

# **Implementace SNMP protokolu pro monitoring a správu embedded zařízení**

Marek Bařina

---

Bakalářská práce  
2014

 Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

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

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Marek Bařina**  
Osobní číslo: **A11079**  
Studijní program: **B3902 Inženýrská informatika**  
Studijní obor: **Informační a řídicí technologie**  
Forma studia: **prezenční**

Téma práce: **Implementace SNMP protokolu pro monitoring a správu embedded zařízení**

Zásady pro vypracování:

1. Prostudujte protokol SNMP (Simple Network Management Protocol).
2. Najděte dostupné implementace protokolu SNMP pro procesory ARM Cortex M.
3. Implementujte SNMP protokol pro sledování některých funkcí vývojové desky s embedded CPU ARM Cortex M (např. stav tlačítek, I/O pinů, vnějších senzorů...).
4. Při testování vaší implementace realizujte také zátěžový test – změřte např. max. počet souběžných SNMP spojení, typický čas pro vyřízení SNMP požadavku, velikost zabrané RAM paměti při zátěžovém testu atd.
5. Vyberte a nainstalujte některou z open source SNMP monitorovacích web aplikací a nastavte ji pro monitorování vašeho zařízení.



Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. MAURO, Douglas R a Kevin J SCHMIDT. Essential SNMP. 2nd ed. Beijing: O'Reilly, 2005, 442 s. ISBN 05-960-0840-6.
2. YIU, Joseph. Definitive guide to the ARM Cortex-M3. Boston: Newnes, c2007, xix, 359 p. ISBN 978-075-0685-344.
3. SLOSS, Andrew N, Dominic SYMES a Chris WRIGHT. ARM system developer's guide: designing and optimizing system software. Amsterdam: Elsevier, 2004, 689 s. ISBN 15-586-0874-5.
4. VALVANO, Jonathan W. Embedded systems: Introduction to the Arm Cortex(TM)-M3 microcontrollers. 2nd ed. s.l.: CreateSpace, 2012, xii, 462 s. ISBN 9781477508992.
5. VALVANO, Jonathan W. Embedded Systems. s.l.: CreateSpace, 2012, xi, 366 s. ISBN 978-1466468863.
6. HEROUT, Pavel. Učebnice jazyka C. 6. vyd. České Budějovice: Kopp, 2009, 271, viii s. ISBN 978-80-7232-383-8.
7. SNMPlink [online]. 1999-2013 [cit. 2014-02-06]. Dostupné z: <http://www.snmplink.org/>

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**  
Ústav informatiky a umělé inteligence

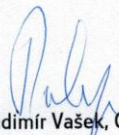
Datum zadání bakalářské práce: **28. února 2014**

Termín odevzdání bakalářské práce: **13. června 2014**

Ve Zlíně dne 28. února 2014

  
prof. Ing. Vladimír Vašek, CSc.  
děkan



  
prof. Ing. Vladimír Vašek, CSc.  
ředitel ústavu

### **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/bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

## **ABSTRAKT**

Cílem této práce bylo implementovat SNMP protokol pro sledování a správu embedded zařízení s procesorem ARM Cortex M. To znamená vytvoření private MIB souboru a jeho implementaci do zvoleného open-source IP stacku s podporou SNMP.

Klíčová slova: SNMP, embedded, ARM Cortex M, monitoring, LwIP

## **ABSTRACT**

The goal of this thesis is to implement SNMP protocol for monitoring and management of embedded device with ARM Cortex M processor. That means creating a private MIB file and its implementation into some open-source IP stack that supports SNMP.

Keywords: SNMP, embedded, ARM Cortex M, monitoring, LwIP

Chtěl bych poděkovat své rodině za podporu během studia, protože bez nich bych se studiu věnovat nemohl. Dále bych chtěl poděkovat vedoucímu své práce Ing. Tomáši Dulíkovi za pomoc a rady při vypracovávání mé práce a za možnost pracovat na tomto zajímavém tématu.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

*„Kdo víno má a nepije, kdo hrozny má a nejí je, kdo ženu má a nelíbá, kdo zábavě se vyhýbá, na toho vemte bič a hůl, to není člověk, to je vůl.“*

- *Jan Werich*

# OBSAH

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 SNMP</b> .....	<b>11</b>
1.1 MANAGEMENT INFORMATION BASE (MIB) A OID .....	12
1.2 STRUCTURE OF MANAGEMENT INFORMATION (SMI).....	14
1.2.1 Textová definice MIB podle SMI: .....	14
1.2.2 Datové typy .....	17
1.3 FORMÁT SNMP ZPRÁVY .....	17
1.4 TYPY SNMP OPERACÍ .....	18
1.5 RFC A VERZE SNMP.....	19
1.5.1 SNMPv1 .....	19
1.5.2 SNMPv2.....	19
1.5.3 SNMPv3.....	20
<b>II PRAKTICKÁ ČÁST</b> .....	<b>21</b>
<b>2 IMPLEMENTACE SNMP</b> .....	<b>22</b>
2.1 LWIP STACK A DALŠÍ KNIHOVNY .....	22
2.2 POUŽITÉ VÝVOJOVÉ NÁSTROJE .....	23
2.2.1 Další použité nástroje.....	24
2.3 POPIS KÓDU .....	25
2.3.1 Struktury a funkce .....	26
2.3.2 Složky a soubory projektu.....	28
<b>3 TESTY</b> .....	<b>30</b>
3.1 ČAS PRO VYŘÍZENÍ SNMP POŽADAVKU .....	30
<b>4 MONITOROVACÍ WEB APLIKACE</b> .....	<b>31</b>
4.1 MULTIKRITERIÁLNÍ ANALÝZA OPEN-SOURCE MONITOROVACÍCH WEB APLIKACÍ .....	31
4.2 CACTI.....	34
4.2.1 Jak Cacti funguje.....	34
4.2.2 Nutné komponenty pro Cacti .....	35
4.2.3 Nastavení Cacti .....	36
4.3 ZABBIX.....	40
4.3.1 Proč používat Zabbix .....	41
4.3.2 Server .....	42
4.3.3 Agent .....	42
4.3.4 Proxy .....	42
4.3.5 Sběr dat.....	42
4.4 CACTI VS. ZABBIX .....	43
<b>SEZNAM POUŽITÉ LITERATURY</b> .....	<b>48</b>

<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>49</b>
<b>SEZNAM OBRÁZKŮ .....</b>	<b>50</b>
<b>SEZNAM TABULEK.....</b>	<b>51</b>
<b>SEZNAM PŘÍLOH.....</b>	<b>52</b>

## ÚVOD

Protokol SNMP se využívá v síťových technologiích pro monitoring a částečně i správu celé sítě. Přes tento prostý protokol je možné sledovat stav všech zařízení v síti a tím mít přehled nad její správnou funkčností, např. kolik prošlo routerem poškozených paketů a podobně. Síťové prvky mají většinou implementovány standardní MIBv2 týkající se síťové komunikace, což jsou přesně definovaná strukturovaná data pro správu zařízení (bude vysvětleno podrobněji). Přes SNMP protokol pak můžeme tyto data získávat ale i měnit. V počítačových sítích se objevují různá zařízení, nejen PC, routery, switche a podobně, ale například tiskárny, kamery a různá specifická embedded zařízení. Každé takové zařízení obsahuje specifická data týkající se jeho funkčnosti a ty chceme monitorovat a popřípadě upravovat, ale standardní MIB pro ně neexistují. Proto si pro ně můžeme vytvořit vlastní tzv. private MIB soubory a v nich si definovat vlastní struktury dat.

V této práci jsem se zabýval základními principy protokolu SNMP a jeho implementací pro embedded zařízení s čipem ARM cortex M. Dále jsem přes tento protokol, pomocí vhodné open-source monitorovací web aplikace, monitoroval stav zařízení.

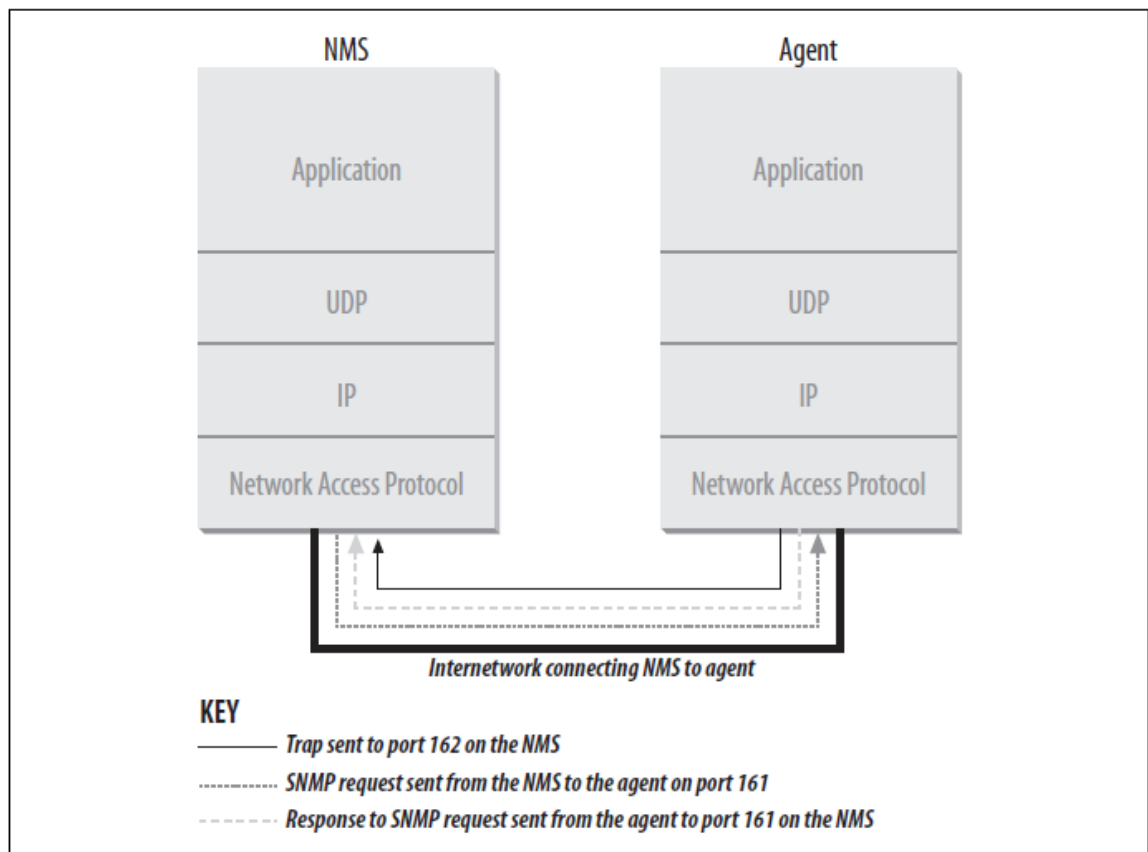
První kapitola obsahuje popis protokolu SNMP a jeho verzí, MIB databáze a pravidel SMI, podle kterých jsem vytvořil vlastní MIB soubor. V druhé kapitole je popsán použitý IP stack, vývojové prostředí Tasking a další programy používané při vývoji. Dále je zde částečně popsán kód implementace vlastního MIB souboru a stručný popis adresářů projektu. Třetí kapitola obsahuje provedené testy hotového programu. V poslední kapitole jsem se zabýval open-source monitorovacími web aplikacemi a jejich nastavením. Nejprve jsem provedl multikriteriální analýzu dostupných open-source web aplikací. Pro moje zařízení jsem nastavil aplikaci Cacti a porovnal svoje zkušenosti a zkušenosti uživatelů Cacti se Zabbixem.

## **I. TEORETICKÁ ČÁST**

## 1 SNMP

SNMP(Simple Network Management Protocol) je aplikační protokol pracující nad UDP založený na modelu klient/server. Slouží ke správě uzlů sítě (servery, pracovní stanice, tiskárny, routery, switche, huby a další - vše, co má IP adresu a podporuje SNMP) a tím umožňuje administrátorům zvyšování výkonu sítě, hledání a řešení problémů na síti nebo plánování růstu sítě. [2]

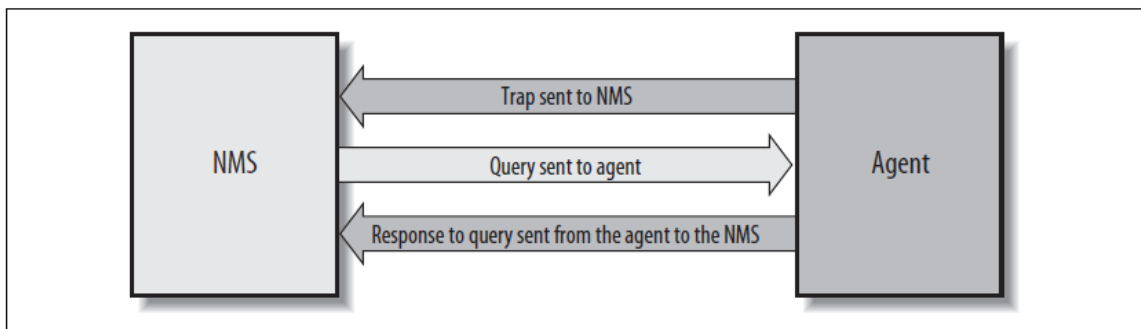
SNMP používá nespolehlivý (unreliable), nespojovaný (connectionless) transportní protokol UDP místo spojového (connection-oriented) protokolu TCP, protože UDP je jednodušší a proto lze implementovat i v nejjednodušších zařízeních, ale hlavně pokud by z nějakého důvodu nešlo na síti vytvořit spojení nutné pro TCP, stala by se síť pomocí protokolu SNMP nespravovatelnou. [2]



Obrázek 1 – TCP/IP komunikační model a SNMP [1]

Protokol SNMP vyžaduje pro komunikaci dvě strany. Jednou entitou je **správce** (NMS) a druhou **agent**. Oproti klasickému smýšlení modelu klient/server, kdy si představujeme mnoho klientů posílajících dotazy na server je tomu u SNMP právě naopak. Správce (klient) posílá dotazy všem agentům (serverům) v síti. SNMP pracuje ve dvou režimech činnosti:

- **Správce posílá dotazy agentům a přijímá odpovědi.** Hodnoty tedy může získávat i více správců a mohou se ptát kdykoliv.
- **Agent zasílá oznámení (trap) na adresu správce.** V nějakých definovaných situacích (překročení nějaké hodnoty nebo i v pravidelném intervalu) odesílá agent jednomu správci hodnoty.

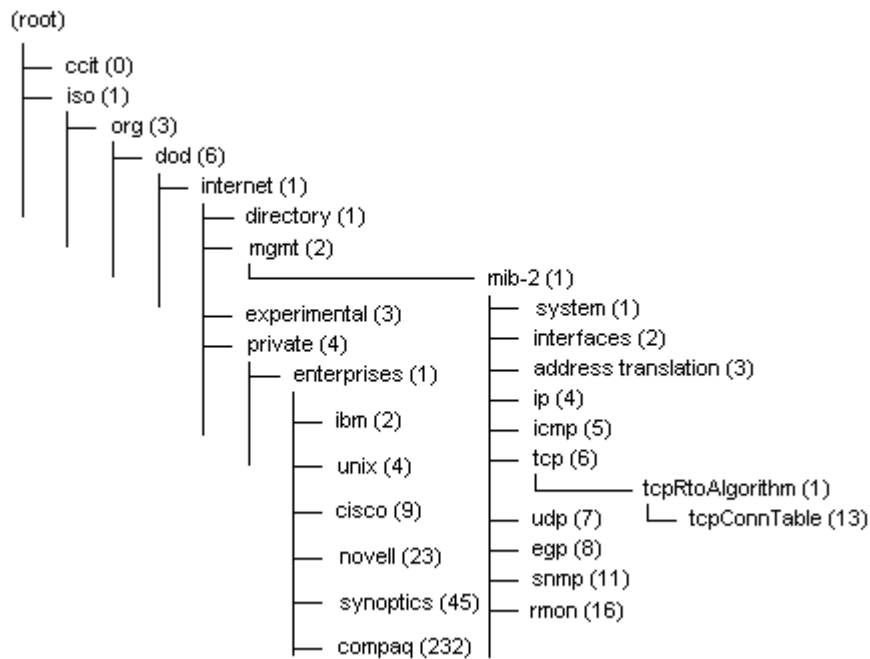


Obrázek 2 – vztah mezi správcem (NMS) a agentem [1]

## 1.1 Management Information Base (MIB) a OID

Management Information Base (MIB) popisuje sadu objektů, které jsou předmětem správy. Spravované zařízení může implementovat jednu nebo více MIB, v závislosti na jeho funkci. Tyto MIB databáze jsou velmi podobné standardním databázím v tom smyslu, že popisují jak strukturu, tak formát dat. MIB jsou napsány podle pravidel Structure of Management Information (SMI). [2]

MIB je tedy datová hierarchická stromová struktura, která odpovídá danému konkrétnímu zařízení a je objektově orientována jako sada SNMP objektů, relací a operací na a mezi objekty.



Obrázek 3 – Stromová struktura MIB

Každý záznam je objektem a má svoje OID (object ID). To je tvořeno sekvencí číselných integerů na cestě z root přes subtree až k danému objektu typu leaf. Tato decimální notace reprezentuje tedy cestu ke každé z funkcí nebo schopností daného zařízení. Jde o podobný systém jako při specifikacích plných cest k souborům v systémech UNIX a DOS, přičemž nejvyšší úroveň začíná v objektu (root). Textový popis slouží jen k naší snadnější orientaci v této struktuře.

Ještě je velmi důležité zmínit, že když se např. ve Wiresharku podíváme na SNMP paket tak uvidíme, že u OID je na konci „.0“. Je to z toho důvodu že MIB má konvenci  $x.y$  kde  $x$  je OID a  $y$  je instance tohoto objektu. Pokud tedy není tento objekt tabulka ale skalární hodnota je na konci vždy „.0“.

Ve větvi internet(1) jsou vytvořeny tři logické skupiny:

- **Management Branch** - ty standardní MIB, které byly vytvořeny orgánem IETF ([www.ietf.org](http://www.ietf.org)), jsou umístěny v části (...internet(1).mgmt(2)) hierarchické struktury MIB a obsahují definované objekty pro některé běžné síťové zařízení a protokoly. Tuto skupinu podporují zařízení většiny výrobců a tak umožňují jejich nezávislou správu. Například 1.3.6.1.2.1.1.5 je OID pro systémové jméno (sysName) z větve (...internet(1).mgmt(2).mib-2(1).system(1)).

- **Experimental Branch** - tato větev zahrnuje MIB, které jsou zatím ve vývoji.
- **Private Branch** - privátní MIB jednotlivých výrobců jsou lokalizovány v části (iso(1).org(3).dod(6).internet(1).private(4).enterprises(1)). Tato větev tak umožňuje jednotlivým výrobcům vytvářet MIB pro svá vlastní zařízení, jimž nedostačují standardní MIB. Tak např. OID 1.3.6.1.4.1.45 reprezentuje cestu k objektům firmy SynOptics, 1.3.6.1.4.1.23 cestu k objektům Novell, 1.3.6.1.4.1.9 cestu k objektům Cisco, atd.

Textová reprezentace MIB je pojmenovaný modul, který musí vyhovovat syntaktickým pravidlům podmnožiny jazyka ASN.1 (Abstract Syntax Notation One) a seskupuje dohromady odpovídající definice. Jeden nebo více těchto modulů je uchováno jako standardní ASCII soubor. [2]

## 1.2 Structure of Management Information (SMI)

SMI je přesný popis struktury dat uložený v MIB databázi. Tyto data mohou být reprezentovány jen dvojím typem jako skaláry nebo tabulky. Abychom mohli získat data ze zařízení, nestačí nám znát pouze OID ale potřebuje znát také jejich datový typ a popřípadě přístupnost dat. SMIV1 (RFC 1155, RFC1212 a RFC1215) přesně definuje, jak jsou data pojmenovány, jaký jim lze přiřadit datový typ a přístupnost. Dále existuje ještě SMIV2 (RFC2578, RFC2579 a RFC2580), která je rozšířením SMIV1.

### 1.2.1 Textová definice MIB podle SMI:

NEW-MIB DEFINITIONS ::= BEGIN

<importy>

<definice modulu>

<definice všech objektů>

<definice implementační požadavků (nepovinné)>

END

Příklad definice importů:

**IMPORTS**

OBJECT-TYPE, NOTIFICATION-TYPE, MODULE-IDENTITY,  
Integer32, Opaque, enterprises, Counter32

**FROM SNMPv2-SMI**

TEXTUAL-CONVENTION, DisplayString, TruthValue

**FROM SNMPv2-TC;**

Příklad definice modulu:

ups **MODULE-IDENTITY**

**LAST-UPDATED** "201404140000Z"

**ORGANIZATION** "organizace"

**CONTACT-INFO**

"M. Barina 721161144"

**DESCRIPTION**

"Enterprise number je přiřazované společností IANA"

::= { enterprises 42422 }

Definice uzlu MIB podle SMIV2:

<jméno uzlu> **OBJECT IDENTIFIER** ::= { <jméno modulu a unikátní OID> }

Příklad:

objects **OBJECT IDENTIFIER** ::= { ups 1 }

Definice objektu MIB podle SMIV2:

<jméno> **OBJECT-TYPE**

**SYNTAX** <datový typ viz. Tabulka 1>

**UnitsParts** <nepovinné, textový popis jednotek (sekundy, volty, ...)>

**MAX-ACCESS** <přístupnost dat: *read-only*, *read-write*, *read-create*, *not-accessible*, a *accessible-for-notify*>

**STATUS** <*current*, *obsolete*, *depricated*>

**DESCRIPTION** "Popis objektu."

**AUGMENTS** { <(umožňuje přidání sloupce do tabulky) jméno tabulky> }

::= { <jméno nadřazeného uzlu a unikátní OID> }

Příklad definice obyčejného objektu:

**led1 OBJECT-TYPE****SYNTAX** DisplayString**MAX-ACCESS** read-write**STATUS** current**DESCRIPTION**

"LED1 status"

::= { objects 1 }

Příklad definice tabulky:

tabulka **OBJECT-TYPE****SYNTAX** SEQUENCE OF strukturaTabulky**MAX-ACCESS** not-accessible**STATUS** current**DESCRIPTION** "Toto je tabulka"

::={ objects 2 }

zaznamTabulky **OBJECT-TYPE****SYNTAX** strukturaTabulky**MAX-ACCESS** read-only**STATUS** current**DESCRIPTION** "Toto je jeden zaznam"**INDEX** {id}

::={ tabulka 1 }

## strukturaTabulky

::=**SEQUENCE**{

id INTEGER,

prvniSloupec INTEGER,

druhySloupec ipAddress

}

Toto je definice nového typu. Každý prvek záznamu (id, prvniSloupec, ...) musí být ještě definován podle definice objektu.

### 1.2.2 Datové typy

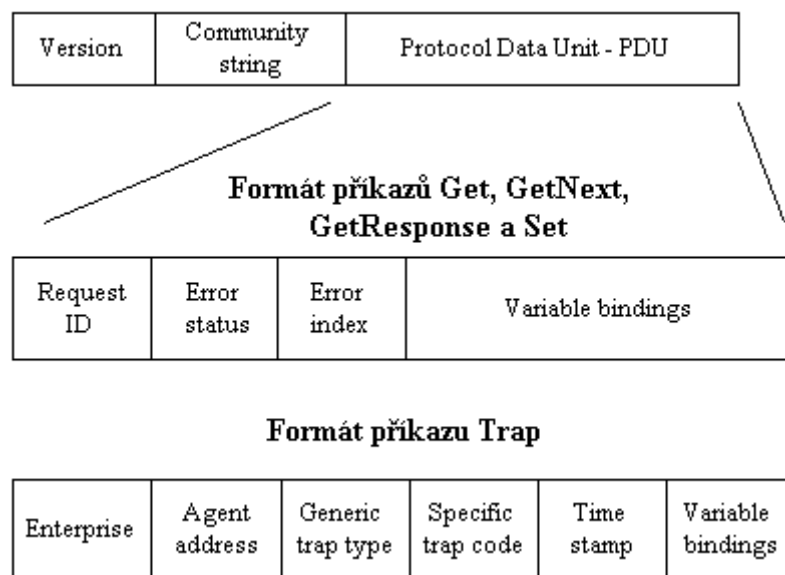
SMI definuje celou řadu datových typů, které jsou uvedeny v tabulce níže, přesto pro většinu aplikací nám bohatě stačí INTEGER (Integer32) a OCTET STRING (DisplayString) pro definici tabulek se používají speciální typy SEQUENCE a SEQUENCE OF.

SMIv1	SMIv2	SMIv2 textové konvence
INTEGER INTEGER (enumerated integer) Gauge Counter TimeTicks OCTET STRING OBJECT IDENTIFIER SEQUENCE SEQUENCE OF IpAddress NetworkAddress Opaque	Integer32 Unsigned32 Gauge32 Counter32 Counter64 BITS	DisplayString (0-255 znaků) PhysAddress MacAddress TruthValue TestAndIncr AutonomousType VariablePointer RowPointer RowStatus TimeStamp TimeInterval DateAndTime StorageType TDomain TAddress

Tabulka 1 – datové typy MIB

Upřesnění a rozsahy jednotlivých typů lze nalézt v literatuře [1] str.44-53.

### 1.3 Formát SNMP zprávy



Obrázek 4 – formát SNMP zprávy [4]

SNMP zpráva sestává ze dvou částí - hlavičky zprávy a vlastní PDU (Protocol Data Unit), viz obrázek č. 4. Hlavička zprávy obsahuje číslo verze protokolu a tzv. Community String.

Vlastní datová část zprávy obsahuje jeden ze specifikovaných SNMP příkazů a příslušný operand (položku objektu, která je předmětem transakce). Všechny pole mohou mít proměnnou délku. [4]

Jednotlivá pole mají následující význam:

- **Request ID** - přiřazuje požadavky s odpověďmi.
- **Error status** - indikuje chybu a její typ.
- **Error index** - přiřazuje chybu dané proměnné z pole variable bindings.
- **Variable bindings** - obsahuje vlastní data SNMP PDU, přiřazuje daným proměnným jejich aktuální hodnoty (vyjma Get a GetNext příkazů ti přiřazují NULL). [4]

SNMP PDU typu trap se trochu liší a obsahuje tyto pole:

- **Enterprise** - identifikuje typ objektu, který vygeneroval trap.
- **Agent address** - je adresa objektu, který vygeneroval trap.
- **Generic trap type, Specific trap code** - identifikují typ a kód trapu.
- **Time stamp** - čas mezi poslední reinitializací sítě a vygenerováním trapu.
- **Variable bindings** - seznam proměnných, které obsahují relevantní informace k danému trapu. [4]

## 1.4 Typy SNMP operací

- **get**
  - žádost o jednu proměnnou
- **getnext**
  - žádost o následující proměnnou
- **getbulk** (SNMPv2 a SNMPv3)
  - žádost o větší množství OID např. celou tabulku
- **set**

- žádost o přepsání jedné proměnné
- **getresponse**
  - odpověď na předešlou službu get, getnext nebo getbulk
- **trap**
  - trap vygenerovaná po definované události
- **notification** (SNMPv2 a SNMPv3)
- **inform** (SNMPv2 a SNMPv3)
- **report** (SNMPv2 a SNMPv3)

## 1.5 RFC a verze SNMP

Výše zmiňované dokumenty RFC jsou vydávány společností IETF (The Internet Engineering Task Force). Ta je zodpovědná za definování standardů v internetovém provozu včetně protokolu SNMP a jeho verzí.

### 1.5.1 SNMPv1

Je první verze SNMP popsaná v RFC 1157 a je to historický IETF standard. Jeho zabezpečení je založeno na tzv. komunitách (communities) což nejsou nic jině než hesla (čistý text). U verze jedna nelze mluvit ani tak o heslech jako spíše o právech přístupu k datům. Komunity u verze jedna jsou tři: *read-only*, *read-write* a *trap*.

### 1.5.2 SNMPv2

Přidává další SNMP službu pro komunikaci mezi správci (NMS) INFORM a dále službu GET-BULK. SNMP v2(RFC 1441, RFC 1452) byl původně vyvinut v roce 1993 a měl vyřešit některá omezení SNMP v1. Obsahoval zvýšenou úroveň bezpečnosti, schopnost požadavku většího množství dat (bulk retrieval), schopnost komunikace manager-manager, zvětšení efektivity protokolu i na jiných přenosových protokolech než IP, plus další vylepšení.

Implementace SNMP v2 byla však příliš složitá (zejména v oblasti bezpečnosti) a proto byla upravena na Community-Based Simple Network Management Protocol version 2 neboli SNMPv2c (RFC 1901). Tato verze používá zabezpečení SNMP v1 s tím rozdílem že si můžeme definovat libovolný community string. [2]

### 1.5.3 SNMPv3

Nepřináší žádné změny v protokolu, pouze přidává další vyšší standardy v zabezpečení jako šifrování a autentizaci klasicky jménem a heslem. Což umožňuje administrátorům plně využít SNMP pro správu sítě např. místo protokolu TELNET. SNMPv3 je popsáno v RFC 3410, RFC 3411, RFC 3412, RFC 3413, RFC 3414, RFC3415, RFC 3416, RFC 3417, RFC 3418, a RFC 2576.

Protokoly SNMP v1 a v2 zabezpečovaly přenášená data pouze pomocí tzv. Community stringu, což byl v podstatě řetězec s heslem. Toto zabezpečení ovšem nebylo dostatečné (řetězec nebyl ani nijak šifrován), a tudíž nebylo vhodné používat tento protokol pro vzdálené nastavování hodnot, ale spíše jen pro monitoring. Protokol SNMP v3 tento problém elegantně řeší. Přidává podporu pro autentizaci a šifrování. K dispozici jsou tyto možnosti zabezpečení:

- **noAuthNoPriv** – komunikace bude probíhat bez autentizace a šifrování (v podstatě stejně jako v případě SNMPv1 a v2)
- **authNoPriv** – komunikace bude probíhat s autentizací (heslo bude zabezpečeno) a bez šifrování (obsah nikoliv)
- **authPriv** – komunikace bude probíhat jak s autentizací, tak i se šifrováním

Pro autentizaci jsou použity zabezpečovací algoritmy **MD5** nebo **SHA**, pro šifrování potom šifrovací algoritmy **DES** nebo **AES**. [1]

K zabezpečení se standardně používá spojení USM (User Security Model) společně s VACM(View Access Control Model). Jedná se v podstatě o zabezpečení pomocí definování uživatelů s různými možnostmi zabezpečení a různým přístupem k datům v závislosti na tom to zabezpečení. Existují ale také další možnosti zabezpečení a ty jsou následující:

- SNMPv3 s TLS a DTLS
- SNMPv3 přes SSH
- SNMPv3 přes Kerberos

Tyto způsoby zabezpečení jsou již ovšem využívány méně, neboť nejsou plně podporovány všemi zařízeními. [7]

## **II. PRAKTICKÁ ČÁST**

## 2 IMPLEMENTACE SNMP

Po podrobném hledání jsem zjistil, že jediným vyhovujícím a zcela bezplatným, open-source TCP/IP stackem implementujícím SNMP protokol je LwIP. Kromě tohoto projektu existuje celá řada jiných implementací, ty jsou ovšem placené nebo přímo vázané na typ procesoru. Protože já v této práci pracuji s vývojovou deskou osazenou čipem CPU STM32P107, hledal jsem i na stránkách výrobce STMicroelectronics, kde pro příklady použití ethernetu je použit právě open-source LwIP stack. Dokumentace k LwIP není velká ani ucelená, protože je vyvíjen a upravován mnoha programátory z celého světa. Informace o použití SNMP v LwIP stacku jsem proto musel hledat především na různých fórech a v emailových korespondencích lidí, kteří se jeho použitím zabírali. Po prostudování zdrojového kódu jsem zjistil, jak je MIB databáze implementována pro standardní MIB-2 a mohl tak napsat vlastní implementaci pro můj vlastní private MIB.

### 2.1 LwIP stack a další knihovny

LwIP je malá nezávislá implementace sady protokolů TCP/IP, která byla původně vyvinuta Adamem Dunkelsem a nyní je vyvíjena řadou programátorů po celém světě. Hlavním cílem LwIP je omezit použité systémové zdroje a přitom mít plný rozsah funkčnosti TCP/IP. To z něj dělá vhodný stack pro použití v embedded systémech, které musí mít alespoň několik desítek kilobajtů RAM a místo pro cca 40 kilobajtů kódu.

Nejnovější informace a fórum o LwIP lze nalézt na:

<http://savannah.nongnu.org/projects/lwip/>

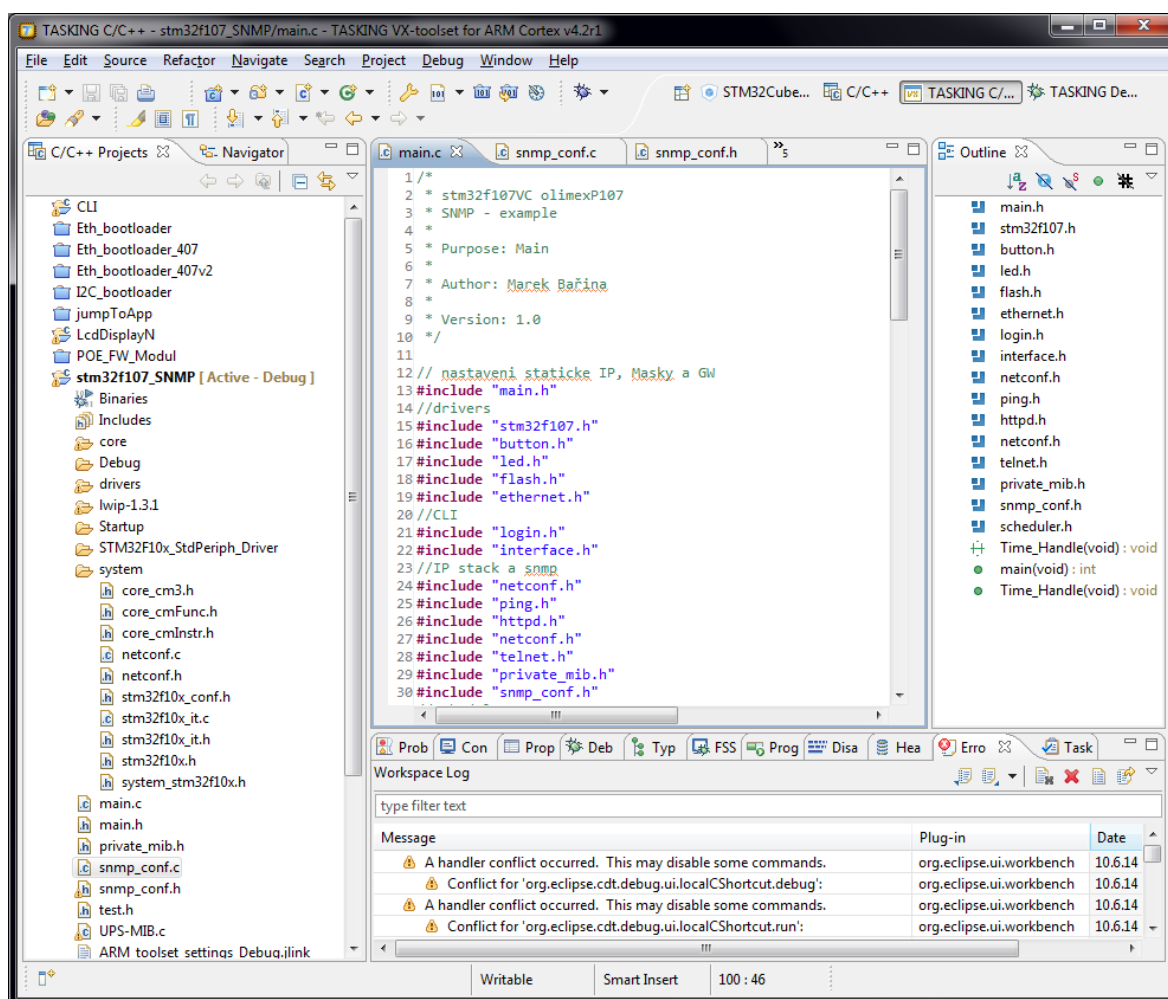
Zdrojový kód LwIP je zveřejněn pod **BSD licencí**. Tato licence pro svobodný software je jednou z nejsvobodnějších. Umožňuje volné šíření licencovaného obsahu, přičemž vyžaduje pouze uvedení autora a informace o licenci, spolu s upozorněním na zřeknutí se odpovědnosti za dílo.

#### Další knihovny

Jediná další použitá knihovna byla knihovna standardních ovladačů periférií pro čipy STM32F10x (složka v projektu: *STM32F10x\_StdPeriph\_Driver*), kterou dodává přímo výrobce STMicroelectronics.

## 2.2 Použité vývojové nástroje

Pro vývoje celého programu bylo používáno vývojové prostředí „Tasking VX-toolset for ARM Cortex“, verze 4.2r1. Je to vývojové prostředí postavené na IDE Eclipse, podporující celou řadu různých ARM procesorů od různých výrobců. Firma Tasking je dceřinou společností firmy Altium.



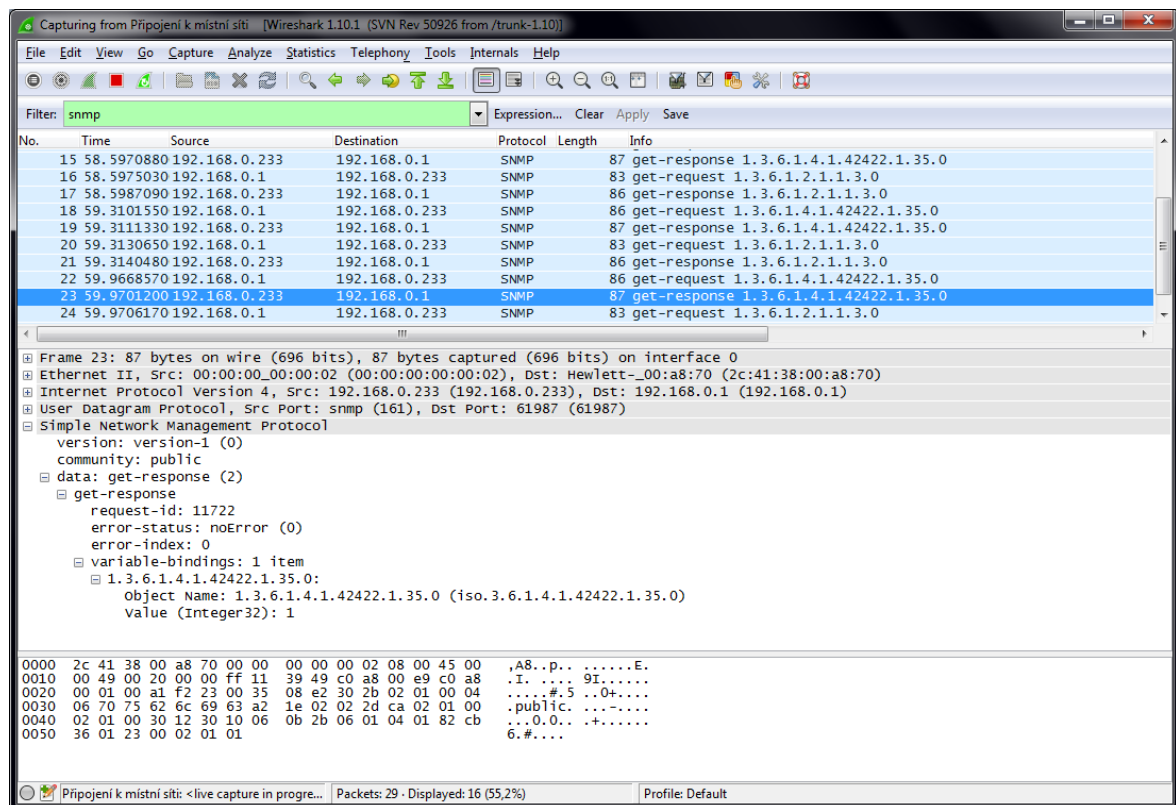
Obrázek 5 – Prostředí vývojového nástroje Tasking VX-toolset for ARM

Jak je vidět z obrázku 5, vývojové prostředí Taskingu je velmi podobné jakémukoli prostředí založeném na Eclipse. Navíc je zde hlavně vytváření projektů pro zvolený čip a vložení jeho zaváděcího kódu a možnost importu standardních knihoven od výrobců těchto čipů. Dále pak obsahuje výborný debugger, který podporuje debugovací zařízení J-Link od firmy SEGGER, ST-Link od firmy STM a miniWiggler od firmy Infineon.

## 2.2.1 Další použité nástroje

### Wireshark

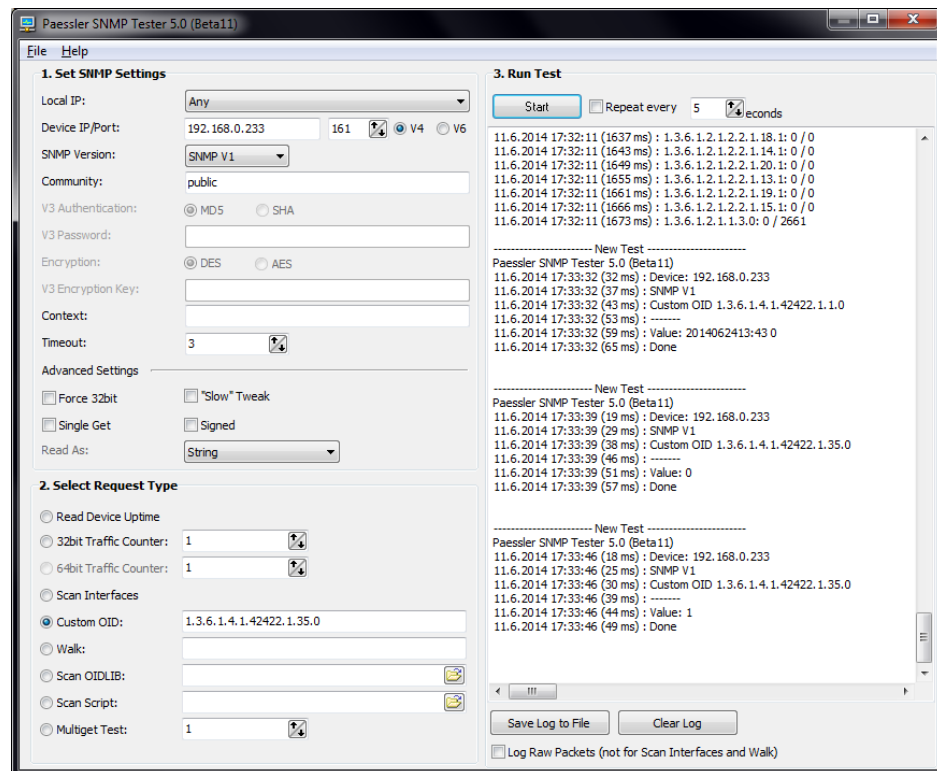
Velmi užitečný byl při vytváření této práce nástroj Wireshark. Díky němu jsem mohl snadno kontrolovat data uvnitř SNMP zpráv a velmi dobře posloužil i pro měření doby odpovědi na SNMP dotaz.



Obrázek 6 – SNMP dotaz zachycený programem Wireshark

### Peassler SNMP Tester 5.0

Při implementování SNMP musíme často posílat testovací SNMP požadavky a je vhodné na to použít nějaký program s grafickým uživatelským rozhraním (GUI). I když by nám stačila knihovna NET-SNMP a mohli bychom posílat dotazy přes příkazový řádek, GUI nadstavba nám práci usnadní. Takovým free programem na posílání SNMP dotazů, umožňující řadu nastavení je například Peassler SNMP Tester.



Obrázek 7 – Paessler SNMP Tester 5.0

## 2.3 Popis kódu

Po prostudování implementace SNMP v LwIP stacku (především souborů `mib2.c` a `mib_structs.c`) jsem vytvořil vlastní implementaci private MIBu pro UPS modul, který jsem napsal podle pravidel SMI z teoretické části (soubor `UPS-MIB.txt`). Proměnné, které jsem v něm definoval, jsem vybral podle dokumentace k nějakému UPS modulu, abych měl více OID, na které se dotazovat. Do tohoto MIBu jsem pak přidal další dvě proměnné, které indikují stav LED diod na vývojové desce. Ty se dají ovládat buď pomocí tlačítek nebo právě přes SNMP.

Z výsledného souboru (`UPS-MIB.c`) popíšu klíčové části a nastíním tak funkci kódu.

### 2.3.1 Struktury a funkce

Jednotlivé uzly (node) MIB databáze jsou v kódu definovány jako struktury:

```

/** node "base class" layout, the mandatory fields for a node */
struct mib_node
{
    /** returns struct obj_def for the given object identifier */
    void (*get_object_def)(u8_t ident_len, s32_t *ident, struct obj_def *od);
    /** returns object value for the given object identifier,
        @note the caller must allocate at least len bytes for the value */
    void (*get_value)(struct obj_def *od, u16_t len, void *value);
    /** tests length and/or range BEFORE setting */
    u8_t (*set_test)(struct obj_def *od, u16_t len, void *value);
    /** sets object value, only to be called when set_test() */
    void (*set_value)(struct obj_def *od, u16_t len, void *value);
    /** One out of MIB_NODE_AR, MIB_NODE_LR or MIB_NODE_EX */
    const u8_t node_type;
    /** array or max list length */
    const u16_t maxlength;
};

```

- První prvek struktury je ukazatel na funkci vracející (volání odkazem) definici objektu (datový typ, přístupnost, ASN typ, ...).
- Druhý je ukazatel na funkci vracející hodnotu požadovaného OID.
- Třetí je ukazatel na funkci, která při požadavku *set* nejprve testuje, jestli jsou zapisovaná data v požadovaném rozsahu.
- Čtvrtý je ukazatel na funkci, která po úspěšné kontrole funkce *set\_test* uloží do proměnné data z požadavku *set*.
- Pátý je typ uzlu.
- Šestý je počet instancí uzlu pro skalární hodnoty 0.

Struktura skalárního uzlu pak vypadá takto:

```

const mib_scalar_node objects_scalar = {
    &objects_get_object_def,
    &objects_get_value,
    &objects_set_test,
    &objects_set_value,
    MIB_NODE_SC,
    0
};

```

V této struktuře jsou již ukazatele na konkrétní funkce zpracovávající data při požadavcích typu *set* a *get*.

Pro vysvětlení principu funkcí použijí funkci `objects_get_object_def` a stručně vysvětlím postup zpracování *get* dotazu.

```
static void objects_get_object_def(u8_t ident_len, s32_t *ident, struct obj_def
*od)
{
    u8_t id;

    /* return to object name, adding index depth (1) */
    ident_len += 1; //??
    ident -= 1;
    if (ident_len == 2)
    {
        od->id_inst_len = ident_len;
        od->id_inst_ptr = ident;

        id = ident[0];
        switch (id)
        {
            case 1: /* date */
                od->instance = MIB_OBJECT_SCALAR;
                od->access = MIB_OBJECT_READ_WRITE;
                od->asn_type = (SNMP_ASN1_UNIV | SNMP_ASN1_PRIMIT | SNMP_ASN1_OC_STR);
                od->v_len = (u16_t)date_length;
                break;

            ...

        }
    }
}
```

Princip je v podstatě jednoduchý, poté co pošleme dotaz *get* se rozbalí UDP paket, zpracuje se SNMP zpráva a podle OID se projde MIB databáze (námi napsané struktury) až k ukazatelům na funkce pro zpracovávání OID. První, co je třeba o OID zjistit, je jeho definice (tzn. datový typ, přístupnost, atd.). Právě to provádí výše uvedená funkce.

Nejdříve je potřeba zjistit *id*, to je předáváno funkci odkazem v ukazateli *\*ident*. V něm je ale uloženo OID.0, tudíž se posuneme o jedno číslo zpět (`ident -= 1;`) a tím získáme *id*. Poté už jednoduše přes `switch`, do ukazatele na strukturu `obj_def *od`, uložíme informace o dotazovaném objektu. Proč, nebo spíše z jakého důvodu je nutné předávat a testovat proměnnou *ident\_len*, která musí být vždy 2, jsem nezjistil ani jsem se to nikde nedočel. Ale jelikož v implementaci MIB-2 to takto bylo napsáno, tak jsem to použil také. Nakonec se zavolá ještě funkce `objects_get_value`, která je řešena velmi podobným způsobem. Ta nám do odpovědi na dotaz *get* vloží data z dotazované proměnné.

```
static void objects_get_value(struct obj_def *od, u16_t len, void *value)
{
    u8_t id;
    unsigned char* uk_char;

    id = od->id_inst_ptr[0];
    switch (id)
    {
        case 1: /* date */
            uk_char = date;
            strncpy(value, (u8_t*)uk_char, len);
            break;

            ...

    }
}
```

Další části kódu jako ovládací funkce tlačítek, LED, ethernetu a podobně zde popisovat nebudu, protože nesouvisí se SNMP.

### 2.3.2 Složky a soubory projektu

Pouze stručně vysvětlím jednotlivé obsahy složek projektu a podstatné soubory.

- **core** – jednoduchá implementace CLI v tomto programu nevyužitá, scheduler (plánovač spouštěných funkcí).
- **Debug** – složka vytvořená Taskingem obsahující výsledky linkeru, hotový binární soubor (stm32f107\_SMNP.abs), mapovací soubor a další.
- **drivers** – inicializační soubory pro nastavení ethernetu, tlačítek a LED diod a jejich ovládací funkce.
- **lwip-1.3.1** – soubory LwIP stacku.
- **Startup** – zaváděcí kód pro mikroprocesor (nastavení hodin, vektoru přerušení apod.).
- **STM32F10x\_StdPeriph\_Driver** – složka od STM obsahující ovladače pro standardní periferie.
- **system** – obsahuje soubory pro ovládání jádra Cortex-M3, soubory s definicemi konstant pro čipy stm32f10x, soubory pro nastavení funkcí přerušení, ale také soubory netconf, kde se provádí přidělení IP a MAC adresy pro LwIP Stack.
- **main.c** – soubor obsahující hlavní program.

- ***main.h*** – hlavičkový soubor kde je definovaná aktuální statická IP adresa MAC, maska a výchozí brána.
- ***snmp\_conf.c*** – soubor s inicializační funkcí snmp a udp, inicializace proměnných vázaných s deskou pro snmp a jejich ovládací funkce.
- ***snmp\_conf.h*** – nastavení community stringu, private OID, inicializace konstantních proměnných nevázaných s deskou pro snmp.
- ***snmp\_test.h*** – hlavičkový soubor s prototypy funkcí ovládající vázané proměnné (je vkládán tam, kde je třeba v kódu tyto funkce volat).
- ***UPS-MIB.c*** – hlavní soubor obsahující implementaci private MIB.
- ***UPS-MIB.txt*** – textová reprezentace implementovaného private MIB podle pravidel SMI.

Ještě je dobré zmínit konfigurační soubor LwIP, který se nachází v *lwip-1.3.1/port/lwipopts.h*. V tomto souboru lze zapínat a vypínat služby LwIP, definovat velikost paměti pro heap IP stacku a podobně.

### 3 TESTY

Pro hotovou implementaci jsem provedl některé základní užitečné testy. Test na maximální počet UDP „spojení“ nemá smysl, protože v LwIP je počet spojení omezen na hodnotu parametru v *lwipopts.h*.

#### 3.1 Čas pro vyřízení SNMP požadavku

V tomto testu jsem posílal *get* požadavky a odečítal časy mezi odesláním požadavku a přijetím odpovědi v programu Wireshark. Výsledky jsou zaznamenány v následující tabulce.

doba odpovědi na jeden get dotaz [s]	doba odpovědi na 10 get dotazů [s]
0,001980	0,013551
0,001737	0,012631
0,000913	0,018295
0,000535	0,012500
0,000915	0,019346
0,000396	0,016164
0,000878	0,013671
0,002014	0,013094
0,000944	0,015610
0,001111	0,013866
0,001732	0,014552
0,000967	0,018049
0,000766	0,013123
0,001030	0,010315
0,000904	0,020736
0,001068	0,014571
0,001118	0,015005

Tabulka 2 – časy pro vyřízení SNMP požadavků

Z tabulky můžeme vyčíst, že typický čas pro vyřízení jednoho SNMP požadavku je cca 1,118 ms a čas vyřízení deseti dotazů je cca 15 ms.

## 4 MONITOROVACÍ WEB APLIKACE

Existují desítky monitorovacích web aplikací napsané v různých jazycích a využívající různých frameworků. V této práci jsem se zabýval jen těmi s bezplatnými, svobodnými licencemi (BSD, GPL).

### 4.1 Multikriteriální analýza open-source monitorovacích web aplikací

Na Wikipedii [9] jsem našel porovnání všech monitorovacích web aplikací, z nichž jsem vybral ty s open-source licencí. Data z tohoto porovnání jsem použil pro multikriteriální analýzu, jejíž data a výsledky jsou v Tabulce 3 a 4.

Při analýze jsem přiřadil všem kritériím stejnou váhu, kromě platformy a uložště, které měly nulovou váhu.

Podle multikriteriální analýzy open-source web aplikací (viz. Tabulky 3, 4) je nejlepším monitorovacím nástrojem **Zenoss** a na druhém místě **Cacti** společně se **Zabbixem** a **openNMS**. Protože Zenoss je převážně napsaný v Pythonu a částečně v Javě a openNMS v Javě a s těmito technologiemi nemám žádné zkušenosti, nemohl jsem je objektivně testovat ani posoudit. V další práci jsem tedy pokračoval pouze s aplikacemi Cacti a Zabbix, které jsou implementovány v PHP a ANSI C.

Název	Shinken	Argus	Ganglia	TelMon	Cacti	Check MK	FreeNATS	Icinga	Monitorix
Cisco IP SLA	Plugin	ANO	NE	ANO	ANO	Plugin	ANO	Plugin	NE
Vytváření logických skupin	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	NE
Trendy	ANO	ANO	ANO	ANO	ANO	ANO	NE	ANO	ANO
Předpovídání trendu	NE	ANO	NE	NE	ANO	ANO	NE	NE	ANO
Automat. zjišťování zařízení	ANO	NE	gmond check in	ANO	Plugin	Plugin	ANO	Plugin	NE
Bez nutnosti agentů	NE	ANO	NE	ANO	ANO	ANO	NE	ANO	NE
SNMP	Plugin	ANO	Plugin	ANO	ANO	ANO	NE	Plugin	ANO
Syslog	Plugin	ANO	NE	ANO	ANO	ANO	Plugin	Plugin	NE
Pluginy	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Triggry / Polachy	ANO	ANO	NE	ANO	ANO	ANO	v PHP kódu	ANO	ANO
Webová aplikace	Prohlížení, Potvrzování, Hlášení	Prohlížení, Potvrzování, Hlášení	Prohlížení	Prohlížení	Plné řízení	Plné řízení	Plné řízení	Plné řízení	Prohlížení
Distribuovaný Monitoring	ANO	ANO	ANO	NE	ANO	ANO	NE	ANO	ANO
Záznamy o hostitelích	Plugin	?	?	NE	ANO	ANO	NE	Plugin	?
Platforma	Python	Perl	C, PHP	Unknown	PHP	C, Python	PHP	C	Perl
Uložiště	Flat file, MySQL, Oracle, Graphite, Sqlite, MongoDB	Flat file, Berkeley DB	RRDtool	RRDTool	RRDtool, MySQL	RRDtool	MySQL	MySQL, PostgreSQL, Oracle Database	RRDtool
Licence	AGPL	Artistic License	BSD	BSD	GPL	GPL	GPL	GPL	GPL
Mapování	ANO	NE	ANO	ANO	Plugin	Plugin	NE	ANO	?
Administrace uživatelů	ANO	ANO	NE	ANO	ANO	ANO	ANO	ANO	ANO
IPv6	ANO	ANO	?	NE	ANO	?	?	ANO	ANO
Datum vydání poslední verze	2013-05	2013-02	2013-02	2009-12	2013-08	2014-02		2014-05	2013-02
Poslední vydaná verze	1.4	3.7	3.5.2007	1.1.2001	0.8.8b	1.2.2004		1.11.2003	3.0.0
MCDA	0,7344	0,79688	0,42188	0,71875	0,9375	0,84375	0,4375	0,78125	0,5313
MCDA %	73	80	42	72	94	84	44	78	53

Tabulka 3 – Multikriteriální analýza 1 [9]

Název	Munin	Nagios	NetXMS	Xy- mon/Hobbit	Zabbix	Zenoss	Perfor- mance Co- Pilot	collectd	OpenNMS
Cisco IP SLA	NE	Plugin	NE	ANO	ANO	ANO	NE	NE	ANO
Vytváření logických skupin	NE	ANO	ANO	ANO	ANO	ANO	ANO	NE	ANO
Trendy	ANO	ANO	ANO	ANO	ANO	ANO	ANO	NE	ANO
Předpoví- dání tren- du	ANO	NE	ANO	NE	NE	ANO	NE	NE	?
Automat. zjišťování zařízení	NE	Plugin	ANO	NE	ANO	ANO	ANO	Push model	ANO
Bez nut- nosti agentů	NE	ANO	NE	NE	ANO	ANO	NE	ANO	ANO
SNMP	ANO	Plugin	ANO	Plugin	ANO	ANO	ANO	ANO	ANO
Syslog	NE	Plugin	ANO	NE	ANO	ANO	ANO	ANO	ANO
Pluginy	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Triggr / Polachy	Částečně	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Webová aplikace	Prohlížení	Plné řízení	Plné řízení	Prohlížení, Potvrzování	Plné řízení	Plné řízení	NE	Prohlížení	Plné řízení
Distribuo- vaný Moni- toring	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO	ANO
Záznaky o hostitelích	?	Plugin	NE	NE	ANO	ANO	ANO	NE	ANO
Platforma	Perl	C, PHP	C++	C, Shell	C, PHP	Python, Java	C Perl, Python, POSIX, MinGW	C	Java
Uložiště	RRDtool	Flat file, SQL	MySQL, MS SQL, Oracle, Postgre- SQL, SQLite	Flat file, RRDTool, MySQL via plugin	Oracle, MySQL, PostgreSQL, IBM DB2, SQLite	ZODB, MySQL, RRDtool	Flat file	RRDtool	JRobin / RRDTool, PostgreSQL
Licence	GPL	GPL	GPL	GPL	GPL	GPL	GPL, LGPL	GPLv2	GPLv3
Mapování	?	ANO	ANO	NE	ANO	ANO	NE	NE	ANO
Adminis- trace uživatelů	?	ANO	ANO	Apache ACL	ANO	ANO	ANO	Apache ACL	ANO
IPv6	ANO	ANO	NE	NE	ANO	ANO	ANO	ANO	ANO
Datum vydání poslední verze	2013-03	14.3.2014	2013-09	2014-02	2014-04	2014-04	2014-03	2013-08	2014-04
Poslední vydaná verze	2.0.12	4.0.4	1.2.2009	4.3.2017	2.2.2003	4.2.2005	3.9.2001	5.4	1.12.2006
MCDA	0,40625	0,78125	0,75	0,48438	0,9375	1	0,6875	0,5	0,9375
MCDA %	41	78	75	48	94	100	69	50	94

Tabulka 4 – Multikriteriální analýza 2 [9]



paktního souboru o stálé velikosti. K zobrazení pomocí grafů slouží opět RRDTool s bohatou nabídkou možností. [5]

V praxi potřebujeme vždy nejprve zadat **zařízení**, kterého se budou akce týkat, pomocí **Devices**. Následně vytvořit zdroj dat (**Data Sources**), ten se vytváří automaticky pomocí šablony. Z datového zdroje se vytvoří graf (**Graph Management**) a přidá se do nějaké skupiny pro zobrazení (**Graph Trees**). [5]

Zařízení zadáme ručně a přiřadíme mu nějakou šablonu (**Host Template**), tím se k danému zařízení budou nabízet vybrané šablony pro grafy (**Graph Template**) a dotazy (**Data Queries**). **Dotazy** slouží k zjištění informací ze zařízení o prvcích stejného typu, jako jsou procesory, porty přepínače, disky, atd., a následně umožňují vytvořit graf z vybraných položek. **Šablona grafu** je automaticky zřetězena s datovou šablonou (**Data Templates**) a vytváří zdroj dat. [5]

#### 4.2.2 Nutné komponenty pro Cacti

Pro uvedení Cacti do provozu je potřeba mít nainstalováno několik věcí.

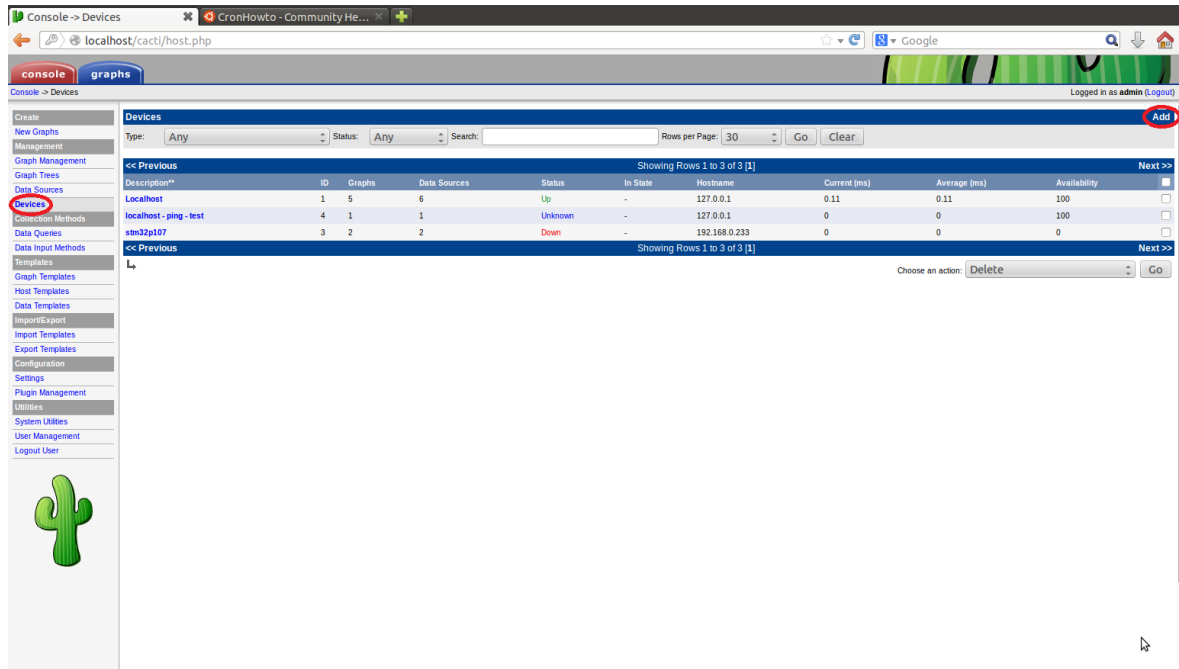
- Webserver (např. Apache)
- MySQL
- PHP
- NET-SNMP
- RRDTool
- Perl
- Cacti 0.8.8b

Postup instalace zde uvádět nebudu, protože se jmně liší pro každý operační systém, ale například pro Windows existuje instalační balíček, který výše zmiňované součásti nainstaluje společně s Cacti a nastaví je. Ovšem tato instalace je vhodná spíše pokud nemáte žádnou z těchto součástí, pokud ano, je lepší je odinstalovat pro případ, že by se spolu „háda-li“.

Postup instalace lze nalézt na oficiálních stránkách Cacti ([www.cacti.net](http://www.cacti.net)) nebo v příloženém manuálu ke Cacti. Důrazně doporučuji se přesně držet instalačních kroků a kroků pro nastavení cyklického spouštění *poller.php* (opět závisí na OS).

### 4.2.3 Nastavení Cacti

Poté co úspěšně instalaci jako první musíme vytvořit zařízení. Klikneme na *Devices* a vpravo na *Add*. Nastavíme mu název, IP, SNMP vlastnosti atd.



Obrázek 9 – Cacti Devices

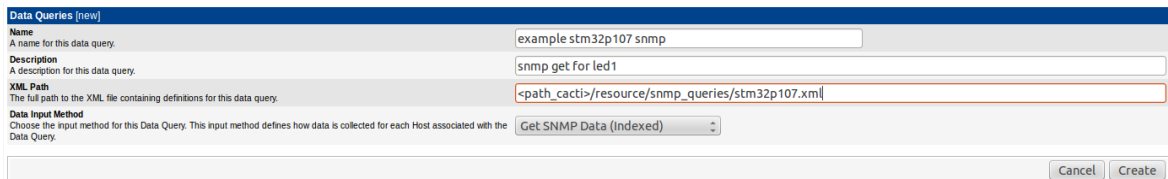
Nyní, když máme vytvořené zařízení je několik způsobů jak získávat data a ty pak zobrazovat v grafech. Nyní musíme Cacti říct jak data získávat použijeme na to příklad z dokumentace pro *SNMP Data Queries*. Na tento způsob nepotřebujeme skript, ale musíme si vytvořit XML soubor popisující strukturu MIB. Jako šablonu jsem použil hotový XML *interfaces.xml*.

## Moje XML:

```
<interface>
  <name>stm32p107 Get SNMP</name>
  <description>Queries a stm32p107</description>
  <oid_index>.1.3.6.1.4.1.42422.1.35</oid_index>
  <oid_num_indexes>.1.3.6.1.4.1.42422.1.0</oid_num_indexes>
  <index_order>led</index_order>
  <index_order_type>numeric</index_order_type>
  <index_title_format>|chosen_order_field|</index_title_format>

  <fields>
    <led>
      <name>led1</name>
      <method>walk</method>
      <source>value</source>
      <direction>input</direction>
      <oid>.1.3.6.1.4.1.42422.1.35</oid>
    </led>
  </fields>
</interface>
```

Nyní vytvoříme data query. Klikneme na *Data Queries* a vpravo na *Add*. Napíšeme název, popis, cestu k XML souboru a jako input method zvolíme *Get SNMP Data (Indexed)*.



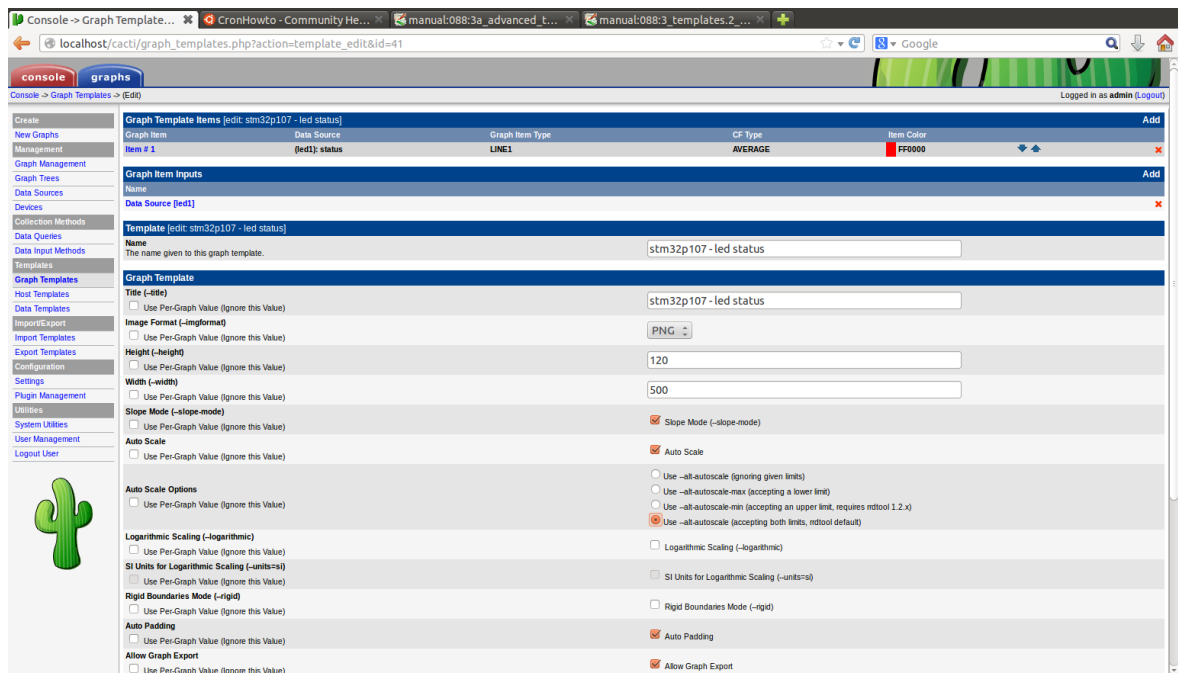
The screenshot shows the 'Data Queries [new]' configuration form in Cacti. It contains the following fields:

- Name:** example stm32p107 snmp
- Description:** snmp get for led1
- XML Path:** <path\_cacti>/resource/snmp\_queries/stm32p107.xml
- Data Input Method:** Get SNMP Data (Indexed)

At the bottom right, there are 'Cancel' and 'Create' buttons.

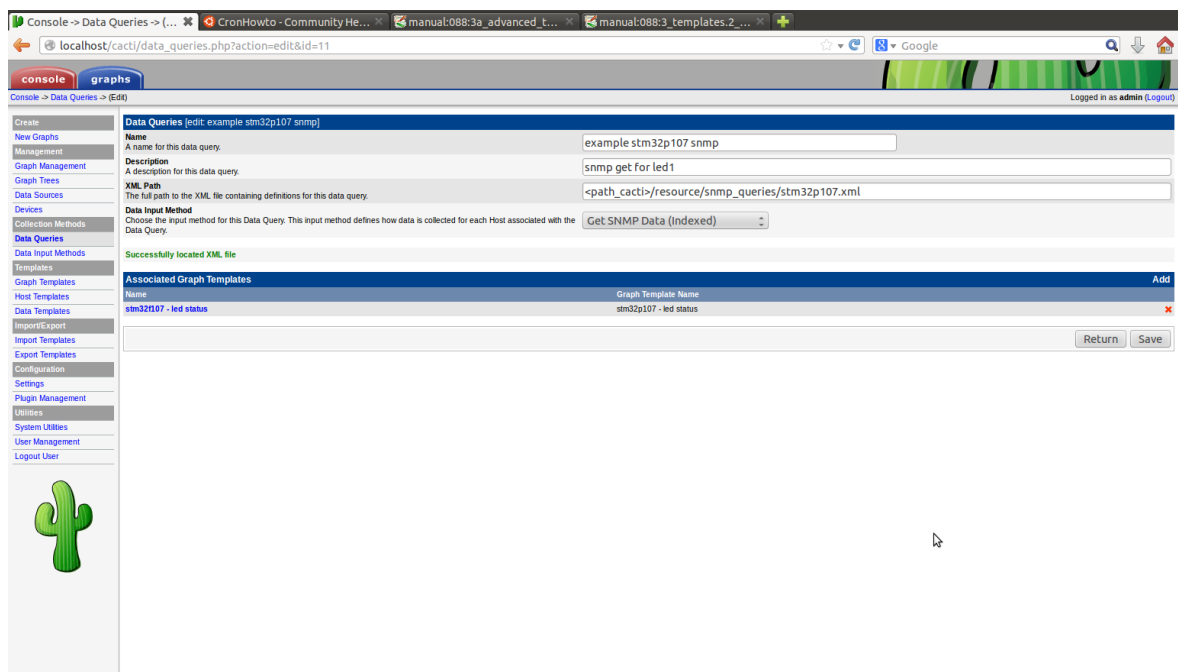
Obrázek 10 – Cacti Data Queries

Dále vytvoříme šablonu grafu. Klikneme na *Graph Templates* a vpravo na *Add*. Napíšeme název, titulek grafu a přidáme jeden *Graph Template Item* kliknutím vpravo nahoře na *Add*. Zde vybereme vlastnosti tohoto itemu (barvu a styl čáry a podobně).



Obrázek 11 – Cacti Graph Template

Poté musíme vytvořenému *Data Query* přiřadit tuto šablonu. To uděláme tak že v *Data Queries* u *Associated Graph Templates* klikneme vpravo na *Add* a vybereme vytvořenou šablonu grafu a klikneme na *create*.

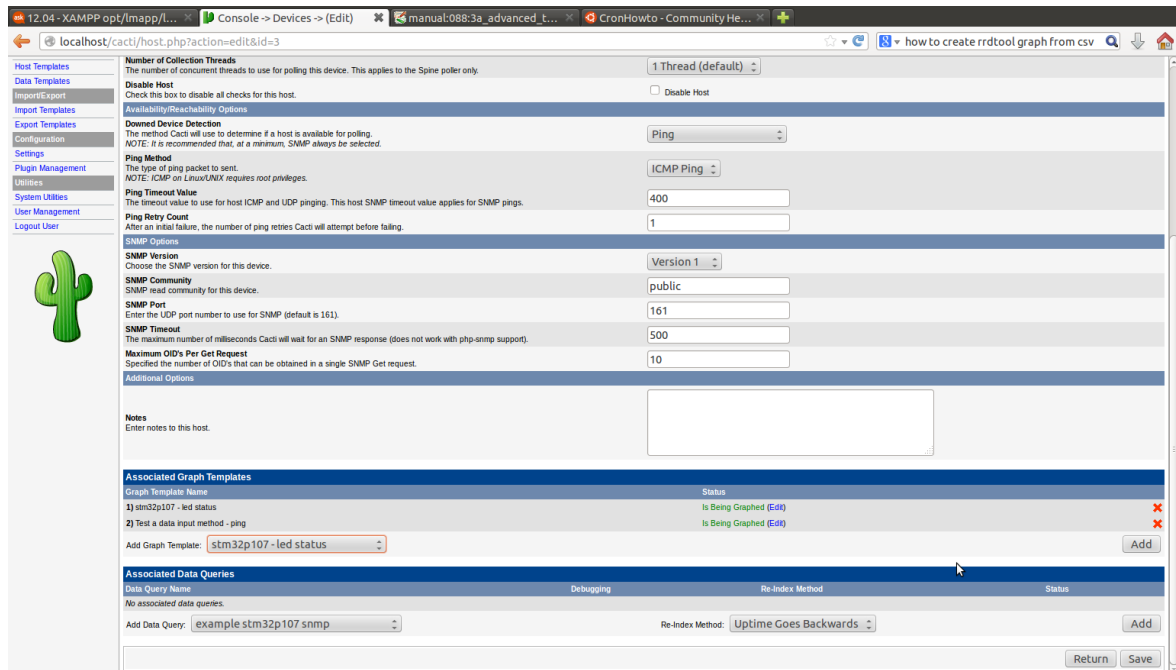


Obrázek 12 – Data Queries Assoc. Graph

Nyní rozklikneme přidanou šablonu, zaškrtneme proměnnou pojmenovanou, tak jak jsme ji uvedli v XML souboru a klikneme *save*.

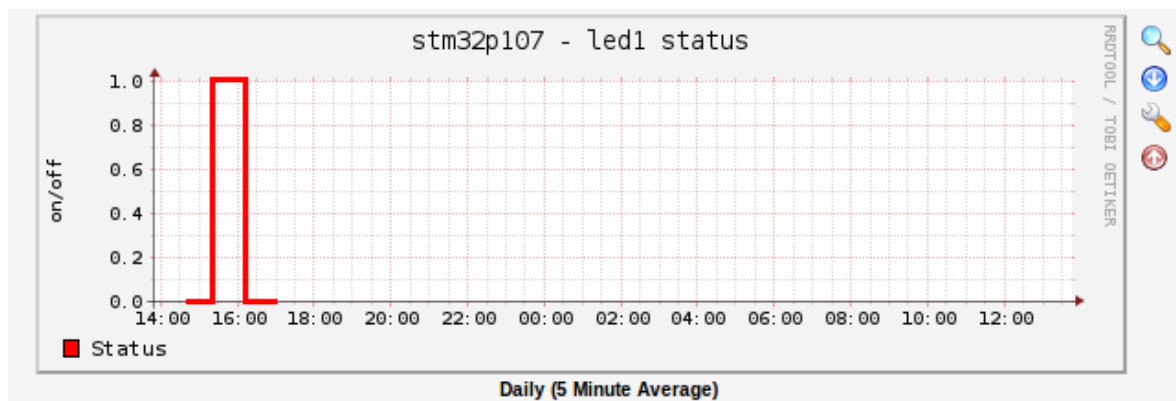
Ještě je třeba vytvořit *Data Template* (udává nám, jak se data ukládají) a ten zřetězit s *Graph Template*. Nejdůležitější je u něj vybrat step podle našeho poller intervalu (implicitně 300s = 5min).

Nakonec klikneme na *Devices*, pak na naše zařízení a u *Associated Graph Templates* mu přiřadíme naši šablonu.



Obrázek 13 - Cacti Device Edit

Pokud jsme při instalaci postupovali správně, vytvořili si všechny šablony a zřetězili je, dostaneme po nějaké chvíli testování takovýto graf.

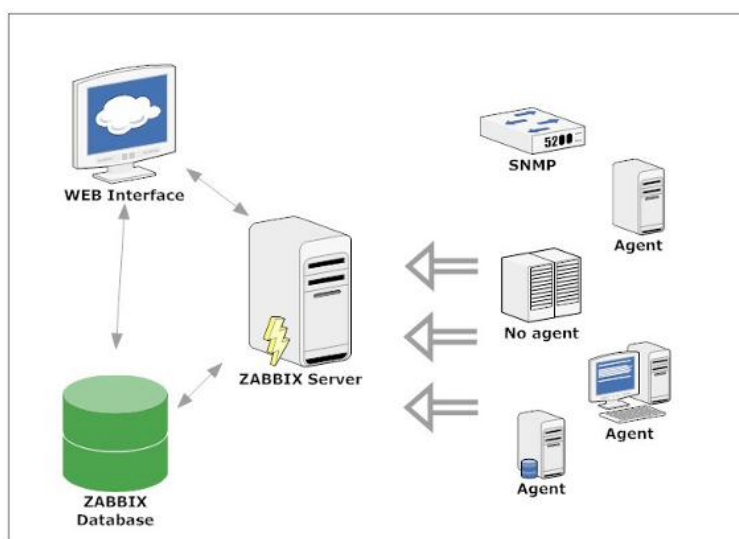


Obrázek 14 – Výsledný RRDTOOL graf

V průběhu pollování dat jsem zapnul LED diodu a po nějaké době ji zase vypnul.

### 4.3 Zabbix

Zabbix slouží k monitorování aktivních síťových prvků (PC, servery, tiskárny, modemy, switche, UPS,...), které jsou připojeny do počítačové sítě. Je tedy možné říci, že můžeme sledovat stav a sbírat různé informace o všem, co má IP adresu. Metody pro sledování a zjišťování informací jsou různé. Počínaje jednoduchým ICMP echo request (ping) přes použití složitějších metod SNMP (Simple Network Management Protocol), IPMI (Intelligent Platform Management Interface), JMX (Java Management Extensions). Je možné použít také k monitoringu SSH/Telnet a nebo použití agenta, který je dostupný pro většinu dnes používaných operačních systémů. Při použití agenta je možné monitorovat informace o stavu hardware (operační paměť, procesor, úložné zařízení,...), ale také systémové informace a stav běžících služeb. V neposlední řadě je možné integrovat do prostředí vlastní externí skripty anebo využít API a vytvořit si tak vlastní testy. Pomocí API lze také komunikovat s jinými nástroji. Co se týká počtu dohlížených zařízení, dle tvůrců, je možné monitorovat přes 100 000 hostů a provádět tak 1 000 000 vyhodnocení za minutu, což je pochopitelně závislé na systémových zdrojích serveru, na kterém je dohledový systém provozovaný. Zabbix může pracovat distribuovaně, což znamená, že v různých vzdálených lokalitách běží Zabbix v režimu proxy a data se následně přenášejí na centrální server. To je vhodné pro velmi robustní a rozsáhlé sítě s velkým počtem zařízení. Dohledový systém je přístupný z webového rozhraní, které slouží zároveň i jako administrační prostředí pro správu a vyhodnocení dat. [8]



Obrázek 15 – Topologie sítě Zabbix [8]

### 4.3.1 Proč používat Zabbix

Dohledových systému je celá řada. Uvedu zde nejpodstatnější body proč používat dohledový systém Zabbix především z technického hlediska. [8]

- rychlost
- otevřenost (licence GPL)
- přehledné prostředí
- intuitivní ovládání
- živý a rychlý vývoj
- mnohostrannost (možné sledovat prakticky cokoliv)
- podpora (fórum, dokumentace, ...)

The screenshot displays the Zabbix web interface. At the top, there is a navigation menu with options like 'Monitoring', 'Inventory', 'Reports', 'Configuration', and 'Administration'. Below this, a 'PERSONAL DASHBOARD' is visible, containing several widgets:

- Status of Zabbix:** A table showing system parameters such as 'Zabbix server is running' (No), 'Number of hosts' (63), and 'Number of triggers' (242).
- System status:** A table showing the status of various host groups like 'Clouds', 'Clusters', 'Database Servers', etc., with columns for Disaster, High, Average, Warning, Information, and Not classified.
- Host status:** A table showing the status of individual hosts, with columns for 'Without problems', 'With problems', and 'Total'.
- Last 20 issues:** A table showing the most recent monitoring issues.
- Web monitoring:** A table showing the status of web services, with columns for 'Ok', 'Failed', and 'Unknown'.

Obrázek 16 – ukázka rozhraní aplikace Zabbix [8]

### 4.3.2 Server

Jedná se o hlavní proces, který řídí logiku celého systému, zajišťuje veškeré činnosti. Jde například o sběr dat a vyhodnocení dat, zápis do databáze a zajištění integrity dat. Server také slouží k zadávání pokynů agentovi pro provedení testů a akcí (restart, vypnutí služeb,...). Je psaný v jazyku C, a pro svůj běh potřebuje jen malé množství systémových prostředků. [8]

### 4.3.3 Agent

Je klientský proces běžící na hostech. Sbírá informace a průběžně je předává serveru. Má velmi malou režii pro svůj běh a stejně jako server je psaný v jazyku C. Vedle sběru dat slouží k provádění akcí např. restart/zastavení běžících služeb. Množství informací, které agent umí sledovat, je závislé na konkrétním operačním systému a verzi. Základní informace o systému, jako je velikost RAM, sledování místa na disku, informace o souborech (velikost, přístup), jsou podporovány všude. Pro přesný výčet funkcí konkrétního operačního systému je nutné obrátit se na aktuální dokumentaci Zabbixu. [8]

### 4.3.4 Proxy

Pokud pracuje Zabbix v režimu proxy, plní částečnou funkci serveru. Místo centrálního serveru sbírá informace o daných hostech, které následně předává centrálnímu serveru. Nahrazuje tak funkci serveru v případech, kdy existuje mnoho lokalit s velkým počtem zařízení, čímž dojde mimo jiné k odlehčení serverové zátěže. Je to jedna z možností distribuovaného monitoringu. Druhou možností je použití Nodů, což bude podrobněji probráno v pozdější části seriálu. [8]

### 4.3.5 Sběr dat

Nejdůležitější funkcí dohledového systému je zajisté sběr dat ve stanoveném časovém intervalu a vyhodnocení těchto dat podle určitých podmínek a závislostí. Zabbix splňuje požadavky na pro-aktivní monitoring, umožňuje pomocí eskalace a scénářů provádět opakovanou notifikaci a automatické řešení problému pomocí vzdálených akcí. [8]

### Příklad scénáře při výpadku služby na serveru

- Poslat zprávu IT administrátorům.
- Pokud není služba dále dostupná restartovat službu.
- Pokud není stále dostupná poslat zprávu IT administrátorům a restartovat server.
- Pokud není služba stále dostupná posílat zprávu IT administrátorům v určitém intervalu do vyřešení problému.
- možnost opakované notifikace do vyřešení problému
- zpožděná notifikace v případě dlouhodobého trvání problému
- notifikace skupinám / uživatelům dle eskalace problému
- kombinace všech možností a tvorba komplexních scénářů

[8]

## 4.4 Cacti vs. Zabbix

Po pročetí několika fór a webových stránek [10] zabývajících se monitorovacími systémy jsem sepsal pozitivní a negativní ohlasy na tyto dvě monitorovací aplikace.

### Cacti

#### Výhody

- Krásné a (pro většinu funkcí) jednoduché webové rozhraní. Pěkné funkce, jako grafy, které lze zvětšit pomocí Javascriptu.
- Povedený systém uživatelů ovládajících Cacti. Každý uživatel může získat přístup jen pro čtení k určitému podstromu grafů.
- Strom, kde jsou grafy umístěny, může být nakonfigurován volně, takže máte přesně požadované zobrazení grafů.

#### Nevýhody

- Používání šablon a datových zdrojů není příliš intuitivní a uživatele bez prostudování dokumentace bude často tápat

- Grafování někdy bezdůvodně přestane pracovat nebo se ztratí hodnoty, přestože server není přetížen. Podle rychlého hledání na internetu se to stává hodně uživatelům.
- Nastavování mnoha systémů je zdlouhavé a nastavování nových šablon ještě zdlouhavější a náchylné k chybám
- Kvalita některých šablon třetích stran je podle ostatních uživatelů dost špatná. Vytvoření nové šablony je únavné, náchylné k chybám, frustrující a blízko k černé magii. Nic pro příležitostné uživatele.
- Frekvence kontrol je standardně 5 minut. Zvýšení frekvence vede k chybějícím údajům a špatným výsledkům.
- Jelikož se používá RRD a jsou potřeba tři platné hodnoty, není vidět, že vaše sledování selhalo, dokud nepočkáte  $3 \times 5 = 15$  minut.

## Zabbix

### Výhody

- Monitoring dostupnosti (jako Nagios) a grafů (jako Cacti) jsou sloučeny do jednoho nástroje.
- Vysoce konfigurovatelný. Uživatel může např. dostat SMS při problému s vysokou závažností v průběhu víkendu a ve všední dny zprávu na Jabber. Dokonce i automatické akce, jako je restartování služeb lze nastavit.
- Shromažďovaná data jsou uložena v databázi (MySQL, PostgreSQL, SQLite) místo na neflexibilního RRD souboru. Skladovací období (historie) lze volně konfigurovat. Zálohování databáze je všechno, co je třeba udělat.
- Systém uživatelů. Uživatelé mohou být omezeni jen na určité zobrazení.
- Grafy v reálném čase. Hodnoty jsou standardně shromažďovány každých 30 sekund. Můžete rychle zjistit, kam jdete.
- Rychlé webové rozhraní.
- Šablony (které mohou odkazovat i na další šablony) ušetří čas při vytváření mnoho kontrol.

- Grafy vykreslují hodnoty v průběhu času a mohou být upravovány. Například, jestli jsou položky vykresleny a jakým způsobem je vykreslujeme. Dokonce i koláčové grafy jsou možné.
- Tři sta dvacet stran PDF manuálu se screenshoty a pěknými odkazy.

### Nevýhody

- Je nutné mnoho pohybů myši a klikání pro nastavení věcí. Například zřízení upozornění v případě, že volné místo na disku na určitém serveru, je příliš nízké, musíte nastavit hostitele, předměty, triggery a akce. Některé klikání se zdá být nadbytečné.
- Chce to trochu trpělivosti pochopit jeho koncept, což se ale dá zvládnout i za půl dne.
- Webové rozhraní je nacpané funkcemi. Pro běžného uživatele je to matoucí při navigaci. V reálném životě síti zjistíte, že při nastavení hostitelských proměnných, ošizování šablon si nezapamatujete všechno, co jste udělali, nebudete mít dobrý přehled o vaší konfiguraci. Zabbix je velmi komplexní, ale bude potřebovat ještě lepší webové rozhraní, aby se správně vypořádal se svými funkcemi.
- Těžké ladění. Proč akce nelze spustit? Kdo měl být upozorněn na určitý trigger? Proč se hodnota stala "neznámou" bez důvodu? Samozřejmě, že existují důvody pro to, co Zabbix dělá. Ale zabere to hodně klikání a hádání místo tohoto aby to bylo uživateli sděleno.

## ZÁVĚR

SNMP je velmi rozšířený a využívaný protokol, jeho implementace pro private MIB byla ale možná až po prostudování jeho problematiky a především po podrobném pročtení zdrojového kódu LwIP stacku, týkajícího se SNMP a několika diskuzí fór o LwIP.

Vývojová deska po implementaci LwIP bezproblémově odpovídala na požadované příkazy *get* a *set* na OID z implementovaného private MIB. Dalším krokem tedy bylo proměnné propojit s námi požadovanými zdroji dat, jako jsou senzory a jiná data, která chceme monitorovat a sestavit tak konkrétní implementaci MIB pro naše embedded zařízení. V mojí implementaci byly reálně využity pouze dvě proměnné pro indikaci stavu LED diod a ostatní byly jen staticky uložená data, neprovázaná s reálnými proměnnými. Ovšem po připojení například UPS modulu k desce, pro který byl private MIB navržen, by nebyl žádný problém provázat tyto proměnné s modulem a přes SNMP sledovat jeho stav (např. zda je deska napájena z baterie, stav baterie, dobu chodu na baterii, atd.).

Současná implementace odpovídá SNMPv1, pouze s tím rozdílem, že si můžeme definovat libovolná „community string“ (heslo), které je ovšem v paketu nezašifrované a tedy snadno čitelné (viz. Obr.6). Tudíž je tato implementace vhodná spíše na monitoring než správu zařízení. Záleží ale na konkrétním embedded zařízení, pokud by například do sítě, ve kterém se zařízení nachází, měly přístup pouze autorizované osoby, nebyla by to tak velká bezpečnostní mezera. Ovšem na kompletní správu sítě přes SNMP by bylo potřeba implementovat nějakou formu šifrování hesla, nejlépe rovnou standard SNMPv3.

Ve chvíli, kdy máme na embedded zařízení implementovaného SNMP agenta, lze jednoduše využít jednu z dostupných SNMP web aplikací, v mém případě jsem zvolil Cacti, která příslušná data monitoruje, ukládá do databáze a poté zpřístupňuje na WWW stránkách v grafové reprezentaci. Nastavení Cacti nebylo úplně snadné, jako se podle dokumentace zdálo, ale tato aplikace poskytuje i další použití pluginů nebo šablon a skriptů vytvořených jinými uživateli. Ovšem při porovnání spokojenosti a zkušeností dlouhodobých uživatelů monitorovacích aplikací se lépe umístila aplikace Zabbix, kterou jsem osobně nevyzkoušel, ale už její rozhraní vypadá daleko intuitivněji a nepoužívají se šablony ve stylu Cacti, které jsou asi nejhorším prvkem Cacti.

Hlavní program, který byl vytvořen jako součást této práce, lze upravit pro libovolnou aplikaci embedded zařízení, stačí pouze pozměnit nadefinované proměnné (OID, datové typy,

...), provázat je s reálnými hodnotami a přes SNMP provádět monitoring. Což bylo mým hlavním cílem.

**SEZNAM POUŽITÉ LITERATURY**

- [1] MAURO, Douglas R a Kevin J SCHMIDT. Essential SNMP. 2nd ed. Beijing: O'Reilly, 2005, 442 s. ISBN 05-960-0840-6.
- [2] SKŘIVÁNEK, Lukáš. FEL ČVUT. *36MPS - SNMP* [online]. 2006, Dostupné z: <http://skriv.wz.cz/MPS/SNMP/index.html>
- [3] Samuraj-cz. BOUŠKA, Petr. *Samuraj-cz* [online]. 2014 [cit. 2014-06-08]. Dostupné z: <http://www.samuraj-cz.com/clanek/snmp-simple-network-management-protocol/>
- [4] *Svět sítí* [online]. 2014 [cit. 2014-06-08]. Dostupné z: <http://www.svetsiti.cz/clanek.asp?cid=Format-SNMP-zprav-1462000>
- [5] *SAMURAJ-CZ* [online]. 2006 [cit. 2014-04-23]. Dostupné z: <http://www.samuraj-cz.com/clanek/cacti-snmp-monitoring-a-grafy/>
- [6] *Simpleweb* [online]. 2014 [cit. 2014-06-08]. Dostupné z: [http://www.simpleweb.org/w/images/9/91/Tutorial\\_Slides\\_Smi.pdf](http://www.simpleweb.org/w/images/9/91/Tutorial_Slides_Smi.pdf)
- [7] *The Net-SNMP Wiki* [online]. 2011. The Net-SNMP Wiki. Dostupné z: <http://net-snmp.sourceforge.net/wiki/>
- [8] *Linuxsoft* [online]. 2013 [cit. 2014-06-12]. Dostupné z: [http://www.linuxsoft.cz/article.php?id\\_article=1963](http://www.linuxsoft.cz/article.php?id_article=1963)
- [9] *Wikipedia*. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014 [cit. 2014-06-13]. Dostupné z: [http://en.wikipedia.org/wiki/Comparison\\_of\\_network\\_monitoring\\_systems#cite\\_note-nagiosipv6-2](http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems#cite_note-nagiosipv6-2)
- [10] *Workaround* [online]. 2014 [cit. 2014-06-13]. Dostupné z: <https://workaround.org/article/tired-of-nagios-and-cacti-try-zabbix>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

SNMP	Simple Network Management Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
NMS	Network Management Station
IETF	Internet Engineering Task Force
MIB	Management Information Base
OID	Object Identifier
RFC	Request For Comments
IANA	Internet Assigned Numbers Authority
LwIP	Lightweighth TCP/IP stack
RRDTool	Round-Robin Database Tool
WMI	Windows Management Instrumentation
MRTG	Multi Router Traffic Grapher
OS	Operační systém
MCDA	Multiple-Criteria Decision Analysis

**SEZNAM OBRÁZKŮ**

<i>Obrázek 1 – TCP/IP komunikační model a SNMP [1]</i> .....	11
<i>Obrázek 2 – vztah mezi správcem (NMS) a agentem [1]</i> .....	12
<i>Obrázek 3 – Stromová struktura MIB</i> .....	13
<i>Obrázek 4 – formát SNMP zprávy [4]</i> .....	17
<i>Obrázek 5 – Prostředí vývojového nástroje Taskin VX-toolset for ARM</i> .....	23
<i>Obrázek 6 – SNMP dotaz zachycený programem Wireshark</i> .....	24
<i>Obrázek 7 – Peassler SNMP Tester 5.0</i> .....	25
<i>Obrázek 8 – ukázka rozhraní aplikace Cacti</i> .....	34
<i>Obrázek 9 – Cacti Devices</i> .....	36
<i>Obrázek 10 – Cacti Data Queries</i> .....	37
<i>Obrázek 11 – Cacti Graph Template</i> .....	38
<i>Obrázek 12 – Data Queries Assoc. Graph</i> .....	38
<i>Obrázek 13 - Cacti Device Edit</i> .....	39
<i>Obrázek 14 – Výsledný RRDTool graf</i> .....	39
<i>Obrázek 15 – Topologie sítě Zabbix [8]</i> .....	40
<i>Obrázek 16 – ukázka rozhraní aplikace Zabbix [8]</i> .....	41

**SEZNAM TABULEK**

<i>Tabulka 1 – datové typy MIB</i> .....	17
<i>Tabulka 2 – časy pro vyřízení SNMP požadavků</i> .....	30
<i>Tabulka 3 – Multikriteriální analýza 1 [9]</i> .....	32
<i>Tabulka 4 – Multikriteriální analýza 2 [9]</i> .....	33

## SEZNAM PŘÍLOH

- P I Zdrojové kódy aplikace
- P II Cacti
- P III Použitý software

## **PŘÍLOHA P III: POUŽITÝ SOFTWARE**

Tasking VX-toolset for ARM	Vývoj aplikace
SEGGER J-Link GDB server	Software pro programátor J-link
Wireshark	Sledování SNMP paketů
Peassler SNMP Tester	Zasílání SNMP get dotazů
SNMPGetSet	Zasílání SNMP get a set dotazů
PowerSNMP Free Manager	Prostý SNMP manager
Microsoft Word 2007	Sazba práce
Cacti	Open-source SNMP web aplikace
Apache	HTTP server
NET-SNMP	SNMP knihovny