Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

FIIT-13428-56270

Bc. Ondrej Perešíni

EXPERIMENTÁLNY VNORENÝ SYSTÉM PRE RIADENIE INTELIGENTNÝCH AKČNÝCH ČLENOV CEZ INTERNET

Diplomová práca

Študijný program: Počítačové a komunikačné systémy a siete
Študijný odbor: 9.2.4 Počítačové inžinierstvo
Miesto vypracovania: Ústav počítačových systémov a sietí, FIIT STU Bratislava
Vedúci práce: doc. Ing. Tibor Krajčovič, PhD.

máj 2013

S T U
F I I T
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Zadanie diplomovej práce

Meno študenta: **Bc. Ondrej Perešini**

Študijný program: Počítačové a komunikačné systémy a siete

Študijný odbor: Počítačové inžinierstvo

Názov práce: **Experimentálny vnorený systém pre riadenie inteligentných akčných členov cez inernet.**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie, zodpovedúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti alebo o priebežné výsledky Vášho riešenia alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav počítačových systémov a sietí, FIIT STU Bratislava

Vedúci práce: **doc. Ing. Tibor Krajčovič, PhD.**

Termíny odovzdania:

podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti:
home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt

V Bratislave dňa 13. februára 2012

doc. Ing. Pavel Čičák, PhD.
poverený vedením ústavu

SLOVENSKÁ TECHNICKÁ UNIVERZITA Fakulta informatiky a informačných technológií BRATISLAVA

S T U
F I I T
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly:	Ondrej Perešíni, Bc.
Študijný program:	Počítačové a komunikačné systémy a siete
Kontakt:	ondrej.peresini@gmail.com

Výskumník:

Meno, priezvisko, tituly:	Tibor Krajčovič, doc. Ing. PhD.

Projekt:

Názov:	Experimentálny vnorený systém pre riadenie inteligentných akčných členov cez internet.
Názov v angličtine:	Experimental embedded system for intelligent actuators control through the Internet.
Miesto vypracovania:	Ústav počítačových systémov a sietí, FIIT STU, Bratislava
Oblasť problematiky:	Vnorené systémy

Text návrhu zadania[2]

Zameranie témy diplomovej práce je z oblasti návrhu a implementácie inteligentných akčných členov zabezpečujúcich riadenie sledovaných veličín pomocou vopred stanovených pravidiel. Takéto pravidlá sa budú ďalej dať definovať a modifikovať cez počítačovú sieť pomocou navrhnutého programového vybavenia. Analyzujte existujúce riešenia inteligentných domácností, spôsoby ich ovládania a porovnajte výhody a nevýhody týchto riešení. Analyzujte dostupné senzorické systémy a navrhnite ich vhodné zapojenie pre vybranú aplikáciu. Vyberte a použite vhodné riešenie vnoreného systému vo forme existujúceho riešenia na báze vývojového kitu. Navrhované riešenie vnoreného systému a senzorov musí dbať najmä na dodržanie čo najnižšej celkovej ceny, minimálnej spotreby pre potrebu napájania zo solárnych článkov a malej veľkosti z pohľadu ľahkého nasadenia do už existujúcej domácnosti. Súčasťou diplomovej práce je aj návrh a implementácia vlastného komunikačného protokolu, ktorý bude slúžiť na ovládanie vnoreného systému s akčnými členmi a prenos informácií zo senzorických systémov. Tieto informácie budú následne zobrazované pomocou implementovaného softvérového riešenia na zvolenej platforme. Pri návrhu komunikačného protokolu a programového vybavenia dbajte na robustnosť riešenia a možnosť ľahkej programovej prenositeľnosti medzi viacerými systémovými platformami. Výsledné riešenie overte na praktických príkladoch, kde bude demonštrované reálne nasadenie systému.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- SUWARTADI, E., GUNAWAN, C., SETIJADI P, A., Machbub, C. First Step Toward Internet Based Embedded Control System. Bandung Institute of Technology, Department of Electrical Engineering, Indonesia. Proceeding of the 5th Asian Control Conference. Dostupné online: http://ascc2004.ee.mu.oz.au/proceedings/papers/P179.pdf (12.11.2011)
- MEIJER, G.C.M. Concepts and focus point for intelligent sensor systems. Delft University of Technology, Department of Electrical Engineering, Netherlands. Proceedings of Eurosensors VII. Dostupné online: http://www.sciencedirect.com/science/article/pii/0924424794801096 (21.11.2011)

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Ondrej Perešíni, konzultoval(a) a osvojil(a) si ho doc. Ing. Tibor Krajčovič, PhD. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 25.1.2012

Perešíni
Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]
Dňa: 31. 1. 2012

Podpis garanta predmetov

[3] 2-3 vedecké zdroje, každý na samostatnom riadku a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKLUTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Počítačové a komunikačné systémy a siete

Autor: Bc. Ondrej Perešíni
Diplomová práca: Experimentálny vnorený systém pre riadenie inteligentných akčných členov cez internet
Vedúci diplomovej práce: doc. Ing. Tibor Krajčovič, PhD.
máj 2013

Diplomová práca sa zaoberá analýzou súčasných spôsobov realizácie inteligentných akčných členov v modernej domácnosti a návrhom vlastného riešenia, ktoré je lacnejšie, modulárnejšie, bezpečnejšie a v neposlednom rade aj otvorenejšie ako v prípade komerčných produktov.

Teoretická časť práce bližšie charakterizuje a zdôvodňuje použitie vybraného hardvéru, šifrovacieho algoritmu, rovnako ako aj rôzne typy prístupov k realizácií autonómnej funkcionality a vzájomnej komunikácie. V práci sú ďalej charakterizované rozdielne typy sieťových komunikačných protokolov ako aj návrh vlastného protokolu na aplikačnej vrstve. Výsledkom teoretickej časti je špecifikácia požiadaviek a návrh riešenia.

Súčasťou praktickej časti je realizácia niekoľkých autonómnych akčných členov, ktoré reagujú na stanovené podnety a zasahujú podľa stanovených pravidiel. Tieto pravidlá je možné ďalej upravovať pomocou navrhnutého používateľského rozhrania v riadiacej jednotke, ktorá je pripojená na internet. Zmeny je tak možné realizovať z akéhokoľvek zariadenia, ktoré je pripojené na internet a umožňuje zobrazovať web stránky. Komunikačné rozhranie dbá na používateľskú prívetivosť a dostatočné zabezpečenie voči rôznym typom útokov.

Výsledkom diplomovej práce je experimentálny prototyp navrhovaného riešenia, ktorý môže slúžiť na priame nasadenie do domácnosti alebo na rozšírenie výsledného produktu o ďalšie autonómne členy. Samostatná kapitola dokumentu je venovaná aj testovaniu a používateľskej príručke, kde je opísaný spôsob inštalácie a konfigurácie výsledného produktu.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Computer and Communication Systems and Networks

Author: Bc. Ondrej Perešíni
Master's Thesis: Experimental embedded system for intelligent actuators control through the internet
Supervisor: doc. Ing. Tibor Krajčovič, PhD.
2013, May

This diploma thesis analyses the current available options of intelligent house actuators realization and propose new own solution, which is less expensive, more robust and more secure against intrusion. The proposed solution is opened and therefore ready for the future improvements.

The theoretical part of this diploma project is dedicated to the proximate characterisation of chosen hardware, encryption algorithm and the mutual communication of the independent modules. Moreover the project describes different types of network communication protocols and suggests the new communication protocol, which is designed and optimised for the usage in the autonomous modules. The result of theoretical part describe proposal of the concrete solution for the whole system.

Separate part of project is dedicated to the description and realisation of some modules with defined specifications. These modules react with environment according to the established rules, which can be easily altered. Central unit is connected to the internet and therefore authorized users can easily modify behaviour of all modules from all over the world. Communication interface is secure against common types of attacks and also user friendly for the better user experience.

The result of this diploma project is experimental prototype of the proposed solution, which is suitable for the direct deployment in the modern households. Separate document chapter is dedicated to the final examination of correct system behaviour. The last chapter contains user guide for system installation and configuration.

Poďakovanie

Môjmu vedúcemu diplomovej práce doc. Ing. Tiborovi Krajčovičovi, PhD. chcem na tomto mieste vysloviť poďakovanie, ktoré si zaslúžil počas vypracovávania tejto práce najmä vďaka ochote pomôcť pri riešení rôznych vzniknutých problémov.

Obsah

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

V súčasnosti patrí medzi moderné trendy vybavovať nové, ale aj staršie bytové priestory rôznymi senzormi a inteligentnými akčnými členmi, ktoré vykonávajú používateľom stanovené príkazy. Na komerčnom trhu existuje hneď niekoľko riešení, ktoré poskytujú danú funkcionalitu. Všetky takéto produkty majú ale viacero spoločných vlastností, ktoré sú pre niektorých používateľov neakceptovateľné. V prvom rade je to ich cena, ktorá sa pri niektorých modeloch šplhá až do astronomických výšok. Vysoká cena vyplýva z lukratívnosti daného odvetvia a často aj zo špecifického návrhu konkrétneho zariadenia na mieru. Takýto prístup ale nie je vždy nevyhnutný a zbytočne zvyšuje koncovú cenu. Zariadenia vhodné do inteligentnej domácnosti by mali byť preto plne modulárne a autonómne. To znamená, že všetky senzory, ale aj akčné členy by mali byť autonómne, schopné decentralizovanej komunikácie a plne modulárne v prípade požadovanej zmeny. Výsledkom je sústava inteligentných členov, ktoré dokážu fungovať autonómne na základe prednastavených vlastností. V prípade nadviazania komunikácie s riadiacou jednotkou by sa zosynchronizovali s okolím a podľa používateľských požiadaviek pozmenili vykonávanú činnosť. Samotná modulárnosť je veľmi dôležitá, pretože umožňuje zapojiť rôznorodé členy do jednotného a komplexného systému. Takýto systém je možné kedykoľvek rozšíriť o nové členy bez starostí o spätnú kompatibilitu s centrálnou jednotkou, pretože aktualizácia o novú funkcionalitu sa dá vykonať jednoduchou zmenou riadiaceho firmvéru.

Ďalším problémom komerčných riešení je ich úplná uzavretosť. Nemodulárne a uzavreté členy neumožňujú rozsiahlu používateľskú konfigurovateľnosť a preto v prípade rozšírenia požadovanej funkcionality sa často stáva, že kompletne celý systém je nutné nahradiť novším, čo je taktiež výrazne finančne náročné. Nad uzavretým systémom nemá používateľ plnú kontrolu a teda ani istotu, či nie sú jeho činnosti a rutiny sledované útočníkom z dôvodu získania osobného prospechu voči sledovanej osobe. Práve bezpečnosť je jeden z faktorov, ktorý mnohí výrobcovia, ale aj používatelia podceňujú. Uveďme príklad: systém inteligentnej domácnosti je pripojený k RFID [1] čítačke používateľských kľúčov a na základe načítaných dát sa otvárajú bezpečnostné dvere. V prvom rade je potrebné zaručiť bezpečnosť samotnej RFID čítačky, následne zabezpečiť komunikačný kanál a v neposlednom rade zabezpečiť aj centrálnu jednotku a celý systém voči vonkajšiemu útoku. Komplexné zabezpečenie tak vyžaduje súhru na viacerých vrstvách, kde je potrebné nastaviť rovnakú bezpečnostnú politiku. V prípade uzavretého riešenia sa používateľ môže

spoliehať len na splnenie všetkých bezpečnostných požiadaviek. V prípade nedostatočnej bezpečnostnej politiky hrozí útok zvnútra cez vlastnú infraštruktúru alebo vonkajší útok, ktorý je možné realizovať napríklad pomocou bezpečnostnej diery v hlavnom sieťovom smerovači. Presmerovanie a odchytenie komunikačných paketov je pre skúseného útočníka jednoduchý problém a v prípade nešifrovanej a slabo zabezpečenej komunikácie tak hrozí jednoduchá replikácia komunikácie, ktorá môže vyústiť k nedovolenému prieniku do domácnosti cez otvorené dvere.

Všetky vymenované problémy, ako aj záujem o vytvorenie vlastného experimentálneho systému splňujúceho všetky stanovené požiadavky nás viedlo k realizácii tejto diplomovej práce. Na našej, ale aj na iných fakultách už bol daný problém viackrát opisovaný v rôznych diplomových prácach. Tieto diplomové práce sa síce zaoberali viacerými vyššie spomínanými problémami ako je vysoká cena alebo uzavretosť systému, avšak my sme si okrem toho stanovili aj ďalšie rozširujúce požiadavky, ktoré napríklad zahrňujú autonómne a plne modulárne inteligentné členy komunikujúce cez decentralizovanú sieť. V celom systéme musí byť navyše zachovaná vysoká úroveň bezpečnosti, pričom niektoré členy môžu komunikovať aj bezdrôtovo. Podrobnejšie špecifikácie požiadaviek a funkcionality sú stanovené v príslušnej kapitole.

Cieľom diplomovej práce nie je len vytvorenie návrhu takéhoto systému, ale aj jeho realizácia v experimentálnej podobe. Takéto riešenie sa aj vďaka požiadavke na plnú modulárnosť bude dať ďalej rozširovať a dopĺňať o ďalšie členy, čo umožní výrazné zlepšenie celkovej funkcionality. V budúcnosti tak môže ktokoľvek pokračovať v rozširovaní predkladaného systému, alebo ho experimentálne nasadiť do domácnosti.

Diplomová práca je rozčlenená na 8 kapitol. Prvá kapitola je venovaná úvodnému opisu aktuálnej situácie. Druhá kapitola obsahuje analýzu problematiky, ktorá popisuje niektoré práce podobného zamerania, komerčné riešenia, charakterizuje vnorené systémy, komunikačné protokoly, šifrovanie, ale aj monitorovanie a reguláciu prostredia. Samostatná kapitola je venovaná špecifikácií požiadaviek na vytváraný systém, spoločne s opisom benefitov navrhovaného riešenia. Rozsiahle kapitoly s návrhom a opisom riešenia podrobne popisujú návrh a realizáciu funkčného prototypu so všetkými vzniknutými problémami. Súčasťou diplomovej práce je aj kapitola s overením výsledkov a záverečné zhodnotenie, ktoré charakterizuje možné smery budúceho rozširovania a zlepšovania tejto práce. Posledná kapitola obsahuje technickú dokumentáciu s programátorskou a používateľskou príručkou.

1.1 Slovník pojmov

- **Ethernet** – technológia počítačových sietí, ktorá sa využíva pre lokálne siete LAN
- **Bluetooth** – bezdrôtová komunikačná technológia, ktorá slúži na nadviazanie spojenia a výmenu údajov medzi dvomi zariadeniami. Pracuje na frekvencii 2,4 GHz
- **Zigbee** – bezdrôtová komunikačná technológia určená pre spojenie nízkoenergetických zariadení na malé vzdialenosti
- **Centrálna jednotka** – slúži na zbieranie dát z externých modulov, používateľskú interakciu a zasielanie príkazov do externých modulov
- **Externý modul** – zbiera údaje o prostredí, posiela ich centrálnej jednotke a podľa nastavení aj zasahuje pomocou akčných členov
- **Watchdog časovač** – hardvérový alebo softvérový prostriedok, ktorý umožňuje bezpečné zotavenie zo stavu zaseknutia (deadlock-u)
- **Arduino shield** – rozširujúci modul, ktorý poskytuje doplnkovú funkcionalitu
- **Ad-hoc sieť** – nezávislá množina komunikujúcich uzlov bez infraštruktúry
- **GRID karta** – bezpečnostná karta s tabuľkou bezpečnostných kódov, ktoré sa používajú na autentifikáciu používateľa
- **Proxy server** – server v počítačovej sieti, ktorý umožňuje klientom nepriame pripojenie k inému serveru. Proxy server umožňuje zvýšiť zabezpečenie, rýchlosť alebo iným spôsobom zjednodušiť komunikáciu
- **Debugging nástroje** – nástroje poskytujúce rozšírenú funkcionalitu na vyhľadávanie programátorských chýb v zdrojovom kóde

1.2 Použité skratky

- **UART** – *Universal Asynchronous Receiver and Transmitter* (sériové asynchrónne komunikačné rozhranie)
- **IP adresa** – *Internet Protocol address* (logický číselný identifikátor sieťového rozhrania)
- **MAC adresa** - *Media Access Control address* (fyzická adresa sieťového zariadenia na linkovej vrstve)
- **DHCP server** - *Dynamic Host Configuration Protocol* (server umožňujúci automatickú konfiguráciu sieťového zariadenia a pridelenia IP adresy zo zoznamu voľných adries)

- **NTP** - *Network Time Protocol* (sieťový protokol slúžiaci na synchronizáciu času medzi klientom a serverom)
- **WOL** - *Wake-on-LAN* (sieťový rámec definovaného štandardu, ktorý zobudí špecifikované zariadenie s podporou tohto štandardu)
- **RFID** - Radio Frequecy IDentification (vysokofrekvenčný identifikačný prvok, ktorý sa používa na identifikácia tovaru, osobných kariet a iných objektov)
- **PWM** - *Pulse Width Modulation* (pulzná šírková modulácia určená na prenos informácie alebo pre plynulé ovládanie zariadení ako elektromotor)
- **HVAC** - *Heating Ventilation and Air Conditioning* (štandard ovládania automatických systémov správy vnútorného prostredia)
- **USB** – *Universal Serial Bus* (univerzálna sériová zbernica slúžiaca na pripájanie rôznych periférií k počítaču a iným zariadeniam)
- **LED** – *Light Emitting Diode* (svetelná indikačná dióda, ktorá vyžaruje svetlo v úzkom spektre)
- **Wi-Fi** - *Wireless Fidelity* (bezdrôtová komunikačná technológia využívaná najmä na pripájanie rôznych zariadení do lokálnej siete a internetu)
- **RISC** - *Reduced Instruction Set Computer* (architektúra s jednoduchšou a obmedzenou inštrukčnou sadou)
- **FLASH** – *Flash memory* (rýchlejšia EEPROM pamäť, uchováva si údaje aj po odpojení napájania)
- **SRAM** - *Static Random-Access Memory* (pamäť tvorená z preklápacích obvodov, po prerušení napájania sa údaje stratia)
- **EEPROM** – *Electrically Erasable Programmable Read-Only Memory* (elektronicky vymazateľná programovateľná pamäť)
- **API -** *Application programming interface* (rozhranie pre jednoduché programovanie aplikácií alebo ovládanie hardvéru)
- **MIPS** – *Million Instructions Per Second* (popisuje výkon mikroprocesora v miliónoch inštrukcií za sekundu)
- **USART** - *Universal Synchronous/Asynchronous Receiver/Transmitter* (komunikačné rozhranie pre sériovú komunikáciu v synchrónnom a asynchrónnom režime)
- **SPI** – *Serial Peripheral Interface* (sériové periférne rozhranie, používa sa pri vzájomnej komunikácií medzi mikroprocesorom a ďalšími obvodmi)

- **I²C** - *Inter-Integrated Circuit* (Two-wire rozhranie, slúži na pripojenie nízko-rýchlostných periférií)
- **JTAG** - *Joint Test Action Group* (štandardizované rozhranie, pomocou ktorého sa dajú programovať a testovať mikroprocesory)
- **POE** - *Power Over Ethernet* (napájanie zariadenia pomocou sieťového rozhrania)
- **IEEE** - *Institute of Electrical and Electronics Engineers* (medzinárodná nezisková profesijná organizácia, ktorá sa vo veľkej miere zasluhuje na tvorbe štandardov)
- **SD** – *Secure Digital* (pamäťová karta založená na Flash technológií, najčastejšie sa používa ako dátové úložisko v malých mobilných zariadeniach)
- **FAT** – *File Allocation Table* (jednoduchý súborový systém vhodný pre vnorené systémy a pamäťové SD karty)
- **WOL** - *Wake-on-LAN* (štandard umožňujúci prebudenie uspatých zariadení cez sieť)
- **HTTPS** - *Hypertext Transfer Protocol Secure* (zabezpečený sieťový protokol pre prenos dát medzi serverom a klientom)
- **SSL** - *Secure Sockets Layer* (protokol s asymetrickým šifrovaním používaný pre zabezpečenú komunikáciu)
- **TLS** - *Transport Layer Security* (nástupca SSL)
- **SMS** - *Short Message Service* (krátka textová správa prenášaná cez GSM sieť)
- **GSM** - *Global System for Mobile Communications* (mobilná telefónna sieť)
- **GPRS** - *General Packet Radio Service* (mobilná dátová služba na GSM sieti)
- **ADC** – *Analog to Digital Converter* (analógovo-číslicový prevodník)
- **VLAN** - *Virtual Local Area Network* (virtuálna sieť v rámci lokálnej siete)
- **DOS** – *Denial Of Service attack* (technika útoku, ktorá posielaním neopodstatnených požiadaviek znefunkčňuje poskytované služby)
- **AČ** – *Akčný člen* (výkonný člen, ktorý je pripojený k externému modulu a vykonáva funkcionalitu ovplyvňovania prostredia)
- **URL** - *Uniform Resource Locator* (univerzálny formát mien používaný na označenie zdroja na internete)
- **NAT** – *Network Address Translation* (preklad medzi verejnou a privátnou sieťovou IP adresou)

2 Analýza problému

Oblasť analýzy vybranej problematiky je značne rozsiahla, ale keďže niektoré časti analýzy sú zhodné s inými diplomovými prácami, tak v tejto kapitole sa budeme sústreďovať najmä na analýzu doplnkových oblastí, ktorými sa dané práce ešte nezaoberali. Napriek tomu v tejto kapitole poskytneme stručné zhrnutie všetkých dôležitých oblastí bez ohľadu na to, či už boli analyzované aj v iných prácach.

Kapitola analýzy problému bližšie popisuje existujúce práce s podobnou tematikou, charakterizuje požiadavky na systém, popisuje komerčne dostupné riešenia, výber vhodného vnoreného systému, opisuje sieťové komunikačné protokoly, spôsoby bezdrôtovej komunikácie, zabezpečenie šifrovaním a charakterizuje jednotlivé prvky monitorovania a regulácie.

2.1 Prehľad diplomových prác s podobnou tematikou

Na našej fakulte bolo riešených niekoľko diplomových prác s podobnou tematikou. V analýze som použil dvojicu z nich, ktoré sa najviac približujú zvolenej oblasti. Obe práce pojednávajú o vnorenom systéme pre vzdialené monitorovanie a riadenie zvoleného prostredia, avšak rozchádzajú sa v spôsobe návrhu a realizácie riešenia. Obidve práce sú bližšie opísané v samostatných kapitolách. Pred začiatkom vlastnej analýzy sme bližšie analyzovali obidve práce, pričom v každej z nich sme vyzdvihli pozitívne a zdôvodnili negatívne dopady daného riešenia z pohľadu našej práce. Vo výsledku nie je ani jeden z analyzovaných prístupov vhodný na realizáciu našej práce, pretože obidve riešenia nie sú dostatočne modulárne, zabezpečené, alebo nesplňujú nami definované podmienky.

2.1.1 Vnorený systém pre vzdialené monitorovanie a riadenie domácnosti - Bc. Peter Jombík [2]

Výsledkom diplomovej práce je návrh a realizácia vnoreného systému na ovládanie domácnosti, ktorý pozostáva z jednej centrálnej jednotky a viacerých externých ovládaných modulov. Centrálna jednotka umožňuje vzdialené zapínanie a vypínanie napájania zariadení, ktoré sú pripojené k elektrickej sieti pomocou centrálnej jednotky, ako aj nastavovanie výstupov a načítavanie vstupov z externých modulov. Ako komunikačný kanál boli zvolené siete typu Ethernet a Bluetooth. Konfigurácia systému pomocou web rozhrania umožňuje nastavovanie správania sa pomocou širokého spektra zariadení.

Výsledný systém vzdialeného riadenia domácich spotrebičov umožnil výrazne znížiť spotrebu elektrickej energie, ako aj riziko krádeže. Výrobná cena prototypu na úrovni 200 € je výrazne nižšia ako v prípade komerčných riešení. Zabezpečenie systému je realizované pomocou spárovania komunikujúcich jednotiek, čo sa v našom ponímaní nedá považovať za autonómne a modulárne riešenie. V prototype je vyriešené aj zabezpečenie web prístupu pomocou hesla, čo znižuje možnosti útokov.

Nedokonalosť danej práce spočíva v nefunkčnosti celého systému v prípade výpadku centrálnej jednotky. Nami navrhovaný modulárny systém by mal fungovať aj v prípade výpadku niekoľkých zariadení spoločne s centrálnou riadiacou jednotkou. Ďalším obmedzením centrálnej jednotky je schopnosť obsluhovať maximálne sedem externých jednotiek, čo síce pre potreby danej práce postačuje, avšak v našom prípade je to značne nedostačujúca hodnota. Celý systém je koncipovaný len na zapínanie a vypínanie výstupov, čo sa síce môže hodiť k zapínaniu svetiel alebo otváraniu dverí, avšak je to absolútne nevyhovujúce pre ovládanie akčných členov. Problémom je aj nemožnosť zaznamenávania vlastností okolitého prostredia pomocou senzorov.

Z analyzovanej práce je pre našu diplomovú prácu vhodná analýza bezdrôtových komunikačných technológií, rozbor jednoduchých komerčných zariadení ovládania domácností, zabezpečenie web prístupu, ale aj návrh sieťového komunikačného protokolu.

2.1.2 Experimentálny vnorený systém pre vzdialené monitorovanie a riadenie - Bc. Stanislav Kišák [3]

Druhá diplomová práca je veľmi podobného zamerania, avšak berie si za cieľ vytvoriť vnorený systém pre vzdialené monitorovanie a riadenie prostredia v uzatvorenom systéme záhradného fóliovníka. Práca pojednáva o rôznych variantoch rozdelenia funkčných modulov nielen medzi vnorené systémy, ale aj rozdelenie na hardvérovú a softvérovú časť. Práca obsahuje analýzu senzorov teploty, vlhkosti a svetelnosti. Súčasťou analýzy sú aj rôzne spôsoby bezdrôtového prepojenia jednotlivých častí systému. Kapitola s návrhom opisuje viaceré spôsoby efektívneho regulovania prostredia.

Vytvorený systém monitoruje teplotu, vlhkosť, svetelnosť a rosný bod prostredia záhradného fóliovníka. Monitorovacia časť pozostáva z riadiacej, komunikačnej a senzorovej jednotky. Medzi riadiacou a komunikačnou jednotkou prebieha sériová UART komunikácia. Monitorovacia jednotka komunikuje s ostatnými časťami systému bezdrôtovo pomocou technológie Zigbee. Nevýhodou je použitie jedinej regulačnej jednotky, ktorá je spoločná pre

celý systém a ovláda všetky akčné členy. Práca síce zodpovedá autorovej špecifikácií, avšak tento prístup je nevhodný pre našu špecifikáciu, ktorá počíta s rozsiahlym prostredím na ktorom je z komunikačných dôvodov vhodnejšie použiť viacero regulačných jednotiek.

Dáta sú zaznamenávané a zbierané pomocou protokolu Syslog, ktorý vyžaduje ďalší monitorovací a reportovací nástroj. Takýto nástroj je značne komplikovaný a nie je schopný efektívne fungovať na jednoduchom vnorenom hardvéri. Súčasťou systému tak musí byť aj vysokovýkonný zaznamenávací a prezentačný nástroj. Takýto návrh značne komplikuje celý systém, pretože sa spolieha na použitie nákladného nástroja, ktorý by pri vhodnom návrhu dokázal nahradiť celý monitorovací a regulačný systém. Týmto by sa obmedzili výhody plynúce z použitia vnorených systémov.

Oddelenie riadiacej a prezentačnej časti považujeme za nedokonalosť návrhu, pretože takýto prístup značne komplikuje používateľské rozhranie. Aj keď sú v riadiacej časti vytvorené rýchle náhľady na prezentačnú časť, tak v prípade neštandardného zobrazenia sa používateľ musí prihlásiť do samostatného prezentačného nástroja a manuálne vykonať zmeny. Riadiaca časť je realizovaná pomocou serverového web frameworku, ktorý taktiež vyžaduje výkonný hardvér, ale poskytuje používateľsky príjemné prostredie a umožňuje interoperabilitu medzi rôznymi zariadeniami.

Diplomová práca sa nezaoberá žiadnym zabezpečením a výsledný systém nie je modulárny ani autonómny. Napriek tomu práca obsahuje kvalitnú analýzu z ktorej môžeme ďalej vychádzať.

2.2 Oblasti bližšej analýzy

Pred ďalšou podrobnejšou analýzou je potrebné najskôr stanoviť požiadavky a identifikovať problémové oblasti, ktoré si vyžadujú dôkladnejšiu analýzu. Navrhovaný prototyp systému na monitorovanie a riadenie prostredia bude obsahovať viacero akčných členov a senzorov. V tejto kapitole je potrebné stanoviť a špecifikovať všetky požiadavky na takýto systém.

2.2.1 Požiadavky na systém

Monitorovací a riadiaci systém pozostávajúci z inteligentných akčných členov, senzorov a riadiacej jednotky bude založený na vnorených systémoch, ktoré vzájomne komunikujú cez počítačovú sieť a vykonávajú požadovanú funkcionalitu. Je preto nutné bližšie analyzovať spôsoby sieťovej komunikácie, ale aj návrhy vlastných komunikačných protokolov na aplikačnej vrstve, ktoré by najlepšie vyhovovali navrhovanému systému. Akčné členy

a senzory sú riadené vlastným vnoreným hardvérom, ktorý musí poskytovať dostatočný výpočtový výkon, minimálnu spotrebu, cenu a zároveň musí obsahovať množstvo komunikačných rozhraní pre najrôznejšie senzory a periférie. Použitý vnorený hardvér tak môže byť rozdielny pre centrálnu jednotku a externé moduly. Pred výberom konkrétneho hardvéru je nutné bližšie analyzovať vhodné a dostupné prostriedky. Uchovávanie zaznamenaných dát vyžaduje dostatočne veľkú dostupnú pamäť. Existuje viacero spôsobov ako realizovať ukladanie veľkého množstva dát. Analýza musí preto pokrývať aj spôsoby realizácie záznamu takýchto dát. Niektoré externé členy môžu využívať bezdrôtovú komunikáciu. Existuje pritom viacero možností realizácie bezdrôtovej výmeny údajov. Všetky komunikačné kanály majú svoje výhody a nevýhody, ktoré je taktiež nutné bližšie analyzovať. Každý člen v systéme vyžaduje napájanie elektrickou energiou. Pri členoch pripojených počítačovou sieťou a inou kabelážou to nie je problém, avšak členy s bezdrôtovou komunikáciou musia používať samostatný napájací obvod. Analýza preto musí pokrývať aj rôzne spôsoby alternatívneho získavania energie pre napájanie bezdrôtových členov, ako aj spôsoby na minimalizovanie ich elektrického príkonu. Analýza musí taktiež obsahovať aj popis vhodných komerčných produktov, ktoré ponúkajú podobnú funkcionalitu.

Samostatnou kapitolou je zabezpečenie celého systému voči nežiadanému prieniku. Systém musí byť odolný nielen voči odchytávaniu dátovej komunikácie, ale aj proti priamemu útoku spôsobom replikácie príkazov. Systém musí používať zabezpečenú komunikáciu nielen pri internej výmene údajov, ale aj pri komunikácií s používateľom na nezabezpečenom komunikačnom kanáli akým je aj internet. Okrem toho systém musí byť zabezpečený aj voči vnútorným poruchám, ktoré môžu vyradiť z činnosti niektoré členy. Na takéto zabezpečenie sa používa Watchdog časovač, ktorý v prípade straty reakcie zo systému ho bezpečne reštartuje a obnoví stratenú komunikáciu.

Monitorovací a riadiaci systém pozostáva z centrálnej riadiacej jednotky a externých modulov. Zatiaľ čo externé moduly zbierajú informácie zo senzorov a vplývajú na prostredie pomocou akčných členov, tak centrálna jednotka slúži na celkovú koordináciu takéhoto systému. Všetky jednotky musia byť autonómne a schopné samostatnej činnosti. Externé jednotky po pripojení k sieti automaticky získajú IP adresu z DHCP servera a načítajú aktuálny čas z internetového časového servera (NTP). Vďaka tomu môžu byť všetky dáta zaznamenávané s presným časom, čo je nevyhnutné pri prezentovaní údajov a nastavovaní hodnôt používateľom. Centrálna jednotka bude obsluhovať používateľské požiadavky pomocou web rozhrania a preto by jej IP adresa mala byť nastavená staticky. Centrálna

jednotka sa v periodických časových intervaloch dopytuje na dostupnosť externých modulov z ktorých získava namerané údaje a nastavuje správanie podľa používateľských preferencií. Centrálna jednotka obsahuje aj kompletný záznam nameraných údajov zo všetkých externých modulov so senzormi. Samostatnou funkcionalitou centrálnej jednotky je prebudenie sieťových zariadení z režimu spánku pomocou WOL paketov. Používateľ tak môže kedykoľvek zapnúť ľubovoľné podporované zariadenie na diaľku a šetriť tak energiou, ktorá by bola spotrebovaná pri neustálej činnosti daných zariadení.

Celý systém musí byť modulárny, čo umožňuje pridávať ľubovoľné externé moduly s rozšírenou funkcionalitou senzorov alebo akčných členov. Pre potreby našej diplomovej práce postačuje demonštrácia trojice rozdielnych typov senzorov a akčných členov. Zo senzorov je nutné implementovať digitálny senzor teploty, analógový senzor svetelnosti prostredia a RFID čítačku pri vchodových dverách. Akčné členy zahrňujú PWM regulované osvetlenie, krokový motorček nastavenia vykurovania a elektronické relé dverí. Špecifikované senzory a akčné členy netvoria konečnú množinu a vďaka vysokej modularite môže byť celý systém kedykoľvek rozšírený o ďalšie akčné členy a senzory. Navrhovaná množina členov umožňuje dostatočnú demonštráciu funkcionality a správania sa celého systému. Doplnenie ďalších členov je už vďaka poskytnutému spôsobu riešenia triviálnym problémom.

2.3 Komerčne dostupné riešenia

Okrem analyzovaných diplomových prác existuje aj množstvo komerčne dostupných riešení. Všetky tieto riešenia majú niekoľko spoločných menovateľov ako vysoká cena, uzavretosť voči zmenám a hlavne vysoká špecializovanosť. Napriek tomu sú takéto riešenia značne populárne. V analýze sa bližšie sústredíme na trojicu rôznych modelov.

2.3.1 Sensaphone Web600 Web-Based Monitoring and Alarm System

Prvý produkt od spoločnosti *Sensaphone* [5] slúži na monitorovanie domáceho prostredia a informovanie používateľa o kritických stavoch akými sú napríklad požiar, nepovolené otvorenie dverí a okien alebo výpadok napájania. Používateľ má k dispozícii 6 vstupov, na ktoré môže pripojiť rôzne senzory. Na konfigurácia zariadenia sa používa web rozhranie, ktoré zobrazuje aktuálne stavy sledovaných veličín, ale aj ich históriu. V prípade kritického prípadu je používateľ informovaný pomocou e-mailu alebo textovej správy.

Obr. 2-1 Sensaphone Web600 sieťový monitorovací systém s alarmom [4]

Výhodou zariadenia je možnosť pripojenia ľubovoľných analógových senzorov, avšak neumožňuje žiadnu formu regulácie prostredia. K zariadeniu sa dá voliteľne dokúpiť aj záložná batéria, ktorá dokáže zariadenie napájať po dobu 2 hodín. V takom prípade je ale nevyhnutné zabezpečiť samostatné napájanie všetkých sieťových prvkov, ktoré zabezpečujú pripojenie do internetu. Predajná cena sa pohybuje na vysokej úrovni: $355.50 [4].

Výrobca	Sensaphone
Model	FGD-W600
Minimálne systémové špecifikácie	CPU s frekvenciou minimálne 1.2 GHz, architektúra Intel Pentium/Celeron family, AMD K6/Athlon/Duron family, alebo iné kompatibilné
	256 MB RAM
	Minimálne jeden USB 2.0 port
	30 MB diskového priestoru
	Operačný systém: Windows 2000 SP4, XP SP2 32-bit, Vista 32-bit
	Minimálne rozlíšenie monitora: Super VGA (800 x 600)
Alarm	E-Mail, Text Messages, SNMP
	Komplexné plánovanie pre každý vstup a profil
	8 úrovní
Výstupy	8 programovateľných profilov so 4 výstupmi
Komunikácia	E-Mail - SMTP
	Textové správy
	Web stránky formátov: HTTP, PDA, WAP a XML
	SNMP – MIB, GET, GETNEXT a SET
	MODBUS®/TCP Slave Conformance Class 0 a 1

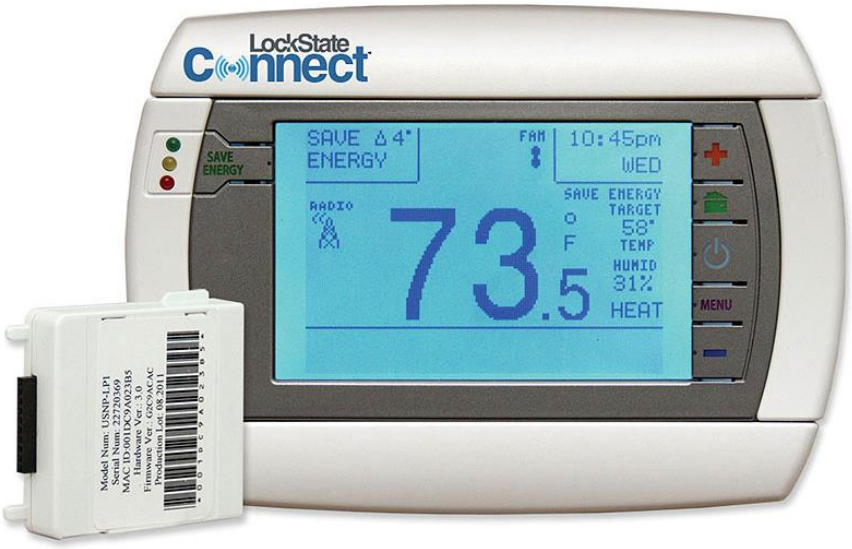
Vstupy	6 univerzálnych vstupov
	2.8K / 10K termistor
	4 – 20 mA prúdová slučka
	12-bit rozlíšenie
Záznam dát	32 000 záznamov s dátumom a časom
	Frekvencia: 1 sekunda až 1 mesiac
	Programovateľný výber kanálov
Komunikačné rozhranie	Ethernet 10/100Base-T
Lokálne indikátory	LED stav alarmu
	LED stav napájania
	LED indikácia sieťového pripojenia a aktivity
Požiadavky na napájanie	Externý zdroj 120 VAC 60Hz
	Voliteľne batéria (FGD-W610) s výdržou dvoch hodín
Prevádzkové podmienky	Vlhkosť : 0% - 90%
	Teplota : 0°C - 50°C
Certifikácia	FCC Part 15-B Compliant
Rozmery	14 x 8 x 3 cm
Hmotnosť	227 g

Tab. 2-1 Technické špecifikácie Sensaphone Web600 [4]

2.3.2 LockState Connect LS-90i Wi-Fi Touch Screen

Druhý produkt od spoločnosti *LockState* [7] už poskytuje bohatšie možnosti. Umožňuje monitorovať a nastavovať vykurovanie, klimatizáciu, ale aj zvlhčovanie vzduchu v domácnosti na diaľku. Okrem toho disponuje veľkým prehľadným dotykovým displejom, ktorý združuje všetky aktuálne informácie na jednom mieste a používateľ tak má okamžitú kontrolu nad celým systémom. Zariadenie sa do internetu pripája pomocou siete Wi-Fi. Napájanie je realizované externým adaptérom, ktorý je vhodné doplniť o záložný zdroj v prípade výpadku elektrickej siete.

Zariadenie umožňuje automatické prepínanie medzi vykurovaním a chladením, pričom k dispozícii sú 3 + 2 stupne regulácie. Podporované sú všetky HVAC systémy, avšak k zariadeniu nie je možné pripojiť vlastné senzory alebo akčné členy. Samotné zariadenie sa predáva za vysokých $299.00 [6].



Obr. 2-2 LockState Connect LS-90i WiFi Thermostat [6]

Výrobca	LockState
Model	LS-90i
Ovládateľné zariadenia	Takmer všetky populárne HVAC systémy spoločne s tepelnými čerpadlami a pomocnými vyhrievačmi, ventilátormi, zvlhčovačmi a odvlhčovačmi
Stupňov vykurovania	3
Stupňov chladenia	2
Riadenie	Automatické prepínanie vykurovania a chladenia
Indikátory opotrebenia	Vzduchový filter, zvlhčovanie podložky a UV žiarovky
Časové plány	2
Uzamknutie	Úplné aj čiastočné
Ovládanie	Dotyková obrazovka s perom
Regulačná teplota	8°C – 35°C
Dostupné konektory	C, H, B, O, W, W2, W3, Y, Y2, RH, RC, G, A, DH a EX
Napájanie	C-Wire, 24 VAC zdroj alebo 3 x AA batérie
Wi-Fi štandardy	B, G a N
Rozmery	4 x 15 x 10 cm
Hmotnosť	900 g

Tab. 2-2 Technické špecifikácie LockState Connect LS-90i [6]

2.3.3 Ecobee EB-EMS-02 Energy Management System

Ďalším analyzovaným produktom je riadiaci systém od spoločnosti *Ecobee* [9], ktorý je určený na ovládanie domácich HVAC systémov. Zariadenie disponuje veľkým grafickým displejom s aktuálnymi informáciami. Zabezpečená Wi-Fi konektivita umožňuje programovanie, riadenie a sledovanie meraných veličín aj na diaľku. Používateľ má k dispozícii prehľadné web rozhranie, ktoré zobrazuje všetky dostupné informácie o domácnosti. Výrobca poskytuje aj vlastné aplikácie na ovládanie domácností zo všetkých populárnych mobilných systémov. Napájanie zariadenia je zabezpečené pomocou externého adaptéra alebo priamo z HVAC systému. Zariadenie umožňuje nastavovať rôzne profily pre každý deň a minimalizovať tak energetické nároky domácnosti.

Obr. 2-3 ecobee EB-EMS-02 Energy Management System [8]

Ako voliteľné príslušenstvo sú k dispozícii aj bezdrôtové externé senzorové moduly, ktoré umožňujú merať napríklad vonkajšiu teplotu. Nevýhodou celého systému je jeho uzavretosť a nemožnosť pripojenia vlastných senzorov a akčných členov. Okrem toho je výrazne premrštená aj predajná cena, ktorá sa pohybuje na úrovni $510.59 [8].

Výrobca	ecobee
Model	EB-EMS-02
Regulácia teploty	Vykurovanie : 7°C - 26°C
	Chladenie : 14°C - 33°C
	Zobrazenie : 5°C - 37°C
	Presnosť: ± 0,5°C
Operačná teplota	-40°C - 70°C, termostat: 0°C - 55°C
Regulácia vlhkosti	Zvlhčovanie : 20% - 50%
	Odvlhčovanie: 30% - 60%

Regulácia vlhkosti	Zobrazenie : 0% - 90%
	Presnosť: ±2%
Operačná vlhkosť	5 - 95%, nekondenzujúca
Rozmery	Termostat: 139.5 x 82.5 x 25mm
	Článok: 118 x 254 x 32mm
Požiadavky napájania	24VAC, 3VA z HVAC systému
Záložná batéria	1x CR2032
Konektory	Y, W (O/B), G, W2 (AUX), R/H, R/C, ACC1, ACC1r, ACC2, ACC2r, ACC3, ACC3r, IN1+, IN1-, IN2+, IN2-, +12V, GND, D+, D-

Tab. 2-3 Technické špecifikácie ecobee EB-EMS-02 [8]

2.4 Vnorený systém

Na realizáciu autonómneho systému vzdialeného monitorovania a riadenia prostredia sú najvhodnejšie vnorené systémy. Charakteristiku ako aj funkcionalitu vnorených systémov je možné nájsť v obidvoch analyzovaných diplomových prácach [2] a [3]. Vnorené systémy vykonávajú špecializované úlohy, pričom sú neoddeliteľnou súčasťou riadených zariadení. Nespornou výhodou vnorených systémov je ich reaktívnosť, špecializácia, spoľahlivosť, cena, veľkosť a v neposlednom rade aj spotreba elektrickej energie. Vďaka všetkým týmto vlastnostiam sú vnorené systémy mimoriadne vhodné do kritického nasadenia s obmedzenými prostriedkami. Zásluhou vysokej špecializovanosti ponúkajú tieto systémy rovnakú, alebo aspoň dostatočnú funkcionalitu v porovnaní s plnohodnotnými výpočtovými prostriedkami. V kontexte našej práce sú vnorené systémy mimoriadne vhodné pre svoju nízku cenu, príkon, dostatočný výpočtový výkon a možnosti pripojenia špecializovaných vstupno-výstupných zariadení, akými sú napríklad senzory alebo iné akčné členy. Práve realizovaná programová výbava umožňuje vytvárať z obyčajných pasívnych členov inteligentné zariadenia schopné autonómnej prevádzky.

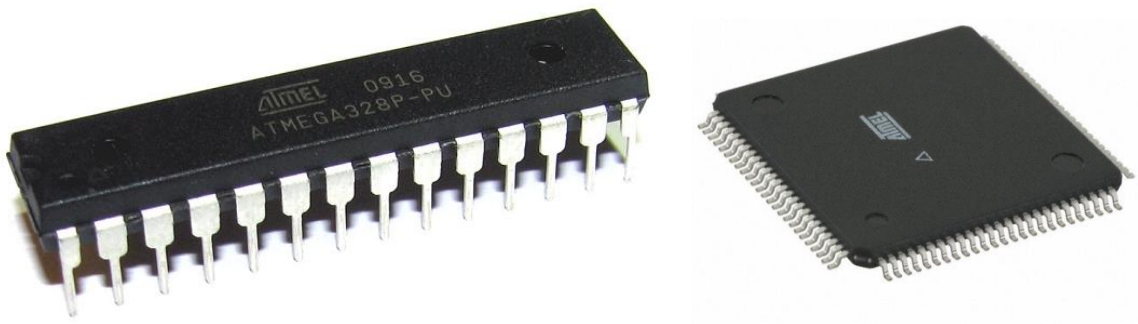
Pred návrhom konkrétneho riešenia je nutné bližšie analyzovať viacero mikroprocesorov, ktoré by mohli tvoriť základ navrhovaného systému. Túto analýzu už nie je potrebné vykonávať, nakoľko sa pri výbere konkrétneho mikroprocesora môžeme do značnej miery opierať o analýzu, ktorú sme vykonali už v predchádzajúcej bakalárskej práci [10]. Požiadavky kladené na náš systém sa ale mierne odlišujú a hlavný dôraz je kladený na minimálnu spotrebu, schopnosť pojať množstvo rôznorodých vstupno-výstupných zariadení a hlavne minimálnu cenu. Medzi hlavné požiadavky tak nepatrí obrovský výpočtový výkon,

keďže všetky analyzované mikroprocesory poskytujú dostatočný výkon pre sieťovú komunikáciu, ale aj ovládanie akčných členov a senzorov.

2.4.1 Výber mikroprocesora

Na rozdiel od použitého mikroprocesora v bakalárskej práci [10], ktorý poskytuje vysoký výkon spoločne s vyššou cenou, je pre realizáciu autonómnych modulov postačujúci omnoho jednoduchší procesor. Medzi hlavné požiadavky na vybraný mikroprocesor patrí najmä nízka cena, minimálna spotreba a bohaté možnosti pripojenia periférnych senzorov a akčných členov. Okrem externých modulov bude systém obsahovať aj centrálnu jednotku, ktorá bude zabezpečovať synchronizáciu medzi jednotlivými modulmi. Požiadavky na obidva typy mikroprocesorov sa takmer zhodujú, avšak mikroprocesor v centrálnej jednotke by mal obsahovať väčšie množstvo dostupnej pamäti FLASH a SRAM. Väčšie pamäťové nároky vyplývajú zo špecifikácie, kde centrálna jednotka musí komunikovať a zaznamenávať údaje zo všetkých modulov. Okrem toho bude poskytovať aj rozhranie pre používateľskú interakciu.

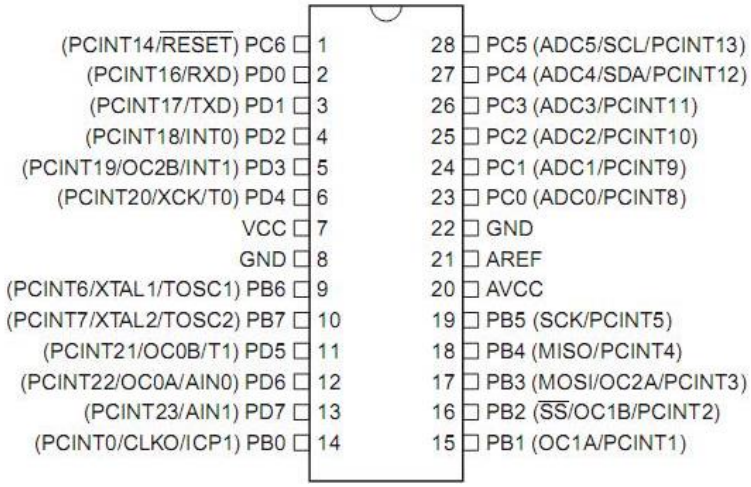
Spomedzi architektúr Atmel AVR ATmega, ARM, Intel 8051 a Intel Atom, ale aj ďalších je na realizáciu navrhovaného systému najvhodnejšia práve architektúra AVR ATmega, ktorá poskytuje dostatočný výkon pri nízkej spotrebe, ale najmä najpriaznivejšiu cenu. Architektúru AVR ATmega využíva niekoľko mikroprocesorov, ktoré sa rozlišujú najmä veľkosťami integrovaných pamätí a počtom V/V portov. Pri výbere konkrétnych modelov mikroprocesorov sme zohľadnili aj dostupnosť vývojových platforiem, ktoré sú na nich postavené. Pre externé moduly je postačujúci mikroprocesor Atmel ATmega 328, zatiaľ čo pre centrálnu jednotku s rozšírenými pamäťovými nárokmi je vhodný Atmel ATmega 2560. Obidva mikroprocesory sú založené na 8-bitovej AVR architektúre so zjednodušenou inštrukčnou sadou RISC. Rozlišujú sa najmä v spôsobe zapuzdrenia, veľkosti dostupnej pamäte a počtom vstupno-výstupných pinov. Atmel ATmega 328 je na vybranej vývojovej platforme použitý v PDIP balení, zatiaľ čo ATmega 2560 v TQFP balení.



Obr. 2-4 Atmel ATmega 328 v PDIP a ATmega 2560 v TQFP balení [11][12]

Atmel ATmega 328 [13]

Pre riadenie externých modulov postačuje aj mikroprocesor s menšou dostupnou pamäťou. Spomedzi všetkých modelov je najvhodnejší a najdostupnejší Atmel ATmega 328, ktorý disponuje 32 KB Flash pamäťou (z toho je 0,5 kB vyhradených pre Arduino bootloader), 2 kB SRAM a 1 kB EEPROM pamätí. Na komunikáciu s okolím slúži 23 digitálnych V/V pinov a 6 alebo 8 analógových vstupov podľa použitého balenia. K dispozícií je celkovo 6 PWM kanálov. Mikroprocesor obsahuje dvojicu 8-bitových a jedno 16-bitové počítadlo s časovačom. Pre našu prácu je nevyhnutný aj programovateľný watchdog časovač. Na komunikáciu s perifériami slúžia komunikačné rozhrania USART, SPI a I^2C. Maximálna taktovacia frekvencia je 20 MHz s výkonom 20 MIPS. Mikroprocesor obsahuje 6 úsporných režimov, ktoré môžeme využiť pri externých moduloch s napájaním na batériu.

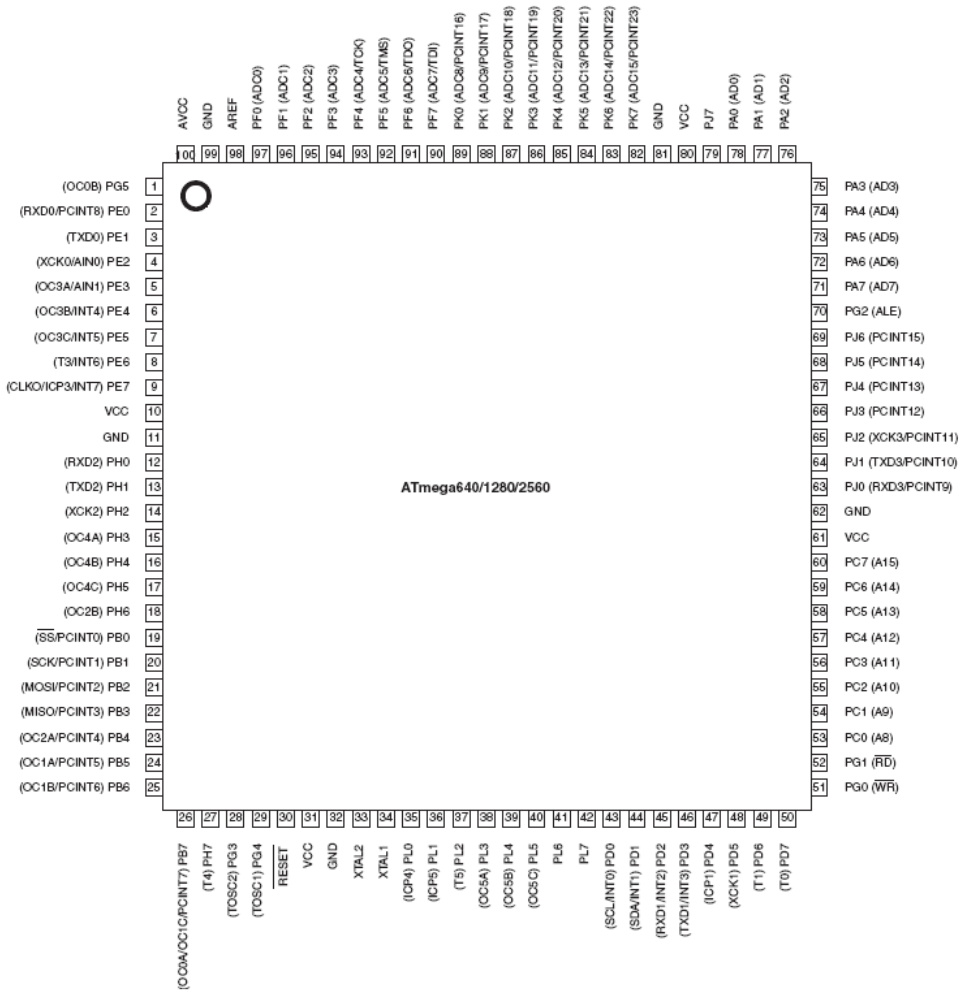


Obr. 2-5 Schéma zapojenia Atmel ATmega 328 v PDIP balení [14]

Atmel ATmega 2560 [15]

Pre centrálnu jednotku je nutný mikroprocesor s rozšírenou pamäťou. Výpočtový výkon pritom môže byť zachovaný, avšak predajná cena by taktiež mala byť čo najnižšia. Stanoveným kritériám vyhovuje mikroprocesor Atmel ATmega 2560, ktorý je vo väčšine technických parametrov len rozšírením mikroprocesora ATmega 328. Dostupná Flash pamäť narástla na 256 kB, z čoho má Arduino bootloader rezervovaných 8 kB. Pamäť SRAM je rozšírená na celkovú veľkosť 8 kB a pamäť EEPROM na 4 kB. Mikroprocesor obsahuje dvojicu 8-bitových a štvoricu 16-bitových počítadiel s časovačom. Na komunikáciu s okolím je vyhradených 86 V/V pinov, 16 PWM kanálov a 12 analógových vstupov. Na komunikáciu s okolím slúžia 4 USART, SPI a I^2C rozhrania. Pre možnosť rýchleho programovania a testovania obsahuje mikroprocesor aj JTAG rozhranie. Okrem toho má mikroprocesor k dispozícií aj programovateľný watchdog časovač. Maximálna taktovacia frekvencia je 16

MHz s výkonom 16 MIPS. Pre zníženie spotreby je k dispozícií 6 úsporných režimov. Výrobca poskytuje k mikroprocesoru knižnicu QTouch, ktorá umožňuje použitie kapacitných dotykových tlačidiel a iných dotykových prvkov. Atmel ATmega 2560 je k dispozícií v TQFP a CBGA baleniach.



Obr. 2-6 Schéma zapojenia Atmel ATmega 2560 v TQFP balení [16]

2.4.2 Výber vývojovej platformy

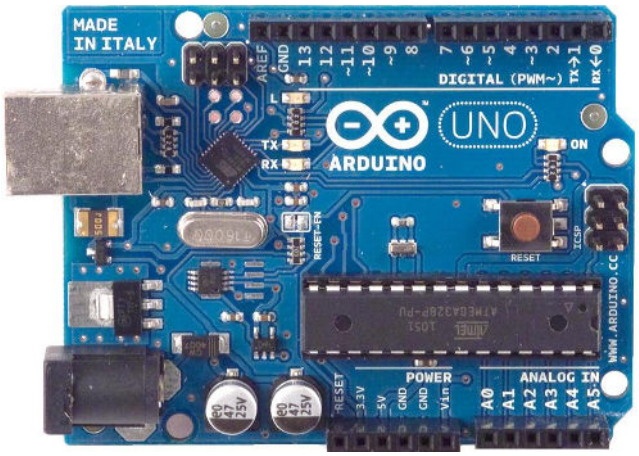
Na urýchlenie a zjednodušenie fázy prototypovania je vhodné použiť niektorú dostupnú vývojovú platformu, ktorá obsahuje okrem dosky plošných spojov s vyvedenými konektormi aj osadený mikroprocesor. Mikroprocesory sa vyrábajú v rôznych puzdrách a niektoré sú natoľko miniaturizované, že ich nie je možné prispájkovať voľnou rukou, ale len špecializovaným strojom. Okrem toho samotný návrh a výroba dosky plošných strojov zaberie nezanedbateľnú časť dostupných zdrojov. Najvhodnejšou praxou pri tvorbe prototypu je použitie niektorej z komerčne dostupnej vývojovej platformy a až v pred produkčnej fáze vykonať optimalizácie, ktoré znížia produkčné náklady a umožnia lepšiu miniaturizáciu. Diplomová práca sa zaoberá návrhom experimentálneho prototypu modulov a preto je

nanajvýš vhodné použiť niektorú z dostupných vývojových platforiem, ktorá obsahuje vybraný mikroprocesor. Medzi najznámejšiu a najvhodnejšiu platformu pre riešenie našej práce patrí Arduino [17]. Arduino je komerčne dostupná vývojová platforma, ktorá je vhodná na návrhy vlastných prototypov. Po kvalifikačnom otestovaní prototypu je následne možné zahájiť výrobu vlastných špecializovaných zariadení. Platforma Arduino nie je vhodná na priemyselné nasadenie a vďaka tomu si udržiava veľmi priaznivú cenu. Na programovanie je určené bezplatne dostupné opensource vývojové prostredie [18]. Výhodou prototypovacej platformy Arduino je aj malá veľkosť, energetická úspornosť a široké možnosti pripojenia rôznych periférnych zariadení. Základná funkcionalita sa dá jednoducho rozširovať pomocou doplnkových modulov (shield-ov).

Vývojová platforma Arduino obsahuje viacero modelov s vybranými mikroprocesormi. Najvhodnejšie modely sme vyberali najmä s ohľadom na dostupnosť a cenu. Pre externé moduly je vhodný Arduino UNO a pre centrálnu jednotku Arduino Mega 2560. Obidva modely dedia všetky vlastnosti osadených mikroprocesorov, avšak nevyužívajú všetky dostupné porty. Pre našu prácu to nie je až tak veľmi výrazné negatívum, pretože aj znížený počet portov postačuje na pripojenie všetkých špecifikovaných periférií aj s rezervou pre ďalšie rozširovanie v budúcnosti. Schéma zapojenia Arduino Uno a Arduino Mega 2560 sa nachádza v Prílohe A a Prílohe B na konci dokumentu.

Model vývojovej platformy	Arduino UNO
Mikroprocesor	ATmega 328
Pracovné napätie	5 V
Vstupné napätie	7 – 12 V
Maximálne napätie	6 – 20 V
Digitálnych V/V pinov	14 (z toho 6 podporuje PWM)
Analógových vstupov	6
Maximálny prúd na V/V pinoch	40 mA
Maximálny prúd na 3,3 V pinoch	50 mA
Flash pamäť	32 KB (0,5 KB má rezervovaný bootloader)
SRAM pamäť	2 kB
EEPROM pamäť	1 kB
Taktovacia frekvencia	16 MHz

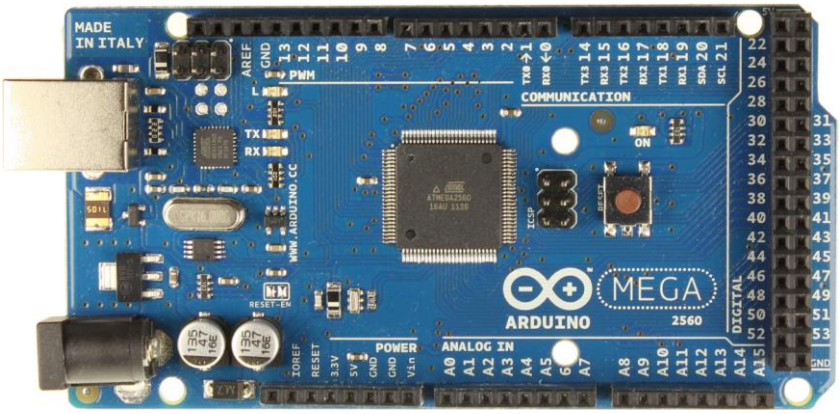
Tab. 2-4 Technické špecifikácie modelu Arduino UNO [19]



Obr. 2-7 Arduino UNO [19]

Model vývojovej platformy	Arduino Mega 2560
Mikroprocesor	ATmega 2560
Pracovné napätie	5 V
Vstupné napätie	7 – 12 V
Maximálne napätie	6 – 20 V
Digitálnych V/V pinov	54 (z toho 15 podporuje PWM)
Analógových vstupov	16
Maximálny prúd na V/V pinoch	40 mA
Maximálny prúd na 3,3 V pinoch	50 mA
Flash pamäť	256 KB (8 KB má rezervovaný bootloader)
SRAM pamäť	8 kB
EEPROM pamäť	4 kB
Taktovacia frekvencia	16 MHz

Tab. 2-5 Technické špecifikácie modelu Arduino Mega 2560 [20]



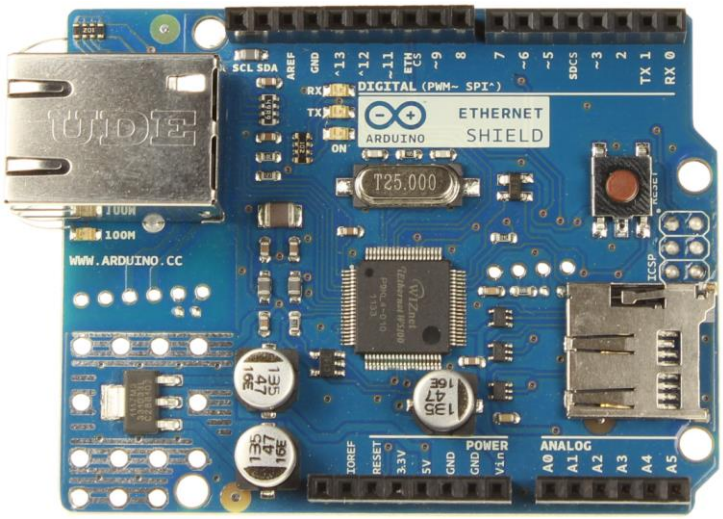
Obr. 2-8 Arduino Mega 2560 [20]

2.4.3 Sieťový shield

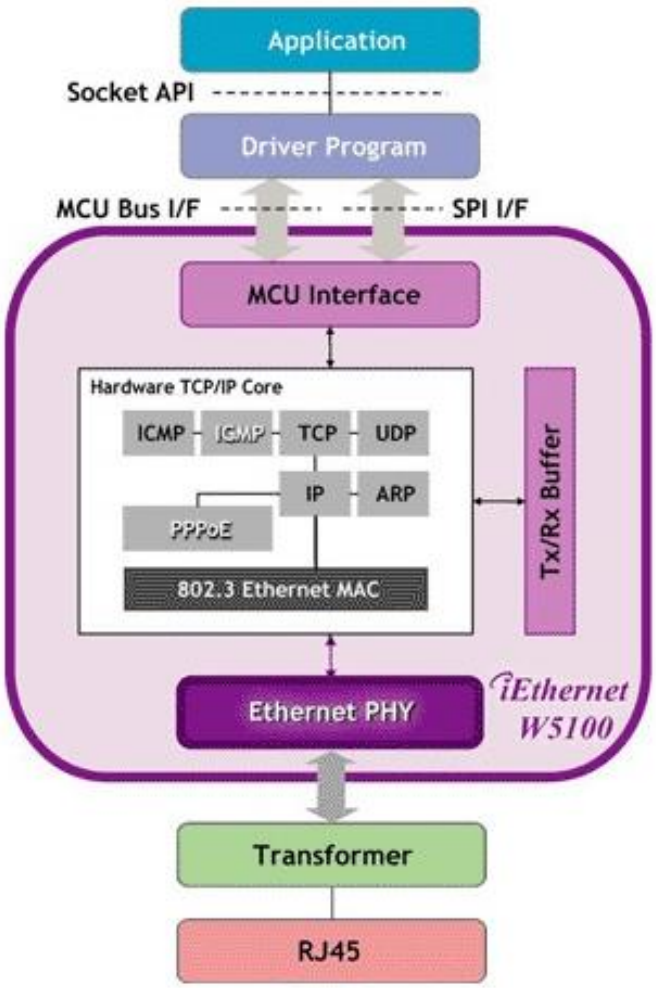
Vývojová platforma Arudino umožňuje priamo osadzovať ďalšie rozširovacie moduly (Arduino shield), ktoré rozširujú dostupnú funkcionalitu. Arduino shield-y vyrábajú viaceré spoločnosti, ktoré okrem samotnej hardvérovej časti ponúkajú aj softvérové riešenia v podobe knižníc, ktoré sa dajú do vytvoreného projektu jednoducho pridať. Takéto knižnice poskytujú API rozhranie pre jednoduché ovládanie pridaného hardvérového modulu.

Niektoré Arduino modely obsahujú aj sieťové komunikačné rozhranie RJ-45, avšak sú horšie dostupné a aj drahšie. Priamo na fakulte máme dostupných niekoľko vývojových platforiem Arduino, avšak žiadna z nich neobsahuje sieťový port, ktorý je nevyhnutný pre vzájomnú komunikáciu modulov. Modul so sieťovým rozhraním a pridruženým riadiacim obvodom sa dá zakúpiť ako Ethernet Arduino shield. Ten sa pripojí k existujúcej platforme Arduino a po implementovaní softvérových knižníc je možné priamo využívať API rozhrania a komunikovať s ostatnými zariadeniami. Takýto prístup je pre realizáciu našej práce výhodnejší ako nákup ďalších vývojových platforiem so sieťovým rozhraním.

Na trhu je dostupný relatívne veľký počet rôznych sieťových shield-ov, ktoré používajú rovnaké ale aj rôzne sieťové riadiace obvody. Tieto obvody sa pripájajú k hlavnému mikroprocesoru a sprostredkovávajú mu sieťovú konektivitu. Odlišné sieťové obvody používajú rôzne API rozhrania a preto nie sú väčšinou vzájomne kompatibilné. V sieťových shield-och je najrozšírenejší riadiaci obvod WIZnet W5100 [21]. Práve na tento obvod je prispôsobená aj väčšina z dostupnej programovej výbavy a preto je pri kúpe sieťového shield-u vhodné siahnuť práve po modeli s týmto obvodom. Schéma zapojenia sieťového shield-u sa nachádza v prílohe C.



Obr. 2-9 Arduino Ethernet Shield [22]



Obr. 2-10 Blokový diagram obvodu WIZnet W5100 [23]

Riadiaci obvod WIZnet W5100 podporuje 10BaseT/100BaseTX sieťové rozhranie s automatickým nastavením komunikačnej rýchlosti a komunikačných pinov podľa použitej kabeláže. Riadiaci sieťový obvod komunikuje s mikroprocesorom pomocou sériovej SPI linky. Jeho nevýhodou je neschopnosť spracovania fragmentovaných paketov a obmedzená vyrovnávacia pamäť s veľkosťou 16 kB, ktorá slúži na uskladnenie prijatých a odosielaných rámcov. Táto pamäť môže pretiecť v prípade vysokého zahltenia siete, čo môže viesť k strate údajov. Rôznym obmedzeniam sieťovej komunikácie je venovaná samostatná kapitola, ktorá pojednáva o problémoch sieťových shield-ov a navrhuje spôsoby riešenia.

Blokový diagram obvodu WIZnet W5100 zobrazuje rozloženie funkčných blokov a ich vzájomnú komunikáciu. Na sieťovom rozhraní sú prijímané rámce, ktoré sa pretransferujú do kompatibilného formátu určeného na ďalšie na spracovanie. Prijaté rámce sa následne uložia do vyrovnávacej pamäti, kde čakajú na spracovanie. Obvod dokáže spracovať protokoly TCP, UDP, ICMP, IPv4 ARP, IGMP a PPPoE. Spracovaný paket sa následne

prepošle hlavnému riadiacemu mikroprocesoru, ktorý pomocou API rozhrania dokáže priamo pristupovať k požadovaným dátam. Jedným z cieľov tejto práce je navrhnúť vlastný efektívny komunikačný protokol, ktorý sa nachádza na vyšších vrstvách a spracováva ho tak riadiaci mikroprocesor. Komunikačná knižnica pre platformu Arduino je integrovaná priamo vo vývojovom prostredí.

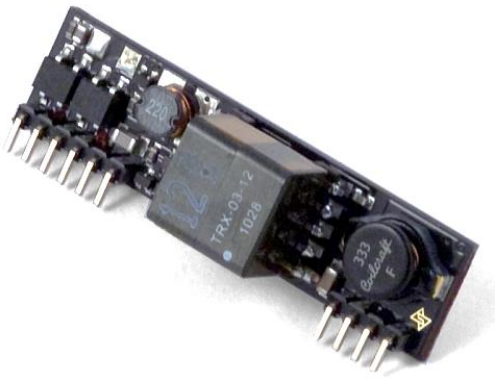
2.4.4 Zaznamenávanie údajov

Namerané hodnoty zo senzorov, ako aj aktuálne nastavenia jednotlivých modulov musia byť niekde uchovávané. Externé moduly sa z dôvodu redukcie nákladov budú spoliehať len na vstavanú EEPROM pamäť, ktorá dokáže uchovať 1 kB dát. Aj keď to nie je mnoho, tak pre špecifikované použitie to postačuje, pretože všetky namerané hodnoty bude uchovávať centrálna jednotka. Centrálna jednotka disponuje EEPROM pamäťou s veľkosťou 4 kB, avšak ani táto veľkosť nedostačuje pre uchovávanie všetkých nameraných hodnôt a preto je nutné použiť alternatívny spôsob ich uchovávania. Súčasťou niektorých sieťových shield-ov je aj konektor na pripojenie microSD pamäťových kariet. Takéto pamäťové karty poskytujú ideálny pomer medzi kapacitou a cenou. Predstavujú tak vhodné riešenie pre zaznamenávanie všetkých nameraných hodnôt aj s bohatou históriou. Pamäťové SD karty najčastejšie používajú súborový systém FAT, ktorý je svojou jednoduchosťou vhodný aj pre použitie vo vnorených systémoch. Na prístup k údajom bude musieť použiť niektorú z dostupných knižníc, ktorá umožňujú priame spracovávanie uchovávaných dát. Namerané údaje budeme uchovávať na SD karte, zatiaľ čo nastavenia v EEPROM pamäti.

2.4.5 Napájanie

Vývojovú platformu Arduino je možné napájať aj programovať priamo z USB portu, ktorý poskytuje 5 V napätie. Mikroprocesor pracuje s rovnakým napätím, avšak pre zvyšné periférie je vyhradený ďalší napäťový stabilizátor s výstupným napätím 3,3 V. Pri reálnom nasadení ale nie je možné z dôvodu zvýšenej spotreby napájať všetky moduly pomocou USB rozhrania a napájanie je možné zabezpečiť aj pomocou štandardného napájacieho zdroja s výstupným napätím 7 až 12 V. Vstupné napätie je následne stabilizované na štandardných 5 V. Arduino obsahuje logiku, ktorá automaticky prepína napájací vstup a tým zabraňuje vzájomným interferenciám viacerých napájacích zdrojov. Obidva spôsoby napájania nie sú veľmi vhodné na použitie v externých moduloch, avšak Ethernet shield umožňuje použiť Power Over Ethernet (POE) napájanie, ktoré je založené na štandarde IEEE 802.3af. K modulom tak nie je nutné okrem sieťového kábla pripájať žiadne ďalšie napäťové zdroje. Napájanie je realizované priamo pomocou sieťového kábla, pričom napäťový zdroj sa nachádza

v sieťovom prepínači. Štandardný sieťový shield nepodporuje napájanie pomocou POE, avšak na trhu sú dostupné rôzne POE moduly, ktoré sa pripájajú k sieťovým shield-om a sprístupňujú takúto funkcionalitu. Sieťový shield ale musí umožňovať pripojenie takéhoto modulu.



Obr. 2-11 Arduino Ethernet PoE modul [24]

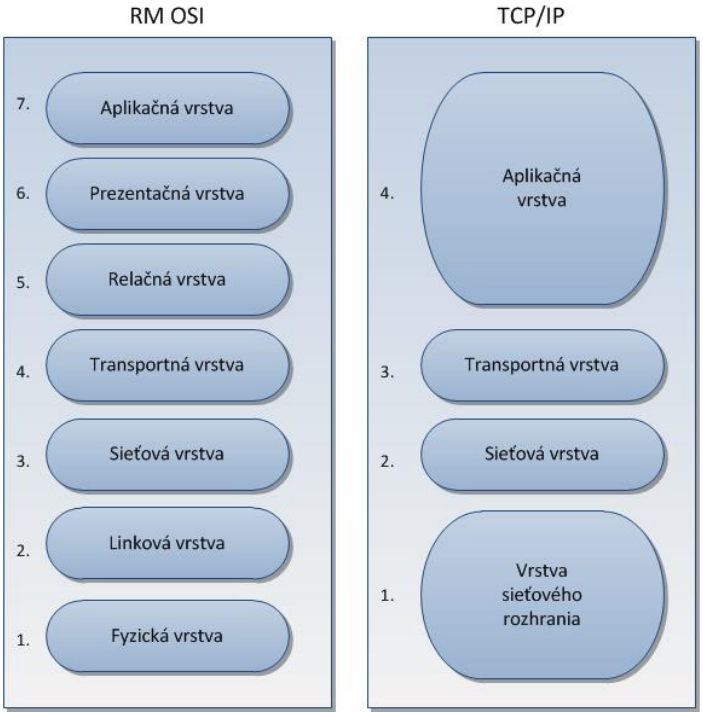
Napájanie pomocou POE nie je dostatočne bezpečné, pretože v prípade výpadku sieťového prepínača stratia všetky moduly napájanie. Toto zvýšené riziko znefunkčnenia celého systému (*Single Point of Failure*) sa dá obmedziť záložným zdrojom. Ako najvhodnejšie riešenie sa ukazuje použitie záložnej batérie, ktorá sa automaticky aktivuje v prípade výpadku napájania. Bezdrôtové moduly nemajú prístup k POE a preto budú napájané len z vlastnej batérie alebo iného zdroja. Aby bola zabezpečená nepretržitá činnosť, tak takéto moduly by mali byť vybavené ďalšími zdrojmi elektrickej energie, akými sú napríklad solárne články. Okrem toho musia takéto moduly dbať aj na minimalizovanie spotreby, ktoré je možné realizovať pomocou úsporných režimov mikroprocesora.

2.5 Komunikačné sieťové protokoly

Komunikácia medzi externými modulmi, centrálnou jednotkou a používateľom bude prebiehať cez počítačovú sieť typu Ethernet IEEE 802.3. Niektoré externé moduly sa budú viazať na bezdrôtovú komunikačnú sieť Wi-Fi IEEE 802.11. Ethernet je súperiaca počítačová sieť s prepínaním paketov, ktorá môže byť hviezdicovej alebo zbernicovej topológie. V súčasnosti je najpoužívanejšia hviezdicová topológia, ktorá pre každé pripojené zariadenie zaručuje vlastný sieťový segment. Vďaka tomu sú všetky zariadenia oddelené, čo zabraňuje vzniku komunikačných kolízií. Pre broadcastovú sieť Wi-Fi to ale neplatí, pretože zdieľa jedno spoločné komunikačné médium. Okrem vznikajúcich kolízií je potrebné zaistiť aj zvýšenú bezpečnosť, pretože komunikáciu môže zachytiť každé zariadenie v dosahu. Siete Wi-Fi poskytujú vlastné spôsoby šifrovania, avšak v našej práci počítame aj s vlastným doplnkovým šifrovaním na úrovni navrhovaného protokolu.

2.5.1 Sieťový protokolový zásobník

Sieťová architektúra je podľa funkcionality rozdelená na vrstvový model. Každá vrstva poskytuje určitú funkcionalitu pre vyššie vrstvy, zatiaľ čo sa spolieha na nižšie vrstvy. Systém, ktorý implementuje tieto vrstvy do jedného celku sa nazýva protokolový zásobník. Na nasledujúcom obrázku sú predstavené dva protokolové zásobníky.



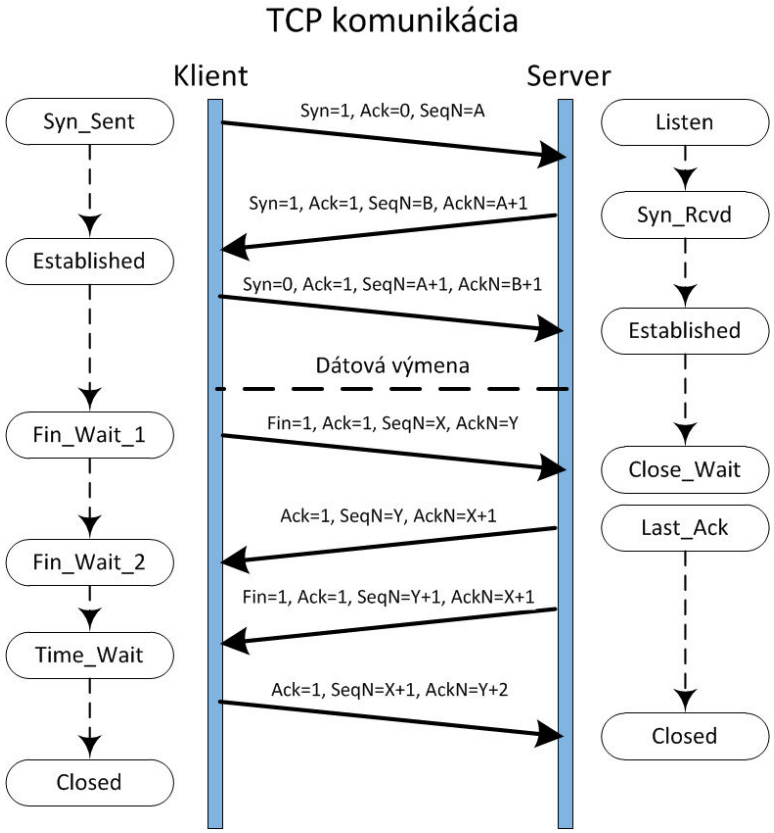
Obr. 2-12 Vrstvy protokolových zásobníkov RM OSI a TCP/IP

Zatiaľ čo RM OSI predstavuje abstraktný referenčný model, tak protokolový zásobník TCP/IP je v súčasnosti najrozšírenejší. Skladá sa zo štyroch vrstiev, kde každá vrstva poskytuje určitú funkcionalitu. Tieto vrstvy sa pri prenose dát postupne zvrchu vzájomne zapuzdrujú, pričom cieľové zariadenie postupuje opačným poradím a dátový rámec sa postupne rozkladá až na aplikačnú vrstvu s požadovanými údajmi. Protokolový zásobník môže byť okrem prvej vrstvy implementovaný softvérovo, hardvérovo alebo ich kombináciou. Sieťový obvod WIZnet W5100 spracúva prvé tri vrstvy TCP/IP zásobníka hardvérovo, takže na implementáciu zostáva len efektívne navrhnutie vlastného protokolu na aplikačnej vrstve, ktorý bude poskytovať požadovanú funkcionalitu pre všetky moduly v systéme. Okrem toho si musíme stanoviť, ktoré protokoly a na ktorej vrstve budeme pri komunikácií používať. Vrstva sieťového rozhrania je daná fyzickým pripojením, kde použijeme siete Ethernet alebo Wi-Fi. Štandardne dostupné komunikačné moduly podporujú na druhej vrstve iba sieťový protokol IP, avšak to nepredstavuje problém, keďže tento protokol je v súčasnosti takmer exkluzívne používaný v týchto typoch sieťach, pričom zvyšné protokoly sú používané len v špecifických

prípadoch. Na tretej transportnej vrstve môžeme použiť protokoly TCP a UDP. Transportný protokol TCP je robustnejší a spojovo orientovaný a preto vyžaduje nadviazanie komunikácie pred začiatkom výmeny údajov. Protokol UDP je menší a rýchlejší bez nadväzovania spojenia, avšak môže dochádzať k strate a duplikácií paketov, alebo iným nežiadaným javom. Obidva protokoly sú bližšie opísané v samostatných kapitolách.

2.5.2 Transmission Control Protocol

TCP protokol je spojovo orientovaný sieťový protokol na transportnej vrstve protokolového zásobníka. Výhodou TCP protokolu je jeho spoľahlivosť, ktorá je docielená nadväzovaním spojenia, potvrdzovaním príjmu, retransmisiou pri neúspešnom prenose a dynamickým škálovaním veľkosti prenosového okna pre maximalizáciu efektivity prenosu. Pred prenosom údajov sa musí klient pripojiť na server, ktorý počúva na nastavenom porte a inicializovať spojenie (*3-way handshake*). Po inicializovaní spojenia môže nastať vzájomná dátová výmena, pri ktorej je každý paket očíslovaný, čo zabraňuje prehádzaniu, strate a duplikácií paketov. Každý paket okrem toho obsahuje aj kontrolné súčty pre detekciu prenosových chýb.



Obr. 2-13 Schéma inicializácie a ukončenia TCP komunikácie [25]

Ukončenie spojenia je realizované pomocou *4-way handshake* alebo *skrátenej 3-way handshake* výmeny. Spojenie sa musí ukončiť na obidvoch stranách nezávisle. Schéma

nadviazania a ukončenia spojenia je naznačená na predchádzajúcom obrázku. TCP protokol je pre nadviazanie spoľahlivej dátovej výmeny medzi externými modulmi a centrálnou jednotkou ideálny, avšak prináša mnohé implementačné problémy, ktoré vychádzajú z obmedzení použitého hardvéru.

2.5.3 User Datagram Protocol

Transportný protokol UDP nie je na rozdiel od TCP spojovo orientovaný, čo umožňuje jednoduchšiu a rýchlejšiu výmenu údajov. Prenos a ani doručenie paketov nie je zaručené, čo predstavuje problém najmä v preťažených sieťach, kde môže dochádzať k ich zahadzovaniu. Výhodou UDP protokolu je jeho minimálna veľkosť, jednoduchosť a rýchlosť, vďaka čomu je vhodný na prenos časovo citlivých údajov. Súčasťou protokolu je aj kontrolný súčet, ktorý zahodí poškodené pakety a zaručí korektnosť prijatých dát. Keďže UDP protokol nevyžaduje nadväzovanie spojenia so serverom, tak je vhodný na rýchlu výmenu správ a na posielanie broadcast-ov z vyšších vrstiev, ktoré budeme používať v navrhovanom systéme. Protokol UDP využíva aj viacero ďalších služieb, ktoré budeme v práci používať.

Broadcasty

Sieťové protokoly používajú adresovanie, ktoré zaručuje doručenie údajov adresátovi. Niekedy je ale nevyhnutné poslať správu viacerým alebo všetkým uzlom. V takom prípade sa používa multicast-ová alebo broadcast-ová komunikácia. Zatiaľ čo v prípade multicast-u sa správa zašle len vybranej skupine, tak broadcast správu príjmu všetky uzly v broadcast-ovej doméne. Broadcast-ová doména združuje všetky zariadenia so spoločným adresovaním na sieťovej vrstve, ktoré patria pod jeden IP subnet. Pre broadcast-ovú adresu je zakaždým vyhradená posledná IP adresa z daného rozsahu (napríklad 192.168.2.255 s maskou siete 255.255.255.0) a adresa FF:FF:FF:FF:FF:FF na vrstve sieťového rozhrania. V našej práci budeme používať broadcast-ové správy pri detekcií všetkých dostupných externých modulov. Do siete sa tak vyšle broadcast-ová správa, na ktorú odpovedia všetky dostupné moduly.

NTP

Network Time Protocol je sieťový protokol slúžiaci na synchronizáciu aktuálneho času medzi sieťovými zariadeniami. Vývojová platforma Arduino neobsahuje obvody reálneho času, avšak špecifikácia vyžaduje, aby jednotlivé záznamy obsahovali časovú pečiatku. Existujú dva odlišné spôsoby riešenia. Buď ku každému modulu doplniť obvod reálneho času, alebo implementovať NTP protokol. Druhá možnosť je aj s ohľadom na existujúcu sieťovú infraštruktúru vhodnejšia. Aj v prípade, že by lokálne moduly nemali prístup na internet, tak nie je problém do uzavretej siete pridať vlastný NTP server, ktorý by poskytoval presný čas.

NTP rámec obsahuje 64-bitový časový identifikátor, kde prvých 32-bitov určuje aktuálnu sekundu a ďalších 32-bitov frakciu sekundy. Takéto rozdelenie umožňuje teoretickú presnosť na úrovni 2^{-32} sekundy (233 pikosekúnd), pričom v praxi sa presnosť pohybuje okolo 1 milisekundy. Na identifikáciu sekúnd sa používa 32-bitová hodnota, takže počítadlo pretečie každých 136 rokov. Za počiatok časového intervalu sa používa rok 1900 a preto dôjde k najbližšej kritickej situácií v roku 2036. NTP používa na komunikáciu transportný protokol UDP na porte 123.

Wake on LAN

Štandard Wake on LAN umožňuje prebudiť sieťové zariadenia z režimu spánku. Aby mohlo byť zariadenie prebudené, tak musí podporovať tento štandard a musíme poznať jeho MAC adresu. Prebudenie je inicializované po prijatí správy, ktorá sa nazýva Magic Packet. Táto správa sa posiela ako broadcast na linkovej vrstve. Magic Packet obsahuje na ľubovoľnom mieste v dátovej časti rámcu 6 bajtov FF:FF:FF:FF:FF:FF, ktoré nasleduje 16-krát zopakovaná MAC adresa cieľového zariadenia. Minimálna dĺžka dátovej časti Magic Packetu je tak 102 bajtov, avšak tento rámec môže byť ľubovoľne obalený ďalšími dátami alebo vrstvami. V praxi sa tak na WOL bežne používa UDP protokol, ktorý je smerovaný na port 7 alebo 9. Napriek tomu sa po prijatí rámca rozbaľovanie nedostane až po transportnú vrstvu, pretože dátová časť Magic Packet-u je hardvérovo detegovaná už na vrstve sieťového rozhrania.

2.6 Komunikácia bezdrôtových modulov

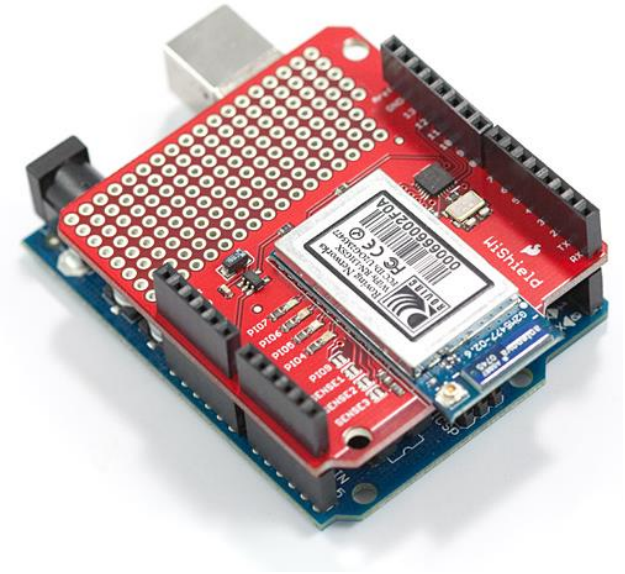
Niektoré externé moduly nemôžu byť z dôvodu fyzického obmedzenia sieťovej kabeláže priamo pripojené na sieť. V takom prípade je vhodné zabezpečiť komunikáciu pomocou niektorej bezdrôtovej technológie zabezpečujúcej výmenu údajov. V analyzovaných diplomových prácach [2] a [3] boli bližšie popisované komunikačné technológie Zigbee, Bluetooth a Wi-Fi. Ku všetkým technológiám bolo poskytnuté zhrnutie ich výhod a nevýhod. Z dôvodu špecifikácie práce a aktívneho využívania sieťovej komunikácie sú pre našu prácu technológie Zigbee a Bluetooth nevyhovujúce. Obidve technológie vyžadujú pre sieťovú komunikáciu použitie prekladača, ktorý by prekladal sieťové rámce na kompatibilnú komunikačnú formu danej technológie. Bezdrôtová technológia by tak vlastne fungovala iba ako transparentný prenosový kanál. Takýto prekladač by v našom prípade zbytočne komplikoval návrh a realizáciu práce, pričom by porušoval princíp modulárnosti a autonómnosti všetkých modulov. V rámci zachovania sieťového modelu komunikácie

a celkovej modularity sme sa rozhodli používať sieť typu Wi-Fi, ktorá je značne rozšírená v domácnostiach a preto nie je nevyhnutné vybudovávať novú komunikačnú sieť. Technológia Wi-Fi prináša aj niektoré nevýhody, medzi ktoré patrí relatívne vysoká spotreba elektrickej energie, ktorá je kritická pri napájaní na batérie. Výberom vhodného bezdrôtového adaptéra je možné tento nedostatok výrazne minimalizovať.

2.6.1 WiFly Arduino shield

Pre Arduino sú podobne ako v prípade sieťových shield-ov dostupné aj rôzne Wi-Fi shield-y, ktoré umožňujú pripojenie na existujúcu bezdrôtovú infraštruktúru. Jednotlivé Wi-Fi shield-y sa rozlišujú najmä použitými bezdrôtovými modulmi, ktoré majú rôzne vlastnosti a komunikačné parametre. V našom regióne nie je dostupných veľa Wi-Fi shield-ov a preto sme sa pre našu prácu rozhodli použiť jeden z najrozšírenejších shield-ov, ktorý vyrába spoločnosť Sparkfun [26].

Sparkfun Wifly shield používa bezdrôtový adaptér RN-131G od spoločnosti Roving Networks [27]. Adaptér je kompatibilný so sieťami 802.11b a 802.11g a širokým spektrom šifrovacích a autentifikačných algoritmov (WEP-128, WPA-PSK (TKIP), WPA2-PSK a EAP-TLS pre WPA1 & WPA2). Modul obsahuje vstavanú anténu, avšak pomocou u.FL konektora je možné pripojiť aj výkonnejšiu externú anténu. Bezdrôtový modul má veľmi nízku spotrebu energie. Pri 3,3 V napájaní spotrebúva v režime spánku len 4 uA, pri prijímaní 40 mA a pri maximálnom zaťažení 212 mA. So sieťových funkcií sú podporované protokoly DHCP, DNS, ARP, ICMP, NTP, Telnet a FTP. Komunikácia môže byť realizovaná prostredníctvom protokolov UDP aj TCP, avšak súčasne môže byť aktívny len jeden a prepnutie je realizované až po resete. Bezdrôtový modul RN-131G komunikuje s riadiacim mikroprocesorom pomocou UART rozhrania. WiFly shield obsahuje okrem bezdrôtového modulu aj mikroprocesor SC16IS750, ktorý slúži na preklad rozhraní SPI na UART. Komunikácia s Arduinom tak prebieha pomocou rozhrania SPI a UART rozhranie zostáva voľné na programovanie a ďalšie využitie. Na dátovú výmenu, ale aj konfiguráciu sa tak používa rovnaké rozhranie, čo prináša niekoľko obmedzení. Okrem iného nie je možné súčasné nastavovať komunikačné parametre a prenášať užitočné dáta. Modul poskytuje len dáta z aplikačnej sieťovej vrstvy, z ktorej nie je možné určiť odosielateľa a iné komunikačné parametre. Tieto nedokonalosti daného riešenia budeme musieť odstrániť vhodným návrhom komunikačných protokolov a rozhraní. Na platforme Arduino existuje pre WiFly shield množstvo knižníc od rôznych komunít. Výberom najvhodnejšej knižnice sa budeme zaoberať v časti s riešením. Schéma zapojenia bezdrôtového shield-u sa nachádza v prílohe D.



Obr. 2-14 Sparkfun Arduino WiFly shield [26]

2.7 Spôsoby zabezpečenia

Jedným z hlavných cieľov tejto diplomovej práce je vytvoriť bezpečný systém, ktorý by bolo možné nasadiť do reálnej domácnosti. Z tohto dôvodu musí poskytovať rozšírené možnosti zabezpečenia, ktoré zabránia prieniku neoprávneného používateľa s cieľom pozmenenia správania sa systému. Úroveň zabezpečenia musí byť dodržaná na viacerých vrstvách. Musí byť zabezpečená interná komunikácia medzi modulmi, ale aj vonkajšia linka do internetu, ktorá poskytuje používateľské rozhranie. Okrem toho by každý modul mal obsahovať mechanizmus na bezpečné reštartovanie v prípade neočakávanej akcie. Všetky tieto aspekty sú popísané v nasledujúcich kapitolách.

2.7.1 Šifrovanie sieťovej komunikácie

Sieťová a najmä bezdrôtová komunikácia je náchylná na odpočúvanie a duplikáciu rámcov, čo môže predstavovať pre celý systém výraznú hrozbu. V prípade že útočník zistí formát komunikačných správ a podarí sa mu preniknúť do siete, tak môže pozmeniť správanie sa akčných členov, čo môže mať katastrofálne účinky. Azda najväčšie riziko hrozí v prípade neoprávneného otvorenia bezpečnostných dverí, alebo iného ovplyvnenia správania sa systému. Týmto negatívnym javom sa dá predísť pomocou kompletného šifrovania všetkých komunikačných správ.

Podrobnejšia analýza šifrovacích algoritmov sa nachádza v predchádzajúcej bakalárskej práci [10]. Požiadavky na použitý šifrovací algoritmus zostávajú nezmenené. Výpočtový výkon vybraných mikroprocesorov nie je vysoký a preto je vhodné použiť veľmi rýchly a jednoduchý šifrovací algoritmus. Z tohto dôvodu môžeme ihneď vylúčiť všetky

asymetrické šifrovacie algoritmy, ktoré nie sú na použitie v jednotnom systéme vhodné a sú aj výpočtovo náročnejšie. Zo symetrických šifrovacích algoritmov je pre použitie v navrhovanom systéme najvhodnejšia bloková šifra XTEA, ktorá na rozdiel od prúdových šifier spracováva dáta po blokoch pevnej šírky, vďaka čomu nemôže útočník určiť ani celkovú dĺžku užitočnej informácie.

Šifrovací algoritmus XTEA

Šifrovací algoritmus XTEA (*eXtended TEA*) je založený na staršom algoritme TEA (*Tiny Encryption Algorithm*). Tvorcovia algoritmu si stanovili za cieľ vyvinúť veľmi rýchlu šifru, ktorá by bola vhodná na použitie vo vnorených systémoch s obmedzenými hardvérovými prostriedkami. Algoritmus XTEA patrí medzi blokové šifry so šírkou 64 bitov. K šifrovaniu sa používa tajný zdieľaný kľúč o veľkosti 128 bitov. Výhodou algoritmu je relatívne vysoký stupeň odolnosti, ktorý postačuje na zabezpečenie navrhovaného systému. Vysoká rýchlosť šifrovacieho algoritmu spočíva z krátkeho zdrojového kódu, ktorý používa iba základné inštrukcie sčítania, funkcie XOR a bitových posunov. Navyše v každom kroku sa spracúva len polovica zo 64-bitového bloku dát.

Šifrovací algoritmus XTEA je dostupný bez patentovej záťaže a je voľne dostupný. Pre vývojovú platformu Arduino je priamo dostupná knižnica implementujúca šifrovacie a dešifrovacie funkcie zvoleného algoritmu [28].

2.7.2 Zabezpečenie web servera

Používateľské rozhranie bude tvorené web serverom. K rozhraniu sa bude dať pripojiť nielen z intranetu, ale po nastavení presmerovania portov na smerovači aj z internetu. Otvorenie komunikačného rozhrania smerom do internetu môže priniesť útoky na systém a preto by mal byť voči takýmto útokom zabezpečený. Existuje viacero prístupov a realizácií zabezpečeného web servera, avšak nie všetky prístupy sa dajú použiť na obmedzených hardvérových prostriedkoch vnorených systémov. Najefektívnejší spôsob ponúka implementácia zabezpečeného HTTPS protokolu, ktorý šifruje komunikáciu pomocou asymetrických šifier SSL alebo TLS. Na realizáciu asymetrického šifrovania ale vybraný mikroprocesor neposkytuje dostatočné hardvérové prostriedky. Aspoň čiastočným riešením je požadovanie používateľskej autentifikácie pomocou HTTP protokolu. Prihlasovacie údaje sú kódované algoritmom Base64 [29], ktorý je relatívne ľahko prelomiteľný. Útočníkovi prakticky postačuje sledovať sieťovú komunikáciu, aby bol schopný napadnúť systém. Z tohto dôvodu je nutné rozšíriť zabezpečenie o dodatočnú autentifikáciu kritických operácií. Autentifikácia

môže prebiehať pomocou GRID karty, z ktorej by používateľ zadával identifikačné znaky. Takéto riešenie je ale nepohodlné pre používateľa a v prípade dlhodobého procesu sledovania môže útočník získať všetky bezpečnostné políčka z tejto karty. Omnoho praktickejším a bezpečnejším je autentifikácia pomocou SMS správ, ktoré sa posielajú cez GSM sieť. Pred vykonaním kritickej operácie je vygenerovaný autentifikačný kód, ktorý je zaslaný na používateľský mobilný telefón. Používateľ následne zadá autentifikačný kód, čím je overená oprávnenosť operácie. Na prelomenie tohto spôsobu zabezpečenia by útočník musel ovládnuť nielen internetovú sieť, ale aj mobilnú komunikačnú sieť, čo je veľmi nepravdepodobné. Na vývojovú platformu Arduino je možné pomocou rozhraní UART alebo SPI pripojiť GSM modul, vďaka ktorému je možné zrealizovať tento spôsob autentifikácie.

GSM modul

GSM a GPRS modul SIM900 od spoločnosti SIMcom [30] umožňuje pripojenie k sieťam GSM 850, EGSM 900, DCS 1800 a PCS 1900. Umožňuje nielen zasielanie SMS správ, GSM hlasové hovory, ale aj vytvorenie TCP/IP spojenia cez GPRS sieť. Ovládanie modulu prebieha pomocou GSM AT príkazov, avšak na zjednodušenie implementácie sú dostupné knižnice [31], ktoré efektívne zabaľujú takúto komunikáciu do jednoduchšej formy. Na prepojenie s vývojovou platformou Arduino sme zvolili shield od spoločnosti IComsat [32], ktorý používa na komunikáciu sériové UART rozhranie. Shield obsahuje okrem portu na SIM kartu aj dvojicu rôznych konektorov na anténu a vstup na mikrofón spoločne s výstupom na reproduktory v prípade hlasového hovoru. Vstupné komunikačné napätie sa pohybuje v rozsahu 4,5 až 5,5 V, čo vyhovuje pre priame použitie s platformou Arduino. Shield spotrebúva pri maximálnej záťaži v špičkách až 2 A a preto je nutné zabezpečiť adekvátne napájanie. GSM modul bude pripojený k centrálnej jednotke s POE napájaním a tak zvýšená spotreba nepredstavuje problém. Schéma GSM modulu sa nachádza v prílohe E.

Obr. 2-15 ICOMSAT Arduino GSM shield [32]

2.7.3 Watchdog časovač

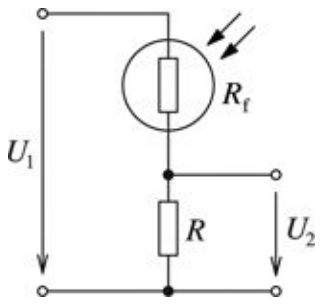
Vybraté mikroprocesory obsahujú programovateľný Watchdog časovač, ktorý môže byť využitý viacerými spôsobmi. Buď sa použije na zobudenie zariadenia z režimu spánku, aby vykonalo požadovanú funkcionalitu alebo na reštartovanie modulu v prípade neočakávaného správania. Pomocou úsporného režimu mikroprocesora a Watchdog časovača je pri voľbe vhodného časovania možné ušetriť až 99% spotrebovávanej energie, čo sa výrazne prejaví na životnosti batérie. Watchdog časovač musí byť resetovaný v pravidelných intervaloch. Ak sa z nejakého dôvodu časovač nestihne obnoviť a pretečie stanovenú hodnotu, tak sa vyšle resetovací signál na oživenie zariadenia. Watchdog časovač je napojený na vlastný 128 kHz oscilátor, pričom najpoužívanejšie časové limity sú 15, 30, 60, 120, 250, 500, 1000, 2000, 4000 a maximálnych 8000 ms. Pri výbere časového limitu je mimoriadne dôležité stanoviť jeho optimálnu hodnotu, aby nedochádzalo k resetovaniu aj pri vykonávaní platných operácií. Časový limit vychádza z časovej náročnosti všetkých operácií, pričom je odporúčané pripočítať ešte 50% časovú rezervu.

2.8 Monitorovanie prostredia

Cieľom diplomovej práce nie je predstaviť riadiaci systém s obrovským množstvom senzorov, ale vyvinúť prototyp, ktorý by v dostatočnej miere reprezentoval schopnosti navrhnutého systému. Z tohto dôvodu je vhodné použiť senzory s rôznorodými rozhraniami, ktoré efektívne znázornia možnosti navrhovaného systému. Pre demonštráciu možností sme sa rozhodli použiť analógový senzor osvetlenia, digitálny teplomer a sériovú RFID čítačku.

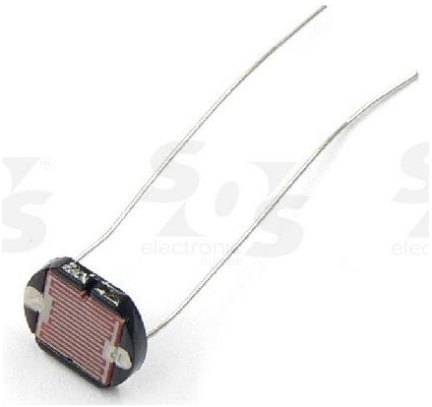
2.8.1 Fotorezistor

Na meranie intenzity slnečného žiarenia využijeme fotorezistor, čo je pasívna analógová súčiastka, ktorej odpor klesá s rastúcou intenzitou dopadajúceho svetla. Na načítavanie hodnôt sa používajú ADC prevodníky. Vybraný mikroprocesor obsahuje 10-bitové prevodníky, čo umožňuje rozlišovať 1024 rôznych intenzít osvetlenia. Zapojenie fotorezistora sa nachádza na nasledujúcom obrázku.



Obr. 2-16 Schéma zapojenia fotorezistora

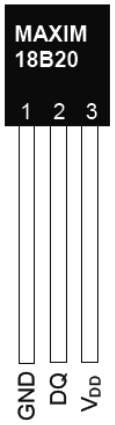
Pri zapojení je nutné brať ohľad na maximálne možné zaťaženie fotorezistora, s čím súvisí výber vhodne veľkého odporu do napäťového deliča. Výsledná meraná hodnota zodpovedá napätiu U_2. Spomedzi dostupných fotorezistorov je pre realizáciu našej práce najvhodnejší model FW300 od výrobcu PERKIN ELMER [33]. Aj keď je fotorezistor určený až pre napätie do 300 V, tak v našej práci budeme využívať napätie o dva rády nižšie, čo nevadí, lebo sledujeme zmenu odporu, na ktorý nemá vstupné napätie vplyv. Odpor pri svetelnom toku 10 LUX je 8 kOhm, po vypnutí svetla sa po sekunde zvýši na 100 kOhm a po desiatich sekundách na 300 kOhm. V kapitole s návrhom riešenia je popísaná schéma zapojenia spoločne s vhodne zvolenými hodnotami odporov v napäťovom deliči.



Obr. 2-17 Fotorezistor PERKIN ELMER FW300 [33]

2.8.2 Digitálny teplotný senzor

Pre meranie teploty okolitého prostredia použijeme digitálny teplotný senzor DS18B20 [34], ktorý komunikuje s mikroprocesorom pomocou zbernice 1-Wire. Táto zbernica vyžaduje iba jedinú dátovú linku na napájanie a aj komunikáciu. Každé zariadenie pripojené k tejto zbernici má unikátny 64-bitový identifikátor. Pred samotným dátovým prenosom musí mikroprocesor osloviť zariadenie s konkrétnym identifikátorom, ktoré následne pošle svoje údaje. Zbernica 1-Wire podporuje aj určitú formu broadcast správ, ale len v prípade, ak je k zbernici pripojené iba jediné zariadenie. V prípade broadcast správy a súčasnej odpovedi viacerých senzorov nastane na zbernici komunikačná kolízia. Senzor je napájaný priamo z dátovej linky vo forme parazitného napájania. Jeho operačná teplota je od -55°C do +125°C s presnosťou ±0.5°C (v rozsahu -10°C do +85°C). Rozlíšenie senzora je možné nastaviť v rozsahu 9 až 12 bitov, pričom maximálna prístupová doba pri najvyššej presnosti je 750 ms. Pre zjednodušenie programovania existujú pre platformu Arduino rôzne knižnice, ktoré implementujú nízkoúrovňovú komunikáciu a poskytujú priame komunikačné funkcie.



Obr. 2-18 Digitálny teplotný senzor MAXIM DS18B20 [33]

2.8.3 Sériová RFID čítačka

Pri bezpečnostných dverách bude umiestnená RFID čítačka, ktorá autentifikuje používateľov. Technológia RFID je používaná v širokom spektre zariadení, najčastejšie ako identifikátor tovarov alebo personálnych kariet. RFID systém pozostáva z čítačky a viacerých unikátnych kľúčov, ktoré vzájomne komunikujú vo vysokofrekvenčnom pásme. Čítačka vysiela do okolia v pravidelných intervaloch pulzy. Ak sa v okolí čítačky objaví RFID kľúč, tak cez anténu si nabije napájací kondenzátor a odošle odpoveď, najčastejšie 40-bitový unikátny identifikátor.

Na trhu je dostupné množstvo RFID čítačiek, avšak v našom regióne je dostupný a cenovo najprijaznivejší model ID-12 od spoločnosti ID Innovations [35]. Oproti modelu ID-20 má skrátený komunikačný rozsah, avšak vstavaná anténa s dosahom 12 cm [36] plne postačuje pre potreby dverového snímača. V prípade nutnosti je možné dosah rozšíriť pridaním externej antény. Model ID-12 umožňuje len čítanie obsahu kľúčov a pre zápis sú určené modely s koncovkou RW, ktoré sú ale pre naše použitie zbytočné. Čítačka komunikuje s EM 4001 kompatibilnými kľúčmi na frekvencii 125 kHz, pričom s riadiacim mikroprocesorom komunikuje pomocou UART rozhrania. Modul umožňuje prepínať medzi ASCII a Wiegand26 kódovaním. Okrem komunikačných liniek obsahuje ID-12 aj výstup na LED a bzučiak, ktorými môže byť identifikované načítanie karty. Modul pracuje so vstupným napätím 5 V a priemerným prúdovým zaťažením 30 mA.

Obr. 2-19 RFID čítačka ID Innovations ID-12 [35]

2.9 Regulácia prostredia

Podobne ako pri senzoroch, tak aj akčné členy by mali reprezentovať schopnosti navrhovaného systému. V diplomovej práci sa nebudeme zaoberať akčnými členmi so štandardizovaným HVAC rozhraním, ale budeme demonštrovať možnosti vybraných mikroprocesorov. Rozšírenie systému o ovládanie HVAC zariadení je vhodná oblasť na budúce rozširovanie tejto práce. Pri návrhu regulátorov bude nutné dbať aj na optimalizáciu vstupno-výstupných charakteristík a zadefinovanie správania podľa hysteréznej krivky alebo konštanty.

2.9.1 LED osvetlenie

Reguláciu prostredia budú zabezpečovať tri typy akčných členov. V prvom prípade sa bude jednať o LED osvetlenie, ktorého intenzita bude ovládaná pomocou PWM rozhrania na procesore. LED osvetlenie je akčný člen, ktorý bude priradený k senzoru osvetlenia. Čím nižšie bude osvetlenie v miestnosti, tým bude LED osvetlenie intenzívnejšie.

2.9.2 Relé na otváranie dverí

V prípade RFID čítačky a bezpečnostných dverí by akčný člen vykonával otváranie dverí, avšak nakoľko je takýto člen úzko špecifický podľa typu zamykacieho mechanizmu a cieľom našej práce je len demonštrovanie funkcionality, tak tento člen nahradíme pomocou relé. Na výstupy z relé je následne možné kedykoľvek zapojiť akýkoľvek akčný člen a rozšíriť tak celkovú funkcionalitu. Návrh, doplnenie a testovanie takéhoto akčného člena je vhodná oblasť na budúce rozšírenie tejto práce.

2.9.3 Krokový motorček ovládania vykurovania

K teplotnému senzoru bude priradený krokový motorček nastavujúci intenzitu vykurovania, ktorý zabezpečí plynulú reguláciu teplotného ventilu. Existuje obrovské množstvo najrôznejších krokových motorčekov, ktoré sa rozdeľujú podľa veľkosti krokov a spôsobu ovládania na unipolárne a bipolárne. Zatiaľ čo pre ovládanie unipolárnych motorov postačuje jedna polarita napätia, tak bipolárne vyžadujú obidve polarity a teda aj zložitejšie zapojenie. Pre našu prácu sme si zvolili bežný unipolárny krokový motorček, ktorý sme mali k dispozícii. Vybraný krokový motorček poskytuje dostatočný výkon a presnosť pre dokonalé riadenie, avšak výstupy z mikroprocesora Atmel ATmega 328 nedisponujú požadovanými napäťovými a prúdovými charakteristikami. Z tohto dôvodu sme si pre zosilnenie signálu vybrali výkonový člen ULN2803A [37], ktorý splňuje kladené požiadavky a je dostupný za veľmi nízku cenu. Integrovaný obvod obsahuje osem kanálov, ktoré umožňujú ovládať motory až do napätia 50 V a prúdu 500 mA. Paralelným zapojením viacerých kanálov je

možné maximálny výstupný prúd ďalej zvyšovať. Návrh a schéma implementácie sa nachádza v kapitole s návrhom riešenia.

Obr. 2-20 Unipolárny krokový motorček

3 Špecifikácia funkcionality systému

Predbežná špecifikácia spoločne so základnými požiadavkami na systém už boli popísané v kapitole s analýzou. Kapitola špecifikácie funkcionality systému bližšie špecifikuje všetky vlastnosti systému.

Systém pozostáva z jednej centrálnej a viacerých externých modulov. Všetky moduly sú plne autonómne a teda schopné plniť špecifikovanú funkcionalitu aj bez napojenia na centrálnu jednotku. Každý modul má vlastný riadiaci firmvér a prednastavené konfiguračné hodnoty vo vstavanej EEPROM pamäti. Napájanie modulov je realizované pomocou POE napájania z počítačovej siete a záložnej batérie. Niektoré moduly, ktoré komunikujú cez bezdrôtovú sieť WiFi môžu mať zabezpečený alternatívny zdroj energie v podobe batérie a solárneho článku. Na moduly s obmedzeným napájaním sú kladené špecifické požiadavky na znižovanie energetického príkonu. Významné šetrenie je možné dosiahnuť pomocou úsporného režimu mikroprocesora a úsporného Wi-Fi shield-u. Externé moduly sa po spustení inicializujú a podľa nastavení si do vstavanej pamäti začnú ukladať namerané hodnoty. Centrálna jednotka periodicky kontroluje lokálnu sieť na prítomnosť nových modulov, zbiera informácie od aktívnych modulov a nastavuje ich správanie. Funkcionalita komponentov a ich inicializačné procesy sú bližšie opísané v príslušných kapitolách.

3.1 Centrálna jednotka

Centrálna jednotka je hlavný riadiaci prvok systému. Po pripojení k sieti s POE napájaním sa spustí inicializácia sieťového zásobníka. Centrálna jednotka umožňuje pripojenie používateľov aj z internetu a preto je nutné na lokálnom smerovači korektne nastaviť presmerovanie portu 80. Po úspešnej inicializácií sa jednotka pripojí k NTP serveru a aktualizuje si svoj čas. Až následne začne vyhľadávať externé moduly, ktoré sú pripojené k lokálnej sieti. Dátová komunikácia s externými modulmi prebieha pomocou sieťového protokolu UDP. Spoľahlivý a robustnejší protokol TCP je použitý pri komunikácii s prezentačným serverom a jednotlivými používateľmi. Podrobnejšia sieťová komunikácia s jednotlivými zasielanými správami je opísaná v kapitole so sieťovou komunikáciou.

3.2 Externé moduly

Externé moduly zbierajú informácie o prostredí a vplývajú na okolie pomocou akčných členov. Všetky moduly sú plne autonómne a teda schopné plnohodnotnej činnosti bez

centrálnej jednotky. Pre zmenu regulačných vlastností a zisťovania hodnôt prostredia je ale nutné prepojiť externé moduly s centrálnou jednotku. Vzájomné prepojenie všetkých systémových komponentov je realizované cez sieť Ethernet. Bezdrôtové moduly sa pripájajú pomocou siete Wi-Fi na prístupový bod, kde prebieha konverzia rámcov IEEE 802.11 na IEEE 802.3. Tento proces je plne autonómny a pre vyššie komunikačné vrstvy aj plne transparentný.

Po pripojení k sieti si každý modul vyžiada pridelenie IP adresy z DHCP servera. V prípade potreby môžu mať moduly pridelenú aj statickú IP adresu. DHCP protokol neposkytuje len IP adresu zariadenia a masku podsiete, ale aj ďalšie dôležité parametre, ako adresu sieťovej brány, ktorá sa využíva pri pripojení do internetu. Pomocou brány je realizovaná komunikácia so zariadeniami mimo lokálnej siete. Po úspešnom získaní IP adresy sa modul pripojí k NTP serveru, od ktorého si vyžiada aktuálny čas. Aktuálny čas sa využíva pri zaznamenávaní údajov zo senzorov a šifrovaní sieťovej komunikácie. Každý modul následne načíta nastavenia z EEPROM pamäti a začne vykonávať stanovené činnosti. Po dobu, kým modul nie je prepojený s centrálnou jednotkou, si podľa nastavení ukladá namerané údaje do EEPROM pamäti. Táto pamäť je ale obmedzená a preto si modul uchováva len niekoľko posledných hodnôt. Po vzájomnom prepojení si následne centrálna jednotka vyžiada preposlanie týchto údajov. Podrobnejšia sieťová komunikácia s jednotlivými zasielanými správami je opísaná v nasledujúcej kapitole.

3.3 Sieťová komunikácia

Vzájomná komunikácia medzi centrálnou jednotkou a externými modulmi prebieha z dôvodu závažných obmedzení TCP komunikácie na sieťových shield-och pomocou UDP správ. Využívaná je pritom komunikácia typu unicast aj broadcast. Celkovo sme pre komunikáciu zadefinovali 7 rôznych typov správ: *Hello, Identify, Request, Post data, Post config*, *Config* a *Ack*. *Hello* paket je jediná správa, ktorá je posielaná všetkým staniciam formou broadcast-u. Zvyšná komunikácia okrem NTP požiadaviek prebieha len medzi centrálnou jednotkou a externými modulmi formou unicast-u. V prípade časovej synchronizácie posielajú jednotlivé moduly požiadavku na získanie aktuálneho času na lokálny alebo globálny NTP server. Je to tak jediná dátová komunikácia externých modulov so serverom mimo lokálnej siete. Pre zvýšenie robustnosti celého riešenia je možné vytvoriť NTP server aj v rámci lokálnej siete. Z dôvodu zvýšenia prehľadnosti a zjednodušenia analýzy sieťovej komunikácie sme pre UDP komunikáciu zvolili voľné porty 2050 pre centrálnu jednotku a 2051 pre externé moduly.

Hello paket

Po úspešnej inicializácií a časovej synchronizácií s NTP serverom sleduje každý externý modul lokálnu sieť a očakáva *Hello* broadcast správu od centrálnej jednotky. Centrálna jednotka posiela *Hello* paket periodicky, čím si udržuje zoznam aktuálne dostupných externých modulov. Ak externý modul neprijme dlhšiu dobu *Hello* paket, tak predpokladá odpojenie centrálnej jednotky a prepne sa na lokálny režim, keď si namerané dáta ukladá do EEPROM pamäti. Externé moduly prijímajú pakety aj počas lokálneho režimu, avšak odpovedajú len na *Hello* pakety a iné požiadavky z bezpečnostného hľadiska ignorujú. V prípade spustenia centrálnej jednotky a prijatia *Hello* paketu je pôvodná činnosť ihneď obnovená. Externé moduly si zároveň z *Hello* paketu zaznamenajú IP adresu centrálnej jednotky, na ktorú budú v prednastavených intervaloch posielať namerané hodnoty.

Identify paket

Po prijatí *Hello* paketu sa musia všetky dostupné externé moduly identifikovať pomocou *Identify* paketu. Táto správa obsahuje unikátne identifikačné číslo každého modulu a jeho typ a textové pomenovanie. Centrálna jednotka si na základe tejto správy udržuje databázu externých modulov a ich IP adries. V prípade konfigurácie externého modulu pomocou *Config* paketu je dátová výmena smerovaná priamo na požadované zariadenie. Unikátne identifikačné číslo externých modulov slúži aj na zosúladenie nameraných dát a ich prezentáciu, keďže identifikátor na základe IP adresy je len dočasný a po uplynutí DHCP časového limitu môže byť každému modulu pridelená iná IP adresa. Ak centrálna jednotka zistí pomocou *Identify* paketu nový externý modul, alebo externý modul v lokálnom režime, tak si od neho vyžiada všetky zaznamenané dáta pomocou správy *Request*.

Request paket

Request paket posiela centrálna jednotka vybranému externému modulu, od ktorého požaduje zaslanie svojich nameraných dát a nastavení. Centrálna jednotka si neudržuje globálne nastavenia všetkých externých modulov pretože v prípade zmeny alebo pridania nového externého modulu by mohol nastať konflikt. Externé moduly odpovedajú na *Request* paket správou *Post data* alebo *Post config*.

Post data paket

Post paket môže obsahovať buď aktuálnu hodnotu zo senzora prostredia, alebo záznam z EEPROM pamäte. Takýto záznam pozostáva z jednotlivých položiek nastavení alebo nameraných hodnôt z času lokálneho režimu. Staré namerané hodnoty sa môžu odstrániť až po prijatí potvrdzovacej správy *Ack*, ktorú zasiela centrálna jednotka.

Post config paket

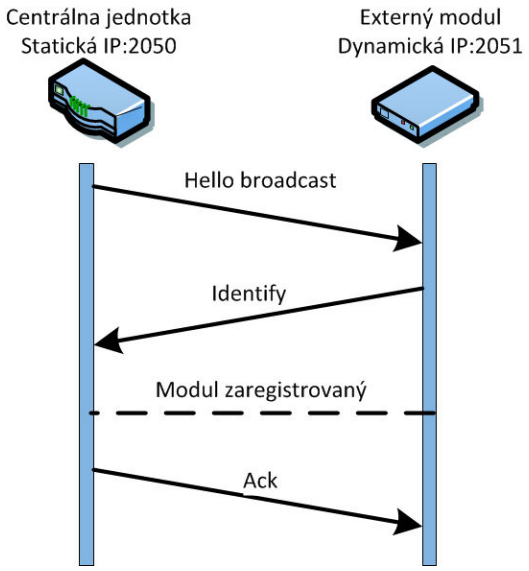
Správa *Post config* slúži na informovanie centrálnej jednotky o aktuálnych nastaveniach externých modulov. Centrálna jednotka vyžaduje túto správu v prípade zmeny konfigurácie vybraného modulu a z tohto dôvodu predchádza *Config paket-u.*

Config paket

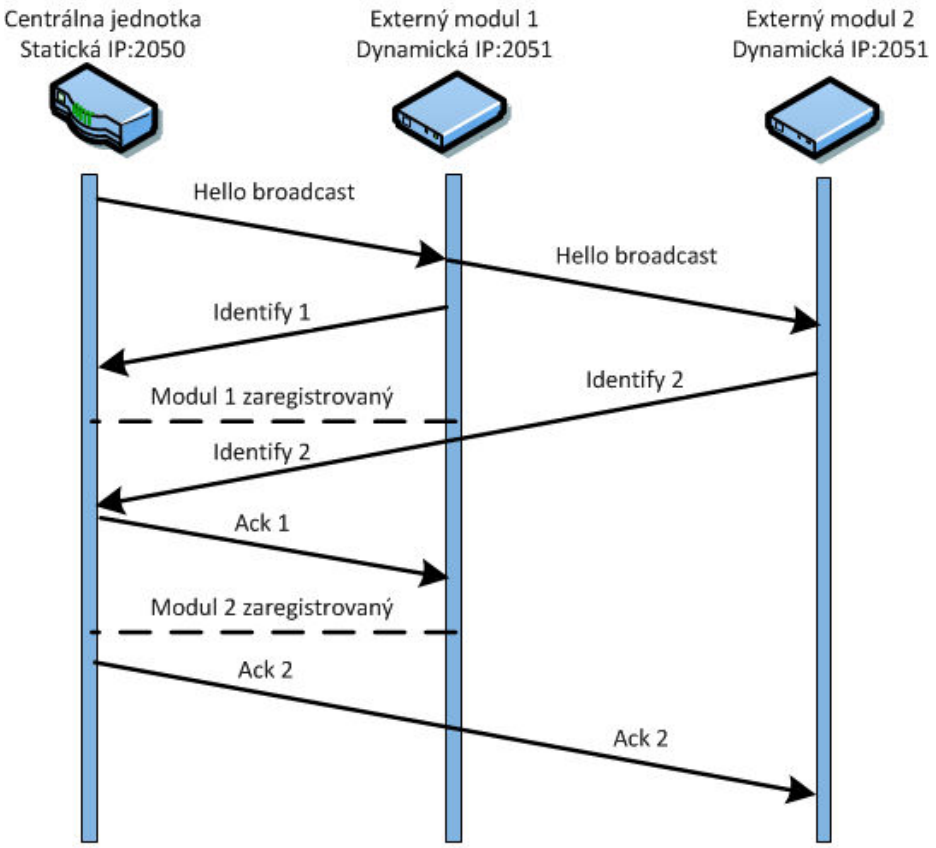
V prípade požiadavky na zmenu nastavení externého modulu posiela Centrálna jednotka *Config* správu. Zmena nastavení sa môže týkať nových komunikačných parametrov, identifikačných parametrov alebo spôsobu ovládania akčného člena. Po vykonaní konfiguračných zmien sa externý modul automaticky reštartuje a aplikuje zmeny. O úspešnosti vykonania zmien je centrálna jednotka informovaná pomocou správy *Ack*.

Ack paket

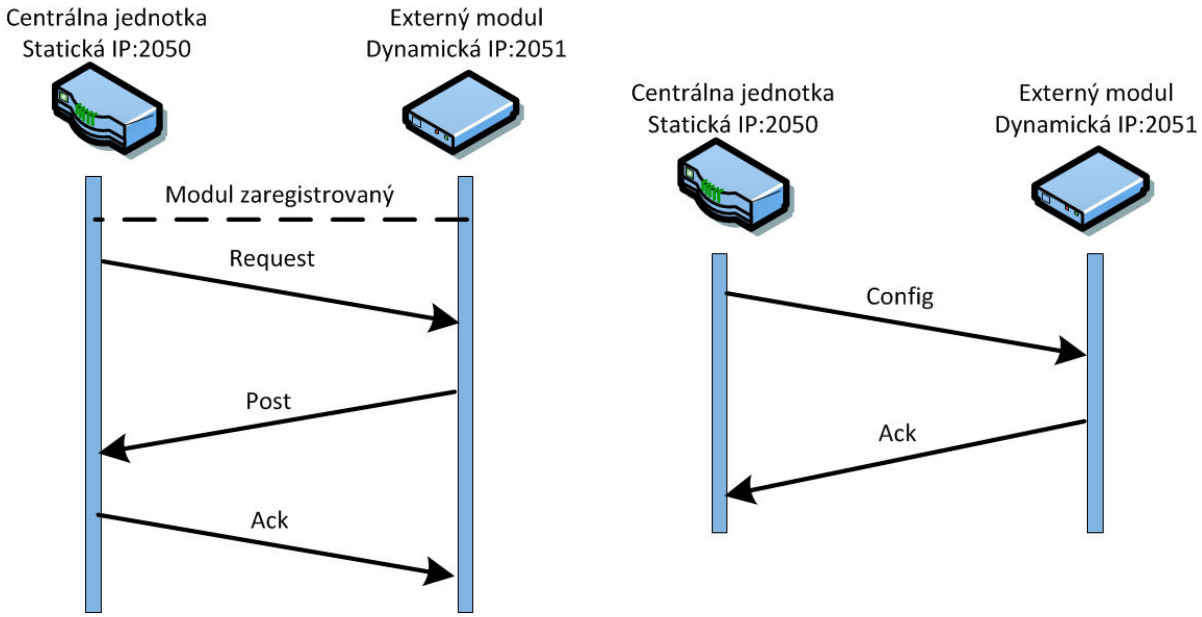
UDP komunikácia je nespoľahlivá a o prípadnej strate paketu nie je informovaná zdrojová ani cieľová stanica komunikácie, preto tento nedostatok musíme vyriešiť vlastným potvrdzovaním. *Ack* paket potvrdzuje poslednú správu komunikácie a zabezpečuje retransmisiu v prípade straty danej správy. Správa *Ack* môže potvrdzovať viacero typov správ a preto musí príjemcu informovať aj o type správy ktorú potvrdzuje. V sieti môže dochádzať aj k zdvojovaniu UDP správ a z tohto dôvodu musia všetky používané správy obsahovať sekvenčné číslo. Zdvojené správy príjemca zahodí a odpovie správu *Ack*, ktorá obsahuje typ a sekvenčné číslo prijatej správy. Správa *Ack* nasleduje za správami *Identify*, *Post data*, *Post config* a *Config*. Podrobnejšia špecifikácia a návrh vnútornej štruktúry paketov je bližšie rozpísaný v kapitole návrhu riešenia. Časový priebeh jednotlivých správ a ich vzájomná náväznosť je znázornená na nasledujúcich obrázkoch.



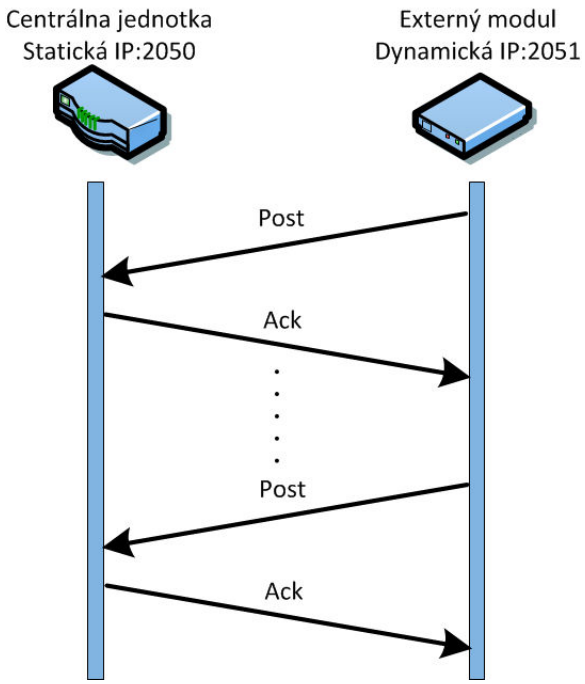
Obr. 3-1 Identifikácia a registrácia externého modulu



Obr. 3-2 Identifikácia a registrácia viacerých externých modulov



Obr. 3-3 Požiadavka na poslanie dát a konfigurácia externého modulu



Obr. 3-4 Pravidelné posielanie nameraných údajov

Komunikácia medzi centrálnou jednotkou a používateľmi je realizovaná pomocou protokolu HTTP, ktorý využíva TCP sieťový protokol. Používateľ si tak môže zobraziť informácie a vykonať konfiguráciu pomocou ľubovoľného zariadenia, ktoré disponuje internetovým prehliadačom. Podrobnejšie informácie o prezentačnom rozhraní sa nachádza v kapitole s používateľským prostredím.

3.4 Obmedzenia sieťovej komunikácie

Vnorené systémy majú obmedzený výpočtový výkon a zväčšia nie sú priamo stavané na sieťovú komunikáciu a preto prichádzajú na rad niektoré obmedzenia. Pre zabezpečenie vzájomnej komunikácie medzi jednotlivými sieťovými prvkami by bolo najvhodnejšie použiť spoľahlivý sieťový protokol TCP. Tento protokol je dostatočne robustný, zabezpečuje automatické potvrdzovanie doručenia správ a prípadne aj ich retransmisiu. Oproti tomu je sieťový protokol UDP nespoľahlivý, avšak výrazne rýchlejší, keďže je menší a nevyžaduje nadväzovanie spojení. Protokol TCP má na použitých sieťových shield-och niekoľko výrazných obmedzení a z tohto dôvodu sme boli pre implementáciu našej práce nútení použiť protokol UDP a navrhnúť vlastné mechanizmy potvrdzovania správ a odstraňovania duplicitných paketov. Azda najvýraznejším obmedzením Ethernet shield-u sú maximálne štyri súčasné TCP spojenia. Špecifikácia počíta s viacerými externými modulmi a preto by bolo nutné zakaždým nadväzovať a rušiť vytvorené spojenia, čo je časovo aj dátovo náročné. Navyše pri posielaní správ na odpojený modul môže prichádzať k dočasným zaseknutiam

systému, ktorý čaká na odpoveď od modulu a nemôže tak spracovávať ďalšie požiadavky a resetovať Watchdog časovač, ktorý spôsobí zresetovanie modulu. Ďalším problémom je posielanie broadcast správ, ktoré by pracovali s protokolom UDP, zatiaľ čo zvyšná komunikácia v TCP. Takáto implementácia by spôsobovala komplikácie najmä s WiFly shield-om, ktorý nedokáže súčasne pracovať s UDP a aj TCP protokolmi a pri ich prepínaní musí byť zakaždým resetovaný.

Vybrané procesory majú nízky výpočtový výkon a súčasne zabezpečujú šifrovanie celej komunikácie, tým sa tieto obmedzenia ešte výraznejšie prehlbujú. Arduino Ethernet aj WiFly shield-y komunikujú pomocou sériového rozhrania, ktoré poskytuje niekoľkokrát nižšiu dátovú priepustnosť ako dané sieťové rozhranie. Najväčším problémom je tak prijímanie rámcov vo vyššej frekvencii ako sú zariadenia schopné spracovávať. Takéto rámce sa ukladajú do vyrovnávacej pamäti a spracované sú až keď sa uvoľní strojový čas. Problém nastane v prípade, keď vyrovnávacia pamäť pretečie a nové prijaté rámce sa už nemajú kam ukladať. Takáto situácia nie je bežná a nastáva len v prípade rozsiahlej broadcast-ovej komunikácie, ktorú musí spracovávať každé zariadenie v danej sieti. Môže sa tak stať že v rozsiahlej a preťaženej sieti nastane výpadok niektorých zariadení. Krátkodobé výpadky sú eliminované pomocou potvrdzovacích *Ack* paketov, avšak dlhodobý výpadok má za následok prepnutie externých modulov do lokálneho režimu. Problém zahltenia siete sa dá vyriešiť vyčlenením celého systému do samostatnej virtuálnej VLAN siete. Týmto spôsobom sa nielen zvýši bezpečnosť, ale navyše sa zníži vyťaženosť siete a všetkých sieťových zariadení.

Samostatným problémom sú útoky typu DOS, ktoré úmyselne preťažujú systém neopodstatnenými požiadavkami. Najkritickejšie miesto v prípade takéhoto útoku sú externé moduly s WiFly shield-om, ktoré sa naviažu na odosielateľa takejto správy. Pravidelné posielanie neplatných správ tak presmeruje komunikáciu na útočníka a preruší komunikáciu s centrálnou jednotkou. Keďže sú všetky správy šifrované, tak útočník sa nedostane k ich obsahu, avšak efektívne obmedzí činnosť týchto modulov. Ochrana voči tomuto typu útoku je netriviálna, avšak pri útoku z vonkajšej siete by mala byť vyriešená na vstupnom smerovači. Navrhnutý systém je slabo odolný voči tomuto typu útoku a preto je pred jeho nasadením do praxe vyžadovaná úprava sieťovej bezpečnostnej politiky.

3.5 Zabezpečenie

Prvá úroveň zabezpečenia spočíva vo vyčlenení zariadení do vlastnej VLAN siete. K modulom tak majú prístup len špecifikované zariadenia. Druhá úroveň zabezpečenia spočíva

v šifrovaní komunikačných správ, ktoré sú vymieňané medzi jednotlivými zariadeniami. Pre zvýšenie zabezpečenia je každú sekundu generovaný nový šifrovací kľúč. Generovanie nových kľúčov zabraňuje duplikovaniu požiadaviek, ktoré môžu ovplyvniť správanie systému. Útočník tak nedokáže zreplikovať správu a poslať príkaz napríklad na otvorenie dverí. Dočasný šifrovací kľúč je generovaný z aktuálneho času a hlavného hesla a preto je mimoriadne dôležité, aby boli všetky zariadenia časovo zosynchronizované. Správu je možné dešifrovať len v prípade poznania šifrovacieho algoritmu, generátora dočasného kľúča a hlavného hesla.

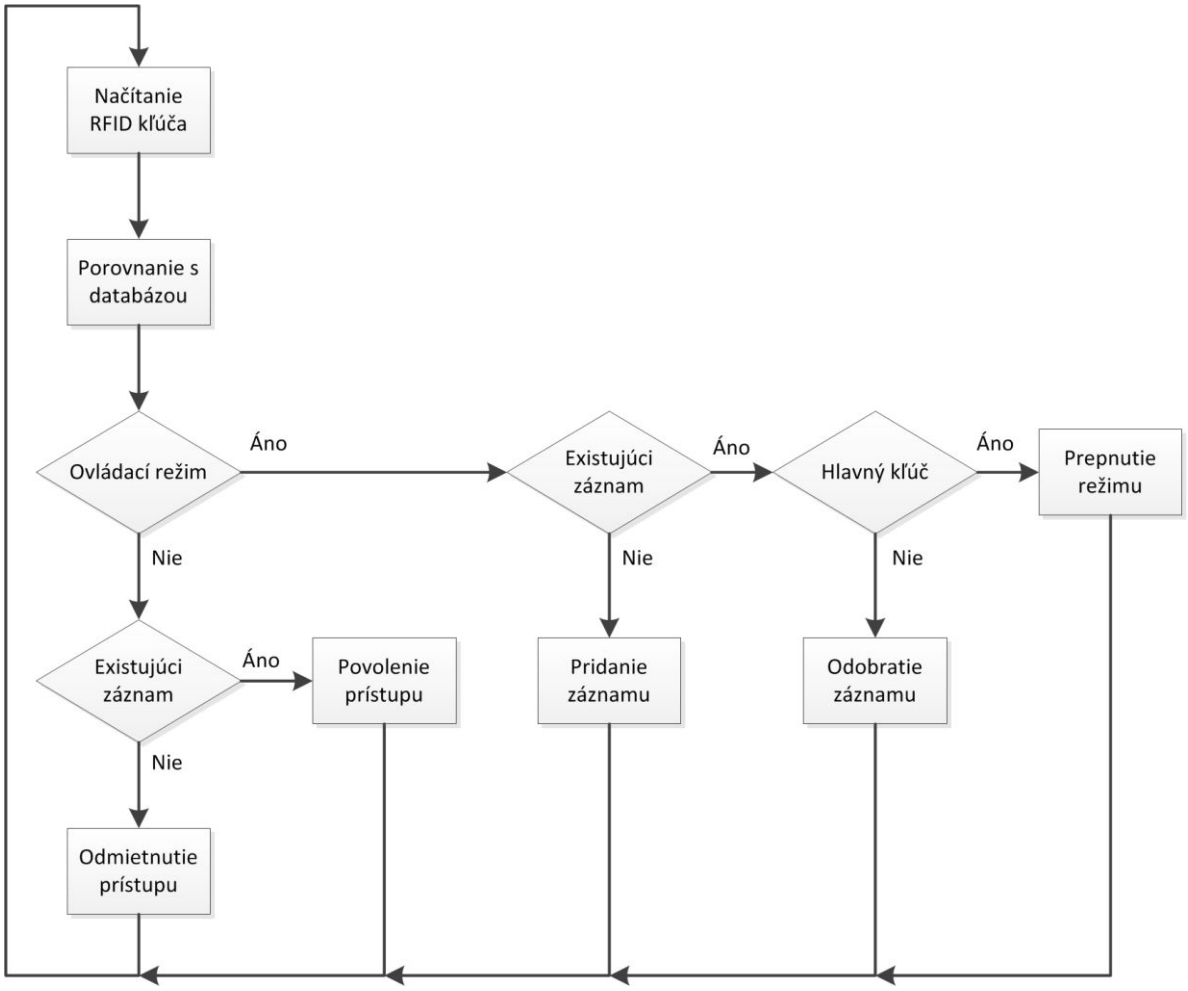
Tretia úroveň zabezpečenia sa sústreďuje na zabezpečenie centrálnej jednotky voči neželaným prienikom z internetu. Voči takýmto prienikom sa dá brániť jednak obmedzením povolených IP adries na hlavnom smerovači, ale aj autentifikáciou používateľov pri prístupe. V analýze sme zdôvodnili, že na rozsiahlu a bezpečnú autentifikáciu neposkytuje vybraný mikroprocesor dostatočný výpočtový výkon, avšak aspoň základná autentifikácia pomocou Base64 kódovania môže zabrániť väčšej časti útokov.

Štvrtá úroveň zabezpečenia je aktivovaná v prípade vykonávania kritických príkazov, akými sú napríklad otvorenie bezpečnostných dverí. Každý modul umožňuje definíciu kritických operácií, pri ktorom prebieha autentifikácia pomocou SMS v GSM sieti. Toto zabezpečenie zabraňuje vykonávaniu nepovolených kritickým operácií, ktoré môžu viesť k ujme. Poslednou úrovňou nepriameho zabezpečenia je Watchdog časovač, ktorý reštartuje modul v prípade neočakávaných operácií a straty reakcie.

3.6 Monitorovanie a riadenie prostredia

Ako už bolo bližšie charakterizované aj v kapitole s analýzou, tak navrhovaný systém bude obsahovať trojicu senzorov a trojicu akčných členov, ktoré budú demonštrovať celkové možnosti navrhovaného systému. K aktuálnym hodnotám z jednotlivých senzorov ale aj k ich histórií sa bude dať pristupovať pomocou používateľského rozhrania na centrálnej jednotke. Používateľské rozhranie umožní sledovať nielen aktuálny stav jednotlivých senzorov, ale umožní aj ovplyvňovať charakteristiky správania sa akčných členov. Jednotlivé externé moduly budú disponovať senzormi osvetlenia, teploty alebo RFID čítačkou pri bezpečnostných dverách. Zapojenie fotorezistora, snímajúceho osvetlenie, musí byť navrhnuté tak, aby optimálne reprezentovalo hodnoty osvetlenia pri bežnom použití. Výhodou digitálneho teplomera je jeho nezávislosť na okolitom prostredí, ktorá nevyžaduje žiaden špecifický návrh. Cieľom diplomovej práce je návrh funkčného prototypu, preto sa v tejto

práci nebudeme zaoberať optimálnym návrhom umiestnenia modulov a ani ich krycím puzdrom. RFID čítačka umožní prístup len uloženým RFID kľúčom a ostatné odmietne. Pomocou hlavného kľúča bude možné pridávať alebo odoberať jednotlivé prístupové RFID kľúče. Vývojový diagram čítania a spravovania RFID kľúčov je znázornený na nasledujúcej schéme.



Obr. 3-5 Vývojový diagram správania sa RFID čítačky

Na riadenie prostredia postačuje jednoduchá logika, ktorá bude ovplyvňovať správanie sa akčných členov na základe vstupných hodnôt zo senzorov. Aby sa predišlo rôznym problémom s rozkmitaním riadenia, tak používateľ bude mať možnosť nastaviť oneskorovací čas akčného člena. Vhodný čas oneskorenia dokáže zabrániť väčšine problémov, ktoré vznikajú rozkmitaním a pre cieľ tejto práce je takýto prístup plne postačujúci. Rozšírenie o ďalšiu logiku inteligentného správania sa je vhodnou oblasťou pre budúce rozšírenie.

Cieľom práce nie je navrhnúť špecifické riešenie na mieru vybranej domácnosti, ale predstaviť funkčný prototyp. Z tohto dôvodu budú použité len zjednodušené akčné členy,

ktoré dostatočne demonštrujú funkcionalitu systému. Návrh plnohodnotných akčných členov je vhodná oblasť na budúce rozširovanie tejto práce. Z akčných členov bude použité PWM ovládané LED osvetlenie, ktoré bude reprezentované klasickou svetelnou LED diódou. Riadiaci signál externého modulu je možné zosilniť a použiť aj na reálne osvetlenie. Ventil vykurovania bude riadený krokovým motorčekom. Bezpečnostné dvere s RFID čítačkou budú otvárané pomocou riadiaceho relé, ktoré môže byť v budúcnosti pripojené na zabezpečovací mechanizmus dverí.

Samostatným prostriedkom riadenia prostredia bude aj implementácia štandardu zobúdzania zariadení pomocou WOL sieťových rámcov. Po zadaní MAC adresy v používateľskom prostredí sa vygeneruje WOL rámec, ktorý prebudí požadované zariadenie. Pri výbere vhodných senzorov a akčných členov nebolo cieľom implementovať komunikáciu s inteligentnými akčnými členmi HVAC v domácnosti, ale demonštrovať vzájomnú komunikáciu modulov a predstaviť jednotlivé funkcie konceptuálneho modelu systému.

3.7 Používateľské rozhranie

Ako už bolo špecifikované v predchádzajúcich kapitolách, tak najvhodnejší spôsob reprezentácie používateľského rozhrania umožňujúceho zobrazovať aktuálne údaje a pozmeňovať správanie akčných členov je web rozhranie. Takéto rozhranie je kompatibilné s obrovským množstvom najrôznejších zariadení od počítačov, cez mobily až po elektronické čítačky kníh. Web rozhranie musí byť čo najjednoduchšie a veľmi intuitívne. Cieľom diplomovej práce nie je vytvorenie graficky úchvatného a prepracovaného rozhrania, ale predvedenie spôsobu zobrazovania základných informácií. Budúce rozšírenie tejto práce sa tak môže zamerať na vylepšenie používateľského rozhrania, ktoré môže byť realizované úplne odlišným spôsobom. Pre zachovanie spätnej kompatibility by bolo pôvodné používateľské rozhranie ponechané, avšak pribudlo by nové používateľské rozhranie vo forme aplikácií na najpoužívanejšie mobilné systémy ako Android alebo iOS. Takéto rozhranie by nebolo obmedzované HTML kódom a pomalou centrálnou jednotkou, ktorá by tak len preposielala dáta a fungovala ako zabezpečený proxy server.

Napriek tomu, že používateľské rozhranie bude zobrazovať len jednoduché výstupy a nastavenia jednotlivých modulov, tak by bolo vhodné doplniť aj zobrazenie časových priebehov nameraných hodnôt v prehľadných grafoch. Centrálna jednotka ale nemá dostatočný výpočtový výkon na generovanie takýchto grafov a preto je nutné na túto funkcionalitu vyhradiť nezávislý server, ktorý by zo vstupných údajov generoval požadované

grafické výstupy. Centrálna jednotka môže pristupovať na lokálny grafický server, alebo môže použiť službu tretích strán. Pre potreby tejto práce je výhodnejšie použiť niektorú z bezplatných služieb s touto funkcionalitou. Nielen pre platformu Arduino je dostupná služba COSM [38], ktorá je známejšia pod starším pomenovaním Pachube. Služba COSM je dostupná bezplatne, avšak má obmedzenie na maximálne 100 aktualizácií za minútu, čo pre naše použitie plnohodnotne postačuje. Pomocou jednoduchých HTML GET dotazov je možné generovať dynamické grafy so zvolenými parametrami, ktoré je možné jednoducho vložiť do web stránky používateľského rozhrania. K službe je priamo dostupná API knižnica [39], ktorá umožňuje jednoduché posielanie údajov aj z platformy Arduino. Údaje sú ukladané na COSM serveroch a preto nie je nutné pred generovaním grafov opätovne posielať všetky zaznamenané dáta. Používateľské rozhranie musí umožniť aj zakázanie tejto služby v prípade ak používateľ nesúhlasí s podmienkami použitia alebo nedôveruje službe.

3.8 Benefity navrhovaného konceptu

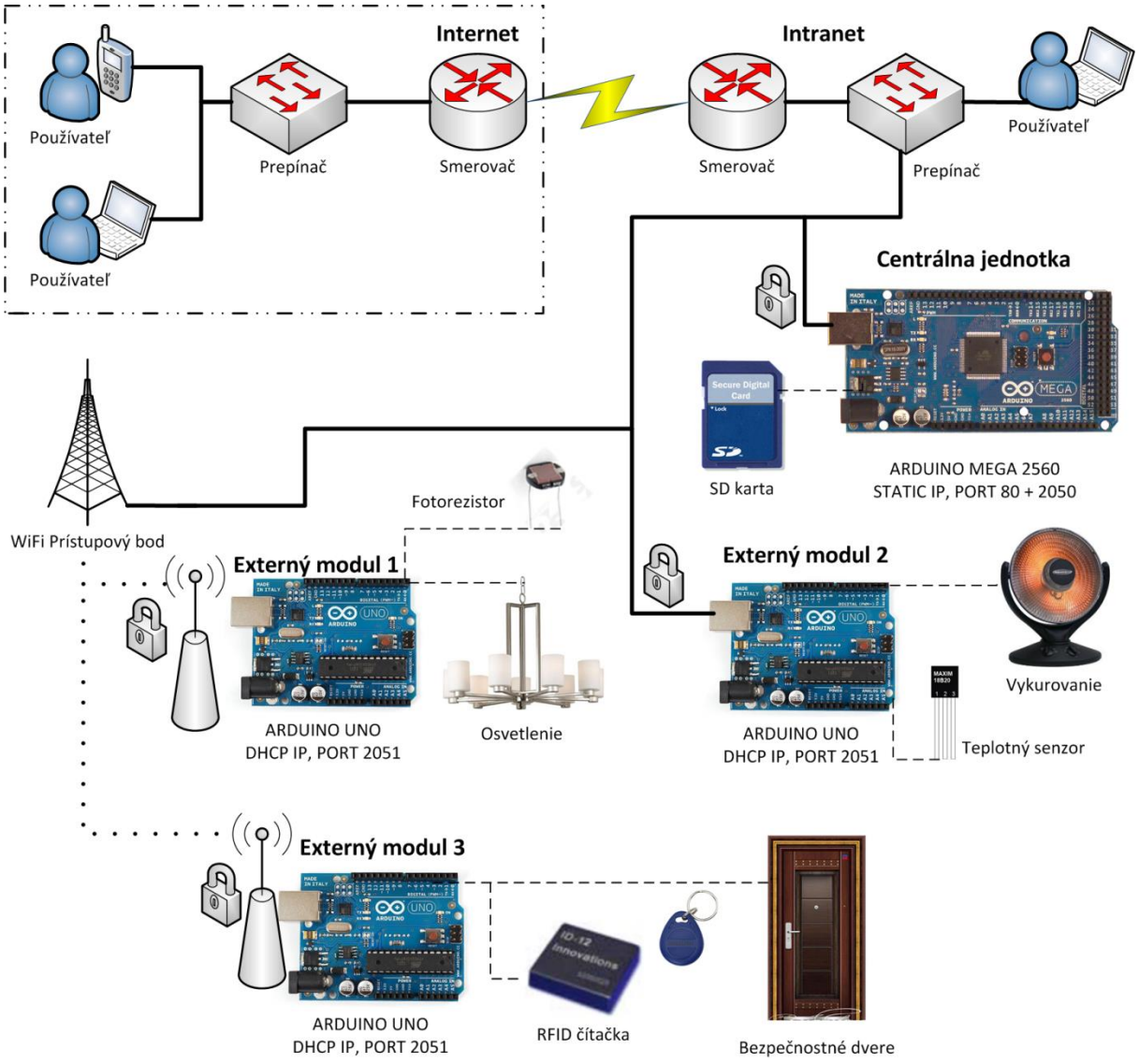
Navrhovaný systém poskytuje oproti komerčným riešeniam a iným dostupným projektom hneď niekoľko výhod. Azda najväčšou výhodou je priaznivá cena prototypu a plná autonómnosť externých modulov, ktoré sú schopné samostatnej činnosti aj bez centrálnej jednotky. Odhadovaná cena celého systému so všetkými modulmi, senzormi a akčnými členmi je len 400 EUR. Nesporným benefitom navrhovaného systému je aj jeho vysoká škálovateľnosť a modulárnosť, ktorá umožňuje jednoduché rozširovanie funkcionality v budúcnosti. Škálovateľnosť dovoľuje jednoduché pridávanie množstva externých modulov s rôznymi senzormi a akčnými členmi. Plnohodnotné šifrovanie zabezpečuje celý systém voči viacerým typom útokov, ktoré majú za cieľ získať informácie alebo vykonať nepovolenú operáciu. Jednotlivé moduly používajú štandardizované sieťové a bezdrôtové rozhranie, takže môžu fungovať aj na existujúcej sieťovej infraštruktúre.

4 Návrh riešenia

Kapitola návrhu riešenia bližšie opisuje použité hardvérové komponenty, ich zapojenie, vzájomnú interakciu, softvérové ovládače a vývojové nástroje spoločne s podrobným návrhom predstavovaného riešenia.

4.1 Konceptuálny model

Na nasledujúcom obrázku sa nachádza základný konceptuálny model systému. Obrázok popisuje 4 oblasti systému: externé moduly so senzormi a akčnými členmi, centrálnu jednotku s pamäťovou kartou, privátnu LAN sieť a pripojenie používateľov z lokálnej siete a internetu na systém. Návrh riešenia celého systému, ako aj všetkých jeho častí je opísaný v nasledujúcich kapitolách.



Obr. 4-1 Bloková schéma navrhovaného systému

4.2 Schémy modulov

Keďže centrálna jednotka, ktorá pozostáva z platformy Arduino Mega 2560 a Ethernet shield-u nevyžaduje žiadnu ďalšiu hardvérovú modifikáciu, tak je na tomto mieste zbytočné uvádzať kompletné schémy týchto komponentov. Rovnako ako centrálna jednotka, tak aj externé moduly používajú Ethernet alebo Wifly shield-y v kombinácií s platformou Arduino Uno. Schémy všetkých použitých komerčných hardvérových komponentov sa nachádzajú v prílohe tejto práce. Nasledujúce podkapitoly popisujú návrhy schém zapojenia senzorov a akčných členov v rámci jednotlivých modulov, ktoré sú rozdelené podľa funkcionality.

4.2.1 Modul so svetelným senzorom

Modul so svetelným senzorom je z pohľadu návrhu zapojenia najjednoduchší. Modul obsahuje fotorezistor FW300 a akčný člen v podobe svetelnej LED diódy. Fotorezistor a aj LED dióda musia mať pre optimálnu funkcionalitu pridružený vhodne zvolený rezistor. Zatiaľ čo rezistor R2 slúži ako predradný odpor k LED dióde, tak rezistor R1 funguje ako napäťový delič vďaka ktorému môžeme merať odpor fotorezistora. Pri reálnom použití sa odpor fotorezistora mení v rozmedzí 100 až 20 kOHM, čo pri 1 kOHM rezistore R1 určuje rozsah vstupného napätia ADC na 0,2-4,6 V. Pre dodatočnú filtráciu napätia je k fotorezistoru pridružený filtračný kondenzátor. Intenzita LED osvetlenia je ovládaná PWM reguláciou na porte *D5* a intenzita osvetlenia je načítavaná cez port *A5*. Schéma zapojenia a schéma dosky plošného spoja modulu so svetelným senzorom sa nachádza v prílohách F a G.

4.2.2 Modul s teplotným senzorom

Modul s teplotným senzorom obsahuje dvojicu akčných členov: signalizačnú viacfarebnú RGB LED diódu a krokový motorček ventilu vykurovania. RGB LED dióda so spoločnou katódou je zapojená ako trojica obyčajných LED diód, avšak zlúčenie trojice svetelných diód do jedného puzdra umožňuje plynulú reguláciu farebného výstupu, ktorý bude signalizovať teplotu prostredia. Rovnako ako v prípade LED diódy z prvého modulu, tak aj viacfarebná RGB LED dióda bude ovládaná PWM reguláciou, čo umožní meniť nielen farebnosť ale aj intenzitu osvetlenia. RGB LED dióda tak musí byť pripojená k portom *D3*, *D5* a *D6*, ktoré disponujú PWM funkcionalitou. Procesor ATmega328 v platforme Arduino dovoľuje maximálne 40 mA prúdové zaťaženie výstupov, čo nepostačuje na napájanie krokového motorčeka. Okrem toho je ich výstupné napätie len 5 V, čo je výrazne menej ako pracovné napätie štandardného motorčeka, ktoré je najbežnejšie 12 V. Z tohto dôvodu je do návrhu nutné zakomponovať vstup na externý zdroj napájania a zosilňovač (ULN2803A), ktorý umožní riadenie unipolárneho krokového motorčeka. Pre zjednodušenie návrhu plošného

spoja sme na ovládanie krokového motorčeka použili porty *D2*, *A4*, *A2* a *D8*. Popis softvérovej časti riadenia motorčeka sa nachádza v kapitole s opisom riešenia. Okrem akčných členov sa v schéme nachádza aj digitálny teplotný senzor DS18B20, ktorý komunikuje s riadiacim procesorom po zbernici 1-Wire. K zbernici na zvolenom porte *D9* je možné pripojiť ľubovoľný počet teplotných senzorov, ktoré budú podľa návrhu pracovať v režime s parazitným napájaním s 4,7 kOHM rezistorom medzi 5 V napájaním a zbernicou. Schéma zapojenia a schéma dosky plošného spoja modulu s teplotným senzorom sa nachádza v prílohách H a I.

4.2.3 Modul s RFID čítačkou

Tretí modul obsahuje RFID čítačku, LED diódu indikácie čítania, spínacie relé a zvukový bzučiak indikácie autentifikácie. RFID čítačka ID-12 je k hlavnému procesoru pripojená pomocou sériového rozhrania na port *D6*. Čítačka je napájaná priamo z platformy Arduino, pričom vstup *RES* je aktívny v stave s nízkou úrovňou. V našom návrhu nevyužívame reset RFID čítačky a z toho dôvodu môže byť vstup *RES* v trvalom stave s vysokou úrovňou. Vstup *+/-* určuje formát výstupu, ktorý je pre ACSII kódovanie nutné nastaviť na stav s nízkou úrovňou. Pin č.10 na ID-12 využijeme na pripojenie svetelnej LED diódy na indikáciu prebiehajúceho čítania karty. Prečítané čísla kariet sú cez sériovú zbernicu poslané riadiacemu procesoru, ktorý ich porovná s databázou platných kariet. V prípade detekcie hlavnej karty je sprístupnená možnosť pridávania alebo odoberania existujúcich záznamov. Pri detekcii platnej karty je na zadefinovaný čas nastavený výstup *D8* na stav s vysokou úrovňou, čo spôsobí prepnutie relé a zopnutie napájania pre mechanizmus otvárania dverí. Tento mechanizmus nie je súčasťou diplomovej práce, avšak návrh počíta s jeho neskorším doplnením. Súčasťou tretieho modulu je aj bzučiak, ktorý je ovládaný procesorom cez port *D9*. Procesor signalizuje pomocou bzučiaka načítanie platnej alebo neplatnej karty. Schéma zapojenia a schéma dosky plošného spoja modulu s RFID senzorom sa nachádza v prílohách J a K.

4.3 Komunikačný protokol

Návrh komunikačného protokolu vychádza zo špecifikácie sieťovej komunikácie. Centrálna jednotka a externé moduly si počas činnosti vymieňajú dáta, ktoré sú logicky rozdelené do viacerých typov správ. V špecifikácii sme si zadefinovali správy *Hello, Identify, Request, Post data, Post config, Config* a *Ack*. Jednotlivé správy majú špecifikovanú funkcionalitu a pevnú štruktúru. V nasledujúcej tabuľke je znázornená navrhnutá štruktúra komunikačných paketov.

Identifikačná hlavička		Dáta			Ukončenie
Typ správy	Sekvenčné číslo	Veľkosť	Dáta s výplňou	Kontrolný súčet	Časová pečiatka
1 B	4 B	1 B	(8x – 2) B	1 B	4 B
Nešifrované		Šifrované			Nešifrované

Tab. 4-1 Štruktúra komunikačných paketov

Pre dodržanie princípu jednotnosti používajú všetky pakety rovnaký formát, ktorý pozostáva z identifikačnej hlavičky, dátovej časti a ukončenia paketu. Zatiaľ čo identifikačná hlavička a ukončenie paketu sú v nezašifrovanej forme, tak dátová časť je už šifrovaná. Hlavička a ukončenie paketov nie sú šifrované z dôvodu rýchlejšieho spracovania a zahodenia duplicitných a neplatných UDP paketov, ktoré môžu vznikať v sieti.

4.3.1 Hlavička paketov

Identifikačná hlavička obsahuje identifikátor typu správy a sekvenčné číslo paketu. Identifikátor jednoznačne definuje typ správy, ktorý stanovuje spôsob spracovania dátovej časti. Identifikátor typu správy má veľkosť 1 B, čo vytvára priestor pre 256 rôznych typov správ, avšak v súčasnej práci rozlišujeme len 7 typov správ, ktoré sú zadefinované v nasledujúcej tabuľke.

Typ správy	*Hello*	*Identify*	*Request*	*Post_Data*	*Post_Cfg*	*Config*	*Ack*
Identifikačné č.	0x00	0x01	0x02	0x03	0x04	0x05	0x06

Tab. 4-2 Identifikačné čísla správ

Sekvenčné číslo v hlavičke paketu jednoznačne identifikuje komunikačnú reláciu medzi centrálnou jednotkou a externým modulom, čím umožňuje kontrolu doručenia správ. Správa *Ack* ukončuje aktuálnu reláciu. V hlavičke *Ack* paketu sa nachádza sekvenčné číslo aktuálnej relácie a v dátovej časti je uložené sekvenčné číslo nasledujúcej relácie, ktoré je oproti predchodcovi zväčšené o veľkosť poslaných dát. V prípade ak nie je do stanoveného časového limity prijatý *Ack* paket, tak sa správa považuje za nedoručenú a opätovne sa prepošle s rovnakým sekvenčným číslom. V prípade ak modul prijme dva rovnaké typy správ s rovnakým sekvenčným číslom, tak druhú správu automaticky a bez zbytočného spracúvania zahodí. *Hello* paket má celú identifikačnú hlavičku zaplnenú nulami, keďže pri pravidelnom broadcaste nemá význam kontrolovať sekvenčné čísla a typ správy má taktiež kód 0x00. Odpoveď na túto správu v podobe *Identify* paketu tak slúži nielen na identifikáciu externého modulu, ale aj na synchronizáciu sekvenčných čísel.

4.3.2 Dátová časť paketov

Dátová časť paketov je rozdelená na tri časti. Keďže celá dátová časť je šifrovaná blokovou šifrou XTEA so 64 bitovou veľkosťou šifrovacieho bloku, tak jej veľkosť musí byť násobkom tohto čísla. Efektívne dáta môžu mať pritom rôznu veľkosť a preto sú na správnu veľkosť doplnené nulami. Prvý bajt dátovej časti indikuje efektívnu dĺžku dát, ktorá môže byť 1 až 254 B. Táto veľkosť je pre našu prácu plne postačujúca z dôvodu, že prijímanie väčších paketov je náročné na obmedzenú RAM pamäť v použitých procesoroch. Posledný bajt dátovej časti je vyhradený pre kontrolnú sumu, ktorá je počítaná funkciou XOR na jednotlivých bajtoch efektívnych dát. Obsahy efektívnych dátových častí pre jednotlivé typy správ sa nachádzajú v nasledujúcich tabuľkách.

Efektívne dáta: Hello paket (0x00)		
byte ID (1 B)	**byte Typ (1 B)**	**string Meno (12 B)**
ID (1 B)	0x00 - Centrálna jednotka	Meno (12 B)

Tab. 4-3 Efektívna dátová časť Hello paketu

Efektívne dáta: Identify paket (0x01)		
byte ID (1 B)	**byte Typ (1 B)**	**string Meno (12 B)**
ID (1 B)	0x01 - Svetelný modul	Meno (12 B)
ID (1 B)	0x02 - Teplotný modul	Meno (12 B)
ID (1 B)	0x03 - RFID modul	Meno (12 B)

Tab. 4-4 Efektívna dátová časť Identify paketu

Efektívne dáta: Request paket (0x02)
byte Požiadavka (1 B)
0x01 - Pošli aktuálnu hodnotu zo senzora
0x02 - Pošli uloženú hodnotu z EEPROM
0x03 - Pošli nastavenia modulu

Tab. 4-5 Efektívna dátová časť Request paketu

Efektívne dáta: Post data paket (0x03)			
byte Typ (1 B)	**bool Záznam (1 B)**	**u_long Čas (4 B)**	**string Hodnota (~ B)**
0x01 - Svetelnosť	Z EEPROM (1 B)	Čas (4 B)	Svetelnosť (2 B)
0x02 - Teplota	Z EEPROM (1 B)	Čas (4 B)	Teplota (2 B)
0x03 - RFID	Z EEPROM (1 B)	Čas (4 B)	RFID Tag + Pozícia (6 B)

Tab. 4-6 Efektívna dátová časť Post data paketu

Efektívne dáta: Post config paket (0x04)		
byte Periodicita (1 B)	int Regulácia (2 B)	byte Oneskorenie (1 B)
Zápis a periodicita (1 B)	Cieľová hodnota (2 B)	Oneskorenie AČ (1 B)

Tab. 4-7 Efektívna dátová časť Post config paketu

Efektívne dáta: Config paket (0x05)				
byte ID (1 B)	string Meno (12 B)	byte Periodicita (1 B)	int Regulácia (2 B)	byte Oneskorenie (1 B)
ID (1 B)	Meno (12 B)	Zápis a periodicita (1 B)	Cieľová hodnota (2 B)	Oneskorenie AČ (1 B)

Tab. 4-8 Efektívna dátová časť Config paketu

Efektívne dáta: Ack paket (0x06)	
byte Type (1 B)	u_long Nasledujúce sekv. č. (4 B)
Typ potvrdenia (1 B)	Sekvenčné číslo (4 B)

Tab. 4-9 Efektívna dátová časť Ack paketu

4.3.3 Ukončenie paketov

Ukončovacia časť paketu s veľkosťou 4 bajty obsahuje časovú pečiatku, ktorá je nevyhnutná pre generovanie zodpovedajúceho dešifrovacieho kľúča. Generovanie šifrovacích kľúčov nemôže byť viazané na presný čas, pretože môže nastať situácia, keď sa zariadenia čiastočne časovo rozsynchronizujú alebo sa poslaný paket zdrží v sieti a v takom prípade bude aktuálny dešifrovací kľúč neplatný. Ak by sa v správe neposielala časová pečiatka šifrovania, tak prijímajúca strana by musela otestovať viacero dešifrovacích kľúčov kým by správu úspešne dešifrovala alebo vyhodnotila ako neplatnú. Takýto prístup by nadmerne zaťažoval pomalý procesor, čo by malo negatívny vplyv na celý systém. Prijímajúca strana dokáže okamžite po prijatí paketu rozhodnúť, či nie je časová pečiatka príliš stará a či sa nejedná o útok spôsobom replikácie dátových paketov. Po overení platnosti časovej pečiatky sa z nej a hlavného šifrovacieho kľúča vygeneruje dočasný dešifrovací kľúč, ktorý sa použije na dešifrovanie prijatej správy. Z dešifrovaných dát sa vypočíta kontrolná suma, ktorá sa porovná s kontrolnou sumou v dátovej časti paketu a až pri jej zhode sa začnú spracovávať prijaté dáta. Chyba kontrolnej sumy signalizuje chybu v sieťovom prenose, hlavnom šifrovacom kľúči alebo sa jedná o neplatný, vygenerovaný paket.

4.4 Rozdelenie EEPROM pamäte

Vo vstavanej EEPROM pamäti sa nachádzajú nielen nastavenia jednotlivých modulov, ale aj záznamy zo senzorov v čase lokálneho režimu. Pamäť je rozdelená na tri hlavné oblasti: identifikačnú, konfiguračnú a záznamovú. Identifikačná oblasť obsahuje identifikačné číslo,

meno externého modulu a heslo. Typ externého modulu nie je nutné ukladať, keďže pri zmene typu je nutné nahrať nový riadiaci firmvér, ktorý v sebe obsahuje potrebný identifikátor. Oblasť s konfiguračnými nastaveniami obsahuje šifrovacie heslo, nastavenie ukladania a periodicity záznamov v lokálnom režime a nastavenia cieľovej hodnoty akčného člena. Najvyšší bit z bajtu ukladania záznamov reprezentuje zapnutie záznamu a zvyšné bity sú vyhradené na nastavenie periodicity snímania senzorov a posielania paketov. Z dôvodu obmedzeného počtu zápisov do EEPROM pamäti môže používateľ zakázať ukladanie záznamov. Posledná záznamová časť pamäte obsahuje dáta záznamov z obdobia lokálneho režimu. Jeden dátový záznam sa skladá z časovej pečiatky a zaznamenanej hodnoty. Modul s RFID senzorom musí v pamäti uchovávať aj zaregistrované RFID kľúče. Rozdelenie EEPROM pamäte pre jednotlivé externé moduly je zhrnuté v nasledujúcich tabuľkách.

Údaje v EEPROM	ID	Meno	Heslo	Záznam a periodicita	Cieľová hodnota AČ	Oneskorenie AČ	Záznamy meraní
Veľkosť	1 B	12 B	16 B	1 B	2 B	1 B	6 B
Pozícia	0	1 - 12	13 - 28	29	30 - 31	32	33 – 1023

Tab. 4-10 Rozdelenie EEPROM pamäte v externých moduloch bez RFID senzora

Údaje v EEPROM	Identifikácia	Nastavenia	Hlavný kľúč	Počet kľúčov	Počet záznamov	Registrované kľúče	Záznamy meraní
Veľkosť	29 B	4 B	5 B	1 B	1B	5x B	9 B
Pozícia	0 - 28	29 - 32	33 - 37	38	39	40 - y	z - 1023

Tab. 4-11 Rozdelenie EEPROM pamäte v externých moduloch s RFID senzorom

Veľkosť záznamu s nastaveniami akčného člena je variabilná podľa typu externého modulu. Zatiaľ čo RFID čítačka musí mať zaznamenané všetky kľúče prístupových kariet, tak na nastavenie tepelného regulátora postačuje len niekoľko parametrov. Od veľkosti nastavení akčného člena sa odvíja aj maximálny počet záznamov meraní. Záznamy sa do pamäte zapisujú až do jej úplného zaplnenia a potom sa začnú jednotlivé záznamy prepisovať.

Štruktúra rozdelenia EEPROM pamäte v centrálnej jednotke je oproti externým modulom značne zložitejšia. V EEPROM pamäti sa už neuchovávajú záznamy meraní z jednotlivých externých modulov, pretože tie sú uchovávané na SD karte, avšak oproti externým modulom pribudli rozšírené sieťové nastavenia, nastavenia komunikačných služieb a GSM služieb. Aj keď by centrálna jednotka mala mať statickú IP adresu, tak ponechávame aj možnosť jej dynamického pridelenia z DHCP servera. Rozdelenie EEPROM pamäte pre centrálnu jednotku je zhrnuté v nasledujúcej tabuľke.

Údaje v EEPROM	Kontrolný flag	Použi DHCP	DHCP obnova	MAC adresa	IP adresa	Gateway adresa	Maska siete
Veľkosť	1 B	1 B	1 B	6 B	4 B	4 B	4 B
Pozícia	0	1	2	3 - 8	9 - 12	13 - 16	17 - 20
Údaje v EEPROM	DNS server	Web Port	Použi COSM	Meno	Heslo	GSM číslo	Použi GSM
Veľkosť	4 B	4 B	1 B	12 B	16 B	15 B	1 B
Pozícia	21 - 24	25 - 28	29	30 - 41	42 - 57	58 - 72	73

Tab. 4-12 Rozdelenie EEPROM pamäte v centrálnej jednotke

4.5 Softvérové knižnice

Pri implementovaní navrhnutého systému sme využili niekoľko softvérových knižníc, ktoré poskytujú funkcie vykonávajúce nízkoúrovňové prístupy. Navrhnutú prácu by bolo možné realizovať aj bez použitia týchto knižníc, avšak celková časová náročnosť práce by zbytočne výrazne narástla. Naprogramovanie vlastných nízkoúrovňových funkcií by malo výhodu v šetrení pamäťového priestoru, keďže by bola implementovaná len požadovaná funkcionalita a nie celá robustná knižnica. Softvérové knižnice sú umiestnené v adresári *\Arduino\libraries*. Kompletné zdrojové súbory aj s príkladmi použitia softvérových knižníc sa nachádzajú na CD nosiči, ktorý je súčasťou tejto práce. Nasledujúce podkapitoly popisujú jednotlivé použité softvérové knižnice.

4.5.1 Ethernet

Sieťová Ethernet knižnica je priamou súčasťou implementačného prostredia, kde poskytuje základnú funkcionalitu pre sieťovú komunikáciu pomocou obvodu WIZnet W5100. Z knižnice využívame funkcie získavania IP adries pomocou DHCP servera, posielanie a prijímanie UDP paketov, TCP klient pre posielanie dát na službu COSM a TCP server, ktorý poskytuje web rozhranie pre používateľov. V zdrojovom kóde je nutné použiť *#include <SPI.h>*, *#include <Ethernet.h>* a *#include <EthernetUdp.h>*, ktoré sprístupnia požadované komunikačné funkcie.

Spustenie sieťového rozhrania sa vykonáva pomocou funkcie *Ethernet.begin(mac),* ktorá si od DHCP servera vyžiada pridelenie IP adresy. V prípade použitia statickej IP adresy je potrebné zavolať funkciu s parametrami *(mac, ip)*, kde *ip* je typu *IPAddress*. Ethernet shield nemá z výroby hardvérovo špecifikovanú MAC adresu a preto je ju nutné pred každou sieťovou komunikáciou softvérovo zadefinovať pomocou bajtového poľa ako napríklad *byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED}*. Samozrejmosťou je, že všetky zariadenia v lokálnej sieti musia mať rôzne MAC adresy. Funkcia získania IP adresy z DHCP servera

nerieši problém vypršania času prenájmu adresy, ktorý je bližšie opísaný v kapitole s problémami s implementáciou.

Pre posielanie a prijímanie UDP paketov je najskôr nutné vytvoriť inštanciu *EthernetUDP Udp* a následne spustiť prijímanie na lokálnom porte pomocou funkcie *Udp.begin(localPort)*, kde hodnota *localPort* je typu *unsigned int*. Prijatie dát je indikované nenulovou návratovou hodnotou funkcie *Udp.parsePacket(),* ktorá vyjadruje počet prijatých bajtov. Načítavanie prijatého UDP paketu do lokálneho zásobníka je vykonávané funkciou *Udp.read(packetBuffer, packetSize).* Posielanie UDP paketov je realizované pomocou troch funkcii. Funkcia *Udp.beginPacket(IPaddress, port)* nastavuje komunikačné parametre, funkcia *Udp.write(packetBuffer, packetSize)* číta dáta z lokálneho zásobníka a funkcia *Udp.endPacket()* realizuje dátový prenos.

Vytváranie spojenia cez protokol TCP je o niečo zložitejšie, avšak v skratke popíšeme najhlavnejšie funkcie. Pre posielanie dát je nutné vytvoriť inštanciu *EthernetClient client*, inicializovať spojenie funkciou *client.connect(server, port)* a poslať dáta pomocou *client.write(packetBuffer, packetSize).* Spojenie je možné ukončiť funkciou *client.stop().* Pre vytvorenie TCP servera je nutné vytvoriť inštanciu *EthernetServer server(port),* čakať na klientov *EthernetClient client = server.available()* a prijímať ich požiadavky pomocou funkcie *client.read()* v prípade ak funkcie *client.connected()* a *client.available()* vracajú platnú hodnotu. Rovnako ako v prípade posielania dát je spojenie možné ukončiť funkciou *client.stop().*

4.5.2 WiFly [40]

Knižnica WiFly umožňuje konfiguráciu a komunikáciu s WiFly modulom. Pred prvým použitím je potrebné v súbore *Configuration.h* zmeniť položku *#define SHIELD_REVISION* na revíziu WiFly shield-u. My používame najstaršiu revíziu v1.4 (*SHIELD_REVISION 1)*, ktorá má 12 MHz kryštál a nemá hardvérový reset. Pomocou programu *SpiUartTerminal.ino* môžeme priamo komunikovať s WiFly modulom. Konfiguračný režim sa zapína pomocou zadania *$$$* do konzoly, ktorá ho musí poslať vo formáte bez ukončenia riadku *(No line ending)*. Ďalšia konfigurácia sa už vykonáva s ukončovaním riadku *(Carrige return).* Konfiguračné príkazy sú popísané v kapitole s konfiguráciou WiFly modulu.

Pred začiatkom komunikácie je nutné inicializovať modul pomocou príkazu *SpiSerial.begin()*, ktorý využíva knižnicu SoftwareSerial. Komunikácia s WiFly shield-om je oproti Ethernet shield-u značne jednoduchšia, avšak neumožňuje pokročilé nastavovania

komunikačných parametrov. Tieto parametre je nutné nastaviť v predstihu pomocou terminálu *SpiUartTerminal*. Dátová výmena je možná len na aplikačnej vrstve a prebieha pomocou funkcií *SpiSerial.read()* a *SpiSerial.write(data, size).* Veľkosť dostupných dát je možné zistiť pomocou funkcie *SpiSerial.available()* a prípadne ich zmazať pomocou *SpiSerial.flush().*

4.5.3 EEPROM

Knižnica EEPROM je priamou súčasťou implementačného prostredia a sprístupňuje funkcie operácií nad touto pamäťou. Pred použitím funkcii je nutné pridať knižnicu pomocou *#include <EEPROM.h>*. Operácie zápisu a čítania pracujú s blokom dát o veľkosti 1 B. Zápis sa vykonáva príkazom *EEPROM.write(address, content)* a čítanie obsahu pamäte pomocou *EEPROM.read(address).* V rámci zdrojového kódu je vhodné zadefinovať ďalšie pomocné funkcie ako napríklad *initializeEEPROM(),* ktorá sformátuje a zinicializuje EEPROM pamäť do vhodnej štruktúry.

4.5.4 SoftwareSerial

Použité vývojové moduly Arduino UNO obsahujú jediné sériové rozhranie, ktoré sa používa na programovanie a vstupno-výstupné operácie. Väčšina shield-ov v navrhovanom systéme komunikuje pomocou SPI rozhrania, avšak RFID čítačka a WiFly shield používa sériové rozhranie. Z tohto dôvodu potrebujeme dvojicu ďalších softvérových sériových rozhraní. Platforma Arduino umožňuje vytvoriť pomocou knižnice SoftwareSerial sériové rozhranie na ľubovoľnom digitálnom pine. Knižnica SoftwareSerial je priamou súčasťou implementačného prostredia. Pre sprístupnenie funkcií knižnice musíme do zdrojového kódu pridať *#include <SoftwareSerial.h>*. Pre inicializovanie softvérového rozhrania je nutné vytvoriť inštanciu so zadefinovanými portami pomocou funkcie *SoftwareSerial mySerial(inPort, outPort)* a spustiť s požadovanou komunikačnou rýchlosťou pomocou *mySerial.begin(9600).*

4.5.5 MemoryFree [41]

Knižnica MemoryFree umožňuje zisťovať zaplnenie pamäte RAM počas vykonávania programu. Keďže použité procesory majú dosť nízku kapacitu pamäte, tak pri implementácií návrhu musíme postupovať s čo najväčšou efektivitou. Problémom pretečenia pamäte je zlá detegovateľnosť, keďže program môže fungovať ďalej, avšak niektoré dáta sú poškodené. Na odhalenie takýchto situácií a zisťovanie kritických oblastí zdrojového kódu využijeme práve knižnicu MemoryFree. Knižnica je sprístupnená po pridaní *#include <MemoryFree.h>*. Aktuálna veľkosť voľnej pamäte RAM vo zvolenej oblasti zdrojového kódu sa zisťuje doplnením funkcie výpisu *Serial.println(freeMemory()).* Hodnoty sú po spustení programu vypisované v konzole.

4.5.6 Watchdog timer

Knižnica Watchdog timer (WDT) je priamou súčasťou implementačného prostredia a sprístupňuje funkcionalitu tohto časovača, ktorý slúži na bezpečné reštartovanie zariadenia v prípade neočakávanej operácie alebo na zobudenie z režimu spánku. V zdrojovom kóde je nutné sprístupniť knižnicu pomocou *#include <avr/wdt.h>*. Pre zjednodušenie nastavovania je vhodné zadefinovať hodnotu časovača napríklad na 2 sekundy pomocou príkazu *#define WDT_TIME WDTO_2S*. Watchdog časovač sa inicializuje funkciou *wdt_enable(WDT_TIME).* Časovač musí byť pravidelne resetovaný funkciou *wdt_reset(),* inák sa procesor automaticky zresetuje. V prípade volania funkcii s extrémne dlhou dobou vykonávania, ktorá je dlhšia ako maximálna doba Watchdog časovača, je nevyhnutné ho vypnúť pomocou funkcie *wdt_disable()* a opätovne ho zapnúť v nasledujúcom kroku zdrojového súboru.

4.5.7 XTEA [28]

Knižnica XTEA zabezpečuje šifrovanie komunikačných dát. Funkcie knižnice sú sprístupnené po pridaní *#include <Xtea.h>*. Keďže pre každý dátový paket používame šifrovanie s unikátnym dočasným kľúčom, tak pre každú novú reláciu šifrovania alebo dešifrovania musíme vytvoriť novú dočasnú inštanciu *Xtea crypto(tempKey)*. Šifrovanie prebieha pomocou funkcie *crypto.encrypt(XTEApacketBuffer + iter)*, ktorá v jednom kole zašifruje 8 bajtov dát. Rovnako je volaná aj funkcia dešifrovania *crypto.decrypt(XTEApacketBuffer + iter)*. Počet vykonaní funkcie šifrovania alebo dešifrovania je daný podielom veľkosti spracovávaných dát a veľkosti šifrovacieho bloku.

4.5.8 OneWire [42]

Knižnica OneWire obsahuje nízkoúrovňové funkcie poskytujúce pohodlné programátorské rozhranie pre prácu so zariadeniami komunikujúcimi cez zbernicu 1-Wire. Pre sprístupnenie funkcií je potrebné vytvoriť inštanciu *OneWire ds(pin)*. Vyhľadávanie dostupných senzorov sa vykonáva funkciou *ds.search(addr),* ktorý do premennej *addr* uloží adresu senzora. Po dokončení vyhľadávania je potrebné zresetovať zbernicu pomocou funkcií *ds.reset_search()* a *ds.reset()*. Pre spustenie teplotnej konverzie je potrebné vybrať senzor a nastaviť register podľa zvolených parametrov: *ds.select(addr); ds.write(0x44,1)*. Pre dokončenie konverzie je nutné počkať zadefinovaný čas, ktorý je daný jej presnosťou a opätovne zresetovať zbernicu. Pre načítanie nameraných hodnôt zvolíme adresu a opätovne nastavíme register: *ds.reset(); ds.select(addr); ds.write(0xBE).* Následne vykonáme 9 čítaní pomocou funkcie *data[i] = ds.read(),* kde je teplota reprezentovaná prvými dvoma bajtami.

4.5.9 SD karta

SD knižnica sprístupňuje funkcie prístupu k SD karte. Knižnica je priamou súčasťou implementačného prostredia a obsahuje množstvo funkcií pre prácu s SD kartou, z ktorých použijeme len niekoľko najvýznamnejších. Pre sprístupnenie knižnice je nutné použiť *#include <SD.h>* a vytvoriť globálne premenné *Sd2Card card; SdVolume volume* a *SdFile root.* Inicializácia SD karty prebieha pomocou funkcie *card.init(SPI_HALF_SPEED, chipSelect),* kde si môžeme vybrať medzi polovičnou alebo plnou rýchlosťou komunikácie (*SPI_FULL_SPEED)*. Informácie o type karty a použitom súborovom systéme sa dajú získať pomocou funkcií *card.type(), volume.init(card)* a *volume.fatType().* Z počtu a veľkosti sektorov je možné vypočítať celkovú kapacitu karty. Prehľadanie súborov na karte sa vykonáva príkazmi *root.openRoot(volume)* a *root.ls(LS_R | LS_DATE | LS_SIZE).* Súbor je možné otvoriť pomocou funkcie *file.open(&root, filename, access_flags)* a čítať alebo zapisovať dáta pomocou funkcií *file.write() a file.read(buffer, size)*.

4.5.10 WebServer [43]

Knižnicu WebServer Webduino [43] používame na zjednodušenie implementácie používateľského rozhrania, ktoré je založené na báze web stránok. Knižnica poskytuje kompletnú funkcionalitu, akú sme si zadefinovali v špecifikácií. Pre sprístupnenie knižnice je nutné použiť *#include "WebServer.h"* a vytvoriť objekt typu *WebServer* s požadovanými parametrami. Pomocou funkcie *webserver->setDefaultCommand(&defaultCmd)* je možné zadefinovať správanie sa v prípade prístupu na koreňovú stránku. Ďalšie stránky sa pridávajú príkazom *webserver.addCommand("stranka.html", &command)*. Po pridaní všetkých požadovaných stránok je ešte potrebné spustiť samotný server pomocou funkcie *webserver->begin()* a v cykle *loop()* pravidelne kontrolovať aktuálne požiadavky používateľov cez funkciu *webserver.processConnection(buffer, &length).* Autentifikácia používateľského oprávnenia na prístup k danej stránke sa kontroluje pomocou príkazu *server.checkCredentials(verification_string)*, ktorého vstup obsahuje verifikačný string zaznamenaný v BASE64 kódovaní. Server umožňuje aj dynamické vyhodnocovanie zadaných adries, čo je nevyhnutné pre dynamické sprístupňovanie stránok zobrazujúcich informácie o pripojených externých moduloch. K špecifickým adresám je možné pristúpiť pomocou funkcie *webserver->setUrlPathCommand(&accessCmd)* a *accessCmd(WebServer &server, WebServer::ConnectionType type, char **url_path, char *url_tail, bool tail_complete).* Vo funkcii *accessCmd* je následne možné naprogramovať logiku analýzy URL adries. Túto funkciu sme ďalej rozšírili nielen o možnosť zobrazovať dynamické stránky, ale pridali aj

logiku sťahovania a mazania uložených súborov na SD karte. K súborom sa pristupuje pomocou HTML príkazov GET a DELETE. V knižnicu WebServer sme taktiež zväčšili veľkosť vyrovnávacej pamäti *buffer* na 300 B, čo výrazne zvýšilo rýchlosť komunikácie a zmenšilo fragmentáciu veľkých TCP paketov. Okrem toho sme predefinovali objekt *m_client* na *public*, čo nám umožňuje flexibilnejšie obsluhovať viacero TCP relácií.

4.5.11 GSMSHIELD [31]

Na pokročilú autentifikáciu používateľa využíva systém GSM modul, s ktorým je možné cez sériovú zbernicu komunikovať priamo pomocou AT príkazov alebo použiť vhodnú knižnicu, ktorá takúto komunikáciu zjednodušuje. K vybranému GSM modulu existuje viacero dostupných knižníc, avšak na základe pamäťových nárokov a ďalších vlastností sme si zvolili knižnicu GSMSHIELD [31]. Pre správne fungovanie GSM modulu v IComsat shield-e bolo nutné vykonať niekoľko modifikácií tejto knižnice. Najdôležitejšou zmenou bolo predefinovanie komunikačných portov v súbore *GSM.h* a následne úprava funkcie *gsm.begin(9600)* tak, aby korektne spúšťala aj vypnutý GSM modul. Pomocou tejto funkcie a definície *#include "SIM900.h"* sa GSM inicializuje. Úspešné spustenie a funkčnú komunikáciu je možné overiť pomocou funkcie *gsm.getStatus(),* ktorá by mala vracať hodnotu 0x02. Podľa typu komunikácie vo forme hlasu alebo SMS správ je ďalej nutné zadefinovať *#include "sms.h"* alebo *#include "call.h"*. V našej práci sme využívali len SMS komunikáciu, kde sme zadefinovali nový objekt *SMSGSM sms* a komunikáciu zabalili do ďalších vytvorených funkcií *sendSms(*data)* a *receiveSms().*

4.6 Prebraté funkcie

Počas implementácie riešenia sme okrem opísaných softvérových knižníc použili niekoľko dostupných zdrojových kódov, ktoré vykonávajú požadovanú funkcionalitu a neboli sme tak nútení programovať už existujúce funkcie.

4.6.1 Spracovanie NTP paketov

Pri generovaní a spracovávaní NTP paketov sme sa inšpirovali zdrojovým kódom z knižnice *Ethernet*, ktorý popisuje získavanie aktuálneho času z NTP servera. NTP paket je generovaný periodicky pomocou funkcie *sendNTPpacket(timeServer)*, ktorá podľa špecifikácii vytvorí a pošle NTP požiadavku na nastavený server s portom 123. Prijatá odpoveď je zanalyzovaná a z údajov je vypočítaný presný čas, ktorý používame na šifrovanie a vytváranie časových pečiatok.

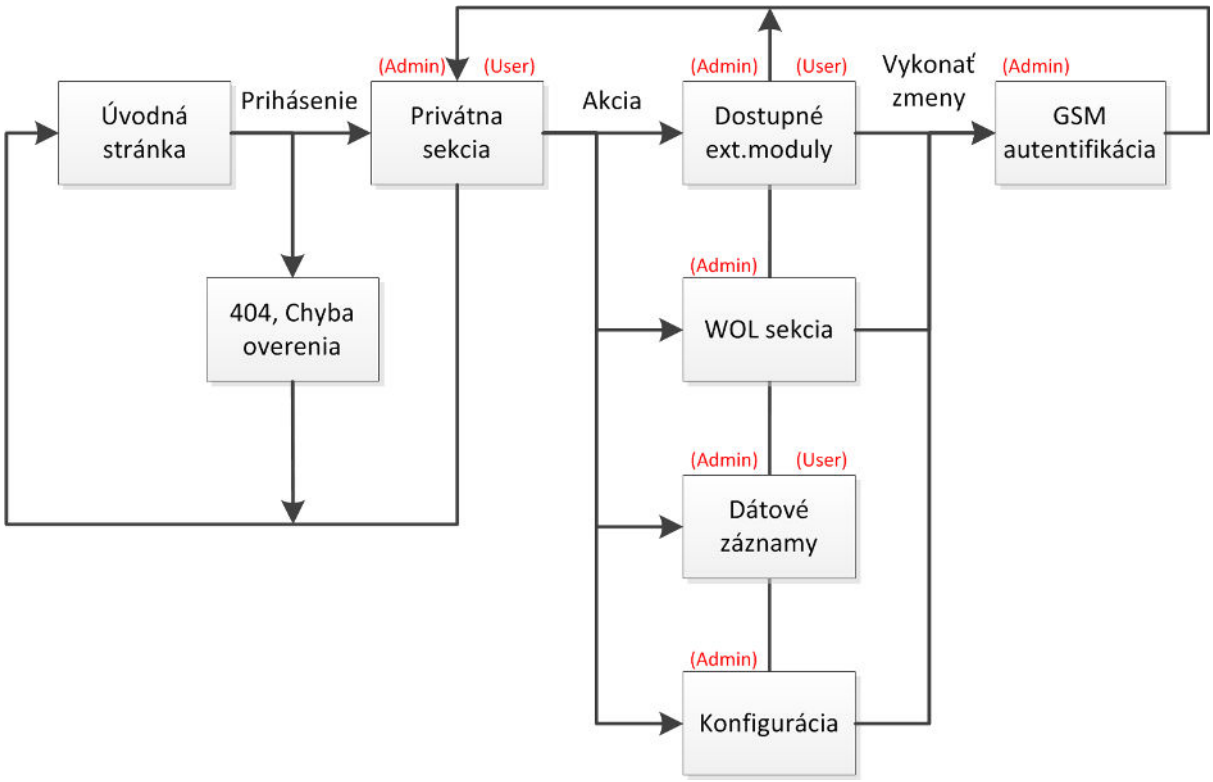
4.6.2 Čítanie RFID kľúčov

Modul s RFID senzorom komunikuje s ID12 modulom cez sériovú zbernicu. Posielané dáta je nutné zanalyzovať a zverifikovať pomocou kontrolného súčtu. Neplatné alebo nekompletné dáta sa ďalej nespracúvajú. Pri tvorbe funkcie *TagRead()* sme sa inšpirovali zdrojovým kódom zverejneným priamo na stránke Arduino [44]. Funkcia vracia pozitívny výsledok len pri korektne načítanom kľúči, ktorý zapíše do globálnej premennej *tagID[5]*.

4.7 Používateľské rozhranie

Používateľské rozhranie slúži na konfiguráciu, spravovanie a sledovanie správania sa systému. Je to jediné rozhranie medzi navrhovaným systémom a používateľom a z tohto dôvodu musí byť dostatočne robustné, informatívne ale aj jednoduché. Hlavným cieľom diplomovej práce je návrh a implementácia celého systému, pričom používateľské rozhranie má skôr informatívny charakter. V špecifikácii sme si stanovili podobu takéhoto rozhrania vo forme web stránok. Používateľské rozhranie je tak z pohľadu hardvérových požiadaviek veľmi nenáročné a teda aj schopné behu s obmedzenými hardvérovými prostriedkami.

V špecifikácií sme si stanovili požiadavky na základnú autentifikáciu používateľa a preto musí navrhované web rozhranie obsahovať úvodnú stránku, ktorá používateľa informuje o danom systéme a pre ďalšie pokračovanie požaduje zadanie autentifikačných údajov. K systému sa môžu prihlásiť viacerí používatelia s rôznymi právami a systém ich musí vzájomne rozlišovať a poskytovať im rôzne služby na základe ich úrovne privilegovanosti. Administrátor s najvyššou úrovňou privilegovanosti má právo na zmenu všetkých systémových vlastnosti, ktoré musia byť cez takéto rozhranie jednoducho konfigurovateľné. Používateľské rozhranie musí okrem konfiguračných možností sprístupňovať aj zobrazovanie a zmenu operačných vlastností jednotlivých externých modulov. Zobrazenie informácií o dostupných externých modulov tak musí byť vyriešené dynamicky a plne automaticky. Súčasťou informačnej karty o danom externom module je aj grafický priebeh hodnôt, ktorý znázorňuje správanie sa prostredia za niekoľko posledných hodín. Doplnkovú časť používateľského rozhrania tvorí ovládanie správania sa akčných členov a posielanie WOL paketov, ale aj možnosť stiahnutia všetkých dostupných dátových a chybových záznamov. Rozdelenie a prepojenie jednotlivých častí používateľského rozhrania spoločne s privilegovanosťou používateľov je znázornené na nasledujúcej schéme.



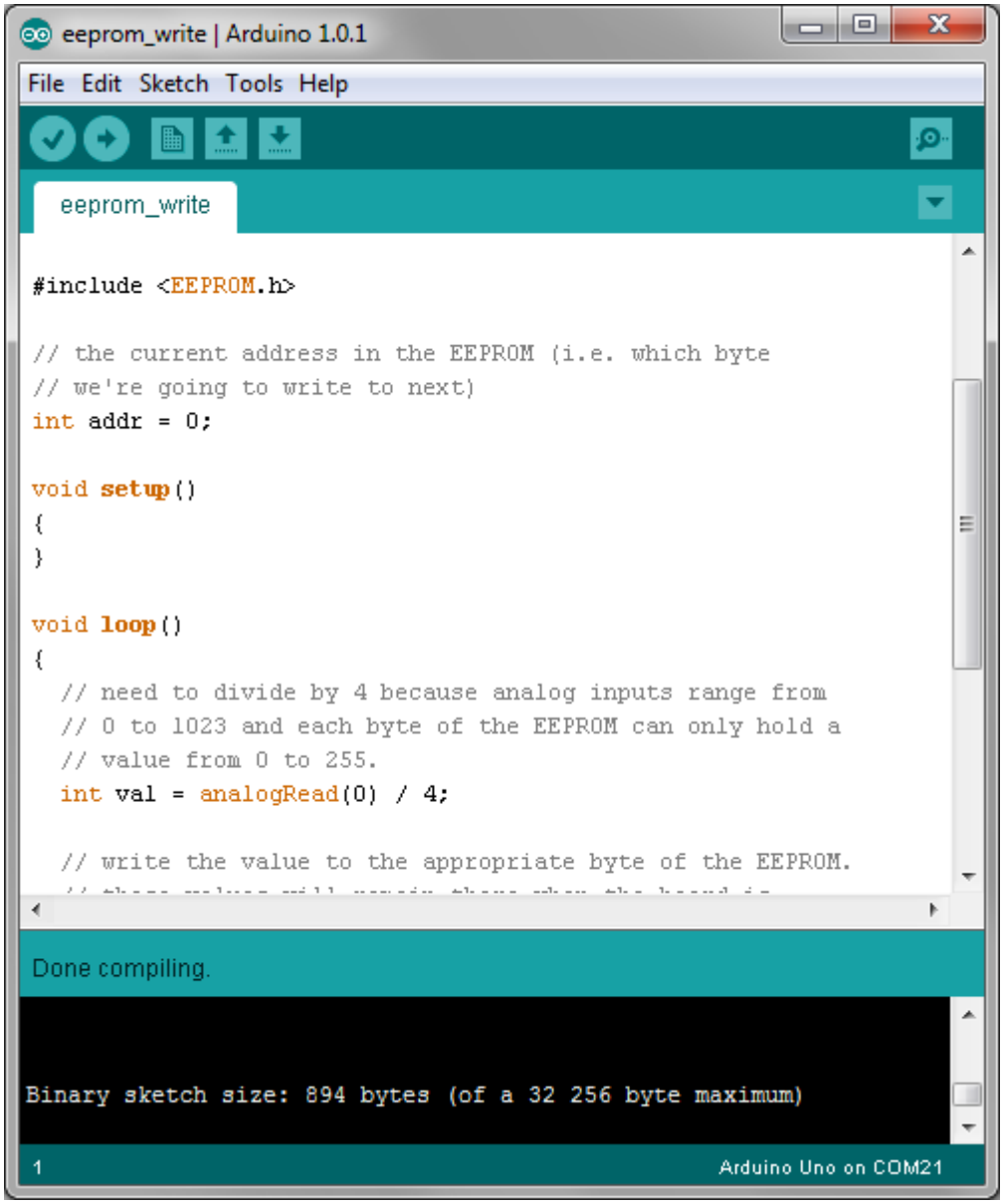
Obr. 4-2 Schéma používateľského prostredia a privilegovanosti používateľov

4.8 Implementačné prostredie

Pre vývojovú platformu Arduino je dostupné open-source implementačné prostredie pre operačné systémy Windows, Mac OS X a Linux. Implementačné prostredie je naprogramované v jazyku Java a využíva softvérové nástroje *Processing*, *avr-gcc* a iné. Zdrojové kódy implementačného prostredia sú dostupné na *Github-e* projektu [45].

Implementačné prostredie Arduino obsahuje jednoduchý editor zdrojového kódu so základnými textovými funkciami. Zdrojový kód je možné rýchlo skompilovať a jednoducho nahrať do rôznych prototypovacích modulov cez USB port. Na chyby v zdrojovom kóde je programátor ihneď upozornený. Vývojové prostredie Arduino zobrazuje po skompilovaní zdrojového kódu aj obsadenosť Flash pamäte na programovanom procesore. Nevýhodou je nemožnosť zistenia pamäťových nárokov na RAM, ktoré je tak možné zistiť len s pomocou vhodnej knižnice počas behu programu. Podrobnejšie informácie o pamäťových nárokoch sa nachádzajú v kapitole s opisom riešenia. Na používateľské vstupno-výstupné operácie je možné použiť vstavanú sériovú konzolu, ktorá je súčasne jediným spôsobom získania spätnej väzby z prototypu. Implementačné prostredie neumožňuje žiadnu formu debugging-u a preto sa programátor môže spoliehať len na konzolové výpisy. Veľkou výhodou implementačného prostredia Arduino sú pravidelné aktualizácie, ktoré prinášajú opravy chýb a dopĺňajú

softvérové knižnice pre podporu najrôznejšieho dostupného hardvéru. Súčasťou knižníc bývajú aj príklady použitia, ktoré rýchlym spôsobom oboznámia programátora so základným spôsobom používania poskytovaných rozhraní. Inštalačné súbory implementačného prostredia spoločne s ďalšími užitočnými utilitami sa nachádzajú na CD nosiči, ktorý je súčasťou tejto práce. Implementačné prostredie je znázornené na nasledujúcom obrázku.



Obr. 4-3 Implementačné prostredie Arduino

5 Opis riešenia

Kapitola s opisom riešenia obsahuje viacero podkapitol, ktoré podrobnejšie charakterizujú vykonanú hardvérovú a softvérovú časť implementácie systému. Nielen počas fázy implementácie čiastkových funkcii ale aj vo fáze testovania sme narazili na niekoľko implementačných problémov, ktoré sú spoločne s ich riešeniami podrobnejšie popísané v nasledujúcej kapitole.

5.1 Problémy spojené s implementáciou

Použité softvérové knižnice a hardvérové prostriedky majú určité nedostatky, s ktorými sme sa stretli až počas implementácie. Keďže sme venovali dostatočne veľa času kvalitnému návrhu, tak nás tieto nedostatky nezaskočili a navrhli sme efektívne riešenie vzniknutých problémov.

5.1.1 Obnova DHCP prenájmu

DHCP server prenajíma IP adresy len na obmedzený čas. Jednoduchá implementácie knižnice *Ethernet* nepočíta s týmto faktorom a sieťové rozhranie si tak privlastní IP adresu natrvalo. V zdrojovom kóde sme implementovali funkciu *Ethernet.maintain(),* ktorá skontroluje časovú platnosť prenájmu IP adresy a v prípade blížiaceho sa limitu požiada DHCP server o obnovu prenájmu. Túto funkciu nevoláme v každom behu slučky *Loop(),* ale len v stanovené časové intervaly pomocou zadefinovanej premennej *DHCP_MAINTAIN_DELAY*. Štandardnú hodnotu sme nastavili na 10 sekúnd, čo šetrí strojový čas. WiFly modul si automaticky obhospodaruje DHCP adresy a preto na externých moduloch s bezdrôtovou konektivitou nie je potrebné vykonávať obnovu IP adries.

5.1.2 Nežiaduce pretečenie Watchdog časovača

Pretečenie Watchdog časovača spôsobí reset celého systému, čo je síce požadovaná funkcionalita, avšak v niektorých prípadoch nežiaduca. Hodnota Watchdog časovača sa určuje na základe najhoršieho prípadu, ku ktorému pripočítame ešte určitú rezervu. Takéto riešenie je možné použiť na deterministický systém, avšak sieť Ethernet je v základe nedeterministická, kde môže oneskorenie nadobúdať ľubovoľné hodnoty. Dobrým príkladom je DHCP server, ktorému môže poskytnutie IP adresy trvať aj niekoľko sekúnd. V takomto prípade nevieme určiť efektívnu hodnotu Watchdog časovača, alebo je jeho maximálna hodnota (8 sekúnd) nedostačujúca. Oblasti zdrojového kódu, ktorých vykonanie môže trvať viac než tento limit označíme za kritické. Pred každou kritickou časťou kódu musíme zrušiť

Watchdog časovač pomocou funkcie *wdt_disable()* a po vykonaní kritického kódu časovač opätovne inicializovať *wdt_enable(WDT_TIME).* Takéto riešenie ale nie je optimálne, keďže kritické operácie nie sú vykonávané s Watchdog ochranou a opätovná konfigurácia časovača zaberie určitý strojový čas.

5.1.3 Pretečenie časovačov

Externé moduly, ale aj centrálna jednotka používa na zaznamenávanie časových pečiatok a intervalov funkciu *millis(),* ktorá vracia počet milisekúnd od spustenia programu. Maximálna veľkosť tohto identifikátora je pritom 32 bitov, takže po približne 50 dňoch od spustenia nastane jeho pretečenie. Pretečenie časovača má prakticky devastačný účinok na celý systém pretože viacero funkcií nie je na to pripravených. Riešenie je pritom veľmi jednoduché, avšak nie úplne elegantné. Počas behu programu pravidelne kontrolujeme tento časový identifikátor a v prípade pretečenia zresetujeme celý systém, čo síce na krátku chvíľu znefunkční modul, avšak obnoví všetky časové premenné a funkcionalitu systému.

5.1.4 Rozsynchronizovanie času

Externé moduly a centrálna jednotka využívajú na ukladanie dát a šifrovanie komunikácie časovú pečiatku, ktorá nemôže byť staršia ako stanovený limit. Platforma Arduino UNO a ani Arduino Mega 2560 neobsahuje hodiny reálneho času. Tento nedostatok kompenzujeme zisťovaním aktuálneho času z NTP servera. Pôvodný návrh počítal s obnovou času až po pretečení časového počítadla, avšak vo fáze overenia čiastkového riešenia sme zistili, že kryštály hodinových cyklov sa medzi jednotlivými modulmi značne odlišujú. Tento nedostatok spôsobuje výrazné časové rozsynchronizovanie modulov a po určitom čase až nemožnosť šifrovanej komunikácie. Z tohto dôvodu sme boli nútení zisťovať aktuálny NTP čas v pravidelnejších intervaloch, ktoré sme zadefinovali v premennej *NTP_RESEND_DELAY*. Štandardne je tento časovač nastavený na 5 minút.

5.1.5 Nedostatok voľnej RAM pamäti

Procesor ATmega 328 disponuje pamäťou SRAM s kapacitou 2 kB. Po inicializácií je pre používateľa dostupných približne 1800 B, čo je veľmi málo, keďže len samotný sieťový paket môže dosahovať veľkosť až 1500 B. Pre vykonávanie programu tak nezostane takmer žiadne voľné miesto v pamäti RAM a použité šifrovanie ešte ďalej zdvojnásobuje pamäťové nároky. Z tohto dôvodu sme obmedzili maximálnu veľkosť paketov na rozumnejších 265 B, čo plne postačuje pre navrhnutý komunikačný protokol a žiadnym spôsobom neobmedzuje funkcionalitu zariadení.

Keďže implementačné prostredie neposkytuje náhľad na využitie pamäte RAM a v prípade jej pretečenia môžeme nastať neočakávané správanie systému, tak sme do programu implementovali knižnicu *MemoreFree*, ktorá zisťuje využitie pamäti RAM v kritických oblastiach programu. Do zdrojového kódu sme na kritické miesta vložili výpis *Serial.println(freeMemory()),* pomocou ktorého sme sledovali stav zaplnenia pamäte. Po zhodnotení výsledkov sme museli určité časti zdrojového kódu preprogramovať pre dosiahnutie vyššej pamäťovej efektivity.

V pamäti RAM sa okrem vyrovnávacích pamätí paketov nachádzajú aj jednotlivé premenné, ktoré majú tiež relatívne vysoké pamäťové nároky. Platí to najmä pre premenné typu *String*, ktoré sa používajú pri konzolových výpisoch alebo sieťovej komunikácii. Napriek tomu, že tieto premenné môžu byť statického typu, tak sa nachádzajú nielen v pamäti Flash, ale aj v RAM. Implementačné prostredie umožňuje zadefinovať tieto premenné pomocou funkcie *F*, ktorá ich uloží len do pamäti Flash a RAM zostane voľná. Funkcia *F* sa používa spoločne s funkciou konzolového výpisu: *Serial.println(F("vypis"))*. Keďže táto funkcia je viazaná len na konzolové výpisy, tak v prípade web servera s množstvom rôznych HTML výrazov sme si museli vytvoriť vlastnú funkciu *char* getString(const char* str)*, ktorá funguje podobným spôsobom a umožňuje jednoducho operovať so staticky zadanými premennými v pamäti Flash. Vstupné premenné je nutné staticky zadefinovať pomocou funkcie *PROGMEM*.

V používateľskom rozhraní používame aj grafické prvky, ktoré sú preberané priamo z centrálnej jednotky. Jednotlivé obrázky sme zakódovali v BASE64 kódovaní a uložili v samostatnom súbore *metaData.h.* Odčlenenie obrázkov spoločne s ďalšími dátovými súbormi zväčšuje prehľadnosť zdrojového kódu a zároveň šetrí dostupnú pamäť RAM, pretože všetky premenné z tohto súboru sú staticky zadefinované v pamäti Flash.

Napriek všetkým vymenovaným spôsobom šetrenia a konsolidácie pamäte RAM sme pri vývoji centrálnej jednotky narazili na neprekonateľný problém absolútnej obsadenosti pamäte s dostupnou kapacitou len 8 kB. Pôvodný návrh počítal s limitom na 256 externých modulov, kde je nutné pre každé zariadenie uchovávať jeho IP adresu, sekvenčné číslo, typ, stav a iné premenné. Vysoký počet modulov spoločne s vysokými pamäťovými nárokmi použitej dátovej štruktúry neumožňoval súčasnú operáciu všetkých modulov systému. Z tohto dôvodu sme museli upraviť požiadavky a znížiť maximálny počet externých modulov na 200. Napriek zmenšeniu limitu je tento počet postačujúci, pretože skôr ako sa vyčerpá počet

modulov, tak systém narazí na ďalšie limity najmä zo strany procesorovej a komunikačnej rýchlosti. V súčasnej implementácií nechávame v operačnej časti zdrojového kódu rezervu približne 1 kB RAM, ktorá by mala postačovať aj pre väčšie pakety a ďalšiu doplnenú funkcionalitu.

5.1.6 Konflikt na sériovej zbernici

Platforma Arduino UNO poskytuje jediné sériové rozhranie, ktoré je použité na programovanie procesora a výpisy cez sériovú konzolu. RFID čítačka taktiež používa sériovú zbernicu. Z tohto dôvodu nie je možné súčasné pripojenie RFID čítačky a programovania mikroprocesora, pretože na sériovej linke vznikajú komunikačné konflikty. Problém je možné vyriešiť dvoma spôsobmi: vo finálnom riešení po naprogramovaní riadiaceho softvéru pripojiť RFID čítačku na uvoľnené sériové rozhranie, ktoré sa už nebude používať alebo vytvoriť softvérové sériové rozhranie. Softvérové rozhranie je náročnejšie na pamäť a strojový čas, avšak pre testovacie prototypy je výhodnejšie. Softvérové sériové rozhranie je možné jednoducho vytvoriť pomocou knižnice *SoftwareSerial*. Vývojová platforma Arduino Mega obsahuje viacero hardvérových sériových rozhraní a preto nie je pre pripojenie GSM shield-u vyžadovaná knižnica *SoftwareSerial*.

5.1.7 Konflikt portov na centrálnej jednotke

Centrálna jednotka využíva súčasne Ethernet aj GSM shield. Napriek možnosti nastavenia niektorých komunikačných portov na GSM shield-e nastáva konflikt na porte D10, ktorý využívajú obidva shield-y. Riešením je odpojenie portu D10 z GSM shield-u a jeho prepojenie na voľný port D8. Zmenu nastavení portov je nutné vykonať aj v použitej knižnici *GSMSHIELD* a súbore *GSM.h.*

5.1.8 Problémy s WiFly

WiFly shield komunikuje s Arduinom pomocou SPI zbernice, cez ktorú sa posielajú riadiace príkazy a aj komunikačné dáta. Medzi komunikačným a konfiguračným režimom sa prepína zaslaním znakov *$$$*. V komunikačnom režime sú riadiacemu procesoru posielané len dáta z aplikačnej vrstvy, čo znemožňuje zisťovanie informácií z nižších vrstiev, akými sú napríklad IP adresa odosielateľa. Z tohto dôvodu nedokáže WiFly modul zistiť IP adresu centrálnej jednotky, na ktorú by mal smerovať všetku komunikáciu. Umožňuje ale posielať správy na posledné zariadenie, ktoré s ním komunikovalo. Táto funkcionalita, ktorá je pomenovaná *UDP Auto Pairing* umožňuje odpovedať na broadcast správy z centrálnej jednotky, avšak je veľmi náchylná na útok generovaním paketov. *UDP Auto Pairing* sa podľa datasheetu [27] nastavuje v konfiguračnom móde pomocou príkazov *set ip host 0.0.0.0* a *set*

ip flags 0x80. Táto konfigurácia ale nebola funkčná a odhalili sme chybu v datasheete, kde mal by uvedený príkaz *set ip flags 0x40*.

S funkciou *UDP Auto Pairing* strácame možnosť synchronizácie času pomocou NTP serverov, keďže WiFly sa po inicializácií naviaže na prvého odosielateľa od ktorého prijal UDP paket. Z tohto dôvodu sme časovú synchronizáciu zabezpečili pomocou Hello správ z centrálnej jednotky. V prípade výpadku centrálnej jednotky je daný modul naďalej funkčný, avšak časom sa zvyšuje jeho časová nepresnosť.

Pri aktivácií funkcie *UDP Auto Pairing* a zmeny IP adresy na centrálnej jednotke sa komunikácia úplne zablokuje, keďže WiFly modul posiela všetky dáta na starú IP adresu. Tento problém sme vyriešili pravidelným zapínaním režimu spánku, ktorý spôsobí nové načítanie komunikačných adries. Aby sme zabránili príliš častému uspávaniu modulu s možnosťou straty komunikačných paketov, tak modul sme nakonfigurovali tak, aby sa do režimu spánku prepol len pokiaľ je komunikácia neaktívna aspoň po dobu jednej minúty. Konfigurácia režimu spánku sa vykonáva príkazmi *set sys sleep <sekúnd>* a *set sys wake <sekúnd>*. Pri prvotnej inicializácií alebo po zobudení vypisuje WiFly modul identifikačné údaje, čo aplikácia nesprávne identifikuje ako prichádzajúce pakety. Takéto informačné výpisy je možné zakázať príkazom *set sys printlvl 0*.

Ďalším problémom bolo záhadné odpájanie z Wi-Fi siete a strata konektivity. Tento problém sme vyriešili aktualizáciou riadiaceho firmvéru, ktorý už nie je dostupný na oficiálnej adrese a museli sme vykonať alternatívnu konfiguráciu pomocou nasledujúcich krokov:

```
set ftp address 0
set dns name rn.microchip.com
save
ftp update
factory RESET
reboot
```

Po aktualizácií riadiaceho firmvéru už problémy s komunikáciou nevznikali, avšak použité WiFly shield-y revízie v1.4 nemajú zapojený resetovací vstup. Pre vykonanie resetu je tak nutné spustiť konfiguračný režim a zadať príkaz *reboot*.

Nevýhodou nemožnosti prístupu k nižším sieťovým vrstvám je aj fragmentovanie paketov, ktoré môže nastávať pri intenzívnej komunikácií. Dáta sú posielané automaticky

a procesor tak nerozhoduje o ukončení paketu a poslaní dát. Fragmentovanie paketu automaticky znamená jeho znehodnotenie, keďže na druhej strane ho nie je možné dešifrovať. WiFly umožňuje 3 spôsoby spúšťania posielania dát: na základe veľkosti dát, špeciálneho znaku alebo časovača. Veľkosť dát a ani špeciálny znak nie sú pre nás použiteľné, keďže posielané dáta majú variabilnú veľkosť a šifrovanie znemožňuje použitie špeciálnych znakov. Môžeme sa tak spoliehať len na časové spúšťanie posielania dát, ktoré je nutné nastaviť na vhodnú hodnotu pomocou konfiguračného príkazu *comm time <milisekúnd>*. Za bezpečnú hodnotu časovača sme označili hodnotu 5 ms. Napriek tomu sa WiFly modul občas správa nevypočítateľne a spája alebo skracuje posielané a prijímané dáta. Tento problém je efektívne vyriešený vďaka navrhnutému protokolu s kontrolou straty paketov a ich preposielaním.

5.1.9 Problémy s dĺžkou konverzie teploty

Pôvodný návrh systému vykonával odčítavanie zo senzorov podľa požiadavky napríklad po prijatí Request paketu. Problémom je čas potrebný na skonvertovanie analógovej teploty na digitálnu hodnotu s maximálnou presnosťou 12 bitov, ktorý je až 750 ms. Počas tejto doby bol systém prakticky nefunkčný a neodpovedal ani na ďalšie požiadavky a preto dochádzalo ku komunikačným chybám a iným problémom. Riešením je buď zníženie presnosti alebo kontinuálne meranie teploty na pozadí. Ako vhodnejšie sme zvolili druhé riešenie, ktoré zaručuje maximálnu presnosť. Pri požiadavke na odčítanie teploty sa odošle posledná načítaná hodnota, čo aj pri sekundovom oneskorení merania nevadí.

5.1.10 Obmedzenia používateľského rozhrania

Ethernet shield použitý v centrálnej jednotke dokáže obslúžiť maximálne 4 TCP relácie. Jedna relácia je používaná na COSM komunikáciu a pre používateľské rozhranie tak zostávajú už len 3 voľné relácie. Po ich vyčerpaní je znemožnená ďalšia komunikácia s novými používateľmi, až kým nedôjde k zrušeniu niektorej z obsadených relácií. Správanie sa niektorých internetových prehliadačov je ale neštandardné a aj v prípade jedinej požiadavky na web stránku otvárajú viacero komunikačných relácií, čo zahlcuje systém. Z tohto dôvodu sme museli upraviť knižnicu *WebServer* a sprístupniť funkciu *m_client.flush(),* ktorú voláme po vykonaní každej požiadavky. Relácie sa tak síce musia zakaždým otvárať nanovo, avšak všetci klienti majú zabezpečený prístup k používateľskému rozhraniu.

5.1.11 Problémy s vysokou intenzitou komunikácie

Pri softvérovej implementácií sme dbali na minimalizovanie používania hardvérových prostriedkov. Výkon použitých procesorov je značne nízky, avšak na špecifikované požiadavky plne postačuje. Problém nastáva v prípade zahltenia systému neplatnou

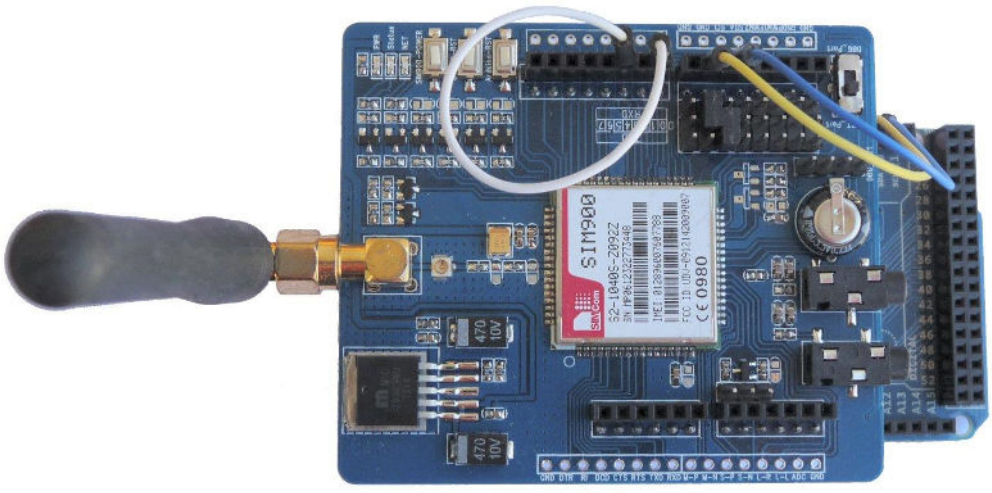
komunikáciou či už vo forme útoku alebo chyby. Z tohto dôvodu sme zdrojový kód upravili takým spôsobom, aby čo najväčšie množstvo prijatých neplatných paketov bolo odfiltrovaných ešte pred začiatkom procesu spracovávania. Samotný proces spracovania a dešifrovania je výkonovo náročný a prípadný útok tak môže rýchlo zablokovať komunikáciu medzi zariadeniami. Softvérovú implementáciu sme upravili tak, aby aj v prípade straty konektivity alebo sieťového útoku externé moduly naďalej vykonávali špecifikovanú funkcionalitu ovládania akčných členov.

5.2 Hardvérová realizácia

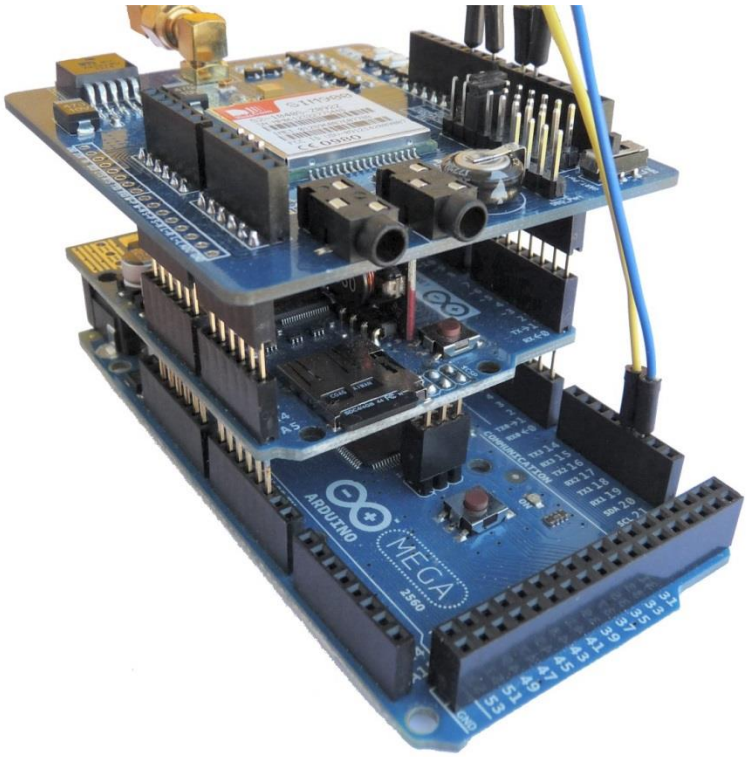
Výsledkom hardvérovej realizácie diplomovej práce sú plnohodnotné shield-y, ktoré je možné priamo použiť s platformou Arduino. Pri návrhu a realizácií dosky plošných spojov sme museli optimalizovať rozmiestnenie a zapojenie jednotlivých komponentov tak, aby bolo možné dané shield-y rozšíriť alebo použiť aj s ďalšími modelmi. Návrh zapojenia tak počíta s možnosťou rôznych kombinácií použitého modelu vývojovej platformy, sieťového shield-u alebo WiFly shield-u. Nasledujúce podkapitoly opisujú detaily o hardvérovej realizácií jednotlivých modulov.

5.2.1 Centrálna jednotka

Centrálna jednotka pozostáva z platformy Arduino Mega 2560, Ethernet shield-u a GSM shield-u, ktorý sme umiestnili na vrchol vzniknutej veže. Keďže na porte D10 nastáva medzi použitými shield-ami komunikačný konflikt, tak na GSM shield-e sme prerušili daný spoj a prepojili ho na port D9. Výhodou platformy Arduino Mega 2560 je viacero dostupných hardvérových sériových rozhraní, ktoré sme takto využili a komunikáciu z portov D5 a D6 presunuli na RX1 a TX1. Napájanie je realizované pomocou POE v Ethernet shield-e.



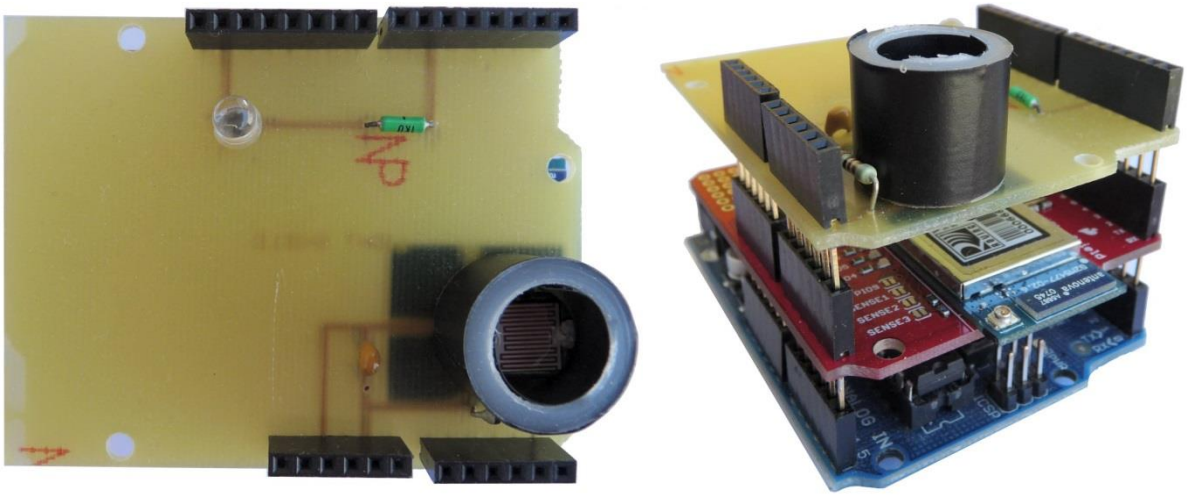
Obr. 5-1 Zapojenie komunikačných portov na centrálnej jednotke



Obr. 5-2 Pohľad na centrálnu jednotku

5.2.2 Svetelný modul

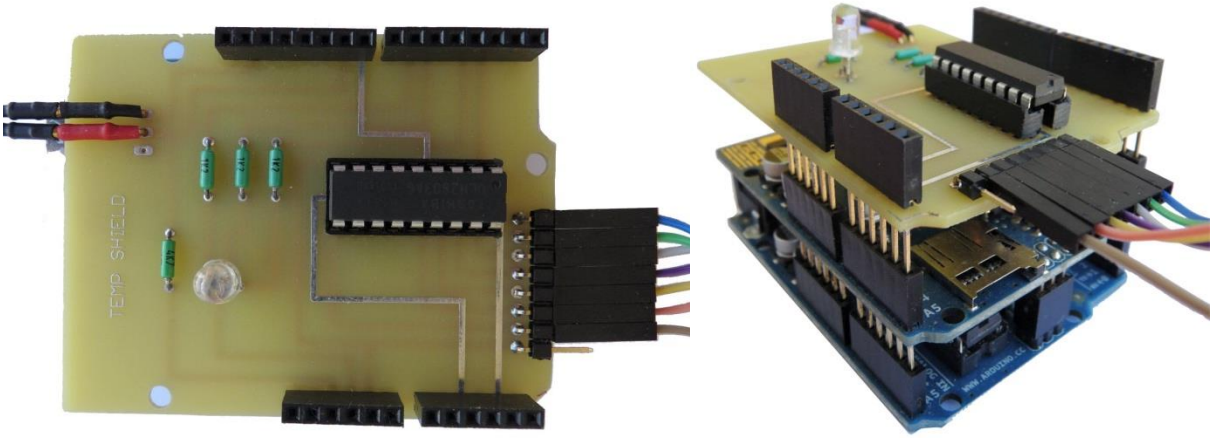
Svetelný modul má spomedzi všetkých externých jednotiek najjednoduchšie zapojenie. K svojej činnosti vyžaduje len napájanie, ktoré realizujeme napájacím adaptérom. Na komunikáciu je použitý WiFly modul. Svetelný senzor a PWM LED osvetlenie sú od seba oddelené pre minimalizovanie miery vzájomného ovplyvňovania a senzor navyše obsahuje svetelné tienenie, ktoré prepúšťa len priamo dopadajúce osvetlenie. Spodná časť senzora je odizolovaná tak, aby nedochádzalo k osvetľovaniu od indikačných diód z WiFly shield-u.



Obr. 5-3 Pohľad na svetelný modul

5.2.3 Teplotný modul

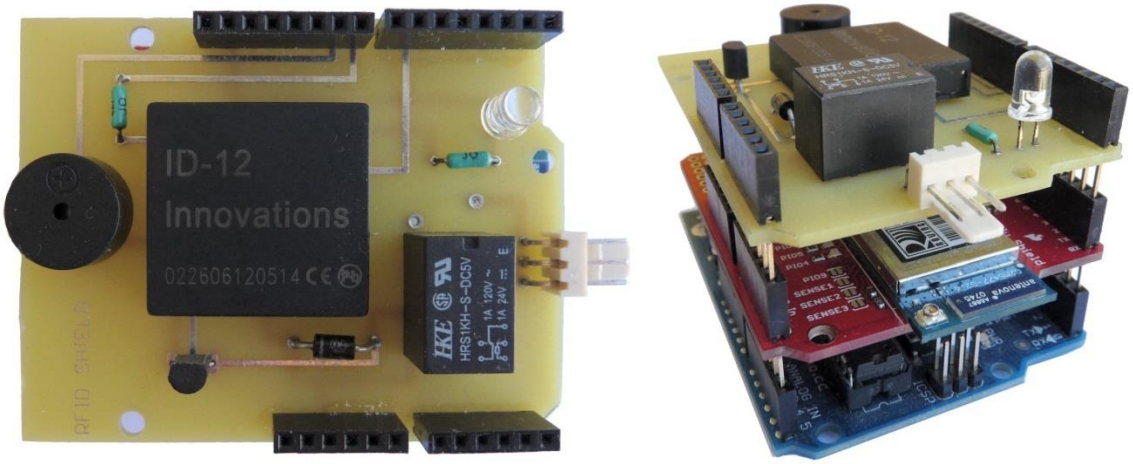
Teplotný modul obsahuje viacero V/V konektorov, ktoré slúžia na napájanie alebo pripojenie teplotného senzora a akčného člena v podobe krokového motorčeka. Vďaka oddeleniu týchto členov od tela modulu je ich možné umiestniť na vhodnejšiu pozíciu. Teplotný senzor sa pripája pomocou dvojice portov v ľavej časti modulu, zatiaľ čo motorček využíva konektor na pravej strane. Na ovládanie motorčeka slúži 6 portov a zvyšné 2 porty sú vyhradené pre napájací zdroj. V našej práci postačuje na napájanie motorčeka aj 9 V napätie, ktoré získavame z POE napájania Ethernet shield-u. Teplotný modul tak k s svojej činnosti nevyžaduje žiaden ďalší zdroj napájania.



Obr. 5-4 Pohľad na teplotný modul

5.2.4 RFID modul

RFID modul komunikuje s centrálnou jednotkou pomocou WiFly shield-u a preto vyžaduje napájací adaptér. V centrálnej časti modulu sa nachádza RFID čítačka a na signalizáciu úspešného čítania RFID kľúča slúži LED dióda s malým bzučiakom. V pravej časti modulu sa nachádza 5 V relé a konektor na pripojenie externého akčného člena otvárania dverí. Energeticky nenáročné relé je napájané priamo z platformy Arduino.



Obr. 5-5 Pohľad na RFID modul

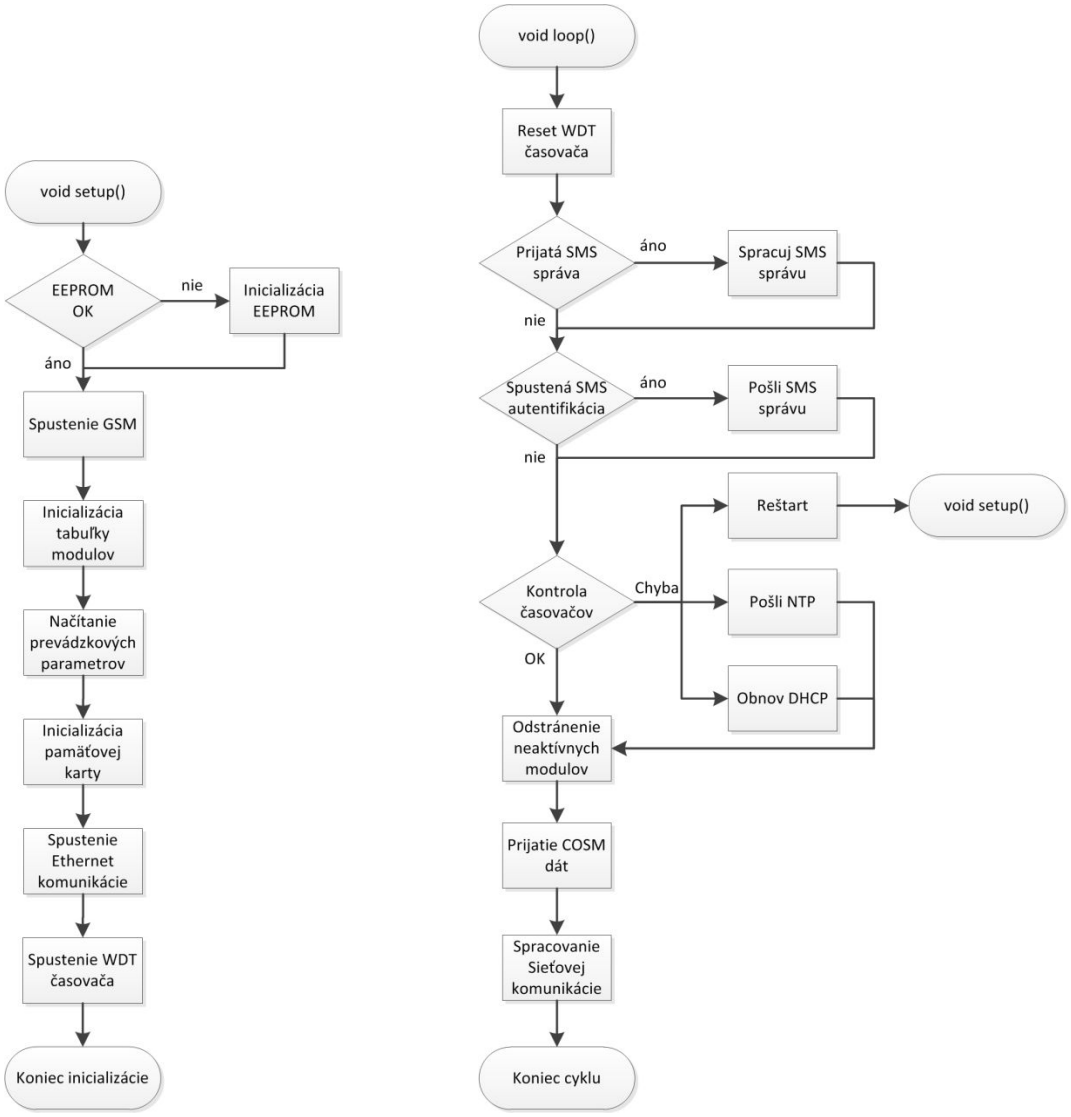
5.3 Softvérová implementácia

V rámci obmedzenia rozsahu dokumentu uvádzame v tejto kapitole len opisy kľúčových implementovaných funkcií, avšak kompletné zdrojové kódy sa nachádzajú na CD nosiči v prílohe práce. Súčasťou riešenia sú aj zdrojové kódy využívajúce softvérové knižnice, ktoré sme opisovali v kapitole s návrhom riešenia. Jednotlivé časti softvérovej implementácie sú hierarchicky rozdelené do viacerých podkapitol.

5.3.1 Operačné diagramy

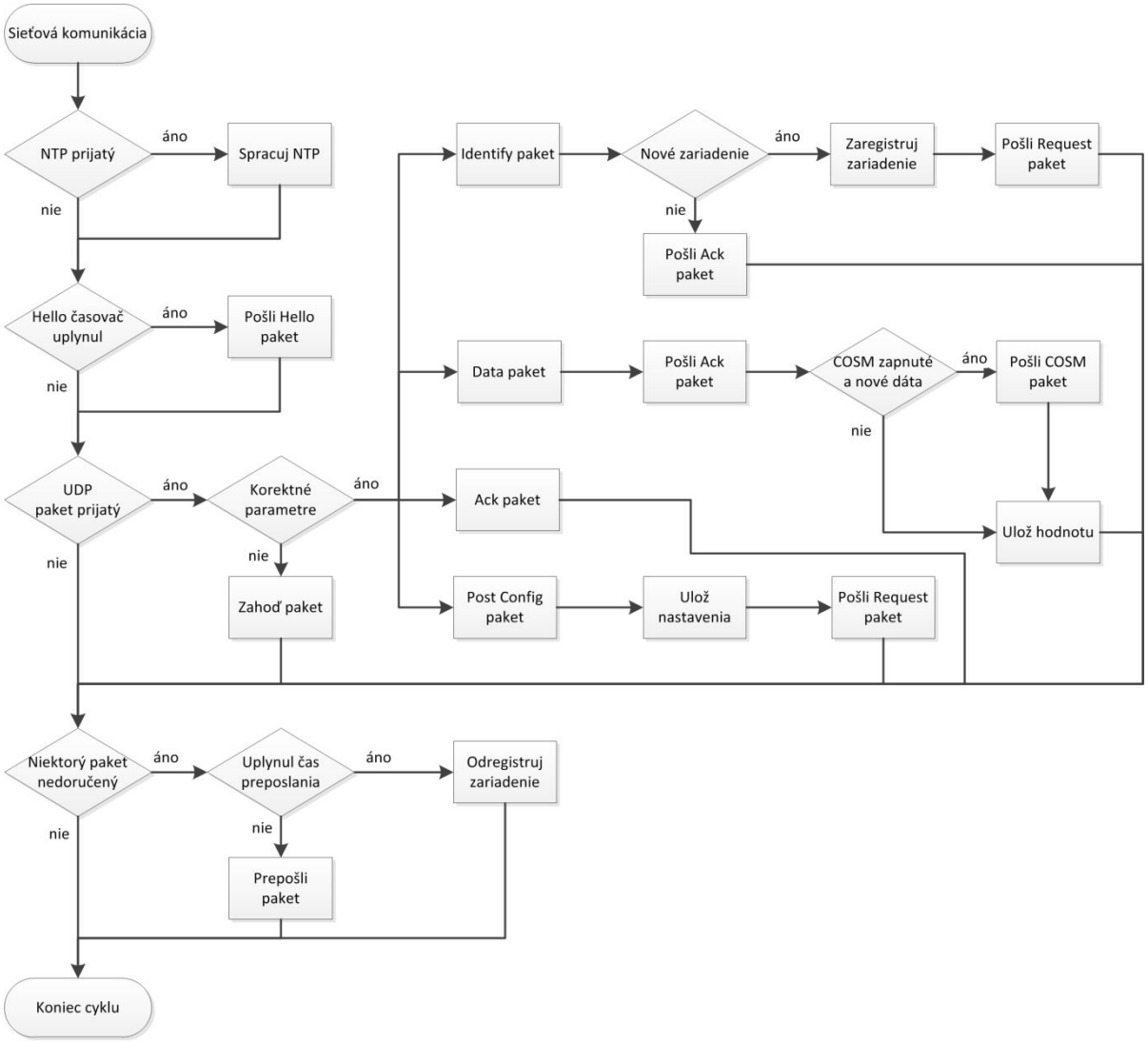
Kapitola s operačnými diagramami znázorňuje logický priebeh inicializačnej a pracovnej slučky. Pri spustení vykonávania zdrojového kódu je najskôr spustená metóda *setup()* a následne v slučke vykonávaná metóda *loop(),* ktorá obsahuje všetky ďalšie funkcie správania.

Centrálna jednotka



Obr. 5-6 Operačné diagramy inicializácie centrálnej jednotky

Po spustení centrálnej jednotky je najskôr overená integrita EEPROM pamäte a následne spustená inicializácia GSM modulu. Po spustení GSM modulu je vytvorená tabuľka externých modulov a načítajú sa všetky prevádzkové parametre z EEPROM pamäte. Až následne sa inicializuje pamäťová karta, sieťová komunikácia a Watchdog časovač. Po ukončení inicializácie sa vykonáva *loop()* slučka, ktorá v každom behu postupne resetuje Watchdog časovač, spracováva SMS správy, kontroluje pretečenie časovačov, označuje neaktívne moduly a vykonáva sieťovú komunikáciu s externými modulmi a COSM serverom.

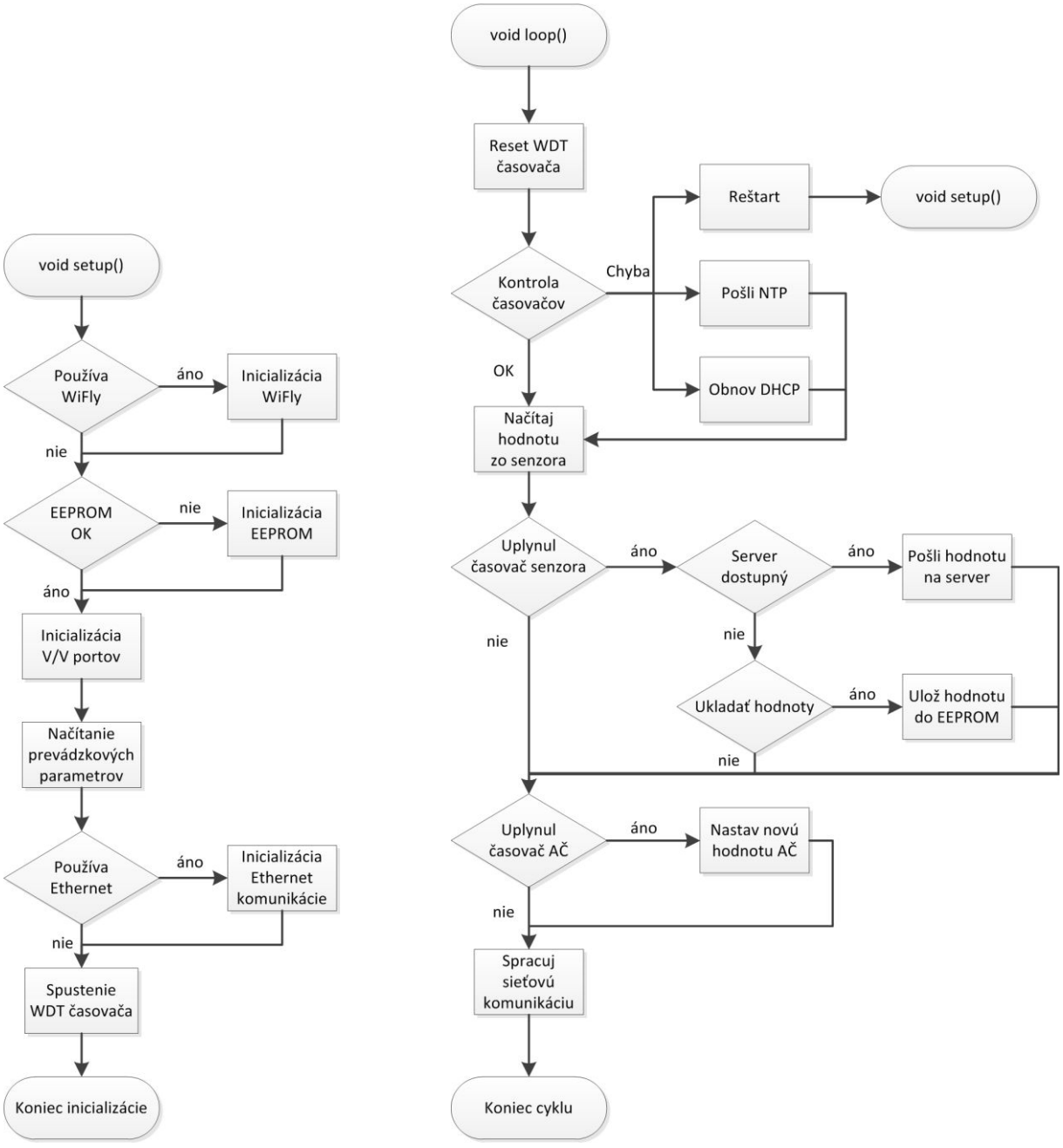


Obr. 5-7 Operačný diagram sieťovej komunikácie centrálnej jednotky

Cyklus sieťovej komunikácie nielen spracováva a odpovedá na prijaté pakety, ale v pravidelných intervaloch posiela Hello správy a kontroluje doručenie predchádzajúcich správ. V prípade komunikačnej chyby a nedoručenia správy je takáto správa znova poslaná. Ak zariadenie neodpovedá dlhšiu dobu, tak je prehlásené za neaktívne a vyradené z komunikácie.

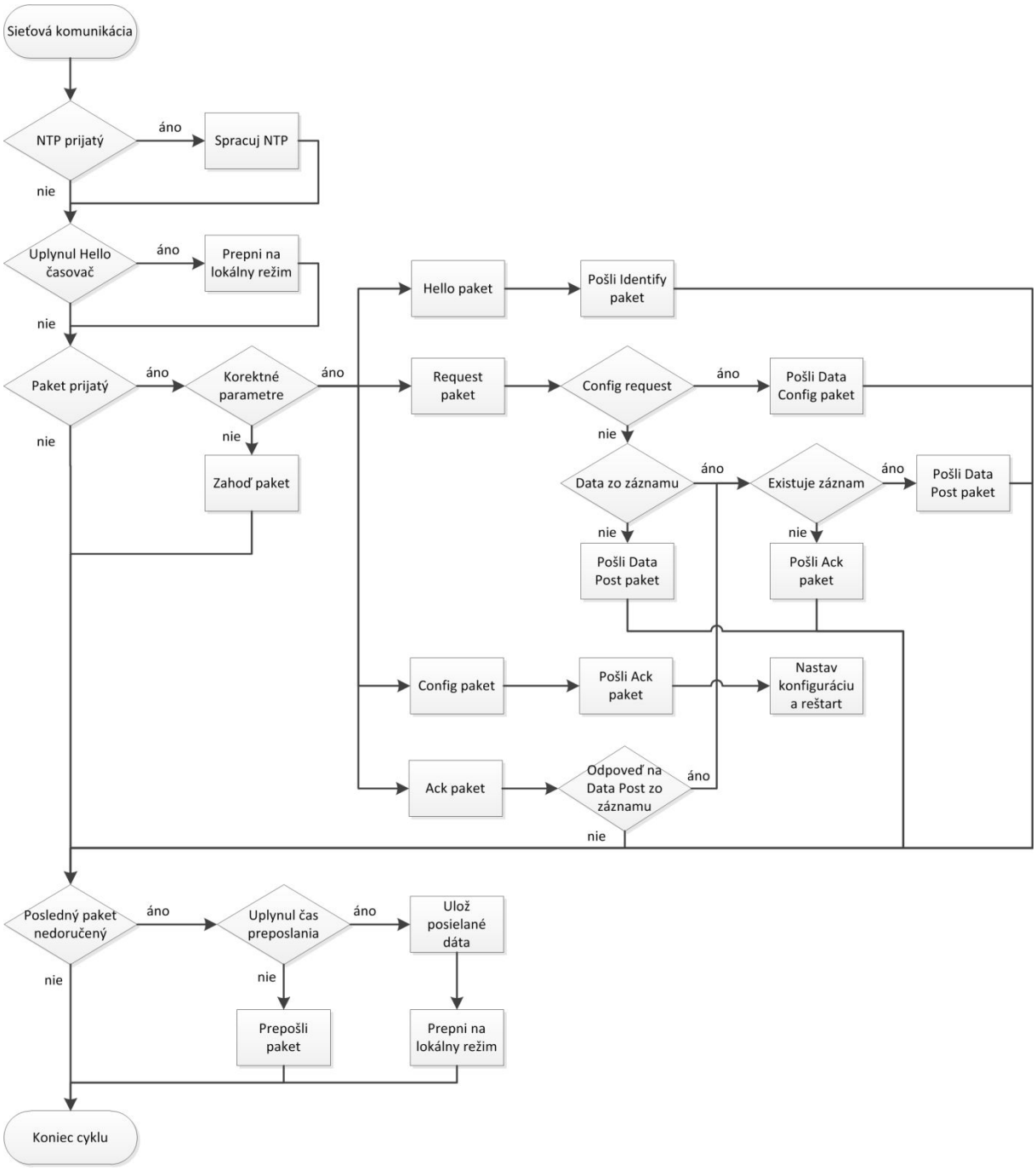
Externé moduly

Po spustení externého modulu je v prípade prítomnosti WiFly shield-u najskôr inicializovaná sieťová komunikácia. Podobne ako aj pri centrálnej jednotke je následne skontrolovaná EEPROM pamäť a potom prebehne inicializácia V/V portov spoločne s načítaním prevádzkových parametrov. V prípade použitia Ethernet shield-u je sieťová komunikácia spustená až na záver spoločne s Watchdog časovačom. Po ukončení inicializácie sa vykonáva *loop()* slučka, ktorá v každom behu postupne resetuje Watchdog časovač, kontroluje pretečenie časovačov, načítava hodnoty zo senzorov, posiela namerané údaje, ovláda akčný člen a vykonáva sieťovú komunikáciu s centrálnou jednotkou.



Obr. 5-8 Operačné diagramy inicializácie externých modulov

Cyklus sieťovej komunikácie má podobnú štruktúru ako v prípade centrálnej jednotky. Najskôr sa spracuje prijatý NTP paket a v prípade straty konektivity sa modul prepne do lokálneho režimu, kde odpovedá len na Hello pakety a zaznamenáva si namerané hodnoty zo senzorov. Externé moduly prijímajú a posielajú len UDP pakety, ktoré sa najskôr zverifikujú, spracujú a následne je na server poslaná odpoveď. Aj externé moduly kontrolujú doručenie predchádzajúcich správ. V prípade komunikačnej chyby a opätovnej neschopnosti doručenia paketu je takáto správa uložená a modul sa prepne na lokálny režim.



Obr. 5-9 Operačný diagram sieťovej komunikácie externých modulov

5.3.2 Komunikačné stavy modulov

Pre detekciu chyby komunikácie alebo nedoručenia správy využívajú všetky moduly identifikačné stavy. Tieto stavy sa dynamicky menia podľa typu očakávanej odpovede a využívajú sa pri opätovnom posielaní strateného paketu. Centrálna jednotka používa rozšírené stavy, kde prvé 4 bity stavu reprezentujú počet zostávajúcich retransmisií a zvyšné 4 bity daný stav. V nasledujúcej tabuľke sú rozpísané jednotlivé stavy.

Stav	**Funkcia**	
	Centrálna jednotka	***Externé moduly***
0x00	Komunikácia v nečinnosti (čaká na *Identify* alebo *Post Data*)	Komunikácia v nečinnosti (čaká na *Hello*)
0x01	Nedefinovaný stav	Čaká na *Ack* (*Identify*)
0x02	Čaká na *Post Data*	Čaká na *Ack* (*Post Data*)
0x03	Čaká na *Post Data* zo záznamu	Čaká na *Ack* (*Post Data* zo záznamu)
0x04	Čaká na *Post Config*	Čaká na *Ack* (*Post Config*)
0x05	Čaká na *Ack*	Nedefinovaný stav

Tab. 5-1 Komunikačné stavy modulov

5.3.3 Implementovaná funkcionalita

V nasledujúcich podkapitolách stručne opíšeme a charakterizujeme implementované funkcie systému. Množstvo funkcií je spoločných pre viacero externých modulov ale aj centrálnu jednotku.

Načítavanie údajov zo senzorov

Hodnoty z použitých senzorov načítavame cez sériovú linku, 1-Wire zbernicu alebo v analógovej forme. Najjednoduchšie sa realizuje načítavanie z ADC prevodníka, na ktorý je pripojený svetelný senzor. Čítanie sa realizuje príkazom *analogRead(A0)*, kde parameter určuje použitý port a ADC prevodník. Keďže ADC prevodník má 10-bitovú rozlišovaciu schopnosť, tak výsledok konverzie môže byť 0 až 1023. Získavanie aktuálnej hodnoty z teplotného senzora zapojeného na 1-Wire zbernici je mierne zložitejšie. Funkcie načítavania hodnôt sú popísané v kapitole s OneWire knižnicou. Výhodou takejto zbernice je možnosť pripojenia viacerých teplotných senzorov, ktoré môžu byť rozmiestnené v okolí externého modulu. Údaje z teplotného a svetelného senzora je nutné načítavať periodicky a preto sme si pre zjednodušenie prístupu zadefinovali metódu *void getSensorValue()*. Táto metóda načíta údaje zo správneho senzora, uloží hodnotu do globálnej premennej *sensorValue* a zapíše čas merania do premennej *measureTime*.

RFID senzor funguje odlišným spôsobom keďže vracia platnú hodnotu len v prípade načítania RFID kľúča. Z tohto dôvodu je taktiež nutné pravidelne volať metódu spracovania, avšak údaje sú zapísané len v prípade pozitívneho načítania kľúča. RFID senzor komunikuje pomocou sériového rozhrania, ktoré je softvérovo emulované cez *SoftwareSerial* knižnicu so zvolenými digitálnymi portami. Pre spracovávanie prichádzajúcich dát sme vytvorili funkciu *boolean TagRead()*, ktorá spracováva a kontroluje prijaté dáta, ktoré následne uloží do globálnej premennej *tagID* a vráti *true*. Prijaté dáta sa následne spracovávajú v ďalších funkciách.

Ovládanie akčných členov

Akčné členy sa rozdeľujú podľa spôsobu ovládania. Zatiaľ čo pri mechanizme bezpečnostných dverí postačuje zopnúť relé a pri LED diódach využiť PWM reguláciu, tak generovanie zvukových signálov pri RFID autentifikácií spoločne s ovládaním krokového motorčeka je už zložitejšie. Zopnutie relé je vďaka tranzistorovému návrhu zapojenia realizované jednoduchým zadefinovaním výstupného pinu pomocou funkcie *pinMode(pin, OUTPUT)* a nastavením na stav s vysokou úrovňou *digitalWrite(pin, HIGH).* PWM regulácia sa aplikuje funkciou *analogWrite(pin, value),* ktorej parameter môže nadobúdať hodnoty od 0 do 255. PWM ovládanie RGB LED diódy sme zabalili do funkcie *void RgbLed(byte red, byte green, byte blue)*, ktorá nastaví zvolené parametre na všetkých portoch. Generovanie zvukového signálu sa realizuje *funkciou tone(SpeakerPin, Frequency, Duration),* ktorej parametre postupne vyjadrujú port s pripojeným bzučiakom, frekvenciu a dĺžku spätnej väzby. Používateľ je tak okamžite informovaný o výsledku načítavania RFID kľúča. Ovládanie unipolárneho krokového motorčeka je realizované postupným zopínaním jednotlivých napájacích fáz, ktoré sú pripojené na vnútorné cievky. Zmena magnetického poľa roztáča rotor požadovanou rýchlosťou. Spätný chod motorčeka je dosiahnutý zopínaním napájacích fáz v opačnom poradí. Ovládanie krokového motorčeka sme zabalili do funkcií *void rotateRwd(int loops)* a *void rotateFwd(int loops)*, ktoré roztáčajú rotor do rôznych smerov. Riadenie akčných členov nie je v súčasnom projekte riešené žiadnou hysteréznou krivkou ani iným riadiacim mechanizmom, avšak v zdrojovom kóde sme aplikovali nastaviteľné oneskorenie riadenia, ktoré umožňuje zabrániť rozkmitaniu riadenia.

EEPROM funkcie

Pamäť EEPROM slúži na ukladanie nastavení modulov a nameraných hodnôt zo senzorov v čase lokálneho režimu. Každý modul používa vlastné nastavenia, ktoré je nutné po naprogramovaní Flash pamäte zapísať aj do EEPROM pamäte. Pre zjednodušenie tohto

procesu sme naprogramovali funkciu *void initializeEEPROM(),* ktorá pamäť najskôr sformátuje a následne zapíše všetky konfiguračné údaje. Zmena nastavení môže byť zrealizovaná po prijatí *Config* paketu, kde sme implementovali funkciu *void setConfiguration(unsigned long* packetBuffer).* Centrálna jednotka používa odlišnú pamäťovú štruktúru a pre správu EEPROM pamäte využíva trojicu funkcií *void set_EEPROM_Default(), void read_EEPROM_Settings()* a *void print_EEPROM_Settings().*

Na správu načítaných dát pri svetelnom a teplotnom module sme vytvorili funkciu zápisu *boolean storeSensorValue()* a čítania *boolean getStoredValue(int* sensorValue, unsigned long* dataTime).* Neplatné dáta je možné odstrániť pomocou funkcie *void removeRecord().* Modul s RFID čítačkou využíva zložitejšiu dátovú štruktúru, ktorá musí uchovávať počet registrovaných kľúčov, hlavný a vedľajšie kľúče, ale aj samotné záznamy z lokálneho režimu. Pri práci s kľúčmi sa využívajú nasledujúce funkcie:

- *boolean getStoredTag(byte* tagRecord, unsigned long* dataTime)*
- *boolean storeTag()*
- *void registerMasterTag()*
- *boolean verifyMasterTag(byte* tagCheck)*
- *boolean searchMasterTag()*
- *byte searchTag(byte* tagCheck)*
- *boolean registerTag()*
- *boolean unRegisterTag()*

Sieťová komunikácia

Funkcie sieťovej komunikácie sú implementované na centrálnej jednotke a aj externých moduloch, avšak ich implementácia sa môže mierne odlišovať podľa typu spracovávaných správ v jednotlivých moduloch. Na inicializáciu sieťového rozhrania slúži funkcia *void setupNetwork().* Následne je nutné v pravidelných intervaloch volať funkcie obnovy DHCP prenájmu *void DHCPcheck()* a *void renewDHCP(int interval)*. Posielanie UDP paketov je realizované viacerými funkciami, ktoré vytvoria dátovú štruktúru, naplnia ju dátami a nakoniec zašifrujú. Funkcie šifrovania sú opísané v samostatnej kapitole. Po úspešnom zašifrovaní dát je vypočítaná kontrolná suma a pridaná časová pečiatka pomocou funkcie *void finishPacket(IPAddress client, EthernetUDP packet, unsigned long* XTEAdataBuffer, byte dataSize, byte packetType, byte ID)*, ktorá vykonáva aj posielanie paketu. Podľa typu posielanej správy sa vyberie vhodná funkcia, do ktorej sa zadajú len platné parametre

a zvyšná logika je vykonávaná vo vnútri funkcií bez nutnosti ďalšieho zásahu. V zdrojovom kóde sme implementovali nasledujúce komunikačné funkcie:

- *void sendHelloPacket(EthernetUDP packet)*
- *void sendIdentifyPacket()*
- *void sendReqPacket(IPAddress client, EthernetUDP packet, byte msgType, byte ID)*
- *boolean sendPostDataPacket(boolean fromRecord)*
- *void sendPostConfigPacket()*
- *void sendConfigPacket(IPAddress client, EthernetUDP packet, byte ID)*
- *void sendAckPacket(IPAddress client, EthernetUDP packet, byte dataSize, byte ackType, byte ID)*

Medzi ďalšie správy posielané cez protokol UDP patrí NTP a WOL. Na posielanie NTP paketov slúži samostatná funkcia *unsigned long sendNTPpacket(IPAddress& address)* a posielanie WOL paketov je realizované podobne ako v prípade posielania UDP broadcast správ. Pre zjednodušenie a sprehľadnenie zdrojového kódu sme pre tento účel vyhradili funkciu *void sendWOLpacket(byte* target),* ktorá vygeneruje Magic Packet a pošle ho UDP broadcastom do siete.

Okrem komunikačného protokolu UDP používame na centrálnej jednotke pri posielaní dát na COSM server a v používateľskom web rozhraní aj protokol TCP. Služba COSM poskytuje HTTP rozhranie pre posielanie dát a požiadaviek na tvorbu grafov. Informácie získané zo senzorov musia byť pravidelne posielané na túto službu, inak sú grafy nekompletné. Na identifikáciu systému sa používajú konštanty *APIKEY* a *FEEDID*, ktoré sú vygenerované pri novej registrácii v službe a zadefinované v zdrojovom kóde. Pre našu prácu máme stanovené nasledujúce konštanty:

- *APIKEY = 6EBEQTvfdlYBs3euWZMFzs1z3SySAKxOUFdXRzJzVExIUT0g*
- *FEEDID = 84641*

Ku COSM serveru môžeme pristupovať priamo pomocou IP adresy *216.52.233.121*, alebo môžeme využiť DNS server s požiadavkou na adresu *api.cosm.com*. V súčasnej implementácii sme za vhodnejšie riešenie zvolili posielanie dát na základe URL adresy. Na posielanie záznamov používame vlastné funkcie *void printCosm(const unsigned char *str)* a *void sendCosm(byte* dataValue, byte devID)*, ktoré automaticky vygenerujú paket požadovaných parametrov a pošlú ho na server.

Používateľské web rozhranie je založené na knižnici WebServer [43], v ktorej sme doplnili a používame viacero funkcií. Medzi najdôležitejšie funkcie patria:

- *void defaultCmd(WebServer &server, WebServer::ConnectionType type, char*, bool)*
- *void accessCmd(WebServer &server, WebServer::ConnectionType type, char **url_path, char *url_tail, bool tail_complete)*
- *void httpNotFound(WebServer &server)*
- *void httpUnauthorized()*

Šifrovanie

Na šifrovanie a dešifrovanie využívame knižnicu XTEA [28], avšak jej funkcionalitu sme rozšírili o generovanie unikátnych šifrovacích kľúčov a dávkové spracovanie vo funkcii *void encryptData(unsigned long* XTEApacketBuffer, int dataSize, unsigned long packetTime)*. Funkcia dešifrovania *boolean decryptPacket(unsigned long* XTEApacketBuffer, byte* packetBuffer, int packetSize)* nevykonáva len obyčajné dešifrovanie, ale spracúva aj prijaté pakety a verifikuje ich kontrolnú sumu a časovú pečiatku. V prípade chyby alebo neplatných parametrov vracia funkcia *False*.

GSM

Služby mobilnej siete GSM využívame v centrálnej jednotke na posielanie SMS správ s autentifikačným kódom a na prijímanie systémových príkazov od používateľa. Pri prvotnom spustení centrálnej jednotky je nutné inicializovať GSM shield pomocou funkcie *gsm.begin(9600)*. Po úspešnej inicializácií je možné posielať SMS správy pomocou funkcie *boolean sendSms(char *data)*. Keďže v našom prípade posielame SMS správy len s autentifikačným kódom, tak implementáciu sme rozšírili o funkciu *boolean sendSmsCode()*, ktorá vygeneruje 4-číselný autentifikačný kód, uloží ho v pamäti a následne pošle používateľovi. Pri každom volaní funkcie *sendSmsCode()* sa vygeneruje nový autentifikačný kód, takže útočník nedokáže vykonať kritickú operáciu bez kompromitácie GSM siete. Ako doplnkový spôsob používateľskej interakcie sme do zdrojového kódu doplnili funkciu *boolean receiveSms()*, ktorá každých 5 sekúnd kontroluje prijaté SMS správy. Ak je od zaregistrovaného čísla prijatá SMS správa s textom „reset“ alebo „restore“, tak centrálna jednotka sa reštartuje, alebo obnoví základné nastavenia. Doplnenie tejto funkcionality umožňuje obnovenie systému aj na diaľku v prípade, že používateľ nastaví nekorektné sieťové parametre a centrálna jednotka je neschopná sieťovej komunikácie.

Debug výpisy

Pre zjednodušenie vyhľadávania chýb a zobrazovanie prevádzkovaných výpisov sme v zdrojovom kóde zadefinovali viacero stupňov Debug výstupov, ktoré sa zobrazujú cez sériovú linku v konzolovom výstupe. Debug výstupy sú rozdelené na rôzne úrovne podľa typu zobrazovaných správ. Základný výpis *DEBUG_MSG* zobrazuje hlavné systémové informácie, akými sú napríklad systémové parametre alebo informačné správy zo senzorov a akčných členov. Výpis *DEBUG_NET* sprístupňuje zobrazovanie informácií o sieťovej komunikácií, prijatých a odoslaných paketoch. Rozšírený výpis *DEBUG_ADV* zobrazuje podrobnejšie informácie ako napríklad obsah prijatých paketov. Chybové výpisy *DEBUG_ERR* a *DEBUG_COM_ERR* obsahujú všetky chybové správy daného modulu. V základnej konfigurácií sú aktívne len chybové výpisy, keďže zvyšné výpisy nemajú pre koncového používateľa žiaden úžitok a len spomaľujú vykonávanie zdrojového kódu. Dôležitou úlohou centrálnej jednotky je aj archivácia chybových výpisov a z toho dôvodu sme implementovali dvojicu funkcií *void Error_P(const char* str)* a *void Error_PC(const char* str)*, ktoré zapisujú chybové hlásenia do súborov na pamäťovej karte. V základnom nastavení sa systémové chyby zapisujú do súboru *ERROR.LOG* a komunikačné chyby do súboru *ERRORCOM.LOG*. Používateľ tak môže priamo z používateľského prostredia nahliadnuť do týchto súborov a odhaliť napríklad autentifikačný útok na systém.

Iné funkcie

Medzi doplnkovú funkcionalitu, ktorú sme implementovali patria rôzne užitočné funkcie akými sú napríklad Reset, zistenie aktuálneho času alebo vyhľadanie identifikátora požadovaného externého modulu. Po vykonaní konfiguračných zmien je na ich aplikáciu nutné vykonať systémový reštart. Na reštartovanie slúži jednoduchá funkcia *void(* device_reset)(void) = 0,* ktorá spôsobí automatické zresetovanie procesora. Niektoré moduly ako napríklad WiFly alebo GSM shield je nutné reštartovať osobitne, nakoľko neobsahujú resetovací vstup, alebo ho majú nakonfigurovaný na odlišnom porte.

Neoddeliteľnou súčasťou všetkých modulov je funkcia *unsigned long CurrentTime(),* ktorá vracia aktuálny čas zistený na základe posledného NTP paketu a časovača *millis()*. V prípade ak modul nedostal odpoveď od NTP servera, tak táto funkcia vracia čas v sekundách od spustenia modulu.

Samostatnou užitočnou funkciou je byte *searchDevID(IPAddress address*), ktorá vracia identifikačné číslo externého modulu na základe IP adresy z prijatého UDP paketu.

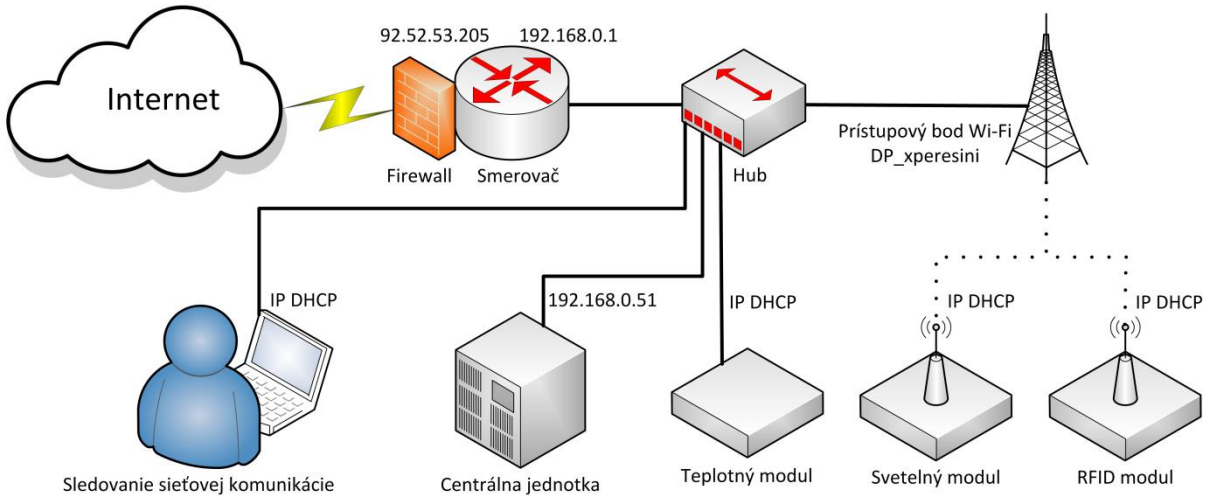
6 Overenie riešenia

Dôležitou súčasťou diplomovej práce je aj časť s testovaním implementovaného riešenia, ktorému sme venovali dostatočnú pozornosť. Počas dlhšieho testovacieho obdobia sme odhalili rôzne chyby, ktoré vznikajú až pri dlhodobej činnosti. Jednou z takýchto chýb bola aj neaktívna obnova DHCP prenájmu, čo v niektorých prípadoch spôsobovalo konflikt pridelených IP adries. Chyby odhalené počas testovania sme bližšie rozanalyzovali a vykonali požadované opravy v zdrojovom kóde. Podrobnejší opis implementačných chýb a problémov sme zaradili do kapitoly s opisom riešenia a problémami spojenými s implementáciou. Vykonané testy spoločne s realizačnými podmienkami sú zhrnuté a stručne opísané v nasledujúcich kapitolách.

6.1 Podmienky testovania

Testy sme realizovali nielen na štandardnej sieťovej topológii ako v prípade návrhu, ale pre detailnejšiu analýzu vzájomnej komunikácie medzi jednotlivými modulmi sme vykonali testy s viacportovým sieťovým opakovačom. Použitie tohto prvku nám umožnilo zachytávať komunikáciu aj na zariadeniach s inou MAC adresou ako je adresát. IP adresy externých modulov sme dynamicky prideľovali pomocou DHCP servera z rozsahu 192.168.0.100 – 192.168.0.200. Testovanie centrálnej jednotky prebiehalo nielen so staticky pridelenou IP adresou 192.168.0.51, ale podobne ako pri externých moduloch sme vyskúšali aj dynamickú adresu. V prípade dynamickej IP adresy na centrálnej jednotke sme nemohli testovať pripojenie z internetu, keďže testovacia sieť sa nachádza za NAT serverom s firewallom, kde sme staticky zadefinovali presmerovanie portu 81 na IP adresu 192.168.0.51. NAT s privátnou IP adresu 192.168.0.1 a verejnou IP adresou 92.52.53.205 tak zároveň slúži ako brána. Po zadaní verejnej adresy 92.52.53.205:81 z ľubovoľného zariadenia na internete by tak malo byť prístupné web rozhranie na centrálnej jednotke. Sieť Ethernet sme taktiež rozšírili pomocou Wi-Fi prístupového bodu, na ktorý sa pripájajú WiFly shield-y z externých modulov. Prístupový bod Wi-Fi funguje v základnom režime, kde medzi sebou prekladá rámce IEEE 802.3 a IEE 802.11. Vytvorená Wi-Fi sieť má SSID „DP_xperesini“ a prístupové heslo „peresini“. Sieť je zabezpečená mechanizmom WPA-Personal. Schéma testovacej sieťovej topológie sa nachádza na nasledujúcom obrázku.

Po vykonaní implementačných testov sme pristúpili k testovaniu systému ako celku s plnohodnotnou funkcionalitou a štandardnou sieťovou topológiou. Pri testovaní sme sa zamerali nielen na pozitívne, ale aj negatívne testovanie.



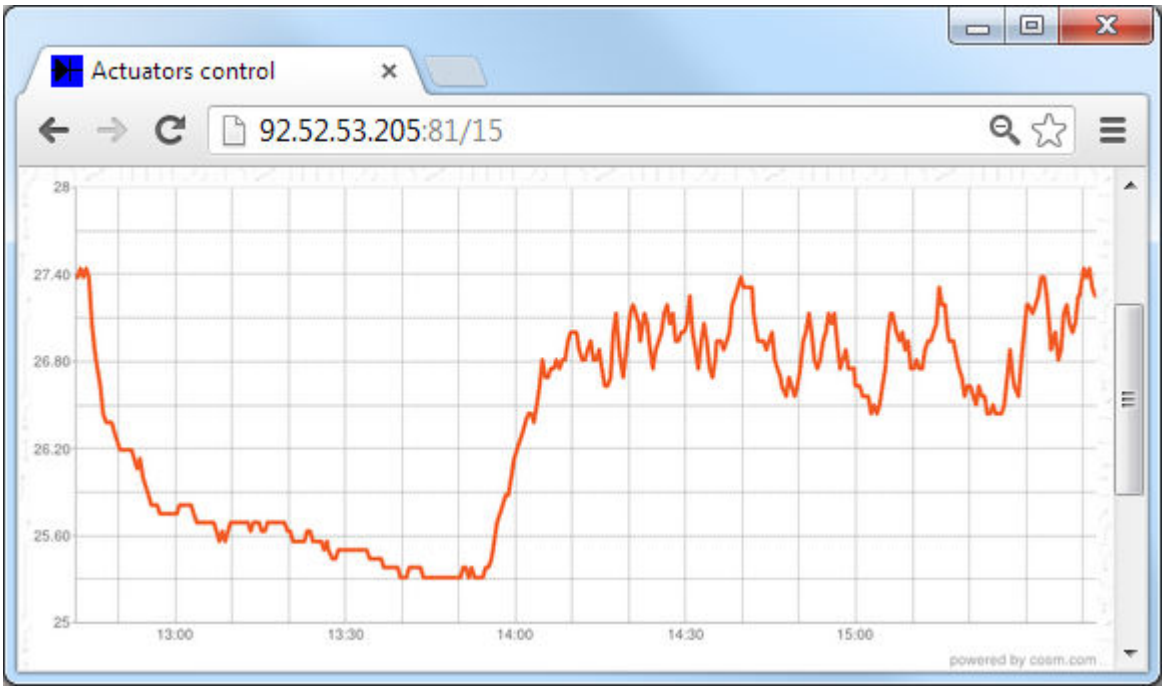
Obr. 6-1 Schéma testovacej sieťovej topológie

6.2 Pozitívne testovanie

Pozitívne testy sú sústredené na otestovanie špecifikovanej funkcionality. Už počas fázy implementácie sme realizovali jednoduché testy, ktoré mali odhaliť rôzne chyby v implementácií alebo návrhu. Najskôr sme sa sústredili na dôkladné otestovanie schopnosti sieťovej komunikácie v rámci sietí Ethernet a Wi-Fi. Následne sme zrealizovali jednoduchú komunikáciu pomocou TCP a UDP protokolov, kde sme zistili rôzne hardvérové obmedzenia použitej platformy. Problémy spôsobovala najmä súčasná komunikácia s viacerými TCP reláciami a jej oneskorenie pri strate spojenia. Prevzatá funkcionalita získavania aktuálneho času z NTP servera bola bezproblémová a nevyžadovala žiadne ďalšie úpravy. Pomocou testovacej sieťovej topológie sme overili korektnosť unicast aj broadcast komunikácie, kde sme zistili a vyriešili problém s WiFly modulom a naviazaním na IP adresu prvého UDP servera. Pri testovaní dynamického prideľovania IP adries z DHCP servera sme odhalili problém s časovým prenájmom, ktorý použitá knižnica úplne ignorovala.

Po úspešných testoch základnej sieťovej funkcionality sme pristúpili k testom navrhnutých externých modulov. Postupne sme testovali správne ovládanie akčných členov a načítavanie zo všetkých senzorov. Pri krokovom motorčeku sme odhalili problém s príliš krátkymi a rýchlymi pulzmi, čo znemožňovalo jeho činnosť. Riešením je vkladanie oneskorenia medzi takýmito pulzmi. Dĺžka oneskorenia sa pre rôzne motorčeky líši a najvhodnejšiu konštantu je možné zvoliť až po dôkladnom otestovaní konkrétneho správania sa. Následne sme úspešne zrealizovali autentifikáciu na základe RFID kľúčov a overili databázu povolených prístupov. Správanie sa RFID modulu zodpovedalo stanovenej špecifikácii. Nezabudli sme overiť ani funkčnosť WOL napájania.

Po úspešných hardvérových testoch sme pristúpili k testovaniu komunikačných správ a zodpovedajúcich stavov jednotlivých modulov. Počas testov sme odhalili drobné pochybenia v implementácií, ktoré sme jednoducho opravili a mohli pristúpiť k testovaniu šifrovanej komunikácie. Od tohto momentu už nebolo možné priamo sledovať dátovú časť posielaných správ, ktorá bola šifrovaná. Korektnosť týchto správ sme si overili dátovými výpismi z modulov, kde prebiehalo dešifrovanie. Šifrovaná komunikácia sa napriek tomu po dlhšom čase desynchronizovala, čo bolo spôsobené vzájomným posunom času v jednotlivých moduloch. Zmerali sme časovú odchýlku medzi jednotlivými modulmi, ktorá bola až 110 sekúnd za deň. Z tohto dôvodu sme zmenšili periódu aktualizácie času z NTP servera. Následne sme pristúpili k testovaniu spojenia s COSM serverom. Výsledkom úspešného testovania boli prvé grafické priebehy meraných veličín. Výsledok testu je znázornený na nasledujúcom obrázku.

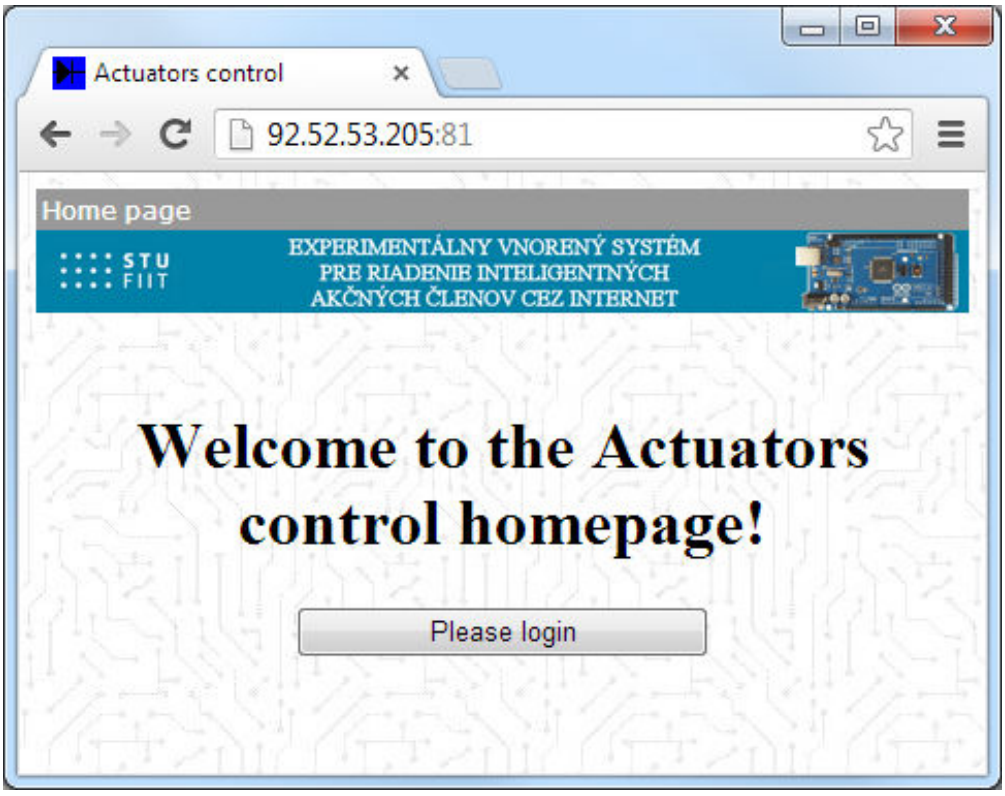


Obr. 6-2 Grafický priebeh merania teploty získaný službou COSM

Podobne sme testovali aj posielanie nameraných hodnôt, ktoré boli uložené v EEPROM pamäti externých modulov. Testy sme sústredili na overenie správnej funkčnosti čítania a zápisu všetkých dátových štruktúr v EEPROM pamäti. V tejto časti sme taktiež overili aj funkcionalitu Wake On Lan, kde sme zaznamenali problémy s niektorými koncovými zariadeniami. Zatiaľ čo stolný počítač umožňuje prebudenie z režimu spánku a má takéto nastavenie aj v BIOS-e, tak na druhom notebooku nebolo prebúdzanie funkčné. Problém spočíva v samotnom notebooku, ktorý nepodporuje túto funkcionalitu.

Ďalším krokom bolo otestovanie prístupu na SD kartu, kde sme overovali správnosť zapisovaných dát. Centrálna jednotka vytvára pri svojej činnosti viacero súborov, ktoré sú rozdelené na chybové záznamy, dátové záznamy a identifikačný súbor konkrétnych externých modulov. Tieto súbory je možné stiahnuť buď cez používateľské rozhranie, alebo prečítaním SD karty. Obidva popisované spôsoby fungujú korektne, avšak napriek zväčšeniu vyrovnávacích pamätí na centrálnej jednotke sa rýchlosť sťahovania pohybuje len okolo 100 kB/s. Táto rýchlosť nemusí vždy postačovať na väčšie súbory, keďže v prípade sťahovania súboru je činnosť centrálnej jednotky pozastavená a komunikácia s externými modulmi prerušená.

Pri implementácií používateľského rozhrania sme narazili na problém s nedostatkom voľnej pamäti RAM, ktorý sa prejavoval náhodnými reštartmi systému alebo poškodením dátových štruktúr. Museli sme pristúpiť k optimalizácií zdrojového kódu a použitých dátových štruktúr. Výsledky optimalizácie sme overovali pomocou knižnice MemoryFree [41]. Aj napriek tomu sme pri implementácií ostatných modulov narazili na nedostatok pamäťovej kapacity a boli sme nútení znížiť maximálny počet externých modulov na 200. V tejto časti testov sme overili aj správnu funkcionalitu Watchdog časovača a funkciu resetovania modulu.



Obr. 6-3 Overenie funkčnosti používateľského rozhrania pri prístupe z internetu

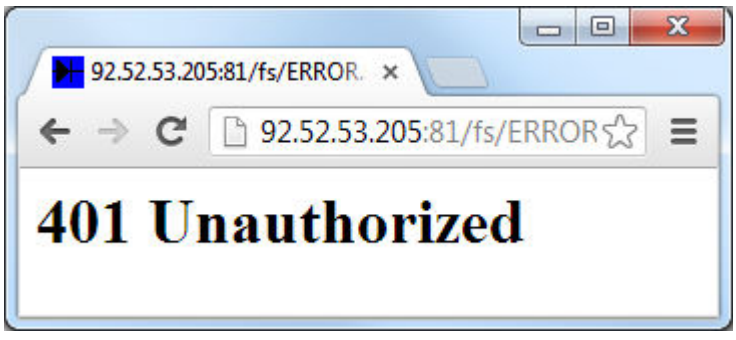
Na záver sme si nechali testovanie používateľského rozhrania, kde sme overili všetky dostupné funkcie a celkové správanie sa systému z dlhodobého hľadiska. Súčasťou tohto testovania bolo aj overenie funkčnosti GSM autentifikácie a prijímania SMS správ.

6.3 Negatívne testovanie

Negatívne testovanie je sústredené na overenia správania sa systému v prípade neplatných alebo chybných požiadaviek. Najdôležitejšie negatívne testy boli sústredené na bezpečnosť, kde sme simulovali posielanie vyfabrikovaných správ, ale aj neautorizovaný prístup v používateľskom rozhraní. Systém bol vďaka dobrému návrhu schopný odolávať takýmto útokom a šifrované správy nie je prakticky možné jednoduchým spôsobom sfalšovať. Nemenej dôležitým testom bolo aj simulovanie preťaženia siete, ktorá zahadzuje UDP pakety a schopnosť systému vysporiadať sa s týmto problémom. Keďže navrhnutý komunikačný protokol obsahuje identifikátory sekvenčného čísla a časovej pečiatky, tak aj v prípade krátkodobej straty konektivity bol systém naďalej plne funkčný. V prípade dlhodobejšieho výpadku sa po obnovení komunikácie retrospektívne prepošlú všetky stratené správy.

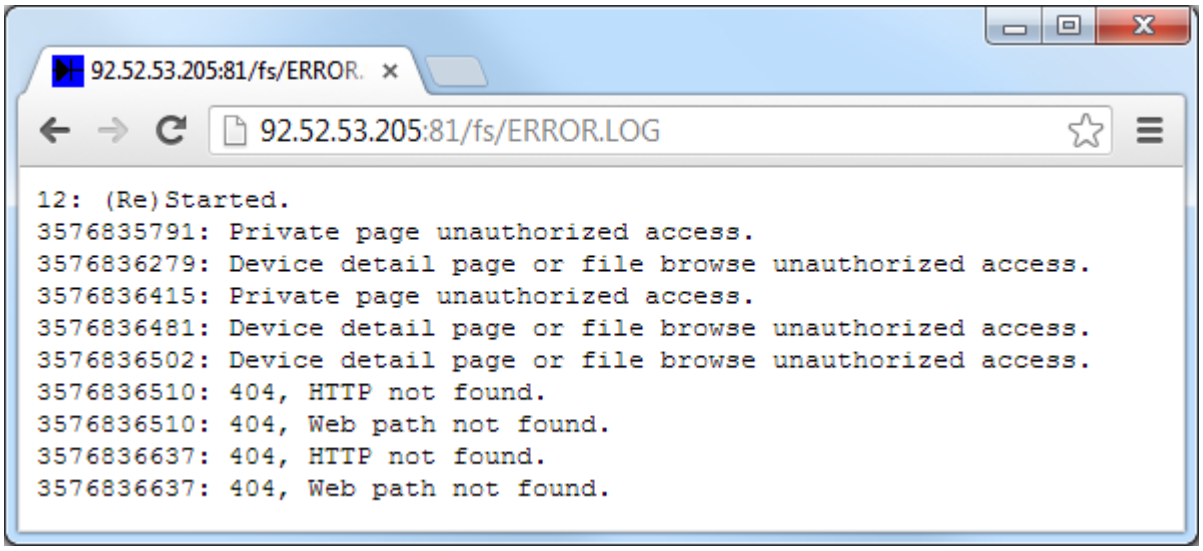
V negatívnych testoch hardvérových prostriedkov sme testovali správanie sa systému po vybratí pamäťovej SD karty. Systém bol naďalej funkčný, avšak neumožňoval ukladať namerané údaje a reakčná doba sa o niečo zvýšila, keďže každý neúspešný pokus prístupu k pamäťovej karte stojí určitý strojový čas. V prípade odobrania niektorého zo shield-ov sa daný modul úplne znefunkční, nakoľko bez sieťovej komunikácie alebo senzorov nemá prevádzka takýchto modulov ani význam. Odpojenie akčných členov nemá na prevádzku systému žiaden vplyv okrem nemožnosti regulácie prostredia.

V poslednej fáze negatívneho testovania sme overovali správanie sa používateľského rozhrania v prípade neplatného alebo neautorizovaného prístupu. V prípade prístupu do privátnej sekcie bez potrebných prihlasovacích údajov vypíše systém chybu autorizácie.



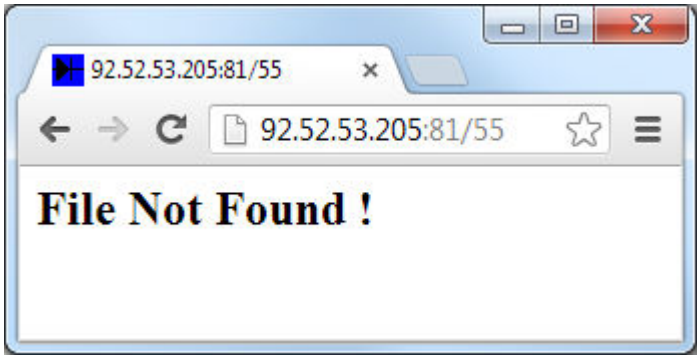
Obr. 6-4 Chyba autorizácie pri prístupe do systému s chybnými prihlasovacími údajmi

Všetky systémové chyby sa ukladajú na pamäťovú kartu do súboru *ERROR.LOG*. Tieto chyby je možné kedykoľvek zobraziť priamo z používateľského rozhrania. Súčasťou chybového výpisu je aj časová pečiatka, ktorá umožňuje presné datovanie chyby. Komunikačné chyby sa zapisujú do súboru *ERRORCOM.LOG*, ktorého veľkosť rastie podľa chybovosti spojenia. V súbore sa nachádzajú záznamy o odpojení externého modulu, straty a chyby spojenia, retransmisie správ a iných komunikačných problémov. Vo viacerých negatívnych testoch sme overili správanie sa systému v prípade rôznych chýb spojenia.



Obr. 6-5 Používateľské rozhranie umožňuje zobraziť chybové záznamy

V prípade požiadavky na neexistujúci súbor alebo neplatnú URL adresu vypíše systém chybu prístupu. Okrem chybných URL požiadaviek sme otestovali aj správanie sa systému v prípade prijatia neplatnej SMS požiadavky. Vyhodnocuje sa nielen platnosť príkazu, ale aj číslo odosielateľa a príkaz je vykonaný iba pri splnení obidvoch požiadaviek. Používateľské rozhranie sme taktiež otestovali na rôznych zariadeniach s rôznymi internetovými prehliadačmi a až na niekoľko chýb vo formátovaní bol výstup všade korektný.



Obr. 6-6 Prístup na neplatnú URL adresu alebo súbor končí chybovým výpisom

7 Zhodnotenie

V rámci práce sa nám podarilo navrhnúť a implementovať experimentálny vnorený systém pre riadenie inteligentných akčných členov cez internet. Všetky požiadavky plynúce zo zadania sme úspešne splnili. Navrhnutý systém umožňuje realizovať autonómne a inteligentné akčné členy za zlomok bežnej komerčnej ceny a je pripravený na budúce rozšírenie. Oproti komerčným produktom poskytuje navrhnutý systém rôzne formy zabezpečenia nielen v podobe šifrovania komunikačných správ. Nesporným benefitom navrhovaného systému je aj vysoká škálovateľnosť a modulárnosť, keďže systém prakticky neobsahuje limit na použitý počet externých modulov a je limitovaný len veľkosťou pamäte RAM v použitom procesore. Vysoká škálovateľnosť a štandardizované sieťové alebo bezdrôtové rozhranie dovoľuje jednoduché pridávanie množstva externých modulov s rôznymi senzormi a akčnými členmi. Napriek tomu nie je nutná zmena alebo dobudovanie ďalšej sieťovej infraštruktúry.

V prvej časti dokumentácie k diplomovej práci sme podrobne analyzovali všetky časti a komponenty, ktoré sme pri riešení práce používali. Analyzovali sme diplomové práce s podobným zameraním a komerčne dostupné systémy, pri ktorých sme zdôvodnili ich výhody a nevýhody. V nasledujúcej analýze sme sa zamerali na výber vhodného procesora a vývojovej platformy spoločne so všetkými modulmi, ktoré rozširujú funkcionalitu pôvodnej platformy. Vybrali a analyzovali sme najvhodnejšie sieťové, bezdrôtové, napájacie a GSM shield-y a moduly. Ďalšia časť analýzy je venovaná sieťovým komunikačným protokolom, ktoré sú používané pri vzájomne sieťovej komunikácií medzi modulmi. Okrem toho sme analyzovali rôzne spôsoby zabezpečenia systému voči rôznym druhom prienikom spoločne s preventívnymi opatreniami. Samostatná časť analýzy je vyhradená senzorom prostredia a akčným členov, ktoré používame v tejto práci.

V podrobnej špecifikácií sme stanovili všetky aspekty systému, jeho správanie a reakcie. V špecifikácií sme určili typy použitých externých modulov a charakterizovali centrálnu jednotku. Veľká časť zo špecifikácie je venovaná opisu sieťových komunikačných správ a ich vzájomných náväzností. Kapitola špecifikácie taktiež pojednáva o obmedzeniach sieťovej komunikácie, ktorá plynie z použitých hardvérových prostriedkov. Samostatná časť kapitoly je vyhradená opisu zabezpečenia a spôsobu monitorovania a riadenia prostredia. Na záver kapitoly je zadefinované používateľské rozhranie a zhrnuté všetky hlavné benefity navrhovaného systému.

Kapitola s návrhom riešenia bližšie opisuje návrh systému podľa viacerých kľúčových aspektov špecifikácie. V kapitole je predstavený konceptuálny model systému a návrh hardvérových schém jednotlivých modulov. Nevyhnutnou časťou návrhu je aj detailný popis štruktúry komunikačného protokolu a rozdelenia dostupnej pamäte. Zvyšná časť kapitoly sa sústreďuje na opis použitých softvérových knižníc a zdrojových kódov. Samostatná časť kapitoly je vyhradená pre schematický model používateľského rozhrania. V závere kapitoly je opísané implementačné prostredie.

Piata kapitola s opisom riešenia pojednáva najmä o problémoch spojených s implementáciou a opisuje realizáciu testovacích prototypov na softvérovej a aj hardvérovej úrovni. Softvérová časť kapitoly predstavuje operačné diagramy jednotlivých modulov systému, pojednáva o komunikačných stavoch a opisuje implementovanú funkcionalitu splňujúcu stanovené špecifikácie.

Nemenej dôležitou kapitolou je aj overenie riešenia, kde sme predstavili podmienky vykonaných testov a opísali rôzne spôsoby pozitívneho, ale aj negatívneho testovania. Výsledky čiastkových testov, ktoré sme vykonali už počas implementácie, odhalili niekoľko drobných problémov, ktoré sme úspešne opravili. Výsledný systém je dostatočne robustný a vhodný aj na priame nasadenie do reálneho prostredia, keďže výsledky vykonaných testov na plnohodnotnom systéme neodhalili ani z dlhodobého hľadiska žiaden významný problém.

Súčasťou diplomovej práce je aj technická dokumentácia, ktorá obsahuje nielen používateľskú, ale aj programátorskú a inštalačnú príručku. Zatiaľ čo používateľská príručka opisuje používateľské rozhranie, tak programátorská a inštalačná príručka pojednáva o konfiguračných premenných a implementačnom prostredí. Samostatná časť kapitoly je venovaná opisu konfigurácie WiFly modulu.

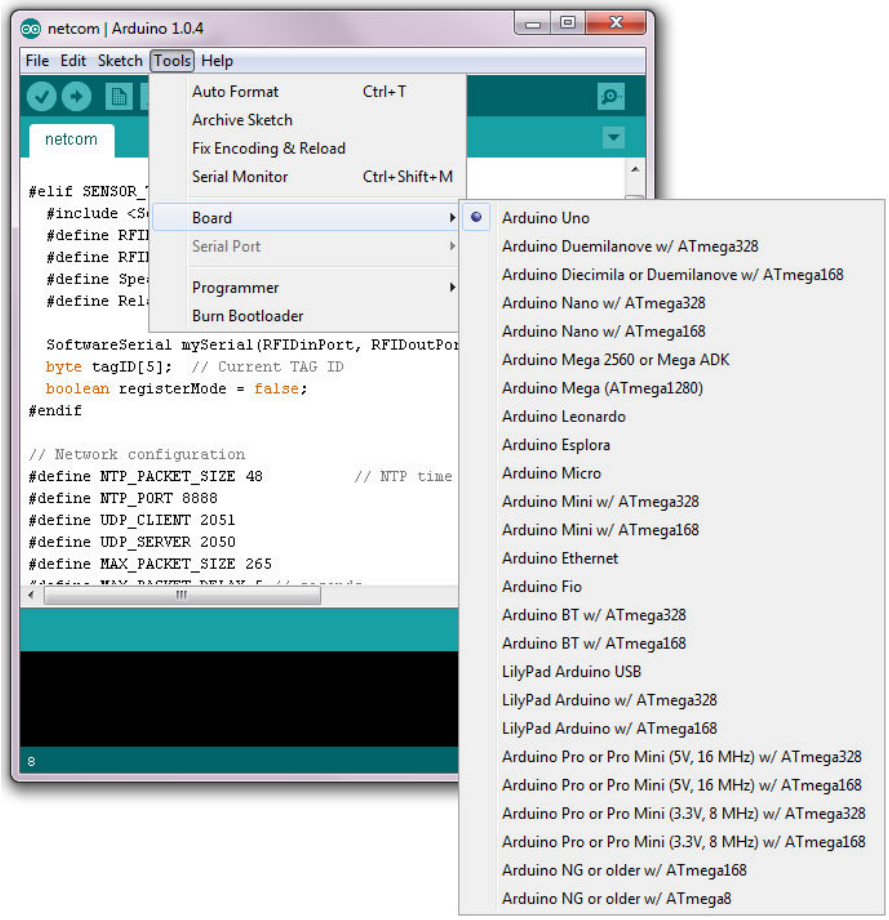
Diplomová práca je vo viacerých aspektoch pripravená na budúce rozširovanie funkcionality a zlepšovanie poskytovaných služieb. Výsledné riešenie tak predstavuje vhodného kandidáta na dopracovanie z fázy prototypu až k reálnemu komerčnému nasadeniu, ktoré môže mať z dôvodu bohatej funkcionality a nízkej ceny značný úspech.

8 Technická dokumentácia

Kapitola s technickou dokumentáciou obsahuje programátorskú a používateľskú príručku. Programátorská príručka je určená používateľom, ktorí chcú vykonať zmeny v programovej výbave systému alebo pozmeniť štandardné správanie sa systému. Používateľská príručka bližšie opisuje používateľské rozhranie a všetky jeho dostupné funkcie.

8.1 Programátorská a inštalačná príručka

Na vykonanie úprav, skompilovanie a naprogramovanie zvoleného modulu je najvhodnejšie použiť implementačné prostredie Arduino [45]. Prostredie nie je nutné inštalovať, ale postačuje ho len jednoducho spustiť a otvoriť existujúci projekt. Zdrojové súbory projektu môžu byť umiestnené na ľubovoľnom mieste, avšak pred kompiláciou je potrebné do adresára *Arduino\libraries* nakopírovať zdrojové súbory použitých knižníc. Všetky použité knižnice aj s potrebnými modifikáciami sa nachádzajú na priloženom médiu. Následne je potrebné zvoliť použitú platformu cez kontextovú položku *Tools->Board->Arduino Uno* alebo *Arduino Mega 2560*. Nastavenie použitej platformy je znázornené na nasledujúcom obrázku.



Obr. 8-1 Pred kompiláciou zdrojového kódu je nutné zvoliť použitú platformu

V prípade súčasného pripojenia viacerých vývojových platforiem je taktiež nutné zašpecifikovať správny sériový port cez kontextovú položku *Tools->Serial Port*. Po skompilovaní je možné modul naprogramovať a pomocou konzoly sledovať debug výstupy.

8.1.1 Zmeny v konfiguračných nastaveniach

Prvotná konfigurácia modulu prebieha pomocou konfiguračných definícií, ktoré sa nachádzajú na začiatku zdrojového kódu. V nasledujúcej tabuľke sú popísané základné konfiguračné parametre a ich štandardné nastavenie pre centrálnu jednotku.

#define	Štandardná hodnota	Funkcia
DEBUG_MSG	*False*	Systémové výpisy
DEBUG_ERR	*True*	Systémové chybové výpisy do súboru
DEBUG_NET	*False*	Komunikačný výpisy
DEBUG_ADV	*False*	Podrobné komunikačné výpisy
DEBUG_COM_ERR	*True*	Komunikačné chybové výpisy do súboru
CONFIG_REWRITE	*False*	Prepísanie nastavení v EEPROM
WEBDUINO_AUTH_REALM	*Insert autentification data*	Text posielaný pri autentifikácií
NTP_PACKET_SIZE	*48 (b)*	Veľkosť NTP paketu
MAX_PACKET_SIZE	*265 (b)*	Maximálna veľkosť paketu
NTP_PORT	*8888*	Port NTP klienta
UDP_CLIENT	*2051*	Port UDP klienta
UDP_SERVER	*2050*	Port UDP serveru
MAX_PACKET_DELAY	*5 (s)*	Maximálne oneskorenie paketu
NTP_RESEND_DELAY	*300 (s)*	Perióda posielania NTP paketov
NTP_RETRY_DELAY	*5000 (ms)*	Perióda retransmisie chybného NTP
COM_RETRY_DELAY	*500 (ms)*	Perióda retransmisie chybného paketu
DEVICE_ACTIVE_TIME	*32000 (ms)*	Maximálny čas aktivity externého modulu pri strate konektivity
HELLO_PERIOD	*5 (s)*	Perióda posielania Hello paketu
WDT_TIME	*WDTO_8S*	Perióda Watchdog časovača

Tab. 8-1 Základné konfiguračné parametre centrálnej jednotky

Zmenou týchto konfiguračných parametrov je možné pozmeniť správanie sa systému napríklad pri nekvalitnom Wi-Fi spojení kde dochádza k častým stratám UDP paketov. Zmenu zvyšných sieťových parametrov a šifrovacieho kľúča pre centrálnu jednotku je možné vykonať vo funkcii *set_EEPROM_Default()*.

Základné konfiguračné parametre pre externé moduly sú popísané v nasledujúcej tabuľke.

#define	Štandardná hodnota	Funkcia
WIFLY	*False*	Použitý je WiFly alebo Ethernet shield
DEBUG_MSG	*False*	Systémové výpisy
DEBUG_NET	*False*	Komunikačný výpisy
DEBUG_ADV	*False*	Podrobné komunikačné výpisy
DEBUG_ERR	*True*	Chybové výpisy
CONFIG_REWRITE	*False*	Prepísanie nastavení v EEPROM
SENSOR_TYPE	*0x01*	Typ senzora (1 = svetelný, 2 = teplotný, 3 = RFID)
ID	*1*	Identifikačné číslo senzora (1 – 199)
SENSOR_NAME	*EthLightSens*	Meno senzora (max. 12 znakov)
KEY1 – KEY4	*X, Y, Z, Q*	Šifrovací kľúč
RECORD_PERIODICITY	*139 (s – 128)*	Perióda snímania senzora a posielania dátových paketov
ACTUATOR_TARGET	*450*	Cieľová hodnota akčného člena
CONTROL_DELAY	*5 (s)*	Oneskorenie akčného člena
NTP_PACKET_SIZE	*48 (b)*	Veľkosť NTP paketu
NTP_PORT	*8888*	Port NTP klienta
UDP_CLIENT	*2051*	Port UDP klienta
UDP_SERVER	*2050*	Port UDP serveru
NTP_RESEND_DELAY	*300 (s)*	Perióda posielania NTP paketov
NTP_RETRY_DELAY	*5000 (ms)*	Perióda retransmisie chybného NTP
DHCP_FAIL_DELAY	*3600000 (ms)*	Perióda retransmisie DHCP požiadavky
DHCP_MAINTAIN_DELAY	*10000 (ms)*	Perióda kontroly DHCP prenájmu
MAX_PACKET_SIZE	*265 (b)*	Maximálna veľkosť paketu
MAX_PACKET_DELAY	*5 (s)*	Maximálne oneskorenie paketu
COM_RETRY_DELAY	*500 (ms)*	Perióda retransmisie chybného paketu
COM_RESET_DELAY	*4200 (ms)*	Maximálny čas retransmisie
HELLO_TIMEOUT	*32 (s)*	Maximálny čas aktívneho režimu pri strate konektivity
WDT_TIME	*WDTO_2S*	Perióda Watchdog časovača

Tab. 8-2 Základné konfiguračné parametre externých modulov

Po vykonaní všetkých požadovaných zmien je potrebné preformátovať a opätovne inicializovať EEPROM pamäť, v ktorej sa nachádzajú všetky nastavania. Prepísanie EEPROM pamäti pri spustení modulu sa nastavuje pomocou *#define CONFIG_REWRITE*. Po premazaní pamäte je nutné odstrániť túto konfiguráciu, pretože inak by sa modul resetoval do základného nastavenia pri každom spustení, čo nie je z pohľadu systému a konfigurovateľnosti z používateľského rozhrania vôbec vhodné.

Pri zmene zapojenia shield-ov je nutné zmeniť použité porty v zdrojovom kóde alebo v náležiacich knižniciach. V štandardnom zapojení je potrebné prerušiť port D10 na GSM shield-e a prepojiť ho na port D8. Pri zmene konfiguračných parametrov Wi-Fi siete je nutné nanovo vykonať konfiguráciu WiFly modulov. Bližší opis konfiguračných krokov sa nachádza v nasledujúcej kapitole.

8.1.2 Konfigurácia WiFly modulu [27]

WiFly modul si dokáže uložiť konfiguračné nastavenia a po spustení sa automaticky pripojiť na nakonfigurovanú Wi-Fi sieť. Vďaka tejto funkcionalite nie je nutné pri každom spustení odznova nastavovať parametre v konfiguračnom režime. Základná konfigurácia asociácie s Wi-Fi sieťou sa vykonáva príkazmi pomocou programu *SpiUartTerminal* z knižnice.

set wlan auth 3 //(WPA1 a WPA2-PSK autentifikácia)
set wlan channel 0 //(automatické priradenie komunikačného kanála)
set wlan ext_antenna 0 //(použitie vstavanej antény)
set wlan join 1 //(automatické pripojenie na sieť so zhodným parametrami)
set wlan phrase peresini //(nastavenie prístupového hesla na Wi-Fi sieť)
set wlan rate 12 //(štandardná komunikačná rýchlosť 24 Mbit/s)
set wlan ssid DP_xperesini //(SSID pomenovanie Wi-Fi siete)

Parametre Wi-Fi siete sú už nastavené, avšak na realizáciu komunikácie je ešte nutné nakonfigurovať komunikačný protokol, automatické párovanie s UDP serverom, zakázanie zbytočných výpisov a časovače *wake* a *sleep*.. Konfigurácia sa vykonáva nasledovnými príkazmi:

set ip dchp 3 // (automatické získanie IP adresy z DHCP servera v prípade potreby)
set ip host 0.0.0.0 // (automatické spárovanie s prvým UDP serverom)
set ip localport 2051 // (zdrojový port)
set ip remote 2050 // (cieľový port)
set ip protocol 0 // (UDP protokol)
set ip flags 0x40 // (mód automatického párovania)
set sys sleep 70 // (modul sa uspí po 70 sekundách od poslednej komunikácie)
set sys wake 1 // (na zresetovanie modulu postačuje spánok s dĺžkou 1 sekunda)
set sys printlvl 0 // (zakáže zbytočné konzolové výpisy)

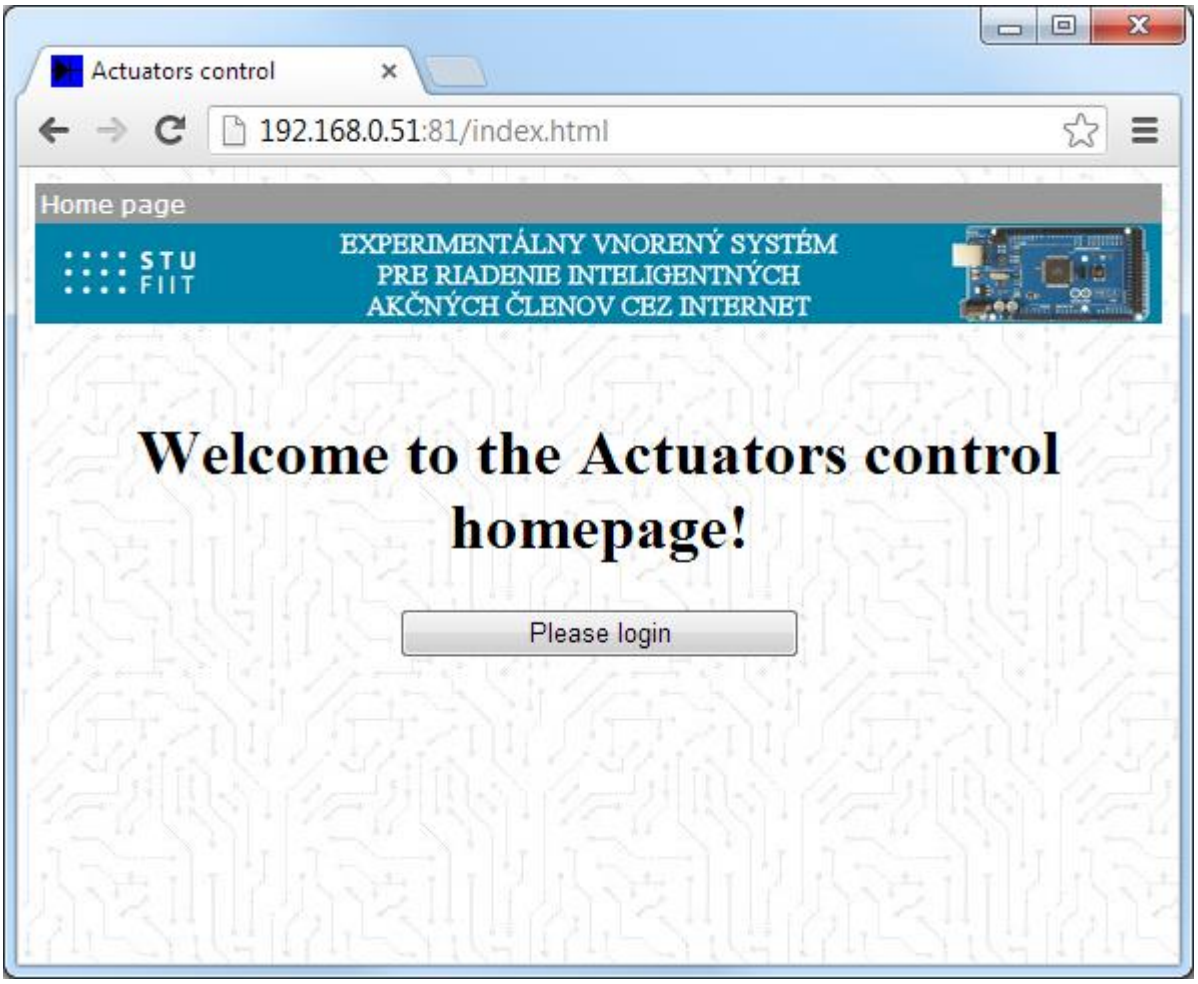
Po nakonfigurovaní požadovaných parametrov uložíme nastavenia a modul reštartujeme:

save // (uloženie nastavení)
reboot // (reštartovanie pre aplikovanie nastavení)

Po vykonaní reštartu sa WiFly automaticky pripojí na zadanú Wi-Fi sieť a modul si už môže vymieňať UDP správy s centrálnou jednotkou, ktorej IP adresa sa automaticky zdeteguje.

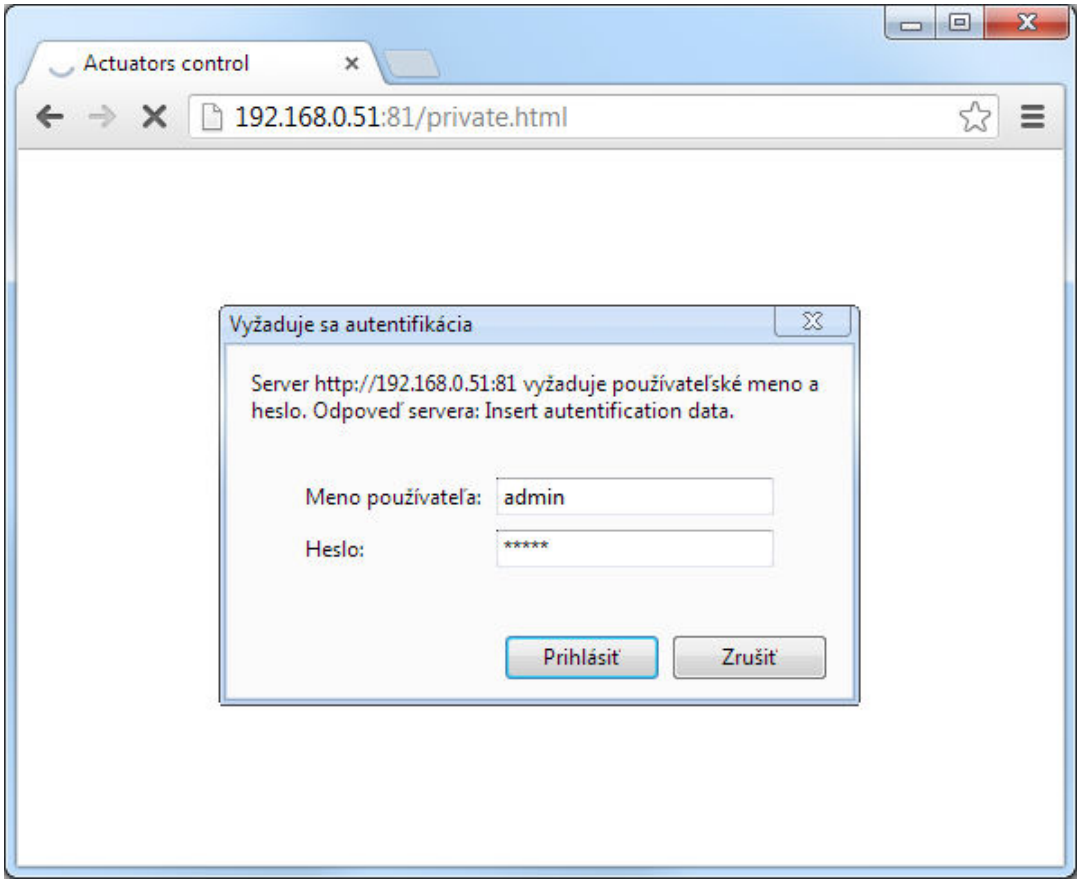
8.2 Používateľská príručka

Používateľská príručka sa sústreďuje na opis používateľského prostredia. Používateľ sa k používateľskému prostrediu pripája pomocou internetového prehliadača z ľubovoľného zariadenia. V rámci intranetu postačuje zadať IP adresu a port centrálnej jednotky, avšak v prípade prístupu z internetu cez NAT server je najskôr nutné zadefinovať presmerovanie požadovaného portu. Po správnej konfigurácií je používateľské rozhranie dostupné prakticky z ľubovoľného zariadenia a ľubovoľného miesta. Po zadaní adresy sa zobrazí úvodná obrazovka používateľského rozhrania, ktorá je znázornená aj na nasledujúcom obrázku.



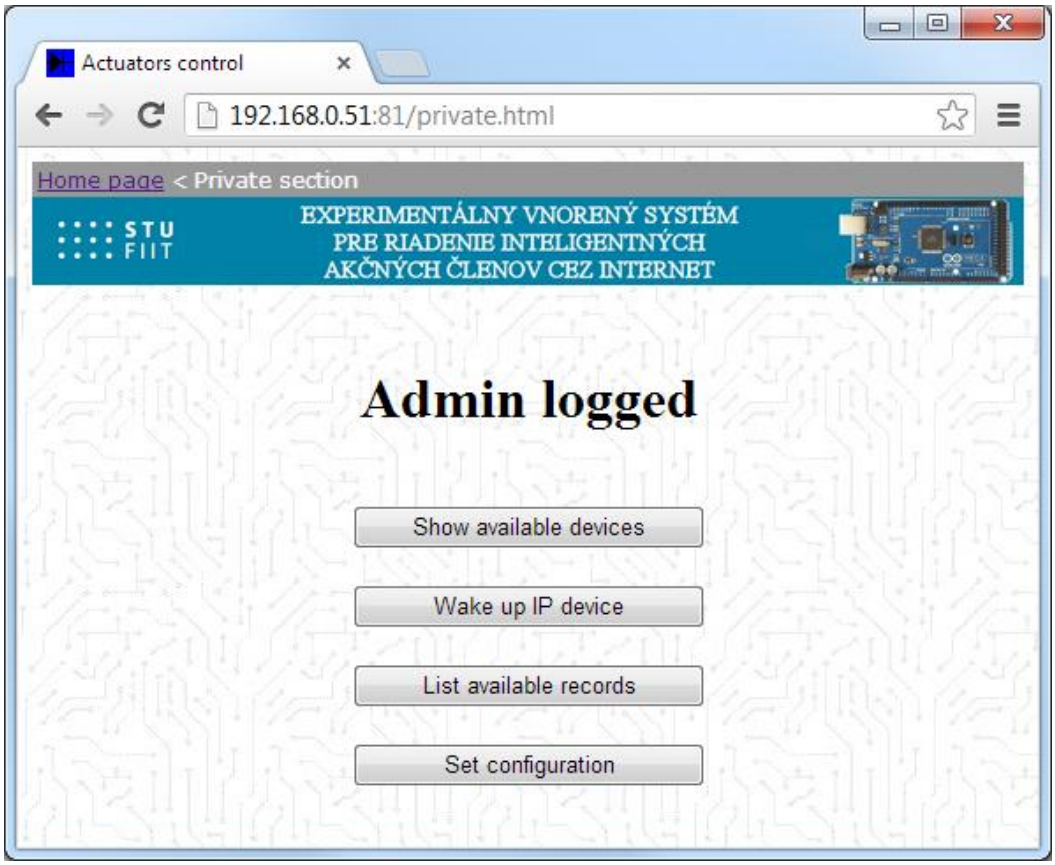
Obr. 8-2 Úvodná obrazovka používateľského rozhrania

Po kliknutí na tlačidlo *Please login* sa zobrazí výzva na zadanie prihlasovacích údajov. Štandardné prihlasovacie údaje sú *user/user* a *admin/admin*. Prihlasovacie údaje je možné pozmeniť v zdrojovom kóde. Používateľ *user* a administrátor *admin* majú rôzne prístupové práva. Používateľské konto umožňuje zobraziť len záznamy zo senzorov, ich aktuálnu hodnotu a časový priebeh. Nie je možné vykonať akúkoľvek zmenu v systéme. Konto administrátora má plné práva a môže preto vykonávať ľubovoľnú konfiguráciu.

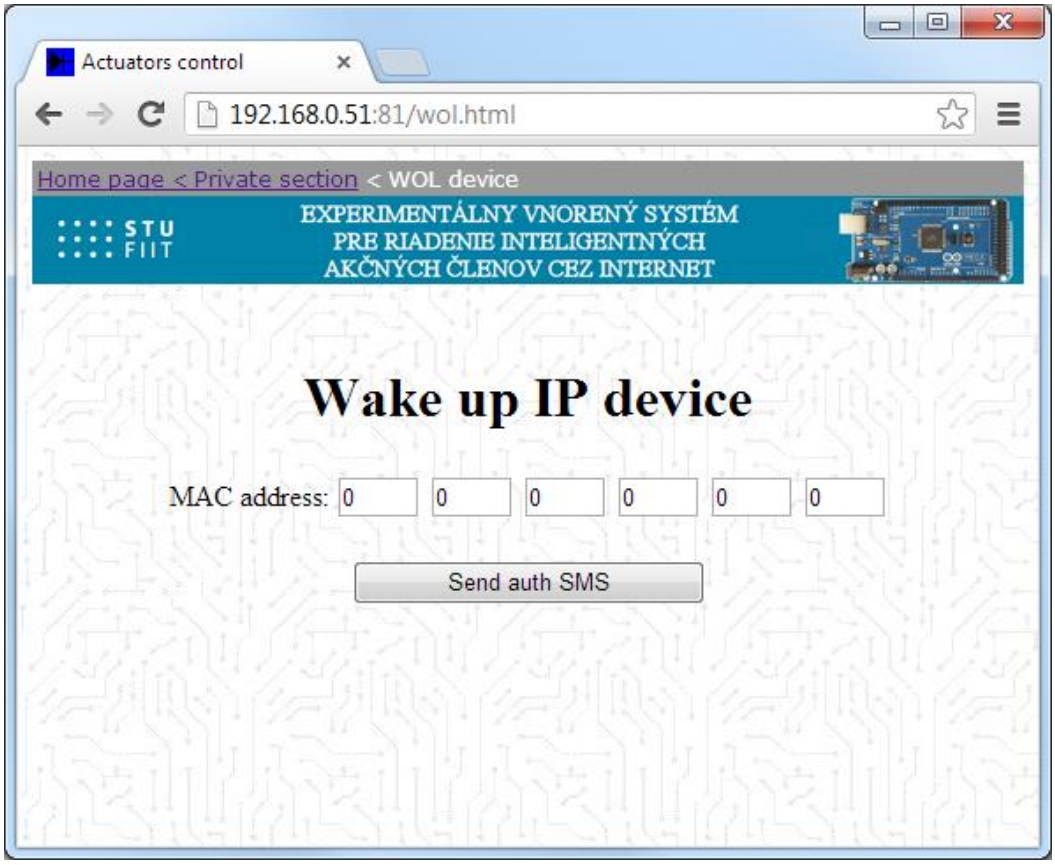


Obr. 8-3 Zadávanie prihlasovacích údajov

Po prihlásení do systému je zobrazená obrazovka s dostupnou funkcionalitou. Administrátor má dostupné možnosti zobrazenia aktuálnych zariadení, posielanie WOL paketov, zobrazenie dátových záznamov na pamäťovej SD karte a nastavenie konfigurácie. Používateľ má k dispozícii len zobrazenia aktuálnych zariadení a dátových záznamov. Po kliknutí napríklad na *Wake up IP device* je zobrazená obrazovka s možnosťou posielania WOL paketov. Po zadaní požadovanej MAC adresy, zaslaní a zapísaní autentifikačného kódu z SMS správy je poslaný WOL Magic packet, ktorý prebudí zvolené zariadenie. Obidve popísané obrazovky sú znázornené na nasledujúcich obrázkoch. V celom používateľskom rozhraní je v hornej časti zobrazovaná navigačná lišta, ktorá identifikuje aktuálnu obrazovku a jej hierarchiu v rámci rozhrania. K predchádzajúcej obrazovke sa dá dostať kliknutím na jej názov.

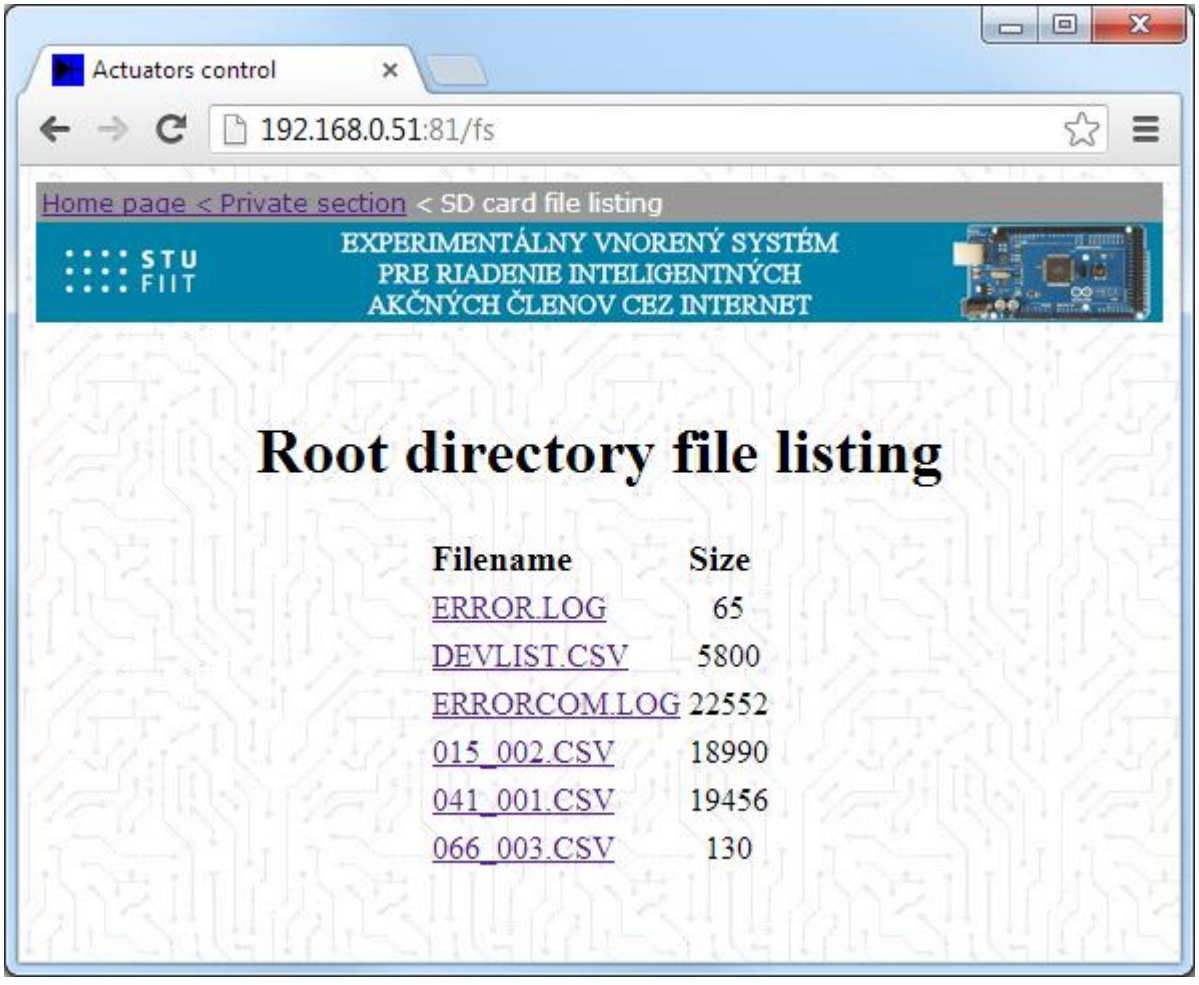


Obr. 8-4 Obrazovka po prihlásení administrátora



Obr. 8-5 Obrazovka posielania WOL paketov

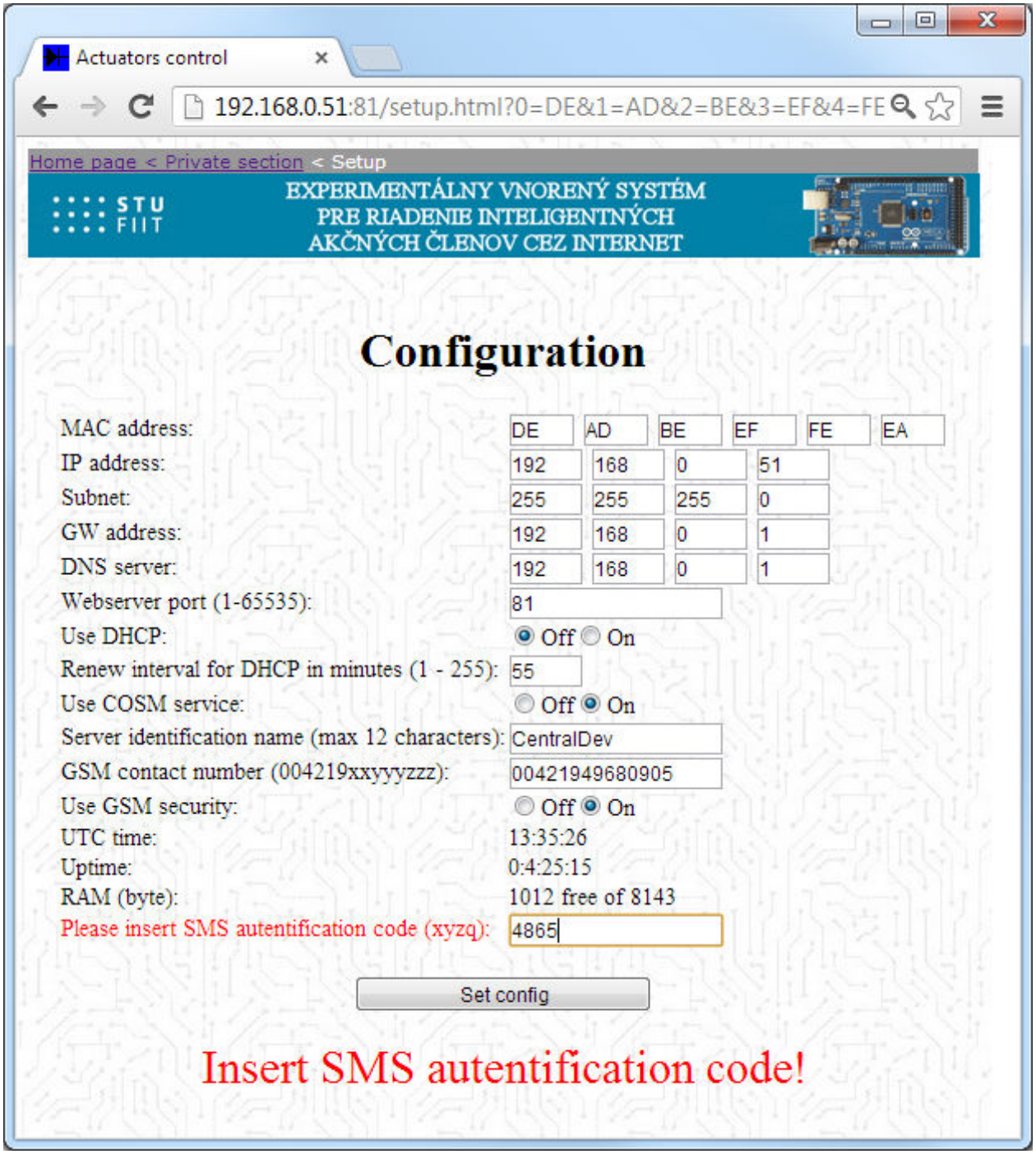
Ďalšou dostupnou funkciou je zobrazenie dátových záznamov, ktoré sú uložené na pamäťovej SD karte. K zoznamu súborov je možné pristúpiť pomocou URL adresy s ukončením */fs*, alebo aktivovaním tlačidla *List available records.* Zoznam súborov obsahuje okrem názvov súborov aj ich veľkosť. Po kliknutí na názov je možné súbor stiahnuť alebo uložiť. Medzi základné súbory patria záznamy zo zariadení, ktoré je možné identifikovať názvom, ktorý začína identifikačným číslom a typom modulu. Súbor typu CSV je možné jednoducho použiť na jednoduché extrahovanie záznamov alebo tvorbu špecifických grafov. Ďalšími súbormi sú *ERROR.LOG* a *ERRORCOM.LOG*, ktoré obsahujú systémové a komunikačné chybové výpisy. Samostatný súbor *DEVLIST.CSV* obsahuje konfiguračné údaje externých modulov. Súbor je spracúvaný len lokálne a preto jeho porušenie alebo zmazanie neovplyvní správanie systému. Obrazovka s výpisom súborov je znázornená na nasledujúcom obrázku.



Obr. 8-6 Obrazovka výpisu dostupných súborov

Konfiguračná obrazovka je prístupná cez tlačidlo *Set configuration* a umožňuje zmenu sieťových parametrov ako MAC a IP adresu, masku siete, adresu brány a DNS servera, port používateľského rozhrania, použitie a periódu obnovy DHCP, použitie COSM služby, názov

centrálnej jednotky, telefónne číslo a použitie GSM autentifikácie. Konfiguračná obrazovka navyše zobrazuje systémový čas, dobu činnosti a voľnú RAM pamäť. Po vykonaní zmien a stlačenia tlačidla *Send auth SMS* je na uložené telefónne číslo poslaná SMS správa s autentifikačným kódom, ktorý je potrebné zadať do políčka *Please insert autentification code*. Po úspešnom overení kódu sú požadované zmeny uložené a centrálna jednotka reštartovaná. Konfiguračná obrazovka je znázornená na nasledujúcom obrázku.

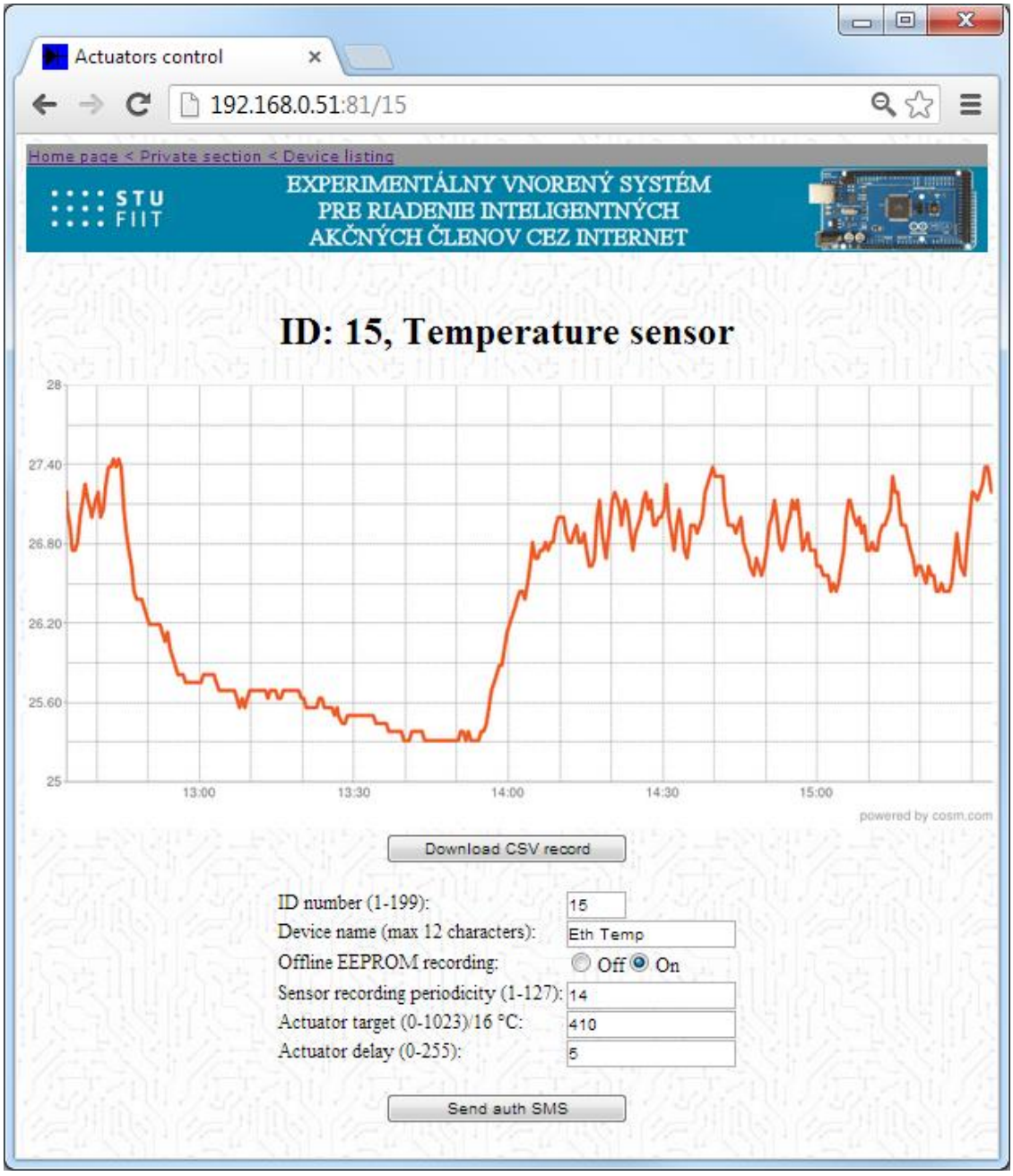


Obr. 8-7 Konfiguračná obrazovka

Azda najdôležitejšou obrazovkou používateľského rozhrania je zobrazenie grafických priebehov meraných veličín. K obrazovke sa pristupuje pomocou tlačidla *Show available devices*. Obrazovka zobrazuje len aktívne externé moduly s ktorými prebieha dátová komunikácia. Informácie o každom module obsahujú jeho identifikačné číslo, názov a grafický priebeh za posledných 24 hodín. Detailný výpis o zvolenom module sa zobrazí po kliknutí na graf. Obrazovka s dostupnými modulmi je znázornená na nasledujúcom obrázku.

Obr. 8-8 Obrazovka s dostupnými externými modulmi

Obrazovka s detailným zobrazením informácií o zvolenom externom module obsahuje podrobnejší grafický priebeh hodnôt, ktorý je skrátený na obdobie 3 hodín, identifikačné číslo modulu a jeho názov, periodicitu záznamu, cieľovú hodnotu akčného člena a jeho oneskorenie. Nastavenie modulu umožňuje zakázať zaznamenávanie nameraných hodnôt v lokálnom režime a šetriť tak EEPROM pamäť. Po vykonaní požadovaných zmien je taktiež nutné zadať autentifikačný kód z prijatej SMS správy. V prípade prihlásenia s právami používateľa nie je možné meniť tieto hodnoty. Obrazovka s detailným zobrazením modulu je na nasledujúcom obrázku.



Obr. 8-9 Obrazovka s detailným zobrazením informácií o externom module

Zoznam použitej literatúry

[1] FIDIS: A Structured Collection on Information and Literature on Technological and Usability Aspects of Radio Frequency Identification (RFID). http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp3-del3.7.literature_RFID.pdf (2012-04-09)

[2] JOMBÍK, P.: Vnorený systém pre vzdialené monitorovanie a riadenie domácnosti : Diplomová práca. FIIT STU, Bratislava, 2011

[3] KIŠÁK, S.: Experimentálny vnorený systém pre vzdialené monitorovanie a riadenie : Diplomová práca. FIIT STU, Bratislava, 2011

[4] SMARTHOME: Sensaphone Web600. http://www.smarthome.com/7008/Sensaphone-Web600-Web-Based-Monitoring-and-Alarm-Notification-System/p.aspx (2012-04-12)

[5] SENSAPHONE: Sensaphone Web600 Web-Based Monitoring System. http://www.sensaphone.com/sensaphone_web600.php (2012-05-05)

[6] SMARTHOME: LockState Connect LS-90i WiFi. http://www.smarthome.com/53279/LockState-Connect-LS-90i-Wi-Fi-Touch-Screen-Thermostat-with-3-Stages-of-Heating/p.aspx (2012-04-12)

[7] LOCKSTATE: LS-90i Internet Thermostat. http://www.lockstateconnect.com/ProductDetails.asp?ProductCode=LS-90i (2012-05-05)

[8] SMARTHOME: ecobee EB-EMS-02 Energy Management System. http://www.smarthome.com/29921/ecobee-EB-EMS-02-Energy-Management-System-with-Touchscreen-Internet-Thermostat/p.aspx (2012-04-12)

[9] ECOBEE: The ecobee Smart Thermostat. http://www.ecobee.com/solutions/home/smart/ (2012-05-05)

[10] PEREŠÍNI, O.: Rekonfigurovateľná architektúra pre hardvérové šifrovanie údajov : Bakalárska práca. FIIT STU, Bratislava 2011

[11] SPARKFUN.: Electronics catalog (ATMega328 – TQFP). http://www.sparkfun.com/products/9261 (2012-05-04)

[12] SEEKIC.: ATMEGA2560-16AU Datasheet. http://datasheet.seekic.com/datasheet/Atmel_ATMEGA2560-16AU.html (2012-05-05)

[13] ATMEL: ATmega 328 DATASHEET. http://www.atmel.com/Images/doc8161.pdf (2012-06-05)

[14] JASONVOLK: ATMega328. http://jasonvolk.com/electronics/avr-electronics/2010/atmega328/ (2012-05-05)

[15] ATMEL: ATmega 2560 DATASHEET. http://www.atmel.com/Images/doc2549.pdf (2012-06-05)

[16] SIPHEC: ATm2560-UT-TB. http://www.siphec.com/item/ATm2560-UT-TB.html (2012-05-06)

[17] ARDUINO: Web stránka projektu. http://arduino.cc/en/ (2012-05-04)

[18] ARDUINO: Software download. http://arduino.cc/en/Main/Software (2012-05-06)

[19] ARDUINO: Arduino UNO. http://arduino.cc/en/Main/ArduinoBoardUno (2012-05-05)

[20] ARDUINO: Arduino Mega 2560. http://arduino.cc/en/Main/ArduinoBoardMega2560 (2012-05-05)

[21] WIZNET: W5100 DATASHEET. http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5100_Datasheet_v1.2.2.pdf (2012-05-06)

[22] ARDUINO: Ethernet Shield. http://arduino.cc/en/Main/ArduinoEthernetShield (2012-05-06)

[23] WIZNET: W5100 Block diagram. http://www.wiznet.co.kr/Sub_Modules/en/product/Product_Detail.asp?cate1=5&cate2=7&cate3=26&pid=1011 (2012-05-06)

[24] ELEKTROMODELY.COM: Arduino Ethernet PoE modul. http://elektromodely.com/modely/product_info.php?products_id=6405 (2012-05-06)

[25] KOTOČOVÁ, M.: Počítačové a komunikačné siete II: Prednášky. FIIT STU, Bratislava 2010

[26] SPARKFUN: Wifly Shield. https://www.sparkfun.com/products/9367 (2012-05-07)

[27] ROVING NETWORKS: WIFLY GSX DATASHEET. http://www.sparkfun.com/datasheets/Wireless/WiFi/WiFlyGSX-um2.pdf (2012-05-07)

[28] GITHUB: Arduino Xtea library https://github.com/franksmicro/Arduino/tree/master/libraries/Xtea (2012-05-08)

[29] IETF: The Base16, Base32, and Base64 Data Encodings RFC. https://tools.ietf.org/html/rfc4648 (2012-05-08)

[30] SIMCOM: GSM/GPRS Module – SIM900. http://wm.sim.com/producten.aspx?id=1019 (2013-04-08)

[31] CODE.GOOGLE: GSM shield arduino library. http://code.google.com/p/gsm-shield-arduino/ (2013-04-12)

[32] ITEADSTUDIO: ICOMSAT GSM shield. http://imall.iteadstudio.com/im120417009.html (2013-04-08)

[33] SOS: Fotorezistor PERKIN ELMER FW300. http://www.sos.sk/?str=371&artnum=28919&name=perkin-elmer-fw300-95503513 (2012-05-08)

[34] MAXIM: Digital Thermometer DS18B20 DATASHEET. http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf (2012-05-08)

[35] SOS: RFID čítačka ID12. http://www.sos.sk/?str=371&artnum=54214&name=id-innovations-id12 (2012-05-08)

[36] SPARKFUN: ID series DATASHEET. http://www.sparkfun.com/datasheets/Sensors/ID-12-Datasheet.pdf (2012-05-08)

[37] TEXAS INSTRUMENTS: ULN2803A DATASHEET. http://www.ti.com/lit/ds/symlink/uln2803a.pdf (2012-05-08)

[38] COSM: Main page. https://cosm.com/ (2012-11-08)

[39] GITHUB: Cosm Library for Arduino. https://github.com/cosm/cosm-arduino (2012-11-08)

[40] GITHUB: WiFly-Shield library. https://github.com/sparkfun/WiFly-Shield (2012-11-10)

[41] ARDUINO: Available Memory. http://playground.arduino.cc/Code/AvailableMemory (2012-11-12)

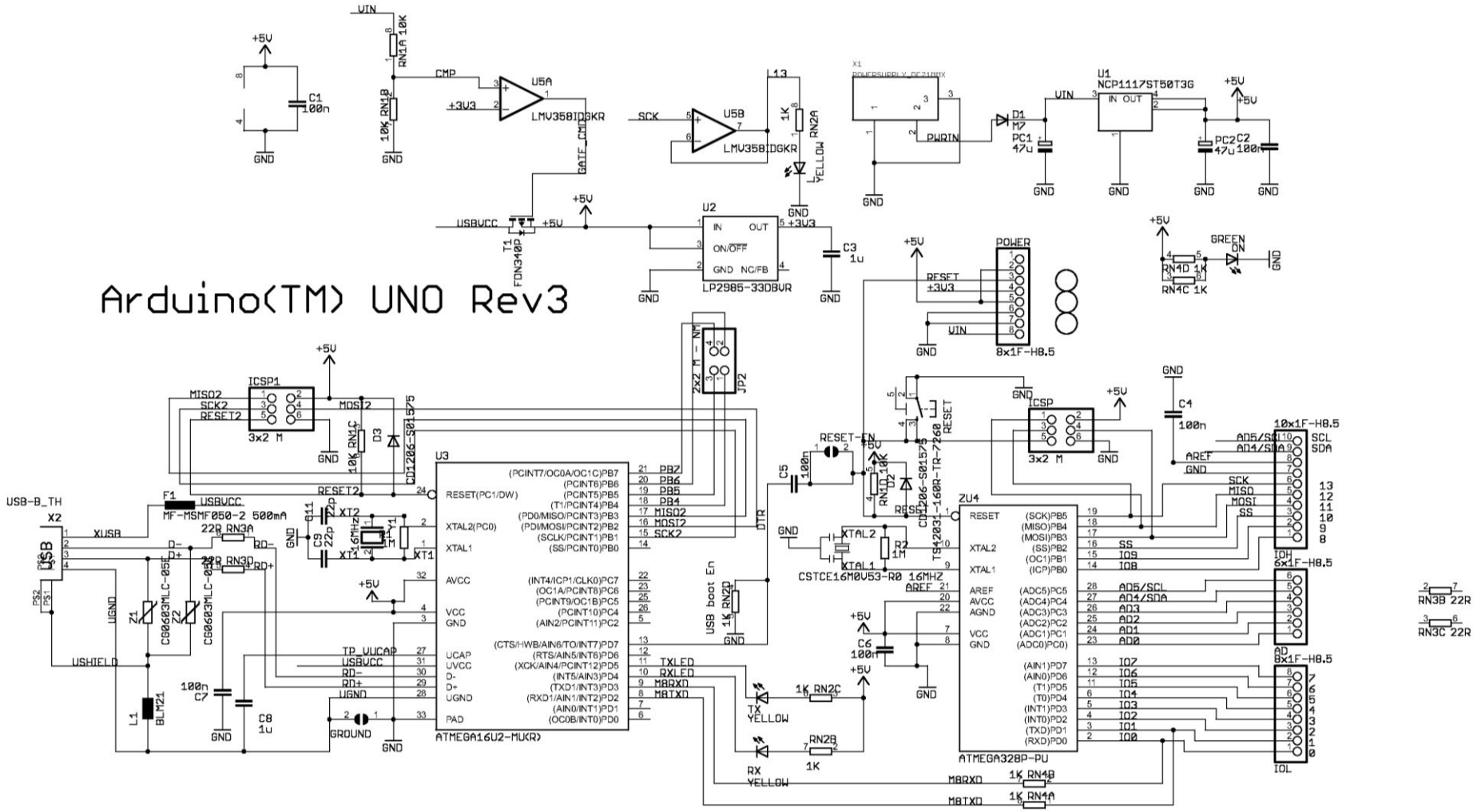
[42] ARDUINO: Dallas Semiconductor's 1-Wire Protocol. http://playground.arduino.cc/Learning/OneWire (2012-11-16)

[43] GITHUB: Webduino library. https://github.com/sirleech/Webduino (2012-11-10)

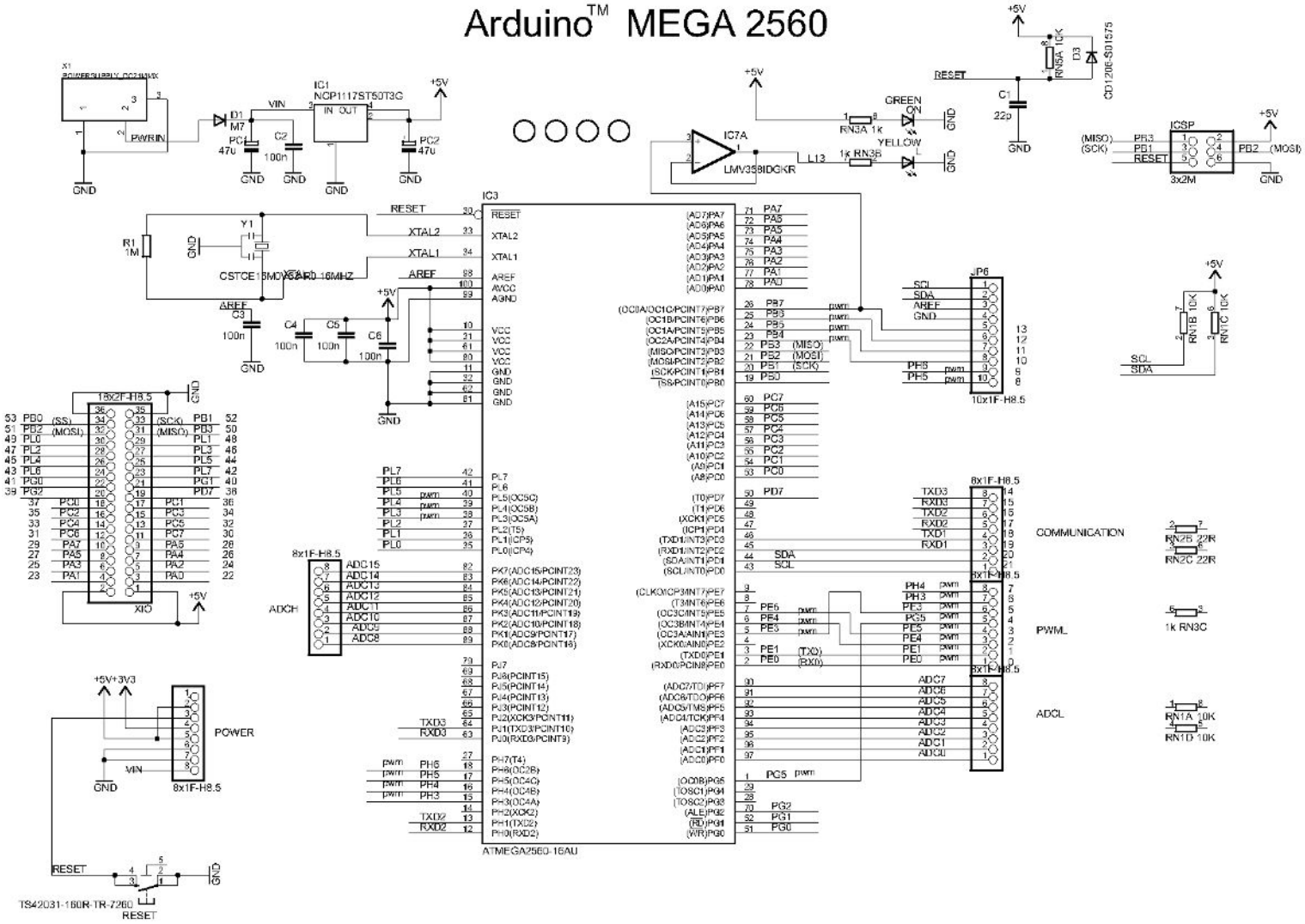
[44] ARDUINO: Code for the Innovations ID-12 RFID tag reader. http://playground.arduino.cc/Code/ID12 (2012-11-10)

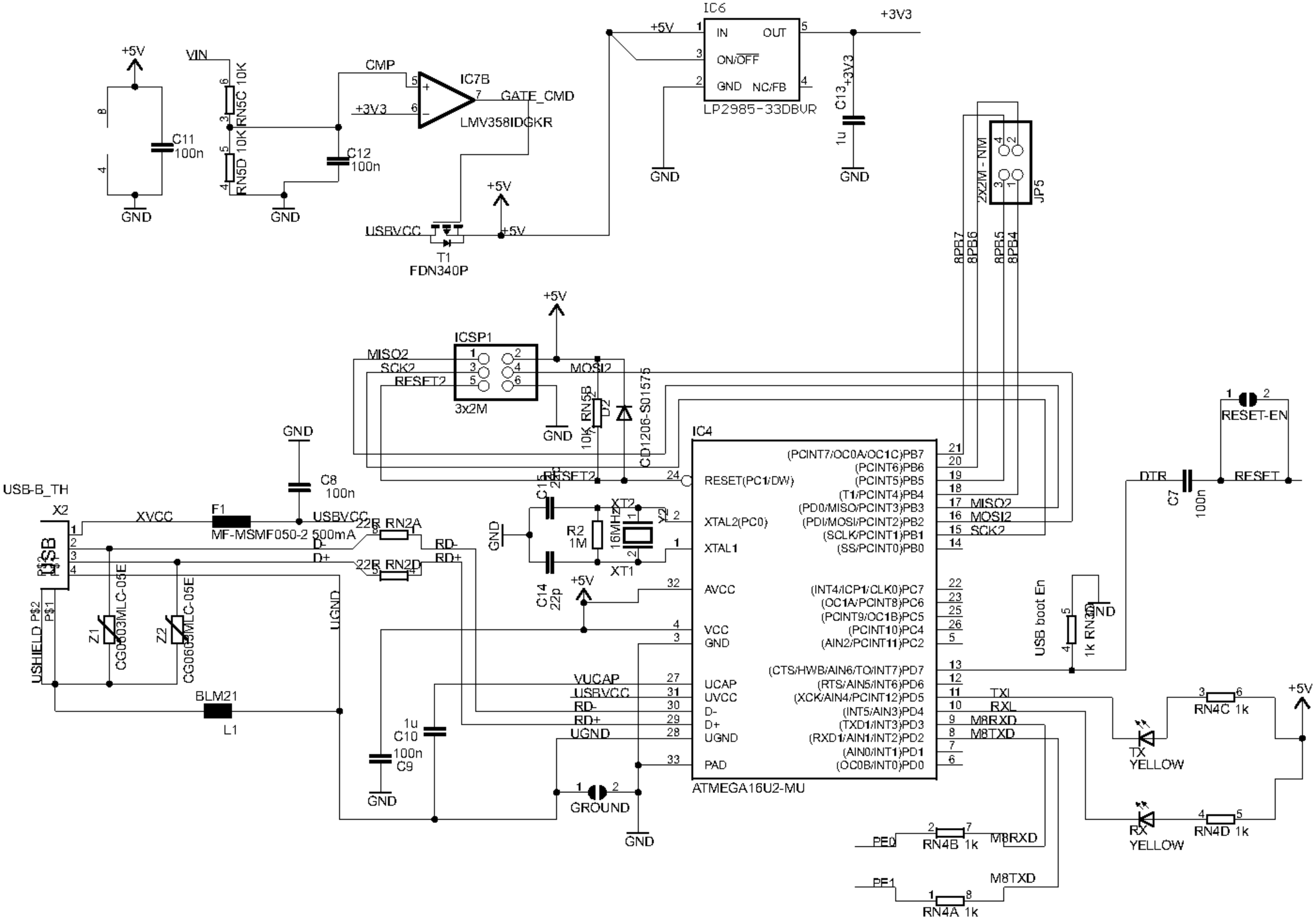
[45] GITHUB: Open-source project Arduino. https://github.com/arduino/Arduino (2012-11-16)

Príloha A – Schéma Arduino Uno R3 [19]



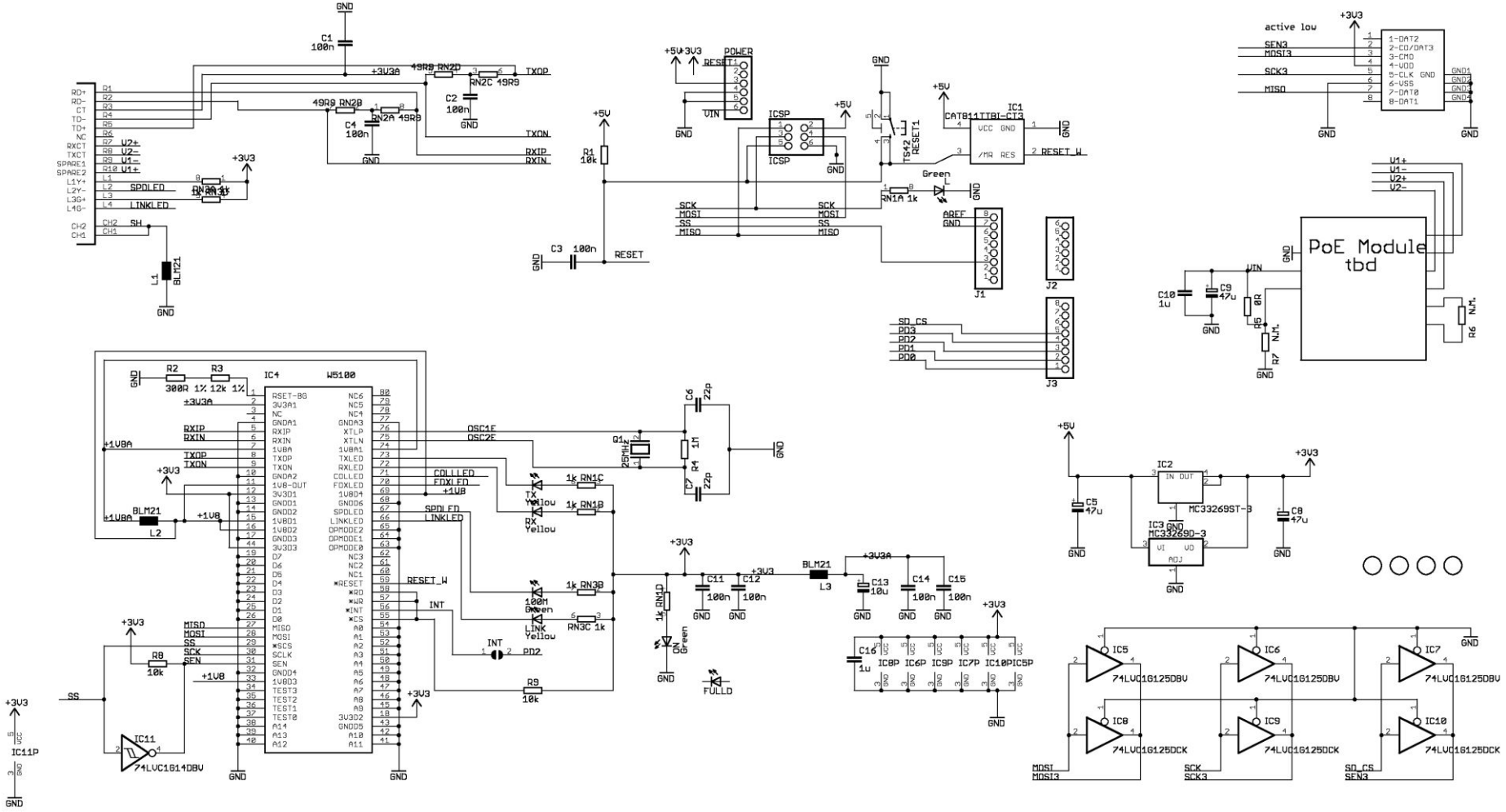
Príloha B – Schéma Arduino Mega 2560 R3 [20]



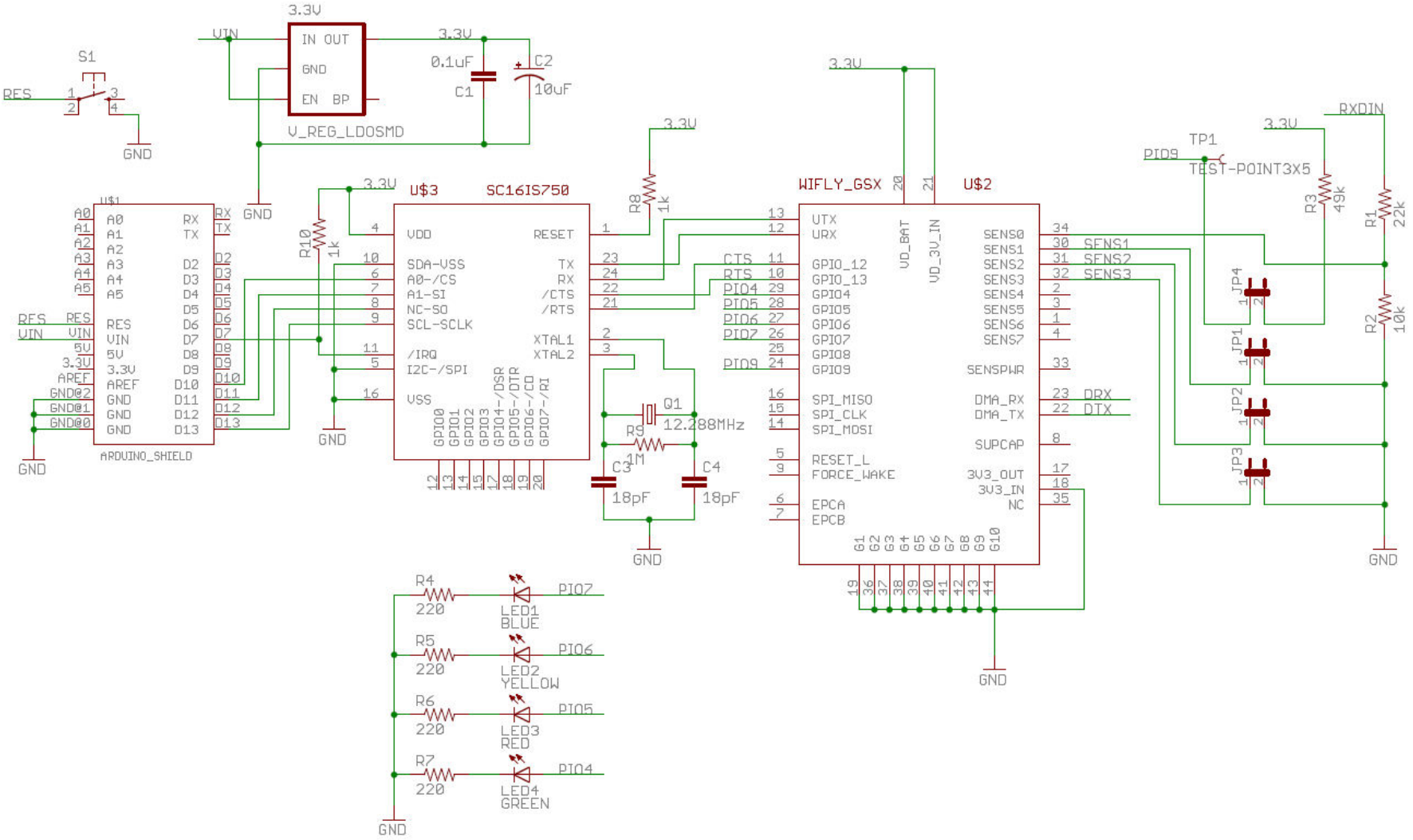


+5V
VIN
RN5C 10K
RN5D 10K
C11
100n
GND
CMP
IC7B
GATE_CMD
+3V3
LMV358IDGKR
C12
100n
USBVCC
T1
FDN340P
IC6
IN
OUT
ON/OFF
GND
NC/FB
LP2985-33DBVR
C13
1u
2X2M - NM
JP5
8PB7
8PB6
8PB5
8PB4
ICSP1
MISO2
SCK2
RESET2
MOSI2
3x2M
RN5B
10K
D2
CD1206-S01575
IC4
(PCINT7/OC0A/OC1C)PB7
(PCINT6)PB6
(PCINT5)PB5
(T1/PCINT4)PB4
(PD0/MISO/PCINT3)PB3
(PDI/MOSI/PCINT2)PB2
(SCLK/PCINT1)PB1
(SS/PCINT0)PB0
RESET(PC1/DW)
XTAL2(PC0)
XTAL1
AVCC
VCC
GND
UCAP
UVCC
D-
D+
UGND
PAD
(INT4/ICP1/CLK0)PC7
(OC1A/PCINT8)PC6
(PCINT9/OC1B)PC5
(PCINT10)PC4
(AIN2/PCINT11)PC2
(CTS/HWB/AIN6/TO/INT7)PD7
(RTS/AIN5/INT6)PD6
(XCK/AIN4/PCINT12)PD5
(INT5/AIN3)PD4
(TXD1/INT3)PD3
(RXD1/AIN1/INT2)PD2
(AIN0/INT1)PD1
(OC0B/INT0)PD0
ATMEGA16U2-MU
USB-B_TH
X2
XVCC
F1
MF-MSMF050-2 500mA
C8
100n
RN2A
RN2D
22R
RD-
RD+
C15
22p
C14
22p
R2
1M
XT2
16MHz
XT1
Y2
Z1
Z2
CG0603MLC-05E
USHIELD
BLM21
L1
UGND
C10
1u
C9
100n
VUCAP
GROUND
RESET-EN
DTR
C7
100n
RESET
USB boot En
RN4C 1k
RN4D 1k
RN4B 1k
RN4A 1k
TXL
RXL
M8RXD
M8TXD
TX
RX
YELLOW
PE0
PE1

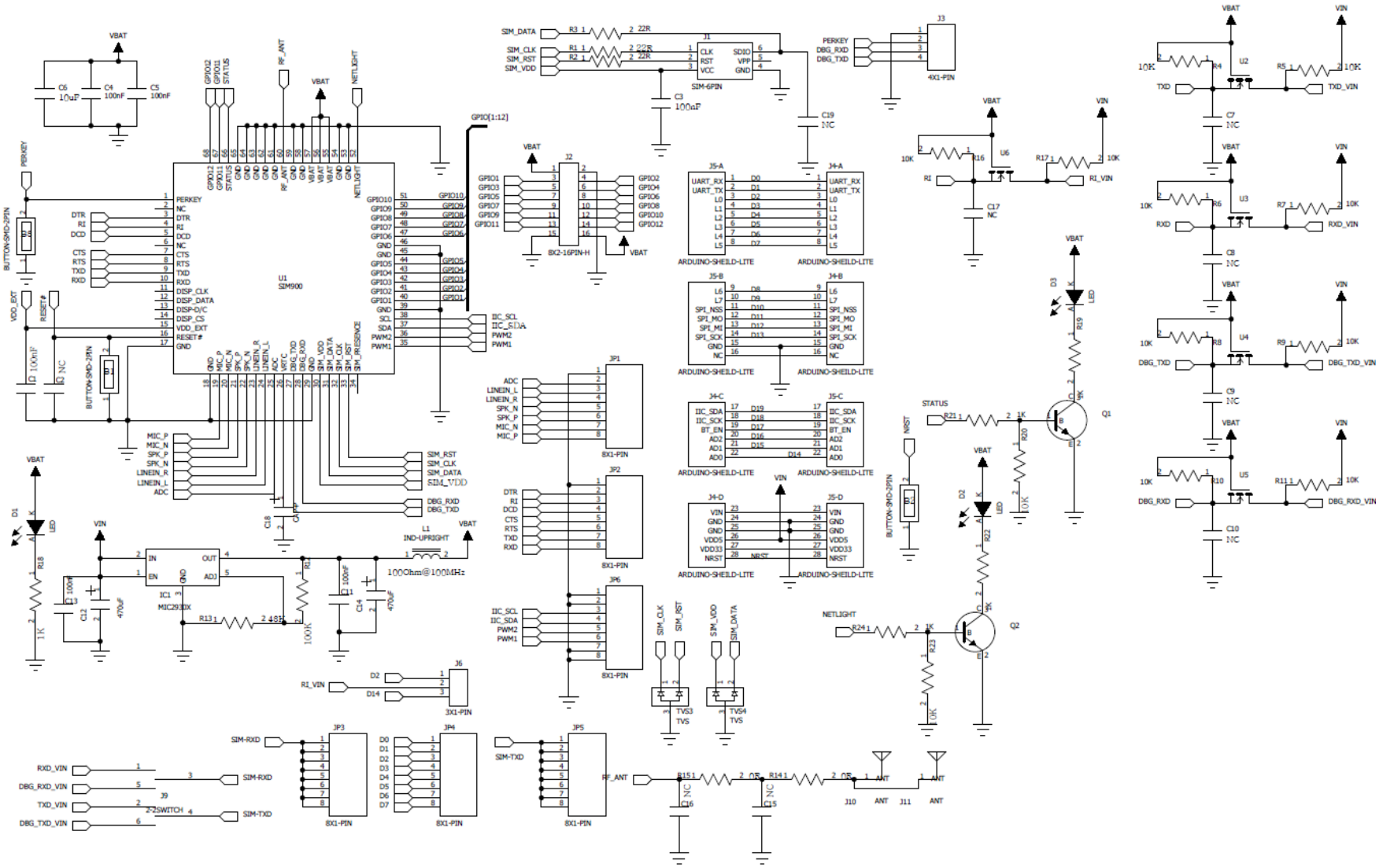
Príloha C – Schéma Arduino Ethernet shield [22]

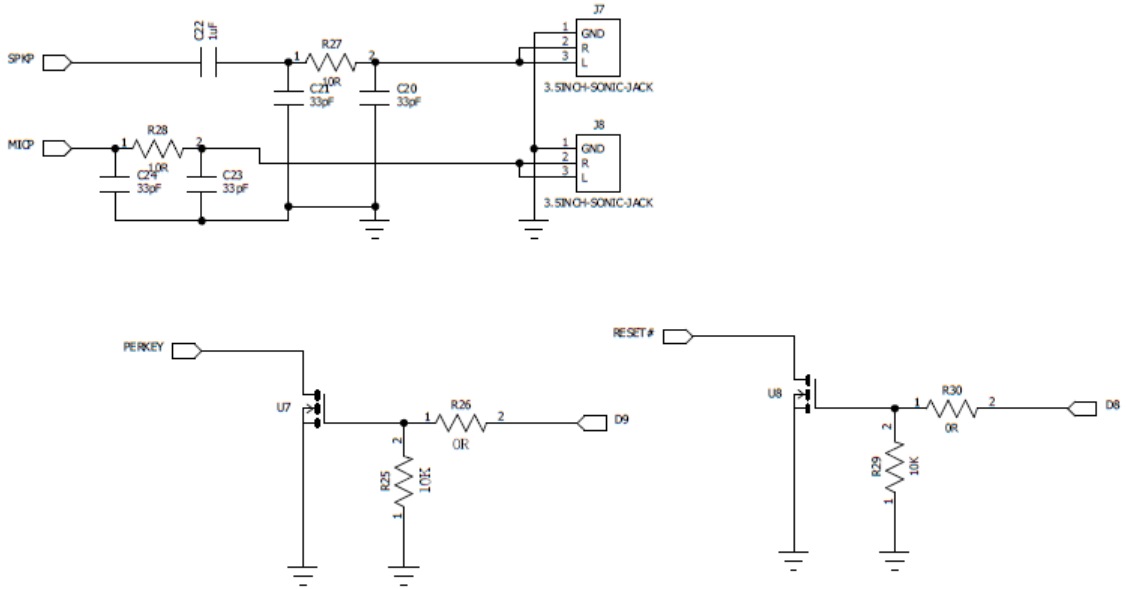


Príloha D – Schéma Arduino WiFly shield [26]

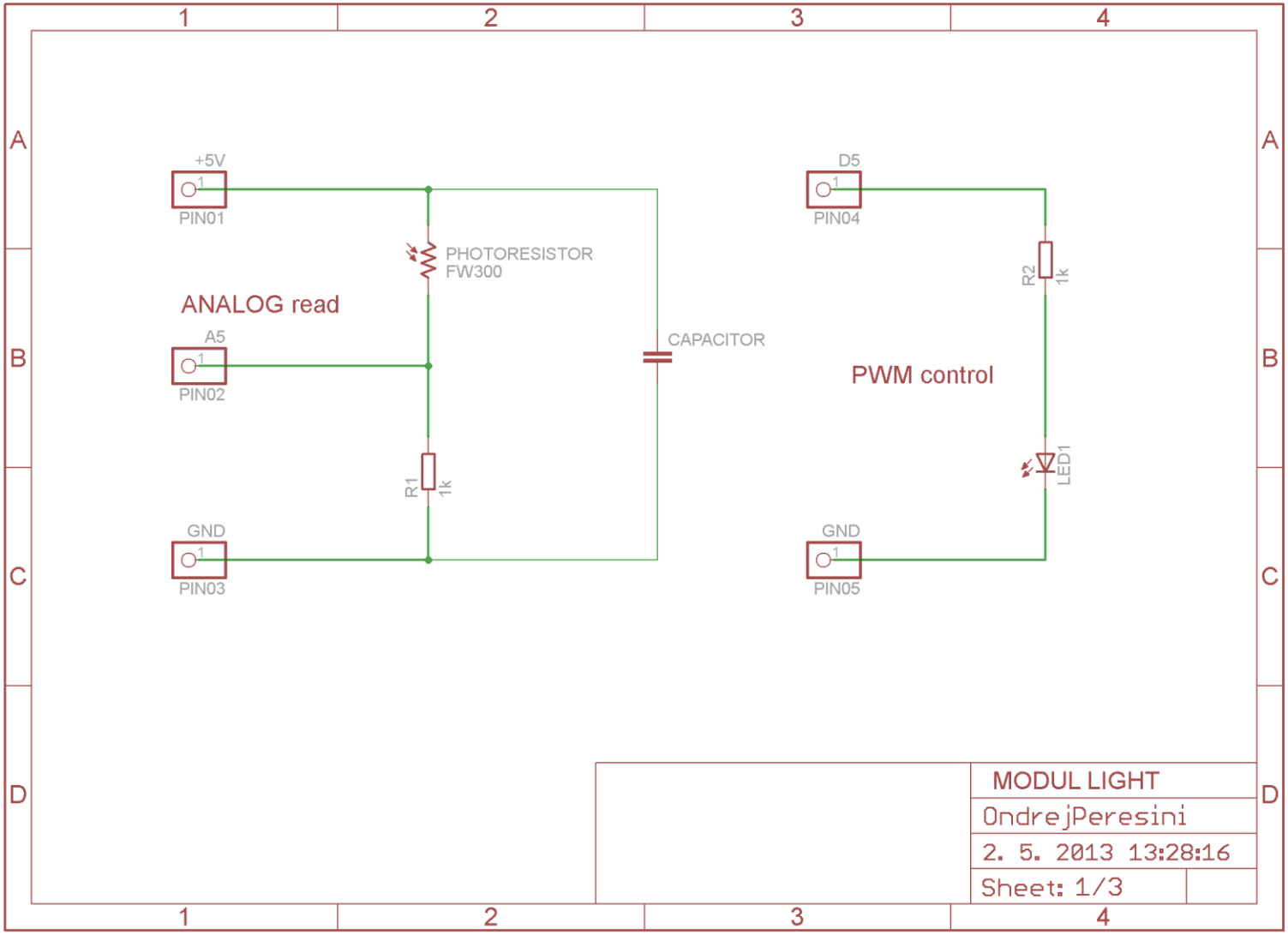


Príloha E – Schéma GSM/GPRS IComsat shield [32]

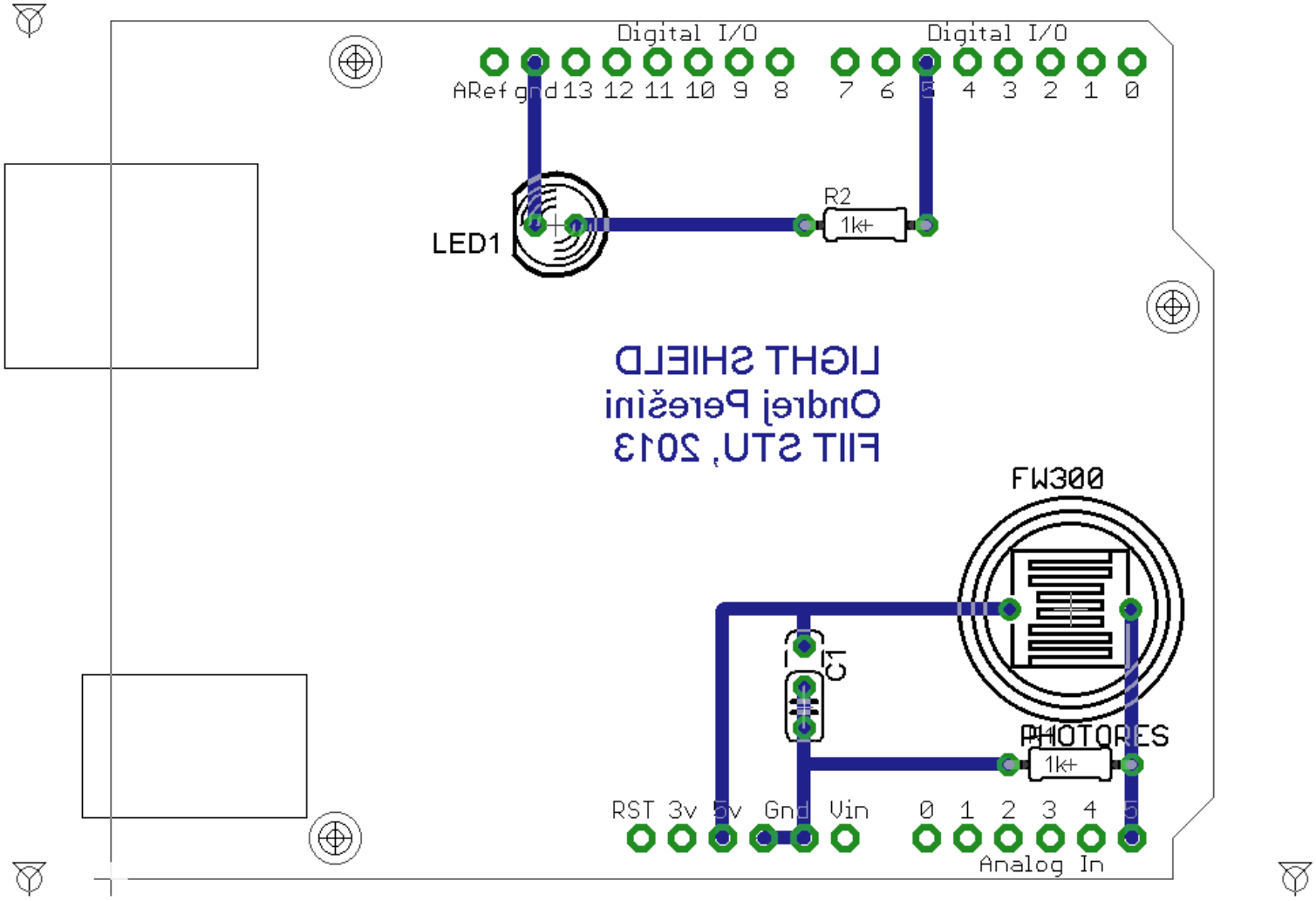



SPKP
C22
1uF
R27
10R
C21
33pF
C20
33pF
J7
GND
R
L
3.5INCH-SONIC-JACK
MICP
R28
10R
C24
33pF
C23
33pF
J8
GND
R
L
3.5INCH-SONIC-JACK
PERKEY
U7
R26
0R
D9
R25
10K
RESET#
U8
R30
0R
D8
R29
10K

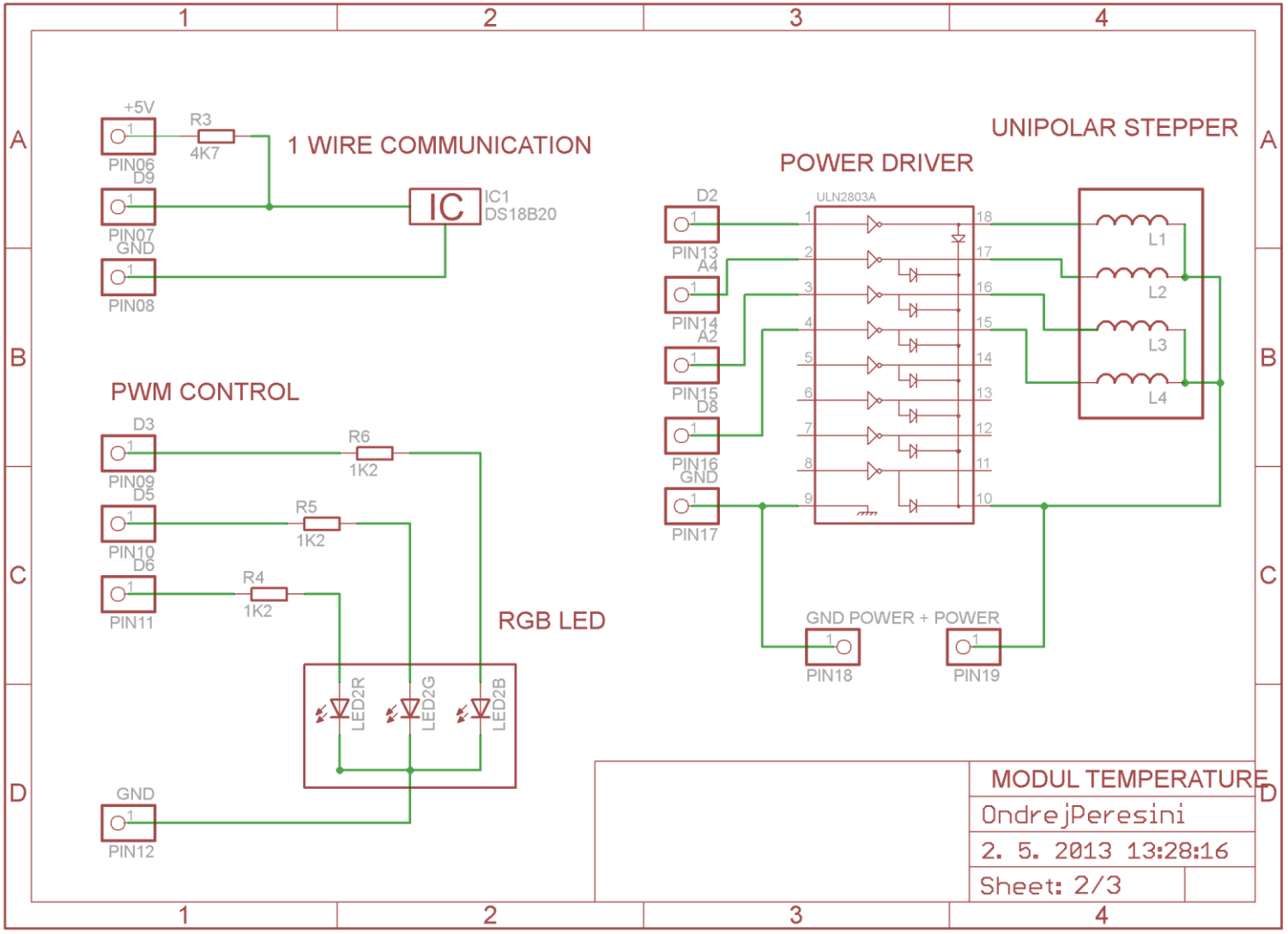
Príloha F – Schéma zapojenia: Svetelný modul



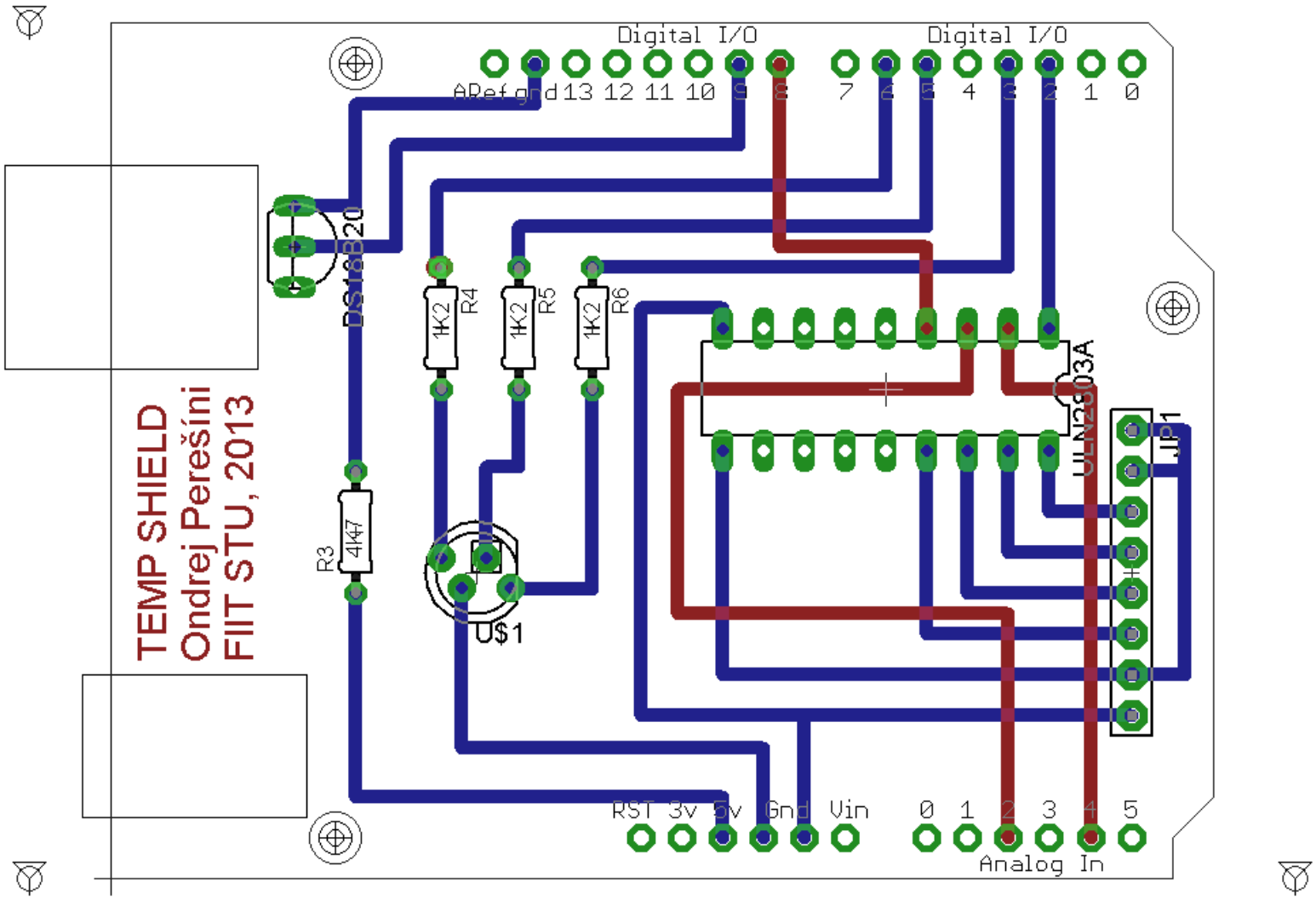
Príloha G – Schéma plošného spoja: Svetelný modul



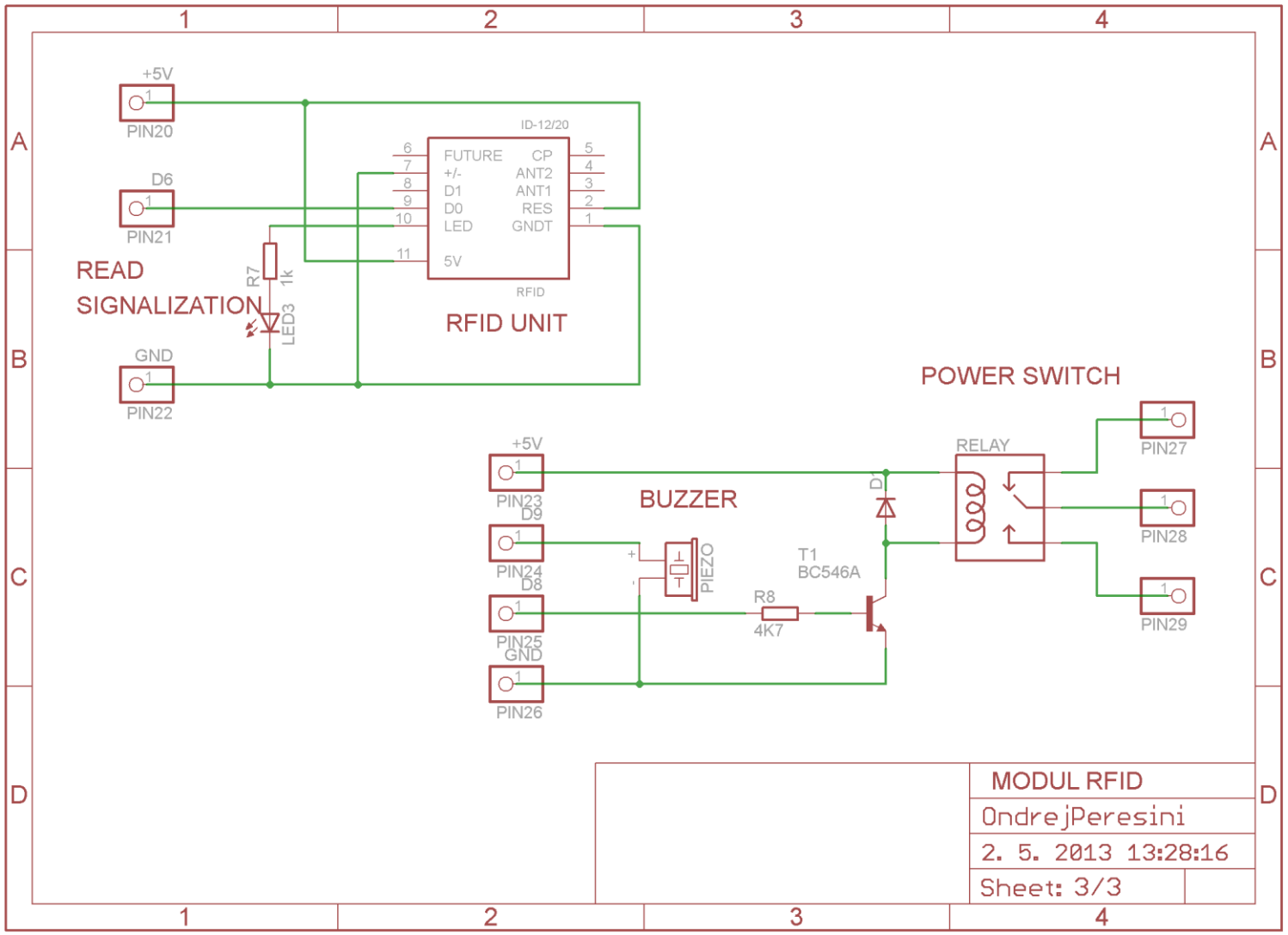
Príloha H – Schéma zapojenia: Teplotný modul



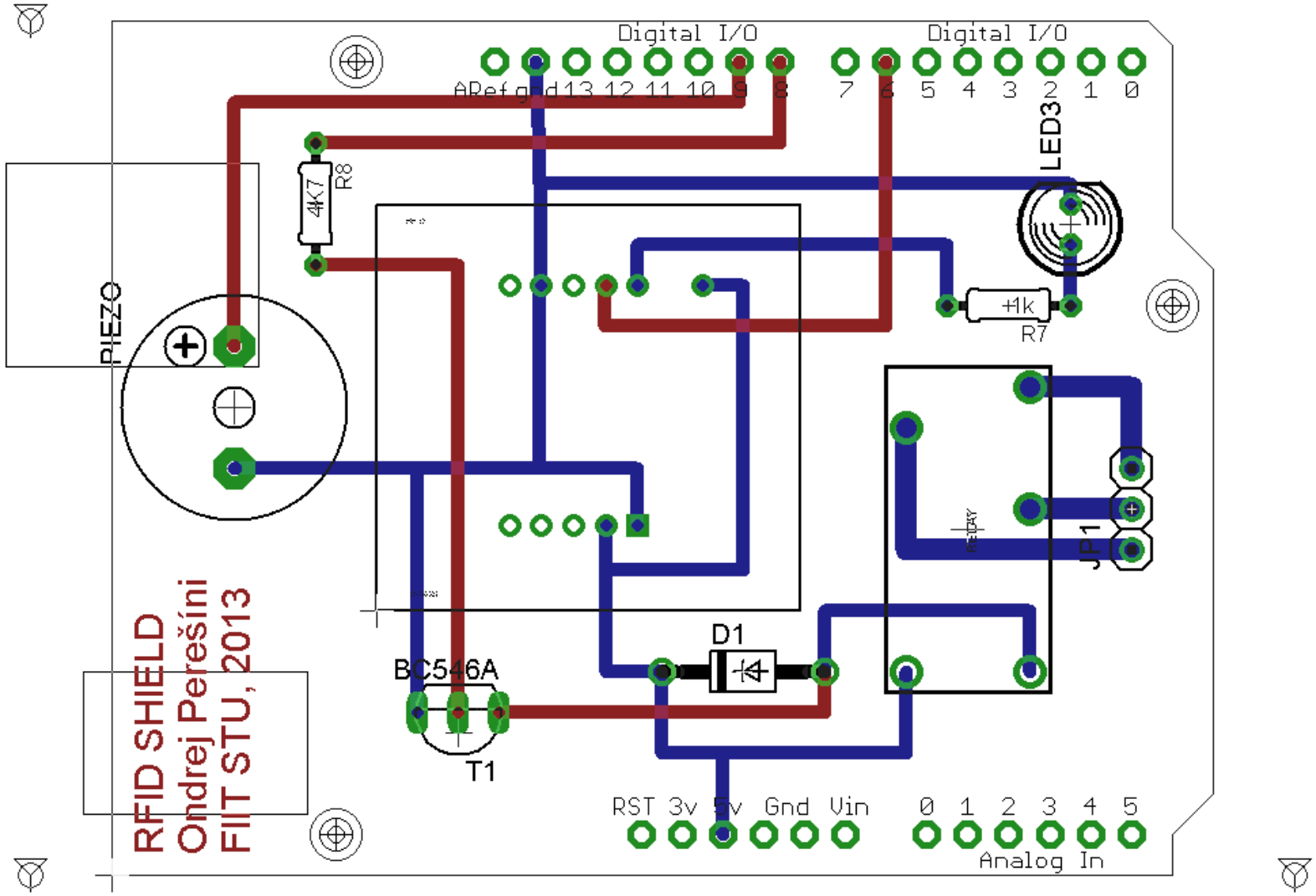
Príloha I – Schéma plošného spoja: Teplotný modul



Príloha J – Schéma zapojenia: RFID modul



Príloha K – Schéma plošného spoja: RFID modul



Príloha L – Obsah priloženého média

Súčasťou diplomovej práce je aj priložené CD médium, ktoré obsahuje všetky súbory a využívané aplikácie súvisiace s touto prácou. Súbory sú pre zlepšenie prehľadnosti rozdelené do adresárov s nasledujúcou hierarchiou:

- *Aplikacie* – použité aplikácie súvisiace s prácou.
    - *DHCP* – aplikácia vykonávajúca funkciu DHCP servera.
    - *Eagle* – aplikácia na tvorbu schém zapojenia a dosiek plošných spojov.
    - *Packet Builder* - aplikácia na faktorizáciu a posielanie paketov pre testovacie účely.
    - *Putty* – terminálový klient na sledovanie sériového výstupu z vývojovej platformy.
- *Dokument* – tento dokument a anotácie v elektronickej forme.
- *Kniznice* – modifikované zdrojové kódy všetkých použitých softvérových knižníc.
- *Literatura* – súbory s použitou literatúrou a kópie web stánok.
    - *Schemy shieldov* – schémy zapojenia použitých shield-ov.
- *Schemy* – schémy zapojenia a dosiek plošných spojov externých modulov z programu Eagle.
- *Vyvojove prostredie* – inštalačný súbor použitého vývojového prostredia.
- *Zdrojove kody* – zdrojové kódy implementovaného systému.
    - *Centralna jednotka* – súbory so zdrojovými kódmi centrálnej jednotky.
    - *Externe moduly* – zdrojový kód externých modulov.