Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vzdálené ovládání mikropočítačů pomocí mobilní aplikace

Jan Ouředník

© 2018 ČZU v Praze

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Ouředník

Informatika

Název práce

Vzdálené ovládání mikropočítačů pomocí mobilní aplikace

Název anglicky

Remote control of microcomputers per Mobile application

Cíle práce

Práce se zaměřuje na možnosti využití mikropočítačů a jejich následné modifikace pomocí instalovatelných zařízení, která rozšiřují možnosti jejich využití. Analýza směřuje hlavně k oblasti využití SW ke vzdálenému ovládání mikropočítače z mobilních zařízení.

Hlavním cílem práce je zkoumání využití různých skupin mikropočítačů v dané oblasti a následně ukázkového systému s vybraným mikropočítačem, doplňkovými zařízeními a vlastním SW pro vzdálenou správu přes mobilní zařízení.

Metodika

Práce se sestává ze dvou hlavním částí, popisu teoretických východisek a praktické části. Metodika zpracování přehledu problematiky vychází ze studia odborných informačních zdrojů. Na základě syntézy zjištěných poznatků pak bude formulován přehled stávajícího stavu problematiky a teoretická východiska pro praktickou část práce.

Praktická část práce bude spočívat v návrhu systému na bázi vybraného mikropočítače, ke kterému bude dále navržen a implementován SW pro dálkové ovládání z mobilního zařízení. Ukázkový systém bude představovat dálkově ovládaný univerzální přepínač řízený mikropočítačem.

Doporučený rozsah práce

35-40 stran

Klíčová slova

mikropočítač, SW, Mobilní zařízení, Bluetooth, Ovládání

Doporučené zdroje informací

Arduino Android Blueprints, Schwartz, Marco; Buttigieg, Stefan Packt Publishing 2014, ISBN: ISBN number:9781784390389, ISBN number:9781784391683.

Beginning Android Application Development, Lee, Wei-Meng John Wiley & Sons, , Incorporated 2011, ISBN: ISBN number:9781118017111, ISBN number:9781118087299, EDICE: Wrox beginning guides.

RAAB, Stefan a Madhavi W. CHANDRA. Cisco: mobilní IP technologie a aplikace. Praha: Grada, 2007. 299 s. ISBN 80-247-1611-9.

TUTTLEBEE, Walter. Software Defined Radio Enabling Technologies. Chichester: John Wiley, 2003. 440 s. ISBN 978-0-470-85263-7.

Předběžný termín obhajoby

2017/18 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 7. 3. 2018

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 7. 3. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 08. 03. 2018

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vzdálené ovládání mikropočítačů pomocí mobilní aplikace" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 13.3.2018 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Jiřímu Brožkovi, Ph.D. za jeho cenné rady, odborné vedení při tvorbě bakalářské práce a čas, který mi věnoval při konzultačních hodinách.

Vzdálené ovládání mikropočítačů pomocí mobilní aplikace

Abstrakt

Tato bakalářská práce se zabývá možnostmi využití mikropočítačů, které se vzdáleně ovládají pomocí mobilních zařízení. V teoretické části jsou definovány typy bezdrátové komunikace, které jsou nezbytnou součástí pro vzdálené ovládání mikropočítačů. Následně je popsán samostatný mikropočítač a jeho typy dělení, kde jsou zobrazeny dvě různé mikroprocesorové desky. Dalším tématem jsou softwary Arduino IDE a Android Studio, ve kterých byla vytvořena mobilní aplikace pro vzdálené ovládání mikropočítačů. Poslední kapitolou jsou programovací jazyky, které se využívaly již ve zmíněných softwarech.

Analytická část bakalářské práce se zabývá návrhem systému, který je založen na bázi vybraného mikropočítače, ke kterému je navržen a implementován software pro dálkové ovládání vycházející z mobilního zařízení. V závěru práce je zobrazen systém představující dálkově ovládaný univerzální přepínač řízený mikropočítačem a následně je popsáno jeho využití v praxi.

Klíčová slova: mikropočítač, SW, mobilní zařízení, bluetooth, Arduino, Arduino IDE, ovládání, programovací jazyky, Raspberry Pi

Remote control of microcomputers per Mobile application

Abstract

This bachelor thesis describes possibilities of using microcomputers, which are remotely controlled by mobile devices. The theoretical part defines the types of wireless communication, which are an essential part of the remote control of microcomputers. Then there is described a microcomputer and its types, where are two different microprocessor boards. Another topic is the Arduino IDE and Android Studio software, in which a mobile application for remote control of microcomputers was created. The last chapter is the programming languages that were already used in the mentioned software.

The analytical part of the bachelor thesis is focused on the design of a system based on a selected microcomputer, to which is designed and implemented the software for remote control based on mobile device. At the end of the thesis is shown a system representing a remote controlled universal switch controlled by a microcomputer and then there is described its use in practice.

Keywords: microcomputer, SW, mobile device, Bluetooth, Arduino, Arduino IDE, control, programming languages, Raspberry Pi

Obsah

1 Úvod

Dříve pouze futuristické využití elektroniky dnes bývá každodenní záležitostí. Žijeme v době, ve které se čím dál více rozrůstá takzvaný „smart“ fenomén a s ním vzrůstá i zájem o zařízení s ním spjatá. Pojem chytré domácnosti už zdaleka není žádnou neznámou. Pro nedotčené jedince tímto pojmem je nutné zmínit, že se jedná o zjednodušení, někdy až plné zautomatizování každodenních úkonů pomocí využití počítačové techniky.

Tato bakalářská práce se zabývá odvětvím mobilních zařízení, pomocí kterých lze snadno a jednoduše ovládat ostatní zařízení. V rámci možností rozsahu práce bylo zaměření zúženo čistě na přepínání koncového zařízení mezi polohami zapnuto a vypnuto formou přerušování zdrojového napětí.

Nejvíce využívanou funkcí tohoto odvětví je možnost ovládání kompletní světelné elektroinstalace v domácnosti pomocí mobilní aplikace. Stejně jednoduché jako je zhasnutí jedné lampičky, ale muže být vypnutí jakéhokoliv elektrického zařízení, ať už se jedná o větrák, filtraci u zahradního bazénu, nebo kompletního odpojení zásuvky od napětí.

Dílčí části bakalářské práce budou směřovány k vytvoření funkčního prototypu univerzálního přepínače napětí pro koncové zařízení. Pro výsledné ovládání bude vyvinuta vlastní aplikace, která uživateli umožní z jeho mobilního zařízení udělat dálkový ovladač k vytvořenému přepínači.

2 Cíl práce a metodika

2.1 Cíl práce

Práce se zaměřuje na možnosti využití mikropočítačů a jejich následné modifikace pomocí instalovaných zařízení, která rozšiřují možnosti jejich využití. Analýza směřuje hlavně k oblasti využití SW ke vzdálenému ovládání mikropočítače z mobilních zařízení.

Hlavním cílem práce je zkoumání využití různých skupin mikropočítačů v dané oblasti a následného ukázkového systému s vybraným mikropočítačem, doplňkovými zařízeními a vlastním SW pro vzdálenou správu přes mobilní zařízení.

2.2 Metodika

Práce se skládá ze dvou hlavních částí, popisu teoretických východisek a praktické části. Metodika zpracování přehledu a problematiky vychází ze souhrnu studia odborných informačních zdrojů, explanací terminologie a pojmů, které jsou využívány. Na základě syntézy zjištěných poznatků pak bude formulován přehled stávajícího stavu problematiky a teoretická východiska pro praktickou část práce.

Praktická část práce bude spočívat v návrhu systému na bázi vybraného mikropočítače, ke kterému bude dále navržen a implementován SW pro dálkové ovládání z mobilního zařízení. Závěr bakalářské práce bude ukázkový systém představující dálkově ovládaný univerzální přepínač řízený mikropočítačem.

3 Přehled řešené problematiky

3.1 Bezdrátová komunikace

Bezdrátová komunikace je označována jako přenos informací prostřednictvím spojení dvou subjektů, které nejsou propojeny mechanicky. Nejběžnější bezdrátové technologie využívají rádiové vlny a tato technologie je vnímána jako jeden z oborů v telekomunikacích. Zahrnuje různé typy pevných, mobilních a přenosných aplikací, mobilních telefonů, kapesních počítačů (anglicky *Personal digital assistant*), GPS a bezdrátových sítí. Název bezdrátové připojení určuje připojení k počítačové sít (nejčastěji k internetu), které využívá bezdrátovou komunikaci. Tuto technologii bezdrátové komunikace objevil srbský vynálezce a fyzik Nikola Tesla.[1]

3.1.1 Wi-Fi

Wi-Fi je technologie popisující bezdrátovou komunikaci v počítačových sítích. Bezdrátové sítě jsou popisovány standardem IEEE 802.11x (písmeno *x* je označení verze). Funguje ve stejném frekvenčním pásmu jako Bluetooth, a tudíž se na jeho využití vztahují stejná omezení. Motivem vytvoření technologie Wi-Fi byla náhrada síťových kabelů, avšak původním cílem bylo zajištění vzájemného bezdrátového připojení přenosných zařízení a dále jejich propojování na lokální sítě LAN (anglicky *Local Area Netwrok*). Později Wi-Fi začala být využívána i k bezdrátovému připojení do sítě Internet v rámci rozsáhlejších lokalit. Tato technologie je prakticky ve všech přenosných zařízení (telefony, tablety, herní konzole, osobní počítače, digitální audio přehrávače, tiskárny).[2]

Bezdrátová síť může být vybudována několika způsoby v závislosti na požadované funkci. Avšak ve všech případech je součástí identifikátor SSID (anglicky *Service Set Identifier*), což je označení pro řetězec až 32 tzv. ASCII znaků, dle kterých se jednotlivé sítě rozlišují. Identifikátor bezdrátové sítě SSID je v pravidelných intervalech vysílán jako broadcast, tudíž si všichni uživatelé mohou zobrazit dostupné bezdrátové sítě, ke kterým je možné se připojit. Existují mezinárodní standardy, aby mezi sebou mohla komunikovat

[1] JENÍČEK, Vladimír. *Globalizace světového hospodářství*. Praha: C. H. Beck, 2002. 152 s. ISBN 978-80-7179-787-1.
[2] ZAORAL, Ondřej a Josef TKÁČ. *Průvodce světem kapesních počítačů: aneb PDA na dlani* [online]. Grada Publishing, 2005. 208 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=rbpaAgAAQBAJ

taková zařízení, která pochází od jiných výrobců a různých platforem. Institut IEEE (anglicky *Institute of Electrical and Electronic Engineers*), kde probíhá specifikace bezdrátových lokálních sítí, které jsou publikovány pod číslem 802.11. Nejčastější vyskytující se standardy jsou 802.11b (pracuje na frekvenčním pásmu 2,4 GHz při teoretické propustnosti 11 Mbit/s) a 802.11g (spojuje rychlost 802.11a a operační pásmo klasického Wi-Fi – nová zařízení 802.11g mohou komunikovat se starším Wi-Fi). Standard 802.11b má nejvyšší dosažitelnou rychlost 11 Mb/s a u standardu 802.11g je to až 54 Mb/s. [3] V Příloze I je zobrazena diference mezi standardy IEEE 802.11af a 802.11ah.

3.1.2 Bluetooth

Anglický název Bluetooth je odvozen podle dánského krále Haralda Blátanda II. (940-981 A.D.), jehož vlastní jméno bylo Harald Gromsson. Jeho přezdívka Harold Blátand vznikla kvůli jeho velké vášni k borůvkám, jelikož jeho zuby byly permanentně obarveny modrou barvou. Blátand (anglicky *Bluetooth*) znamená v překladu modrý zub. Harold využil svých diplomatických schopností a sjednotil velkou část území dnešního Norska, Švédska a Dánska. Právě této analogie bylo využito pro pojmenování technologie Bluetooth sloužící k usnadnění vzájemné komunikace.[4]

Jedná se o technologii pro budování personálních sítí LAN, která je založená na bezpečných spojeních se zařízeními na malou vzdálenost. Nejznámější aplikací Bluetooth je propojení mobilního telefonu s náhlavní soupravou, ovšem je možnost spojení kancelářské tiskárny, klávesnice, PDA a GPS zařízení, čtečky čárového kódu apod. Jednotlivé složky v síti Bluetooth mohou identifikovat a pak komunikovat s různými Bluetooth zařízeními. Standard pro Bluetooth je vyvíjen skupinou, která se nazývá Bluetooth Special Interest Group.[5] Tato technologie bezdrátové komunikace Bluetooth je podobná technologii, která je využívána v mobilních zařízeních – je označována jako pokrytí spektra frekvenčními toky (anglicky *frequency-hopping spread spectrum*). Ve Spojených státech amerických a v Evropě se využívá pásmo 2,45 GHz, které je

[3] CAFOUREK, Bohdan. *Windows 7: kompletní příručka*. Praha: Grada, 2010. Profesionál. 326 s. ISBN 978-80-247-3209-1.
[4] PRABHU, C.S.R. a A. Prathap REDDI. *Bluetooth technology and its applications with JAVA and J2ME* [online]. Eastern economy ed. New Delhi: Prentice-Hall of India, 2006. 340 s. [cit. 2018-02-18]. Dostupné z: http://bit.ly/2oIJqar
[5] SOSINSKY, Barrie. *Mistrovství – počítačové sítě* [online]. Albatros Media a.s., 2016. 840 s. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=qwbqCwAAQBAJ

považováno za volné, a na něm vysílá několik zařízení. Se stejným frekvenčním rozsahem pracují také bezdrátové sítě 802.11 g, mobilní zařízení a další. Na této frekvenci ovšem vyzařují radiaci i mikrovlnné trouby, a proto mohou rušit mobilní telefony, bezdrátové sítě, ale i technologii Bluetooth.[6]

Přesný rozsah dané frekvence ve Spojených státech je stanoven na 2400 až 2483,6 MHz. Toto pásmo je dále děleno na 79 neobvyklých kanálů o šířce 1 MHz. Fyzická spojení mezi jednotlivými prvky mohou mít rychlost až 1 Mb/s a je to dosaženo technikou tzv. Gaussovského klíčování posunem frekvencí (anglicky *Gaussian Frequency-Shift Keying*).[7]

Zařízení Bluetooth obsahují přijímače rozčleněné do tří tříd

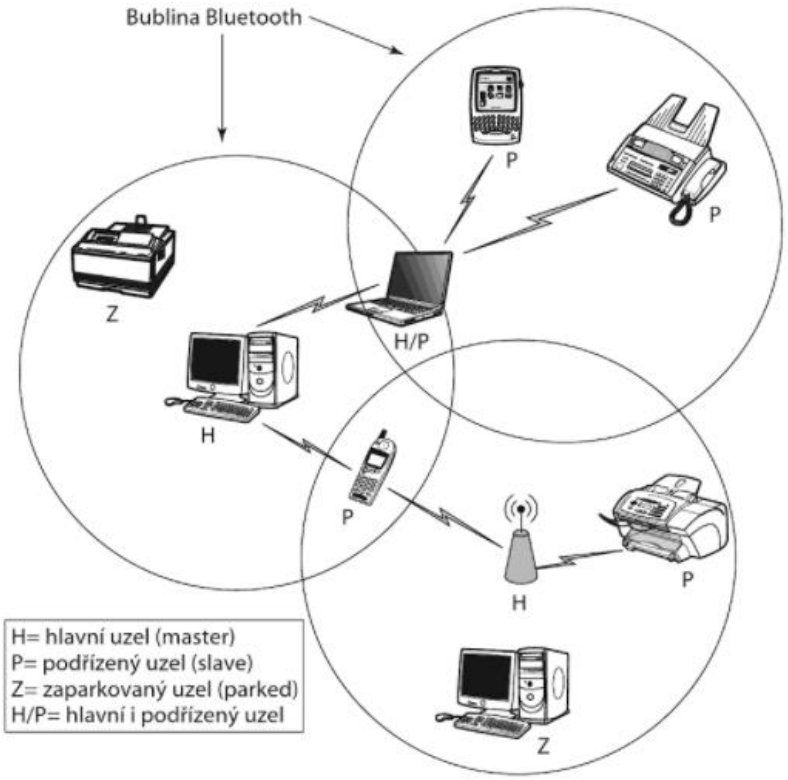
- 1. třída – 100 mW s dosahem 100 metrů
- 2. třída – 2,5 mW s dosahem 10 metrů
- 3. třída – 1 mW s dosahem jen jednoho metru[8]

Všechny tyto standardy jsou používány pro všesměrové vysílání a přijímání dat. Bluetooth díky nízkému výkonu signálu není schopen překonat zdivo, na rozdíl od mobilních zařízení, jejichž výkon se pohybuje okolo 3 wattů. Zdali se zařízení, které je určené ke kratšímu dosahu (např. 2. třídy), pokusí spojit se zařízením určený s delším dosahem (např. 1. třídy), je dosah méně výkonného zařízení o určený dosah prodloužen. Sítě Bluetooth jsou označovány za malou síť, která se značí tzv. pikonet. Pro jejich vybudování je potřeba rozbočovač (hub) Bluetooth, který má svůj vlastní vysílač i s přijímačem. Malá síť, pikonet, je definována jako ad hoc síť Bluetooth zařízení. Jedná se o decentralizovanou síť, ve které může každý uzel předávat data všem ostatním uzlům. Tento daný typ sítí je označován jako rozptýlená síť (anglicky *scaternett*). V mnoha přenosných počítačích, ale i dalších typech zařízení se vyskytuje již zabudovaný rozbočovač Bluetooth. Existují i Bluetooth rozbočovače, které se dají zapojit do USB

[6] POGUE, David. *Mac OS X Snow Leopard* [online]. Albatros Media, 2016. s. 952. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=SwvqCwAAQBAJ
[7] SOSINSKY, Barrie. *Mistrovství – počítačové sítě* [online]. Albatros Media a.s., 2016. 840 s. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=qwbqCwAAQBAJ
[8] What is the range of Bluetooth. *Scienceabc.com* [online]. [cit. 2018-02-23]. Dostupné z: https://www.scienceabc.com/innovation/what-is-the-range-of-bluetooth-and-how-can-it-be-extended.html

portu, do sběrnice či mají formu rozšiřující PCI karty. V sítí Bluetooth se lze vyskytovat nejvýše osm zařízení, které jsou navzájem propojené. Uživatelé mohou do sítě libovolně vstupovat či ji opustit. Bezdrátové sítě mají většinou tuto ad hoc podobu.[9]

Obrázek 1: Malá síť Bluetooth



Zdroj: SOSINSKY, Barrie. Mistrovství – počítačové sítě [online]. Albatros Media a.s., 2016. 840 s. [cit. 2018-02-18].
Dostupné z: https://books.google.cz/books?id=qwbqCwAAQBAJ

3.1.3 ZigBee

Jedná se o bezdrátovou komunikační technologii podobnou jako je Bluetooth, která je určena pro spojení nízko výkonových zařízení v sítích PAN na malé vzdálenosti do 75 metrů. ZigBee je vystavená na standardu IEEE 802.15.4 a je poměrně novým standardem, který se stal platným v listopadu 2004. Pomocí použití multiskokového ad hoc směrování je umožněna komunikace i na větší vzdálenosti bez přímé radiové viditelnosti jednotlivých zařízení. Hlavní určení směřuje do aplikací v průmyslu a senzorových sítí, pracuje v bez

[9] SOSINSKY, Barrie. *Mistrovství – počítačové sítě* [online]. Albatros Media a.s., 2016. 840 s. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=qwbqCwAAQBAJ

licenčních pásmech kolem 868 MHz, 902-928 MHz a také 2,4 GHz. Přenosová rychlost je uváděna 20, 40, 250 kbit/s.[10]

Technologie ZigBee patří do skupiny bezdrátových sítí PAN (anglicky *Personal Area Networks*) a je navržena jako jednoduchá a flexibilní technologie pro tvorbu rozsáhlejších bezdrátových sítí, u kterých není požadován přenos většího objemu dat. Tato technologie má několik vlastností, mezi které patří spolehlivost, jednoduchá a nenáročná implementace, velmi úsporná spotřeba energie a také příznivá cena. Pomocí těchto vlastností je technologie ZigBee uplatňována u několika aplikací, které lze zařadit do několika kategorií: automatizace budov (ovládání světel, kontrola přístupu, ochrana), počítačové periferie (klávesnice, bezdrátové myši), spotřební elektronika (elektrospotřebiče využívané pomocí dálkového ovládání), zdravotnictví (pacientské monitory). Standard je definován třemi základními režimy přenosu dat díky různorodosti předpokládaných aplikací. Prvním režimem přenosu dat je periodicky se opakující (přenos dat z čidel), druhý režim představuje nepravidelné přenosy (externí události, např. stisknutí tlačítka uživatelem) a posledním režimem jsou opakující se přenosy, u kterých je požadavek určen na malé zpoždění (bezdrátové počítačové periferie – klávesnice a myši).[11]

3.1.4 Mobilní sítě

Nejrozšířenější standard pro mobilní zařízení je buňková technologie GSM (francouzsky *Groupe Spécial Mobile*). V současnosti tuto technologii využívá více než 5 miliard uživatelů ve více než 200 zemích světa, avšak tyto čísla nejsou přesná, jelikož většina uživatelů vlastní více SIM karet. Skoro 1 miliarda uživatelů GSM/UMTS využívá datové služby 3G technologie W-CDMA, která vychází z CDMA. Jednou z primárních vlastností GSM je SIM (anglicky *Subsciber Identity Module*) karta – obsahuje informace potřebné k přihlášení uživatele do sítě a je v ní uložen telefonní seznam a SMS. Síť GSM je založena na tzv. celulární struktuře – lokace, která je obsluhovaná, je rozdělaná na buňky (cely). Velké množství sousedících buněk vytváří navzájem svazek, kde jsou

[10] ADÁMKOVÁ, Věra. *Hodnocení vybraných metod v kardiologii a angiologii pro praxi* [online]. Graga Publishing, 2016. 150 s. [cit. 2018-02-21]. Dostupné z: https://books.google.cz/books?id=hDgCDQAAQBAJ
[11] Tamtéž.

každé z nich přiděleny kmitočtové kanály, které se nesmějí reprodukovat u ostatních buněk téhož svazku.[12] [13] V Příloze II je uvedený princip buňkové mobilní sítě.

Systém GSM je označován jako veřejný radiotelefonní systém – je dostupný jednotlivým uživatelům, kteří jsou schopni navázat spojení prostřednictvím mobilních zařízení s účastníky mobilních sítí jak svého, tak i operátorů, ale také i s uživateli pevné veřejné telefonní sítě (datové sítě). Celý systém je složen ze tří základních prvků: systém základnových stanic, síťový a spínací systém a operační systém. Spojení, které prochází mezi mobilním zařízením a základnovou stanicí, probíhá prostřednictvím radiových vln šířících se vzduchem. V roce 1982 byla vytvořena specifikace pro panevropský digitální celulární radiotelefonní systém GSM, který probíhal na frekvenci 900 MHz. Později však byla rozšířena na frekvenci 1800 MHz – systém GSM je nyní rozšířen po celé Evropě a také je možné využívat sítě jednotlivých operátorů. Základní datová komunikační rychlost je 14,4 Kbit/s. Technologie zvaná přepínání okruhů, HSCSP (Anglicky *High Speed Cicuit Switched Data*), umožňuje datový přenos, který dosáhne rychlosti až 115,2 Kbit/s.[14]

3.1.5 Radiové moduly

Radiofrekvenční modul je obvykle menší elektronické zařízení, které je používáno pro vysílání či příjem rádiových signálů mezi dvěma zařízeními. Uvnitř zabudovaného systému je potřeba bezdrátové komunikace s jiným zařízením. Této bezdrátové komunikace může být dosaženo prostřednictvím optické komunikace či rádiových frekvencí. Většina aplikací využívá rádiové moduly obsahující vysílač s přijímačem (viz Příloha III). Elektronický rádiový design je notoricky složitý kvůli citlivosti radiových obvodů, přesnosti součástek a jejich uspořádání potřebné pro dosažení provozu na určité frekvenci. Kromě toho musí radiofrekvenční model vyžadovat pečlivé sledování při průběhu výrobního procesu, aby nedošlo k nepříznivému ovlivnění výkonu. Radiofrekvenční modely se nejčastěji využívají jako monitorovací systémy, otevíraní garážových vrat, dálkové ovládání inteligentní senzorové aplikace a automatizace

[12] RAAB, Stefan a Madhavi W. CHANDRA. *Cisco: mobilní IP technologie a aplikace*. Praha: Grada, 2007. 299 s. ISBN 80-247-1611-9.
[13] PAVEL, Burian. *Internet inteligentních aktivit* [online]. Praha: Grada, 2014. 336 s. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=ATqCBAAAQBAJ
[14] Tamtéž.

domácností. Několik nosných kmitočtů se běžně používá v komerčně dostupných radiofrekvenčních modulech, včetně pásem ISM (anglicky *industrial, scientific and medical)* – pásma pro rádiové vysílání v průmyslovém, zdravotnickém a vědeckém oboru (433.92 MHz, 915 MHz a 2400 MHz). Rádiové moduly mohou splňovat definovaný protokol pro komunikace ve vysokorychlostní síti, jako jsou ZigBee, Bluetooth či Wi-Fi, nebo si mohou vytvořit vlastní protokol.[15] [16]

3.2 Mikropočítač

Název mikropočítač se stal populárním až po zavedení minipočítače, kdy ve své povídce Umírající noc (anglicky *The Dying* Night) Isaac Asimov použil termín mikropočítač. Tato povídka byla publikována v roce 1956 v časopise *The Magazine Of Fantasy and Science Fiction*. Mikropočítač je označován jako zařízení, které je výrazně menší, než je stolní osobní počítač, obsahující mikroprocesor (je zapotřebí, aby vlastnil alespoň jednu centrální procesorovou jednotku, CPU), paměť a vstupní / výstupní zařízení.[17]

Mikrokontroler, označován také jako jednočipový počítač, je ve většině případů monolitický integrovaný obvod, který obsahuje kompletní mikropočítač. Ovšem mikrokontroler sám o sobě nemá praktický přínos. Propojili se s externími periferiemi či zařízením, pak poskytuje přidanou hodnotu v podobě inteligentního zařízení. Mikrokontroler je právě prostředníkem v procesu komunikace mezi jedincem a reálným světem. Veškeré informace načítány mikrokontrolerem jsou získány z externích modulů, zařízení či snímačů. Ty jsou následně zpracovány dle definované logiky a výsledky přepošle na určitý výstup. Na základě získaných informací se lze rozhodnout o případné změně, jako je například změna rychlosti otáček či vypnutí osvětlení. Pomocí vstupních rozhraní jsou nastaveny požadované změny, které jsou dále přeposlány do mikrokontroleru. Jako první jsou popisovány displeje jako výstupní a klávesnice jako vstupní rozhraní při dané komunikaci s uživatelem. Do zařízení jsou implementované

[15] TUTTLEBEE, Walter. *Software Defined Radio Enabling Technologies*. Chichester: John Wiley, 2003. 440 s. ISBN 978-0-470-85263-7.
[16] WILLIAMS, Tim. *EMC for Product Designers*. 4. vyd. Burlington: Elsevier, 2007. ISBN 978-0-08-046954-6.
[17] UPTON, Eben a Gareth HALFACREE. *Raspberry Pi* [online]. Praha: Albatros, 2017. 232 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=eQ7qCwAAQBAJ

snímače, aby mohly být získány potřebná data. Jednočipové počítače jsou často součástí vestavěných systémů. Za mikrokontroler je možno označit i hlavní integrovaný obvod v nynějších mobilních zařízení.[18] Viz Příloha IV, kde je zobrazen mikrokontroler AVR.

3.2.1 Arduino

Arduino je fenoménem posledních let a jedná se o jednodeskový počítač založený na mikrokontrolerech ATmega. V období 2005 až 2013 se prodalo více jak 700 000 oficiálních zařízení Arduino. Jednodeskový počítač sám o sobě nepřináší na trh nic nového, mikroprocesory, propojení spínačů, LED diod, různých aktivních a pasivních součástek, tohle vše bylo možné využít již dávno předtím. Dokumentace k určitým mikroprocesorům, schémata zapojení, programovací rozhraní či platformy jako BASIC Stamp 1 již existovaly desítky let dříve. Ovšem platforma Arduino je založena na sociální inovaci. Došlo tak ke zjednodušení programování mikrokontrolerů prostřednictvím IDE, která se stará o všechno na pozadí, a to bez toho, aby uživatel nikterak zasahoval.[19]

3.2.1.1 Arduino Uno

Arduino Uno je mikroprocesorová deska založená na ATmega328P. Má 14 digitálních vstupních / výstupních pinů (z nichž 6 jich lze použít jako výstupy PWM), 6 analogových vstupů, 16 MHz frekvence kmitání křemenného krystalu, USB připojení, napájecí konektor, záhlaví ICSP a tlačítko pro restart. Deska Uno je první ze série desek USB Arduino a referenčních modelů pro platformu Arduino. ATmega328 na Arduino Uno je před programován s bootloaderem, který mu umožňuje nahrát nový kód bez použití externího hardwarového programu. Komunikuje pomocí původního protokolu STK 500. Arduino Uno se liší od všech předchozích desek tím, že nepoužívá čip FTDI USB. Místo toho obsahuje Atmega16U2, který je naprogramován jako sériový port.[20]

[18] SELECKÝ, Matúš. *Arduino* [online]. Praha: Albatros, 2016. 344 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=fyq7DQAAQBAJ
[19] PEREA, Francis. *Arduino Essentials* [online]. Packt Publishing, 2015. 206 s. [cit. 2018-02-19]. Dostupné z: https://books.google.cz/books?id=NrjNBgAAQBAJ
[20] Tamtéž.

Obrázek 2: Arduino Uno REV3

Zdroj: Arduino Uno REV3. *Store.arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

Desku Arduino Uno (viz Příloha V) lze napájet přes USB či přes externí napájecí zdroj. Zdroj energie je automaticky vybrán. Externí napájení (mimo USB) vychází z adaptéru střídavého proudu na stejnoměrný proud nebo z baterie. Deska pak může pracovat na externím zdroji od 6 do 20 voltů. Pokud je zdroji dodáváno méně než 7 voltů, tak se deska může stát nestabilní. Avšak pokud se využívá více jak 12 voltů, tak se regulátor napětí bude přehřívat a může značně poškodit desku. Doporučený rozsah je 7 až 12 voltů. Co se týče paměti, tak ATmega328 vlastní 32 kB (bootloader obsahuje 0,5 kB). Obsahuje také 2 kB paměti SCRAM a 1 kB paměti EEPROM. Každý z 14 digitálních pinů na jednotce Uno může být použit jako vstup či výstup prostřednictvím funkcí pinMode (), digitalWrite () a digitalRead () pracující na 5 voltech (viz Příloha VI). Každý pin může poskytovat nebo přijímat 20 mA a obsahuje vnitřní odpínač (je standardně odpojený). Maximální hodnota je 40 mA, která nesmí být překročena, aby bylo zabráněno poškození mikrokontroleru.[21] Arduino má řadu zařízení pro komunikaci s počítačem. ATmega328 poskytuje sériovou komunikaci UART TTL, která je k dispozici na digitálních pinech 0 (RX) a 1 (TX). ATmega16U2 na desce vysílá tuto sériovou komunikaci přes USB a tím vzniká tzv. virtuální port, který je přijímán počítačovým softwarem. Firmware 16U2

[21] CHOUDRUHI, Kallol. *Learn Arduino Prototyping in 10 days* [online]. Packt Publishing, 2017. 288 s. [cit. 2018-02-21]. Dostupné z: https://books.google.cz/books?id=3nc5DwAAQBAJ

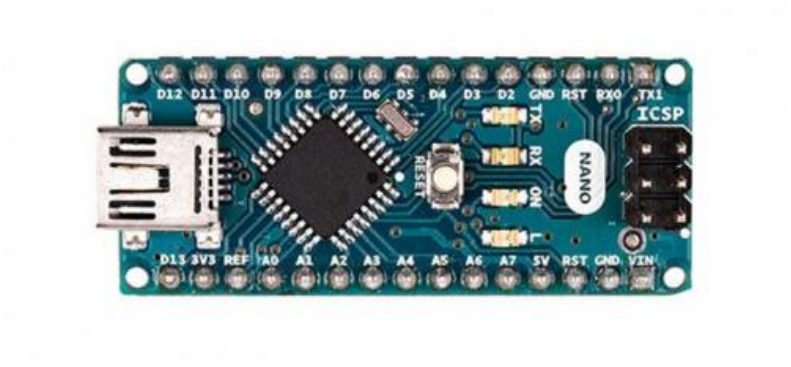
využívá standardní ovladače USB a není tak potřeba využívat externího. V systému Windows je však vyžadován soubor s koncovkou *.inf*. Software Arduino obsahuje sériový monitor umožňující odesílání jednoduchých textových dat jak do desky Arduino Uno, ale také z ní. LED diody RX a TX pak začnou blikat, pokud budou data přenášena.[22]

3.2.1.2 Arduino Nano

Arduino Nano je malá a kompletní deska založená na ATmega328. Má víceméně stejnou funkcionalitu jako Arduino Duemilanove. Lze ho napájet buď přes USB mini, neregulovaný externí napájecí zdroj 6-20 voltů (pin 30) či regulované externí napájení 5 voltů (pin 27). Zdroj energie je automaticky vybrán na nejvyšší zdroj napětí. ATmega328 obsahuje 32 kB (bootloader má 2 kB). Mikrokontroler obsahuje také 2 kB SRAM a 1 kB EEPROM. Stejně jako u Arduino Uno může být každých 14 digitálních pinů použito jako vstup či výstup pomocí funkce pinMode (), digitalWrite () a difitalRead (), které pracují na 5 voltech. Nano má 8 analogových vstupů, z nichž každý poskytuje rozlišení 10 bitů. Ve výchozím nastavení měří až k 5 voltům, ačkoli je možné změnit horní hodnotu a rozsah jako takový pomocí funkce analogReference (). Analogové piny 6 a 7 nejdou použít jako digitální piny, a také některé z nich mají specifické funkce.[23]

[22] DESAI, Pratik. *Python Programming for Arduino* [online]. Packt Publishing, 2015. 400 s. [cit. 2018-02-21]. Dostupné z: https://books.google.cz/books?id=O0PfBgAAQBAJ
[23] PEREA, Francis. *Arduino Essentials* [online]. Packt Publishing, 2015. 206 s. [cit. 2018-02-19]. Dostupné z: https://books.google.cz/books?id=NrjNBgAAQBAJ

Obrázek 3: Arduino Nano



Zdroj: Arduino Nano. *Store.arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://store.arduino.cc/arduino-nano

Arduino Nano má řadu zařízení pro komunikaci s počítačem – Armega328 poskytuje sériovou komunikaci UART TTL (5 voltů), která je k dispozici na digitálních pinech 0 (RX) a 1 (TX). FTDI FT232RL na desce Arduino Nano vysílá tuto sériovou komunikaci přes USB a ovladače FTDI. Software Arduino IDE obsahuje sériový monitor, který zasílá jednoduchá textová data do desky Arduino Nano – LED diody RX a TX začnou blikat, až budou data přenesena prostřednictvím čipu FTDI.[24]

3.2.2 Raspberry Pi

Raspberry Pi je malý jednodeskový počítač, který má jako primární operační systém Rasbian. Jádro systému Raspberry Pi je multimediální procesor typu Soc (anglicky *System-on-chip*) Broadcom BCM 2835. Většina systémových komponent (včetně jeho hlavního a grafického procesoru spolu se zvukovým a komunikačním hardwarem) jsou integrovány do jediné součástky, která je ukrytá pod paměťovým čipem obsahující kapacitu 256 MB uprostřed základní desky. Procesor BCM 2835 se od ostatních procesorů, kterými jsou obsaženy stolní počítače či přenosná zařízení, neliší pouze svým návrhem typu *system-on-chip* je využita jiná architektura instrukční sady ISA (anglicky *instruction set architecture*), která je označována jako ARM. V 80. letech vyvinula architekturu ARM společnost Acorn Computers, avšak tato architektura se v počítačových zařízeních

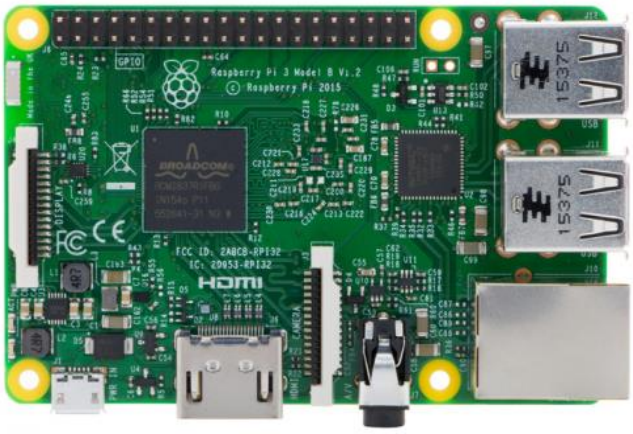
[24] KARVINEN, Tero a Kimmo KARVINEN. *Make Six Embedded Projects with Open Source Hardware and Software.* Sebastopol: O'Reilly Media, 2011. 296 s. ISBN 9781449307233.

objevuje poměrně zřídka. Ovšem v mobilních telefonech se tento typ architektury využívá.[25]

3.2.2.1 Raspberry Pi 3 Model B

V roce 2016 byl tento model uveden do prodeje. Je vybaven CPU vlastnící čtyřjádrový procesor pracující na frekvenci 1 200 MHz (1,2 GHz). Jedná se o ARM Cortex-A53, který je dvojnásobně rychlejší, než CPU ve starším modelu Raspberry Pi 2. Obsahuje také 40 pinů GPIO (vstupy / výstupy) umožňující připojení různých zařízení jako jsou senzory či tlačítka do modelu Raspberry Pi. Tyto piny vyčnívají z desky, takže je možné spojení propojky bez napájení. Dalším komponentem je DSI (anglicky *Display Seriál Interface*) vysokorychlostní sériové rozhraní pro připojení displeje (např. dotyková obrazovka). Hlavním úložištěm pro Raspberry Pi Model B je Micro SD slot. Deska přijímá napájení ze síťového adaptéru prostřednictvím standardního portu micro USB. Obdobně jako předešlý model Raspberry Pi 2 je vybaven 1 GiB operační pamětí. Kromě 64bitového procesoru jsou v modelu zabudované integrované Wi-Fi a Bluetooth moduly.[26]

Obrázek 4: Raspberry Pi 3 Model B



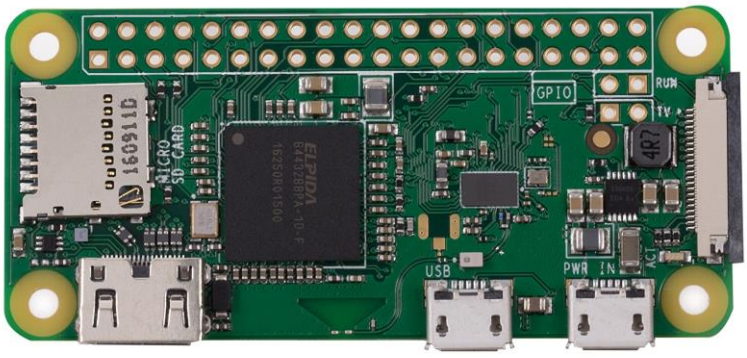
Zdroj: Raspberry Pi 3 Model B. *Element14.com* [online]. [cit. 2018-02-27]. Dostupné z: https://www.element14.com/community/docs/DOC-81294/l/raspberry-pi-3-model-b-with-1gb-of-ram-with-wifi-and-bluetooth-low-energy

[25] UPTON, Eben a Gareth HALFACREE. *Raspberry Pi* [online]. Praha: Albatros, 2017. 232 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=eQ7qCwAAQBAJ
[26] HART-DAVIS, Guy. *Deploying Raspberry Pi in the classroom* [online]. California: Apress/Springer Science+Business Media Finance, 2017. 293 s. [cit. 2018-02-23]. Dostupné z: https://books.google.cz/books?id=-fjFDQAAQBAJ

3.2.2.2 Raspberry Pi Zero W

Začátkem roku 2017 společnost Raspberry představila novou desku s bezdrátovým rozšířením. Tato deska nabízí bezdrátovou komunikaci a nyní může být využívána bez kabelů či jiných komponent. V porovnání s deskou Raspberry Pi 3 Model B je značné velikostní zmenšení (65 mm x 30 mm x 5 mm) a má daleko více možností pro Internet. Avšak liší se zásadní věcí, má integrovanou Wi-Fi kartu, se kterou je možné komunikovat s okolím. Jedná se tedy o malé zařízení, které lze připojit k externímu monitoru či televizoru, ale také k internetu. Tento model umožňuje snadné programování pomocí pinů GPIO a dalších komponentů, jako je například fotoaparát. Raspberry Pi Zero W je vybaven čipovou jednotkou Cypress CYW43438, která poskytuje připojení 800.11n bezdrátovou lokální síť a Bluetooth 4.0. Je vybaven jednojádrovým CPU (ARM11) pracující na frekvenci 1 GHz a paměť RAM obsahující 512 MB. Video a zvuk jsou přenášeny přes mini HDMI konektor. Dále je vybaven dvojicí micro USB (z nichž jedna je určená pro napájení) a čtečkou karet MicroSD.[27]

Obrázek 5: Raspberry Pi Zero W



Zdroj: Raspberry Pi má novou variantu. Cnews.cz [online]. [cit. 2018-02-27]. Dostupné z: https://www.cnews.cz/raspberry-pi-ma-novou-variantu-zero-w-ma-integrovane-wi-fi-bluetooth-radio-za-10/

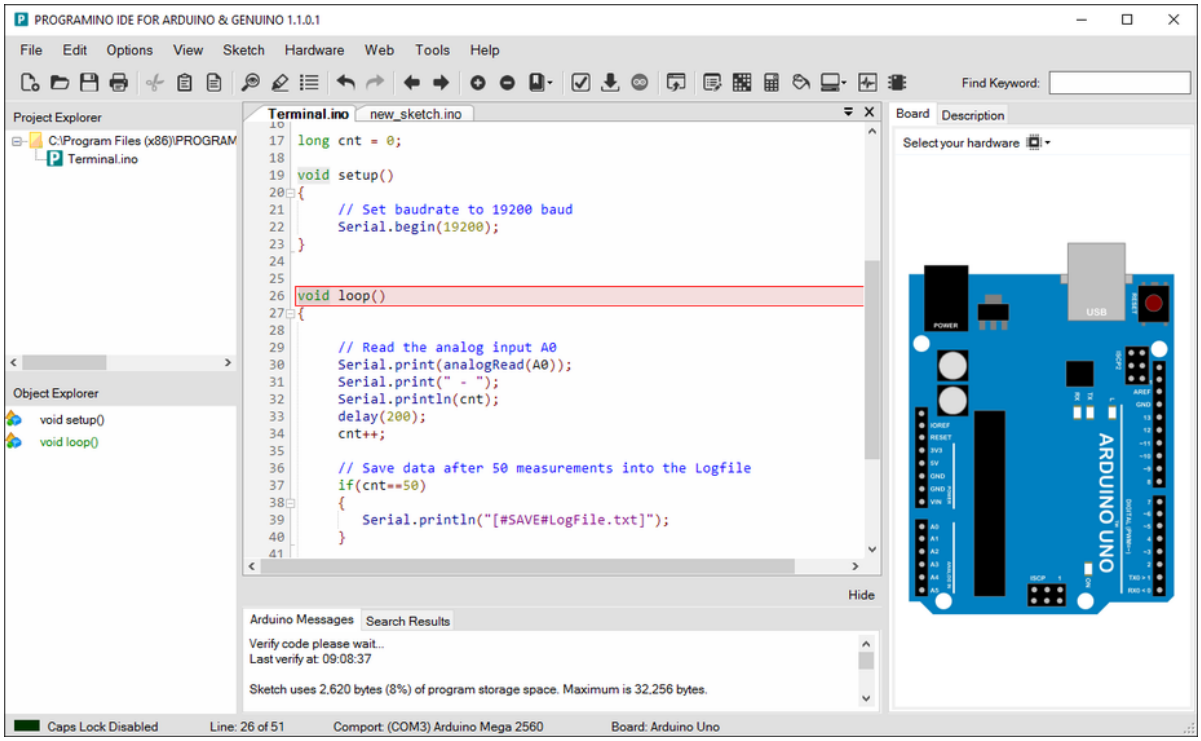
[27] TZIVARAS, Vasilis. *Raspberry Pi Zero W Wireless Projects* [online]. Packt Publishing, 2017. 240 s. [cit. 2018-02-25]. Dostupné z: https://books.google.cz/books?id=fpZGDwAAQBAJ

3.3 Software Arduino IDE

Vývojové prostředí Arduino se skládá z textového editoru po psaní kódu, textové konzole, lišty s příkazy pro obvyklé funkce a několik řad nabídek. Připojuje se na hardware Arduino, kam nahrává programy a navzájem komunikují. V textovém editoru se píšou tzv. sketch (návrhy), které se ukládají do souborů s koncovkou *.ino*. Textový editor obsahuje jak funkce pro vyjímání, tak i pro vkládání, ale i hledání či nahrazování textu. Na konzoli jsou zobrazovány textové výstupy z prostředí Arduino včetně kompletních chybových zpráv a dalších informací. Prostředí Arduino využívá konceptu sketchbook – standardní prostor pro ukládání programů. Při prvním spuštění softwaru Arduino IDE se automaticky vytvoří adresář pro sketchbook. Zobrazení či změna jeho umístění lze v dialogu Preferences. Od verze 1.0 jsou veškeré soubory ukládány s koncovkou *.ino*. Starší verze využívají koncovku *.pde*. V novějších verzích je možné otevírat soubory s koncovkou *.pde*, ale program je automaticky přejmenuje na koncovku *.ino*. Knihovny neboli Libraries nabízejí kvalitnější funkce pro sketche, aby lépe fungovaly s hardwarem či manipulovaly s daty. Do vrchní části sketche se přidá několik dalších příkazů #include a kompiluje se knihovna se sketchem. Jelikož se knihovny nahrávají na desku spolu se sketchem, zabírají tím více místa. Ve chvíli, kdy daný sketch knihovnu nevyužívá, vymaže všechny #include ve vrchní části kódu.[28]

[28] SELECKÝ, Matúš. *Arduino* [online]. Praha: Albatros, 2016. 344 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=fyq7DQAAQBAJ

Obrázek 6: Prostředí programu Arduino IDE

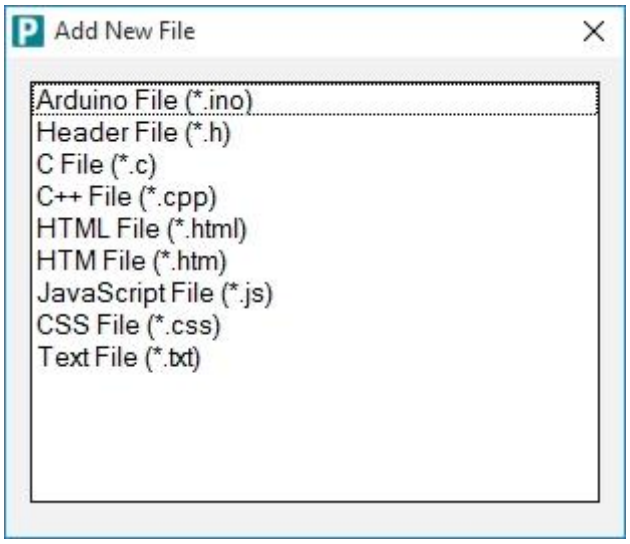


Zdroj: MCU – mikroelektronika. *Mcu.cz* [online]. [cit. 2018-02-24]. Dostupné z: http://mcu.cz/comment-n3958.html

Softwarové prostředí Arduino 1.0.1 bylo přeloženo do několika jazyků. Výchozí nastavení se v IDE nahrává v jazyce, který je nastavený v operačním systému. Při volbě desek má důsledky dvojího typu. Nastavují se parametry, jako je rychlost CPU, které se pak využívají při kompilaci sketchů a určují tak nastavení souborů a pojistek v příkazu burn bootloader. Program Arduino IDE podporuje následující programovací jazyky a soubory (viz Obrázek 7).[29]

[29] Arduino IDE. *Arduino.cz* [online]. [cit. 2018-02-24]. Dostupné z: https://arduino.cz/arduino-ide/

Obrázek 7: Podporované jazyky a soubory



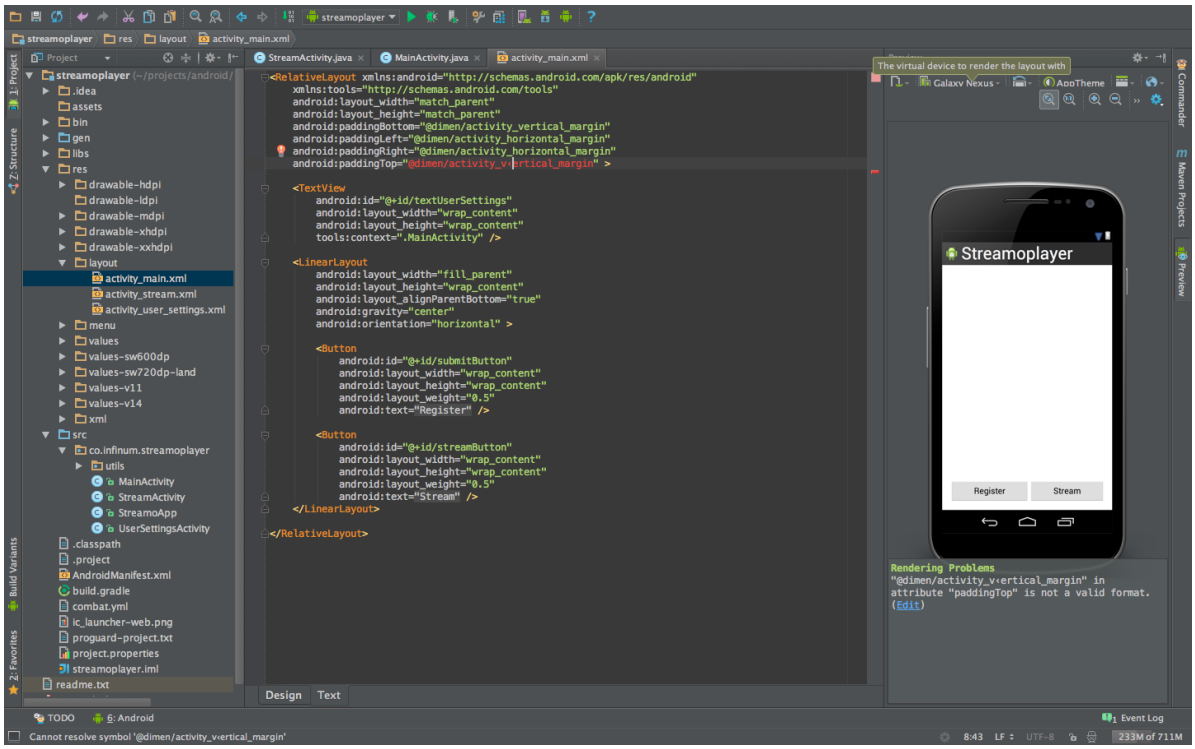
Zdroj: MCU – mikroelektronika. *Mcu.cz* [online]. [cit. 2018-02-24]. Dostupné z: http://mcu.cz/comment-n3958.html

3.4 Android Studio

Android Stuido je oficiálním integrovaným vývojovým prostředím (anglicky *Integrated development environment*) určené pro vývoj aplikací pro Android založené na technologii IntelliJ IDEA. Studio je k dispozici ke stažení v operačních systémech jako jsou Windows, MacOS a Linux. Jedná se též o nahrazení vývojového prostředí Eclispe jako primární IDE pro vývoj aplikací Android. První stabilní verze Android Studia byla spuštěna v prosinci roku 2014 počínaje verzí 1.0. Současná stabilní verze 3.0 byla vydána v říjnu roku 2017 a nyní je k dispozici verze 3.1, která je však v Beta verzi. Každý projekt v aplikaci Android Studio obsahuje jeden či více modulů se soubory zdrojového kódu. Tyto jsou zahrnovány moduly aplikace Android, moduly knihovny a moduly Google App Engine. Software Android Studio zobrazuje ve výchozím nastavení soubory projektu. Tento pohled zobrazení je uspořádán prostřednictvím modulů, které umožňují rychlý přístup ke klíčovým zdrojovým souborům projektu. Veškeré soubory jsou viditelné na nejvyšší vrstvě pod skripty Gradle a každý modul obsahuje složky manifests (obsahuje AndroidManifest.xml soubor), java (obsahuje soubory zdrojového kódu jazyku Java,

včetně testovacího kódu JUnit), res (obsahuje veškeré nekódové zdroje, jako jsou XML, řetězce uživatelského rozhraní a bitmapové obrázky).[30]

Obrázek 8: Vývojové prostředí Android Studia



Zdroj: Android Studio. *Infinum.co* [online]. [cit. 2018-02-27]. Dostupné z: https://infinum.co/the-capsized-eight/android-studio-vs-eclipse-1-0

Software Android Studio podporuje řadu systémů pro správu verzí, včetně Git, GitHub, CVS, Mercurial a Google Cloud Source. Je využívána platformou Gradle jako základ systému stavění – systém je vybaven integrovaným nástrojem z nabídky Android Studio a nezávisle na příkazovém řádku. Použitím flexibilnosti systému Gradle je možné dosažení bez veškerých úprav zdrojových souborů aplikace. Soubory vytvořené v aplikaci jsou pojmenovány *build.gradle*. Jedná se o soubory prostého textu, které využívají Groovy syntaxi k nakonfigurování sestavení, a to i s prvky pluginu Android pro Gradle.[31]

[30] Meet Android Studio. *Developer.android.com* [online]. [cit. 2018-02-27]. Dostupné z: https://developer.android.com/studio/intro/index.html
[31] Tamtéž.

3.5 Programovací jazyky

Jedná se o prostředek pro zápis algoritmů, které mohou být provedeny na počítačovém zařízení. Zápis algoritmu, který je uveden v programovacím jazyce, se nazývá program. Programovací jazyky se obvykle skládají ze strojových kódů pro počítač. Programovací jazyk lze tedy použít pro vytváření programů, které implementují specifické algoritmy.[32]

3.5.1 C

„Jazyk C poskytuje základní konstrukce pro řízení běhu, které jsou nezbytné pro správně strukturované programy: seskupování příkazů, výběr z množiny eventuálních případů (switch), rozhodování (if-else), cykly s testem ukončení na začátku (while, for) či na závěru (do) a předčasný skok z cyklu (break).“ [33]

C je označován jako univerzální programovací jazyk, který se vyznačuje úspornými výrazy, moderními datovými strukturami a obsáhlou množinou operátorů. Není specializován pro konkrétní oblast nasazení, ale neúčast omezení a jeho všeobecnost ho dělají vhodnějším a účinnějším pro většinu úloh než některé jiné jazyky. Jazyk C byl navržen, ale také implementován Dennisem Ritchiem na operačním systému UNIX na počítačovém zařízení DEC PDP-11. Operační systém, kompilátor programovacího jazyka C, a také veškeré aplikace pro UNIX byly napsány v jazyce C. Ovšem programovací jazyk C není spojen s konkrétním hardwarem či systémem a je tedy snadné napsat programy, které budou fungovat na jakémkoliv počítačovém zařízení podporující jazyk C. Většina relevantních myšlenek jazyka C vychází z jazyka BCPI, který byl vyvinut Martinem Richardsem. Ovlivňování BCPL na jazyk C se vyvíjelo nepřímo skrze jazyk B, který byl vytvořen Kenem Thompsonem roku 1970, a to pro první systém UNIX na počítačovém zařízení DEC PDP-7.[34]

[32] SCOTT, Michael Lee. *Programming language pragmatics* [online]. Waltham, MA: Morgan Kaufmann, an imprint of Elsevier, 2016. 992 s. [cit. 2018-02-26]. Dostupné z: https://books.google.cz/books?id=jM-cBAAAQBAJ
[33] KERNIGHAN, W. Brian a Dennis M. RITCHIE. *Programovací jazyk C* [online]. Praha: Albatros, 2017. 288 s. [cit. 2018-02-26]. Dostupné z: https://books.google.cz/books?id=WhDqCwAAQBAJ
[34] Tamtéž.

3.5.2 C++

„Hlavním důvodem vzniku C++ byla potřeba objektově orientovaného programování. Jazyk však poskytuje i jiná příjemná rozšíření, o nichž je dobré vědět. Není však pouhou nadmnožinou jazyka C. Existují konstrukce, které jsou správné v C, ale nikoliv v C++, i konstrukce syntakticky správné v obou jazycích, ale v každém s jiným významem.“ [35]

Programovací jazyk C++ není pouhé rozšíření jazyka C, protože při návrhu jazyka C++ byla komptabilita s jazykem C brána v potaz, avšak nikoliv za každou cenu. V jazyce C jsou konstrukce, které nejsou povoleny v jazyce C++. O prvních zmínkách k cestě jazyku C++ bylo tzv. „C s třídami“. Roku 1985 byla vyvinuta verze, která byla považována za řádný objektově orientovaný programovací jazyk. Jazyk C++ se od této doby dostal na takovou úroveň, že se stal stejně úspěšný jako programovací jazyk C, avšak jedním z primárních důvodů takového úspěchu byla téměř jistá komptabilita s jazykem C, která usnadňovala práci programátorům pracujících v C a zaručila tak použití existujících kódů, které jsou napsané v programovacím jazyce C. V programovacím jazyce C++ lze využívat vedle komentáře ohraničeného znaky */*a*/* také komentář, který začíná znaky // a končí přechodem na další řádek. Je zde zaveden nový datový typ *bool*, který je určen pro logické hodnoty. Hodnoty typu *bool* jsou vytvářeny pomocí relačních a logických operátorů. Deklarace proměnné datového typu struktura nemusí už mít klíčové slovo struct, které je uváděno před názvem struktury. V současné době tento programovací jazyk C++ patří mezi nejrozšířenější.[36] V Příloze VIII je na příkladu zobrazen programovací jazyk C++ s využitím sady Visual Studio 2017.

3.5.3 Java

V roce 1991 vymysleli programovací jazyk Java pánové James Gosling, Patrick Naughton, Chris Warth, Ed Frank a Mike Sheridan, kteří byli ze společnosti Sun Microsystems. Tento programovací jazyk byl původně nazýván „*Oak*“, avšak v roce 1995 byl přejmenován na název Java. V době, kdy se Java vyvíjela, tak se objevil relevantní faktor, který hrál zásadní roli v její budoucnosti. Hlavním faktorem byla celosvětová

[35] PROKOP, Jiří. *Algoritmy v jazyku C a C++*. 2., rozš. a aktualiz. vyd. Praha: Grada, 2012. Průvodce (Grada). 169 s. ISBN 978-80-247-3929-8.
[36] Tamtéž.

webová síť, jelikož díky ní se Java dostala do popředí návrhu počítačových jazyků, jelikož i samotný web vyžadoval přenositelné programy. Programovací jazyk Java přímo souvisí s jazyky C a C++, jelikož syntaxi zdědil z jazyka C a svůj objektový model byl převzat z jazyka C++. Spojitost Javy a jazyků C a C++ je relevantní již z několika důvodů – řada programátorů zná syntaxi jazyka C a C++, tudíž se pak dokáží naučit jazyk Java, a naopak programátoři v jazyce Java se snadněji naučí jazyky C a C++. Díky využití těchto jazyků navrhuje Java silné a logicky konzistentní programovací prostředí, které přidává nové vlastnosti vyžadované online prostředím. Java má tedy společné znaky s programovacími jazyky C a C++, jelikož byla navržena a testována činnými programátory. Java je založena na potřebách a zkušenostech programátorů, kteří jej vymysleli.[37] V příloze IX je na příkladu zobrazen programovací jazyk Java.

3.5.4 XML

XML (anglicky *Extensible Markup* Language) je obecný značkovací jazyk – je zjednodušenou formou staršího jazyka SGML. XML data jsou zahrnovány elementy, textovými daty, atributy a některé speciální prvky, jako jsou například komentáře, sekce CDATA apod. Primární síla jazyka XML je v možnosti specifikace sady přípustných značek a jejich konkrétní strukturu, která se přiřazuje k jednotlivým XML dokumentům. Jestli je daný dokument obsažen pouze značkami z konkrétní definované sady s odpovídající strukturou, je vůči této sadě validní či se jedná o její instanci. Popis této dané sady je označováno jako XML schéma určitého XML dokumentu. Tedy XML schéma zahrnuje popis přípustné struktury XML dokumentů. Součástí specifikace jazyka XML je jazyk DTD, který je určený pro definici XML schématu. Ovšem pro komplikovanější aplikaci či při přesnější vyjádření přípustné struktury XML dokumentů je jazyk DTD nedostačující. Tento problém však řeší jazyk nazvaný XML Schema, který byl vytvořen sdružením W3C a je obsažen početným množstvím nových konstruktů a jeho vyjadřovací síla je výrazně větší. V průběhu vývoje vznikly i další jazyky, které definovaly XML schéma, jako je například RELAX NG, Schematron apod., z nichž každý jazyk má své vlastní specifické vlastnosti a omezení.[38]

[37] SCHILDT, Herbert. *Java 7* [online]. Praha: Albatros, 2016. 664 s. [cit. 2018-02-26]. Dostupné z: https://books.google.cz/books?id=SArqCwAAQBAJ

[38] MLÝNKOVÁ, Irena. *XML technologie: Principy a aplikace v praxi.* Praha: Grada Publishing, 2008. 272 s. ISBN 978-80-247-6688-1.

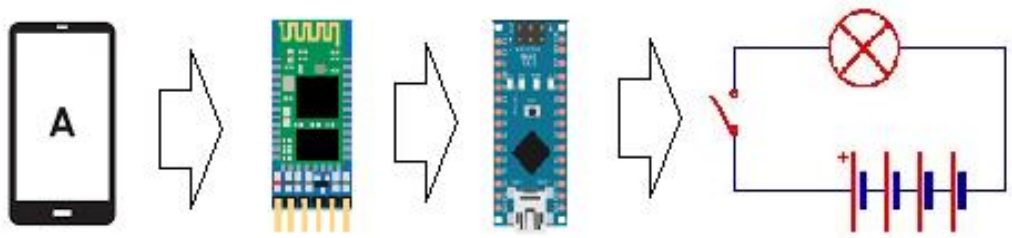
4 Analytická část

Hlavní cílem bakalářské práce je navržení a realizace funkčního komplexního řešení, za účelem přerušení napájení pro koncové zařízení ovládaného mikropočítačem, který přijímá instrukce pomocí připojeného Bluetooth modulu od mobilního zařízení.

4.1 Návrh řešení

Bylo vytvořeno takové komplexní řešení, které je na straně mobilní aplikace jednoduché a intuitivní. Hardwarové řešení bylo z praktického hlediska vytvořeno pro finální využitelnost a funkčnost v minimalistickém měřítku a za tímto účelem byly voleny i dílčí komponenty, které splňují požadavky pro dané řešení.

Obrázek 9: Proces funkčnosti řešení



Zdroj: Vlastní zpracování

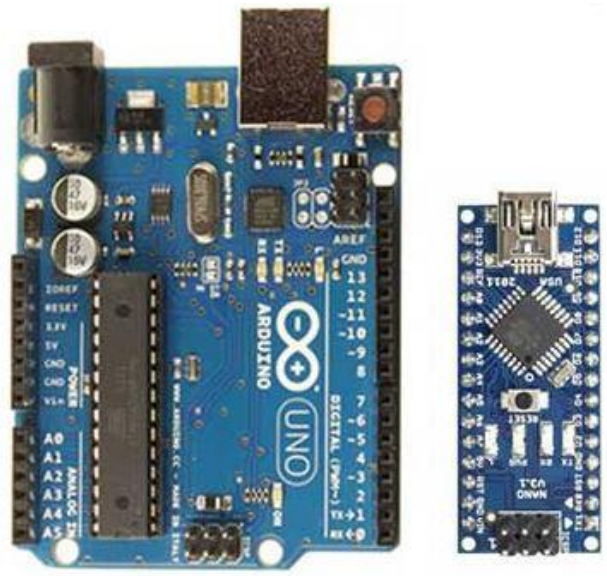
Výběr hardwarových komponentů:

- Arduino Nano
- Bluetooth modul HC-05
- Spínací relé – Arduino relé 2 kanály

Pro sestavení hardwarového obvodu byly zvoleny výše uvedené komponenty. Jako řídící jednotka byla zvolena deska Arduino Nano, jejíž parametry jsou uvedeny v teoretické části. Pro cíl práce z hlediska funkčnosti nehrálo roli, zda bude zvolena deska Aurduino Nano, nebo například deska Arduino Uno. Postačila by jakákoli deska podporující digitální piny a s kompatibilitou s Bluetooth modulem. Deska Arduino Nano byla vybrána, za účelem minimalizace celého obvodu, čistě pro svoji velikost. Na základě kompatibility byl pro komunikaci vybrán Bluetooth modul HC-05, který splňuje základní

parametry pro komunikaci mezi mobilním zařízením a mikropočítačem. V případě, že některá koncová zařízení nebyla napájena přímo z desky Arduino Nano, bylo vybráno za účelem ovládání napájení vnějšího elektrického obvodu dvou kanálové relé, které umožní přepínat napětí až do 230 V. S těmito komponenty by mělo být pokryto široké spektrum uplatnění výsledného zařízení.

Obrázek 10: Velikostní rozdíl modulů Arduino Uno a Arduino Nano



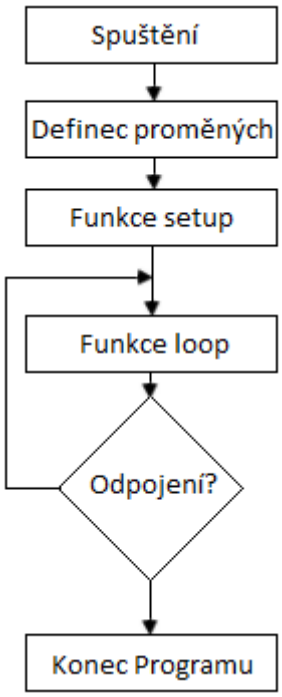
Zdroj: Confronto Arduino Uno Arduino Nano. *Vaglietti.it* [online]. [cit. 2018-02-27]. Dostupné z: http://www.vaglietti.it/confronto-arduino/

4.1.1 Postup řešení návrhu hardwarové části

Jako základ pro řešení hardwarového řešení byla zvolena deska Arduino Nano. Nejjednodušší cestou pro práci s deskami z rodiny Arduino, je volba vývojového prostředí, přímo od společnosti Arduino s názvem Arduino IDE. Arduino IDE je volně ke stažení na oficiálních stránkách Arduino IDE a pro jeho použití není potřeba žádných doplňků, tudíž postačí aplikaci stáhnout a nainstalovat. Pro komunikaci mezi deskou a vývojovým prostředím je nutné desku připojit pomocí kabelu USB-Mini k PC. Není nutné stahovat případné ovladače, počítačové zařízení desku samo rozpozná.

Pokud je poprvé spuštěn projekt v Arduino IDE, vývojové prostředí automaticky vygeneruje dvě základní funkce void *setup(),*pro základní inicializaci a funkci void *loop(),* která běží ve smičce od spuštění až do doby, než je zařízení odpojeno od napájení.

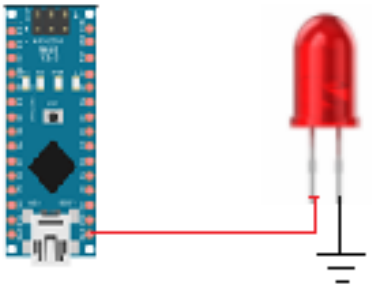
Obrázek 11: Průběh procesu softwaru Arduino IDE



Zdroj: Vlastní zpracování

Prvním krokem v rámci seznámením se s vlastnostmi desky bylo zapojení jednoho koncového zařízení k desce, které z desky bude zároveň napájeno, a které se bude automaticky přepínat na základě programového zpracování. Pro tento účel postačila prostá LED dioda připojená k jednomu z digitálních pinů. Připojení LED diody je zaznamenáno na následujícím schématu.

Obrázek 12: Schéma připojení LED diody k digitálnímu pinu



Zdroj: Vlastní zpracování

Pro zvolenou funkcionalitu je nutné nadefinovat vlastnosti pinu, ke kterému je koncové zařízení připojeno. Pro tyto účely bylo ve funkci *setup()* využito volání funkce *pinMode()*, která má dva vstupní parametry, první označuje číslo pinu a druhý zde se jedná o OUTPUT či INPUT. Pokud je pin definován, muže s ním být ve funkci *loop()* dále pracováno pomocí volání funkce *digitalWrite()*, která stejně jako funce *pinMode()* má dva vstupní parametry, ale liší se ve druhém parametru, který pomocí hodnot LOW a HIGH určuje, zda na daný pin bude puštěno napětí. Automatické přepíná zajistíme zavoláním funkce *delay()*, která pozastaví smyčku funkce *loop()* na dobu určenou hodnotou v jejím parametru. Celkové programové zpracování je zaznamenáno v příloze pod textem.

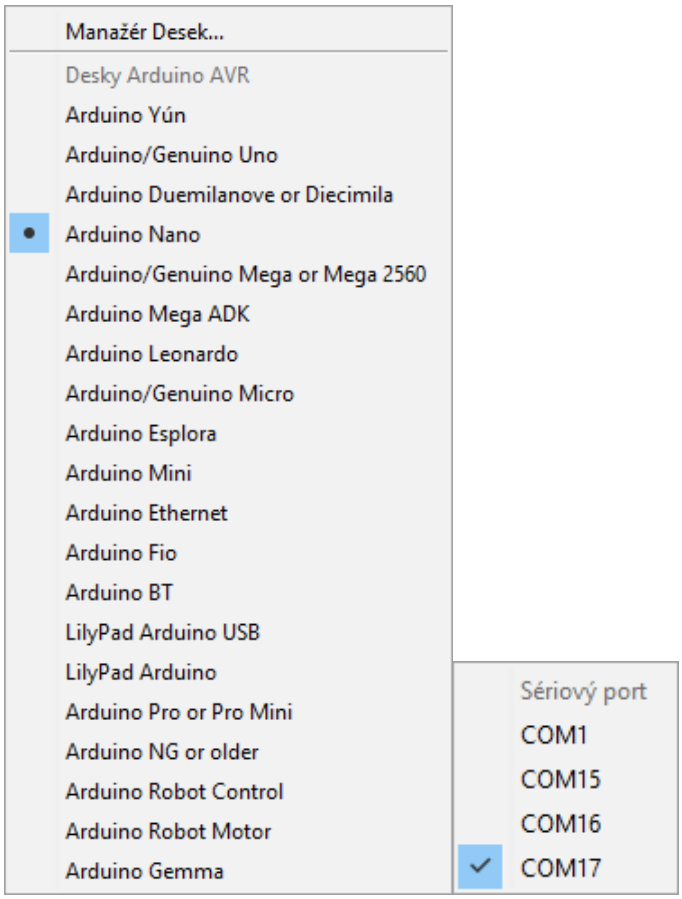
Obrázek 13: Programové zpracování jednoho automatického přepínače

```
boolean svit;
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  if (svit ? svit = false : svit = true);
  if(svit)
    digitalWrite(13, HIGH);
    else
    digitalWrite(13, LOW);
  delay(100);
}
```

Zdroj: Vlastní zpracování

Před nahráním kódu do desky je nutno projít základní nastavení ve vývojovém prostřední Arduino IDE, kde je v záložce „Nástroje“ je nutné vybrat typ desky, na kterou bude kód nahrán a přes jaký sériový port bude kód přenesen. Port se vybírá z prostého důvodu, k počítačovému zařízení může být ve stejný čas připojeno několik stejných zařízení a program by nevěděl, na které z nich má být kód nahrán. Následně už stačí jen nahrát kód a deska hned po zpracování dat začne přepínat nastavený pin.

Obrázek 14: Nastavení softwaru Arduino IDE

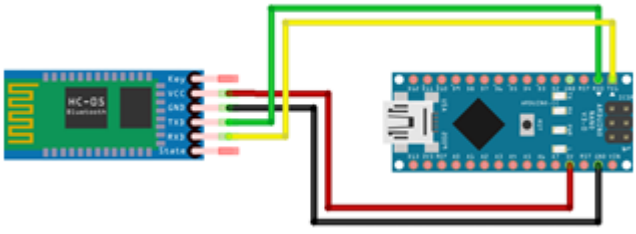


Zdroj: Vlastní zpracování

4.1.2 Přijímání příkazů pomocí Bluetooth modulu

Ve fázi, kdy je zprovozněn přepínač koncového zařízení, je čas pro připojení Bluetooth modulu HC-05, který má ověřenou kompatibilitu s deskou Arduino Nano. Pro připojení k desce jsou přímo určeny piny RX0 a TX1, jejich využití ovšem není podmíněné. Bluetooth modul lze připojit na jakékoli jiné digitální piny. Připojení Bluetooth modulu je znázorněno na následujícím schématu.

Obrázek 15: Schéma připojení Bluetooth modulu



Zdroj: Vlastní zpracování

Programová část musí být doplněna o nastavení Bluetooth, které proběhne za využití třídy Serial a voláním její funkce *begin()*, jejímž parametrem je hodnota představující počet badů, které představující frekvenci s jakou budou data přijímána. Standardně se využívá hodnota 9600, je možné využít i vyšší, ale pro využití v tomto případě bohatě postačí standardní hodnota. Po nastavení Bluetooth, deska začne takzvaně naslouchat. Pro kontrolu, zda nejsou k dispozici nějaká data k přijmutí, nabízí třída Serial další funkci *available()*, která vrací numerickou hodnotu. Pokud je tato hodnota nenulová, můžeme data načíst pomocí funkce *read()*. Načtená data mohou být opět vypsána pomocí funkce *write()*. Vzhledem k tomu, že data není kam vypsat, musí být deska připojena k počítačovému zařízení, následně lze využít nástroj sériový monitor, vývojové prostředí Arduino IDE, který plní funkci tzv. logu. Následující kód zachycuje rozšíření o práci s Bluetooth.

Obrázek 16: Programové zpracování přepínače na základě dat přijatých přes Bluetooth

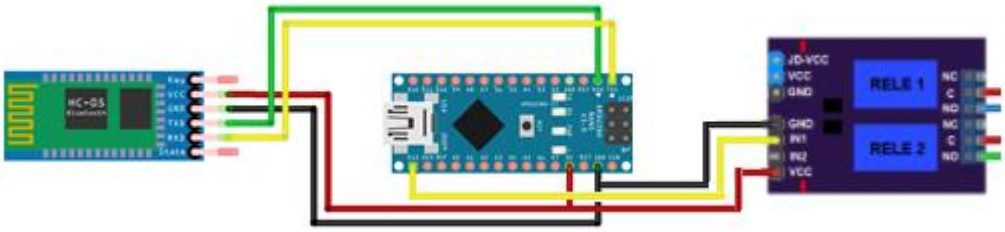
```
char data = 1;
boolean svit;
void setup()
{
  Serial.begin(9600);
    pinMode(13, OUTPUT);
}
void loop()
{
  if (Serial.available() > 0)
  {
    data = Serial.read();
    Serial.print(data+"\n");
    if (data=='d') {
        if (svit ? svit = false : svit = true);
        if (svit)
          digitalWrite(13, HIGH);
          else
          digitalWrite(13, LOW);
    }
  }
}
```

Zdroj: Vlastní zpracování

4.1.3 Finální hardwarové řešení

Vytvoření univerzální řídící jednotky, která dá uživateli možnost přepnutí napětí pro většinu běžných elektronických zařízení, lze nahradit LED diodu přepínacím relé, jež bude přepínat vnější elektrický obvod, na němž je připojeno koncové elektrické zařízení.

Obrázek 17: Schéma obvodu doplněné o relé



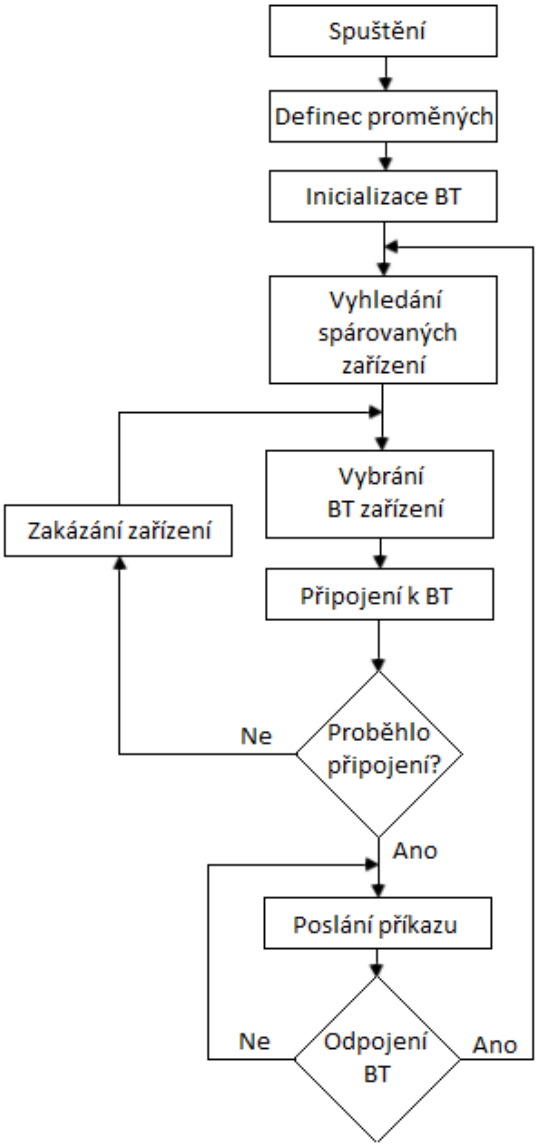
Zdroj: Vlastní zpracování

Celkový počet dvanácti volných digitálních pinů umožnilo rozšíření na 12 jednotlivě přepnutelných elektrických obvodů. Finální programové řešení pro ovládání všech 12 digitálních pinů, se nachází v Příloha XII.

4.2 Návrh mobilní aplikace

Jedním z hlavních cílů bylo vytvoření uživatelsky jednoduché a intuitivní mobilní aplikace, která bude plnit účel univerzálního dálkového přepínače. Od aplikace se očekává, že bude spravovat přístup k Bluetooth zařízení v mobilním telefonu a bude schopna jej ovládat. Hlavní funkcionalitou bude připojení ke vzdálenému Bluetooth zařízení a po navázání spojení umožní uživateli odesílání dat pro přepínání koncové zařízení připojeného k mikropočítači.

Obrázek 18: Schématické znázornění průběhu mobilní aplikace



Zdroj: Vlastní zpracování

Postupné kroky procesu aplikace:

1. Uživatel spustí aplikaci
2. Aplikace provede inicializaci proměnných
3. Aplikace zkontroluje, zda je spuštěný Bluetooth, pokud není, vyžádá si povolení pro spuštění
4. Aplikace provede inicializaci Bluetooth
5. Načtení spárovaných zařízení uložených v nastavení Bluetooth
6. Aplikace čeká, až uživatel vybere Bluetooth zařízení

7. Uživatel vybere jedno ze zařízení
8. Aplikace se pokusí k vybranému zařízení připojit, pokud proběhne připojení, načte se obrazovka s výběrem jednotlivých zařízení připojených k mikropočítači. Pokud připojení proběhne neúspěšně, zakáže vybrané zařízení a vrátí se zpět do kroku 6
9. Uživatel vybere jedno z nabízených zařízení
10. Aplikace odešle signál pro přepnutí vybraného zařízení
11. Uživatel klikne na tlačítko odpojit
12. Aplikace se odpojí od Bluetooth a vrátí se do kroku 5

Obrázek 19: Design mobilní aplikace

Zdroj: Vlastní zpracování

4.2.1 Vytvoření mobilní aplikace

Vzhledem k realizaci datového spojení formou Bluetooth zařízení bylo nutno vyloučit realizaci vlastní aplikace v prostření iOS. Firma Apple sice stále ve svých telefonech podporuje Bluetooth modul, ale pouze pro účely handsfree a podobných zařízení. Pokud uživatel chce využít zařízení pro vzdálený přístup k mikropočítači, narazí

na neřešitelný problém. Jako ideální prostředí tedy bylo zvoleno prostředí operačního systému Andriod, které propojení mobilního zařízení se vzdáleným Bloutooth modulem připojeného k mikropočítači umožňuje.

Pro napsání vlastní aplikace bylo využito vývojové prostředí Android Studio od společnosti Google, vyvinuto přímo za účelem práce s o operačním systémem Andriod. Pro využití vývojového prostředí Android studio není za potřebí žádné registrace, instalační soubor je volně ke stažení přímo na oficiálních stránkách Android Studia. V návaznosti na zvolení vývojového prostředí Android Studio byl kód aplikace psán v jazyce Java.

Android studio poskytuje několik užitečných funkcí a doplňků, které vývojáři značně ulehčí práci a šetří čas. Z těchto funkcí a doplňků je nutno vytknout alespoň tyto dvě, které jsou nad rámec běžného vývojového prostředí a byly aktivně využívány při tvorbě vlastní mobilní aplikace. První z nich je možnost emulovat vlastní mobilní zařízení přímo ve vývojovém prostřední a tím ušetří vývojáři čas strávený tvořením debug souboru, přesunutím jej do telefonu a následnou instalací. Toto je užitečná funkce, bohužel ve fázi vývoje. V momentě, kdy se vývojář pokusí v emulovaném zařízení otestovat Bluetooth zařízení, je ve slepé uličce a není jiné možnosti než připojit reálné mobilní zařízení k počítačovému zařízení. Ve chvíli, kdy je využíváno reálné mobilní zařízení, nastal čas pro využití další funkcionality Android Studia a tou je Android Device Monitor, který umožnuje sledovat veškeré logy z mobilního zařízení v reálném čase. Což značně usnadňuje debuggování aplikace. Pokud je nutné v logu zpětně dohledat záznam, je dovoleno jednoduché filtrování například podle aplikace, či konkrétní zprávy uložené v logu. Například pro nalezení logu se zápisem *Log.d* (TAG, "*Připojení Bluetooth*"); stačí do filtru zadat text "*Připojení Bluetooth*".

Při založení projektu aplikace sama vygeneruje kompletní adresářovou strukturu, se kterou bylo následně pracováno. Za hlavní adresáře je možné považovat manifest, java a res.

- Manifest – obsahuje soubor AndroidManifest.xml
- Java – obsahuje zdrojové kódy

- Res – obsahuje několik dalších podadresářů
    I. Res/layout – uživatelské rozhraní
    II. Res/drawable – obrázky
    III. Res/mipmap – tvorba menu
    IV. Res/values – proměnné uložené v následujících souborech
        i. string.xml
        ii. style.xml
        iii. color.xml

4.2.2 Vlastní kód aplikace

Ve výše uvedených adresářích se nachází několik základních souborů, které se musí být obsaženy v každé android aplikaci a bez kterých aplikaci nelze ani zkompilovat. Pro vytvoření této mobilní aplikace postačilo držet se aktivně těchto tří základních souborů, ve kterých se nachází vlastní kód aplikace.

4.2.2.1 AndroidManifest.xml

AndroidManifest.xml nesmí chybět v žádné Android aplikaci. Přestože aplikace jako taková je psaná v jazyce Java, kód v souboru AndroidManifest.xml je psán formou značkovacího jazyka XML (anglicky *Extensible Markup Language*), který je v dnešní době nepoužívanějším metajazykem.

Soubor obsahuje všechny základní vlastnosti Android aplikace. Nalezneme v něm deklarace vzájemně propojených komponent, kterými jsou například aktivity, služby přijímače a deklaraci uživatelských oprávnění. Pro vytvoření této konkrétní aplikace postačí velice jednoduchý obsah souboru manifest, který nám vygenerovalo samo Android Studio a obsahuje definici samotné aplikace, jež obsahuje pouze jednu aktivitu reprezentující použití obrazovky aplikace.

Pro následnou realizaci aplikace je nutné si uvědomit, že aplikace musí využívat Bluetooth adaptér. Zde v souboru AndroidManifest.xml je nutno k němu nastavit jistá oprávnění, a proto jsou vloženy do následující části kódu řádky 4 a 5, bez kterých by nebylo ani možné využití knihoven pro Bluetooth adaptér.

Obrázek 20: Obsah souboru AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ouednk.znova">
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="znova"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Zdroj: Vlastní zpracování

4.2.2.2 MainActivity

Obsahem tohoto souboru je kód stejnojmenné třídy MainActivity, která představuje gró samotné aplikace. Stejně jako v případě souboru AndroidManifest.xml, i v tomto souboru nám Android Studio vygeneruje základní strukturu kódu, jež je uvedena pod textem. První částí je *packagecom.example.ouednk.bakalskprceouednk*;. Jednoznačně definuje provázání s ostatními soubory v celé adresářové struktuře. Další částí jsou základní importy knihoven import *android.support.v7.app.AppCompatActivity*; import *android.os.Bundle*;, nutné pro základní chod aplikace. Hlavní částí je třída MainActivity, jež je rozšířena o vlastnosti třídy AppCompatActivity. Třída MainActivity obsahuje tzv. Override metodu *onCreate()*. Tato metoda proběhne vždy se spuštěním programu a měla by inicializovat pouze nejnutnější proměnné nutné k následnému chodu programu. V této metodě se nachází volání metody odvozené třídy AppCompatActivity *super.onCreate()*. Poslední před vytvořenou částí je *setContentView(R.layout.activity_main)* za účelem provázání se souborem *activity_main.xml* obsahujícím vlastnosti související s designem aplikace. Kompletní kód se nachází v Příloha X.

Obrázek 21: Obsah souboru MainActivity pro vytvoření aplikace

```
package com.example.ouednk.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState)
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Zdroj: Vlastní zpracování

Hlavní funkcionalitou aplikace je komunikace přes Bluetooth adaptér, pro jeho deklaraci je nejprve nutné naimportovat knihovny BluetoothAdapter, BluetoothDevice a BluetoothSocket. Po deklaraci proměnných, je možné zavolat následující metodu *BTInicializace()* s návratovým typem boolean, ve které proběhne inicializace Bluetooth adaptéru, následná kontrola, zda je Bluetooth v mobilním zařízení zapnut. Dále metoda projde spárovaná zařízení, a pokud najde shodu Bluetooth adres, vrátí kladnou návratovou hodnotu.

Obrázek 22: Programové zpracování metody BTInicializace

```
public boolean BTInicializace() {
    BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if(!zkontrolujZaplejBT(bluetoothAdapter))
        return false;
    Set<BluetoothDevice> bondedDevices = bluetoothAdapter.getBondedDevices();
    if (bondedDevices.isEmpty()) {
        MujLog.setText("Žádná spárovaná zařízení - Nejdříve spárujte zařízení");
    } else {
        MujLog.setText("Spárovaná zařízení");
        for (BluetoothDevice iterator : bondedDevices) {
            if (iterator.getAddress().equals(adresaBTVybrana)) {
                device = iterator;
                MujLog.setText("Nanalezeno požadované zařízení");
                return true;
            } else {
                MujLog.setText("Nenalezeno požadované zařízení");
            }
        }
    }
    return false;
}
```

Zdroj: Vlastní zpracování

Ve fázi, kdy proběhla inicializace Bloutooth zařízení, je možné zavolat metodu *BTPriopojeni()*, která pomocí třídy BluetoothSocket naváže spojení. Nejprve je však nutné socketu nastavit UUID, což je zkratka pro jednorázovou identifikaci objektu v síti. UUID se musí na obou dvou stranách spojení shodovat a v případě připojení k Arduino Bluetooth modulu je použito konkrétně toto UUID: 00001101-0000-1000-8000-00805f9b34fb. Obsah metody je obalen v *try* a *catch*, poněvadž to vyžaduje Java kompilátor v případě použití Input / Output operací. Pro použití Input / Output operací je nutné importovat knihovny OutputStream.

Obrázek 23: Programové zpracování metody BTPripojeni

```
public boolean BTPriopojeni() {
    try {
        socket = device.createRfcommSocketToServiceRecord(PORT_UUID);
        socket.connect();
        pripojeno = true;
        MujLog.setText("Proběhlo připojení k zařízení");
    } catch (IOException e) {
        e.printStackTrace();
        pripojeno = false;
        MujLog.setText("Neproběhlo připojení k zařízení");
    }
    if (pripojeno) {
        try {
            outputStream = socket.getOutputStream();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return pripojeno;
}
```

Zdroj: Vlastní zpracování

Jestliže se aplikaci podařilo úspěšně připojit ke vzdálenému Bluetooth zařízení, nic už nebrání v komunikaci. Následující metoda *PrepniZarizeni()* popisuje samotné odesílání příkazů do připojeného mikropočítače. Jako v předchozím případě musí být využito obalení kódu do *try* a *catch*. Konečná komunikace probíhá odesíláním jednotlivých bajtů pomocí třídy OutputStream a její předepsané metody *write()*.

Obrázek 24: Programové zpracování metody PrepniZarizeni

```
public boolean PrepniZarizeni() {
    if (pripojeno) {
        try {
            outputStream.write(vybranyPrikaz.getBytes());
            MujLog.setText("Příkaz odeslán");
            return true;
        } catch (IOException ex) {
            MujLog.setText("Příkaz se nepodařilo odeslat");
        }
    }
    return false;
}
```

Zdroj: Vlastní zpracování

Pokud se uživatel rozhodne ukončit spojení se vzdáleným zařízením, tak vzhledem k tomu, že bylo použito input / output operací, tak je očekáváno, že aplikace po sobě tzv. uklidí. Od toho je níže uvedená metoda *Odpoj()*, jejíž hlavním účelem je zavolání metody *close()* třídy OutputStream, která ukončí navázané spojení.

Obrázek 25: Programové zpracování metody Odpoj

```
public boolean Odpoj() {
    try {
        outputStream.close();
        socket.close();
        pripojeno = false;
        MujLog.setText("Aplikace se odpojila od zařízení");
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

Zdroj: Vlastní zpracování

4.2.2.3 Activity_Main.xml

Soubor Activity_Main.xml uchovává kompletní design, který byl v aplikaci nastaven. Android Studio umožnuje dvě možnosti zobrazení, první textové, které je zobrazeno zde pod textem, nebo formou designu. Obě dvě zobrazení jsou vzájemně provázané a umožnují vývojáři kombinovat formy psaní designu. Tato práce byla psána tzv. klikající formou v designu, kde bylo vše potřebné sestaveno ve vlastnostech jednotlivých objektů a Android Studio vše převede na kód. Kompletní kód se nachází v Příloha XI.

I tento soubor obsahuje kód zapsaný v jazyce XML. Z ukázky kódu pod textem je asi nejdůležitější si povšimnout, že ID daného objektu je zaznamenáno formou textového řetězce a k tomu Buttonu se v MainActivity přistupuje např. pomocí tohoto příkazu *b1 = (Button) findViewById(R.id.b1)*. Dále jsou zaznamenány vlastnosti jako text, viditelnost, případně kotvení a mnoho dalších.

Obrázek 26: Programové zpracování zápisu tlačítka v jazyce xml

```
<Button
    android:id="@+id/b1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.201" />
```

Zdroj: Vlastní zpracování

4.3 Využití v praxi

Inspirací pro zhotovení této bakalářské práce bylo kompletní ovládání přídavných ventilátorů osazených v počítačové skříni a osvětlení pomocí LED pásků. Počítačová skříň byla za tímto účelem doplněna o manuálním regulátor otáček od značky AKASA, který umožnuje regulaci otáček v rozmezí 0–100 %. Regulátor byl využíván pouze pro dvě polohy, 100 % pokud bylo počítačové zařízení v plné zátěži, nebo 0 % pro úplné vypnutí. Vzhledem k opakujícím se situacím, ve kterých bylo zapomenuto na vypnutí ventilátorů a LED osvětlení ve chvíli, kdy bylo počítačové zařízení mimo zátěž. Uživatel již vykonával jinou činnost, vzdálen od počítačové skříně, při které bylo nežádoucí, aby byl rušen hlukem počítačové skříně, nebo byl oslňován světlem vycházejícího z počítače. Za tímto účelem bylo využito zařízení zhotoveného při tvorbě bakalářské práce. Implementací zařízení do počítačové skříně uživateli vznikla možnost na dálku eliminovat výše zmíněné rušivé faktory pomocí jakéhokoliv mobilního zařízení obsahujícího Bluetooth.

Obrázek 27: Akasa AK-FC

Zdroj: AKASA AK-FC. *Vaglietti.it* [online]. [cit. 2018-02-26]. Dostupné z: http://www.vaglietti.it/confronto-arduino/

Možností využití se nabízí nespočet. Před volbou, zda dané zařízení využít pro přepínání vybraného zařízení, je nutné nejdříve zvážit, zda je žádoucí, aby toto zařízení bylo spouštěno a vypínáno pouze mobilní aplikací. Použitím samotného zhotoveného zařízení se uživatel připraví o možnost manuálního přepínání a je nucen pro každé přepnutí využít mobilní aplikaci. V případě, že by bylo žádoucí zachovat klasické přepínání například pomocí vypínače a k tomu mít možnost zařízení přepínat i přes vytvořenou aplikaci, je možné využití modifikováním schématu tzv. schodišťového vypínače s tím, že jeden z vypínačů by byl nahrazen vytvořeným zařízení.

5 Závěr

Cílem této bakalářské práce bylo navržení a sestavení funkčního prototypu obvodu s mikropočítačem, za účelem přepínání napětí pro koncové elektrické zařízení, který zvládne přijímat příkazy od vzdáleného mobilního zařízení pomocí přenosu dat přes Bluetooth. Bakalářská práce se skládá ze dvou hlavních částí, část teoretická a část analytická.

Teoretická část se zabývá problematikou, která je úzce spjatá se zhotovením praktické částí. Obsahuje charakteristiky dílčích částí vymezujících jejich využití. Selekcí byl na základě těchto charakteristik vybrán vhodný mikropočítač, který plní funkci řídící jednotky. Dále Bluetooth modul, pomocí kterého mikropočítač přijímá instrukce, a v neposlední řadě se teoretická část zabývá vývojovými prostředími a použitými programovacími jazyky.

Praktická část obsahuje kompletní postup při tvorbě funkčního prototypu. V první řadě bylo nutné z vybraných dílčích komponentů sestavit na základě zhotoveného schématu funkční hardwarový obvod. Na základě toho obvodu byl v jazyce C napsán jednoduchý kód, který byl následně nahrán do mikropočítače pro jeho řízení. Následně byla navržena a napsána mobilní aplikace v jazyce Java, která byla napsána za účelem vzdáleného ovládání mikropočítače.

Výsledkem praktické části je sestavení funkčního obvodu, který zvládne pomocí mobilní aplikace přepínat napětí až pro 12 koncových zařízení. Na konci praktické části jsou stručně popsány možnosti finálního využití.

6 Seznam použitých zdrojů

ADÁMKOVÁ, Věra. *Hodnocení vybraných metod v kardiologii a angiologii pro praxi* [online]. Graga Publishing, 2016. 150 s. [cit. 2018-02-21]. Dostupné z: https://books.google.cz/books?id=hDgCDQAAQBAJ

AKASA AK-FC. *Vaglietti.it* [online]. [cit. 2018-02-26]. Dostupné z: http://www.vaglietti.it/confronto-arduino/

Android Studio. *Infinum.co* [online]. [cit. 2018-02-27]. Dostupné z: https://infinum.co/the-capsized-eight/android-studio-vs-eclipse-1-0

Aplikace Hello World v jazyce C++ s využitím sady Visual Studio 2017. *Visualstudio.com* [online]. [cit. 2018-02-27]. Dostupné z: https://www.visualstudio.com/cs/vs/support/hello-world-c-using-visual-studio-2017/?rr=https%3A%2F%2Fwww.google.cz%2F

Arduino – pin mapping. *Arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://www.arduino.cc/en/Hacking/PinMapping168

Arduino 433MHz vysílač + přijímač. *Arduino-shop.cz* [online]. [cit. 2018-02-24]. Dostupné z: https://arduino-shop.cz/arduino/1003-arduino-433mhz-vysilac-prijimac-1427821401.html

Arduino IDE. *Arduino.cz* [online]. [cit. 2018-02-24]. Dostupné z: https://arduino.cz/arduino-ide/

Arduino Nano. *Arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://www.arduino.cc/en/uploads/Main/Arduino_Nano-Rev3.2-SCH.pdf

Arduino Nano. *Store.arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://store.arduino.cc/arduino-nano

Arduino Uno REV3. *Store.arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

CAFOUREK, Bohdan. *Windows 7: kompletní příručka*. Praha: Grada, 2010. Profesionál. 326 s. ISBN 978-80-247-3209-1.

Confronto Arduino Uno Arduino Nano. *Vaglietti.it* [online]. [cit. 2018-02-27]. Dostupné z: http://www.vaglietti.it/confronto-arduino/

DESAI, Pratik. *Python Programming for Arduino* [online]. Packt Publishing, 2015. 400 s. [cit. 2018-02-21]. Dostupné z: https://books.google.cz/books?id=O0PfBgAAQBAJ

Developing a Java Hello World Application. *Jetbrains.com* [online]. [cit. 2018-02-27]. Dostupné z: https://www.jetbrains.com/help/idea/developing-a-javafx-hello-world-application-coding-examples.html

HART-DAVIS, Guy. *Deploying Raspberry Pi in the classroom* [online]. California: Apress/Springer Science+Business Media Finance, 2017. 293 s. [cit. 2018-02-23]. Dostupné z: https://books.google.cz/books?id=-fjFDQAAQBAJ

CHOUDRUHI, Kallol. *Learn Arduino Prototyping in 10 days* [online]. Packt Publishing, 2017. 288 s. [cit. 2018-02-21]. Dostupné z: https://books.google.cz/books?id=3nc5DwAAQBAJ

JENÍČEK, Vladimír. *Globalizace světového hospodářství*. Praha: C. H. Beck, 2002. 152 s. ISBN 978-80-7179-787-1.

KARVINEN, Tero a Kimmo KARVINEN. *Make Six Embedded Projects with Open Source Hardware and Software.* Sebastopol: O'Reilly Media, 2011. 296 s. ISBN 9781449307233.

KERNIGHAN, W. Brian a Dennis M. RITCHIE. *Programovací jazyk C* [online]. Praha: Albatros, 2017. 288 s. [cit. 2018-02-26]. Dostupné z: https://books.google.cz/books?id=WhDqCwAAQBAJ

MCU – mikroelektronika. *Mcu.cz* [online]. [cit. 2018-02-24]. Dostupné z: http://mcu.cz/comment-n3958.html

Meet Android Studio. *Developer.android.com* [online]. [cit. 2018-02-27]. Dostupné z: https://developer.android.com/studio/intro/index.html

Mikrokontroler AVR ATmega328P. *Abc-rc.pl* [online]. [cit. 2018-02-24]. Dostupné z: https://abc-rc.pl/mikrokontroler-avr-atmega328p-pu-mikroprocesor-arduino

MLÝNKOVÁ, Irena. *XML technologie: Principy a aplikace v praxi.* Praha: Grada Publishing, 2008. 272 s. ISBN 978-80-247-6688-1.

PAVEL, Burian. *Internet inteligentních aktivit* [online]. Praha: Grada, 2014. 336 s. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=ATqCBAAAQBAJ

PEREA, Francis. *Arduino Essentials* [online]. Packt Publishing, 2015. 206 s. [cit. 2018-02-19]. Dostupné z: https://books.google.cz/books?id=NrjNBgAAQBAJ

POGUE, David. *Mac OS X Snow Leopard* [online]. Albatros Media, 2016. s. 952. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=SwvqCwAAQBAJ

PRABHU, C.S.R. a A. Prathap REDDI. *Bluetooth technology and its applications with JAVA and J2ME* [online]. Eastern economy ed. New Delhi: Prentice-Hall of India, 2006. 340 s. [cit. 2018-02-18]. Dostupné z: http://bit.ly/2oIJqar

PROKOP, Jiří. *Algoritmy v jazyku C a C++*. 2., rozš. a aktualiz. vyd. Praha: Grada, 2012. Průvodce (Grada). 169 s. ISBN 978-80-247-3929-8.

RAAB, Stefan a Madhavi W. CHANDRA. *Cisco: mobilní IP technologie a aplikace*. Praha: Grada, 2007. 299 s. ISBN 80-247-1611-9.

Raspberry Pi 3 Model B. *Element14.com* [online]. [cit. 2018-02-27]. Dostupné z: https://www.element14.com/community/docs/DOC-81294/l/raspberry-pi-3-model-b-with-1gb-of-ram-with-wifi-and-bluetooth-low-energy

ROUBAL, Pavel. *Informatika a výpočetní technika pro střední školy: Teoretická učebnice* [online]. Praha: Albatros, 2017. 104 s. [cit 2018-02-19]. Dostupné z: https://books.google.cz/books?id=qw0nCwAAQBAJ

SCOTT, Michael Lee. *Programming language pragmatics* [online]. Waltham, MA: Morgan Kaufmann, an imprint of Elsevier, 2016. 992 s. [cit. 2018-02-26]. Dostupné z: https://books.google.cz/books?id=jM-cBAAAQBAJ

SELECKÝ, Matúš. *Arduino* [online]. Praha: Albatros, 2016. 344 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=fyq7DQAAQBAJ

SCHILDT, Herbert. *Java 7* [online]. Praha: Albatros, 2016. 664 s. [cit. 2018-02-26]. Dostupné z: https://books.google.cz/books?id=SArqCwAAQBAJ

SOSINSKY, Barrie. *Mistrovství – počítačové sítě* [online]. Albatros Media a.s., 2016. 840 s. [cit. 2018-02-18]. Dostupné z: https://books.google.cz/books?id=qwbqCwAAQBAJ

TUTTLEBEE, Walter. *Software Defined Radio Enabling Technologies*. Chichester: John Wiley, 2003. 440 s. ISBN 978-0-470-85263-7.

TZIVARAS, Vasilis. *Raspberry Pi Zero W Wireless Projects* [online]. Packt Publishing, 2017. 240 s. [cit. 2018-02-25]. Dostupné z: https://books.google.cz/books?id=fpZGDwAAQBAJ

UPTON, Eben a Gareth HALFACREE. *Raspberry Pi* [online]. Praha: Albatros, 2017. 232 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=eQ7qCwAAQBAJ

What is the range of Bluetooth. *Scienceabc.com* [online]. [cit. 2018-02-23]. Dostupné z: https://www.scienceabc.com/innovation/what-is-the-range-of-bluetooth-and-how-can-it-be-extended.html

What's the Difference Between IEEE 802.11af and 802.11ah. *Mwrf.com* [online]. [cit. 2018-02-23]. Dostupné z: http://www.mwrf.com/active-components/what-s-difference-between-ieee-80211af-and-80211ah

WILLIAMS, Tim. *EMC for Product Designers*. 4. vyd. Burlington: Elsevier, 2007. ISBN 978-0-08-046954-6.

ZAORAL, Ondřej a Josef TKÁČ. *Průvodce světem kapesních počítačů: aneb PDA na dlani* [online]. Grada Publishing, 2005. 208 s. [cit. 2018-02-20]. Dostupné z: https://books.google.cz/books?id=rbpaAgAAQBAJ

7 Seznam obrázků

8 Seznam příloh

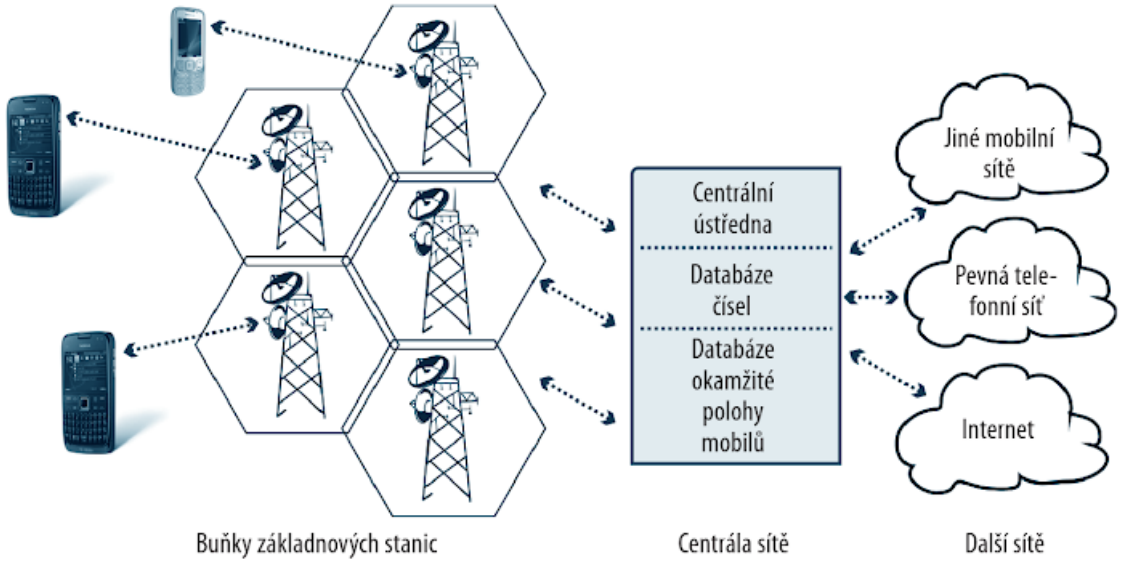
9 Přílohy

Příloha I: Rozdíl mezi standardy IEEE 802.11af a 802.11ah

Standard	Frekvence	Šířka pásma	Max. přenosová rychlost	Dosah
802.11	2,4 GHz	20 MHz	2 Mbps	20 m
b	2,4 GHz	21 MHz	11 Mbps	35 m
a	5 GHz	22 MHz	54 Mbps	35 m
g	2,4 GHz	23 MHz	54 Mbps	70 m
n	2,4 GHz, 5GHz	24 MHz a 40MHz	600 Mbps	70 m
ac	5 GHz	160 MHz	6,93 Gbps	35 m
ad	60 GHz	2,16 GHz	6,76 Gbps	10 m
af	54-790 MHz	6, 7, 8 MHz	26,7 Mbps	1 km
ah	900 MHz	1, 2, 4, 8, 16 MHz	40 Mbps	1 km

Zdroj: What's the Difference Between IEEE 802.11af and 802.11ah. *Mwrf.com* [online]. [cit. 2018-02-23]. Dostupné z: http://www.mwrf.com/active-components/what-s-difference-between-ieee-80211af-and-80211ah

Příloha II: Princip buňkové mobilní sítě



Zdroj: ROUBAL, Pavel. *Informatika a výpočetní technika pro střední školy: Teoretická učebnice* [online]. Praha: Albatros, 2017. 104 s. [cit 2018-02-19]. Dostupné z: https://books.google.cz/books?id=qw0nCwAAQBAJ

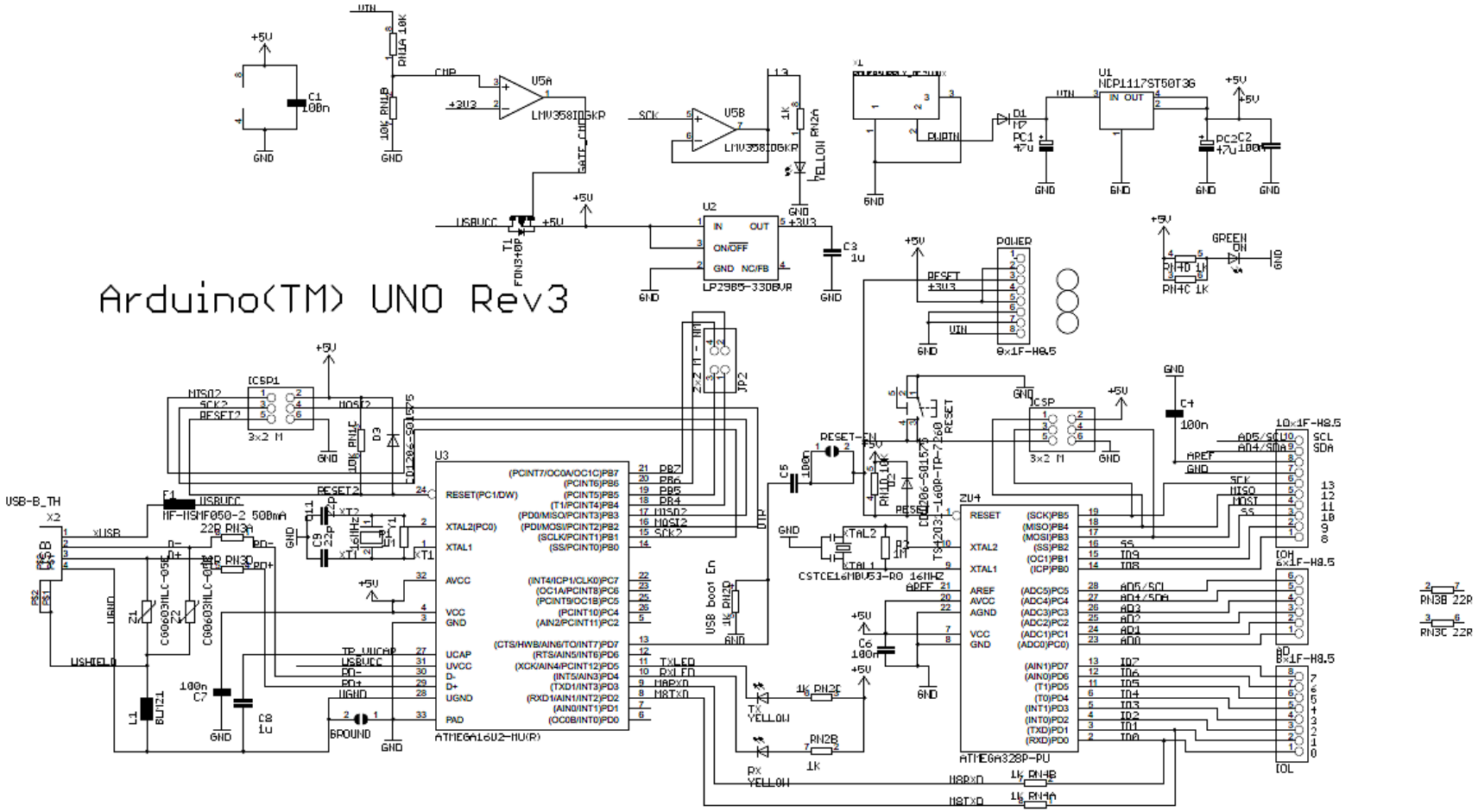
Příloha III: 433 MHz vysílač s přijímačem

Zdroj: Arduino 433MHz vysílač + přijímač. *Arduino-shop.cz* [online]. [cit. 2018-02-24]. Dostupné z: https://arduino-shop.cz/arduino/1003-arduino-433mhz-vysilac-prijimac-1427821401.html

Příloha IV: Mikrokontroler AVR ATmega328P

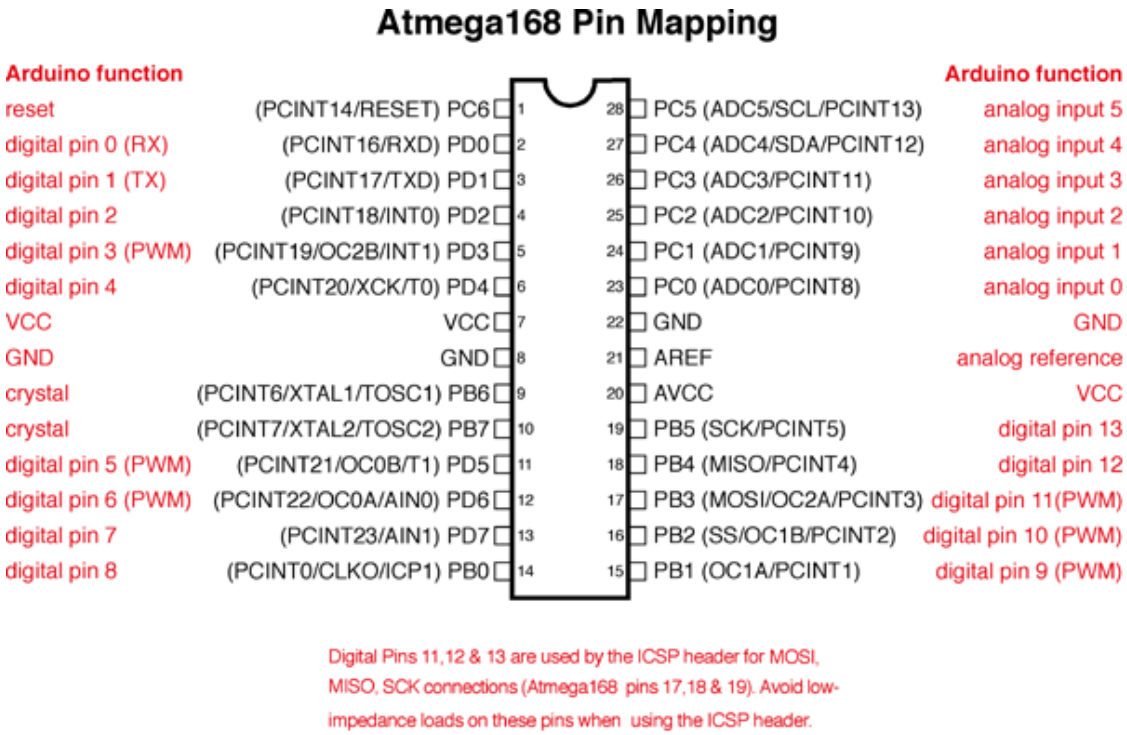
Zdroj: Mikrokontroler AVR ATmega328P. *Abc-rc.pl* [online]. [cit. 2018-02-24]. Dostupné z: https://abc-rc.pl/mikrokontroler-avr-atmega328p-pu-mikroprocesor-arduino

Příloha V: Schéma desky Arduina Uno



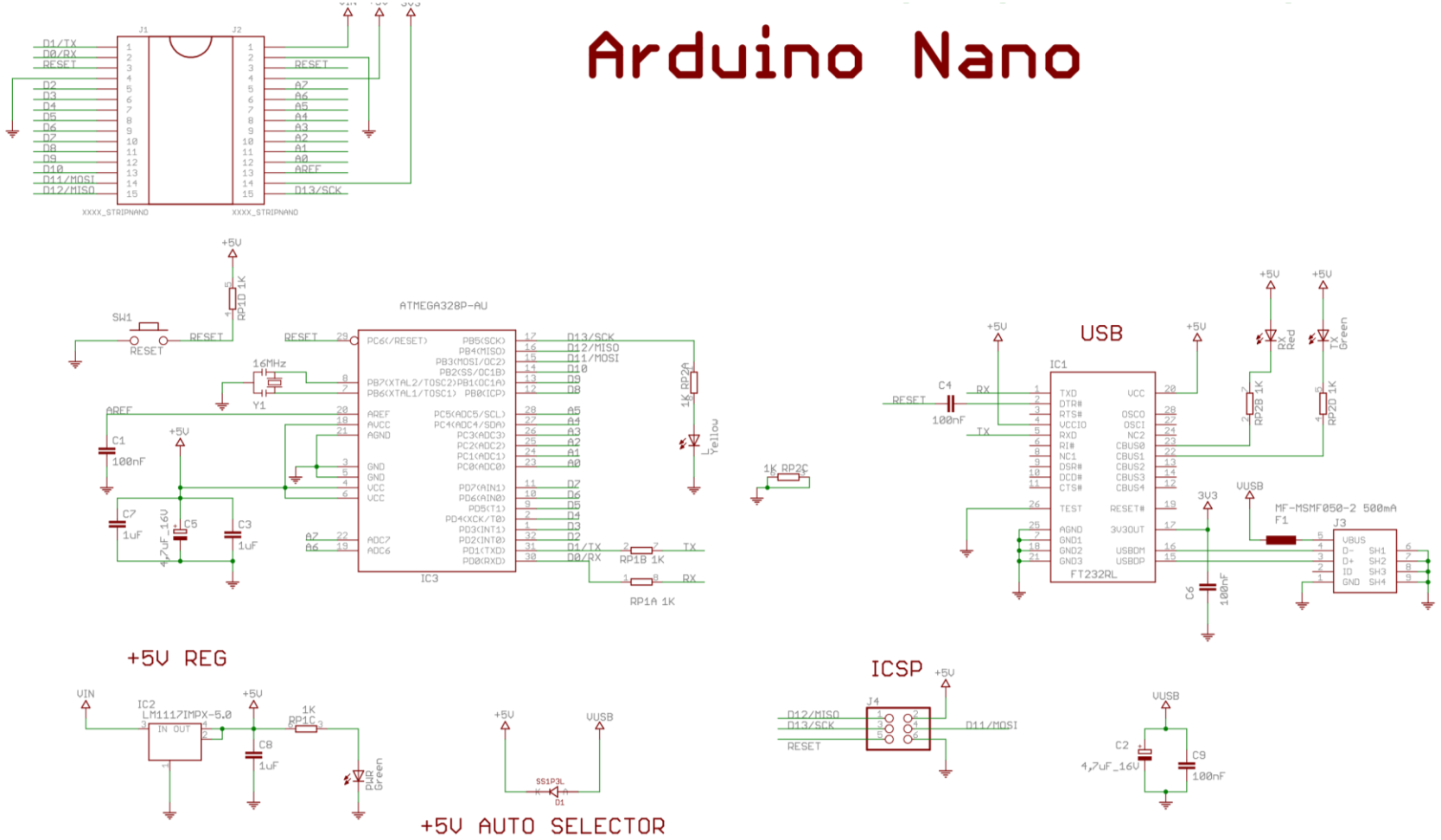
Zdroj: Build your own Arduino. *Electroschematics.com* [online]. [cit. 2018-02-24]. Dostupné z: http://www.electroschematics.com/10955/build-arduino-bootload-atmega-microcontroller-part-1/

Příloha VI: Mapování mezi piny Arduino a porty ATmega328P



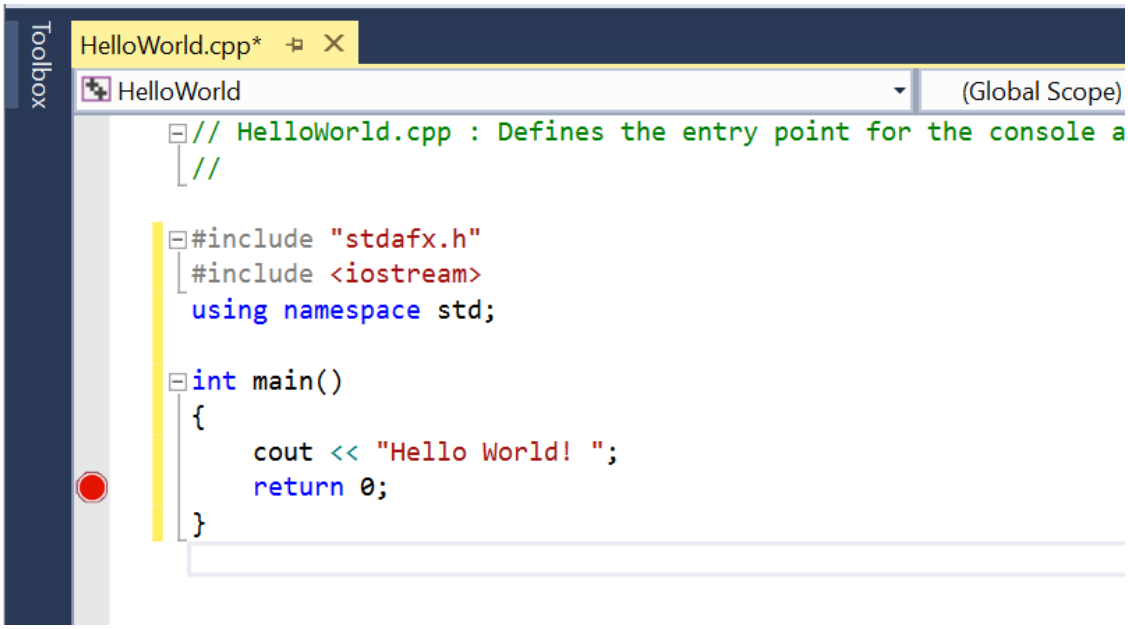
Zdroj: Arduino – pin mapping. *Arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://www.arduino.cc/en/Hacking/PinMapping168

Příloha VII: Schéma desky Arduino Nano



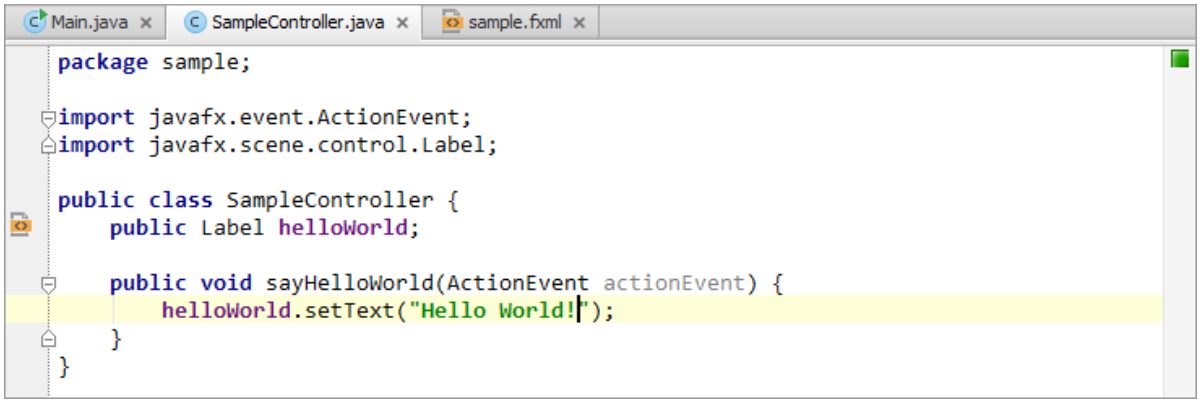
Zdroj: Arduino Nano. *Arduino.cc* [online]. [cit. 2018-02-24]. Dostupné z: https://www.arduino.cc/en/uploads/Main/Arduino_Nano-Rev3.2-SCH.pdf

Příloha VIII: Aplikace Hello World v jazyce C++



Zdroj: Aplikace Hello World v jazyce C++ s využitím sady Visual Studio 2017. *Visualstudio.com* [online]. [cit. 2018-02-27]. Dostupné z: https://www.visualstudio.com/cs/vs/support/hello-world-c-using-visual-studio-2017/?rr=https%3A%2F%2Fwww.google.cz%2F

Příloha IX: Aplikace Hello World v jazyce Java



Zdroj: Developing a Java Hello World Application. *Jetbrains.com* [online]. [cit. 2018-02-27]. Dostupné z: https://www.jetbrains.com/help/idea/developing-a-javafx-hello-world-application-coding-examples.html

Příloha X: Vlastní programové řešení v souboru MainActivity

```
package com.example.ouednk.bakalskprceouednk;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Set;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private Button buttonOdpoj, b1, b2, b3, b4, b5, b6,
nactiBT,bp2,bp3,bp4,bp5,bp6,bp7,bp8,bp9,bp10,bp11,bp12,bp13;
    private TextView MujLog;
    private static String adresaBT1, adresaBT2, adresaBT3, adresaBT4, adresaBT5, adresaBT6,
adresaBTVybrana;
    private final UUID PORT_UUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
    private BluetoothDevice device;
    private BluetoothSocket socket;
    private OutputStream outputStream;
    boolean pripojeno = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        buttonOdpoj = (Button) findViewById(R.id.buttonOdpoj);
        MujLog = (TextView) findViewById(R.id.text);
        b1 = (Button) findViewById(R.id.b1);
        b2 = (Button) findViewById(R.id.b2);
        b3 = (Button) findViewById(R.id.b3);
        b4 = (Button) findViewById(R.id.b4);
        b5 = (Button) findViewById(R.id.b5);
        b6 = (Button) findViewById(R.id.b6);
        bp2 = (Button) findViewById(R.id.bp2);
        bp3 = (Button) findViewById(R.id.bp3);
        bp4 = (Button) findViewById(R.id.bp4);
        bp5 = (Button) findViewById(R.id.bp5);
        bp6 = (Button) findViewById(R.id.bp6);
        bp7 = (Button) findViewById(R.id.bp7);
        bp8 = (Button) findViewById(R.id.bp8);
        bp9 = (Button) findViewById(R.id.bp9);
        bp10 = (Button) findViewById(R.id.bp10);
        bp11 = (Button) findViewById(R.id.bp11);
        bp12 = (Button) findViewById(R.id.bp12);
        bp13 = (Button) findViewById(R.id.bp13);
        nactiBT = (Button) findViewById(R.id.nactiBT);
        listenerOnButtons();
        NastavButtonyPodleBT();
    }
```

```
public void listenerOnButtons() {
    nactiBT.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            Odpoj();
            NastavButtonyPodleBT();
            nactiBT.setText("Znovu načti všechna zařízení");
        }
    });
    b1.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            adresaBTVybrana = adresaBT1;
            if (Pripoj()) {
                UiPoVybraniZarizeni();
            }
            else
                b1.setEnabled(false);
        }
    });
    b2.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            adresaBTVybrana = adresaBT2;
            if (Pripoj()) {
                UiPoVybraniZarizeni();
            }
            else
                b2.setEnabled(false);
        }
    });
    b3.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            adresaBTVybrana = adresaBT3;
            if (Pripoj()) {
                UiPoVybraniZarizeni();
            }
            else
                b3.setEnabled(false);
        }
    });
    b4.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            adresaBTVybrana = adresaBT4;
            if (Pripoj()) {
                UiPoVybraniZarizeni();
            }
            else
                b4.setEnabled(false);
        }
    });
    b5.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            adresaBTVybrana = adresaBT5;
            if (Pripoj()) {
                UiPoVybraniZarizeni();
```

```
        }
        else
            b5.setEnabled(false);
    }
});
b6.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        adresaBTVybrana = adresaBT6;
        if (Pripoj()) {
            UiPoVybraniZarizeni();
        }
        else
            b6.setEnabled(false);
    }
});
buttonOdpoj.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Odpoj();
        NastavButtonyPodleBT();
    }
});
bp2.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        PrepniZarizeni("2");
    }
});
bp3.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        PrepniZarizeni("3");
    }
});
bp4.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        PrepniZarizeni("4");
    }
});
bp5.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        PrepniZarizeni("5");
    }
});
bp6.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        PrepniZarizeni("6");
    }
});
bp7.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        PrepniZarizeni("7");
    }
});
bp8.setOnClickListener(new OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            PrepniZarizeni("8");
        }
    });
    bp9.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            PrepniZarizeni("9");
        }
    });
    bp10.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            PrepniZarizeni("a");
        }
    });
    bp11.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            PrepniZarizeni("b");
        }
    });
    bp12.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            PrepniZarizeni("c");
        }
    });
    bp13.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            PrepniZarizeni("d");
        }
    });
}

public void UiPoVybraniZarizeni() {
    b1.setVisibility(View.INVISIBLE);
    b2.setVisibility(View.INVISIBLE);
    b3.setVisibility(View.INVISIBLE);
    b4.setVisibility(View.INVISIBLE);
    b5.setVisibility(View.INVISIBLE);
    b6.setVisibility(View.INVISIBLE);
    buttonOdpoj.setVisibility(View.VISIBLE);
    bp2.setVisibility(View.VISIBLE);
    bp3.setVisibility(View.VISIBLE);
    bp4.setVisibility(View.VISIBLE);
    bp5.setVisibility(View.VISIBLE);
    bp6.setVisibility(View.VISIBLE);
    bp7.setVisibility(View.VISIBLE);
    bp8.setVisibility(View.VISIBLE);
    bp9.setVisibility(View.VISIBLE);
    bp10.setVisibility(View.VISIBLE);
    bp11.setVisibility(View.VISIBLE);
    bp12.setVisibility(View.VISIBLE);
    bp13.setVisibility(View.VISIBLE);
}

public boolean Pripoj() {
```

```
    if (BTInicializace()) {
        if (BTPriopojeni()) {
            return true;
        }
    }
    return false;
}

public boolean BTInicializace() {
    BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if(!zkontrolujZaplejBT(bluetoothAdapter))
        return false;
    Set<BluetoothDevice> bondedDevices = bluetoothAdapter.getBondedDevices();
    if (bondedDevices.isEmpty()) {
        MujLog.setText("Žádná spárovaná zařízení - Nejdříve spárujte zařízení");
    } else {
        MujLog.setText("Spárovaná zařízení");
        for (BluetoothDevice iterator : bondedDevices) {
            if (iterator.getAddress().equals(adresaBTVybrana)) {
                device = iterator;
                MujLog.setText("Nanalezeno požadované zařízení");
                return true;
            } else {
                MujLog.setText("Nenalezeno požadované zařízení");
            }
        }
    }
    return false;
}

public boolean BTPriopojeni() {
    try {
        socket = device.createRfcommSocketToServiceRecord(PORT_UUID);
        socket.connect();
        pripojeno = true;
        MujLog.setText("Proběhlo připojení k zařízení");
    } catch (IOException e) {
        e.printStackTrace();
        pripojeno = false;
        MujLog.setText("Neproběhlo připojení k zařízení");
    }
    if (pripojeno) {
        try {
            outputStream = socket.getOutputStream();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return pripojeno;
}

public boolean zkontrolujZaplejBT(BluetoothAdapter bluetoothAdapter) {
    if (!bluetoothAdapter.isEnabled()) {
        Intent enableAdapter = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableAdapter, 0);
        return false;
    }
    return true;
}
```

```
public void NastavButtonyPodleBT() {
    buttonOdpoj.setVisibility(View.INVISIBLE);
    b1.setVisibility(View.INVISIBLE);
    b2.setVisibility(View.INVISIBLE);
    b3.setVisibility(View.INVISIBLE);
    b4.setVisibility(View.INVISIBLE);
    b5.setVisibility(View.INVISIBLE);
    b6.setVisibility(View.INVISIBLE);
    bp2.setVisibility(View.INVISIBLE);
    bp3.setVisibility(View.INVISIBLE);
    bp4.setVisibility(View.INVISIBLE);
    bp5.setVisibility(View.INVISIBLE);
    bp6.setVisibility(View.INVISIBLE);
    bp7.setVisibility(View.INVISIBLE);
    bp8.setVisibility(View.INVISIBLE);
    bp9.setVisibility(View.INVISIBLE);
    bp10.setVisibility(View.INVISIBLE);
    bp11.setVisibility(View.INVISIBLE);
    bp12.setVisibility(View.INVISIBLE);
    bp13.setVisibility(View.INVISIBLE);

    int cisloButtonu = 0;
    BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    Set<BluetoothDevice> bondedDevices = bluetoothAdapter.getBondedDevices();
    zkontrolujZaplejBT(bluetoothAdapter);
    for (BluetoothDevice iterator : bondedDevices) {
        cisloButtonu += 1;
        switch (cisloButtonu) {
            case 1:
                b1.setText(iterator.getName() + "\n" + iterator.getAddress());
                b1.setVisibility(View.VISIBLE);
                b1.setEnabled(true);
                adresaBT1 = iterator.getAddress();
                break;
            case 2:
                b2.setText(iterator.getName() + "\n" + iterator.getAddress());
                b2.setVisibility(View.VISIBLE);
                b2.setEnabled(true);
                adresaBT2 = iterator.getAddress();
                break;
            case 3:
                b3.setText(iterator.getName() + "\n" + iterator.getAddress());
                b3.setVisibility(View.VISIBLE);
                b3.setEnabled(true);
                adresaBT3 = iterator.getAddress();
                break;
            case 4:
                b4.setText(iterator.getName() + "\n" + iterator.getAddress());
                b4.setVisibility(View.VISIBLE);
                b4.setEnabled(true);
                adresaBT4 = iterator.getAddress();
                break;
            case 5:
                b5.setText(iterator.getName() + "\n" + iterator.getAddress());
                b5.setVisibility(View.VISIBLE);
                b5.setEnabled(true);
                adresaBT5 = iterator.getAddress();
                break;
            case 6:
                b6.setText(iterator.getName() + "\n" + iterator.getAddress());
```

```
                b6.setVisibility(View.VISIBLE);
                b6.setEnabled(true);
                adresaBT6 = iterator.getAddress();
                break;
            default:
                break;
        }
    }
    MujLog.setText("Spárovaná zařízení");
}

public boolean PrepniZarizeni(String vybranyPrikaz) {
    if (pripojeno) {
        try {
            outputStream.write(vybranyPrikaz.getBytes());
            MujLog.setText("Příkaz odeslán");
            return true;

        } catch (IOException ex) {
            MujLog.setText("Příkaz se nepodařilo odeslat");
        }
    }
    return false;
}

public boolean Odpoj() {
    try {
        outputStream.close();
        socket.close();
        pripojeno = false;
        MujLog.setText("Aplikace se odpojila od zařízení");
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
}
```

Zdroj: Vlastní zpracování

Příloha XI: Vlastní programové řešení v souboru aktivity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.ouednk.bakalskprceouednk.MainActivity">


    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text=""
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.501"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.016" />


<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/textView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:singleLine="true" />

<Button
    android:id="@+id/buttonOdpoj"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="Odpoj"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.959" />

<Button
    android:id="@+id/b1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.266" />

<Button
    android:id="@+id/b2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
```

```
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.373" />

<Button
    android:id="@+id/b3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.48" />

<Button
    android:id="@+id/b4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.588" />

<Button
    android:id="@+id/b5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.695" />

<Button
    android:id="@+id/b6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
```

```
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.803" />

<Button
    android:id="@+id/nactiBT"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="Znovu načti všechna zařízení"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.502"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.082" />

<Button
    android:id="@+id/bp2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:text="Pin2"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp3"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/bp3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:text="Pin3"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp4"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/bp4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:text="Pin4"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp5"
    app:layout_constraintStart_toStartOf="parent" />

<Button
```

```
    android:id="@+id/bp5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:text="Pin5"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp6"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/bp6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginStart="16dp"
    android:text="Pin6"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp7"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/bp7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="80dp"
    android:layout_marginStart="16dp"
    android:text="Pin7"
    android:visibility="invisible"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/bp8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="16dp"
    android:text="Pin8"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp9"
    app:layout_constraintEnd_toEndOf="parent" />

<Button
    android:id="@+id/bp9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="16dp"
    android:text="Pin9"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/bp10"
    app:layout_constraintEnd_toEndOf="parent" />

<Button
    android:id="@+id/bp10"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginEnd="16dp"
```

```
        android:text="Pin10"
        android:visibility="invisible"
        app:layout_constraintBottom_toTopOf="@+id/bp11"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/bp11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="16dp"
        android:text="Pin11"
        android:visibility="invisible"
        app:layout_constraintBottom_toTopOf="@+id/bp12"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/bp12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="16dp"
        android:text="Pin12"
        android:visibility="invisible"
        app:layout_constraintBottom_toTopOf="@+id/bp13"
        app:layout_constraintEnd_toEndOf="parent" />

    <Button
        android:id="@+id/bp13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="80dp"
        android:layout_marginEnd="16dp"
        android:text="Pin13"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</android.support.constraint.ConstraintLayout>
```

Zdroj: Vlastní zpracování

Příloha XII: Vlastní programové řešení pro řízení mikropočítače

```
char data = 1;
boolean p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13;
void setup()
{
  Serial.begin(9600);
  for (int i = 2; i <= 13; i++)
    pinMode(i, OUTPUT);
}
void loop()
{
  if (Serial.available() > 0)
  {
    data = Serial.read();
```

```
Serial.print(data+"\n");
switch (data) {
  case '2':
    if (p2 ? p2 = false : p2 = true);
    if (p2)
      digitalWrite(2, HIGH);
      else
      digitalWrite(2, LOW);
    break;
  case '3':
    if (p3 ? p3 = false : p3 = true);
    if (p3)
      digitalWrite(3, HIGH);
      else
      digitalWrite(3, LOW);
    break;
    case '4':
    if (p4 ? p4 = false : p4 = true);
    if (p4)
      digitalWrite(4, HIGH);
      else
      digitalWrite(4, LOW);
    break;
    case '5':
    if (p5 ? p5 = false : p5 = true);
    if (p5)
      digitalWrite(5, HIGH);
      else
      digitalWrite(5, LOW);
    break;
    case '6':
    if (p6 ? p6 = false : p6 = true);
    if (p6)
      digitalWrite(6, HIGH);
      else
      digitalWrite(6, LOW);
    break;
    case '7':
    if (p7 ? p7 = false : p7 = true);
    if (p7)
      digitalWrite(7, HIGH);
      else
      digitalWrite(7, LOW);
    break;
    case '8':
    if (p8 ? p8 = false : p8 = true);
    if (p8)
      digitalWrite(8, HIGH);
      else
      digitalWrite(8, LOW);
    break;
    case '9':
    if (p9 ? p9 = false : p9 = true);
    if (p9)
      digitalWrite(9, HIGH);
```

```
        else
        digitalWrite(9, LOW);
      break;
      case 'a':
      if (p10 ? p10 = false : p10 = true);
      if (p10)
        digitalWrite(10, HIGH);
        else
        digitalWrite(10, LOW);
      break;
      case 'b':
      if (p11 ? p11 = false : p11 = true);
      if (p11)
        digitalWrite(11, HIGH);
        else
        digitalWrite(11, LOW);
      break;
    case 'c':
      if (p12 ? p12 = false : p12 = true);
      if (p12)
        digitalWrite(12, HIGH);
        else
        digitalWrite(12, LOW);
      break;
      case 'd':
      if (p13 ? p13 = false : p13 = true);
      if (p13)
        digitalWrite(13, HIGH);
        else
        digitalWrite(13, LOW);
      break;
    default:
    Serial.print("");
   }
  }
}
```

Zdroj: Vlastní zpracování