VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

OVLÁDAČ ELEKTRICKÝCH ZAŘÍZENÍ ZALOŽENÝ NA PLATFORMĚ RASPBERY PI
ELECTRONIC MODULES CONTROLLER BASED ON RASPBERY PI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Richard Kučkovský**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Michal Pavlík, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Mikroelektronika a technologie**
Ústav mikroelektroniky

Student: Richard Kučkovský ***ID:*** 174223
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Ovládač elektrických zařízení založený na platformě Raspbery Pi

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte prototyp ovladače elektrických zařízení založený na platformě Raspbery Pi s využitím bezdrátových technologií. Systém by měl podporovat připojení zařízení ve hvězdicové komunikační struktuře.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: Ing. Michal Pavlík, Ph.D.
Konzultant:

doc. Ing. Jiří Háze, Ph.D.
předseda oborové rady

ABSTRAKT

Cieľom práce je navrhnúť bezdrôtový ovládač rôznych elektrických zariadení založený na platforme Raspberry Pi. Navrhnuté a vyrobené moduly bezdrôtovo komunikujú s riadiacou jednotkou. Nastavenie a ovládanie zariadení prebieha pomocou mobilnej aplikácie dostupnej na hlavné mobilné platformy. Ako riadiaca jednotka je použitý mikropočítač Raspberry Pi a bezdrôtová komunikácia je dosiahnutá použitím modulu NRF24l01+.

KLÚČOVÉ SLOVÁ

Raspberry pi, bezdrôtové ovládanie, elektrické zariadenia

ABSTRACT

Main objective of this thesis is to design wireless controller of electric equipment based on Raspberry Pi. Each designed module is communicating with controller wirelessly. Devices are operated by mobile application. As main controller is used microcomputer Raspberry Pi and wireless communication is achieved by using module NRF24l01+.

KEYWORDS

Raspberry Pi, wireless control, electric equipment

KUČKOVSKÝ, R. *Ovládač elektrických zařízení založený na platformě Raspbery Pi.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. XY s. Vedúci bakalárskej práce Ing. Michal Pavlík, Ph.D.

PREHLÁSENIE

Prehlasujem, že svoju bakalársku prácu na tému Ovládač elektrických zariadení založený na platforme Raspberry Pi vypracoval samostatne pod vedením vedúceho semestrálnej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej semestrálnej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto semestrálnej práce som neporušil autorské práva tretích osôb, obzvlášť som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a majetkových a som si plne vedomý následkov porušení ustanovení § 11 a nasledujúcich zákona č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení časti druhej, hlavy VI. diel 4 Trestného zákonníku č. 40/2009 Sb.

V Brne dne ....

(podpis autora)

POĎAKOVANIE

Chcel by som sa poďakovať vedúcemu práce Ing. Michalovi Pavlíkovi, Ph.D., za jeho odbornú pomoc a konzultácie, ktoré mi poskytol pri vypracovaní bakalárskej práce.

OBSAH

ÚVOD

S rozvíjajúcimi sa technológiami sa dostáva čím ďalej, tým viac zariadení do domácností. Medzi najprogresívnejšie rozvíjajúce sa technológie z oblasti domácností patrí koncept inteligentných domov. V súčasnosti existuje už mnoho produktov poskytujúcich možnosť premeniť domácnosť na inteligentnú. Väčšina riešení je však pre bežnú domácnosť finančne neprístupná a častokrát poskytuje množstvo služieb, ktoré nie sú potrebné a zvyšujú zložitosť celého systému. Množstvo lacných riešení naopak ponúka nespoľahlivé ovládače konkrétnych zariadení, ktoré je často potrebné kombinovať, čo má za následok viac nekompatibilných systémov v jednej domácnosti.

Cieľom tejto práce je preskúmať vhodné bezdrôtové technológie a navrhnúť základný koncept ovládača elektrických zariadení a vyrobiť prototyp. Táto práca nemá za cieľ vytvoriť systém inteligentnej domácnosti, avšak princípom svojej funkcie tvorí základ, na ktorom sa takýto systém dá vybudovať. Hlavnou funkciou navrhovaného ovládača je monitorovanie a kontrola teploty v miestnostiach pomocou teplotných senzorov a termoelektrických hlavíc na radiátoroch. Ďalšou súčasťou systému sú ovládače LED svietidiel s nastaviteľnou intenzitou osvetlenia a plynulými prechodmi. Okrem týchto predpripravených aplikácii bude možné ovládať akékoľvek elektrické zariadenie.

Celý systém bude ovládateľný pomocou mobilnej aplikácie, kde užívateľ bude schopný nastaviť požadované vlastnosti systému a ovládať jednotlivé elektrické zariadenia.

1 BEZDRÔTOVÉ TECHNOLÓGIE

Bezdrôtová komunikácia patrí medzi najväčšie objavy ľudstva. Bezdrôtové technológie umožňujú prenášanie informácií bez pomoci elektrických vodičov. Vzdialenosť, na ktorej sa informácie prenášajú, môže byť od pár metrov po niekoľko tisíc kilometrov. Bezdrôtové technológie sa v dnešnej dobe využívajú v takmer každom zariadení a hlavne v komerčnej sfére. Medzi hlavné výhody bezdrôtovej komunikácie patrí užívateľská prívetivosť a rýchlosť.

1.1 Wi-Fi

Wi-Fi patrí medzi najobľúbenejšie a v domácnostiach najpoužívanejšie bezdrôtové technológie. Využíva sa hlavne na bezdrôtové pripojenie k internetu. Pôvodná verzia štandardu IEEE 802.11 bola vydaná v roku 1997 a využívala frekvenciu 2.4 GHz. V roku 1999 bol štandard aktualizovaný a vznikla organizácia, ktorá spravuje obchodnú značku Wi-Fi – Wi-Fi Aliance. Na základe vlastností, ktoré Wi-Fi má, sa zaradzuje nielen medzi osobné počítačové siete – PAN, ale aj medzi lokálne počítačové siete – LAN.

Sieť Wi-Fi sa skladá z jedného alebo viacerých prístupových bodov – AP. Hlavnú funkciu má sieťový identifikátor – SSID. SSID je reťazec ASCII znakov dlhý až 32 znakov, ktorý slúži ako jedinečný identifikátor siete. Tento identifikátor je pravidelne vysielaný z každého prístupového bodu, takže klient je schopný ho prečítať a k sieti sa pripojiť. AP poskytujú možnosť nevysielať SSID a sieť je tým pádom neviditeľná, v takom prípade musí klient toto SSID poznať dopredu, aby sa mohol pripojiť. Pri pripojení je SSID prenášané bez šifrovania a tak je jednoduché ho odchytiť a sieť odhaliť. Na zabezpečenie bezdrôtových sietí sa používajú iné metódy. Jednou z najstarších metód šifrovania je šifrovanie pomocou WEP kľúčov. Štandardne 64 bitové WEP šifrovanie využíva 40 bitový kľúč v reťazci s 24 bitovým inicializačným vektorom, ktorý tvorí RC4 kľúč. Toto šifrovanie však nebolo dostatočné a už v roku 2001 bolo prelomené. V súčasnosti sa používa najmä WPA a WPA2. WPA2 šifrovanie používa namiesto sériového RC4 blokovú šifru AES.

Dosah Wi-Fi je ovplyvnený použitou frekvenciou, silou signálu, ale hlavne použitým typom antény, ktorý určuje, či sa jedná o krátky dosah - niekoľko desiatok metrov, alebo dlhý dosah, ktorý môže byť až niekoľko desiatok kilometrov. Aby však dosiahla požiadavky LAN aplikácii, má Wi-Fi vyššiu spotrebu energie ako

konkurenčné technológie, ktoré ale neposkytujú takú vysokú rýchlosť prenosu dát [1].

1.2 Bluetooth

Bluetooth patrí medzi celosvetovo využívane komunikačné štandardy. Bola uvedená v roku 1999 a odvtedy sa teší veľkej obľube. Slúži hlavne na prepojenie zariadení na krátku vzdialenosť, využíva sa napríklad na prenos súborov medzi mobilnými zariadeniami, v handsfree slúchadlách, alebo na prenos hudby do bezdrôtových reproduktorov. Medzi jeho hlavné výhody patrí nízka spotreba elektrickej energie a cena, ktoré umožnili, aby sa v súčasnosti technológia Bluetooth nachádzala takmer v každom elektrickom zariadení od áut a notebookov až po fitness náramky. Technológia Bluetooth je definovaná štandardom IEEE 802.15.1 a patrí do kategórie osobných počítačových sietí – PAN.

Bluetooth pracuje na frekvencii 2,4 GHz. Dáta sú rozdelené do paketov a prenášané jedným z kanálov. Počas prenosu sa využíva metóda FHSS, ktorá počas jednej sekundy zmení frekvenciu - kanál, na ktorom prebieha prenos dát – 1600-krát. Prenosové kanály sú frekvencie s rozstupom 1 MHz. Táto metóda pomáha predchádzať rušeniu prenosu. Porušené pakety sa prenesú v ďalšom kole iným kanálom. Zariadenia majú niekoľko výkonových tried, ktoré určujú dosah, a to od triedy 3 s dosahom do 10 m, triedy 2 s dosahom 10 m a triedou 1 s dosahom 100 m. Skutočné vzdialenosti sú však závislé na množstve faktorov, a to hlavne na okolitých podmienkach, prekážkach, kvalite výrobku alebo stave batérie.

Vo verzii Bluetooth 1.2 bola prenosová rýchlosť 721 kbit/s. S novšími verziami sa rýchlosť zvyšovala. Vo verzii 3.0 to bolo už 24 Mbit/s. Najnovšia verzia Bluetooth 4 nepriniesla zvýšenie rýchlosti, ale za to výrazne znížila spotrebu energie. Všetky Bluetooth verzie majú spätnú kompatibilitu. Bluetooth využíva model Master-slave. Master môže komunikovať s až siedmimi slave zariadeniami. Všetky pripojené zariadenia zdieľajú hodinový signál z master zariadenia [2].

1.3 Modul NRF24L01+

Modul NRF24L01+ je jednoduchý vysielač-prijímač pracujúci na frekvencii 2,4 GHz navrhnutý na používanie v nízkoenergeticky náročných bezdrôtových aplikáciách. Jeho použitie je jednoduché a na správne zapojenie je potrebný malý počet externých komponentov. Modul NRF24L01+ sa nastavuje a používa pomocou SPI rozhrania. Integrovaný protokol je založený na paketovej komunikácii a umožňuje rôzne módy

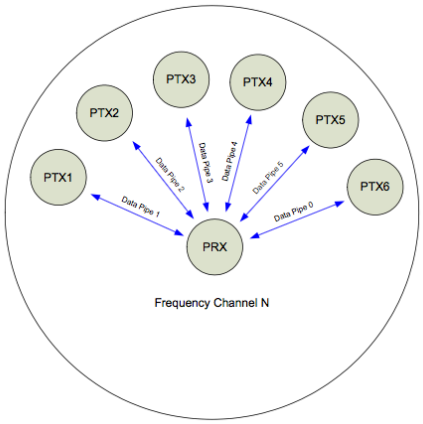
ovládania od manuálneho až po pokročilé automatické ovládanie. Interné FIFO zaručuje hladký prenos dát medzi mikrokontrolérom a bezdrôtovým čipom. Modul používa GFSK moduláciu a má nastaviteľný frekvenčný kanál, silu signálu a rýchlosť prenosu dát. Rýchlosť prenosu dát môže byť 1 Mb/s alebo 2 Mb/s. Pri rýchlosti 1 Mb/s má prijímač o 3 dB lepšiu citlivosť ako pri rýchlosti 2 Mb/s. Vyššia rýchlosť znamená nižšiu priemernú spotrebu a zníženú pravdepodobnosť kolízií. Aby sme dokázali vytvoriť prepojenie, vysielač aj prijímač musia pracovať s rovnakou rýchlosťou. Pri rýchlosti 1 Mb/s jeden kanál používa pásmo 1 MHz a pri 2 Mb/s sú to 2 MHz. Modul NRF24L01+ dokáže pracovať na frekvenciách od 2,400 GHz po 2,525 GHz. Rovnako ako pri rýchlosti musí aj vysielač aj prijímač byť nastavený na rovnaký kanál pre nadviazanie komunikácie [6].

Základné vlastnosti:

- využíva frekvenciu 2.4 GHz,
- dosahuje rýchlosť až 2 Mb/s,
- nízka spotreba elektrickej energie – 22 μA v standby móde,
- obsahuje napäťový regulátor,
- napájacie napätie v rozmedzí 1.9 V až 3.6 V,
- automatické spracovanie paketov,
- možnosť komunikácie až 6 zariadení.

1.3.1 Multiceiver

Multiceiver je vstavaná funkcia modulu, ktorá obsahuje 6 paralelných logických dátových kanálov s unikátnou adresou. Každý kanál má vlastnú fyzickú adresu dekódovaný v samotnom module.

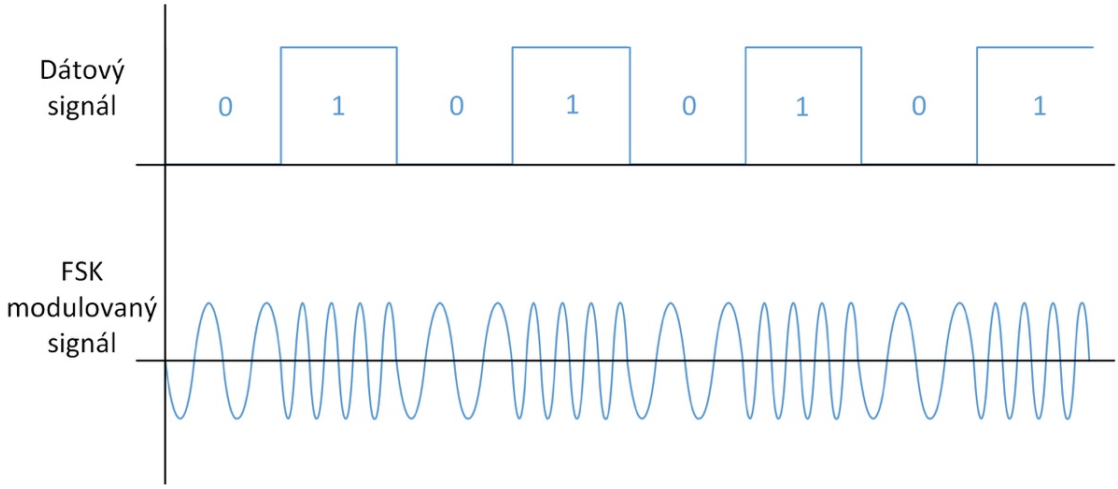


Obr. 1.1: Princíp funkcie multiceiver. Prevzaté z [6]

Modul NRF24L01+ nakonfigurovaný ako prijímač môže prijímať dáta až z 6 dátových kanálov na jednej frekvencii. Každý z týchto kanálov má unikátnu adresu a môže byť kontrolovaný samostatne. Dátové kanály sú vyhľadávané súčasne, ale iba jeden kanál môže prijímať paket v jednom čase.

1.3.2 GFSK modulácia

Kľúčovanie frekvenčným posunom (FSK) je modulačná schéma, pri ktorej je digitálna informácia prenášaná pomocou zmeny frekvencie nosného signálu. Táto metóda modulácie je nezávislá na amplitúde, preto má vysokú odolnosť voči šumu.



Obr. 1.2: Príklad binárnej modulácie FSK.

Namiesto okamžitej zmeny frekvencie pri GFSK prechádza signál navyše cez Gaussov filter, ktorý prechod zjemní. Tento proces má za následok zníženie výkonu postranných pásiem a tým pádom aj obmedzenie interferencie s ostatnými kanálmi [4].

2 NÁVRH RIEŠENIA PROJEKTU

V tejto kapitole je uvedený teoretický základ konceptu ovládania, požiadavky na jeho funkcie a riešenia otázok týkajúcich sa použitých zariadení a softvérového vybavenia. Kapitola obsahuje detailný popis modulov zariadenia, riadiacej jednotky a komunikácie navzájom.

2.1 Požiadavky

Hlavnou požiadavkou bolo použitie bezdrôtových technológií. Hlavným dôvodom tejto požiadavky je použitie systému v domoch s už existujúcou elektroinštaláciou. Na bezdrôtovú komunikáciu medzi modulmi a riadiacou jednotkou bol zvolený modul NRF24L01+, ktorý je detailnejšie popísaný v kapitole Bezdrôtové technológie. K jeho hlavným prednostiam patrí nízka spotreba energie, použitie ISM pásma a jednoduché použitie. Napriek tomu, že modul využíva rovnakú frekvenciu ako Wi-Fi, tak sa tieto dve technológie navzájom nerušia a preto je možné ho používať aj v priestoroch s väčšou hustotou pokrytia Wi-Fi sietí, ako napríklad v bytoch.

Ako riadiaca jednotka bude použitý mikropočítač Raspberry Pi vyvíjaný neziskovou spoločnosťou Raspberry Pi Foundation. Základnou podmienkou riadiacej jednotky je spoľahlivosť. Riadiaca jednotka musí fungovať neustále, aby to dokázala, musí byť spoľahlivý nielen samotný hardvér mikropočítača, ale aj programové vybavenie. Na vytvorenie serverovej aplikácie bežiacej v riadiacej jednotke bude použitý serverový engine *node.js*, ktorý využíva programovací jazyk JavaScript. Hotový program je veľmi stabilný a bez výpadkov.

Nevyhnutnou súčasťou systému je mobilná aplikácia, ktorá musí spĺňať hneď niekoľko požiadaviek:

- Užívateľská prívetivosť – aplikácia by mala byť jednoduchá nielen na ovládanie, ale užívateľ by mal byť schopný vďaka jednoduchému rozhraniu nastaviť zariadenie podľa predstáv, prípadne tieto nastavenia časom meniť.
- Kompatibilita – aplikácia by mala byť dostupná pre hlavné mobilné platformy iOS a Android.

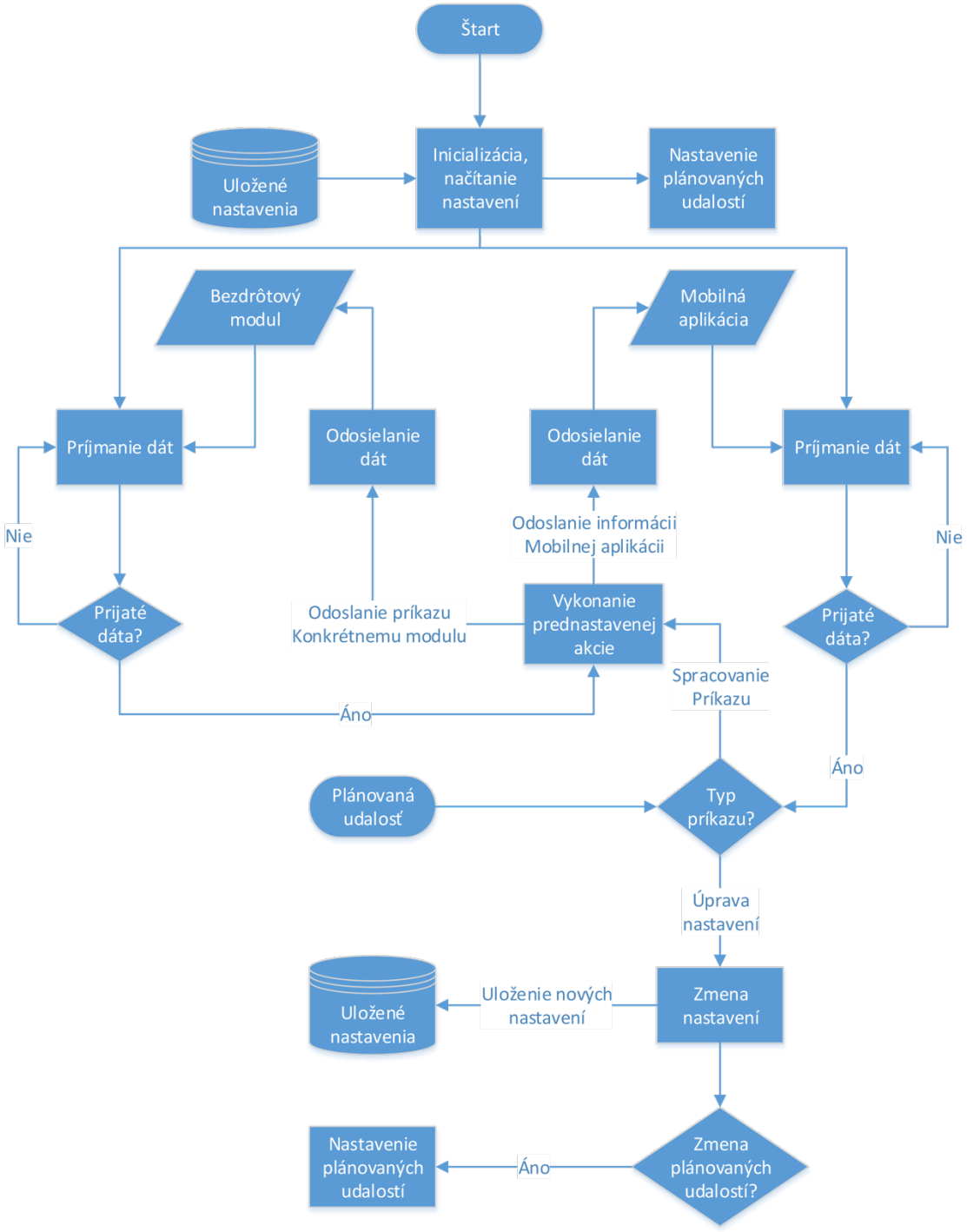
2.2 Riadiaca jednotka

Raspberry Pi je mikropočítač o veľkosti platobnej karty, v prípade najnovšej edície Raspberry Pi Zero sú rozmery ešte menšie. Prvá verzia bola vydaná vo februári 2012 pod názvom Raspberry Pi Model B, po ktorej nasledoval lacnejší a jednoduchší Model A. Raspberry Pi 3 Model B a Raspberry Pi Zero sú najnovšie mikropočítače z tejto rady. Základom mikropočítača je ARM procesor, ktorého parametre sa líšia podľa verzie počítača. Mikropočítač obsahuje aj grafický procesor a podľa verzie aj rôzne periféria. Neobsahuje však žiadne rozhranie pre pevný disk, preto je pre zavedenie systému a ukladanie dát potrebná MicroSD karta. Ako riadiaca jednotka bude použitý konkrétne model Raspberry Pi Zero, a to práve kvôli jeho malým rozmerom a nízkej cene. Vzhľadom na použitie výlučne GPIO portov zariadenia je preto možné použiť akúkoľvek verziu Raspberry s rovnakým rozložením pinov.

Riadiaca jednotka bude okrem samotného mikropočítača obsahovať bezdrôtový modul NRF24L01+ slúžiaci na bezdrôtové pripojenie ďalších modulov (pohybové senzory, ovládače svetiel, ovládače termoelektrických hlavíc).

Vlastnosti:

- komunikácia s mobilnou aplikáciou, odosielanie push hlásení,
- ukladanie nastavení a súvislostí medzi modulmi do databázy,
- komunikácia a riadenie modulov na základe nastavení.



Obr. 2.1: Vývojový diagram hlavného programu riadiacej jednotky.

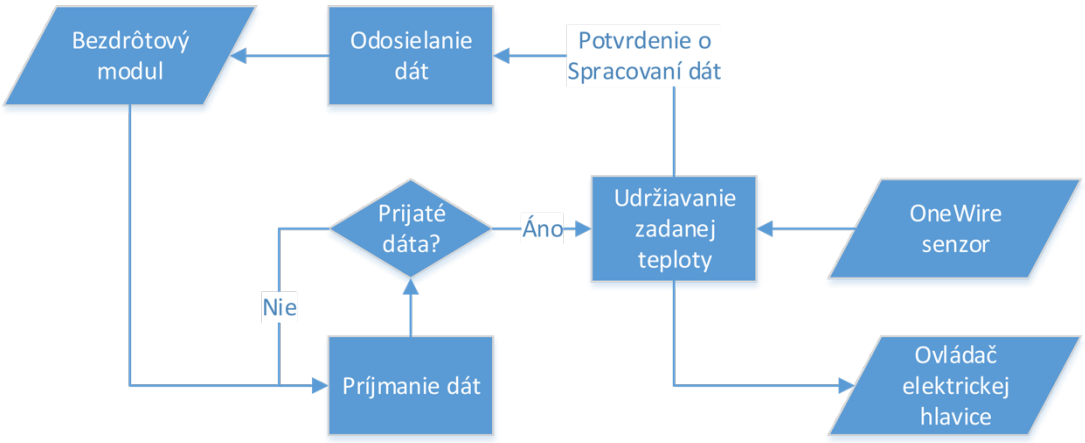
2.3 Ovládač radiátorov

Hlavným prvkom pri ovládaní radiátorov je termo-hlavica ovládaná elektrickým napätím určená na zatváranie a otváranie ventilov radiátorov a podlahových kúrení. Tieto hlavice sa vyrábajú v rôznych verziách s rôznym ovládacím napätím. Z bezpečnostných dôvodov je lepšie použiť hlavicu ovládanú jednosmerným napätím 24 V, čo navyše umožňuje použiť ako spínač rozumne malý tranzistor, vďaka ktorému sa zníži energetická náročnosť, a odstráni sa charakteristický zvuk, ktorý by spôsoboval inak použitý relé spínač.

Ovládač ďalej bude obsahovať bezdrôtový modul, teplotný senzor a mikrokontrolér, ktorý bude celý prvok riadiť.

Vlastnosti:

- po zapnutí vyhľadať riadiacu jednotku a odoslať svoje identifikačné údaje,
- pravidelne odosielať údaje o teplote,
- v prípade požiadavky nastaviť ventil tak, aby bola udržiavaná požadovaná teplota.



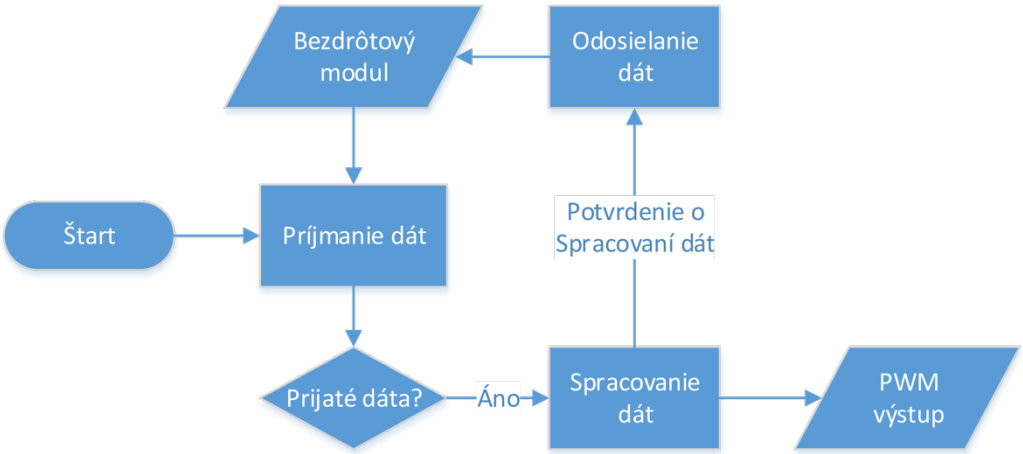
Obr. 2.2: Vývojový diagram modulu ovládača radiátorov.

2.4 Ovládač LED podsvietenia

Ovládač LED podsvietenia bude navrhnutý tak, aby poskytoval možnosť ovládať tri samostatné jednofarebné LED pásy alebo jeden RGB LED pás. Regulácia jasu bude prebiehať pomocou PWM signálu z mikrokontroléra.

Vlastnosti:

- po zapnutí vyhľadať riadiacu jednotku a odoslať svoje identifikačné údaje,
- pravidelne odosielať údaje o teplote,
- v prípade požiadavky z riadiacej jednotky nastaviť PWM signál na príslušných výstupoch.



Obr. 2.3: Vývojový diagram modulu ovládača LED svetiel.

2.5 Pohybový senzor

Pohybový senzor slúži na zjednodušenie a automatizáciu ovládania. Aby nebolo potrebné zakaždým používať mobilnú aplikáciu, stačí v nej nastaviť reakciu na pohyb, pričom reakcia môže byť v rôznych časoch rozličná.

Vlastnosti:

- po zapnutí vyhľadať riadiacu jednotku a odoslať svoje identifikačné údaje,
- po identifikácii v riadiacej jednotke prejsť do režimu spánku,
- pri zaznamenaní pohybu odoslať informácie riadiacej jednotke a vrátiť sa do režimu spánku.

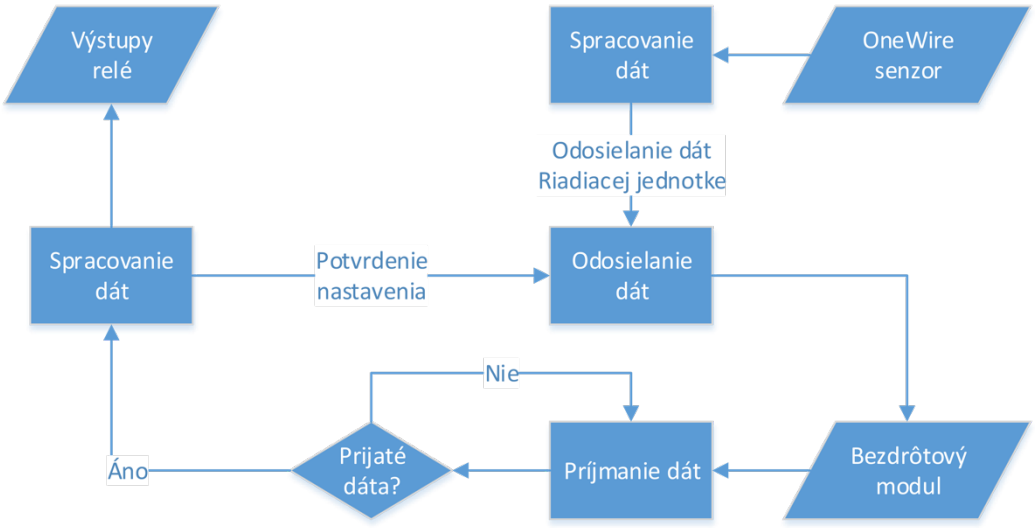
2.6 Relé ovládač

Relé ovládač slúži na ovládanie ostatných ľubovoľných elektrických zariadení. Okrem spínacích prvkov obsahuje jeden teplotný senzor na meranie teploty okolia a tiež prúdové senzory, ktoré kontrolujú prúd pretekajúci spínacím relé. Výsledkom takejto kontroly je možnosť použiť relé spolu s obyčajným vypínačom v schodiskovom zapojení na

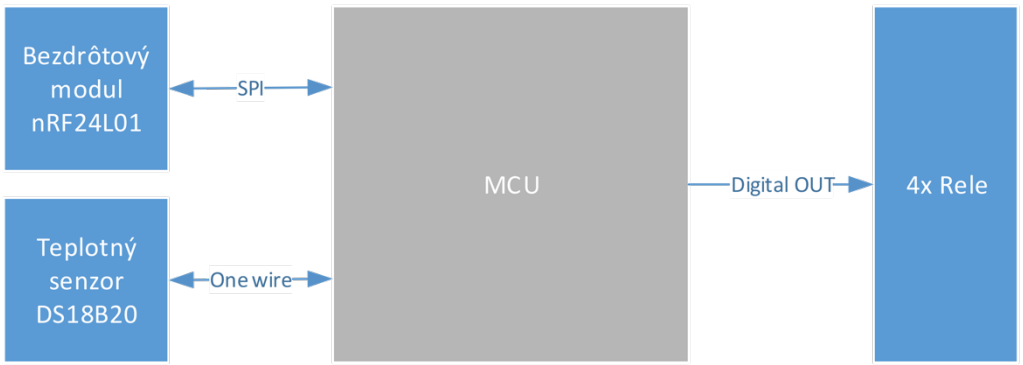
ovládanie svetla, a mobilná aplikácia bude mať vždy správnu informáciu o stave svetla aj v prípade, že bude vypnuté vypínačom.

Vlastnosti:

- po zapnutí vyhľadať riadiacu jednotku a odoslať svoje identifikačné údaje,
- pravidelne odosielať údaje o teplote,
- ak nastane zmena prúdu, odoslať informácie riadiacej jednotke,
- automaticky odpojiť relé, ak teplota alebo prúd presiahnu povolené hranice,
- v prípade požiadavky prepnúť relé do požadovanej polohy.



Obr. 2.4: Vývojový diagram modulu relé ovládača.



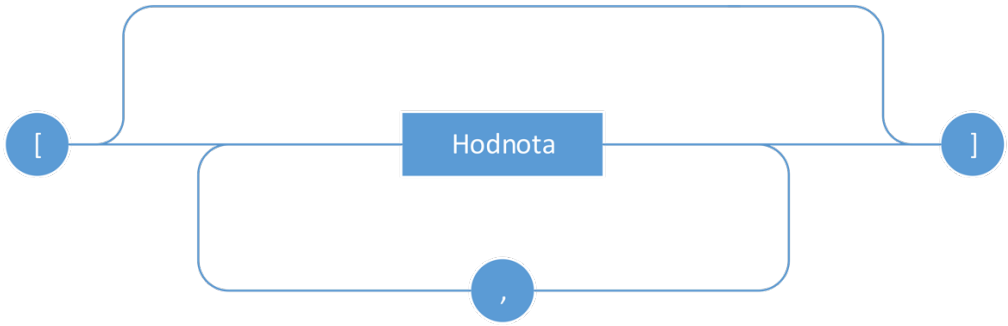
Obr. 2.5: Bloková schéma modulu relé ovládača.

2.7 Komunikácia

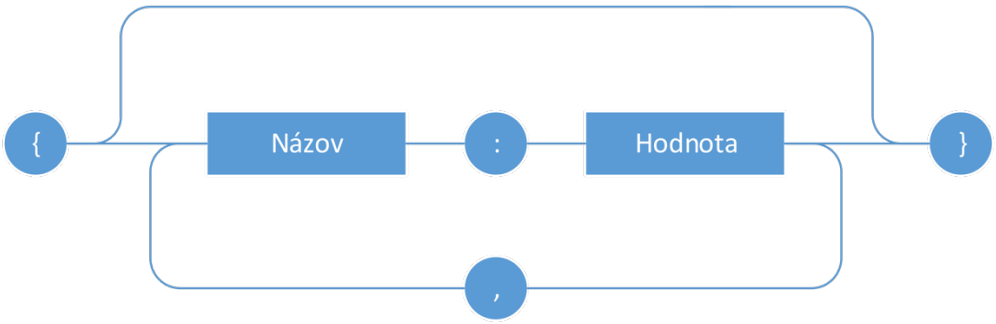
Komunikácia medzi mobilnou aplikáciou a riadiacou jednotkou bude prebiehať pomocou príkazov vo formáte JSON. Pre komunikáciu zariadení so serverom je potrebné navrhnúť vlastný komunikačný protokol, ktorý bude spravovať odosielanie príkazov a potvrdení, protokol by mal podporovať viacero zariadení v hviezdicovej štruktúre. Na komunikáciu medzi MCU a bezdrôtovým modulom NRF24L01+ je použitý protokol SPI.

2.7.1 JSON

JSON je odľahčený formát na výmenu dát. Jeho rozklad a generácia je pre program veľmi jednoduchá a navyše je ľahko čitateľný aj pre človeka. JSON je kompletne jazykovo nezávislý a je možné ho používať spolu s ľubovoľným programovacím jazykom. Používa však konvenciu podobnú programovacím jazykom založeným na jazyku C. Tieto vlastnosti robia z formátu JSON ideálny formát dát na prenos informácií medzi rôznymi programovacími jazykmi [5].



Obr. 2.6: Štruktúra dát typu *array* vo formáte JSON



Obr. 2.7: Štruktúra dát typu *object* vo formáte JSON

2.7.2 Používanie modulu NRF24L01+

Prevádzkové módy

Mód *Power Down* – v tomto móde je modul neaktívny a využíva minimálne množstvo energie. Všetky registre sú dostupné pomocou SPI umožňujú zmenu konfigurácie a prenos dát.

Mód *Standby-I* – je určený na minimalizovanie priemernej spotreby energie, ale zároveň zachovanie rýchlych spúšťacích časov.

Mód *Standby-II* – tento mód je aktívny po vyprázdnení všetkých TX FIFO.

Mód *RX* – je aktívny keď, je modul používaný ako prijímač. V tomto móde prijímač neustále demoduluje signál a posiela dáta do Baseband Protocol Engine, ktorý vyhľadáva platné dáta kontrolou adresy a CRC. V prípade, že paket je platný, sú dáta presunuté do FIFO. Ak je FIFO plné, dáta sú zahodené. Modul ostáva v móde *RX* až pokiaľ nedostane príkaz na prechod do *Standby-I* alebo *Power Down*.

Mód *TX* - v tomto móde modul pracuje ako vysielač. Po vyprázdnení FIFO prechádza do *Standby-I*. Je dôležité, aby sa vysielač nenachádzal v tomto móde viac ako 4 ms.

Prenosová rýchlosť

Modul môže operovať v troch rôznych rýchlostiach: 250 kbps, 1 Mbps a 2 Mbps. Nižšie rýchlosti poskytujú vyššiu stabilitu prenosu, ale zároveň zvyšujú spotrebu energie. Používanie vyšších rýchlostí znižuje pravdepodobnosť kolízií.

Rýchlosť sa nastavuje v *RF_SETUP* registri. Vysielač aj prijímač musia byť konfigurované na rovnakú rýchlosť.

Intenzita

Sila vysielaného signálu sa dá nastaviť v štyroch rôznych intenzitách v *RF_SETUP* registri. Intenzita priamo ovplyvňuje spotrebu energie.

Enhanced ShockBurst

Enhanced ShockBurst je vstavaná vrstva dátového spojenia, ktorá sa stará o paket, časovanie, automatické odpovede a preposielanie chybných paketov. Táto vrstva výrazne zvyšuje efektivitu prenosu bez pridania komplexnosti.

Preambula

Preambula je bitová sekvencia používaná na synchronizovanie demodulátora prijímača a prichádzajúceho streamu. Preambulu tvorí jeden bajt striedajúcich sa jednotiek a núl. To zaručuje dostatočné množstvo prechodov logických úrovní na stabilizáciu prijímača.

Adresa

Adresa zaručuje že, paket je detegovaný a prijatý správnym prijímačom. Dĺžka adresy je konfigurovateľná na 3, 4 alebo 5 bajtov.

Packet control field

Packet control field obsahuje informácie, ktoré sú prevažne využívane Enhanced Shockburst protokolom, a to sú informácie o veľkosti prenášaných dát, PID, Acknowlegment flag. Informácie o veľkosti prenášaných dát sa používajú iba v prípade, keď je povolená dynamická veľkosť dát. PID označuje, či je paket nový alebo preposielaný. Acknowledgment flag určuje, či má byť odoslaná odpoveď na prijaté dáta. Táto funkcia musí zároveň byť povolená v konfiguračných registroch.

Payload

Payload je časť paketu ktorá, prenáša samotné dáta. Enhanced ShockBurst podporuje dva druhy spracovania dát. Prvým je statická dĺžka. Pri použití statickej dĺžky dát musia byť prijímač aj vysielač nakonfigurované na rovnakú dĺžku. Táto dĺžka sa nastavuje v registri *RX_PW_Px* prijímača a na strane vysielača je definovaná dĺžkou nasunutých dát do FIFO.

Dynamická dĺžka dát umožňuje posielať dáta menšie ako 32 bajtov, čo zvyšuje celkovú rýchlosť prenosu.

CRC

CRC alebo Cyclic Redundancy Check sa využíva ako overovací mechanizmus. Môže byť 1 alebo 2 bajtový a je vypočítaný z posielaných dát.

Dĺžka CRC sa nastavuje v *CONFIG* registri. V prípade, že overenie CRC zlyhá, paket je zahodený [6].

2.7.3 SPI

Serial Peripheral Interface je synchrónna sériová komunikácia určená na krátku vzdialenosť. Pôvodne vyvinutá spoločnosťou Motorola a dnes už prakticky štandard. Dáta sú prenášané obojsmerne na samostatných vodičoch v master-slave architektúre. Komunikácia s viacerými slave zariadeniami je možná pomocou pinu CE alebo SS.

Pre začatie prenosu dát master nakonfiguruje hodinový signál v rozmedzí 1 – 10 MHz v závislosti od slave zariadenia. Master potom vyberie zariadenie nastavením logickej nuly na pin CS. Počas jedného hodinového cyklu vždy prebehne obojsmerná komunikácia a to platí aj v prípade jednosmerného príkazu.

Posielané dáta zvyčajne pozostávajú z 8 bitového slova, ktoré je vysúvané z posuvného registra najvýznamnejším bitom ako prvým na pine MOSI a zároveň do toho

istého registra sú nasúvané dáta z pinu MISO. Dáta v posuvných registroch sa spracujú, a v prípade, že sú potrebné ďalšie dáta, proces sa opakuje. Po skončení prenosu master prestane generovať hodinový signál a odznačí slave zariadenie nastavením príslušného pinu CS na logickú úroveň 1.

Okrem zvolenia vhodnej frekvencie musí MCU byť nastavené na správnu polaritu a fázu vzhľadom na dáta. Tieto nastavenia sú konfigurované dvomi bitmi CPOL a CPHA, pričom CPOL označuje aktívnu fázu hodinového cyklu, a CPHA určuje, či sú dáta aktívne na nábežnú alebo zostupnú hranu hodinového signálu.

Okrem bežných štyroch signálov majú SPI zariadenia niekedy aj prerušovací signál, ktorý upozorní MCU že je vyžadovaná komunikácia. Prerušenia nie sú definované v SPI štandarde, ale nie sú ani zakázané [8].

3 RIEŠENIE

3.1 Komunikácia s modulom NRF24L01+

Komunikácia s bezdrôtovým modulom prebieha pomocou SPI príkazov. Posielané príkazy pozostávajú z 8bitovej inštrukcie a dát vždy s najvýznamnejším bitom ako prvým. Celkovo modul rozoznáva 11 inštrukcií. Najpodstatnejšie inštrukcie sú *R_REGISTER* a *W_REGISTER*, ktoré zabezpečujú zápis a čítanie z registrov. Samotné príkazy sú 000X XXXX a 001X XXXX, kde na bitoch 3-8 sa nachádza adresa zvoleného registra. Ďalšie dve dôležité inštrukcie sú *R_RX_PAYLOAD* (0x61) a *W_TX_PAYLOAD* (0xA0) pomocou ktorých sa čítajú a zapisujú samotné prenášané dáta. Po zapísaní inštrukcie nasleduje 1 až 32 bajtov dát, ktoré sa zapíšu alebo prečítajú z príslušného FIFO registra. Príkazy *FLUSH_TX* a *FLUSH_RX* zabezpečia vyprázdnenie príslušných FIFO registrov. Pomocou príkazu *REUSE_TX_PL* je možné znovu odoslať predchádzajúcu správu bez toho, aby bola znovu posielaná cez SPI, správa však nesie byť vymazaná pomocou príkazu *FLUSH_TX*.

Modul má celkovo 28 registrov, ku ktorým je možné pristupovať pomocou príkazov *R_REGISTER* a *W_REGISTER*, prípadne pomocou špeciálnych príkazov určených pre konkrétny register.

Register *CONFIG* na adrese 0x00 obsahuje základné nastavenia modulu ako predvolený mód, nastavenia CRC kontroly paketu a IRQ pinu. Register *RF_SETUP* na adrese 0x06 uchováva nastavenia týkajúce sa samotného vysielania, teda prenosovú rýchlosť a výstupné zosilnenie signálu. *STATUS* register obsahuje informácie o dátach, ktoré sú momentálne dostupné. Popis všetkých registrov je dostupný v prílohe A [6].

Komunikácia s modulom prebieha pomocou zabudovanej SPI jednotky v mikrokontroléri. Pre prehľadnosť túto jednotku nastavuje funkcia *spi_init()*, ktorá nastaví registre MCU na požadované hodnoty.

```
SPCR |= _BV(MSTR); // Nastavenie MCU ako master
SPCR |= _BV(SPE); // Povolenie SPI
SPCR &= ~_BV(CPOL); // Clock Polarity CPOL = 0
SPCR &= ~_BV(CPHA); // Clock Phase CPHA = 0

// Nastavenie frekvencie
SPCR &= ~_BV(SPR0);
SPCR &= ~_BV(SPR1);
SPSR |= _BV(SPI2X);

SPCR &= ~_BV(DORD); // najvýznamnejší bit ako prvý
```

Samotné komunikovanie cez SPI potom prebieha pomocou funkcie *spi_transfer()*, ktorá nahrá do posuvného registra požadované dáta a zabezpečí ich vysunutie. Výstup funkcie sú dáta nasunuté do posuvného registra zo slave zariadenia.

```
static uint8_t spi_transfer(uint8_t data) {
    SPDR = data; // Začiatok prenosu
    while (!(SPSR & _BV(SPIF))); // Počkať na dokončenie
    return SPDR; // Vrátiť prijaté dáta
}
```

Priamo komunikáciu s modulom NRF24L01+ zabezpečuje funkcia *nRF24L01_send_command()* a pomocné funkcie, ktoré sú pre jednoduchosť určené pre konkrétny príkaz napríklad *nRF24L01_read_register()* a *nRF24L01_write_register()*.

```
uint8_t nRF24L01_send_command(nRF24L01 *rf, uint8_t command, void *data,
size_t length) {
    set_low(rf->ss);
    rf->status = spi_transfer(command); // Odoslanie príkazu, vrátená
hodnota je vždy STATUS register

    for (unsigned int i = 0; i < length; i++) // Odosielanie dát
        ((uint8_t*)data)[i] = spi_transfer(((uint8_t*)data)[i]);
    set_high(rf->ss);
    return rf->status;
}

uint8_t nRF24L01_write_register(nRF24L01 *rf, uint8_t reg_address, void
*data, size_t length) { // Pomocná funkcia na zápis do registra

    return nRF24L01_send_command(rf, W_REGISTER | reg_address, data,
length);
}

uint8_t nRF24L01_read_register(nRF24L01 *rf, uint8_t reg_address, void
*data, size_t length) { // Pomocná funkcia na čítanie registra

    return nRF24L01_send_command(rf, R_REGISTER | reg_address, data,
length);
}
```

Po pripojení napájania je vždy potrebné nastaviť požadované registre pre zaistenie funkčnosti komunikácie. Všetky bezdrôtové moduly musia byť nastavené rovnako, aby boli schopné medzi sebou komunikovať. Najdôležitejšie je nastavenie požadovanej frekvencie a rýchlosti prenosu dát. V tomto prípade je potrebné vypnúť aj vstavanú funkciu Enhanced Shockburst vypnutím auto ACK. Tento krok umožní vytvoriť vlastnú vrstvu komunikácie, ktorá bude podporovať viac ako 6 zariadení komunikujúcich medzi sebou. Vykonať tieto zmeny je vďaka pomocnej funkcii *nRF24L01_write_register()* jednoduchý proces.

```
uint8_t data;

// Povolenie CRC, nastavenie módu Standby-I
data = _BV(EN_CRC) | _BV(CRCO) | _BV(PWR_UP) | _BV(PRIM_RX);
nRF24L01_write_register(rf, CONFIG, &data, 1);

// Vypnutie auto ACK
data = 0x00;
nRF24L01_write_register(rf, EN_AA, &data, 1);

// Nastavenie kanála
data = 0x4C;
nRF24L01_write_register(rf, RF_CH, &data, 1);

// Nastavenie rýchlosti na 2 MBPS
data = _BV(0) | _BV(1) |_BV(2) | _BV(RF_DR_HIGH);
nRF24L01_write_register(rf, RF_SETUP, &data, 1);

// Vypnutie dynamic payload
data = 0x00;
nRF24L01_write_register(rf, DYNPD, &data, 1);
nRF24L01_write_register(rf, FEATURE, &data, 1);

// Zapnúť pipe 0 a 1
data = _BV(0) | _BV(1);
nRF24L01_write_register(rf, EN_RXADDR, &data, 1);

// Nastaviť payload dĺžku 32 bajtov
data = 0x20;
nRF24L01_write_register(rf, RX_PW_P0, &data, 1);
```

Pre zabezpečenie základnej bezdrôtovej komunikácie je okrem inicializačných funkcií potrebné vždy nastaviť bezdrôtový modul do správneho módu a v neposlednom rade zabezpečiť prenos dát medzi MCU a modulom. Funkcie *nRF24L01_start_listen()* a *nRF24L01_stop_listen()* nastavujú prechod medzi módmi *Standby-I* a *RX*. Na prechod do módu RX je potrebné nastaviť bit *RX_PRIM* v registri *CONFIG* a zároveň nastaviť logickú jednotku na pin CE [9].

```
void nRF24L01_start_listen(nRF24L01 *rf) {
    uint8_t config;
    nRF24L01_read_register(rf, CONFIG, &config, 1);

    config |= _BV(PRIM_RX); // PRIM_RX = 1
    nRF24L01_write_register(rf, CONFIG, &config, 1);

    set_high(rf->ce); // Prechod do RX
}

void nRF24L01_stop_listen(nRF24L01 *rf) {
    set_low(rf->ce); // Prechod do Standby-I
}
```

Pre prechod naspäť do módu *Standby-I* stačí nastaviť logickú nulu na pin CE. Čítanie a zápis správ z a do FIFO kontrolujú funkcie *nRF24L01_read_recieved_data()* a *nRF24L01_transmit()*.

```
bool nRF24L01_read_received_data(nRF24L01 *rf, char *message) {
    nRF24L01_clear_receive_interrupt(rf);
    char mess[32];
     // Čítanie správy z FIFO
    nRF24L01_send_command(rf, R_RX_PAYLOAD, &mess, 32);
    memcpy(message, mess, 32);
    return true;
}

void nRF24L01_transmit(nRF24L01 *rf, void *msg) {
    // V prípade že je nastavený IRQ resetovať hodnotu
    nRF24L01_clear_transmit_interrupts(rf);

     // Správa do FIFO
    nRF24L01_send_command(rf, W_TX_PAYLOAD, msg, 32);

    uint8_t config;
    nRF24L01_read_register(rf, CONFIG, &config, 1);

    config &= ~_BV(PRIM_RX); // PRIM_RX = 0
    nRF24L01_write_register(rf, CONFIG, &config, 1);

    set_high(rf->ce);
    _delay_us(11); // CE impulz > 10us Prechod do TX
    set_low(rf->ce);
}
```

3.2 Komunikačné rozhranie

Pretože vstavaný komunikačný protokol Enhanced Shockburst ktorý podporuje komunikáciu iba 6 zariadení je deaktivovaný je potrebné použiť vlastný protokol. Nový protokol pracuje na rovnakom princípe, ale na rozdiel od vstavaného je potrebné komunikáciu riadiť pomocou MCU, prináša však výhodu použitia až 255 zariadení. Princíp komunikácie spočíva v tom, aby odosielateľ vedel, že prijal správu v poriadku. Na takéto overenie slúžia ACK správy. Ďalší problém spočíva v tom, že aj ACK správa sa môže stratiť, a preto príjemca musí vedieť identifikovať správy a rozoznať, či danú správu už spracoval.

Payload

Na prenos všetkých informácií potrebných na správne fungovanie nového komunikačného protokolu musí byť použitý 32 bajtový payload, keďže nie je možné zasahovať do procesu tvorenia samotného paketu. Na prenos týchto informácií bola navrhnutá dátová štruktúra ktorá, kombinuje všetky potrebné informácie a samotnú správu do 32 bajtového slova.

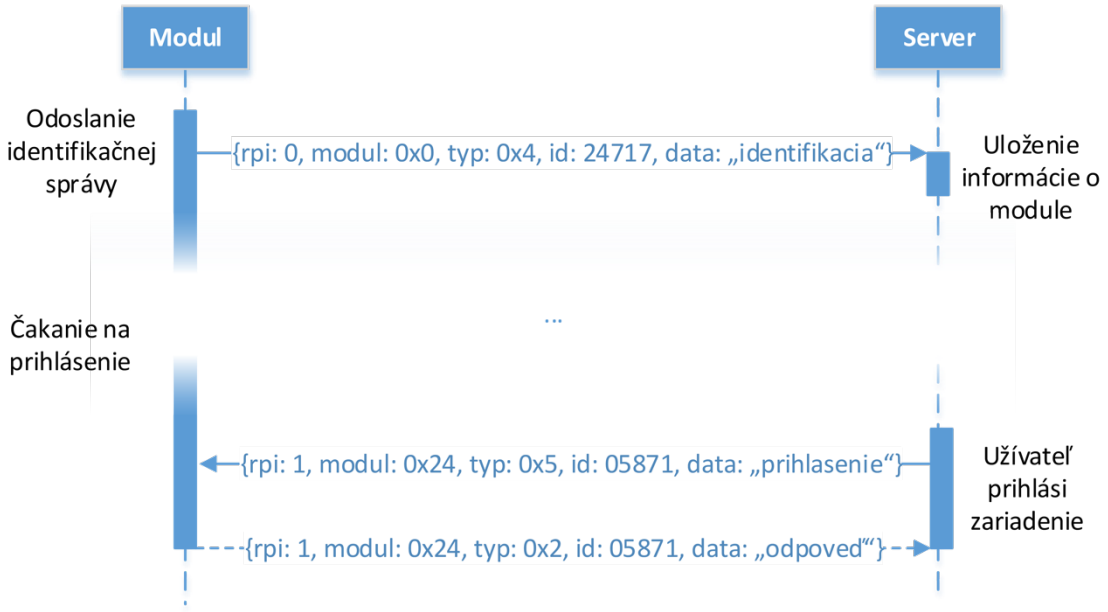
```
typedef struct {
    unsigned rpi   : 4;  // Identifikátor servera
    unsigned modul : 8;  // Identifikátor modulu
    unsigned id    : 12; // Identifikátor správy
    unsigned type   : 8; // Typ správy
    char mes[28];          // Prenášané dáta
} payload_data;
```

Táto štruktúra obsahuje 4 bitovú identifikáciu servera a 8 bitovú identifikáciu modulu, čo umožňuje používať až 15 serverov s 255 modulmi. ID správy slúži na identifikáciu správ. Každá správa má náhodne vygenerované číslo správy odosielateľom, takýmto spôsobom odosielateľ vie priradiť odpoveď k odoslanej správe, identifikátor slúži tiež pre príjemcu, ktorý takto vie odlíšiť nové a opakované správy. Typ správy sprehľadňuje komunikáciu tak, že je jednoduché určiť, ktorým smerom je správa vysielaná a aký je jej účel.

Tab. 3.1: Typy správ komunikačného protokolu

Kód	Popis	Smerovanie
0x0	Príkaz zo servera	RPi -> Modul
0x1	Odpoveď zo servera	RPi -> Modul
0x2	Správa z modulu	Modul -> RPi
0x3	Odpoveď z modulu	Modul -> RPi
0x4	Identifikácia	Modul -> RPi
0x5	Pripojenie do siete	RPi -> Modul
0x6	Požiadavka identifikácie	RPi -> RPi
0x7	Identifikácia	RPi -> RPi

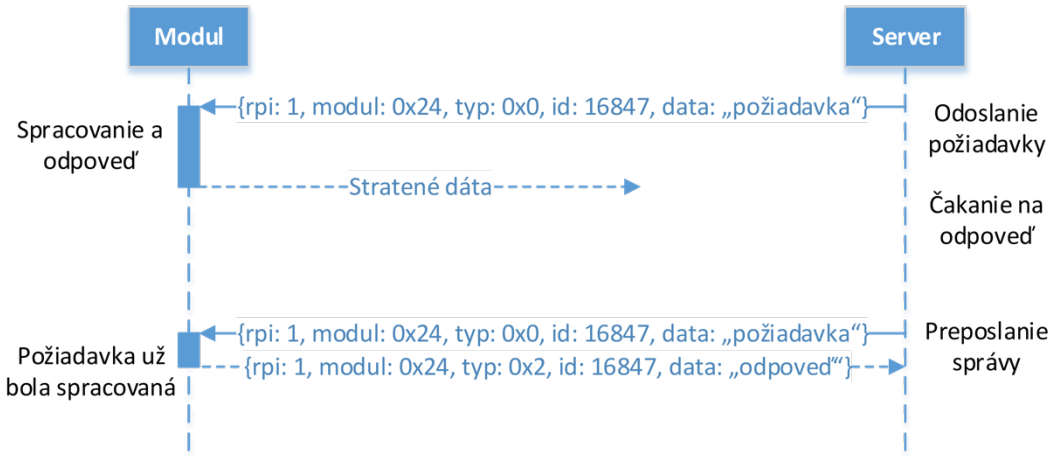
Komunikácia je až na výnimočné prípady inicializovaná serverom. Ak na serveri vznikne požiadavka na odoslanie dát modulu, je zavolaná funkcia *send_data()*. Tá zostaví štruktúru payload_data, kde identifikátor modulu je kód konkrétneho zariadenia ktorému sú dáta určené, id je náhodne vygenerované číslo a typ správy je 0x00. Po odoslaní je nastavený mód *RX* a čaká sa odpoveď. V prípade, že správa nedorazí do určitého času, je odoslaná nová správa s totožným obsahom. Tento proces sa opakuje, až kým nie je doručená odpoveď, prípadne kým sa nevyčerpá počet pokusov o pripojenie. V prípade, že príde odpoveď, overí sa hlavička správy a teda identifikátor servera, identifikátor modulu, id správy a odpovedajúci typ správy. Po kladom overení sú dáta predané serveru.



Obr. 3.1: Identifikácia a prihlásenie modulu

Dáta sú vysielané na spoločnom kanáli, preto príjemca musí prijatú správu pred spracovaním overiť. Kontroluje sa identifikátor servera a modulu. Modul môže prijímať správy iba od jedného servera, zatiaľ čo server musí komunikovať s viacerými modulmi. V prípade, že je správa overená sú dáta predané hlavnému programu, na správu sa odpovie ACK správou a id správy je uložené. Ak sa ACK správa stratí, odosielateľ opakovane posiela rovnakú správu. Prijímateľ porovnáva id správy s predchádzajúcou a pokiaľ sú zhodné odošle ACK správu, ale dáta zahodí.

Zvláštny prípad nastáva pri inicializácií zariadení. Keďže modul nepozná komunikačné parametre, vysiela signál s určitým náhodným časovým odstupom. V tejto identifikačnej správe je uvedený typ zariadenia a jeho náhodne vygenerované dočasné identifikačné číslo. Server túto správu prijme a uloží do databázy. Potvrdenie prijatia zariadenia je podmienené užívateľským vstupom, až po potvrdení je k zariadeniu vyslaná inicializačná správa, ktorá obsahuje jeho dočasné identifikačné číslo, nové id a tiež nové kanály, na ktorých od tohto momentu bude komunikovať.



Obr. 3.2: Priebeh komunikácie s naznačením stratenej správy

$$T = \frac{8 \cdot (preamble + adress + payload + CRC)}{data\ rate}$$

$$T = \frac{8\left[{}^{bit}/_{byte}\right] \cdot (1[byte] + 5[byte] + 32[byte] + 2[byte])}{2 \cdot 10^{6}[{}^{bit}/_{s}]} \tag{3.1}$$

Podľa výpočtu (3.1)[6] trvá odoslanie jednej správy približne 150 us, je však potrebné počítať aj s ďalšími časovými faktormi, ako je trvanie prenosu informácií medzi MCU a bezdrôtovým modulom ktorý, môže dosiahnuť v prípade, že MCU používa interný oscilátor s deličkou kmitočtu až 1ms (3.2)[6]. Vzhľadom na tieto časy bol zvolený čas čakania na odpoveď 4ms.

$$T = \frac{8 \cdot payload}{SPI\ data\ rate}$$

$$T = \frac{8[{}^{bit}/_{byte}] \cdot 32[byte]}{500 \cdot 10^{3}} \tag{3.2}$$

```
if(round > 0) { // V prípade opakovaného odosielania iba preposlať správu
    // preposlanie predchádzajúcej správy
    nRF24L01_retransmit_message(rf);
} else {
```

```
    // V prípade prvej správy zostaviť hlavičku
    send.rpi = rpi;          // adresa servera
    send.modul = address;    // adresa modulu
    send.type = 0x02;        // typ správy odosielanie z modulu
    send.id = id;            // náhodne vygenerované id správy
    memcpy(send.mes, data, 28);
    nRF24L01_transmit(rf, &send); // odoslanie správy
}

_delay_us(400); // čas potrebný k prechodu do módu TX a odoslaniu správy
nRF24L01_start_listen(rf);    // prechod do módu RX
int wait = rand() % 40+40;    // čakanie 4ms + náhodný čas 0-4ms
int i = 0;

while (!(nRF24L01_data_received(rf) || i > wait)) {
    i++;
    _delay_us(100);    // stačí kontrolovať status každých 100us
}

nRF24L01_stop_listen(rf);// po timeoute alebo prijatí správy prechod do
Standby-I
```

3.3 Serverová aplikácia

Hlavná časť serverovej aplikácie je v jazyku JavaScript. Používa *node.js* framework založený na platforme Chrome V8 JavaScript engine, čo umožňuje aplikácii asynchrónne spracovanie viacerých požiadaviek pri zachovaní prehľadného kódu a jednoduchosti. Pre komunikáciu s mobilnou aplikáciou sa používa nadstavba *Socket.io* ktorá pridáva do systému obojstrannú komunikáciu v reálnom čase. Komunikáciu s bezdrôtovým modulom cez SPI mikropočítača zabezpečuje vlastná komunikačná knižnica, ktorá využíva ďalšie voľne dostupné knižnice.

Hlavný program využíva doplnky na komunikáciu s databázou MySQL, zápis a čítanie JSON súborov a nastavovanie a volanie časových udalostí.

Po spustení servera sú načítané základné informácie identifikácia servera a čísla komunikačných kanálov. V prípade, že sa jedná o prvé spustenie, sú tieto údaje vygenerované funkciami komunikačnej knižnice, ktoré zaručia unikátnosť serverového identifikačného čísla medzi okolitými rovnakými zariadeniami. Po úspešnej inicializácii je pripojená MySQL databáza a je načítaný zoznam známych typov zariadení.

Typ zariadenia je identifikovaný 1 bajtom, a teda je teoretický počet rôznych modulov obmedzený číslom 256. Zoznam typov zariadení sa nachádza v module *devices.json*. Každé zariadenie musí byť uvedené a špecifikované v tomto zozname. V zozname musí byť uvedené, o aký typ zariadenia sa jedná, jeho názov a zoznam vstupov a výstupov. Pri každom vstupe a výstupe musí byť jeho názov, informácia hodnotách, aké nadobúda a poradie prvého bajtu v správe, kde sa nachádza informácia o jeho hodnote.

```
{"devices": [
    {
      "type" : "a",
      "name" : "Rele modul",
      "outputs": [ {
          "name" : "Pin 1",
          "type" : "switch",
          "min_value" : 0,
          "max_value" : 1,
          "byte" : 0
        },
  ....
      ],
      "sensors" : [ {
          "name" : "Teplotný senzor",
          "type" : "request",
          "unit" : "°C",
          "value_type" : "int",
          "byte" : 8
        },
  ....
        ]
    }
}
```

Takýmto spôsobom je možné nakonfigurovať ľubovoľné koncové zariadenie, ktoré bude schopné komunikovať so serverovou aplikáciou bez programových zmien v samotnej serverovej aplikácií. Tento systém samozrejme prináša množstvo nevýhod v tom, že obmedzuje spôsob, akým môžu koncové moduly pracovať, avšak pre použitie v domácej automatizácii sú hlavne vyžadované jednoduché jednosmerné príkazy.

Po úspešnom načítaní databázy a zoznamu modulov server začne počúvať na určenom porte a nastaví podľa aktuálneho času všetky plánované udalosti podľa dát z databázy.

Princíp komunikácie medzi serverom a klientom spočíva v emitovaní a poslúchaní na eventy. Eventy môže emitovať ako klient, tak aj server. Každý event môže obsahovať ľubovoľné množstvo dát.

```
socket.on('change_output', function(data, res) {
  changeValue(data, data.value, function(err, response) {
        res(response);
  });
});
```

Funkcia *socket.on()* čaká na event „change_output“ od klienta a po prijatí je zavolaná funkcia *changeValue()* ktorá je zodpovedná za odoslanie požiadavky určenému modulu a jej výstup je odpoveď o úspešnosti vykonania príkazu. V prípade, že výstup bol úspešne zmenený, funkcia emituje event „output_update“ ktorý informuje ostatných práve pripojených klientov o zmene výstupu, a tí tak majú túto informáciu v reálnom čase

zobrazenú v mobilnej aplikácií. Rovnaká funkcia je volaná aj v prípade plánovaných udalostí, ktoré nie sú volané užívateľom cez mobilnú aplikáciu, ale časovým odpočtom, prípadne vstupom niektorého z iných modulov.

3.4 Mobilná aplikácia

Užívateľskú interakciu so systémom zabezpečuje mobilná aplikácia ktorá komunikuje so serverom pomocou nadstavby *socket.io*. Mobilné aplikácie pre platformy iOS a Android sú vyvíjané vo vývojovom prostredí Xcode od spoločnosti Apple a Android Studio spoločnosti Google.

iOS aplikácia

Mobilná aplikácia pre platformu iOS je naprogramovaná v relatívne novom programovacom jazyku Swift rovnako od spoločnosti Apple. Programovací jazyk Swift založený na Objective-C je určený na tvorenie aplikácii pre platformy iOS, macOS, watchOS, tvOS, a tiež Linux. Na platformách spoločnosti Apple používa behovú knihovňu Objective-C čo umožňuje použitie jazyka C, Objective-C, C++ a Swift v jednom programe.

Android aplikácia

Aplikácia pre Android využíva programovací jazyk Java ktorý, je bežný pre aplikácie určené pre platformu Android. Java je na rozdiel od jazyka Swift staršia, jej počiatky siahajú do roku 1991. Jazyk Java je tiež založený na syntaxe jazykov C, vďaka čomu je podobný Jazyku Swift a základné princípy sú podobné. Táto pod podobnosť však zľahčuje vývoj pre obidve platformy minimálne.

Knižnice používané pri vývoji aplikácie sú dostupné pre obidve platformy, aby sa predišlo rozdielom v aplikáciách. Hlavná knižnica je *Socket.io*, ktorá zabezpečuje komunikáciu so serverom. Na zobrazovanie grafov v aplikácii je použitá knižnica Charts, respektíve MPAndroidChart.

Po spustení aplikácia vyhľadá na lokálnej sieti server, pripojí sa a načíta zo servera štruktúru ovládacích prvkov. Tie sú rozdelené do miestností, pričom každá miestnosť môže zobrazovať vstupy zo senzorov, výstupy jednotlivých zariadení a udalosti, na ktoré je systém schopný reagovať. Všetky tieto prvky je užívateľ schopný nastaviť podľa svojich preferencií. Aplikácia je rozdelená do takzvaných tabov, pričom každý poskytuje užívateľovi iné prvky. Prvé dva taby sú pre užívateľa najpodstatnejšie, pretože pomocou nich je možné samotné ovládanie systému. Prvý tab združuje užívateľom vybrané obľúbené ovládače pre rýchly prístup po spustení aplikácie. V druhom tabe sú tieto ovládače rozdelené do jednotlivých miestností. V tretom tabe je možné zobraziť výstup z kamery v prípade, že je dostupná. Štvrtý tab sa používa na správu zariadení. Je možné

priradzovať a mazať zariadenia podľa potreby.

Ovládače a senzory

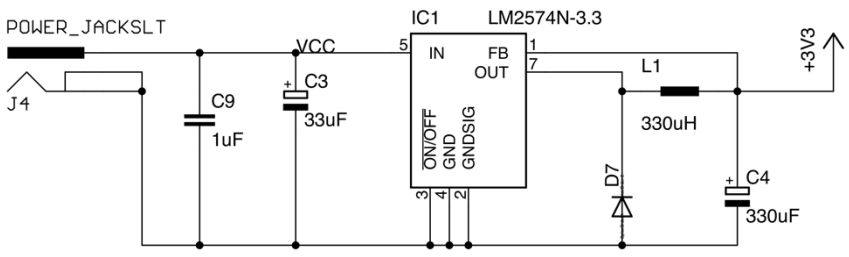
Každý senzor, ktorý je v aplikácii priradený do miestnosti alebo medzi obľúbené, zobrazuje aktuálnu hodnotu. Ďalej je možné dotykom zobraziť graf s približným vývojom hodnoty. Graf je možné zobraziť aj pri pohybovom senzore, kde sa zobrazuje počet spustení senzora každú hodinu, a miesto aktuálnej hodnoty sa zobrazuje čas, ktorý ubehol od posledného spustenia.

Výstupy je možné ovládať tromi druhmi ovládačov. Switch je jednoduchý prepínač, ktorý prepína stavy zapnuté vypnuté, číselné hodnoty týchto stavov musia byť definované v súbore *devices.json*. Slider je druhý typ ovládača, ktorý je vhodný pre výstupné zariadenia, ktoré môžu mať viac hodnôt, napríklad PWM modulácia výstupu pre LED svietidlá v LED module. Špeciálny typ ovládača je ovládač teploty, ktorým je možné nastaviť požadovanú teplotu. Kompletný návod na obsluhu je v prílohe D.

3.5 Prototypy zariadení

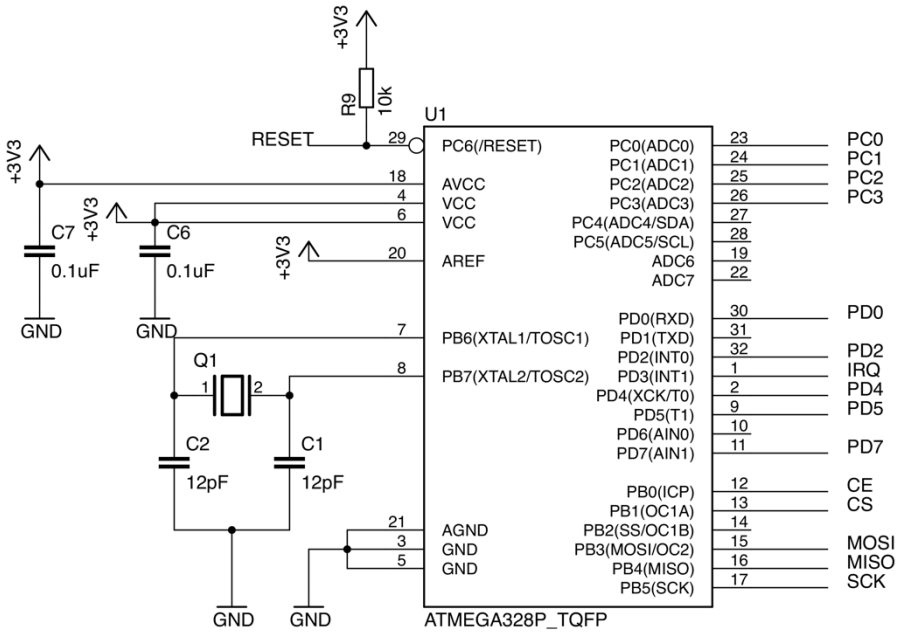
V rámci práce boli podľa požiadaviek navrhnuté a vyrobené štyri prototypy modulov, ktoré sú schopné komunikovať so serverom. Každý modul plní inú funkciu a je zadefinovaný v súbore *devices.json*. Všetky moduly majú rovnakú základnú štruktúru a líšia sa iba v zapojení výstupných pinov mikrokontroléra.

Ako zdroj je použitý spínavý regulátor napätia LM2574N-3.3 v puzdre DIP-8. Napäťový regulátor dosahuje účinnosť 72 % pri vstupnom napätí 12 V. Schéma zapojenia je prevzatá z dokumentácie od výrobcu, pričom boli použité filtračné kondenzátory s mierne vyššou kapacitou aby bolo zabezpečené stabilné napätie v prípade náhlej prúdovej špičky [7].



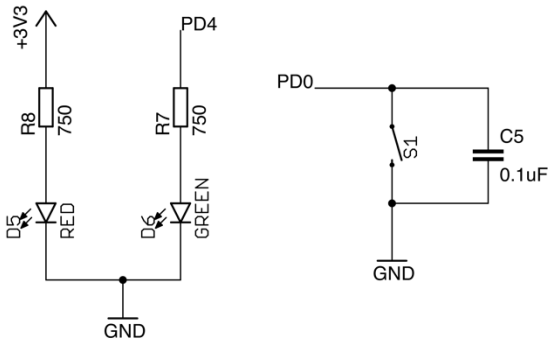
Obr. 3.3: Schéma zapojenia zdroja

Ako riadiace MCU je použitý mikrokontrolér Atmega328, ktorý ponúka dostatočný výkon a pamäť pri zachovaní relatívne nízkej spotreby energie. Nachádza sa v štandardnom zapojení s blokovými kondenzátormi pri napájaní. V návrhu sa počíta aj s prípadným externým oscilátorom, ktorý ale nie je použitý, keďže na stabilný beh aplikácie postačuje interný RC oscilátor s frekvenciou 8 MHz.

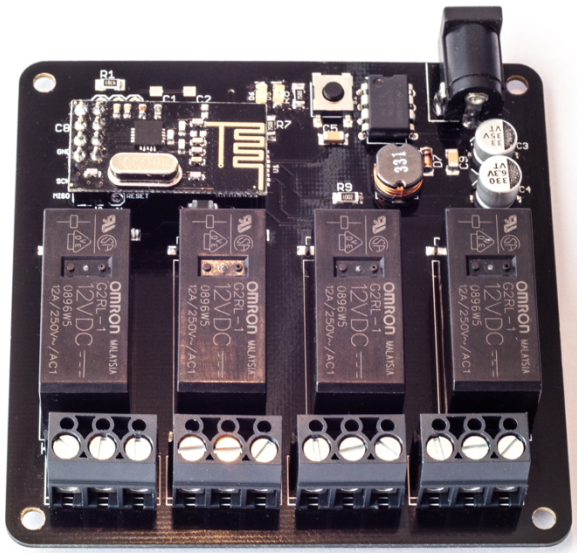


Obr. 3.4: Schéma zapojenia MCU

Dva vstupno-výstupné piny sú použité ako LED indikujúca beh programu a resetovacie tlačidlo, ktoré zaručuje premazanie internej pamäte EEPROM.

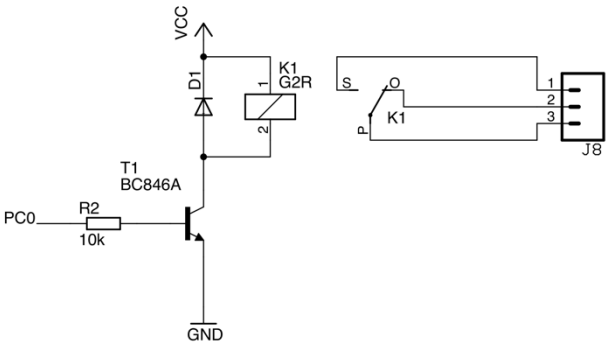


Obr. 3.5: Schéma zapojenia status LED a reset tlačidla

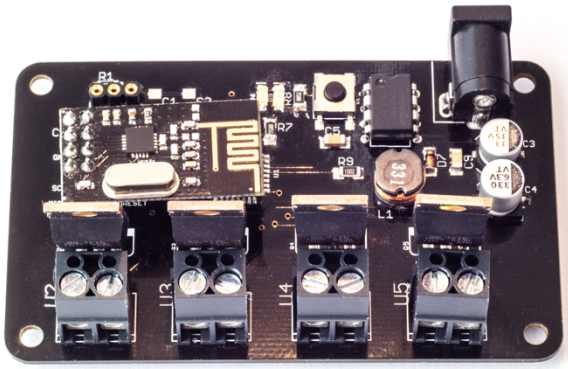


Obr. 3.6: RELÉ modul

V relé ovládači sú použité štyri relé na všeobecné použitie R2RL-1 12DC od výrobcu OMRON. Tieto relé sú schopné spínať až 250 V AC s prúdovým zaťažením 12 A. Relé sú spínané bežným NPN tranzistorom, a preto je paralelne k cievke pripojená dióda, ktorá ochráni tranzistor pri vypínaní.

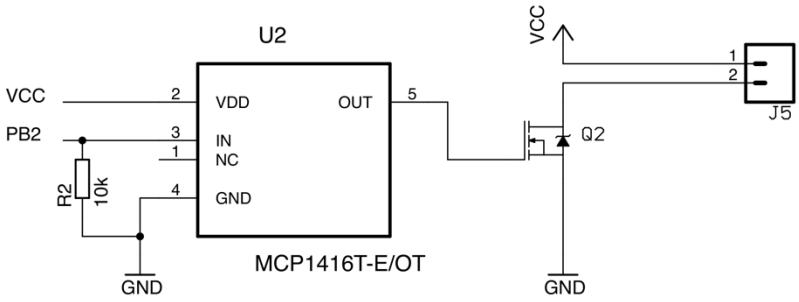


Obr. 3.7: Schéma zapojenia RELÉ

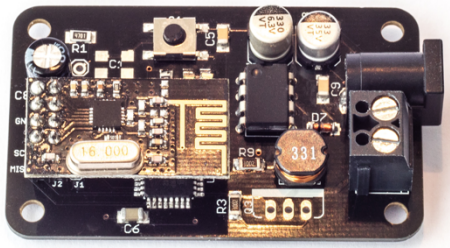


Obr. 3.8: LED modul

LED modul obsahuje štyri výstupné MOSFET tranzistory s indukovaným N kanálom. Pre zabezpečenie správnej funkcie aj v prípade rýchleho PWM sú použité MOSFET budiče MCP1416T.



Obr. 3.9: Schéma zapojenia MOSFET



Obr. 3.10: Modul radiátor

a) b)

Obr. 3.11: Modul senzor a) bez PIR senzora b) s PIR senzorom

4 ZÁVER

Na začiatku práce bolo potrebné sa zoznámiť s technológiami, ktoré je možné použiť alebo sa bežne používajú pri tomto type zariadení. Vybraný modul NRF24l01+ je veľmi jednoduchý na použitie v navrhnutom systéme a poskytuje rozumný kompromis medzi cenou, rýchlosťou prenosu dát a spotrebou elektrickej energie.

Ďalej boli určené požiadavky na systém vyplývajúce z možností použitej bezdrôtovej technológie, bol navrhnutý teoretický návrh ovládača a jeho modulov. Riadiaca jednotka je podľa zadania založená na platforme Raspberry Pi, ktoré poskytuje dostatočný výpočtový výkon pre hlavný program, a taktiež potrebné periférie, ako sú internetové pripojenie na komunikáciu s mobilnou aplikáciou, SPI pre komunikáciu s bezdrôtovým modulom. Okrem riadiacej jednotky systém pozostáva zo štyroch modulov, ktorých kombináciou je možné dosiahnuť ľubovoľný výsledok. Napríklad aktivovanie pohybového senzora spôsobí rozsvietenie svetla večer ale uprostred noci iba jemné LED podsvietenie.

Pri samotnom spracovaní bolo v prvom rade potrebné vytvoriť spoľahlivý komunikačný protokol pre bezdrôtovú komunikáciu modulov. Vytvorený protokol musí spĺňať určité podmienky na jeho funkcie, a preto podporuje komunikáciu 255 rôznych zariadení so serverom a zároveň je schopný vyhnúť sa kolíziám v prenose, spoľahlivo identifikovať odosielateľa aj príjemcu. K výhodám patrí aj schopnosť overiť doručenie správy a zároveň s potvrdením o prijatí správy preniesť dáta. Na vytvorenie spoľahlivého protokolu bolo potrebné najskôr spracovať komunikáciu s modulom NRF24L01+ cez SPI.

Súčasťou systému je aj mobilná aplikácia pre hlavné platformy iOS a Android. Aplikácia poskytuje užívateľovi jednoduché prostredie v ktorom môže celý systém ovládať a nastavovať. Okrem základných príkazov na ovládanie modulov je možné nastaviť udalosti, ktoré sledujú čas kedy sa majú vykonať alebo ich vykonanie závisí na vstupe z vybraného modulu.

Výsledkom práce je užívateľsky prívetivý systém na bezdrôtové ovládanie prevažnej časti elektrických zariadení v domácnosti. Výsledné moduly je možné kombinovať rôzne, prípadne vytvoriť úplne nové podľa potrieb. Napriek tomu, že cieľom práce bolo navrhnúť ovládač elektrických zariadení, bol tento systém doplnený o základné senzory, čo zvyšuje použiteľnosť. Systém navyše okrem sledovania dokáže na zmenu určitých senzorov reagovať.

Navrhnutý systém a prototypy modulov napriek vynaloženej snahe nie sú bezchybné. Počas testovania sa ukázalo niekoľko malých ale aj väčších detailov ktoré obmedzujú jeho použiteľnosť. Hlavný nedostatok v navrhnutom riešení je v zabezpečení. Pri navrhovaní sa z dôvodu prílišnej zložitosti nepočítalo so zabezpečením bezdrôtového prenosu medzi modulmi a serverom. V prípade, že útočník zachytí informácie o použitých komunikačných kanáloch môže do prenosu zasahovať.

Okrem zabezpečenia bezdrôtového prenosu by bolo vhodné pri ďalšom rozširovaní projektu umožniť užívateľovi vytvoriť viac používateľských účtov a spravovať ich práva. Ďalším vylepšením by mohlo byť umožniť užívateľovi nastavovať rozšírenejšie podmienky udalostí a postupne systém transformovať na systém inteligentnej domácnosti pridávaním ďalších funkcií a modulov.

LITERATÚRA

[1] Wi-Fi. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2016-12-14]. Dostupné z: https://en.wikipedia.org/wiki/Wi-Fi

[2] POTŮČEK, M. *Bezdrátový systém pro ovládání domácích spotřebi*čů *hlasem* . Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 64 s. Vedoucí diplomové práce Ing. Ondřej Sajdl, Ph.D.

[3] 5 Types of Wireless Technology For The IoT [online]. LinkLabs 2016 [cit. 2016-12-03]. Dostupné z: https://www.link-labs.com/types-of-wireless-technology/.

[4] GAST, Matthew. 802.11 wireless networks: the definitive guide. 2nd ed. Farnham: O'Reilly, 2005. ISBN 0596100523.

[5] The JSON Data Interchange Format [online]. Ecma International 2013 [cit. 2016-12-03]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[6] nRF24L01+ Single Chip 2.4GHz Transciever Product Specification [online] NORDIC SEMICONDUCTOR 2008 [cit. 2017-5-5] Dostupné z: https://www.nordicsemi.com/eng/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf

[7] LM2574x SIMPLE SWITCHER Product Specification [online] TEXAS INSTRUMENT 2016 [cit. 2016-5-5] Dostupné z: http://www.ti.com/lit/ds/symlink/lm2574.pdf

[8] Serial Peripheral Interface Bus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-05-06]. Dostupné z: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

[9] AVR-nRF24L01+. *AVR-nRF24L01+* [online]. [cit. 2017-05-06]. Dostupné z: https://github.com/antoineleclair/avr-nrf24l01

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

AES Advanced Encryption Standard

AP Access Point

FHSS Frequency Hopping Spread Spectrum

FIFO First In First Out

ISM Industrial, Scientific and Medical

LAN Local Area Network

PAN Personal Area Network

RC4 Rivest Cipher 4

SPI Serial Peripheral Interface

SSID Service Set Identifier

WEP Wired Equivalent Privacy

WPA Wi-Fi Protected Access

PWM Pulse Width Modulation

CRC Cyclic Redundancy Check

MCU Microcontroller Unit

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

A ZOZNAM SPI PRÍKAZOV

```
// SPI Commands
#define R_REGISTER          0x00 // 000A AAAA
#define W_REGISTER          0x20 // 001A AAAA
#define R_RX_PAYLOAD        0x61 // 0110 0001
#define W_TX_PAYLOAD        0xA0 // 1010 0000
#define FLUSH_TX            0xE1 // 1110 0001
#define FLUSH_RX            0xE2 // 1110 0010
#define REUSE_TX_PL         0xE3 // 1110 0011
#define R_RX_PL_WID         0x60 // 0110 0000
#define W_ACK_PAYLOAD       0xA8 // 1010 1PPP
#define W_TX_PAYLOAD_NOACK  0xB0 // 1011 0000
#define NOP                 0xFF // 1111 1111

// Register Map
#define CONFIG      0x00      // Konfiguračný register
#define MASK_RX_DR  6         // IRQ prerušenie pri prijatí správy
#define MASK_TX_DS  5         // IRQ prerušenie pri odoslaní správy
#define MASK_MAX_RT 4         // IRQ prerušenie pri naplnení FIFO
#define EN_CRC      3         // Aktivovanie CRC
#define CRCO        2         // DÍžka CRC
#define PWR_UP      1         // Power Up/Down
#define PRIM_RX     0         // Nastavenie módu RX/TX

#define EN_AA       0x01      // Register Auto acknowlegment
#define ENAA_P5     5
#define ENAA_P4     4
#define ENAA_P3     3
#define ENAA_P2     2
#define ENAA_P1     1
#define ENAA_P0     0

#define EN_RXADDR   0x02      // Aktivovanie RX kanálov
#define ERX_P5      5
#define ERX_P4      4
#define ERX_P3      3
#define ERX_P2      2
#define ERX_P1      1
#define ERX_P0      0

#define SETUP_AW    0x03      // Nastavenie dÍžky adresy
#define AW          0

#define SETUP_RETR  0x04      // Automatické odosielanie
#define ARD         4         // Oneskorenie
#define ARC         0         // Počet pokusov

#define RF_CH       0x05      // Nastavenie frekvencie

#define RF_SETUP    0x06      // RF Nastavenie
#define CONT_WAVE   7
#define RF_DR_LOW   5         // Nastavenie nízkej rýchlosti
#define PLL_LOCK    4
#define RF_DR_HIGH  3         // Nastavenie vysokej rýchlosti
#define RF_PWR      1         // Nastavenie vysielacieho výkonu
```

```
#define STATUS          0x07        // Status register
#define RX_DR           6           // Prijaté data vo FIFO
#define TX_DS           5           // Správa úspečne odoslaná
#define MAX_RT          4           // Vyčerpaný maximálny počet pokusov
#define RX_P_NO_MASK    0x0E        // Adresa prijatej správy
#define STATUS_TX_FULL  0           // Plné FIFO

#define OBSERVE_TX      0x08
#define PLOS_CNT        4           // Počet stratených packetov
#define ARC_CNT         0           // Počet preposlaných paketov

#define RPD             0x09        // Sila prijatého signálu

// Nastavenie adries pre jednotlivé kanály
#define RX_ADDR_P0      0x0A
#define RX_ADDR_P1      0x0B
#define RX_ADDR_P2      0x0C
#define RX_ADDR_P3      0x0D
#define RX_ADDR_P4      0x0E
#define RX_ADDR_P5      0x0F

#define TX_ADDR         0x10        // Adresa odosielania

// Nastavenie dĺžky správy pre jednotlivé kanály
#define RX_PW_P0        0x11
#define RX_PW_P1        0x12
#define RX_PW_P2        0x13
#define RX_PW_P3        0x14
#define RX_PW_P4        0x15
#define RX_PW_P5        0x16

#define FIFO_STATUS     0x17        // FIFO status
#define TX_REUSE        6           // Preposlanie poslednej správy
#define FIFO_TX_FULL    5           // Plné FIFO TX
#define TX_EMPTY        4           // Prázdne FIFO TX
#define RX_FULL         1           // Plné FIFO RX
#define RX_EMPTY        0           // Prázdne FIFO TX

#define DYNPD           0x1C        // Dynamické dĺžka správy
#define DPL_P5          5
#define DPL_P4          4
#define DPL_P3          3
#define DPL_P2          2
#define DPL_P1          1
#define DPL_P0          0

#define FEATURE         0x1D
#define EN_DPL          2
#define EN_ACK_PAY      1
#define EN_DYN_ACK      0
```

B FUNKCIE KOMUNIKAČNÉHO ROZHRANIA

```
nRF24L01 *comunication_init() {
      nRF24L01 *rf = nRF24L01_init();
      rf->ss.port = &PORTB;
          rf->ss.pin = PB1;
          rf->ce.port = &PORTB;
          rf->ce.pin = PB0;
          rf->sck.port = &PORTB;
          rf->sck.pin = PB5;
          rf->mosi.port = &PORTB;
          rf->mosi.pin = PB3;
          rf->miso.port = &PORTB;
          rf->miso.pin = PB4;
      nRF24L01_begin(rf);
      return rf;
}
bool send_data(nRF24L01 *rf, char *buff, uint8_t rpi, uint8_t address,
char data[28], int max_round) {

    payload_data send, recieved;

    int round = 0;
    bool again = true;

    int id = rand() % 50000+1; // id poziadavky

    nRF24L01_stop_listen(rf);

    while(again) {

        send.rpi = rpi; // adresa RPI
        send.modul = address; // adresa modulu
        send.type = 0x02; // odosielanie z modulu
        send.id = id;

        memcpy(send.mes, data, 28);

        nRF24L01_transmit(rf, &send);
        _delay_us(400);

        nRF24L01_start_listen(rf);

        int wait = 50;

        int i = 0;
        while (!(nRF24L01_data_received(rf) || i > wait)) {
            i++;
            _delay_us(100);
        }

        nRF24L01_stop_listen(rf);
        if (i < wait) {

            nRF24L01_read_received_data(rf, &recieved);
```

```
            if(recieved.id == id && recieved.type == 0x01) {
                again = false;
                strncpy(buff,recieved.mes, 28);

                nRF24L01_flush_transmit_message(rf);
                return true;
            }
        } else {
            round++;
            if(round > max_round) {
                again = false;
            }
        }
    }

    return false;
}

void recieve_data(nRF24L01 *rf, char *buff) {
    payload_data recieved, response;

    while(nRF24L01_data_received(rf)){
        nRF24L01_read_received_data(rf, &recieved);
    }

    if(recieved.modul == module.address && recieved.type == 0x00 &&
recieved.rpi == module.rpi) {

        response.rpi = recieved.rpi; // adresa RPI
        response.modul = module.address; // adresa cieloveho modulu
        response.type = 0x03; // odpoved
        response.id = recieved.id; // id poziadavky

        memcpy(response.mes, status, 28);

        nRF24L01_stop_listen(rf);
        nRF24L01_flush_transmit_message(rf);

        nRF24L01_transmit(rf, &response);
        _delay_us(400);

        if(recieved.id == last_message_id) {
            strncpy(buff,"0", 28);
        } else {
            strncpy(buff, recieved.mes, 28);
            last_message_id = recieved.id;
        }

    } else if (recieved.modul == module.address && recieved.type ==
0x05 && recieved.rpi == module.rpi) {

        response.rpi = recieved.rpi; // adresa RPI
        response.modul = module.address; // adresa modulu
        response.type = 0x03; // odosielanie z modulu
        response.id = recieved.id; // id poziadavky

        memcpy(response.mes, recieved.mes, 28);
```

```
        nRF24L01_transmit(rf, &response);
        _delay_us(400);

        strncpy(buff,"0", 28);
    } else {
        strncpy(buff,"0", 28);
    }

    nRF24L01_flush_recieve_message(rf);
    nRF24L01_flush_transmit_message(rf);

    nRF24L01_start_listen(rf);

    return;

}


bool module_init(nRF24L01 *rf) {
    eeprom data = get_eeprom();

    nRF24L01_set_recieve_address(rf, data.rx_pipe);
    nRF24L01_set_recieve2_address(rf, data.rx2_pipe);
    nRF24L01_set_transmit_address(rf, data.tx_pipe);

    if(!data.is_init) {
        int hash;
        if(!data.is_hash) {
            srand(_TIMESTAMP);
            hash = rand();
            data.is_hash = true;
            data.hash  = hash;

            set_eeprom(data);
        } else {
            hash = data.hash;
        }

        bool again = true;

        while(again) {


            payload_data send, recieved;

            send.rpi = 0x00; // adresa RPI
            send.modul = 0x00; // adresa modulu
            send.type = 0x04; // odosielanie z modulu
            send.id = rand() % 50000+1; // id poziadavky

            char temp[28];
            sprintf(temp, "a %08X", hash);
            memcpy(send.mes, temp, 28);

            nRF24L01_flush_transmit_message(rf);
            nRF24L01_transmit(rf, &send);

            _delay_us(400);
```

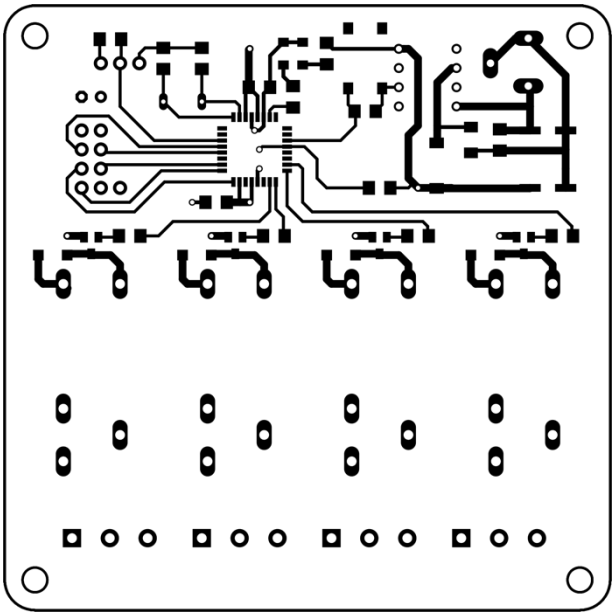
```
            nRF24L01_start_listen(rf);

            int i = 0;
            while (!(nRF24L01_data_received(rf) || i > 30000)) {
                i++;
                _delay_us(100);
            }

            nRF24L01_stop_listen(rf);
            if (i < 30000) {
                nRF24L01_read_received_data(rf, &recieved);

                if(recieved.type == 0x05) {

                    unsigned long recieved_hash  =
(long)strtol(recieved.mes, NULL, 16);
                    if(recieved_hash == hash) {

                        send.rpi = recieved.rpi; // adresa RPI
                        send.modul = recieved.modul; // adresa modulu
                        send.type = 0x03; // odosielanie z modulu
                        send.id = recieved.id; // id poziadavky
                        memcpy(send.mes, recieved.mes, 28);

                        nRF24L01_flush_transmit_message(rf);
                        nRF24L01_transmit(rf, &send);
                        _delay_us(400);
                        char pipe[2];

                        pipe[0] = recieved.mes[9];
                        pipe[1] = recieved.mes[10];
                        uint8_t rx_pipe = strtol(pipe, NULL, 16);

                        pipe[0] = recieved.mes[11];
                        pipe[1] = recieved.mes[12];
                        uint8_t tx_pipe = strtol(pipe, NULL, 16);

                        data.is_init = true;
                        data.rpi = recieved.rpi;
                        data.address = recieved.modul;
                        data.rx_pipe[0] = rx_pipe;
                        data.tx_pipe[0] = tx_pipe;

                        set_eeprom(data);

                        module.rpi = data.rpi;
                        module.address = data.address;

                        nRF24L01_set_recieve_address(rf,
data.rx_pipe);
                        nRF24L01_set_recieve2_address(rf,
data.rx2_pipe);
                        nRF24L01_set_transmit_address(rf,
data.tx_pipe);
                        again = false;
                    }
                }
            } else {
                again = true;
```

```
            }

        }
    } else {
        srand(data.srand);
        data.srand = rand() % 60000;
        set_eeprom(data);

        module.rpi = data.rpi;
        module.address = data.address;
    }

    return true;
}
```

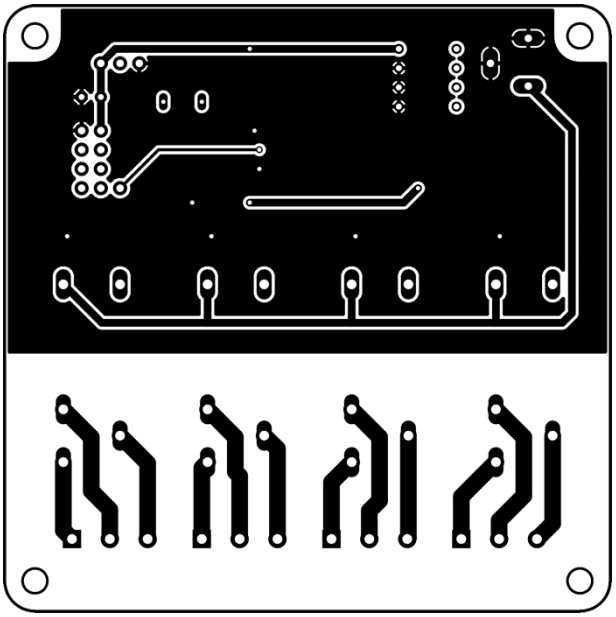
C NÁVRH ZARIADENÍ

C.5 Doska plošného spoja RELÉ modulu

Vrstva TOP



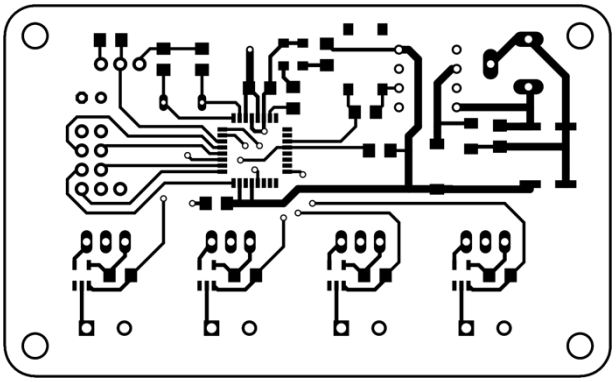
Vrstva BOTTOM



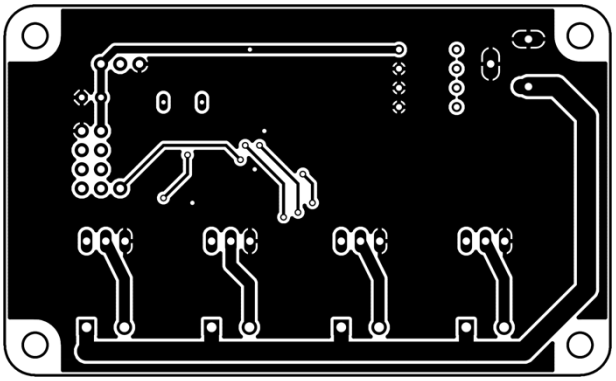
Rozmer dosky 80x80 [mm], mierka M1:1

C.6 Doska plošného spoja LED modulu

Vrstva TOP



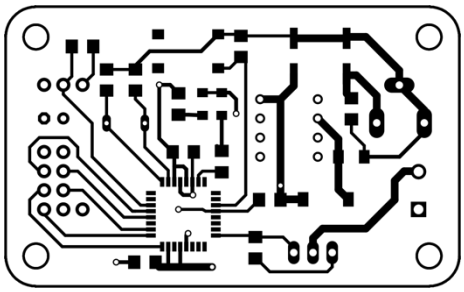
Vrstva BOTTOM



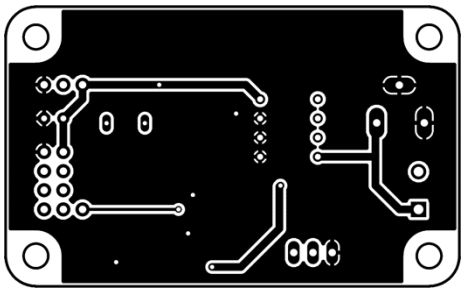
Rozmer dosky 80x49 [mm], mierka M1:1

C.7 Doska plošného spoja Radiátor modulu

Vrstva TOP



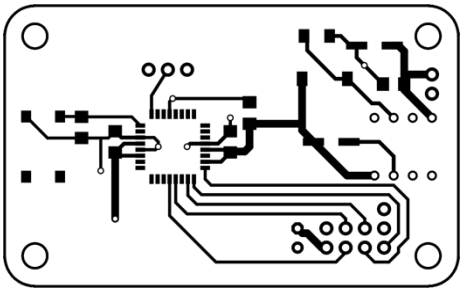
Vrstva BOTTOM



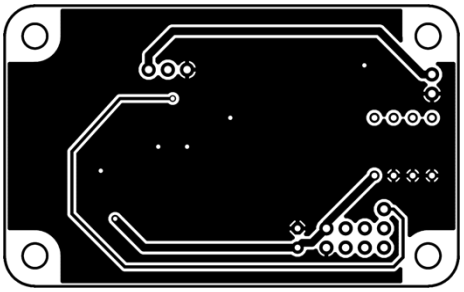
Rozmer dosky 60x37 [mm], mierka M1:1

C.8 Doska plošného spoja Senzor modulu

Vrstva TOP



Vrstva BOTTOM



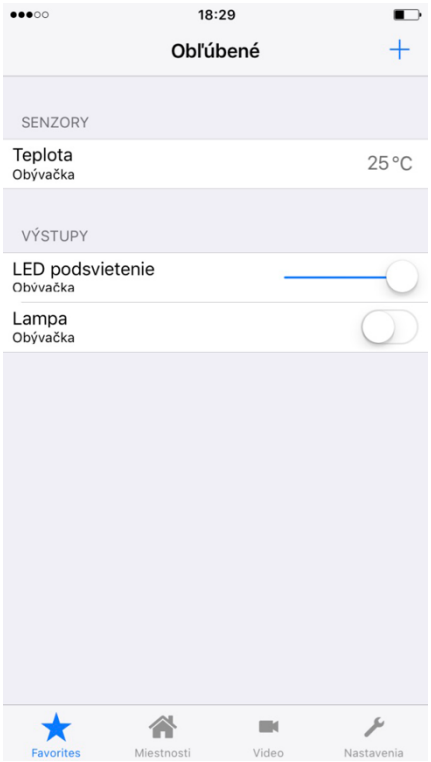
Rozmer dosky 60x37 [mm], mierka M1:1

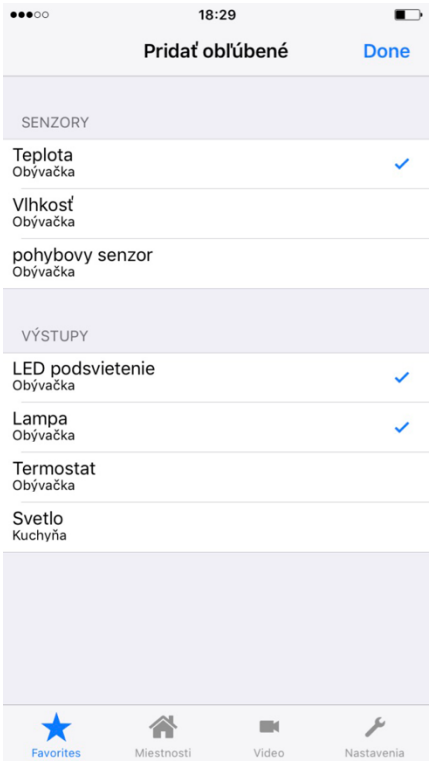
D NÁVOD NA MOBILNÚ APLIKÁCIU

Mobilná aplikácia poskytuje 4 hlavné zobrazenia: Obľúbené, Miestnosti, Kamera, Nastavenia. Každé zobrazenie poskytuje iné informácie a možnosti interakcie používateľa.

Obľúbené

V tomto tabe sa nachádzajú užívateľom vybrané senzory a výstupy vďaka ktorým má rýchly prístup k prvkom ku ktorým pristupuje najčastejšie. Priradenie senzoru a ovládača je rovnaké. Po kliknutí na ikonu PLUS sa zobrazí zoznam senzorov a ovládačov z ktorých je možné vyberať.

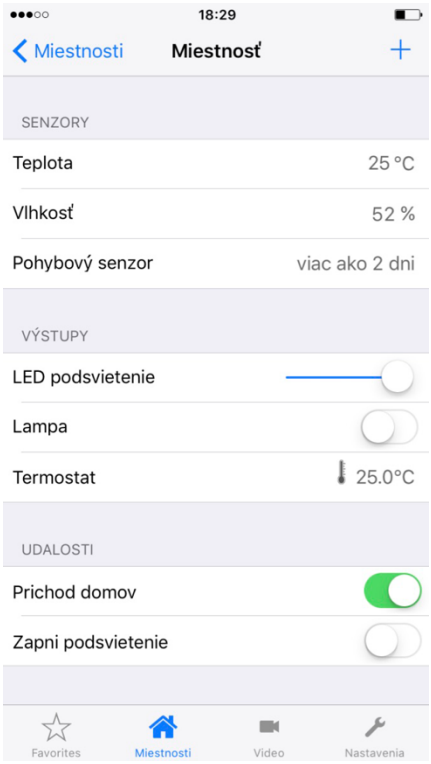


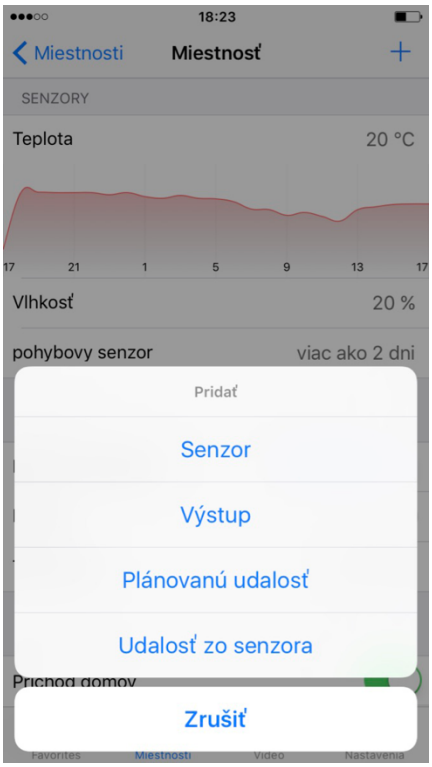


Miestnosti

Defaultným zobrazením je zoznam všetkých miestností. Po kliknutí ikony PLUS sa zobrazí krátky formulár na pridanie miestnosti. Potiahnutím doľava je možné vymazať jednotlivé prvky v miestnosti.

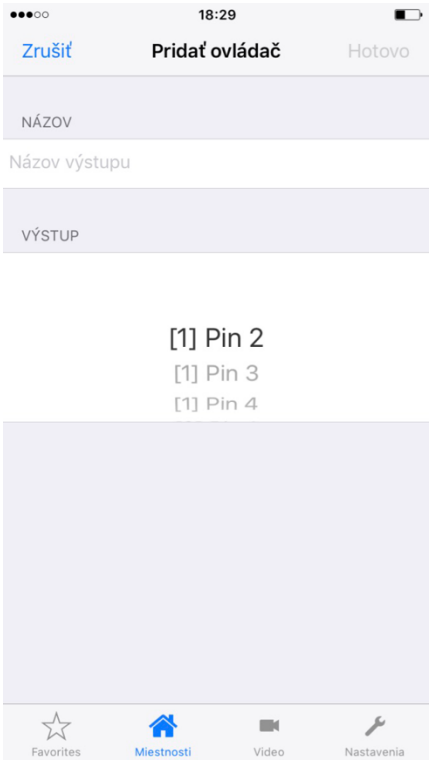
Po kliknutí na miestnosť sú zobrazené všetky ovládače a senzory v miestnosti, tie je možné priradiť rovnako ikonou PLUS v pravom hornom rohu okna. Po kliknutí sa zobrazí rozšírená ponuka.





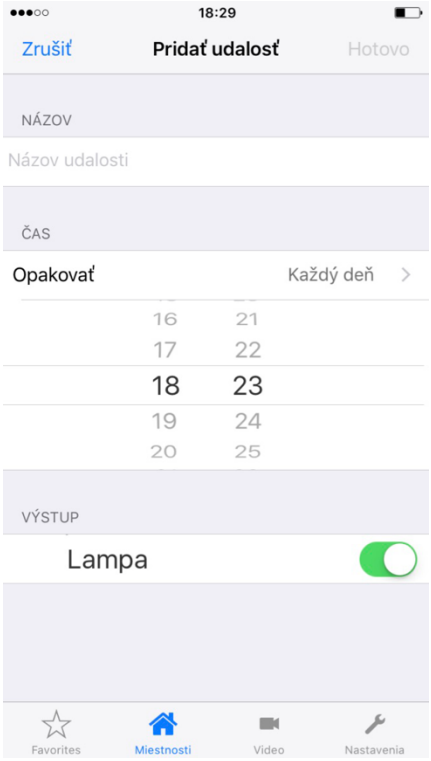
Jednotlivé senzory, výstupy alebo udalosti je možné odstrániť z miestnosti jednoduchým potiahnutím doľava.

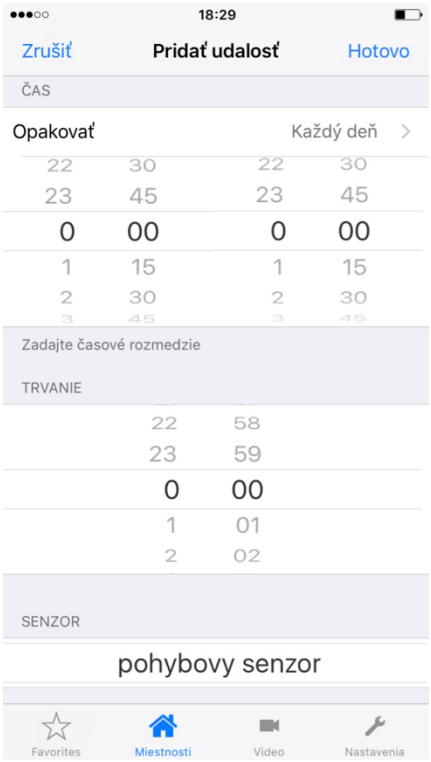
Tu je možné priradiť všetky dostupné senzory do zvolenej miestnosti. Okrem toho je možné vybrať farbu grafu.



V tomto okne je možné priradiť výstupy z modulov do vybranej miestnosti.

Časová udalosť volaná každý zvolený deň o konkrétnom čase. Je potrebné zaškrtnúť vybrané dni (default každý deň v týždni), zvoliť konkrétny čas a vybrať požadovaný výstup a hodnotu ktorú ma nadobudnúť v danom čase. Výstup sa musí nachádzať v zvolenej miestnosti.





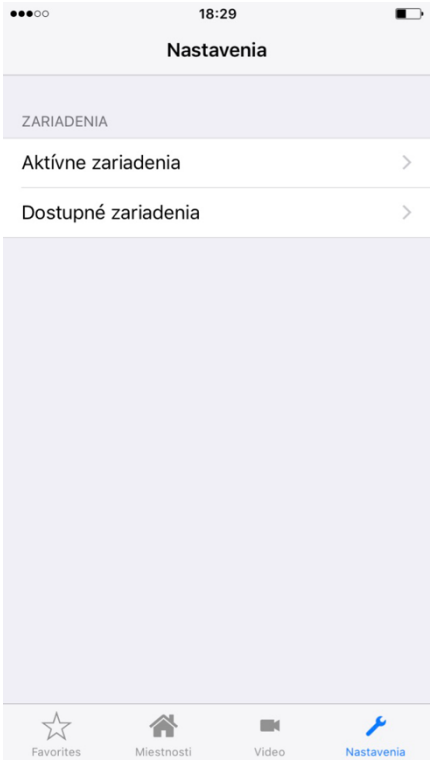
Senzorová udalosť je zavolaná po zmene hodnoty zvoleného senzora. Je možné nastaviť dni a čas kedy má byť udalosť aktívna. Navyše oproti predošlým udalostiam je možné nastaviť trvanie (od 1 minúty po 24 hodín) kedy sa výstup vráti na pôvodnú hodnotu. Výstup aj senzor sa musí nachádzať v zvolenej miestnosti.

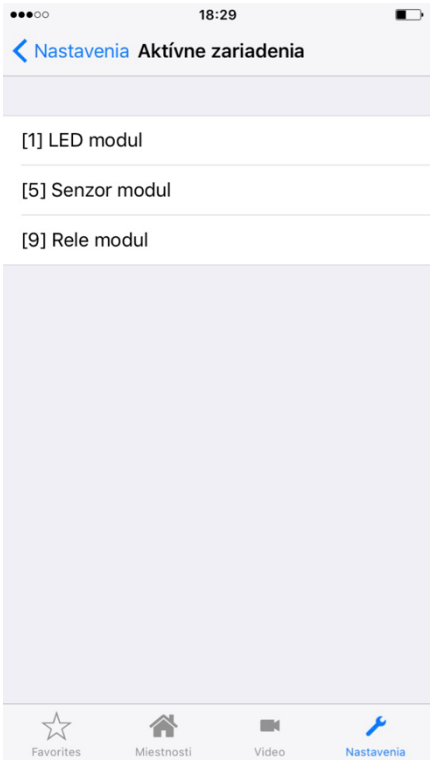
Kamera

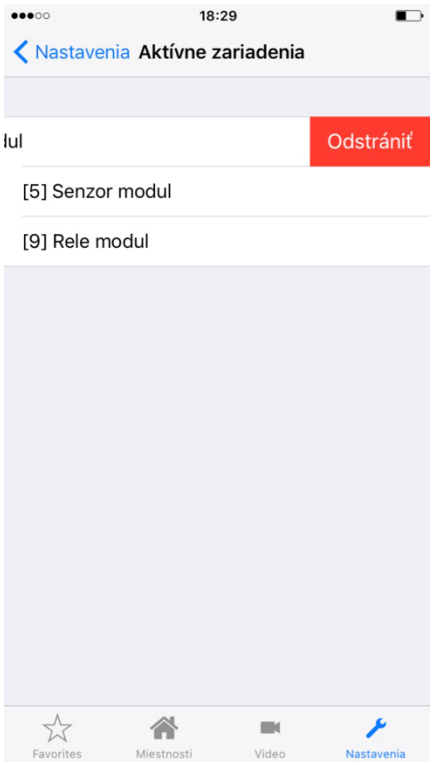
V prípade že je dostupná kamera jej výstup sa zobrazí v aplikácii v jednoduchom okne. Jedná sa o aktuálny pohľad so zobrazením približne 4-5 snímkov za sekundu.

Nastavenia

V tabe nastavenia je možné prezerať aktívne zariadenia. V prípade že sa v okolí nachádza neprihlásené zariadenie bude sa tu zobrazovať a je možné ho prihlásiť. V časti aktívne zariadenia je možné potiahnutím doľava zariadenie vymazať. Po vymazaní budú odstránené všetky nastavené senzory, výstupy a udalosti ovplyvnené týmto zariadením.







Prihlásené moduly je možné odstrániť jednoduchým potiahnutím doľava.