepočtu dat na reálné veličiny. Obrázek 4.7 graficky znázorňuje jednotlivé vrstvy aplikace pro měření elektrických veličin pomocí STPM32. Knihovna umožňuje souběžné použití až 4 samostatných kanálů, kde kanály mohou být v rámci jednoho nebo více fyzických čipů. Aplikace pro zařízení vyvinuté v rámci této práce používá pouze jeden kanál.

[9] Network Control Processor




Obrázek 4.7: Blokový diagram aplikace pro STPM32 (zdroj: autor)

Vrstva *Metro Task* obsahuje jednoduchou implementaci měření základních veličin. *Metrology API* poskytuje základní funkce pro práci s obvody STPM3x. Pro nízkoúrovňová volání používá HAL[10] vrstvu. *Metrology HAL* vrstva obsahuje implementaci komunikace a logiky pro práci s měřicím obvodem. *Platform Abstraction* vrstva izoluje platformě závislá volání vyžadovaná HAL vrstvou. Tato část musí být implementována pro konkrétní použitou platformu.

| Vrstva | Hlavičkový soubor |
|---|---|
| Metro Task | `metroTask.h` |
| Metrology API | `metrology.h` |
| Metrology HAL | `metrology_hal.h` |
| Platform Abstraction | `metrology_platform.h` |

Tabulka 4.2: Hlavičkové soubory knihovny pro STPM32

Komunikační rozhraní je zvoleno během kompilace symbolem pro preprocesor (flag). Nastavením symbolu `UART_XFER_STPM3X` je zvoleno rozhraní UART, symbolem `SPI_XFER_STPM3X` pak rozhraní SPI. Komunikační rozhraní jsou vzájemně výlučná, je tedy nutné zvolit právě jedno. Ukázka použití je znázorněna v příloze D.

[10] HAL – Hardware Abstraction Layer

### Faktory pro výpočet veličin

Pro korektní přepočet na reálné jednotky vyžaduje aplikace konfiguraci hodnot faktorů. Vzorce pro výpočet jednotlivých faktorů vychází ze vztahů pro přepočet bitových hodnot (LSB). Knihovna používá celočíselné hodnoty kvůli lepší kompatibilitě s různými platformami. Pro minimální ztrátu přesnosti jsou použity tisíciny jednotek – tedy miliampér, milivolt atd. Posuvy registrů jsou v knihovně prováděny implicitně a nezohledňují se při výpočtu faktorů. Vztah (4.12) popisuje výpočet faktoru pro napětí, (4.13) pro proud, (4.14) pro činný výkon a (4.15) pro spotřebu energie. Výsledné hodnoty je třeba zaokrouhlit na celá čísla.

$$f_V = 10^2 \cdot \frac{V_{ref} \cdot (1 + \frac{R_1}{R_2})}{cal_V \cdot A_V}[mV] \tag{4.12}$$

$$f_I = 10^2 \cdot \frac{V_{ref}}{cal_I \cdot A_I \cdot k_{si}}[mA] \tag{4.13}$$

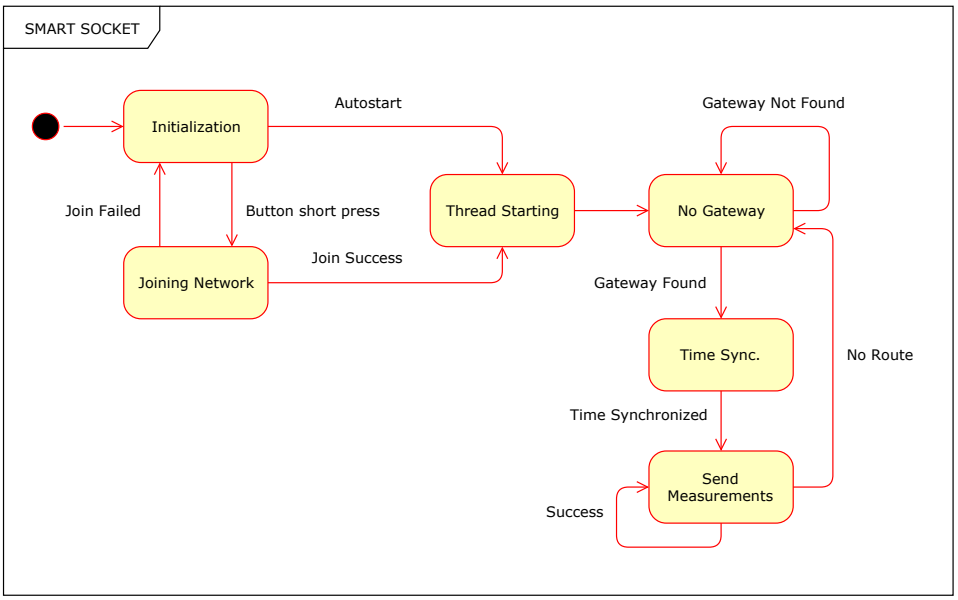
$$f_P = 10^2 \cdot \frac{V_{ref}^2 \cdot (1 + \frac{R_1}{R_2})}{cal_V \cdot cal_I \cdot A_V \cdot A_I \cdot k_{si}}[mW] \tag{4.14}$$

$$f_E = \frac{2^{26} \cdot f_P}{3600 \cdot DClk}[mW] \tag{4.15}$$

Při sériové výrobě bude proces kalibrace čipu, výpočtu faktorů a jejich nahrání proveden automatizovaně při oživování desky plošného spoje testerem vyrobeným na míru. V programu se nastavení hodnot provede pomocí funkce `Metro_Set_Hardware_Factors`. Ukázka použití je v příloze D

### Popis fungování

Součástí zařízení je jediné ovládací tlačítko a signalizační dioda. Krátkým stiskem tlačítka po dobu 1-3 vteřin se vyvolá proces přidání zásuvky do sítě. Dlouhým stiskem delším než 3 vteřiny je vyvolán reset do továrního nastavení, který vymaže všechny informace o konfiguraci sítě a uvede zařízení do iniciálního stavu.




Obrázek 4.8: Stavový diagram zařízení (zdroj: autor)

Běh programu je znázorněn stavovým diagramem 4.8. Po prvním zapojení je zařízení v iniciálním stavu a vyčkává v nečinnosti. Stisknutím tlačítka je vyvolán proces přidání do sítě Thread, který je podrobněji popsán v části 4.4. Pokud se proces nezdaří, zařízení se opět vrátí do iniciálního stavu. V opačném případě se spustí síť Thread. Zařízení je následně přidáno do existující sítě a nastaví si roli dle dané topologie.

Jakmile je zařízení součástí sítě, pošle pomocí protokolu CoAP požadavek na resource `.well-known/core` metodou GET na multicastovou adresu `ff03::2`. V případě, že nedostane odpověď od žádného požadovaného serveru, bude požadavek zasílat znovu se sekundovým intervalem. Jakmile dostane odpověď, uloží si adresu cílového serveru. Následně se provede synchronizace času požadavkem GET na resopurce `time`. Jakmile je celý proces úspěšně proveden, zásuvka v pravidelných intervalech zasílá měřená data na CoAP server. Veškeré zprávy obsahují data ve formátu JSON. Jejich tvar je blíže popsán v části 4.4. Po změně konfigurace sítě je možné provést nový proces při-

dání do sítě pomocí krátkého stisku tlačítka. Signalizační LED diodou jsou indikovány následující stavy:

- Svítí – Zařízení je v iniciální fázi a čeká na připojení do sítě
- Bliká rychle – Probíhá proces přidání do sítě
- Bliká pomalu – Probíhá hledání brány a zahájení komunikace
- Nesvítí – Zásuvka je korektně připojená a komunikuje

### Sestavení programu

Zdrojové kódy obsahují všechny nezbytné součásti programu. Doporučuje se však stáhnout nejnovější opravenou verzi knihovny OpenThread ze stránky projektu na serveru GitHub [35]. Pro sestavení firmware je nezbytné GNU prostředí s nainstalovanými nástroji GNU toolchain pro Arm Cortex-M [12]. Projekt obsahuje instalační soubory pro běžné operační systémy. Firmware se pak sestaví příkazem `make` v kořenovém adresáři projektu. Jeho nahrání do paměti flash je možné provést pomocí rozhraní SWD použitím vhodného programátoru (např. J-Link [19]).

### Portování na jiné platformy

Firmware měřicího zařízení je přizpůsoben pro případné přenesení na jiné platformy. Je možné použít i jinou implementaci knihovny pro síť Thread a CoAP server, je však doporučeno využít platformy podporované projektem OpenThread [46]. Pro správné fungování aplikace je nezbytné implementovat následující hlavičkové soubory. Podrobný popis jednotlivých funkcí je součástí dokumentace zdrojového kódu.

- `metrology_platform.h` – Předepsané funkce jsou využívány knihovnou pro měřicí čip STPM32. V rámci souboru jsou definovány funkce pro přístup ke komunikačnímu rozhraní UART a SPI a jejich inicializace. Konfigurace jednotlivých měřicích kanálů se uchovává v uživatelsky definovaných strukturách `pin_handle_data` (konfigurace ovládacích pinů), `spi_handle_data` (konfigurace SPI rozhraní) a `uart_handle_data` (konfigurace UART rozhraní).

- `led_abstraction.h` – Obsahuje funkce pro ovládání LED kontrolky, která podporuje 4 různé stavy – svítí, nesvítí, bliká pomalu a bliká rychle.

- `button_abstraction.h` – Předepisuje funkce pro hlavní ovládací tlačítko. Je potřeba implementovat přerušení na krátký a dlouhý stisk a vyvolání odpovídající události.

- `rtc_abstraction.h` – Obsahuje funkce pro ovládání modulu reálného času. Pokud není součástí zvoleného mikrokontroléru, musí být použit externí čip jako například PCF8563 [45]. Modul reálného času musí podporovat přerušení časovače, která vyvolají odpovídající událost.

- `thread_abstraction.h` – Poskytuje rozhraní pro funkcionalitu související s používáním sítě Thread. V případě použití implementace OpenThread je možné ponechat výchozí implementaci funkcí.

- `coap_abstraction.h` – Obsahuje funkce pro komunikaci se zařízením gateway prostřednictvím protokolu CoAP. Při použití knihovny OpenThread je opět možné ponechat výchozí implementaci, jinak je potřeba dodržet aplikační logiku v referenční implementaci.

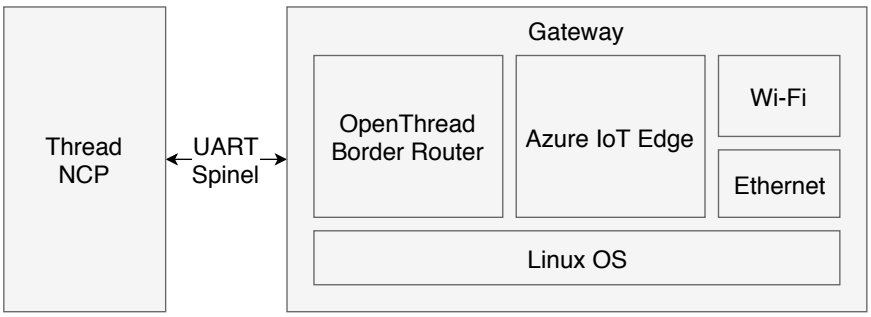
- `smart_socket_config.h` – Obsahuje konfiguraci chování zásuvky.

Podrobné informace jak portovat knihovnu OpenThread na nové platformy jsou na webové stránce projektu [47].

## 4.4 Zařízení gateway

Gateway plní funkci přístupového bodu do vnější sítě a Internetu. Zároveň může sloužit jako centrum domácí automatizace a přebírat některé funkce cloudové platformy (například zpracování a analýzu sbíraných dat). Jedná se o samostatné síťové zařízení, které musí být součástí sítě Thread, aby mohla být sbíraná data předávána do Internetu. Jedna síť může obsahovat více než jednu bránu. Zařízení je

napájeno z elektrické sítě a je umístěno v domácnosti obdobně jako Wi-Fi router.




Obrázek 4.9: Blokový diagram výstupní brány (zdroj: autor)

Diagram 4.9 znázorňuje kompozici výstupní brány. Skládá se ze dvou samostatných hardwarových částí a to ze samotného minipočítače a síťového koprocesoru (Thread NCP). Zařízení gateway musí obsahovat Wi-Fi komunikační rozhraní (volitelně Ethernet). Software je podporovaný pouze na linuxových operačních systémech.

Softwarové řešení je založeno na dvou samostatných projektech. Pro komunikaci se sítí Thread je použit software OpenThread Border Router a firmware síťového koprocesoru OpenThread NCP [37]. Pro spojení se službou Azure IoT Hub je použit projekt Azure IoT Edge v2 [62].

### 4.4.1 Podporované platformy

Jak bylo zmíněno, řešení je podporováno pouze na platformách s linuxovým operačním systémem a podporou GNU. Kvůli minimalizaci rozměrů a spotřeby se doporučuje využít počítače založené na procesorech s ARM architekturou. Minimální požadavky na zdroje jsou: 1 GHz procesor s jedním jádrem, 256 MB RAM, 4 GB perzistentního úložiště (flash). Pro vývoj je doporučen operační systém založený na distribuci Debian nebo Ubuntu, lze však použít další běžné distribuce, které však nejsou řádně otestovány.

V rámci této práce byl použit levný minipočítač CHIP vyvíjený společností Next Thing Co. Je distribuován za výhodnou cenu 9 USD.

Je vybaven vlastním čipem který používá 32bitové jádro ARM Cortex-R8 s architekturou Armv7-R a taktem 1 GHz. Dále obsahuje operační pamětí 512 MB typu DRAM a 4 GB úložiště NAND flash. Je osazen komunikačními rozhraními Wi-Fi b/g/n a Bluetooth 4.0. Výhodou je nízkoúrovňový přístup k periferiím GPIO, SPI, UART a dalším. Velké množství pinů je vyvedeno na konektory. Počítač je možné rozšiřovat dalšími deskami – např. pro podporu SD karet, Ethernet rozhraní apod. Pro vývoj byl použit operační systém založený na platformě Debian s verzí kernelu 4.4 [15].

Software pro gateway byl odzkoušen také na počítačích Raspberry Pi 3 a Raspberry Pi Zero W [51]. Pro komerční využití je vhodné zvolit jiné platformy s lepší stabilitou a komerční podporou. Jako rámcový příklad lze uvést Cascade-500 IoT Gateway vyvíjený společností Rigado [3]. Celé řešení je možné používat na výkonnějších 64bitových platformách, kde lze přenést zátěž z cloudového řešení a využít prvky umělé inteligence, analýzu dat v reálném čase a zpracování velkých dat.

### 4.4.2 OpenThread Border Router

OpenThread Border Router [37] je softwarový produkt, který je součástí projektu OpenThread. Implementuje základní funkcionalitu pro zařízení Border Router definované specifikací Thread. Software poskytuje operačnímu systému rozhraní sítě IPv6 a funkcionalitu pro ovládání sítě.

Součástí aplikace je také responzivní webové rozhraní pro sledování a konfiguraci sítě, které je standardně přístupné na portu 80. OpenThread Border Router dále podporuje konfiguraci sítě a přidávání nových zařízení prostřednictvím mobilní aplikace Thread 1.1 Commissioning App 1.1 [59]. Produkt je certifikovanou komponentou dle Thread verze 1.1.1.

#### OpenThread NCP

Důležitou komponentou zařízení gateway je síťový koprocesor. Je možné použít libovolnou platformu podporovanou projektem OpenThread. Pro tuto práci opět poslouží modul R41Z s mikrokontrolérem KW41Z. Součástí projektu je aplikace pro NCP, která využívá pro

komunikaci se zařízením gateway rozhraní UART nebo SPI. NCP poskytuje rozhraní pokrývající většinu funkcí knihovny OpenThread. Pro komunikaci s operačním systémem je použit průmyslový aplikační protokol Spinel [31].

#### Wpantund

Wpantund je ovladač síťového rozhraní pro OpenThread NCP. Jedná se o samostatnou aplikaci vyvinutou v rámci projektu OpenThread [31]. Ovladač vytváří síťové rozhraní IPv6 sítě WPAN, které je v rámci linuxového systému použitelné stejně jako ostatní standardní síťová rozhraní. Součástí wpantund je aplikace `wpanctl`, která poskytuje základní příkazy konzole pro ovládání a konfiguraci síťového koprocesoru. Součástí software je také rozhraní DBus a mechanismus pro aktualizaci firmware NCP.

Příklad použití aplikace `wpanctl`:

```
$ sudo wpanctl status
wpan0 => [
"NCP:State" => "offline"
"Daemon:Enabled" => true
"NCP:Version" => "OPENTHREAD/20170716-00518-g4e92a73-
   dirty; KW41Z; Mar 26 2018 11:35:28"
"Daemon:Version" => "0.08.00d (/ed49abb; Mar 21 2018
   19:10:24)"
"Config:NCP:DriverName" => "spinel"
"NCP:HardwareAddress" => [DED4040000000000]
]
$ sudo wpanctl scan
   | Joinable | NetworkName | PAN ID | Ch
---+----------+--------------------+--------+----
 1 | NO | "OpenThreadDemo" | 0x1234 | 15
```

### 4.4.3 Azure IoT Edge

Společnost Microsoft vyvíjí v rámci své platformy pro Internet věcí prostředí pokrývající proprietární sítě a jejich integraci se službou Azure IoT Hub. Pro tento účel slouží platforma Azure IoT Edge [62].

Jedná se o komplexní systém umožňující připojení tzv. hraničních zařízení do jednotného prostředí Azure IoT Hub. Návrh se skládá ze 3 základních částí.

- IoT Edge moduly – Na hraničních zařízeních lze spouštět výchozí nebo vlastní moduly v kontejnerovém prostředí Docker [6]. Moduly mohou provádět funkce jako: komunikace s proprietárními sítěmi pro Internet věcí, zpracování a analýza dat sbíraných z různých zařízení, aplikace umělé inteligence a rozpoznání vzorů a další. Vlastní moduly lze programovat v jazycích C, C++, C#, Python a Node.js.
- IoT Edge běhové prostředí – Zajišťuje automatizované nasazení výchozích i vlastních modulů v rámci kontejnerového prostředí. Umožňuje také autentizaci a komunikaci se službou IoT Hub.
- Cloudové rozhraní – Umožňuje vzdálenou konfiguraci, správu a monitoring jednotlivých zařízení v rámci služby Azure IoT Hub.

K zařízení výstupní brány s Azure IoT Edge se mohou připojit také zařízení komunikující protokolem MQTT stejným způsobem, jako se připojují ke koncovému bodu služby IoT Hub. Zařízení brány má vlastní TLS certifikát, může tedy analyzovat obsah zpráv. Konkretní aplikace jsou nasazovány jako moduly formou kontejnerů pro službu Docker. Jednotlivé kontejnery lze nasazovat a konfigurovat pomocí služby IoT Hub.

#### Popis modulu pro zařízení gateway

Moduly jsou spouštěny jako kontejnery v rámci služby Docker běžící na daném zařízení. V rámci této práce je použit jediný modul, který tvoří CoAP server a přijímané zprávy předává na výstupní proud IoT Edge.

Modul implementuje funkcionalitu dle návrhového vzoru *Protocol Translation* [14]. V rámci služby Azure IoT Hub je registrována pouze identita zařízení gateway. Prostřednictvím této identity jsou odesílány zprávy všech uzlů v dané síti. Z pohledu serverových aplikací lze adresovat pouze tuto identitu.

CoAP server poskytuje následující resource:

- `.well-known/core` – Jedná se o základní resource, která poskytuje údaje o službě. Zařízení rozesílají požadavek GET na multicastovou adresu `ff03::2`. Jakmile zařízení obdrží odpověď, pošle na cílovou adresu požadavek POST s parametry `ep={EUI64}` a `lt={timeout}`. Server si pak uloží k danému identifikátoru zařízení IPv6 adresu pro pozdější komunikaci.

- `time` – Resource slouží k získání aktuálního času v milisekundách. Pro potřebu přesnější synchronizace lze využít protokol NTP [32], který využívá protokolu transportní vrstvy UDP.

- `events` – Jedná se o resource pro příjem událostí ze zařízení. Ty jsou pak bez úprav odesílány do výstupního proudu.

Obsah všech zpráv je serializován ve formátu *JSON*. Data zprávy odeslané na službu IoT Hub mají následující podobu:

```
{
  deviceId: "{EUI64}",
  timestamp: {UNIX timestamp when processed},
  payload: {JSON document received from device}
}
```

### 4.4.4 Popis zapojení

Schéma v příloze C znázorňuje zapojení rozšiřující desky sloužící jako NCP koprocesor pro počítač CHIP. Jedná se o základní zapojení modulu U1 tvořeného čipem R41Z. Krystal Q1 je zdrojem hodinového signálu pro modul reálného času. Deska je napájena výstupem z regulátoru počítače s napětím 3,3 V. Komunikace probíhá prostřednictvím rozhraní LPUART0. Na konektor JP1 je vyvedeno rozhraní SWD pro nahrávání firmware a jeho ladění.

Zařízení CHIP musí být napájeno externím napájecím zdrojem s napětím 5 V a velikostí maximálního proudu alespoň 2 A.

### 4.4.5 Instalace a konfigurace

Následující část popisuje v jednotlivých krocích instalaci a konfiguraci softwarového vybavení pro zařízení gateway. Nejprve je potřeba nainstalovat vhodný firmware pro NCP koprocesor. Uvedené návody jsou popsány pro zmíněnou platformu CHIP s použitím čipu KW41Z jako NCP. Na jiných platformách se postup může lišit.

#### OpenThread NCP firmware

Jako první krok je potřeba nahrát příslušný firmware do NCP modulu. Pro sestavení programu je nutné nainstalovat nástroje GNU Embedded Toolchain pro Arm, které slouží k sestavení software pro jádra ARM Cortex-M v GNU prostředí. Přenositelný archiv s nejnovější verzí lze stáhnou z webu společnosti Arm [12].

Následuje postup stažení nejnovější verze OpenThread [35] ze služby GitHub a sestavení firmware pro Border Router NCP:

```
$ git clone https://github.com/openthread/openthread.git
$ cd openthread
$ make -f examples/Makefile-kw41z TMF_PROXY=1
    BORDER_ROUTER=1 COMMISSIONER=1
$ cd output/kw41z/bin
$ arm-none-eabi-objcopy -O binary ot-ncp-ftd
    ot-ncp-ftd.bin
```

V adresáři `output/kw41z/bin/` se poté nachází binární soubor `ot-ncp-ftd.bin`. Nahrání firmware do flash paměti čipu KW41Z lze provést pomocí programátoru J-Link [19]. Moduly R41Z jsou distribuovány s předinstalovaným zavaděčem, který umožňuje nahrání firmware pomocí rozhraní UART nebo metodou OTA[11] [53].

#### Instalace operačního systému

Pro nahrání operačního systému do počítače CHIP lze použít rozšíření do prohlížeče Chrome/Chromium [9], nebo nástroje CHIP SDK [16]. V rámci práce byla použita *Headless* verze operačního systému s kernelem verze 4.4 . Tato verze neobsahuje grafické uživatelské rozhraní a je

[11] OTA – Over the Air

tak vhodná pro odlehčené servery. Následující postup popisuje krok po kroku instalaci software a konfiguraci služeb pro lepší pochopení fungování zařízení gateway. V produkčním prostředí bude nezbytné použít obraz operačního systému s předinstalovaným softwarem.

Pro nahrání souboru s operačním systémem musí být na desce propojen pin FEL se zemí (GND). Po připojení zařízení k USB portu počítače se zařízení spustí v režimu FEL, který umožňuje programování paměti flash. Poté stačí použít jeden z uvedených nástrojů [10]. Po nahrání operačního systému a opětovném připojení k počítači pomocí USB portu je možné přistupovat k terminálu prostřednictvím sériového rozhraní COM. Alternativně se lze k terminálu připojit pomocí rozhraní UART na pinech UART1-TX a UART1-RX. V systému zařízení je pak vytvořen výchozí uživatelský účet správce se jménem `chip` a heslem `chip`. Doporučuje se heslo před pokračováním změnit. Úkony prováděné na samostatném počítači jsou popsány pro operační systém Ubuntu.

Pro další postup je nezbytné připojení k Internetu. Následujícím příkazem lze nastavit konfiguraci Wi-Fi sítě:

```
$ sudo nmcli device wifi connect '{SSID}' password
    '{password}' ifname wlan0
```

Následně je vhodné provést aktualizaci software:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Pro snadnější používání je dále doporučeno použít SSH spojení. Vhledem k většímu vytížení procesoru při instalaci software je pro zajištění stability vhodné použít napájení s maximální velikostí proudu alespoň 2 A.

### Instalace OpenThread Border Router

Dále je potřeba nainstalovat software OpenThread Border Router, který zprostředkuje rozhraní pro konfiguraci a komunikaci se sítí Thread. Border Router využívá NCP zařízení připojené k počítači pomocí rozhraní UART na pinech UART1-TX a UART1-RX. Aby bylo možné

rozhraní používat pro komunikaci s NCP, je nutné deaktivovat službu, která umožňuje používaní terminálu přes UART:

```
$ sudo systemctl disable serial-getty@ttyS0.service
```

Zdrojové kódy OpenThread Border Router jsou dostupné na službě GitHub [38]:

```
$ cd ~
$ git clone https://github.com/openthread/borderrouter
```

Sestavení a instalace závislostí a samotného software se provede pomocí následujících skriptů. Tento proce může na platformě CHIP trvat až hodinu.

```
$ cd borderrouter
$ ./script/bootstrap
$ ./script/setup
```

### Konfigurace síťové komunikace

Následující postup znázorňuje nastavení síťové komunikace výstupní brány. Uvedený návod je přizpůsoben pro platformu CHIP, obdobný postup pro platformu Raspberry Pi je popsán na stránkách projektu [39]. Nejprve je třeba nastavit rozhraní pro komunikaci s NCP zařízením v konfiguračním souboru `/etc/wpantund.conf` služby `wpantund`:

```
set Config:NCP:SocketPath "/dev/ttyS0"
```

Poté se nainstaluje služba `hostapd` pro vytvoření přístupového bodu sítě Wi-Fi pomocí:

```
$ sudo apt-get install hostapd
```

Konfigurace služby je provedena vytvořením nového souboru `/etc/network/interfaces.d/wlan1` obsahujícího nastavení síťového rozhraní `wlan1`:

```
allow-hotplug wlan1
iface wlan1 inet static
    address 192.168.1.2
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
```

V konfiguračního souboru `/etc/hostapd/hostapd.conf` je třeba nastavit následující konfiguraci přístupového bodu:

```
interface=wlan1
driver=nl80211
ssid=gateway
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=chipchip
rsn_pairwise=CCMP
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Konfigurace vytvoří přístupový bod IEEE 802.11/n na kanálu 6 s názvem `gateway`. Připojení je zabezpečeno pomocí WPA2-PSK s klíčem `chipchip`. V souboru `/etc/systemd/system/hostapd.service` je nutné nastavit následující konfiguraci služby `hostapd`:

```
[Unit]
Description=Hostapd IEEE 802.11 Access Point
After=sys-subsystem-net-devices-wlan1.device
BindsTo=sys-subsystem-net-devices-wlan1.device
```

```
[Service]
Type=forking
PIDFile=/var/run/hostapd.pid
ExecStart=/usr/sbin/hostapd -B /etc/hostapd/hostapd.conf
    -P /var/run/hostapd.pid

[Install]
WantedBy=multi-user.target
```

V souboru `/etc/rc.local` musí být vložen před řádek `exit 0` následující příkaz, který zajistí automatické spuštění služby `hostapd` po startu systému:

```
sudo service hostapd start
```

Po restartu počítače by pak měl být viditelný Wi-Fi přístupový bod s SSID `gateway` a heslem `chipchip`. Nyní je třeba nainstalovat službu `dnsmasq` sloužící jako DNS a DHCP server.

```
$ sudo apt-get install dnsmasq
```

Konfigurace v souboru `/etc/dnsmasq.conf` bude upravena následujícím způsobem:

```
interface=wlan1
listen-address=192.168.1.2
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
dhcp-range=192.168.1.50,192.168.1.150,12h
```

To nastaví službu na poslech na IP adrese `192.168.1.2`. DNS služba bude přeposílat požadavky na servery společnosti Google na adrese `8.8.8.8`. DHCP služba bude přidělovat IP adresy v rozmezí `192.168.1.50` až `192.168.1.150`.

Nakonec je třeba nainstalovat a nakonfigurovat službu `tayga` pro překlad síťových adres (NAT) z IPv6 do IPv4.

```
$ sudo apt-get install tayga
```

V konfiguračním souboru `/etc/tayga.conf` se změní nastavení NAT následujícím způsobem:

```
prefix 64:ff9b::/96
dynamic-pool 192.168.255.0/24
ipv6-addr 2001:db8:1::1
ipv4-addr 192.168.255.1
```

Aktivace služby `tayga` je provedena nastavením atributu `RUN` v souboru `/etc/default/tayga`:

```
RUN="yes"
```

Nyní zbývá nastavit pravidla pro přeposílání komunikace v konfiguraci síťového rozhraní:

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
$ sudo sh -c "echo 1 > /proc/sys/net/ipv6/conf/all/
    forwarding"
```

Je třeba se ujistit, že je povoleno přeposílání paketů IPv4 v souboru `/etc/sysctl.conf`:

```
net.ipv4.ip_forward=1
```

Následující skript nastaví pravidla směrování mezi rozhraními `wlan0` a `wlan1` a uloží konfiguraci *NAT*:

```
$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j
    MASQUERADE
$ sudo iptables -A FORWARD -i wlan0 -o wlan1 -m state
    --state RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -i wlan1 -o wlan0 -j ACCEPT
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Do souboru `/etc/rc.local` pak musí být před řádek `exit 0` vložen příkaz, který po startu systému zavede předchozí konfiguraci `iptables`:

```
iptables-restore < /etc/iptables.ipv4.nat
```

Po aplikaci zmíněného postupu je nakonfigurován software OpenThread Border Router, tvořící výchozí bránu do lokální sítě a Internetu. Operační systém má nyní k dispozici síťové rozhraní `wpan0`. Pro konfiguraci sítě lze využít aplikaci `wpanctl` nebo webové rozhraní dostupné na adrese `192.168.1.2` po připojení k přístupovému bodu.

### Instalace Azure IoT Edge

Druhou nedílnou součástí výstupní brány je aplikace Azure IoT Edge. Ta ke svému fungování potřebuje nainstalované běhové prostředí Docker [14]. Nejprve je nezbytné nainstalovat potřebné závislosti:

```
$ sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg2 \
software-properties-common
```

Samotnou aplikaci Docker je možné nainstalovat z oficiálního repozitáře produktu [11] spuštěním následujících příkazů:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg
    | sudo apt-key add -
$ echo "deb [arch=armhf] https://download.docker.com/
   linux/debian \
$(lsb_release -cs) stable" | \
$ sudo tee /etc/apt/sources.list.d/docker.list
$ sudo apt-get update
$ sudo apt-get install docker-ce
```

Produkt Azure IoT Edge se instaluje pomocí nástroje `pip`. Potřebné závislosti se nainstalují následujícími příkazy:

```
$ sudo apt-get install python-pip
$ sudo pip install -U setuptools pip
$ sudo apt-get install python2.7-dev libffi-dev
    libssl-dev -y
```

Pro architekturu ARM je v tuto chvíli dostupná pouze *preview* verze běhového prostředí Azure IoT Edge, která se nainstaluje příkazem:

```
$ sudo pip install -U azure-iot-edge-runtime-ctl --pre
```

Po aplikaci předchozího postupu je připravené běhové prostředí Azure IoT Edge. Konfigurace služby bude popsána později.

#### Sestavení obrazu kontejneru pro Docker

Před konfigurací IoT Edge je potřeba nahrát kontejner do nějakého registru. Protože je celé řešení postaveno na platformě Microsoft Azure, je vhodné využít například službu Azure Container Registry. Instanci služby je možné vytvořit dle návodu v dokumentaci [49]. Po vytvoření Container Registry se musí nastavit autentizace služby Docker:

```
$ docker login {login server} -u {username} -p {password}
```

Nyní je potřeba sestavit modul a vytvořit kontejner pro službu Docker. Zdrojové soubory se nachází v adresáři `iot_edge_modules`. Pro sestavení použijeme libovolný počítač s nainstalovanou aplikací Docker. Lze jej sestavit i na zařízení CHIP, to však může trvat výrazně déle.

Kontejner je rozdělen do tří samostatných částí, které lze použít pro tvorbu dalších modulů a nebude tak docházet ke zbytečné duplikaci kontejnerů se stejnou funkcionalitou. Soubor `Dockerfile.base` obsahuje instrukce pro sestavení kontejneru s nainstalovanými nástroji Azure IoT SKD pro jazyk *C*. `Dockerfile.coap` po sestavení obsahuje nainstalovanou knihovnu `libcoap`, která je vyžadována modulem pro bránu. Veškeré instrukce pro vytvoření a instalaci kontejneru se samotným modulem jsou pokryté souborem `Dockerfile.gateway`.

Následující příkazy sestaví potřebné části, vytvoří nový kontejner označený názvem `gateway` a nainstalují program modulu:

```
$ sudo docker build -t baseimage -f Dockerfile.base .
$ sudo docker build -t coap -f Dockerfile.coap .
$ sudo docker build -t gateway -f Dockerfile.opaque .
```

Dále se nahraje daný kontejner do zvoleného registru:

```
$ docker tag gateway {login server}/gateway
$ docker push
```

### Konfigurace Azure IoT Hub a zavedení modulů pro Docker

V tuto chvíli je třeba vytvořit novou instanci služby IoT Hub dle návodu v dokumentaci [50]. Funkcionalita pro hraniční zařízení je dostupná pouze v rámci škálování alespoň na úrovni Standard. Pomocí webového portálu služby Azure dále vytvořte identitu zařízení IoT Edge s libovolným identifikátorem (*device ID*). Identita má přidělen primární klíč pro autentizaci.

Jakmile je vytvořená identita hraničního zařízení, zbývá nakonfigurovat běhové prostředí IoT Edge na gateway. Nastavení přihlašovacích údajů pro přístup ke službě Container Registry (nebo jinému zvolenému registru) se provede příkazem:

```
$ sudo iotedgectl login --address {login server}
    --username {username} --password {password}
```

Další příkaz nakonfiguruje identitu zařízení IoT Edge pomocí přihlašovacího řetězce, který lze nalézt ve webovém portálu:

```
$ sudo iotedgectl setup --connection-string {connection
    string} --auto-cert-gen-force-no-passwords
```

Nyní se spustí běhové prostředí Azure IoT Edge:

```
$ sudo iotedgectl start
```

Po spuštění by měl v rámci služby Docker běžet proces s názvem `edgeAgent`. Zbývá nakonfigurovat moduly zařízení IoT Edge prostřednictví služby IoT Hub [50]. Nastavení modulů lze provést opět pomocí webového portálu. Je třeba nastavit libovolné jméno modulu (např. `gateway`), URI ve tvaru `{login server}/coapserver` a použít následující JSON pro konfiguraci kontejneru:

```
{
  "NetworkMode": "host",
  "HostConfig": {
    "PortBindings": {
      "5683/udp": [
        { "HostPort": "5683/udp" }
      ]
    }
  }
}
```

V nastavení *Specify Routes* se nastaví směrování všech zpráv z modulu `gateway` (nebo jiného uživatelem definovaného) do výstupního proudu `upstream` pomocí JSON konfigurace:

```
{
  "routes": {
    "coapToCloud": "FROM /messages/modules/gateway/* INTO
       $upstream"
  }
}
```

Události obsahující měřená data jsou ve výchozím stavu modulem posílána na výstup `coapOutput`. Pro jejich izolovaný odběr lze využít výstup `/messages/modules/gateway/outputs/coapOutput` [60].

Ostatní možnosti je možné ponechat ve výchozím nastavení. Po potvrzení se daný kontejner stáhne na zařízení gateway a modul by měl být následně ve stavu *running*. Po restartování brány se všechny služby spustí automaticky a není potřeba žádná uživatelská akce.

### 4.4.6 Konfigurace sítě a přidání zařízení

Síť lze zformovat dvěma způsoby. První možnost je použít webové rozhraní na portu 80 [40]. V hlavním menu stačí zvolit možnost *Form*, vyplnit požadované údaje a pomocí tlačítka *Form* vytvořit síť. Druhou možností je využít aplikaci `wpanctl`.

Nejprve je nutné vygenerovat sdílený klíč pro autentizaci Commissioner zařízení. Klíč se generuje z XPAN ID, názvu sítě a zvoleného textového tajemství s délkou alespoň 6 znaků pomocí následujícího skriptu:

```
$ cd ~/borderrouter/tools/
$ ./pskc {secret} {XPAN ID} {network name}
```

Zde je příklad s výchozími hodnotami webového rozhraní:

```
$ ./pskc 123456 1111111122222222 OpenThreadDemo
  61e1206d2c2b46e079eb775f41fc7219
```

Těmito příkazy se poté nastaví konfigurace sítě (s využitím vygenerovaného klíče):

```
$ sudo wpanctl setprop Network:PANID 0x1234
$ sudo wpanctl setprop Network:XPANID 1111111122222222
$ sudo wpanctl setprop Network:Key
    00112233445566778899aabbccddeeff
$ sudo wpanctl setprop Network:PSKc --data
    61e1206d2c2b46e079eb775f41fc7219
$ sudo wpanctl config-gateway -d "fd11:22::"
```

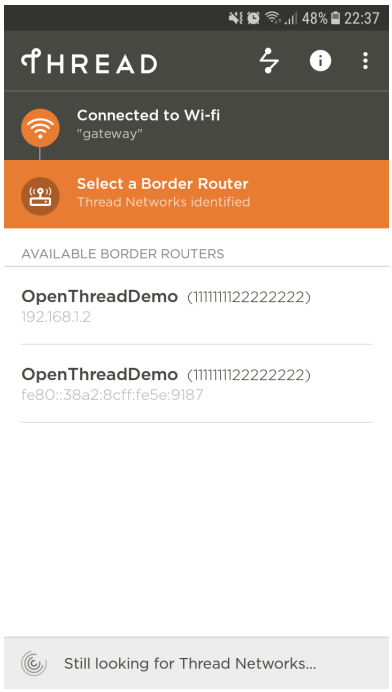
Nakonec je zformována síť s názvem `OpenThreadDemo` příkazem:

```
$ sudo wpanctl form "OpenThreadGuide"
```

Pokud se síť povedlo úspěšné zformovat, měl by příkaz `wpanctl status` vypsat výstup podobný tomuto:

```
$ sudo wpanctl status
  wpan0 => [
  "NCP:State" => "associated"
  "Daemon:Enabled" => true
  "NCP:Version" => "OPENTHREAD/20170716-00518-g4e92a73-
     dirty; KW41Z; Mar 26 2018 11:35:28"
  "Daemon:Version" => "0.08.00d (/ed49abb; Mar 21 2018
     19:10:24)"
  "Config:NCP:DriverName" => "spinel"
  "NCP:HardwareAddress" => [DED4040000000000]
  "NCP:Channel" => 15
  "Network:NodeType" => "leader"
  "Network:Name" => "OpenThreadDemo"
  "Network:XPANID" => 0x1111111122222222
  "Network:PANID" => 0x1234
  "IPv6:LinkLocalAddress" => "fe80::dcfa:d87f:5747:d007"
  "IPv6:MeshLocalAddress" => "fd11:1111:1122:0:c4c1:d699:
     f5bf:a26f"
  "IPv6:MeshLocalPrefix" => "fd11:1111:1122::/64"
  "com.nestlabs.internal:Network:AllowingJoin" => false
  ]
```

Pro přidání zařízení je možné využít mobilní aplikaci Thread 1.1 Commissioning App [59]. Po připojení k přístupovému bodu brány nakonfigurovaném v části 4.4.5 a spuštění aplikace by se měla zobrazit úvodní obrazovka jako na obrázku 4.10.




Obrázek 4.10: Úvodní obrazovka Thread 1.1 Commissioning App (zdroj: autor)

Nyní je třeba zvolit bránu `OpenThreadDemo` a zadat heslo `123456` (nebo použít vlastní údaje z předchozího postupu). Dále je třeba zadat údaje o přidávané zásuvce – identifikátor zařízení *EUI-64* a klíč *PSKD* nakonfigurovaný v souboru `smart_socket_config.h`. To je možné provést ručním zadáním nebo vyfocením QR kódu, který obsahuje řetězec ve tvaru:

```
v=1&&eui={EUI-64}&&cc={PSKD}
```

Proces přidání na samotné zásuvce se zahájí krátkým stiskem tlačítka. LED dioda by měla při správném průběhu blikat rychle, což signalizuje proces přidání. Následně by měla blikat pomalu a poté přestat svítit. Pokud se proces nezdařil, kontrolka zůstane svítit stejně

jako v iniciálním stavu. O úspěšném přidání by měla informovat také mobilní aplikace.

### 4.4.7 Licence použitého software

Veškeré zdrojové kódy vytvořené v rámci této diplomové práce jsou volně použitelné v rámci licence 3-Clause BSD License [58]. Licenční podmínky použitého software třetích stran jsou obsažené v přiložených zdrojových souborech.

Veškerý software vytvořený v této práci slouží pouze pro verifikaci prototypového řešení. Není vhodné software používat v produkčním prostředí a autor nenese zodpovědnost za případné újmy způsobené jeho používáním.

# 5 Vyhodnocení řešení

Poslední část práce se zabývá testováním a vyhodnocením vytvořeného prototypu. Cílem je zjistit, do jaké míry prototyp systému splňuje požadavky zadání, pokusit se navrhnout případná zlepšení a nasměrovat další postup vývoje.

## 5.1 Testování řešení

Síť Thread využívá pásmo 2,4 GHz a minimalizuje výstupní výkon radiového modulu. Jak bylo popsáno v části 2.2, signál na této frekvenci má vyšší útlum a hůře prochází překážkami. Důležitým kritériem testování je tedy schopnost sítě dostatečně pokrýt budovu signálem tak, aby všechna zařízení tvořila souvislou topologii.

Během vývoje byly dostupné pouze dva exempláře komunikačních modulů, byla tedy testována pouze přímá komunikace mezi dvěma uzly. Byla vybrána 3 různá prostředí – cihlová stavba (rodinný dům), železobetonová stavba (kancelářská budova) a volné prostranství (přijímač i vysílač byly umístěny 1,5 metru nad zemí). Interiér budov byl navíc zarušen běžným provozem Wi-Fi sítí. Při testování byla sledována vzdálenost mezi vysílačem a přijímačem, průměrná odezva, indikátor síly přijímaného signálu (RSSI), dispozice umístění komunikujících uzlů a překážek a počet výpadků. Pro každý test bylo provedeno 10 pokusů o odeslání dat. Do odezvy se promítá i doba zpracování požadavku na zařízení gateway (nejedná se o ping).

| Vzdálenost | Odezva | Dispozice | RSSI | Výpadky |
|---|---|---|---|---|
| 1 m | 49 ms | přímá viditelnost | -53 dBm | žádné |
| 4 m | 52 ms | přímá viditelnost | -71 dBm | žádné |
| 4 m | 54 ms | různá poschodí | -73 dBm | žádné |
| 6 m | 56 ms | různá poschodí | -77 dBm | žádné |
| 6 m | 56 ms | sousední místnost | -81 dBm | žádné |
| 10 m | 62 ms | sousední místnost | -90 dBm | žádné |
| 13 m | 61 ms | přes 2 místnosti | -90 dBm | 2/10 |
| 16 m | | přes 2 místnosti | bez spojení | |

Tabulka 5.1: Testování komunikace v cihlové budově

| Vzdálenost | Odezva | Dispozice | RSSI | Výpadky |
|---|---|---|---|---|
| 3 m | 58 ms | přímá viditelnost | -71 dBm | žádné |
| 6 m | 62 ms | přímá viditelnost | -75 dBm | žádné |
| 20 m | 56 ms | přímá viditelnost | -75 dBm | žádné |
| 40 m | 55 ms | přímá viditelnost | -82 dBm | žádné |
| 2 m | 62 ms | různá poschodí | -83 dBm | 1/10 |
| 8 m | 102 ms | různá poschodí | | 7/10 |
| 5 m | | přes 2 poschodí | bez spojení | |
| 12 m | 70 ms | sousední místnost | -78 dBm | 1/10 |
| 20 m | 81 ms | sousední místnost | -90 dBm | 3/10 |

Tabulka 5.2: Testování komunikace v železobetonové budově

| Vzdálenost | Odezva | Dispozice | RSSI | Výpadky |
|---|---|---|---|---|
| 3 m | 47 ms | přímá viditelnost | -71 dBm | žádné |
| 10 m | 55 ms | přímá viditelnost | -75 dBm | žádné |
| 25 m | 52 ms | přímá viditelnost | -80 dBm | žádné |
| 40 m | 58 ms | přímá viditelnost | -81 dBm | žádné |
| 50 m | 55 ms | přímá viditelnost | -77 dBm | žádné |
| 75 m | 59 ms | přímá viditelnost | -86 dBm | žádné |
| 90 m | 65 ms | přímá viditelnost | -90 dBm | 1/10 |
| 100 m | 71 ms | přímá viditelnost | | 6/10 |
| 120 m | | přímá viditelnost | bez spojení | |

Tabulka 5.3: Testování komunikace na volném prostranství

Testování ukázalo, že dva uzly jsou v cihlové budově schopné spolehlivě komunikovat v sousední místnosti nebo poschodí na vzdálenost alespoň 10 metrů. Za předpokladu, že bude v každé místnosti rodinného domu umístěna alespoň jedna zásuvka, by mělo být možné zformování souvislé sítě a zajištění spolehlivé komunikace. V kancelářské budově je větší útlum signálu mezi podlažími a více rušení ve stejném pásmu. Při rozumném rozmístění komunikujících uzlů by ale mělo být možné vytvoření souvislé topologie i mezi podlažími. Chování na volném prostranství překonalo očekávání a potvrdila se udávaná maximální vzdálenost 100 metrů.

Bylo také sledováno chování při výpadcích jednotlivých částí systému. Při výpadku a následné ztrátě konektivity zařízení zásuvky s gateway dochází k opakovanému hledání CoAP serveru. Jakmile je nějaký uzel sítě opět v dosahu, dojde k opětovnému připojení bez nutnosti vnějšího zásahu. Zařízení pak dostane odpověď serveru a pokračuje v komunikaci. Dojde-li k výpadku konektivity s internetovou síti na straně gateway, služba IoT Edge umožňuje dočasné uchovávání odesílaných zpráv. Nedojde tak ke ztrátě dat.

Prototyp zařízení byl pro testování vytvořen pomocí základních vývojových desek dodávaných výrobci použitých mikročipů. Z toho

důvodu nemá smysl testovat a analyzovat odchylky měření, protože je deska osazena součástkami s jinými vlastnostmi.

## 5.2 Známé chyby

Během vývoje prototypu bylo odhaleno několik chyb v komponentách třetích stran, které omezují některé funkcionality výsledného systému. Všechny nalezené problémy byly nahlášeny a potvrzeny k řešení. Chyby nijak neovlivnily implementaci a po jejich opravení stačí aktualizovat dotčené nástroje.

První chyba nastává při pokusu o přidání zásuvky do sítě pomocí mobilní aplikace [20]. Při navazování DTLS spojení mezi Joiner a Commissioner zařízeními občas dochází k chybě při odesílání rámce se *server hello* fyzickou vrstvou NCP koprocesoru. Chyba se zjevně vyskytuje v portu knihovny OpenThread na platformu KW41Z. U jiných platforem nebylo zmíněné chování nahlášeno.

Další chybné chování bylo odhaleno v rámci aplikace Azure IoT Edge. Při konfiguraci modulu pro použití síťového módu *bridge* dojde k nespecifikované chybě při nasazování kontejneru. Modul pak není správně zaveden a je z pohledu služby IoT Hub ve stavu *backup*. Problém lze obejít ponecháním výchozího módu sítě. Systém běžící v rámci kontejneru ale v takovém případě nemá přímý přístup k rozhraní `wpan0` a IPv6 adresy zásuvek jsou maskovány adresou výchozí brány virtuální sítě. V takovém případě je nezbytné nastavit NAT překlad adres na IPv4 adresy.

Poslední nalezený problém je spjatý s platformou CHIP. Fungování hardware je poměrně nestabilní a dochází k občasnému zastavení zařízení. Při vývoji dokonce došlo k situaci, kdy byla při takovém výpadku porušena perzistentní paměť a selhalo zavedení operačního systému po opětovném spuštění zařízení. Bylo zjištěno že k problému dochází při větší zátěži (sestavení programu apod.), kdy se nadměrně zahřívá procesor. Problém také zřejmě souvisí s nedostatečným a nestabilním napájením. Je tak potřeba umožnit dostatečné odvětrávání tepla a zařízení napájet zdrojem s maximálním proudem alespoň 2 A. Musí být také použit dostatečně odstíněný přívodní kabel.

## 5.3 Návrhy na zlepšení

Kromě zmíněných chyb obsahuje výsledný prototyp několik nedostatků, které omezují funkčnost řešení a měly by být během dalšího vývoje odstraněny.

V rámci projektu OpenThread je použita implementace zabezpečení komunikace a kryptokrafických funkcí v rámci knihovny MbedTLS [55]. Výchozí implementace počítá všechny kryptografické funkce jádrem procesoru. Na pomalejších platformách to způsobuje delší prodlevy v komunikaci. Například během ustavení DTSL spojení dochází k opakovanému posílání zpráv *retransmission*, což občas vede k selhání procesu přidání zařízení do sítě.

Zařízení používá ke komunikaci s bránou Mesh-Local adresování. Při výpadcích dochází ke změnám IPv6 adresy zařízení. To způsobí selhání komunikace se zařízením gateway a je nutné znovu rozeslat multicastový požadavek a znovu zaregistrovat zařízení. Při slabší síle signálu může docházet k tomuto chování poměrně často. Bylo by vhodnější změnit koncept takovým způsobem, aby se při komunikaci používaly globální adresy (GUA). Změny v topologii sítě by pak neomezovaly komunikaci. Místo hledání serveru pomocí multicastové adresy je možné zaregistrovat jeho adresu ve sdílených datech sítě spravovaných Leader uzlem.

## 5.4 Další postup

Zde jsou v několika bodech popsány dílčí činnosti, které by měly následovat během dalšího vývoje:

- Otestovat software zásuvky na jiné platformě – Od začátku realizace této práce se na trhu objevilo několik nových platforem, které podporují síť Thread. Některé jsou stále ve fázi zasílání vzorků, ale jejich prodej by měl začít během roku 2018. Jako lepší alternativu je možné použít mikrokontrolér nRF52840 vyráběný společností Nordic Semiconductor [34]. Ten podporuje síť Thread, Zigbee ale také technologii Bluetooth Mesh v rámci Bluetooth 5.0. Navíc poskytuje větší hardwarové zdroje a obsahuje kryptografický koprocesor a další užitečné periferie.

- Integrovat zařízení do webového rozhraní – Součástí existující platformy pro Internet věcí vyvíjené společností TESCO SW je webové rozhraní, které nyní umožňuje ovládání Wi-Fi zásuvek. Systém pro komunikaci využívá službu Azure IoT Hub. Je potřeba webové rozhraní rozšířit o další typ zařízení.

- Implementovat obousměrnou komunikaci – Modul brány IoT Edge je schopný rozpoznat IPv6 adresy jednotlivých zařízení. Pro odesílání dat stačí jednosměrná komunikace. V případě potřeby vzdáleného ovládání bude nezbytné implementovat symetricky komunikaci s CoAP serverem na straně zařízení. Z pohledu služby Azure IoT Hub je možné příkazy pro zařízení implementovat pomocí funkcionality *Direct Method*.

- Návrh a realizace DPS – Pro řádné otestování navrženého zařízení bude nutné odborně navrhnout desku plošného spoje s ohledem na dodržení izolačních vzdáleností silnoproudé části a zajištění co největší miniaturizace.

- Najít adekvátní platformu pro zařízení brány – Platforma CHIP není vzhledem k limitované podpoře a přetrvávajícím chybám v hardware vhodná pro komerční využití. Bude tedy potřeba najít adekvátní alternativu, kterou by bylo možné prodávat a snadněji udržovat.

- Automatizovat nasazení a aktualizace software gateway – Nové verze aplikační logiky lze snadno distribuovat prostřednictvím modulů služby Azure IoT Edge. Bude však nutné zajistit uživatelsky přívětivou aktualizaci běhového prostředí IoT Edge a OpenThread Border Router. Vhodnou cestou je například distribuce balíčků v rámci vlastního repozitáře.

- Implementovat aktualizaci firmware zásuvek OTA – V budoucnu bude také nezbytné distribuovat nové verze firmware do jednotlivých zásuvek a dalších zařízení. Bude nutné implementovat funkcionalitu pro bezdrátovou aktualizaci firmware.

# Závěr

Cílem diplomové práce bylo navrhnout zařízení pro měření elektrických veličin na připojeném spotřebiči, které bude komunikovat pomocí vhodné bezdrátové sítě. Zařízení má být zabudováno přímo v elektrické zásuvce a je určeno pro použití v budovách v řádu desítek až stovek kusů. Z analyzovaných technologií byla vybrána síť Thread. Jedná se o poměrně nový protokol umožňující vytváření samoorganizujících se mesh sítí s nízkými požadavky na výkon a napájení jednotlivých uzlů. Komunikace je implicitně založená na protokolu IPv6 s využitím běžně používaného transportního protokolu UDP. Tím je umožněno použití standardních aplikačních protokolů jako CoAP.

Pro realizaci modulu zásuvky byla vybrána platforma KW41Z. Ta kromě sítě Thread podporuje také Zigbee a Bluetooth 4.2. Pro měření elektrických veličin byl vybrán osvědčený specializovaný mikročip STPM32. V základní implementaci jsou sledovány veličiny jako napětí v elektrické síti, proud procházející spotřebičem, příkon spotřebiče a jeho celková spotřeba energie. Zapojení může fungovat se zátěží velikosti až 16 A (3600 W).

Součástí práce je také software pro hraniční zařízení, které slouží jako výchozí brána sítě Thread do Internetu. Zároveň přeposílá události přijaté z jednotlivých zařízení do cloudové služby Azure IoT Hub. Ta slouží ke sběru dat, správě a ovládání zařízení. Zařízení gateway je možné jednoduchým způsobem rozšiřovat o další funkcionalitu a může v budoucnu sloužit jako centrum domácí automatizace pro zařízení komunikující prostřednictvím služby IoT Hub. Pro komunikací zařízení s gateway byl zvolen odlehčený aplikační protokol CoAP. Síť je možné snadno ovládat pomocí webového rozhraní nebo mobilní aplikace.

Při implementaci systému byly prozkoumány nové technologie, které jsou na trhu jen několik let nebo ještě vůbec nejsou běžně dostupné. Výsledné řešení je tak unikátní a poskytuje prostor pro další rozvoj. Důsledkem je však také skutečnost, že bylo odhaleno několik chyb v produktech třetích stran. Při dalším vývoji je tedy nutná součinnost s jejich výrobci.

Testování ukázalo, že je možné zařízení používat tak, aby síť pokryla obytné domy, administrativní budovy a případně výrobní haly. Na otevřeném prostranství, v halách a otevřených kancelářích je dosah až 100 metrů. Kvůli použité frekvenci 2,4 GHz je šíření signálu horší především v železobetonových kancelářských budovách. Pro spolehlivé pokrytí tak bude nutné navrhnout vhodné rozmístění jednotlivých zásuvek s ohledem na nalezení cesty mezi různými poschodími. Díky napájení z elektrické sítě mohou být zásuvky neustále aktivní a vytvářet tak síť směrovačů pro použití dalších typů zařízení.

Nakonec byly shrnuty podklady pro další rozvoj systému. Měla by být dále implementována plnohodnotná obousměrná komunikace s případným využitím přístupu *Identity Translation* [14]. Je také vhodné zkusit přenést aplikaci na další platformy podporující síť Thread, které budou na trhu dostupné až během roku 2018. Řešení bude potřeba plně integrovat do stávající platformy pro chytré zásuvky. Díky službě IoT Hub je integrace měřicích zařízení snadná. Bude ale nutné implementovat funkcionalitu pro vzdálené ovládání zařízení gateway a sítě Thread.

# Literatura


[1] ASSP for metering applications with up to four independent 24-bit 2nd order sigma-delta ADCs, 4 MHz OSF and 2 embedded PGLNA. *STMicroelectronics* [online]. STMicroelectronics, 2018 [cit. 2018-05-08]. Dostupné z: http://www.st.com/en/data-converters/stpm32.html

[2] KRATOCHVÍL, Petr. Bezpečnost internetu věcí. *CHIP*. 2016, **2016**(12), 58-60.

[3] Cascade-500 IoT Gateway. *Rigado* [online]. Rigado, 2018 [cit. 2018-05-08]. Dostupné z: https://www.rigado.com/products/iot-gateways/

[4] HUI, Jonathan W. a Pascal THUBERT. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. *IETF Tools* [online]. IETF Tools Team, 2011 [cit. 2018-05-07]. Dostupné z: https://tools.ietf.org/html/rfc6282

[5] RESCORLA, Eric. Datagram Transport Layer Security Version 1.2. *IETF Tools* [online]. RTFM, 2012 [cit. 2018-05-08]. Dostupné z: https://tools.ietf.org/html/rfc6347

[6] *Docker - Build, Ship, and Run Any App, Anywhere* [online]. Docker, 2018 [cit. 2018-05-08]. Dostupné z: https://www.docker.com/

[7] TRČÁLEK, Antonín. Dokonalé propojení. *Computer Extra: Chytrá domácnost*. 2018, **2018**(1), 32-35. ISSN 1210-8790.

[8] External Thread Commissioning. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/guides/border_router/external_commissioning

[9] *Flasher* [online]. Next Thing Co., 2017 [cit. 2018-05-08]. Dostupné z: https://flash.getchip.com/

[10] Flash CHIP With an OS. *Next Thing Co. Documentation* [online]. Next Thing Co., 2018 [cit. 2018-05-08]. Dostupné z: https://docs.getchip.com/chip.html#flash-chip-with-an-os

[11] Get Docker CE for Debian. *Docker Documentation* [online]. Docker, 2018 [cit. 2018-05-08]. Dostupné z: https://docs.docker.com/install/linux/docker-ce/debian/

[12] GNU Arm Embedded Toolchain. *Arm Developer* [online]. Arm Limited, 2018 [cit. 2018-05-08]. Dostupné z: https://developer.arm.com/open-source/gnu-toolchain/gnu-rm

[13] *Hexade Smartplug* [online]. Olomouc: TESCO SW, 2017 [cit. 2018-05-06]. Dostupné z: https://www.smartplug.cz/

[14] How an IoT Edge device can be used as a gateway. *Microsoft Docs* [online]. Microsoft, 2017 [cit. 2018-05-08]. Dostupné z: https://docs.microsoft.com/en-us/azure/iot-edge/iot-edge-as-gateway

[15] CHIP Hardware. *Next Thing Co. Documentation* [online]. Next Thing Co., 2018 [cit. 2018-05-08]. Dostupné z: https://docs.getchip.com/chip.html#chip-hardware

[16] CHIP-SDK. *GitHub* [online]. Next Thing Co., 2017 [cit. 2018-05-08]. Dostupné z: https://github.com/NextThingCo/CHIP-SDK

[17] IPv6 Addressing. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-07]. Dostupné z: https://openthread.io/guides/thread_primer/ipv6_addressing

[18] *IQRF - Technology for wireless* [online]. Jičín: IQRF Tech, 2016 [cit. 2018-05-06]. Dostupné z: https://www.iqrf.org/

[19] J-Link Debug Probes. *SEGGER* [online]. SEGGER Microcontroller, 2018 [cit. 2018-05-08]. Dostupné z: https://www.segger.com/products/debug-probes/j-link/

[20] Joiner fails to join network with external commissioning. *GitHub* [online]. 2018 [cit. 2018-05-09]. Dostupné z: https://github.com/openthread/borderrouter/issues/126

[21] HAO, Feng. J-PAKE: Password-Authenticated Key Exchange by Juggling. *IETF Tools* [online]. Newcastle University, 2017 [cit. 2018-05-08]. Dostupné z: https://tools.ietf.org/html/rfc8236

[22] KINETIS-SDK: Software Development Kit for Kinetis MCUs. *NXP Semiconductors* [online]. NXP Semiconductors, 2018 [cit. 2018-05-08]. Dostupné z: https://www.nxp.com/support/developer-resources/reference-designs/software-development-kit-for-kinetis-mcus:KINETIS-SDK

[23] Libzwaveip - Control Z-Wave devices from your IP network. *GitHub* [online]. GitHub, 2018 [cit. 2018-05-06]. Dostupné z: https://github.com/Z-WavePublic/libzwaveip

[24] Mandatory Security Implementation for All Z-Wave Certified IoT Devices Takes Effect Today. In: *Z-Wave Alliance* [online]. Z-Wave Alliance, 2018 [cit. 2018-05-06]. Dostupné z: https://z-wavealliance.org/mandatory-security-implementation-z-wave-certified-iot-devices-takes-effect-today/

[25] Mapa pokrytí. *IoT portál* [online]. IoT portál, 2018 [cit. 2018-05-06]. Dostupné z: https://www.iot-portal.cz/mapa-pokryti/

[26] Member Companies of the Z-Wave Alliance. *Z-Wave Alliance* [online]. Austin: Z-Wave Alliance, 2018 [cit. 2018-05-06]. Dostupné z: https://z-wavealliance.org/z-wave_alliance_member_companies/

[27] HOLÁSEK, Vít. *Měření spotřeby elektrické energie v domácnosti*. Brno, 2016. Bakalářská práce. Masarykova Univerzita.

[28] KELSEY, Richard. Mesh Link Establishment. *IETF Tools* [online]. Silicon Labs, 2015 [cit. 2018-05-08]. Dostupné z: https://tools.ietf.org/html/draft-ietf-6lo-mesh-link-establishment-00

[29] Mesh Profile. In: *Mesh Networking Specifications* [online]. Bluetooth SIG, 2018 [cit. 2018-05-07]. Dostupné z: https://www.bluetooth.com/specifications/mesh-specifications

[30] MKW41Z/31Z/21Z Data Sheet. *NXP Semiconductors* [online]. NXP Semiconductors, 2018 [cit. 2018-05-08]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MKW41Z512.pdf

[31] Network Co-Processor Support. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/guides/ncp

[32] MILLS, David L. Network Time Protocol (NTP). *IETF Tools* [online]. M/A-COM Linkabit, 1985 [cit. 2018-05-08]. Dostupné z: https://tools.ietf.org/html/rfc958

[33] Node Roles and Types. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-07]. Dostupné z: https://openthread.io/guides/thread_primer/node_roles_and_types

[34] nRF52840. *Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR* [online]. Nordic Semiconductor, 2018 [cit. 2018-05-11]. Dostupné z: https://www.nordicsemi.com/eng/Products/nRF52840

[35] OpenThread. *GitHub* [online]. GitHub, 2018 [cit. 2018-05-08]. Dostupné z: https://github.com/openthread/openthread

[36] *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-07]. Dostupné z: https://openthread.io/

[37] OpenThread Border Router. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/guides/border_router

[38] OpenThread Border Router. *GitHub* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://github.com/openthread/borderrouter

[39] OpenThread Border Router Build and Configuration. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/guides/border_router/build

[40] OpenThread Border Router Web GUI. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/guides/border_router/web_gui

[41] OpenThread C++ API Reference. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/reference

[42] Our Members. *Thread Group* [online]. Thread Group, 2018 [cit. 2018-05-07]. Dostupné z: https://www.threadgroup.org/thread-group#OurMembers

[43] Our Members. *Zigbee Alliance* [online]. Zigbee Alliance, 2017 [cit. 2018-05-06]. Dostupné z: http://www.zigbee.org/zigbeealliance/our-members/

[44] BETTS, Dominic. Overview of the Azure IoT Hub service. *Microsoft Docs* [online]. Redmond: Microsoft, 2018 [cit. 2018-05-06]. Dostupné z: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-iot-hub

[45] PCF8563 - Product data sheet. *NXP Semiconductors* [online]. NXP Semiconductors, 2015 [cit. 2018-05-08]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/PCF8563.pdf

[46] Platforms. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/platforms

[47] Porting OpenThread to New Hardware Platforms. *OpenThread* [online]. Nest Labs, 2018 [cit. 2018-05-08]. Dostupné z: https://openthread.io/guides/porting

[48] Power Line Communications. *Microchip Technology* [online]. Microchip Technology, 2018 [cit. 2018-05-07]. Dostupné z: http://www.microchip.com/design-centers/smart-energy-products/power-line-communications

[49] Quickstart: Create a container registry using the Azure portal. *Microsoft Docs* [online]. Microsoft, 2018 [cit. 2018-05-08]. Dostupné z: https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-portal

[50] GREMBAN, Kelly. Quickstart: Deploy your first IoT Edge module to a Linux or Mac device. *Microsoft Docs* [online].

Microsoft, 2018 [cit. 2018-05-08]. Dostupné z: https://docs.microsoft.com/en-us/azure/iot-edge/quickstart-linux

[51] *Raspberry Pi - Teach, Learn, and Make with Raspberry Pi* [online]. Raspberry Pi Foundation, 2018 [cit. 2018-05-08]. Dostupné z: https://www.raspberrypi.org/

[52] HEDRICK, Charles. *Routing Information Protocol* [online]. Rutgers University, 1988 [cit. 2018-05-08]. Dostupné z: https://tools.ietf.org/html/rfc1058

[53] R41Z. *Rigado* [online]. Rigado, 2016 [cit. 2018-05-08]. Dostupné z: https://www.rigado.com/products/modules/r41z/

[54] R41Z Module for Thread and Bluetooth 4.2 LE. *Rigado* [online]. Rigado, 2016 [cit. 2018-05-08]. Dostupné z: http://go.rigado.com/R41Z-Data-Sheet

[55] *SSL Library mbed TLS / PolarSSL* [online]. ARM Limited, 2016 [cit. 2018-05-08]. Dostupné z: https://tls.mbed.org/

[56] STM32 firmware for STPM32 access and a basic metrology application. *STMicroelectronics* [online]. STMicroelectronics, 2018 [cit. 2018-05-08]. Dostupné z: http://www.st.com/content/st_com/en/products/embedded-software/evaluation-tool-software/stsw-stpm002.html

[57] SHELBY, Zach, Klaus HARTKE a Carsten BORMANN. The Constrained Application Protocol (CoAP). *IETF Tools* [online]. Universitaet Bremen: IETF Tools Team, 2014 [cit. 2018-05-07]. Dostupné z: https://tools.ietf.org/html/rfc7252

[58] The 3-Clause BSD License. *Open Source Initiative* [online]. Open Source Initiative, 2018 [cit. 2018-05-08]. Dostupné z: https://opensource.org/licenses/BSD-3-Clause

[59] Thread 1.1 Commissioning App. In: *Google Play* [online]. Google Commerce, 2017 [cit. 2018-05-08]. Dostupné z: https://play.google.com/store/apps/details?id=org.threadgroup.commissioner&hl=cs

[60] Understand how IoT Edge modules can be used, configured, and reused. *Microsoft Docs* [online]. Microsoft, 2018 [cit. 2018-05-09]. Dostupné z: https://docs.microsoft.com/en-us/azure/iot-edge/module-composition

[61] POSTEL, Jon. User Datagram Protocol. *IETF Tools* [online]. IETF Tools Team, 1980 [cit. 2018-05-07]. Dostupné z: https://tools.ietf.org/html/rfc768

[62] What is Azure IoT Edge. *Microsoft Docs* [online]. Microsoft, 2018 [cit. 2018-05-08]. Dostupné z: https://docs.microsoft.com/en-us/azure/iot-edge/how-iot-edge-works

[63] What is Thread. *Thread Group* [online]. Thread Group, 2018 [cit. 2018-05-07]. Dostupné z: https://www.threadgroup.org/What-is-Thread

[64] STAUDEK, Jan. *Wireless Sensor Networks, ZigBee* [online]. 2018 [cit. 2018-05-07]. Dostupné z: https://www.fi.muni.cz/usr/staudek/vyuka/PA151/09_wpan_zb.pdf. Masarykova Univerzita.

[65] Zigbee Alliance and Thread Group Successfully Demonstrate Products Running Zigbee-s Universal Language for Smart Devices on Thread Networks. *Zigbee Alliance* [online]. Zigbee Alliance, 2016 [cit. 2018-05-07]. Dostupné z: http://www.zigbee.org/zigbee-alliance-and-thread-group-successfully-demonstrate-products-running-zigbees-universal-language-for-smart-devices-on-thread-networks/

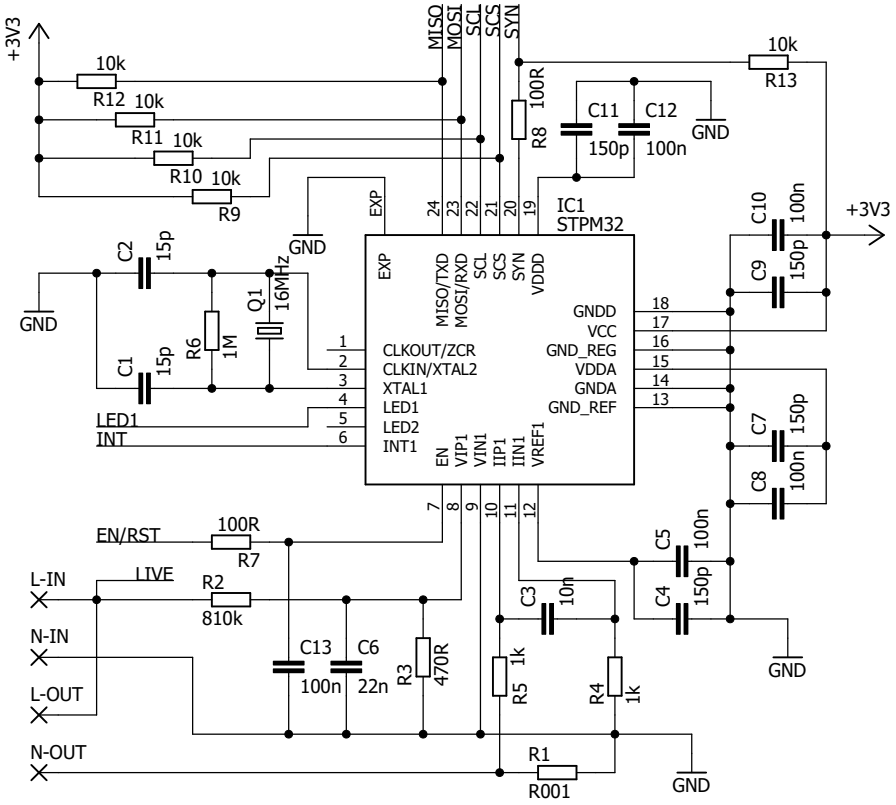
[66] Zigbee 3.0. *Zigbee Alliance* [online]. Zigbee Alliance, 2018 [cit. 2018-05-07]. Dostupné z: http://www.zigbee.org/zigbee-for-developers/zigbee-3-0/

[67] Z-Wave Networking Basics. In: *Z-Wave the Public Standard* [online]. Austin: Silicon Laboratories, 2018 [cit. 2018-05-06]. Dostupné z: http://zwavepublic.com/sites/default/files/APL13031-2%20-%20Z-Wave%20Networking%20Basics.pdf

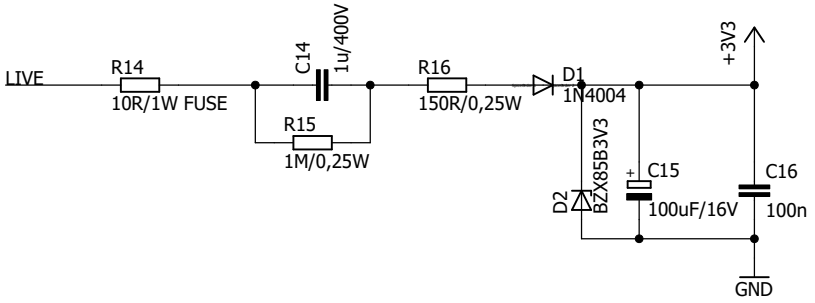
# A Seznam elektronických příloh

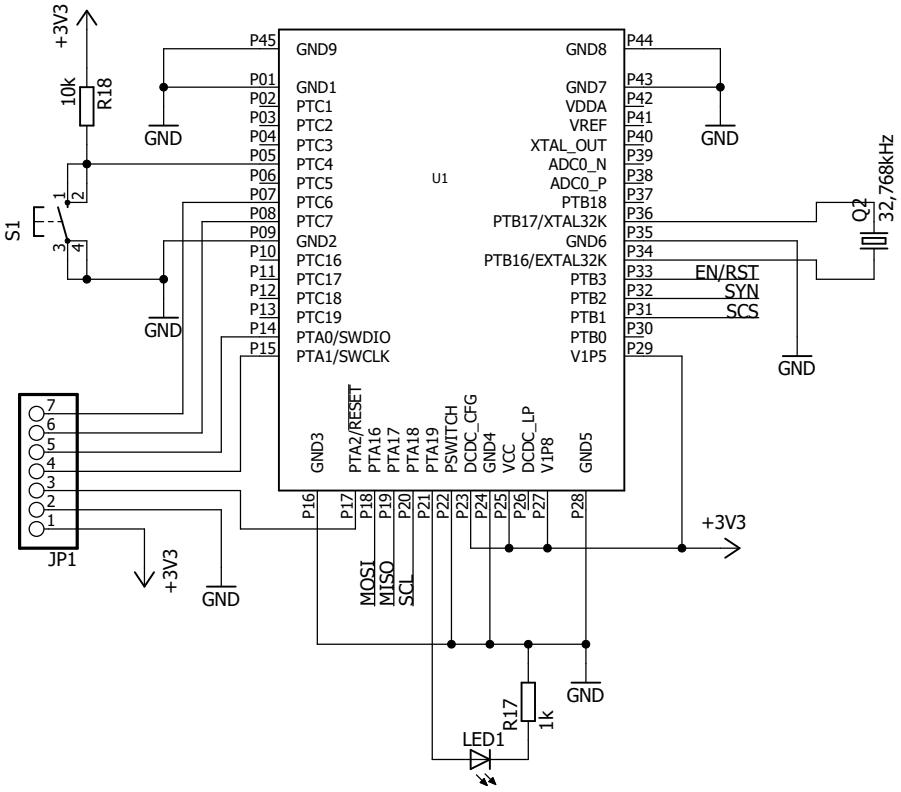
- Adresář `iot_edge_modules` – Obsahuje zdrojové kódy nezbytné k sestavení a nasazení modulu pro zařízení gateway v rámci prostředí Azure IoT Edge.
- Adresář `docs` – Obsahuje výkresy se schématy zapojení zařízení zásuvky a NCP rozšíření pro počítač CHIP. Obsahuje také projekt vytvořený v software Eagle.
- Adresář `smart_socket` – Obsahuje zdrojové soubory nezbytné pro sestavení firmware zásuvky.

# B Schéma zapojení měřicího zařízení



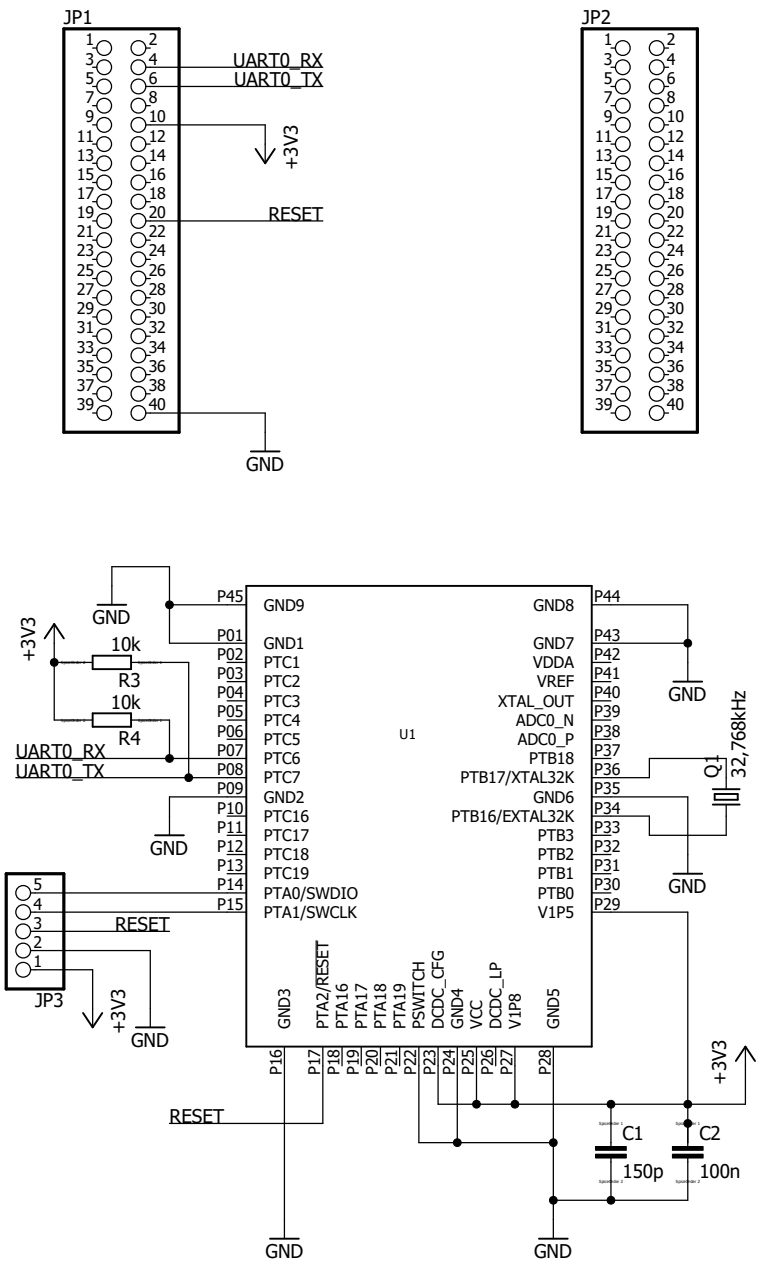

| | | |
|---|---|---|
| | | |
| TITLE: SMART SOCKET | | |
| Document Number:<br>Author: Vít Holásek | | REV: |
| Date: 14.05.2018 21:31 | Sheet: 1/2 | |




| | |
|---|---|
| TITLE: SMART SOCKET | |
| Document Number:<br>Author: Vít Holásek | REV: |
| Date: 14.05.2018 21:31 | Sheet: 2/2 |

# C Schéma zapojení NCP rozšiřující desky




| | |
|---|---|
| TITLE: CHIP SHIELD | |
| Document Number:<br>Author: Vít Holásek | REV: |
| Date: 16.05.2018 22:05 | Sheet: 1/1 |

# D Ukázka použití knihovny ovladače pro STPM32

```
#include <stdio.h>

// Metrology includes
#include "metrology_platform.h"
#include "metroTask.h"

int main(void) {
  // Initialize platform dependent modules
  metrology_platform_init();
  // Initialize metrology application and STPM3x chip
  if (METRO_Init() != METRO_OK) {
    printf("Failed to init metrology device");
    return 1;
  }
  // Configure calibrations
  Metro_Set_V_Calibration(CHANNEL_1, 0x800);
  Metro_Set_C_Calibration(CHANNEL_1, 0x800);
  // Set calculation factors (example for SmartPlug)
  uint32_t power_factor = 9800246;
  uint32_t energy_factor = 11422;
  uint32_t voltage_factor = 116274;
  uint32_t current_factor = 8428;
  Metro_Set_Hardware_Factors(CHANNEL_1, power_factor,
     energy_factor, voltage_factor, current_factor);

  while (1) {
    metrology_platform_wait_microseconds(3000000U);
    // Latch STPM3x registers data
    METRO_Latch_Measures();
    // Read STPM3x registers data and update measurement
       calculations
    METRO_Update_Measures();
    // Print results
    printf("RMS current: %d mA\r\nRMS voltage: %d mV\r\n"
       , metroData.rmscurrent,
      metroData.rmsvoltage);
  return 0;
}
```