

**ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE**

**Fakulta elektrotechnická**

**Katedra ekonomiky, manažerství a humanitních věd**

Realizace osvětlení akvária

Akvarium Light Design

Individuální projekt

Studijní program: Elektrotechnika, energetika a management

Název předmětu: A1B13IND

Vedoucí práce: Ing. Brejcha Michal, Ph.D.

**Zpracoval: Lukáš Pozníček**

**Praha 2016**

Zadání individuálního projektu

Název tématu:

**Realizace osvětlení akvária**

**Vedoucí**: Ing. Brejcha Michal, Ph.D.

**Student**: Lukáš Pozníček

**Katedra:** Elektrotechnologie K13113

**Zadání:**

* Zvolte zdroje světla pro akvárium.
* Seznamte se s platformou ARDUINO a způsobem programování.
* Navrhněte konstrukci osvětlení terária a jeho propojení s řídícím modulem.
* Vytvořte program umožňující regulaci osvětlení akvária.

**Literatura:**

1. HAASZ, Vladimír. *Elektrická měření: přístroje a metody*. Vyd. 2. Praha: Vydavatelství ČVUT, 2005, 337 s. ISBN 80-010-2731-7.
2. EVANS, Brian. *Beginning Arduino programming*. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2011, xx, 252 p. ISBN 14-302-3777-5.
3. VOBECKÝ, Jan a Vít ZÁHLAVA. *Elektronika: součástky a obvody, principy a příklady*. 3., rozš. vyd. Praha: Grada Publishing, 2005, 220 s. ISBN 80-247-1241-5.
4. Další literatura podle pokynů zadavatele

Obsah

[ÚVOD 4](#_Toc439677858)

[1. TEORETICKÁ ČÁST 5](#_Toc439677859)

[1.1. ARDUINO 5](#_Toc439677860)

[1.1.1. HISTORIE 5](#_Toc439677861)

[1.1.2. HARDWARE 5](#_Toc439677862)

[1.1.3. VÝVOJOVÉ PROSTŘEDÍ A PROGRAMOVACÍ JAZYK 6](#_Toc439677863)

[1.2. LED OSVĚTLENÍ 8](#_Toc439677864)

[1.2.1. SVĚTELNÉ ZDROJE 8](#_Toc439677865)

[1.2.2. FUNKCE 9](#_Toc439677866)

[1.2.3. ELEKTRICKÉ VLASTNOSTI 9](#_Toc439677867)

[1.2.4. REGULACE JASU LED 10](#_Toc439677868)

[1.3. POTŘEBY AKVÁRIA 11](#_Toc439677869)

[1.3.1. POTŘEBNÝ VÝKON LED OSVĚTLENÍ 13](#_Toc439677870)

[1.3.2. VÝBĚR 13](#_Toc439677871)

[2. PRAKTICKÁ ČÁST 15](#_Toc439677872)

[2.1. OSVĚTLOVACÍ RAMPA 15](#_Toc439677873)

[2.2. VÝKONOVÁ DESKA 15](#_Toc439677874)

[2.3. SCHÉMA ZAPOJENÍ 17](#_Toc439677875)

[2.4. PROGRAMOVÁNÍ 18](#_Toc439677876)

[2.4.1. SÉRIOVÁ KOMUNIKACE 18](#_Toc439677877)

[2.4.2. I2C 18](#_Toc439677883)

[2.4.3. RTC 19](#_Toc439677884)

[2.4.4. EEPROM 19](#_Toc439677885)

[2.4.5. PWM 19](#_Toc439677886)

[2.5. PROGRAM PRO ARDUINO 20](#_Toc439677887)

[ZÁVĚR 24](#_Toc439677888)

[POUŽITÁ LITERATURA 25](#_Toc439677889)

[PŘÍLOHY 26](#_Toc439677890)

# ÚVOD

Akvaristika se v České republice těší velké popularitě a i díky tomu patří ČR v současné době k největším vývozcům akvarijních ryb na světě. Mnoho akvaristů vynakládá značné úsilí, aby svým chovancům v nádrži zabezpečili podmínky, které věrně napodobují jejich přirozené prostředí. Toho se dosahuje především pečlivou kontrolou mnoha různých parametrů vody, množství přiváděných živin, kvalitním osvětlením a z toho plynoucí co nejvěrnější imitace denního osvětlení. A právě zmíněný poslední bod je pro tento projekt nejdůležitější.

Cílem toho projektu je vytvořit základ pro zařízení, které bude sloužit pro věrnou imitaci průběhu slunečního světla během dne a měsíčního svitu během noci. Při realizaci tohoto projektu je nutné brát ohledy na potřeby akvária, realizovatelnost zařízení a v neposlední řadě také na program pro obsluhu.

Zařízení by tedy mělo být schopné zpracovat uživatelsky zadané hodnoty. Na základě těchto hodnot by mělo být za pomocí PWM regulace ovládané LED osvětlení, které bude co nejvěrněji imitovat východ slunce, denní svit, západ slunce a shodné cykly i pro měsíční svit. Všechny tyto parametry by mělo být možné zadat bez nutnosti úpravy zdrojového kódu s příslušným opatřením proti zadání neplatných hodnot. Zařízení musí být také schopno poradit si s výpadky proudu. Při splnění těchto podmínek by sestava měla mít přijatelný estetický vzhled zkombinovaný s maximální funkčností.

# TEORETICKÁ ČÁST

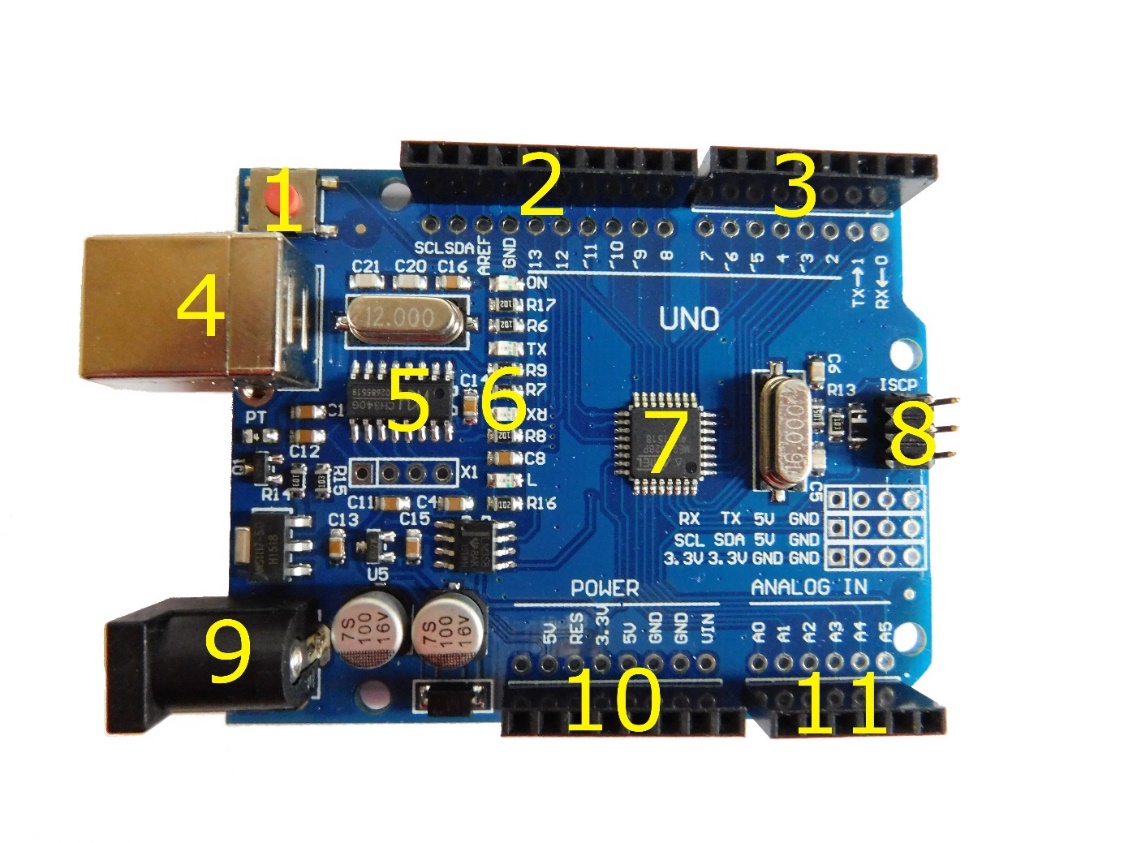
## ARDUINO

### HISTORIE

Arduino pochází z italského Interaction Design Institute, kde se v roce 2005 rozhodli vytvořit dostupný vývojový kit. Programová část byla založena na programovacím jazyku Proccesing, který má vlastní editor určený k výuce programování. Za dobu existence arduina již vzniklo spoustu různých typů a jelikož se jedná o opensource projekt, vzniklo také mnoho neoficiálních klonů. Každé Arduino je založeno na procesoru firmy Atmel, který doplňují další elektronické komponenty. Pro všechny doposud vytvořené desky je typické jednotné grafické zpracování s převažující modrou barvou. Mnoho typů také nese přídavek ke jménu například „Rev3“, který označuje verzi dané desky. Od té původní se většinou liší jen drobně, například vhodnějším uspořádáním součástek apod.

### HARDWARE

Pro tento projekt byla vybrána deska s názvem Arduino Uno, která je v současné době asi jedna z nejpoužívanějších. Tato deska je osazena procesorem ATmega328, který je vzhledem k požadavkům tohoto projektu zcela dostačující. Výrobcem těchto čipů je norská firma Atmel. Čipy ATmega patří do rodiny čipů s architekturou AVR, což jsou osmibitové procesory typu RISC s harvardskou architekturou (tzn. má oddělený paměťový prostor pro program a pro data). Jednotlivé použité typy (ATmega8, ATmega168, ATmega328) se liší především velikostí vnitřní paměti RAM a vnitřní paměti FLASH. Každý čip obsahuje kromě CPU i některé periferie – několik časovačů, sériové a paralelní porty, A/D převodníky pro zjišťování analogových hodnot atd. Na desce Arduino je kromě čipu i několik dalších obvodů, především to jsou stabilizátory napájecího napětí a obvody, které zajišťují komunikaci s PC.



*deska Arduino UNO*

Popis desky:

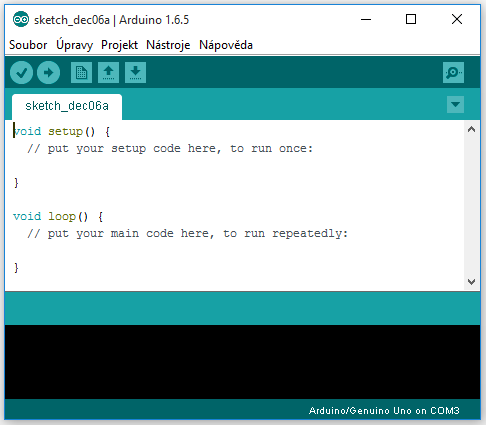
1. resetovací tlačítko; používá se, pokud chceme program spustit znovu od začátku
2. – 3. digitální piny; vývody označené ~ podporují PWM modulaci
3. USB konektor typu B; slouží k propojení desky a PC
4. USB seriál převodník – převodník univerzální sériové linky na USB; zajišťuje komunikaci mezi čipem a PC
5. Indikační LED
6. Hlavní čip desky
7. ICSP konektor pro externí programování hlavního čipu
8. Napájecí konektor
9. Napájecí výstupy procesorového kitu pro případné propojení s dalšími moduly
10. Analogové vstupy; používají si pro měření analogových hodnot

### VÝVOJOVÉ PROSTŘEDÍ A PROGRAMOVACÍ JAZYK

Vývojové prostředí pro psaní kódů se nazývá Arduino IDE (Integrated Development Enviroment). Je napsané v jazyce Java a jedná se o software vzniklý z vývojového prostředí Processing. To bylo mírně upraveno, byly přidány funkce a podpora jazyka Wiring.

Arduino je možné programovat v jazyce C nebo C++, nejvhodnější je ale použít knihovnu Wiring. Ta je v současnosti velmi rozšířená a díky její komplexnosti se o ní dá uvažovat jako o samostatném programovacím jazyce.

Pomocí vývojového prostředí můžeme vytvářet programy, ukládat je anebo také nahrávat přímo do Arduina. Prostředí má jednoduché ovládání, skládá se z několika částí: nahoře najdeme nabídky, pod nimi je panel s tlačítky, textový editor a dole je pruh pro zprávy, který nám oznamuje různé chybové hlášky. Dále se nesmí zapomenout na důležitou pomůcku, která se jmenuje Serial Monitor. Slouží k obousměrné komunikaci mezi stolním počítačem a Arduinem. S její pomocí lze po sériové lince přijímat a odesílat čísla nebo znaky.



*Arduino IDE*

## LED OSVĚTLENÍ

### SVĚTELNÉ ZDROJE

Světelnými zdroji se myslí tělesa vyzařující optické, zpravidla viditelné záření. Mohou to být buď přírodní zdroje (slunce, měsíc, blesk…) nebo umělé zdroje (svíčka, žárovka, výbojka, LED...). Umělý světelný zdroj je zdroj optického záření. Přeměňuje druh energie (v našem případě eklektické) na energii elektromagnetického záření v optickém intervalu světla.

Podle principu přeměny elektrické energie se světelné zdroje dělí na zdroje teplotní (obyčejné žárovky), výbojové (zářivky) a elektroluminiscenční (LED).

K nejdůležitějším technickým parametrům světelných zdrojů patří:

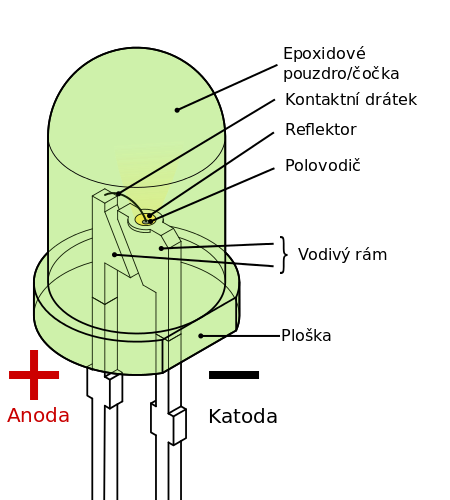
* konstrukční: vnější rozměry konstrukce přívodů, hmotnost, výška světelného středu, rozměry svítícího tělesa a další;
* elektrické: příkon světelného zdroje, napětí napájecí sítě, velikost a druh proudu a napětí na zdroji;
* světelné: světelný tok, spektrální složení, svítivost a její prostorové rozložení, jas;
* životnost světelného zdroje: celková doba jeho svícení až do okamžiku, kdy je nepoužitelný nebo se za nepoužitelný považuje podle potřebných kritérií.

K nejdůležitějším provozním parametrům patří:

* měrný výkon (účinnost): vyjádřený v lumenech na Watt (lm/W), charakterizuje efektivnost přeměny elektrické energie na světelnou;
* spolehlivost;
* kompatibilita: kompatibilita světelného zdroje s osvětlovací soustavou;
* ekonomičnost: cena světelného zdroje je jedním z hlavních faktorů určujících efektivnost osvětlovací soustavy a musí být posuzována s dalšími parametry, určujícími investiční a provozní náklady osvětlovací soustavy jako celku, včetně možných vlivů na bezpečnost a produktivitu práce.

### FUNKCE

LED (Light Emiting Diode) je polovodičová součástka vyzařující světlo. Jedná se o elektronickou součástku obsahující jeden PN přechod, který při průchodu elektrického proudu emituje optické záření. Pásmo spektra záření je závislé na chemickém složení použitého polovodiče. První diody vyzařovaly světlo červené barvy, posléze se objevily diody se zelenou, oranžovou, žlutou a nakonec modrou barvou. S poslední modrou barvou přišla i možnost vyvinutí diody bílé barvy, zářící v celé oblasti viditelného spektra. Bílé světlo lze získat dvěma způsoby. První spočívá v míšení světla červené, zelené a modré LED. Druhý způsob je využití fosforescence luminoforu.



*struktura LED*

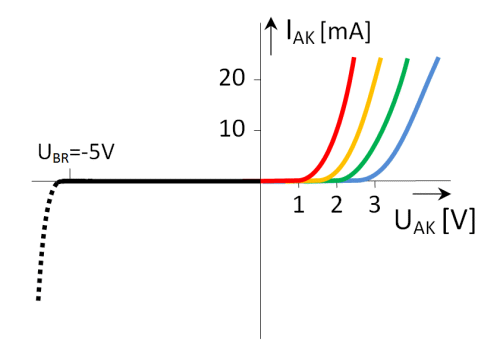
### ELEKTRICKÉ VLASTNOSTI

Voltampérová charakteristika LED má téměř stejný průběh jako běžná polovodičová dioda. Liší se pouze hodnotami hraničních napětí v propustném a závěrném směru.

Zvyšuje-li se napětí na diodě v propustném směru, zpočátku LED neprochází téměř žádný proud. Od určité hodnoty napětí dojde k zlomu a dioda se začne otevírat a na další drobné zvyšování napětí reaguje prudkým nárůstem procházejícího proudu. Proud vzhledem k napětí roste téměř exponenciálně. Zvyšuje-li se dále napětí, dioda zvyšuje svůj jas, až dosáhne svého maxima. Závislost svítivosti diody LED na proudu je v této oblasti téměř lineární. Při dalším zvyšování proudu nad povolenou mez může dojít k tepelnému přetížení polovodičového přechodu a poruše diody.

Připojí-li se dioda na zdroj napětí v závěrném směru, při jeho zvyšování od nuly diodou neprochází téměř žádný proud. Chová se nevodivě. Při určité úrovni napětí v závěrném směru dojde k průrazu a diodou začne náhle procházet velký proud. Vlivem oteplení bývá tento průraz ve většině případů destruktivní, polovodičový přechod je zkratován a dioda je trvale poškozena. Průrazné napětí v závěrném směru je u elektroluminiscenční diody oproti usměrňovacím diodám velmi nízké.

Vzhledem ke tvaru voltampérové charakteristiky se tyto diody musejí napájet ze zdroje proudu. Pokud se nevyžaduje vysoká účinnost zapojení, můžeme se přiblížit proudovému zdroji tím, že připojíme LED v sérii s rezistorem omezujícím protékající proud ke zdroji stálého napětí.



*VA charakteristika LED*

### REGULACE JASU LED

Nejjednodušší způsob regulace proudu protékajícího diodou je pomocí předřadného rezistoru. K regulaci jasu je možné použít i jednoduchý regulátor s tranzistorem. Pokročilejší způsob regulace využívá pulzně šířkovou modulaci (PWM - Pulse Width Modulation). Tehdy LED protéká pulzní proud. Pulzy musejí mít vyšší frekvenci, než je lidské oko schopné zachytit, což vyvolá zdání konstantního svitu. Změnou střídy těchto pulzů se mění jas. Pulzně šířková modulace se používá při napájení výkonových LED, kde by tepelná ztráta na předřadném rezistoru značně snižovala účinnost napáječe nebo při automatickém ovládání.



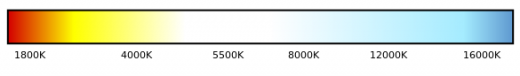
*příklad PWM modulace*

## POTŘEBY AKVÁRIA

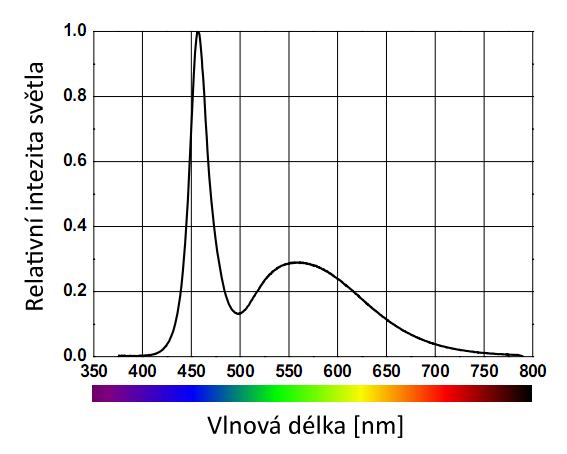
Výběr vhodného LED osvětlení musel být v tomto případě proveden především s ohledem na potřeby rostlin v akváriu. Akvarijní živočichové jsou totiž v dnešní době umělým chovem natolik zdegenerovaní, že podmínky vhodné pro rostliny budou plně dostatečné i pro samotné živočichy. Akvarijní rostliny vyžadují k růstu přiměřené množství světelné energie. Největšího přírůstku pak dosáhnou při optimální kombinaci časové délky a intenzity osvětlení a správné volbě zdroje světla. Nutným předpokladem je samozřejmě dodržení optimálních podmínek ostatních faktorů růstu rostlin v akváriu (teplota, koncentrace a správný poměr jednotlivých živin rozpuštěných ve vodě, zvláště pak dostatek oxidu uhličitého, alespoň minimální proudění vody atd.).

Jednotlivé složky slunečního záření mají pro růst rostlin různý vliv, který se mění jak během dne, tak během celkového života rostliny. Stejně tak se v přírodě mění intenzita dopadajícího světla, která závisí například na denní době, typu oblačnosti, hloubce vodního sloupce atd. Údaje o barevné teplotě záření a jeho výskytu jsou uvedeny v následující tabulce.

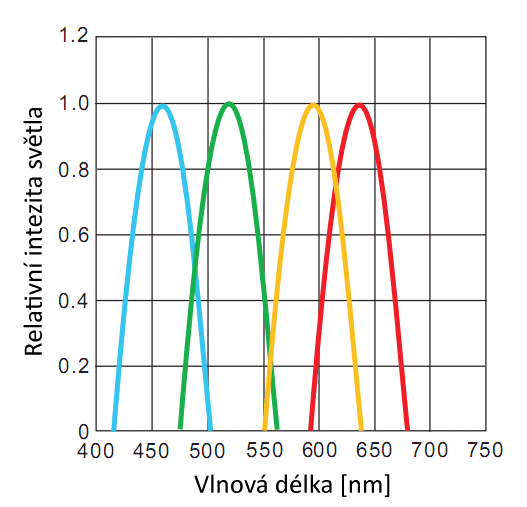
|  |  |
| --- | --- |
| Barevná teplota [K] | Výskyt |
| 1 850 | Sluneční záření při východu slunce |
| 2 380 | Sluneční záření 1/2 hodiny po východu slunce |
| 3 500 | Sluneční záření 1 hodinu po východu slunce |
| 4 000 | Sluneční záření 1 a 1/2 hodiny po východu slunce |
| 4 100 | Měsíční světlo |
| 6 000 | Jasné polední světlo |
| 7 000 | Lehce zamračená obloha |
| 8 000 | Oblačno, mlhavo |
| 10 000 | Silně zamračená obloha |



*chromatičnost různých barevných teplot*



*vlnová délka bílé LED*



*vlnové délky barevných LED*

### POTŘEBNÝ VÝKON LED OSVĚTLENÍ

Základním parametrem jsou rozměry akvária. Akvárium, které má délku 50 cm, je ideální osvětlovat páskem, který je stejně dlouhý. Z rozměrů plyne další veličina, a tou je objem. Výkon by měl být spíše naddimenzovaný, neboť se dá posléze díky PWN regulaci opět snížit. Pro klasická akvária se volí konstanta 0,15 (objem akvária [V]·0,15 [k] = výkon [W]) a pro akvária s náročnějšími rostlinami konstanta 0,3. [8] Při výběru LED pásků se udává poměr teplé a studené barvy v poměru 2:1. Níže je uveden výpočet pro akvárium o rozměrech 50x25x30 cm, na které je osvětlovací rampa dimenzována.

### VÝBĚR

#### Základní požadavky kladené na zdroj umělého osvětlení pro akvária:

* Zajištění optimálního růstu rostlin
* Hospodárný a bezpečný provoz
* Vytvoření příznivého estetického vjemu

#### Rozdělení LED pásků:

* Jednobarevné pásky bez silikonu

Pásek není zalit silikagelem. LED pásek je vhodný do profilů a na místa kde nevyžaduje ochranu proti prachu a vodě.

* Jednobarevné pásky v silikonu

Celý pásek je zalitý transparentním silikagelem, což umožňuje umístit jej do vlhkého prostředí. LED pásek se dá dobře udržovat v čistotě a je zde malá pravděpodobnost poškození.

* Vícebarevné RGB pásky bez silikonu

Provedení viz „Jednobarevné pásky bez silikonu“. Pásek obsahuje tři základní barvy (RGB), ale díky řídící jednotce dokáže tvořit více barevných kombinací.

* Vícebarevné RGB pásky v silikonu

Provedení viz „Jednobarevné pásky v silikonu“. Pásek obsahuje tři základní barvy (RGB), ale díky řídící jednotce dokáže tvořit více barevných kombinací.

* Speciální voděodolné pásky IP 68

Pásky s nejvyšším stupeň ochrany proti vodě a jsou určeny pro trvalé ponoření.

Pro akvárium byly vybrány dva pásky bez silikonu denní bílé a jeden RGB pásek bez silikonu pro simulaci východu slunce, západu slunce a měsíčního svitu. Celkový příkon této kombinace je 7,2 W (viz tabulka 1).

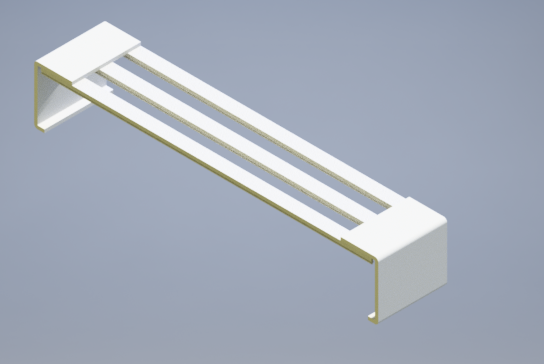
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| počet | délka | výkon | typ | výsledný výkon |
| [ks] | [cm] | [W/m] | [-] | [W/m] |
| 2 | 45 | 4,8 | denní bílá | 4,32 |
| 1 | 40 | 7,2 | RGB | 2,88 |

*tabulka 1*

# PRAKTICKÁ ČÁST

## OSVĚTLOVACÍ RAMPA

Osvětlovací rampa je vyrobena ze dvou ramen ohýbaného plexiskla a tří hliníkových lišt. V hliníkových lištách jsou umístěny dva pásky teplé bílé na krajích a uprostřed jeden RGB pásek. Všechny lišty jsou opatřeny difuzory, které mají za úkol bránit před přímým kontaktem s vodou a znečištění. Prostřední difuzor, kryjící RGB pásek, je jako jediný mléčné barvy (zbývající jsou čiré) a to z důvodu lepšího optického slití generovaných barev. Detailní výkres je uveden v příloze č. 2.

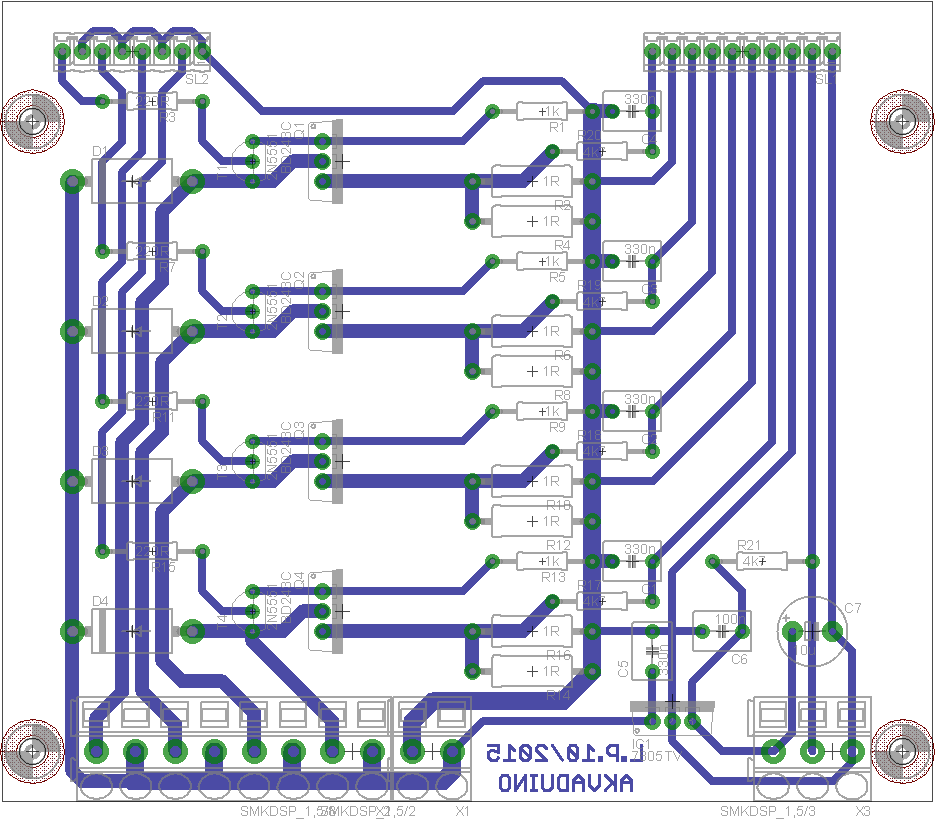


*Vizualizace osvětlovací rampy*

## VÝKONOVÁ DESKA

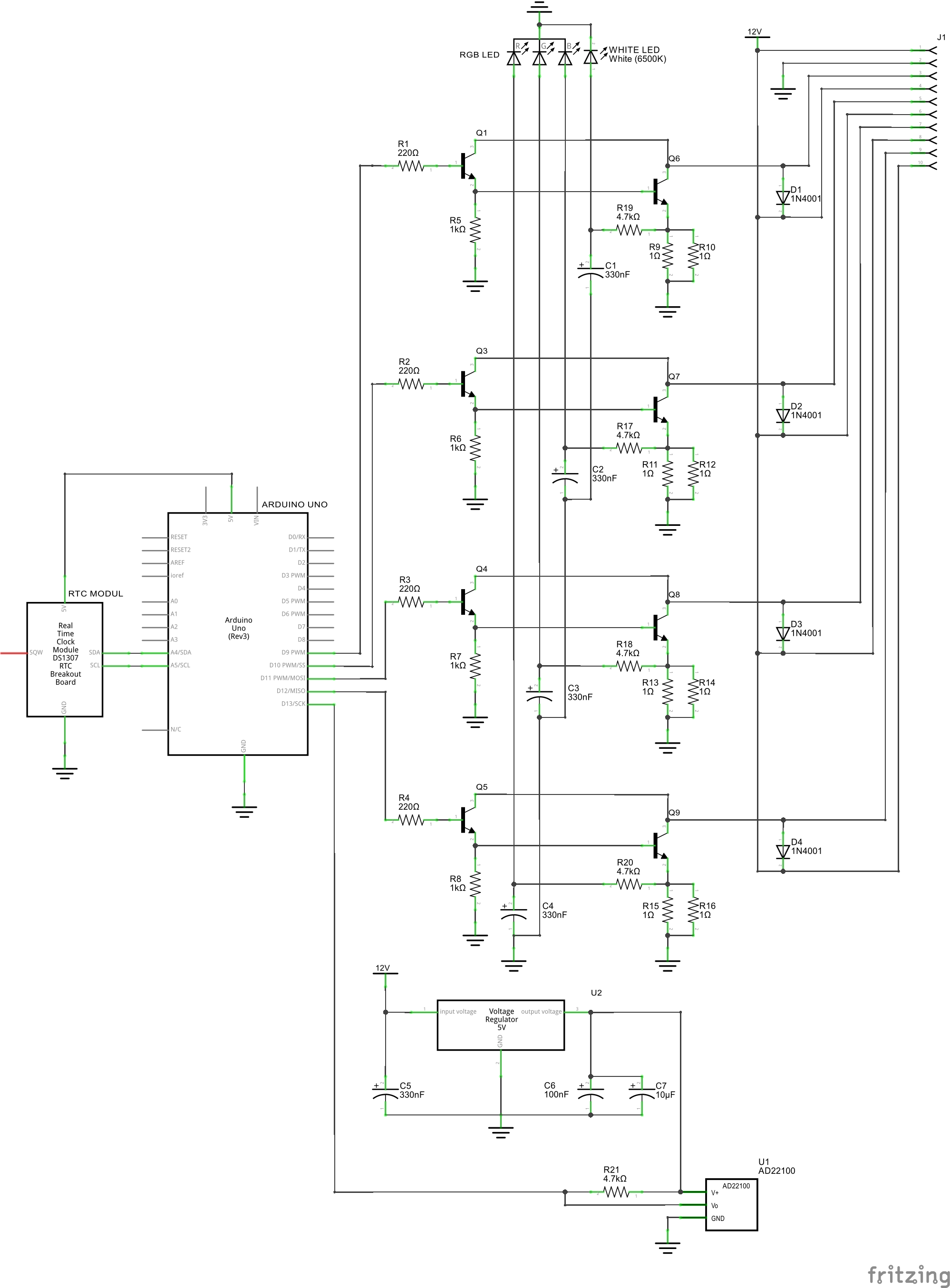
Výkonová deska slouží primárně pro napájení LED osvětlení pomocí přivedených PWM signálů. Napájení je realizováno pomocí Darligntonova zapojení dvou bipolárních tranzistorů. Toto zapojení se využívá tam, kde je třeba zvýšit proudové zesílení výkonových tranzistorů. V tomto případě slouží pro zesílení PWM signálu z desky Arduina, která nemá potřebný výkon na to, aby mohla přímo napájet osvětlovací rampu. Na výkonové desce se nachází toto zapojení čtyřikrát. Tři slouží pro napájení RGB pásku a jedno slouží pro napájení bílého pásku.

Sekundární vlastností desky je možnost snímání teploty pomocí teplotního čidla, které zde má ro tento účel vlastní stabilizátor napětí. Poslední využití je pro sledování střední hodnoty protékaných proudů, které může sloužit například pro výpočet spotřeby energie nebo kontrole funkčního stavu LED pásků. Obě tyto možnosti jsou realizovány tak, že pro jejich využití stačí snímat signál z kolíkové lišty SL1 a elektronicky jej zpracovat. Tyto možnosti nejsou momentálně využity a budou aplikovány až při dalším rozvoji zařízení.



*Layout výkonové desky*

## SCHÉMA ZAPOJENÍ



*Schéma zapojení celého zařízení*

## PROGRAMOVÁNÍ

### SÉRIOVÁ KOMUNIKACE

Sériová komunikace RS232/ USB je základní prostředek jak z PC komunikovat s Arduinem. Dá se využít i k jiným věcem, než je jen programování, a to například ke čtení hodnot proměnných (paměti) Arduina, na kterém právě běží určitý program. Pomocí sériové linky se také dají posílat jednoduché textové příkazy. Ke zmíněnému čtení hodnot v textové podobě slouží Serial monitor, který je součástí programovací prostředí ArduinoIDE.

Příklad příkazů pro použití sériové komunikace:

|  |  |
| --- | --- |
| Příkaz | Popis |
| Serial.begin(val) | zahájení komunikace |
| Serial.print(text) | Posílá ASCII data na stejném řádku |
| Serial.println(text) | na konci ASCII dat zalomí řádku |
| Serial.avaiable() | vrací počet bytů dostupných v bufferu |
| Serial.read() | vezme první byte z bufferu a přečte ho |
| Serial.end() | ukončí komunikaci |



### I2C

I2C je multi-masterová počítačová sériová sběrnice, která je používána k připojování nízkorychlostních periferií k základní desce a která umožňuje komunikovat po dvou vodičích až se 128 zařízeními. Jeden vodič je pro udávání taktu (SCL – clock line), druhý pro data (SDA – data line) a každé zařízení má svoji 7bitovou adresu. U tohoto typu komunikace je vždy jedno zařízení řídící (master) a ostatní jsou řízena (slave). Pro komunikaci je třeba implementovat knihovnu *Wire.h*.

Příklad příkazů pro použití I2C:

|  |  |  |
| --- | --- | --- |
| Příkaz | Použití | Popis |
| Wire.begin(adr) | master, slave | připojí zařízení ke sběrnici |
| Wire.beginTransmission(adr) | master | zahájí komunikaci se zařízením |
| Wire.endTransmission() | master | ukončí komunikaci se zařízením |
| Wire.requestFrom(adr,p) | master | žádání master o data od slave |
| Wire.available() | master, slave | vrací počet bytů čekajících na zpracování |
| Wire.write(data,delka) | master, slave | odeslání informací po I2C |
| Wire.read() | master, slave | čtení dat z I2C |

### RTC

Real Time Clock (RTC), česky hodiny reálného času, jsou počítačové hodiny (většinou ve formě integrovaného obvodu), které udržují údaj o aktuálním čase. Výhodou toho zařízení je jeho malá spotřeba, a tedy možnost napájení z baterie. Ulehčuje procesoru počítání času a díky napájení z baterie běží i při přerušení napájení.

V tomto případě je použit RTC modul s označením DS3231. Jeho přesnost je zajišťována vnitřním oscilátorem, který není ovlivňován vnějšími faktory, a jeho odchylka se pohybuje v řádu jednotek minut za rok. Modul je připojen pomocí již zmíněné sběrnice I2C. Zdrojový kód pro práci s tímto modulem je komplikovanější, a nelze tedy pouze vytknou příklad příkazů.

### EEPROM

EEPROM (Electrically Erasable Programmable Read-Only Memory) je paměť, která uchová uložené hodnoty i v případě vypnutí nebo restartování zařízení. Tato paměť se využívá především u zařízení, kde nedochází často k přepisům paměti. Velikost této paměti u použité desky Arduino UNO je 1 kB. Za zmínku také stojí zajímavost, že u pamětí EEPROM se uvádí životnost 100 000 cyklů. Pro použití je třeba implementovat knihovnu *EEPROM.h*.

Příklad příkazů pro použití EEPROM:

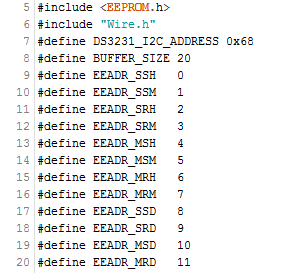
|  |  |
| --- | --- |
| Příkaz | Popis |
| EEPROM.write(adr, val) | zapíše na adresu *adr* hodnotu *val* |
| EEPROM.read(adr) | přečte hodnotu na adrese *adr* |
| EEPROM.lenght() | vrátí velikost paměti EEPROM |

### PWM

Vzhledem k tomu, že Arduino nemá analogový výstup, používá digitální výstup, který je označen vlnovkou (~). Těchto výstupů má Arduino pět a je možné je použít pro pulzně šířkovou modulaci (PWM) a ovládají se pomocí funkce *analogWrite(pin, hodnota)*, kde pin je číslo výstupního pinu a hodnota je číslo v rozsahu jednoho neznaménkového bajtu, tedy 0 až 255. Frekvence výstupního signálu je přibližně 490 Hz. V tomto případě jsou použity výstupy s PWM modulací čtyři. Tři slouží pro modulaci RGB pásku a jeden slouží pro modulaci bílého pásku.

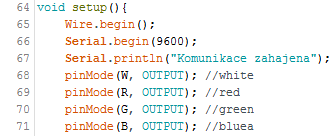
## PROGRAM PRO ARDUINO

Protože celý zdrojový kód obsahuje více než 600 řádků, je v této kapitole popsáno jen pár důležitých částí. Kód v plném rozsahu je uveden jako Příloha č. 1.



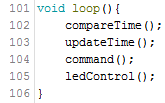
Výpis 1: Symbolické konstanty

Jedná se úplný počátek kódu, který obsahuje implementaci knihoven <EEPROM.h> {5} pro využití paměti a „Wire.h“ {6} pro komunikaci prostřednictvím I2C. Dále je zde deklarovaná velikost bufferu {8}, který se chová jako pole a slouží k přijímání jednoduchých textových příkazů. Na dalších řádcích {9-20} je deklarace adres v paměti EEPROM pro jednotlivá data, konkrétně údaje o časech spuštění jednotlivých cyklů.



Výpis 2: Inicializace programu

Void setup() {64} je jedna ze základních funkcí, kterou musí obsahovat každý kód. V této funkci jsou příkazy, které proběhnou pouze jednou při spuštění programu. V tomto případě jde o spuštění komunikace I2C {66}, spuštění komunikace po sériové lince {67}, výpis kontrolního hlášení o navázání komunikace {67} a nastavení pinů, které budou sloužit jako výstupy PWM modulace {68-71}.



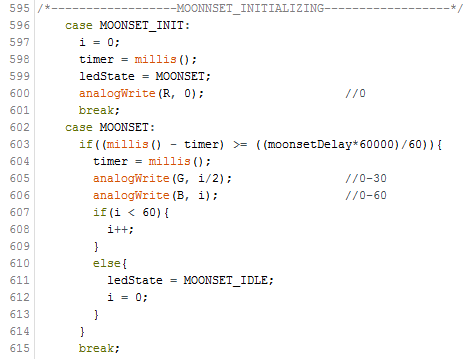
Výpis 3: Hlavní programová smyčka

Void loop() {101} je druhá ze základních funkcí, kterou musí obsahovat každý kód. V této funkci jsou příkazy, které budou probíhat stále dokola, dokud nedojde k odpojení zařízení od napájení. Zde se nacházejí funkce, které jsou postupně vyvolávány. Funkce compareTime() {102} porovnává uživatelem zadané časy jednotlivých cyklů svícení a při souhlasu s aktuálním časem spustí daný cyklus. Funkce updateTime() {103} aktualizuje čas z RTC modulu každých 5 vteřin. Funkce command() {104} slouží ke čtení bufferu a přijímání uživatelem zadaných příkazů. Funkce ledControl() {105} slouží řízení cyklu osvětlení LED pásků.



Výpis 4: Identifikace příkazu zasleného po sériové lince - funkce command()

Zde je výše zmíněná funkce command() {203}, která sleduje znaky přicházející po sériové lince do bufferu. V případě že nalezne znak „#“, spustí zapisování a to až do doby, než narazí na znak „$“. Po přijmutí takovéhoto řetězce analyzuje znaky mezi nimi. V případě, že přijde například řetězec v podobě příkazu „#SAH08$“, nastaví hodinu východu slunce na hodnotu 8. Příkaz „#SAM30$“ nastaví minuty východu slunce na hodnotu 30 a výsledný čas východu slunce bude 8:30. Funkce má také základní ošetření proti případům, kdy dojde k zadání nesprávné hodnoty.



Výpis 5: Část stavového automatu ve funkci ledControl()

Tato ukázka je část kódu z funkce ledControl() {486} a inicializuje východ měsíce. V praxi funguje tak, že na piny G (zelená) a B (modrá) bude posílat PWM modulaci s postupně zvyšující se střídou a to až do hodnoty 60 z 255 možných. Prodlevu mezi jednotlivými nárůsty určuje uživatel a to definováním hodnoty moonsetDelay {603} v minutách, které určují celkový čas tohoto cyklu. Výsledné složení barev po dokončení toho cyklu bude R:0, G:30 a B:60.

# ZÁVĚR

Podařilo se vytvořit základ pro zařízení, které je v současném stavu schopno sloužit pro věrnou imitaci průběhu slunečního světla během dne. Prozatím je schopné zpracovat jednoduché textové příkazy zadané pomocí programovacího softwaru ArduinoIDE. Na základě těchto příkazů a za pomocí PWM regulace ovládá LED osvětlení, které se snaží co nejvěrněji imitovat východ slunce, denní svit, západ slunce a shodné cykly i pro měsíční svit. Zařízení je schopné poradit si s odpojením od napájení, tedy i s výpadky proudu a po opětovném spuštění naváže cyklem, který odpovídá času v RTC.

V rámci bakalářské práce by zařízení mělo být zdokonaleno. Hlavním cílem bude připojení a zprovoznění WifiShieldu, který bude umožňovat ovládat zařízení z notebooku nebo chytrých zařízení pomocí Wi-Fi připojení. Dále by se měl rozrůst uživatelský přístup do ovládání zařízení, mělo by být možné přepnout zařízení do manuálního módu a v reálném čase nastavovat na osvětlovací rampě libovolné barvy. V poslední řadě bude nutné vytvořit ekonomickou část tohoto projektu v podobě nákladové analýzy a provozních nákladů.

# POUŽITÁ LITERATURA

1. HAASZ, Vladimír. *Elektrická měření: přístroje a metody*. Vyd. 2. Praha: Vydavatelství ČVUT, 2005, 337 s. ISBN 80-010-2731-7.
2. EVANS, Brian. *Beginning Arduino programming*. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2011, xx, 252 p. ISBN 14-302-3777-5.
3. VOBECKÝ, Jan a Vít ZÁHLAVA. *Elektronika: součástky a obvody, principy a příklady*. 3., rozš. vyd. Praha: Grada Publishing, 2005, 220 s. ISBN 80-247-1241-5.
4. HEBL, Jiří, Karel DVOŘÁČEK, Vladimír DVOŘÁČEK, Petr ŽÁK, *Světlo a osvětlování.* FCC Public s.r.o, 2013, 622 s. ISBN 978-80-86534-21-3.
5. ZÁHLAVA, Vít. *Návrh a konstrukce desek plošných spojů*. Praha: Vydavatelství ČVUT, 2005, 77 s. ISBN 80-01-03351-1.
6. VODA, Zbyšek & tým HW Kitchen. *Průvodce světem Arduina.* Nakladatelství Martin Stříž, 2015, 240 s. ISBN 978-80-87106-90-7.
7. KROUPA, Miloš. *Akvárium příručka pro začátečníky*. Nakladatelství Robimaus, 2010, 72 s. ISBN 978-80-87293-11-9.
8. Jak vybrat LED osvětlení do akvária. Rostlinna-akvaria.cz [online]. [cit. 2016-01-02]. Dostupné z: <http://www.rostlinna-akvaria.cz/jak-vybrat-led-osvetleni-do-akvaria-147>

# PŘÍLOHY

**PŘÍLOHA Č. 1**

#include <EEPROM.h>

#include "Wire.h"

#define DS3231\_I2C\_ADDRESS 0x68

#define BUFFER\_SIZE 20

#define EEADR\_SSH 0

#define EEADR\_SSM 1

#define EEADR\_SRH 2

#define EEADR\_SRM 3

#define EEADR\_MSH 4

#define EEADR\_MSM 5

#define EEADR\_MRH 6

#define EEADR\_MRM 7

#define EEADR\_SSD 8

#define EEADR\_SRD 9

#define EEADR\_MSD 10

#define EEADR\_MRD 11

byte sunsetHour; //vychod slunce hodina

byte sunsetMinute; //vychod slunce minuta

byte sunriseHour; //zapad slunce hodina

byte sunriseMinute; //zapad slunce minta

byte moonsetHour; //vychod mesice hodina

byte moonsetMinute; //vychod mesice minuta

byte moonriseHour; //zapad mesice hodina

byte moonriseMinute; //zapad mesice minuta

byte sunsetDelay; //vychod slunce trvani

byte sunriseDelay; //zapad slunce trvani

byte moonsetDelay; //vychod mesice trvani

byte moonriseDelay; //zapad mesice trvani

int W = 5;

int R = 10;

int G = 9;

int B = 11;

typedef enum{

IDLE\_STATE,

SUNSET\_IDLE, SUNSET\_INIT, SUNSET\_SOON, SUNSET\_MIDDLE, SUNSET\_LATE,

SUNRISE\_IDLE, SUNRISE\_INIT, SUNRISE\_SOON, SUNRISE\_MIDDLE, SUNRISE\_LATE,

MOONSET\_IDLE, MOONSET\_INIT, MOONSET,

MOONRISE\_IDLE, MOONRISE\_INIT, MOONRISE

}

ledState\_t;

ledState\_t ledState;

// Convert normal decimal numbers to binary coded decimal

byte decToBcd(byte val){

return( (val/10\*16) + (val%10) );

}

// Convert binary coded decimal to normal decimal numbers

byte bcdToDec(byte val){

return( (val/16\*10) + (val%16) );

}

// Convert ascii symbols to decimal numbers

byte asciiToDec(byte tens, byte ones){

return((tens-'0')\*10) + (ones-'0');

}

void setup(){

Wire.begin();

Serial.begin(9600);

Serial.println("Komunikace zahajena");

pinMode(W, OUTPUT); //white

pinMode(R, OUTPUT); //red

pinMode(G, OUTPUT); //green

pinMode(B, OUTPUT); //bluea

//inicializace casu z EEPROM

sunsetHour = EEPROM.read(EEADR\_SSH);

sunsetMinute = EEPROM.read(EEADR\_SSM);

sunsetDelay = EEPROM.read(EEADR\_SSD);

sunriseHour = EEPROM.read(EEADR\_SRH);

sunriseMinute = EEPROM.read(EEADR\_SRM);

sunriseDelay = EEPROM.read(EEADR\_SRD);

moonsetHour = EEPROM.read(EEADR\_MSH);

moonsetMinute = EEPROM.read(EEADR\_MSM);

moonsetDelay = EEPROM.read(EEADR\_MSD);

moonriseHour = EEPROM.read(EEADR\_MRH);

moonriseMinute = EEPROM.read(EEADR\_MRM);

moonriseDelay = EEPROM.read(EEADR\_MRD);

for (int r = 0 ; r < 2 ; r++){

for (int g = 0 ; g < 2 ; g++){

for (int b = 0 ; b < 2 ; b++){

digitalWrite(R,r);

digitalWrite(G,g);

digitalWrite(B,b);

delay(500);

}

}

}

digitalWrite(R,0);

digitalWrite(G,0);

digitalWrite(B,0);

}

void loop(){

compareTime();

updateTime();

command();

ledControl();

}

void updateTime(){

static long time = 0;

if((millis() - time) >= 5000){

time = millis();

//Serial.println(time/1000);

displayTime();

}

}

byte setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte month, byte year){

// sets time and date data to DS3231

if(second > 59 || minute > 59 || hour > 23 ||

dayOfWeek > 7 || dayOfWeek < 1 ||

dayOfMonth > 31 || dayOfMonth < 1 ||

month > 12 || month < 1|| year > 99)

{

return 0;

}

Wire.beginTransmission(DS3231\_I2C\_ADDRESS);

Wire.write(0); // set next input to start at the seconds register

Wire.write(decToBcd(second)); // set seconds

Wire.write(decToBcd(minute)); // set minutes

Wire.write(decToBcd(hour)); // set hours

Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)

Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)

Wire.write(decToBcd(month)); // set month

Wire.write(decToBcd(year)); // set year (0 to 99)

Wire.endTransmission();

return 1;

}

void readDS3231time(byte \*second, byte \*minute, byte \*hour, byte \*dayOfWeek, byte \*dayOfMonth, byte \*month, byte \*year){

Wire.beginTransmission(DS3231\_I2C\_ADDRESS);

Wire.write(0); // set DS3231 register pointer to 00h

Wire.endTransmission();

Wire.requestFrom(DS3231\_I2C\_ADDRESS, 7);

// request seven bytes of data from DS3231 starting from register 00h

\*second = bcdToDec(Wire.read() & 0x7f);

\*minute = bcdToDec(Wire.read());

\*hour = bcdToDec(Wire.read() & 0x3f);

\*dayOfWeek = bcdToDec(Wire.read());

\*dayOfMonth = bcdToDec(Wire.read());

\*month = bcdToDec(Wire.read());

\*year = bcdToDec(Wire.read());

}

void displayTime(){

byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

// retrieve data from DS3231

readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);

// send it to the serial monitor

Serial.print(hour, DEC);

// convert the byte variable to a decimal number when displayed

Serial.print(":");

if (minute<10){

Serial.print("0");

}

Serial.print(minute, DEC);

Serial.print(":");

if (second<10){

Serial.print("0");

}

Serial.println(second, DEC);

/\*Serial.print(" ");

Serial.print(dayOfMonth, DEC);

Serial.print("/");

Serial.print(month, DEC);

Serial.print("/");

Serial.print(year, DEC);

Serial.print(" Day of week: ");

switch(dayOfWeek){

case 1:

Serial.println("Sunday");

break;

case 2:

Serial.println("Monday");

break;

case 3:

Serial.println("Tuesday");

break;

case 4:

Serial.println("Wednesday");

break;

case 5:

Serial.println("Thursday");

break;

case 6:

Serial.println("Friday");

break;

case 7:

Serial.println("Saturday");

break;

}\*/

}

void command(){

static char pole[BUFFER\_SIZE];

static byte idx = 0;

char znak;

while(Serial.available() > 0){

znak = Serial.read();

if(znak == '#'){

idx = 0;

}

else if(znak == '$' && idx > 0){

pole[idx] = '\0';

Serial.print("[");

Serial.print(pole);

Serial.println("]");

znak = pole[0];

switch (znak) {

case 'S':

Serial.println("set\_time");

if (pole[1]=='A' && pole[2] == 'H' && asciiToDec(pole[3], pole[4]) <= 24){ //nastaveni hodiny vychodu slunce, prikaz #SAH..

sunsetHour = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_SSH, sunsetHour);

Serial.println("sunset\_hour\_time\_set");

}

else if (pole[1]=='A' && pole[2] == 'M' && asciiToDec(pole[3], pole[4]) <= 59){ //nastaveni minuty vychodu slunce, prikaz #SAM..

sunsetMinute = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_SSM, sunsetMinute);

Serial.println("sunset\_minute\_time\_set");

}

else if (pole[1]=='A' && pole[2] == 'D' && asciiToDec(pole[3], pole[4]) <= 99){ //nastaveni delay vychodu slunce, prikaz #SAD..

sunsetDelay = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_SSD, sunsetDelay);

Serial.println("sunset\_delay\_set");

}

else if (pole[1]=='P' && pole[2] == 'H' && asciiToDec(pole[3], pole[4]) <= 24){ //nastaveni hodiny zapada slunce, prikaz #SPH..

sunriseHour = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_SRH, sunriseHour);

Serial.println("sunrise\_hour\_time\_set");

}

else if (pole[1]=='P' && pole[2] == 'M' && asciiToDec(pole[3], pole[4]) <= 59){ //nastaveni minuty zapadu slunce, prikaz #SPM..

sunriseMinute = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_SRM, sunriseMinute);

Serial.println("sunrise\_minute\_time\_set");

}

else if (pole[1]=='P' && pole[2] == 'D'&& asciiToDec(pole[3], pole[4]) <= 99){ //nastaveni delay zapadu slunce, prikaz #SPD..

sunriseDelay = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_SRD, sunriseDelay);

Serial.println("sunrise\_delay\_set");

}

else if (pole[1]=='M' && pole[2] == 'H' && asciiToDec(pole[3], pole[4]) <= 24){ //nastaveni hodiny vychodu mesice, prikaz #SMH..

moonsetHour = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_MSH, moonsetHour);

Serial.println("moonset\_hour\_time\_set");

}

else if (pole[1]=='M' && pole[2] == 'M' && asciiToDec(pole[3], pole[4]) <= 59){ //nastaveni minuty vychodu mesice, prikaz #SMM..

moonsetMinute = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_MSM, moonsetMinute);

Serial.println("moonset\_minute\_time\_set");

}

else if (pole[1]=='M' && pole[2] == 'D' && asciiToDec(pole[3], pole[4]) <= 99){ //nastaveni delay vychodu mesice prikaz #SMD..

moonsetDelay = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_MSD, moonsetDelay);

Serial.println("moonset\_delay\_set");

}

else if (pole[1]=='T' && pole[2] == 'H' && asciiToDec(pole[3], pole[4]) <= 24){ //nastaveni hodiny zapadu mesice, prikaz #STH..

moonriseHour = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_MRH, moonriseHour);

Serial.println("moonrise\_hour\_time\_set");

}

else if (pole[1]=='T' && pole[2] == 'M' && asciiToDec(pole[3], pole[4]) <= 59){ //nastaveni minuty zapadu mesice, prikaz #STM..

moonriseMinute = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_MRM, moonriseMinute);

Serial.println("moonrise\_minute\_time\_set");

}

else if (pole[1]=='T' && pole[2] == 'D' && asciiToDec(pole[3], pole[4]) <= 99){ //nastaveni delay vychodu mesice prikaz #STD..

moonriseDelay = asciiToDec(pole[3], pole[4]);

EEPROM.write(EEADR\_MRD, moonriseDelay);

Serial.println("moonrise\_delay\_set");

}

else if (pole[1]=='T' && pole[2] == 'T'){ //nastaveni aktualniho casu RTC #STT SEC MIN HOD DVT DVM MES ROK

if(setDS3231time(asciiToDec(pole[3], pole[4]),

asciiToDec(pole[5], pole[6]),

asciiToDec(pole[7], pole[8]),

asciiToDec(pole[9], pole[10]),

asciiToDec(pole[11], pole[12]),

asciiToDec(pole[13], pole[14]),

asciiToDec(pole[15], pole[16])))

{

Serial.println("time\_set");

}

else

{

Serial.println("INCORRECT\_VALUES");

}

}

else

Serial.println("INCORRECT\_VALUES");

break;

case 'P':

if (pole[1]=='A'){ //vypise cas vychodu slunce, prikaz #PA

Serial.print("Sun set time: ");

if (sunsetHour < 10){

Serial.print("0");

}

Serial.print(sunsetHour, DEC);

Serial.print(':');

if (sunsetMinute < 10){

Serial.print("0");

}

Serial.println(sunsetMinute, DEC);

}

else if (pole[1]=='P'){ //vypise cas zapadu slunce, prikaz #PP

Serial.print("Sun rise time: ");

if (sunriseHour < 10){

Serial.print("0");

}

Serial.print(sunriseHour, DEC);

Serial.print(':');

if (sunriseMinute < 10){

Serial.print("0");

}

Serial.println(sunriseMinute, DEC);

}

else if (pole[1]=='M'){ //vypise cas vychodu mesice, prikaz #PM

Serial.print("Moon set time: ");

if (moonsetHour < 10){

Serial.print("0");

}

Serial.print(moonsetHour, DEC);

Serial.print(':');

if (moonsetMinute < 10){

Serial.print("0");

}

Serial.println(moonsetMinute, DEC);

}

else if (pole[1]=='N'){ //vypise cas zapadu mesice, prikaz #PN

Serial.print("Moon rise time: ");

if (moonriseHour < 10){

Serial.print("0");

}

Serial.print(moonriseHour, DEC);

Serial.print(':');

if (moonriseMinute < 10){

Serial.print("0");

}

Serial.println(moonriseMinute, DEC);

}

else if (pole[1]=='T'){ //vypise aktualni cas, prikaz #PT

Serial.print("Current time: ");

Serial.println("print\_time");

displayTime();

}

break;

case 'D':

if (pole[1]=='A'){ //vypise delay vychodu slunce, prikaz #DA

Serial.print("Sun set lenhgt: ");

Serial.print(sunsetDelay, DEC);

Serial.println(" min.");

}

else if (pole[1]=='P'){ //vypise delay zapaduu slunce, prikaz #DP

Serial.print("Sun rise lenhgt: ");

Serial.print(sunriseDelay, DEC);

Serial.println(" min.");

}

else if (pole[1]=='M'){ //vypise delay vychodu mesice, prikaz #DM

Serial.print("Moon set lenhgt: ");

Serial.print(moonsetDelay, DEC);

Serial.println(" min.");

}

else if (pole[1]=='N'){ //vypise delay zapadu mesice, prikaz #DN

Serial.print("Moon rise lenhgt: ");

Serial.print(moonriseDelay, DEC);

Serial.println(" min.");

}

break;

}

}

else{

if(idx >= (BUFFER\_SIZE - 1)){

Serial.println("ERROR\_LONG\_COMMAND");

idx = 0;

}

else{

pole[idx] = znak;

idx++;

}

}

}

}

void compareTime(){

ledState\_t state;

int deltaHourA, deltaHourB, deltaMinuteA, deltaMinuteB;

byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month, &year);

if(ledState == IDLE\_STATE || ledState == SUNSET\_IDLE || ledState == SUNRISE\_IDLE ||

ledState == MOONSET\_IDLE || ledState == MOONRISE\_IDLE)

{

//////////////////////////////////

state = SUNSET\_INIT;

deltaHourA = hour - sunsetHour;

if(deltaHourA < 0) deltaHourA = hour + 24 - sunsetHour;

deltaMinuteA = minute - sunsetMinute;

if(deltaMinuteA < 0) deltaMinuteA = minute + 60 - sunsetMinute;

deltaHourB = hour - sunriseHour;

if(deltaHourB < 0) deltaHourB = hour + 24 - sunriseHour;

deltaMinuteB = minute - sunriseMinute;

if(deltaMinuteB < 0) deltaMinuteB = minute + 60 - sunriseMinute;

if(deltaHourA > deltaHourB){

state = SUNRISE\_INIT;

deltaHourA = deltaHourB;

deltaMinuteA = deltaMinuteB;

}

else if(deltaHourA == deltaHourB){

if(deltaMinuteA > deltaMinuteB){

state = SUNRISE\_INIT;

deltaHourA = deltaHourB;

deltaMinuteA = deltaMinuteB;

}

}

deltaHourB = hour - moonsetHour;

if(deltaHourB < 0) deltaHourB = hour + 24 - moonsetHour;

deltaMinuteB = minute - moonsetMinute;

if(deltaMinuteB < 0) deltaMinuteB = minute + 60 - moonsetMinute;

if(deltaHourA > deltaHourB){

state = MOONSET\_INIT;

deltaHourA = deltaHourB;

deltaMinuteA = deltaMinuteB;

}

else if(deltaHourA == deltaHourB){

if(deltaMinuteA > deltaMinuteB){

state = MOONSET\_INIT;

deltaHourA = deltaHourB;

deltaMinuteA = deltaMinuteB;

}

}

deltaHourB = hour - moonriseHour;

if(deltaHourB < 0) deltaHourB = hour + 24 - moonriseHour;

deltaMinuteB = minute - moonriseMinute;

if(deltaMinuteB < 0) deltaMinuteB = minute + 60 - moonriseMinute;

if(deltaHourA > deltaHourB){

state = MOONRISE\_INIT;

deltaHourA = deltaHourB;

deltaMinuteA = deltaMinuteB;

}

else if(deltaHourA == deltaHourB){

if(deltaMinuteA > deltaMinuteB){

state = MOONRISE\_INIT;

deltaHourA = deltaHourB;

deltaMinuteA = deltaMinuteB;

}

}

//////////////////////////////////////////////////////////////////////

if (state == SUNSET\_INIT && ledState != SUNSET\_IDLE){

//sunSet();

ledState = SUNSET\_INIT;

}

if (state == SUNRISE\_INIT && ledState != SUNRISE\_IDLE){

//sunRise();

ledState = SUNRISE\_INIT;

}

if (state == MOONSET\_INIT && ledState != MOONSET\_IDLE){

//moonSet();

ledState = MOONSET\_INIT;

}

if (state == MOONRISE\_INIT && ledState != MOONRISE\_IDLE){

//moonRise();

ledState = MOONRISE\_INIT;

}

}

}

void ledControl(){

static int i = 0;

static unsigned long timer = 0;

/\*------------------SUNSET\_INITIALIZING------------------\*/

switch(ledState){

case SUNSET\_INIT:

i = 0;

timer = millis();

ledState = SUNSET\_SOON;

analogWrite(B, 0); //0

break;

case SUNSET\_SOON:

if((millis() - timer) >= ((((sunsetDelay\*60000)/8)\*1)/255)){

timer = millis();

analogWrite(R, i); //0-255

analogWrite(G, (i\*0.19)); //0-50

if(i < 255){

i++;

}

else{

ledState = SUNSET\_MIDDLE;

i = 255;

analogWrite(G, 50); //50

analogWrite(B, 0); //0

}

}

break;

case SUNSET\_MIDDLE: //i musi byt 255

if((millis() - timer) >= ((((sunsetDelay\*60000)/8)\*5)/35)){

timer = millis();

analogWrite(R, i); //255-220

if(i > 220){

i--;

}

else{

ledState = SUNSET\_LATE;

i = 0;

}

}

break;

case SUNSET\_LATE:

if((millis() - timer) >= ((((sunsetDelay\*60000)/8)\*2)/255)){

timer = millis();

analogWrite(R, 220+(i\*35/255)); //220-255

analogWrite(G, 50+(i\*205/255)); //50-255

analogWrite(B, i); //0-255

analogWrite(W, i); //0-255

if(i < 255){

i++;

}

else{

ledState = SUNSET\_IDLE;

i = 0;

}

}

break;

/\*------------------SUNRISE\_INITIALIZING------------------\*/

case SUNRISE\_INIT:

i = 255;

timer = millis();

ledState = SUNRISE\_SOON;

break;

case SUNRISE\_SOON:

if((millis() - timer) >= ((((sunsetDelay\*60000)/8)\*2)/255)){

timer = millis();

analogWrite(R, 220+(i\*7/51)); //255-220

analogWrite(G, 50+(i\*41/51)); //255-50

analogWrite(B, i); //255-0

analogWrite(W, i); //0-255

if(i > 0){

i--;

}

else{

ledState = SUNRISE\_MIDDLE;

i = 220;

analogWrite(G, 50); //50

analogWrite(B, 0); //0

}

}

break;

case SUNRISE\_MIDDLE: //i musi byt 255

if((millis() - timer) >= ((((sunsetDelay\*60000)/5)\*2)/35)){

timer = millis();

analogWrite(R, i); //220-255

if(i < 255){

i++;

}

else{

ledState = SUNRISE\_LATE;

i = 255;

analogWrite(B, 0); //0

}

}

break;

case SUNRISE\_LATE:

if((millis() - timer) >= ((((sunsetDelay\*60000)/8)\*1)/255)){

timer = millis();

analogWrite(R, i); //255-0

analogWrite(G, (i\*10/51)); //50-0

if(i > 0){

i--;

}

else{

ledState = SUNRISE\_IDLE;

i = 0;

}

}

break;

/\*------------------MOONSET\_INITIALIZING------------------\*/

case MOONSET\_INIT:

i = 0;

timer = millis();

ledState = MOONSET;

analogWrite(R, 0); //0

break;

case MOONSET:

if((millis() - timer) >= ((moonsetDelay\*60000)/60)){

timer = millis();

analogWrite(G, i/2); //0-30

analogWrite(B, i); //0-60

if(i < 60){

i++;

}

else{

ledState = MOONSET\_IDLE;

i = 0;

}

}

break;

/\*------------------MOONRISE\_INITIALIZING------------------\*/

case MOONRISE\_INIT:

i = 60;

timer = millis();

ledState = MOONRISE;

analogWrite(R, 0); //0

break;

case MOONRISE:

if((millis() - timer) >= ((moonriseDelay\*60000)/60)){

timer = millis();

analogWrite(G, i/2); //30-0

analogWrite(B, i); //60-0

if(i > 0){

i--;

}

else{

ledState = MOONRISE\_IDLE;

i = 0;

}

}

break;

default:

i = 0;

break;

}

}

**PŘÍLOHA Č. 2**