Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Bakalářská práce

Programovatelný stmívač světel pro domácí zvířata

Ondřej Červenka

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

6. května 2015

Poděkování

Děkuji panu Ing. Pavlu Kubalíkovi, Ph.D. za čas, který mi věnoval a zejména za cenné rady a odborné vedení mé bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. května 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Červenka, Ondřej. *Programovatelný stmívač světel pro domácí zvířata.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Práce se zabývá ovládáním světel pomocí programovatelného zařízení. Cílem práce bylo prozkoumant současný stav problematiky a vytvořit řešení umožňující vzdálené ovládání světel vhodných pro domácí zvířata pomocí počítače.

Výsledkem práce je zařízení založené na vývojové desce Arduino Yún. Zařízení umožňuje programovat dvě diody pomocí webového rozhraní. Diody mohou být řízeny časovačem ve formě RTC obvodu nebo fotosenzorem. Je také možno je ovládat pomocí tlačítek na zařízení. Stav zařízení je zobrazován na displej.

Webová stránka umožňuje ovládání a konfiguraci jednotlivých diod a nastavení hodin zařízení.

Klíčová slova Arduino, stmívání světel, webová aplikace

Abstract

This thesis deals with controlling lights via programmable device. Goal was to explore current solutions and create solution that allows remote control of lights that are suitable for pets.

Result is a device based on Arduino Yún developement board. Device allows to program two LED with web interface. Diodes can be controlled by timer in form of RTC circuit or by photosensor. It is also possible to control them with buttons placed on the device.

Web interface allows controlling and configuration of each diode and adjusting RTC clock.

Keywords Arduino, light dimming, web application

Obsah

Odkaz na tuto práci viii

Úvod **1**

1 Analýza a návrh **3**
1.1 Výběr desky 3
1.1.1 Arduino Uno 3
1.1.2 Teensy 3.1 4
1.1.3 Raspberry Pi Model B+ 5
1.1.4 Arduino Yún 6
1.1.5 Shrnutí výběru desky 8
1.2 Výběr zdroje světla 9
1.3 Výběr periferií k desce 9
1.3.1 RTC obvod 9
1.3.2 Displej 9
1.3.3 Senzor světla 10
1.3.4 Tlačítka 11
1.4 Způsob komunikace s PC 11
1.4.1 Sériová linka 11
1.4.2 Webový server na domácí síti 11
1.4.3 Využití služby třetí strany 12
1.5 Shrnutí výběru komunikace 14
1.6 Ukládání nastavení 14
1.7 Závěr analýzy 15
1.8 Seznámení se s deskou Arduino Yún a vývojovým prostředím . 15
1.8.1 Vývojové prostředí Arduino IDE 15
1.8.2 Deska Arduino Yún 17
1.9 Hardwarový návrh zařízení 22
1.10 Softwarový návrh zařízení 22

2 HW Implementace **25**
2.1 Ovládání zářivek . . . 25
2.2 Komunikace s periferiemi . . . 26
2.2.1 Displej . . . 26
2.2.2 RTC obvod . . . 26
2.3 Sestavení zařízení . . . 27
2.3.1 Deska Arduino Yún . . . 27
2.3.2 Displej a RTC . . . 28
2.3.3 Napájení zářivek . . . 28

3 SW Implementace **31**
3.1 Ovládání zářivek . . . 31
3.1.1 Třída *Bulb* . . . 31
3.2 Komunikace s periferiemi . . . 35
3.2.1 Displej . . . 35
3.2.2 RTC obvod . . . 38
3.3 Komunikace zařízení s PC . . . 39
3.3.1 Konfigurace WiFi sítě . . . 40
3.4 Webové rozhraní . . . 42
3.4.1 Zasílání požadavků . . . 43
3.4.2 Struktura stránek . . . 44

4 Testování **47**
4.1 Testy provádněné při sestavování zařízení . . . 47
4.2 Testy periferií . . . 47
4.2.1 Displej . . . 47
4.2.2 RTC obvod . . . 48
4.2.3 Fotosenzory, tlačítka a LED . . . 48
4.3 Test webového rozhraní a komunikace se zařízením . . . 48
4.4 Test ukládání a načítaní konfiguračních souborů . . . 49
4.4.1 Arduino IDE 1.6.3 . . . 49
4.5 Celkový test zařízení . . . 49

Závěr **51**
Budoucí práce . . . 51

Literatura **53**

A Seznam použitých zkratek **55**

B Obsah přiloženého CD **57**

Seznam obrázků

1.1 Deska Arduino Uno 4
1.2 Teensy 3.1 5
1.3 Raspberry Pi Model B+ 6
1.4 Deska Arduino Yún 7
1.5 LCD s I2C modulem 10
1.6 Zapojení fotorezistoru 10
1.7 Komunikace pomocí reverse HTTP 13
1.8 Arduino IDE 16
1.9 Schéma komunikace na desce Arduino Yún 17
1.10 HW blokové schéma 22
1.11 SW blokové schéma 24

2.1 Schéma zapojení LED 25
2.2 I2C piny Arduina 26
2.3 Tištěný spoj s RTC obvodem 27
2.4 Připojení I2C periferií 28
2.5 Konečná podoba zařízení 29

3.1 Zobrazení stavu na displej 35
3.2 Rozhraní pro konfiguraci desky Arduino Yún 42
3.3 Ukázka webového rozhraní stmívače 43

Seznam tabulek

1.1 Porovnání vybíraných desek 8

3.1 Tabulka REST volání 41

Úvod

V současné době se trh s produkty umožňujícími řízené ovládání světel zaměřuje spíše na rozsáhlejší řešení určená pro použití v budovách. Cena těchto systémů se pak pohybuje v řádu desítek tisíc korun.

V této práci jsem se zaměřil na návrh a implementaci cenově dostupného zařízení, určeného predevším pro chovatele domácích mazlíčků. Zařízení umožňuje pokročilejší nastavení světel než standardní produkty určené pro chovatele a zároveň je jednoduché na instalaci a provozování. Cena zařízení by neměla překročit 3000 Kč včetně DPH.

Pro realizaci jsem zvolil oblíbenou platformu Arduino, konkrétně desku Arduino Yún[1], která kromě mikrokontroléru obsahuje Atheros AR9331 SoC. Ten umožní komunikovat s mikrokontrolérem pomocí WiFi nebo Ethernetu.

Hlavní výhodou platformy Arduino je, že již v základu poskytuje množství užitečných knihoven, které jsou díky rozsáhlé komunitě stále rozšiřovány. Je tedy dostupná podpora velkého množství více i méně běžných periferií a služeb. Tyto knihovny jsou k dispozici ve formě zdrojových kódu, takže je možné je dále upravovat pro specifické potřeby aplikace. Další z výhod je rychlé a jednoduché programování desky pomocí USB.

Kapitola 1

Analýza a návrh

Možnosti osvětlení pro domácí zvířata, které jsou dnes na trhu pro chovatele k dispozici, většinou nabízejí pouze jednoduché ovládání světel bez možnosti složitější konfigurace. Pokud bychom chtěli nějaké komplexnější řízení, museli bychom sáhnout spíše po dražších zařízeních určených pro rozsáhlejší domáci automatizaci.

V dnešní době však roste popularita víceúčelových vývojových desek, ke kterým lze připojit velké množství nejrůznějších periferií a ty pak pomocí mikroprocesoru na desce ovládat. Asi nejrozšířenější je platforma Arduino, založená na mikrokontrolérech Atmel, existuje však celá řada dalších přípravků, například desky postavené na platformě PICAXE společnosti Microchip.

Výkonnější desky jako například Beaglebone Black nebo Raspberry Pi, využívající procesory s architekturou ARM, pak mohou sloužit i jako plnohodnotné stolní PC či multimediální centrum.

Taková deska nám poslouží jako základ pro vytvoření stmívače, který bude umožňovat rozsáhlejší nastavení, snadné ovládání a zároveň bude cenově dostupný.

1.1 Výběr desky

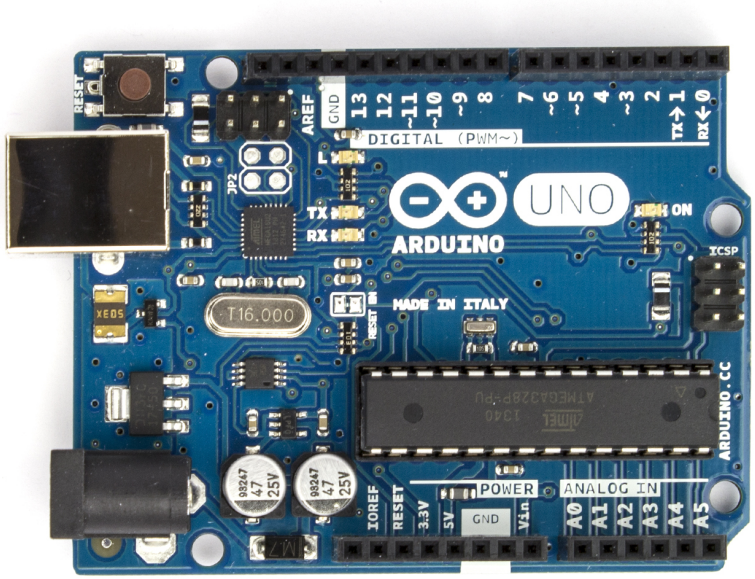
Pro realizaci zařízení jsem vybíral přípravek, který by byl cenově dostupný, umožňoval připojení potřebných periferií (pomocí sběrnice I2C a GPIO pinů) a snadné programování, nejlépe pomocí USB. Další kritérium výběru byla dostupnost knihoven pro ovládání periferních zařízení.

V současné době je na trhu mnoho přípravků a vývojových desek vhodných k realizaci stmívače, k bližšímu prozkoumání jsem vybral následující čtyři.

1.1.1 Arduino Uno

První deska od společnosti Arduino[2], s mikrokontrolérem ATmega328. Má 14 digitálních GPIO pinů, z čehož 6 umožňuje PWM výstup, který lze využít

Obrázek 1.1: Deska Arduino Uno. Obrázek převzat z `http://arduino.cc/en/uploads/Main/ArduinoUno_R3_Front.jpg`



ke stmívání LED. Kromě toho má také 6 analogových vstupů, což usnadňuje čtení fotosenzoru, neboť nejsou potřeba žádné další převodníky.

Mikrokontlér ATmega3228[3] disponuje flash pamětí o velikosti 32 KiB, z čehož 0.5 KiB zabírá zavaděč umožňující programování přes USB. Po smazání zavaděče je možno programovat desku přes ICSP. Dále mikrokontrolér obsahuje 2KiB SRAM a 1KiB EEPROM. Frekvence hodin je 16 MHz.

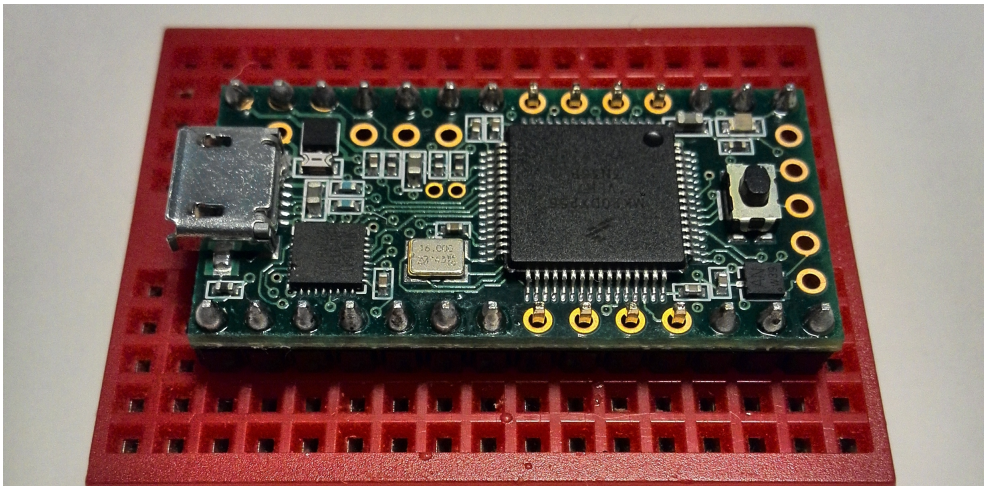
Tato deska umožňujňuje komunikovat s dalšími zařízeními pomocí sběrnic SPI, I2C a sériové linky. Přes sériovou linku je většinou realizována i komunikace s PC. Pokud bychom chtěli nějakou formu pokročilejší komunikace (Bluetooth, WiFi, Ethernet), bylo by nutné přikoupit některý z mnoha podporovaných modulů.

Cena této desky se pohybuje kolem 600 Kč, na trhu je však mnoho kompatibilních klonů, které mohou být i levnější.

1.1.2 Teensy 3.1

Vývojová deska je osazena 32 bitovým procesorem ARM Cortex-M4 a oproti Arduinu Uno poskytuje především mnohem větší výkon. Má 34 GPIO pinů, z nichž 21 umožňuje analogový vstup a 12 PWM výstup. Také zde najdeme sběrnice I2C, SPI, CAN a sériovou linku. Všechny digitální piny mají toleranci na 5 V, deska však využívá 3.3V signály. Na všech pinech lze také využít

Obrázek 1.2: Teensy 3.1



přerušení. Teensy obsahuje také vestavěný RTC obvod, je ale nutné připájet 32 768Hz krystal a 3V baterii.

Programová paměť procesoru činí 256 KiB, RAM 64 KiB. Desku je tedy možné využít i pro rozsáhlejší aplikace, které by například u Arduina Uno představovaly problém. Dále zde najdeme EEPROM o velikosti 2 KiB. Základní frekvence hodin je 72 MHz, v případě potřeby je však možné procesor snadno přetaktovat na 96 MHz.[4]

Deska používá zavaděč HalfKay a je možno ji programovat přímo přes USB konektor. Je také kompatibilní se všemi standardními Arduino funkcemi a množstvím knihoven. Programování desky lze po instalaci pluginu Teensyduino provádět přímo z Arduino IDE.

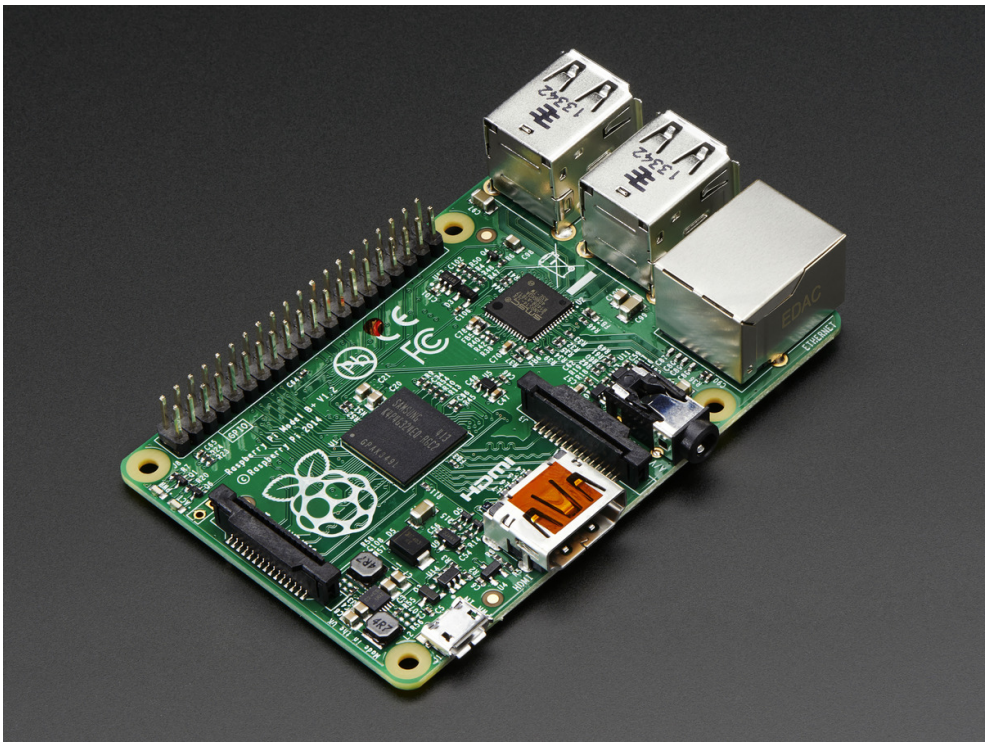
Cena desky je zhruba 750 Kč, dostupnost v České republice je však poměrně omezená, na objednávku je možno ji sehnat třeba na `http://www.snailshop.cz`.

1.1.3 Raspberry Pi Model B+

Vylepšená verze oblíbeného jednodeskového počítače Raspberry Pi, založená na SoC Broadcom BCM2835[5]. Disponuje 40 GPIO piny, z těch však pouze jeden podporuje PWM výstup. Pro více výstupů je potřeba PWM vytvářet softwarově, což by však pro naše potřeby bylo dostačující. Dále Raspberry Pi neobsahuje piny umožňující analogový vstup, takže pro čtení fotosenzoru by bylo nutné použít externí A/D převodník (například MCP3002).

Pro potřeby našeho zařízení je možno využít sběrnice SPI, I2C, sériovou linku a také čtyři USB porty. Počítač mimo to poskytuje i HDMI výstup pro připojení monitoru, ethernetový konektor a 3,5mm jack. Z těch bychom však pravděpodobně využili pouze ethernetový konektor pro komunikaci se stmívačem po síti. Bohužel chybí WiFi přijímač, bylo by jej tedy nutné zakoupit zvlášť a využít jednoho z USB portů.

Obrázek 1.3: Raspberry Pi Model B+. Obrázek převzat z `http://www.adafruit.com/images/1200x900/1914-01.jpg`



Model B+ je osazen procesorem ARM1176JZF-S s frekvencí 700 MHz a disponuje 512 MiB RAM. Procesor podporuje několik operačních systémů, z nichž nejrozšířenější je Raspbian, postavený na linuxové distribuci Debian. Díky přítomnosti operačního systému a potřebných vývojových nástrojů je tedy možné programovat a ladit aplikace přímo na Raspberry Pi. Celý operační systém je umístěn na microSD kartě, což umožňuje systémy v případě potřeby snadno měnit.

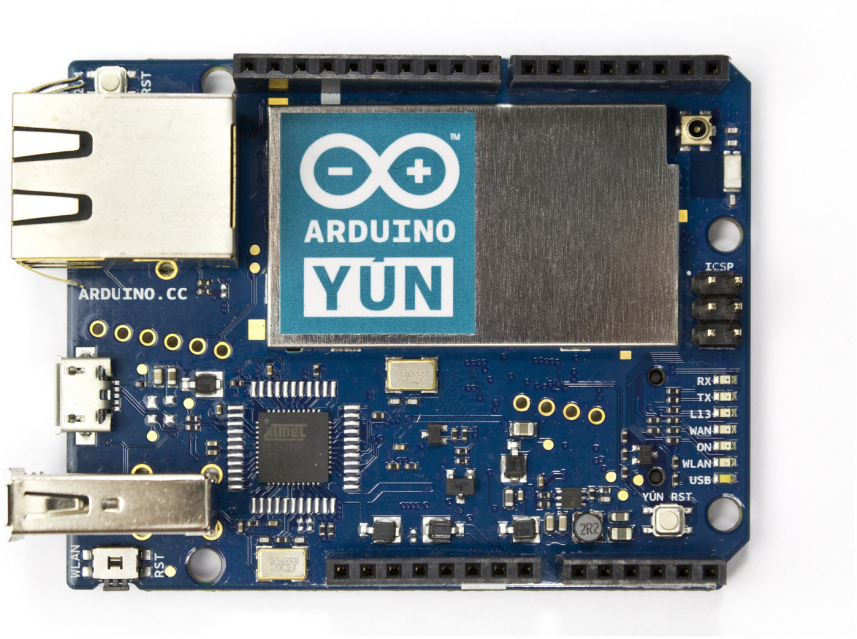
Díky popularitě je stejně jako u platformy Arduino k dispozici pestrá škála knihoven. Za zmínku stojí výrazná podpora jazyka Python, jakožto doporučeného pro vývoj na platformě Raspberry Pi, kromě něj lze však aplikace vyvíjet i v jazycích C, C++, Java a dalších.

Cena Raspberry Pi Model B+ se v současné době pohybuje zhruba kolem 1200 Kč.

1.1.4 Arduino Yún

Deska Arduino Yún kombinuje mikrokontrolér Atmel ATmega32U4[6], známý z desky Leonardo, a SoC Atheros AR9331[7], používaný mimo jiné v routerech a dalších síťových prvcích.

Obrázek 1.4: Deska Arduino Yún. Obrázek převzat z `http://arduino.cc/en/uploads/Main/ArduinoYunFront_2.jpg`



Deska poskytuje 20 GPIO pinů, z toho 7 umožňujících PWM výstup a 12 analogový vstup.

Vedle sběrnic SPI, I2C a sériové linky nabízí deska i ethernetový port, slot pro microSD kartu a jeden USB port. Výhodou oproti Raspberry Pi je přítomnost WiFi přijímače a vysílače. Vysílač se nám hodí, pokud bychom chtěli se stmívačem komunikovat bezdrátově v místě, kde není k dispozici WiFi síť.

V systému Atheros AR9331 je použit MIPS procesor s frekvenci 400 MHz. Dále má k dispozici 64 MiB RAM a 16 MiB flash paměti sloužící jako úložiště pro operační systém, v případě nedostatku místa je však možné systém přesunout na microSD kartu stejně jako u Raspberry Pi. Jako operační systém slouží linuxová distribuce OpenWrt.

ATmega32U4 disponuje 32 KiB flash paměti (narozdíl od desky Arduino Uno zde zavaděč zabírá 4 KiB), 2,5 KiB SRAM a 1 KiB EEPROM pro prezistentní ukládání dat. Frekvence hodin mikrokontroléru je 16 MHz.

Komunikace mezi mikrokontrolérem a linuxovou částí je realizována pomocí sériové linky a knihovny *Bridge*, dodávané Arduinem. Ta implementuje mnoho užitečných funkcí pro pokročilejší komunikaci s linuxovou částí a síť. Yún také podporuje všechny standardní knihovny pro platformu Arduino.

Cena desky Arduino Yún se pohybuje v rozmezí od 1800 do 2100 Kč.

	Arduino Uno	**Teensy 3.1**	**Raspberry Pi B+**	**Arduino Yún**[1]
Procesor	ATmega328	ARM Cortex-M4	BCM2835 SoC	AR9331 SoC (ATmega32U4)
Flash	32 KiB	256 KiB	SD karta	16 MiB (32 KiB)
RAM	2 KiB	64 KiB	512 MiB	64 MiB (2,5 KiB)
frekvence	16 MHz	72 MHz (96 MHz)[2]	700 MHz	400 MHz (16 MHz)
GPIO	14	34	40	20
Periferie	I2C, SPI, UART, PWM, analogový vstup	I2C, SPI, UART, CAN, PWM, analogový vstup	I2C, SPI, UART, PWM, USB, SD, Ethernet, HDMI, 3,5mm jack	I2C, SPI, UART, PWM, analogový vstup, USB, SD, Ethernet, WiFi
OS	-	-	Raspbian a další	OpenWrt
Cena	600 Kč	750 Kč	1200 Kč	1800 až 2100 Kč

Tabulka 1.1: Porovnání vybíraných desek

1.1.5 Shrnutí výběru desky

Pokud bychom počítali s komunikací s PC pouze po sériové lince, bylo by Arduino Uno i Teensy naprosto dostačujícím řešením. Deska Teensy navíc zaujme výkonným ARM procesorem a vyšší pamětí.

Pokud však chceme implementovat i složitější ovládání, například přes webové rozhraní, bylo by nutné použít nějaký vhodný modul. Arduino i další výrobci nabízí moduly (shieldy) realizující připojení k síti přes WiFi či Ethernet určené přímo pro desku Uno, jsou však poměrne drahé, navíc bychom si museli vybrat mezi připojením přes Ethernet nebo přes WiFi.

Raspberry Pi i Arduino Yún jsou vhodné pro realizaci stmívače ovládaného pomocí webového rozhraní či s využitím nějaké služby třetí strany a internetu. Vyšší cena desky Yún je částečně vyvážena tím, že není nutné dokupovat WiFi přijímač a A/D převodníky. Oproti tomu výhody Raspberry Pi nad Arduinem (výkonější procesor, HDMI výstup, více USB portů) jsou pro nás vcelku nepodstatné.

Řekl bych tedy, že obě desky se pro realizaci stmívače hodí zhruba stejně. Já jsem vybral Arduino Yún, a to z především z toho důvodu, že mne zaujala kombinace mikrokontroléru a linuxového prostředí.

[1] Hodnoty pro mikrokontrolér ATmega324U jsou uvedeny v závorce.
[2] Po přetaktování.

1.2 Výběr zdroje světla

Jako zdroj světla nám poslouží výkonové LED, které lze snadno stmívat pomocí PWM výstupů desky. Zvolené diody mají výkon 1 W, průrazné napětí 350 mA a průrazný proud 3 V[8]. V případě potřeby by bylo možné použít i silnější LED.

Protože jednotlivé piny Arduina mohou poskytovat maximálně 40 mA proudu, není možné dosáhnout maximálního jasu při připojení diod přímo k pinům. K napájení LED je tedy nutné použít externí zdroj.

Samotné stmívání se realizuje pomocí tranzistoru, řízeného PWM výstupem desky. Zde jsem zvolil tranzistorové pole ULN2003A[9], umožňující spínat až 500 mA na kanál. Pole obsahuje 7 Darlingtonových tranzistorů, já však využiji pouze dva, pro ovládání dvou nezávislých diod. Výhodou je přítomnost 2.7kΩ resistorů na vstupních pinech tranzistorů, což umožní připojit přímo PWM výstup Arduina bez dalších součástek.

1.3 Výběr periferií k desce

Kromě samotných LED bude zařízení využívat RTC obvod umožňující nastavení času stmívaní a rozsvěcení světel, fotorezistory umožňující reakci na okolní osvětlení, tlačítka pro manuální ovládání diod a displej zobrazující stav zařízení.

1.3.1 RTC obvod

Bohužel kromě Teensy 3.1 ani jedna z desek, ze kterých jsem vybíral, neobsahuje RTC obvod. U Arduina Yún a Raspberry Pi se předpokládá, že čas se bude udržovat a synchronizovat pomocí internetu. Pokud bychom však chtěli zařízení, které může fungovat nezávisle na připojení k internetu, musíme použít nějaký externí hodinový obvod.

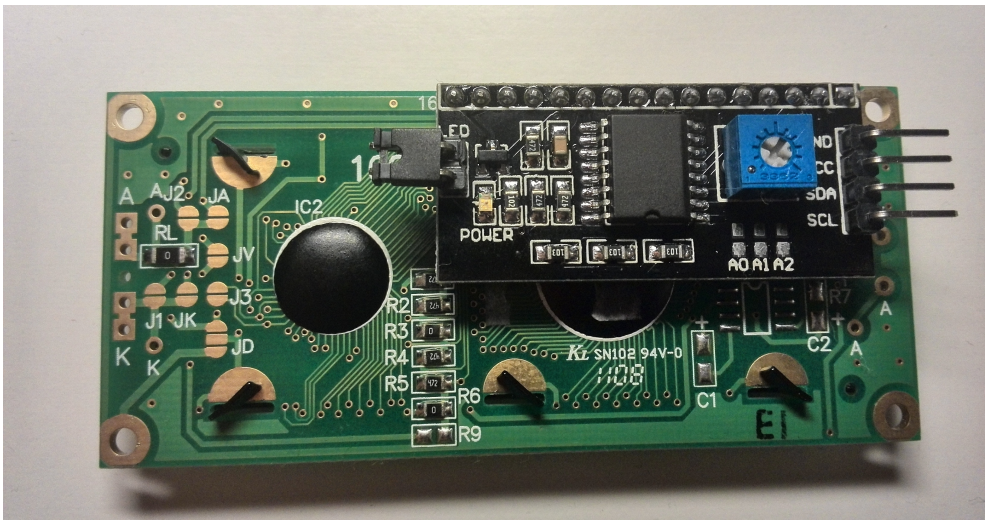
Já jsem zvolil obvod DS1307 s externím krystalem, který je cenově velmi dostupný a jeho přesnost je pro naše potřeby dostačující. Obvod dovoluje připojení záložní baterie, není tedy nutné ho při každém odpojení zařízení od napájení seřizovat. Komunikace s deskou je realizována pomocí sběrnice I2C.

K obvodu existuje knihovna *RTClib*, určená pro Arduino. Knihovna je volně dostupná na `https://github.com/adafruit/RTClib` a dodává všechny potřebné funkce pro čtení a seřizování obvodu, a také třídu DateTime jako jednoduchý kontejner pro čas a datum.

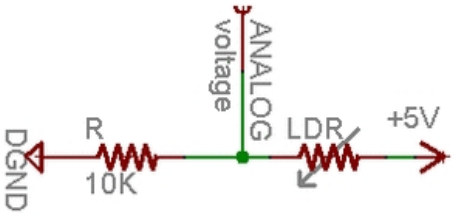
1.3.2 Displej

Jako displej nám postačí standardní LCD s 16x2 znaky, s řadičem Hitachi HD44780. Ten však pro ovládání vyžaduje buď sedm nebo jedenáct pinů (podle toho zda bude operovat ve čtyřbitovém nebo osmibitovém módu)[10]. Proto

Obrázek 1.5: LCD s I2C modulem



Obrázek 1.6: Zapojení fotorezistoru. Obrázek převzat z: `https://learn.adafruit.com/assets/458`



jsem se rozhodl využít I2C modul[11], díky kterému se počet potřebných pinů sníží na dva (ve skutečnosti však nebudou potřeba žádné piny navíc, neboť piny I2C sběrnice stejně využiji pro připojení RTC obvodu). Modul také dovoluje ovládat kontrast displeje pomocí potenciometru.

K modulu je dodávána knihovna pro Arduino, implemetující všechny funkce potřebné pro ovládání displeje. Knihovna je postavena nad oficiální Arduino knihovnou *LiquidCrystal* a je opět volně dostupná na `www.geeetech.com/Documents/LiquidCrystal_I2Cv1-1.rar`.

1.3.3 Senzor světla

Pro detekci intenzity světla je možno využít například fotorezistor. Jednu stranu rezistoru připojíme k napájecímu (5 V) pinu Arduina a druhou přes pull-down rezistor k uzemňovacímu pinu. Poté už stačí jen připojit místo, kde je fotorezistor napojen na pull-down rezistor, k jednomu z analogových vstupů desky[12].Úroveň intenzity světla je určena změnou napětí vytvořenou

snížením či zvýšením odporu fotorezistoru při změně osvětlení.

Čtení hodnoty senzoru lze realizovat pomocí funkce *analogRead*(), která přečte požadovaný analogový vstup a vrátí hodnotu z intervalu <0, 1023>. V tomto případě bude platit, že čím vyšší hodnota, tím vyšší intenzita světla.

1.3.4 Tlačítka

Zařízení bude dovolovat jednoduché ovládání pomocí dvou tlačítek, z nichž každé rozsvěcí/zhasína jednu zářivku. Ovládání tlačítky bude nezávislé na hodnotě fotorezistoru (zhasne/rozsvítí se bez ohledu na intenzitu světla).

Tlačítka je možno připojit přes 10kΩ pull-down rezistor k libovolnému digitálnímu pinu desky a číst pomocí funkce *digitalRead*(). Bylo by také možné využít externího přerušení, které umožňují piny číslo 2 a 3, pro naše potřeby však postačí kontrolovat stav tlačítek v hlavním cyklu programu.

1.4 Způsob komunikace s PC

Stmívač bude také možno ovládat a nastavovat pomocí PC. Komunikace s PC může být realizována mnoha způsoby, s využitím sériové linky, WiFi, Ethernetu či Bluetooth. V další části jsem se zaměřil na ty, které deska Arduino Yún podporuje bez nutnosti instalace dalších modulů.

1.4.1 Sériová linka

Nejjednodušší způsob komunikace je přes sériovou linku. Tu je možno na desce realizovat buď pomocí samostatných pinů RX a TX nebo přes micro USB konektor. Pro připojení k počítači tak stačí pouze micro USB kabel a není potřeba další redukce. Na straně desky Arduino poskytuje knihovnu *Serial*, obsahující funkce pro inicializaci, čtení a zápis na sériovou linku.

Na straně PC by bylo možné s deskou komunikovat pomocí terminálu (například PuTTy), toto řešení však není příliš uživatelsky přívětivé. Další možností je tvorba grafické aplikace, která by komunikaci obstarala. Potřebné rozhraní je k dispozici například v jazycích Java nebo Python, se kterými by bylo možné vytvořit příjemné grafické prostředí pro ovládání stmívače.

Vzhledem k volbě desky by však bylo škoda nevyužít sofistikovanější možnosti komunikace s PC, nicméně sériová linka nám stále dobře poslouží pro ladění programu. Dále je sériová linka využita pro komunikaci mirkoknotroléru s linuxovým prostředím, od tohoto jsme však odstíněni knihovnou *Bridge*.

1.4.2 Webový server na domácí síti

Další možnost je využít webový server běžící na desce. Ten primárně slouží pro nastavení připojení desky k síti, je však možno na něj nahrát i další webové

stránky. Můžeme zde tedy provozovat webové rozhraní pro ovládání stmívače, které bude po připojení do domácí sítě dostupné ze všech počítačů v ní.

Toto řešení má oproti předchozímu několik výhod. Asi nejvýznamnější je možnost ovládat zařízení bezdrátově ze kteréhokoliv PC v domácnosti. Dále odpadá nutnost instalovat speciální aplikaci, ke všemu postačí pouze prohlížeč.

V případě, že domácí síť není k dispozici, je možno buď připojit zařízení přímo k PC pomocí ethernetového kabelu nebo využít WiFi vysílače na desce. Ten umožní Arduinu vytvořit vlastní WiFi síť, ze které lze přistupovat k webovému rozhraní pro ovládání stmívače.

Pro komunikaci s mikrokontrolérem po síti jsou Arduinem dodávány knihovny *YunClient* a *YunServer*. Ty nám umožní přijímat požadavky klientů ve formátu REST volání[13] a odpovídat na ně přímo v prostředí mikrokontroléru.

1.4.3 Využití služby třetí strany

K ovládnání stmívače by bylo také možné využít některou ze služeb, která by umožnila komunikovat se zařízením prostřednictvým internetu. Výhodou tohoto řešení je možnost přístupu k zařízení i mimo domácí síť, například z telefonu. Nevýhoda je nutnost připojit k internetu jak Arduino Yún, tak zařízení, kterým chceme stmívač ovládat. Také je nutné mít vytvořený účet pro danou službu.

Zde jsou dvě ze služeb, které by bylo možno pro komunikaci se zařízením použít.

1.4.3.1 Amazon Web Services `https://aws.amazon.com`

AWS je cloudová služba společnosti Amazon. Poskytuje nejrůznější služby jako například úložiště dat či virtuální servery. Pro nás je zajímavá především služba SQS (Simple Queue Service), určená pro posílání zpráv mezi zařízeními.

Služba je realizována jako jednoduchá fronta, kde jsou uchovávány zprávy do té doby, dokud nedojde k jejich zpracování či nevyprší jejich životnost [14].

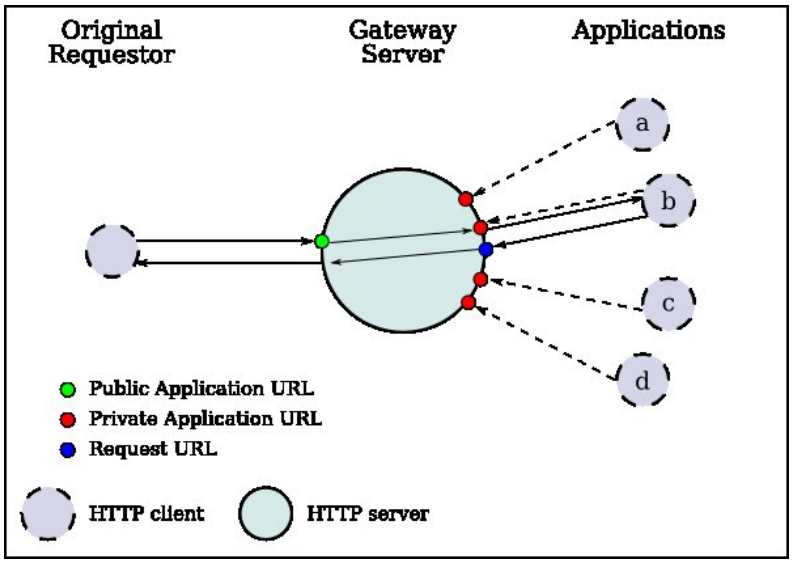
Přístup k frontě je řízen pomocí služby Amazon IAM (Identity and Access Managment). Ta nám umožní vytvořit uživatele a nastavit jim příslušná práva. Uživatel je identifikován pomocí klíče a bezpečnostního kódu.

Samotnou komunikaci mezi Arduinem a AWS je možno zajistit pomocí knihovny *boto*, implementované v Pythonu. Pomocí jednoduchého skriptu je pak možné číst zprávy z fronty a přeposílat je na mikrokontrolér.

Zasílání zpráv do fronty lze realizovat pomocí webové stránky hostované na Amazon Simple Storage Service. Ta může přistupovat k AWS API s použitím JavaScriptu. Lze také umožnit autentizaci uživatele pomocí Google+ či Facebook účtu.[14]

Služba SQS je zdarma první měsíc pro jeden milion požadavků. Poté je nutné za každý milion požadavků měsíčně zaplatit 0,5 dolaru.

Obrázek 1.7: Komunikace pomocí reverse HTTP. Obrázek převzat z: `http://reversehttp.net/overview.svg`



1.4.3.2 Yaler `https://yaler.net`

Služba Yaler je založena na takzvaném reverse HTTP. To nám umožňuje komunikovat mezi dvěma HTTP klienty pomocí REST volání, podobně jako by spolu komunikovaly klient a server.

Účastníci protokolu jsou aplikace a gateway server. Aplikace je HTTP klient, který chce pomocí gateway serveru reagovat na požadavky uživatelů. V našem případě by se jednalo o stmívač. Gateway server je HTTP server, zde provozovaný společností Yaler, který využijeme k přeposílaní požadavků aplikaci a vracení odpovědi uživatelům.[15]

Pro používání služby je třeba se registrovat na stránkách yaler.net a vytvořit si takzvanou relay doménu. Ta je využita pro identifikaci na straně Yaler serveru. Samotnou komunikaci uživatele se zařízením zajistí skript v Pythonu, spuštěný na linuxové části Arduina. Ten přijme požadavek z gateway serveru, předá ho mikrokontroléru pomocí sériové linky a případnou odpověď pošle zpět na gateway server, který ji předá klientovi. Autorem skriptu je Bo Peterson[16], a skript spolu s ukázkovým příkladem je dostupný na `https://github.com/bopeterson/yunYaler`.

Z uživatelského hlediska je tedy možné se zařízením komunikovat pomocí REST volání ve tvaru:

```
1 http://try.yaler.net/relay-domena/arduino/prikaz/param1/param2
```

Na straně mikrokontroléru dojde k předání řetězce, ze kterého je možné vyparsovat jaký příkaz, a případně s jakými parametry, se má provést. Zároveň je možné pomocí funkce *Bridge.put*() předat linuxové straně řetězec, který

bude odchycen skriptem a přes gateway server předán klientovi jako odpověď na požadavek.

Vzhledem k tomu, že všechny odpovědi klientovi vznikají na mikrokontroléru, není možné touto cestou dostat k uživateli například celou HTML stránku. Pro komfortnější ovládání zařízení by tedy bylo třeba vytvořit aplikaci umožňující posílání příkazů a zpracování odpovědí.

V době testování této možnosti komunikace (léto 2014) byla služba Yaler v omezeném rozsahu (limitovaný počet relay domén na účtu) k dispozici zdarma, nicméně dnes je po uplynutí třicetidenní zkušební lhůty placená. Základní verze s jednou relay doménou a podporou přístupu přes web nebo SSH stojí 10 švýcarských franků (asi 250 korun) bez DPH ročně.

1.5 Shrnutí výběru komunikace

Z výše uvedených možností jsem se nakonec rozhodl pro komunikaci pomocí webové stránky provozované přímo na desce. Toto řešení sice neumožní přístup k zařízení mimo domácí síť, to nám ale vzhledem k jeho povaze až tolik nevadí. Důležitější je možnost provozovat zařízení i v místech bez přístupu k internetu (například chaty), což by nám komunikace pomocí služeb třetích stran neumožnila. Další výhodou je funkčnost bez nutnosti registrace, dalšího nastavování a případně i bez poplatků za využívání služby.

Nicméně jsem zkusil zařízení částečně implementovat také pomocí služby Yaler a velmi mě zaujala, především rychlostí zpracování požadavků. V případě potřeby komunikovat se zařízením mimo domov (například ovládání termostatu pomocí mobilního telefonu) by se jednalo o zajímavou možnost řešení.

1.6 Ukládání nastavení

Zařízení by také mělo být schopno uchovat nastavení stmívače v případě odpojení od napájení. Toho lze docílit dvěma způsoby.

První možnost je využít EEPROM na mikrokontroléru ATMega32u4. Ta má velikost 1 KiB, což by pro naše potřeby bylo dostačující. K Arduinu je pro tyto potřeby dodávána knihovna obsahující funkce *read*() a *write*(), umožňující čtení a zápis jednoho bytu na danou adresu paměti. Nevýhodou tohoto řešení je omezený počet zápisů EEPROM, udávaná životnost paměti je sto tisíc zápisů/mazání[17].

Takovéto množství by reálně nemělo představovat problém, přesto mi přišlo jako lepší řešení využít linuxové strany desky a SD karty. Pomocí knihovny *FileIO* je totiž možno přistupovat k filesystému OpenWrt přímo z mikrokontroléru. To nám dovolí vytvářet, číst a zapisovat do konfiguračních souborů stmívače uložených na SD kartě.

Samotný konfigurační soubor bude obsahovat data o nastavení jednotlivých zářivek, tedy čas rozsvěcení a zhasínaní, příznaky pro reakci na časovač, na

senzor intenzity světla a práh reakce na tento senzor. Čtení souboru proběhne na záčátku při inicializaci programu stmívače, zápis při každé změně nastavení.

1.7 Závěr analýzy

Jako základ pro naše zařízení použijeme desku Arduino Yún. Ta se postará o obsluhu periferií i komunikaci s uživatelem. Deska bude doplněna o RTC obvod, LCD displej, na kterém bude zobrazen stav zařízení, a SD kartu pro uložení konfigurace stmívače a webového rozhraní. Dále budou přidána tlačítka a fotosenzory sloužící k ovládání zářivek.

Zdrojem světla budou dvě výkonnové LED. Každou z nich bude možno ovládat nezávisle a bude moci být řízena fotosenzorem a časovačem využívajícím RTC. Ruční ovládání zářivek bude možné pomocí tlačítek nebo ve webovém rozhraní.

Ovládání a nastavování stmívače bude probíhat přes statické webové stránky s využitím Javascriptu. Stránky budou provozovány na webovém serveru běžícím na linuxové straně desky Arduino Yún a budou tedy přístupné z každého počítače připojeného do stejné sítě. Díky tomu bude možno zařízení umístit na jedno místo a pohodlně ovládat z celé domácnosti.

1.8 Seznámení se s deskou Arduino Yún a vývojovým prostředím

Před samotnou implenetací stmívače je dobré blíže se seznámit s možnostmi, které nám deska a vývojové prostředí Arduino IDE nabízí.

1.8.1 Vývojové prostředí Arduino IDE

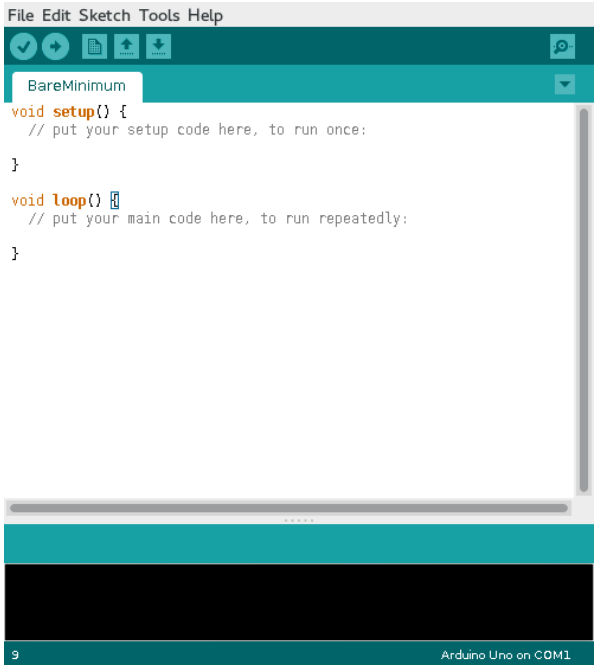
Pro vytváření programů na Arduino deskách bylo vytvořeno vývojové prostředí Arduino IDE. S jeho pomocí je také možné programy nahrávat na mikrokontroléry na deskách. V této práci bylo využíváno především IDE verze 1.5[1], které přináší podporu pro desku Yún. Kromě nástrojů pro kompilaci a nahrávání programu IDE obsahuje další užitečné funkce, jako například terminál pro komunikaci po sériové lince či správu knihoven.

Co se týče samotného psaní kódu, není Arduino IDE příliš komfortní, především pro rozsáhlejší programy. Je zde však možnost využít jiný externí editor (například Sublime Text či Vim) a vývojové prostředí Arduina použít pouze pro kompilaci a nahrání programu.

Samotný program je psán v C/C++, struktura je však mírně odlišná, než jsme zvyklí. Místo funkce *main*() jakožto vstupního bodu tvoří program funkce

[1]Při kompilaci programu v IDE verze 1.6.3 jsem narazil na potíže s některými funkcemi, tato chyba je popsána v sekci 4.4.1.

Obrázek 1.8: Arduino IDE



setup(), volaná jednou při jeho spuštění a funkce *loop*(), tvořící nekonečnou smyčku pro běh programu:

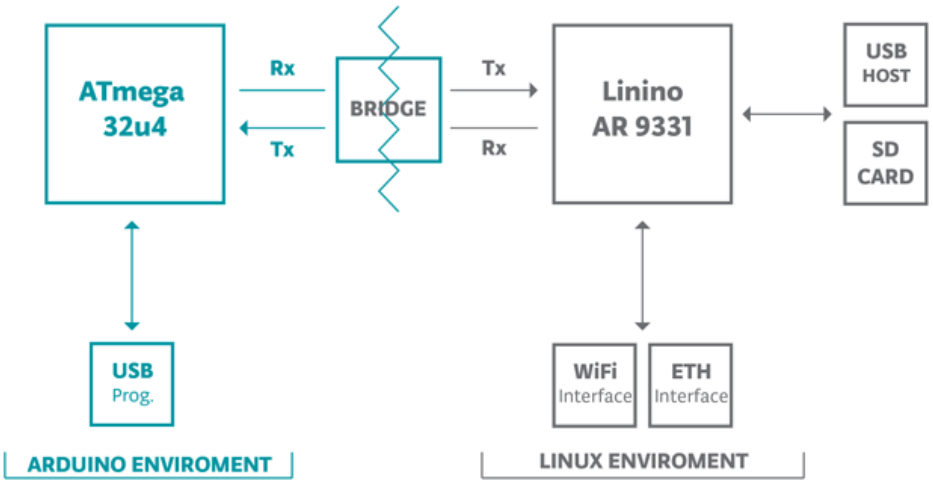
```
void setup() {
  // inicializace programu
  // spusteno jednou
}

void loop() {
  // hlavni smycka programu
}
```

Jde tedy o obdobu

```
int main()
{
  //inicializace programu
  while(1)
  {
    //hlavni smycka programu
  }
  return 0;
}
```

pokud bychom program vyvíjeli v C například s pomocí Atmel Studia. To je ostatně také možné, protože v našem případě stále jde o desku s mikro-

Obrázek 1.9: Schéma komunikace na desce Arduino Yún. Obrázek převzat z: `http://arduino.cc/en/uploads/Main/BridgeInShort.png`



kontrolérem značky Atmel, nebudeme však mít přístup k Arduino knihovnám, které vývoj výrazně usnadňují.

1.8.1.1 Nahrávání programu

Standardně jsou desky Arduino programovány pomocí sériové linky. Deska Yún však dovoluje využít k programování WiFi rozhraní. To oproti sériové lince poskytuje tu výhodu, že kromě programu pro mikrokontrolér může být na desku nahrána také stránka asociovaná s programem (webové rozhraní pro ovládání či zobrazování hodnot senzorů, stránka s informacemi o zařízení atd.).

V projektu je nutno vytvořit složku `www`, do které se uloží zdrojový kód stránky. K samotné desce je pak nutno připojit SD kartu, a na té v kořenovém adresáři vytvořit složku `arduino` s podsložkou `www`. Zde se pak při nahrávání vytvoří složky k jednotlivým projektům. Poté je automaticky vytvořen link `sd`, umístěný ve složce `www` v kořenovém adresáři systému OpenWrt, kde se nachází webové rozhraní pro konfiguraci desky. To zajistí, že stránky pro náš projekt jsou dostupné na adrese `arduino.local/sd/nazevProjektu`, v případě stmívače tedy `arduino.local/sd/ArduinoWebDimmer`. V případě, že chceme od sebe odlišit více desek, je možno název `arduino` změnit.

1.8.2 Deska Arduino Yún

Jak již bylo řečeno, deska je založena na propojení dvou odlišných prostředí, a to systému na čipu Atheros AR9331 s linuxovou distribucí OpenWrt a

mikrokontroléru ATMega32u4. Hardwarově je propojení těchto částí rešeno pomocí sériové linky, softwarově se o něj stará knihovna *Bridge*. Ta poskytuje oboustrannou komunikaci s OpenWrt pomocí funkcí *get*(), *put*() a $transfer()$. Zprávy zaslané knihovnou jsou na linuxové straně interpretovány pomocí Python skriptu.[18]

Je tedy vidět, že knihovna neposkytuje rozsáhlou sadu funkcí, slouží totiž pouze jako základ pro další Arduino knihovny. Ty pak přidávají složitější funkce, jako je spouštení linuxových příkazů, správa souborového systému nebo emulace terminálu. Pro implementaci stmívače nás nejvíce budou zajímat knihovny $YunClient$[19], $YunServer$[20] a $FileIO$[21].

1.8.2.1 OpenWrt

Jak již bylo zmíněno, na desce běží linuxová distribuce OpenWrt. U našeho zařízení bude sloužit především jako zprostředkovatel síťové komunikace pomocí REST volání. Dále právě zde poběží web server s rozhraním pro ovládání stmívače a na připojenou SD kartu budou ukládány konfigurační soubory stmívače. Konfiguraci systému je možné provést buď pomocí terminálu a SSH protokolu, nebo využít webové rozhraní $LuCi$ umožňující snadné nastavení všech potřebných parametrů.

OpenWrt využívá package manager *opkg*, je také možné doinstalovat manager *pip* pro instalaci Python balíčků. Pro potřeby stmívače si však vystačíme s již nainstalovanými programy.

Jako úložište pro systém slouží flash paměť o velikosti 16 MiB, z čehož instalace OpenWrt zabere asi 9 MiB. Zbytek by pro nás mohl být dostačující, je však vhodnější využít SD kartu, neboť ta nám poskytne místo, kam můžeme snadno nahrávat webové stránky přímo pomocí Arduino IDE a také nám dovolí využít knihovnu $FileIO$.

Pomocí návodu dostupného na stránkách společnosti Arduino[22] je možné kartu rozdělit na dvě části. Jedna bude naformátována na systém FAT32 a poslouží jako úložiště pro naše data. Druhá pak jako filesystem pro OpenWrt, což by v případě rozšiřování funkcí zařízení dovolilo instalaci potřebných balíčků bez obav o vyčerpání místa.

1.8.2.2 $YunServer$

Implementace třídy $YunServer$, poskytující metody pro přijímaní klientů. Zde nás budou zajímat metody *accept*() a $listenOnLocalHost()$.

Metoda $YunServer.accept()$ Přijme dalšího klienta serveru.

Parametry:

Žádné.

Návratová hodnota:
Instance třídy *YunClient*, umožňující komunikaci s klientem. Pokud není žádný k dispozici, instance vrátí false při testování v boolovském kontextu.

Metoda *YunServer.listenOnLocalHost*() Nastaví server, aby očekával příchozí spojení. Server komunikuje na portu 5555 na localhostu.

Parametry:
Žádné.

Návratová hodnota:
Žádná.

1.8.2.3 *YunClient*

Knihovna implementuje třídu *YunClient* určenou pro komunikaci s klienty webového serveru. My zde využijeme metody *readStringUntil*(), *print*() a *parseInt*(), zděděné ze třídy *Stream* a také metodu *stop*().

Metoda *YunClient.readStringUntil*(*terminator*) Metoda čte data ze vstupního streamu, dokud nenarazí na ukončovací znak nebo nevyprší časový limit (nastavitelný pomocí *YunClient.setTimeout*()).

Parametry:
terminator Ukončovací znak.

Návratová hodnota:
Datový typ *string*, obsahující řetězec znaků načtených ze vstupního streamu až do detekce ukončovacího znaku.

Metoda *YunClient.print*(*data*, *base*) Zapíše data do výstupního streamu klienta.

Parametry:

data Data určená k zapsání. Může se jednat o datový typ *char*, *byte*, *int* nebo *string*. Čísla jsou zapsána jako řetězce číslic.

base (nepovinný) Soustava, která má být použita pro zápis čísel (`BIN`, `OCT`, `DEC`, `HEX`). Defaultně je nastaveno `DEC`.

Návratová hodnota:
Kolik bytů bylo odesláno.

Metoda *YunClient.parseInt*() Hledá další platný *int* v datovém streamu. Pokud takový není nalezen do 1 sekundy (nastavitelné pomocí *YunClient.setTimeout*()), vrátí 0.

Parametry:
Žádné.

Návratová hodnota:
Další integer z datového streamu nebo 0, pokud žádný nebyl nalezen.

Metoda *YunClient.stop*() Odpojí klienta od serveru.

Parametry:
Žádné.

Návratová hodnota:
Žádná.

1.8.2.4 *FileIO*

Tato knihovna dovoluje přistupovat k souborovému systému SD karty připojené k desce přímo z prostředí mikrokontroléru. Přístup je realizován pomocí tříd *FileSystem* a *File*. Tyto třídy implementují všechny standardní souborové operace, nás však bude zajímat především kontrola existence souboru, otevření a zavření souboru, a čtení a zápis dat.

Na to využijeme metody *exists*() a *open*() třídy *FileSystem* a metody *print*(), *read*() a *close*() třídy *File*.

Metoda *FileSystem.exists*(*filename*) Otestuje, zda daný soubor nebo adresář na SD kartě připojené k linuxové části desky existuje.

Parametry:
filename Název souboru.

Návratová hodnota:
Boolovská hodnota, true pokud soubor existuje, jinak false.

Metoda *FileSystem.open*(*filename*, *mode*) Otevře soubor umístěný na SD kartě. Pokud tento soubor neexistuje, vytvoří ho.

Parametry:

filename Název souboru.

mode (nepovinný) Určuje mód otevření souboru, možnosti jsou `FILE_READ` a `FILE_WRITE`. Defaultní možnost je `FILE_READ`.

Návratová hodnota:
Instance třídy *File*, odkazující na otvíraný soubor. Pokud se soubor nepodaří otevřít, tato instance bude vyhodnocena jako false v boolovském kontextu.

Metoda *File.print*(*data*, *base*) Zapíše data do souboru.

Parametry:

data Data určená k zapsání. Může se jednat datový typ *char*, *byte*, *int* nebo *string*. Čísla jsou zapsána jako řetězce číslic.

base (nepovinný) Soustava, která má být použita pro zápis čísel (`BIN`, `OCT`, `DEC`, `HEX`). Defaultně je nastaveno `DEC`.

Návratová hodnota:
Kolik bytů bylo do souboru zapsáno.

Metoda *File.read*() Čte data ze souboru.

Parametry:
Žádné.

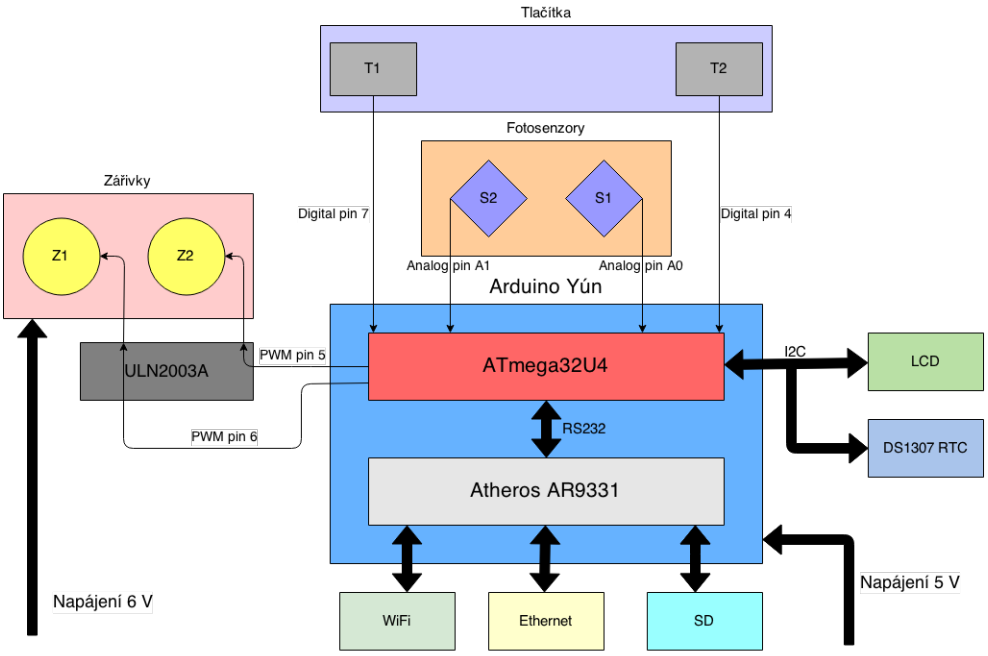
Návratová hodnota:
Další čtený byte nebo -1, pokud již nejsou žádná data k dispozici.

Metoda *File.close*() Zavře soubor a zajistí, že data jsou fyzicky zapsána.

Parametry:
Žádné.

Návratová hodnota:
Žádná.

Obrázek 1.10: HW blokové schéma.



1.9 Hardwarový návrh zařízení

Hardwarově bude zařízení postaveno především na spolupráci mikrokontroléru ATmega32U4 a SoC Atheros AR9331, komunikujících pomocí sériové linky.

Mikrokontrolér obstará obsluhu periferií, čtení stavu tlačítek a fotosenzorů, a také konfiguraci a ovládání zářivek. Všechny periferie budou připojeny pomocí GPIO pinů desky Yún. RTC obvod a displej budou komunikovat po I2C sběrnici a využijí tedy piny *sda* a *scl*.

Fotosenzory budou připojeny k pinům umožňující analogový vstup, tlačítka k digitálním pinům. Piny s PWM výstupem budou připojeny k vstupním pinům obvodu ULN2003A a budou tak ovládat jednotlivé zářivky.

AR9331 zajistí připojení k síti pomocí WiFi nebo Ethernetu, v případě potřeby také poslouží jako WiFi hotspot. Je k němu též připojena SD karta sloužící jako úložiště pro konfigurační soubory a webové stránky.

1.10 Softwarový návrh zařízení

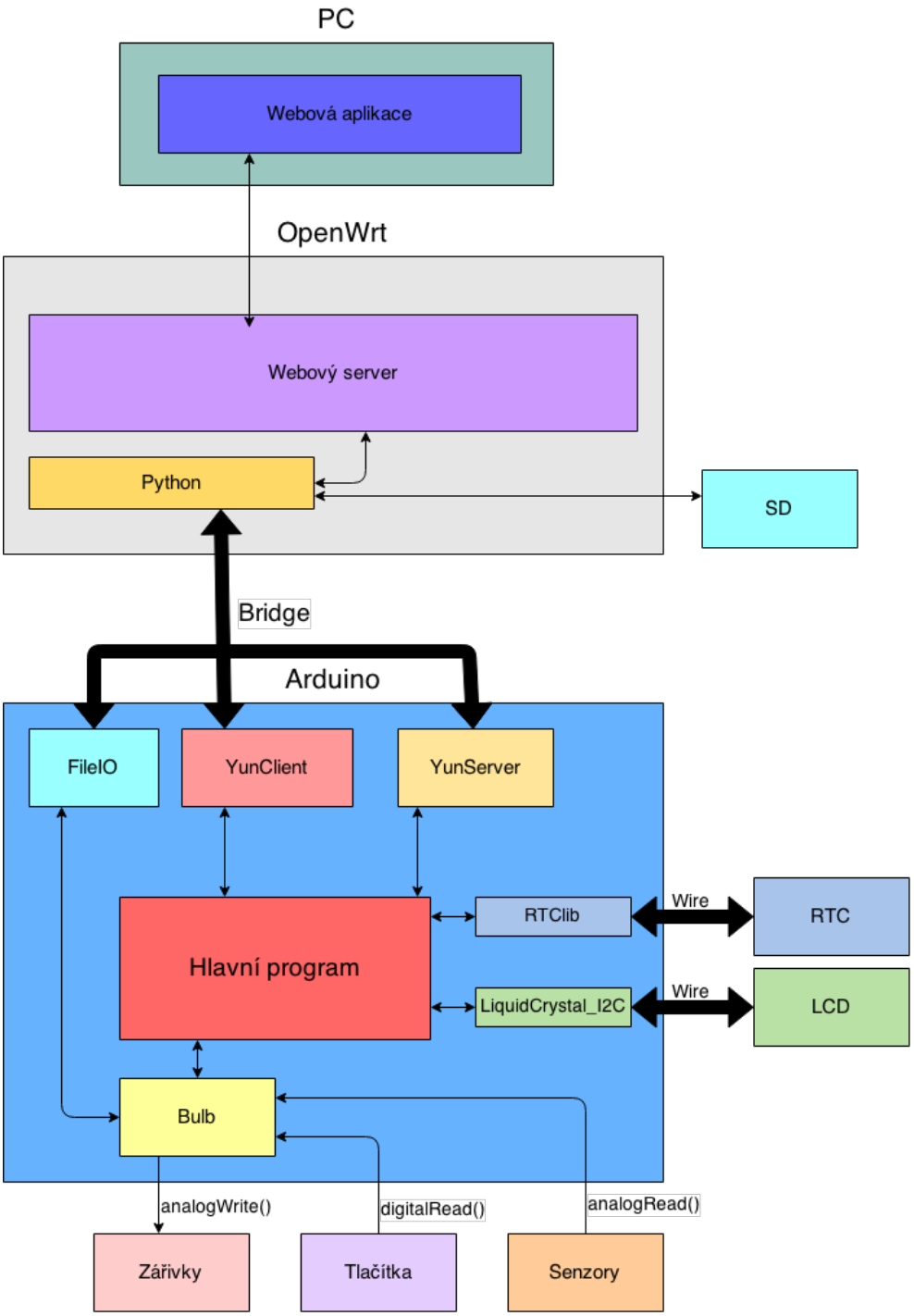
Softwarový návrh odráží ten hardwarový a je založen na komunikaci operačního systému OpenWrt a Arduino knihoven v programu běžícím na mikrokontroléru. Základ je poskytnut knihovnou *Bridge*, na kterou navazují další knihovny popsané v sekci 1.8.2.

Kromě těch bude program využívat knihovny pro obsluhu I2C periferií, a také moji třídu *Bulb*, která implementuje konfiguraci a řízení jednotlivých zářivek.

K ovládání PWM výstupů využijeme funkci $analogWrite(pin, hodnota)$, pro čtení hodnot fotosenzoru pak funkci $analogRead(pin)$. Stav tlačítek zjistíme pomocí funkce $digitalRead(pin)$. Tyto funkce jsou k dispozici v základní knihovně *Arduino*.

OpenWrt pak zajistí webový server pro provoz uživatelského rozhraní stmívače, obsluhu SD karty a přeposílání požadavků mikrokontroléru. Uživatelské rozhraní bude mít formu jednoduchých webových stránek, využívajících HTML a Javascript.

Obrázek 1.11: SW blokové schéma.



Kapitola 2

HW Implementace

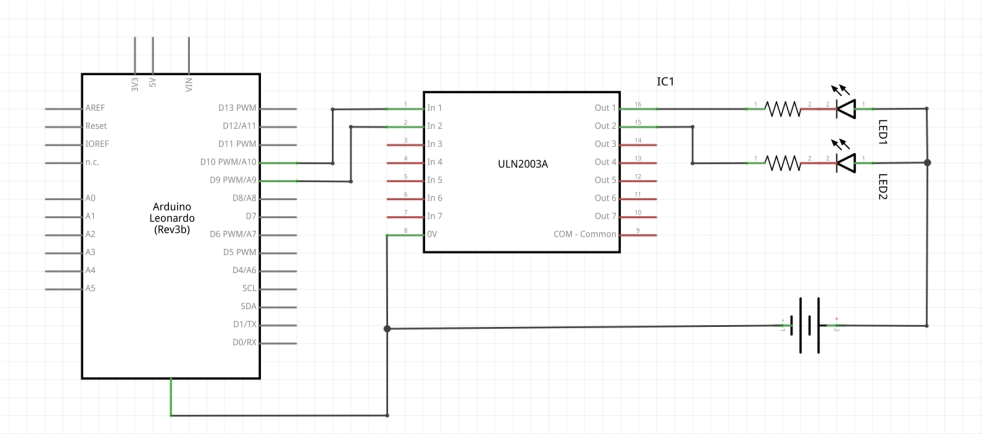
2.1 Ovládání zářivek

Zářivky jsou řízeny pomocí PWM výstupů desky (konkrétně jde o piny číslo 5 a 6). Ty jsou spojeny se vstupními piny obvodu ULN2003A. K výstupním pinům jsou pak přes 100Ω resistory připojeny jednotlivé LED, jak je vidět z obrázku 2.1. Je také nutné pamatovat na spojení uzemňovacích pinů Arduina, tranzistorového pole a zdroje napětí pro diody.

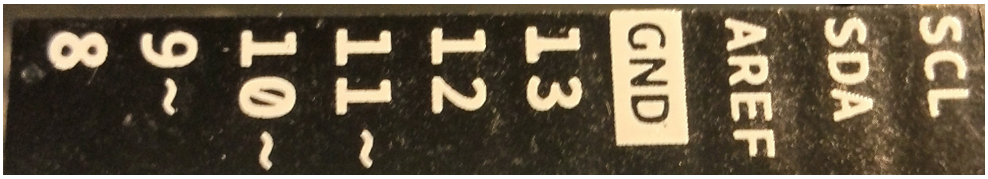
Každá z diod má také vlastní fotosenzor připojený k jednomu z analogových vstupů (konrétně piny označené A0 a A1). Diody ani fotosenzory nejsou k desce připájeny napevno, ale jsou připojovány pomocí svorkovnic. Je tedy možné je umístit na vhodná místa bez ohledu na polohu desky.

[1]Deska Arduino Yún je na obrázku nahrazena deskou Arduino Leonardo, která má stejné rozmístění pinů.

Obrázek 2.1: Schéma zapojení LED[1].



Obrázek 2.2: I2C piny Arduina.



2.2 Komunikace s periferiemi

Kromě tlačítek, fotosenzorů a diod zařízení obsahuje LCD zobrazující čas, datum a stav jednotlivých zářivek, a také RTC obvod sloužící jako časovač pro zářivky. Obě tyto periferie jsou připojeny pomocí sběrnice I2C a využívají knihovny vycházející z Arduino knihovny *Wire*. Ta implementuje komunikaci po této sběrnici.

Vzhledem k tomu, že obě periferie sdílejí stejnou sběrnici, je možno je připojit pouze pomocí napájecího a uzemňovacího pinu a pinů *sda* a *scl* přenášejících datový, respektive hodinový signál. Obě periferie potřebují napětí 5 V, je tedy možné je napájet z libovolného digitálího pinu (v módu `OUTPUT`) desky nebo pomocí 5V pinu. Deska i periferie také využívají 5V logickou úroveň, není tedy potřeba měnič úrovní a obě periferie můžeme připojit přímo k desce bez obav z poškození součástek.

Jako uzemňovací pin je využit nejbližší *gnd* pin desky (viz obrázek 2.2), k napájení slouží pin číslo 12 (zde je třeba nastavit mód pinu na `OUTPUT` a zapsat na něj logickou 1). Pin číslo 13, který je z hlediska pozice vhodnější, nemohl být použit, protože je k němu připojena červená LED umístěná na desce. Ta by jednak omezovala napětí potřebné k napájení periferií, a jednak by ji nebylo možné využít v programu stmívače.

2.2.1 Displej

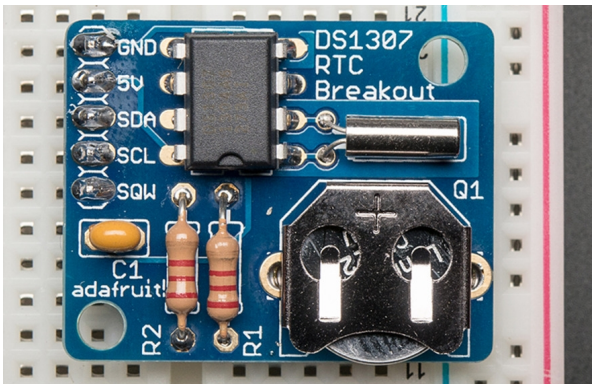
K displeji je nejprve nutno připájet I2C modul. Zde je potřeba dát pozor na uspořádání pinů displeje, neboť modul počítá s uzemňovacím pinem na krajní pozici. Některé displeje však mohou mít piny cyklicky posunuté.

Po připájení modulu je již možné připojit displej k desce pomocí I2C pinů. Modul také obsahuje potenciometr pro regulaci kontrastu. Zde se ovšem ukázalo, že displej je prakticky nečitelný, pokud není kontrast nastaven na maximum.

2.2.2 RTC obvod

Jako RTC nám slouží obvod DS1307 s vnějším krystalem a zálohovací baterií. Jelikož jsem chtěl ušetřit místo a práci s pájením, vybral jsem stavebnici od firmy Adafruit (`http://www.adafruit.com/product/264`).

Obrázek 2.3: Tištěný spoj s RTC obvodem. Obrázek převzat z `http://www.adafruit.com/images/970x728/x264-00.jpg.pagespeed.ic.v276mktUKh.jpg`



Ta kromě obvodu a dalších potřebých součástek obsahuje i tištěný spoj, na který lze vše snadno připájet. K Arduinu se pak obvod připojí pomocí kolíkové lišty a I2C pinů.

2.3 Sestavení zařízení

Jednotlivé části stmívače jsem nejprve testoval s pomocí nepájivých kontakních polí, finální zařízení je však připájeno na univerzální plošný spoj. Spolu s deskou Arduino Yún, displejem a hodinami tak tvoří jeden celek.

Jako univerzální plošný spoj jsem zvolil vrtanou desku 160×100 mm s kulatými pájecími body se standradní roztečí 2,54 mm. Tato volba se později ukázala jako nepříliš vhodná, neboť deska s pásovými spoji místo bodů by pravděpodobně usnadnil pájení zařízení. Nicméně se všechny prvky podařilo připájet bez výraznějších problémů.

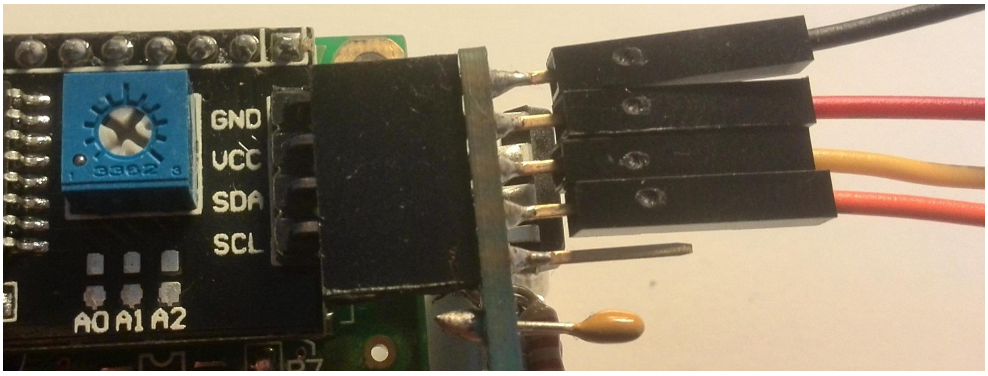
2.3.1 Deska Arduino Yún

Deska je ke spoji připojena pomocí dvou kolíkových lišt, nasazených na dutinkové vývody jednotlivých pinů. Zde bohužel menší problém představuje formát Arduina, který dodržuje rozteč dutinek 2,54 mm pouze do pinu číslo 7 na liště blíže ethernetového portu. Pak je na desce umístěna drobná mezera, která znemožňuje nasazení delší kolíkové lišty.

Kvůli tomuto můžeme přes kolíky přistupovat pouze k pinům číslo 0 až 7 a A0 až A5 (analogové vstupy desky). To nám však pro ovládání dvou diod pomocí PWM výstupů a čtení tlačítek a fotosenzorů postačuje. Displej a hodiny jsou připojeny zvlášť pomocí propojovacích kabelů.

Dále je potřeba připojit 5V výstup a uzemňovací pin desky. To je řešeno pomocí propojek napevno připájených k desce. Modrá propojka slouží pro

Obrázek 2.4: Připojení I2C periferií.



přípojení uzemění, červená pro připojení 5 V.

2.3.2 Displej a RTC

Displej i RTC jsou umístěné na sloupcích přišroubovaných k univerzálnímu spoji a k Arduinu jsou připojeny pomocí propojovacích kabelů. Vzhledem k tomu, že obě periferie sdílejí stejnou sběrnici, bylo nutné nějakým způsobem spojit jejich piny.

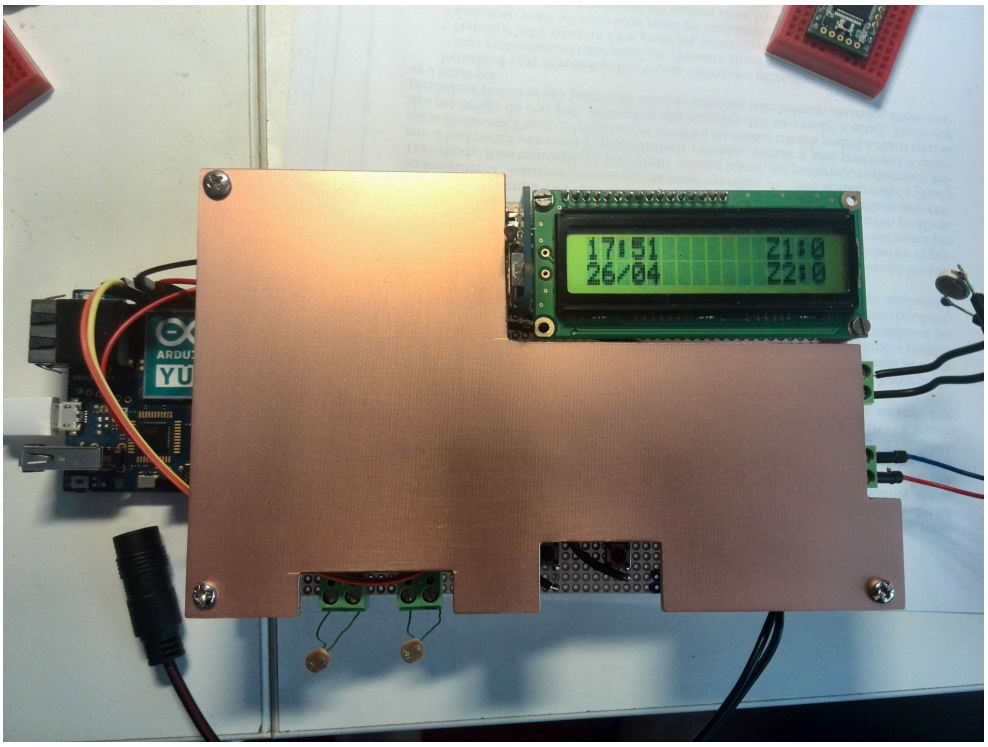
Zde jsem využil toho, že I2C modul displeje i destička s RTC obvodem (obrázek 2.3) mají piny ve stejném pořadí (tedy `GND`, `VCC`, `SDA`, `SCL`), a místo standardní kolíkové lišty jsem na zmíněnou destičku připájel dutinkovou lištu určenou pro tvorbu modulů pro Arduino. Ta má oproti standardní dutinkové liště delší kolíky, což umožňuje z druhé strany připojit další konektory.

V tomto případě se jedná o propojovací kabely dutinka/kolík, kterými jsou obě periferie připojeny k Arduinu. Připojení obou periferií je vidět na obrázku 2.4.

2.3.3 Napájení zářivek

Displej, hodiny, tlačítka a fotosenzory jsou napájeny pomocí 5V výstupů desky, zářivky však vzhledem k výkonu potřebují vlastní napájení. To je připojeno pomocí 2,1mm souosého napájecího konektoru. Volný konec tohoto konektoru je pak připájen do pravého dolního rohu desky, odkud je napěťový vodič veden k zářivkám a uzemňovací vodič do středu desky. Tam je spojen s uzemňovacími vodiči dalších součástek.

Obrázek 2.5: Konečná podoba zařízení.



Kapitola 3

SW Implementace

3.1 Ovládání zářivek

Zářivky je možno rozsvěcet a zhasínat buď pomocí tlačítek umístěných přímo na zařízení nebo pomocí webového rozhraní, kde najdeme tlačítka *On* a Off pro každou z diod.

Dále mohou zářivky reagovat na časovač (rozsvícení a zhasnutí v určitý čas) a fotosenzor (rozsvícení, pokud intenzita světla překročí určitou mez, například při rozsvícení hlavního světla v místnosti, a zhasnutí, pokud pod tuto mez klesne). Tyto možnosti lze opět nastavit pomocí webového rozhraní.

Ruční ovládání je nezávislé na automatickém. Je tedy možné zářivku zhasnou například pomocí tlačítka, i když by podle intenzity měla svítit.

Nastavení je při každé změně uloženo do konfiguračního souboru zářivky, který se nachází na SD kartě připojené k desce. Zvolená konfigurace je tak uchována i v případě odpojení zařízení od napájení a není tedy nutné zářivky znovu nastavovat.

Čtení konfiguračního souboru se provede při inicializaci programu a v případě selhání (neplatná data, poškozený soubor atp.) je zářivka nastavena na implicitní hodnoty (stmívání v 20:00, rozsvěcení 8:00, práh senzoru 500, reakce na senzor a časovač vypnuty).

3.1.1 Třída *Bulb*

V programu stmívače jsou jednotlivé zářivky reprezentovány instancemi třídy *Bulb*. Ta implementuje metody pro ovládání a nastavování zářivek, a také načítání a ukládání dat do konfiguračních souborů. Každou instanci tvoří jedna LED a k ní přiřazený fotosenzor.

Naše zařízení obsluhuje dvě zářivky, budeme tedy potřebovat dvě instance třídy *Bulb*. Ty jsou pro snadnější přístup uloženy v globálním poli, deklarovaném a inicializovaném při spuštění programu.

```
1 Bulb bulbArray[BULB_N] = {Bulb(BULB_1_PIN, BULB_1_PHOTOPIN,
    BULB_1_PATH), Bulb(BULB_2_PIN, BULB_2_PHOTOPIN, BULB_2_PATH)};
```

V hlavní smyčce programu je pak volána metoda *bulbControl*() každé zářivky, která zkontroluje stav časovače a fotosenzoru a vyvolá případnou změnu stavu. Kromě toho je v každém cyklu kontrolován stav tlačítek pro případné ruční rozsvícení či zhasnutí zářivky.

3.1.1.1 Atributy třídy

***int* m_pin** Číslo pinu zářivky.

***int* m_photoPin** Číslo pinu fotosenzoru.

***int[4]* m_timer** Nastavení časovače (hodina/minuta zhasínání a hodina/minuta rozsvěcení).

***int* m_threshold** Práh reakce na fotosenzor.

***int* m_oldLightIntensity** Hodnota senzoru z předchozího cyklu programu.

***bool* m_isAuto** Proměnná určující zda má zářivka reagovat na fotosenzor.

***bool* m_isTimer** Proměnná určující zda má zářivka reagovat na časovač.

***bool* m_sensorChange** Proměnná značící změnu na senzoru.

***bool* m_isOn** Proměnná značící stav zářivky.

***const char** * m_path** Cesta ke konfiguračním souboru na SD kartě.

3.1.1.2 Metody třídy

Konstruktor Vytvoří instanci třídy *Bulb* se zadanými parametry, tedy pinem LED, pinem fotosenzoru a cestou ke konfiguračnímu souboru. Ostatní atributy jsou nastaveny na implictiní hodnoty.

Parametry:

pin Číslo pinu, ke kterému je připojena dioda.

photoPin Číslo pinu, ke kterému je připojen fotosenzor.

path Cesta ke konfiguračnímu souboru na SD kartě.

Návratová hodnota:
Instance třídy *Bulb*.

Metoda *bulbControl*(*hour*, *minute*, *second*, *fadeDelay*) Metoda zkontroluje nastavení zářivky a v případě potřeby rozsvítí nebo zhasne. Zde bylo potřeba zajistit, aby bylo možné zářivku zhasnou nebo rozsvítit nezávisle na hodnotě fotosenzoru.

V každém cyklu hlavní smyčky se tedy při volání této metody zkontroluje hodnota fotosenzoru a reakce na něj se vyvolá, pouze pokud změna již překročila určitou mez.

```
1  int lightIntensity = analogRead(m_photoPin); //cteni fotosenzoru
2
3  if((lightIntensity > m_oldLightIntensity + SENSOR_TOLERANCE)
4    || (lightIntensity < m_oldLightIntensity - SENSOR_TOLERANCE)){
5    //aktualizace tridni promenne m_oldLightIntensity
6    m_oldLightIntensity = lightIntensity;
7    m_sensorChange = true;
8    Serial.println("Sensor change"); //debugovaci vypis
9  }
10 else
11   m_sensorChange = false;
```

Kontrola reakce na fotosenzor

Parametry:

hour Aktuální hodina.

minute Aktuální minuta.

second Aktuální sekunda.

fadeDelay Prodleva mezi jednotlivými cykly při snižování/zvyšování jasu v milisekundách.

Návratová hodnota:
Pokud došlo ke změně stavu zářivky tak true, jinak false.

Metoda *setTime*(*downHour*, *downMin*, *upHour*, *upMin*) Metoda pro nastavení času pro vypínání a zapínání zářivky. Je zde také provedena kontrola platnosti zadávaného času. Nastavuje se pouze hodina a minuta, samotná reakce se provede v `hh::mm::00`.

Parametry:

downHour Hodina zhasínání.

downMin Minuta zhasínání.

upHour Hodina rozsvěcení.

upMin Minuta rozsvěcení.

Návratová hodnota:
False pokud byl zadán neplatný čas, nebo pokud byly časy rozsvěcení a zhasínání shodné, jinak true.

Metoda *setThreshold*(*threshold*) Nastavuje práh reakce na fotorezisor, v rozmezí od 100 do 1000, přičemž čím nižší číslo, tím větší intenzita světla.

Parametry:

threshold Hodnota prahu reakce.

Návratová hodnota:
False pokud byla zadaná hodnota mimo interval <100,1000>, jinak true.

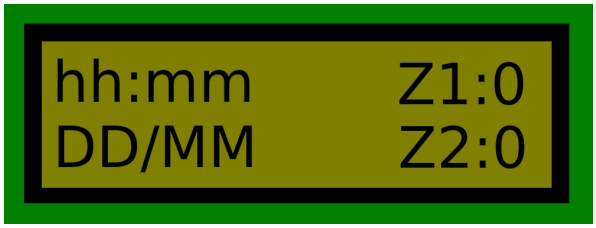
Metoda *readConfigFile*() Metoda přečte konfigurační soubor, uložený na SD kartě, a nastaví podle něj parametry zářivky.

Cesta k tomuto souboru je uložena v proměnné *m_path*. Konfigurační soubor obsahuje údaje o nastavení časovače, prahu fotosenzoru a proměnných *m_isAuto* a *m_isTimer*. Každý údaj je uložen na samostatném řádku. Načítání se tedy provádí po bytech s tím, že znak `\n` slouží jako oddělovač.

```
while((c = configFile.read()) != EOF)
  {
    if(c == '\n')
    {
      bufferPointer = 0;
      int val = atoi(readBuffer);

      //reinicializace bufferu
      for(int i = 0; i < READ_BUFFER_SIZE; i++)
        readBuffer[i] = 0;

      if(paramPointer < BULB_PARAMS)
      {
        //ukladani parametru zarivky
        bulbParams[paramPointer] = val;
        paramPointer++;
      }
    }
    else
    {
      readBuffer[bufferPointer] = c;
      //kruhove prepisovani bufferu
      bufferPointer = (bufferPointer + 1) % READ_BUFFER_SIZE;
    }
  }
```

Načítání konfiguračního souboru

Obrázek 3.1: Zobrazení stavu na displej.



Parametry:
Žádné.

Návratová hodnota:
True pokud byla správně načtena data, false pokud soubor s daty neexistuje, obsahuje málo parametrů nebo některý z parametrů byl neplatný.

Metoda *writeConfigFile*() Metoda pro zápis dat do konfiguračního souboru. Data jsou zapsána po řádcích v pořadí *m_timer*[0] – *m_timer*[3] (nejprve hodina a minuta zhasínání, poté hodina a minuta rozsvěcení), *m_threshold*, *m_isAuto* a *m_isTimer*.

Parametry:
Žádné.

Návratová hodnota:
Pokud soubor neexistuje tak false, jinak true.

Výše byly popsány pouze významnější metody třídy *Bulb*. Kromě nich obsahuje ještě standardní metody pro přístup k třídním proměnným, metody pro ovládání diod *fadeOn*(), *fadeOff*() a *toggle*() a metodu *toString*() pro snadný výpis stavu zářivky.

3.2 Komunikace s periferiemi

Jak již bylo zmíněno v sekci 1.3, Arduino s displejem a RTC obvodem komunikuje po sběrnici I2C pomocí knihoven *LiquidCrystal_I2C* a *RTClib*. Obě jsou postaveny na knihovně *Wire*, dodávané spolu s vývojovým prostředím.

3.2.1 Displej

Displej není kvůli čitelnosti obnovován v každém cyklu programu, ale je aktualizován pokaždé, když dojde ke změně zobrazovaných údajů. Funkce pro zobrazení dat na displej je tedy volána minimálně každou minutu.

Na platformě Arduino bohužel není podporována funkce *snprintf*(), kterou bychom mohli použít pro přípravu řetězce vypisovaného na displej. To však

nepředstavuje větší problém, neboť tuto funkci potřebujeme víceméně pouze pro rozšíření o nulu při zobrazování času ve formátu `hh:mm` a data ve formátu `DD/MM`. Zde můžeme rozšíření implementovat jednoduchým makrem.

```
#define TIME_FORMAT(num) ((num < 10) ? "0" + String(num):String(
    num))
```

Makro pro formátování času a data

Samotné zobrazování dat je provedeno pomocí metod *print*(), *setCursor*() a *clear*() třídy *LiquidCrystal_I2C*.

```
void printStatus(DateTime dateTime)
{
  //vymazani lcd
  lcd.clear();
  //cteni stavu zarivek
  String state1 = String(bulbArray[0].isOn());
  String state2 = String(bulbArray[1].isOn());

  //nastaveni pozice pro stav zarivek
  lcd.setCursor(12, 0);
  lcd.print("Z1:" + state1);
  lcd.setCursor(12, 1);
  lcd.print("Z2:" + state2);

  //zobrazeni casu a data
  printTimeDate(dateTime);
}
```

Zápis stavu zařízení

```
void printTimeDate(DateTime dateTime)
{
  //nacteni udaju
  int _hour = dateTime.hour();
  int _min = dateTime.minute();
  int _month = dateTime.month();
  int _day = dateTime.day();

  lcd.setCursor(0, 0);
  lcd.print(TIME_FORMAT(_hour) + ":" + TIME_FORMAT(_min));
  lcd.setCursor(0, 1);
  lcd.print(TIME_FORMAT(_day) + "/" + TIME_FORMAT(_month));
}
```

Zápis času a data

3.2.1.1 Třída *LiquidCrystal_I2C*

Jak již bylo zmíněno, k ovládání displeje přes I2C modul použijeme knihovnu *LiquidCrystal_I2C*. Ta obsahuje stejnojmennou třídu s metodami pro ini-

cializaci, zápis a mazání LCD. Instance této třídy je deklarována globálně a inicializována na začátku programu.

Konstruktor Vytvoří instanci třídy *LiquidCrystal_I2C*.

Parametry:

lcd_Addr Adresa zařízení na I2C sběrnici (v našem případě `0x27`).

cols Počet sloupců displeje (16).

rows Počet řádku displeje (2).

Návratová hodnota:
Instance třídy *LiquidCrystal_I2C*.

Metoda $print(val, base)$ Zobrazí řetězec na displej, počínaje pozicí kurzoru. Znaky, které se již nevejdou na řádek, jsou zahozeny.

Parametry:

val Zobrazovaná hodnota nebo řetězec.

base V jaké soustavě má být hodnota zobrazena.

Návratová hodnota:
Žádná.

Metoda $setCursor(col, row)$ Nastaví pozici kurzoru.

Parametry:

col Sloupec kurzoru.

row Řádek kurzoru.

Návratová hodnota:
Žádná.

Metoda $clear()$ Vymaže znaky zobrazené na LCD.

Parametry:
Žádné.

Návratová hodnota:
Žádná.

3.2.2 RTC obvod

Ke komunikaci s hodinami po I2C sběrnici použijeme knihovnu *RTClib*. V té najdeme třídu *RTC_DS1307* určenou pro náš obvod.

V případě, že bychom chtěli RTC obvod ze stmívače úplně odstranit, bylo by možné využít třídu *RTC_Milis*, založenou na funkci *milis*() (vrátí čas od spuštění Arduina v milisekundách). Toto řešení však samozřejmě nepodporuje bateriovou zálohu, navíc po přibližně padesáti dnech dojde k přetečení datového typu (unsigned long) a vynulování hodin[23].

Kromě těchto tříd obsahuje knihovna také třídu *DateTime* pro ukládání a manipulaci s časem a datem.

Z knihovny využijeme především metodu *now*(), vracející instanci třídy *DateTime* s aktuálním časem a metodu *adjust*() pro seřízení hodin. Třída bohužel neobsahuje metody pro nastavení zvlášť času a data, proto jsem pro tento účel připravil funkce *setRtcTime*() a *setRtcDate*().

```
1  bool setRtcTime(int _hour, int _minute)
2  {
3    //kontrola rozsahu zadavanych hodnot
4    if (_hour > 23 || _hour < 0 || _minute > 59 || _minute < 0 )
5    {
6      return false;
7    }
8
9    DateTime date = rtc.now();
10   //nahrazeni aktualniho casu pozadovanym casem
11   DateTime newDate (date.year(), date.month(), date.day(), _hour,
       _minute, date.second());
12
13   rtc.adjust(newDate);
14   printStatus(newDate);
15
16   return true;
17 }
```

Funkce pro seřízení času

```
bool setRtcDate(int _day, int _month, int _year)
{
  //kontrola rozsahu zadavanych hodnot
  if (_month > 12 || _month < 1)
    return false;

  int monthDays[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
     31}; //pocty dnu v mesici

  //test prestupneho roku
  if((_year % 4 == 0  && _year % 100 != 0) || _year % 400 == 0)
    monthDays[1]++;

  if (_day < 1 || _day > monthDays[_month - 1])
    return false;

  DateTime date = rtc.now();
  //nahrazeni aktualniho data pozadovanym datem
  DateTime newDate (_year, _month, _day, date.hour(),
                    date.minute(), date.second());

  rtc.adjust(newDate);
  printStatus(newDate);

  return true;
}
```

Funkce pro seřízení data

3.3 Komunikace zařízení s PC

Komunikace s počítačem je realizována pomocí webového serveru, spuštěném na linuxové části Arduina. Ten přijímá klienty a předává jejich požadavky ve formě REST volání pomocí sériové linky mikrokontroléru, kde běží program stmívače. Mikrokontrolér požadavky přijímá a zpracovává pomocí tříd *YunServer* a *YunClient*, popsaných v sekci 1.8.

Deska rozeznává tři možné přístupové body pro REST. Jsou to */arduino*, */data* a */mailbox*. Nás bude zajímat */arduino*, protože tento bod předává celý řetězec přidaný k URL přímo mikrokontroléru[24]. Požadavky tedy mají tvar `arduino.local/arduino/příkaz/parametr1/parametr2/...` (`arduino` je implicitní název desky, který je možno změnit).

Řetězec `příkaz/parametr1/parametr2/...` je tedy předán programu stmívače, který vyparsuje jaký příkaz s jakými parametry se má provést a odpoví klientovi. Jednotlivé příkazy a jejich parametry jsou blíže popsány v tabulce 3.1.

```
YunClient client = server.accept();
DateTime now = rtc.now();

if (client) {
    //funkce pro zpracovani pozadavku
    process(client, now);
    client.stop();
}
```

Přijímání nového klienta (na straně mikrokontroléru)

Pro zpracování požadavku klienta jsem připravil funkci *process*(). V ní je nejprve načítán řetězec požadavku, dokud se nenarazí na znak `/`. Tím je načten typ příkazu. Poté se pomocí metody *client.parseInt*() načítají jednotlivé parametry a nakonec je pomocí metody *client.print*() zaslána odpověď klientovi.

```
void process(YunClient client, DateTime dateTime)
{
  //nacteni prikazu
  String command = client.readStringUntil('/');
  //promenne pro ulozeni nacitanych parametru
  int led;
  int clientHour, clientMinute, clientDay, clientMonth,clientYear;
  command.trim(); //odstraneni mezer
  Serial.println(command); //debugovaci vypis
  if (command == "ledOn") {
    led = (client.parseInt() - 1) % BULB_N;

    bulbArray[led].fadeOn(FADE_DELAY);
    printStatus(dateTime);
    client.print(F("Zapnuto"));
    return;
  }
  ...
}
```

Ukázka funkce pro zpracování požadavku klienta

3.3.1 Konfigurace WiFi sítě

Pro přístup k webovému serveru je nutné být připojen ke stejné síti jako deska. Desku lze k síti připojit buď pomocí ethernetového kabelu nebo WiFi adaptéru. V druhém případě je nutné nejprve zadat konfiguraci sítě, ke které se chceme připojit. Nejjednodušší způsob jak to učinit, je pomocí webového rozhraní pro konfiguraci desky, dostupného z adresy `arduino.local` (místo `arduino` může být uživatelem zvolený název zařízení).

Zde najdeme konfiguraci WiFi sítě, možnost změny jména desky a přístupového hesla (defaultně „arduino”) a mnoho dalších nastavení pomocí roz-

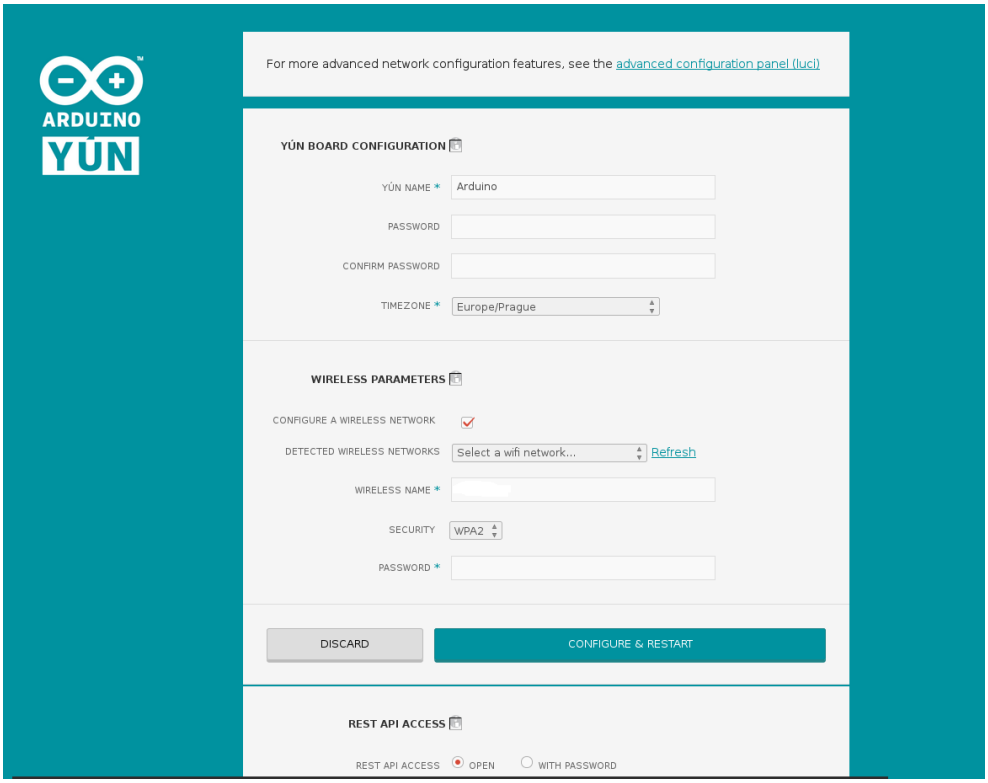
Příkaz	**Parametry**	**Popis**
ledOn	číslo zářivky	Rozsvítí danou zářívku.
ledOff	číslo zářivky	Zhasne danou zářivku.
time	hodina minuta	Nastaví čas na RTC obvodu zařízení.
date	den měsíc rok	Nastaví datum na RTC obvodu zařízení.
setTimer	číslo zářivky hodina rozsvěcení minuta rozsvěcení hodina zhasínání minuta zhasínání	Nastaví čas rozsvěcení a zhasínání zvolené zářivky.
setOther	číslo zářivky práh reakce na fotosenzor příznak reakce na časovač příznak reakce na fotosenzor	Nastaví další parametry zvolené zářivky.
ledCheck	číslo zářivky	Vrátí klientovi řetězec popisující stav zářivky.
rtcCheck	-	Vrátí klientovi řetězec s datem a časem nastaveným na RTC.

Tabulka 3.1: Tabulka REST volání

hraní *LuCi*. Je zde také možnost povolit přístup k REST API bez nutnosti zadávat heslo, což je klíčové pro fungování komunikace se stmívačem.

V případě, že se chceme připojit na novou WiFi síť, je možné buď připojit desku pomocí ethernetového kabelu a bezdrátové připojení nakonfigurovat pomocí tohoto rozhraní, nebo použít WiFi vysílač na desce. Pokud podržíme tlačítko pro resetování WiFi adaptéru (nachází ve vedle USB Host portu) déle než 5 sekund, dojde k resetování WiFi do továrního nastavení.

Obrázek 3.2: Rozhraní pro konfiguraci desky Ardino Yún včetně nastavení přístupu k REST API.



V tomto módu vytvoří zařízení vlastní WiFi síť s názvem *ArduinoYun – XXXXXXXXXXXX* (místo „X" je MAC adresa desky). Po připojení k této síti je možné přistoupit ke konfiguračnímu rozhraní a pomocí něj nastavit parametry sítě, na kterou se má Arduino připojit[24].

Toto tlačítko také slouží pro uvedení instalace OpenWrt do továrního stavu, a to v případě, že ho podržíme déle než 30 sekund.

WiFi vysílač se dá také využít v místech, kde bychom chtěli bezdrátový přístup k zařízení, ale kde nemáme k dispozici žádnou WiFi síť.

3.4 Webové rozhraní

Pomocí REST volání je možné stmívač ovládat z libovolného prohlížeče nebo například z příkazové řádky pomocí cURL. Pro větší pohodlí jsem však vytvořil uživatelské rozhraní ve formě statických webových stránek, vytvořených pomocí HTML a Javascriptu.

Server se stránkami běží přímo na zařízení (viz sekce 1.8.1.1) a je dostupný pro všechny počítače v dané síti na adrese `arduino.local/sd/ArduinoWebDimmer`.

Obrázek 3.3: Ukázka webového rozhraní stmívače - konfigurace zářivky číslo 1.

Pro ovládání jsem připravil několik javascriptových funkcí, které jsou umístěny v souboru `ledScript.js`.

3.4.1 Zasílání požadavků

Pro zasílání požadavků a získávání odpovědí využívají stránky minimalizovanou verzi knihovny *jQuery*, nazvanou *zepto.js* (ke stažení na `http://zeptojs.com/`). Z této budeme potřebovat pouze funkci *load*() pro zaslání požadavku a nahrání odpovědi do požadovaného HTML elementu[25].

```
1 function ledOn(ledNum)
2 {
3   $('#content' + ledNum).load('/arduino/ledOn/' + ledNum); //$ je
      volani jQuerry knihovny
4 }
```

Ukázka použití funkce load() ve funkci pro rozsvícení zářivky.

Funkce *load*() tedy vyšle požadavek na zvolené URL představující nějaký příkaz z REST API (zde `arduino.local/arduino/ledOn/1`) a odpověď, odeslanou mikrokontrolérem pomocí *client.print*() (v tomto případě jde o *client.print("Nastaveno")*), nahraje do HTML elementu s id *content*.

Funkci *ledOn*() s vhodným parametrem pak stačí přiřadit k odpovídajícímu tlačítku.

```
<input type="button" class="button1" onclick="ledOn(1)"value="On">
```

HTML tlačítko pro rozsvícení zářivky číslo 1.

Na tomto principu fungují všechny javascriptové funkce pro komunikaci se stmívačem, liší se pouze volané URL (podle příkazu, který se má provést - viz tabulka 3.1) a prvek, kam být nahrána odpověď.

3.4.2 Struktura stránek

Webové rozhraní obsahuje stránku zobrazující status zařízení, stránky pro ovládání a nastavení jednotlivých zářivek a stránku umožnující seřídit hodiny stmívače. Kromě toho zde také najdeme krátkou nápovědu popisující jednotlivé ovládací prvky.

Navigace mezi stránkami je řešena pomocí menu, které můžeme vidět na obrázku 3.3. Toto menu je pomocí jednoduchého skriptu nahráváno do určeného HTML elementu každé ze stránek.

3.4.2.1 Status

Stránka zobrazuje stav jednotlivých zářivek a hodin zařízení. Ten je získán pomocí funkce *statusCheck*(), která vyšle potřebné požadavky a odpovědi nahraje na zvolené místo na stránce, jak je popsáno v sekci 3.4.1. Tato funkce je volána pomoci *onload* atributu HTML tagu *body* a je tedy vykonána pokažde při načtení stránky.

```
<body onload="statusCheck()">
...
<\body>
```

Načítání stavu zařízení

Pro obnovení stavu zařízení tedy stačí obnovit stránky například pomocí klávesy F5 nebo znovu kliknout na tlačítko *Status* v postranním menu.

3.4.2.2 Ovládání zářivek

Zde najdeme jednoduché ovládání zářivky pomocí tlačítek *On* a Off. Je zde také tlačítko *Status*, které načte stav zářivky. Tlačítka jsou pomocí atributu *onclick* svázána s javascriptovými funkcemi *ledOn*(), $ledOff()$ a *ledCheck*(), které pouze volají příslušné příkazy z tabulky 3.1.

3.4.2.3 Nastavení zářivek

Na této stránce (můžeme ji vidět na obrázku 3.3) se nachází rozhraní pro nastavení zářivky. To dovoluje nastavit časy rozsvícení a zhasínání, práh reakce na fotosenzor a příznaky reakce na fotosenzor a časovač.

Časy se nastavují zvlášť a jsou zadány do textových polí, odkud jsou po stisknutí tlačítka *Nastavit* načteny funkcí *setTimer*(). Ta provede kontrolu načítaných dat a v případě, že jde o platné časy ve formátu `hh:mm`, vyvolá příslušný požadavek. Pokud jsou data neplatná, je uživatel upozorněn. Formát dat je kontrolován pomocí regulárního výrazu.

Funkce načítá data z textboxů pomocí *document.getElementById*() a k tlačítku je připojena pomocí atributu *onclick*.

Práh reakce na fotosenzor je zadáván pomocí textového pole, příznaky pomocí checkboxů. Načtení a předání dat stmívači je realizováno stejně jako v případě časů rozsvěcení a zhasínání, pouze je při stisku druhého tlačítka *Nastavit* volána funkce *setOther*().

V případě, že si uživatel přeje nastavit pouze příznaky, je možno nechat pole pro hodnotu prahu prázdné. Funkce do příkazu doplní hodnotu -1 a mikrokontrolér tak pozná, že ji má ignorovat.

3.4.2.4 Nastavení hodin

Stejně jako nastavení časů jednotlivých zářivek je řešeno i nastavení hodin zařízení. Nastavuje se zvlášť čas ve formátu `hh:mm` a zvlášť datum ve formátu `DD/MM/RRRR`.

I zde jsou načítaná data kontrolována pomocí regulárního výrazu a v případě neplatnosi je vyvolána chybová hláška.

Kapitola 4

Testování

Jednotlivé části zařízení jsem nejprve testoval samostatně. Poté jsem provedl celkový test.

4.1 Testy provádněné při sestavování zařízení

Při sestavování zařízení jsem testoval především, jestli jsou kontaky součástek skutečně správně připájeny a dochází mezi nimi ke spojení. Toto jsem prováděl pomocí multimetru s funkcí akustické zkoušky obvodu. Také jsem testoval, zda je na příslušných spojích naměřen správný odpor.

Zde jsem narazil pouze na dva méně závažné problémy. Tlačítka, která jsem na zařízení použil, měla poměrně krátké piny a z druhé strany desky tak vyčnívala pouze malá část. Tu se mi sice podařilo připájet, nicméně při stisku jednoho z tlačítek (když se deska lehce prohne) došlo k prasknutí připájeného spoje.

Tlačítko jsem tedy připájel znovu a problém by už neměl nastat. Pokud by se opakoval, bylo by pravděpodobně nutné vyměnit tlačítka za nějaká s delšími vodiči.

Druhý problém se týkal napájecího konektoru, který se asi po dvou dnech po připájení desce prodřel a musel tedy být vyměněn za nový.

Kromě těchto drobnějších závad nedošlo při pájení zařízení na univerzální spoj k žádným dalším problémům.

4.2 Testy periferií

4.2.1 Displej

Displej jsem testoval pomocí jednoduchého programu, který na něj zobrazoval jednotlivé znaky zaslané pomocí sériové linky.

Program umožňoval zápis znaků, mazání znaků pomocí klávesy *Backspace* a také přesun kurzoru pomocí směrových kláves. Tím jsem otestoval chování

metod *print*() a *setCursor*() třídy *LiquidCrystal_I2C*. Dále jsem otestoval chování metody *print*() při zobrazování celých řetězců.

Všechny funkce se při testu chovaly podle očekávání, pouze se ukázalo, že kvůli čitelnosti bude vhodné omezit obnovování displeje na minimum, jak je zmíněno v sekci 3.2.1.

4.2.2 RTC obvod

Zkouška RTC obvodu proběha podobně. Tady jsem použil zdrojový kód dodávaný s knihovnou, který slouží jako ukázka základních funkcí. Program každé tři sekundy pošle po sériové lince informace o času, datu a další údaje. Výstup programu jsem zobrazoval pomocí terminálu *Putty*. Poté jsem otestoval seřízení obvodu pomocí metody *adjust*().

I zde vše proběhlo v pořádku a knihovna fungovala podle očekávání.

4.2.3 Fotosenzory, tlačítka a LED

Další části zařízení jsem samostatně testoval pomocí nepájivých kontaktních polí. Jednalo se pouze o jednoduché zkoušky ve formě čtení nebo zápisu na příslušném pinu.

Jediný menší problém vznikl, když jsem fotosenzor místo způsobu pospsaného v sekci 1.3.3 zapojil obráceně, tedy rezistor a vodič připojený k analogovému vstupu desky jsem umístil mezi senzor a napětí a ne mezi senzor a zem. Čtené hodnoty tak byly obrácené (čím nižší hodnota, tím intenzivnější světlo). Tento problém jsem ve finální verzi zařízení odstranil.

4.3 Test webového rozhraní a komunikace se zařízením

Funkce webového rozhraní jsem testoval nejprve samostatně, bez zasílání požadavků Arduinu. Volání příkazů z REST API pomocí *jQuery* funkce *load* jsem nahradil jednoduchými výpisy. Těmi jsem testoval, zdali jsou po stisknutí tlačítka načítány správné údaje ze stránky. Zkoušel jsem také reakce na špatně zadaný nebo prázdný vstup. Zde, až na několik menších chyb, proběhlo všechno v pořádku.

Dále jsem testoval komunikaci se zařízením, kdy jsem příchozí požadavky zobrazoval pomocí sériové linky. Zde opět vše proběhlo bez probémů

Nakonec jsem se rozhodl otestovat parsování parametrů z řetězce REST volání pomocí metody *parseInt*(). Chtěl jsem především ověřit chování této metody v případě neúplného či neplatného příkazu. Ukázalo se, že metoda funguje podle očekávání a v případě volání na prázdný řetězec vrací nulu.

Načtení uživatelem zadaných údajů z webového rozhraní a jejich předání mikrokontroléru na desce tedy proběhlo v pořádku.

4.4 Test ukládání a načítaní konfiguračních souborů

V tomto testu jsem zkoušel, zdali je nastavení zářivek správně ukládáno a načítáno z konfiguračních souborů na SD kartě. Zajímalo mě, co se stane v případě, že konfigurační soubory neexistují, nebo obsahují neplatná či neúplná data. Také jsem kontroloval zápis dat při změně nastavení zářivky.

Kontrolu obsahu souborů jsem prováděl přímo v operačním systému OpenWrt, ke kterému jsem přistupoval pomocí SSH.

Při tomto testu se ukázalo, že je nutno inicializaci programu stmívače doplnit o čekací smyčku, která počká na naběhnutí linuxové části desky. Bez tohoto čekání docházelo k pokusu o přístup k SD kartě v době, kdy operační systém ještě nebyl zcela připraven a kontrola existence souboru pomocí metody *FileSystem.exists*() tak vracela false.

4.4.1 Arduino IDE 1.6.3

Před testováním přístupu ke konfiguračním souborům jsem provedl aktualizaci Arduino IDE na verzi 1.6.3. Tato verze však obsahuje chybu v knihovnách dodávaných spolu s IDE (kontrétně se jedná o knihovnu určenou pro manipulaci s SD kartou).

Program zkompilovaný v této verzi vývojového prostředí se choval velice nestabilně, docházelo k velkým prodlevám při načítání a zápisu na SD kartu a několikrát i k zamrznutí mikrokntroléru desky. Při kompilaci programu ve verzích 1.5.x a 1.6.0 se přitom zařízení chová normálně.

Tento problém je zmíněn na oficiálním fóru společnosti Arduino (`http://forum.arduino.cc/`), dá se tedy předpokládat, že v dalších verzích vývojového prostředí bude opraven.

4.5 Celkový test zařízení

Nakonec jsem provedl celkovou zkoušku stmívače. Při té jsem testoval, zdali spolu jednotlivé části komunikují tak, jak mají, funguje uživatelské rozhraní a zářivky správně reagují na různá nastavení.

Nejdříve jsem otestoval, zda webové rozhraní zobrazuje správné hodnoty a zda je z něj skutečně možné ovládat zářivky, měnit jejich nastavení a seřizovat RTC obvod. Dále byla testována reakce zářivek na časovač a na fotosenzor.

Poté jsem vyzkoušel ovládání pomocí tlačítek, nejprve samostatně, a potom spolu se zapnutou reakcí na fotosenzor. Zde jsem se chtěl ujistit, že zářivka půjde zhasnout pomocí tlačítek nebo webového rozhraní, i když by podle fotosenzoru měla svítit.

V tomto testu se vyskytlo několik menších problémů, které se mi však podařilo opravit. Zařízení by se tedy mělo chovat dle předpokladů.

Závěr

V této práci jsem nejprve prozkoumal několik vývojových desek, které jsou dnes na trhu k dispozici a z nich vybral přípravek Arduino Yún, který slouží jako základ pro implementaci programovatelného stmívače světel pro domácí zvířata. Dále jsem zvolil vhodné periferie a způsob komunikace s PC.

V implementační části práce jsem vytvořil zařízení umožňující automatizované i ruční ovládání dvou LED zářivek pomocí fotosenzorů, časovače a tlačítek. Také jsem připravil webové rozhraní, pomocí kterého je možno stmívač snadno nastavovat a ovládat.

Výsledné zařízení jsem otestoval a jeví se být plně funkční.

Budoucí práce

Pro budoucí práce mě napadá několik vylepšení. Asi nejzřetelnější by bylo nahrazení univerzálního plošného spoje tištěným spojem ve formě Arduino Shieldu, který by bylo možno nasadit přímo na dutinkové lišty desky. Dále by se dalo experimentovat s různými zdroji světla a jejich ovládáním.

Softwarová část by se dala poměrně snado rozšířit o logování intenzity světla v průběhu dne nebo o monitorování pomocí webkamery připojené k USB Host portu Arduina Yún. Také by se dalo uvažovat o snímání dalších údajů, jako například teplota či vlhkost.

Z rozsáhlejších projektů se nabízí port programu na jinou vývojovou desku. Zde by určitě byla zajímavá volba ESP8266, což je extrémně levný (asi 2 - 3 dolary) SoC s WiFi připojením, který může fungovat buď samostatně, nebo propojený s jiným mikrokontrolérem pomocí sériové linky.

V době, kdy jsem vybíral vhodnou desku (léto 2014), nebylo ESP8266 ještě výrazně rozšířené, dnes má však poměrně rozsáhlou komunitu a vznikají například porty Arduino knihoven nebo podpora jazyka Lua.

Vzhledem k tomu, že mikrokontrolér ATmega32U4 umístěný na desce Yún neposkytuje příliš velké možnosti (zejména flash paměť 32 KiB je poměrně

omezující, zvlášť při použití většího množství knihoven), by dalším zajímavým projektem mohlo být vytvoření desky postavené na stejném principu.

Deska by mohla propojovat právě Atheros AR9331 či jiný SoC v výkonnějším mikrokontrolérem. Pokud by se jako základ použilo například Teensy 3.1 (viz 1.1.2), které je kompatibilní s většinou Arduino knihoven, neměl by být větší problém naportovat knihovnu *Bridge* a knihovny na ni navazující.

Vznikla by tak deska v podstatě shodná s deskou Yún, pouze s výkonnějším mikrokontrolérem.

Literatura

[1] Arduino: *Arduino Yún Overview*. [cit. 2015-02-20]. Dostupné z: http://arduino.cc/en/Main/ArduinoBoardYun?from=Products.ArduinoYUN

[2] Arduino: *Arduino Uno Overview*. [cit. 2015-02-21]. Dostupné z: http://arduino.cc/en/Main/arduinoBoardUno

[3] Atmel: *ATmega328 datasheet*. [cit. 2015-02-21]. Dostupné z: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

[4] PJRC: *Teensy USB Development Board*. [cit. 2015-02-21]. Dostupné z: https://www.pjrc.com/teensy/

[5] Raspberry Pi Foundation: *Raspberry Pi Documentation*. [cit. 2015-02-21]. Dostupné z: http://www.raspberrypi.org/documentation/

[6] Atmel: *ATmega32U4 datasheet*. [cit. 2015-02-20]. Dostupné z: http://www.atmel.com/Images/7766s.pdf

[7] Atheros: *Atheros AR9331 datasheet*. [cit. 2015-02-20]. Dostupné z: http://www.openhacks.com/uploadsproductos/ar9331_datasheet.pdf

[8] ShenZhen GeTian Optoelectronics Co.,LTD: *LED WHITE 1W datasheet*. [cit. 2015-02-24]. Dostupné z: http://www.gme.cz/img/cache/doc/518/120/vykonova-led-led-power-white-1w-130lm-120-datasheet-1.pdf

[9] ST Microelectronics: *ULN2003A datasheet*. [cit. 2015-02-24]. Dostupné z: http://www.gme.cz/img/cache/doc/380/005/uln2003a-datasheet-1.pdf

[10] Arduino: *LiquidCrystal tutorial*. [cit. 2015-02-24]. Dostupné z: http://arduino.cc/en/Tutorial/LiquidCrystal

[11] Geeetech: *Serial I2C 1602 16x2 Character LCD Module*. [cit. 2015-02-24]. Dostupné z: http://www.geeetech.com/wiki/index.php/Serial_I2C_1602_16%C3%972_Character_LCD_Module

[12] Adafruit: *Using a photocell*. [cit. 2015-02-26]. Dostupné z: https://learn.adafruit.com/photocells/using-a-photocell

[13] Arduino: *Bridge tutorial*. [cit. 2015-02-26]. Dostupné z: http://arduino.cc/en/Reference/YunBridgeLibrary

[14] Bechynský, Š.: *Moje garáž, můj hrad*. [cit. 2011-03-04]. Dostupné z: http://www.zdrojak.cz/clanky/moje-garaz-muj-hrad-pripojujeme-amazon/

[15] Garnock-Jones, T.: *Reverse HTTP - Specifications*. LShift Ltd., 2009, [cit. 2011-03-04]. Dostupné z: http://reversehttp.net/reverse-http-spec.html

[16] Peterson, B.: *Internet of things with Arduino Yun and Yaler*. [cit. 2015-02-24]. Dostupné z: http://asynkronix.se/internet-of-things-with-arduino-yun-and-yaler/

[17] Arduino: *EEPROM Library*. [cit. 2015-03-08]. Dostupné z: http://www.arduino.cc/en/Reference/EEPROM

[18] Arduino: *Bridge Library for Arduino Yún*. [cit. 2015-03-14]. Dostupné z: http://arduino.cc/en/Tutorial/Bridge

[19] Arduino: *YunClient Reference*. [cit. 2015-03-15]. Dostupné z: http://arduino.cc/en/Reference/YunClientConstructor

[20] Arduino: *YunServer Reference*. [cit. 2015-03-15]. Dostupné z: http://arduino.cc/en/Reference/YunServerConstructor

[21] Arduino: *FileIO Reference*. [cit. 2015-03-15]. Dostupné z: http://arduino.cc/en/Reference/YunFileIOConstructor

[22] Arduino: *How to expand the Yún disk space*. [cit. 2015-03-22]. Dostupné z: http://arduino.cc/en/Tutorial/ExpandingYunDiskSpace

[23] Arduino: *Arduino Language Reference*. Dostupné z: http://arduino.cc/en/Reference/HomePage

[24] Arduino: *Arduino Yún Guide*. [cit. 2015-04-14]. Dostupné z: arduino.cc/en/Guide/ArduinoYun

[25] Arduino: *Temperature Web Panel*. [cit. 2015-04-14]. Dostupné z: http://arduino.cc/en/Tutorial/TemperatureWebPanel

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

AWS Amazon Web Services

CAN Controller Area Network

EEPROM Electrically Erasable Programmable Read-Only Memory

GPIO General Purpose Input/Output

I2C Inter-Integrated Circuit

IAM Identity and Access Managment

IDE Integrated Development Environment

ICSP In-Circuit Serial Programming

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

LCD Liquid-Crystal Display

LED Light-Emitting Diode

MAC Media Access Control

MIPS Microprocessor without Interlocked Pipeline Stages

OPKG Open PacKaGe Management

PWM Pulse Width Modulation

RTC Real Time Clock

RAM Random Access Memory

REST Representational State Transfer

SD Secure Digital

SPI Serial Peripheral Interface

SoC System on Chip

SQS Simple Queue Service

SSH Secure Shell

SRAM Static Random Access Memory

UART Universal Asynchronous Receiver/Transmitter

URL Uniform Resource Locator

USB Universal Serial Bus

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
    - `Zpráva.tex`
  - `impl` ..... zdrojové kódy implementace
    - `ArduinoWebDimmer` ..... Projekt pro Arduino IDE
      - `Bulb` ..... Implentace třídy Bulb
        - `Bulb.cpp`
        - `Bulb.h`
      - `www` ..... Webové rozhraní
        - `help.html`
        - `index.html`
        - `led1.html`
        - `led2.html`
        - `led1Setup.html`
        - `led2Setup.html`
        - `rtcSetup.html`
        - `styly.css`
        - `ledScript.js`
        - `script.js`
        - `zeptmo.min.js`
      - `ArduinoWebDimmer.ino` ..... Hlavní program
- `text` ..... text práce
  - `Zpráva.pdf` ..... text práce ve formátu PDF