**Technická univerzita v Košiciach**
**Fakulta elektrotechniky a informatiky**

# SNMP podpora pre 8-bit procesor ATMEL

**Diplomová práca**

**2013** **Bc. Radovan Kovalčik**

**Technická univerzita v Košiciach**
**Fakulta elektrotechniky a informatiky**

# SNMP podpora pre 8-bit procesor ATMEL

**Diplomová práca**

| | |
|---|---|
| Študijný program: | Informatika |
| Študijný odbor: | 9.2.1 Informatika |
| Školiace pracovisko: | Katedra počítačov a informatiky (KPI) |
| Školiteľ: | Ing. Peter Feciľak, PhD. |
| Konzultant: | |

**Košice 2013** **Bc. Radovan Kovalčik**

**Abstrakt v SJ**

Hlavným cieľom tejto práce je implementácia protokolu SNMP (Simple Network Management Protocol) do 8-bitového mikrokontroléru ATMEL ATmega162. V prvej časti práce je analyzovaná architektúra mikrokontrolérov rady AVR. V ďalšej časti práce je detailne popísaný protokol SNMP a MIB (Management Information Base), ktorá slúži na adresovanie objektov v protokole SNMP. Zvyšná časť práce je zameraná na návrh a implementáciu vzdialeného riadenia mikrokontrolérov ATmega162 a ATtiny2313 protokolmi SNMP a HTTP (Hypertext Transfer Protocol). Protokol HTTP je využitý pri implementácii grafického používateľského rozhrania vzdialeného riadenia portov mikrokontroléru.

**Kľúčové slová**

Riadenie, mikrokontrolér, AVR, SNMP, HTTP

**Abstrakt v AJ**

The main purpose of this work is the implementation of SNMP (Simple Network Management Protocol) protocol into 8-bit microcontroller ATMEL ATmega162. In the first part of this work is analyzed architecture of AVR microcontroller. In the next part is described structure of SNMP protocol and MIB (Management Information Base) which is used for addressing objects inside SNMP. In the rest of work is designed and implemented remote control of ATmega162 and ATtiny2313 microcontrollers by protocols SNMP and HTTP (Hypertext Transfer Protocol). HTTP is used for implementation of graphical user interface.

**Kľúčové slová v AJ**

Control, microcontroller, AVR, SNMP, HTTP

**TECHNICKÁ UNIVERZITA V KOŠICIACH**
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

# ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **9.2.1 Informatika**
Študijný program: **Informatika**

Názov práce:

**SNMP podpora pre 8-bit procesor ATMEL**
SNMP support for 8-bit ATMEL

Študent (tituly, meno, priezvisko): **Bc. Radovan Kovalčik**
Školiteľ (tituly, meno, priezvisko): **Ing. Peter Feciľak, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce (tituly, meno, priezvisko):
Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Analyzovať architektúru jednoúčelových počítačov (mikrokontrolérov) najmä so zameraním na produkty spoločnosti ATMEL
2. Analyzovať možnosti prepojenia mikroprocesorov Attiny2313, Atmega162 so sieťovým rozhraním za účelom prijatia SNMP správy v prostredí IP siete
3. Porovnať možnosti riadenia vzdialeného zariadenia prostredníctvom protokolov ako je SNMP, telnet, SSH, HTTP
4. Navrhnúť a aplikačne realizovať mechanizmus riadenia mikroprocesorov Attiny2313 a Atmega162 prostredníctvom protokolov SNMP a HTTP
5. Vypracovať dokumentáciu podľa štandardov katedry

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 26.04.2013
Dátum zadania diplomovej práce: 31.10.2012

prof. Ing. Ján Kollár, CSc.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -12-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

**Čestné vyhlásenie**

Vyhlasujem, že som diplomovú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice 26. 4. 2013 .....

*Vlastnoručný podpis*

**Poďakovanie**

Na tomto mieste by som rád poďakoval vedúcemu práce Ing. Petrovi Feciľakovi, PhD. za cenné rady a pripomienky v priebehu konzultácií. Zároveň by som chcel poďakovať rodičom, starým rodičom, rodine a priateľom, ktorí ma podporovali počas celého štúdia na vysokej škole.

# Predhovor

Protokol SNMP je sieťový konfiguračný protokol, ktorý v dnešnej dobe slúži najmä na konfiguráciu, riadenie a monitorovanie sieťových zariadení, ktorými sú hlavne prepínače a smerovače. Jeho výhoda je v jednoduchej implementácii a širokom spektre použitia. Z tohoto dôvodu je zároveň vhodný aj pre riadenie iných sieťových zariadení založených napríklad na mikrokontroléroch.

Obľúbenosť 8-bitových mikrokontrolérov stále rastie kvôli ich malej veľkosti a nízkej spotrebe pri relatívne vysokom výkone, jednoduchom návrhu zariadení a komplexných možnostiach programovania. Pripojenie mikrokontrolérov do siete internet a riadenie ich činnosti prostredníctvom protokolu SNMP vytvára nový rozmer vzdialeného riadenia zariadení prakticky z ktorejkoľvek časti sveta.

Táto práca vznikla ako dôsledok potreby vzdialeného riadenia mikrokontrolérov rady AVR z prostredia počítačovej siete. Hlavná časť práce je zameraná na pripojenie mikrokontroléru ATmega162 do IP siete a implementáciu protokolu SNMP pre potreby vzdialeného riadenia. Riadenie prostredníctvom protokolu HTTP je riešené webovou stránkou s grafickým používateľským rozhraním.

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Slovník termínov

**AES** (Advanced Encryption Standard) symetrická bloková šifra.

**ALU** aritmeticko-logická jednotka.

**ASN.1** (Abstract Syntax Notation One) štandard pre kódovanie a dekódovanie informácií pri prenose v telekomunikačných a počítačových sieťach.

**ATmega162** 8-bitový mikrokontrolér rady MEGA AVR so 16KB programovej flash pamäte, 1KB SRAM pamäte a 512B pamäte EEPROM.

**ATtiny2313** 8-bitový mikrokontrolér rady TINY AVR s 2KB programovej flash pamäte, 128B SRAM pamäte a 128B pamäte EEPROM.

**AVR** rada mikrokontrolérov firmy Atmel.

**CSS** (Cascading Style Sheets) jazyk pre popis vzhľadu webových stránok.

**CRC** (Cyclic Redundancy Check) kontrolný súčet.

**DES** (Data Encryption Standard) symetrický šifrovací algoritmus.

**DMA** (Direct Memory Access) priamy prístup do pamäte.

**EEPROM** elektronicky vymazateľná pamäť len pre čítanie.

**ENC28J60** integrovaný obvod slúžiaci na pripojenie mikrokontrolérov do siete Ethernet.

**Ethernet** technológia počítačových sietí pre lokálne siete.

**HTML** (Hypertext Markup Language) značkovací jazyk určený na tvorbu webových stránok.

**HTTP** (Hypertext Transfer Protocol) aplikačný protokol pre prenos hypermédií.

**LED** (Light-emmiting Diode) svetlo emitujúca dióda.

**MCU** mikrokontrolér.

**MIB** (Managment Information Base) báza riadených informácií a objektov.

**OSI model** model sieťovej komunikácie vyvinutý ako štandard ISO.

**PDU** (Protocol Data Unit) údajová časť SNMP správy.

**PHP** interpretovaný jazyk využívaný pri tvorbe webových stránok.

**PWM** (Pulse Width Modulation) modulácia šírky impulzu.

**RISC** (Reduced Instruction Set Computer) počítač s redukovanou inštrukčnou sadou.

**SNMP** (Simple Network Managment Protocol) protokol správy sieťových zariadení.

**SoC** (System on Chip) riešenie celého systému integrovaného v jednom čipe.

**SPI** (Serial Peripheral Interface) sériové rozhranie pre prenos informácií.

**SRAM** statická pamäť s náhodným prístupom.

**SSH** (Secure Shell) aplikačný protokol šifrovaného vzdialeného terminálu.

**TCP** spoľahlivý transportný sieťový protokol.

**Telnet** aplikačný protokol vzdialeného terminálu.

**UART (USART)** (Universal Asynchronous Receiver and Transmitter) sériový štandard prenosu informácií.

**UDP** nespoľahlivý transportný sieťový protokol.

**USB** (Universal Serial Bus) univerzálna sériová zbernica.

# Úvod

Aj v dnešnej dobe 64 bitových procesorov a množstvom zariadení postavených na architektúre ARM sa 8 bitové mikrokontroléry tešia veľkej obľube v rôznych priemyselných odvetviach. Hlavným, ale zďaleka nie jediným odvetvím, ktoré hojne využíva vysoký výkon a veľmi malú spotrebu mikrokontrolérov je automobilový priemysel. Jednoduchý návrh zariadení spojený s jednoduchým programovaním pomocou jazykov assembler a C ich predurčujú do rôznych priemyselných aplikácií.

Vývoju 8-bitových mikrokontrolérov sa v dnešnej dobe venuje niekoľko veľkých hardvérových firiem. Medzi popredných výrobcov 8-bitových mikrokontrolérov patrí firma ATMEL, ktorej produkty s jadrom AVR sú použité v tejto práci.

Spolu so širokým uplatnením 8-bitových mikrokontrolérov v rôznych zariadeniach vzniká požiadavka ich vzdialeného riadenia. Vďaka rozšírenosti internetu je veľkou výhodou riadenie zariadení postavených na mikrokontroléroch práve po sieti internet. Samotné 8-bitové mikrokontroléry však štandardne nie sú vybavené rozhraním pre pripojenie do siete internet. Preto je jednou z úloh tejto práce nájsť optimálne možnosti pre pripojenie mikrokontroléru do počítačovej siete.

Keďže už existuje množstvo protokolov, pomocou ktorých je možné mikrokontroléry vzdialene riadiť, nie je potrebné vytvárať nový protokol. Použitie niektorého z existujúcich protokolov zároveň zabezpečuje vysokú mieru prenositeľnosti riešení a kompatibility medzi rôznymi systémami. Pre riadenie mikrokontrolérov bol primárne zvolený protokol SNMP, ktorý je v grafickom používateľskom rozhraní doplnený protokolom HTTP.

Hlavným cieľom tejto práce je pripojenie mikrokontroléru do IP siete, implementácia protokolu SNMP softvérom mikrokontroléru, a tým sprístupnenie možnosti vzdialeného riadenia portov mikrokontroléru po sieti internet. Ďalším cieľom je vytvorenie grafického používateľského rozhrania s využitím protokolov HTTP a SNMP.

Práca je rozdelená na šesť kapitol. Prvá kapitola obsahuje formuláciu zadania úloh, ktoré sú v práci riešené.

Druhá kapitola obsahuje analýzu architektúry mikrokontrolérov a ponúka prehľad mikrokontrolérov vyrábaných firmou ATMEL spolu s popisom a rozdelením jednotlivých typov mikrokontrolérov.

V tretej kapitole sa nachádza popis alternatív pripojenia mikrokontrolérov do IP siete a popis integrovaného obvodu ENC28J60. Zároveň je tu analyzovaná architektúra protokolu SNMP vrátane kódovania ASN.1, ktoré tento protokol pri prenose využíva.

Štvrtá kapitola obsahuje návrh mechanizmu riadenia mikrokontroléru zahŕňajúci návrh pripojenia mikrokontroléru do IP siete, porovnanie rôznych protokolov riadenia a návrh riadenia prostredníctvom protokolov SNMP a HTTP.

Piata kapitola sa zaoberá implementáciou vzdialeného riadenia mikrokontroléru na reálnom mikrokontroléri zapojenom na vývojovej doske. Táto časť obsahuje popis vývojového prostredia a detailný popis implementácie riadenia portov pre mikrokontroléry ATmega162 a ATtiny2313. Zároveň táto kapitola obsahuje implementáciu pripojenia mikrokontroléru do IP siete prostredníctvom integrovaného obvodu ENC28J60 a detailný popis implementácie protokolu SNMP v mikrokontroléri. V závere tejto časti sa nachádza popis používateľského prostredia riešeného prostredníctvom technológie PHP a protokolu HTTP.

Šiesta kapitola je záver, ktorý obsahuje zhrnutie celej práce, hodnotenie pohľadu autora na spracovanú problematiku a možnosti ďalšieho rozšírenia implementácie.

# 1 Formulácia úlohy

Prvou úlohou tejto práce je analýza architektúry jednoúčelových počítačov (mikrokontrolérov) najmä so zameraním na produkty spoločnosti ATMEL. Konkrétne ide hlavne o radu mikrokontrolérov AVR a ich vnútornú architektúru.

Druhá úloha pozostáva z analýzy možnosti prepojenia mikroprocesorov ATmega162 a ATtiny2313 so sieťovým rozhraním za účelom prijatia SNMP správy v prostredí IP siete. Za týmto účelom je potrebné porovnať rôzne možnosti pripojenia mikrokontroléru do IP siete. Zároveň je potrebné analyzovať a popísať štruktúru, implementáciu a kódovanie protokolu SNMP.

Treťou úlohou je porovnať možnosti riadenia vzdialeného zariadenia prostredníctvom protokolov ako je SNMP, Telnet, SSH a HTTP. Všetky tieto protokoly sú vhodné na vzdialené riadenie zariadení v IP sieťach a úlohou je porovnanie výhod a nevýhod jednotlivých protokolov a formulácia dôvodov prečo sú pre implementáciu zvolené protokoly SNMP a HTTP.

Štvrtou úlohou je návrh a realizácia mechanizmu riadenia mikroprocesorov ATmega162 a ATtiny2313 prostredníctvom protokolov SNMP a HTTP. Táto úloha pozostáva zo samotného návrhu pripojenia mikrokontroléru do IP siete a softvérovej implementácie protokolu SNMP vo vybraných mikrokontroléroch. Zároveň táto úloha zahŕňa vytvorenie riadenia mikrokontroléru prostredníctvom protokolu HTTP.

Poslednou úlohou je vypracovanie dokumentácie podľa štandardov katedry, ktorá zahŕňa systémovú a používateľskú príručku k vytvoreným softvérovým a hardvérovým riešeniam.

# 2 Analýza architektúry mikrokontrolérov

## 2.1 Mikrokontrolér

Mikrokontrolér [12], monolitický mikropočítač, "microcontroller" alebo MCU predstavuje počítač umiestený v jedinom puzdre, integrovaný na spoločnom kremíkovom substráte. Je charakterizovaný vysokou integráciou, nízkou spotrebou a nízkou cenou. Na rozdiel od mikroprocesorov pre všeobecné použitie mikropočítač obsahuje okrem centrálnej procesorovej jednotky i pamäťový podsystém a vstupno-výstupný podsystém.

Mikrokontroléry sa často používajú v rôznych strojoch a zariadeniach, ako sú automobily, periférne zariadenia počítačov, domáce spotrebiče, hračky a pod. Pretože ich veľkosť, cena a spotreba je v porovnaní so systémom realizovaným na báze mikroprocesora, externých pamätí a vstupno-výstupných obvodov mimoriadne nízka, v jednotlivých aplikáciách sú dnes mikrokontroléry bez vážnej konkurencie. Mnoho vstavaných systémov úspešne pracuje s minimálnymi požiadavkami na veľkosť programu. Ich vstupno-výstupné zariadenia môžu byť tiež veľmi jednoduché, dvojhodnotové, vo forme spínačov a relé. V takýchto aplikáciach je možné využiť čo najjednoduchší mikrokontrolér s minimálnymi nárokmi na výpočtový výkon a s jednoduchými vstupno-výstupnými obvodmi.

Vo väčšine prípadov mikrokontroléry nemajú vyvedenú adresnú a údajovú zbernicu, pamäťový podsystém majú integrovaný spolu s procesorovou jednotkou a vstupno-výstupnými obvodmi v jednom puzdre.

Mikrokontroléry používajú Harvardskú aj von Neumannovu koncepciu. Harvardská koncepcia, pri ktorej je striktne oddelený pamäťový priestor programu a údajov, ponúka niektoré výhodné vlastnosti. Dĺžka slova programu nie je viazaná na dĺžku spracovaných údajov, ktorá býva 4, 8, 16, 32 prípadne až 64 bitov. To umožňuje jednoducho, bez nadväznosti na dĺžku spracovaných údajov, optimalizovať dĺžku inštrukcie

daného inštrukčného súboru. Najmä pri RISC mikrokontrolér̈och, ktoré spracúvajú inštrukciu z pohľadu používateľa v jednom inštrukčnom cykle, je táto skutočnosť veľmi zaujímavá. Ako príklad je možno uviesť 8-bitové RISC mikrokontroléry firmy Microchip, ktoré sa vyrábajú v širokej variete inštrukčných súborov a zodpovedajúcich dĺžok slova programu. Pri Harvardskej koncepcii nie je možné flexibilne využívať pamäť na základe konkrétnych požiadaviek aplikácie. Niektoré aplikácie vyžadujú veľký rozsah pamäte údajov pri malých nárokoch na pamäť programu, iné naopak. V prípade von Neumannovej koncepcie je možné využívať jednu spoločnú pamäť, ako pamäť programu aj údajov na základe aktuálnych požiadaviek aplikácie.

## 2.2 Rozdelenie riešení firmy Atmel

Mikrokontroléry firmy Atmel ponúkajú rôzne typy architektúr, ktoré sa vyznačujú nízkou spotrebou, vysokorýchlostným prepojením s optimálnym využitím šírky pásma a bohatou podporou periférnych rozhraní. Všetky produkty zároveň umožňujú široké možnosti konfigurácie, vďaka ktorým je možné využívať mikrokontroléry v rôznych oblastiach využitia [1].

Základné rozdelenie procesorov firmy Atmel:

- Atmel AVR 8-bit a 32-bit mikrokontroléry
- Mikrokontroléry založené na procesore ARM
- Bezdrôtové mikrokontroléry – 8-bitové mikrokontroléry s modulmi ZigBee, označované ako Z-Link. Tieto mikrokontroléry sú navrhnuté v súlade so štandardom 802.15.4 – ZigBee. Obsahujú 8-bitové jadro AVR RISC s nízkou spotrebou a RF prijímač a vysielač [12].
- Mikrokontroléry s architektúrou Intel 8051 – 8-bitové mikrokontroléry na báze jadra Intel 8051. Spoločnosť ponúka široký sortiment mikrokontrolérov špecializovaných na rôzne aplikácie (MP3, USB, čítačka pamäťových kariet a pod.).

### 2.2.1 Mikrokontroléry založené na procesore ARM

V oblasti 32-bitových mikrokontrolérov sa v produkcii spoločnosti ATMEL stretávame s dvoma základnými triedami [12]. Sú to:

- Mikrokontroléry triedy AT91CAP – vyvinuté ako systémy SoC na podporu špeciálnych zákazníckych riešení. Obsahujú procesorové jadro ARM, rýchlu internú pamäť, široký rozsah periférií a rozhraní a MP (Metal Programmable) blok, ktorý umožňuje obvod doplniť špeciálnymi funkciami na základe požiadaviek vyplývajúcich z aplikácie.
- Mikrokontroléry triedy AT91SAM – univerzálne mikrokontroléry navrhnuté na báze jadra ARM. Po viac než desaťročných skúsenostiach spoločnosti s implementáciou ARM procesorov tvorí dnes produkciu spoločnosti 23 mikrokontrolérov na báze jadra ARM7TDMI a 10 mikrokontrolérov na báze jadra ARM920T a ARM926EJ-S.

### 2.2.2 Mikrokontroléry rady AVR

Rada 8 a 32-bitových mikrokontrolérov firmy Atmel je zameraná na vysoký výkon, malú spotrebu a umožňuje flexibilný dizajn zariadení. Mikrokontroléry tejto rady sú založené na architektúre vhodnej pre programovanie v jazykoch assembler, C a C++. Zároveň umožňujú vysokú prenositeľnosť kódu medzi rôznymi mikrokontrolérmi vďaka rovnakej architektúre pričom je možné aj počas vývoja prejsť na iný typ mikrokontroléru s najvhodnejšími parametrami veľkosti pamätí a dostupnosťou potrebných periférií [1].

Rozdelenie:

- 32-bitové mikrokontroléry AVR UC3 – je séria 32-bitových AVR mikrokontrolérov s vysokým výkonom a nízkou spotrebou. Sú vybavené dvojvstupovou SRAM pamäťou, viacvrstvovou údajovou zbernicou, radičom priameho prí-

stupu do pamäte a systémom udalostí pre periférie. Niektoré verzie obsahujú zabudovanú jednotku pre výpočty v pohyblivej rádovej čiarke (FPU). Podporujú USB, Ethernet a rýchle sériové rozhrania.

- Mikrokontroléry AVR XMEGA – založené na 8-bitovej architektúre AVR dosahujú vysokú integráciu periférií ako 12-bitové A/D a D/A prevodníky, podporu priameho prístupu do pamäte (DMA), AES a DES kryptografické moduly, 32 kanálov PWM, 8 rozhraní UART, 4 SPI kanály, modul generovania kontrolného súčtu CRC a podporu USB zbernice.
- Mikrokontroléry megaAVR – 8-bitové mikrokontroléry s jadrom AVR. Obsahujú veľké množstvo programovateľných vstupov a výstupov, rozsiahlu programovú aj údajovú pamäť a rozsiahle možnosti periférií. Pritom si stále zachovávajú nízku spotrebu pomocou technológie picoPower.
- Mikrokontroléry tinyAVR – 8-bitové mikrokontroléry s jadrom AVR, ktoré ponúkajú vysokú efektivitu v malom prevedení. Ich limitáciou je menšie množstvo programovej aj operačnej pamäte. Okrem menšieho puzdra s menším počtom vývodu a menším počtom periférií sa vnútornou štruktúrou nijako nelíšia od megaAVR.
- Mikrokontroléry pre správu batérií – určené pre správu Li-ion batérií. Sú navrhnuté pre maximalizáciu životnosti a energie dodanej batériou v jednom nabíjacom cykle. Obsahujú dodatočné analógovo-digitálne prevodníky pre meranie nabitia batérie a úrovne napätia.
- Mikrokontroléry AVR pre automobilový priemysel – sú vhodné pre automobilový priemysel a podobné odvetvia. Vyznačujú sa širokým rozsahom pracovných teplôt. V jednom čipe obsahujú AVR mikrokontrolér, regulátor napätia, kontrolu nízkeho napätia, jednotku pre výpočty v pohyblivej rádovej čiarke, ochranu zdrojového kódu, vysokorýchlostný Ethernet a USB.

## 2.3 Architektúra 8-bitových mikrokontrolérov AVR

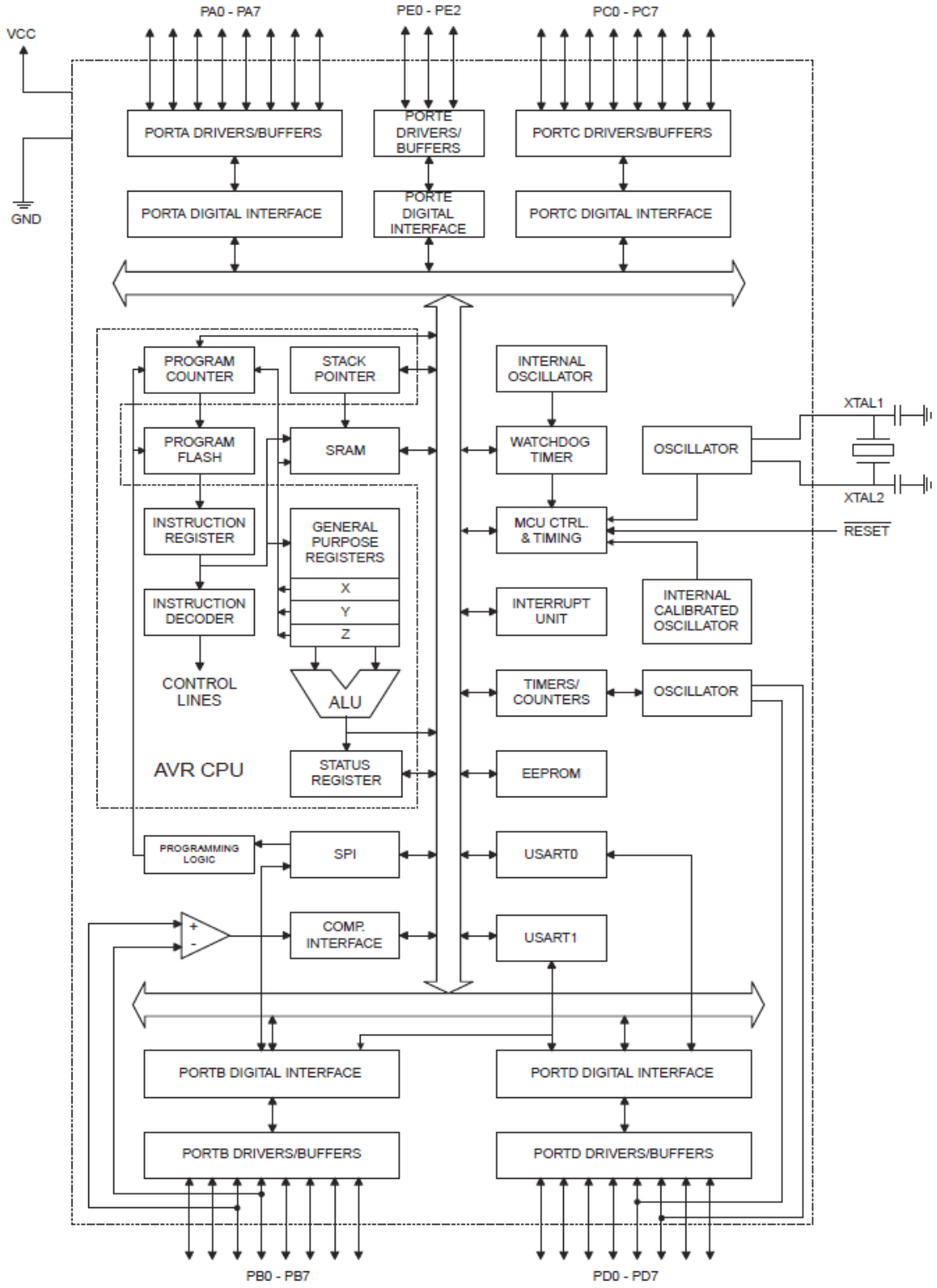
Celá rodina mikrokontrolérov AVR, tinyAVR a megaAVR je založená na rovnakej architektúre. Jednotlivé mikrokontroléry sa líšia vo veľkostiach pamätí flash, SRAM a EEPROM, v type a počte periférií. Nasledujúca časť obsahuje popis architektúry mikrokontroléru ATmega162 ako zástupcu rodiny mikrokontrolérov AVR.

Mikrokontrolér ATmega162 [2], ktorého bloková schéma sa nachádza na obrázku 2 – 1, implementuje registrovú architektúru, pri ktorej sú operandy aj operácie uložené v registroch, ktoré sú priamo prepojené s procesorom (CPU). To znamená, že pred vykonaním operácie sú načítané všetky potrebné údaje do CPU a výsledok operácie je rovnako uložený do registra. Počas vykonávania programu procesor používa registre čím minimalizuje pomalý prístup do pamäte. Spolu s registrovou architektúrou je inštrukčná sada založená na koncepte RISC. RISC procesor obsahuje veľmi jednoduché a efektívne operácie. Komplexné inštrukcie sú budované z týchto veľmi jednoduchých operácií. Toto umožňuje efektívne vykonávania programu. ATmega162 obsahuje 131 inštrukcií typu RISC.

Mikrokontrolér obsahuje 32 8-bitových registrov všeobecného použitia, ktoré sú priamo prepojené s aritmeticko-logickou jednotkou procesora (ALU) vrámci CPU. Procesor implementuje Harvardskú architektúru čo znamená, že obsahuje oddelené pamäte a zbernice pre ukladanie programu a údajov. Registrová Harvardská architektúra v spojení s RISC inštrukčnou sadou umožňujú rýchle a efektívne vykonávania programu a je vďaka ním možné dokončiť každý hodinový cyklus jednu strojovú inštrukciu v jazyku assembler. Pri taktovaní 16MHz je tak možné vykonať 16 miliónov inštrukcií za sekundu.

### 2.3.1 Organizácia pamäte mikrokontroléru ATmega162

ATmega16 obsahuje 3 typy pamätí: flash EEPROM, statickú RAM a bajtovo adresovateľnú EEPROM pre údaje.




**Obrázok 2 – 1**: Bloková schéma mikrokontroléru ATmega162

Pamäť flash EEPROM slúži na uloženie programu alebo veľkého poľa konštánt definovaných ako globálne premenné v programe. Flash pamäť nie je napäťovo závislá čo znamená, že si zachováva svoj obsah aj po odpojení napájacieho napätia. ATmega162 obsahuje 16KB opakovane programovateľnej flash pamäte. Táto pamäť je organizovaná do 8K úložných miest, z ktorých každá má veľkosť 16 bitov. Pamäť flash je možné programovať aj keď je mikrokontrolér priamo zapojený v obvode pomocou technológie ISP (In System Programming).

Bajtovo adresovateľná EEPROM pamäť sa používa na permanentné uloženie údajov a ich opätovné načítanie počas behu programu. Rovnako ako flash je táto pamäť napäťovo nezávislá. Je užitočná pri ukladaní údajov, ktoré musia byť uchované po strate napájania ale môžu byť často menené. ATmega162 obsahuje 512 bajtov EEPROM.

Statická RAM pamäť je napäťovo závislá, čo znamená že po strate napájania sa stratí celý obsah tejto pamäte. Je možné do nej zapisovať a čítať z nej počas behu programu. ATmega162 obsahuje 1000 bajtov (presne 1120) pamäte SRAM. Malá časť (96 pamäťových miest) SRAM je použitých pre registre všeobecného použitia používanými CPU a tiež pre konfiguráciu periférií. Počas behu programu je SRAM použitá na uloženie globálnych premenných, dynamickú alokáciu premenných a poskytuje pamäťový priestor pre zásobník.

### 2.3.2 Systém vstupno-výstupných portov mikrokontroléru

Atmel ATmega162 obsahuje päť 8-bitových vstupno-výstupných portov všeobecného použitia nazývané PORTA, PORTB, PORTC, PORTD a PORTE. Všetky tieto porty majú alternatívne funkcie, ktorých aktiváciou sa z výstupov portu stávajú vstupno-výstupné periférie.

Pre každý port existujú 3 registre: údajový register (PORT), ktorý sa používa na výstup údajov z portu, smerový register (DDR), ktorý umožňuje nastaviť každý vývod

portu ako vstupný alebo výstupný a vstupný register (PIN) používaný na čítanie údajov z portu.

Tabuľka 2 – 1 popisuje nastavenia potrebné pre konfiguráciu špecifického vývodu portu ako vstup alebo výstup. V prípade vstupu môže byť vývod nastavený tak, aby sa správal ako zdroj prúdu alebo pracovať v režime vysokej impedancie. Ak je vývod nastavený ako výstupný môže byť nastavený do logickej ”0” alebo logickej ”1”. Obyčajne sú porty nakonfigurované na začiatku programu a sú im priradené počiatočné hodnoty.

| DDR | PORT | Vstup/výstup | Pull-up | Popis |
|---|---|---|---|---|
| 0 | 0 | vstup | nie | Troj stavový (vysoká impedancia). |
| 0 | 1 | vstup | áno | Vývod sa správa ako zdroj prúdu. |
| 1 | 0 | výstup | nie | Výstup log. ”0”. |
| 1 | 1 | výstup | nie | Výstup log. ”1”. |

**Tabuľka 2 – 1**: Nastavenie vývodov portu

# 3 Analýza pripojenia mikrokontroléru do IP siete

## 3.1 Pripojenie mikrokontrolérov do IP siete

Mikrokontroléry s jadrom AVR štandardne neobsahujú rozhranie pre pripojenie do siete Ethernet. Implementácia rozhrania siete Ethernet priamo v mikrokontroléri je možná ale kvôli veľkej zložitosti štandardu Ethernet je pomerne komplikovaná a navyše by bolo možné implementovať toto rozhranie len do mikrokontrolérov s veľkou programovou flash pamäťou a rovnako veľkou pamäťou SRAM. Pri tomto riešení by mohol nastať prípad, že by veľká časť programovej pamäte bola obsadená implementáciou rozhrania Ethernet a neostalo by miesto na ďalšiu nevyhnutnú funkcionalitu ako je implementácia vzdialeného riadenia.

Z tohoto dôvodu je pre pripojenie mikrokontroléru do IP siete použité externé riešenie, ktoré implementuje sieťovú funkcionalitu, a ktoré s mikrokontrolérom komunikuje sériovým rozhraním. V dnešnej dobe je možné vyberať z veľkého množstva riešení do rôznych výrobcov. Väčšina integrovaných obvodov, ktoré implementujú prevodník z Ethernetu na sériové rozhranie je však v prevedení SMD s veľmi malými rozmermi a hustou integráciou a preto nie sú veľmi vhodné pre zapojenia konštruované na kontaktnom poli. Tento nedostatok odstránil integrovaný obvod ENC28J60, ktorý je dostupný v DIP prevedení a v 28 vývodovom puzdre obsahuje implementáciu fyzickej a spojovej vrstvy OSI modelu čím umožňuje zariadeniu prijímať IP pakety prostredníctvom sériového rozhrania SPI.

Inou možnosťou pripojenia mikrokontroléru je kompletné riešenie prevodníku Ethernet na sériové rozhranie, ktorý implementuje fyzickú, spojovú, sieťovú aj transportnú vrstvu OSI modelu a úplne tak odstraňuje potrebu vývoja sieťovej komunikácie. Programátor sa tak môže zamerať priamo na implementáciu protokolu aplikačnej vrstvy. Predstaviteľom tejto triedy je napríklad Lantronix XPort.

### 3.1.1 Integrovaný obvod ENC28J60

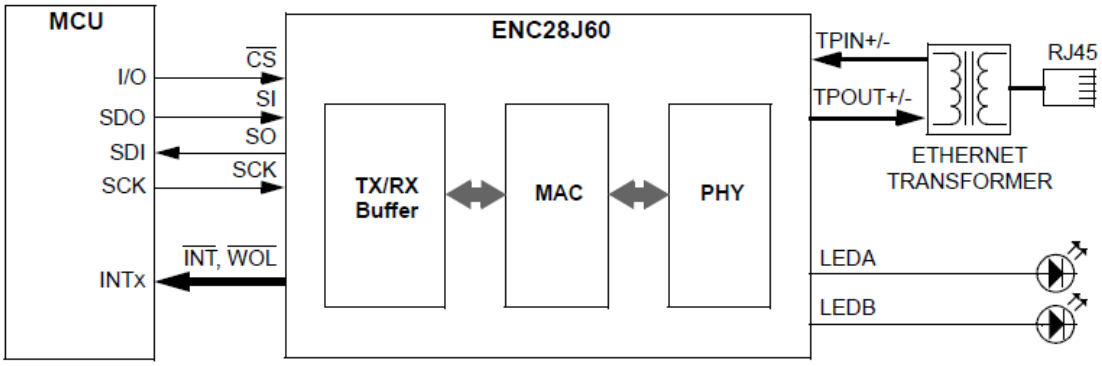
Integrovaný obvod ENC28J60 [5] je samostatný radič rozhrania Ethernet so zabudovaným sériovým rozhraním SPI. ENC28J60 spĺňa štandard IEEE 802.3, ktorý zahŕňa množstvo filtračných schém sieťovej komunikácie pre limitovanie prichádzajúcich paketov. Tiež obsahuje zabudovaný DMA modul a hardvérový výpočet kontrolného súčtu IP hlavičky. Komunikácia s mikrokontrolérom je realizovaná prostredníctvom dvoch výstupov prerušení a sériového rozhrania SPI s údajovým tokom do 10Mb/s. Dva výstupy sú určené pre LED diódy stavu siete a indikácie sieťovej aktivity.

Typické zapojenie integrovaného obvodu ENC28J60 a jeho prepojenie s Ethernetovým konektorom a mikrokontrolérom je zobrazené na obrázku 3 – 1. Dvojitý oddeľovací transformátor a niekoľko pasívnych súčiastok je všetko, čo je potrebné na pripojenie mikrokontroléru do 10Mbps siete Ethernet.

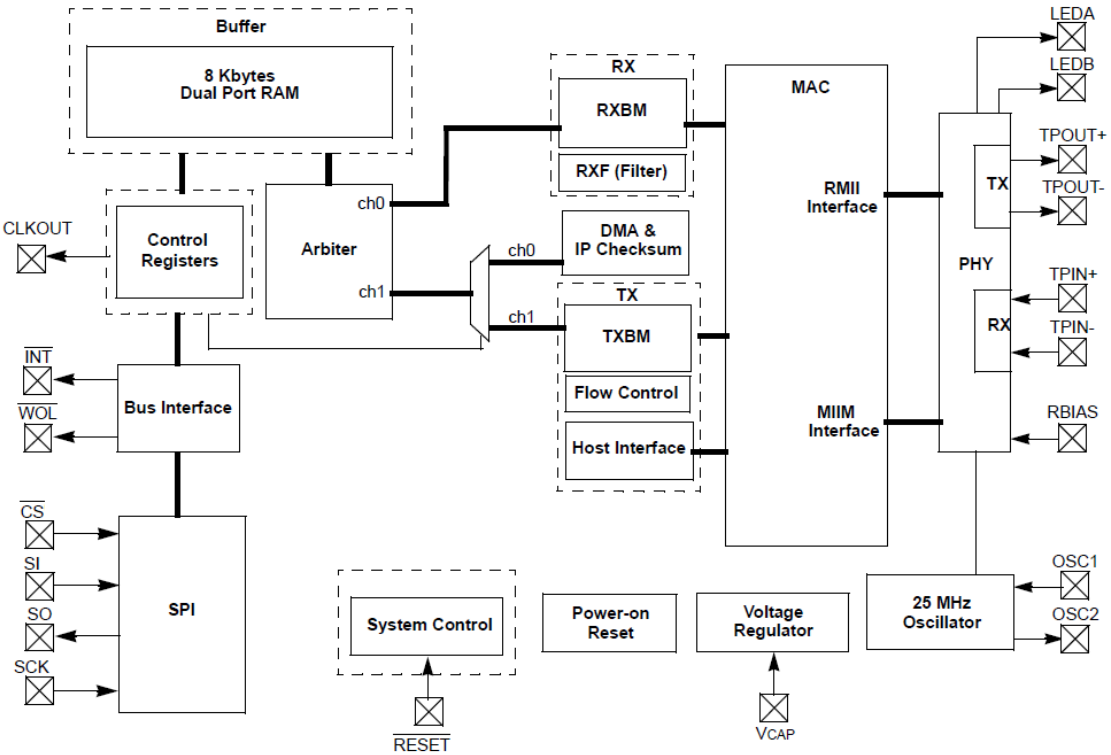
ENC28J60 pozostáva zo siedmich hlavných funkčných blokov (obr. 3 – 2):

1. SPI rozhranie, ktoré sprostredkúva komunikáciu medzi mikrokontrolérom a integrovaným obvodom ENC28J60
2. Riadiaci register, ktorý sa používa na riadenie a monitorovanie obvodu.
3. Zásobník dvojvstupovej pamäte RAM pre prijaté a odosielané pakety.
4. Arbiter riadiaci prístup do zásobníka pamäte RAM pri vzniku požiadavky od radiča priameho prístupu do pamäte (DMA).
5. Rozhranie zbernice interpretujúce údaje a príkazy prijaté cez SPI rozhranie.
6. MAC (Medium Access Control) modul implementujúci IEEE 802.3 logiku.
7. PHY (Physical Layer) modul, ktorý kóduje a dekóduje analógové údaje nachádzajúce sa na vstupnom rozhraní konektora RJ-45.

Zariadenie tiež obsahuje podporné bloky ako oscilátor, regulátor napätia, menič logickej úrovne a systémovú logiku umožňujúcu prácu s 5V napäťovými úrovňami.




**Obrázok 3 – 1**: Typické zapojenie integrovaného obvodu ENC28J60




**Obrázok 3 – 2**: Blokový diagram integrovaného obvodu ENC28J60

## 3.2 Riadenie mikrokontroléru z IP siete

Funkcia manažmentu siete je súčasťou aplikačnej vrstvy protokolovej architektúry aj napriek tomu, že sa nejedná o aplikáciu v pravom slova zmysle. Je to súhrn činností, ktoré aplikáciám slúžia, t.j. umožňujú ich hladký priebeh [14].

Riešenie manažmentu sieťových zariadení v rámci architektúry TPC/IP, protokol SNMP (Simple Network Managment Protocol), je dnes najrozšírenejším riešením v sieťach. Základné aspekty, ktoré sú určujúce pre každý prístup k manažmentu v heterogénnych sieťach sú nasledujúce:

- organizácia manažmentu – ako je manažment organizovaný medzi zainteresované prostriedky
- štruktúra informácií pre manažment – aký je formát a atribúty informácií, pomocou ktorých sa vykonáva riadenie a zmeny údajov
- uloženie informácií pre manažment – kde a ako sa informácie pre manažment ukladajú (ako vyzerá báza informácií pre manažment, MIB)
- protokol manažmentu – ako sú informácie určené pre manažment dostupné (aké operácie je možné nad nimi vykonávať) a ako sa prenášajú.

### 3.2.1 Protokol SNMP

SNMP [14] je asynchrónny, transakčne orientovaný protokol založený na transakciách typu požiadavka/odpoveď medzi sieťovým manažérom a agentom v sieti. Prevažná väčšina monitorovacích a riadiacich činností sa vykonáva na základe výzvy zo strany manažéra. Iba malá časť správ prichádza k manažérovi bez predošlej výzvy, sú to takzvané pasce (trap), správy o mimoriadny udalostiach, na základe ktorých manažér iniciuje ďalšie činnosti.

SNMP v zásade využíva len dva príkazy, a to na vyhľadávanie (”prečítaj hodnotu

v MIB", get) a na zmenu premennej ("prepíš hodnotu v MIB", set) namiesto celého súboru imperatívnych príkazov. Protokolová entita na vzdialenom počítači zaháji interakciu s agentom sídliacim na sieťovom prostriedku prostredníctvom príkazov get alebo set. Minimalizácia typov operácií a správ protokolu SNMP sleduje základnú požiadavku na manažment vrámci internetu – jednoduchosť a jednoduchú implementáciu.

Protokol SNMPv1 je definovaný 3 dokumentami RFC:

- RFC1155 – Štruktúra a identifikácia raidených informácií v sietiach založených na TCP/IP [15].
- RFC1213 – Báza riadených informácií pre sieťový manažment sietí založených na TCP/IP: MIB-II [11].
- RFC1157 – Jednoduchý protokol pre správu siete (SNMP) [4].

Prvé dva dokumenty popisujú štruktúru a organizáciu údajov, ktoré je možné prostredníctvom SNMP protokolu spravovať a tretí dokument popisuje špecifikáciu samotného protokolu.

Pomocou SNMP sú získavané a menené hodnoty premenných v MIB agenta. Komunikácia medzi protokolovými entitami je uskutočňovaná výmenou správ, pričom každá správa je nezávisle reprezentovaná v rámci jedného UDP datagramu použitím jednoduchého pravidla kódovania štandardu ASN.1. Správa pozostáva z identifikátora verzie, komunitného reťazca a protokolovej údajovej jednotky (PDU). Protokolová entita prijíma správy na UDP porte 161, ktorý je určený pre všetky typy správ s výnimkou trap správy. Trap správy môžu byť prijímané na UDP porte 162 pre ďalšie spracovanie. Implementácia protokolu nevyžaduje správy dlhšie ako 484 bajtov. Je nevyhnutné, aby všetky implementácie SNMP podporovali 5 typov PDU: `GetRequestPDU`, `GetNextRequestPDU`, `GetResponsePDU`, `SetRequestPDU` a `TrapPDU` [4].

### 3.2.2 Organizácia a architektúra manažmentu protokolu SNMP

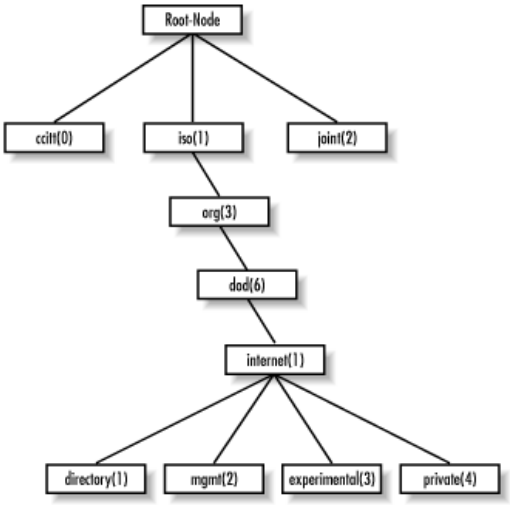
Funkcie a úloha manažmentu sa distribuujú medzi dvoch účastníkov [14]:

- Manažér – programové vybavenie, ktoré je umiestnené na stanici sieťového manažmentu. Manažér s riadenými objektami manipuluje prostredníctvom agentov. Každý manažér má teda na starosti skupinu sieťových zariadení, nad ktorými vykonáva dohľad. Manažér u seba centralizuje väčšinu riadiacich úloh (monitorovanie, kontrola, konfigurácia) a je zodpovedný za zber informácií o stave riadených objektov od agentov.
- Agent – programové vybavenie umiestnené na riadenom sieťovom zariadení, ktoré udržuje databázu riadených objektov (MIB, napr. informácie o konfigurácii alebo prevádzke) a komunikuje s manažérom. Agent vykonáva operácie (čítanie a modifikácia) na riadených objektoch predovšetkým na základe oprávnených výziev k operáciám od manažéra a odpovedá na dopyty manažéra vyhľadaním údajov v MIB.

### 3.2.3 Štruktúra informácií pre manažment protokolu SNMP

Informácie pre manažment sú informácie o riadených objektoch vo vnútri otvoreného systému, ktoré je možné prenášať alebo inak ovplyvňovať s využitím protokolov pre manažment. Sú nezávislé na architektúre systému manažmentu rovnako ako na mechanizme komunikácie. Pre dosiahnutie vzájomnej prepojiteľnosti medzi otvorenými systémami je nevyhnutné, aby každý z nich mal rovnakú interpretáciu týchto informácií [14].

Informácie o riadených objektoch sú oddelené od funkčných modelov a sú uložené v báze informácií pre manažment (MIB, Managment Information Base). MIB je model riadených objektov (abstrakcia systémových prostriedkov bez ohľadu na potrebu riadenia), ktoré sú prístupné pre agentov a manipulovateľné manažérmi. MIB




**Obrázok 3 – 3**: Strom identifikátorov objektov v MIB

je hierarchicky usporiadaná množina objektov definovaná syntaxou a sémantikou.

Množina pravidiel spôsobu definície a identifikácie premenných MIB je popísaná štruktúrou informácií pre manažment (SMI, Structure Managment Informantion), ktorá predpisuje použitie formálneho abstraktného jazyka ISO ASN.1 (Abstract Syntax Notation One) zaručujúceho jednoznačnú formu aj obsah jednotlivých objektov.

### 3.2.4 Identifikátory objektov v MIB

Riadené objekty sú organizované do stromovej hierarchie, ktorá je základom pre pomenovanie objektov v protokole SNMP. Identifikátor objektu pozostáva zo série celých nezáporných čísel zodpovedajúcich uzlom stromu oddelených bodkami. Jednoduchšie čitateľná forma zápisu identifikátorov objektov pozostáva z reťazcov, ktoré sú alternatívami pre číselné pomenovanie uzlov stromu. Obrázok 3 – 3 zobrazuje základnú štruktúru stromu identifikátorov objektov, ktorá sa používa pre protokol SNMP [10].

Hlavnou časťou, zaujímavou z pohľadu protokolu SNMP, je podstrom globálnej MIB

`iso(1).org(3).dod(6).internet(1)`, ktorý je v číselnej podobe reprezentovaný ako `1.3.6.1` alebo `iso.org.dod.internet`. Vetva directory z tohoto podstromu nie je momentálne využívaná. Vetva managment alebo mgmt definuje štandardnú množinu riadených objektov internetu. Experimentálna vetva je vyhradená pre testovacie a výskumné účely. Objekty pod vetvou private sú definované organizáciami, ktoré sú zodpovedné za objekty v ich vetvách.

### 3.2.5 Štruktúra správ protokolu SNMP

V dokumente RFC1157 [4] je štruktúra protokolu SNMPv1 definovaná pomocou notácie ASN.1 (zdrojový kód 1). Každá správa sa teda skladá z verzie, komunitného reťazca, ktorý slúži ako bezpečnostný prvok a údajov. Pričom údaje sú zabalené do jedného z 5 typov PDU: `GetRequestPDU`, `GetNextRequestPDU`, `GetResponsePDU`, `SetRequestPDU` a `TrapPDU`.

**Zdrojový kód 1: Štruktúra správ protokolu SNMP**

```
1  RFC1157-SNMP DEFINITIONS ::= BEGIN
2    IMPORTS
3      ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
4        FROM RFC1155-SMI;
5
6    Message ::= SEQUENCE {
7      version
8        INTEGER {
9          version-1(0)
10       },
11     community
12       OCTET STRING,
13     data
14       ANY
15   }
16
17   PDUs ::= CHOICE {
18     get-request
19       GetRequest-PDU,
20     get-next-request
21       GetNextRequest-PDU,
22     get-response
23       GetResponse-PDU,
24     set-request
25       SetRequest-PDU,
```

```
26      trap
27        Trap-PDU
28    }
29
30    RequestID ::=
31      INTEGER
32    ErrorStatus ::=
33      INTEGER {
34        noError(0),
35        tooBig(1),
36        noSuchName(2),
37        badValue(3),
38        readOnly(4),
39        genErr(5)
40      }
41    ErrorIndex ::=
42      INTEGER
43
44    VarBind ::=
45      SEQUENCE {
46        name
47          ObjectName,
48        value
49          ObjectSyntax
50      }
51    VarBindList ::=
52      SEQUENCE OF
53        VarBind
54
55    GetRequest-PDU ::=
56    [0]
57      IMPLICIT SEQUENCE {
58        request-id
59          RequestID,
60        error-status
61          ErrorStatus,
62        error-index
63          ErrorIndex,
64        variable-bindings
65          VarBindList
66      }
```

Každá správa je označená jedinečným identifikátorom reprezentovaným celým číslom `request-id`. Tento identifikátor slúži na spárovanie požiadaviek a odpovedí. SNMPv1 ponúka 5 typov chybových hlásení reprezentovaných celými číslami v premennej `error-status` a celým číslom v premennej `error-index`, ktoré slúži na bližšiu špecifikáciu chyby.

Pojem premenná označuje inštanciu riadeného objektu. Dvojica `VarBind` slúži na priradenie mena premennej a jej hodnoty. `VarBindList` je zoznam premenných a ich hodnôt. V niektorých typoch PDU je dôležitý len názov a nie hodnota premennej ako napríklad pri `GetRequestPDU`. V tomto prípade je časť s hodnotou premennej ignorovaná protokolovou entitou. Avšak časť s hodnotou musí mať správnu ASN.1 syntax aj kódovanie. Odporúča sa použiť ASN.1 hodnota `NULL` pre hodnoty takýchto PDU.

`GetRequestPDU`, `GetNextRequestPDU`, `GetResponsePDU` ako aj PDU typu `SetRequestPDU` majú podobnú štruktúru s tým, že sa líšia iba v identifikátore PDU. Štruktúra `TrapPDU` sa od ostatných líši ale keďže v tejto práci nebude `TrapPDU` využitá jej popis nie je uvádzaný.

`GetRequestPDU` má číselné označenie 0, za ktorým nasleduje `request-id` označujúce poradové číslo PDU. Chybové hlásenie a index chyby majú vždy nulovú hodnotu. Na konci PDU sa nachádza zoznam premenných. Protokolová entita potvrdzuje `GetRequestPDU` vykonaním ktoréhokoľvek použiteľného pravidla z nasledujúceho zoznamu:

1. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných identifikátor objektu nezodpovedá presne žiadnemu identifikátoru objektu v MIB, nad ktorým je možné vykonať operáciu `get`, potom prijímajúca entita odošle `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý nastaví na `noSuchName` a `error-index` označujúcim premennú, ktorá túto chybu spôsobila.

2. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných je objekt agregovaný typ, potom prijímajúca entita odošle `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý v tomto prípade nastaví na hodnotu `noSuchName` a `error-index` označujúcim premennú, ktorá túto chybu spôsobila.

3. Ak veľkosť GetResponsePDU, generovanej ako je popísané nižšie, presiahne lokálny limit, potom prijímajúca entita odošle GetResponsePDU s identickým obsahom okrem error-status, ktorý nastaví na tooBig a premennú error-index na hodnotou 0.

4. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných hodnota objektu nemôže byť získaná pre príčiny, ktoré neboli popísané v predchádzajúcich prípadoch, potom prijímajúca entita odošle GetResponsePDU s identickým obsahom okrem error-status, ktorý v tomto prípade nastaví na hodnotu genErr a error-index označujúcim premennú, ktorá túto chybu spôsobila.

Ak nie je aplikované žiadne pravidlo z predchádzajúcich, potom prijímajúca entita odošle GetResponsePDU takú, že každej premennej je priradená jej hodnota, error-status je noError, error-index je nastavený na 0 a request-id má rovnakú hodnotu ako hodnota request-id prijatej správy.

Štruktúra GetNextRequestPDU je rovnaká ako štruktúra GetRequestPDU s tým rozdielom, že GetNextRequestPDU má číselné označenie 1. Protokolová entita potvrdzuje GetNextRequestPDU vykonaním ktoréhokoľvek použiteľného pravidla z nasledujúceho zoznamu:

1. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných tento identifikátor nie je lexikografickým predchodcom ktoréhokoľvek objektu v MIB, nad ktorým je možné vykonať operáciu get, potom prijímajúca entita odošle GetResponsePDU s identickým obsahom okrem error-status, ktorý nastaví na noSuchName a error-index označujúcim premennú, ktorá túto chybu spôsobila.

2. Ak veľkosť GetResponsePDU generovanej ako je popísané nižšie presiahne lokálny limit, potom prijímajúca entita odošle GetResponsePDU s identickým obsahom okrem error-status, ktorý v tomto prípade nastaví na hodnotu

`tooBig` a `error-index` s hodnotou nula.

3. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných hodnota objektu, ktorý je lexikografickým nasledovníkom premennej nemôže byť získaná pre príčiny, ktoré neboli popísané v predchádzajúcich prípadoch, potom prijímajúca entita odošle `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý v tomto prípade nastaví na `genErr` a `error-index` označujúcim premennú, ktorá túto chybu spôsobila.

Ak nie je aplikované žiadne pravidlo z predchádzajúcich, potom prijímajúca entita odošle `GetResponsePDU` takú, že každej premennej je priradený identifikátor a hodnota premennej, ktorá je lexikografickým nasledovníkom pôvodnej premennej, `error-status` je nastavený na `noError`, `error-index` je 0 a hodnota `request-id` má rovnakú hodnotu ako je hodnota `request-id` prijatej správy.

Štruktúra `SetRequestPDU` je rovnaká ako štruktúra `GetRequestPDU` s tým rozdielom, že `SetRequestPDU` má číselné označenie 3. Protokolová entita potvrdzuje `SetRequestPDU` vykonaním ktoréhokoľvek použiteľného pravidla z nasledujúceho zoznamu:

1. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných meno objektu nezodpovedá presne žiadnemu menu objektu v MIB, nad ktorým je možné vykonať operáciu `set`, potom prijímajúca entita odošle správu typu `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý nastaví na `noSuchName` a `error-index` označujúcim premennú, ktorá túto chybu spôsobila.

2. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných hodnota, neobsahuje správny typ alebo dĺžku, potom prijímajúca entita odošle správu `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý nastaví na `badValue` a `error-index` označujúcim premennú, ktorá túto chybu spôsobila.

3. Ak veľkosť `GetResponsePDU` generovanej ako je popísané nižšie presiahne lokálny limit, potom prijímajúca entita odošle `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý v tomto prípade nastaví na `tooBig` a `error-index` na hodnotou nula.

4. Ak pre ktorýkoľvek identifikátor premennej v zozname premenných hodnota objektu nemôže byť zmenená pre príčiny, ktoré neboli popísané v predchádzajúcich prípadoch, potom prijímajúca entita odošle `GetResponsePDU` s identickým obsahom okrem `error-status`, ktorý nastaví na hodnotu `genErr` a `error-index` označujúcim premennú, ktorá túto chybu spôsobila.

Ak nie je aplikované žiadne pravidlo z predchádzajúcich, potom je každej premennej v zozname premenných priradená nová hodnota a prijímajúca entita odošle `GetResponsePDU` s identickým obsahom, aký mala `SetRequestPDU`.

### 3.2.6 Popis notácie ASN.1

ASN.1 (Abstract Syntax Notation One) [8] je medzinárodne štandardizovaná notácia pre špecifikáciu údajových štruktúr s vysokým stupňom abstrakcie nezávislá na výrobcoch, platforme a jazyku. Podporuje pravidlá, ktoré určujú presné poradie bitov na reprezentáciu údajových štruktúr, ktoré majú byť prenesené po počítačovej sieti. Množstvo softvérových platforiem a programovacích jazykov obsahuje nástroje na mapovanie údajových štruktúr na objekty ASN.1.

Notácia ASN.1 je definovaná ako štandard ISO/IEC 8824-1 [6] a kódovanie v štandarde ITU-T X.690 [7].

# 4 Návrh mechanizmu riadenia mikrokontrolérov

Nasledujúca časť práce bude zameraná na návrh riešenia pripojenia mikrokontroléru do IP siete a jeho následné vzdialené riadenie, ktoré bude realizované pomocou protokolu SNMP implementovaného priamo v mikrokontroléri a protokolu HTTP implementovaného vo vzdialenom webovom serveri, ktorý bude poskytovať grafické používateľské rozhranie a s využitím technológie PHP bude riadiť mikrokontrolér protokolom SNMP.

## 4.1 Prepojenie mikrokontroléru do IP siete

Pre riadenie mikrokontroléru prostredníctvom IP siete je nevyhnutné tento mikrokontrolér do IP siete fyzicky pripojiť. Keďže 8-bitové mikrokontroléry rady AVR neobsahujú priamo rozhranie pre pripojenie do siete Ethernet a softvérová implementácia Ethernetu priamo v mikrokontroléri je nevýhodná, ako už bolo spomenuté v analýze pripojenia do IP siete, bolo potrebné nájsť iné riešenie pripojenia mikrokontroléru do siete Ethernet.

Pre účely tejto práce sa ako najvýhodnejšie ukázalo použiť modul, ktorý komunikuje s mikrokontrolérom prostredníctvom sériového rozhrania, ktorým mikrokontrolér disponuje.

V počiatočnej fáze vývoja protokolu SNMP bol pre pripojenie do siete Ethernet a následnú komunikáciu prostredníctvom IP protokolu použitý Lantronix XPort, ktorý komunikoval s mikrokontrolérom prostredníctvom dvojice vodičov sériovej zbernice RS-232. Výhoda tohoto riešenia spočívala v tom, že Lantronix XPort úplne implementuje sieťovú a transportnú vrstvu OSI modelu a tým odľahčuje mikrokontrolér od kódu, ktorý by bol na implementáciu týchto vrstiev potrebný. Mikrokontrolér tak po sériovej zbernici dostáva priamo údaje z UDP alebo TCP paketu a môže ich spracovať. Jednou z nevýhod je statická konfigurácia Lantronix XPortu.

Je nutné priamo určiť, z ktorej IP adresy budú údaje presmerované do sériovej zbernice, a na ktorú IP adresu a port majú byť odosielané údaje prijímané sériovou zbernicou. Táto vlastnosť spôsobuje, že by bolo možné pripojený mikrokontrolér riadiť iba z jedného počítača s vopred definovanou IP adresou. Neskôr sa však ukázalo, že ani táto funkcionalita nie je pomocou Lantronix XPortu realizovateľná, pretože manažér SNMP protokolu posiela správy z náhodného portu a na tomto porte očakáva aj odpoveď. Lantronix XPort ale nedokáže informovať mikrokontrolér o tom, z ktorého UDP portu správa prišla a mikrokontrolér nedokáže nastaviť Lantronix XPort na odosielanie správy z iného ako preddefinovaného UDP portu. Táto funkcionalita by mala byť realizovateľná v prípade TCP protokolu ale keďže protokol SNMP je primárne určený pre protokol UDP bolo nevyhnutné nahradiť Lantronix XPort iným riešením.

Z tohoto dôvodu bol pre vývoj implementácie riadenia mikrokontrolérov prostredníctvom SNMP protokolu použitý integrovaný obvod ENC28J60. Tento narozdiel od Lantronix XPortu komunikuje s mikrokontrolérom po sériovej zbernici SPI. Nevýhodou tohoto riešenia je, že obvod ENC28J60 implementuje iba fyzickú (PHY) a spojovú (MAC) vrstvu OSI modelu a zvyšné vrstvy je nutné implementovať priamo v mikrokontroléri. Ide o implementáciu IP zásobníka (IP stack), ktorý zahŕňa protokoly IP, ARP, ICMP, UDP a TCP. Táto funkcionalita vyžaduje pomerne veľké množstvo programovej (flash) a operačnej (SRAM) pamäte.

## 4.2 Porovnanie protokolov SNMP, Telnet, SSH a HTTP

Riadenie vzdialeného zariadenia pripojeného do IP site je možné realizovať pomocou rôznych protokolov aplikačnej vrstvy OSI modelu. Jednou z možností je navrhnutie a implementácia vlastného protokolu. Toto riešenie vyžaduje vývoj a testovanie vlastného protokolu, pre ktorý je potrebné implementovať aj nástroje pre riadenie na strane klienta.

Oveľa výhodnejšie je využitie niektorého z už existujúcich protokolov. Najvýhodnejšie je použitie niektorého z protokolov Telnet, SSH, HTTP alebo SNMP. Tieto sú dostatočne rozšírené a všetky ponúkajú rozsiahle možnosti riadenia vzdialeného sieťového zariadenia.

### 4.2.1 Protokol Telnet

Úlohou Telnetu [13] je poskytovať všeobecnú obojsmernú osem bitovú bajtovo orientovanú komunikáciu. Jeho hlavným cieľom je poskytovať štandardnú metódu pripájania sa k vzdialeným terminálovým zariadeniam a terminálovo orientovaným procesom. Protokol môže byť tiež použitý na komunikáciu medzi terminálmi a medzi procesmi navzájom. Telnet využíva architektúru klient/server.

Ide o protokol virtuálneho terminálu. Protokol sprostredkúva možnosť prihlásiť sa zo vzdialeného počítača pre interaktívnu prácu na inom počítači. Protokol potrebuje spoľahlivú transportnú vrstvu (pracuje nad TCP). Číslo portu používané protokolom Telnet je štandardne 23 [14].

Základné charakteristiky protokolu Telnet:

- Definuje sieťový virtuálny terminál, ktorý poskytuje jednotné rozhranie pre prístup k vzdialeným systémom a umožňuje zistiť či susedná aplikácia funguje, signalizovať prerušenia, požadovať prerušenia vzdialeného procesu a pod..
- Umožňuje klientovi aj serveru vzájomne rovnocenné vyjednávanie voliteľných režimov z určitej preddefinovanej štandardnej množiny (napr. sedem alebo osem bitový prenos), takže je možné využívať viac služieb ako poskytuje základná definícia virtuálneho terminálu.
- Považuje spojenie za symetrické, nerozlišuje medzi terminálmi a procesmi, takže dovoľuje, aby sa klientom stal ktorýkoľvek proces.

### 4.2.2 Protokol SSH

SSH [10] vytvára kanál pre spúšťanie terminálu na vzdialenom počítači s obojsmerným šifrovaním medzi lokálnym a vzdialeným počítačom. Z aplikačnej stránky je podobný Telnetu avšak vďaka šifrovaniu je výrazne bezpečnejší.

Protokol SSH je špecifikácia popisujúca ako vytvárať zabezpečenú komunikáciu v sieti. Protokol zahŕňa tieto oblasti:

- Autentifikácia – spoľahlivo určuje identitu používateľa.
- Šifrovanie – šifruje údaje čím sa stávajú nečitateľné pre útočníkov.
- Integrita – garantuje, že údaje prechádzajú sieťou bez zmeny treťou osobou.

### 4.2.3 Protokol HTTP

HTTP [14] je aplikačný protokol vyvinutý pre distribuované, spolupracujúce informačné systémy používajúce hypermédia. Jedná sa o všeobecne použiteľný, objektovo orientovaný protokol. Protokol HTTP definuje súbor pravidiel pre prístup k súborom rôzneho charakteru a pre prenos informácií. Dovoľuje prístup k prostriedkom dostupným z rôznych aplikácií, preto zjednodušuje implementáciu používateľského agenta.

Protokol HTTP je založený na požiadavkách a odpovediach v režime klient/server a ponúka radu metód a formátov správy pre špecifikáciu účelu danej požiadavky. HTTP potrebuje spoľahlivú transportnú službu so spojením (TCP) a využíva port 80.

### 4.2.4 Protokol SNMP

Protokol SNMP, detailne popísaný v analytickej časti práce, je bitovo orientovaný protokol využívajúci transportný protokol UDP. Slúži primárne na vzdialenú kon-

figuráciu a správu sieťových zariadení v IP sieťach. Jeho hlavnou výhodou je malé množstvo typov správ a spolu s tým spojená jednoduchá implementácia. Pri protokole SNMP je oddelená časť riadenia a časť údajov. Údaje sú uložené v objektoch s vlastnými identifikátormi organizovanými v báze riadených objektov MIB.

### 4.2.5 Porovnanie protokolov

Pri výbere vhodného protokolu pre riadenie mikrokontroléru bolo potrebné zohľadniť niekoľko kritérií. Porovnanie protokolov (tab. 4 – 1) zobrazuje hlavné rozdiely medzi jednotlivými protokolmi. V stĺpcoch tabuľky sa nachádzajú jednotlivé protokoly a v riadkoch vlastnosti týchto protokolov. Prvou porovnávanou vlastnosťou je typ komunikácie. Táto vlastnosť rozdeľuje protokoly na textovo orientované, pre spracovanie ktorých je potrebné rozpoznávať textové informácie a bajtovo orientované, ktoré sú kódované do podoby bajtov reprezentujúcich požadované informácie. Druhý riadok tabuľky porovnáva použitie transportnej služby príslušným protokolom. Protokoly Telnet, SSH a HTTP využívajú spoľahlivú transportnú službu TCP zatiaľčo protokol SNMP využíva na prenos datagramovú službu UDP. Tretí riadok porovnáva relatívnu zložitosť implementácie s ohľadom na softvérové a hardvérové vybavenie mikrokontroléru. V tomto smere sú na implementáciu najjednoduchšie protokoly SNMP a Telnet, nasledované protokolom HTTP a najzložitejším protokolom SSH, ktorý vyžaduje použitie šifrovacích algoritmov.

Z uvedeného vyplýva, že najvýhodnejšími protokolmi na implementáciu riadenia v mikrokontroléri sú protokoly Telnet a SNMP. Z týchto dvoch má vďaka jednoduchšej implementáciu UDP protokolu oproti TCP výhodu SNMP. Protokol SNMP je navyše priamo určený na monitorovanie a riadenie sieťových zariadení a zariadení pripojených do počítačovej siete. Bajtovo orientovaná komunikácia je pre 8-bitový mikrokontrolér rovnako veľkou výhodou, pretože oproti textovej komunikácii nie je nutné rozpoznávať reťazce. Všetky tieto výhody vytvárajú z protokolu SNMP ideálneho kandidáta na implementáciu v mikrokontroléri. Práve z tohoto dôvodu bol

| | Telnet | SSH | HTTP | SNMP |
|---|---|---|---|---|
| Typ komunikácie | bajtová | bajtová | textová | bajtová |
| Transportný protokol | TCP | TCP | TCP | UDP |
| Zložitosť implementácie | malá | veľká | stredne veľká | malá |

**Tabuľka 4 – 1**: Porovnanie vlastností protokolov Telnet, SSH, HTTP a SNMP

protokol SNMP zvolený ako hlavný protokol riadenia mikrokontroléru.

Protokol HTTP je vhodný na tvorbu grafického používateľského rozhrania s využitím webového servera, ktorý bude pomocou tohoto protokolu sprostredkúvať používateľovi grafický výstup riadenia. Používateľ tak má na výber, či použije na riadenie priamo protokol SNMP alebo webovú stránku s grafickým používateľským rozhraním.

## 4.3 Riadenie mikrokontroléru protokolom SNMP

Protokol SNMP je vďaka svojej jednoduchosti a rozšírenosti veľmi vhodný na riadenie nielen sieťových zariadení ale aj iných zariadení pripojených do IP siete. Spracovanie SNMP správy nevyžaduje veľké množstvo systémových zdrojov a preto je vhodné aj na implementáciu v 8-bitovom mikrokontroléri. Softvér vykonávajúci úlohu SNMP agenta v mikrokontroléri prijíma SNMP správy, overuje ich štruktúru a následne spracuje požadovanú operáciu, ktorá je buď typu `get` alebo `set` nad objektom v MIB.

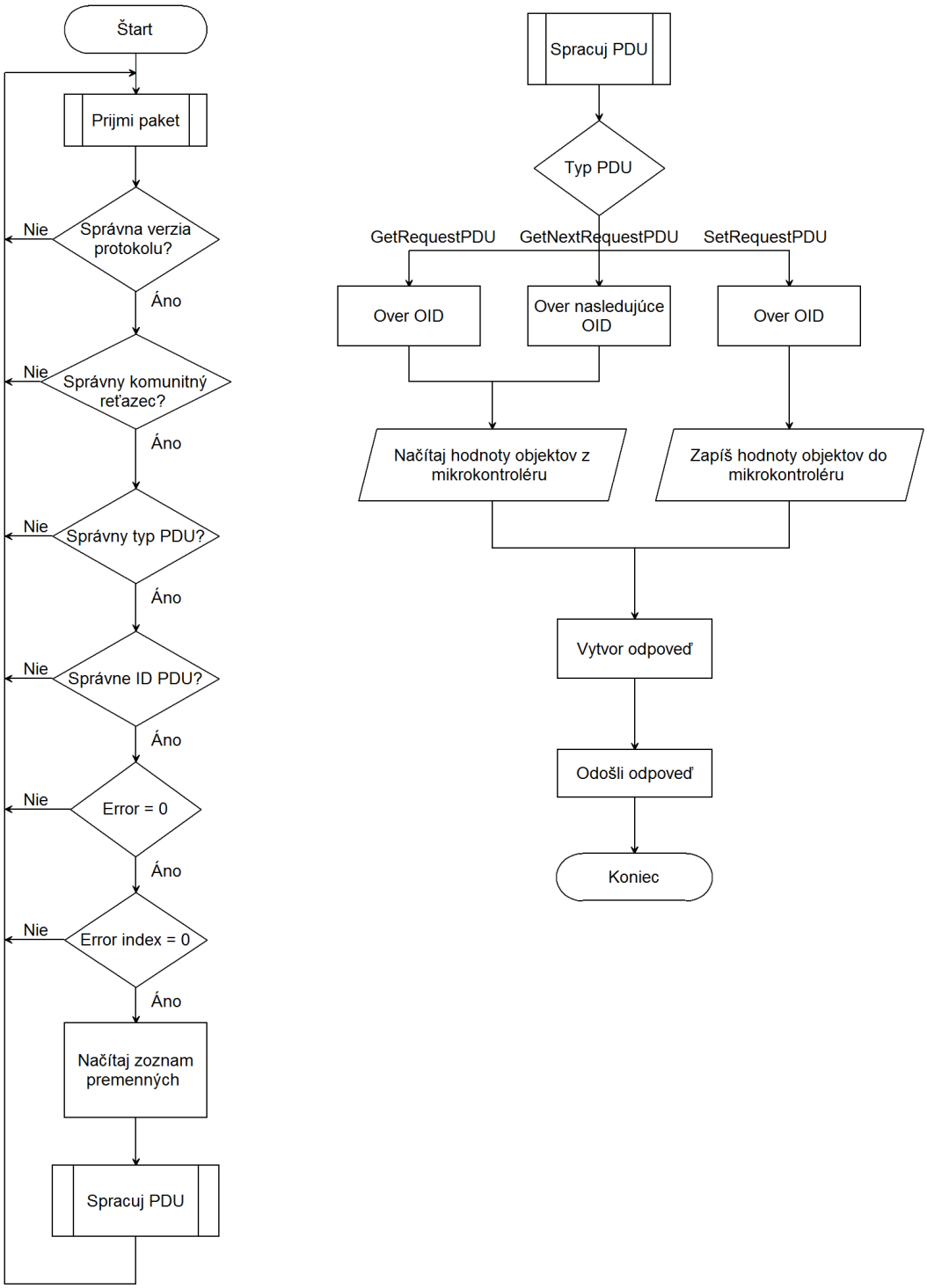
Algoritmus riadenia (obr. 4 – 1) pozostáva z nekonečného cyklu, ktorý sa pri každom prechode pokúsi spracovať jednu SNMP správu. Po prijatí SNMP paketu sa najskôr overí verzia protokolu. Podporovaná je iba SNMP verzie 1. Pri inej verzii sa spracovanie SNMP správy ukončí a začne sa spracovávanie ďalšej správy. Po overení verzie sa overuje správnosť komunitného reťazca, ktorý v protokole SNMP implementuje vrstvu zabezpečenia. Ak je komunitný reťazec v SNMP pakete nesprávny,

pokračuje sa spracovaním ďalšieho paketu. Ďalej nasleduje overenie typu PDU. Podporované typy, ktoré sa ďalej spracúvajú sú GetRequestPDU, GetNextRequestPDU a SetRequestPDU, pri iných typoch sa SNMP paket zahodí a pokračuje sa spracovaním ďalšieho paketu. Po overení typu paketu nasleduje kontrola identifikátora správy, chybového hlásenia a indexu chyby. Pri akýchkoľvek nesprávnych údajoch týchto parametrov program pokračuje spracovaním ďalšej správy. Po overení všetkých potrebných parametrov nasleduje načítanie zoznamu premenných. Nakoniec je spracované samotné PDU.

Spracovanie PDU začína vetvením podľa typu PDU. V prípade `GetRequestPDU` a `SetRequestPDU` nasleduje overenie identifikátorov objektov (OID). V prípade chybného identifikátora je index premennej nastavený do `error-index` v PDU a odošle sa odpoveď `GetResponsePDU` s príslušným chybovým hlásením. Pre `GetNextRequestPDU` je vyhľadaný najbližší identifikátor objektu v zozname objektov. V ďalšom kroku sa buď získajú hodnoty od mikrokontroléru a naplnia sa nimi príslušné premenné v PDU v prípade `GetRequestPDU` a `GetNextRequestPDU` alebo sa zapíšu hodnoty prijaté v SNMP správe do mikrokontroléru v prípade `SetRequestPDU`. Nakoniec sa vytvorí `GetResponsePDU` požadovaného tvaru, ktorá je následne odoslaná ako odpoveď na prichádzajúcu SNMP požiadavku.

## 4.4 Riadenie mikrokontroléru protokolom HTTP

Protokol HTTP je vhodné využiť na ovládanie mikrokontroléru na vyššom stupni, čo predstavuje grafické používateľské rozhranie. Aj keď by bolo možné implementovať riadenie pomocou HTTP protokolu priamo do softvéru mikrokontroléru vyžadovalo by to veľa programovej pamäte a grafické používateľské rozhranie by nemohlo mať takú bohatú funkcionalitu ako v prípade externého riešenia. Preto bolo zvolené riadenie HTTP protokolom pomocou webovej stránky uloženej na vzdialenom webovom serveri, ktorý dokáže adresovať riadený mikrokontrolér jeho IP adresou.




**Obrázok 4 – 1**: Vývojový diagram algoritmu spracovania SNMP správy

Pre grafické používateľské rozhranie je výhodné použiť technológie HTML, CSS a PHP. HTML spolu s CSS budú slúžiť ako prezentačná vrstva PHP riešenia. Jazyk PHP je vhodný najmä preto, že obsahuje knižnice pre prácu so SNMP protokolom.

# 5 Implementácia riadenia mikrokontrolérov

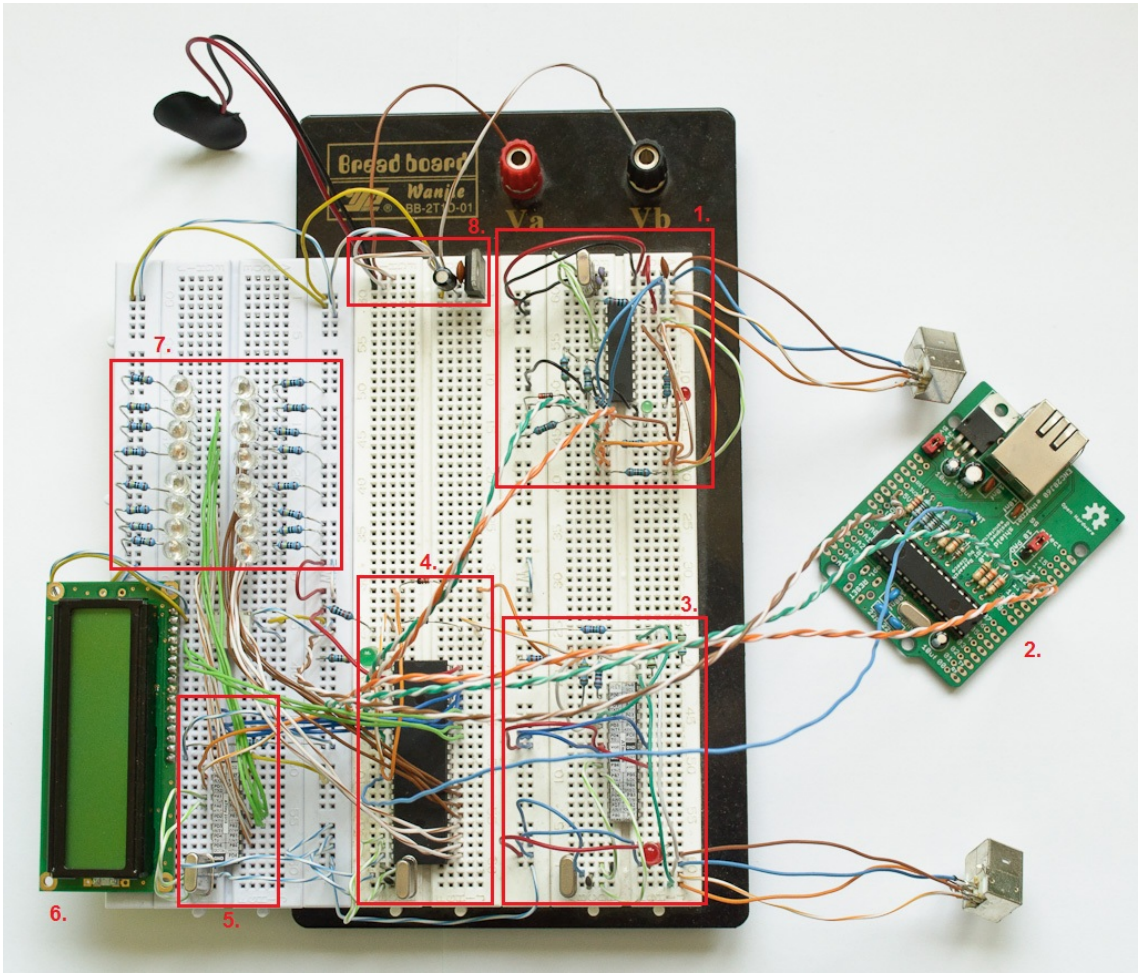
## 5.1 Popis vývojového prostredia

Pre účely tejto práce a vývoj aplikácie pre mikrokontroléry ATmega162 a ATtiny2313 bolo navrhnuté a skonštruované vývojové prostredie (obr. 5 – 1) pozostávajúce z nasledujúcich častí:

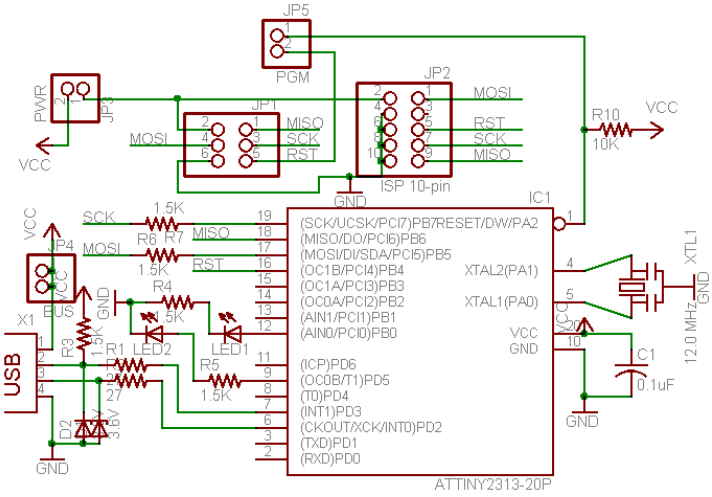
1. USB programátor mikrokontrolérov rady AVR
2. Modul pre pripojenie mikrokontroléru do siete Ethernet s integrovaným obvodom ENC28J60
3. Virtuálny USB sériový port
4. Mikrokontrolér ATmega162
5. Mikrokontrolér ATtiny2313
6. Diagnostický LCD displej
7. LED diódy zobrazujúce hodnoty vývodov na portoch mikrokontrolérov
8. Stabilizovaný napájací zdroj

### 5.1.1 Programátor mikrokontrolérov rady AVR

USB ISP (In System Programming) programátor mikrokontrolérov rady AVR slúži na programovanie flash a EEPROM pamätí mikrokontrolérov ako aj na nastavovanie bitov poistiek (fuses). Schéma zapojenia (obr. 5 – 2) je prevzatá z internetovej stránky `http://www.ladyada.net/make/usbtinyisp/`. Zapojenie obsahuje mikrokontrolér ATtiny2313, 12MHz kryštál na presné časovanie USB zbernice, dvojicu zenerových diód na zníženie napätia USB zbernice na 3,3V, LED diódy na signalizáciu napájania a programovania a niekoľko ďalších pasívnych súčiastok.




**Obrázok 5 – 1**: Vývojové prostredie



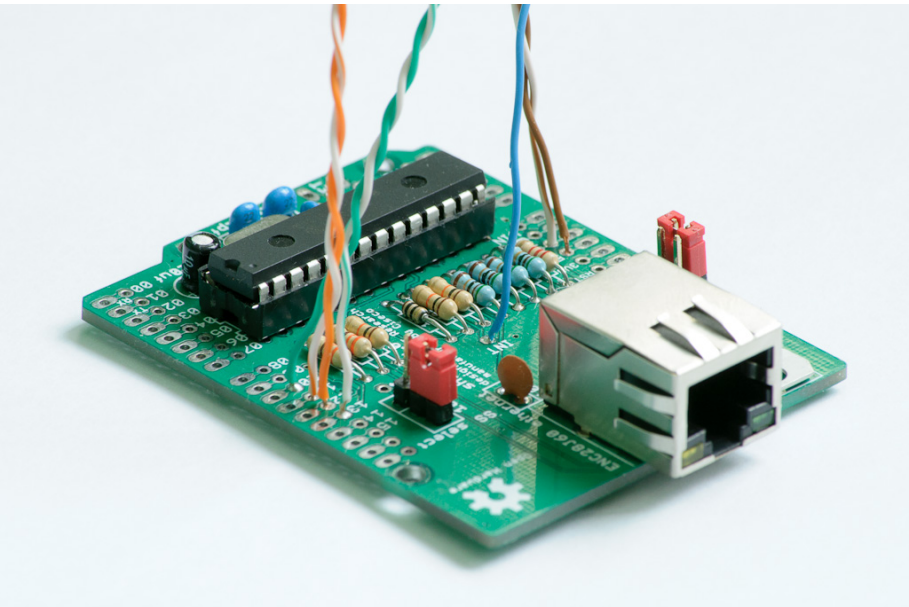

**Obrázok 5 – 2**: Schéma zapojenia USB programátora

Pomocou ovládača (driver) a programu avrdude je možné v operačnom systéme MS Windows aj Linux programovať väčšinu dnes vyrábaných mikrokontrolérov rady AVR.

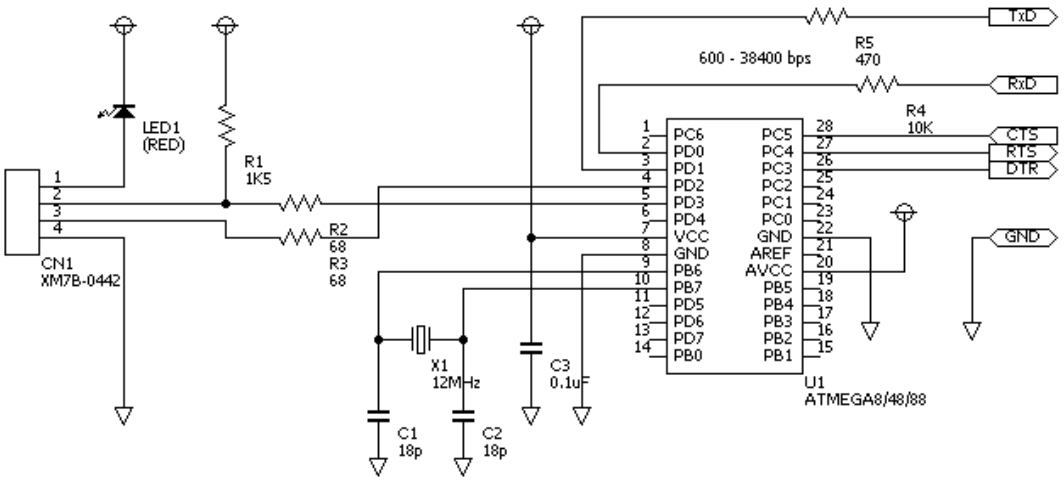
### 5.1.2 Modul pre pripojenie mikrokontroléru do siete Ethernet

Ako modul pre pripojenie mikrokontroléru (obr. 5 – 3) bol použitý Arduino Ethernet Shield od firmy Ciseco. Ide o stavebnicu, ktorá sa dodáva vo forme plošného spoja a súčiastok. Tento plošný spoj je potrebné osadiť súčiastkami a jednotlivé súčiastky prispájkovať. Modul obsahuje integrovaný obvod ENC28J60, kryštál 25MHz, konektor RJ-45, ktorý obsahuje oddeľovacie transformátory, stabilizátor napätia LD33CV a niekoľko ďalších pasívnych súčiastok ako sú kondenzátory a rezistory potrebné pre správnu činnosť zariadenia.

Prepojenie s mikrokontrolérom je realizované cez štyri vodiče sériovej zbernice SPI (MISO, MOSI, SCK a SS) a jeden vodič prerušenia.



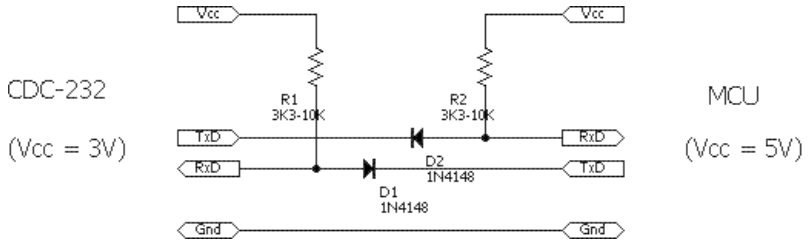
**Obrázok 5 – 3**: Modul pre pripojenie mikrokontroléru do siete Ethernet




**Obrázok 5 – 4**: Schéma zapojenia USB virtuálneho sériového portu

### 5.1.3 Virtuálny sériový port

Keďže v dnešnej dobe sa už sériové rozhranie RS-232 na väčšine nových matičných dosiek počítačov nenachádza, bolo potrebné pre účely ladenia aplikácie vytvoriť virtuálny sériový port. Toto riešenie virtuálneho sériového portu pochádza z internetovej stránky `http://www.recursion.jp/avrcdc/cdc-232.html` a pomocou mikrokontroléru ATmega8 a ovládača pre operačné systémy MS Windows a Linux vytvára virtuálny sériový port. Takto bolo možné sledovať sériovú komunikáciu medzi mikrokontrolérmi alebo im poslať akúkoľvek správu priamo z počítača. Schéma zapojenia (obr. 5 – 4) obsahuje mikrokontrolér ATmega8, 12MHz kryštál pre presné časovanie USB zbernice, LED diódu, ktorá slúžia ako signalizácia napájania a zároveň znižuje napätie na 3,3V, ktorými je mikrokontrolér napájaný. Okrem toho schéma obsahuje niekoľko rezistorov a kondenzátorov. Pre prepojenie s mikrokontrolérom pracujúcim s 5V napájaním je potrebná zmena napäťových úrovní. Na tento účel bola použitá dvojica diód a odporov, ktorých schéma je na obrázku 5 – 5.



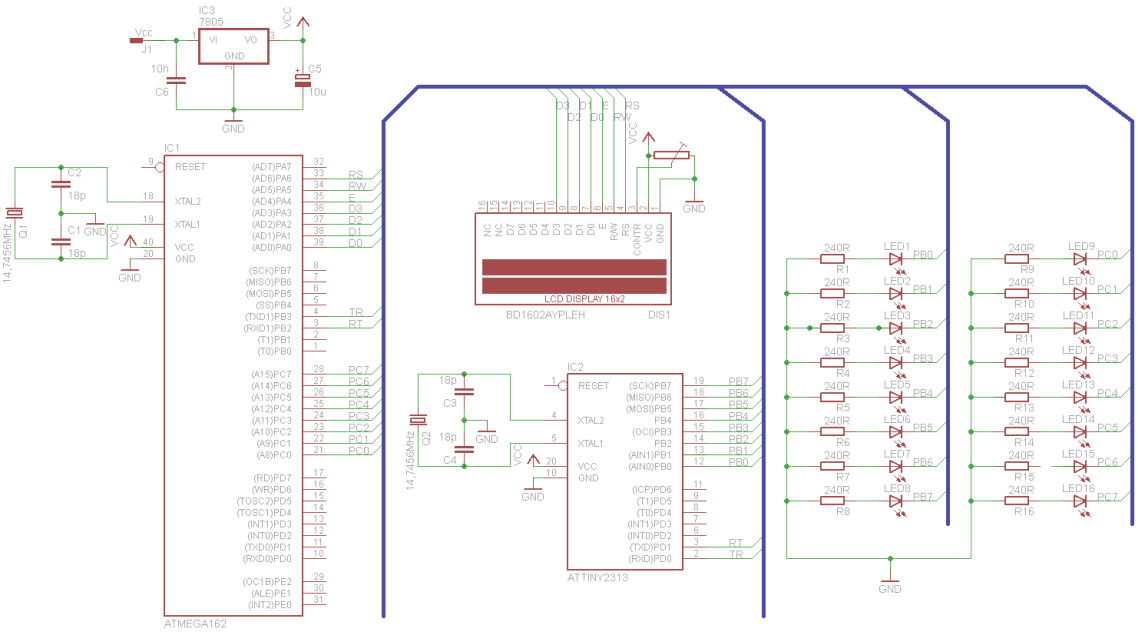

**Obrázok 5 – 5**: Schéma zapojenia prevodníku napäťových úrovní

### 5.1.4 Mikrokontrolér ATmega162

Veľká časť tejto práce je zameraná na implementáciu SNMP protokolu do mikrokontroléru ATmega162, ktorého schéma sa nachádza na obrázku 5 – 6. Vďaka svojej veľkej flash pamäti a dostatočne veľkej SRAM pamäti umožňuje implementovať veľké množstvo rozsiahleho aplikačného kódu. Jeho časovanie je vo vývojovom prostredí realizované kryštálom s frekvenciou 14,7456MHz, ktorý je veľmi vhodný pri používaní sériového rozhrania USART, pretože nespôsobuje žiadnu chybovosť vplyvom nepresného časovania. K mikrokontroléru je pripojená diagnostická LED dióda, ktorá signalizuje prijatie paketu od integrovaného obvodu ENC28J60. Na tento obvod je mikrokontrolér ATmega162 pripojený štyrmi vodičmi sériového rozhrania SPI a jedným vodičom prerušenia. Pripojenie diagnostického LCD displeja je realizované štvorvodičovou údajovou zbernicou a troma riadiacimi signálmi pripojenými na port A. Prepojenie s mikrokontrolérom ATtiny2313 zabezpečuje dvojica vodičov RX a TX sériového rozhrania USART1. Druhé sériové rozhranie USART0 je pripojené dvojicou vodičov RX a TX na virtuálny sériový port a umožňuje tak výpis údajov priamo do sériovej konzoly v počítači. Na port C je pripojených osem LED diód, ktoré zobrazujú invertované hodnoty na vývodoch portu.

### 5.1.5 Mikrokontrolér ATtiny2313

Mikrokontrolér ATtiny2313 je podobne ako ATmega162 časovaný externým kryštálom 14,7456MHz pre bezporuchovú prácu sériového rozhrania USART, pomocou




**Obrázok 5 – 6**: Schéma zapojenia mikrokontrolérov ATmega162 a ATtiny2313

ktorého sú tieto dva mikrokontroléry prepojené. Okrem toho je tento mikrokontrolér pripojený na osem LED diód, ktoré zobrazujú invertované hodnoty na vývodoch portu B.

### 5.1.6 Zvyšné časti vývojového prostredia

Kým USB programátor a virtuálny sériový port sú napájané priamo zo zbernice USB, zvyšok obvodu je pre pomerne veľký odber prúdu hlavne LED diód napájaný z externého transformátora s použitím stabilizátora napätia L7805CV a dvojice kondenzátorov na filtráciu vstupného a výstupného napätia.

Vývojové prostredie tiež obsahuje LCD displej typu BD1602AYPLEH, ktorý obsahuje 2 riadky po 16 znakov. Displej slúži na zobrazovanie rôznych diagnostických hodnôt a správ a je veľmi užitočný pri ladení programu. S mikrokontrolérom je prepojený štvorvodičovou údajovou zbernicou a trojicou riadiacich signálov. Odporový trimer $10k\Omega$ slúži na reguláciu jasu displeja.

Ku každému z mikrokontrolérov ATmega162 a ATmega2313 je pripojených osem LED diód, ktoré zobrazujú invertované hodnoty na týchto portoch. Ak je vývod portu v logickej nule LED dióda svieti. Každá LED dióda je pripojená na ochranný odpor 240Ω.

## 5.2 Implementácia riadenia mikrokontroléru ATmega162

Hlavnou časťou tejto práce je riadenie mikrokontroléru ATmega162. Tento mikrokontrolér obsahuje päť vstupno výstupných portov: 8-bitové porty `PORTA`, `PORTB`, `PORTC`, `PORTD` a 3-bitový `PORTE`. Softvér mikrokontroléru umožňuje vzdialené riadenie portov A, C a D. Porty B a E nie sú je prístupné, pretože sa využívajú na komunikáciu s Ethernetovým modulom a vzdialeným mikrokontrolérom prostredníctvom sériovej zbernice USART.

Každý port mikrokontroléru obsahuje trojicu registrov `PORT`, `DDR` a `PIN`, ktorých funkcia bola bližšie popísaná v analytickej časti. Riadenie protokolom SNMP umožňuje čítať hodnoty všetkých týchto registrov pre každý port a zápis hodnôt do registrov `PORT` a `DDR`.

Protokol SNMP vyžaduje pre adresovanie riadených objektov ich jednoznačné identifikátory (OID) uložené v báze riadených údajov (MIB). MIB je celosvetová databáza udržiavaná a organizovaná spoločnosťou IANA (Internet Assigned Numbers Authority). Pre pridanie vlastnej MIB do tejto databázy je potrebné vlastný MIB zaregistrovať u tejto spoločnosti. Pre účely tejto práce bola využitá experimentálna vetva globálnej MIB, ktorá sa využíva pri prototypovaní nových zariadení. Táto vetva má adresu `1.3.6.1.3` a touto adresou budú začínať identifikátory všetkých objektov použitých v tejto práci.

Samotná MIB pre mikrokontrolér ATmega162 (zdrojový kód 2) obsahuje import vetvy experimental a typu `Gauge` z RFC1155-SMI a typu `OBJECT-TYPE` z RFC-1212. Ďalej nasleduje pripojenie identifikátora objektu atmega162 k vetve experi-

mental ako položka 1. Za touto položkou je pripojená trojica portov A, C a D a každý port má pripojenú trojicu registrov `PORT`, `DDR` a `PIN`, ktoré sú typu `Gauge` čo je celé nezáporné číslo v hodnote od 0 do 255.

**Zdrojový kód 2: Báza informácií (MIB) pre ATmega162**

```
1  ATMEGA162-MIB DEFINITIONS ::= BEGIN
2
3  IMPORTS
4          experimental, Gauge
5              FROM RFC1155-SMI
6          OBJECT-TYPE
7              FROM RFC-1212;
8
9  atmega162      OBJECT IDENTIFIER ::= { experimental 1 }
10
11
12 remotePortA      OBJECT IDENTIFIER ::= { atmega162 1 }
13
14 remotePortC   OBJECT IDENTIFIER ::= { atmega162 2 }
15
16 remotePortD        OBJECT IDENTIFIER ::= { atmega162 3 }
17
18 portA OBJECT-TYPE
19     SYNTAX  Gauge (0..255)
20     ACCESS  read-write
21     STATUS  mandatory
22     DESCRIPTION
23             "AVR ATmega162 PORTA value."
24     ::= { remotePortA 1 }
25
26 ddrA OBJECT-TYPE
27     SYNTAX  Gauge (0..255)
28     ACCESS  read-write
29     STATUS  mandatory
30     DESCRIPTION
31       "AVR ATmega162 DDRA value."
32     ::= { remotePortA 2 }
33
34 pinA OBJECT-TYPE
35     SYNTAX  Gauge (0..255)
36     ACCESS  read-only
37     STATUS  mandatory
38     DESCRIPTION
39             "AVR ATmega162 PINA value."
40     ::= { remotePortA 3 }
41
42 portC OBJECT-TYPE
43     SYNTAX  Gauge (0..255)
44     ACCESS  read-write
45     STATUS  mandatory
```

```
46      DESCRIPTION
47              "AVR ATmega162 PORTC value."
48      ::= { remotePortC 1 }
49
50  ddrC OBJECT-TYPE
51      SYNTAX  Gauge (0..255)
52      ACCESS  read-write
53      STATUS  mandatory
54      DESCRIPTION
55        "AVR ATmega162 DDRC value."
56      ::= { remotePortC 2 }
57
58  pinC OBJECT-TYPE
59      SYNTAX  Gauge (0..255)
60      ACCESS  read-only
61      STATUS  mandatory
62      DESCRIPTION
63              "AVR ATmega162 PINC value."
64      ::= { remotePortC 3 }
65
66  portD OBJECT-TYPE
67      SYNTAX  Gauge (0..255)
68      ACCESS  read-write
69      STATUS  mandatory
70      DESCRIPTION
71              "AVR ATmega162 PORTD value."
72      ::= { remotePortD 1 }
73
74  ddrD OBJECT-TYPE
75      SYNTAX  Gauge (0..255)
76      ACCESS  read-write
77      STATUS  mandatory
78      DESCRIPTION
79        "AVR ATmega162 DDRD value."
80      ::= { remotePortD 2 }
81
82  pinD OBJECT-TYPE
83      SYNTAX  Gauge (0..255)
84      ACCESS  read-only
85      STATUS  mandatory
86      DESCRIPTION
87              "AVR ATmega162 PIND value."
88      ::= { remotePortD 3 }
89
90  END
```

Identifikátory všetkých riadených objektov mikrokontroléru ATmget162 sa nachádzajú v tabuľke 5 – 1. Pomocou týchto identifikátorov je možné adresovať objekty v SNMP správe a daným registrom priradiť hodnoty alebo z nich hodnoty získať.

| Názov objektu | Identifikátor objektu |
|---|---|
| PORTA | 1.3.6.1.3.1.1.1.0 |
| DDRA | 1.3.6.1.3.1.1.2.0 |
| PINA | 1.3.6.1.3.1.1.3.0 |
| PORTC | 1.3.6.1.3.1.2.1.0 |
| DDRC | 1.3.6.1.3.1.2.2.0 |
| PINC | 1.3.6.1.3.1.2.3.0 |
| PORTD | 1.3.6.1.3.1.3.1.0 |
| DDRD | 1.3.6.1.3.1.3.2.0 |
| PIND | 1.3.6.1.3.1.3.3.0 |

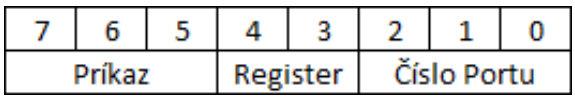
**Tabuľka 5 – 1**: Identifikátory riadených objektov pre ATmega162

## 5.3 Implementácia riadenia mikrokontroléru ATtiny2313

Mikrokontrolér ATtiny2313 je malý a veľmi výkonný mikrokontrolér. Jeho nevýhodou je malé množstvo pamäte SRAM (128 bajtov) a rovnako malá programová pamäť flash (2kB).

Z tohoto dôvodu nie je možné do tohoto mikrokontroléru priamo implementovať pripojenie do siete Ethernet pomocou Ethernetového modulu ani podporu SNMP protokolu. Tento nedostatok je riešený vzdialeným riadením mikrokontroléru ATtiny2313 prostredníctvom už implementovanej podpory SNMP v mikrokontroléri ATmgea162 s využitím jednoduchého sériového komunikačného protokolu. Vďaka tomuto riešeniu je možné k mikrokontroléru ATmgea162 pripojiť akýkoľvek iný mikrokontrolér, ktorý je vybavený sériovou zbernicou USART, a ktorý obsahuje implementáciu sériového komunikačného protokolu popísaného nižšie.

ATtiny2313 obsahuje dva vstupno výstupné 8-bitové porty `PORTB` a `PORTD`. Keďže port D sa používa na časovanie a komunikáciu prostredníctvom sériovej zbernice, je pre riadenie mikrokontroléru protokolom SNMP sprístupnený iba port B.




**Obrázok 5 – 7**: Formát správy sériového protokolu

| | 4 | 3 |
|---|---|---|
| PORT | 1 | 0 |
| DDR | 0 | 1 |
| PIN | 0 | 0 |

**Tabuľka 5 – 2**: Typ registra portu v sériovom protokole

Implementácia sériového protokolu sa nachádza v súbore `usart_comm.c`. Ide o jednoduchý bajtovo orientovaný protokol, ktorý umožňuje získavanie a ukladanie hodnôt registrov portov prostredníctvom sériovej zbernice USART.

Funkcie `usart_init`, `usart_send` a `usart_recv` slúžia na prácu s rozhraním USART. Prvá funkcia nastavuje prenosovú rýchlosť a povoľuje použitie USARTu. Druhá a tretia funkcia slúžia na príjem a odoslanie jedného bajtu.

Všetky funkcie sériového protokolu sú kódované do jedného bajtu (obr. 5 – 7). Významovo najmenšie tri bity obsahujú číslo portu, nad ktorým sa má príkaz vykonať. Bity 3 a 4 definujú typ registra podľa tabuľky 5 – 2 a významovo najväčšie tri bity reprezentujú príkaz podľa tabuľky 5 – 3. Príkaz `GetCount` slúži na získanie počtu dostupných portov mikrokontroléru. Príkazy `GetValue` a `SetValue` slúžia na získanie ale nastavenie hodnoty príslušného portu a registra.

Pomocou takto definovaného protokolu je možné riadiť porty akéhokoľvek mikrokontroléru, ktorý podporuje rozhranie USART a implementuje funkcie definované v sériovom protokole. Nemusí sa pritom nutne jednať o mikrokontrolér AVR od firmy ATMEL.

| | 7 | 6 | 5 |
|---|---|---|---|
| GetCount | 0 | 0 | 0 |
| GetValue | 0 | 0 | 1 |
| SetValue | 0 | 1 | 0 |

**Tabuľka 5 – 3**: Príkazy sériového protokolu

## 5.4 Implementácia pripojenia do IP siete

Ako už bolo spomenuté v predchádzajúcich častiach, mikrokontrolér je do IP siete pripojený prostredníctvom Ethernetového modulu založeného na integrovanom obvode ENC28J60, ktorý s mikrokontrolérom komunikuje po sériovej zbernici SPI. Integrovaný obvod ENC28J60 hardvérovo implementuje fyzickú a spojovú vrstvu OSI modelu a vyžaduje, aby funkcionalita vyšších vrstiev bola implementovaná softvérovo v mikrokontroléri.

Pre pripojenie mikrokontroléru do IP siete je tak potrebné softvérovo implementovať nasledovnú funkcionalitu:

- Modul pre ovládanie integrovaného obvodu ENC28J60 po zbernici SPI.
- Modul protokolu IPv4 – nevyhnutný pre pripojenie do IP siete.
- Modul protokolu ARP – umožňujúci preklad IP adries na MAC adresy.
- Modul protokolu ICMP – umožňujúci testovať dostupnosť zariadenia pomocou nástroja ping.
- Modul protokolu UDP – pre komunikáciu cez bezstavový protokol UDP.
- Modul protokolu TCP – pre komunikácie cez protokol TCP.

Súhrnne sa tieto moduly označujú ako IP zásobník (IP stack) pre integrovaný obvod ENC28J60. Na internete je možné nájsť niekoľko IP zásobníkov s otvoreným zdrojovým kódom vhodných pre riešenie pripojenia mikrokontroléru AVR do IP siete.

V práci bol použitý IP zásobník z webovej stránky `http://tuxgraphics.org/`. Tento IP zásobník obsahuje implementáciu všetkých vyššie uvedených modulov nevyhnutných pre prácu mikrokontroléru v IP sieti.

V nasledujúcej časti bude uvedená základná kostra programu pre pripojenie mikrokontroléru do IP siete s implementáciou protokolov IP, ARP, ICMP a UDP. Kostra (zdrojový kód 3) pozostáva z inicializačnej časti a nekonečného cyklu, ktorý spracováva prijaté údajové pakety.

**Zdrojový kód 3: Základná kostra programu**

```
1  #include <avr/io.h>
2  #include "ip_arp_udp_tcp.h"
3  #include "enc28j60.h"
4  #include "timeout.h"
5  #include "avr_compat.h"
6  #include "net.h"
7
8  static uint8_t mymac[6] = {0x54, 0x55, 0x58, 0x10, 0x00, 0x24};
9  static uint8_t myip[4] = {192, 168, 2, 150};
10
11 #define MYWWWPORT 80
12 #define MYUDPPORT 161
13
14 #define BUFFER_SIZE 450
15 static uint8_t buf[BUFFER_SIZE + 1];
16
17 int main(void){
18   uint16_t plen,dat_p;
19   uint8_t payloadlen = 0;
20
21   enc28j60Init(mymac);
22   enc28j60clkout(2);
23   _delay_loop_1(50);
24
25   enc28j60PhyWrite(PHLCON, 0x476);
26   _delay_loop_1(50);
27
28   init_ip_arp_udp_tcp(mymac, myip, MYWWWPORT);
29
30   while(1){
31     plen = enc28j60PacketReceive(BUFFER_SIZE, buf);
32
33     if(plen == 0)
34       continue;
35
36     if(eth_type_is_arp_and_my_ip(buf, plen)) {
37       make_arp_answer_from_request(buf);
```

```
38        continue;
39      }
40
41      if(eth_type_is_ip_and_my_ip(buf, plen) == 0) {
42        continue;
43      }
44
45      if(buf[IP_PROTO_P] == IP_PROTO_ICMP_V && buf[ICMP_TYPE_P] ==
            ICMP_TYPE_ECHOREQUEST_V) {
46        make_echo_reply_from_request(buf, plen);
47        continue;
48      }
49
50      if(buf[IP_PROTO_P] == IP_PROTO_UDP_V) {
51        payloadlen = buf[UDP_LEN_L_P] - UDP_HEADER_LEN;
52        make_udp_reply_from_request(buf, &buf[UDP_DATA_P], payloadlen,
              MYUDPPORT);
53      }
54    }
55 }
```

Prvých 6 riadkov kódu vkladá potrebné knižnice IP zásobníku. Za knižnicami sa nachádza deklarácia a inicializácia MAC adresy a IP adresy, ktorými sa mikrokontrolér pripája do IP siete. Ďalej nasleduje definícia WWW portu a UDP portu, na ktorých bude mikrokontrolér spracovávať prichádzajúce pakety. Na konci globálnej časti sa nachádza definícia zásobníka pre prichádzajúce pakety veľkosti 451 bajtov, ktorý zároveň limituje maximálnu veľkosť prijímaných a odosielaných paketov. Od riadku 17 začína funkcia `main`, na začiatku ktorej sa nachádzajú deklarácie potrebných premenných. Na riadkoch 21 až 23 nasleduje inicializácia obvodu ENC28J60 s použitím definovanej MAC adresy, zmena výstupného časovania integrovaného obvodu ENC28J60 pre prípady časovania mikrokontroléru priamo týmto obvodom a čakacia slučka dlhá približne 12ms. Na riadku 25 sa nachádza nastavenie správania LED diód na konektore RJ-45, ktoré sú pripojené k integrovaného obvodu ENC28J60 tak, že prvá dióda signalizuje stav prenosovej linky a druhá príjem alebo vysielanie. Po nastavení správania LED diód nasleduje opäť čakacia slučka dlhá 12ms. Na riadku 28 je inicializácia IP zásobníka a všetkých jeho súčastí s nastavením IP adresy, MAC adresy a WWW portu. Po inicializácii IP zásobníka inicializačná časť končí a nasleduje nekonečná slučka, ktorá spracováva prichádzajúce

pakety. Na začiatku slučky (riadok 30) sa program pokúsi prijať do integrovaného obvodu ENC28J60 platný IP paket. Ak nie je pripravený žiadny paket a vrátená dĺžka prijatého paketu `plen` je rovná nule, program pokračuje ďalšou iteráciou cyklu. Na riadku 36 sa nachádza overenie typu paketu. Ak je paket typu ARP a IP adresa cieľa je zhodná z adresou mikrokontroléru vytvorí sa pomocou tohoto paketu ARP odpoveď a pokračuje sa ďalšou iteráciou cyklu. Riadok 41 testuje či paket je typu IP a cieľová adresa je zhodná s adresou mikrokontroléru. V prípade, že to tak nie je paket zahodí a pokračuje ďalšou iteráciou cyklu. Riadok 45 overuje či je paket typu ICMP a či ide o žiadosť o echo (ping). Ak áno, je pomocou paketu vytvorená príslušná odpoveď a program pokračuje ďalšou iteráciou cyklu. Riadok 50 obsahuje test či je paket typu UDP. V prípade UDP paketu sa vypočíta veľkosť UDP časti a následne sa z údajov vytvorí odpoveď, ktorá posiela späť na zdrojovú IP adresu celý obsah prichádzajúceho UDP paketu.

## 5.5 Implementácia protokolu SNMP

Nasledujúca časť práce obsahuje podrobný popis implementácie protokolu SNMP v mikrokontroléri s architektúrou AVR v programovacom jazyku C. Spracovanie SNMP správy pozostáva z niekoľkých častí a je ukončené buď bez odpovede klientovi alebo vygenerovaním a odoslaním SNMP odpovede.

SNMP správa je odosielaná ako údajová časť UDP paketu kódovaná pomocou notácie ASN.1, preto je potrebné pred samotným spracovaním dekódovať správu a overiť jej formát. Keďže UDP je nespoľahlivý protokol, môže pri prenose dôjsť k poškodeniu údajov v pakete. Protokol SNMP túto skutočnosť rieši tak, že na každú SNMP požiadavku očakáva odpoveď. V prípade, že odpoveď nepríde do stanoveného časového limitu, je požiadavka odosielaná znovu sieťovým manažérom. Za overovanie a riešenie chýb je teda primárne zodpovedný sieťový manažér, čím odpadá potreba riešiť túto funkcionalitu v prostredí agenta.

Celá údajová časť UDP paketu sa pri spracovaní nachádza v zásobníku reprezentovanom poľom bajtov. Metóda `snmp_process_request`, ktorá vykonáva celú funkcionalitu SNMP protokolu, má dva parametre. Prvým je smerník na zásobník so správou, ktorá sa má spracovať a druhým je veľkosť údajovej časti SNMP správy, ktorá by mala byť totožná s veľkosťou SNMP správy.

Spracovanie SNMP paketu prebieha v nasledujúcich krokoch:

1. Overenie štruktúry a dĺžky prijatej SNMP správy
2. Spracovanie SNMP správy a získanie alebo uloženie hodnôt prijatých v SNMP správe do alebo z príslušného portu mikrokontroléru
3. Skonštruovanie odpovede na prijatú SNMP správu

V prvej fáze sa overuje štruktúra, úplnosť a dĺžka prijatej SNMP správy. Keďže na kódovanie SNMP správ sa používa notácia ASN.1, je potrebné zároveň dekódovať túto správu do podoby, v ktorej ju môže mikrokontrolér ďalej spracovávať. Každá SNMP správa musí začínať bajtom `30h`, ktorý v ASN.1 kódovaní reprezentuje postupnosť. Ide o postupnosť verzie SNMP protokolu, komunitného reťazca a údajov, ktoré obvykle reprezentujú PDU. Hneď po overení postupnosti nasleduje jej veľkosť, ktorá sa uloží do lokálnej premennej `size`. Okrem samotnej veľkosti postupnosti sa uloží aj pozícia bajtu s veľkosťou v zásobníku so správou, ktorá bude použitá pri vytváraní odpovede na SNMP správu.

Nasleduje overenie verzie protokolu. Keďže mikrokontrolér podporuje iba SNMP verzie 1, ktorá je kódovaná hodnotou `0`, načíta sa pomocou funkcie `read_uint8` nezáporné celé číslo a overí sa či je jeho hodnota `0`. V prípade, že ide o inú verziu protokolu prichádzajúca SNMP správa sa zahodí a funkcia končí s návratovou hodnotou `0`.

Po overení verzie nasleduje overenie komunitného reťazca, ktorý slúži ako bezpečnostný prvok. Na overenie sa použije funkcia `check_commstring`, ktorá najprv

overí, či v SNMP správe nasleduje reťazec reprezentovaný typom `04h`, ďalej overí či zodpovedá dĺžka komunitného reťazca a nakoniec porovná reťazec s lokálnym komunitným reťazcom, ktorý je nastavený na `"private"`, pomocou funkcie `strncmp` z knižnice `stdlib`. Ak je komunitný reťazec nesprávny SNMP správa sa zahodí a funkcia končí návratovou hodnotou `0`.

Nakoniec sa overí štruktúra a dĺžka PDU a vytvorí sa zoznam objektov, ktorých hodnoty sa majú spracovať. Nasledujúci bajt SNMP správy reprezentuje typ PDU. Skontroluje sa či prijatá PDU zodpovedá jednej z nasledujúcich: `GetRequestPDU` (hodnota `A0h`), `GetNextRequestPDU` (hodnota `A1h`) alebo `SetRequestPDU` (hodnota `A3h`). V prípade iného typu je SNMP správa zahodené a funkcia končí návratovou hodnotou `0`.

Ďalším bajtom v poradí je veľkosť samotnej PDU, ktorej pozícia sa rovnako ako pozícia veľkosti SNMP správy uloží do lokálnej premennej pre generovanie odpovede. Štruktúra PDU pozostáva z identifikátora správy, chybového stavu, indexu chyby a zoznamu premenných objektov.

Pri identifikátore správy sa overuje iba typ, ktorý by mal byť celé číslo a dĺžka. Chybový stav aj index chyby sú reprezentované celým číslom a vo všetkých typoch prijímaných správ by mali obsahovať hodnotu `0`. Ak tieto podmienky nie sú splnené, ukončí sa spracovanie správy, správa sa zahodí a funkcia končí sa návratovou hodnotou `0`.

Poslednou fázou overenia je overenie zoznamu premenných, pri ktorom sa zároveň skonštruuje údajová štruktúra typu spájaný zoznam, ktorý obsahuje zoznam premenných obsiahnutých v SNMP správe, a ktorý je možné neskôr spracovať. Pred načítaním zoznamu premenných sa do lokálnej premennej uloží začiatočná pozícia zoznamu premenných, ktorá bude použitá pri vytváraní odpovede.

Na načítanie zoznamu premenných slúži funkcia `read_varbindlist`. Zoznam premenných je postupnosť jednotlivých premenných, pričom každá premenná je repre-

zentovaná ako postupnosť identifikátora objektu a hodnoty objektu. Pri načítaní sa najprv skontroluje či zoznam premenných začína sekvenciou a do lokálnej premennej `size` sa načíta jeho veľkosť. Potom sa v cykle spracovávajú jednotlivé premenné, až kým veľkosť spracovaných premenných nie je rovná alebo väčšia ako veľkosť celého zoznamu premenných. V cykle sa najskôr kontroluje či daná premenná začína postupnosťou a následne sa uloží veľkosť tejto postupnosti. V ďalšom kroku sa vytvára spájaný zoznam tak, že ak aktuálny smerník obsahuje hodnotu `NULL`, vytvorí sa nová položka zoznamu dynamickou alokáciou pamäte prostredníctvom funkcie `malloc` z knižnice `stdlib`. Pokračuje sa získaním identifikátora objektu, ktorý má v ASN.1 notácii typ s hodnotou `06h`. Po overení typu sa načíta jeho veľkosť, ktorá sa uloží do štruktúry reprezentujúcej položku spájaného zoznamu premenných. V ďalšom kroku je alokovaná pamäť pre identifikátor objektu a následne je identifikátor bajt po bajte skopírovaný do alokovanej pamäte. Týmto končí načítanie identifikátora objektu a nasleduje načítanie hodnoty objektu. Typ a veľkosť hodnoty objektu sú uložené do štruktúry. Potom je alokovaná pamäť pre hodnotu a celá hodnota je skopírovaná do tejto pamäte. Nakoniec sa nastaví smerník na nasledujúcu položku zoznamu a cyklus sa opäť opakuje.

Po spracovaní zoznamu premenných sa overí, či spracovaná dĺžka SNMP správy zodpovedá dĺžke prijatých údajov v UDP datagrame. V prípade nerovnosti funkcia končí s návratovou hodnotou `0` a SNMP paket je zahodený.

Po overení štruktúry a dĺžky SNMP správy a skonštruovaní zoznamu premenných nasleduje samotné spracovanie PDU časti správy, kde sa podľa typu PDU buď získajú hodnoty od mikrokontroléru alebo sa príslušné hodnoty z premenných v SNMP správe nastavia, ako hodnoty portov mikrokontroléru.

Spracovanie PDU začína viaccestným vetvením podľa typu PDU. `GetRequestPDU` a `GetNextRequestPDU`, ktoré slúžia na získanie údajov z mikrokontroléru, sú spracované spoločne. Rozdiel v spracovaní týchto dvoch typov PDU je iba v odlišnom overení správnosti identifikátora objektu. Zatiaľčo pri `GetRequestPDU` sa používa

funkcia `check_oid`, pri `GetNextRequestPDU` je použitá funkcia overujúca lexikografického nasledovníka identifikátora s názvom `check_next_oid`.

Funkcia `check_oid` overuje či v MIB mikrokontroléru existuje daný identifikátor objektu. Pre každú premennú v zozname premenných získanom pri overovaní SNMP správy je overený identifikátor objektu nasledovne. Najprv sa overí dĺžka identifikátora, keďže všetky platné identifikátory majú rovnakú dĺžku. Potom sa bajt po bajte porovná identifikátor v zozname premenných s globálnym identifikátorom. V ďalšom kroku sa overuje číslo portu, ktoré musí byť menšie alebo rovné počtu portov a zároveň väčšie ako 0. Ďalej sa overí typ registra portu, ktorý musí byť v rozsahu 1 až 3 v prípade `GetRequestPDU` a 1 až 2 v prípade `SetRequestPDU`. Nakoniec sa overí či posledný bajt identifikátora objektu je 0. V prípade akejkoľvek chyby funkcia vracia index premennej, ktorá chybu spôsobila a `0` v prípade, že všetky identifikátory sú platné.

Funkcia `check_next_oid` je oproti funkcii `check_oid` výrazne zložitejšia. Jej úlohou je overiť či zadaný identifikátor má v MIB lexikografického nasledovníka, a keď ho nájde, upraviť pôvodný identifikátor objektu na identifikátor nasledovníka, ktorý bude neskôr použitý na získanie hodnoty. Najprv sa overí, či identifikátor objektu zo zoznamu premenných začína rovnako ako globálny identifikátor objektu. Nasleduje vetvenie podľa dĺžky identifikátora na menší alebo rovný globálnemu identifikátoru, rovný dĺžke o 1 väčšej ako je globálny identifikátor a dlhší identifikátor. V prvom prípade kde je identifikátor menší alebo rovný dĺžke globálneho identifikátora vždy nasleduje prvý port a prvý typ registra portu a tento identifikátor je vytvorený namiesto pôvodného. V druhom prípade sa testujú dva prípady. Či je hodnota bajtu za dĺžkou globálneho identifikátora 0 a opäť nasleduje identifikátor prvého portu a prvého typu registra portu alebo nenulové číslo, pri ktorom sa vráti identifikátor nasledovného čísla portu a prvého typu registra. Posledný tretí prípad je najkomplexnejší, pretože zahŕňa hľadanie nasledujúcich identifikátorov pre identifikátory dlhšie ako predošlý prípad. Najprv sa overí, či na bajte identifikujúcom číslo

portu je hodnota menšia alebo rovná ako počet portov. Ďalej sa overuje, či typ registra portu v prijatom identifikátore objektu je v platnom rozsahu registrov portov. V prípade akejkoľvek chyby vráti funkcia index premennej, ktorá chybu spôsobila inak v prípade, že sú všetky premenné v poriadku vráti hodnotu `0`.

Ak v predošlom spracovaní došlo k chybe identifikátora, vytvorí sa odpoveď typu `GetResponsePDU`, ktorá má nastavenú chybu typu `NO_SUCH_NAME` a index chyby je index premennej, ktorá chybu spôsobila. Zvyšok PDU je rovnaký aký mala prichádzajúca správa.

Po overení všetkých premenných v zozname je použitá funkcia `get_oid_values`, ktorá naplní zoznam premenných hodnotami registrov portov mikrokontroléru nasledovne. V cykle pre každú premennú zvlášť sa zistí typ portu podľa identifikátora objektu OID a získa sa príslušná hodnota registra portu. Potom sa nastaví typ premennej na nezáporné celé číslo a jeho hodnota sa vloží do príslušnej premennej.

Spracovanie `GetRequestPDU` a `GetNextRequestPDU` končí zapísaním nového zoznamu premenných do zásobníka s prichádzajúcou správou tak, že pôvodný zoznam premenných je nahradený získanými hodnotami a týmto vznikne odpoveď typu `GetResponsePDU`. Na zápis nového zoznamu premenných slúži metóda s názvom `write_varbindlist`, najprv do zásobníka uloží hodnotu `0x30h` reprezentujúcu sekvenciu a následne získa veľkosť nového zoznamu premenných volaním funkcie `get_varbind_size`. Táto funkcia prechodom cez zoznam premenných a zistením veľkosti jednotlivých premenných zistí dĺžku nového zoznamu premenných. Získaná veľkosť je následne zapísaná do zásobníka. Ďalej funkcia `write_varbindlist` pokračuje prechodom cez zoznam premenných zápisom všetkých sekvencií obsahujúcich identifikátor objektu OID a hodnotou daného objektu.

Spracovanie `SetRequestPDU` prebieha podobne ako `GetRequestPDU` tým, že najskôr pomocou funkcie `check_oid` overia všetky identifikátory objektov s tým rozdielom, že pri `SetRequestPDU` nie je možné adresovať typ registra portu `PIN`,

ktorý slúži len pre vstup hodnôt. Spracovanie ďalej pokračuje overením typov a veľkosťou hodnôt v prichádzajúcom zozname premenných. Na tento účel slúži funkcia `check_oid_values`, ktorá pre každú zo zoznamu premenných overí či sa jedná o nezáporné celé číslo a či hodnota nie je väčšia ako 255. V prípade chyby je odoslaná `GetResponsePDU` s rovnakým obsahom ako prijatá správa s indexom chyby nastaveným na číslo premennej zo zoznamu premenných, ktorá chybu spôsobila a typom chyby `BadValue`.

V ďalšom kroku sú nastavené hodnoty portov pomocou funkcie `set_oid_values`, ktorá podobne ako `get_oid_values` zapíše príslušnú hodnotu do príslušného registra portu. Nakoniec sa vytvorí `GetResponsePDU` zapísaním zoznamu premenných funkciou `write_varbindlist` podobne ako pri `GetRequestPDU` a spracovanie `SetRequestPDU` sa končí.

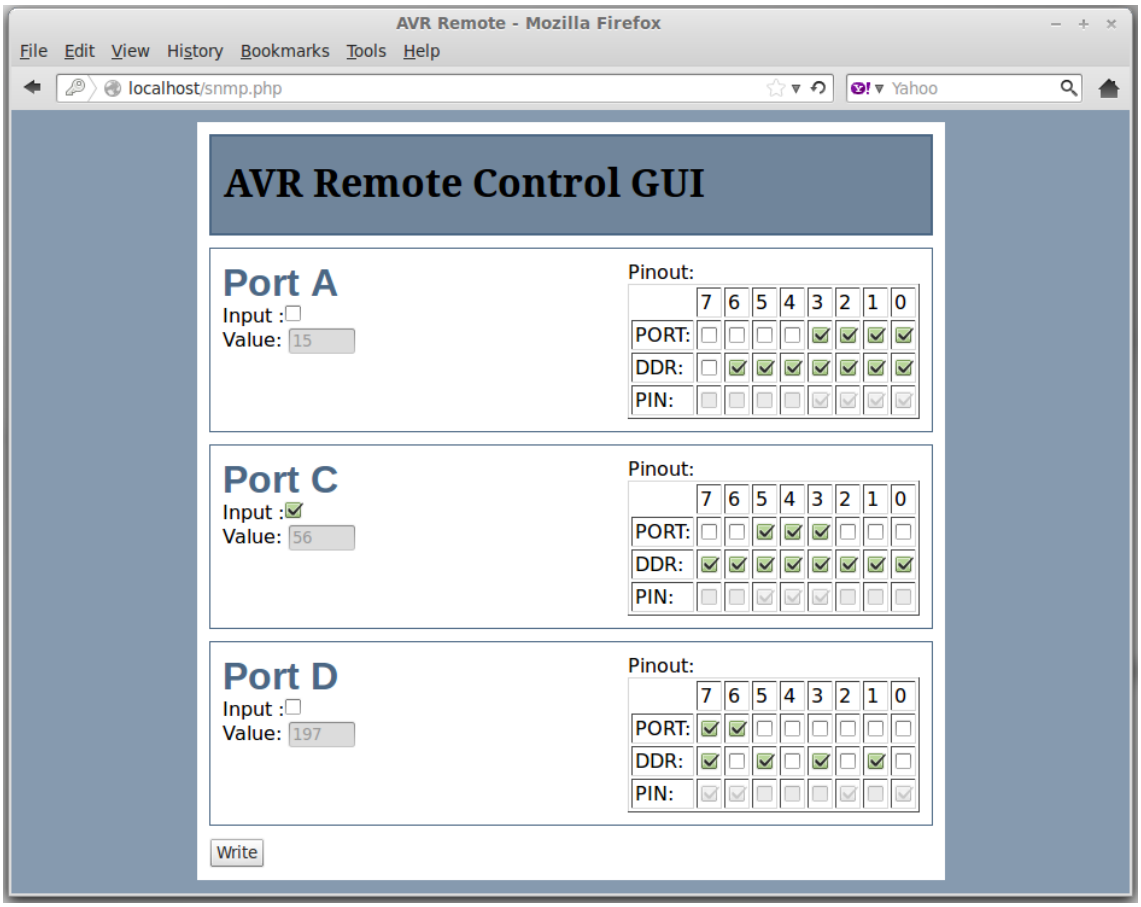
Po spracovaní PDU funkcia `free_varbindlist` uvoľní pamäť, v ktorej boli uchovávané údaje o zozname premenných.

## 5.6 Implementácia riadenia protokolom HTTP

Keďže jednou z úloh tejto práce bola implementácia riadenia mikrokontroléru prostredníctvom protokolu HTTP, bol tento protokol použitý pre vytvorenie grafického používateľského rozhrania v podobe webovej stránky. Ako základ slúži webový server, ktorý pomocou protokolu HTTP komunikuje s klientom. Webová stránka s grafickým používateľským rozhraním je založená na technológiách HTML, PHP, CSS a JavaScript. Samotné riadenie však naďalej prebieha prostredníctvom protokolu SNMP, ktorého metódy sú volané z interpretéra jazyka PHP.

Grafické používateľské rozhranie (obr. 5 – 8) zobrazuje hodnoty registrov jednotlivých portov mikrokontroléru a umožňuje nastavenie ich hodnôt.

Používateľské rozhranie používa jazyk PHP a jeho rozšírenie, ktoré umožňuje posie-




**Obrázok 5 – 8**: Grafické používateľské rozhranie riadenia mikrokontroléru

lať a prijímať SNMP správy priamo v prostredí PHP. Funkcie pre prácu s protokolom SNMP sú oddelené od zobrazovacej časti a nachádzajú sa v súbore `snmp_util.php`. Ide o funkcie `snmp_load`, `snmp_get` a `snmp_set`. Funkcia `snmp_load` slúži na načítanie bázy informácií MIB pomocou PHP funkcie `snmp_read_mib` a jej volaním sa načítava MIB príslušného riadeného mikrokontroléru. Funkcia `snmp_get` prostredníctvom funkcie snmprealwalk postupným prechodom získa všetky hodnoty MIB z mikrokontroléru a vráti ich ako pole identifikátorov objektov OID a príslušných hodnôt. Funkcia `snmp_set` využíva funkciu PHP s názvom snmpset na nastavenie hodnoty objektu do mikrokontroléru.

Grafické používateľské rozhranie je zabezpečené menom a heslom. Implementácia zabezpečenia sa nachádza v súboroch `login.php`, `logout.php` a `lock.php`. Súbor `login.php` zobrazuje prihlasovací formulár, overuje správnosť zadaného mena a hesla a ukladá relačnú premennú, pomocou ktorej sa bude overovať prihlásenie používateľa. Súbor `logout.php` slúži na odhlásenie používateľa a súbor `lock.php` overuje prihlásenie používateľa a v prípade, že používateľ nie je prihlásený, zobrazí prihlasovací formulár.

Prostredníctvom grafického používateľského rozhrania je možné nastaviť každý vývod portu zvlášť pre registre `PORT` a `DDR`. Zmena registra `PIN` nie je možná, pretože sa jedná o vstupný register. Zaškrtávacie pole `Input` slúži na nastavenie alebo resetovanie celého registra `DDR` pre príslušný port. Textové pole `Value` zobrazuje desiatkovú hodnotu registra `PIN` príslušného portu. Pre zápis nastavených údajov alebo načítanie nových hodnôt slúži tlačítko `Write`, ktoré zapíše všetky nastavené hodnoty a zároveň načíta nový stav všetkých portov.

# 6 Záver

Hlavným cieľom tejto práce bolo pripojiť mikrokontrolér ATmega162 od firmy ATMEL do IP siete a implementovať podporu riadenia tohoto mikrokontroléru prostredníctvom protokolu SNMP. V prvej časti je čitateľ oboznámený s ponukou riešení založených na mikroprocesoroch, ktorá zahŕňa rady ARM, AVR, Intel 8051 a riešenia založené na bezdrôtových technológiách s modulmi Zig-Bee. Keďže práca je primárne zameraná na mikrokontroléry rady AVR, je architektúra tejto rady analyzovaná detailnejšie.

Ďalšia časť práce je zameraná na analýzu pripojenia mikrokontrolérov rady AVR do prostredia IP siete. Po analýze rôznych riešení bol na začiatku vývoja práce zvolený Lantronix XPort, ktorý sa neskôr ukázal ako nepoužiteľný pre požadovanú funkcionalitu a bol nahradený riešením založeným na integrovanom obvode ENC28J60. Pomocou tohoto integrovaného obvodu je mikrokontrolér pripojený do siete Ethernet. Keďže tento integrovaný obvod implementuje iba fyzickú a spojovú vrstvu OSI modelu, bolo nutné implementovať vyššie vrstvy priamo v softvéri mikrokontroléru. Na implementáciu protokolov IP, ARP, ICMP, UDP a TCP bol použitý IP zásobník s otvoreným zdrojovým kódom, ktorý priamo podporuje komunikáciu s integrovaným obvodom ENC28J60.

Implementácii samotného protokolu SNMP predchádza detailná analýza štruktúry tohoto protokolu. Protokol SNMP verzie 1, ktorý bol v tejto práci implementovaný, využíva päť typov správ: GetRequest, GetNextRequest, GetResponse, SetRequest a Trap. Z týchto sú v mikrokontroléri implementované iba prvé štyri. Posledná správa slúži na signalizovanie výnimočnej udalosti a v prípade vzdialeného riadenia portov by ju bolo možné využiť v prípade, že by ňou bolo signalizované prerušenie od vonkajšej udalosti, ako napríklad zmena hodnoty na porte. Riešenie prerušení ale nebolo v tejto práci spracované a preto táto funkcionalita ostáva ako možnosť budúceho rozšírenia.

Protokol SNMP využíva na svoj prenos v prostredí IP siete transportný protokol UDP, v ktorom sú všetky časti SNMP správy kódované pomocou notácie ASN.1. Táto abstraktná notácia umožňuje kódovanie a dekódovanie nielen SNMP správ ale využíva sa aj v iných prípadoch v telekomunikačných a sieťových prenosoch. Správy kódované notáciou ASN.1 je potrebné pred spracovaním dekódovať. Dekódovanie a overenie správnej štruktúry správy je dôležité hlavne preto, že je na prenos použitá nespoľahlivá transportná služba UDP, pri ktorej nie je garantovaná bezchybnosť prenosu. Po dekódovaní a overení správy získavame zoznam premenných a informáciu o type správy. Správy GetRequest a GetNextRequest slúžia na získanie hodnôt z mikrokontroléru zatiaľčo správa SetRequest slúži na zápis hodnôt do mikrokontroléru. V oboch prípadoch je ako odpoveď použitá správa GetResponse.

Implementácia vzdialeného riadenia mikrokontroléru z IP siete prostredníctvom protokolu SNMP pozostáva zo spracovania IP paketov IP zásobníkom, následné získanie údajov z UDP datagramu, ktoré reprezentujú SNMP správu kódovanú notáciou ASN.1, dekódovania a overenia štruktúry správy, vykonania príslušnej funkcionality a odoslania odpovede sieťovému manažérovi agentom implementovaným v softvéri mikrokontroléru.

Pomocou uvedenej implementácie SNMP protokolu je možné riadiť mikrokontrolér s využitím štandardných SNMP utilít v operačnom systéme alebo pomocou špecializovaných nástrojov určených pre riadenie sieťových zariadení implementujúcich protokol SNMP. Okrem týchto možností bolo vytvorené grafické používateľské rozhranie využívajúce protokol HTTP, ktoré používa externý webový server, na ktorom sú umiestené webové stránky implementujúce grafické používateľské rozhranie. Webové stránky sú postavené na technológiách HTML, PHP, CSS a JavaScript.

Aj keď je špecifikácia protokolu SNMP verzie 1 považovaná za zastaranú, stále sa v dnešnej dobe implementuje v rôznych sieťových zariadeniach hlavne pre svoju jednoduchosť a spätnú kompatibilitu so staršími zariadeniami. Jednou z možností rozšírenia tejto práce by mohla byť implementácie protokolu SNMP verzie 2c, ktorá

sa do verzie 1 veľmi nelíši, ale zároveň nepodporuje kvalitnú úroveň zabezpečenia. Iná možnosť je implementácia verzie 3, ktorá je síce najkomplikovanejšia na implementáciu, ale poskytuje vysokú úroveň zabezpečenia prenášaných informácií a zároveň implementuje komplexnú autentifikačnú schému vďaka ktorej poskytuje oveľa väčšiu bezpečnosť ako verzie 1 a 2c.

Ďalšou možnosťou rozšírenia tejto práce by mohla byť implementácia riadenia rôznych periférií mikrokontroléru, ako napríklad čítačov/časovačov, generátora PWM signálu, analógovo-digitálneho prevodníku a pod.. Aj keď v tejto práci je riešené iba riadenie všeobecných vstupno-výstupných portov mikrokontroléru, rozšírenie o iné periférie by nebolo veľmi zložité a navyše by to zväčšilo možnosti využitia vzdialeného riadenia mikrokontrolérov.

Celkovo boli v práci splnené všetky vopred vytýčené ciele. V prvej časti bola analyzovaná architektúra mikrokontrolérov so zameraním na produkty firmy ATMEL. Zároveň boli analyzované možnosti pripojenia mikrokontroléru AVR do IP siete a zvolené riešenie s integrovaným obvodom ENC28J60. V časti návrhu boli porovnané protokoly Telnet, SSH, SNMP a HTTP, z ktorých boli pre implementáciu riadenia mikrokontroléru vybrané protokoly SNMP a HTTP. V implementačnej časti bolo zostrojené vývojové prostredie s mikrokontrolérmi ATmega162 a ATtiny2313, v ktorých bol implementovaný kód riadiaci tieto mikrokontroléry vzdialene z prostredia IP siete. Protokol SNMP bol implementovaný priamo do mikrokontroléru ATmega162 zatiaľčo mikrokontrolér ATtiny2313 je riadený prostredníctvom pripojenia na mikrokontrolér ATmega162 pomocou navrhnutého sériového protokolu. V poslednej časti implementácie bolo vytvorené grafické používateľské rozhranie pre vzdialené riadenie mikrokontrolérov založené na protokole HTTP a technológiách PHP, HTML, CSS a JavaScript.

# Zoznam použitej literatúry

[1] Atmel Corporation. *Atmel Corporation: Microcontrollers, 32-bit, and touch solutions* [online]. 2013 [cit. 2013-03-31].
Dostupné z: `http://www.atmel.com/`

[2] BARRETT, Steven F. a Daniel J. PACK. *Atmel AVR microcontroller primer programming and interfacing*. San Rafael, Calif.: Morgan & Claypool, 2008, 180 s. ISBN 978-159-8295-429.

[3] CAMERA, Dean. *Using the USART in AVR-GCC* [online]. 2013 [cit. 2013-04-01].
Dostupné z: `http://deans-avr-tutorials.googlecode.com/svn/trunk/USART/Output/USART.pdf`

[4] CASE, J., M. FEDOR, M. SCHOFFSTALL a J. DAVIN. RFC 1157. *A Simple Network Management Protocol (SNMP)*. MIT Laboratory for Computer Science, 1990, 36 s.
Dostupné z: `http://www.ietf.org/rfc/rfc1157.txt`

[5] Data Sheet. *ENC28J60: Stand-Alone Ethernet Controller with SPI™ Interface*. U.S.A.: Microchip Technology Inc., 2004.
Dostupné z: `http://ww1.microchip.com/downloads/en/devicedoc/39662a.pdf`

[6] ITU-T STUDY GROUP 17. ISO/IEC 8824-1. *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*. 2008, 180 s.
Dostupné z: `http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.680-200811-I!!PDF-E&type=items`

[7] ITU-T STUDY GROUP 17. ISO/IEC 8825-1. *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical En-*

*coding Rules (CER) and Distinguished Encoding Rules (DER)*. 2008, 27 s. Dostupné z: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.690-200811-I!!PDF-E&type=items

[8] LARMOUTH, J. *ASN.1 Complete* [online]. Open Systems Solutions, 1999, 366 s. [cit. 2013-04-02]. Dostupné z: http://www.oss.com/asn1/larmouth.html

[9] LEAVER, Chris. *Introduction to Atmel AVR microcontroller development: using free software with worked examples*. S.l.: Sylvania Books, 2010, 256 s. ISBN 978-095-6728-500.

[10] MAURO, D. *Essential SNMP*. Cambridge: O Reilly, 2001, 313 s. ISBN 05-960-0020-0.

[11] MCCLOGHRIE, K. a M. ROSE. RFC 1213. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. Hughes LAN Systems, Inc., 1991, 70 s. Dostupné z: http://www.ietf.org/rfc/rfc1213.txt

[12] MIČEK, J. a O. KARPIŠ. *Monolitické mikropočítače s jadrom ARM7: architektúra, programovanie a aplikácie*. Žilina: EDIS–vydavateľstvo Žilinskej univerzity, 2009, 308 s. ISBN 978-80-554-0134-8.

[13] POSTEL, J. a J. REYNOLDS. ISI. RFC 854. *Telnet protocol specification*. 1983, 15 s. Dostupné z: http://tools.ietf.org/html/rfc854

[14] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. České Budějovice: Kopp, 2004, 607 s. ISBN 80-723-2236-2.

[15] ROSE, M. a K. MCCLOGHRIE. RFC 1155. *Structure and Identification of Management Information for TCP/IP-based Internets*. Hughes LAN Systems, 1990, 22 s. Dostupné z: http://www.ietf.org/rfc/rfc1155.txt

# Zoznam príloh

**Príloha A** Používateľská príručka

**Príloha B** Systémová príručka

**Príloha C** CD