Návrh přístupového systému pomocí platformy Arduino

Richard Konečný

Bakalářská práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav bezpečnostního inženýrství

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Richard Konečný**
Osobní číslo:	**A17094**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Forma studia:	**Prezenční**
Téma práce:	**Návrh přístupového systému pomocí platformy Arduino**
Téma práce anglicky:	**The Design of an Access System Using the Arduino Platform**

Zásady pro vypracování

1. Vypracujte literární rešerši zaměřenou na systém kontroly vstupu.
2. Specifikujte vývojové platformy založené na Arduino.
3. Navrhněte a realizujte prototyp systému kontroly vstupu.
4. Vytvořte programové vybavení pro přístupový systém.
5. Ověřte funkčnost navrženého systému.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-93-8.
2. LUKÁŠ, Luděk. Bezpečnostní technologie, systémy a management I. Zlín: VeRBuM, 2011. ISBN 978-80-87500-05-7.
3. ŠČUREK, Radomír. Biometrické metody identifikace osob v bezpečnostní praxi. VŠB TU Otrava, 2008.
4. KOLAJA, Martin. Využití přístupových systémů v průmyslu komerční bezpečnosti. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006, 67 s. Dostupné také z: http://hdl.handle.net/10563/806. Tomas Bata University in Zlín. Faculty of Applied Informatics, Ústav elektrotechniky a měření. Vedoucí práce Kindl, Jiří.
5. MANN, Burkhard. C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy – linkery, práce s ATMEL AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky. Praha: BEN – technická literatura, 2003. ?C & praxe. ISBN 80-7300-077-6.

Vedoucí bakalářské práce: **Ing. Stanislav Kovář**
Ústav bezpečnostního inženýrství

Datum zadání bakalářské práce: **17. prosince 2019**
Termín odevzdání bakalářské práce: **25. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

Ing. Jan Valouch, Ph.D.
ředitel ústavu

- Ve Zlíně dne 7. prosince 2019

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 3. srpna 2020

Richard Konečný, v. r.
podpis diplomanta

ABSTRAKT

Práce se věnuje návrhu a realizaci přístupového systému s využitím vývojové platformy Arduino. Teoretická část práce obsahuje terminologii a princip přístupových systémů, stejně jako seznámení s platformou Arduino. Praktická část zahrnuje návrh a realizaci přístupového systému, včetně popisu činnosti jednotlivých komponent. Přístupový systém nabízí verifikaci uživatele pomocí karty, přístupového kódu a biometrického otisku. Součástí práce je schéma zapojení a zdrojové kódy nezbytné pro činnost systému.

Klíčová slova: elektronický systém kontroly vstupu, autentizace, přístupový systém, platforma Arduino, tag, čtečka otisků prstů, RFID čtečka

ABSTRACT

The work is devoted to the design and implementation of the access system using the Arduino development platform. The theoretical part contains terminology and principle of access systems, as well as introduction to the Arduino platform. The practical part includes the design and implementation of the access system, including the description of the operation of individual components. The access system offers user authentication by card, access code and biometric fingerprint. Part of the work is the wiring diagram and source codes necessary for the system operation.

Keywords: electronic access control system, authentication, access system, Arduino platform, tag, fingerprint reader, RFID reader

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Stanislavu Kovářovi za vedení, užitečné rady a připomínky během vypracování bakalářské práce.

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

OBSAH

ÚVOD

Přístupové systémy patří mezi základní prvky ochrany objektů a jejich využití v nejprimitivnější formě se datuje k počátkům lidské společnosti. Lidé využívali své vynalézavosti k zabezpečení svého majetku, ať už se jednalo o domy, statky, či dobytek. Postupem času lidé využili vědy a vývoje technologií, čímž zdokonalili své vynálezy do současné podoby. Jednalo se o generační procesy předávání znalostí a zkušeností, jež vedly ke zefektivnění přístupových systémů a zvýšení úrovně zabezpečení.

V posledních letech lze zaznamenat rapidní nárust přístupových systémů napříč společností od menších podniků po korporáty. Účelem těchto systémů je zabránění přístupu neoprávněných osob do objektu. Elektronické systémy kontroly vstupu se mohou kombinovat s ostatními systémy např. docházkové systémy nebo s výdejem stravy. Při spojení s docházkovými systémy mluvíme o tzv. integrovaném identifikačním systému kontroly vstupu. Autentizace uživatelů nejčastěji probíhá pomocí RFID čipu nebo karty.

Nástup biometrických systémů značně zefektivnil procesy identifikace a verifikace uživatelů. Doposud byla potřeba znalost konkrétní informace, např. hesla či přítomnost identifikačního předmětu, např. token, avšak biometrické systémy tyto limity odstranily. Ověřování probíhá s využitím neměnných fyziologických nebo behaviorálních vlastností. Značné rozšíření biometriky lze nalézt zejména v smartphonech nebo noteboocích, kde se jedná nejčastěji o otisk prstu nebo rozpoznání obličeje.

Jelikož jde technologický vývoj stále dopředu, není složité svépomoci vyrobit vlastní přístupový systém. Jednou z možností, jak levně sestavit přístupový systém je využití platformy Arduino. Uvedená platforma je kompatibilní s širokým spektrem komponent dostupných na českém i zahraničním trhu.

I. TEORETICKÁ ČÁST

1 ELEKTRONICKÉ SYSTÉMY KONTROLY VSTUPU

Hlavním účelem elektronických systémů kontroly vstupu je zabránění přístupu neoprávněné osobě do objektu, nebo zamezení přístupu do střežených prostor s důležitými informacemi nebo daty. Tyto systémy poskytují informace o nepovoleném vstupu, počtu osob, které se právě nacházejí v určité zóně a celkové sledování návštěvnosti objektu. [1]

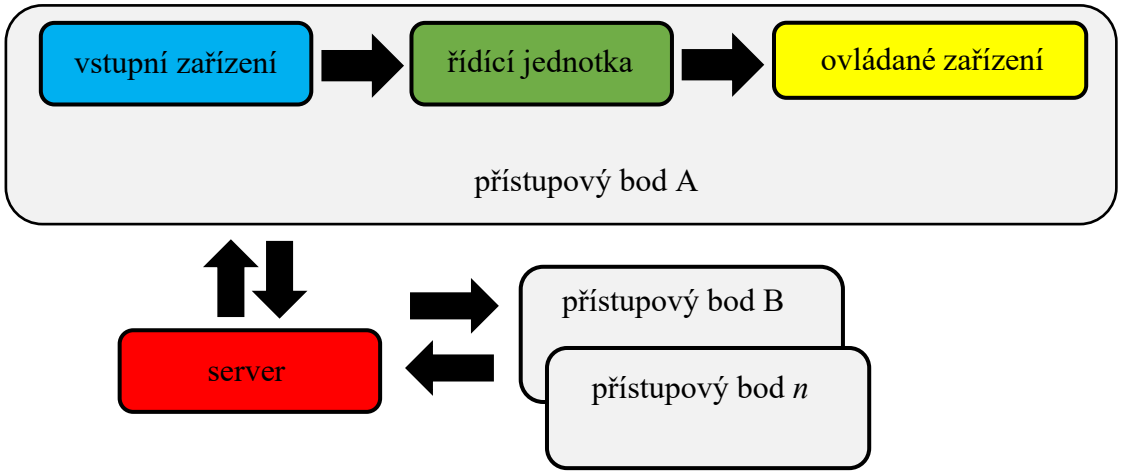
Dělení podle funkce:

- **Elektronické systémy kontroly vstupu** – podle oprávnění uživatele stanovují povolení vstupu k cennostem, informacím nebo do chráněný místností. [1]
- **Docházkové systémy** – starají se o ověření přístupu do objektu, zaznamenávají informace o času a důvodu průchodu. Získané informace poslouží pro výpočet mzdy. [1]

Při spojení systémů kontroly vstupu s docházkovými systémy lze mluvit o tzv. integrovaném identifikačním systému kontroly vstupu. [1]

Elektronické systémy kontroly vstupu (dále jen ESKV) jsou definovány normou ČSN EN 60839: Poplachové a elektronické systémy část 11-1: Elektronické systémy kontroly vstupu. Tato norma vešla v platnost 11.6.2016. Byla vytvořena jako náhrada normy ČSN EN 50133-1, která platila od března 2001 a platnost skončila po zavedení nové normy. [2]

Blokové schéma znázorňuje nejjednodušší podobu přístupového systému. Prvním blokem je vstupní zařízení, které bývá tvořeno např. Radio Frequency Identification (dále jen RFID) čtečkou, klávesnicí nebo čtečkou otisků prstů. Druhým důležitým blokem je řídící jednotka, primární úkol řídící jednotky spočívá v příjmu vstupních informací a porovnání s informacemi uloženými v paměti. Pokud se informace shodují, tak řídící jednotka aktivuje ovládané zařízení. Ovládané zařízení lze vytvořit pomocí např. závory, turniketu nebo brány. [3], [4]



Obr. 1. Blokové schéma přístupového systému.

Celý přístupový bod je schopen komunikovat např. se serverem, na kterém jsou uloženy data. K přenosu dat mezi serverem a přístupovým bodem lze využít např. přenosovou linku RS485, následně pak lze mluvit o vytvoření přístupového sytému. Data uložená na serveru se většinou každý den aktualizují, čímž dojde i k obnovení dat uložených na řídící jednotce. Ve velkých budovách se vyskytuje jeden server, který zajišťuje komunikaci s přístupovými body.

Tyto přístupové body bývají tvořeny např. vstupní zámek, vstupní brána, nebo třeba turnikety. Účel kontroly vstupu spočívá v určení a evidenci: [4], [5]

- **KDO** – jaká osoba se dostane do objektu,
- **KAM** – do kterých místností, nebo úseků se osoba dostane,
- **KDY** – kdy je osobě povoleno se pohybovat po objektu.

Způsob vkládání dat do řídící jednotky:

- do programu v PC přes Universal Serial Bus (dále jen USB) kabel,
- pomocí klávesnice u přístupového bodu,
- s využitím rádiových vln.

Autentizace znamená ověření, zda je uživatel tím, za koho se vydává. Porovnání se nejčastěji provádí s nějakou databází údajů. V databázi jsou uloženy hesla uživatelů, jedinečné ID karet nebo biometrický údaj. Databáze bývá uložena na serveru. [3] Publikace [3] rovněž uvádí několik způsobů verifikace pomocí:

- **znalosti** – heslo, pin,
- **předmětu** – identifikační karta, přívěšek,
- **sám sebou** – biometrický otisk, geometrie ruky. [4]

Na základě počtu využitých metod, lze rozlišit:

- **jednofázovou autentizaci** – pro přístup je využita pouze jedna metoda, např. heslo,
- **dvoufázovou autentizaci** – přístup je založen na využití dvou metod, např. heslo a čip,
- **třífázovou autentizaci** – pro přístup se využívají tři metody, např. heslo, čip a otisk prstu. [4]

1.1 Autentizace heslem

Následující metoda souvisí se znalostí informace. Při přístupu do daného objektu je uživatel vyzván, aby zadal ověřovací údaje, např. pin, nebo heslo. Pin lze zadávat nejčastěji pomocí klávesnice nebo dotykového displeje. Následně je provedena kontrola s databází, zda se zadaný pin, nebo heslo shoduje s informací uloženou v databázi. Pokud je zadaná informace správná, tak je uživateli povolen přístup do objektu. V případě zadání špatné informace je uživateli přístup zamítnut. Velké rozšíření lze zaznamenat v osobních počítačích při přístupu do systému nebo na webové servery. Síla hesla se odvíjí od použitých znaků, zda jsou použity malá písmena, velká písmena, číslice, speciální znaky a zda heslo obsahuje nějakou logickou návaznost nebo souvislost. Zajištění bezpečného hesla musí být podmíněno, pravidelnou změnou zadaného hesla. Nevýhoda této metody spočívá v zachycení hesla neoprávněnou osobou, případně předávání hesla mezi uživateli. Z toho důvodu autentizace pomocí hesla bývá využívána především pro nízký způsob zabezpečení, nebo při přihlašování na určité webové stránky. [5], [6]

1.2 Autentizace předmětem

Jedná se o metodu, která je založena na vlastnictví nějakého předmětu. Tímto předmětem bývají RFID karty, magnetické karty nebo karty s čárovým kódem. Uživatel tímto předmětem prokazuje svoji totožnost při vstupu do objektu. Předmět bývá často označován jako token. Při vstupu je uživatel vyzván, aby přiložil token. Po přiložení se načte identifikační údaj, který se porovná s údajem uloženým v databázi. Pokud se identifikační údaj shoduje,

tak je přístup povolen. Za nevýhodu lze považovat odcizení, nebo vytvoření kopie, se kterou se lze dostat do objektu. Často se proto využívá kombinace tokenu s heslem nebo biometrickým údajem. [4], [5]

1.2.1 RFID karty

Identifikace pomocí elektromagnetických vln. Snímač bývá tvořen z radiofrekvenčního vysílače a přijímače ovládané procesorem. Elektromagnetická vlna je odeslána čtečkou, tato vlna je následně zpracována anténou tagu. Naindukované napětí zapříčiní vznik elektrického proudu, pomocí kterého je nabíjen kondenzátor v tagu. Jakmile napětí na kondenzátoru dosahuje stanovenou hodnotu, tak dochází k aktivaci mikroprocesoru. ID tagu je následně odesláno čtečce. Dosah u klasických čteček bývá od 5 cm do 2 cm. [7], [8]

RFID tag – bývá tvořen z integrovaného obvodu a antény, která je tvořena malou drátěnou cívkou. Celek je ukryt v ochranném materiálu, plastová karta, nebo plastový čip na klíče. Každý tag od výroby obsahuje jedinečné ID, které nelze změnit ani vytvořit jeho kopii. S RFID tagy se lze setkat neustále. Při návštěvě aquaparku se s RFID čipem lze dostat pouze do zaplacené zóny. Další příklad lze nalézt v obchodě, kde se potraviny označují RFID čipem v podobě nálepky, která slouží pro ochranu zboží před odcizením. Ale nejčastějším případem je použití RFID karet v přístupových systémech. Při zamezení přístupu neoprávněných osob do objektu. Každý zaměstnanec vlastní svou kartu, se kterou se prokazuje při vstupu a následně při pohybu v objektu. Ve většině firem se ještě využívá kombinace s výdejem stravy. Výhoda spočívá v jedinečném identifikačním údaji každého tagu. Nemělo by být snadné vytvořit přesnou kopii tagu. Zde jsou uvedeny provedení RFID tagu. [7]

Obr. 2. RFID karta.

Obr. 3. RFID čip.

Obr. 4. RFID náramek.

Pasivní systémy – tagy neobsahují baterii, přeměňují rádiový signál snímače na energii, kterou využívají ke krátkému napájení a k přenosu identifikačních údajů. Jelikož je vysílací výkon tagu celkem nízký, tak tyto tagy komunikují na menší vzdálenosti v řádech centimetrů. [5], [9]

Aktivní systémy – tagy jsou opatřeny baterií, z toho důvodu komunikují na větší vzdálenost v řádech stovek metrů. Jelikož tag obsahuje baterii, dochází i k jeho zvětšení rozměrů. Uplatnění nacházejí v kontejnerové dopravě, kde není možné provádět kontrolu v těsné blízkosti. Na rozdíl od pasivních tagů disponují snímáním teploty, vlhkosti nebo tlaku. Tagy s baterií lze rozdělit na dvě kategorie: aktivní s baterií – vysílají nepřetržitě informaci bez rozdílu na činnost čtečky, pasivní s baterií – rozdíl spočívá ve vysílání informace pouze po aktivaci čtečkou. [5], [9]

Podle frekvence lze rozlišit tři skupiny RFID:

- nízkofrekvenční – FL (124 nebo 134 kHz), dosah 0,2 až 0,5 m,
- vysokofrekvenční – HF (13,56 MHz), čtecí dosah do cca 1 m,
- ultra frekvenční – UHF (860 až 960 MHz, celosvětově nejednotná frekvence). [10]

1.2.2 Magnetické karty

Bývají složeny z magnetického pásku umístěného na identifikační kartě, který po zmagnetování vytvoří malé permanentní magnety. Kterým jsou přiděleny logické úrovně 1 pro zmagnetování a 0 pro nezmagnetovaní. Zápis je proveden protažením karty s proužkem přes čtecí zařízení. Uložená data lze kdykoliv změnit, nebo upravit. Tento systém neumožňuje bezkontaktní autentizaci, jelikož je nutný přímý kontakt přes čtecí zařízení. Kvůli častému protahování přes čtecí zařízení dochází k opotřebení. Mezi nevýhody patří, snadné vytvoření kopie dané karty, často se proto využívají v kombinaci se zadáváním hesla. Dříve byly tyto karty využívány pro platby v obchodech. [3], [4]

Obr. 5. Magnetická karta.

1.2.3 Karty s čárovým kódem

Patří mezi nejjednodušší identifikační předmět. Čárový kód je tvořen černými pruhy různých tlouštěk na bílém podkladu. Dohromady se označují jako prvky. Čtečka čárových kódů se skládá z optoelektronického snímače. Při snímání čárového kódu je vyslaný světelný paprsek černými pruhy pohlcen a od světlých mezer se odrazí zpět do čtečky. Světelné paprsky jsou zpracovány pomocí foto senzoru a převedeny na elektrický signál. Synchronizace je zajištěna pomocí prvních a posledních proužků. Nevýhodou je snadné vytvoření autentické kopii. [3], [4]

Obr. 6. Karta s čárovým kódem.

1.3 Biometrická autentizace

Metoda využívá neměnných fyziologických a behaviorálních vlastností uživatelů pro identifikaci. Fyziologické vlastnosti lze označit, jako ty od narození a není možné je změnit např. otisk prstu, geometrie ruky, geometrie tváře, duhovka oka nebo sítnice oka. Behaviorální vlastnosti jsou naučené nebo odpozorované od okolních lidí, dají se přeučit nebo odnaučit do této skupiny patří: např. dynamika hlasu, dynamika chůze, dynamika podpisu nebo psaní na klávesnici. [6]

Identifikace není založená na znalosti, ani vlastnictví předmětu. Biometrika je rychlou a spolehlivou metodou při ověřování totožností uživatelů. Výhodou také je, že postupem života jsou biometrické znaky pořád stejné. Další a poměrně důležitou výhodou je, že nelze vytvořit přesnou kopii otisku prstu, nebo kopii krevního řečiště. Princip těchto systémů spočívá ve snímání biometrických vlastností, porovnání s databází a vyhodnocení. U uvedené autentizace lze s největší pravděpodobností předpokládat, že ji provádí určitá osoba. Zneužití nastává v případě, když je daná osoba někým cizím přinucena. Při ověřování uživatelů se využívají dva způsoby: [4], [6]

- **verifikační mód** – uživatel dopředu zadá svoje identifikační údaje a biometrický údaj. Proběhne následně kontrola, zda k zadanému identifikačnímu údaji náleží daný biometrický údaj, [1]
- **identifikační mód** – uživatel zadá biometrický údaj a systém prohledává celou databázi, aby zjistil totožnost uživatele. [1]

Treshold – slouží k nastavení prahu citlivosti při ověřování uživatelů. Pokud je práh citlivosti nastaven na Th = 82, tak přístup je povolen pouze pro osoby se shodou Th ≥ 82. [1]

1.3.1 Otisk prstu

Identifikace podle otisku prstu patří mezi nejrozšířenější biometrické metody. V posledních letech lze zaznamenat rozšíření otisků prstů do laptopů, tabletů nebo smartphonů. Pomocí kterého se uživatel přihlašuje do systému. Uplatnění lze také nalézt při přihlašování do internetového bankovnictví, kde se většina uživatelů přihlašuje pomocí otisku prstu.

Samotná metoda obrazců papilárních linií sloužící pro identifikaci se poprvé začala využívat koncem 19. stol. Za prvního objevitele je považován Sir Francis Galton, který definoval identifikační prvky na prstu. S jejich využitím lze snáze identifikovat uživatele. Tyto prvky jsou základem vědního bádání o otisku prstu. [6]

Lze rozlišit dva typy snímání papilárních linií: statické snímání – uživatel podkládá prst na snímací část čtečky a není potřeba s prstem nijak pohybovat. Prst je lehce položen, není třeba využití většího tlaku na senzor. V případě velkého tlaku nastane poškození senzoru. Po přiložení prstu a následném sejmutí na snímači mohou zůstat skryté otisky, což je z jistého pohledu nehygienické. Druhý typ snímání je označován jako snímání šablonové – zde uživatel postupně přejíždí prstem po snímací čtečce. Snímač má tvar úzkého pruhu. Snímaný obraz bývá složen z několika pásků. Při opětovném přejetí prstu přes snímač nastane očištění snímače a snímač je tak relativně udržován v čistém stavu. Na snímači tak nezůstávají staré otisky, což lze považovat za patřičnou výhodu. Při přikládání prstu a opětovném přejíždění přes snímač je nutné dodržovat přesný postup, což bývá v řadě případů častým problémem. Odlišné otisky mají i jednovaječná dvojčata. [6]

1.3.2 Geometrie ruky

Rozpoznání podle geometrie ruky vyvinul David Sidlauskas v roce 1985. Metoda je založená na měření délky, tloušťky a šířky všech prstů na ruce, která je umístěna na desce s pěti polohovými kolíky. Na zaznamenaném obrazu ruky lze zpozorovat kolem 31 000 polohových bodů, které slouží pro další měření. Zaznamenání obrazu je tvořeno ze dvou stran, jeden shora a druhý pohled z boku. Systémy s rozpoznáním geometrie ruky lze nalézt v řadě přístupových systémech. Uživatel přiloží dlaň na snímač, který zaznamená identifikační hodnoty a provede vyhodnocení. [5], [6]

1.3.3 Geometrie tváře

Rozpoznávání podle geometrie tváře se řadí mezi nejpřirozenější metody sloužící k identifikaci. Rozpoznání je zajištěno s využitím víceúrovňových obrazců, označované jako šedé

obrazce. K zajištění správné identifikace se srovnávají specifické pozice očí, nosu, úst, uší, čelistí a jejich vzdálenosti. Rozpoznání podle tváře nevyžaduje žádnou aktivitu ani kontakt s uživatelem. K rozpoznávání geometrie tváře lze využít dva různé typy: geometrický, který souvisí se specifickými rysy tváře a fotometrický, který využívá samotný obraz tváře. Jistý problém nastává u dvojčat, jelikož se v některých specifických pozicí shodují. [5], [6]

1.3.4 Duhovka oka

Identifikaci pomocí duhovky oka lze také využít v přístupových systémech. Jelikož se každé dvě duhovky, včetně dvojčat liší. Duhovku tvoří barevná část, zbarvení i struktura duhovky pochází z genetické informace. Ale u vzorkování duhovky tomu tak není, to je čistě náhodné. Při kontrole nemusí osoba konat žádnou činnost, ani není vyžadován přímý kontakt. Pro snímání se požaduje vysoce kvalitní digitální kamera s infračerveným osvětlením oka. Snímání duhovky se ukládá do fázorových diagramů. V těchto fázorových diagramech bývají obsaženy informace o výskytech, natočení a pozici plošek. S využitím těchto dat lze vytvořit duhovkovou mapu, která se při kontrole porovnává. [6]

1.3.5 Dynamika chůze

Metoda podle dynamiky chůze je zásadní při identifikování pachatelů. Uplatnění také nachází na letištích nebo nádražích. Metoda je založena na vyhodnocování křivek drah, jež představují určitou část lidského těla. Každý člověk disponuje určitým stylem chůze, který závisí na jeho těžišti a pohybových schopnostech. Lze tedy zjistit, zda člověk má zvednutou pravou ruku, nebo třeba drží zbraň. [6]

1.3.6 Dynamika podpisu

Metoda dynamiky podpisu je založena na skutečnosti, že každý člověk využívá k podpisu jedinečné vlastnosti. Pro vyhodnocení dynamiky podpisu se používají speciální pera s podložkou, nebo elektronická tabule. Z podpisu lze vyčíst tlak při psaní, nebo tah pera. Mezi základní dynamické vlastnosti patří rychlost psaní, akcelerace, sklon písma, směr tahu a tlak na podložku. Tyto vlastnosti se následně zaznamenávají do trojrozměrného souřadnicového systému, kde proběhne srovnání. [5], [6]

1.3.7 Dynamika hlasu

Identifikace pomocí hlasu byla dříve využívána pouze v kriminalistických metodách, ale postupně se rozšiřuje i do běžného prostředí. Ověření lze provést s předem nahraným vzorkem. Jako nahraný vzorek lze označit slovo, slovní spojení, nebo i celou větu. Danou nahrávku bývá velmi složité napodobit, jelikož každá osoba disponuje specifickou barvou hlasu, rytmem nebo třeba intonací. Vstupním zařízením bývá nejčastěji mikrofon, případně v některých případech se používá telefon. Náročnější rozpoznání hlasu probíhá, pokud se osoba nachází mezi jinými osobami, nebo je v okolí šum. Výhodou této metody je poměrně nízká cena a spolehlivost. [5], [6]

Vlastnosti biometrického údaje:

- universálnost – jakýkoliv člověk by jimi měl disponovat,
- měřitelnost – charakteristiky musí být změřitelné,
- trvanlivost – musí být zaručeno, že se charakteristika nebude v čase měnit,
- rozlišitelnost – mezi dvěma lidmi vždy musí být určitá rozdílnost. [1]

Dělení biometrických systémů:

- **pasivní** – požaduje se nastavení zkoumané části před snímačem, následně už není vyžadován žádný pohyb, např. geometrie ruky,
- **aktivní** – po uživateli se vyžaduje provedení určité aktivity např. mluvení, chůze,
- **kontaktní** – bývá vyžadován nutný kontakt zkoumané části se snímačem, např. přiložení otisku prstu,
- **bezkontaktní** – není vyžadován kontakt zkoumané části se snímačem, bývá požadována pouze přímá viditelnost na danou část, např duhovka oka. [1]

Třídy identifikace

V následující tabulce jsou uvedeny třídy identifikace. Celkem jsou čtyři třídy identifikace, kde třída 0 je nejnižší a třída 3 nejvyšší.

Tab. 1. Třídy identifikace. [1], [3], [5]

Třída 0	• není požadována přímá identifikace • přístup se realizuje např. pomocí aktivace tlačítka nebo čidel pohybu • nutná přítomnost fyzické ostrahy, za účelem kontroly průkazu totožností osobě vstupující do objektu
Třída 1	• založena na znalosti dat (heslo, pin, ID) • po zadání údajů proběhne porovnání s údaji uloženými v paměti
Třída 2	• založena na použití identifikačních prvků nebo biometrie • přístup se realizuje po přiložení identifikační karty, přívěšku na klíče, náramku, otisku prstu, nebo třeba dynamiky hlasu
Třída 3	• základ spočívá v propojení třídy 1 a třídy 2 • přístup se realizuje pomocí kombinace biometriky, nebo přístupové karty ve spojení se zadáním hesla, nebo pinu

Třídy přístupu

V následující tabulce jsou specifikovány třídy přístupu. Při přístupu se využívá třída A, nebo třída B.

Tab. 2. Třídy přístupu. [1], [3], [5]

Třída A	• používá se v případech, kdy nejsou požadavky na časový filtr ani není vyžadováno ukládání přístupových transakcí
Třída B	• používá se v případech, kdy se vyžaduje použití časových filtrů, a jsou kladeny požadavky na ukládání přístupové transakce • otevření přístupu bez oprávnění • odmítnutý přístup, otevření přístupu mimo povolenou dobu

2 PLATFORMY ZALOŽENÉ NA ARDUINU

Za vývoj jsou považováni lidé z italského Interaction Design Instute. Prvotním záměrem vzniku Arduina bylo vytvoření snadno ovladatelné a levné vývojové sady pro studenty, kteří si z jistých finančních důvodů nemohli dovolit dražší desky. Arduino vzbudilo u studentů zájem, což motivovalo jeho tvůrce o rozšíření do celého světa. Rozšíření je hlavně přisuzováno sdílením různých schémat nebo návodů, jelikož se jedná o Open Source projekt. Od roku 2005 se vytvořilo velké množství různých desek. Některé desky vznikaly ve spolupráci se společností např. Intel. Na trhu lze také nalézt neoficiální typy tzv. klony, které se rozlišují např. absencí oficiálních popisů Arduino. Obecně bývá platforma Arduino tvořena třemi důležitými částmi: základní deskou s mikropočítačem např. Arduino Due, vývojové prostředí pro tvorbu programu např. Arduino Integrated Development Environment (dále jen IDE) a knihovnami které je nutné importovat do software. [11]

Platforma Arduino je kompatibilní s nejrůznějšími senzory a snímači, může se jednat o senzor tlaku, ultrazvukový senzor, čtečku otisků prstů a řadu dalších. S využitím těchto komponent lze vytvořit nejrůznější zapojení. Platforma je vhodná pro začátečníky, jelikož je velmi srozumitelná a jednoduchá na pochopení. Samotné programování v IDE je také celkem jednoduché. Na trhu lze také nalézt podobné platformy např. Raspberry Pi, Banana Pi, Pine64.

Raspberry Pi 4 obsahuje 64-bit procesor s frekvencí 1.5 GHz, poskytuje uživateli čtyři USB porty typu A nebo úložiště 4 GB. Je tedy oproti Arduinu Mega výkonnější a o pár stokorun i dražší. Banana Pi M64 je osazena 4jádrovým procesor Allwinner A64, nabízí uživateli např. slot na microSD, WIFI připojení nebo třeba 8 GB interní úložiště Embedded MultiMediaCard (dále jen eMMC). Cena u Banana Pi M64 je srovnatelná s cenou u Rasperry Pi 4. Pouze u Arduina Mega lze nalézt 54 digitálních vstupů/výstupů, ostatní uvedené platformy nabízí patřičně méně vstupů/výstupů na základní desce. [12], [13]

2.1 Typy základních desek Arduino

Základem každé desky Arduino je mikrokontroler, který nejčastěji dodává firma Atmel, nebo případně Intel. Procesor je na desce osazen spolu s dalšími elektronickými komponenty, zajišťujícími správnou funkčnost desky. Pro všechny desky je specifické modré grafické zbarvení. U některých desek lze za názvem nalézt označení např. Rev3 nebo R3, což definuje číslo verze. Odlišnosti u jednotlivých desek bývají v rozmístění součástek, nebo vzhledu, ale nedochází k velkým změnám. Většina desek disponuje převodníkem, který

slouží pro komunikaci mezi PC (USB) a čipem. Objevují se ale i takové desky, které převodník postrádají, jsou možná dvě vysvětlení. Prvním typem jsou desky, kde dochází k zmenšení rozměrů a následné úspoře místa. Zde se využívá externí převodník. Druhým typem jsou desky, které obsahují převodník zabudovaný v čipu procesoru. Následně zde budou uvedeny a charakterizovány některé typy desek. [11]

2.1.1 Arduino Mini

Deska byla navržena tak, aby byla vhodná pro úsporu místa. Jelikož došlo k zmenšení rozměrů, tak došlo k odstranění i některých zásadních portů, např. USB. Uživatel tak při programování musí využít externí USB-Serial převodník. Deska je osazena procesorem ATmega328 s taktovací frekvencí 16 MHz. Na základě svých malých rozměrů nachází uplatnění např. v dálkových ovladačích nebo vypínačích. [11]

Tab. 3. Specifikace Arduino Mini. [14]

mikrokontroler:	ATmega328
provozní napětí:	5 V
digitální vstupy/výstupy:	14
PWM:	6
analogové vstupy:	8
Flash paměť:	32 kB
SRAM:	2 kB
EEPROM:	1 kB
taktovací frekvence:	16 MHz
rozměry:	3,33 x 1,85 cm

Jelikož deska disponuje velmi malými rozměry, není možné na desku přímo připojit vodiče. Je proto nutné použití paticí, pomocí kterých se deska propojí např. s nepájivým polem, kde už lze připojit vodiče. [11]

Obr. 7. Deska Arduino Mini. [14]

2.1.2 Arduino Micro

Jedná se o nejmenší desku Arduina. Základ tvoří mikrokontroler Atmega32U4. Deska disponuje 20 digitálními vstupy/ výstupy, ze kterých lze 7 použít jako Pulse Width Modulation (dále jen PWM) výstupy, dále se na desce nachází 12 analogových vstupů. K propojení s PC lze využít micro USB port. Mezi desky, které obsahují převodník zabudovaný v čipu, patří právě desky Arduino Micro. Patřičnou výhodou je možnost posílat příkazy (např. stisk klávesy, posunutí myši) počítači, který si myslí, že se jedná o myš, nebo klávesnici. U jiných desek je nutné správně přeprogramovat převodník. Pro vytvoření např. vlastní klávesnice je tedy vhodnější použití desky Arduino Micro. [11], [15]

Tab. 4. Specifikace Arduino Micro 1/2. [15]

mikrokontroler:	ATmega32U4
provozní napětí:	5 V
vstupní napětí (doporučené):	7–12 V
vstupní napětí (limit):	6-20 V
digitální vstupy/výstupy:	20
PWM:	7
analogové vstupy:	12
DC proud na I/O pin:	20 mA
DC proud pro 3,3 V pin:	50 mA

Tab. 5. Specifikace Arduino Micro 2/2. [15]

Flash paměť:	32 kB
SRAM:	2,5 kB
EEPROM:	1 kB
taktovací frekvence:	16 MHz

Obr. 8. Arduino Micro. [15]

2.1.3 Arduino Uno

Deska je osazena procesorem ATmega328P. Pro připojení k počítači slouží USB port typu B. Deska poskytuje 14 digitálních vstupů/výstupů, ze kterých lze 6 využít jako PWM, dále se na desce nachází 6 analogových vstupů. Pro napájení desky lze využít dva způsoby: pomocí USB kabelu, nebo pomocí adaptéru, který lze připojit pomocí konektoru na základní desce. [11]

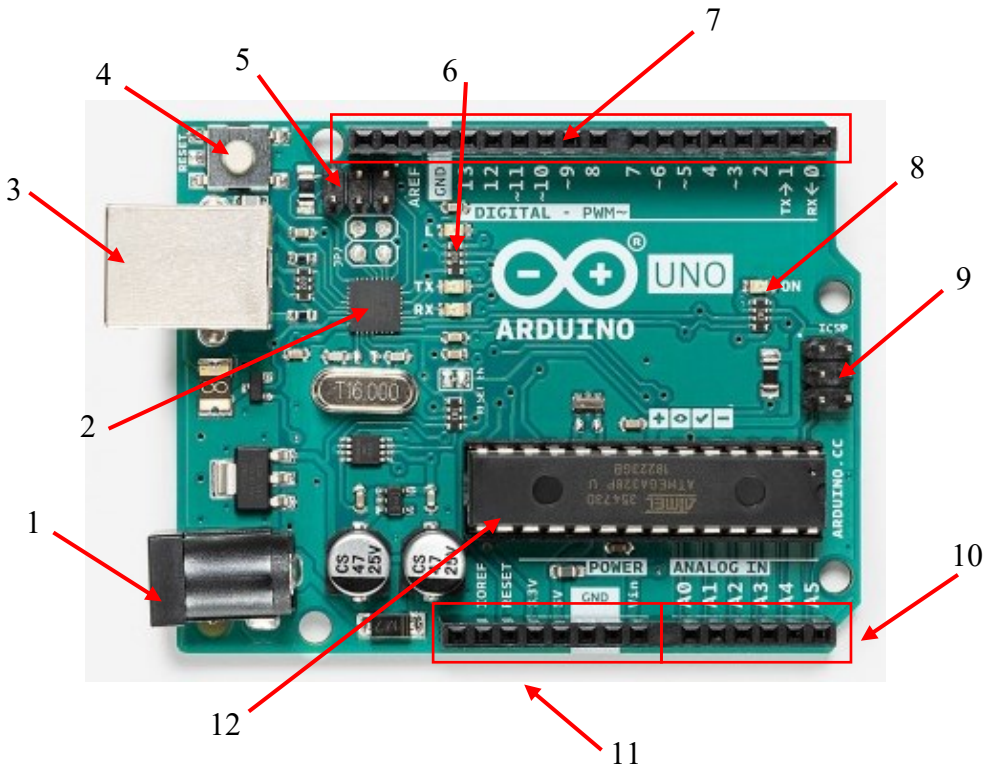
Tab. 6. Specifikace Arduino Uno 1/2. [16]

mikrokontroler:	ATmega328P
provozní napětí:	5 V
vstupní napětí (doporučené):	7–12 V
vstupní napětí (limit):	6-20 V
digitální vstupy/výstupy:	14
PWM:	6

Tab. 7. Specifikace Arduino Uno 2/2. [16]

analogové vstupy:	6
DC proud na I/O pin:	20 mA
DC proud pro 3,3 V pin:	50 mA
Flash paměť:	32 kB
SRAM:	2 kB
EEPROM:	1 kB
taktovací frekvence:	16 MHz

Tato hlavní verze posloužila pro vývoj dalších dvou specifických desek. Jedná se o Arduino Ethernet, které disponuje stejnou výbavou jako Arduino UNO, pouze s rozdílem, že místo USB portu se deska osadila portem pro ethernet. Tento port lze využít pro připojení k síti. Deska Arduino Ethernet ještě navíc obsahuje slot pro připojení microSD karty. Druhým typem desky je Arduino Bluetooth. V tomto případě je USB port nahrazen Bluetooth modulem, který se využívá pro bezdrátovou komunikaci. [11]



Obr. 9. Popis desky Arduino Uno. [16]

Tab. 8. Popis desky Arduino Uno. [16]

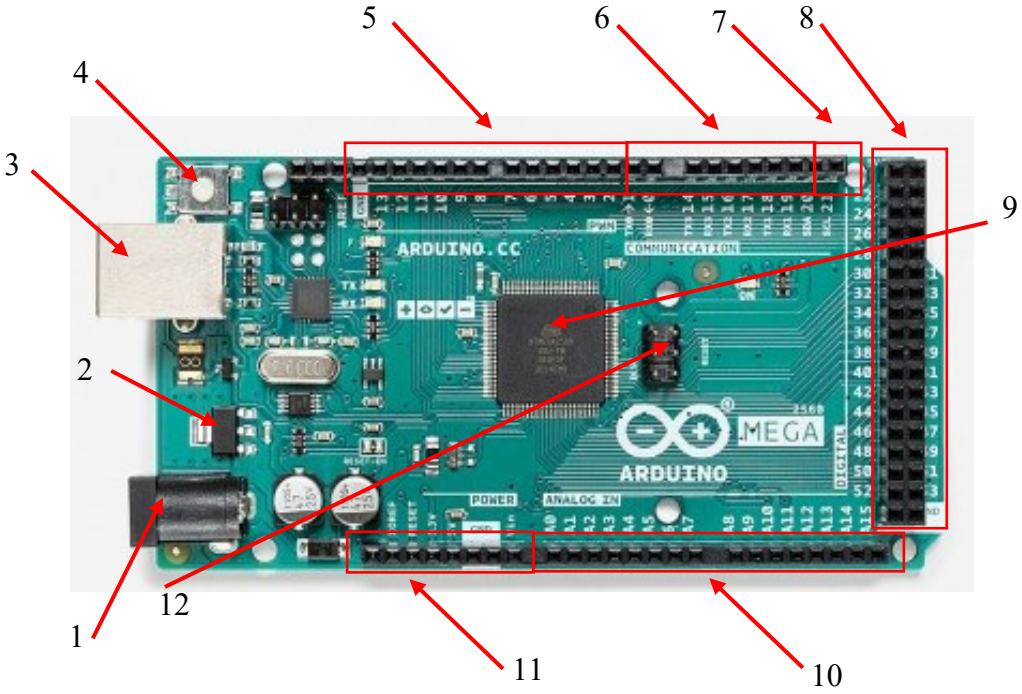
1	napájecí konektor	7	digitální vstupy/výstupy
2	USB – serial převodník	8	signalizační LED dioda, signalizuje připojené napětí
3	konektor USB typ B	9	ICSP piny pro externí programování mikrokontroleru
4	tlačítko reset	10	analogové vstupy
5	ICSP piny pro externí programování USB-serial převodníku	11	napájecí výstupy
6	signalizační LED diody	12	mikrokontroler ATmega328P

2.1.4 Arduino Mega 2560

U desky Arduino Mega došlo ke zvětšení rozměrů, což zajišťuje větší počet pinů a prostor pro větší čipy. Digitálních vstupů/ výstupů na desce lze nalézt 54, ze kterých 15 lze použít jako PWM. Na desce se nachází Two Wire Interface (dále jen TWI), někdy označováno jako Inter-Integrated Circuit (dále jen I2C) piny, jedná se o dva piny s označením SDA a SCL, tyto piny slouží např. pro připojení Liquid Crystal Display (dále jen LCD) displeje. K programování přes PC slouží USB port typu B, pro externí programování mikrokontroleru lze využít piny In Circuit Serial Programming (dále jen ICSP). Tyto piny se nachází uprostřed desky. Specifickou odnoží uvedené desky je Arduino Mega ADK, které disponuje jedním USB portem navíc. Slouží k připojení zařízení s Androidem přes Android Debuger Bridge (dále jen ADB). Pro napájení desky lze využít USB kabel, nebo adaptéru, pro který je na desce připraven konektor. [11], [17]

Tab. 9. Specifikace Arduino Mega. [17]

mikrokontroler:	ATmega2560
provozní napětí:	5 V
vstupní napětí (doporučené):	7-12 V
vstupní napětí (limitní):	6-20 V
digitální vstupy/výstupy:	54
PWM:	15
analogové vstupy:	16
DC proud na I/O pin:	20 mA
DC proud pro 3,3 V pin:	50 mA
Flash paměť:	256 kB
SRAM:	8 kB
EEPROM:	4 kB
taktovací frekvence:	16 MHz



Obr. 10. Popis desky Arduino Mega. [17]

Tab. 10. Popis desky Arduino Mega. [17]

1	napájecí konektor	7	TWI (I2C) komunikace
2	regulátor 5 V	8	digitální vstupy/výstupy
3	USB konektor typ B	9	mikrokontroler ATmega2560
4	resetovací tlačítko	10	analogové vstupy
5	PWM výstupy	11	napájecí vstupy
6	sériová komunikace	12	ICSP piny pro externí programování mikrokontroleru

2.1.5 Arduino Due

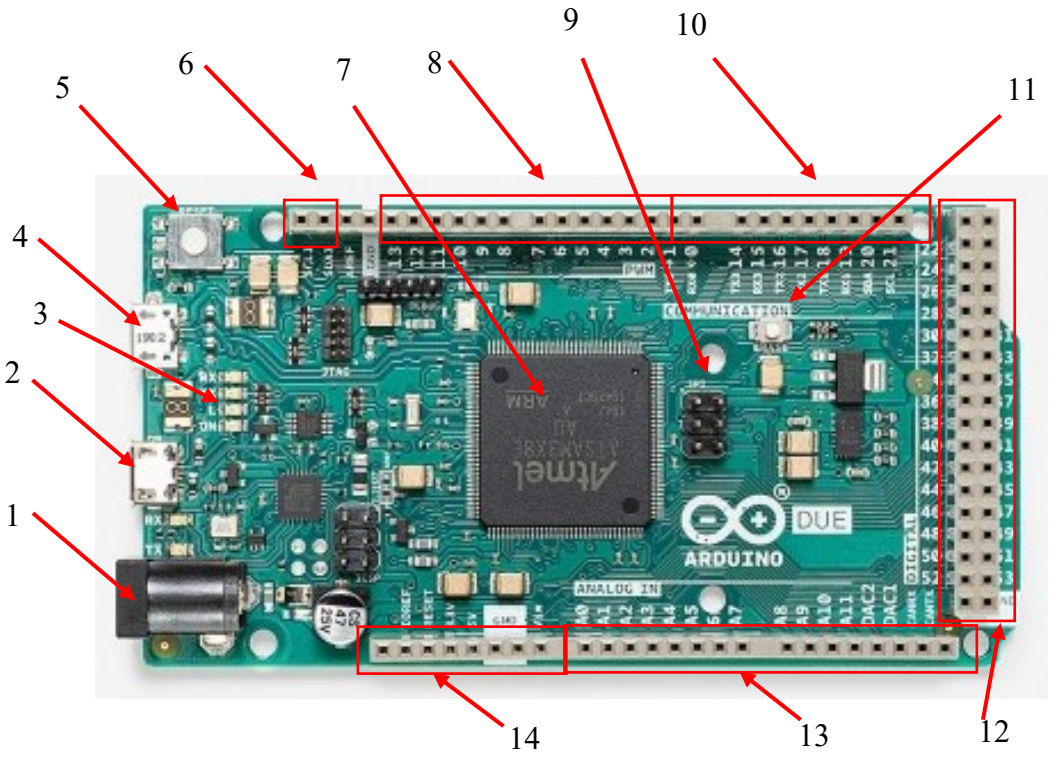
Arduino Due běží na výkonnějším čipu než Arduino Mega, je taky považováno za jeho nástupce. Deska je osazena procesorem Atmel SAM3X8E s taktovací frekvencí 84 MHz a 32bitovým jádrem. Ve srovnání s ostatními deskami, které disponovaly pouze 8bity a nejvýše 16 MHz, zde došlo ke slušnému zlepšení. Také na první pohled lze zpozorovat, že se na desce nachází dva porty microUSB. První port slouží k programování čipu a pomocí druhého microUSB lze připojit např. klávesnici, nebo myš. Oproti ostatním deskám, zde nastává menší rozdíl v pracovním napětí, které je pouze 3,3 V. Vyšší napětí by způsobilo poškození desky. [11]

Tab. 11. Specifikace Arduino Due 1/2. [18]

mikrokontroler:	Atmel SAM3X8E
pracovní napětí:	3,3 V
vstupní napětí (doporučené):	7-12 V
vstupní napětí (limitní):	6-16 V
digitální vstupy/výstupy:	54
PWM:	12
analogové vstupy:	12

Tab. 12. Specifikace Arduino Due 2/2. [18]

analogové výstupní piny:	2 (DAC)
celkový proud na všech I/O pinech:	130 mA
DC proud pro 3,3 V pin:	800 mA
DC proud pro pin 5 V pin:	800 mA
Flash paměť:	512 kB
SRAM:	96 kB
taktovací frekvence:	84 MHz



Obr. 11. Popis desky Arduino Due. [18]

Tab. 13. Popis desky Arduino Due 1/2. [18]

1	napájecí konektor	8	PWM výstupy
2	programovací USB port	9	SPI komunikace

Tab. 14. Popis desky Arduino Due 2/2. [18]

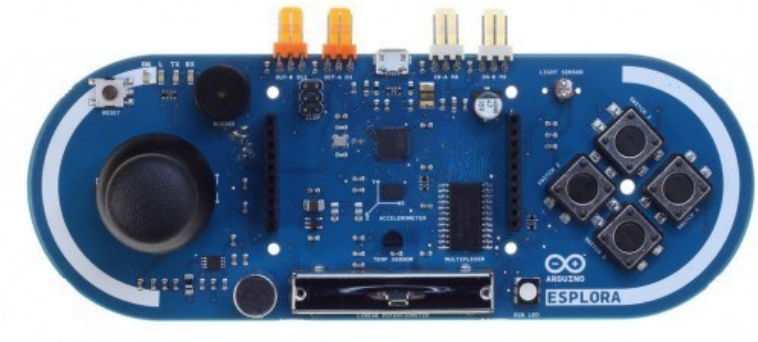
3	signalizační LED diody	10	sériová komunikace
4	USB port pro připojení zařízení	11	tlačítko na vymazání
5	resetovací tlačítko	12	digitální vstupy/výstupy
6	I2C komunikace	13	analogové vstupy
7	mikrokontroler Atmel SAM3X8E	14	napájecí vstupy

2.1.6 Arduino Esplora

Jedná se o desku, která disponuje velkým množstvím komponentů. Na desce lze nalézt joystick, teploměr, tlačítka, světelnou signalizaci nebo bzučák. Nabízí také možnost připojení LCD obrazovky pomocí vyvedených konektorů. Deska je osazena procesorem ATmega32u4. Přítomnost velkého množství komponentů na desce je velkou výhodou, jelikož u ostatních desek by bylo nutné dané komponenty složitě připojovat. Na desce také lze nalézt tlačítko pro resetování. Jednotlivé stavy jsou signalizovány pomocí čtyř Light Emitting Diode (dále jen LED) diod: ON, L, RX, TX. Pokud LED dioda ON svítí zeleně, tak probíhá příjem dat, LED dioda L je přímo připojena k mikrokontroleru a lze ji nastavovat přes pin 13. LED diody RX a TX signalizují příjem nebo přenos informací pomocí USB komunikace. Arduino Esplora se od ostatních desek liší nejen spoustou komponentů, ale i svým tvarem. Jedná se o speciální typ desky, se kterou lze vytvořit vlastní herní konzoli pro ovládání her. [11], [19]

Tab. 15. Specifikace Arduino Esplora. [19]

mikrokontroler:	ATmega32u4
pracovní napětí:	5 V
Flash paměť:	32 kB
SRAM:	2,5 kB
EEPROM:	1 kB
taktovací frekvence:	400 MHz



Obr. 12. Arduino Esplora. [19]

2.1.7 Arduino Yún

Pro Arduino Yún byl využit jak čip ATmega32u4, tak ještě navíc čip Atheros AR9331, který podporuje komunikaci s operačním systém Linux. Konkrétně distribuci založenou na OpenWrt nazvanou Linio OS. Tím se liší od ostatních desek, které tuto možnost nemají. Komunikace mezi oběma čipy je zajištěna pomocí bridge. Na desce lze nalézt micro USB port, který slouží pro programování, dále USB port typu A. Pro posílání dat po síti lze využít ethernetový port, který se na desce také nachází. Případně lze využít WiFi modul, kterým deska disponuje. Další výhodou je slot pro microSD kartu. Na desce se nachází 20 digitálních vstupů a výstupů pro připojení různých komponentů. Z toho sedm pinů slouží jako výstupy PWM. Dalších 12 analogových vstupů na desce je určeno pro další využití. [11], [20]

Tab. 16. Specifikace Arduino Yún 1/2. [20]

mikrokontroler:	ATmega32U4
pracovní napětí:	5 V
digitální vstupy/výstupy:	20
PWM:	7
analogové vstupy/výstup:	12
celkový proud na všech I/O pinech:	40 mA
DC proud pro 3.3 V pin:	40 mA
Flash paměť:	32 kB
SRAM:	2,5 kB

Tab. 17. Specifikace Arduino Yún 2/2. [20]

EEPROM:	1 kB
taktovací frekvence:	16 MHz
procesor:	Atheros AR9331
provozní napětí:	3,3 V
Flash paměť:	16 MB
SRAM:	2,5 kB
EEPROM:	1 kB
taktovací frekvence:	400 MHz

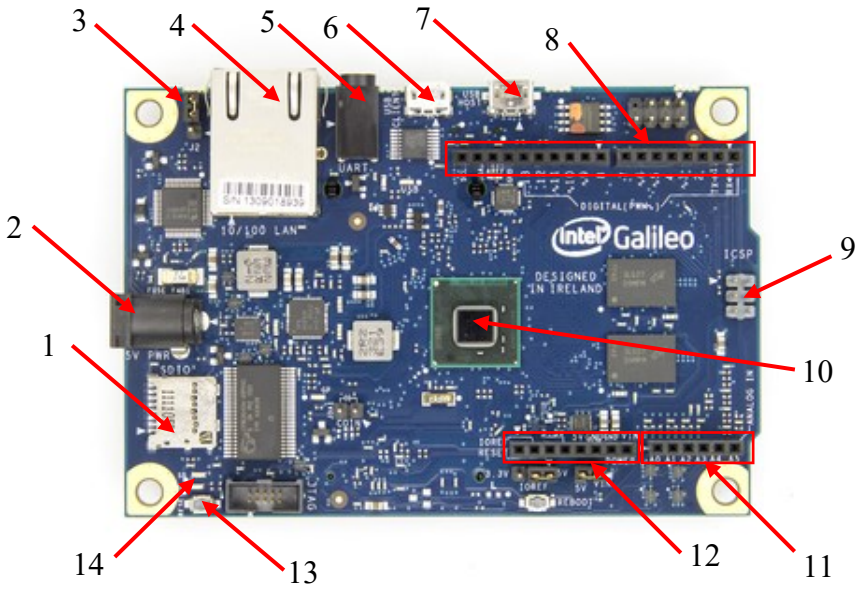
Obr. 13. Arduino Yún. [20]

2.1.8 Arduino Intel Galileo

Jedná se o desku, která vznikla ve spolupráci se společností Intel. Deska je osazena čipem Intel Quark SoC X1000, jedná se o 32bitový procesor, který pracuje na frekvenci 400 MHz. Deska disponuje dvěma USB porty, slotem na microSD karty a portem pro Ethernet. Výhodou je přítomnost mini-PCI Express slotu, pomocí kterého lze snadno připojit další přídavné karty. [21]

Tab. 18. Specifikace Arduino Intel Galileo. [21]

mikrokontroler:	Intel Quark SoC X1000
pracovní napětí:	3,3 V nebo 5 V
vstupní napětí (doporučené):	5 V
digitální vstupy/výstupy:	14
PWM:	6
analogové vstupy:	6
celkový proud na všech I/O pinech:	80 mA
DC proud pro 3,3 V pin:	800 mA
DC proud pro 5 V pin:	800 mA
Flash paměť:	512 kB
SRAM:	512 kB
RAM:	256 MB DDR3
EEPROM:	8k8
taktovací frekvence:	400 MHz



Obr. 14. Popis desky Arduino Intel Galileo. [21]

Tab. 19. Popis desky Arduino Intel Galileo. [21]

1	slot na microSD	8	digitální vstupy/výstupy
2	napájecí konektor	9	ICSP komunikace
3	I2C	10	mikrokontroler IntelQuark
4	ethernet konektor	11	analogové vstupy
5	RS-232 port	12	napájecí vstupy
6	USB port pro programování	13	resetovací tlačítko
7	USB port pro připojení zařízení	14	indikační LED

2.2 Srovnání uvedených platforem

Bylo zde uvedeno několik různých vývojových platforem Arduino, každá deska disponuje určitými vlastnostmi, které zase jiná nemá, nebo má lepší. Při výběru platformy záleží, zda uživatel požaduje např. více digitálních vstupů /výstupů, nebo raději slot na microSD kartu. Třeba na desce Arduino Mega lze nalézt 54 vstupů/výstupů, ale není zde slot na microSD kartu ani WIFI modul. Oproti tomu Arduino Yún nabízí k dispozici 20 digitálních vstupů/výstupů a také slot na microSD kartu. Jako většina desek, bývá osazena čipem ATmega32u4, ale navíc obsahuje také čip Atheros AR9331. Tento čip podporuje spolupráci s distribucí operačního systému Linux. Jako desku malých rozměrů, ale s dostatečnými vstupy a výstupy, lze využít Arduino Micro. Deska obsahuje 20 vstupů/výstupů, na desce je mino jiné umístěn také microUSB port pro programování. Většina menších desek musí využívat externí převodník, ale Arduino Micro disponuje převodníkem zabudovaným přímo v čipu procesoru. Může tak posílat příkazy např. stisk klávesy do počítače, který je přesvěcen, že to posílá myš nebo klávesnice. Pro vytvoření vlastní herní konzole lze využít Arduino Esplora. Tato deska obsahuje joystick, tlačítka, světelnou signalizaci, nebo možnost připojení LCD displeje pomocí připravených konektorů. Deska disponuje také specifickým tvarem, který připomíná herní konzoli. Komponenty, které se na desce nachází jsou velkou výhodou, jelikož uživatel nemusí složitě připojovat tlačítka pro ovládání. [15], [17], [19], [20]

2.3 Paměti používané u platforem Arduino

Paměť slouží jako nepostradatelný prvek počítače i mikropočítače. Bývá využívána pro ukládání dat i software s nimiž mikropočítač pracuje. Paměť lze rozdělit do tří skupin: registry –

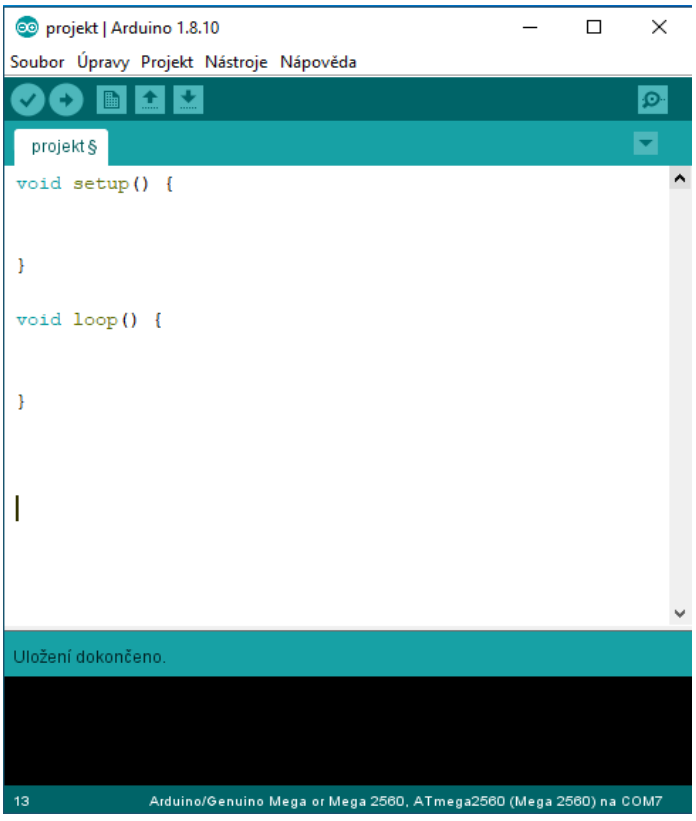
slouží k dočasnému uložení právě zpracovávané informace, jsou tvořeny paměťovým místem na čipu procesoru, vnitřní paměti – jsou tvořeny polovodičovými součástkami, nacházejí se na základní desce, vnější paměti – jsou realizovány pomocí externích disků, využívají se k dlouhodobému záznamu. Existuje několik typů pamětí např. podle způsobu ukládání nebo podle způsobu mazaní paměti. [22]

- **Read Only Memory (dále jen ROM) paměť** – data jsou uložena při výrobě, zapsaná data nelze nijak měnit, jsou určeny pouze ke čtení. [22]
- **Programable Read Only Memory (dále jen PROM) paměť** – po vyrobení neobsahuje žádná data. Je pouze na uživateli, aby vytvořil daný zápis. Zápis lze provést pouze jednou a je nevratný. Po provedení zápisu má paměť stejné vlastnosti jako paměť ROM. [22]
- **Eraseable Programable ReadOnly Memory (dále jen EPROM) paměť** – uložení dat lze provést pomocí speciálního programátoru. Uloženou informaci lze vymazat působením ultrafialového záření. Paměti bývají realizovány pomocí unipolárních tranzistorů. [22]
- **Electrically EPROM (dále jen EEPROM) paměť** – jsou podobné pamětím EPROM, ale zde se uložení dat i mazání provádí elektronicky. Uložení dat do paměti je tak mnohem rychlejší než u EPROM. Výroba je založena na použití speciálních tranzistorů, které byly vyrobeny technologií MNOS. [22]
- **FLASH paměť** – jsou velmi podobné pamětem EEPROM, rozdíl ale nastává při mazání informace, které probíhá také elektronicky a lze danou operaci provést přímo v počítači. Jelikož zápis probíhá po blocích, tak programování je provedeno v řádech sekund. Mazání probíhá také po blocích, ale je zdlouhavější, jelikož dochází k mazání bloků různé velikosti. [22]
- **Random Access Memory (dále jen RAM) paměť** – paměti jsou určeny k zápisu dat i pro čtení. Tyto paměti bývají energeticky závislé a oproti pamětím ROM i rychlejší.
    - **Static Random Access Memory (dále jen SRAM) paměť** – statická paměť. Data jsou uložena po dobu připojení ke zdroji napájení. Bývají tvořeny bistabilním klopným obvodem.

- **Dynamic Random Access Memory (dále jen DRAM) paměť** – dynamická paměť. Data jsou uložena s využitím elektrického náboje na kondenzátoru. Nevýhodou tohoto náboje je jeho samovolné vybíjení i při připojení ke zdroji napájení. Z toho důvodu je nutné pravidelně paměť obnovovat, tzv. refresh, aby nedocházelo ke ztrátě informací. [22]

2.4 Programování Arduina

Pro programování Arduina lze využít řadu software např: C, C++, nebo Arduino IDE, který je napsaný v jazyce Java. Arduino IDE je volně stažitelné ze stránek: Arduino.cc, nebo Arduino.org. Jedná se asi o nejrozšířenější software pro programování Arduina. Po vytvoření nového projektu, se zobrazí dvě hlavní smyčky: void setup a void loop. Kód, který se provádí po zmáčknutí tlačítka reset, nebo při nahrání programu do Arduina, se zapisuje do smyčky void setup. Tento kód se provede pouze jednou. Opakující kód se zapíše do smyčky loop. Tyto dvě smyčky tvoří základ každého programu. [11]



Obr. 15. Vývojové rozhraní Arduino IDE.

Na obrázku lze vidět dvě hlavní smyčky void setup a void loop. V horním panelu se nachází tlačítka s označením „fajfka“ v kolečku, pomocí tohoto tlačítka lze program ověřit, zda se v programu nenacházejí nějaké chyby. Případné chyby se zobrazují ve spodní černé části.

Dále se v horní části nachází tlačítko s označením šipka v kolečku, toto tlačítko slouží k nahrání programu do Arduina. Na pravé straně panelu se nachází lupa, pomocí které lze zobrazit sériový monitor. Na spodním panelu lze vidět připojené Arduino Mega 2560 na portu COM7. [11]

2.5 Sériová komunikace

Sériová komunikace slouží k posílání určitých dat do Arduina, nebo pokud bývá vyžadováno vypsání některých dat od Arduina. Využití také nachází např. pokud se vyžaduje, že při přiložení karty ke čtečce, se zobrazí na sériovém monitoru ID karty. Důležitou součást sériová komunikace plní při ladění programu. Pokud třeba dojde k zaseknutí na určitém místě a programátor neví, kde nastala chyba. Tak si jednoduše na sériový monitor zobrazí jednotlivé kroky programu. V případě komunikace s Arduinem, lze taktéž využít sériový monitor. Do horní lišty lze zadat příkaz a enterem se příkaz potvrdí. Pro zahájení komunikace se ve smyčce void setup musí nacházet příkaz Serial.begin(9600). Parametr v závorce udává rychlost dané komunikace. Následně se do smyčky void loop napíše příkaz k odeslání údajů z Arduina do PC. K tomu slouží příkaz Serial.println(“text“). [11]

II. PRAKTICKÁ ČÁST

3 NÁVRH A REALIZACE SYSTÉMU KONTROLY VSTUPU

Cílem práce bylo navrhnout a realizovat elektronický systém kontroly vstupu založený na třífázovém ověření uživatele.

Stručný popis funkce prototypu elektronického systému kontroly vstupu: pro přístupový systém byla využita platforma Arduino. První ověření bude založené na vlastnictví RFID karty, nebo čipu, druhé ověření bude založeno na znalosti hesla. A třetí ověření bude probíhat na základě biometrického otisku. Bude zajištěno, že při ověřování musí být dodrženo přesné pořadí ověřování, tzn. tag, heslo, otisk. Nesmí být možné, aby po přiložení tagu byl rozpoznán otisk. Kombinací těchto tří metod přístupu bude dosaženo toho, že přístup se zpřístupní pouze oprávněnému uživateli.

Po zapnutí systému se po uživateli požaduje, aby přiložil tag. Pokud uživatel přiloží nesprávný tag, tak se na LCD displeji zobrazí nápis „nesprávný tag, přístup zamítnut“ rozezní se bzučák a rozsvítí se červená LED dioda, po uplynutí 3000 ms se bzučák vypne a LED dioda zhasne. Na sériovém monitoru se zobrazí nápis „nesprávný tag“, ID nesprávného tagu a „přístup zamítnut“. Při přiložení správného tagu se na LCD displeji zobrazí nápis „správný tag“ a rozsvítí se první zelená LED dioda, která zůstane svítit. Na sériovém monitoru se zobrazí typ přiloženého čipu nebo karty.

Následně je uživatel vyzván, aby zadal heslo. Pokud uživatel zadá špatné heslo, tak se na LCD displeji zobrazí nápis „špatné heslo, přístup zamítnut“. Rozezní se bzučák a rozsvítí se červená LED dioda, po uplynutí 3000 ms se bzučák vypne a LED dioda zhasne. Na sériovém monitoru se zobrazí špatné heslo, které uživatel zadal a zobrazí se nápis „přístup zamítnut“. Při zadání správného hesla se zobrazí na LCD displeji nápis „správné heslo“. Rozsvítí se druhá zelená LED dioda. Na sériovém monitoru se zobrazí, zda se jedná o master, user nebo guest heslo. Každý tag má přidělené své heslo a nelze je zaměnit.

Poté je uživatel vyzván k přiložení otisku prstu. Pokud uživatel přiloží prst, který je uložený v paměti čtečky, ale není pro něj povolen přístup, tak se na LCD displeji zobrazí nápis „nesprávný otisk, přístup zamítnut“. Na sériovém monitoru se zobrazí #(číslo) otisku, který uživatel přiložil. V případě, že prst přiloží cizí uživatel, jehož otisk není uložen v paměti čtečky, tak se na LCD displeji zobrazí nápis „neznámý otisk, přístup zamítnut“. Jestliže uživatel přiloží správný prst, tak se na LCD displeji zobrazí nápis „správný otisk, přístup povolen“. Rozsvítí se třetí zelená LED dioda a dveře jsou otevřeny po dobu 4000 ms. Po uplynutí

prodlevy zelené LED diody zhasnou. Na sériovém monitoru se zobrazí #(číslo) otisku, který byl přiložen. A program se vrací na začátek, kde se čeká na přiložení tagu.

Přidání nového otisku do paměti čtečky otisků prstů, bude probíhat pomocí MASTER karty. Při přiložení karty se na LCD displeji zobrazí nápis "MASTER karta, probíhá program přidání otisku". Vkládání bude probíhat přes sériový monitor, kde si uživatel zvolí pod jakým #(číslo) se daný otisk uloží. Uložení otisku je možné pro # 1-127, při zadání vyššího čísla se program vrací na začátek. Po přiložení prstu na čtečku a následném uložení otisku se na LCD displeji zobrazí nápis „otisk uložen". Následně pak stačí pro daný # povolit přístup. V programu jsou přednastavené otisky s #1, #2 pro žlutý čip, s #3, #4 pro modrý čip a pro správnou kartu jsou otisky #5, #6. Další otisky je potřeba nejdříve nadefinovat v programu. V případě špatné komunikace se čtečkou se na sériovém monitoru zobrazí nápis „chyba komunikace". Při načtení chaotického snímku se na sériovém monitoru zobrazí nápis „chaotický snímek". Po přidání nového otisku, nebo v případě nějaké chyby se program vrací na začátek, kde dochází ke kontrole čipu nebo karty.

K odstranění uloženého otisku poslouží REMOVE karta. Při přiložení karty se na LCD displeji zobrazí nápis "REMOVE karta, probíhá program odstranění otisku". Odstranění bude probíhat přes sériový monitor, uživatel si zvolí daný #(číslo), pod kterým je uložený otisk a chce jej odstranit. Pokud bude zadáno větší číslo než 127, tak se program vrací na začátek. Po odstranění se na LCD displeji zobrazí nápis „otisk odstraněn". Pokud by nebyla navázána komunikace se čtečkou, tak se na sériovém monitoru zobrazí nápis „chyba komunikace". Jakmile je otisk odstraněn, nebo nastane chyba komunikace, tak se program vrací na začátek, kde dochází ke kontrole čipu nebo karty.

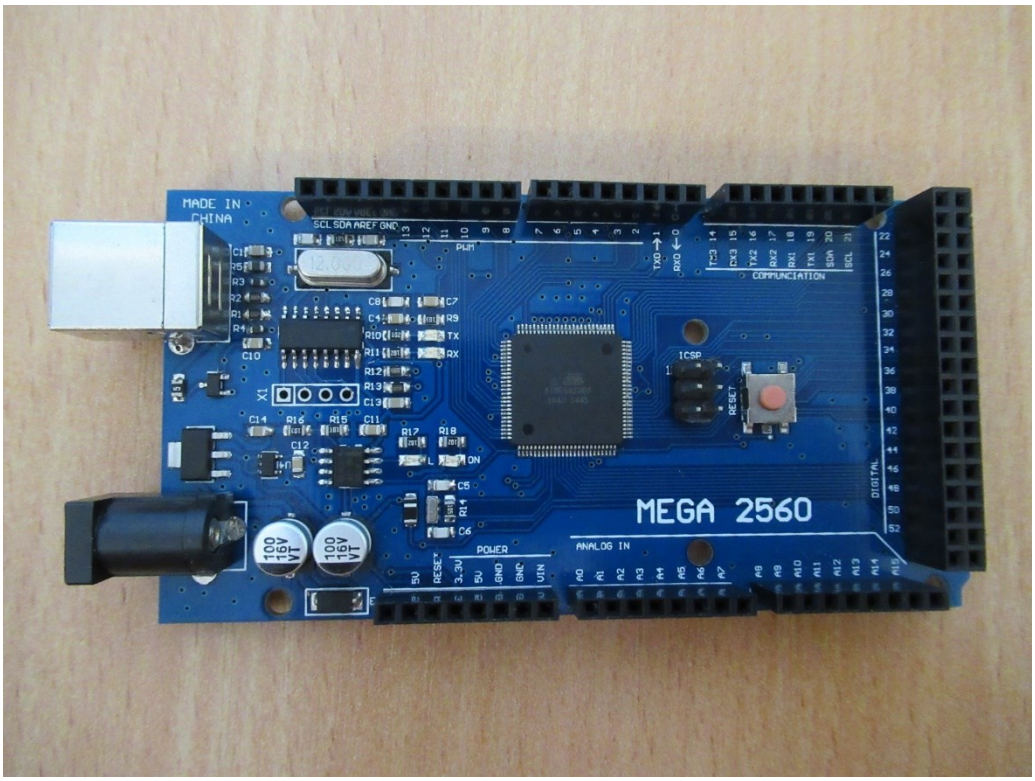
Vymazání paměti čtečky otisků prstů bude probíhat pomocí EMPTY karty. Při přiložení karty se na LCD displeji zobrazí nápis "EMPTY karta, probíhá program vymazání paměti". Po vymazání paměti se na LCD displeji zobrazí nápis „paměť vymazána". Pokud by nastala chyba při navazování spojení se čtečkou, tak se na sériovém monitoru zobrazí nápis „chyba komunikace". Následně se program vrací na začátek, kde dochází ke kontrole čipu nebo karty.

3.1 Použité komponenty

Pro sestavení přístupového systému posloužily níže uvedené komponenty. U každé součástky se nachází fotografie, popis součástky a specifikace.

3.1.1 Arduino Mega

Na první pohled je patrné, že se nejedná o originální desku, ale o klon Arduino Mega. Nicméně deska je úplně stejná jako originál, rozdíl lze zaznamenat např. že se na desce nenachází nápis Arduino a specifický znak. Dalším rozdílem je odstín barvy desky, který je oproti originální desce více tmavší. Jedná se o největší model základové desky, kterou Arduino vyrábí. Základem desky je procesor ATmega2560 s taktovací frekvencí 16 MHz. Deska Arduino Mega obsahuje 54 digitálních vstupů/výstupů což poslouží pro uvedenou sestavu. Na desce se nachází I2C rozhraní, jedná se o dva piny SCL a SDA, které jsou nezbytné pro připojení LCD displeje. K programování se využije USB port typu B, pomocí kterého nastane propojení desky s počítačem. Během propojení s počítačem bude deska trvale napájena napětím. Provozní napětí, se kterým deska pracuje je 5 V, na desce také lze nalézt port 3,3 V pro připojení určitých komponentů např. RFID čtečky. [23]



Obr. 16. Arduino Mega.

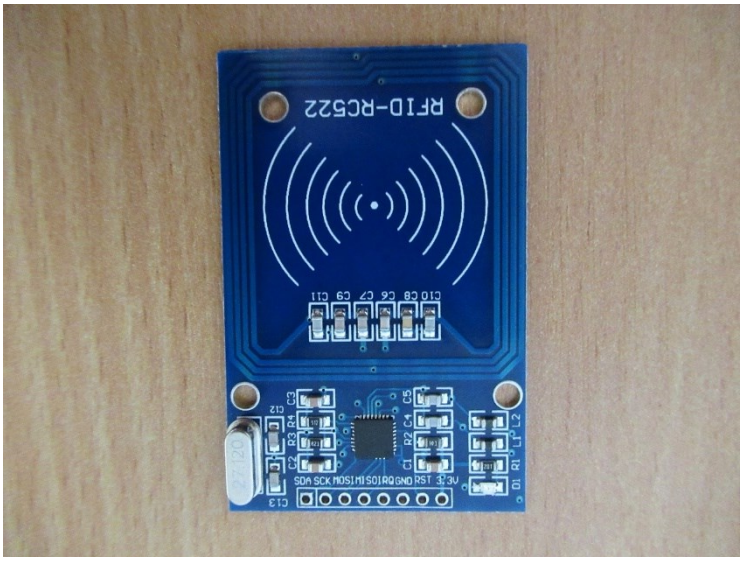
3.1.2 RFID čtečka karet RC522

RFID čtečka pracuje na frekvenci 13,56 MHz. Obsahuje integrovanou anténu umístěnou na základní desce. Napájecí napětí čtečky je 3,3 V s provozním proudem 13 až 26 mA. Teplota, při které čtečka pracuje se pohybuje kolem -20 až 80 °C. Uvedená RFID čtečka je

kompatibilní jak s Arduinem, tak i s Raspberry Pi. Na čtečce lze nalézt celkem osm vývodů pro připojení. Komunikace probíhá přes Serial Peripheral Interface (dále jen SPI) sběrnici. V tabulce jsou uvedeny vývody, které se na čtečce nachází a piny, které lze využít k propojení. [24]

Tab. 20. Zapojení vývodů z RFID čtečky na piny Arduina

vývody na RFID čtečce	**piny na Arduinu**
VCC	3,3 V
GND	GND
RESET	48
SDA (synchronní data)	53
SCK (clock)	52
MOSI (master OUT/slave IN)	51
MISO (master IN/slave OUT	50
IRQ (přerušení)	není zapojeno

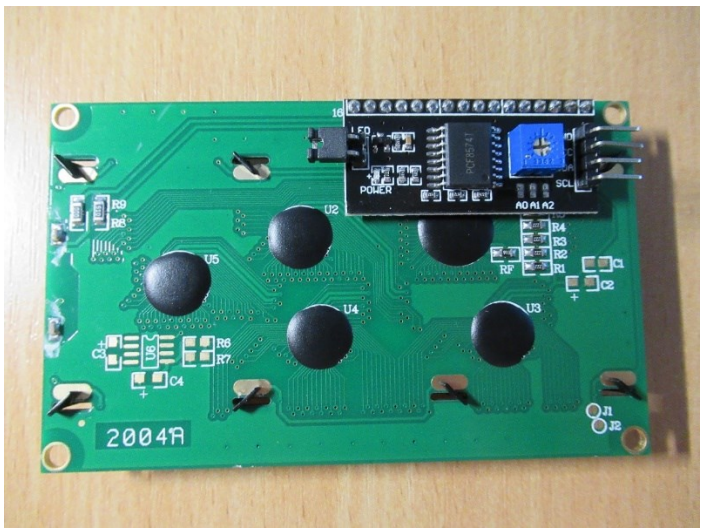


Obr. 17. RFID čtečka RC522.

3.1.3 LCD display

LCD displej je výstupním zařízením, na kterém lze zobrazit přednastavené hlášky. Displej dovede zobrazit 20 znaků po čtyřech řádcích. Napájení se zajistí pomocí 5 V. Tento displej obsahuje na zadní straně I2C převodník, což je velkou výhodou při připojování pinů. Jelikož při použití převodníku stačí zapojit pouze piny SCL a SDA. Samozřejmě nelze opomenout, připojení napájení a uzemnění. Jak bylo uvedeno výše, pro vývody SCL a SDA slouží na desce Arduina piny se stejným označením. Komunikace s Arduinem probíhá přes TWI rozhraní. Na I2C převodníku se nachází malá modrá součástka, jedná se o potenciometr, kterým lze nastavovat žluté podsvícení displeje. Při koupi nového displeje bývá většinou podsvícení na nule, takže není nic vidět. [25]

Obr. 18. LCD displej.



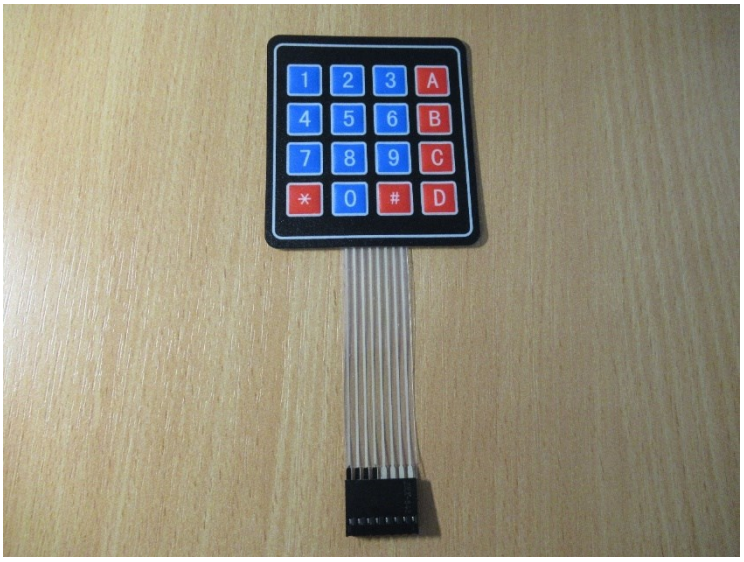
Obr. 19. LCD displej a I2C převodník.

3.1.4 Membránová klávesnice 4x4

Klávesnice obsahuje celkem 16 tlačítek v matici 4x4. Jedná se o vstupní zařízení, pomocí kterého uživatel zadává vstupní hodnoty. Zadané hodnoty jsou následně zpracovány mikroprocesorem. Klávesnice disponuje celkem 8 vývody. Její životnost se udává 100 milionů stisknutí. Odezva při stisku tlačítka se udává 1 ms. V tabulce je uvedené zapojení vývodů z klávesnice na piny Arduina. [26]

Tab. 21. Zapojení vývodů z klávesnice na piny Arduina

vývody na klávesnici	piny na Arduinu
1	2
2	3
3	4
4	5
5	9
6	6
7	7
8	8



Obr. 20. Membránová klávesnice 4x4.

3.1.5 Servo motor SG92R

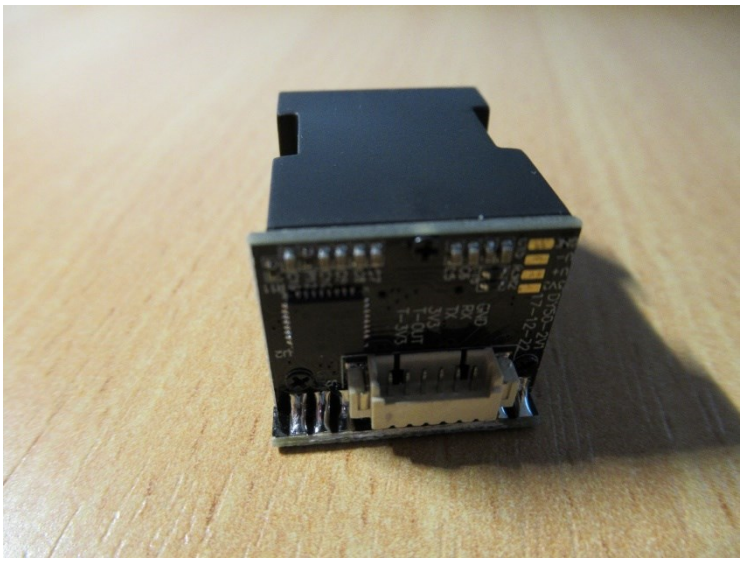
Jedná se o výstupní zařízení, které se využívá např. při ovládání RC modelů. Používá se k natočení do určité polohy a udržení v dané poloze. Napájecí napětí servo motoru je 5 V. Tyto motory umožňují obvykle otáčení v rozsahu 0 až 180°. Servo obsahuje tři vývody: napájení, uzemnění a poslední pin je připojen na výstupní pin 31 Arduina. [27]

Obr. 21. Servo motor SG92R.

3.1.6 Čtečka otisků prstů

Čtečka pracuje s napájecím napětím 3,3 V. Teplota, při níž může čtečka pracovat se pohybuje v rozmezí od -20 až do 50 °C. Čtečka obsahuje vnitřní paměť, do které se ukládají otisky prstů. Paměť dovede zaznamenat až 162 otisků prstů. Odezva snímání otisku by měla být menší než 1 s. Po připojení čtečky k napájecímu napětí začne průhledné místo pro vložení prstu svítit zeleně, to signalizuje, že je čtečka aktivní. Na čtečce se nachází šest vývodů. Napájení, uzemnění, RX – vyslaný signál, zapojen na pin 10, TX – přijímaný signál, zapojen na pin 11. Další dva vývody T-OUT a T-3V3 v následujícím případě není potřeba zapojovat, slouží pro přesnější snímání. [28]

Obr. 22. Čtečka otisků prstů.



Obr. 23. Čtečka otisků prstů, připojovací konektor.

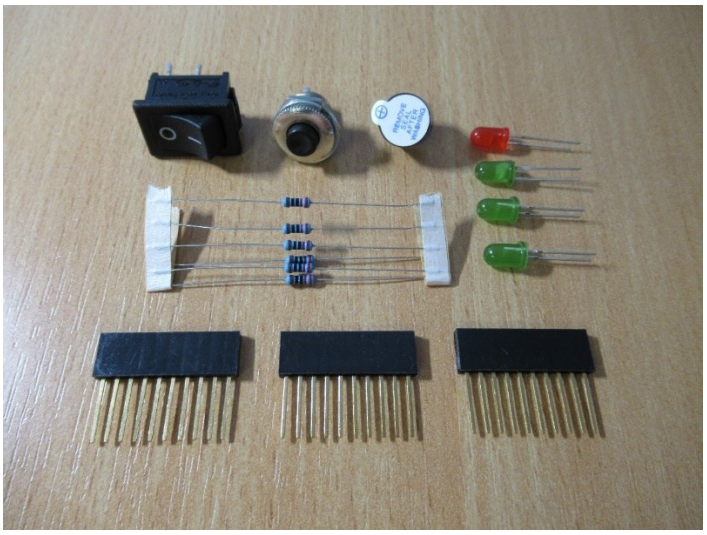
3.1.7 Ostatní komponenty

Dále byly použity LED diody, červená pro signalizaci špatného tagu, hesla a otisku. A tři zelené pro signalizaci správného tagu, hesla a otisku. Anoda LED diody se přes rezistor 220R připojí na výstupní pin Arduina, rezistor se zde nachází z důvodu, aby nedošlo k odpálení LED diody. Katoda se následně připojí na uzemnění. Zvuková signalizace bude zajištěna pomocí bzučáku s frekvencí 2,3 kHz. Bzučák obsahuje dva vývody, jeden vývod tvoří uzemnění a druhý vývod se připojí na výstupní pin Arduina s označením 33. Pro celkový reset prototypu poslouží externě připojené tlačítko. Svorkovnice pro GND, 5 V a 3,3

V budou řešeny s využitím stohovatelné 10 pinové dutinkové lišty, jejíž konce se navzájem proletují. [29]

Tab. 22. Zapojení LED diod na piny Arduina

LED diody	**piny na Arduinu**
červená	29
zelená_tag	27
zelená_heslo	25
zelená_otisk	23

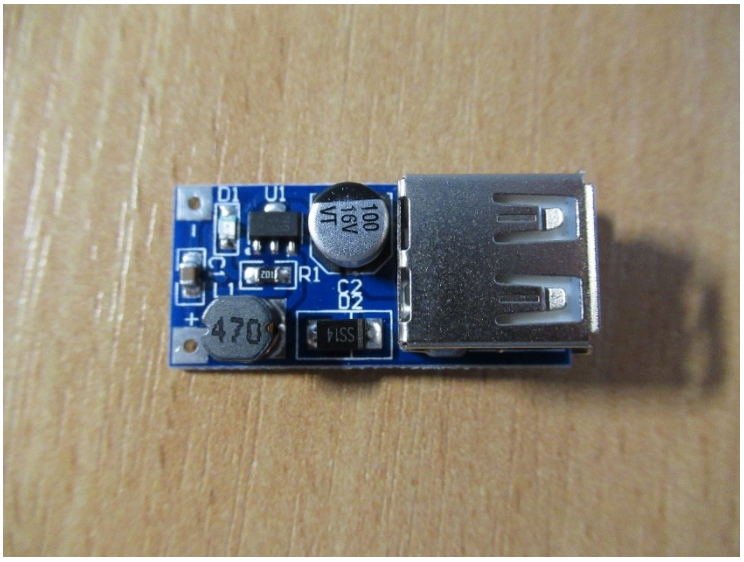


Obr. 24. Ostatní komponenty.

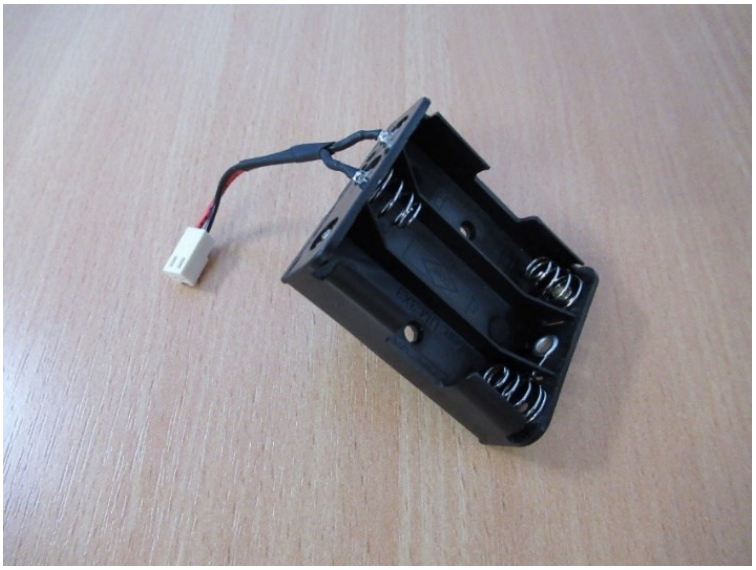
Obr. 25. Propojovací kabely.

3.2 Napájení prototypu

Napájení celého přístupového systému je řešeno pomocí 3 x AA baterií a step-up měniče, který dovede napájet zařízení 5 V. Na modulu se nachází dva vývody označené + a -, na tyto vývody se připojí box na baterii. Pomocí USB portu typu A se přes USB kabel připojí přístupový systém. Vstupní napětí pro měnič je v rozmezí 0,9 až 5 V. Na měniči je osazena LED dioda, která signalizuje připojení zdroje napětí. Na kladném vodiči je umístěn spínač, který zajistí funkci zapínacího a vypínacího tlačítka pro celý systém v případě napájení přes AA baterie. [30]



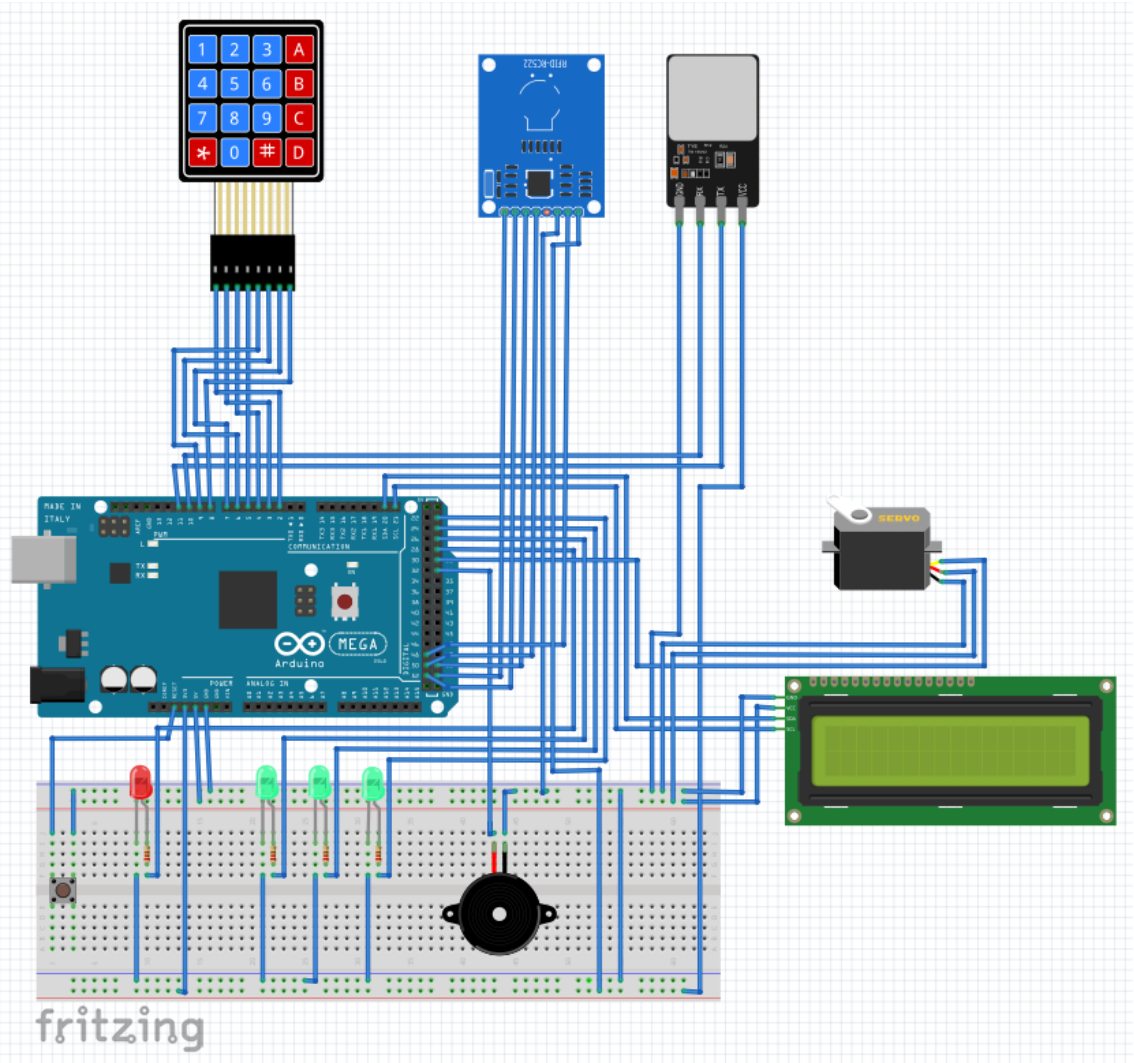
Obr. 26. Step-up měnič.



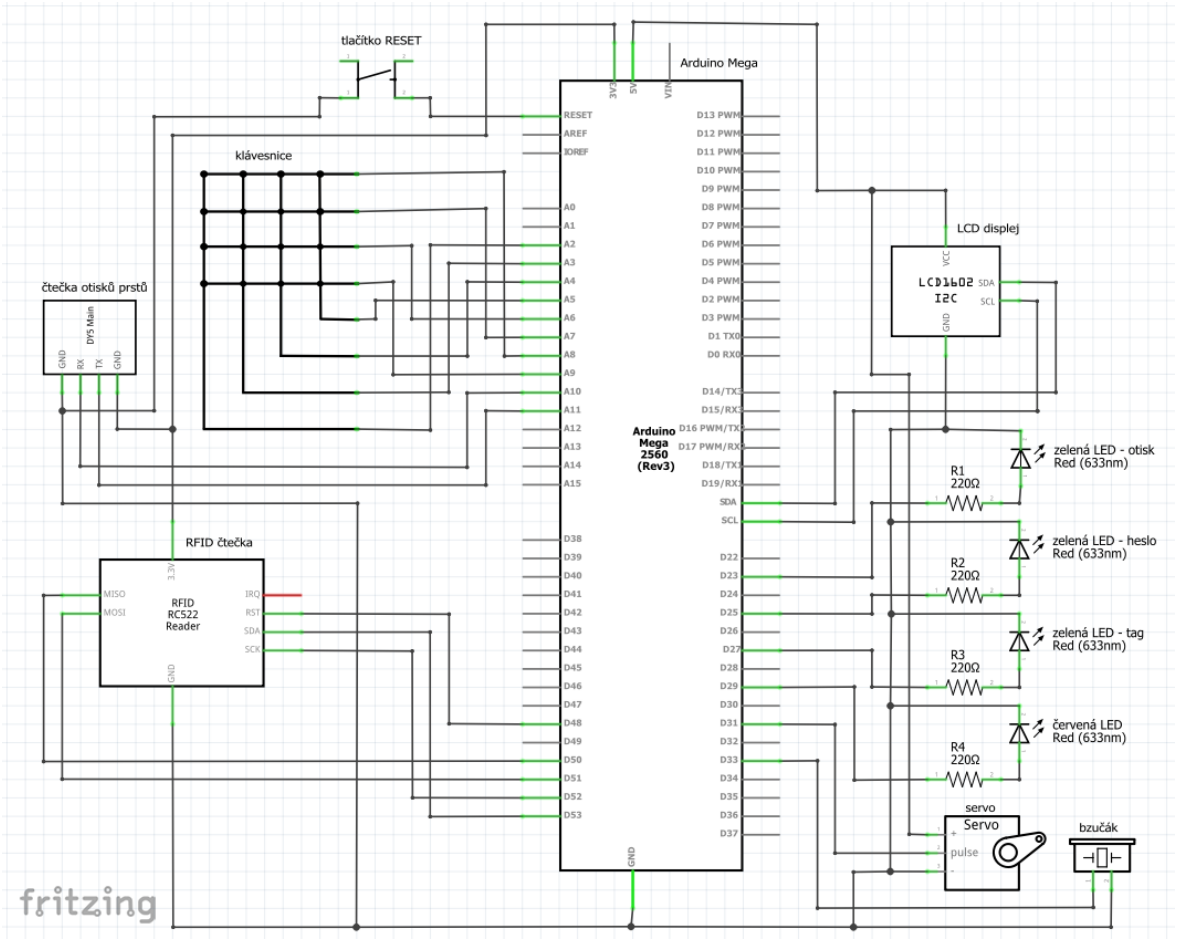
Obr. 27. Box na baterie.

3.3 Schéma prototypu

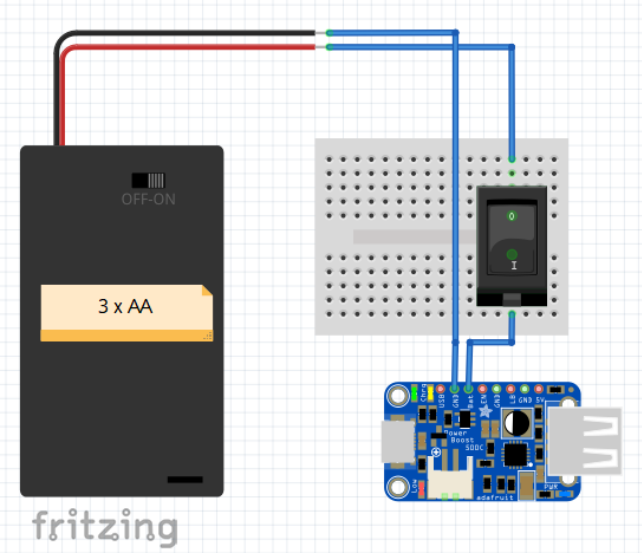
Pro tvorbu schématu posloužil volně stažitelný software Fritzing [31]. Jelikož většina součástek ve stažené knihovně nebyla obsažena, bylo potřeba dané součástky importovat.



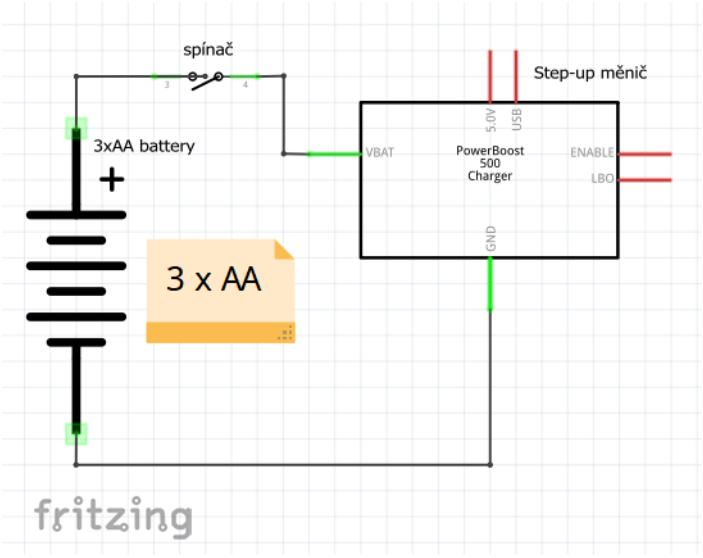
Obr. 28. Schéma zapojení.



Obr. 29. Schéma zapojení.



Obr. 30. Schéma připojení step-up měniče.

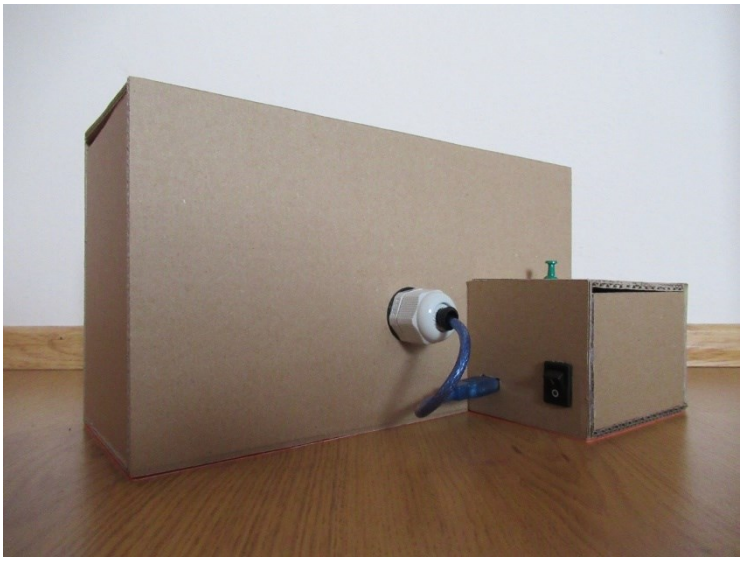


Obr. 31. Schéma připojení step-up měniče.

3.4 Vytvořený prototyp

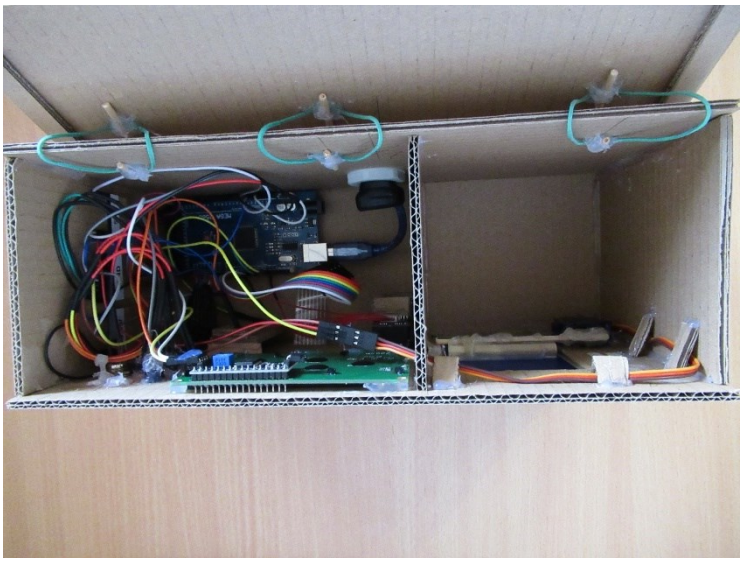
Samotný prototyp systému kontroly vstupu byl vytvořen převážně z kartonové proložky tloušťky 5 mm. Tento materiál byl zvolen z důvodu nízké hmotnosti a snadné manipulace.

Obr. 32. Přední strana prototypu.



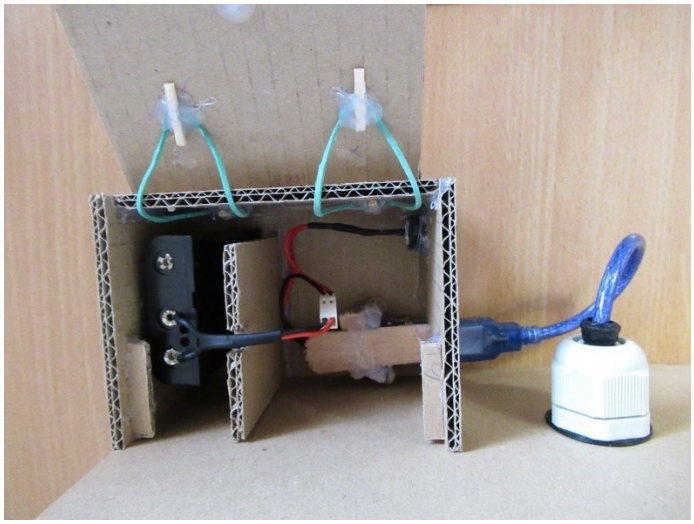
Obr. 33. Zadní strana prototypu.

Menší část slouží jako zdroj energie, zde jsou uloženy baterie a hlavní spínač. Pro připojení obvodu ke zdroji napětí slouží USB kabel, který je vyveden z hlavní části přes kabelovou průchodku PG 16. Tento USB kabel slouží také k propojení s počítačem.



Obr. 34. Rozmístění komponentů uvnitř prototypu.

Vnitřní prostor prototypu je rozdělena na dvě části. Levá část obsahuje základní desku Arduina, RFID čtečku a další komponenty, v pravé části se nachází pouze servo motor, který ovládá dveře.



Obr. 35. Uložení bateriového boxu a step-up měniče.

Obr. 36. Karty k ovládání prototypu.

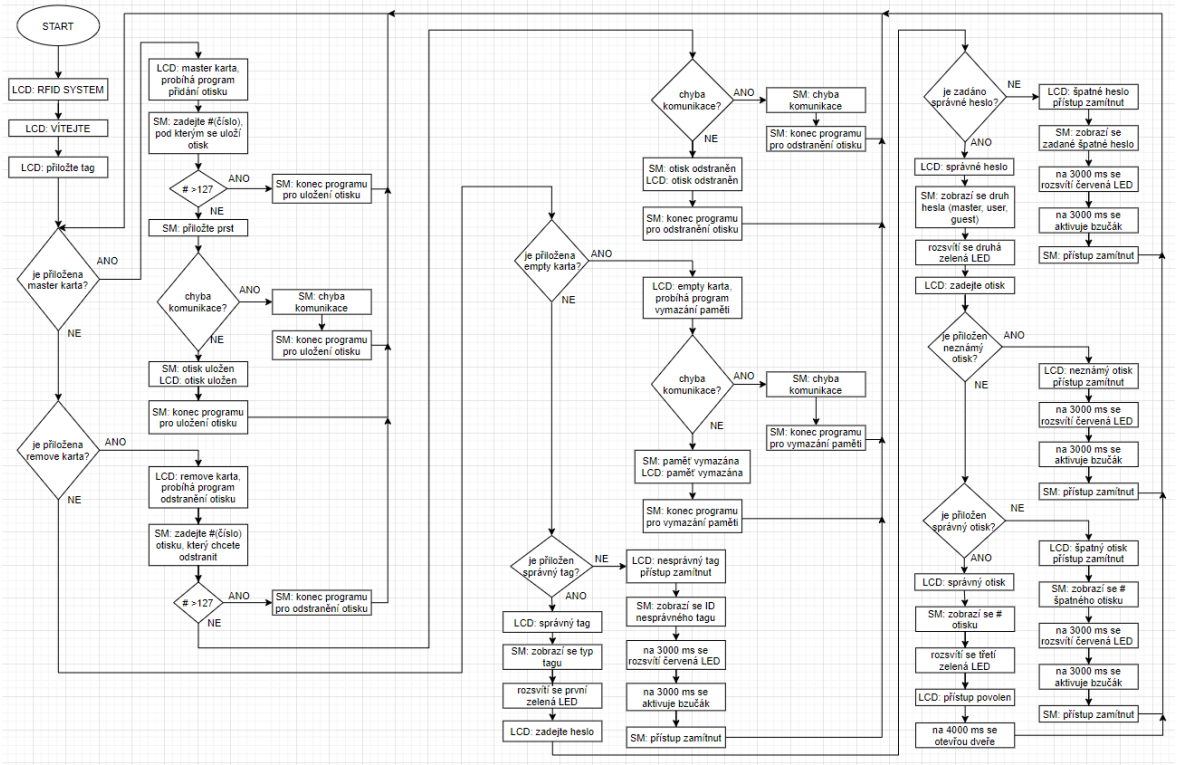
Obr. 37. Tagy k ovládání prototypu.

4 PROGRAMOVÉ VYBAVENÍ

V následující kapitole bude nejdřív uveden vývojový diagram celého programového vybavení. Následně zde budou popsány jednotlivé části zdrojového kódu.

4.1 Vývojový diagram

Před začátkem vlastní tvorby programu byl vytvořen vývojový diagram. Pro sestavení vývojového diagramu posloužila webová stránka Diagrams.net [32], která je ideální volbou pro návrhy různých vývojových diagramů. Ve vývojovém diagramu lze nalézt část pro MASTER kartu, část pro REMOVE kartu, část pro EMPTY kartu. Dále se v diagramu nachází samotný program kontroly tagu, hesla a otisku. Ve vývojovém diagramu jsou podrobně popsány jednotlivé kroky programu.



Obr. 38. Vývojový diagram.

4.2 Popis zdrojového kódu

Před začátkem smyčky void setup je nutné provést základní nastavení. Nejdřív se provede importování knihovny pro RFID čtečku, LCD displej, klávesnici, servo motor a čtečku otisků prstů. Dále je nutné nastavení potřebných pinů pro komunikaci. Následně definování tagů, pro které je povolen přístup. Taktéž otisky a povolená hesla.

Ve smyčce void setup se nadefinují výstupní piny pro LED diody, servo motor a bzučák. Nachází se zde také nastavení podsvícení pro LCD displej, nebo servo motor.

Pro zajištění správného pořadí ověřování posloužily boolean stavy. Na začátku smyčky void loop probíhá první kontrola stav_1, zda je „true“. Jedná se o stav, který vrátí program vždy na začátek. V případě správného tagu se stav_1 změní na „false“, pro nesprávný tag je stav_1 „true“. Před kontrolou zadaného hesla se ověřuje boolean stav_1, pokud je „false“ tak se pokračuje na kontrolu hesla. V případě zadaného správného hesla je zde využit druhý stav_2, který se nastaví na „true“, pro nesprávné heslo je stav_1 „true“. Před kontrolou zadaného otisku probíhá ověření boolean stav_2, pokud je „true“, tak se pokračuje dál. Po zadání otisku se vždy stav_1 nastaví na „true“ a program se vrátí na kontrolu karty. Boolean stavy také byly využity pro přiřazení otisků k tagům.

Celý zdrojový kód je celkem rozdělen do tří částí. První část slouží ke kontrole přiloženého tagu. Kontrola probíhá pomocí funkce if, celkem se nabízí pět různých variant. Správný tag, MASTER karta, REMOVE karta, EMPTY karta a nesprávný tag. Pro MASTER, REMOVE a EMPTY karty byly vytvořeny tři samostatné podprogramy, umístěné pod smyčkou void loop. Druhá část se zabývá kontrolou hesla, nejdřív se zjišťuje, zda souhlasí počet znaků. Následně se porovnává zadané heslo s heslem uloženým v paměti. Třetí část kódu je zaměřená na kontrolu otisku prstu, kontrola probíhá pomocí funkce if, a lze definovat tři stavy. Neznámý otisk, správný otisk a špatný otisk. Rozdíl mezi špatným a neznámým otiskem spočívá v tom, že špatný otisk je uložen v paměti čtečky, ale není pro něj povolen přístup. Oproti tomu neznámý otisk v paměti čtečky uložen není. Následně zde budou uvedeny klíčové části zdrojového kódu nezbytného pro správnou funkčnost celého systému. [33], [34]

4.2.1 Importování knihoven

Zde probíhá importování knihoven pro RFID čtečku, LCD displej, klávesnici, servo motor a čtečku otisků prstů. Aby bylo možné knihovny importovat, tak byly nejdříve staženy ze stránek Arduina.

```
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <Servo.h>
#include <SPI.h>
#include <Adafruit_Fingerprint.h>
```

Obr. 39. Kód – knihovny.

4.2.2 Nastavení pro vybrané komponenty

Pro čtečku otisků prstů bylo třeba definovat piny pro RX a TX. Pin RX, který zajišťuje vyslaný signál je zapojen na pin 10 a pin TX, který zajišťuje přijímaný signál na pin 11.

```
SoftwareSerial mySerial(11, 10);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
```

Obr. 40. Kód – nastavení čtečky otisků prstů.

Zobrazení textu zajišťuje LCD displej typu 0x27 se čtyřmi řádky o dvaceti znacích. U RFID čtečky se nastavil vývod SDA na pin 53 a vývod RESET na pin 48.

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
MFRC522 mfrc522(53, 48);
Servo sg90;
```

Obr. 41. Kód – nastavení displeje a čtečky.

4.2.3 Nastavení otisků, pro které je povolený přístup

Pomocí MASTER karty lze přidat nové otisky do systému. Pokud uživatel při zadávání nového otisku bude definovat # 1–6, tak je automaticky pro otisk přístup povolen. Pokud ale uživatel bude definovat jiný #, např. 8, tak se otisk do paměti uloží, ale nebude pro něj povolen přístup. V takovém případě musí uživatel ještě obdobným způsoben nadefinovat nový otisk s # 8 a přiřadit daný otisk k tagu.

```
# define OTISK_1 1
# define OTISK_2 2
# define OTISK_3 3
# define OTISK_4 4
# define OTISK_5 5
# define OTISK_6 6
```

Obr. 42. Kód – otisky.

4.2.4 Nastavení ID tagů, pro které je povolen přístup

K prototypu jsou dodány 3 kusy RFID čipů a 5 kusů RFID karet. Pro dva čipy je přístup povolen a pro jeden čip přístup povolen není. Tři karty slouží pro nastavení otisků: MASTER karta, REMOVE karta a EMPTY karta. Jedna karta má přístup povolený a pro poslední kartu je přístup zamítnut.

```
//nastavení UID tagu, pro které je povolen přístup a přiřazení otisků
# define tag_1  "C0 67 A1 2B"    // žlutý čip     -> OTISK_1, OTISK_2 -> guest_heslo
# define tag_2  "00 39 55 D3"    // modrý čip     -> OTISK_3, OTISK_4 -> user_heslo
# define tag_3  "D6 E6 93 C9"    // správná karta -> OTISK_5, OTISK_6 -> master_heslo

# define tag_4  "C6 A7 91 C9"    // REMOVE karta
# define tag_5  "06 D3 A2 C9"    // MASTER karta
# define tag_6  "16 68 8E C9"    // EMPTY karta
```

Obr. 43. Kód – nastavení povolených tagů.

4.2.5 Nastavení hesla, pro které je povolen přístup

K modrému tagu je přiřazené user heslo, ke žlutému tagu guest heslo a k správné kartě master heslo.

```
char user_heslo[4] = {'1', '0', '3', '0'};

char guest_heslo[4] = {'1', '2', '2', '3'};

char master_heslo[4] = {'A', 'D', '#', '0'};
```

Obr. 44. Kód – nastavení povolených hesel.

4.2.6 Kontrola přiloženého tagu

První funkce *if* slouží k nalezení tagu, který uživatel přiloží ke čtečce. Pokud je podmínka splněna, tak se přechází na druhou funkci, která zajišťuje čtení identifikačního údaje z tagu. Do proměnné tag se uloží jedinečné ID tagu, který uživatel přiložil na čtečku. Následně se provede kontrola, zda přiložený tag odpovídá definovanému formátu. Identifikační údaj musí být v hexadecimálním formátu a obsahovat celkem osm znaků dělených po dvojicích.

```
 // slouží pro hledání tagu
if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;
}

// čtení tagu
if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;
}

//čtení z tagu
String tag = "";  // do proměnné tag se uloží ID tagu, který byl přiložen

// kontrola, zda ID tagu odpovídá předepsanému formátu
for (byte k = 0; k < mfrc522.uid.size; k++)
{
  tag.concat(String(mfrc522.uid.uidByte[k] < 0x10 ? " 0" : " "));
  tag.concat(String(mfrc522.uid.uidByte[k], HEX));
}
tag.toUpperCase();  // převádí ID tagu na velká písmena
```

Obr. 45. Kód – přiložení tagu.

```
if (tag.substring(1) == (tag_1)|| tag.substring(1) == (tag_2)|| tag.substring(1) == (tag_3))
```

Obr. 46. Kód – porovnání tagu.

Po převedení ID tagu na velká písmena dochází pomocí funkce if ke kontrole, zda ID přiloženého tagu souhlasí s uloženým ID tagu v systému. Pokud údaj souhlasí, zobrazí se určená hláška a rozsvítí se zelená LED dioda. V opačném případě se rozezní bzučák, rozsvítí se červená LED dioda a zobrazí se definovaná hláška.

4.2.7 Kontrola zadaného hesla

Pokud je stav_1 roven hodnotě *false*, což znamená, že byl přiložen správný tag, dochází ke kontrole zadaného hesla. Heslo_stisk je akce, která se vyvolá při každém stisku klávesy. Do proměnné *heslo*, se uloží hodnoty zadané uživatelem. Následně se kontroluje čtyřmístná délka hesla.

```
  if (stav_1 == false)
  {
heslo_stisk = tl_stisk.getKey();
if (heslo_stisk)
{
  heslo[h++] = heslo_stisk;
  lcd.print("x");
}
if (h == 4)
{
```

Obr. 47. Kód – zadání hesla.

Po zadání čtyř znaků se provede porovnání s hesly, která jsou uložena v systému.

```
// heslo: toto heslo zadá uživatel
// user_heslo: (1030), je přiřazené k modrému čipu
// guest_heslo: (1223), je přiřazené k žlutému čipu
// master_heslo: (AD#0), je přiřazené k správné kartě
if ((stav_z == true && (!(strncmp(heslo, guest_heslo, 4))))
|| (stav_m == true && (!(strncmp(heslo, user_heslo, 4))))
|| (stav_k == true && (!(strncmp(heslo, master_heslo, 4)))))
```

Obr. 48. Kód – porovnání hesla.

4.2.8 Kontrola přiloženého otisku

Pokud je stav_2 roven hodnotě *true,* to znamená, že i zadané heslo je správné, tak nastává kontrola otisku. Kontrola otisku probíhá pomocí tří funkcí *if*. První funkce vyfotí prst přiložený na čtečku.

```
k = finger.getImage();
if (k == FINGERPRINT_OK)
{
  // pokud proběhlo v pořádku, tak pokračuje dál
}
else
{
 return k;
}
```

Obr. 49. Kód – vyfocení otisku.

Druhá funkce se zabývá převodem snímku na šablonu funkce. Pokud každá ze tří funkcí proběhne v pořádku, program pokračuje dál. V opačném případě se proces vrací na začátek ověření.

```
k = finger.image2Tz();
if (k == FINGERPRINT_OK)
{
  // pokud proběhlo v pořádku, tak pokračuje dál
}
else
{
 return k;
}
```

Obr. 50. Kód – převedení snímku na šablonu funkce.

Třetí část slouží k prohledání aktuální funkce otisků, které musí odpovídat uloženým šablonám. Shodný otisk je uložen ve fingerID.

```
k = finger.fingerFastSearch();
if (k == FINGERPRINT_OK)
{
  // pokud proběhlo v pořádku, tak pokračuje dál
}
```

Obr. 51. Kód – shodný otisk je uložen ve fingerID.

Pokud je šablona funkce shodná s dříve uloženou šablonou otisku, je přístup povolen. V opačném případě je přístup zamítnut.

```
// žlutý čip - OTISK_1, OTISK_2
// modrý čip - OTISK_3, OTISK_4
// správná karta - OTISK_5, OTISK_6
else if ((stav_z == true && (finger.fingerID == OTISK_1 || finger.fingerID == OTISK_2))
      || (stav_m == true && (finger.fingerID == OTISK_3 || finger.fingerID == OTISK_4))
      || (stav_k == true && (finger.fingerID == OTISK_5 || finger.fingerID == OTISK_6)))
```

Obr. 52. Kód – porovnání a přiřazení otisků.

Při povolení přístupu nastane otočení servo motoru o 90°, čím je zajištěno otevření dveří. Souběžně se rozsvítí třetí zelená LED dioda. Po uplynutí časové prodlevy se dveře uzavřou a LED diody zhasnou.

```
sg90.write(0);
digitalWrite(zelena_otisk_LED, HIGH);
delay(PRODLEVA_4);
sg90.write(90);
stav_1 = true;
```

Obr. 53. Kód – správný otisk.

Po přiložení nesprávného nebo cizího otisku se rozezní bzučák a rozsvítí se červená LED dioda.

```
digitalWrite(BZUCAK, HIGH);
digitalWrite(cervena_LED, HIGH);
delay(PRODLEVA_3);
digitalWrite(BZUCAK, LOW);
digitalWrite(cervena_LED, LOW);
lcd.clear();
stav_1 = true;
```

Obr. 54. Kód – nesprávný otisk.

5 OVĚŘENÍ FUNKČNOSTI

Následující kapitola se zabývá ověřením všech funkcí, které byly dříve specifikovány. Jsou zde demonstrovány jednotlivé stavy, které by mohly nastat za běžného provozu. Na fotografiích lze nalézt hlášky z LCD displeje, které se zobrazují při provozu. Souběžně s fotkami jsou zde uvedeny také screeny ze sériového monitoru. Na sériovém monitoru je uvedeno vždy o něco více informací než na LCD displeji, jelikož u sériového monitoru je omezený zobrazovací prostor. Na konci kapitoly se nachází testování poplachů False Acceptance Rate (dále jen FAR) a False Rejection Rate (dále jen FRR).

5.1 Správná karta, správné heslo, správný otisk

Po zapnutí prototypu se zobrazí nápis přiložte tag, po přiložení správného tagu, je uživatel vyzván, aby zadal heslo. Při zadání správného hesla, je uživatel systémem vyzván, pro přiložení prstu na čtečku. Jakmile je přiložen správný prst, tak se dveře otevřou.

Obr. 55. Displej – přiložte tag.

Obr. 56. Displej – správný tag.

Obr. 57. Displej – zadejte heslo.

Obr. 58. Displej – správné heslo.

Obr. 59. Displej – zadejte otisk.

Obr. 60. Displej – správný otisk.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
použitý tag: modrý čip
použité heslo: user
použitý otisk: #4
PŘÍSTUP POVOLEN
//------------------------------------------------------------------------//
```

Obr. 61. Monitor – správný tag, heslo a otisk.

Z výpisu na sériový monitor je patrné, že uživatel přiložil modrý čip, zadal heslo user a přilož otisk #4.

5.2 Správná karta, správné heslo, špatný otisk

Zde lze vidět, že uživatel přiložil otisk, který je uložen ve čtečce, ale není pro něj povolený přístup. Dvě zelené LED diody signalizují, že uživatel přiložil správný tag a zadal správné heslo.

Obr. 62. Displej – špatný otisk.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
použitý tag: modrý čip
použité heslo: user
použit nesprávný otisk: #36
PŘÍSTUP ZAMÍTNUT
//------------------------------------------------------------------------//
```

Obr. 63. Monitor – špatný otisk.

Na sériovém monitoru se nachází záznam o přiložení otisku s #36, pomocí kterého lze identifikovat uživatele.

5.3 Správná karta, správné heslo, neznámý otisk

Zde došlo k přiložení otisku cizí osobou, která nemá povolený přístup.

Obr. 64. Displej – neznámý otisk.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
použitý tag: žlutý čip
použité heslo: user
použit neznámý otisk
PŘÍSTUP ZAMÍTNUT
//------------------------------------------------------------------------//
```

Obr. 65. Monitor – neznámý otisk.

5.4 Správná karta, špatné heslo

Pokud uživatel zadá špatné heslo, na LCD displeji se zobrazí pouze hláška. Na sériovém monitoru lze zjistit, jaké heslo uživatel zadal.

Obr. 66. Displej – špatné heslo.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
použitý tag: správná karta
použito nesprávné heslo: 1379□
PŘÍSTUP ZAMÍTNUT
//------------------------------------------------------------------------//
```

Obr. 67. Monitor – špatné heslo.

5.5 Nesprávný tag

Při přiložení nesprávného tagu se v sériovém monitoru zobrazí ID tagu, který byl přiložen na čtečku.

Obr. 68. Displej – nesprávný tag.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
36 A0 86 C9: NESPRÁVNÝ TAG
PŘÍSTUP ZAMÍTNUT
//------------------------------------------------------------------------//
A0 5E E1 28: NESPRÁVNÝ TAG
PŘÍSTUP ZAMÍTNUT
//------------------------------------------------------------------------//
```

Obr. 69. Monitor – nesprávný tag.

První ID odpovídá nesprávné kartě a druhé ID odpovídá červenému čipu.

5.6 Přidání nového otisku

Pro přidání nového otisku slouží rta, na sériovém monitoru se postupně zobrazují jednotlivé řádky, podle toho, co uživatel zrovna provádí. Nachází se zde také celkový počet uložených otisků ve čtečce, který je zatím 0.

Obr. 70. Displej – MASTER karta.

Obr. 71. Displej – otisk uložen.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
MASTER karta
//------------------------------------------------------------------------//
Programová část pro přidání otisku
v paměti senzoru je uloženo:  0 otisků prstů
zadejte číslo od 1 do 127, pod kterým se uloží otisk prstu
přístup je povolen pro otisky 1,2,3,4,5,6 ostatní je potřeba nastavit
otisk bude uložen s #3
přiložte prst, který chcete uložit #3
prosím stále držte prst na čtečce
snímek převeden
vytvoření pro #3
otisk uložen!
prst můžete sejmout
//------------------------------------------------------------------------//
konec programu pro přidání otisku
můžete pokračovat přiložením identifikačního tagu na čtečku
//------------------------------------------------------------------------//
```

Obr. 72. Monitor – MASTER karta.

5.7 Odstranění uloženého otisku

Odstranění otisku je možné pouze s využitím REMOVE karty. Postup je obdobný jako u přidávání otisku.

Obr. 73. Displej – REMOVE karta.

Obr. 74. Displej – otisk odstraněn.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
REMOVE karta
//------------------------------------------------------------------------//
Programová část pro odstranění otisku
v paměti senzoru je uloženo:  3 otisků prstů
zadejte číslo od 1 do 127, pod kterým je uložen otisk, který chcete odstranit
bude odstraněn otisk s #:4
otisk odstraněn!
//------------------------------------------------------------------------//
konec programu pro odstranění otisku
můžete pokračovat přiložením identifikačního tagu na čtečku
//------------------------------------------------------------------------//
```

Obr. 75. Monitor – REMOVE karta.

5.8 Vymazání paměti čtečky otisků prstů

K vymazání celé paměti čtečky otisků prstů slouží EMPTY karta. Karta se pouze přiloží ke čtečce, po zobrazení hlášky je hotovo.

Obr. 76. Displej – EMPTY karta.

Obr. 77. Displej – paměť vymazána.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
EMPTY karta
//------------------------------------------------------------------------//
Programová část pro vymazání paměti čtečky otisků prstů
všechny otisky byly odstraněny!
//------------------------------------------------------------------------//
konec programu pro vymazání paměti čtečky otisků prstů
můžete pokračovat přiložením identifikačního tagu na čtečku
//------------------------------------------------------------------------//
```

Obr. 78. Monitor – EMPTY karta.

5.9 Nesprávná hodnota pro uložení otisku

Při ukládání nového otisku, nebo při odstranění uloženého otisku lze zadávat pouze čísla do 127. Po zadání většího čísla, otisk nepůjde uložit ani odstranit.

```
//------------------------------------------------------------------------//
Vítejte v sériovém monitoru přístupového systému Arduino
zde se zobrazí ID tagu přiložené na RFID čtečku
//------------------------------------------------------------------------//
MASTER karta
//------------------------------------------------------------------------//
Programová část pro přidání otisku
v paměti senzoru je uloženo:  4 otisků prstů
zadejte číslo od 1 do 127, pod kterým se uloží otisk prstu
přístup je povolen pro otisky 1,2,3,4,5,6 ostatní je potřeba nastavit
otisk bude uložen s #135
nesprávná hodnota pro uložení otisku
//------------------------------------------------------------------------//
konec programu pro přidání otisku
můžete pokračovat přiložením identifikačního tagu na čtečku
//------------------------------------------------------------------------//
```

Obr. 79. Monitor – nesprávná hodnota pro uložení otisku.

5.10 False Acceptance Rate – FAR

Uvedený koeficient slouží ke zjištění pravděpodobnosti, zda je cizí osoba přijata jako oprávněná. Koeficient FAR lze označit jako chybu II. druhu. Vpuštění neoprávněné osoby do objektu, může mít za následek poškození majetku nebo odcizení, lze tedy takovou chybu označit jako velmi závažnou. [6]

$$FAR = \frac{N_{FA}}{N_{IIA}} \cdot 100\ [\%] \tag{1}$$

- N_{FA} – počet chybných přijetí
- N_{IIA} – celkový počet pokusů osob bez oprávnění ke vstupu [6]

Při testování bylo využito 40 otisků od osob, kterým nebylo uděleno oprávnění ke vstupu. Všechny otisky byly vyhodnoceny správně, výsledný koeficient FAR = 0 %.

5.11 False Rejection Rate – FRR

Uvedený koeficient slouží ke zjištění pravděpodobnosti, zda oprávněná osoba je systémem odmítnuta. Koeficient FRR lze označit jako chybu I. druhu. Zamítnutí vstupu oprávněné osobě, působí pro uživatele nepříjemně, ale daná chyba nemá z pohledu bezpečnosti zásadní význam. Jedná se pouze o nevýhodu, kterou lze opravit. [6]

$$FRR = \frac{N_{FR}}{N_{EIA}} \cdot 100\ [\%] \tag{2}$$

- N_{FR} – počet chybných odmítnutí
- N_{EIA} – celkový počet pokusů osob s oprávněním ke vstupu [6]

Taktéž i v následujícím testování bylo využito 40 otisků, ale jednalo se o otisky uživatelů, kterým přístup byl povolen. Zde v jednom případě byla vyhodnocena chyba, výsledný koeficient FRR = 2,5 %. Lze tedy konstatovat, že testování v obou případech bylo úspěšné.

ZÁVĚR

Cílem bakalářské práce bylo navrhnout a realizovat prototyp elektronického systému kontroly vstupu založený na platformě Arduino. Byla zde podrobně rozebrána problematika a principy přístupových systémů, které posloužily pro pochopení základních funkcí. Získané znalosti byly využity při samotné konstrukci systému, který v jisté míře odráží funkce od reálných systémů. Uživateli je přístup umožněn pomocí RFID karty, přístupového kódu či biometrického otisku.

Úvodní část práce popisuje problematiku přístupových systémů, jako jsou základní pojmy či metody identifikace a verifikace. Kromě terminologie nabízí práce detailní popis platformy Arduino, včetně specifikací vybraných produktů vhodných pro účely bakalářské práce.

Praktická část se věnuje popisu jednotlivých komponent, zvolených pro návrh přístupového systému. K dispozici je také konstrukce prototypu napájení, včetně podrobných schémat zapojení napájecího obvodu i celého prototypu, jež byla vyrobena prostřednictvím nástroje Fritzing.

Následující část práce se zabývá programovým vybavením systému, jehož základem byl vývojový diagram vyrobený pomocí aplikace Diagrams.net. Kód byl psaný v jazyce C v prostředí Arduino IDE, k jeho chodu byly využit knihovny jednotlivých komponent.

Závěrečná část práce se věnuje ověření funkčnosti prototypu. Testování probíhalo v několika cyklech, přičemž se testovaly, jak jednotlivé komponenty, tak systém jako celek. Testováním se získávaly hodnoty FAR a FRR, jež pomohly ověřit spolehlivost systému.

Uvedený prototyp bude sloužit jako výukový model pro studenty technických předmětů k ilustraci principů a vývoje bezpečnostních systémů.

SEZNAM POUŽITÉ LITERATURY

[1] LAPKOVÁ, Dora. Režimová opatření – přednáška. 2018. Zlín: Fakulta aplikované informatiky – Univerzita Tomáše Bati ve Zlíně

[2] ČSN EN 60839-11-1 (334593): Poplachové a elektronické bezpečnostní systémy – Část 11-1: Elektronické systémy kontroly vstupu – Požadavky na systém a komponenty. 2016. Brno-Zábrdovice: Technické normy, © 2003-2020. Dostupné také z: https://shop.normy.biz/detail/94585

[3] LUKÁŠ, Luděk. Bezpečnostní technologie, systémy a management I. Zlín: VeR-BuM, 2011. ISBN 978-80-87500-05-7.

[4] DRGA, Rudolf. Elektronické bezpečnostní systémy – přednáška. 2019. Zlín: Fakulta aplikované informatiky – Univerzita Tomáše Bati ve Zlíně

[5] KOLAJA, Martin. Využití přístupových systémů v průmyslu komerční bezpečnosti. Zlín: Univerzita Tomáše Bati ve Zlíně, 2006, 67 s. Dostupné také z: http://hdl.handle.net/10563/806. Tomas Bata University in Zlín. Faculty of Applied Informatics, Ústav elektrotechniky a měření. Vedoucí práce Kindl, Jiří.

[6] ŠČUREK, Radomír. Biometrické metody identifikace osob v bezpečnostní praxi. VŠB TU Otrava, 2008.

[7] Jak fungují RFID čtečky. ESP [online]. Ústí nad Labem: ESP holding, © 2011-2014 [cit. 2020-07-10]. Dostupné z: https://esp.cz/cs/blog/funguji-rfid-ctecky

[8] VOJTĚCH, L. RFID – technologie pro internet věcí. Access server [online]. České vysoké učení technické v Praze, FEL, 12.02.2009, 18 [cit. 2020-07-10]. ISSN 1214-9675. Dostupné z: http://access.fel.cvut.cz/view.php?cisloclanku=2009020001

[9] HERŠTUS, Michal. RFID – principy, typy, možnosti použití. AUTOMA: časopis pro automatizační techniku [online]. Děčín: Automa-časopis pro automatizační techniku, 2011, 7, 64 [cit. 2020-07-10]. ISSN 1210-9592. Dostupné z: http://automa.cz/rfid-%E2%80%93-principy-typy-moznostipouziti-44083.html

[10] VOJÁČEK, Antonín. Používané RFID frekvence a jejich vliv na čtení a zápis tagu. Automatizace.hw.cz [online]. Praha 4: HW server s.r.o, 2015 [cit. 2020-07-10]. Dostupné z: https://automatizace.hw.cz/komponenty-prumyslove-sbernice-a-komunikace/vice-i-mene-bezne-rfid-frekvence-a-jejich-vliv-na-vlastnosti-tagu.html

[11] VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-93-8.

[12] Raspberry Pi 4 Model B – 4 GB RAM. RPishop.cz [online]. Roudné: RPishop.cz, © 2020 [cit. 2020-07-10]. Dostupné z: https://rpishop.cz/raspberry-pi-4b/1598-raspberry-pi-4-model-b-4gb-ram-765756931182.html

[13] Banana Pi M64. RPishop.cz [online]. Roudné: RPishop.cz, © 2020 [cit. 2020-07-10]. Dostupné z: https://rpishop.cz/banana-pi/1482-banana-pi-m64.html

[14] Arduino Mini. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-mini-05

[15] Arduino Micro. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-micro

[16] Arduino Uno Rev3. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[17] Arduino Mega 2560 Rev3. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-mega-2560-rev3

[18] Arduino Due. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-due

[19] Arduino Esplora. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-esplora

[20] Arduino Yún. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://store.arduino.cc/arduino-yun

[21] Intel Galileo. Arduino.cc [online]. Arduino, © 2020 [cit. 2020-07-12]. Dostupné z: https://www.arduino.cc/en/ArduinoCertified/IntelGalileo

[22] Polovodičové paměti [online]. Brno: SSPBrno, 2011, 29 [cit. 2020-07-10]. Dostupné z: https://moodle.sspbrno.cz/pluginfile.php/8820/mod_resource/content/1/Polovodicove_pameti.pdf

[23] Klon Arduino Mega precise. Arduino-shop.cz: IOT Vývojové platformy [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/946-eses-klon-arduino-mega-precise.html

[24] Arduino RFID čtečka s vestavěnou anténou. Arduino-shop.cz: IOT Bezdrátové periferie [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/833-arduino-rfid-ctecka-s-vestavenou-antenou.html

[25] I2C 20x4 display pro jednodeskové počítače. Arduino-shop.cz: IOT Výstupní periferie [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/1421-eses-i2c-20x4-display-pro-jednodeskove-pocitace.html

[26] Membránová klávesnice 4x4 pro jednodeskové počítače. Arduino-shop.cz: IOT Vstupní periferie [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/824-eses-membranova-klavesnice-4x4-pro-jednodeskove-pocitace.html

[27] Mini servo motor 9g 2.5kg pro RC auta, lodě a letadla SG92R. Arduino-shop.cz: IOT Výstupní periferie [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/1661-mini-servo-motor-9g-2.5kg-pro-rc-auta-lode-a-letadla-sg92r.html

[28] Čtečka otisků prstů. Arduino-shop.cz [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/1642-ctecka-otisku-prstu.html

[29] Bzučák 5 V 2.3 KHz. Arduino-shop.cz: Elektronické součástky [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/1251-bzucak-5v-2.3-khz.html

[30] Step-up měnič z 0.9V~5V na 5V USB. Arduino-shop.cz: Napájení, zdroje, měniče [online]. Havlíčkův Brod [cit. 2020-07-12]. Dostupné z: https://arduino-shop.cz/arduino/1020-step-up-menic-z-0.9v-5v-na-5v-usb.html

[31] Fritzing [online]. Berlín: Fritzing.org, © 2020 [cit. 2020-07-10]. Dostupné z: https://fritzing.org/home/

[32] Diagram.drawio. Diagrams.net [online]. Diagrams.net, © 2005-2020 [cit. 2020-07-10]. Dostupné z: https://www.diagrams.net

[33] FRIED, Limor. Reference třídy otisků prstů Adafruit_Fingerprint. Adafruit.github.io [online]. Doxygen, 1.8.13 [cit. 2020-07-10]. Dostupné z: https://adafruit.github.io/Adafruit-Fingerprint-Sensor-Library/html/class_adafruit___fingerprint.html#a828ce078a1bbd3c8295792ac946eef56

[34] MANN, Burkhard. C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy – linkery, práce s ATMEL AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky ... Praha: BEN – technická literatura, 2003. ?C & praxe. ISBN 80-7300-077-6.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ADB	Android Debuger Bridge
cm	Centimetr
ČSN	Česká technická norma
DDR3	Double-Data-Rate 3
DRAM	Dynamic Random Access Memory
EEPROM	Electrically Eraseable Programable Read-Only Memory
eMMC	Embedded MultiMediaCard
EN	Evropská norma
EPROM	Eraseable Programable Read-Only Memory
ESKV	Elektronické systémy kontroly vstupu
FAR	False Acceptance Rate
FL	Low Frequency
FRR	False Rejection Rate
GND	Ground
HF	High Frequency
I2C	Inter-Integrated Circuit
ICSP	In Circuit Serial Programming
IDE	Integrated Development Environment
kB	Kilobyte
kHz	Kilohertz
LCD	Liquid Crystal Display
LED	Light Emitting Diode
m	Metr
mA	Miliampér
MB	Megabyte

MHz	Megahertz
MISO	Master In Slave Out
ms	Milisekunda
MOSI	Master Out Slave In
PWM	Pulse Width Modulation
RAM	Random Access Memory
RFID	Radio Frequency Identification
ROM	Read Only Memory
PROM	Programable Read-Only Memory
RX	Receiver
s	Sekunda
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TWI	Two Wire Interface
TX	Transmitter
UHF	Ultra High Frequency
USB	Universal Serial Bus
VCC	Voltage Common Collector
V	Volt
°C	Stupeň Celsia

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÝ KÓD

```
// importování knihoven

#include <MFRC522.h>

#include <LiquidCrystal_I2C.h>

#include <Keypad.h>

#include <Servo.h>

#include <SPI.h>

#include <Adafruit_Fingerprint.h>


SoftwareSerial mySerial(11, 10);                    // nastavení pinů čtečky otisků prstu (Tx, Rx)

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);  // nastavení čtečky otisků prstů


LiquidCrystal_I2C lcd(0x27, 20, 4); // nastavení pro LCD display (označení LCD, počet znaků na jednom řádku, počet řádků

MFRC522 mfrc522(53, 48);            // nastavení pinů pro MFRC522 mfrc522(SDA_PIN, RESET_PIN)

Servo sg90;                         // nastavení servo motoru


// nastavení otisků, pro které je povolený přístup

# define OTISK_1 1

# define OTISK_2 2

# define OTISK_3 3

# define OTISK_4 4

# define OTISK_5 5

# define OTISK_6 6
```

```
// definování proměnných a přiřazení pinů
# define cervena_LED 29
# define zelena_tag_LED 27
# define zelena_heslo_LED 25
# define zelena_otisk_LED 23
# define SERVO 31
# define BZUCAK 33

// definování časových prodlev
# define PRODLEVA_1 200
# define PRODLEVA_2 2000
# define PRODLEVA_3 3000
# define PRODLEVA_4 4000

//nastavení user hesla: 1030
char user_heslo[4] = {'1', '0', '3', '0'};

//nastavení guest hesla: 1223
char guest_heslo[4] = {'1', '2', '2', '3'};

//nastavení master hesla: AD#0
char master_heslo[4] = {'A', 'D', '#', '0'};

//nastavení UID tagu, pro které je povolen přístup a přiřazení otisků
# define tag_1  "C0 67 A1 2B"    // žlutý čip     -> OTISK_1, OTISK_2 -> guest_heslo
```

```
# define tag_2  "00 39 55 D3"    // modrý čip     -> OTISK_3, OTISK_4 -> user_heslo
# define tag_3  "D6 E6 93 C9"    // správná karta -> OTISK_5, OTISK_6 -> master_heslo

# define tag_4  "C6 A7 91 C9"    // REMOVE karta
# define tag_5  "06 D3 A2 C9"    // MASTER karta
# define tag_6  "16 68 8E C9"    // EMPTY karta


char heslo[4];            // proměnná pro uložení hesla, které zadá uživatel
boolean stav_1 = true;    // boolean pro změnu režimů, použitý pro tag a heslo
boolean stav_2 = false;   // boolean pro změnu režimů, použitý pro otisk
boolean stav_m = false;   // boolean pro modrý čip
boolean stav_z = false;   // boolean pro žlutý čip
boolean stav_k = false;   // boolean pro správnou kartu

char heslo_stisk = 0;     // funkce, která se vyvolá při každém stisku klávesy
int h = 0;                // proměnná pro zjištění počtu znaků hesla

// definování sloupců a řádků klávesnce
const byte radky = 4;
const byte sloupce = 4;
// nastavení pinů pro klávesnici
byte radek_piny[radky] = {8, 7, 6, 9};
byte sloupec_piny[sloupce] = {5, 4, 3, 2};

// zobrazení znaků na klávesnici
```

```
char znaky_k[radky][sloupce] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// nastavení pro klávesnici, pořadí: řádky, sloupce
Keypad tl_stisk = Keypad( makeKeymap(znaky_k), radek_piny, sloupec_piny, radky, sloupce);

// ------------------------------------------------------------------------------------------
// -----------------------------------smyčka void setup---------------------------------------
// ------------------------------------------------------------------------------------------
void setup()
{
  Serial.begin(9600);   // komunikační rozhranní se sériovou linkou
  finger.begin(57600);  // komunikační rozhranní s čtečkou otisků prstů
  delay(5);             // časová prodleva

  // nastavení výstupních pinů Arduina
  pinMode(BZUCAK, OUTPUT);
  pinMode(cervena_LED, OUTPUT);
  pinMode(zelena_tag_LED, OUTPUT);
  pinMode(zelena_heslo_LED, OUTPUT);
  pinMode(zelena_otisk_LED, OUTPUT);
```

```
  sg90.attach(SERVO);   // nastavení pinu 9 pro servo
  sg90.write(90);       // nastavení počáteční pozice "90"
  lcd.begin();          // LCD display
  lcd.backlight();      // podsvícení displeje LCD
  SPI.begin();          // nastavení SPI komunikační sběrnice
  mfrc522.PCD_Init();   // MFRC522
  lcd.clear();          // vymazání LCD obrazovky na začátku

  // výpis na sériový monitor - úvodní nápis
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("Vítejte v sériovém monitoru přístupového systému Arduino");
  Serial.println("zde se zobrazí ID tagu přiložené na RFID čtečku");
  Serial.println("//--------------------------------------------------------------------//");

}
// slouží pro master kartu, kontrola zadaných znaků
// master_k zadá uživatel v sériovém monitoru
int master_k(void) {
  int mk = 0;

  while (mk == 0) {
    while (! Serial.available()); // získává (kontroluje) počet znaků dostupných ze sériové linky
    mk = Serial.parseInt();       // vrací první celé platné číslo ze sériového monitoru
                                  // znaky, které nejsou celá čísla, nebo znaménka jsou přeskočeny
```

```
  }
  return mk;
}

// slouží pro remove kartu, kontrola zadaných znaků
// remove_k zadá uživatel v sériovém monitoru
int remove_k(void) {
  int rk = 0;

  while (rk == 0) {
    while (! Serial.available()); // získává (kontroluje) počet znaků dostupných ze sériové linky
    rk = Serial.parseInt();       // vrací první celé platné číslo ze sériového monitoru
                                  // znaky, které nejsou celá čísla, nebo znaménka jsou přeskočeny
  }
  return rk;
}

// ------------------------------------------------------------------------------------------
// -------------------------------------smyčka void loop-----------------------------------------
// ------------------------------------------------------------------------------------------
void loop()
{
  // ---------------------------------------------------------------------------
  // ----------------------------kontrola přiloženého tagu-------------------------
  // ---------------------------------------------------------------------------
```

```
 // kontrola boolean - stav_1
 if (stav_1 == true)
 {
  // prvotní nastavení režimů boolean
  stav_2 = false; // stav pro otisk
  stav_m = false; // stav pro mordý čip
  stav_z = false; // stav pro žlutý čip
  stav_k = false; // stav pro správnou kartu

  // nastevení všech LED na hodnotu LOW na začátku
  digitalWrite(cervena_LED, LOW);
  digitalWrite(zelena_tag_LED, LOW);
  digitalWrite(zelena_heslo_LED, LOW);
  digitalWrite(zelena_otisk_LED, LOW);

  //efekt zapínání
  lcd.setCursor(0, 0);              // nastavení pozice kurzoru
  lcd.print("    RFID SYSTEM");      // na LCD displeji se zobrazí nápis "RFID SYSTEM"
  delay(PRODLEVA_1);                // definovaná časová prodleva
  lcd.setCursor(0, 1);              // nastavení pozice kurzoru
  lcd.print("      VITEJTE  ");      // na LCD displeji se zobrazí nápis "VITEJTE"
  delay(PRODLEVA_2);                // definovaná časová prodleva
  lcd.setCursor(0, 2);              // nastavení pozice kurzoru
  lcd.print("    PRILOZTE TAG   ");      // na LCD displeji se zobrazí nápis "PRILOZTE
TAG"
```

```
 // slouží pro hledání tagu
if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;
}

// čtení tagu
if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;
}

//čtení z tagu
String tag = "";  // do proměnné tag se uloží ID tagu, který byl přiložen

// kontrola, zda ID tagu odpovídá předepsanému formátu
for (byte k = 0; k < mfrc522.uid.size; k++)
{
  tag.concat(String(mfrc522.uid.uidByte[k] < 0x10 ? " 0" : " "));
  tag.concat(String(mfrc522.uid.uidByte[k], HEX));
}
tag.toUpperCase();  // převádí ID tagu na velká písmena

// výpis na sériový monitor, vypíše se název tagu
// pokud je přiložen správný tag, tak se zobrazí název tagu,
// pokud je přiložen nepsrávná tag, tak se zobrazí jeho ID a nápis "NESPRÁVNÝ TAG"
 if (tag.substring(1) == (tag_1))
 { Serial.print("použitý tag:"); Serial.println(" žlutý čip"); stav_z = true; }
```

```
    if (tag.substring(1) == (tag_2))
    { Serial.print("použitý tag:"); Serial.println(" modrý čip"); stav_m = true; }
    if (tag.substring(1) == (tag_3))
    { Serial.print("použitý tag:"); Serial.println(" správná karta"); stav_k = true; }

    if (tag.substring(1) != (tag_1) && tag.substring(1) != (tag_2) && tag.substring(1) != (tag_3)&&
        tag.substring(1) != (tag_4) && tag.substring(1) != (tag_5) && tag.substring(1) != (tag_6))
    { Serial.print(tag.substring(1)); Serial.println(": NESPRÁVNÝ TAG");
    Serial.println("PŘÍSTUP ZAMÍTNUT");
    Serial.println("//--------------------------------------------------------------------//");}

//_____________________________________________________________________
_______
   //_____________________kontrola tagu, pro které je povolen přístup _____________________

//_____________________________________________________________________
_______
   if (tag.substring(1) == (tag_1)|| tag.substring(1) == (tag_2)|| tag.substring(1) == (tag_3))
// kontrola tagu od prvního znaku

   //_____________________________________________________________________
   //_____________________________________správný tag_____________________________________
   //_____________________________________________________________________
   {
    // pokud se shoduje ID tagu s databází
```

```
    lcd.clear();                    // smaže vše, co bylo na LCD displeji
    lcd.print("SPRAVNY TAG");          // na LCD displeji se zobrazí nápis "SPRAVNY TAG"
    digitalWrite(zelena_tag_LED, HIGH);  // rozsvítí se zelená LED dioda signalizující správný tag
    delay(PRODLEVA_2);              // definovaná časová prodleva
    lcd.clear();                    // smaže vše, co bylo na LCD displeji
    lcd.print("ZADEJTE HESLO:");        // na LCD displeji se zobrazí nápis "ZADEJTE HESLO"
    lcd.setCursor(0, 1);            // nastavení pozice kurzoru
    stav_1 = false;                 // změna režimu stav_1 na false
  }

  //_____________________________________________________________________
  //_____________________REMOVE karta_____________________
  //_____________________________________________________________________
  else if (tag.substring(1) == (tag_4)) // kontrola remove karty od první pozice
  {
    // výpis na sériový monitor, vypíše se název karty
    Serial.println("REMOVE karta");

    // výpis na LCD
    lcd.clear();                    // smaže vše, co bylo na LCD displeji
    lcd.setCursor(0, 0);            // nastavení pozice kurzoru
    lcd.print("REMOVE KARTA");          // na LCD displeji se zobrazí nápis "REMOVE KARTA"
    lcd.setCursor(0, 1);            // nastavení pozice kurzoru
```

```
    lcd.print("PROBIHA PROGRAM");       // na LCD displeji se zobrazí nápis "PROBIHA PROGRAM"
    lcd.setCursor(0, 2);                // nastavení pozice kurzoru
    lcd.print("ODSTRANENI OTISKU");             // na LCD displeji se zobrazí nápis "ODSTRANENI OTISKU"

    //programová část pro odstranění otisku
    Serial.println("//--------------------------------------------------------------------//");
    Serial.println("Programová část pro odstranění otisku");
    finger.getTemplateCount();              // slouží k zjištění počtu uložených otisků v paměti čtečky

    // výpis na sérivý monitor, vypíše se počet uložených otisků
    Serial.print("v paměti senzoru je uloženo:  "); Serial.print(finger.templateCount); Serial.println(" otisků prstů");

    // výpis na sérivý monitor
    Serial.println("zadejte číslo od 1 do 127, pod kterým je uložen otisk, který chcete odstranit");

    // proměnná pro odstranění otisku
    // v proměnné remove_k je hodnota, kterou zadal uživatel, tato hodnota se uloží do proměnné otisk_del
    int otisk_del = remove_k();
    if (otisk_del == 0) {     // číslo 0 není povoleno pro uložení ani odstranění otisku
     return;
    }
```

```
    // výpis na sérivý monitor
    Serial.print("bude odstraněn otisk s #:");Serial.println(otisk_del);

    // zde je podprogram pro odstranění otisku
    // podprogram pokračuje pod koncem smyčky void loop
    OdstraneniOtisku(otisk_del);
  }

  //_____________________________________________________________________
  //_____________________________MASTER karta____________________________
  //_____________________________________________________________________
  else if (tag.substring(1) == (tag_5))  // kontrola master karty od první pozice
  {
    // výpis na sériový monitor, vypíše se název karty
    Serial.println("MASTER karta");

    // výpis na LCD
    lcd.clear();                    // smaže vše, co bylo na LCD displeji
    lcd.setCursor(0, 0);            // nastavení pozice kurzoru
    lcd.print("MASTER KARTA");      // na LCD displeji se zobrazí nápis "MASTER KARTA"
    lcd.setCursor(0, 1);            // nastavení pozice kurzoru
    lcd.print("PROBIHA PROGRAM");   // na LCD displeji se zobrazí nápis "PROBIHA PROGRAM"
    lcd.setCursor(0, 2);            // nastavení pozice kurzoru
```

```
    lcd.print("PRIDANI OTISKU");          // na LCD displeji se zobrazí nápis "PRIDANI OTISKU"

    //programová část pro přidání otisku
    Serial.println("//--------------------------------------------------------------------//");
    Serial.println("Programová část pro přidání otisku");
    finger.getTemplateCount();            // slouží k zjištění počtu uložených otisků v paměti čtečky

    // výpis na sérivý monitor, vypíše se počet uložených otisků
    Serial.print("v paměti senzoru je uloženo:  "); Serial.print(finger.templateCount); Serial.println(" otisků prstů");

    // výpis na sérivý monitor
    Serial.println("zadejte číslo od 1 do 127, pod kterým se uloží otisk prstu");
    Serial.println("přístup je povolen pro otisky 1,2,3,4,5,6 ostatní je potřeba nastavit");

    // proměnná pro ukládání nového otisku
    // v proměnné master_k je hodnota, kterou zadal uživatel, tato hodnota se uloží do proměnné otisk_nv
    int otisk_nv = master_k();
    if (otisk_nv == 0) {      // číslo 0 není povoleno pro odstranění otisku
    return;
    }

    // výpis na sérivý monitor
    Serial.print("otisk bude uložen s #");Serial.println(otisk_nv);
```

```
    // zde je podprogram pro uložení otisku
    // podprogram pokračuje pod koncem smyčky void loop
    PridaniOtisku(otisk_nv);
  }

  //_____________________________________________________________________
  //_____________________________EMPTY karta_____________________________
  //_____________________________________________________________________
  else if (tag.substring(1) == (tag_6)) // kontrola empty karty od první pozice
  {
    // výpis na sériový monitor, vypíše se název karty
    Serial.println("EMPTY karta");

    // výpis na LCD
    lcd.clear();                    // smaže vše, co bylo na LCD displeji
    lcd.setCursor(0, 0);            // nastavení pozice kurzoru
    lcd.print("EMPTY KARTA");       // na LCD displeji se zobrazí nápis "EMPTY KARTA"
    lcd.setCursor(0, 1);            // nastavení pozice kurzoru
    lcd.print("PROBIHA PROGRAM");   // na LCD displeji se zobrazí nápis "PROBIHA PROGRAM"
    lcd.setCursor(0, 2);            // nastavení pozice kurzoru
    lcd.print("VYMAZANI PAMETI");   // na LCD displeji se zobrazí nápis "VYMAZANI PAMETI"

    //programová část pro vymazání paměti
```

```
    Serial.println("//--------------------------------------------------------------------//");
    Serial.println("Programová část pro vymazání paměti čtečky otisků prstů");

    // zde je podprogram pro odstranění všech otisků
    // podprogram pokračuje pod koncem smyčky void loop
    VymazaniPameti();
  }

  //_____________________________________________________________________________
  //_____________________________nesprávný tag_____________________________
  //_____________________________________________________________________________
  else
  {
    // pokud se neshoduje ID tagu s databází
    // vrací se na začátek RFIDMode == true
    lcd.clear();                    // smaže vše, co bylo na LCD displeji
    lcd.setCursor(0, 0);            // nastavení pozice kurzoru
    lcd.print("NESPRAVNY TAG");              // na LCD displeji se zobrazí nápis "NESPRAVNY TAG"
    lcd.setCursor(0, 1);            // nastavení pozice kurzoru
    lcd.print("PRISTUP ZAMITNUT");     // na LCD displeji se zobrazí nápis "PRISTUP ZAMITNUT"
    digitalWrite(BZUCAK, HIGH);       // rozezní se bzučák
    digitalWrite(cervena_LED, HIGH);    // rozsvítí se červená LED dioda
    delay(PRODLEVA_3);              // definovaná časová prodleva
    digitalWrite(BZUCAK, LOW);        // po uplynutí času, se bzučák ztlumí
```

```
   digitalWrite(cervena_LED, LOW);       // po uplynutí času, LED dioda zhasne
   lcd.clear();                          // smaže vše, co bylo na LCD displeji
  }
}
  // ---------------------------------------------------------------------------
  // -------------------------kontrola zadaného hesla-----------------------------
  // ---------------------------------------------------------------------------
  // kontrola boolean - stav_1
   if (stav_1 == false)
   {
  heslo_stisk = tl_stisk.getKey();   // uložení znaku hesla
  if (heslo_stisk)                   // heslo_stisk - vyvolá se při každém stisku klávesy
  {
   heslo[h++] = heslo_stisk;         // uložení hesla do proměnné
   lcd.print("x");                   // zástupný znak, který se zobrazí na displeji po zadání hesla
  }
  if (h == 4)                        // kontrola, zda zadané heslo obsahuje čtyři znaky
  {
   delay(PRODLEVA_1);
   // kontrola zda se zadané heslo shoduje s heslem uloženým v systému a obsahuje 4 znaky
   // heslo: toto heslo zadá uživatel
   // user_heslo: (1030), je přiřazené k modrému čipu
   // guest_heslo: (1223), je přiřazené k žlutému čipu
   // master_heslo: (AD#0), je přiřazené k správné kartě
   if ((stav_z == true && (!(strncmp(heslo, guest_heslo, 4))))
   || (stav_m == true && (!(strncmp(heslo, user_heslo, 4))))
```

```
    || (stav_k == true && (!(strncmp(heslo, master_heslo, 4)))))


    //_____________________________________________________________________________
    //_____________________________správné heslo_____________________________
    //_____________________________________________________________________________
    {
     // pokud se heslo shoduje s heslem uloženým v programu
     lcd.clear();                        // smaže vše, co bylo na LCD displeji
     lcd.print("SPRAVNE HESLO");                  // na LCD displeji se zobrazí nápis "SPRAVNE HESLO"
     digitalWrite(zelena_heslo_LED, HIGH);     // rozsvítí se zelená LED dioda signalizující správné heslo
     delay(PRODLEVA_2);                    // definovaná časová prodleva
     lcd.clear();                        // smaže vše, co bylo na LCD displeji

    // výpis na sériový monitor, vypíše se název hesla
     if (!(strncmp(heslo, user_heslo, 4)))      // použité heslo: user
     {
      Serial.println("použité heslo: user");   // výpis na sériový monitor
     }

     if (!(strncmp(heslo, guest_heslo, 4)))     // použité heslo: guest
     {
      Serial.println("použité heslo: guest");  // výpis na sériový monitor
     }
```

```
 if (!(strncmp(heslo, master_heslo, 4)))   // použité heslo: master
 {
  Serial.println("použité heslo: master"); // výpis na sériový monitor
 }

 lcd.print("ZADEJTE OTISK:");            // na LCD displeji se zobrazí nápis "ZADEJTE OTISK"
 h = 0;                                 // vynulování proměnné pro zjištění počtu znaků hesla
 stav_2 = true;                         // změna režimu stav_2 na true
}

//_____________________________________________________________________________
//_____________________________nesprávné heslo_____________________________
//_____________________________________________________________________________
else
{
 lcd.clear();                   // smaže vše, co bylo na LCD displeji
 lcd.setCursor(0, 0);           // nastavení pozice kurzoru
 lcd.print("SPATNE HESLO");          // na LCD displeji se zobrazí nápis "SPATNE HESLO"
 lcd.setCursor(0, 1);           // nastavení pozice kurzoru
 lcd.print("PRISTUP ZAMITNUT");     // na LCD displeji se zobrazí nápis "PRISTUP ZAMITNUT"

 // výpis na sériový monitor, vypíše se zadané nesprávné heslo
 Serial.print("použito nesprávné heslo: "); Serial.println(heslo);
 Serial.println("PŘÍSTUP ZAMÍTNUT");
```

```
    Serial.println("//--------------------------------------------------------------------//");

    digitalWrite(BZUCAK, HIGH);         // rozezní se bzučák
    digitalWrite(cervena_LED, HIGH);    // rozsvítí se červená LED dioda
    delay(PRODLEVA_3);                  // definovaná časová prodleva
    digitalWrite(BZUCAK, LOW);          // po uplynutí času, se bzučák ztlumí
    digitalWrite(cervena_LED, LOW);     // po uplynutí času, LED dioda zhasne
    lcd.clear();                        // smaže vše, co bylo na LCD displeji
    h = 0;                              // vynulování proměnné pro zjištění počtu znaků hesla
    stav_1 = true;                      // změna režimu stav_1 na true
   }
  }
  }

  // ------------------------------------------------------------------------
  // ---------------------------kontrola zadaného otisku----------------------------
  // ------------------------------------------------------------------------
  // kontrola boolean - stav_2
  if (stav_2 == true)
  {
  //__________snímač vyfotí prst přiložený na čtečku____________________
  //-------------------------------------------------------------
 uint8_t k = -1;
 k = finger.getImage();
 if (k == FINGERPRINT_OK)                    // proběhlo v pořádku
 {
```

```
  // pokud proběhlo v pořádku, tak pokračuje dál
}
else
{
 return k;                              // pokud něco selhalo, vrací k
}

 //_________senzor převede snímek na šablonu funkce_____________________
 //-------------------------------------------------------------
 k = finger.image2Tz();
 if (k == FINGERPRINT_OK)                    // proběhlo v pořádku
 {
   // pokud proběhlo v pořádku, tak pokračuje dál
 }
 else
 {
  return k;                             // pokud něco selhalo, vrací k
 }

 //_________senzor prohledá aktuální funkce otisků,_____________________
 //aby odpovídaly uloženým šablonám, shodný otisk je uložen v fingerID
 //-------------------------------------------------------------------
k = finger.fingerFastSearch();
if (k == FINGERPRINT_OK)                    // proběhlo v pořádku
{
  // pokud proběhlo v pořádku, tak pokračuje dál
```

```
  }

  //_____________________________________________________________________
  //_____________________________neznámý otisk_____________________________
  //_____________________________________________________________________
  if (k == FINGERPRINT_NOTFOUND)
  {
   // otisk nesouhlasí s uloženou šablonou
   lcd.clear();                    // smaže vše, co bylo na LCD displeji
   lcd.setCursor(0, 0);            // nastavení pozice kurzoru
   lcd.print("NEZNAMY OTISK");     // na LCD displeji se zobrazí nápis "NEZNAMY OTISK"
   lcd.setCursor(0, 1);            // nastavení pozice kurzoru
   lcd.print("PRISTUP ZAMITNUT");  // na LCD displeji se zobrazí nápis "PRISTUP ZAMITNUT"

   // výpis na sériový monitor
   Serial.println("použit neznámý otisk");
   Serial.println("PŘÍSTUP ZAMÍTNUT");
   Serial.println("//--------------------------------------------------------------------//");

   digitalWrite(BZUCAK, HIGH);     // rozezní se bzučák
   digitalWrite(cervena_LED, HIGH);  // rozsvítí se červená LED dioda
   delay(PRODLEVA_3);              // definovaná časová prodleva
   digitalWrite(BZUCAK, LOW);      // po uplynutí času, se bzučák ztlumí
   digitalWrite(cervena_LED, LOW);  // po uplynutí času, LED dioda zhasne
```

```
   lcd.clear();                    // smaže vše, co bylo na LCD displeji
   stav_1 = true;                  // změna režimu stav_1 na true
   }


//_____________________________________________________________________
_________
  //_____________________kontrola otisků, pro které je povolen přístup_____________________

//_____________________________________________________________________
_________
  // žlutý čip - OTISK_1, OTISK_2
  // modrý čip - OTISK_3, OTISK_4
  // správná karta - OTISK_5, OTISK_6
  else if ((stav_z == true && (finger.fingerID == OTISK_1 || finger.fingerID == OTISK_2))
      || (stav_m == true && (finger.fingerID == OTISK_3 || finger.fingerID == OTISK_4))
      || (stav_k == true && (finger.fingerID == OTISK_5 || finger.fingerID == OTISK_6)))

   //_____________________________________________________________________
   //_____________________________správný otisk_____________________________
   //_____________________________________________________________________
   {
   lcd.clear();                    // smaže vše, co bylo na LCD displeji
   lcd.setCursor(0, 0);            // nastavení pozice kurzoru
   lcd.print("SPRAVNY OTISK");     // na LCD displeji se zobrazí nápis "SPRAVNY
OTISK"
   lcd.setCursor(0, 1);            // nastavení pozice kurzoru
```

```
  lcd.print("PRISTUP POVOLEN");          // na LCD displeji se zobrazí nápis "PRISTUP POVOLEN"

  // výpis na sériový monitor, vypíše se zadaný # otisku
  Serial.print("použitý otisk: #");Serial.println(finger.fingerID);
  Serial.println("PŘÍSTUP POVOLEN");
  Serial.println("//--------------------------------------------------------------------//");

  sg90.write(0);                      // dveře otevřeny
  digitalWrite(zelena_otisk_LED, HIGH);    // rozsvítí se zelená LED dioda signalizující správný otisk
  delay(PRODLEVA_4);                  // definovaná časová prodleva
  sg90.write(90);                     // dveře uzavřeny
  stav_1 = true;                      // změna režimu stav_1 na true
 }

 //____________________________________________________________________________
 //_____________________________nesprávný otisk_____________________________
 //____________________________________________________________________________
 else
 {
  // otisk nesouhlasí s uloženou šablonou
  lcd.clear();                        // smaže vše, co bylo na LCD displeji
  lcd.setCursor(0, 0);                // nastavení pozice kurzoru
  lcd.print("SPATNY OTISK");          // na LCD displeji se zobrazí nápis "SPATNY OTISK"
  lcd.setCursor(0, 1);                // nastavení pozice kurzoru
```

```
    lcd.print("PRISTUP ZAMITNUT");          // na LCD displeji se zobrazí nápis "PRISTUP
ZAMITNUT"

    // výpis na sériový monitor, vypíše se zadaný # nesprávného otisku
    Serial.print("použit nesprávný otisk: #");Serial.println(finger.fingerID);
    Serial.println("PŘÍSTUP ZAMÍTNUT");
    Serial.println("//--------------------------------------------------------------------//");

    digitalWrite(BZUCAK, HIGH);            // rozezní se bzučák
    digitalWrite(cervena_LED, HIGH);       // rozsvítí se červená LED dioda
    delay(PRODLEVA_3);                     // definovaná časová prodleva
    digitalWrite(BZUCAK, LOW);             // po uplynutí času, se bzučák ztlumí
    digitalWrite(cervena_LED, LOW);        // po uplynutí času, LED dioda zhasne
    lcd.clear();                           // smaže vše, co bylo na LCD displeji
    stav_1 = true;                         // změna režimu stav_1 na true
  }
  }
  }

    // ---------------------------------------------------------------------------------------
    // --------------------------------podprogram pro odstranění otisku-------------------------
-
    // ---------------------------------------------------------------------------------------
    void OdstraneniOtisku(int otisk_del){
    uint8_t k = -1;
```

```
// kontrola, zda uživatel nezadal větší číslo než 127,
// pokud je zadáno větší číslo než 127, tak se program pro odstranění otisku ukončí
// pokud je zadáno menší číslo než 127, tak se pokračuje dál
if (otisk_del > 127)
{
  Serial.println("nesprávná hodnota pro odstranění otisku");   // výpis na sériový monitor

  //výpis na sériový monitor - konec programu pro odstranění otisku
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("konec programu pro odstranění otisku ");
  Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
  Serial.println("//--------------------------------------------------------------------//");
  return k;
}

// ---------------------------------------------------------------
// ---------------slouží k vymazání otisku z paměti---------------
// ---------------------------------------------------------------
k = finger.deleteModel(otisk_del);

if (k == FINGERPRINT_OK)             // proběhlo v pořádku
{
// výpis na sériový monitor
Serial.println("otisk odstraněn!");

//výpis na LCD
```

```
  lcd.clear();                    // smaže vše, co bylo na LCD displeji
  lcd.setCursor(0, 0);            // nastavení pozice kurzoru
  lcd.print("OTISK ODSTRANEN");    // na LCD displeji se zobrazí nápis "OTISK ODSTRANEN"
  delay(PRODLEVA_3);              // nastavení časové prodlevy

  //výpis na sériový monitor
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("konec programu pro odstranění otisku ");
  Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
  Serial.println("//--------------------------------------------------------------------//");
  }
  else if (k == FINGERPRINT_BADLOCATION)      // neplatné umístění
  {
  Serial.println("neplatné umístění");        // výpis na sériový monitor

  //výpis na sériový monitor - konec programu pro odstranění otisku
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("konec programu pro odstranění otisku ");
  Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
  Serial.println("//--------------------------------------------------------------------//");
  return k;
  }
  else if (k == FINGERPRINT_FLASHERR)         // nelze provést zápis do paměti
  {
  Serial.println("nelze provést zápis do paměti"); // výpis na sériový monitor
```

```
  //výpis na sériový monitor - konec programu pro odstranění otisku
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("konec programu pro odstranění otisku ");
  Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
  Serial.println("//--------------------------------------------------------------------//");
  return k;
  }
  else if (k == FINGERPRINT_PACKETRECIEVEERR)      // chyba komunikace
  {
  Serial.println("chyba komunikace");              // výpis na sériový monitor

  //výpis na sériový monitor - konec programu pro odstranění otisku
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("konec programu pro odstranění otisku ");
  Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
  Serial.println("//--------------------------------------------------------------------//");
  return k;
  }
  }

// ---------------------------------------------------------------------------------------------
// ----------------------------------podprogram pro přidání otisku--------------------------------
// ---------------------------------------------------------------------------------------------
void PridaniOtisku(int otisk_nv) {
uint8_t k = -1;
```

```
// kontrola, zda uživatel nezadal větší číslo než 127,
// pokud je zadáno větší číslo než 127, tak se program pro přidání otisku ukončí
// pokud je zadáno menší číslo než 127, tak se pokračuje dál
if (otisk_nv > 127)
{
 Serial.println("nesprávná hodnota pro uložení otisku");    // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro přidání otisku
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro přidání otisku ");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;
}

//výpis na sériový monitor
Serial.print("přiložte prst, který chcete uložit #"); Serial.println(otisk_nv);

while (k != FINGERPRINT_OK)
{

// ------------------------------------------------------------
// ----------snímač vyfotí prst přiložený na čtečku----------
// ------------------------------------------------------------
k = finger.getImage();
```

```
  if (k == FINGERPRINT_OK)                          // proběhlo v pořádku
  {
    Serial.println("prosím stále držte prst na čtečce");  // výpis na sériový monitor
  }
  else if (k == FINGERPRINT_IMAGEFAIL)              // chyba zobrazování
  {
   Serial.println("chyba zobrazování");             // výpis na sériový monitor

   //výpis na sériový monitor - konec programu pro přidání otisku
   Serial.println("//--------------------------------------------------------------------//");
   Serial.println("konec programu pro přidání otisku ");
   Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
   Serial.println("//--------------------------------------------------------------------//");
   return k;
  }
}

  // ------------------------------------------------------------
  // ----------snímač převede snímek na šablonu funkce----------
  // ------------------------------------------------------------
  k = finger.image2Tz(1);
  if (k == FINGERPRINT_OK)                          // proběhlo v pořádku
  {
    Serial.println("snímek převeden");              // výpis na sériový monitor
  }
  else if (k == FINGERPRINT_IMAGEMESS)              // chaotický snímek
```

```
{
 Serial.println("chaotický snímek");                // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro přidání otisku
 Serial.println("//-----------------------------------------------------------------//");
 Serial.println("konec programu pro přidání otisku ");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//-----------------------------------------------------------------//");
 return k;
}
else if (k == FINGERPRINT_PACKETRECIEVEERR)          // chyba komunikace
{
 Serial.println("chyba komunikace");                // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro přidání otisku
 Serial.println("//-----------------------------------------------------------------//");
 Serial.println("konec programu pro přidání otisku ");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//-----------------------------------------------------------------//");
 return k;
}
else if (k == FINGERPRINT_FEATUREFAIL)               // selhání při identifikaci otisku
{
 Serial.println("selhání při identifikaci otisku");  // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro přidání otisku
```

```
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro přidání otisku ");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;
}

// časová prodleva
delay(PRODLEVA_2);

//výpis na sériový monitor
Serial.print("vytvoření pro #");  Serial.println(otisk_nv);

// ---------------------------------------------------------------
// --------------------uložení záznamu otisku----------------------
// ---------------------------------------------------------------
k = finger.storeModel(otisk_nv);
if (k == FINGERPRINT_OK)               // proběhlo v pořádku
{
  Serial.println("otisk uložen!");      //výpis na sériový monitor
  Serial.println("prst můžete sejmout");      //výpis na sériový monitor

  //výpis na LCD
  lcd.clear();                        // smaže vše, co bylo na LCD displeji
  lcd.setCursor(0, 0);                // nastavení pozice kurzoru
```

```
   lcd.print("OTISK ULOZEN");                // na LCD displeji se zobrazí nápis "OTISK ULOZEN"
   delay(PRODLEVA_3);                  // nastavení časové prodlevy

   //výpis na sériový monitor - konec programu pro přidání otisku
   Serial.println("//------------------------------------------------------------------//");
   Serial.println("konec programu pro přidání otisku ");
   Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
   Serial.println("//------------------------------------------------------------------//");
  }
  else if (k == FINGERPRINT_BADLOCATION)           // neplatné umístění
  {
   Serial.println("neplatné umístění");            // výpis na sériový monitor

   //výpis na sériový monitor - konec programu pro přidání otisku
   Serial.println("//------------------------------------------------------------------//");
   Serial.println("konec programu pro přidání otisku ");
   Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
   Serial.println("//------------------------------------------------------------------//");
   return k;
  }
  else if (k == FINGERPRINT_FLASHERR)              // nelze provést zápis do paměti
  {
   Serial.println("nelze provést zápis do paměti");   // výpis na sériový monitor

   //výpis na sériový monitor - konec programu pro přidání otisku
```

```
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro přidání otisku ");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;
}
else if (k == FINGERPRINT_PACKETRECIEVEERR)        // chyba komunikace
{
 Serial.println("chyba komunikace");               // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro přidání otisku
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro přidání otisku ");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;
 }
}

 // ---------------------------------------------------------------------------------------
 // -------------------------podprogram pro vymazání paměti čtečky otisků prstů----------------
-----
 // ---------------------------------------------------------------------------------------
 void VymazaniPameti(){
 uint8_t k = -1;
```

```
// ------------------------------------------------------------
// ---------------------slouží k vymazání paměti-----------------
// ------------------------------------------------------------
delay(PRODLEVA_3);
k = finger.emptyDatabase();
if (k == FINGERPRINT_OK)                          // proběhlo v pořádku
{
  Serial.println("všechny otisky byly odstraněny!");  // výpis na sériový monitor

  //výpis na LCD
  lcd.clear();                                // smaže vše, co bylo na LCD displeji
  lcd.setCursor(0, 0);                        // nastavení pozice kurzoru
  lcd.print("PAMET VYMAZANA");                      // na LCD displeji se zobrazí nápis "PAMET VYMAZANA"
  delay(PRODLEVA_3);                          // nastavení časové prodlevy

  //výpis na sériový monitor - konec programu pro vymazání paměti čtečky otisků prstů
  Serial.println("//--------------------------------------------------------------------//");
  Serial.println("konec programu pro vymazání paměti čtečky otisků prstů");
  Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
  Serial.println("//--------------------------------------------------------------------//");
}
else if (k == FINGERPRINT_BADLOCATION)            // neplatné umístění
{
  Serial.println("neplatné umístění");            // výpis na sériový monitor
```

```
 //výpis na sériový monitor - konec programu pro vymazání paměti čtečky otisků prstů
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro vymazání paměti čtečky otisků prstů");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;
}
else if (k == FINGERPRINT_FLASHERR)                    // nelze provést zápis do paměti
{
 Serial.println("nelze provést zápis do paměti");     // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro vymazání paměti čtečky otisků prstů
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro vymazání paměti čtečky otisků prstů");
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;

}
else if (k == FINGERPRINT_PACKETRECIEVEERR)          // chyba komunikace
{
  Serial.println("chyba komunikace");                // výpis na sériový monitor

 //výpis na sériový monitor - konec programu pro vymazání paměti čtečky otisků prstů
 Serial.println("//--------------------------------------------------------------------//");
 Serial.println("konec programu pro vymazání paměti čtečky otisků prstů");
```

```
 Serial.println("můžete pokračovat přiložením identifikačního tagu na čtečku");
 Serial.println("//--------------------------------------------------------------------//");
 return k;
}
}
```