Uchování a interpretace dat z poplachových a nepoplachových komponent přes protokol MQTT

Martin Krasula

Bakalářská práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav bezpečnostního inženýrství

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Krasula**
Osobní číslo:	**A18349**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**Kombinovaná**
Téma práce:	**Uchování a interpretace dat z poplachových a nepoplachových komponent přes protokol MQTT**
Téma práce anglicky:	**The Storage and Interpretation Of Data From Alarm and NON Alarm Components Using the MQTT Protocol**

Zásady pro vypracování

1. Zpracujte způsoby uchování a interpretace dat.
2. Popište protokol MQTT a výstupní rozhraní poplachových a nepoplachových komponent.
3. Navrhněte řešení pro získání informací z poplachových a nepoplachových komponent pro přenos přes protokol MQTT.
4. Navrhněte vhodné řešení uchování a intepretace dat získané přes protokol MQTT.
5. Realizujte navrhovaný systém.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. MQTT: The Standard for IoT Messaging. *MQTT* [online]. MQTT.org, 2020 [cit. 2020-11-13]. Dostupné z: https://mqtt.org/
2. MQTT Essentials – A Lightweight IoT Protocol. Packt Publishing (April 14, 2017), April 14, 2017. ISBN 978-1787287815.
3. PULVER, Tim, 2019. *Hands-On Internet of Things with MQTT: Build connected IoT devices with Arduino and MQ Telemetry Transport (MQTT)*. 1. USA: Packt. ISBN 978-1789341782.
4. Creating your MySQL Database: Practical Design Tips and Techniques. Birmingham, B27 6PA, UK: Packt Publishing, 2006. ISBN 1-904811-30-2.
5. NIXON, Robin, 2018. *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*. 5. -: O'Reilly Media. ISBN 978-1491978917.
6. IBRAHIM, Dogan a Ahmet IBRAHIM. The Official ESP32 Book. Elektor Digital, 2017. ISBN 978-1907920639.

Vedoucí bakalářské práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **19. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

Ing. Jan Valouch, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 10. května 2021

Martin Krasula, v. r.
podpis studenta

ABSTRAKT

Bakalářská práce se zaměřuje na možnosti získání, uchování a interpretaci dat z poplachových a nepoplachových komponent komunikujících přes standardizovaný otevřený protokol MQTT. V bakalářské práci jsou popsány metody pro uchování dat a jejich možná interpretace, komunikační protokol MQTT a výstupní rozhraní poplachových a nepoplachových komponent.

Z popsaných možností uchování a interpretace dat je vybrán nejvhodnější pro výstupní data získaná přes komunikační protokol MQTT z poplachových a nepoplachových komponent, který je následně realizován pomocí návrhu.

Klíčová slova: protokol MQTT, poplachové a nepoplachové komponenty, interpretace dat, uchování dat

ABSTRACT

The bachelor's thesis focuses on the possibility of obtaining, storing and interpreting data from alarm and non-alarm components communicating via a standardized open protocol MQTT.

The bachelor thesis describes methods for data storage and interpretation, communication protocol MQTT and output interface of alarm and non-alarm components. From the described possibilities of data storage and interpretation, the most suitable for the output data obtained via the MQTT communication protocol from alarm and non-alarm components is selected, which is then implemented by design.

Keywords: MQTT protocol, alarm and non-alarm components, data interpretation, data storage

Děkuji doc. Ing. Petru Šilhavému, Ph.D. za pomoc při vedení bakalářské práce. Mé poděkování patří také manželce a rodinně za jejich podporu.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Uchování dat a následná jejich interpretace je pojem, který člověk provádí automaticky a mnohdy si to ani neuvědomuje. Je to základ našeho vnímání, při kterém pomocí našich smyslů dokážeme určit, zda je ráno, když vstáváme venku světlo či tma nebo jestli je předmět horký nebo studený. Všechny tyto informace si dokážeme zapamatovat a následně interpretovat, když se nás na ně někdo zeptá.

V případě technického vybavení musíme přemýšlet po menších krocích. Především musíme mít k uchování informaci, kterou potřebujeme nejdříve získat. K tomu nám může posloužit jednoduché rozhraní mezi strojem a člověkem, pomocí kterého budeme informaci zadávat ručně podle našich vjemů. Jako příklad by mohlo posloužit požární tísňové tlačítko na zdi. Tento způsob je ale v některých případech nevyhovující, například když nemáme možnost fyzicky tísňové tlačítko zmáčknout. V tomto případě můžeme použít senzor, který pomocí měření některé fyzikální veličiny, nám dokáže tuto informaci poskytnout i bez naší fyzické přítomnosti.

Získanou informaci, ale musíme některým způsobem od senzoru přenést, aby mohla být uchována a následně interpretována a k tomu slouží standardizovaný protokol MQTT, který se používá pro komunikaci mezi centrálním bodem a jeho klienty. Klientem může být poplachová komponenta, která nám v případě vloupání do bytu tuto informaci poskytne. Případně to může být jen informace z nepoplachové komponenty, kterou může reprezentovat teploměr umístěný na balkóně.

Získanou a přenesenou informaci do centrálního bodu pak lze uchovat v moderním pojetí v databázi, v které budou informace připraveny pro následnou intepretaci například pro systém chytrého domu, mobilní aplikaci či ve webovém rozhraní.

I. TEORETICKÁ ČÁST

1 TERMINOLOGIE

Mikrokontroler – Integrovaný obvod obsahující všechny části mikropočítače. Používá se nejčastěji pro jednoúčelové aplikace například pro regulaci teploty, ovládání vzduchotechniky.

Wi-Fi – Jedná se o bezdrátovou komunikaci v počítačových sítí dle standardu IEEE 802.11. Pracující na linkové vrstvě síťového modelu ISO/OSI, využívající frekvenční pásmo 2,4 GHz a 5 GHz.

Bluetooth – Jedná se o bezdrátovou komunikaci mezi několika zařízeními dle standardu IEEE 802.15.1. Standard byl vytvořen jako bezdrátová náhrada sériového drátového rozhraní.

Ethernet – Jedná se o souhrn technologií pro počítačové sítě dle standardu IEEE 802.3. Pro vytváření počítačových sítí se používají přenosová media jako koaxiální kabel, kroucená dvojlinka či optický kabel.

Internet – Jedná se o celosvětový systém propojených počítačových sítí, využívající sadu protokolů TCP/IP.

CCTV – Jedná se o uzavřený kamerový systém používající se k monitorování prostorů. Tento kamerový záznam je pak zobrazován na monitorech v dohledových centrech a ukládán.

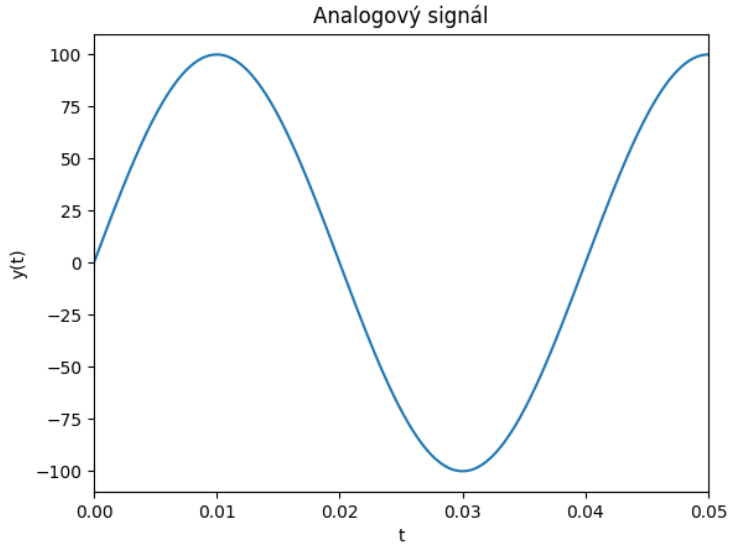
TCP/IP – Jedná se o sadu protokolů využívajících se pro komunikaci v počítačové síti. Sada protokolů je rozdělena do čtyř vrstev a to aplikační, síťové, transportní a aplikační. Vrstvy jsou rozděleny dle hierarchické činnosti na principu, kdy daná vrstva využívá služeb nižší vrstvy a poskytuje svoje služby vrstvě vyšší.

2 UCHOVÁNÍ DAT

Uchování dat je proces, při kterém se data a v nich obsažené informace uloží takovým způsobem, při kterém se zachovají pro možnost jejich následného využití. V dnešní době se data nejčastěji ukládají digitálně tedy v binární podobě a z tohoto důvodu vzniklo velké množství kódovacích algoritmů a převodníků pro převod analogového spojitého signálu na digitální. Jako příklad lze uvést rozpoznávání řeči a následné vyhodnocování případných příkazů v ní obsažených přes Váš mobilní telefon, nebo jen triviální věc jako telefonní hovor. [1] Všeobecně tedy platí, že data jsou uchovávána v digitální či analogové podobě, takovým způsobem, který umožnuje jejich následné opětovné přečtení. [2]

2.1 Analogová data

Analogová data jsou tvořena spojitým analogovým signálem [2]. Mohou být uchována pomocí některého fyzikálního jevu. Například pomocí změn magnetického pole na magnetickou pásku. Další možnost je jejich digitalizace, která se provádí přes analogově digitální převodník. [3] Na obrázku 1 je ukázka analogového signálu.



Obr. 1. Analogový signál

2.2 Digitální data

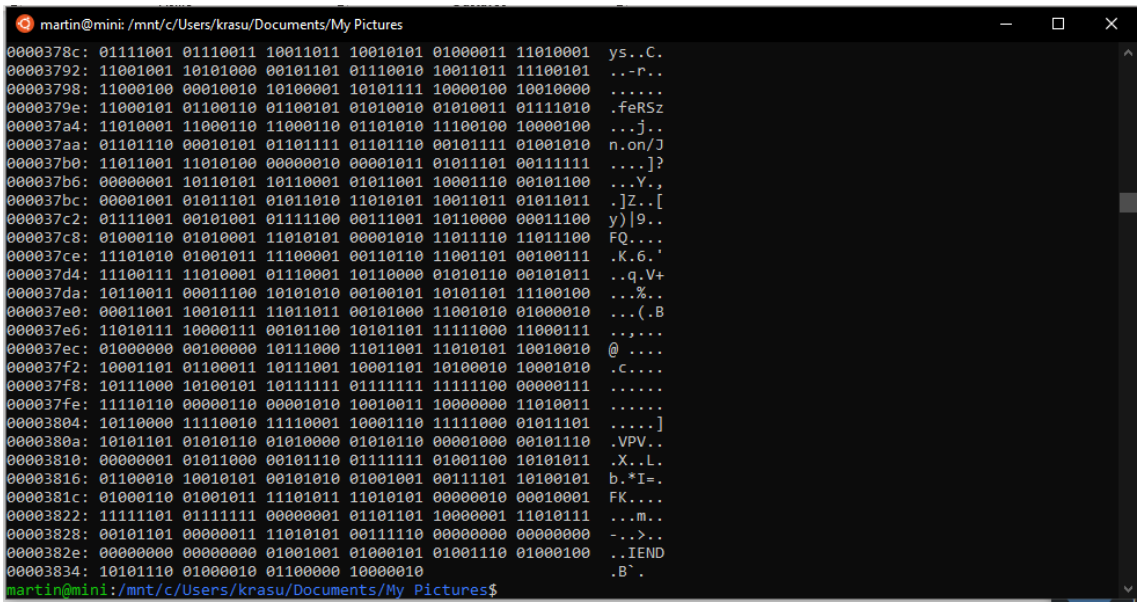
Digitální data jsou reprezentována posloupností jedniček a nul, tedy binárními hodnotami, nad kterými je vytvořena abstrakce, která udává, o jaká data se jedná. [2] [3]

2.2.1 Textová data

Textová data jsou tvořena znaky, které jsou kódovány tak, aby mohla být uložena v binární podobě. [4] Znaky, které lze použít jsou definované použitou znakovou sadou, která určuje, jaká binární hodnota reprezentuje použitý znak. Jednotlivé řádky textu jsou od sebe oddělené zakončovacím znakem, který je pro jednotlivé operační systémy rozdílný a při přenosu mezi operačními systémy je třeba na tuto problematiku pamatovat. Operační systém založený na Unixu standardně používá ukončovací znak „line feed“ (LF), zatímco operační systém Mac OS od firmy Apple používá „carriage return“ (CR) a operační systém MS-Windows od firmy Microsoft ukončuje řádky dvojicí znaků CR, LF. [3] Při přenosu textového soboru tedy může dojít k jeho nekorektnímu zobrazení v případě použití textového editoru, který nepodporuje či nemá nastavenou správnou detekci konce řádků. Nejpoužívanější znakové sady jsou ASCII a UNICODE. [4]

2.2.2 Binární data

Binární data jsou všechny soubory, které potřebují pro předání informace v nich obsažené nějaký nástroj, který je umí interpretovat. [2] Jako příklad lze uvést video z analogových kamer CCTV, které z digitalizujeme a máme ho binárně uložené na pevném disku. Jedná se tedy o binární data, které můžeme pomocí video přehrávače interpretovat a zobrazit jako video, ale bez video přehrávače se jedná pouze o posloupnost jedniček a nul, které nám mnoho informací nesdělí. [3] Na obrázku 2 je zobrazena fotka v binárním formátu.



Obr. 2. Zobrazení binárních dat přes příkaz xxd

2.3 Záznamové média

Záznamové medium slouží pro uložení dat a v nich obsažených informací pomocí fyzikálních principů a na nich vytvořených aplikačních použití. [4] Může se jednat o pevný disk, flash paměť, optická média či cloudové úložiště. Cloudové úložiště v době psaní této bakalářské práce získalo většinový podíl pro uchovávání dat z chytrých mobilních telefonů a operačních systémů v kterých jsou aplikace pro přístup k nim přímo integrované. [5]

2.3.1 Pevný disk

Pevný disk je záznamové medium fungující na principu uchování informace za pomoci změn magnetického pole. Každý pevný disk obsahuje minimálně jednu plotnu s magnetickým substrátem a hlavičku, která se po ní pohybuje. Díky změnám směru proudu v cívce umístěné na hlavičce, může hlavička zapisovat na plotnu data. Čtení dat funguje opačným způsobem, kdy se indukuje napětí na cívce hlavičky při průchodu magnetickým polem diskové plotny. [5] V době psaní bakalářské práce bylo možné pořídit pevný disk o velikosti 18TB.

2.3.2 Optický disk

Optický disk je záznamové medium fungující na principu uchování informace pomocí laseru, který upraví povrh optického disku takovým způsobem, aby změnil odraz světelného paprsku. Zápis probíhá od středu disku spirálovitě směrem až k jeho okraji. Čtení probíhá pomocí detekce změn odrazu světelného paprsku. [6] V době psaní bakalářské práce jsou optické disky na ústupu a používají se pouze okrajově pro zvykový záznam ve standardu CD a pro distribuci filmů a her ve standardu Blu-Ray.

2.3.3 Flash paměť

Flash paměť je záznamové médium fungující na stejném principu jako elektronicky mazatelná a programovatelná paměť. Skládá se z unipolárních tranzistorů se dvěma izolovanými hradly. Jedno hradlo slouží pro řízení a druhé pro uchování náboje, který reprezentuje uloženou informaci. [2] Flash paměť se rychle rozšířila a díky ní i došlo k vytlačení optických disků jako přenosových medii a jejich nahrazení flash disky.

2.3.4 Cloudové úložiště

Cloudové uložiště poskytují provozovatelé jako službu, pro ukládání dat. [7] V principu se jedná o sítově uložiště, ke kterému je vytvořená aplikační vrstva umožňující jednoduchý

přistup a ovládání z webového prohlížeče či speciální aplikace z jakéhokoliv místa, ve kterém je dostupné internetové připojení. [7] Poskytovatelé této služby většinou mají dostupnou kapacitu rozdělenou v několika tarifech. Základní tarif obsahující jednotky až desítky gigabajtů volného místa bývá zdarma a za další se musí platit měsíční poplatek. Samotné servery obsahující pevné případně polovodičové disky mohou být díky Internetu, který je decentralizovaný a globální, umístěné klidně na druhé straně planety. Na to poukazují odpůrci této služby, že ukládaná data jsou mimo stát v rukách soukromích společností a může dojít k jejich zneužití.

2.4 Databáze

Databáze je soubor dat, který je organizovaný a má definovanou strukturu. Slouží pro uchování dat a jejich následné znovu použití. [8] Samotný databázový soubor dat se spravuje specializovanými programy, které umožňují jeho editaci a čtení. Tyto programy se nazývají systém řízení báze dat. Pro práci se souborem dat, se nejčastěji používá strukturovaný dotazovací jazyk Structured Query Language (SQL). [9]

2.4.1 Jazyk SQL

Jazyk SQL se používá pro práci s databázemi, který jej podporují, jedná se v tomto ohledu především o relační databáze. Nejedná se však o plnohodnotný programovací jazyk, jako je například C++ či Java. Neobsahuje řídící konstrukce a správu paměti či podporu objektového programovaní jako již zmíněné programovací jazyky. [8]

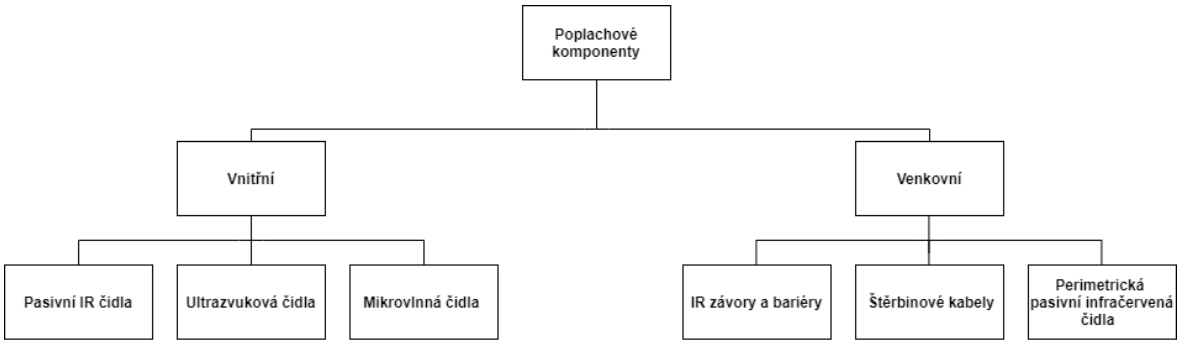
Je buď implementován přímo v rozhraní určeného pro správu databáze, například přes phpmyadmin nebo SQL Server Management Studio, které mají přívětivé grafické uživatelské prostředí. Případně lze pomocí konzole přímo psát příkazy a dotazovat se pomocí nich databázového serveru. Další častá varianta je integrace SQL do plnohodnotného programovacího jazyka, který skrz SQL příkazy přistupuje k dané databázi. Z tohoto pohledu se dá považovat SQL za dotazovací jazyk, v kterém je sice možné vytvářet sofistikované příkazy pro náhledy nad databází či její správu, ale sám není určený pro vytváření aplikačních řešení. [10] Na obrázku 3 je ukázka vytvoření databáze a tabulky pomocí jazyka SQL.

```
1   CREATE DATABASE components;
2   CREATE TABLE  alarms (
3       AlarmID int,
4       AlarmName varchar(255),
5       AlarmType varchar(255),
6       Code int,
7       Place varchar(255)
8   );
9
```

Obr. 3. Vytvoření databáze a tabulky

2.4.2 Hierarchická databáze

Hierarchická databáze představuje datový model organizovaný do stromové struktury. [8] Z toho vyplívá, že hierarchická databáze vyžaduje, aby měl každý prvek pouze jednoho nadřízeného. Nadřízený prvek může mít jeden či více podřízených prvků. Při čtení dat z hierarchické databáze je potřeba projít celou stromovou strukturu od kořenového uzlu. Historicky se jedná o první databázový model vytvořený společností IBM v 60. letech. [10] Na obrázku 4 je ukázka modelu hierarchické databáze na poplachových komponentech.



Obr. 4. Model hierarchické databáze

2.4.3 Síťová databáze

Síťová databáze představuje datový model, ve kterém jsou data organizována jako uzly. [8] Každý uzel může být spojený s jakýmkoliv počtem dalších uzlů. Takové spojení se tvoří jako rovinný graf. Informace, jak jsou jednotlivé uzly na sobě závislý určují ukazatele, kteří udávají, jaký uzel má na daný uzel návaznost. Při čtení dat se prochází graf cestou, která je tvořena ukazateli. [10]

2.4.4 Relační databáze

Relační databáze představuje datový model, ve kterém jsou data organizována do relací, ty jsou reprezentovány tabulkami obsahujícími určitý počet řádků a sloupců. Sloupec slouží jako atribut, tudíž má název a určující datový typ. Řádek je již samotný záznam uložený v relační databázi. [8] Mezi jednotlivými tabulkami je možné vytvářet pomocí primárního a cizího klíče vazby, které následně slouží jako propojení záznamů a určují poté vztahy jednotlivých tabulek. [10] Z toho vyplívá, že relační model má striktní politiku pro zachování integrity dat.

2.4.5 Objektová databáze

Objektová databáze představuje datový model, ve kterém jsou data organizována jako objekty. [8] Oproti relačním databázím je organizováním dat tímto způsobem zajištěna možnost velké škálovatelnosti. Každý objekt má identifikátor, který ukazuje na místo ve virtuální paměti, kde je objekt uložený. Z objektově orientovaného programování si bere principy vícenásobné dědičnosti, zapouzdření a polymorfismu. Pro manipulaci s daty se používají jazyky ODL (Object Definition Language) a OQL (Object Query Language). Ve většině moderních programovacích jazyků jako je Java, C#, C++ se používají pouze v knihovně určené pro práci s tímto typem databází a programátor používá pouze rozhraní poskytnuté tou danou knihovnou, která umožňuje objekt v databázi vytvořit stejně jako objekt v daném programovacím jazyce. [10] Na obrázku 5 je tento přístup ukázán v jazyce Java pomocí funkce, která vytvoří objekt z parametrů jí předaných, který následně uloží do databáze.

```
1   void storeWarning (string componentName, int warningNumber) {
2     Warning warning1 = new Warning(componentName, warningNumber);
3     db.Set(warning1);
4   }
5
```

Obr. 5. Funkce pro uložení parametrů do databáze

3 INTERPRETACE DAT

Intepretace dat slouží pro předaní informace, která je v datech uložená takovým způsobem, aby mohla být tato informace dále zpracovaná, například člověkem či programem nebo strojem. Jedná se o důležitý aspekt v informačních technologiích, se kterým se při používání výpočetní techniky setkáváme. Jako příklad lze uvést rozsvícení displeje u mobilního telefonu. Po zmáčknutí tlačítka dojde k vyvolání přerušení na procesoru způsobené změnou logické hodnoty na jeho vstupu a následném zavolání vektoru přerušení, který odkazuje na rutinu pro obsluhu přenosu dat mezi řadičem displeje a procesorem, v téže rutině může být načtený čas z jednotky reálného času, který se následně po zapsání do paměti řadiče displeje na samotném displeji zobrazí. Už jen při této naprosto obyčejné činnosti dochází k několika interpretacím dat, nejdříve stavu tlačítka, následně přečtením času z jednotky reálného času a pak samotný výstup na displeji, z toho lze vyvodit, že data se nějakým způsobem při používání výpočetní techniky interpretují. [11]

Při intepretaci dat z poplachových a neoplachových komponent lze uvažovat obdobným způsobem. Získaná data se můžou reprezentovat jako stav dané komponenty a ten může být interpretován ve vizualizaci přes webové rozhraní čí aplikaci pro osobní počítač nebo chytrý telefon.

3.1 Webová aplikace

Webové aplikace se poskytují uživatelům pomocí webového serveru. Jedná se o velmi rozšířenou formu poskytování různých aplikačních řešení pro interpretování dat, například z databázových serverů a multimediálních galerií. [12] Jejich vývoj akceleroval se zvyšujícím se počtem uživatelů Internetu, kdy klasické statické webové stránky přestaly splňovat požadavky pro interakci s uživatelem. Dnešní webové prohlížeče by se dali se svojí komplexitou přirovnat k operačním systémům, a dokonce i firma Google se svým operačním systémem Chrome OS to jen potvrzuje. [13] Webové aplikace se často vytvářejí pomocí hypertextového značkovacího jazyka, kaskádových stylů a Javascriptu.

3.1.1 Hypertextový značkovací jazyk

Hypertextový značkovací jazyk (HTML) je základ většiny webových stránek a webových aplikací. [12] Udává jim strukturu, jak budou zobrazovány. Jeho syntaxe se skládá ze značek, takzvaných tagů, které jsou uzavřeny v ostrých závorkách spolu s jejich vlastnostmi. Jeho vývoj je přizpůsobován aktuálním webovým požadavkům. [13] Na

obrázku 6 je ukázka jednoduché webové stránky, na které se zobrazí název této bakalářské práce spolu s jejím popisem.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>Bakalářská práce</title>
 </head>
 <body>
  <h1>Uchování a interpretace dat z poplachových a nepoplachových komponent přes protokol MQTT</h1>
  <p>Bakalářská práce se zaměřuje na možnosti uchování a interpretace dat z poplachových a nepoplachových
     komponent komunikujících přes standardizovaný otevřený protokol MQTT.</p>
 </body>
</html>

```

Obr. 6. Ukázka HTML

3.1.2 Kaskádové styly

Kaskádové styly (CSS) slouží pro definování jakým způsobem se bude element napsaný v HTML zobrazovat. [12] Může se tedy pomocí něj definovat například velikost textu, jeho font, barva a zarovnání. [13] Na obrázku číslo 7 je ukázka CSS.

```
body {
    background-color: rgb(150, 207, 224);
  }

h1 {
    color: rgb(0, 0, 0);
    text-align: left;
  }

p {
    font-family: verdana;
    text-align: left;
    font-size: 13px;
  }

```

Obr. 7. Ukázka CSS

3.1.3 Javascript

Javascript je programovací objektově orientovaný jazyk, který se především používá pro oživení a rozpohybování webových stránek a webových aplikací. [12] Jeho implementaci mají všechny webové prohlížeče a díky tomu je jeho použití velmi jednoduché. Programátor se nemusí starat o to, zda má uživatel nainstalované potřebné knihovny, stačí pouze definovat od jaké verze prohlížeče je webová stránka či aplikace otestována. Napsané funkce se vkládají jako script přímo do zdrojového kódu, případně se přidávají jako vzdálená knihovna, která se načítá spolu s webovou stránkou. Z toho vyplívá, že běh

Javascriptu je na straně uživatele na rozdíl od PHP, které se zpracovává na straně serveru. [13]

3.1.4 Node.js

Node.js je softwarové řešení pro spouštění webových aplikací napsaných v Javascriptu na straně serveru. Zajišťuje běh aplikace a následnou její distribuci jako službu. [14] Při psaní této bakalářské práce se jednalo o nejrozšířenější nástroj pro vývoj webových aplikací.

3.2 Aplikace pro osobní počítač

Aplikace pro osobní počítač tvoří jeho vybavení pro určitý druh činnosti, která se na něm provádí. Každý uživatel má své vlastní portfolio aplikací, které používá a do značné části i reflektuje zaměření a činnost daného uživatele. [15] Aplikace pro osobní počítač se vytváří pomocí frameworku, operačního systému a programovacího jazyka. Nejpoužívanější frameworky pro multiplatformní aplikace jsou Qt pro programovací jazyk C++ a Tkinter pro programovací jazyk Python.

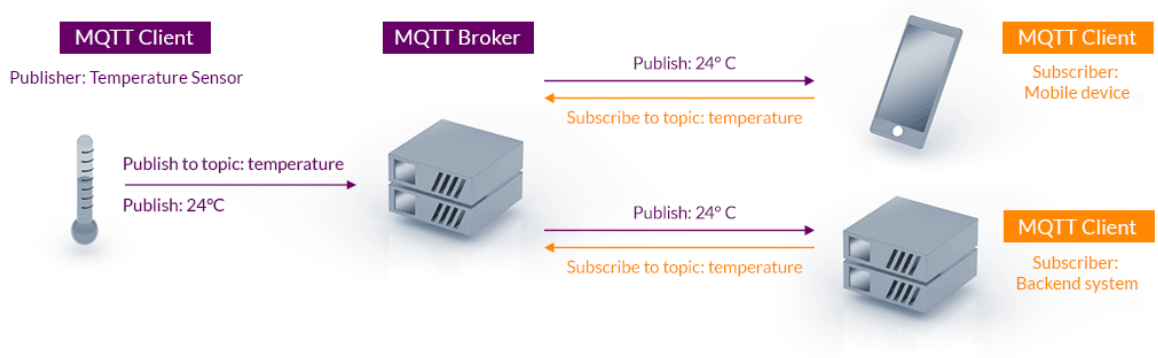
3.3 Aplikace pro chytrý telefon

Aplikace pro chytrý telefon pomalu nahrazují klasické aplikace pro osobní počítač i samotné osobní počítače. Důvodů je několik, jedním z nich je že chytrý mobilní telefon nosí všichni u sebe a je díky tomu hned dostupný. Díky neustále rostoucímu výkonu těchto zařízení, zvětšujícím se displejům a tím pracovní plochy se zážitek z používání aplikací na chytrém telefonu v některých případech dá srovnávat s tím, na co jsme zvyklý z osobních počítačů a některé aplikace například pro živé rozpoznávání textu a jeho překlad i překonávají klasické rozhraní mezi uživatelem a aplikací z osobních počítačů. [15] Aplikace se vytváří pomocí frameworku a programovacích jazyků. Pro operační systém Android to je například Android Studio s programovacím jazykem Java a Kotlin. Pro operační systém IOS se používá Xcode a programovací jazyk Swift.

4 PROTOKOL MESSAGE QUEUING TELEMETRY TRANSPORT

Protokol Message Queuing Telemetry Transport (MQTT) byl standardizován organizací Organization for the Advancement of Structured Information Standards (AOSIS) a mezinárodní organizací pro standardy jako ISO/IEC 20922. [16]

Jedná se o sítový protokol pro přenos malých datových zpráv mezi zařízeními. Obvykle běžící přes TCP/IP, případně jiný sítový protokol poskytující obousměrnou bezztrátovou komunikaci. [16] Na obrázku číslo 8 je funkční diagram protokolu MQTT.



Obr. 8. Funkční diagram protokolu MQTT [16]

4.1 Klient

Klient v komunikaci přes protokol MQTT může zastávat dvě role. První z nich je publikovatel (angl. Publish) [17], který do zprostředkovatele (angl. Broker) publikuje zprávy s obsahem, nebo odběratel který ze zprostředkovatele (angl. Broker) zprávy s obsahem odebírá. [18]

4.2 Zprostředkovatel

Zprostředkovatel (angl. Broker) zastává při komunikaci přes protokol MQTT roli prostředníka se kterým komunikují jeho klienti. [17]

4.3 Předmět komunikace

Předmět komunikace slouží pro určení, o jaké data se jedná. Zadává se ve formátu řetězce, ve kterém jsou jednotlivé stupně oddělené lomítkem. [17] Jako příklad lze uvést následující řetězec “chata/alarmy”. V případě, že publikovatel bude publikovat data s tímto

předmětem komunikace, tak odběratel je pomocí definování stejného řetězce může získat. [18]

4.4 Průběh komunikace

Průběh komunikace pomocí protokolu MQTT lze popsat následovně. Klient znázorňující roli odběratele, který chce odebírat určitá data identifikovaná předmětem komunikace ze zprostředkovatele, s ním naváže komunikaci. [17] Úspěšné připojení zprostředkovatel klientovi potvrdí. Poté klient informuje zprostředkovatele, jaký předmět chce odebírat. Tuto operaci zprostředkovatel také potvrdí. Zprostředkovatel si tuto informaci uloží a v případě že jsou do předmětu publikována data jiným klientem představujícím roli publikovatele, tak o tom zprostředkovatel informuje klienta odebírající daný předmět a data mu pošle. [18]

4.4.1 Klient – publikovatel

Klienta představující roli publikovatele lze názorně ukázat pomocí příkazu „mosquitto_pub“ v linuxovým operačním systému. Parametr „-h“ slouží pro definovaní IP adresy zprostředkovatele. [17] Parametr „-m“ udává publikovaná data. Parametr „-t“ udává adresu předmětu, kam se mají data publikovat. Parametr „-d“ značí debug výpis informací. [18] Na obrázku číslo 9 je průběh komunikace ukázán.

```
root@flowserver:~# mosquitto_pub -h localhost -m "Status=0xE3" -t chata/alarmy -d
Client mosqpub|16026-flowserve sending CONNECT
Client mosqpub|16026-flowserve received CONNACK (0)
Client mosqpub|16026-flowserve sending PUBLISH (d0, q0, r0, m1, 'chata/alarmy', ... (11 bytes))
Client mosqpub|16026-flowserve sending DISCONNECT
root@flowserver:~#
```

Obr. 9. Publikování dat přes protokol MQTT

4.4.2 Zprostředkovatel

Roli zprostředkovatele lze názorně ukázat pomocí příkazu „mosquitto“ v linuxovém operačním systému. Parametr „-v“ slouží pro spuštění výpisu. [17] Zprostředkovatel otevře port a na něm poslouchá požadavky pro odběr a publikování dat. [18] Na obrázku 10 je vidět průběh zpracování komunikace od klienta z kapitoly 4.4.1 publikující data.

```
root@flowserver:~# mosquitto -v
1618216325: mosquitto version 1.5.7 starting
1618216325: Using default config.
1618216325: Opening ipv4 listen socket on port 1883.
1618216325: Opening ipv6 listen socket on port 1883.
1618216519: New connection from ::1 on port 1883.
1618216519: New client connected from ::1 as mosqpub|16026-flowserve (c1, k60).
1618216519: No will message specified.
1618216519: Sending CONNACK to mosqpub|16026-flowserve (0, 0)
1618216519: Received PUBLISH from mosqpub|16026-flowserve (d0, q0, r0, m0, 'chata/alarmy', ... (11 bytes))
1618216519: Received DISCONNECT from mosqpub|16026-flowserve
1618216519: Client mosqpub|16026-flowserve disconnected.
```

Obr. 10. Zprostředkování komunikace přes protokol MQTT

4.4.3 Klient – odběratel

Klienta představující roli odběratele lze názorně ukázat pomocí příkazu „mosquitto_sub“ v linuxovým operačním systému. [17] Parametr „-h“ slouží pro definovaní IP adresy zprostředkovatele Parametr „-t“ udává adresu předmětu, který chce klient odebírat. Parametr „-d“ značí debug výpis informací. [18] Na obrázku číslo 11 je průběh komunikace ukázán. Klient představující roli odběratele si periodicky ověřuje dostupnost zprostředkovatele, který ji potvrzuje.

```
root@flowserver:~# mosquitto_sub -h localhost -t chata/alarmy -d
Client mosqsub|16159-flowserve sending CONNECT
Client mosqsub|16159-flowserve received CONNACK (0)
Client mosqsub|16159-flowserve sending SUBSCRIBE (Mid: 1, Topic: chata/alarmy, QoS: 0)
Client mosqsub|16159-flowserve received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|16159-flowserve received PUBLISH (d0, q0, r0, m0, 'chata/alarmy', ... (11 bytes))
Status=0xE3
Client mosqsub|16159-flowserve sending PINGREQ
Client mosqsub|16159-flowserve received PINGRESP
Client mosqsub|16159-flowserve sending PINGREQ
Client mosqsub|16159-flowserve received PINGRESP
```

Obr. 11. Odebírání dat přes protokol MQTT

5 VÝSTUP Z POPLACHOVÝCH A NEPOPLACHOVÝCH KOMPONENT

Výstup z poplachových a nepoplachových komponent je vždy informace, která nám reprezentuje stav dané komponenty. [19] V této kapitole si představíme poplachové a nepoplachové komponenty. Možnosti jejich připojení k nadřazeným systémům pomocí jejich analogových či digitálních výstupů.

5.1 Poplachová komponenta

Poplachová komponenta slouží pro ochranu života, majetku případně prostředí, v kterém je instalována. [19]

5.1.1 Režim zastřeženo

Režim zastřeženo je reprezentován softwarovým řešením nadřízeného systému, který vyhodnocuje stav poplachových a tísňových komponent. V režimu zastřeženo jsou nadřazeným systémem detekovány zóny a v nich jednotlivé komponenty, které v případě změny svého stavu mohou hlásit případné narušení chráněného prostoru či prostředí. Nadřazený systémem může tuto skutečnost reflektovat dle svého nastavení například vyhlášením poplachu a odesláním této informace do dohledového centra. [19]

5.1.2 Režim odstřeženo

Režim odstřeženo je reprezentován softwarovým řešením nadřízeného systému, který v případě detekování změny stavu zóny a v ní poplachové komponenty, kromě zaznamenání této skutečnosti nepodniká žádné další kroky. [19]

5.1.3 Příklad poplachové komponenty

Jako příklad poplachové komponenty lze uvést tísňové tlačítko HB 304. Je určené do prostředí, v kterém je potřeba mít možnost tajně vyhlásit poplach v případě mimořádné události. Jako příklad lze uvést čerpací stanice či směnárny měn, ve kterých v případě mimořádné situace nenápadné vyhlášení poplachu a tím přivolání pomoci může mít značný vliv na její průběh. Tísňové tlačítko HB 304 na obrázku 12 pracuje na stejnosměrné napětí 7 až 15 V a obsahuje vstup COM pro připojení požadovaného napětí určeného pro spínání výstupu, který lze připojit do smyčky NO či NC. [20]

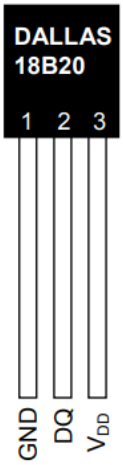
Obr. 12. Tísňové tlačítko HB 304

5.2 Nepoplachová komponenta

Nepoplachová komponenta nemá hlavní funkci v ochraně života, majetku či prostředí. [19] Může se jednat o zařízení pro měření teploty, vlhkosti vzduchu případně venkovního osvětlení.

5.2.1 Příklad nepoplachové komponenty

Jako příklad nepoplachové komponenty lze uvést teplotní senzor DS18B20, který díky komunikaci pomocí sběrnice 1 – Wire se skvěle hodí pro implementaci do každé místnosti. Pro svůj provoz vyžaduje tři dráty, v případě použití parazitního režimu postačí i dva. Teplotní rozlišení je od –55°C do +125°C s přesností půl stupně celsia ve středovém pásmu. [21] Na obrázku 13 je zobrazen senzor DS18B20 v provedení through-hole (THT).



Obr. 13. Teplotní senzor
DS18B20 [21]

5.3 Způsoby připojení

Poplachová i nepoplachová komponenta má výstup informaci o jejím stavu která lze přenést několika způsoby do nadřazeného systému. [19] Nadřazený systém může být představován ústřednou PZTS, případně PLC či jiným kontrolérem s výpočetní jednotkou.

5.3.1 Drátové smyčky

Připojení pomocí drátových smyček je realizováno pomocí tří vodičů. Dva z nich slouží pro napájení dané komponenty a pro napěťovou úroveň pro COM vstup. Třetí z nich tvoří smyčku, do které je možné připojit i více komponent dle projektové specifikace, která tvoří zónu, kterou následně vyhodnocuje nadřazený systém. Ten může měřit odpor dané smyčky v zóně a tím zjišťovat její stav. [19]

Smyčka Normally Closed (NC)

Smyčka Normally Closed (NC) je realizována pomocí sériového zapojení rozpínacích kontaktů poplachových a tísňových komponent. Nadřazený systém detekuje dva stavy, první z nich je v případě, kdy je smyčka sepnutá. Takový stav je považován za normální a reprezentuje klidový stav komponent. Druhý stav je vyvolán v případě, kdy je smyčka rozepnutá, tím komponenty předávají informaci o jejich aktivaci. [19]

Smyčka Normally Open (NO)

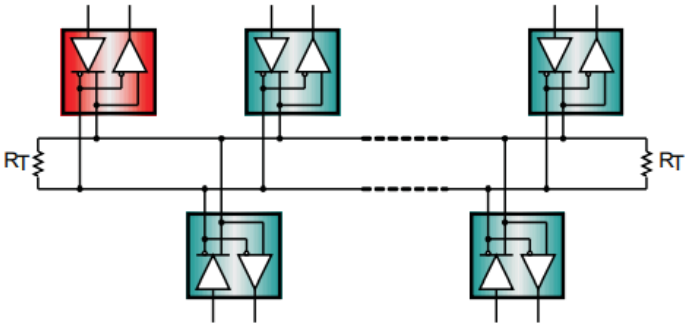
Smyčka Normally Open (NO) je realizována fyzicky stejným způsobem, jako smyčka NC s tím rozdílem, že nadřazený systém detekuje normální stav smyčky v případě jejího

rozepnutého stavu. V reálnem prostředí se tato smyčka používá pouze ve specifických případech, protože nelze pomocí jejího normálního stavu detekovat, zda například neodpadl drát, či útočník jí nepřestřihl. [19]

5.3.2 Sběrnicové zapojení

RS-485

Sériová sběrnice RS-485 slouží pro připojení až 32 zařízení definovaných adresou. [19] Samotná komunikace probíhá po dvou drátech bud polovičním duplexem. To znamená že zařízení mohou přijímat i vysílat data, ale nemohou to dělat zároveň. Případně je možnost rozšíření této sběrnice na plně duplexní při použití čtyř datových vodičů. Přenos je prováděn rozdílem napětí na vodičích tedy diferenciálně, pro delší vzdálenosti je nutné kvůli rozdílu potenciálu propojit i zemnicí vodič. Standard nedefinuje protokol přenosu dat pouze fyzickou vrstvu. Nejčastějším protokolem pro přenos dat po sériové sběrnici RS-485 je Modbus a vzdálenost pro komunikaci pomocí této sběrnice je ve stovkách metrů. [22] Na obrázku 14 je ukázka zapojení zařízení pomocí sběrnice RS-485.

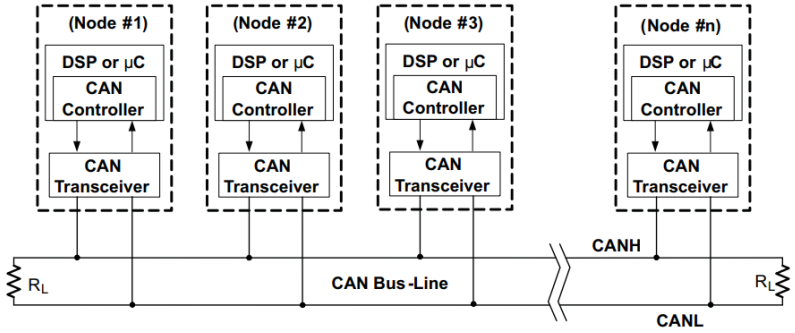


Obr. 14. Zařízení na sběrnici RS-485 [22]

Controller Area Network

Sériová sběrnice Controller Area Network (CAN) slouží pro připojení zařízení, které jsou kriticky závislé na přenosu dat. Již na úrovni přenosu jsou datové rámce s vyšší prioritou určené menším identifikátorem datového rámce přeneseny přednostně. To je umožněno tím, že zařízení jsou na sběrnici synchronizované a vzorkují současně. V případě kolize při vysílání dat, tedy zařízení snažící se odeslat rámec s vyšším identifikátorem vysílání zanechá a pokusí se o to později. Podle specifikace se používá bud 11bitový identifikátor datového rámce v rozšířené variantě 29bitový. Samotný datový rámec je tvořen až osmi

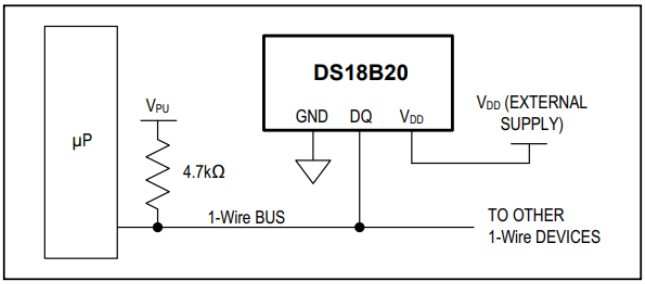
bajty. Pro přenos se používají dva dráty a je možný podle přenosové rychlosti až na desítky metrů. Tato sběrnice se díky svým vlastnostem nejvíce používá v automobilovém a leteckém průmyslu. [23] Na obrázku 15 je ukázka zařízení komunikujících pomocí sběrnice CAN.



Obr. 15. Funkční diagram sběrnice CAN [23]

1 – Wire

Sériová sběrnice 1 – Wire slouží pro připojení k zařízením, pro které není kritická rychlost přenosu dat, ale jednoduchá možnost jejich připojení k řídícímu systému. Samotná komunikace probíhá po jednom drátu s využitím zemnícího vodiče a pull-up odporu. [24] Jednotlivá zařízení připojená ke sběrnici jsou identifikována podle 64bitového sériového čísla. [21] To umožnuje mít na sběrnici připojeno několik zařízení. Délka drátu je závislá na úbytku napětí mezi řídícím systémem a připojenými zařízeními. Z toho vyplývá, že požadavky pro zařízení nejsou velké, a proto je tato sběrnice hojně využívaná v elektronických součástkách pro měření teploty, pro přístup k EEPROM paměti a pro identifikační čipy obsahující unikátní řetězce dat. [24] Na obrázku 16 je ukázka zařízení komunikující pomocí sběrnice 1 - Wire.



Obr. 16. Připojení zařízení na sběrnici 1 – Wire [24]

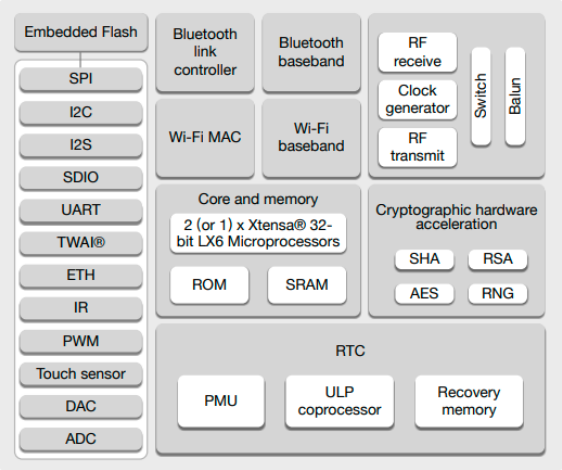
6 NÁVRH ŘEŠENÍ PRO ZÍSKÁVANÍ INFORMACÍ

Pro získávání informací z poplachových a nepoplachových komponent bude použit mikrokontroler ESP32 od firmy Espressif z důvodu jeho technických parametrů, které naprosto vyhovují požadavkům pro získávání informací z poplachových a nepoplachových komponent. Jako další jeho výhoda oproti jiným mikrokontrolerům je cena pohybující se v desítkách korun, kvalitní technická dokumentace, aktivní komunita vývojářů a dostupnost v obchodech zaměřujících se na prodej elektrických součástek. Mikrokontroler ESP32 má v sobě integrovaný Wi-Fi a Bluetooth modul pro připojení k síti a k dalším zařízením. Obsahuje také dostatek GPIO pinů pro připojení vstupů od poplachových a nepoplachových komponent. [25]

Poplachové a nepoplachové komponenty budou připojeny k řídící desce obsahující mikrokontroler ESP32 pomocí smyčky, případně pomocí datové sběrnice 1 – Wire, či CAN. Řídící deska bude připojena přes Wi-Fi k Internetu a dále bude komunikovat přes protokol MQTT se serverem. Server bude hrát roli zprostředkovatele (angl. Broker) a bude poskytovat patřičnou službu pro zpracování MQTT komunikace. Řídící deska bude v této komunikaci tedy publikovatel (angl. Publish), který získaná data z poplachových a nepoplachových komponent bude publikovat k zprostředkovateli dle komunikačního standardu protokolu MQTT.

6.1 Mikrokontroler ESP32

Mikrokontroler ESP32 od firmy Espressif je nástupce velmi používaného mikrokontroleru ESP8266, který lze nalézt ve spoustě zařízení vyžadujících připojení přes Wi-Fi. Jako příklad lze uvést chytrá žárovka či meteostanice. Oproti předchůdci je vyroben modernější 40nm technologií, díky které má menší spotřebu a lepší tepelné vlastnosti. [25] Mikrokontroler v sobě již obsahuje moduly pro Wi-Fi a Bluetooth spojení, které se skvěle hodí pro použití tohoto mikrokontroleru v IoT odvětví, kde nízká spotřeba a konektivita patří mezi klíčové vlastnosti. [26] Na obrázku 17 je kompletní hardwarová specifikace mikrokontroleru ESP32.



Obr. 17. Specifikace mikrokontroleru ESP32 [27]

6.1.1 Řídící deska Lolin D32

Řídící deska Lolin D32 od firmy Wemos je založená na mikrokontroleru ESP32. Z toho přebírá možnosti periferií a konektivity. Je určená pro instalace v prostředí, kde je potřeba mít napájení zálohované baterií. Proto je vybavena napěťovým regulátorem, pomocí kterého může fungovat s lithium-iontovými bateriemi. Tento obvod také zajišťuje nabíjení baterie v případě dostupnosti napájení z hlavního zdroje. Obsahuje též ochranu baterie proti úplnému vybití, které by jí mohlo poškodit. [28] Na obrázku 18 je řídící deska Lolin D32 zobrazena.

Obr. 18. Deska Lolin D32

Tab. 1. Technická specifikace Lolin D32 [28]

Technické specifikace	
Operační napětí	**3.3V**
Podporovaný typ baterií	**Lipo 3.7V**
Konektor pro připojení baterie	**PH-2 2.0mm**
Počet digitální I/O pinů	**22**
Počet vstupních analogových pinů	**6 (VP, VN, 32, 33, 34, 35)**
Počet výstupních analogových pinů	**2 (25, 26)**
Pin statusové LED diody	**GPIO5**
Frekvence procesoru	**240MHz**
Flash pamět	**4Mb**
Velikost	**57*25.4mm**
Váha	**6.1g**

6.2 Specifikace desky pro získávání dat

Specifikace desky pro získávání informací slouží pro definování požadavků, které by měla deska splňovat.

6.2.1 Specifikace rozměrů

Navrhovaná deska by měla být rozměrově do 10 cm na šířku a 10 cm na výšku.

6.2.2 Specifikace napájení

Deska pro získávání dat by měla být provozována při napětí od 5 V do 25 V.

6.2.3 Specifikace signalizace stavů

Deska pro získávání dat by měla obsahovat signalizační led diody pro informování o stavu připojení k Wifi síti, provozu z baterie a běhu firmwaru.

6.2.4 Specifikace binárních vstupů

Deska pro získávání dat by měla obsahovat diskrétní vstupy oddělené pomocí optočlenů od řídící desky pro připojení poplachových a nepoplachových komponent.

6.2.5 Specifikace analogových vstupů

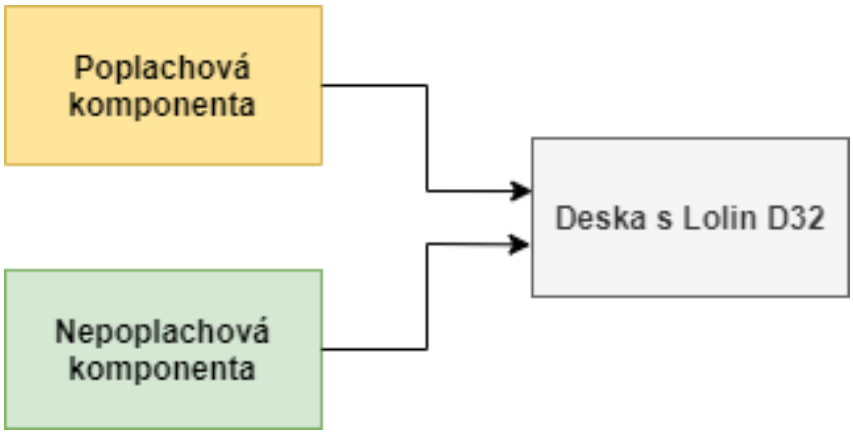
Deska pro získávání dat by měla mít dva analogové vstupy pro měření stavu připojené baterie a pro vyhodnocování odporově vyvážené smyčky, tento vstup by měl také obsahovat možnost kalibrace, pro různé typy odporově vyvážených smyček pomocí potenciometru.

6.2.6 Specifikace datových sběrnic

Deska pro získávání dat by měla obsahovat podporu pro datové sběrnice CAN a 1 – Wire pro připojení komunikace s poplachovými a nepoplachovými komponentami.

6.3 Vývojový diagram pro získání dat

Vývojový diagram pro získání dat je znázorněn na obrázku číslo 19.



Obr. 19. Vývojový diagram pro získání dat

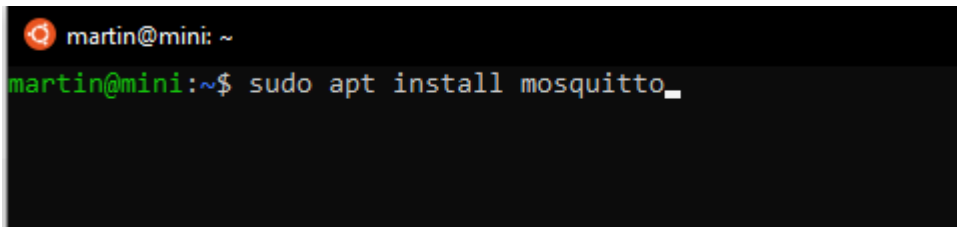
7 NÁVRH ŘEŠENÍ PRO UCHOVÁNÍ DAT

Uchování dat z poplachových a nepoplachových komponent bude realizováno pomocí serveru s Linuxovým operačním systémem z důvodu jeho otevřené licence nezatížené žádnými poplatky za jeho využívání. Díky této licenci má Linux velmi dobrou podporu vývojářů programů se stejnou licenční politikou. Jako další z důvodů je výborná integrace programovacího jazyka Python, databázových systémů a pokročilý balíčkovací systém. Na serveru bude databáze se skriptem napsaným v programovacím jazyce Python. Tento skript bude sloužit pro obsluhu databáze a služby pro zpracování MQTT komunikace. [29]

Pro zpracování MQTT komunikace bude použit zprostředkovatel Eclipse Mosquitto a pro uchování dat knihovna SQLite s pomocí které bude vytvořena a obsluhována databáze.

7.1 Eclipse Mosquitto

Eclipse Mosquitto je zprostředkovatel pro zpracování MQQT komunikace na straně serveru. Jedná se o program s otevřeným zdrojovým kódem, v kterém jsou implementované potřebné funkce pro zpracování MQTT protokolu. Jedná se o zprostředkovatele s velmi širokou podporou zařízení, od malých jednodeskových počítačů až po nasazení v dedikovaných serverech. [30] Většina Linuxových distribucí ve svých repositářích tento program obsahuje, proto je jeho instalace velmi jednoduchá a stačí použít pouze balíčkovací sytém s potřebným parametrem viz. Obrázek 20.



Obr. 20. Instalace Mosquitto

7.2 SQLite

SQLite je knihovna implementující kompletní relační databázový systém, [8] který je optimalizován pro použití s aplikacemi, které potřebují uchovávat data v souboru na lokálním stroji na kterém je daná aplikace spuštěná. Knihovna je uveřejněná pod licencí volného díla, díky které nepodléhá autorským právům a může být implementována do jakýchkoliv projektů. Instalace knihovny je velmi jednoduchá. [31] V Linuxovém prostředí bývá již obsažena v základních programech všech hlavních distribucí případně lze

doinstalovat jedním příkazem z repositářů pomocí balíčkovacího systému viz. Obr. 20 pouze se změnou parametru pro instalaci „sqlite3“.

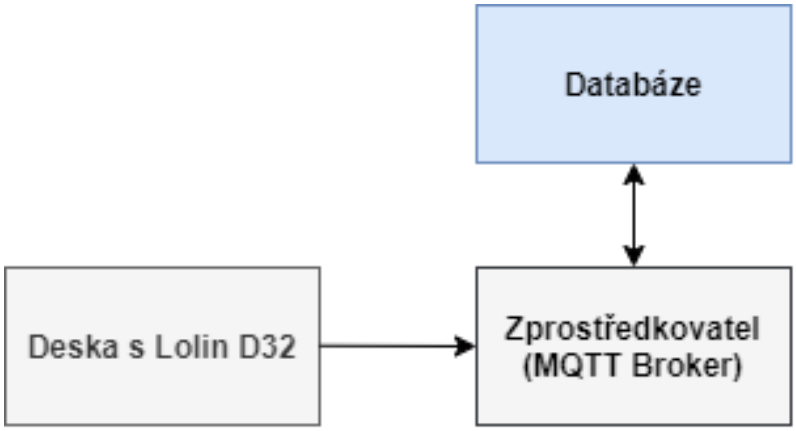
7.3 Specifikace pro uchování dat

Uchování a zpracování dat by mělo probíhat na serveru s veřejnou IP adresou, díky které se k ní budou moct připojovat klienti pomocí Internetu na přesně stanovený otevřený port. Samotné ukládání dat by mělo být prováděno aplikací bez správcovských oprávnění z důvodu bezpečnosti.

Databáze vytvořená pro uchování dat přes protokol MQTT by měla obsahovat sloupce pro všechny přenášená data s časovým razítkem pro jejich případné pozdější zpracování.

7.4 Vývojový diagram pro uchování dat

Vývojový diagram pro uchování dat je znázorněn na obrázku číslo 21.



Obr. 21. Vývojový diagram pro uchování dat

8 NÁVRH ŘEŠENÍ PRO INTERPRETACI DAT

Interpretace dat z poplachových a nepoplachových komponent bude realizována pomocí grafické uživatelské aplikace napsané v programovacím jazyce Python. Tato aplikace bude dle protokolu MQTT odběratelem (angl. subscriber) dat ze zprostředkovatele (angl. Broker). Programovací jazyk Python bude využit z důvodu jeho vlastností jako je podpora v operačních systémech, kde stačí pouze nainstalovat program pro jeho intepretaci. Pro vytvoření grafického uživatelského rozhraní bude použita knihovna Tkinter, která slouží pro vytváření grafických aplikací v programovacím jazyku Python. [32]

8.1 Knihovna Tkinter

Tkinter je základní knihovna pro vytváření grafických aplikací v programovacím jazyce Python. Jeho největší předností je jednoduchost, se kterou se dají grafické aplikace vytvářet. Pro jednoduchou grafickou aplikaci stačí pouze importovat knihovnu a poté zadat jednotlivé prvky se souřadnicemi, kde se mají prvky zobrazit. [33] Knihovna podporuje všechny prvky, které jsou potřeba pro vytvoření plnohodnotného uživatelského rozhraní. Jedná se například o tlačítka, textové vstupy, popisky, obrázky a grafy.

8.2 Specifikace aplikace pro interpretaci dat

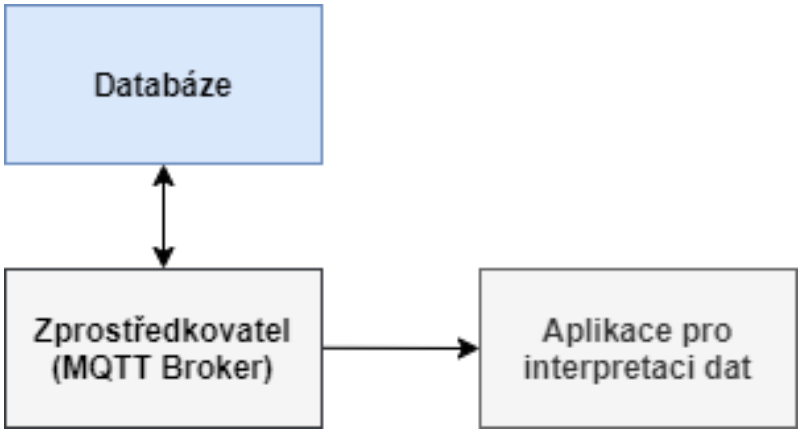
Specifikace aplikace pro interpretaci dat slouží pro definování požadavků, které by měla výsledná aplikace splňovat.

8.2.1 Specifikace grafického prostředí

Aplikace by měla obsahovat prvky pro interpretaci dat získaných přes protokol MQTT a také obsahovat stavovou informaci o připojení ke zprostředkovateli. Velikost okna aplikace by měla korespondovat s využitými grafickými prvky tak, aby byla zajištěna jejich přehlednost.

8.3 Vývojový diagram interpretace dat

Vývojový diagram pro interpretaci dat je znázorněn na obrázku číslo 22.



Obr. 22. Vývojový diagram interpretace dat

II. PRAKTICKÁ ČÁST

9 REALIZACE NÁVRHU PRO ZÍSKÁVÁNÍ DAT

Získávání dat z poplachových a nepoplachových komponent je realizováno pomocí desky ALANA CR s řídící deskou Lolin D32. Název desky je vytvořen ze začátečních písmen anglických slov „**Al**aram **a**nd **N**on **A**laram **C**omponents **R**eader“.

9.1 Vytvoření schématu pro desku

Schéma pro desku ALANA CR je vytvořeno v programu KiCAD. Schéma desky je tvořeno pomocí symbolů znázorňujících použité součástky, textových popisů a symbolů napájení. Mezi těmito symboly je vytvořeno dle funkční logiky a katalogových listů spojení tak, aby výsledné schéma tvořilo funkční celek dle specifikace uvedené v teoretické části. Použité součástky jsou převážně pro povrchovou montáž s rozměrem 1206 pro jejich jednoduché osazení. Pro napájení řídící desky byl zvolen lineární stabilizátor L7805, který díky jeho malým rozměrům, provozní teplotě a výstupnímu napětí a proudu plně vyhovuje pro použití na desce ALANA CR. Na diskrétní vstupy jsou použity optočleny LTV-356T, kteří mají tranzistorový výstup vhodný pro sepnutí vstupního pinu řídící desky s mikrokontrolerem ESP32. Pro datovou komunikaci CAN je zvolen převodník SN65HVD230 kompatibilní s rozhraním mikrokontroleru. Konektory pro připojení napájení, datových sběrnic a vstupů jsou zvoleny s ohledem na velikost desky a použití s kabeláží s průřezem do 0,5mm. Veškeré použité součástky jsou uvedeny v tabulce 2. Celé schéma desky je v příloze 1.

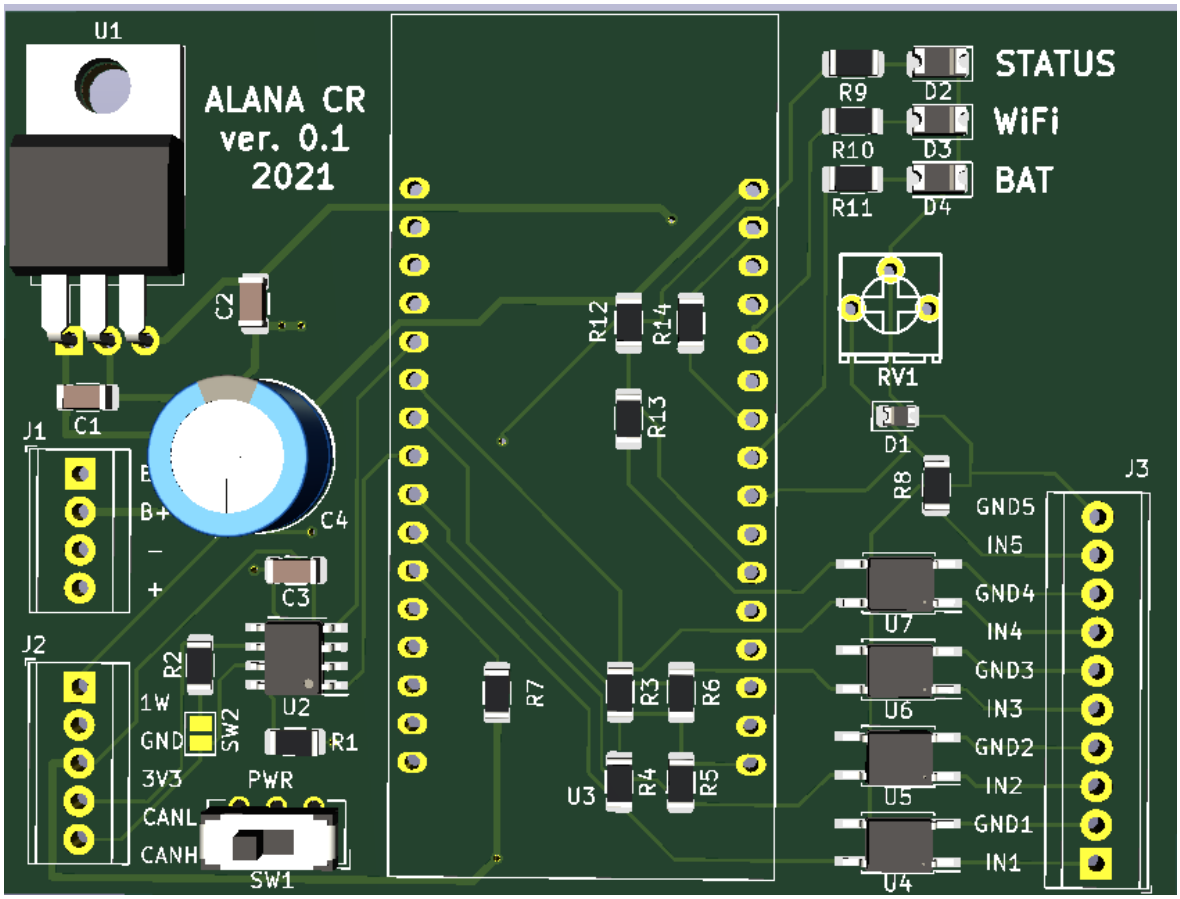
Tab. 2. Použité součástky pro desku ALANA CR

Reference	Počet	Hodnota	Pouzdro
C1	1	330nF	Capacitor_SMD:C_1206_3216Metric
C2 C3	2	100nF	Capacitor_SMD:C_1206_3216Metric
C4	1	220mF	Capacitor_THT:CP_Radial_D10.0mm_P3.50mm
D1	1	D_Zener	Diode_SMD:D_0805_2012Metric
D2 D3 D4	3	LED	Diode_SMD:D_1206_3216Metric
J1	1	Conn_01x04	TerminalBlock_TE-Connectivity:TerminalBlock_TE_282834-4_1x04_P2.54mm_Horizontal
J2	1	Conn_01x05	TerminalBlock_TE-Connectivity:TerminalBlock_TE_282834-5_1x05_P2.54mm_Horizontal
J3	1	Conn_01x10	TerminalBlock_TE-Connectivity:TerminalBlock_TE_1-282834-0_1x10_P2.54mm_Horizontal
R12 R13	2	47k	Resistor_SMD:R_1206_3216Metric
R2	1	120	Resistor_SMD:R_1206_3216Metric

R1 R3 R4 R5 R6 R14	6	10k	Resistor_SMD:R_1206_3216Metric
R7	1	4k7	Resistor_SMD:R_1206_3216Metric
R8	1	9k1	Resistor_SMD:R_1206_3216Metric
R10	1	470	Resistor_SMD:R_1206_3216Metric
R9 R11	3	1k	Resistor_SMD:R_1206_3216Metric
RV1	1	4k7	bp:Bourns_3362P
SW1	1	SW_DPDT_x2	Button_Switch_THT:SW_CuK_JS202011CQN_DPDT_Straight
SW2	1	SW_DIP_x01	Jumper:SolderJumper-2_P1.3mm_Open_Pad1.0x1.5mm
U1	1	L7805	Package_TO_SOT_THT:TO-220-3_Horizontal_TabDown
U2	1	SN65HVD230	Package_SO:SOIC-8_3.9x4.9mm_P1.27mm
U3	1	Lolin_D32	bp:DIP-32_885_ELL
U4 U5 U6 U7	4	LTV-356T	Package_SO:SO-4_4.4x3.6mm_P2.54mm

9.2 Vytvoření desky plošných spojů

Deska plošných spojů je vytvořena v programu KiCAD pomocí jeho funkce pro návrh desek plošných spojů. Dle využitých součástek a požadovaných rozměrů ve specifikaci v teoretické části bylo zvoleno dvouvrstvé řešení. Na vrchní vrstvě jsou umístěny SMD součástky spolu s THT, které se však pájí ze spodní strany. Pří vytváření bylo myšleno na to, aby rychlé sběrnice jako například CAN měly co nejkratší cesty, kladné napětí a zem vedly vedle sebe, aby nevznikaly rušící smyčky a rozlitá zem ve spodní vrstvě sloužila pro nejkratší možnou cestu pro tekoucí proud. Na obrázku 23 je zobrazena 3D vizualizace vytvořené desky plošných spojů.



Obr. 23. Deska plošných spojů ALANA CR

9.3 Výroba desky plošných spojů

Výroba desky ALANA CR byla provedena ve firmě PragoBoard s.r.o pomocí pool servisu.

Výsledná cena je tvořena velikostí desky, počtem vrstev a použitými technologiemi. Deska byla zadána do výroby podle specifikací uvedených v tabulce 3. Pro výrobu byla vygenerována data ve formátu gerber uvedená v příloze 2. Cena byla stanovena na 700 Kč bez daně.

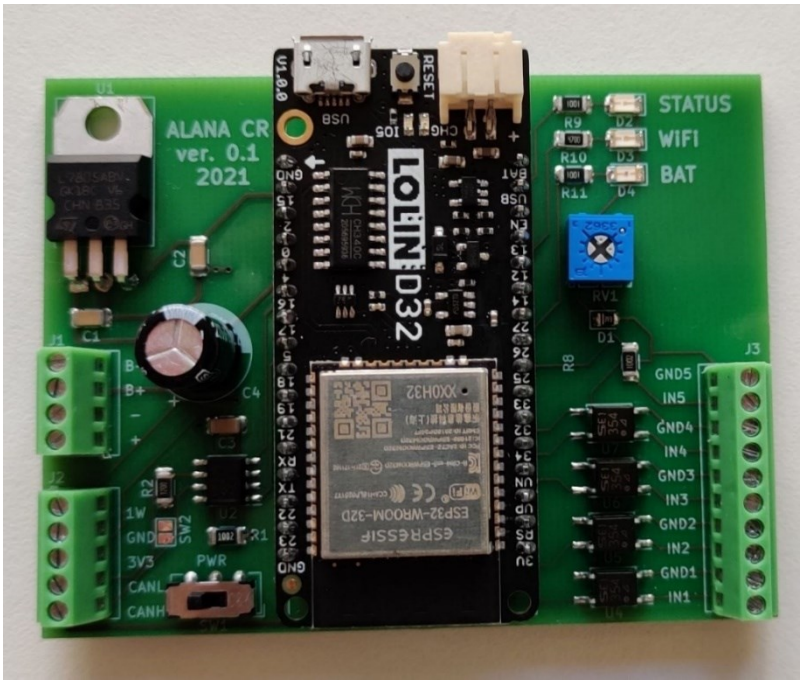
Tab. 3. Výrobní specifikace

Specifikace	Hodnota
Název desky plošných spojů	ALANA_CR
Rozměry	58.7x76.7 mm
Počet vrstev	2
Materiál	Isola DE104 – Tg 135 °C 1,6 mm

Tloušťka vnější základní měděné vrstvy	18um
Výroba	Pool servis
Průměr vrtání	> 0.3 mm min. průměr vrtání, >150um min. spoj/mezera/mezikruží
Maska	2x zelená nepájivá maska
Potisk	1x bílý servisní potisk
Technologie	Nanesení roztavené cínové pájky, elektrický test

9.3.1 Osazení desky plošných spojů

Osazení desky plošných spojů bylo provedeno za použití pájecí pasty, pájecí pece a pájky s cínem. Deska plošných spojů byla vyčištěna od nečistot a mastnoty, poté pomocí folie vytvořené z Gerber dat pomocí laseru byla na přesně určená místa nanesena pájecí pasta. Na pájecí pastu byly položeny pinzetou jednotlivé SMD součástky dle seznamu v tabulce 2. Po osazení SMD součástek byla deska umístěna do pájecí pece, v které prošla pájecí procedurou. Ostatní součástky typu THT byly poté zapájeny pomocí pájky ručně. Na obrázku číslo 24 je výsledná osazená deska ALANA CR.



Obr. 24. Osazená deska ALANA CR

9.4 Vytvoření firmwaru

Firmware pro desku ALANA CR je vytvořen v programovacím jazyku C++ za využití knihoven a uživatelského rozhraní programu Arduino IDE. Kód je rozdělen do tří hlavních sekcí. Jednotlivé sekce jsou popsány dále v této kapitole. Celý kód je uveden v příloze v souboru ALANA_CR_FW_ver.0.1.ino.

9.4.1 První sekce

V první sekci jsou uvedené používané knihovny, definovány parametry jako čísla pinů desky, rozdělení EEPROM paměti a IP adresa zprostředkovatele. Na obrázku číslo 25 je ukázka kódu z této sekce.

```
#include <CAN.h>
#include <DallasTemperature.h>
#include <PubSubClient.h>
#include <EEPROM.h>
#include <OneWire.h>
#include <WiFi.h>
#include <WiFiClient.h>

#define EEPROM_SIZE 256
#define WIFI_NAME_SIZE 32
#define WIFI_PASSWORD_SIZE 64

#define WIFI_NAME_EEPROM_ADDRESS 0
#define WIFI_PASSWORD_EEPROM_ADDRESS 32

#define INPUTS_EEPROM_ADDRESS 99
#define INPUT_ADC_EEPROM_ADDRESS_LSB 100
#define INPUT_ADC_EEPROM_ADDRESS_MSB 101
#define BATTERY_EEPROM_ADDRESS 102
#define ONEWIRE_EEPROM_ADDRESS 103
#define CAN_EEPROM_ADDRESS 104

#define STATUS_LED 23
#define WIFI_LED 12
#define BATT_LED 26

#define INPUT_1 19
#define INPUT_2 18
#define INPUT_3 17
```

Obr. 25. Ukázka první sekce firmwaru

9.4.2 Druhá sekce

V druhé sekci pojmenované „setup“ je provedena inicializace používaných periferií, inicializace EEPROM paměti a nastavení režimu Wi-Fi modulu. Na obrázku číslo 26 je ukázka kódu z této sekce.

```
// Init LEDs
pinMode(STATUS_LED, OUTPUT);
digitalWrite(STATUS_LED, HIGH);
pinMode(WIFI_LED, OUTPUT);
digitalWrite(WIFI_LED, LOW);
pinMode(BATT_LED, OUTPUT);
digitalWrite(BATT_LED, LOW);

// Init Optocouplers
pinMode(INPUT_1, INPUT);
pinMode(INPUT_2, INPUT);
pinMode(INPUT_3, INPUT);
pinMode(INPUT_4, INPUT);

// Init power detection
pinMode(BATT_MODE, INPUT);

// Set Wifi mode
Serial.print("Connecting to ");
Serial.println(ssidClient);

WiFi.mode(WIFI_STA);
WiFi.begin(ssidClient.c_str(), passClient.c_str());
```

Obr. 26. Ukázka druhé sekce firmwaru

9.4.3 Třetí sekce

Ve třetí sekci pojmenované „loop“ je nekonečná smyčka pro zpracování vstupů, vyhodnocení napětí baterie, zpracování sběrnic a komunikace se zprostředkovatelem. Na obrázku číslo 27 je ukázka kódu z této sekce.

```
input_1 = !digitalRead(INPUT_1);
input_2 = !digitalRead(INPUT_2);
input_3 = !digitalRead(INPUT_3);
input_4 = !digitalRead(INPUT_4);
input_5 = analogRead(INPUT_5);

batt_mode = !digitalRead(BATT_MODE);

int packetSize = CAN.parsePacket();
if (packetSize) {
  Serial.print(CAN.packetId(), HEX);
  Serial.println(CAN.packetDlc());
  while (CAN.available()) {
    Serial.print((char)CAN.read());
  }
  Serial.println();
}

inputs = input_1 << 0 | input_2 << 1 | input_3 << 2 | input_4 << 3 | batt_mode << 4;
```

Obr. 27. Ukázka třetí sekce firmwaru

10 REALIZACE NÁVRHU PRO UCHOVÁNÍ DAT

Pro uchování a zpracování dat byl na serveru s veřejnou IP adresou nainstalován zprostředkovatel Eclipse Mosquitto a spuštěn jako služba na portu 1883. Zprostředkovatel komunikace přes protokol MQTT na tomto portu poslouchá příchozí požadavky pro komunikaci. Pro uchování dat získaných přes protokol MQTT byla vytvořena aplikace, která odebírá ze zprostředkovatele přijatá data a ukládá je do relační databáze pomocí knihovny SQLite.

10.1 Aplikace pro uchování dat

Aplikace pro uchování dat pojmenovaná mqqtt_to_db ukládá data získaná přes protokol MQTT do relační databáze pomocí knihovny SQLite. Aplikace je rozdělená do čtyř sekcí, ve kterých se provádí například specifikace knihoven, navázání komunikace se zprostředkovatelem a ukládání dat do databáze. Celý kód je uveden v příloze v souboru mqtt_to_db.py.

10.1.1 Knihovna SQLite

Knihovna SQLite vložená do aplikace používající programovací jazyk Python umožnuje vytvářet databáze a pracovat s nimi pomocí jazyka SQL. V aplikaci byl vytvořen objekt z této knihovny s funkcemi, pomocí kterých lze provádět SQL příkazy nad databází. Mezi tyto příkazy patří „execute“ pro vykonání SQL příkazu zadaného v parametru volání této funkce a „commit“ pro zapsání změn do databáze.

10.1.2 Vytvoření tabulky

Vytvoření tabulky se provádí přes SQL příkaz „CREATE TABLE mqtt (sloupce)“ s uvedenými sloupci korespondujícími s daty získanými přes protokol MQTT a jejich identifikátorem spolu s časovým razítkem. Na obrázku číslo 28 je ukázán výřez z vytváření tabulky.

```
try:
    connToDB.execute('''CREATE TABLE mqtt (id text, time text, power int, batt int,
    conn.commit()
except sqlite3.Error as err:
    print("Create table: " + str(err))
```

Obr. 28. Vytvoření tabulky

10.1.3 Vložení dat to databáze

Vložení dat do databáze se provádí přes SQL příkaz „INSERT INTO mqtt (sloupce) VALUES (hodnoty)“ s uvedenými sloupci a daty určenými pro uložení do databáze. V aplikaci jsou to data získaná přes protokol MQTT. Na obrázku 29 je ukázán výřez z ukládání dat do databáze.

```
connToDB.execute("INSERT INTO mqtt (id, time, power, batt, in1, in2, in3, in4, in5, onewire, can) \
                 VALUES (?,?,?,?,?,?,?,?,?,?,?)",(str(msg.topic), now.strftime("%d/%m/%Y %H:%M:%S"),
                 batt, batt_val, input1, input2, input3, input4, input5, onewire, can))
conn.commit()
```

Obr. 29. Vložení dat do tabulky

10.1.4 Získání dat z databáze

Získání dat zapsaných v databázi se provádí přes SQL příkaz „SELECT * FROM mqtt WHERE id“. Na obrázku číslo 30 je ukázáno získání dat z tabulky, ve které jsou uložená data získaná přes protokol MQTT.

```
try:
    connToDB.execute("SELECT * FROM mqtt WHERE id = \'" + str(msg.topic) + "\';")
    response = connToDB.fetchall()
    print(response)
except sqlite3.Error as err:
    print("Select: " + str(err))
```

Obr. 30. Získání dat z tabulky

11 REALIZACE NÁVRHU PRO INTERPRETACI DAT

Pro interpretaci dat je vytvořená aplikace ALANA CR viewer, ve které se zobrazují data získaná přes protokol MQTT ze zprostředkovatele. Aplikace je rozdělná do dvou vláken. V prvním vlákně se zpracovává komunikace přes protokol MQTT, konkrétně odběr dat. V druhém vlákně je zpracováván grafický výstup. Celý kód je uveden v příloze v souboru ALANA_CR_viewer.py.

11.1 Inicializace

Inicializace slouží pro nakonfigurování jednotlivých prvků aplikace. Jako například označení názvů, které se mají zobrazit a na jakých souřadnicích se mají zobrazit získaná data pro intepretaci. Uživatelské rozhraní je zvoleno tak, aby bylo jasně určeno kam jednotlivé informace získané přes protokol MQTT dle logické návaznosti patří. Na obrázku 31 je ukázka kódu sloužící pro inicializaci.

```
# Init gui
gui.title('ALANA CR viewer')
gui.geometry('400x380')
gui.resizable(False, False)

# Init elements
frame = Frame(height = 350, width = 385, bd = 3, relief = 'groove').place(x = 7, y = 25)
Label(text = "ALANA CR viewer").place(x = 7, y = 4)
Label(text = "ver. 0.1").place(x = 350, y = 4)
Label(text = "Status:", font =('Helvetica', 14, 'bold')).place(x = 15, y = 32)
Label(text = "Battery", font =('Helvetica', 12, 'bold', 'underline')).place(x = 15, y = 65)
Label(text = "Running on battery :", font =('Helvetica', 12, 'bold')).place(x = 35, y = 90)
Label(text = "Battery charged :", font =('Helvetica', 12, 'bold')).place(x = 35, y = 115)
Label(text = "Inputs", font =('Helvetica', 12, 'bold', 'underline')).place(x = 15, y = 140)
Label(text = "Input 1:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 165)
Label(text = "Input 2:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 190)
Label(text = "Input 3:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 215)
Label(text = "Input 4:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 240)
Label(text = "Input 5:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 265)
Label(text = "Data bus", font =('Helvetica', 12, 'bold', 'underline')).place(x = 15, y = 290)
Label(text = "1-Wire:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 315)
Label(text = "CAN:", font =('Helvetica', 12, 'bold')).place(x = 35, y = 340)

batt_Str.set("none")
Label(textvariable = batt_Str, font =('Helvetica', 11, 'bold')).place(x = 205, y = 90)
batt_val_Str.set("none")
Label(textvariable = batt_val_Str, font =('Helvetica', 11, 'bold')).place(x = 205, y = 115)
input1_Str.set("none")
Label(textvariable = input1_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 165)
input2_Str.set("none")
Label(textvariable = input2_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 190)
input3_Str.set("none")
Label(textvariable = input3_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 215)
input4_Str.set("none")
Label(textvariable = input4_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 240)
input5_Str.set("none")
Label(textvariable = input5_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 265)
onewire_Str.set("none")
Label(textvariable = onewire_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 315)
can_Str.set("none")
Label(textvariable = can_Str, font =('Helvetica', 11, 'bold')).place(x = 105, y = 340)

# Init threads
mqttThread = threading.Thread(target = mqtt_handler)
mqttThread.daemon = True
mqttThread.start()
```

Obr. 31. Ukázka inicializace kódu

11.2 Připojení ke zprostředkovateli

Připojení ke zprostředkovateli se provádí pomocí vytvoření objektu s klientem v roli odběratele, kterému se předají parametry pro připojení k serveru. Na obrázku 32 je ukázka funkce zpracovávající odebíraná data ze zprostředkovatele.

```
def subscribe(client: mqtt_client):
    def on_message(client, userdata, msg):
        global batt_gui
        global batt_val_gui
        global input1_gui
        global input2_gui
        global input3_gui
        global input4_gui
        global input5_gui
        global onewire_gui
        global can_gui
        data = msg.payload
        print(f"Received `{msg.payload}` from `{msg.topic}` topic")
        try:
            index = data.index("PWR_DET=".encode())
            batt_gui = int(data[ index + 8 : data.index(";".encode(), index)])
            index = data.index("BATT=".encode())
            batt_val_gui = int(data[ index + 5 : data.index(";".encode(), index)])
            index = data.index("IN1=".encode())
            input1_gui = int(data[ index + 4 : data.index(";".encode(), index)])
            index = data.index("IN2=".encode())
            input2_gui = int(data[ index + 4 : data.index(";".encode(), index)])
            index = data.index("IN3=".encode())
            input3_gui = int(data[ index + 4 : data.index(";".encode(), index)])
            index = data.index("IN4=".encode())
            input4_gui = int(data[ index + 4 : data.index(";".encode(), index)])
            index = data.index("IN5=".encode())
            input5_gui = int(data[ index + 4 : data.index(";".encode(), index)])
            index = data.index("1WIRE=".encode())
            onewire_gui = float(data[ index + 6 : data.index(";".encode(), index)])
            index = data.index("CAN=".encode())
            can_gui = data[ index + 4 : data.index(";".encode(), index)]
        except Exception as e:
            print(str(e) + "\n")
    client.subscribe(topic)
    client.on_message = on_message
```

Obr. 32. Ukázka funkce pro připojení ke zprostředkovateli

11.3 Zpracování grafických prvků

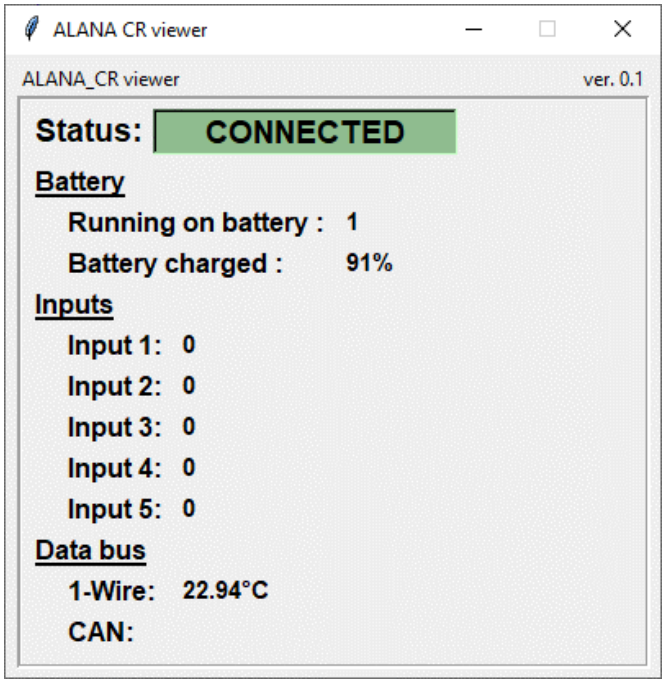
Zpracování grafických prvků se provádí ve vlákně s nekonečným cyklem, ve kterém se do jednotlivých prvků zapisují požadovaná data. Spojení se zprostředkovatelem je zobrazováno ve statusu, který v případě ztráty spojení zobrazí varovnou hlášku. K informacím o baterce a teplotě jsou přidány jednotky. Na obrázku číslo 33 je ukázka kódu pro zpracování grafických prvků.

```
while(1):

    if connected:
        varStatus.set("CONNECTED")
        textStatus.config(bg = 'dark sea green')

        batt_Str.set(batt_gui)
        batt_val_Str.set(str(batt_val_gui) + '%')
        input1_Str.set(input1_gui)
        input2_Str.set(input2_gui)
        input3_Str.set(input3_gui)
        input4_Str.set(input4_gui)
        input5_Str.set(input5_gui)
        onewire_Str.set(str(onewire_gui) + '°C')
        can_Str.set(can_gui)

    else:
        varStatus.set("COMM. ERROR")
        textStatus.config(bg = 'red')

        batt_Str.set("none")
        batt_val_Str.set("none")
        input1_Str.set("none")
        input2_Str.set("none")
        input3_Str.set("none")
        input4_Str.set("none")
        input5_Str.set("none")
        onewire_Str.set("none")
        can_Str.set("none")

    time.sleep(0.1)
```

Obr. 33. Zpracování grafických prvků

11.4 Výsledná aplikace

Výsledná aplikace vytvořená v programovacím jazyce Python spolu s knihovnou Tkinter je plně multiplatformní a závislá pouze na podpoře Pythonu v daném operačním systému. Pro účely interpretace dat přes protokol MQTT je plně škálovatelná a splňuje specifikace uvedené v návrhu. Na obrázku číslo 34 je ukázka spuštěné aplikace ALANA CR veiwer v operačním systému Microsoft Windows 10.



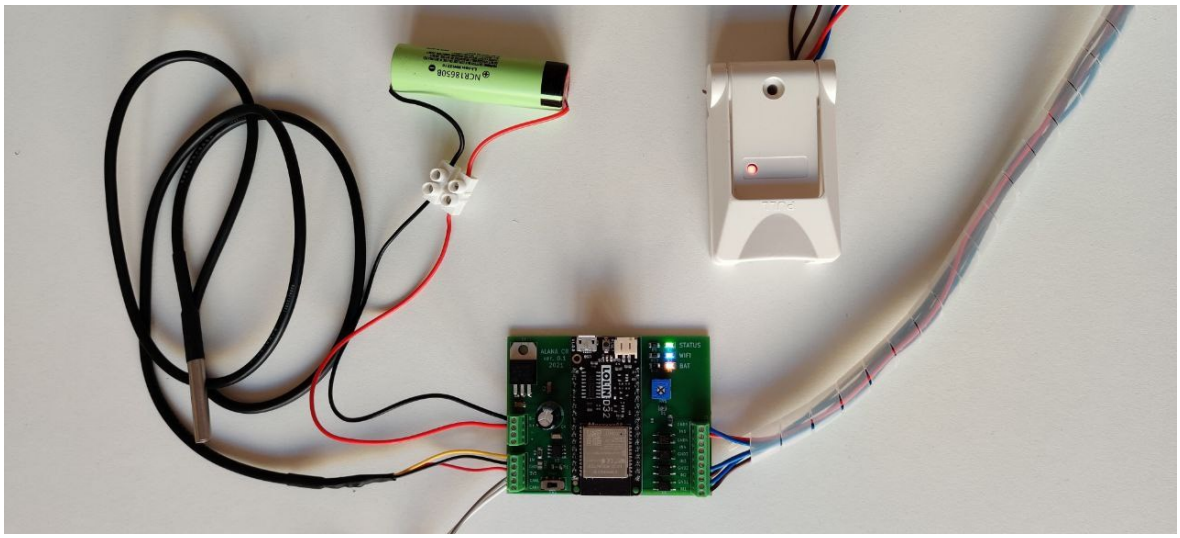
Obr. 34. Aplikace ALANA CR

12 OTESTOVÁNÍ SYSTÉMU

Otestování systému je provedeno otestováním všech jeho dílčích celků. Jedná se o získání informací z poplachových a nepoplachových komponent pomocí desky ALANA CR. Přenesení získaných informací přes protokol MQTT pomocí zprostředkovatele Eclipse Mosqutto. Uložení této informace do relační databáze pomocí knihovny SQLite a interpretace získané informace pomocí aplikace ALANA CR viewer.

12.1 Získání informací

Získání informací pro otestování systému je provedeno připojením poplachové a nepoplachové komponenty k desce ALANA CR. Pro poplachovou komponentu je použito tísňové tlačítko popsané v kapitole 5.1.3. Nepoplachová komponentu je reprezentována teplotním čidlem DS18B20 popsaným v kapitole 5.2.1. Deska ALANA CR bude napájena pomocí lithium – iontové baterie Panasonic 18650. Pro nasimulování odporově vyvážené smyčky je použito napětí ze zdroje, který slouží pro napájení poplachové komponenty kalibrované pomocí potenciometru na desce. Pro CAN komunikaci je použit převodník USB2CAN. Na obrázku číslo 35 jsou ukázány připojené zmíněné prvky k desce ALANA CR.



Obr. 35 Komponenty připojené k desce ALANA CR

12.2 Uchování informací

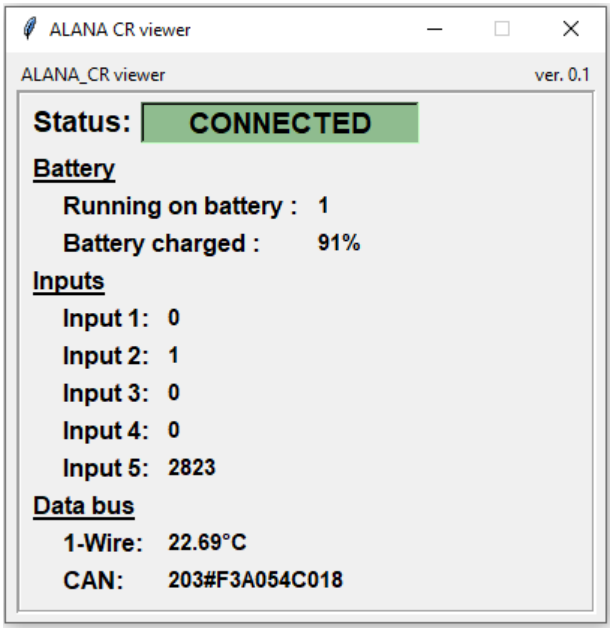
Uchování informací pomocí relační databáze je otestováno přečtením zapsaných dat získaných přes protokol MQTT a zpracovaných aplikací mqtt_to_db. Na obrázku číslo 35 jsou ukázaná přečtená data z databáze pomocí programu DB Browser for SQLite.

	id	time	power	batt	in1	in2	in3	in4	in5	onewire	can
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	alana_cr/data	27/04/2021 18:38:55	1	91	0	1	0	0	2821	22.37	203#F3A054C018
2	alana_cr/data	27/04/2021 18:38:57	1	91	0	1	0	0	2828	22.37	203#F3A054C018
3	alana_cr/data	27/04/2021 18:38:59	1	91	0	1	0	0	2832	22.37	203#F3A054C018
4	alana_cr/data	27/04/2021 18:39:01	1	91	0	1	0	0	2825	22.37	203#F3A054C018
5	alana_cr/data	27/04/2021 18:39:02	1	91	0	1	0	0	2821	22.37	203#F3A054C018
6	alana_cr/data	27/04/2021 18:39:04	1	91	0	1	0	0	2821	22.37	203#F3A054C018
7	alana_cr/data	27/04/2021 18:39:06	1	91	1	0	0	0	2823	22.37	203#F3A054C018
8	alana_cr/data	27/04/2021 18:39:08	1	91	1	0	0	0	2809	22.37	203#F3A054C018
9	alana_cr/data	27/04/2021 18:39:10	1	91	1	0	0	0	2829	22.37	203#F3A054C018

Obr. 36 Uchovaná data poplachových a nepoplachových komponent v databázi

12.3 Interpretování informací

Interpretování informací získaných přes protokol MQTT z poplachových a nepoplachových komponent je otestováno spuštěním aplikace ALANA CR viewer s korektním zobrazením stavu komponent připojených k desce ALANA CR. Na obrázku 37 je zobrazena spuštěná zmíněná aplikace. Všechny komponenty se zobrazují korektně. Deska ALANA CR je napájena pomocí lithium-iontové baterie a tato informace je zobrazena i s kapacitou v procentech. Jednička u vstupu 2 nám udává informaci, že tísňové tlačítko je otevřené. Hodnota na vstupu 5 udává nasimulovanou odporově vyváženou smyčku a na CAN sběrnici se posílá zobrazený rámec.



Obr. 37 Interpretovaná data v aplikaci ALANA CR

ZÁVĚR

V bakalářské práci byly zpracovány způsoby uchování a interpretace dat. Výstup z poplachových a nepoplachových komponent. Protokol MQTT včetně ukázky komunikace. Poté byl vytvořen návrh pro získání, uchování a interpretování informací z poplachových a nepoplachových komponent přes protokol MQTT. Jednotlivé návrhy byly realizovány a následně otestovány jako celek. Při otestování bylo ověřeno, že navrhovaný a realizovaný systém plní zadané specifikace.

SEZNAM POUŽITÉ LITERATURY

[1] MALÝ, Martin. Data, čipy, procesory: vlastní integrované obvody na koleni [online]. CZ.NIC, z.s.p.o. Praha: CZ.NIC, z.s.p.o., 2020 [cit. 2021-04-14]. CZ.NIC. ISBN 978-80-88168-56-0. Dostupné z: https://knihy.nic.cz/files/edice/Data_cipy_procesory.pdf

[2] BALASA, Florin, ed. Data Storage. Croatia, 2010 by INTECH d.o.o: IntechOpen, 2010. ISBN 978-953-307-063-6.

[3] MALÝ, Martin. Hradla, volty, jednočipy: úvod do bastlení [online]. CZ.NIC, z.s.p.o. Praha: CZ.NIC, z.s.p.o., 2017 [cit. 2021-04-14]. CZ.NIC. ISBN ISBN978-80-88168-23-2. Dostupné z: https://knihy.nic.cz/files/edice/hradla_volty_jednocipy.pdf

[4] DALE, Nell a John LEWIS. Computer Science Illuminated. 7th Edition. Jones & Bartlett Learning, 2019. ISBN 978-1284155617.

[5] MINASI, Mark. Pevné disky od A do Z. Praha: Grada, 1992. ISBN 80-856-2335-8.

[6] CROWLEY, Paul a Dave KLEIMAN. CD and DVD Forensics [online]. Syngress: Elsevier, 2006 [cit. 2021-04-14]. ISBN 978-1-59749-128-0. Dostupné z: https://doi.org/10.1016/B978-1-59749-128-0.X5000-5

[7] Co je cloudové úložiště? Microsoft Azure [online]. Microsoft, 2021 [cit. 2021-04-14]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-cloud-storage/

[8] JOHNSON, James L. Database: Models, Languages, Design. New York: Oxford University Press, 1997. ISBN 978-0195107838.

[9] Co je to databáze? Oracle [online]. Oracle, 2021 [cit. 2021-04-14]. Dostupné z: https://www.oracle.com/cz/database/what-is-database/

[10] POKORNÝ, Jaroslav a Ivan HALAŠKA. Databázové systémy. Vyd. 2. přeprac. Praha: Vydavatelství ČVUT, 2003. ISBN 80-010-2789-9.

[11] KHOL, Josef. Interpretace: nástin teorie a praxe interpretování. Praha: Academia, 1989. ISBN 80-200-0169-7.

[12] LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Brno: Computer Press, 2011. ISBN 978-802-5135-396.

[13] PILGRIM, Mark. Ponořme se do HTML5 [online]. CZ.NIC. Praha: CZ.NIC, z.s.p.o., [2015] [cit. 2021-04-14]. CZ.NIC. ISBN 978-80-905802-6-8. Dostupné z: https://knihy.nic.cz/files/edice/html5.pdf

[14] About Node.js. Node JS [online]. OpenJS Foundation, 2021 [cit. 2021-04-14]. Dostupné z: https://nodejs.org/en/about/

[15] LAVRINČÍK, Jan. OPERAČNÍ SYSTÉMY [online]. Olomouc, 2018 [cit. 2021-04-14]. Dostupné z: https://www.mvso.cz/files/operacni-systemy.pdf. Moravská vysoká škola Olomouc, o. p. s.

[16] MQTT: The Standard for IoT Messaging. MQTT [online]. MQTT.org, 2021 [cit. 2021-04-14]. Dostupné z: https://mqtt.org/

[17] HILLAR, Gaston C. MQTT Essentials - A Lightweight IoT Protocol. Birmingham: Packt Publishing, 2017. ISBN 978-1787287815.

[18] MALÝ, Martin. Protokol MQTT: komunikační standard pro IoT. *Root* [online]. Internet Info, 2016 [cit. 2021-04-14]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/

[19] DRGA, Rudolf. Elektronické bezpečnostní systémy: Poplachové zabezpečovací a tísňové systémy [online]. Zlín: Univerzita Tomáše Bati ve Zlíně, 2013 [cit. 2021-04-15].

[20] HB304. Carrier [online]. Carrier, 2020 [cit. 2020-07-09]. Dostupné z: https://firesecurityproducts.com/en/product/intrusion/HB304/42756

[21] MAXIM. DS18B20 Programmable Resolution 1-Wire Digital Thermometer [online]. San José, 951 34, USA: Maxim Integrated, 2021 [cit. 2021-04-14]. Dostupné z: https://www.gme.cz/data/attachments/dsh.530-067.1.pdf

[22] KUGELSTADT, Thomas. The RS-485 Design Guide [online]. Dallas, Texas 75265: Texas Instruments, 2016 [cit. 2021-04-14]. Dostupné z: https://www.ti.com/lit/an/slla272c/slla272c.pdf?ts=1615621759270

[23] CORRIGAN, Steve. *Introduction to the Controller Area Network (CAN)* [online]. Dallas, Texas 75265: Texas Instruments, 2016 [cit. 2021-04-14]. Dostupné z: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1615622780419

[24] MANIYAR, Sashavalli. *1 – Wire Communication with PIC Microcontroller* [online]. Chandler, Arizona, USA: Microchip Technology, 2008 [cit. 2021-04-14]. Dostupné z: http://ww1.microchip.com/downloads/en/appnotes/01199a.pdf

[25] IBRAHIM, Dogan a Ahmet IBRAHIM. The Official ESP32 Book. Oude Rijksweg Noord 64A, 6114 JG Susteren, Netherlands: Elektor International Media, 2017. ISBN 978-1907920639.

[26] ESPRESSIF SYSTEMS. ESP32 [online]. Shangai: ESPRESSIF SYSTEMS, 2021 [cit. 2021-04-14]. Dostupné z: https://www.espressif.com/en/products/socs/esp32

[27] ESPRESSIF SYSTEMS. ESP32 Series: Datasheet [online]. V3.6. Shanghai: Espressif Systems, 2021 [cit. 2021-04-14]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[28] D32. *Wemos* [online]. wemos.cc, 2019 [cit. 2021-04-14]. Dostupné z: https://www.wemos.cc/en/latest/d32/d32.html

[29] What is Linux? Red Hat [online]. Raleigh, North Carolina, United States: Red Hat, 2021 [cit. 2021-5-8]. Dostupné z: https://www.redhat.com/en/topics/linux/what-is-linux#:~:text=Linux%C2%AE%20is%20an%20open,resources%20that%20do%20the%20work.

[30] Eclipse Mosquitto: An open source MQTT broker. https://mosquitto.org/ [online]. Eclipse Foundation, 2021 [cit. 2021-04-15].

[31] About SQLite. SQLite [online]. SQLite Consortium, 2021 [cit. 2021-4-27]. Dostupné z: https://www.sqlite.org/about.html

[32] About Python? Python [online]. Wilmington, Delaware, United States: Python Software Foundation, 2021 [cit. 2021-5-8]. Dostupné z: https://www.python.org/about/

[33] Python interface to Tcl/Tk. https://docs.python.org/3/library/tkinter.html [online]. Python, 2021 [cit. 2021-04-15].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AOSIS – Organization for the Advancement of Structured Information Standards

CAN – Controller Area Network

CCTV – Closed-circuit television

CR – Carriage return

CSS – Kaskádové styly

GPIO – General-purpose input/output

HTML – Hypertextový značkovací jazyk

IoT - Internet of Things

IP – Internet Protocol

LF – Line feed

MQTT – Protokol Message Queuing Telemetry Transport

NC – Normally Open

NO – Normally Closed

ODL – Object Definition Language

OQL – Object Query Language

OS – Operační systém

PLC – Programovatelný logický automat

PZTS – Poplachový zabezpečovací a tísňový systém

SMD – Surface Mount Device

SQL – Structured Query Language

TCP/IP – Transmission Control Protocol/Internet Protocol

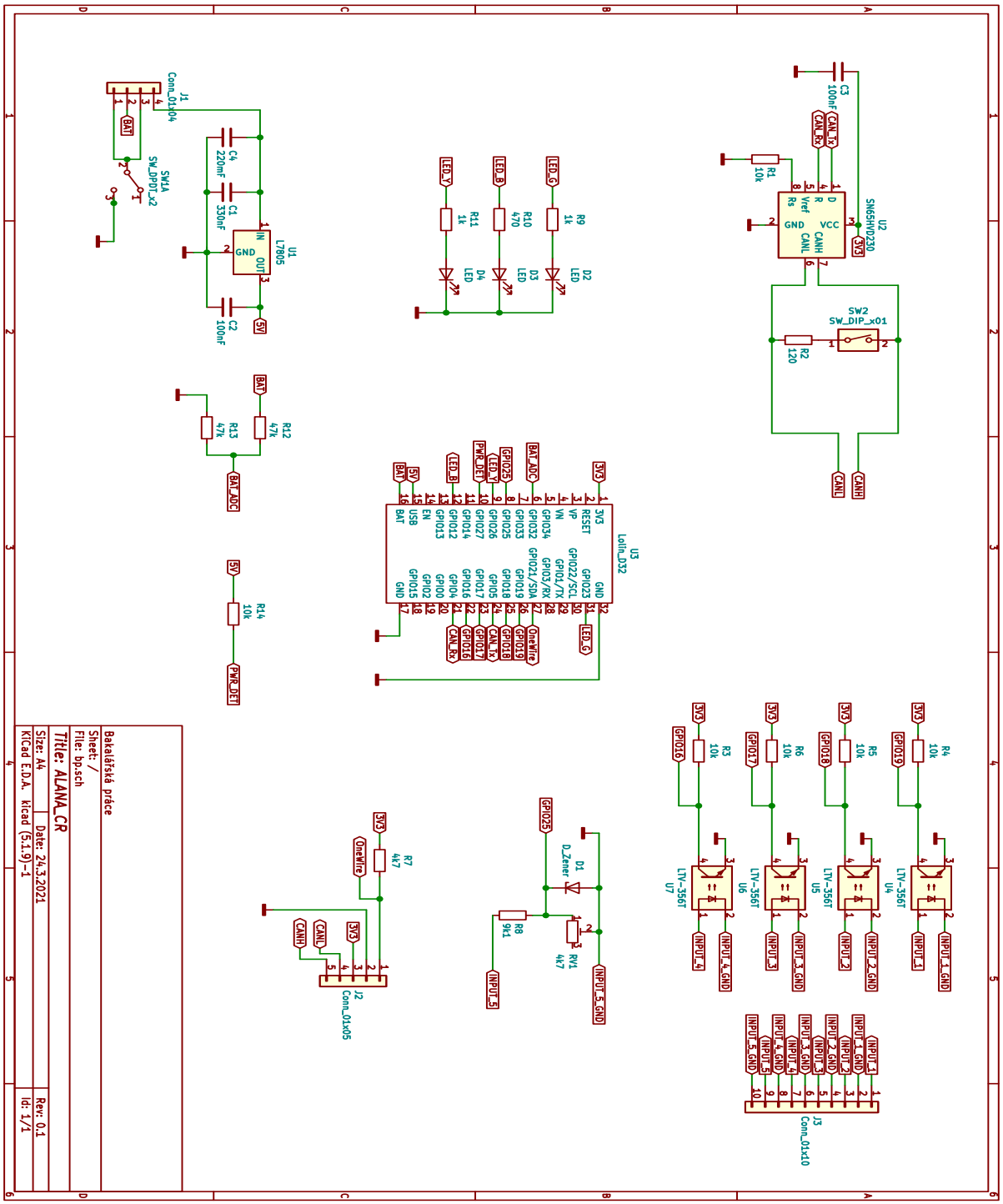
THT – Through-hole technology

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: SCHÉMA DESKY ALANA CR



PŘÍLOHA P II: DESKA ALANA CR VE FORMÁTU GERBER