VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

PŘÍSTUPOVÝ SYSTÉM PODPORUJÍCÍ VÍCE ZPŮSOBŮ IDENTIFIKACE

ACCESS SYSTEM CONTROLLER SUPPORTING MORE TYPES OF IDENTIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ HAVLÍČEK**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. ZDENĚK VAŠÍČEK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Havlíček Lukáš**
Program: Informační technologie
Název: **Přístupový systém podporující více způsobů identifikace**
Access System Controller Supporting More Types of Identification
Kategorie: Vestavěné systémy

Zadání:

1. Seznamte se s principem realizace přístupových systémů, technologií RFID, standardem MIFARE a Bluetooth Low Energy (BLE).
2. Navrhněte kompaktní zařízení, které bude umožňovat řízení přístupu do určité lokality (objekt, místnost) s tím, že bude možné použít více způsobů identifikace dle preference uživatele - minimálně přístupové karty MIFARE a technologie BLE (např. přítomnost mobilního zařízení, chytrých hodinek, apod. s dostatečnou sílou signálu; nebo přístup realizovaný skrze BLE protokol). Navrhněte vhodné konfigurační rozhraní.
3. Navržené zařízení implementujte formou prototypu a ověřte jeho funkčnost.
4. Diskutujte vlastnosti navrženého řešení (např. bezpečnost) a možnosti dalšího rozšíření.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Vašíček Zdeněk, doc. Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Tato práce se zabývá návrhem a implementací řídicí jednotky přístupového systému s podporou více způsobů identifikace. Navržený systém podporuje jako způsoby identifikace přístupové karty MIFARE a detekci zařízení s Bluetooth Low Energy. Zařízení je realizováno pomocí vývojové desky ESP32. Zobrazení historie průchodů a konfigurace systému je řešeno pomocí webového rozhraní, které je poskytováno pomocí webserveru běžícím na tomto zařízení.

Abstract

This thesis deals with design and implementation of access system control unit supporting more types of identification. Designed system supports identification using access cards MIFARE and detection of Bluetooth Low Energy devices. Device is realised using the ESP32 development board. The display of passage history and system configuration is solved using a web interface, which is provided by a web server running on this device.

Klíčová slova

přístupové systémy, vestavěné systémy, ESP32, Arduino, Mifare, BLE, RTC

Keywords

access systems, embedded systems, ESP32, Arduino, Mifare, BLE, RTC

Citace

HAVLÍČEK, Lukáš. *Přístupový systém podporující více způsobů identifikace*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Zdeněk Vašíček, Ph.D.

Přístupový systém podporující více způsobů identifikace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Zdeňka Vašíčka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Lukáš Havlíček
28. května 2020

Poděkování

Tímto bych chtěl poděkovat vedoucímu práce panu Doc. Ing. Zdeňku Vašíčkovi, Ph.D. za odborné vedení, cenné připomínky a konzultace, které mi poskytl při vypracování této práce.

Obsah

1 Úvod **3**

2 Možnosti identifikace osob a související technologie **4**
2.1 Technologie RFID 4
2.2 Technologie NFC 4
2.3 Bezkontaktní karty Mifare 5
2.3.1 Vnitřní struktura paměti karet MIFARE 6
2.4 Standard Bluetooth Low Energy 6
2.4.1 Bluetooth vs Bluetooth Low Energy 7
2.4.2 Princip Bluetoth Low Energy 7
2.5 Další možnosti identifikace 9
2.5.1 Biometrická identifikace 9
2.5.2 Klávesnice 9
2.5.3 Vzdálená identifikace 9

3 Realizace vestavěných zařízení **10**
3.1 Mikrokontroléry 10
3.2 Hodiny reálného času 11
3.3 Paměť zařízení 12
3.4 Sběrnice I2C 12
3.5 Sběrnice SPI 15
3.6 Konfigurace vestavěných zařízení 16
3.6.1 Konfigurace pomocí klávesnice 16
3.6.2 Konfigurace pomocí aplikace 16
3.6.3 Konfigurace pomocí webového rozhraní 16
3.6.4 Realizace uživatelského rozhraní využívajících webových technologií 17
3.6.5 Webové technologie 17

4 Návrh **19**
4.1 Návrh Hardware 19
4.1.1 Čtečka karet a karty Mifare 19
4.1.2 Napájení 19
4.1.3 Řídicí jednotka 19
4.1.4 Hodiny reálného času 20
4.1.5 Displej 21
4.1.6 Zvuková signalizace 21
4.2 Návrh Firmware 21
4.2.1 Přístup pomocí MIFARE 21

4.2.2 Přístup pomocí BLE . . . 22
4.2.3 Řídicí jednotka . . . 22
4.2.4 Konfigurační rozhraní . . . 22
4.2.5 Displej . . . 26
4.2.6 Zvuková signalizace . . . 27
4.2.7 Historie průchodů . . . 27

5 Implementace **29**
5.1 Programovací jazyk . . . 29
5.2 Struktura projektu . . . 31
5.3 Webserver . . . 31
5.3.1 Knihovna ESPAsyncWebServer . . . 31
5.3.2 Knihovna WebServer . . . 32
5.3.3 Webové rozhraní . . . 32
5.4 Flash paměť . . . 33
5.5 Karty Mifare . . . 34
5.6 Bluetooth Low Energy . . . 34
5.7 Displej . . . 35
5.8 Bzučák . . . 36
5.9 Historie průchodů . . . 36
5.10 Ověření funkčnosti . . . 36
5.11 Bezpečnost . . . 38

6 Závěr **39**

Literatura **40**

A Zapojení zařízení **42**

Kapitola 1

Úvod

Přístupový systém je vestavěné zařízení, jehož primárním úkolem je zúžení přístupu do určité lokality pouze na vybrané osoby. Mimo to může takový systém také často plnit i další role jako je např. evidence docházky, neboť již z principu shromažďují záznamy o průchodu. Přístupový systém se skládá ze dvou hlavních částí, kterými je mechanismus zabezpečující blokování přístupu a řídicí jednotka. Jako systém pro blokování přístupu může být využito například elektronického zámku. Jako řídicí jednotka se dá následně považovat elektronika kontrolující, zda daná osoba má oprávnění pro přístup a následné povolení přístupu (například odemčení zámku). Řídicí jednotka může využívat pro kontrolu oprávnění různé způsoby. Jedním z nejrozšířenějších řešení je implementace využívající čtečky přístupových karet, kde přístupová karta slouží jako identifikační médium. Takový systém je typický pro komerční provozy s mnoha uživateli. Pro domácí použití se typicky používá identifikace pomocí přístupového hesla zadaného pomocí klávesnice. V případě požadavku na vyšší stupeň bezpečnosti lze použít různé biometrické senzory, např. senzor otisku prstu apod.

V dnešní době se rozšířila nositelná elektronika a řada uživatelů vlastní buď mobilní telefon nebo chytrý náramek či hodinky. Všechny technologie spojuje jeden prvek a sice, že mají schopnost bezdrátové komunikace pomocí standardu Bluetooth, typicky Bluetooth Low Energy (BLE). Cílem této práce je vytvoření řídicí jednotky, která podporuje více způsobů identifikace a využívá trendu poslední doby. Kromě přístupové karty je snahou využít k autorizaci za pomoci detekce přítomnosti bezdrátových BLE zařízení. A to za účelem zlepšení způsobu identifikace pro osoby využívající přístupové systémy. Pomocí přítomnosti známého zařízení (obsahující BLE) v dostatečné blízkosti, lze předpokládat, že osoba chce projít dveřmi a je jí automaticky povolen přístup bez nutnosti přiložení přístupové karty ke čtečce.

Tato práce je členěna následovně. Kapitola 2 popisuje možné způsoby identifikace u přístupových systémů a technologie, které s nimi souvisí. V následující kapitole 3 jsou probrány prvky související s realizací vestavěného systému, jako je výběr vhodného mikrokontroléru, popis vybraných sběrnic pro připojení externích periferií a realizace konfiguračního rozhraní.

Návrh řídicí jednotky je popsán v kapitole 4, kde se postupně zabýváme hardwarovou a softwarovou částí.

Poslední kapitola 5 poté obsahuje implementační detaily a zhodnocení výsledného zařízení. Je zde popsán výběr programovacího jazyka a vhodného prostředí. Dále je zde popsána implementace, u které jsou krátké ukázky kódu. V poslední části této kapitoly je provedeno ověření funkčnosti a diskutována bezpečnost zařízení.

Kapitola 2

Možnosti identifikace osob a související technologie

V této kapitole jsou popsány různé způsoby identifikace, které lze použít pro identifikaci osob či objektů. Jako první jsou zde stručně popsány související technologie RFID a NFC, které jsou využívány k identifikaci či bezdrátové komunikaci. Další částí této kapitoly je poté popis standardu Bluetooth Low Energy, který je využívám převážně pro komunikaci u nositelné elektroniky a přístupové karty Mifare, které jsou běžně využívány pro identifikaci uživatelů. V poslední části této kapitoly jsou uvedeny některé další možnosti identifikace osob, jako může být využití klávesnice nebo biometrické identifikace.

2.1 Technologie RFID

RFID (Radio-frequency identification) využívá, jak již plyne z názvu, pro komunikaci rádiové vlny. Pro komunikaci využívá převážně frekvenci 125 kHz, 134 kHz (nízkofrekvenční) a 13,56 MHz (Vysokofrekvenční), kam také spadá standard MIFARE. Ale může využívat i jiné, a to Ultra frekvenční (frekvence okolo 900 Mhz a 2,4 Ghz) anebo frekvence ve volném pásmu ISM. Princip fungování RFID se dělí na tzv. tagy a čtečky.

Tagy mohou být aktivní nebo pasivní. Využívají se pro identifikaci osob, zboží, zvířat atd. V pasivním režimu spoléhá tag na přenos energie pomocí indukce ze čtečky, díky tomu, že pasivní tag nepotřebuje baterii, tak je podstatně levnější, než aktivní tag. Na druhou stranu, protože potřebuje získat energii od čtečky, má velice omezený dosah, řádově několik cm. Aktivní tag využívá baterie a dává o sobě vědět, že existuje, aniž by byl přiložen ke čtečce. Vzhledem k tomu, že potřebuje baterii tak je dražší, ale díky tomu má delší dosah, který může být až několik set metrů. Jako třetí typ tagu existuje kombinace předešlých, čímž je pasivní tag s baterií, který vysílá pouze v případě přiložení ke čtečce.

Čtečky mohou být rozděleny také na pasivní a aktivní. Kde pasivní čtečka (pouze v kombinaci s aktivním tagem) čeká na signál od tagu pro komunikaci. A aktivní čtečka vysílá "dotazovací signál" tagu pro komunikaci. [21]

2.2 Technologie NFC

NFC je zkratka pro Near-field Communication, jedná se o standard pro bezdrátovou komunikaci na krátkou vzdálenost, většinou v dosahu do 4 cm. NFC funguje na frekvenci 13.56 MHz s datovým přenosem do 424 kb/s. Stejně jako RFID, tak NFC se dělí na aktivní

a pasivní zařízení. Kde aktivní zařízením může například být chytrý telefon a pasivním zařízením tag.

Aktivní NFC zařízení může být v jednom z následujících módů:

- Card emulation - Kde aktivní zařízení emuluje kartu například pro platbu.
- Read/Write - Zařízení komunikuje s pasivním tagem.
- Peer-to-peer - Komunikace dvou aktivních zařízení za účelem výměny informací.

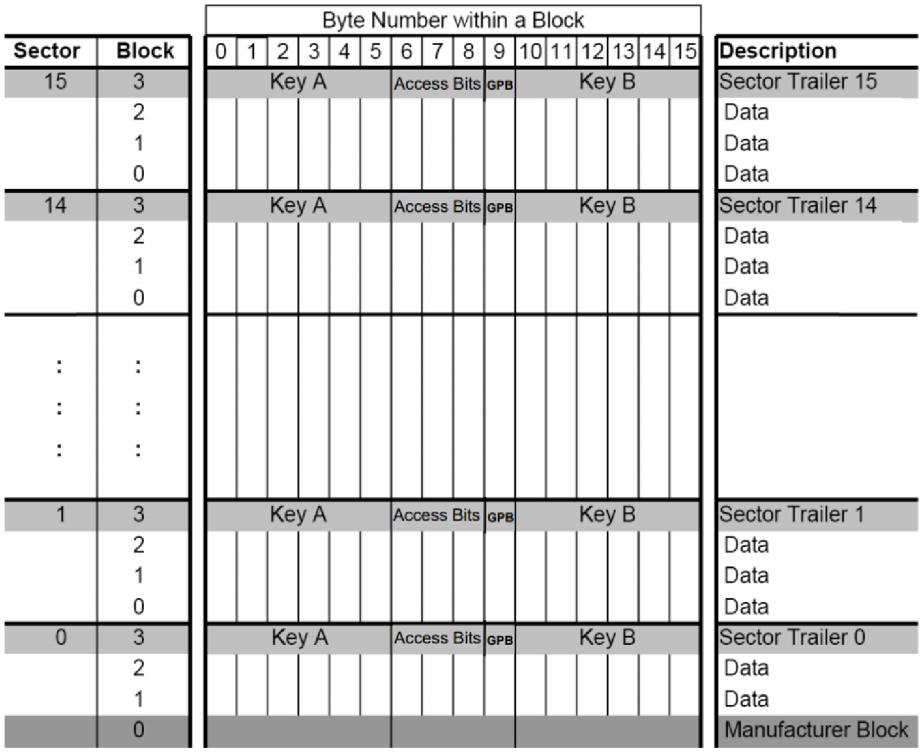
NFC standard zahrnuje komunikační protokoly a formáty dat, je založen na standardu RFID a je definován v ISO/IEC 18092. [15]

2.3 Bezkontaktní karty Mifare

Bezkontaktní karty Mifare Classic byly vyvinuty firmou NXP Semiconductors a na trh byly uvedeny již v roce 1994. Karty se poměrně rychle staly populární, převážně kvůli jejich nízké ceně. Karet bylo celkem vyrobeno několik miliard kusů a předpokládá se, že je stále několik set milionů kusů v oběhu. Bezpečnost karet je založena na algoritmu CRYPTO1, kde bezpečnost byla založena také na tom, že algoritmus byl implementován pomocí HW a byl neveřejný. Jeho skrytost ale nevydržela dlouho, v roce 2007 dva němečtí vědci předvedli část tohoto algoritmu pomocí metody zpětného inženýrství. V roce 2008 se výzkumné skupině na univerzitě Radboud podařilo metodou zpětného inženýrství odhalit celý algoritmus CRYPTO1 a chtěli tento úspěch publikovat. Proti této publikaci byla společnost NXP a rozhodla se zahájit soudní proces. Tento proces nakonec dopadl ve prospěch zveřejnění tohoto výzkumu a tím přestal v roce 2008 být tento algoritmus tajný. Toto vedlo postupně k různým metodám a nástrojům, jak tyto karty prolomit, rozšifrovat data uložená na nich a případně je celé okopírovat, což podstatně poškodilo reputaci těchto karet. Existují i novější typy MIFARE, které zatím nebyly prolomeny a jsou nekopírovatelné, jako je například Mifare Desfire EV1, které byly uvedeny na trh v roce 2006 a Mifare Desfire EV2 v roce 2013, ale tyto karty jsou dražší než původní Mifare karty a z toho důvodu se původní karty prodávají a používají i nadále, ale nejsou doporučeny do míst, kde je vyžadována vysoká bezpečnost.

U přístupových systémů by zvolení typu karty mělo záviset na tom, kde systém bude použit a jaká je potřeba bezpečnost, ale stále je potřeba pro kopii karty, získat kartu originální, takže pokud se neoprávněná osoba dostane k originální kartě, tak ji stejně může zneužít. Liší se tedy pouze v tom, zda je možné kartu okopírovat (ale je potřeba speciálního vybavení) nebo kartu okopírovat nelze.

Existuje několik způsobů pro zvýšení bezpečnosti karet Mifare Classic. Jedním z nich například je, že systém má uložené povolené UID (jedinečné identifikátory) karet a neakceptuje jiné karty než ty na seznamu. Toto zabraňuje, aby systém nepřijal podvrženou kartu, která má správný obsah paměti, ale nepatří do systému. Další možností je kromě UID karty ukládat taky specifické informace zároveň v systému a v paměti karty a ověřovat je při použití karty. Ale všechno toto jsou pouze ztížení padělku karty, které nefungují v případě, kdy neoprávněná osoba okopíruje kartu kompletně se všemi informacemi, včetně UID. Přestože je UID určeno pouze pro čtení už výrobou karty a nedá se toto změnit, tak existují neoficiální karty, které mají měnitelné UID a tím pádem lze vytvořit perfektní kopii karty a proti tomu se nedá nijak bránit. Jedinou překážkou je poté, že neoprávněná osoba musí mít speciální vybavení (pro čtení a zápis karty a speciální kartu s měnitelném



Obrázek 2.1: Uspořádání paměti mifare classic 1K (převzato z [14])

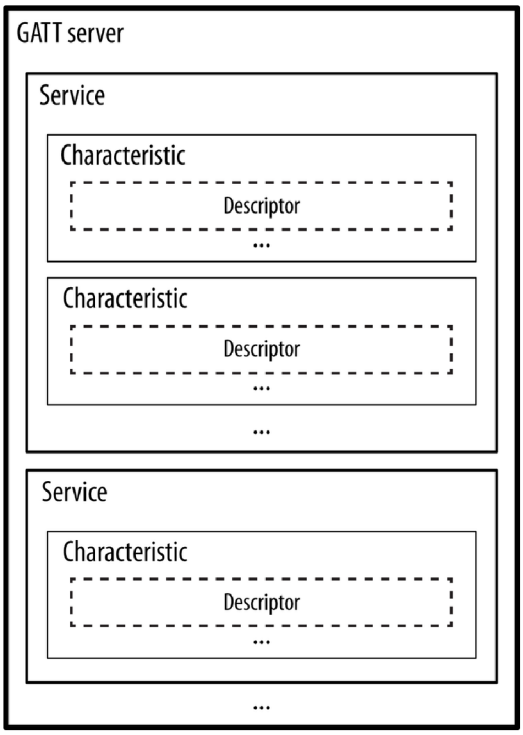
UID), což v dnešní době není až takový problém sehnat. Poslední překážkou je poté, že neoprávněná osoba musí mít fyzicky originální kartu, aby ji mohla okopírovat. Tím pádem se dá do jisté míry na kartu nahlížet jako na obyčejný klíč ke dveřím, pokud se k němu dostane neoprávněná osoba, dokáže s ním otevřít dveře. [1]

2.3.1 Vnitřní struktura paměti karet MIFARE

Jak lze vidět na obrázku 2.1, tak paměť karty Mifare Classic 1K je rozdělena do 16 sektorů. Každý tento sektor je dále rozdělen do 4 bloků, kde každý blok má velikost 16 B, pro celkovou velikost sektoru 64 B. Poslední blok daného sektoru vždy obsahuje klíče A a B pro přístup k bloku (klíč B je nepovinný) a oprávnění k danému bloku. Tím pádem každý sektor obsahuje pouze 48 B (3 bloky) pro užitečná data. První blok prvního sektoru je speciální blok, obsahuje data z výroby jejichž součástí je například i UID (jedinečný identifikátor), tento blok je také chráněn proti zápisu a dá se z něj pouze číst. [14]

2.4 Standard Bluetooth Low Energy

Bluetooth je standard pro bezdrátovou komunikaci pracující v pásmu 2,4 GHz. S rozvojem mobilních zařízení a nositelné elektroniky se ukázalo, že standard Bluetooth je energeticky nevýhodný a je potřeba energeticky výhodnější varianty. Pro tyto účely vznikl nový standard Bluetooth Low Energy, který je určen pro zařízení, kde je spotřeba energie kritická, jako jsou bateriově napájená zařízení.



Obrázek 2.2: Hirearchie BLE (převzato z [20])

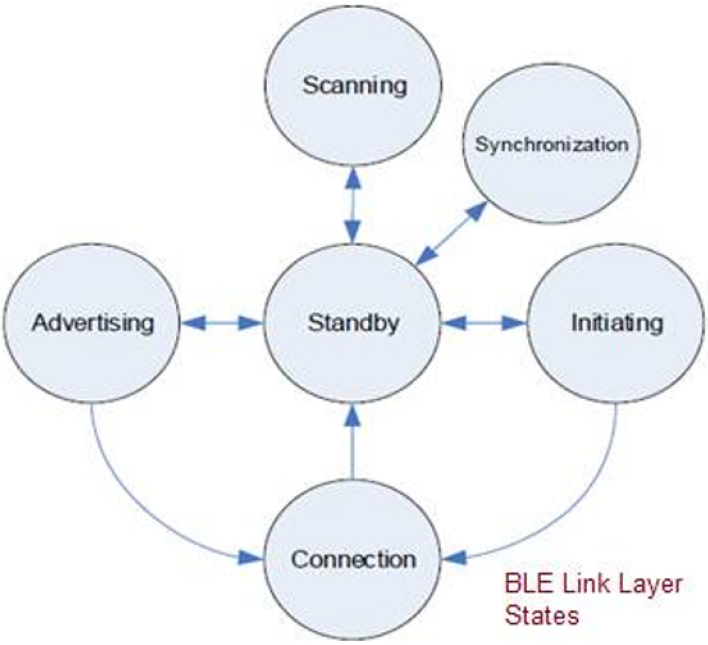
2.4.1 Bluetooth vs Bluetooth Low Energy

Tyto dva standardy mají společným rysem stejné operační pásmo 2.4 GHz, ale ve většině ostatních vlastností se liší. Klasické Bluetooth je využíváno převážně pro přenos větších objemů dat a zařízení je aktivní po celou dobu. Spojení mezi dvěma zařízeními trvá řádově několik set milisekund a je využíváno tedy pro přenos audia, videa, souborů apod. Využívá se například pro bezdrátová sluchátka na poslech hudby. Bluetooth Low Energy se na rozdíl od klasické varianty využívá pro krátký přenos malých dat a je většinu času v režimu spánku, čímž šetří energii. Jeho spojení s dalším zařízením je podstatně rychlejší, než u klasického Bluetooth a trvá pouze řádově několik milisekund. Jako příklad zařízení mohou být například fitness náramky, bezdrátová klávesnice nebo například části chytré domácnosti.

Zásadním rozdílem je také, že jsou zpětně nekompatibilní. Tím pádem Bluetooth zařízení nemůže komunikovat s Bluetooth Low Energy zařízením, a to stejné platí i naopak. Zařízení, která podporují obě tyto verze Bluetooth (například mobilní telefon), řeší tuto nekompatibilitu přepínáním mezi verzemi.

2.4.2 Princip Bluetoth Low Energy

Bluetooth Low energy funguje na principu server a klient, což je znázorněno na obrázku 2.3, kde lze vidět, že zařízení se může dostat do stavu spojení pouze ze stavu "advertising" a "initiating", kde jedno ze zařízení se nabízí ostatním zařízením (server) a druhé zařízení, které na toto odpoví požadavkem o spojení (klient). Následně si zařízení vyměňují informace pomocí „služeb“ a „charakteristik“ (Service a Characteristic), kde server poskytuje 1 nebo více služeb. Každá služba může obsahovat stejně tak 1 a více charakteristik, jak lze vidět



Obrázek 2.3: Stavy BLE (převzato z [17])

na obrázku 2.2. Dále navíc každá charakteristika může obsahovat „popisy“ (Descriptors), které obsahují informace o dané charakteristice. Všechny tyto části obsahují UUID (univerzální jedinečný identifikátor). Existuje spousta již předdefinovaných UUID těchto služeb a charakteristik definovaných standardem Bluetooth [1]. Tyto oficiální UUID jsou 16bitové. V případě, že chcete vytvořit vlastní specifickou službu či charakteristiku, protože vám nevyhovuje jedna z předdefinovaných, tak je možno si pro toto vygenerovat vlastní UUID, které je na rozdíl od předdefinovaného 128bitové.

Charakteristiky mají různé vlastnosti, kde mezi nejdůležitější, které patří za zmínku jsou: Read, Write a Notify. Read je určeno pro čtení. Příkladem může být čtení času ze zařízení. Write je pro zápis. Příkladem může být nastavení času na zařízení. A Notify, která slouží pro notifikaci klientů, může například sloužit k oznámení klesnutí baterie pod určitou hranici.

Spojení dvou BLE zařízení se skládá z následujících 3 fází [6]:

1. Výměna informací o sobě, „domluva“ na typu zabezpečení, možnost požadavku o bonding.
2. Výměna klíčů (navázání bezpečného spojení).
3. Bonding (volitelné) - Uložení si klíčů, které jsou využity pro bezpečné spojení, pro příští spojení. Při opětovném navázání spojení těchto dvou zařízení nedochází k výměně klíčů, ale jsou využity tyto uložené.

Možné způsoby zajištění bezpečného spojení [6]:

- Just Works – Základní způsob. Klíče jsou vyměněny bez zásahu uživatele. Liší se u zařízení ve verzi 4.2, kde se využívá Diffieho–Hellmanova algoritmu s využitím elip-

[1] Některé předdefinované UUID lze najít například na: https://www.bluetooth.com/specifications/gatt/services/

tických křivek, který zajišťuje vyšší bezpečnost než u starších verzí, kde se využívalo normálního Diffieho–Hellmanova algoritmu.

- Out of Band – Spojení je navázáno jinou technologií, než BLE, například pomocí NFC.
- Passkey – Jedno ze zařízení vygeneruje šestimístné číslo, které je zadáno na druhé zařízení, zde je potřeba aby alespoň jedno zařízení mělo obrazovku a druhé klávesnici (tyto informace si zařízení vymění v první fázi).
- Numeric comparison – Pouze u verze 4.2. Je potřeba aby obě zařízení obsahovaly obrazovku, spojení je provedeno stejně jako u Just Works metody, ale nakonec obě zařízení nezávisle na sobě, vygenerují potvrzovací hodnotu, která se zobrazí na displeji a uživatel sám zkontroluje, jestli se čísla shodují a bylo opravdu navázáno spojení mezi těmito dvěma zařízeními.

2.5 Další možnosti identifikace

V této části jsou popsány vybrané další možnosti identifikace, které jsou obecně využívány u přístupových systémů. Jejich použití se liší převážně na základě umístění zařízení a požadované bezpečnosti.

2.5.1 Biometrická identifikace

Jednou z možností identifikace je pomocí biometrické charakteristiky, což jsou jedinečné vlastnosti člověka. Pro identifikace je poté možno použít například otisk prstu, tvar obličeje, oční duhovku nebo hlas. Tento způsob identifikace je jeden z nejlepších možných způsobů, protože jednoznačně identifikuje uživatele a podvrhnutí většiny těchto biometrických charakteristik je velice obtížné a díky tomu je identifikace dostatečně bezpečná. V dnešní době obsahuje značná část mobilních zařízení senzor otisku prstu pro identifikaci a novější typy mobilních zařízení obsahují i identifikaci pomocí tvaru obličeje či scanu oční duhovky.

2.5.2 Klávesnice

Identifikaci pomocí klávesnice lze vyřešit i pomocí numerické klávesnice s jedním univerzálním heslem pro přístup. U tohoto způsobu je poté ale problém, že při každém odebrání uživatele by bylo potřeba heslo změnit a všichni stávající uživatelé by dostali nové heslo, které by používali. Tento způsob je také méně bezpečný z důvodu snadné replikace. Jediné co musí útočník znát, je heslo, které je zadáno uživatelem na klávesnici a lze odhadnout například pozorováním uživatele, jak toto heslo zadává.

2.5.3 Vzdálená identifikace

Systém by také mohlo jít ovládat vzdáleně, i když tento způsob nebývá příliš využívaný u přístupových systémů. Pro tento způsob může být využito například SMS zpráv, které jsou zaslány systému pomocí sítě GSM nebo pomocí požadavku z aplikace předaného pomocí sítě internet. Díky tomu lze vzdáleně povolit přístup osobě, co potřebuje učinit průchod, ale není v systému registrována. Tento způsob je značně omezující pro umístění zařízení, protože je potřeba aby zde byla dostupná síť GSM nebo internet.

Kapitola 3

Realizace vestavěných zařízení

Tato kapitola se zabývá částmi vestavěných systémů. Vestavěný systém se typicky skládá z řídicí jednotky (např. mikrokonrolér) a připojených periferií. V první části této kapitoly jsou popsány mikrokontroléry a jsou zde srovnány různé varianty. Následující část je věnována pro hodiny reálného času a paměti zařízení, které jsou u vestavěných systémů využívány. Další částí jsou vybrané sběrnice I2C a SPI, které jsou běžně využívané k propojení mikrokontroléru a externích periferií. Poslední část této kapitoly je věnována konfiguraci vestavěných systémů a stručnému přehledu technologií, které jsou ke konfiguraci využívány.

3.1 Mikrokontroléry

Mikrokontrolér neboli také jednočipový počítač, je integrovaný obvod, obsahující vše potřebné k vykovávání programu sám o sobě, jako například paměť FLASH pro program, RAM paměť pro proměnné programu apod. Díky tomu a také díky jejich nízké ceně, se využívají jako řídicí jednotky vestavěných systémů, proto je nutné pro takový systém vybrat vhodný mikrokontrolér. Na trhu je spousta druhů a každý se něčím odlišuje, ať už je to počet vstupně/výstupních pinů, velikost paměti, rychlost procesoru nebo cena je potřeba toto zohlednit při výběru.

Momentálně je pravděpodobně na trhu nejpopulárnější Ardunio, přesněji jeho verze Arduino Uno, které je na trhu od roku 2005. Je tomu hlavně díky jeho nízké ceně, lehkému použití, poměrně vysokými možnostmi použití a spoustě návodů na internetu pro začátečníky. Ardunio má kromě verze Uno více variant, které se liší jak cenou, tak dalšími parametry, a proto je vhodné nevybrat si vždy tu nejpoužívanější variantu, ale tu nejvíce vhodnou. Jako příklad dalších variant Arduina, může být Mega, Mini, Micro, Nano a další, které už ale nejsou tak známe, jako ty vyjmenované.

V poslední době také začíná být populární ESP, které je také levné a poskytuje mnohem více možností než Arduino. ESP je novější než Arduino, verze ESP8266 vyšla v roce 2014 a jeho nástupce ESP32 vyšel v roce 2016. Z důvodu že je ESP32 podstatně novější tak pro něj neexistuje zatím tolik návodů na internetu a například i knihovny nejsou úplně odladěné a občas se v nich vyskytne chyba. Z tohoto důvodu ESP32 není vhodné příliš pro začátečníky, a proto není ESP zatím tak populární a rozšířené jako Arduino, přestože poskytuje více možností.

Jak lze vidět z tabulky 3.1, tak přestože ESP stojí podobně jako Arduino Uno, tak má lepší parametry. Hlavní jeho výhodou je že obsahuje již v základní verzi WiFi a Bluetooth verze 4.2 (které obsahuje i Bluetooth Low Energy), u Arduina by se dalo přikoupit externí

Parametry	Arduino Uno [3]	Aruino Mega [2]	ESP32 [18]
Operační napětí	5 V	5 V	3.3 V
GPIO	20x	70x	34x
Flash paměť	32 KB	256 KB	4 MB (až 16 MB)
SRAM	2 KB	8 KB	520 KB
Frekvence CPU	16 MHz	16 MHz	160 až 240 MHz
Wifi	Ne	Ne	Ano
Bluetooth	Ne	Ne	Ano (verze 4.2)
Cena (oficiální)	540 Kč	940 Kč	490 Kč
Cena (aliexpress)	85 Kč	210 Kč	90 Kč

Tabulka 3.1: Tabulka srovnání mikrokontrolérů

modul nebo koupit jejich dražší variantu, která například obsahuje WiFi. V uvedené tabulce jsou vypsány pouze některé rozdíly mezi jednotlivými mikrokontroléry, liší se ve spoustě dalších věcí, které jsou ale více specifické, jako například počet SPI rozhraní, počet kanálu PWM atd.

3.2 Hodiny reálného času

Hodiny reálného času, nejčastěji označované jako RTC (Real-time clock), umožňují zařízení pracovat s reálným časem. Většinou zařízení reaguje na změny nezávisle na reálném čase, když přijde změna na vstupu, tak na ni reaguje, například otevřením dveří po přiložení správné karty na čtečku. V případě, že zařízení potřebuje reagovat vůči reálnému času, jako může být zpoždění o určitý počet vteřin nebo uložení aktuálního času v závislosti na externí události, musí zařízení využít hodin reálného času. Hodiny reálného času existují v různých variantách, které se liší spotřebou, cenou a přesností. Většina mikrokontrolérů má již interní RTC, včetně vývojové desky ESP32. Interní RTC u ESP32 je poměrně nepřesné, což nemusí být problém, pokud je například potřeba počkat určitý počet vteřin, kde přesnost nehraje tak velkou roli, jestli zařízení čekalo 2 vteřiny přesně nebo 2 vteřiny a 1 milisekundu. Na druhou stranu, pokud je nepřesné RTC použito pro udržení aktuálního času a pokud je zařízení v provozu několik měsíců nebo i několik let, tak tato nepřesnost způsobí již znatelný rozdíl a aktuální čas na zařízení nebude správný. Tato nepřesnost se dá vyřešit využitím přesnějšího RTC. Jednou z možností je například externí RTC modul, který se používá pro udržení aktuálního času. Externí RTC má velkou výhodu, že obsahuje baterii, která toto RTC napájí. Díky tomu je udržen aktuální čas i při výpadku napájení. V případě výpadku napájení při využití interního RTC by došlo ke ztrátě aktuálního času a bylo by potřeba tento čas opětovně natavit.

Většina generátoru hodin je velice přesná, a proto se pro jejich přesnost využívá jednotky ppm[1].

Vývojová deska ESP32 podporuje více zdrojů hodin reálného času. V následujícím výčtu jsou vybrány tři způsoby, které připadají v úvahu.

- Vnitřní RTC - Jeho frekvence je nestabilní a závislá na teplotě, jeho přesnost je kolem 5 %. Využívá se například pro zpoždění programu, kde není potřeba přesný čas nebo kdy je ESP32 v režimu spánku z důvodu snížení spotřeby.

[1]PPM neboli parts per milion, pro demonstraci například 1 000 000 ppm je stejné jako 100 %

- Externí krystal 32kHz - Je potřeba, aby k čipu ESP32 byl připojen externí krystal. Přesnost se poté liší podle vybraného krystalu. Obecně zde je přesnost kolem 10-30 ppm což je 0,001 - 0,003 %. Při porovnání s předchozími 5 % je zde vidět znatelně vyšší přesnost.
- Externí zdroj RTC - Přesnost zde závisí na vybraném modulu a bývá v podobných hodnotách jako u předchozí varianty, která má přesnost kolem 10-30 ppm. Existují také ale moduly s přesností například 2ppm (0,0002 %), což je ještě 10x přesnější než předchozí varianta externího krystalu.

3.3 Paměť zařízení

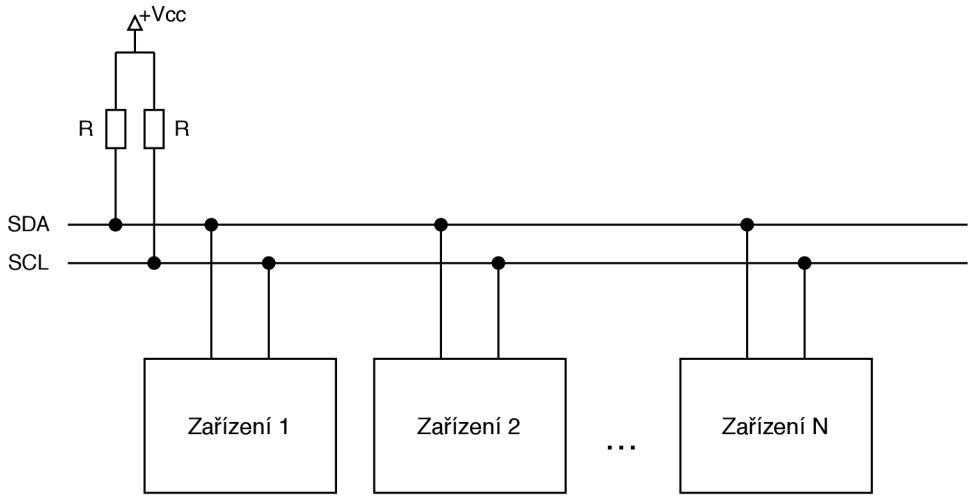
Paměť zařízení se dělí na hlavní 2 typy paměti, které se liší parametry a tím pádem pro jaký typ dat je daná paměť využita.

- RAM - Tato paměť je volatilní, to znamená, že v případě ztráty napájení jsou taktéž ztracena data v dané paměti. Tato paměť se vyznačuje vysokou rychlostí, ale za cenu vyšších pořizovacích nákladů, proto nemívá paměť RAM příliš vysokou kapacitu, u mikrokontrolérů bývá řádově v kB. Pro porovnání dnešní počítače mívají běžně RAM v řádu GB. U mikrokontrolérů se vyžívá pro proměnné programu. Proměnné programu bývají využity často, a proto je potřeba aby k nim přístup byl co nejrychlejší a paměť byla dostatečně rychlá.
- FLASH - Tento typ paměti je nevolatilní, to znamená, že v případě ztráty napájení jsou data zachována. Tato paměť je značně pomalejší, než paměť RAM, díky tomu je ale levnější a mívá větší kapacitu. Tato kapacita bývá u mikrokontrolérů v řádu kB či MB, záleží na vybraném mikrokontroléru. Pro porovnání u dnešních počítačů bývá kapacita běžně v řádu GB až TB. Tato paměť se využívá pro data, která v systému musí zůstat i při výpadku napájení, jako je firmware nebo například databáze známých přístupových karet. U paměti Flash je potřeba vyhnout se častým zápisům do paměti, protože Flash paměť má omezený počet přepisů, který bývá v řádu 10000 až 1000000 přepisů.

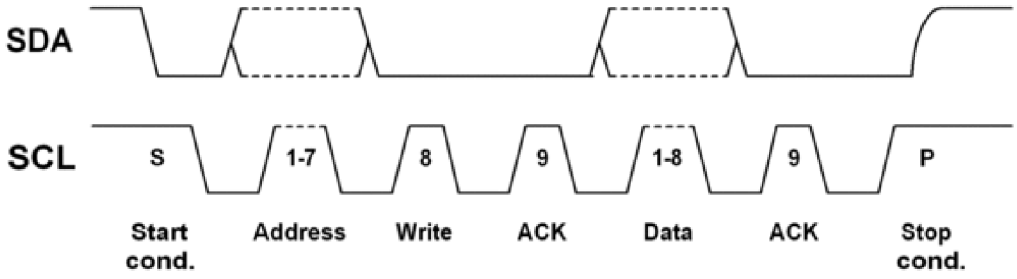
3.4 Sběrnice I2C

Sběrnice I2C, někdy také označována jako IIC, je standard, který umožňuje komunikaci několika zařízení pomocí pouze dvou vodičů. I2C bylo poprvé uvedeno v roce 1982 společností Philips (dnešní NXP). I2C se používá pro zařízení, která nepotřebují příliš vysokou komunikační rychlost, jako může být například D/A převodník, různé senzory nebo například také paměť zařízení. Díky jejímu jednoduchému použití a zapojení je poměrně hodně využívaným typem připojení. Využívá se převážně na spojení více komponent na desce plošných spojů, ale využívá se i pro spojení více komponent pomocí kabelů, kde může ale nastat problém s parazitními parametry, jako je kapacita či odpor, pokud jsou vodiče příliš dlouhé nebo jsou připojené pomocí spojky.

Sběrnice I2C využívá pouze 2 vodičů pro připojení všech zařízení. Tyto dva vodiče se obecně označují jako SDA a ASCL. SDA slouží pro přenos dat a SCL slouží pro přenos hodinového signálu. Komunikace se dělí vždy na zařízení zahajující komunikaci (zařízení označované jako Master) a zařízení se kterým toto zařízení komunikuje (zařízení označované



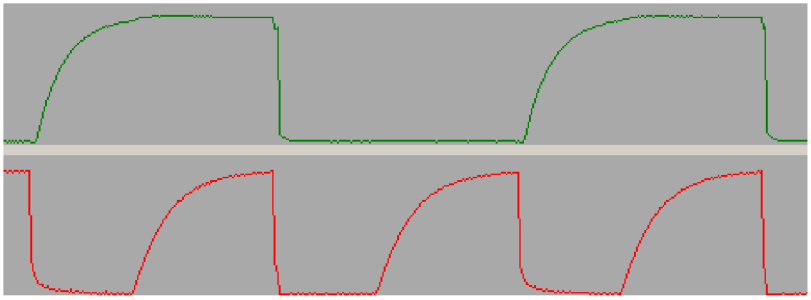
Obrázek 3.1: Sběrnice I2C



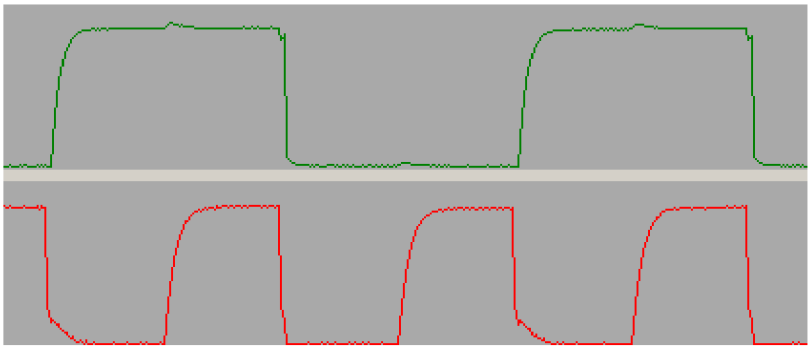
Obrázek 3.2: Typická komunikace na sběrnici I2C (převzato z [https://www.i2c-bus.org/addressing/])

jako Slave). Díky tomu, že je přenášen hodinový signál, tak zařízení typu Slave nepotřebují vnitřní generátor hodin, stačí aby jej obsahovalo zařízení typu Master, tím taktéž odpadá problém se synchronizací generátoru hodinového signálu, který je potřeba řešit u jiných typů sběrnic.

Na obrázku 3.1 lze vidět klasické zapojení I2C sběrnice. Lze zde vidět využití dvou vodičů SDA a SCL, ke kterým jsou připojeny rezistory vůči kladnému napětí tzv. pull-up rezistory. Ty jsou velice důležité z důvodu principu komunikace, která je popsána dále. Dále zde lze vidět připojená zařízení označená 1 až N. Ve většině schématech znázorňujících I2C sběrnici, bývá jedno zařízení označeno za Master a zbytek zařízení za Slave, což může být matoucí, jelikož každé zařízení může být Master nebo Slave z důvodu, že sběrnice I2C podporuje více Master zařízení na jedné sběrnici. Při využití více Master zařízení je potřeba, aby došlo k synchronizaci generátoru hodinové signálu, tento postup definuje protokol I2C, ale většina zapojení bývá se zapojením pouze jednoho Master zařízení. Maximální teoretický počet zařízení, označený v obrázku jako N, je dán rozsahem adresy. Každé zařízení musí mít unikátní adresu vůči dané sběrnici. Adresa bývá typicky 7bitová, tím pádem lze na sběrnici zapojit zároveň až 128 zařízení, ale několik adres je rezervováno pro speciální použití a ve výsledku zbývá volných pouze 112 adres. Je také podporován 10bitový rozsah adres, díky čemuž se dá získat až 1024 adres, ale 10bitový rozsah se momentálně příliš nevyužívá.

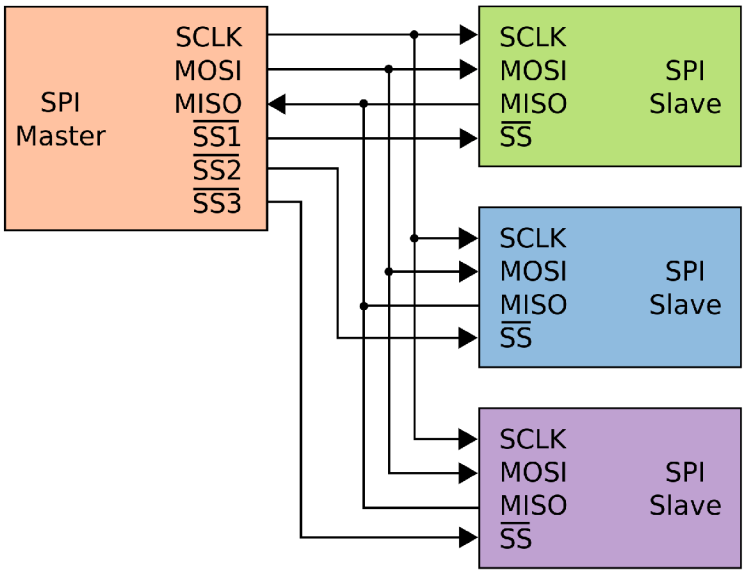


Obrázek 3.3: Zobrazená komunikace na sběrnici I2C, SDA nahoře, SCL dole, hodnoty rezistorů - 10 K, Kapacita sběrnice - 300 pF (převzato z [https://www.i2c-bus.org/i2c-primer/termination-versus-capacitance/])



Obrázek 3.4: Zobrazená komunikace na sběrnici I2C, SDA nahoře, SCL dole, hodnoty rezistorů - 10 K, Kapacita sběrnice - 300 pF (převzato z [https://www.i2c-bus.org/i2c-primer/termination-versus-capacitance/])

Na obrázku 3.2 lze vidět typickou komunikaci protokolu I2C. Komunikace začíná z klidové polohy, která je kladné napětí, z toho důvodu jsou k vodičům připojeny pull-up rezistory, které zajišťují tuto kladnou klidovou polohu. Komunikace začíná ve chvíli, kdy Master zařízení vyšle Start bit (v obrázku označený jako S), a tím přejde z kladného napětí na záporné. Přesně naopak tomu je pak u konce komunikace kdy se přechází ze záporného napětí na kladné. Obě tyto části, jak konec, tak začátek, musí probíhat v době, kdy je hodinový signál v logickém stavu 1, neboli je zde kladné napětí. Přenos dat následně probíhá po bytech, kde po každém bytu přichází potvrzovací bit (v obrázku označen jako ACK), který má logickou úroveň 0. Pokud by na místě, kde má být potvrzovací bit, byla logická úroveň 1 (obecně označovaný jako NACK), tak se jedná o chybu v přenosu a je potřeba případně daný přenos opakovat. Samotné čtení hodnoty z datového vodiče většinou probíhá na náběžné hraně hodinového signálu, proto je potřeba, aby tato hrana byla dostatečně kvalitní, což souvisí s volbou pull-up rezistorů. Na obrázku 3.3, lze vidět vyobrazený průběh na sběrnici I2C s ne příliš vhodnými rezistory, kde je průběh zkreslený a přechod 0 -> 1 je pomalý a mohlo by díky tomuto docházet k chybnému přenosu. Na druhém obrázku 3.4 jsou rezistory zvoleny lépe. Je zde vidět průběh, který se více podobá vhodnému obdélníkovému průběhu signálu. [12]



Obrázek 3.5: Příklad zapojení zařízení na sběrnici SPI (převzato z [https://cs.wikipedia.org/wiki/Serial_Peripheral_Interface])

3.5 Sběrnice SPI

SPI sběrnice se stejně jako I2C využívá pro komunikaci mezi zařízeními, ale od I2C se poměrně liší. Má vyšší rychlost přenosu, nepotřebuje zapojení externích pull-up rezistorů, ale potřebuje více vodičů na zapojení. Na obrázku 3.5, lze vidět příklad zapojení 4 zařízení, kde jedno zařízení je označeno jako Master a ostatní zařízení jsou označeny jako Slave. Sběrnice SPI je navržena pro funkci s jedním Master zařízením a N zařízeních Slave, na rozdíl od I2C, která je navržena již s podporou více Master zařízení. Při porovnání se zapojením sběrnice I2C (na obrázku 3.1) lze vidět, že SPI využívá více vodičů a obecně má složitější schéma zapojení. Stejně jako tomu je u I2C, tak hodinový signál zde generuje pouze Master zařízení a díky tomu není potřeba, aby ostatní zařízení obsahovala vlastní generátor hodinového signálu a docházelo k synchronizaci tohoto generátoru. Hodinový signál je na obrázku označen jako SCLK (zkratka pro Serial Clock). Dalším velkým rozdílem oproti I2C je že komunikace je obousměrná, a ne pouze jednosměrná, proto je potřeba další vodič navíc při zapojení. O přenos dat se tedy starají dva vodiče, které jsou v obrázku označené jako MOSI (zkratka pro Master Out Slave In) a MISO (zkratka pro Master IN Slave OUT), již z názvu tohoto označení vyplývá směr komunikace, díky čemuž se následně zařízení jednodušeji zapojuje. Dalším vodičem na obrázku je SS (zkratka pro Slave Select), který slouží pro výběr zařízení, se kterým má probíhat komunikace. Nevyužívá se tedy adres, jako tomu bylo u I2C. Díky tomu odpadá problém, kdy máme například více zařízení se stejnou adresou danou výrobcem a nedá se změnit nebo problém s rozsahem adres, kde bylo omezení na maximální počet zařízení na sběrnici. Na druhou stranu je potřeba, aby ke každému zařízení vedl vodič, což u většího počtu zařízení může být problematické. [19]

3.6 Konfigurace vestavěných zařízení

Existuje několik možností, jak konfigurovat vestavěná zařízení. V případě levných zařízení, kdy je kladen důraz na minimální množství komponent, se typicky využívá externí programátor, který je nutné připojit. V případě přítomnosti klávesnice či tlačítek lze využít konfigurace pomocí určitých kombinací stisknutí tlačítek pro danou konfiguraci. Toto jsou typické cesty, které lze nalézt u mnoha komerčně dostupných produktů. V dnešní době se však nabízí řada pokročilejších a uživatelsky přívětivějších variant, které nikterak nenavyšují cenu zařízení. Mnoho vývojových desek v dnešní době má součástí modul WiFi, který lze využít ke zprostředkování konfiguračního rozhraní.

3.6.1 Konfigurace pomocí klávesnice

U vestavěných systémů se lze často setkat s konfigurací pomocí klávesnice (většinou obsahující pouze číslice a pár speciálních znaků). Konfigurace je poté řešena například kombinací určitých kláves. Pokud zařízení neobsahuje nebo obsahuje pouze malý displej, tak je poté konfigurace často dost neintuitivní a je potřeba se řídit návodem. Tento způsob konfigurace se používá převážně u systémů, které neobsahují WiFi.

3.6.2 Konfigurace pomocí aplikace

Jednou z možností je využití nativní aplikace. Ta je určena pouze pro jeden konkrétní operační systém jako je Android, IOS, Windows atd. Z tohoto vyplývá problém s kompatibilitou, kde je pro každý operační systém potřeba vytvořit vlastní aplikaci, což je časově náročné. Aplikace nemusí být ale pouze nativní, existují také hybridní aplikace či progresivní webové aplikace, které využívají webových technologií, díky čemuž zůstává jádro aplikace stejné pro všechny operační systémy. Nativní aplikace většinou běží plynuleji, než další zmíněné varianty a mají například přístup k různým funkcím, ke kterým se u hybridních aplikací poté přistupuje pomocí pluginů.[4]

Problémem u aplikací je, že aplikace musí projít schvalovacím procesem před tím, než bude umístěna na App Store[2] či Google Play[3]. Toto by se dalo obejít distribuováním aplikace mimo App Store či Google Play, ale pro uživatele je pravděpodobně lepší, když aplikace bude na uvedené službě.

3.6.3 Konfigurace pomocí webového rozhraní

U webového rozhraní odpadá závislost na operačním systému, protože je konfigurační rozhraní zobrazeno pomocí webového prohlížeče, který obsahuje skoro každý operační systém. Díky tomu se snižuje časová náročnost, protože není potřeba pro každý operační systém vyvíjet aplikaci zvlášť a stejně tak není potřeba se zabývat schvalovacím procesem aplikace pro umístění do App store nebo Google Play. Nevýhodou je poté, že zařízení musí obsahovat HTML stránky a tím je zabrána část Flash paměti.

[2] Schvalovací proces pro App Store https://developer.apple.com/app-store/review/

[3] Schvalovací proces pro Google Play https://support.google.com/googleplay/android-developer/answer/9815348

3.6.4 Realizace uživatelského rozhraní využívajících webových technologií

Pro účely konfigurace pomocí aplikace či webového rozhraní je možno využít dvou způsobů, kdy se zařízení připojí buď k již existující síti WiFi nebo zařízení spustí svou vlastní síť WiFi.

1. Připojení se k síti WiFi - Zařízení je připojeno k jiné existují WiFi síti. Pro zobrazení webového rozhraní poté stačí být připojen na stejné síti, jako je připojeno zařízení. Pokud je na dané síti i přístup k internetu, tak se toto dá využít k načtení prvků, které mohou být uloženy jinde, než na zařízení nebo lze například provést synchronizaci času vůči serveru NTP[4]. Nevýhodou je poté, že zařízení vyžaduje v dosahu síť WiFi, protože se jedná o přístupový systém, který může být umístěn i do prostředí, kde WiFi síť není k dispozici, tak by tento způsob byl značně omezující.

2. Spuštění WiFi sítě, ke které se uživatel připojí - U této varianty odpadá nutnost jiné WiFi sítě v dosahu, díky tomu není omezené umístění výsledného zařízení v dosahu existující WiFi sítě. Vzhledem k tomu, že u tohoto způsobu nemá zařízení přístup k internetu, tak nelze provést synchronizaci času pomocí NTP, stejně tak nelze načíst externí prvky z internetu a je potřeba aby vše obsahovalo zařízení uložené v paměti. Z tohoto důvodu se například nedá využít frameworku pro výsledné stránky, protože tyto frameworky jsou většinou načítány externě nebo zabírají příliš mnoho paměti a nelze je tak umístit do paměti zařízení. Nemožnost využití frameworku také částečně ztěžuje vývoj webového rozhraní.

3.6.5 Webové technologie

Pro účely realizace konfigurace pomocí webového rozhraní je potřeba implementovat webové rozhraní, které komunikuje se zařízením. Proto jsou v této části stručně uvedené související technologie, potřebné při implementaci.

HTTP

HTTP (zkratka pro Hyper Text Transfer Protocol) je bezstavový protokol pro komunikaci na aplikační úrovni, který je definován v RFC 2126 [8]. HTTP protokol pracuje na principu požadavků a odpovědí. Klient zašle na server požadavek obsahující typ metody, URI, verzi protokolu a obsah zprávy. Server poté odpoví pomocí kódu, který značí kategorii odpovědi (například kód 200 - úspěch, 404 - nenalezeno), verzi protokolu a obsah zprávy. Mezi nejčastější typ metody patří GET a POST.

- GET - Používá se pro získání dat ze serveru specifikovaných pomocí URI. Lze takto získat například HTML stránky. Pomocí této metody lze ale také odeslat data na server, například data z formuláře. Data z tohoto formuláře jsou obsažena v URL, tím pádem je například může vidět uživatel v adresním řádku prohlížeče.

- POST - Používá se pro zaslání dat na server. Tyto data jsou specifikována v těle HTTP požadavku, díky tomu nejsou vidět například v adresní řádce prohlížeče.

[4]NTP protokol se využívá pro synchronizaci času více viz například: https://cs.wikipedia.org/wiki/Network_Time_Protocol

HTML

HTML (zkratka pro Hyper Text Markup Language) je značkovací jazyk využívaný pro zobrazení webových stránek. Pomocí HTML je definována struktura webové stránky, k tomuto je využíváno "tagů" a případně jejich atributů. Tyto tagy jsou většinou v páru, značící začátek a konec, například <h1>Nadpis</h1>. Některé speciální jsou ale samostatné jako je například
, které slouží pro vložení znaku nového řádku. V klasickém případě je skupina bílých znaků nahrazena jednou mezerou a řádek je zalomen až je potřeba, díky tomu nelze například zobrazit "normálně"text, který používá pro zalomení znaků nového řádku.

CSS

CSS (zkratka pro Cascading Style Sheets) se využívá pro definování stylu zobrazení HTML prvků. Pomocí stylů lze definovat například barvu, velikost, styl písma, odsazení nebo i také možnost zobrazení nového řádku, pomocí znaku nového řádku a ne pouze pomocí tagu
 (white-space: pre-line). Tyto styly je možné přiřadit k prvkům pomocí externích, interních nebo inline stylů.

- Externí - Styly jsou načteny z externího souboru.
- Interní - Styly jsou obsaženy v daném HTML souboru pomocí tagu <style>.
- Inline - Styly jsou obsaženy přímo u HTML tagu pomocí atributu style.

JavaSript

JavaScript je programovací jazyk využívaný na webu. Lze pomocí něj měnit strukturu a obsah HTML prvků, styly CSS, zasílat HTTP požadavky nebo například kontrolovat správnost vložených dat ve formuláři. Hlavní výhodou JavaScriptu na webové stránce je, že je spuštěn na straně klienta a není tím zatěžován server.

Ajax

Ajax (zkratka pro Asynchronous JavaScript And XML) je technologie využívaná pro komunikace se serverem na webu. Pomocí této technologie je možné získat data z webserveru i po načtení stránky, aktualizovat stránku bez nutnosti její znovu načtení anebo zaslání dat na webserver na pozadí. Díky tomu je například možné pravidelně aktualizovat hodnotu zobrazenou na webové stránce, bez nutnosti znovu načtení celé stránky. Díky tomu, že není přenášena celá stránka, ale například pouze jedna hodnota, tak je snížen datový přenos po síti nebo také zatížení zařízení. [16]

Kapitola 4

Návrh

Tato kapitola se zabývá návrhem vestavěného systému, která je rozdělena do dvou hlavních částí kde je postupně popsána hardwarová část a softwarová část. Navržený systém se skládá z řídicí jednotky, která obsahuje WiFi a Bluetooth. K této řídicí jednotce jsou poté připojeny periferie jako je čtečka karet na bázi čipu MFRC522, LCD displej a zvuková signalizace. Při návrhu byla převážně vzata v úvahu cena výsledného zařízení a snadné použití pro uživatele. Detailní blokové schéma lze vidět na obrázku 4.1.

4.1 Návrh Hardware

První částí této kapitoly je návrh hardwarové části. V této části jsou popsány hardwarové prvky využité v navrženém systému a jejich alternativy, které byly zvažovány při návrhu.

4.1.1 Čtečka karet a karty Mifare

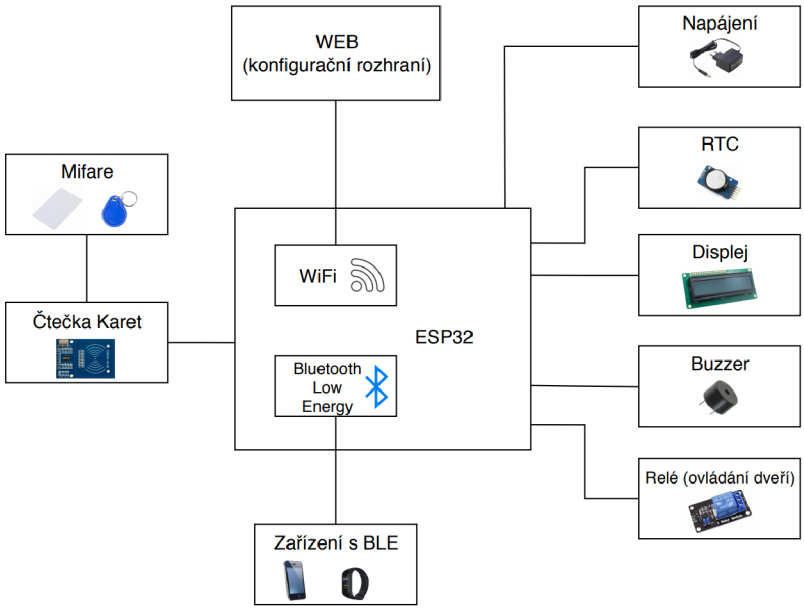
Na trhu existuje několik variant čteček karet pro karty Mifare, které se liší především cenou a formou provedení. Může být připojena k zařízení pomocí USB, což je využito například při připojení čtečky k počítači a nehodí se pro toto zařízení. Další možnou čtečkou by byla například čtečka, komunikující dvouvodičovým protokolem WIEGAND, ale tento typ čtečky je poměrně dražší a neumožňuje získat jiné informace o kartě nežli identifikační číslo. Další možností je využít čtečku karet na bázi čipu MFRC522 [13], která komunikuje pomocí sběrnice SPI (viz sekce 3.5), což je v kombinaci s mikrokontrolérem ideální, zároveň je velice levná, v porovnání s předchozím typem je asi 10x levnější. Z tohoto důvodu jsem se rozhodl vybrat tento typ čtečky. Jako typ karty Mifare, jsem se rozhodl využít Mifare Classis 1K (viz sekce 2.3), převážně z důvodu nízké ceny a dostupnosti.

4.1.2 Napájení

Napájení bude řešeno externě, tím pádem bude potřeba zajistit připojení zařízení do elektrické sítě. Pokud by k zařízení nebyl připojen displej, byla by jeho spotřeba o dost nižší a dalo by se uvažovat i o napájení z baterie, ale i přesto by se musela baterie jednou za čas měnit a napájení z elektrické sítě je u tohoto typu zařízení více praktičtější.

4.1.3 Řídicí jednotka

Jako řídicí jednotku jsem se rozhodl využít vývojovou desku ESP32, která obsahuje WiFi a Bluetooth. Díky tomu že je ESP32 poměrně levné a není potřeba Wifi a Bluetooth kupovat



Obrázek 4.1: Blokové schéma návrhu zařízení

jako externí periferie, tak bude výsledné zařízení levnější, než při výběru jiné varianty. Zároveň jak již bylo znázorněno v tabulce 3.1, tak ESP32 má také podstatně více paměti, kde by například, u mikrokontroléru rodiny Atmel použitých na platformě Arduino, mohlo dojít k nedostatku paměti a bylo by potřeba dokoupit externí modul s dodatečnou pamětí.

4.1.4 Hodiny reálného času

Hodiny reálného času (RTC) lze u zařízení získat více možnými způsoby (viz sekce 3.2). Pro příklad demonstrace důležitosti přesnosti těchto hodin, je zde ukázána maximální možná chyba podle typu zdroje hodin. Ve výpočtech je chyba ukázána na 30 dnech běhu zařízení, což je 2592000 sekund.

- Interní RTC s chybou 5 % - $2592000 * 0,05 = 129600$ sekund. Za dobu běhu 30 dní se může čas lišit až o 129600 sekund což je 36 hodin.
- Externí krystal s chybou 0,002 % (20 ppm) - $2592000 * 0,00002 = 51.84$ sekund. Za dobu běhu 30 dní se může čas lišit až o necelých 52 sekund, což není ani 1 minuta.
- Externí modul RTC s chybou 0,002 % (2 ppm) - $2592000 * 0,000002 = 5.184$ sekund. Za dobu běhu 30 dní se může čas lišit až o něco málo přes 5 sekund.

Z uvedených výpočtů lze vidět, že použití RTC s přesností 5 % je pro delší dobu běhu prakticky nereálné. Přesnost 20 ppm, která má na 30 dnech nepřesnost kolem jedné minuty, je dostatečně přesná, pro použití pro delší běh, ale například na roku běhu by už tento posun mohl být kolem 12 minut, což sice není velká chyba, ale už to není tak příliš zanedbatelné. Z tohoto důvodu jsem se v návrhu rozhodl použít externí modul RTC DS3231 [11], který má chybu 2 ppm. Tento modul je zapojen pomocí sběrnice I2C (viz sekce 3.4). Mohlo by se stát že zařízení přijde o napájení, například z důvodu výpadku elektrické sítě. Díky tomu, že externí RTC modul obsahuje baterii, tak výpadek napájení neovlivní funkčnost RTC a díky tomu navržené zařízení neztratí aktuální hodnotu času i při výpadku napájení.

4.1.5 Displej

Jako displej jsem vybral LCD displej 2x16[1] hlavně z důvodu nízké ceny, aktuální dostupnosti a malé velikosti. Malá velikost může být brána jako výhoda i nevýhoda, příliš velký displej může vypadat prázdně a nevyužitě a příliš malý displej zase může být omezující. Tento typ displeje bývá ve dvou variantách, které se liší způsobem zapojení, buď pomocí 16 pinů, které zajišťují funkcionalitu displeje nebo pomocí přídavného modulu, který je připojen k těmto 16 pinům a řídicí jednotka je poté pouze propojena s tímto přídavným pomocí sběrnice I2C (viz sekce 3.4). Z důvodu zapojení, které vyžaduje značně méně pinů, a tím je neblokuje pro jiné použití, jsem se rozhodl pro variantu displeje s tímto přídavným modulem pro zapojení pomocí sběrnice I2C.

4.1.6 Zvuková signalizace

Pro zvukovou signalizaci jsem se rozhodl využít bzučáku (angl. Buzzer). Při výběru bzučáku existují hlavní 2 varianty. První varianta je bzučák s vnitřním generátorem, k jeho zapojení je potřeba pouze přivést napětí na jeho vývody. Druhou variantou je bzučák bez vnitřního generátoru. Jeho zapojení je značně složitější, protože je potřeba generovat obdélníkový signál externě. Díky tomu je sice možné použít frekvenci podle preference a ne danou výrobcem, ale na úkor složitosti. Vzhledem k tomu, že zde je využit bzučák pouze pro zvukovou signalizaci, tak nezáleží příliš na zvolené variantě. Pravděpodobně není potřeba přesnou frekvenci a stačila by frekvence daná výrobcem, tím pádem by varianta s vnitřním generátorem byla vhodnější, z důvodu lehčího zapojení. Ale z důvodu dostupnosti této součástky a faktu že na výsledném zařízení nebude výběr dělat zásadní rozdíl, jsem se rozhodl použít variantu bzučáku bez vnitřního generátoru, přesněji model BPT-14 [5].

4.2 Návrh Firmware

V druhé části této kapitoly je popsán návrh firmwaru. Je zde popsán způsob ukládání známých karet a BLE zařízení, hlavní smyčka programu a návrh konfiguračního rozhraní s popisem jednotlivých sekcí, do kterých je rozhraní rozděleno a možnosti konfigurace v dané části rozhraní. Dále je zde popsáno rozvržení LCD displeje a návrh vlastních znaků na displeji, popis zvukové signalizace a způsob ukládání historie průchodů s ohledem na životnost paměti Flash.

4.2.1 Přístup pomocí MIFARE

Systém bude mít uložen whitelist UID karet, které mají povolené interakci se systémem, zároveň bude ke každé kartě přiřazeno uživatelské jméno, komu daná karta patří. Dále budou karty rozděleny na dva typy karet. Běžné karty a administrátorské karty. Běžná karta bude sloužit pro přístup, lze ní tedy pouze ovládat dveře. Administrátorské karty budou sloužit pouze ke konfiguraci zařízení. Informace o uživatelském jméně a typu karty by mohly být uloženy přímo na dané kartě, ale z důvodu zvýšení bezpečnosti jsou tyto informace uložené v paměti zařízení, aby se zamezilo možnosti, kdy by si uživatel neoprávněně upravil hodnoty uložené na kartě.

Navržené zařízení bude dále podporovat dva pracovní módy, dle kterých se liší funkčnost běžných karet.

[1] 2x16 se v souvislosti s displeji používá jako označení pro 2 řádky a 16 znaků na řádku

1. Normální mód - Zde běžná karta způsobí dočasné odemknutí dveří, na nastavený počet vteřin v konfiguraci a zároveň vypíše na displej čas průchodu, speciální značku značící průchod a uživatelské jméno přiřazené ke kartě.

2. Trigger mód - Zde běžná karta způsobí trvalé odemčení/zamčení dveří (změna stavu na opačný) a na displej je vypsán čas, kdy nastala tato změna, speciální značka, značící zamčené či odemčené dveře a taktéž uživatelské jméno.

Maximální počet karet jsem navrhl 300, z důvodu, že je potřeba mít databázi karet uloženou i v paměti RAM, pro rychlý přístup a paměť RAM nemá tak vysokou kapacitu.

4.2.2 Přístup pomocí BLE

Druhá možnost identifikace uživatele je pomocí BLE zařízení, které je v roli serveru, což může být například fitness náramek. Uživateli je poté povolen průchod, pokud je známé BLE zařízení v dostatečné blízkosti. Tato vzdálenost je určena na základě síly signálu zařízení (RSSI) a je možné pro každé zařízení určit vlastní hraniční hodnotu. Tento způsob identifikace má být ulehčení pro uživatele, kdy uživatel není nucen vytahovat přístupovou kartu a přiložit ji ke čtečce. Místo toho stačí, aby měl například zmíněný fitness náramek na ruce a po příchodu ke dveřím je mu automaticky povolen průchod. Samozřejmě ne všem uživatelům se může tento koncept líbit, a proto je možnost vypnout průchod pomocí BLE v konfiguraci na webovém rozhraní. Z důvodu bezpečnosti je při prvním spojení vyžadován bonding (viz sekce 2.4.2). Při odstranění zařízení z databáze známých zařízení je potřeba také odstranit záznam o bondingu v paměti řídicí jednotky. Maximální počet známých zařízení jsem navrhl 100. U BLE zařízení je potřeba ukládat bonding, který například na rozdíl od karet MIFARE zvyšuje velikost využité paměti.

4.2.3 Řídicí jednotka

Řídicí jednotka obstarává hlavní funkčnost zařízení. Zajišťuje komunikaci s ostatními prvky systému jako je displej, RTC, čtečka karet, bzučák a řízení přístupu. Dále obstarává funkčnost WiFi a Bluetooth, které jsou již součástí této jednotky. Hlavní smyčka programu se dělí na 2 části, které by šly označit jako stav systému.

- Pracovní stav - V hlavní smyčce programu je pravidelně aktualizován čas na displeji, kontrola zda uživatel nepřiložil kartu nebo zda nebylo nalezeno známé BLE zařízení v dosahu (část detekce BLE je řešena jako paralelní proces, protože scan blízkých zařízení trvá několik vteřin a blokoval by takto hlavní smyčku). V případě přiložení karty či nalezeného BLE zařízení jednotka ověří, zda se nachází v databázi a podle toho odmítne či povolí přístup.

- Stav konfigurace - V hlavní smyčce je stále aktualizován čas na displeji. Neprovádí zde pravidelnou kontrolu přiložených karet a vyhledávání BLE zařízení, ale pouze reaguje na požadavky podle konfigurace.

4.2.4 Konfigurační rozhraní

Konfiguraci zařízení je možno provést více způsoby. Konfigurace by mohla být provedena například pomocí klávesnice či jiných tlačítek připojených k zařízení, pomocí mobilní aplikace nebo pomocí webového rozhraní (viz sekce 3.6). Z důvodu, že zařízení má být snadné

pro použití uživatelem navrhuji konfiguraci pomocí webového rozhraní, díky tomu bude také konfiguraci možné provést nezávisle na platformě, kterou uživatel využije.

Návrh Konfigurace

Zařízení se přepne do stavu konfigurace přiložením administrátorské karty ke čtečce, tím se zapne síť WiFi, ke které se může uživatel připojit. Tento způsob jsem navrhl, aby u zařízení nebyla omezena možnost umístění i do oblasti bez pokrytí existující WiFi sítí. Na spuštěné WiFi síti si uživatel může zobrazit webové rozhraní na dané adrese. Pro připojení ke spuštěné WiFi síti musí uživatel zadat heslo. Díky tomu je zamezeno, že se k síti WiFi připojí neoprávněná osoba. Jméno a heslo spuštěné WiFi sítě bude možné změnit na webovém rozhraní. Pro případ, že uživatel zapomene nastavené heslo, lze provést reset jména a hesla WiFi do továrního nastavení přiložením 4x za sebou administrátorské karty ke čtečce do 10 s, pokud je zapnutá síť WiFi. O průběhu resetu, či kolikrát je ještě potřeba přiložit kartu pro reset, je uživatel informován pomocí výpisu na displeji.

Na webovém rozhraní následně budou 4 samostatné stránky, ke kterým se dá přistoupit pomocí menu v horní části obrazovky. Konfiguraci je poté možno na webovém rozhraní ukončit, čímž se zařízení přepne zpět do pracovního módu. Mohlo by se například stát, že administrátor zapomene konfiguraci ukončit a zařízení by takto zůstalo v módu konfigurace. Z tohoto důvodu je zde nastaven časovač dvou minut. Po dvou minutách neaktivity na webovém rozhraní řídicí jednotka ukončí konfiguraci a přepne se sama zpět do pracovního módu.

Konfigurační rozhraní bude dostupné na dané adrese, kterou jsem zvolil jako 192.168.3.1, na této adrese je vrácena základní stránka, která uživatele automaticky přesměruje na stránku s konfigurací hodnot zařízení. Pokud by automatické přesměrování nefungovalo, bude na této stránce zobrazeno také menu pro přístup k ostatním stránkám. Menu bude obsahovat následující položky:

- Konfigurace hodnot zařízení - Zde se dá nastavit aktuální čas zařízení, pracovní mód zařízení (viz sekce 4.2.1), čas po který budou dveře odemčeny v normálním módu. Nebo také možnost vypnout si detekci BLE (viz sekce 4.2.2) nebo vypnutí zvukové signalizace (viz sekce 4.2.6). Dále se zde dá nastavit jméno a heslo spuštěné WiFi sítě. Heslo musí splňovat požadavky na alespoň 8 znaků/čísel.

- Konfigurace BLE - Zde je správa známých BLE zařízení. Je zde možnost spustit scan blízkých BLE zařízení. Tento scan trvá 5 sekund a následně jsou zobrazeny výsledky v tabulce, kde je možnost si jedno ze zařízení vybrat pro připojení. Po připojení k vybranému zařízení je zobrazována a pravidelně aktualizována aktuální hodnota síly signálu, která určuje vzdálenost BLE zařízení od řídicí jednotky. Toto slouží pro nastavení hraniční hodnoty, která slouží pro odemčení dveří k průchodu uživatelem. Pokud se toto zařízení přiblíží dostatečně blízko a překročí tím tuto nastavenou hodnotu, jsou mu odemčeny dveře pro průchod a zůstávají odemčené dokud se zase zařízení nevzdálí dostatečně daleko, aby nebyla překročena nastavená hraniční hodnota. Kromě hraniční hodnoty se zde přiřazuje k vybranému zařízení uživatelské jméno, které slouží pro historii průchodů a pro zobrazení na displeji. Po nastavení těchto hodnot je potvrzením přidáno zařízení do známých zařízení. V případě že vybrané zařízení již bylo mezi známými zařízeními, tak jsou pouze jeho hodnoty aktualizovány na nové.

Další věcí na stránce, je možnost získání všech známých BLE zařízení a jejich následná konfigurace bez nutnosti aby dané zařízení bylo v dosahu. Známá zařízení jsou vyobrazeny jako tabulka, kde je možno upravit uživatelské jméno nebo hraniční hodnotu pro odemčení dveří (i když její upravení je pravděpodobně lepší s připojeným BLE zařízením). Je zde také možnost zařízení odstranit ze známých zařízení.

- Konfigurace karet - Zde je možno spravovat známé přístupové karty. Konfigurace je podobná konfiguraci BLE. Je zde možnost přidání nové karty ke známým kartám, toto je provedeno nastavením uživatelského jména a vybráním typu karty, jako běžnou nebo administrátorskou, jak bylo popsáno v sekci 4.2.1. Po potvrzení je uživatel vyzván po přiložení karty ke čtečce. Tuto akci musí uživatel provést do 10 s, jinak je akce stornována. O stavu registrace je uživatel informován na webovém rozhraní. Pokud již karta byla v seznamu známých karet, dojde k aktualizaci hodnot na nové.

  Podobným způsobem je možno odstranit kartu ze systému, po zvolení akce odstranění karty je uživatel vyzván aby přiložil kartu ke čtečce a tím bude karta odstraněna ze systému. Na tuto akci má stejně jako u přidání nové karty 10 s, jinak je akce stornována.

  Stejně jako tomu bylo u konfigurace BLE, tak lze zde získat tabulku všech známých karet v systému a upravit jejich hodnoty. Dá se zde změnit uživatelské jméno nebo také typ karty, a lze zde i kartu odstranit ze systému.

- Historie průchodů - Zde je vypsána historie průchodů ve formátu čas - typ průchodu - uživatelské jméno, kde každý záznam je na vlastním řádku. Více v samostatné sekci 4.2.7.

- Vypnutí konfigurace - Poslední položkou je pouze tlačítko v menu, sloužící pro vypnutí konfigurace, přesněji tedy sítě WiFi.

Podporované URL

Webserver bude na následujících URL, které budou vyžity v menu, vracet HTML soubory:

- / - Stránka načtená po zadání adresy 192.168.3.1 do prohlížeče. Vrací základní stránku index.html, která automaticky přesměruje uživatele na konfiguraci hodnot zařízení (požadavek /config). Pokud by toto automatické přesměrování nefungovalo, obsahuje zde také menu k přístupu na ostatní stránky.

- /config - Vrací HTML soubor s konfigurací hodnot zařízení.

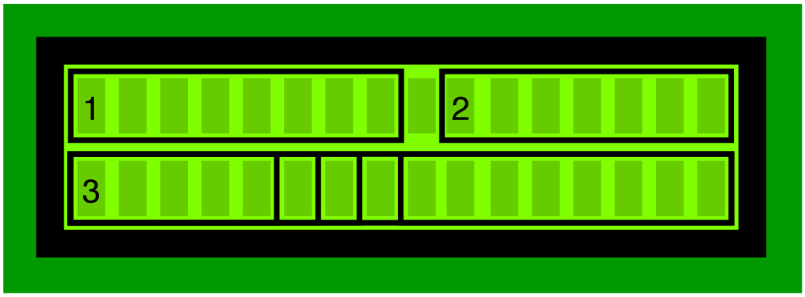
- /ble_config - Vrací HTML soubor s konfigurací BLE zařízení.

- /cards_config - Vrací HTML soubor s konfigurací karet Mifare.

- /history - Vrací HTML soubor, kde bude zobrazena historie průchodů.

Pro odeslání požadavků či formulářů z webového rozhraní je využito technologie AJAX, díky čemuž není po každém požadavku načítána celá stránka a nemusí zařízení přistupovat znovu do paměti FLASH, ve které je tato stránka uložena. Některé prvky webového rozhraní by nebylo možné bez této technologie vyřešit, jako je například pravidelná aktualizace hodnoty síly signálu připojeného zařízení. Z tohoto důvodu nezáleží u odeslání dat na server zda bude využito metody GET nebo POST, protože při odesílání požadavků na pozadí není URL viditelná pro uživatele. Pro konfiguraci budou využity následující URL:

- /start_scan - Spustí scan blízkých BLE zařízení, který trvá 5 s. Výsledky jsou poté zaslány zpět.
- /select_ble_device - Zvolení jednoho z nalezených zařízení a následné připojení k tomuto zařízení.
- /update_selected_ble_device - Vrací hodnotu síly signálu připojeného zařízení. Případně zprávy Connecting, pokud se řídicí jednotka stále připojuje k zařízení nebo Disconnected pokud není řídicí jednotka k žádnému zařízení připojena.
- /save_ble_device - Zvolené zařízení je uloženo do databáze jako známé zařízení.
- /get_known_ble - Vrací seznam známých BLE zařízení.
- /remove_ble_table - Odstranění známého BLE zařízení z databáze.
- /update_ble_table - Aktualizace hodnot známého BLE zařízení.
- /new_mifare_card - Přidání nové karty do systému. Pro přidání karty do systému je potřeba přiložit kartu ke čtečce do 10s.
- /remove_mifare_card - Odstranění karty ze systému. Pro odstranění karty ze systému je potřeba přiložit kartu ke čtečce do 10s.
- /update_new_card_state - Informace o průběhu přidání či odstranění karty ze systému.
- /get_known_mifare_cards - Vrací seznam známých karet.
- /remove_mifare_card_table - Odstranění známé karty ze systému.
- /update_mifare_card_table - Aktualizace hodnot známé kary.
- /get_device_time - Získání času ze zařízení.
- /device_config_set_time - Nastavení času na zařízení.
- /device_config_set_params - Natavení konfiguračních hodnot zařízení.
- /get_device_params - Získání aktuálních konfiguračních hodnot zařízení.
- /get_wifi_name - Získání aktuálního jména WiFi.
- /device_config_set_wifi - Nastavení nového jména a hesla WiFi.
- /get_device_history - Získání historie průchodů.
- /wifi_off - Ukončení konfigurace (vypnutí WIFi).



Obrázek 4.2: Návrh rozvržení na displeji 1. Aktuální čas zařízení 2. Stav zařízení 3. Poslední akce systému

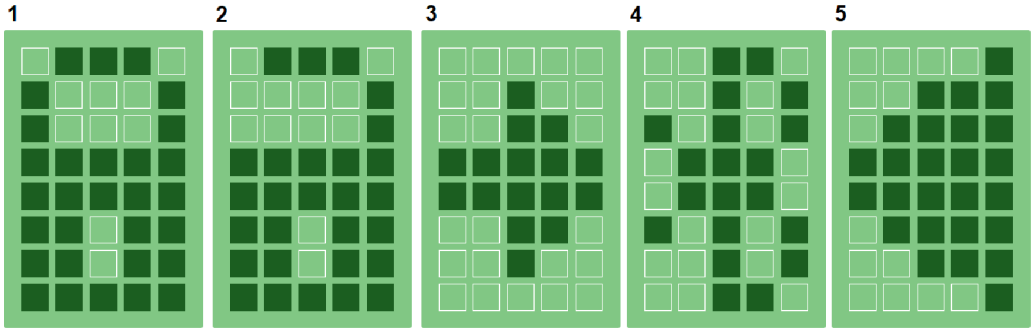
4.2.5 Displej

Displej slouží převážně pro rychlé a jednoduché zjištění stavu systému, které by jinak muselo být například na webové stránce v konfiguraci. Na obrázku 4.2 lze vidět rozvržení displeje. Na displeji v části označené "1" bude zobrazen aktuální čas zařízení ve formátu HH:MM:SS, vzhledem k tomu, že zařízení si ukládá historii průchodů, tak je potřeba aby byl čas správný. Díky tomu, že je čas vypsán na displeji, lze jednoduše zjistit případný špatný čas a následně jej opravit na webovém rozhraní. V části označené "2" bude aktuální stav zařízení, kde bude vypsán aktuální mód (trigger nebo normální) a jestli je zapnuté či vypnuté BLE a bzučák. Část označená jako "3" je rozdělena na 3 části: levá, prostřední a pravá část (i když na obrázku to může být matoucí s rozdělením na 5 částí). Celá tato část se využívá pro poslední stav systému. V první části nejvíce vlevo je zobrazen čas poslední akce ve formátu HH:MM. V prostřední části je zobrazena ikonka posledního stavu. A v poslední části nejvíce napravo je poté zobrazeno uživatelské jméno uživatele, který akci provedl. Tato část je také využívána pro některé akce (související s konfigurací pomocí WiFi) jako celek, kde je přes celou část pouze vypsána poslední akce systému. Z obrázku si lze povšimnout, že v části "3" je pro uživatelské jméno vyhrazeno místo pro 8 znaků. Uživatelské jméno není omezeno na přesný počet znaků a jeho výpis poté záleží na jeho délce. V následujícím výčtu lze vidět návrh výpisu v závislosti na délce uživatelského jména.

1. Kratší než 8 znaků - Jméno je vypsáno se zarovnáním vlevo a zbylé místo je ponecháno prázdné.
2. Přesně 8 znaků - Jméno je vypsáno celé.
3. Delší než 8 znaků - Je vypsáno pouze prvních 8 znaků z uživatelského jména.

Pro určité akce či stav systému jsem navrhl vlastní znaky, které budou zobrazeny na displeji. Pro návrh těchto znaků jsem využil externího nástroje[2]. Na obrázku 4.3 lze vidět navržené znaky: uzamčený a odemčený zámek značící trvalé uzamčení a odemčení pomocí karty, šipka značící průchod pomocí karty, BLE značící stav zapnuté či vypnuté identifikace pomocí BLE a také průchod pomocí BLE a bzučák značící zapnutí či vypnutí zvukové signalizace.

[2]Nástroj je dostupný zde: https://maxpromer.github.io/LCD-Character-Creator/



Obrázek 4.3: Návrh vlastních znaků na displeji 1. Zamčený zámek 2. Odemčený zámek 3. Šipka značící průchod 4. BLE 5. Bzučák

4.2.6 Zvuková signalizace

Bzučák nebo obecně zvuková signalizace je v navrženém systému použita pro zjednodušení použití systému. Pomocí zvukové signalizace uživatel zjistí úspěch či neúspěch akce. Například se může stát, že uživatel bude mít pohromadě více přístupových karet a systém tak nemusí správně kartu rozeznat a neproběhne identifikace uživatele. Tento případ by šlo zjistit například tím, že není akce vypsaná na displeji, ale taková implementace by byla z praktického hlediska nevhodná. Proto navržený systém obsahuje zvukovou signalizaci. Systém takto úspěšnou identifikaci uživatele (přiložení karty nebo BLE zařízení v dostatečné blízkosti) signalizuje jedním pípnutím a neúspěšnou identifikaci (neznámá karta) dvěma pípnutími. Samozřejmě každému nemusí zvuková signalizaci vyhovovat, a proto je možné ji vypnout v konfiguraci.

4.2.7 Historie průchodů

Zařízení si ukládá historii průchodů v normálním módu, v módu trigger si zařízení ukládá změnu stavu tj. odemčení či zamčení dveří. Z tohoto důvodu je potřeba aby zařízení obsahovalo přesný čas o který se stará externí modul RTC (viz sekce 4.1.4). Historie průchodů se dá využít ke kontrole, pokud by například nastal problém. Dalším využitím by mohla být kontrola docházky, ale vzhledem k tomu, že zařízení není připojeno k internetu, tak by toto bylo poměrně nepraktické.

Historii jsem navrhl v následujícím formátu: Datum - akce - uživatelské jméno, kde každý záznam je oddělen pomocí nového řádku. Datum je ve formátu D.M.Y H:M:S. Typ akce může být průchod pomocí karty, průchod pomocí BLE, trvalé odemčení pomocí karty nebo trvalé uzamčení pomocí karty. Uživatelské jméno poté závisí na uživatelském jménu přiřazeném k dané kartě či BLE zařízení.

Z důvodu, že historie průchodů by měla zůstat uchována i při výpadku napájení tak je potřeba ji ukládat do paměti FLASH, která má ale omezenou kapacitu a omezený počet přepisů, z tohoto důvodu je potřeba navrhnout implementaci tak, aby docházelo co k nejméně přepisů paměti. Řešení tohoto problému jsem navrhl pomocí dvou oddělených částí historie. Jakmile by byla přesáhnuta celková kapacita, tak je první část historie celá smazána a na její místo je umístěna druhá polovina historie. Na místo, kde byla druhá část jsou poté ukládány nové záznamy historie průchodů. Tento cyklus se opakuje při každém přesáhnutí kapacity historie. Z důvodu omezeného počtu zápisů do paměti FLASH jsem navrhl ukládání několika záznamů prvně do paměti RAM a až po překročení zvolené hod-

noty jsou tyto záznamy zapsány jako celek do paměti FLASH, čímž je snížen počet zápisů do paměti. Protože je potřeba tuto historii mít i v paměti RAM, aby s ní bylo možné pracovat, tak jsem její maximální kapacitu navrhl 50 kB. Tato kapacita je maximální počet uložených záznamů, po překročení této hodnoty je odstraněna první polovina historie, aby bylo uvolněno místo pro novou historii. Celkový počet záznamů o průchodu je poté závislý na délce uživatelských jmen a na způsobu uložení v paměti FLASH.

Kapitola 5

Implementace

V této kapitole jsou popsány implementační detaily jednotlivých částí systému. Prvně je zde popsán výběr programovacího jazyku, vývojového prostředí a frameworku, pomocí kterých byla implementace realizována. U každé části systému je uvedeno ve kterém zdrojovém souboru se nachází a použité knihovny, případně alternativy knihoven, které připadaly v úvahu. Dále jde zde popsán způsob implementace a u některých částí systému je i ukázka zajímavých částí kódů. Poslední částí této kapitoly je ověření funkčnosti a dále je zde diskutována bezpečnost výsledného zařízení.

5.1 Programovací jazyk

Před samotnou implementací je potřeba si rozmyslet výběr vhodného programovacího jazyka, vývojového prostředí (IDE) a nebo také frameworku. Podle čehož se následně odvíjí složitost implementace, rychlost výsledného zařízení či požadavek na paměť zařízení. Vzhledem k tomu, že se bude jednat o programování mikrokontroléru, je potřeba vybrat jazyk pro to určený. Většinou se jedná o jazyk, který vychází z jiného jazyka a je patřičně upraven, aby se dal použít pro mikrokontroléry.

Vybrat se dá z mnoha programovacích jazyků. Mezi nejpoužívanější patří pravděpodobně C a C++. Jejich hlavní výhodou je rychlost a nízká paměťová náročnost. Při porovnání například s jazykem Python, tak v C/C++ trvá aplikace obecně vytvořit déle, ale za to aplikace poté běží rychleji, v Pythonu naopak tedy je vývoj aplikace rychlejší s tím, že aplikace je poté více náročnější na běh. Právě z důvodu "jednoduchosti"dává v poslední době více a více lidí přednost právě například Pythonu než C/C++ při tvorbě klasických aplikací, při tvorbě programu pro mikrokontroléry se stále více využívá C/C++ právě z důvodu již zmíněné vyšší rychlosti.

Samozřejmě ne každému C/C++ vyhovuje a pokud někdo programuje pouze v jednom programovacím jazyku tak by přechod na C/C++ mohl být více náročný a je lepší zvolit jiný jazyk. Jednou z možností je MicroPython [9], který implementuje Python 3.4 (s vybranými prvky z vyšších verzí). MicroPython implementuje pouze podmnožinu funkcí standardních knihoven. Některé funkce tedy můžou chybět, ale syntax zůstává stejná. Jedná se tedy o odlehčenou verzi Pythonu pro mikrokontroléry. Od MicroPythonu je následně poté odvozený jazyk Circuit Python. Další možností může být NodeMcu, který vychází z programovacího jazyku Lua.

Z důvodu, že je momentálně nejpoužívanější C/C++ jsem se pro tuto variantu rozhodl i já při výběru. Tím že je nejpoužívanější se dá očekávat dostupných více zdrojových ma-

teriálů, více knihoven apod. S tímto souvisí i výběr frameworku. Hlavní dvě varianty jsou ESP-IDF a Arduino. ESP-IDF jak už vyplývá z názvu je přímo určen pro ESP, skládá se z balíku funkcí k řízení chodu ESP a je určen převážně pro jazyk C. Jeho použití je značně složitější, některé základní operace vyžadují volání více různých funkcí, které na sobě závisí. Díky tomu má programátor větší kontrolu nad vykonávanými instrukcemi, ale za úkor zmíněné vyšší složitosti. Framework Arduino je založen na frameworku pro vývojové desky Arduino a je určen převážně pro jazyk C++. Jeho použití je jednodušší. Operace, které se například skládají z více různých funkcí v ESP-IDF, jsou zde spojeny do jedné funkce, která vše provede. Díky tomu je použití jednodušší, ale programátor má ve výsledku nižší kontrolu nad tím, co se vykonává. Další relativní výhodou je, že Arduino framework provádí některé nastavení automaticky a nemusí je provádět programátor. Díky tomu, že framework Arduino je postaven na ESP-IDF, tak zde lze využívat i funkcí z ESP-IDF a díky tomu dosáhnout stejné kontroly jako u ESP-IDF. Výhodou je také zaměření pro jazyk C++, který nabízí více funkcionality na rozdíl od jazyku C. Jako může být využití objektů nebo prvků, které mají automatickou správu paměti. Například při potřebě dynamicky alokovaného seznamu je v jazyku C potřeba řešit alokaci paměti při vytváření nebo realokaci při přidání nových prvků, obecně tedy správu paměti pro tento seznam musí programátor dělat ručně. V jazyku C++ lze využít vektoru[1]. Stejně tak tato výhoda nastává při práci s řetězci[2] či dalšími prvky. Nevýhodou Arduino frameworku je poté, že je výsledný program mírně větší a zabírá tím více paměti Flash, která by mohla být využita pro jiná data.

Existuje také více variant pro nahrání programu do zařízení a s tím také souvisí editor textu, ve kterém je vhodné program psát. Jednou z možností by bylo program nahrávat ručně, pomocí nástrojů od Espressif[3] a program psát v libovolném editoru. Ale nahrávání programu manuálně může být dost neefektivní, a proto je lepší vybrat si editor, který umožňuje nahrání programu přímo z něj. Osobně jsem se rozhodoval mezi dvěma editory, jedním z nich bylo Arduino IDE a druhý Visual Studio Code s rozšířením PlatformIO.

Arduino IDE je určeno pro vývojové desky Arduino, ale dá se zde doinstalovat podpora i pro vývojovou desku ESP32, poté je potřeba ještě doinstalovat vybraný framework. Arduino IDE obsahuje základní funkcionalitu jako je překladač, programátor a sériový monitor. Dále obsahuje i správu knihoven, kde se dá vyhledat a importovat přímo v IDE knihovnu z internetu, ale kvůli tomu, že je toto IDE určeno převážně pro Arduino, tak se může stát, že zde některá knihovna bude chybět a bude potřeba ji doinstalovat externě. Nevýhodou může být zobrazení otevřených souborů, které jsou vyobrazeny jako záložky, chybí tomu tedy stromová struktura a může to být při větších projektech značně nepřehledné. Dále nepodporuje zobrazení souboru mimo daný projekt, což je také omezující.

Výhodou rozšíření PlartformIO, které je dostupné jako běžný balíček pro VSC (zkratka pro Visual Studio Code) je, že při instalaci platformy ESP automaticky instaluje všechny potřebné vývojové nástroje jako je překladač GCC, programátor a knihovny Espressif. PlatformIO také obsahuje možnost vyhledání a instalace externích knihoven z internetu přímo v IDE. Další výhodou je lepší vyobrazení projektu pomocí stromové struktury, možnost otevření více projektů zároveň nebo otevření externích souborů, které nejsou součástí projektu.

Z výše uvedených důvodů jsem primárně použil VSC s PlatformIO. V průběhu práce jsem ale využil i Arduino IDE, a to z důvodu problému se sériovým monitorem. Tento

[1] https://en.cppreference.com/w/cpp/container/vector

[2] https://en.cppreference.com/w/cpp/string/basic_string

[3] Espressif je společnost, která stojí za vznikem ESP32 nebo také ESP8266.

problém se mi stal po automatickém updatu rozšíření PlatformIO. V rámci odhalování co způsobilo nefunkčnost bylo tedy dobré mít i jinou možnost interakce se zařízením.

5.2 Struktura projektu

PlatformIO má určitou strukturu projektu, od toho se odvíjí ve které složce se nachází určité soubory. Struktura se může mírně lišit podle vybraného frameworku, ale základní struktura zůstává stejná. Projekt je rozdělen do následujících složek [10]:

- Soubor platformio.ini - Slouží ke konfiguraci projektu. Struktura souboru je ve formátu INI[4]. Je zde například nastaven typ frameworku, typ vývojové desky, rychlost sériové komunikace nebo také rozložení paměti Flash.
- Složka include - Pro hlavičkové soubory, které mohou být využívány u více projektů.
- Složka src - Zde je zdrojový kód programu (soubory typu .c, .cpp, .h atd.)
- Složka lib - Slouží k vložení specifických knihoven do projektu.
- Složka test - Slouží pro testy pro daný projekt.
- Složka data - Soubory a složky uložené zde jsou určené pro nahrání do Flash paměti zařízení.

5.3 Webserver

Protože konfigurace zařízení je realizována pomocí webového rozhraní běžícího na řídicí jednotce, je nutné implementovat jednoduchý webový server, který bude schopen komunikace pomocí protokolu HTTP. V rámci frameworku Arduino pro ESP32 je již obsažena knihovna, která tuto funkcionalitu obstarává s názvem WebServer[5]. Další knihovnou, která ale není základně obsažena, je asynchronní verze webserveru s názvem ESPAsyncWebServer[6].

5.3.1 Knihovna ESPAsyncWebServer

Tato knihovna je asynchronní, to znamená, že v jeden daný okamžik může být navázáno více spojení nebo že v průběhu odesílání odpovědi může být již zpracováván další požadavek na rozdíl od první zmíněné knihovny, kde toto nelze. Tato knihovna funguje na principu callback funkcí[7], kde se takto pro typ HTTP požadavku přednastaví obslužná rutina, která se vykoná ve chvíli, kdy přijde daný HTTP požadavek. Velkou výhodou je také propojení se souborovým systémem SPIFFS (viz sekce 5.4) a možnost vložení proměnných do souboru. Toto se provede vložením proměnné do souboru ve tvaru `%VAR%`. V kódu se následně vytvoří funkce vypadající následovně:

```
String processor(const String& var){
  if(var == "VAR"){
    return "value";
```

[4] https://en.wikipedia.org/wiki/INI_file
[5] https://github.com/espressif/arduino-esp32/tree/master/libraries/WebServer
[6] https://github.com/me-no-dev/ESPAsyncWebServer
[7] Callback funkce je funkce předaná jako argument jiné funkci.

```
  }
}
```

Výpis 5.1: Ukázka nahrazení proměnné za hodnotu v souboru pomocí funkce této knihovny.

Tato funkce se využívá při spojení se souborovým systémem, kde v souboru je uložen soubor (například .html) a je v něm potřeba vyobrazit hodnotu určité proměnné. Funkce se následně předá jako argument zároveň s názvem souboru a až přijde daný HTTP požadavek, tak se přečte daný soubor a tyto proměnné budou nahrazeny hodnotami dané funkcí processor() a to bude následně zasláno jako odpověď. Toto velice usnadňuje práci programátorovi, protože nemusí vytvářet vlastní způsob vyobrazení hodnot proměnných na webu.

Nevýhodou je poté, že se při odpovědi na požadavek nemůže využívat funkcí delay() nebo yield(), stejně tak funkcí, které tyto funkce využívají. Obecně tedy nelze po požadavku provést déle trvající operaci, jinak je program zastaven kvůli WatchDog, který kontroluje, že úloha spuštěná pomocí FreeRTOS[8] neblokuje příliš dlouho procesor. Díky tomu nelze například provést scan blízkých BLE zařízení (který může trvat i několik sekund) po obdržení HTTP požadavku a odeslat jako odpověď výsledek tohoto scanu. Z tohoto důvodu jsem musel nahradit tuto knihovnu za synchronní webserver, který tolik neusnadňuje práci. Tuto změnu jsem provedl až v pokročilé fázi programu a přepis programu pro jinou knihovnu stálo dost úsilí.

5.3.2 Knihovna WebServer

Synchronní varianta webserveru neobsahuje propojení se souborovým systémem SPIFFS, tím pádem ani neobsahuje možnost nahrazení proměnných, které jsou potřeba umístit do výsledné stránky a vše toto musí programátor zajistit sám. Další nevýhodou je nemožnost zpracování požadavku, pokud je právě odesílaná odpověď. Vzhledem k tomu, že webserver bude využit pouze pro konfiguraci, ke které bude mít přístup pouze pověřená osoba (osoba s administrátorskou kartou). Tím pádem lze očekávat, že webserver bude v jednu chvíli využívat právě jedna osoba a není tento problém až tak závažný. V souvislosti s tímto problémem, jsem narazil na další problém, který bylo nenačítání CSS stylů, které byly externě uloženy a načteny pomocí:

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

Výpis 5.2: Ukázka načtení CSS stylů z externího souboru.

Tyto styly nebyly načteny, protože ve chvíli, kdy zařízení odesílalo HTML soubor, tak prohlížeč nalezne na začátku odkaz na CSS styly a zašle požadavek na webserver. Protože ale webserver právě odesílá HTML soubor, tak nedokáže požadavek zpracovat a je ignorován. Tento problém řeší právě asynchronní knihovna, kterou ale ze zmíněného problému delších operací nebylo možno použít. Tento problém naštěstí lze vyřešit pomocí inline CSS stylů nebo vložení všech CSS stylů do HTML tagu <head>, což částečně zvětšuje velikost souborů a snižuje čitelnost zdrojového HTML souboru.

5.3.3 Webové rozhraní

Programová část ohledně webserveru je v souboru website.cpp, zde jsou nastaveny podporované požadavky a odeslání odpovědí na tyto požadavky.

[8] FreeRTOS je operační systém pro mikrokontroléry. viz https://www.freertos.org/

HTML stránky jsou poté umístěny ve složce data, která je nahrána do paměti FLASH zařízení. Protože je odesílání požadavků na server řešeno technologií AJAX, tak je součástí HTML souborů část Javascript, která implementuje odesílání a zpracování požadavků na pozadí. Ve složce data jsou následující HTML soubory:

- index.html - Stránka načtená po zadání adresy 192.168.3.1 do prohlížeče, která automaticky přesměruje uživatele na konfiguraci hodnot zařízení (požadavek /config).
- config.html - Stránka s konfigurací hodnot zařízení.
- ble_config.html - Stránka s konfigurací BLE zařízení.
- cards_config.html - Stránka s konfigurací karet Mifare.
- history.html - Stránka s historií průchodu.

5.4 Flash paměť

Ukládání do paměti FLASH je řešeno pomocí souborového systému SPIFFS, který implementuje knihovna[9] obsažená v frameworku Arduino. Jedná se o souborový systém určený pro paměti zařízení s pamětí SPI NOR FLASH. SPIFFS podporuje vyrovnání opotřebení nebo také kontrolu konzistence. V aktuální verzi nepodporuje možnost vytvoření adresářů nebo detekci špatných bloků paměti. [7]

Díky využití souborového systému je možné mít v paměti FLASH uloženy HTML soubory pro webové rozhraní. K nahrání těchto souborů do paměti FLASH zařízení slouží nástroj, který je součástí platformIO. V paměti FLASH jsou také pomocí souborů uložena data, u kterých je potřeba přetrvání i při ztrátě napájení. Pro konfigurační soubory jsem využil formátu CSV [10] a pro historii průchodů textového souboru. Data jsou rozdělena do následujících souborů:

- ble.csv - databáze známých BLE zařízení ve formátu: adresa;hraniční hodnota;uživatelské jméno, kde každý záznam je na novém řádku
- mifare.csv - databáze známých BLE zařízení ve formátu: UID;typ karty;uživatelské jméno, kde každý záznam je na novém řádku
- device_config.csv - konfigurační hodnoty zařízení ve formátu: bzučák, bluetooth, pracovní mód, doba odemknutí zámku po přiložení karty, kde každá hodnota je na novém řádku
- wifi_config.csv - konfigurační hodnoty WiFi ve formátu: jméno, heslo, kde každá hodnota je na novém řádku
- history1.txt a history2.txt - historie průchodů ve formátu: čas záznamu - typ průchodu - uživatelské jméno, kde každý záznam je na novém řádku

Pravděpodobně by bylo možné spojit soubory pro konfiguraci hodnot zařízení a konfiguraci WiFi, ale z důvodu rozdělení těchto konfigurací i na webovém rozhraní jsou rozděleny i zde. Data do souborů je možno zapsat jako znakový řetězec například pomocí funkce printf.

[9] https://github.com/espressif/arduino-esp32/tree/master/libraries/SPIFFS
[10] CSV je zkratka pro comma-separated values.

Nastává zde nevýhoda s interpretací číslic, kde při výpisu pomocí znakového řetězce využijí více paměti.

Ukázka zápis do souboru v paměti FLASH pomocí SPIFFS:

```
File soubor = SPIFFS.open("/jmeno_souboru.txt","w");
soubor.printf("ukazka");
soubor.close()
```

Výpis 5.3: Ukázka zápisu do souboru pomocí souborového systému SPIFFS.

5.5 Karty Mifare

Programová část karet MIFARE je umístěna v souboru mifare.cpp. Pro komunikaci s vybranou čtečkou je využito externí knihovny[11], která je šířena pod UNLICENSE[12]. Čtečka je připojena pomocí sběrnice SPI a dalších pinů (SS a RTS) u kterých je ale potřeba definovat na kterých pinech řídicí jednotky se nachází. Toto se provede inicializací třídy MFRC522, které se předají hodnoty těchto pinů. Přes tuto třídu je poté kontrolováno zda ke čtečce nebyla přiložená nová karta a pokud ano, tak je možné pomocí této třídy přistoupit k datům na kartě.

Pro účel usnadnění práce se známými kartami jsem si vytvořil pomocnou strukturu, která vychází z hodnot uložených v paměti FLASH. Skládá se z UID karty, typu karty a uživatelského jména. Struktura v programu vypadá následovně:

```
struct mifare_card
{
  MFRC522::Uid uid;
  int type;
  String username;
}
```

Výpis 5.4: Pomocná struktura pro Mifare.

V programu jsou známé karty uloženy v paměti FLASH, protože ale přístup do této paměti je pomalý, tak jsou karty zároveň také uloženy v proměnné programu, která obsahuje všechny známé karty a takto se nemusí při každém přiložení karty přistupovat do paměti FLASH.

5.6 Bluetooth Low Energy

Programová část související s BLE je umístěna v souboru ble.cpp. Pro komunikaci s BLE zařízeními bylo využito knihovny BLE[13], která je již součástí frameworku Arduino. Tato knihovna využívá funkcí z frameworku ESP-IDF a vytváří sadu tříd, které mají za účel usnadnit práci s BLE. Ve výsledku zde ale není obsažena tak dobrá dokumentace jako k funkcím z ESP-IDF. Z tohoto důvodu jsem byl nucen některé specifické funkce hledat ve zdrojových kódech knihovny, aby bylo využito převážně této knihovny a nebyl program kombinace této knihovny a funkcí z ESP-IDF. I přes to je v kódu využito přímo funkcí

[11] https://github.com/miguelbalboa/rfid

[12] UNLICENSE je označení pro použití či modifikace bez omezení. Více viz https://unlicense.org

[13] https://github.com/espressif/arduino-esp32/tree/master/libraries/BLE

z ESP-IDF a to u odstranění bondingu zařízení, protože jsem tuto funkcionalitu ve využité knihovně nenašel.

Pro známá BLE zařízení jsem si na základě hodnot uložených ve FLASH paměti vytvořil pomocnou strukturu. Obsahuje hraniční hodnotu síly signálu, adresu zařízení a uživatelské jméno. Struktura v programu vypadá následovně:

```
struct ble_zarizeni
{
  int rsi_limit;
  BLEAddress *adress;
  String username;
}
```

Výpis 5.5: Pomocná struktura pro BLE zařízení.

Stejně jako u karet Mifare, jsou známá zařízení uložena v paměti FLASH a zároveň i v proměnné programu, aby se při ověření zařízení nemuselo přistupovat do paměti FLASH.

5.7 Displej

Programová část související s displejem je umístěna v souboru device_config.cpp. Pro komunikaci s LCD displejem bylo využito externí knihovny LiquidCrystal_I2C [14]. Tato knihovna zajišťuje komunikaci s displejem pomocí sběrnice I2C. Při vytvoření třídy pro tento displej je potřeba předat adresu zařízení na dané sběrnici, počet sloupců a počet řádků. Vložení textu na displej se poté provádí pomocí nastavení kurzoru na dané místo na displeji a zápisu znaku. Tato knihovna podporuje výpis na displej například pomocí funkce printf, pomocí které jsem řešil většinu výpisů. Je zde potřeba dát si pozor na proměnlivou délku vypsaného řetězce. Pokud je například na displeji vypsáno 8 znaků a při příštím výpisu pouze 5, tak je nahrazeno prvních 5 znaků těmito novými a poslední 3 znaky zde zůstanu z prvního výpisu. Pokud zde nemají být zobrazeny znaky z předchozího výpisu je potřeba rozšířit druhý řetězec například o mezery, které jsou vypsány jako prázdný znak.

Vybraný displej podporuje vložení až 7 vlastních znaků. Například takto vypadá vložení znaku zámku (první znak na obrázku 4.3) a jeho zobrazení na displeji.

```
byte custom_char_lock[] = {
  B01110,
  B10001,
  B10001,
  B11111,
  B11111,
  B11011,
  B11011,
  B11111
};
lcd.createChar(0, custom_char_lock);
lcd.write(0);
```

Výpis 5.6: Ukázka přidání vlastního znaku na displej.

[14] https://github.com/johnrickman/LiquidCrystal_I2C

5.8 Bzučák

Programová část související s bzučákem je umístěna v souboru device_config.cpp. Protože byl využit bzučák bez vnitřního generátoru, tak je potřeba generovat obdélníkový signál se střídou 1:1 na zvolené frekvenci. Jako frekvenci jsem zvolil hodnotu 4 kHz, z důvodu nejvyšší hlasitosti právě na této dané frekvenci, tuto frekvenci lze nelézt v dokumentaci [5] k vybranému bzučáku. Pípnutí bzučáku je poté zajištěno přivedením tohoto obdélníkové signálu na vybraný pin na vybranou dobu, v mém případě jsem vybral pin 27 a dobu 100 ms (tato hodnota značí jak dlouho trvá pípnutí bzučáku). V programu je poté potřeba přiřadit vybranou frekvenci a rozlišení k PWM kanálu a tento kanál přiřadit k vybranému pinu, kanál je možno zvolit z rozmezí 0-15. Ze zvoleného rozlišení se následně určuje střída, toto rozlišení se určuje počtem bitů, které může být v rozmezí 1-16 bitů. V programu jsem použil rozlišení 8 bitů, při přivedení obdélníkového signálu na vybraný kanál je potřeba zvolit střídu, pro střídu 1:1 je tato střída 128. Rozlišení se udává v bitech, tím pádem $2^8 = 256$ a pro zvolení střídy 1:1 je potřeba polovina tohoto rozlišení $\frac{256}{2} = 128$. Bylo by ale možné využít jakéhokoliv rozlišení s následnou poloviční hodnotou tohoto rozlišení.

5.9 Historie průchodů

Programová část související s ukládáním a čtením historie průchodu je umístěn v souboru history.cpp. Vzhledem k tomu, že je pro ukládání do paměti FLASH využito souborového systému SPIFFS, tak mám záznamy uložené pomocí dvou textových souborů uložených v paměti FLASH, kde každý má maximální kapacitu 25 kB, pro celkovou kapacitu 50 kB. První záznamy jsou ukládány do prvního souboru, po překročení maximální kapacity jsou záznamy ukládány do druhého souboru. Po překročení kapacity druhého souboru (tím i celkové kapacity) je první soubor odstraněn a na jeho místo je umístěn druhý soubor. Na místo druhého souboru je umístěn soubor nový, kde pokračuje ukládání historie průchodů, tento proces se opakuje s každým přesáhnutím kapacity. Aby docházelo k minimálnímu počtu přepisů paměti FLASH, tak je přesun souboru řešen pouhým přejmenováním. Další snížení počtu zápisů do paměti FLASH je řešeno pomocí dočasného úložiště v proměnné v programu, kde je nastavena kapacita pro 100B. Nové záznamy jsou prvně akumulovány v této proměnné a po přesáhnutí kapacity jsou zapsány jako celek do paměti FLASH. Nevýhodou tohoto přístupu je možnost přijít o data v této proměnné při výpadku napájení, proto je kapacita této proměnné nastavena na nízkou hodnotu aby byla v tomto případně ztracena pouze minimální část dat. Velikost záznamu se liší podle velikosti uživatelského jména, pokud by se uvažovalo o průměrném záznamu 30 B, tak v případě výpadku napájení budou ztraceny maximálně poslední 3 záznamy. Maximální kapacita je tedy 50 kB v paměti FLASH a 100 B v této proměnné, při průměrném záznamu 30 B je maximálně uloženo 1670 záznamů o průchodu.

5.10 Ověření funkčnosti

Ověření funkčnosti probíhalo manuálně pomocí série testů, kde jsem se snažil pokrýt veškerou funkcionalitu zařízení. Pro ověření funkčnosti přístupu bylo využito dvou karet Mifare a fitness náramku SAS82. Pro simulaci zámku na dveřích jsem využil svítivé diody (LED), aby šlo snadno poznat stav zámku. Konfigurační rozhraní bylo zobrazeno pomocí notebooku

s operačním systémem Windows. Ověření funkčnosti se poté lišilo podle zvolené konfigurace na webovém rozhraní.

Normální mód

- Možnost přístupu pomocí přítomnosti BLE, kde po umístění fitness náramku v dostatečné vzdálenosti byl zámek odemčen a po umístění fitness náramku mimo tuto vzdálenost byl zámek zpět uzamčen.
- Jestli je při vypnutí možnosti přístupu pomocí BLE zámek uzamčen i když je známé zařízení v dosahu.
- Zda je zámek odemčen na specifikovanou dobu (specifikovanou v konfiguračním rozhraní) po přiložení běžné karty.
- Zda bzučák pípne pouze při první detekci fitness náramku a neopakuje se pípnutí dokud je fitness náramek v dosahu nebo zda bzučák pípne po přiložení karty.
- Jestli je bzučák neaktivní po jeho vypnutí na webovém rozhraní.

Trigger mód

- Nemožnost přístupu pomocí přítomnosti BLE zařízení.
- Zda je zámek trvale odemčen po přiložení běžné karty a po opakovaném přiložení je trvale uzamčen.
- Zda bzučák pípne po přiložení karty.
- Jestli je bzučák neaktivní po jeho vypnutí na webovém rozhraní.

Konfigurační rozhraní

- Možnost přepnutí pracovního módu, nastavení doby po kterou je zámek odemčen, možnost vypnutí/zapnutí detekce BLE a zvukové signalizace.
- Možnost změny jména a hesla WiFi a následné otestování připojení a konfigurace.
- Zda lze zaregistrovat novou kartu a otestování její funkčnosti.
- Zda lze známou kartu odstranit a otestování její nefunkčnosti.
- Zda lze známou kartu upravit/odstranit v tabulce, kde jsou vypsány známé karty a otestování upravených hodnot.
- Změna administrátorské karty na běžnou a naopak a jejich následné otestování správného rozlišení.
- Přidání nového BLE zařízení a otestování jeho funkčnosti.
- Odstranění známého BLE zařízení a otestování jeho nefunkčnosti.
- Zda lze známé BLE zařízení upravit/odstranit v tabulce, známých BLE zařízení a otestování upravených hodnot.

- Zda je správná historie průchodů.
- Zda nejsou data nastavená v konfiguračním rozhraní ztracena při výpadku napájení.

Displej

- Zda je na displeji vypsán aktuální čas.
- Zda jsou po změně v konfiguračním rozhraní upravena data na displeji.
- Zda vypadají vlastní znaky, tak jak byly navrženy.
- Zda je uživatelské jméno vypsáno správně nezávisle na jeho délce.

Z důvodu charakteru těchto testů je bylo potřeba provést manuálně, protože automatické testování není možné (nebo by to bylo velice obtížné) provést. Převážně z důvodu, že se jedná o hardwarové zařízení kde je potřeba testovat interakci s jinými prvky jako jsou přístupové karty nebo fitness náramek. Nebo testovat zda jsou na displeji zobrazeny správné údaje nebo zda funguje zvuková signalizace.

Pro ověření funkčnosti bylo zařízení sestaveno formou prototypu podle tabulky A.1, která se nachází v příloze. V průběhu implementace bylo zařízení testováno za účelem odstranění chyb. Výsledné zařízení splňuje tyto testy a jeho chování odpovídá návrhu.

5.11 Bezpečnost

Bezpečnost zařízení se odvíjí od vybraných možností přístupu. Pro vybrané karty Mifare Classic je systém bezpečný do chvíle, kdy někomu není odcizena přístupová karta. Odcizenou kartu lze zneužít pro přístup, kde není možné odlišit která osoba danou kartu využije. Odcizenou kartu by bylo možné také zkopírovat, ale k tomuto je potřeba speciálních nástrojů. Zkopírování karty by šlo zabránit využitím dražších karet Mifare Desfire, ale nelze zabránit zneužití odcizené karty. Zde by bezpečnost šla navýšit například pomocí kamery u přístupu a šlo by takto zjistit, která osoba využila neoprávněně přístupu.

U přístupu pomocí BLE je využíváno bondingu, kde si zařízení při prvním spojení uloží použité klíče a tyto klíče jsou použity při následujících spojení. Při pokusu o podvržení BLE zařízení, tak toto zařízení nebude obsahovat tyto uložené klíče a nemělo by být možné jej podvrhnout. Stejně jako u přístupových karet nelze zamezit případnému odcizení zařízení, které pak může být zneužito pro přístup.

Pro přístup do konfiguračního rozhraní je využito administrátorské karty a také hesla u spuštěné WiFi. Díky tomu je nemožné dostat se na konfigurační rozhraní pouze odcizením administrátorské karty, protože je zde ještě potřebné heslo. Stejně tak nelze například podniknout útok na WiFi, protože je mimo stav konfigurace vypnuta.

Pro zvýšení bezpečnosti by šlo využít identifikace pomocí otisku prstu nebo scanu oční duhovky. Kde je velice obtížné identifikaci podvrhnout nebo provést případné odcizení.

Kapitola 6

Závěr

Cílem této práce bylo vytvoření řídicí jednotky přístupového systému podporující více možností identifikace. Jako první bylo potřeba seznámit se s obecným řešením přístupových systémů, různými možnostmi identifikace uživatele, různými mikrokontroléry nebo také možnostmi realizace konfiguračního rozhraní.

Výsledné zařízení bylo poté realizováno formou prototypu s využitím vývojové desky ESP32. Zařízení podporuje identifikaci uživatele pomocí karet Mifare a detekce Bluetooth Low Energy zařízení. Pro účely konfigurace bylo využito webového rozhraní, díky kterému je konfigurace snadná pro použití uživatelem. Zařízení také obsahuje displej a zvukovou signalizaci pomocí bzučáku, díky čemuž uživatel snadno rozpozná stav systému. Dále zařízení ukládá do paměti záznamy o průchodu, které lze využít například ke kontrole průchodů. Přínosem této práce je usnadnění identifikace uživatele pomocí technologie Bluetooth Low Energy, která je v poslední době stále více používána. Výsledné zařízení splnilo vlastnosti popsané v návrhu, které byly otestované pomocí sady testů popsané v poslední kapitole. V této kapitole byla také diskutována bezpečnost výsledného zařízení. Tato práce mi dala nové zkušenosti v oblasti návrhu vestavěných systémů, převážně související s vývojovou deskou ESP32.

Jako rozšíření tohoto zařízení by mohlo být více možností přístupu, jako je například otisk prstu, scan oční duhovky nebo klávesnice. Zatímco otisk prstu a scan oční duhovky by přispěli na bezpečnosti zařízení, tak klávesnice by pravděpodobně snížila bezpečnost zařízení. Dalším rozšířením by mohlo být odesílání záznamů o průchodu po síti na server, kde by následně mohl být docházkový systém.

Literatura

[1] Almeida, M. Hacking Mifare Classic Cards [online]. 2014 [cit. 2020-04-20]. Dostupné z: https://www.blackhat.com/docs/sp-14/materials/arsenal/sp-14-Almeida-Hacking-MIFARE-Classic-Cards-Slides.pdf.

[2] Arduino. ARDUINO MEGA 2560 REV3 [online]. 2020 [cit. 2020-04-16]. Dostupné z: https://store.arduino.cc/arduino-mega-2560-rev3.

[3] Arduino. ARDUINO UNO REV3 [online]. 2020 [cit. 2020-04-16]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3.

[4] Aversente, F. Native, Hybrid and Progressive Web Applications — Building a Mobile App Today [online]. 2019 [cit. 2020-05-16]. Dostupné z: https://medium.com/twodigits/native-hybrid-and-progressive-web-applications-building-a-mobile-app-today-db076642eb40.

[5] BESTAR ELECTRONICS INDUSTRY. PIEZO BUZZER [online]. 2001 [cit. 2020-05-17]. Dostupné z: https://www.tme.eu/Document/55fc62c1868e41e3e831a2ff777f44f1/bpt-14.pdf.

[6] Bon, M. A Basic Introduction to BLE Security [online]. 2016 [cit. 2020-04-11]. Dostupné z: https://www.digikey.com/eewiki/display/Wireless/A+Basic+Introduction+to+BLE+Security.

[7] Espressif Systems. SPIFFS Filesystem [online]. 2020 [cit. 2020-05-20]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/spiffs.html.

[8] Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L. et al. Hypertext Transfer Protocol – HTTP/1.1 [Internet Requests for Comments]. RFC 2616. RFC Editor, 1999 [cit. 2020-05-21]. Dostupné z: http://www.rfc-editor.org/rfc/rfc2616.txt.

[9] George, D. P. a Sokolovsky, P. MicroPython language and implementation [online]. 2020 [cit. 2020-05-15]. Dostupné z: https://docs.micropython.org/en/latest/reference/index.html.

[10] Kravets, I. Platformio project init [online]. 2020 [cit. 2020-05-15]. Dostupné z: https://docs.platformio.org/en/latest/core/userguide/project/cmd_init.html.

[11] Maxim Integrated. DS3231 [online]. 2015 [cit. 2020-04-20]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf.

[12] NXP Semiconductors. I2C-bus specification and user manual [online]. 2014 [cit. 2020-05-07]. Dostupné z: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[13] NXP Semiconductors. MFRC522 [online]. 2016 [cit. 2020-04-20]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

[14] NXP Semiconductors. MIFARE Classic EV1 1K - Mainstream contactless smart card IC for fast and easy solution development [online]. 2018 [cit. 2020-04-20]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MF1S50YYX_V1.pdf.

[15] Rahul, A., G, G. K., H, U. K. a Rao, S. Near Field Communication (NFC) Technology: A Survey. International Journal on Cybernetics & Informatics. 2015, sv. 4, č. 2, s. 133–144, [cit. 2020-04-08]. DOI: 10.5121/ijci.2015.4213.

[16] Refsnes Data. AJAX Introduction [online]. 2019 [cit. 2020-05-16]. Dostupné z: https://www.w3schools.com/js/js_ajax_intro.asp.

[17] RF Wireless World. BLE States | Standby, Advertising, Scanning, Initiating, connection, Synchronization [online]. 2012 [cit. 2020-04-11]. Dostupné z: https://www.rfwireless-world.com/Terminology/BLE-States-and-State-Diagram.html.

[18] Seeed Technology. GeeekNET ESP32 Development Board [online]. 2020 [cit. 2020-04-16]. Dostupné z: https://www.seeedstudio.com/GeeekNET-ESP32-Development-Board-p-2945.html.

[19] Texas Instruments. Keystone Architecture Serial Peripheral Interface (SPI) [online]. 2012 [cit. 2020-05-07]. Dostupné z: http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf?&ts=1590159406050.

[20] Townsend, K., Cufí, C., Akiba a Davidson, R. Getting Started with Bluetooth Low Energy [online]. 2014 [cit. 2020-04-11]. Dostupné z: https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html.

[21] Weis, S. A. RFID (Radio Frequency Identification): Principles and Applications [online]. 2007 [cit. 2020-04-08]. Dostupné z: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.5224&rep=rep1&type=pdf.

Příloha A

Zapojení zařízení

ESP32 piny	Periferie
GND	GND displeje, RTC modulu, čtečky karet a bzučáku
3V3	Vcc displeje, RTC modulu a čtečky karet
GPIO18	SCLK čtečky karet
GPIO19	MISO čtečky karet
GPIO21	SDA displeje a RTC modulu
GPIO22	SCL displeje a RTC modulu
GPIO23	MOSI čtečky karet
GPIO25	output pin
GPIO27	pin bzučáku
GPIO32	SS čtečky karet
GPIO33	RST čtečky karet

Tabulka A.1: Propojení pinů ESP32 s externími periferiemi.