ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA TECHNOLOGIÍ A MĚŘENÍ

BAKALÁŘSKÁ PRÁCE

Přenositelnost aplikací mezi OS Android, iOS a Windows Phone

Tomáš Halama **2017**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš HALAMA**
Osobní číslo:	**E13B0116P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komerční elektrotechnika**
Název tématu:	**Přenositelnost aplikací mezi OS Android, iOS a Windows Phone**
Zadávající katedra:	**Katedra technologií a měření**

Zásady pro vypracování:

1. Prostudujte problematiku psaní přenositelného kódu pro malá zařízení na platformách Android, iOS, Windows Phone a proveďte srovnání.
2. Vytvořte přenositelnou aplikaci s GUI (např. na platformách Xamarin, JavaFX apod.) pro všechny uvedené OS a proveďte srovnání.
3. Vytvořte multiplatformní aplikaci využívající hardwarová rozhraní (USB host, popř. BlueTooth, Wi-Fi) pro komunikaci s periferiemi (čidla, komunikace s centrální jednotkou) pro uvedené operační systémy a srovnejte funkcionalitu na uvedených platformách.
4. Zhodnoťte navržený software a možnosti vývoje na zkoumaných platformách.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Apple Developer [online]. (c) 2015 [cit. 31.3.2015]. Dostupné z: https://developer.apple.com**
2. **DiMarzio, J.F. Programujeme hry pro Android 4, Brno: Computer Press, 2012, ISBN 978-80-251-3754-3.**
3. **Android Developers [online]. (c) 2015 [cit. 31.3.2015]. Dostupné z: http://developer.android.com/develop/index.html**
4. **Windows Dev Center [online]. (c) 2015 [cit. 31.3.2015]. Dostupné z: https://dev.windows.com**
5. **Developer Centrum Xamarin [online]. (c) 2015 [cit. 31.3.2015]. Dostupné z: http://developer.xamarin.com**

Vedoucí bakalářské práce: **Ing. Petr Kropík, Ph.D.**
Katedra teoretické elektrotechniky

Konzultant bakalářské práce: **Ing. Aleš Krutina**
Regionální inovační centrum elektrotechniky

Datum zadání bakalářské práce: **15. října 2015**

Termín odevzdání bakalářské práce: **8. června 2017**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Ing. Aleš Hamáček, Ph.D.
vedoucí katedry

V Plzni dne 14. října 2016

Abstrakt

Předkládaná bakalářská práce je zaměřena na porovnání a zhodnocení nástrojů, které umožňují multiplatformní vývoj. V bakalářské práci je popsán postup vývoje ukázkové aplikace, vytvořené nástrojem Xamarin. V této práci je srovnána základní typologie tří vybraných multiplatformních nástrojů Xamarin, Appcelerator Titanium a Gluon Mobile.

Klíčová slova

Android, Windows Phone, Universal Windows Platform, iOS, Xamarin, Appcelerator Titanium, Gluon Mobile, multiplatformní vývoj, TCP komunikace, Model-View-ViewModel, automatizace domu.

Abstract

This bachelor thesis is focused on the comparison and evaluation of tools which allows multiplatfrom development. In the bachelor thesis it is described the process of development of the sample project developed by Xamarin. This thesis compares the basic typology of three multiplatform tools Xamarin, Appcelerator Titanium and Gluon Mobile.

Key words

Android, Windows Phone, Universal Windows Platform, iOS, Xamarin, Appcelerator Titanium, Gluon Mobile, multiplatform development TCP communication, Model-View-ViewModel, home automation.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

……

podpis

V Plzni dne 25.5.2017 Tomáš Halama

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Petrovi Kropíkovi, Ph.D. za cenné profesionální rady, připomínky, odborné konzultace a metodické vedení práce.

Obsah

Úvod

Tato bakalářská práce se zabývá problematikou přenositelnosti kódu pro mobilní zařízení mezi operačními systémy Android, Windows Phone a iOS. Proto v této práci nejsou popsány programovací jazyky JAVA a C#.

V první kapitole se budu zabývat jednotlivým platformám, které přenositelnost kódu mezi různými operačními systémy umožňují. Další část je zaměřena na vytvořenou přenositelnou aplikaci pro mobilní platformy, jež bude komunikovat s centrální jednotkou a sbírat z ní data.

Seznam symbolů a zkratek

SDK.....Software Development Kit

JDK.....Java Development Kit

NDK.....Native Development Kit

IDE.....Integrated Development Environment

XML.....Extensible Markup Language

XAML.....Extensible Application Markup Language

WP.....Windows Phone

iOS.....operační systém společnosti Apple

XNU.....X is Not Unix

UWP.....Universal Windows Platform

OS.....operační systém

ARM.....označení architektury procesorů

JIT.....Just In Time

JNI.....Java Native Interface

ART.....Android Runtime

API.....Application Programming Interface

Windows CE.....Windows Embedded Compact

Windows NT.....Windows New Technology

.NET.....soubor technologií pro osobní počítače s OS Microsoft Windows

IoT.....Internet of Things

UI.....User Interface

ID.....identifikátor

MVVM.....Model-View-ViewModel

1 Analýza mobilních platforem a multiplatformních nástrojů

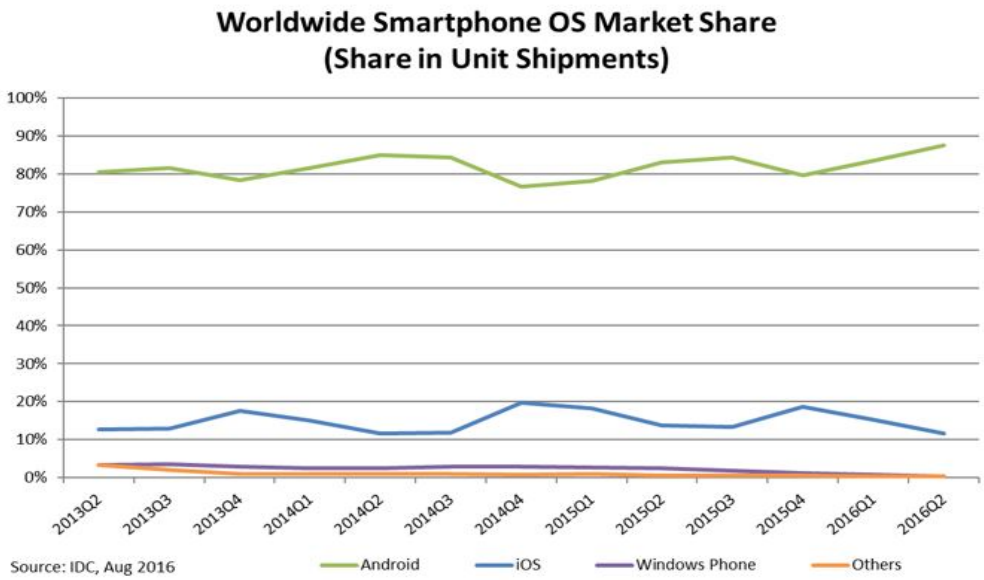
V této kapitole se zabývám popisem nejvíce rozšířených mobilních platforem na trhu. Jsou zde vysvětleny hlavní body vývoje nativní aplikace pro určité mobilní platformy, vytváření uživatelského rozhraní a jakým způsobem se vytvořené aplikace dají zpřístupnit na trh. Dále jsou zde popsány používané frameworky pro tvorbu hybridní aplikace.

1.1 Mobilní Platformy

Na současném trhu jsou nejvíce používány tři platformy: Android, iOS a Windows Phone. Podle analytické společnosti IDC [1] má největší podíl na trhu Android, a to 87,6% ve druhém kvartálu 2016. Na druhém místě se vyskytuje platforma iOS (11,7%), na třetím místě Windows Phone s 0,4% podílu na trhu. Zbylých 0,3% podílu na trhu představují ostatní platformy (BlackBerry OS, Tizen, Bada apod.). Android zvýšil svůj podíl na trhu o 3,3% od třetího kvartálu 2015. Zato podíl na trhu operačního systému iOS klesl o 1,7% a Windows Phone o 0,4%, který zastupuje nová platforma od Microsoftu, Windows 10 Mobile. Problémem Windows 10 Mobile je, že poslední mobilní zařízení (Microsoft Lumia 650) přímo určené pro tento operační systém uvedl Microsoft na trh naposledy v únoru roku 2016. [2] Údajně bude Lumia 650 poslední Lumií vůbec a Microsoft plánuje pro rok 2017 představit Surface Phone.

Z tohoto výzkumu lze usoudit, že v dnešní době se spíše vyplatí vyvíjet aplikace pro Android, iOS a univerzální Windows 10, který lze spustit jak na desktopu, tak na mobilním zařízení s Windows 10 Mobile.

V této práci se zaměříme na dvě nejvíce používané mobilní platformy, tj. Android a iOS. Také se zde budeme zabývat operačním systémem Windows 10.



Obr.1: Podíl na trhu jednotlivých platforem (převzato z [1])

Android

Android[3], představující největší podíl na trhu, je postaven na Linuxovém jádře, díky čemuž je open source. OS Android vznikl jako projekt stejnojmenné společnosti Android (2003), ale v roce 2005 jej odkoupila společnost Google. [4]

Android aplikace se kompilovaly nejdříve v JAVA virtual machine, ty se poté kompilovaly ve virtuálním stroji Dalvik, který využíval tzv. Just-in-time kompilaci, kdy se kompilace provede před spuštěním aplikace. Android v roce 2013 (od verze 4.2) představil nový způsob kompilace s názvem ART (Android RunTime). Jedná se o Ahead-Of-Time kompilaci, kdy se program kompiluje v průběhu instalace aplikace. [5] Cílem ART měla být větší plynulost běhu aplikace. V roce 2016 byl Dalvik plně nahrazen kompilátorem ART. [6]

Podle testu, kde se porovnávají tyto dva virtuální stroje z vybraných benchmarků [7], je ART v některých případech o více než 50% úspornější.

Nativní aplikace pro Android jsou popsány v programovacím jazyce JAVA. Pro jejich vývoj je potřeba mít nainstalovaný JAVA SDK a Android SDK (Software development kit), který obsahuje API (Application Programming Interface) knihovny a pro vývoj potřebné vývojářské nástroje, nastavení a ladění. Jako vývojové prostředí se používá

Eclipse IDE, do kterého je ovšem zapotřebí nainstalovat Android Developer Tools. Velice oblíbeným programem mezi vývojáři je IntelliJ IDEA s přímou podporou Android aplikací. Dále lze do používaných vývojových prostředí pro Android zahrnout Android Studio a NetBeans.

Uživatelské rozhraní si mohou standardně upravovat, hlavní obrazovku dokonce přizpůsobit i uživatelé pomocí aplikací třetích stran. Napříč tomuto faktu ovšem existují základní standardy UI nastavené Androidem na jejich developerských stránkách. Na horní straně se nachází Status Bar, který se dá vysunout. Pod ním se nachází Action Bar, který slouží jako navigační prvek.

Návrhový vzor pro Android aplikace se nazývá Model-View-Presenter (MVP). [8] Vrstva Model reprezentuje třídy přistupující k datům. Vrstva View se implementuje pomocí XML souborů, kam se přidávají prvky pomocí tagů. Vrstva Presenter slouží jako mezivrstva, která konvertuje a selektuje data, které jsou potřebné k zobrazení.

Oficiálním místem pro prodej Android aplikací je Google Play. [9] Vývojář je povinen odeslat aplikaci na Google Play ve formě balíčku s příponou APK. S hierarchií open source systému se Google rozhodl, že pro distribuci nové aplikace není potřeba schvalovacího procesu. Jedinou překážkou je registrace vývojáře na Google Play Developer Console a zaplacení poplatku 25 amerických dolarů.

iOS

iOS je mobilní operační systém založený společností Apple Inc v roce 2007. Operační systém je založený na jádře XNU, podobně jako OS X. S OS X má taktéž společné frameworky Core Foundation a Foundation Kit.

Operační systémy společnosti Apple jsou poměrně uzavřený systém, proto pro vývoj aplikací je potřeba mít Mac počítač s OS X, vývojové prostředí Xcode a samozřejmě iOS SDK. Aplikace jsou programovány v jazyce Objective-C. Testovat aplikace lze pouze v zabudovaném emulátoru v programu OS X. [10]

Základní principy uživatelského rozhraní se podobají UI Android. V horní části se nachází vysouvatelný Status Bar. Na spodní straně obrazovky může být zobrazený panel s nástroji. Výhodou programování uživatelského rozhraní pro iOS je, že existuje poměrně malý a přesně daný počet typů obrazovek.

Návrhový model, typický pro iOS aplikace, se nazývá Model-View.Controller (MVC). Tento model je důležitou součástí komunikace mezi daty a jejich zobrazení. Vrstva Model reprezentuje data a operační logiku nad nimi, vrstva View jsou všechny kontejnery a kontrolní prvky zobrazené na obrazovce. Vrstva Controller obsahuje obsluhu událostí ovládacích prvků a obsluhu událostí spojených s životním cyklem stránky. [11]

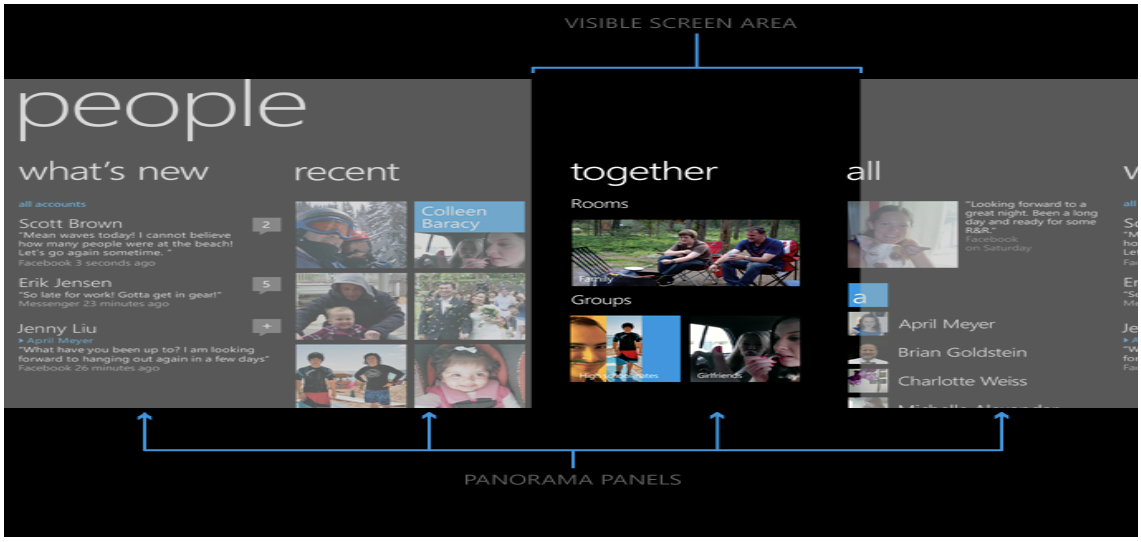
Aby mohl vývojář testovat aplikaci na reálném zařízení, je potřeba se zaregistrovat na iOS Developer Program a platit ročně 99 amerických dolarů. [12] Aplikace se dají zveřejňovat na oficiálním iTunes App Store. Pro publikaci aplikace je potřeba projít testovacím procesem, procesem zabezpečení a schválení aplikace. Aplikace se musí zabalit do bezpečnostního balíčku přes distribuční profil, který identifikuje aplikaci. Takto vytvořený balíček se poté odešle přes webový nástroj iTunes Connect spolu s informacemi o aplikaci na schválení.

Windows Phone

Platforma Windows Phone je vyvíjena společností Microsoft od roku 2010. Předchůdcem operačního systému WP byl Windows Mobile založený na jádře systému Windows CE. Vzhled tohoto systému se velice podobal klasickému desktopovému Microsoft Windows. V roce 2010 představil Microsoft novou mobilní platformu Windows Phone 7, založenou na jádře Windows CE, který byl zaměřen již více na běžné uživatele a uživatelské rozhraní bylo přívětivější a intuitivnější než předchůdce. Toto specifické prostředí se nazývá Metro, které tvoří tzv. dlaždice. V roce 2012 byl vydán operační systém druhé generace řady Windows Phone, a to Windows Phone 8. Tento systém používá jádro ze systému Windows NT. Díky rozdílným jádrům nejsou systémy WP 7 a WP 8 kompatibilní. Nejnovějším mobilním systémem je Windows 10 mobile, který byl představen v roce 2015. Jeho velkou předností je sjednocení desktopového zařízení s mobilním zařízením. Tím lze vytvářet jednotná prostředí univerzálních aplikací. Windows 10 mobile běží na architektuře procesorů ARM. [13]

Nativní aplikace pro Windows Phone jsou popsány v jazyce C++ nebo C# na platformě .NET. Oficiální vývojové prostředí je IDE Visual Studio, které již obsahuje Windows Phone SDK, takže jej netřeba zvlášť instalovat. SDK obsahuje všechny potřebné nástroje včetně emulátoru pro testování aplikace. Nejlepší je ovšem testovat aplikace na reálném Windows Phone telefonu.

Uživatelské rozhraní je popsáno jazykem XAML, který vychází z XML. Ten definuje vytváření instancí jednotlivých ovládacích prvků a jejich vzájemné vnořování. Hlavní zásady vzhledu Windows Phone popisuje Metro design language, jehož hlavní myšlenkou je jednoduchý grafický návrh a práce s typografií. Metro tvoří tzv. dlaždice, které jsou interaktivní. Obrazovka běžící aplikace se dělí na tři základní části. V horní části se nachází Status bar, který informuje o stavu systému. Pod ním je oblast pro ovládací prvky aplikace a ve spodní části se nachází Application bar- volitelná součást aplikace, na které se mohou zobrazovat ikony s často používanými akcemi. Aplikace jsou většinou strukturované do úrovní, kde největší ovládací prvek je Panorama Control [14], který reprezentuje obsah rozdělený do několika sekcí.



Obr.2: Panorama control (převzato z [14])

Jako návrhový vzor slouží Model-View-ViewModel (MVVM). Tato architektura rozděluje kód do třech vrstev. Vrstva Model (přesněji Data Model) zpřístupňuje data z úložiště, v případě potřeby i operační logiku nad nimi. Vrstva View obsahuje definici uživatelského prostředí v jazyce XAML, obsluhu událostí a kód, který pracuje s uživatelským rozhraním. ViewModel spojuje předchozí dvě vrstvy a zprostředkovává data z vrstvy Model v přístupné formě pro vrstvu View. Výhodou tohoto modelu je snadná

změna uživatelského rozhraní, aniž by se muselo zasahovat do jiného kódu, protože to je vytvářeno výhradně jazykem XAML, a proto je zcela izolováno od vrstev dat. [15]

Hotová aplikace pro Windows Phone se dává na oficiální místo Windows Store. Aby tato hotová aplikace mohla být vložena do Windows Store, musí být vývojář zaregistrovaný ve Windows Dev Center s poplatkem 19 amerických dolarů, pro firmy je cena 99 amerických dolarů. Před vložením na trh musí aplikace projít schvalovacím procesem. Aplikace se zabalí spolu s popisem a obrázek aplikace do balíčku s příponou APPX. Schvalovací proces má několik etap. První tři etapy Pre-processing, Security a Technical compliance se provedou do několika hodin, poté funkčnost obsahu provádí zaměstnanci, a to může trvat několik dní. [16]

1.2 Nativní aplikace vs Hybridní aplikace

Nativní aplikace

Nativní aplikace se vytvářejí pro konkrétní platformu a hardware (fotoaparát, GPS apod.). Tyto aplikace jsou popsány programovacím jazykem pro dané platformy. Tedy, když vyvíjíme čistě nativní aplikaci a chceme ji použít do více platforem, musíme pro každou platformu vytvořit vlastní implementaci uživatelského rozhraní a datového modelu. V podstatě vytvoříme různé aplikace, co se týče vnitřní logiky. Výhodou těchto aplikací je, že jsou stavěny na dané platformy, a proto jsou rychlé a spolehlivé. Nevýhodou je nepřenositelnost kódu, cena (pro každou platformu jiný vývojový tým) a nutnost použít více programovacích jazyků.

Hybridní aplikace

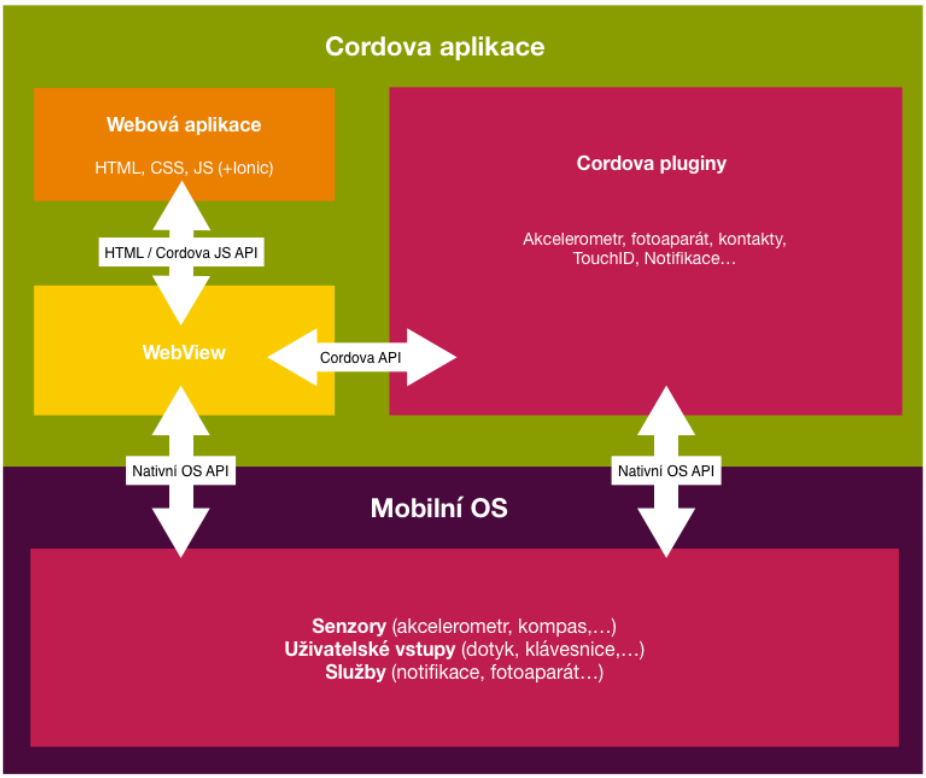
Hybridní aplikace je kompromis, který nenabízí takovou rychlost jako nativní aplikace, ale jsou konstruovány pro dobrou přenositelnost mezi jednotlivými platformami a pomocí pluginů využívají hardware jednotlivých platforem. Míru přenositelnosti určuje především nástroj použitý pro vývoj. Pro tento vývoj se používají nástroje třetích stran, které umožňují psát kód v jednom programovacím jazyce, dodávají jeho unifikované API a aplikační vrstvu, která se stará o zobrazování uživatelského rozhraní a překládání univerzálních volání API do specifických volání API pro danou platformu.

1.3 Framework pro multiplatformní aplikaci

Framework je nástroj, který umožňuje vyvíjet multiplatformní aplikaci. Obsahuje podpůrné programy, knihovny apod. Na trhu jich je celá řada a liší se svým přístupem pro vývoj těchto aplikací. Některé frameworky jsou založené na webových technologiích (HTML5, CSS, JS), jiné používají jazyk JAVA, C# nebo C++.

Cordova

Cordova [17] je nástroj určený pro vývoj aplikací v jazyce HTML, Javascript a CSS. Cordova je jakýsi obal, který pomocí šablon, jež mají jednu WebView komponentu, zabalí vytvořenou hybridní aplikaci do výsledného balíčku a vytvoří nativní aplikaci pro určitý OS. Šablony se generují podle toho, na jaké platformy chceme aplikaci podporovat. Cordova obsahuje pluginy, které zpřístupňuje nativní API (fotoaparát, filesystém apod.).



Obr.3: Diagram Cordova hybridní aplikace (převzato z [16])

jQuery Mobile

Unifikovaný framework využívá HTML5 [18] a CSS3 vlastnosti. Staví na jQuery a jQuery UI, což jsou knihovny pracující s JavaScriptem. Dá se říci, že jQuery Mobile

rozšiřuje klasické jQuery (JS) o HTML5 a CSS3. Je to jeden z nejznámějších hybridních frameworků, které jsou zadarmo. [19] Má velkou komunitu a podporuje všechny tři námi zkoumané platformy.

Ionic

Ionic [20] je framework obsahující HTML5 SDK, pomocí něhož lze vyvíjet software (hybridní aplikace). Vytváří aplikace pomocí webových technologií (HTML, CSS, JS). Je postaven na frameworku AngularJS. Ionic se zaměřuje především na vzhled a interakci s UI aplikace. Tento framework je poměrně mladý a bohužel nepodporuje Windows Phone. Je k dostání zadarmo.

PhoneJS

Tento framework [21] je zaměřen na tři hlavní mobilní platformy: Android, iOS a Windows Phone. Díky tomuto faktu má velice atraktivní nativní vzhled UI. Je založen na frameworku Knockout, který používá knihovnu JS a rozšiřuje tuto knihovnu o HTML s CSS. PhoneJS lze zadarmo použít pouze pro nekomerční účely.

Gluon

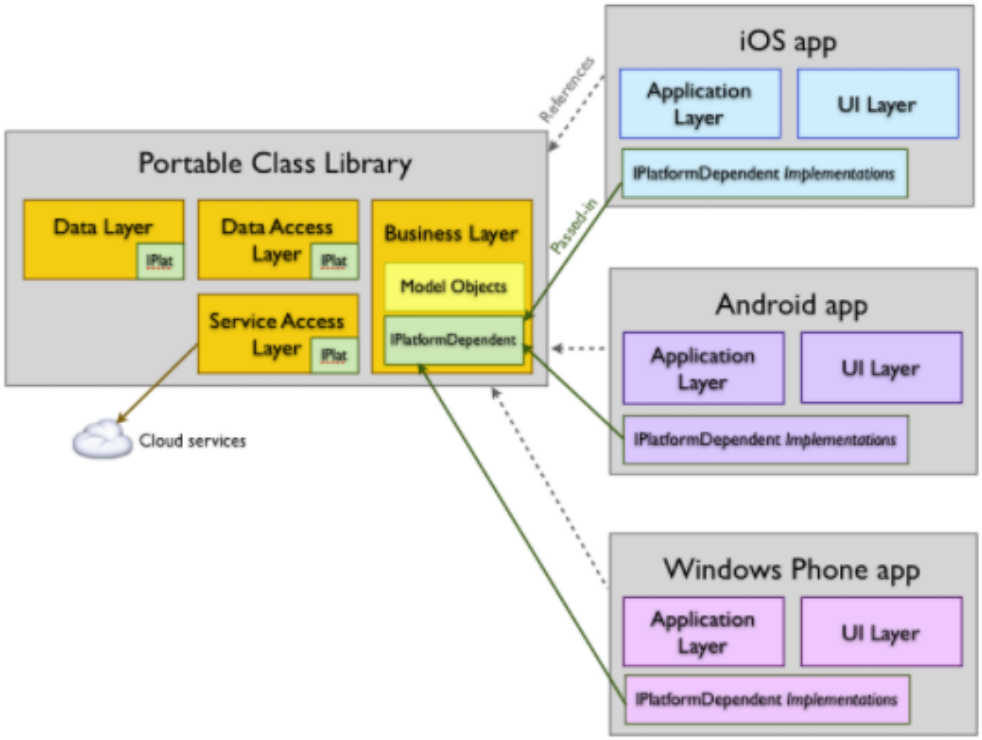
Gluon [22] umožňuje programovat multiplatformní aplikace programovacím jazykem JavaFX. Framework se skládá ze dvou hlavních prvků. První částí je Gluon Mobile, která obsahuje knihovny ovládací prvky uživatelského rozhraní na straně klienta. Dále pak obsahuje důležité složky pro komunikaci se službami třetích stran a pro komunikaci s další části tohoto frameworku, kterým je Gluon CloudLink, starající se o serverovou část. Gluon cílí hlavně na Android, iOS a desktopové systémy Windows. Pro vývoj aplikací na operačních systémech Android a Windows je dostupný plugin ve vývojovém prostředí NetBeans, Eclipse a IntelliJ IDEA. Aplikace pro operační systém iOS je potřeba programovat na počítači s OS X.

Appcelerator

Hlavním produktem společnosti Appcelerator je framework Appcelerator Titanium. Titanium pak dále rozšiřuje Appcelerator Platform [23], který umožňuje spravovat celý životní cyklus aplikace a analyzovat je. Tento multiplatformní nástroj je open-source. Mobilní aplikace se programují ve vývojářském prostředí Appcelerator Studio za pomocí programovacího jazyka JavaScript. Hotové hybridní aplikace se pak kompilují do nativních Android a iOS aplikací.

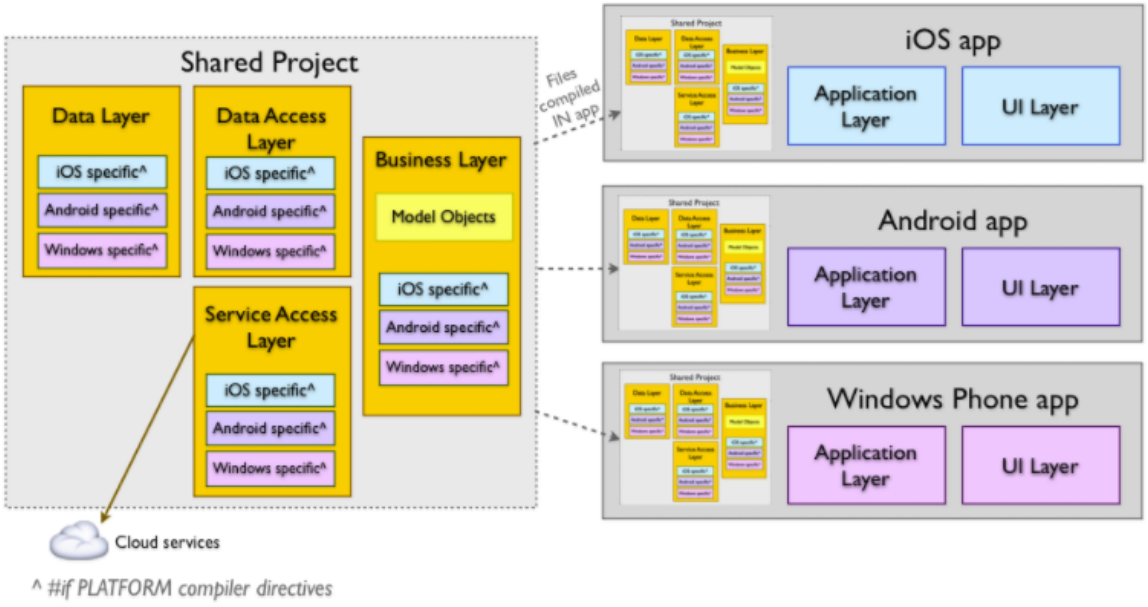
Xamarin

Xamarin umožňuje vyvíjet aplikace pro Android a iOS v jazyce C# (pro Windows Phone se píší aplikace v C# – není třeba Xamarin používat). [24] Xamarin vlastní své IDE, ale lze jej použít i ve Visual Studio IDE. I když se aplikace píší ve stejném jazyce a jednotlivé platformy sdílí stejný zdrojový kód, výsledná implementace je na každé platformě odlišná. Aplikace pro iOS jsou kompilované do ARM assembleru, připojí se .NET framework a nevyužité třídy se odstraní pro snížení velikosti aplikace. iOS neumožňuje generování kódu za běhu, proto jsou některé funkce omezeny. Aplikace pro Android jsou zkompilovány do IL (byte code), poté jsou zabaleny do MonoVM + JIT. Aplikace poté zároveň běží s JAVA/ART (Android Runtime) a spolupracuje s nativními typy pomocí JNI. Xamarin je rozdělen na 3 hlavní podprodukty- Xamarin.Android, Xamarin.iOS a Xamarin.Forms. Xamarin byl placený produkt s taxou 999 dolarů/rok, pro studenty a pedagogy zadarmo. To již není pravda, protože Microsoft [25] koupil produkty Xamarinu v únoru roku 2016 a nyní je ke stažení zdarma.



Obr.4: Základní techniky pro sdílení multiplatformního kódu - Portable (převzato z [27])

V Xamarin nelze vyvíjet všechny tři platformy na stejném operačním systému. V OS X nelze vyvíjet aplikace pro Windows Phone, protože operační systém neobsahuje WP SDK. Pro vyvíjení iOS je zase třeba OS X. Tím však lze docílit, že aplikaci pro iOS budeme vyvíjet na počítači s OS Windows ve Visual Studio IDE a k němu budeme mít připojený přes síť vzdálený přístroj s OS X. [26]



Obr.5: Základní techniky pro sdílení multiplatformního kódu - Shared (převzato z [27])

Vývojář si v Xamarin frameworku může zvolit, jakým způsobem bude chtít aplikaci vytvořit [27]. První možností je vytvoření nativních aplikací pro každou platformu zvlášť pomocí knihoven Xamarin.Android a Xamarin.iOS. K vyvíjení multiplatformní aplikace lze použít takzvaný sdílený projekt (Shared Project). Sdílený projekt dovoluje volat specifické knihovny .NET jádra zvlášť pro každou platformu. Další možností multiplatformního projektu je Portable Class Library (PCL). Jedná se o rozšířenější variantu, kde PCL je knihovna, v které lze vytvářet kód za použití omezeného .NET frameworku. Tento výběr frameworku se provádí při založení nového projektu. K .NET nástrojům se přidají zároveň i knihovny Xamarinu. I přes omezení .NET lze nahradit chybějící knihovny nainstalováním knihoven třetích stran.

2 Tvorba aplikace Home Automation

2.1 Návrh projektu

Cílem této bakalářské práce je vytvoření mobilní aplikace, která komunikuje s čidlem nebo centrální jednotkou pomocí hardwarového rozhraní, jako je například Wifi, Bluetooth, USB host. Aplikace musí být vytvořena nástrojem, jenž umožňuje multiplatformní vývoj, čili za použití jednoho programovacího jazyka a unifikovaného kódu pro více operačních systémů (Android, iOS, Windows 10 mobile). K vyřešení hlavních otázek, jaký framework a z čeho sbírat data, jsem si vytvořil myšlenkovou mapu (příloha B), na níž jsou vyobrazeny různé možnosti multiplatformního vývoje, čidel, způsob připojení apod.

Po porovnání různých možností a konzultaci s vedoucím práce jsem se rozhodl, že bych chtěl prozkoumat potenciál multiplatformního vývoje společnosti Microsoft, a proto byla vytvořena aplikace pomocí nástroje Xamarin s hlavním programovacím jazykem C#. Zařízení, z kterého bude mobilní aplikace sbírat data, je komplexního charakteru. Toto zařízení je rozděleno na dvě části. První částí je Raspberry Pi 3, který slouží jako hub a je centrální jednotkou pro sběr dat. Na tento hub je připojená druhá část, Arduino Uno, k níž je připojeno teplotní čidlo, PIR senzor a relé shield pro ovládání světel či jiného zařízení. Tento systém lze použít k automatizaci domu. Celá práce je rozdělena na 3 projekty. První projekt je pro Arduino Uno, psaný programovacím jazykem C++ ve vývojovém prostředí Arduino IDE. Další projekt je realizovaný pro Raspberry Pi, který se stará o sběr, následné zobrazení dat a změnu stavů relé. Tento projekt je napsán v prostředí Visual Studio programovacím jazykem C# jako Universal Windows Platform, protože jako operační systém byl zvolen Windows 10 IoT Core. Třetí projekt je aplikace pro mobilní telefon, jejíž nástroje jsou již zmíněné na začátku tohoto odstavce.

2.2 Návrh komunikace

Vzájemná komunikace mezi jednotlivými systémy je velice důležitá část projektu. Pro komunikaci mezi Arduino Uno a Raspberry Pi byla vybrána I^2C komunikace. Komunikace mobilního zařízení s Raspberry Pi byla realizována TCP protokolem.

I²C sběrnice

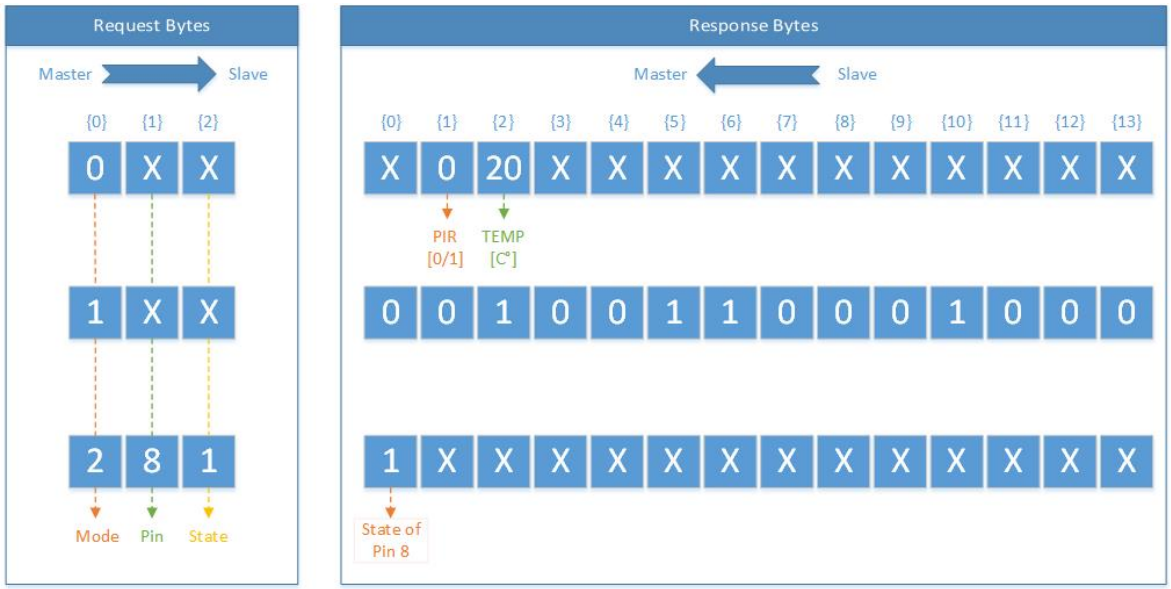
I²C je datová interní sběrnice sloužící pro komunikaci a přenos dat mezi nízkorychlostními periferiemi a základní deskou [28]. Základní desku lze také nazvat jako nadřazenou (master), která zahajuje a ukončuje komunikaci (v našem případě nadřazenou složkou bude Raspberry Pi). Periferiím se říká podřízená (slave) zařízení, adresovaná nadřazeným systémem. Přenos je obousměrný a probíhá pouze po dvou vodičích (data SDA a hodiny SCL). Na jednu společnou sběrnici je možné připojení více podřízených zařízení s různou adresou. To nám umožňuje dívat se na Arduino zařízení jako na jednu místnost domu.

Pro správnou funkci je potřeba si nejdříve nadefinovat, jak bude probíhat komunikace nadřízeného s podřízeným. Vzhledem k připojeným čidlům a relé shieldu se určily tři příkazy:

- Přečti stav senzorů (Read sensors)
- Přečti stav (Read device´s state)
- Nastav stav (Set device state)

Následně je nadefinován protokol, který určuje pravidla pro komunikaci po sběrnici. Jedná se o posloupnost bytů, které jsou posílány do cílových periferií. Nadřazené posílá žádost (request) danému podřízenému zařízení, které poté provede požadovanou akci a vyšle odpověď(response).

Pro komunikaci master → slave jsou přiřazeny 3 byty. První byte nabývá hodnoty 0-2. Ten určuje mód, jaká akce se provede (0 – Read Sensors, 1 – Read Device´s State, 2 – Set Device State). Druhý byte stanoví, jakému pinu na Arduino desce je žádost určena. Třetí byte nastavuje na daném pinu hodnotu 0 nebo 1 (zapnout/vypnout). První a druhý byte je určený pouze pro mód 2.



Obr.6: Příklady komunikace po I²C sběrnici (vlastní)

Komunikaci slave → master je přiřazeno 14 bytů. Tyto byty jsou obrazem pinů na Arduino desce (0-13 a A0-A3). Pokud nadřízený pošle žádost: Read Sensors, podřízené zařízení zaznamená hodnoty čidel a poté je pošle. Tato čidla jsou na předem vyhrazených pinech desky a odesílají se byty pouze jen z těchto pinů. Po žádosti 1 – Přečti stav odešle podřízený všechny byty, které nabývají hodnoty 1 nebo 0 (zapnuto/vypnuto). Mód 2 nastaví stav daného pinů na 1 nebo 0, poté podřízené zařízení odešle odpověď o změně stavu pouze jedním bytem, který nabývá hodnoty 1 a 0.

TCP komunikace

TCP je často používaným protokolem v internetové síti. [29] Přes TCP protokol lze obousměrně přenášet data. Předností této komunikace je spolehlivé doručování ve správném pořadí, rozlišování a rozdělování dat pro více aplikací.

Pro připojení dvou zařízení je potřeba mít na jedné straně klienta a na druhé server. Pro správné navázání je potřeba definovat port u obou zařízení. V našem případě se jedná o port 1200. Server na daném portu naslouchá do té doby, dokud se na ten samý port nepřipojí klient. Pak probíhá komunikace.

Pro odeslání dat není potřeba v C# definovat byty, jako tomu bylo u I²C, ale stačí zprávu ve tvaru *string* vložit jako parametr do metody, která zajišťuje odeslání zpráv

klientovi. Příjem zprávy probíhá tak, že je nadeklarován buffer o velikosti 1024 bytů a do tohoto bufferu se uloží zpráva od serveru. Zpráva je následně dekódována do tvaru string.

2.3 Projekt HomeAutomationArduinoSketch

Projekt HomeAutomationArduinoSketch je naprogramován v prostředí Arduino IDE pro Arduino Uno desku. Sketch definuje konfiguraci pinů na desce, čtení dat z PIR senzoru a teplotního čidla DHT11 a komunikaci s Raspberry Pi po I^2C sběrnici. Sketch byl napsán v Arduino IDE programovacím jazykem C++. HomeAutomationArduinoSketch byl testován na reálném zařízení Arduino Uno (Rev3).

V úvodu sketche je vložena knihovna *DHT.h* pro teplotní čidlo a knihovna *Wire.h* potřebná pro I^2C sběrnici. Poté je nadefinována podřízená adresa pro dané Arduino zařízení a konkrétní typ teplotního čidla.

Následuje rezervace pinu 2 a 3 pro PIR senzor a teplotní čidlo. Globálním proměnným *Value_Temperature* a *Value_passiveIR* se přiřazují aktuální hodnoty čidel. Jelikož jsou nestálé, je k nim přiřazeno klíčové slovo *volatile.* Dále je zapotřebí nadefinovat proměnné protokolu typu byte a inicializovat DHT čidlo pomocí metody z knihovny *DHT.h*.

V bloku *void setup()* při spuštění programu je potřeba inicializovat využívané piny na desce, připojit se k I^2C sběrnici pomocí *Wire.begin()*, nastavení počátečních hodnot pro příjem přenosu od nadřízeného (*Wire.onReceive*) a vyvolat funkci k sestavení dat po požadavku nadřízeného (*Wire.onRequest*).

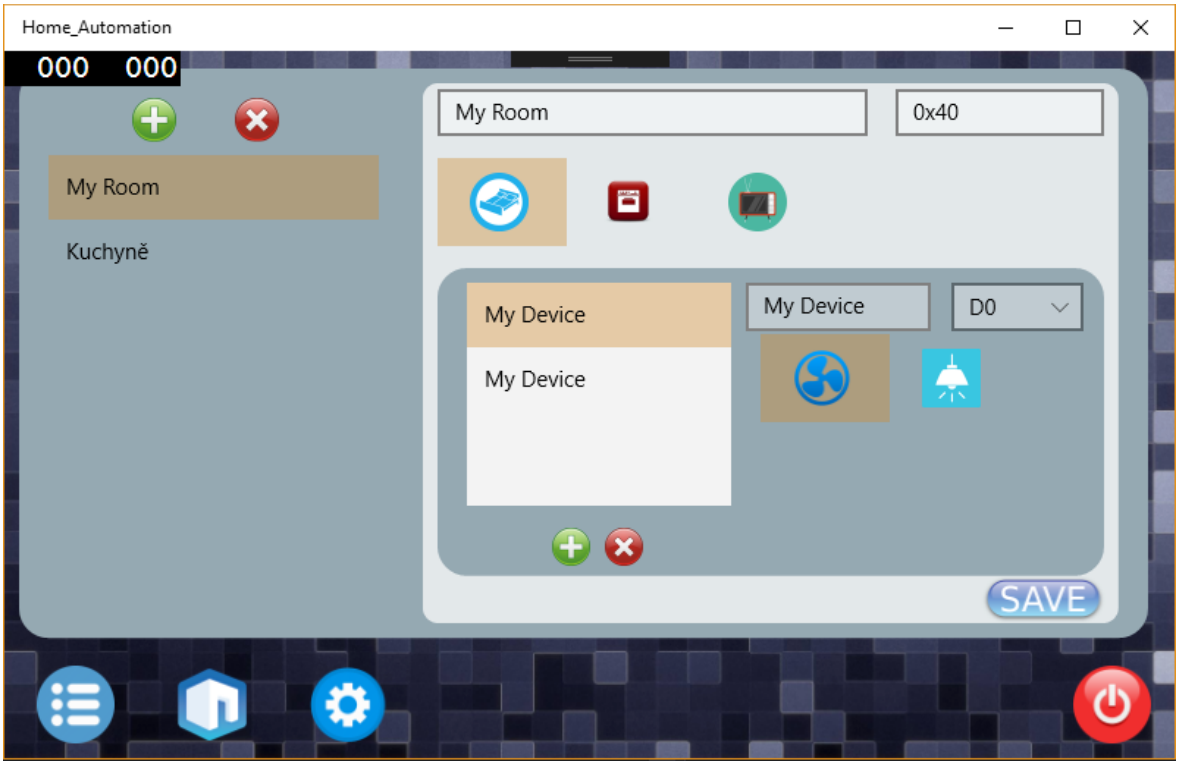
Druhý blok *void loop()* slouží pouze pro čtení dat z čidel. Z PIR čidla rozlišujeme pouze dvě hodnoty, a to true nebo false, zda je přítomen pohyb. U teplotního čidla registrujeme krátké celé číslo.

Funkce *ReceiveData()* přijme požadavek odesílaný nadřízeným danému podřízenému zařízení a zjistí hodnoty jednotlivých bytů. Pokud první byte reprezentuje režim 2, zapíše se hodnota (Value) k příslušnému pinu.

Po přijetí požadavku funkce *RequestData()* přes *switch* vybere příslušný režim, zjistí hodnoty čidel, nebo stav příslušných pinů, a poté odešle odpověď do Raspberry Pi pomocí *Wire.write()*.

2.4 Projekt Home_Automation_RPi

Home_Automation_Rpi je více komplexního charakteru než sketch pro Arduino. Úkolem tohoto projektu, který je nahrán do centrální jednotky Raspberry Pi, komunikovat s podřízeným zařízeními pomocí I^2C sběrnici, odesílat a zpracovávat data. Následně jsou informace z Arduino Uno, reprezentující jednu místnost, zobrazena ve vytvořeném grafickém uživatelském rozhraní. Projekt byl inspirován projekty z internetových stránek hackster.io [30] a testován na reálném zařízení Raspberry Pi 3.



Obr.7: Konfigurační stránka aplikace Home_Automation_RPi (vlastní)

V tomto uživatelském rozhraní lze odstraňovat, editovat a přidávat různá nastavení místností, zobrazovat aktuální teplotu, sledovat detekci pohybu a sepínat či rozepínat relé připojené například k ventilátoru. Uvedené informace, zda je sepnuto relé nebo aktuální teplota, se odesílají do příslušného mobilního zařízení. Jak již bylo zmíněno, aplikace pro hub je napsaná v prostředí Visual Studio 2015 Community jako Universal Windows

Project. V tomto projektu byl využit programovací jazyk C# pro data a XAML pro grafické uživatelské rozhraní.

2.4.1 Pomocné knihovny

Ve složce Library se nachází pomocné knihovny, kde jsou napsané potřebné funkce pro komunikaci I^2C a základní nastavení pro zařízení, senzory a místnosti. Složka se dále dělí na podsložky Communication a Core.

Communication

V této podsložce se nachází třída I2CHelp.cs. Jak již název napovídá, zde se definují byty požadavků pro režimy odesílané do podřízeného zařízení a inicializuje se I^2C sběrnice.

Core

Třída **Enviromental** v podsložce Sensors určuje základní vlastnosti PIR senzoru a teplotního čidla. Pro PIR senzor jsou nastaveny dva stavy: *None* a *HumanDetected*. Pro teplotní čidlo jsou zde na výběr tři jednotky teploty.

Ve třídě **Device** jsou vyměřeny konstanty pro jednotlivé piny Arduino zařízení, výčtový typ pro zjištění stavů zařízení a pinů. Dále se zde zjišťuje, zda zařízení je připojeno na dané adrese a jsou zde metody pro zjištění a změnu stavu pinů.

Třída **Room** již definuje konkrétní nastavení místnosti. Získává přístup k datům z předchozích tříd. Přidává nebo aktualizuje podřízené zařízení.

Třída **Home** převezme seznam místností i s aktuální konfigurací od třídy Room. Aktuální konfigurace místností se v této třídě ukládá do lokálního úložiště i načítá. Pokud není žádná uložená pozice, vrátí metoda *LoadHome()* prázdný objekt.

2.4.2 Grafické uživatelské rozhraní

Jednotlivé stránky grafického uživatelského rozhraní ve formátu xaml pro možnou správu systému v grafickém prostředí na obrazovce připojené k Raspberry Pi lze najít ve složce Pages. K těmto vzhledům je potřeba ještě počítat MainPage.xaml. Tato stránka je oddělená, protože byla vytvořena zároveň se založením projektu. K příslušnému xaml vzhledu patří podsoubor s příponou *.cs* pro naprogramování funkcí rozhraní.

MainPage je úvodní stránka po spuštění aplikace. Tato stránka pouze zobrazuje datum a čas a naviguje na stránky konfigurace a seznam místností. V MainPage.xaml.cs jsou metody k připojení podřízeného zařízení a ke kolekci dat ze senzorů. Ostatní stránky jsou koncipovány tak, že se otevřou v popředí této hlavní stránky, takže v pozadí je vidět úvodní obrázek plochy a dole je místo pro navigační tlačítka.

Stránka **ConfigurationPage** je velice důležitou částí projektu. Zde se nastavují místnosti, přiřazuje se k nim název, obrázek, adresa podřízeného zařízení. Dále se zde nachází konfigurace pro relé zařízení. Přiřadí se pin, na kterém se dané relé nachází, název, obrázek, a poté se uloží příkazem *Save*. Všechna tlačítka jsou v podobě obrázků, která jsou společně s tapetou plochy a ostatními obrázky uloženy ve složce Images.

Úlohou **RoomPage** je zobrazit všechny nastavené místnosti. Stránka je transparentní a zobrazuje se tam pouze *ListView* uložených místností s názvem a charakteristickým obrázkem.

Po kliknutí jedné z místností v RoomPage se zobrazí stránka **DevicePage.** Tato stránka se dělí na dvě části. První je *StackPanel*, kde se zobrazuje pomocí asynchronního vlákna detekce pohybu a teplota. Druhou částí je *ListView*, ve které jsou zobrazeny nastavení pro relé zařízení, které se nakonfigurovali v ConfigurationPage a zde se zavolali pomocí metody *LoadDevices()*. Pokud se na prvek v *ListView* klikne, tak pomocí metody *lstViewDevices_DoubleTapped()* se změní stav označeného relé.

2.4.3 Ikony

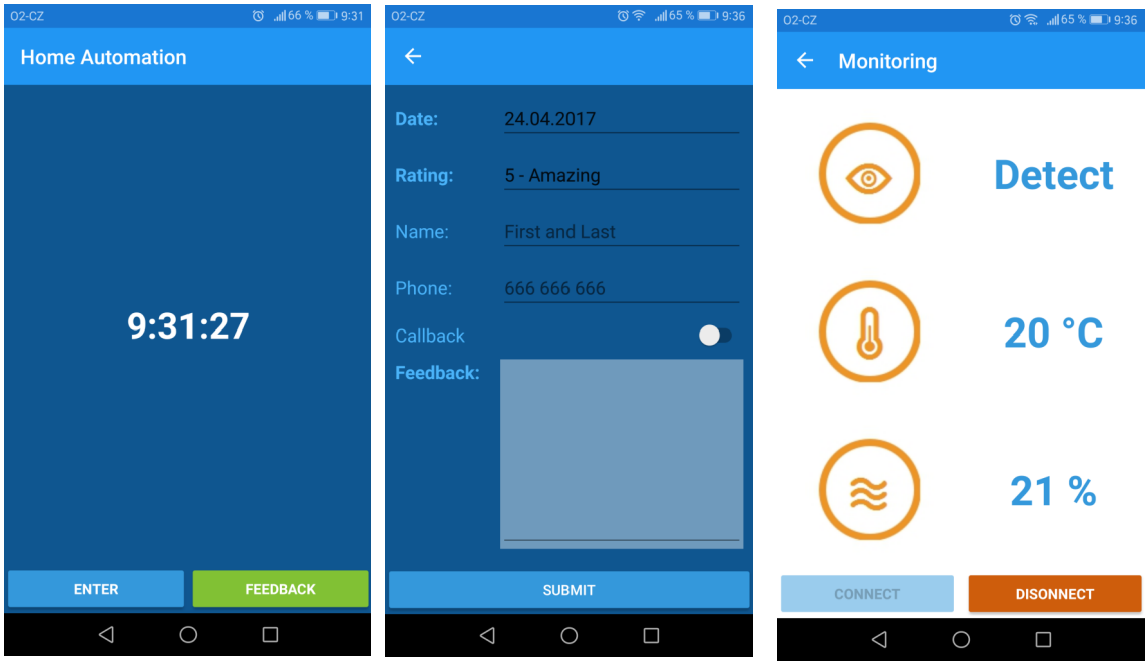
Ikony a jiné obrázky se nachází ve složce Images. V této složce se nachází obrázky reprezentující dané tlačítko, stav, či charakteristiku místnosti (ložnice, pokoj, kuchyně).

V podsložce Configuration jsou obrázky pro tlačítka přidat, odebrat, uložit. Podsložka Device obsahuje ikony různých domácích zařízení, které chceme pomocí relé modulu ovládat. Obrázky tlačítek v úvodní obrazovce se vyskytují v podsložce MainPage. V podsložce Room jsou ikony představující místnosti. StateDevice nám reprezentuje obrázky čidel. V poslední podsložce Wallpaper se nacházejí tapety na pozadí, které jsem zkoušel, aby aplikace vypadala příjemně po vizuální stránce.

2.5 Projekt HomeMobile

HomeMobile nám umožňuje přijímat a poté zobrazovat data z centrální jednotky do našeho mobilního zařízení. Aplikace, z pohledu běžného uživatele, se skládá ze tří hlavních stránek. Úvodní stránka zobrazuje aktuální čas a slouží k navigaci. Další stránkou je zpětná vazba, kde uživatel může vyplnit a odeslat údaje s připomínkami autorovi. Hlavní stránka slouží k zobrazování hodnot z čidel v reálném čase. Projekt byl napsán ve vývojovém prostředí Visual Studio 2015 Community programovacím jazykem C# a XAML. Aplikace Home Mobile byla testována na reálném zařízení Honor 8 32GB Dual Sim.

V projektu jsem se řídil návrhovým vzorem Model-View-ViewModel. Díky tomuto modelu jsem separoval datovou vrstvu od grafické mezivrstvou ViewModel, a zajistila se nezávislost View-Model. K záměru uskutečnění návrhového vzoru MVVM jsem využil knihovnu třetí strany *MvvmHelpers*. Dostává se tak určité čistoty a dobré přehlednosti v kódu.



Obr.8: Stránky aplikace HomeMobile pro Android (vlastní)

Xamarin.Forms neobsahuje některé funkce, které jsou zapotřebí v tomto projektu (TCP komunikace, MVVM, Email), proto je zapotřebí nainstalovat knihovny třetí strany. Hlavní výhodou využití je udržení sdíleného kódu – nemusí se psát specifický kód pro každou platformou zvlášť. Nevýhodou knihoven třetí stran je nedůvěryhodnost zdroje (je třeba zjišťovat, zda knihovna nezpůsobí problémy) a zvýšení velikosti projektu.

Abychom dosáhli co největší přenositelnosti, použil jsem Xamarin PCL (Portable Class Library), přenositelnou verzi. Celý projekt se rozděluje do čtyř podprojektů: HomeMobile (Portable), HomeMobile.Droid, HomeMobile.iOS, HomeMobile.UWP.

Projekty, v jejichž názvu je přívlastek určité platformy, obsahují důležité knihovny, které zajišťují překlad do nativního prostředí dané platformy. Pokud je potřeba napsat specifický kód pro každou platformu zvlášť, tak se tento kód objeví právě v těchto podprojektech. Dále pak předávají řízení aplikace ve svých hlavních třídách přenositelnému podprojektu HomeMobile (Portable). Do specifických projektů je třeba ukládat i statické soubory, jako jsou obrázky nebo ikony, které jsou použity v aplikaci.

Projekt HomeMobile (Portable) obsahuje kromě datového modelu i uživatelské rozhraní, aplikační vrstvu a business logiku. Hlavní třída Portable projektu se nazývá

App.xaml.cs, na kterou odkazují projekty daných platforem při startu aplikace. Tato třída poté přebírá chod celé aplikace a definuje základní složky.

2.5.1 Grafické uživatelské rozhraní

Celá vrstva grafického uživatelského rozhraní se nachází ve složce Pages. Všechny vzhledy jsou typu *ContentPage*. Kromě úvodní obrazovky je do ostatních stránek vnořen *StackLayout* a ve FeedbackPage byl vložen *ScrollView*. Pro umístění jednotlivých prvků byla vytvořena mříž ze sloupců a řádků pomocí *Grid*. Každý prvek má pak jasně danou pozici.

Úvodní stránka **HomePage.xaml** obsahuje pouze aktuální čas, který je zobrazován pomocí textu (label) a *BindingContext* vychází z ViewModel. Dále se zde nachází dvě tlačítka, která pouze odkazují na další stránky. Metody pro tato tlačítka se nachází v podsouboru HomePage.xaml.cs.

Ve **FeedbackPage.xaml** se nachází formulář, který uživatel vyplní a po následném stisknutí tlačítka *Submit* jej odešle na předem určený email. Tento formulář se skládá ze dvou jednořádkových textových vstupů (Entry) pro jméno a telefonní číslo, výběru data (DatePicker) pro zadání času, klasického výběru určených možností (Picker) k ohodnocení aplikace, přepínače (Switch), zda bude chtít uživatel odpověď od vývojáře a z víceřádkového textového vstupu (Editor). Příslušné elementy jsou popsané texty. Dále je zde *ActivityIndicator*, který se aktivuje po stisknutí tlačítka *Submit*. Veškeré akce (binding) jsou v mezivrstvě viewModel. Vytvoření instance viewModelu se provádí v podsouboru FeedbackPage.xaml.cs.

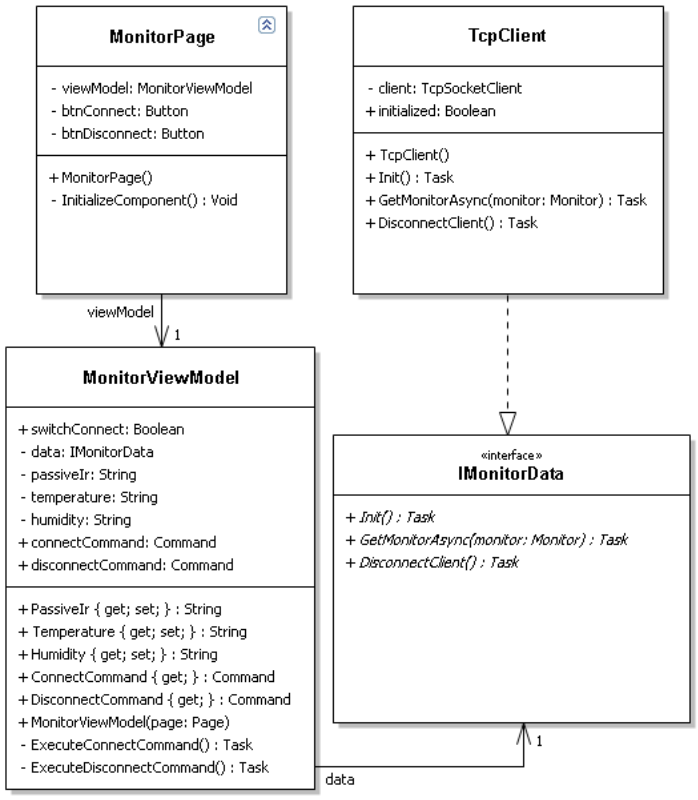
MonitorPage.xaml je hlavní stránkou celého projektu. Oproti FeedbackPage je tato stránka chudší. Obsahuje pouze ikony pro identifikaci čidla a ke každé ikoně textový výstup pro vizualizaci hodnoty čidla. Tlačítko *Connect* připojuje zařízení k TCP serveru. Tlačítko *Disconnect* poté odpojuje TCP připojení. Jako u předchozí stránky, tak i u této se všechny akce nastavují ve viewModelu, na který se odkazuje v MonitorPage.xaml.cs.

2.5.2 Mezivrstva ViewModel

Složka ViewModel obsahuje všechny mezivrstvy pro stránky, jež byly popsány v kapitole 2.4.1. V těchto třídách se nastavují veškeré akce pro stisk tlačítka, změnu parametrů textového výstupu a textových vstupů. Základní třídou je **ViewModelBase.cs**. Tato třída dědí z knihovny *MvvmHelpers*, a tím definuje podstatu Model-View-ViewModel. Z ViewModelBase.cs dědí ostatní ViewModel třídy (kromě HomeViewModel), a lze tak nastavit titul/podtitul stránky, *isBusy*, ikonu a další prvky přímo v těchto třídách. Díky této skutečnosti nemusí programátor akce (binding) programovat ve skrytých podtřídách v xaml, ale má tyto funkce v jasně dané složce. Tím se zvyšuje přehlednost práce.

HomeViewModel.cs nedědí z ViewModelBase, protože je zde pouze zobrazení aktuálního času. *StartTimer* definuje sekundový interval pro nastavení *DateTime*. V setteru je pak nastaven event, který *DateTime* mění.

Aby šla data z formuláře předat dál do třídy, která zajistí odesílání a přijímání dat, byla využita funkce **DependencyService**. *DependencyService* se také využívá, pokud je potřeba napsat specifický kód pro každou platformu zvlášť. Pro použití této služby je potřeba vytvořit rozhraní, implementovat metodu z rozhraní do cílové třídy a získat přístup k této metodě pomocí *IFeedbackData data = DependencyService.Get<IFeedbackData>()*.



Obr.9: Znázornění komunikace tříd pomocí služby DependencyService (vlastní)

Ve **FeedbackViewModel.cs** se event při stisku tlačítka provádí pomocí *Xamarin.Forms.Command*. *Command* vyvolá metodu *ExecuteSaveFeedbackCommand()*, zde se aktivuje *ActivityIndicator* a v metodě *data.AddFeedbackAsync()*, jejíž parametrem je nová instance *Feedback* z modelové vrstvy, se nastaví hodnoty, které se odešlou na email. Všechny údaje, které uživatel nastavuje na stránce, se ukládají přes akci(binding) do regionu *Properties* v této třídě.

MonitorViewModel.cs má strukturu podobnou jako FeedbackViewModel.cs. Hodnoty z čidel se ukládají do regionu *Properties*. Data se přijímají přes *DependencyService*.

2.5.3 Vrstva Model

Datová vrstva se nachází ve složce Model, kde jsou třídy, které představují datovou část aplikace. První třída se jmenuje **Feedback.cs**. Druhá třída nese název **Monitor.cs**. Názvy tříd jsou logicky odvozeny od názvů stránek, ke kterým jsou přiděleny. V našem případě se jedná pouze o nastavení, ve kterých se uloží hodnoty z modelView přes *DependencyService*. Ve třídě Feedback.cs je navíc definovaný formát zprávy.

2.5.4 Rozhraní pro službu DependencyService

V třídách pod složkou Interface jsou rozhraní, v nichž se nachází *Task* metody využívající služba *DependencyService*. Parametrem těchto metod jsou nastavení datových tříd. Podstata těchto rozhraní byla vysvětlena v kapitole 2.4.2.

2.5.5 Komunikační služby

Komunikační služby, které se starají o odesílání či přijímání dat, se nachází ve složce Services. Třídy dědí metody z rozhraní, díky čemu se zajistí správná funkce služby *DependencyService*.

Třída **EmailFeedback.cs** obsahuje pouze jednu metodu. Metoda *AddFeedbackAsync* zajišťuje odeslání dat z vyplněného formuláře na stránce zpětné vazby. Odeslání dat na určený email zajišťuje *emailTask.SendEmail()*. Pro tyto instrukce byla použita knihovna třetí strany Plugin.Messaging.

Oproti předchozí třídě se v třídě **TcpClient.cs** nachází více metod. První metoda inicializuje TCP komunikace, v další metodě se nachází kód pro přijímání zpráv. Poslední metoda ukončuje spojení klienta se serverem.

2.6 Podíl sdíleného kódu v projektu HomeMobile

V projektu HomeMobile byl kladen důraz, aby co největší množství kódu bylo sdíleného a nebylo potřeba psát specifický kód pro každou platformu zvlášť. Framework Xamarin umožnil implementovat aplikační a datovou vrstvu do sdíleného kódu, díky čemuž se nemusel psát specifický kód pro jednotlivé platformy.

Grafické uživatelské rozhraní je též definováno v přenositelném kódu, ale je pak renderováno pomocí nativních komponent, což způsobuje v jednotlivých platformách specifický vzhled, jenž na pohled nevypadá uživatelsky příjemně. Pro uživatelsky přátelský vzhled aplikace se parametry grafického uživatelského rozhraní nastavují v projektu konkrétní platformy.

Do projektů jednotlivých operačních systému se pouze musely vložit obrázky ve formátu jpg. V těchto projektech bylo potřeba nainstalovat knihovny, které byly použité

v přenositelného projektu. Pro správnou funkčnost TCP komunikace se nastavovala povolení Internet (Client & Server).

2.7 Komplikace

Při programování je vždy jisté, že programátor se dostane do situace, kdy nastane nějaká komplikace v podobě chyby v programu. Tyto chyby se musí odstranit drobnou úpravou v lepším případě, nebo v horším případě přepsat značnou část kódu. V průběhu vytváření aplikace je hledání a následné opravování chyb nejvíce časově náročné. Řešení většiny problémů se dá najít na webových stránkách stackoverflow.com [31] a forums.xamarin.com [32].

Nejčastější chybou v tomto projektu byla syntaktická chyba. Syntaktickou chybu překladač rozezná, zobrazí její umístění a zamezí aplikaci dalšímu běhu. Následně se kompilace zastaví a programátor ji jednoduše opraví.

Dalším častým problémem jsou běhové chyby. Běhové chyby se nezobrazí při překladu kódu, ale objeví se až po nějaké uživatelské akci. Typickou běhovou chybou byl *NullReferenceException*. Tato chyba proběhla v případě, kdy například v aplikaci volá nějaká metoda prázdný odkaz.

Komplikace nastaly i v počáteční fázi programování, tj. při zakládání projektů a následné konfiguraci. Založený projekt musí mít dostatečně krátkou cestu a bez diakritiky, jinak dojde k problémům kompilace. Když je projekt založený, musí se programátor ujistit, zda má správnou cestu pro SDK a JDK Android. Kontrolu lze provést v nastavení Visual Studio. V Xamarin projektu pro Android dochází často k nenalezení *Android.Support* knihoven. Tyto knihovny se pak musí přeinstalovat. Pokud se projekt kopíruje do jiného počítače, musí se všechny podprojekty znovu načíst (reload project).

Jinou komplikací může být emulátor. Emulátory jsou obecně pomalé, zatěžují paměť i procesor a mnohdy nastanou nepředvídatelné situace. V této práci byly použity dva emulátory. Xamarin Android Player je poměrně rychlý emulátor, který se spouští v programu VirtualBox. Emulátor ovšem nečekaně přestal fungovat. Při zapnutí emulátoru

systém Windows zobrazil chybové hlášení *System Service Exceptiom* na modré obrazovce. Druhý emulátor byl základní z Android Software Development Kit. Tento emulátor byl, oproti Xamarin emulátoru, výrazně pomalejší.

3 Porovnání multiplatformních nástrojů

V této kapitole jsou porovnány tři multiplatformní nástroje. Jedná se o Xamarin, Appcelerator a Gluon. Pro srovnání byly vybrány projekty, které slouží jako vzorové a jsou volně ke stažení na oficiálních stránkách jednotlivých platforem. Porovnávat se bude instalace nástrojů, struktura kódu, podpora produktu. Obsah projektu v tomto případě není podstatný. Všechny projekty budou zaměřeny na mobilní operační systém Android.

Projekt, jenž je vytvořen nástrojem Xamarin, se nazývá HomeMobile. Projekt byl vytvořen pro tuto bakalářskou práci a jeho popis je zmíněn v kapitole 2.4. Jako ukázkový projekt pro nástroj Appcelerator Titanium byl zvolen Employee Directory [33], který byl vytvořen na ukázku firemního adresáře. Další ukázkový projekt [34] je pro multiplatformní nástroj Gluon Mobile. Aplikace tohoto projektu zobrazuje padesát států. Každý stát je pak doplněn o informace polohy, rozlohy atd.

3.1 Instalace multiplatformních nástrojů

Instalace multiplatformních nástrojů je velice jednoduchá a intuitivní. Na oficiálních stránkách jsou odkazy, kde jsou poté instrukce krok po kroku. Nejdříve je ovšem nutné vědět, v jakém prostředí bude jednotlivý multiplatformní nástroj provozován. Xamarin je použit ve vývojovém prostředí Visual Studio Community 2015. Nástroj Gluon je nainstalován ve vývojovém prostředí IntelliJ IDEA. Jediný multiplatformní nástroj Appcelerator Titanium byl použit ve svém vlastním vývojovém prostředí Appcelerator Studio, protože nelze jej nainstalovat jako doplněk v žádném známém vývojovém prostředí.

Doplněk Xamarin nástroje pro Visual Studio se nainstaluje tak, že se spustí instalátor vývojového prostředí, který je ke stažení na oficiálních stránkách Microsoftu. Po spuštění je poté potřeba zvolit sadu funkcí multiplatformního vývoje. Důležitou položkou je Android Native Development Kit (NDK), Android Software Development Kit (SDK), a Java development Kit (JDK). Tyto sady jsou nedílnou součástí vývoje pro Android. Sady lze zkontrolovat ve vývojovém prostředí pod cestou *Tools→Options...→Xamarin*.

Instalace nástroje Gluon neprobíhá přes instalátor, ale stačí jej doinstalovat pouze jako doplněk. Při spuštění vývojového prostředí IntelliJ IDEA se zobrazí úvodní okno, poté se klikne na záložku Configure, kde je složka Plugins. Tato složka se otevře, do vyhledávače se zadá Gluon Plugin. Po vyhledání jej uživatel nainstaluje.

Appcelerator Titanium má své vlastní vývojové prostředí. Pro získání zkušební licence (nemožnost dávat aplikace do obchodů aplikací) a Appcelerator Studio se musí uživatel zaregistrovat na oficiálních stránkách multiplatformního nástroje. Po registraci a přihlášení v nabídce stránky najdeme Appcelerator Studio, vybereme si verzi (Windows, Mac, Linux) a stáhneme instalátor. Uživatel pak nainstaluje studio podle instrukcí.

3.2 Vytvoření nového projektu

Xamarin umožňuje vytvořit aplikaci hned několika způsoby, proto programátor musí vědět již při zakládání projektu, jakým způsobem bude aplikaci vyvíjet. Nejdříve si musí vybrat, zda projekt bude sdílený (Share) nebo přenositelný (Portable). Rozdíl mezi sdíleným a přenositelným projektem je vysvětlen v kapitole 1.3.7. Další možností v nabídce tvorby nových projektů je způsob vytváření grafického uživatelského rozhraní. Programátor si může vybrat, jestli chce programovat pomocí značkovacího jazyka XAML, nebo programově v kódu za pomocí volání rozhraní pro programování (API - Application Programming Interface). Po výběru postupu v projektu programátor zadá jméno projektu, název celého souboru projektů (Solution), určí lokaci na disku a vytvoří projekt.

Appcelerator Titanium a jeho studio není tak bohatý na možnosti vývoje, jako to je u multiplatformního nástroje Xamarin. Při založení Appcelerator projektu si může programátor zvolit, jakou strukturu bude mít hlavní stránka aplikace (tabbed, master-detail, single). Po zvolení vzhledu aplikace programátor zvolí jméno projektu, jedinečné id aplikace a cílovou platformu. Po zadání všech kritérií se vytvoří nový projekt.

Založení projektu nástroje Gluon Mobile je obdobné jako u Appcelerator Titanium. Při založení projektu má programátor na výběr, zda chce jednotný nebo vícenásobný vzhled. U vícenásobného pohledu může navíc využít značkovací jazyk FXML odvozený od jazyka XML. Po výběru způsobu vývoje se zobrazí okno pro název hlavní třídy, název

balíku (shodný s id aplikace u Appcelerator), výběru cílových platforem. Nakonec se vybere verze SDK, určí se název projektu a lokace projektu.

3.3 Struktura projektu

Struktura projektu pro Xamarin je přehledná. Na začátku projektu se vytvoří několik projektů, jejichž název napovídá, který projekt na co slouží. V projektech s příponou určité platformy se píší specifické kódy pro danou platformu. Projekt, který nese pouze samotný název a v závorce je definice způsobu vývoje (Portable nebo Share), má na starosti multiplatformní část aplikace. Ve všech projektech jsou potom soubory References a Properties. V souboru References jsou nainstalované všechny použité knihovny. V Properties se nastavují možnosti projektu (minimální verze systému, povolení internetu, wifi apod.). Projekt HomeMobile je koncipován tak, že jednotlivé třídy jsou zařazeny do složek, jejichž názvy definují účel těchto tříd. Například ve složce Pages jsou všechny třídy, ve kterých je napsán kód pro grafické uživatelské rozhraní. Pomocí složek dostává programátor přehlednou strukturu projektu.

Po vytvoření projektu nástrojem Appcelerator Titanium se vytvoří složka app, kde se píše kód pro aplikace. Ve složce plugins jsou knihovny, které byly využity. Dále se zde nachází licenční soubory, výchozí ikona a důležitý soubor tiapp.xml, ve kterém je nastavení celého projektu (název, verze, cílové platformy). V projektu Employee Directory jsou třídy také rozděleny (podobně jako to je u Xamarin) do jednotlivých podsložek ve složce app. Pokud programátor ovšem chce napsat specifický kód pro danou platformu, vytvoří složku s názvem platformy do podsložky, která reprezentuje daný účel, a do ní udělá třídu pro určitou platformu.

V projektu Fifty States pro multiplatformní nástroj Gluon Mobile se veškerý kód píše ve složce src. V této složce jsou podsložky pro specifické kódy daných platforem a podsložka main, v které se píše multiplatformní kód. Srovnání struktur jednotlivých projektů se nachází v příloze G.

3.4 Podpora multiplatformních nástrojů

Komunita a její podpora multiplatformního vývoje je velice důležitá pro volbu nástroje, který chce programátor využívat. Každý nástroj by měl mít kvalitní a přehlednou

dokumentaci svých knihoven. Pokud potřebuje programátor poradit s nějakým postupem, musí framework disponovat kvalitní základnou uživatelů, kteří jsou ochotni poradit, ať už osobní korespondencí nebo na oficiálním či neoficiálním fóru.

Xamarin na svých stránkách udává, že jejich nástroje využívá nebo využilo [35] přes jeden milion čtyři sta tisíc vývojářů. Programátoři ke svým otázkám využívají oficiálního fóra Xamarinu [32] a známé fórum Stackoverflow [31]. V polovině května roku 2017 bylo na oficiálním fóru na téma Xamarin.Forms přes dvacet šest tisíc otázek, z toho nezodpovězených bylo přibližně dvacet procent [36]. Na Stackoverflow bylo v tom samém období necelých osm tisíc otázek, přičemž zhruba čtyřicet procent není zodpovězených [37].

Appcelerator na oficiálních stránkách píše, že jej používá přes devět set sedmdesát tisíc vývojářů [38]. Appcelerator nemá své oficiální fórum, proto vývojáři tohoto nástroje vkládají otázky na Stackoverflow. V květnu roku 2017 na tomto fóru bylo zaevidováno tři tisíce dvě stě otázek. Vláken bez odpovědí je v tomto případě přibližně čtyřicet procent [38].

Gluon Mobile neuvádí, kolik vývojářů jejich služeb využívá. Své vlastní fórum také nemá. Lidé své dotazy ke Gluon Mobile pokládají na Stackoverflow. Ve stejném období, jako u předchozích nástrojů, bylo na Stackoverflow pouze dvě stě padesát sedm vláken týkajících se frameworku Gluon. Nezodpovězených dotazů bylo zhruba padesát pět procent [40].

Závěr

Cílem této bakalářské práce bylo zanalyzovat možnosti multiplatformního vývoje a vytvořit testovací aplikaci, která bude využívat hardwarová rozhraní pro komunikaci s periferiemi, jako jsou například čidla a centrální jednotka. Nejdříve byly popsány tři nejpoužívanější mobilní platformy v dnešní době. Dále byly představeny multiplatformní nástroje, které cílí na zmiňované mobilní operační systémy. Další kapitola byla věnována praktické části, kde byl rozebrán vývoj testovací aplikace. Poslední kapitola byla věnována základnímu srovnání tří multiplatformních nástrojů.

Pro tvorbu testovací aplikace byl vybrán nástroj Xamarin. Hlavními důvody byly znalosti programovacího jazyka C# autora této práce a prozkoumání multiplatformních možností společnosti Microsoft. Mobilní aplikace komunikuje pomocí TCP komunikace s centrální jednotkou Raspberry Pi 3 (nadřízená část) a sbírá z ní data o teplotě, vlhkosti a stavu pasivního infračerveného senzoru, jež jsou připojeny k Arduino Uno (podřízená část). Dokázalo se, že celý kód mobilní aplikace, napsaný autorem bakalářské práce, je sdílený a nemusel se psát specifický kód pro jednotlivé platformy zvlášť. Dále se ověřovala funkcionalita na jednotlivých mobilních operačních systémech. Úspěšně byla aplikace otestována na platformě Android, Windows 10 Mobile a Windows 10 Desktop (viz. příloha D). Aplikace nemohla být spuštěna na operačním systému iOS, protože nebyly prostředky k samotnému testování.

Po srovnání tří multiplatformních nástrojů v poslední kapitole a celkové práci na frameworku Xamarin autor bakalářské práce usoudil, že Xamarin je velice dobrou cestu pro vývoj multiplatformních aplikací nejen kvůli veliké komunitě, ale také podpory nadnárodní společnosti Microsoft, která koupila společnost Xamarin v roce 2016. Tento výsledek je ovšem subjektivní, protože autor této práce více inklinuje k programovacímu jazyku C# než k jazyku JAVA.

V budoucnu autor práce plánuje tento projekt rozšířit. Všechny aplikace jsou ve fázi testování, proto bude potřeba je upravit pro potřeby běžného uživatele. Centrální jednotka i mobilní aplikace rozezná více než jednu připojenou podřízenou část, jenž představuje jednu místnost. Centrální jednotka nebude připojena k podřízenému zařízení drátově,

ale komunikace bude probíhat bezdrátově. Mobilní aplikace bude schopna ovládat relé, které jsou připojené k podřízenému zařízení. V plánu je také propojení tohoto projektu s cloudovou službou Microsoft Azure.

Všechny kódy, videa, postupy, komunikace a řešení problémů s vedoucím bakalářské práce jsou na katedrálním git repozitáři [41].

Seznam literatury a informačních zdrojů

[1] IDC RESEARCH, INC., *Smartphone OS Market Share, 2016 Q2* [online]. 2016 [cit. 2016-11-21]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[2] WINDOWS MOBILE MANIA, *Celkové shrnutí současného dění a blízká budoucnost Windows 10 Mobile – máme se na co těšit?* [online]. 2016 [cit. 2016-9-21]. Dostupné z: https://wmmania.cz/clanek/celkove-shrnuti-soucasneho-deni-a-blizka-budoucnost-windows-10-mobile-mame-se-na-co-tesit/

[3] ANDROID, *Android* [online]. 2017 [cit. 201-04-20]. Dostupné z: https://www.android.com/

[4] ELGIN, B., *Google Buys Android for Its Mobile Arsenal* [online]. 2005 [cit. 2016-10-20]. Dostupné z: http://webcitation.org/5wk7sIvVb

[5] GOOGLE INC., *ART and Dalvik* [online]. 2016 [cit. 2016-10-21]. Dostupné z: https://source.android.com/devices/tech/dalvik/index.html

[6] WIKIPEDIA, *Android (operační systém)* [online]. 2016 [cit. 2016-01-20]. Dostupné z: https://cs.wikipedia.org/wiki/Android_%28opera%C4%8Dn%C3%AD_syst%C3%A9m%29

[7] GEORGIEV, B. A., SILLITTI, A., SUCCI, G. *Open Source Mobile Virtual Machines: An Energy Assessment of Dalvik vs. ART* [online]. 2014 [cit. 2016-10-21]. Dostupné z: http://link.springer.com/chapter/10.1007%2F978-3-642-55128-4_12

[8] LEIVA, Antonio. *MVP for Android: how to organize the presentation layer* [online]. 2014 [cit. 2016-10-21]. Dostupné z: http://antonioleiva.com/mvp-android/

[9] GOOGLE INC. *Get Started with Publishing* [online]. 2015 [cit. 2016-10-21]. Dostupné z: http://developer.android.com/distribute/googleplay/start.html

[10] APPLE INC. *About the iOS Technologies* [online]. 2014 [cit. 2016-10-26]. Dostupné z: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html

[11] APPLE INC. *Model-View-Controller* [online]. 2014 [cit. 2016-10-26]. Dostupné z: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html#//apple_ref/doc/uid/TP40008195-CH32-SW1

[12] APPLE INC. *Choosing a Membership* [online]. 2016 [cit. 2016-10-26]. Dostupné z: https://developer.apple.com/support/compare-memberships/

[13] MICROSOFT. *Panorama control for Windows Phone 8* [online]. 2015 [cit. 2016-10-28]. Dostupné z: https://msdn.microsoft.com/library/windows/apps/ff941104

[14] MICROSOFT. *Using the Model-View-ViewModel Pattern* [online]. 2012 [cit. 2016-10-28]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh821028.aspx

[15] MICROSOFT. *Windows Store Policies* [online]. 2016 [cit. 2017-04-18]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/dn764944.aspx

[16] CORDOVA. *Overview* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://cordova.apache.org/docs/en/5.1.1/guide/overview/

[17] ZDROJÁK.CZ. *Vyvíjíme hybridní aplikace v Ionicu: Úvod a instalace* [online]. 2015 [cit. 2017-04-18]. Dostupné z: https://www.zdrojak.cz/clanky/vyvijime-hybridni-aplikace-v-ionicu/

[18] JQUERY MOBILE. About [online]. 2017 [cit. 2017-04-18]. Dostupné z: http://jquerymobile.com/about/

[19] VÁVRŮ, Jiří. *JQuery Mobile*. Brno: Computer Press, 2013. ISBN 978-80-251-3811-3.
[20] IONIC. *Docs* [online]. 2017 [cit. 2017-04-18]. Dostupné z: http://ionicframework.com/docs/
[21] KNOCKOUTJS. *Introduction* [online]. 2017 [cit. 2017-04-18]. Dostupné z: http://knockoutjs.com/documentation/introduction.html
[22] GLUON. *Gluon Mobile Documentation* [online]. 2017 [cit. 2017-04-18]. Dostupné z: http://docs.gluonhq.com/charm/4.3.2/
[23] APPCELERATOR. *Overview* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://docs.appcelerator.com/platform/latest/#!/guide/Appcelerator_Platform
[24] XAMARIN, Inc. *Introduction to Xamarin* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/#Introduction_to_Xamarin
[25] AOL, Inc. *Microsoft Is Buying Mobile Cross-Platform Development Company Xamarin* [online]. 2016 [cit. 2017-04-18]. Dostupné z: https://techcrunch.com/2016/02/24/microsoft-is-buying-mobile-cross-platform-development-company-xamarin/
[26] XAMARIN, Inc. *Part 1 – Understanding the Xamarin Mobile Platform* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_1_-_understanding_the_xamarin_mobile_platform/
[27] XAMARIN, Inc. *Application fundamentals* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://developer.xamarin.com/guides/cross-platform/application_fundamentals/
[28] HW SERVER, s. r.o. *Stručný popis sběrnice I2C a její praktické využití k připojení externí eeprom 24LC256 k mikrokontroléru PIC16F877* [online]. 2017 [cit. 2017-01-22]. Dostupné z: http://vyvoj.hw.cz/navrh-obvodu/strucny-popis-sbernice-i2c-a-jeji-prakticke-vyuziti-k-pripojeni-externi-eeprom-24lc256
[29] WIKIPEDIA, *TCP/IP* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/TCP/IP
[30] HACKSTER, Inc. *hackster.io* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://www.hackster.io/
[31] STACK EXCHANGE, Inc. *stack overflow* [online]. 2017 [cit. 2017-04-21]. Dostupné z: http://stackoverflow.com/
[32] XAMARIN, Inc. *Xamarin Forums* [online]. 2017 [cit. 2017-04-21]. Dostupné z: https://forums.xamarin.com/
[33] AXWAY, Appcelerator. *Sample App Walkthrough: Corporate Directory App* [online]. 2017 [cit. 2017-05-10]. Dostupné z: http://www.appcelerator.com/blog/2016/03/sample-app-walkthrough-corporate-directory-app/
[34] GLUON, *The 50 States App* [online]. 2017 [cit. 2017-05-10]. Dostupné z: http://docs.gluonhq.com/samples/fiftystates/
[35] XAMARIN, Inc. *The Xamarin Story* [online]. 2017 [cit. 2017-05-14]. Dostupné z: https://www.xamarin.com/about
[36] XAMARIN, Inc. *Xamarin.Forms* [online]. 2017 [cit. 2017-05-14]. Dostupné z: https://forums.xamarin.com/categories/xamarin-forms
[37] STACK EXCHANGE, Inc. *stack overflow* [online]. 2017 [cit. 2017-05-14]. Dostupné z: https://stackoverflow.com/questions/tagged/xamarin.forms

[38] AXWAY, Appcelerator. *Mobile App Development* [online]. 2017 [cit. 2017-05-14]. Dostupné z: http://www.appcelerator.com/

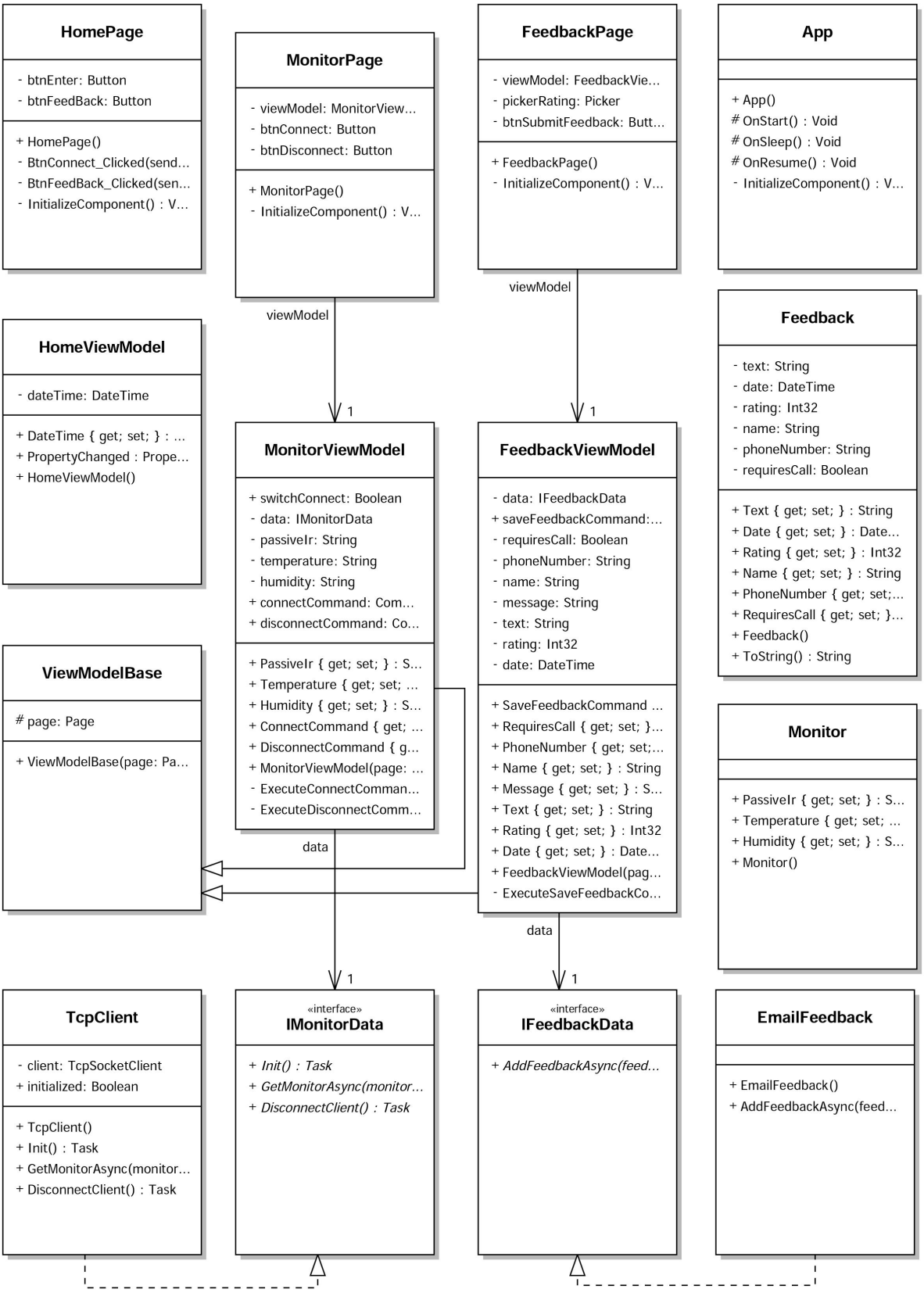
[39] STACK EXCHANGE, Inc. *stack overflow* [online]. 2017 [cit. 2017-05-14]. Dostupné z: https://stackoverflow.com/questions/tagged/appcelerator

[40] STACK EXCHANGE, Inc. *stack overflow* [online]. 2017 [cit. 2017-05-14]. Dostupné z: http://www.stackoverflow.com/questions/tagged/gluon

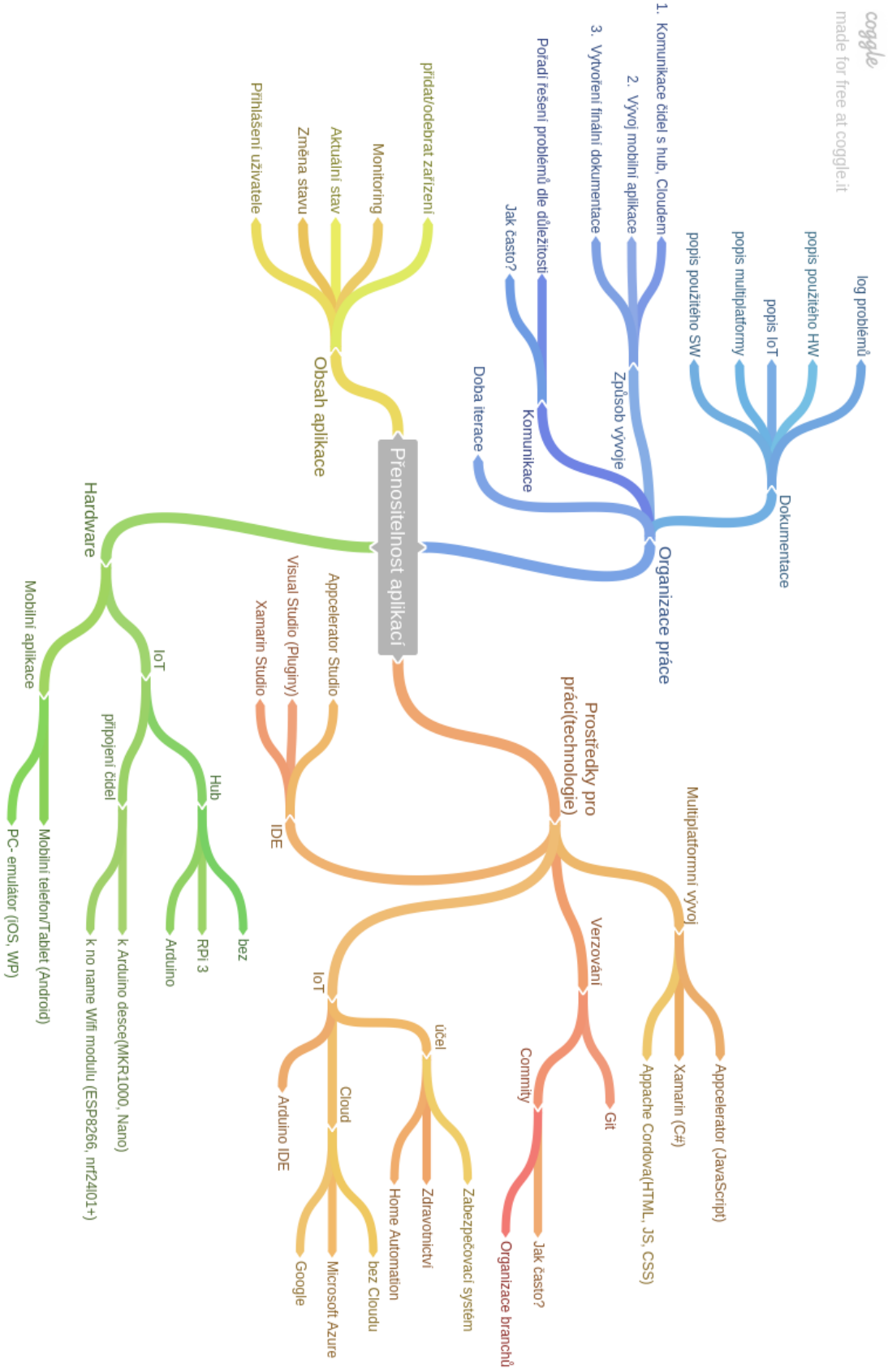
[41] KTE FEL. *GitLab* [online]. 2017 [cit. 2017-05-16]. Dostupné z: https://edison.fel.zcu.cz:444/

Přílohy

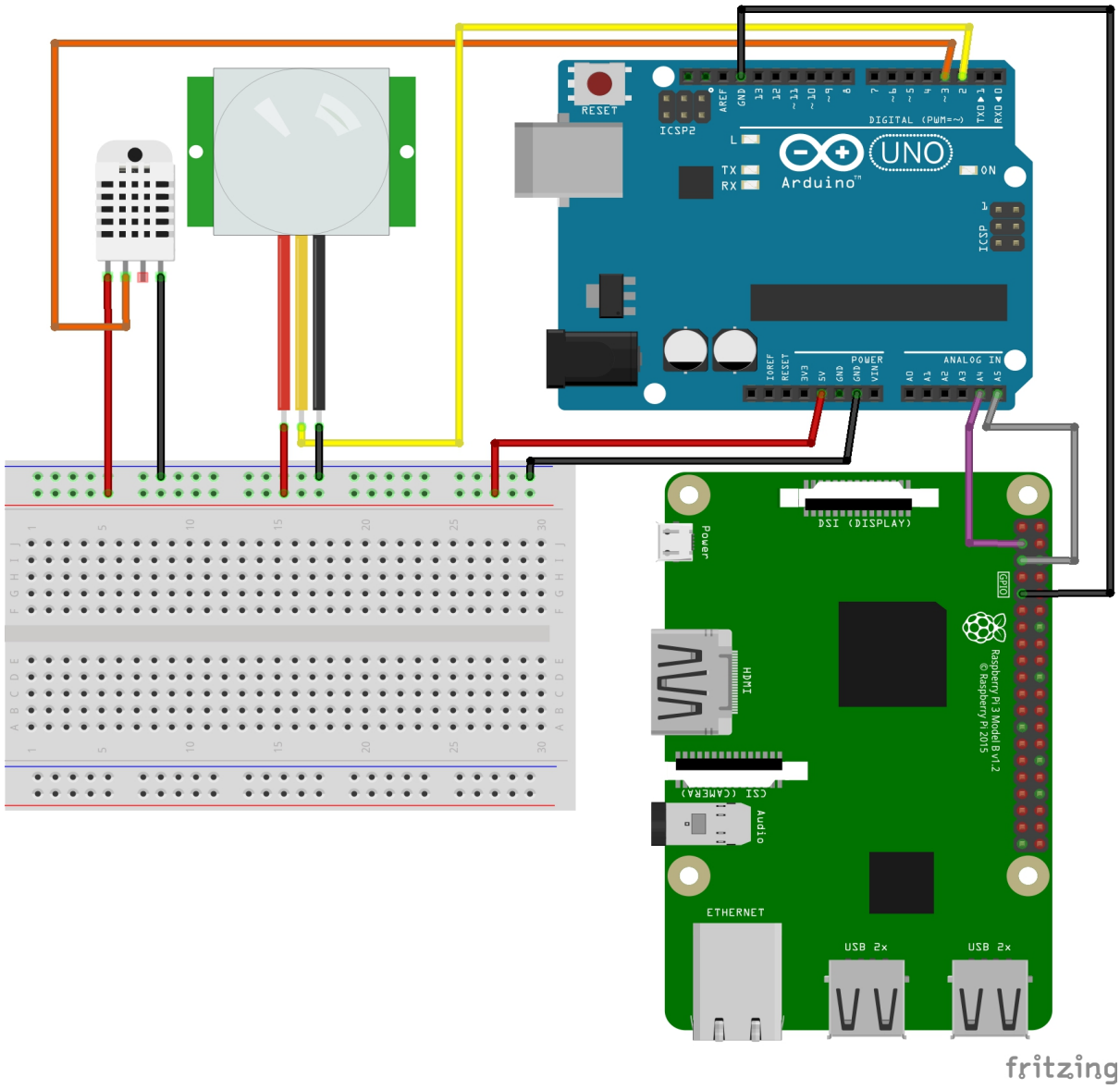
Příloha A – UML diagram projektu HomeMobile



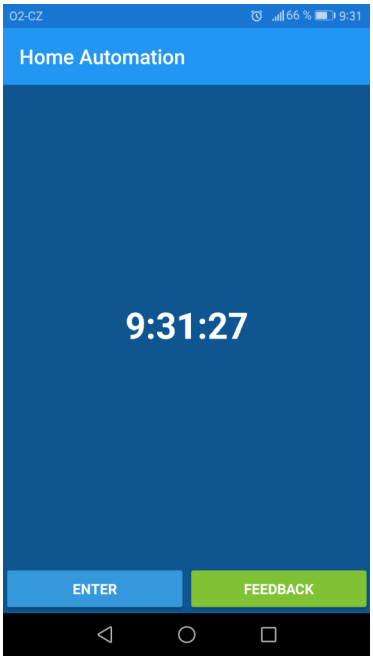
Příloha B – Myšlenková mapa

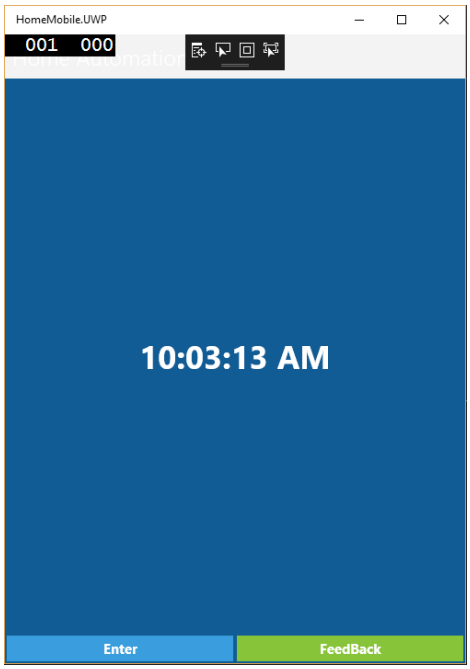


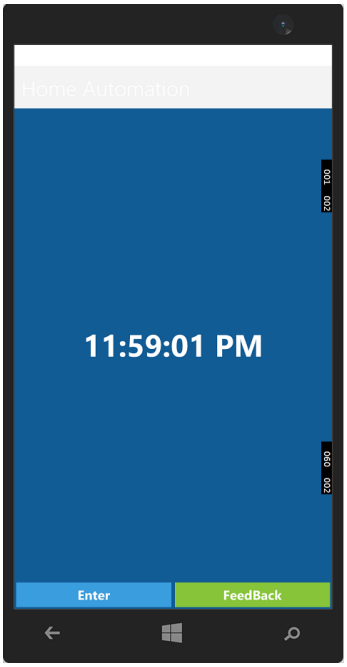
Příloha C – Schéma zapojení

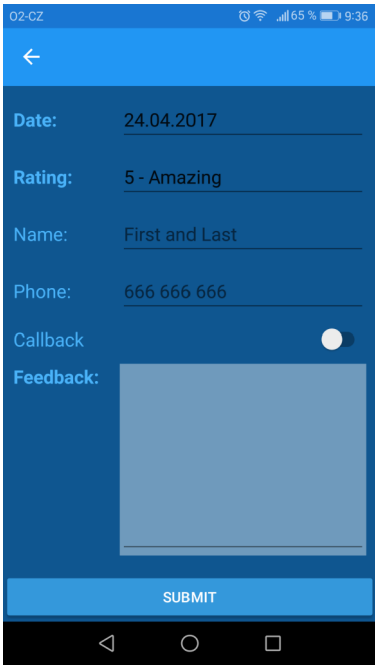


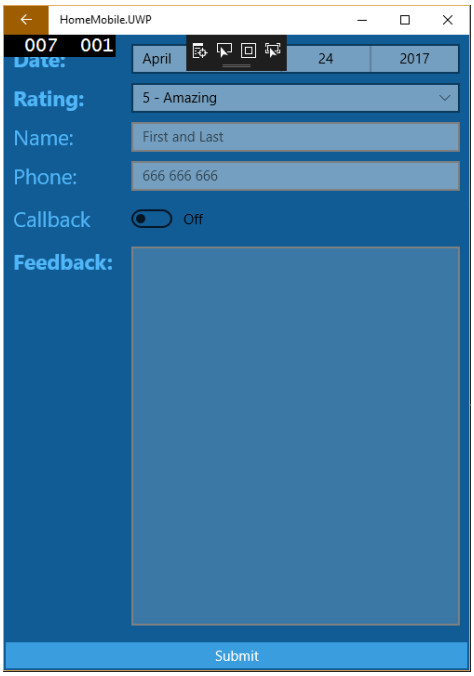
Příloha D – Snímky obrazovek aplikace HomeMobile (Android, UWP PC, UWP Mobile)

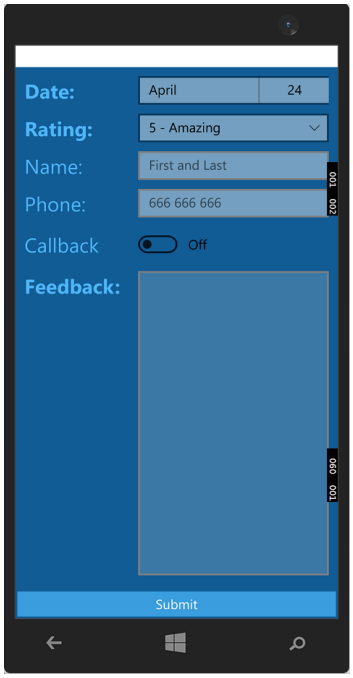


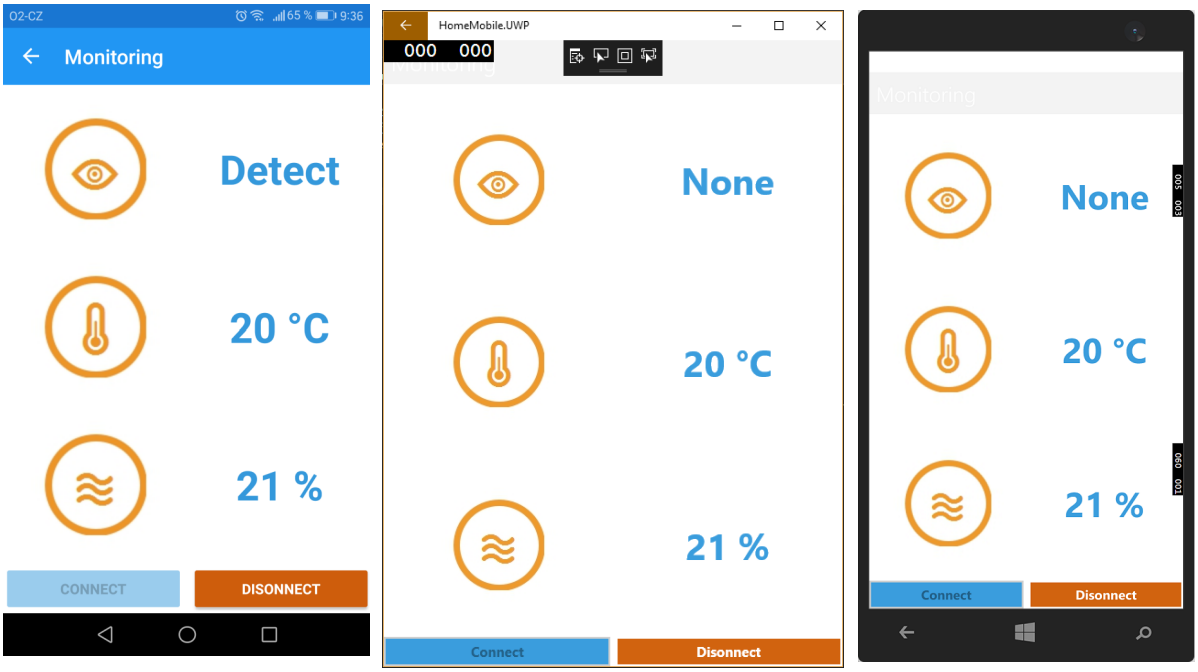




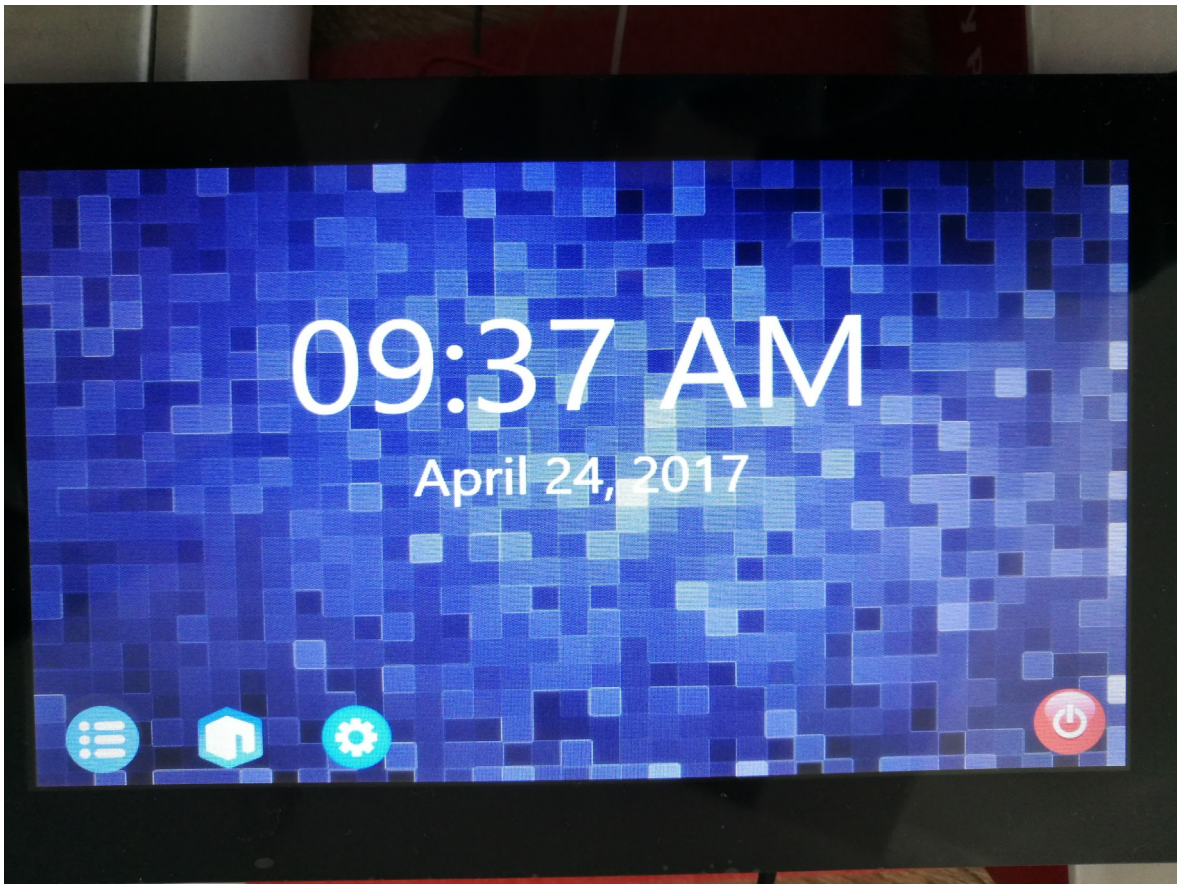


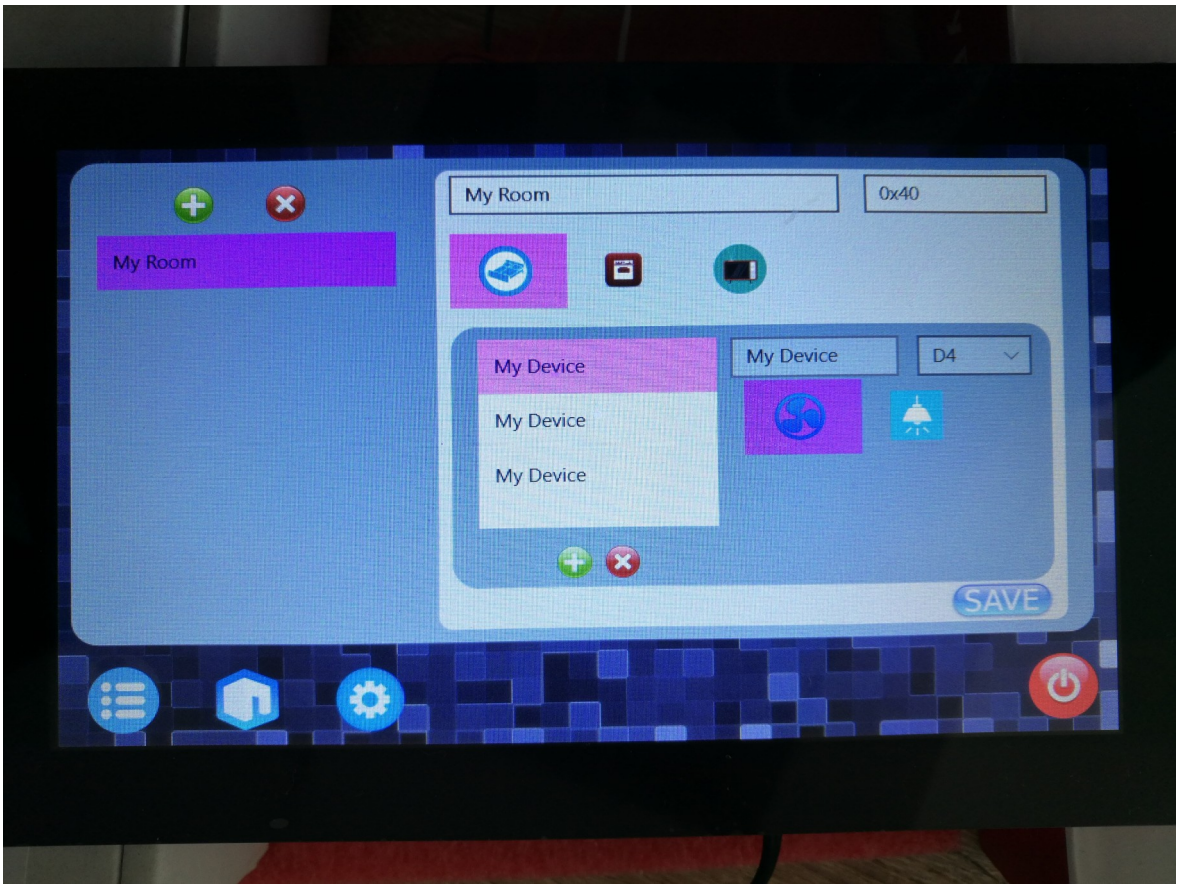


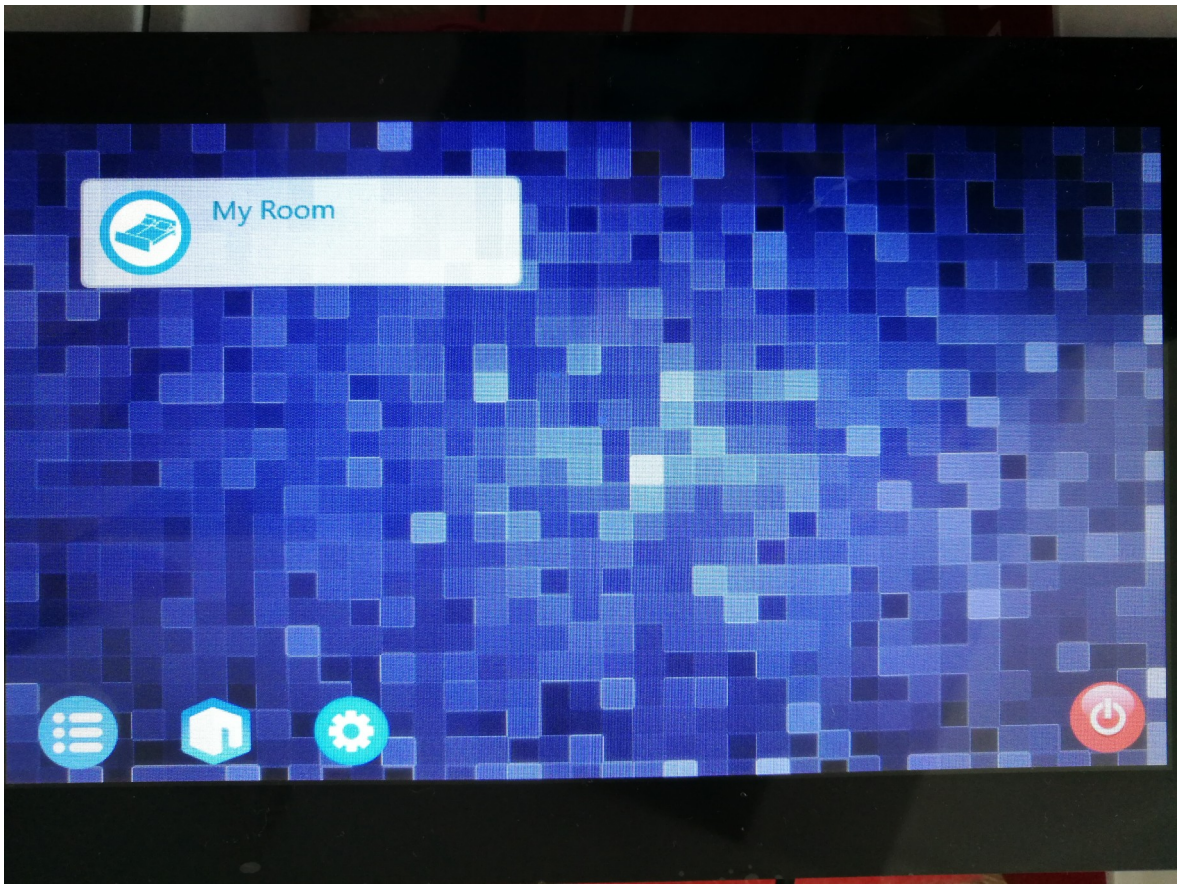



O2-CZ
Monitoring
Detect
20 °C
21 %
CONNECT
DISONNECT
HomeMobile.UWP
000 000
None
20 °C
21 %
Connect
Disonnect
Monitoring
None
20 °C
21 %
Connect
Disonnect

Příloha E – Snímky obrazovek aplikace Home_Automation_RPi na přístroji Raspberry Pi

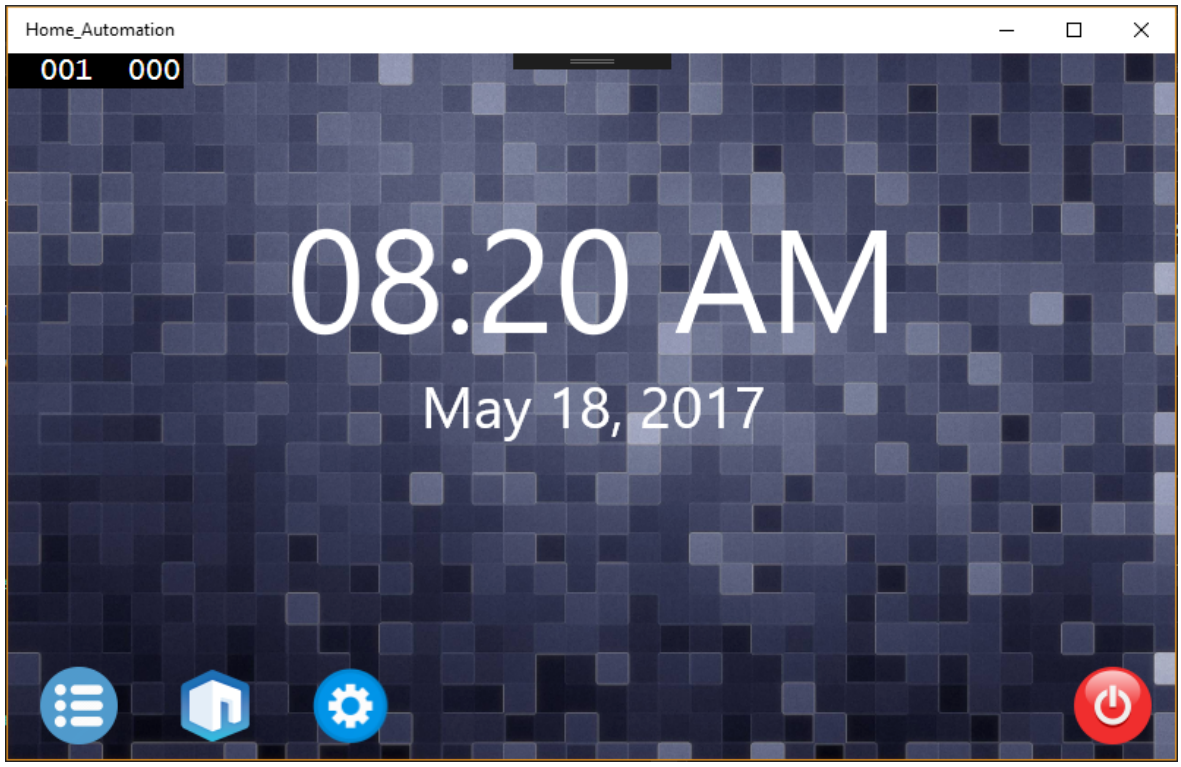


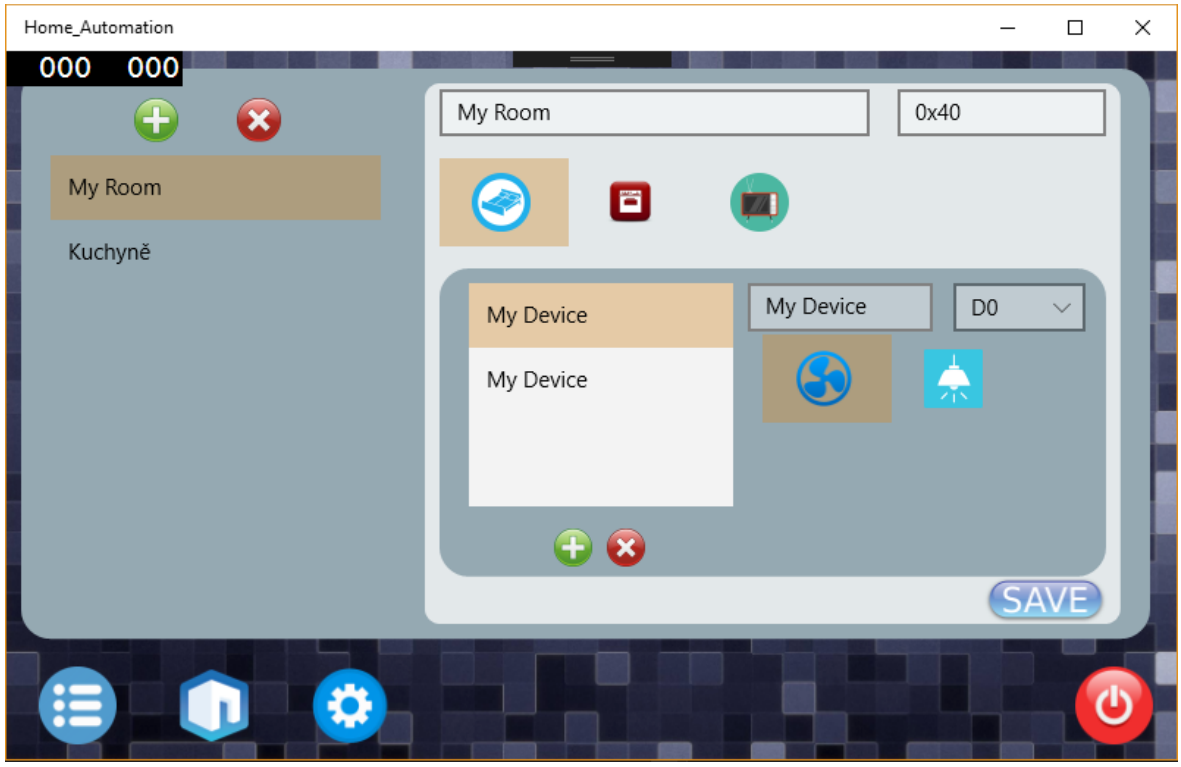


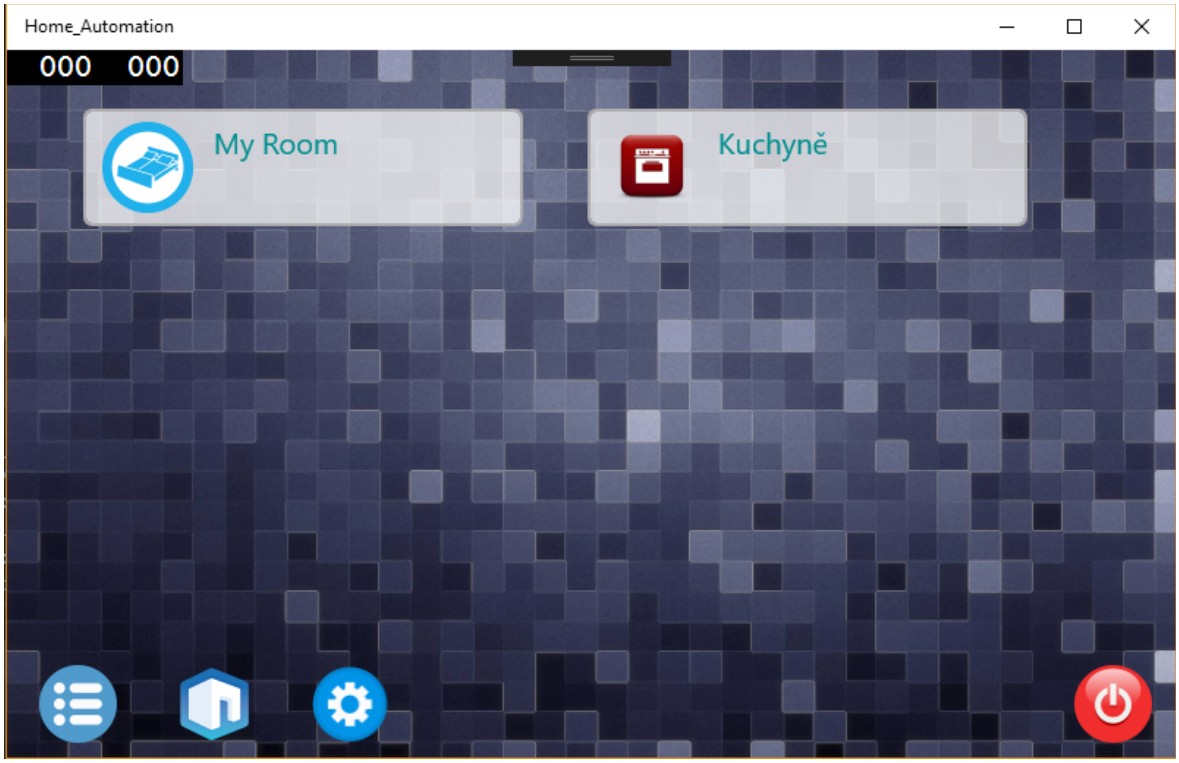

My Room

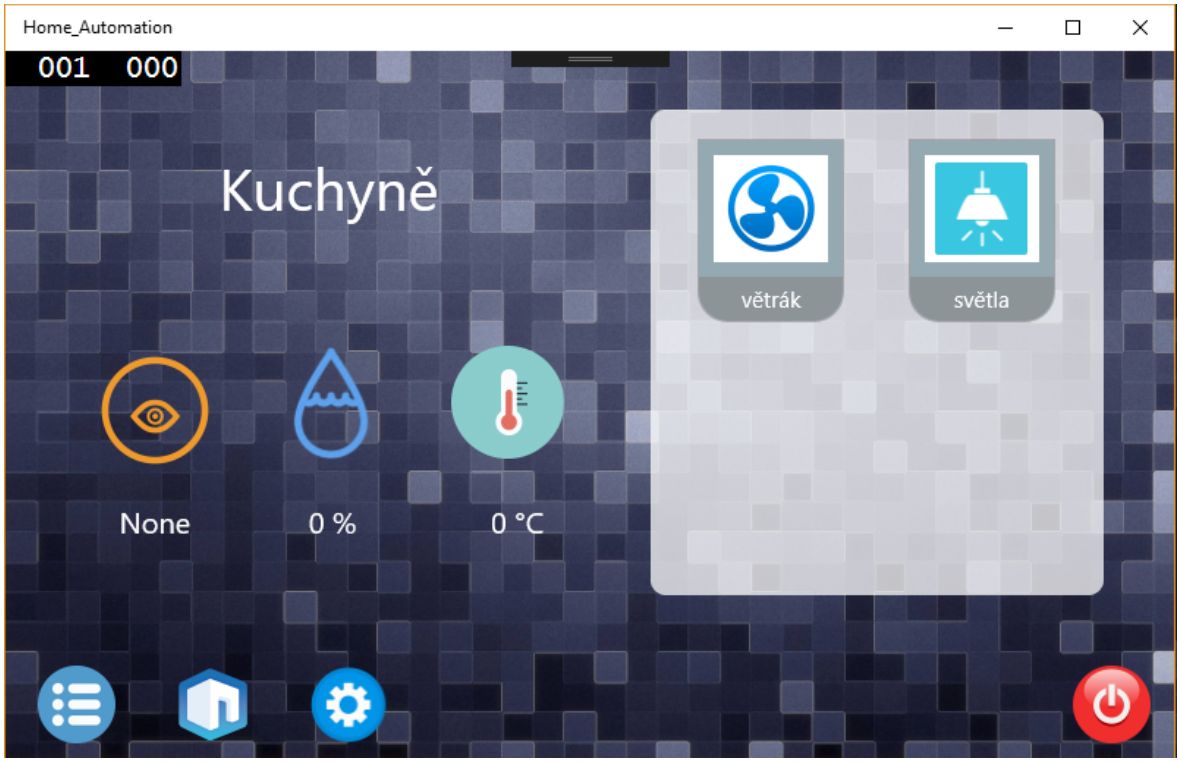

My Room
My Device
My Device
My Device
None
20 %
21 °C

Příloha F – Snímky obrazovek aplikace Home_Automation_RPi na počítači s Windows 10






Home_Automation
000 000
My Room
Kuchyně


Home_Automation
001 000
Kuchyně
větrák
světla
None
0 %
0 °C

Příloha G – Ukázky struktur kódu jednotlivých multiplatformních nástrojů (Xamarin, Appcelerator, Gluon)