

**SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE**

ZAVRŠNI RAD

**ELEKTRONIČKA BRAVA UPRAVLJANA
NFC TEHNOLOGIJOM**

Petar Kaselj

Split, srpanj 2021.



Preddiplomski studij: **Elektrotehnika i informacijska tehnologija**

Smjer/Usmjerenje: **Elektronika i računalno inženjerstvo**

Oznaka programa: 112

Akademска godina: 2020./2021.

Ime i prezime: **PETAR KASELJ**

Broj indeksa: 10-2018

ZADATAK ZAVRŠNOG RADA

Naslov: **ELEKTRONIČKA BRAVA UPRAVLJANA NFC TEHNOLOGIJOM**

Zadatak: Projektirati sustav za nadzor i upravljanje pristupom ulaznim vratima. Za identifikaciju korisnika koristiti NFC čitač i brojčanu tipkovnicu. Izraditi programsku podršku na platformi po izboru (RPi, Arduino...) koja će omogućavati: definiranje korisnika i registriranje NFC tagova, definiranje grupe korisnika, autentifikaciju korisnika, upravljanje električkom bravom, detektiranje statusa vrata, provjeru zapisa otključavanja električke brave, te definiranje i upravljanje alarmima.

Prijava rada: 03.03.2021.

Rok za predaju rada: 12.07.2021.

Rad predan: 09.07.2021.

Datum obrane: 22.07.2021.

Mentor:

doc. dr. sc. Duje Čoko

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom "Elektronička brava upravljana NFC tehnologijom" pod mentorstvom prof. dr. sc. Duje Čoke pisao samostalno, primjenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrasirajući naveo u završnom radu citirao sam i povezao s korištenim bibliografskim jedinicama.

Student



Petar Kaselj

SADRŽAJ

1	UVOD.....	1
2	SKLOPOVLJE	3
2.1	Raspberry Pi	3
2.2	Ulagani uređaji	5
2.2.1	PN532 RFID/NFC čitač i kartice.....	5
2.2.2	Numerička tipkovnica.....	6
2.3	Izlazni uređaji	7
2.3.1	LCD zaslon	7
2.3.2	Zujalica	8
2.4	Relej.....	9
2.5	Pretvarač naponskih razina.....	10
3	SOFTVER	13
3.1	Arhitektura sustava	13
3.2	Sloj <i>Startup</i> procesa	14
3.3	Radni sloj	17
3.3.1	<i>HardwareDaemon</i>	17
3.3.2	<i>MainApp</i>	20
3.3.3	<i>DatabaseGateway</i>	22
3.4	<i>Driver</i> sloj	24
3.4.1	<i>Mailbox</i>	24
3.4.2	Ostali moduli	25
4	ZAKLJUČAK.....	27
	LITERATURA	28
	POPIS OZNAKA I KRATICA	30
	SAŽETAK	32
	SUMMARY	33

1 UVOD

Kontrola pristupa važan je dio svakodnevnog života, pogotovo u današnje doba rastuće globalne svijesti o ugroženosti privatnosti prosječnog potrošača. Naglim razvojem tehnologije posljednjih godina u zapadnim zemljama dolazi do razvoja tržišta potrošačke elektronike koje se sve više i više okreće prema olakšavanju svakodnevnog života – komforu.

Tako se posljednjih godina na tržištu pojavljuju *Smart Home* uređaji i sustavi, osobna vozila dolaze s mobilnim aplikacijama koje omogućavaju određenu daljinsku kontrolu vozila, dok u sferama amaterske elektronike (*Arduino*) na popularnosti dobivaju IoT projekti koji postaju dostupni sve široj populaciji.

Povećanje komfora često dolazi pod cijenu manje sigurnosti što zbog manje kvalitetnih, ali jeftinijih uređaja proizvedenih u zemljama s jeftinijom radnom snagom i blažim sustavom oporezivanja (da bi se maksimizirao profit), a što zbog sve većeg stupnja tehničkog znanja potrebnog za rukovanje tim uređajima koji postaju sve dostupniji prosječnim potrošačima. Osnovni korak u osiguravanju ovakvih sustava leži u **fizičkom** ograničenju pristupa kritičnim dijelovima sustava.

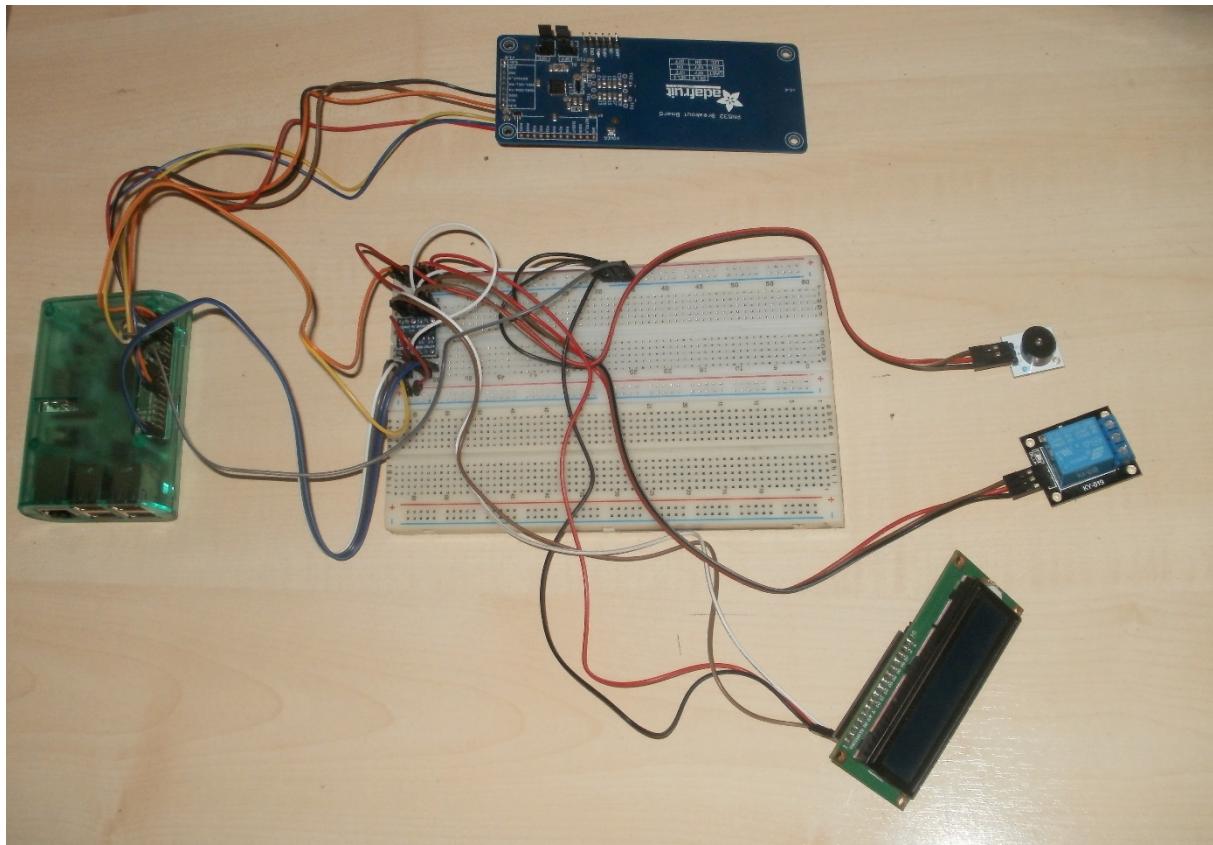
Tradicionalne metode kontrole pristupa, kao što su ključ i brava, ne mogu više pratiti dinamiku svakodnevnog života pogotovo u industrijskim primjenama gdje je mogućnost ograničavanja i praćenja pristupa određenim resursima od kritične važnosti. Jedno od mogućih rješenja obrađeno je u ovom radu.

Obrađeno rješenje koristi RFID/NFC tehnologiju da bi kontroliralo (autoriziralo) pristup prostoriji s električnom bravom koristeći RFID kartice. Osim RFID kartica postoji i numerička tipkovnica kojom se može ne samo autorizirati preko zaporke, već i obavljati određene operacije kao što su dodavanje, brisanje i izmjena podataka (npr. RFID kartice, zaporke, osobe) i određenih parametara. Svi podaci pohranjeni su u lokalnoj SQLite bazi podataka.

Osim autorizacije, sustav evidentira sve pokušaje autorizacije i/ili izvođenja operacije. Postoje dva načina rada elektroničke brave (*Fail Safe* i *Fail Secure*) načini rada koji se postavljaju u konfiguracijskoj datoteci. Za audio-vizualnu signalizaciju, sustav koristi aktivnu zujalicu (engl. *Active buzzer*) i 16x2 LCD zaslon.

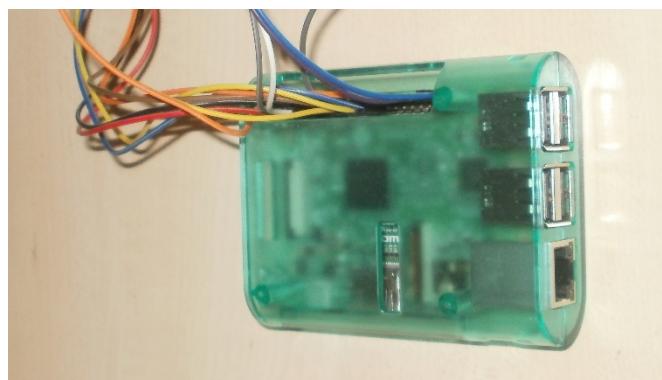
Sustav je projektiran da bude pouzdan i modularan tako da se minimiziraju troškovi održavanja i buduće nadogradnje, te da se kao takav može koristiti u stvarnim situacijama.

2 SKLOPOVLAJE



Slika 2.1 Kompletno sklopovalo rješenje

2.1 Raspberry Pi



Slika 2.2 Raspberry Pi 3B

Kao platforma za razvoj korišten je ugrađeni sustav *Raspberry Pi 3B* sa službenim *Raspbian Lite* operacijskim sustavom, jedan od varijanti 32 bitnog *Debian Linux-a*, instaliranim na SD kartici. Same dimenzije *Raspberry Pi*-ja iznose 85 mm x 56 mm x 17 mm.

Napaja se preko posebnog AC adaptera koji je na *Raspberry* spojen preko *MicroUSB* utora. Napajanje mora davati konstantnu naponsku razinu od 5 V te mora moći izdržati do 2,5 A što nam daje maksimalnu snagu od 12,5 W koja se može disipirati na *Raspberry*-ju

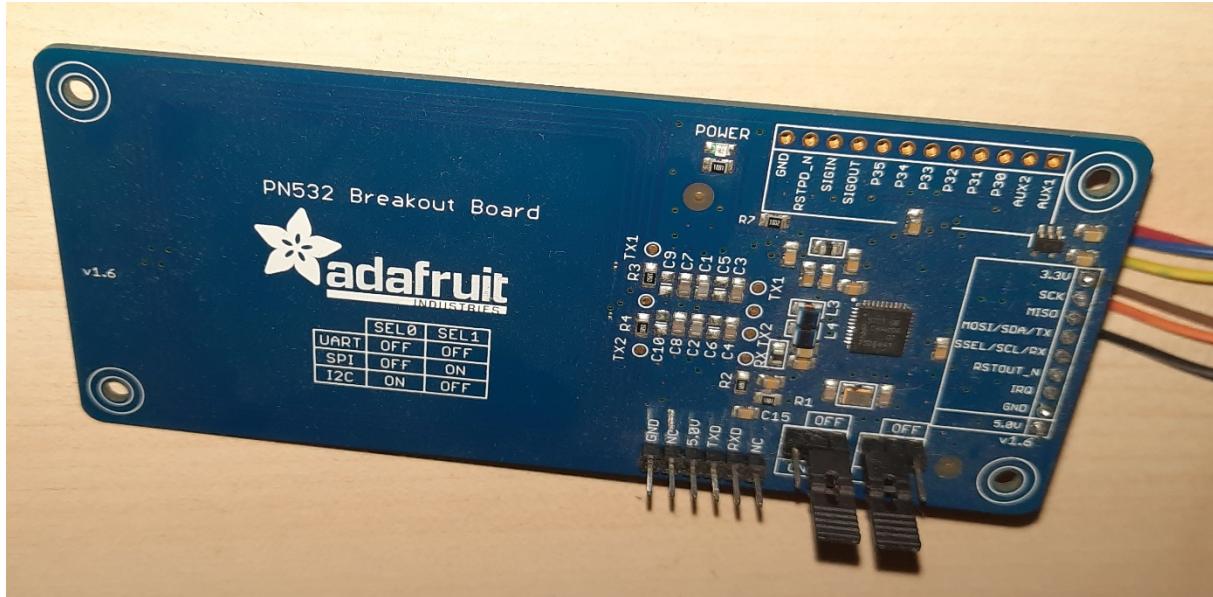
Raspberry je dizajniran oko *SoC* čipa Broadcom BCM2837 [1] koji posjeduje četverojezgreni procesor baziran na 64 bitnoj *ARM Cortex A-53* mikroarhitekturi, s brzinom takta od 1.2 GHz, koja implementira *ARMv8* skup naredbi. Korišteni model isto tako posjeduje 1 GB radne memorije.

Na mrežu se povezuje preko ugrađenog *Wi-Fi* (802.11 b/g/n 2,4 GHz) BCM43438 modula ili preko 100 Base *Ethernet* utora. *Raspberry*-ju je moguće pristupiti terminalom preko SSH protokola koristeći jedan od specijaliziranih računalnih programa npr. PuTTY.

Za pristup perifernim uređajima *Raspberry* posjeduje 4 USB 2.0 utora te 40 digitalnih ulaza/izlaza opće namjene (GPIO) koji funkcioniraju na 3,3 V CMOS logičkim razinama, odnosno nisu tolerantni na 5 V TTL logičke razine što je jedan od bitnijih faktora uzet u obzir prilikom projektiranja sustava. Ostali utori nisu korišteni te će samo biti spomenuti: HDMI utor, četveropolni stereo audio i kompozitni video utor, CSI utor za službenu *Raspberry Pi* kameru te DSI utor za službeni *Raspberry Pi* zaslon na dodir.

2.2 Ulazni uređaji

2.2.1 PN532 RFID/NFC čitač i kartice



Slika 2.3 PN532 RFID/NFC čitač

Za očitavanje RFID kartica korišten je PN532 [2] RFID/NFC modul proizvođača *Adafruit*. Same dimenziije čitača iznose 51 mm x 117,7 mm x 1,1 mm. Čitač se napaja s naponskom razinom od 3,3 V iako je moguće i napajanje s naponskom razinom od 5 V. Računalo s čitačem komunicira preko SPI komunikacijskog protokola koji koristi četiri žice (MISO, MOSI, SCK, CS). Čitač se povezuje direktno na *Raspberry* SPI GPIO pinove zbog toga što čitač i *Raspberry* GPIO funkcioniraju na 3,3 V CMOS logičkim razinama. Osim SPI moguće je s čitačem komunicirati koriteći I2C i UART komunikacijske protokole. Programska podrška za čitač razvijena je koristeći biblioteku *libnfc* koja sadrži C programsko sučelje za upravljanje RFID/NFC uređajima uključujući PN532.

2.2.2 Numerička tipkovnica



Slika 2.4 Delock numerička tipkovnica

Osim RFID/NFC čitača, korištena je i numerička tipkovnica za upravljanje sustavom. Korištena je USB numerička tipkovnica tvrtke *Delock* [3]. Dimenzije tipkovnice iznose 124 mm x 81 mm x 21 mm. Numerička tipkovnica fleksibilniji je ulazni uređaj nego RFID/NFC čitač s obzirom da tipkovnica omogućava ne samo autorizaciju, što je zapravo jedina funkcija RFID/NFC čitača, već i određenu kontrolu nad sustavom kao što je dodavanje/brisanje/izmjena podataka i parametara sustava npr. registriranje nove RFID kartice moguće je unošenjem posebne naredbe ("/1111") na numeričku tipkovnicu. Jedina prednost RFID/NFC čitača je jednostavnost i brzina upotrebe kod autorizacije, što je i najčešće korištena funkcija.

2.3 Izlazni uređaji

2.3.1 LCD zaslon

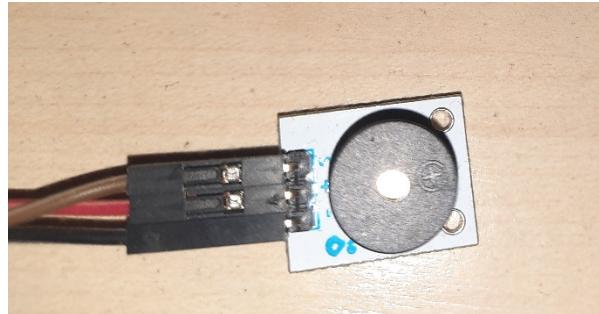


Slika 2.5 16x2 LCD zaslon sa PCF8574 na stražnjoj strani

Da bi korisnik dobio povratnu informaciju od sustava, jedan od korištenih izlaznih uređaja je 16x2 LCD zaslon [4]. Dimenzije zaslona iznose 82 mm x 35 mm x 18 mm. Zaslon je plave boje s pozadinskom osvjetljenjem i bijelim znakovima te ima 16x2 konfiguraciju odnosno 2 retka sa 16 znakova po retku.

Zaslon na sebi sadrži 8 bitni GPIO I2C ekspander PCF8574 [5] što omogućuje da se komunikacija između *Raspberry*-ja i LCD zaslona odvija preko I2C protokola što smanjuje potreban broj žica. Zaslon nije izravno spojen na GPIO priključke *Raspberry*-ja već koristi dvosmjerni pretvarač logičkih razina da bi povezao 3,3 voltne *Raspberry* GPIO izvode sa zaslonom koji funkcioniра na 5 V TTL naponskim razinama.

2.3.2 Zujalica

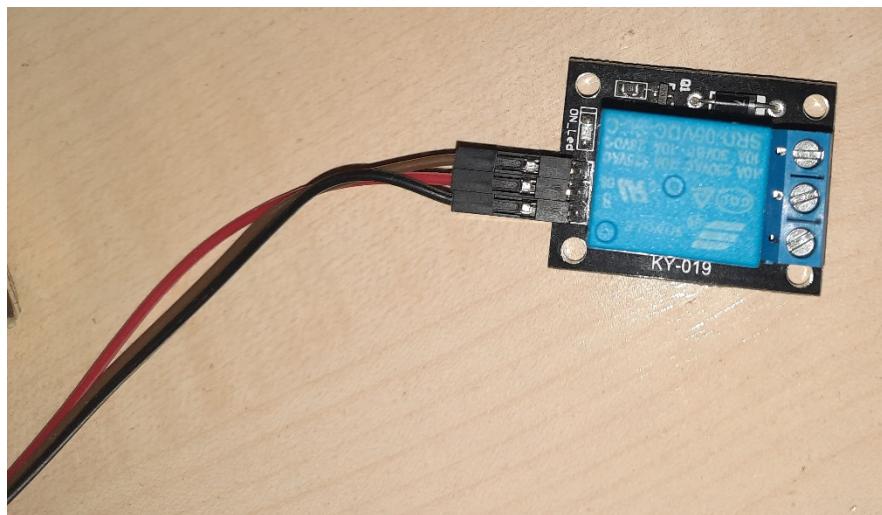


Slika 2.6 Aktivna zujalica VMA319

Za zvučnu signalizaciju sustav koristi piezoelektričnu aktivnu zujalicu tvrtke *Velleman*. Zujalica ima 3 izvoda: napajanje, signalni priključak i referentni priključak. Zujalica zahtijeva napajanje od 5 V zbog čega se signalni priključak na *Raspberry* spaja preko već spomenutog pretvarača logičkih razina.

Zujalica proizvodi ton kada se na signalni priključak dovede napon od 5 V. Zvučni signali se razlikuju po trajanju i broju prekida. Osnovni signal predstavlja jedan kratki zvučni impuls, uspješno obavljena operacija predstavljena je s dva kratka uzastopna zvučna impulsa dok je greška predstavljena s dva uzastopna zvučna impulsa od kojih je prvi kraćeg trajanja nego drugi.

2.4 Relej



Slika 2.7 KY-019 relej modul

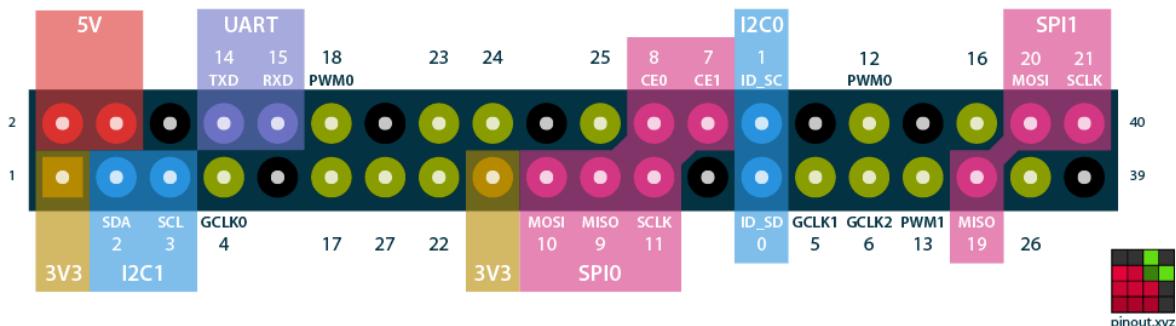
Zbog funkcionalne sličnosti, kao zamjena za električnu bravu korišten je modul KY-019 sa Songle SRD-05VDC elektromehaničkim relejem [6]. Dimenzije modula iznose 27 mm x 34 mm. Korišteni modul ima 6 izvoda od kojih se 3 izvoda (napajanje, upravljački priključak i referentni priključak) koriste za upravljanje relejem dok se ostala 3 izvoda (NO, NC i zajednički priključak) koriste za upravljanje teretom/izlazom.

Upravljački priključak se spaja na jedan *Raspberry* GPIO priključak preko pretvarača naponskih razina s obzirom da se modul napaja naponom od 5 V do 12 V. Maksimalne električne karakteristike izlaznih (NC/NO/zajednički) priključaka iznose 10A/250VAC i 10A/30VDC.

Odabir priključaka NC/NO za upravljanje vratima ovisi o odabiru *Fail Safe* ili *Fail Secure* načina rada u konfiguracijskoj datoteci, što je detaljnije opisano u kasnijim poglavljima.

2.5 Pretvarač naponskih razina

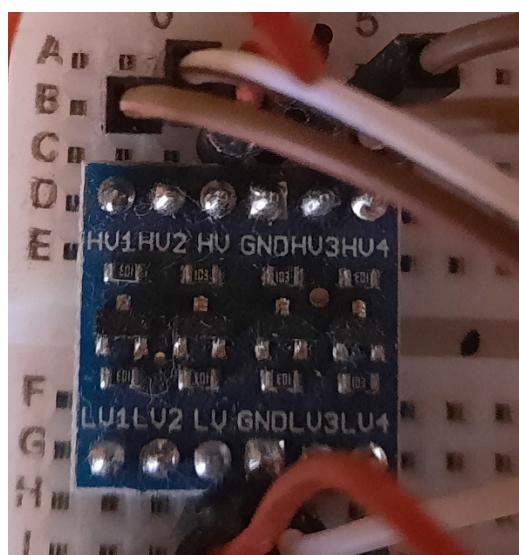
Raspberry Pi GPIO BCM numbering



Slika 2.8 Raspored izvoda na Raspberry Pi 3B modelu

S obzirom da *Raspberry Pi* GPIO priključci funkcijoniraju na 3,3 V CMOS logičkim naponskim razinama dolazi do problema kada se na *Raspberry* moraju priključiti uređaji koji funkcijoniraju na 5 V TTL logičkim razinama. Iako *Raspberry Pi* ima izvode koji mogu dati napon od 5 V, ostali GPIO pinovi nisu tolerantni na naponske razine iznad 3,3 V.

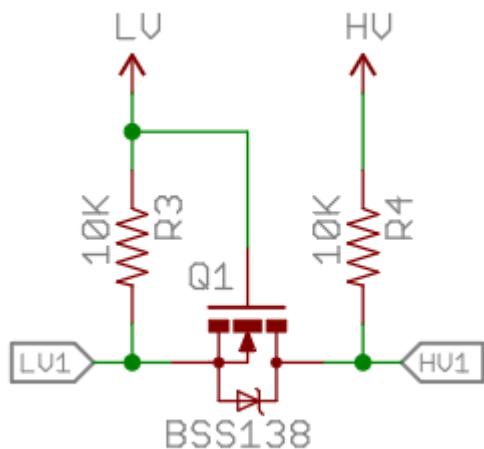
Da bi se riješio ovaj problem, korišten je četverokanalni dvosmjerni pretvarač logičkih naponskih razina s 3,3 V na 5 V hrvatske tvrtke E-radionica [7].



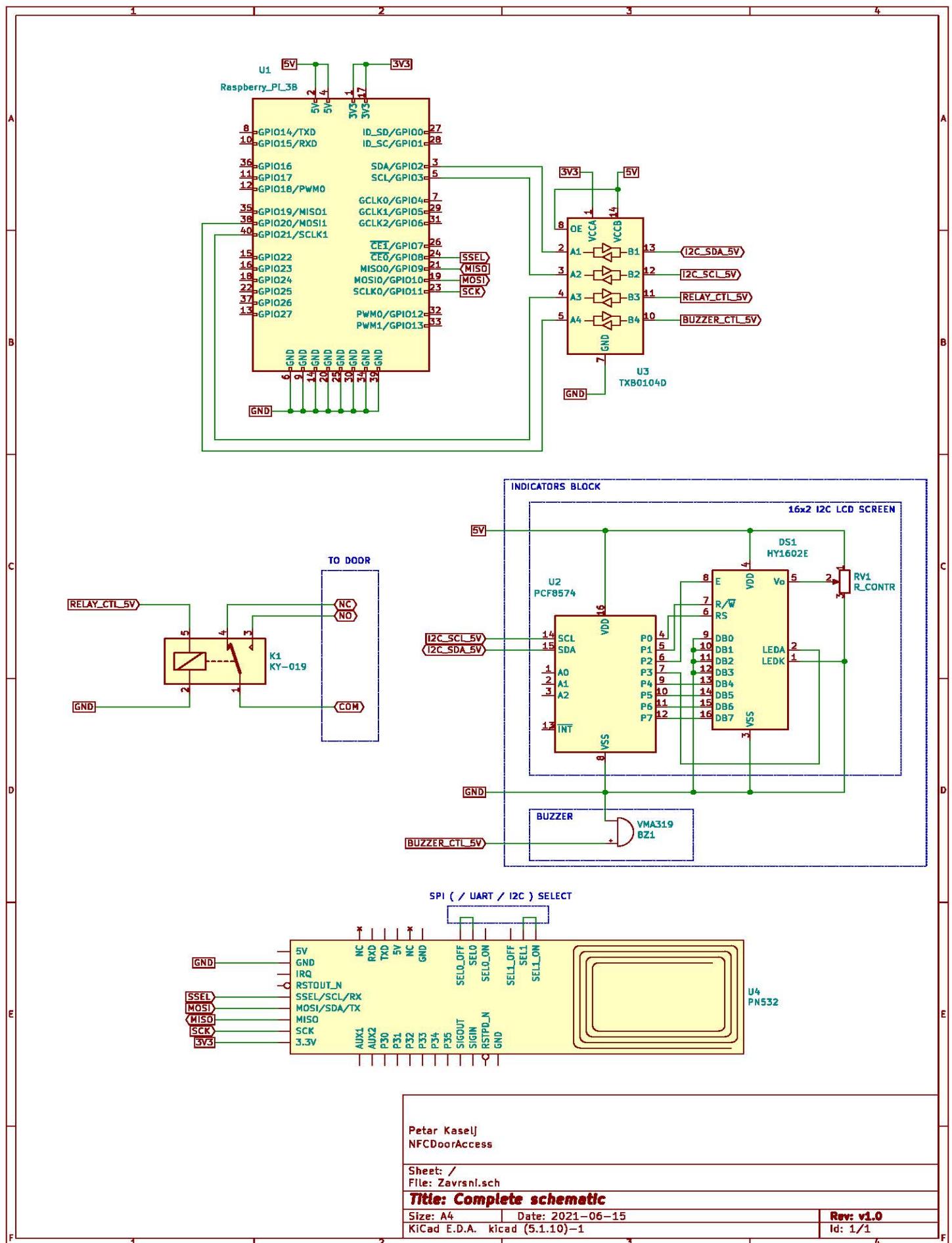
Slika 2.9 Pretvarač logičkih naponskih razina

Pretvarač ima 2 izvoda za referentne razine, koje su u ovom slučaju priključene na referentni izvod na *Raspberry*-ju (GND). Iako je pretvarač projektiran da pretvara naponske razine 3,3 V i 5 V ipak je potrebno dovesti naponsku referencu od 5 V na izvod HV odnosno 3,3 V na izvod LV.

Pretvarač ima četiri kanala sa dva izvoda po kanalu – jedan za višenaponsku (5 V) i jedan za niženaponsku (3,3 V) stranu. Uređaji koji funkcioniraju na 5 V TTL logičkim razinama spajaju se na višenaponsku stranu kanala dok se 3,3 V CMOS *Raspberry* GPIO priključci spajaju na niženaponsku stranu kanala. Pretvarač je dvosmjeran što omogućuje pretvaranje obosmjernih komunikacijskih protokola kao što je I2C protokol korišten za povezivanje na LCD modul.



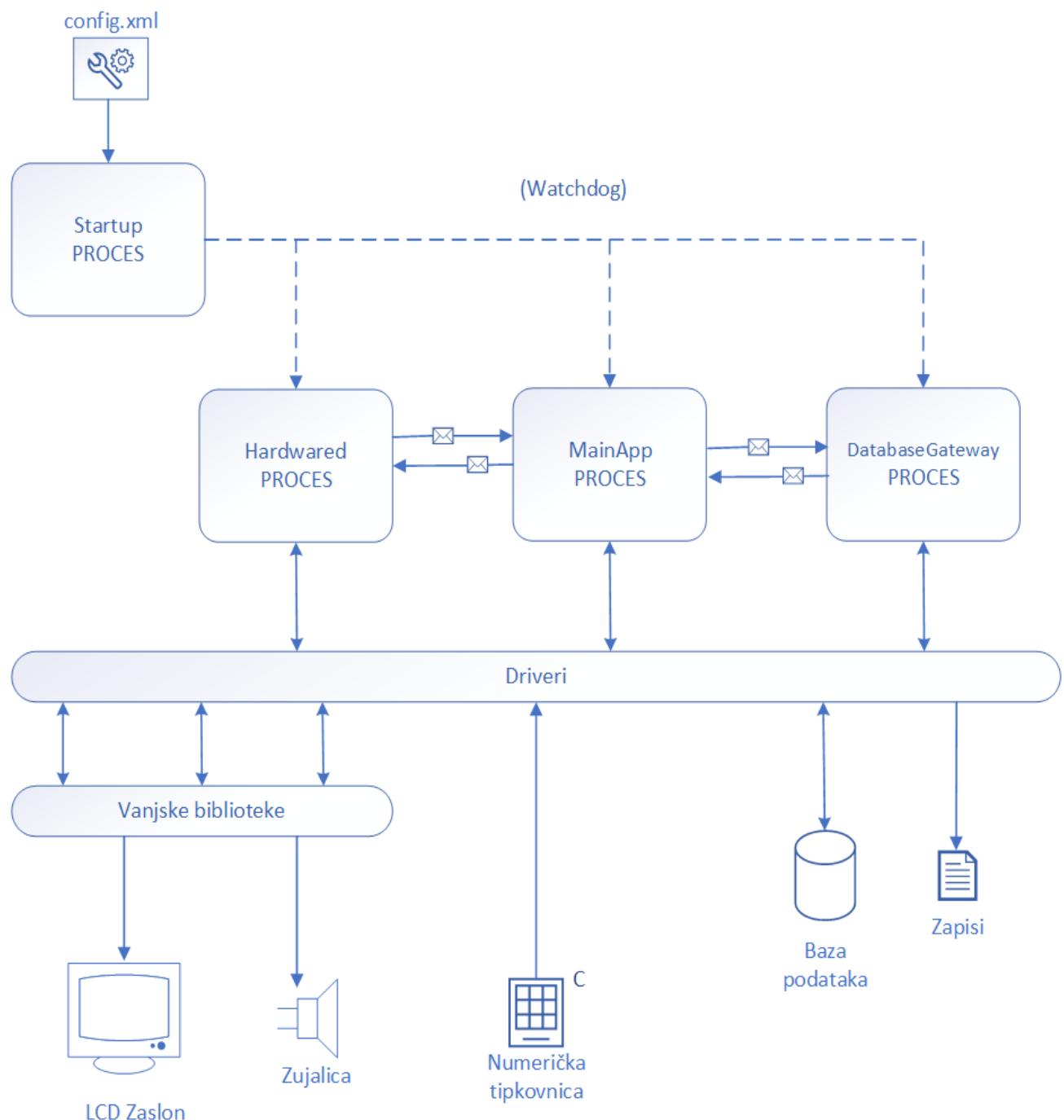
Slika 2.10 Shematski prikaz jednog kanala pretvarača logičkih razina [8]



Slika 2.11 Električna shema cjelokupnog sklopa

3 SOFTVER

3.1 Arhitektura sustava



Slika 3.1 Shematski prikaz arhitekture programskog rješenja

Na slici "Slika 3.1 Shematski prikaz arhitekture programskog rješenja" može se vidjeti pojednostavljeni blokovski prikaz arhitekture programskog rješenja.

Arhitektura se temelji na troslojnom hijerarhijskom sustavu koji se sastoji od sljedećih slojeva, poredanih od hijerarhijski najvišeg do najnižeg:

- Sloj *Startup* procesa – zadužen za pokretanje i nadziranje sustava
- Sloj radničkih procesa (radni sloj) – implementira funkcionalnost sustava
- *Driver* sloj – služe kao sučelje osnovnih funkcionalnosti ostatku sustava

Svaki sloj se oslanja na sebi podređeni sloj. Osim navedenih slojeva, *Driver* sloj se nerijetko oslanja odnosno koristi funkcionalnosti vanjskih biblioteka kao što su *libnfc*, *Qt*, *pigpio* te *sqlite3* C API.

Sama programska podrška razvijena je u C++ programskom jeziku s određenim dijelovima koji su razvijeni koristeći Qt C++ razvojni okvir. Sustav je projektiran za korištenje na *Raspberry Pi* 3B platformi te je za automatizaciju izgradnje (engl. *Build automation*) korišten jezik *CMake*. Sustav se pokreće kao *daemon* odnosno servis na *Linux* operacijskom sustavu.

U sljedećim poglavljima bit će opisani pojedini slojevi.

3.2 Sloj *Startup* procesa

Sloj *Startup* procesa sastoji se od *Startup* procesa koji je zadužen za pokretanje i nadzor nad ostalim procesima to jest procesima radnog sloja. Osim *Startup* procesa, u navedeni sloj spada i *GlobalProperties* biblioteka, razvijena pomoću Qt C++ razvojnog okvira, koja implementira funkcionalnosti za čitanje datoteke *config.xml* te njeno parsiranje u strukturu *Properties* koja predstavlja globalne postavke zajedničke cijelom sustavu.

```

<?xml version="1.0" encoding="utf-8"?>
<Settings name="NFCDoorAccess">
    <!-- Omitted -->
    <Keypad>
        <PipeName>pipe_keypad</PipeName>
        <IstreamPath>/dev/input/event0</IstreamPath>
        <ReadTimeout_ms>10</ReadTimeout_ms>
    </Keypad>
    <Mailbox queue_size="10" msg_size="200" default_timeout_ms="1000">
        <DatabaseGatewayMailbox>
            <Name>mailbox.dbgw</Name>
            <Timeout_ms>1000</Timeout_ms>
        </DatabaseGatewayMailbox>
    </Mailbox>
    <Startup>
        <Hardwared>
            <Path>HardwareDaemon.ln</Path>
        </Hardwared>
        <MainApp>
            <Path>MainApplication.ln</Path>
        </MainApp>
        <DatabaseGateway>
            <Path>DatabaseGateway.ln</Path>
        </DatabaseGateway>
    </Startup>
    <Database>
        <Path>hidden</Path>
        <Tables>
            <!-- Database structure definition (columns) - hidden -->
        </Tables>
    </Database>
    <Indicators>
        <Buzzer>
            <!-- BCM PIN -->
            <Pin>20</Pin>
            <PingDuration_ms>10</PingDuration_ms>
        </Buzzer>
        <Door>
            <!-- Uncomment FailSafe or FailSecure to enable -->
            <FailSecure />
            <!-- <FailSafe /> -->
            <Pin>21</Pin>
            <OpenTime_ms>3000</OpenTime_ms>
        </Door>
        <LCD>
            <I2C>
                <Bus>1</Bus>
                <!-- 0x27 == 39 -->
                <Address>39</Address>
            </I2C>
            <IdleMessage>EMOVIS</IdleMessage>
            <MessageLingerTime_ms>1500</MessageLingerTime_ms>
            <InterCommandWaitTime_us>500</InterCommandWaitTime_us>
        </LCD>
    </Indicators>
</Settings>

```

Kod 3.1 Primjer strukture config.xml datoteke

```

struct Properties
{
    // ----- Keypad
    unsigned int KEYPAD_BUFFER_SIZE;
    unsigned int KEYPAD_READ_TIMEOUT_MS;
    // ----- RFID
    unsigned int RFID_BUFFER_SIZE;
    unsigned int RFID_READ_TIMEOUT_MS;
    unsigned int RFID_SAME_CARD_TIMEOUT_MS;
    // ----- Database
    std::string DB_PATH;
    // ----- Mailbox
    int QUEUE_SIZE;
    int MAX_MSG_SIZE;
    std::string MAIN_MB_NAME;
    std::string DBGW_MB_NAME;
    unsigned int DATABASE_MB_TIMEOUT;
    std::string DATABASE_LOG_THREAD_MAILBOX_NAME;
    std::string HARDWARED_MB_NAME;
    // ----- Kernel
    // ----- Watchdog
    std::string WATCHDOG_SERVER_NAME;
    std::string HARDWARED_WATCHDOG_NAME;
    std::string MAIN_WATCHDOG_NAME;
    std::string DATABASE_WATCHDOG_NAME;
    unsigned int WATCHDOG_SERVER_PERIOD_MS;
    unsigned int HARDWARED_WD_TIMEOUT_MS;
    unsigned int HARDWARED_WD_TTL;
    unsigned int MAIN_WD_TIMEOUT_MS;
    unsigned int MAIN_WD_TTL;
    unsigned int DB_WD_TIMEOUT_MS;
    unsigned int DB_WD_TTL;
    // ----- Startup
    std::string HARDWARED_EXECUTABLE;
    std::string MAIN_APP_EXECUTABLE;
    std::string DBGW_EXECUTABLE;

    // ----- Keypad
    std::string KEYPAD_PIPE_NAME;
    std::string KEYPAD_ISTREAM_PATH;

    // ----- Tables

    // ----- Indicators
    std::string INDICATORS_MAILBOX_NAME;
    unsigned int INDICATORS_MB_TIMEOUT_MS;
    unsigned int BUZZER_BCM_PIN;
    DoorFailSystem DOOR_FAIL_SYSTEM;
    unsigned int DOOR_BCM_PIN;

    // std::string LOG_FILE_SUFFIX;

    // ----- General
};

```

Kod 3.2 Primjer strukture Properties

Ostatak sustava ima pristup globalnim postavkama preko statičke funkcije klase *GlobalProperties* sa sljedećom definicijom:

```
static Properties GlobalProperties::Get();
```

implementiranom kao *singleton* funkcija, da bi se izbjegla nepotrebna čitanja *config.xml* datoteke pri svakom pozivu funkcije *Get()*.

3.3 Radni sloj

Radni sloj, odnosno sloj radničkih procesa, sastoji se od tri procesa:

- *HardwareDaemon* - proces zadužen za upravljanje ulazno/izlaznim jedinicama
- *MainApp* - proces zadužen za upravljanjem tokom stanja sustava (glavni automat)
- *DatabaseGateway* - proces zadužen za pristup bazi podataka

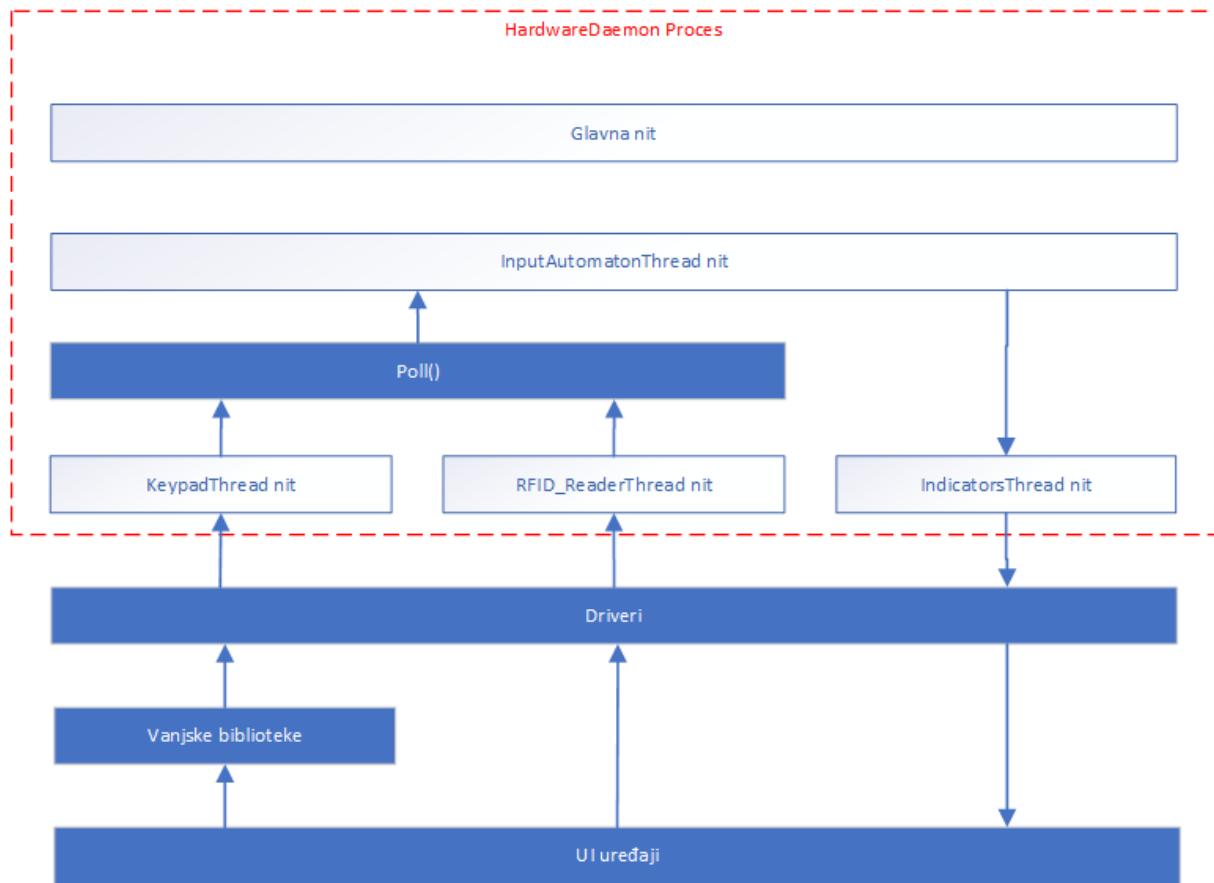
Procese pokreće glavni *Startup* proces, te ih nadzire putem softverskog *Watchdog* modula.

Procesi međusobno komuniciraju prenošenjem poruka koristeći sustav za razmjenu poruka temeljen na POSIX *message queue*-ovima.

3.3.1 *HardwareDaemon*

Proces zadužen za upravljanje ulazno/izlaznim jedinicama kao što su numerička tipkovnica, RFID/NFC čitač, LCD zaslon itd. pri čemu intenzivno koristi biblioteke sebi podređenog sloja, *Driver* sloja, koje apstrahiraju upravljanje uređajima preko API-ja.

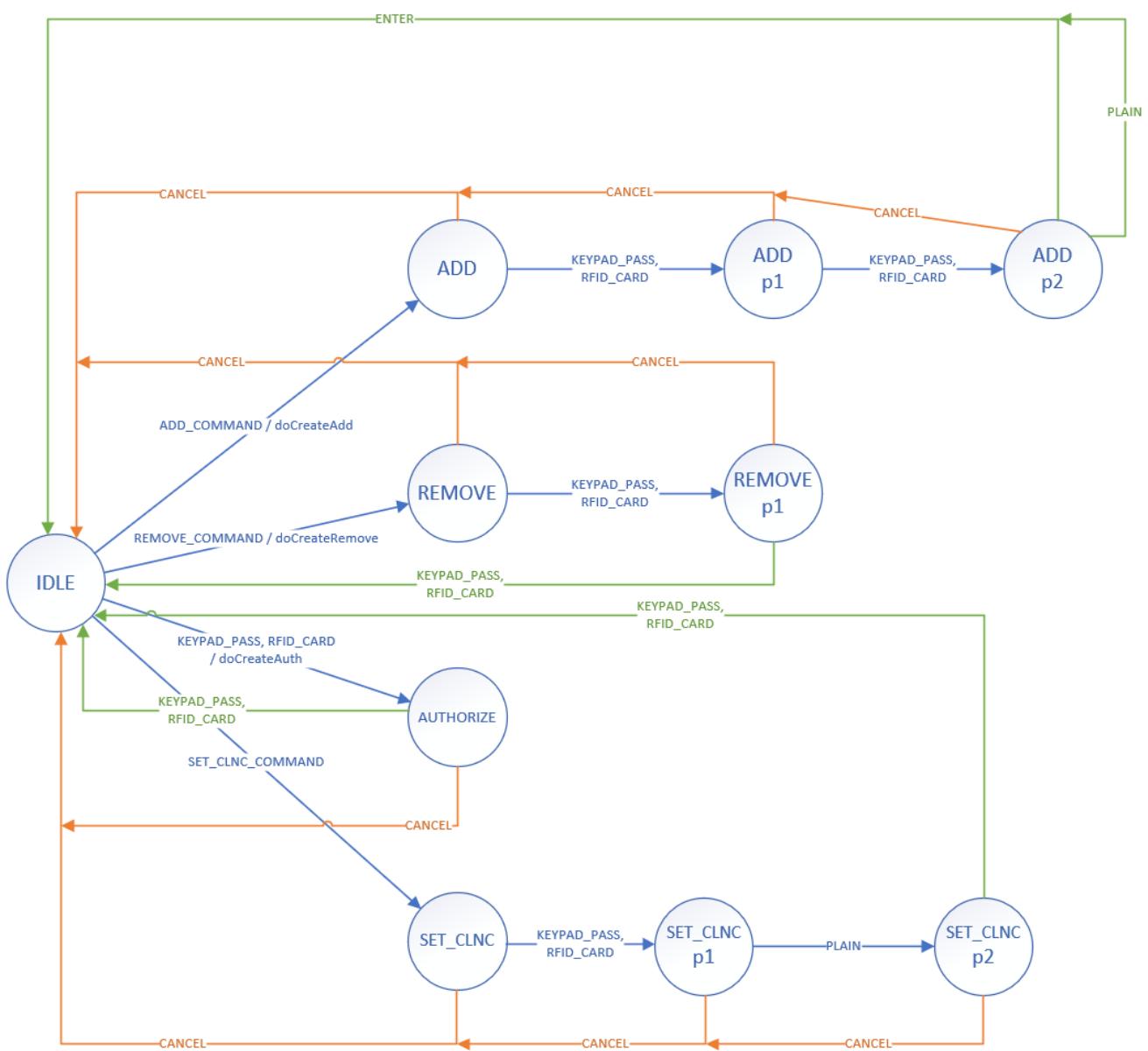
Za svaki ulazni uređaj pokreće jednu programsku nit koja čita podatke s ulaznog uređaja te ih, često uz neku jednostavnu obradu, pošalje na imenovani cjevovod (FIFO).



Slika 3.2 Blokovski prikaz ustroja Hardwaredaemon procesa

Posebna programska nit implementira automat koji pomoću standardne *Linux* funkcije `poll()` [9] koja prati definirani niz opisnika datoteka (engl. *file descriptor*) te signalizira kada je jedan od opisnika datoteka spreman za ulazno/izlazne operacije. S obzirom da se imenovani cjevovodi na *Linux*-u opisuju opsinicima datoteka, funkcija `poll()` se može koristiti i nad cjevovodima signalizirajući kada postoje podaci u cjevovodu koji je moguće pročitati.

Navedeni automat, nazvan *InputAutomaton*, služi za sastavljanje zahtjeva u obliku poruka (objekti klase *CommandMessage*) koje se prosljeđuju *MainApp* procesu.



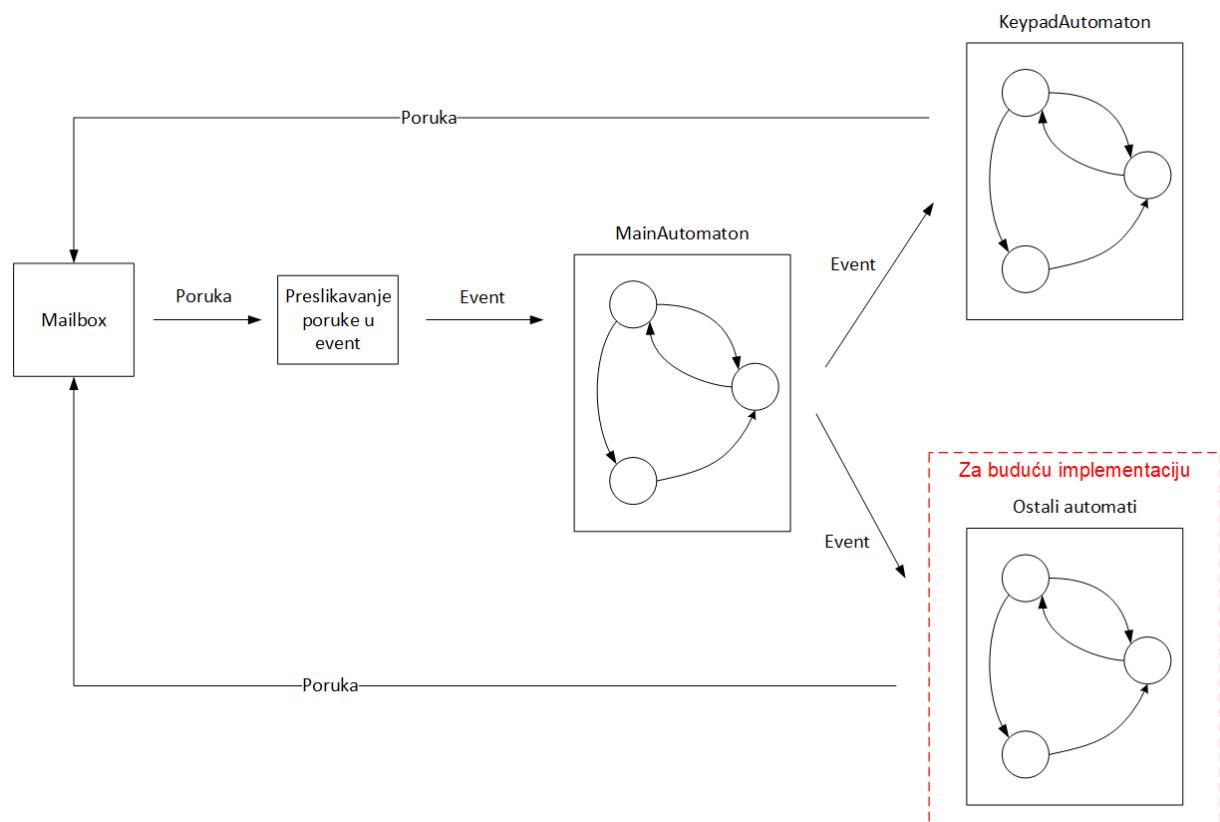
Slika 3.3 Prikaz stanja i prijelaza automata InputAutomaton

Na slici "Slika 3.3 Prikaz stanja i prijelaza automata InputAutomaton" imena na strelicama predstavljaju događaje/okidače (engl. *event*) automata.

U slučaju uspješno izvršenog ciklusa stanja (prijelaz označen zelenom strelicom) automat prosljeđuje uneseni zahtjev, odnosno objekt klase *CommandMessage* koji opisuje unesene podatke kao što je naprimjer zahtjev za dodavanjem RFID kartice, procesu *MainApp*.

Osim ulaznih uređaja, proces *HardwareDaemon* stvara i programsku nit *IndicatorsThread* koja upravlja izlaznim uređajima kao što su zujalica, LCD zaslon i električna brava. *IndicatorsThread* posjeduje svoj sandučić (engl. *Mailbox*) preko kojeg samostalno obrađuje zahtjeve nad izlaznim uređajima.

3.3.2 MainApp



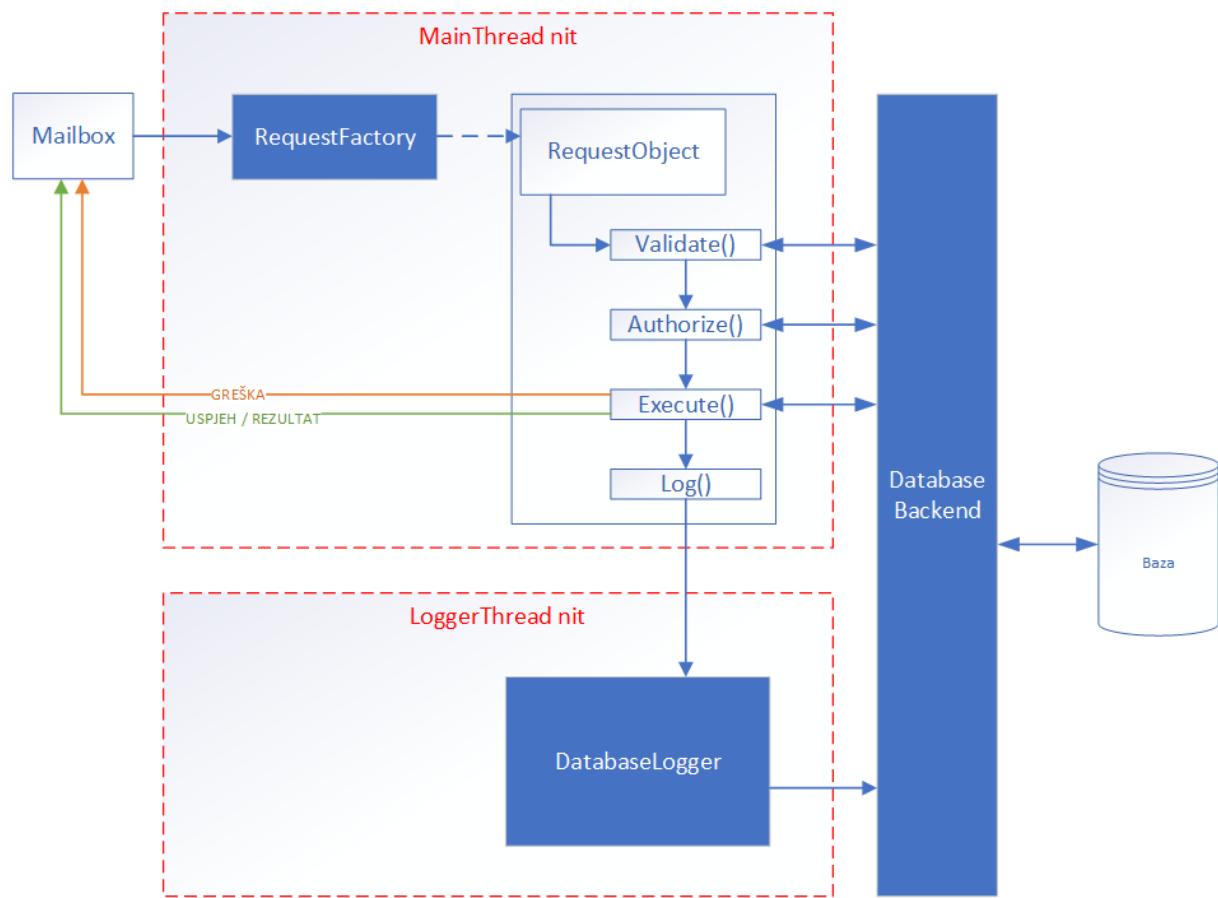
Slika 3.4 Blokovski prikaz strukture procesa *MainApp*

Proces *MainApp* zadužen je za upravljanjen globalnim stanjem sustava odnosno sesijama sustava. Sastoji se od dva sloja međusobno povezanih automata od kojih je prvi sloj zadužen da osigura da se u bilo kojem trenutku izvršava samo jedna sesija i sastoji se od jednog automata nazvanog *MainAutomaton*. Drugi sloj automata zadužen je za izvršavanje sesije. Sastoji se od jednog automata, nazvanog *KeypadAutmaton*, u trenutnoj implementaciji ali svojom strukturom podržava više automata ovisno o ulaznim/izlaznim krajnjim točkama.

Automati drugog sloja služe za upravljanjem sustavom na razini više poruka, odnosno za zahtjeve koji traže da stanje sustava ostane očuvano, za razliku od zahtjeva koji ne ovise o stanju sustava (osim iznimnih stanja kao što su greške) kao što su zahtjevi za autorizacijom i otvaranjem vrata.

Osnovna funkcija podsustava kojeg čini *MainApp* proces je prenošenje poruka između ulazno/izlaznog podsustava (*HardwareDaemon* proces) i podsustava za pristup bazi podataka (*DatabaseGateway* proces).

3.3.3 DatabaseGateway



Slika 3.5 Blokovski prikaz unutrašnje strukture DatabaseGateway modula

Podsustav *DatabaseGateway* zadužen je za obradu zahtjeva nad bazom podataka što uključuje proveru, brisanje i dodavanje podataka u bazu podataka. Oslanja se intenzivno na višeslojni sustav pristupa bazi podataka (engl. *Database Backend*).

RequestObject klasa predstavlja jedan zahtjev nad bazom podataka, te funkcioniра kao bazna klasa prema kojoj se projektiraju klase za svaki od zahtjeva. Svaki zahtjev se izvršava u četiri koraka/funkcije:

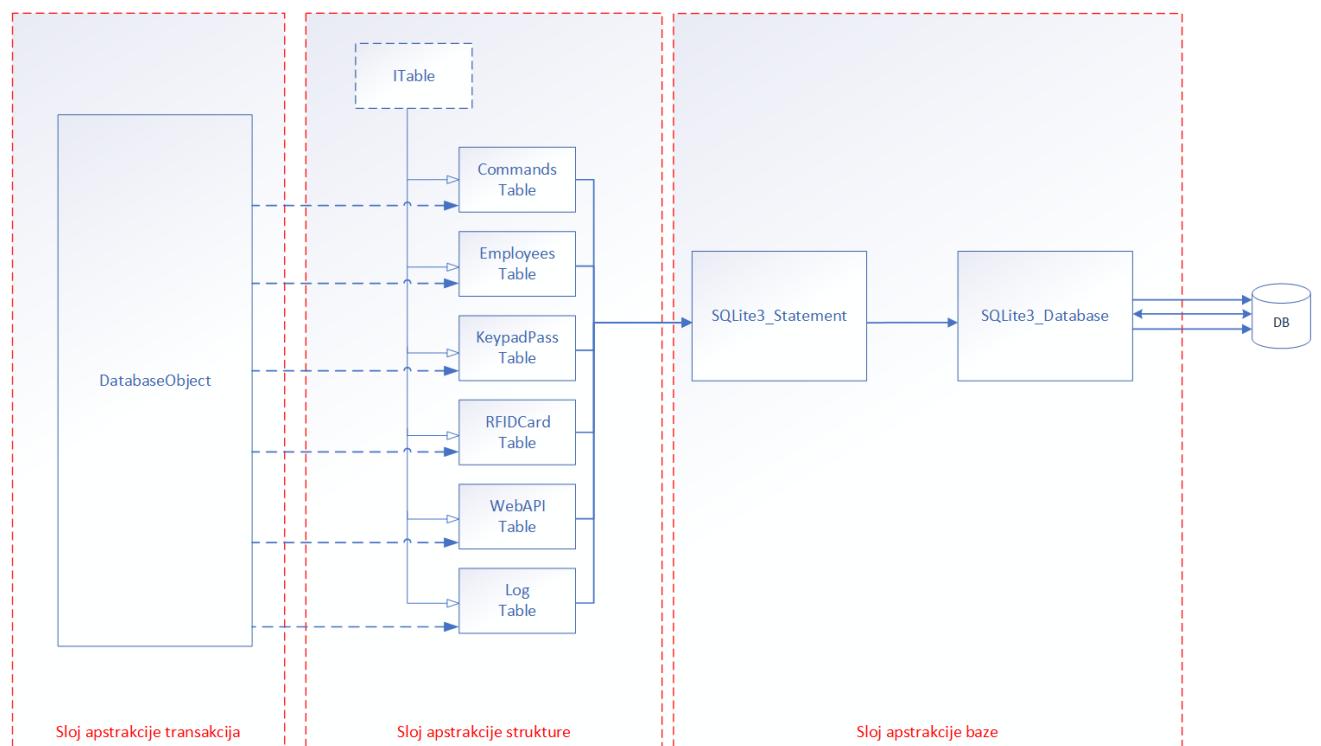
- Provjera ispravnosti podataka u zahtjevu (engl. *Validate*)
- Autorizacija podnositelja zahtjeva u odnosu na potrebnu razinu prava (engl. *Authorize*)
- Izvršavanje zahtjeva (engl. *Execute*)

- Zapisivanje izvedene operacije (engl. *Log*)

implementiranih u baznoj klasi *RequestObject*. Svaki zahtjev se izvodi kao posebna klasa koja nasljeđuje klasu *RequestObject* te reimplementira (engl. *override*) navedene četiri funkcije.

Prema slici "**Error! Reference source not found.**" može se vidjeti da se sustav za pristup bazi podataka sastoji od tri sloja:

- *Sloj apstrakcije baze* - sloj apstrakcije osnovnih funkcija nad *SQLite* bazom
- *Sloj apstrakcije strukture* - sloj apstrakcije strukture tablica
- *Sloj apstrakcije transakcija* - sloj apstrakcije više jednostavnih operacija nad tablicama



Slika 3.6 Slojevita struktura sustava za pristup bazi podataka

3.4 *Driver* sloj

Driver sloj predstavlja skupinu hierarhijski organiziranih i povezanih dinamičkih biblioteka čija je zadaća implementirati zadalu funkcionalnost niske razine kao što je naprimjer upravljanje ulazno/izlaznim uređajima ili upravljanje zapisima (engl. *log*).

U sljedećim poglavljima bit će razrađen dio istaknutih biblioteka.

3.4.1 *Mailbox*

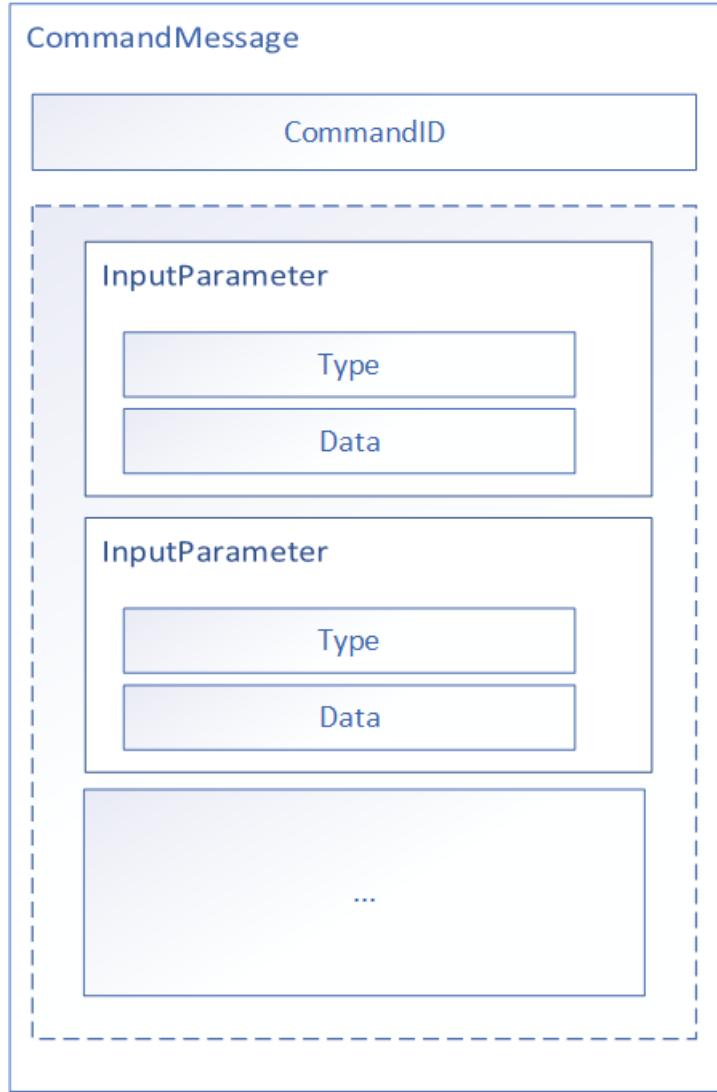
Jedan od sustava međuprocesne i međunitne komunikacije temelji se na slanju poruka nazvanom *Mailbox* (sandući u slobodnom prijevodu). Omogućava slanje poruka između procesa zbog činjenice da se temelji na POSIX *message queue* [10] koji koristi samu funkcionalnost *Linux* jezgre za prenošenje poruka. Projektiran je po uzoru na TCP/IP model komunikacije te omogućava bespojno i spojevno slanje poruka (spojevni paketi zahtjevaju potvrdu primitka). Osim toga, omogućava blokirajuće primanje poruka, neblokirajuće primanje poruka i primanje poruka s vremenskim ograničenjem (engl. *timeout*).

Sama struktura *Mailbox* sustava podijeljena je na tri sloja navedenih od hijerarhijski najvišeg do najnižeg:

- *DataMailbox* - implementira funkcionalnost slanja naprednih poruka (objekata)
- *SimplifiedMailbox* - implementira funkcionalnost slanja niza bajtova kao poruka
- *POSIX message queue* - implementiran u *Linux* jezgri, osnova *Mailbox*-a

Sve napredne poruke sloja *DataMailbox* su predstavljene vlastitim klasama koje nasljeđuju zajedničku baznu klasu. Najzanimljiviju klasu poruka predstavlja *CommandMessage*.

CommandMessage se u suštini sastoji od identifikatora *CommandID*, koji označava tip *CommandMessage* poruke, i niza parametara predstavljenih kao objekti klase *InputParameter*.



Slika 3.7 Struktura poruke klase CommandMessage

Ovakva struktura omogućava fleksibilnu prilagodbu bilokojoj upotrebi s obzirom da identifikator `CommandID` i parametri `InputParameter` nemaju značenje sami po sebi već se reimplementiraju ovisno o zahtjevima projekta.

3.4.2 Ostali moduli

Driver sloj sadrži još niz biblioteka čije funkcionalnost i ustroj ne utječu izravno na funkciranje samog sustava, a svojom kompleksnošću izlaze iz okvira trenutnih razmatranja. Posvećeno im je sljedećih nekoliko paragrafa sa kratkim opisom i objašnjenjem.

Jedan od bitnijih dijelova *Driver* sloja su *ILogger* i srodne biblioteke koje pružaju mogućnost stvaranja zapisa, najčešće u tekstualnu datoteku kao što to implementira biblioteka *Logger*. *ILogger* predstavlja baznu klasu koju nasljeđuju sve ostale biblioteke koje implemetiraju mogućnost stvaranja zapisa što omogućava jednostavnu i dinamičku promjenu načina stvaranja zapisa koristeći dinamički polimorfizam klase i objekata. Slijedi kratki pregled klasa koje nasljeđuju *ILogger*:

- *Logger* - stvaranje zapisa u obliku tekstualnih datoteka
- *NullLogger* - nul objekt - ignorira dane zapise
- *ThreadLoggerClient*, *ThreadLoggerServer* - adapteri za višenitno zapisivanje
- *LoggerToStdOut* - ispisivanje zapisa na zaslonu

Drugi bitni dio *Driver* sloja čini biblioteka *Kernel* koja sadržava minimalnu funkcionalnost potrebnu za usklađen rad svih dijelova sustava. Biblioteka implementira funkcije za zapisivanje grešaka, upozorenja i informacija na način standadiziran na razini sustava/projekta. Isto tako implementira i sustav za prisilno prekidanje izvršavanja procesa u slučaju greške. Osim funkcija, biblioteka implementira i određene konstante zajedničke svim dijelovima sustava. Naslanja se na biblioteke za stvaranje zapisa.

Najvažniji i najkompleksniji dio *Driver* sloja čini *Watchdog* skup biblioteke koje omogućavaju nadzor sustava i oporavak od zastoja. Način na koji funkcioniraju je tako što *Watchdog* server osluškuje puls (engl. *heartbeat*) signale/poruke podređenih *Watchdog* klijenata (najčešće procesi ili programske niti). U slučaju izostanka pulsa nekog od klijenata određen broj puta (parametar TTL - engl. *Time To Live*), server diže uzbunu. Način na koji je *Watchdog* implementira garantira najbrže moguće prenošenje pulseva korištenjem segmenata memorije dijeljenih među procesima te dinamičko uključivanje/isključivanje klijenata.

Osim navedenih biblioteka *Driver* sloj sadrži još biblioteka kao što su biblioteke koje pružaju standardizirano sučelje za upravljanje funkcijama specifičnim operacijskom sustavu te sučelja koja implementiraju RAII programsku filozofiju za sigurnije upravljenje resursima. Primjeri takvih biblioteka su razne biblioteke za korištenje cjevovoda, dijeljenih segmenata memorije, upravljanje procesima i mnoge druge.

4 ZAKLJUČAK

U ovome radu obrađena je jedna implementacija sustava kontrole pristupa. Rješenje (implementacija) omogućava kontrolu pristupa preko električne brave, predstavljene relejem, s audio-vizualnim indikatorima kao što su zujalica i LCD zaslon. Temeljeno je na razvojnoj platformi Raspberry Pi 3B koja pokreće *Raspbian Lite* operacijski sustav, jedan od OS-ova iz *Linux* skupine temeljen na *Debian Linux*-u, a sustav je razvijan koristeći C++ programski jezik uz neke dijelove koji su razvijeni uz pomoć Qt C++ razvojnog okvira.

Sama autorizacija moguća je preko RFID/NFC kartica, za što sustav posjeduje PN532 RFID/NFC čitač, ili PIN-a sastavljenog od 4 do 10 znamenki preko numeričke tipkovnice. Sustavom se može do određene mjere upravljati preko numeričke tipkovnice, naprimjer registriranje novih kartica/PIN-ova, brisanje istih, mijenjanje razine prava pojedine osobe.

Svi podaci su lokalizirani u lokalnoj SQLite bazi podataka, dok sama struktura sustava omogućava i jednostavno proširenje na udaljenu bazu podataka. Svaka osoba registrirana u sustavu posjeduje oznaku razine prava. Postoje tri glavne skupine razina prava; osobe bez prava pristupa, gosti samo s pravom pristupa i osoblje s pravom pristupa koje je organizirano u proizvoljne hijerarhijske razine prava koje određuju ovlasti odnosno mogućnosti izvršavanja naredbi. Svaka naredba, kao što je registriranje nove kartice, ima dodijeljenu minimalnu razinu prava potrebnu za izvršavanje, koja je postavljena u bazi podataka.

Osim kontrole pristupa, sustav stvara zapise za svaku autorizaciju (pristup) i izvršene naredbe koje se spremaju u bazu podataka iz koje se kasnije mogu iščitavati. Sustav je pokrenut kao servis (engl. *daemon* na *Linux* OS-u) te ga je potrebno samo uključiti bez dodatnog postavljanja da bi ispravno radio. Sustav posjeduje određen broj modula specijaliziranih za nadzor i oporavak od kvarova da bi radio što je moguće pouzdanije. U slučaju kvara, sami operacijski sustav ponovno pokrene proces. Neke od postavki moguće je mijenjati u datoteci nazvanoj *config.xml*.

LITERATURA

- [1] Broadcom Corporation, "BCM2835 Peripherals," 6 February 2012. [Online]. Available: <https://datasheets.raspberrypi.org/bcm2835/bcm2835-peripherals.pdf>. [Accessed 3 July 2021].
- [2] NXP, "PN532 User Manual," 5 November 2007. [Online]. Available: <https://www.nxp.com/docs/en/user-guide/141520.pdf>. [Accessed 2021 July 3].
- [3] Links, "Numerička tipkovnica DELOCK KeyPad, crna, USB," Delock, [Online]. Available: <https://www.links.hr/hr/numericka-tipkovnica-delock-keypad-crna-usb-101200271>. [Accessed 3 July 2021].
- [4] Shenzhen Eone Electronics Co., Ltd., "Specification for LCD Module 1602A-1," [Online]. Available: <https://www.elecrow.com/download/LCD1602.pdf>. [Accessed 3 July 2021].
- [5] Texas Instruments, "PCF8574 Remote8-Bit I/O Expander for I2C Bus," March 2015. [Online]. Available: <https://www.ti.com/lit/ds/symlink/pcf8574.pdf>. [Accessed 3 July 2021].
- [6] Songle Relay, "Relay ISO9002 SRD," [Online]. Available: <https://datasheetspdf.com/pdf-file/720556/Songle/SRD-05VDC-SL-C/1>. [Accessed 3 July 2021].
- [7] e-radionica.com, "Logic-level converter (made by e-radionica)," e-radionica.com, [Online]. Available: <https://e-radionica.com/en/logic-level-converter-eradionica.html>. [Accessed 3 July 2021].
- [8] P. Alberts, "Logic_Level_Bidirectional," 26 September 2013. [Online]. Available: http://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf. [Accessed 3 July 2021].
- [9] M. Kerrisk, "poll(2) — Linux manual page," 22 March 2021. [Online]. Available: <https://man7.org/linux/man-pages/man2/poll.2.html>. [Accessed 3 July 2021].

[10] M. Kerrisk, "mq_overview(7) — Linux manual page," 9 June 2020. [Online]. Available: https://man7.org/linux/man-pages/man7/mq_overview.7.html. [Accessed 3 July 2021].

POPIS OZNAKA I KRATICA

API	<i>Application Programming Interface</i>
ARM	<i>Advanced RISC Machines</i>
BCM	<i>Broadcom (Inc.)</i>
CMOS	<i>Complementary Metal-Oxide-Semiconductor</i>
CS	<i>Chip Select</i>
CSI	<i>Camera Serial Interface</i>
DSI	<i>Display Serial Interface</i>
FIFO	<i>First In First Out</i>
GB	<i>GigaByte</i>
GND	<i>GrouND</i>
GPIO (Pin)	<i>General Purpose Input/Output (Pin)</i>
HDMI	<i>High Definition Media Interface</i>
HV	<i>High Voltage</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol</i>
LCD	<i>Liquid Crystal Display</i>
LV	<i>Low Voltage</i>
MISO	<i>Master In Slave Out</i>
MOSI	<i>Master Out Slave In</i>
NC	<i>Normally Closed</i>
NFC	<i>Near Field Communications</i>
NO	<i>Normally Open</i>
OS	<i>Operating System</i>

POSIX	<i>Portable Operating System Interface</i>
RAII	<i>Resource acquisition is initialization</i>
RFID	<i>Radio Frequency IDentification</i>
SCK	<i>Serial ClocK</i>
SD (<i>Card</i>)	<i>Secure Digital (<i>Card</i>)</i>
SoC	<i>System on Chip</i>
SPI	<i>Serial Peripheral Interface</i>
SQL	<i>Structured Query Language</i>
SSH	<i>Secure SHell</i>
TCP	<i>Transmission Control Protocol</i>
TTL (<i>Family</i>)	<i>Transistor-Transistor Logic (<i>Family</i>)</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>
USB	<i>Universal Serial Bus</i>

SAŽETAK

ELEKTRONIČKA BRAVA UPRAVLJANA NFC TEHNOLOGIJOM

U posljednje vrijeme na popularnosti sve više dobivaju elektroničke metode kontrole pristupa. Jedno od takvih sustava obrađeno je u ovome radu. Obrađeni sustav funkcionira tako što preko ulaznih perifernih uređaja, kao što su numeročka tipkovnica i RFID/NFC čitač, prepoznaže osobu koja želi pristupiti određenoj prostoriji (ili resursu) zaštićenoj elektroničkom bravom (relejem). Koristeći programsku podršku pokrenutu na razvojnoj platformi *Raspberry Pi*, sustav autorizira i evidentira osobu koja želi pristupiti prostoriji, obavijesti korisnika preko audio-vizualnih perifernih uređaja (LCD zaslon, zujalica) te otključava vrata na određeno vrijeme u slučaju da osoba posjeduje prava pristupa. Glavna tema ovoga rada je arhitektura sustava koja je razvijena tako da sustav bude pouzdan i održiv te lako nadogradiv.

Ključne riječi: kontrola pristupa, RFID, NFC, sigurnost, *Raspberry Pi*

SUMMARY

NFC-CONTROLLED ELECTRONIC LOCK

Electronic access control systems have been steadily gaining on popularity lately. One of the forementioned systems has been described in this thesis. The system reads input from its peripheral devices, such as numeric keypad and RFID/NFC reader, and uses the data to identify and authorize the person trying to gain access to a certain room (or a resource) protected by an electronic lock (relay). Using software running on embedded platform Raspberry Pi, the system authorizes and records the person trying to get access to the room, notifies the person using its audio-visual peripheral devices (LCD screen, buzzer) and unlocks the door for a certain amount of time if the person has access privileges. The main thematic of this thesis is the system architecture which has been designed to be reliable, maintainable and easily upgradeable.

Keywords: access control, RFID, NFC, security, *Raspberry Pi*