VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace systému pro správu záložních zdrojů napájení na platformě Raspberry Pi

Implementation of system for Management of Backup Power Supply Sources on Raspberry Pi platform

2019 Tomáš Jindra

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Jindra**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Implementace systému pro správu záložních zdrojů napájení na platformě Raspberry Pi.
Implementation of System for Management of Backup Power Supply Sources on Raspberry Pi platform.

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je implementovat modulární Open Source systém pro správu záložních zdrojů napájení. Důraz bude kladen na modularitu daného řešení ve vztahu k možným druhům zdrojů záložního napájení - především generátory, baterie a obnovitelné zdroje. Součástí systému bude možnost monitoringu skrze webové rozhraní. Předpokládá se rovněž implementace API rozhraní pro snadnou správu a automatizaci.

1. Analyzujte možnosti záložního napájení a stávající implementace systému pro jejich správu.
2. Navrhněte vlastní řešení, a to jeho hardwarovou i softwarovou stránku.
3. Řešení implementujte ve zvoleném programovacím jazyce - součástí implementace musí být podpora alespoň 3 typů zdrojů napájení.
4. Hotové dílo řádně zdokumentujte a otestujte.

Seznam doporučené odborné literatury:

[1] UPTON, Eben; HALFACREE, Gareth. Raspberry Pi user guide. Chichester, West Sussex, UK: Wiley, c2012, 248 p. ISBN 978-1-118-46446-5.
[2] HERTZOG, Raphaël; MAS, Roland. The Debian administrator's handbook: Debian squeeze from discovery to mastery. Lulu Com, 2012. ISBN 979-109-1414-005.
[3] BRADBURY, Alex; EVERARD, Ben; WINDER, Russe. Learning Python with Raspberry Pi. John Wiley & Sons, 2014. ISBN 978-1118717059.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Daniel Stříbný**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2019 .....

Na tomto místě bych rád poděkoval vedoucímu mé práce, panu Ing. Danielovi Stříbnému, za jeho mimořádnou ochotu a optimismus, jež velmi podpořili vývoj této práce.

Abstrakt

Tato bakalářská práce je zaměřená na analýzu a vývoj systému pro správu záložních zdrojů napájení za využití platformy Raspberry Pi. K řešení softwaru byl použit programovací jazyk Python pro vývoj ovládacího programu a společně s webovým frameworkem Django k vytvoření webových stránek. Stránku hardwarové části zastávají použité součástky pro monitoring stavu systému zároveň se součástkami pro jeho ovládání. Vytvořené řešení dává uživateli k nahlédnutí detaily stavu systému a současně poskytuje možnost kontroly nad průběhem. Nicméně systém funguje automaticky bez nutnosti zásahu.

Klíčová slova: Raspberry Pi, zdroje napájení, Python, Django, bakalářská práce

Abstract

This bachelor thesis is focused on analysis and development of a system for management of backup power sources with use of Raspberry Pi platform. For software solution was used Python programming language for development of control program and together with Django web framework for creation of website. On the behalf of the hardware part components for monitoring of the system status were used along with components for the system control. The created solution provides user with a view of detailed information about the state of system, while enabling him to control it manually. Nevertheless the system works automatically without the need for intervention.

Key Words: Raspberry Pi, power sources, Python, Django, bachelor thesis

Obsah

Seznam použitých zkratek a symbolů

HTML	–	Hyper Text Markup Language
WWW	–	World Wide Web
API	–	Application Programming Interface
USB	–	Universal Serial Bus
A/D	–	analog-to-digital converter
ADC	–	analog-to-digital converter
UDP	–	User Datagram Protocol
UDP	–	General-purpose input/output
JS	–	JavaScript
CLI	–	Command line interface
JS	–	Graphical user interface
IP	–	Internet Protocol address
HDMI	–	High-Definition Multimedia Interface
URL	–	Uniform Resource Locator

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce se zabývá problematikou náhradního napájení sítě ze záložních zdrojů systémem, jež kontroluje jeho funkci. Tento systém by měl fungovat automaticky bez nutnosti zásahu uživatele. Pokud však uživatel bude chtít do systému zasáhnout a při běhu něco změnit, měl by mu k tomu poskytnout možnost. Zároveň by měl uživateli poskytnout možnost na webových stránkách sledovat stav systému díky kterému bude mít uživatel přehled nejen o momentální situaci, ale také jednodušší orientaci v případě rozhodnutí o zásahu do průběhu. Tohoto se dosáhne pomocí implementace různých monitorovacích prvků, které budou zároveň pomáhat lepší funkcionalitě celého systému. Výsledný produkt by měl být schopen sám ovládat stav sítě, přizpůsobovat se různým situacím a poskytovat možnost rozšíření o případné další moduly, jež by měly býti schopny společně spolupracovat a vytvářet ve finále větší, složitější a spolehlivější systém. Takovýto systém si umíme přestavit v praxi při napájení důležitých části budov, například serverovny obsahující důležitá data, vyžadující neustálý přísun elektrického proudu. To však nevyřazuje možnosti použití v zařízeních menšího rázu a významu. Právě naopak by se tento systém hodil pravděpodobně nejvíce pro tyto účely, díky minimální formě potřebných zařízení a možnosti využití nepříliš finančně náročných prvků. Například pro soukromou domácnost, kde by pro se pro něj našlo využití v případě, že by se ocitla bez elektrického proudu, třeba po přírodní živelné pohromě, jež by měla za následek výpadek dodávky elektrického proudu v dané oblasti.

Celá práce se dá rozdělit na několik částí, ve kterých budou postupně popisovány jednotlivé kroky, které by bylo zapotřebí udělat v případě ambicí o vytvoření funkčního systému, jako je tento. Nejprve je nutno zabývat se možnostmi různých druhů záložního napájení, které se na trhu nabízí, společně se schématem zapojení, v jakém by fungovala. Následně by bylo vhodné se zaměřit na některé z již existujících jiných řešení a na jejich ohodnocení. Po zvolení vhodných zařízení a vytvoření odpovídajícího modelu, je třeba se zabývat vhodným systémem implementace pro jeho fungování a správu. Další části této práce se následně věnují popisu mého řešení tohoto úkolu, kde se postupně budu zabývat jednotlivě každým z výše uvedených bodů dopodrobna.

2 Analýza možností záložního napájení

V dnešní době je prakticky každá domácnost připojena k dodavateli elektřiny jediným připojovacím bodem. Co když se však vyskytne na trase problém, jež toto spojení na určitou časovou dobu znemožní. Takováto situace již sice nastane zřídka, nicméně se může přihodit prakticky kdykoliv a kdekoliv. V případě krátkého přerušení dodávky elektřiny nemusíme ani zaregistrovat, že k něčemu takovému došlo. Avšak jsou situace, kdy může porucha trvat hodiny nebo v extrémních případech dokonce i celé dny. V takovýchto situacích nám pak mohou vypomoci [5]zdroje záložního napájení, které by tento problém mohly vyřešit. Konkrétně se jedná o:

- Baterie - Dle jednotlivých typu vydrží různou dobu. Po vybití je potřeba baterii znovu nabít jiným zdrojem.
- Generátory - Existují různé typy s velkou rozmanitostí. Nejsou omezeny kapacitou jako baterie, ale pracují pouze tak dlouho, dokud mají k dispozici hnací médium. (Benzín, nafta, fotony).

Nyní se podíváme na to, jak by mělo naše schéma vypadat, protože od toho se bude vyvíjet výběr jednotlivých komponentů pro řešení. Důležité je, že určená platforma, na které systém bude fungovat, je jednodeskový počítač Raspberry Pi [1]. Raspberry Pi má vstup napájení 5V, 2A, které může být dodáváno z různých typů zdrojů za pomocí převodníků či k tomu uzpůsobeného kabelu (například z USB hubu). Nicméně, když bude Raspberry Pi s naším programem napojená přímo na elektrický okruh, při výpadku proudu vypadne také, čímž nám problém nebude schopné vyřešit. Takovéto situaci je proto třeba předejít použitím minimálně jedné baterie v daném obvodu. Když bude Raspberry Pi napájeno z baterie, případný výpadek proudu se ho nijak nedotkne, pokud bude baterie nabitá a bude moci zásobovat Raspberry Pi. Tímto krokem máme v tuto chvíli nezávislou jednotku s naším ovládacím systémem nabíjenou baterií a zbytek záložních zdrojů můžeme poskládat dle vlastního uvážení. Avšak je třeba, aby připojené záložní zdroje, které začnou dodávat proud do naší vnitřní sítě, zároveň buďto napájeli nabíječku baterie pro Raspberry Pi, či tuto funkci obstarávala síť samotná, abychom se znovu nedostali k výpadku našeho centrálního mozku.

U generátoru je mimo napětí 230V také důležité, o jaký typ generátoru se jedná a jaký má výkon. Při použití solárních panelů či větrných generátorů se nemůžeme spoléhat na to, že budou vždy ideální podmínky pro jejich nejvyšší efektivitu a budou v době nouze vydávat své maximum. To oproti tomu umí zajistit generátory na fosilní paliva. U těchto je ale zase provoz nákladnější, hlučnější a neekologický. Z tohoto důvodu je dobré, s ohledem na co největší spolehlivost celého systému mít alespoň jeden takovýto generátor na fosilní paliva jako základ a libovolný počet jiných záložních zdrojů. Díky této podmínky bude zajištěn v případě nouze spolehlivý zdroj, na jehož použití samozřejmě nemusí dojít, pokud budou dodatečné zdroje dodávat dostatek energie.

Když jsou vybrány záložní zdroje, je potřeba umět je ovládat, tedy zapínat, vypínat a připojovat k síti dle vybraného schématu. K tomuto účelu existují spousty uzpůsobených modelů

startérů a přepínačů uzpůsobených přesně pro daný typ zařízení a kompatibilních s možností ovládání za pomocí Raspberry Pi. Nicméně pokud nezvolíme jedny z těch nejdůmyslnějších součástek na trhu, jejichž možnostem a vlastnostem bude také odpovídat i vysoká cena, bude ke správnému ovládání modelu potřeba používat jisté senzory. Například takové, které systému jsou schopny sdělit informaci, zda ze záložních zdrojů aktuálně proudí energie do sítě, či nikoliv. Konkrétně bude nutné monitorovat jisté vlastnosti záložního zdroje, jež se u odlišných typů zdrojů budou lišit, a to kvůli správnému ovládání systému. Bude pravděpodobně také žádoucí zapojit i jiné typy senzorů, jež nebudou vázány na zdroje energie, ale takovéto by již sloužily čistě jako informativní a tedy nejsou v řešení nezbytně potřebné.

Cílem práce je implementovat modulární Open Source systém pro správu záložních zdrojů napájení. Důraz bude kladen na modularitu daného řešení ve vztahu k možným druhům zdrojů záložního napájení - především generátory, baterie a obnovitelné zdroje. Součástí systému bude možnost monitoringu skrze webové rozhraní. Předpokládá se rovněž implementace API rozhraní pro snadnou správu a automatizaci.

2.1 Analýza stávajícího systému pro správu záložních zdrojů napájení

V této části bych se velmi rád věnoval jednomu z již existujících a v praxi používaných řešení. Konkrétně se jedná o klasický systém záložního napájení nemocnice za využití dieselového generátoru a kontrolního střediska.

Organizace jako banky, burzy, kasína a spousta dalších budou samozřejmě také využívat těchto systémů k zachování například běžících serverů, či funkčnosti elektrických zámků. Tak proč se zaměřit zrovna na řešení používané v nemocnici, když existuje takový rozsah jiných, případně i lákavějších míst, kam se orientovat. Odpověď na tuto otázku je jednoduchá. Lidský život nelze vyvážit zlatem či penězi. Kdyby záložní systém selhal například ve zmiňované bance či kasínu, přišlo by dané zařízení v horším případě o nějakou část svých financí a ušlý zisk za uplynulou dobu do spravení problému. Pokud by problém nastal ve společnosti zabývající se úschovnou a přístupností dat, mohla by takováto společnost ztratit všechna svá neuložená data z posledních akcí. Kdyby však k tomu samému selhání v kritické situaci došlo v nemocnici, byli by už v ohrožení nikoliv data, ale lidské životy. Jen si představte, kolik může být v takové střední nemocnici pacientů, jejichž životy závisí na funkci přístrojů nebo jsou právě v takový kritický moment operováni na sále. To je dle mého názoru přesně ten důvod, proč tyto systémy použité v nemocnicích budou o mnoho kvalitnější a preciznější, než kdekoliv jinde.

Jak takový průměrný systém správy záložních zdrojů v nemocničním středisku vlastně vypadá? Jako záložní zdroj napájení zde bývá nejčastěji použit dieselový generátor a to pro svou efektivitu, jednoduchost, spolehlivost a možnost uskladnění náhradního paliva či přívozu nového. Dieselový generátor dokáže udržet stálý neoscilující přísun energie do celé sítě. Start dieselového generátoru je také velmi rychlý, tudíž jediné negativy jsou zde neekologičnost, hlučnost, velká spotřeba paliva a cena provozu. Nicméně v dnešní moderní době se i tyto negativy dají překvapivě značně snížit a existují nemocniční centra, ve kterých uslyšíte hlasitěji sekačku na trávu

za oknem, než fungující generátor v místnosti vedle vás. Tento dieselový generátor je většinou ovládán prakticky pouhým senzorem. U dieselového generátoru se nachází baterie, která je připojená na rozvodní elektrickou síť nemocnice. Tato baterie zásobí senzor, jež detekuje přísun elektřiny ze sítě. V případě, že přísun elektřiny přestane a senzor to zjistí, vyšle signál a dieselový generátor se obratem zapne, aby zásoboval místní síť zevnitř. V celkovém pojetí minimalistický, ale dostatečný a spolehlivý systém.

Nemocniční přístroje a dokonce i některé okruhy používají dva druhy switchů. Který z nich je použit, záleží na požadavcích zařízení nebo systému. Prvním z nich je make before break switch. Tento typ switchů při přepínání z jednoho okruhu na druhý naváže s druhou stranou spojení těsně před odpojením od původního okruhu. Tímto je pro přístroj zajištěn neustálý přísun napájení. Druhým typem switchů je opak toho prvního, switch break before make. Break before make switche jsou používány například pro obracení polarity u přístrojů, které udržují svoji polaritu v jedné pozici za pomoci napětí, dokud nejsou přerušeny.

2.2 Analýza požadavků implementace systému

Pro správné pochopení požadavků, vzhledu a funkcionality výsledného produktu, je třeba si rozebrat jednotlivé části zadání.

- Implementace modulárního systému znamená, že se po nás chce vytvoření programu, jež bude pracovat s tzv. moduly. Moduly v tomto kontextu znamenají malé části, které budou použitelné na více místech a ve více případech. Když je tedy kladen důraz na modularitu vůči druhům zdrojů záložního napájení, je žádoucí, aby výsledný kód dokázal pracovat s co nejširším spektrem takovýchto zdrojů a zároveň, aby měl uživatel možnost kdykoliv připojit další.

- Implementace systému s otevřeným softwarem naznačuje, že bude výsledný produkt zadarmo volně dostupný pro širokou veřejnost, která bude mít veškeré pravomoci ho libovolně používat a upravovat.

- Z implementace monitoringu skrze webové rozhraní vyplývá, že bude třeba k finálnímu řešení rovněž implementovat různé senzory, které uživateli sdělí více o stavu jeho součástek systému či sítě. Hodnoty těchto senzorů budou následně k nahlédnutí na webových stránkách, jež budou k tomuto účelu uzpůsobeny. Zároveň budou nabízet možnost pro ruční ovládání uživatelem.

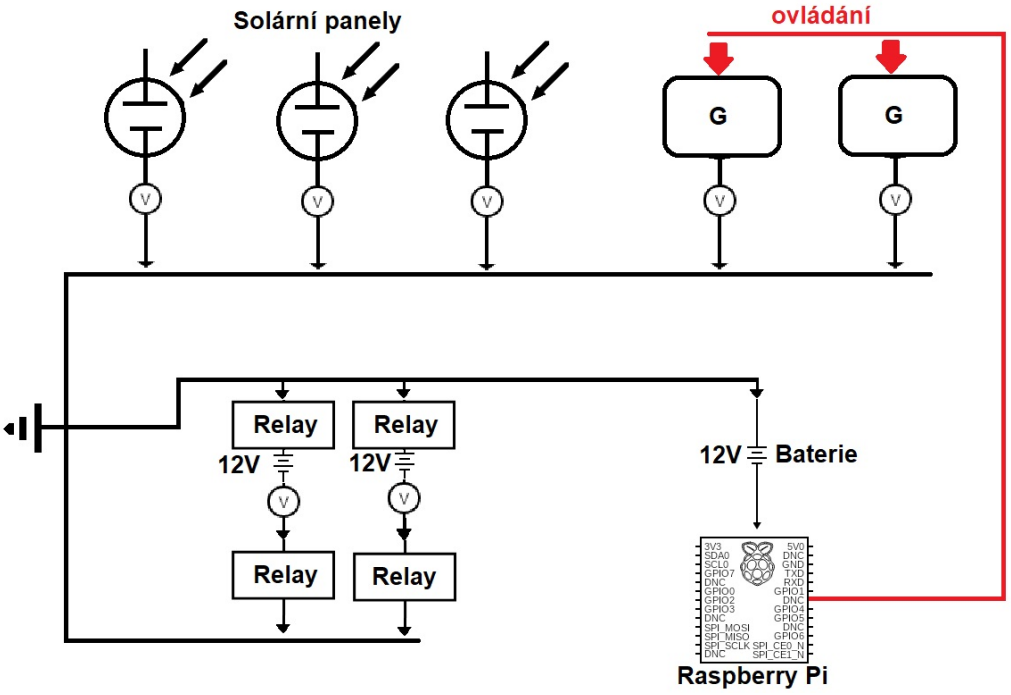
- Implementace API rozhraní zastává sbírku procedur, funkcí, tříd či protokolů určité knihovny, které může programátor využívat pro práci s daným systémem, či získání informací.

Předpokládá se tedy vytvoření modulárního programu s webovým rozhraním, jež bude získávat informace z co možná nejrozsáhlejších možností zařízení a tyto zobrazovat na web. Dále je zapotřebí vybrat programovací jazyk. Raspberry Pi podporuje velké množství programovacích

jazyků, jako Python, Asembly, C, Java, C++. Tedy možností, ze kterých lze vybrat ten správný prostředek pro tvorbu se naskýtá opravdu mnoho. O vytváření webových stránek platí to samé. Pro úspěšnou tvorbu zle využít například JS, HTML, či Python, který je v dnešní době zřejmě nejpoužívanějším programovacím jazykem na této platformě.

3 Návrh vlastního řešení hardwaru

V mém návrhu řešení pracuji se schématem 3 typů zdrojů, konkrétně s 12V olověným akumulátorem, solárním panelem a benzínový generátorem a to z mnoha důvodů. Použitím typu 12V olověného akumulátoru se úmyslně přidávám k velké komunitě jejich uživatelů a to z prostých důvodů. Nejprve je třeba zmínit cena takového akumulátoru, která je velmi přívětivá a je také jedním z důvodů, proč jsou tolik rozšířené. Díky široké škále typů produktů je možné vybrat si vždy ten ideální do potřebné situace a to samé platí pro součástky určené k práci s tímto akumulátorem. Mezi jeho další vlastnosti pak patří například vysoká spolehlivosti, odolnost, bezúdržbovost či schopnost dodávat velmi vysoký proud. Jelikož pracuji s možností zapojení nejen jako nabíječe Raspberry Pi, ale jako zdroj náhradního napájení, je pro mne důležité zjišťovat stav akumulátoru, čehož dosáhnu měřením napětí na jeho výstupu. Při řešení možnosti zapojení do systému generátory založené na obnovitelných zdrojích, jsem se rozhodl pro typ solární převážně z důvodu dostupnosti. Nevylučuji možnost, že by některé zařízení rozsáhlejších struktur mohly využívat například větrné elektrárny. Bohužel pro normálního uživatele je mnohem přijatelnější prakticky ze všech pohledů vybrat si raději solární typ nežli větrný. A konečně mou třetí volbou se stal generátor založený na spalování fosilních paliv, konkrétně benzínu. Jakožto nejméně ekologická z dostupných možností však nejvíce dominuje svou spolehlivostí a přesně ta je hlavním bodem mého systému. Stejně tak jako olověný akumulátor je u benzínového generátoru potřeba kontrolovat jeho stav k vyčerpání, čehož dosáhnu monitoringem jeho vlastností.



Obrázek 1: Můj návrh řešení zapojení jednotlivých zařízení

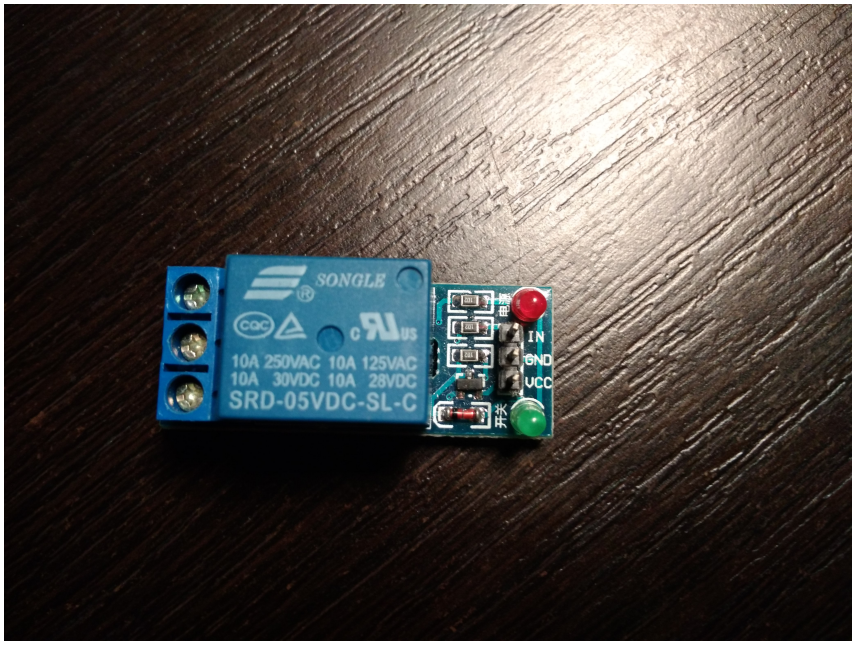
Z těchto zařízení jsou nejméně dvě potřebné pro minimálně základní funkcionalitu systému, který se alespoň v tomto bude velmi podobat systému zapojení, jež byl zmíněn dříve u systémů používaných nemocnicemi. Ve středu všeho bude zapojen mikropočítač Raspberry Pi napájený baterií. Jakožto alespoň jediný náhradní zdroj energie bude připojen k síti benzínový generátor. Všechny jakékoliv další zařízení z výše uvedených, které by chtěl uživatel zapojit navíc, budou sloužit pouze k zlepšení již plně funkčního systému. Nicméně každé takové další připojené zařízení bude systému dodávat delší výdrž a tedy také spolehlivost. Pokud budou zapojeny solární panely, mohlo by v ideálním případě při výpadku dodávky elektřiny dojít k situaci, kdy by tyto solární elektrárny plně zastoupily výpadek elektřiny svým vlastním provozem. Pokud budou navíc zapojeny další baterie, tak při výpadku bude první energie odebírána z nich. Až se tyto baterie vyčerpají, bude teprve zapnut generátor. V případě zapojení více generátorů, bude ve chvíli jejich potřeby zapojen vždy jen jeden a až bude plně vyčerpán, přejde se plynule na další. Všechny akce v mém návrhu systému je třeba ovládat něčím, s čím dokáže pracovat Raspberry Pi. Pro tuto funkci ovladače jsem vybral jednoduché [13]spojové relé, jež bude nadmíru vyhovovat těmto podmínkám a mezi jehož funkce bude patřit spínání pro start generátoru na fosilní paliva. Použitím relé se navíc maximalizovala škála typů a druhů zapojení, jelikož se jedná o jednoduchý spínač, který není omezen kompatibilitou, dokud je možnost jeho využití v elektrickém obvodu za pomocí součástek k tomu uzpůsobených. Díky tomuto typu ovládání může uživatel zapojit do obvodu i součástky s atypickým spínačem. Stačí dostat výstup či vstup zařízení na propojení v obvodu kabeláží, kde jej následně již ovládá relé. Tímto způsobem získává uživatel možnost použít prakticky jakékoliv zařízení, bez nutnosti používání vyhrazených řad produktů uzpůsobených pro kompatibilitu s Raspberry Pi. Tímto je zajištěna funkcionalita celého obvodu.

Tabulka 1: Tabulka hodnot napětí odpovídajících stavu nabití baterie

Nabití baterie	**Napětí na 12V baterii**	**Napětí na 24V baterii**
100%	12,70V	25,40V
95%	12,64V	25,25V
90%	12,58V	25,16V
85%	12,52V	25,04V
80%	12,46V	24,92V
75%	12,40V	24,80V
70%	12,36V	24,72V
65%	12,32V	24,64V
60%	12,28V	24,56V
55%	12,24V	24,48V
50%	12,20V	24,40V
45%	12,16V	24,32V
40%	12,12V	24,24V
35%	12,08V	24,16V
30%	12,04V	24,08V
25%	12,00V	24,00V
20%	11,98V	23,96V
15%	11,96V	23,92V
10%	11,94V	23,88V
5%	11,92V	23,84V
0%	11,90V	23,80V

```
import RPi.GPIO as GPIO
import time

def relayon(ktera):
   #funkce menici stav rele z vypnuteho na zapnute
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(ktera, GPIO.OUT)
    GPIO.output(ktera,GPIO.LOW)
    return
```

Výpis 1: Funkce způsobující změnu stavu relé z vypnutého na zapnuté



Obrázek 2: Vzhled ovládacího relé a jeho pinů

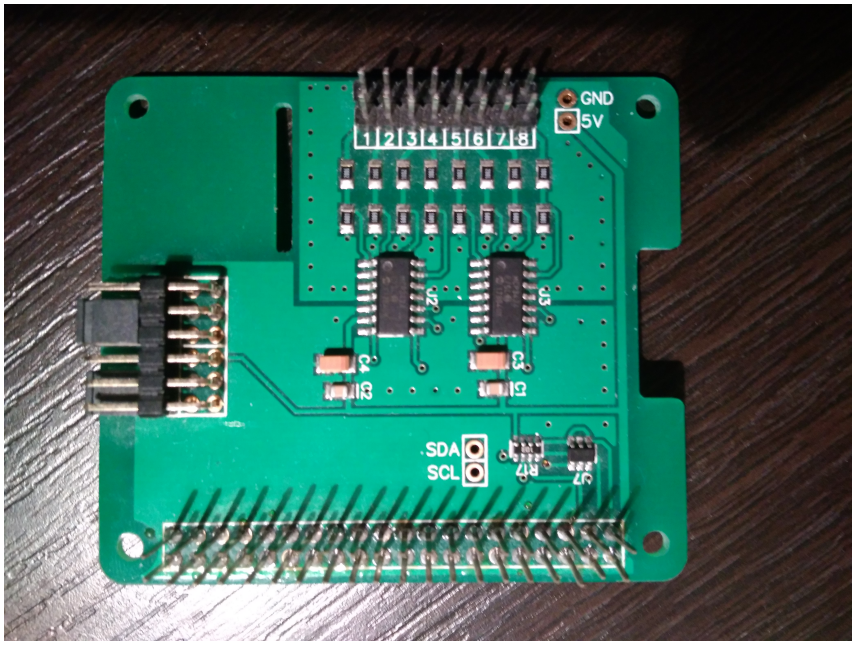
3.1 Hardware součástky pro monitoring

Monitoring v mém návrhu řešení hraje také velmi důležitou roli. Nejen že dodává uživateli všechny potřebné informace ohledně stavu systému, které mu dopomáhají v případě toho, že se rozhodne něco změnit sám. Navíc je velmi důležitý pro program samotný, jelikož většina z nich napomáhá ve správně a přesnější funkcionalitě hlavního řídícího úseku.

Jednu z hlavních rolí v získávání informací pro můj program zastupuje analogovo-digitální konvektor ADC Pi Plus Kit. [14]Tento 17-bitový board, jež je kompatibilní s Raspberry Pi se velmi lehko napojuje a zajišťuje možnost připojení až osmi vstupů, díky osmi kanálům. Princip jeho fungování tkví ve třech MCP3424 A/D převodnících, každý obsahuje 4 analogové vstupy. Díky kompatibilitě se zařízením Raspberry Pi tento produkt za přívětivou cenu nabízí ohromný komfort a jeho instalace a používání jsou nadmíru přívětivé. Díky tomuto zařízení se v mém řešení nabízí možnost měření napětí na libovolných úsecích sítě, což mi v systému zapojení dovoluje měřit aktuální stav baterií a jiných prvků.

```
from ADCPi import ADCPi

#funkce pro získání voltáže na pinu a
def measure(a):
   adc = ADCPi(0x68, 0x69, 12)
   return (adc.read_voltage(a))
```

Výpis 2: Funkce pracující s přídavným boardem ADC PI Plus Kit k získání napětí na úseku připojeného k odpovídajícímu pinu

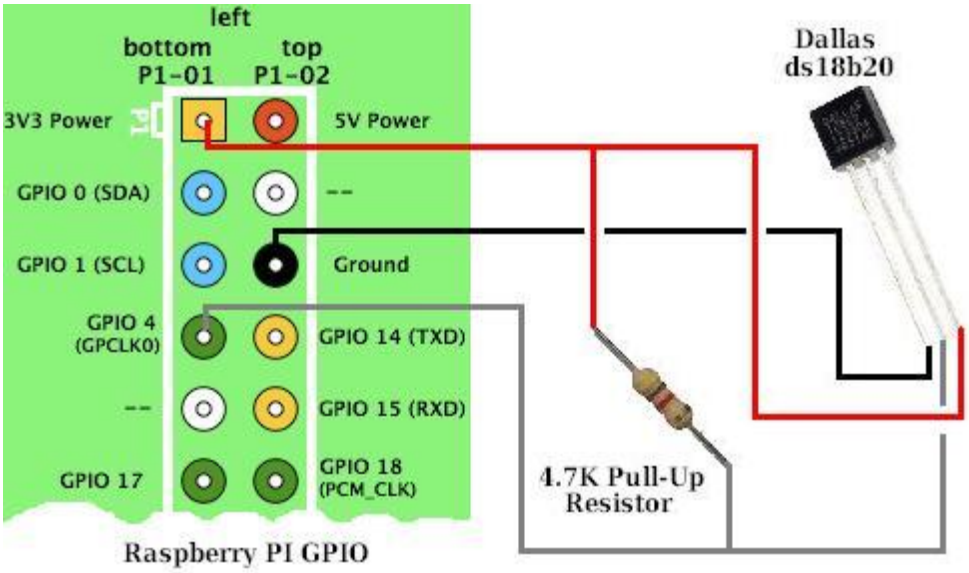


Obrázek 3: Boardu převodníku ADC

Pro důslednější sledování situace v okruhu zařízení jsem mezi součástky monitoringu přidal [8]Dallas 18B20 teploměr, což je nejpoužívanější druh a model pro tento typ zařízení. Díky tomuto zařízení je uživatel schopen zjistit teplotu v prostředí, kde se teploměr nachází. Primárním účelem tohoto teploměru je monitorovat oblast a ujišťovat uživatele, že se v dané blízkosti příliš enormně nemění teplota, což by mohlo značit například požár. [10]Pro úspěšné připojení tohoto teploměru k Raspberry Pi je však vyžadován ještě rezistor, který se musí připojit na třetí pin tohoto zařízení a přemostit přes druhý pin. Výstupní napětí na tomto pinu je totiž příliš silné, jelikož pin 1 Raspberry Pi je nastaven na 3,3 V, což by způsobilo přetížení pinu. Jakožto pouhý informační prvek je teploměr v mém řešení jeden z méně potřebných zařízení a obstarává také menší funkcionalitu. Pokud se uživatel rozhodne, že jej nebude potřebovat, funkcionalitu systému to nijak neovlivní.

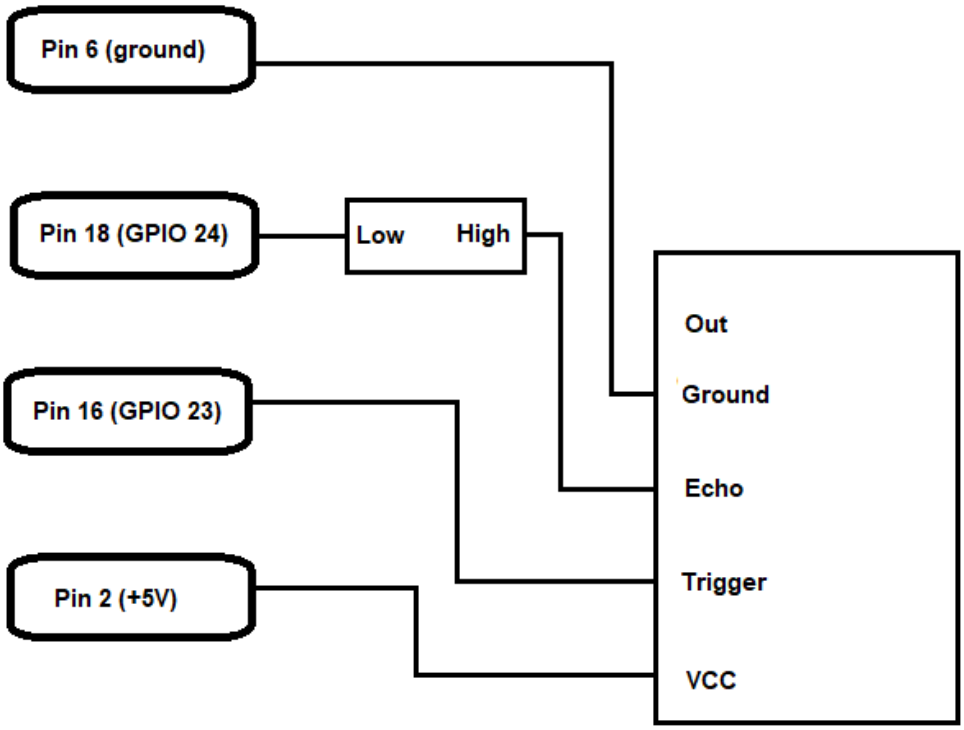
```
from w1thermsensor import W1ThermSensor

#Funkce pracující s knihovnou W1ThermSensor, která si vyhledá teploměr připojen
    k zařízení a získá jeho hodnotu
def teplota():

    sensor = W1ThermSensor()

    return sensor.get_temperature()
```

Výpis 3: Funkce pro práci s teploměrem

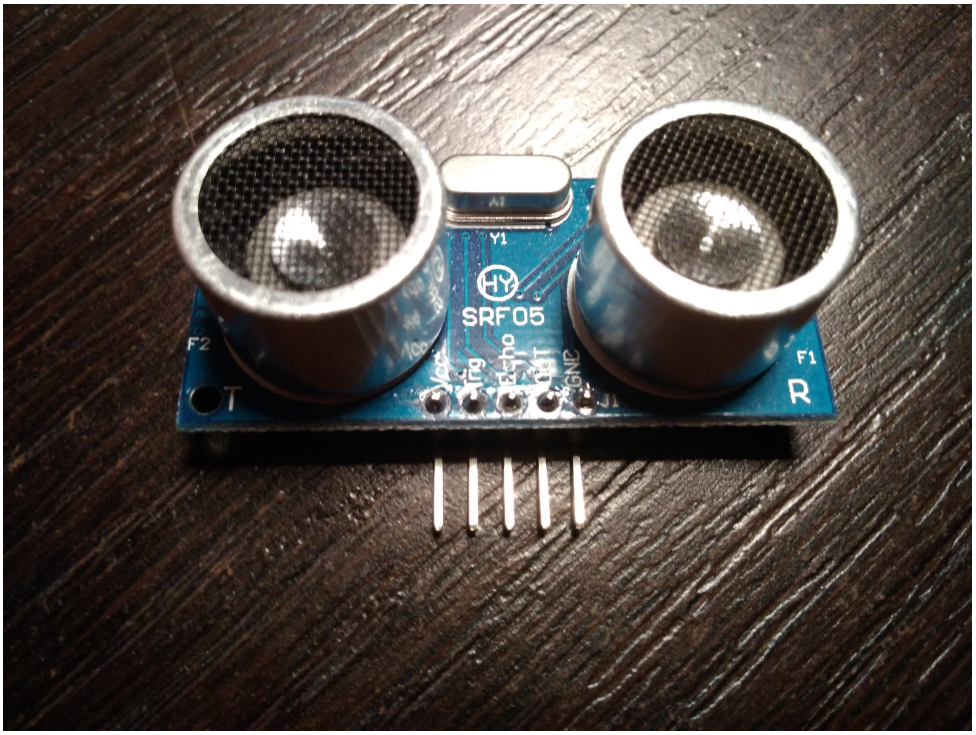


Obrázek 4: Zapojení teploměru společně s rezistorem na odpovídající piny Raspberry Pi (Zdroj: [9]`http://www.reuk.co.uk`)

Dalším zařízením pro sledování stavu, tentokráte hladiny kapaliny, je [11]HC-SR04 modul. Tento ultrazvukový měřící modul je uzpůsoben k měření vzdálenosti od protější plochy za pomocí ultrazvukových vln. V mém schématu modul plní funkci zjišťování obsahu nádrže generátoru. Tento přístroj byl vybrán ze dvou důvodů. Prvním důvodem byla bezpečnost zařízení, ve které ultrazvukový senzor výrazně dominuje v porovnání s jinými typy senzorů. Pokud ho porovnáme například s analogovými senzory, které je potřeba vložit do kapaliny a detekují hladinu pomocí průchodu napětí, i když samy o sobě nebezpečné nejsou, tak v případě zkratu by nebylo dobré je mít ponořené v benzínu. Oproti tomu ultrazvukový měřící modul nemá zapotřebí dotýkat se hladiny pro správnou funkcionalitu a tedy představuje bezpečnější možnost. Druhým důvodem se stal fakt, že je toto zařízení malé velikosti, tudíž by se pro přesné měření dalo vložit, samozřejmě až po zajištění voděodolnosti, přes většinou malý vstupní otvor do prostoru nádrže a přidělat na vrchní víko, kde se ho nebude kapalina dotýkat. Pokud tuto metodu uživatel nebude preferovat, může senzor uložit na víko, odkud měří stejně dobře. [12]Bohužel však díky vysokému výstupu na jednom z pinů, jež Raspberry Pi zachytává, je třeba výstup z jeho Echo portu provézt přes konvertor z +5V na 3,3V. K tomuto účelu v mém schématu poslouží jednoduchý, 8-pinový, obousměrný konvertor úrovní z 5V na 3,3V. Díky tomuto konvertoru se ultrazvukové měřícímu modulu podaří úspěšně komunikovat s mikropočítačem Raspberry Pi.



Obrázek 5: Příklad správného zapojení ultrazvukového modulu HC-SR04



Obrázek 6: Ultrazvukového modulu HC-SR04

```
import RPi.GPIO as GPIO
import time
#Funkce pro získání vzdálenosti objektu od senzoru na principu měření času
    putování vyslaného ultrazvukového signálu k objektu a zpět. Výsledná vzdá-
    lenost je následně podělena rychlostí a počtem cest signálu
def mervzdal(trig, echo):
    GPIO.setmode(GPIO.BCM)

   #Nastavení pinů Raspberry Pi
    GPIO.setup(trig, GPIO.OUT)
    GPIO.setup(echo, GPIO.IN)

    GPIO.output(trig, False)
    time.sleep(2)

    GPIO.output(trig, True)
    time.sleep(0.00001)
    GPIO.output(trig, False)

   #posílání signálu
    while GPIO.input(echo)==0:
        pulse_start = time.time()
    #přijímání
    while GPIO.input(echo)==1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start

   #vypočítání finální vzdálenosti dle vzorce v = s/t
    distance = pulse_duration * 17150
    distance = round(distance,2)
    GPIO.cleanup()
    return distance
```

Výpis 4: Funkce pracující s modulem HC-SR04 k získání vzdálenosti od protější plochy.

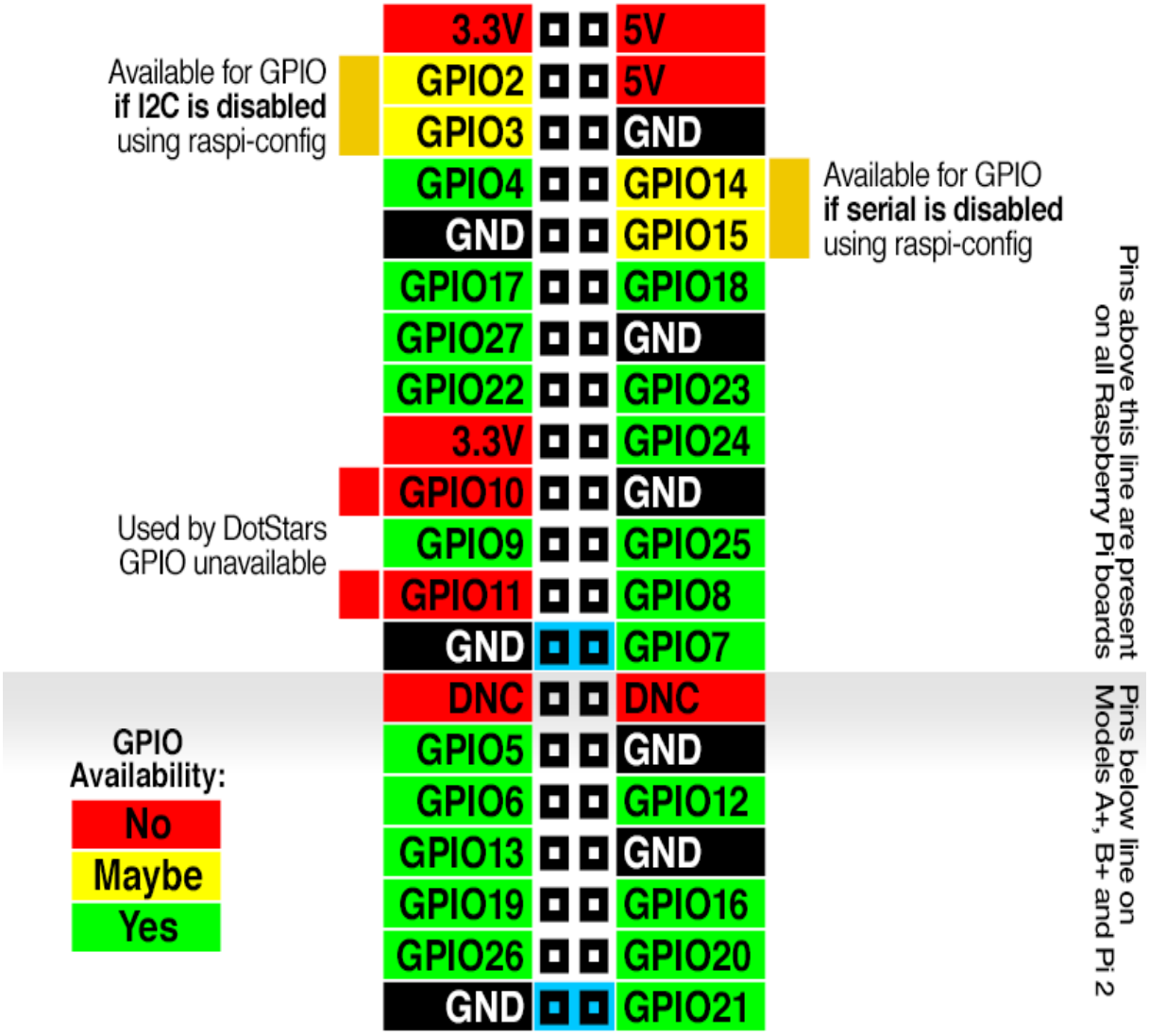
4 Návrh vlastního řešení softwaru

Mým cílem v této části bylo splnit všechny požadavky zadání Práce. Tento úkol obsahuje vytvoření programu, který by obstarával funkcionality všech připojených modulů na základě zapojeného schéma, za pomocí využití monitorovacích modulů pro získávání informací. Tento program má za úkol komunikovat s webovými stránkami, na kterých by mělo probíhat zobrazení jeho výstupu. Zároveň jsem se musel zabývat možností propojení systému s dalšími stejnými systémy běžícími na jiných zařízeních Raspberry Pi. Rozhodl jsem se tedy vytvořit dvě samostatné části s různými funkčnostmi, které by si mezi sebou posílaly informace a díky spolupráci by tvořily jeden velký celek. Pro tuto komunikaci mezi mikropočítači jsem se rozhodl vytvořit ještě jeden malý program, který by zastával část funkčnosti, kterou bylo potřeba sloučit pro zajištění správně funkcionality a přehlednosti.

Obrázek 7: Vzhled jednočipového počítače Raspberry Pi

4.1 Raspberry NOOBS - Raspbian

Pro práci se Raspberry Pi jsem si vybral operační systém [1]NOOBS založený na Raspbianu, který by měl začátečníkům v jeho používání, díky uživatelského rozhraní na rozdíl od Raspbian LITE, podat pomocnou ruku. Důvodem toho je fakt, že obsahuje spoustu programů a rozhraní pro práci a vývoj, které se při instalaci operačního systému nainstalují samy a které pro mne, byly velmi důležité. Konkrétně se jednalo o programovací nářadí pro programovací jazyk Python, jež jsem se rozhodl použít pro vývoj programů. Dalším hlavním rozdílem je ten, že Raspbian LITE nemá GUI, nýbrž má pouze CLI a tedy práce v NOOBS pro mne byla mnohem přívětivější. Důležité je také zmínit, že Raspbian je založený na starší verzi [2]Debianu, tedy z větší části má vlastnosti Linuxu, což nám dodá na volnosti v manipulaci se systémem.



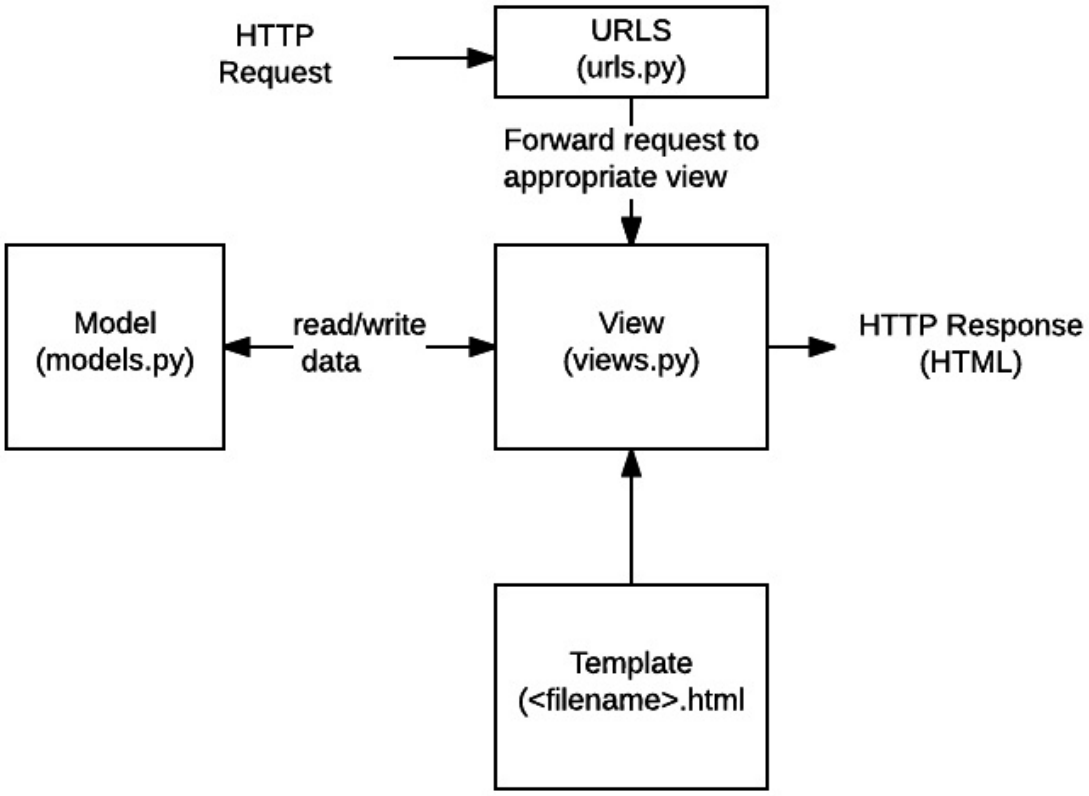
Obrázek 8: Schéma pinů na Paspberry Pi
(Zdroj: [6]`https://pythonprogramming.net`)

4.2 Hlavní program

Tématem hlavního programu bylo převážně řešení funkcionality systému a komunikace s jinými zařízeními. Tato část celku se měla zabývat jak jednotlivým nastavením všeho potřebného na samotném Raspberry Pi, jako například porty pro ovládání relé, tak i zajišťovat celkovou funkcionality a provádět téměř všechny potřebné operace. Pro vytvoření tohoto programu jsem si vybral programovací jazyk [3]Python, který jakožto jeden z nejpoužívanějších jazyků na platformě Raspberry Pi nabízí širokou škálu možností. Navíc pro mne jakožto nováčka, mezi všemi možnými nabízenými variantami, byl jednou z nejlepších možností nejen díky svému srozumitelnému prostředí a syntaxi, ale také například kvůli množství knihoven vytvářených jak vývojáři, tak komunitou. Vyznačuje se také svou velkou uživatelskou komunitou, která je v případě problémů velmi nápomocná.

4.3 Webové stránky

Pro tvorbu webových stránek jsem se rozhodl jít směrem web frameworku [4]Django, který nabízí skvělé propojení programovacího jazyku Python a webového rozhraní. Samotný Django framework patří mezi jedny z nejpoužívanějších programů pro takovéto úkoly. Díky rozšířené uživatelské komunitě existují spousty užitečných přídavků a rozšíření, které se právě pro takovouto práci hodí. Django spojuje programovací jazyk Python s jazykem HTML, CSC a je-li zapotřebí, tak i JS v k tomu určených částech, které společně tvoří jeden výstup. Díky možnosti kombinace všech těchto různých činitelů je uživateli prakticky garantován kvalitní výsledek, pokud se naučí s jeho systémem pracovat. Nicméně pro efektivní vývoj za využití tohoto programu je potřeba získání širokého spektra znalostí a vývoj v něm není jedním z jednoduchých, jelikož jak již zmíněno výše, je zapotřebí propojovat vícero částí do funkčního celku. Program funguje na spojení HTML stránek založených na bázi HTML jazyka a CSS, které uvidí uživatelé jako výstupy na webových stránkách a jejichž funkčnost je zajištěna kódem psaným v jazyce Python, který se nachází v pozadí. Toto propojení je až magické a dovoluje těm skutečně zasvěceným jedincům vytvářet nadmíru složité práce.



Obrázek 9: Princip fungování schématu Django systému
(Zdroj: [7]`https://developer.mozilla.org`)

5 Řešení - implementace

5.1 Hlavní program - Python

Můj vytvořený hlavní program pracuje na principu jednoho samostatně fungujícího programu, který se stará prakticky o všechno, pokud vynecháme zobrazování informací pro uživatele, jeho možnost zásahu do průběhu a přijímání zpráv od slave zařízení. Funkcionalita programu staví na získání schématu zapojení od uživatele díky využití konfiguračního souboru s informacemi, které musí uživatel nastavit před zavedením do provozu. Do tohoto souboru s názvem conf.txt se od uživatele předpokládá zadání informací založených na jeho zapojeném schématu, jako je například počet zapojených zařízení, nebo zdali zapojil teploměr. Po spuštění si program rozebere tento konfigurační soubor a dle získaných hodnot nastaví zbytek své funkčnosti na daném zařízení Raspberry Pi. Program při výkonu své funkce neustále v určitých intervalech kontroluje připojená zařízení a jejich hodnoty, dle kterých se rozhodne o vykonávání určitých funkcí pro zajištění dodávky elektrického proudu do sítě. Všechny získané informace navíc posílá části s webovou stránkou, která na oplátku posílá zpět informaci o případné ruční změně uživatelem v systému. Takovouto změnu hlavní část programu vždy provede, i kdyby měla znamenat snížení jeho efektivity.

```
Master
Name zarizeni: Sklep
Generator
        Relay_PIN: 17
                Name: Strecha
                        Hladina_PIN_TRIG:15
                        Hladina_PIN_ECHO:16
                        Hloubka Nadrze:58
                                Meric Napeti_PIN:1
Generator
        Relay_PIN: 18
                Name: Sklep
                        Hladina_PIN_TRIG:21
                        Hladina_PIN_ECHO:22
                        Hloubka Nadrze:158
                                Meric Napeti_PIN: 2
Solarni Panel
        Name: Strecha_1
                Meric Napeti_PIN: 4

Baterie
        Relay_PIN_nap:27
                Relay_PIN_pos:28
                        Name: Hlavni
                                Meric Napeti_PIN: 7
Teplomer
Sit napeti_PIN: 35
Cas = 15
```

Obrázek 10: Ukázka správně nastaveného konfiguračního souboru

```
#načtení obsahu souboru
file = open('conf.txt', 'r')
soubor = file.read()
file.close()

#úprava dat pro snadnější manipulaci
a = soubor.lower().split('\n',)

#Třídění informací získaných ze souboru a uložení do příslušných proměnných pro
     brzké použití
for x in a:
    if "master" in x:
        position = True
    elif "slave" in x:
        position = False

    if "name zarizeni:" in x:
        b = x.split("name zarizeni:")[1]
        while b.endswith(" "):
            b = b[:-1]
        while b.startswith(" "):
            b = b[1:]
        jmeno = b

    if "generator" in x:
        previous = 1
    if "solarni panel" in x:
        previous = 2
    if "baterie" in x:
        previous = 3
```

Výpis 5: Ukázka přečtení konfiguračního souboru a patřičného roztřídění informací

V hlavním programu se obstarává zjišťování hodnot z připojených monitorovacích přístrojů. Tyto hodnoty jsou společně s dalšími důležitými informacemi ukládány do souboru, se kterým pracuje Django. Program si ve svém hlavním průběhu hlídá napětí na jednotlivých částech obvodu dle schématu. Díky tomuto je schopen zjišťovat stav sítě a následně vhodně upravit jeho fungování za pomocí relé.

```
#otevreni kontrolniho souboru a zjisteni, zdali bude potreba neco menit
  for x in open("/home/pi/Projekt/Projekt/parameters/zmena.txt").
      readlines():
      if "zmen:" in x:
          b = x.split("zmen:")[1]
          while b.endswith(" "):
               b = b[:-1]
          while b.startswith(" "):
              b = b[1:]
          zmena = b
  #pokud nastala změna, tak ji provedeme
  if "m" in zmena:
      zmena = zmena.split("m")[1]
      zmena = int(zmena)
      if zmena < len(gen_rel) or zmena == len(gen_rel):
          if gen_stav[zmena-1] == False:
              RelayOn.relayon(gen_rel[zmena - 1])
              gen_stav[zmena-1] = not gen_stav[zmena-1]
          else:
              RelayOff.relayoff(gen_rel[zmena - 1])
              gen_stav[zmena - 1] = not gen_stav[zmena - 1]
```

Výpis 6: Ukázka kódu kontroly stavu sítě a následné úpravy v případě vstupu uživatele.

Jelikož jsem tomuto programu implementoval možnost rozšíření o master-slave vztah, který nastavuje uživatel v konfiguračním souboru před spuštěním, umí program komunikovat po lokální sítí se svým protějškem. Oba účastníci neustále naslouchají provozu a posílají si zprávy v dle nastaveného časového intervalu pro změnu. V takovémto případě slave sám netvoří žádný výstup do webového rozhraní. Místo toho jej posílá masterovi, který ho zobrazuje za něj. Tímto způsobem se na stránce mastera dozvíme nejen informace o jeho systému, ale i o systému jeho slave zařízení. Všechny z nich je možné upravovat

```
import socket
#Funkce posílá zprávu v encriptované formě na ip adresu druhého zařízení.
def sendmes(udp_ip, message):
    port = 5005
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.sendto(str.encode(message), (udp_ip, port))
    return)
```

Výpis 7: Posílání zprávy druhému zařízení za pomocí UDP paketů.

V případě, že je program spuštěn se zadanou funkcí slave, musí znát pro možnost posílání výstupních zpráv IP svého mastera. Proto se do konfiguračního souboru v takovýchto případech připisuje daná IP adresa.

```
import socket
import subprocess

proc = subprocess.Popen(["hostname -I", "/etc/services"], stdout=subprocess.
    PIPE, shell=True) # Program získá výsledek linux příkazu pro získání IP
(out,err) = proc.communicate()

UDP_IP = out  # Tuto poté používá k přijímání
UDP_PORT = 5005

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

sock.bind((UDP_IP, UDP_PORT))

while True:
   data, addr = sock.recvfrom(1024)

   print "recieved message:", data

   file = open("devi.txt","w")
   file.write(data)
   file.close()
```

Výpis 8: Program pro přijímání packetů UDP a uložení nesoucí zprávy

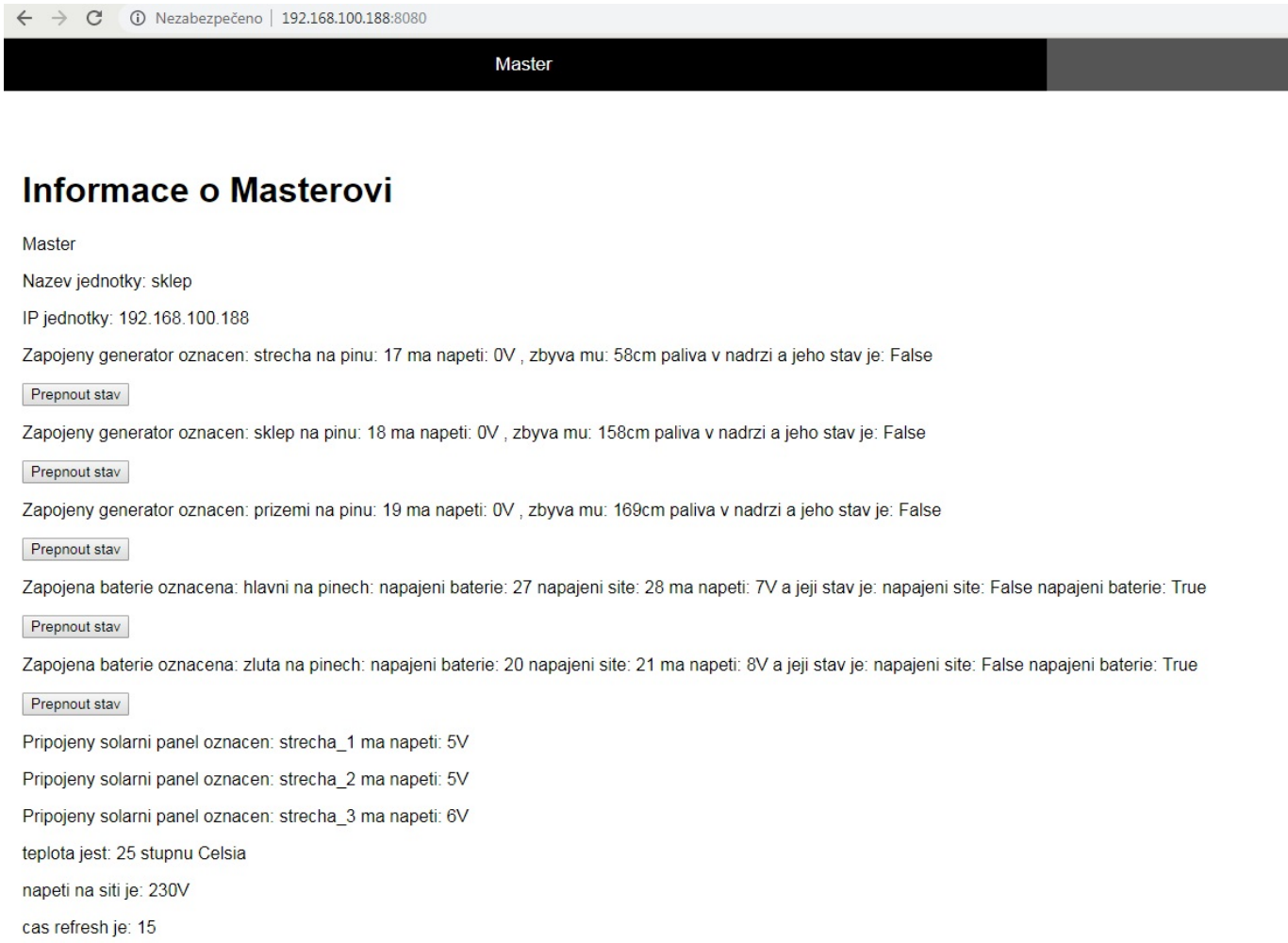
5.2 Webové stránky - Django

[15]Po spuštění aktivních webových stránek na lokální síti dle uživatelského návodu se na IP adrese jednotky master spojené se zadaným portem rozjede server dostupný z lokální sítě. Pouze jednotky s hodností master jsou uzpůsobeny k zobrazováni průběhu na webových stránkách a to jak systému svého, tak k němu připojených slave zařízení. Hlavní funkcionalitu těchto stránek obstarává více souborů, které dohromady utváří jeden výstupní celek ve formě webových stránek, které se zobrazí po připojení uživatele na uvedenou adresu. Hlavní grafickou podobu stránek obstarávají části HTML, které mají zakódovanou funkčnost v pozadí díky využití Python kódu. Program zodpovědný za vyvolávání HTML stránek se odkáže v tomto Python kódu na použitý odkaz a použije jemu příslušnou aktivitu kódu pro vyvolání stránky s danou funkcionalitou.

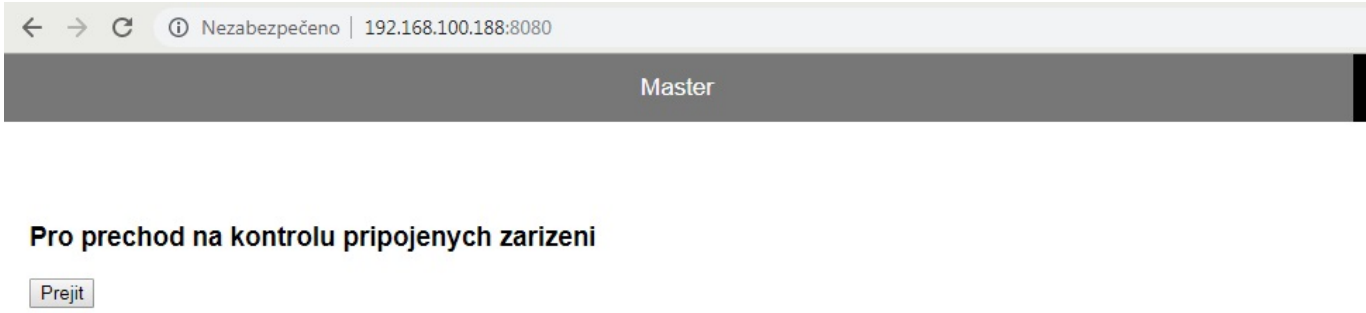
```
#!/usr/bin/env python
import os
import sys

if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "Projekt.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        # The above import may fail for some other reason. Ensure that the
        # issue is really that Django is missing to avoid masking other
        # exceptions on Python 2.
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

Výpis 9: Program pro spuštění Django website

5.3 Webové stránky - finální vzhled



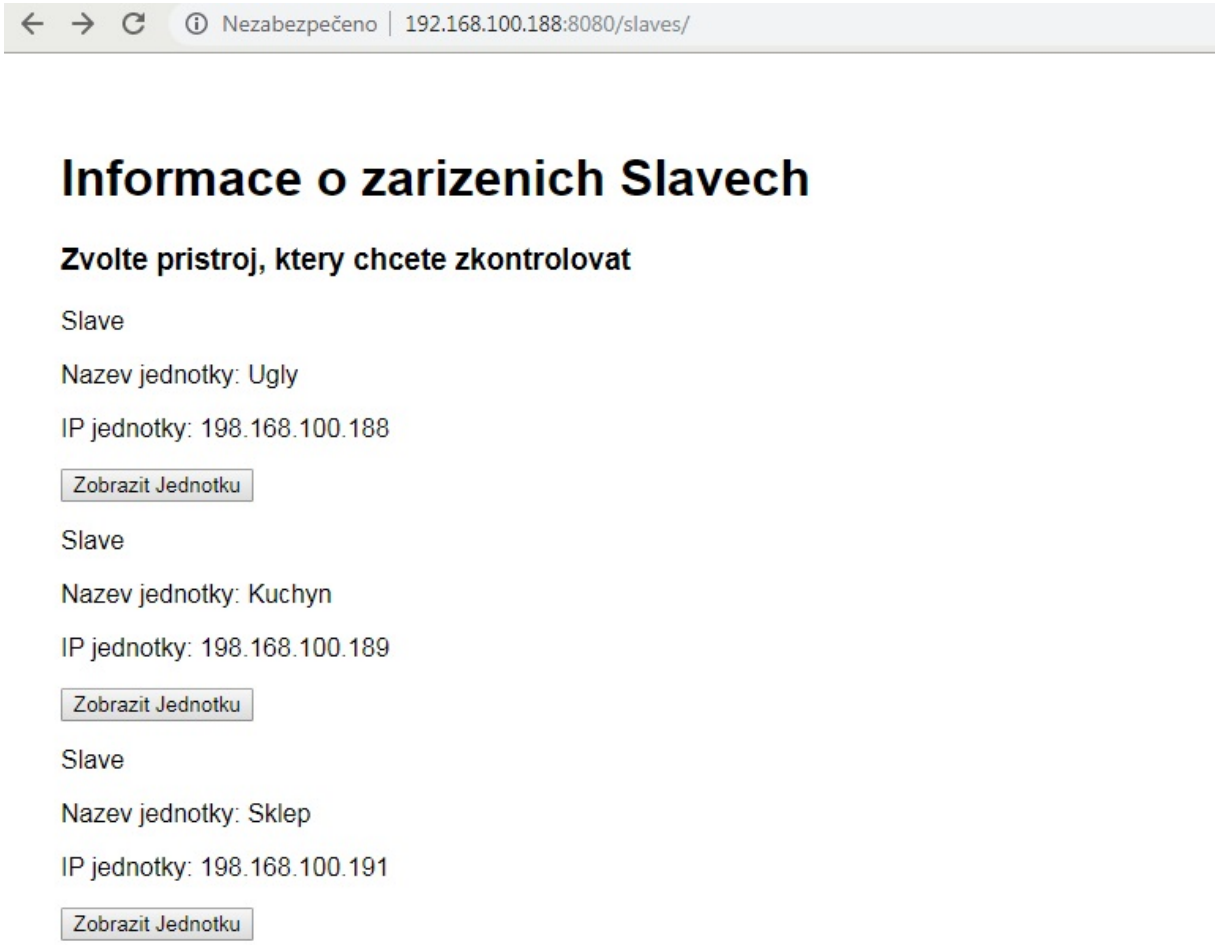
Obrázek 11: Vzhled hlavní strany webových stránek programu na záložce Master
Na této domovské stránce jsou vypsány získané informace o schématu v provozu společně s jeho ovládáním. Při přepnutí na záložku Slaves dostane uživatel jiné možnosti.



Obrázek 12: Vzhled hlavní strany webových stránek programu na záložce Slaves
Zde se uživateli nabízí možnost navigace k výběru výpisu informací od Slave zařízení.

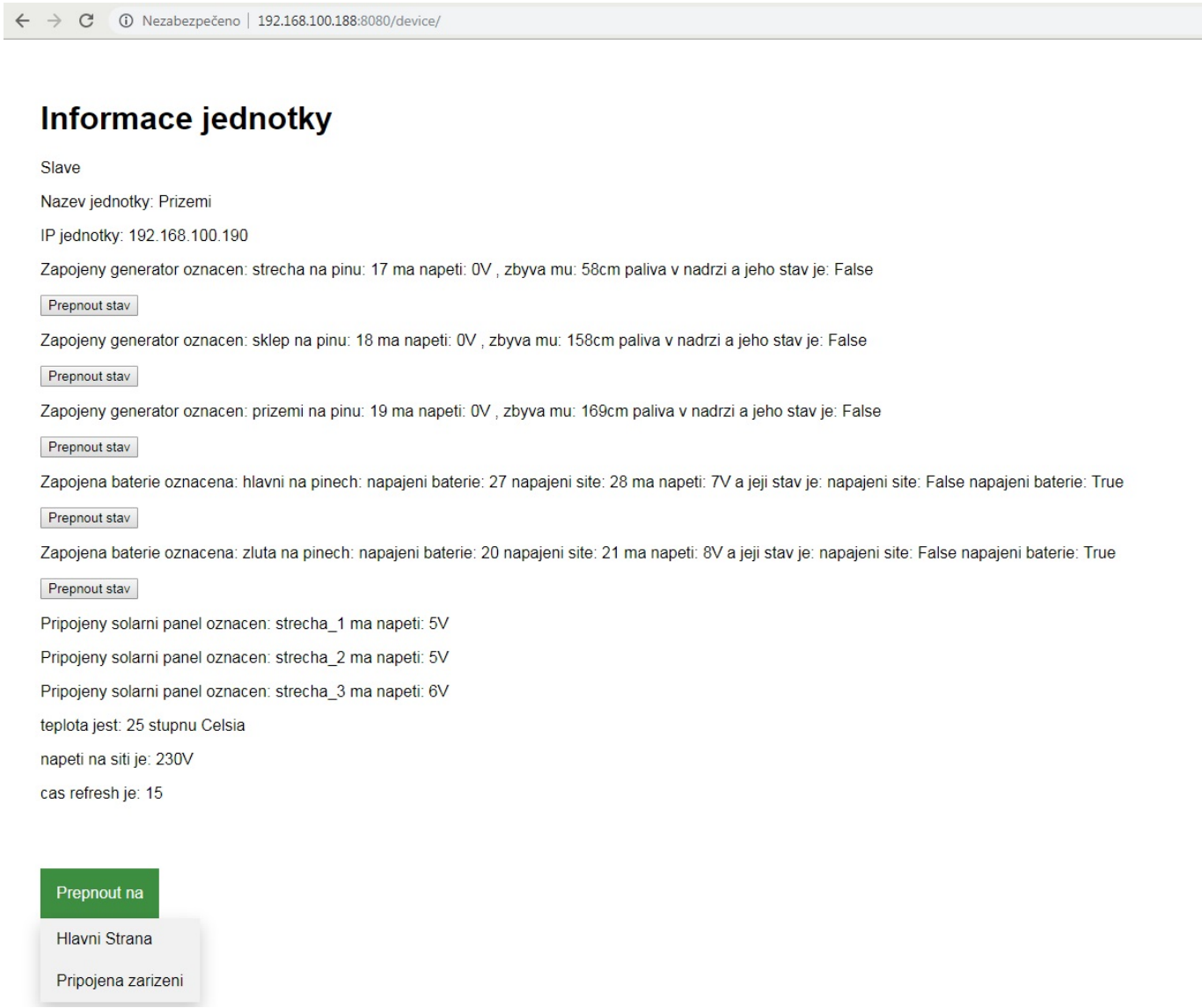
5.4 Testování

Finální produkt, tedy hotový projekt systému pro správu záložních zdrojů napájení nebyl bohužel kvůli časovým obtížím nikdy vyzkoušen v praxi za plného provozu s alespoň průměrným potenciálem. Nicméně byly odzkoušeny jednotlivé části jak programu, tak téměř všech použitých zařízení ve schématu. Při těchto testech byly nastaveny a přizpůsobeny podmínky, které měly simulovat plynulý běh zařízení, které připojeny nebyly, pro ozkoušení funkčnosti s částmi, které připojeny byly. V těchto testech systém obstál a prokázal tedy svou funkčnost, která ať již plnohodnotně nevyzkoušená, by za plného nasazení fungovala dle požadavků a předpokladů.



Obrázek 13: Webové stránky - rozcestník

Po zvolení možnosti k přechodu z hlavní stránky je uživatel převeden na tuto podstránku, jejíž obsah je dynamicky tvořen a umožňuje zvolení zařízení, na které se chce uživatel podívat.



Obrázek 14: Webové stránky - připojena zařízení

Podoba stránky s informacemi Slave zařízení dle předešlého výběru uživatele. Opět obsahuje možnost ručního ovládání. Pokud se bude chtít uživatel dostat zpět na některou z původních stránek, může k tomu použít navigační tlačítko.

6 Další možná rozšíření

Vytvořený systém pro správu záložních zdrojů napájení dle mého názoru obsahuje všechny důležité prvky, jež by takovýto systém obsahovat měl a možná ještě něco navíc. Pokud se zamyslím nad možnostmi dalšího rozšíření, napadá mne možnost úplného předělání systému funkcionality a to na způsob, kdy by každý jednotlivý prvek tohoto systému mohl být nastaven od základu uživatelem do podrobných detailů, jak jeho samotné funkcionality, tak přínosu do systému. K takovému rozšíření by byla potřeba změna většiny výrobních nastavení zapojeného zařízení uživatelem, následována přesným nastavením využití v programu. Tyto změny by však dle mého názoru byli již v naprosté většině případů přehnané a nepotřebné.

Pokud se na možnosti rozšíření budu dívat z pohledu možností spojení s jinými možnými systémy, jsem toho názoru, že by se nejspíše dalo vytvořit projekt, který by mohly najít v dnešním světě využití a třeba si i získal popularitu. Jakožto takové uvedu příklady.

- Program, který by využíval předpovědi počasí k získání energie díky solárním panelům nebo větrným elektrárnám a z dostatečné energie, kterou by mohl ukládat například v bateriích, by spouštěl domácí spotřebiče jako např. pračku či sušičku. Tyto spotřebiče by se tedy prakticky provozovaly zdarma. Dále pak orientovat přesunutelné procesy a tedy i část spotřeby energie z denního režimu na režim noční, kdy cena elektřiny klesá. Například ohřev vody v bojleru, zapnutí myčky na nádobí, či spuštění tichého robotického vysavače.
- Program běžící na Raspberry Pi, jež by po zaznamenání signálu z dálkového ovladače podobného klíčům od auta provedl akce jako: otevření dveří od garáže auta, zapnutí domovních světel, zapnutí topení, zatáhnutí žaluzií a případně ovládal další inteligentní části domu.

7 Závěr

Zadáním této bakalářské práce bylo vytvořit systém pro správu záložních zdrojů napájení, které by se staraly o přívod energie do sítě v případě vypojení hlavní dodávky elektřiny. Cílem této práce bylo seznámení s jednodeskovým mikropočítačem Raspberry Pi, jeho možnostmi, rozšířeními, moduly a kompatibilními přístroji. Jakožto finální produkt této práce byl simulován a vytvořen jeden z případných možných schémat. Používaná technologie ani způsoby jejího užívání nebyly novým přínosem. Nicméně vytvořený výsledný produkt-systém, byl mnou sestaven dle mých vlastních postupů a metod, které ho dovedly až ke konečnému cíli.

Literatura

[1] UPTON, Eben; HALFACREE, Gareth. *Raspberry Pi user guide*, 2nd Edition, Chichester, West Sussex, UK: John Wiley and Sons Ltd, c2014, 298 p. ISBN 978-1-118-79546-0.

[2] HERTZOG, Raphaël; MAS, Roland. *The Debian Administrator's Handbook*: Debian Squeeze from Discovery to Mastery. Free Software Foundation, 2012. ISBN 979-10-91414-01-2.

[3] BRADBURY, Alex; EVERARD, Ben; WINDER, Russe. *Learning Python with Raspberry Pi*. John Wiley & Sons, 2014. ISBN 978-1118717059.

[4] VINCENT S., William. *Django for Beginners: Build websites with Python and Django*. 2.2. Amazon Media EU, 2018, 342p. ISBN 978-1983172663.

[5] ROZENBLAT, Lazar. *Home Generator: Selecting, Sizing And Connecting: The Complete Guide*. 2nd edition. CreateSpace Independent Publishing Platform, 2015, 58p. ISBN 978-1507536643.

[6] Harrison, *GPIO (General Purpose Input Output) Pins - Raspberry Pi tutorial* [online][Citace: 5.4.2019] `https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Home_page`

[7] cybergenerx, *Django Tutorial Part 5: Creating our home page* [online]2019[Citace: 5.4.2019] `https://pythonprogramming.net/gpio-raspberry-pi-tutorials/`

[8] BERGMANS san, *Temperature Measurements* [online] 2018[Citace: 7.4.2019] `https://www.sbprojects.net/projects/raspberrypi/temperature.php`

[9] REUK, *DS18B20 Temperature Sensor with Raspberry Pi* [online][Citace: 3.4.2019] `http://www.reuk.co.uk/wordpress/raspberry-pi/ds18b20-temperature-sensor-with-raspberry-pi/`

[10] POUTER les, *DS18B20 Temperature Sensor With Python (Raspberry Pi)* [online]2017[Citace: 7.4.2019] `https://bigl.es/ds18b20-temperature-sensor-with-python-raspberry-pi/`

[11] Matt, *Ultrasonic Distance Measurement Using Python – Part 1* [online]2012[Citace: 9.4.2019] `https://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/`

[12] ModMyPi LTD, *HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi*[online]2014[Citace: 9.4.2019] `https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi`

[13] MakinThings, *Controlling Any Device Using a Raspberry Pi and a Relay Module* [online]2018[Citace: 12.4.2019] `https://www.instructables.com/id/Controlling-Any-Device-Using-a-Raspberry-Pi-and-a-/`

[14] Arnaud Boudou, *PiCheckVoltage* [online]2013[Citace: 11.4.2019] `https://github.com/aboudou/picheckvoltage`

[15] RPI ADMIN *Raspberry Pi Django Tutorial* [online]2019[Citace: 3.4.2019]`https://raspberrypituts.com/raspberry-pi-django-tutorial-2017/`