

# Centralizovaný řídicí systém na bázi TCP/IP

Centralized control system based on TCP/IP

Bc. Richard Bachánek

---

Diplomová práce  
2014

 Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2013/2014

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Richard Bachánek**  
Osobní číslo: **A12288**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Bezpečnostní technologie, systémy a management**  
Forma studia: **prezenční**

Téma práce: **Centralizovaný řídicí systém na bázi TCP/IP**

Téma anglicky: **A Centralized Control System Based on TCP/IP**

Zásady pro vypracování:

1. Navrhněte a popište strukturu navrhovaného řídicího systému.
2. Softwarově i hardwarově realizujte senzorický subsystém řídicího systému.
3. Realizujte subsystém pro řízení periférií výkonových zařízení – řízení jasu, spínání osvětlení, či řízení motorů.
4. Navrhněte a realizujte komunikační rozhraní – drátové a bezdrátové řešení na bázi TCP/IP.
5. Realizujte systém pro zpracování informací z obou subsystémů.
6. Výsledky presentujte na reálném modelu.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **NOVÁK, Petr.** Mobilní roboty: pohony, senzory, řízení. Vyd. 1. Praha: BEN – technická literatura, 2004, 247 s. ISBN 80-730-0141-1.
2. **LÁNÍČEK, R.** Elektronika obvodů, součástky, děje. Praha, 1998, 478 s. ISBN 80-860-5625-2.
3. **VÁŇA, Vladimír.** Mikrokontroléry ATMEL AVR: programování v jazyce C : popis a práce ve vývojovém prostředí CodeVisionAVR C. 1. vyd. Praha: BEN – technická literatura, 2003, 215 s. ISBN 3-7723-4154-3.
4. **MANN, Burkhard.** C für Mikrocontroller: ANSI-C, C-Compiler/Linker, Echtzeitbetriebssysteme, C-Programmierbeispiele, Tools für die Programmierung, Tipps und Tricks ; mit 30 Tabellen ; lauf CD-ROM: Beispielsammlung, AVR-Studio, Debugger, Free- und Shareware-Tools, Applikationshinweise. Poing: Franzis, 2000. ISBN 37-723-4154-3.
5. **STENGL, Jens Peer a Jenö TIHANYI.** Výkonové tranzistory MOSFET. 1. české vyd. Praha: BEN – technická literatura, 1999, 191 s. ISBN 80-860-5654-6.

Vedoucí diplomové práce:

**Ing. Ján Ivanka**

Ústav bezpečnostního inženýrství

Datum zadání diplomové práce:

**7. února 2014**

Termín odevzdání diplomové práce:

**27. května 2014**

Ve Zlíně dne 7. února 2014

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



doc. RNDr. Vojtěch Křesálek, CSc.  
*ředitel ústavu*

## **ABSTRAKT**

Cílem diplomové práce je návrh a realizace systému řízení pro soustavu detektorů a aktorů. Tyto periferie jsou řešeny moderním a levným způsobem, který ke komunikaci s nadřazeným systémem využívá TCP/IP protokol a bezdrátovou síť WIFI. V průběhu textu je diskutováno zabezpečení komunikace v rámci intranetu a Internetu, ale i možné alternativy přenosu informace. Praktická část práce pojednává o softwarovém a hardwarovém řešení jednotlivých periferií a také funkci řídicího systému, která je společně s periferiemi demonstrována na diferenčně řízeném robotovi, ovládaném prostřednictvím bezdrátové WIFI sítě.

Klíčová slova: Řídicí systém, detektor, aktor, robot, TCP/IP, WIFI.

## **ABSTRACT**

The aim of the thesis is the design and implementation of the management system for a set of detectors and actuators. These peripherals are designed with modern and cost-effective way to communicate with the master system uses the TCP / IP protocol and WIFI wireless network. Throughout the text is discussed security communication in the intranet and the Internet, as well as possible alternatives to transmission of information. The practical part deals with software and hardware solutions to individual devices and the control system, which along with peripherals demonstrated to differentially control the robot, controlled via wireless WIFI network.

Keywords: Control system, detector, actuator, robot, TCP/IP, WIFI.

Rád bych tímto poděkoval svému vedoucímu diplomové práce Ing. Jánů Ivankovi za ochotu, pomoc, cenné rady, připomínky, za věnovaný čas k úpravě a návrhům formy zpracování diplomové práce.

**Prohlašuji, že**

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

**Prohlašuji,**

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

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD .....</b>	<b>9</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>11</b>
<b>1 ŘÍDICÍ SYSTÉMY V BUDOVÁCH .....</b>	<b>12</b>
1.1 HISTORIE ŘÍDICÍCH SYSTÉMŮ .....	12
1.2 CENTRALIZOVANÉ ŘÍDICÍ SYSTÉMY .....	13
1.3 HYBRIDNÍ ŘÍDICÍ SYSTÉMY .....	14
1.4 DECENTRALIZOVANÉ ŘÍDICÍ SYSTÉMY .....	14
1.5 FUNKCE ŘÍDICÍCH SYSTÉMŮ .....	15
<b>2 TEORIE A PROGRAMOVÁNÍ MIKROKONTROLERŮ .....</b>	<b>16</b>
2.1 MIKROKONTROLER .....	16
2.1.1 CPU (Central Processing Unit) .....	17
2.1.2 Paměť .....	17
2.1.3 I/O rozhraní (vstupní/výstupní rozhraní) .....	18
2.1.4 Časovač/čítač .....	18
2.1.5 Sériové rozhraní .....	18
2.2 PROGRAMOVÁNÍ MIKROKONTROLERU .....	19
2.2.1 Programovací software .....	19
2.2.2 Nahrání programu do mikrokontroleru .....	20
2.2.2.1 Nahrání bootloderu .....	20
2.2.2.2 Nahrání řídicího programu do samostatného MCU .....	22
<b>II PRAKTICKÁ ČÁST .....</b>	<b>23</b>
<b>3 VLASTNÍ ŘEŠENÍ .....</b>	<b>24</b>
3.1 STRUKTURA NAVRHOVANÉHO ŘÍDICÍHO SYSTÉMU .....	24
3.2 REALIZOVANÉ ŘEŠENÍ .....	26
3.2.1 Detektor .....	26
3.2.2 WI-FI Robot .....	26
3.2.3 AC Dimmer .....	26
<b>4 ŘÍZENÍ MIKROKONTROLERU .....</b>	<b>27</b>
<b>5 SUBSYSTÉM DETEKTORŮ .....</b>	<b>29</b>
5.1.1 Komunikace .....	30
<b>6 AKČNÍ ČLENY .....</b>	<b>32</b>
6.1 AKTOR - TESTOVACÍ ROBOT .....	32
6.1.1 Konstrukce .....	32
6.1.2 Zapojení .....	33
6.1.3 Řídicí aplikace .....	36
6.2 AKTOR – AC DIMMER .....	39
6.2.1 Zapojení .....	41
6.2.2 Řídicí aplikace .....	45
<b>7 VZDÁLENÍ ŘÍZENÍ .....</b>	<b>48</b>
<b>8 ZABEZPEČENÍ PŘENOSU .....</b>	<b>50</b>
<b>ZÁVĚR .....</b>	<b>51</b>
<b>ZÁVĚR V ANGLIČTINĚ .....</b>	<b>53</b>

<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>55</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>57</b>
<b>SEZNAM OBRÁZKŮ .....</b>	<b>58</b>
<b>SEZNAM TABULEK .....</b>	<b>59</b>
<b>SEZNAM VZORCŮ.....</b>	<b>60</b>

## ÚVOD

Úkolem diplomové práce je návrh a realizace centralizovaného řídicího systému pro menší objekty, jako jsou rodinné domy či menší firmy, pro které jsou ceny stávajících řešení stále příliš vysoké. Navrhovaný způsob využívá ke komunikaci s centrálním řídicím zařízením TCP/IP protokol (Transmission Control Protocol/Internet Protocol – primární přenosový protokol/protokol síťové vrstvy), který zaručuje kompatibilitu s moderními PC sítěmi a veškerou současnou výpočetní technikou, díky čemuž je tento způsob unikátní a výrazně rozšiřuje jeho možnosti oproti konkurenčním řešením. Protože je možné využít jako sběrnici WI-FI síť a pro řízení a monitorování jednotlivých prvků, které představují detektory a akční členy, například mobilní telefon, tablet či notebook, také výrazně snížíme cenu celého řídicího systému. Není nutné instalovat přídavnou kabeláž pro propojení prvků se systémem, protože se o komunikaci stará bezdrátová síť, kterou již většina objektů disponuje. Není dokonce nutné pořizovat terminály pro ovládání systému, jelikož lze realizovat přístup z PC, nebo mobilního telefonu a to dokonce vzdáleně – pomocí veřejné telekomunikační sítě Internet.

Určité formy řízení se v současné době uplatňují již téměř v každé budově. Může se jednat například o regulaci teploty pomocí termostatu, automatické spínání osvětlení, zavlažování, ale také monitorování stavu senzorů. Výsledný efekt je takový, že máme v objektu řadu na sobě nezávislých subsystémů, které nemají kolektivní informace z ostatních subsystémů a které je nutné řídit samostatně pomocí speciálních terminálů navržených pouze pro ně samotné. S rostoucí potřebou společnosti na pohodlí a automatizaci některých procesů, je takový trend prakticky neudržitelný a není ekonomické ani užitečné, aby řízené periferie v objektu nebyly nijak integrovány do jednoho celku, který by umožnil jejich centrální řízení.

Je tedy patrné, že popisovaný systém je schopen plnit nejen funkce řídicích systémů, ale také současných bezpečnostních systémů, a to s minimálními pořizovacími náklady. Pro většinu z nás se stali výpočetní technologie nedílnou součástí běžného života. Mobilní telefony, notebooky či tablet PC využívá stále více lidí. Přesto, že se výpočetní výkon těchto zařízení stále zvyšuje, potenciál, který nabízí, zůstává nevyužitý. Ve spojení s moderní elektronikou, kterou představují mikrokontrolery nebo hradlová pole, mohou mobilní telefony a další podobná zařízení získat zcela nový rozměr. Mohou sloužit pro vzdálené řízení a monitorování periférií, umístěných v objektu a v případě dostupného Internetového připojení lze tento vzdálený přístup realizovat na jakoukoliv vzdálenost. V

průběhu diplomové práce je diskutována také realizace vzdáleného přístupu, návrh a konstrukce jednotlivých periférií umístěných v objektu, jejichž srdcem je právě zmiňovaný mikrokontroler. Cílem je poukázat na jednoduchost takto koncipovaného řešení a především razantní snížení ceny celého řídicího systému díky využití TCP/IP protokolu a bezdrátové technologii WI-FI.

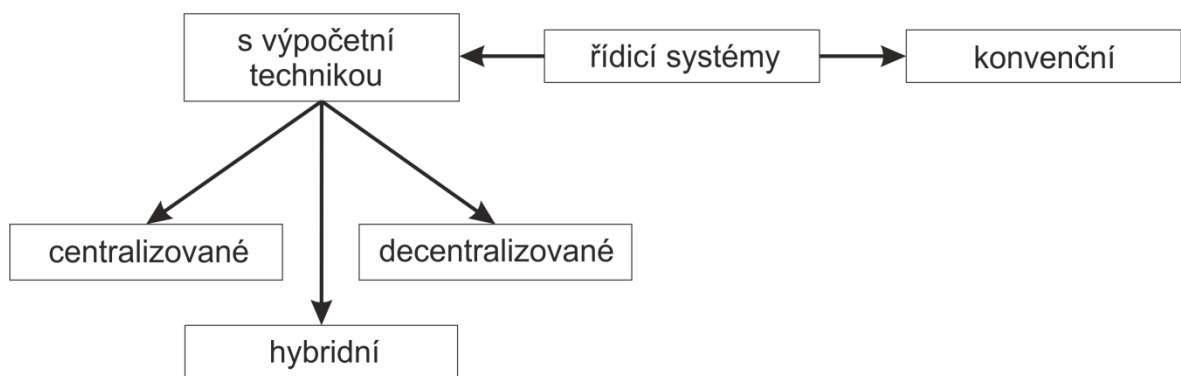
V posledních letech se řídicí systémy stále více rozšiřují také do menších objektů, z kterých se pak stávají inteligentní budovy. Tyto systémy jsou velmi nákladné, což také zpomaluje jejich vývoj, z důvodu malého zájmu ze strany veřejnosti. V podstatě existují dva odlišné přístupy k výše zmíněnému řešení – jedná se o centralizované a decentralizované systémy. První zmiňované disponují jednou centrální jednotkou, která sbírá a vyhodnocuje informace ze všech periférií rozmístěných v celém objektu. Pro realizaci systému popisovaného v tomto textu byl zvolen právě tento přístup z důvodu jednoduchosti, bezpečnosti ale také nižším nákladům na realizaci. V následujících kapitolách je popsán jeden z možných způsobů, jak lze veškeré subsystémy v objektu integrovat do jednoho centrálního systému a jeho prostřednictvím je monitorovat a ovládat.[8]

## **I. TEORETICKÁ ČÁST**

# 1 ŘÍDICÍ SYSTÉMY V BUDOVÁCH

## 1.1 Historie řídicích systémů

První řídicí systémy se začaly objevovat kolem roku 1970 s nástupem TTL (Transistor Transistor Logic) obvodů. Tento standard se využívá pro implementaci digitálních integrovaných a logických obvodů. Vychází z použití technologie bipolárních křemíkových tranzistorů. První zařízení využívající tuto technologii byly především digitální váhy, dále obráběcí a svařovací automaty. V obr. 1 jsou řazeny systémy využívající tuto technologii do oboru konvenčních řídicích systémů. O několik let později, s nástupem procesorů od firmy Intel, převzal řízení systémů první 8bitový procesor technologie NMOS (N-type Metal Oxide Semiconductor), což umožnilo aplikaci výpočetní techniky do oblasti řízení. Jeho taktovací frekvence byla 1-2MHz (až 2 000 000 operací za sekundu) a stal se ve své době snad nejrozšířenějším procesorem pro účely řídicích systémů, ale také pro první domácí počítače. Převážně se využíval v řadě průmyslových odvětví pro účely automatizace a řízení strojů. Krátce na to se již začaly objevovat první 32bitové procesory, které dokázaly ovládat komplexní zařízení používané například v leteckém průmyslu, jako jsou různé manipulátory pro řízení motorů, lopatek turbín atd.

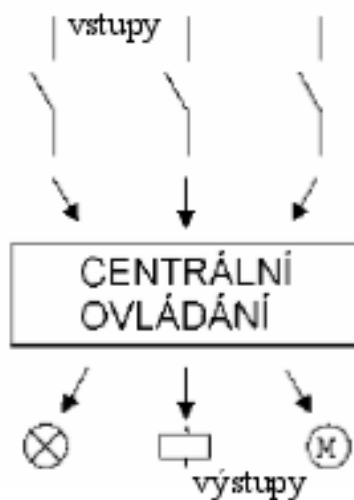


Obr. 1. Rozdělení řídicích systémů z pohledu historie

Současné moderní řídicí systémy využívají dva rozdílné přístupy. Jedná se o centralizovaný a decentralizovaný způsob řízení. Nelze říci, který přístup je lepší, či efektivnější. Každý disponuje jistými výhodami či omezeními, ale funkce, které oba dokážou plnit, jsou prakticky totožné. Kombinací centralizovaného a decentralizovaného systému získáme hybridní systém, který kombinuje výhody obou způsobů. Subsystem vstupů bývá obvykle řešen jako decentralizovaný systém a výstupy jsou řešeny centralizovaně. [6][7][8]

## 1.2 Centralizované řídicí systémy

V objektech vybavených centrálním řídicím systémem jsou pro řízení elektrických spotřebičů využívány vstupy a výstupy řídicího systému a prvky které sdružuje, jsou propojeny pomocí hvězdicové topologie. Každý jednotlivý senzor, či aktor je samostatně propojen (drátově, bezdrátově) s nadřazeným systémem. Všichni účastníci spolu mohou komunikovat prostřednictvím centrálního zařízení.

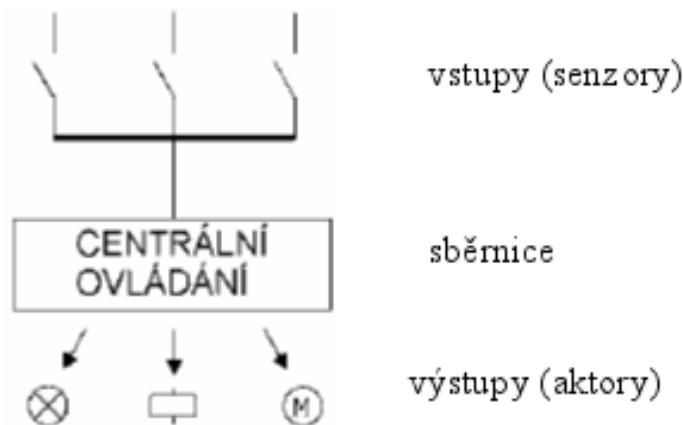


Obr. 2. Struktura centralizovaného řídicího systému

Z obr. 2 lze přesně vidět, jak tento způsob řízení probíhá. Vstupy jsou do systému přivedeny samostatně, stejně jako výstupy. Díky tomu lze každý prvek, ať už detektor, či akční člen, řídit a monitorovat samostatně.[6][16]

### 1.3 Hybridní řídicí systémy

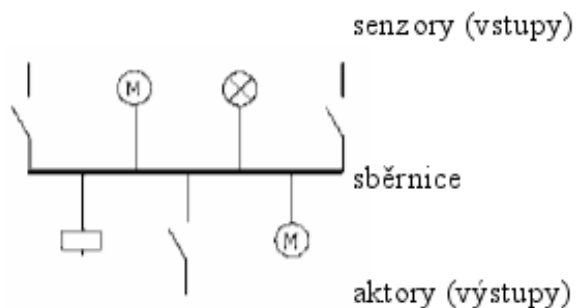
V hybridních systémech se detektory, tedy vstupy zapojují do společné sběrnice a výstupy jsou zapojeny pomocí hvězdicové topologie. Jedná se tedy o částečně decentralizovaný řídicí systém. Prvky přivedené na vstupy řídicího systému pomocí sběrnice obsahují mikroprocesor s pamětí, což jim dává možnost pracovat nezávisle (obr. 3).[6][16]



Obr. 3. Struktura hybridního řídicího systému

### 1.4 Decentralizované řídicí systémy

Řídicí systémy tohoto typu nedisponují centrální řídicí jednotkou. Každý účastník, tedy detektory či akční členy obsahují vlastní mikroprocesor s pamětí (mikrokontroler), který jim propůjčuje určitou „inteligenci“. Každý samostatný prvek je tedy schopen pracovat nezávisle. Veškeré členy se propojují pomocí společné sběrnice.[6][16]



Obr. 4. Struktura decentralizovaného řídicího systému

## 1.5 Funkce řídicích systémů

Řídicí systém je srdcem každé budovy, v které implementujeme nějakou formu řízení. Díky němu získá budova určitou formu inteligence a také zcela nový rozměr v jejím řízení a monitorování. Kromě automatizovaných úloh, jako je například automatické řízení žaluzií, nebo rolet na základě intenzity dopadajícího světla, lze získávat také informace o spotřebě energií a na základě nich vytvářet statistiku a zvýšit tak efektivnost provozu. Funkce řídicích systémů jsou patrné z následujícího seznamu, který ale netvoří kompletní výčet možných funkcí, který by byl velmi obsáhlý.

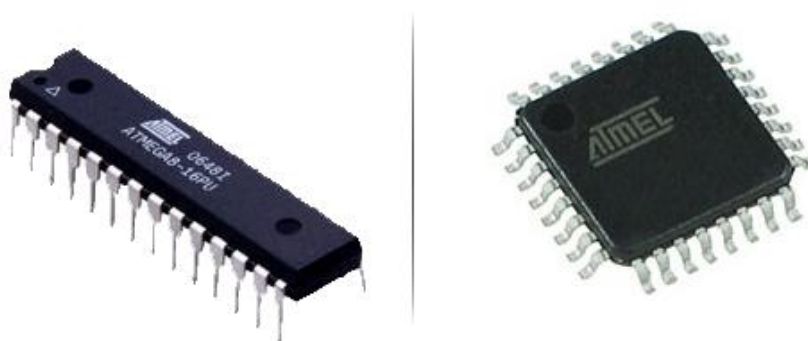
- měření fyzikálních veličin v zájmovém prostoru,
- regulace technologických procesů v budově,
- logické řízení,
- zálohování a ukládání informací o procesu,
- analýza procesu – předvídání mezních a havarijních situací,
- informace pro správce, či uživatele,
- příjem a zpracování povelů,
- realizace vyšších řídicích algoritmů - koordinace, optimalizace, aplikace metod, umělé inteligence,
- poskytování dat uživateli, prostřednictvím informační sítě.

Realizace zmíněných automatizačních systémů, které jsou v současnosti dostupná na trhu, přináší řadu výhod, ale také výrazně zvyšuje investice do objektu a není jisté, zda bude úspora za energie ekonomická, vzhledem k pořizovacím nákladům řídicího systému. Proto je nutné nacházet nová řešení a přístupy, jakými lze dosáhnout podstatně nižších pořizovacích nákladů a přitom zachovat všechny výhody. Jedním z takových řešení je navrhovaný řídicí systém na bázi protokolu TCP/IP. Právě díky kompatibilitě se zmíněným protokolem lze celý systém i jeho instalaci zjednodušit a také výrazně zlevnit, což by mohlo také přispět k rychlejšímu rozvoji těchto technologií.[6][7][8]

## 2 TEORIE A PROGRAMOVÁNÍ MIKROKONTROLERŮ

### 2.1 Mikrokontroler

Mikrokontrolery (MCU, nebo  $\mu\text{C}$ ) jsou programovatelné elektronické součástky, které se podobají standardním integrovaným obvodům, viz obr. 5. Česky se také někdy označují jako jednočipové počítače, či mikropočítače. Jsou to monolitické integrované obvody, které obsahují mikroprocesor (CPU), paměť, vstupně-výstupní rozhraní a další periferní obvody (například AD převodníky, čítače, časovače, atd.).



Obr. 5. Mikrokontroler Atmel ATmega8\* / SMD provedení

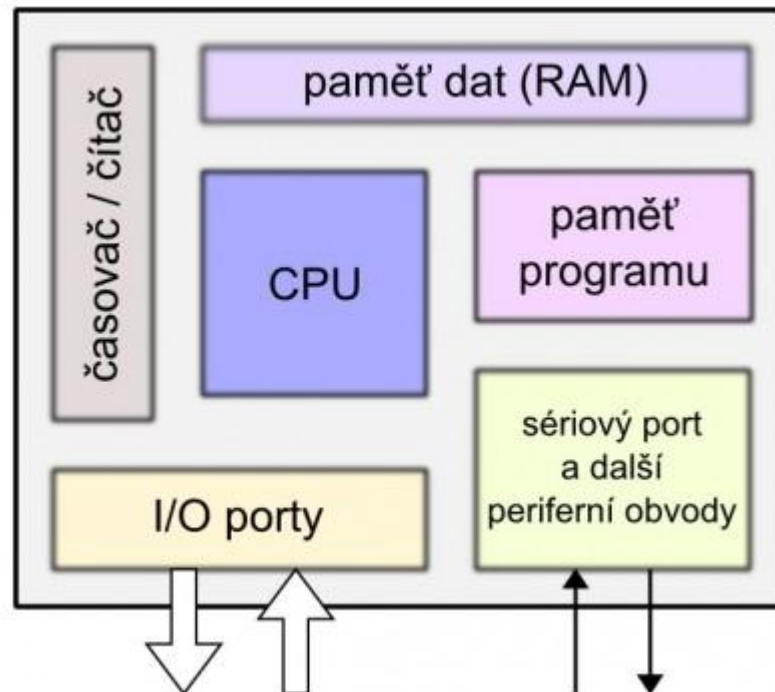
Jednočipové počítače vynikají výbornou spolehlivostí a kompaktností a využívají se především v jednoúčelových aplikacích, jako například řízení, či regulace a mnoha jiných.

Z pohledu historie lze dělit architekturu mikrokontrolerů následovně:

- Harvardská architektura
- Von Neumannova architektura

Von Neumannova architektura využívá společnou paměť pro program i data, díky čemuž není nutné rozlišovat instrukce pro přístup k paměti dat a paměti programu. To vede k zjednodušení samotného čipu. Další výhodou je existence pouze jedné datové sběrnice, která distribuuje oba typy dat, což je výhodné například v případě použití externích pamětí, čímž se reguluje potřebný počet vstupů a výstupů. Harvardská architektura paměť pro program a data, na rozdíl od předešlé architektury odděluje, což zvyšuje technologickou náročnost výroby mikrokontroleru, protože je nutné použít dvě sběrnice. Výhodou ale potom je, že lze použít jiné bitové šířky pro datovou a programovou sběrnici. Tato možnost se využívá poměrně často, takže můžeme najít například 8bitové mikrokontrolery s šířkou programové sběrnice 12, 14, nebo i 16 bitů. Protože lze u tohoto typu architektury, díky

existenci dvou sběrnic, lze číst data i instrukci v jeden okamžik, zvýší se tak rychlost vykonávání instrukcí. Základní strukturu mikrokontroleru lze vyčíst z blokového schématu na obr. 6.[2][3][9]



Obr. 6. Základní struktura mikrokontroleru

### 2.1.1 CPU (Central Processing Unit)

CPU neboli mikroprocesor je nejdůležitější částí mikrokontroleru. Jeho úkolem je načítání, dekódování a vykonávání jednotlivých instrukcí, které jsou uloženy v programové paměti a také řídí činnost celého mikrokontroleru. Dále obsahuje ALU (Arithmetic Logic Unit), pomocí které může vykonávat aritmetické a logické operace.[9]

### 2.1.2 Paměť

Další podstatnou částí každého mikrokontroleru je paměť, do které se ukládá program samotný (převedený na instrukce) a také data s kterými mikrokontroler pracuje. Jelikož se zde využívá především Harvardská architektura, bývá paměť pro data a program oddělena. V dnešní době se paměť pro program realizuje jako FLASH paměť, kterou lze v případě potřeby snadno přehrát. Její kapacita se pohybuje kolem 256kB pro program a 16kB pro data. Pro uložení proměnných a výsledků aritmetických operací bývá použita statická paměť RAM (random-access memory). Zde platí stejná omezení jako v případě běžných PC – paměť se smaže po odpojení mikrokontroleru od napájecího napětí. Kromě výše

zmíněných pamětí dnešní mikrokontrolery obvykle obsahují další paměť typu EEPROM (Electrically Erasable Programmable Read-Only Memory), která slouží k zálohování důležitých dat.[9]

### 2.1.3 I/O rozhraní (vstupní/výstupní rozhraní)

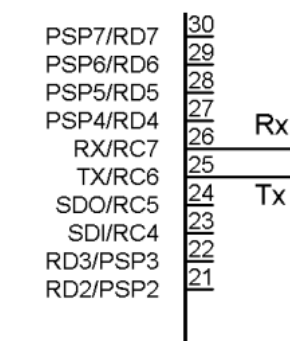
Jedná se o porty mikrokontroleru, ke kterým se připojují vstupní, nebo výstupní zařízení. Zprostředkovávají tedy výměnu informací prostřednictvím změny napětí na vstupu, či výstupu, mezi mikrokontrolerem a jinou součástkou, nebo obvodem. V praxi se ke vstupům často připojují například tlačítka, snímače, detektory atd., výstupy například pro komunikaci s LCD (Liquid Crystal Display).[9]

### 2.1.4 Časovač/čítač

Mikrokontrolery dále obsahují jeden, či více čítačů/časovačů, se kterými lze pracovat a které lze také konfigurovat. Popis konfigurace, která je mnohdy značně rozsáhlý, můžeme najít v datasheetu daného mikrokontroleru. Čítače a časovače umožňují programu mikrokontroleru například měřit časové intervaly, detekovat počet impulsů za určitý časový interval a podobně.[9]

### 2.1.5 Sériové rozhraní

Sériové rozhraní, které dnešní MCU disponují, může nabývat softwarové a hardwarové podoby. Mikrokontrolery obvykle obsahují jeden, či více hardwarových vstupů/výstupů sériové komunikace (USART - Universal Synchronous/Asynchronous Receiver and Transmitter) a libovolný počet softwarových (nastavitelných) vstupů/výstupů USART.



Obr. 7. Vstup/Výstup USART

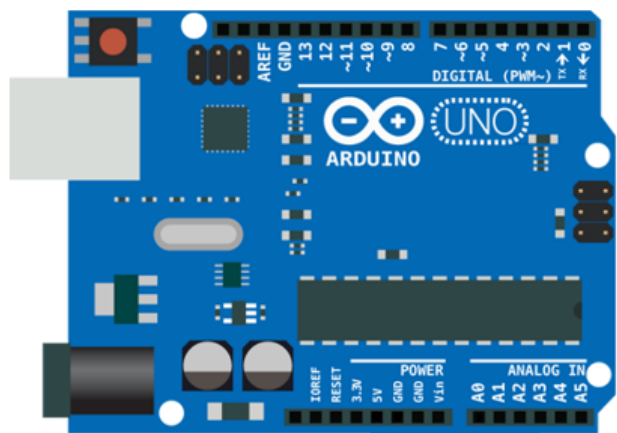
Na obr. 7 můžeme vidět hardwarový vstup – bývá označen jako Rx a výstup, označen jako Tx, rozhraní USART, pomocí kterých je mikrokontroler schopen sériově komunikovat s dalším hardwarem – mikrokontrolerem, PC, WI-FI či Bluetooth modulem, atd.[9]

## 2.2 Programování mikrokontroleru

### 2.2.1 Programovací software

Pro programování mikrokontrolerů použitých při výrobě zařízení v praktické části je použit programovací software Arduino a stejnojmenný vývojový kit, který slouží k tvorbě a testování programů. Kit lze poměrně jednoduše přeprogramovat na zařízení, kterému říkáme programátor, jehož funkcí je nahrání programu a bootloaderu do samostatného mikrokontroleru. Tento postup je podrobněji popsán v následující kapitole.

Programovací jazyk pro Arduino se nazývá Arduino Programmable Language, který je založen na jazyku Wiring, což je jazyk velmi podobný programovacímu jazyku C, který je určen pro vývoj software pro jednočipové počítače. Na obr. 8 vlevo je znázorněno IDE (Integrated Development Environment) programovacího jazyku arduino a vpravo vývojový kit.



Obr. 8. IDE programovacího SW Arduino | HW Arduino (vývojový kit)

Program se nahrává do MCU prostřednictvím USB (Universal Serial Bus) rozhraní. Tato open-source platforma založená na mikrokontroleru ATmega od firmy Atmel, může být využita k vytváření samostatných interaktivních zapojení nebo může být připojeno k software na počítači.

Každý program lze rozdělit na tři základní části. Nejprve deklarujeme knihovny, proměnné a další nastavení v případě potřeby, jako je například IP, MAC adresa zařízení, porty sériové komunikace, atd. Výchozí nastavení programu definujeme ve funkci *setup*. Zde probíhá také inicializace sériové komunikace a další nastavení, která potřebujeme vykonat pouze jednou po spuštění programu. Poslední částí je funkce *loop*. Jedná se o nekonečnou smyčku, v které probíhá tělo programu. Může se například jednat o čtení hodnot ze senzorů, či spínání a úprava napětí na výstupech mikrokontroleru.[4][10]

## 2.2.2 Nahrání programu do mikrokontroleru

Jak již bylo řečeno v předchozí kapitole, vývojový kit Arduino lze kromě testování programu využít také jako hardwarový programátor – tedy zařízení, které je schopno programovat samostatné mikrokontrolery. Tento postup se hodí především pro stavbu konkrétních zařízení, pro která již máme naprogramovaný a otestovaný program. Přesto, že je tento vývojový kit velmi kompaktní, není rentabilní používat ho v konkrétních aplikacích, ale pouze pro účely vývoje a testování.

Přípravu mikrokontroleru lze rozdělit do dvou kroků:

- Nahrání bootloaderu
- Nahrání samotného programu

I když není vždy nutné nahrávat do mikrokontroleru i bootloaderu, který navíc zabírá určitou paměť, je práce s mikrokontrolerem, který bootloaderu obsahuje mnohem jednodušší, proto je výhodné bootloaderu v MCU používat.[3][10]

### 2.2.2.1 Nahrání bootloaderu

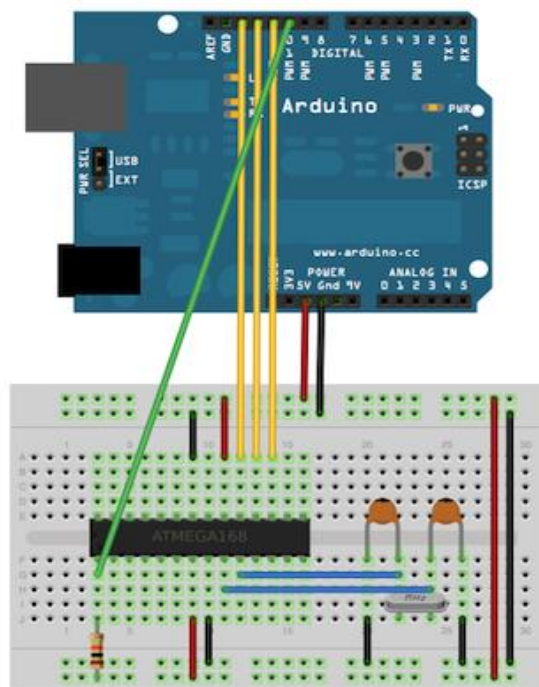
Bootloader, česky se někdy označuje jako zavaděč, je speciální program, který slouží k zavedení aplikace do paměti mikrokontroleru a její následné spuštění. Ve světě počítačů se stará o spuštění zavaděče BIOS (Basic Input-Output System) a příkladem zavaděče operačního systému může být například dnes využívaný Grub, či starší Lilo. Zavaděč se postará o zavedení operačního systému z příslušného média (pevný disk, či jeho určitý oddíl), který následně spustí. Ve světě mikrokontrolerů je situace velmi podobná. Po zavedení napětí na určené piny mikrokontroleru se jako první spustí bootloader. Ten je uložen na konci FLASH paměti. Bootloader je potom schopen nahrát, například prostřednictvím rozhraní USART, nový program do paměti MCU a následně skočí na jeho první instrukci.

Pro nahrání bootloaderu je nutné využít zapojení z obr. 9. Rezistor je zapojen na pin reset. Jeho hodnota je 10k $\Omega$ . Dále je použit externí 16MHz krystal a dva 18-22pF keramické kondenzátory.

Bootloader lze nahrát do mikrokontroleru v následujících krocích:

1. Nahrát program ArduinoISP (In-System Programming) do mikrokontroleru na vývojové desce. Je nutné vybrat vhodnou desku a rozhraní, pomocí kterého bude program nahrán do MCU.
2. Propojit vývojový kit a mikrokontroler dle obr. 9
3. Zvolit některou z následujících položek ve výběru vhodné desky:
  - a. Arduino Duemilanove
  - b. Nano w/ ATmega328
4. Nahrát bootloaderu do MCU prostřednictvím: Tools -> Burn Bootloader -> w/ Arduino as ISP.

Bootloader je nutné nahrát pouze jednou. Pokud budeme v budoucnu chtít přehrát hlavní program v mikrokontroleru, nemusíme bootloaderu znovu nahrávat. Pokud nahrání proběhlo úspěšně, kabely propojující piny 10, 11, 12, 13 můžeme odpojit.

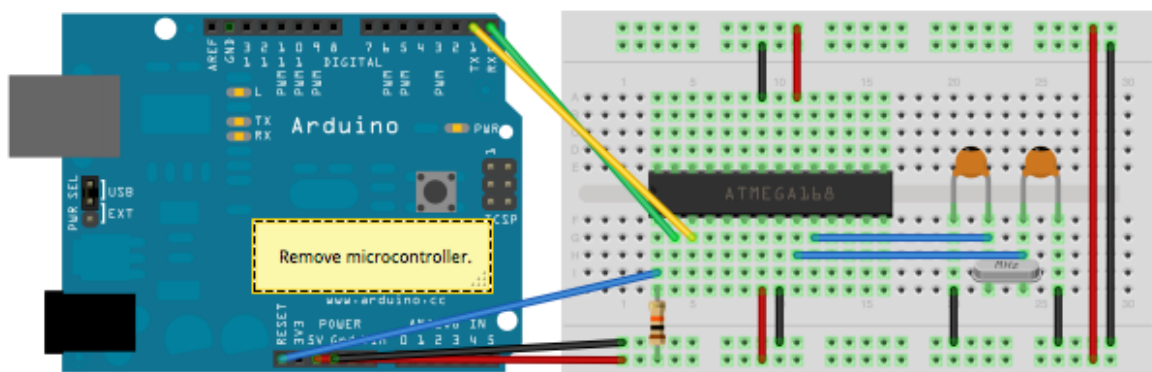


Obr. 9. Nahrání bootloaderu do mikrokontroleru

Teoreticky není nutné používat externí oscilátor, ale MCU bude potom schopen pracovat pouze na 8MHz. Pro jednodušší aplikace je tato frekvence ale pořád dostačující. Procesor v této konfiguraci dokáže zpravovat až 8 milionů operací za 1 sekundu.[10]

### 2.2.2.2 Nahrání řídicího programu do samostatného MCU

Nahráním bootloaderu do mikrokontroleru pouze tuto součástku připravíme pro její budoucí programování. Dalším krokem je tedy nahrání řídicího programu. Zapojení mikrokontroleru a vývojového kitu je zobrazeno na obr. 10. Schéma zůstává prakticky totožné jako v předchozím případě, změníme pouze způsob komunikace a co je nejpodstatnější – je nutné odstranit mikrokontroler z vývojového kitu.



Obr. 10. Nahrání programu do mikrokontroleru

Program se do mikrokontroleru nahraje pomocí převodníku USB – serial, který representuje integrovaný obvod FTDI na vývojovém kitu Arduino. Po tomto kroku je mikrokontroler připravený k zabudování do samostatného obvodu. MCU je vhodné umístit na patici pro snadné vyjmutí a přeprogramování, což se může stávat relativně často.

Příprava MCU je poměrně jednoduchá. Důležité je nejprve nahrát bootloader, díky kterému je možné mikrokontroler snadno programovat. Následné nahrání programu je tedy velmi jednoduché a lze opakovat dle potřeby.

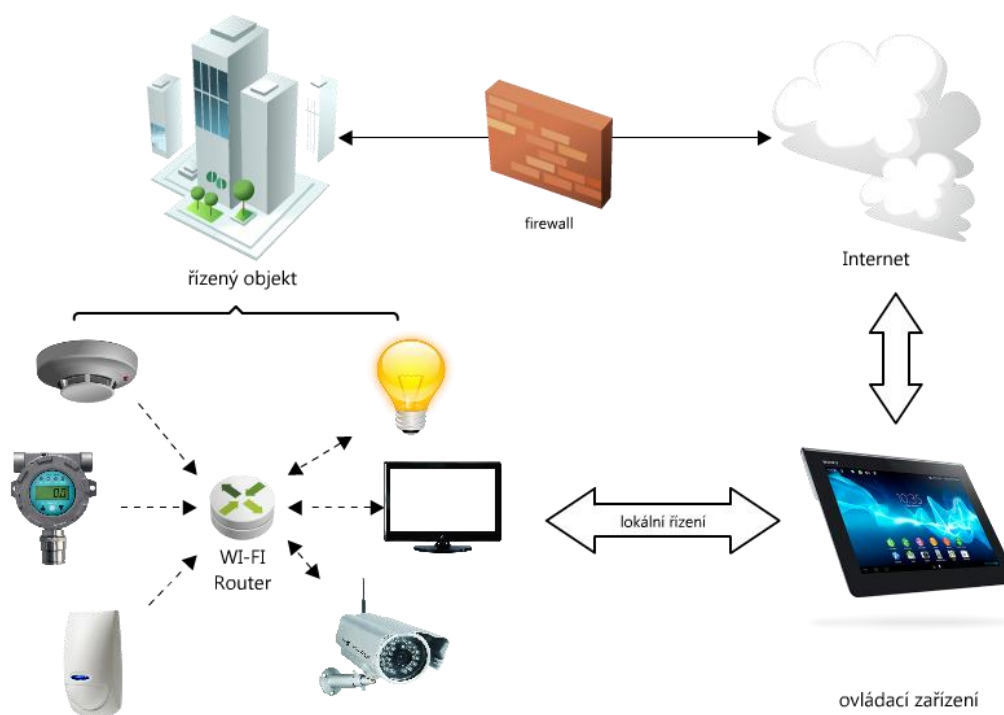
Při zabudování mikrokontroleru do samostatného obvodu musíme brát ohled také na to, jaký takt procesoru bude aplikace potřebovat. V případě náročných operací je nutné počítat také s připojením RC oscilátoru, díky kterému může MCU pracovat až na 16MHz.[10]

## **II. PRAKTICKÁ ČÁST**

### 3 VLASTNÍ ŘEŠENÍ

#### 3.1 Struktura navrhovaného řídicího systému

Cílem práce bylo navrhnout centralizovaný řídicí systém pro objekt, který bude možné jednoduše a levně vyrobit a zároveň bude schopen konkurovat, co se možností týče, současným řešením. Z těchto důvodů bylo zvoleno využití TCP/IP protokolu, prostřednictvím kterého mezi sebou jednotlivé periferie komunikují a díky kterému je zaručena kompatibilita s moderními PC sítěmi a dalšími zařízeními z oblasti IT (mobilní telefony, notebooky), kterými lze celý systém pohodlně ovládat. Budoucí uživatelé už vlastně disponují veškerým potřebným hardwarem, kromě samotných periférií určených pro měření fyzikálních veličin a řízení procesů, spojených s provozem budovy (takovéto periferie označujeme jako detektory a aktory). Určité formy řízení se v současnosti uplatňují snad v každém objektu, ať už jde o rodinný dům, či firemní budovu. Nejčastěji se jedná o regulaci teploty, provoz zabezpečovacího systému, jednoduché řízení osvětlení, atd. Ve výsledku pak máme v objektu množství, na sobě nezávislých, subsystémů. Jejich monitorování a řízení je možné přes samostatné terminály vyrobené pro tyto specifické účely.



Obr. 11. Struktura navrhovaného řídicího systému

V následujících kapitolách je popsán jeden z možných způsobů, jak lze veškeré subsystemy v objektu integrovat do jednoho centrálního systému a jeho prostřednictvím je monitorovat a ovládat. Z obr. 11 je patrné, jak navrhovaný systém koncipován. Řízený objekt je vybaven množstvím senzorů, které mohou poskytovat informace o stavu měřených veličin – teplotu, pohyb v prostoru, optické jevy spojené s vznikajícím požárem, nadměrné množství CO<sub>2</sub> a mnoho dalšího. Další předností je již zmíněná integrace jednotlivých subsystemů, které na obrázku výše reprezentuje subsystem požárních detektorů, subsystem osvětlení, bezpečnostní a kamerový subsystem, atd. Centrální řídicí systém budovy má tak přístup ke všem potřebným informacím, na základě kterých může automatizovaně rozhodovat a řídit samostatně kterýkoliv prvek. Pro inspiraci je uveden následující seznam příkladů, ale možnosti jsou prakticky neomezené a v reálném systému by si uživatel tyto pravidla a závislosti vytvářel na základě aktuálních potřeb.

- Automatické otevření oken z důvodu nadměrné koncentrace CO<sub>2</sub> (sepnutí větrání, klimatizace)
- Automatické natočení kamer na místo kde vznikl poplach
- Automatická regulace teploty v jednotlivých místnostech na základě dat z teplotních čidel
- Sepnutí výstražného osvětlení v případě požáru
- Uzamčení samostatných místností v důsledku narušení objektu
- Automatické ovládání rolet a žaluzií
- Řízení robotických systémů pro úklid prostorů

Z obr. 11 dále vyplývá, že jsou dvě možné varianty, kterými uživatel může objekt monitorovat a ovládat. Jedná se o vzdálené a lokální řízení. Lokální řízení probíhá v rámci objektu prostřednictvím stejné sítě, která sdružuje detektory a akční prvky. Vzdálené řízení se logicky nabízí díky TCP/IP protokolu, který je na tyto účely prakticky určen. Situace je velmi podobná vzdálené správě serverů.[11]

## 3.2 Realizované řešení

Pro reprezentaci dílčích prvků systému, znázorněného na obr. 11 byla zvolena stavba tří zařízení, které budou komunikovat s nadřazeným systémem prostřednictvím bezdrátové WI-FI sítě, nebo rozhraní ethernet. Tato zařízení jsou stručně popsána níže.[11]

### 3.2.1 Detektor

Zařízení, které detekuje změny fyzikálních veličin v určitém prostoru. Změna je vyhodnocena v MCU a odeslána k řídicímu systému, kde je zobrazena uživateli v grafické podobě.[11]

### 3.2.2 WI-FI Robot

Robot reprezentuje jakýkoliv typ akčního prvku, který ke své primární funkci využívá systém pohonných jednotek. Pod takovými prvky si lze představit například následující typy aplikací:

- otevírání/zavírání oken
- řízení žaluzií/rolet
- řízení ventilů
- robotické vysavače a servisní roboty, atd.

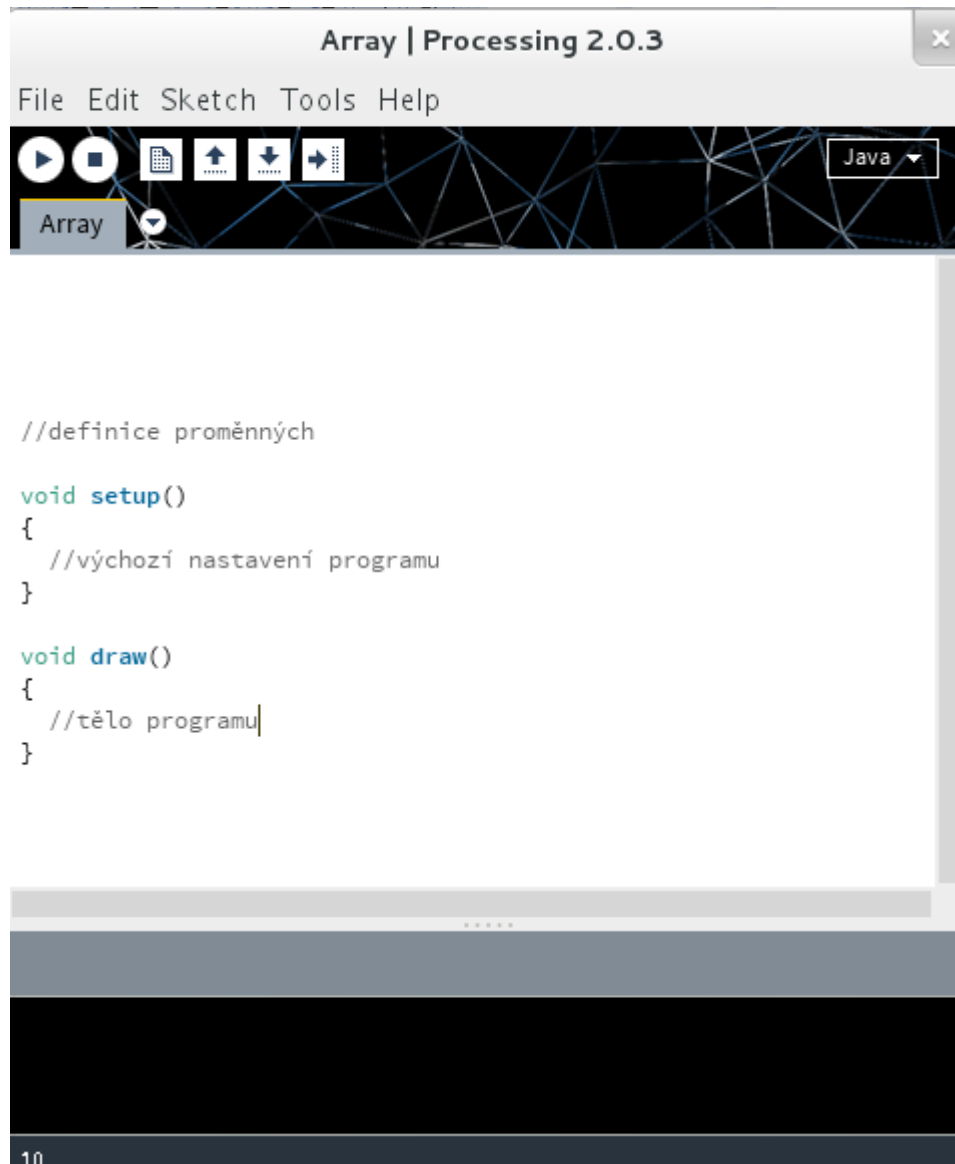
Řízení a monitorování robota probíhá bezdrátově, pomocí WI-FI sítě. Řídicí aplikace posílá do řídicí jednotky robota strukturované data, která obsahují informace o změně otáček jednotlivých motorů. Komunikace směrem k robotovi probíhá pouze tehdy, pokud dojde ke změně trajektorie. Řídicí jednotka zasílá data k řídicímu systému, pokud je detekována překážka před robotem.[11]

### 3.2.3 AC Dimmer

Dimmer pracuje se střídavým napětím 230V. Jeho funkcí je regulovat intenzitu osvětlení, na základě nastavení uživatele. Intenzitu je možné nastavit jak interně, pomocí potenciometru na zařízení, tak bezdrátově, prostřednictvím WI-FI sítě. Dimmer dále vysílá k nadřazenému systému informace o svém nastavení. Pokud tedy dojde ke změně interního nastavení, projeví se aktualizované nastavení i v řídicím systému.[11]

## 4 ŘÍZENÍ MIKROKONTROLERU

Pro programování řídicích aplikací byl využit programovací jazyk Processing. Tento jazyk je open-source a disponuje vlastním IDE, které je patrné na obr. 12. Slouží převážně pro práci s mikrokontrolery a dále práci s mediálními soubory a aplikacemi.



Obr. 12. IDE programovacího SW Processing

Základní struktura programu je na první pohled velmi podobná programu v jazyce Arduino. Nejprve definujeme proměnné a dodatečné knihovny. Ve funkci *setup*, podobně jako v předchozím případě, definujeme výchozí nastavení programu. Místo funkce *loop* je zde použita funkce *draw*, kde píšeme tělo programu. Tento jazyk disponuje množstvím knihoven, které výrazně zjednoduší realizaci výsledné aplikace. Můžeme tedy velmi

jednoduše komunikovat například se sériovým rozhraním počítače a připojeného mikrokontroleru, ale také pomocí technologie TCP/IP. Obojí se liší pouze definicí komunikačního rozhraní. V případě USB lze použít následující definici:

- `import processing.serial.*;`
- `Serial myPort;`

Ve funkci *setup* nastavíme následující parametry:

- `String portName = Serial.list()[0];`
- `myPort = new Serial(this, portName, 9600);`

Do proměnné *portName* se uloží aktivní USB spojení. V systému Linux například `/dev/ttyACM0`, které je následně použito v konstruktoru třídy *Serial*. Její instanci máme uloženou v proměnné *myPort*, s kterou můžeme v programu pohodlně pracovat a využívat ji například k zprostředkování přenosu informací, který je patrný z následující funkce *serialEvent*.

- ```
void serialEvent(Serial myPort)
{
    inByte = myPort.read();
}
```

V proměnné *inByte* je uložena informace, kterou se v případě úspěch podařilo načíst z aktivního USB rozhraní.

Výše zmíněné příkazy jsou základem pro sestavení jakékoliv komunikace počítače s externím mikrokontrolerem. Pokud se rozhodneme namísto USB využít například ethernetové rozhraní, pouze přidáme relevantní knihovnu a vytvoříme instanci následujícím postupem.

- `Client myClient;`

Ve funkci *setup* nastavíme následující parametry:

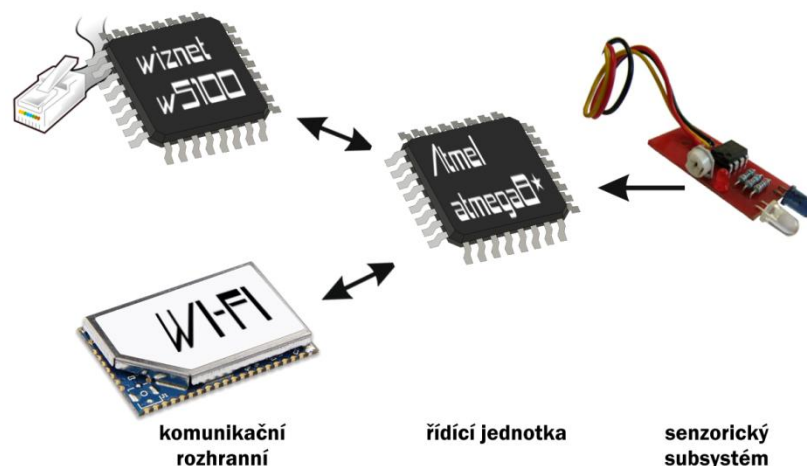
- `myClient = new Client(this, "20.0.0.3", 2500);`

V těle programu se budeme dotazovat instance *myClient*, na které je dostupné připojení s klientskou aplikací.

S využitím integrovaných knihoven je vytvoření spojení mezi MCU a počítačem poměrně snadné a pohodlné. Při výběru rozhraní se definice liší pouze nepatrně.[12]

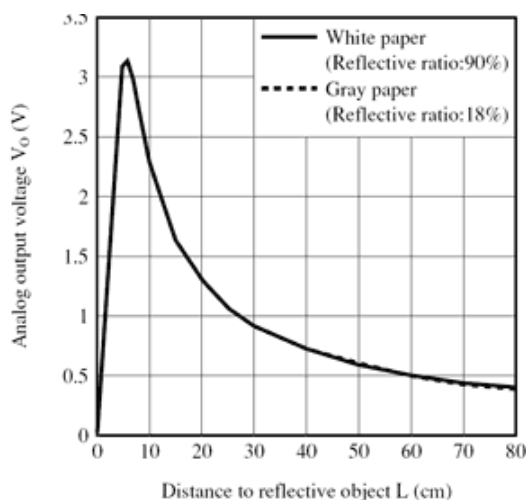
## 5 SUBSYSTEM DETEKTORŮ

Základní prvek výše popsaného systému představuje detektor. Slouží k měření fyzikálních veličin v zájmovém prostoru a disponuje vlastní vyhodnocovací jednotkou, sensorickým subsystémem (ten může představovat například PIR, MV senzor, dále CO<sub>2</sub> či různé optické senzory) a rozhraním pro přenos informace do společné sítě, který je možné zprostředkovat jak drátově, tak bezdrátově prostřednictvím WI-FI technologie.



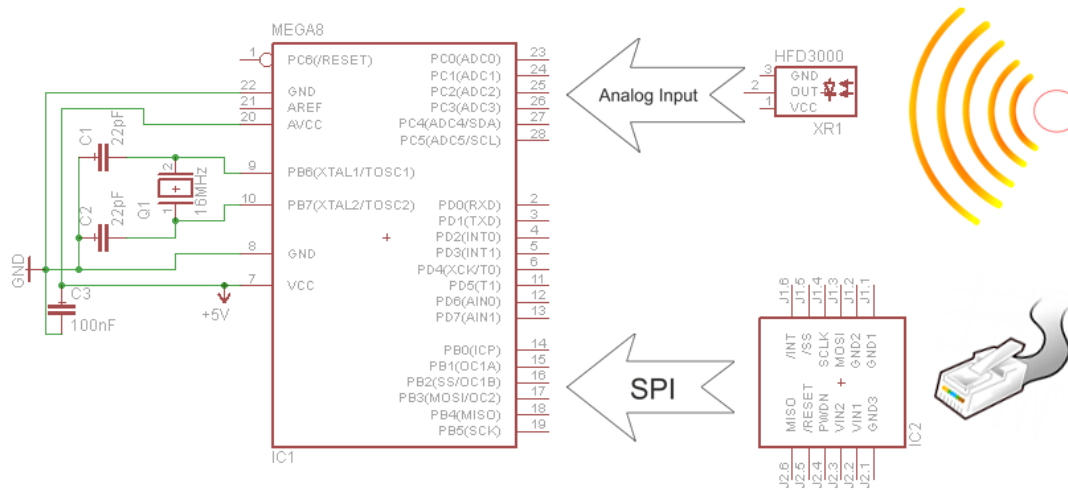
Obr. 13. Blokové schéma obecného detektoru

Detekční část představuje senzor Sharp GP2Y0A21, který umožňuje měření vzdálenosti na základě časového rozdílu mezi přijatou a odraženou vlnou. Vzdálenost je reprezentována změnou napětí na výstupu senzoru dle funkce znázorněné na obr. 14. Z grafu jsou dále patrné detekční možnosti. Senzor je schopen měřit vzdálenost objektu od ~7cm do 70 – 80cm. Maximální vzdálenost je vyjádřena nejnižším napětím (cca 0,4V).



Obr. 14. Charakteristika senzoru GP2Y0A21

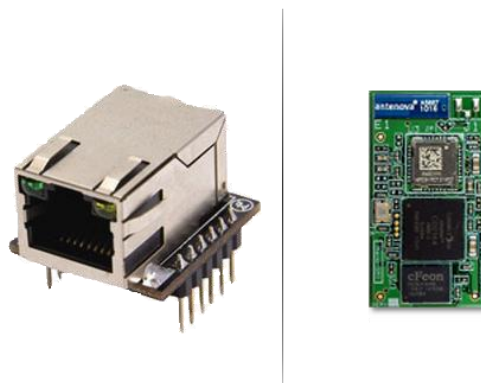
Změnu napětí vyhodnocuje MCU Atmel ATmega8, disponující šesti vstupy vybavenými A/D převodníkem (piny 23-28 na obr. 15), ke kterým je možné zapojit jeden či více různorodých senzorů s analogovým výstupem.[11]



Obr. 15. Návrh IR detektoru s rozhrním Ethernet

### 5.1.1 Komunikace

Komunikaci lze zprostředkovat mnoha způsoby. Využití TCP/IP protokolu není podmínkou. Stejně tak je možné využít technologie Bluetooth, nebo ZigBee, či jednoduchý digitální rádiový přenos. Pokud ale požadujeme, aby bylo s detektory a ostatními prvky možné komunikovat prostřednictvím standardního domácího routeru, musíme využít technologie WI-FI, nebo ethernet. Moduly, které můžeme k těmto účelům pohodlně využít, jsou na obr. 16.



Obr. 16. Moduly WIZ820io | Nano WiReach™ SMT

Ethernetovou vrstvu představuje modul od firmy Wiznet, WIZ820io. Komunikuje s MCU pomocí SPI (Serial Peripheral Interface) rozhraní. Výrobce poskytuje knihovny, pro komunikaci s výše zmíněným modulem, což podstatně zjednodušuje práci s tímto obvodem.

WI-FI komunikaci je možno realizovat například pomocí modulu Nano WiReach™ SMT od společnosti ConnectOne. S mikrokontrolerem, na rozdíl od ethernetového modulu, komunikuje prostřednictvím sériové komunikace (USART). Modul tedy představuje určitý převodník sériová komunikace -> WI-FI. Datová propustnost modulu činí 3Mbps.

Příklad programu s použitím poskytované knihovny je znázorněn v tab. 1. V konfiguraci klienta je nutné nastavit MAC adresu zařízení, jeho IP adresu a adresu serveru, ke kterému se detektor připojí. Při spuštění programu nastartuje komunikace se zadanou IP a MAC a provede se připojení k serveru na 23. portu (telnet). Program funguje v nekonečné smyčce – načte se hodnota z analogového pinu 23, kde je k MCU připojen výstup IR senzoru a v případě, že je dostupné spojení se serverem, zapíše naměřenou hodnotu na server.[9][11]

*Tab. 1. Příklad programu v MCU pro IR detektor - ethernet*

| Konfigurace                                                                                                                                                        | Nastavení programu                                                                        | Program                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <pre>byte mac[]={0xDE,0xAD,0xBE, 0xEF, 0xFE, 0xED }; char server[] = {10,0,0,2}; IPAddress ip(10,0,0,17); EthernetClient client; const int analogInPin = A0;</pre> | <pre>if (!Ethernet.begin(mac)) Ethernet.begin(mac, ip); client.connect(server, 23);</pre> | <pre>sensorValue = analogRead(analogInPin); if (client.available()) client.println(sensorValue);</pre> |

## 6 AKČNÍ ČLENY

Aktory se v síti prezentují (stejně jako v případě IP detektorů) jako klientské zařízení s rozdílem, že komunikaci je nutno řešit duplexně. Aktory lze tedy nejen monitorovat ale také řídit. Pro reprezentaci aktoru byla zvolena stavba WI-FI robota s IR detektorem, popsaným v předchozí kapitole.[11]

### 6.1 Aktor - testovací robot

#### 6.1.1 Konstrukce

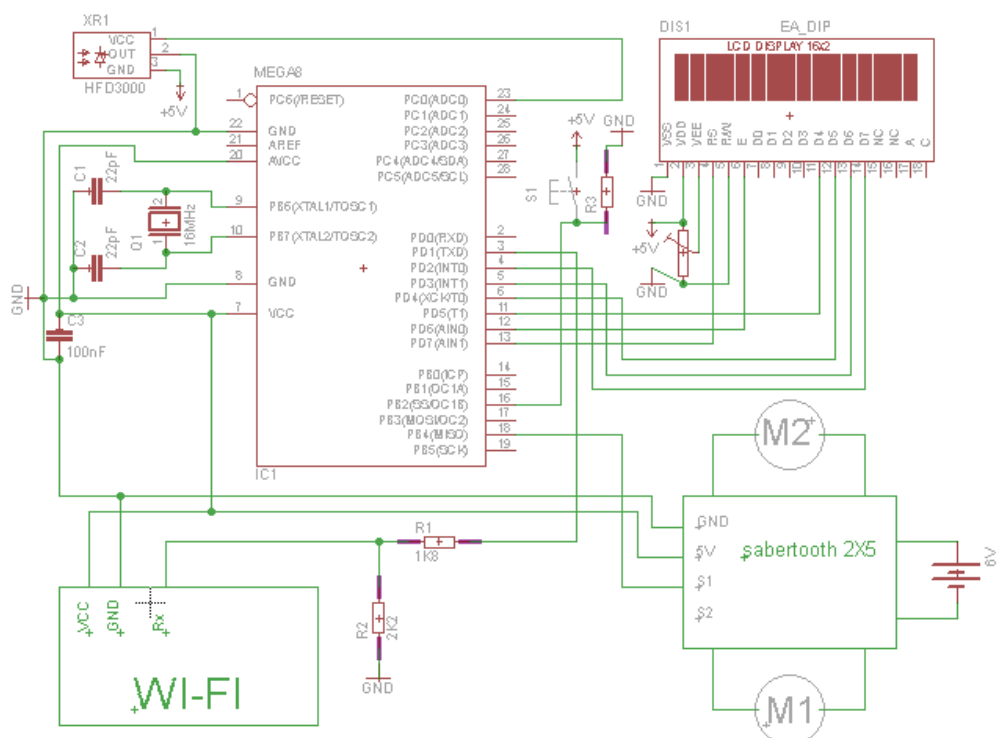


Obr. 17. Akční člen → WI-FI robot

Po konstrukční stránce se jedná o diferenčně řízeného robota. Tento způsob řízení se využívá převážně v průmyslových odvětvích. Výhodou jsou například manévrovací schopnosti, které přináší, ale jsou s ním spojené také určité problémy. Diferenční řízení je složitější především kvůli nutnosti sledování zpětné vazby otáček rotorů. V jistých případech může nastat stav, kdy působí různě velký odpor na jednotlivá kola – rozdíl rychlosti kol způsobí vychýlení robota z určené trajektorie. Pokud tomuto jevu chceme předejít, je nutné sledovat okamžitou úhlovou rychlost každého kola, což umožní výpočet aktuálních otáček. Rozdíl, který díky tomuto jevu nastane, je pak nutné připočítat ke kolu, na které působí větší odpor a úměrně zvýšit jeho otáčky.[1][11]

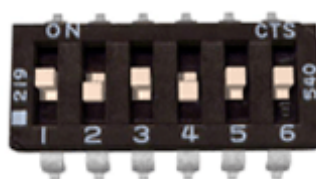
### 6.1.2 Zapojení

Na obr. 18 je podrobné zapojení WI-FI robota. Řídicí jednotku představuje stejný MCU, jako v předchozím případě. Kromě senzoru a připojení komunikačního rozhraní, je dále použit LCD (liquid crystal display) displej 2x16 znaků, který slouží k zobrazení informací o připojené WI-FI síti a dále stavu otáček motorů robota a jeho senzorů. Displej komunikuje s MCU pomocí sériového rozhraní. K dispozici jsou také knihovny, díky kterým je možné displej velmi snadno používat.



Obr. 18. Schematické zapojení WI-FI robota

Pohonnou jednotku představuje digitální regulátor motorů Sabertooth 2x5A. Komunikuje s MCU prostřednictvím sériové komunikace a rovněž výrobce volně poskytuje knihovny. Modul disponuje poměrně rozsáhlými možnostmi nastavení. V tomto případě je využita následující konfigurace (obr. 19).



Obr. 19. Konfigurace Sabertooth 2x5A

- Analogové napětí – ke vstupům S1 a S2 připojíme potenciometry jako děliče napětí a můžeme jimi řídit otáčky motorů
- Servopuls 1-2 ms – využívá stejný způsob řízení jako modelářské servomotory, nebo regulátory, každých 20 ms se opakuje řídicí puls, jehož délka určuje otáčky motorů. Je to velmi užitečné v případě, že chceme libovolné vozítko řídit přímo pomocí RC soupravy nebo prostě jen vyzkoušet mechanické a jízdní vlastnosti.
- Sériové rozhraní – sériová komunikace USART mezi řídicím MCU a tímto modulem.

Program aplikace v mikrokontroleru, který řídí pohyb robota, je zobrazen níže. Aplikace je kompatibilní s modulem WiFi shield, který zajistí připojení k WI-FI síti. Nutné je využití knihoven:

- SoftwareSerial.h – sériová komunikace s regulátorem motorů
- SabertoothSimplified.h – knihovna pro práci s regulátorem
- SPI.h a WiFi.h – knihovny pro komunikaci s WI-FI modulem
- LiquidCrystal.h – práce s displejem

Dále je nutné deklarovat proměnné. Důležité hodnoty jsou popsány v zdrojovém kódu. Jedná se o nastavení LCD, evidenční údaje pro připojení k síti a proměnné pro regulátor motorů.

```
#include <SoftwareSerial.h>
#include <SabertoothSimplified.h>
#include <SPI.h>
#include <WiFi.h>
#include <LiquidCrystal.h>
.
// deklarace proměnných
.
LiquidCrystal lcd(27, 26, 25, 24, 23, 22); // komunikace s LCD displejem

char ssid[] = "robot"; // SSID síť
char pass[] = "*****"; // heslo (typ WPA, nebo WEP)
IPAddress ip(20, 0, 0, 3); // IP adresa robota

WiFiServer ServIN(2500); // vytvoří připojení k serveru na daném portu
.
.
.
SoftwareSerial SWSerial(20, 14); // komunikace s regulátorem motorů
SabertoothSimplified ST(SWSerial);
.
```

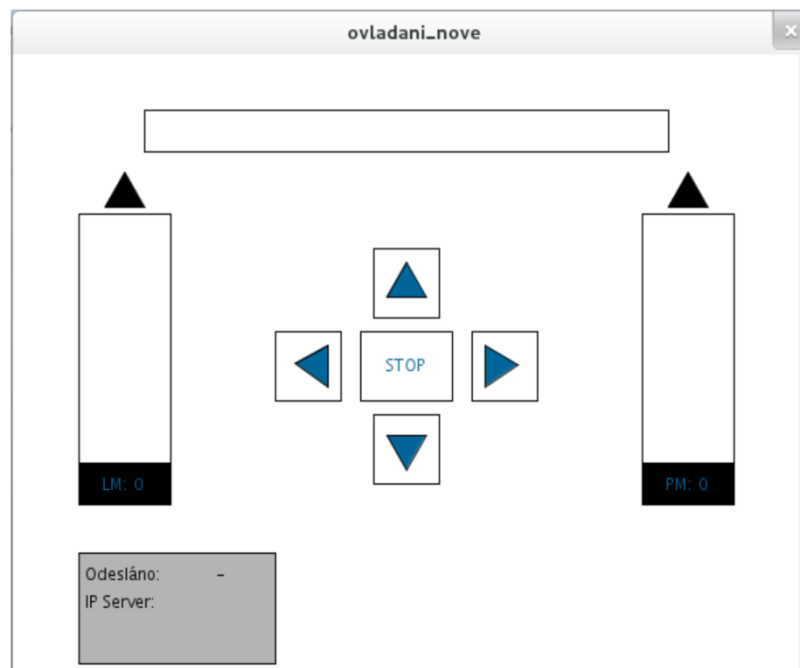
```
.  
void setup() {  
    SWSerial.begin(9600);  
    lcd.begin(16, 2);  
    Serial.begin(9600);  
    pinMode(led, OUTPUT);  
    if (WiFi.status() == WL_NO_SHIELD) {  
        Serial.println("WiFi shield not present");  
        while(true);  
    }  
    WiFi.config(ip);  
    while ( status != WL_CONNECTED) {  
        status = WiFi.begin(ssid, pass);  
        delay(10000);  
    }  
    ServIN.begin();  
    printWifiStatus();  
}  
  
void loop() {  
    WiFiClient client = ServIN.available();  
    if (client) {  
        if (!alreadyConnected) {  
            client.flush(); // smazání bufferu  
            alreadyConnected = true;  
        }  
        if (client.available() > 0) {  
            String in = "";  
            balik[i] = char(client.read());  
        }  
        for (int i = 0; i<3; i++) {  
            LmotS[i] = balik[i];  
            RmotS[i] = balik[(i+3)];  
        }  
        int Lmot = atoi(LmotS);  
        if ((Lmot < 64) & (Lmot > -64))  
            ST.motor(2, Lmot);  
        else  
            Lmot = 64;  
        int Rmot = atoi(RmotS);  
        if ((Rmot < 64) & (Rmot > -64))  
            ST.motor(1, -(Rmot*4));  
        else  
            Rmot = 64;  
    }  
    else {  
        client.stop();  
        indikace = 0;  
        ST.motor(2, 0);  
        ST.motor(1, 0);  
    }  
}
```

```
}  
  
void printWifiStatus() {  
.  
    // vytvoří údaje o stavu připojení (kvalita signálu, SSID, IP, PORT, MAC...)  
.  
}  
}
```

Funkce *look()* provádí hlavní funkci programu. V nekonečném cyklu se dotazuje síťového rozhraní, z kterého čte, nebo na které zapisuje data. Nejdříve tedy vytvoříme instanci objektu *client*, v které je dostupné připojení k serveru. Na této instanci následně čteme či zapisujeme potřebná data.

Hodnoty, které obdržíme pomocí síťového rozhraní, je nutné transformovat do tvaru, kterému bude rozumět MCU robota. Data z řídicí aplikace je nutné transformovat do podoby, která se co nejpohodlněji přenáší přes síť. V tomto případě byla zvolena podoba string, do které se data transformují před odesláním. V MCU robota dochází k zpětné transformaci string hodnot na celočíselný tvar a dále k dělení dat pro levý a pravý motor.[1][6][11][13]

### 6.1.3 Řídicí aplikace



Obr. 20. Řídicí software WI-FI robota

Zkrácený zápis zdrojového kódu aplikace, který je vypsán níže poukazuje, jak program funguje. Pro práci se síťovým rozhraním je nutné využít knihovnu `processing.net.*`, z které si následně vytvoříme instanci `myClient`. Zde bude uloženo připojení na definované adrese a příslušném portu. V tomto případě `20.0.0.3:2500`.

Aplikace (obr. 20) se bezprostředně po spuštění pokusí vytvořit spojení mezi robotem a serverem. V případě, že proběhlo připojení v pořádku, umožní ovládání příslušného zařízení. Funkce `keyPressed()`, `indikace()` a `zapis()` jsou obslužné funkce programu a jejich účel je stručně popsán ve zdrojovém kódu níže.

```
import processing.net.*;
Client myClient;
.
//deklarace proměnných
.
void setup() {
    size(600,450);
    myClient = new Client(this, "20.0.0.3", 2500);
}

void keyPressed(){
.
    // funkce slouží k odchyťování stisknutých kláves
.
}
void indikace(int A, int B, int S) {
.
    // Zobrazení stavu motorů
.
}
void zapis(int L, int R) {
    // zápis proměnných na server
    myClient.write(LMS + PMS);
}

// Hlavní funkce programu:
void draw() {
    update(mouseX, mouseY);
    objekty();
    int x1;
    switch(result) {
        case NORTH:
            fill(0);
            rect(275,140,50,50);
            fill(0, 102, 153);
            triangle(285, 175, 315, 175, 300, 150);
```

```
        fill(0);
        if ((y1 < 476) & (y1 > 115))
            if (Lmot < Pmot)
                y1 = y2--;
            else
                y2 = y1--;
        Lmot = Pmot = round(map(y1, 295, 115, 0, 64));
        zapis(Lmot,Pmot);
        break;

    case SOUTH:
        fill(0);
        rect(275,260,50,50);
        fill(0, 102, 153);
        triangle(285, 275, 315, 275, 300, 300);
        fill(0);

        if ((y1 < 475) & (y1 > 114))
            if (Lmot < Pmot)
                y2 = y1++;
            else
                y1 = y2++;
        Lmot = Pmot = round(map(y1, 295, 115, 0, 64)); break;
    case STOP:
        fill(0);
        rect(265,200,70,50);
        fill(0, 102, 153);
        text("STOP", 285, 229);
        fill(0);
        y1 = 295;
        y2 = 295;
        Lmot = Pmot = 0;
        zapis (Lmot,Pmot);
    break;

    case EAST:
        fill(0);
        rect(350,200,50,50);
        fill(0, 102, 153);
        triangle(215, 223, 240, 210, 240, 240); // 3.1
        fill(0);
        if ((y2 < 476) & (y2 > 114))
            if ((Pmot >= Lmot) & ((Lmot & Pmot) < 0))
                y2--;
            else
                y2++;
        Pmot = round(map(y2, 295, 115, 0, 64));
        zapis (Lmot,Pmot);
    break;
```

```

    case WEST:
        fill(0);
        rect(200,200,50,50);
        fill(0, 102, 153);

        triangle(385, 224, 360, 210, 360, 240);
        fill(0);
        if ((y1 < 476) & (y1 > 114))
            if ((Pmot <= Lmot) & ((Lmot & Pmot) < 0))
                y1--;
            else
                y1++;
        Lmot = round(map(y1, 295, 115, 0, 64));
        zapis (Lmot,Pmot);
    break;
}
indikace(Lmot,Pmot,result);
delay(10);
}

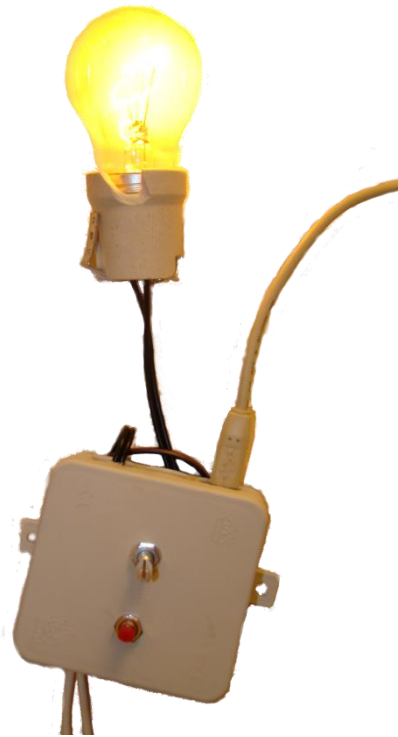
```

Hlavní funkce programu - *draw()*, určí na základě nastavovaných parametrů směr robota a rychlost otáček obou motorů. Dále se zde volají ostatní obslužné funkce, nezbytné pro běh aplikace, jako například funkce *zapis()*, která se volá při každém zmáčknutí příslušné klávesy. Aby nedocházelo k nastavení hodnot, které by způsobily například rozdílnou rotaci obou motorů, jsou využity podmínky, které ošetří vstup uživatele a zabrání zadání nepřístupných hodnot. Na síťový port se tedy zapisuje pouze validní hodnota.[11]

## 6.2 Aktor – AC dimmer

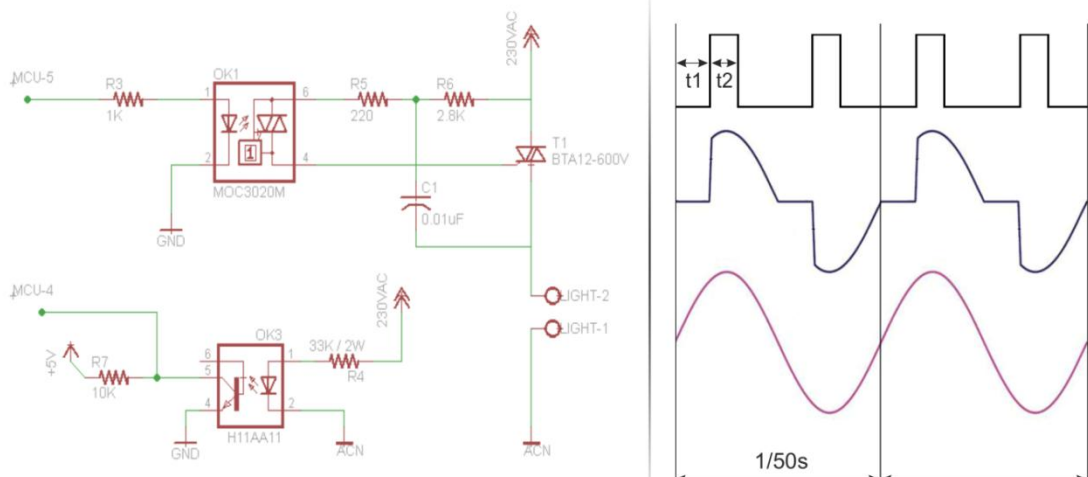
Dimmer - neboli tlumič světla (obr. 21), je zařízení, které reguluje intenzitu jasu osvětlení. Presentovaný výrobek využívá k tomuto efektu technologii zvanou fázové řízení. Prakticky se používá například pro následující aplikace:

- regulace osvětlení,
- řízení otáček indukčních motorů,
- řízení elektrických odporových pecí a topných těles.



Obr. 21. Aktor -> AC Dimmer

Obvod byl testován s maximální zátěží 1kW, při které stále nebylo nutné triak nijak chladit. Výrobce udává možnost zátěže až do 12A 600V, při použití výkonnějších triků dokonce podstatně více. Na obr. 22 je znázorněno, jak tato technologie funguje.



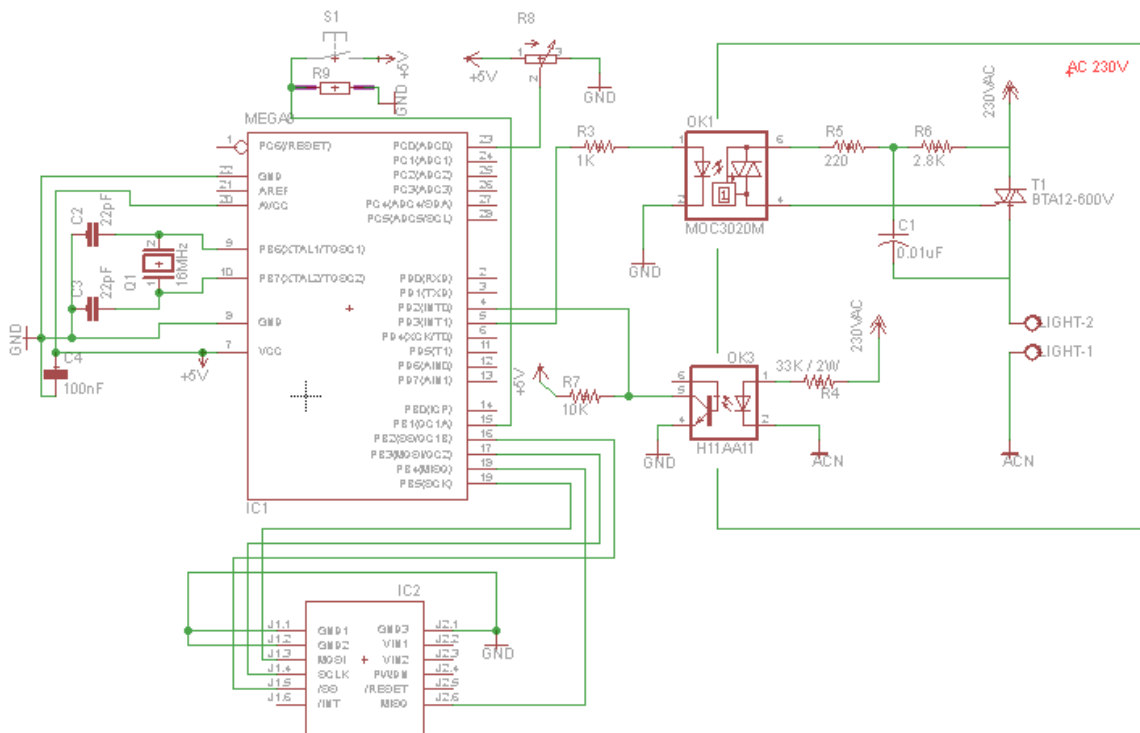
Obr. 22. Schéma AC dimmeru 230V | fázové řízení

Triak BTA12 je spínán optorelátkem MOC3020, které je připojeno k MCU přes rezistor R3. MOC3020 funguje zároveň jako galvanický oddělovač výkonového obvodu od

nízkonapěťového, do kterého se za žádných okolností nemůže dostat fázové napětí. Pro docílení regulace osvětlení je nutné synchronizovat frekvenci mikrokontroleru s frekvencí v napájecí síti (v ČR 50Hz). Toho docílíme pomocným obvodem pro zjištění průchodů nulou (tzv. Zero Crossing detekce), který představuje optotranzistor H11AA1. Každý průchod fáze osou x při nulovém napětí (nastává 2x za každou periodu) vyvolá impuls, který je zaznamenán mikrokontrolerem. Triak zůstává po každém průchodu nulou vypnutý po stanovenou dobu - na obr 22. označena jako t1. Křivka představuje napětí na bráně triaku a udává, kdy bude triak spínán a vypínán. Protože triak zůstává sepnutý i po odpojení napětí z brány, musíme nastavit vhodně dobu t2, aby bylo napětí na bráně nulové až do příštího průchodu fáze nulou, kdy se triak opět vypne a je odstartován čas t1.

Software v MCU využívá k přesnému řízení průběhu napětí na bráně triaku časovač a přerušení (tzv. interrupts). Jedná se o funkci procesoru, která umožní dočasně přerušit standartně prováděnou rutijní operaci, vyřešit novou úlohu a k předchozí rutině se opět vrátit. Tím docílíme určité obdoby multitaskingu, kterým více zmíněný MCU bohužel nepodporuje.[4][5][11]

### 6.2.1 Zapojení



Obr. 23. Schéma AC Dimmer 230V

Z obr. 23 je patrné, z čeho se zařízení skládá. Nejdůležitější částí je hlavní mikrokontroler ATMEGA8, který řídí výkonový obvod, zprostředkovává spojení se serverem a také interní řízení, které umožňuje ovládat obvod pomocí vestavěného potenciometru na zařízení (připojen na pin 23). Potenciometr nastavuje délku mezi spínacími pulsy pro výkonový obvod. Tato hodnota je přičtena k času, kdy byl detekován poslední průchod nulou (ZC). V případě, že hlavní MCU obdržel aktualizovanou hodnotu ze serveru, je hodnota z potenciometru ignorována, což je patrné ze zdrojového kódu níže. Tlačítko, které je připojeno na digitální port 15, spíná výkonový obvod. Dále je zakomponováno ethernetové rozhraní, které v tomto případě reprezentuje modul WIZ820io. Následující zdrojový kód v hlavním MCU popisuje, jak funguje časování a jakým způsobem je řízená výsledná intenzita světla.

Nutností je využít pouze knihovny pro práci s ethernetovou vrstvou (v tomto případě ethernet shield). Následně se definují konstanty, které reprezentují porty pro detekci průchodů nulou (ZC) a spínání triaku prostřednictvím optorelé MOC3020. Konstanty MAX a MIN level určují maximální přístupnou hodnotu pro nastavení intenzity.

```
#include <SPI.h>
#include <Ethernet.h>

#define DETECT 0 // interrupt 0 = pin 2
#define GATE_1 29 // triak gate

#define MAX_LEVEL 99
#define MIN_LEVEL 0
#define FADE_DELAY 15

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,0,102);
IPAddress server(192,168,0,101);
EthernetClient client;

char input[3];
.
.
.
// proměnnou nastavíme jako nestálou pro změnu jak, s ní bude kompilátor pracovat
volatile uint16_t dim_level, light_1 = MAX_LEVEL , light_2 = MIN_LEVEL;

boolean fade_down_1 = false;
boolean fade_down_2 = true;
boolean start_low_fix = true;
```

```
void setup() {
  Serial.begin(9600);
  Ethernet.begin(mac, ip);
  if (client.connect(server, 10002))
    Serial.println("connected");
  else
    Serial.println("connection failed");

  // Nastaví přerušeni z Pin 2 (interrupt 0) pro ZC na náběh
  attachInterrupt(DETECT, zero_cross_detect, RISING);

  // nastavení moc3020:
  pinMode(GATE_1, OUTPUT); // řízení brány triaku
  digitalWrite(GATE_1, LOW); // vypne triak

  // nastavení časovače
  cli();
  TCCR1A = 0x00; // clear
  TCCR1B = 0x00; // clear
  TIMSK1 = 0x02; // zapne compare na výstupu A
  OCR1A = 1250; // = 128 kroků na 50Hz bez násobiče
  sei();
}
ISR(TIMER1_COMPA_vect) {
  .
  // nastavuje intenzitu žárovky
  .
}
void zero_cross_detect() {
  TCNT1 = 0; // reset timer – počítání od nuly
  TCCR1B = 0x09; // start časovače bez násobiče a reset tcnt1
  start_low_fix = false;
  dim_level = 0;
}

void loop() {
  if (start_low_fix) // vyčká na první ZC
    return;

  sensorValue = analogRead(analogInPin);
  outputValue = map(sensorValue, 0, 1023, 0, 99);
  while((pom!=outputValue)&&(pom!=outputValue-1)){
    pom=outputValue;
    light_1=99-outputValue;
    if(outputValue< 10) all = "00" + String(outputValue);
    if((outputValue < 100) && (outputValue > 10)) all = "0" +
    string(outputValue);
    if((outputValue <= 128) && (outputValue >= 100)) all = "" +
    String(outputValue);
  }
}
```

```

        trideni(all);
        delay(10);
    }
    if (client.available()>0) {
        for (int i=0; i<3; i++) {
            if (client.available()>0) {
                input1[i] = client.read();
            }
            delay(10);
        }
        int out = atoi(input1);
        lcd.setCursor(5, 1);
        lcd.print(out);
        light_1=99-out;
    }
}

// transformace dat přijatých ze serveru
void trideni(String prom) {
    for (int i=0;i<3;i++) {
        input[i]=prom[i];
        client.print(input[i]);
    }
}

```

Nastavení časovače MCU ATMEGA8 je poměrně komplikované, ale všechny potřebná nastavení lze vyčíst s datasheetu součástky. Princip spočívá ve vyresetování časovače, při každém průchodu fáze nulou. Jak již bylo řečeno výše, tento jev je detekován pomocí obvodu H11AA1, který je přiveden na port 4 hlavního MCU. Díky resetu časovače dochází k synchronizaci frekvence mikrokontroleru s frekvencí v napájecí síti. Délku periody v napájecí spočítáme dle následujícího vzorce:

$$T = \frac{1}{f} = \frac{1}{50} = 0.02s \quad (1)$$

Z následující rovnice jsme získali délku jedné periody. Víme, že k průchodu fáze nulou dochází 2x za každou periodu. Délka jedné periody je tedy:

$$\frac{1}{T} = \frac{0.2}{2} = 0.1s \quad (2)$$

Časovač MCU funguje na frekvenci 16MHz, což je 16000000 cyklů za 1s. Stejným výpočtem zjistíme délku jednoho hodinového cyklu časovače:

$$T = \frac{1}{f} = \frac{1}{16000000} = 6.25 \times 10^{-8}s \quad (3)$$

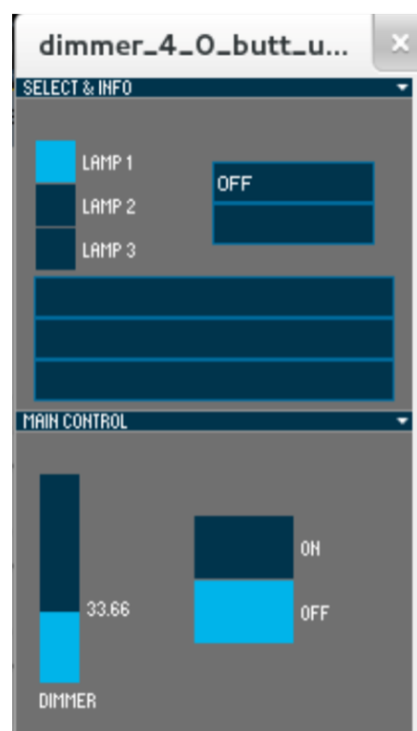
Porovnáním obou frekvencí získáme, kolik hodinových cyklů se vejde do ½ sinusové vlny.

$$\frac{0.1}{6.25 \times 10^{-8}} = 1600000 \text{ cyklů} \quad (4)$$

Všechny potřebné výpočty pro synchronizaci MCU jsou hotové. Následně sestavíme řídicí aplikaci popsanou níže.[11]

### 6.2.2 Řídicí aplikace

Řídicí program umožňuje nastavovat výslednou intenzitu osvětlení na základě posuvníku vlevo dole na obr. 24. Nastavená hodnota je převedena na string a odeslána ke koncovému zařízení. Dále je možné volit, kterou žárovku chceme ovládat, jelikož v praxi se setkáme s případy, kdy bude nutné nezávisle regulovat více světel. Dále aplikace obsahuje spínací tlačítko, které světlo vypne či zapne.



Obr. 24. Řídicí software pro AC dimmer

Dále nás tento program informuje o stavu připojení k síti a o hodnotách, které jsou odeslané, či byly přijaty z koncového zařízení. Ze zdrojového kódu níže je patrné, jak řídicí aplikace funguje.

V tomto případě je komunikace směřována na USB rozhraní. Pokud budeme chtít využít síťové rozhraní, stačí použít příslušnou knihovnu a definovat spojení dle pravidel, pro síťovou komunikaci, popsaných výše.

```
import processing.serial.*;
```

```
Serial serialcom;  
int port = 10002;
```

```
void setup() {  
    size(200, 320);  
    noStroke();  
    smooth();  
    gui();  
    String port = serialcom.list()[0];  
    serialcom = new Serial(this, port, 9600);  
}
```

```
void gui() {  
    // grafické rozhraní aplikace  
}
```

```
void draw() {  
    background(150);  
    light=round(dimmerer.getValue());  
    if ((preLight!=light) ){  
        preLight=light;  
        if(light< 10) all = "00" + (light);  
        if((light < 100) && (light > 10)) all = "0" + (light);  
        if((light <= 128) && (light >= 100)) all = "" + (light);  
        println(all);  
        time = millis();  
        pom=1;  
        serialcom.write(all);  
    }  
    if((millis() - time >= wait)&&pom!=0){  
        time = millis();  
        pom=0;  
    }  
    // příjem a transformace dat  
    if (serialcom.available()>0) {  
        inbite = serialcom.readChar();  
        if (inbite == '0')  
            for (int i=0;i<2;i++) {  
                if (serialcom.available()>0) input[i] = serialcom.readChar();  
            }  
  
        incom = ""+input[0]+input[1];  
        d = int(incom);  
    }  
}
```

```
        dimmerer.setValue(d);
    }
    switchStateLamp.activate(stav);
    delay(25);
}

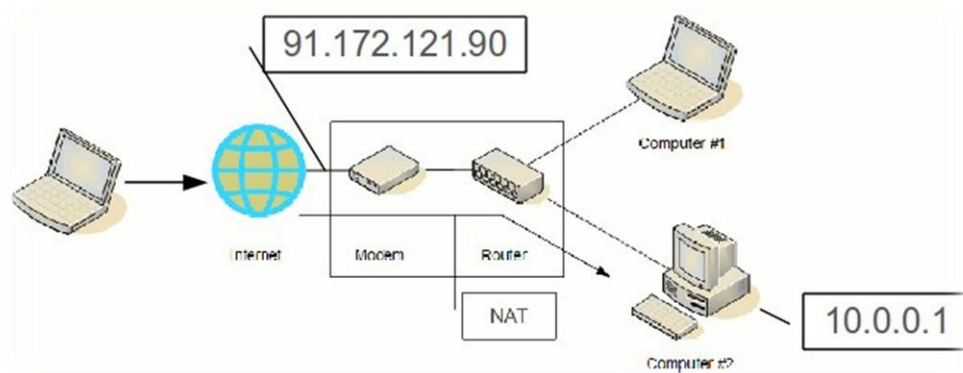
void switchLamp(int lamp) {
.
    // přepínání světel
.
}

void switchStateLamp(int state) {
.
    // spínání/vypínání žárovky
.
}
```

Aplikace je poměrně jednoduchá. Po vykreslení aktivních prvků, je v nekonečném cyklu odčítán stav přijímacího portu. Na tomto portu se nachází definice USB připojení ke koncovému zařízení. Data, která jsou načtena, je nutné transformovat z hodnoty string, která je výhodná pro komunikaci přes USB port na celočíselnou hodnotu, pomocí které aplikace indikuje stav světla. Přijaté bity se párují do trojic a aplikace takto strukturovaná data chápe, jako přijatou hodnotu, kterou v případě úspěšné validace nastaví na příslušném posuvníku. Je tedy indikován také stav nastavené hodnoty potenciometru na zařízení. Stejným způsobem je řešeno odesílání nastavených dat v řídicí aplikaci. Z nastavených hodnot vytvoříme trojice string hodnot, které následně odesíláme přes komunikační rozhraní ke koncovému zařízení, které takto strukturovaným datům rozumí.[11]

## 7 VZDÁLENÍ ŘÍZENÍ

Vzdálené řízení je navrženo podobným způsobem, jakým bychom v praxi řídili vzdálený počítač, či server. Zařízení, které nám zprostředkuje tuto funkci je standardní router s NAT (Network Address Translation). Jak tato technologie funguje, je patrné na obr. 25. Cílový počítač s řídicí aplikací (10.0.0.1) se na Internetu prezentuje veřejnou IP adresou routeru. Po připojení k této adrese z Internetu dojde k přesměrování na cílový počítač.



Obr. 25. Network Address Translation

NAT je funkcí routeru, prostřednictvím kterého se připojujeme do Internetu, tedy routeru na straně uživatele, ne providera. Pro uživatele nepředstavuje tedy žádný problém, aktivovat tuto funkci na svém routeru. Podmínkou je pouze veřejná IP adresa, kterou se router prezentuje na Internetu.

Princip NATu je poměrně jednoduchý:

- klient vyšle požadavek na bránu vnitřní sítě
- router pakety zachytí, změní jejich IP adresu na svou vnější
- router pakety označí tak, že je odešle z náhodného TCP portu
- router si do tabulky zapíše, který port zvolil a který klient k němu patří
- při přijetí odpovědi provede router reverzní akci a pakety vrátí klientovi

Konfigurace funkce NAT je patrná z obr. 26. Provádí se na routeru, pomocí kterého se připojujeme k Internetu prostřednictvím webového prohlížeče. Je nutné definovat službu, ke které se budeme chtít prostřednictvím NATu směřovat. Pokud není služba v seznamu

předdefinovaných, je nutné ji přidat manuálně, definováním portu a IP adresy cílového a koncového zařízení. V tomto případě je využit telnet protokol, který používá port 23 a směřuje se na adresu 10.0.0.1. K této adrese a tomuto portu budeme směřováni, pokud se pokusíme prostřednictvím Internetu připojit k adrese 91.172.121.19.

| Server Name               | External Port Start | External Port End | Protocol | Internal Port Start | Internal Port End | Server IP Address | WAN Interface | Enable                              | Remove                   | Edit |
|---------------------------|---------------------|-------------------|----------|---------------------|-------------------|-------------------|---------------|-------------------------------------|--------------------------|------|
| Secure Shell Server (SSH) | 22                  | 22                | TCP      | 22                  | 22                | 10.0.0.2          | ppp1          | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| http                      | 80                  | 80                | TCP      | 80                  | 80                | 10.0.0.15         | ppp1          | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| h                         | 8080                | 8080              | TCP      | 8080                | 8080              | 10.0.0.2          | ppp1          | <input type="checkbox"/>            | <input type="checkbox"/> | Edit |
| http                      | 80                  | 80                | TCP      | 80                  | 80                | 10.0.0.15         | ppp1          | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| Telnet Server             | 23                  | 23                | TCP      | 23                  | 23                | 10.0.0.1          | ppp1          | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| Secure Shell Server (SSH) | 22                  | 22                | TCP      | 22                  | 22                | 10.0.0.2          | ppp0.1        | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| http                      | 80                  | 80                | TCP      | 80                  | 80                | 10.0.0.2          | ppp0.1        | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| h                         | 8080                | 8080              | TCP      | 8080                | 8080              | 10.0.0.2          | ppp0.1        | <input type="checkbox"/>            | <input type="checkbox"/> | Edit |
| http                      | 80                  | 80                | TCP      | 80                  | 80                | 10.0.0.15         | ppp0.1        | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |
| Telnet Server             | 23                  | 23                | TCP      | 23                  | 23                | 10.0.0.1          | ppp0.1        | <input checked="" type="checkbox"/> | <input type="checkbox"/> | Edit |

Obr. 26. Tabulka NAT

Vzdálené řízení je možné na základě předchozího nastavení velmi snadno realizovat. Podmínkou je, aby serverová aplikace byla spuštěna na počítači s IP 10.0.0.1. Zde máme dostupné veškeré proměnné ze senzorů. Vzdálený klient se k aplikaci připojí pomocí telnet portu, podobně, jako se připojují senzory a akční členy.

Stinnou stránkou tohoto řešení je bezpečnost. Řízení přes Internet vyžaduje aplikaci co možná nejúčinnějších bezpečnostních prvků, jako je firewall, ale také bezpečný síťový protokol. Využitý protokol telnet je již poměrně zastaralý a jsou popsány způsoby, jak tento protokol napadnout. V další kapitole je popsán způsob, jak data procházející přes telnet protokol efektivně šifrovat pomocí známé asymetrické šifry RSA, což výrazně zvyšuje bezpečnost použitého protokolu.[14][15]

## 8 ZABEZPEČENÍ PŘENOSU

Šifrovací algoritmus lze navrhnout na základě šifry známé jako RSA (*Rivest, Shamir, Adleman*, tedy RSA). Je to v současnosti hojně využívaná šifra a její licence umožňuje zároveň volné využití jak pro firmy, tak běžné uživatele. Z těchto důvodů se stala součástí digitálních podpisů, internetového bankovníctví, mobilní komunikace, nebo například armádní sdělovací techniky. Vzhledem k využití méně výkonných MCU je v tomto případě šifra zjednodušena o náhodný výběr vhodných, velkých prvočísel. Je známo, že využití jakékoliv asymetrické šifry přináší vysoké nároky na výpočetní výkon. Ten lze zařízení zajistit osazením výkonnějšího MCU, například s mikroprocesorem typu ARM.

Po technické stránce lze použitý šifrovací algoritmus popsat následovně: Po volbě vhodných prvočísel se vypočítá tzv. modul  $n$  – je součinem těchto čísel a hodnota modulu Eulerovi funkce  $\varphi(n)$ . Na základě následující podmínky  $e < \varphi(n)$  je volen veřejný exponent  $e$ . Další podmínkou je absence společných dělitelů veřejného exponentu s modulem. Tímto postupem získáme veřejný klíč, dvojici  $(n, e)$  a soukromým klíč  $(n, d)$ , kde pro  $d$  (multiplikativní inverze), platí následující podmínka:

$$d^{-1} \equiv e \pmod{\varphi(n)} \quad (5)$$

Výpočet multiplikativní inverze je posledním krokem, vedoucím ke stanovení soukromého a veřejného klíče.

Šifrování jednotlivých znaků zprávy  $m$ , odesílaných komunikačním kanálem (soketem), je realizováno následujícím vzorcem:

$$x \equiv m^e \pmod{n} \quad (6)$$

Dešifrování pak analogicky vzorcem:

$$m \equiv x^d \pmod{n} \quad (7)$$

## ZÁVĚR

Prezentovaný projekt slouží jako alternativa k stávajícím řídicím či bezpečnostním systémům. Přináší řadu výhod, jako je cena a jednoduchost celého zařízení, které umožňuje nasazení jak pro malé objekty, tak firmy, nebo další specifitější aplikace.

Oproti současným systémům, které jsou využívány v bezpečnostních aplikacích, nabízí přenesení poplachové informace a ovládání objektu v reálném čase, a to zcela zdarma přes Internet. S tím jsou spojená také určitá rizika – v první řadě bezpečnost, nicméně s použitím stávajících technologií, které disponují poměrně silným zabezpečením, je velmi obtížné dobře zabezpečený systém jakkoliv napadnout. Kromě výše zmíněného, lze dále zabezpečovat vnitřní komunikaci například asymetrickou šifrou RSA, která je ale nutné vhodně upravit možnostem použitého MCU.

Podobné systémy se v praxi využívají už poměrně běžně. Jejich nasazení se uplatňuje především v průmyslových objektech, kde je nutné řídit a monitorovat velké množství periférií, ale stále více se tyto systémy dostávají i do běžných domácností. Bohužel je jejich cena stále nepřiměřená, což odradí řadu potenciálních zájemců. S využitím navrhované technologie by se cena celého zařízení výrazně snížila se zachováním veškeré funkcionality.

Bezpečnost objektu a jeho obyvatel je jistě na prvním místě, ale i snížení nákladů za energie, či pohodlí, jsou v dnešní době také velmi podstatné, proto se jeví využití řídicích systémů, které automatizují procesy v objektu a vzdáleně ho monitorují, potenciálně velmi výhodné a dá se předpokládat, že se tento průmysl bude v budoucnu podstatně více rozvíjet a navrhovaná technologie by se zcela jistě mohla v této oblasti prosadit a konkurovat stávajícím řešením, za zlomek ceny.

V průběhu diplomové práce je řešena struktura současných řídicích systémů, jejich konstrukce a způsoby, jakými lze tyto systémy realizovat. Řízení a monitorování periférií probíhá pomocí jednoho centrálního zařízení, které všechny subsystemy sdružuje do jediného celku, který lze snadno řídit z nadřazené serverové aplikace. Oproti decentralizovanému přístupu, tak přináší mnohem jednodušší a levnější systém komunikace, který je zprostředkovan prostřednictvím WI-FI routeru a propůjčuje možnost vzdáleného řízení, díky TCP/IP paketům, kterými systém také komunikuje s jeho perifériemi, jejichž návrh je popsán a realizován v praktické části.

K praktickému nasazení popisovaného řešení do praxe, je nutné výše diskutovaný systém podstatně více rozšířit o různé typy detektorů a akčních členů, které poskytují řídicímu systému informace a umožňují mechanické či elektronické ovládání dílčích subsystémů. Univerzální platforma pro detektor i akční člen je navržena, popsána a realizována v průběhu praktické části. Její hlavní částí je MCU s připojeným komunikačním rozhraním, který se stará jak o výměnu dat mezi řídicí aplikací a klienty, ale také řeší nezbytné výpočty a transformuje tak získaná data pro snadnou interpretaci serverem. Tato platforma je tedy základem pro jakýkoliv typ periférií, s kterými je řídicí systém schopen na základě určených pravidel komunikovat. Jako presentace akčních členů je využit WI-FI robot, který reprezentuje způsoby mechanického řízení pomocí motorů a tlumič světla, jehož princip lze snadno aplikovat také na regulaci zařízení, které lze řídit změnou velikosti napětí (teplota, otáčky motorů atd.), čímž je pokryt téměř kompletní rozsah funkcí, které nabízí konkurenční systémy.

Využití řídicích systémů pro menší budovy je stejně důležité, jako v průmyslových objektech a v budoucnu se budou tyto systémy do běžných domácností aplikovat stále více. Mohou zabraňovat vzniku požáru, detekovat pokus o vloupání, chránit bezpečnost obyvatel a také zvyšovat jejich pohodlí. Na všechny tyto aspekty je brán v moderní společnosti důraz a proto je důležité se podobnými návrhy zabývat a najít tak nejvhodnější způsob, jakým řídicí a bezpečnostní systémy realizovat.[8][11]

## ZÁVĚR V ANGLIČTINĚ

This project is an alternative to the existing control and security systems. It brings many benefits, such as cost and simplicity of the device that allows deployment as small objects as companies or other specific applications. In comparison with current systems, which are used in security applications, offers transferring alarm information and control objects in real-time, completely free of charge via the Internet. To this are also some risks - primarily safety, however, using existing technologies, which have relatively strong safety, it is very difficult to attack good- secure system. Except to the above, can also provide internal communications such as RSA asymmetric encryption, but that is necessary to properly adjust it the options used MCU.

Similar systems are used in practice for quite normally. Their deployment is used mainly in industrial buildings where is a needed to manage and monitor a large number of peripherals, in recent years, increasingly these systems have come into common households. Unfortunately, their price is still inadequate, which deter many potential candidates. With the use of the proposed technology, the price of the entire device significantly reduced while maintaining full functionality.

The safety of the building and its inhabitants is certainly in the first place, but also reduce energy costs, or comfort, are now also vital, therefore, appears to be the use of control systems that automate processes in place and monitor it remotely, potentially highly competitive and it can be expected that this industry will be in future much more developed and proposed technology would certainly in this area could promote and compete with existing solutions at a fraction of the price.

During the thesis is solved structure of current control systems, their design and the ways in which these systems can be realized. Control and monitoring of field devices is via one central device that brings together all subsystems into a single unit that is easy to control from the parent server applications. In comparison the decentralized approach brings a lot easier and cheaper communication system, which is mediated via WI-FI router and gives the possibility of remote control, thanks to TCP/IP packets that the system also communicates with the peripherals whose design is described and implemented in the practical part.

The practical deployment of the solution described in practice, it is necessary above discussed system much more extended with different types of detectors and actuators that provide information and control system enables mechanical or electronic control subsystems. Universal platform for detector and actuator is designed, documented and implemented during the practical part. Its main component is connected to the MCU communication interface that takes care of the data exchange between control applications and clients, but also addresses the necessary calculations and thus transforms the obtained data for easy interpretation by the server. The platform is thus the basis for any type of peripheral, with which the control system is able to communicate with the identified rules. As the presentation of the actuators is used WI - FI robot , which represents the way of mechanical control using motor and dimmer, the principle of which can be easily applied also to control devices that can be controlled by varying the voltage (temperature, engine speed, etc.), which is covered almost the entire range of features offered by competing systems.

**SEZNAM POUŽITÉ LITERATURY**

- [1] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. Vyd. 1. Praha: BEN - technická literatura, 2004, 247 s. ISBN 80-730-0141-1.
- [2] LÁNIČEK, R. *Elektronika obvody, součástky, děje*. Praha, 1998, 478 s. ISBN 80-860-5625-2.
- [3] VÁŇA, Vladimír. *Mikrokontroléry ATMEL AVR: programování v jazyce C : popis a práce ve vývojovém prostředí CodeVisionAVR C*. 1. vyd. Praha: BEN - technická literatura, 2003, 215 s. ISBN 3-7723-4154-3.
- [4] MANN, Burkhard. *C für Mikrocontroller: ANSI-C, C-Compiler/Linker, Echtzeitbetriebssysteme, C-Programmierbeispiele, Tools für die Programmierung, Tipps und Tricks ; mit 30 Tabellen ; [auf CD-ROM: Beispielsammlung, AVR-Studio, Debugger, Free- und Shareware-Tools, Applikationshinweise]*. Poing: Franzis, 2000. ISBN 37-723-4154-3.
- [5] STENGL, Jens Peer a Jenő TIHANYI. *Výkonové tranzistory MOSFET*. 1. české vyd. Praha: BEN - technická literatura, 1999, 191 s. ISBN 80-860-5654-6.
- [6] OTČENÁŠEK, Martin. *Distribuované řídicí systémy a jejich využití v praxi*. Brno, 2008. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií.
- [7] TOMAN, Karel. Decentralizované sběrníkové systémy. In: [Http://www.tzb-info.cz](http://www.tzb-info.cz) [online]. 2007 [cit. 2014-05-22]. Dostupné z: <http://www.tzb-info.cz/4213-decentralizovane-sbernicove-systemy>.
- [8] Inteligentní budovy. *Intelligentní budovy* [online]. 2009 [cit. 2014-05-22]. Dostupné z: <http://www.intelligentni-budovy.cz/>.
- [9] Mikrokontrolery PIC. *Mikrokontrolery PIC* [online]. 2004 [cit. 2014-05-22]. Dostupné z: <http://mikrokontrolery-pic.cz/zaciname/co-je-to-mikrokontroler/>.
- [10] *Arduino: arduino manual* [online]. 2000 [cit. 2014-05-22]. Dostupné z: <http://playground.arduino.cc/Main/ManualsAndCurriculum>.
- [11] BACHÁNEK, Richard. Centralizovaný systém řízení s podporou TCP/IP. In: Zlín, s. 12. Dostupné z: [http://stoc.fai.utb.cz/PDF/bachanek\\_richard.pdf](http://stoc.fai.utb.cz/PDF/bachanek_richard.pdf).
- [12] Processing. *Processing* [online]. 2000 [cit. 2014-05-22]. Dostupné z: <http://www.processing.org/reference/>.

- [13] NAVRÁTIL, Josef. Ardubot. *Ardubot* [online]. 2006 [cit. 2014-05-22]. Dostupné z: [http://www.josefnav.cz/Ardubot\\_clanek.html](http://www.josefnav.cz/Ardubot_clanek.html).
- [14] Skryté sítě a NAT. *Skryté sítě a NAT* [online]. 2000 [cit. 2014-05-22]. Dostupné z: <http://zam.opf.slu.cz/botlik/Cd-II/CD-nat/skry1.htm>.
- [15] KRČMÁŘ, Petr. Proč není NAT totéž co firewall. *Root.cz* [online]. 2007 [cit. 2014-05-22]. Dostupné z: <http://www.root.cz/clanky/proc-neni-nat-totez-co-firewall/>.
- [16] VAŇUŠ, Jan. VŠB - TU OSTRAVA. *Systémová technika budov a bytů*. 2003. Dostupné z: <http://fei1.vsb.cz/kat420/vyuka/TZB/systemova%20technika%20budov.pdf>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

|              |                                                             |
|--------------|-------------------------------------------------------------|
| TCP/IP       | Transmission Control Protocol/Internet Protocol             |
| TTL          | Transistor Transistor Logic                                 |
| NMOS         | N-type metal-oxide-semiconductor                            |
| MCU, $\mu$ C | Microcontroller Unit                                        |
| CPU          | Central Processing Unit                                     |
| RAM          | Random Access Memory                                        |
| EEPROM       | Electrically Erasable Programmable Read-Only Memory         |
| LCD          | Liquid Crystal Display                                      |
| USART        | Universal Synchronous/Asynchronous Receiver and Transmitter |
| IDE          | Integrated Development Environment                          |
| USB          | Universal Serial Bus                                        |
| IP           | Internet Protocol                                           |
| MAC          | Media Access Control Address                                |
| BIOS         | Basic Input-Output System                                   |
| IT           | Information Technology                                      |
| PIR          | Passive Infrared Sensor                                     |
| MV           | mikrovlnný                                                  |
| A/D          | analog/digital                                              |
| SPI          | Serial Peripheral Interface                                 |
| IR           | Infra Red                                                   |
| LCD          | Liquid Crystal Display                                      |
| AC           | Alternating Current                                         |
| NAT          | Network Address Translation                                 |
| RSA          | Rivest, Shamir, Adleman                                     |
| ARM          | Advanced RISC Machine                                       |

**SEZNAM OBRÁZKŮ**

|                                                                               |    |
|-------------------------------------------------------------------------------|----|
| <i>Obr. 1. Rozdělení řídicích systémů z pohledu historie .....</i>            | 12 |
| <i>Obr. 2. Struktura centralizovaného řídicího systému .....</i>              | 13 |
| <i>Obr. 3. Struktura hybridního řídicího systému .....</i>                    | 14 |
| <i>Obr. 4. Struktura decentralizovaného řídicího systému .....</i>            | 14 |
| <i>Obr. 5. Mikrokontroler Atmel ATmega8* / SMD provedení .....</i>            | 16 |
| <i>Obr. 6. Základní struktura mikrokontroleru .....</i>                       | 17 |
| <i>Obr. 7. Vstup/Výstup USART .....</i>                                       | 18 |
| <i>Obr. 8. IDE programovacího SW Arduino   HW Arduino (vývojový kit).....</i> | 19 |
| <i>Obr. 9. Nahrání bootloaderu do mikrokontroleru .....</i>                   | 21 |
| <i>Obr. 10. Nahrání programu do mikrokontroleru.....</i>                      | 22 |
| <i>Obr. 11. Struktura navrhovaného řídicího systému .....</i>                 | 24 |
| <i>Obr. 12. IDE programovacího SW Processing.....</i>                         | 27 |
| <i>Obr. 13. Blokové schéma obecného detektoru.....</i>                        | 29 |
| <i>Obr. 14. Charakteristika senzoru GP2Y0A21 .....</i>                        | 29 |
| <i>Obr. 15. Návrh IR detektoru s rozhrním Ethernet .....</i>                  | 30 |
| <i>Obr. 16. Moduly WIZ820io   Nano WiReach™ SMT.....</i>                      | 30 |
| <i>Obr. 17. Akční člen → WI-FI robot.....</i>                                 | 32 |
| <i>Obr. 18. Schematické zapojení WI-FI robota .....</i>                       | 33 |
| <i>Obr. 19. Konfigurace Sabertooth 2x5A.....</i>                              | 33 |
| <i>Obr. 20. Řídící software WI-FI robota.....</i>                             | 36 |
| <i>Obr. 21. Aktor → AC Dimmer .....</i>                                       | 40 |
| <i>Obr. 22. Schéma AC dimmeru 230V   fázové řízení.....</i>                   | 40 |
| <i>Obr. 23. Schéma AC Dimmer 230V .....</i>                                   | 41 |
| <i>Obr. 24. Řídící software pro AC dimmer.....</i>                            | 45 |
| <i>Obr. 25. Network Address Translation.....</i>                              | 48 |
| <i>Obr. 26. Tabulka NAT.....</i>                                              | 49 |

## SEZNAM TABULEK

*Tab. 1. Příklad programu v MCU pro IR detektor - ethernet***Chyba! Záložka není definována.**1

**SEZNAM VZORCŮ**

- (1) Výpočet délky jedné periody napětí 50Hz
- (2) Výpočet délky  $\frac{1}{2}$  periody napětí 50Hz
- (3) Délka jednoho hodinového cyklu MCU
- (4) Výpočet délky jednoho hodinového cyklu MCU
- (5) Výpočet multiplikační inverze
- (6) Vzorec pro šifrování zpráv pomocí RSA
- (7) Vzorec pro dešifrování zpráv pomocí RSA