VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BEZDRÁTOVÉ OVLÁDÁNÍ LED SVĚTEL

WIRELESS CONTROL OF LED LIGHTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. VÁCLAV MARTINKA**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Dr. Ing. PAVEL ZEMČÍK**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání diplomové práce

Student: **Martinka Václav, Bc.**
Program: Informační technologie Obor: Počítačové a vestavěné systémy
Název: **Bezdrátové ovládání LED světel**
Wireless Control of LED Lights
Kategorie: Uživatelská rozhraní
Zadání:

1. Prostudujte způsob řízení světel počítačem se zaměřením na embedded systémy, bezdrátovou komunikaci a LED světelné zdroje.
2. Navrhněte zdroj světla s LED řízený embedded systémem a způsob jeho bezdrátového ovládání prostřednictvím WiFi.
3. Popište a diskutujte možnosti navrženého řešení a popište předpokládané vlastnosti a možné varianty ovládání.
4. Navržený systém implementujte a demonstrujte jeho funkčnost.
5. Diskutujte dosažené výsledky a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 19. května 2021
Datum schválení: 30. října 2020

Abstrakt

Tato práce práce řeší návrh a implementaci Wi-Fi LED osvětlení. To je tvořeno různými moduly – vypínače, tlačítka, jedno i vícebarevné světelné zdroje – a centrální jednotkou, která obstarává vzájemnou komunikaci a automatizaci. Výsledkem práce je návrh prototypu a několika modulů, které jsou postaveny na bázi mikrokontroléru ESP32. Jejich obslužný program je implementován v jazyce C++. Interakce s uživatelem probíhá skrze webovou stránku. Hlavním výsledkem práce je demonstrace možnosti použití Wi-Fi k ovládání světelných zdrojů a vytvoření komplexního osvětlení za pomoci kombinace jednoduchých modulů a pravidel.

Abstract

This work deals with the design and implementation of Wi-Fi LED lighting. It consists of various modules - switches, buttons, single or multicolored light sources - and a central unit that provides mutual communication and automatizacion. The result of the work is the design of a prototype and several modules, which are built on the basis of the microcontroller ESP32. Their utility is implemented in C++. The user interacts through the website. The main result of this thesis is a demonstration of the possibility of using Wi-Fi to control light sources and creating complex lighting using a combination of simple modules and rules.

Klíčová slova

LED osvětlení, bezdrátové ovládání, řízení mikropočítačem, Wi-Fi, modul ESP32

Keywords

LED lighting, wireless control, microcomputer control, Wi-Fi, module ESP32

Citace

MARTINKA, Václav. *Bezdrátové ovládání LED světel*. Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík

Bezdrátové ovládání LED světel

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci na téma *Bezdrátové ovládání LED světel* vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Václav Martinka
14. května 2021

Poděkování

Děkuji mému vedoucímu prof. Dr. Ing. Pavlu Zemčíkovi za odpornou pomoc při vypracování diplomové práce.

Obsah

Kapitola 1

Úvod

Současný rozvoj elektroniky, především její neustálé zmenšování i zlevňování vede k tomu, že dnes není problém do patice žárovky integrovat celý miniaturní počítač, pomocí něhož můžeme takovou žárovku dálkově ovládat. To přináší zvýšení komfortu, možnost vzdálené kontroly ale i nižší spotřebu díky řízení jasu. A právě bezdrátové ovládání osvětlení je cílem této práce.

Současným uživatelům už nestačí mít dálkový ovladač, kterým změní jas žárovky v místnosti, kde se nachází. Oni by rádi světla ovládali tím, co mají u sebe, což je dnes nejčastěji mobilní telefon. A ještě lépe, pokud by světla sami poznali, že mají změnit jas, nebo se rozsvítit... Není to tak dlouho, co se toto mohlo zdát jako nereálné, či minimálně uživatelsky nepřívětivé nebo zbytečné. Dnes už není takový problém toho dosáhnout pomocí nepřeberného množství hotových chytrých žárovek a jejich výběr i schopnosti se budou dále pouze zvětšovat.

Protože mě oblast vestavěných systémů a chytrého osvětlení láká a rád bych do ní více pronikl, rozhodl jsem se navrhnout vlastní řešení bezdrátového osvětlení. To by mělo podporovat jak světelné zdroje, tak právě i bezdrátové vypínače. Všechny tyto moduly bude dohromady spojovat centrální jednotka, která bude řídit jejich integraci a sama bude zajišťovat i automatizaci. Uživatel s ní, ale případně i s jednotlivými moduly, bude komunikovat pomocí Wi-Fi skrze webový prohlížeč na svém zařízení, přičemž by mělo být jedno, jestli se jedná o počítač nebo smartphone.

Práce je dělena do čtyř kapitol. Začíná tímto úvodem, za nímž následuje kapitola, která má za úkol uvést čtenáře do současných technologií potřebných k vytvoření takového zařízení. Kromě samotných světelných zdrojů jsou zde uvedeny i základní elektronické součástky a jejich typická zapojení, doplněny o popis mikrokontroléru. V druhé části této kapitoly se věnuji počítačovým sítím a jejich protokolům, následovaných přehledem několika současných řešení bezdrátového osvětlení. Třetí kapitola hodnotí současná řešení, tedy jejich výhody či nevýhody a na její závěr definuje klíčové vlastnosti, kterých by mělo být v rámci této práce dosaženo. Následuje čtvrtá závěrečná kapitola, která přináší popis návrhu a vývoje funkčního vzorku.

Kapitola 2

Současné technologie bezdrátově ovládaného osvětlení

V úvodu této kapitole s aktuálními technologiemi na poli řízení světel, bezdrátové komunikace a LED osvětlení. Úvodem jsou představeny některé současné řešení bezdrátového osvětlení. Následně jsou popsány mikrokontroléry – klíčové prvky pro stavbu řiditelného osvětlení. Ty pro svůj běh potřebují další diskrétní součástky, proto je dále popsáno několik nejdůležitějších včetně jejich typického zapojení. Závěrem této kapitoly je popsána bezdrátová komunikace, tedy jí používané technologie a protokoly. Informace obsažené v této kapitole se týkají pouze technologií, které mají bezprostřední vztah k práci a jsou důležité k pochopení jejího obsahu. Více toho rozsah diplomové práce nedovoluje.

2.1 Existující řešení bezdrátově ovládaného osvětlení

V současné době na trhu existuje nepřeberné množství bezdrátově ovládaných světelných zdrojů od různých výrobců, nabízejících různé funkce i vzájemnou kompatibilitu či naopak nekompatibilitu [18]. V rámci této podkapitoly je popsáno několik konkrétních výrobků, jejich vlastnosti, způsob komunikace a přístup k ovládání.

2.1.1 Philips Hue LightStrip Plus

Jedná se dvou metrový RGBW[1] LED pásek včetně řídící jednotky a napájecího zdroje. Pomocí dodatečných modulů jej lze prodloužit až na deset metrů. S okolím komunikuje skrze Bluetooth. Uživatel má na výběr buď ovládání přímo z mobilního telefonu bez možnosti větší integrace s ostatními prvky nebo pořízení centrální jednotky nazývané *Philips Hue Bridge*, což mu umožní pokročilé možnosti ovládání[2].

Pro přímou komunikaci pomocí Bluetooth je nutné použít mobilní aplikaci *Philips Hue Bluetooth*. Ta má na Google Play (4,5 hvězdičky) i App Store (4,7 hvězdičky) poměrně vysoké hodnocení. Z popisu vyplývá několik omezení daných technologií. Zejména kratší dosah, výrobce konkrétně uvádí 30 stop, tedy devět metrů. To může být v rámci rodinného domu v kombinaci s několika zdmi málo. Druhým omezením je pak maximální počet

[1] Red (červená), Green (zelená), Blue (modrá), White (bílá) – tedy pásek je schopen svítit libovolnou barvou, ale na rozdíl od běžných RGB pásků obsahuje i bílé diody, díky čemuž je bílé světlo přirozenější.

[2] Převzato z https://www.philips-hue.com/cs-cz/p/hue-white-and-color-ambiance-lightstrip-plus-v4-%E2%80%93-2-metry/8718699703424.

zařízení, který je limitován na 10. Aplikace umožňuje změnu barvy a jasu a to i pro více svítidel zároveň, popř. využití jedné z přednastavených scén[2].

Obrázek 2.1: Philips Hue LightStrip Plus v4 - 2 metry[2].

Zajímavou možností je pořízení *Hue Bridge*. S tím komunikuje nejspíše pomocí ZigBee, čímž je odstraněno omezení krátkého dosahu, byť popis výrobce je na tyto informace poměrně skoupý. Hue Bridge přináší možnost integrace s domácími asistenty i službami třetích stran, jako jsou např. IFTTT či Samsung SmartThings [13]. Bridge se do sítě připojuje pomocí síťového kabelu, což může být pro někoho nevýhodou. Pro ovládání pak slouží aplikace *Philips Hue*, která je dostupná pro iOS i Android s dobrým hodnocením 4,6 hvězdičky. Osvětlení lze organizovat po jednotlivých pokojích a usnadnit tak ovládání. Do systému lze připojit celkem až 50 světel a 12 ovládacích prvků, to může být pro větší domácnost limitující[2].

2.1.2 Sonoff MINI

Nejedná se přímo o světelný zdroj, nýbrž o Wi-Fi vypínač. Je určen k montáži v kombinaci se současným vypínačem a svítidlem. To na jednu stranu přináší možnost snadno upravit současné osvětlení na bezdrátově ovládané, na druhou strnu toto řešení nabízí pouze omezené možnosti. Například tak nelze měnit jas či barvu. K ovládání slouží aplikace *eWelink*, která je společná pro více zařízení využívající tuto platformu[3]. I v tomto případě je možné ovládání pomocí domácích asistentů. Hodnocení aplikace je nižší, pro Android je to ještě přijatelných 4,1 hvězdy, ale pro iOS pouhých 2,7.

Pro pokročilé uživatele je k dispozici dokumentované jednoduché REST API, což usnadňuje případnou integraci do jiného systému[3]. Vypínače Sonoff vnitřně používají MCU z rodiny ESP [17]. Mírnou úpravou zařízení lze zpřístupnit sériový port a nahrát alternativní

[3]Převzato z https://sonoff.tech/product/wifi-diy-smart-switches/sonoff-mini.

Obrázek 2.2: Sonoff MINI Wi-Fi vypínač (*uprostřed*), připojuje se k běžnému vypínači (*vpravo*), přičemž se očekává montáž do běžné stavební krabice (*vlevo*)[3].

firmware. Buď lze zvolit nějaký existující, např. Tasmota nebo ESPHome či si napsat zcela vlastní. Tento přístup je vhodný pouze pro pokročilé uživatele, kterým vypínač tak může posloužit jako hotové hardwarové řešení.

2.1.3 IKEA TRÅDFRI

Chytré osvětlení IKEA souhrnně pojmenované *TRÅDFRI* poslouží jako demonstrace řešení využívající technologii ZigBee. IKEA nabízí zařízení pro kompletně bezdrátově ovládané osvětlení domácnosti. Tedy žárovky, kompletní svítidla i ovladače a čidla (aktuálně nabízí pouze pohybový senzor) [16]. Bezdrátové vypínače i čidla jsou napájeny bateriově a umožňují manuální párování se svítidly[4].

Obrázek 2.3: IKEA TRÅDFRI žárovka[5] a ovladač[4].

[4]Převzato z https://www.ikea.com/cz/cs/p/tradfri-dalkove-ovladani-30443124.

Sice díky tomu není nezbytně nutné pořizovat centrální jednotku (*bránu*), která by řídila komunikaci. Brána slouží jako most mezi technologií ZigBee a Wi-Fi, čímž přináší možnost ovládat osvětlení i z mobilního telefonu, popř. integraci s jinými zařízeními v síti. IKEA konkrétně umožňuje skrze aplikaci samozřejmě měnit barvu a jas, s pokročilých funkcí nabízí pouze časovače[5].

Tyto omezené možnosti by mělo jít řešit použitím jiné brány. Certifikace ZigBee by měla zajistit vzájemnou kompatibilitu, tudíž by mělo jít kombinovat bránu, žárovky i ovladače různých značek.

2.1.4 TP-Link Tapo L530E, MiPow Playbulb Smart

Tyto dvě žárovky zastupují v tomto přehledu jednodušší řešení. První zmíněná komunikuje pomocí Wi-Fi, druhá pomocí Bluetooth. Obě shodně potřebují pro své ovládání mobilní aplikaci a neumožňují pokročilejší integraci s ostatními prvky. Žárovka od TP-Linku umožňuje i hlasové ovládání skrze domácího asistenta[6]. Její aplikace `TP-Link Tapo` má dobré hodnocení (4,6 a 4,7 hvězdy v aplikačních obchodech) a sdružuje i další zařízení téže značky. Naproti tomu produkt firmy MiPow s aplikací `PLAYBULB X`[7] už tak propracovaný není. Jeho aplikace dosahuje velmi nízkého hodnocení - 2,2 hvězdy na Google Play a 2,8 hvězdy v App Store. Byť popis aplikace také slibuje správu více zařízení, tak je zde limit dosahu Bluetooth signálu. Mimo to uživatelé hlásí problém s párováním.

Obrázek 2.4: TP-LINK Tapo L530E (*vlevo*)[6] a MiPow Playbulb Smart (*vpravo*)[7].

2.1.5 ESPHome

Nejedná se o hotové řešení, jako v předchozích příklad, nýbrž o open source firmware pro mikrokontrolér ESP32 (a jeho předchůdce ESP8266). Součástí tohoto projektu je konfi-

[5]Převzato z https://www.ikea.com/cz/cs/p/tradfri-zarovka-led-e27-806-lumenu-bezdratove-stmivatelne-teple-bile-kulata-opalove-bila-90408797.

[6]Převzato z https://www.tp-link.com/cz/home-networking/smart-bulb/tapo-l530e.

[7]Převzato z https://www.mipow.com/collections/smart-light/products/mipow-smart-bulb.

gurátor, ve kterém si může uživatel vytvořit vlastní prvek chytré domácnost bez potřeby většího programování. Výsledné moduly komunikují s centrální jednotkou *Home Assistant*[8] pomocí MQTT protokolu[9]. Toto řešení cílí na pokročilé uživatele, kteří chtějí mít kontrolu nad všemi prvky své domácnosti a nevystačí si s funkcemi běžně dostupných řešení.

2.1.6 5m RGB LED sestava

Posledním vybraným řešením je komplet LED pásku, zdroje a dálkového ovladače. Ten typicky používá technologii infračerveného paprsku nebo 433MHz přenos. Tento ovladač je buď vázán na daný pásek, tudíž není možné jedním ovladačem řídit barvu a jas různých pásků nebo naopak může docházet k nechtěnému ovládání více pásků jedním ovladačem. Integraci s jinými prvky či ovládání z telefonu není možné.

Obrázek 2.5: Kompletní 5m RGB sestava s rádio frekvenčním ovladačem a zdrojem[10].

Na obrázku 2.5 je vyobrazen pětimetrový RGB led pásek. Jedná se o komplet se zdrojem i ovladačem. Ten ke komunikaci používá 433MHz přenos, tedy není vyžadována přímá viditelnost mezi ovladačem a přijímačem[10]

2.2 Popis mikrokontroléru a dalších součástek

Mikrokontrolér či mikropočítač (anglicky MicroController Unit) zkráceně MCU je pokročilý integrovaný obvod realizující funkci jednoduchého počítače. Oproti klasickému osobnímu počítači je zde kladen důraz na miniaturizaci, nízký odběr a snadnou integraci do elektronických obvodů [9]. Využívá se především jako součást vestavěných (*embedded*) systémů, se kterými komunikuje pomocí elektrických signálů. Oproti realizaci diskrétními součástkami přináší jednodušší návrh a rychlejší vývoj[11]. Není totiž nutné navrhovat zapojení,

[8] Open source software řídícího systému chytré domácnosti

[9] Převzato z https://esphome.io.

[10] Převzato z https://www.ledshopik.cz/5m-rgb-led-sestava-7-2w-m-30led-m-ip65-rf-ovladani-24-tlacitek-x1727.

[11] Převzato z přednášky: Růžička, R. *IMP: Vestavné systémy – Úvod.* 2017. FIT VUT v Brně.

místo toho stačí popsat chování zařízení pomocí programovacího jazyka a tento program následně nahrát do MCU. V případě, že je později zjištěna chyba nebo je potřeba změnit nějakou vlastnost, stačí jen nahrát opravený program.

Principiálně se stále jedná o počítač tak, jak si ho představí běžný uživatel. Proto obsahuje i stejné komponenty, byť integrované v jednom těle. Jádrem je procesorová jednotka, která vykonává program uložený v trvalé paměti. Mimo ni je tu dále přítomna i dočasná paměť pro běh programu. Klíčovou vlastností MCU jsou moduly, což jsou komponenty rozšiřují jeho schopnosti[11]. Jedná se především o vstupně/výstupní porty, analogové převodníky, časovače či různá komunikační rozhraní.

2.2.1 Centrální procesorová jednotka (CPU)

CPU se stará o samotné vykonávání programu uloženého v paměti. Program se skládá z posloupnosti instrukcí, které jsou vykonávány postupně jedna po druhé. Množina podporovaných instrukcí se nazývá *instrukční sada*. CPU se skládá z několika dílčích jednotek[11]:

Aritmeticko-logická jednotka neboli ALU, jak už z názvu vyplývá, provádí matematické a logické operace[11]. Tedy různé součty a násobení, ale také např. vyhodnocování podmínek, které slouží k větvení programu.

Registry představují velmi rychlou paměť přímo v jádru CPU. Jejich množství a velikost závisí na konkrétní architektuře a zaměření MCU. Protože ALU zpravidla neumí přímý přístup do programové nebo dočasné paměti, je nutné data nejdřív načíst do registru, poté je pomocí ALU zpracovat a následně opět uložit do paměti[11]. Registry mají v MCU ještě druhou důležitou funkci, slouží ke konfiguraci a komunikaci s moduly. Např. změna hodnoty na výstupním pinu se provádí změnou hodnoty registru [11]. Obdobně data přijatá z komunikačního rozhraní se zapisují také do patřičného registru. Tyto řídící registry mohou být vnitřně mapovány do určité části paměti.

Řadič řídí tok dat mezi registry, ALU a pamětmi [10]. Zároveň se také stará o vykonávání programu, tedy načítání jednotlivých instrukcí z paměti, jejich postupné dekódování a následné vykonání.

2.2.2 Paměť

V MCU se typicky nachází hned několik různých druhů paměti. Liší se především rychlostí nebo trvanlivostí dat a samozřejmě i cenou. Pro uložení programu se většinou používá paměť postavená na technologii FLASH, která je nevolatilní[12]. Lze se setkat i s pamětí typu ROM[13], ze které lze pouze číst a program je do ní nahrán už při výrobě [9]. Pro proměnné programu je zde přítomna paměť typu RAM[14], která je zpravidla volatilní, tedy po odpojení napětí se smaže [10]. Některá MCU obsahují ještě další paměť pro uložení uživatelských dat i po vypnutí, jiné ji buď simulují vyhrazeným blokem v programové paměti nebo ji neobsahují vůbec a je nutné použít externí obvod.

[12] Nevolatilní paměť je schopna udržet informaci i bez napájecího napětí.

[13] ROM je zkratkou pro Read Only Memory, česky paměť pouze pro čtení.

[14] RAM, neboli Random-Access Memory, je paměť s náhodným přístupem umožňují číst adresovaná data bez nutnosti sekvenčního přístupu.

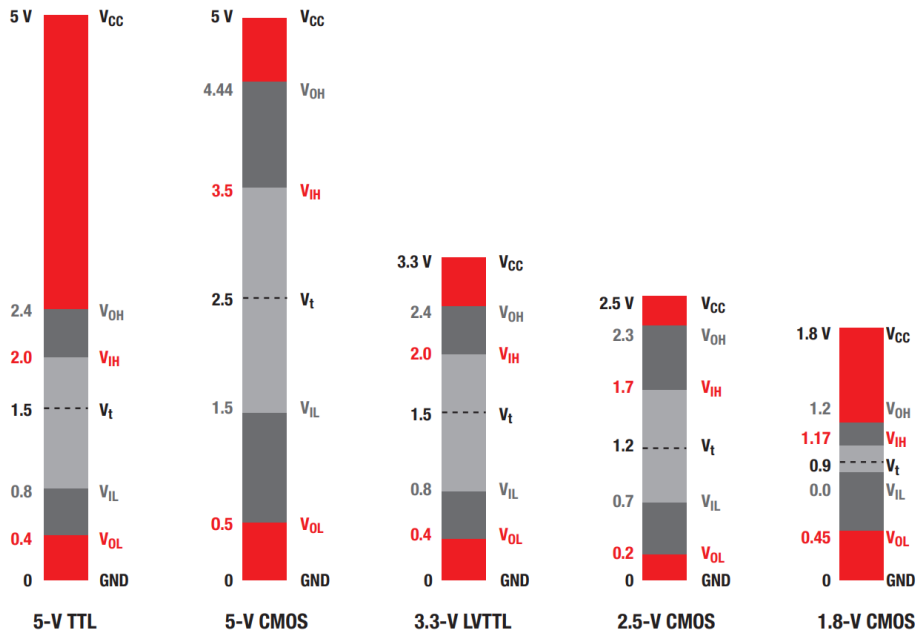
2.2.3 Rozšiřující moduly mikrokontroléru

Jak už bylo zmíněno výše, moduly jsou důležitou součástí MCU a poskytují mu další funkce. Zároveň umožňují realizovat různé jednoduché činnosti mimo CPU (např. počítání impulsů), díky čemuž je možné CPU buď uspat a snížit spotřebu nebo na CPU počítat něco jiného a zrychlit tak odezvu. Níže je uvedeno několik nejčastějších modulů.

Vstupně/výstupní porty anglicky též *General-purpose input/output* (zkráceně GPIO) umožňují MCU snadno komunikovat s okolím pomocí elektrických signálů. Porty jsou tvořeny piny, které jsou připojeny na jednotlivé elektrické vývody pouzdra. Tyto piny mohou být buď vstupní, výstupní nebo kombinované [11]. Každý port je doplněn o několik registrů, které určují směr pinu, zapisují hodnotu, reflektují přečtenou hodnotu, připojují *pull down* nebo *pull up* rezistor apod[15].

Signály jsou binární a nabývají dvou hodnot. Jejich napěťová hranice je dána technologií výroby a napájecím napětím. Logická jednička je představována napětím blízkým tomu napájecímu, naopak logická nula je připojena na tzv. referenční zem, neboli 0 V[15].

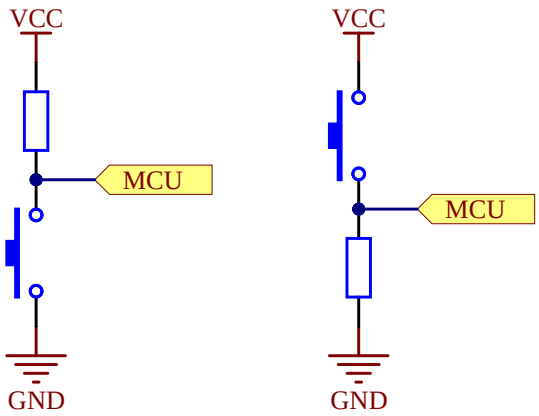
V závislosti na technologii je zde přibližně 30% tolerance, takže piny nejsou tolik náchylné na šum. Problémová je oblast okolo poloviny napájecího napětí (*zakázané pásmo*), kdy výsledná hodnota zpravidla není definována a může vést k chybám ve vykonávaní programu[15]. Obrázek 2.6 ilustruje bezpečné úrovně napětí pro logickou nulu i jedničku. Obdobným problémem jsou nepřipojené piny, ty slouží jako anténa a přijímají šum. Pokud je z nich čtena hodnota, není možné předem deterministicky určit, co bude přečteno[15].



Obrázek 2.6: Napěťové úrovně různých technologií. V_{CC} je napájecí napětí, GND zem. *Zakázané pásmo*, se nachází okolo napětí V_t (světle šedá oblast). V_{IH} a V_{IL} jsou hraniční bezpečná napětí pro logickou jedničku a nulu. Hodnoty V_{OH} a V_{OL} jsou pak minimálním (respektive maximálním) napětím, které se objeví na výstupu[17] pinu.

[15]Převzato z přednášky: Růžička, R. *IMP: Vestavné systémy – Porty MCU, jednoduchý vstup a výstup.* 2017. FIT VUT v Brně.

Toto nechtěné chování řeší tzv. *pull up* nebo *pull down* rezistor. V případě pull up připojen k napájecímu napětí a udržuje tak pin trvale v logické jedničce. Pull down rezistor je naopak připojen na zem a udržuje hodnotu na logické nule[15]. Hodnota tohoto rezistoru se pohybuje v řádech kilo ohmů, tudíž jím protékají velmi nízké proudy. Ne každý MCU je vybaven oběma typy rezistorů, popř. je neumožňuje připojit na libovolný pin. Proto je nutné zkontrolovat datasheet, popř. je do obvodu doplnit externě. V některých případech je potřeba, aby se pin nacházel ve stavu, jako by byl odpojený. Toho se docílí tak, že se nastaví jako vstupní, čímž se dostane do stavu tzv. *vysoké impedance*[15].



Obrázek 2.7: Zapojení pull up (*vlevo*) a pull down (*vpravo*) rezistorů.

A/D a D/A převodník neboli analogově digitální či digitálně analogový převodník je poměrně častým modulem moderních MCU. Tyto převodníky převádí digitální (diskrétní) signál na analogový (spojitý), popř. naopak[16]. Většina aplikací zpravidla vyžaduje digitalizaci analogových signálů, proto bývají MCU vybaveny hned několika A/D převodníky, zatímco D/A převodník v jednodušších MCU nenajdeme.

Převod vždy probíhá vůči tzv. *referenčnímu napětí*, které se typicky používá stejné jako napájecí, ale lze použít i jiné, které musí být přivedeno na patřičný pin. Samotný výsledek A/D převodu není přesná hodnota napětí, nýbrž ukazatel mapovaný do rozsahu 0 – V_{REF} [11]. To mimo jiné znamená, že bez použití externích součástek nelze měřit napětí větší, než je referenční. Toto mapování pak probíhá s různým rozlišením, kdy jeho snížení sice vede k nižší spotřebě, ale za cenu nižší přesnosti[16]. Převod opačným směrem, tedy vytvoření analogového signálu, pak probíhá velmi obdobně.

Časovač je velmi důležitým a rozšířeným modulem. Umožňuje procesoru přenechat jednoduché počítání pulsů na externí modul a věnovat se jiným výpočtům nebo svým uspáním snížit spotřebu [11]. MCU obsahuje zpravidla více různých časovačů využívající různé technologie, které ovlivňují spotřebu, přesnost i cenu[18].

Principem časovače je počítání pulsů. Ty může čítat buď od spuštění (tuto hodnotu lze využít k časování v rámci programu) nebo jen do určité hodnoty a poté vyvolat přerušení či nějakou akci [11]. Toho lze využít např. pro generování PWM signálu, ale stejně tak pro uspání programu na určitou dobu apod.

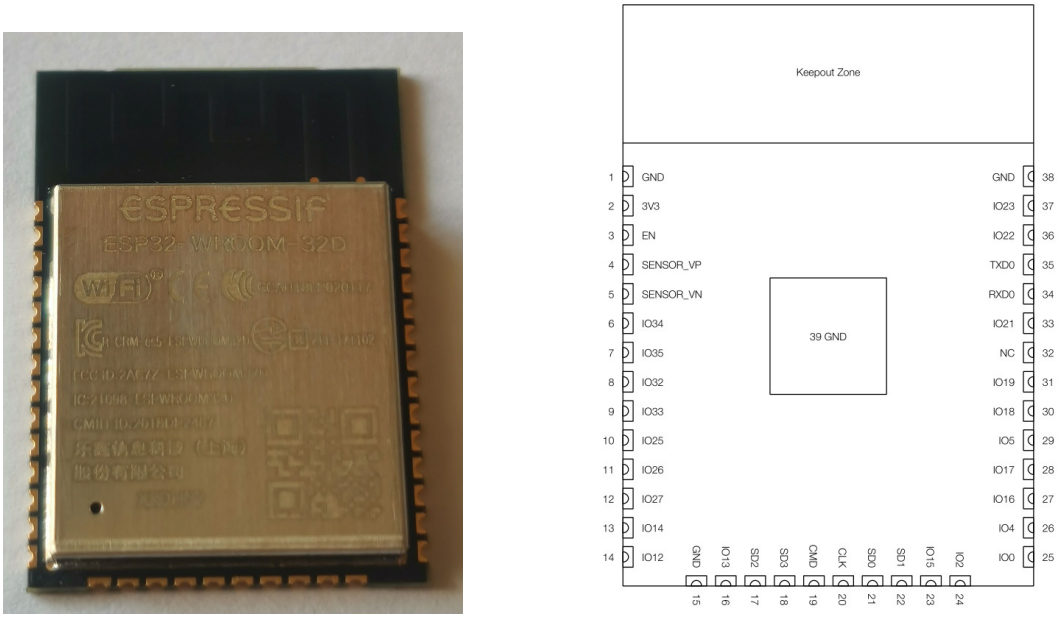
[16]Převzato z přednášky: Růžička, R. *IMP: Analogové vstupy a výstupy*. 2017. FIT VUT v Brně.
[17]Převzato z https://www.ti.com/lit/sg/sdyu001ab/sdyu001ab.pdf.
[18]Převzato z přednášky: Růžička, R. *IMP: Čítače a časovače*. 2017. FIT VUT v Brně.

Moderní MCU obsahují samozřejmě i další moduly. Zmínit lze např. *řadič přerušení*, pomocí kterého lze na základě externích vstupů pozastavit program a spustit předem určenou funkci. Dále moduly pro komunikaci pomocí různých sběrnic, jako např. UART, I^{2}C, SPI či CAN. Na závěr lze ještě zmínit watchdog, což je modul, který kontroluje, zda se MCU nezacyklil a případně jej restartuje[11].

2.2.4 Mikrokontrolér ESP32

Příkladem MCU může být rodina mikrokontrolérů ESP32 od firmy Espressif. Nižší verze těchto MCU používají pouze jedno jádro běžící na 160 MHz. Střední a vyšší verze mají dvě jádra s frekvencí 240 MHz. Programátor má k dispozici vždy minimálně 4 MB paměti pro program a jeho data[19]. Dočasná paměť (RAM) má kapacitu 520 KB[20].

Na čipu je integrováno velké množství základních i pokročilých modulů. Nachází se zde 36 GPIO pinů, přičemž na většinu z nich lze připojit pull up a pull down rezistory. Ty doplňuje několik A/D i D/A převodníků, podpora pro dotykové senzory, řízení LED pomocí PWM a velké množství různých komunikačních rozhraní (SPI, UART, I^{2}C, ...). Zřejmě nejzajímavějšími moduly jsou Wi-Fi 802.11 b/g/n a Bluetooth v4.2 (s podporou BLE) [1]. Společně s podporou pro nízkoodběrový běh jsou tyto MCU vhodné pro stavbu bezdrátových zařízení a IoT.



Obrázek 2.8: Mikrokontrolér `ESP32-WROOM-32D` a zapojení jeho pinů, tzv. *pinout*[20].

2.2.5 Elektronické součástky a jejich možná zapojení

Integrace mikrokontroléru do obvodu vyžaduje použití dalších součástek. V následujících odstavcích jsou uvedeny nejzákladnější součástky včetně jejich možnosti využití a typického zapojení.

[19] Čerpáno z https://www.espressif.com/en/products/modules/esp32.

[20] Čerpáno z https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf.

Rezistor, též chybně označovaný jako *odpor*, je základní elektronickou součástkou. Jeho funkcí v obvodu je kladení elektrického odporu. Toho se využívá k omezení protékajícího proudu vytvořením úbytku napětí [4]. Vztah odporu, napětí a proudu popisuje *Ohmův zákon*

$$U = R \cdot I$$

kde U je úbytek napětí, I protékající proud a R odpor rezistoru.

Obrázek 2.9: Schématická značka rezistoru (*vlevo*) a ukázka několika rezistorů v různých hodnotách a provedeních[21].

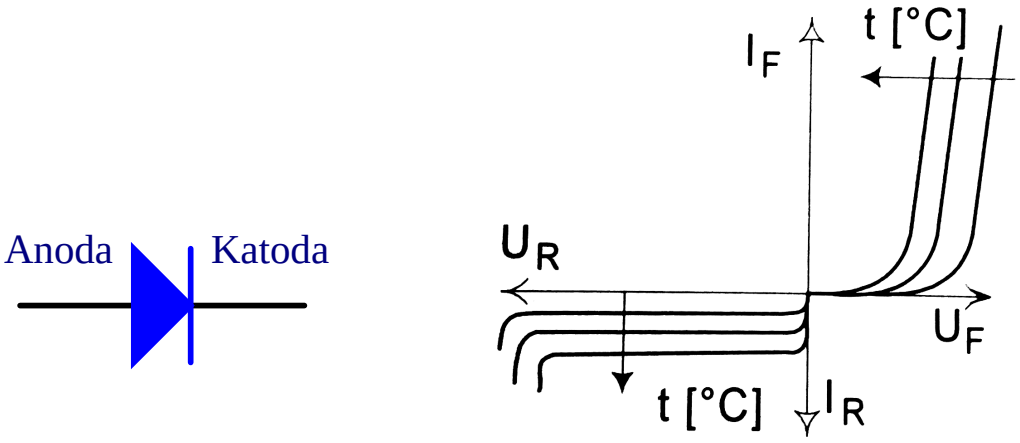
Dioda je jednoduchou, přesto důležitou elektronickou součástkou. Jedná se o nejjednodušší polovodičovou součástku s jedním PN přechodem [5]. Mezi ty složitější pak patří tranzistor, který se zasloužil o rozvoj elektroniky.

Dioda má dva vývody, které se nazývají *anoda* a *katoda*. Je možné ji zapojit v *propustném* nebo *závěrném* směru, podle toho, na který vývod je přivedeno kladnější napětí [5]. Rozdílnost těchto zapojení je dobře patrná na volt-ampérové charakteristice diody, která je znázorněna na obrázku 2.10. Dále je zde patrná i závislost na teplotě, především to, že s rostoucí teplotou roste i ochota diody (a polovodičů obecně) vést elektrický proud. Proto je nutné polovodiče při vyšších výkonech chladit, jinak by mohlo dojít k jejich zničení nadměrným proudem.

Propustný směr Je-li na anodu přivedeno kladnější napětí než na katodu, tedy platí $U_A > U_K$, pak je dioda zapojena v propustném směru. V rozsahu $0 - U_F$, kde U_F je tzv. *prahové napětí* (pro běžnou křemíkovou diodu je jeho hodnota asi 0,7 V) dioda nepropouští žádný proud, chová se tedy jako izolant. Po jeho překonání se naopak z diody stává velmi dobrý vodič a začíná vést proud. Pokud nedojde k jeho omezení, např. pomocí rezistoru, může dojít k tepelné destrukci součástky. Maximální proud se odvíjí od konkrétního modelu [5]. Běžně se jedná o jednotky ampér, u výkonových diod to mohou být i stovky.

Závěrný směr V případě, že platí $U_A < U_K$, jedná se o zapojení v závěrném směru. Dioda se chová téměř jako izolant (propouští jednotky mikroampér) bez ohledu na rostoucí napětí. Je-li překročen výrobní limit (maximální závěrné napětí) dojde k proražení PN přechodu a z diody se stává opět vodič. Toto proražení bývá zpravidla nevratné a součástka je zničena [5].

[21] Čerpáno z https://www.slideserve.com/turner/component-identification.
[22] Čerpáno z literatury [5].



Obrázek 2.10: Schématická značka diody s vyznačenou anodou a katodou a její voltampérová charakteristika včetně závislosti na teplotě[22].

LED je zkratkou pro *Ligt-Emitting Diode* [9], česky *světlo emitující dioda*, ale často se lze setkat s nevhodným označením LED dioda – slovo dioda se v tomto případě totiž vyskytuje hned dvakrát. LED je velmi podobná klasické diodě. Světlo zde vzniká při průchodu proudu PN přechodem. To znamená, že je nutné ji zapojit v propustném směru. Samozřejmě i LED mají povolené maximální proudové zatížení, které je nutné dodržet, jinak dojde k nevratnému poškození PN přechodu. Proto je potřeba vždy použít ochranný rezistor. Další společnou vlastností je prahové napětí, tedy napětí, od kterého se LED otevře a začne propouštět proud a zároveň tedy vyzařovat světlo. Hodnota tohoto prahového napětí představuje úbytek napětí na diodě [6]. Liší se podle barvy, výkonu i výrobce, přičemž se pohybuje v rozmezí přibližně 1,5 V – 4 V [9].

Obrázek 2.11: Schématická značka LED a její různá provedení.

Výpočet ochranného rezistoru vyžaduje znalost dvou veličin – napájecí napětí a úbytek napětí na diodě. V případě, že v zapojení figuruje více diod, existuje několik kom-

[23]Čerpáno z Bezstarosti, J. *Tranzistor polopatě* [RoboDoupě]. Prosinec 2011. Dostupné na: http://robodoupe.cz/wp-content/uploads/2012/01/tranzistor_polopate.pdf.

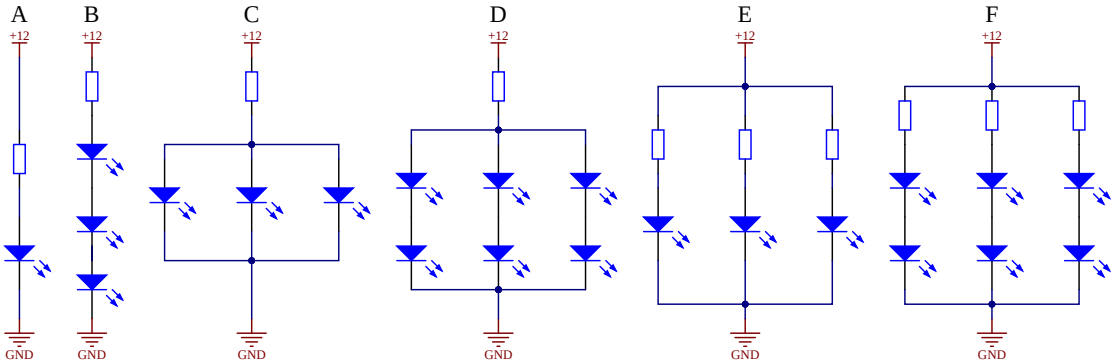
binací zapojení, názorně to ukazuje obrázek 2.12. Ve všech případech postačuje Ohmův zákon[23], pouze je nutná jeho případná modifikace:

$$R = \frac{U_R}{I} = \frac{U_{IN} - U_{LED}}{I}$$

Tedy je nutné znát požadovaný úbytek napětí na rezistoru (odpovídá rozdílu napájecího napětí a úbytku napětí na diodě) a proud protékající obvod, respektive požadovaný proud, který bude protékat diodou[23]. Tento vzorec je vhodný pro zapojení A, C a E z obrázku 2.12. Přičemž v případě varianty C je nutné, aby jednotlivé LED měly stejný úbytek napětí. Pokud jsou LED zapojeny sériově, je pouze nutné odečíst úbytky napětí všech diod v sérii.

$$R = \frac{U_{IN} - U_{LED_1} - U_{LED_2} - \ldots - U_{LED_n}}{I}$$

Z obrázku 2.12 by se tento způsob využil u zapojení B, D a F. Obdobně i zde pro zapojení D platí, že na jednotlivých větvích musí docházet ke stejnému úbytku napětí.



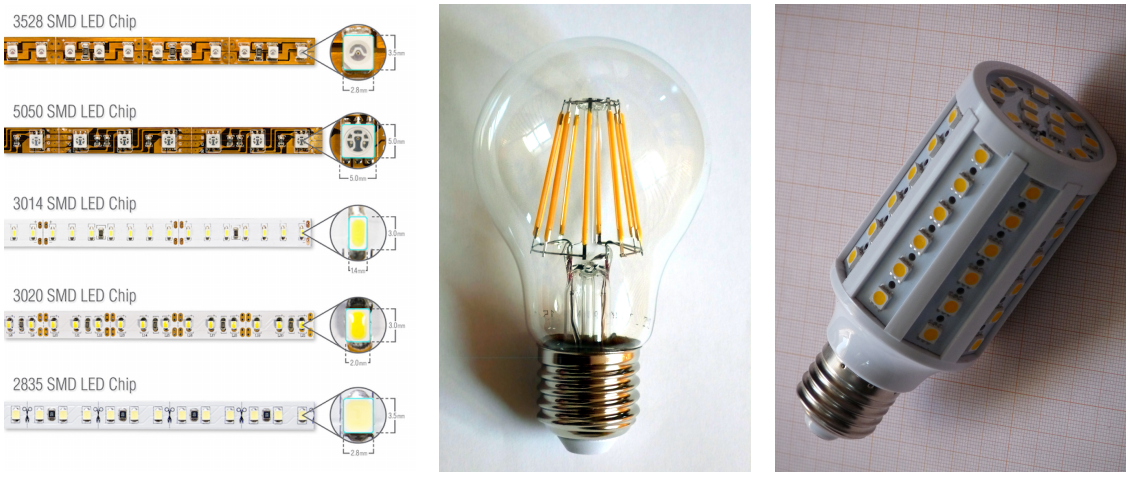
Obrázek 2.12: Příklad různých zapojení LED a ochranných rezistorů.

Výsledná hodnota s největší pravděpodobností nebude vyráběna, proto je nutné zvolit nejbližší podobný (vždy se volí hodnota vyšší). Dále je nutné ověřit výkonovou ztrátu na rezistoru P ($P = U_R \cdot I_{LED}$), protože přebytečná elektrická energie je na něm přeměněna na teplo a mohlo by dojít k jeho zničení. Při vyšších výkonech je proto nutné zvolit i patřičně výkonný rezistor[23].

V kombinaci s bezdrátovým ovládáním se v současné době jedná o asi nejdynamičtěji rozvíjející se druh osvětlení [6]. Díky masivnímu zvýšení výkonu za posledních několik let lze totiž LED použít nejenom jako indikační či signalizační kontrolky, ale taktéž jako náhradu žárovek či zářivek.

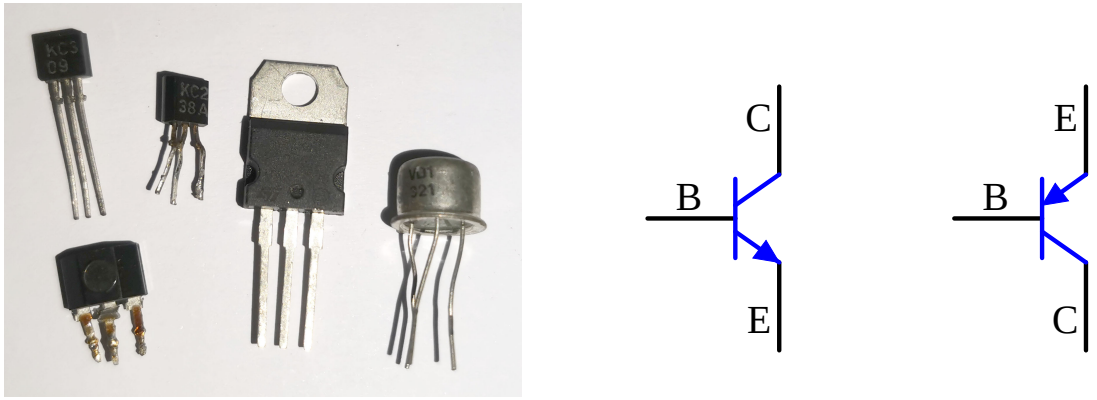
Samotný LED světelný zdroj se skládá ze dvou částí – elektrického zdroje a LED modulů. Hotová řešení se prodávají typicky ve tvaru připomínající klasickou žárovku, a to včetně stejné patice. Díky tomu je výměna uživatelsky velmi snadná. Mezi další častá provedení lze zařadit tzv. LED pásky. Jedná se samolepící pásku, která má na jedné straně lepidlo a na druhé straně slabou měděnou vrstvu, na kterou jsou připájeny LED v sérioparalelním zapojení včetně patřičných rezistorů [6] (varianta F z obrázku 2.12). Tyto pásky se vyrábí v různých barevných i výkonných provedeních. Je nutné je doplnit o napájecí obvod a případně i řídící elektroniku. Na závěr je vhodné ještě zmínit různé hotové LED reflektory,

které ve svém těle integrují vše potřebné a do domácí elektrické sítě se připojují typicky pomocí svorkovnice [6]. Na obrázku 2.13 jsou ukázány některé z možností LED osvětlení.



Obrázek 2.13: Různá provedení LED osvětlení. Zcela vlevo je příklad různých LED pásků[24], uprostřed[25] a vpravo[26] pak ukázka dvou tzv. LED žárovek.

Tranzistor Poslední, ale zřejmě nejdůležitější zde popsanou součástkou je tranzistor. Ten nahradil méně spolehlivé, rozměrné a žravé elektronky, čímž se zasloužil o rozvoj a miniaturizaci elektroniky. Nejjednodušším typem jsou *bipolární* tranzistory, které jsou řízeny proudem. Druhou skupinou jsou *unipolární*, ty jsou ovládány elektrickým polem (napětím), což umožňuje ještě více snížit spotřebu. Mimo to existují i tranzistory *kombinované* [5]. Dále budou uvažovány pouze bipolární tranzistory. Princip těch ostatních je až na drobné odchylky podobný.



Obrázek 2.14: Několik tranzistorů a jeho schématická značka – NPN (*levá*) a PNP (*pravá*). Pojmenované vývody jsou *báze* (B), *emitor* (E) a *kolektor* (C).

Tranzistor je tvořen dvěma PN přechody a podle uspořádání je pak dělíme na NPN a PNP. Jejich společnou vlastností je schopnost zesilovat proud, tedy malá změna proudu na vstupu vyvolá velkou změnu proudu na výstupu. Obrázek 2.14 mimo jiné znázorňuje

[24] Převzato z https://www.flexfireleds.com/comparison-between-3528-leds-and-5050-leds.
[25] Převzato z https://commons.wikimedia.org/wiki/File:LEDfilamentLightBulbE27.jpg.
[26] Převzato z https://en.wikipedia.org/wiki/File:LED_E27_corn.JPG.

schématickou značku pro obě varianty. Jak je vidět, tranzistor má tři vývody: *báze* (B), *emitor* (E) a *kolektor* (C). Vstupní proud na bázi, neboli *bázový proud* I_B řídí proud mezi emitor a kolektorem, tedy *kolektorový proud* I_C [9]. Varianta tranzistoru (NPN/PNP) pak definuje, jakým směrem proud teče.

Tranzistor jako spínač (či zesilovač) je jedním ze základních zapojení tranzistoru. Využití nalezne např. při spínaní zátěže pomocí mikrokontroléru, jehož maximální zatížení jednoho pinu je poměrně nízké (jednotky až desítky mA), zatímco je potřeba spínat stovky mA či jednotky A. Druhým využitím je spínaní zátěže s rozdílným napájecím napětím.

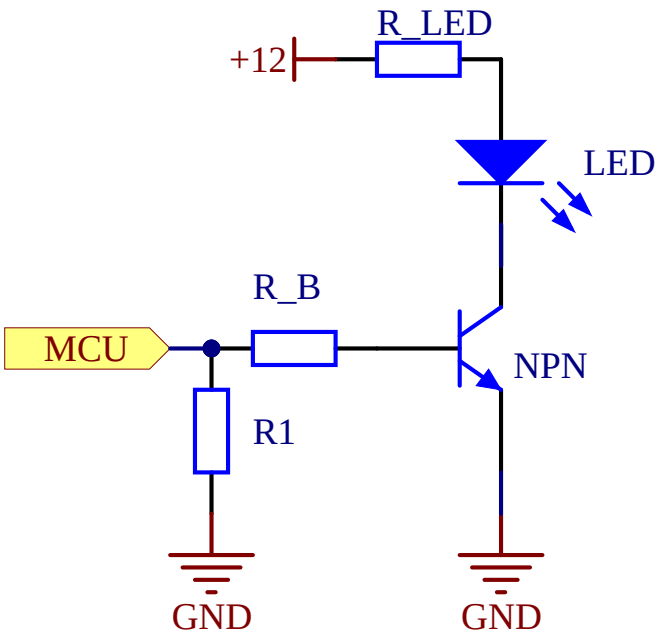
Schéma zapojení pro spínaní zátěže (zde představované LED) pomocí MCU je na obrázku 2.15. Výstupní pin mikrokontroléru zde spíná LED napájenou 12 V. Význam a hodnoty jednotlivých rezistorů jsou popsány níže[23]. Výběr vhodného tranzistoru se odvíjí od spínaného proudu a napětí. Tyto informace jsou uvedeny v datasheetu. Klíčové jsou následující údaje:

Kolektorový proud I_C je maximální proud, který může téct přes kolektor, prakticky se jedná o maximální proud, který lze tranzistorem spínat[23].

Proudový zesilovací činitel h_FE definuje, kolikrát je proud kolektorem vyšší než proud bází [9].

Saturační napětí mezi kolektorem a emitorem V_{CEsat} udává úbytek napětí mezi těmito vývody[23].

Saturační napětí mezi bází a emitorem V_{BEsat} obdobně jako V_{CEsat}[23].



Obrázek 2.15: Zapojení NPN tranzistoru jako spínače.

Pomocí těchto údajů je lze vypočítat hodnoty jednotlivých rezistorů:

$\mathbf{R_{LED}}$ představuje ochranný rezistor LED. Pro tranzistor jako takový není důležitý. Jako výpočet se byl již uveden v podkapitole 2.2.5, pouze s tím rozdílem, že kromě úbytku napětí na LED je nutné od napájecího napětí odečíst i V_{CEsat}[23]. Tedy úbytek napětí na

R_{LED} je

$$U_{R_{LED}} = U \ - U_{LED} - V_{CEsat}$$
$$R_{LED} = \frac{U \ - U_{LED} - V_{CEsat}}{I_{LED}}$$

R_1 lze teoreticky vynechat. Jeho význam je důležitý v kombinaci s MCU. V závislosti na konkrétním modelu nemusí být daný pin při inicializaci nastaven na nízkou logickou hodnotu a mohlo by tak dojít k nechtěnému sepnutí tranzistoru a tím pádem i ke krátkému bliknutí diodou. R_1 zde představuje pull-down rezistor (viz 2.2.3)[23]. Jeho hodnota není kritická – lze použít třeba 10 kΩ – nebo může být použit stejný, jako jsou pull-down rezistory daného MCU.

R_B je naopak velmi důležitý. Slouží jako ochrana tranzistoru před zničením nadměrným proudem. Základem výpočtu jeho hodnoty je bázový proud I_B, který je h_{FE} krát menší než kolektorový proud (tedy proud protékající zátěží, v tomto případě diodou)[23]. Pokud je tranzistor použit jako spínač, použije se hodnota alespoň třikrát větší, než jaká by byla potřeba pro plné otevření.

$$I_B = \frac{I_C}{h_{FE}} \cdot 3$$

Dále je nutné určit potřebný úbytek napětí U_{R_B} na rezistoru R_B. K tomu potřebujeme znát napětí na výstupu MCU U_{IN} (nebo obecně hodnotu napětí, kterým bude tranzistor spínán) a saturační napětí V_{BE}[23]. Ta se spočte jako

$$U_{R_B} = U_{IN} - V_{BE}$$

samotná hodnota se pak určí pomocí Ohmova zákona

$$R_B = \frac{U_{R_B}}{I_B} = \frac{U_{IN} - V_{BE}}{\frac{I_C}{h_{FE}} \cdot 3} = \frac{(U_{IN} - V_{BE}) \cdot h_{FE}}{3 \cdot I_C}$$

Na závěr je nutné ověřit výkonovou ztrátu na tranzistoru. Jedná se o součet výkonů tekoucích přes bázi a kolektor[23]. Pokud by byla tato ztráta vyšší, než povoluje výrobce, bude nutné zvolit jiný tranzistor.

$$P = V_{CE} \cdot I_C + V_{BE} \cdot I_B$$

2.3 Technologie řízení jasu

Existuje několik různých přístupů ke řízení jasu svítidla. Při výběru je nutné zohlednit nejen fyzikální princip vznikání světla, ale i samotný výkon svítidla či požadavky na celkovou přesnost.

2.3.1 Napájení stejnosměrným proudem

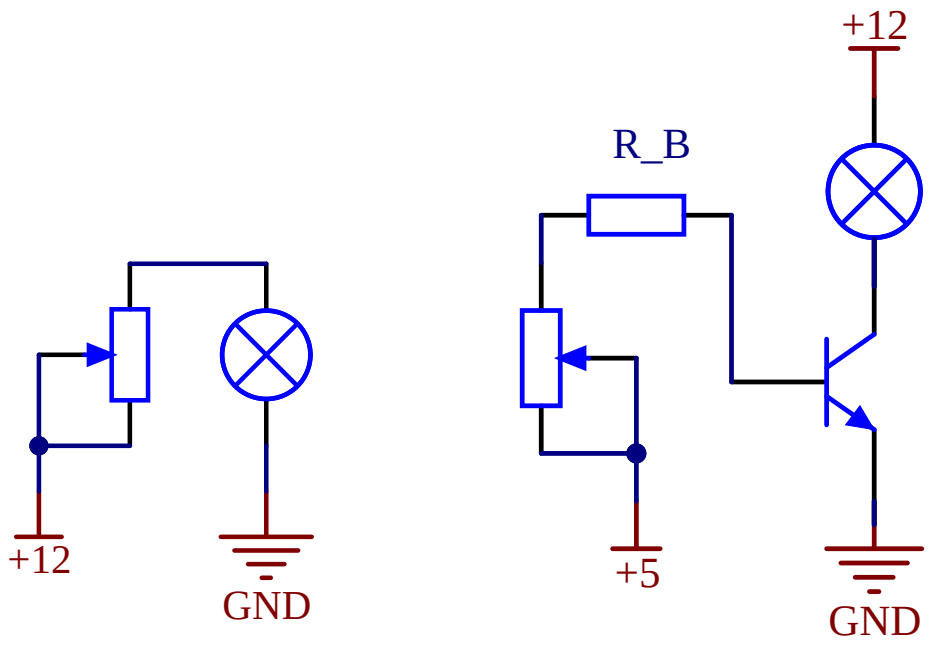
Následující postup vychází z *Ohmova zákona* $U = R \cdot I$, respektive ze vzorce pro výpočet výkonu $P = U \cdot I$. Jejich kombinací lze dojít k závislosti výkonu na napětí, popř. proudu:

$$P = I^2 \cdot R = \frac{U^2}{R}$$

Bude-li pro napájení použit konstantní zdroje, je nutné použít proměnný rezistor, např. potenciometr. Nejjednodušší zapojení s použitím potenciometru je na schématu 2.16(a). Veškerý proud procházející žárovkou prochází zároveň i přes potenciometr. Toto řešení by bylo použitelné pouze pro žárovky s výkonem v jednotkách wattů, jinak by došlo k poškození potenciometru.

Rozšířené zapojení na obrázku 2.16(b) je doplněno o NPN tranzistor, který zde slouží jako zesilovač. Malý proud procházející přes potenciometr do báze tranzistoru zde slouží k řízení proudu tekoucím žárovku. Každá jeho změna je několika set násobně zesílena, proto lze takto řídit i výkonné žárovky. Limitem už není maximální zatížení potenciometru, ale maximální proudové zatížení tranzistoru. Potenciometr v tomto příkladě může být nahrazen D/A převodníkem mikrokontroléru.

Bohužel tento přístup nelze pro řízení LED aplikovat. Jsou zde dvě hlavní překážky. Prvním problémem je prahové napětí. To se liší nejen v závislosti na barvě a výrobci, ale i mezi jednotlivými kusy v dané sérii (zejména u levnějších modulů). Proto by bylo nutné hodnotu rezistoru R_B kalibrovat samostatně pro každou vyrobenou diodu, což by bylo velmi nepraktické. Druhou překážkou je pak samotná VA charakteristika diody, kdy plného jasu dosahuje velmi rychle po překročení prahového napětí. To by kladlo vysoké nároky na přesnost komponent. Takové zapojení by pak bylo velmi háklivé na okolí vlivy a stejně tak i nákladné. K řízení LED by bylo nutné buď použít proudový zdroj nebo aplikovat zcela jiný přístup, a to řízení jasu pomocí PWM.



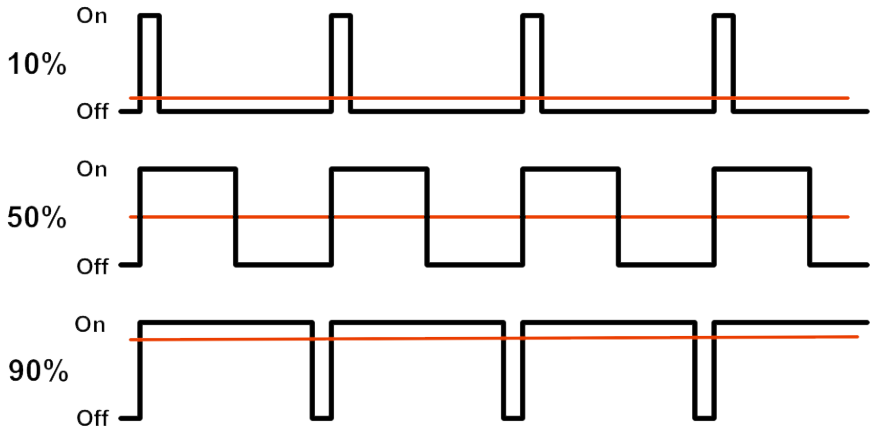
(a) Zakladní zapojení s potenciometrem

(b) Rozšířené zapojení s využitím NPN tranzistoru

Obrázek 2.16: Nejjednodušší možné zapojení proměnného rezistoru pro řízení jasu osvětlení.

2.3.2 Technologie PWM

PWM, neboli pulzně šířková modulace, je zcela opačný přístup. Napájecí napětí i proud nejsou nijak omezovány, ale místo toho je omezena doba, po kterou je tato energie distribuována do spotřebiče, v tomto případě do LED modulu [3]. Napájení je tedy poskytováno pulsně a šířka těchto pulsů, přesněji řečeno poměr mezi stavem zapnuto a vypnuto (odborně nazýván *střída*[18]) ovlivňuje celkový výkon. Využívá se zde setrvačnost lidského oka, pokud bude frekvence pulsů dostatečně vysoká, člověk nebude schopen vnímat blikání. Místo toho nastane iluze, že LED změnila svůj jas.



Obrázek 2.17: PWM signál s různou střídou[27].

Oproti metodě uvedené u žárovky má PWM nespornou výhodu v podpoře v MCU, kdy zpravidla i ty levnější a jednodušší modely jsou vybaveny časovači schopnými generovat tento signál [18]. Druhou výhodou může být možnost úspory elektrické energie, protože přebytečná energie zde není pálena na teplo pomocí rezistoru, ale díky odpojení zdroje se vůbec nespotřebuje.

2.4 Bezdrátová komunikace a komunikační protokoly

Schopnost navázat bezdrátové spojení mezi různými zařízeními je pro tuto práci klíčová. K přenosu informací se využívá různých fyzikálních principů, přičemž současně nejpoužívanějším je využití rádiových vln. V rámci této podkapitoly je rozebráno několik konkrétních implementací bezdrátového přenosu dat. Těmi jsou Wi-Fi, Bluetooth, Zigbee, rádiová frekvence 433 Mhz a infra paprsek. Tyto technologie pak doplňují různé protokoly. Zmíněny jsou především protokoly z počítačových sítí, jako IP, UDP, TCP, HTTP či MQTT.

2.4.1 Wi-Fi

Bezdrátový přenos dat Wi-Fi je definován ve standardu IEEE 802.11. Jedná se bezdrátovou variantu ethernetu. Může fungovat na různých frekvencích, v současnosti používá pouze tři a to 2,4 GHz, 5 GHz a 60 GHz. Ve všech třech případech se jedná o tzv. bezlicenční pásmo, na kterém může za určitých podmínek vysílat kdokoli bez nutnosti vlastnit speciální povolení. Standart je neustále vyvíjen s cílem dosáhnout vyšších přenosových rychlostí, snížení odezvy a zvýšení bezpečnosti. Jednotlivé verze se označují písmenkem (např. 802.11.n). Jednotlivé

[27] Převzato z http://www.siriusmicro.com/chrp3/pwm-c.html.

sítě se od sebe odlišují pomocí SSID (*Service Set Identifier*). Jedná se o řetězec o maximální délce 32 znaků [7]. Toto SSID je broadcastem vysíláno do okolí (toto chování lze potlačit, ale fakticky se nejedná o žádnou ochranu), čímž informuje zařízení v dosahu o přítomnosti sítě. Aby nebyl umožněn vstup do sítě libovolnému zařízení, je velmi vhodné vyžadovat autorizaci heslem.

Jelikož se fakticky stále jedná o LAN[28], fungují skrze ni vyšší protokoly počítačových sítí [7]. Přehled několika z nich:

MAC adresa slouží k jednoznačné identifikaci jednotlivých síťových rozhraní v rámci sítě [12]. MAC adresa je jedinečná 48bitová fyzická adresa. Prvních 24 bitů je tzv. OUI (Organizational Unique Identifier), tedy identifikátor výrobce. Zbylých 24 bitů je přiděleno výrobcem a mělo by být unikátní, aby nedocházelo ke kolizi. Komunikace na této úrovni probíhá pomocí tzv. rámců.

Internet Protocol zkráceně IP je o úroveň víš nad MAC adresami. Data jsou posílána po blocích, tedy tzv. *datagramech*. Tyto bloky jsou adresovány v rámci pomocí IP adresy. V případě IPv4 je 32bitová, novější IPv6 používá 128bitové adresy [12]. Kromě cílové adresy je do hlavičky datagramu doplněna i adresa zdroje, kontrolní součet a další informace.

Internet Protokol nezaručuje správné doručení datagramů, a proto je označován za nespolehlivý. Potřebuje-li aplikace spolehlivý přenos, je nutné ho implementovat pomocí některé z vyšších vrstev [12].

TCP neboli *Transmission Control Protocol* slouží ke spolehlivému přenosu dat pomocí IP. Než je vůbec zahájen přenos samotných dat, je nutné nejdříve navázat spojení s druhou stranou – *handshake*. Odesílaná data jsou dělena do segmentů a doplněna o TCP hlavičku. Přijaté segmenty jsou ověřeny pomocí kontrolních součtů a seřazeny do správného pořadí. Poté je potvrzeno přijetí. Pokud odesílatel nedostane informaci o přijetí, zahájí opětovný přenos. Díky tomu lze garantovat, že data budou přenesena kompletně a bez chyb [12]. Cenou je pomalejší navázání spojení a v nespolehlivých sítích i nižší rychlost přenosu vlivem častých opakovaných přenosů.

Na úrovni TCP je zároveň zavedena i adresace jednotlivých aplikací na jedné IP adrese pomocí portů. Jedná se o 16bitový číselný identifikátor, přičemž platí, že jeden port může v jednu chvíli využívat jen jedna aplikace [12]. Standart definuje tři skupiny portů a zároveň přiděluje jednotlivé porty konkrétním službám. Tzv. *nejběžnější služby* používají porty 0 – 1023. Např. HTTP běží na portu 80. Rozsah 1024 – 49151 slouží pro registrované služby. Zbylé porty 49152 – 65535 může využít jakákoli aplikace bez omezení [12].

UDP je zkratkou pro *User Datagram Protocol* a je doplňkem k TCP. Také používá porty a pro přenos dat IP. Oproti TCP ho ale lze označit na nespolehlivý, protože negarantuje přijetí datagramů ani jejich správné pořadí. Naproti tomu není nutné navázat spojení ani potvrzovat úspěšný přenos [12]. Cílí především na aplikace, kde je důležitější rychlost odezvy než bezchybnost přenosu. Příkladem mohou být např on-line hry nebo mediální vysílání.

HTTP je zkratkou pro *HyperText Transfer Protocol*, kde už název protokolu naznačuje jeho funkci, tedy přenos hypertextových dokumentů. Běžně běží na TCP portu 80 [12]. Tento protokol je podrobněji popsán v následující podkapitole 2.5.4.

[28] Local Area Network, česky místní síť

MQTT *Message Queuing Telemetry Transport* je jednoduchý protokol sloužící k předávání zpráv mezi klienty pomocí centrálního bodu. Cílí především na malá jednoduchá zařízení. Komunikace probíhá skrze TCP port 1883, přičemž spolehlivost definuje na třech úrovních – není zaručeno doručení zprávy, zpráva bude doručena alespoň jednou, zpráva bude doručena právě jednou. Je navržen na principu *publisher – subscriber*, tedy existuje jeden centrální bod (*broker*), který řídí výměnu zpráv. Ty jsou třízeny do témat (*topic*) a jednotlivá zařízení mohou v daném tématu buď publikovat nebo poslouchat [15]. Obsah zpráv není protokolem definovaný. Protože cílovou skupinou jsou jednoduchá zařízení, minimalizuje se množství přenášených dat.

2.4.2 Bluetooth

Standart Bluetooth (IEEE 802.15.1) poskytuje odlišný přístup k bezdrátové komunikaci. Obdobně jako Wi-Fi používá frekvenci 2,4 GHz, ale kromě formy komunikace specifikuje i jednotlivé podporované protokoly. Těchto protokolů je velké množství, takže je vybráno jen několik málo povinných. Zbylé jsou volitelné a zařízení je nemusí implementovat. Příkladem povinného protokolu může být SDP (*Service Discovery Protocol*), který slouží k získání informace o podporovaných protokolech. Naproti tomu jako nepovinné lze označit například protokol pro přenos zvuku či polohovací zařízení – myš[29]. Je zřejmé, že pomocí bezdrátových sluchátek není možné ovládat kurzor na obrazovce, a naopak myš neobsahuje reproduktor, tudíž nemusí umět přijímat zvuk. Proto by bylo zbytečné tyto specifické protokoly na těchto zařízeních podporovat.

Bluetooth je neustále vyvíjeno. Dřívější verze se zaměřovaly především na vzájemnou kompatibilitu a rychlost přenosu. Aktuální verze dále řeší i dosah a spotřebu[29]. Díky tomu je možné postavit zařízení, které bude na jedno nabytí baterie schopné běžet i několik měsíců.

2.4.3 ZigBee

Standart ZigBee (IEEE 802.15.4) patří do stejné skupiny bezdrátových sítí jako Bluetooth. Také definuje nejenom bezdrátovou komunikaci, ale i komunikační protokol. Oproti Bluetooth a Wi-Fi není cílem co nejvyšší přenosová rychlost, protože přenášené zprávy jsou poměrně krátké. Mimo jiné je podporována i topologie sítě typu mesh, kdy jednotlivá zařízení fungují zároveň jako vysílače, čímž rozšiřují efektivní dosah sítě[30].

2.4.4 Infra paprsek a rádiová frekvence 433 MHz

Tyto principy jsou zde uvedeny spíše pro úplnost. Přenos informace pomocí infra paprsku je náchylný na rušení a vyžaduje přímou viditelnost. V současnosti se používá především v dálkových ovladačích, kde stačí přenést několika bytové informace. Nevýhodou je potřeba přímé viditelnosti, popř. v omezené míře lze využít odrazu světelného paprsku. Tímto nedostatkem rádiová frekvence 433 MHz netrpí. Ta stejně jako např. 2,4 Ghz spadá do bezlicenčního pásma a je proto využívána v různých aplikacích. Příkladem mohou být například domácí meteostanice s venkovními čidly. Oba tyto způsoby komunikace spojuje absence jednotného komunikačního protokolu.

[29] Čerpáno z *What is Bluetooth Technology: basics & overview* [Electronics Notes]. Dostupné na: https://www.electronics-notes.com/articles/connectivity/bluetooth/what-is-bluetooth-technology-basics-summary.php.

[30] Čerpáno z *What is Zigbee?* [Electronics Notes]. Dostupné na: https://zigbeealliance.org/solution/zigbee/.

2.5 Webové technologie (HTML, styly, JavaScript, HTTP)

World Wide Web, neboli WWW či jen *web* je zřejmě nejpoužívanější službou běžící na Internetu[31]. Web se používá k přístupu k tzv. webovým stránkám a pro ukládání, ale i nahrávání dokumentů či jiných dat. Na pozadí je použit protokol HTTP, který byl lehce zmíněn v předchozí podkapitole. Dále bude popsán podrobněji včetně dalších důležitých technologií týkajících se webu.

2.5.1 HTML

HTML je zkratkou pro *HyperText Markup Language*. Jedná se textový formát souborů, který je zaměřen především na strukturovaný popis dat. Neustále se vyvíjí – aktuální je verze HTML5, která se více zaměřuje na strukturování obsahu a popis vzhledu přenechává na kaskádových stylech, což je asi nejmarkantnější rozdíl od předchozí verze [2]. Cílem HTML je tedy data obalit tzv. tagy (obdobně jako XML[32], ze kterého vychází) a definovat tak různé celky webové stránky (hlavička, menu, zvýrazněný text, odkazy, ...).

2.5.2 Kaskádové styly

Kaskádové styly, anglicky *Cascading Style Sheets* popř. zkráceně CSS, jsou důležitým doplňkem HTML souborů. Umožňují definovat různé parametry vzhledu jednotlivých prvků stránky (ohraničení, pozicování, velikost atd.). CSS používá takzvané selektory, tedy pravidla pro výběr prvku či skupiny prvků. Důležitou vlastností je ona kaskádovitost z názvu, tedy to, že podřazený prvek přebírá některé vlastnosti z nadřazeného (například velikost písma). Zároveň konkrétnější pravidlo má vyšší váhu než obecné [2]. V aktuální verzi CSS3, lze nově definovat i animace či plynulé přechody při změně některé z vlastností (např. plynulá změna barvy a šířky elementu při najetí myší).

2.5.3 JavaScript

V současnosti se jedná o nejjednodušší a nejrozšířenější možnost, jak oživit jinak statickou webovou stránku. Programátor tedy může definovat, jak má stránka reagovat na vstupy od uživatele (akce myší, psaní na klávesnici, vstupy z dotykové obrazovky), popř. na pozadí navázat komunikaci se serverem a dynamicky načítat data bez nutnosti opustit aktuálně načtenou stránku. To znamená, že skript na pozadí vyvolá HTTP dotaz na server a posléze zpracuje odpověď (nebo chybu), přičemž uživatel ani nepostřehne, že probíhá komunikace, tedy pokud o tom není informován skriptem samotným [14]. Některé z důležitých vlastností nově přebírá samotné HTML5 a CSS3, jedná se především o možnost validovat zadaná data do formuláře (bez nutnosti je odesílat na server) v případě HTML5, popř. o animace a přechody v rámci CSS3. Ovšem na to ostatní je stále nutné použít JavaScript.

S použitím JavaScriptu souvisí i datový formát JSON, což je zkratkou pro *JavaScript Object Notation*. Je to značkovací jazyk, který slouží k popisu JavaScriptových objektů ve formě textu a usnadňuje tak jejich přenos mezi serverem a klientem. JSON neumí specifikovat strukturu přenášených dat ani jejich datové typy [14]. Úkolem aplikace je tyto informace určit na základě kontextu, popř. se vyrovnat s chybnými daty.

[31] Celosvětová počítačová síť
[32] Extensible Markup Language

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Ukázka</title>
  </head>
  <body>
    <div class="modry-text">
      Prázdná stránka!
    </div>
  </body>
</html>
```

```css
div {
  font-size: 200%;
  color: red;
}

div.modry-text {
  color: blue;
}
```

```json
{
  "svetla" : [
    {
      "jas": 80,
      "barva": "#42fe42"
    },
    {
      "jas": 90,
      "barva": "#87c1a9"
    }
  ]
}
```

Obrázek 2.18: Ukázka jednoduché html stránky (*vlevo*) obsahující pouze jednu větu, změnu stylu textu pomocí CSS (*uprostřed*), tedy jeho zvětšení na dvojnásobnou velikost a změnu barvy na modrou. Zcela *vpravo* je příklad struktury ve formátu JSON.

2.5.4 HTTP

Jak už bylo zmíněno v předchozí podkapitole, slouží HTTP, neboli *HyperText Transfer Protocol*, k přenosu nejen html souborů v rámci internetu. Protokol používá princip serveru, ke kterému se připojují jednotlivý klienti, přičemž komunikace probíhá stylem dotaz – odpověď. Jedná se o textový protokol, tedy veškeré dotazy i odpovědi jsou lidsky čitelné. Komunikaci vždy zahajuje klient, tedy zpravidla webový prohlížeč na pokyn uživatele. To znamená, že se naváže spojení (typicky pomocí TCP na portu 80) a odešle se dotaz. Server dotaz zpracuje a vygeneruje odpovídající odpověď. V původní verzi ukončoval spojení server, v novější verzi záleží na vzájemné dohodě mezi serverem a klientem [8]. Nových verzí vzniklo už několik, v současnosti se pracuje na další. Níže je uveden přehled několika starších verzí, důraz je dán především na verzi `HTTP/1.0`, která přináší vše, co je pro návrh jednoduchého zařízení potřeba.

HTTP 0.9 bylo první veřejnou verzí. Zavedlo princip dotaz – odpověď a formát dotazu. Tato verze HTTP znala jen jednu metodu, kterou byla metoda `GET`. Validní dotaz na server tak vypadal následovně: `GET /index.html<CR><LF>` (symboly `CR` a `LF` zde symbolizují zakončení řádku). Odpovědí byl soubor `index.html` ve formě streamu ASCII znaků. Na závěr přenosu server uzavřel spojení, čímž klient dostal znamení, že má k dispozici celý soubor a může jej vykreslit [8]. Případná chyba byla přenesena také ve formě textu, tudíž klient nemohl jednoduše určit, zda a proč k chybě došlo, bez toho, aby se snažil pochopit odpověď.

HTTP/1.0 bylo velmi důležitým nástupcem. Přineslo dvě nové metody – `HEAD` (podobné `GET`, ale odpovědí je pouze HTTP hlavička bez těla. Lze použít např. k zjištění, zda na serveru neexistuje nová verze dokumentu) a `POST` (umožňuje odesílat data na server, ty jsou součástí dotazu). Dále pak přináší koncept HTTP hlaviček [8]. První řádek dotazu je podobný, jako v předchozí verzi, nově přibila informace o verzi protokolu, kterou klient používá.

Další řádky pak představují jednotlivé hlavičky (verze prohlížeče, podporované formáty, jazyk, cookies atd.) ve formátu `Jméno: hodnota<CR><LF>`. Tělo zprávy je odděleno prázdným řádkem, přičemž metody `GET` a `HEAD` mají tělo prázdné, metoda `POST` zde má odesílaná data [8].

```
GET /index.html HTTP/1.0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept-Language: cs-CZ,cs;q=0.9
```

Obrázek 2.19: Příklad HTTP/1.0 dotazu.

Odpověď nově neobsahuje pouze dokument, ale na prvním řádku je uveden stavový kód odpovědi. Za ním mohou následovat různé HTTP hlavičky ukončené opět prázdným řádkem [8]. Na závěr je prostor pro samotnou odpověď.

```
HTTP/1.0 200 OK
Date: Tue, 09 Mar 2021 20:27:56 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8

<!DOCTYPE html>
...
```

Obrázek 2.20: Příklad HTTP/1.0 odpovědi.

Zavedení stavových kódů řeší jeden z neduhů předchozí verze, tedy obtížné rozpoznání chyby. Skupina kódů je poměrně rozsáhlá (současná verze jich definuje několik desítek) a do budoucna snadno rozšiřitelná [8]. Každý kód má své třímístné číslo, přičemž první číslice je dělí do pěti skupin.

1xx Informační kódy informují o stavu serveru [8].

2xx Úspěch potvrzují úspěšné vyřízení požadavku. Např. velmi častý kód `200 OK`, který je následován daty, o které klient požádal. Druhým zajímavým (z novější verze `HTTP/1.1`) je `204 No Content`, který potvrzuje úspěšné přijetí a zpracování dat, ale klientovy nejsou zaslána žádná data jako odpověď [8].

3xx Přesměrování slouží k informování o nové adrese na které se dokument či server nachází [8].

4xx Klientská chyba značí, že dotaz od klienta nebyl validní [8].

5xx Serverová chyba znamená, že server z jakéhokoli důvodu nezvládl požadavek zpracovat [8].

HTTP/1.1 cílí na snížení režie přenosu. Snižuje režii nutnou pro navazování TCP spojení tím, že implicitně zapíná podporu pro udržení jednoho spojení pro více dotazů. To vyžaduje povinné použití hlavičky `Content-Length` v odpovědi informující o celkové velikosti odpovědi. Dále přibyla podpora více virtuálních WWW serverů na stejné IP adrese a portu. Proto musí být v dotazu uvedena hlavička `Host` obsahující adresu WWW serveru [8]. Přibily také nové metody, hlavičky i stavové kódy.

Novější verze (1.2, 2.0 a 3.0) se dále zaměřují na efektivitu protokolu a také jeho bezpečnost. Není nezbytně nutné je zde podrobněji popisovat.

Přenos dat od klienta na server

Byť to tak z předchozího popisu nemusí vypadat, tak k přenosu dat lze použít dva různé přístupy. Každý z nich přináší určité výhody i nevýhody a výběr vhodné metody záleží tak především na konkrétní situaci.

Metoda `POST` je součástí protokolu už od verze 1.0 a je přímo určená k nahrávání dat. Ty jsou umístěna do těla dotazu, tudíž není omezena jejich délka ani formát.

Metoda `GET` na první pohled slouží přesně k opačnému účelu – získávání dat, ale přesto ji lze využít i k nahrávání. Postupuje se tak, že se data pro server vhodným způsobem zakódují do URL adresy dotazu. Od samotné adresy se oddělují otazníkem a lze tak přenášet pouze jednoduchá data ve tvaru `jméno=hodnota`. Větší množství těchto dvojic se od sebe odděluje ampersandem. Je možné uvést i pouze `jméno` bez hodnoty, ale některé servery to nemusí podporovat[33]. Příklad použití:

```
http://192.168.1.122/nastav.php?jas=90&barva=54de5a
```

Výhodou tohoto přístupu je možnost si takový příkaz snadno uložit jako odkaz, popř. jej dokáže sestavit i méně pokročilý uživatel pouhým zapsáním do webového prohlížeče bez nutnosti použít nějaký pokročilejší software. V tomto ohledu je použití metody POST o mnoho složitější. Na druhou stranu je nutné řešit přenos ne-ASCII znaků, složitějších datových struktur a většího množství dat. Protokol sice nedefinuje maximální velikost URL adresy, přičemž doporučuje podporovat minimálně 8 000 znaků, na druhou stranu podpora dlouhých adres mezi browsery a servery je různá a nedoporučuje se překračovat 2 000 znaků[34].

Tento přístup lze dále modifikovat pro malé množství dat. Lze je zakódovat přímo do adresy dotazu, který server zpětně zpracuje na jednotlivé proměnné. Adresa pro změnu barvy jasu z předchozího příkladu by tak vypadala například následovně:

```
http://192.168.1.122/nastav/jas/90
```

Na serveru se samozřejmě žádný dokument `90` ve složce `jas` nenachází. Server si adresu vnitřně přeloží a zavolá soubor `nastav` s parametrem `jas=90`. Výhodou tohoto přístupu jsou lidsky čitelnější odkazy, což může být v určitých aplikacích podstatná výhoda.

[33] Čerpáno z *Uniform Resource Identifier (URI): Generic Syntax* [RFC3986]. Dostupné na https://tools.ietf.org/html/rfc3986.

[34] Čerpáno z *What is the maximum length of a URL in different browsers?* [Stack Overflow]. Dostupné na: https://stackoverflow.com/a/417184.

Kapitola 3

Zhodnocení současného stavu a upřesnění záměru zadání

Z předchozí kapitoly je patrné, že existují dostatečné technologie pro návrh i realizaci bezdrátového osvětlení do domácnosti. Toho jsou si vědomi i výrobci a v současnosti nabízí různé řešení tohoto problému. Ty se od sebe liší jak základními parametry (jako jsou barva vyzařovaného světla, světelný výkon, životnost aj.), tak i použitou technologií a schopnostmi. Samozřejmě je důležité neopomenout i výslednou cenu za dané řešení. Ta není pouze součtem cenovek jednotlivých prvků, ale je nutné do ní započíst i náklady spojené s výměnou stávající elektroinstalace, popř. síťové infrastruktury a čas strávený instalací a nastavením nového osvětlení. Samozřejmě je nutné zhodnotit i celkový uživatelský komfort.

3.1 Hodnocení existujících řešení

Uživatel si může vybrat několik různých přístupů, jak dosáhnout bezdrátově ovládaného osvětlení. Záleží pak, zda chce bezdrátově ovládat celou domácnost nebo pouze několik světel. Jestli chce koupit hotové řešení v jedné krabici nebo naopak kombinovat prvky různých výrobců, či si je navrhnout a vyrobit zcela sám.

Název/Řada	Barva	Cena	Komunikační protokol	Hodnocení aplikace[1]	Nástěnné vypínače
Philips Hue	RGBW	137,-	ZigBee[2]	4,5/4,7	Ano
IKEA TRÅDFRI	Bílá	25,-/75-[3]	ZigBee	3,8/4,4	Ano
RF LED pásek	RGB	55,-	Ne	-	Ne
TP-LINK Tapo L530E	RGB	50,-	Neoficiální	4,6/4,7	Ne
PLAYBULB Smart Bulb	RGBW	285,-	Ne	2,2/2,8	Ne
Sonoff MINI	-	350,- za modul	HTTP	4,1/2,7	Ano
ESPHome	-	-	MQTT	-	Ano

Tabulka 3.1: Porovnání různých řešení. Pro cenové srovnání byly vybrány modely popsané v kapitole 2.1 a pro jednodušší porovnání byla cena přepočtena za 100 lumenů.

[1] Hodnocení mobilní aplikace je ve tvaru Google Play/Apple App Store.
[2] Výrobce oficiálně nespecifikuje a není zaručena kompatibilita s jinými prvky.
[3] RGB varianta.

Výše uvedení tabulka 3.1 souhrnně porovná různé řešení skrze jejich technické vlastnosti. Na základě této tabulky jsou v další podkapitole specifikovány požadované klíčové vlastnosti. Cena se s ohledem na různé schopnosti a výkon špatně porovnává, navíc některá řešení pro plné využití svého potenciálu vyžadují pořízení tzv. *brány*. Přesto je zajímavé, že suverénně nejdražší žárovka *Smart Bulb* patří k horším řešením. Firmware *ESPHome* pak nemá vůbec specifikovanou cenu, protože ta se odvíjí od použitého hardware.

Philips Hue je kompletní řešení od jednoho výrobce. Je určeno především pro uživatele, co nechtějí řešit technické pozadí a spíše než technologie, kterou to používá je zajímá, zda to funguje. Tedy si troufám tvrdit, že se jedná o drtivou většinu uživatelů. Byť to Philips nikde viditelně neuvádí, tak používá technologie ZigBee. Kombinace s výrobky jiných výrobců by podle všeho měla být možná, ale není ze strany výrobce nijak zmíněna. Řada *Hue* nabízí velké množství svítidel a světelných zdrojů v různém provedení. Ty doplňuje dálkově ovládaná zásuvka či bezdrátový vypínač, což usnadní přechod konzervativních uživatelů na tento systém.

IKEA TRÅDFRI do jisté míry připomíná *Philips Hue*. Sice nenabízí tak široký ekosystém výrobků nebo propracovanou aplikaci, ale naopak nízkou cenu a uživateli ověřenou kompatibilitu s jinými systémy na bázi ZigBee. Pokročilejším uživatelům pak přináší další možnosti, např. bezdrátové ovládání pro libovolný LED pásek nebo jednoúčelové tlačítko se sadou nálepek, které je možné nastavit pro vykonání určené akce.

RF LED pásek patří (s ohledem na délku pásku) mezi jednodušší a levnější řešení. Vzhledem k nutnosti používat dedikovaný dálkový ovladač a nízké možnosti integrace s jinými prvky se hodí spíše tam, kde uživatel nechce bezdrátově ovládat celou domácnost, ale pouze specifické svítidlo. Např. osvětlení obývacího pokoje k večernímu sledování televize. V tomto případě může být toto řešení zcela dostačující.

TP-LINK Tapo L530E je samostatnou Wi-Fi žárovkou. Uživatel se musí spokojit s ovládáním skrze mobilní aplikaci nebo hlasového asistenta. Komunikační protokol výrobce nezveřejnil, ale lze dohledat jeho neoficiální implementaci[4]. Podle všeho žárovky naslouchají na portu 9999, přičemž zprávy jsou přenášeny pomocí UDP protokolu. Kromě RGB žárovky lze pořídit také bílou nebo dálkově ovládanou zásuvku, jiné možnosti výrobce v tuto chvíli nenabízí. Nemožnost snadné integrace s okolím a absence nástěnných vypínačů je velkou překážkou pro použití v rámci celé domácnosti. Ale pokud výrobce začne nabízet i vypínače, popř. si uživatel vytvoří vlastní, mohlo by se jednat o zajímavou a jednoduchou alternativu.

PLAYBULB Smart Bulb v tomto seznamu zastupuje žárovky ovládané pouze skrze Bluetooth. Nutno dodat že tento typ osvětlení z nabídky postupně mizí, s ohledem na možnosti, které nabízí konkurence je to pochopitelné. Předchozí žárovka *TP-LINK Tapo* je sice taky jednoduchá, ale díky Wi-Fi nabídne ovládání v rámci dosahu domácí sítě. Naopak žárovku *PLAYBULB* omezuje dosah Bluetooth, takže prakticky je ovladatelná jen v rámci dané místnosti, případně ještě ze sousedních místností v závislosti na tloušťce zdí. Dále může docházet k problémům s párováním Bluetooth a obecně uživatelé vyjadřují nespokojenost s funkčností aplikace.

Sonoff MINI sice není světelným zdrojem, ale naopak nabízí jednoduchou možnost, jak změnit běžné vypínače na dálkově ovládané. Sice neumožňují ovládat jas, na druhou

[4]Dostupné na https://github.com/konsumer/tplink-lightbulb.

stranu není nutné měnit stávající vypínače či osvětlení. Tedy někteří uživatelé mohou ve zdi mít plytké elektroinstalační krabice, což přeměnu zkomplikuje. Pokročilí uživatelé mohou nahrát alternativní firmware a využít tak pouze výrobcem dodaný hardware v kombinaci s vlastním softwarovým řešením.

ESPHome na rozdíl od ostatních zde zmíněných řešení není komerčním výrobkem. Je to pouze firmware pro mikrokontroléry firmy Espressif. Ty někteří výrobci využívají i ve svých výrobcích a lze tak kombinovat profesionálně vypadající produkt a vlastní softwarové řešení (např. výše zmíněné vypínače firmy *Sonoff*). Kromě osvětlení jsou zde implementovány i jiné prvky chytré domácnosti. Pokročilým uživatelům tak dává spoustu možností, jak si vytvořit chytrou domácnost i bezdrátově ovládané osvětlení přesně podle svých možností. Součástí projektu není centrální jednotka, což sice dává uživateli volnost v jejím výběru, na druhou stranu to komplikuje realizaci.

3.2 Nevýhody a kompromisy současných řešení

V souvislosti s předchozí podkapitolou je nutné zmínit i kompromisy případně nevýhody daného řešení. Přičemž některé z nich jsou dány použitým řešením, jiné pak až konkrétní realizací daného výrobce.

Dálkový ovladač včetně jeho nevýhod byl už zmíněn dříve. Týká se především různých LED pásků dodávaných jako set, ale třeba i nejlevnější žárovky z řady *Philips Hue*. Řešením by bylo velké množství ovladačů pevně umístěných na daných místech (např. u dveří, jako klasický vypínač), které by doplnila sada přenosných ovladačů, aby uživatel mohl profitovat z dálkového ovládání. Ovšem nemožnost snadné integrace s jinými prvky či absence ovládání z telefonu dělají toto řešení poměrně neatraktivní.

Uzavřený komunikační protokol nebo nemožnost integrace je problémem pouze některých (spíše jednoduchých) zařízení. Ze sedmy zmíněných výrobků se to sice týká hned tří, ale výběr byl udělán tak, aby zastoupil různé přístupy, proto z této hodnoty nelze odvozovat obecné závěry. Je čistě na uživateli, zda jej to bude nějak omezovat nebo naopak raději použije pokročilejší řešení.

Výměna současného osvětlení není nic, co by ekonomicky i ekologicky zaměřený uživatel chtěl podstoupit. Další překážkou může být i neexistence vhodné alternativy (např. designové žárovky do svítidla). Řešením může být postupná výměna prvků za dálkově ovládané, jenže v tom případě se naplno neprojeví výhody kompletně dálkově ovládaného osvětlení, a navíc může nastat situace, kdy starší zařízení nebudou kompatibilní s novými. Druhou možností je změna současných svítidel na dálkově ovládané např. pomocí vypínačů firmy *Sonoff*, dálkově ovládaných zásuvek (např. *Philips Hue*) nebo bezdrátového ovladače pro RGB pásek (*IKEA TRÅDFRI*).

Závislost na aplikaci je poměrně velkou překážkou. Zejména pokud neexistuje i jiný způsob ovládání. Sáhnout na známé místo a cvaknout vypínačem je totiž mnohonásobně rychlejší a spolehlivější než hledat aplikaci ve smartphonu. Aplikace by měla být pouze alternativou ke klasickému stylu ovládání, případně by měla zpřístupňovat pokročilejší možnosti (spínání pomocí hodin, změna jasu či barvy atd.). Mimo to může narazit uživatel na nekompatibilitu s verzí jeho operačního systému či systému samotného.

Závislost na internetovém připojení je neopodstatněným požadavkem. Výrobce tak může řešit možnost ovládat či kontrolovat osvětlení i když je uživatel mimo domácí síť. Ovšem tato volba by měla být volitelná, ale hlavně by výpadek připojení k internetu, popř. odstávka serveru výrobce neměla způsobit nemožnost osvětlení ovládat. Naštěstí tento problém není zas tak rozšířený. Ze zmíněných modelů by se tento problém neměl týkat žádného z nich.

Nespecifikované chování při výpadku spojení nebo proudu je naproti tomu dosti rozšířené. Respektive výrobce toto chování nějakým způsobem ve svém software specifikoval, ale tuto specifikaci již nenapsal do dokumentace. Uživatel tak zpravidla neví, jak se zařízení po výpadku proudu zachová, případně toto chování nelze ovlivnit. Přitom se může jednat o klíčovou vlastnost. Pokud by světlo po zapnutí vždy naběhlo do stejného stavu jako před výpadkem, bylo by možné jej osadit do lustru přes klasický vypínač a dálkově ovládat jen jeho jas. To stejné pak lze říct i o výpadku spojení s centrální jednotkou. V některých situacích, např. by mohlo být žádoucí zapnout osvětlení na nízký jas během výpadku spojení ve večerních hodinách u světla na chodbě bez oken. Oproti předchozímu bodu se tento týká všech modelů.

3.3 Určení klíčových vlastností

Na základě předchozích informací jsem se rozhodl pro návrh a výrobu vzorku zařízení, které by se svými vlastnostmi blížilo k firmware *ESPHome*. Chtěl jsem si vyzkoušet implementaci obdobného projektu a při tom některé věci vyřešit jinak (absence řídící jednotky, méně přívětivé uživatelské rozhraní). Požadované vlastnosti by se daly shrnout následovně:

- Ke komunikaci slouží domácí Wi-Fi síť
- Ovládání skrze webové rozhraní (bez nutnosti použít speciální aplikaci)
- Komunikace probíhá pouze lokálně
- Existence různých modulů (světelné zdroje, vypínače, tlačítka, čidla, ...)
- Řídící jednotka je součástí řešení
- Každý modul je ovladatelný i samostatně skrze plnohodnotné webové rozhraní

Oproti ESPHome jsem se rozhodl jít jednodušší cestou, tedy omezit se pouze na osvětlení a vynechat tak různá čidla a jiné prvky chytré domácnosti. Naopak výsledné řešení obsahuje i centrální řídící jednotku. Ovládání i konfigurace probíhá graficky skrze webové prostředí, které je více zaměřeno na osvětlení a použití na mobilních zařízeních. Z bezdrátové technologie jsem si vybral Wi-Fi, především pro její větší dostupnost a jednodušší integraci do stávající domácí sítě.

Práce cílí na pokročilejší uživatele, proto součástí výsledku musí být i dobře specifikovaný komunikační protokol. Ten doplní podpůrné funkce pro snadné vytváření vlastních nových modulů. Samozřejmě nelze zapomenout ani na samotnou hardwarovou realizaci, tedy návrh schéma zapojení i DPS pro jednotlivé moduly. Ty doplňuje i schéma prototypové desky modulu. Aby si budoucí uživatelé mohli moduly snadno vyrobit i doma, tak bude vhodné, když navržené DPS nebudou příliš složité a upřednostní se klasická pouzdra součástek před SMD.

Kapitola 4

Experimentální realizace bezdrátově ovládaného osvětlení

Před samotnou realizací je nejdříve nutné specifikovat základní vlastnosti zařízení a cíle, kterých má být dosaženo. To bylo provedeno v minulé kapitole. Na jejich základě je vytvořen obecný návrh, který je postupně konkretizován až do finální podoby. Konkrétně se jedná o rozdělení zadání na jednotlivé podproblémy. Těmi jsou návrh hardwarového řešení, mezimodulová komunikace, interakce s uživatelem a centrální jednotka.

V rámci hardwarové části se jedná především o schéma zapojení a odpovídající desku plošných spojů. Na to navazuje návrh a realizace programu pro mikrokontrolér i řídící jednotku. Bude vhodné, pokud oba tyto programy budou vycházet z jednoho společného základu, což zajistí jednotný přístup k řešení jednotlivých problémů (např. komunikačního protokolu) a sníží množství potřebného kódu. Interakce s uživatelem bude probíhat pomocí webového prohlížeče, tedy bude nutné navrhnout odpovídající webovou stránku. Závěrem se musí otestovat všechny tyto části a také určit potenciální rizika a celkovou bezpečnost použitých řešení.

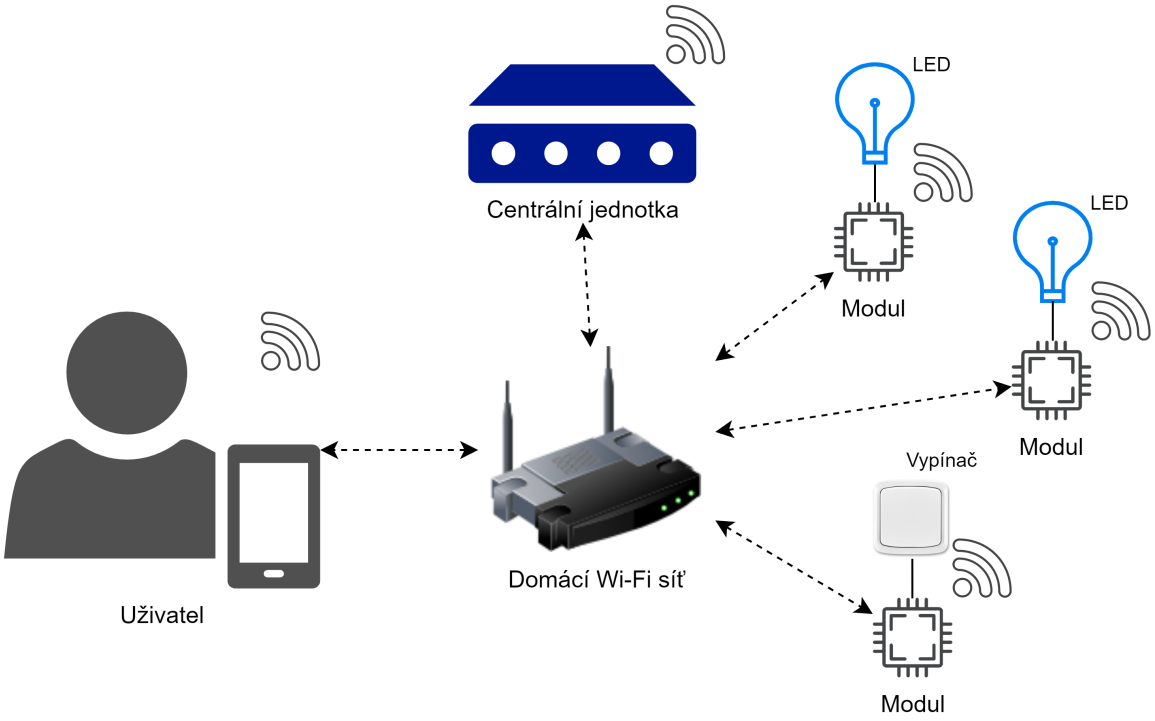
4.1 Základní návrh a výběr komponent

Návrh začíná obecným blokovým schématem, které pomůže lépe vizualizovat jednotlivé části úkolu a jejich vzájemné vazby. Konkrétně se jedná o podproblémy zmíněné v úvodu této kapitoly – hardwarová část, komunikační protokol, webový interface a centrální jednotka. Výhodou členění na menší bloky je možnost pracovat na nich průběžně, např. návrh webové stránky je zcela nezávislý na použitém hardwaru.

4.1.1 Základní návrh

Jedním z prvků specifikace zadání bylo členění osvětlení do tzv. modulů, přičemž každý modul má být do jisté míry autonomní. Uživatel s ním může komunikovat buď přímo pomocí webové stránky nebo nepřímo skrze centrální jednotku, která sdružuje všechny moduly na jednom místě. Ta zároveň umožňuje vytvoření různých pravidel a automatizaci. Výsledný systém osvětlení použitý v domácnosti tak bude tvořen větším množství jednodušších modulů a jednou centrální jednotkou se sadou pravidel.

Blokové schéma na obrázku 4.1 znázorňuje klíčové bloky práce. Na levé straně se nachází uživatel, který se systémem komunikuje pomocí Wi-Fi. Z pohledu uživatele k tomu slouží webová stránka, kterou si zobrazí na libovolném zařízení, například na jeho chytrém



Obrázek 4.1: Základní blokové schéma.

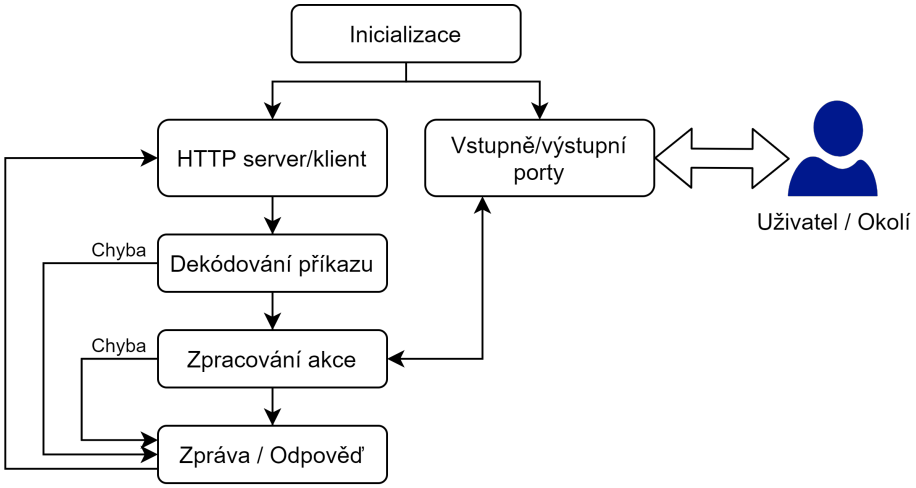
telefonu, který má zpravidla po ruce. Z technického pohledu to vyžaduje specifikovat komunikační protokol mezi webovou stránkou a zařízením. Druhou stranou systému (pravá strana schématu) jsou samotné moduly. I ty komunikují pomocí Wi-Fi. Přičemž se může jednat buď o moduly, které něco vykonávají (např. svítí) nebo naopak slouží k přímé interakci s uživatelem (vypínač). S těmito prvky může komunikovat uživatel přímo nebo pomocí řídící jednotky. Ta může na základě pravidel zasílat příkazy i bez nutnosti interakce uživatele.

4.1.2 Chování modulu a centrální jednotky

Jak jednotlivé moduly tak centrální jednotka mají společnou základní programovou smyčku, tu znázorňuje obrázek 4.2. Nejdříve samozřejmě proběhne inicializace mikrokontroléru, jeho modulů (Wi-Fi, GPIO, FLASH paměť) a HTTP serveru. Důvod použití HTTP protokolu je rozebrán v následující podkapitole 4.1.3. Následně modul čeká na příchozí HTTP příkaz, který dekóduje a pokud byl validní, tak i vykoná. O výsledku, ať už úspěšném nebo neúspěšném, informuje odesílatele příkazu pomocí HTTP odpovědi. Ta zároveň může i obsahovat nějaká data, v závislosti na konkrétním příkazu.

Kromě toho se moduly (i centrální jednotka) mohou stát i původcem nějakého příkazu. U modulů se bude jednat typicky o reakci na vstup od uživatele (stisknutí vypínače) nebo změnu stavu (např. změna jasu). Centrální jednotka pak generuje příkazy na základě vyhodnocení pravidel.

Jednotlivé příkazy představují buď tzv. *akce* nebo *události*. Obojí jsou to klíčové prvky definující chování jednotlivých modulů. Akce jsou příkazy, které může vyvolat uživatel skrze webové prostředí řídící jednotky nebo modulu, popř. jsou vyvolány automaticky na pozadí. Akce může získávat informace (aktuální jas) nebo je měnit. Každý modul obsahuje sadu



Obrázek 4.2: Základní programová smyčka obsažená v modulu i řídící jednotce.

základních akcí, které musí implementovat a dále sadu rozšiřujících akcí, která definuje samotné chování konkrétního modulu (např. bílá žárovka nebude mít akci pro změnu barvy). Kódování akce do HTTP příkazu (respektive URL adresy) vypadá následovně

```
http://adresaModulu/jmenoAkce?parametr1=hodnota1&parametr2
```

kdy bude spuštěna akce se jménem `jmenoAkce` a dvěma parametry – `parametr1` s hodnotou `hodnota1` a `parametr2` bez konkrétní hodnoty.

Oproti akcím je událost vyvolána na modulu a směrována vždy na centrální jednotku. Může být vyvolána buď nějakým externím vstupem (např. vypínač) nebo jinou akcí. Události jsou základním kamenem pro vytváření pravidel a s tím související automatizace.

4.1.3 Komunikační protokol modulu, řídící jednotky a webového rozhraní

Výběr vhodného komunikačního protokolu závisí na několika okolnost. Je nutné zvážit jednak výkon samotného zařízení, ale také očekávané množství, délku a obsah přenášených zpráv nebo to, zda se má jednat spíše o textově nebo naopak binárně orientovaný přenos dat. V tomto konkrétním případě bude mezi modulem a řídící jednotkou přenášeno pouze malé množství krátkých zpráv. Spíše, než přenosová rychlost je důležitá spolehlivost a doba odezvy. Tedy nároky na samotný protokol jsou poměrně nízké. Nabízí se tak použití protokolu MQTT, který byl přímo navržen pro tento účel.

MQTT	HTTP
binární	textový
nespecifikuje obsah zprávy	nespecifikuje obsah zprávy
hůře čitelné surové zprávy	snadno čitelné surové zprávy
kratší zprávy	delší zprávy
nutnost souběhu s HTTP serverem	postačí jeden server
nutnost překladu z/na HTTP	přímá podpora HTTP
pravidla vyhodnoceny na daném modulu	pravidla vyhodnoceny na centrální jednotce

Tabulka 4.1: Porovnání protokolů MQTT a HTTP.

Pokud by byl použit protokol MQTT, tak by jednotlivé moduly informovaly centrální jednotku o událostech (stisknutí tlačítka, změna jasu), a ta by tyto informace přeposílala pouze těm modulům, které si o to zažádali. Vykonávání pravidel by tak bylo realizováno přímo na jednotlivých modulech. Nevýhodou tohoto přístupu je správa pravidel, která není centralizovaná. Popř. by bylo nutné její centralizaci simulovat, kdy by uživatel pravidla definoval skrze rozhraní centrální jednotky, která by na pozadí tyto informace předávala jednotlivým modulům.

Druhým vhodným protokolem je HTTP. A to zejména proto, že modul má poskytovat uživatelské rozhraní skrze webovou stránku. Tudíž na modulu musí být implementován HTTP server, byť postačí nějaká primitivní implementace. Pokud by vedle něj běžel i MQTT server, bylo by nutné buď vyřešit překlad HTTP příkazů na MQTT, aby mohl uživatel ovládat modul ze svého webového prohlížeče nebo implementovat druhou sadu příkazů i pro HTTP. Jednodušší varianta je pak použít pouze HTTP. Nevýhodou je vetší množství přenášených dat, protože HTTP je protokol textový a u krátkých zpráv je velikost hlavičky větší než zpráva samotná. Naopak výhodou může být snazší diagnostika přenášených zpráv (jedná se o ASCII text) a jejich generování ať už z konzole (nástroj *telnet*) nebo přímo pomocí webového prohlížeče.

Třetí volbou se nabízí vytvoření zcela vlastního protokolu. To sice umožňuje vytvořit protokol zcela na míru potřebám a omezit přenos zbytečných informací. Ale stále tu zůstává potřeba souběhu s HTTP serverem a navíc to přináší komplikace pro integraci s jinými zařízeními či budoucí rozvoj.

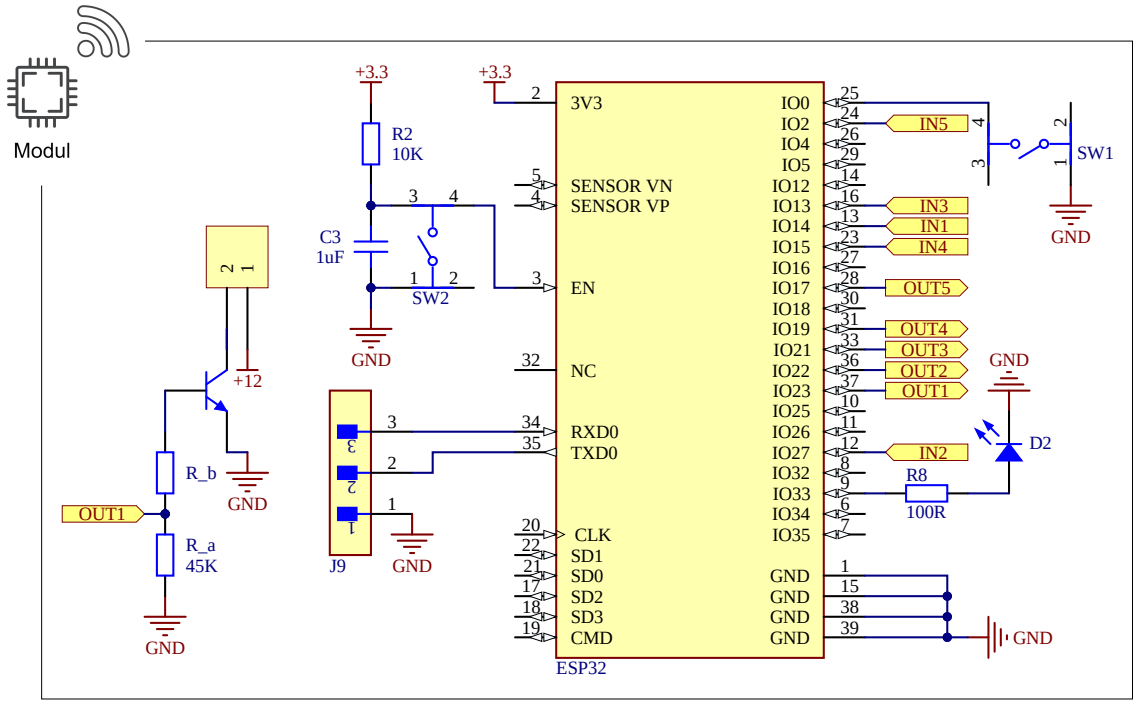
Protože tato práce cílí na pokročilejší uživatele, kteří by mohli chtít diagnostikovat komunikaci nebo vytvářet nové moduly, zvolil jsem protokol HTTP. V návaznosti na toto rozhodnutí je potřeba určit způsob, jakým budou přenášeny data. HTTP má k tomu určeno metodu *POST*, jejíž použití samo o sobě nepřináší žádné větší nevýhody. Snad jen nutnost načíst celou hlavičku a následně i tělo zprávy, které obsahuje samotná užitečná data. Naproti tomu přenos dat metodou *GET* umožňuje zpracovat pouze první řádek zprávy a zbylé ignorovat, protože nenesou již žádné další potřebné informace (v tomto konkrétním případě). Byť to přináší omezení v maximálním množství přenesených dat a nutnosti escapovat speciální znaky, tak jsem s ohledem na rychlejší zpracování zvolil tento přístup.

4.1.4 Výběr komponent a první schéma zapojení

Srdcem každého inteligentního zařízení je v současné době počítač. Nejinak tomu je i u bezdrátově řízeného osvětlení. Na trhu lze najít nepřeberné množství různých mikrokontrolérů. Já si vybral poměrně nový `ESP32`, což je dvoujádrový 32bitový MCU integrující Wi-Fi, přičemž jedno jádro je vyhrazeno pro běh uživatelského programu a druhé pro obsluhu Wi-Fi a síťové komunikace. Díky integraci Wi-Fi přímo na čipu bude možné dosáhnout nižší spotřeby, vyšší rychlosti a malých rozměrů zařízení, jelikož bude potřeba jen minimální množství externích součástek. Mezi další přednosti tohoto MCU patří velké množství vstupně/výstupních pinů. Díky tomu je možné generovat PWM signál pro řízení jasu až 16 různých LED. Je vybaven i podporou pro dotykové senzory, což bude možné využít pro vytvoření dotykových vypínačů.

Schéma na obrázku 4.3 představuje zjednodušené zapojení prototypové desky (kompletní schéma je v příloze na obrázku A.1). Srdcem je výše zmíněný ESP32. Jeho napájení obstarává hotový modul step down měniče (`QS-1205CME-3A`) doplněný o kondenzátory pro vykrytí napěťových špiček. Napájecí napětí zapojení může být v rozsahu (5 V – 24 V), přičemž je přímo použito pro napájení ovládaných LED modulů, proto se ve schématu

uvažuje běžných 12 V. Pro přehlednost je tato část zapojení uvedena pouze v kompletním schématu v příloze A.1. Mikrokontrolér vyžaduje zdroj schopný dodat minimálně 500 mA, byť průměrná spotřeba se pohybuje okolo 80 mA. Použitý napájecí modul je schopný dodat až 3 A (1,5 A dlouhodobě bez chlazení) – to je dostatečná rezerva.



Obrázek 4.3: Schéma zapojení prototypové desky modulu.

Dále jsou zde přítomny dvě tlačítka. Jedno (`SW2`) pro restart MCU doplněné o RC článek, dle doporučení výrobce. Druhé (`SW1`) pak během restartu slouží pro přepnutí do programovacího módu. Kromě toho ho lze využít i jako vstupní za běhu programu. UART komunikace je vyvedena na konektor pro snadné programování a komunikaci s PC. Konektor `J3` (zakreslen v příloze) budou sloužit k přepnutí do režimu přístupového bodu s výchozím nastavením. To vše doplňují dvě indikační diody, první (`D1`) indikující přítomnost napájecího napětí (opět zakreslena pouze v příloze) a druhá (`D2`) je ovladatelná z MCU. Běžnou indikační diodou může téct proud typicky do 20 mA, při úbytku napětí okolo 1,8 V. Hodnota ochranných rezistorů `R1` a `R8` pro proud 15 mA se určí následovně

$$R_{LED} = \frac{U_{IN} - U_{LED}}{I} = \frac{3,3\ \text{V} - 1,8\ \text{V}}{0,015\ \text{A}} = 100\ \Omega$$

Tou nejdůležitější částí prototypové desky je deset GPIO pinů vyvedených na svorkovnici. Pět z nich slouží jako vstupní, přičemž konkrétní piny byly zvoleny jednak s ohledem na jejich umístění v rámci DPS, ale i tak, aby bylo možné je připojit na AD převodník, snímat dotyk nebo využít druhý UART modul. Druhá pětice je posílena NPN tranzistory. Hodnota rezistoru R_a odpovídá pull-up rezistorům mikrokontroléru, tedy 45 kΩ. Hodnota rezistoru R_b se odvíjí od použitého tranzistoru a spínané zátěže. V příkladu je použit běžný NPN tranzistor `BC337`, který může spínat maximálně 0,5 A a jeho zesilovací činitel při

tomto proudu je 100. Nejdříve se určí potřebný bázový proud

$$I_B = \frac{I_C}{h_{FE}} \cdot 3 = \frac{0,5\ \text{A}}{100} \cdot 3 = 15\ mA$$

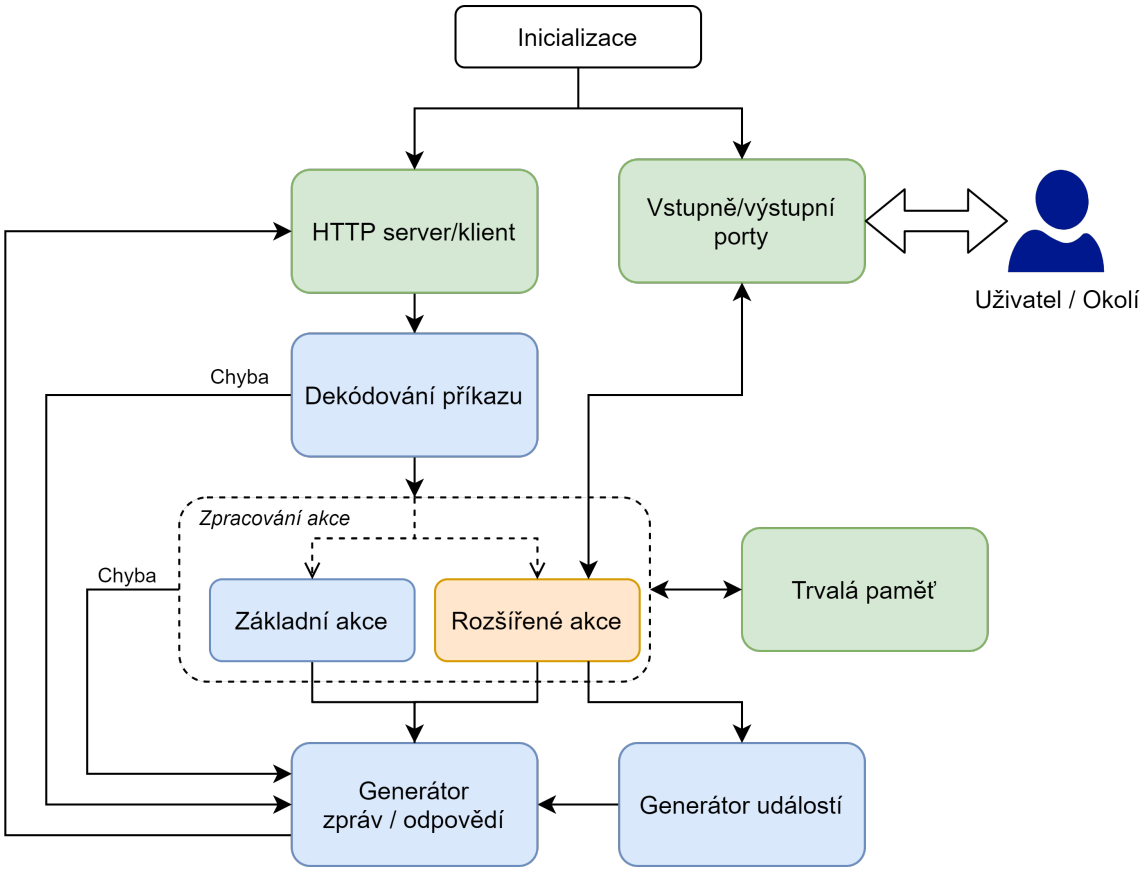
který musí být nižší než maximální zatížení jednoho pinu (40 mA), což je splněno. Na základě jeho hodnoty se následně vypočte i bázový rezistor R_b

$$R_b = \frac{U_{IN} - V_{BE}}{I_B} = \frac{3,3\ \text{V} - 1,2\ \text{V}}{0,015\ \text{A}} = 140\ \Omega$$

Toto zapojení je univerzální a může být použito k různým účelům pouhým nahráním nového programu. Pomocí něj bude možné snadno testovat vyvíjený program a jeho funkce. V další části práce pak budou navrhnuty i miniaturizované řešení pro konkrétní moduly.

4.2 Implementace modulu a jeho síťové komunikace

Jedním z cílů při návrh řešení je snadné přidávání dalších modulů a portace na jiný hardware. To mimo jiné vyžaduje důsledné oddělení jednotlivých části kódu do samostatných logických celků. Schéma 4.4 ilustruje rozdělení modulu na jednotlivé celky a jejich vzájemné vazby. Barva celku pak vyjadřuje závislost na konkrétním hardwaru, popř. modulu.



Obrázek 4.4: Podrobnější schéma základní programové smyčky modulu.

Modré bloky jsou totožné pro všechny moduly bez ohledu na konkrétní hardware či funkce modulu

Oranžový blok přináší další schopnosti modulu. Proto také vyžaduje přístup k GPIO portům daného MCU.

Zelené bloky jsou implementačně závislé na konkrétním hardware. Zároveň obstarávají komunikaci s vnějším světem.

Z obrázku 4.4 je patrné, že základní modul (modré bloky) negeneruje žádné události ani neumí přistupovat ke vstupně/výstupním portům. Tyto klíčové vlastnosti definují až jednotlivé moduly (oranžový blok). Naopak při vytváření nových modulů odpadá nutnost řešit komunikační protokol, trvalou paměť apod.

4.2.1 MCU implementace

Je patrné, že větší část kódu slouží ke zpracování a generování zpráv (tedy implementaci komunikačního protokolu) a je společná pro všechny moduly. Vytvoření nového modulu pak spočívá pouze v definici a implementaci dodatečných akcí. Použitý jazyk `C++` nabízí řešení v podobě tříd, dědičnosti a polymorfismu. V rámci kódu je definována mateřská třída `Module`, která implementuje všechny modré bloky a poskytuje základní sadu akcí. Jednotlivé moduly jsou pak potomky této třídy.

Implementace hardwarové části je podobná. Zde se jedná o třídy `Server` a `Io`. Třída `Server`, jak už je patrné z názvu, řeší zpracování a vytváření HTTP hlaviček a obecně síťovou komunikaci. Její potomek pak musí zajistit korektní přístup k síťovému hardwaru na dané platformě. Druhá třída `Io` pak poskytuje společný interface pro přístup ke vstupně/výstupním pinům a trvalé paměti.

Každý modul implementuje základní sadu akcí. Jejich kompletní přehled je v příloze B. Mezi nejdůležitější (kromě těch pro získání webové stránky a s tím souvisejících souborů) patří akce `set` a `get`. Ty slouží pro nastavění (popř. přečtení) hodnoty různých parametrů modulu (např. tak lze nastavit jméno modulu či jeho IP adresu). Třída `Module` poskytuje pomocné funkce pro rozšíření základní sady parametrů o další. Tak lze např. dodefinovat parametr *brightness*, jehož obslužná funkce pomocí metod třídy `Io` nastaví PWM signál na daném pinu a uloží tuto hodnotu do trvalé paměti (kvůli možnosti ji načíst po restartu zařízení). Na závěr vytvoří událost, na kterou může patřičně reagovat řídící jednotka.

Následující podrobnější popis tříd demonstruje jejich účel v rámci kódu a ilustruje, co je nutné udělat pro vytváření nového modulu.

Třída `Module`, jak už bylo zmíněno, řeší inicializaci modulu, čtení zpráv, obsluhu akcí, sestavování odpovědí a vytváření událostí. Objekt této třídy obsahuje seznam základních akcí společných pro všechny moduly v poli `mainActions`. Každá akce je reprezentována řetězcem obsahujícím její jméno a ukazatelem na funkci, která se má při jejím vyvolání spustit (*callback funkce*). Toto pole pak doplňuje druhé se jménem `additionalActions`, které vyplní potomek této třídy. Pomocí něj je možné doplnit do modulu další akce. Pro jednoduchou iteraci předanými parametry akce jsou zde makra `DATA_ITERATOR_START` (začátek bloku) a `DATA_ITERATOR_END` (ukončení bloku), která vytvoří dvě stringové proměnné – `param` se jménem parametrů a `val` s jeho hodnotou (může být i prázdná).

Akce `set` a `get` přistupujícím k různým vlastnostem modulu (*parametrům*) používají pro svůj běh pole `mainSets` a `mainGets`. Ty opět obsahují řetězec s názvem čteného

či zapisovaného parametru a ukazatel na obslužnou funkci. Stejně tak každý modul může seznam podporovaných parametrů rozšířit pomocí polí `additionalSets`, respektive `additionalGets`. Oddělení do dvou polí umožňuje mít rozdílnou sadu parametrů, které lze nastavit od těch, které lze číst.

Modul dále může vyvolat událost, popř. i více událostí v jeden moment. Ty se zapisují do bufferu a ve chvíli, kdy se předpokládá, že již další nebudou vznikat, tedy typicky ve chvíli, kdy je dokončeno zpracování akce, se odešlou řídící jednotce. Aby měla řídící jednotka přehled o tom, které události může modul vyvolat, mělo by být správně inicializováno pole `additionalEvents` s jejich seznamem. Základní modul žádné události nevyvolává, proto ani neexistuje pole `mainEvents`.

Konstruktor inicializuje pomocné buffery, tedy ten pro sestavení odpovědi (není tak nutné pro každou odpověď znovu alokovat paměť) a pro seznam vyvolaných událostí. Zároveň s tím načte z trvalé paměti potřebné údaje, jako vlastní jméno a port nebo IP adresu a port řídící jednotky. Obdobné chování se očekává i u konstruktoru potomků. Modul jako takový se pak spustí zavoláním metody `run`. Ta inicializuje HTTP server a odešle *hello* zprávu řídící jednotce. Následně aktivně čeká na příchozí komunikaci. Každá přijatá zpráva je zpracována a pokud je validní, tak i spuštěna daná akce. Během toho je postupně sestavena kompletní odpověď i s případnými daty, odeslána a uzavřeno spojení. Pokud byly během vykonání vyvolány nějaké události, jsou odeslány na řídící jednotku.

Třída `Server` obsahuje metody pro navázání a přijetí spojení. Protože implementace je závislá na cílovém operačním systému (popř. hardware, pokud není žádný systém použit), jedná se o čistě virtuální metody. Tedy definují rozhraní třídy, ale samotné nic neimplementují. Tyto metody vrací objekty třídy `Socket`, která se stará o jednotný přístup k síťovému rozhraní pro různé systémy. Třída `Socket` umožňuje příjem a odesílání zpráv. Kromě toho jsou v této třídě pomocné metody realizující HTTP protokol, zejména parsování a vytváření HTTP hlaviček. I zde je část metod čistě virtuální a musí být implementovány v potomcích této třídy.

Třída `IO` obsluhuje rozhraní mezi softwarem a hardwarem. Tedy abstrahuje přístup ke vstupně výstupním pinům, časovačům a k trvalé paměti. Při tvorbě modulu pak není nutné řešit technickou stránku věci. U pinů lze určit zda se jedná o vstupní, výstupní nebo na něj bude připojen PWM kanál. Na vstupní piny mohou být připojeny pull up a pull down rezistor nebo registrovány přerušení pro obsluhu stisknutí tlačítka či změnu stavu vypínače. Součástí obsluhy přerušení je i detekce dlouhého stisku a odstranění zákmitů. Výstupní piny s PWM kanálem pak vnitřně využívají knihovnu mikrokontroléru `ESP32` pro plynulou změnu jasu, přičemž toto chování lze potlačit.

Použitý mikrokontrolér obsahuje čtyři 64bitové časovače. V rámci této práce jsou použity jen dva, jeden pro jednorázové opožděné volání funkce. Jeho inicializace se provede pouhým zavoláním funkce `initDelayTimer`, které je jako parametr předána callback funkce a doba v sekundách, po jejímž uplynutí bude daná funkce spuštěna. Poté již stačí časovač spustit funkcí `delayTimerStart`, popř. jej lze také pozastavit, či zcela zastavit. Druhý časovač slouží pro opakované volání. Je inicializován celkovou dobu trvání akce a počtem volání dané funkce v tomto časovém úseku.

V rámci běhu na operačním systému UNIX je trvalá paměť simulována jako textový soubor, ve kterém jsou data uspořádána ve formě JSON. Výstupní piny jsou simulovány výstupem konzole, vstupní piny a časovače simulovány nejsou.

4.2.2 Komunikační protokol

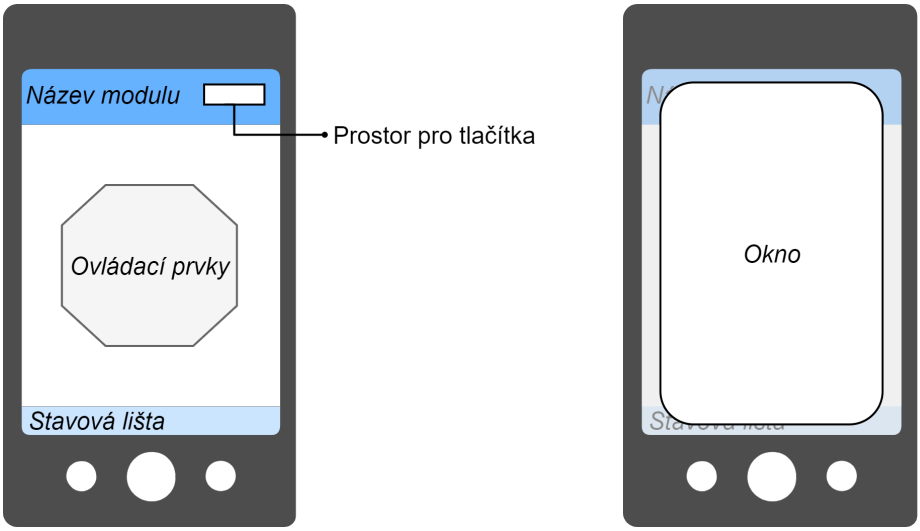
Komunikační protokol je postaven nad protokolem HTTP, konkrétně jeho verzí 1.1. Použití HTTP přináší několik výhod. Je použito TCP spojení, čímž je zajištěna spolehlivost komunikace. Není nutné programovat implementaci uživatelské aplikace, protože se použije internetový prohlížeč, který má HTTP už implementované. Jedná se o jednoduchý textově založený protokol, takže je snadno uživatelsky čitelný (výhoda pro vývoj) a v tomto konkrétním případě postačuje implementovat pouze jeho podmnožinu – příkaz `GET`.

Protokol funguje na principu dotaz – odpověď. Spojení vždy navazuje prvek, který chce zaslat dotaz (dotazem se v tomto případě myslí i příkaz). Otevře TCP spojení (typicky na portu 80, který je výchozím pro HTTP protokol) a sestaví HTTP dotaz. Dotazovaný prvek zpracuje příchozí komunikaci, vyhodnotí ji, zašle patřičnou odpověď a ukončí spojení. V této jednoduché implementaci není využita možnost zaslání více dotazů jedním spojením.

Základem komunikace jsou tzv. akce. Ty jsou kódovány přímo do URL. Např. seznam všech akcí podporovaných modulem na adrese 192.168.0.11 (s implicitním portem 80) je dostupný na adrese *http://192.168.0.11/actionList*. Na každý takový příkaz je potřeba patřičně odpovědět. V HTTP k tomu slouží stavové kódy. Konkrétně chyby jsou značeny čísly 4xx. Naopak úspěch používá řadu 2xx. Zajímavé jsou především `204 No Content` pro úspěšné zpracování akce bez potřeby odeslat nějaká data jako odpověď a `200 OK` pro odpověď s daty, která jsou vložena do těla zprávy ve formě JSON.

4.3 Návrh a realizace uživatelského rozhraní

Lze předpokládat, že většina akcí bude vyvolána pomocí pravidel v řídící jednotce. Sekundárním ovládacím prvkem pak bude mobilní telefon. Proto je důležité webovou stránku přizpůsobit i pro malé dotykové obrazovky. Tedy menší množství větších ovládacích prvků. Samozřejmě se nesmí opomenout vykreslení na klasické velké obrazovce. Aby se zabránilo zahlcení modulu, musí být uživatel informován o probíhající komunikaci s modulem a mělo by mu být znemožněno vyvolat další akci, než se dokončí ta předchozí.

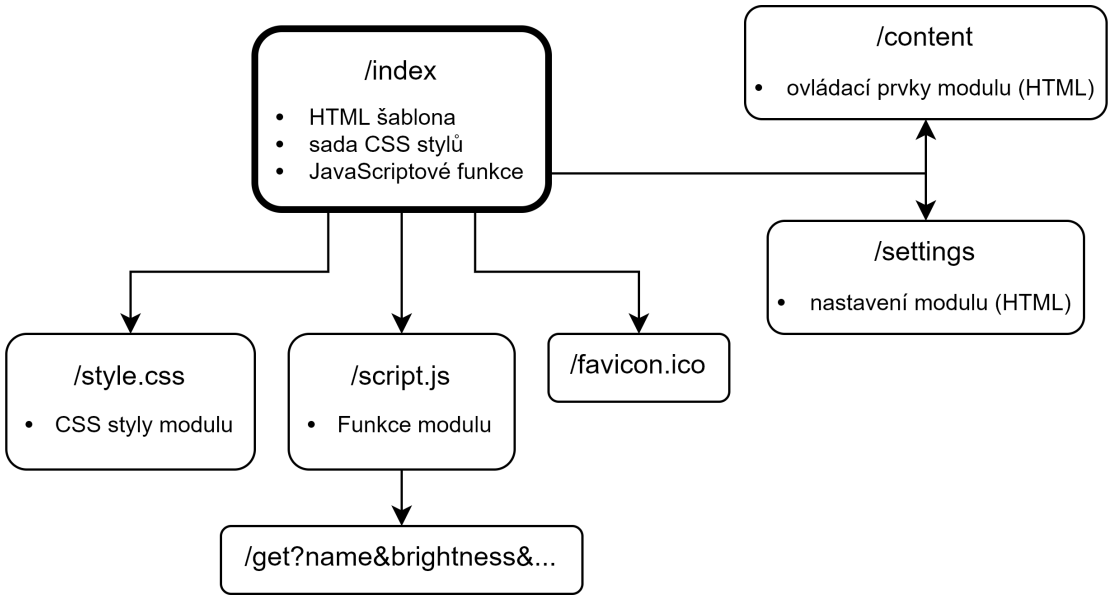


Obrázek 4.5: Návrh vzhledu webové stránky.

Sice je každý modul tvořen jako samostatně ovladatelný, jistě by všechny moduly měly sdílet jednotný design. Proto byla navržena jednotná šablona webové stránky. Její návrh je zakreslen na obrázku 4.5. Konkrétně jsou zde znázorněny dva různé stavy. Vlevo se nachází výchozí rozložení, vpravo pak vizualizace otevření dialogového okna (např. s nastavením modulu). Protože webové rozhraní je poměrně jednoduché, bude vhodné zvolit moderní přístup a aplikovat jednostránkový design. To znamená, že uživatel se fakticky nachází stále na jedné stránce, která se mu překresluje, aniž by ji opustil. Příkladem může být ono dialogové okno z obrázku 4.5.

4.3.1 Členění souborů a jejich integrace do kódu aplikace

Pro jednodušší vývoj se dělí je webové stránky do více souborů. Přičemž je vhodné oddělit od sebe nejen sémantický popis stránky (HTML kostra), její vzhled (CSS styly) a chování (JavaScriptové funkce), ale i jednotlivé logické celky. Stejný postup byl zvolen při realizaci uživatelského rozhraní této práce. Toto rozdělení včetně popisu, který soubor se stará o načtení dalšího se nachází na obrázku 4.6. Podrobnější vysvětlení jednotlivých částí obrázku se nachází v následujících odstavcích.



Obrázek 4.6: Jednotlivé bloky webové stránky. Zvýrazněná část je základní soubor dodaný modulem. Ten definuje jednotlivé závislosti a stará se korektní vykreslení celé stránky.

Jednotlivé soubory se spojí do jednoho během kompilace a vloží se do kódu aplikace jako konstantní proměnná. To umožňuje běh i na systémech bez podpory souborů a zároveň to snižuje počet HTTP dotazů na modul. Aby bylo možné tento postup automatizovat, je nutné stanovit několik pravidel, zejména pojmenování a umístění patřičných souborů a implementovat jednoduchý preprocesor, který se postará o složení více souborů do jednoho. Podrobnosti jsou popsány v příloze C.2.

Protože moduly založené na mikrokontroléru ESP32 běží na jednom vlákně, dokážou v jednu chvíli odpovídat pouze na jeden dotaz. Takže byť moderní webové prohlížeče snižují dobu potřebnou k načtením vyšším počtem souběžných dotazů, tak v tomto konkrétním případě by to bylo kontraproduktivní. Spojení všeho do jednoho souboru tak fakticky snižuje

dobu odezvy. Mimo to se i šetří výpočetní výkon MCU, protože není potřeba webovou stránku postupně skládat ve své paměti, nýbrž je okamžitě připravená k odeslání.

HTML soubory

HTML ve verzi 5 se snaží oprostit od toho, jak budou data vykreslena (to přenechává CSS) a místo toho pouze popisuje význam jednotlivých položek. Základem všech modulů je společná HTML kostra. V blokovém schématu 4.6 se jedná o zvýrazněný blok dostupný na adrese `/index` nebo zkráceně přímo na `/`. V rámci této kostry je definováno základní rozložení stránky – hlavička, tělo a patička. V hlavičce jsou uvedeny závislosti na zbylých souborech. Webový prohlížeč se tak automaticky postará o načtení CSS stylů (`/style.css`), JavaScriptových funkcí (`/script.js`) a ikony web stránky (`/favicon.ico`).

Každý modul pak musí obsahovat tři další soubory, byť mohou být prázdné. První se vkládá na konec hlavičky a může tak snadno přidat závislosti na dalších souborech, ale i jiná metadata (například titulek stránky). Druhým důležitým souborem je část *Ovládací prvky* z obrázku 4.5. Každý modul má vlastní ovládací prvky, proto musí být snadno definovatelné mimo šablonu. Oddělení této části je také důležité pro správnou funkčnost řídící jednotky, viz 4.3.2. Posledním neméně důležitým souborem je specifikace nastavení. I to je totiž pro různé moduly různé. V rámci tohoto souboru je pouze definováno, co může uživatel nastavit. O samotné vykreslení vstupních prvků se stará JavaScriptová třída `Settings` (viz níže). Podrobnosti k jejímu použití jsou uvedeny přímo v jejím zdrojovém souboru.

CSS soubory

CSS definuje styl jednotlivých prvků. V tomto konkrétním případě specifikuje vzhled základního rozložení (šířka okna, výška hlavičky apod.) a společných prvků (tlačítka, dialogové okno atd.). Všechna CSS pravidla jsou při kompilaci vložena do hlavičky HTML šablony a jsou tak automaticky dostupná pro všechny moduly. Každý modul pak může doplnit vlastní sadu CSS pravidel. Ty jsou umístěny v externím souboru `style.css` (je dostupný na adrese `/style.css`), aby mohly být v případě potřeby načteny řídící jednotkou, viz 4.3.2. Mimo to může modul do hlavičky šablony vložit sadu před-připravených stylů, např. styly pro ovládací prvek žárovky či tlačítka.

JavaScriptové soubory

JavaScript je jednou z technologií, která umožňuje oživit statické webové stránky. Může sloužit k vytváření animací, vykreslování oken, ale i ke generování samotného obsahu. Každý modul využívá hned několik takových souborů, ve kterých je vytvořena sada tříd a pomocných funkcí, které umožňují fungování webové stránky a usnadňují přidávání nových funkcí či celých modulů. Všechny tyto funkce, třídy, proměnné či konstanty jsou obdobně jako CSS pravidla vloženy přímo do HTML hlavičky šablony. Doplňuje je soubor `script.js` (dostupný na stejnojmenné adrese), obsahující funkce specifické pro daný modul.

Z obecných tříd se jedná především o třídu `Ajax`, která slouží k asynchronnímu volání akcí modulu. Třída dotazy řadí do fronty, aby nedošlo k zahlení modulu. Zároveň umožňuje automaticky uzamknout uživatelské prostředí, dokud nedorazí odpověď. Zpracování odpovědí může probíhat dvěma různými způsoby. Buď je registrována funkce, která se má po daném dotazu spustit, nebo je dotaz volán přímo i s callback funkcí. První přístup je vhodný zejména pro situaci, kdy může být daná akce modulu vyvolána v různých částech

kódu a její odpověď je přitom zpracována vždy stejně. Naopak druhý přístup je ideální k jednorázovému spuštění nějaké akce.

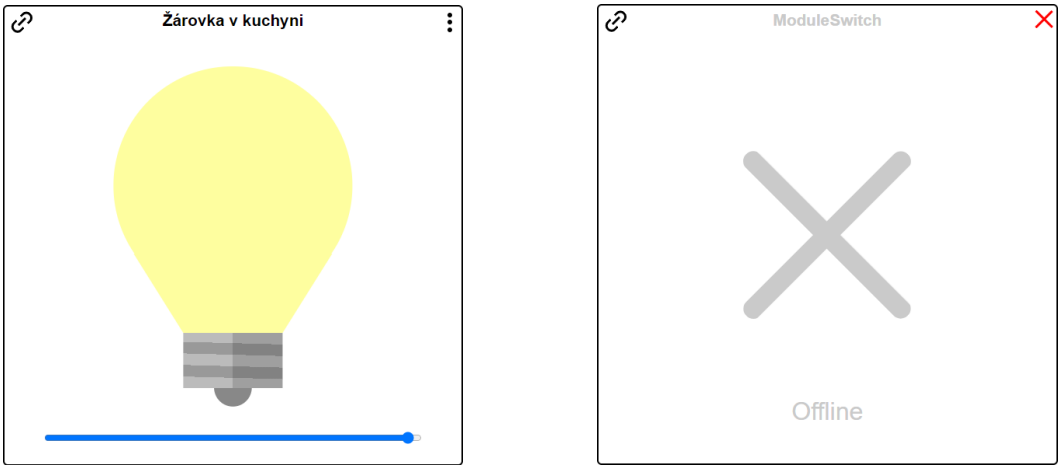
Třída `Settings` slouží ke generování obsahu okna s nastavením modulu (v rámci HTML je pouze popsáno co bude možné nastavit). K tomu využívá třídu `Window`, která obstarává vykreslení okna. Uživatelské rozhraní může obsahovat více oken, ale v jednu chvíli může být vykresleno pouze jedno. O to se právě stará tato třída.

Další třídy jsou platné pouze pro některé moduly. Například třída `Bulb`, která se stará o vykreslení tlačítka ve tvaru žárovky a ovládacích prvků pro jas a barvu (pokud se jedná o vícebarevnou žárovku). Třída realizuje i obsluhu akcí uživatele (např. změnu jasu) a zasílá odpovídající příkazy na modul skrze objekt třídy `Ajax`. Druhým zástupcem je třída `Button` pro tlačítka.

Tyto třídy doplňuje sada pomocných funkcí. Ty např. zjednodušují vytváření nových prvků stránky, obsluhu proměnných, které nemusí být definovány, zobrazení vyskakovacího okna apod. Na závěr je nutno ještě zmínit funkci `main`, která je spuštěna po načtení stránky. Ta vytvoří objekty výše zmíněných tříd a vykoná funkce, které si jednotlivé moduly zaregistrovali.

4.3.2 Řídící jednotka

Uživatelské rozhraní řídící jednotky je do jisté míry stejné, jako u každého jiného modulu. Používá stejnou šablonu i stejné postupy pro vytvoření stránky. Hlavní rozdíl je v ovládacích prvcích, kdy řídící jednotka jako taková žádný vlastní ovládací prvek nemá. Místo toho na jednom místě sdružuje ovládací prvky všech registrovaných modulů. Proto je nutné, aby každý modul důsledně oddělil společná data od těch specifických. To je důvodem, proč je část `content` z diagramu 4.6 načtena až za běhu a není zahrnuta během kompilace. Řídící jednotce totiž postačí pro každý modul načíst právě tento soubor a s ním pak ještě jeho styl (`style.css`) a JavaScript (`script.js`). Naopak není nutné načítat šablonu, jelikož ta je včetně stylů a scriptů pro všechny moduly společná.



Obrázek 4.7: Ukázka dlaždice žárovky a vpravo modulu, který neodpověděl na dotaz a je tedy nejspíš *offline*.

Každý modul je reprezentován tzv. *dlaždicí*. Ta je tvořena obalujícím rámečkem s názvem a doplněna o další ovládací prvky – tlačítko nastavení, odkaz na stránku modulu a tlačítko pro odstranění modulu(u online modulů je přesunuto do nastavení). Každá z dlaždic

používá vlastní objekt třídy `Ajax`, tedy vyvolání komunikace blokuje pouze danou dlaždici. Pořadí dlaždic si může uživatel změnit, tato změna je uložena do paměti řídící jednotky.

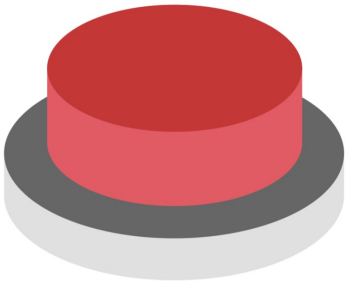
Samozřejmostí je responzivní design, kdy se počet dlaždic na jeden řádek plynule přizpůsobuje šířce stránky. Tedy na mobilním telefonu se dlaždice zobrazí nad sebou, zatímco na tabletu se vedle sebe vykreslí dvě či tři (v závislosti na rozlišení) a na počítači to může být třeba i pět.

4.3.3 Využití CSS grafiky

Žádná uživatelsky přívětivá webová stránka se neobejde bez obrázků a ikon. Jejich přiměřené použití má pozitivní vliv na přehlednost a uživatelský zážitek. Nejjednodušší možností jsou klasické rastrové obrázky. Ovšem každý takový obrázek znamená HTTP dotaz na modul, kde navíc zabírá cenné místo v paměti. Proto převládá snaha uchovávat ikony ve vektorové podobě. Oblíbeným přístupem je použití speciálního fontu, který místo písmen a číslic obsahuje ikony. Použití je jednoduché, na server je nutné umístit jen jeden soubor s fontem. Ten navíc může obsahovat pouze reálně využité symboly, tudíž nebude ani příliš velký. Ovšem jedná se o další soubor, na který je nutné se dotázat a který je potřeba přenést z modulu k uživateli.

```
<div class="button">
  <div class="head">
  </div>
  <div class="base">
  </div>
</div>
```

```
.button .base,
.button .base:after {
    width: 40vh;
    border-radius: 20vh/10vh;
}
.button .base {
    background: #e0e0e0;
    height: 24vh;
}
.button .base:after {
    background: #666666;
    height: 20vh;
    border-radius: 20vh/10vh;
}
```

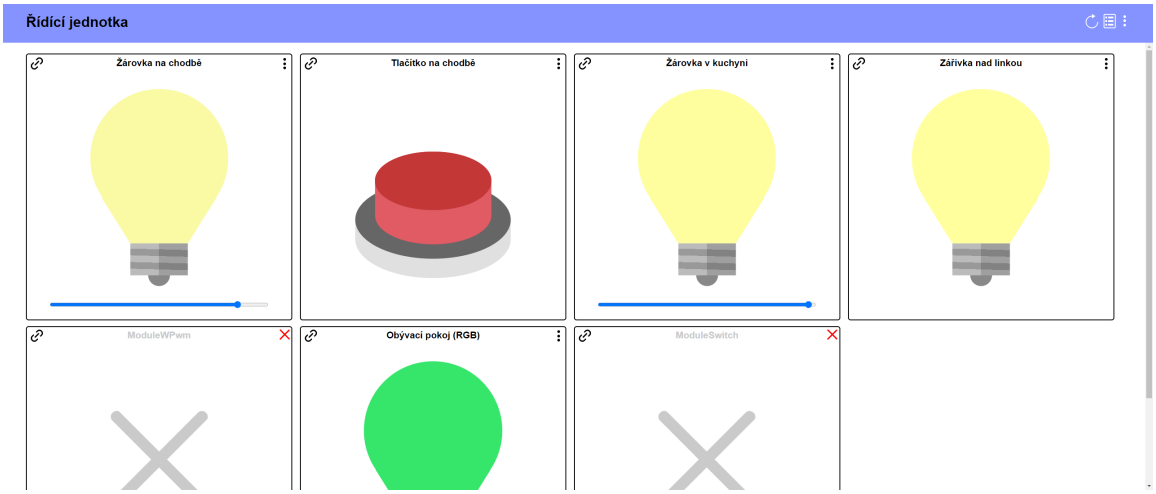


Obrázek 4.8: Příklad tlačítka kresleného pomocí CSS. *Vlevo* jsou potřebné HTML tagy, *uprostřed* ukázka CSS pravidel a *vpravo* pak výsledná grafika.

Druhou zajímavou alternativou je potřebnou grafiku vykreslit pomocí CSS. To ve své verzi 3 přináší velkou škálu možností stylování jednotlivých prvků. A právě pomocí kombinace stylů prvků, jejich rámečků, stínů apod. včetně případné rotace a posunutí umožňuje vykreslit i poměrně složitou grafiku. V rámci této práce byl použit právě tento přístup a veškeré ikony ovládací prvky jsou tvořeny pouhou kombinací CSS pravidel.

4.4 Realizace řídící jednotky

Řídící jednotka je důležitou součástí systému ovládání osvětlení. Sdružuje ovládání všech modulů na jednom místě a zároveň přidává možnost vytvářet pravidla a automatizovat tak celé řešení. Aby bylo možné toho dosáhnout, je potřeba vyřešit správu modulů, tedy jejich přidání a odebrání ze seznamu, či případnou aktualizaci konfigurace. Stejně tak je nutné vyřešit správu pravidel a jejich vyhodnocování.



Obrázek 4.9: Snímek obrazovky uživatelského rozhraní řídící jednotky.

4.4.1 Správa modulů

Každý modul se musí nejdříve zaregistrovat. O to se stará automaticky, kdy při každém spuštění odešle `hello` zprávu na adresu řídící jednotky. Ve výchozím nastavení se tato zpráva odešle jako *broadcast*, tudíž uživatel nemusí nic konfigurovat. Ovšem lze také specifikovat pevnou IP adresu a port řídící jednotky. *Hello* zpráva obsahuje typ modulu, port, na kterém naslouchá, jedinečné ID a nastavené jméno. Jméno jako takové sice není nezbytně nutné, ale pro uživatele je jistě přehlednější vidět jméno modulu, než jeho ID (to se ostatně zobrazuje za jménem). Řídící jednotka na základě ID zjistí, zda se jedná o nový nebo stávající modul. Nové moduly musí uživatel nejdříve schválit. Jedná se o bezpečnostní prvek, který zabraňuje snadnému proniknutí do systému.

Kromě hello zprávy poskytuje řídící jednotka sadu příkazů pro práci jak s registrovanými moduly (změna pořadí a odstranění), tak i s těmi, co teprve čekají na schválení (schválení a odstranění). Seznam všech příkazů je v příloze B.2.1.

4.4.2 Ovládání modulů

K ovládání modulů by se dalo přistupovat dvěma různými způsoby. V prvním případě by řídící jednotka sloužila jako prostředník komunikace, kdy by všechny příkazy od uživatele směřovaly na ni. Tedy by řídící jednotka uživatelskému rozhraní poskytovala jednotné API pro všechny moduly. Tyto příkazy by pak byly přeloženy na příkazy konkrétního modulu a odeslány na něj. Výhodou je ono jednotné API a s tím např. spojená jednodušší tvorba mobilní aplikace. Nevýhodou pak nutnost všechny příkazy překládat a vyšší latence způsobená komunikací skrze prostředníka. Navíc by nebylo možné připojit do systému nový modul, jehož typ řídící jednotka nezná, protože by neuměla přeložit příkazy od uživatele na API modulu.

V rámci této práce byl použit druhý přístup, kdy řídící jednotka pouze poskytuje seznam modulů. V rámci webového rozhraní řídící jednotky jsou uživateli načtena všechna uživatelská rozhraní modulů. Jednotlivé příkazy jsou pak odeslány přímo na daný modul. Hlavní výhodou tohoto přístupu je především jednoduchost a univerzálnost. Řídící jednotka vůbec nemusí implementovat vlastní API. Stejně tak v jejím kódu nemusí být seznam podporovaných modulů a jejich akcí.

4.4.3 Správa a vyhodnocování pravidel

Možnost vytvářet pravidla a automatizovat tak chování je důležitou vlastností chytrého osvětlení domácnosti. Při realizaci bylo použito řešení pomocí *událostí*. Každý modul může vyvolat jednu (nebo i více) událost, kterou zašle na řídící jednotku. Ta na základě původce (ID modulu) a typu události zjistí, zda existuje pravidlo, kterého se událost týká. Pokud ano, zašle daný příkaz na cílový modul. Každé pravidlo je tedy tvořeno čtveřicí informací. Nabízí se otázka, jak je uspořádat v paměti. Pouhé pole struktur by bylo implementačně jednoduché, ale vyhledávání pravidla by znamenalo vždy projít celé pole. Pro zvýšení efektivity je vhodné pole indexovat jedním z prvků definující pravidlo. K dispozici je ID modulu který událost vyvolal a typ události – tedy informace, které dostane řídící jednotka. Protože ID modulů se může pohybovat ve velkém rozsahu hodnot, bylo by nutné jej nějakým způsobem normalizovat. Naproti tomu všechny typy událostí jsou jasně definovány pomocí výčtového typu a mají tak kromě řetězcové interpretace i číselnou.

Indexem do pole pravidel je tedy typ události. Pro jednu událost může samozřejmě existovat více pravidel, tedy je použito pole ukazatelů na pole struktur. Podle typu události je zvoleno odpovídající pole struktur, které je následně sekvenčně prohledáno a všechna odpovídající pravidla jsou vykonána.

Událost kromě svého typu může nést i hodnotu. Tu nastavuje modul a pravidlo na ni může nějakým způsobem reagovat. Kromě prostého porovnání na rovnost bylo implementováno i celočíselné porovnání na nerovnost, tedy zda je hodnota větší či menší než zadaná. Příkladem je změna jasu žárovky, která vyvolá událost `lightBrightness` a jako hodnota je odeslán nový jas. Uživatel si tak může vytvořit pravidlo, které se vyvolá pouze když jas žárovky stoupne nad danou hodnotu.

Události zároveň nabízí snadnou integraci s jinými systémy. K tomu slouží typ události `other`. V kombinaci s neurčeným původcem tak lze reagovat na externí vstupy. Jedinou podmínkou je, že daný systém umí vyvolat konkrétní HTTP požadavek. Obdobně by bylo možné události využít k rozšíření funkčnosti řídící jednotky. Ta například neobsahuje časovač, ale má připravené události `timeEventStart` pro vyvolání a `timeEventEnd` pro ukončení časově závislé události. Pro pokročilého uživatele nebude problém si napsat jednoduchý skript (např. v pythonu), který v zadanou dobu požádá o následující webovou stránku:

```
http://adresaRidiciJednotky/event?id=0&timeEventStart
```

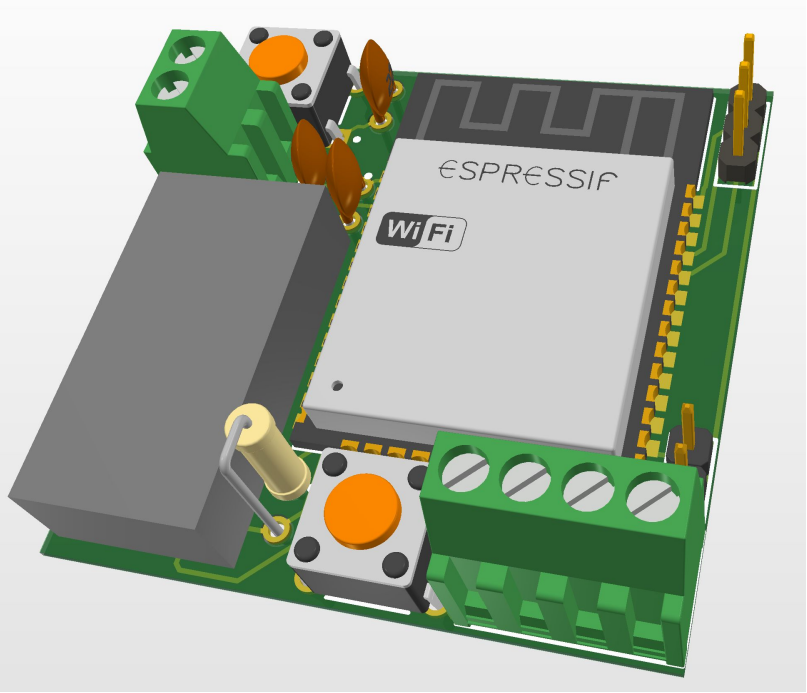
Tím se v řídící jednotce spustí pravidla, která nemají uvedený zdroj (proto `id=0`) a reagují na událost `timeEventStart`.

4.5 Realizace několika konkrétních modulů

Součástí této práce je i realizace několika modulů osvětlení. Každý modul se skládá z desky plošných spojů, softwaru pro mikrokontrolér a webového rozhraní. V průběhu práce bylo realizováno pět základních a tři rozšířené moduly. Ty demonstrují schopnosti počítačového řízení ovládání a variabilitu výsledného řešení.

4.5.1 Základní moduly

Byly navrženy tři moduly pro ovládání osvětlení. A to pro nestmívatelné světlo, stmívatelné jednobarevné světlo a RGB světlo. Všechny změny jasu a barvy probíhají plynule a nedochází tak k nepříjemnému blikání. Tyto moduly slouží jako demonstrace použití třídy



Obrázek 4.10: 3D vizualizace DPS pro zapojení se třemi vstupy

`IO` jí implementované obsluhy výstupních pinů a PWM kanálů. Každá změna vyvolá odpovídající událost (změna jasu, vypnutí apod.), která je odeslána na řídící jednotku a může sloužit jako spouštěč některého z pravidel. Mimo to může uživatel definovat výchozí stav (jas a barva) po zapnutí, popř. nechá modul obnovit stav před vypnutím.

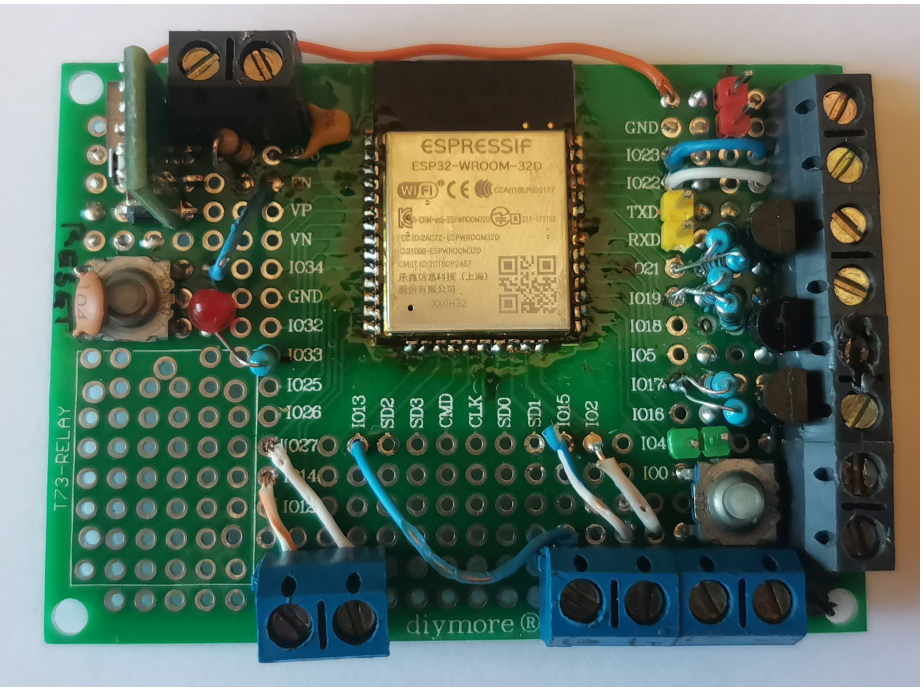
Schéma zapojení a deska plošných spojů je uvedena v příloze A.2. Zapojení vychází z prototypové desky, tedy používá stejné piny, aby nebylo nutné v kódu nic měnit. Byly odebrány nevyužité součástky a zmenšeny rozměry desky. Další miniaturizace by byla možná použitím druhé vrstvy a SMD pouzder součástek.

Výstupní moduly doplňují dva vstupní moduly. Oba používají totožnou desku, na které obsluhují jeden vstup, ale deska samotná byla navržena pro snímání až tří různých vstupů. Na ten může být připojeno buď tlačítko (po stisknutí se automaticky vrací do původní polohy) nebo přepínač (zůstává v dané poloze), kteří uživatelé znají také jako nástěnný vypínač. Obsluha signálů ze vstupu je řešena pomocí přerušení, včetně odstranění zákmitů bez nutnosti použití externích součástek. Obdobně jako moduly osvětlení, i toto zapojení vychází z prototypové desky a je uvedeno v příloze A.3.

4.5.2 Rozšířené moduly

Mnou navržené řešení umožňuje snadné rozšíření základních modulů různými způsoby, které mohou uživateli přinést zvýšený komfort. K tomu slouží třída `IO` popsána v podkapitole 4.2.1. Byly implementovány tři ukázky využití těchto možností. Konkrétně se jedná o světlo s odloženým zapnutím (popř. vypnutím), kombinaci světla s tlačítkem a vypínač s kontrolkou.

Na první pohled se jedná o prvky běžně se vyskytující i v tzv. hloupém osvětlení. Ovšem jejich chování bylo oproti nim vylepšeno. Rozšířením těchto modulů, popř. jejich kombinací s těmi základními si může pokročilý uživatel snadno vytvořit osvětlení zcela dle



Obrázek 4.11: Realizace prototypové desky. Oproti zapojené A zde chybí indikační LED `D1` a výstupy `OUT1` a `OUT2` nejsou posíleny tranzistory.

svých představ. Pro tyto moduly nebyla navržena speciální deska plošných spojů (úspora rozměrů by byla minimální), místo ní lze použít prototypovou desku.

Světlo s odloženým zapnutím a vypnutím

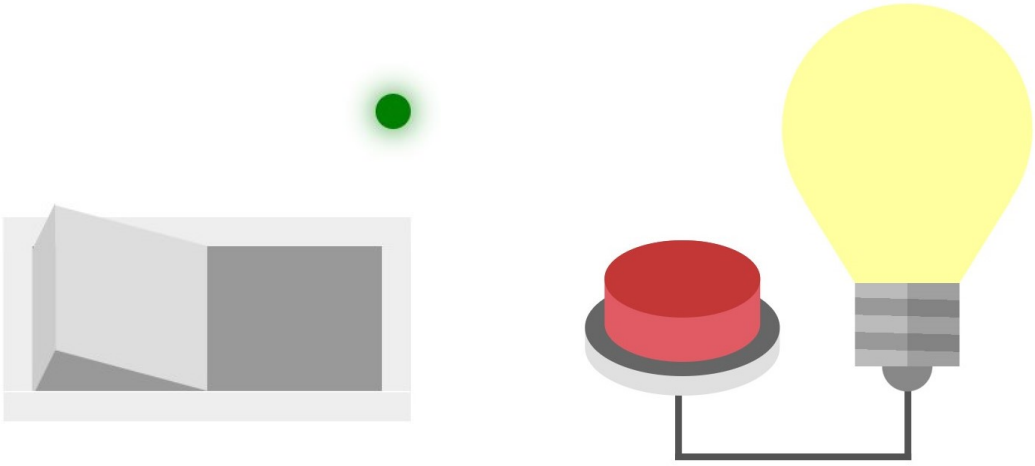
U tohoto modulu si může uživatel určit zpoždění, než dojde k vykonání zaslané akce. Na rozdíl od běžně prodávaných analogových časovačů je čas definován s přesností na sekundy, což je pouze implementační, nikoli technologické omezení. Mimo to může určit, zda bude tato změna skoková nebo plynulá. První zmíněná varianta najde využití například k osvícení prostoru před vchodovými dveřmi. Uživatel si při odchodu rozsvítí, zamkne, schová klíče a světlo se po dané době automaticky zhasne. Naproti tomu plynulá změna může zpříjemnit ranní vstávání. Grafické rozhraní je stejné, jako u běžného světla. Pouze byla doplněna indikace, zda běží časovač. Ten je možné zastavit a danou akci vykonat okamžitě.

Implementačně se jedná o použití časovačů mikrokontroléru. Ten je třídou `IO` správně inicializován a v reakci na akci uživatele spuštěn. Po uplynutí dané doby časovač vyvolá přerušení, které obslouží definovaná funkce, tedy rozsvítí, zhasne nebo změní jas diody. V zapojení není třeba nic měnit a lze využít DPS navrženou pro základní modul jednobarevného světla.

Světlo s tlačítkem

Kombinace světla a tlačítka přináší uživateli nespornou výhodu. Toto řešení je odolné vůči nefunkčnosti Wi-Fi sítě. Při tom je zachována možnost dálkové ovládání i schopnost generovat události. Příkladem použití může být lampička na nočním stolku. Krátké stisknutí

tlačítka ovládá lampičku, dlouhé vyvolá událost, které spustí pravidlo ke zhasnutí stropní světla v ložnici.



Obrázek 4.12: Ovládací prvek vypínače s kontrolkou (*vlevo*) a světla s tlačítkem (*vpravo*).

Při použití prototypové desky se očekává připojení tlačítka na vstup `IN1` a LED na výstup `OUT1`, popř. je nutné provést odpovídající změnu v kódu. Na vstupní pin tlačítka je připojen pull up rezistor, tedy musí tlačítko spínat zem.

Vypínač s indikační LED

Posledním příkladem rozšířeného modulu je vypínač s dvoubarevnou indikační LED. Konkrétně jsou uvažovány barvy zelená a červená, ale jejich souběžné rozsvícení vytvoří iluzi žlutooranžové. Barva diody je nezávislá na stavu tlačítka, naopak je ovládána pouze skrze síť a může tak reagovat na stav osvětlení. Závisí pouze na uživateli, jaká pravidla si vytvoří. Zelená může značit stav, kdy je světlo zhasnuto (obdobně jako doutnavka v dnešních vypínačích), žlutá situaci, kdy světlo s odloženým zapnutím spustilo svůj odpočet a červená svítící světlo. Vypínač je očekáván na vstupu `IN1` ke kterému je připojen pull up rezistor, tudíž musí spínat zem. Výstupní diody jsou očekávány na výstupech `OUT1` (zelená) a `OUT2` (červená). Není nezbytně nutné použití tranzistorů, jen je třeba nezapomenout na ochranný rezistor. Zapojení pak může být totožné s indikační diodou `D2`.

4.6 Testování funkčnosti a splnění požadavků

Testování je nedílnou součástí vývoje jakéhokoli zařízení i softwaru. Je nutné definovat co a jak má být testováno a také, jakých výsledků má být dosaženo. Tato práce se skládá z několika do určité míry samostatných celků (hardwarové zapojení, software MCU a webové rozhraní), které tudíž mohou být testovány samostatně. Následně je také nutné otestovat jejich vzájemnou integraci a funkčnost celku jako takového.

Každá z jednotlivých částí práce vyžaduje jiný přístup k testům. Je zřejmé, že u hardwarového zapojení je nutné ověřit zcela jiné hodnoty a vlastnosti než v případě webového rozhraní. Cílem testování není pouze potvrdit, že výsledek odpovídá specifikaci ze zadání, ale také odhalit případné nedokonalosti či chyby. Ty mohou mít negativní vliv na výsledný dojem z používání, ale také mohou způsobit i bezpečnostní rizika, kterým se věnuji dále.

Komunikace

Komunikace probíhá skrze Wi-Fi síť protokolem HTTP. Bylo ověřeno, že modul umí vysílat vlastní Wi-Fi síť, ke které se může uživatel připojit. Taktéž byla ověřena schopnost připojení k existující domácí síti, včetně situace, kdy uživatel zadá chybné údaje, popř. se připojení nezdaří. Modul se v tomto případě automaticky přepne do režimu přístupového bodu.

Komunikační protokol

Jeho implementace je totožná jak pro řídící jednotku tak pro moduly. To usnadňuje testování, kdy lze využít běžný počítač a všechny dostupné pokročilé nástroje, bez potřeby přímé interakce s mikrokontrolérem. Kromě zpracování validních zpráv byla ověřena i odolnost vůči chybným zprávám. Ať už neúplným nebo s chybnými datovými typy. Takové zprávy jsou ignorovány a odpovědí je HTTP návratový kód 400.

Grafické rozhraní

Interakce s uživatelem je řešena formou webové stránky. Z pohledu uživatele je GUI tou nejdůležitější částí, a to z toho důvodu, že je to jediná část, se kterou přichází do styku. Proto je nutné testování webového rozhraní věnovat patřičnou pozornost. Kromě možnosti ovládat a konfigurovat modul bylo nutné ověřit i to, zda může uživatel zadat nevalidní hodnoty, což by samozřejmě neměl, a jak se s nimi případně vyrovná nejen GUI, ale i modul. Dále bylo třeba ověřit chování, pokud dojde k výpadku internetového spojení s některým z modulů nebo i řídící jednotkou.

Uživatelské rozhraní se ukázalo jako velký zdroj chyb. Důvodem je použití JavaScriptu, jakožto interpretovaného jazyku. Případné chyby totiž nejsou odhaleny během kompilace (která vůbec neprobíhá), ale zpravidla až za běhu, kdy se interpret pokusí spustit neplatnou funkci nebo přistoupit k neinicializovanému atributu. Tento fakt ještě dále umocňuje dynamické načítání dat.

Kromě chování webového rozhraní byla ověřena i správnost vykreslování. K tomu posloužil běžný notebook s externím monitorem, na kterém se pomocí vývojářských nástrojů simulovalo rozlišení zařízení až do šířky 2 520 pixelů. Obdobně různé poměry stran. Testování na PC doplnilo ověření funkčnosti na mobilním telefonu. V obou případech byl použit webový prohlížeč založený na projektu *Chromium*.

V případě komerčního produktu by bylo nutné zajistit testování na reprezentativním vzorku uživatelů, kteří používají různé webové prohlížeče (případně jejich verze), operačními systémy a typy displejů či rozlišení obrazovek. Takto rozsáhlé testování je mimo rozsah a možnosti diplomové práce.

Řídící jednotka

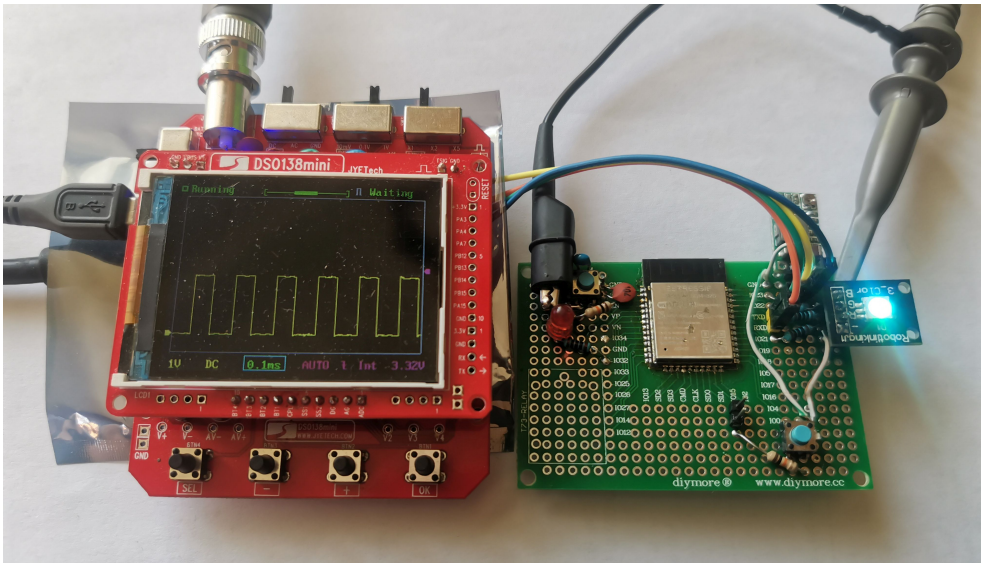
Důležitou částí této práce je řídící jednotka a jako taková vyžadovala důkladné testování. Byla testována správnost reakce na *hello* zprávu – neznámé moduly musí být přiřazeny do seznamu ke schválení, stávající pouze předají aktuální konfiguraci. Zde se ukázalo, že je teoreticky možné nabourat systém posláním falešné *hello* zprávy se stejným ID, jako má některý z již schválených modulů. Tento vektor útoku je rozebrán dále v podkapitole 4.7.

Správu modulů doplňuje správa pravidel. Řídící jednotka správně vyhodnotila různá pravidla. Problém nastal v případě, kruhové závislosti pravidel. Tedy pokud vykonání pravidla vedlo k zavolání akce, která toto pravidlo spouští. V tu chvíli došlo k zacyklení. Bylo

proto nutné vytvořit parametr `noEvent`, který zabrání vyvolání události. Pravidlo, které má být poslední pak musí přidat tento parametr k danému volání.

Moduly

Jednotlivé moduly byly vyvíjeny a testovány obdobně jako řídící jednotka primárně na počítači. Poté, co bylo dosaženo požadované funkčnosti, byl program zkompilován a nahrán na MCU, kde byla ověřena komunikace s hardwarem a funkčnost zapojení jako taková.



Obrázek 4.13: Průběh PWM signálu na displeji osciloskopu při testování zapojení na univerzální desce.

Testovaní hardwaru je poměrně složité a je mimo rozsah této diplomové práce. Základní chyby, jako např. zkrat, dokáže detekovat již software pro kreslení schémat. Obdobně tak i případné chyby v návrhu DPS (chybějící součástka, prohozené či nepřipojené piny aj.). Ovšem každý výrobek uvedený na evropský trh musí splňovat tzv. *elektromagnetickou kompatibilitu*, kterou lze ověřit pouze v laboratoři. Dále je třeba otestovat citlivost na výkyvy napájecího napětí, stabilitu při dlouhodobém běhu, běh v extrémních podmínkách apod. To vše vyžaduje pokročilé laboratorní přístroje.

4.7 Potenciální zranitelnosti modulů a řídící jednotky

Jako každé jiné zařízení připojené do sítě, tak i moduly a řídící jednotka představují určité riziko a jsou náchylné na rozličné útoky. V následujících odstavcích jsou rozebrány některé možné vektory útoku, včetně postupu, jak jim zabránit a výčtu případných škod, které mohou způsobit.

Zařízení mohou fungovat ve dvou režimech. Buď v omezeném režimu, kdy si vytváří vlastní Wi-Fi síť. Ten je sice primárně určen pouze k jejich konfiguraci, ale samozřejmě v určitých situacích může stejně tak sloužit jako hlavní režim. Druhým režimem je pak komunikace skrze již existující uživatelovu síť, což je preferovaná možnost umožňující vzájemnou komunikaci více modulů.

Nešifrovaná komunikace

Zařízení s uživatelem i řídící jednotkou komunikují nešifrovaně. Protože se předpokládá použití v domácí síti, která je zpravidla od okolního světa odstíněna routerem s NATem a firewallem, tak to nepředstavuje tak závažný problém. Pokročilejší uživatelé (kteří si spíše pořídí takovéto osvětlení) pak budou umět oddělit zařízení od zbytku sítě (ale i internetu, který pro tyto zařízení není potřeba) pomocí VLAN[1]. Nešifrování komunikace pak pro uživatele znamená, že nemusí řešit certifikáty, přičemž ostatní členové domácnosti sice mohou zjistit, že nějaká IP adresa rozsvítila světlo, ale to mohou i pouhým pohledem do vedlejšího pokoje.

Problém nastává v situaci, kdy by moduly běžely s veřejnou IP adresou nebo by byly jiným způsobem přístupné z internetu. Popř. kdyby se nejednalo o jednu domácnost, ale např. budovu s kancelářskými místnostmi, které by z nějakého důvodu využívaly jednu síť, což už je samo o sobě velké bezpečnostní riziko. Pak by už jistě bylo žádoucí skrýt komunikaci. Šifrování, tedy použití HTTPS, přináší možnost skrýt obsah komunikace před ostatními. Takže již nebude možné odposlechnout, zda uživatel pouze změnil jas, nebo zhasl světlo. Ale stále neřeší skrytí informace, že ke komunikaci vůbec došlo (to lze řešit jedině oddělenou sítí) ani nezabrání nežádoucímu ovládání modulu nepovolaným uživatelem.

Neautorizovaný přístup

Moduly ani řídící jednotka momentálně nevyžadují žádnou autorizaci, která by zabraňovala nežádoucí konfiguraci nebo ovládání. Toto řešení bylo zvoleno s ohledem na jednoduchost používání a cílení na domácnosti. Autorizace přístupu pak vyžadoval minimálně dvě úrovně uživatelů – *správce*, který by mohl provádět i konfiguraci a *uživatele*, kteří by mohli pouze ovládat modul. Tyto uživatelé by museli být nastaveni na každém modulu, popř. by řídící jednotka musela umět propagovat seznam uživatelů a jejich hesel na jednotlivé moduly. V kombinaci se šifrováním by pak bylo zcela bezpečné (pokud by realizace tohoto řešení neobsahovala nějaké chyby) vystavit modul do internetu.

Absence autentizace

Je nutné definovat rozdíl mezi autentizací a autorizací. Autorizace opravňuje uživatele k provedení akce, zatímco autentizace ověřuje, zda se za uživatele nevydává nikdo jiný. V tomto případě se nejedná o uživatele, ale o modul. Protože při zasílání příkazů není ověřen odesílatel, tak by se případný útočník mohl vydávat za jeden z modulů. Podmínkou je získání přístupu do sítě, viz níže.

Zranitelné jsou především *hello* zprávy, kdy by mohl útočník podvrhnout ID za jiné existujícího modulu a obejít tak proces schvalování, jelikož modul s daným ID je již schválen. Tím by přesměroval veškerou komunikaci tohoto modulu na sebe. Dokonce je tak možné změnit i typ modulu. Jediné řešení této zranitelnosti je podepisování *hello* zpráv.

Získání přístupu do sítě

Ve výchozím nastavení funguje modul v režimu vysílání sítě chráněné heslem s využitím šifrování WPA2. Pokud by bylo zařízení vyráběno sériově, tak by bylo heslo generováno náhodně. Tento přístup je v současnosti brán jako bezpečný. Sice již existuje novější standart WPA3, ale jeho podpora skrze různá zařízení zatím není dostatečně rozšířena. Modul lze

[1] Virtuální LAN, tedy logická podsíť oddělená od ostatních VLAN

samozřejmě přepnout do režimu, kdy k přihlášení do jím vysílané sítě není vyžadováno heslo, ale tuto konfigurací musí provést uživatel ručně, grafické rozhraní ji nenabízí. Pokud by bylo vypnuto přihlašování heslem, je útok velmi jednoduchý - stačí se přihlásit k této Wi-Fi síti. Pokud nebude zabezpečení vypnuto, musí být případný útok veden buď vůči šifrování WPA2 (jeho prolomení by znamenalo problém celosvětového rozsahu) nebo vůči jeho implementaci na mikrokontroléru `ESP32`.

Ovšem lze předpokládat, že drtivá většina modulů bude využívat již existující Wi-Fi síť. Variantu, kdy již má útočník přístup do této sítě, není nutné příliš rozvádět, protože to pro uživatele představuje mnohem větší riziko v jiných ohledech. Naopak možností, jak zneužít modul pro získání přístupu do sítě mnoho není. Modul neposkytuje akci, která by umožnila přečíst heslo do sítě z paměti. Pokud není v kódu nějaká chyba typu *buffer overflow*, tak by tato informace měla být bezpečně uložena ve FLASH paměti modulu.

Rizika útoku

V případě, že útočník získá přístup do sítě (bez ohledu na to, zda ji vysílá modul nebo uživatel) tak zároveň s tím získá plnou kontrolu nad modulem (případně řídící jednotkou). Tedy jej může libovolně ovládat, pokud vysílá vlastní Wi-Fi síť, tak změnou hesla zcela unést, ale také jen pasivně sledovat jeho komunikaci. Aktivní útok jako nevyžádané ovládání (např. rozsvícení na plný výkon v půlce noci) či změna hesla je sice otravné, ale uživatelem velmi brzo odhalitelné. Řešením je pak restart zařízení, přičemž záleží, zda bylo slabinou jednoduché (či žádné) heslo nebo jiná zranitelnost. Podle toho by buď postačilo použít silnější heslo nebo by bylo nutné vyřešit danou zranitelnost. V krajním případě bohužel zařízení nepoužívat.

Nebezpečnějším útokem je pouhé pasivní sledování komunikace. V případě, že se jedná o Wi-Fi šířenou modulem, bylo by možné odposlechnout heslo domácí Wi-Fi sítě během prvotní konfigurace modulu. Mimo to lze informace o stavu jednotlivých světel (případně pomocí sledování změn) určit, zda je někdo doma, popř. typický čas odchodu do práce apod. Tato informace může být užitečná např. pro zloděje. Na druhou stranu, přístup k těmto informacím je podmíněn přístupem do sítě. A pokud má záškodník přístup do domácí sítě, může za pomoci aktivity jednotlivých zařízení získat ty stejné informace. Naopak i mnohé další.

Ochrana před útokem

Jak už bylo nastíněno výše, moduly neumožňují moc způsobů, jak je zneužít. Tedy kromě nežádoucích programátorských chyb typu *buffer overflow* a používání modulů na nezabezpečené Wi-Fi. Pro zvýšení bezpečnosti by bylo možné pro komunikaci použít zabezpečené HTTPS místo použitého HTTP. Mikrokontrolér obsahuje hardwarovou podporu pro šifrování, takže by jedinou překážkou bylo nasazení certifikátů. Druhým krokem ke zvýšení bezpečnosti by byla povinná autorizace pro konfiguraci, případně i pro ovládání zařízení. S tím pak souvisí autentizace modulů při komunikaci s řídící jednotkou. Podmínkou pro nasazení těchto technologií je přítomnost HTTPS. V opačném případě by se heslo od uživatele či podpis modulu přenášely nešifrovaně a mohl by je každý odposlechnout.

Kapitola 5

Závěr

Cílem této práce bylo navrhnout a experimentálně realizovat bezdrátově ovládané LED osvětlení do domácnosti. To se podařilo, tedy bylo dosaženo cíle.

Prostudoval jsem způsob řízení světel počítačem se zaměřením na vestavěné systémy a bezdrátovou komunikaci, výsledek je uveden v druhé kapitole. Na základě nabytých vědomostí jsem navrhl zdroj světla s LED řízený mikrokontrolérem a způsob jeho ovládání prostřednictvím Wi-Fi. Tento návrh je shrnut v kapitole čtyři. Úvodem jsem navrhl zapojení víceúčelové prototypové desky, jejíž srdcem je mikrokontrolér ESP32. Interakce s uživatelem probíhá skrze webové rozhraní, příkazy jsou přenášeny pomocí protokolu HTTP.

Začátkem vývoje byla navržena základní struktura komunikace, kostra uživatelského rozhraní a rozdělení zdrojového kódu tak, aby bylo snadné implementovat další moduly osvětlení či přidat podporu jiného hardware. Během implementace pak byly tyto návrhy postupně konkretizovány či mírně upraveny. Například došlo ke zjednodušení komunikačního protokolu, což vedlo k rychlejší odezvě.

Kromě funkčního vzorku vznikla i sada modulů, a to ve formě návrhu jejich desky plošných spojů a implementace odpovídajícího softwaru včetně webového rozhraní. Vytvořený funkční vzorek obsahuje pět výstupů, pět vstupů a řeší napájení mikrokontroléru či nahrávání programu. Navržené moduly pak cílí na miniaturizaci a specializují se na danou funkci jako je ovládání RGB pásku nebo čtení vstupu z vypínače. Kromě pěti běžných modulů vznikly i tři rozšířené moduly sloužící jako demonstrace schopností výsledného řešení. Pokročilí uživatelé ocení jednoduchý komunikační protokol včetně jeho popisu v rámci přílohy této práce a snadnou cestu k vytvoření dalších modulů, popř. úpravě těch stávajících. Řídící jednotka je ve formě softwaru implementovaná v jazyce C++ pro systém Linux.

Během práce jsem se zdokonalil v oblasti programování mikrokontrolérů, zejména pak těch od firmy Espressif. Taktéž jsem si prohloubil znalosti o fungování počítačových sítí, především pak protokolu HTTP. Bezdrátové osvětlení už dnes najdeme ve spoustě domácností a myslím si, že v budoucnu se s ním budeme setkávat stále častěji.

Na práci bych chtěl pokračovat vytvořením dalších modulů, těmi mohou být pohybová čidla, dvojvypínač či napojení na další prvky chytré domácnosti. Dále bych rád rozšířil zpracování událostí a pravidel.

Literatura

[1] Allafi, I. a Iqbal, T. Design and implementation of a low cost web server using ESP32 for real-time photovoltaic system monitoring. In: *2017 IEEE Electrical Power and Energy Conference (EPEC)*. Saskatoon, Kanada: [b.n.], 2017. DOI: 10.1109/EPEC.2017.8286184.

[2] Castro, E. a Hyslop, B. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. Brno, Česká Republika: Computer Press, 2012. ISBN 978-80-251-3733-8.

[3] Cheng, Y.-S., Chen, J.-H., Liu, Y.-H. a Wang, S.-C. Development of wireless RGB LED dimming control technology using smart phone. In: *2014 International Conference on Intelligent Green Building and Smart Grid (IGBSG)*. Tchaj-pej, Tchaj-wan: [b.n.], 2014. DOI: 10.1109/IGBSG.2014.6835220.

[4] Doleček, J. *Moderní učebnice elektroniky - 1. díl*. Praha, Česká Republika: BEN - technická literatura, 2005. ISBN 80-7300-146-2.

[5] Doleček, J. *Moderní učebnice elektroniky - 2. díl*. Praha, Česká Republika: BEN - technická literatura, 2005. ISBN 80-7300-161-6.

[6] Janík, D. *Zdroje světla*. Brno, Česká Republika, 2014. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií.

[7] Köhre, T. *Stavíme si bezdrátovou síť Wi-Fi*. Brno, Česká Republika: Computer Press, 2004. ISBN 80-251-0391-9.

[8] Kosek, J. *PHP – tvorba interaktivních internetových aplikací*. Praha, Česká Republika: Grada, 1998. ISBN 80-7169-373-1.

[9] Malý, M. *Hradla, volty, jednočipy: úvod do bastlení*. Praha, Česká Republika: CZ.NIC, 2017. ISBN 978-80-88168-23-2.

[10] Malý, M. *Porty, bajty, osmibity: počítače na koleni*. Praha, Česká Republika: CZ.NIC, 2019. ISBN 978-80-88168-39-3.

[11] Matoušek, D. *Práce s mikrokontroléry ATMEL AT899S8252: [měření, řízení, a regulace pomocí několika jednoduchých přípravků]. díl 2.* 1. vyd. Praha, Česká Republika: BEN - technická literatura, 2002. ISBN 80-7300-066-0.

[12] Matoušek, P. *Síťové služby a jejich architektura*. Brno, Česká Republika: Publishing house of Brno University of Technology VUTIUM, 2014. ISBN 978-80-214-3766-1.

[13] Philips Hue Go. *Computer Act!ve*. Londýn, Velká Británie: Dennis Publishing Ltd. 2016, č. 466, s. 29. ISSN 1461-6211. Dostupné z: http://search.proquest.com/docview/1764753310/.

[14] Žára, O. *JavaScript : programátorské techniky a webové technologie*. Brno, Česká Republika: Computer Press, 2015. ISBN 978-80-251-4573-9.

[15] Sadio, O., Ngom, I. a Lishou, C. Lightweight Security Scheme for MQTT/MQTT-SN Protocol. In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. Granada, Španělsko: [b.n.], 2019. DOI: 10.1109/IOTSMS48152.2019.8939177.

[16] Ikea Trådfri. *Computer Act!ve*. Londýn, Velká Británie: Dennis Publishing Ltd. 2017, č. 505, s. 25. ISSN 1461-6211. Dostupné z: http://search.proquest.com/docview/1940475564/.

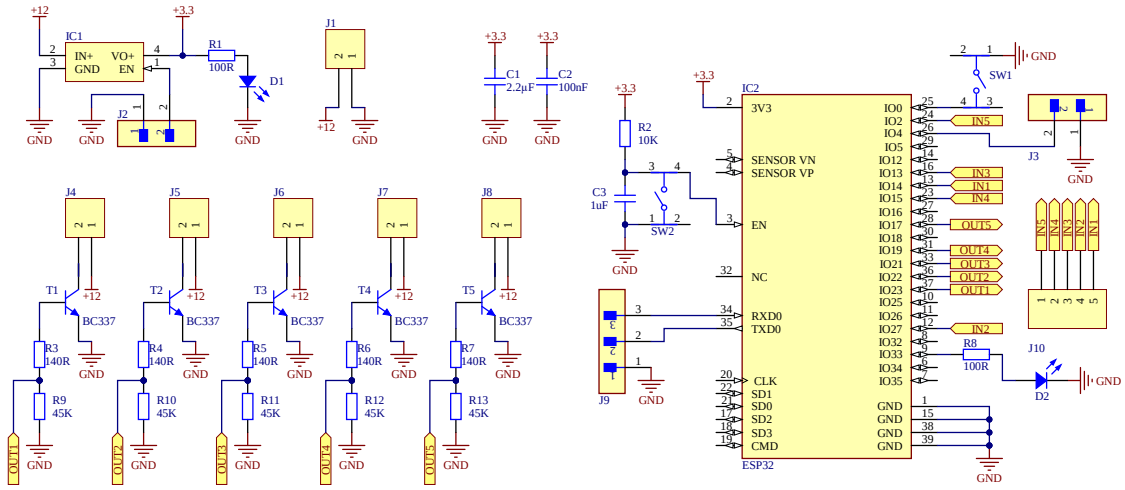
[17] Valov, N., Ivanova, D. a Valova, V. Increasing energy efficiency of a gas boiler using remote access control. In: *2020 7th International Conference on Energy Efficiency and Agricultural Engineering (EE AE)*. Ruse, Bulharsko: [b.n.], 2020, s. 1–4. DOI: 10.1109/EEAE49144.2020.9279063.

[18] Zhiguo, M. a Haiyan, W. Design of LED light parameters controlled based on WiFi. In: *2017 14th China International Forum on Solid State Lighting: International Forum on Wide Bandgap Semiconductors China (SSLChina: IFWS)*. Peking, Čína: [b.n.], 2017. DOI: 10.1109/IFWS.2017.8245972.

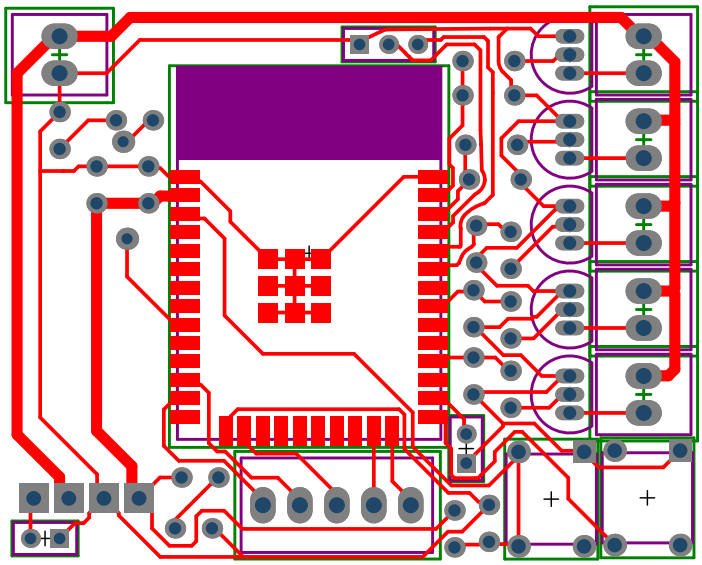
Příloha A

Schémata a DPS

A.1 Prototypová deska

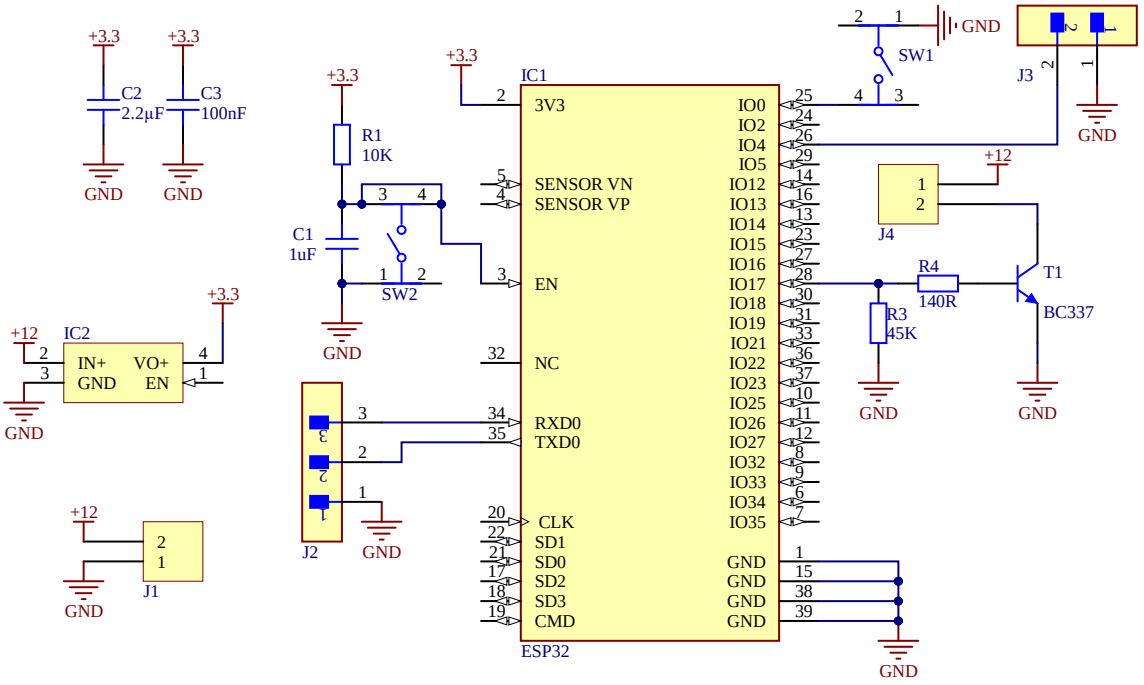


Obrázek A.1: Schéma prototypové desky

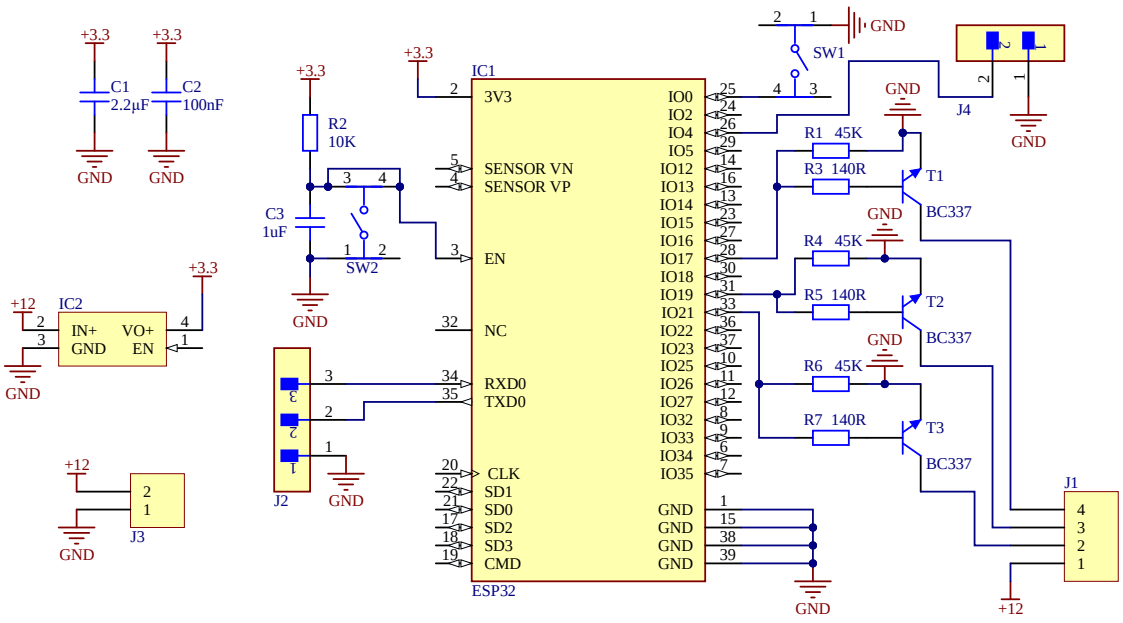


Obrázek A.2: DPS prototypové desky

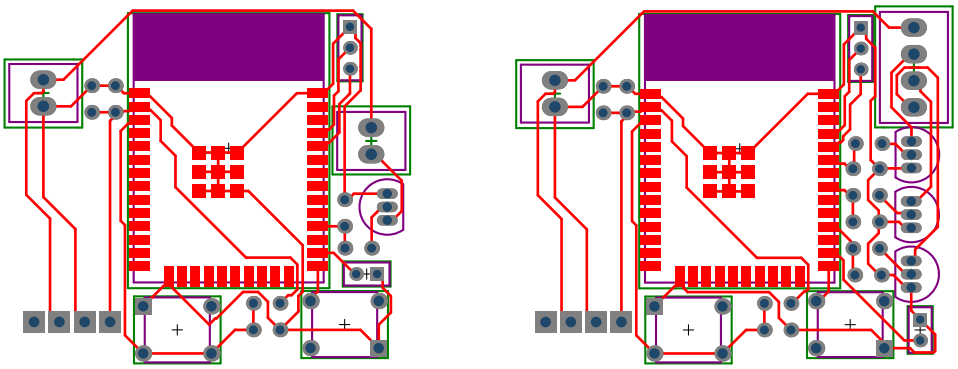
A.2 Jednobarevné a RGB světlo



Obrázek A.3: Schéma zapojení s jedním výstupem

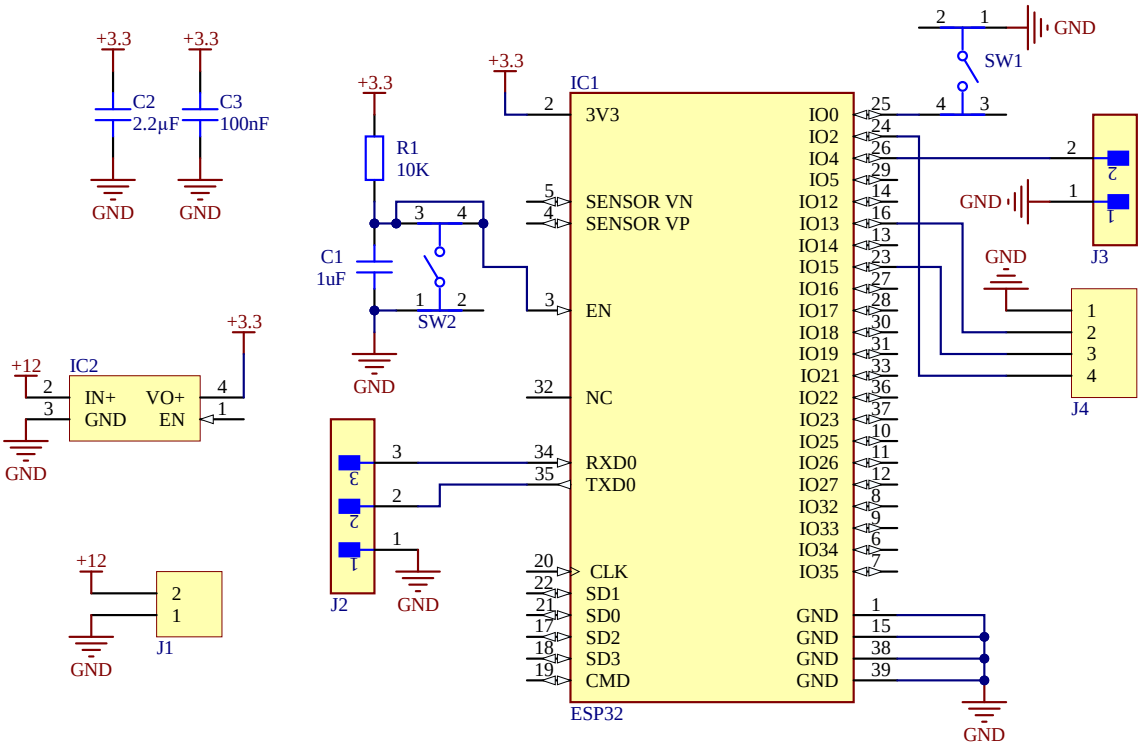


Obrázek A.4: Schéma zapojení se třemi výstupy

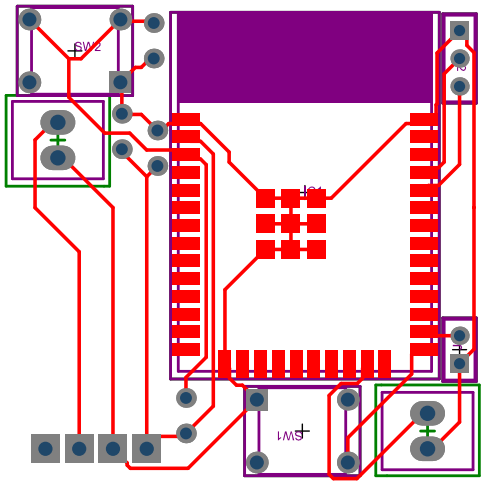


Obrázek A.5: DPS zapojení s jedním a třemi výstupy

A.3 Modul se vstupy (tlačítko)



Obrázek A.6: Schéma zapojení se třemi vstupy



Obrázek A.7: DPS zapojení se třemi vstupy

Příloha B

Seznam akcí a parametrů modulů

Vyvolání akce proběhne zavoláním adresy `http://{IP adresa}:{port}/{jméno akce}`, přičemž výchozí port je 80. Některé akce vyžadují povinné parametry, jiné mohou přijímat i nepovinné. Viz 2.5.4.

B.1 Všechny moduly

Každou akci lze spustit s parametrem `noEvent` bez hodnoty. Tento parametr zakáže vyvolání událostí pro tento příkaz.

B.1.1 Seznam akcí

`/` html soubor se základní stránkou, hlavička obsahuje obecné JavaScriptové funkce a CSS styly.

`/index` totéž jako `/`.

`/index.html` totéž jako `/`.

`/content` hlavní tělo webové stránky modulu (html soubor).

`/settings` html soubor s nastavením modulu, nutno inicializovat pomocí objektu třídy `Settings` v rámci JavaScriptu.

`/style.css` styly konkrétního modulu.

`/script.js` skripty konkrétního modulu.

`/favicon.ico` ikona společná pro všechny moduly.

`/get` obecný příkaz pro získání libovolného definovaného parametru, lze i více najednou. Př. `http://localhost:8081/get?name&port`, odpověď `{"name": "Jmeno", "port": 80}`.

`/set` obecný příkaz pro nastavení libovolného definovaného parametru, lze i více najednou. Př. `http://localhost:8081/set?name="Jmeno"`

`/getList` seznam všech definovaných parametrů, které lze přečíst. Odpovědí je *json*, obsahující pole stringů na indexu *gets*.

`/setList` seznam všech definovaných parametrů, které lze nastavit. Odpovědí je *json*, obsahující pole stringů na indexu *sets*.

`/getAll` seznam všech definovaných parametrů ke čtení včetně jejich hodnot, vhodné pro ladící účely.

`/setCtrlPort` nastaví port řídící jednotky. Při tom **nevyvolá** odesílání *hello* zprávy. Tento příkaz používá řídící jednotka, když přijme *hello* zprávu skrze broadcast. Vyžaduje parametr `port`.

`/actionList` seznam všech možných akcí daného modulu. Odpovědí je *json*, obsahující pole stringů na indexu *actions*.

`/eventList` seznam všech událostí, které může modul vyvolat. Odpovědí je *json*, obsahující pole stringů na indexu *events*.

B.1.2 Parametry `get`

`broadcastHello [bool]` informace, zda se má hello zpráva zasílat jako broadcast (`true`) nebo TCP na zadanou adresu (`false`).

`ctrlIp [string]` IP adresa řídící jednotky. Pouze pokud `broadcastHello = false`.

`ctrlPort [int]` port řídící jednotky. Pouze pokud `broadcastHello = false`.

`port [int]` port na kterém naslouchá modul.

`name [string]` jméno modulu.

`id [int]` ID modulu.

`type [string]` typ modulu.

`wifiMode [int]` režim Wi-Fi. 0 – přístupový bod (výchozí), 1 – stanice.

`apSsid [string]` SSID vysílané Wi-Fi.

`apPassword [string]` heslo vysílané Wi-Fi.

`apEncryption [int]` typ zabezpečení. 0 – WPA2, 1 – bez hesla.

`apChannel [int]` kanál vysílané Wi-Fi.

`stationSsid [string]` SSID sítě, na kterou se modul pokusí připojit.

`stationPassword [string]` heslo sítě, ke které se modul pokusí připojit.

`stationEncryption [int]` typ zabezpečení. 0 – WPA2, 1 – bez hesla.

B.1.3 Parametry `set`

`broadcastHello [bool]` informace, zda se má hello zpráva zasílat jako broadcast (`true`) nebo TCP na zadanou adresu (`false`).

`ctrlIp [string]` IP adresa řídící jednotky. Pouze pokud `broadcastHello = false`.

`ctrlPort [int]` port řídící jednotky. Pouze pokud `broadcastHello = false`.

`port [int]` port na kterém naslouchá modul.

`name [string]` jméno modulu.

`wifiMode [int]` režim Wi-Fi. 0 – přístupový bod (výchozí), 1 – stanice.

`apSsid [string]` SSID vysílané Wi-Fi.

`apPassword [string]` vždy vrací ********.

`apEncryption [int]` typ zabezpečení. 0 – WPA2, 1 – bez hesla.

`apChannel [int]` kanál vysílané Wi-Fi.

`stationSsid [string]` SSID sítě, na kterou se modul pokusí připojit.

`stationPassword [string]` vždy vrací ********.

`stationEncryption [int]` typ zabezpečení. 0 – WPA2, 1 – bez hesla.

B.2 Modul *Controller*

B.2.1 Seznam akcí

`/hello` zpracování hello zprávy, kterou vysílá každý modul při inicializaci http serveru. Povinně musí obsahovat tyto parametry: `id` – jedinečné ID modulu, `type` – typ modulu, `port` – port, na kterém modul poslouchá a `name` – jméno modulu.

`/event` zpracování události. Musí obsahovat parametr `id` s ID modulu, který vyvolal událost nebo 0 pro nespecifikovaného původce. Dalšími parametry jsou jména jednotlivých událostí, může jich být i více. Každá událost může nést i libovolnou hodnotu.

`/approveModule` schválení jednoho modulu s daným ID. Vyžaduje parametr `id`.

`/removeModule` odstranění jednoho modulu s daným ID. Vyžaduje parametr `id`.

`/modulesList` seznam všech registrovaných modulů. Odpovědí je *json*, obsahující pole objektů modulů na indexu *modules*.

`/modulesForApprovalList` seznam modulů čekajících na schválení. Odpovědí je *json*, obsahující pole objektů modulů na indexu *modules*.

`/moveModule` změna pořadí modulu s daným ID na novou pozici. Vyžaduje parametr `id` a `index`.

`/addRule` přidání pravidla. Povinně musí obsahovat tyto parametry: `source` – jedinečné ID modulu vyvolávající akci (0 pro libovolný), `event` – typ události, `target` – ID cílového modulu (0 pro všechny moduly) a `action` – vyvolaná akce. Volitelně může být definováno `value`, kdy první znak určuje typ porovnání (=><) a zbytek hodnotu. = vyžaduje identickou hodnotu, >< pak celočíselnou hodnotu.

`/removeRule` odstraní pravidlo. Povinně musí obsahovat tyto parametry: `event` – typ mazané události, `index` – pořadí mazané události v seznamu všech událostí tohoto typu.

`/rulesList` seznam všech pravidel. Odpovědí je *json*. Na indexu *rules* je objekt, kde klíčem je jméno události a hodnotou pole s pravidly.

B.3 Modul *Button*

B.3.1 Seznam akcí

`/click` simulace stisknutí tlačítka.

`/longClick` simulace dlouhého stisknutí tlačítka.

B.4 Modul *Light*

B.4.1 Parametry `get`

`state [bool]` informace, zda světlo svítí.

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on].

B.4.2 Parametry `set`

`on [void]` rozsvítí světlo.

`off [void]` zhasne světlo.

`toogle [void]` změní režim světla (on ↔ off).

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on].

B.5 Modul *LightBrightness*

B.5.1 Parametry `get`

`brightness [int]` jas světla.

`state [bool]` informace, zda světlo svítí.

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on/onMax/onSet].

`afterStartVal [int]` jas, který se nastaví po spuštění je-li `afterStart = onSet`.

B.5.2 Parametry `set`

`on [void]` rozsvítí světlo.

`off [void]` zhasne světlo.

`toogle [void]` změní režim světla (on ↔ off).

`brightness [int]` změní jas světla.

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on/onMax/onSet].

`afterStartVal [int]` jas, který se nastaví po spuštění je-li `afterStart = onSet`.

B.6 Modul *LightButton*

Kombinuje parametry `set` a `get` modulu *Light* a k tomu přidává akce modulu *Button*.

B.7 Modul *LightRgb*

B.7.1 Parametry `get`

`brightness [int]` jas světla.

`color [string]` barva světla ve tvaru RRGGBB.

`state [bool]` informace, zda světlo svítí.

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on/onSet].

`afterStartBrightness [int]` jas, který se nastaví po spuštění je-li `afterStart = onSet`.

`afterStartColor [string]` barva ve tvaru `RRGGBB`, která se nastaví po spuštění je-li `afterStart = onSet`.

B.7.2 Parametry `set`

`on [void]` rozsvítí světlo.

`off [void]` zhasne světlo.

`toogle [void]` změní režim světla on ↔ off.

`brightness [int]` změní jas světla.

`color [string]` barva světla ve tvaru RRGGBB.

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on/onSet].

`afterStartBrightness [int]` jas, který se nastaví po spuštění je-li `afterStart = onSet`.

`afterStartColor [string]` barva ve tvaru `RRGGBB`, která se nastaví po spuštění je-li `afterStart = onSet`.

B.8 Modul *LightSlow*

B.8.1 Parametry `get`

`brightness [int]` aktuální jas světla.

`state [string]` stav, v jakém se světlo nachází [on/off/fadeOn/fadeOff].

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on].

`onDuration [int]` doba trvání rozsvěcení světla v sekundách.

`offDuration [int]` doba trvání zhasínání světla v sekundách.

`fade [bool]` změna jasu bude plynulá (`true`) nebo skoková (`false`).

B.8.2 Parametry `set`

`on [void]` spustí časovač pro rozsvícení světla.

`off [void]` spustí časovač pro zhasnutí světla.

`fastOn [void]` okamžitě rozsvítí světlo.

`fastOff [void]` okamžitě zhasne světlo.

`afterStart [string]` akce, která proběhne po spuštění [auto/off/on].

`onDuration [int]` doba trvání rozsvěcení světla v sekundách.

`offDuration [int]` doba trvání zhasínání světla v sekundách.

`fade [bool]` změna jasu bude plynulá (`true`) nebo skoková (`false`).

B.9 Modul *Switch*

B.9.1 Seznam akcí

`/change` simulace změny stavu tlačítka.

B.10 Modul *SwitchIndicator*

B.10.1 Seznam akcí

`/change` simulace změny stavu tlačítka.

B.10.2 Parametry `get`

`indicator [string]` aktuální barva indikační LED [none/green/yellow/red].

B.10.3 Parametry `set`

`indicator [string]` nastaví barvu indikační LED [none/green/yellow/red].

Příloha C

Vytvoření nového modulu

Vytvoření nového modulu je poměrně jednoduché, zejména jedná-li se o úpravu již existujícího. Každý modul potřebuje implementovat potomka `C++` třídy `Module` a vytvořit jeho webovou stránku. To vše za pomoci existujících pomocných funkcí.

C.1 Dědění třídy `Module`

Mateřská třída `Module` se nalézá ve složce `src/Module`. Do stejné složky patří i implementace nového modulu. Jména souborů **musí** být ve tvaru `Module<Jméno modulu>.cpp` a `Module<Jméno modulu>.hpp`. Tedy např. `ModuleExample.cpp`. Díky tomu bude možné automaticky vygenerovat pravidla pro překlad a stejně tak i vkládat správné soubory při překladu.

V hlavičkovém souboru se pak očekává třída se stejným jménem, jako je jméno souboru, dědící třídu `Module`. Obrázek C.1 ukazuje příklad jednoduché třídy `ModuleExample`. Třída implementuje jednu dodatečnou akci `akce1`, která zároveň vyvolá událost typu `OTHER_EVENT`. Mimo to lze nastavit i číst parametr `parametr1`, který je typu `bool`.

V rámci konstruktoru jsou načteny data z trvalé paměti, popř. inicializovány při prvním spuštění modulu a zaregistrovány všechny výše zmíněné callback funkce.

```
class ModuleExample : public Module {
    ModuleExample(Id id) {
        if (init) { // Tyto hodnoty zapsat pouze pri prvnim spusteni
            io->save("hodnota", true);
        }
        else { // Jinak nacist
            io->load("hodnota", _hodnota);
        }

        additionalActions = actions; // Zaregistruji akce
        additionalActionsCount = sizeof(actions) / sizeof(Action);

        additionalSets = sets; // Zaregistruji set funkce
        additionalSetsCount = sizeof(sets) / sizeof(Param);

        additionalGets = gets; // Zaregistruji get funkce
        additionalGetsCount = sizeof(gets) / sizeof(Param);

        additionalEvents = events; // Zaregistruji seznam udalosti
        additionalEventsCount = sizeof(events) / sizeof(Event);
    }

private:
    static bool _hodnota;

    // Callback funkce pro akci
    static HttpAnswer akce1(HttpData) {
        ... // Vykonani akce 1
        addEvent(OTHER_EVENT); // Vyvolani udalosti
        return {NO_CONTENT_204, nullptr, 0}; // Akce nevraci zadna data
    }
    // Funkce pro parametr akce set
    static void hodnota(std::string &val) {
        _hodnota = stringToBool(val); // Zpracovani parametru, popr. jeho hodnoty
        io->save("hodnota", _hodnota); // Ulozeni do trvale pameti
    }
    // Funkce pro parametr akce get
    static Data parametr1() {
        return {&hodnota, BOOL_T}
    }
    // Seznam vsech akci
    static constexpr Action actions[] = {
        {"akce1", akce1},
    }
    // Seznam vsech nastavitelnych parametru
    static constexpr Param sets[] = { {.name = "parametr1", .set = parametr1}, }
    // Seznam vsech ziskatelnych parametru
    static constexpr Param gets[] = { {.name = "parametr1", .set = parametr1}, }
    // Seznam udalosti, které může modul vyvolat.
    static constexpr Event events[] = { OTHER_EVENT, };
}
```

Obrázek C.1: Ukázka třídy `ModuleExample`

C.2 Webová stránka modulu

Vytvoření webové stránky modulu je obdobně jednoduché. Společné soubory se nachází ve složce `WWW/template`. Soubory pro konkrétní modul patří do stejnojmenné podsložky, tedy např. `WWW/ModuleExample`. Zde je nutné vytvořit 5 souborů (mohou být i prázdné). Popis jejich obsahu následuje dále.

C.2.1 `content.html`

Obsahuje html popis ovládacích prvků modulu. Tedy zpravidla grafické tlačítko symbolizující daný modul (například žárovka). Preprocesor nahradí všechny výskyty řetězce `__ID__` skutečným ID při kompilaci. Tak lze zajistit, že HTML *id* daného elementu bude skutečně jedinečné (je nutné počítat s tím, že v rámci webové stránky řídící jednotky se tento soubor v ní může vyskytnout hned několikrát).

C.2.2 `header.html`

Obsah tohoto souboru bude při kompilaci připojen na konec *header* sekce šablony. Lze tak doplnit např. atribut *title*. Dále lze tímto způsobem vložit společné pomocné soubory ze složky `WWW/template`. Soubory `ajax.js`, `functions.js`, `main.js`, `settings.css`, `settings.js`, `style.css`, `window.css` a `window.js` se vkládají automaticky. Ostatní je nutné vložit ručně. I zde lze využít preprocesor. Konkrétně formule `/* #{NAZEV} */` bude nahrazena odpovídajícím souborem. Např. pro pomocné soubory žárovky následujícím způsobem

```
<script>
/* #{BULB_JS} */
</script>

<style>
/* #{BULB_CSS} */
</style>
```

Obrázek C.2: Ukázka souboru `header.html`

Momentálně jsou podporovány pouze soubory `BULB_JS`, `BULB_CSS` pro moduly světel a `BUTTON_JS` s `BUTTON_CSS` pro tlačítka.

C.2.3 `script.js`

Prostor pro JavaScript konkrétního modulu. V tomto souboru **musí** být implementována funkce `<NazevModulu>_<IdModulu>(tile)`. I v rámci tohoto souboru jsou pomocí preprocesoru nahrazeny všechny výskyty řetězce `__ID__` skutečným ID, takže by jemné funkce mohlo vypadat např. takto `ModuleExample___ID__`. Tato funkce je volána při vykreslení modulu v rámci okna řídící jednotky. Jejím úkolem je doplnit údaje (zejména jméno) do objektu třídy *Tile* a naopak získá přístup k inicializovanému objektu třídy *Ajax*.

Pokud je k webovému rozhraní modulu přistoupeno přímo, není automaticky volána žádná funkce. Spuštění funkce je nutné zaregistrovat pomocí `afterMain.push(func)`, což zajistí volání až ve chvíli, kdy je vše potřebné inicializováno. Tedy např. objekt `ajax` stejnojmenné třídy *Ajax*. Zároveň by měla být někde v kódu do proměnné `reloadFunc` přiřazena

funkce, která se má vykonat při stisknutí tlačítka pro znovu-načtení obsahu. Soubor může samozřejmě obsahovat další pomocné funkce či třídy, to není nijak omezeno.

C.2.4 `settings.html`

HTML souboru s nastavením modulu. Lze využít generátor implementovaný v souboru `settings.js`, kde se nachází i jeho popis.

C.2.5 `style.css`

CSS pravidla pro vykreslení daného modulu.

Příloha D

Obsah přiloženého CD

Na přiloženém CD se nachází následující soubory a složky:

`code` – veškeré zdrojové kódy pro mikrokontrolér a webové rozhraní.

`code\doc` – dokumentace generovaná nástrojem *Doxygen* ve formě webové stránky.

`code\src\Hw` – implementace platforem.

`code\src\Module` – implementace modulů.

`datasheet` – datasheety použitých součástek.

`DP` – latexové soubory této práce.

`DP\obrazky` – obrázky použité v rámci práce.

`DP\schemata` – soubory programu *Altium Designer* pro generování obrázků do teoretické části práce.

`DP\zdroje` – uložené webové stránky ve formátu *pdf*, které byli použity jako podpůrná literatura.

`DPS` – soubory programu *Altium Designer* se zapojením a DPS jednotlivých modulů.

`demo.mp4` – videoukázka výsledného řešení.