Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Tvorba knihovny pro mikrokontroler pro mesh IoT sítě

Bc. Jakub Prikner

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jakub Prikner

Systémové inženýrství a informatika
Informatika

Název práce

Tvorba knihovny pro mikrokontroler pro mesh IoT sítě

Název anglicky

Creation of a library for a microcontroller for mesh IoT networks

---

Cíle práce

Tématem diplomové práce je bezdrátový přenos dat mezi mikrokontrolery. Hlavním cílem práce je vývoj knihovny pro komunikaci mezi bezdrátovým modulem IQRF a platformou Arduino.

Dílčími cíli jsou:
1. analýza aktuálních možností v oblasti bezdrátové komunikace mikrokontrolerů v síti typu mesh, analýza obecných praktik v rámci tvorby softwarových knihoven,
2. vývoj nové knihovny pro usnadnění komunikace mezi platformou Arduino a bezdrátovým modulem, využívajícím technologii IQRF mesh,
3. implementace do stávajícího řešení,
4. návrh rozvoje knihovny o další funkce.

Metodika

Metodika v teoretické části diplomové práce bude založena na analýze a rešerši odborných zdrojů. Praktická část bude obsahovat návrh požadovaných funkcí a jejich implementaci v podobě nové knihovny. Ta bude pro platformu Arduino napsána v jazyce C a bude představovat výstup praktické části. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry práce.

Doporučený rozsah práce

70

Klíčová slova

Mikrokontroler, knihovna, Arduino, IQRF, mesh síť, internet věcí, IoT.

Doporučené zdroje informací

AL-FUQAHA, Ala. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials: Volume 17, Issue 4, [2015]. ISSN 2373-745X.

BLUM, Jeremy. Exploring Arduino: tools and techniques for engineering wizardry. Indianapolis, IN: Wiley, [2013]. ISBN 978-1-118-54936-0.

KUCHTA, Radek, Radimír VRBA, Vladislav ŠULC. Smart Wireless Communication Platform IQRF, Mobile and Wireless Communications Network Layer and Circuit Level Design, Salma Ait Fares and Fumiyuki Adachi (Ed.): InTech, [2010]. ISBN 978-953-307-042-1. Dostupné z: http://www.intechopen.com/books/mobile-and-wireless-communications-network-layer-and-circuit-level-design/smart-wireless-communication-platform-iqrf

WHEELER, David A. Program Library HOWTO: [2003]. Dostupné z: http://www.tldp.org/HOWTO/Program-Library-HOWTO

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 11. 10. 2019

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 14. 10. 2019

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 08. 02. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Tvorba knihovny pro mikrokontroler pro mesh IoT sítě" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 3. 2021 _____

Bc. Jakub Prikner

Poděkování

Rád bych touto cestou poděkoval své rodině za výraznou podporu po celou dobu tvorby této práce. Poděkování patří také vedoucímu mé práce Ing. Michalu Stočesovi, Ph.D., za veškerou pomoc, poskytnuté rady a motivaci.

Dále bych rád poděkoval katedře informačních technologií PEF ČZU za zapůjčení dvou kusů transceiveru TR-72DA, dvou kusů adaptéru IQRF-KON-RASP-01 a dvou kusů mikrokontroleru Arduino Uno. Dále děkuji společnosti JoTio Tech za zapůjčení dvou kusů bateriového vývojového kitu IQRF-DK-EVAL-04a, jednoho kusu programátoru IQRF-CK-USB-04a, jednoho kusu vývojového relé kitu IQRF-DDC-RE-01 a jednoho kusu mikrokontroleru Arduino Mega 2560.

Tvorba knihovny pro mikrokontroler pro mesh IoT sítě

Abstrakt

Diplomová práce se zabývá bezdrátovou komunikací mezi prvky pro internet věcí a volně tak navazuje na autorovu bakalářskou práci. Cílem diplomové práce bylo vytvořit softwarovou knihovnu, usnadňující použití mikrokontrolerů Arduino společně s bezdrátovými moduly IQRF. V teoretické části jsou popsány možnosti bezdrátové komunikace pro IoT zařízení. Zároveň je zde detailněji rozebrána topologie typu mesh a technologie IQRF. Literární rešerše je zaměřena také na tvorbu softwarových knihoven. Konkrétní knihovna je následně vytvořena v praktické části práce. V závěru jsou zformulovány návrhy a doporučení na další vývoj a vylepšení této knihovny komunitou po jejím zveřejnění.

Klíčová slova: Mikrokontroler, knihovna, Arduino, IQRF, mesh síť, internet věcí, IoT.

Creation of a library for a microcontroller for mesh IoT networks

Abstract

This diploma thesis deals with the wireless communication between the Internet of Things elements and freely follows the author's bachelor thesis. The main goal of this thesis was to create a software library facilitating use of Arduino microcontrollers together with the IQRF wireless modules. In the theoretical section the various options for wireless communication for IoT devices are described. The mesh topology and the IQRF technology are also discussed here. The literary research is focused also on the creation of the software libraries. A particular library is then developed in the practical section of the thesis. At the end of the thesis some suggestions and recommendations are formulated, purposed for future development and improvement of the library.

Keywords: Microcontroller, library, Arduino, IQRF, mesh, network, internet of things, IoT.

Obsah

1 Úvod

Jako téma mé diplomové práce jsem si zvolil tvorbu knihovny pro mikrokontroler Arduino, která usnadní jeho použití společně s bezdrátovými moduly IQRF. K této volbě mne vedla má dlouhodobá záliba: vývoj vlastního řešení inteligentní domácnosti s využitím běžně dostupných prvků – mikrokontrolerů Arduino, jednodeskových minipočítačů Raspberry Pi, různých senzorů a akčních členů. Ve své bakalářské práci jsem se zabýval všemi fázemi vývoje automatizace ovládání interiérových žaluzií s využitím platforem Arduino a Raspberry Pi, od návrhu takového systému až po implementaci do funkční makety okna pro účel prezentace výsledku práce. Pro celkovou komplexnost práce však již nebyl prostor pro dořešení bezdrátové komunikace mezi řídicím prvkem (Raspberry Pi) a jednotlivými okny (Arduino + bezdrátový modul). Komunikace zde probíhala bezdrátově v režimu klient – server pouze s jedním klientským prvkem.

Po zpětném zhodnocení jsem dospěl k závěru, že zvolený bezdrátový 2,4GHz modul NRF24L01+ je pro komunikaci ve víceprvkové konfiguraci nevhodný, neboť nemá softwarově vyřešen režim mesh sítě. Během magisterského studia na ČZU jsem objevil existenci bezdrátových transceiverů české společnosti IQRF Tech, využívajících pro komunikaci jejich vlastní technologii IQMESH.

Bezprostředně poté, co jsem se na jedné zájmové školní přednášce od IQRF Alliance s IQRF technologií seznámil, jsem byl jejími možnostmi nadšen. Z hlediska komunikačních vlastností i energetické náročnosti při provozu se transceivery IQRF jevily jako plně vyhovující. Začal jsem zjišťovat, zda a jakým způsobem by bylo možné tyto prvky zakomponovat do stávajícího systému řízení žaluzií. Jako jediný vážnější nedostatek se ukázala být faktická neexistence uceleného obslužného rozhraní – tzv. knihovny – pro interakci s periferiemi mikrokontrolerů platformy Arduino. Po několika konzultacích se zakladateli společnosti JoTio Tech, s.r.o., kteří v té době s IQRF aktivně pracovali, jsem se rozhodl chybějící rozhraní sám vytvořit s tím, že se toto zároveň stane tématem mé diplomové práce. S tímto souhlasil také vedoucí mé diplomové práce, bylo tedy vytvořeno zadání práce, které bylo následně i formálně schváleno.

Po dokončení a publikaci práce bude knihovna volně dostupná komunitě kolem platformy Arduino, a to prostřednictvím vlastního repozitáře na platformě GitHub, popřípadě GitLab. Počítám s tím, že komunita rozšíří základní verzi knihovny o další potřebné funkce a vylepšení.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této diplomové práce je vytvořit softwarovou knihovnu pro komunikaci mezi mikrokontrolerem Arduino a bezdrátovým modulem IQRF.

Dílčími cíli jsou:
1. analýza aktuálních možností v oblasti bezdrátové komunikace mikrokontrolerů v síti typu mesh, analýza obecných praktik v rámci tvorby softwarových knihoven,
2. vývoj nové knihovny pro usnadnění komunikace mezi platformou Arduino a bezdrátovým modulem, využívajícím technologii IQRF mesh,
3. implementace do stávajícího řešení,
4. návrh rozvoje knihovny o další funkce.

2.2 Metodika

Metodika v teoretické části diplomové práce bude založena na analýze a rešerši odborných zdrojů. Praktická část bude obsahovat návrh požadovaných funkcí a jejich implementaci v podobě nové knihovny. Ta bude pro platformu Arduino napsána v jazyce C a bude představovat výstup praktické části. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry práce.

3 Teoretická východiska

Nejprve je uveden svět internetu věcí a jeho principů, včetně příkladů z praxe. Dále je zde pojednáno o možnostech komunikace mezi zařízeními, o technologii IQRF, platformě Arduino a tvorbě knihoven v jazyce C++ pro tuto platformu.

3.1 Internet věcí

Internet věcí neboli zkráceně IoT (z anglického Internet of Things) je v dnešní době již velmi široký pojem, a to nejen díky neustále rostoucímu počtu různých zařízení, které lze za součást IoT považovat. Dále budou uvedeny některé stěžejní oblasti, kde je možné IoT technologie aplikovat a kde se s nimi lze setkat, společně s konkrétními příklady takových aplikací.

3.1.1 Prvky IoT

Poměrně výstižně IoT popisuje obrázek Součásti IoT. Bez těchto klíčových prvků či disciplín, které jsou součástí světa IoT, by převážná většina projektů v této oblasti nemohla být realizována.

Obrázek 1 Součásti IoT [1]

Identifikace (Identification)

Identifikace je pro IoT zásadní kvůli zajištění párování služeb s jejich požadavky. Existuje mnoho identifikačních metod, jako například elektronické produkt kódy (EPC). Při adresování v IoT je klíčové rozlišovat mezi ID objektu a jeho adresou. ID objektu odpovídá jeho názvu, například „T1“ pro konkrétní teplotní senzor a adresa objektu odpovídá jeho adrese v komunikační síti. [1]

Měření (Sensing)

Měření v IoT znamená shromažďování dat z odpovídajících objektů v rámci sítě a jejich odesílání do datového skladu, databáze či cloudu. IoT senzory mohou být různé druhy chytrých měřicích zařízení, aktuátory, či nositelné senzory. [1]

Komunikace (Communication)

Technologie komunikace v IoT propojují heterogenní objekty v jeden celek, aby tak mohly zajišťovat specifické služby chytrých řešení. Příklady komunikačních protokolů, ve světě IoT používaných, mohou být WiFi, Bluetooth, Z-wave, LoRaWAN či LTE-Advanced. Využívají se také různé specifické komunikační technologie jako RFID, Near Field Communication (NFC) a ultra-wide bandwidth (UWB). [1]

Zpracování (Computation)

Výpočetní jednotky (mikrokontrolery, mikroprocesory, SOC – systémy na čipu, FPGA) a softwarové aplikace reprezentují výpočetní možnosti IoT. Pro řízení a běh IoT aplikací bylo vyvinuto mnoho hardwarových platforem, z nichž nejznámější jsou Arduino, Raspberry Pi, Intel Galileo či BeagleBone. Toto doplňuje spousta softwarových platforem poskytujících IoT funkcionality. Vedle těchto platforem hrají klíčovou roli také operační systémy, které musí běžet po celou dobu provozu IoT zařízení. Cloudové platformy tvoří další významnou výpočetní oblast pro IoT. Tyto platformy poskytují zázemí chytrým zařízením, aby mohla odesílat data do cloudu, dále aby zde mohla být zpracovávána *Big Data* v reálném čase a také, aby koncoví uživatelé měli možnost takto získané informace zužitkovat. Komerčních i bezplatných cloud platforem pro správu IoT služeb existuje celá řada. [1]

Služby (Services)

Do IoT služeb patří služby pro chytrou domácnost i chytré budovy, které propojují systémy automatizace budov do internetu. Řadí se sem i inteligentní dopravní systémy, průmyslová automatizace, chytré aplikace pro zdravotní péči, chytré rozvodné elektrické sítě či koncepty chytrých měst. Rozdělení IoT služeb ilustruje Obrázek 2. [1]

Sémantika (Semantics)

Sémantika v IoT značí schopnost různých zařízení šikovně vytěžovat znalosti pro poskytování požadovaných služeb. Vytěžování znalostí zahrnuje získávání a využívání prostředků, rozpoznávání a analýzu dat takovým způsobem, aby na základě toho bylo provedeno správné rozhodnutí a poskytnuta konkrétní služba. Sémantika představuje mozek

celého IoT, neboť dokáže správně cílit odesílané požadavky. Tuto vlastnost podporují technologie jako *Resource Description Framework* (RDF) či *Web Ontology Language* (OWL). V roce 2011 byl W3C konsorciem přijat formát *Efficient XML Interchange* (EXI) jako doporučení. Formát EXI je pro IoT důležitý kvůli své vlastnosti optimalizovat XML aplikace pro prostředí s omezenými zdroji. EXI převádí XML zprávy do binární reprezentace a tím pomáhá snižovat nároky na tok dat a šetřit úložnou kapacitu. [1]

Obrázek 2 Hlavní oblasti uplatnění IoT technologií [1]

3.1.2 IoT a big data

Podobně jako IoT, rovněž i pojem „velká data“ – tzv. big data – tvoří v rámci informačních technologií dnes již samostatnou vědní disciplínu.

Základní charakteristikou je malé množství dat o velkých objemech nebo naopak velké množství drobných dat. Tato data také bývají velmi často nestrukturovaná – ať už se jedná o data popisující nějaké jevy či chování, data ze sociálních sítí, sběr vědeckých dat a veličin, či měření ze senzorických sítí různých druhů. [2]

Big Data obecně charakterizují tzv. „4V“:

Volume (objem) – Objem dat roste exponenciálně, nikoliv lineárně. I velké objemy malých dat mohou vyústit v Big Data.

Variety (komplexnost) – různé formáty, typy a struktury. Od částečně strukturovaných XML dokumentů až po nestrukturovaná multimédia. Dochází zde k směšování strukturovaných a nestrukturovaných dat.
Velocity (rychlost) – Data jsou generována s vysokou rychlostí, je proto je nezbytné je také rychle zpracovávat. Dávkové zpracování dat nahrazují data v proudech (streaming).
Veracity (rozmanitost, neurčitost) – Nejistota vzhledem k nekonzistenci, neúplnosti, prodlevám, či aproximacím. [2]

3.1.3 Oblasti aplikace IoT technologií

3.1.3.1 Chytrá města

Jedním z nových trendů v IoT je snaha o vznik a rozvoj „chytrých měst“ – tzv. smart cities. Součástí chytrého města mohou být například následující prvky.

Chytrá parkoviště

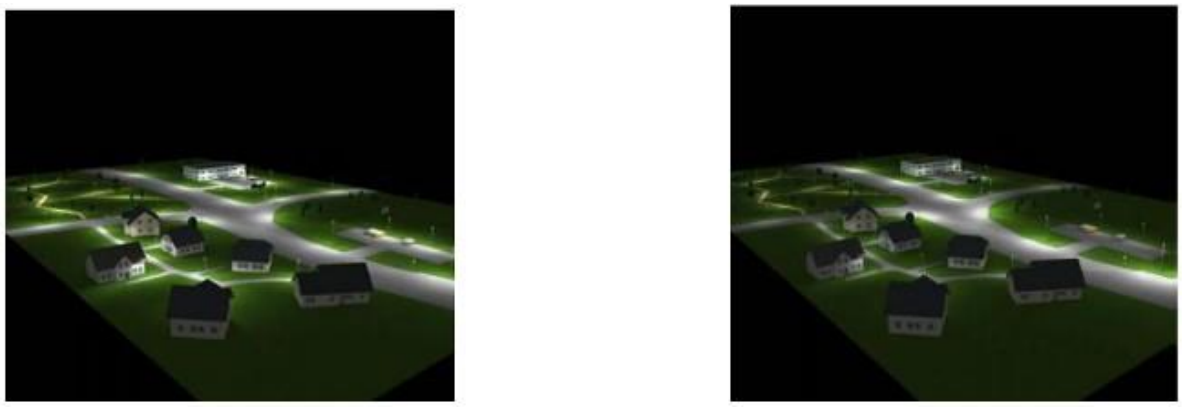
Každé parkovací místo má svůj senzor, který hlídá, zda je místo obsazené či volné. Aktuální stav jednotlivých parkovacích míst může být pro snazší orientaci řidiče barevně signalizován, kdy červená barva značí obsazené místo a zelená barva volné. Senzory dále mohou reportovat svůj stav řídicímu systému. Díky tomu lze řidičům již při vjezdu na informační tabuli zobrazit celkový počet volných a obsazených míst na parkovišti. Tyto údaje mohou dále využít i chytré navigace v automobilech při plánování tras.

Zastávky MHD a lavičky

Tato místa jsou typicky vybavena WiFi konektivitou a možností dobít si zde mobilní telefon. Především chytré zastávky mohou pro občany zastávat i funkci informační, a to například prostřednictvím interaktivního dotykového displeje či jiného rozhraní v kombinaci s reproduktory. Díky internetovému připojení je možné z centrálního místa operativně měnit zobrazované informace. Příkladem může být zobrazení aktuálních informací v případě mimořádné situace, pátrání po osobách pohřešovaných či hledaných, nebo různá sdělení a výzvy města směrem k občanům. Elektrickou energii ke svému provozu čerpají buď z elektrické sítě, nebo z vestavěných solárních panelů.

Lampy veřejného osvětlení

Konvenční soustavy pouličních lamp neumožňují jakoukoli interakci, vyjma zapínání a vypínání celých větví. Současně správě veřejného osvětlení chybí informace o stavu jednotlivých lamp, o případné poruše se dozví většinou až na základě hlášení od občanů. [3] V České republice již bylo realizováno několik pilotních projektů zaměřených na konstrukci a provoz inteligentních lamp veřejného osvětlení. S využitím bezdrátové mesh IoT technologie IQRF jičínské společnosti Microrisc získaly běžné sodíkové lampy možnost vzdáleného zhasínání či rozsvěcení, a to jednotlivě. Každá lampa má totiž v mesh síti svoji unikátní adresu, pomocí které s ní lze komunikovat. V případě LED lamp lze tyto navíc i stmívat, což vede k další úspoře provozních nákladů na soustavu. Stmívání lamp veřejného osvětlení je v souladu s českou legislativou, upravuje jej norma ČSN CEN/TR 13201-1: „*...osvětlení lze snížit až o 75% jmenovité hladiny osvětlení. Snížení osvětlení o více než 50% jmenovité hladiny osvětlení musí být podloženo analýzou změn intenzity provozu na uvažované pozemní komunikaci a schváleno příslušným silničním správním úřadem.*“ Legálně tedy lze stmívat až o 50 % bez nutnosti řešit právní výjimky pro jednotlivé instalace či provádět jiné legislativní kroky. To ilustruje Obrázek 3. [4]



Obrázek 3 Srovnání intenzity svitu veřejného osvětlení – plná intenzita v celé aglomeraci (bez regulace – vlevo), resp. plná intenzita na veřejných komunikacích, tlumená v obytné zástavbě (regulace stmíváním – vpravo) [4]

Řízení lamp za pomoci IQRF představuje ještě další výhodu v podobě možnosti monitorovat aktuální stav jednotlivých zařízení. Pokud tedy dojde k poruše na konkrétní lampě, není nutné spoléhat na nahlášení závady veřejností či se ji snažit aktivně lokalizovat. V kombinaci s alerting mechanismy může monitorovací systém problém ihned automaticky nahlásit zodpovědným osobám a oprava se tak může uskutečnit velmi rychle po vzniku závady. Samozřejmou součástí monitoringu je také sledování spotřeby elektrické energie

jednotlivých světel i celé soustavy, což umožňuje provádět další analýzy a také maximalizovat ekonomickou efektivitu provozu. [4]

3.1.3.2 **Inteligentní budovy**

Zde jsou možnosti aplikace IoT prvků velmi široké. Součástí inteligentních budov může být systém automaticky řízené ventilace a topení, kdy si budova sama hlídá teplotu uvnitř. K její regulaci systém otevírá a zavírá okna, spouští či vytahuje žaluzie a rolety, zapíná či vypíná topení a klimatizaci. Při regulaci vytápění lze zohlednit i počet osob, nacházejících se v jednotlivých místnostech – zjištění takové informace je realizovatelné s využitím speciálních senzorů. [5]

V případě zasedacích místností se jejich aktuální obsazenost může přímo promítat do interního informačního systému pro zaměstnance, hledající například volné prostory pro schůzku či jen místo, kde budou mít více klidu na práci. V tomto případě je obsazenost vyhodnocována prostřednictvím senzorů, přítomných v místnostech. Informační systém tak dokáže navigovat pracovníka do nejbližší neobsazené místnosti. Toto řešení v současné době ve své budově v Praze využívá například společnost Microsoft. [6]

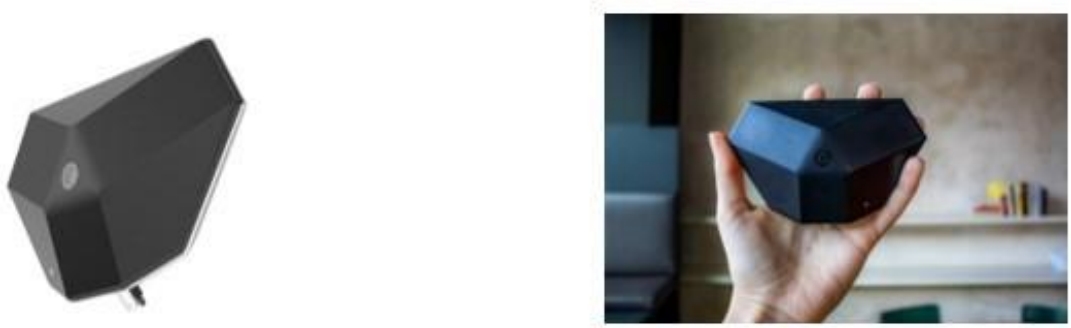
Způsobů, jak monitorovat obsazenost zasedacích místností a společných prostor, je mnoho, a odvíjí se především od použité technologie detekce osob. Mezi méně přesná a flexibilní řešení se řadí tlakové snímače v židlích či optické závory při vstupech do těchto prostor.

Tlakové snímače poskytují pouze informaci o tom, kolik osob v daný okamžik sedí na svých místech. Z toho lze při znalosti celkového počtu míst k sezení v dané místnosti vypočítat její obsazenost v procentech či zbývající volná místa, a tyto údaje následně zobrazit na informačním panelu u vstupu do místnosti. Zjistit celkový počet osob v daném prostoru, včetně stojících či pohybujících se, tímto způsobem však není možné.

Senzory v podobě optických závor dnes již umí nejen počítat průchody, ale i rozpoznat směr, kterým se daný objekt pohybuje. Umožňují tak poskytnout poměrně přesnou informaci o aktuálním počtu osob uvnitř prostoru, ale i o celkovém počtu průchodů jedním i druhým směrem. Příkladem může být LoRaWAN People Counter nizozemské společnosti IMBuildings. [7]

Jak již jeho název napovídá, toto zařízení využívá pro komunikaci bezdrátovou IoT síť LoRaWAN (o této technologii bude pojednáno dále). Detekci směru pohybu umožňuje IR technologie s paprsky orientovanými v horizontální rovině.

Finančně nejnákladnější možností jsou kamery s rozpoznáváním osob v obrazu za pomoci neuronových sítí natrénovaných k tomuto účelu, či termální kamery detekující osoby podle vyzářeného tepla. Bonusem u obou těchto metod snímání obrazu je možnost vytvářet tzv. heat mapy – pomocí trasování pohybujících se objektů ve snímané scéně lze vizuálně znázornit místa, kudy lidé nejčastěji chodí a kde se nejvíce zdržují. Získaná data lze dobře využít zejména v maloobchodním prodeji pro analýzy prodejnosti zboží v korelaci s tím, jak se zákazníci po prodejní ploše pohybují. S ohledem na v současné době probíhající globální pandemii koronaviru naleznou tyto kamerové technologie své uplatnění i v boji proti šíření nákazy. Monitorováním počtu přítomných osob a autonomním hlídáním vůči nastavené maximální kapacitě mohou pomoci v dodržování zavedených hygienických opatření, například nevpuštěním dalších osob nad tento limit. Jako konkrétní zařízení lze zmínit VIVIDI české startupové společnosti Iterait (viz Obrázek 4). [8]



Obrázek 4 Zařízení VIVIDI Gen. 2 [9]

S testováním tohoto senzoru má autor osobní praktické zkušenosti. Za předpokladu dostatečné úrovně osvětlení ve snímaném prostoru disponuje zařízení relativně vysokou přesností a spolehlivostí detekce, a to včetně zjištění pohlaví a momentální nálady osob v zorném poli. Současně s tímto lze jistě ocenit i důraz, jaký společnost Iterait klade na dodržení souladu s GDPR. Přestože VIVIDI pracuje na principu analýzy obrazu z kamery, která je jeho součástí, vše se děje interně uvnitř senzoru a na server se odesílají pouze anonymizovaná data o přítomných objektech. Lokální zpracování obrazu zajišťuje Edge AI TensorFlow Processing Unit. Zařízení komunikuje přes síť WiFi či mobilní LTE. Obrázek 5 zobrazuje místnost z pohledu VIVIDI. Vpravo je pak část informačního panelu webového rozhraní. Zde je kromě aktuálního počtu osob uvnitř místnosti a nastavené kapacity patrná i již zmíněná heat mapping funkce.

Obrázek 5 Zařízení VIVIDI, informační panel webové aplikace [8][autor]

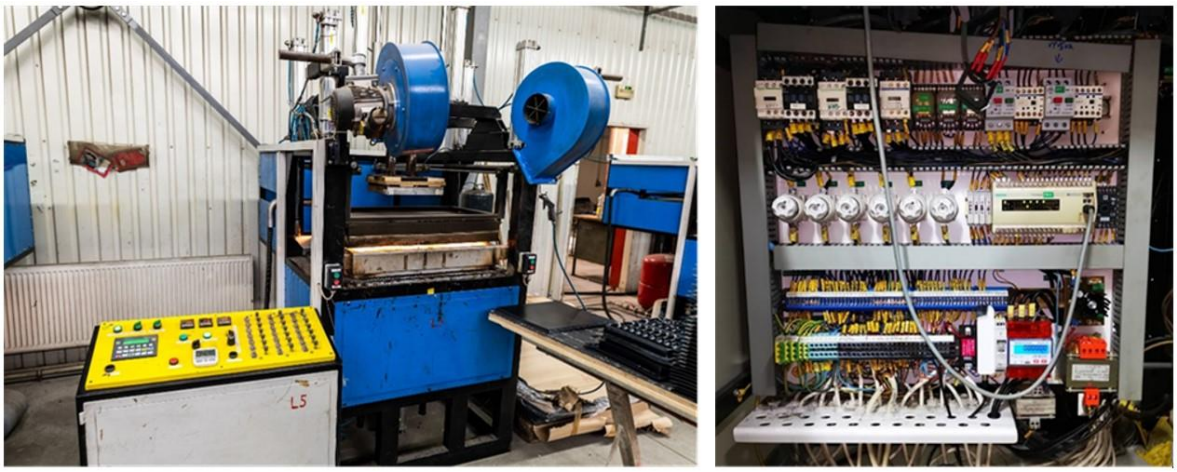
3.1.3.3 IoT v průmyslu

Tzv. Průmysl 4.0 je dnes již ustálený pojem a IoT technologie jsou jeho přirozenou součástí. Z celé řady možností uplatnění IoT v průmyslu zde bude zmíněn jeden konkrétní příklad.

Monitoring telemetrie výrobních strojů

Výrobní linky v automobilkách, výrobních závodech. Stroje a linky na zpracování oceli, plastů či textilu. Plně automatizované provozy i takové provozy, ve kterých je lidská obsluha součástí výrobního procesu. Zde všude může implementace i poměrně jednoduché a levné IoT technologie pomoci k výraznému zlepšení efektivity výroby, snížení spotřeby energií na jeden výrobní cyklus, ale také predikovat blížící se selhání důležitých komponent.

Zde lze uvést příklad z praxe, konkrétně ze zavedené společnosti, disponující více než 25letou praxí v oblasti výroby plastových výlisků vakuovým tvářením. Vedení společnosti však postrádalo jakýkoli přehled o efektivitě vlastní výroby. Dlouhodobě vysoké náklady na elektřinu tak byly impulzem ke změně v podobě nasazení IoT řešení.



Obrázek 6 Stroj na zpracování plastových výlisků metodou vakuového tváření [10][autor]

Neboť firma své stroje na termoforming plastů (viz Obrázek 6 vlevo) sama konstruovala, existovala k nim schémata zapojení. To značně usnadnilo vestavbu IoT prvků. Byl instalován třífázový digitální elektroměr (viz Obrázek 6 vpravo dole) společně se senzorem pro sledování počtu pracovních cyklů. Senzor byl napojen na výstup z PLC automatu. Komunikace senzoru s elektroměrem probíhá prostřednictvím průmyslové datové sběrnice M-Bus. Pro maximální usnadnění vestavby byl zvolen bezdrátový způsob přenosu nasbíraných telemetrických dat ze stroje. V zázemí výrobní haly tak byla dále nainstalována brána pro IoT síť LoRaWAN, pomocí níž přenos dat probíhá. Odtud data proudí již standardní cestou přes internet na server do time series databáze InfluxDB. Následně jsou zobrazována v nástroji pro vizualizaci dat Grafana, kde je pro tento projekt vytvořen samostatný informační panel (Obrázek 7).

Majitel firmy má nyní exaktní přehled jak o celkové spotřebě elektrické energie, tak i o počtech pracovních cyklů v průběhu dne, a díky detailnímu zachycení spotřeby v čase také o energetické náročnosti vztažené na výrobu jednotlivých kusů. Na grafech (Obrázek 7) níže jsou kromě spotřeby energie pokryté produkcí patrné i časové úseky, kdy stroj sice spotřebovává energii, avšak nevyrábí. Tento nežádoucí stav bývá typicky způsoben zaměstnancem, který od obsluhy stroje na delší dobu odejde, přitom však nechá puštěnou vývěvu.

Obrázek 7 Infopanel Grafana pro monitoring telemetrie výroby [10][autor]

3.1.3.4 IoT ve zdravotnictví

Také v oblasti zdravotnictví existuje mnoho příležitostí k uplatnění IoT prvků. Na přelomu roku 2020 a 2021, s příchodem prvních zásilek vakcín proti novému koronaviru, se zvýšila poptávka po způsobech, jak spolehlivě monitorovat dodržování skladovacích a přepravních podmínek některých typů těchto vakcín. V případě té od americké společnosti Pfizer jsou podmínky obzvláště náročné, předepsaná skladovací teplota je -70 °C. Toto již vyžaduje speciální mrazicí boxy a další vybavení. Ke sledování aktuální teploty v boxu již nestačí

běžné teplotní senzory v IoT světě jinak hojně využívané, je zde nezbytné použít speciální termočlánek s odpovídajícím rozsahem a přesností měření.
V nemocnicích lze pro zabezpečení některých druhů movitého majetku (lůžka, kolečková křesla a další) před zcizením či neoprávněnou manipulací opatřit tyto objekty speciálními tagy, pracujícími na principu Bluetooth Low Energy. Stejně tak lze těmito tagy v podobě návštěvnických kartiček vybavit personál nemocnice, pacienty i jejich doprovod. Díky technologii snímačů rozmístěných v chodbách, na pokojích a v ostatních prostorech nemocnice tak lze udržovat přehled o jejich pohybu po areálu. Toto lze spojit i s přístupovými systémy a například pacientům, mířícím na ortopedii zamezit vstup do ostatních částí nemocnice – především tam, kde by jim mohlo hrozit zvýšené riziko nákazy.

3.1.3.5 **IoT v domácnostech**

Do povědomí široké veřejnosti se IoT zařízení a technologie dostávají především díky stále širším možnostem využití v domácnostech. Výrazně tomu dopomáhají společnosti jako Philips či IKEA se svými programy chytrých LED osvětlení Philips Hue [11] či IKEA s řadou produktů Home smart [12], ale například i chytré zásuvky Sonoff s měřením spotřeby elektřiny a ovládatelné pomocí WiFi [13], dále chytré robotické vysavače a další domácí pomocníci. Nedávno se připojil i německý Lidl s vlastní nabídkou smart home produktů [14]. Venkovní předokenní rolety, ale i interiérové žaluzie již několik let umožňuje svým vlastním systémem řídit firma Somfy [15]. Všechny tyto prvky pomáhá ovládat Google se svoji řadou hlasových asistentů Google Nest [16], dále je zde i široká škála Bluetooth reproduktorů různých výrobců s vestavěnými asistenty Amazon Alexa či Google Assistant.

3.2 **Komunikační rozhraní**

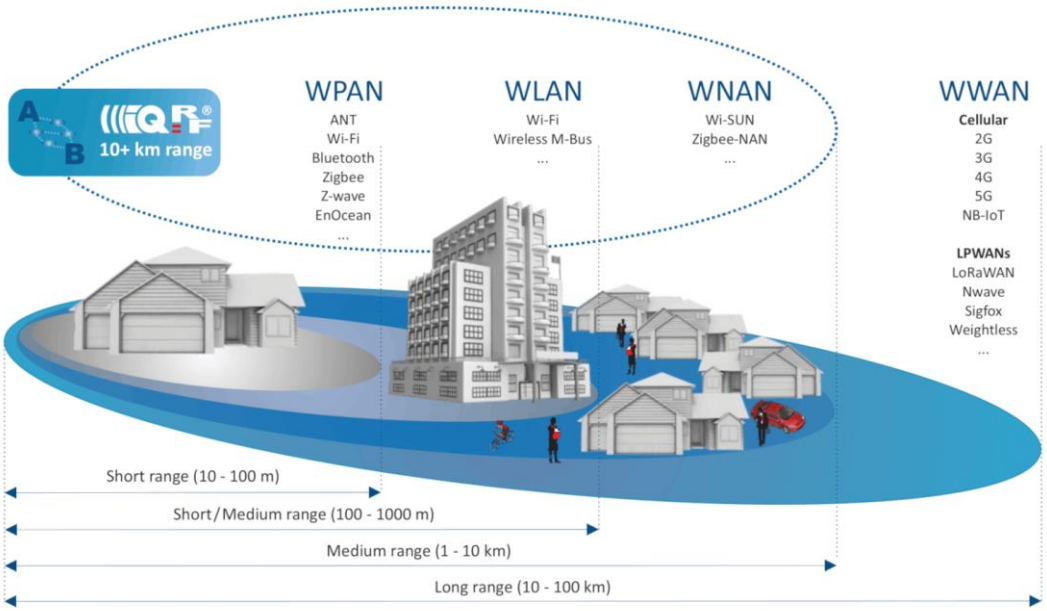
Počítačové sítě lze dělit dle různých aspektů. Lze je členit na drátové či bezdrátové, dle použitého přenosového média, oblasti dosahu dané sítě (pokrytého území) či její topologie.
Převážná většina IoT zařízení potřebuje nějakým způsobem komunikovat s okolím. Může se jednat o přímou interakci s koncovým uživatelem (pasažér MHD využívající služeb chytré zastávky nebo řidič automobilu hledající volné místo k zaparkování na chytrém parkovišti), či o sběr a odesílání dat na server, kde jsou data vyhodnocována a dále využita.

IoT zařízení však nejsou jen senzory a další pasivní prvky, ale také akční členy: spínače, motory pohánějící brány, relé ovládající osvětlení, rolety či jiné komponenty.
Tyto IoT prvky jsou často rozmístěny různě v prostoru, proto nebývá snadné, a mnohdy ani možné, přivést k nim strukturovanou datovou kabeláž. Mnohdy je problém i samotné napájení těchto prvků, které však lze řešit bateriemi či akumulátory. V případě návrhu komunikačních rozhraní je pak nutné zvolit bezdrátové řešení, založené na některé z dostupných technologií. S ohledem na bateriové napájení je současně třeba na toto brát ohled a vybírat pouze z těch technologií, které nabízí energeticky nenáročný provoz, tak, aby byla zaručena nejdelší možná životnost baterií či akumulátorů.

3.2.1 Bezdrátové metody komunikace

3.2.1.1 Typy bezdrátových sítí dle dosahu

WPAN (wireless personal area network): Jejich dosah je jen v řádu metrů, typicky do 10 m. Propojují dvě či více zařízení s nízkým odběrem energie. Typickým příkladem je Bluetooth Low Energy (BLE).
WLAN (wireless local area network): Díky standardům rodiny 802.11x (technologie WiFi) je tento druh bezdrátové sítě dnes nejvíce rozšířen mezi běžnými uživateli v rámci domácností. Dosah těchto sítí bývá většinou do 100 metrů.
WMAN (wireless metropolitan area network): Rozšíření bezdrátových LAN sítí na větší plochu, jako je území obce či města. Zástupcem WMAN sítí je například technologie WiMAX, která pracuje převážně v licencovaných pásmech, podobně jako většina ostatních technologií WMAN sítí.
WWAN (wireless wide area network): Konektivita zde pokrývá rozlehlá území, nejčastěji celých států. WWAN sítě jsou pro hlasové i datové služby obvykle využívány mobilními operátory a jsou provozovány v licencovaných pásmech.
Jako speciální podkategorii WWAN sítí lze chápat sítě typu **LP-WAN** (low power wide area network, viz Obrázek 8). Jejich nejrozšířenějším zástupcem je technologie LoRa, pracující na síti LoRaWAN. [17]



Obrázek 8 Kategorie bezdrátových sítí dle dosahu [18]

3.2.1.2 Frekvenční pásma bezdrátové komunikace

Jednotlivé rádiové kmitočty a podmínky jejich užívání upravuje Český Telekomunikační Úřad. Pro IoT zařízení jsou důležitá tzv. bezlicenční pásma, tedy frekvence, které mohou zařízení využívat, aniž by bylo nutné mít pro jejich provoz zvláštní povolení.
IoT prvky mohou volně využívat tato kmitočtová pásma: 27 MHz, 433 MHz, 868 - 876 MHz, 915 - 921 MHz, 2,4 GHz. [19]

3.2.1.3 Topologie bezdrátových sítí

Volba vhodné struktury sítě může výrazně ovlivnit její efektivitu. U bezdrátových sítí jsou to především parametry ovlivňující celkovou rychlost komunikace, ztrátovost paketů, ale i způsob, jakým jsou v dané síti řešeny případné kolize. Pro všechny tyto, ale i další parametry, může být rozhodující volba vhodné topologie sítě.

Point-to-point

Nejjednodušším typem je síť *point-to-point*. Jde o propojení mezi dvěma nody. V praxi je tento druh zapojení využíván jen velmi zřídka. [20]

Kruh

V *kruhové* topologii je třetí node propojen s oběma ostatními nody. Každý další node je následně přidán mezi dva stávající nody a tvoří tak smyčku. Data jsou přenášena kruhem v jednom směru či obousměrně. Každý node příchozí data prozkoumá a vykoná na základě nich akci, případně je předá dále, dokud nedorazí k cílovému zařízení. [20]

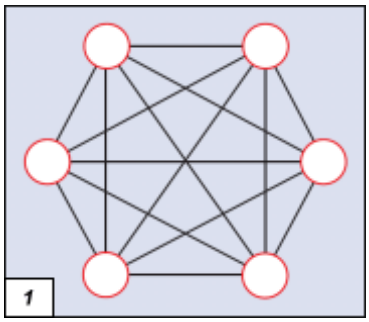
Hvězda

V bezdrátových technologiích obecně má největší uplatnění topologie *hvězda*, kde se každý prvek připojuje k centrálnímu uzlu, který rozesílá data, kamkoli je třeba. Nejběžnějším příkladem může být domácí síť. Všechna připojená zařízení, jako jsou smartphony, tablety, tiskárny či notebooky se připojují k bezdrátovému přístupovému bodu, který je obvykle zároveň i v roli routeru. Topologie hvězda je vhodná pro souběžné propojení kabelových i bezdrátových zařízení v jedné síti. [20]

Mesh

Sítě typu *mesh* se člení na dva typy – plná a částečná. V plné mesh síti (její podobu ilustruje Obrázek 9) je každý node propojen s každým přímo. Takto robustní řešení se používají například v armádních sítích, kde je stoprocentní redundance zásadní. Oproti tomu v částečné mesh síti je každý node připojen k jednomu nebo několika dalším nodům. Připojení k více nodům zvyšuje odolnost sítě a také pomáhá zvýšit efektivní dosah – node A sice nemusí být v přímém dosahu nodu C, ale zpráva mu bude i přesto doručena, díky cestě přes node B. [20]

Mesh sítě se nejvíce rozvíjí v oblasti IoT, teoreticky totiž dovolují připojit nekonečné množství zařízení na nekonečnou vzdálenost. S využitím mesh sítí lze budovat skutečně robustní řešení chytrých domácností či měst. [20]



Obrázek 9 Mesh síť [21]

3.2.2 **Fyzická komunikační rozhraní**

3.2.2.1 **SPI**

SPI (Serial Peripheral Interface) je plně duplexní sériové rozhraní, které je využíváno pro komunikaci po sběrnici mezi mikrokontrolery. Umožňuje obousměrnou komunikaci mezi jedním řídicím zařízením (master) a jedním nebo více podřízenými (slave) zařízeními. Protože pro SPI protokol neexistuje formální standard, mohou se různá zařízení chovat poněkud odlišně. Komunikační linka vypadá následovně:

MOSI (Master Out, Slave In)

Používá se k odesílání dat z master zařízení na slave zařízení.

MISO (Master In, Slave Out)

Využívána k odesílání dat ze slave zařízení na master zařízení.

SCLK (Shared / Serial Clock)

Tato linka je určena pro signál, kterým jsou synchronizována sériová data s přijímajícím zařízením. Díky tomu toto zařízení ví, kdy má přečíst vstup.

SS (Slave Select)

Linka indikující výběr podřízeného zařízení. [22]

3.2.2.2 **UART**

Název tohoto rozhraní je zkratkou anglického Universal Asynchronous Receiver / Transmitter. Je určeno pro sériovou komunikaci. Nastavit lze pro synchronní či asynchronní režim. Obecné vlastnosti periferie UART jsou:

- Plně duplexní komunikace,
- programovatelná rychlost (tzv. baud rate),
- možnost generovat přerušení,
- hardwarová kontrola parity,
- programovatelná délka 8 nebo 9 bitů,
- programovatelné 1 nebo 2 stop bity,
- 5kanálové rozhraní DMA. [23]

3.3 IQRF

Podkapitola pojednává o české technologii IQRF. Jsou zde obsaženy informace o vývoji společnosti, charakteristice bezdrátových prvků, komunikačních rozhraní a protokolů, které tyto prvky využívají.

3.3.1 Vývoj společnosti

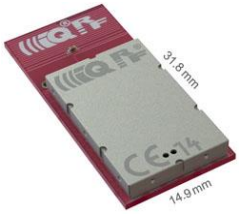
V roce 1991 byla založena česká společnost MICRORISC, s.r.o.. Tato firma se již od svého vzniku zaměřuje na výzkum a vývoj v oblasti nových technologií a elektronických komponent.
První bezdrátový transceiver IQRF společnost představila na konci roku 2004. Od roku 2010 začala se svým řešením bezdrátové komunikace sbírat nejrůznější ocenění v rámci veletrhů, konferencí i v odborných publikacích. V roce 2013 byl s cílem spojovat vývojáře, výrobce, univerzity, ale i vývojová střediska a systémové integrátory, využívající technologii IQRF, založen nadnárodní projekt IQRF Alliance. Roku 2017 pak byla založena společnost IQRF Tech, s.r.o.. Ta má nyní na starost veškeré činnosti a aktivity, týkající se IQRF, včetně samotného vývoje. [24]

3.3.2 O technologii IQRF

IQRF je platforma pro bezdrátový přenos malých objemů dat nízkou rychlostí a s nízkou spotřebou energie. Dosah mezi prvky je v budovách v řádu desítek metrů, ve volném prostranství pak v řádu stovek metrů. Silně tedy závisí na konkrétní konfiguraci a terénních podmínkách. Celou IQRF mesh síť můžeme kategorizovat jako WLAN až WWAN, v závislosti na její velikosti dle dané aplikace (graficky znázorňuje Obrázek 8 v podkapitole 3.2.1.1). Technologie IQRF může být využita pro telemetrii, ovládání průmyslových strojů, automatizaci budov a měst. Zde může jít kupříkladu o chytré parkování, ovládání a kontrolu stavu lamp veřejného osvětlení, měření koncentrace různých plynů v ovzduší, či sledování aktuální teploty a vlhkosti vzduchu. [18]

3.3.3 **IQRF transceiver**

Pro účely této práce byl zvolen model transceiveru TR-72DA (viz Obrázek 10). Jde o velmi malý prvek o délce přibližně 32 mm a šířce 15 mm. I přes to, že tento typ disponuje pouze integrovanou anténou, jeho výrobcem udávaný dosah ve volném prostranství za ideálních podmínek činí až 500 metrů. [25]



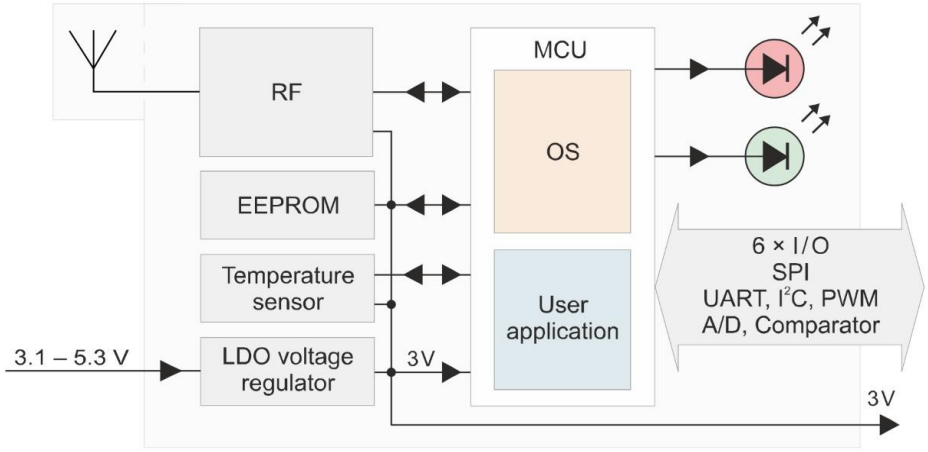
Obrázek 10 IQRF TR-72DA [25]

Parametry transceiveru TR-72DA jsou:

- frekvenční pásmo 868 a 916 MHz (softwarově nastavitelné)
- vysoký vysílací výkon 10 mW
- 6 I/O pinů
- EEPROM 32 KB
- anténa integrovaná na PCB
- regulátor napětí, vstupní napájecí napětí 3,1 – 5,3 V
- montáž na SIM konektor KON-SIM-02
- přesný teplotní senzor (verze TR-72DAT) [25]

3.3.3.1 **Blokové schéma**

Obrázek 11 popisuje blokové schéma IQRF transceiveru TR-72D.



Obrázek 11 Blokové schéma IQRF TR-72 [25]

3.3.3.2 Energetické režimy, spotřeba

U modulů IQRF je dostupných celkem 8 energetických režimů, ze kterých lze softwarově volit dle potřeby pro danou aplikaci. V Tabulka 1 jsou tyto režimy přehledně uvedeny.

Režim	Spotřeba
Hluboký spánek	1,7 µA (všechny periferie vypnuty, RF IC v režimu Standby)
Spánek	2,3 µA (všechny periferie vypnuty, RF IC v režimu Sleep)
Běžný provoz	
RF spánek	1,4 mA
RF připraven	2,8 mA
RX režim	
STD	12,1 mA
LP	260 µA
XLP	18,5 µA
TX režim	8,3 mA – 25 mA (dle RF vysílacího výkonu)

Tabulka 1: provozní režimy IQRF transceiveru [26]

Nejzajímavější a pro tuto práci důležité jsou režimy TX a LP, resp. XLP.

Vysílání (TX, transmit)

V TX režimu se spotřeba pohybuje v rozmezí 8,3 mA – 25 mA v závislosti na nastaveném vysílacím výkonu. Vysílací časy jsou však velice krátké, proto ani takto vysoké hodnoty spotřeby nejsou z celkového pohledu nijak zásadní.

Přijímání (RX, receive)

Spotřeba energie v LP (low power) režimu dosahuje u modulu TR-72D hodnoty 260 μA.

V režimu XLP (extra low power) je modul ještě o řád úspornější, spotřeba zde činí přibližně 18,5 μA.

Dle dokumentace IQRF lze modul v XLP režimu provozovat teoreticky až 7 let na jednu baterii o kapacitě 1 Ah (napětí 3,6 V), při nejvyšším RF vysílacím výkonu, s celkovými 300 MB přijatých a 200 MB odeslaných dat. [27]

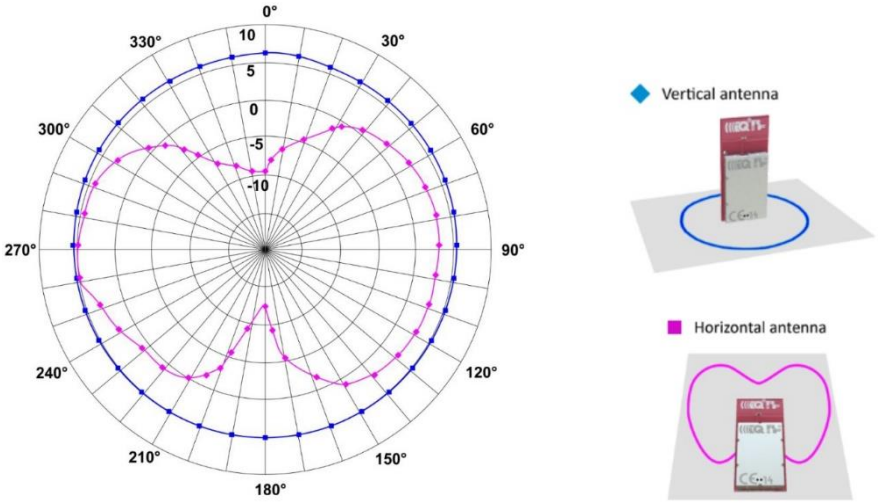
3.3.3.3 **RF dosah**

Dosah RF signálu závisí silně na následujících podmínkách:

Hardware – konstrukční vlastnosti IoT zařízení, jehož je transceiver součástí, zvláště pak umístění TR modulu v rámci celku, návrh PCB, uzemnění, vodivé plochy, umístění v blízkosti kovových objektů či baterií. Pro zajištění efektivního dosahu a spolehlivé konektivity nesmí být v blízkosti transceiveru umístěny žádné komponenty, které by mohly dosah ovlivnit. Výrazně jej mohou snížit i nevodivé části, jako je PCB základní desky, umístěné pod anténou.

Dosah také ovlivňuje fyzické uspořádání transceiverů, především vzájemná orientace antén s ohledem na polarizace a vyzařovací charakteristiky. Příklady správných a nesprávných vzájemných uspořádání, ale i vyzařovací charakteristiky modulu ilustrují Obrázek 12 a Obrázek 13.

Aplikační software – RF výkon lze volit z celkem osmi dostupných úrovní. Pro zvýšení odolnosti vůči RF rušení lze příchozí RF signál filtrovat podle jeho síly. [26]



Obrázek 12 TR-7xDA výstupní RF výkon [v dBm] vs. orientace antény [26]

Obrázek 13 Příklady správných a nesprávných uspořádání párů TR-72DA [26]

3.3.4 Komunikace mezi IQRF prvky

Pro přenos dat mezi prvky IQRF existují dva různé způsoby komunikace. Jedná se o tzv. non-networking režim (peer-to-peer) a networking režim (IQMESH). [28]

3.3.4.1 Non-networking režim

Zde jsou propojena všechna zařízení způsobem peer-to-peer. Je možné takto propojit dvě nebo více zařízení IQRF mezi sebou. V tomto režimu jsou data dostupná všem zařízením v síti bez možnosti omezení, není možno nastavit síťové funkce, jako například směrování. V tomto režimu se transceivery programují přímo pod IQRF operačním systémem, což vyžaduje znalosti programování v jazyce C. [28]

3.3.4.2 **Networking režim**

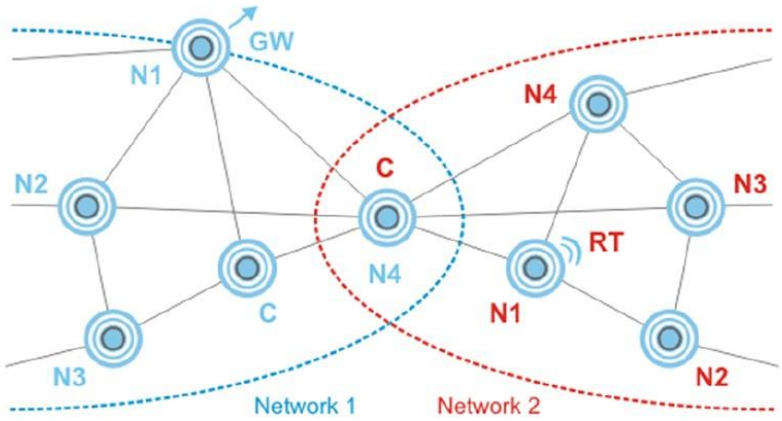
V tomto režimu je v každé IQRF mesh síti přítomen právě jeden řídicí prvek, tzv. koordinátor. Ten zvládá řídit provoz v síti až do počtu 239 nodů. Pro tento typ spojení používá IQRF pojmenování IQMESH.

Je zde podporováno automatizované směrování paketů včetně redundantních cest a další pokročilé funkce. Tento režim umožňuje využít protokol DPA (Direct Peripheral Access) a příkaz FRC (Fast Response Command) pro hromadný sběr dat ze sítě. Díky vyšší aplikační vrstvě (DPA) se obejde většinou bez nutnosti programování. [28]

3.3.4.3 **IQMESH**

Protokol IQMESH byl definován v roce 2005 jako základní komunikační protokol pro IQRF zařízení a pro nasazení v takových aplikacích, kde je vyžadována nízká spotřeba energie a kde postačí nízká přenosová rychlost dat. Uplatnění nalezne například v domácí a podnikové automatizaci či telemetrii.

Obrázek 14 zobrazuje příklad řetězení sítí IQMESH. [29]



Obrázek 14 Řetězení sítí IQMESH [29]

IQMESH využívá síťovou topologii mesh a je založen na synchronizovaném směrovém zaplavení. V rámci jedné sítě podporuje až 240 zařízení – jeden koordinátor který síť řídí, a 239 nodů. [30]

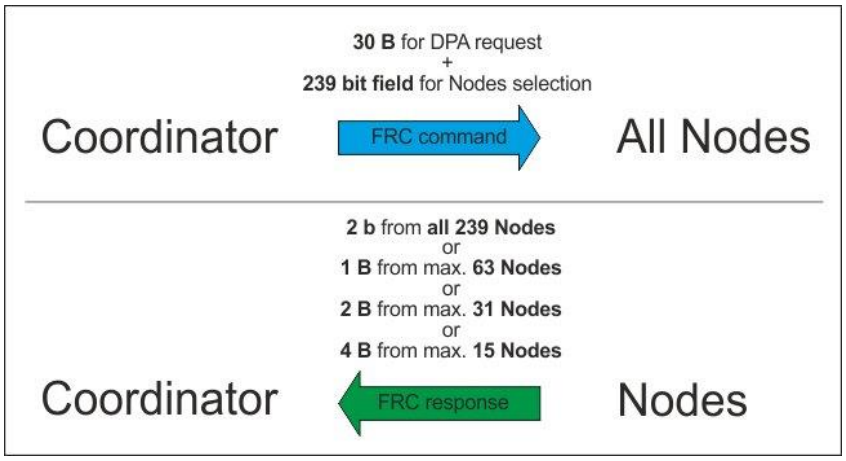
Protokol IQMESH podporuje jak individuální, tak i skupinové adresování, stejně tak jako síťový broadcasting. Vedle standardních vlastností jako bonding a discovery podporuje také přímou adresaci periferií. Protokol zvládá jak komunikační schéma point-to-point, tak i více

komplexní síťové topologie, jako hvězda či mesh. Jeden byte uvnitř IQMESH paketu je vyhrazen pro určení směrovacího algoritmu.

3.3.4.4 **FRC: Fast Response Command**

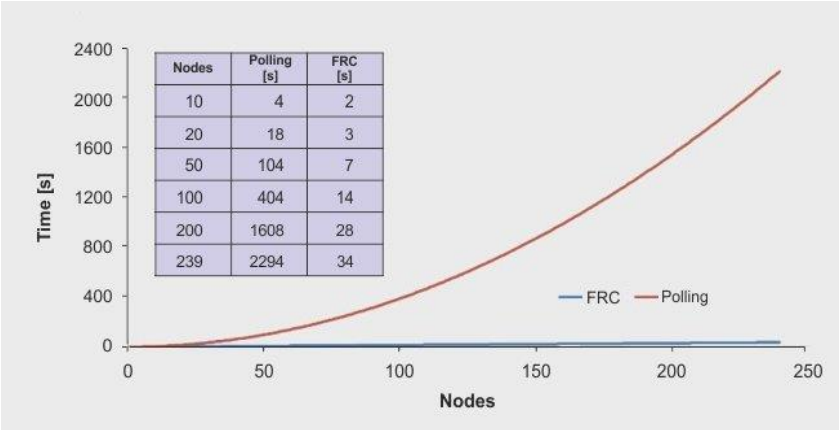
Příkaz FRC umožňuje efektivní hromadný sběr dat z většího množství nodů. Ideální uplatnění je pro sběr stejného typu informace. Příkladem mohou být lampy veřejného osvětlení, kde lze vysláním jednoho FRC povelu všechny lampy rozsvítit a současně od nich obdržet informaci o jejich stavu. [30]

Množství dat, které lze sbírat v závislosti na počtu nodů, popisuje Obrázek 15:



Obrázek 15 Princip Fast Response Command [31]

Obrázek 16 ukazuje srovnání rychlosti sběru dat způsoby polling (červeně) a FRC (modře).



Obrázek 16 Polling vs. FRC [31]

3.3.5 **Zabezpečení**

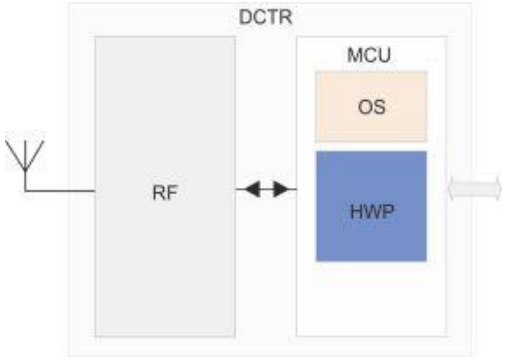
Bezdrátová komunikace v mesh síti mezi IQRF prvky je standardně šifrována pomocí AES-128. Pro přístup do sítě a dešifrování provozu se zadává přístupové heslo, tzv. *Access Password*.
Další, volitelnou možností, je šifrování samotného obsahu jednotlivých zpráv. To lze provést nastavením uživatelského hesla, tzv. *User Key*.
V IoT sítích je taková úroveň zabezpečení poměrně neobvyklá a tedy nadstandardní. Běžně dostupná IoT zařízení většinou nedisponují dostatečně výkonným hardwarem pro šifrování a dešifrování provozu. Pro tento účel by musely být vybaveny dedikovaným kryptočipem.
Kromě IQRF svoji komunikaci šifrují také například IoT platformy Zigbee a BigClown. Obě rovněž využívají standardu AES-128.

3.3.6 **DPA: Direct Peripheral Access**

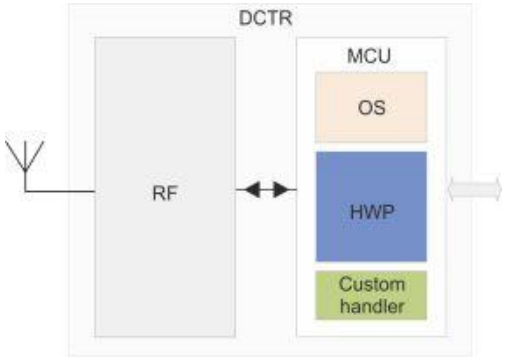
DPA je patentovaná technologie IQRF. Jde o jednoduchý bytově-orientovaný protokol pro ovládání periferií a služeb zařízení v síti IQMESH prostřednictvím rozhraní SPI nebo UART.
DPA Využívá vlastní třívrstvou architekturu:

- Operační systém,
- vrstvu DPA s předkonfigurovaným HW profilem (HWP) z výroby,
- tzv. Custom DPA Handler.

Předkonfigurovaný HWP (Obrázek 17) plně dostačuje pro velkou část uživatelských aplikací. Díky tomu není třeba cokoliv dodatečně programovat.
V případě, že výchozí HWP nevyhovuje potřebám aplikace, pak je možné využít Custom DPA Handler (Obrázek 18). Tím se prakticky rozšíří základní HWP o dodatečnou funkcionalitu dle návrhu uživatele. Toto však již vyžaduje znalosti programování v jazyce C. [32]



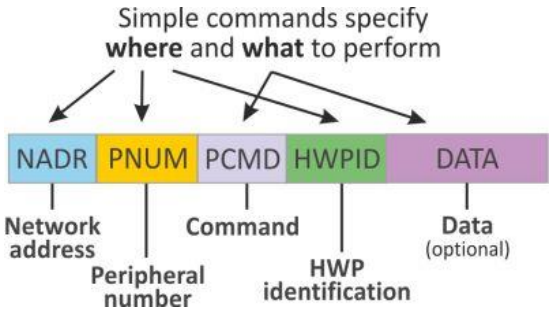
Obrázek 17 Předkonfigurovaný HWP [32]



Obrázek 18 HWP + Custom DPA handler [32]

3.3.6.1 Struktura DPA paketu

Následující Obrázek 19 graficky znázorňuje podobu DPA paketu. Pod obrázkem je tato struktura popsána.



Obrázek 19 DPA paket [33]

NADR

Síťová adresa zařízení. Přestože má velikost 2 byty, 2 B adresace není podporována (vyšší byte je ignorován).

PNUM

Číslo periferie o velikosti 1 bytu. 0x00 značí koordinátor, 0x01 node, 0x06 červenou LED, 0x07 zelenou LED, 0x0A teploměr, 0x0C UART a 0x0D FRC. 0x5E značí IQRF standardní senzor.

PCMD

Příkaz, specifikující akci, která má být provedena. Povolený rozsah hodnot závisí na vztaženém typu periferie. Nejvýznamnější bit je vyhrazen pro indikaci DPA odpovědi.

HWPID

HW Profile ID unikátně určuje zařízení, jeho funkci, uživatelskou periferii, a podobně.

DATA

Volitelná datová část zprávy. [32]

3.3.7 **Standard IQRF Sensor**

Pod pojmem standardní senzor IQRF definuje některé druhy senzorů. Standard je implementován za použití jedné DPA periferie a tří DPA příkazů. Je určen pro různé kombinace senzorů nehledě na to, zda jde o zařízení obsahující pouze jeden senzor, více senzorů stejného typu nebo kombinaci více senzorů různých typů. Zařízení podporující tento standard může obsahovat až 32 senzorů předdefinovaných typů (množství). Při navracení senzorických dat v DPA odpovědi v reakci na DPA požadavek či navrácené z FRC, každý typ senzoru definuje množství, jednotku a rozlišení.

Senzory implementované daným zařízením jsou adresovány vzestupnou posloupností indexů od 0 do 31. V rámci této posloupnosti musí být využity všechny indexy (čísla indexů nelze přeskakovat).

PNUM ID standardu je 0x5E. [34]

3.3.7.1 **DPA příkazy**

Vyčítání senzorů

Tímto příkazem lze číst data ze zvolených senzorů. Volitelně mohou být také zapsána data do těch senzorů, které to podporují. Podporováno je až 32 senzorů.

Požadavek

NADR	PNUM	PCMD	HWPID	0 … 3	4 … 3 + n * 5
NADR	0x5E	0x00	0xXXXX or 0xFFFF	Bitmap	WrittenData

Tabulka 2 Datový rámec pro požadavek na vyčítání senzorů [34]

Bitmapa – 32bitová bitmapa specifikuje senzory, ze kterých budou čtena data. Existují tři možnosti, jak lze bitmapu použít. Detaily a očekávaná odpověď na požadavek jsou rozebrány dále.
Čtení prvního senzoru – zde není bitmapa použita. To znamená že požadavek neobsahuje žádná data. V tomto případě není pole WrittenData přítomno. Čtení senzorů – ostatní hodnoty bitmapy.

WrittenData – volitelná data k zápisu do senzorů. Datový blok WrittenData se skládá ze skupiny pěti bytů pro každý senzor. První byte skupiny značí index senzoru, další čtyři byty jsou data k zápisu do senzoru. Význam zapisovaných dat je buď standardizován nebo proprietární. Pokud délka WrittenData není násobkem 5 nebo některý z uvedených senzorů nepodporuje zápis dat, pak je vrácen chybový kód ERROR_DATA_LEN. Pokud není validní obsah zapisovaných dat, je vrácen chybový kód ERROR_DATA. [34]

Odpověď

NADR	PNUM	PCMD	HWPID	ErrN	DpaValue	0 … n
NADR	0x5E	0x80	0xXXXX	0	?	Data

Tabulka 3 Datový rámec odpovědi při vyčítání senzorů [34]

Data – obsah závisí na formátu bitmapy z Požadavku:
1. čtení prvního senzoru:
Totožné, jako 2. možnost, jen s hodnotou bitmapy rovnou 0x00.00.00.01 (byty 0x01, 0x00, 0x00, 0x00), tzn. je zvolen první senzor (index 0).
2. čtení senzorů:
Zde data obsahují seznam hodnot senzorů ze všech zvolených senzorů:
Data = [hodnota 1. zvoleného senzoru], [hodnota 2. zvoleného senzoru], ..., [hodnota posledního zvoleného senzoru].

Délka a formát hodnoty každého senzoru závisí na typu daného senzoru. Pokud se data ze všech zvolených senzorů nevejdou do maximální rezervované délky pole Data (u DPA 3.00 tvoří jeho délka 56 bytů), pak je navrácen chybový kód ERROR_FAIL. [34]

3.3.7.2 **Typy senzorů**

Typ senzoru je jednobytová hodnota, která specifikuje typ (množství) senzorů, stejně jako velikost senzorických dat v případě příkazu pro vyčítání senzorů a FRC. Tento byte využívá následující formát ke specifikaci délky dat a typu senzoru (bity "x"):
[0000.0000] – nedefinováno
[0xxx.xxxx] => 2 byty dat (127 možností)
[100x.xxxx] => 1 byte dat (32 možností)
[101x.xxxx] => 4 byty dat (32 možností)
[11xx.xxxx] => proměnlivý počet bytů (64 možností), první data byte specifikuje počet zbývajících datových bytů (např.: 5, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE). [34]

3.3.7.3 **Vybrané typy senzorů**

Standard Sensor aktuálně nabízí celkem 40 typů senzorů, které lze využít. Žádný z nich však není přímo určen k ovládání servomotoru. Uvedu nyní proto z dostupných možností takové typy, které jsou z mého pohledu nejvhodnější pro účely použití se servomotorem a interakci s celou sestavou v okně.

Binary Data7 [0x81]
Návratová hodnota tohoto typu senzoru je 7 datových bitů blíže nespecifikovaného významu. Může být použit senzorem k navracení chybových stavů, pro binární vstupy, čítače a podobně.
Podoba dat je následující:
[ebbb.bbbb]
bit 0…6 – binární data.
bit 7 – specifikuje chybu senzoru. Pokud je tento bit nastaven, pak daná binární data nejsou validní. Binární datové bity 0-6 musí být vynulovány. [34]

Data Block [0xC0]

Návratová hodnota tohoto typu je množství datových bytů nespecifikovaného významu. Může být použit senzorem pro navrácení většího množství bytů. První byte zde určuje množství zbývajících datových bytů. [34]

Relative Humidity [0x80]

V případě typu pro relativní vlhkost má návratová hodnota velikost jednoho bytu. Jednotkou je zde jedno procento, rozlišení je 0,5 % a praktický rozsah je od 0 % do 100 %. Pokud je vrácena hodnota 0xEE (tj. 119 %), je vyhodnocena jako chyba senzoru. Ostatní hodnoty nejsou definovány.

Podoba dat je následující:

[iiii.iiif]

bit 0 – desetinná část (jednotka 1/2 procenta).

bit 1…7 – celočíselná část. [34]

Low Voltage [0x06]

V případě typu senzoru pro nízké napětí je návratová hodnota dvoubytová se znaménkem. Jednotkou je jeden Volt, rozlišení je 1/16 V = 0,0625 V a rozsah činí ±2047,9375 V. Hodnota 0x8000 (tj. -2048 V) značí chybu senzoru.

Podoba dat je následující:

[siii.iiii.iiii.ffff]

bit 0…3 – desetinná část (jednotka 1/16 V).

bit 4…14 – celočíselná část (jednotka 1 V).

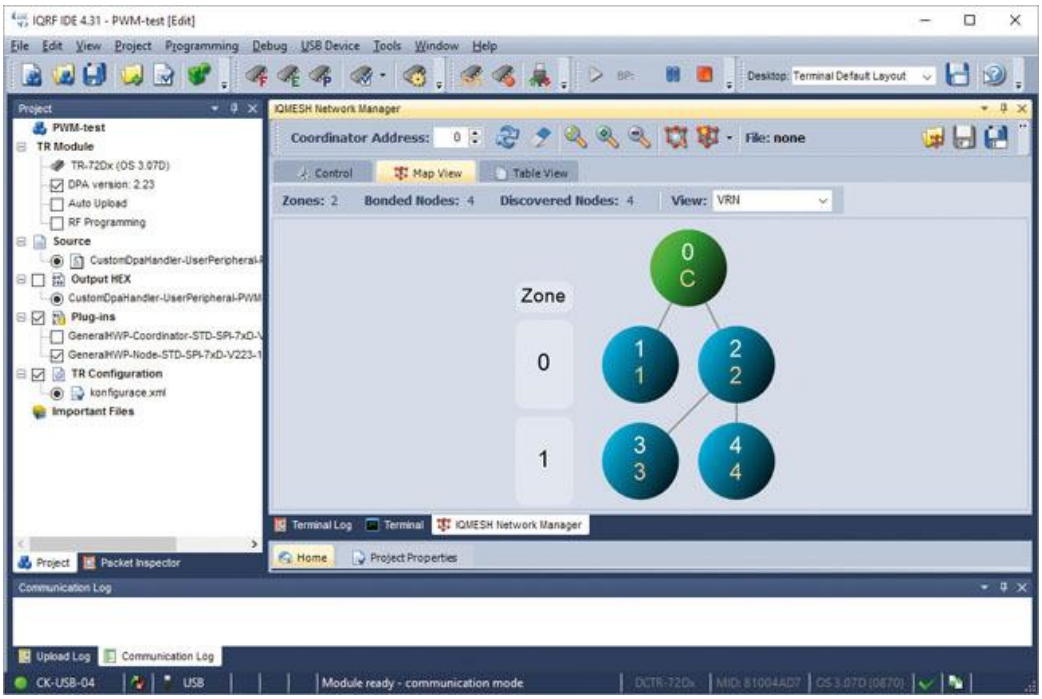
bit 15 – znaménko. [34]

3.3.8 **IQRF IDE**

IQRF IDE je desktopové vývojové prostředí, vyvinuté společností IQRF. Je volně k dispozici na webových stránkách společnosti, dostupné je z odkazu *https://www.iqrf.org/support/downloads*. V této aplikaci lze konfigurovat jednotlivé prvky sítě, provádět nastavení koordinátoru i ostatních nodů a nahrávat firmware do transceiverů. Spárováním koordinátoru a nodů konstruujeme mesh síť.

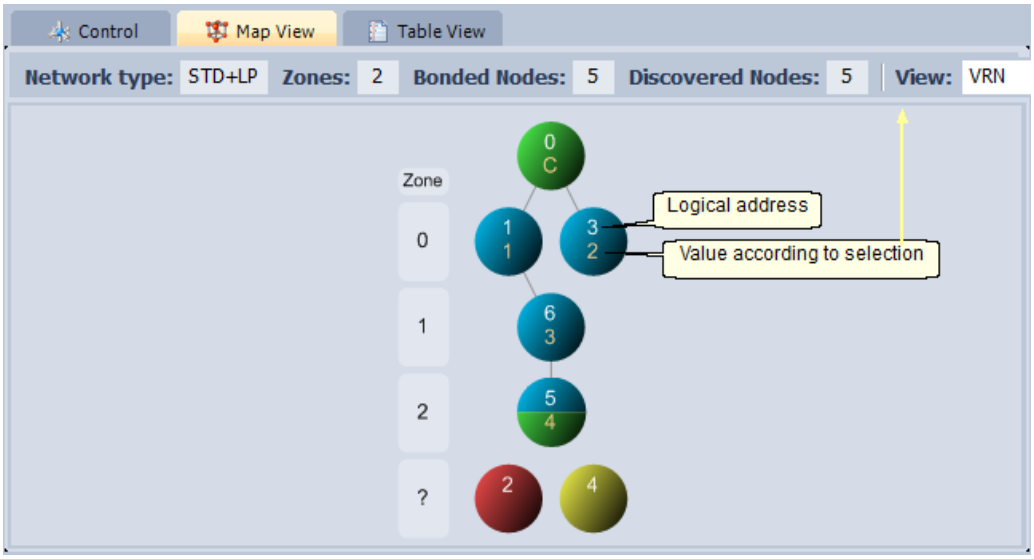
Aplikace také poskytuje náhled aktuální podoby mesh sítě ve formě grafického zobrazení prvků a vazeb mezi nimi.

V sekci „Project“ lze volit jednotlivé součásti, které budou nahrány do transceiveru a budou tak definovat jeho funkce a chování.



Obrázek 20 IQRF IDE [35]

Koordinátor je zobrazen vždy zeleně (Obrázek 20 – koordinátor je vidět nahoře, nody jsou pod ním) a v každé síti může být pouze jeden. Ostatní nody jsou zobrazeny v modré barvě, pokud jsou již součástí dané mesh sítě a mohou se účastnit směrování paketů. Node, který v síti zatím není, ale je v dosahu a odpovídá na polling (tj. příkaz, který koordinátor posílá do sítě, aby odhalil dostupné nody), se zobrazí žlutě. Červený je pak ten node, který je zcela nedostupný. Tyto možné stavy nodů zachycuje Obrázek 21.



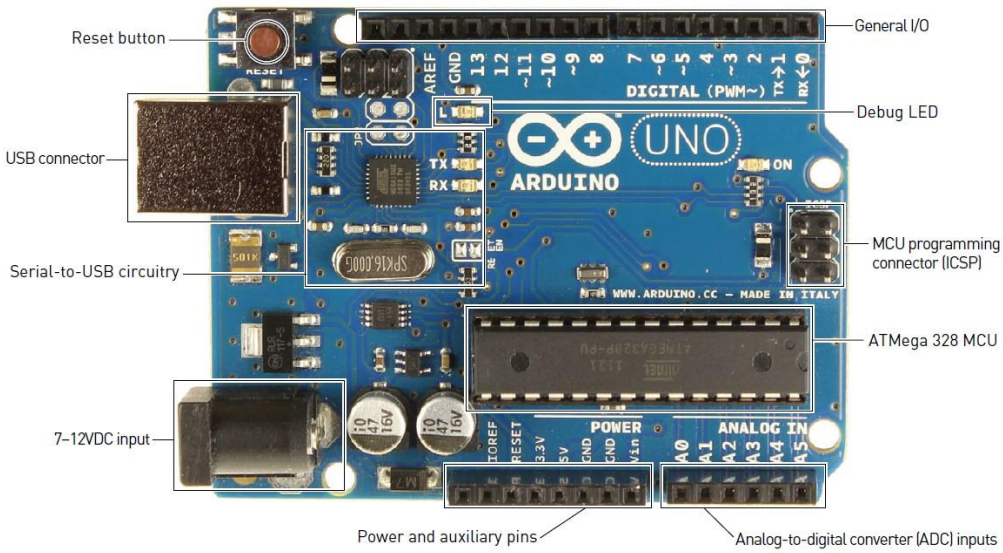
Obrázek 21 Grafické znázornění prvků sítě [36]

IQRF IDE obsahuje velké množství různých nástrojů usnadňujících konfiguraci a vývoj. Je zde přítomen debugger, textový terminál, *IQMESH Network Manager* (součást IQRF IDE obsahující sadu nástrojů pro práci s IQMESH sítí), ale mimo jiné i Packet Inspector, který umožňuje monitorovat tok dat a provoz v IQRF mesh síti / zobrazit obsah příchozích i odchozích paketů (z pohledu koordinátoru). Tato funkce je velice důležitá pro jakýkoliv vývoj a ladění software na straně nodu.

3.4 Arduino

Arduino (Obrázek 22 – ve verzi Uno) je druh miniaturního jednodeskového počítače a zároveň vývojová platforma.

Řídí jej mikrokontroler ATmega společnosti Atmel. Platforma Arduino je založena na open-source principu, hardware i software jsou tedy vydané pod GNU GPL licencí. Kdokoliv tak má možnost volně kopírovat či upravovat hardwarový návrh Arduino desek i jejich programové vybavení. [22]



Obrázek 22 Arduino Uno [22]

3.4.1 **Specifikace, princip funkce**

Mikrokontrolery Arduino disponují sběrnicemi SPI, UART, I2C i OneWire. To dává uživateli možnost zvolit si konkrétní sběrnici, která je pro jeho projekt nejvhodnější. [22] [37]

Pomocí sběrnic lze k Arduinu připojit nejrůznější periferní prvky, vstupní i výstupní moduly a senzory. Přes sběrnice UART či SPI je možné k mikrokontroleru připojit IQRF transceiver. Základem libovolného programu, který má v mikrokontroleru běžet, musí být vždy funkce *setup ()* a *loop ()*.

Kód, který je uvnitř funkce *setup* se spouští vždy jen jednou, na začátku běhu programu. Je vhodné sem umístit jednorázová nastavení, jako například inicializace sériové linky včetně určení její rychlosti v baud/s, deklarace vstupních či výstupních pinů a jejich přiřazení periferiím.

Obsah funkce *loop* je vykonáván cyklicky ve smyčce, po celou dobu, co je Arduino zapnuté. S tím je třeba počítat, například pokud chceme, aby mikrokontroler vykonával určitou činnost v daných intervalech – pak je nutné mezi jednotlivé příkazy pro dané úkony zařadit funkci *delay ()* se zadaným atributem času v milisekundách, po který je třeba, aby Arduino čekalo. [22]

3.4.2 **Energetické režimy, spotřeba**

Následující Tabulka 4 Arduino a režimy spotřeby energie mapuje energetické režimy, které lze na mikrokontroleru Arduino nastavit, na jednotlivé systémové funkce a periferie zařízení, které jsou při zvolených režimech v provozu.

Table 9-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

Tabulka 4 Arduino a režimy spotřeby energie [38]

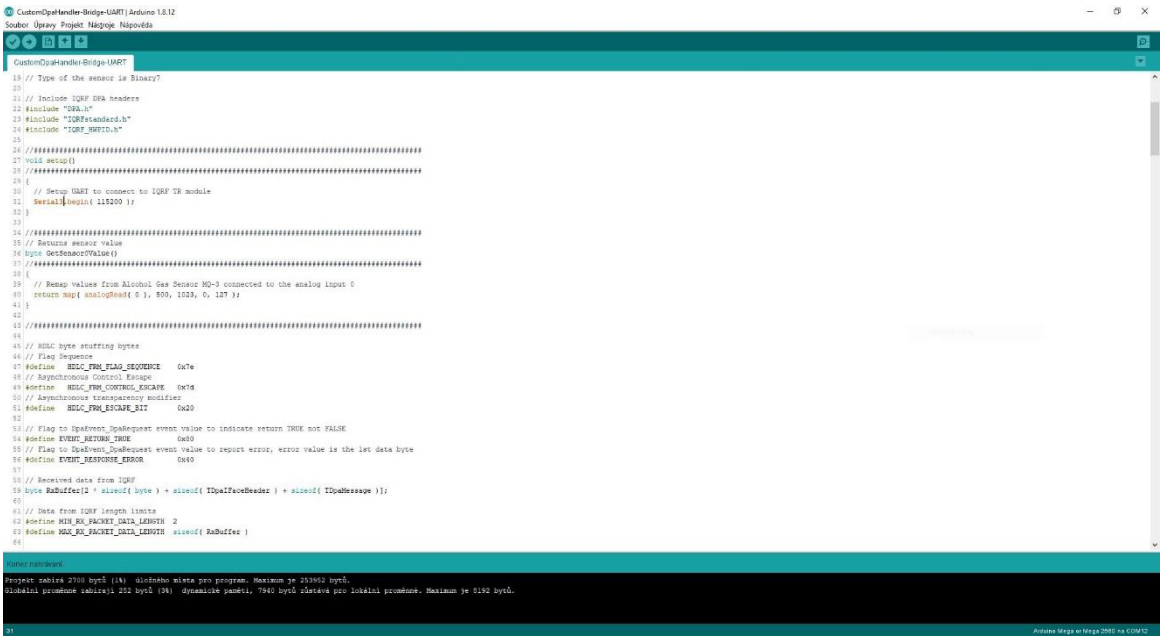
3.4.3 **Vývojové prostředí**

Psát programový kód pro nejrůznější vestavné (embedded) systémy a zařízení lze ve velkém množství různých editorů či integrovaných vývojových prostředí (Integrated Development Environment, IDE). Pro platformu Arduino je základním prostředím Arduino IDE. Existují však i jiné možnosti, například kombinace Microsoft Visual Studio Code a PlatformIO.

3.4.3.1 **Arduino IDE**

Jde o bezesporu nejznámější a pravděpodobně i nejrozšířenější IDE, pokud jde o vývoj pro Arduino. Téměř každý, kdo s touto platformou začíná, si po pořízení svého prvního mikrokontroleru stáhne a nainstaluje právě toto vývojové prostředí. Obrázek 23 ukazuje, že IDE má poměrně jednoduchý vzhled. Díky tomu se sice snadno ovládá, disponuje však spíše

jen nezbytnými základními funkcemi, jako je barevné zvýrazňování syntaxe, monitor komunikace po sériové lince či tlačítka pro kompilaci souboru a jeho nahrání do mikrokontroleru. Dalo by se tedy říci, že se spíše jedná o pokročilejší textový editor obohacený o tyto funkcionality.

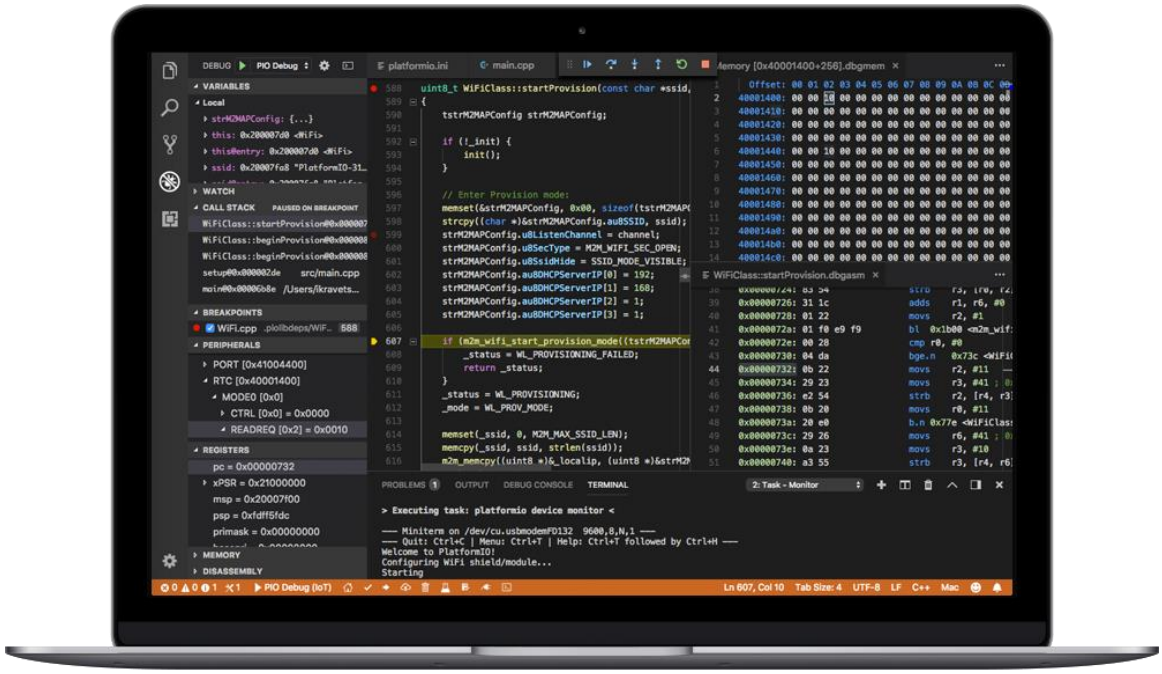


Obrázek 23 Arduino IDE [autor]

3.4.3.2 **Visual Studio Code a PlatformIO**

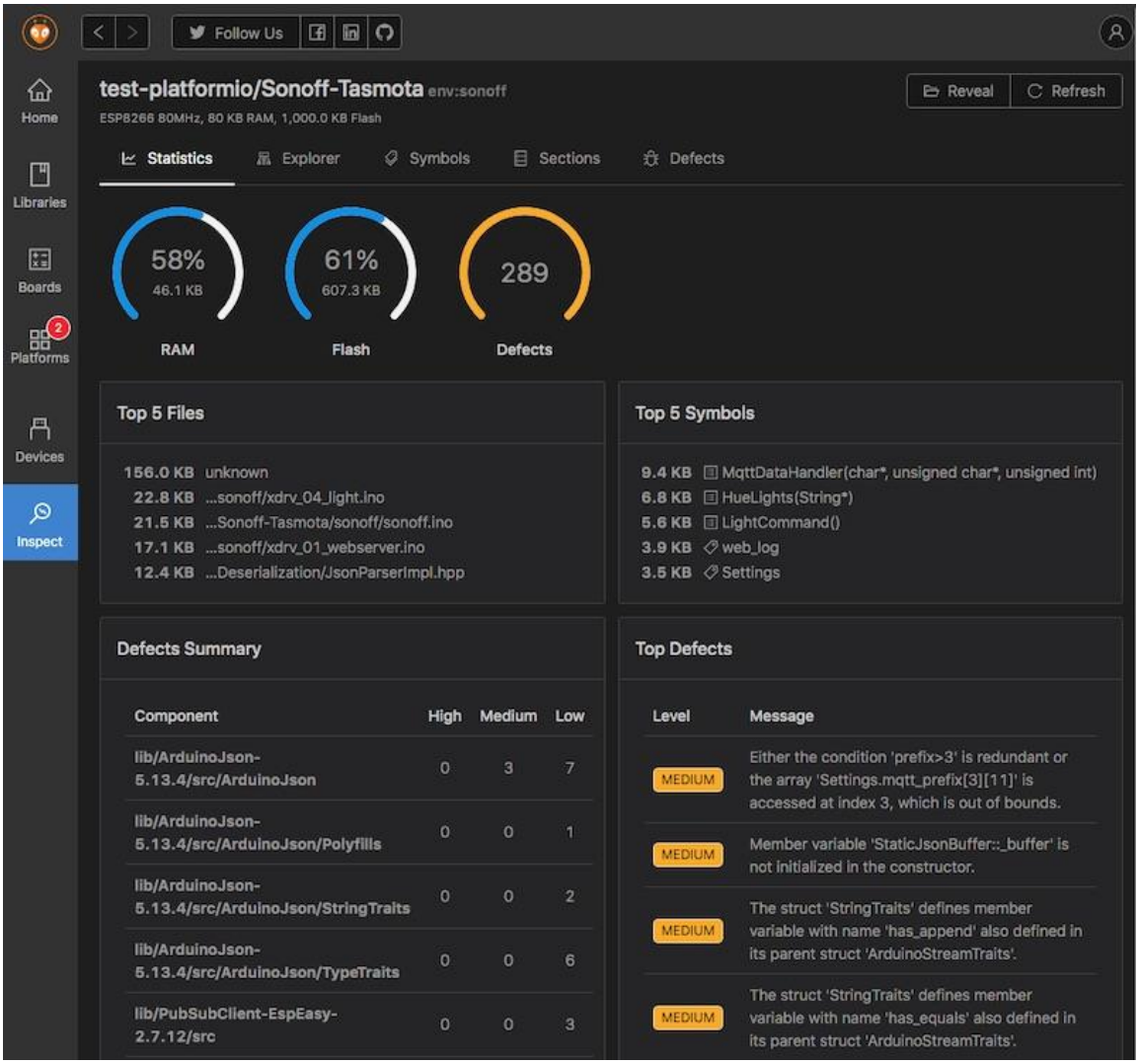
V roce 2015 společnost Microsoft vydala první verzi nového vývojového prostředí. Visual Studio Code [39] bylo vytvořeno jako odlehčená, volně dostupná verze (zdrojové kódy jsou zveřejněny na GitHub a tak vznikla i řada upravených odnoží). Dnes již však obsahuje nespočet rozšíření a pluginů a je tak velmi komplexním a seriózním nástrojem pro tvorbu kódu, ať se jedná o programování v jazycích kompilovaných (C/C++) či interpretovaných (Python), vysokoúrovňových či nízkoúrovňových (Assembler). Zároveň jde o první větší software vydaný Microsoftem jak pro Windows, tak také pro operační systém Linux. [40]
Platforma pro vývoj na embedded zařízení PlatformIO (viz Obrázek 24) je napsaná čistě v jazyce Python a není závislá na žádných dalších knihovnách či nástrojích operačního systému. Je vydávána pro operační systémy Windows, Mac OS i Linux, a to včetně ARM verze Linuxu, takže je možné ji spustit třeba i na jednodeskových kapesních minipočítačích formátu Raspberry Pi. Nemá však podobu samostatného vývojového prostředí, nýbrž

se instaluje v podobě rozšíření do jiného IDE, jakým je právě Visual Studio Code. Zde lze nainstalovat velmi snadno v prostředí Visual Studio Marketplace. [41]



Obrázek 24 PlatformIO IDE [41]

PlatformIO obsahuje integrovaný nástroj pro ladění kódu (tzv. debugger), podporuje tvorbu a spouštění unit testů, včetně jejich orchestace či automatizovaném spouštění na vzdálených strojích. Platforma dále podporuje nástroje pro statickou analýzu kódu (Obrázek 25). Díky tomu lze snadno a rychle identifikovat případné problémové (např. paměťově či výkonově náročné) části kódu. [41]



Obrázek 25 PlatformIO a statická analýza kódu [41]

Library Dependency Finder je klíčovou částí PlatformIO build systému. Ten má na starosti zpracování zdrojových souborů psaných v jazyce C/C++ a vyhledává v nich direktivy *#include*, na základě nichž předává adresáře obsahující požadované hlavičkové soubory kompilátoru. [42]

3.4.3.3 Srovnání vývojových prostředí

Z uvedeného představení obou IDE není pochyb o tom, Arduino IDE je skutečně strohým vývojovým prostředím. Kromě vestavěných vzorových programů (examples) a řady předinstalovaných často používaných knihoven nabízí jen textový editor se zvýrazňováním syntaxe, okno terminálu pro výpis zpráv kompilátoru a monitor sériové linky, otevírající se v samostatném novém okně. Správné nastavení cílového typu desky (Arduino Nano,

Micro, Uno, Mega a další), typu čipu (např. ATmega 328P pro Arduino Nano či ATmega 2560 pro Arduino Mega), COM portu a typu programátoru je plně ponecháno na uživateli. Při chybném nastavení těchto parametrů se nemusí podařit program do mikrokontroleru vůbec nahrát či může dojít k jiným chybám.

PlatformIO v rámci IDE Visual Studio Code oproti tomu nabízí možností a usnadnění spoustu. Jedna ze základních vlastností je, že prostředí samo detekuje COM port, na kterém je kontroler připojen. Typ desky se volí již při zakládání projektu a je zapsán v konfiguračním souboru platformio.ini, kompilátor tedy vždy ví, pro kterou desku je daný projekt určen a nemůže se tak stát, že by byl kód omylem zkompilován pro jiný typ. Velkou výhodu představuje funkce „IntelliSense Auto-Complete“, což je vlastnost automatického našeptávání a doplňování klíčových slov na základě kontextu a několika prvních napsaných znaků. Tato funkce může pomoci výrazně zrychlit práci a minimalizovat možné chyby či překlepy. Další pozitivní vlastností je přítomnost zmenšeného náhledu na celý aktuálně otevřený soubor. Tato funkcionalita, umístěná po pravé straně a přítomná mimo jiné v textovém editoru Sublime Text, dokáže obzvláště u velmi dlouhých souborů pomoci výrazně zrychlit navigaci po těchto souborech. Našeptávání i zmenšený náhled pro představu ukazuje Obrázek 26, kde jsou obě tyto funkce zachyceny. Užitečná může být i funkce automatického upozorňování na chybně zapsané klíčové výrazy. Slovo, kde se chyba vyskytne, je podtrženo červenou vlnovkou a po levé straně vedle čísla řádku se objeví žlutý symbol žárovky.

Obrázek 26 Funkce našeptávání a navigačního náhledu ve VS Code [autor]

Na závěr srovnání lze konstatovat, že pro skutečně jednoduché projekty malého rozsahu či jen samostatné programy Arduino IDE jistě plně postačí. Pro netriviální projekty středního a většího rozsahu však již příliš vhodné není. Pro takové použití lze doporučit kombinaci Visual Studio Code a PlatformIO. Kromě zrychlení práce díky našeptávači, hlídání překlepů a chyb v kódu bez nutnosti kompilace, uživatele příjemně překvapí i přítomnost nástrojů pro ladění či statickou analýzu kódu a také celkový vzhled a přívětivost navigace v prostředí. [43]

3.5 Knihovny pro C, C++ a mikrokontrolery

Podkapitola zkoumá možnosti v oblasti psaní knihoven, především v programovacích jazycích rodiny C.

3.5.1 Knihovny obecně

Rozdíl mezi knihovnou a běžnou aplikací je v absenci funkce *main* v případě knihovny, není je proto možné přímo samostatně spouštět. Knihovny zejména definují datové typy, obsahují také specifické funkce, které lze využít v dalších programech a aplikacích. Dělí se na statické a dynamické. [44]

3.5.2 **Statické knihovny**

Koncept statických knihoven je poměrně snadný – jakmile kompilátor přeloží zdrojový kód do binární objektové podoby, je žádoucí tyto objekty uchovat pro další, pozdější použití jiným programem. Jinými slovy, statické knihovny se k aplikaci linkují staticky v době překladu. Po kompilaci se tyto soubory archivují a vznikne z nich jediný binární soubor – statická knihovna. Tu je pak možné snadno přiřadit do jiného projektu prostřednictvím hlavičkových souborů. Toto vše lze uskutečnit za podmínky, že linker rozumí formátu, ve kterém je taková knihovna uložena, a dokáže s ním dále pracovat. Názvy statických knihoven v Linuxu mají příponu *.a*. [44] [45]

3.5.3 **Dynamické knihovny (sdílené knihovny)**

Oproti konceptu statických knihoven, které vznikly již během počátků programování, přišel koncept dynamických knihoven mnohem později, a to až v souvislosti s nástupem operačních systémů podporujících multitasking (umožňujících vykonávat více procesů současně). *Dynamické knihovny neboli sdílené knihovny jsou odděleny od binárního souboru aplikace a při jejich provozu se využívá tzv. dynamické linkování až za běhu aplikace. Jejich výhodou je, že mohou být sdíleny více aplikacemi, které běží zároveň a to tak, že operační systém je natáhne do paměti jenom jednou a ostatní aplikace je využívají až ve chvíli, kdy je to potřeba (tedy když volají knihovní funkce). Jméno knihovny podléhá zvyklostem v používaném operačním systému. Například matematická knihovna se v Linuxových systémem nazývá libm.so, ve Windows by to mohlo být jméno m.dll.* [44] [45]

3.5.4 **Knihovny pro Arduino**

Mikrokontrolery Arduino lze programovat v C/C++. Knihovny pro ně tvořené tak tvoří dva typy zdrojových souborů. První typ s příponou *.c* či *.cpp* obsahuje deklarace datových typů i konstant, dále jsou zde definovány globální proměnné a implementovány jednotlivé funkce. Ve druhém typu s příponou *.h* jsou obsaženy deklarace veřejných datových typů, konstant či globálních proměnných, ale i prototypy veřejných funkcí. Jedná se o takzvaný hlavičkový soubor. Jeho obsah by měl být patřičně zdokumentován, hlavičkový soubor slouží jako rozhraní a musí být přístupný každému svému uživateli. [44]

4 Vlastní práce

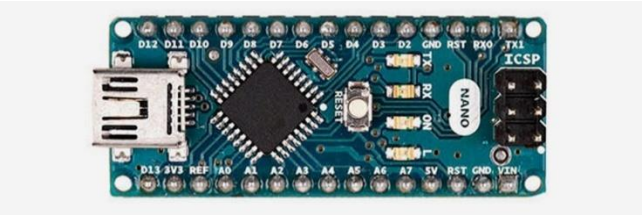
4.1 Výchozí stav

V rámci bakalářské práce byl vytvořen funkční prototyp vestavného informačního systému pro domácnost. Vše vznikalo za použití open source technologií, open source softwaru i hardwaru.

Hlavním účelem takového systému bylo usnadnit ovládání interiérových žaluzií prostřednictvím automatizace jejich chodu s možností manuální kontroly.

Cílem tedy bylo s volbou vhodných komponent vytvořit takové řešení, které bude schopno plně autonomního provozu. Dílčí cíle dále zahrnovaly zajištění provozu na akumulátor, minimalizaci rozměrů použitých komponent pro integraci do horní lišty žaluzií a dosažení maximálně energeticky úsporného provozu celku, s ohledem na kapacitu akumulátoru a jeho výdrž na jedno nabití.

Jako hlavní řídicí prvek systému byl zvolen mikrokontroler Arduino Nano, jež zobrazuje Obrázek 27. Důvodem této volby byly jeho miniaturní rozměry, fakt, že se jedná o open source hardware a také praktická zkušenost autora s programováním v jazyce C / C++.



Obrázek 27 Arduino Nano [46]

Ovládání náklonu žaluzií bylo možné řešit několika způsoby, od lineárních aktuátorů, přes miniaturní krokové motory, až po servomotory. Kvůli své jednoduchosti použití byl pro pohon zvolen servomotor (viz Obrázek 28).

Obrázek 28 Servomotor EMAX [47]

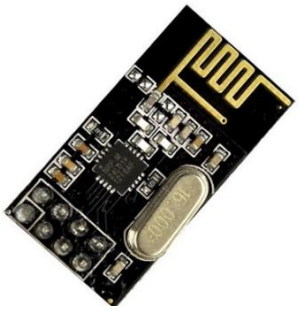
Napájení bylo vyřešeno pomocí čtyř kusů NiMH akumulátorů formátu AA, zapojených sériově. Při plném nabití tak jejich napětí v součtu nepřesáhlo 5,5 Voltu, tj. hraniční hodnotu, při které může být čip ATmega 328P ještě bezpečně provozován. Toto napětí zároveň již umožňuje provoz servomotoru, a tedy nebylo nutné použít step-down či step-up měnič, který by upravoval výstupní napětí akumulátoru.

Kvůli redukci spotřeby energie bylo nezbytné fyzicky vyřadit z provozu regulátor napětí, přítomný na Arduino desce, díky kterému je za normálních okolností možné mikrokontroler napájet napětím až 12 Voltů. Tento krok tedy znamenal omezení maximálního vstupního napájecího napětí, poskytovaného akumulátorem, na již zmíněných 5,5 Voltu. Lithium-iontové a lithium-polymerové akumulátory disponují nominálním napětím 3,7 Voltu na článek. To je příliš nízká hodnota pro provoz servomotoru, proto byl nakonec zvolen akumulátor typu NiMH s nominálním napětím 1,2 Voltu na článek. Při použití čtyř článků v sérii bylo dosaženo téměř ideální nominální hodnoty 4,8 Voltu. Další výhodu představovalo, že těmto akumulátorům nevadí tzv. hluboké vybití. Při dlouhodobém provozu tak není nutná pravidelná kontrola stavu nabití akumulátorů uživatelem, aniž by hrozilo jejich nevratné poškození.

Pro redukci spotřeby energie byla provedena také další opatření. Do napájecího obvodu mezi servomotor a akumulátor byl zařazen NPN tranzistor, prostřednictvím něhož byl motor odpojen od napájení po většinu času, kdy jím nebylo potřeba otáčet. Jinak by servo stále drželo nastavenou pozici za kontinuálního odběru proudu v řádu jednotek miliampér. Také byla fyzicky vyřazena z provozu LED dioda na desce, která indikuje zapnuté napájení Arduina, a kterou nelze softwarově vypnout. Nakonec bylo Arduino softwarově uvedeno do nejhlubšího z pěti možných režimů spánku, do tzv. „sleep mode – power down“ módu. V tomto režimu je vypnuta většina komponent Arduina až na piny pro obsluhu přerušení a tzv. watchdog timer, který se stará o periodické probouzení mikrokontroleru ze spánku. Nejdelší možný interval jednoho spánkového cyklu, který zde lze nastavit, je přibližně osm sekund. Pro účel práce to však bylo vyhovující, neboť kromě autonomního chodu byla zachována i možnost ovládat systém s relativně nízkou latencí manuálně, a současně během spánku klesla spotřeba celé sestavy až na 0,19 mA. Při kapacitě akumulátorů přesahující 2000 mAh to znamenalo teoretickou výdrž na jedno nabití přesahující jeden rok provozu.

Předposledním krokem bylo zajistit bezdrátovou komunikaci sestavy při zachování velmi nízkého odběru energie. K tomu byl využit bezdrátový modul NRF24L01+ od Nordic

Semiconductor (viz Obrázek 29). Ten pracuje v bezlicenčním pásmu 2,4 GHz a jeho hlavními výhodami jsou nízká cena a velmi úsporný provoz. Spotřeba modulu se pohybuje v řádu jednotek miliampér během komunikace, při nečinnosti pak v řádu jednotek mikroampér.



Obrázek 29 2,4 GHz modul NRF24L01+ [48]

Na závěr zbývalo zabudovat všechny prvky do makety okna, zkonstruované pro prezentaci k obhajobě práce, a zvolit vhodný informační systém pro interakci s uživatelem. Pro tento účel byl vybrán open source software OpenHab, který je určený pro ovládání nejrůznějších IoT prvků a domácí automatizaci.

4.2 Cílový stav

Jak bylo řečeno v samotném úvodu této práce, bezdrátový modul NRF24L01+ nebyl příliš vhodnou volbou, především z důvodu neexistence funkční softwarové knihovny pro Arduino, která by umožňovala provozovat tyto bezdrátové prvky v síti typu mesh. Bezdrátové transceivery IQRF mají komunikační mesh vrstvu již vyřešenu díky IQRF OS běžícímu v modulech a patentované technologii IQMESH, zajišťující směrování paketů v mesh síti. O této technologii bylo pojednáno v kapitole 3.3.4. Při vývoji obslužných rozhraní, tzv. DPA handlerů, však společnost IQRF Tech věnovala platformě Arduino pozornost spíše jen okrajově. V IQRF repozitáři se vzorovými obslužnými programy z kategorie *Custom DPA Handler* lze najít jen celkem dva soubory určené pro Arduino (soubory s příponou „.ino“). Těmi jsou *CustomDpaHandler-Bridge-UART.ino* a *CustomDpaHandler-Bridge-SPI.ino*, neboli handler pro obecnou komunikaci prostřednictvím rozhraní UART, resp. SPI. [32]

Ovládání servomotoru či čtení telemetrických dat však znamená nutnost zajistit, aby Arduino správně porozumělo rozličným příkazům, přicházejícím mesh sítí od koordinátoru. To prakticky představuje potřebu naprogramovat novou nadstavbu – knihovnu, vhodně doplňující základní programový kód pro Custom DPA Handler Bridge od IQRF.

4.2.1 Specifikace parametrů knihovny

Právě tvorba obslužného rozhraní je hlavním cílem jak této kapitoly, tak i celé práce. Aby mohl být tento cíl splněn, je nezbytné učinit potřebné dílčí kroky. Programování knihovny předchází příprava v podobě správného fyzického zapojení komponent, ověření funkčnosti komunikace, správného nastavení IQRF transceiverů, konstrukce IQMESH sítě, a také příprava vývojového prostředí pro Arduino.

Hlavní součástí knihovny by měla být funkce či sada funkcí, umožňující manipulaci se servomotorem připojeným na digitální pin mikrokontroleru, a to na základě zprávy odeslané z koordinátoru cílovému nodu. Rozhraní by mělo dále nabízet možnost čtení aktuální pozice servomotoru a její odeslání koordinátoru. Rozšířením knihovny by pak mohlo být čtení a odesílání informace o stavu nabití akumulátoru, pro možnost včasného upozornění systémem na blížící se nutnost jeho dobití.

Zároveň s dokončením programové části by měla být provedena také statická analýza celého kódu, který bude v mikrokontroleru spouštěn. K tomu lze využít funkcionality pro statickou analýzu, která je již přímo obsažena ve zvoleném vývojovém prostředí. Součástí výstupu kapitoly, věnující se samotnému vývoji, by tedy měly být výsledky statické analýzy kódu, poskytující představu o využitých prostředcích. Sledováno je využití RAM a flash paměti zařízení, spolu s detekovanými defekty v kódu, včetně vyhodnocení míry jejich závažnosti.

4.3 Fyzické zapojení komponent

IQRF transceiver byl s mikrokontrolerem Arduino propojen zprvu přes SPI sběrnici. Přes veškerou snahu se však nepodařilo komunikaci prostřednictvím SPI zprovoznit. Byla proto využita druhá možnost, která se nabízela – jednodušší rozhraní UART. Oběma uvedeným sběrnicím se věnuje kapitola 3.2.2.

4.3.1 **Pracovní zapojení nodu s Arduino Mega**

Jak již bylo zmíněno, propojení TR modulu s Arduino mikrokontrolerem je realizováno pomocí UART. Výhodu představuje snadné použití tohoto rozhraní, fyzicky je třeba jen propojit záporné póly obou zařízení (GND), dále připojit vysílací pin (TX) Arduina na přijímací pin (RX) transceiveru a naopak. V programové části pak stačí jedním příkazem zahájit komunikaci po příslušné sériové lince. Nevýhodou UART však je, že sériová linka, kterou využívá, je během komunikace blokována a nelze ji ve stejný okamžik použít například pro sériovou komunikaci s počítačem. Arduino Nano disponuje jediným UART rozhraním, není tedy možné komunikovat s TR modulem a zároveň sledovat výstup sériové linky kontroleru. Tento problém byl vyřešen přistoupením k použití mikrokontroleru Arduino Mega 2560, který poskytuje UART rozhraní celkem čtyři. První z nich slouží pro komunikaci přes USB, druhé bylo vybráno k propojení kontroleru s TR modulem.

Na modelu TR-72D nejsou přítomny pájecí kontakty a transceiver tak není možno připojit přímo. Je nutné použít převodník či vývojový kit, obsahující SIM konektor KON-SIM-02, do kterého se TR modul vkládá.

Pro účely vývoje a testování je přímo určený vývojový kit IQRF-DK-EVAL-04A (Obrázek 30 níže), byl proto zvolen pro většinu praktické části práce. Kit disponuje mimo jiné dobíjecím lithium-polymerovým 400mAh akumulátorem s možností dobíjení přes externí piny či mikro USB konektorem, dále dvěma tlačítky (SW1 a SW2), paticí KON-SIM-02 pro TR moduly a množstvím externích pinů, dovolujícím připojit k modulu různé druhy periferií.

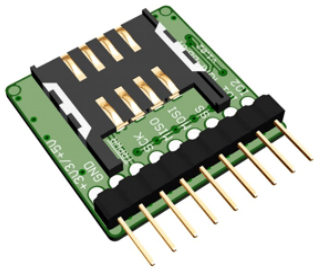
Obrázek 30 IQRF-DK-EVAL-04A [49]

Propojení Arduino Mega a TR modulu přes IQRF-DK-EVAL-04a je tedy provedeno následovně:

- pin *19* (RX1, Arduino) na pin *C5* (TX, TR modul),
- pin *18* (TX1, Arduino) na pin *C8* (RX, TR modul),
- pin *GND* (Arduino) na pin *C4* (GND, TR modul).

4.3.2 **Konečné zapojení nodu s Arduino Nano**

Pro propojení IQRF transceiveru a mikrokontroleru Arduino Nano je nezbytné použít IQRF Breakout board IQRF-BB-01 nebo IQRF-SHIELD-02, které obsahují převodník napětí. Důvodem je, že Arduino pracuje s 5V logikou a IQRF transceiver s 3,3V logikou, a proto je není možné propojit přímo.
Pro účely této práce byly zakoupeny dva kusy převodníku IQRF-BB-01:



Obrázek 31 Převodník IQRF-BB-01 [50]

Zapojení tohoto převodníku je obdobné jako v případě vývojového kitu:
- pin *19* (RX1, Arduino) na pin č. 3, označený *SS* (C5 – TX, TR modul),
- pin *18* (TX1, Arduino) na pin č. 5, označený *MISO* (C8 – RX, TR modul),
- pin *GND* (Arduino) na pin č. 8, označený *GND* (C4, TR modul).

Po dokončení potřebných nastavení, naprogramování obslužného rozhraní a zprovoznění komunikace v pracovním zapojení bude pro konečný provoz použit právě tento převodník.

4.3.3 **Pracovní zapojení koordinátoru**

Druhý IQRF modul – koordinátor – je zapojen do USB programátoru IQRF-CK-USB-04a. Ten je až na absenci akumulátoru vizuálně jinak velmi podobný vývojovému kitu DK-EVAL-04a. Komunikace s počítačem probíhá přes USB a s modulem standardně pomocí SPI rozhraní.

4.3.4 **Konečné zapojení koordinátoru**

Pro běžný provoz bude nezbytná tzv. IQRF brána (gateway). Lze ji buď zakoupit jako hotový produkt, případně ji vytvořit propojením IQRF transceiveru v roli koordinátoru a počítače s obslužnou aplikací. IQRF gateway bude vyrobena svépomocí za použití jednodeskového minipočítače Raspberry Pi s běžícím operačním systémem Linux (Raspberry Pi OS). Z fyzického pohledu se na pole GPIO pinů na Raspberry Pi pouze nasadí adaptér IQRF-KON-RASP-01 (viz Obrázek 32) s konektorem KON-SIM-02, do kterého se zasune transceiver.

Obrázek 32 Adaptér IQRF-KON-RASP-01 [51]

4.3.5 **Pracovní zapojení servomotoru**

Servomotor disponuje celkem třemi vodiči – dvěma pro napájení a jedním signálním pro ovládání serva. Kladný napájecí vodič má barvu obvykle červenou, záporný je většinou černé či hnědé barvy. Signálový vodič bývá žlutý či bílý.
Napájecí konektory servomotoru jsou připojeny na příslušné piny Arduina, tj. kladný *+5V* a záporný *GND*. Konektor signálového vodiče obsadil digitální pin *D9*.

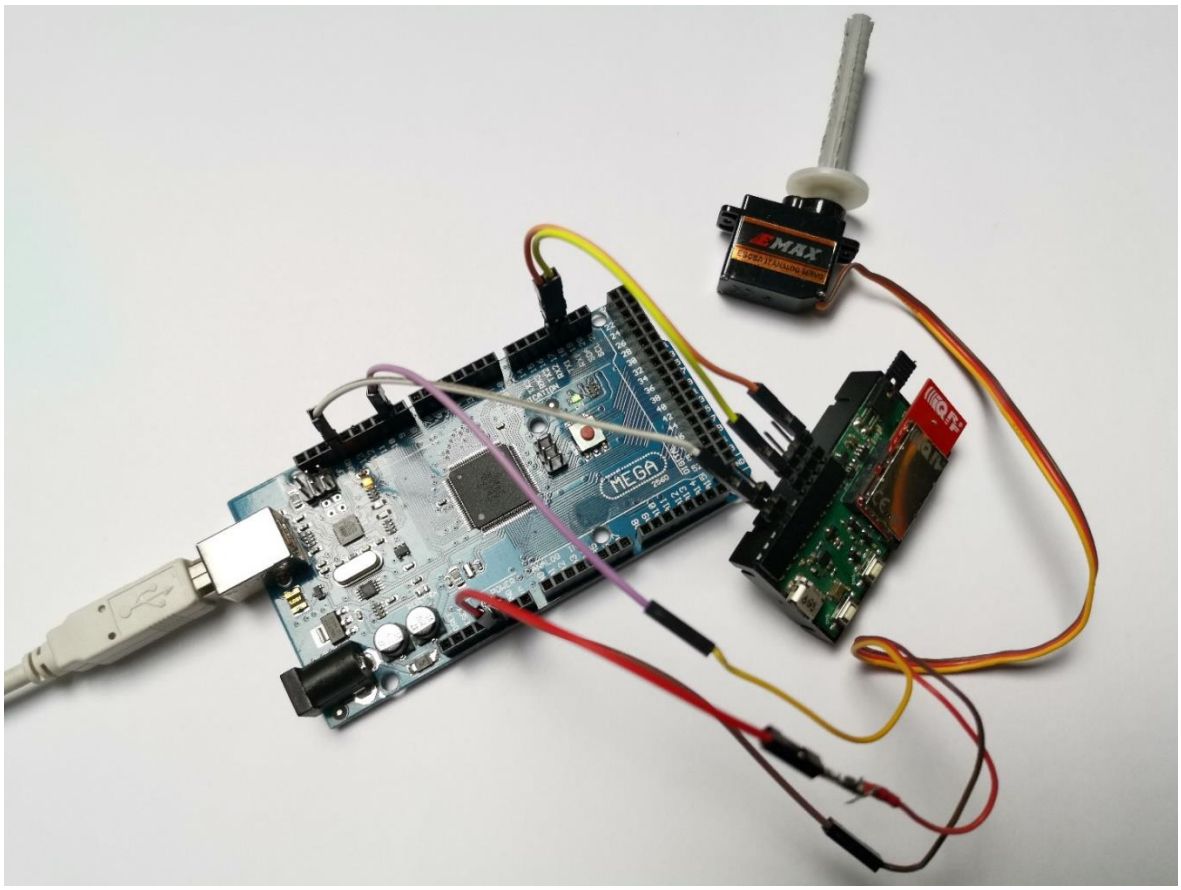
4.3.6 **Konečné zapojení servomotoru**

Zapojení v provozním režimu na Arduino Nano bude u servomotoru velice podobné současnému pracovnímu zapojení. Pro ovládání může být však zvolen jiný řídicí pin, na který bude připojen signálový vodič serva. Největší rozdíl bude představovat zapojení kladného vodiče, který musí být pro běžný provoz připojen přímo ke zdroji (akumulátoru) přes tranzistor, kterým bude přívod energie pro servo po dobu nečinnosti vypínán. Toto

opatření je nezbytné kvůli jinak vysoké spotřebě energie, kontinuálně odebírané motorem. Tato problematika je podrobněji řešena ve výsledcích práce.

4.3.7 **Sestava pro vývoj a testy**

Programování obslužného rozhraní probíhá na sestavě, skládající se z celkem tří klíčových komponent: mikrokontroleru Arduino Mega 2560, IQRF modulu TR-72DA v roli nodu, zasunutém ve vývojovém kitu IQRF-DK-EVAL-04a, a mikroserva EMAX. Všechny tři komponenty jsou propojeny tak, jak je v podkapitole 4.3 uvedeno. Pro lepší představu celou sestavu zachycuje následující Obrázek 33.



Obrázek 33 Sestava komponent pro vývoj: Arduino Mega, IQRF transceiver a servomotor [autor]

4.4 Tvorba obslužného rozhraní

Po správném propojení všech nezbytných komponent lze již přistoupit k vlastní práci na tvorbě samotné knihovny.

4.4.1 Vývojové prostředí

Aby mohlo obslužné rozhraní vzniknout, je nezbytné mít nainstalován potřebný software. Na straně koordinátoru, tj. řídicí straně, je tímto softwarem IQRF IDE. To je používáno jednak pro ověřování během programování rozhraní, zda koordinátoru přichází od nodu správná data, ale také je zde provedeno počáteční nastavení obou transceiverů a je do nich nahrán potřebný firmware a software.

Na straně nodu, tj. straně řízené, je používáno vývojové prostředí Microsoft Visual Studio Code. S použitím volitelného rozšíření PlatformIO se z Visual Studio Code stává vývojová platforma pro embedded (vestavná) zařízení, jakým je právě Arduino. Tato kombinace softwaru byla upřednostněna před Arduino IDE především na základě dodatečných funkcionalit a výhod, které oproti tomuto IDE nabízí, a které jsou blíže popsány v kapitole teoretické části práce.

4.4.1.1 IQRF IDE

Po instalaci vývojového prostředí IQRF IDE je z webových stránek IQRF [52] stažen zip archiv s balíčkem IQRF Startup Package pro aktuální verzi operačního systému IQRF OS 4.04D. Po jeho rozbalení je v IQRF IDE otevřen vzorový projekt s názvem *IoT-StarterKit-01-demo*. Ten by měl obsahovat správná přednastavení, co se týče používaných knihoven a dalších součástí projektu, a měl by tak být vhodný pro začátky s platformou IQRF.

4.4.1.2 Visual Studio Code a PlatformIO

Po nainstalování Visual Studio Code a jeho rozšíření PlatformIO bylo také nutné nainstalovat podporu pro platformu Atmel AVR [53] a Arduino knihovnu *Servo* [54] pro pozdější ovládání servomotoru mikrokontrolerem. Poté již bylo možné založit nový PlatformIO projekt – zvolit ze seznamu cílový hardware, pro který se bude vyvíjet, vybrat umístění v počítači, kde bude projekt uložen, a nakonec projekt pojmenovat. Název projektu byl zvolen příznačně: *IQRF_servo_control*.
Prvním důležitým souborem projektu je *platformio.ini* – zde je obsažena základní konfigurace projektu. Obsahu souboru v tomto projektu je následující:

```
; PlatformIO Project Configuration File
```

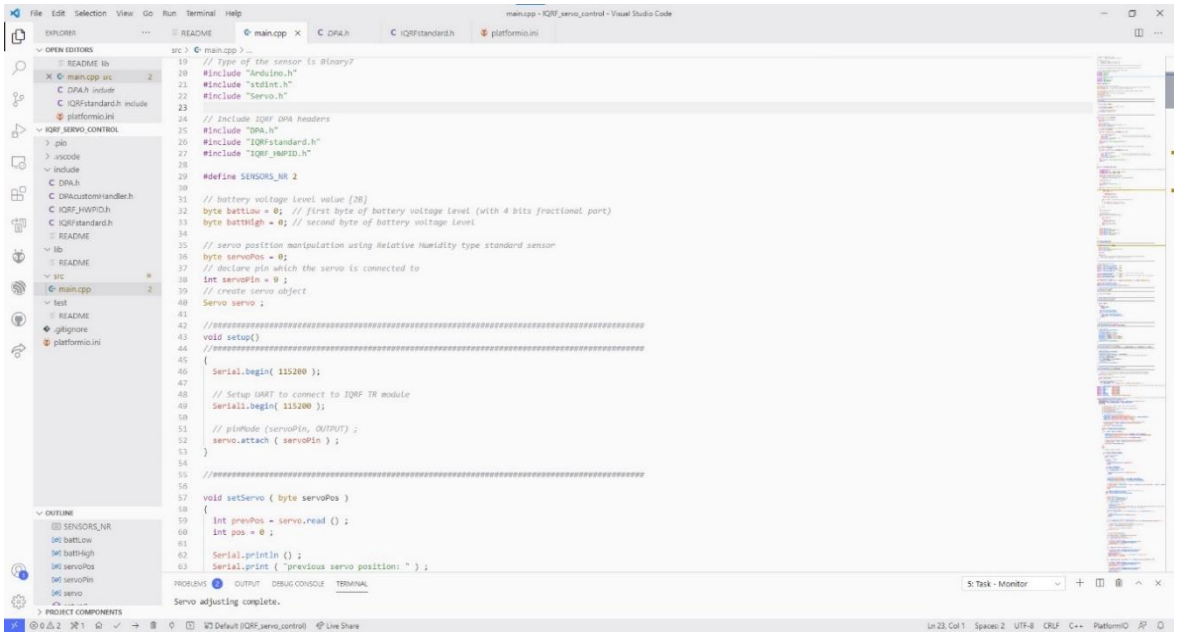
```
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:megaatmega2560]
platform = atmelavr
board = megaatmega2560
framework = arduino
monitor_speed = 115200
lib_deps = arduino-libraries/Servo@^1.1.7
```

Obsah souboru platformio.ini se vytvořil automaticky, spolu se založením projektu. Vyplněna tedy byla platforma, HW deska (board) a framework, odpovídající volbě při zakládání projektu. Rychlost komunikace po sériové lince (*monitor_speed*, v baudech) se předvyplnila na výchozí hodnotu 115200 baud. Poslední položkou jsou závislosti v podobě vložených knihoven (*lib_deps*). Tato položka se mění také automaticky, podle toho, jaké knihovny jsou v průběhu práce do projektu přidávány. Zde je již vidět knihovna pro ovládání serva, která byla přidána jako první.

Dalším stěžejním souborem je *main.cpp*, nacházející se uvnitř projektu v podadresáři *src*. Zde jsou umisťovány i případné další zdrojové soubory projektu. Soubor *main.cpp* je však jakýmsi základním stavebním kamenem projektu, obsahuje pro Arduino klíčové funkce *setup* a *loop* a vždy se spouští jako první.

V projektovém podadresáři *lib* pak jsou umisťovány specifické (privátní) knihovny, které je v projektu třeba použít.

Obrázek 34 zobrazuje rozšíření PlatformIO ve Visual Studio Code IDE. Po levé straně se nachází menu s adresářovou strukturou projektu.



Obrázek 34 Visual Studio Code a PlatformIO [autor]

4.4.2 Konfigurace IQRF transceiverů

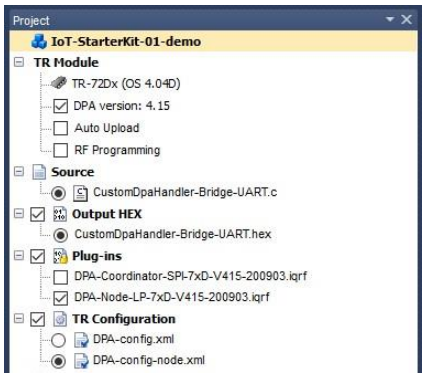
Po korektním fyzickém zapojení obou zařízení, tedy připojení IQRF programátoru a Arduino Mega USB kabelem k počítači, propojení IQRF vývojového kitu a servomotoru s Arduino Mega, je možné zahájit konfiguraci obou transceiverů.

Při vkládání modulu do programátoru či jeho vyjímání je nutné mít programátor buď odpojen od USB nebo během manipulace držet stisknuté tlačítko *SW2* (z obou tlačítek to, které se nachází blíže SIM konektoru), čímž dojde k odpojení napájení modulu a nehrozí tak jeho poškození zkratováním. Podobný postup platí i pro vývojový kit, u kterého lze buď odpojit napájení baterie či opět vyřadit napájení modulu pomocí tlačítka *SW2*.

4.4.2.1 Konfigurace nodu

Jako první je nastavován node, který bude připojen k Arduinu. V sekci Project musí být u „TR Module“ zvoleno DPA verze 4.15. U „Source“ je nutné přidat zdrojový soubor *CustomDpaHandler-Bridge-UART.c* a mít jej vybrán. Dále je třeba zaškrtnout „Output HEX“, aby došlo k nahrání výstupního HEX souboru, který se kompilací vytvoří ze zdrojového souboru. U „Plug-ins“ části je nutné zvolit *DPA-Node-LP-7xD-V415-200903.iqrf*. Posledním krokem je nastavení parametrů pro transceiver v „TR Configuration“. Zde je třeba po otevření *DPA-config.xml* v samostatném okně v záložce

„DPA“ zaškrtnout „Custom DPA Handler“ a v záložce „Security“ nastavit Access Password. Ostatní nastavení lze ponechat na předdefinovaných hodnotách. Toto nastavení je vhodné uložit pod novým názvem pro případnou budoucí potřebu dalšího použití. Nyní je vše připraveno pro kompilaci a nahrání souborů do paměti modulu. Úspěšně dokončené nahrávání se projeví kromě zprávy na spodní stavové liště i zvukovým signálem. Poté je možné modul z programátoru vyjmout a vložit modul druhý. Níže (Obrázek 35) lze vidět správnou konfiguraci pro node.

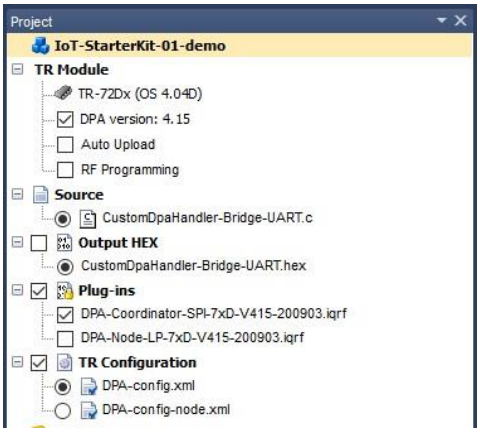


Obrázek 35 Konfigurace IQRF nodu [autor]

4.4.2.2 **Konfigurace koordinátoru**

Druhý modul je v roli koordinátoru, jeho nastavení částečně kopíruje předchozí nastavení nodu. Je však nutné, aby volba „Output HEX“ nebyla označena. Dále v „Plug-ins“ je třeba zvolit *DPA-Coordinator-SPI-7xD-V415-200903.iqrf* a v „TR Configuration“ v *DPA-config.xml* nemít označenu volbu „Custom DPA Handler“. Na závěr je nezbytné nastavit stejné Access Password, jako u nodu. Po úspěšném nahrání takto připravené konfigurace do modulu jej lze v programátoru ponechat. Od tohoto okamžiku je modul v roli koordinátoru a s využitím IQRF IDE z něj budou odesílány příkazy nodu. Níže (Obrázek 36) lze vidět správnou konfiguraci platnou pro koordinátor.

Obecně je důležité dbát na to, aby měly všechny transceivery v rámci jedné mesh sítě stejnou verzi IQRF OS, shodnou verzi DPA protokolu a nastavené totožné Access Password.



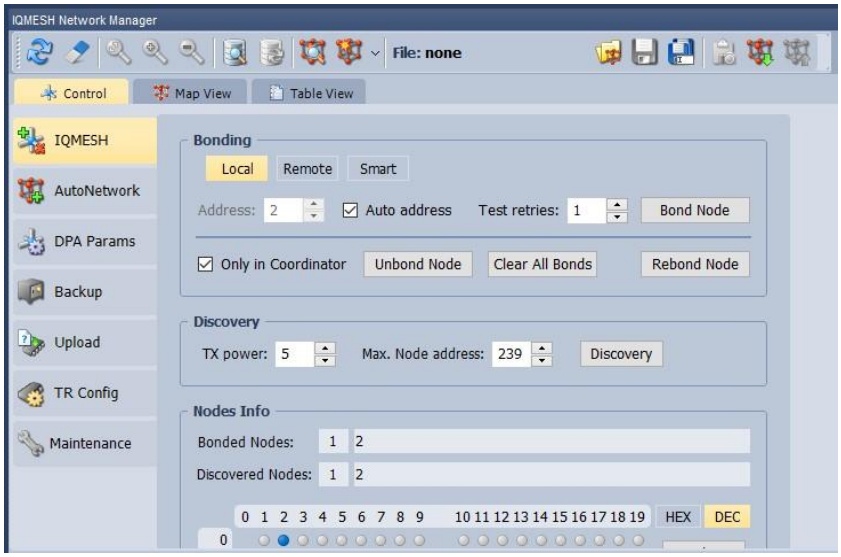
Obrázek 36 Konfigurace IQRF koordinátoru [autor]

4.4.2.3 **Konfigurace IQMESH sítě**

Jakmile je modul v roli nodu vložen do vývojového kitu, lze přistoupit ke konfiguraci mesh sítě.

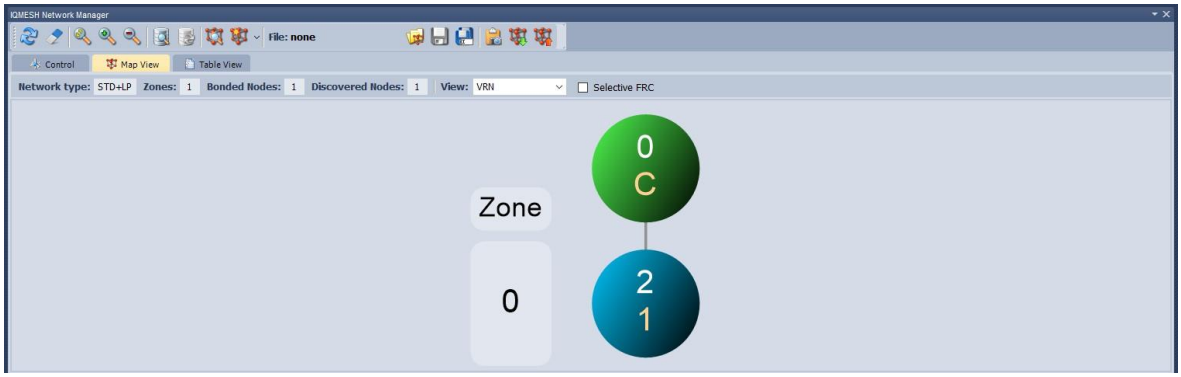
Koordinátor se v okně *IQMESH Network Manager* objeví okamžitě, je však v síti prozatím sám, node je nutné do sítě přiřadit.

Připojení nodů do sítě probíhá v záložce „Control“ v sekci IQMESH. Nejrychlejší je připojit node lokálně. K tomu slouží volba „Local“ v části „Bonding“. Zde stačí mít aktivní volbu „Auto address“ a po stisku „Bond Node“ je zahájen proces připojování. Nyní je nutné krátce stisknout tlačítko *SW1* (tlačítko blíže USB konektoru, v rohu vývojového kitu). Pokud má node nastavené stejné parametry sítě a Access Password, jako koordinátor, měl by být úspěšně připojen. Lokální bonding je zobrazen níže (Obrázek 37).



Obrázek 37 IQMESH node bonding [autor]

Nalezený node se nyní objeví v grafickém náhledu sítě pod záložkou „Map View“. Dále nezbytné vykonat tzv. *Discovery* s enumerací sítě – „Enumerate (full)“. Tím je vybudována směrovací struktura sítě. [55] Koordinátor si tímto způsobem také zjistí o připojených nodech všechny dostupné informace a přiřadí jim síťové adresy *Virtual Routing Number* (Obrázek 38).



Obrázek 38 IQMESH síť pro vývoj knihovny [autor]

4.4.3 Použité Arduino knihovny

Následující knihovny jsou pro vývoj rozhraní nezbytné a jsou v programu využity.

4.4.3.1 "Arduino.h"

Jedná se o základní knihovnu pro Arduino SDK. Obsahuje include definice pro vložení některých dalších často používaných knihoven, jako například *string.h* pro práci s řetězci,

math.h pro matematické operace či knihovnu *stdlib.h*. Dále jsou zde přítomny některé knihovny typické pro platformu AVR, jako je *avr/io.h* pro vstupně-výstupní operace či *avr/interrupt.h* pro obsluhu přerušení. Knihovna *Arduino.h* také definuje řadu specifických konstant a operací. [56]

4.4.3.2 **"stdint.h"**

Knihovna *stdint.h* obsahuje definice celočíselných datových typů a s nimi souvisejících konstant. [57]

4.4.3.3 **"Servo.h"**

Knihovna *Servo.h* poskytuje mikrokontroleru možnost ovládat servomotory z kategorie hobby. Jedná se převážně o modelářská serva. Zahrnuje obnovovací intervaly či definice minimální i maximální šířky pulzu, kterou lze pro ovládání serva použít. Obsahuje také funkce pro přímou manipulaci se servomotorem. [54]

4.4.3.4 **"DPA.h"**

Jde o obecnou knihovnu umožňující použití protokolu Direct Peripheral Access. [32]

4.4.3.5 **"IQRFstandard.h"**

Knihovna *IQRFstandard.h* obsahuje definice všech typů standardních senzorů, které IQRF aktuálně specifikuje. [32]

4.4.3.6 **"IQRF_HWPID.h"**

Zde jsou obsaženy jednotlivé identifikátory HW profilů (HWPID), používaných k jednoznačné specifikaci funkcionalit zařízení, uživatelských periferií, specifikaci jejich chování, a podobně. [32]

4.4.4 **Ověření funkce, čtení sériové linky**

Prvním krokem po přípravě prostředí, zprovoznění modulů a mesh sítě, je ověření správné funkce. Nejprve proběhl pokus rozsvítit červenou LED diodu na nodu. Po odeslání příkazu

z koordinátoru se skutečně krátce rozsvítila a v Packet Inspectoru se objevila potvrzující odpověď od nodu. Komunikace mezi oběma prvky v IQMESH síti tedy probíhá správně.

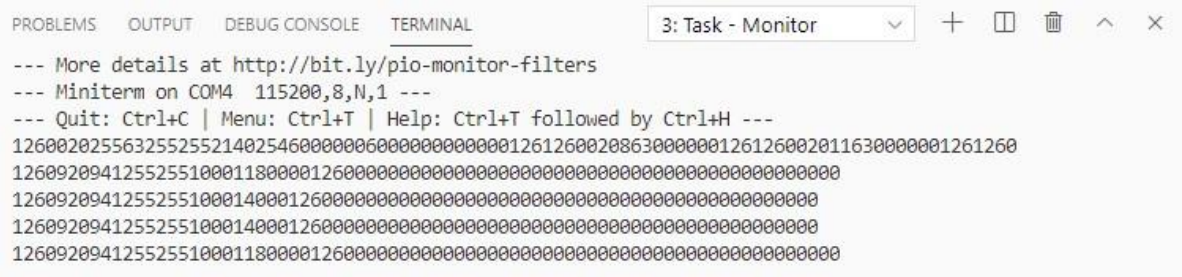
Následně je třeba zjistit, zda mikrokontroleru přichází zprávy po sériové lince. Node by měl touto cestou vyslat zprávu přinejmenším během discovery procesu, případně po obdržení některých dalších specifických typů zpráv z koordinátoru. Pro tento účel byl vytvořen triviální program, jež je pro ukázku níže vložen. Ten v *setup* části pouze nastaví a zahájí komunikaci po obou sériových linkách – „Serial“ pro USB komunikaci s PC a „Serial1“ pro UART připojení k nodu. V *loop* smyčce pak čte obsah sériové linky a ukládá jej do pole bajtů o velikosti 64 B. Tato velikost byla zvolena kvůli maximální možné délce zprávy, která činí právě 62 B. [32] Pokud pole obsahuje také jiné číslo než jen nuly, obsah je vypsán na sériovou linku počítače, a tedy zobrazen v sériové konzoli.

Program pro čtení sériové linky:

```
#include <Arduino.h>

void setup()
{
  Serial.begin (115200);  // setup serial interface for USB
  Serial1.begin (115200); // setup serial interface for IQRF
}

void loop()
{
  byte arr[62]; // maximum message length is 62 bytes
  bool flag = false; // flag for a serial status indication

  for ( int i = 0; i < 62; i++ )  // zero the array
    arr[i] = 0;

  Serial1.readBytes ( arr, 62
);  // read all available information from node via
  UART and copy it to the array

  for ( int i = 0; i < 62; i++ )  // look for non-
zero bytes inside the array
    if ( arr[i] != 0 ) flag = true;  // if any byte is non-
zero, continue to print

  if ( flag == false ) return ;  // if nothing came, restart the loop

  for ( int i = 0; i < 62; i++ )  // print the content to the USB serial
    Serial.print ( arr[i] );
```

```
  Serial.println ();
  delay (1000);  // wait 1 second
}
```

Po úspěšné kompilaci a nahrání programu do mikrokontroleru byla v počítači otevřena sériová konzole a zároveň z koordinátoru spuštěna discovery procedura. V terminálu byla téměř okamžitě vypsána příchozí zpráva. Toto bylo následně ještě zopakováno i s jinými druhy požadavků. Záznam sériové linky je zachycuje Obrázek 39.



Obrázek 39 Čtení dat ze sériové linky v PlatformIO IDE [autor]

4.4.5 Vlastní knihovna (komunikační rozhraní)

Obslužné rozhraní je tvořeno za využití vzorového programu pro Arduino od IQRF – *CustomDpaHandler-Bridge-UART.ino*, kde je již obsažena základní komunikační logika a prvky specifické pro DPA protokol.

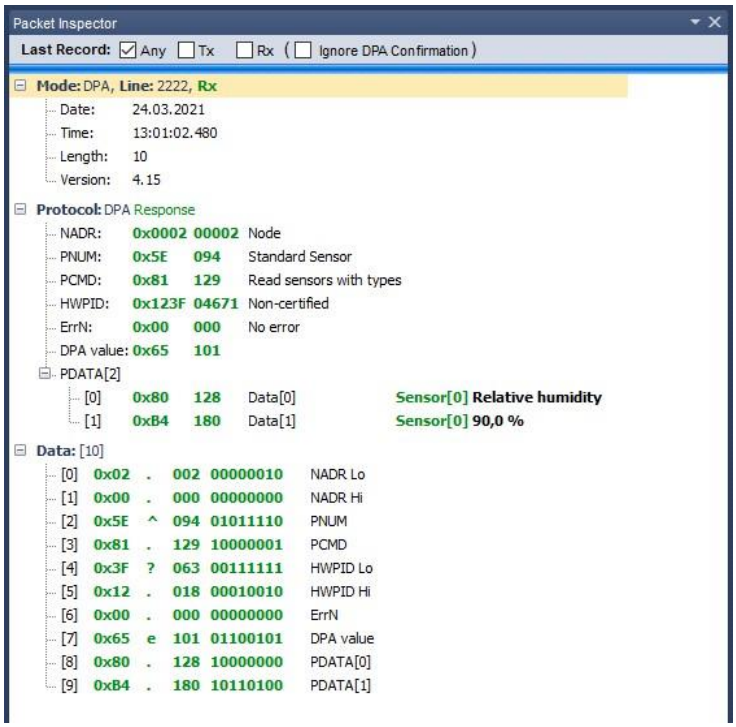
Práce je nejprve tvořena v jednom *main.cpp* souboru společně s výše zmíněným kódem od IQRF. Následně po zprovoznění potřebných funkcí dochází k transformaci vlastního kódu do podoby knihovny ve formě samostatných zdrojových a hlavičkových souborů.

Důležité části kódu jsou zde z důvodu vysvětlení jejich funkce vloženy. Samozřejmostí je přítomnost komentářů v anglickém jazyce přímo v kódu. Ten tak bude moci být po drobných úpravách zveřejněn na GitHub pro možné využití či rozšíření knihovny komunitou kolem Arduino.

4.4.5.1 IQRFservo.cpp a IQRFservo.h

Knihovna *IQRFservo* je nejdůležitějším prvkem celé práce. Je zde implementována funkce řízení servomotoru. Knihovna obsahuje také čtení nastaveného úhlu. To může být užitečné v klientské aplikaci pro zobrazení aktuálních stavů jednotlivých oken.

Pro účely ovládání i vyčítání stavu serva je využita IQRF standardní periferie pro relativní vlhkost – *Relative Humidity*. Její velikost činí 1 B, kompletní specifikaci lze nalézt v kapitole 3.3.7. Obrázek 40 zachycuje výstup z Packet Inspectoru, kde je tato periferie vidět. Příchozí hexadecimální hodnota *0xB4* je dekadických 180 a představuje maximální možný nastavený úhel servomotoru. IQRF IDE tuto hodnotu interpretuje jako 90 %. Tento typ senzoru využívá rozlišení 0,5 %, což je zde jedna dekadická jednotka, 100 % by tedy bylo dekadicky 200.



Obrázek 40 IQRF Pacet Inspector – čtení pozice servomotoru [autor]

Hlavičkový soubor *IQRFservo.h* má tuto podobu:

```
// include necessary libraries
#include "Servo.h"
#include "stdint.h"
#include "Arduino.h"

class IQRFservo
{
```

```
private:

public:
    IQRFservo  ();
    ~IQRFservo ();

    // initialise the servo
    void init     ( int pinNr );
    // turn servo to specified position at a certain speed
    void setAngle ( byte angle, int delayTime );
    // returns current angle set
    byte getAngle ();

    Servo servo;
};
```

Jak je patrné, všechny důležité metody pro obsluhu servomotoru jsou členy třídy *IQRFservo*. Níže se nachází implementace těchto metod tak, jak je obsažena v souboru *IQRFservo.cpp*.

```
void IQRFservo::init ( int pinNr )
{
  this -> servo.attach ( pinNr ) ;
}
```

Metoda *init* je volána na počátku běhu programu, v části *setup*. Jejím vstupním parametrem je celočíselná hodnota, představující číslo pinu, na který má být servo připojeno. V těle metody dojde k připojení servomotoru na příslušný pin. Metoda nevrací žádná data.

```
void IQRFservo::setAngle ( byte angle, int delayTime )
{
  // determine current position of the servo
  int prevPos = getAngle () ;
  int i = 0 ;

  // if the requested position is higher than the current one, increase angl
e
  if (prevPos < angle)
  {
    // tell servo to go to position given by the variable 'angle'
    for (i = prevPos; i <= angle; i += 1)  // in steps of 1 degree
    {
      this->servo.write (i);
      delay ( delayTime );  // wait "delayTime" for the servo to reach the p
osition
    }
    return ;
  }
```

```
  // if the requested position is lower than the current one, decrease angle
  if (prevPos > angle)
  {
    // tell servo to go to position given by the variable 'angle'
    for (i = prevPos; i >= angle; i -= 1)  // in steps of 1 degree
    {
      this->servo.write (i);
      delay ( delayTime );  // wait "delayTime" for the servo to reach the p
osition
    }
    return ;
  }
}
```

Účelem metody *setAngle* je nastavit požadovaný úhel servomotoru. Jejími vstupními parametry jsou jednobytová hodnota požadovaného úhlu a celočíselná hodnota nastaveného zpoždění, ovlivňujícího rychlost otáčení.

V této metodě je využívána i další třídní metoda, *getAngle*. Ta má za úkol zjistit aktuální polohu, ve které se servo nachází. Poloha je uložena do proměnné *prevPos*, která je využita dále při otáčení motorem.

Pokud je požadovaný úhel vyšší než současná pozice serva, je vykonán kód uvnitř první podmínky. V cyklu, běžícím od aktuální polohy až po požadovanou, je po jednotkových krocích úhel zvyšován až do dosažení cílové hodnoty. Rychlost otáčení lze řídit díky parametru uvnitř vložené funkce *delay*. Čím vyšší je hodnota *delayTime*, tím pomaleji se servo otáčí, neboť se zvyšuje časová prodleva mezi jednotlivými inkrementy.

V případě, že je požadovaný úhel naopak nižší než aktuální, probíhá vše obdobně, pouze opačným směrem – hodnota úhlu motoru se postupně dekrementuje.

```
byte IQRFservo::getAngle ()
{
  return this->servo.read () ;
}
```

Metoda *getAngle* vrací aktuání pozici servomotoru, která byla přečtena metodou *read* z knihovny *Servo*. Tato návratová hodnota je jednobytová (datový typ *byte*).

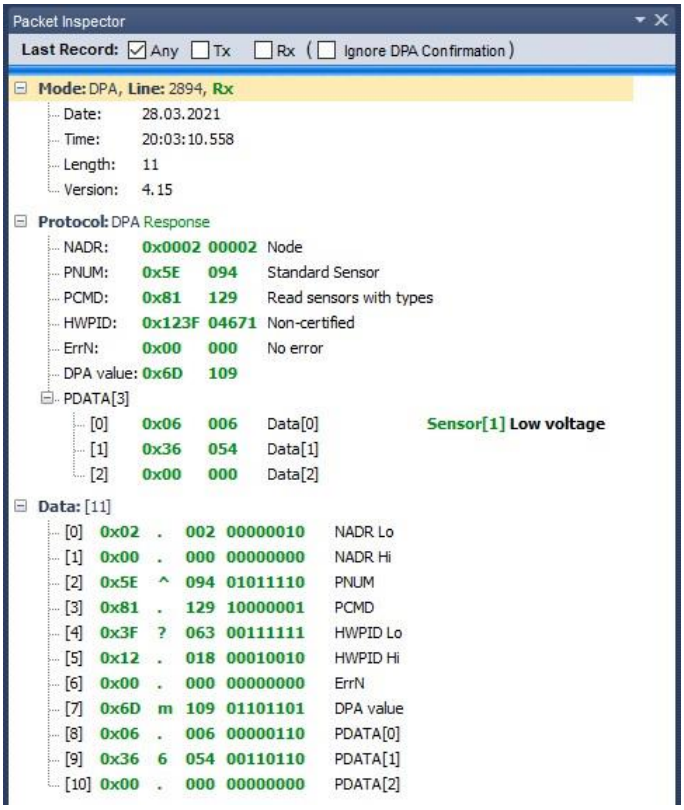
4.4.5.2 IQRFbattery.cpp a IQRFbattery.h

Knihovna *IQRFbattery* představuje rozhraní pro čtení úrovně nabití akumulátoru.

Lze ji v budoucnu rozšířit o další metody, například, pokud by bylo třeba odesílat příkazy pro nastavení vybíjecí charakteristiky akumulátoru či pro jinou interakci s tímto rozhraním. V současné době je hodnota naměřeného napětí akumulátoru simulována prostou číselnou hodnotou (během tvorby rozhraní nebyl akumulátor k mikrokontroleru připojen).

Existence této knihovny je důležitá pro zajištění uživatelského komfortu při dlouhodobém provozu celého systému. Jejím hlavním přínosem je umožnění průběžně monitorovat napětí připojeného akumulátoru. Při poklesu napětí pod určitou definovanou mez pak lze v informačním systému nastavit, aby určitým způsobem reagoval, například upozorněním uživatele přímo v aplikaci, e-mailem či SMS zprávou na skutečnost, že akumulátor v konkrétním okně bude brzy vybitý.

Knihovna využívá IQRF standardní periferii *Low Voltage*, která je pro přenos informace o elektrickém napětí určena. Obrázek 41 tuto periferii zobrazuje v Packet Inspectoru.



Obrázek 41 IQRF Packet Inspector – čtení úrovně nabití akumulátoru [autor]

Hlavičkový soubor *IQRFbattery.h* má tuto podobu:

```
// include necessary libraries
```

```
#include "stdint.h"
#include "Arduino.h"

class IQRFbattery
{
private:
    // format measured voltage to the IQRF-
specific LowVoltage type of sensor [2B]
    void formatLowVoltage  () ;

public:
    IQRFbattery  ();
    ~IQRFbattery ();

    // performs a battery voltage check and stores its value
    void checkBatteryLevel () ;

    // measured voltage [in mV]
    unsigned short batt ;

    // IQRF format battery voltage level value [2B]
    byte battLow ;  // first byte of battery voltage level (with 4 bits frac
tional part)
    byte battHigh ; // second byte of battery voltage level
};
```

Níže se nachází implementace těchto metod tak, jak je obsažena v souboru *IQRFbattery.cpp*.

```
IQRFbattery::IQRFbattery ()
{
    batt = 0;
    battLow = 0;
    battHigh = 0;
}
```

V konstruktoru *IQRFbattery* jsou inicializovány všechny tři třídní proměnné.

```
void IQRFbattery::checkBatteryLevel ()
{
  batt = 3375;

  formatLowVoltage () ;
}
```

Metoda *checkBatteryLevel* nemá žádné vstupní parametry ani návratovou hodnotu. V současné době obsahuje jen simulovanou naměřenou hodnotu napětí (nastaveno je aktuálně 3,375 V). Jednotkou této veličiny je 1 mV, hodnota jednoho Voltu by tedy byla

zapsána jako „1000“. Dále zajišťuje volání privátní třídní metody *formatLowVoltage* pro převod do formátu pro IQRF. Pro provoz zařízení v běžném režimu se zapojeným akumulátorem bude nutné tuto metodu doplnit o měření napětí.

```
void IQRFbattery::formatLowVoltage  ()
{
  // written using some info about bit operations from: https://www.codesdope.com/blog/article/set-toggle-and-clear-a-bit-in-c/
    unsigned short i;
    unsigned short low = (batt % 1000) / 62.5;
    unsigned short high = batt / 1000;

    // integral part
    int x = 7;

    for (i = 1 << 11; i > 0; i = i / 2)
    {
        if ( i > 8 )  // write all 8 bits of upper byte
          if ( high & i )
            battHigh |= 1 << x;

        if ( i <= 8 ) // write upper 4 bits of lower byte
          if ( i == 8 ) x = 7;  // re-set x
            if ( high & i ) battLow |= 1 << x;

        x--;
    }

    // fractal part
    for ( i = 1 << 3; i > 0; i = i / 2 )
    {
        if ( low & i )
          battLow |= 1 << x ;

        x--;
    }
}
```

Podstata metody *formatLowVoltage* tkví v převodu naměřené hodnoty napětí do formátu, který definuje IQRF Standard Sensor pro typ senzoru *Low Voltage*. Jeho hodnota má velikost 2 B, z toho 4 spodní bity jsou vyhrazeny pro desetinnou část. Specifikace tohoto senzoru je také součástí kapitoly 3.3.7. Základem této metody jsou bitové operace, před jejichž vykonáním se hodnota napětí rozdělí do dvou proměnných, *low* a *high*. Původní hodnota napětí má maximální velikost 2 B a tedy 16 bitů.

Proměnná *low* obsahuje pouze desetinnou část přepočtenou na šestnáctiny Voltu (rozlišení tohoto typu senzoru dle specifikace). To představuje informaci o velikosti 4 bity. V proměnné *high* je uložena celočíselná část převedená na Volty. Proměnná je typu *unsigned short*, tedy 16bitová, použito z ní však je pouze spodních 12 bitů.

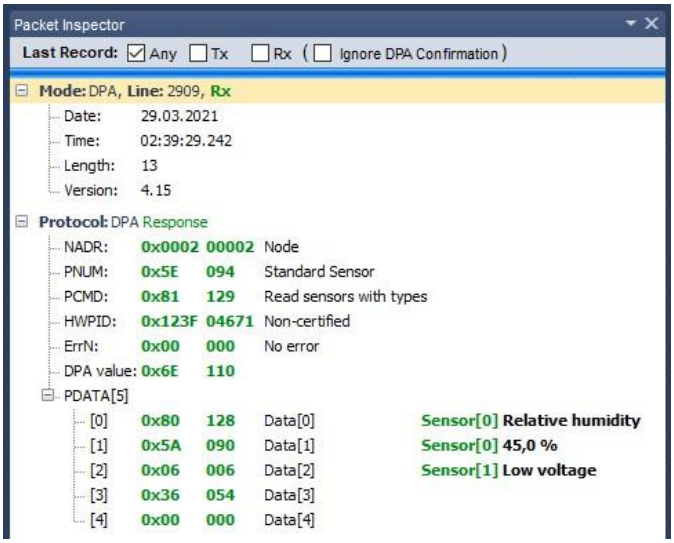
Následně je provedena řada bitových operací. Do proměnné *battHigh* jsou zapsány bity 4–10 proměnné *high* (nejvyšší 11. bit je vyhrazen znaménku). Spodní 4 bity proměnné *battLow* však musí korespondovat se spodními 4 bity proměnné low, a její horní 4 bity musí být doplněny spodními 4 bity (bity 0–3) proměnné *high*.

Příklad funkce:

Proměnná *batt* obsahuje číslo *3375*. Do proměnné *low* je tedy uloženo *6* (3375 mod 1000 = *375* a 375 / 62,5 = *6*). V Proměnné *high* je uloženo číslo *3* (celočíselná hodnota po dělení 3375 tisícem). Číslo 3 má binární podobu v jednom bytu *0000 0011*. Číslo 6 lze binárně zapsat jako *0000 0110*. Výsledná dvoubytová hodnota napětí splňující IQRF standard je tedy *0000 0000 0011 0110*. Toto číslo je v hexadecimální podobě rovno 0x36, či 54 dekadicky. Shoduje se tak s přijatou hodnotou *Data[1]* u *PDATA* (Obrázek 41), kde je zobrazen výpis IQRF Packet Inspectoru po odeslání dotazu na hodnotu senzoru *Low Voltage*.

4.4.5.3 **main.cpp**

Hlavní program *main.cpp* je tvořen na základě vzorového IQRF kódu pro Custom DPA Handler pro UART rozhraní. Obrázek 42 ukazuje funkční výčet senzorů včetně jejich aktuálních hodnot.



Obrázek 42 Packet Inspector – výčet senzorů [autor]

Bloky kódu zobrazené níže jsou kombinací původního kódu IQRF a použitím nových knihovních funkcí. První blok představuje začátek funkce *main* a obsahuje nejprve *include* direktivy pro jednotlivé knihovny, následně globální objekty a proměnné, nezbytné pro správnou funkci obou nově implementovaných typů senzoru, a nakonec funkci *setup*.

Inicializace

```
// include necessary core libraries
#include "stdint.h"
#include "Arduino.h"

// Include IQRF DPA headers
#include "DPA.h"
#include "IQRFstandard.h"
#include "IQRF_HWPID.h"

// include own created libraries
#include "IQRFservo.h"
#include "IQRFbattery.h"

// number of sensors currently in use (for future use - to be implemented)
#define SENSORS_NR 2
//###############################################################################
#######
// servo - specific declarations:

// IQRF-specific type of servo object
IQRFservo servo ;
// Declare a pin which the servo should be connected to (currently pin D9)
int servoPin = 9 ;
// Servo position as demanded from coordinator
byte servoPos = 0;
// Delay in [ms] that affects servo turning speed.
Higher value means slower speed.
// Please choose value from the interval of <10,100>.
int delayTime = 10;
//###############################################################################
#######
// battery - specific declarations:

// IQRF-specific type of battery object
IQRFbattery battery ;

//###############################################################################
#######
```

```
void setup()
{
  Serial.begin  ( 115200 ); // Setup USB serial for PC connection
  Serial1.begin ( 115200 ); // Setup UART to connect to IQRF TR module

  // initialise servo object with the selected pin to attach
  servo.init ( servoPin );

  //  perform a battery voltage check after boot
  battery.checkBatteryLevel () ;
}
```

Ve funkci *main* je nejprve vytvořen objekt *servo* typu *IQRFservo*. Jsou deklarovány a inicializovány proměnné pro specifikaci pinu, kterým lze servo ovládat, ukládání požadavku na novou úhlovou pozici serva a časovou prodlevu pro možnost řídit kromě úhlu náklonu také rychlost otáčení. Dále je vytvořen objekt *battery* typu *IQRFbattery*. Ve funkci *setup* jsou spuštěny obě sériové linky, obdobně jako v úvodní ukázce pro ověření funkce. Také je zde přiřazením příslušného pinu inicializováno servo a je provedeno prvotní měření stavu akumulátoru.

Zápis dat - IQRFservo

```
// write request for sensor #1 (servo control)
if ( ( _DpaDataLength == 9 || _DpaDataLength == 14 ) && _DpaMessage.Request.
PData[4] == 1 )
{
  // sensor type == [0x80] Relative Humidity
  // hence we are interested only in the first data byte [6]
  servoPos = _DpaMessage.Request.PData[5] ;

  // turn servo to the requested position
  turnServo = true ;
}
```

Pokud je splněna vstupní podmínka a typ příchozí zprávy značí požadavek zápisu na standardní senzor typu *Relative Humidity*, je do proměnné *servoPos* uložena hodnota šestého příchozího bytu pole *PData*, obsahující požadavek na změnu polohy servomotoru. Protože otočení motorem trvá příliš dlouho na to, aby node mohl včas odeslat koordinátoru potvrzující zprávu a došlo by k vypsání chyby, byla v kódu zavedena binární proměnná *turnServo*. Tato proměnná na konci DPA funkce indikuje, zda byl typem příchozí zprávy požadavek na otočení. Pokud ano, je po odeslání zprávy koordinátoru zavolána metoda *setAngle* třídy *IQRFservo*. Toto je vidět v kódu níže.

```
// Return DPA response
```

```
ResponseCommand(returnFlags,_DpaDataLength, returnDataLength, (byte*)&_DpaMessage);

if (turnServo) servo.setAngle ( servoPos, delayTime ); // finally, turn the servo
```

Čtení dat – IQRFbattery a IQRFservo

Podmínka tohoto typu je v kódu obsažena celkem třikrát. Důvodem je, že IQRF koordinátor se může nodu dotazovat na hodnoty, popřípadě typy a hodnoty přítomných senzorů, a to buď jednotlivě nebo dotazem na všechny senzory naráz.

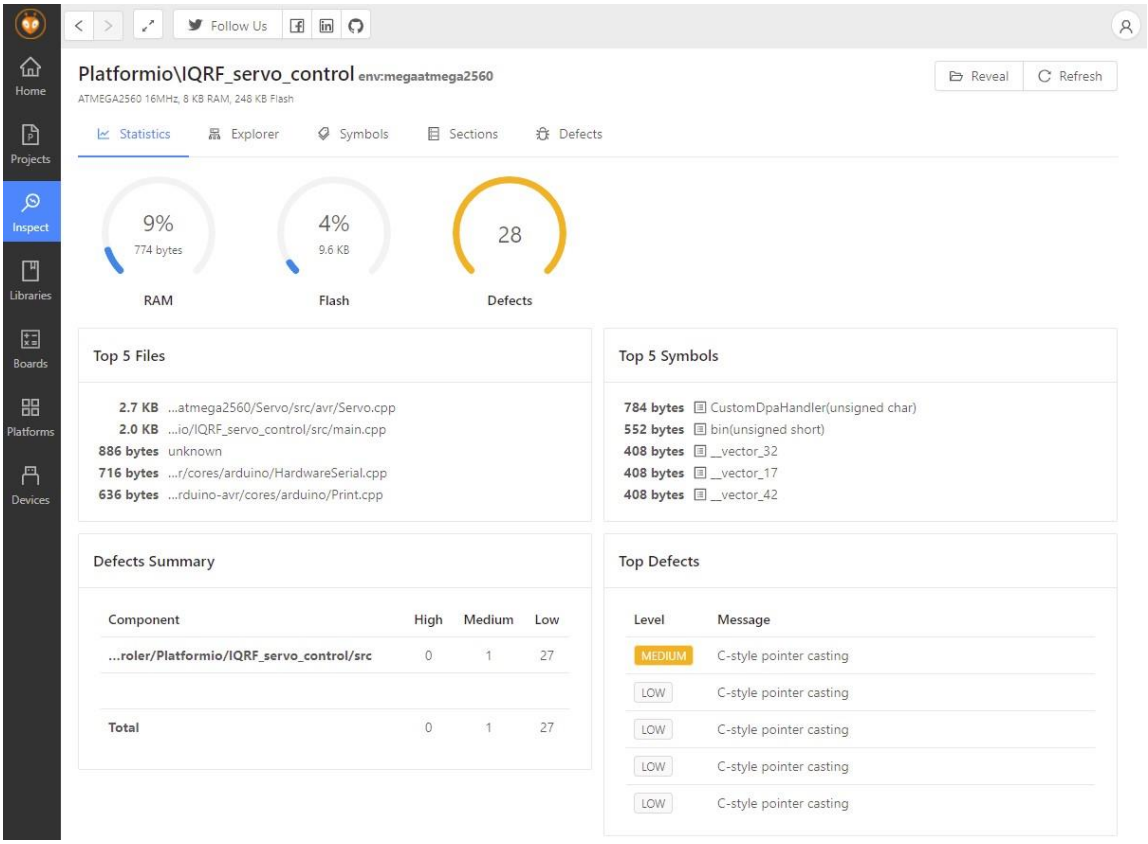
Dotaz, obsahující v prvním bytu pole *PData* hodnotu 1 znamená požadavek na hodnotu *Relative Humidity*, tedy na nastavený úhel servomotoru. Hodnota 2 značí dotaz na senzor *Low Voltage*, tedy na aktuální naměřené napětí akumulátoru. Pokud přijde hodnota 3, dotaz je směřován na první a druhý senzor (3 je binárně *11*, to znamená dotaz na senzory na pozicích nultého a prvního bitu). Hodnota 255 představuje broadcast požadavek, dotaz tedy cílí na všechny dostupné senzory daného zařízení. V tomto programu jsou definovány senzory dva, hodnota 3 a 255 tak zde prakticky znamená týž požadavek. Následující blok kódu pokrývá poslední zmíněný případ, byl proto vybrán pro demonstraci dotazu na oba senzory.

```
 if ( _DpaMessage.Request.PData[0] == 3 || _DpaMessage.Request.PData[0] == 255 )
{
  // Return also sensor type?
  if ( _PCMD == PCMD_STD_SENSORS_READ_TYPES_AND_VALUES )
    *pResponseData++ = STD_SENSOR_TYPE_HUMIDITY ;

  if ((_DpaDataLength == 9 || _DpaDataLength==14)&&_DpaMessage.Request.PData[4]==1)
    *pResponseData++ = servoPos ;
  else
    *pResponseData++ = servo.getAngle () ;

  if ( _PCMD == PCMD_STD_SENSORS_READ_TYPES_AND_VALUES )
    *pResponseData++ = STD_SENSOR_TYPE_LOW_VOLTAGE;
  // Return sensor data
  battery.checkBatteryLevel ();
  *pResponseData++ = battery.battLow;  // little endian => low goes first
  *pResponseData++ = battery.battHigh;
}
```

Pokud má příchozí *_PCMD* nastaven typ *PCMD_STD_SENSORS_READ_TYPES_AND_VALUES*, je splněna první vnitřní podmínka a do pole bytů *pResponseData* je nejprve zapsán typ prvního senzoru, tedy *Relative Humidity*. Následující podmínka zjišťuje, zda byl zároveň s dotazem na stav senzorů poslán také příkaz ke změně polohy serva. Pokud ano, je do pole *pResponseData* dále přidána hodnota nového úhlu, který bude záhy nastaven. Pokud ne, zavolá se metoda *getAngle* třídy *IQRFservo* pro zjištění aktuálního úhlu. Tím je připravena první část odpovědi pro koordinátor, poskytující informace o prvním senzoru. Další podmínka je shodná s první popsanou. Tedy, pokud přišel požadavek na hodnoty včetně typů senzorů, ke kterým patří, je do pole dále přiřazen typ senzoru pro měření napětí. Na závěr je zavolána metoda *checkBatteryLevel* třídy *IQRFbattery*, která zajistí provedení měření napětí a přípravu formátu odpovědi pro typ senzoru *Low Voltage*. Do pole jsou postupně přidány oba byty třídních proměnných *battLow* a *battHigh*, je zavolána funkce *ResponseCommand* z předchozího bloku kódu, která zajistí odeslání odpovědi koordinátoru, a poté je případně vykonán pohyb servomotoru.
Jak v Packet Inspectoru vypadá odpověď na tento konkrétní dotaz na typy a hodnoty obou senzorů, ukazuje Obrázek 42 v úvodu této podkapitoly.

4.4.6 **Statická analýza kódu**

Obrázek 43 zachycuje výsledek provedené statické analýzy celého kódu. Na platformě Arduino Mega 2560 program zabírá příznivá 4 % (9,6 KB) z celkových 256 KB flash paměti a pouhých 9 % (774 B) z celkových 8 KB paměti SRAM. Defekty kódu mají povětšinou pouze mírný charakter závažnosti. V případě cílového použití na mikrokontroleru Nano bude procentuální využití vyšší z důvodu menší velikosti obou pamětí. Nano disponuje pouze 32 KB flash paměti a 2 KB SRAM. [46] Program zde tedy bude zabírat 30 % flash, resp. 39 % SRAM paměti.



Obrázek 43 Statická analýza kódu v PlatformIO, cílová deska: Arduino Mega [autor]

4.5 Instalace a dokončení

Na základě předposledního dílčího cíle je nyní nezbytné provést následující kroky pro uvedení systému do provozního stavu. Nejprve je třeba do cílového mikrokontroleru Arduino Nano nahrát obslužný program s případnými drobnými úpravami v kódu. Ty jsou dány především rozdílnými parametry obou mikrokontrolerů, zejména odlišným počtem a mapováním pinů. Jakmile Nano obsahuje funkční program, může být propojeno se servomotorem, akumulátorem a prostřednictvím převodníku také s IQRF transceiverem v roli nodu. Takto připravená sestava může být nakonec nainstalována do konzoly žaluzií na maketě okna.

Výsledek provedené instalace je představen níže (Obrázek 44). Prototyp systému řízení žaluzií v maketě okna, ovládaný bezdrátovou mesh technologií IQRF, je nyní připraven pro budoucí prezentaci a testování dalších funkcí.

Obrázek 44 Maketa okna s nainstalovaným systémem řízení žaluzií; zleva – servomotor, Arduino Nano, IQRF transceiver, akumulátor [autor]

5 Výsledky a diskuse

Oproti bakalářské práci se zde programová část pro mikrokontroler týká především obsluhy bezdrátové komunikace a zpracování řídicích povelů z koordinátoru či odesílání telemetrických dat na koordinátor. V bakalářské práci byla řešena především logika řízení servomotoru a nízkoenergetický provoz celé sestavy.

IQRF jistě nebylo jedinou technologií, která se nabízela pro druh aplikace, daný touto prací. Dalším kandidátem byl modul XBee založený na technologii ZigBee.

Spotřeba energie tohoto transceiveru v režimu spánku [58] je srovnatelná se spotřebou modulu IQRF, popsanou v kapitole 3.3.3. Nevýhodou však může být vysílací výkon základní varianty modulu, jehož hodnota činí pouhý 1 mW a je tak desetkrát nižší oproti IQRF modulu. Maximální dosah modulu uvnitř budov je deklarován jen na 30 metrů, to je podstatně méně oproti IQRF. V případě varianty XBee PRO je vysílací výkon shodný s IQRF, tedy maximálně 10 mW (výkon je softwarově nastavitelný). Dalším důležitým faktorem je cena modulu. Protože původní myšlenka celého projektu spočívala především ve tvorbě open-source řešení, které bude násobně levnější než komerčně dostupné systémy, je důraz na nízkou cenu jednotlivých komponent zásadním hodnoticím kritériem. V případě výkonnější PRO varianty činí cena jednoho kusu v e-shopu Mouser Electronics 705 Kč. Včetně DPH, které bude započteno při doručení zboží z USA, vychází konečná cena na 850 Kč za kus. To je téměř trojnásobek ceny transceiveru IQRF. V rámci EU je navíc těžko proveditelné tyto moduly zakoupit, dostupné jsou jen pro objednání z USA. IQRF moduly lze z jičínské provozovny firmy obdržet již během jednoho až dvou dnů po objednání. Po zvážení všech uvedených faktů byla zvolena technologie IQRF. S jejím využitím by neměla být ekonomická efektivita projektu výrazněji negativně odchýlena od původních kalkulací provedených v bakalářské práci. [59]

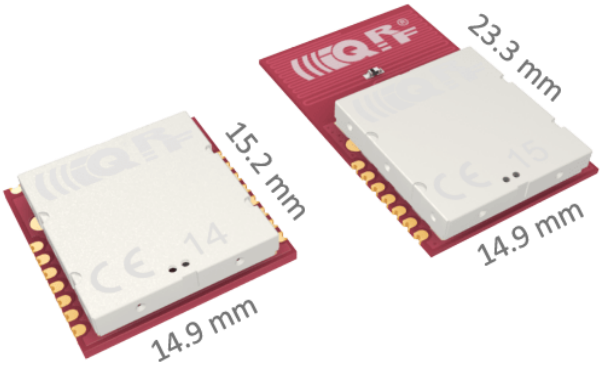
Výstup praktické části práce – zdrojové kódy obou knihoven i obsah funkce *main.cpp* – budou před jejím publikováním zveřejněny v GitHub repozitáři, který byl za tímto účelem založen. Webová adresa repozitáře je následující: *https://github.com/jprikner/IQRF-Arduino.* Obě knihovny budou šířeny pod open source licencí.

5.1 Doporučení pro další vývoj

5.1.1 Úpravy pro zvýšení ekonomické a energetické efektivity

Pro skutečně nízkoenergetický provoz celé sestavy je třeba provést několik dalších opatření, na které v této práci již nebyl prostor.

Důležitým krokem v tomto směru je z ekonomických důvodů, ale i z hlediska možností fyzické vestavby, zvážit nutnost použití převodníku IQRF-BB-01. Cena jednoho kusu s DPH v roce 2021 činí 220 Kč. Pro modelový příklad uvedený v bakalářské práci [59], ve kterém byla kalkulována instalace systému na celkem devíti oknech, znamená (při započítání 9ks TR modulů namísto původních NRF24L01+ a při zachování cen a množství ostatních komponent) tato téměř dvoutisícová položka zvýšení pořizovacích nákladů o přibližně jednu sedminu. V případě nepoužití převodníku by bylo kvůli absenci konektoru KON-SIM-02 nutné přistoupit k připojení transceiveru k Arduinu přímo připájením. Vhodnějším modulem než doposud zmiňovaný TR-72DA by v takovém případě byl model TR-76DA (Obrázek 45 níže), který disponuje kromě kontaktů pro vložení do SIM konektoru na spodní straně také plochami po stranách pro SMT (povrchovou) montáž. [60]



Obrázek 45 IQRF TR-76D / TR-76DA [60]

S tím se pojí doporučení, které by bylo dobré provést v každém případě – zajistit provoz celé sestavy na 3,3 V, tak, aby v případě potřeby nebylo použití převodníku nezbytné. Zvýší se tím mimo jiné univerzálnost celého řešení.

Lze navrhnout například použití dobíjecího lithium-iontového článku o nominálním napětí 3,7 V (napětí při plném nabití přibližně 4,2 V) a zařazení step-down měniče, který by zajistil stabilní provozní napětí 3,3 Voltu pro Arduino i transceiver. S ohledem na zajištění minimalizace spotřeby energie celku je třeba brát zvýšený zřetel na parametry step-down

měniče, zejména na jeho spotřebu bez zátěže, neboť bude permanentně připojený v napájecím obvodu. Dalším nezbytným prvkem při tomto řešení bude i měnič typu step-up, a to pro pohon servomotoru. Tento měnič již nemusí splňovat přísné parametry ohledně účinnosti, servomotor totiž musí být během nečinnosti vyřazen z provozu pomocí tranzistoru. Důvodem je princip funkce a chování servomotoru, který po dobu, kdy je pod napětím, „drží“ nastavený úhel. V důsledku této své vlastnosti kontinuálně spotřebovává proud v řádu jednotek miliampér, aktuální hodnota se liší dle velikosti a typu serva, a také dle vstupního napětí. Na konkrétním servomotoru EMAX, vyobrazeném v kapitole 4.1, bylo v rámci bakalářské práce naměřeno 7 mA v klidovém stavu. Pokud bude tranzistor v obvodu zařazen mezi zdroj a měnič, vyřadí jej vždy spolu se servomotorem a spotřeba této části systému tak bude při vypnutém stavu nulová (únikový proud se liší dle použitého typu tranzistoru, jeho hodnota nicméně bývá natolik nízká, že tento faktor lze zanedbat).

5.1.2 Úpravy softwarové části

5.1.2.1 Hromadný sběr dat pomocí FRC

Jedna z velice zdařilých a důležitých vlastností IQRF sítě je možnost sbírat data ze všech nodů naráz pomocí jediného Fast Response Command (FRC) požadavku.
V případě projektu, jako je tento, může být v rámci jedné domácí instalace počítáno s množstvím od pár jednotek (nejmenší byty) až po několik desítek (velké rodinné domy, vily) kusů oken, která je třeba osadit systémem řízení interiérových žaluzií. Pro rychlé plošné zjištění stavů systémů ve všech oknech by bylo vhodné doimplementovat pro Arduino podporu sběru dat pomocí FRC. Jedním FRC požadavkem vyslaným z koordinátoru by tak mělo být možné shromáždit data o stavech všech akumulátorů, pozicích nastavených na jednotlivých servomotorech či případných dalších funkcích, které mohou být v budoucnu přidány.

5.1.2.2 Indikace možných stavů systému

IQRF transceiver se může nacházet ve stavu, kdy je buď připojen (tzv. bonded) do IQMESH sítě či nikoliv (tzv. unbonded). Pokud připojen není, modul indikuje tento stav rychlým blikáním červené LED diody. V momentě, kdy dojde k jeho připojení do sítě, indikuje toto jedním bliknutím zelenou LED, následovaným jedním bliknutím červenou LED. Od této

chvíle je připojen a již se žádnou indikací neprojevuje (LED diody zhasnuty). Pokud je přerušeno napájení (např. během úkonů údržby či výměny akumulátorů), po obnovení napájení indikuje node připojený do sítě tento stav jedním bliknutím červenou LED.
Popsané stavy lze vizuálně kontrolovat, pokud se transceiver nachází například na stole v programátoru či je nainstalován s odkrytými LED diodami na dobře přístupném místě.
V případě modulu zabudovaného do horní konzoly žaluzií však LED diody viditelné nejsou, modul je z velké části skryt uvnitř konzoly. Je tedy nutné využít jiný způsob indikace. Pro signalizaci se přímo nabízí využití žaluzií, které jsou již k systému připojeny přes servomotor.
Dalším navrhovaným rozšířením tedy je vytvořit sadu indikací prostřednictvím změny náklonu žaluzií. Příklad: node nepřipojený do sítě dává tento stav najevo nastavením žaluzií do jejich střední polohy (otevřeno) a následně rychlým drobným otáčením v rozsahu -10° až +10° od tohoto středu, opakujícím se v 15s intervalech trvale až do doby připojení do sítě či odpojení napájení. Připojení k IQMESH síti, stejně jako právě obnovené napájení, může být indikováno rychlým otočením žaluzií v plném rozsahu nejprve na minimální hodnotu a poté na maximální, s následným pomalým nastavením na střed.

5.1.2.3 **Rozšíření obslužného rozhraní**

Díky zveřejnění v Git repozitáři mohou být obě vytvořené knihovny dále upravovány komunitou, například přidáváním dalších funkcí.
Jedno z možných vylepšení vychází z předchozí podkapitoly, doporučující implementovat indikaci stavů, ve kterých se radiová část systému může nacházet. Pro realizaci této indikace lze upravit knihovnu *IQRFservo* přidáním nových metod, které budou zajišťovat potřebné chování.
Knihovna *IQRFbattery* by mohla být doplněna o metodu realizující výpočet zbývající kapacity baterie v procentech. V takovém případě by bylo vhodné počítat s různými vybíjecími charakteristikami běžně dostupných typů akumulátorů. Z nich by byla při inicializaci systému vybrána ta, která by odpovídala danému typu připojeného akumulátoru. Typ akumulátoru by mohl být buď nastavován ručně v kódu, či lépe, by mohlo být využito možnosti zapisovat data do IQRF standardního senzoru, a informaci o připojeném typu danému zařízení nastavit vzdáleně. Protože by byla koordinátoru odesílána procentuální

hodnota, bylo by ideální využít pro tento účel typ senzoru *Relative Humidity*, podobně jako je tomu u řízení úhlu servomotoru.

5.1.3 Úpravy pro zvýšení uživatelského komfortu

Pokud má být celý systém nasazen v běžném domácím prostředí, je nezbytně nutné zajistit určitou míru přívětivosti jeho obsluhy. K tomu mohou dopomoci následující dodatečná vylepšení.

5.1.3.1 IQRF Gateway

Brána pro IQRF komunikaci je velice důležitým funkčním prvkem, pokud má být systém ovládán mimo IQRF IDE. Pro implementaci libovolného uživatelského rozhraní určeného k interakci se systémem musí být nejprve zprovozněna IQRF gateway. [61]
Doporučeno je použití jednodeskového minipočítače Raspberry Pi, jakožto platformy, která je cenově dostupná a současně dostatečně výkonná i pro provoz uživatelského rozhraní. Ideálním operačním systémem pro doporučenou gateway je Raspberry Pi OS, vycházející z distribuce OS Debian rodiny GNU/Linux. Jak bylo řečeno v kapitole 4.3.4, připojení IQRF transceiveru se v tomto případě realizuje pomocí adaptéru IQRF-KON-RASP-01, který se nasadí na příslušné GPIO piny Raspberry Pi. Softwarová část obnáší stažení balíčku IQRF Gateway Daemon [62] z repozitáře IQRF, jeho instalaci a spuštění jako systémovou službu init systému *systemd*. Podrobnosti instalace jsou popsány v dokumentaci IQRF gateway. Bránu lze konfigurovat ve webové aplikaci, kterou je také třeba nejprve obdobným způsobem nainstalovat. [61]

5.1.3.2 Home Assistant: uživatelský informační systém

Funkční a přívětivé uživatelské rozhraní je základem jakékoli domácí automatizace. Pro systém autonomně řízených žaluzií je potřeba mít možnost ovládat jednotlivá okna, místnosti, či celý byt či dům. Také musí být možno sledovat stavy žaluzií, tedy jejich aktuální nastavenou polohu, úroveň nabití akumulátoru či případné další dodatečně doplněné funkce. Uživatelské rozhraní tedy musí být dostatečně robustní, univerzální a natolik široce konfigurovatelné, že toto vše umožní. Zároveň musí podporovat komunikaci s IQRF Gateway Daemonem.

Vhodným informačním systémem, který tyto specifika splňuje, je například open-source software Home Assistant [63]. Tento systém disponuje kromě webového rozhraní i mobilní aplikací pro Android a iOS. Home Assistant podporuje řadu platforem včetně OS Windows, MacOS a OS Linux. Na Raspberry Pi jej lze nainstalovat buď jako samostatný operační systém Home Assistant OS, spustit jej jako Home Assistant Container např. v prostředí Docker, či jej nainstalovat standardní cestou pomocí instalačního manažera *python pip*. Podrobnosti instlace jsou přehledně zdokumentovány na webové adrese *https://www.home-assistant.io/installation/raspberrypi*. [64]

6 Závěr

Bezdrátový bateriový provoz chytrých zařízení pro domácí automatizaci byl ještě před několika lety prakticky nereálný. Díky vývoji technologií dnes již existují řídicí prvky o velikosti mikrotužkové baterie i bezdrátové moduly velikosti SIM karty, pomocí kterých lze ovládat nejrůznější připojené periferie či sbírat naměřená data.

V teoretické části byly popsány principy internetu věcí a souvisejících technologií. Toto bylo podpořeno uvedením praktických příkladů konkrétních aplikací, včetně systémů pro domácí automatizaci. Dále byly představeny vlastnosti a kategorie bezdrátových síťových technologií, se zaměřením na sítě typu mesh. Technologie IQRF, jako jeden ze zástupců tohoto druhu sítí, byla popsána detailněji. Zvláštní pozornost byla věnována těm aspektům, které byly později klíčové pro tvorbu praktické části práce. Šlo zejména o části specifikace DPA protokolu, věnující se použití periferií typu IQRF Standard Sensor. Tento druh periferie sdružuje širokou škálu typů fyzických senzorů, pro které je implementováno komunikační rozhraní. Dvě z těchto rozhraní, jejichž vlastnosti se jevily být nejblíže potřebám systému, byly později v praktické části zvoleny pro řízení a čtení dat ze systému. Část kapitoly se také věnovala platformě Arduino, pro kterou byla knihovna určena. Zde byly diskutovány vlastnosti dvou vybraných vývojových prostředí, z nichž jedno bylo později vybráno pro potřeby praktické části práce. V závěru kapitoly bylo také pojednáno o softwarových knihovnách.

Praktická část se věnovala nejprve představení stavu, ve kterém se projekt nacházel na počátku realizace práce. Následně byly popsány varianty fyzického zapojení jednotlivých komponent systému jak pro vlastní vývoj, tak i pro pozdější provoz. Poté byla započata vlastní práce na knihovně, která byla rozdělena na dva samostatné logické celky. První celek sdružuje funkce spojené s obsluhou servomotoru, druhý se zaměřuje na čtení elektrického napětí akumulátoru. U obou takto rozdělených knihoven byla vysvětlena činnost jednotlivých třídních metod. Jakmile byla programová část hotova, byla ještě provedena statická analýza celého kódu pro zjištění jeho paměťové náročnosti a případných defektů. Program v tomto ohledu bez problému obstál. Závěr kapitoly se věnoval vestavbě celé sestavy do připravené makety okenního rámu se žaluziemi.

Poté byly diskutovány alternativní možnosti v podobě jiných technologií na stejném principu. Byla také sestavena doporučení pro další rozvoj. V softwarové oblasti byly doporučeny kroky vedoucí ke zlepšení použitelnosti systému běžnými uživateli domácnosti,

k dalšímu snížení energetické náročnosti systému, a také k rozšíření obou vytvořených knihoven o další možné metody.

Knihovní funkce, vytvořené v rámci diplomové práce, jistě nebudou konečnou podobou tohoto projektu. I po dokončení práce bude vývoj celého řešení pokračovat v souladu s myšlenkami a doporučeními z kapitoly 5.1. Projekt bude rozvíjen až do podoby, kdy bude moci být celý systém nasazen v prostředí běžné domácnosti a jeho používání bude dostatečně jednoduché i pro laickou obsluhu.

7 Seznam použitých zdrojů

[1] AL-FUQAHA, Ala a kol. *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications* [online]. IEEE Communications Surveys & Tutorials, 2015, [cit. 2021-03-05]. ISSN 2373-745X. Dostupné z: https://ieeexplore.ieee.org/document/7123563

[2] SVOBODA, Martin. *B4M36DS2 Databázové systémy 2*. Praha, 2017. Dostupné také z: http://www.ksi.mff.cuni.cz/~svoboda/courses/171-B4M36DS2/. Prezentace k přednáškám z předmětu. ČVUT, Fakulta elektrotechnická.

[3] LEADER LIGHT Street LED lighting. *IQRF Tech* [online]. [cit. 2021-03-05]. Dostupné z: https://www.iqrf.org/case-study/street-led-lighting

[4] KOBZA, Jaromír. Řízení osvětlovacích soustav veřejného osvětlení (VO). In: *Univerzita Palackého v Olomouci, katedra geoinformatiky* [online]. [cit. 2021-03-06]. Dostupné z: https://www.cenia.cz/wp-content/uploads/2019/07/Kobza_2019.pdf

[5] Building automation: Research and Technology Center MICRORISC. *IQRF Tech* [online]. [cit. 2021-03-06]. Dostupné z: https://www.iqrf.org/case-study/building-automation

[6] PUČELÍK, Karel. V kancelářích se scházíme, abychom sdíleli informace, říká Lenka Čábelová z Microsoftu. *E15* [online]. [cit. 2021-03-06]. Dostupné z: https://www.e15.cz/rozhovory/v-kancelarich-se-schazime-abychom-sdileli-informace-rika-lenka-cabelova-z-microsoftu-1339857

[7] LoRaWAN People Counter. *IMBUILDINGS Netherlands* [online]. [cit. 2021-03-06]. Dostupné z: https://www.imbuildings.com/lorawan-people-counter

[8] VIVIDI. *ITERAIT* [online]. [cit. 2021-03-12]. Dostupné z: https://vividi.io/

[9] Zelená umělá inteligence z ČR pomáhá v boji s covid-19. *ČTK* [online]. 2021 [cit. 2021-03-12]. ISSN 1213-5003. Dostupné z: https://www.ceskenoviny.cz/tiskove/zpravy/zelena-umela-inteligence-z-cr-pomaha-v-boji-s-covid-19/2006767

[10] Altron Smart Services. *Altron* [online]. [cit. 2021-03-12]. Dostupné z: https://www.altron.net/smart-services/

[11] Philips Hue. *Philips* [online]. [cit. 2021-03-12]. Dostupné z: https://www.philips-hue.com/

[12] IKEA Home smart. *IKEA* [online]. [cit. 2021-03-12]. Dostupné z: https://www.ikea.com/cz/cs/product-guides/ikea-home-smart-system/

[13] Sonoff S26 Wi-Fi Smart Plug. *Sonoff* [online]. [cit. 2021-03-12]. Dostupné z: https://sonoff.tech/product/smart-plug/s26/

[14] Lidl Smart Home. *Lidl* [online]. [cit. 2021-03-12]. Dostupné z: https://www.lidl.cz/smart-home

[15] Vnitřní žaluzie: flexibilní regulace světla. *Somfy* [online]. [cit. 2021-03-12]. Dostupné z: https://www.somfy.cz/produkty/interierove-stineni/vnitrni-zaluzie

[16] Google Nest. *Google Store* [online]. [cit. 2021-03-12]. Dostupné z: https://store.google.com/product/nest_audio

[17] ROUTERSWITCH TECH. Types of Wireless Network & Wireless Topologies. *Router-Switch.com* [online]. 2014 [cit. 2021-03-15]. Dostupné z: https://blog.router-switch.com/2014/01/types-of-wireless-network-wireless-topologies

[18] What is IQRF?. *IQRF UK* [online]. [cit. 2021-03-16]. Dostupné z: https://iqrf.co.uk/what-is-iqrf.html

[19] Využívání vymezených rádiových kmitočtů. *Český telekomunikační úřad* [online]. [cit. 2021-03-16]. Dostupné z: https://www.ctu.cz/vyuzivani-vymezenych-radiovych-kmitoctu

[20] SKøIEN, Kristoffer Rist. Network Topologies in wireless solutions. *Nordic Semiconductor: Get Connected Blog* [online]. 2020 [cit. 2021-03-16]. Dostupné z: https://blog.nordicsemi.com/getconnected/wireless-network-topologies

[21] HYNČICA, Ondřej. Bezdrátové sítě typu mesh. *Automa – časopis pro automatizační techniku* [online]. 2005 [cit. 2021-03-16]. Dostupné z: https://automa.cz/cz/casopis-clanky/bezdratove-site-typu-mesh-2005_12_30826_1141

[22] BLUM, Jeremy. *Exploring Arduino: tools and techniques for engineering wizardry*. Second edition. Indianapolis: Wiley, 2020. ISBN 978-1-119-40537-5.

[23] PRAUZEK, Michal. Druhy komunikačních rozhraní. In: *VŠB – Technická univerzita Ostrava: Vestavěné řídicí systémy, přednáška 6* [online]. 2017 [cit. 2021-03-16]. Dostupné z: https://homel.vsb.cz/~pra132/vrs/VRS_prednaska6.pdf

[24] Company profile. *IQRF Tech* [online]. [cit. 2021-03-17]. Dostupné z: https://www.iqrf.org/sales/company

[25] TR-72D Transceiver series. *IQRF Tech* [online]. [cit. 2021-03-17]. Dostupné z: https://www.iqrf.org/product-detail/tr-72d

[26] TR-72D RF Transceiver Module Series Data Sheet. *IQRF Tech* [online]. 2020 [cit. 2021-03-17]. Dostupné z: https://static.iqrf.org/Datasheet_TR-72D_200525.pdf

[27] Low power modes: IQRF wireless is extremely low power. *IQRF Tech* [online]. [cit. 2021-03-17]. Dostupné z: https://www.iqrf.org/technology/low-power-modes

[28] How to start?: The first IQRF design. *IQRF Tech* [online]. [cit. 2021-03-17]. Dostupné z: https://www.iqrf.org/how-to-start

[29] KUCHTA, Radek, Radimir VRBA a Vladislav SULC. Smart Wireless Communication Platform IQRF. *Mobile and Wireless Communications Network Layer and Circuit Level Design* [online]. InTech, 2010 [cit. 2021-03-18]. ISBN 978-953-307-042-1. Dostupné z: doi:10.5772/7711

[30] SPURNÁ, Ivona. Jak na IQRF: směrování v síti a Fast Response Command. *Root.cz* [online]. 2019 [cit. 2021-03-18]. Dostupné z: https://www.root.cz/clanky/jak-na-iqrf-smerovani-v-siti-a-fast-response-command

[31] FRC® - Fast Response Command. *IQRF Tech* [online]. [cit. 2021-03-18]. Dostupné z: https://www.iqrf.org/technology/iqmesh/frc

[32] IQRF DPA Framework V4.15: Technical Guide. *IQRF Tech* [online]. 2020 [cit. 2021-03-18]. Dostupné z: https://doc.iqrf.org/DpaTechGuide/

[33] IQRF technology overview. *HDI Electronics* [online]. [cit. 2021-03-18]. Dostupné z: http://www.hdi-electronics.fr/en/iqrf-smarter-wireless-simply/do-it-wireless-simply/

[34] IQRF Standard Sensor. *IQRF Tech* [online]. 2021 [cit. 2021-03-20]. Dostupné z: https://www.iqrfalliance.org/techdoc_files/IQRF-StandardSensor_V015.pdf

[35] SPURNÁ, Ivona. Bezdrátová technologie IQRF. *ATP Journal* [online]. 2016 [cit. 2021-03-20]. Dostupné z: https://www.atpjournal.sk/rubriky/prehladove-clanky/bezdratova-technologie-iqrf-2.html?page_id=23425

[36] IQRF IDE Help. *IQRF Tech* [online]. [cit. 2021-03-20]. Dostupné z: https://doc.iqrf.org/IQRF-IDE-Help

[37] Arduino Uno datasheet. *Farnell* [online]. [cit. 2021-03-21]. Dostupné z: https://www.farnell.com/datasheets/1682209.pdf

[38] MORRISSEY, Donal. *Arduino, Zigbee and Embedded Development: Sleeping Arduino - Part 1* [online]. 2010 [cit. 2021-03-21]. Dostupné z: http://donalmorrissey.blogspot.cz/2010/04/putting-arduino-diecimila-to-sleep-part.html

[39] Microsoft Visual Studio Code. *Microsoft* [online]. [cit. 2021-03-21]. Dostupné z: https://code.visualstudio.com/

[40] BOŘÁNEK, Roman. Visual Studio Code: nový editor od Microsoftu zdarma. I pro Linux. *Root.cz* [online]. 2015 [cit. 2021-03-21]. Dostupné z: https://www.root.cz/zpravicky/visual-studio-code-novy-editor-od-microsoftu-zdarma-i-pro-linux/

[41] PlatformIO IDE. *PlatformIO labs* [online]. [cit. 2021-03-21]. Dostupné z: https://platformio.org

[42] Library Dependency Finder. *PlatformIO labs* [online]. [cit. 2021-03-21]. Dostupné z: https://docs.platformio.org/en/latest/librarymanager/ldf.html

[43] COOK, Jeremy S. Arduino Visual Studio Code: Arduino IDE vs. PlatformIO. *ARROW* [online]. 2020 [cit. 2021-03-21]. Dostupné z: https://www.arrow.com/en/research-and-events/articles/arduino-visual-studio-code-arduino-ide-vs-platformio

[44] MARTINEK, David. *Jak na projekty v jazyce C: Moduly a knihovny* [online]. [cit. 2021-03-23]. Dostupné z: http://jaknaprojekty.davidm.cz/moduly.html

[45] WHEELER, David A. *Program Library HOWTO* [online]. 2003 [cit. 2021-03-23]. Dostupné z: https://tldp.org/HOWTO/Program-Library-HOWTO

[46] Arduino Nano. *Components101.com* [online]. [cit. 2021-03-24]. Dostupné z: https://components101.com/microcontrollers/arduino-nano

[47] Servomotor EMAX. *Ready Made RC* [online]. [cit. 2021-03-24]. Dostupné z: https://www.readymaderc.com/store/images/emax-es08aii.jpg

[48] Komunikujeme bezdrátově s NRF24L01+. *Arduino8.cz* [online]. [cit. 2021-03-24]. Dostupné z: http://www.arduino8.cz/komunikujeme-bezdratove-s-nrf24l01-1-cast-zapojeni)

[49] DK-EVAL-04A: Universal development kit for TR modules. *IQRF Tech* [online]. [cit. 2021-03-24]. Dostupné z: https://www.iqrf.org/product-detail/dk-eval-04a

[50] IQRF-BB-01. *Wobchod.cz* [online]. [cit. 2021-03-24]. Dostupné z: https://www.wobchod.cz/products/detail/IQRF-BB-01

[51] KON-RASP-01: Adapter to connect TR to Raspberry Pi. *IQRF Tech* [online]. [cit. 2021-03-24]. Dostupné z: https://www.iqrf.org/product-detail/kon-rasp-01

[52] Downloads. *IQRF Tech* [online]. [cit. 2021-03-31]. Dostupné z: https://www.iqrf.org/support/downloads

[53] Atmel AVR. *PlatformIO labs* [online]. [cit. 2021-03-24]. Dostupné z: https://docs.platformio.org/en/latest/platforms/atmelavr.html

[54] Arduino Servo library. *Arduino.cc* [online]. [cit. 2021-03-24]. Dostupné z: https://www.arduino.cc/reference/en/libraries/servo

[55] Discovery: Automated creating a routing structure. *IQRF Tech* [online]. [cit. 2021-03-24]. Dostupné z: https://www.iqrf.org/technology/iqmesh/discovery

[56] ArduinoCore-avr library. *GitHub.com* [online]. [cit. 2021-03-24]. Dostupné z: https://github.com/arduino/ArduinoCore-avr/blob/master/cores/arduino/Arduino.h

[57] Arduino stdint.c library. *GitHub.com* [online]. [cit. 2021-03-24]. Dostupné z: https://github.com/codebendercc/arduino-core-files/blob/master/v102/hardware/tools/avr/lib/avr/include/stdint.h

[58] Xbee Datasheet. *Sparkfun.com* [online]. [cit. 2021-03-27]. Dostupné z: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

[59] PRIKNER, Jakub. *Návrh informačního systému pro domácnost* [online]. 2017 [cit. 2021-03-31]. Dostupné z: https://is.ambis.cz/th/uvaab/. Bakalářská práce. AMBIS vysoká škola, a.s. Vedoucí práce Vladimír BENEŠ.

[60] TR-76D Transceiver series. *IQRF Tech* [online]. [cit. 2021-03-27]. Dostupné z: https://www.iqrf.org/product-detail/tr-76d

[61] IQRF Gateway. *IQRF Tech* [online]. [cit. 2021-03-27]. Dostupné z: https://docs.iqrf.org/iqrf-gateway/

[62] IQRF Gateway Daemon. *GitHub.com* [online]. [cit. 2021-03-27]. Dostupné z: https://gitlab.iqrf.org/open-source/iqrf-gateway-daemon

[63] Home Assistant. *HomeAssistant.io* [online]. [cit. 2021-03-27]. Dostupné z: https://www.home-assistant.io

[64] Home Assistant Raspberry Pi installation. *HomeAssistant.io* [online]. [cit. 2021-03-27]. Dostupné z: https://www.home-assistant.io/installation/raspberrypi

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

8 Přílohy

Příloha A Programový kód knihovny IQRF_servo_control – přiloženo elektronicky v zip souboru.

Struktura projektu v přiloženém zip souboru: