VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

MODULÁRNÍ PŘÍSTUPOVÝ SYSTÉM PRO PROVOZ SAUNY

MODULAR ACCESS SYSTEM FOR SAUNA FACILITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAEL KINC**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÁCLAV ŠIMEK,**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Kinc Michael**
Program: Informační technologie
Název: **Modulární přístupový systém pro provoz sauny**
Modular Access System for Sauna Facility
Kategorie: Vestavěné systémy
Zadání:

1. Zabývejte se technologiemi bezdrátového přenosu dat, přičemž se zaměřte zejména na standard RFID a WiFi.
2. Seznamte se s moduly na bázi mikrokontroléru ESP32 a platformou Raspberry Pi. Dále připravte stručnou rešerši existujících řešení přístupových systémů s modulární architekturou.
3. Na základě poznatků z bodů 1 a 2 navrhněte architekturu vlastního řešení modulárního přístupového systému pro provoz sauny.
4. Zvolte vhodné komponenty a na obvodové úrovni proveďte technickou realizaci potřebných funkčních modulů dle navržené architektury.
5. Vytvořte obslužný firmware pro jednotlivé typy modulů zajišťujících otevírání dveří přístupovým čipem a načtení klientských karet zákazníků.
6. Implementujte uživatelské rozhraní pro ovládání přístupového systému na bázi webové aplikace běžící v okně prohlížeče a taktéž jednoduchý informační systém.
7. Celé řešení otestujte a ověřte funkčnost v reálném provozu.
8. Vyhodnoťte výsledky a navrhněte, jak by se případně systém dal v budoucnu rozšířit.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 4 zadání.

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

Vedoucí práce: **Šimek Václav, Ing.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Přístupové systémy jsou velmi důležitým mechanismem pro kontrolu pohybu osob po prostorách jakéhokoliv objektu či areálu. Cílem této práce je návrh vlastního modulárního přístupového systému pro zařízení sauny, které zahrnuje chytré zámkové moduly pro otevírání dveří či šatních skříněk, a modul pro čtení klientských karet. Celý systém je přitom možné kontrolovat vzdáleně prostředím webové aplikace s informačním systémem.

Abstract

Access systems are very important mechanism for controlling the movement of people in building. The aim of this work is to design own modular access system for sauna facility, which includes smart modules for opening doors or wardrobes, and the module for reading client cards. The entire system can be controlled remotely via web application with the information system.

Klíčová slova

přístupový systém, mikrokontroklér, ESP32, návrh DPS, informační systém, PHP, Laravel, HTML, TailwindCSS

Keywords

access system, microcontroller, ESP32, PCB layout, information system, PHP, Laravel, HTML, TailwindCSS

Citace

KINC, Michael. *Modulární přístupový systém pro provoz sauny.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek,

Modulární přístupový systém pro provoz sauny

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Václava Šimka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

. . . . . .
Michael Kinc
10. května 2021

Poděkování

Rád bych poděkoval mému vedoucímu panu Ing. Václavu Šimkovi za odborné konzultace při realizace této bakalářské práce. Dále bych chtěl velice poděkovat mému kamarádovi Ing. Viktoru Kovaříkovi za praktické rady zejména v oblasti návrhu desek plošných spojů. Velké poděkování také patří mojí rodině za velkou podporu a zázemí při studiu a tvorbě této práce. Děkuji též mému kamarádovi Marku Šťastnému za podporu a příjemné diskuze.

Obsah

Kapitola 1

Úvod

Přístupový systém je velmi důležitým zabezpečovacím mechanismem v každé provozovně. Pomocí něj je možné kontrolovat nebo omezovat pohyb osob po areálu. V případě saun či wellness center takový systém může zahrnovat elektronické čipové RFID zámky do skříněk a dveří v celé provozovně, které jsou zároveň napojeny na informační systém, přičemž pracovník na recepci může celou provozovnu sledovat. Takový systém je však často velmi nákladný a mnohdy neodpovídá konkrétním požadavkům provozovatele.

Zavedení takového systému ulehčuje život jak majiteli, tak klientům. Majitel při případné ztrátě klíče nemusí měnit celý zámek a kupovat nový klíč, ale stačí pouze v informačním systému přiřadit dané skříňce nový RFID náramek. Klient se také při svém pobytu v provozovně nemusí starat o klíč ke své skříňce, ale RFID náramek má po celou dobu na svém zápěstí.

Práce je rozdělena na několik částí. V kapitole 2 je popsán způsob bezdrátového přenosu dat pomocí standardů Wi-Fi, Bluetooth a Zigbee. Také je zde popsán způsob bezkontaktní identifikace pomocí RFID a způsob výměny dat mezi klientem a serverem prostřednictvím síťových komunikačních protokolů. Dále je čtenář v kapitole 3 seznámen s architekturou a sběrnicemi mikrokontrolérů, zejména jsou zde popsány mikrokontroléry ESP32 a platforma Raspberry Pi. Kapitola 4 pak obsahuje rešerši existujících přístupových systémů a návrh vlastního řešení a v kapitole 5 je popsán způsob implementace tohoto navrženého řešení. Kapitola 6 pak obsahuje způsob testování tohoto systému, jeho výsledky a popisuje možnosti budoucího rozvoje.

Cílem této práce je pro existující provozovnu sauny v Brně na míru vytvořit kompletní řešení chytrých zámkových modulů do skříněk a přístupových dveří, dále modul pro čtení klientských karet na recepci a informační systém v podobě webové aplikace, kde je možné celý systém spravovat. Byl kladen důraz především na cenu a specifické požadavky provozovny, jimiž jsou zejména elektronická správa permanentek, řízení přístupu a pohybu osob v provozovně a požadavek na vzdálenou kontrolu celé provozovny skrze informační systém a webovou aplikaci.

Kapitola 2

Bezdrátová komunikace

Bezdrátové technologie jsou dnes všudypřítomné a nabízejí obrovské množství funkcí, přičemž jejich dostupnost se zvyšuje díky investicím do pevné infrastruktury a také kvůli snižující se ceně a velikosti zařízení.

Tato kapitola popisuje několik základních, často využívaných standardů pro bezdrátovou komunikaci. Zaměřuje se především na Wi-Fi a RFID, které jsou pro tuto práci podstatné. V závěru jsou pak shrnuty důležité protokoly pro komunikaci po síti.

2.1 Wi-Fi

Bezdrátové sítě LAN[1] jsou každodenní součástí našich životů. Kolem roku 1990 se začalo objevovat mnoho technologií pro bezdrátové sítě LAN, z nichž tou jednoznačně nejdůležitější je standard IEEE 802.11, širší veřejnosti známý jako Wi-Fi [20], který byl ratifikován organizací IEEE[2] v roce 1997 a ve své první verzi obsahoval maximální přenosovou rychlost 2 Mbit/s. Původní verze tohoto standardu definuje základní pravidla pro bezdrátové služby.

Postupem času bylo vytvořeno několik modulací původního standardu 802.11 obsahující rozdíly především na vrstvě fyzického rozhraní. Zařízení využívající některý z protokolů 802.11 pracují se dvěma základními frekvenčními rozsahy – 2,4 GHz a 5 GHz. První jmenovaný rozsah je nelicencované frekvenční pásmo a zařízení 802.11 v něm mohou soupeřit například s telefony nebo mikrovlnnými troubami. Mezi těmito různými zařízeními může docházet k interferenci. Bezdrátové sítě LAN pracující v rozsahu 5 GHz mají kratší přenosovou rychlost a jsou náchylnější k vícecestnému šíření. Jednotlivé standardy, jejich maximální přenosové rychlosti a frekvenční pásma popisuje tabulka 2.1.

Hojně používané jsou v dnešní době standardy 802.11n a 802.11ac, které využívají antény s více vstupy a více výstupy (MIMO), to znamená dvě a více antén na vysílající straně a dvě a více antén na straně přijímající. Takové řešení umožňuje využívat vysoké přenosové rychlosti. U standardu 802.11ac potom dochází k vylepšení v podobě MU–MIMO (Multi–User MIMO), které dovoluje přenášet data různým klientům současně díky chytrému systému front.

Nejnovějším standardem je 802.11ax, nebo-li Wi-Fi 6. Jedná se o vylepšeného nástupce standardu 802.11ac. Tento nový standard poskytuje velké zvýšení maximální přenosové rychlosti oproti předchozí generaci. [20, 21, 26]

[1] LAN – Local Area Network (místní síť – označuje počítačovou síť pokrývající malé geografické území např. domácnost)

[2] IEEE – Institute of Electrical and Electronics Engineers (Institut pro elektrotechnické a elektronické inženýrství)

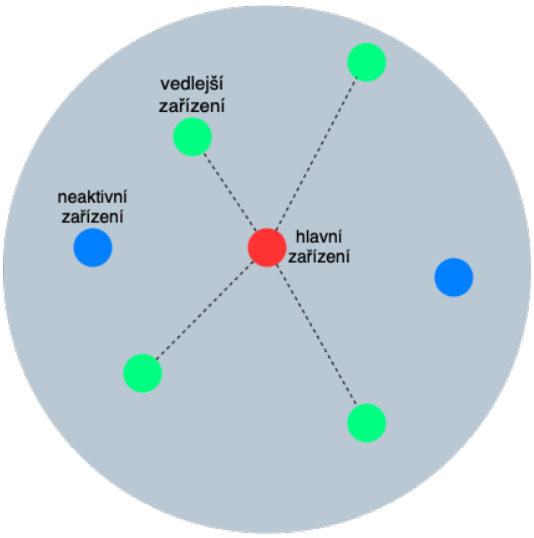
Standard	Označení	Pásmo [GHz]	Maximální přenosová rychlost [Mbit/s]
původní IEEE 802.11	–	2,4–50	2
IEEE 802.11a	Wi-Fi 1	5	54
IEEE 802.11b	Wi-Fi 2	2,4	11
IEEE 802.11g	Wi-Fi 3	2,4	54
IEEE 802.11n	Wi-Fi 4	2,4/5	600
IEEE 802.11ac	Wi-Fi 5	5	6930
IEEE 802.11ax	Wi-Fi 6	2,4/5/6	10530

Tabulka 2.1: Přehled standardů IEEE 802.11 [20, 21, 26]

2.2 Bluetooth

Bluetooth je technologie umožňující bezdrátovou komunikaci na krátkou vzdálenost mezi několika zažízeními jako jsou například mobilní telefon, osobní počítač nebo bezdrátová sluchátka. Bylo vytvořeno švédskou společností Ericsson Mobile Communications roku 1994 a je definováno normou IEEE 802.15.1 [20].

Bluetooth operuje ve stejném frekvenčním pásmu jako technologie Wi-fi a to ve 2,4 GHz. Zařízení připojená přes Bluetooth se organizují sama, není tedy nutný žádný přístupový bod. Tato zařízení jsou organizována do takzvaného piconetu, což je počítačová síť, ve které je organizováno až osm zařízení, z nichž jedno je hlavní a ostatní vedlejší. V této síti také může být až 255 neaktivních zařízení, které čekají na pokyn od hlavního zařízení, které je změní na vedlejší a zahájí s nimi komunikaci [20].



Obrázek 2.1: Organizace zařízení do piconetu při použití technologie Bluetooth. Obrázek inspirován [20]

Součástí nové specifikace Bluetooth 4.0 představené v roce 2010 je také Bluetooth LE (Low Energy). Hlavním vylepšením je extrémně nízká spotřeba a větší dosah. Je tedy určena na výměnu kratších informací nižší rychlostí a takto připojená zařízení vydrží být napájena baterií velmi dlouhou dobu [28].

2.3 Zigbee

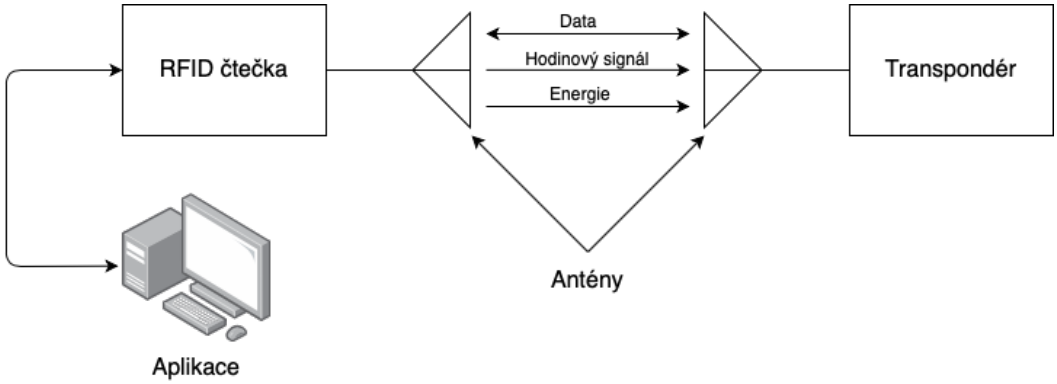
Zigbee je technologie, podobně jako Bluetooth, umožňující spojení zařízení v osobní síti na malé vzdálenosti a je definováno normou IEEE 802.14.5. Zigbee disponuje nižším výkonem a nižší datovou rychlostí, a tak se zaměřuje především na komunikaci zařízení jako jsou například světelná a teplotní čidla a nástěnné spínače. Pro tato zařízení jsou specifikace technologie Zigbee vhodné a snižují tak ekonomické a energetické náklady [20, 21].

2.4 RFID

RFID (Radio Frequency Identification) je bezdrátová komunikační technologie sloužící k jednoznačné identifikaci označených osob, zvířat a věcí. V dnešní době se využívá v mnoha oblastech lidské působnosti. Příkladem mohou být systémy pro kontrolu přístupu, bezklíčový vstup a identifikace zaměstnanců, zařízení pro sledování a správu hospodářských zvířat, identifikace služebních vozidel nebo pásky na zápěstí či kotníku kojenců sloužící k jejich identifikaci [19].

Systém RFID využívá k identifikaci technologii bezdrátové rádiové komunikace a skládá se ze tří základních komponentů, kterými jsou [19]:

1. transpondér (nebo také štítek) skládající se z polovodičového čipu a antény,
2. zařízení pro čtení a zápis, které je složeno z antény, modulu radiofrekvenční elektroniky a modulu řídící jednotky,
3. kontrolér (nebo také hostitel), pod čímž si můžeme představit počítač, na kterém běží databáze a základní ovládací software.



Obrázek 2.2: Základní stavební bloky RFID systému. Obrázek inspirován [17].

2.4.1 RFID transpondér

Transpondérem mohou být karty, nálepky, pásky a jiné. Jejich základní funkcí je ukládání a přenos dat do čtecího zařízení. Z hlediska fungování dodávek energie můžeme transpondéry rozlišovat na aktivní a pasivní.

Pasivnímu transpondéru je energie pro jeho provoz poskytována prostřednictvím magnetického a elektromagnetického pole čtečky. Toto pole je poskytováno prostřednictvím antény transpondéru. Jednoduše řečeno je energie emitovaná čtečkou použita k přenosu

dat ze čtečky do transpondéru a zpět do čtečky. Pokud je transpondér umístěn mimo dosah čtečky, nemá žádná napájení, a tak nebude vysílat žádné signály.

Aktivní transpondér má svůj zdroj energie, kterým může být například baterie. Takový zdroj se používá k napájení čipu a transpondér tedy nepotřebuje magnetické nebo elektromagnetické pole přijímané čtečkou. U obou typů transpondéru je však toto pole nezbytné pro přenos dat. Aktivní transpondéry mají podstatně vyšší komunikační rozsah. [17]

Obrázek 2.3: RFID transpondéry

2.4.2 Čtecí zařízení

Čtecí zařízení v podstatě funguje jako bridge mezi RFID transpondérem a počítačem. Má několik základních funkcí. Jak už její název napovídá, její nejdůležitější funkcí je čtení dat z transpondéru. V případě použití pasivního transpondéru slouží jako jeho zdroj napájení. Dále se také stará o přenost dat do počítače a z něj [19].

Příkladem může být čtecí zařízení MFRC522, které je používáno v této práci a podrobněji popsáno v kapitole 4.3.2.

2.4.3 MIFARE

MIFARE je obchodní značka společnosti NXP Semiconductors, která zahrnuje integrované obvody použité zejména v bezkontaktních kartách. Produkty této značky splňují mezinárodní standard ISO/IEC 14443, který je uplatňován ve většině dnes používaných bezkontaktních kartách.

Každý MIFARE transpondér obsahuje 4 nebo 7 bytové UID[3], které jednoznačně identifikuje danou kartu nebo čip. To umožňuje v systémech ověřovat totožnost. [1]

2.5 Komunikační protokoly

Protokol můžeme definovat jako soubor syntaktických a sémantických pravidel určující výměnu informace mezi alespoň dvěma koncovými body. Protokol může specifikovat mnoho vlastností jako je navázání spojení, adresování, přenos dat a zpracování chyb. [25]

[3] UID – Unique identifier (jednoznačný identifikátor)

Při komunikaci po síti je zpráva se získanými daty před odesláním k příjemci zabalena do ethernetového rámce nebo může být využita některá z bezdrátových technologií. Data, která mají být posílána po síti jsou ovšem potřeba zabalit do komunikačních protokolů. Následující část popisuje ty nejdůležitější z nich.

2.5.1 TCP

TCP (Transmission Control Protocol) je protokol zajišťující spolehlivý přenos dat. Načítá data z aplikační vrstvy jako proud dat a seskupené do jednotlivých paketů je posílá po síti k cíli. Protokol TCP je spojově orientovaný, protože před samotným doručováním dat musí mezi oběma stranami proběhnout takzvaný handshaking, což znamená, že si obě strany musí stanovit parametry přenosu dat. TCP spojení je plně duplexní, což znamená, že oba koncové uzly si mohou zprávy zasílat současně [20].

2.5.2 HTTP

HTTP (HyperText Transfer Protocol) je internetový protokol, který je určen ke komunikaci s webovými servery. Tento protokol je implementován v klientském i serverovém programu. Definuje způsob, jakým klienti žádají o data z webových serverů nebo jak data na tyto servery posílají.

Při této komunikaci je využíván transportní protokol TCP. Klient tedy nejprve inicializuje TCP spojení se serverem, poté odesílá zprávy s HTTP požadavky a přijímá HTTP zprávy s odpověďmi. Podobně tento princip probíhá na straně serveru [20].

Formát zprávy HTTP

Zprávy HTTP jsou psány běžným textem a dají se rozdělit na dva druhy. Prvním typem jsou zprávy požadavku, které klient odesílá na server. Tato zpráva může mít několik řádků, přičemž spodní hranice je jeden řádek. První řádek zprávy požadavku se nazývá řádek požadavku a obsahuje tři informace: metodu, URL[4] adresu, verzi HTTP. Nejpoužívanějšími metodami jsou `GET`, `POST`, `PUT` a `DELETE`. Metoda `GET` se používá v případě, že klient žádá po serveru data. Metoda `POST` se používá tehdy, když klient potřebuje serveru předat informace například z webového formuláře ke zpracování. Metoda `PUT` se používá v případě, kdy klient potřebuje nahrát objekty na server. Odstranění těchto objektů se pak provádí metodou `DELETE` [20].

Následující řádky zprávy požadavku se nazývají řádky záhlaví. Ty obsahují modifikátory a metainformace obsahu. Příkladem může být požadavek na kódování, autentizaci nebo určení toho, že spojení po dokončení požadavku zůstane otevřené. Tyto řádky jsou udávány názvem položky a hodnotou oddělené dvojtečkou. Každý řádek je ukončen symbolem prázdného řádku, který reprezentuje dvojice znaků `\r\n`. Za řádky záhlaví volitelně následuje tělo zprávy, které je odděleno prázdným řádkem, tedy sekvencí `\r\n\r\n`. Na obrázku 2.1 se nachází příklad HTTP zprávy požadavku [20].

```
GET / HTTP/1.1\r\n
Host: example.org\r\n
Accept-Language: cs-cz\r\n
```

[4] URL – Uniform Resource Locator (jednotná adresa zdroje – slouží ke specifikaci umístění zdrojů informací na internetu)

```
Connection: keep-alive\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: com.apple.trustd/2.0\r\n
\r\n
```

Výpis 2.1: Příklad HTTP zprávy požadavku

Druhým typem jsou zprávy odpovědi, které server zasílá klientovi v návaznosti na jeho HTTP požadavek. Taková zpráva obsahuje tři části. První řádek se nazývá stavový a obsahuje pole s verzí protokolu, stavový kód a tomu příslušící slovní popis. Za stavovým řádkem následují řádky záhlaví. Ty obsahují metainformace těla zprávy, které následuje za řádky záhlaví. Na obrázku 2.2 můžeme vidět odpověď na předchozí požadavek uvedený na obrázku 2.1 [20].

```
HTTP/1.1 200 OK\r\n
Server: nginx\r\n
Content-Type: text\html\r\n
Content-Length: 503\r\n
Last-Modified: Thu, 22 Apr 2021 22:00:00 UTC\r\n
Cache-Control: public, no-transform, must-revalidate, max-age=29468\r\n
Connection: keep-alive\r\n
\r\n
<!doctype html>\n
<html>\n
```

Výpis 2.2: Příklad HTTP zprávy s odpovědí

REST

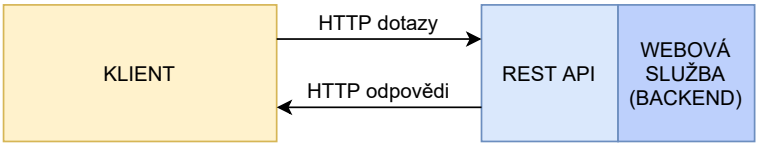
REST (*Representational state transfer*) označuje architekturu rozhraní pro distribuované systémy, kterou v roce 2000 navrhl Roy Fielding ve své disertační práci.

Klientské aplikace komunikují se servery prostřednictvím rozhraní zvaného API[5]. Jedná se o sadu funkcí, díky které je výrazně zjednodušena interakce mezi dvěma programy. Architektura REST je velice často používána při návrhu API. Takové API, které odpovídá architektuře REST se nazývá REST API. Webová služba, která obsahuje REST API, se nazývá *RESTful*. Aby mohlo být rozhraní považováno za *RESTful*, musí splňovat pět podmínek [24]:

- **Klient-server** – Jedná se o architekturu, která odděluje uživatelské rozhraní od serverové logiky. Tyto části mohou být nasazeny nezávisle na sobě a to za použití libovolného programovacího jazyka nebo technologie.
- **Jednotné rohzraní** – Všechny komponenty webu (klienti, servery nebo síťoví zprostředkovatelé) musí mít jednotné rozhraní. Pokud se však některá komponenta odchyluje od zavedených principů, může dojít k poruše.
- **Vrstvený systém** – Tento systém umožňuje nasadit mezi klientem a serverem síťové zprostředkovatele. Tento zprostředkovatel zachytí komunikaci mezi klientem a serverem pro daný účel. To vede k vynucení zabezpečení a vyrovnání zatížení.

[5] API – Application Programming Interface (rozhraní pro programování aplikací)

- **Cacheable** (schopen být uložen v mezipaměti) – Po webovém serveru je vyžadováno, aby deklaroval schopnost uložení mezipaměti pro každou odpověď. Výhodu ukládání do mezipaměti může zaznamenat klient v podobě snížení latence. Taktéž se snižují náklady a zvyšuje spolehlivost celé aplikace.
- **Bezstavový** – Každý klient musí zahrnovat veškeré důležité informace potřebné k jakékoliv interakci s webovým serverem. Komplexnost celé komunikace je řízena klientem, což přispívá k tomu, že webový server tak může obsloužit větší množství klientů.



Obrázek 2.4: Architektura *RESTful* aplikace. Obrázek inspirován [24]

Kapitola 3

Platformy pro realizaci vestavěných systémů

Vestavěný systém je systém hardwaru a softwaru, který je navržen k provádění určitých funkcí ve větším elektrickém nebo mechanickém zařízení. Těmito zařízeními mohou být například mobilní telefony, chytré hodinky nebo automobily. Moderní vestavěné systém jsou velmi často řízeny mikrokontroléry.

Mikrokontrolér, nebo také jednočipový počítač, je integrovaný obvod, který umí vykonávat jednoúčelové aplikace a řídit zařízení, do kterých je vestavěn. Používá tedy centrální procesorovou jednotku (CPU), paměť s náhodným přístupem (RAM), ROM a vstupy/výstupy mikroprocesoru. Jejich využití je velice široké. Mikrokontroléry můžeme najít například v klimatizaci, televizi, mobilních telefonech nebo také v chytrých žárovkách. Hodí se zejména pro jednoduché úlohy s malým výpočetním výkonem, avšak tento výkon se stále zvyšuje. I tento fakt přispívá k masovému rozšíření mikrokontrolérů.

Tato kapitola popisuje několik druhů hojně používaných platforem pro realizace vestavěných systému, jejich architekturu a sběrnice, které využívají pro komunikaci s periferiemi, které jsou k nim připojeny. Tento přehled není plně reprezentativní, přičemž jsou zde zmíněny takové možnosti, které jsou z mého pohledu pro realizaci této práce nejvhodnější.

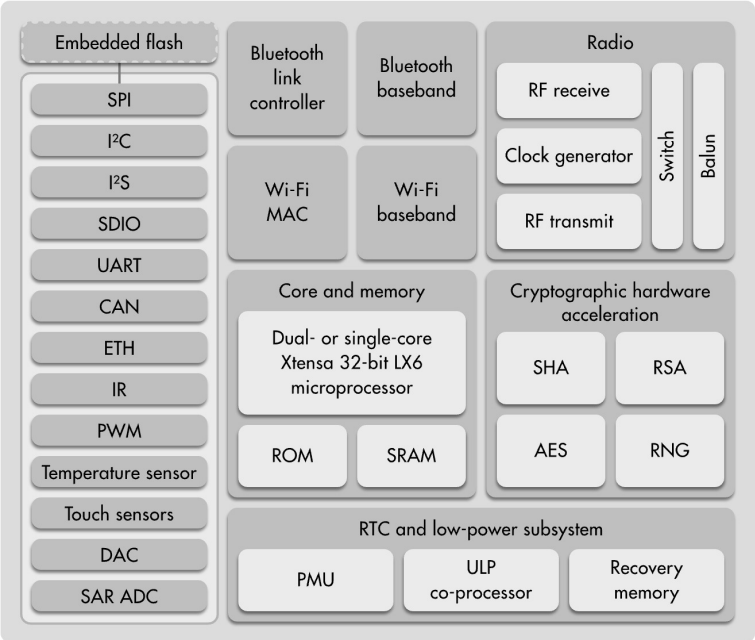
3.1 ESP32

ESP32 je označení nízkonákladového mikrokontroléru navrženého čínskou společností Espressif Systems. Jde o nástupce úspěšného mikrokontroléru ESP8266 vydaného v roce 2016. Jedná se o Wi-Fi řešení s přenosovou rychlostí až 150 Mbs, které je schopno samostatně spouštět aplikace. Existuje mnoho druhů modelů tohoto mikrokontroléru s různými výkonostními specifikacemi. Díky jeho nízké ceně se dnes velmi často používá v chytrých domácnostech a v nejrůznějších IoT řešeních [16].

ESP32 pracuje s napětím 3.3 V, obsahuje dvoujádrový, 32 bitový procesor Tensilica, konkrétně model Xtensa L108 o frekvenci až 240 MHz, který je postavený na architektuře RISC. Jeho RAM má velikost 530 kB, ROM má velikost 448 kB a obsahuje 34 vstupně–výstupních (GPIO) konektorů. K mikrokontroléru ESP32 je možno připojit až 3 zařízení komunikující přes sériové rozhraní SPI, až 2 zařízení přes I^2C a 2 zařízení přes I^2S. Dále je možno připojit až 3 zařízení přes UART. Tento mikrokontrolér také disponuje možností komunikace s ostatními zařízeními přes Bluetooth 4.2 a to jak v klasické verzi, tak v úsporné

verzi LE (Low Energy). Wi-fi modul plně podporuje protokol 802.11n pro 2,4 GHz pásmo [16].

Na platformě ESP2 se také nachází moduly pro hardwarovou podporu kryptografie, konkrétně se jedná o podporu standardu AES, algoritmů SHA a RSA a generátoru náhodných čísel RNG.



Obrázek 3.1: Funkční blokový diagram mikrokontroléru ESP32[1]

3.1.1 Varianty

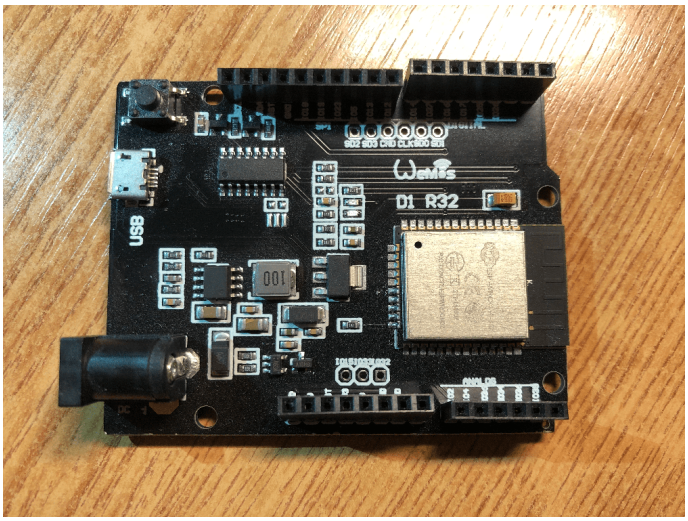
Jak již bylo naznačeno výše, existuje velké množství variant založených na platformě ESP32. V podstatě se jedné o samotné SoC[2], moduly a vývojové desky. Mezi nejznámější moduly patří ESP32-WROOM-32, který obsahuje anténu na desce plošných spojů. Jeho nejnovější verze WROOM-32E, která je využívána i v této práci, obsahuje až 16 MB FLASH paměti [16]. Moduly WROOM jsou znázorněn na obrázku 3.2.

Dalším typem jsou vývojové desky založené na ESP32. Příkladem může být deska ESP32-DevKitC, kterou vyrábí přímo společnost Espressif Systems, nebo také deska Wemos D1 R32, která byla použita i v této práci k prototypování při vývoji firmwaru. Výhodou těchto desek je, že obsahují USB/TTL převodník, který slouží k nahrávaní obslužného firmwaru. Poté stačí tuto desku jen připojit vhodným USB kabelem k počítači. Dále tyto desky obvykle obsahují programovatelné LED diody a tlačítka, z nichž jedno zpravidla slouží k restartu [16]. Příklad desek je znázorněny na obrázku 3.3.

[1] Převzato z http://esp32.net

[2] SoC – System on Chip (Systém na čipu – integrovaný obvod, který sdružuje všechny části počítače do jediného čipu)

Obrázek 3.2: Moduly ESP32-WROOM-32E



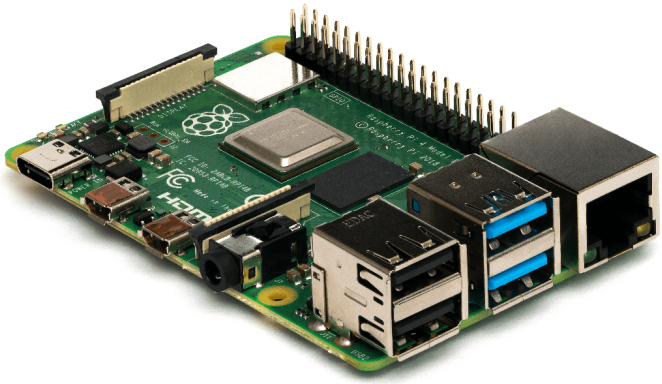
Obrázek 3.3: Vývojová deska Wemos D1 R32

3.2 Raspberry Pi

Raspberry Pi je označení pro jednodeskový počítač, který je co do velikosti srovnatelný s platební kartou. První verze tohoto počítače byla vydána už v roce 2012 a od té doby si Raspberry Pi získalo velkou popularitu nejen díky své ceně, velikosti, ale také výkonu. Nejnovější verze Raspberry Pi 4 Model B vydaná v roce 2019 disponuje čtyřjádrovým ARM procesorem a až 8 GB RAM paměti (dle varianty) [10].

Nejnovější varianta počítače Raspberry Pi obsahuje USB 2.0 a USB 3.0 porty oba po dvou kusech. Vedle nich se nachází gigabitový ethernet port. Dále se na desce nachází dva micro-HDMI porty, které podporují připojení až dvou monitorů současně a to při rozlišení 4K a šedesáti snímcích za sekundu. Je zde také přítomen port pro vložení SD karty, na které je typicky nahrán operační systém. Raspberry Pi také obsahuje 28 vstupně-výstupních pinů, které slouží pro připojení nejrůznějších periferií [10].

Raspberry Pi 4 také podporuje Bluetooth Low Energy a bezdrátové připojení přes Wi-Fi 5 [10].



Obrázek 3.4: Jednodeskový počítač Raspberry Pi 4 Model B[3]

3.3 ARM Cortex-M

ARM Cortex-M je rodina 32bitových procesorů navržených společností ARM. Tato řada procesorů je nejčastěji součástí čipů mikrokontrolérů a jsou navrženy pro energeticky optimalizované integrované obvody [30].

Mezi hlavní výhody procesorů rodiny ARM Cortex-M patří nízká spotřeba energie, vysoký výkon, vynikající energetická účinnost daná zejména rychlým prováděním úkolů, přičemž systém může zůstat delší dobu v režimu spánku. Dalšími výhodami jsou konfigurovatelné řadiče přerušení a jednoduchost použití. Při vývoji obslužného firmwaru si programátor vystačí s programovacím jazykem C [30].

Procesorů, které patří do této rodiny, je celá řada, přičemž se liší zejména výkonem a velikostí instrukční sady. Příkladem mohou být procesory Cortex-M3 a Cortex-M4, které jsou vysoce výkonné a založené na architektuře ARMv7-M. Jejich využití je typicky v mikrokontrolérech. Příkladem mikrokontrolérů, které využívají procesory Cortex-M4 jsou zařízení Kinetis K od společnosti NXP Semiconductors [30].

Dalšími procesory z rodiny ARM Cortex-M jsou Cortex-M0, Cortex-M0+ a Cortex-M1. Ty jsou založeny na architektuře ARMv6-M a disponují menší instrukční sadou. První dva jmenované procesory se využívájí především v nízkonákladových mikrokontrolérech, přičemž tyto procesory jsou velmi málo energeticky náročné. Procesor Cortex-M1 je pak navržen zejména pro FPGA[4] aplikace [30].

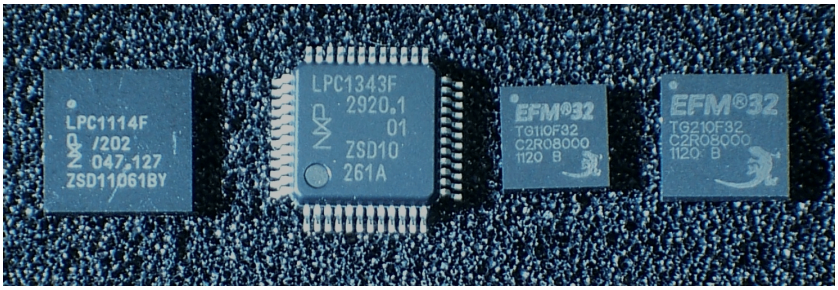
Využití těchto procesorů je velmi široké. jak již bylo zmíněno, velmi často se tyto procesory vyskytují v mikrokontrolérech, ale svoje využití najdou také v automobilovém průmyslu, v průmyslových řídících aplikacích nebo v zařízeních určených pro koncové zákazníky jako jsou LCD panely nebo jiné spotřební zboží [30].

Na obrázku 3.5 jsou pak znázorněny příklady některých mikrokontrolérů využívajících procesory z rodiny ARM Cortex-M.

[3] Převzato z https://cs.m.wikipedia.org/wiki/Soubor:Raspberry_Pi_4_Model_B_-_Side.jpg

[4] FPGA – Field Programmable Gate Array (programovatelná hradlová pole)

[5] Převzato z https://commons.wikimedia.org/wiki/File:ARM_Cortex-M0_and_M3_ICs_in_SMD_Packages.jpg



Obrázek 3.5: Příklady mikrokontrolérů založených na procesorech Cortex-M0 a Cortex-M3 od společnosti NXP[5]

3.4 Sběrnice

Sběrnice je skupina signálových vodičů, jejichž účelem je mimo jiné zajistit přenos dat mezi několika elektronickými zařízeními. Existuje několik rozhraní, které mikorokontroléry využívají pro komunikaci s periferiemi.

3.4.1 SPI

SPI (*Serial Peripheral Interface*) je sériové rozhraní, které se využívá k připojení a komunikaci s periferními zařízeními. Svoje využití však hojně nachází i pro sériový přenos obecně [22].

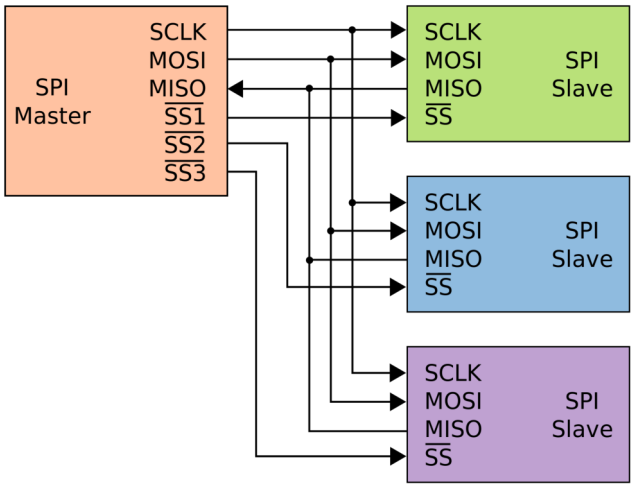
Jedná se o rozhraní typu master-slave. Zařízení připojené přes SPI může být v jednom ze dvou režimů. V praxi to znamená to, že existuje jedno centrální zařízení (master), které iniciuje komunikaci s ostatními zařízeními (slaves). Master bývá zejména mikrokontrolér, periferie pak vystupují jako slaves [22].

Rozhraní SPI využívá čtyři signály [13]:

- Hodinový signál (SCLK) odeslaný z master do všech slave zařízení. Všechny SPI signály jsou s tímto časem synchronní.
- Signál SS (*Slave select*) pro každé zařízení (slave) určuje, které z nich bude komunikovat s hlavním zařízením (master). Master zajistí, že signál SS bude po dobu komunikace v logické nule (aktivní). V určitém čase je možno komunikovat pouze s jedním zařízením slave. Ostatní slave zařízení, jejichž SS je vstup, jsou tedy v logické jedničce (neaktivní) a do komunikace nijak nezasahují.
- Datový vodič MOSI (*Master Out-Slave In*) je výstupem zařízení master a vstupem zařízení slave.
- Datový vodič MISO (*Master In-Slave Out*) je vstupem zařízení master a výstupem zařízení slave.

U rozhraní SPI probíhá výměna dat vždy oběma směry. Říkáme tedy, že jde o obousměrné spojení (full-duplex). [13]

[6] Převzato z https://commons.wikimedia.org/wiki/File:SPI_three_slaves.svg



Obrázek 3.6: Topologie sběrnice SPI znázorňující rozhraní s více zařízeními[6]

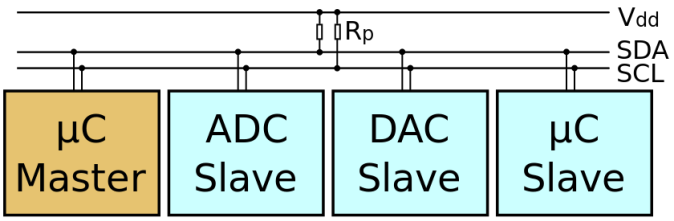
3.4.2 I²C

I²C (*Inter-Integrated Circuit*) je sériové rozhraní, které bylo navrženo a vyvinuto společností Philips Semiconductor. Podobně jako rozhraní SPI, je sběrnice I²C typu master-slave, kde zařízení master iniciují komunikaci. Zařízení takto připojené komunikují způsobem half-duplex, to znamená, že se zařízení střídají v posílání dat [22].

U tohoto rozhraní není potřeba vybírat slave zařízení, které bude komunikovat se zařízením master. I²C využívá pouze dva vodiče – jeden datový (SDA - Serial Data) a druhý přenášející synchronizační signál (SCL - Serial Clock). K těmto dvěma vodičům lze zapojit více zařízení master a zařízení slave, které spolu komunikují [22].

Komunikaci iniciuje zařízení master hranou z nuly na jedničku na SDA při SCL v logické jedničce. Vzorkování dat je zajištěno pouze při náběžné hraně SCL. Při ukončení přenosu probíhá přesně opačný postup, než při jejím zahájení a tedy generování hrany z nuly na jedničku na SDA [13].

Výhodou tohoto rozhraní je například adresování, přičemž s pomocí již zmíněných dvou vodičů je možné připojit až 127 různých zařízení [13].



Obrázek 3.7: Topologie sběrnice I²C znázorňující rozhraní s více zařízeními[7]

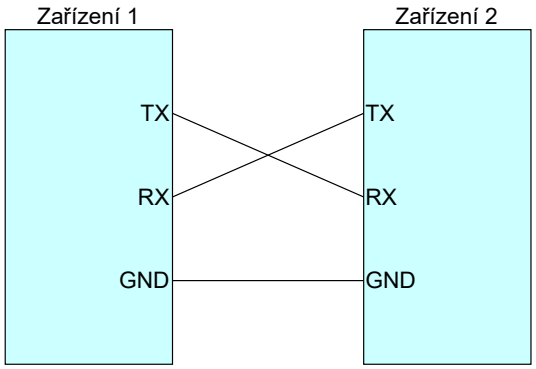
[7] Převzato z https://cs.wikipedia.org/wiki/Soubor:I2C.svg

3.4.3 UART

UART (*Universal Asynchronous Receiver-Transmitter*) není komunikační protokol jako SPI a I^2C. Nejčastěji je UART realizován jako hardwarové zařízení. Účelem této sběrnice je odesílat a přijímat data. Jeho výhodou je, že používá k přenosu pouze dva vodiče. Obě spolu komunikující zařízení mají dvojici pinů TX a RX. Pin TX na prvním zařízení je spojen s pinem RX na druhém zařízení. Stejným způsobem je propojen pin TX na druhém zařízení s pinem RX na zařízení prvním [14, 13].

UART se stará o takzvaný asynchronní přenos dat, což znamená, že při komunikaci není potřeba hodinový signál. Místo toho zařízení, které odesílá data, přidává k odesílaným datům příznakový bit startu a konce, tudíž příjemce pozná, kdy má začít číst data [14].

Data přenášená přes sběrnici UART jsou organizována do paketů, přičemž každý z nich obsahuje pět až devět bitů datových, volitelně paritní bity, jeden start bit a jeden až dva stop bity, které daný paket ukončují [14].



Obrázek 3.8: Dvě zařízení připojené přes UART. Obrázek inspirován [14].

Kapitola 4

Současný stav a návrh řešení

Důležitou částí před samotným návrhem vlastního řešení je průzkum trhu a shrnutí výhod či nevýhod již existujících řešení podobných systémů. Nově navrhovaný systém by měl mít jisté benefity oproti stávajícím řešením a měl by být inovativní alespoň v některých svých prvcích.

Dalším důležitým krokem je provést kvalitní návrh systému, který bude odpovídat předem definovaným požadavkům. V případě dobrého návrhu je snazší následná implementace bez zbytečných oprav chyb.

V této kapitole jsou stručně popsány existující řešení modulárních přístupových systémů a dále také můj vlastní návrh takového systému, který spočívá zejména v analýze požadavků na takový systém, návrh hardwarových modulů a webové aplikace.

4.1 Existující řešení

Elektronických přístupových systémů existuje poměrně velké množství. Avšak mnoho firem se specializuje pouze na skříňkové systémy. Některé z těchto firem navíc poskytují pouze off-line řešení. Takovou společností je například NESSY, s.r.o[1].

Další ze společností, která dodávají kompletní elektronický skříňkový systém, je například IKOS CZ, s.r.o.[2] nebo IVAR, a.s[3]. Systém společnosti IVAR je postavený na RFID zařízeních podporujících čipy MIFARE a dle mého názoru by byla pro daný účel nejvhodnější ze všech již dostupných možností. Skříňkový systém této firmy je možné aplikovat na stávající skříňky, tudíž není nutné do provozovny pořizovat kompletně nové skříňky. Tento systém lze spravovat online, avšak jako nedostatek vidím v softwaru, který je třeba instalovat na lokální počítač. Tento aspekt tedy nesplňuje požadavek majitele na správu a kontrolu provozovny odkudkoliv. Dalším nedostatkem je omezení systému pouze na šatní skříňky. V systému tedy chybí možnost správy vstupních dveří a klientských permanentek.

K dosažení požadovaných specifik za použití existujících řešení by bylo tedy nutné kombinovat systémy od více výrobců, což by značně zatížilo pracovníky provozovny a celkově by to vedlo spíše k znepříjemnění práce.

[1] https://www.nessy.cz/satni-zamkove-systemy
[2] https://www.ikos.cz/category/13/software-ikos-p3
[3] https://info.ivar.cz/skrinkove-systemy/

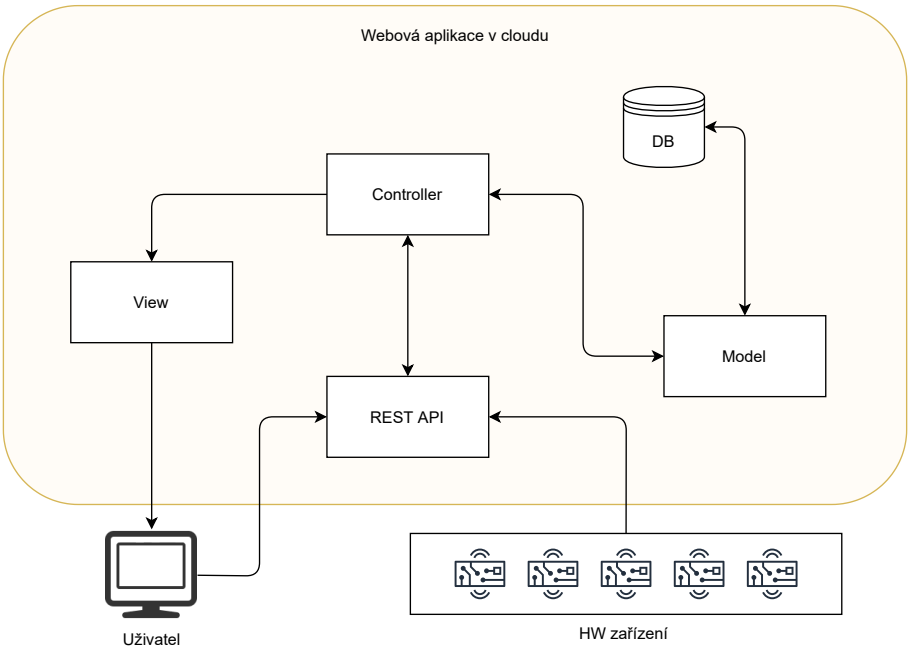
4.2 Obecný návrh řešení

Tato práce se dá rozdělit do dvou částí – hardware a informační systém s webovou prezentací. Hardwarová část se skládá ze dvou modulů. První částí je modul, který ovládá chytrý zámek v šatních skříňkách a vstupních dveřích do provozovny, a druhou je čtecí modul na recepci v sauně, který zaměstnancům provozovny slouží ke skenování klientských permanentek a k přiřazovaní RFID čipů ke skříňkám (klíče). V informačním systému jsou pak evidovány veškeré údaje o klientských kartách, skříňkách a klíčích.

Návrh vlastního řešení přístupového systému byl vytvořen na základě analýzy požadavků na daný systém. Ten by měl splňovat několik hlavních aspektů:

- **Cena** – Cena realizace a provozu kompletního systému by měla být co nejnižší, zejména pak za hardwarové komponenty.
- **Intuitivnost a jednoduchost** – Práce se systémem by měla být snadná a všechny prvky systému by měly být intuitivní. Zejména by měl být celý systém uživatelsky přívětivý i pro netechnicky zdatné uživatele.
- **Autorizace** – Každý uživatel musí mít svoje přístupové údaje do systému. Autorizovaný uživatel pak spadá do určité role a každá taková role má jiná přístupová práva pro ovládání systému.
- **Dostupnost** – Celý systém musí být přístupný odkudkoliv. Požadavkem je kontrola provozovny vzdáleně.

Na obrázku 4.1 je znázorněna architektura systému navrhovaného v této práci. Celý systém je navržen jako MVC aplikace (vysvětleno v kapitole 5.3.1) s REST API komunikující s uživateli a hardwarovými zařízeními přes HTTP. Serverová část tohoto systému bude nasazena v cloudu.



Obrázek 4.1: Architektura navrhovaného systému

Detailní popis návrhu všech částí systému bude popsán v dalších částech této kapitoly. V kapitole 4.3 je vysvětlen způsob návrhu hardwarových zařízení, v kapitole 4.5 je objasněn návrh relační databáze pro informační systém, v kapitole 4.4 je pak popsán návrh serverové části aplikace, kam spadá i návrh rolí a přístupových práv pomocí diagramu případů užití, a nakonec je v kapitole 4.6 shrnut návrh uživatelského rozhraní.

4.3 Návrh hardwaru

V návrhu byl využit mikrokontrolér ESP32-WROOM-32E, který komunikuje přes Wi-Fi se serverem. Pro tento účel je možné využít například i Raspberry Pi, které je ale moc velké a disponuje příliš mnoho funkcemi, které pro účel práce nejsou potřeba.

Jelikož je vhodné mít celý modul ve skříňkách kompaktní a odizolovaný, rozhodl jsem se pro všechny moduly vyrobit desky plošných spojů, které obsahují sloty pro zapojení RFID čtecího zařízení a připojení FTDI modulu, které slouží k nahrání firmwaru do mikrokontroléru, napájecí svorkovnici a svorkovnici pro připojení elektromagnetického zámku. Všechny osazené moduly budou poté umístěny do na míru navržené krabičky vytištěné na 3D tiskárně.

V této podkapitole je popsán způsob návrhu jednotlivých hardwarových modulů. Jsou zde popsány jednotlivé části návrhu schématu zapojení obou vytvářených koncových zařízení. Kompletní schéma zapojení se nachází v příloze.

4.3.1 Napájecí systém a programovací rozhraní

Celá deska kromě zámku potřebuje ke svému napájení 3,3 V a maximálně 0.3 A. Jelikož zámek potřebuje 12 V, je třeba k desce dovést dvě napájecí větve - na 3,3 V a na 12 V. To by ale znamenalo mnoho kabeláže, zapojení v šatních skříňkách by nevypadalo esteticky a navíc by se celé provedení prodražilo. Proto bylo zvoleno napájení pouze 12 V větví a na desce byl použit lineární stabilizátor napětí. Ten je dle dokumentace [8] schopen vstupních 12 V regulovat na výstupních 3,3 V potřebných pro napájení zbylých komponentů na desce.

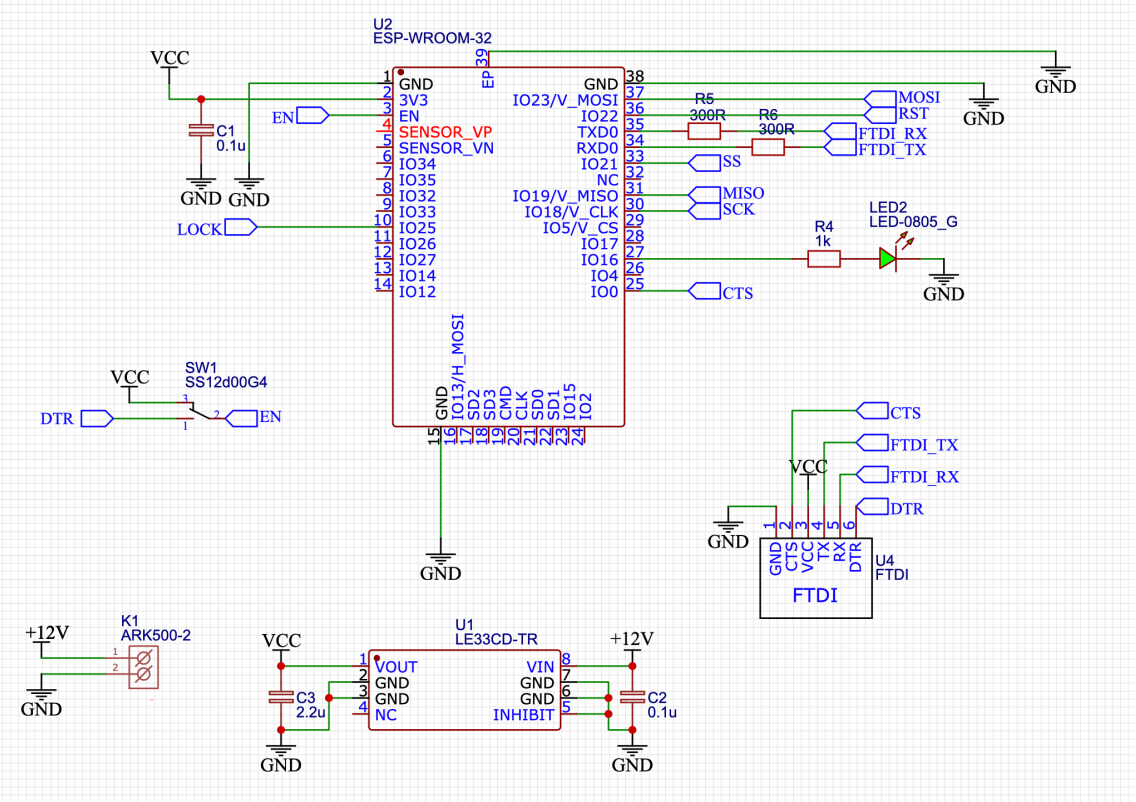
Nahrávání firmwaru je řešeno přes USB/TTL převodník s čipem FT232RL od společnosti FTDI, pro který se na navržené desce nachází dutinková lišta, kam je možné tento modul vsunout. Tento převodník komunikuje s mikrokontrolérem pomocí sběrnice UART.

Výstupní napětí lineárního regulátoru je přivedeno na přepínač. Pokud je tento přepínač v poloze EN (Enable), mikrokontrolér ESP32 je zapnut a připraven k práci. Druhá poloha tohoto přepínače znamená trvalé vypnutí mikrokontroléru. Pokud je ale k desce připojen modul FTDI, tato poloha přepínače umožní nahrát firmware do mikrokontroléru ESP32 a FTDI se stará o jeho reboot.

4.3.2 Čtecí zařízení

Pro účely této práce bylo vybráno RFID čtecí zařízení MFRC522 od společnosti NXP Semiconductors. Jedná se o integrovaný obvod pro bezkontaktní komunikaci na frekvenci 13,56 MHz. Toto zařízení je navrženo pro komunikaci s transpondéry ISO/IEC 14443 A/-MIFARE, které nemusí disponovat dalšími aktivními obvody. Zařízení MFRC522 disponuje jak funkcí čtení, tak i funkcí zápisu na vzdálenost do 50 milimetrů [9].

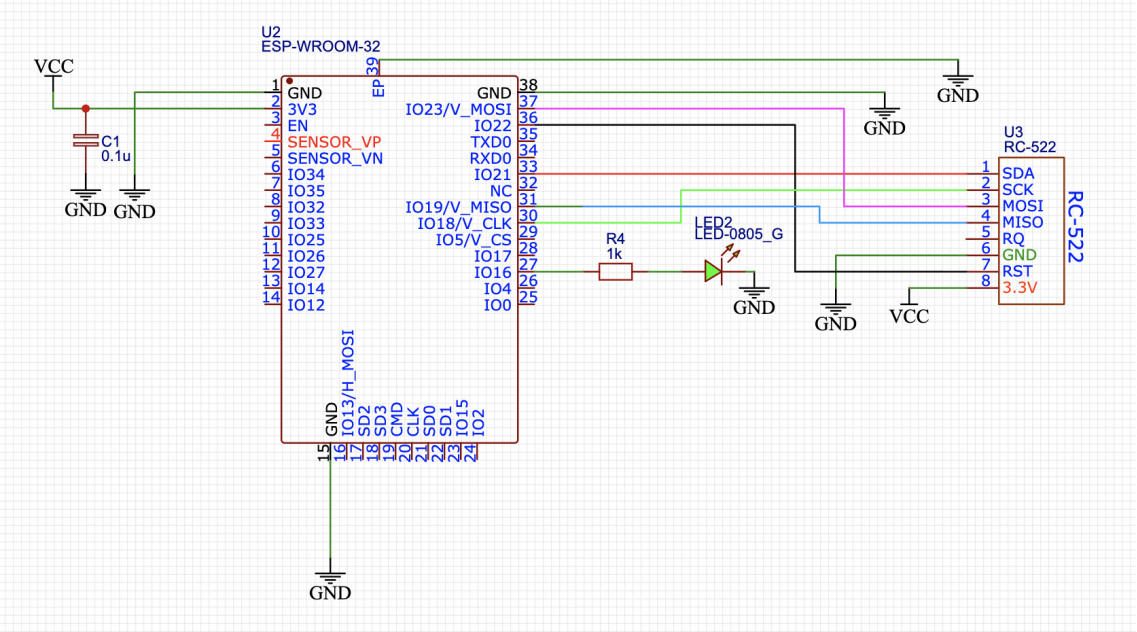
Pro tuto práci byly zvoleny karty a čipové hodinky s RFID čipem Mifare S50 s pamětí 1 kB. Čipové hodinky mají náramek z nylonu a jsou tedy vhodné do wellness center, plováren, saun a jsou odolné proti prachu špíně a vodě.



Obrázek 4.2: Schéma zapojení napájecího systému a programovacího rozhraní

Obrázek 4.3: RFID čtecí zařízení MFRC522

Čtečka RFID karet potřebuje ke svému napájení napětí 3,3 V a je připojena k mikrokontroléru ESP32 prostřednictvím sběrnice SPI, jak je vidět na obázku 4.4. Datový vodič vede od pinu jedna (SDA) na čtečce do vstupně–výstupního pinu `IO21` na zařízení ESP32.



Obrázek 4.4: Schéma zapojení čtečky MRFC522 k mikrokontroléru ESP32-WROOM-32E

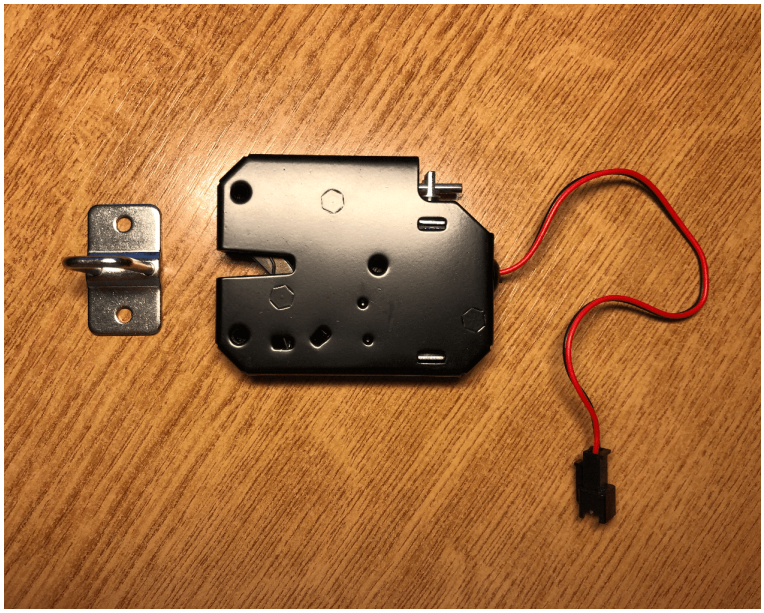
4.3.3 Elektromagnetický zámek

Pro otvírání šatních skříněk byl zvolen elektromagnetický zámek znázorněný na obrázku 4.5, který pro svoje otevření potřebuje 12 V a 2 A stejnosměrným proudem. Obsluha tohoto zámku je navržena pomocí elektromagnetického relé a tranzistoru NPN. Celé zapojení je inspirováno schématem zapojení modulu Wemos Relay Shield [12]. Nicméně se zapojení dá zkontrolovat jednoduchými výpočty. Relé použité v této práci je určeno pro napětí 12 V na cívce a odpor jejího vinutí je dle dokumentace [11] 400 Ω. Vyčtením ze zmíněné dokumentace nebo použitím Ohmova zákona zjistíme, že relé poteče proud 30 mA. Aby tranzistor v mém obvodu sepnul 30 mA, je potřeba, aby proud tekoucí jeho bází měl alespoň velikost proudu tekoucí kolektorem vyděleným proudovým zesilovacím činitelem tranzistoru (hFE) [23]. Z dokumentace k použitému tranzistoru [2] lze vyčíst, že proudový zesilovací činitel má pro velikost proudu na kolektoru 100 mA velikost nejméně 250. Pokud vydělíme spočtených 30 mA tímto činitelem, dostaneme minimální proud, který sepne relé v schématu, když teče bází tranzistoru, tedy 0,12 mA. Z dokumentace lze také vyčíst, že v sepnutém stavu bude napětí mezi emitorem a kolektorem 0,7 V. Mikrokontrolér ESP32 dává na výstupu napětí 3,3 V, na rezistor R3 pak zbývá napětí 2,6 V. Aplikací Ohmova zákona zjistíme, že proud, který by tímto rezistorem procházel, kdyby před ním nebyly zapojeny dva další rezistory, by byl 2,6 mA. Výsledný proud protékající bází tranzistoru bude tedy o něco nižší, ale bude několikanásobně větší, než minimální požadovaný proud pro sepnutí.

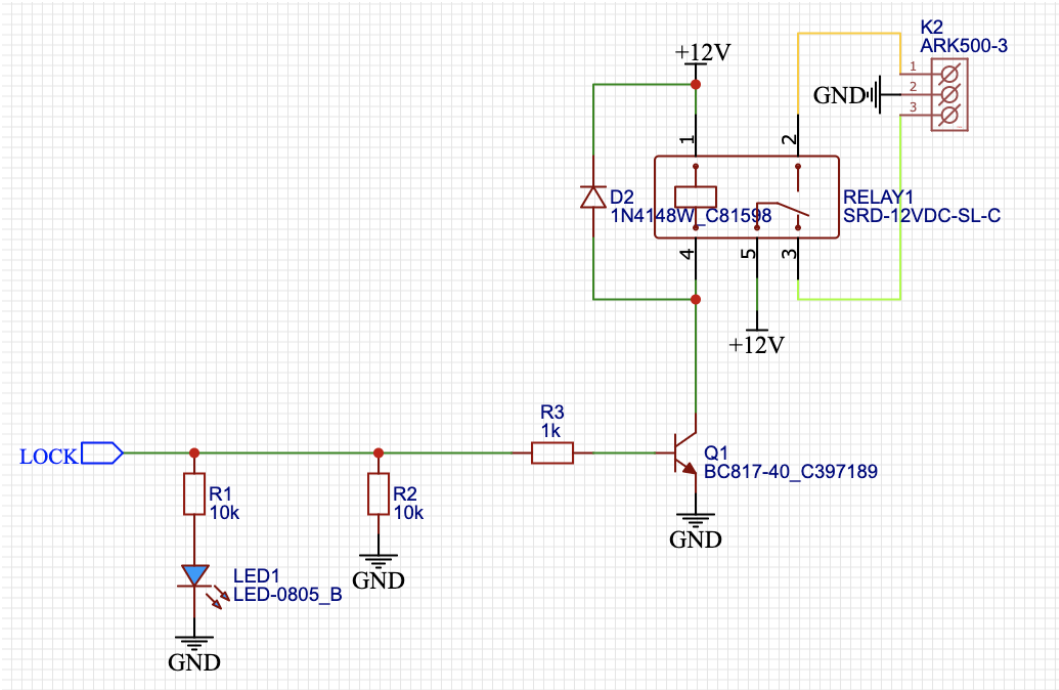
K relé bylo také nutné paralelně připojit diodu. Důvodem je ochrana proti přepětí, které může vytvořit cívka relé při rozepnutí proudu. Tato napěťová špička může následně poškodit spínací tranzistor.

Pro otevírání dveří byl zvolen standardní elektromagnetický bzučák, který ke svému fungovaní potřebuje stejné podmínky, jako elektromagnetický zámek do skříněk.

Logika obsluhy tohoto zámku je znázorněna na obrázku 4.6.



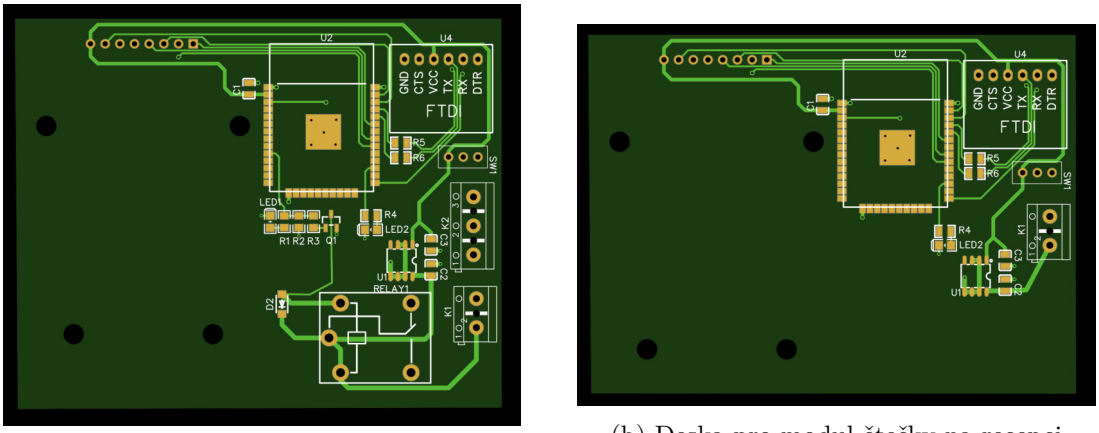
Obrázek 4.5: Elektromagnetický zámek sloužící k otevírání šatních skříněk



Obrázek 4.6: Schéma zapojení logiky otevírání elektromagnetického zámku

4.3.4 Návrh desek plošných spojů

Po nakreslení schématu bylo třeba navrhnout rozložení komponentů na deskách plošných spojů. Bylo využito online editoru EasyEDA a výsledné nákresy desek jsou znázorněny na obrázku 4.7. Při návrhu desek plošných spojů byl kladen důraz zejména na velikost celého modulu, který bude umístěn ve skříňce. Také bylo třeba si dát pozor na umístění svorkovnic, které jsou záměrně na pravé straně desky, kvůli správnému vedení kabeláže ve skříňkách.



(a) Deska zámkových modulů

(b) Deska pro modul čtečky na recepci

Obrázek 4.7: Návrh obou desek plošných spojů

4.4 Návrh serverové části

Hlavním úkolem serveru je ukládání důležitých informací o klientských kartách a klíčích pro otevírání skříněk či dveří. Serverová část je z hlediska vývoje nejen webových aplikací velice komplexní částí. Serverová část je jádrem celého systém, přičemž je zde implementována logika celé webové aplikace a definován způsob práce s databází a daty. Server se také stará o požadavky, které klienti posílají na server a žádají odpovědi v podobě dat.

Serverová část byla navržena jako *RESTful* webová aplikace komunikující nad protokolem HTTP. Tudíž všichni uživatelé a hardwarová zařízení na server odesílají HTTP požadavky a přijímají HTTP odpovědi z tohoto serveru. Server poskytuje API, které využívají jak klientské aplikace, tak hardwarová zařízení pro komunikaci se serverem.

4.4.1 Přístupová práva

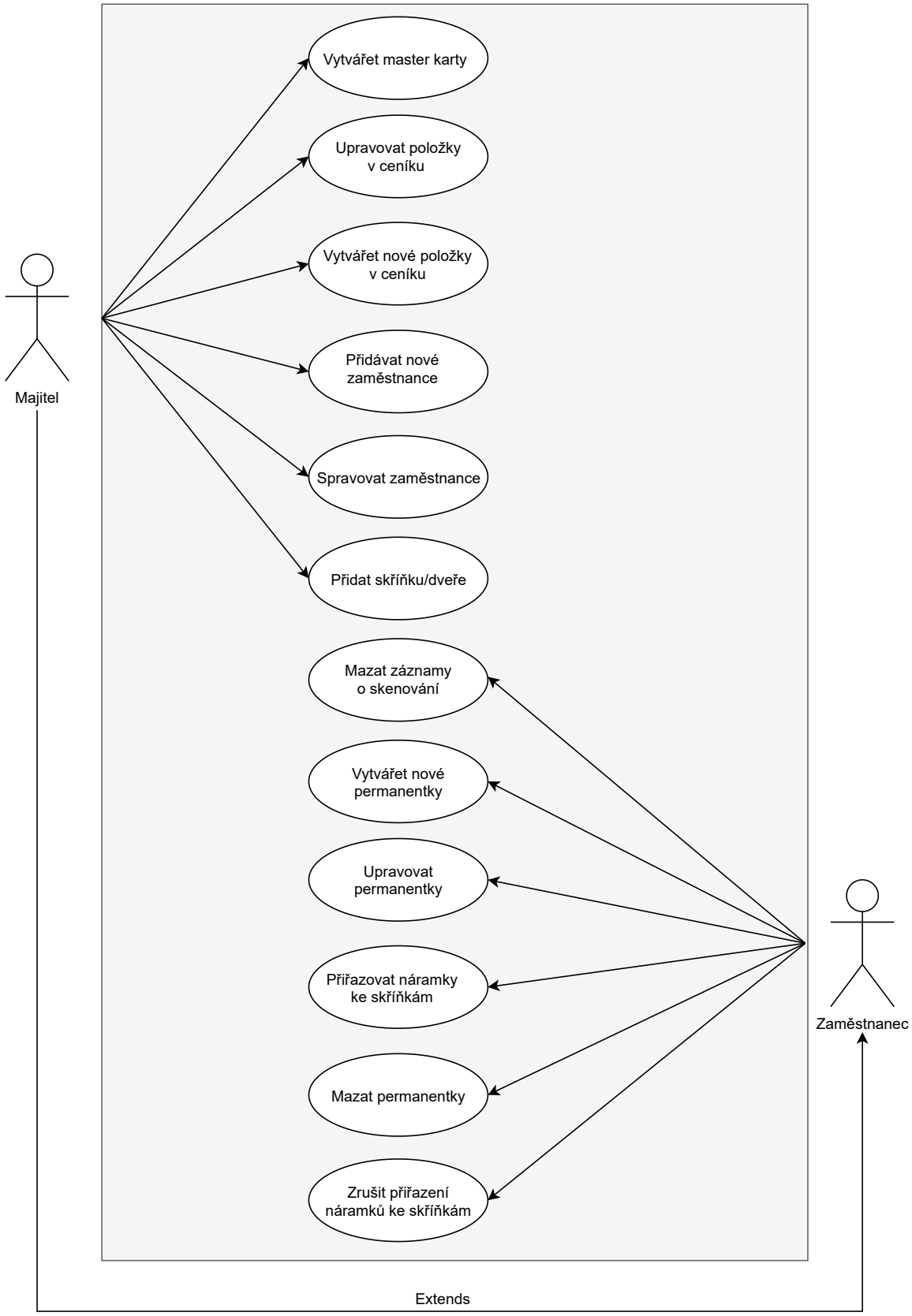
V systému je třeba také rozlišovat role uživatelů. Každá taková role má různé vlastnosti a práva k vykonání určitých akcí.

Mezi dvě skupiny lze řadit:

- Majitel
- Zaměstnanec

Uživatel s rolí *Majitel* má na starost správu systému, do které spadá přidávání a úprava zaměstnanců, vytváření a úprava takzvaných master karet, které dokáží otevřít jakoukoliv skříňku nebo dveře. *Majitel* může také upravovat položky v ceníku a vytvářet položky nové. Uživatel s rolí *Zaměstnanec* pak může provádět mazání záznamů o skenování, vytváření, mazání a úpravu permanentek a přiřazování náramků ke skříňkám případně rušení. Tyto dvě role jsou hierarchicky provázány, kde uživatel ve skupině *Zaměstnanec* je na nižší úrovni, než uživatel ve skupině *Majitel*. To znamená, že *Majitel* má všechna práva, jako *Zaměstnanec*.

Na obrázku 4.8 jsou graficky znázorněny operace, které uživatel v dané roli může provádět. Toto znázornění bylo vytvořeno pomocí diagramu případu užití, což jeden z diagramů definovaných v jazyku UML. Tento diagram zachycuje aktéry a vztahy mezi nimi, přičemž tento vztah znamená, že aktér může vykonávat určitou funkci [18].



Obrázek 4.8: Diagram případů užití

4.5 Návrh databáze

Součástí systému je také databáze, která úzce souvisí se serverovou částí . V ní je třeba evidovat uživatele aplikace, skříňky, klienty, RFID transpondéry (karty a klíče ke skříňkám a dveřím) a záznamy o skenování. Tyto položky vystupují v návrhu jako entitní množiny. U každé takové množiny je třeba evidovat následující údaje:

- **Uživatel** – primární klíč, jméno, příjmení, role, kontaktní údaje a přihlašovací údaje do systému,
- **Skříňka** – jednoznačné identifikační číslo skříňky a její stav (volná nebo obsazená),
- **Dveře** – jednoznačný identifikátor daných dveří, specifikace, které karty dané dveře mohou otevírat (jestli pouze master karty nebo i klientské náramky),
- **Klient** – primární klíč, jméno, příjmení, e-mail, telefonní číslo, typ permanentky (5 vstupů, 10 vstupů nebo celosezónní), zbývající počet vstupů,
- **Dveře** – UID karty a její typ (permanentka, klíč ke skříňkám nebo dveřím přístupným klientům, nebo master karta, která slouží k otevření čehokoliv v provozovně),
- **Záznam o skenování** – jedná se o tabulku, ve které jsou uložena všechna provedená skenování na čtečce na recepci a která obsahuje pouze primární klíč a UID karty jako atribut,
- **Ceník** – tabulka, která obsahuje název položky a její cenu.

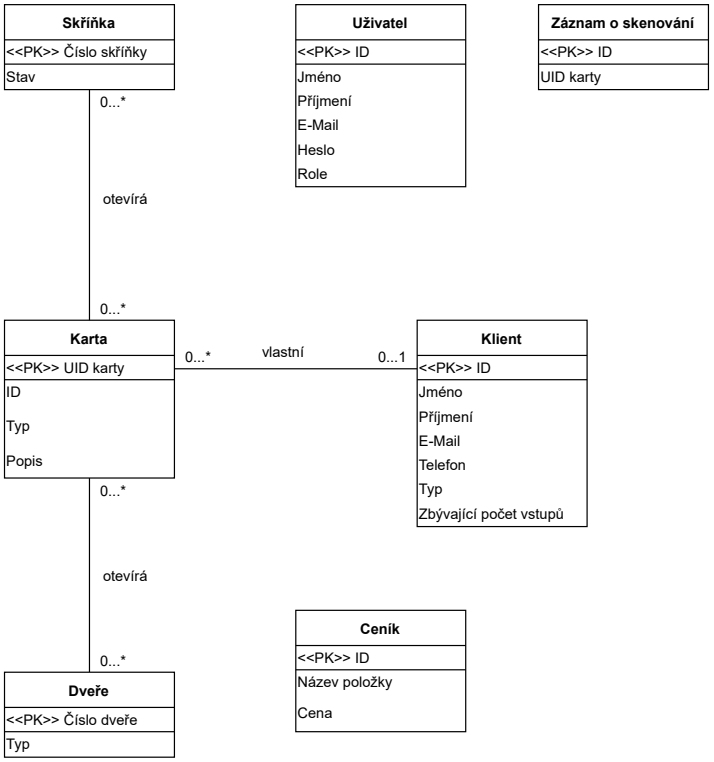
Návrh databáze tohoto systému byl modelován pomocí *ER diagramu*, kde jsou modelovány entity reprezentující tabulky databáze a vztahy mezi nimi. *ER diagram* je jedním z diagramů používaných pro konceptuální znázornění dat v softwarovém inženýrství. Tento diagram je zpravidla modelován pomocí modelovacího jazyka UML (*Unified Modeling Language*). Každá entita v databázi má svůj název a musí mít primární klíč, který jednoznačně identifikuje záznam v tabulce. Dále tato entita obsahuje atributy. Parametry asociace mezi entitami jsou typ vztahu a kardinalita vazby, což je počet vztahů, ve kterých participuje jedna entita [18]. Na obrázku 4.9 je znázorněn návrh databáze pomocí *ER diagramu*.

4.6 Návrh uživatelského rozhraní

Další částí je návrh přívětivého a intuitivního rozhraní, které bude uživatelům sloužit k interakci s navrhovaným systémem. Toto rozhraní bude majiteli a zaměstnancům sauny sloužit pro správu permanentek a zámkových modulů.

Tato aplikace je navržena ve stylu takzvaného *dashboardu*, což je typ grafického uživatelského rozhraní, které přehledně ukazuje uživatelům nejdůležitější informace o systému. Toto rozhraní se skládá z ovládacího panelu, který bývá obvykle umístěn v levé nebo vrchní části obrazovky, a hlavního zobrazovacího okna s požadovanými informacemi, které v případě mého systému bude zobrazovat seřazená data především v přehledných tabulkách.

Do této sekce se uživatel dostane pouze po autentizaci. Stránka s přihlášením bude dostupná z webové prezentace sauny a přístup do *dashboardu* mají pouze zaměstnanci a majitel provozovny. Po úspěšném přihlášení je uživatel přesměrován na hlavní stránku klientské aplikace. Ta zobrazuje tabulku se záznamy o skenování. Je to zejména z toho důvodu, že zaměstnanec sauny bude při své práci se systémem v této sekci trávit největší množství času.



Obrázek 4.9: ER diagram databáze

Navigační panel dále bude obsahovat sekce *Permanentky*, *Skříňky*, *Dveře*, *Master karty*, *Ceník* a *Uživatelé*. Při kliknutí na svoje uživatelské jméno je pak možné zobrazit informace o svém profilu, kde lze změnit svoje přihlašovací údaje.

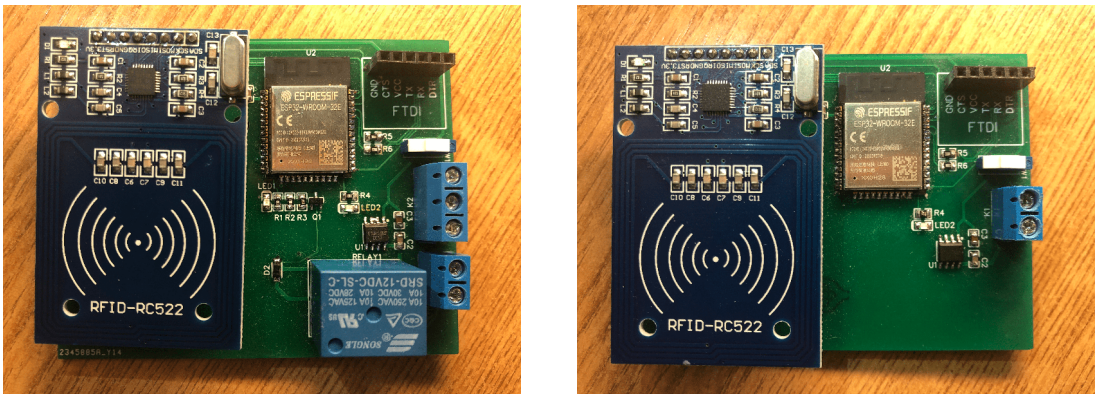
Kapitola 5

Implementace

V této kapitole je popsán způsob implementace modulárního přístupového systému. Jsou zde popsány technologie, programovací jazyky a knihovny, které jsem si pro realizaci firmwaru mikrokontroléru a informačního systému vybral, a způsob implementace celého systému zejména prostřednictvím fragmentů důležitých pasáží kódu.

5.1 Realizace koncových zařízení

Po návrhu obvodového zapojení koncových zařízení a schématu desek plošných spojů přišla na řadu jejich technická realizace. Navržené desky byly poslány do výroby, k čemuž byla využita čínská společnost JLCPCB[1].



(a) Osazená deska zámkových modulů (b) Osazená deska pro modul čtečky na recepci

Obrázek 5.1: Osazené desky plošných spojů

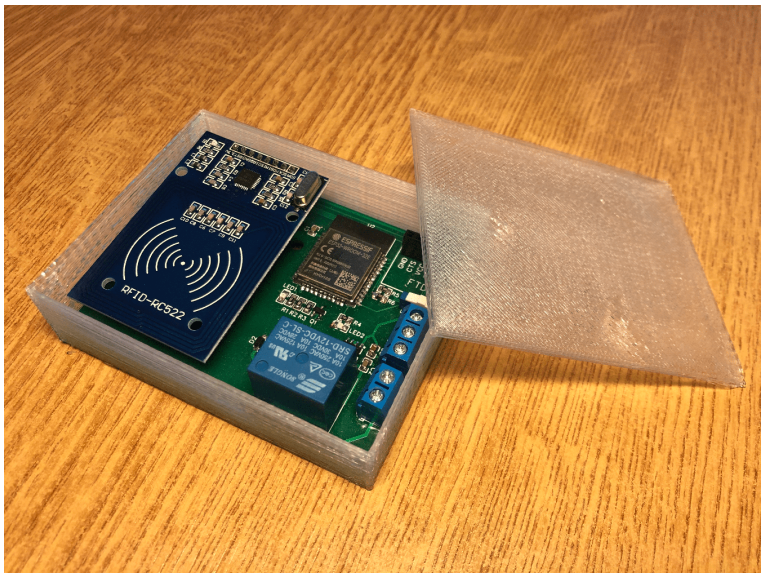
Obě desky byly osazeny potřebnými komponenty, jimiž jsou mikrokontrolér ESP32-WROOM-32E, lineární stabilizátor napětí LE33CD-TR, relé SRD-12VDC-SL-C, tranzistor BC817-40, dutinkové lišty pro připojení RFID čtecího zařízení MFRC522 a FTDI modulu k nahrávání firmwaru a další potřebné komponenty jako jsou rezistory a kondenzátory. Dále také deska obsahuje jednu zelenou status LED diodu a jednu modrou LED diodu, která svítí při aktivaci relé.

[1] https://jlcpcb.com

5.1.1 Krabičky na moduly

Zhotovené desky je před samotnou instalací do provozovny umístit do krabiček. K tomu byla využita 3D tiskárna. K návrhu modelu krabičky byl využit program OpenSCAD a generátor krabiček, který se mi podařilo nalézt na platformě pro sdílení 3D modelů[2].

K tisku krabičky byl použit průhledný PLA (polyaktidová vlákna).



Obrázek 5.2: 3D krabička pro koncová zařízení

5.1.2 Napájení

Po realizaci modulů bylo potřeba vymyslet, jakým způsobem budou připojeny k napájení. Modul na recepci je napájen standardním běžně dostupným 12 V napájecím adaptérem. Avšak napájení modulů ve skříňkách a u vstupních dveří do provozovny bylo potřeba implementovat sofistikovaněji. Pro tento účel byl zvolen ATX zdroj značky EVOLVE s výkonem 450 W. ATX zdroj je standardem mezi napájecími zdroji pro stolní počítače. Takové zdroje disponují ochranami proti nadproudu, přepětí nebo zkratu, takže nemůže dojít k poškození zdroje či zámkových modulů. Napájecí zdroj s takovým výkonem je pro účel této práce více než dostačující i se značnou rezervou.

Podle ATX specifikace [6] disponuje zdroj několika napájecími větvemi. Vodiče těchto větví jsou typicky barevně označeny. Po účel napájení zámkových modulů byla využita 12 V větev označena žlutou barvou a GND větev označena černou barvou. Původní konektory byl odstřiženy a k těmto dvěma větvím bylo připájeno celkem 22 vodičů (11 napájecích, 11 zemnících), které budou napájet celkem deset skříňkových modulů a jeden modul u vstupních dveří. Po zapojení do zásuvky ale takto upravený zdroj nebude fungovat. Proto bylo ještě potřeba spojit zelený EN (*Enable*) vodič s GND vodičem. Všechny připájené části byly zaizolovány smršťovacími bužírkami, případně izolační páskou. Výsledný upravený ATX zdroj je na obrázku 5.3.

[2] https://www.thingiverse.com/thing:2594893

Obrázek 5.3: Upravený ATX zdroj pro napájení zámkových modulů

5.2 Programování mikrokontroléru

První částí programové implementace bylo programování firmwaru pro obsluhu navržených desek. Programování ESP32 je možné ve známém a hojně používaném vývojovém prostředí Arduino IDE. Programy se programují typicky v jazyce Wiring, což je open-source framework, který slouží k programování mikrokontrolérů, zejména pak platformy Arduino. Jeho syntaxe je velmi podobná programovacím jazykům C/C++, není tedy těžké se při znalosti těchto dvou programovacích jazyků v syntaxi jazyka Wiring zorientovat.

Důležitými rysy programování mikrokontroléru ESP32 v prostředí Arduino IDE jsou dvě hlavní funkce, bez kterých výsledný program nelze přeložit. První z nich je funkce `setup()`, kde se nachází počáteční nastavení programu, která jsou spuštěna pouze jednou ihned po spuštění mikrokontroléru. Druhou funkcí je `loop()`, jenž je automaticky periodicky volána po celou dobu, kdy je mikrokontrolér připojen k napájení. Obsahuje tedy kód, který běží neustále.

V této práci byly vytvořeny dva firmwary. Jeden, který se stará o obsluhu zámkových modulů. Jeho úkolem je odesílat HTTP požadavky na server a zpracovávat odpovědi. Na základě této odpovědi je klientovi umožněn, případně neumožněn, přístup do provozovny nebo skříňky.

Druhý firmware slouží k obsluze čtecího zařízení umístěného na recepci. Tento modul se stará o odesílání záznamů o skenování na server.

5.2.1 Knihovna pro čtení RFID karet

Důležitou součástí firmwaru mikrokontroléru je knihovna pro práci se čtečkou RFID karet MFRC522[3]. Ta slouží pro práci se čtečkou použitou v této práci připojenou přes rozhraní SPI. Tato knihovna umožňuje mikrokontroléru čtení a zápis na RFID transpondér přiložený ke čtečce. Ve výpise 5.1 je znázorněna práce s touto knihovnou.

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

```
int getUid() {
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return 0;
  }
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return 0;
  }
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
     uid.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));
     uid.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  uid.toUpperCase();
  if (ledState == LOW) {
    ledState = HIGH;
  }
  else {
    ledState = LOW;
  }
  digitalWrite(ledPin, ledState);
  delay(200);
  if (ledState == LOW) {
    ledState = HIGH;
  }
  else {
    ledState = LOW;
  }
  digitalWrite(ledPin, ledState);
  return 1;
}
```

Výpis 5.1: **Fragment kódu pro skenování RFID karet a čtení ID karty.** V této části kódu lze vidět, jakým způsobem je ve funkci `getUid()`, která je volána z funkce `loop()`, implementováno čtení karet. Nejprve je pomocí metody `PICC_IsNewCardPresent()` ověřeno, že se karta nachází v blízkosti RFID čtečky. Následně je metodou `PICC_ReadCardSerial()` přečteno identifikační číslo. Pokud jedna z těchto podmínek selže, což se typicky děje v případech, že uživatelská karta ke čtečce není přiložena, funkce končí s návratovou hodnotou nula. V případě úspěchu je do proměnné `uid` uloženo její identifikační číslo o velikost 4 nebo 7 bytů (dle typu karty). Jako ověření správného skenování karty zde slouží bliknutí LED diody, která se nachází na pinu `IO16` mikrokontroléru ESP32. Funkce po úspěšném přečtení vrací číslo jedna a běh programu opět pokračuje ve funkci `loop()`.

5.2.2 Knihovna ArduinoOTA

Další důležitou částí firmwaru je knihovna ArduinoOTA[4]. Tato knihovna je využita pro vzdálené nahrání programu do mikrokontroléru ESP32 přes síť Wi-Fi. Pokud například v budoucnu bude třeba upravit nebo aktualizovat firmware v zámkových modulech v provozovně sauny, není nutné jednotlivě každý modul připojovat kabelem k počítači, ale vše je možné provést bezdrátově.

[4] https://www.arduino.cc/reference/en/libraries/arduinoota/

Po správné konfiguraci v prostředí ArduinoIDE se síťový port mikrokontroléru objeví mezi sériovými COM porty. Po vybrání síťového portu lze nahrát program běžným způsobem.

Nevýhodou komunikace OTA (over-the-air) je ta, že nemůžeme při nahrávání programu sledovat dění na sériovém portu. Případná chyba, která by se v programu nacházela, pak může znemožnit bezdrátovou aktualizaci.

5.3 Informační systém

Informační systém má podobu webové aplikace, což přináší několik výhod. Zejména tu, že se taková aplikace nemusí instalovat lokálně na žádný počítač, je dostupná odkudkoliv a to i na jakémkoliv mobilním zařízení, které má přístup k internetu. Jednou z motivací pro vznik tohoto systému byl požadavek celou provozovnu kontrolovat vzdáleně, webová aplikace tedy tento požadavek splňuje.

5.3.1 Použité technologie a nástroje

Pro tvorbu takové webové aplikace bylo třeba zvolit vhodné nástroje. V této podkapitole jsou shrnuty všechny nástroje a technologie, které jsem při implementaci webové aplikace použil.

HTML

HTML (Hypertext Markup Language) je název značkovacího jazyka, který se používá při vývoji webových stránek. Je podporován téměř všemi zařízení a každá stránka obsahuje alespoň malé množství HTML kódu. Tento jazyk je vhodný pro definici významu jednotlivých bloků na vytvářené webové stránce [15].

Jazyk HTML byl navržen počátkem 90. let a od té doby se neustále vyvíjí. Poslední verzí je HTML5, který vychází ze svých předchůdců, ale obsahuje několik vylepšení, jako jsou nové doplňkové elementy, které slouží pro popis obsahu (`article`, `section`, `nav` a další). Tento jazyk také ve své poslední verzi obsahuje podporu přehrávání multimédií bez nutnosti instalace doplňkových knihoven [15].

CSS

CSS (Cascading Style Sheets) je jazyk, který se používá pro popis toho, jak jednotlivé elementy napsané v HTML vypadají. Tento jazyk nabízí řadu vlastností, například pro formátování textu, jako je velikost písma, barva, tloušťka písma a styl fontu. Dále se v jazyce CSS typicky definuje rozvržení jednotlivých HTML elementů a používá se také pro návrh responzivního designu, aby výsledná webová stránka byla korektně a čitelně zobrazena na mobilních zařízeních [15].

Jazyk CSS vznikl později než HTML. K jeho rozšíření došlo až v roce 1996. Taktéž tento jazyk prošel vývojem, přičemž poslední verzí je CSS3, která obsahuje možnosti využití spousty nových vizuálních stylů [15].

Tailwind

Tailwind[5] je tak zvaný CSS utility framework. To znamená, že na jednu stranu umožňuje pro stylování komponentů využít předdefinované třídy, ale na druhou stranu ponechává vývojáři volnost v tom, jak daná komponenta bude vypadat, na rozdíl od jiných CSS frameworků jako je například Bootstrap, který má již předdefinované celé komponenty. Přístup tohoto frameworku značně urychluje vývoj webových stránek

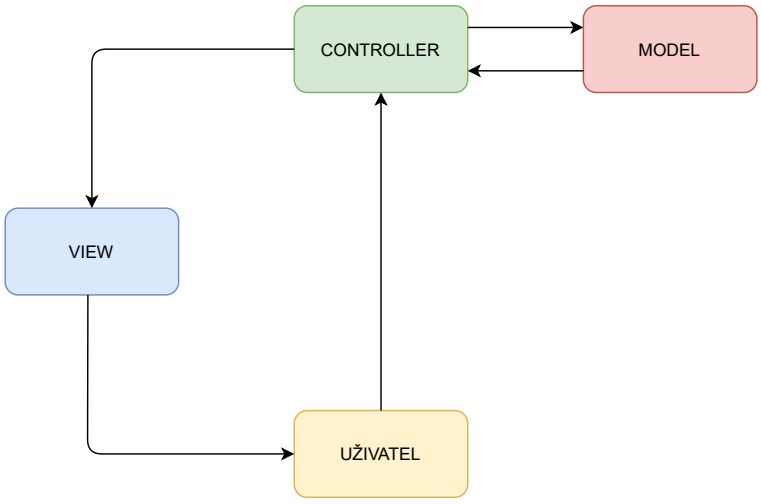
Laravel

Informační systém pro řízení přístupového systému sauny byl implementován za použití frameworku Laravel, který používá jazyk PHP. Laravel je jeden z nejoblíbenějších a nejpoužívanějších PHP frameworků na světě [29]. První beta verze Laravelu byla představena v roce 2011 a jeho tvůrcem je Taylor Otwell.

Instalace Laravelu a založení projektu je velmi jednoduché. K instalaci a aktualizaci Laravelu je potřeba nástroj Composer[6]. Jedná se o o nástroj určený pro správu knihoven a závislostí v PHP. Ovládá se prostřednictvím příkazové řádky a instalace Laravelu a následné vytvoření nového projektu lze provést příkazy. Těmi jsou `composer global require "laravel/installer"` a `laravel new projectName` [27].

Architektura frameworku Laravel je postavena na takzvaném Model-View-Controller (MVC). Jedná se návrhový vzor, který je složen ze tří základních částí [27]:

- *Model* představuje databázovou tabulku nebo její záznam.
- *View* představuje rozhraní, ve kterém jsou data vizualizovány uživateli. Nejčastěji je implementována jako stránka v jazycích HTML a CSS.
- *Controller* představuje řídící jednotku, která typicky reaguje na akce uživatele a zajišťuje změny v *modelu*, přičemž tyto změny jsou následně reflektovány v prezentaci uživateli (*view*).



Obrázek 5.4: Schéma návrhového vzoru Model–View–Controller (MVC). Inspirováno [27]

[5] https://tailwindcss.com
[6] https://getcomposer.org

Všechny části konceptu MVC lze v Laravelu vytvářet pomocí nástroje Artisan. Jedná se o rozhraní příkazové řádky, které je součástí instalace Laravelu. Obsahuje mnoho příkazů využitelných při implementaci webové aplikace. Například lze pomocí nich vytvářet nové třídy, generovat databázové tabulky a mnoho dalšího [27].

Laravel pro tvorbu frontendu (*view*) využívá šablonovací systém zvaný Blade. Tento systém umožňuje používání čistého PHP kódu v HTML šablonách. Všechny šablony Blade jsou kompilovány do PHP kódu a uloženy do mezipaměti, dokud tato šablona není změněna, což přináší výrazné snížení režie. Blade šablona používá speciální příponu `.blade.php` [3].

Hlavním úkolem webové aplikace je zpracování HTTP požadavků (2.5.2) od uživatele a vytvoření odpovědi. K tomu Laravel využívá takzvané *routes*. V souboru `routes/web.php` mohou být definovány taková webové cesty, které navštíví koncový uživatel. V případě komunikaci s API je možné cesty definovat také v `routes/api.php` [27].

```
Route::get('/home', function() {
    return view('home');
})->name('home');
```

Výpis 5.2: **Fragment kódu znázorňující definici webové cesty (*route*)**. Tento kód znamená, že v případě, že uživatel zadá do URL svého prohlížeče cestu `/home`, je na tuto podstránku přesměrován vrácením příslušné Blade šablony, v tomto případě s názvem `home`. `Route::get()` tedy představuje zpracování `GET` požadavku protokolu HTTP. V případě potřeby použití jiných metod HTTP požadavku stačí napsat její název za dvě dvojtečky, tedy například `Route::post()` či `Route::delete()`.

Pro práci s databází Laravel využívá nástroj s názvem Query Builder. Tento nástroj poskytuje pohodlné rozhraní pro tvorbu a spouštění dotazů nad databází. Funguje s řadou používaných databázových systémů, včetně MySQL, které používám v této práci. Nástroj Query Builder také nabízí ochranu před SQL injection útoky [5].

```
$clients = DB::table('clients')->get();
foreach ($clients as $client) {
    echo $client->surname;
}
```

Výpis 5.3: **Fragment kódu, který znázorňuje práci s nástrojem Query Builder**. Metoda `get()` vrací instanci třídy `Collection`, jenž slouží pro práci s datovými poli. Tato instance obsahuje výsledky dotazu nad databází. V cyklu je pak znázorněn přístup ke sloupci obsahující křestní jméno daného klienta.

Laravel taktéž obsahuje Eloquent, což je systém ORM[7] postavený na Query Builderu. Eloquent používá takzvaný ActiveRecord, což je návrhový vzor, který umožňuje vytvářet databázové tabulky reprezentovány jako třídy v jazyce PHP, jenž dědí od třídy `Model`. Každá instance takové třídy je jeden řádek (záznam) tabulky [7].

Z hlediska implementace databázové struktury ve frameworku Laravel je důležité zmínit také takzvané Migrations. Jedná se o řízení verzí databáze aplikace, která pomáhá zachovat konzistenci, které by bylo stěží dosaženo při ručním psaní MySQL příkazů. Migrace se vytváří pomocí nástroje Artisan [4].

[7] ORM – Object-Relational Mapping (objektově relační mapování – technika zajišťující konverzi dat mezi objektově orientovaným programovacím jazykem a relační databází)

Laravel Livewire

Livewire[8] je fullstackový framework pro Laravel. Umožňuje vytvářet reaktivní a dynamické aplikace při zachování šablonovacího systému Blade a Laravel na pozadí. Díky Livewire je možné vytvářet podobné aplikace jako s frameworky Vue.js a React, avšak bez jediného řádku kódu v JavaScriptu.

Pro práci s Livewire je třeba si vytvořit takzvané komponenty, tedy třídy, které dědí od třídy `Component`. Každá taková komponenta má svůj životní cyklus a je nezávislá. Tyto komponenty se vytváří pomocí Artisan příkazu `php artisan make:livewire componentName`. Vygeneruje se třída komponenty a Blade šablona.

V případě uživatelské interakce s komponenty využívá Livewire AJAX požadavky, které odesílá na server, přičemž tato komponenta je znovu vygenerována podle toho, co se v ní změnilo.

Laravel Jetstream

Laravel Jetstream[9] je rozšíření do frameworku Laravel, které poskytuje výchozí implementaci pro uživatelskou registraci a přihlašování, emailové ověření nebo API podporu. Toto rozšíření je celé navržené pomocí Tailwind a při její instalaci lze použít Livewire.

5.3.2 Serverová část

Tato sekce je věnována implementaci serverové části a to zejména způsobu vytvoření uživatele, přiřazení práv a komunikace s klientskými aplikacemi a hardwarovými zařízeními. Jak již bylo zmíněno v části popisující návrh tohoto systému, architektura serverové části dodržuje požadavky na *RESTful* aplikaci, tudíž je potřeba zejména oddělit klientskou aplikaci od serverové logiky a požadavky odesílané hardwarovými zařízeními nebo klientskými aplikacemi musí být bezstavové. V této části jsou popsány důležité nebo zajímavé detaily implementace serverové části aplikace.

Vytvoření uživatele a přiřazení rolí

Jak už bylo zmíněno v návrhu, v systému jsou evidovány dva druhy rolí. Nové uživatele typicky přidává do systému majitel. Zaměstnanec ke správě uživatelů nemá přístup. K práci s uživatelskými rolemi a právy byl využit balíček rozšíření Spatie[10].

```
use App\Models\User;
use Illuminate\Support\Facades\Hash;
use Spatie\Permission\Models\Role;

Role::create(['name' => 'owner']);
Role::create(['name' => 'employee']);
$user = User::create([
    'name' => "Michael Kinc",
    'email' => "kinc.michael@gmail.com",
    'password' => Hash::make("Heslo8956"),
]);
```

[8] https://laravel-livewire.com
[9] https://jetstream.laravel.com/2.x/introduction.html
[10] https://spatie.be/docs/laravel-permission/v4/introduction

```
12 $user->assignRole('owner');
```

Výpis 5.4: **Fragment kódu znázorňující způsob vytvoření uživatele a rolí pomocí balíčku Spatie.** K vytvoření uživatele je použita metoda `create()`, ve které jsou specifikovány hodnoty sloupců nově vytvářeného záznamu. Uživatelské heslo se do databáze ukládá bezpečně zašifrované pomocí hašovací funkce Bcrypt. Nemůže tedy dojít ke zneužití hesel uložených v databázi. Přiřazení role k instanci uživatele se provádí pomocí funkce `assignRole()`, ve které je specifikován název přiřazované role.

Přijímání a zpracování požadavků

Pokud uživatel nebo hardwarové zařízení komunikuje se serverem, využívá k tomu API. Ke zpracování webových cest jsou využity Routes popsané v kapitole 5.3.1. Pokud uživatel pošle požadavek (typicky zadáním požadované webové cesty do řádku URL ve svém prohlížeči), je vyvolána Route a uživateli se zobrazí požadovaná stránka. Cesty, které využívá uživatel v rámci práce se systémem, jsou definovány v souboru `routes/web.php`.

Kromě uživatelů se serverem komunikují také hardwarová zařízení. Modul čtečky klientských karet na recepci odesílá na server POST dotazy s UID karty. Tento záznam je uložen do tabulky záznamů o skenování a je připraven na zpracování uživatelem. Zámkové moduly pak odesílají GET dotazy s UID karty, kterou uživatel přiložil ke čtečce. V tomto dotazu mikrokontrolér řídící zámkový modul očekává odpověď ze serveru, ve které bude řečeno, zda karta se zaslaným UID je oprávněna otevřít danou skříňku nebo dveře. Tyto cesty jsou definovány v souboru `routes/api.php` a parametry z požadavků jsou předány ke zpracování kontrolerům. Odpověď ze serveru pak obsahuje kromě HTTP hlaviček také tělo s informacemi ve formátu JSON.

Ve výpisu je 5.5 ukázka validního HTTP dotazu, který posílá serveru dotaz, zda karta s UID 5000 je oprávněna otevřít dveře s identifikačním číslem jedna. Ve výpisu 5.6 je pak odpověď na tento dotaz, který říká, že daná karta je oprávněna otevřít dveře číslo jedna.

```
GET /api/open/door?uid=5000&door_id=1 HTTP/1.1
Host: 192.168.1.9:8000
```

Výpis 5.5: Příklad validního HTTP dotazu při komunikaci se serverem

```
HTTP/1.1 200 OK
Host: 192.168.1.9:8000
Date: Fri, 30 Apr 2021 19:49:14 GMT
Connection: close
X-Powered-By: PHP/8.0.2
Cache-Control: no-cache, private
Content-Type: application/json
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 58
Access-Control-Allow-Origin: *

{
    "card_id": "5000",
```

```
        "door_id": "1",
        "open": "true"
    }
```

Výpis 5.6: HTTP odpověď ze serveru.

5.3.3 Databáze

V této sekci je popsán způsob implementace databázové části systému a to včetně způsobu mapování tabulek pomocí systému ORM. Klient v případě požadavku obdrží data z této databáze. Tato data jsou v databázi perzistentně uložena a slouží pro správu celého informačního systému.

Pro implementaci této části byla využita databáze *MySQL*, která je podporována frameworkem Laravel. Tato sekce obsahuje dva fragmenty kódu, které znázorňují způsob implementace databázových tabulek ve frameworku Laravel.

```
1  class Client extends Model
2  {
3      use HasFactory;
4      protected $guarded = [];
5
6      public function cards() {
7          $this->hasMany(Card::class);
8      }
9
10     public static function search($search) {
11         return empty($search) ? static::query()
12             : static::where('email', 'like', '%'.$search.'%')
13                 ->orWhere('name', 'like', '%'.$search.'%')
14                 ->orWhere('surname', 'like', '%'.$search.'%');
15     }
16 }
```

Výpis 5.7: **Fragment kódu znázorňující definování modelu klienta a vazby 1:N (Klient:Karta).** Je patrné, že tato třída dědí od třídy `Model` a obsahuje dvě metody. První slouží pro definici vazby mezi entitními množinami v databázi. Druhá metoda je implementací vyhledávání v tabulce.

```
1  class CreateClientsTable extends Migration
2  {
3      /* Run the migrations. */
4
5      public function up()
6      {
7          Schema::create('clients', function (Blueprint $table) {
8              $table->bigIncrements('id');
9              $table->timestamps();
10             $table->string('name');
11             $table->string('surname');
12             $table->string('email')->unique();
13             $table->string('phone');
14
15             $table->enum('type', ['5_entries', '10_entries', 'season']);
16             $table->integer('entries')->nullable();
17         });
```

```
18      }
19      /* Reverse the migrations. */
20
21      public function down()
22      {
23          Schema::dropIfExists('clients');
24      }
25  }
```

Výpis 5.8: **Fragment kódu znázorňující způsob vytvoření tabulky klientů a definici sloupců této tabulky pomocí migrace.** Tato třída dědí od třídy `Migrtation` a obsahuje dvě metody. První slouží pro vytvoření databázové tabulky a specifikace jejích sloupců. Druhá pak slouží na vymazání tabulky z databáze.

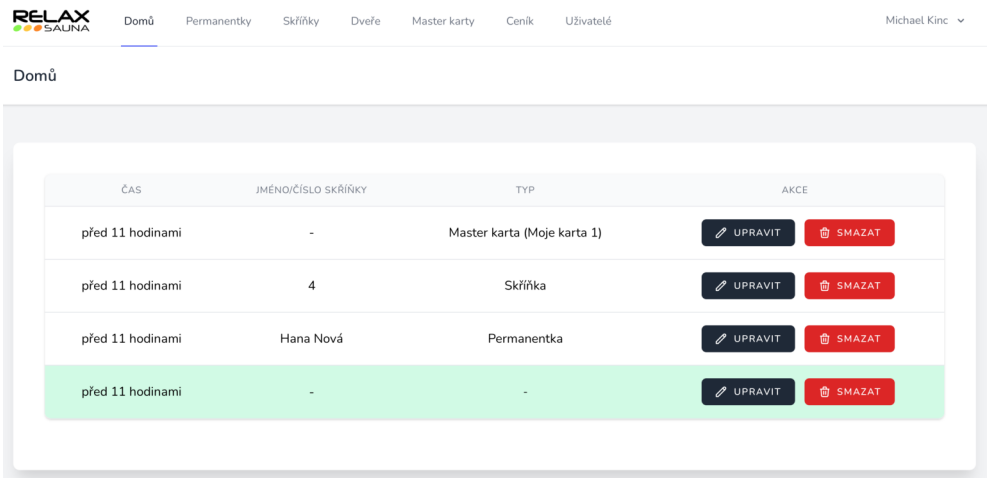
5.3.4 Uživatelské rozhraní

Uživatelské rozhraní zpracovává vstupy od uživatele a prezentuje výsledky těchto vstupů. Slouží tedy k interakci mezi uživatelem a systémem. Tato sekce popisuje způsob implementace uživatelského rozhraní a znázorňuje, jak její jednotlivé části vypadají.

Kostra šablony tohoto *dashboardu* a vzhled některých komponent jsou vygenerovány rozšířením Laravel Jetstream. Vygenerované i vlastní části systému jsou vytvořeny pomocí značkovacího jazyka HTML a CSS frameworku Tailwind.

Celé uživatelské rozhraní je rozděleno do několika sekcí podle druhu svého obsahu. Některé sekce jsou přístupné pouze uživatelům s rolí *Majitel* a to zejména sekce *Ceník*, *Dveře* a *Uživatelé*.

Na obrázku 5.5 lze vidět úvodní stránku informačního systému, která se uživateli zobrazí ihned po přihlášení. Tato stránka obsahuje tabulku se záznamy o skenování karet čtečkou na recepci. Zelený řádek tabulky značí novou kartu, která v systému ještě není evidována, tudíž ji lze vytvořit jako permanentku, master kartu nebo ji přiřadit jako klíč ke skříňce.

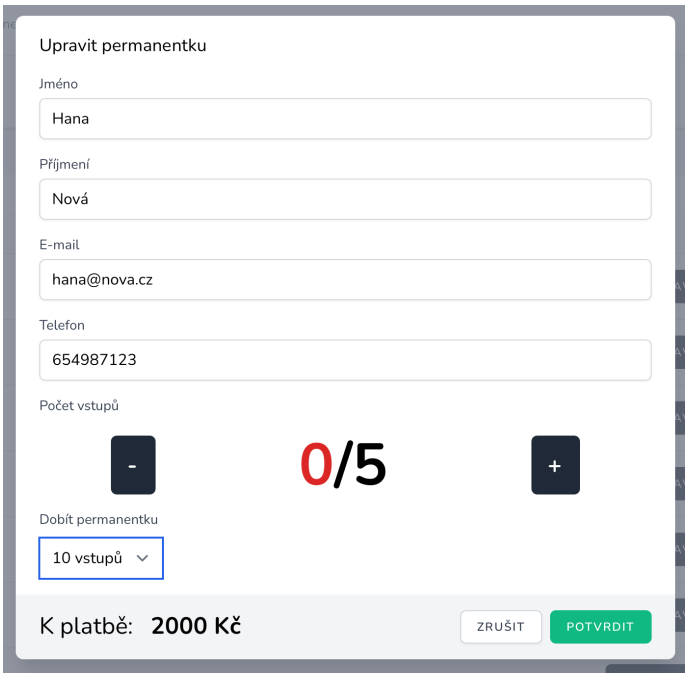


Obrázek 5.5: Domovská stránka informačního systému se záznamy o skenování

Na obrázku 5.6 lze vidět formulář zobrazený v modálním okně, který slouží k editaci permanentky klienta. Je zde znázorněna situace, kdy klient už na své kartě nemá žádný volný vstup, tudíž je uživateli nabídnuta možnost tuto kartu klientovi dobít. Při zvolení požadovaného typu permanentky se zobrazuje cena, kterou za tuto položku má klient zapla-

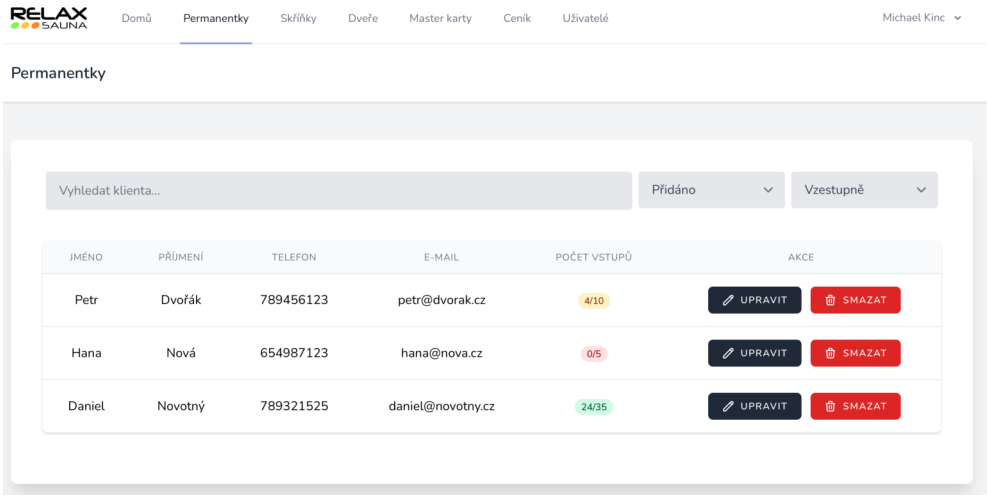
tit. Jednotlivé položky ceníku jsou uloženy v databázi a může je upravovat majitel v sekci *Ceník*.

Všechny formuláře v systému obsahují také validátor. Nemůže se tedy stát, že uživatel při vyplňování formuláře napíše email nebo telefonní číslo ve špatném formátu nebo že nějakou položku tohoto formuláře ponechá prázdnou.



Obrázek 5.6: Ukázka dobíjení permanentky

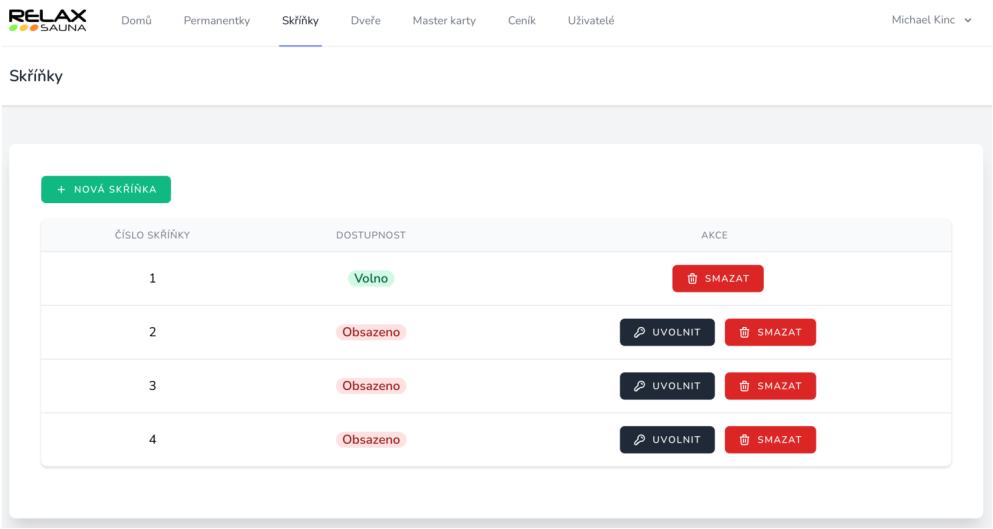
Na obrázku 5.7 lze vidět sekci s tabulkou, která zobrazuje všechny permanentky klientů evidované v systému. Tabulku lze seřadit podle data přidání, jména, příjmení a emailu a to buď vzestupně nebo sestupně a je možné také v tabulce konkrétního klienta vyhledat. Po kliknutí na tlačítko *Upravit* se zobrazí modální okno z obrázku 5.6.



Obrázek 5.7: Tabulka obsahující všechny evidované klienty (permanentky)

Obrázek 5.8 znázorňuje sekci s tabulkou skříněk evidovaných v systému. Lze z ní jednoduše vyčíst její číslo a obsazenost. Také lze v této sekci do systému přidat novou skříňku, případně ji smazat. V řádcích obsazených skříněk (ty, které mají přiřazený RFID náramek) se také nachází tlačíko, pomocí kterého je možné skříňku uvolnit. To znamená, že se od této skříňky odpáruje přiřazený RFID náramek. Toto tlačítko lze použít například v případě, že klient ztratí svůj náramek a tím pádem jej nelze odpárovat načtením na recepci.

Podobný princip je uplatněn i v případě master karet. Ztráta této karty by znamenala vysoké bezpečnostní riziko, přičemž by umožňovala potenciálnímu nálezci prakticky volný přístup do provozovny. Každá taková karta je v systému označena popisem a lze ji jednoduše v případě ztráty v tabulce těchto karet vyhledat a vymazat.



Obrázek 5.8: Tabulka obsahující všechny skříňky evidované v systému

Kapitola 6

Testování

Testování je poslední fáze vývoje aplikace před nasazením. Tato fáze nesmí být podceněna. Před nasazením do reálného provozu by měl celý systém projít minimálně jedním kolem testovaní, na kterém se bude podílet cílová skupina uživatelů, kteří se systémem budou pracovat. Testování se týká také hardwarových zařízení. Ty musí být funkční za každých okolností a bez výpadků.

Informační systém pro saunu je dostupný na adrese `http://relax.michaelkinc.net`. Na přihlašovací stránku do administrátorské části se lze dostat kliknutím na tlačítko *Administrace* v zápatí stránky. Přístupové údaje do systému se nachází v tabulce 6.1.

Role	Uživatelské jméno	Heslo
Majitel	majitel@test.cz	123456
Zaměstnanec	zamestnanec@test.cz	654321

Tabulka 6.1: Tabulka přihlašovacích údajů do systému

6.1 Způsob testování

Testování bylo rozděleno na dvě části. V první části proběhlo testování zaměřené pouze na hardwarové moduly. V druhé části proběhlo testování výsledného řešení jako celku, včetně testování informačního systému majitelem sauny.

6.1.1 Koncová zařízení

Koncová zařízení jsou navržena pro práci výhradně v on-line režimu prostřednictvím HTTP komunikace se serverem. Zde je potřeba myslet na to, co se v případě výpadku či dočasné nedostupnosti Wi-Fi připojení stane. V případě výpadku fungují zámky dále v off-line režimu. Ke každé skříňce jsou napevno přiřazeny záložní čipové náramky, které otevřou skříňku i při takovém výpadku připojení. Po obnovení připojení je modul opět schopen dotazování se serveru o informace.

Druhým aspektem, který je z hlediska testování koncových hardwarových zařízení nutný vzít v úvahu, je výpadek elektrického proudu. Elektromagnetické zámky jsou ve výchozím stavu zamčeny, to znamená, že v případě výpadku proudu je taková skříňka nedostupná. Při instalaci zámkových modulů proto bude do provozovny nainstalován záložní zdroj, který v případě výpadku elektrického proudu poskytne rezervu v řádu několika desítek minut.

Kompletní zapojení bylo testováno v týdenním zátěžovém provozu, kde s koncovými zařízeními byla čas od času provedena interakce. Systém fungoval bez problému po celou dobu provozu.

6.1.2 Kompletní systém

Celý systém včetně webové aplikace byl otestován majitelem provozovny. Z důvodu protipandemických restrikcí v době vzniku této práce však nebylo možné nasadit a otestovat systém v reálném provozu. Dále byl systém předložen členům rodiny a to z důvodu rozšíření uživatelské testovací základny.

Pro testování byla vytvořena sada úkonů, které měli uživatelé v systému provést. Tento proces jsem sledoval a následně vyhodnotil, co uživatelům šlo bez problému, kde se naopak pozastavili. Tyto úkony jsou tvořeny několika položkami, kterými jsou:

- Přihlásit se do systému,
- Provést skenování novou kartou,
- Vytvořit novou permanentku,
- Přiřadit čipový náramek k dané skříňce,
- Zrušit přiřazení náramku ke skříňce,
- Manuálně (bez skenování) upravit kontaktní údaje klienta na jeho permanentce,
- Dobít permanentku daným počtem vstupů,
- Přidat do systému novou skříňku a dveře,
- Vytvořit master kartu.

6.2 Získaná zpětná vazba

Každý uživatel postupně prošel všechny zadané body a nikdo s orientací v systému neměl žádné zásadní problémy. Na základě zpětné vazby však ve finální verzi systému bylo provedeno několik změn, se kterými můj původní návrh nepočítal:

- Při manuální editaci klientských karet byla poněkud nepřehledná orientace v tabulce a to zejména v situacích, kdy je v systému velké množství takových permanentek. Tabulka byla tedy přepracována, aby se dala seřadit dle jednotlivých sloupců podle abecedy. Také bylo implementováno vyhledávání v tabulce, což výrazně urychluje práci při správě klientských karet.
- Při dobíjení klientských karet se zobrazuje cena v závislosti na druhu dobíjené permanentky. Zaměstnanec provozovny má tedy okamžitý přehled o tom, kolik má zákazník zaplatit. Tyto ceny jsou editovatelné majitelem sauny v sekci *Ceník*. Na webových stránkách sauny se také v závislosti na změně ceny dynamicky mění tabulka ceníku.

6.3 Další možný vývoj

Otestovaný systém je připraven k používání v ostrém provozu. Potenciál celé práce z hlediska použitelnosti a rozšířitelnosti je dle mého názoru daleko širší. Systém mám v plánu v nejbližší době rozšířit o rezervační systém s registračním formulářem na webu. Sauna má omezenou kapacitu osob a tři časové turnusy, a proto je třeba si vést záznamy o rezervacích, které v současné době fungují zejména prostřednictvím mobilního telefonu a e-mailu. Cílem je tedy vnést do rezervací řád a rozšířit tímto způsobem vytvořený systém.

Vytápění sauny na požadovanou teplotu je třeba zapnout přibližně hodinu a půl před příchodem zákazníků. Velmi zajímavým rozšířením by bylo chytré ovládání vytápění sauny, kdy by se dalo vytápění zapnout prostřednictvím tlačítka ve webové aplikaci. Také by zde vznikala možnost vzdálené regulace teploty.

Kapitola 7

Závěr

V této práci byly nastudovány a popsány základní informace o bezdrátovém přenosu dat, mikrokontrolérech a sumarizovány informace o již existujících řešeních modulárních přístupových systémů. Cílem této práce bylo na základě nabytých znalostí na míru navrhnout vlastní systém přístupového systému pro existující provozovnu sauny v Brně. Tento cíl byl splněn, celý systém je funkční a může být nasazen do reálného provozu. Seznámil jsem se také s nejrůznějšími technologiemi používanými k návrhu hardwarových zařízení, k programování mikrokontrolérů a ke tvorbě webových aplikací.

Celý systém vytvořený v rámci této práce byl vyvíjen takovým způsobem, aby mohl být nasazen v cloudu a majitel provozovny mohl provozovnu kontrolovat vzdáleně. Prostřednictvím webové aplikace je možné vzdáleně spravovat permanentky a kontrolovat aktuální obsazenost skříněk. Taktéž systém bude poskytovat komfort zákazníkům sauny, kteří doposud museli při svém pobytu opatrovat klíče, které se nedaly komfortně umístit na zápěstí. Celý systém je také levný, přičemž cena jednoho modulu použitého v provozovně se pohybuje v řádech nižších stovek korun.

Tuto práci bych rád rozvíjel i v budoucnu, neboť se domnívám, že tento systém (nebo nějaká jeho část) by mohl být využitelný i v jiných oblastech a provozovnách, kterými mohou být například fitness centra, aquaparky nebo sídla firem, které potřebují elektronicky kontrolovat přístup do budovy a pohyb svých zaměstnanců. K mým nejbližším cílům rozvoje této práce patří tvorba rezervačního systému a nasazení celého systému do ostrého provozu, k čemuž bohužel v rámci této práce nedošlo z důvodu uzavřených provozoven.

Práce mi dala hodně zkušeností z mnoha odvětví informačních technologií. Zejména jsem si vyzkoušel navrhnout a technicky realizovat funkční hardwarové moduly, přičemž s návrhem, výrobou a osazením desek plošných spojů jsem se dosud nesetkal.

Literatura

[1] *About MIFARE* [online]. [cit. 2021-04-20]. Dostupné z: https://www.mifare.net/en/about-mifare/.

[2] *BC817: NPN General Purpose Transistor* [online]. [cit. 2021-03-16]. Dostupné z: https://www.gme.cz/data/attachments/dsh.912-032.2.pdf.

[3] *Blade Templates* [online]. [cit. 2021-04-20]. Dostupné z: https://laravel.com/docs/8.x/blade.

[4] *Database: Migrations* [online]. [cit. 2021-04-20]. Dostupné z: https://laravel.com/docs/8.x/migrations.

[5] *Database: Query Builder* [online]. [cit. 2021-04-20]. Dostupné z: https://laravel.com/docs/8.x/queries.

[6] *Desktop Platform Form Factors Power Supply Design Guide* [online]. [cit. 2021-04-28]. Dostupné z: https://www.intel.com/content/dam/www/public/us/en/documents/guides/power-supply-design-guide-june.pdf.

[7] *Eloquent: Getting Started* [online]. [cit. 2021-04-20]. Dostupné z: https://laravel.com/docs/8.x/eloquent.

[8] *LE00AB/C Series: Very Low Drop Voltage Regulators With Inhibit* [online]. [cit. 2021-03-16]. Dostupné z: https://www.gme.cz/data/attachments/dsh.934-021.1.pdf.

[9] *MFRC522: Standard performance MIFARE and NTAG frontend* [online]. [cit. 2021-03-16]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

[10] *Raspberry Pi 4 Model B Datasheet* [online]. [cit. 2021-04-15]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf.

[11] *SRD Series: Subminature High Power Relay* [online]. [cit. 2021-03-16]. Dostupné z: https://www.hadex.cz/spec/l597a.pdf.

[12] *Wemos Relay Shield Documentation* [online]. [cit. 2021-03-16]. Dostupné z: https://www.wemos.cc/en/latest/d1_mini_shield/relay.html.

[13] Bidlo, M. *Principy sériové komunikace, sériová komunikační rozhraní* [online]. [cit. 2021-03-04]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIMP-IT%2Flectures%2F04-IMP-seriova_kom.pdf&cid=13324.

[14] Campbell, S. *Basics of UART Communication* [online]. [cit. 2021-03-02]. Dostupné z: https://www.circuitbasics.com/basics-uart-communication/.

[15] Casto, E. a Hyslop, B. *HTML5 a CSS3*. 1. vyd. Computer Press, 2013. ISBN 978-80-251-3733-8.

[16] Colban, N. *Kolban's Book on ESP32*. Leanpub, 2017.

[17] Finkenzeller, K. *RFID Handbook*. 3. vyd. Wiley, 2007. ISBN 978-0-470-69506-7.

[18] Grässle, P. et al. *UML 2.0 in Action: A Project-Based Tutorial*. 1. vyd. Packt Publishing, 2005. ISBN 1-904811-55-8.

[19] Hunt, V. D. et al. *RFID: A Guide to Radio Frequency Identification*. Wiley-Interscience, 2007. ISBN 978-0-470-10764-5.

[20] Kurose, J. F. a Ross, K. W. *Computer Networking: A Top-Down Approach*. 7. vyd. Pearson, 2017. ISBN 978-0-13-359414-0.

[21] Labiod, H. et al. *Wi-Fi, Bluetooth, ZigBee and WiMAX*. Springer, 2007. ISBN 978-1-4020-5396-2.

[22] Leens, F. An introduction to I2C and SPI protocols. *IEEE Instrumentation Measurement Magazine*. 2009, sv. 12, č. 1, s. 8–13. DOI: 10.1109/MIM.2009.4762946.

[23] Láníček, R. *Elektronika, obvody - součástky - děje*. Ben - Technická literatura, 1998. ISBN 80-86056-25-2.

[24] Massé, M. *REST API Design Rulebook*. 1. vyd. O'Reilly Media, Inc., 2012. ISBN 978-1-449-31050-9.

[25] Matoušek, P. *Síťové služby a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1.

[26] Poole, I. *What is WiFi: IEEE 802.11* [online]. [cit. 2021-01-18]. Dostupné z: https://www.electronics-notes.com/articles/connectivity/wifi-ieee-802-11/what-is-wifi.php.

[27] Stauffer, M. *Laravel: Up & Running*. 2. vyd. O'Reilly Media, Inc., 2019. ISBN 978-1-492-04121-4.

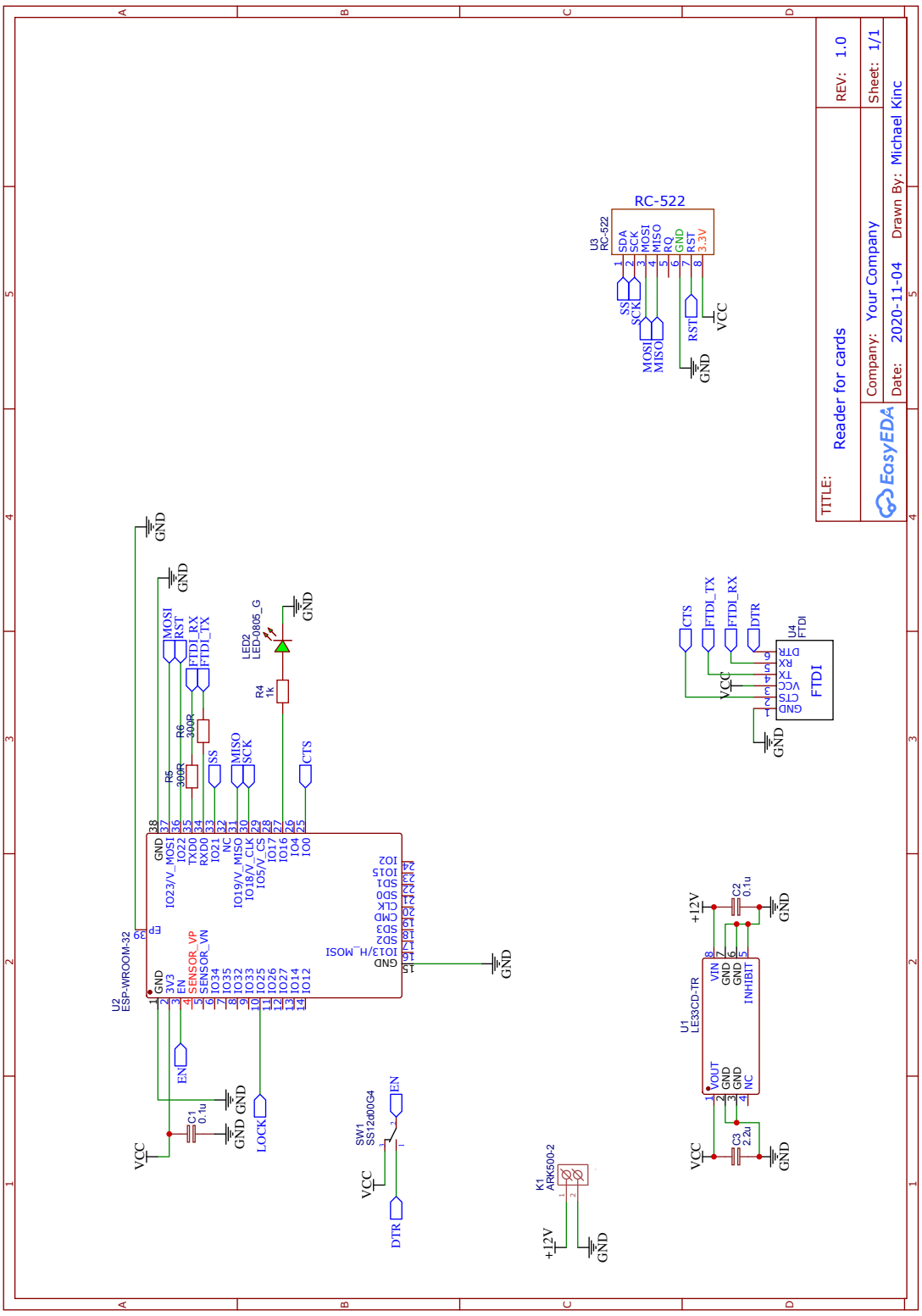
[28] Townsend, K. et al. *Getting Started with Bluetooth Low Energy*. 1. vyd. O'Reilly Media, Inc., 2014. ISBN 978-1-491-94951-1.

[29] Walley, K. *Top 6 Most Used PHP Frameworks for Web Development 2020* [online]. [cit. 2021-04-18]. Dostupné z: https://dev.to/websitedesignnj/top-6-most-used-php-frameworks-for-web-development-2020-46o5.

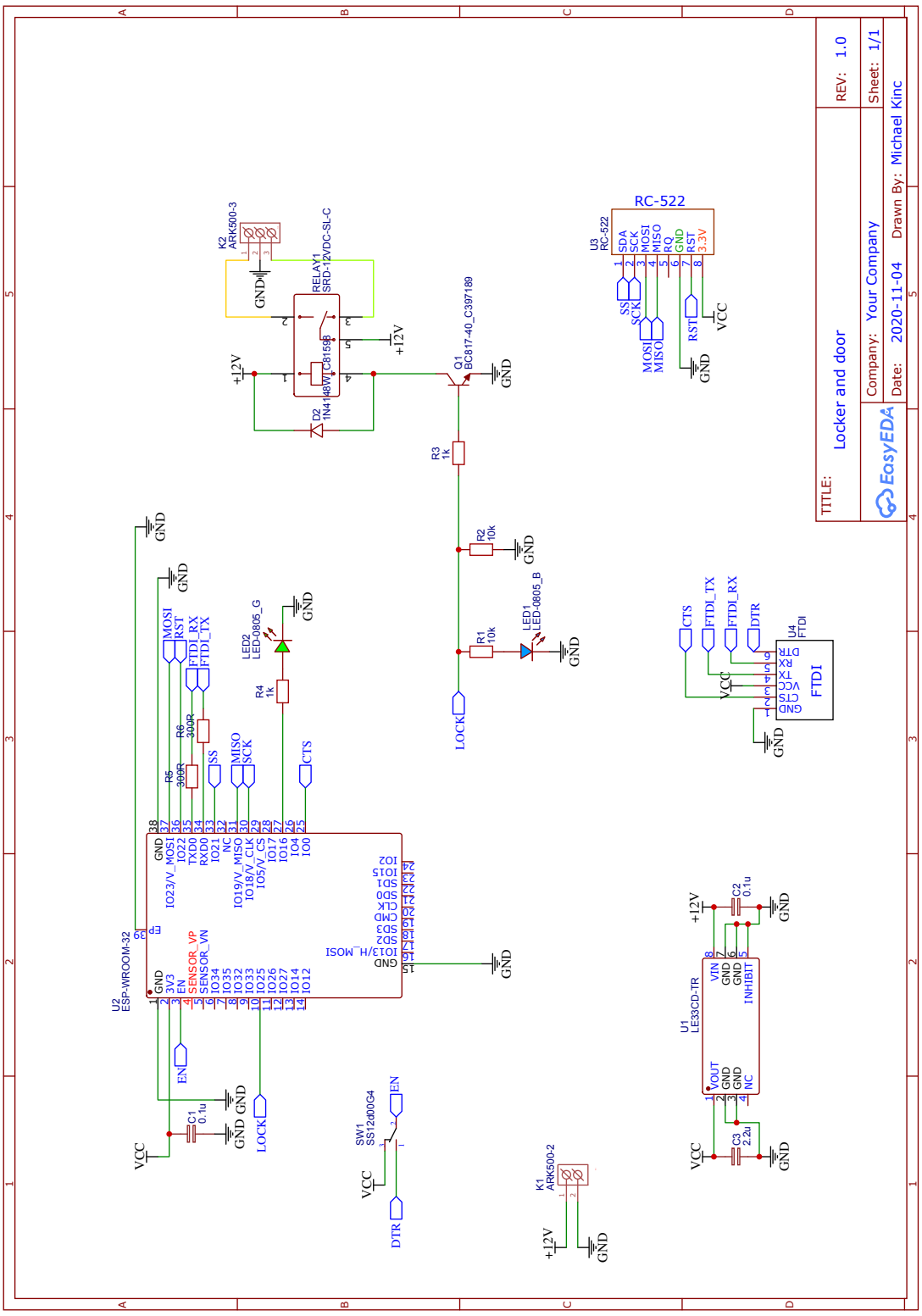
[30] Yiu, J. *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors*. 3. vyd. Newner, 2014. ISBN 978-0-12-408082-9.

Příloha A

Schémata obvodového zapojení



Obrázek A.1: Schéma zapojení čtečky klientských karet



Obrázek A.2: Schéma zapojení čtečky klientských karet

Příloha B

Obsah přiloženého DVD

- `/`
  - `README.txt` ..... podrobnosti a návod k instalaci
  - `docs`
    - `thesis` ..... technická zpráva a její zdrojové soubory
    - `diagrams` ..... schémata zapojení koncových zařízení
    - `pcb_layouts` ..... schémata desek plošných spojů
  - `videos` ..... demonstrační videa
  - `src`
    - `mcu` ..... zdrojové kódy pro mikrokontroléry
    - `laravel_app` ..... zdrojové kódy webové aplikace v Laravelu
    - `diagrams` ..... zdrojové kódy schémat zapojení koncových zařízení
    - `pcb_layouts` ..... zdrojové kódy schémat desek plošných spojů
    - `3d_print` ..... 3D model využitý pro tisk krabiček