WIRELESS CONTROL OF EMBEDDED M2M DEVICES USING UNLICENSED ISM BAND

Martin Štůsek

Master Degree Programme (2), FEEC BUT

E-mail: xstuse01@stud.feec.vutbr.cz

Supervised by: Pavel Mašek

E-mail: masekpavel@feec.vutbr.cz

Abstract: This research work follows the idea of Machine-to-Machine (M2M) communication designing the radio controlled devices using the ISM radio band of 433 MHz. Implemented solution consists of transmitter unit (Raspberry Pi) and sets of receiver units based on Arduino Uno platform. To offer better user experience, simple management interface on the side of transmitter is imple- mented.

Keywords: Arduino Uno, ISM 433 MHz, Machine-to-Machine (M2M), OSGi, Raspberry Pi

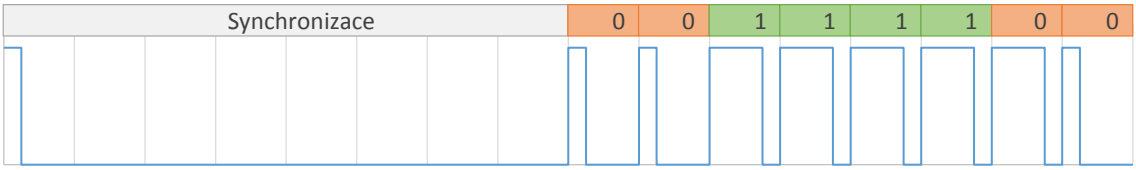
1. ÚVOD

Tato práce se zabývá bezdrátovým ovládáním elektronických zařízení s využitím jednodeskového počítače Raspberry Pi a 8-bitového mikrokontroleru Arduino Uno. Raspberry Pi je použito jako server s vysílací částí zajišťující interakci s uživatelem. Mikrokontroler Arduino slouží jako klient pro přijímání dat umožňující ovládání elektrických zařízení. K bezdrátové komunikaci je využito vysílačů/přijímačůpracujících na frekvenci 433 MHz.

2. IMPLEMENTACE BEZDRÁTOVÉHO PROTOKOL

Jelikož na frekvenci 433 MHz komunikují zejména zařízení s nízkým výpočetním výkonem tzv. embedded zařízení, je při komunikaci mezi zařízeními využito jednoduchého klíčování typu OOK (On- Off Keying). Komunikace v pásmu ISM (Industrial, Scientific and Medical) na 433 MHz je bezlicenční, a proto je využívána velkým množstvím zařízení. Z důvodu možného rušení souběžně komunikujícími zařízeními je ale nutné každý povel vyslat opakovaně. Nejmenší doporučený počet opakování je tři, z důvodu velkého počtu komunikujících zařízení není ovšem tato hodnota dostačující – proto je v této implementaci každý kód odeslán desetkrát.

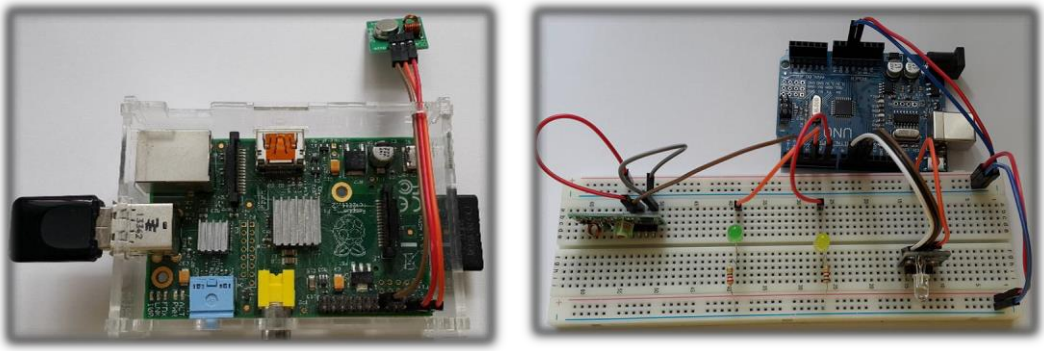
Implementovaný protokol vychází z kódování využívaného v čipech EV1527, kdy jsou využity pulzy o délce 300 až 500 μs. Jak lze vidět na Obr. 1, přenos je zahájen jedním pulzem vysoké úrovně a 31 pulzy nízké úrovně. Po synchronizační sekvenci jsou přenášena již samotná data. `Log 1` je vyjádřena třemi pulzy vysoké úrovně a jedním pulzem nízké úrovně. Naopak `log 0` je definována jedním pulzem vysoké úrovně a třemi pulzy nízké úrovně. Pro adresaci zařízení je využito protokolu s délkou 8 bitů– lze tedy adresovat až 255 zařízení (adresu 0 nelze využít). Pro přenos samotných příkazůje využit 32bitový protokol umožňující nastavení každé složky RGB LED v 255 úrovních [1].



Obrázek 1: Průběh 8 bitové zprávy při přenosu čísla „62“.

3. VYSÍLACÍ Č ÁST

Jako zařízení vysílající příkazy je využito Raspberry Pi model B gen. 1, které je s vysílačem propojeno skrze GPIO (General-purpose input/output), viz Obr. 2a. Vysílací jednotka také implementuje aplikaci zajišťující správnou interpretaci a převod příkazůz uživatelských vstupů. Dále je vytvořen webový server poskytující uživatelské rozhraní pro ovládání připojených zařízení.



(a) Vysílací část. (b) Přijímací část.

Obrázek 2: Zapojení zařízení pro testování.

3.1. BACKEND

Celý sytém je napsán v jazyce Java s využitím OSGi (Open Service Gateway Initiative) frameworku Knopflerfish. Standard OSGi definuje sadu doporučení pro dynamický systém balíčků(bundles) v jazyce Java. Tento systém umožňuje instalovat, spouštět či zastavovat jednotlivé balíčky bez nutnosti restartu programu. Základní filozofií OSGi je tvořit jednoduché aplikace, které poskytují jednu, přesně definovanou, službu například pro HTTP (Hypertext Transfer Protocol) komunikaci.

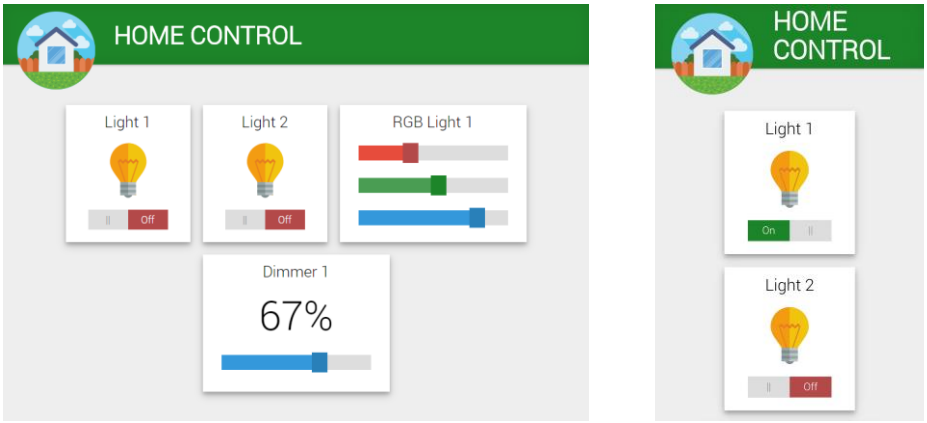
Základními prvky architektury OSGi jsou tzv. `bundles`, což je sada tříd rozšířená o speciální soubor `manifest` – poskytuje dodatečné informace pro OSGi class loader. OSGi definuje také sadu služeb, která umožňuje komunikaci mezi jednotlivými balíčky, v této práci je to zejména vrstva služeb (`service layer`) a `OSGi Event Admin` [2].

Výsledná aplikace je rozdělena do tří samostatných balíčků:

- **ItemsRegistry** – definuje datovou strukturu položek a udržuje v paměti seznam všech položek s jejich aktuálními hodnotami. Tento balíček musí být spuštěn jako první, v opačném případě není možné spustit zbylé balíčky, které jsou na něm závislé. Jako registr položek slouží objekt typu `ConcurentHashMap`, který poskytuje bezpečný přístup k položkám i ve vícevláknových aplikacích.
- **Transmitter** – stará se o zpracování přijatých příkazůa samotné vyslání povelů. Komunikace s GPIO portem je realizována pomocí knihovny Pi4J [3], která umožňuje přímí přístup na GPIO rozhraní z prostředí Java.
- **Web API** – zajišťuje komunikaci s webovým uživatelským rozhraním. Ke komunikaci je využito protokolu WebSocket, který je základní součástí jazyka HTML5 (HyperText Markup Language). Na základě vlastností protokolu WebSocket je komunikace mezi serverem a uživatelem velmi rychlá bez viditelného zpoždění, které je patrné u technologií využívající pooling. Data jsou následně přenášena jako objekty typu JSON (JavaScript Object Notation).

3.2. UŽIVATELSKÉ ROZHRANÍ

Aplikaci lze ovládat skrze vytvořené uživatelské rozhraní, viz Obr. 3. Prvek `Light` slouží jako spínač se dvěma stavy. `Dimmer` umožňuje nastavit výstupní hodnotu v rozsahu 0 až 255 (vhodné pro stmívače) a `RGB` umožňuje ovládat jednotlivé složky RGB diody.



Obrázek 3: Uživatelské rozhraní aplikace.

4. PŘÍJÍMACÍ ČÁST

Jak bylo zmíněno v úvodní části, jako přijímací část slouží mikrokontroler Arduino Uno s rádiovým přijímačem, viz Obr 2b. Příjem dat je signalizován externím přerušením, z tohoto důvodu je nutné datový pin přijímače připojit k pinu 2 nebo 3 – u mikroprocesor Atmega 328p mohou pouze piny 2 a 3 vyvolat externí přerušení.

Maximální délka přenesených dat je z důvodu využití datového typu `unsigned long` pro příkazy omezena na 32 bitů. Není také možné přenést nulová data, tzn. v sekvenci dat se musí vyskytovat alespoňjeden bit logické hodnoty 1. Nulová data by totiž mohla být snadno zaměněna se šumem.

Zpracování příkazůse skládá ze dvou částí, nejprve jsou vyslána data obsahující adresu zařízení a v další fázi jsou přenesena data s příkazem. Pokud jsou přijata data pro zařízení typu `switch` je na výstup digitálního pinu zapsána logická hodnota 0 nebo 1, dle typu příkazu. Pro zařízení typu `dimmer` či `RGB`, jsou použity analogové výstupní piny v režimu PWM (Pulse Width Modulation).

5. ZÁVĚR

V této práci byl proveden popis charakteristiky bezdrátového ovládání zařízení a jednoduchého protokolu pro bezdrátovou komunikaci. Dále byla popsána implementace vysílací části postavené na Raspberry Pi a přijímací části využívající zařízení Arduino Uno.

Důraz byl kladen na (i) popis vysílací části, která obsahuje logiku pro zpracování a vyslání příkazů, (ii) rozhraní pro komunikaci skrze protokol WebSocket a (iii) vytvořený webový server s uživatelským rozhraním.

REFERENCE

[1] KnowHow_LineCoding. *GitHub* [online]. 2016 [cit. 2016-03-02]. Dostupné z: https://github.com/sui77/rc-switch/wiki/KnowHow_LineCoding

[2] OSGITM ALLIANCE. *OSGi: The Dynamic Module System for Java* [online]. 2015 [cit. 2015-11-22]. Dostupné z: https://www.osgi.org

[3] *The Pi4J Project: Connecting Java to the Raspberry Pi* [online]. 2015 [cit. 2016-03-02]. Dostupné z: http://pi4j.com/