České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Dálkové ovládání LED

Závěrečná zpráva

Ondřej Červenka

Vedoucí práce: Ing. Peter Macejko
Název předmětu: Internet of Things
Kód předmětu: MI-IOT

Praha, 9. května 2016

Obsah

1 Specifikace zadání

Cílem práce je vytvoření sítě zařízení, která umožní dálkové ovládání LED osvětlení. Síť bude obsahovat dálkový ovladač, desku řídící LED a centrálu pro přístup k ostatním zařízením pomocí webového rozhraní.

Kromě toho bude možné do sítě přidávat další zařízení, se kterými bude centrála komunikovat. Dalkový ovladač a deska ovládající osvětlení spolu naopak mohou komunikovat nezávisle na centrále.

1.1 Řídící desky

1.1.1 Vysílač a přijímač

Základem pro každé zařízení je deska Arduino Nano s mikrokontrolérem ATmega328. Tato deska má dostatek PWM výstupů pro ovládání LED i analogových vstupů vhodných pro čtení fotosenzoru.

Dále obsahuje sběrnice `I2C`, `SPI` a sériovou linku, pomocí kterých lze připojit další moduly. V tomto případě by se jednalo především pro modul pro bezdrátovou komunikaci.

Deska se dá snadno programovat pouze s pomocí USB, nejsou tedy potřeba další vývojové prostředky. Programová paměť činí 32 KB, což by pro tuto aplikaci mělo být naprosto dostačující.

Software pro Arduino bude implementován v jazyce C++ s využitím Arduino knihoven.

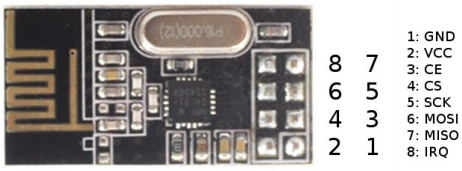
1.1.2 Centrála

Jako centrálu použijeme jednodeskový počítač Raspberry Pi. To poskytuje sběrnice pro připojení zvoleného komunikačního modulu a zároveň umožní provoz serveru s webovým rozhraním pro ovládání sítě zařízení.

Na Raspberry Pi poběží operační systém Raspbian. Teoreticky by mělo být možné použít jakýkoliv operační systém, Raspbian však poskytuje největší podporu při připojování periferií (v našem případě jde především o `SPI` sběrnici pro připojení komunikačního modulu).

Většina softwaru pro Raspberry Pi bude implementována v jazyce Python, pro webové rozhraní využijeme microframework Flask[1].

1.2 Komunikace



Obrázek 1: nRF24L01. Obrázek převzat z https://nathan.chantrell.net/blog/wp-content/uploads/2013/08/nrf24l01.jpg

Komunikace v síti bude realizována pomocí modulů `nRF24L01`. `nRF24L01` je modul pro rádiovou komunikaci od firmy Nordic Semiconductor, využívající frekvenci 2.4 GHz. Modul je k dispozici ve více verzích, většinou umožňujících komunikaci s mikrokontrolérem pomocí sběrnice `SPI`.

[1] http://flask.pocoo.org/

Jak na Arduino deskách, tak na Raspberry Pi využijeme pro ovládání modulu knihovnu RF24[2].

2 Implementace

2.1 Síť

Modul `nRF24L01` při adresování využívá prijímacích (`reading pipe`) a vysílacich (`writing pipe`) kanálů. Vysílací i přijímací kanál je definován svou adresou.

Každý modul může mít najednou otevřený jeden vysílací a až zapisovacích kanálů. Otevřené kanály se dají v případě potřeby za běhu měnit.

Délka adresy může být libovolná (defaultně 5 bytů), při volbě adresy je však třeba dát pozor na to, že přijímací kanály jednoho zařízení se mohou lišit pouze v posledním bytu (pro adresaci zbylých pěti kanálů jsou použity první 4 byty adresy prvního kanálu)

Pokud chceme oboustranou komunikaci dvou zařízení, potřebujeme dvojici kanálů, z nichž jeden bude vysílací kanál pro první zařízení a přijímací pro druhé zařízení a druhý naopak.

V naší síti třech zařízení tedy budeme potřebovat šest komunikačních kanálů, jak je vidět na obrázku 2. Každé Arduino má otevřené dva přijímací kanály, jeden pro Raspberry Pi a jeden pro druhé Arduino. Dále pak každé Arduino zapisuje do příslušného vysílacího kanálu buď pro Raspberry nebo pro druhé Arduino.

Raspberry Pi nemá jako centrála permanentně otevřený žádný přijímací kanál, ale pouze zasílá zprávy na ostatní zařízení, a pak na zvoleném kanálu čeká na odpověď.

Adresy kanálů jsou pro Arduina pevně definovány, pro jejich změnu je tedy nutné programy znovu zkompilovat a nahrát. Oproti tomu Raspberry Pi žádné pevně definované kanály nemá, komunikace přes centrálu je tedy možná s každým zařízením, které má otevřený nějaký kanál (kvůli kolizím by tento kanál měl být vyhrazený pouze pro komunikaci s centrálou).

Do sítě je tedy možné snadno přidávat další zařízení nezávisle na těch, které už v síti jsou (pokud nová zařízení potřebují komunikovat pouze s centrálou nebo mezi sebou).

2.2 Komunikační protokol

Komunikace se zařízeními bude založena na zprávách velikosti 10 bytů s následujícím formátem[3]:

0	1	2 - 5	6 - 9
příkaz	číslo odesílatele	timestamp	data

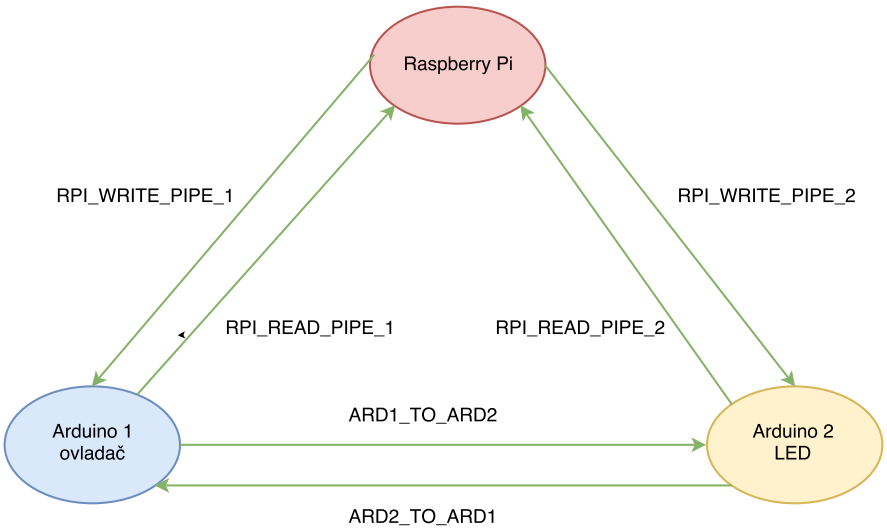
Možné příkazy budou záviset na cílovém zařízení. V základní verzi sítě bude deska řídící ovětlení odpovídat na zprávy v tabulce 1. Pomocí těchto příkazů bude možné ovládat až 4 diody (s číslem 0 - 3) buď pomocí dálkového ovládání nebo centrály.

Všechna zařízení budou na příchozí zprávy odpovídat buď zprávou `OK` nebo `not supported` v případě, že danou funkci nepodporují. Zpráva `OK` také slouží pro předání návratové hodnoty v datové sekci zprávy. Například pro zprávu `LED state` budou ve čtyřech datových bytech zprávy stavy pinů diod v pořadí 0, 1, 2, 3 (hodnoty PWM v rozsahu 0 - 255).

Software pro dálkový ovladač nepočítá se žádnými dalšími periferiemi kromě tlačítek, takže toto zařízení odpovídá na všechny dotazy zprávou `not supported`. Stejně tak centrála pouze přijímá odpovědi, a není jí tedy možné zasílat nečekané zprávy (Raspberry Pi aktivně neposlouchá na přijímacích kanálech).

[2] https://tmrh20.github.io/RF24/

[3] Timestamp zatím není implementován.



Obrázek 2: Diagram sítě

Příkazy				
číslo	**příkaz**	**parametry**	**návratová hodnota**	**popis**
0x01	LED toggle	číslo LED	stav LED	přepnutí stavu LED
0x02	LED on	číslo LED	-	zapnutí LED
0x03	LED off	číslo LED	-	vypnutí LED
0x04	LED state	-	stav všech LED	získání současného stavu všech LED
0x05	LED write	hodnota PWM číslo LED	-	nastavení střídy PWM pro danou LED
Odpovědi				
číslo	**příkaz**	**parametry**	**návratová hodnota**	**popis**
0x00	OK	-	data	potvrzení a vracení dat
0xFF	not supported	-	-	zařízení nepodporuje zaslaný příkaz

Tabulka 1: Tabulka příkazů

2.3 Dálkový ovladač a LED deska

Obě zařízení jsou postavena na řídící desce Arduino Nano, jak je zmíněno v sekci 1.1. Program ovladače je velmi jednoduchý, pouze při stisknutí tlačítka zašle zprávu pro přepnutí stavu příslušné diody. Zároveň na všechny příchozí žádosti odpovídá `not supported`.

Zatím není implementován žádný potvrzovací protokol, při neúspěšném odeslání zprávy (a nerozsvícení LED) je tedy nutné stisknout tlačítko znovu.

Program pro desku ovládající diody je jen o něco složitější. Toto zařízení podporuje všechny příkazy z tabulky 1 a komunikuje jak s dálkovým ovladačem, tak s centrálou. Má tedy otevřené dva přijímací kanály, a podle čísla zařízení v příchozí zprávě přepíná vysílací kanál a zasílá odpověď.

Deska umožňuje připojení čtyř diod k pinům 3, 5, 6 a 9. Tyto piny umožňují PWM výstup s rozlišením 8 bitů.

2.4 Centrála

2.4.1 Zasílání zpráv

Bohužel se mi nepodařilo zkompilovat Python wrapper pro knihovnu RF24 na Raspberry Pi, komunikace s ostatnímy zařízeními v síti tedy bude realizována pomocí jednoduchého programu napsaného v C++ a ovládáného z příkazové řádky. Tento program pak je možné používat v Python skriptu například pomocí modulu `subprocess`.

Formát vstupu programu je:

```
./rpi_rf24.out adresa_read_pipe adresa_write_pipe cislo_prikazu data
```

Všechna data jsou zadávána v hexadecimálním formátu (bez `0x`). Adresy mají velikost 5 bytů, číslo příkazu 1 byte a data 4 byty. Po spuštění program zapíše příkaz a data na adresu vysílacího kanálu (první argument) a na adrese přijímacího kanálu (druhý argument) čeká na odpověď.

V případě, že se zprávu nepodaří odeslat, nebo vyprší doba čekání na odpověď, opakuje program až 5 pokusů o odeslání zprávy[4].

Vzhledem k úmyslu volat program z Python skriptu je výstup co nejúspornější. Jde tedy pouze o 10 bytů přijaté zprávy, oddělených mezerami. Ostatní informace jsou vypisovány na chybový výstup.

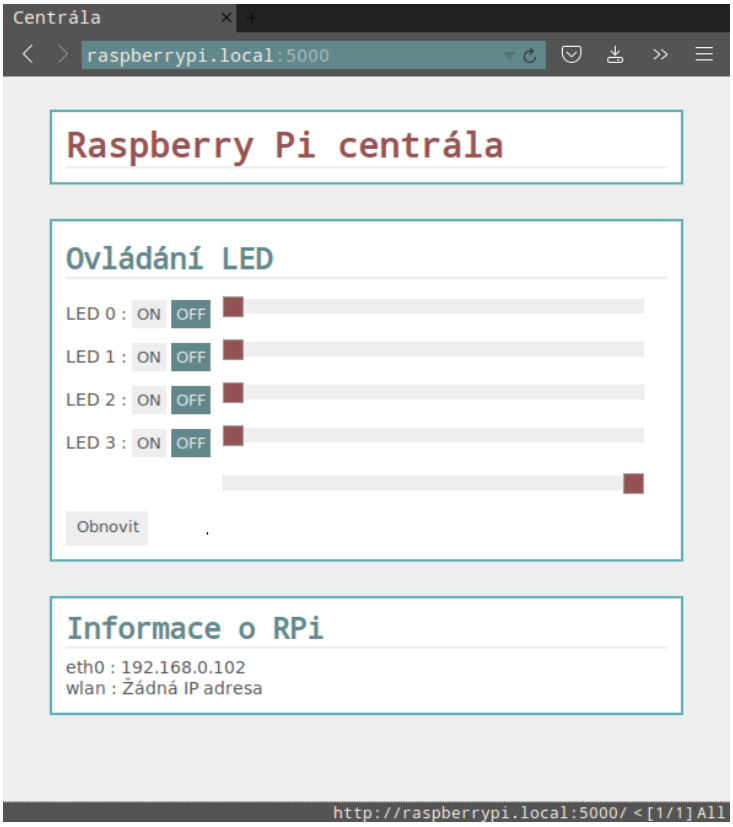
2.4.2 Třída Device

Pro základní reprezentaci zařízení v síti při psaní Python skriptu jsem vytvořil třídu `device`. Ta poskytuje jednoduché rozhraní pro volání programu `rpi_rf24` zmíněného v secki 2.4.1.

Atributy	
reading_pipe	kanál, na kterém bude centrála poslouchat při komunikaci se zařízením
writing_pipe	kanál, do kterého bude centrála zapisovat
dev_num	číslo zařízením, které instance reprezentuje
name	symbolický název zařízení

Třída poskytuje jedinou metodu `send_cmd(cmd_num, cmd_data)`, která zašle zvolený příkaz na zařízení, které instance třídy reprezentuje a vrátí pole bytů s odpovědí. Parametr `cmd_data` je zadáván ve formě stringu hexadecimálních hodnot každého bytu. V případě, že se nepodaří zařízení kontaktovat, metoda vyvolá vyjímku `Device_error` s informacemi o chybě.

[4] Tady pozor na příkazy jako přepnutí stavu LED, protože pokud centrála nedostane odpověď v žádném z pěti pokusů, tak není jisté v jakém stavu LED vlastně je (zpráva tam mohla dorazit, ale odpověď se stratila).



Obrázek 3: Webové rozhraní

Z této třídy poté mohou dědit další třídy reprezentující specializovaná zařízení. V našem případě jde o třídu reprezentující LED desku, jejíž metody odpovídají příkazům z tabulky 1.

2.4.3 Webové rozhraní

Webové rozhraní je implementováno v Pythonu[5] pomocí frameworku Flask. Server běží na doméně `raspberrypi.local` a portu 5000 a je přístupný v místní síti. Při nahrávání domovské stránky je od LED desky získán pomocí příkazu `LED state` stav diod. V případě, že se desku nepodaří kontakovat, je zobrazena chybová hláška.

Komunikace se serverem je založena na jednoduchých REST voláních. Pro ovládání LED desky budeme potřebovat pouze jedno:

`http://raspberrypi.local:5000/led/<led_num>/<value>`

Toto volání nastaví pin diody číslo `<led_num>` pomocí funkce `analogWrite` na hodnotu PWM určenou parametrem `<value>` v rozsahu 0-255. K zaslání zprávy je použit program `rpi_rf4`.

Po vyřízení server přesměruje uživatele zpět na domovskou stránku (`/`), čímž se aktualizuje stav diod. Stav zobrazený ve webovém rozhraní (po obnovení) by tedy měl vždy odpovídat skutečnému stavu.

Hodnotu PWM lze nastavit pomocí posuvníků (obrázek 3, spodní posuvník nastavuje ostatní posuvníky najednou). Po obnovení stránky posuvníky zobrazují hodnotu nastavenou na příslušných diodách.

[5] Python 3

3 Závěr

V práci se podařilo vytvořit funkční zařízení, umožňující ovládat diody buď pomocí dálkového ovladače, nebo přes webové rozhraní.

Při tvorbě práce jsem se snažil především vytvořit základ, na kterém by se dalo snadno stavět při budoucím rozšiřování zařízení. Bohužel jsem tedy nestihl implementovat všechny funkce které jsem chtěl, alespoň některé však plánuji dodělat.

3.1 Další práce

Mezi první funkci, která mi na zařízení chybí je sbírání dat z různých senzorů a vykreslování grafů na webové stránce. Zatím jsem připravil pouze třídu pro logování hodnot a tvorbu grafu, je tedy potřeba doplnit čtení senzorů na zařízeních v síti a případné přepočítávání hodnot. To by měla obstarat další třída dědící z třídy `Device`.

Dále by bylo vhodné vytvořit konfigurační soubor[6] pro centrálu, který bude popisovat síť a zařízení v ní. Zatím centrála komunikuje pouze s LED deskou, což je napevno napsáno v Python kódu a přidávání dalších zařízení touto cestou by bylo nepohodlné.

Také bych chtěl připravit html šablony popisující grafickou reprezentaci jednotlivých specializovaných zařízení ve webovém rozhraní.

3.2 Zdrojové kódy

Zdrojové kódy práce jsou k dispozici na https://github.com/ggljzr/mi-iot-semestralka. Repozitář plánuji v případě rozšiřování aktualizovat.

[6] Dobře čitelný formát je například https://github.com/toml-lang/toml.