Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Ovládání domácnosti pomocí mobilní aplikace

2020

Tomáš Prajza

Vedoucí práce: RNDr. Arnošt Večerka

Studijní obor: Aplikovaná informatika, kombinovaná forma

Bibliografické údaje

Autor:	Tomáš Prajza
Název práce:	Ovládání domácnosti pomocí mobilní aplikace
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, kombinovaná forma
Vedoucí práce:	RNDr. Arnošt Večerka
Počet stran:	34
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Tomáš Prajza
Title:	Household control via mobile application
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, combined form
Supervisor:	RNDr. Arnošt Večerka
Page count:	34
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Tato bakalářská práce se zabývá vytvořením mobilní aplikace pro ovládání modelu domácnosti. V úvodní části práce je popsán použitý hardware, zejména centrální prvek, kterým je Raspberry Pi. Následuje popis serverové a mobilní aplikace. Na závěr je popsána komunikace mezi aplikacemi.

Synopsis

First and foremost this thesis deals with the development of a smartphone application meant for management and the control of a household model. The introductory part of this thesis will focus on a detailed description of the hardware that was incorporated in this project, especially the Raspberry Pi around which the whole model is build. Following that will be a thorough analysis and explanation of the server-side software and the smartphone application. Lastly, there will be description of the communication that is use between the server-side software and the application which will be running on a smartphone.

Klíčová slova: Chytrá domácnost; Raspberry Pi; Mobilní aplikace; Android; Xamarin;

Keywords: Smart home; Raspberry Pi; Mobile aplication; Android; Xamarin;

Děkuji vedoucímu bakalářské práce RNDr. Arnoštu Večerkovi za cenné rady a čas, které vedly ke zkvalitnění této práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

1 Úvod

Dnes už téměř každý vlastní chytrý telefon neboli „smartphone“, který neslouží jen pro volání a posílání SMS, jako tomu bylo dříve. Mobilní telefon se stal pro většinu lidí dominantním typem zařízení pro připojení k internetu.
Jeho největší předností jsou mobilní aplikace. První aplikace byly součástí operačních systémů. Jednalo se o aplikace na prohlížení fotek, e-mailů a podobné základní funkce. Časem začaly vznikat i aplikace, které byly mnohem sofistikovanější. Existuje množství nejrůznějších aplikací. Jedno však mají společné a to, že nám usnadňují život. Příkladem takové aplikace, je aplikace na ovládání domácnosti. [4]

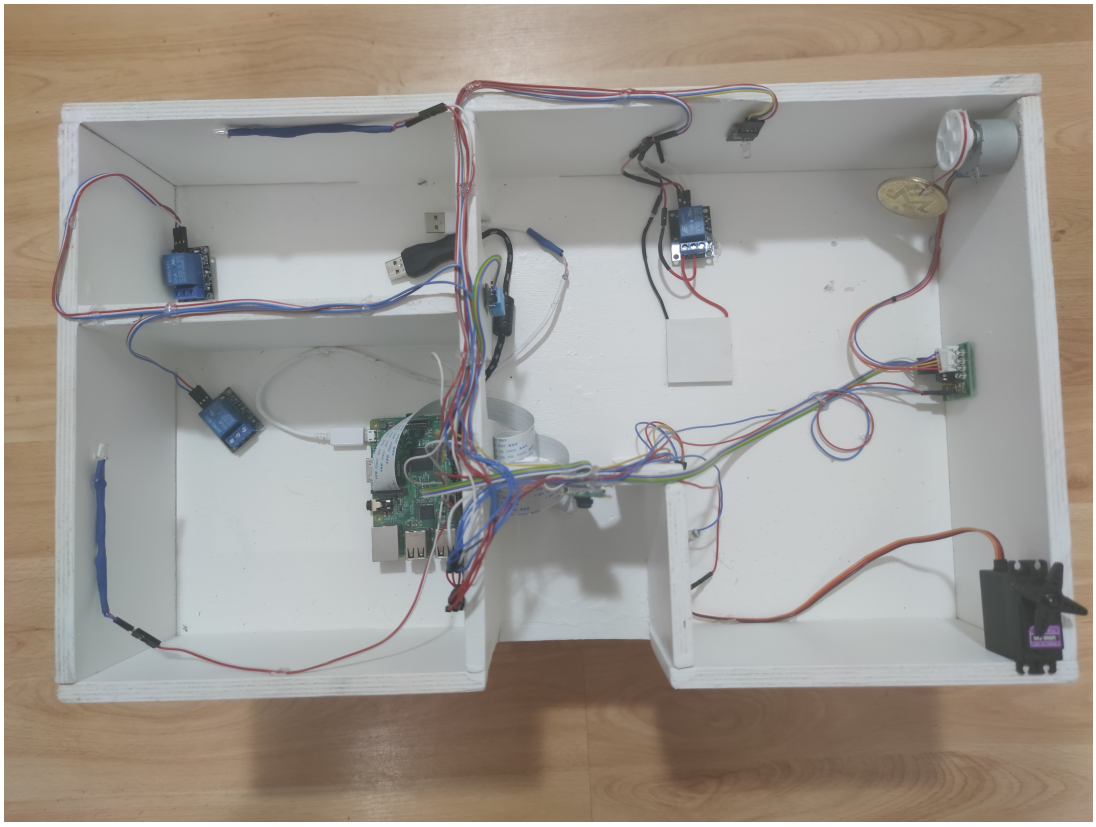
Chytrá domácnost, označovaná jako „smart home“ nám umožňuje mít svůj domov pod kontrolou. Jedním kliknutím v aplikaci můžeme řídit osvětlení a různé spotřebiče. Získáváme také určitý přehled o stavu domácnosti, kdy můžeme sledovat hodnoty z senzorů a čidel. Dále je možné v reálném čase sledovat záznam z kamer, což napomáhá v zabezpečení domácnosti.

2 Cíle

Cílem práce je vytvořit mobilní aplikaci pro OS Android v nástroji Xamarin, která bude umožňovat vzdálené ovládání instalovaných zařízení v modelu domácnosti. Mobilní aplikace bude pomocí Wi-Fi propojena s mikropočítačem Raspberry Pi, který bude ovládat jednotlivá zařízení, jako jsou: světla, otopná tělesa, žaluzie atd. Aplikace bude dále umožňovat stream z kamery v domácnosti.

3 Použitý hardware

Součástí práce je model domácnosti, který můžete vidět na obrázku 1. Tento model simuluje půdorys bytu 2+kk zmenšený v poměru 1 : 20. Obsahuje několik hardwarových zařízení, které jsou v modelu rozmístěny a v následujících kapitolách budou popsány.



Obrázek 1: Model domácnosti

Nejdůležitějším zařízením je Raspberry Pi, sloužící jako centrální prvek domácnosti, k němuž jsou připojeny všechny ostatní zařízení (periferie).

3.1 Raspberry Pi

Jednodeskový mikropočítač Raspberry Pi je počítač založený na platformě ARM. Primárním operačním systémem je Raspberry Pi OS (dříve nazýván Raspbian). Vyvinuto bylo již několik generací tohoto počítače, které se liší výkonem a zamýšleným použitím. [3]

Model 3B (obr. 2), použitý v tomto projektu obsahuje:

- procesor ARMv8 1.2GHz quad-core
- 4x USB 2.0
- 40x piny GPIO
- 1x HDMI s podporou FullHD (1080 p 30 fps)
- 1x ethernet port (10/100Mbit/s)
- 1x slot pro kameru (CSI)
- 1x slot pro displej (DSI)
- 1x 4pinový 3,5 mm audio jack + kompozitní video
- 1x napájecí micro USB port
- 1x slot pro micro SD(XC) kartu
- Wi-Fi 802.11n
- Bluetooth 4.1

Výkonnou funkcí Raspberry Pi je řada kolíků GPIO (univerzální vstup/výstup) podél horního okraje desky. Na současných deskách Raspberry Pi je nalezena 40pinová záhlaví GPIO. Jakýkoli z pinů GPIO lze označit v softwaru jako vstupní nebo výstupní pin a použít pro širokou škálu účelů.

Na rozdíl od počítače Arduino je možné Raspberry Pi použít nejen k ovládání různých zařízení (pomocí GPIO pinů), ale i k samotnému vývoji příslušných aplikací. Lze ho též použít jako multimediální přehrávač videa či hudby nebo jen pro přístup k internetu (dodávané linuxové distribuce obsahují webový prohlížeč a další potřebné aplikace) [5].

3.1.1 Instalace

Raspberry Pi se běžně dodává bez paměti a operačního systému. Jako paměť slouží SD karta, na kterou je třeba operační systém nahrát. Instalace probíhá pomocí programu *Raspberry Pi Imager*. Ten je zdarma ke stažení na webu: https://www.raspberrypi.org.

Obrázek 2: Raspberry Pi 3 Model B.

Při instalaci je třeba zvolit operační systém. Zde je na výběr z mnoha možností. Je důležité zvážit, co se pro daný projekt hodí. Defaultní možnost: *Raspberry Pi OS (32-bit)* je včetně grafického prostředí, které vytěžuje Raspberry Pi a zabírá spoustu místa. Pro toto řešení byla zvolena možnost *Raspberry Pi OS Lite (32-bit)*, která je bez grafického rozhraní. Dále je třeba zvolit SD kartu, na kterou chceme vybraný operační systém nahrát.

Po instalaci a připojení k síti je možné se přihlásit přes zabezpečený komunikační protokol SSH. Výchozí přihlašovací jméno je: *pi* a heslo: *raspberry*.

3.1.2 Nastavení přístupového bodu

Jelikož je potřeba s Raspberry Pi komunikovat, je nezbytné nastavit Wi-Fi síť, ke které se následně bude možné připojit v mobilním zařízení, na kterém bude spuštěna mobilní aplikace.

Raspberry Pi nabízí i možnost nastavení mostu mezi wifi sítí kterou Raspberry Pi vysílá a Ethernetem, který je připojený k Raspberry Pi. To má velkou přidanou hodnotu, jelikož po připojení k Wi-Fi neztrácíme připojení k internetu.

Nejdříve se přihlásíme k Raspberry Pi pomocí protokolu SSH a nainstalujeme program *hostapd* pomocí příkazu `sudo apt install hostapd`, který složí k nastavení přístupového bodu *(Acess pointu)*. Povolíme službu přístupového bodu pomocí následujících příkazů.

```
1 sudo systemctl unmask hostapd
2 sudo systemctl enable hostapd
```

Zdrojový kód 1: povolení hostapd

Instalace je tímto kompletní. Nyní je potřeba přístupový bod nakonfigurovat. Nejdříve je zapotřebí vytvořit most. Pro vytvoření stačí založit nový soubor prostřednictvím příkazu: `sudo nano /etc/systemd/network/bridge-br0.netdev`. V souboru nastavíme následující parametry.

```
1 [NetDev]
2 Name=br0
3 Kind=bridge
```

Zdrojový kód 2: Nastavení mostu

Pro přemostění připojení mezi Ethernetem a Wi-Fi, přidáme rozhraní pro Ethernet, toho docílíme vytvořením následujícího souboru: `sudo nano /etc/systemd/network/br0-member-eth0.network` V souboru nastavíme tyto parametry:

```
1 [Match]
2 Name=eth0
3
4 [Network]
5 Bridge=br0
```

Zdrojový kód 3: Nastavení Ethernetu

Dále je nezbytné povolit službě vytvářet most při startu Raspberry Pi. Toho docílíme spuštěním příkazu: `sudo systemctl enable systemd-networkd`

Pomocí změny souboru: `sudo nano /etc/hostapd/hostapd.conf`. Nastavíme Wi-Fi. V kódu můžete vidět například „country code“, pro Českou republiku je hodnota CZ, dále jméno sítě, heslo a popřípadě další parametry, které je možné nastavit.

Po tomto nastavení je přístupový bod připraven. Stačí pouze restartovat Raspberry Pi pomocí příkazu: `sudo reboot`. [6]

3.1.3 Nastavení streamu z kamery

Nejdříve je nutné kameru připojit k Raspberry Pi. Deska kamery se připevňuje k pomocí plochého kabelu, přímo na určený konektor, který je umístěn na desce Raspberry Pi. Dále je potřeba kameru zprovoznit a nakonfigurovat. Příkazem `sudo raspi-config` se dostaneme k základnímu nastavení Raspberry pi. V sekci *Možnosti rozhraní* zvolíme položku *Camera* a nastavíme příznak *enable* na hodnotu *true*. Po této změně je nutné restarovat Raspberry Pi.

Pro streamování (živý přenos videa) využijeme program motion. Instalace začíná spuštěním příkazu `sudo apt-get install motion`. Dále je potřeba nastavit

```
country_code=CZ
interface=wlan0
bridge=br0
ssid=SmartHome // jmeno sítě
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase= you_foud_an_easter_egg //heslo
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Zdrojový kód 4: Nastavení hostapd.conf

aby program běžel na pozadí, k tomu je potřeba nastavit parametr *start motion daemon* na hodnotu *yes* v konfiguračním souboru `sudo nano /etc/default/motion`.

Na závěr je potřeba nastavit konfigurační parametry videa. Nastavení se nachází v souboru `sudo nano /etc/motion/motion.conf`. Je zde obsažen port pro streamování videa, kvalita a velikost videa.

```
Stream_port=8081
webcontrol_port 8080
Stream quality 50
daemon on
framerate 100
width 640
height 480
```

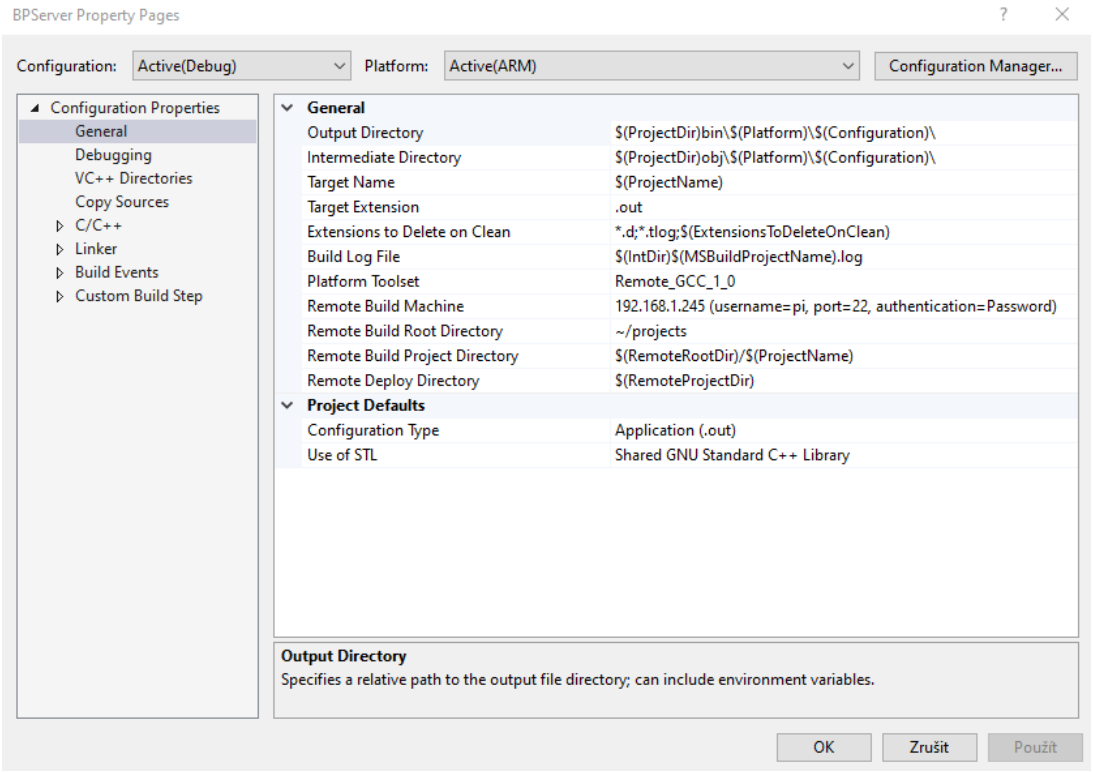
Zdrojový kód 5: Konfigurační soubor - motion.conf

3.1.4 Vzdálená kompilace

Pro pohodlný vývoj na Raspberry Pi je třeba nastavit vzdálenou kompilaci, která nám umožní vzdáleně kompilovat projekt, ve Visual Studiu přímo na Raspberry Pi. Toho docílíme nastavením projektu v průzkumníku řešení.

V nabídce vlastnosti, sekce obecné nastavíme parametr *Configuration Type* na hodnotu: **Application (.out)**.

Dále je nezbytné nastavit parametr *Remote Build Machie*, což je zařízení v síti, na kterém chceme vzdáleně kompilovat. Nabízí se zde seznam zařízení, které je možné nastavit v *Connection Manager*. Pro přidání nového zařízení je



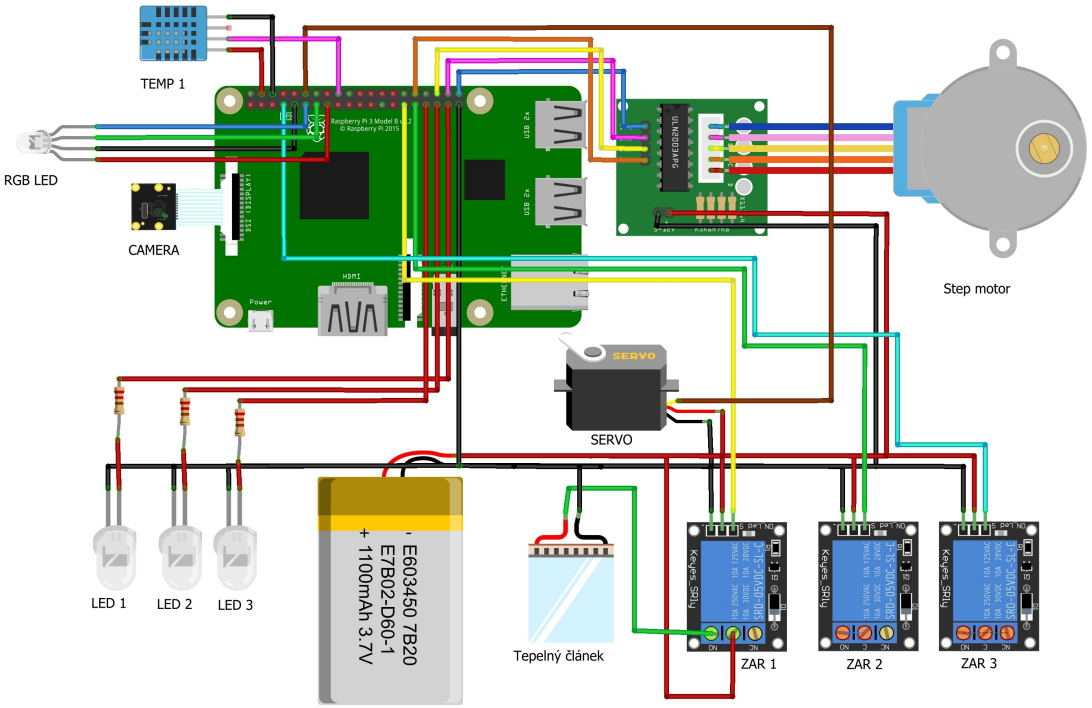
Obrázek 3: Vzdálená kompilace

potřebné zadat: *Host name, port, a přihlašovací údaje k zařízení.* Toto nastavení je zobrazeno na obrázku: 3.

3.2 Periferie

V modelu domácnosti je zapojeno několik periferií, ty jsou jednoduše připojitelné k rozhraní vývojových platforem. Tato zařízení (periferie) jsou připojeny k Raspberry Pi, které je řídí, díky ovládání GPIO pinů.

Periferie můžou být vstupní a výstupní. Vstupní periferií je v tomto řešení senzor teploty a vlhkosti. Jako výstupní periferie budeme používat například LED diody, relé a motory. Použité periferie a jejich propojení s centrální jednotkou (Raspberry Pi) můžete vidět na obrázku 4, který byl vytvořen v programu Fritzing [10]. Tyto periferie v modelu reprezentují světla, různá zařízení a senzory v domácnosti. [13]



Obrázek 4: Schéma propojení periferií

3.2.1 Diody

LED dioda vyzařuje jednobarevné světlo. V tomto řešení je použita bílá dioda o průměru 5 mm, která má dva konektory, kde je delší z konektorů označen jako „+“ nazýván také jako „Anoda“ a kratší „-“ označován jako „Katoda“. Pro připojení k Raspberry Pi, propojíme „+“ s některým GPIO pinem a „-“ propojíme do pinu označeného jako GND (Ground). Takto zapojená LED dioda by nějakou dobu svítila, ale kvůli ochraně, jak diody a především vnitřních obvodů Raspberry Pi, se přidává ještě jedna důležitá část, kterou je rezistor. U obyčejných diod, při napětí 5 V se většinou používá rezistor o odporu 130 ohmů.

(a) LED Dioda

(b) RGB LED

Obrázek 5: Diody (zdroj: [7])

RGB LED dioda oproti běžné LED diodě, umožňuje zobrazovat jakoukoliv barvu. Obsahuje 4 konektory, konektor pro každou základních barev (červená, zelená, modrá) a jeden pro uzemnění (GND). Pro připojení k Raspberry Pi, propojíme konektor GND s pinem „GNR“, stejně jako u LED diody a následně zbylé konektory s GPIO piny. Požadované barevné kombinace pak docílíme tím jaké piny spínáme. Například pro dosažení fialové barvy sepneme GPIO piny pro červenou a modrou na nejvyšší intenzitu. Intenzita je reprezentována jako celé číslo s maximální hodnotou 255.

3.2.2 Relé

Relé funguje na principu spínače, kde se dle signálu řídí zda budou určité části napájeny. U relé modulů jsou na každé straně většinou tři konektory. Pro napájení relé se využívá konektor „VCC“, dále je zde konektor označený jako „GRN“, který je nutné zapojit do pinu „Ground“ na Raspberry Pi.

Obrázek 6: Relé (zdroj: [7])

Na závěr je třeba propojit nejdůležitější konektor „SIG“ (někdy označován taky jako „IN“). Jak už zkratka napovídá, tento konektor slouží pro zasílání signálu a řídí zda jde relé sepnuto. Signálový konektor lze propojit s vybraným GPIO pinem. Jak už bylo řečeno, relé je jen určitý spínač, pro jeho využití je třeba zapojit přívod elektrického proudu a také vybraný spotřebič. Na obrázku: 4, je znázorněno schéma propojení periferií, na kterém lze pozorovat zapojení relé (ZAR 1) a tepelného článku.

3.2.3 Motory

Krokový motor se pohybuje v krocích. K jeho ovládání je potřeba řídící jednotka, obsahující 4 konektory, které je nutné propojit s GPIO piny. Dále obsahuje konektor pro napájení a uzemnění. Nezbytnou součástí je také rozhraní pro připojení k samotnému motoru. Pro řízení předáváme počet kroků, které se mají provést, popřípadě počtem otoček motoru označováno jako „revolution“. Jedna otočka je dána počtem kroků. V tomto řešení byl použit motor „28BYJ-48“ s řídící jednotkou „ULN2003“. Krokový motor v modelu domácnosti simuluje žaluzie, dal by se také využít na další chytré řešení v domácnosti.

(a) Servo (b) Krokový motor

Obrázek 7: Motory (zdroj: [7])

Servomotor označován zkráceně „servo“, oproti klasickému motoru nabízí možnost nastavení přesné polohy (úhlu). Obsahuje tři konektory, stejně jako u ostatních typů periferií jsou zde piny pro napájení a uzemnění. Konektor pro signál je nutné připojit s GPIO pinem. Jelikož servo využívá „PWM“ signál, je třeba zvolit pin podporující tento typ signálu. V tomto řešení je použito servo „MG996R“, které je připojeno na GPIO pinu 18, simulující možnost otevírání oken.

3.2.4 Senzory a čidla

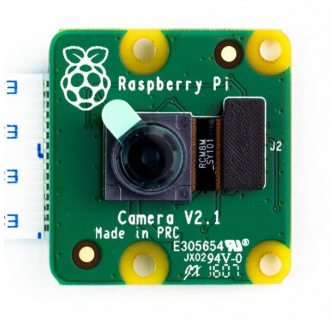
Chytrá domácnost nesouží jen k ovládání zařízení. Díky senzorům a čidlům získáváme nejen přehled, ale sofistikovanější čidla nás můžou upozornit například na pohyb v domácnosti nebo také na špatnou kvalitu vzduchu.

Obrázek 8: Senzor teploty a vlkosti (zdroj: [7])

Model domácnosti obsahuje senzor teploty a vlhkosti „DHT11“, který je schopen měřit teplotu v rozsahu 0 - 60 stupňů Celsia s přesností 2 stupně. Disponuje třemi konektory (signál, napájení a uzemnění). Pro připojení k Raspberry Pi propojíme signálový konektor s GPIO pinem.

3.2.5 Kamera

V modelu domácnosti je umístěna kamera, která je připojena k Raspberry Pi pomocí plochého kabelu. Kamera umožňuje živý přenos z modelu domácnosti přímo v aplikaci. V tomto řešení je použita kamera „Pi Camera Module V2.1“, která je schopna snímat video ve formátu 1080p při 30 fps.



Obrázek 9: Kamera (zdroj: [7])

4 Serverová aplikace

Je konzolová aplikace, která běží na Raspberry Pi. Zajišťuje zpracování dat z mobilní aplikace a vykonání příslušných operací. Základním typem operací je ovládání **GPIO pinů**, na které jsou připojeny periferie. K centrální jednotce je možné připojit až 27 zařízení, toto omezení je dáno počtem GPIO pinů na Raspberry Pi.

Na serveru je umístěn konfigurační soubor **devices.txt**, obsahující seznam zařízení, které jsou na Raspberry Pi připojeny. Tento soubor je nutné při změně periferií aktualizovat a restartovat aplikaci. Zařízení jsou reprezentována jako řádky v konfiguračním souboru. Kde je uveden jejich typ, kanál na kterém komutují a seznam pinů které používají. Při startu serverové aplikace je tento soubor zpracován a zařízení jsou nahrána do paměti jako objekty třídy **„Device“**. Na této třídě jsou zavedeny metody pro nastavení, a ovládání jednotlivých zařízení.

```
1   1;101;26
2   1;102;19
3   1;103;13
4   2;201;5
5   2;202;6
6   2;203;4
7   3;301;22-27-17
8   4;401;24
9   5;501;21-20-16-12
10  6;601;18
```

Zdrojový kód 6: Konfigurační soubor

4.1 Nastavení GPIO

Pro správné fungování je třeba nastavit příslušné piny. Pro inicializaci slouží příkaz `wiringPiSetupGpio()`, kterým zároveň volíme typ číslování pinů a musí být vždy umístěn na začátku metody `main()`. Pro číslování tedy využíváme GPIO adresu. Nastavení pinů probíhá v metodě „Setup“, kde se dle typu zařízení nastaví použité piny (kód: 7). Důležitým bodem je především nastavení **PinMode**, který udává jakého je pin typu.

4.2 Zpracování dat

Na server přichází požadavky. Ty jsou reprezentovány jako textový řetězec. V příchozím textovém řetězci je obsažena dvojice číselných kódů. Kanál (*channel*) a pozice (*position*), tyto kódy jsou odděleny pomlčkou. Při zpracování požadavku se dle kanálu nalezne odpovídající zařízení a je na něm spuštěna metoda „Execute“ (kód: 8) s parametrem pozice.

```
1  void Device::SetUp() {
2    switch (type)
3    {
4      case 3: // rgb led
5      pinMode(pin[0], OUTPUT);
6      pinMode(pin[1], OUTPUT);
7      pinMode(pin[2], OUTPUT);
8      digitalWrite(pin[0], 0);
9      digitalWrite(pin[1], 0);
10     digitalWrite(pin[2], 0);
11     break;
12     .
13     .
14     .
15   }
```

Zdrojový kód 7: Nastavení GPIO pinů

Zpracování dat nekončí pouze provedením dané operace. Vždy je očekávána odpověď, zda byla operace provedena. Pro výstupní periferie musí serverová aplikace odeslat data zpět klientovi. Ty jsou odeslána v podobném formátu, a to v textovém řetězci. Pokud je hodnot více, jsou odděleny speciálním znakem. Například pro čtení dat ze senzoru teploty a vlhkosti server zasílá tyto hodnoty zpět.

```
string Device::Execute(int process) {
  string response = "done"; // odpověď pokud je potřeba (teploměr)
  switch (type)
  {
    case 3: // RGB LED
    switch (process) {
      case 1: // white
      SetColor(255, 255, 255);
      .
      .
      .
      default: // none
      SetColor(0, 0, 0);
      break;
    }
    break;
    case 4: // teploměr
    response = GetTemp();
    break;

    case 5: //motor
    MotorMove(process);
    break;

    case 6: // servo
    ServoMove(process);
    break;

    default: // ledky a zařízení
    Switch(process);
    break;
  }
  return response;
}



void Device::Switch(int x) {
  if (x==1)
  digitalWrite(pin[0], HIGH);
  else
  digitalWrite(pin[0], LOW);

}
```

Zdrojový kód 8: Zpracování dat

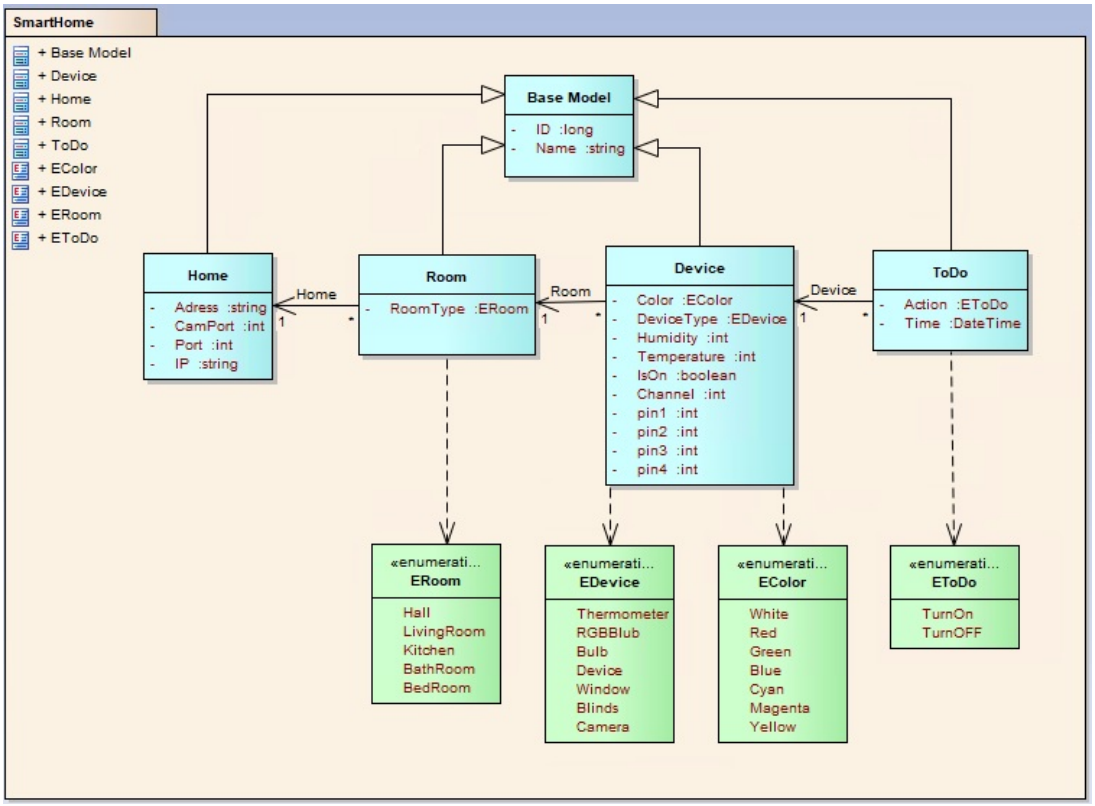
5 Mobilní aplikace

Mezi dva hlavní používané mobilní operační systémy (platformy) dnes patří Android a iOS. Tato práce se zaměřuje speciálně na vývoj mobilní aplikace pro Android. Aplikace je naprogramována na platformě *Xamarin*, ta je spravovaná Microsoftem a je určená pro nativní vývoj aplikací pro operační systémy Android, iOS, Windows, Mac, Tizen a další.

Hlavní funkcí mobilní aplikace je přehledné zobrazení s možností ovládat jednotlivé zařízení v modelu domácnosti. Mobilní aplikace odesílá požadavky na serverovou aplikaci a očekává odpověď. [4]

5.1 Model

Na diagramu tříd (Obrázek: 10) je znázorněn datový model, který zachycuje strukturu a zobrazuje vzájemné vztahy jednotlivých tříd. Model tříd reprezentuje vymezenou část reality, kterou je domácnost obsahující místnosti a jednotlivá zařízení.



Obrázek 10: Datový model mobilní aplikace

Základní třída je *BaseModel*, která obsahuje *id* a *název*. Dědí z ní všechny ostatní třídy.

Třída *Home* má v tomto řešení pouze jednu instanci a to znamená, že máme pouze jednu domácnost s místnostmi *(room)* a zařízeními *(device)*, které můžeme ovládat. Model je připravený na rozšíření, kde by bylo možné spravovat i více domácností. To je zajištěno tím, že na třídě *Home* jsou definovány IP adresy a porty pro připojení k serverové aplikaci.

Třída *ToDo* slouží k definování akcí, které se vykonají automaticky v definovaný čas. Slouží tedy jako plánovač.

Instance tříd *(objekty)* jsou vytvářeny v mobilní aplikaci při prvním spuštění a slouží jako demo. Uživatel může tyto objekty editovat v nastavení aplikace, které je dostupné po kliknutí na ikonu v pravém horním rohu na nástěnce. V následujícím kódu je uveden příklad vytvoření objektů *home* a *device*.

```
1   Homes.AddWithID(new Home()
2   {
3     IP = "10.3.141.1",
4     Port = 8082,
5     Name = "Můj dům",
6     Address = "Hobitín 17, Kraj",
7     CamPort = 8081
8   });
9
10  Devices.AddWithID(new Device()
11 {
12   Name = "Světlo",
13   DeviceType = Models.Enum.EDevice.Bulb,
14   Channel = 102,
15   Pin1 = 19,
16   Room = Rooms.Items.Find(x => x.RoomType == Models.Enum.ERoom.
        Bedroom && x.Home.ID == SelectedHome.ID)
17 }) ;
```

Zdrojový kód 9: Vytvoření objektů

5.2 Aplication view model

Třída uchovávající si stav aplikace, používá se jako singleton. Obsahuje metody pro práci s daty, odesílání požadavků a čtení odpovědi ze serveru. Uchování stavu je zařízeno pomocí listů jednotlivých modelů, případně instancí jednotlivých modelů (Selected). Celá instance třídy *AppVM* lze uložit/načíst do/ze souboru a tím je zaručeno, že uživatelská data jsou k dispozici i po vypnutí a zapnutí aplikace.

5.2.1 Metody třídy AppVM

- *public async Task<bool> SaveDataToFile()* – metoda sloužící pro uložení dat do souboru.
- *public async Task<bool> LoadDataFromFile()* – metoda sloužící pro načítání dat ze souboru.
- *public async Task<bool> TurnOnOffDevice()* – metoda pro zapnutí/vypnutí vybraného zařízení.
- *public async Task<bool> TurnOnOffMultiChoiseDevice(bool changeChoise = false)* – metoda určená pro přepínání stavů více zařízení, které je optimalizované pro více možností jeho běhu (RGB), pokud je parametr changeChoise == false, tak se zařízení vypne/zapne(do jeho posledního stavu), pokud je changeChoise == true, tak se jen mění jeho stav, tzn. změní se barva.
- *public async Task<bool> GetInfoAboutDevice()* – metoda pro zjištění stavu zařízení, která má poskytovat nějaké informace (teploměr).
- *public async Task CreateToDo(ToDo toDo)* – metoda která vytvoří akci v plánovači a zároveň provede uložení akce do souboru.
- *public async Task<bool> UpdateToDo(ToDo todo)* – metoda pomocí které je možné vybranou akci upravit a následně uložit do souboru.
- *public async Task<bool> DeleteToDo(long id)* – metoda pro mazání plánovaných akcí.
- *public async Task EditHome(Home value)* – metoda pro editaci domácnosti. V modelu se zatím počítá pouze s jedinou instancí třídy „Home“.
- *public async Task DeleteRoom()* – metoda pro smazání místnosti.
- *public async Task EditRoom()* – metoda pro editaci místnosti.
- *public async Task CreateRoom()* – metoda pro vytvoření místnosti.
- *public async Task DeleteDevice()* – metoda pro smazání zařízení.
- *public async Task EditDevice()* – metoda pro editaci zařízení.
- *public async Task CreateDevice()* – metoda pro vytvoření zařízení.
- *public async Task<List<ToDo» PrepareDataForToDoAdapter()* – metoda připravující data pro obrazovku s akcemi. Pro vybrané zařízení, vrátí seznam plánovaných akcí.

- *public async Task<List<AuxiliaryModels.DashboardButtons» PreperaDataForDevicesAdapter()* – metoda vracející data pro úvodní obrazovku, vrátí zařízení domácnosti (home), uspořádaná v pomocném modelu, který usnadní horizontální vykreslování jednotlivých zařízení.

5.2.2 FileHelper

Třída slouží pro ukládání/načítání uživatelských dat ze souboru. Uživatelská data (AppVM) se ukládají/načítají v podobě Json a následně jsou převedena do instance třídy AppVM. Používá následující metody.

- *Public async Task<bool> Save* – slouží pro ukládání dat do souboru.
- *Public async Task<bool> Read* – slouží pro načítání dat ze souboru.

5.3 Aktivita

Aktivita *(Activity)* referuje na konkrétní obrazovku naší aplikace a definuje celý svůj život od otevření až do zavření, čemuž říkáme životní cyklus aktivity. Ten obsahuje několik možných fází a metod, které se volají podle toho, jak se mezi fázemi životního cyklu přechází.[2]

Pro své potřeby a usnadnění práce byl vyroben základní předek pro jednotlivé aktivity. Díky předdefinovaným metodám je obecně kód čitelnější a na každé zděděné aktivitě probíhá stejné pořadí volaných metod, případně jsou v tomto sledu volání zavolány i specifické metody potřebné na různých aktivitách. Jako další důležitou položku obsahuje tato aktivita i instanci objektu AppVM.

5.3.1 Metody aktivity

- *Protected virtual async Task OnCreateSetting()* – v této metodě se zavolají níže popsané metody, nastaví se instance objektu AppVM a tím dojde k nastavení aktivity.
- *Protected virtual void SetGuiData()* – metoda pro nastavení grafických prvků, například obarvení ikon dle stavu jednotlivých zařízení, vyplnění hodnot do textových polí apod.
- *Protected virtual void SetGuiReferences()* – metoda pro narefencování grafických prvků z layoutu.
- *Protected virtual void SetEventsListener ()* – metoda pro nastavení eventů, pro grafické prvky a další prvky aktivity.

Aktivita dále obsahuje metody pro spuštění dlouhotrvajících operací, které se spustí na pozadí, uživateli se v době vykonávání operace ukáže loader indikující dlouhotrvající operaci.

```
1     protected override void SetGuiData()
2     {
3       name.Text = home.Name;
4       address.Text = home.Address;
5       ip.Text = home.IP;
6       port.Text = home.Port.ToString();
7       camport.Text = home.CamPort.ToString();
8     }
9   }
```

Zdrojový kód 10: Příklad metody SetGuiData()

```
1     protected override void SetGuiReferences()
2     {
3     leftIcon = FindViewById<ImageView>(Resource.Id.
          SettingHome_AppBarLeftIcon);
4     rightIcon = FindViewById<ImageView>(Resource.Id.
          SettingHome_AppBarRightIcon);
5     name= FindViewById<EditText>(Resource.Id.SettingHome_Name);
6     address = FindViewById<EditText>(Resource.Id.SettingHome_Address)
          ;
7     ip = FindViewById<EditText>(Resource.Id.SettingHome_IP);
8     port = FindViewById<EditText>(Resource.Id.SettingHome_Port);
9     camport = FindViewById<EditText>(Resource.Id.SettingHome_CamPort)
          ;
10    }
```

Zdrojový kód 11: Příklad metody SetGuiReferences()

- *Public async void RunOnBackgroundWithLoader(string msg, Task startTask, Task endTask)* – metoda, která jako parametry přebírá string msg, což je text, který se zobrazí uživateli, dále startTask (dlouhotrvající operaci), a také endTask která se má vykonat po dokončení startTask.
- *Public async void RunOnBackgroundWithLoader(string msg, Func<Task> startTask, Func<Task> endTask)* – metoda funguje jako výše uvedená, jenom místo tasku přebírá jako parametry funkce.

Jako další rozšíření je dostupná metoda pro zobrazení výběrového dialogu, kde se třída, reprezentující jednotlivé prvky výběrového seznamu skládá ze dvou proměnných a to public long ItemID a public string ItemName.

- *public void ShowDialogWithSelection(string header, List<DialogWithSelectionItem> items)* – metoda, která jako paramtery přebírá string, představující hlavičku dialogu, následně list s prvky, ze kterých lze vybírat.

```
1       protected override void SetEventsListener()
2     {
3
4     leftIcon.Click += (s, e) =>
5     {
6       OnBackPressed();
7     };
8     rightIcon.Click += (s, e) =>
9     {
10      RunOnBackgroundWithLoader(GetString(Resource.String.
            Global_Loader), Edit, null);
11    };
12
13    }
```

Zdrojový kód 12: Příklad metody SetEventsListener()

5.4 Uživatelské rozhraní

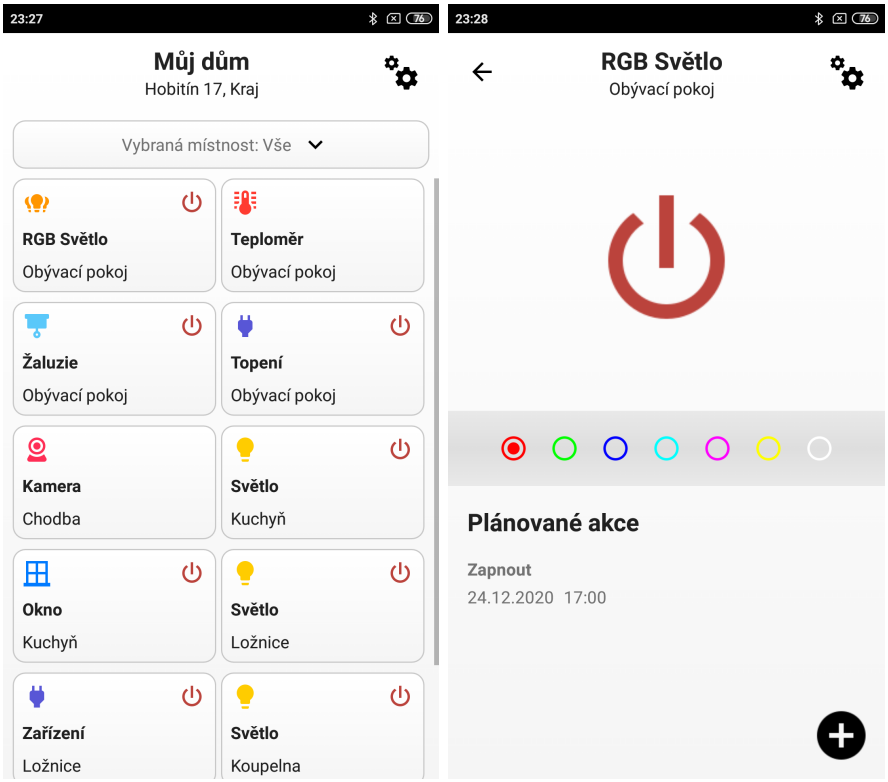
Uživatelské rozhraní, známé také pod zkratkou *UI*, slouží k interakci člověka se strojem (anglicky human-machine interaction). Cílem této interakce je efektivně manipulovat a kontrolovat stroj z pozice člověka mezitím, co stroj kontinuálně informuje člověka o svém stavu tak, aby mohl člověk provádět rozhodnutí. [8]

V této části se konečně podíváme, jak vypadá vytvořená mobilní aplikace. Jedná se o jednoduchou aplikaci, kde se po spuštění otevře dashboard, z anglického překladu „nástěnka“, na které se zobrazují klíčové informace (Obrázek: 11). Na nástěnce vidíme detail domácnosti, její název, adresu a seznam karet, které reprezentují jednotlivá zařízení. Dále je zde výběrový seznam sloužící k výběru místností domácnosti. Jelikož může být v domácnosti zařízení více, je žádoucí, aby si uživatel mohl vyfiltrovat zařízení pro danou místnost. Po kliknutí na kartu se přechází na detail zařízení. Na každé kartě jsou zobrazeny informace o zařízení. Vidíme ikonu zařízení, jeho název, místnost ve které se nachází a také ikonu, která slouží jako rychlá akce a umožňuje uživateli ovládat dílčí zařízení přímo z nástěnky.

Na obrazovce, která slouží jako detail zařízení (obrázek: 11) je umístěna velká ikona pro pohodlné ovládání zařízení. Je zde také zobrazeno umístění zařízení a jeho název. Pro zařízení, která můžou mít více stavů, je umístěna i sekce pro výběr stavu. Například pro RGB světlo je na detailu zobrazen seznam přepínačů, který slouží pro výběr barvy (obrázek: 11).

Obrazovka pro detail zařízení není jednotvárná, přizpůsobuje se typu zařízení, které je zobrazeno. Například pro teploměr se zobrazují hodnoty teploty a vlhkosti v dané místnosti a pro kameru se zobrazí video přes celou obrazovku.

Ve spodní části obrazovky je také umístěna sekce pro možnost zadávání plá-



(a) Dashboard mobilní aplikace (b) Detail zařízení

Obrázek 11: Uživatelské rozhraní

novaných akcí. To nám umožňuje nastavit, kdy se má zařízení vypnout/zapnout. Pro přidání plánované akce slouží *floating button*, který se nachází pravém dolním rohu a přesměruje nás na obrazovku vytvoření plánované akce.

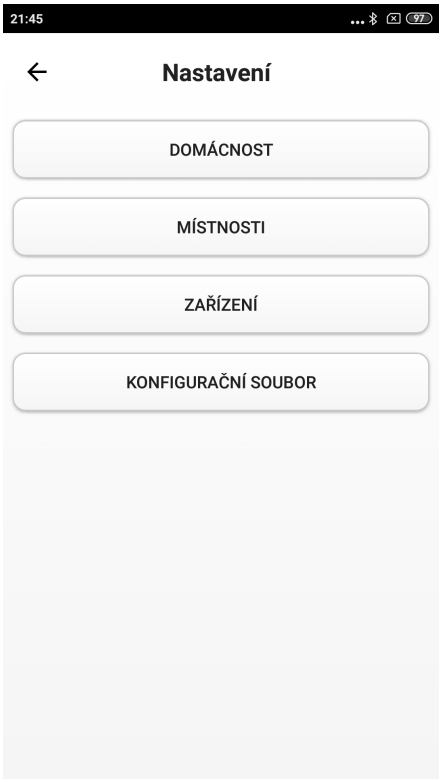
5.5 Nastavení aplikace

Pro možnost uživatelského nastavení je v aplikaci možné editovat jednotlivé entity (domácnost, místnosti a zařízení). Nastavení domácnosti spočívá v možnosti definování IP adresy serverové části, portů a popisných informací.

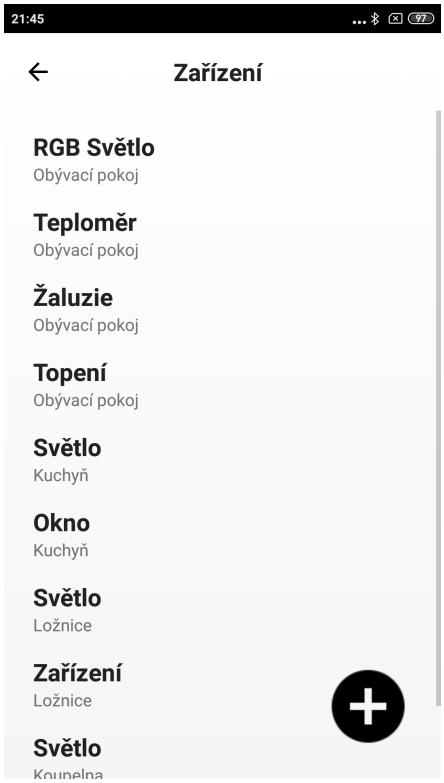
Dále je možné editovat místnosti a zařízení. Možnost definování zařízení přímo v aplikaci, značně rozšiřuje použitelnost aplikace, ale nese s sebou i povinnost toto nastavení udržovat na serverové aplikaci. Obě aplikace totiž musí znát zařízení, kterým rozumí a ví jak s nimi pracovat.

Toto nastavení obsahuje i různé kontroly. Nelze odstranit místnost pokud má navázané zařízení, dále musí být zajištěno, že má zařízení unikátní kanál pro komunikaci se serverovou aplikací.

Dalším rozšířením je generování konfiguračního souboru přímo v mobilní aplikaci a jeho automatické zkopírování do schránky (obrázek: 12 (d)).

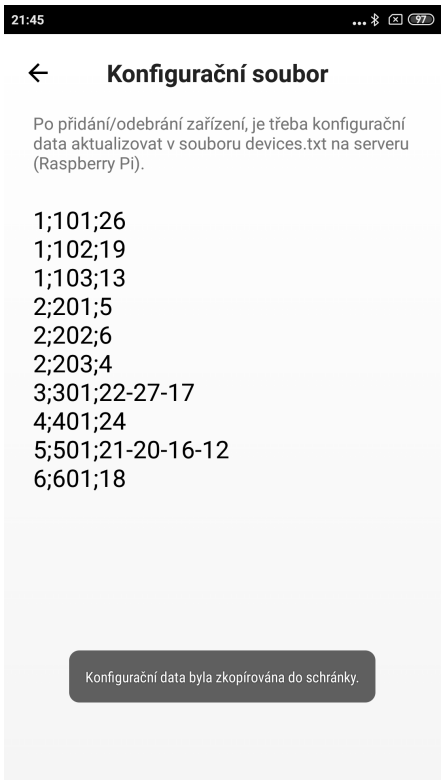


(a) Nastavení



(b) Seznam zařízení

(c) Editace zařízení



(d) Konfigurační data

Obrázek 12: Nastavení aplikace

5.6 Servisní služba

Pro spuštění plánovaných akcí z třídy ToDo, je nutné zajistit, aby na pozadí běžela služba, která bude vykonávat plánované akce, i když je aplikace minimalizovaná. Servisní služba je vlastně démon běžící na pozadí aplikace, který je spuštěn dlouhodobě a není v přímém kontaktu s uživatelem, dále není závislý na tom, zda je uživatel přihlášen. Démon je spuštěn při startu aplikace, jeho úkolem je vyčkávat v nečinnosti na nějakou událost, tu posléze obsloužit a zajišťovat tak různé úkoly bez nutnosti interakce s uživatelem.[9]

Pracuje tak, že se jednou za minutu spustí metoda servisní služby, a ta ověří, zda má nějaký objekt z třídy *ToDo* čas menší nebo roven aktuálnímu času. Pokud ano, provede se akce (je zaslán požadavek na serverovou aplikaci). Po úspěšném provedení je tento objekt odstraněn.

```
1    foreach(var todo in appVM.ToDos.Items.FindAll(x => x.Time <=
        lastTime))
2   {
3     if (appVM.tcpClient.Connect())
4     {
5       if (todo.Device.DeviceType != Models.Enum.EDevice.RGBBulb)
6       appVM.tcpClient.Send(todo.Device.Channel + "-" + ((int)todo.
           Action).ToString());
7       else
8       appVM.tcpClient.Send(todo.Device.Channel + "-" + (((int)todo.
           Action) == 0 ? "0" : ((int)todo.Device.Color).ToString()));
9       var response = appVM.tcpClient.Receive();
10      appVM.Devices.Items.Find(x => x.ID == todo.Device.ID).IsOn = ((
           int)todo.Action) != 0;
11      wasChangies = true;
12      appVM.tcpClient.Disconnect();
13      Console.WriteLine("spusteni akce probehlo");
14    }
15  }
```

Zdrojový kód 13: Metoda servisní služby

6 Komunikace

Pro komunikaci mezi serverovou a mobilní aplikací slouží protokol TCP. Použitím tohoto protokolu mohou aplikace mezi sebou vytvořit spojení, přes které mohou obousměrně přenášet data. Protokol garantuje spolehlivé doručování ve správném pořadí. [12]
Na straně serveru komunikaci obstarávají následující metody.

- *bool getData(int sockfd)* - metoda pro příjem a zpracování dat.
- *void sendData(int sockfd)* - metoda odesílající data klientovi.

Na straně mobilní aplikace komunikaci obstarávají následující metody.

- *Public bool Connect()* – metoda navazující spojení se serverem.
- *Public void Disconnect ()* – metoda ukončující spojení se serverem.
- *Public string Receive()* – metoda která načte odpověď serveru.
- *Public void Send(string message)* – metoda odesílající data na server.

Závěr

Tato práce v úvodní kapitole představila model domácnosti a jeho hardwarové části. Následně byla detailně popsána instalace a nastavení Raspberry Pi, které slouží jako centrální prvek domácnosti. Zevrubně byly popsány periferie a jejich připojení k centrálnímu prvku. Je znázorněno i schéma, které zobrazuje propojení jednotlivých hardwarových částí.

Dále byla představena serverová aplikace, běžící na Raspberry Pi, která zpracovává příchozí data a řídí jednotlivá zařízení v modelu domácnosti. Stěžejní částí bylo vytvoření mobilní aplikace, která slouží k ovládání domácnosti. Byl představen její datový model, princip ukládání dat, použité metody a popis uživatelského rozhraní.

Původní požadavky byly rozšířeny o možnost automatického spuštění plánovaných akcí na jednotlivých zařízeních, což vyžadovalo následné zavedení servisní služby. Dalším důležitým rozšířením byla možnost nastavení zařízení, jak v mobilní, tak serverové aplikaci. Na závěr bylo popsáno komunikační rozhraní mezi aplikacemi.

Conclusions

The introduction of this thesis introduced the household model and its hardware components. Following that in detail was described the installation process and setting up of the Raspberry Pi, which serves as the households central unit. Peripheral units which are connected to the central unit were described thoroughly. A detailed diagram of the individual hardware parts and its connection is included.

Furthermore, server-side software was introduced, which runs on the Raspberry Pi, processing the data and controls the individual peripheral units. The most important part was the development of the smartphone application which is then used to control the whole household model. The data model, the concept of storing the data, the methods that were applied and the description of the user interface were introduced.

The original requirements were extended by the possibility of automatic execution of tasks, which then required the implementation of the services. Another important function was the possibility of changing the settings of the hardware components on both the smartphone side and the server-side. Lastly, the communication interface between softwares was described.

A Obsah přiloženého DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

bin/
: Obsahuje složku s klientskou aplikací, kde je umístěn APK soubor pro instalaci mobilní aplikace, dále složku se serverovou aplikací včetně spustitelného souboru BPServer. V dílčích složkách jsou umístěny všechny zdrojové kódy.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programů se všemi potřebnými (příp. převzatými) zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu.

readme.txt
: Instrukce pro instalaci a spuštění programu Program, včetně všech požadavků pro jeho bezproblémový provoz. / Instrukce pro nasazení serverové aplikace, včetně všech požadavků pro její bezproblémový provoz.

Navíc DVD obsahuje:

video/
: Ukázkové video, které demonstruje provoz modelu domácnosti.

Literatura

[1] MARK REYNOLDS. *Mobile Application Development for Android*. Packt Publishing Ltd, 2014. ISBN 1783559179.

[2] DAN HERMES. *Xamarin Mobile Application Development Cross-Platform C# and Xamarin.Forms Fundamentals*. Berkley, United States, aPress, 2015. ISBN 9781484202159.

[3] UPTON, EBEN A GARETH HALFACREE. *Raspberry Pi: uživatelská příručka. Brno: Computer Press*, 2013. ISBN 978-80-251-4116-8

[4] WIKIPEDIA. *Mobilní aplikace — Wikipedia, The Free Encyclopedia*. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Mobilní_aplikace

[5] WIKIPEDIA. *Raspberry Pi — Wikipedia, The Free Encyclopedia* [online]. Dostupné z: https://cs.wikipedia.org/wiki/Raspberry_Pi.

[6] RASPBERRYPI.ORG. *raspberrypi.org* [online]. Dostupné z: https://www.raspberrypi.org/documentation/configuration/wireless/access-point-bridged.md.

[7] ARDUINO SHOP. *arduino-shop.cz* [online] Dostupné z: https://arduino-shop.cz.

[8] WIKIPEDIA. User interface — Wikipedia, The free encyclopedia [online]. Dostupné z: https://en.wikipedia.org/wiki/User_interface

[9] WIKIPEDIA. *Démon (software) — Wikipedia, The Free Encyclopedia* [online]. Dostupné z: https://cs.wikipedia.org/wiki/D%C3%A9mon_(software).

[10] FRITZING. *fritzing.org* [online]. Dostupné z: Dostupné z: https://fritzing.org/.

[11] PAVEL KAMENÍK. *Příkazový řádek v Linuxu. Brno: Computer Press*. 2011. ISBN 978-8.0-251-2819-0.

[12] RITA PRUŽMANOVÁ. *TCP/IP v kostce. KOPP*. 2009. ISBN 978-80-7232-388-3.

[13] MIKE RILEY *Programming Your Home: Automate with Arduino, Android, and Your Computer*. Pragmatic Bookshelf, 2012. ISBN 1934356905.