JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Přírodovědecká fakulta

Aplikovaná informatika

Bakalářská práce

Notifikace v mobilních zařízeních

České Budějovice 21. dubna 2015

Vypracoval: Tomas Moravec

Vedoucí práce: Ing. Václav Novák, CSc.

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Moravec**
(jméno, příjmení, tituly)

Obor – zaměření studia: 1801R001 / Aplikovaná informatika

Katedra/ústav, kde bude práce vypracovávána: Ústav aplikované informatiky

Školitel: Ing. Václav Novák, CSc., vacnovak@prf.jcu.cz, M: 606 666 694
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
.....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Notifikace v mobilních zřízeních

Cíle práce:

Systém notifikace je vhodný pro sledování nových verzí knihoven a programů tedy řízení životního cyklu software.

1. Student navrhne systém řízení životního cyklu aplikací použitím notifikace.
2. Vytvoří systém jež bude schopen rozesílat notifikace a to i do bezdrátových sítí.
3. Stanoví architekturu a požadavky na systém příjemce využívající bezdrátových sítí jako např. Bluetooth, Zigbee, apod. , tak aby bylo možno sledovat životní cykly připojených aplikací
4. Provede příslušné testy a zhodnocení.

Základní doporučená literatura:

- KOCHAN, Stephen G. Programming in Objective-C 2.0. 2. vyd. Pearson Education, 2008. 2. ISBN 0321605543.
- KOCHAN, Stephen G. Programming in Objective-C (4th Edition) (Developer's Library). December 26, 2011. Addison-Wesley Professional, 2011. 4th Edition. ISBN 0321811909.
- KNASTER, Scott a Mark DALRYMPLE. Learn Objective-C on the Mac (Learn Series). January 7, 2009. Apress, 2009. ISBN 1430218150.

- Expanded Bluetooth Notification Support in iOS 7 Could Point Towards an iWatch [online] 2013 [cit. 26. 6. 2013] odkaz: http://selfscreens.com/archives/3693/expanded-bluetooth-notification-support-in-ios-7-could-point-towards-an-iwatch/
- iOS Developer Library [online] 2013 [cit. 26. 6. 2013] Odkaz: https://developer.apple.com/library/prerelease/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothOverview/CoreBluetoothOverview.html#//apple_ref/doc/uid/TP40013257-CH2-SW1
- ADDIE Model [online] 2010 [cit. 26. 6. 2013] Odkaz: http://www.learning-theories.com/addie-model.html

Financování práce: .....

Vedoucí práce: Ing. Václav Novák, CSc..... podpis: V. Novák

U externích vedoucích fakultní garant práce: ..... podpis: .....

Garant oboru bak. studia, pokud je obor zajišťován jinou katedrou/ústavem, než ze které je školitel (nepožaduje se u oboru biologie): ..... podpis: .....

Vedoucí katedry: RNDr. Libor Dostálek podpis: .....

Případný souhlas vedoucího ústavu AV: ..... podpis: .....

V Českých Budějovicích dne 9. 1. 2014 Podpis studenta: .....

Bibliografické údaje

Autor, Moravec Tomáš 2015: Notifikace v mobilních zařízeních.

[Notification in mobile devices. Bc.. Thesis, in Czech.] – 48 p. Faculty of Science,

The University of South Bohemia, České Budějovice, Czech Republic.

Anotace v češtině

Tato bakalářská práce se zabývá notifikacemi v mobilních zařízeních s operačním systémem iOS od společnosti Apple. Je zde popsána a vysvětlena funkčnost notifikací a Bluetooth v praxi. Důležitou částí práce je aplikace využívající jak notifikace, tak bluetooth. Na závěr je uvedena možnost budoucího rozšíření aplikace a její využitelnost.

Anotace v angličtině

This thesis deals with the notification on mobile devices with iOS from Apple. There is also described and explained in practice functionality of notifications and Bluetooth. The important part is the application using both notification and bluetooth. In conclusion, given the possibility of future expansion and usability of the application.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 22. dubna 2015

Podpis _____

Poděkování

Mé poděkování patří Ing. Václavu Novákovi, CSc. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

Úvod

V dnešní době mobilních zařízení je kladen veliký důraz na příslušenství. Velmi známé jsou sportovní náramky využívající bluetooth technologii ke komunikaci s mobilním zařízením a informovaním tak uživatele o své fyzické aktivitě. Mnoho příslušenství je využíváno i lékařství kde jsou například váhy, které odesílají informace rovnou do mobilního telefonu nebo tlakoměry nebo glukometry. Poslední dobou se rozrůstá příslušenství v podobě hodinek které více ulehčují práci s mobilním zařízením.

Cílem této práce bylo vytvořit aplikaci využívající bluetooth pro komunikaci mezi dvěma iOS přístroji, kde je jeden přístroj využit jako vysílač rozesílající informace do okolí. Jako využití by mohla být cílená reklama pro uživatele iOS. Kdy obchodník zveřejní své vysílací uuid a pokud si uživatel přeje zachytávat informace, uloží si uuid do přijímače. Aplikace bude kontrolovat v okolí signál vysílače, a pokud se přiblížíme k nějakému obchodu využívající vysílač s naším uloženým uuid, uživatel obdrží notifikaci, že se nachází v blízkosti daného obchodu a pomocí notifikace mu může být odeslána například sleva na kávu, pokud se tam nyní zastaví.

Součástí této práce je vysvětlení notifikací a jejich funkcí a vysvětlení bluetooth 4.0. Dále aplikace pro testování spojení dvou iOS zařízení, rozesílání notifikací a zhodnocení zátěže aplikace na iPhonu 5S.

1. Notifikace

Notifikace se dělí do několika skupin. První skupina je tzv. User Notification a druhá skupina je Silent Notification.

1.1. User Notification – iOS 7

Notifikace jsou jednoduchá oznámení, která umožňují vypnutým aplikacím dát o sobě vědět. Je několik možných způsobů, jak informovat uživatele pomocí notifikace.

- Označením v pravém horním rohu na ikoně aplikace. V červeném kroužku, který obsahuje číslo s počtem notifikací čekajících na odpověď od uživatele.
- Pokud přístroj nemáme po ruce, tak je možné informovat o nové notifikaci pomocí zvuku.
- Upozorněním „Alerts“, což je vyskakovací okno. Na iOS se toto okno sroluje z horního okraje obrazovky na určitou dobu a poté zmizí. Na OS X se objevuje v notifikačním centru v rohu monitoru.

Alerts se mohou objevit na obrazovce zařízení mnoha způsoby.

- Modal alerts
    - Notifikace se zobrazí uprostřed obrazovky a uživatel je nucen na ně okamžitě reagovat. Tyto notifikace je možné buď uzavřít, kde to notifikace bere, jako že je uživatel informován, nebo potvrdí notifikaci a ta uživateli otevře příslušnou aplikaci, pro kterou byla notifikace určena.
- Pop up Notification
    - Notifikace se nenápadně objeví v horní části obrazovky, aby nepřekážela uživateli při práci a po několika vteřinách opět zmizí. Pokud na ní bude chtít uživatel okamžitě reagovat, stačí na notifikaci poklepnout. Pokud měl uživatel zapnutou jinou aplikaci, notifikace ji ukončí do pozadí a přenese do popředí aplikaci, pro kterou byla notifikace určená.
- Lock screen Notification
    - Tyto notifikace se zobrazují v notifikačním centru, které se zobrazí při táhnutí shora dolů na displeji zařízení. Zde je notifikace uchovávána do doby, než se jí bude chtít uživatel věnovat, nebo do doby, kdy už není aktuální.

Notifikace se do zařízení dostávají dvěma způsoby.

- Local notifications - tyto notifikace se dostávají přímo od samotné aplikace. O lokální notifikace se stará sama aplikace, proto pro ně není nutné internetové připojení.

- Push notifications - tyto notifikace se dostávají ze serveru, neboli z APNs - Apple Push Notification Service. Apple Push Notification Service dodává a směřuje notifikaci od zadaného poskytovatele služby, až po zařízení, pro které je notifikace určena. Tyto notifikace směřují pouze jedním směrem, proto nelze informovat o stoprocentním doručení poskytovatele. Push notifikace lze rozesílat více zařízením najednou. Pokud uživatel využívá jednu aplikaci na více zařízeních, je možné, aby aplikace notifikovala všechna zařízení. (1)

1.2. Silent Notification – iOS 7

Silent notification, neboli tiché notifikace, jsou určeny pro aplikace, u kterých nevyžadujeme, aby na novy obsah uživatel reagoval. Notifikace se ze serveru doručí do aplikace, kde aplikace v pozadí provede potřebné změny a dále už nic nedělá. Uživatel po přepnutí do aplikace, má již nový obsah stažený bez nutnosti čekání na načtení. Tyto notifikace se převážně využívají u aplikací na více zařízeních, kde má uživatel rozpracovanou práci například na tabletu a pomocí notifikací se obsah aktualizuje i na mobilním zařízení bez vědomí uživatele.

2. User Notifications

V iOS 8 je nejprve nutné registrovat uživatelské notifikace. Pokud tedy naše aplikace využívá uživatelské notifikace, tak se při prvním spuštění aplikace zeptá, zda uživatel povoluje aplikaci využívat notifikace.

Registrace:

```
UIUserNotificationType types = UIUserNotificationTypeBadge |
UIUserNotificationTypeSound| UIUserNotificationTypeAlert
```

obr. č. 1.1

Jako první je třeba určit, jaký typ notifikace chceme, aby aplikace dovolovala. Na obrázku 1.1 jsou využity veškeré dostupné typy a jsou uloženy do proměnné types.

- UIUserNotificationTypeBadge
    - Označení v pravém horním rohu na ikoně aplikace. V červeném kroužku, který obsahuje číslo s počtem notifikací čekajících na odpověď od uživatele.
- UIUserNotificationTypeSound
    - Pokud přístroj nemáme po ruce, tak je možné informovat o nové notifikaci pomocí zvuku.
- UIUserNotificationTypeAlert
    - Upozornění „Alerts“, což je vyskakovací okno. Na iOS se toto okno sroluje z horního okraje obrazovky na určitou dobu a poté zmizí. Na OS X se objevuje v notifikačním centru v rohu monitoru.

Jako druhý vytvoříme objekt, do kterého uložíme námi vybrané typy, viz. obr. č. 1.1, které budeme chtít použít.

```
UIUserNotificationSettings *mySettings = [UIUserNotificationSettings
settingsForTypes:types categories:nil];
```

obr. č. 1.2

Poslední bod kódu na obr. č. 1.2 categories, který je nil[1] má co dočinění s akcemi notifikací, které podrobněji projdeme později.

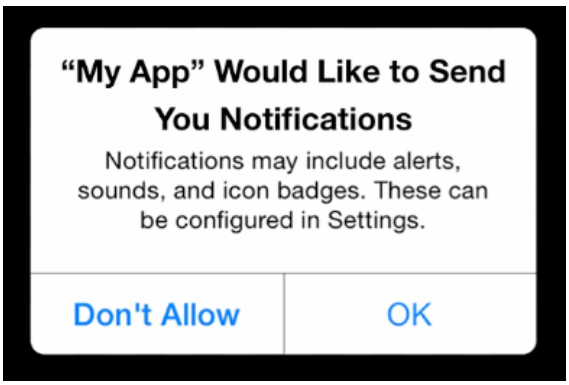
[1] Ukazatel objektu na nic(prázdno)

Nakonec objekt předáme instanci pro nastavení registrací, viz. obr. č 1.3

```
[[UIApplication sharedApplication]registeredUserNotificationSettings:mySettings];
```

obr. č. 1.3

Po splnění těchto tří kroků se po zapnutí aplikace objeví okno, viz. obr. č. 1.4 ,dotazující se uživatele, zda povoluje používání notifikací. Zároveň se aplikace uloží do nastavení, kde je možné toto rozhodnutí libovolně změnit.



obr. č. 1.4

Po odsouhlasení nebo neodsouhlasení, zda může aplikace využívat notifikace se zavolá v kódu UIApplicationDelegate Callback, který ověří, jaké to nastavení pro notifikace bude použito. Tyto informace jsou obsaženy v kódu z obr. č. 1.1.

```
-    (void)application:(UIApplication *)application didRegisterUserNotificationSettings:
                        (UIUserNotificationSettings*)notificationSettings {

                        //Uzivatel povolil příjmání user notifikací tohoto typu
                    UIUserNotificationType allowedTypes = [notificationSettings types];
}
```

obr. č. 1.5

Je ale potřeba zajistit, aby aplikace neoznamovala notifikace, pokud je uživatel později vypnul. Proto je potřeba před zavoláním delegáta Callback ověřit, jaké nastavení je uloženo pro používání notifikací, viz obr č. 1.6.

```
-(void)getReadyForNotification {
    UIUserNotificationSettings *currentSettings = [[UIApplication
    sharedApplication] currentUserNotificationSettings];

    [self checkSettings:currentSettings];
}
```

obr. č. 1.6

3. Notification Action

Notifikační akce v iOS7 se ve všech zobrazení chovala stejně a to tak, že po stisknutí na notifikaci se spustila daná aplikace patřící k té notifikaci. V iOS8 přibylo více možností jak využít akce v notifikacích. Na zamčené obrazovce k notifikaci přibyly hned dvě nové akce a to přijmout a zamítnout. Tyto akce se přidaly jak do notifikace v notifikačním centru, tak do notifikačního baneru. Navíc, v notifikačním okně na hlavní obrazovce je možné využít až čtyři notifikační akce.

Pro používání notifikační akce je zapotřebí se držet těchto tří po sobě jdoucích kroků:

- registrovat akci
- push/schedule[2] lokalní notifikace
- manipulace s akcí

Využití a jakým způsobem notifikační akce fungují si předvedeme na příkladu notifikace z kalendáře, kdy někdo pošle pozvánku z kalendáře na nějakou událost.

3.1. Registrace notifikační akce

Při registrování notifikační akce je nejprve potřeba vytvořit určitou akci. Nyní si vytvoříme akci, viz obr. č. 2.1, pro přijmutí pozvánky z kalendáře.

```
UIMutableUserNotificationAction *acceptAction =
[[UIMutableUserNotificationAction alloc] init];
```

obr. č. 2.1

[2] push notifikace ze serveru a schedule, neboli plánované notifikace z aplikace

Nadále je potřeba identifikovat akci, aby je bylo mezi sebou možné rozeznat a hlavně aby je bylo možné později zavolat v kódu. Pro identifikování akce postačí tento krátký kus kódu, viz obr. č. 2.2

```
acceptAction.idetifier = @“ACCEPT_IDENTIFIER”;
```

obr. č. 2.2

Jako další bod nastavíme titulek akce. Titulek akce je ve formátu string a je viditelný pro uživatele. Titulek se zobrazuje pří zobrazení notifikace. Pro titulek akce použijeme následující kód na obr č. 2.3

```
acceptAction.title = @“Accept”;
```

obr č. 2.3

Další bod určuje, zda je potřeba mít aplikaci zapnutou, nebo může zůstat na pozadí při provedení akce. Pokud necháme aplikaci na pozadí, je tím rozuměno, že i zařízení může být v tu chvíli uzamčené. U kalendáře můžeme nechat aplikaci na pozadí, viz obr č. 2.4

```
acceptAction.activationMode = UIUserNotificationActivationModeBackground;
```

obr č. 2.4

Předposlední bod určuje, zda má být akce destruktivní. Pokud je nastaveno na ano, bude notifikační akce v notifikačním centru zobrazena červeně. Při zadání ne, je akce zobrazena modře. Toto nastavení by se v praxi mohlo použít například u notifikační akce pro “smazat”.

```
acceptAction.destructive = NO;
```

obr č.2.5

Poslední část se zabývá tím, zda je potřeba autentifikace uživatele. Pokud je nastaveno, že je autentifikace potřeba a zařízení bude uzamčeno, uživatel bude po stisknutí notifikační akce požádán o zadání kódu, nebo o přiložení otisku prstu. Na tomto příkladu můžeme ponechat NO, protože po stisknutí není potřeba zapínat aplikaci a může se naše volba uložit na pozadí, viz obr. č. 2.4

```
acceptAction.authenticationRequired = NO;
```

obr. č. 2.6

Shrnutí naší notifikační akce “Accept”

Action Property	Value
title	@“Accept”
activationMode	UIUserNotificationActivationModeBackground
destructive	NO
authenticationRequired	NO

Na porovnání je zde shrnutí notifikační akce pro zahození emailu “Trash”

Action Property	Value
title	@“Trash”
activationMode	UIUserNotificationActivationModeBackground
destructive	YES
authenticationRequired	YES

Na rozdíl od Accept akce bude tlačítko zabarveno červenou barvou a po stisknutí na notifikační akci bude potřeba vložit kód nebo otisk prstu.

Jako další notifikační akci můžeme mít akci pro odpověď “Reply”

Action Property	Value
title	@“Reply”
activationMode	UIUserNotificationActivationModeForeground
destructive	NO
authenticationRequired	YES*

Protože pro odpověď je potřeba určité UI, tak není možné tuto akci provádět na pozadí, ale bude potřeba zapnout příslušnou aplikaci pro odpověď. Zároveň bude nutné ověřit uživatele kódem nebo otiskem, protože bude potřeba zapnout aplikaci. Proto at je nastaveno na obr.č. 2.6 YES a nebo NO, nebude to mít žádný vliv.

3.2. Kategorie

Po vytvoření potřebných akcí je rozdělíme do kategorií. Jako příklad použiji akce pro příjem pozvánky z kalendáře.

Např. kategorie:

- **Invite**
    - Accept
    - Maybe
    - Decline

Nejprve je zapotřebí vytvořit novou kategorii. Kategorii vytvoříme podobně jako notifikační akci. Kód pro vytvoření kategorie, viz obr. č. 2.7. Poté nastavíme identifikace kategorii stejně, jako tomu bylo u notifikačních akcí. Identifikaci nastavíme viz obr. č. 2.8.

```
UIMutableUserNotificactionCategory *inviteCategory =
[[UIMutableUserNotificationCategory alloc] init];
```

obr. č. 2.7

```
inviteCategory.identifier = @“INVITE_CATEGORY”;
```

obr. č. 2.8

Nakonec je potřeba nastavit Context. Podle toho, jaký context budeme chtít využívat, se nám budou notifikační akce zobrazovat. Můžeme použít Default, viz obr. č. 2.9, který zobrazuje až čtyři notifikační akce. Nebo použijeme Minimal, viz obr. č. 2.10, který dokáže využít pouze dvě notifikační akce.

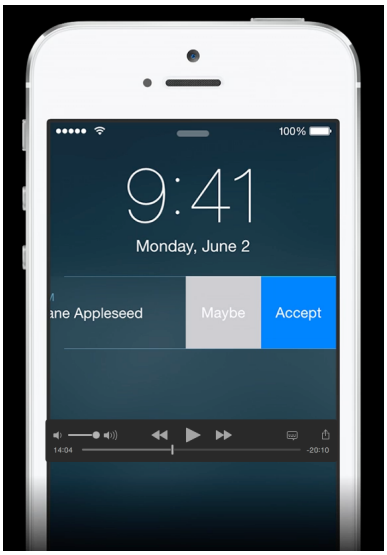
```
[inviteCategory setActions:@[acceptAction, maybeAction, declineAction]
forContext:UIUserNotificationActionContextDefault];
```

obr. č. 2.9

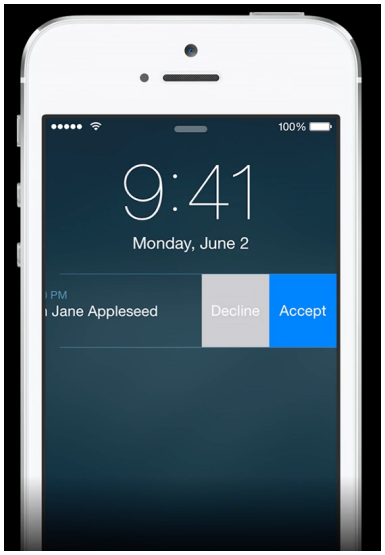
```
[inviteCategory setActions:@[acceptAction, declineAction]
forContext:UIUserNotificationActionContextMinimal];
```

obr. č. 2.10

Při použití Default se nám na uzamčené obrazovce zobrazí notifikace a po přejetí prstem zprava doleva se nám zobrazí první dvě notifikační akce viz obr. č. 2.11. Ale pokud chceme, aby se nám zobrazily určité dvě akce, tak použijeme Minimal viz obr. č. 2.12. Nevýhoda Minimal je, pokud se notifikace zobrazí na odemčené ploše, tak se zobrazí pouze tyto dvě akce. Ale při použití Default se nám zobrazí všechny námi určené akce, viz obr. č. 2.13



obr. č. 2.11



obr. č. 2.12

obr. č. 2.13

3.3. Nastavení kategorií

Vytvořené kategorie je zapotřebí registrovat. Abychom toto mohli udělat, je potřeba vytvořit NSSet, viz obr. č. 2.14, kde vypíšeme veškeré kategorie které jsme vytvořili.

```
NSSet *categories = [NSSet setWithObjects:inviteCategory, alarmCategory, …
```

obr. č. 2.14

Poté NSSet předáme nastavení notifikací, viz obr. č. 2.15, a nakonec zaregistrujeme toto nastavení do sharedApplication, viz obr. č. 2.16.

```
UIUserNotificationSettings *settings =
[UIUserNotificationSettings settingsFotTypes:types categories:categories];
```

obr. č. 2.15

```
[[UIApplication sharedApplication]registerUserNotificationSettings:settings];
```

obr. č. 2.16

3.4. Nastavení kategorie pro Lokální a Push notifikace

U Push notifikací je nutné v Push Payload[3] nastavit, do které kategorie příchozí notifikace patří. To se řeší přidáním jednoho řádku kódu, viz obr. č. 2.17. Velikost těchto Payload bylo zvětšeno u iOS8 na 2KB maximální velikosti z 256b.

```
{
        "aps" : {
                "alert" : "Byl jsi pozván",
                "category" : "INVITE_CATEGORY",
        }
}
```

obr. č. 2.17

U lokální notifikace standardně vytvoříme notifikaci, viz obr. č. 2.18, a navíc přídáme do které kategorie notifikace patří, viz obr č. 2.19, a dále vytváříme notifikaci jako předtím naplánováním notifikace, viz obr. č. 2.20.

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
```

obr. č. 2.18

[3] Payload je JSON kod obsahující nastavení a obsah příchozí notifikace ze vzdáleného serveru.

```
notification.category = @“INVITE_CATEGORY”;
```

obr. č. 2.19

```
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

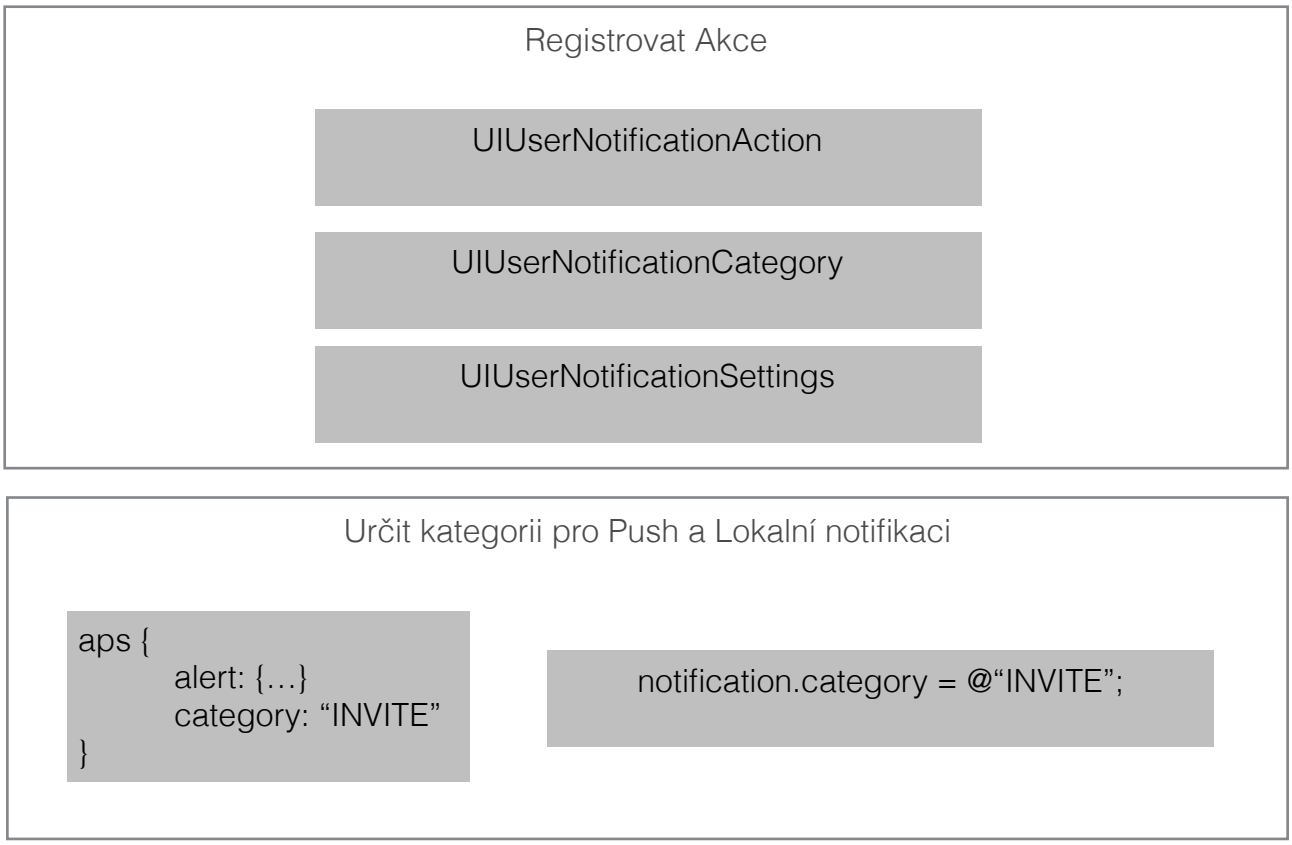
obr. č. 2.20

3.5. Manipulace s akcemi

Po odeslání notifikace a zobrazení uživateli, který na určitou notifikaci kliknul, je zapotřebí zařídit správnou manipulaci s akcí, neboli co se má stát po stisknutí na notifikaci. Pokud notifikace dorazí na uzamčenou nebo odemčenou obrazovku a aplikace není zapnutá je potřeba zapnout danou aplikaci a poté provést notifikační akci.

3.6. Souhrn pro používání notifikačních akcí

Pro správné používání notifikačních akcí je třeba dodržet tento postup:



Zařídit manipulaci s akcemi přídáním metody do UIApplication delegata

application:hadleActionWithIdentifier:
forRemoteNotification:completionHandler:

application:hadleActionWithIdentifier:
forLocalNotification:completionHandler:

4. Remote notifikace

Pro remote notifikace máme dva typy

1. User type remote notification

Náš APNs[4] server odešle Push notifikaci do našeho zařízení pomocí alertu v Payload, viz obr. č. 3.01, a následně se objeví na obrazovce notifikační okno s přijatou notifikací s akcemi.

2. Silent type remote notification

Náš APNs server odešle Push notifikaci do našeho zařízení pomocí content-available: 1, viz obr. č. 3.02, iOS8 probudí aplikaci na pozadí a synchronizuje nová data která jsou na serveru dostupná. Poté, až uživatel spustí aplikaci, bude mít aktualizovaná data již v sobě. Tento způsob se například používá u messengeru.

```
aps {
        alert: {…}
}
```

obr. č. 3.01

```
aps {
        content-avaliable: 1
}
```

obr. č. 3.02

[4] Apple push notification server

5. Location notifikace

Při použití lokačních notifikací máme aplikaci a chceme aplikaci umožnit nás upozornit, když například dorazíme na určité místo na mapě. Lokační notifikace nás může upozonit i když místo opustíme, nebo může i přátelům odeslat naší polohu.

Pro tyto notifikace používáme *UILocalNotification*

- Notifikace se spustí při odchodu nebo při příchodu na námi určené místo na mapě
- notifikace se může spustit jednou nebo vícekrát
- lokální notifikaci je potřeba registrovat do *Core location*

5.1. Registrace lokačních notifikací

Pro správné fungování lokačních notifikací je potřeba vytvořit CLLocationManager, viz obr. č. 4.01, a poté nastavit delegáta pro naší notifikaci, viz obr. č. 4.02, Jako u všech notifikací je potřeba se dotázat uživatele, zda aplikace může sledovat jeho polohu, viz obr. č. 4.03.

```
CLlocationManager *locMan = [[CLLocationManager alloc] init];
```

obr. č. 4.01

```
locMan.delegate = self;
```

obr. č. 4.02

```
[locMan requestWhenInUseAuthorization];
```

obr. č. 4.03

Pokud uživatel povolí sledovat polohu, může aplikace sledovat uživatele ,když bude spuštěna, ne když poběží na pozadí.

5.2. Core Location registration callbacks

Nyní potřebujeme vyřešit zpětné volání z *CLLocationManager*

```
// Manager delegate obdrží toto zpětné volání, že u CLLocationManager se změnil
autorizační status, neboli že uživatel povolil používání lokačních notifikací

- (void)locationManager:(CLLocationManager *)manager
        didChangeAuthorizationStatus:(CLAuthorizationStatus)status {


// Jediné co potřebujeme je ujistit se, že je opravdu aplikace oprávněna používat
lokační notifikace
        BOOL canUseLocationNotifications =
                (status == kCLAuthorizationStatusAuthorizedWhenInUse);

        if (canUseLocationNotifications){

                // pokud jsou lokační notifikace od uživatele povoleny můžeme je
začít plánovat
                [self startShowingLocationNotifications];
        }
}
```

5.3. Notification scheduling

Nyní nastavíme plánování lokační notifikace, když například dorazíme na určené místo

```
- (void)startShowingNotifications {

    // Nejprve je potřeba, tak jako u každé notifikace, vytvořit instanci UILocalNotification
    UILocalNotification *locNotification = [[UILocalNotification alloc] init];

    // a nastavit, jaký typ notifikace to bude a popřípadně jaký text se zobrazí uživateli. Zde se po příchodu na určené místo zobrazí zpráva “Dorazil jsi na místo”.
    locNotificaiton.alertBody = @“Dorazil jsi na místo.”;

    // Zde máme nastaveno, zda má námi zvolenou pozici aplikace reagovat vícekrát nebo jen jednou. Defaultně je nastaveno, že se notifikace zobrazí jen jednou proto by tento řádek nemusel být napsán a automaticky by se notifikace zobrazila jednou a poté už by na danou pozici nereagovala.
    locNotification.regionTriggersOnce = YES;

    // Zde se nastavuje podle čeho má být spoštěč určen. Tady je používán Circular Region, kde jsou uloženy souřadnice místa a jak velkej radius pro aktivaci má být kolem souřadnic a nakonec idetifikátor zvolené pozice pro případ, kdy máme víc upozornění na více místech.
    locNotification.region = [[CLCircularRegion alloc]
                              initWithCenter: LOC_COORDINATE
                              radius: LOC_RADIUS
                              identifier: LOC_IDENTIFIER];

    [[UIApplication sharedApplication]
    // Nyní můžeme notifikaci naplánovat, tak jako kteroukoliv jinou notifikaci
        cheduleLocalNotification: localNotification];
```

5.4. Notification handling

Pokud uživatel dorazil na místo určení a aplikace běží na pozadí, provede se spuštění upozornění a dorazí lokální notifikace se zprávou, že dorazil na místo určení a rozešle přátelům, že dorazil na místo určení. Ale pokud je aplikace zapnutá a běží na popředí, tak se zavolá tato část kódu.

```
- (void) application: (UIApplication *) application didReceiveLocalNotification:
        (UILocalNotification *) notification
{
        // Zkontroluje, zda region není NILL
        CLRegion *region = notification.region;

        if (region) {
                // Rozešle zprávu přátelům, že dorazil na místo
                [self tellFriendsUserArrivedAtRegion:region];
        }
}
```

6. Bluetooth

Pro využívání Bluetooth v iOS a v OS X je zapotřebí využívat Core Bluetooth framework, který poskytuje veěkeré potřebné třídy pro práci s ním. Aplikace může například zjišťovat, prozkoumavát a vzájemně reagovat s nízkoenergetickým periferním zařízením. Mac a iOS můžou mít funkci jako nízkoenergetické periferní zařízení pomocí Bluetooth 4.0 od OS X v10.9 a iOS 6.

V Bluetooth komunikaci jsou zásadní dva klíčové prvky

- Centrála (Central)
- Periferie (Peripheral)

Každý z těchto dvou klíčových prvků má svou roli v Bluetooth komunikaci.

- Centrála většinou využívá informace, které přebírá od periferie
- Periferie rozesílá, doslova inzeruje, kolem sebe nějaké informace, které cetrála zachytí a přebírá

7. Aplikace “Find Me”

Aplikace Find Me je jednoduchý program, který využívá Bluetooth 4.0 pro komunikaci a GPS pro určení pozice. Aplikace dokáže jak informace vysílat tak dokáže fungovat jako přijímač. Aplikace neustále vyhledává vysílač se shodným UUID a po nalezení se automaticky spárují. Přijímač informuje, zda byl v blízkosti nalezen nějaký vysílač a pomocí notifikací o tom informuje uživatele. Uživatel je poté informován o vzdálenosti k vysílači a o síle signálu pomocí Bluetooth. Pokud uživatel ztratí signál s vysílačem, bude o tom v notifikaci informován a aplikace se spustí do opětovného vyhledávání.

7.1 Základní nastavení

Při psaní této aplikace bylo nutné si určit, na kterých iOS zařízench bude aplikace využívána. Pro tento účel bylo vybráno univerzální použití jak pro iPhone tak pro iPad. Dále je dobré si hned na začátku určit, jaké framoworky budeme potřebovat.

Pro aplikaci byly využity tyto frameworky

- CoreGraphics.framework
    - obsahuje ovladače pro Quartz 2D drawing API. Quartz je využíván pro vektorovou grafiku v OS X a iOS.
- Foundation.framework
    - poskytuje spojení k mnoha funkcím v Core Foundation framework ke kterým patří:
        - Collection data types (např. array, sets…)
        - Bundles
        - String managment
        - Date and time managment
        - URL a stream manipulation
        - Bonjour
        - Communication port management
        - Cache support

- UIKit.framework
    - poskytuje přístup ke grafickým a hardwarovým ovladačům v iOS jako jsou např:
        - Cut, copy and paste
        - vytváření PDF
        - podporu sdílení přes Twitter, Facebook a ostatní služby
        - informace o stavu baterie
        - Proximity sensor
        - podpora Push notifikací
        - podpora Local notifikací
        - …
- CoreLocation.framework
    - poskytuje lokační a směrové informace aplikacím. Využívá zabudovanou GPS, operátora a Wi-Fi pro určení zeměpisné délky a šířky.
- CoreBluetooth.framework
    - dovoluje aplikaci využívat Bluetooth službu v zařízení např:
        - Vyhledávání a spárování Bluetooth zařízení
        - Vytvořit ze zařízení periferii pro poskytování informací
        - Vysílat iBeacon informace z iOS
        - Být informován o dostupných periferiích v okolí

Od iOS 8 je nutné na každou službu, kterou aplikace podporuje a využívá, se uživatele ptát. Aplikace využívá lokační a bluetooth hardware.

- Pro GPS - v Custom iOS Target Properties je nutné zadat:

Bundle version	String	1
NSLocationAlwaysUsageDescription	String	Find Me
Executable file	String	$(EXECUTABLE_NAME)

NSLocationAlwaysUsageDescription pro neustále monitorování lokace, ať už je aplikace zapnutá na popředí, ale i když je zapnutá v pozadí. Díky tomu můžeme mít aplikace stále zapnutou. Bude vyhledávat vysílající zařízení a bude je moci detekovat a notifikovat uživatele.

- Pro Notifikace - je kód možno napsat kdekoliv, kde je proveden minimálně před prvním použitím notifikace, neboli notifikace nikdy nedorazí pokud tato část kódu nebude provedena jedinkrát od instalace aplikace do zařízení. Kód je podrobně vysvětlen v kapitole 2.

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    UIUserNotificationType type = UIUserNotificationTypeBadge | UIUserNotificationTypeSound | UIUserNotificationTypeAlert;

    UIUserNotificationSettings *mySettings = [UIUserNotificationSettings settingsForTypes:type categories:nil];

    [[UIApplication sharedApplication] registerUserNotificationSettings:mySettings];
}
```

Dále bylo zapotřebí připravit si UUID pro spárování se správným zařízením. Neboť aplikace vyhledává pouze zadané UUID a zobrazí periferii se stejným UUID.

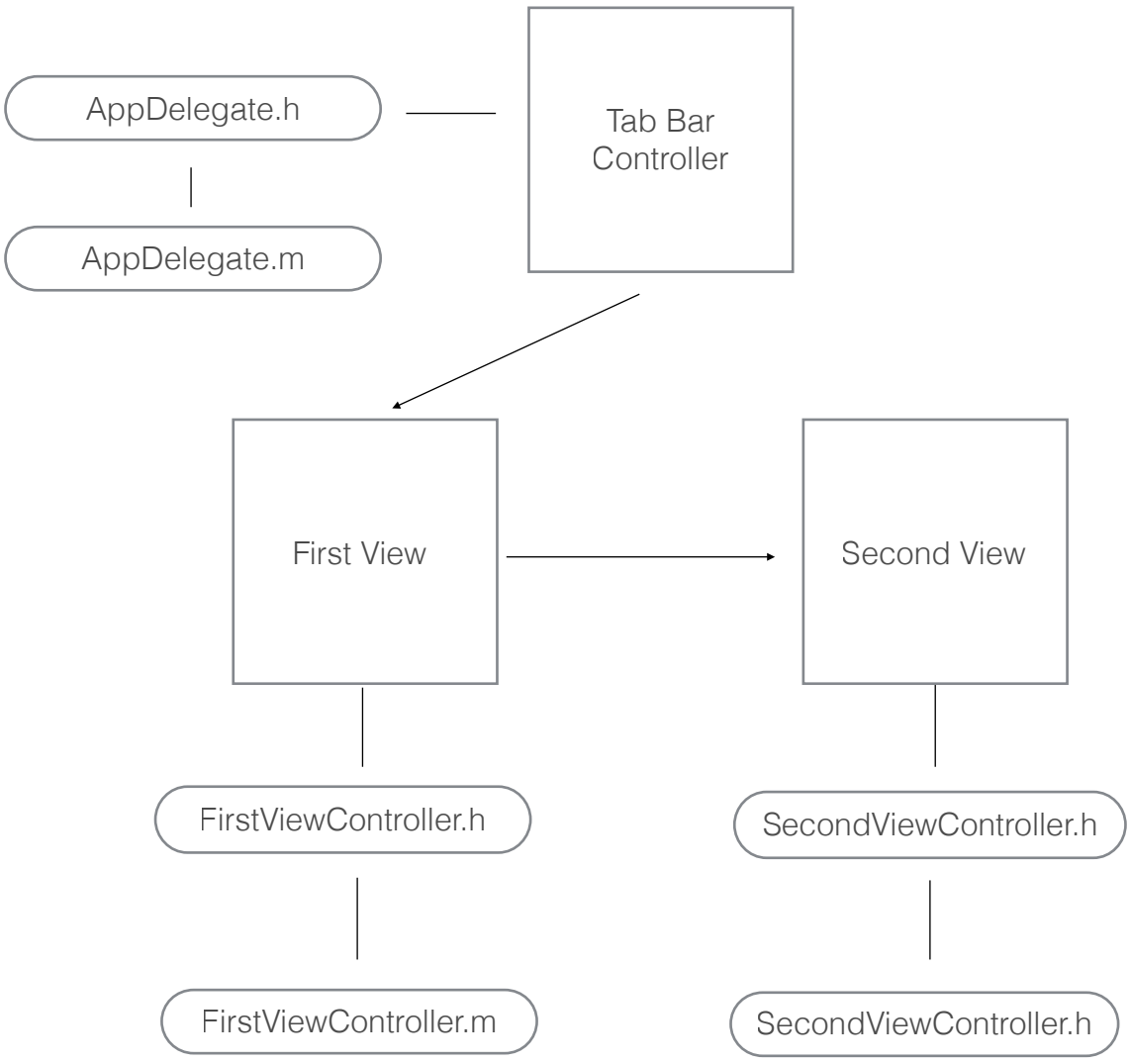
```
static NSString *uuid = @"1C4B9AF2-8DA6-41B1-87CD-2AC288BCA266";
static NSString *beaconId = @"com.morysoft.findme";
```

Dále připravit ID pro označení dané periferie. Tato informace se rozesílá a je možné jí přidat k nutnosti pro vyhledávání, pokud máme více periferií se stejným UUID ale rozdílným ID nebo pouze ponechat jako identifikace mezi více periferií.

Tyto dvě hodnoty jsou neměnné, a proto je možné zadat jako statické NSString hodnoty.

7.2 Uživatelské rozhraní

Jako uživatelské rozhraní, dále UI, je využitý Tab Bar Controller, který umožňuje přepínat na obrazovce mezi jednotlivým View pomocí tlačítek na spodu obrazovky. Každé View má vlastní třídu, neboli svou header (.h) a implementation (.m).

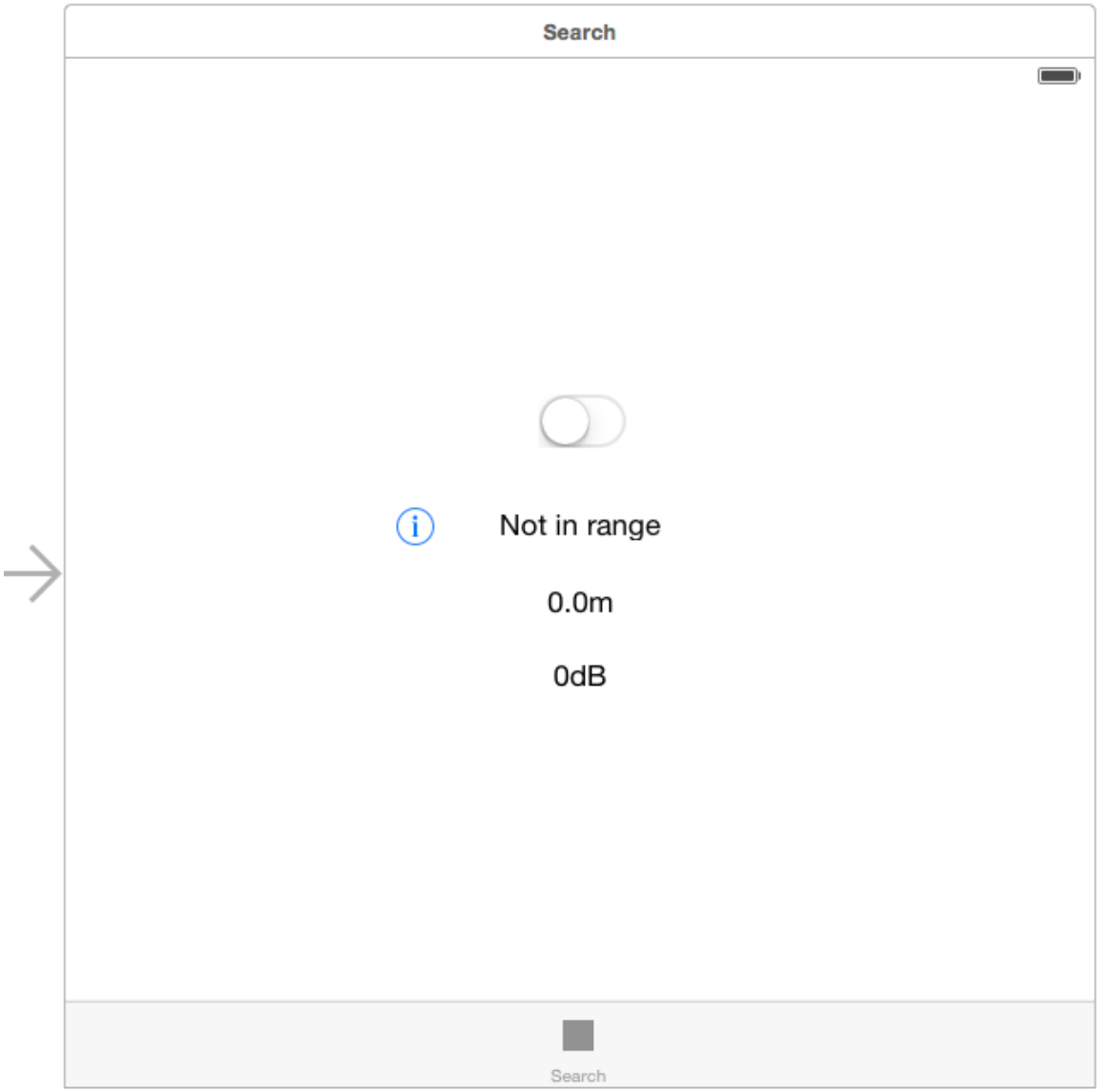


7.3.1 First View (Hide)

Zobrazení First View je používáno jako periferie pro rozesílání signálu. Pro periferii není potřeba žádného UI, ale jelikož bude moci uživatel vybírat mezi periferií a centrálou, je zde umístěn přepínač pro zapnutí a vypnutí vysílání. Dole na liště je možné přepnout na Second View, neboli na centrálu pro příjem vysílání.

7.3.2 Second View (Search)

Zobrazení Second View je už více uživatelské než první. Obsahuje taktéž přepínač pro zapnutí a vypnutí vyhledávání. Navíc je uživatel informován, zda probíhá vyhledávání nebo zda už periferii vyhledal. Po spojení s periferií je zobrazena vzdálenost od periferie a síla Bluetooth signálu v decibelech.



7.4 Kódování

Aplikace je psaná v Objective-C a v programu X-Code 6.3. Aplikace je univerzální a proto je možné jí naistalovat jak na iPhone tak na iPad. Použitelnost aplikace je od modelu iPhone 4S a iPad 2, který již využívá Bluetooth 4.0.

7.4.1 First View Controller

Jako první bylo potřeba definovat, jaké knihovny bude tento View Controller využívat. Importovány zde byly dvě a to CoreLocation a CoreBluetooth.

```
#import "FirstViewController.h"
#import <CoreLocation/CoreLocation.h>
#import <CoreBluetooth/CoreBluetooth.h>
```

Díky využitím těchto frameworků jsme dostali přístup a možnost zavolat Delegáty a metody.

```
@interface FirstViewController () <CBPeripheralManagerDelegate, UIAlertViewDelegate, UITextFieldDelegate>
```

Staticky se napsal NSString pro UUID a beaconID. UUID je univerzální unikátní identifikátor a je využíván pro rozpoznání vysílače. Pokud má vysílač shodné UUID tak se přístroje spárují. BeaconID je využíván pro identifikace vysílače. Je možné mít více vysílačů s totožným UUID a rozdělovat je a povolovat přístup až podle beaconID.

```
static NSString *uuid = @"1C4B9AF2-8DA6-41B1-87CD-2AC288BCA266";
static NSString *beaconId = @"com.morysoft.findme";
```

Při načtení First View Controlleru je potřeba alokovat peripheralManager, který umožňuje vysílání Bluetooth signálu.

```
-(void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

    if (!peripheralManager)
    {
        peripheralManager = [[CBPeripheralManager alloc] initWithDelegate:self queue:dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0)];
    }
    else
    {
        peripheralManager.delegate = self;
    }
}
```

Metoda switchUpdate kontroluje stav přepínače pro zapnutí a vypnutí vysílání. Pokud se vypínač zapne je zavolána metoda UpdateAdvertisedRegion.

```
- (IBAction)switchUpdate:(UISwitch *)sender {
    self.enabled = sender.on;
    [self updateAdvertisedRegion];
}
```

V updateAdvertisedRegion metodě se jako první kontroluje zda je zapnuté Bluetooth na zařízení a pokud není zapnuté je uživatel pomocí notifikace upozorněn, že je potřeba Bluetooth zapnout pro vysílání. Dále je potřeba zkontrolovat jaký stav na vypínači je zda je zapnutý nebo vypnutý. Pokud je zapnutý, vysílání se spustí pomocí námi vytvořeného peripheralManageru (startAdvertising a stopAdvertising).

```
-(void)updateAdvertisedRegion
{
    if (peripheralManager.state < CBPeripheralManagerStatePoweredOn)
    {
        NSString *title = NSLocalizedString(@"Bluetooth must be enabled", @"");
        NSString *message = NSLocalizedString(@"To configure your device as a beacon", @"");
        NSString *cancelButtonTitle = NSLocalizedString(@"OK", @"Cancel button title in configuration Save Changes");
        UIAlertView *errorAlert = [[UIAlertView alloc] initWithTitle:title message:message delegate:self cancelButtonTitle:cancelButtonTitle
            otherButtonTitles:nil];
        [errorAlert show];

        return;
    }

    [peripheralManager stopAdvertising];
    self.infoLabel.text = @"Stoped...";

    if(self.enabled)
    {
        NSDictionary *peripheralData = nil;
        NSUUID *uid = [[NSUUID alloc] initWithUUIDString:uuid];

        region = [[CLBeaconRegion alloc] initWithProximityUUID:uid identifier:beaconId];
        peripheralData =  [region peripheralDataWithMeasuredPower:@-59];
        if(peripheralData)
        {
            [peripheralManager startAdvertising:peripheralData];
            self.infoLabel.text = @"Transmiting...";
        }
    }
}
```

O stavu vysílání je uživatel informovám přepisováním informačního Labelu (infoLabel).

```
#import <UIKit/UIKit.h>

@interface FirstViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *infoLabel;

@end
```

Nakonec pokud uživatel nechce vysílat, ale vyhledávat vysílače tak přepnutím na jíný View Controller je vysílání zastaveno zavoláním metody viewDidDisappear.

```
-(void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];

    peripheralManager.delegate = nil;
    [peripheralManager stopAdvertising];
    self.infoLabel.text = @"Stoped...";}
```

7.4.2 Second View Controller

U přijímače stačí importovat pouze CoreLocation framework, který se využije pro uložení lokace kde byl vysílač nalezen.

```
#import "SecondViewController.h"
#import <CoreLocation/CoreLocation.h>
```

Při načtení Second View Controller se zaregistrují notifikace pro aplikaci. Zde jsou použité notifikace všech tří typů - Badge, Sound, Alert. Tyto typy jsou uloženy do nastavení notifikací a jsou zaregistrovány. Registrace se projeví tak, že je uživatel dotázán, zda povoluje notifikace pro tuto aplikaci.

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    UIUserNotificationType type = UIUserNotificationTypeBadge | UIUserNotificationTypeSound | UIUserNotificationTypeAlert;

    UIUserNotificationSettings *mySettings = [UIUserNotificationSettings settingsForTypes:type categories:nil];

    [[UIApplication sharedApplication] registerUserNotificationSettings:mySettings];
}
```

Metoda switchUpdate kontroluje stav přepínače pro zapnutí a vypnutí vysílání. Pokud se vypínač zapne je zavolána metoda UpdateMonitorRegion.

```
- (IBAction)switchUpdate:(UISwitch *)sender {
    self.enable = sender.on;

    [self updateMonitorRegion];
}
```

Metoda UpdateMonitorRegion jako první alokuje NSUUID pomocí našeho uuid, které je staticky uloženo do NSString. Dále je alokován CLBeaconRegion, který určuje pro jaký region s konkrétním uuid a identifikátorem je vyhledávání určeno. Toto je velmi dobré pokud máme více vysílačů, ale chceme přijímat pouze určitý vysílač a ostatní nevyhledávat. Je možné je podle rozdílných regionů určit.

```
-(void)updateMonitorRegion
{
    NSUUID *uid = [[NSUUID alloc] initWithUUIDString:uuid];
    CLBeaconRegion *region = [[CLBeaconRegion alloc] initWithProximityUUID:uid identifier:beaconId];

    if (region != nil) {
        [self.locationManager stopMonitoringForRegion:region];
    }
```

V této metodě je kontrolován přepínač, a pokud je zapnutý, alokuje se locationManager, který využívá GPS lokaci. Dále je nutné zeptat se uživatele zda povoluje využívání polohových služeb a poté je možné spustit GPS lokaci pro určitý region a spustit vyhledávání pomocí Bluetooth 4.0.

```
if (self.enable) {
    if (region) {

        self.locationManager = [[CLLocationManager alloc] init];
        self.locationManager.delegate = self;
        [self.locationManager requestAlwaysAuthorization];
        [self.locationManager startMonitoringForRegion:region];
        self.infoLabel.text = @"Searching...";
        [self.locationManager startRangingBeaconsInRegion:region];
    }
}
```

Pokud vyhledávání vypínačem vypneme je vyhledávání ukončeno a hodnoty resetovány.

```
else
{
    [self.locationManager stopMonitoringForRegion:region];
    self.infoLabel.text = @"Stoped...";

    self.rangeLabel.text = @"Not searching";
    self.metersLabel.text = @"0.0m";
    self.dbLabel.text = @"0dB";

    [self.locationManager stopRangingBeaconsInRegion:region];

    region = nil;
}
```

Další metoda didEnterRegion, je metoda patřící do knihovny CoreLocation. Je volána pokud Bluetooth nalezne námi vyhledávaný vysílač a vytvoří notifikaci, která informuje uživatele o dané situaci. Automaticky se uloží GPS souřadnice kde byl vysílač nalezen.

```
-(void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion *)region
{
    UILocalNotification *findNotification = [[UILocalNotification alloc] init];
    findNotification.alertBody = @"You are near the beacon!";
    findNotification.soundName = UILocalNotificationDefaultSoundName;

    [[UIApplication sharedApplication] scheduleLocalNotification:findNotification];
}
```

Další podobná metoda didExitRegion, reaguje na odchod z místa kde byl nalezen signál. Uživatel je opět informován pomocí notifikace o změně stavu a hodnoty jsou resetovány.

```
-(void)locationManager:(CLLocationManager *)manager didExitRegion:(CLRegion *)region
{
    UILocalNotification *lostNotification = [[UILocalNotification alloc] init];
    lostNotification.alertBody = @"You lost your beacon!";
    lostNotification.soundName = UILocalNotificationDefaultSoundName;

    [[UIApplication sharedApplication] scheduleLocalNotification:lostNotification];

    self.infoLabel.text = @"Lost...";
    self.rangeLabel.text = @"Not in range";
    self.metersLabel.text = @"0.0m";
    self.dbLabel.text = @"0dB";
}
```

Metoda didRangeBeacons se prochází stále dokola při zapnutí vyhledávání startRangingBeaconsInRegion. Pokud není žádný dostupný vysílač, tak je uživatel informován na displeji, že aplikace vyhledává vysílače a hodnoty jsou resetovány.

```
-(void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray *)beacons inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] == 0)
    {
        self.infoLabel.text = @"Searching...";
        self.rangeLabel.text = @"Not in range";
        self.metersLabel.text = @"0.0m";
        self.dbLabel.text = @"0dB";
        self.infoButton.tintColor = [UIColor redColor];
        return;
    }
```

Když aplikace najde vysílač, je uživatel upozorněn textem na displeji a vyhodnocují se hodnoty pro informování uživatele na displeji.

Zde mohou být čtyři hodnoty:

- Unkown - je stav, při kterém není dostupná žádná informace o vysílači.
- Far - je stav, kdy je signál vysílače velmi slabý a jsme od něho vzdálení.
- Near - je stav, kdy signál a vzdálenost od vysílače je tak kolem 50%.
- Immediate - je stav, kdy jsme velmi blízko vysílače - do desítek metrů.

```
self.infoLabel.text = @"Found...";

NSString *message;
UIColor *color;

CLBeacon * beacon = [beacons firstObject];
switch (beacon.proximity) {
    case CLProximityUnknown:
        message = @"Unknown";
        color = [UIColor redColor];
        break;
    case CLProximityFar:
        message = @"Far";
        color = [UIColor yellowColor];
        break;
    case CLProximityNear:
        message = @"Near";
        color = [UIColor greenColor];
        break;
    case CLProximityImmediate:
    default:
        message = @"Immediate";
        color = [UIColor greenColor];
        break;
}
```

Informace o vzdálenosti a síle signálu v dB jsou aktualizovány na displelji a pokud je hodnota proximity stejná aplikace neaktualizuje údaje.

```
self.metersLabel.text = [NSString stringWithFormat:@"%.2fm", beacon.accuracy];
self.dbLabel.text = [NSString stringWithFormat:@"%lddB", beacon.rssi];

if (beacon.proximity != self.previousProximity) {
    [self.rangeLabel setText:message];
    self.previousProximity = beacon.proximity;
    self.infoButton.tintColor = color;
}
```

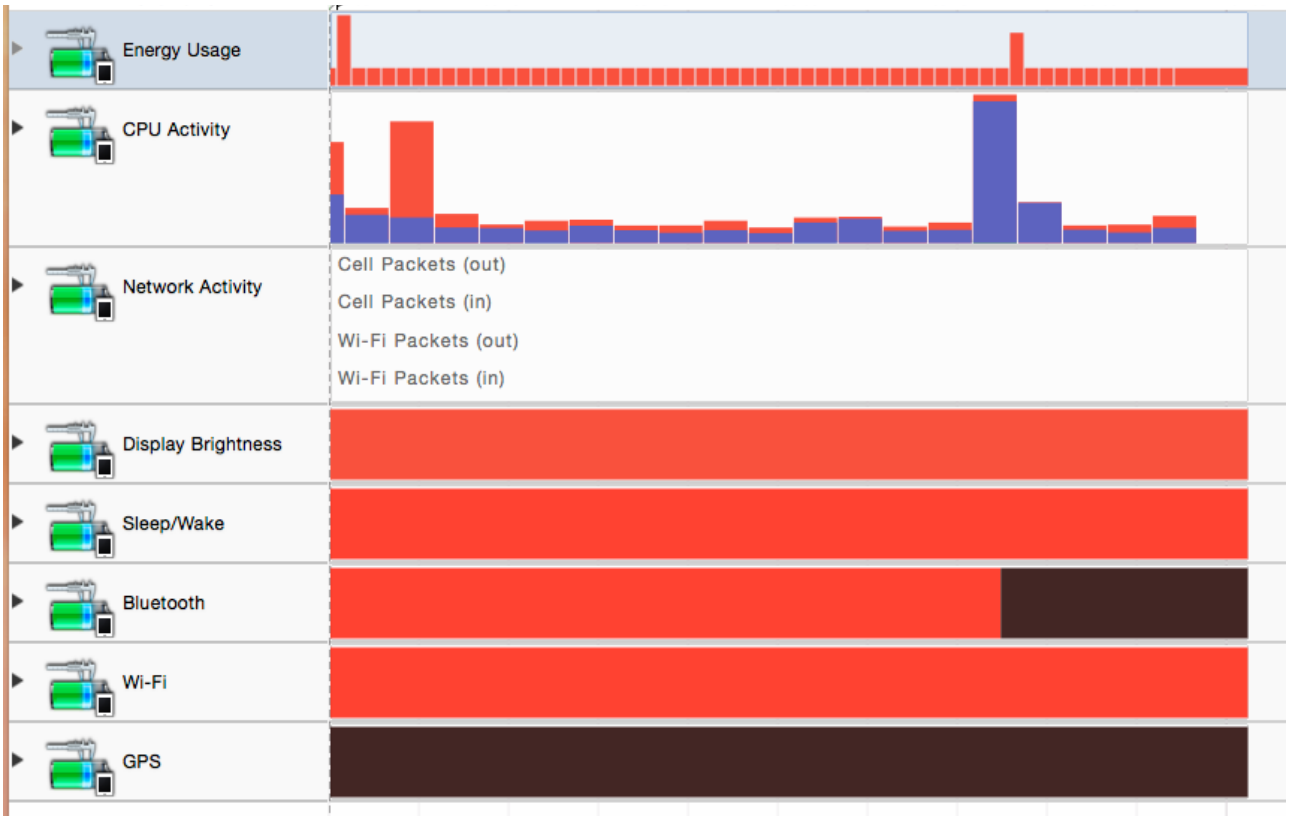
7.4.3. AppDelegate

V této třídě byla přidána pouze jedna metoda starající se o lokální notifikace. Metoda didReceiveLocalNotification se stará o to, aby byl uživatel informován pomocí notifikace i když má aplikaci spuštěnou na popředí. Pokud bude mít uživatel spuštěnou aplikaci a dostane notifikaci o nalezeném či ztraceném vysílači, bude informován vyskakovacím oknem, jinak by notifikace nedorazila.

```
-(void)application:(UIApplication *)application didReceiveLocalNotification:(UILocalNotification *)notification {
    UIAlertView * av = [[UIAlertView alloc] initWithTitle:@"You are near the beacon!"
                                                  message:notification.alertBody
                                                 delegate:NULL
                                        cancelButtonTitle:@"OK"
                                        otherButtonTitles:nil, nil];
    [av show];
}
```

8. Testování

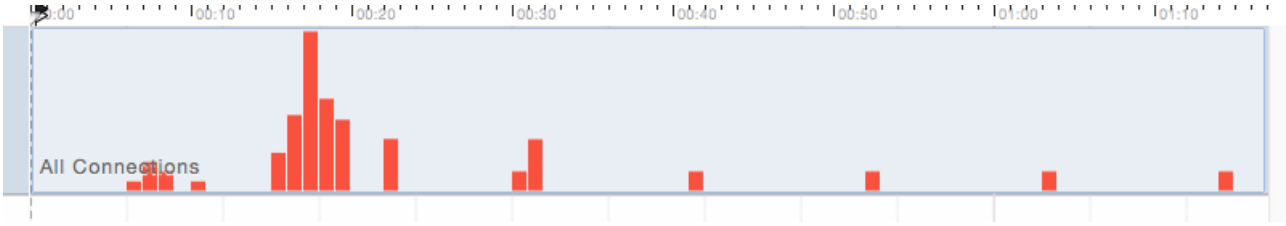
Při testování jsem kladl hlavní důraz na spotřebu baterie, neboť to je hlavní problematika dnešní doby u mobilních zařízení. Testováno bylo jak vysílání pro účely rozesílání například reklamních informací, a také přijímání pro vyhledávání vysílačů.



- Spotřeba energie při spuštěné aplikaci je minimální, a to jak při vysílání, tak přijímání. Dle grafu je vidět, že ke konci byl Bluetooth modul vypnut a spotřeba stoupla

pouze pro účel vypnutí a opět klesla na minimální hodnotu. Spotřeba aplikace se pohybovala kolem 0,1% až 0,5% za hodinu.

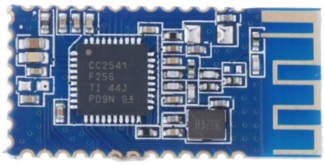
V dalším testu byla data vysílána pomocí Bluetooth 4.0, kde je na grafu vidět, jak aplikace vysílá signál v určitých intervalech, když není připojena na přijímač. V čase od 0:15 do 0:20 byl vysílač připojen na přijímač.



9. Budoucí možné úpravy aplikace

Aplikace je spíše zaměřená na testování komunikace mezi dvěma iOS přístroji. Bylo by možné aplikaci zdokonalit tak, aby byl přijímač schopný číst QR kódy obsahující UUID vysílače a tím by bylo možné přidávat vysílače, které by přijímač mohl vyhledávat. Informace obsažené v notifikacích, odesílaných prostřednictvím vysílače by mohli obsahovat obchodní sdělení a mohly by sloužit jako reklama pro zachytávající přijímače.

Další možností je využít Blouetooth modul - např. HM-10 CC2541,



kdy je možná dostavba na Bluetooth vysílač. Takto dodělaný vysílač má velikost kolem 5 - 10 cm, i s baterií, a je možné mít kontrolu například nad svým majetkem a pokud se od vysílače vzdálíme, bude nás informovat aplikace. Aplikace může také sledovat stav baterie, pokud to modul umožňuje.

10. Závěr

Využitím této konektivity pomocí bluetooth 4.0 a rozesíláním notifikací se otevírá možnost, jak nově rozesílat cílenou reklamu svým zákazníkům. Dále je možnost rozšíření pomocí hardwaru, který je soběstačný a rozesílá informace o své poloze a stavu a uživatel může být informován při vzdálení se od vysílače.

Tato práce měla za cíl napsat aplikaci využívající bluetooth pro komunikaci mezi dvěma iOS zařízeními. Aplikace informuje uživatele o svém stavu a umožňuje uživateli zvolit si, zda chce aplikaci využít jako vysílač, nebo jako přijímač. Aplikace je univerzální a je možné jí využít jak na iPhonu, tak na iPadu. Omezení je nutnost, aby přístroj vlastnil technologii bluetooth 4.0 a minimální verzi iOS 8. Aplikace není náročná na procesor ani na využití baterie.

Tato aplikace je dobrá pro porozumění funkčnosti komunikace pomocí bluetooth a využívání lokálních notifikací. Je možné pozdější rozšíření tohoto zdrojového kódu pro přídání více funkcí.

Seznam použité literatury

- KOCHAN, Stephen G. *Programming in Objective-C 2.0* 2. vyd. Pearson Education 2008. 2. ISBN 0321605543

- HEGARTY, Paul. *iPad and iPhone Application Development.* 2011 Stanford. [iTunes U]

https://itunes.apple.com/cz/itunes-u/ipad-iphone-application-development/id473757255?mt=10

Internetové zdroje:

- Core Bluetooth Programming Guide [on-line] 18.09.2013 [cit. 21.04.2015] odkaz:

https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html

- Core Location Framework Reference [on-line] 18.09.2013 [cit. 21.04.2015] odkaz:

https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/

- iBeacon for Developers [on-line] 2015 [cit. 21.04.2015] odkaz:

https://developer.apple.com/ibeacon/

Seznam příloh

Příloha 1: Zdrojový kód aplikace

Přílohy k bakalářské práci

Zdrojový kód aplikace

```
//  AppDelegate.h
//  Find Me
//
//  Created by Tomas Moravec on 24.03.15.
//  Copyright (c) 2015 Tomas Moravec. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;


@end


//
//  AppDelegate.m
//  Find Me
//
//  Created by Tomas Moravec on 24.03.15.
//  Copyright (c) 2015 Tomas Moravec. All rights reserved.
//

#import "AppDelegate.h"
#import <CoreLocation/CoreLocation.h>

@interface AppDelegate () <UIApplicationDelegate,
CLLocationManagerDelegate>

@property CLLocationManager *locationManager;

@end

@implementation AppDelegate


- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to
inactive state. This can occur for certain types of temporary
interruptions (such as an incoming phone call or SMS message) or when
the user quits the application and it begins the transition to the
background state.
```

```
    // Use this method to pause ongoing tasks, disable timers, and
throttle down OpenGL ES frame rates. Games should use this method to
pause the game.
}

- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data,
invalidate timers, and store enough application state information to
restore your application to its current state in case it is terminated
later.
    // If your application supports background execution, this method is
called instead of applicationWillTerminate: when the user quits.
}

- (void)applicationWillEnterForeground:(UIApplication *)application {
    // Called as part of the transition from the background to the
inactive state; here you can undo many of the changes made on entering
the background.
}

- (void)applicationDidBecomeActive:(UIApplication *)application {
    // Restart any tasks that were paused (or not yet started) while the
application was inactive. If the application was previously in the
background, optionally refresh the user interface.
    application.applicationIconBadgeNumber = 0;
}

- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if
appropriate. See also applicationDidEnterBackground:.
}

-(void)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification {
    UIAlertView * av = [[UIAlertView alloc] initWithTitle:@"You are near
the beacon!"

message:notification.alertBody
                                                    delegate:NULL
                                          cancelButtonTitle:@"OK"
                                          otherButtonTitles:nil, nil];
    [av show];
}
@end
```

```
//
//  FirstViewController.h
//  Find Me
//
//  Created by Tomas Moravec on 24.03.15.
//  Copyright (c) 2015 Tomas Moravec. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface FirstViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *infoLabel;

@end

//
//  FirstViewController.m
//  Find Me
//
//  Created by Tomas Moravec on 24.03.15.
//  Copyright (c) 2015 Tomas Moravec. All rights reserved.
//

#import "FirstViewController.h"
#import <CoreLocation/CoreLocation.h>
#import <CoreBluetooth/CoreBluetooth.h>

CBPeripheralManager *peripheralManager = nil;
CLBeaconRegion *region = nil;



@interface FirstViewController () <CBPeripheralManagerDelegate,
UIAlertViewDelegate, UITextFieldDelegate>

@property (strong, nonatomic) IBOutlet UISwitch *enabledSwitch;
@property BOOL enabled;

-(void)updateAdvertisedRegion;

@end

@implementation FirstViewController

static NSString *uuid = @"1C4B9AF2-8DA6-41B1-87CD-2AC288BCA266";
static NSString *beaconId = @"com.morysoft.findme";

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a
nib.
}

-(void)viewDidAppear:(BOOL)animated
```

```
{
    [super viewDidAppear:animated];

    if (!peripheralManager)
    {
        peripheralManager = [[CBPeripheralManager alloc]
initWithDelegate:self
queue:dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0)];
    }
    else
    {
        peripheralManager.delegate = self;
    }
}

-(void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];

    peripheralManager.delegate = nil;
    [peripheralManager stopAdvertising];
    self.infoLabel.text = @"Stoped...";}

-(void)peripheralManagerDidUpdateState:(CBPeripheralManager *)peripheral
{

}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)switchUpdate:(UISwitch *)sender {
    self.enabled = sender.on;
    [self updateAdvertisedRegion];
}


-(void)updateAdvertisedRegion
{
    if (peripheralManager.state < CBPeripheralManagerStatePoweredOn)
    {
        NSString *title = NSLocalizedString(@"Bluetooth must be
enabled", @"");
        NSString *message = NSLocalizedString(@"To configure your device
as a beacon", @"");
        NSString *cancelButtonTitle = NSLocalizedString(@"OK", @"Cancel
button title in configuration Save Changes");
        UIAlertView *errorAlert = [[UIAlertView alloc]
initWithTitle:title message:message delegate:self
cancelButtonTitle:cancelButtonTitle otherButtonTitles:nil];
        [errorAlert show];

        return;
    }
```

```
    [peripheralManager stopAdvertising];
    self.infoLabel.text = @"Stoped...";

    if(self.enabled)
    {
        NSDictionary *peripheralData = nil;
        NSUUID *uid = [[NSUUID alloc] initWithUUIDString:uuid];

        region = [[CLBeaconRegion alloc] initWithProximityUUID:uid
identifier:beaconId];
        peripheralData =  [region peripheralDataWithMeasuredPower:@-59];
        if(peripheralData)
        {
            [peripheralManager startAdvertising:peripheralData];
            self.infoLabel.text = @"Transmiting...";
        }
    }
}

@end

//
//  SecondViewController.h
//  Find Me
//
//  Created by Tomas Moravec on 24.03.15.
//  Copyright (c) 2015 Tomas Moravec. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface SecondViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *infoLabel;

@property (weak, nonatomic) IBOutlet UILabel *rangeLabel;
@property (weak, nonatomic) IBOutlet UILabel *metersLabel;
@property (weak, nonatomic) IBOutlet UILabel *dbLabel;
@property (weak, nonatomic) IBOutlet UIButton *infoButton;

@end

//
//  SecondViewController.m
//  Find Me
//
//  Created by Tomas Moravec on 24.03.15.
//  Copyright (c) 2015 Tomas Moravec. All rights reserved.
//

#import "SecondViewController.h"
#import <CoreLocation/CoreLocation.h>

@interface SecondViewController () <CLLocationManagerDelegate>
```

```
@property (weak, nonatomic) IBOutlet UISwitch *enableSwitch;

@property BOOL enable;
@property CLProximity previousProximity;
@property (nonatomic) CLLocationManager *locationManager;

-(void)updateMonitorRegion;

@end

@implementation SecondViewController

static NSString *uuid = @"1C4B9AF2-8DA6-41B1-87CD-2AC288BCA266";
static NSString *beaconId = @"com.morysoft.findme";

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a
nib.

    UIUserNotificationType type = UIUserNotificationTypeBadge |
UIUserNotificationTypeSound | UIUserNotificationTypeAlert;

    UIUserNotificationSettings *mySettings = [UIUserNotificationSettings
settingsForTypes:type categories:nil];

    [[UIApplication sharedApplication]
registerUserNotificationSettings:mySettings];
}


- (IBAction)switchUpdate:(UISwitch *)sender {
    self.enable = sender.on;

    [self updateMonitorRegion];
}


-(void)updateMonitorRegion
{
    NSUUID *uid = [[NSUUID alloc] initWithUUIDString:uuid];
    CLBeaconRegion *region = [[CLBeaconRegion alloc]
initWithProximityUUID:uid identifier:beaconId];

    if (region != nil) {
        [self.locationManager stopMonitoringForRegion:region];
    }

    if (self.enable) {
        if (region) {

            self.locationManager = [[CLLocationManager alloc] init];
            self.locationManager.delegate = self;
            [self.locationManager requestAlwaysAuthorization];
            [self.locationManager startMonitoringForRegion:region];
            self.infoLabel.text = @"Searching...";
```

```
            [self.locationManager startRangingBeaconsInRegion:region];
        }
    }
    else
    {
        [self.locationManager stopMonitoringForRegion:region];
        self.infoLabel.text = @"Stoped...";

        self.rangeLabel.text = @"Not searching";
        self.metersLabel.text = @"0.0m";
        self.dbLabel.text = @"0dB";

        [self.locationManager stopRangingBeaconsInRegion:region];

        region = nil;
    }

}

-(void)locationManager:(CLLocationManager *)manager didEnterRegion:
(CLRegion *)region
{
    UILocalNotification *findNotification = [[UILocalNotification alloc]
init];
    findNotification.alertBody = @"You are near the beacon!";
    findNotification.soundName = UILocalNotificationDefaultSoundName;

    [[UIApplication sharedApplication]
scheduleLocalNotification:findNotification];
}

-(void)locationManager:(CLLocationManager *)manager didExitRegion:
(CLRegion *)region
{
    UILocalNotification *lostNotification = [[UILocalNotification alloc]
init];
    lostNotification.alertBody = @"You lost your beacon!";
    lostNotification.soundName = UILocalNotificationDefaultSoundName;

    [[UIApplication sharedApplication]
scheduleLocalNotification:lostNotification];

    self.infoLabel.text = @"Lost...";
    self.rangeLabel.text = @"Not in range";
    self.metersLabel.text = @"0.0m";
    self.dbLabel.text = @"0dB";
}

-(void)locationManager:(CLLocationManager *)manager didRangeBeacons:
(NSArray *)beacons inRegion:(CLBeaconRegion *)region
{
    if ([beacons count] == 0)
    {
        self.infoLabel.text = @"Searching...";
        self.rangeLabel.text = @"Not in range";
        self.metersLabel.text = @"0.0m";
```

```
        self.dbLabel.text = @"0dB";
        self.infoButton.tintColor = [UIColor redColor];
        return;
    }

    self.infoLabel.text = @"Found...";

    NSString *message;
    UIColor *color;

    CLBeacon * beacon = [beacons firstObject];
    switch (beacon.proximity) {
        case CLProximityUnknown:
            message = @"Unknown";
            color = [UIColor redColor];
            break;
        case CLProximityFar:
            message = @"Far";
            color = [UIColor yellowColor];
            break;
        case CLProximityNear:
            message = @"Near";
            color = [UIColor greenColor];
            break;
        case CLProximityImmediate:
        default:
            message = @"Immediate";
            color = [UIColor greenColor];
            break;
    }

    self.metersLabel.text = [NSString stringWithFormat:@"%.2fm",
beacon.accuracy];
    self.dbLabel.text = [NSString stringWithFormat:@"%lddB",
beacon.rssi];

    if (beacon.proximity != self.previousProximity) {
        [self.rangeLabel setText:message];
        self.previousProximity = beacon.proximity;
        self.infoButton.tintColor = color;
    }
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```