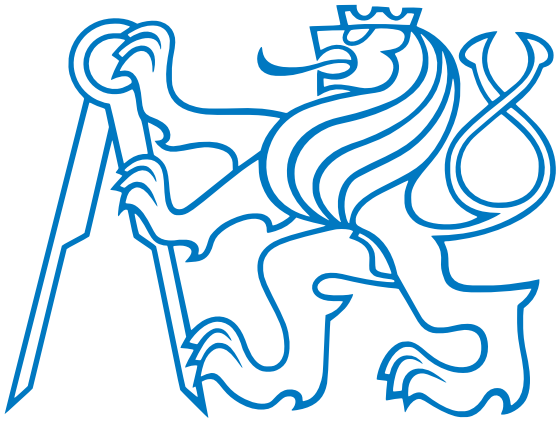
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA MĚŘENÍ



BAKALÁRSKA PRÁCA

ARM9 platforma ako low-cost riešenie pre plnohodnotné embedded aplikácie

Autor:
Stanislav DROZD

Vedúci práce:
Ing. Ján TOMLAIN

Praha, máj 2016


4/2016

Názov práce: ARM9 platforma ako low-cost riešenie pre plnohodnotné embedded aplikácie

Autor: Stanislav Drozd

Katedra (ústav): Katedra měření

Vedúci bakalárskej práce: Ing. Ján Tomlain

e-mail vedúceho: tomlajan@fel.cvut.cz

Abstrakt V predloženej práci prezentujeme využitie procesora s jadrom ARM9 vo vstavaných zariadeniach. Objektom skúmania bol procesor Freescale i.MX28 založený na architektúre ARM9, ktorý je implementovaný v priemyselne využívanom počítačovom module Ka-Ro TX28. Tento modul je vsadený do riadiacej jednotky systému na získavanie odpočtov z meračov energií od firmy CSNet nazývaného DAVE, ktorý má všetky potrebné periférie pre náš výskum. Vypracovali sme návrh, v ktorom použijeme DAVE ako riadiacu jednotku prístupového systému do posilňovne s ovládaním zámkov dvoch dverí s použitím troch RFID Mifare čítačiek, snímaním priestoru pomocou pohybových senzorov a komunikáciou so vzdialeným databázovým a poštovým serverom prostredníctvom privátnej LAN siete. Pri návrhu sme počítali aj s možnosťou rozšírenia tohto systému o ďalší elektricky ovládaný zámok a RFID modul. Spomínaný návrh sme zrealizovali a pripravili na uvedenie do prevádzky.

Kľúčové slová: ARM9, Ka-Ro TX28, embedded, RFID, Mifare

Title: ARM9 Platform as Low-cost Solution for Full-featured Embedded Applications

Author: Stanislav Drozd

Department: Department of Measurement

Supervisor: Ing. Ján Tomlain

Supervisor's e-mail address: tomlajan@fel.cvut.cz

Abstract In the present work we study the usage of ARM9 based processor in embedded devices. The main subject of this research was Freescale i.MX28 processor based on ARM9 architecture, which is implemented in the computer module Ka-Ro TX28 commonly used in industry. This module is embedded into a system controller (called DAVE) designed for data collection from energy meters , which has all the necessary peripherals for our research. We have developed a proposal in which is the DAVE used as a controller in access control system for a gym. This system is composed of two electric door locks, three Mifare RFID readers and motion sensors. It communicates with remote database and mail server via a private LAN. In the design we counted also with the possibility of extending this system by another electrically actuated door lock and RFID module. The mentioned proposal was implemented and now is prepared for commissioning.

Keywords: ARM9, Ka-Ro TX28, embedded, RFID, Mifare

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2015-16**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	**Stanislav Drozd**
Studijní program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**ARM9 platforma ako low-cost riešenie pre plnohodnotné embedded aplikácie**
Název tématu anglicky:	**ARM9 Platform as Low-cost Solution for Full-featured Embedded Applications**

Pokyny pro vypracování:

Primárny cieľ práce: Vytvorenie aplikácie spolupracujúcej s embedded HW platformou ARM9

Čiastkové ciele:

I.) analýza procesora Freescale i.MX287 najmä z pohľadu architektúry a periférii

II.) oživenie embedded HW core modulu KARO TX28(s)

III.) naprogramovanie samostatnej aplikácie pre výmenu dát medzi serverom a embedded klientom

IV.) evaluácia a testovanie platformy s ohľadom na jej stabilitu a korektné fungovanie

Seznam odborné literatury:

[1] i.MX28 Applications Processors for Consumer Products, Freescale Semiconductor (dostupné online)

[2] i.MX28 Applications Processor Reference Manual, Freescale Semiconductor (dostupné online)

[3] ARMSDK-VM Virtual Appliance A preconfigured Linux system (dostupné online)

[4] Languages for Embedded Systems and their Applications ISBN 978-1-4020-9714-0

Vedoucí bakalářské práce:	Ing. Ján Tomlain
Datum zadání bakalářské práce:	8. prosince 2015
Platnost zadání do[1]:	30. září 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta elektrotechnická L. S.

Doc. Ing. Jan Holub, Ph.D.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 8. 12. 2015

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s *Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací*.

V Prahe dňa 27. mája 2016 Stanislav Drozd

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

KAPITOLA 1

ZOZNAM SKRATIEK

API - Application Programming Interface - rozhranie pre programovanie aplikácií

CAN - Controller Area Network

CPU - Central Processing Unit - centrálna procesorová jednotka

ČVUT - České vysoké učení technické

DB - Databáza

DC - Direct Current - jednosmerný prúd

DCE - Data Circuit-terminating Equipment - zariadenie ukončujúce dátový okruh (napr. modem)

DDR - Double Data Rate

DNS - Domain Name System - systém názvov domén

DPS - Doska Plošných Spojov

DTE - Data Terminal Equipment - koncové zariadenie prenosu dát (napr. počítač)

EZS - Elektronický zabezpečovací systém

GPIO - General-Purpose Input/Output - multifunkčné vstupno-výstupné rozhranie

GUI - Graphical User Interface - grafické používateľské rozhranie

HASL - Hot Air Solder Leveling

ISO/OSI - International Organization for Standardization / Open Systems Interconnection

kH/s - kilo Hash za sekundu

I/O - Input/Output - vstup/výstup

IP - Internet Protocol - Internetový protokol

LAN - Local Area Network - lokálna počítačová sieť

LCD - Liquid crystal display - displej s kvapalnými kryštálmi

LED - Light-Emitting Diode - luminiscenčná dióda

MAC - Media Access Control - riadenie prístupu k médiu

MCU - MicroController Unit - mikrokontrolér

MH - Mega Hash

MMC - MultiMediaCard

MS - Memory Stick

OS - Operating System - operačný systém

OTG - On-The-Go

PC - Personal Computer - osobný počítač

PIR - Passive Infrared Sensor - pasívny infračervený senzor

PMU - Power Management Unit - riadiaca jednotka napájania

PWM - Pulse Width Modulation - impulzová šírková modulácia

REST - Representational State Transfer

RFID - Radio Frequecy IDentification - vysokofrekvenčná identifikácia

ROM - Read-Only Memory - permanentná pamäť

RTC - Real-Time Clock - hodiny reálneho času

RxD - Receive Data - prijímané dáta (v DTE z DCE)

SDIO - Secure Digital Input Output

SDK - Sinkuleho a Dejvická kolej

SMD - Surface Mount Device - súčiastka určená na povrchovú montáž

SPI - Serial Peripheral Interface - sériové periférne rozhranie

SDRAM - Synchronous Dynamic Random Access Memory - synchrónna dynamická pamäť s priamym prístupom

SRAM - Static Random Access Memory - statická pamäť s priamym prístupom

SÚZ - Správa účelových zařízení ČVUT

TxD - Transmit Data - vysielané dáta (z DTE do DCE)

UART - Universal Asynchronous Receiver/Transmitter - asynchrónne sériové rozhranie

USB - Universal Serial Bus - univerzálna sériová zbernica

KAPITOLA 2

ÚVOD

Priemyselná revolúcia priniesla do spoločnosti mnohé zmeny, od výroby parných strojov až po automatizáciu bežných pracovných postupov. Čím boli ovplyvnené myšlienky priemyselných velikánov, nevieme. Je ale známe, že človek je od prírody tvor pohodlný, avšak vynaliezavý. Vďaka týmto dvom vlastnostiam si dokáže život postupne zjednodušovať. Aj to je jeden z kľúčov k rozkvetu vedy a techniky, akú poznáme dnes. S istotou môžeme povedať, že stavebným kameňom moderných technológií sú počítače. Život bez nich si už väčšina z nás ani nedokáže predstaviť. Za posledné polstoročie sa životný štýl rapídne zmenil a nám nezostáva nič iné, ako sa prispôsobiť a naplno využiť technológie, ktoré sú dostupné k nášmu ďalšiemu rozvoju.

Jedným z primárnych prvkov používaných v automatizácii sú embedded, čiže vstavané, zariadenia, ktoré sú riadené malými počítačmi. Tie sú narozdiel od jednočipových mikrokontrolérov omnoho vyspelejšie, či už výpočtovým výkonom alebo množstvom a typmi dostupných rozhraní. Na trhu je dostupná celá rada rôznych embedded core modulov, ako napr. APC - Rock, BeagleBoard, Phytec - Cosmic Board, MicroZed, PandaBoard a mnoho ďalších. Nesmieme zabudnúť ani na populárnu „malinu“ - Raspberry Pi. Výhodou väčšiny z nich je veľká komunita, ktorá za nimi stojí a s tým spojená široká podpora a množstvo projektov a rôznej dokumentácie. Je to síce obdivuhodné, ale majú jednu zásadnú nevýhodu - svojimi parametrami nie sú vhodné na využívanie v priemysle. Dá sa povedať, že väčšina z nich je priamo určená pre vývoj a zábavu.

V tejto práci sa zameriame na cenovo dostupný, procesorom Freescale i.MX28 osadený, embedded modul Ka-Ro TX28. Základným cieľom práce je analyzovať procesor Freescale i.MX28, pripraviť embedded modul do prevádzky a naprogramovať aplikáciu, ktorá na ňom bude vykonávať požadované operácie.

Ako reálny projekt bude navrhnutý prístupový systém do posilňovne na Sinkuleho koleji, ktorý bude pozostávať z riadiacej jednotky, RFID čítačiek, elektrického systému otvárania dverí a pohybových senzorov. Používatelia sa pred vstupom do posilňovne budú autorizovať pomocou RFID Mifare kariet, akým je napríklad študentský preukaz *ISIC*. Projekt ponesie názov *Elektra*.

Celá práca bude rozdelená na 2 základné časti - hardware a software.

V časti hardware bude opísaná fyzická štruktúra systému a všetky jeho hmotné prvky - napájací zdroj, riadiaca jednotka, obvod vstupov a výstupov, 2 typy RFID modulov a senzory.

Časť software bude pojednávať o rôznych použitých programových konštrukciách. V prvom rade pôjde o hlavný riadiaci program napísaný v skriptovacom jazyku Python, ktorý je však vhodný aj na väčšie projekty. Tento program bude zabezpečovať komunikáciu so vzdialeným databázovým serverom, autorizáciu osôb pomocou RFID kariet, posielanie inštrukcií mikrokontrolérom v RFID čítačkách pre svetelnú a zvukovú signalizáciu a v neposlednom rade ovládanie zámkov dverí a zber dát zo senzorov pohybu a otvorených dverí.

V kapitole *Plány do budúcnosti* budú načrtnuté možnosti ďalšieho vylepšenia systému, ktoré boli vymyslené počas práce na projekte a už nebolo možné ich zrealizovať.

KAPITOLA 3

CIELE PRÁCE

Hlavným cieľom tejto práce je zostaviť zariadenie, ktoré by demonštrovalo možnosť využitia platformy ARM9 vo vstavaných zariadeniach. Tento hlavný cieľ bol rozdelený do niekoľkých čiastkových cieľov.

Prvým je analýza procesora Freescale i.MX287 z pohľadu architektúry a periférií. Tento krok je dôležitý na oboznámenie sa s možnosťami procesora a jeho následné začlenenie do návrhu uceleného systému. Bude využitá jeho implementácia v embedded module Ka-Ro TX28, ktorý sa medzi ostatnými modulmi podobných parametrov javí ako rozumný kompromis medzi cenou a výkonom.

Druhým krokom bude oživenie samotného hardvérového modulu. Ka-Ro TX28 je potrebné pre využitie jeho periférií vsadiť buď do vývojového kitu na to určeného, alebo využiť nejaké zariadenie vyrobené na mieru. Takýto návrh by už bol nad rámec našej práce, preto bude použitý jeden z modulov od firmy CS-Net.Čo sa naviac týka hardvérovej časti, bude potrebné navrhnúť zariadenia, kde jedno z nich bude slúžiť na demonštráciu komunikácie s procesorom i.MX287 po sériovej linke a druhé na testovanie funkčnosti vstupno-výstupného rozhrania. Zo softvérovej časti bude potrebné, pre oživenie core modulu, pripraviť operačný systém, ktorý sa bude starať o funkčnosť systému a chod ďalších obslužných aplikácií.

Ďalším cieľom je naprogramovanie samostatnej aplikácie pre výmenu dát medzi serverom a embedded klientom. V prvom rade pôjde o napísanie programu na komunikáciu s databázovým serverom. Ak sa bude jednať o začlenenie do už existujúceho celku, bude potrebné prispôsobiť aj stávajúcu databázu. Následne bude program rozširovaný o ďalšie funkcie a výmenu dát s inými zariadeniami.

Na záver je plánovaná evaluácia a testovanie platformy s ohľadom na jej stabilitu. Tento cieľ má skôr dlhodobejší charakter, pretože, aby sa dalo s istotou povedať, či a ako je navrhnutý systém stabilný, je potrebné jeho začlenenie do každodennej prevádzky a dlhodobé testovanie.

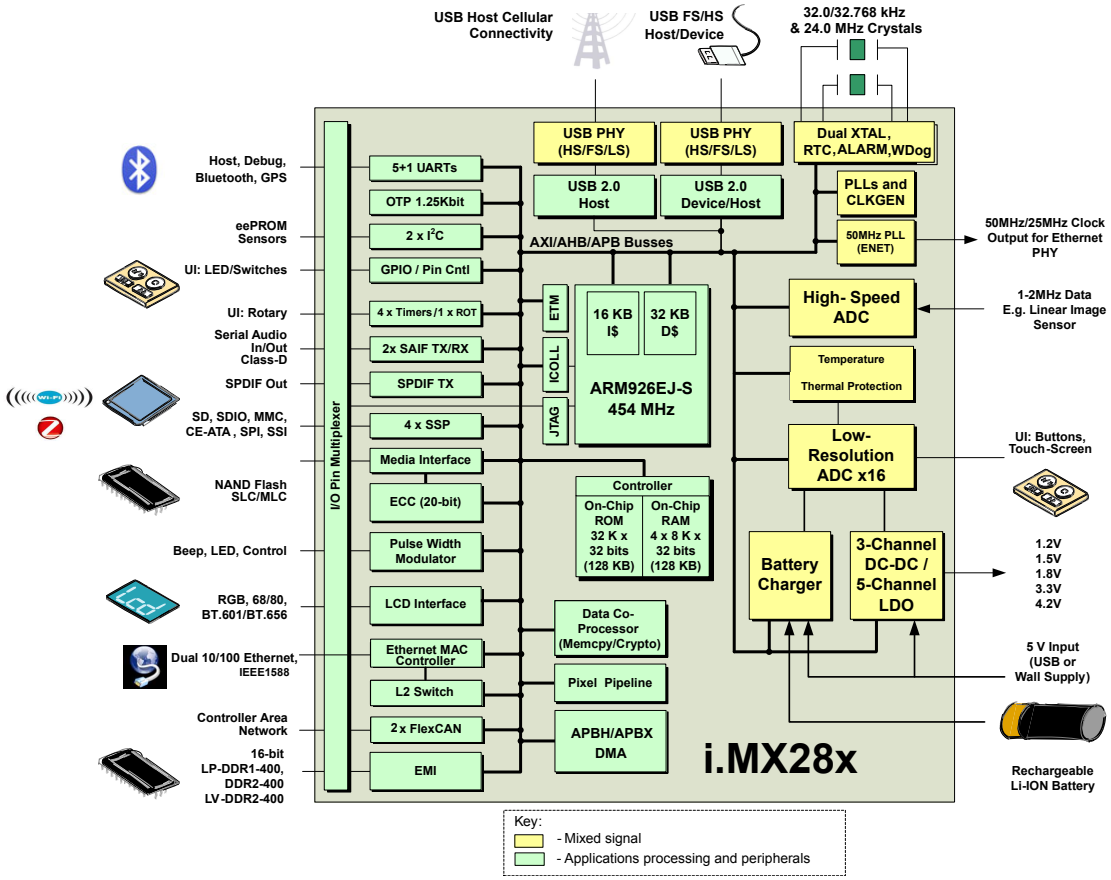
KAPITOLA 4

ANALÝZA PROCESORA FREESCALE I.MX28

Ako zástupca architektúry ARM9 bol zvolený procesor Freescale i.MX28 využívaný v embedded zariadeniach. Ako je uvedené v katalógovom liste [6], dokáže pracovať s taktovaciou frekvenciou až 454 MHz. Procesor je vsadený do puzdra MAPBGA-289 s rozmermi 14 x 14 mm a rozostupom pinov 0,8 mm.

Základné prvky [7]

- CPU ARM926EJ-S
  - vyrovnávacia pamäť: 16 KB pre inštrukcie a 32 KB pre dáta
- 128 KB integrovanej nízko-odberovej pamäte SRAM zabudovanej priamo v čipe
- 128 KB integrovanej maskou programovateľnej pamäte ROM zabudovanej v čipe
- podpora až 8 NAND Flash pamäťových médií
- 4 synchrónne sériové porty (SSP) pre SDIO/MMC/MS/SPI
- 10/100 Mb/s Ethernet, 3-portový L2 switch
- 2x USB 2.0
  - z toho jedno s podporou OTG
- LCD kontrolér
- 6x UART
  - 5x UART pre aplikácie s rýchlosťou do 3,25 Mb/s s hardvérovým riadením dátového toku
  - 1x UART s rýchlosťou do 115 Kb/s určený na debugovanie
- 2x I^2C master/slave rozhranie, rýchlosť do 400 Kb/s
- 4x 32-bitový časovač
- 8x PWM
- GPIO s možnosťou prerušení
- PMU - jednotka slúžiaca na riadenie napájania



Obrázok 4.1: Zjednodušený blokový diagram rozhraní procesora i.MX28 [6]

- trojitý DC-DC prevodník
- napäťové regulátory
- nabíjanie akumulátorov

Okrem vyššie uvedeného ponúka procesor Freescale i.MX28 množstvo ďalších funkcií, ako napr. podporu 16-bitových DDR, DDR2 a LV-DDR2 pamätí, CAN, zvukové rozhrania, RTC, analógovo-digitálne prevodníky či kontrolér pre dotykové obrazovky.

4.1. Ka-Ro TX28

Obrázok 4.2: Počítačový modul Ka-Ro TX28 [11]

Vhodným kandidátom na testovanie procesora s jadrom ARM9 je počítačový modul Ka-Ro TX28 od nemeckej firmy Ka-Ro electronics, ktorý využíva práve vyššie opisovaný procesor Freescale i.MX287.

Parametre [11]

- RAM - 128 MB DDR2 SDRAM (16bit)
- ROM - 128 MB NAND Flash
- napájanie - 3,1 - 5,5 V
- rozmery - puzdro SO-DIMM 200 (67,6 mm x 26 mm x 3,6 mm)
- prevádzková teplota - -40 .. 85 °C
- podporované OS - Windows Embedded CE a Linux

Rozloženie pinov je štandardné pre celú sériu modulov TX, ide o takzvaný TX-DIMM pinout. Z tohto hľadiska je možné modul TX28 v zariadení vymeniť za iný zo série TX, s rozdielnou hardvérovou konfiguráciou. Je zrejmé, že TX moduly sú kompletné počítače kompaktných rozmerov vhodné na použitie vo vstavaných zariadeniach.

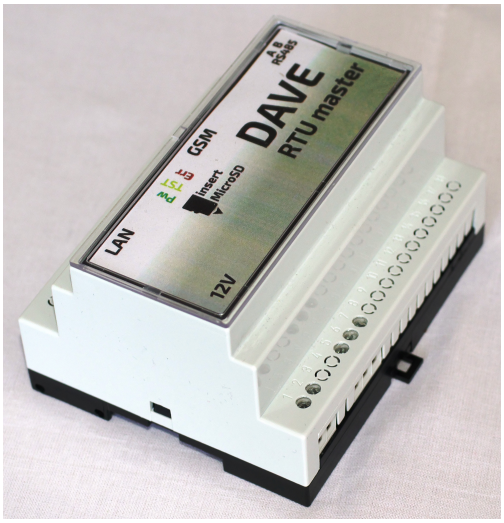
4.2. Porovnanie Ka-Ro TX28 s inými embedded platformami

Cieľom tohoto odstavca je porovnanie nami zvoleného zástupcu embedded platforiem s inými podobnými bežne používanými core modulmi, konkrétne s Raspberry Pi [1] a BeagleBone [3]. Keďže model TX28 je v rade TX momentálne najnižším ponúkaným typom, čo sa parametrov týka, bude aj v prípade Raspberry Pi uvažovaný model B. Toto porovnanie je zamerané na prvky dôležité pri návrhu embedded zariadení. Políčka označené symbolom „-“ označujú buď prvok nedostupný na danom zariadení, alebo absenciu tejto informácie v katalógovom liste výrobcu.

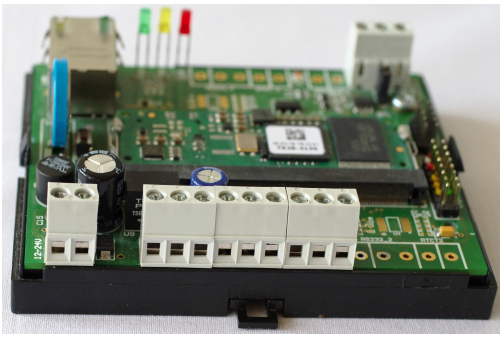
	Ka-Ro TX28	Raspberry Pi model B	BeagleBone
procesor	ARM9, 454 MHz	ARM11, 700 MHz	ARM Cortex-A8, 500/720 MHz
RAM	128 MB, DDR2 400 MHz	512 MB	128/256 MB, DDR2 400 MHz
vstavaná pamäť	128 MB, NAND Flash	-	32 KB EEPROM
RTC	DS1339	-	-
napájanie	3,1 - 5,5 V	5 V	5 - 5,2 V
rozmery	67,6 x 26 x 3,6 mm	85 x 56 x 17 mm	86 x 53 mm
teplotný rozsah	-40 - 85 °C	-	-
Ethernet	2x 100 Mb/s	1x 100 Mb/s	1x 100 Mb/s
USB	USB 2.0 OTG, USB 2.0 host	2x USB 2.0 host	USB 2.0 client, USB 2.0 host
LCD kontrolér	✓	✗	✓
CAN	3	-	1
UART	6	1	3
I^2C	2	1	1
PWM	8	-	8
GPIO	cca 100	26	65
hmotnosť	6 g	40 g	39,68 g
cena	2100 Kč	1200 Kč	2200 Kč

Tabuľka 4.1.: Porovnanie Ka-Ro TX28 s Raspberry Pi a BeagleBone

4.3. DAVE RTU master



(a) s krytom



(b) bez krytu

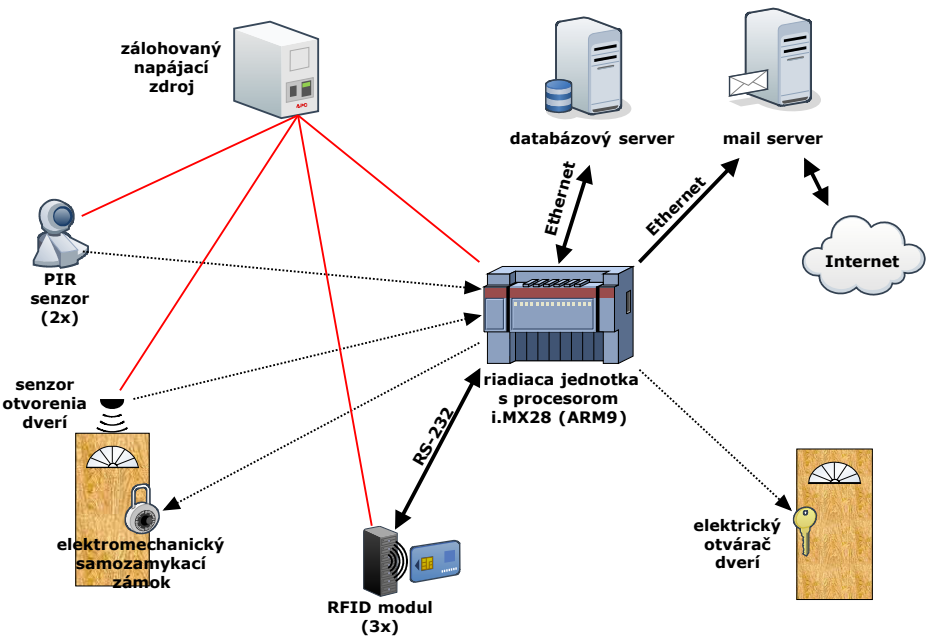
Obrázok 4.3: DAVE RTU master

Ide o riadiacu jednotku od firmy CSNet pôvodne určenú na zber a spracovávanie odpočtov z meračov energií. Je osadený počítačovým modulom Ka-Ro TX28 popísaným v odstavci 4.1. Jeho výhodou je, okrem toho, že obsahuje modul Ka-Ro TX28, možnosť uchytenia na DIN lištu a prítomnosť všetkých potrebných portov pre túto prácu - Ethernet, 6 sériových liniek RS-232 a GPIO.

KAPITOLA 5

HARDWARE

5.1. Štruktúra systému



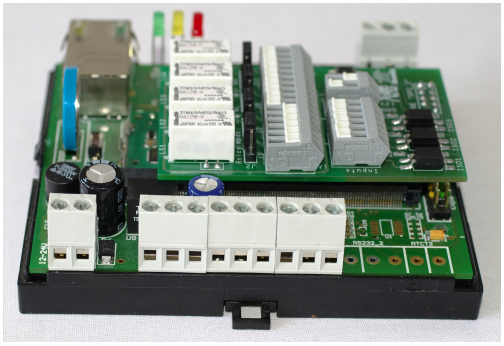
Obrázok 5.1: Blokový diagram

Systém, pracovne nazvaný *Elektra,* je zložený z niekoľkých blokov (viď obrázok 5.1), kde každý plní špecifickú funkciu. Mozgom celej sústavy je priemyselný počítačový modul Ka-Ro TX28 osadený procesorom s jadrom ARM9. Prostredníctvom Ethernetu si vymieňa informácie s databázovým serverom alebo posiela e-maily pomocou poštového servera. Z dôvodu bezpečnosti sú tieto zariadenie prepojené

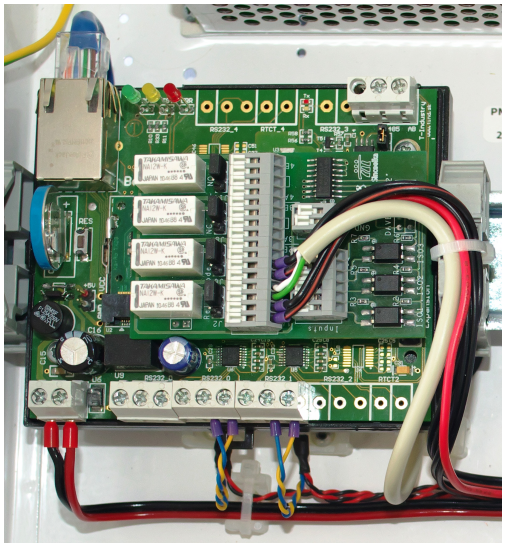
cez privátnu lokálnu sieť bez prístupu na Internet. Keďže ide o prístupový systém do posilňovne, je potrené ovládať otváranie dverí. Do spomínanej posilňovne sa vstupuje cez dvoje dvere. Prvými sa vstúpi do chodby, odkiaľ sa vchádza aj do iných miestností. Tento priestor si nevyžaduje vysokú mieru zabezpečenia, preto sa dvere nezamykajú, ale z vonkajšej strany je umiestnená guľa. Otvoria sa privedením napätia na cievku elektrického otvárača umiestneného v zárubni. Z vonkajšej strany je na stene umiestnená RFID čítačka komunikujúca s riadiacim modulom po sériovej linke RS-232. Druhé dvere sú vybavené elektromechanickým samozamykacím zámkom, ktorý je taktiež ovládaný centrálnou riadiacou jednotkou. Tu sú RFID čítačky umiestnené na oboch stranách dverí. Poloha dverí (otvorené/zatvorené) je monitorovaná jazýčkovým kontaktom. Pohyb v posilňovni detekuje dvojica PIR senzorov. O napájanie jednotlivých segmentov sa stará zdroj jednosmerného napätia 12 V zálohovaný akumulátorom. Okrem uvedeného sa táto práca zaoberá aj zostavením USB RFID čítačky/zapisovačky potrebnej pre administratívne účely spojené s registráciou používateľov do informačného systému.

5.1.1. riadiaca jednotka

Ako riadiaca jednotka slúži modul *DAVE* opísaný v kapitole 4.3. Pre projekt *Elektra* sú na ňom využité rozhrania RS-232, Ethernet a GPIO. Pre posilnenie vstupov a výstupov bol vytvorený externý modul (viď 5.5), osadený predovšetkým optočlenmi a relátkami, pripájaný k DAVE cez kolíkovú lištu.



(a) bez kabeláže



(b) osadený v rozvádzači

Obrázok 5.2: DAVE s rozširujúcim I/O modulom

5.1.2. napájací zdroj

O napájanie všetkých jednotlivých segmentov sa stará spínaný zdroj (AC/DC menič) PS-BOX-12V3A18AH v plechovom boxe (280 x 292 x 80 mm) s možnosťou pripojenia záložného akumulátoru, určený pre použitie v EZS.

Parametre

Napájanie	180 - 260 V~
Výstupné napätie	12 - 14 V=
Maximálny prúdový odber	3 A
Záložný akumulátor	12 V; 2,4 Ah
Detekcia a LED signalizácia	výpadok napájania, preťaženie výstupu
Ochrana vstupu	proti prepätiu
Ochrana výstupu	proti preťaženiu a skratu
Ochrana akumulátora	proti prepólovaniu a vybitiu bod 9,5 V

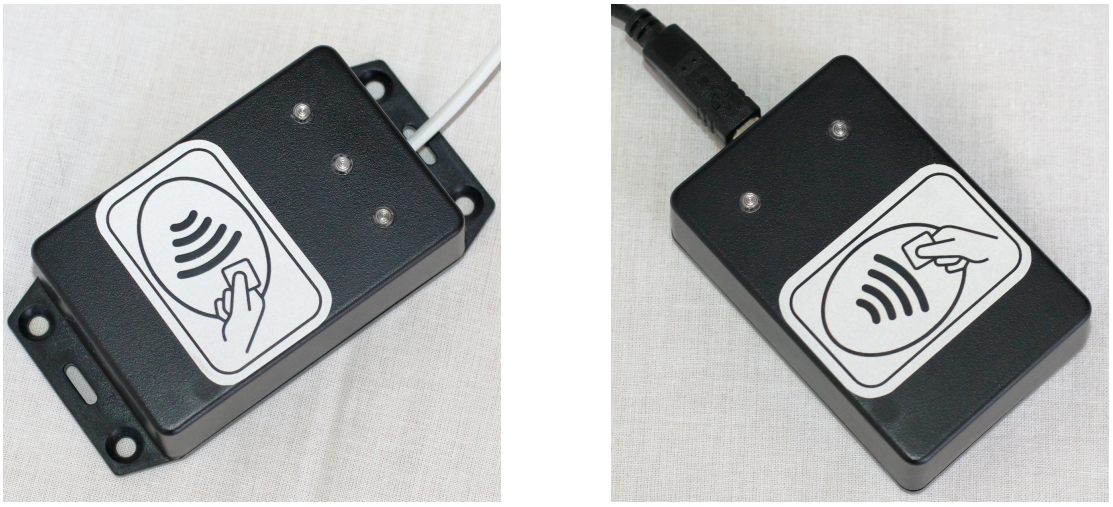
Obrázok 5.3: Napájací zdroj [2]

5.1.3. RFID čítačka

Objektom záujmu tejto práce bol aj návrh dvoch typov RFID čítacích/zapisovacích zariadení. Prvým je modul určený na prichytenie o stenu vedľa dverí, komunikujúci po linke RS-232 s riadiacou jednotkou. Slúži na autorizáciu osôb prostredníctvom Mifare kariet pred vstupom do objektu. Druhý typ RFID modulu komunikuje cez USB s počítačom administrátora, ktorý jednotlivým používateľom vykonáva registráciu ich Mifare kariet.

5.1.4. zámok

Vo dverách do posilňovne je použitý elektromechanický samozamykací zámok *BERA* vybavený funkciou *PANIK,* ktorá funguje tak, že kľučky na oboch stranách dverí sú uchytené na oddelenom štvorhrane, takže stlačenie jednej neovplyvní polohu tej druhej. Kľučka z vnútornej strany je aktívna stále, takže je dodržaná vyhláška č. 23/2008 Sb. zo dňa 29. 1. 2008 *o technických podmínkách požární ochrany staveb*. Zjednodušene povedané, ak slúžia tieto dvere ako núdzový východ, musia byť zkonštruované tak, aby ich bolo možné otvoriť jednou rukou bez použitia kľúča, stlačením kľučky smerom dole. Z vonkajšej strany dverí je zámok odomknuteľný buď použitím kľúča cez cylindrickú vložku alebo kľučkou po privedení elektrického impulzu na elektromagnetickú cievku v mechanizme zámku. Pri zatvorení dverí je zaisťovacia strelka zatlačená o zárubňu, čo vyvolá dvojbodové uzamknutie - automatické vysunutie závory a zablokovanie strelky zámku.



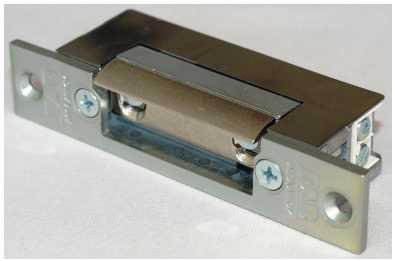
(a) RS-232 (b) USB

Obrázok 5.4: RFID modul

Obrázok 5.5: Elektromechanický samozamykací zámok BERA

5.1.5. dverný otvárač

Dvere do chodby pred posilňovňou sa otvárajú vybudením elektromagnetickej cievky otvárača *BEFO PROFI* umiestneného v zárubni. Pri zamknutí týchto dverí by bol otvárač neúčinný, preto bola zo zámku odstránená závora. Z vonkajšej strany dverí je namiesto kľučky umiestnená guľa. Aby sa dali dvere otvoriť aj kľúčom, bola v zámku zachovaná pôvodná cylindrická vložka.



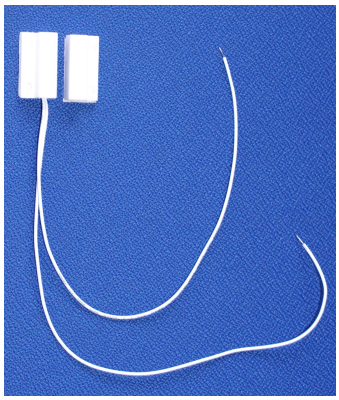
Obrázok 5.6: Elektrický dverný otvárač BEFO PROFI

5.1.6. senzory

Zo softvérového hľadiska je nutné vedieť, či sa v miestnosti ešte niekto zdržiava alebo či už všetci odišli a niekto sa napríklad zabudol pri odchode odhlásiť priložením RFID karty k vnútornej čítačke. Na tieto účely slúži dvojica PIR senzorov zaznamenávajúcich pohyb v miestnosti. Je možné nastaviť intenzitu svetla, od ktorej začína spínať a dobu zotrvania integrovaného relé v zopnutom stave od zaznamenania pohybu.

Pre účely štatistiky je na dvere umiestnený magnetický jazýčkový kontakt, ktorý je pri zatvorených dverách zopnutý a pri otvorení dverí sa naopak rozopne.

(a) PIR



(b) jazýčkový kontakt

Obrázok 5.7: Senzory

5.2. Plošné spoje

V rámci tejto bakalárskej práce bolo potrebné navrhnúť 3 rôzne plošné spoje[1]. Navrhované sú pomocou profesionálneho programu Cadence® OrCAD® 16.6. Zároveň bola od základu navrhnutá prevažná väčšina použitých spájkovacích plôšok a puzdier súčiastok, pretože ako povedal Ing. V. Záhlava „každý robí chyby a tak môžeme veriť iba tomu, čo si sami navrhneme". Jednotlivé plošné spoje boli napanelizované a poslané do výroby ako jeden kus. O výrobu sa postarala pražská firma PragoBoard s.r.o. prostredníctvom ich služby POOL servis. Ide o výrobu dosiek plošných spojov rôznych typov od rôznych zákazníkov, naukladaných na jeden prírez. Tento postup je vhodný pre prototypovú výrobu, pretože zákazníkovi nie je účtovaný poplatok za zhotovenie filmových podkladov, avšak tieto podklady sa nearchivujú a tým pádom sa nedajú v budúcnosti opätovne použiť.

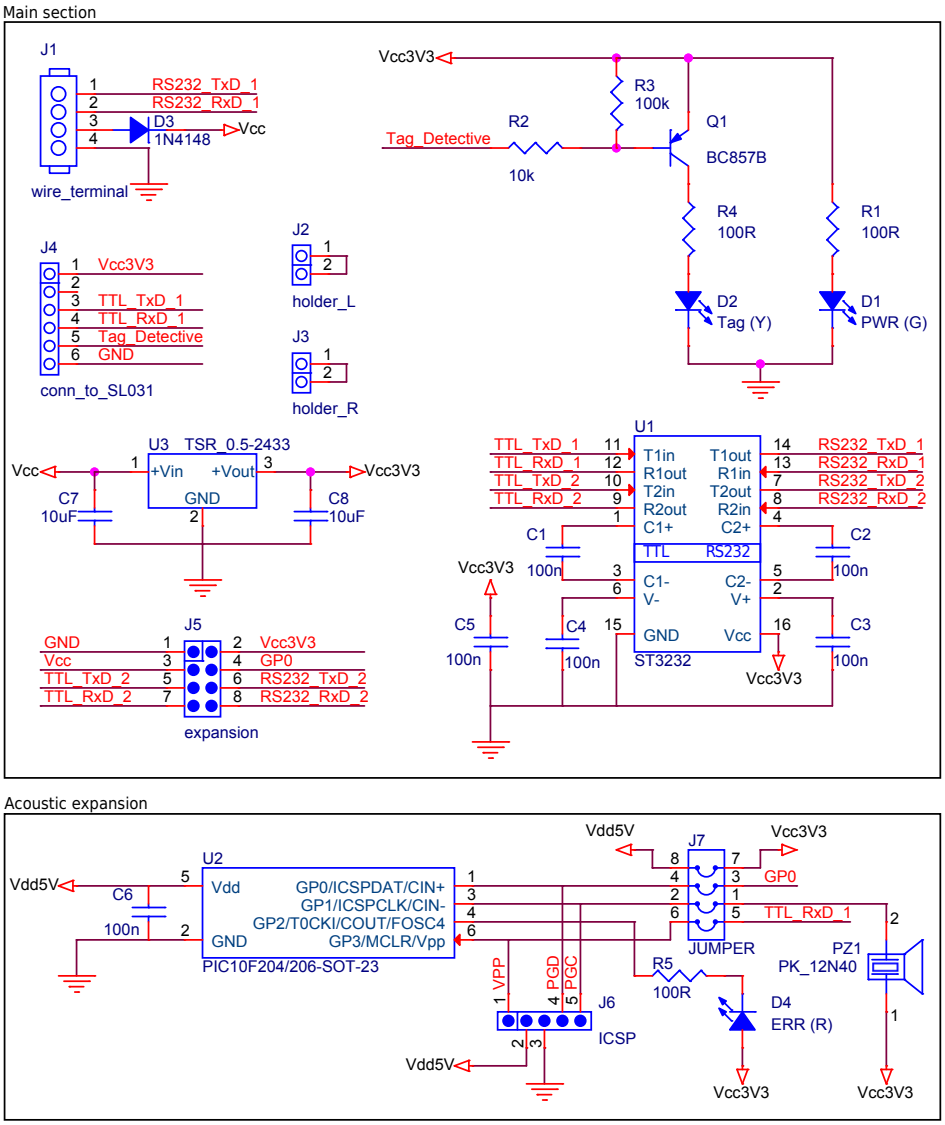
Parametre DPS [18]:

- materiál FR4 1,5 mm, Cu 18 μm
- zelená nepájivá maska
- vrchná servisná potlač
- elektrické testovanie
- povrchová úprava - HASL alebo chemické pozlacovanie
- frézovaný vonkajší obrys (priemer frézy 2,4 mm)
- spoj / medzera ≧ 150 μm, vŕtaná diera ≧ 0,3 mm

[1] Detailnejšie popísané v nasledujúcich podkapitolách

5.3. RFID modul

5.3.1. Popis zapojenia



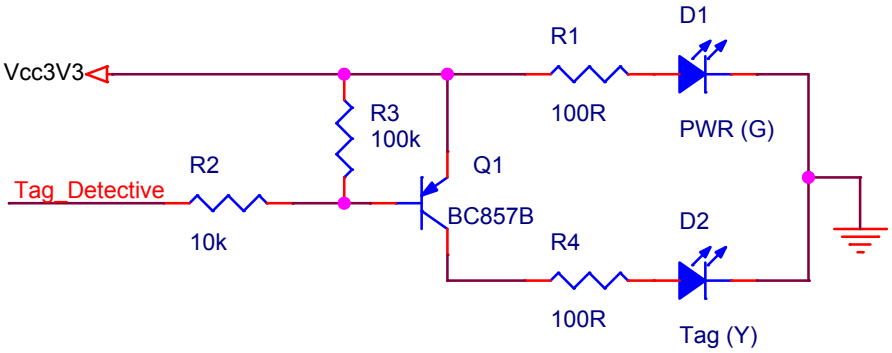
Obrázok 5.8: Schéma RFID modulu

Napájanie

Napájacie napätie zo zdroja 12 V je privedené cez svorkovnicu Wago 218 a diódu chrániacu obvod pred prepólovaním na DC/DC menič TSR 0.5-2433. Niektoré RFID čítačky môžu byť od riadiaceho modulu vzdialené kľudne aj 15 metrov. Aby sa predišlo problémom s nízkym napájacím napätím v dôsledku jeho úbytku na prívodných vodičoch, nie je RFID modul napájaný priamo napätím 3,3 V z riadiaceho modulu, ale je naň privedené napätie 12 V zo zdroja a toto následne DC/DC meničom zmenšené na požadovaných 3,3 V. Najnáchylnejšou súčiastkou obvodu, čo sa požadovaného napájacieho napätia týka, je prevodník RS-232 ↔ TTL, ktorý pre svoju činnosť požaduje minimálne napätie 3 V. Použitý DC/DC prevodník je od firmy TRACO® POWER. Podľa katalógového listu [24] má široký rozsah vstupného na-

pätia, od 4,75 do 32 V. Je pinovo kompatibilný s klasickými stabilizátormi rady LM78xx. Jeho výhodou oproti nim je vysoká účinnosť (až 91%) a z nej vyplývajúce nízke tepelné straty a nepotrebnosť prídavného chladiča. Maximálny výstupný prúd sa pohybuje na úrovni 0,5 A. Na vstupe aj výstupe je opatrený filtračnými kondenzátormi s kapacitou 10 µF (C7, C8).

Signalizácia



Obrázok 5.9: Schéma zapojenia signalizačných LED

Zapnutý stav je signalizovaný zelenou LED diódou pripojenou cez rezistor R1.

Výpočet predradného rezistoru pre LED Použitá LED dióda má úbytok napätia v priepustnom smere U_f = 2,2 V a maximálny priepustný prúd I_f = 30 mA. Napájaná je zo zdroja U = 3,3 V. Hodnota predradného rezistoru R_1 sa vypočíta nasledovne:

$$R_1 = \frac{U - U_f}{I_f} = \frac{3,3 - 2,2}{0,03} \doteq 37 \ [\Omega]$$

Zo štandardnej rezistorovej rady E12 (viď príloha B) by tomu zodpovedal rezistor s hodnotou 39 Ω, avšak, aby nebola táto LED dióda prevádzkovaná na hrane svojich možností, bude vhodné trochu obmedziť prúd a použiť rezistor s hodnotou napr. 100 Ω. Jeho hodnota nie je kritická.

Modul RFID Mifare čítačky/zapisovačky SL031, ktorý sa pripája k obvodu 6-pinovou kolíkovou lištou, podporuje signalizáciu detekcie RFID karty („tagu") nízkou úrovňou na výstupe (pin 5). Podľa katalógu [22], ak nie je v dosahu žiadny tag, tak by mal byť tento výstup v logickej jednotke, avšak meraním bolo zistené, že na ňom v takomto prípade v skutočnosti nie je plné napätie. Výrobca neuvádza konkrétne parametre tohoto výstupu a preto je lepšie posilniť ho tranzistorom, aby nedošlo k jeho nadmernému zaťaženiu v prípade priameho pripojenia LED diódy.

Výpočet hodnôt rezistorov pre spínanie tranzistora Ako spínací prvok je použitý tranzistor typu PNP BC857B. Podľa katalógového listu [15] je saturačné napätie medzi kolektorom a emitorom U_{CEsat} = -75 mV (pri kolektorovom prúde I = -10 mA), čo je zanedbateľná hodnota. Pri výpočte predradného rezistoru nebude uvažovaná a postup bude rovnaký ako v predchádzajúcom prípade. Použitá LED dióda má žltú farbu a podobné parametre: úbytok napätia v priepustnom smere U_f = 2,1 V a maximálny priepustný prúd I_f = 30 mA. Analogicky zvolíme hodnotu predradného rezistoru R_4 = 100 Ω.

Pre výpočet bázového rezistoru R_2 je nutné poznať ešte prúdový zosilňovací činiteľ h_{FE}, ktorý vyjadruje pomer kolektorového prúdu I_C a bázového prúdu I_B, saturačné napätie medzi bázou a emitorom U_{BEsat} a veľkosť kolektorového prúdu I_C. Minimálna udávaná hodnota prúdového zosilňovacieho činiteľa h_{FE} je 220 a U_{BEsat} = -750 mV. I_C sa vypočíta zo zvolenej hodnoty R_4 nasledovne:

$$I_C = -\frac{U - U_f}{R_4} = -\frac{3,3 - 2,1}{100} = -12 \ [mA]$$

Pokiaľ je tranzistor použitý ako spínač, mal by byť bázový prúd I_B podľa [12] aspoň o 30 % vyšší, než aký je potrebný na plné otvorenie tranzistoru.

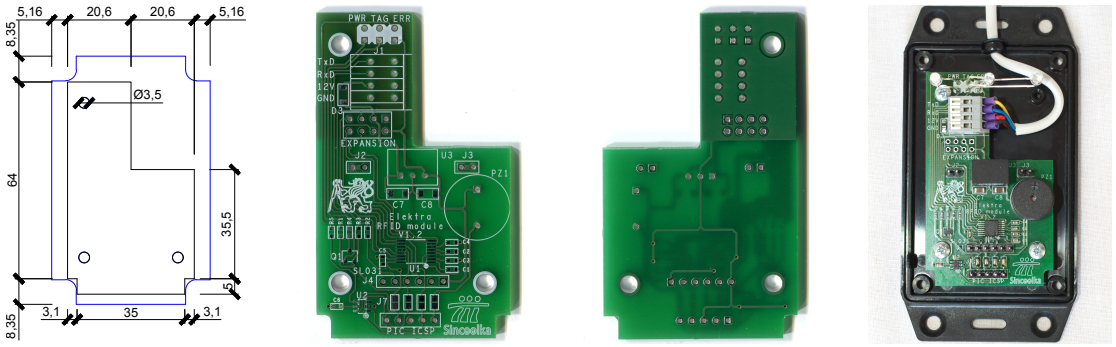
$$I_B = \frac{I_C}{h_{FE}} \cdot 1,3 = \frac{-12\ mA}{220} \cdot 1,3 = -71 \ [\mu A]$$

$$R_2 = \frac{-U - U_{BEsat}}{I_B} = \frac{-3,3 + 0,75}{-71 \cdot 10^{-6}} = 57,04 \ [k\Omega]$$

Hodnota R_2 nie je kritická. Pri voľbe R_2 = 10 kΩ bude tranzistor s určitosťou v saturácii a z výstupu modulu SL031 potečie prúd len približne -0,4 mA. Ako už bolo naznačené, štruktúra ani parametre tohoto výstupu nie sú známe a ak by sa náhodou jednalo o open-collector a výstup by bol v logickej jednotke, nemusel by sa tranzistor celkom uzavrieť. Pre istotu je v obvode „pull-up" rezistor R_3, ktorý zaistí jeho kompletné uzatvorenie. Jeho hodnota je bežne volená ako desaťnásobok hodnoty R_2.

$$R_3 = 10 \cdot R_2 = 100 \ [k\Omega]$$

5.3.2. Doska plošných spojov



(a) rozmery [mm] (porovnanie s vnútorným rozmerom krabičky)

(b) zvrchu

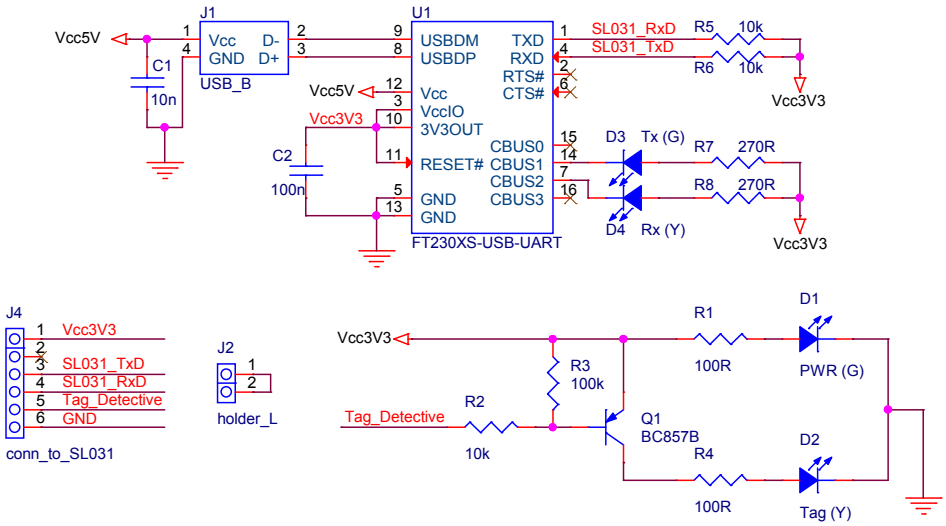
(c) zospodu

(d) po osadení

Obrázok 5.10: RFID modul - DPS

5.4. USB RFID modul

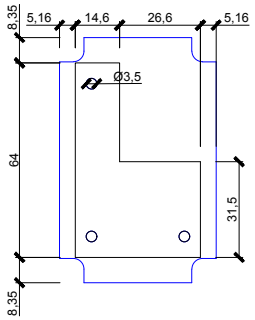
5.4.1. Popis zapojenia



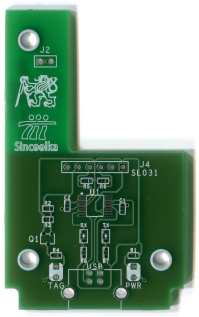
Obrázok 5.11: Schéma RFID USB modulu

Toto zapojenie má narozdiel od 5.3 namiesto prevodníku RS-232 ↔ TTL prevodník USB ↔ UART. Použitý prevodník FT230XS je výhodný z toho dôvodu, že nepotrebuje pre svoj chod žiadne špeciálne ovládače a po pripojení k PC sa tvári ako sériový port, čo zjednodušuje programovú implementáciu. Je použité katalógové zapojenie, kde je celý obvod napájaný priamo z USB portu. Ďalšou výhodou požitého prevodníku je zdroj 3,3 V, ktorý má v sebe implementovaný. Na signalizáciu komunikácie na USB sú použité dve SMD LED diódy. Zapojenie LED diód signalizujúcich zapnutý stav a priložený tag je zhodné s 5.3.

5.4.2. Doska plošných spojov



(a) rozmery [mm] (porovnanie s vnútorným rozmerom krabičky)



(b) zvrchu

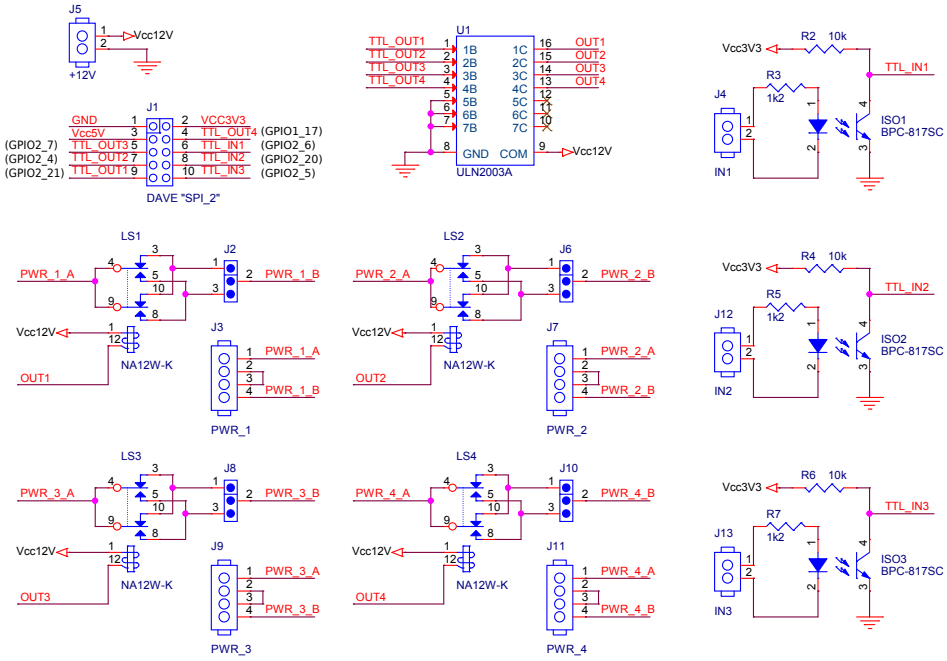
(c) zospodu

(d) po osadení (s modulom SL031)

Obrázok 5.12: RFID USB modul - DPS

5.5. Modul vstupov a výstupov pre KARO TX28

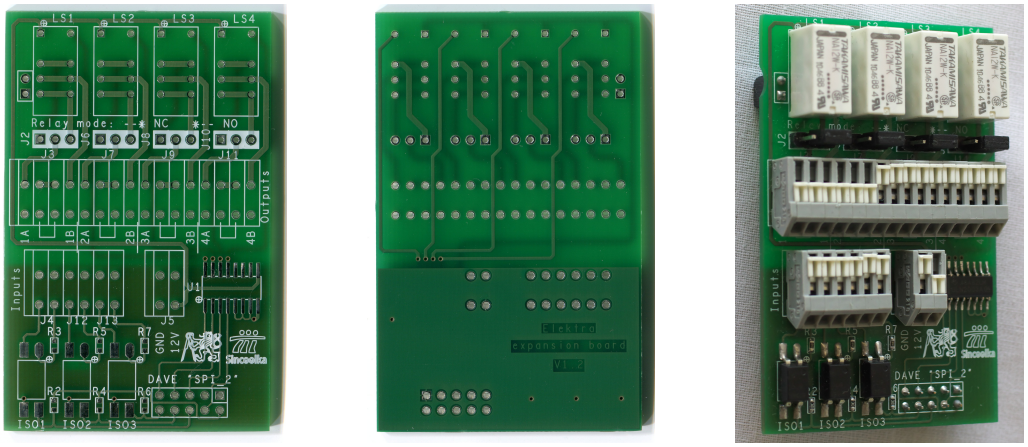
5.5.1. Popis zapojenia



Obrázok 5.13: Schéma RFID USB modulu

Obvod obsahuje 4 výkonové spínacie prvky - relé, a 3 optočleny pre galvanické oddelenie vstupov. Pre spínanie relátok je privedené napájanie 12 V priamo zo zdroja. Na ich spínanie slúži obvod ULN2003A, ktorý v sebe okrem spínacích tranzistorov obsahuje aj ochranné diódy. Predradné rezistory v optočlenoch sú počítané pre napätie 12 V.

5.5.2. Doska plošných spojov



(a) zvrchu (b) zospodu (c) po osadení

Obrázok 5.14: I/O modul - DPS

KAPITOLA 6

SOFTWARE - HLAVNÝ PROGRAM

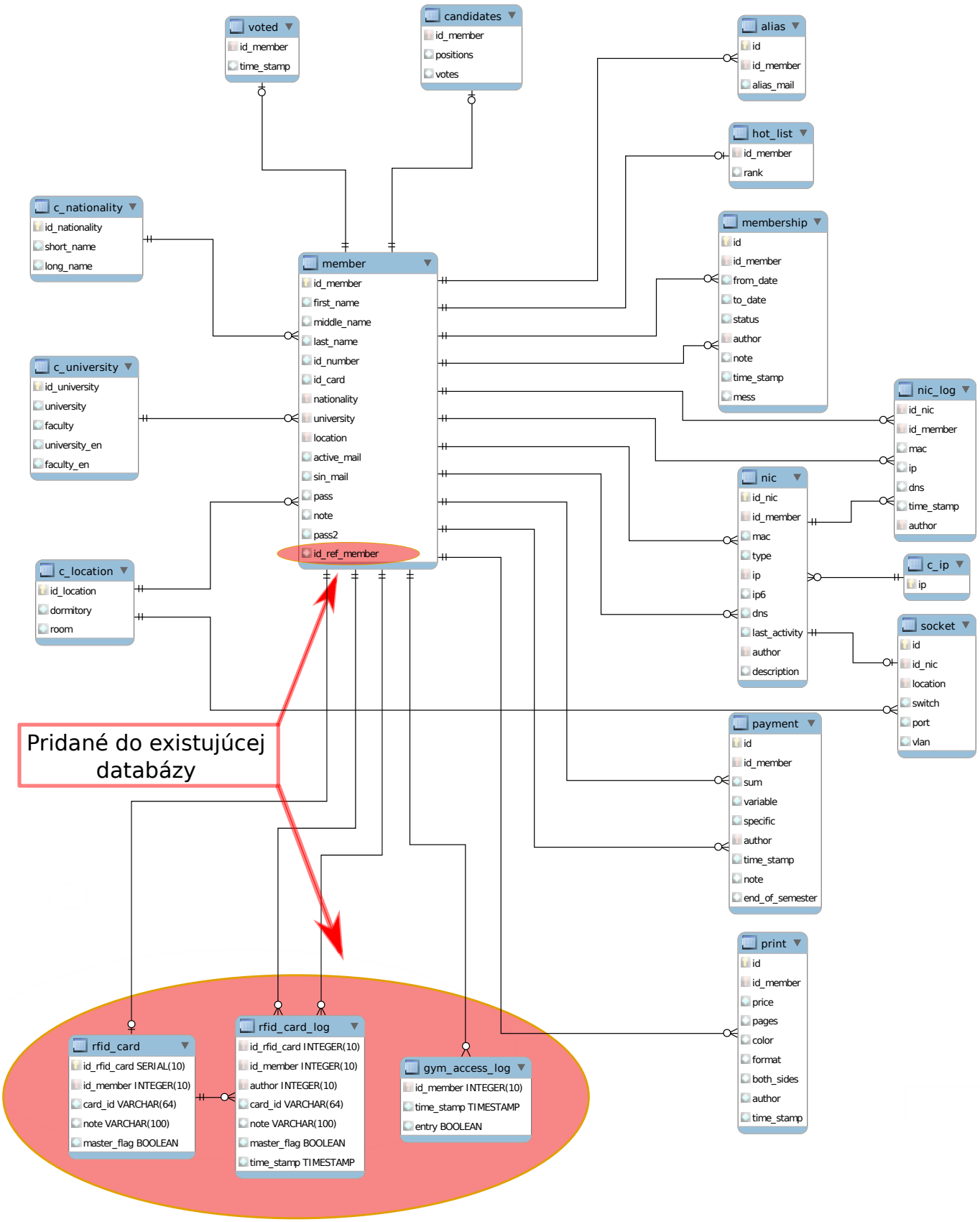
6.1. Databáza

Celý navrhovaný systém nebude fungovať ako samostatná jednotka[1], ale bude integrovaný do väčšieho celku - už existujúceho informačného systému s názvom SINIS. Je to informačný systém klubu Sincoolka, spravujúci používateľské kontá jeho členov. Jeho základom je databáza postavená na PostgreSQL, čo je sofistikovaný a voľne šíriteľný databázový systém. Hlavnou časťou databáze SINIS je tabuľka *member*, v ktorej sú uložené základné identifikačné údaje o každom členovi, ako napr. meno, priezvisko, rodné číslo, národnosť či prístupové heslo. K tejto tabuľke sa pripájajú ďalšie tabuľky s rozširujúcimi údajmi. Najdôležitejšie z nich sú:

- *membership* - dátum platnosti členstva, funkcia člena v klube;
- *payment* - záznamy o zaplatených členských príspevkoch;
- *nic* - informácie o sieťových kartách (typ, MAC adresa, IP adresa, DNS záznam...);
- *print* - pokyny k tlači na klubovej tlačiarni (počet strán, cena, formát...).

Pre potreby projektu *Elektra* bolo potrebné spomínanú databázu rozšíriť. Boli pridané 3 nové tabuľky (*rfid_card*, *rfid_card_log* a *gym_access_log*) a stĺpec do tabuľky *member*.

[1] Aj keď by s drobnými úpravami bez problémov mohol.



Obrázok 6.1: Databáza informačného systému SINIS

6.1.1. Inkorporácia nových prvkov do databáze SINIS

Referenčný používateľ

Aktuálne majú prístup do posilňovne na Sinkuleho koleji iba členovia klubu Sincoolka. Členstvo v klube je určené pre študentov ubytovaných na Sinkuleho a Dejvickej koleji. Ak uchádzač túto podmienku nespĺňa, môže o jeho prijatí do klubu rozhodnúť predseda. Do posilňovne majú aktuálne prístup iba ubytovaní na SDK, pretože iba tí majú preukaz, ktorý si na vrátnici môžu vymeniť za kľúče od posilňovne. Správcovia posilňovne prišli s návrhom[2] umožnenia prístupu do posilňovne aj ľuďom ubytovaným mimo SDK. Z dôvodu zodpovednosti klubu Sincoolka za priestory posilňovne, ktoré má v prenájme od SÚZ, bude toto umožnené iba takým záujemcom mimo SDK, ktorí budú sprevádzaní vopred určeným členom klubu s prístupom do posilňovne. Aby to bolo technicky realizovateľné, budú sa musieť aj takéto osoby zaregistrovať do klubu, zaplatiť členský príspevok a navyše bude v DB do tabuľky *member* pridaný stĺpec *id_ref_member* do ktorého sa uloží referencia práve na toho zvoleného člena ubytovaného na SDK. Pri vstupe do posilňovne sa budú musieť obaja identifikovať svojou RFID kartou.

Príkaz na pridanie stĺpca do DB

```
1 ALTER TABLE member ADD COLUMN id_ref_member INTEGER DEFAULT NULL;
```

Rfid_card

Tabuľka *rfid_card* slúži na ukladanie záznamov o RFID mifare kartách. Každý člen môže mať iba jeden záznam, teda môže mať zaregistrovanú iba jednu kartu.

Rozbor jednotlivých stĺpcov

- id_rfid_card - poradové číslo záznamu
  - s každým novým záznamom automaticky inkrementované
  - unikátne
- id_member - unikátny identifikátor člena
  - povinný záznam
  - unikátne - týmto je zabezpečený maximálne jeden záznam pre každého člena
  - odkaz na tabuľku *member*
- card_id - identifikátor karty, resp. jeho zahashovaná podoba
  - povinný záznam
  - unikátne - to zabezpečí, že jedna karta nebude registrovaná na viac ako jedného člena
- note - poznámka
  - nepovinný záznam
  - slúži na vloženie administrátorovej poznámky, napr. typ karty
- master_flag - príznak nadradenej karty
  - ak je posilňovňa prázdna, treba na vstup dve karty - dve osoby (z dôvodu bezpečnosti); ak je tento príznak „true", stačí na vstup jedna karta
  - napr. pre správcov posilňovne alebo pre upratovaciu službu

[2] Tento návrh vznikol na základe podnetov od samotných cvičencov, ktorí požadovali prístup do posilňovne pre svojich známych, ubytovaných mimo SDK.

Príkaz na pridanie tabuľky do DB

```
CREATE TABLE rfid_card (
    id_rfid_card SERIAL UNIQUE,
    id_member INTEGER NOT NULL UNIQUE REFERENCES member,
    card_id VARCHAR (128) UNIQUE, NOT NULL,
    note VARCHAR (100),
    master_flag BOOLEAN DEFAULT 'false',
    PRIMARY KEY (id_rfid_card),
    CONSTRAINT rfid_card_id_member_fk FOREIGN KEY (id_member)
        REFERENCES member(id_member) MATCH FULL
);
```

Rfid_card_log

Tabuľka *rfid_card_log* slúži, ako už sám názov napovedá, na logovanie všetkých zmien v tabuľke *rfid_card*.

Rozbor stĺpcov odlišných od tabuľky rfid_card

- author - autor nového záznamu/zmeny
  - povinný záznam
  - odkaz na *id_member* z tabuľky *member*
- time_stamp - čas vytvorenia záznamu

Príkaz na pridanie tabuľky do DB

```
CREATE TABLE rfid_card_log (
    id_rfid_card INTEGER NOT NULL,
    id_member INTEGER NOT NULL,
    author INTEGER NOT NULL,
    card_id VARCHAR(128) NOT NULL,
    note VARCHAR(100),
    master_flag BOOLEAN DEFAULT 'false',
    time_stamp TIMESTAMP WITHOUT TIME ZONE DEFAULT NOW(),
    FOREIGN KEY (id_rfid_card) REFERENCES rfid_card(id_rfid_card) MATCH FULL,
    FOREIGN KEY (id_member) REFERENCES member(id_member) MATCH FULL,
    FOREIGN KEY (author) REFERENCES member(id_member) MATCH FULL
);
```

Gym_access_log

Tabuľka *gym_access_log* je určená na zaznamenávanie časov pohybu ľudí do a z posilňovne.

Rozbor stĺpcov tabuľky

- id_member - identifikačné číslo osoby, ktorá otvorila dvere
  - povinný záznam
  - odkaz na *id_member* z tabuľky *member*
- entry - príznak príchodu/odchodu, určuje sa podľa RFID čítačky, ku ktorej bola priložená karta
  - „true” - vstup do posilňovne
  - „false” - odchod z posilňovne
- time_stamp - čas vytvorenia záznamu

Príkaz na pridanie tabuľky do DB

```
CREATE TABLE gym_access_log (
    id_member INTEGER NOT NULL,
    entry BOOLEAN NOT NULL,
    time_stamp TIMESTAMP WITHOUT TIME ZONE DEFAULT NOW(),
    FOREIGN KEY (id_member) REFERENCES member(id_member) match FULL
);
```

6.2. Buildroot

Samotný hardvérový modul počítača Ka-Ro TX28 je bez operačného systému prakticky nepoužiteľný. Inštalácia takéhoto OS je príliš komplikovaná na to, aby bola do detailov rozobraná v tejto práci a z tohto dôvodu budú ďalej rozpísané iba základné princípy.

Jedným zo spôsobov, ako pripraviť všetky potrebné súbory je použiť na to určený buildovací nástroj, schopný predkompilovať celý systémový image, ktorý sa následne len prenesie do hardvérového core modulu. Vhodným adeptom je Buildroot [4] - linuxový nástroj, ktorý zjednodušuje a automatizuje proces zostavovania kompletného linuxového systému pre embedded zariadenia. Je to v podstate balík zdrojových kódov a make scriptov. Princíp je jednoduchý - je potrebné nastaviť všetky požadované parametre, spustiť kompiláciu a potom už len počkať na jej dokončenie. Znie to síce jednoducho, ale na to, aby všetko fungovalo tak ako má, je potrebné venovať konfigurácii značnú pozornosť.

Ďalším spôsobom je použitie ARMSDK-VM [10] priamo od Ka-Ro electronics. Ide o virtuálny stroj založený na Linuxovej distribúcii Debian 6 (Squeeze), spúšťaný na virtualizačnej platforme, akou je napr. VMware Workstation alebo VirtualBox. Slúži na prispôsobenie všetkých častí systému - RedBoot, U-Boot, Linux kernel a root-file-system. V tejto práci však bude použitý Buildroot, testovaný na linuxovej distribúcii Debian 8 (jessie)[3].

Jednou z výhod Buildrootu je fakt, že na vykonanie žiadnej funkcie nie sú potrebné rootovské práva, takže je možné všetko buildovať ako normálny používateľ. Tým je chránený systém pred nechceným chovaním sa balíčkov počas kompilácie.

Pre nastavovanie požadovaných parametrov sa používa nástroj *menuconfig* (viď obrázok 6.2). Pre úpravu parametrov linuxového jadra je dostupný nástroj *linux-menuconfig*. Oba sa spúšťajú z hlavného adresára Buildrootu:

```
make menuconfig
make linux-menuconfig
```

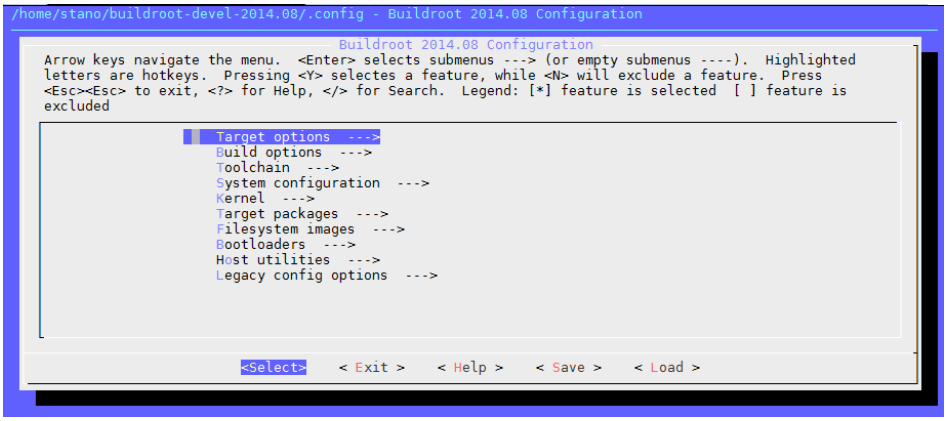
menuconfig - výber programov a služieb, ktoré chceme do systému zakompilovať, nastavenie hostname a hesla pre roota, ssh, knižnice, textový editor...

linux-menuconfig - voľba potrebných ovládačov, zariadení, zberníc...

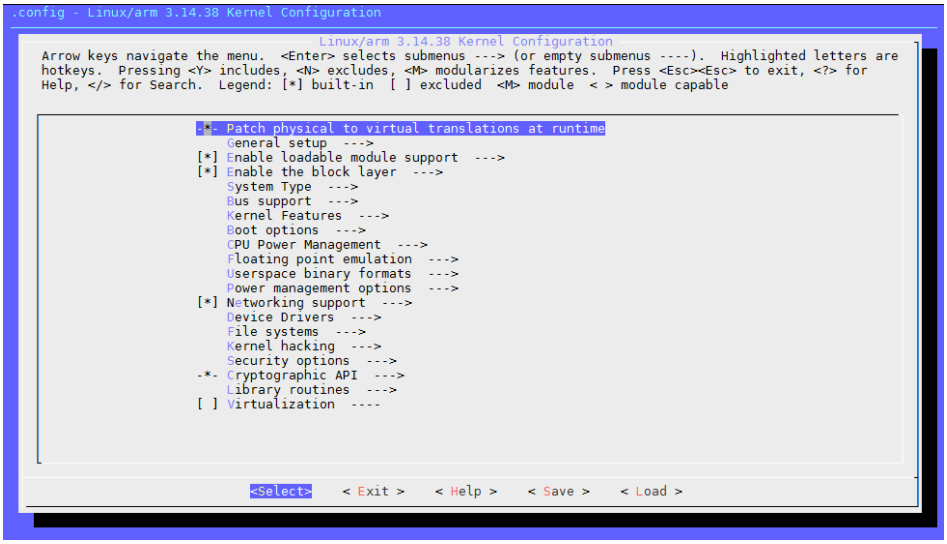
Porovnanie systému vytvoreného Buildrootom a Raspbianu, teda linuxovej distribúcie určenej priamo pre moduly Raspberry Pi, by nebolo úplne adekvátne, pretože Raspbian obsahuje navyše množstvo grafických balíčkov. Pre lepšie porovnanie ale vezmeme do úvahy projekt *Minibian* [21]. Ide o linuxový image vychádzajúci z distribúcie Raspbian, avšak okresaný o GUI a iné nepotrebné nástroje. To ho priamo predurčuje na použitie v embedded zariadeniach akým je aj prístupový systém, ktorým sa zaoberá táto práca.

Kompiluje sa iba to, čo je naozaj nutné, resp. to, čo je požadované. Z toho vyplývajú nasledujúce výhody oproti predpripraveným systémom.

[3] Niektoré použité príkazy uvedené ďalej sa môžu na iných distribúciách nepatrne líšiť.



(a) menuconfig



(b) linux-menuconfig

Obrázok 6.2: Konfiguračné nástroje Buildrootu

Výhody Buildroot vs. Minibian

- rýchlosť - Podľa prednášky Petra Waschingera [25] je takto vytvorený systém pozoruhodne rýchly. Pri použití na Raspberry PI dokáže údajne nabootovať v priebehu 5 sekúnd, čo je iba tretina času potrebného pre nabootovanie systému pri použití Minibianu.
- stabilita - Dôvodom vyššej stability je fakt, že prevažná väčšina potrebných modulov je už zakompilovaná do jadra - potrebné balíčky sa zvolia ešte pred kompiláciou.
- množstvo požadovanej pamäte - Vystačí si bez problémov s menej ako 128 MB pamäte, zatiaľ čo Minibian potrebuje 477 MB.

Zjednodušený postup

1. Stiahnutie Buildroot z internetového zdroja, napr. https://buildroot.org/downloads/buildroot-2016.02.tar.gz
2. Konfigurácia (*make menuconifg*).
   a) Je potrebné mať nainštalovanú knižnicu *ncurses*.

```
1 apt-get install libncurses5-dev
```

3. Stiahnutie balíčkov pre možnosť off-line kompilácie (*make source*).
4. Kompilácia (*make,* alebo pre tvorbu verzií *make RELEASE=1*).
5. Príprava TFTP serveru (napr. atftpd)

```
1     # instalacia
2 apt-get install atftpd
3
4     # konfiguracia
5 # v subore /etc/default/atftpd zmenit USE_INETD=true na USE_INETD=false
6 invoke-rc.d atftpd start
7 mkdir /srv/tftp
8 chmod -R 777 /srv/tftp
9 chown -R nobody /srv/tftp/
10 /etc/init.d/atftpd restart
11
12    # instalacia klienta pre otestovanie funkcnosti
13 apt-get install atftp
```

6. Príprava DHCP serveru (napr. dnsmasq)

```
1     # instalacia
2 apt-get install dnsmasq
3
4     # konfiguracia
5 # v subore /etc/dnsmasq.conf :
6 #   interface=eth0
7 #   dhcp-range=eth0,192.168.0.50,192.168.0.150,12h   interface, rozsah,
      doba zapozicania IP adresy
8 #   dhcp-host=00:0c:c6:7b:33:61,192.168.0.128        priradenie
      konkretnej IP adresy
9 /etc/init.d/dnsmasq restart
```

7. Upload súborov do modulu TX28

6.3. Riadiaci program

6.3.1. Štruktúra súborov a tried

```
Elektra
├── card_registration.py
├── database_manager.py
│   └── class database
├── functions.py
│   ├── class karo_io
│   ├── class mail
│   ├── class db
│   ├── class mifare
│   └── class hash
├── hash_test.py
├── hash_test_karo.py
├── hash_test_MDC2.py
├── main_script.py
└── rest_api-server.py
```

6.3.2. card_registration.py

Slúži na registráciu RFID kariet do systému. Administrátor sa najprv prihlási pomocou svojho prihlasovacieho mena a hesla, čím sa získa jeho ID ktoré je neskôr použité pri záznamoch do logu. Testuje sa, či používateľ, ktorému sa práve registruje karta, nemá nejakú už v systéme, prípadne či aktuálne registrovaná karta už nie je zaregistrovaná pod niektorým z členov.

6.3.3. database_manager.py

Slúži na komunikáciu s REST serverom z ktorého získava dáta, prípadne naň zapisuje. Zatiaľ ide len o pracovnú verziu, ktorá bude čoskoro doplnená. Základný princíp je ale funkčný.

6.3.4. functions.py

Ide o súbor rôznych tried a funkcií využívaný z rôznych častí programu.

class karo_io Ovládanie GPIO portu. Pred každým zápisom/čítaním si skontroluje, v akom stave sa nachádza daný I/O pin a prípadne ho prestaví, čím je zabezpečená prvotná konfigurácia, ale aj odolnosť voči ich manuálnemu, prípadne inému nechcenému prekonfigurovaniu.

class mail Posielanie e-mailov prostredníctvom poštového serveru v privátnej LAN sieti.

class db Obsahuje funkcie na komunikáciu s databázovým serverom v privátnej LAN sieti.

class mifare Súbor rôznych funkcií na posielanie inštrukcií RFID Mifare čítačkám a prijímanie odpovedí. Obsahuje taktiež funkciu na posielanie príkazov mikrokontrolérom v čítačkách pre požadovanú signalizáciu.

class hash Obsahuje funkcie pre hashovanie vstupných dát, vrátane funkcie s názvom *elektra* využívajúcej funkciu *hashlib.pbkdf2_hmac()* rozoberanej v časti 7.3.3.

6.3.5. hash_test*.py

Tieto testy sú rozobrané v časti 7.3.

6.3.6. main_script.py

Hlavný program. Na začiatku sa spustí jeden hlavný proces, ktorý následne spustí ďalší proces pre každú jednu pripojenú RFID čítačku, ktorý s ňou potom autonómne komunikuje.

6.3.7. rest_api-server.py

Serverová aplikácia pre HTTP REST API komunikujúca priamo s databázou. Rovnako ako pri *database_manager.py* ide iba o testovaciu verziu.

KAPITOLA 7

SOFTWARE - OSTATNÉ

7.1. Komunikačný protokol pre RFID reader/writer SL031

Komunikačný protokol modulu SL031 [22] je znakovo (bytovo) orientovaný [16], čo znamená, že pri sériovej komunikácii nie sú jednotlivé bity prenášané samostatne, ale sú združené do slabík s pevným počtom bitov, v tomto prípade 8. Pri vzájomnej komunikácii dvoch (alebo viacerých) zariadení je potrebné zabezpečiť, aby príjemca rozumel formátu prenášaných dát a takisto aby dokázal správne identifikovať jednotlivé časti tohoto dátového bloku. Tento požiadavok je najvýraznejší na najnižšej úrovni, kde sa už neprenášajú jednotlivé bity, ale celé bloky dát. Ide o spojovú vrstvu (data link layer) ISO/OSI modelu, kde sa tieto bloky nazývajú rámce (frames).

Nasledujúce tabuľky demonštrujú formát rámcov pri komunikácii medzi klientom (napr. PC) a RFID modulom SL031:

PC → SL031

	Preambula	Dĺžka	Príkaz	Dáta	Kontrolný súčet
Počet bajtov:	1	1	1	0 - x	1

Preambula	vždy s hodnotou 0xBA (1011 1010)
Dĺžka	vyjadruje počet bajtov od príkazu po kontrolný súčet
Príkaz	hodnota vyjadrujúca požadovanú operáciu, viď tabuľka A.1
Dáta	premenlivá dĺžka závislá na type príkazu
Kontrolný súčet	lineárny XOR kontrolný súčet 7.2.1

SL031 → PC

	Preambula	Dĺžka	Príkaz	Stav	Dáta	Kontrolný súčet
Počet bajtov:	1	1	1	1	0 - x	1

Preambula	vždy s hodnotou 0xBD (1011 1101)
Dĺžka	vyjadruje počet bajtov od príkazu po kontrolný súčet
Príkaz	hodnota vyjadrujúca požadovanú operáciu, viď tabuľka A.1
Stav	reakcia na požadovanú operáciu, viď tabuľka A.2
Dáta	premenlivá dĺžka závislá na type príkazu
Kontrolný súčet	lineárny XOR kontrolný súčet 7.2.1

Na prenos týchto znakov sa využíva UART s nasledujúcimi možnosťami konfigurácie:

inštrukcia [bin]	signalizačné zariadenie
x01x	LED
x10x	bzučiak
x00x	LED + bzučiak

Tabuľka 7.1.: Zoznam povolených príkazov pre audiovizuálnu notifikáciu prostredníctvom PIC10F206

Prenosová rýchlosť	9600 / 19200 / 57600 / 115200 Bd
Počet dátových bitov	8
počet stop bitov	1
Parita	bez parity
Riadenie dátového toku	žiadne

7.2. Komunikačný protokol pre PIC

Mikrokontrolér PIC10F206 použitý v RFID module, určený na zvukovú a vizuálnu interpretáciu rôznych stavov používateľovi, zdieľa sériový komunikačný kanál s SL031. Štandard EIA RS-232 [5] nedefinuje spôsob spojenia viac ako dvoch zariadení[1]. Základná myšlienka tohoto štandardu je postavená na plne duplexnej komunikácii DTE s DCE. Ďalej budeme uvažovať trojvodičovú verziu RS-232 (RxD, TxD a GND). Elektrické špecifikácie tohoto štandardu dovoľujú využívať aj určitú formu zbernicovej komunikácie. Prijímať správy viacerými zariadeniami súčasne nie je problém - stačí paralelne spojiť RxD vodiče. Pri paralelnom spojení TxD vodičov za účelom vysielania viacerých zariadení na zbernicu by pri súčasnom vysielaní dvoch alebo viacerých zariadení nastávali kolízie, ktoré štandard nijako nerieši. Ak sú pre rôzne zariadenia na tejto zbernici určené rôzne správy, je potrebné vyriešiť ich adresovanie na spojovej vrstve.

SL031 využíva pre príjem dát preambulu (0xBA). Kým nedostane takýto rámec, iné prijaté rámce zahadzuje. Pre jednosmerné posielanie inštrukcií z riadiaceho počítača do mikrokontroléru bol navrhnutý obdobný protokol. Ide o 4 bajty dlhý rámec začínajúci preambulou a ukončený kontrolným súčtom.

PC → MCU[2]

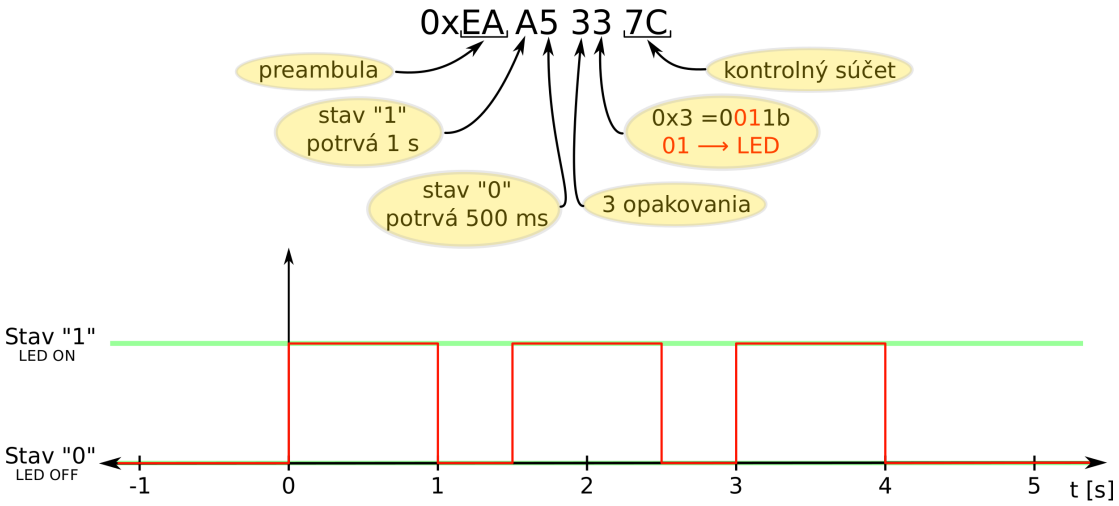
Preambula	stav „1”	stav „0”	Počet periód	Príkaz	Kontrolný súčet
8	4	4	4	4	8

Preambula	vždy s hodnotou 0xEA (1110 1010)
Dĺžka stavu „1”	násobok 100 ms, počas ktorého bude podľa príkazu v jednej perióde daný výstup v stave zapnutom
Dĺžka stavu „0”	násobok 100 ms, počas ktorého bude podľa príkazu v jednej perióde daný výstup v stave vypnutom
Počet periód	počet opakovaní zvoleného režimu
Príkaz	voľba požadovaného signalizačného zariadenia, viď tabuľka 7.1
Kontrolný súčet	lineárny XOR kontrolný súčet 7.2.1

Inštrukcie 7.1 nie sú zvolené náhodne, ale vyplývajú z algoritmu spínania LED a bzučiaku mikrokontrolérom. LED aj bzučiak sú spínané privedením logickej nuly (GND) na I/O pin, kde sú pripojené. Prijatý nibble (štvorica bitov) je presunutý priamo na port GPIO. Na hodnote tretieho a nultého bitu nezáleží,

[1] Nie je definovaný ani v novších revíziách štandardu EIA-232.

[2] Číslo pod každým úsekom nasledujúcej tabuľky vyjadruje jeho veľkosť počtom bitov.



Obrázok 7.1: Príklad interpretácie riadiacej správy pre MCU

pretože piny prislúchajúce týmto bitom sú nastavené ako vstupy a na ich funkciu nemajú vplyv. Príklad jedného rámca správy pre PIC je na obrázku 7.1.

7.2.1. Výpočet kontrolného súčtu

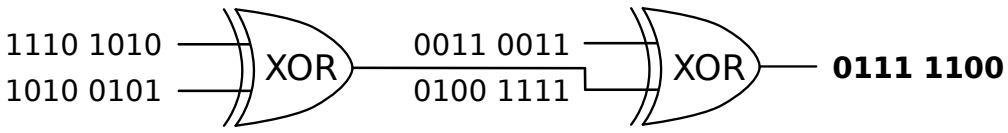
Každý dátový rámec poslaný medzi riadiacim počítačom a RFID modulom je vybavený kontrolným súčtom, ktorý využíva lineárny XOR. Nech **x** a **y** sú binárne premenné a $\oplus$ je operátor XOR. Potom výrok **(x** $\oplus$ **y)** je pravdivý (má pravdivostnú hodnotu 1) vtedy, ak práve jedna z premenných **x**, **y** je pravdivá - viď tabuľka 7.2. Každý bit kontrolného súčtu je výsledkom vzájomného XORovania podmnožiny bitov danej správy s rovnakou váhou.[3] Výsledná hodnota kontrolného súčtu je pridaná na koniec správy. [8]

x	**y**	**x** $\oplus$ **y**
0	0	0
0	1	1
1	0	1
1	1	0

Tabuľka 7.2.: Pravdivostná tabuľka operácie XOR

Príklad výpočtu

Majme správu zloženú z troch bajtov: B_1 = 0xEA, B_2 = 0xA5, B_3 = 0x3.



Obrázok 7.2: Grafické znázornenie výpočtu kontrolného súčtu

[3] Pojem váha je to isté ako určitá mocnina základu Z. [13]

1. Jednotlivé bajty prepíšeme do binárneho tvaru.
   - B_1 = 1110 1010
   - B_2 = 1010 0101
   - B_3 = 0011 0010
2. Vykonáme operáciu XOR s B_1a B_2.

$\oplus$	1110 1010
	1010 0101
=	0100 1111

3. Výsledok predchádzajúcej operácie „vyXORujeme" s ďalším bajtom v poradí.

$\oplus$	0100 1111
	0011 0011
=	0111 1100

4. V prípade väčšieho počtu bajtov opakujeme krok 3 až po B_N, kde **N** je poradové číslo posledného bajtu.
5. Hľadaný kontrolný súčet je výsledok poslednej iterácie - v tomto prípade to je 0111 1100b = 0x7C.

7.3. Hashovanie

Používatelia, resp. osoby oprávnené vstupovať do zabezpečeného objektu, sa autorizujú pomocou unikátneho identifikačného čísla ich RFID karty. Kartou sa rozumie akékoľvek zariadenie vybavené čipom a integrovanou anténou s kondenzátorom, schopné niesť informáciu a komunikovať s čítacím zariadením. Táto karta môže mať rôzne podoby, od štandardnej plastovej karty s rozmermi 8,5 x 5,4 cm, cez kľúčenky až po rôzne nálepky. Tvar puzdra nie je dôležitý, podstatná je elektronika vo vnútri. Každá karta má od výroby priradené unikátne číslo, tzv. UID. Veľkosť UID závisí od typu karty, štandardne to je 4 alebo 7 bajtov. Je zrejmé, že ak je toto UID použité na identifikáciu používateľov, tak s ním treba zaobchádzať ako s citlivým údajom.[4] Ak by sa k týmto údajom dostal nejaký útočník, mohol by si pomocou nich vytvoriť klon danej karty alebo využiť zariadenie na jej simuláciu, čo by mu umožnilo prístup do zabezpečeného objektu. Dobre zabezpečená databáza je základ, ale to stačiť nebude, pokiaľ sa budú heslá posielať sieťou v plain texte, teda v nezmenenej podobe. Útočník by mohol komunikáciu odpočúvať a tieto heslá (v tomto prípade UID) získať. Riešením je aplikovanie takzvanej hashovacej funkcie.[5]

Hashovacia funkcia

Ako uvádza [17], hashovacou funkciou sa obecne rozumie zobrazenie *h*, ktoré priraďuje správe ako vstupu výstup označovaný slovom hash, resp. je to zobrazenie, ktoré reťazcu ľubovolnej dĺžky priraďuje reťazec pevnej dĺžky. Z tejto definície je zrejmé, že existencia kolízií (rôznych vstupov s rovnakým výstupom) je nevyhnutná. Kryptografická hashovacia funkcia má navyše určitú bezpečnostnú vlastnosť práve vo vzťahu k možným kolíziám.

Ideálna hashovacia funkcia sa vyznačuje nasledujúcimi vlastnosťami:

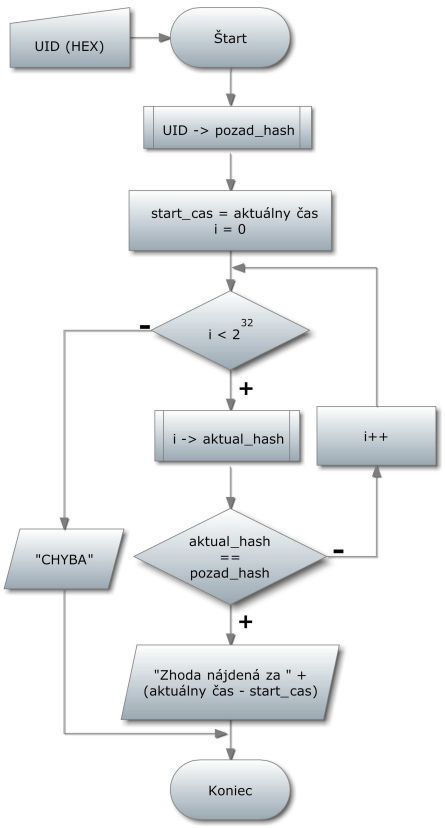
- praktická efektívnosť - pre dané x je výpočet $h(x)$ efektívny,
- mixujúce zobrazenie - pre každý vstup x má výstupná hodnota "náhodný" charakter,
- rezistencia voči kolíziám - z výpočetného hľadiska je neuskutočniteľné nájsť dva vstupy x, y $(x \neq y)$, aby $h(x) = h(y)$,
- rezistencia vzorov - pre danú hodnotu hashu h je výpočetne neuskutočniteľné nájsť vstupný reťazec x tak, že $h = h(x)$.

[4] Rovnako, ako s akýmkoľvek prístupovým heslom.

[5] V slovenčine sa občas používa aj pojem „haš" alebo „hašovacia funkcia".

Je zbytočné pokúšať sa o implementáciu takejto funkcie, keďže už existujú rôzne varianty, ktoré sú mnohými odborníkmi a rokmi praxe overené. Každá je niečím špecifická, ale pre naše potreby bude najpodstatnejšia doba výpočtu. V prípade RFID kariet so 4 bajtovým identifikátorom je reálna možnosť, že by útočník so známeho hashu dokázal zistiť identifikátor karty prostým odskúšaním všetkých možností. Týchto možností je celkom 2^{32}(približne $4{,}3.10^9$), čo by teoreticky nemal byť problém ani pre bežný počítač. Pre ilustráciu bol vykonaný test desiatich rôznych hashovacích funkcií zameraný na odhalenie UID so známeho hashu.

7.3.1. Test hashovacích algoritmov 1



Obrázok 7.3: Vývojový diagram algoritmu na testovanie hashovacích funkcií

Tento test prebiehal na serveri s procesorom Intel® Xeon® E5645 taktovanom na frekvenciu 2,4 GHz s celkovým počtom fyzických jadier 6 a dvomi vláknami na jedno jadro. Tento server je dlhodobo vyťažený pod 2 % s rovnomerným rozložením záťaže na všetky jadrá procesoru, takže výsledky testu by nemali byť skreslené inými procesmi. Pre urýchlenie tohoto testu bol testovací program napísaný ako multiprocesový, a pre každé z UID prebiehalo hashovanie paralelne pre každú zo skúmaných funkcií ako samostatný proces. Každý z týchto procesov tak mohol bežať na inom procesorovom vlákne, čo značne urýchlilo celý test. Tabuľka 7.3 a graf 7.4 znázorňujú porovnanie času potrebného na získanie identifikátoru z jeho známeho hashu pre 10 rôznych hashovacích funkcií, získaného z 11 meraní s rôznym UID, v priemere zodpovedajúcemu UID 0x70E2B0D7 pri použití iteratívneho algoritmu, ktorý postupne prechádzal všetky kombinácie hexadecimálnych čísel v rade 0x00000000, 0x00000001, ... , 0xFFFFFFFF. Skončil pri takom čísle, ktorého hash bol zhodný s hľadaným hashom. Postup je naznačený na obrázku 7.3. Použitý algoritmus má časovú zložitosť O(n), teda lineárnu. V praxi to znamená, že koľkokrát sa zvýši počet vstupných dát, toľkokrát sa zvýši čas potrebný na získanie požadovaného výsledku.

Z grafu 7.4 jasne vyplýva, že pri takto malých[6] vstupných hodnotách sa príliš neprejavujú rozdiely v jednotlivých algoritmoch na dobe potrebnej pre dosiahnutie výsledku. Rozdiel medzi najrýchlejšou a najpomalšou funkciou predstavuje len 29,71 %[7], takže pri výbere tej správnej bude hlavným kritériom jej bezpečnosť, teda náročnosť nájdenia:

1. pôvodnej správy, ktorá zodpovedá jej výstupnému odtlačku,

2. dvoch rozdielnych správ s rovnakým odtlačkom.

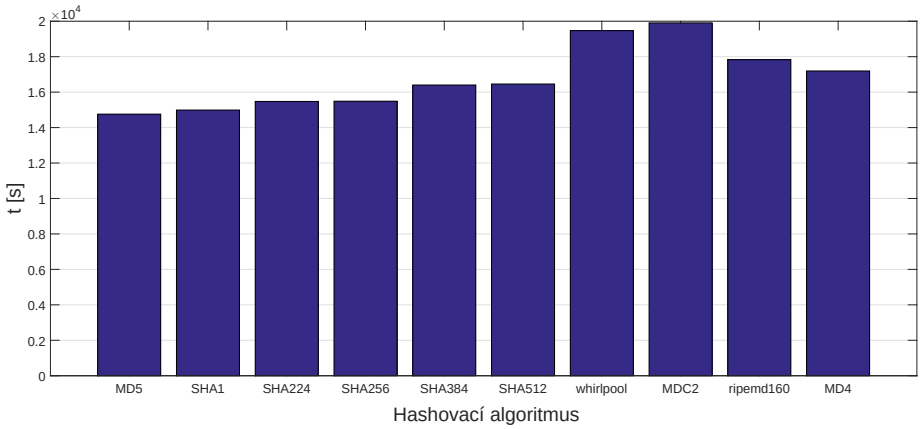
[6] Pri väčších vstupných dátach (jednotky MB) by sa už možno výraznejšie prejavili rozdiely v zložitosti jednotlivých algoritmov, avšak takýto výskum by už bol nad rámec tejto práce.

[7] Postup výpočtu: $\frac{x_2 - x_1}{(x_1 + x_2)/2} \cdot 100\ \%$

	typ hashovacej funkcie				
UID [hex]	**MD5**	**SHA1**	**SHA224**	**SHA256**	**SHA384**
5920410b	3:15:46	3:20:57	3:24:12	3:21:23	3:34:45
bca430ac	6:47:07	6:57:17	7:03:58	7:08:28	7:34:27
0116f258	0:02:15	0:02:35	0:02:20	0:02:32	0:02:47
70e26a85	4:05:41	4:10:39	4:35:02	4:25:40	4:41:05
5c3231ac	3:25:41	3:30:16	3:32:59	3:29:55	3:47:10
3cf33bac	2:14:13	2:15:19	2:24:20	2:21:27	2:29:42
9087c3a5	5:24:47	5:17:23	5:32:15	5:29:18	5:49:37
2aa8c5a5	1:31:25	1:32:48	1:35:32	1:36:46	1:42:20
8ae66a85	4:54:38	5:08:22	5:09:35	5:18:03	5:36:30
72c36985	4:07:40	4:18:48	4:16:19	4:20:57	4:37:02
fffffff	9:16:41	9:13:33	9:40:37	9:44:54	10:11:41
priemer	4:05:59	4:09:49	4:17:55	4:18:08	4:33:22
veľkosť výstupu [Byte]	16	20	28	32	48

	typ hashovacej funkcie				
UID [hex]	**SHA512**	**whirlpool**	**MDC2**	**ripemd160**	**MD4**
5920410b	3:35:19	4:17:24	4:26:22	3:54:38	3:50:29
bca430ac	7:35:36	8:58:12	9:09:39	8:14:04	7:56:53
0116f258	0:02:46	0:03:00	0:03:06	0:02:49	0:02:55
70e26a85	4:39:22	5:36:15	5:45:41	5:06:05	4:56:17
5c3231ac	3:49:03	4:28:57	4:32:53	4:02:53	3:56:45
3cf33bac	2:26:14	2:55:15	2:59:02	2:41:39	2:33:07
9087c3a5	5:58:17	7:02:41	7:11:12	6:26:19	6:07:30
2aa8c5a5	1:43:24	1:59:44	2:05:56	1:50:55	1:45:38
8ae66a85	5:39:56	6:34:15	6:43:37	6:07:54	5:47:21
72c36985	4:37:49	5:29:29	5:38:38	4:58:07	4:55:49
fffffff	10:09:28	12:04:05	12:13:46	11:03:32	10:39:29
priemer	4:34:18	5:24:29	5:31:48	4:57:10	4:46:34
veľkosť výstupu [Byte]	64	64	16	20	16

Tabuľka 7.3.: Porovnanie časov potrebných pre získanie UID zo známeho hashu



Obrázok 7.4: Porovnanie rýchlosti hashovacích funkcií

7.3.2. Test hashovacích algoritmov 2

Pre ukážku bol vykonaný ďalší test, tentokrát zameraný na nájdenie času potrebného na vyskúšanie všetkých vstupných kombinácií pri použití najpomalšej hashovacej funkcie z predchádzajúceho experimentu - MDC2. Program bol spustený na tom istom stroji, s tým rozdielom, že bol testovaný iba jeden hashovací algoritmus a pre zvýšenie efektivity bol rozdelený na 12 procesov.

Skončil s výsledkom 5140 sekúnd, čo je v prepočte približne 1 hodina a 26 minút. Je teda jasné, že ak je možné behom niekoľkých minút odhaliť identifikátoy RFID kariet z ich hashu, tak ich prosté hashovanie nie je dostatočne bezpečné. Z toho dôvodu je potrebné využiť iné metódy pre zvýšenie náročnosti samotného procesu odhalenia vstupného reťazca. Rýchlosť hashovania bola 835,6 kH/s.

7.3.3. Solenie a spomaľovanie

Ako napísal Taylor Hornby [9], keď si myslíte, že ak kryptogafická hashovacioa funkcia spracuje nejaké heslo, bude v bezpečí, ste na omyle. Existuje mnoho spôsobov[8], ako získať pôvodné heslo z jeho prostého hashu celkom rýchlo. Našťastie existuje zopár jednoducho implementovateľných techník, ktoré výrazne znížia efektivitu týchto útokov.

Vyhľadávacie a dúhové tabuľky fungujú iba za predpokladu, že rovnakým vstupom priradí určitá hashovacia funkcia vždy rovnaký obraz. Ak by dvaja používatelia mali náhodou RFID kartu s rovnakým identifikátorom[9], budú aj ich hashe zhodné. Tomuto sa dá zabrániť pridaním soli - náhodne meniaceho sa reťazca ku každému UID pred jeho hashovaním. Nepísaným pravidlom je, že dĺžka soli by mala byť minimálne taká, ako výstup použitej hashovacej funkcie. Soľ by mala byť generovaná kryptografickým pseudonáhodným generátorom, ktorý narozdiel od obyčajného pseudonáhodného generátoru poskytuje vyšší level náhodnosti a nepredvídateľnosti. V Pythone túto funkciu zabezpečuje:

```
1 os.urandom(n)
```

kde n je počet požadovaných bajtov. Bežnou praxou pri autentifikácii používateľov je:

1. identifikácia - používateľ sa predstaví (napr. prihlasovacím menom),
2. autentizácia - overenie identity (napr. heslom).

V takomto prípade sa dá soľ uložiť do databázy spolu s hashom a pri identifikácii znovu načítať. Pri autentizácii sa soľ pridá k zadanému heslo, zahashuje a výsledok sa porovná s hashom z databázy. V našom

[8] Príkladom môže byť slovníkový útok, brute-force útok alebo útok s využitím dúhových tabuliek.
[9] Takáto situácia by teoreticky nikdy nemala nastať.

prípade prebieha autorizácia - iba jeden krok, v ktorom sa overí, či používateľ má prístup na základe UID jeho RFID karty. Problém je v tom, že ak by sme pri registrácii karty uložili soľ k hashu, pri autorizácii by sme nevedeli, z ktorého záznamu v databáze túto soľ vytiahnuť a následne priložiť k UID pre potreby hashovania. Riešením by bolo ukladať soľ priamo na kartu, čo by poskytlo vysokú mieru zabezpečenia, avšak s aktuálne využívaným hardvérom to pri RFID kartách typu Mifare DesFire nie je možné. Pre viac informácií viď 8.1. Použitie konštantnej soli sa neodporúča, avšak ak sa k nej útočník nedostane a získa nejakým spôsobom iba hash, aj to mu dokáže skomplikovať nájdenie UID.

Soľ zabezpečí, že útočník nemôže použiť špecializovaný útok ako napríklad vyhľadávacie či dúhové tabuľky na rýchle odhalenie veľkého súboru hashov, avšak to ich neochráni pred použitím brute-force alebo slovníkového útoku na každý hash samostatne. Výkonné grafické karty alebo špeciálne zariadenia priamo na to určené sú schopné vypočítať miliardy hashov za sekundu. Cieľom je spomaliť hashovanie natoľko, aby bolo pre útočníka použitie hrubej sily príliš zdĺhavé, ale nie až tak veľa, aby to pocítili samotní používatelia. V Pythone na to existuje špeciálna funkcia [19], ktorá vykoná požadované spomalenie závislé na zadanom počte iterácií.

```
1  hashlib.pbkdf2_hmac(hash_name, password, salt, iterations, dklen=None)
2  # priklad
3  >>> import hashlib
4  >>> hash = hashlib.pbkdf2_hmac('sha256', b'0123456789abcd', b'sol-Elektra',
       100000)
5  >>> print ":".join("{:02x}".format(ord(c)) for c in hash)
6  3f:d5:e3:65:3f:2f:24:1b:dc:b4:28:7c:e5:39:42:25:32:b7:40:31:eb:83:8b:ba:4d:4d
       :cb:92:46:c0:1a:61
7  # o jednu iteraciu viac
8  >>> hash = hashlib.pbkdf2_hmac('sha256', b'0123456789abcd', b'sol-Elektra',
       100001)
9  >>> print ":".join("{:02x}".format(ord(c)) for c in hash)
10 8c:33:e5:3a:7f:85:44:c3:72:6f:80:e0:88:a7:a0:ff:03:8a:d6:59:f6:e8:c9:3f:ad
       :06:35:52:0d:36:49:9d
```

Nie je založená iba na obyčajnej iterácii, takže ju nie je možné paralelizovať v hardvéri. Nasledujúci test je zameraný na nájdenie optimálneho počtu iterácií.

7.3.4. Test hashovacích algoritmov 3

Aby bolo možné nejak rozumne nastaviť spomalenie hashovacieho procesu, bola otestovaná jeho rýchlosť na procesore Freescale i.MX28 s frekvenciou 454 MHz použitom v priemyselnom embedded module KARO TX28 za použitia algoritmu SHA512. 10 MH bolo vypočítaných za 4903 s, z čoho vyplýva rýchlosť 2040 H/s. Funkcia *hashlib.pbkdf2_hmac()* má lineárnu časovú náročnosť[10] v závislosti na počte iterácií. Pri testovaní 10 miliónov iterácií vyšla rýchlosť pokusom na 20833 iterácií za sekundu. Odporúčané spomalenie je asi pol sekundy, takže počet iterácií bude nastavený na 10000. Pri tejto konfigurácii pomocou funkcie *hashlib.pbkdf2_hmac()* trvalo 10000 hashov 8438 s, z čoho vyplýva rýchlosť 1,19 H/s, takže došlo k 1721 násobnému spomaleniu. Ak tento výsledok premietneme na test v podkapitole 7.3.2, tak by pomocou brute-force útoku vyskúšal všetky možnosti nie za hodinu a pol, ale približne za 3 mesiace, čo je podstatný rozdiel. Samozrejme, že to sú iba odhady a treba ich brať s určitou rezervou, ale pre predstavu by mali stačiť.

Ak uvážime fakt, že je použitá aj soľ a prevažná väčšina[11] používaných mifare kariet budú typu DesFire so 7 bajtov dlhým identifikátorom[12], dá sa hovoriť o celkom slušnom zabezpečení v porovnaní s ukladaním UID do databáze v originálnej podobe.

[10] Tento fakt bol overený.

[11] Ak nie všetky.

[12] V takom prípade vzrastá počet možných kombinácií UID vzhľadom k testovanému 4-bajtovému UID takmer 17-milión-násobne.

KAPITOLA 8

PLÁNY DO BUDÚCNOSTI

Ako už bolo naznačené v úvode, táto kapitola pojednáva o možnostiach ďalšieho vylepšovania tejto práce. Sú tu zhrnuté nápady autora, ktoré už nebolo možné z určitých dôvodov zrealizovať.

8.1. Zápis dát na mifare karty

Prvotná myšlienka identifikácie používateľov bola postavená na základe unikátneho čísla ich RFID kariet. Z bezpečnostných dôvodov nie sú tieto identifikátory ukladané do databáze priamo, ale prechádzajú hashovacím procesom. Pre zvýšenie bezpečnosti by bol každej RFID karte priradený reťazec náhodných znakov - soľ, ktorý by bol uložený na karte v zabezpečenom sektore. Tento reťazec znakov by následne vstupoval do hashovacieho procesu spolu s UID danej mifare karty (viď 7.3.3), čo by znemožnilo prípadnému útočníkovi použitie slovníkového alebo brute-force útoku na nájdenie identifikátora karty z jeho obrazu.

Problém je v tom, že aktuálne využívaná RFID čítačka/zapisovačka SL031 dokáže čítať/zapisovať údaje z/na Mifare 1k, Mifare 4k aj Mifare UltraLight, avšak z Mifare DesFire dokáže prečítať iba UID. Študentské karty ISIC sú práve typu Mifare DesFire a prevažne tieto karty budú využívané v systéme *Elektra*, preto je aktuálne zbytočné implementovať použitie soli.

Riešením je použiť iný typ RFID čítacieho/zapisovacieho modulu. Príkladom môže byť SL060, taktiež od firmy StrongLink ako aktuálne používaný modul SL031. Tieto moduly zú navzájom pinovo kompatibilné, s tým rozdielom, že SL060 nevyužíva piny 2 a 5 (IN a Out), čo v tomto prípade znamená, že by sa pri ich zámene nerozsvietila signalizačná LED pri priložení karty.

8.2. Snímanie a zber obrázkov z kamier

V priestore posilňovne sa nachádzajú 2 IP kamery a ďalšia je pred jej vstupom. Tieto kamery by sa dali využiť na zachytávanie obrázkov návštevníkov pri otvorení dverí (z vonkajšej aj z vnútornej strany). Z vnútornej strany dverí je kľučka, vďaka panikovej funkcii, aktívna non-stop, a preto sa dajú dvere otvoriť aj bez priloženia karty k čítačke. Vďaka senzoru na dverách sa dá táto udalosť detekovať aj pri odchode z objektu. Zachytené fotografie spolu so zaznamenaným časom otvorenia dverí a prípadne aj identifikačným číslom vlastníka použitej RFID karty by mohli v prípade krádeže napomôcť k jednoduchšej identifikácii páchateľa.

8.3. Štatistika návštevnosti

Na webových stránkach klubu, v časti určenej pre posilňovňu, by mohli byť informácie o aktuálnom počte návštevníkov a rôzne štatistiky za predošlé obdobia. Mohol by sa viesť rebríček členov s najdlhším časom stráveným v posilňovni.

8.4. Posielanie denných štatistík e-mailom

Posielanie denných štatistík e-mailom správcom posilňovne, prípadne administrátorom, je najjednoduchšie splniteľný cieľ. Systém je na to už takmer pripravený. Tieto správy by sa posielali vždy na konci dňa. Ich obsahom by bol napr. počet návštevníkov za deň, počet otvorení dverí, menovitý zoznam členov a zoznam tých, ktorý sa pri odchode neodhlásili. Presný obsah týchto správ bude potrebné detailne prekonzultovať so správcami posilňovne.

8.5. Úprava programu

Ďalším pomerne jednoducho splniteľným cieľom je úprava programu spočívajúca v sprehľadnení kódu a lepšom využití dostupných funkcií. Základným predpokladom pre tieto úkony je hlbšia znalosť jazyka Python. Hlavnou časťou tohoto cieľa je návrh štruktúry konfiguračného súboru a napísanie funkcie pre jeho parsovanie. Toto riešenie by bolo omnoho elegantnejšie ako súčasná konfigurácia využívajúca konštanty definované na začiatku súboru.

KAPITOLA 9

ZÁVER

Hneď na začiatku práce sme si stanovili za cieľ navrhnúť systém s využitím platformy ARM9, ktorý by sa dokázal svojimi parametrami a stabilitou radiť medzi profesionálne priemyselné embedded zariadenia a zároveň sa svojou cenou zaradiť do kategórie low-cost počítačových embedded modulov, akými je napríklad BeagleBoard, MicroZed či Pandaboard, ktoré však nie sú dimenzované do náročných podmienok, kde sú kladené vysoké nároky na ich funkcionalitu.

Na začiatku boli preskúmané hardvérové možnosti procesora Freescale i.MX28 ako takého a následne sme preštudovali vlastnosti modulu Ka-Ro TX28, ktorý využíva spomínaný procesor. Porovnaním tohto modulu s populárnym Raspberry Pi a BeagleBone sme sa uistili o vhodnosti jeho využitia v plnohodnotných embedded zariadeniach. Vynikal medzi nimi svojimi rozmermi a hmotnosťou, ale aj pracovným teplotným rozsahom či počtom rozhraní, ktoré sú v priemyselných aplikáciách také dôležité. Na demonštráciu jeho využiteľnosti sme si zvolili za cieľ navrhnúť prístupový systém, ktorý bude v budúcnosti inštalovaný do priestorov posilňovne Sinkuleho koleje ČVUT.

Ďalším krokom bolo zostavenie systému nazvaného Elektra zo samostatných blokov tak, aby spolu dokázali plniť požadovanú funkciu. Z bezpečnostných dôvodov bol zvolený zálohovaný napájací zdroj a elektromechanický zámok s panikovou funkciou, ktorý umožňuje bezproblémové odomknutie dverí z vnútornej strany v každej situácii. Pre detekciu pohybu v objekte bola použitá dvojica PIR senzorov a jeden jazýčkový magnetický senzor na vstupné dvere. Venovali sme sa taktiež návrhu hardvérových modulov pre RFID komunikáciu s Mifare kartami dvoch typov - pre identifikáciu osôb pred vstupom do objektu, komunikujúcich po RS-232 a USB RFID modulov pre administratívne účely, a modulu vstupov a výstupov pre ovládanie zámkov a zber dát zo senzorov. Plošné spoje pre tieto moduly boli vyrobené na profesionálnej úrovni firmou PragoBoard.

Po zostavení hardvérovej časti systému prišiel na rad softvér. Najprv išlo o oživenie modulu TX28 pomocou linuxového systému vygenerovaného Buildrootom. Potom bola rozšírená databáza stávajúceho informačného systému klubu Sincoolka, starajúceho sa o počítačovú infraštruktúru v priestoroch, kam bude Elektra nainštalovaná. Na riadenie systému bol napísaný program v jazyku Python, pozostávajúci z viacerých súborov a tried, slúžiaci na riadenie komunikácie s RFID čítačkami, komunikáciu s databázovým serverom prostredníctvom HTTP REST API, či posielanie e-mailov. Našim ďalším cieľom je prehĺbiť si znalosti z programovacieho jazyka Python a následne vylepšiť riadiaci program napísaný počas tejto bakalárskej práce. Pre mikrokontrolér PIC v moduloch RFID čítačiek bol napísaný obslužný program v jazyku symbolických adries, implementujúci nami navrhnutý protokol pre príjem riadiacich správ prostredníctvom softvérového UARTu. Počas práce sa z hľadiska zvýšenia bezpečnosti vyskytla potreba analýzy možných spôsobov hashovania, preto bola preskúmaná aj táto oblasť napriek jej absencii v primárnych cieľoch práce.

Systém sa javil veľmi stabilne, počas testovania a ladenia programu v časovom horizonte troch mesiacov nenastali žiadne komplikácie a neboli detekované ani žiadne výpadky jednotlivých funkcií. Bol vykonaný

test zameraný na výdrž batérie. Simulácia plnej prevádzky prebiehala tak, že sa striedavo spínala cievka elektromechanického zámku a dverného otvárača na dobu 5 s periódou 150 sekúnd. Batéria vydržala napájať obvod vyše 7,5 h, čo je viac než dostačujúce na pokrytie bežných výpadkov dodávky elektrickej energie. V pohotovostnom režime (zapnutý stav celého systému bez zopnutých zámkov) vydrží batéria niekoľkonásobne dlhšie - podľa testov viac ako jeden deň. Dôkladnejšiu analýzu platformy bude možné vykonať, ako sme na začiatku predpokladali, až po zavedení systému do ostrej prevádzky od zimného semestra 2016.

Platforma ARM9 sa ukázala ako veľmi schopné a vhodné riešenie pre nasadenie vo vývoji plnohodnotných embedded zariadení. Jej výhodou je predovšetkým priaznivý pomer cena/výkon. Prednosťou použitého minipočítača TX28 je jeho kompatibilita s ďalšími modulmi zo série TX, takže tu je možnosť jeho jednoduchej výmeny v pätici DIMM200 za výkonnejší bez nutnosti zásahu do DPS.

Odovzdaním a prezentovaním bakalárskej práce tento projekt rozhodne nekončí, ba práve naopak, uvedením systému do prevádzky sa ešte len začnú odhaľovať jeho ďalšie možnosti. Platforma ARM9 má v praxi široké uplatnenie, tak prečo to nevyužiť?

POUŽITÁ LITERATÚRA

[1] *RPi Hardware* [online]. 2016. [cit. 21. 5. 2016]. Dostupné z: http://elinux.org/RPi_Hardware.

[2] AH ELECTRONICS S.R.O. *PS-BOX-13V3A18Ah, zálohovaný zdroj v boxu* [online]. [cit. 23. 5. 2016]. Dostupné z: http://www.ahel.cz/images/produkty/foto_large_3239_fullsize.jpg.

[3] BEAGLEBOARD.ORG. *BeagleBone Rev A5 System Reference Manual*, 2012. Dostupné z: https://github.com/CircuitCo/BeagleBone-RevA5/raw/master/BeagleBone_revA5_SRM.pdf.

[4] BUILDROOT DEVELOPERS. *The Buildroot user manual*, 2016. Dostupné z: https://buildroot.org/downloads/manual/manual.pdf.

[5] ELECTRONICS INDUSTRIES ASSOCIATION. *EIA standard RS-232-C Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*. International Organization for Standardization, Geneva, Switzerland, 1969.

[6] FREESCALE SEMICONDUCTOR. *i.MX28 Applications Processors for Consumer Products*, 2012. Dostupné z: http://cache.freescale.com/files/32bit/doc/data_sheet/IMX28CEC.pdf.

[7] FREESCALE SEMICONDUCTOR. *i.MX28 Applications Processor Reference Manual*, 2010. Dostupné z: http://free-electrons.com/~maxime/pub/datasheet/MCIMX28RM.pdf.

[8] GAURAVARAM, P. – KELSEY, J. Linear-XOR and additive checksums don't protect Damgård-Merkle hashes from generic attacks. In *Topics in Cryptology–CT-RSA 2008*. Springer, 2008. s. 36–51.

[9] HORNBY, T. *Salted Password Hashing - Doing it Right* [online]. 2016. [cit. 14. 5. 2016]. Dostupné z: https://crackstation.net/hashing-security.htm.

[10] KA-RO ELECTRONICS. *ARMSDK-VM Virtual Appliance A preconfigured Linux system*, 2011. Dostupné z: https://www.karo-electronics.com/fileadmin/download/ARMSDK-VM%20-%20A%20Preconfigured%20Debian%206%20%28Squeeze%29%20VMware%20virtual%20machine/ARMSDK-VM-Guide-Debian_6.0_Squeeze.pdf.

[11] KA-RO ELECTRONICS. *TX28*, 2016. Dostupné z: https://www.karo-electronics.com/fileadmin/download/Datasheets/TX28-Datasheet.pdf.

[12] MARTELL, M. *Using Bipolar Transistors As Switches* [online]. 2009. [cit. 15. 5. 2016]. Dostupné z: http://www.rason.org/Projects/transwit/transwit.htm.

[13] MATOUšEK, D. *Číslicová technika - základy konstruktérské praxe*. Praha : BEN-Technická literatura, 2001. ISBN 80-7300-025-3.

[14] MILTON, H. J. *The Selection of Preferred Metric Values for Design and Construction*. U. S. DEPARTMENT OF COMMERCE / National Bureau of Standards, 1978. Dostupné z: https://www.gpo.gov/fdsys/pkg/GOVPUB-C13-f5fea679df4c3a1c2e3e1dd63488707c/pdf/GOVPUB-C13-f5fea679df4c3a1c2e3e1dd63488707c.pdf.

[15] NXP SEMICONDUCTORS. *BC856; BC857; BC858 PNP general purpose transistors*, 2004. Dostupné z: http://www.nxp.com/documents/data_sheet/BC856_BC857_BC858.pdf.

[16] PETERKA, J. Znakově orientovaný, bitově orientovaný protokol. *CHIPweek*. 1995, 29. Dostupné z: http://www.earchiv.cz/a95/a529k130.php3.

[17] PINKAVA, J. Hashovací funkce v roce 2004. *Crypto-World č*. 2004, 9, s. 15–18.

[18] PRAGOBOARD S.R.O. *POOL servis* [online]. 2012. [cit. 29. 4. 2016]. Dostupné z: https://www.pragoboard.cz/pool_servis.

[19] PYTHON SOFTWARE FOUNDATION. *hashlib – Secure hashes and message digests* [online]. 2016. [cit. 21. 5. 2016]. Dostupné z: https://docs.python.org/3/library/hashlib.html.

[20] RADETZKI, M. *Languages for Embedded Systems and Their Applications: Selected Contributions on Specification, Design, and Verification from FDL'08*. 36. Springer Science & Business Media, 2009. ISBN 978-1-4020-9713-3.

[21] SOLTOGGIO, L. *MINIBIAN: MINImal raspBIAN image for Raspberry Pi* [online]. 2016. [cit. 21. 5. 2016]. Dostupné z: https://minibianpi.wordpress.com.

[22] STRONGLINK. *Mifare Reader/Writer SL031 User Manual*, 2015. Dostupné z: http://www.stronglink-rfid.com/download/SL031-User-Manual.pdf.

[23] STRONGLINK. *NFC Reader/Writer Module SL060 User Manual*, 2015. Dostupné z: http://www.stronglink-rfid.com/download/SL060_User_Manual.pdf.

[24] TRACO POWER. *DC/DC Converters TSR 0.5 Series, 0.5 A Switching Regulator*, 2014. Dostupné z: http://www.tracopower.com/products/tsr05.pdf.

[25] WASCHINGER, P. Buildroot, jak na to... InstallFest, 2015.

Prílohy

PRÍLOHA A

SÚHRN PRÍKAZOV A STAVOVÝCH KÓDOV PRE SL031

Príkaz	Popis
0x01	Select Mifare card
0x02	Login to a sector
0x03	Read a data block
0x04	Write a data block
0x05	Read a value block
0x06	Initialize a value block
0x07	Write master key (key A)
0x08	Increment value
0x09	Decrement value
0x0A	Copy value
0x10	Read a data page (UltraLight & NTAG203)
0x11	Write a data page (UltraLight & NTAG203)
0x12	Download Key
0x13	Login sector via stored Key
0x50	Go to Power Down mode
0xF0	Get firmware version

Tabuľka A.1.: SL031 - prehľad príkazov

Status	Popis
0x00	Operation succeed
0x01	No tag
0x02	Login succeed
0x03	Login fail
0x04	Read fail
0x05	Write fail
0x06	Unable to read after write
0x08	Address overflow
0x09	Download Key fail
0x0D	Not authenticate
0x0E	Not a value block
0xF0	Checksum error
0xF1	Command code error

Tabuľka A.2.: SL031 - prehľad stavových kódov

PRÍLOHA B

RAD REZISTOROV E12

Rezistory, ako aj iné pasívne súčiastky, je nutné vyrábať s veľkým rozsahom hodnôt tak, aby bola každá dekáda pokrytá rovnomerne so zaručenou toleranciou. Ako sa píše v [14], ako prvý upozornil na potrebu systematického pokrytia určitého rozsahu čo najmenším počtom prvkov francúzsky inžinier Colonel Charles Renard (1849 - 1905), v rokoch 1877 až 1879, keď študoval prvky používané pri konštrukcii balónov. Pri jeho výskume zistil, že sa pre tieto účely používa 425 rôznych druhov lán. Pre zredukovanie tohto počtu vynašiel veľkostný systém založený na geometrickom rade, vďaka čomu sa podarilo pokryť celý rozsah len pomocou 17 druhov lán.

Renardov systém bol založený na sérii čísel zvolených tak, že každý piaty krok zvýšil veľkosť desaťnásobne:

$$a \cdot g^5 = 10a \quad \rightarrow \quad g = \sqrt[5]{10}$$

a tak prišiel k nasledujúcej sérii vzťahov:

$$a, \quad a\sqrt[5]{10}, \quad a\left(\sqrt[5]{10}\right)^2, \quad a\left(\sqrt[5]{10}\right)^3, \quad a\left(\sqrt[5]{10}\right)^4, \quad 10a$$

Po vyjadrení, zaokrúhlení koeficientov na prakticky použiteľné hodnoty a dosadení čísla 10 za *a,* získal tento rad:

$$10, \quad 16, \quad 25, \quad 40, \quad 63, \quad 100$$

ktorý môže pokračovať oboma smermi. Pomer susedných prvkov je približne 1,6, čo znamená, že každá hodnota je približne o 60 % vyššia ako tá pred ňou. Renardov systém sa používa aj v elektrotechnike v rôznych podobách ako E*x*, kde *x* vyjadruje počet intervalov na dekádu.

Jedným z najpoužívanejších rezistorových radov je E12. Ako už bolo naznačené vyššie, ide o geometrický rad, v ktorom je podiel dvoch po sebe nasledujúcich čísel približne konštantný. Tolerancia hodnôt sa pohybuje v rozmedzí ± 10 %.

1 Ω	10 Ω	100 Ω	1 kΩ	10 kΩ	100 kΩ	1 MΩ	10 MΩ
1,2 Ω	12 Ω	120 Ω	1,2 kΩ	12 kΩ	120 kΩ	1,2 MΩ	
1,5 Ω	15 Ω	150 Ω	1,5 kΩ	15 kΩ	150 kΩ	1,5 MΩ	
1,8 Ω	18 Ω	180 Ω	1,8 kΩ	18 kΩ	180 kΩ	1,8 MΩ	
2,2 Ω	22 Ω	220 Ω	2,2 kΩ	22 kΩ	220 kΩ	2,2 MΩ	
2,7 Ω	27 Ω	270 Ω	2,7 kΩ	27 kΩ	270 kΩ	2,7 MΩ	
3,3 Ω	33 Ω	330 Ω	3,3 kΩ	33 kΩ	330 kΩ	3,3 MΩ	
3,9 Ω	39 Ω	390 Ω	3,9 kΩ	39 kΩ	390 kΩ	3,9 MΩ	
4,7 Ω	47 Ω	470 Ω	4,7 kΩ	47 kΩ	470 kΩ	4,7 MΩ	
5,6 Ω	56 Ω	560 Ω	5,6 kΩ	56 kΩ	560 kΩ	5,6 MΩ	
6,8 Ω	68 Ω	680 Ω	6,8 kΩ	68 kΩ	680 kΩ	6,8 MΩ	
8,2 Ω	82 Ω	820 Ω	8,2 kΩ	82 kΩ	820 kΩ	8,2 MΩ	

Tabuľka B.1.: Rad rezistorov E12

PRÍLOHA C

NÁKLADY NA PROJEKT

V tejto prílohe sú zhrnuté ceny jednotlivých komponentov projektu *Elektra*. Nemá zmysel rozpisovať ceny do detailov za každú súčiastku, preto sú niektoré zahrnuté do väčšieho celku, napr. položka *RFID modul* v sebe zahŕňa cenu plošného spoja, krabičky, modulu SL031 a jednotlivých súčiastok.

Projekt financovala *Studentská unie ČVUT*.

položka	cena/ks [Kč]	ks	cena celkovo [Kč]
napájací zdroj PS-BOX 12V3A18Ah	1587	1	1587
bezúdržbový akumulátor 12 V 2,4 Ah	374	1	374
elektromachanický zámok BERA	7107	1	7107
kovanie HOPPE	734	1	734
delený štvorhran	249	1	249
pancierová priechodka 50 cm	203	1	203
kábel k zámku 6 m	300	1	300
elektrický otvárač BEFO PROFI	825	1	825
riadiaca jednotka DAVE	6000	1	6000
PIR senzor	245	2	490
I/O modul pre DAVE	700	1	700
RFID modul	1300	3	3900
USB RFID modul	1100	2	2200
SPOLU			24 669

Tabuľka C.1.: Celkové náklady projektu Elektra

PRÍLOHA D

OBSAH PRILOŽENÉHO CD

CD	
├── datasheety	katalógové listy použité pri práci
├── DPS	dáta k plošným spojom (schémy, *.BRD, gerber data)
│ ├── expansion	dáta k rozširujúcej doske I/O
│ ├── gerber	gerber dáta napanelizovaných DPS
│ ├── packages	navrhnuté footprinty k použitým súčiastkam
│ ├── padstacks	navrhnuté spájkovacie plôšky k footprintom
│ ├── rfid_modul	dáta k RFID modulu
│ └── rfid_usb_modul	dáta k RFID USB modulu
├── SW	software
│ ├── Elektra	zdrojové *.py súbory hlavného programu
│ └── PIC	zdrojové súbory pre PIC10F206 (MPLAB IDE)
└── BP_Drozd_Stanislav_2016.pdf	text bakalárskej práce