České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

Řízení technologických procesů v koncepci IoT

Process control in the IoT concept

Bakalářská práce

Studijní program: Komunikace, Multimédia a Elektronika

Studijní obor: Síťové a informační technologie

Vedoucí práce: Ing. Lukáš Vojtěch, Ph.D.

Ivan Eroshkin

Praha 2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Eroshkin** Jméno: **Ivan** Osobní číslo: **434675**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra telekomunikační techniky**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Síťové a informační technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Řízení technologických procesů v koncepci IoT

Název bakalářské práce anglicky:

Process control in the IoT concept

Pokyny pro vypracování:

Navrhněte a realizujte nízkonákladový systém pro dohled nad technologií galvanického vylučování kovů v laboratorním pojetí. Využijte síťových a komunikačních technologií pro aplikace Internetu věcí, včetně základního řešení problémů spolehlivosti a bezpečnosti. Pro ukládání měřených dat a jejich zpracování analyzujte možnosti užití lokálních pamětí, serverových aplikací či služeb Cloudu.

Seznam doporučené literatury:

[1] Schlesinger, M.; Paunovic, M.: Modern Electroplating 5th Edition, Amazon Prime, SBN: 978-0470167786.
[2] Dokumentace dostupná na http://www.iqrf.org [on-line]
[3] Dokumentace dostupná na http://vocore.io/ [on-line]
[4] Dokumentace dostupná na http://www.manson.com.hk/products/detail/158 [on-line]

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Lukáš Vojtěch Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2017** Termín odevzdání bakalářské práce: **26.05.2017**

Platnost zadání bakalářské práce: **30.09.2018**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání	Podpis studenta

Anotace:

Tato práce se zabývá návrhem nízkonákladového systému v koncepci Internetu věcí pro dohled nad technologii galvanického vylučování kovů. Cílem je navrhnout funkční systém, který bude sbírat data a následně je zobrazovat. Součásti práce je též řešení problému spolehlivosti a bezpečnosti navrženého systému, včetně řešení problematiky ukládání dat v rámci omezené paměťové kapacity.

Klíčová slova:

Internet věcí, průmysl 4.0, analýza v reálném čase, kompilace OpenWRT/LEDE, Lua, LuCI, rrdtool, návrh systému, galvanoplastika, DS18B20, embedded zařízení, VoCore, Manson HCS-3400 USB, bakalářská závěrečná práce.

Summary:

This work deals with creating a system for controlling of electroforming process in IoT concept. The main aim is to create working system which can collect and show this data. Part of this work is also solving problem of reliability and security of created system and smart data storing in case of small memory capacity.

Keywords:

Internet of things, Industry 4.0, real-time analysis, compile of OpenWRT/LEDE, Lua, LuCI, rrdtool, creating a system, electroforming, DS18B20, embedded platforms, VoCore, Manson HCS-3400 USB, bachelor work.

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

V Praze dne 25.5.2017

……

podpis bakalanta

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Lukáši Vojtěchovi za trpělivost a vstřícnost při tvorbě této práce, panu Ing. Zbyňku Kocurovi za jeho rady a konzultace při implementace práce. V neposlední řadě děkuji mým přátelům a rodině za podporu po celou dobu mého studia.

Obsah

1. Internet věcí

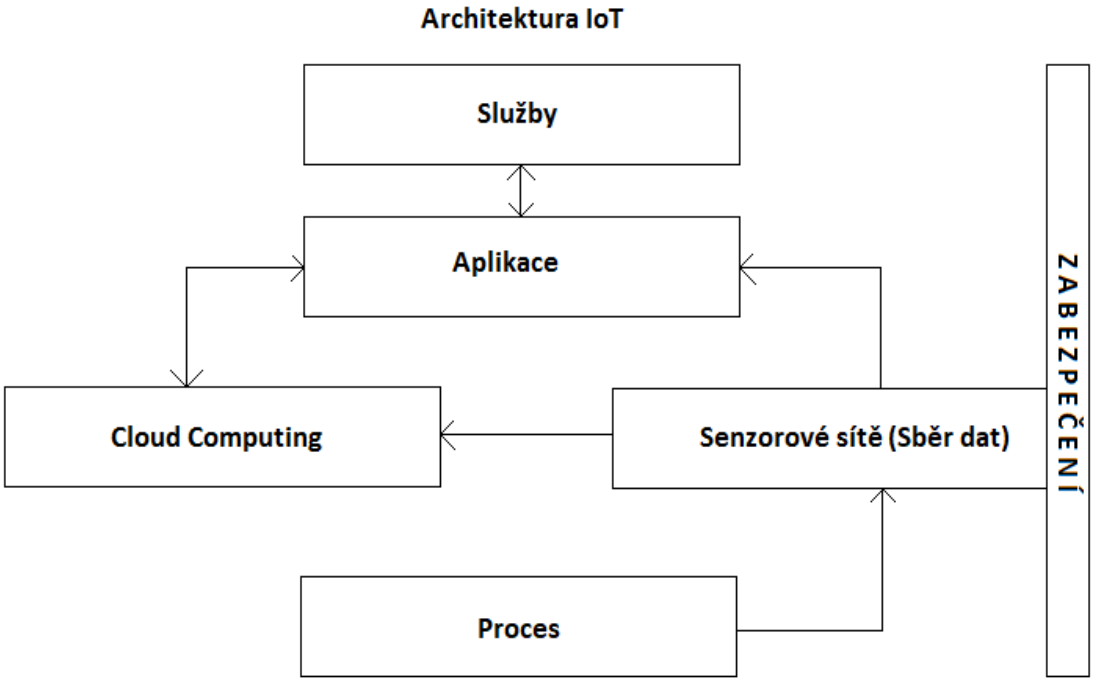
Pojem Internet věcí (Internet of Things - IoT) se v dnešní době využívá víc a víc. Skoro každá firma se snaží něco vytvořit a říct, že se to vztahuje k Internetu věcí. Co to vůbec je? Původní myšlenka je, že všechny věci by měly nějak spolu komunikovat a někdo by je měl řídit. Takovou definici v roce 1999 řekl pan Kevin Ashton [1] (zakladatel výzkumné skupiny Auto-ID, Massachusetts Institute of Technology (MIT)). V roce 2013 na konferenci CISCO pan Bryce Barnes [2] definoval Internet věcí takto: „Je to chytré spojení chytrých zařízení, kterým se objekty mohou navzájem vnímat a komunikovat“. Pojem „Internet věcí“ je o tom, že všechny věci nějak mezi sebou spolu komunikují a „žijí“ svým vlastním digitálním životem.

Dnes se na vývoji IoT podílejí špičkové firmy, jako jsou: AT&T, IBM, Google, Intel, Cisco, Oracle, Microsoft, Texas Instruments, ARM & Atmel, Bosch, Samsung, Apple, Ericsson, GE a další.

V současné době se začali objevovat produkty v portfoliu některých společností, které poskytují služby fungující v koncepci Internetu věcí. Jsou to většinou nabídky na pronájem kapacity a služeb ve vlastním data centru. Existují i společnosti, které řeší každý případ individuálně a vytváří unikátní řešení pro své zákazníky. První případ se vztahuje zejména k telekomunikačním operátorům, které se živí zejména z poskytování služeb a přenosových kapacit. Druhý případ je o společnostech, které investují do rozvoje embedded zařízení, nebo produkují vlastní platformy na zakázku. Tyto společnosti se také zabývají programováním vlastních výrobků a následně poskytováním služeb na základě navržených systému. Z výše uvedené definice a současného stavu trhu v tomto segmentu plyne, že Internet věcí je zejména o službách.

Pokud se podíváme do průmyslu, zjistíme, že již dlouho předpovídá 4. průmyslová revoluce – Industry 4.0. Základní myšlenkou je snaha zajistit co nejefektivnější a nejvýhodnější výrobu z hlediska nákladu, času a minima chyb vzniklých při výrobě. Hlavním směrem je automatizace procesů a jako důsledek odstranění lidského faktoru, který vyvolává chyby a možné ztráty továrny. Průmyslovou utopii je plně automatizovaná továrna, která samostatně navrhne, vyrobí a doručí zboží zákazníkovi, který toto zboží objedná přes chytrou aplikaci, poskytovanou továrnou, nebo internet. Je zřejmě, že taková továrna vůbec nepotřebuje zaměstnanci a je schopná samostatně naplánovat veškerý provoz [3]. Je to utopie, která je dnes trendem a ke které se směrují výrobci. Pro tyto účely se dnes navrhují monitorovací systémy, které sbírají data ze strojů, resp. konaných procesů a okamžitě se zobrazují v tzv. real-time analýze, což jsou většinou průběhy důležitých parametrů procesů nebo statistické údaje. Občas se aplikují i výpočty, pokud sledované procesy jsou relativně komplikované nebo je požadovaná určitá predikce.

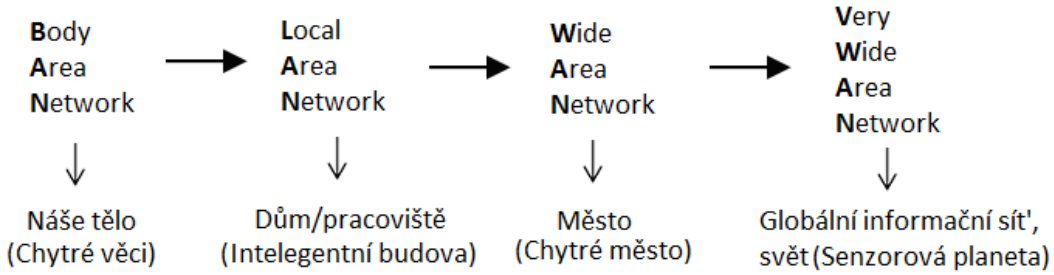
Na následujícím obrázku jsem se pokusil zobrazit architekturu Internetu věcí:



Obrázek 1

Z obrázku je patrné, že dokážeme-li pomocí senzorů nasbírat data z nějakého procesu, zobrazíme jejich průběh, na základě kterého bychom mohli sledovat a maximálně efektivně řídit proces, dokonce ho ovládat pomocí nějaké aplikace.

Rob van Kranenburg [1] (zakladatel evropské rady pro Internet věcí - IoT EU council) předpovídá IoT velkou budoucnost a popisuje 4 kroky rozvoje Internetu věcí (viz.: následující obrázek).



Obrázek 2

Zatím jsme ve druhém kroku a snažíme se dostat k třetímu. V budoucnu se Internet věcí bude týkat každého aspektu běžného života. To může být:

- *Chytrý průmysl* – M2M monitorování;
- *Chytrá doprava* – chytré řízení provozu;
- *Chytré zdravotnictví* – detekce pohybu, sledování pacientů, inteligentní přístroje;
- *Chytré prostředí* – detekce požáru, znečištění;
- *Inteligentní budova*;
- *Chytré zabezpečení;*
- *Jiné obory*.

V této práci jsem se snažil aplikovat výše zmíněnou koncepci na použití Internetu věcí v průmyslu, kterým bylo pro příklad zvoleno galvanické vylučování kovů, jako nejvhodnější proces pro návrh automatizačního systému.

2. Krátce o galvanoplastice

Proces galvanoplastiky anebo elektroformování (z anglického „electroforming“) je využíván mnoho desítek let. Jedná se o formování různých geometrických tvarů pomocí elektrochemického vylučování kovových povlaků na primární model. Využívá se elektrolýza, která umožňuje vytvářet vrstvy tloušťkou od desítek mikrometrů až po jednotky milimetrů [4].

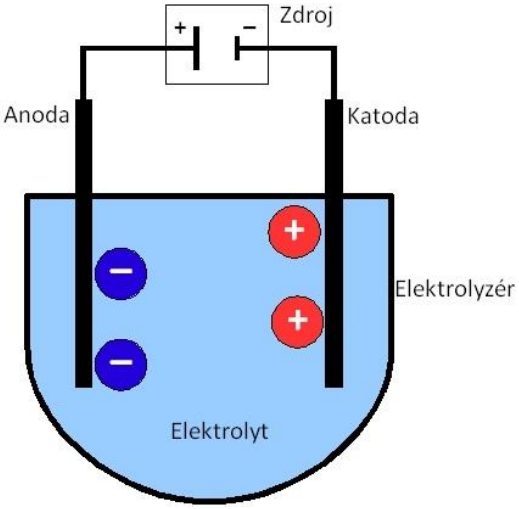
Primární model bývá nejčastěji snímatelný – vytavením nebo oddělením – z čeho vzniká velice kvalitní a přesné otisky nebo formy původního modelu. Tato technologie umožňuje produkovat vysoce kvalitní detaily, výroba kterých klasickými strojírenskými způsoby je velice obtížná.

Galvanoplastiku objevil německý a ruský vědec Moritz Hermann von Jacobi v roce 1836. V tuto dobu elektroformování se využívalo zejména v umění, pro výrobu soch, které jsou i dodnes vyráběny tímto způsobem s použitím mědi.

Galvanoplastika se využívá v mnoha oborech v současnosti a to jsou:

- Strojírenství;
- Elektrotechnika;
- Umění;
- Fyzika;
- Letecký průmysl;
- Kosmický průmysl.

Podstatou tohoto procesu je tzv. elektrolýza. Jedná se o chemickou reakci, která se koná v elektrolytu – vodivé kapalině obsahující směs kationtů a aniontů. K tomu, aby se tato reakce konala, potřebujeme zapůsobit elektřinou na elektrolyt. Zpravidla se využívá stejnosměrný proud. Důsledkem toho jsou chemické změny na elektrodách. Působení elektrického proudu vyvolává pohyb kladných iontů k záporné elektrodě (katoda) a záporných iontů ke kladné elektrodě (anoda). Pro zjednodušenou představu, schématický průběh elektrolýzy je uveden na následujícím obrázku. [5]



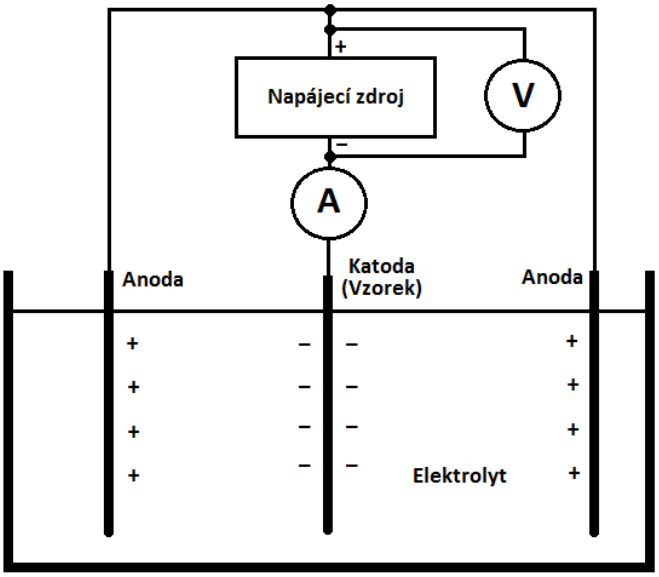
Obrázek 3 [6]

V galvanoplastice vzorek, který chceme pokrýt vrstvou, hraje roli katody a je umístěn mezi dvěma anodami ve vzdálenosti 15-20 cm pro pokrytí s obou stran. Anody by měly být z látky, která pokryje vzorek. Při průchodu elektrického proudu od anod se odštěpují molekuly, které se usazují na katodě a tím vytváří vrstvu. [7]

Pro úspěšné použití musíme dodržovat několik podmínek:

1. Vzorek musí být elektrický vodivý. Pokud není, pokryjeme ho tenkou vrstvou vodivé látky. V případě, že chceme pokrýt jenom vybrané oblastí vzorku, pokryjeme je vodivou vrstvou, resp. pokryjeme nevodivou vrstvou nepožadované oblasti, aby nedocházelo k osazení molekul na nepožadovaných místech.
2. Vana nesmí být elektricky vodivá. Jinak dojde k pokrytí povrchu stěn vany a neefektivnímu využití látky.
3. Čím výrazněji je reliéf vzorku, tím dále musí být umístěn od anody.

Na následujícím obrázku je uvedeno schéma zapojení pro galvanoplastiku.



Obrázek 4

1. Faradayův zákon elektrolýzy umožňuje odvodit teoretickou hmotnost vyloučené z katody látky:

$m = A \cdot I \cdot t$, kde m – je hmotnost vyloučené látky, A – je elektrochemický ekvivalent látky, I – je proud, t – je čas konání procesu (nebo čas působení proudu) [4]. Stoji za zmínku uvést ten fakt, že v praxi je vždycky teoretická hmotnost větší, než faktická. To je dáno přitomnosti přiměsí v elektrolytu, které se nevyuživají během reakci.

Z toho lze odvodit hmotnost mědi pro vytvoření vrstvy určité tloušťky, proud a čas, nutný na tento proces. m = S*0,9*C, kde m je hmotnost mědi [g], S je plocha vzorku [cm^2], C je požadovaná tloušťka mědi [mm]. [8]

S použitím této hmoty lze hned vypočíst čas dle vztahu: T = m/(1,2*I), kde T je čas [hodiny], I je proud [A], m je hmotnost mědi [g]. [8]

Proud a čas jsou vzájemně úměrné: čím větší proud dodáme, tím menší čas potřebujeme na vytvoření vrstvy. V praxi se při dodání příliš velkého proudu zpravidla zhoršuje kvalita vrstvy.

I = J*S, kde I je proud [A], J je proudová hustota [A/dm^2], S je plocha povrchu vzorku [dm^2]. [8]

Na závěr bych zmínil, že se rozlišuje několik druhů galvanoplastiky:

- Galvanostegie – vytváření tenké kovové vrstvy na kovovém prvku (zejména pro ochranu železa proti korozím)
- Elektrická obnova detailu – jedná se o velice perspektivní odvětví, které umožňuje obnovit staré kovové součástky „nasycením“ určitou látkou.

3. Stanovení cíle a požadavků

Cílem této práce je návrh systému v koncepci Internetu věcí, který by umožnil zjednodušené sledování a řízení procesu galvanického vylučování kovů. Informaci uvedenou v kapitole 2 potřebujeme k navržení analýzy.

Hlavními požadavky, na které byl kladen důraz, jsou následující:

1. Umožnění sledování důležitých parametrů procesu, jako je teplota, průběh proudu a napětí, otáčky ventilátoru ve ventilaci a vykreslení grafů.
2. Snadný přístup ke statistice.
3. Efektivní komunikace zařízení mezi sebou a jejich konfigurace.
4. Spolehlivost systému.
5. Volba vhodných součástek.
6. Cena.

4. Embedded zařízení

Pro dodržování podmínky nízkých nákladů budeme navrhovat systém na základě embedded zařízení, které můžeme snadno zakoupit za relativně nízké ceny. Jsou to zařízení malých rozměrů, navržené pro nějaký konkrétní účel s výrazně redukovaným hardwarem oproti počítači (PC) a podstatně menším výkonem, který je skoro přesně postačující na požadovanou úlohu. Zpravidla neobsahuje žádné grafické prostředky, prostředky pro ovládání, grafickou kartu, nebo obsahuje její velice redukovanou podobu, nemá síťové karty atd. Klasickou náplni podobného zařízení je samotný procesor, GPIO porty na připojení dalších potřebných součástek, resp. modulů, vestavená pevná paměť malé kapacity, antény pro komunikaci nebo přenos dat a další bezdrátové spojení (např. Wi-Fi).

V IoT se převážně používá embedded zařízení z důvodu malé spotřeby energie, docela postačujícího výkonu a nízkých cen oproti hotovým řešením monitorovacích systémů poskytovaných velkými výrobci. [9]

V rámci této úlohy jsem taky využil embedded zařízení pro návrh systému. Hlavním důvodem byla nízká cena, postačující výkon na splnění stanovené úlohy a již existující nástroje na sbírání a znázornění dat. Více v následujících kapitolách.

5. VoCore

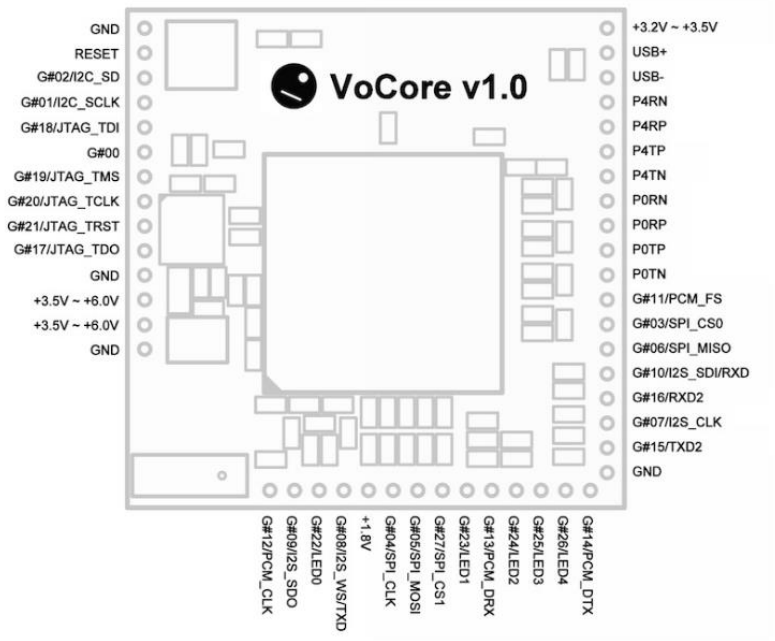
Zařízení VoCore bylo vyvinuto v Čině návrháři Qin Wei, Tong Wu a Thomasem Hommersem [10]. První novinky se datují prosincem roku 2013. Zveřejnění a zahájení výroby se uskutečnilo na podzim roku 2014. Od této doby bylo vyvinuto dvě verze VoCore a několik modifikací. Krátce o každé z nich zmíníme v dalších podkapitolách.

5.1 VoCore

První verze VoCore se objevila na podzim roku 2014 a získala široky zájem u fanoušků embedded platforem. Toto zařízení má rozměr o trochu větší než mince, nízkou cenu a poměrně dobrý výkon. Charakteristiky jsou uvedeny na následujícím obrázku.

SIZE	25.6mm x 25.6mm x 3.0mm
CPU	RT5350, 360 MHz, MIPS 24K
MEMORY	32MB, SDRAM
STORAGE	16M NOR on board
WIRELESS	802.11n, 1T1R, speed up to 75Mbps.
ANTENNA	one on board antenna.
ETHERNET	5 ports, up to 100Mbps.
USB	Support USB 2.0, up to 480MHz.
GPIO	>=30 (pinmux)
UART	x2 (UART1 for debug console)
POWER SUPPLY	3.6V ~ 6.0V
POWER CONSUMPTION	~130mA~190mA, 5V input.

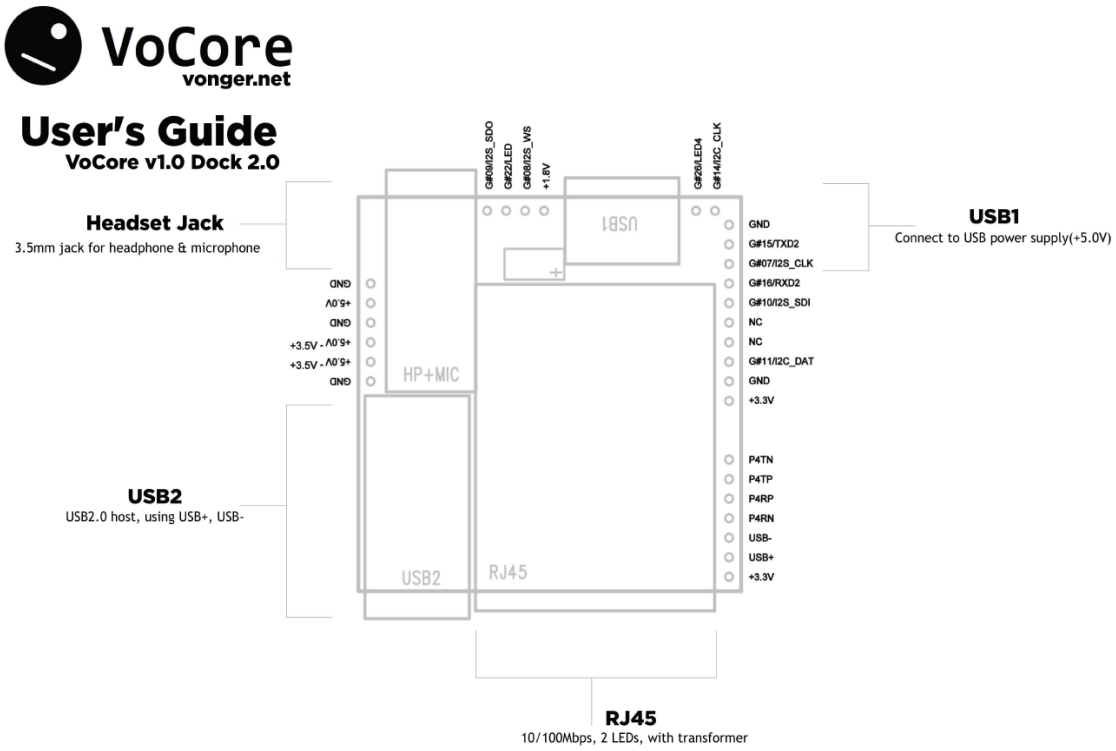
Obrázek 5 [11]



Obrázek 6 - schéma zařízení [12]

5.1.1 VoCore + Dock

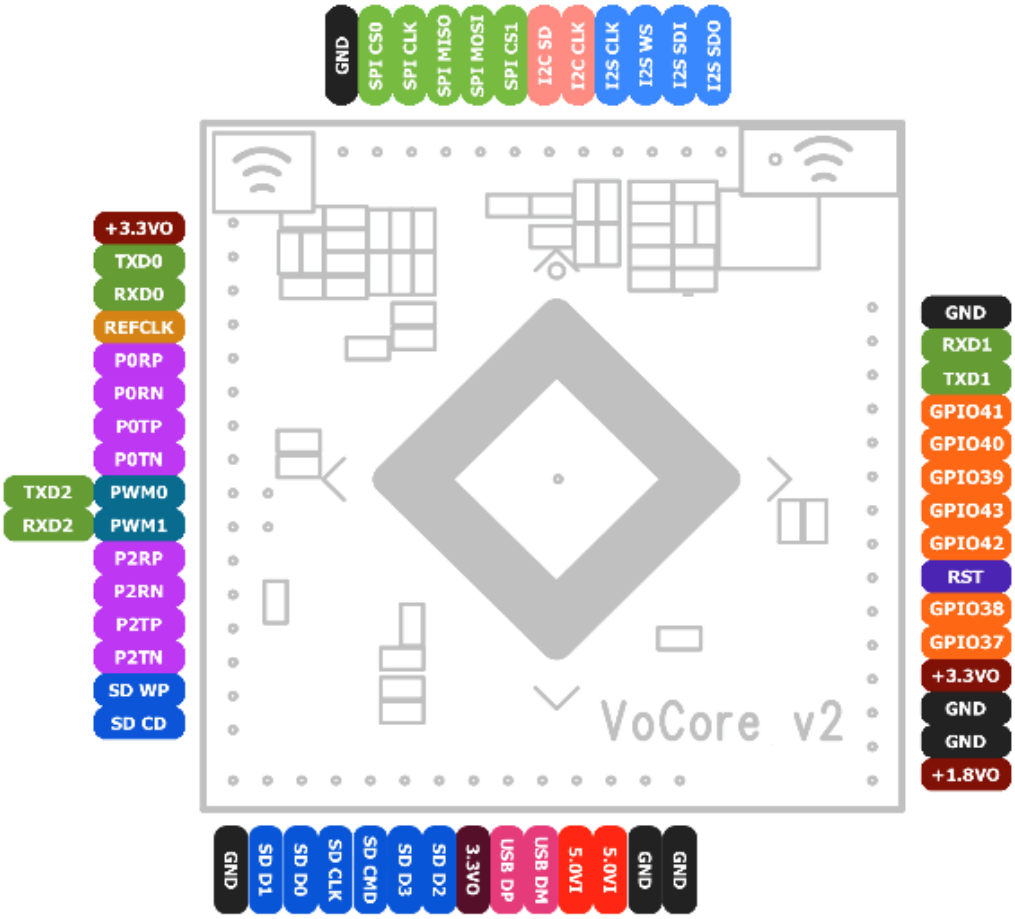
Základní desku je možné rozšířit o tzv. Dock station, který poskytne více prostoru a jednoduchost pro připojování dalších modulů. Jedná se o dodatečný funkční blok, který je pevně spojen s původním plošným spojem VoCore. Tento blok obsahuje 2 porty USB 2.0 – jeden na napájení, druhý na komunikaci, 1x Ethernet, 1x slot pro SD kartu a audio výstup [11]. Funkční schéma Dock station je následující:



Obrázek 7 [12]

5.2 VoCore2

Na podzim roku 2016 byl na startovači Indiegogo [13] zahájen výběr peněz na výrobu druhé verze VoCore. Tentokrát výrobce představil několik modifikací: od levné verze pro studenty (VoCore2 Lite), až po již populární VoCore s Dock Station (tedy se nazývá VoCore2 Ultimate). Ve srovnání s VoCore byl použit výkonnější procesor, větší paměťová kapacita, dvě antény oproti jedné v první verzi, větší přenosová rychlost bezdrátového spoje Wi-Fi (až do 300 Mb/s), poklesla spotřeba elektrické energie [13].



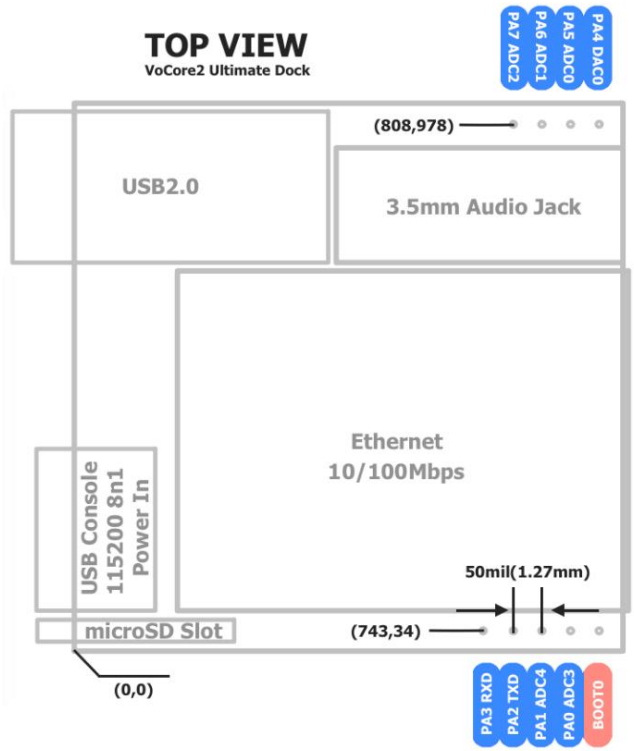
Obrázek 8 [14]

5.2.1 VoCore2 Ultimate

Tato verze je odlišná od předchůdce tím, že je rozšířena o Dock Station. Výhodou je opět, jako u VoCore Dock, možnost zapojení USB, Ethernetu, SD Card, audio výstupu. Navíc Dock Station obsahuje 4x A/D převodníky a 1x D/A převodník [15]. Technické údaje jsou znázorněny na následujícím obrázku.

SIZE	28mm x 30mm x 30mm
CPU	MT7628AN, 580 MHz, MIPS 24K
MEMORY	128MB, DDR2, 166MHz
STORAGE	16M NOR on board, support SDXC up to 2TB
WIRELESS	802.11n, 2T2R, speed up to 300Mbps.
ANTENNA	One U.FL slot, one on board antenna.
ETHERNET	1 port/5 ports, up to 100Mbps.
USB	Support USB 2.0, up to 480MBit/s.
PCIe 1.1	Supported
GPIO	>=40 (pinmux)
UART	x3 (UART2 for debug console)
PWM	x4
A/D CONVERT	4 channels
D/A CONVERT	1 channels
AUDIO PLAYBACK	Support
AUDIO RECORD	Support
DEBUG CONSOLE	On board driver-free USB2TTL
SHELL	Included
POWER SUPPLY	3.6V ~ 6.0V
POWER CONSUMPTION	74mA wifi standby, 230mA wifi full speed, 5V input.

Obrázek 9 – specifikace VoCore2 Ultimate [15]



Obrázek 10 - schéma Dock Station pro VoCore2 Ultimate [14]

5.2.2 Další modifikace VoCore2

Jsou i další druhy VoCore2, které se trochu liší v hardwarovém uspořádání a oblastech použití:

- VoCore2 Lite – nejlevnější zařízení s trochu redukovaným hardwarem a trochu menším výkonem oproti VoCore2. Je určeno zejména pro studenty a začátečníky ke zkoumání embedded zařízení a na pokusy s nimi [13].
- VoCore2 + PoE Dock – řeší snahu o vytvoření routeru co nejmenších rozměru. V podstatě je to redukovaný VoCore2 Ultimate, který obsahuje jen PoE port [13].
- VoCore2 + Audio Dock – je navržen jako analogie MP3 přehrávače, speciálně pro poslech hudby, která se přenáší pomocí Wi-Fi do zařízení s DLNA nebo Airplay protokolem. [13]

Následující obrázek uvádí porovnání VoCore1, VoCore2 a VoCore2 Lite.

VoCore family

	VOCORE	VOCORE2 LITE	VOCORE2
Price	19.99USD	**3.99USD**	11.99USD
CPU	RT5350, 360MHz	MT7688AN, 580MHz	MT7628AN, 580MHz
Memory	32MB SDRAM	64MB DDR2	**128MB DDR2**
Storage	16MB NOR	8MB NOR	**16MB NOR**
Antenna Slot	x1	x1	**x2**
On-Board ANTENNA	√	×	√
Wireless Speed	~75Mbps	~150Mbps	**~300Mbps**
Ethernet Port	x5	x1 / x5*	x1 / x5*
Ethernet Speed	100Mbps	100Mbps	100Mbps
SPI Master	√	√	√
SPI Slave	√	√	√
SPI DMA	×	√	√
USB 2.0 Host	√	√	√
USB 2.0 OTG	√	×	×
PCIe 1.1	×	×	√
SD Support	SPI	√	√
GPIO	> 30**	>=40**	>= 40**
UART	x2	x3	x3
PWM	–	x2	x2
Power Consumption	138mA	**74mA**	**74mA**

Obrázek 11 [14]

6. Součástky

V této kapitole probereme veškeré použité součástky, které byly zvoleny pro návrh systému. Taky prodiskutujeme jejich výhody.

6.1 Napájecí zdroj

Galvanoplastika je často slaboproudý proces, který požaduje stejnosměrný průběh napětí a proudu, velice zřídka – střídavý. Hodnoty napětí málokdy přesahují mez 15 V, hodnoty proudu 5 A. Z těchto důvodu byl zvolen napájecí zdroj HCS-3400 USB od výrobce Manson, umožňující dodat napětí do 16 V a proud do 40 A. Hlavní výhodou je možnost vzdáleného ovládání zdroje přes převodník USB-serial [16]. Taky je možné vzdáleně nastavovat hodnoty proudu a napětí, odebírat jak současné, tak i přednastavené hodnoty. Jsou i další možnosti ovládání zdroje příslušnými příkazy, popsanými v manuálu [16]. Tento zdroj může být připojen k Raspberry Pi 3 anebo dalšímu embedded zařízení [17]. V rámci této práce se pokusím odebírat okamžité hodnoty proudu a napětí, zobrazené na displeje přímo ze zdroje a zároveň je zobrazovat.

6.2 Teplotní senzor

Hlavními požadavky na teplotní senzor je měřitelný rozsah teplot a rozlišení, se kterým čidlo dokáže změřit teplotu. S ohledem na tyto požadavky byl zvolen číslicový teploměr Dallas DS18B20 od výrobce MAXIM, který je funkčně založen na teplotní závislosti PN přechodu. Vyhodnocuje se změna napětí na přechodu v závislosti na teplotě. DS18B20 poskytuje poměrně dobré rozlišení 0,5°C s možnosti desetinásobného zlepšení přesnosti měření kompenzováním chyby měření [18]. Teplotní senzor umožňuje měřit teploty v rozsahu -55°C až +125°C, v rozsahu -10°C až 85°C s rozlišením 0,5°C [19].

Teplotní senzor má tři piny – GND, DQ (Data In/Out) a V_{DD} (napájení v rozsahu 3,0 V až 5,5 V). Veškerá komunikace probíhá přes datovou linku – DQ. Existuje možnost napájení senzoru pouze jedním drátem (společně s GND) – využívá se vestavený kondenzátor, který se nabije a působí jako zdroj napětí při velkých úrovních signálů procházejících datovou linkou. Každý senzor má unikátní 64-bitový identifikační kód, díky kterému lze provozovat více senzoru na stejném vedení. [19]

Komunikace senzorů s mikrokontrolerem probíhá pomocí 1-wire protokolu vyvinutého výrobcem. Používá se koncepce Master-Slave, kde Slave je vždy senzor. První bit zprávy je vždycky signifikantní. 1-wire sběrnice požaduje externí odpor velikosti přibližně 5 kΩ. Pro přístup k senzoru se využívá transakční sekvence v následujícím pořadí:

- Inicializace – Master vyšle resetovací puls a čeká až mu zařízení Slave odpoví „prezenčním“ pulsem;
- ROM příkazy – tyto příkazy zjišťují identifikační údaje (sériové číslo a typ zařízení Slave);
- DS18B20 funkční příkaz – umožňují Masterovi číst a zapisovat data do paměti DS18B20 a také zajišťují konverzi teploty.

Je důležité dodržovat tento postup při každém přístupu k senzoru. Senzor nebude reagovat na příkazy, pokud vynecháme i jeden bod této sekvence. Ovšem výjimkou jsou dva ROM příkazy – Search ROM [F0h] a Alarm Search [ECh]. Po těchto příkazech se zařízení master (většinou je mikrokontroler nebo počítač) musí vrátit k prvnímu kroku. [19]

1-wire protokol využívá různé typy signálů – resetovací puls, prezenční puls, „Načti 1“ (READ 1), „Načti 0“ (READ 0), „Zapiš 1“ (WRITE 1), „Zapiš 0“ (WRITE 0). Veškeré signály vysílá Master s výjimkou prezenčního pulsu, který pošle Slave jako odpověď na resetovací puls. Veškeré hodnoty se zapisuji nebo načítají během tzv. zapisovacích nebo načítacích rámců. Jeden rámec umožňuje načíst nebo zapsat jeden bit a to je 1 nebo 0. Zapisovací rámec má dobu trvání 1 µs až 60 µs. Master inicializuje zápis snížením napěťové úrovně na 1-wire sběrnici na minimum. Tuto napěťovou úroveň na stráně Slave navýší použitý odpor na maximum. Pro vytvoření rámce „zapiš 1“, musí Master uvolnit sběrnici (snížit napěťovou úroveň na minimum) na dobu 15 µs, pro vytvoření rámce „zapiš 0“ – Master navyšuje napětí po dobu alespoň 60 µs (na stráně Slave odpor sníží napěťovou úroveň na minimum). Princip je jednoduchý – za výše uvedenou dobu DS18B20 navzorkuje signál a zapíše logickou „1“, pokud je vzorkovaná úroveň napětí vysoká, nebo 0 v opačném případě. Rámce „Načti 1“ nebo „Načti 0“ májí minimální délku trvání 60 µs. Až Master inicializuje načítací rámec, Slave začne vysílat data. Navýšení napětí na sběrnicí je interpretováno logickou 1. Pro vysílání logické „0“, DS18B20 uvolní sběrnicí po celou dobu načítacího rámce. Minimální časový odstup mezi načítacími, i zapisovacími rámci je 1 µs. [19]

6.3 Přenosové prostředky

Ke komunikaci napájecího zdroje a embedded zařízení použijeme převodník USB-seriál – Silicon Hills cp210x. Ovládání zajistí program napsaný v jazyce Lua, který bude řídit komunikaci mezi zařízením a napájecím zdrojem.

Pro napájení a odebírání informace z teplotního senzoru použijeme měděné vodiče z UTP kabelu, kterými propojíme piny senzoru spolu s příslušnými GPIO porty mikrokontroleru. DS18B20 bude komunikovat prostřednictvím 1-wire protokolu, který zastupují příslušné balíčky v kernelu operačního systému UNIX - OpenWRT/LEDE (viz. kapitola 7.3.1). Samotné připojení se sestaví ze třech vodičů – zem, napájení a datová linka – připojených na příslušné GPIO porty embedded zařízení. Nesmíme zapomenout doplnit odpor o velikosti 4,7 kΩ mezi datovou linku a napájení.

Přístup k mikrokontroleru se bude provádět pomocí vestaveného Wi-Fi přístupového bodu na přednastavenou IP-adresu – 192.168.1.1. Ovládání zařízení a monitorování procesu galvanického vylučování kovů umožní vhodné upravení stávající infrastruktury LuCI – webového rozhrání na ovládání a nastavování embedded zařízení, běžících na systému OpenWRT/LEDE.

6.4 Embedded zařízení

Ovládání všech výše zmíněných komponentů se bude provádět prostřednictvím embedded zařízení - VoCore + Dock. Výkon, technické prostředky a paměťová kapacita VoCore bohatě stačí pro splnění této úlohy. Rozdíl mezi VoCore1 a VoCore2 je v podstatě zanedbatelný, a to z pohledu prostředků nutných ke splnění stanovené úlohy. Možnosti ukládání nasbíraných dat je řešeno v následující kapitole.

6.5 Zabezpečení

Pokusíme se prodiskutovat veškeré možné cesty a prostředky, které poskytuje VoCore na úlohu zabezpečení komunikace.

V menu "Network" položka „Wireless" webového rozhrání LuCI, po rozkliknuti bezdrátového rozhrání, je sekce „Wireless Security", která umožňuje zašifrovat komunikaci přes Wi-Fi následujícími způsoby:

- WEP Open Systém – základem je šifrovací algoritmus RC4. Klient nemusí poskytovat ověřovací údaje. Datové rámce se zakódují pomocí WEP-klíče, pokud klient má správný klíč;
- WEP Shared Key – Klient požaduje autorizaci u přístupového bodu. Přístupový bod pošle klientovi výzvu, kterou klient zašifruje vlastním WEP-klíčem a pošle zprávu zpět. Přístupový bod se pokusí dešifrovat tuto zprávu pomocí vlastního klíče. Pokud se mu podaří dešifrovat zprávu a klíče se shodují, pošle pozitivní odpověď o navázaní spoje;
- WPA-PSK (Pre-Shared Key) – je náhradou WEP a používá silnější šifrovací algoritmus. Požaduje jenom jedno heslo pro každý bod sítě (v našem případě embedded zařízení). Pokud zadané heslo je totožné s nastaveným, umožní se spojení;
- WPA2-PSK – je náhradou WPA, do kterého byly implementované všechny povinné prvky, dle doporučení IEEE 802.11i, a to především algoritmus CCMP založený na AES;
- Kombinace WPA-PSK a WPA2-PSK – kombinace dvou výše uvedených postupů. [20, 21]

Dále lze použít i filtr na MAC-adresy, který umožňuje povolit přístup a přihlášení jenom těm zařízením, kterých MAC-adresy byly zaneseny do databáze. Tento způsob zajišťuje dobré zabezpečení vzhledem k tomu, že jenom ta osoba, která fyzicky ovládá zařízení existující v této databázi, má přístup k datům. V tomto případě k útoku na systém dojde jenom v případě, že se útočník zjistí MAC-adresu zařízení, které poskytuje přístup, nasimuluje jej a použije pro vniknutí. Z hlediska software je to relativně dobré zabezpečení. V tomto případě je veškerá zodpovědnost na majitelovi přístupového zařízení.

Veškera komunikace bude probíhat v rámci lokální sítě, která nebude mít přístup na Internet, čímž zabráníme možné útoky z vnějšku.

6.6 Cloud vs. Interní paměť zařízení

Nyní probereme možností ukládání dat s využitím serverových aplikací, příkladem kterých může být Cloud a možnost ukládání dat do interní paměti zařízení.

Cloud je perspektivní nástroj pro zpracování velkého množství dat, který zpravidla poskytuje možnost sběru dat z geograficky vzdálených objektů a přístup k nim odkudkoli ze světa, velký výpočetní výkon a jako jeho důsledek „okamžitou“ analýzu dat. Dle poskytovaných služeb, se služby Cloud dělí na:

- IaaS – Infrastructure as a Service – uživatel používá celou infrastrukturu a má maximální kontrolu nad řízením, procesem a výpočtem probíhajícím v Cloudu;
- PaaS – Platform as a Service – uživatel používá nabídnuté řešení a nemá možnost optimalizovat čas výpočtu;
- SaaS – Software as a Service – hotové řešení ve tvaru jednoduché aplikace.

Dle naší úlohy by nám nejvíce vyhovovala první nebo druhá možnost – pronajmout si výpočetní kapacitu a dále v ní vytořit analýzu nasbíraných dat. V tomto případě embedded zařízení bude provádět jenom sběr dat a nic více, veškeré zpracování proběhne v Cloudu. Toto řešení má několik nevýhod:

- požaduje připojení na Internet a budeme muset vyřešit bezpečnost v sítí;
- pokud realizujeme cyklické ukládání dat, nevyužije se veškerá pronajatá kapacita;
- pokud po nějaké době dojde k zaplnění kapacity, budeme muset pronajmout další kapacitu nebo vymazat data, což způsobí další náklady;
- zvýší se náklady na systém;
- nevyužije se celý výkon embedded zařízení.

Oproti Cloudu, embedded zařízení obsahuje nástroje na sběr a analýzu dat a umožňuje ji realizovat v interní paměti bez jakéhokoliv přenosu dat. Operační systém LEDE (v minulosti OpenWRT) již umí zobrazovat průběhy. Stačí se podívat do menu „Status“ položka „Realtime Graphs“. Bohužel tato sekce neumožňuje ukládání dat na dobu delší, než 15 minut. Existuje i jiný nástroj na zobrazování dat v embedded zařízení – rrdtool. Práci s tímto nástrojem je věnována kapitola 7.3.2. Rrdtool převážně pracuje se soubory formátu .rrd a umožňuje cyklické ukládání dat na předem definovanou dobu. Soubor, který obsahuje hodnoty sbírané každou minutu během týdne má velikost 80 kB. Zvolené zařízení VoCore má k dispozici 16 MB interní paměti, což bohatě stačí pro uložení třech veličin – teplota, proud a napěti. Embedded zařízení již má veškerou nutnou infrastrukturu na sběr, ukládání a znázornění měřených dat.

Z důvodů malého množství sbíraných dat a existenci infrastruktury v embedded zařízení pro splnění úlohy možnost použití serverových aplikací, jako jsou Cloudové služby, zamítáme.

7. Experiment

7.1 Podmínky

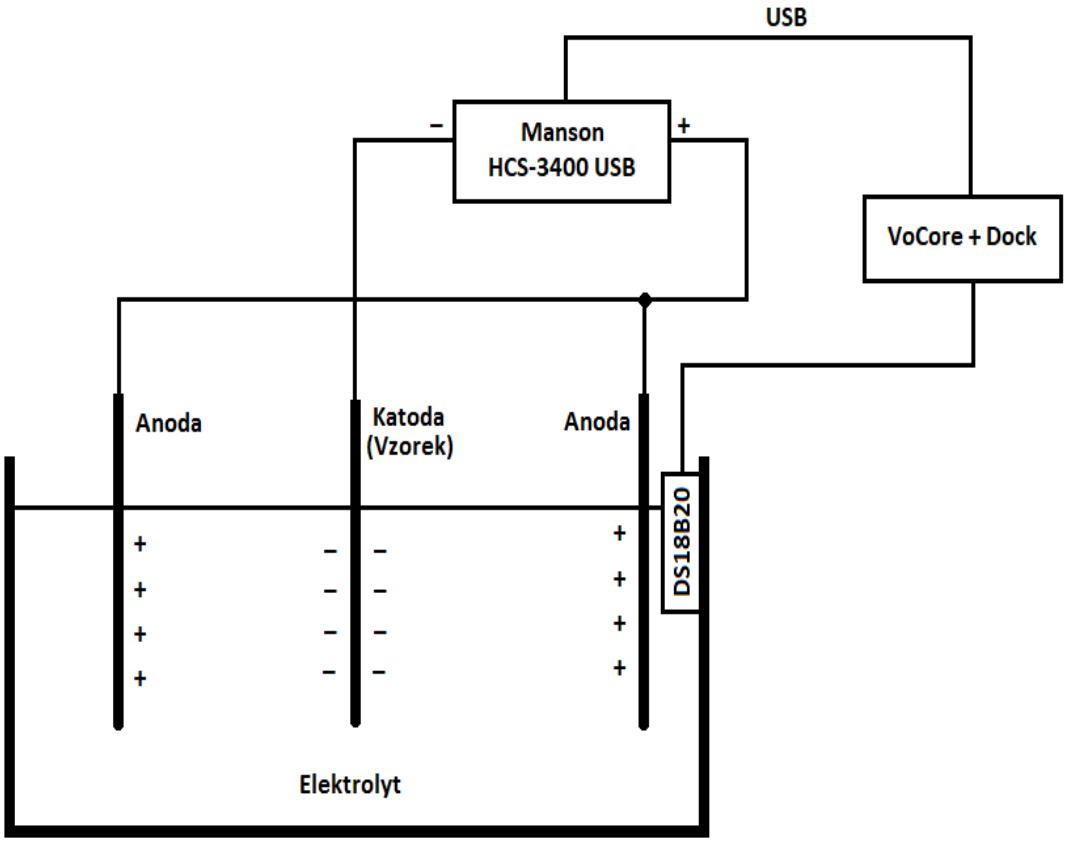
Omezíme se na kontrolu odebírání dat ze senzoru a napájecího zdroje a správné zobrazování jejich průběhu. Umožníme uživateli drobné změny parametrů a snadný přístup k datům. Vzhledem k časové náročnosti práce se nebudeme zabývat predikcí, jako je výpočet tloušťky vrstvy s časem a podobné. Jelikož nemáme k dispozici stroj na galvanické vylučování kovů, experiment se omezí na drobné zkoušky navržené funkcionality systému. Kontrola funkčnosti a vlastní experiment budou výrazně jednoduché a budou se sestavovat z několika bodů. Pokusíme se zkontrolovat následující nároky na navržený systém:

- změna údaje teploměru a správné zobrazení teploty okolí – ověříme dotykem ruky, ponořením do teplé vody, atd;
- nasimulujeme možný průběh proudu a napětí na napájecím zdroji;
- otestujeme real-time přenos dat – kontrola okamžité změny údajů.

Splnění těchto požadavků simuluje veškeré nutné funkce ke konání procesu galvanického vylučování kovů.

7.2 Schéma zapojení

Uvedeme výsledné schéma zapojení navrhovaného systému k jeho budoucí integraci do procesu elektroformování.



Obrázek 12 – výsledné schéma zapojení

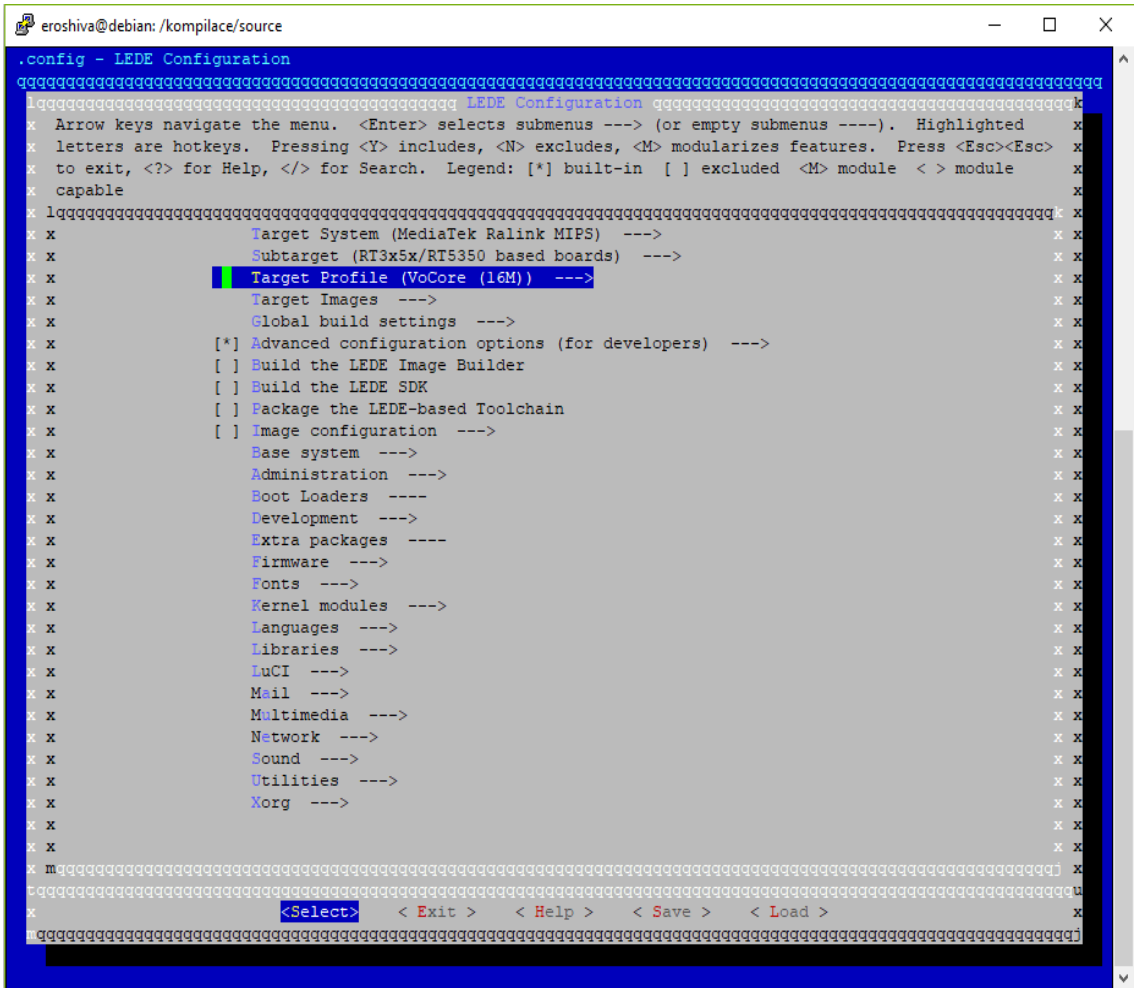
7.3 Konfigurace

Je důležité správně propojit a nakonfigurovat VoCore + Dock a příslušná zařízení, abychom dosáhli požadovaných výsledků. Dále uvedeme postup konfigurace s podrobným popsáním každého kroku, použitými příkazy a dalšími využitými pomůckami.

7.3.1 Kompilace systému LEDE

Před tím, než nachystáme komunikaci s napájecím zdrojem a teploměrem, musíme zkompilovat vlastní systém, který bude obsahovat veškeré potřebné nástroje ke splnění této úlohy. Budeme kompilovat systém LEDE pro zařízení VoCore.

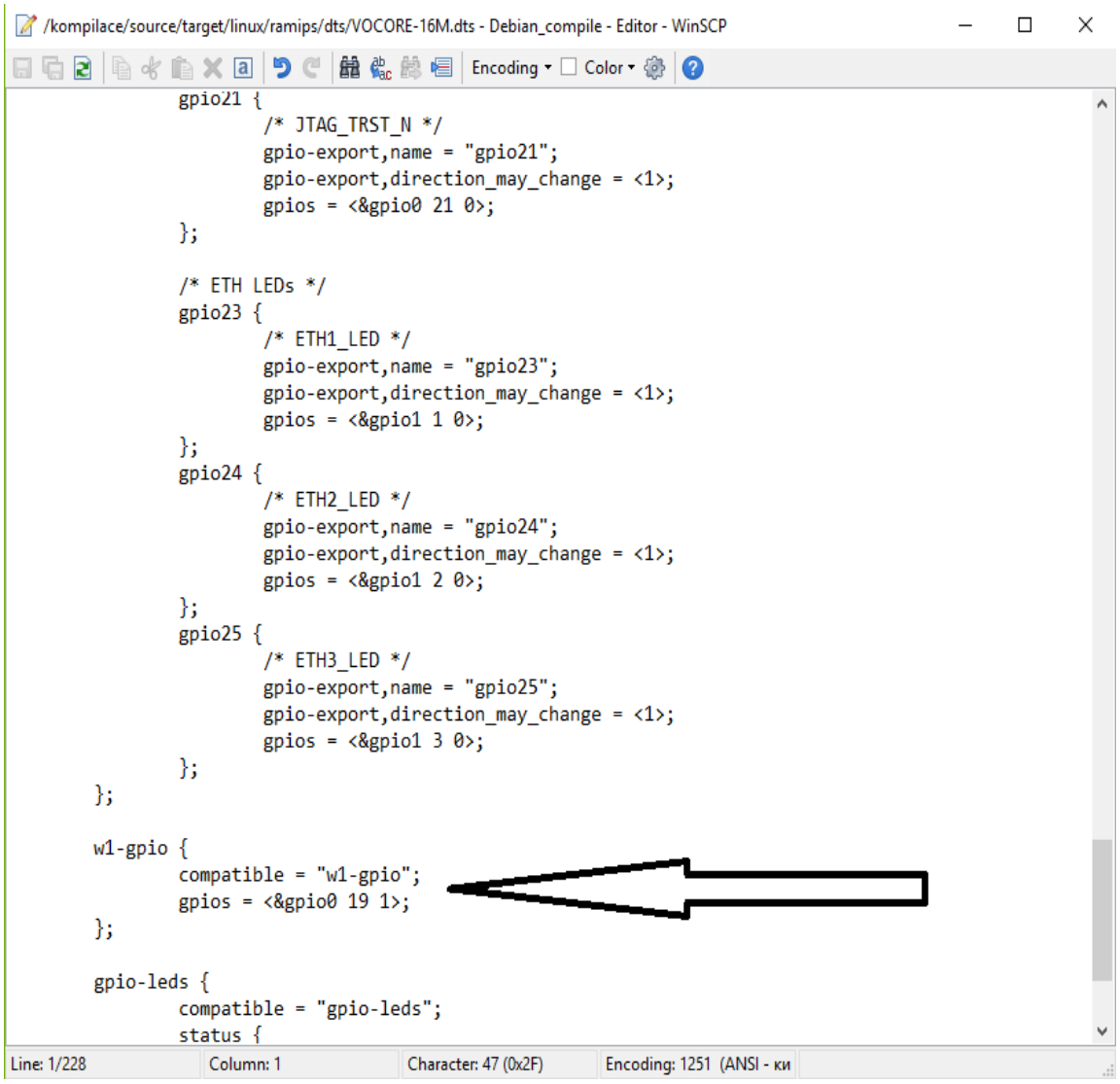
Vytvoříme v Oracle VM VirtualBox virtuální stroj, na kterém běží OS Linux Debian. Nastavíme tento stroj v souladu s pokyny, popsanými na oficiálních webových stránkách projektu LEDE [22]. Vytvoříme nový adresář, do kterého stáhneme z gitu soubory nutné ke kompilaci systému pomocí příkazu *„git clone https://git.lede-project.org/source.git“*. Přemístíme se do nově vzniklého adresáře /source. Provedeme obnovení balíčku příkazem *„./scripts/feeds update -a“* a následně je nainstalujeme příkazem *„./scripts/feeds install -a“*. Nyní máme všechno připravené ke kompilaci. Provedeme *„make menuconfig“* a zvolíme první tři položky dle obrázku č. 7.



Obrázek 13 – kompilace OS LEDE

- Target System – volba platformy;
- Subtarget – volba typu procesoru;
- Target Profile – volba zařízení, pro které budeme kompilovat systém.

Dále musíme upravit DTS soubor TAK, aby systém se správně zkompiloval pro VoCore + Dock. Defaultně s tímto rozšířením (Dock) LEDE nepočítá. Vyjdeme z tohoto menu, přesuneme se do adresáře *…/source/target/linux/ramips/dts/* a vyhledáme soubor, který se jmenuje VOCORE-16M.dts. Tento soubor změníme v souladu s doporučením od výrobce [23]. Zároveň odstraníme v tomto souboru definici GPIO portu č. 19, na který potom připojíme teplotní senzor (JTAG rozhrání v této úloze nepotřebujeme). Ještě můžeme pod definici LED přidat kompatibilitu GPIO 19 s w1 (viz následující obrázek).



Obrázek 14

Po těchto krocích provedeme v adresáři *…/source/ „make defconfig“*, aby se nastavily veškeré defaultní závislosti pro zařízení VoCore. Poté voláme *„make menuconfig“* a v položce *„Kernel modules“* zvolíme následující balíčky:

- kmod-usb-serial;
- kmod-usb-serial-cp210x;
- kmod-w1;
- kmod-w1-master-gpio;
- kmod-w1-gpio-custom;
- kmod-w1-slave-therm;

Dále v položce *„Languages“* nastavíme podporu programovacího jazyku Lua a zvolíme další dva balíčky – *luac* (kompilátor lua) a *lua-rs232* (knihovna na komunikaci se sériovým rozhraním). V položce *„LuCI“* nastavíme základní podporu webového rozhraní LuCI. Naposled v položce *„Utilities -> databases“* zvolíme balík *rrdtool1*. Nyní máme zvolené veškeré potřebné balíky. Vystoupíme z menu s uložením změn. Kompilaci zahájíme příkazem *„make V=s“*. Až tento proces skončí, v adresáři *…/bin/targets/ramips/rt305x/* se objeví soubory s již zkompilovaným systémem.

7.3.2 Práce s rrdtool

Zaměříme se na způsob sbírání a zobrazování dat tak, jak získaná data budeme ukládat a zobrazovat. Pro tyto účely se nejlépe hodí nástroj rrdtool, který poskytuje velké možnosti pro ukládání a znázornění dat. Rrdtool pracuje s různými formáty souborů. Pro naše účely se nejlépe hodí formát .rrd, což je zkrátka, která znamená Round Robin Databases. Round Robin je technika, která využívá fixované množství dat a ukazatel na každý prvek. Toto je hlavní důvod, proč využívat tento formát. Omezené množství uložených dat má v důsledku omezení velikosti souboru, tzn. soubor nepřesáhne určitou velikost a nezabere zbytečně moc místa v paměti zařízení. Po dosáhnutí horní meze uložených dat soubor začne ukládat nová data od začátku, tudíž ukládaní dat je cyklické a „nekonečné“. Nástroj rrdtool umí ukládat, načítat, zobrazovat a pracovat s daty souborů formátu .rrd. Jedním z důvodů použití rrdtool je možnost převést naměřené hodnoty do grafické podoby. [24]

Probereme tedy základní příkazy na ovládání rrdtool. Zkusíme vytvořit testovací soubor. Toto se provede pomocí příkazu [25]:

„rrdtool create jméno_souboru.rrd -- požadované parametry“.

Kupříkladu vytvoříme soubor test.rrd a hned uvedeme příklad v programovacím jazyku Lua.

„require(„luci.sys“).call(‘rrdtool create teplota.rrd --step 60 DS:test:GAUGE:120:U:U RRA:AVERAGE:0.5:1:10080‘)“

Knihovna luci.sys umožňuje z programu zadat příkaz do konzole, který se následně vykoná. Dále probereme předané rrdtool parametry pro vytvoření souboru. Výše uvedený příkaz vytvoří soubor test.rrd, který bude akceptovat 1 hodnotu každých 60 vteřin (--step 60). Dále definujeme datový zdroj - data source (DS), který pojmenujeme *test* a který bude typu GAUGE (určen pro ukládání dat jako je teplota, počet lidí v místnosti atd. [23]). Jeden soubor formátu .rrd může obsahovat více než jeden datový zdroj. Pokud během 120 vteřin do souboru nebude zapsána žádná další hodnota, soubor se automaticky uloží hodnotu *UNKNOWN*. U:U představuje akceptovatelné meze hodnot. V našem případě jsou U – unlimited. Dále musíme definovat alespoň jeden Round Robin Archiv – RRA, kterých může být maximálně 4 (podle typů, a to jsou AVERAGE, MAX, MIN, LAST). Například RRA bude využíván pro ukládání 1 hodnoty během 7 dnů (10080*60 vteřin = 168 hodin = 7 dnů). Přičemž parametr AVERAGE znamená, že pokud se na vstupu do souboru, během zadaného intervalu zápisu, objeví více než 1 hodnota, rrdtool je zprůměruje a uloží tuto průměrnou hodnotu. Dále lze použit i další parametry, kromě AVERAGE [25]:

- MAX – uloží největší získanou hodnotu během intervalu;
- MIN – uloží nejmenší získanou hodnotu;
- LAST – uloží poslední získanou hodnotu.

Máme připravený soubor pro zapisování dat. Podíváme se jakým způsobem můžeme do něj data ukládat [24].

„rrdtool update test.rrd čas:hodnota“

Tento příkaz uloží do souboru test.rrd požadovanou hodnotu v určitý čas. Pro real-time ukládání se využije následující tvar tohoto příkazu [26]:

„rrdtool update test.rrd N:hodnota“,

kde parametr N (now) zajišťuje načtení aktuální hodnoty času ve vteřinách z operačního systému. Do souboru lze zapsat dvě a více hodnot. De-facto je omezení určeno pouze operačním systémem.

Výpis obsahu soboru se provede příkazem [26]:

„rrdtool fetch test.tmp AVERAGE“

Bohužel výsledek tohoto příkazu není moc přehledný. Potřebujeme tak tato výstupní data znázornit jiným způsobem. Rrdtool umožňuje převést data ze souboru .rrd do grafické podoby několika způsoby:

- Sjednotit všechny průběhy do jednoho obrázku (nebude uživatelsky přehledné);
- Vykreslit každý průběh zvlášť;
- Vytvořit kombinaci požadovaných průběhů.

Nebudeme však nyní zbytečně komplikovat úkol a vykreslíme každý průběh zvlášť. Pro tyto požadavky je výhodný graf, který vytvoříme následujícím příkazem:

„rrdtool graph test.rrd --parametry DEF:mytest=test.rrd:test:AVERAGE
LINE3:mytest#FF0000”

Zde do parametrů patří možnosti nastavit časový interval vykreslení (--start, --end), šířka a výška obrázku (--height, --width), legenda (--title) a další [24]. Po parametrech musíme definovat proměnnou pro vykreslení dat (lze provádět i matematické operace). Toto už je však nad rámec této úlohy - zde žádné transformace dat nepotřebujeme. Definice výstupního průběhu se provádí pomocí proměnné mytest, která ukazuje na data uložená v archivu AVERAGE souboru test.rrd, Data Source test. Na konci je definice průběhu: tloušťka je 3 pixely (LINE3), vykreslovaný průběh je uložen v proměnné mytest, barva průběhu je červená (dle RGB v hexadecimálním tvaru). [24]

Nástroj rrdtool umí více funkcí, než bylo uvedeno v této kapitole. Většinu z nich pro splněni zadaní nepotřebujeme, a proto je nebudeme uvádět.

V následujících dvou kapitolách stvoříme program, který bude v nekonečné smyčce sbírat data z napájecího zdroje a senzoru teploty, ukládat je a následně vykreslovat průběhy. Nebudeme uvádět celý kód, popíšeme jenom důležité kroky. Program se bude jmenovat *monitorovani.lua* a bude umístěn v adresáři */www/luci-static/resources/monitorovani/* spolu se soubory, do kterých budou uložená data, a obrázky, které znázorňují tyto průběhy.

7.3.3 Napájecí zdroj

Napájecí zdroj nebyl určen ke komunikaci pomocí terminálu, ale poskytuje možnost vzdáleného ovládaní. Výrobce poskytuje vlastní program, který je ke stažení z webových stránek společnosti Manson. Tento program není kompatibilní s OpenWRT a proto musíme vytvořit vlastní prográm, který zajistí komunikaci, výběr a ukládání aktuálních hodnot z napájecího zdroje. Existuje možnost ovládání zdroje pomocí příkazové řádky v Linuxu. Využijí se příkazy podobné tomuto: *„echo „GETD\r“ > /dev/ttyS0“*, kde „\r“ na konci znamená CR (carriage return – návrat na začátek řádku), příkaz GETD vrátí aktuální hodnoty, které se zobrazují na displeji. Toto není úplně vhodná forma komunikace pro tuto úlohu.

Komunikace napájecího zdroje a VoCore bude probíhat pomocí sériového rozhrání vytvořeného na bázi USB. Program bude napsán v programovacím jazyce Lua, který se široce využívá v embedded zařízeních. Pro naše účely stačí standardní knihovny, které jsou k dispozici v minimální konfiguraci Lua pro OpenWRT/LEDE. Přidali jsme pouze dvě knihovny navíc – luac a lua-rs232 (viz. Kapitola 7.3.1). Postupme k návrhu vlastního programu, který bude komunikovat se zdrojem, ovládat jej, popřípadě řídit. Nakonfigurujeme příslušný port ve VoCore, dle pokynů popsaných v manuálu napájecího zdroje [16]:

```
-- Knihovna pro komunikaci sériovým rozhráním
rs232 = require("luars232")

-- Inicializace COM-portu
port_name = "/dev/ttyUSB0" -- pro zjištění lze analyzovat syslog
local out = io.stderr
local volt = 0 curr = 0
```

```
-- Otevřeme port
local e, p = rs232.open(port_name)
if e ~= rs232.RS232_ERR_NOERROR then
        -- handle error
        out:write(string.format("can't open serial port '%s', error: '%s'\n",
                                port_name, rs232.error_tostring(e)))
        return
end

-- Nastavíme příslušné parametry dle manuálu
assert(p:set_baud_rate(rs232.RS232_BAUD_9600) == rs232.RS232_ERR_NOERROR)
assert(p:set_data_bits(rs232.RS232_DATA_8) == rs232.RS232_ERR_NOERROR)
assert(p:set_parity(rs232.RS232_PARITY_NONE) == rs232.RS232_ERR_NOERROR)
assert(p:set_stop_bits(rs232.RS232_STOP_1) == rs232.RS232_ERR_NOERROR)
assert(p:set_flow_control(rs232.RS232_FLOW_OFF)  == rs232.RS232_ERR_NOERROR)
```
[28]

Data ze zdroje vyčteme pomocí příkazů popsaných v manuálu [16]. Pro sledování provozu procesu nás zajímají především skutečná data, proto použijeme příkaz GETD, který vrací odpověď ve tvaru NnnNPppP[CR]OK[CR] ([CR] – znamená carriage return, návrat na začátek řádku).

```
-- Pošleme příkaz na zjištění dat
err, len_written = p:write("GETD\r")
assert(e == rs232.RS232_ERR_NOERROR)

-- Proměnná, do které budeme načítat data
local vystup = ""

while string.len(vystup) < 8 do -- Načteme prvních 8 znaku, což je NnnNPppP

local read_len = 8 -- načteme 8 bitů - to je 1 písmeno (číslo ve formě řetězce)
local timeout = 100 -- doba čekání na odpověď v milisekundách
local err, data_read, size = p:read(read_len, timeout)
assert(e == rs232.RS232_ERR_NOERROR)
vystup = vystup .. data_read
end

-- Musíme dočíst údaje
while string.len(vystup) < 13 do -- K celkové délce výstupu musíme přečíst 2x '\r', abychom se zbavili kolizí při příštích načítáních

local read_len = 8 -- načteme 8 bitů - to je 1 písmeno
local timeout = 100 -- doba čekání na odpověď v milisekundách
local err, data_read, size = p:read(read_len, timeout)
assert(e == rs232.RS232_ERR_NOERROR)
vystup = vystup .. data_read
end
```

Dále musíme upravit načtená data do tvaru, který by byl vhodný na jejich zapisování do souborů.

```
-- Převedeme načtená data z řetězce do čísel
-- Napětí jsou první 4 symboly
volt = tonumber(string.sub(vystup, 1, 4))
volt = volt/100 -- protože je čtyřmístní číslo

-- Proud jsou symboly 5 až 8
curr = tonumber(string.sub(vystup, 5, 8))
curr = curr/100 -- protože je čtyřmístní číslo
```

Tedy uložíme získané hodnoty do souborů napeti.rrd a proud.rrd, které jsou umístěné v adresáři */www/luci-static/resources/monitorovani/*. Provedeme následné znázornění formou vykreslení obrázků napeti.png a prod.png

```
-- Přidání nových údajů do databází
require ("luci.sys").call('rrdtool update
/www/luci-static/resources/monitorovani/napeti.rrd N:' .. volt)

require ("luci.sys").call('rrdtool update
/www/luci-static/resources/monitorovani/proud.rrd N:' .. curr)

--Vykreslení obrázků
require ("luci.sys").call('rrdtool graph
/www/luci-static/resources/monitorovani/napeti.png --title="Prubeh napeti"' .. lazy ..
tstart .. tend.. height .. width .. fsize .. ' DEF:v=napeti.rrd:volt:AVERAGE LINE3:v' .. colour)

require ("luci.sys").call('rrdtool graph
/www/luci-static/resources/monitorovani/proud.png --title="Prubeh proudu"' .. lazy ..
tstart .. tend .. height .. width .. fsize .. ' DEF:cur=proud.rrd:curr:AVERAGE LINE3:cur' .. colour)
```

Knihovna *„luci.sys“* imituje zadání systémových příkazů přes konzoli a proto se hodí pro snadnou implementací příkazů [29]. Proměny lazy, tstart, tend, height, width, fsize a colour jsou parametry načtené z konfiguračního souboru „nastaveni“. Veškeré obrázky budou uložený v adresáře */www/luci-static/resources/monitorovani/*. To se provede pomocí následujícího kódu, kde jsme předem definovali defaultní stavy parametrů, pokud by došlo k jejich neexistenci v konfiguračním souboru (UCI vrátí hodnotu *„nil“*):

```
-- Knihovna na použití UCI pro načítaní parametrů
require("uci")
local uci = uci.cursor()

-- Definice proměnných, do kterých se načtou parametry.
local lazy = "" tstart = "" tend = "" height = "" width = "" colour = "#FF0000" fsize = ""
```

```
-- Načítaní parametrů pro vykreslení grafů
local h = uci:get("nastaveni", "nastaveni", "height")
local w = uci:get("nastaveni", "nastaveni", "width")
local c = uci:get("nastaveni", "nastaveni", "colour")
local l = uci:get("nastaveni", "nastaveni", "lazy")
local st = uci:get("nastaveni", "nastaveni", "tstart")
local e = uci:get("nastaveni", "nastaveni", "tend")
local fs = uci:get("nastaveni", "nastaveni", "fullsize")
```

Tyto parametry musíme poupravit na vhodný výstupní tvar, který se poté předá do příkazu rrdtool graph. Převedeme je do tvaru řetězce, přidáme na začátek odpovídající hlavičky a ošetříme možnou neexistenci hodnot v konfiguračním souboru.

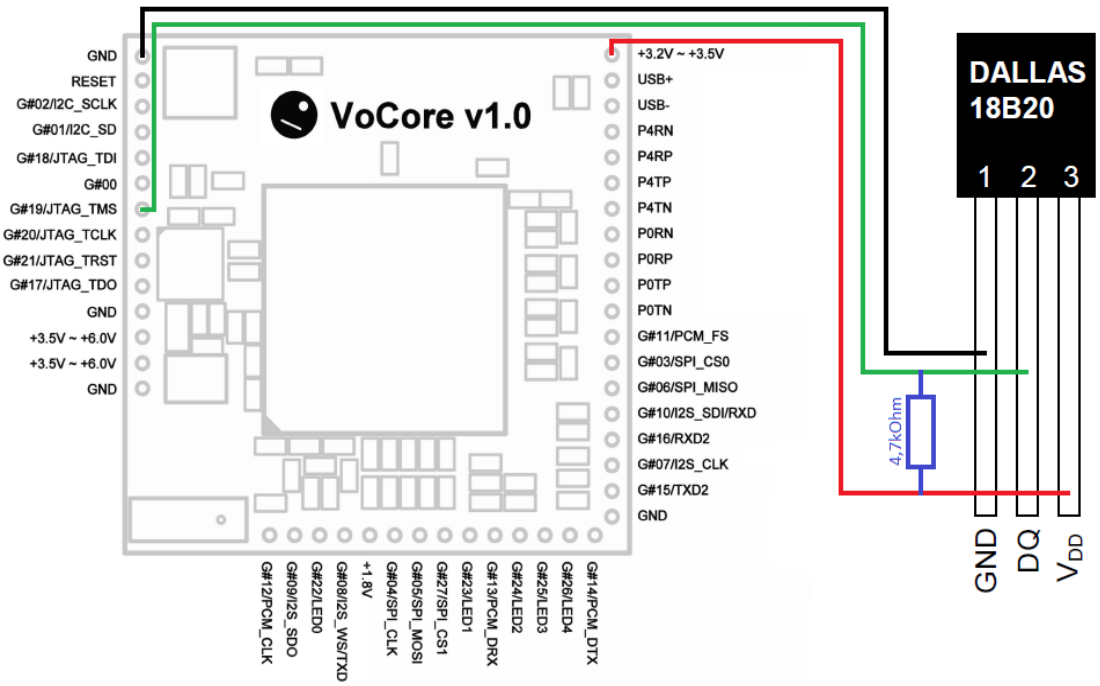
```
-- Převedení parametrů na příkazový tvar
if l == "1" then lazy = " --lazy" end
if h ~= nil then height = " --height " .. h end
if w ~= nil then width = " --width " .. w end
if c ~= nil then colour = c end
if st ~= nil then tstart = " --start " .. st end
if e ~= nil then tend = " --end " .. e end
if fs == "1" then fsize = " --full-size-mode" end
```

Výše uvedené kroky sjednotíme do nekonečné smyčky, aby se po splnění těchto kroků cyklus opakoval.

Implementaci vykreslených dat do webového rozhraní LuCI probereme v kapitole 7.3.5.

7.3.4 Teplotní senzor

Nyní připojíme teplotní senzor DS18B20 k VoCore. Pro připojení využijeme tři vodiče a zapojíme jeho piny dle následujícího schéma:



Obrázek 15 - *Pin 1 (GND) - GND; Pin 2 (DQ) - GPIO 19; Pin 3 (V_{DD}) – napájení 3,2 V – 3,5 V [11, 19].*

Z manuálu je patrné, že k napájení potřebujeme rozsah napětí 3,0 V až 5,5 V [23]. Nesmíme zapomenout na odpor velikosti 4,7 kΩ, který připojíme mezi napájení a datovou linkou. Připojíme senzor k VoCore. Zpustíme zařízení a podíváme se do adresáře */sys/bus/w1/devices/* nebo */sys/devices/w1_bus_master1/*, ve kterých by se měl objevit další adresář jmenující se 28-xxx. Pokud k tomuto došlo, VoCore rozpoznal připojený senzor a umí s ním komunikovat. Data ze senzoru získáme ze souboru *w1_slave*, který se nachází v adresáře /28-xxx/. V našem případě je cesta k tomuto souboru */sys/devices/w1_bus_master1/28-0000084895dc/w1_slave*, kde *28-0000084895dc* je identifikační číslo senzoru.

Uvedeme jenom samotné načítání a zpracování dat ze senzoru.

```
-- Načteme data ze senzoru
f = io.open("/sys/devices/w1_bus_master1/28-0000084895dc/w1_slave", "r")
t = f:read("*all")
f:close()

-- Zpracování řetězce
p = string.find(t, "t=")
teplota = tonumber(string.sub(t, p+2))/1000 -- protože teplota je uvedená bez rozlišení desetinného místa
```

Ukládání a vykreslení dat se provede podobně jako u napájecího zdroje.

7.3.5 Zobrazení dat

Zobrazení již nasbíraných dat bylo provedeno pomocí nástroje rrdtool. V podstatě veškerý postup vizualizace dat spočívá ve splnění třech následujících kroků:

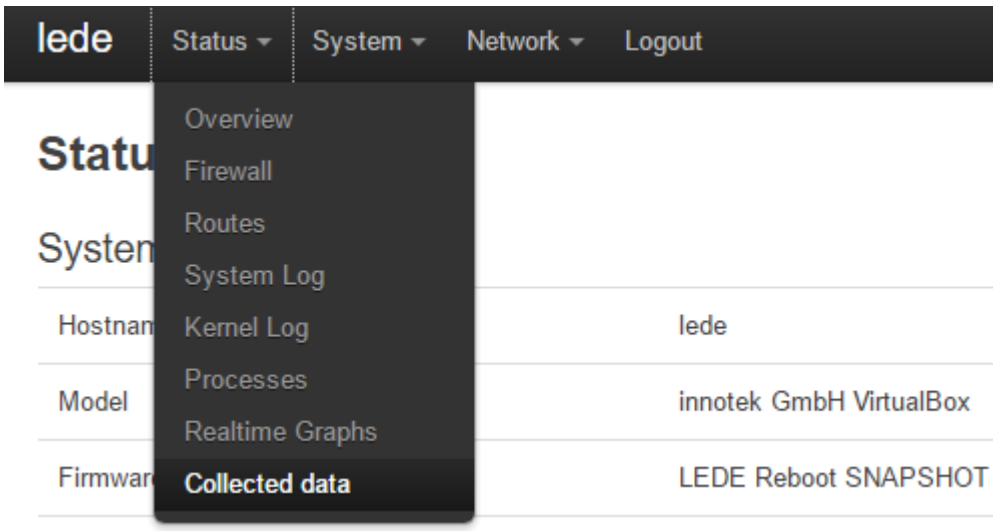
1. Nasbírat data do souborů formátu .rrd
2. Vytvořit z těchto souborů obrázky formátu .png
3. Přidat tyto obrázku na webovou stránku (ideálně vytvořit novou položku)

Krok č. 1 a č. 2 jsme splnili ve dvou minulých kapitolách a máme data uložená v souborech napeti.rrd, proud.rrd a teplota.rrd v adresáři */www/luc-static/resources/monitorovani/*. Taky máme hotovou vizualizaci – obrázky napeti.png, proud.png a teplota.png. Zbývá je implementovat do webového rozhrání [28].

Máme k dispozicí tři obrázky s naměřenými průběhy, které chceme vykreslit v grafickém rozhraní LuCI. Vytvoříme novou položku v menu „Status“, která se bude jmenovat „Collected data“. Abychom toto zrealizovali, vytvoříme soubor *statistics.lua*, který přidá novou položku v menu „Status“. Soubor musíme uložit do složky */usr/lib/lua/luci/controller/*. Kód programu pro přidání nové položky je následující:

```
module("luci.controller.statistics", package.seeall)

function index()

entry({"admin", "status", "statistics"}, alias("admin", "status", "statistics", "grafy"), _("Collected data"), 80)
-- Vytvoření nové položky na pozici 80, která bude poslední v pořadí. Po její rozkliknuti se defaultně zobrazí grafy
entry({"admin", "status", "statistics", "grafy"}, template("statistics/grafy"), _("Grafy"), 10).leaf=true
--Vytvoření první sekce, ve které budeme pozorovat grafy. Za provoz grafů je zodpovědný soubor grafy.htm umístěny ve složce .../luci/view/statistics/…
entry({"admin", "status", "statistics", "nastaveni"}, cbi("statistics/nastaveni"), _("Nastaveni"), 20).leaf=true
-- Vytvoření druhé sekce, ve které budeme měnit nastavení. Za to bude zodpovědný soubor nastaveni.lua umístěny ve složce .../luci/model/cbi/statistics/…

end
```

V menu „Status“ by se tedy měla objevit nová položka, která zatím neumí nic zobrazit.



Obrázek 16

Naprogramujeme obsah této položky a umožníme uživateli měnit nastavení [30]. Vytvoříme další soubor – nastaveni.lua - a umístíme jej v následujícím adresáři - */usr/lib/lua/luci/model/cbi/statistics/*. Tedy celková cesta k souboru by měla vypadat takto - */usr/lib/lua/luci/model/cbi/statistics/nastaveni.lua*. Následující kód je schopen načíst a změnit data v konfiguračním souboru /etc/config/nastaveni, který obsahuje některé údaje umožňující uživateli měnit parametry výstupních obrázků. Bohužel, po každé provedené změně uživatelem, se zařízení musí restartovat, aby bylo možné aplikovat nová nastavení.

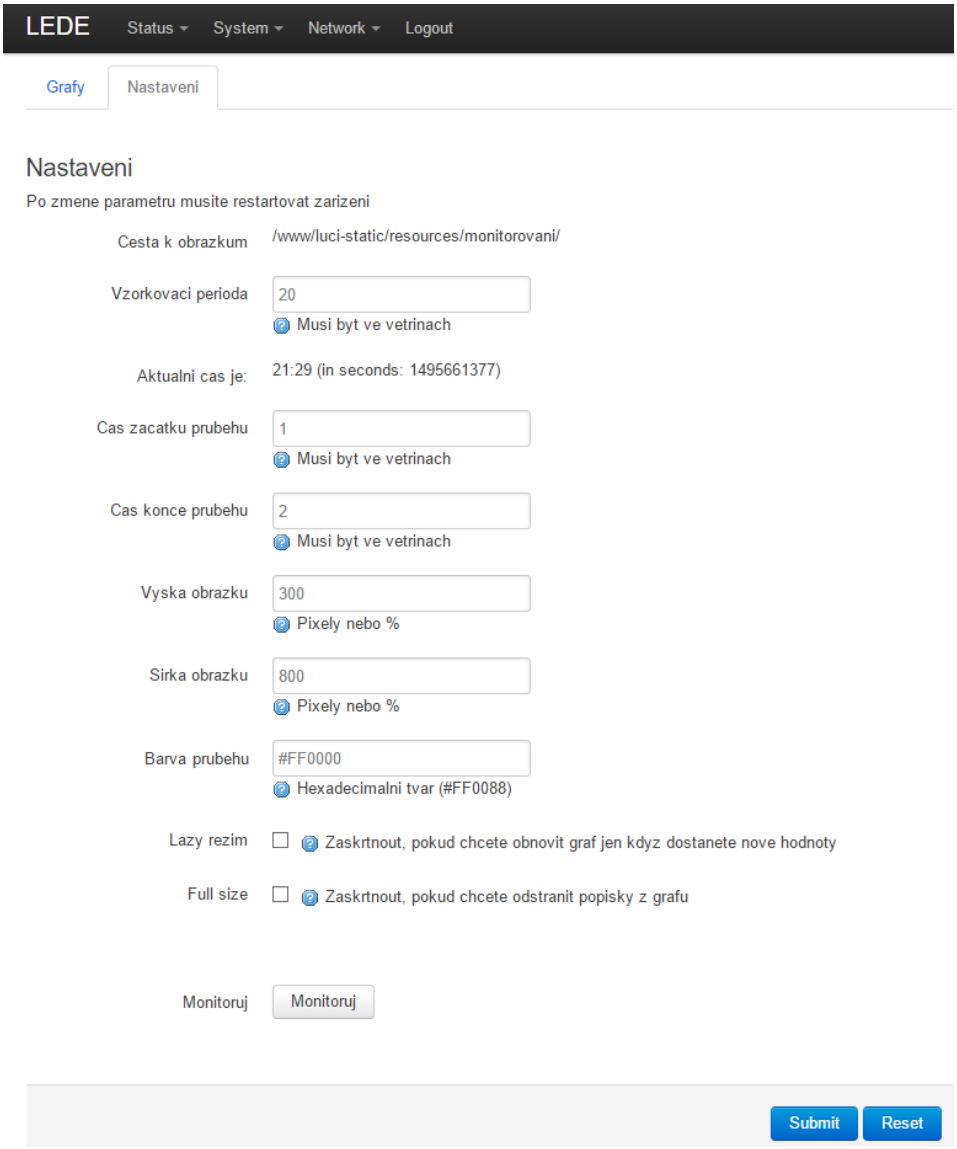
```
m = Map("nastaveni", nil, nil)
-- Mapuje se konfigurační soubor nastavení umístěný v /etc/config/
n = m:section(NamedSection, "nastaveni", "nastaveni", translate("Nastaveni"))
-- Adresace na sekci nastavení v souboru /etc/config/nastaveni
sr = n:option(DummyValue, "src", translate("Cesta k obrazkum"))
-- Adresace k vlastnosti src v souboru /etc/config/nastaveni

-- Nastaveni doby načítání vzorku
tout = n:option(Value, "timeout",
translate("Vzorkovaci perioda"), translate("Musi byt ve vetrinach"))
tout.rmempty = false -- Důležitý parametr, bez kterého nebude fungovat program

-- Zobrazení aktuálního času, pro snadnější nastavení dvou dalších polí
x = n:option(DummyValue, "True", translate("Aktualni cas je: "))
text = os.date()
p = string.find(text, ":")
x.default = string.sub(text, p-2, p+2) .. " (in seconds: " .. os.time() .. ")"

st = n:option(Value, "tstart",
translate("Cas zacatku prubehu"), translate("Musi byt ve vterinach"))
st.rmempty = true -- Tento parametr není nutný pro zobrazení obrázku a proto můžeme
povolit prázdnou hodnotu na vstupu (v případě true - vymaže ze souboru tuto hodnotu).

e = n:option(Value, "tend",
translate("Cas konce prubehu"), translate("Musi byt ve vterinach"))
e.rmempty = true -- Taky není nutný pro zobrazení obrázku
```

```
h = n:option(Value, "height",
translate("Vyska obrazku"), translate("Musi byt v pixelech nebo v %"))
h.rmempty = false -- Zde zakážeme uživateli možnost vymazat tuto sekci, pokud na vstupu dostaneme prázdnou hodnotu

w = n:option(Value, "width",
translate("Sirka obrazku"), translate("Musi byt v pixelech nebo v %"))
w.rmempty = false

c = n:option(Value, "colour", translate("Barva prubehu"),
translate("Musi byt v hexadecimalnim tvaru (#FF0088)"))
c.rmempty = false

l = n:option(Flag, "lazy", translate("Lazy rezim"),
translate("Zaskrtnout, pokud chcete obnovit jen kdyz dostanete nove hodnoty"))
l.rmempty = true

fs = n:option(Flag, "fullsize", translate("Full size"),
translate("Zaskrtnout, pokud chcete odstranit popisky"))
fs.rmempty = true

f = SimpleForm("Galvanika", nil, nil)
t = f:section(SimpleSection, nil, nil)

--Přidáme tlačítko na nastartování programu
mon = t:option(Button, "check", translate("Monitoruj"))
mon.inputstyle = "enable"
mon.write = function(self, section)
return require ("luci.sys").call
('lua /www/luci-static/resources/monitorovani/monitorovani.lua')
end

return m, f
```

Konfigurační soubor spolupracující s tímto programem vypadá takto:

```
config section 'nastaveni'
        option src '/www/luci-static/resources/monitorovani/'
        # adresář, kde budou uloženy obrázky
        option timeout ‘20’                     # vteřiny
        option height '300'                     # pixely
        option width '800'                      # pixely
        option colour '#FF0000'                 # hexadecimální tvar
        option tstart '1'                       # vteřiny
        option tend '2'                         # vteřiny
        option lazy '0'                         # obnovit, pokud dojde k obnoveni obrázku (v důsledku nových údajů)
      option fullsize '0'           # -D --full-size-mode - odstranění popisku v grafu
```

Výsledek práce tohoto kódu je na následujícím obrázku.



Obrázek 17 - LuCI, sekce nastavení

Nyní zbývá naprogramovat samostatné vykreslení těchto průběhů. Vytvoříme soubor grafy.htm v následujícím adresáři - /usr/lib/lua/luci/view/statistics/.

```
<html>
        <%+header%> <!-- Přidání hlavičky na stránku -->
<body>
<h3>Zde jsou grafy</h3>
<%
        local teplota = resource .. "/monitorovani/teplota.png"
        local napeti = resource .. "/monitorovani/napeti.png"
        local proud = resource .. "/monitorovani/proud.png"
```

-- Zde je do proměnných uložena cesta k obrázkům. Parametr resource obsahuje cestu k adresáři /resource,

-- ve kterém se nachází veškerá animace a obrázky použité v LuCI. Je rozumné ukládat vlastní obrázky sem,

-- abychom se nemuseli obtěžovat se správným nastavením uhttpd serveru. V adresáři /resource

```
-- vytvoříme adresář /monitorovani, který bude obsahovat veškeré řídicí soubory a jejich produkty -> obrázky.
%>
<h5>Prubeh teploty</h5>
        <img src="<%=teplota%>"/>
                <!-- Přidání obrázku teplota.png na stránku -->
<h5>Prubeh napeti</h5>
        <img src="<%=napeti%>"/>
<h5>Prubeh proudu</h5>
        <img src="<%=proud%>"/>
</body>
        <%+footer%> <!-- Přidání patičky na stránku -->
</html>
```

Tento jednoduchý kód umožní vykreslit obrázky [30], vyprodukované nástrojem rrdtool. Pokud chceme zaktualizovat informace, musíme obnovit stránku.

7.4 Zhodnocení dosažených výsledků

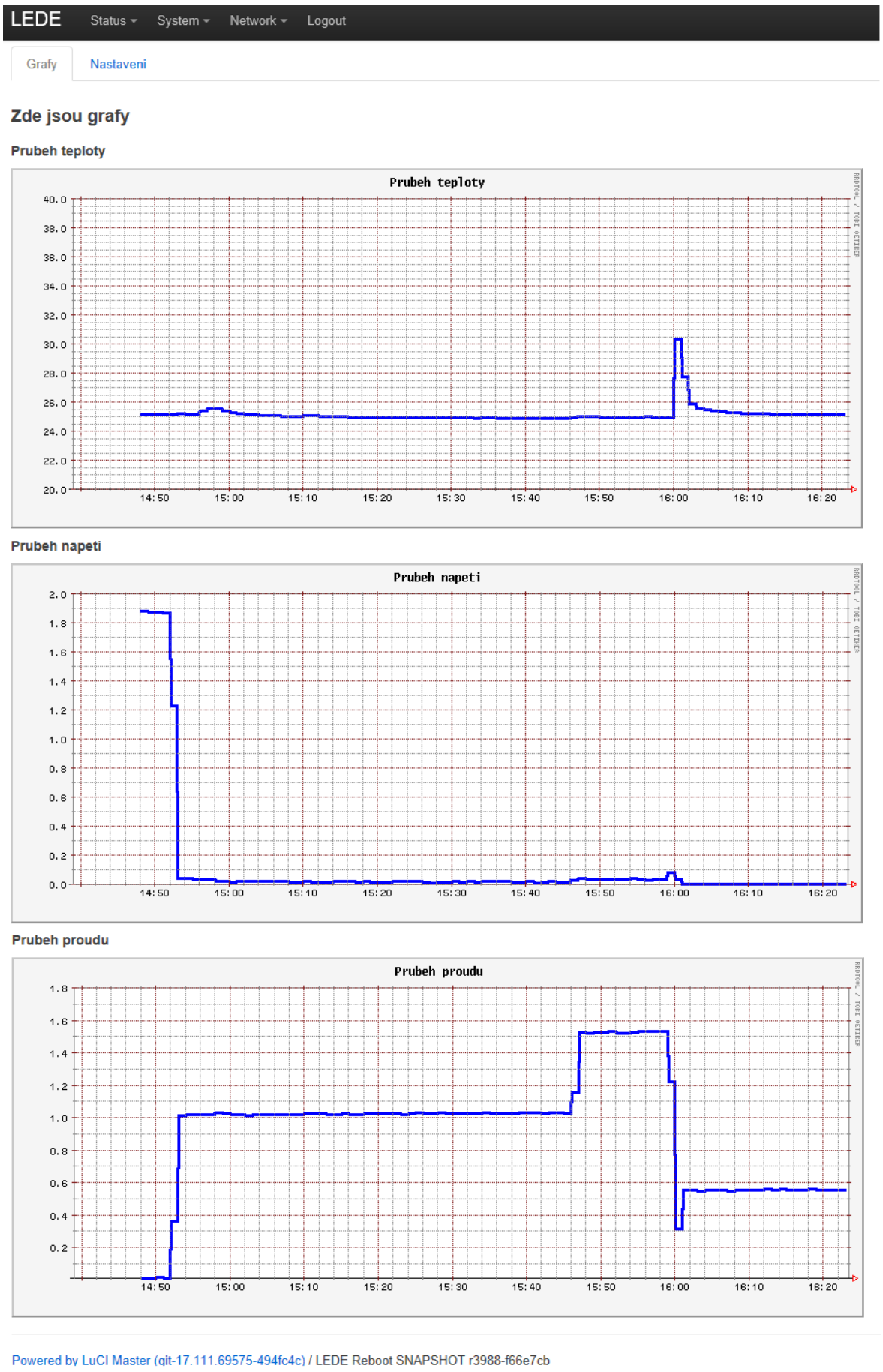
Nyní představíme a zhodnotíme dosažené výsledky po provedených krocích konfigurace v minulých kapitolách. Ukládání dat bylo provedeno v interní paměti zařízení, a to do souborů .rrd poskytnutých nástrojem rrdtool. Výhody a nevýhody tohoto řešení jsme probrali v kapitole 6.5 Cloud vs. Interní paměť. Veškeré soubory nutné pro správné konání monitorovacího procesu, včetně sbíraná data, jsme uložili do adresáře */www/luci-static/resources/monitorovani/*. Ten byl zvolen především z důvodu jednoduchosti následného implementování obrázků s průběhy do webového rozhrání (viz. kapitola 7.3.5 Zobrazení dat). Samotné sbírání dat se provádělo pomocí vlastního programu v jazyce Lua (monitorovani.lua), který byl umístěn ve stejném adresáři spolu s obrázky.

/www/luci-static/resources/monitorovani

Name	Size	Changed	Rights	Owner
..		21.04.2017 14:57:21	rwxr-xr-x	root
monitorovani.lua	6 KB	21.05.2017 19:47:34	rw-r--r--	root
napeti.png	54 KB	15.05.2017 18:39:45	rw-r--r--	root
napeti.rrd	80 KB	15.05.2017 18:40:14	rw-r--r--	root
proud.png	54 KB	15.05.2017 18:39:48	rw-r--r--	root
proud.rrd	80 KB	15.05.2017 18:40:14	rw-r--r--	root
teplota.png	62 KB	15.05.2017 18:39:53	rw-r--r--	root
teplota.rrd	80 KB	15.05.2017 18:40:14	rw-r--r--	root

Obrázek 18

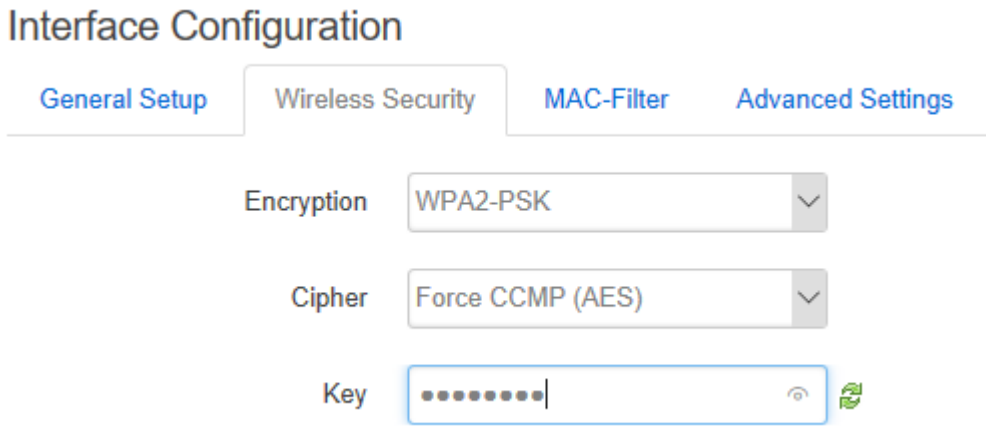
Komunikace mezi napájecím zdrojem a VoCore se prováděla sériovým rozhraním, se senzorem – pomocí 1-wire protokolu (kapitoly 7.3.3 Napájecí zdroj a 7.3.4 Teplotní senzor). Získané hodnoty napětí, proudu a teploty byly uloženy a znázorněny pomocí nástroje rrdtool, kterému jsme věnovali kapitolu 7.3.2 Práce s rrdtool. Grafický výstup měření vypadá takto:


LEDE
Status
System
Network
Logout
Grafy
Nastaveni
Zde jsou grafy
Prubeh teploty
Prubeh teploty
40.0
38.0
36.0
34.0
32.0
30.0
28.0
26.0
24.0
22.0
20.0
14:50
15:00
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
Prubeh napeti
Prubeh napeti
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
14:50
15:00
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
Prubeh proudu
Prubeh proudu
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
14:50
15:00
15:10
15:20
15:30
15:40
15:50
16:00
16:10
16:20
Powered by LuCI Master (git-17.111.69575-494fc4c) / LEDE Reboot SNAPSHOT r3988-f66e7cb

Obrázek 19

Navržený systém je spolehlivý. Pokud dojde k poruše VoCore, napájecí zdroj bude stále fungovat, je nezávislý na VoCore. Proces galvanického vylučování kovů se nezastaví. Pokud se stane, že se útočníkovi podaří proniknout do systému, nepoškodí to samotný proces galvanoplastiky. VoCore zajišťuje pouze sběr dat, neposkytuje ovládání a možnost měnit parametry. Jediné, co může v tomto případě provést útočník, je změnit hodnoty proudu a napětí, což bude znázorněno ve výsledném průběhu, nebo vypnout napájení (odpojením od svorek zdroje). Aby k výše zmíněnému případu vniknutí přes rozhraní Wi-Fi sítě nedošlo, diskutoval jsem možnosti zabezpečení v kapitole 6.5. Pokud se tak stane, dojde k jedné z výše zmíněných situací a ovlivní se pouze výsledný report (zobrazené grafy a logy z měření). Samotnému procesu uškodí pouze fyzický zásah. Pokud dojde k poškození teplotního senzoru, VoCore pouze přestane sbírat data. Probíhajícímu technologickému procesu to také nepoškodí.

Řešení bezpečnosti úlohy bylo vysvětleno v kapitole 6.5 Zabezpečení. Dle popsaných možností jsem zvolil šifrování WPA2-PSK s šifrovacím klíčem CCMP(AES). Přidal jsem také do databáze MAC-adresu vlastního přístupového zařízení (nebudu jí uvádět z vlastních bezpečnostních důvodu). Taky jsem v systému nastavil heslo pro uživatele *„root“* (pro přístup k webovému rozhraní).



Obrázek 20

Navržený systém má několik slabých míst:

- Deklarované rozlišení teplotního senzoru 0,5°C je dobré rozlišení, ale ve výsledném zobrazení průběhu způsobí rozkmitání hodnot, a to i kdyby aktuální hodnota byla stejná (jak je vidět na obrázku 19). Opravu lze realizovat dle pokynů popsaných v [18];
- Komunikace sériovým rozhraním – pokud obsadíme sériový port programem na načítání dat, nemůžeme již znovu otevřít port a komunikovat se zdrojem pro jiné účely. Tento stav způsobí určité potíže při implementace možnosti vzdáleného ovládání zdroje;
- Relativně velká spotřeba elektrické energie realizovaného systému (kolem 250 mA [31]), což znemožňuje použití bateriového napájení po dlouhou dobu.

Ovšem systém má i mnoho zajímavých výhod:

- Ukládání velkého množství dat v rámci omezené paměťové kapacity;
- Lokální síť bez přístupu na internet zajistila zabezpečení proti útokům „ze vnějšku“;
- Malé rozměry a snadná integrace do procesu;
- Odolnost vůči případné poruše monitorovacího systému – samotný technologický proces to nepoškodí.

Obrázek 21 – navržený systém

VoCore + Dock je připojen k napájecímu zdroji převodníkem USB-seriál (černý kabel). Teplotní senzor připojen pomocí třech vodičů (červený, zelený, černý) na GPIO porty zařízení. Samotný VoCore se napájí z baterky, ke které je připojen přes USB (bílý kabel). Ovládání zdroje se provádí knoflíky „Voltage“ a „Current“. Nastavené hodnoty objeví na svorkách „AUX Output“.

Obrázek 22 – navržený systém

8. Závěr

Během této práce jsem prozkoumal komponenty pro návrh systému podpory galvanického vylučování kovů v koncepci Internetu věcí. Přiklonil jsem se k použití VoCore + Dock, který poskytuje dobrý výkon při malých rozměrech a relativně nízké ceně. Také jsem zvolil zcela populární teplotní senzor DS18B20 od firmy MAXIM. Jeho hlavní výhodou je dobrá citlivost v požadovaném rozsahu měřených teplot. Napájecí zdroj Manson HCS-3400 USB usnadnil proces sbírání hodnot napětí a proudů. Před možnosti využívat Cloud pro ukládání dat, jsem dal přednost interní pamětí zařízení. Výhody a nevýhody tohoto řešení jsou probrané v kapitole 6.6 Cloud vs. Interní paměť zařízení.

V průběhu vypracování této úlohy jsem se potkal s různorodými problémy, které zahrnovali rozsáhlou problematiku programování embedded zařízení – kompilace systému, mapování GPIO portů, práce s balíčky, dohledávání různých informací (např. kapitola 7.3.2 práce s rrdtool), připojení komponentů, oživení zařízení po špatném upgradu a nahrávání nové firmwaru pomocí bootloaderu. Pokusil jsem se ošetřit všechny možné chyby, ke kterým by mohlo dojít.

Dosažené výsledky nejsou konečné. Tento systém lze samozřejmě dále rozvíjet. Budoucí aktivity by mohly realizovat následující body:

- umožnit uživateli vzdálené ovládání zdroje – manuál k napájecímu zdroje [16] obsahuje další příkazy na ovládání, včetně možností nastavit hodnotu. Musí se vyřešit komunikace sériovým portem (respektive vyřešit způsob, jak sdílet komunikační rozhrání);
- vytvořit možnost predikování tloušťky vyloučené vrstvy na vzorku, v závislosti na čase a proudovém množství – použijí se vztahy z kapitoly 2, které aplikujeme na aktuální hodnoty získané z procesu;
- umožnit nastavování času konání procesu, nebo volit tloušťku požadované vrstvy, což jsou de-facto vzájemně úměrné veličiny – realizuje se snadno pomocí časovače a vypínání napětí na svorkách příkazem „SOUT1“ [16];
- realizovat hlášení ukončení procesu posíláním SMS nebo e-mailu – požaduje se připojení na internet a odpovídající aplikace (již existující nebo vlastní), která posílá buď e-maily, nebo push-notifikace na mobil;
- připojit VoCore na Internet a vytvořit webovou aplikaci, která by umožňovala objednávat výrobky dle potřeb zákazníku, nastavit nutné parametry ke konání tohoto procesu. Automaticky by se vypočítávali parametry a následně zaslali do výroby, kde úloha bude zařazená do fronty. Zákazník bude vědět, za jak dlouho jeho požadavek bude splněn – realizuje se snadno na tomtéž nebo dalším embedded zařízení s použitím stávající infrastruktury LuCI. Musíme připojit zařízení na internet a vyřešit bezpečnost v síti. Pro tyto účely je možné použit Cloud.

8.1 Ekonomické zhodnocení

Celková cenová rozvaha komponentů tohoto projektu je znázorněná v následující tabulce:

Součástka	**Cena**	**Počet kusů**
Manson HCS-3400 USB	6200 Kč	1x
VoCore + Dock	1100 Kč	1x
DS18B20	68 Kč	1x
Celkově:	7368 Kč	

Tabulka 1

Uvedené ceny jsou přibližné a byly převzaty z webových stránek e-shopů [32, 33, 34] dne 15.5.2017. Nejdražším komponentem je napájecí zdroj, který je zpravidla součástí zařízení pro konání procesu. Ostatní položky – embedded zařízení a teplotní senzor - jsou relativně levné. Dle mého názoru, se podařilo navrhnout požadovaný nízkonákladový systém.

9. Seznam použitých zdrojů

[1] BORODIN Vladimir Alexeevich. The Internet of things – the next stage of the digital revolution. In: *Educational resources and technologies* [online]. Moscow Witte University, 2014. [cit. 2.3.2017]. Dostupné z: http://www.muiv.ru/vestnik/pdf/pp/ot_2014_2_178-182.pdf

[2] ALBERT Mark. MTConnect in Context. In: *Modern Machine Shop* [online]. Gardner Business Media, Inc., 2015. [cit. 2.3.2017]. Dostupné z: http://www.mmsonline.com/columns/mtconnect-in-context

[3] HEL Ilya. Průmysl 4.0: co to je 4. Průmyslová revoluce?. In: *Hi-news.ru* [online]. i10.ru, 2015. [cit. 2.3.2017]. Dostupné z: https://hi-news.ru/business-analitics/industriya-4-0-chto-takoe-chetvertaya-promyshlennaya-revolyuciya.html

[4] Electroforming = Galvanoplastika [online]. Electroforming s.r.o. [cit. 5.3.2017]. Dostupné z: http://www.electroforming.cz/cs/technologie

[5] Elektrolýza. In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation. Stránka byla naposled edit. 2.5.2017 v 12:19. [cit. 15.5.2017]. Česká verze. Dostupné z: https://cs.wikipedia.org/wiki/Elektrol%C3%BDza

[6] Průběh elektrolýzy. In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation, 2008. [cit. 5.3.2017]. Dostupné z: https://cs.wikipedia.org/wiki/Elektrol%C3%BDza#/media/File:Elektrol%C3%BDza.jpeg

[7] ODNORALOV N.V. Galvanoplastika pro domácnost. In: *Vsyakaya Vsyačina* [online]. Ananiev V.V. [cit. 9.3.2017]. Dostupné z: http://anytech.narod.ru/homegalv.htm

[8] AFANASYEV Y. Galvanoplastika. In: *Vsyakaya Vsyačina* [online]. Ananiev V.V. [cit. 10.3.2017]. Dostupné z: http://anytech.narod.ru/galvano.htm

[9] ROUSE Margaret. Definitions embedded systém. In: *IoT agenda TechTarget* [online]. TechTarget, 2016 [cit. 21.3.2017]. Dostupné z: http://internetofthingsagenda.techtarget.com/definition/embedded-system

[10] VoCore. In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation. Stránka byla naposled edit. 27.4.2017 v 07:13. [cit. 30.4.2017]. Anglická verze. Dostupné z: https://en.wikipedia.org/wiki/VoCore

[11] *VoCore (Old version)* [online]. VoCore Studio, 2014. [cit. 30.4.2017]. Dostupné z: http://vocore.io/v1.html

[12] *Manuálové stránky pro VoCore* [online]. VoCore Studio, 2014. [cit. 30.4.2017]. Dostupné z: http://vonger.cn/upload/vocore.manual.pdf

[13] *VoCore2: $4 Coin-sized Linux Computer with WiFi* [online]. Indiegogo, 2016. [cit. 30.4.2017]. Dostupné z: https://www.indiegogo.com/projects/vocore2-4-coin-sized-linux-computer-with-wifi#/

[14] *VoCore2: Release Version Mount Map* [online]. VoCore Studio, 2016. [cit. 30.4.2017]. Dostupné z: http://vonger.cn/?p=2666

[15] *VoCore 2 Ultimate* [online]. VoCore Studio, 2016. [cit. 30.4.2017]. Dostupné z: http://vocore.io/v2u.html

[16] *Manuálové stránky pro Manson HCS-3400* [online]. Manson Engineering Industrial Ltd. [cit. 10.4.2017]. Dostupné z:

http://www.manson.com.hk/getimage/index/action/images/name/5652d6a6211ea.pdf
[17] Remote programming lab. Grade switching mode power supply *Manson HCS-3400* [online]. Manson Engineering Industrial Ltd. [cit. 10.4.2017].
Dostupné z: http://www.manson.com.hk/products/detail/158
[18] *Jak zlepšit přesnost DS18B20* [online]. AdVentX. [cit. 15.4.2017].
Dostupné z: https://www.adventx.com.ua/automation/article/ds18b20_precision/
[19] *Manuálové stránky pro Dallas DS18B20* [online]. maxim integrated. [cit. 15.4.2017].
Dostupné z: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
[20] Wired Equivalent Privacy. In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation. Stránka byla naposled edit. 17.2.2017 v :1703. [cit. 19.5.2017]. Česká verze.
Dostupné z: https://cs.wikipedia.org/wiki/Wired_Equivalent_Privacy
[21] Wi-Fi Protected Access. In: *Wikipedie: otevřená encyklopedie* [online]. Wikimedia Foundation. Stránka byla naposled edit. 16.6.2016 v 11:51. [cit. 19.5.2017]. Česká verze.
Dostupné z: https://cs.wikipedia.org/wiki/Wi-Fi_Protected_Access
[22] *Manuálové stránky ke kompilaci systému LEDE* [online]. LEDE project. [cit. 17.4.2017].
Dostupné z: https://lede-project.org/docs/guide-developer/install-buildsystem
[23] *Manuálové stránky na kompilaci firmware pro VoCore + Dock* [online]. VoCore Studio. [cit. 20.4.2017]. Dostupné z: http://vonger.cn/?p=1934
[24] *Manuálové stránky rrdtool graph* [online]. Tobias Oetiker. [cit. 22.4.2017].
Dostupné z: https://oss.oetiker.ch/rrdtool/doc/rrdgraph.en.html
[25] *Manuálové stránky rrdtool create* [online]. Tobias Oetiker. [cit. 22.4.2017].
Dostupné z: https://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html
[26] *Manuálové stránky rrdtool* [online]. Tobias Oetiker. [cit. 22.4.2017].
Dostupné z: http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html
[27] *Manuálové stránky Init Scripts* [online]. OpenWRT Wiki. [cit. 25.4.2017].
Dostupné z: https://wiki.openwrt.org/doc/techref/initscripts
[28] *Manuálové stránky knihovny lua-rs232* [online]. [cit.26.4.2017].
Dostupné z: http://ccgi.dougrice.plus.com/cgi-bin/wiki.pl?Serial-Lua
[29] *Manuálové stránky knihovny luci.sys* [online]. [cit. 27.4.2017].
Dostupné z: https://luci.pandorabox.org.cn/en/modules/luci.sys.html
[30] *Manuálové stránky na strukturu LuCI* [online]. [cit. 23.4.2017].
Dostupné z: https://github.com/openwrt/luci/wiki
[31] *Power consumption reduction* [online]. [cit. 23.5.2017]
Dostupné z: http://forum.vocore.io/viewtopic.php?f=10&t=252
[32] *Manson HCS-3400 USB. Zdroj programovatelný laboratorní* [online]. TME Czech Republic s.r.o. [cit. 21.5.2017]. Dostupné z:
http://www.tme.eu/cz/details/hcs-3400-usb/jednokanalove-napajeci-zdroje/manson/
[33] *VoCore v1.0 – OpenWRT Linux SBC with WiFi and Docking Board [0159-V]* [online]. ameriDroid. [cit. 21.5.2017]. Dostupné z: http://ameridroid.com/products/vocore-v1-0-openwrt-linux-sbc-with-wifi-and-docking-board
[34] *DS18B20* [online]. GME electronic, spol s.r.o. [cit. 21.5.2017].
Dostupné z: https://www.gme.cz/ds18b20