Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Možnosti využití Raspberry Pi 3 pro vzdálené řízení domácnosti

Bakalářská práce

Tibor Krejčí

Školitel:
Ing. Michal Šerý, Ph.D.

České Budějovice 2017

Krejčí, T.,2017: Možnosti využití Raspberry Pi 3 pro vzdálené řízení domácnosti. [Possibilities to use the Raspberry Pi 3 for remote control of home. Bc. Thesis, in Czech.] – 47 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace:

Tato práce se zabývá návrhem a realizací vlastní jednoduché inteligentní domácnosti založené na počítači Raspberry Pi 3 model B. Důraz u této práce je kladen na vzdálené řízení aplikace skrze webový server. Samotná aplikace je primárně zaměřena na řízení intenzity umělého i přirozeného světla pomocí stmívače a ovládání naklápění okenních žaluzií. Dalšími prvky jsou poté řízení teploty a řízení přívodu elektrického proudu do spotřebičů. Aplikace a webový server je vytvořen ve vývojovém prostředí Qt.

Annotation:

This work deals with the proposal and realization of own simple intelligent home based on Raspberry Pi 3 model B computer. Emphasis of this work is putted on the remote direction of the application through a web server. Application is primary specialized to the direction of intensity of artificial and natural light, using dimmer and control tilt of window jalousie. Another elements are the direction of temperature and control of electricity feed to the appliances. The application and web server is developed by Qt.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své [bakalářské – diplomové] práce, a to [v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných Přírodovědeckou fakultou] elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 18. dubna 2017 .....

Tibor Krejčí

Poděkování

V první řadě bych chtěl poděkovat Petru Bravancovi, za podporu v začátcích a za poskytnutí web serveru, bez kterého by výsledek zdaleka nevypadal tak, jak vypadá nyní. Dále bych chtěl poděkovat Ing. Miroslavu Stružinskému za pomoc s realizací. Nemohu opomenout pomoc vedoucího své práce Ing. Michala Šerého, Ph.D., který mi poskytl cenné rady a pomohl práci dokončit včas.

Obsah

Seznam použitých zkratek

IOT	Internet Of Things
HW	HardWare
ARM	Advanced RISC Machine
LED	Light Emiting Diode
OS	Operating System
SSR	Solid State Relay
LED	Light Emitting Diode
GPIO	General Purpose Input/Output
PWM	Pulse Width Modulation
DPS	Deska Plošných Spojů
GUI	Graphic User Interface
JSON	Java Script Object Notation
API	application performance interface
W1B	1 Wire Bus
I2C	Inter-Integrated Circuit

1 Úvod

Tématika inteligentní domácnosti je dnes velmi populární. Cena komponent potřebných pro realizaci je velmi příznivá. Inteligentní řízení využívají jak malé domácnosti, tak průmysl, jejichž primárním cílem je ušetřit, a to buď finance za zbytečně spotřebovanou energii, či za pracovní sílu, potřebnou pro zacházení s daným vybavením.

Díky popularitě myšlenky inteligentní domácnosti se touto problematikou zabývá velké množství firem, jmenovitě: Samsung [1], Huawei [2], Sony [3], Bigclown [4], Home assistant [5] a další. Toto množství jmenovaných výrobců s sebou nese problém unifikovaných, neustále se opakujících řešení (nejčastěji vzdálené ovládání svítidel či zásuvek). Z toho plyne částečná absence řešení vyrobených na míru zákazníkovi. Existují zde sice výjimky, ale ty se projevují výrazně vyšší cenou provedení. Přijít tedy v tomto odvětví s něčím novým je téměř nemožné, nicméně možnost vyvinout individuální, cenově dostupný systém ovládání inteligentní domácnosti může být praktické.

2 Rozbor a návrh systému

Před započetím návrhu inteligentní domácnosti je nezbytné zvolit základní koncepci a cíle navrhovaného systému. Základní schopností inteligentní domácnosti bude samostatně řídit a kontrolovat skalární fyzikální veličiny jako je teplota a světlo. Další schopností inteligentní domácnosti je například ovládání přívodu elektrického proudu do určitých komponent. Důležitým prvkem inteligentní domácnosti je její ovládání. V dnešní době, kdy valná většina lidí používá chytrý telefon, se nabízí možnost řídit inteligentní domácnost primárně pomocí tohoto zařízení.

2.1 Stanovení cílů práce

Jednotlivé kroky návrhu a realizace otevřené univerzální platformy, určené pro ovládání a řízení akčních členů inteligentní domácnosti jsou následující:

- stručné teoretické předpoklady z hlediska automatizace, pro řízení inteligentní domácnosti
- seznámení se s parametry počítače Raspberry Pi 3 model B a porovnání jeho výbavy s dalšími alternativami na trhu
- volba senzorů a akčních členů pro řízení a regulaci fyzikálních veličin, vyskytujících se v inteligentní domácnosti. Mezi hlavní zkoumané veličiny patří teplota a intenzita osvětlení
- zvolení vhodného softwarového vybavení počítače pro tvorbu vlastní aplikace počínaje volbou OS, vývojového prostředí a webového serveru. Tvorba aplikace a popis mechanismu komunikace mezi aplikací a webovým serverem
- potřebné úkony nutné pro instalaci a běh aplikace na čistém, nenaprogramovaném počítači

2.2 Řízení z hlediska automatizace

Před zahájením realizace je třeba zvážit možnosti řízení, zda je potřeba soustavu regulovat, či postačí prosté ovládání. Pokud je přidána zpětná vazba, lze řízení nazvat zpětnovazební regulací. U inteligentní domácnosti postačí dva druhy akčních členů. Akční členy dvoustavové a vícestavové. Tyto dva druhy akčních členů mohou vyžadovat odlišný přístup z hlediska ovládání, kdy dvoustavový akční člen lze ovládat prostým spínáním a vícestavový může vyžadovat speciální řídicí systém pro nastavení požadované hodnoty.

2.3 Regulace teploty

Teplotu v daném prostoru lze řídit několika způsoby. Řízení teploty často podléhá omezením již nainstalovaných zařízení. Akční členy, kterými lze ovlivnit teplotu jsou: topení, okno, klimatizace a ventilační systém. Pro řízení teploty v prostoru je vhodné systém doplnit zpětnou vazbou, teplotním čidlem s vhodným rozsahem. Dále je třeba rozhodnout, jakým druhem akčního členu bude nastavená teplota ovlivněna. Tedy zda postačí dvoustavový akční člen, či vícestavový. Posledním nezbytným parametrem je ekologická stránka. Pokud je uvažován prostor disponující okny, topením i klimatizací, je možné v rámci možností udržovat maximální přesnou nastavenou teplotu, ale za cenu neustálého běhu klimatizace i topení, kdy v extrémním případě mohou běžet tyto spotřebiče zároveň, což může být například v průmyslových kalibračních laboratořích žádoucí, nicméně v domácích prostorech maximálně nechtěný jev. Vyspělý řídicí systém by tedy měl mít možnost tyto parametry vhodně nastavit v závislosti na požadavcích zákazníka. Řídicí systém by také měl dbát na limity regulace. Často s danými komponenty nelze dosáhnout požadovaných hodnot. Pro správné dosažení požadovaných parametrů je potřeba také vhodně umístit teplotní čidlo. Bude-li uvažováno jedno teplotní čidlo pro jednu místnost, je vhodné toto čidlo umístit do místnosti tam, kde je třeba dosáhnout nastavených parametrů. Další možností je rozmístění více čidel po místnosti a při zpracování dat z naměřených hodnot, počítat průměr. V případě průměrování většího množství dat je důležité mít dostatek výpočetního výkonu řídicího systému a také mít dostatek vstupů na řídicím systému.

2.4 Regulace světla

Světlo v prostoru, z hlediska řízení, je nutné rozdělit do dvou kategorií, na světlo přirozené a světlo umělé:

- světlo přirozené: Pokud je v prostoru možné přijímat i světlo přirozené, lze intenzitu tohoto světla řídit, a to buď prostým spínáním, nebo sofistikovanější metodou, vícestavovou regulací. Pro tuto regulaci je potřeba mít na vstupu akční členy, tedy žaluzie a pohon, který jimi bude hýbat
- světlo umělé: Přirozené světlo není k dispozici po celých 24 hodin, a tak, pro získání stálé intenzity je zapotřebí i světlo umělé. Zdrojem umělého světla může být klasická žárovka, výbojka, LED svítilna a další. Pro zachování stálé intenzity je třeba mít k dispozici vícestavovou regulaci v podobě stmívače, který intenzitu reguluje

2.5 Řízení přívodu elektrického proudu do spotřebičů

Jedná se o dvoustavové řízení, které může být doplněno o zpětnou vazbu, ale může být provozováno i bez ní. S tímto druhem řízení lze řešit naprostou většinu případů ovládání prvků inteligentní domácnosti. Výhodou této regulace je, že ji lze aplikovat na valnou většinu komponent bez sebemenších úprav. Realizaci sice bude věnována kapitola, nicméně nyní je třeba problematiku lehce nastínit. Z hlediska realizace je důležité, jak často je třeba daný obvod spínat. Pakliže se realizace provádí pomocí stykače, je třeba dbát na maximální počet možných sepnutí při určitém elektrickém proudu. V případě, že by stykač měl životnost tisíc sepnutí, není vhodné jej aplikovat tam, kde ke spínání dochází například stokrát denně. V případě častých spínání elektrických obvodů je třeba zvolit spínání pomocí polovodičového prvku, typicky tranzistoru, triaku, tytristoru, SSR a dalších. Polovodiče mají jako spínače řádově vyšší životnost, nicméně mohou být dražší a realizace obvodů s těmito prvky bývá výrazně složitější.

3 Realizace

3.1 Volba řídicího počítače

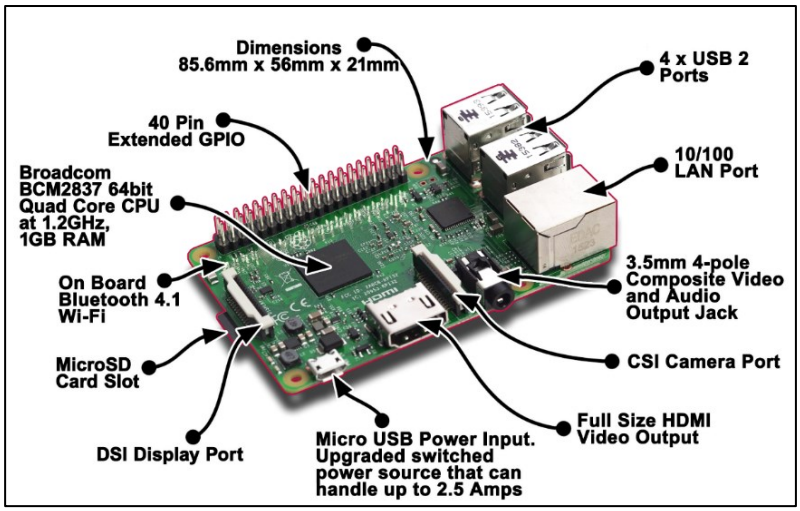
Volba řídicího počítače je stěžejní část vzdáleného řízení domácnosti. Cílem je mít výkonný multifunkční stroj, který může fungovat i jako server. Z hlediska ekologie je výhodné zvolit takový počítač, který minimálně spotřebovává elektrickou energii. Důležitým faktorem je také poměr cena/výkon/vybavenost. Z hlediska rozměrů je ideální mít počítač co nejmenší. Dalším důležitým faktorem výběru je počet sběrnic, portů, konektorů, bezdrátových protokolů, kterými počítač disponuje. Často opomíjený, ale důležitý faktor výběru, je také prostředí, ve kterém má počítač fungovat. Kritické faktory jsou: vlhkost, prašnost, teplota a další méně časté faktory (například radiace). Při použití počítače v extrémních podmínkách, je třeba výslednou realizaci těmto podmínkám přizpůsobit, například izolací onoho zařízení od okolního prostředí. V tomto případě je nutné zařízení dostatečně chladit. Při výběru řídicího počítače je třeba hledět i na podporu různých operačních systémů počítačem. Dnešní počítače disponují takovým výkonem, že nejen, že mohou běžet i jako server, ale je možné na tomto zařízení danou aplikaci lze i vyvíjet, což je velmi užitečné. Jsou-li při volbě vynechány počítače založené na architektuře x86/x64, tak se nejdostupnější architekturou stává ARM. Ten je při použití s OS Linux velmi multifunkční, to ale nelze tvrdit o všech OS. Microsoft IOT vyvíjí operační systém pro ARM počítače, nicméně zatím nenabízí samostatně stojící OS fungující jako server. Jako základ byl zvolen počítač Raspberry Pi3 model B.

3.1.1 Raspberry Pi3 model B

Raspberry Pi3 model B technické specifikace, viz Tabulka 1. Doplňující informace viz Obrázek 1. Budování inteligentní domácnosti se odvíjí od vstupně-výstupních, nízko úrovňových periferních zařízení (GPIO), na kterých jsou fyzicky přístupné sběrnice a komunikační protokoly, viz Tabulka 2 a Obrázek 2. Z obrázků a tabulek je patrné, že Raspberry Pi 3 nedisponuje analogovými vstupy a ani PWM výstupy. Pokud tedy je pro realizaci zapotřebí například PWM, musí být doplněno přídavným zařízením.

• Architektura	ARMv8 (64/32-bit)
• Procesor	Broadcom BCM2837, 1,2 GHz 64-bit quad-core ARM Cortex-A53
• Grafická karta	Broadcom VideoCore IV @ 250 MHz
• Operační paměť	1 GB (sdílené s GPU)
• USB	4x 2.0
• Video vstup	15-pin Mipi camera (CSI)
• Video výstupy	HDMI 1.3, kompozitní video (skrz 3.5 jack), display Mipi (DSI)
• Audio vstup	skrze I²S
• Audio výstup	analog přes 3,5 mm jack; digitální přes HDMI a jako revize 2 desky, I²S
• Úložný prostor	MicroSDHC slot
• Síť	10/100 Mbit / s Ethernet, 802.11n, Bluetooth 4.1
• Periferní nízkoúrovňové zařízení	40x GPIO
• Napájení	5V micro USB
• Rozměry	85.6mm x 56mm x 21mm
• Hmotnost	45 g

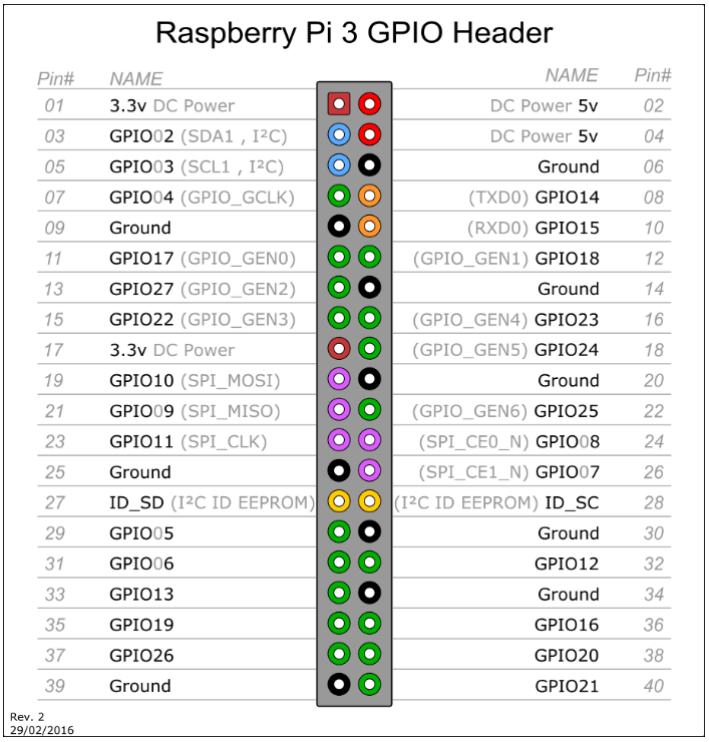
Tabulka 1, specifikace Raspberry Pi3 model B [6]



Obrázek 1, počítač Raspberry Pi3 model B [7]

• I2C	Inter-Integrated Circuit, GPIO 02, 03
• SPI	Serial Peripheral Interface, GPIO 07, 08, 09, 10, 11.
• UART	Universal Synchronous / Asynchronous Receiver and Transmitter, GPIO 14, 15
• W1B	1-Wire bus, GPIO 04

Tabulka 2, Sběrnice raspberry Pi 3 [6]



Obrázek 2, GPIO a dostupné sběrnice na Raspberry Pi3 model B [7]

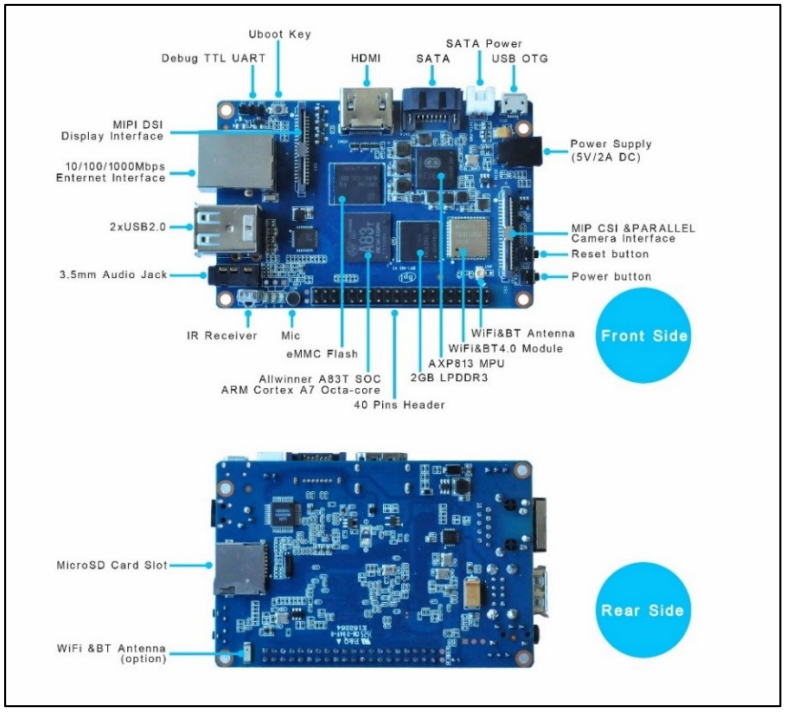
3.1.2 Alternativy k Raspberry Pi3 model B

Raspberry Pi3 model B je velmi výkonný, kompaktní, víceúčelový počítač, ale zdaleka není na trhu jediný. Existuje mnoho alternativ, některé jsou dokonce i výkonnější či lépe vybavené.

Brand	Raspberry Pi	Raspberry Pi	Raspberry Pi	Banana Pi	Banana Pi	Orange Pi	Orange Pi	Orange Pi	Orange Pi
Board	Zero	2 Model B	3 Model B	M2	M3	One	PC	Plus	Plus 2
SoC Vendor	Broadcom	Broadcom	Broadcom	Allwinner	Allwinner	Allwinner	Allwinner	Allwinner	Allwinner
SoC Chip	BCM2835	BCM2836	BCM2837	A31s	A83T	H3	H3	H3	H3
SoC Process	40nm	40nm	40nm	40nm	28nm	28nm	28nm	28nm	28nm
CPU Cores	1	4	4	4	8	4	4	4	4
CPU Design	ARM1176JZF-S	Cortex A7	Cortex A53	Cortex A7	Cortex A7	Cortex A7	Cortex A7	Cortex A7	Cortex A7
CPU Freq	1GHz	0.9GHz	1.2GHz	1GHz	1.8GHz	1.5GHz	1.5GHz	1.5GHz	1.5GHz
CPU Instruction	ARMv6	ARMv7	ARMv8	ARMv7	ARMv7	ARMv7	ARMv7	ARMv7	ARMv7
GPU Vendor	Broadcom	Broadcom	Broadcom	PowerVR	PowerVR	ARM	ARM	ARM	ARM
GPU Design	VideoCore IV	VideoCore IV	VideoCore IV	SGX544MP2	SGX544MP(1/2?)	Mali 400MP2	Mali 400MP2	Mali 400MP2	Mali 400MP2
GPU Freq	250MHz	250MHz	400MHz	355MHz	700MHz	600MHz	600MHz	600MHz	600MHz
H264 Dec	1080P30	1080P30	1080P30	1080P60	1080P60	1080P60	1080P60	1080P60	1080P60
H264 Enc	1080P30	1080P30	1080P30	1080P30	1080P60	1080P30	1080P30	1080P30	1080P30
H265 Dec	None	None	None	None	1080P30	4KP30	4KP30	4KP30	4KP30
Memory	512MB DDR2	1GB DDR2	1GB DDR2	1GB DDR3	2GB DDR3	512MB DDR3	1GB DDR3	1GB DDR3	2GB DDR3
Memory Freq	400MHz	400MHz	400MHz	432MHz	672MHz	672MHz	672MHz	672MHz	672MHz
Storage	MicroSD	MicroSD	MicroSD	MicroSD	MicroSD/USB SATA 2.0	MicroSD	MicroSD	MicroSD/USB SATA 2.	MicroSD/USB SATA 2.0
Storage Onboard	None	None	None	None	8GB eMMC	None	None	8GB eMMC	16GB eMMC
Storage	SD2.0	SD2.0	SD2.0	SD3.0	SD2.0/eMMC 4.4	SD3.0	SD3.0	SD3.0/eMMC 4.5	SD3.0/eMMC 4.5
USB 2.0	1 OTG	4	4	4 + 1 OTG	2 + 1 OTG	1 + 1 OTG	3 + 1 OTG	4 + 1 OTG	4 + 1 OTG
USB 3.0	0	0	0	0	0	0	0	0	0
Ethernet	None	100Mb	100Mb	1Gb RTL8211E	1Gb RTL8211E	100Mb	100Mb	1Gb	1Gb
Wireless	None	None	802.11N BCM43438	802.11N AP6181	802.11N AP6212	None	None	802.11N RTL8189ETV	802.11N RTL8189ETV
Bluetooth	None	None	Bluetooth 4.1	None	Bluetooth 4.0 AP6212	None	None	None	None
IrDA	None	None	None	Yes	Yes	None	Yes	Yes	Yes
HDMI	1200P60	1200P60	1200P60	1200P60	1200P60	4KP30	4KP30	4KP30	4KP30
RTC	No	No	No	No	No	No	No	No	No
Power Reg	5V1A	5V2A	5V2.5A	5V2A	5V2A	5V2A	5V2A	5V2A	5V2A
Power Plug uUSI	Yes	Yes	Yes	No	Some Boards	No	No	No	No
Power Plug Barre	No	No	No	4/1.7mm	4/1.7mm Some Boards	4/1.7mm	4/1.7mm	4/1.7mm	4/1.7mm
Android	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Linux Mainline	Yes	Yes	Yes	Yes	4.6	Yes	Yes	Yes	Yes

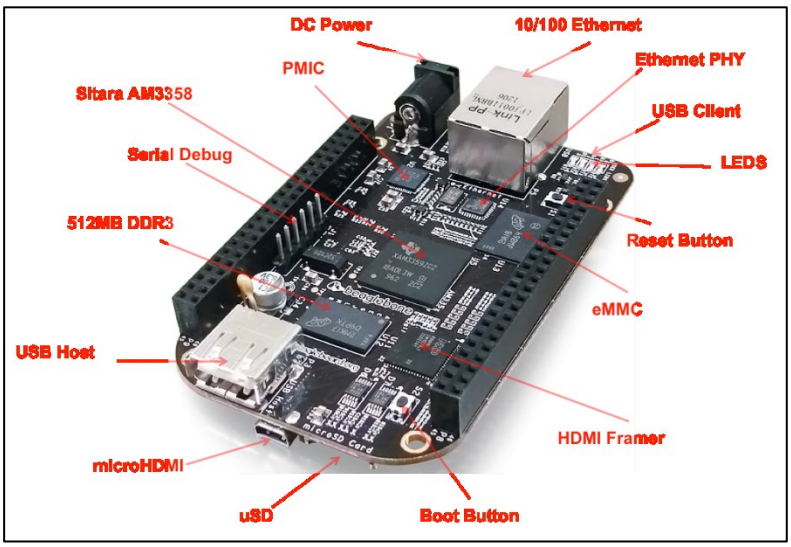
Obrázek 3, porovnání Raspberry Pi3 k alternativám [8]

Obrázek 3 znázorňuje porovnání Raspberry Pi 3 s adekvátními alternativami, kde vítězí BananaPi M3 viz Obrázek 4, které dokonce nabízí i SATA konektor, ale z finančního hlediska je oproti Raspberry Pi 3 o přibližně třetinu dražší.

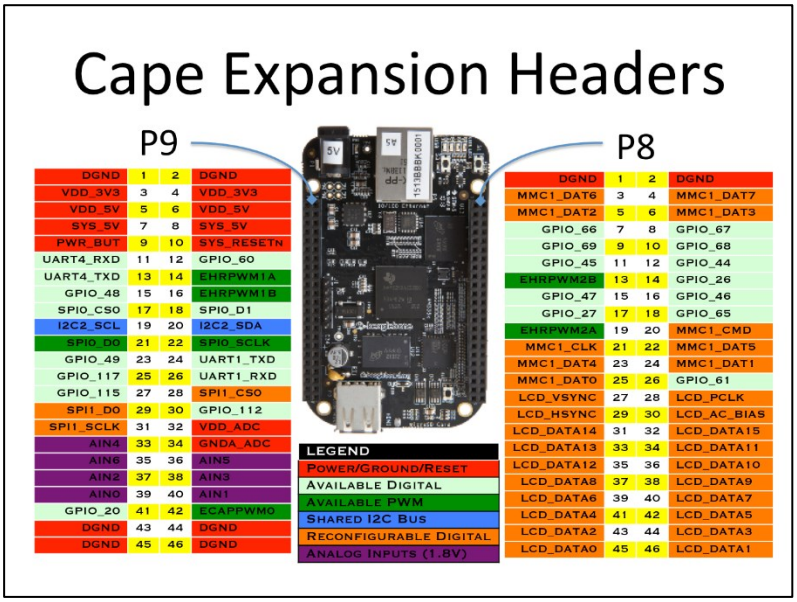


Obrázek 4, Banana Pi M3 [9]

Nelze opomenout ani Beaglebone black, viz Obrázek 5 a Obrázek 6, jehož předchůdce odstartoval rozvoj těchto počítačů [10], ale také disponuje větším množstvím vstupně-výstupních periferních zařízení, PWM kanálů a analogových vstupů, kterými Raspberry Pi nedisponuje. Beaglebone navíc nabízí v Linuxu hotové knihovny v C++ pro použití komunikačních protokolů, kde Raspberry Pi 3 zaostává. Více informací o knihovnách v kapitole se softwarem.



Obrázek 5, Beaglebone black [11]



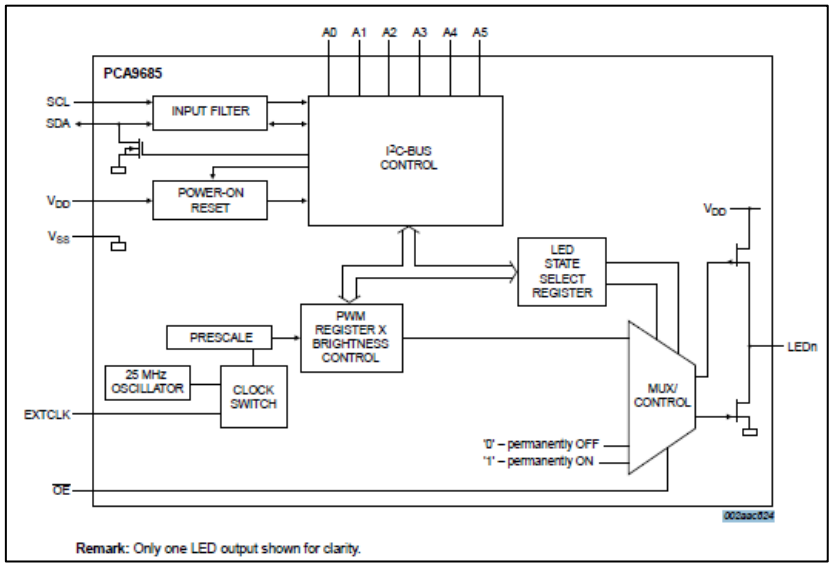
Obrázek 6, Beaglebone black pinout [11]

V omezené míře lze také použít i Arduino jako server [12], ale toto řešení není adekvátní v porovnání s Raspberry Pi. Dále je zde platforma Intel Joule [13], která disponuje velkým výpočetním výkonem, ale také přibližně čtyřnásobnou cenou.

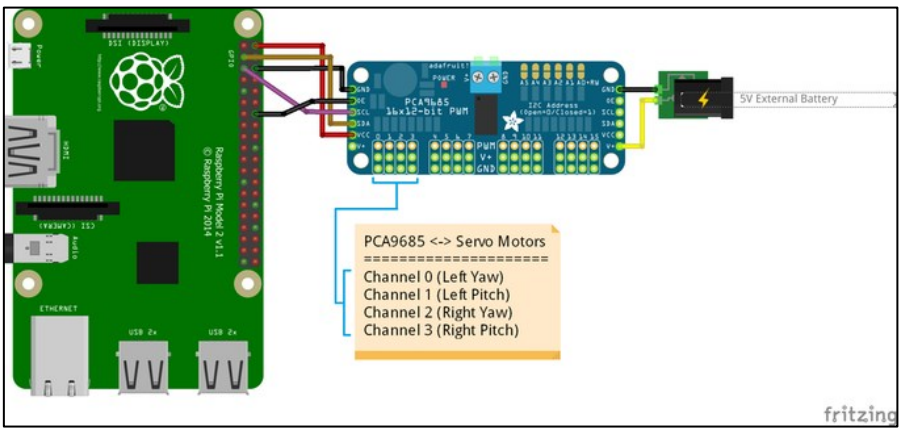
3.2 Volba periferií, senzoriky

3.2.1 PWM modul

Raspberry Pi3 model B není vybaveno moduly, které by uměly nativně vytvářet PWM, proto musí být v případě potřeby modul dodán externě. PCA9685 [14] je dvanáctibitový, šestnácti kanálový LED kontrolér, schopný pracovat s PWM v rozsahu frekvencí 24 Hz – 1526 Hz (reálný frekvenční rozsah se může lehce lišit). PWM modul může být použit jako základ pro stmívání světel, ovládání servomotorů, regulaci asynchronních motorů atp. Na Obrázek 7 je blokové schéma a na Obrázek 8 je znázorněné propojení modulu obsahujícího PCA9685 s Raspberry Pi skrze I2C sběrnici. K realizaci stačí připojit periferie řízené PWM a externí zdroj, například pro výkonné motory.



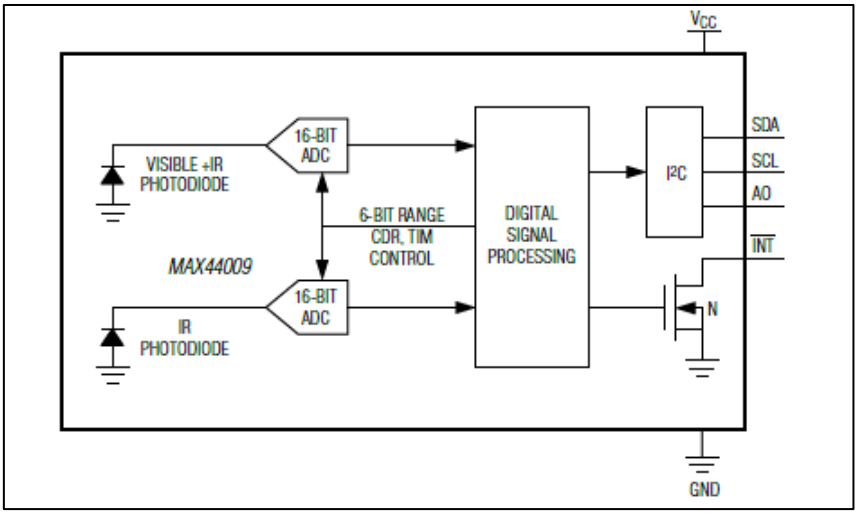
Obrázek 7, blokové schéma PCA9685 [15]



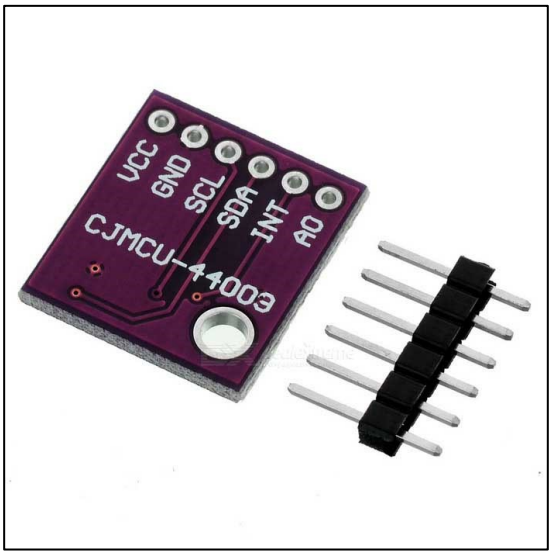
Obrázek 8, propojení Raspberry Pi 3 s PCA9685 [16]

3.2.2 Senzor intenzity osvětlení

MAX44009 nabízí snímání v širokém rozsahu (od 0,045 do 188000 Lux) [17]. MAX44009 je tedy vhodný pro snímání světla v prostoru. Blokové schéma znázorněno na Obrázek 9. Propojení s Raspberry Pi je realizováno pomocí I2C. DPS, na které se samotný chip nachází, lze zakoupit například pod názvem CJMCU-44009, viz Obrázek 10.



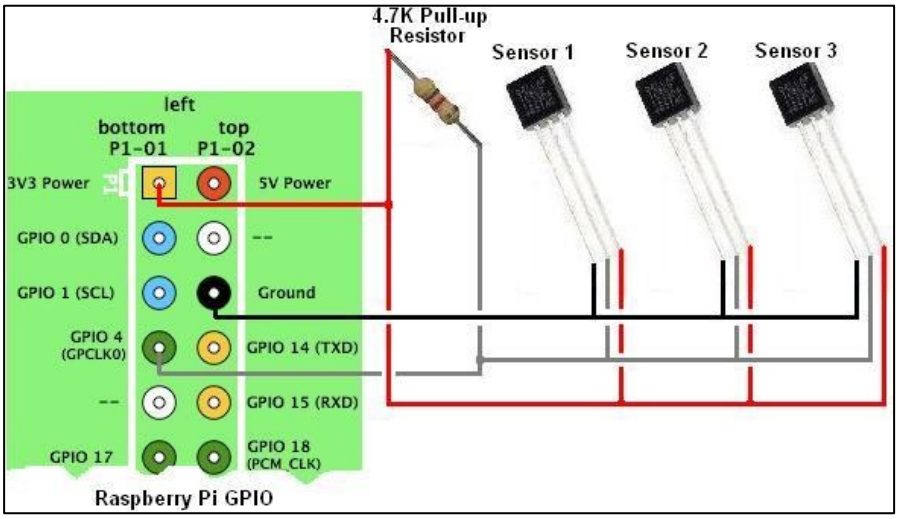
Obrázek 9, blokové schéma MAX44009 [17]



Obrázek 10, CJMCU-44009

3.2.3 Teploměr DS18B20

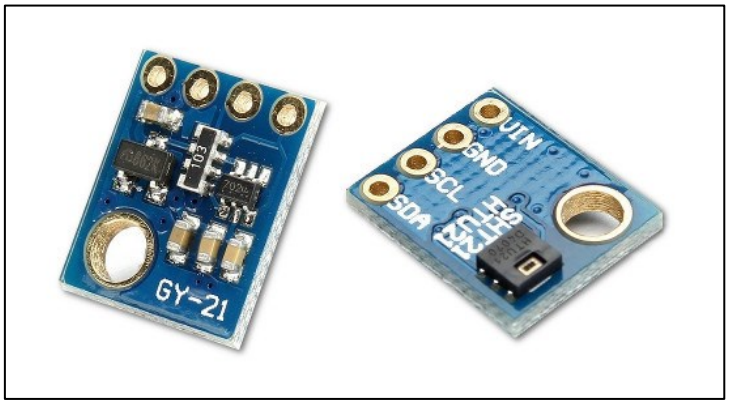
Pravděpodobně nejznámějším a nejpoužívanějším digitálním senzorem ve spojení s Raspberry Pi je DS18B20, běžící na sběrnici W1B. Tento teploměr disponuje dostatečným rozsahem teplot (od -55°C do +125°C s přesností ±0.5°C) [18] pro použití v inteligentní domácnosti. Zapojení jednoho senzoru (jednoho či více) je patrné na Obrázek 11.



Obrázek 11, zapojení DS18B20 [19]

3.2.4 Teplotní a vlhkostní senzor HTU21D

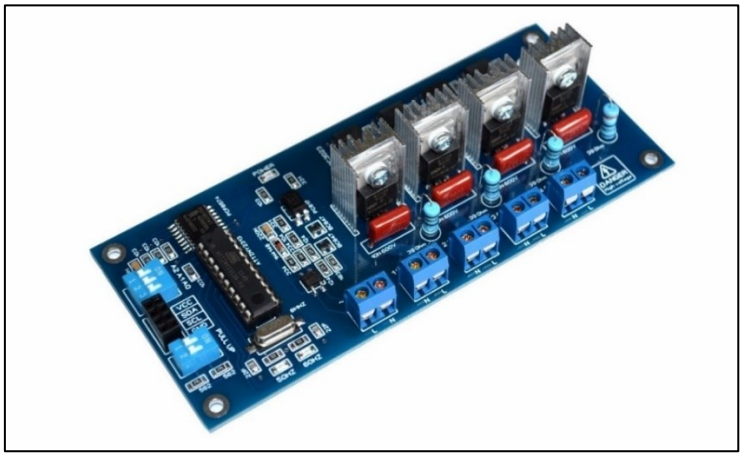
Pokud by bylo zapotřebí s teplotou měřit i vlhkost, či by nestačil DS18B20, může jako alternativa posloužit senzor HTU21D běžící na I2C sběrnici. Hotovou osazenou DPS lze zakoupit v provedení GY-21, viz Obrázek 12. Pomocí HTU21D lze například měřit vlhkost ve skleníku, či řídit spouštění zvlhčovače vzduchu.



Obrázek 12, GY-21 [20]

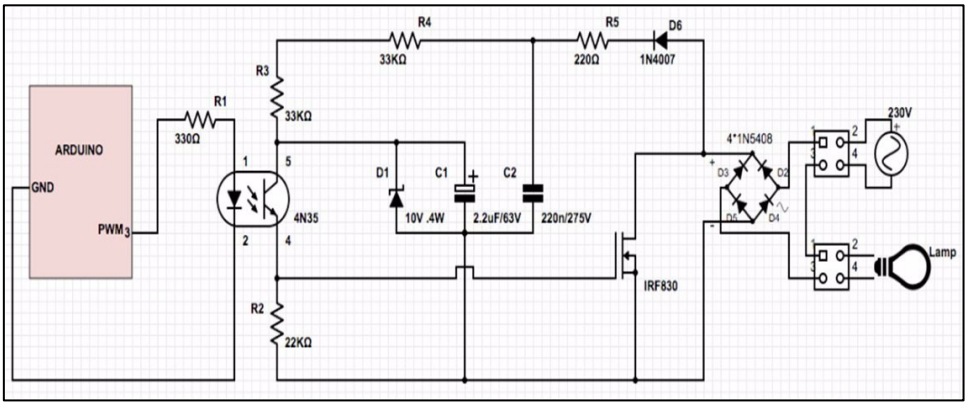
3.2.5 PWM stmívač

Výběr cenově dostupného stmívače, fungujícího dohromady s Raspberry Pi je značně omezený. Běžný triakový stmívač nelze spolehlivě ovládat nesynchronní PWM modulací, jelikož pulzy musí být synchronizovány s frekvencí sítě. Pokud by synchronizaci mělo provádět samotné Raspberry Pi, bylo by touto úlohou značně zatěžované. Úlohu synchronizace může provádět jiný systém nezávislý na Raspberry Pi, ale cena takovéto realizace je poté vyšší. Je tedy možné zakoupit I2C triakový stmívač viz Obrázek 13, běžící v kooperaci s mikroprocesorem ATTINY2313, nicméně cena takovéhoto provedení se blíží ceně samotného Raspberry Pi.



Obrázek 13, snart Arduino dimmer [21]

Pokud ale řídicí systém disponuje PWM modulací, lze za její pomoci měnit plochu pod křivkou síťového kmitočtu, tedy rozdělit časový průběh síťového napětí na pulzy, jejichž šířku lze měnit PWM střídou. Na Obrázek 14 je zrealizovaný jednoduchý stmívač [22] založený na principu spínání unipolárního tranzistoru podle frekvence a střídy PWM. Cena takovéto realizace poté odpovídá přibližně pětině ceny stmívače s mikroprocesorem.



Obrázek 14, schéma PWM stmívače s unipolárním tranzistorem

3.2.6 Ovládání naklápění žaluzií

Řešení akčního členu pro ovládání naklápění žaluzií je specifické a závisí na druhu žaluzií. Vzhledem k velkému množství výrobců žaluzií není možné vytvořit univerzální systém pro ovládání všech žaluzií, ale je třeba tento systém přizpůsobit pro žaluzie daného výrobce. Hotová řešení ovládání žaluzií některých výrobců často obsahují pouze elektrické ovládání žaluzií bez regulace a zpětné vazby. Řešení bez regulace lze upravit na zpětnovazební, popřípadě je možné regulační akční člen vytvořit. Cílem této realizace je i nízká pořizovací cena. Realizace tedy byla zvolena vlastní. Zde jsou body potřebné k realizaci:

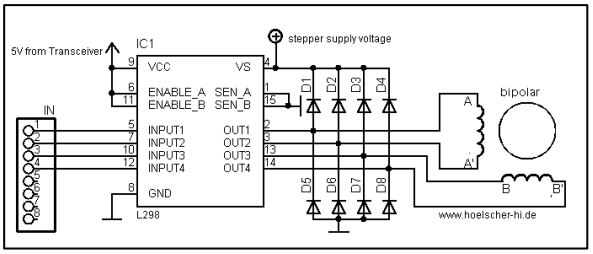
- síla potřebná pro otočení největším žaluziovým okenním modulem „Euro žaluzií“ (145x90 cm) byla experimentálně stanovena na:

$$F_z = 20 \text{ [N]} \tag{1}$$

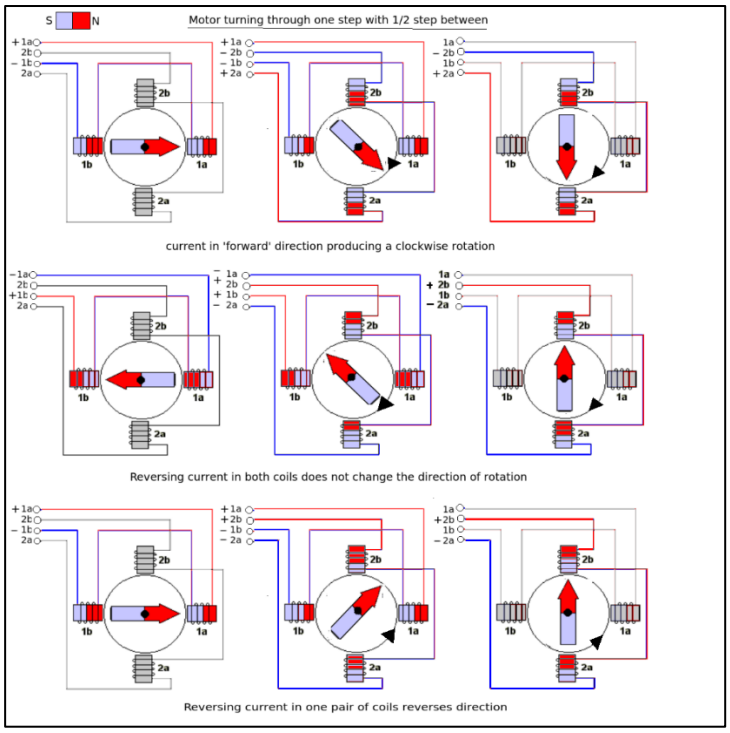
- délka potřebná pro přetočení žaluzií v plném rozsahu (z počáteční krajní polohy do konečné krajní polohy), v případě stejného modulu žaluzií, byla experimentálně stanovena na:

$$l_z = 5 \text{ [cm]} \tag{2}$$

- určení pohonu, pro řízené ovládání žaluzií. Je výhodné zvolit takový pohon, kterému je možné zadat konkrétní hodnotu natočení. Nejdostupnější variantou je buď volba krokového motoru, nebo modelářského servomotoru. Krokový motor nabízí kontinuální otáčení a přesně stanovený krok otočení. Nevýhodou je nutnost stanovení nulové polohy, což lze provést koncovým spínačem. Další nevýhodou je poměr síla/cena, kde s momentem síly motoru cena vzrůstá. Krokový motor je možné zpřevodovat, nicméně převodovka znamená další náklady na realizaci. V případě ovládání velkého počtu žaluzií může cena realizace značně vzrůst. Ovládání krokového motoru se obvykle realizuje pomocí H-můstku, ten lze zakoupit v provedení L298, viz Obrázek 15. Do vstupů L298 se vysílají správné kombinace čtyř 0-1 signálů a motor se podle nich hýbe, kombinace viz Obrázek 16

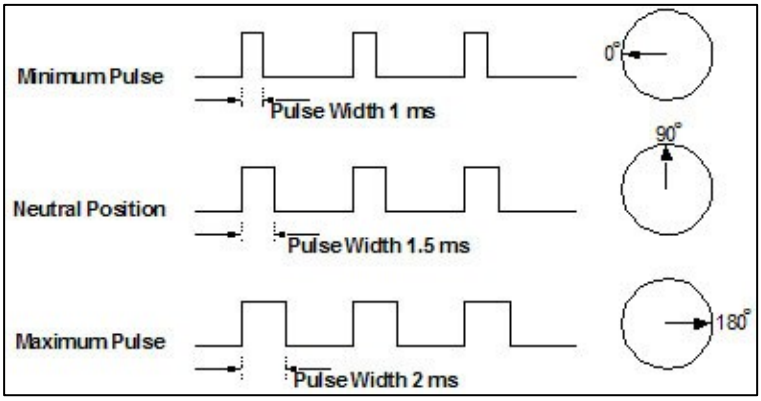


Obrázek 15, obvod L298 [23]



Obrázek 16, krokový motor, kombinace signálů potřebných pro otáčení rotoru [24]

Oproti krokovému motoru kombinuje modelářský servomotor výhody dobrého poměru cena/síla. V digitálním provedení lze servomotor snadno ovládat změnou střídy PWM modulace, viz Obrázek 17. Nevýhodou modelářského servomotoru je často omezený rozsah otáčení, který činí obvykle maximálně 180°.



Obrázek 17, servomotor ovládání natočení [25]

Digitální servomotor DS3115MG(A) 180° (Obrázek 18) disponuje momentem síly:

$$M_s = 15\ [\text{kg}] / 1\ [\text{cm}] \tag{3}$$

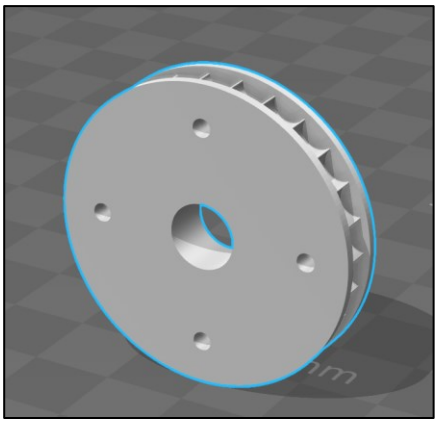
Pokud je zapotřebí vykonat určitou dráhu se servomotorem omezeným svým úhlem rotace, tak lze použít vzorec:

$$r = \frac{l_z}{\pi}\ [\text{cm}] \tag{4}$$

Poloměr ramena pro vykonání dráhy 5 cm, je vypočítán a zaokrouhlen na 1,6 cm. Pro dimenzi servomotoru je bezpečnější zaokrouhlit výsledek nahoru a na celá čísla, tedy na 2 cm. S ramenem o r = 2 cm vychází maximální síla, kterou je servomotor DS3115MG(A) s tímto ramenem schopný vyvinout, na 7.5 kg, což je více než je potřeba, nicméně předimenzování pohonu je pro tuto aplikaci výhodné.

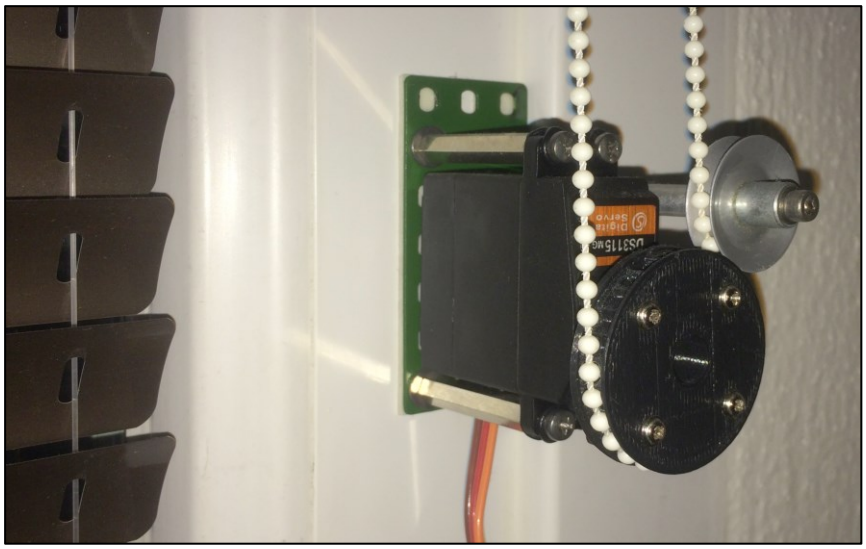
Obrázek 18, DS3115MG(A) 180°

Realizace byla po konzultaci s vedoucím práce změněna z původního krokového motoru na servomotor, z důvodu nízké pořizovací ceny servomotoru. K pohonu je zapotřebí vyrobit ozubené kolečko pasující do řetízku žaluzií a na nosnou konstrukci servomotoru. Ozubené kolečko lze vyrobit například na 3D tiskárně, viz Obrázek 19 a příloha 1.



Obrázek 19, ozubené kolo řetízku žaluzií

Nosná konstrukce může být zhotovena například ze stavebnice Merkur, viz Obrázek 20.



Obrázek 20, nosná konstrukce mechaniky ovládání žaluzií

3.3 Software

3.3.1 Volba operačního systému

- Windows 10 IoT Core:

  Projekt Windows 10 IoT se ukázal jako zajímavý, leč prozatím neúplný projekt. Například jádru chybí podpora W1B sběrnice, jejíž podpora je řešena externí knihovnou třetí strany. Nicméně ne zcela se daří takovouto knihovnu plnohodnotně využívat. Jelikož se o vyčítání dat ze sběrnice pokouší přímo procesor, často se stává, že výsledkem z čidla DS18B20 je i teplota -200°C.

  Problémem z hlediska budování inteligentní domácnosti se ukazuje chybějící podpora sběrnic jádrem OS, jak už bylo zmíněno, například W1B. Dalším problémem je nestabilita jádra, která je postupně opravována aktualizacemi. Při zkušebním provozu byl například problém s připojením k wifi, který znamenal opakovaný samovolný restart zařízení, který při běhu inteligentní domácnosti nesmí nastat.

  Pozitivní je naopak moderní přístup společnosti Microsoft ke komunikaci Raspberry Pi s dalšími zařízeními. Není tedy velký problém propojit Raspberry Pi s aplikací běžící v mobilním zařízení.

  Programování Raspberry Pi je možné skrze Microsoft Visual Studio Community, běžícím na jiném x86/x64 počítači. Nelze programovat na samotném Raspberry Pi [26]. Platí zde tedy master-slave vztah. Výhodou programování na master počítači je šetření dostupných prostředků na slave stroji, kde se nemusí nacházet složité vývojové prostředí. Dále je, oproti linuxovým platformám, snadné propojení obou počítačů skrze například wifi, kam poté lze snadno nahrát již zkompilovaný kód.

- Linux Raspbian [27]:

  Distribuce OS Raspbian je, komunitou oficiálně podporovanou, upravenou verzí Linux Debian, přizpůsobenou pro ARM počítače. Raspbian podporuje všechny sběrnice, dostupné na Raspberry Pi. Raspbian, díky dlouhému vývoji, patří mezi nejstabilnější OS [28] a mohou na něm běžet různá vývojová prostředí. Lze tedy na této platformě psát a testovat danou aplikaci, což je výhodné. Mezi vyspělá otevřená vývojová prostředí patří Qt, Python, Java a další. Nevýhodou je pro program nutný běh Linux kernelu a dalších služeb, které berou část výkonu Raspberry Pi. Raspbianu je dále třeba vytknout omezenou podporu knihoven pro práci se senzory a akčními členy. V porovnání například s Arduinem, které má takovéto knihovny k dispozici ve

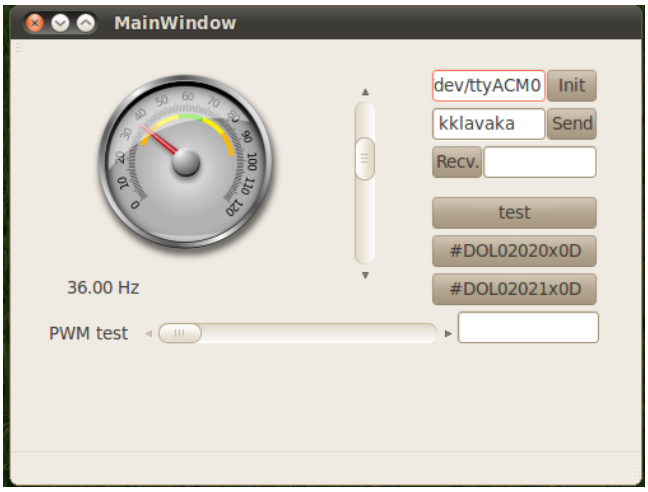
vývojovém prostředí, je programátor s Raspberry Pi odkázaný na knihovny třetích stran, či musí nativní knihovny Arduina přepsat pro použití s Raspberry. Přepis knihoven se může zdánlivě jevit jako snadný, nicméně je třeba brát v potaz, že nadřazené knihovny pro ovládání sběrnic jsou psány odlišně, a tedy i jinak komunikují.

3.3.2 Volba vývojového prostředí

Před volbou prostředí a jazyku je třeba zvážit, jaký druh aplikace bude vyvíjen. Cílem práce je vyvinout aplikaci, která bude vzdáleně ovládat akční členy a vyčítat hodnoty ze senzorů. Pro komunikaci s akčními členy se na Raspbianu převážně hodí jazyky C, C++, Java a Python. Velkým omezením při volbě jazyku jsou znalosti programátora, z tohoto pohledu je nejvýhodnější programování v C++.

Problematická část programu je jeho komunikace s okolím vně Raspberry Pi. Lze využít například komunikaci skrze Linux GUI vzdálenou plochu, nicméně tato varianta je komplikovaná a neefektivní. Lze také komunikovat skrze otevřené porty pomocí aplikace u klienta. Toto řešení je elegantní, nicméně znamená spoustu překážek, zejména různé aplikace pro různé platformy. Fatální je poté vývoj ovládací aplikace pro uzavřené platformy, kde nelze do zařízení nahrát jinou než ověřenou aplikaci, a je třeba jí nejprve umístit do obchodu, což vždy nelze snadno udělat. Nevýhody obou řešení lze vyřešit pomocí webového serveru. Webový server je platformě nezávislý a lze tedy ovládat aplikaci jak z mobilního zařízení, tak ze stolního počítače. Problémem je samotný server, který musí aktivně komunikovat s aplikací. PHP nabízí tvorbu kódu běžícího přímo ve webové části. Nevýhodou je omezená podpora akčních členů a senzorů. Tedy pro běh komplexní aplikace PHP není příliš vhodné. Vhodným řešením je kombinace web serveru postaveného na C++.

Qt je C++ based vývojové prostředí [29], které je nejbližší programátorovi znalému základní programování v C či C++. Qt nabízí i grafickou nadstavbu, lze tedy relativně snadno vytvořit vlastní aplikaci s GUI (oknem), viz Obrázek 21. Qt nabízí i tvorbu webového grafického prostředí, nicméně toto prostředí nefunguje jako server, tedy nelze jím řešit serverovou část aplikace.



Obrázek 21, ukázka grafické nadstavby aplikace vytvořené v Qt

3.3.3 Volba C++ web serveru

Stěžejní a časově nejnáročnější je volba C++/Qt based serveru. Tato varianta volby serveru se i v dnešní době ukazuje jako velmi okrajová záležitost, které se ve světě otevřeně věnuje jen hrstka firem či jedinců. Česká firma Hobrasoft [30], (Obrázek 22) nabízí webový server vytvořený v Qt a s otevřeným kódem pod licencí GPL.

HOBRASOFT

Obrázek 22, logo firmy Hobrasoft

Principiální popis komunikace HTTPD níže vypsaný v bodech ukazuje jak probíhá komunikace mezi serverem a klientem:

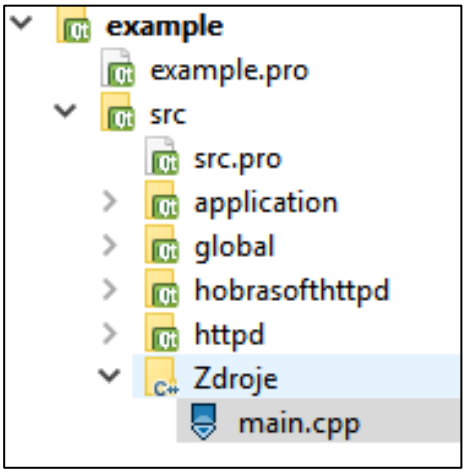
- server:
  - aplikace vytvořená v C++ a Qt
  - zabudovaný http server
  - A: server poskytuje statické stránky (čisté html, obrázky, css, javascript)
  - B: server poskytuje json api pro přístup k vnitřním proměnným běžící aplikace
- klient:
  - aplikace se zobrazuje ve webovém prohlížeči s podporou HTML5
  - jako první se nahraje statický obsah (html, obrázky, css, javascript)
  - poté se v prohlížeči spustí program v JavaScriptu
  - program v JavaScriptu se znovu připojí na server a vyžádá si přes json api informaci o vnitřním stavu aplikace na serveru

- podle hodnot načtených přes json api se upraví podoba statického dokumentu nataženého a zobrazeného dříve
- propojení může běžet neustále a dynamicky upravovat podobu zobrazené stránky
- pokud je potřeba reagovat na uživatelské vstupy (stisknutí tlačítka a podobně), opět se vyvolá kus JavaScript kódu, který se připojí na server a předá přes json api potřebné údaje

3.3.4 Aplikace

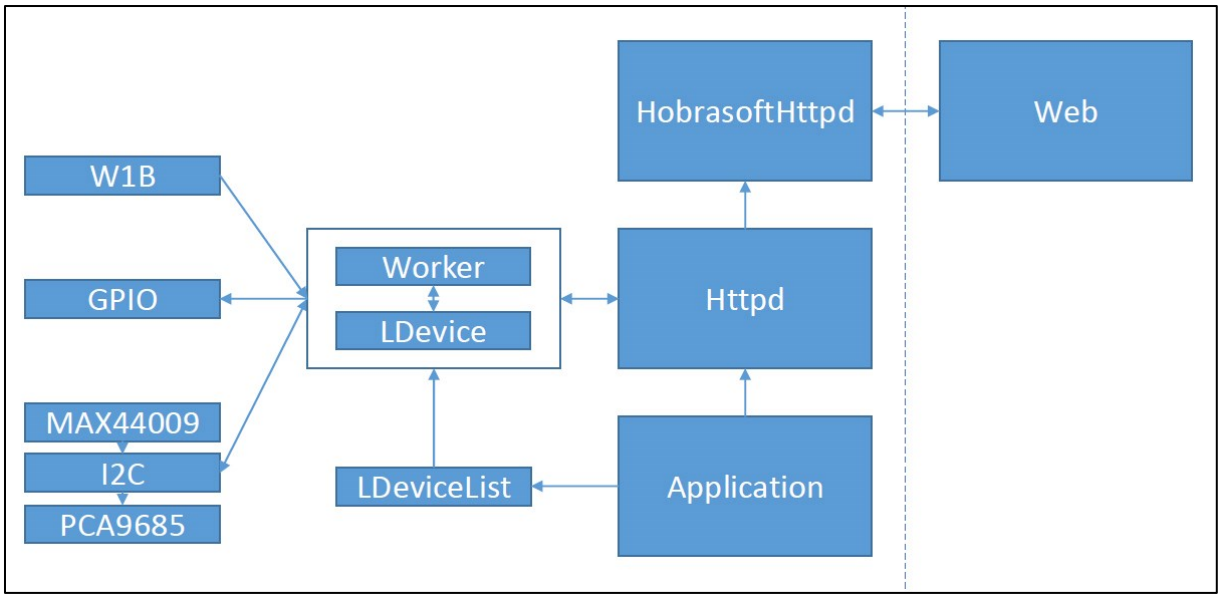
Struktura projektu v Qt (Obrázek 23). Pro pořádek, příklad inteligentní domácnosti, projekt pojmenován **Example**, ten v sobě nese sub projekt **src.pro**, který obsahuje sub projekty:

- **application** - aplikace s logikou inteligentní domácnosti
- **global** - určení složek pro dočasné soubory
- **hobrasoftthttpd** - Hobrasoft server pro komunikaci s vnějškem
- **httpd** - kontroléry propojující server s aplikací



Obrázek 23, struktura projektu v Qt

Blokové schéma struktury projektu, pro lepší orientaci v programu, znázorněno na Obrázek 24. Bloky znázorňují jednotlivé třídy či propojené třídní celky. Šipky poté znázorňují směr toku informací či postup tvorby instancí tříd. Přerušovaná čára značí komunikaci vně aplikace skrze JSON.



Obrázek 24, vnitřní struktura projektu

Start aplikace a serveru (pro přehlednost, znázorněny pouze stěžejní části kódu). `Main.cpp` vytvoří instanci třídy `Application.cpp`:

```
Example::Application *Example::application = NULL;

int main (int argc, char *argv[]) {
    Example::application = new Example::Application(0);
    return app.exec();
}
```

Konstruktor třídy `Application.cpp`, skrze signály a sloty, procedurou `init()`, vytvoří seznam dostupných zařízení `LDeviceList.cpp` a spustí webový server procedurou `Httpd()`:

```
using namespace Example;
Application * Application::instance = NULL;

Application::Application(QObject *parent) : QObject(parent) {
    instance = this;
    QTimer::singleShot(0,this,SLOT(init()));
}
void Application::init() {
    m_list = new LDeviceList(this);
    new Example::Httpd::Httpd(this);
}
LDeviceList * Application::list() const {
    return m_list;
}
```

Dostupná zařízení jsou dopředu a pevně určené položky (instance třídy `LDevice.cpp`), které jsou vytvořeny konstruktorem třídy `LDeviceList.cpp`, a které obsahují procedury pro přímou obsluhu fyzických komponent. Jedná se tedy o položky, které má server za běhu ovládat.

```
LDeviceList::LDeviceList(QObject *parent) : QObject(parent) {
    setObjectName("LDeviceList");
    addDevice( new LDevice(this,  "1", 1, 0x40, 100 , 1, 180,
1014, 1, 0));
}
```

Popis třídy `LDevice.cpp`: třída obsahuje i třídu `Worker.cpp`, která je vytvořena tak, že běží v jiném vláknu. Jiné vlákno je zhotoveno proto, aby mohla nezávisle běžet regulační smyčka, která nebude blokovat vlákno obsluhující server a vyčítání hodnot z čidel. `LDevice.cpp` je nejhlubší položka uvnitř mechanismu, která obsahuje veškeré metody potřebné pro práci, zadávání a vyčítání hodnot:

```
namespace Example{
class Worker : public QObject {
    Q_OBJECT
  public:
    Worker();
  signals:
    void akceHotova(int);
  public slots:
    void slotWorkerValues(int,int,int,QString, int, QString);
  private slots:
    void slotUdelejAkci();
  private:
    MAX44009 * lux;
    int m_workerValue;
    int m_workerLuxValue;
    int m_regulationValue;
    int aktualniPIDHodnota, m_pozadovanaPIDHodnota,
rozdilPIDHodnot, predchoziRozdilPIDHodnot, sumaPIDHodnot;
    int koefInt, koefDer, koefProp, Int, Der, Prop, setValue,
predchPID;
    QTimer *m_timer;
    int m_PIDRequired;
    int m_PIDInterval;
    int m_PIDRun;
    QString m_workerId;
```

```
    int m_44009Addr;
    QString m_w1bCestaKSoboru;
    int m_ds18b20IntTemp;
    double m_ds18b20DoubleTemp;
    QString m_ds18b20CtiRadku;
    QString m_ds18b20StringTemp;
    QString m_w1bAddr;
};
class LDevice : public QObject
{
    Q_OBJECT
public:
    explicit LDevice(QObject *parent, const QString& id, int
i2c, int pca9685Bus, int ferq, int pca9685Address, int
minValue, int maxValue, int akcniClenI2c, int akcniClenGPIO);
    void udelejAkci(int);
    void setActuatorValue(int);
    void setWorkerIntensity(int);
    void setWorkerInterval(int);
    void setWorkerAutorun(int);
    void setWorkerI2cAddr(int);
    void setWorkerW1bAddr(std::string);
    int getActuatorValue() const;
    int getSensorValue() const;
    int getWorkerAutorun() const;
    int getWorkerInterval() const;
    int getWorkerRequired() const;
    int getI2cSensor() const;
    QString getW1bSensor() const;
    const QString& id() const;
    int pca9685Bus() const;
    int pca9685Freq() const;
    int pca9685Address() const;
    QVariantMap webStatus() const;
    void getLuxTemp();
  signals:
    void    statusChanged(const LDevice *);
    void    workerValues(int, int, int, QString, int,
QString);
  private slots:
    void signalSetValue(int);
    void    updateStatus();
  private:
    QThread m_thread;
    QString m_id;
    int m_BusI2c;
    int m_pca9685Bus;
    int m_pca9685Freq;
    int m_pca9685Address;
    float m_setActuatorValue;
```

```
    float m_minActuatorValue;
    float m_maxActuatorValue;
    float m_actuatorRange;
    int m_relativeActuatorValue;
    float m_getSensorValue;
    QTimer *m_timer;
    int     m_counter;
    int m_workerRequiredValue;
    int m_workerInterval;
    int m_workerRun;
    int m_workerI2cSensorAddr;
    int m_akcniClenI2c;
    int m_akcniClenGPIO;
    float m_requiredLuxValue;
    QString m_w1bCestaKSoboru;
    int m_ds18b20IntTemp;
    double m_ds18b20DoubleTemp;
    QString m_ds18b20CtiRadku;
    QString m_ds18b20StringTemp;
    QString m_workerW1bSensorAddr;
    QString m_ds18b20SensorName;
};
```

Popis komunikace webové stránky s třídou `LDevice.cpp`. Pro příklad uveden popis nastavení úrovně žaluzií. Na webové stránce je zadána nová hodnota posuvníku s id **A1**.

```
<input type="range" min="0" max="100" id="A1"></input>
```

Akce posuvníku vyvolá JSON skript, který zavolá metodu controlleru aplikace `setPwm` a předá jí vnitřní identifikátor **1** a hodnotu v rozsahu **0-100**.

```
    $('#A1').change(function(){
    $.getJSON('/setPwm/1?Intensity='+$(this).val());
    });
```

V Qt aplikaci jsou zhotoveny takzvané kontroléry, které poslouchají JSON skripty webových stránek. Předání hodnoty tohoto příkladu poslouchá kontrolér `controllerSetPwm.cpp`. Tento kontrolér převede zadanou hodnotu do datového typu `int`. Kontrolér dále zavolá položku seznamu s požadovaným identifikátorem a předá hodnotu metodě `setActuatorValue`(), která jej zpracuje. Nakonec kontrolér pošle webu zpět informaci, že úspěšně došlo k předání hodnoty metodou `serviceOK`().

```
void ControllerSetPwm::serviceIdGet
(HobrasoftHttpd::HttpRequest *request,
HobrasoftHttpd::HttpResponse *response, const QString& id) {
    int intensity = request->parameter("Intensity").toInt();
    QVariantMap data;
    data ["intensity"] = intensity;
    LIST->device(id)->setActuatorValue(intensity);
    serviceOK(request, response, data);
}
```

Pro úplnost je třeba dodat, že kontroléry jsou shromážděny ve třídě `requestMapper.cpp`, do které přijde zadaná hodnota ještě před kontrolérem. V tomto mapperu se vyskytují názvy uvedené na webu, v tomto případě `setPwm`.

```
void RequestMapper::service(HttpRequest *request,
HttpResponse *response) {
    QString path = request->path();
    #define ROUTER(adresa, trida) \
        if (path.startsWith(adresa)) { \
            HttpRequestHandler *controller = new trida
(connection()); \
            controller->service(request, response); \
            return; \
            }
    ROUTER("/setPwm", ControllerSetPwm);
    ROUTER("/getPwm", ControllerGetPwm);
    ROUTER("/setVal", ControllerSetLux);
    ROUTER("/getVal", ControllerGetLux);
    ROUTER("/setInterval", ControllerSetInterval);
    ROUTER("/getInterval", ControllerGetInterval);
    ROUTER("/setAutorun",  ControllerSetAutorun);
    ROUTER("/getAutorun",  ControllerGetAutorun);
    ROUTER("/getWorkerVal", ControllerGetWorkerVal);
    ROUTER("/getI2cSensor", ControllerGetSensor);
    ROUTER("/setI2cSensor", ControllerSetSensor);

    HttpRequestHandler::service(request, response);
    response->flush();
}
```

Zpět ke kontroléru. `setActuatorValue()` je již existující veřejná metoda instance třídy `LDevice.cpp`. Zadaná hodnota z webu je předána neveřejné položce `m_setActuatorValue`, dále je zadaná hodnota zpracována pro metodu `setPWM()` třídy `PCA9685.cpp`. Třída `PCA9685.cpp` již dále komunikuje skrze I2C s hardwarem, který vykoná požadovanou operaci. Poslední zajímavou událostí je emitování signálu

statusChanged(), tento signál je propojen s kontroléry pro vyčítání dat na webu a tím dojde k obnově dynamických dat na webové stránce.

```
void LDevice::setActuatorValue(int value)
{
    if (m_akcniClenI2c == 1){
    m_setActuatorValue = value;
m_relativeActuatorValue=m_minActuatorValue+m_actuatorRange*(m
_setActuatorValue/100);
    if (m_relativeActuatorValue > m_maxActuatorValue)
        m_relativeActuatorValue = m_maxActuatorValue;
    if (m_relativeActuatorValue < m_minActuatorValue)
        m_relativeActuatorValue = m_minActuatorValue;
    PCA9685 pwm(m_BusI2c, m_pca9685Bus);
    pwm.setPWMFreq (m_pca9685Freq);
    pwm.setPWM(m_pca9685Address,0,m_relativeActuatorValue);
    emit statusChanged(this);
    }
```

Popis komunikace třídy LDevice.cpp s webovou stránkou. Pro příklad uveden popis vyčtení nastavené úrovně žaluzií. Uvnitř třídy LDevice.cpp se nachází metoda webStatus(). Tato metoda obsahuje datový typ QVatiantMap, který dokáže vytvořit strukturovaný řetězec dat s uživatelem zvolenými identifikátory a který je možné, díky Hobrasoft Httpd, propojit s JSON na webové stránce. Tento řetězec na požádání předá hodnoty vnitřních proměnných instance třídy LDevice.cpp. Pokud je metoda webStatus() zavolána, dojde k vyčtení požadovaných dat.

```
QVariantMap LDevice::webStatus() const {
    QVariantMap data;
    data["id"]                  = m_id;
    data["pca9685Bus"]          = m_pca9685Bus;
    data["pca9685Address"]      = m_pca9685Address;
    data["pca9685Freq"]         = m_pca9685Freq;
    data["setActuatorValue"]    = m_setActuatorValue;
    data["getSensorValue"]      = m_getSensorValue;
    data["workerRequiredValue"] = m_workerRequiredValue;
    data["workerInterval"]      = m_workerInterval;
    data["workerRun"]           = m_workerRun;
    data["workerI2cSensorAddr"] = m_workerI2cSensorAddr;
    data["workerW1bSensorAddr"] = m_workerW1bSensorAddr;
    return data;
}
```

Volání metody `webStatus()` je řízeno kontrolérem `controllerGetPwm.cpp`. Pakliže došlo k zadání nové hodnoty, byl emitován signál `statusChanged()`. Tento signál je propojen se slotem `sendUpdate()`. Tento slot je zároveň i metodou kontroléru, která vyčte data metodou `webStatus()`.

```
void ControllerGetPwm::serviceIdEvents
(HobrasoftHttpd::HttpRequest *request,
HobrasoftHttpd::HttpResponse *response, const QString& id) {
    LDevice *ldevice = LIST->device(id);
        connect(ldevice, SIGNAL(statusChanged(const LDevice
*)),
                this,   SLOT(   sendUpdate(const LDevice
*)));
        sendUpdate(ldevice);
}
void ControllerGetPwm::sendUpdate(const LDevice *device) {
    serviceEvent(NULL, NULL, device->webStatus());
}
```

Změny dat očekává JSON na webové stránce. Pokud dojde ke změně, Java script tuto změnu zachytí a zavolá funkci `updateStatusB1()`.

```
    var source = new EventSource("/getPwm/1/events");

    source.addEventListener("status", function(event) {
            updateStatusB1(JSON.parse(event.data));
            }, false);
```

Funkce `updateStatusB1()` si poté vyčte aktuální hodnotu.

```
function updateStatusB1(data) {
    if (typeof data == 'undefined') { return; }
     {
          $("#B1").val(data.setActuatorValue);
     }}
```

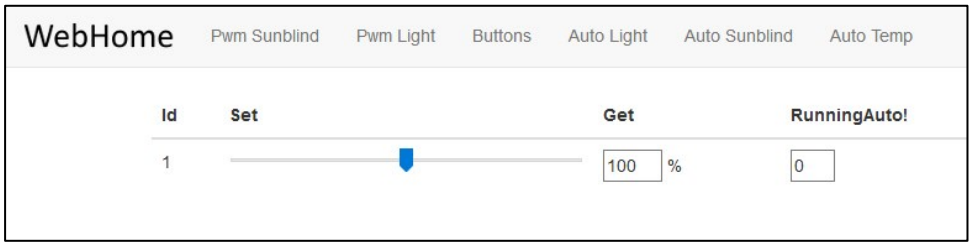
Aktuální hodnota je předána webové stránce, která ji zobrazí.

```
<input type="text" size="3" value="get" width="20"
readonly="readonly" id="B1"> %</input>
```

Stejným způsobem komunikují i ostatní kontroléry. Pro každou položku, která je k dispozici na webové stránce, je potřeba nezávislý kontrolér.

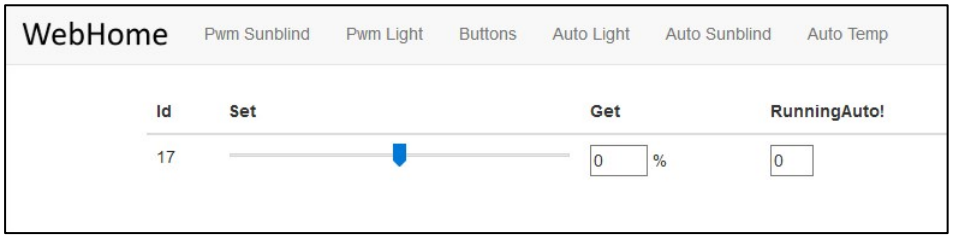
Webový design má díky prostředí bootstrap [31] dvě podoby, podobu mobilní a podobu pro desktop, do které se dle potřeby dynamicky zkonvertuje.

- **Pwm Sunblind** (Obrázek 25): Obsahuje posuvník, **Id** odpovídá položce seznamu uvnitř aplikace. **Id** jsou zvoleny tak, že hodnoty **1-16** jsou předurčeny pro I2C PWM žaluzie, **17-33** jsou předurčeny pro I2C PWM řízení světel. Počet **Id** je určen moduly PCA9685, které obsahují 2x16 PWM kanálů, kde první skupina se od té druhé liší frekvencí PWM (stmívač potřebuje frekvenci vyšší, servomotory zase nižší). Vyšší hodnoty **Id** (**34** a výš) jsou předurčeny pro jiná zařízení, většinou on-off tlačítka, jejichž počet je poté omezený počtem GPIO pinů Raspberry Pi 3. V případě rozšíření aplikace o další I2C PWM moduly je poté nutná její úprava. Položka **Set** je určena pro přímé nastavení akčního členu. Položka **Get** je zde pro vyčtení aktuální nastavené hodnoty. Položka **RunningAuto!** je zde pro upozornění, zda na daném kanálu není nastaven automatický režim, který by mohl anulovat uživatelem ručně nastavenou hodnotu.



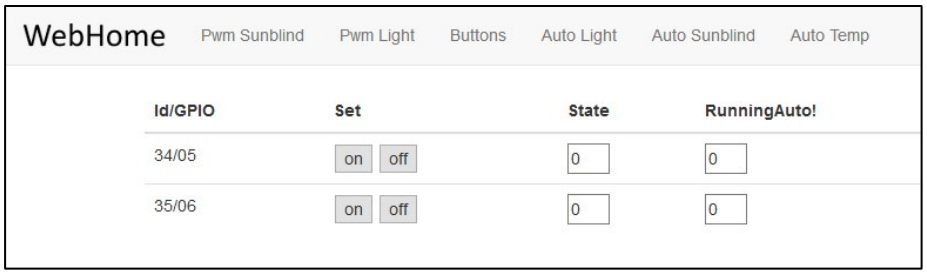
Obrázek 25, PWM Sunblind

- **Pwm Light** (Obrázek 26): Stejná logika jako **Pwm Sunblind**, pouze akčním členem je stmívač.



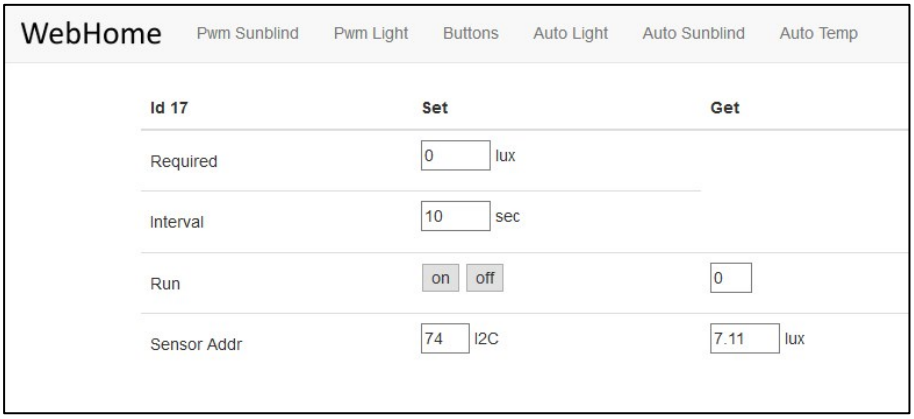
Obrázek 26, PWM Light

- **Buttons** (Obrázek 27): Zde je logika stejná jako v předchozím případě, jen **Get** je zde nahrazeno položkou **State**, který vyčítá aktuální nastavenou hodnotu. Posuvník je dále v tomto případě nahrazen tlačítky.



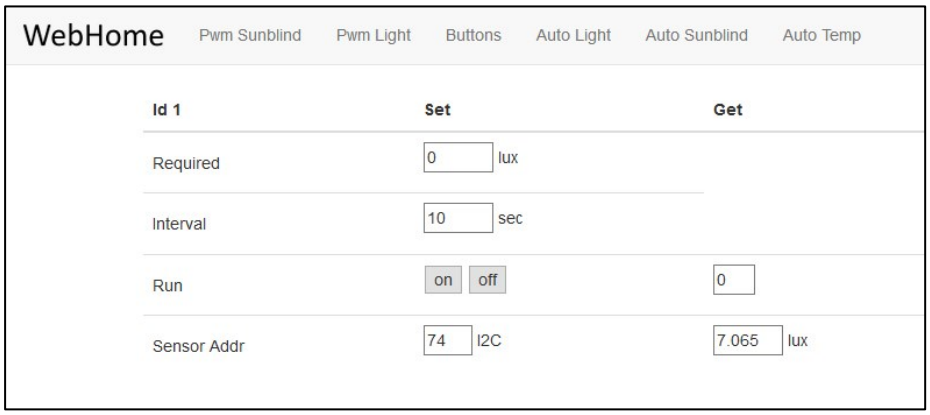
Obrázek 27, Buttons

- **Auto Light** (Obrázek 28): Nastavuje PID regulaci pro **Id 17**. Položkou **Required** je určena cílová hodnota, kterou má program dosáhnout. PID regulace je z důvodu pomalého vyčítání hodnot z akčních členů také pomalá. PID regulace má omezený počet cyklů pro docílení požadované hodnoty, běží jen omezenou dobu a poté se uspí. Intervalem je tedy určena doba opakovaného spouštění PID regulace. Položkou **Run** se PID regulace pro daný kanál uvádí do chodu. **Sensor Addr** je určen pro možnou změnu adresy senzoru. Je tedy možné mít větší počet světel řízených různými senzory. **Get** slouží pro ověření nastavené hodnoty, jedná se přímo o hodnoty vyčtené ze senzoru.



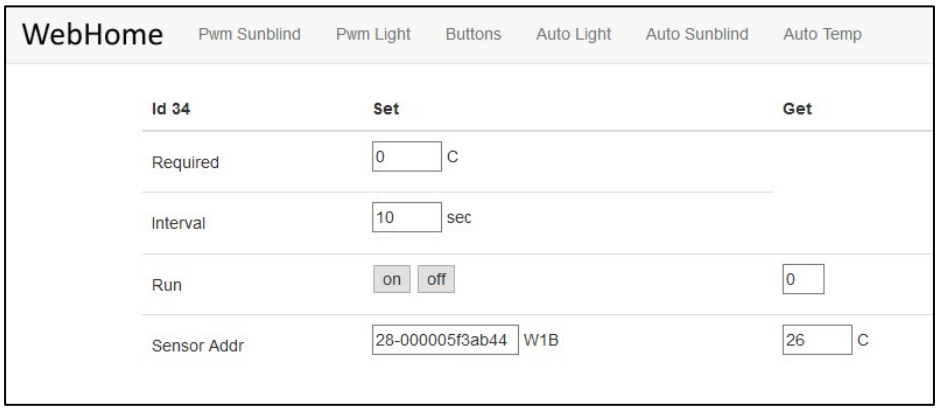
Obrázek 28, Auto Light

- **Auto Sunblind** (Obrázek 29): Stejná logika jako **Auto Light**, pouze jsou akčním členem žaluzie.



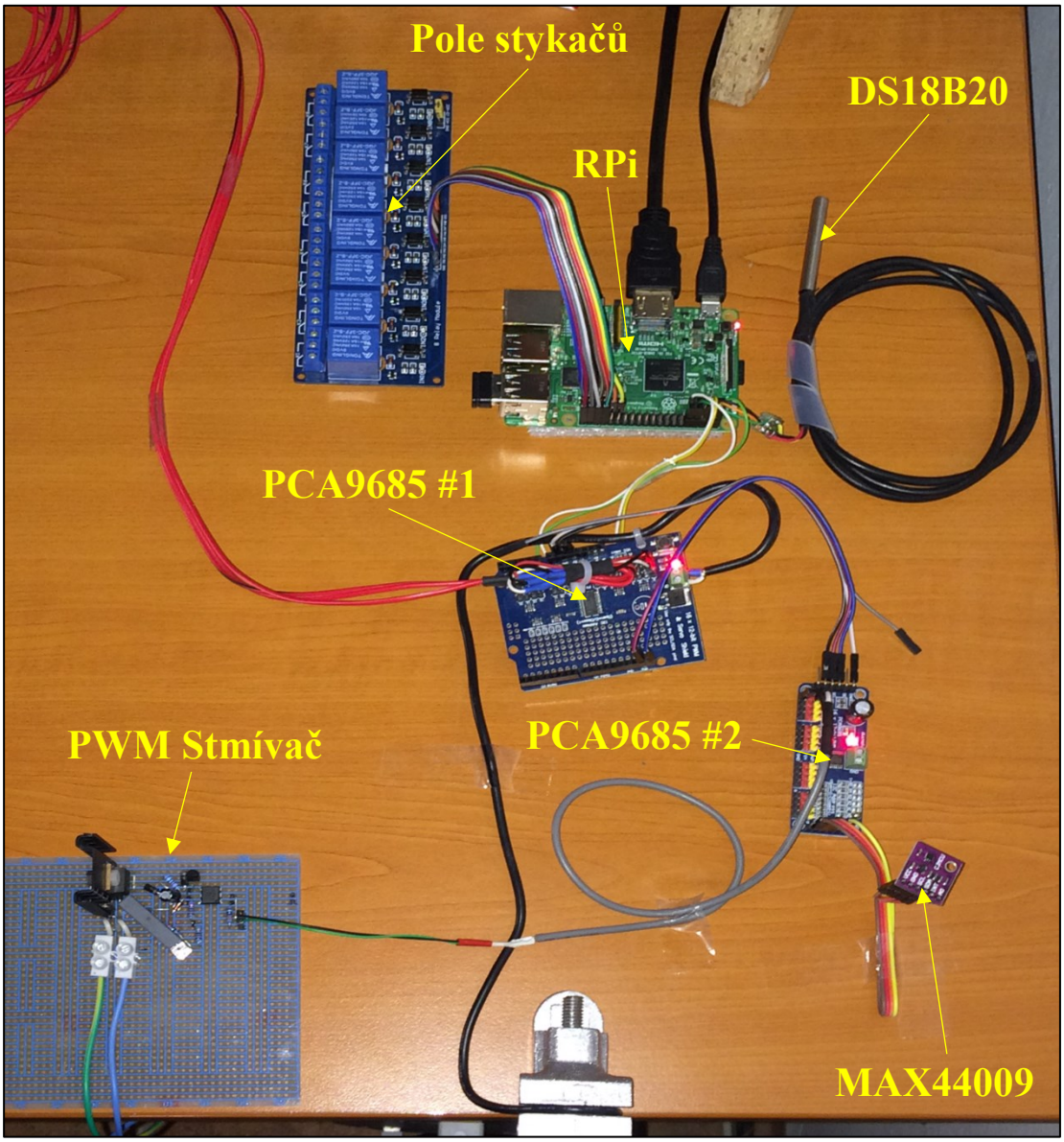
Obrázek 29, Auto Sunblind

- **Auto Temp** (Obrázek 30): Teplota již není řízená PID regulací, ale prostým on-off principem a akčním členem je v tomto případě stykač. Pokud požadovaná teplota převyšuje změřenou čidlem, dojde ke spuštění výstupu. Zde je výhodné experimentálně měnit hysterezi, ta v tomto případě odpovídá položce **Interval**.



Obrázek 30, Auto Temp

3.4 Zapojení



Obrázek 31, zapojení

Popis zapojení dle Obrázek 31:

- RPi - Raspberry Pi 3 model B, propojené s periferiemi skrze GPIO, I2C a W1B
- PCA9685 #1, realizace s přídavným napájením (přivedené černým vodičem do svorek pro napájení). Tento PWM modul ovládá servomotor žaluzií, který je připojen červenými vodiči, na obrázku není vidět. S Raspberry Pi je modul propojen pouze vodiči I2C (SDA+SCL), 5V a GND

- PCA9685 #2, realizace bez přídavného napájení, určená pro nevýkonové PWM aplikace, tedy pro stmívač. Aby došlo k rozlišení zařízení na sběrnici I2C, je tomuto modulu změněná adresa, to se provádí propájením plošek **A0**+**RW** na jednom z modulů
- MAX44009, pouze připojený skrze PCA9685 #2 na I2C
- DS18B20, ve standardním zapojení na W1B
- Pole stykačů, propojené s GPIO výstupy – příprava na ovládání topení, ventilátorů atd. Chybí pouze zásuvky pro připojení zástrček spotřebičů
- PWM stmívač, vstup připojen na první kanál PCA9685 #2, Výstup propojen s lampičkou

3.5 Instalace

3.5.1 Raspbian

Instalace se provádí dle oficiálních postupů na stránkách výrobce [32].

3.5.2 Qt

Instalace se provádí skrze příkazovou řádku instalací série následujících příkazů:

```
sudo apt-get install qt4-dev-tools
sudo apt-get install qtcreator
sudo apt-get install gcc
sudo apt-get install xterm
sudo apt-get install git-core
sudo apt-get install subversion
```

3.5.3 Aktivace W1B

Aby fungoval teploměr DS18B20 na Raspberry Pi, je třeba nejprve aktivovat W1B sběrnici, která v základu není na Raspbianu aktivována. Aktivace se provádí následující operací skrze příkazovou řádku

```
sudo nano /boot/config.txt
```

Dojde k otevření nano editoru a otevře se konfigurační soubor OS, zde je třeba připsat následující řádku a uložit. Po restartu systému již bude W1B funkční.

```
dtoverlay=w1-gpio
```

3.5.4 Aktivace I2C

Sběrnice I2C je také v základu deaktivována, zprovoznění opět skrze příkazovou řádku:

```
sudo apt-get install i2c-tools python-smbus
```

Po instalaci potřebných souborů je třeba přidat příkaz do konfiguračního soboru systému podobně jako u W1B:

```
sudo nano /boot/config.txt
```

Dojde k otevření nano editoru a otevře se konfigurační soubor OS, zde je třeba připsat následující řádku a uložit. Po restartu systému již bude I2C funkční.

```
dtparam=i2c_arm=on
```

3.5.5 Instalace WebHome

Je více způsobů, jak provozovat aplikaci WebHome. Lze vytvořit hotovou aplikaci skrze qmake (to je Qt obdoba cmake), či aplikaci provozovat skrze vývojové prostředí. Aplikace se nachází v příloze, konkrétně příloha 2, tu je třeba zkopírovat do Raspbianu. Dále je potřeba do systému umístit konfigurační soubor pro server, ten je třeba umístit do:

```
/home/pi/.config/hobrasoft
```

Konfigurační soubor je také přítomen v uvnitř přílohy 2. Umístění aplikace do OS počítá s umístěním na plochu. Pokud by umístění bylo jiné, je třeba změnit i cestu k dokumentům v konfiguračním souboru pro server.

3.5.6 Další požadavky pro běh WebHome

jQuery a další knihovny, které jsou součástí skriptů ve webové části, mohou pro správný běh potřebovat přístup k internetu. Pokud webová část aplikace nefunguje dle předpokladu, je třeba zařízení dodat přístup k internetu.

4 Závěr

Úkolem této práce bylo téma „Možnosti využití Raspberry Pi 3 pro vzdálené řízení domácnosti“. Primární úkol práce byl splněn, realizace tohoto úkolu je univerzální, funkční a s velkým potenciálem pro další vývoj a integraci do domácnosti. Využití je možné jak v domácích podmínkách, tak i v průmyslu. Možnost propojit primitivní aplikaci s webovým serverem dává možnosti, které se mohou hodit takřka kdekoliv, kde je třeba vzdálený přístup. Velkou výhodou je možnost vyvíjet, ovládat a provozovat aplikaci na jednom jediném počítači. Problémem této aplikace je stabilita a spolehlivost. Ošetřit všechny výjimky v programu není snadné. U Linuxu mohou způsobit problémy například práva čtení či zápisu do souboru. Spolehlivost Raspberry Pi 3 model B také vysoce snižuje umístění OS na paměťovou kartu, která často podlehne vysokému počtu zápisů dat. Problém stability paměťové karty sice lze řešit například USB externím pevným diskem, nicméně tak dochází ke komplikaci realizace.

Toto téma práce je velmi rozsáhlé a žádá si velké množství času. V realizaci tedy nebyly provedeny veškeré implementace.

5 Odkazy a reference

[1] „www.samsung.com,“ 08 11 2016. [Online]. Available: www.samsung.com/uk/smartthings/. [Přístup získán 08 11 2016].

[2] „www.huawei.com,“ 08 11 2016. [Online]. Available: www.huawei.com/minisite/iot/en/smarthome.html. [Přístup získán 08 11 2016].

[3] „www.sony.com,“ 08 11 2016. [Online]. Available: www.sony.com/regional/smart-home. [Přístup získán 08 11 2016].

[4] „www.bigclown.com,“ 08 11 2016. [Online]. Available: www.bigclown.com. [Přístup získán 08 11 2016].

[5] „home-assistant.io,“ 08 11 2016. [Online]. Available: home-assistant.io. [Přístup získán 08 11 2016].

[6] „Wikipedia.org, Raspberry_Pi_3,“ 22 12 2016. [Online]. Available: https://cs.wikipedia.org/wiki/Raspberry_Pi#Raspberry_Pi_3. [Přístup získán 22 12 2016].

[7] „element14,“ 20 01 2017. [Online]. Available: element14.com/RaspberryPi. [Přístup získán 20 01 2017].

[8] „LoverPi,“ 30 11 2016. [Online]. Available: https://www.loverpi.com/blogs/news/94801153-raspberry-pi-3-banana-pi-m3-orange-pi-plus-2-odroid-c2-spec-comparison. [Přístup získán 30 11 2016].

[9] „banana-pi,“ 14 12 2017. [Online]. Available: http://www.banana-pi.org/m3.html. [Přístup získán 14 12 2017].

[10] „robodoupe,“ 02 03 2017. [Online]. Available: http://robodoupe.cz/2014/beaglebone-black-vic-nez-jen-alternativa-k-raspberry-pi/. [Přístup získán 02 03 2017].

[11] „beagleboard,“ 20 12 2017. [Online]. Available: http://beagleboard.org/support/bone101. [Přístup získán 20 12 2017].

[12] „root,“ 08 01 2017. [Online]. Available: https://www.root.cz/clanky/arduino-webovy-server-i-klient-do-ruky/. [Přístup získán 08 01 2017].

[13] „software.intel.com,“ 18 12 2016. [Online]. Available: https://software.intel.com/en-us/iot/hardware/joule/dev-kit. [Přístup získán 18 12 2016].

[14] „adafruit,“ 20 12 2017. [Online]. Available: www.adafruit.com/datasheets/PCA9685.pdf. [Přístup získán 20 12 2017].

[15] „www.esp8266-projects.com,“ 26 12 2016. [Online]. Available: http://www.esp8266-projects.com/2016/06/pca9685-16channel-12-bit-pwm-ic-bus-led.html. [Přístup získán 26 12 2016].

[16] „hackster,“ 08 01 2017. [Online]. Available: https://www.hackster.io/kurt_e_clothier/motion-controlled-servos-using-leap-motion-7d2c18. [Přístup získán 08 01 2017].

[17] „maximintegrated,“ 11 03 2017. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX44009.pdf. [Přístup získán 11 03 2017].

[18] „datasheets.maximintegrated.com,“ 30 11 2016. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. [Přístup získán 30 11 2016].

[19] „reuk,“ 16 12 2016. [Online]. Available: http://www.reuk.co.uk//OtherImages/connect-multiple-ds18b2-i2c-sensors-raspberry-pi.jpg. [Přístup získán 16 12 2016].

[20] „adafruit.com,“ 08 01 2017. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/1899_HTU21D.pdf. [Přístup získán 08 01 2017].

[21] „ebay,“ 06 02 2017. [Online]. Available: http://www.ebay.com/itm/4CH-I2C-AC-Dimmer-Light-Bulb-Smart-Home-Arduino-Raspberry-110V-220V-/122046013103. [Přístup získán 06 02 2017].

[22] „diyhacking,“ 28 12 2017. [Online]. Available: https://diyhacking.com/arduino-lamp-dimmer/. [Přístup získán 28 12 2017].

[23] „hoelscher-hi,“ 10 03 2017. [Online]. Available: http://www.hoelscher-hi.de/hendrik/light/stepper/bistp.gif. [Přístup získán 10 03 2017].

[24] „imgur.com,“ 05 01 2017. [Online]. Available: https://i.stack.imgur.com/tqWmd.png. [Přístup získán 05 01 2017].

[25] „servocity,“ 18 01 2017. [Online]. Available: https://www.servocity.com/media/wysiwyg/cms/support/tech-tips/how-does-a-servo-work/Untitled-165.jpg. [Přístup získán 18 01].

[26] „qt,“ 18 01 2017. [Online]. Available: http://doc.qt.io/qt-5/supported-platforms.html. [Přístup získán 18 01 2017].

[27] „wikipedia.org,“ 06 01 2017. [Online]. Available: https://cs.wikipedia.org/wiki/Raspbian. [Přístup získán 06 01 2017].

[28] „wikipedia.org,“ 24 12 2016. [Online]. Available: https://cs.wikipedia.org/wiki/Raspberry_Pi#Software. [Přístup získán 24 12 2016].

[29] „qt,“ 18 01 2017. [Online]. Available: https://wiki.qt.io/Qt_for_Beginners. [Přístup získán 18 01 2017].

[30] „www.hobrasoft.cz,“ 12 01 2017. [Online]. Available: www.hobrasoft.cz. [Přístup získán 12 01 2017].

[31] „bootstrap,“ 05 12 2016. [Online]. Available: www.getbootstrap.com. [Přístup získán 05 12 2016].

[32] „www.raspberrypi.org,“ 10 12 2016. [Online]. Available: https://www.raspberrypi.org/help/videos/#noobs-setup. [Přístup získán 10 12 2016].

[33] „www.qt.io,“ 28 12 2016. [Online]. Available: www.qt.io. [Přístup získán 28 12 2016].

6 Přílohy

Obsah CD:

- Dokumentace – v této složce je uložena tato práce v pdf formátu.
- Příloha 1 – zde je uložen výkres a 3D modely ozubeného kolečka řetízku žaluzií.
- Příloha 2 – zde se nachází aplikace WebHome se všemi knihovnami a projekty.
- Reference – tento adresář obsahuje použité elektronické zdroje uvedené v kapitole odkazy a reference.

Seznam obrázků