Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Využití Apache Cordova k práci s Photon Particle

Bakalářská práce

František Mastil

Školitel: PhDr. Milan Novák, Ph.D.

České Budějovice 2017

Bibliografické údaje

Mastil, F., 2017: Využití Apache Cordova k práci s Photon Particle. [Utilization of Apache Cordova to work with Photon Particle. Bc. Thesis, in Czech] – 80 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Cílem bakalářské práce je vytvoření metodiky k tvorbě aplikací prostřednictvím Apache Cordova (Ionic 2). Dále zprovoznění lokálního cloudu, na kterém bude fungovat mikrokontroler Particle Photon. Vytvořené metodiky budou ověřeny na praktických řešení.

Klíčová slova

Ionic 2, Apache Cordova, Particle Photon, Cloud, IoT

Abstract

The goal of the thesis is to create a methodology for making applications via the Apache Cordova (Ionic 2). Further commissioning of the local cloud, on which the microcontroller Particle Photon will operate. Created methodologies will be verified on practical solutions.

Keywords

Ionic 2, Apache Cordova, Particle Photon, Cloud, IoT

Prohlášení

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 13.04.2017

František Mastil

Poděkování

Děkuji PhDr. Milanu Novákovi, Ph.D. za odborný dohled a čas věnovaný vedení této práce. Dále děkuji rodině za podporu při studiu.

Obsah

Úvod

V současné době dochází k velkému rozšíření moderních technologií v běžném životě od sledování denních aktivit přes malé senzory, až po ovládání funkčnosti celých domů využitím vestavěných zařízení s připojením k internetu. Zavádí se termín internet věcí (Internet of Things) IoT.

Do skupiny IoT lze zařadit senzory a další zařízení, které jsou schopny se připojit k internetové síti. Tyto zařízení dokážou sbírat a vyměňovat data. IoT dovoluje vzdáleně kontrolovat a ovládat zařízení přes síť. Tím vytváří mnoho možností pro integraci fyzického světa do počítačové logiky, dovoluje automatizovat činnosti, které dříve vyžadovaly dohled obsluhy, tím usnadňují každodenní život a šetří čas i peníze. Díky neustálému pokroku a vývoji chytrých zařízení dochází ke kombinaci a využití v chytrých domácnostech, inteligentních sítích, dopravě a mnohém dalším.[1]

Internet věcí je považován za další fázi vývoje informační společnosti, který podle odhadů do roku 2020 bude součástí každodenního života. Přesto existuje několik překážek, které mohou rozvoj IoT zpomalit, včetně přechodu na IPv6, které mají společný soubor standardů, a rozvoj energetických zdrojů pro miliony až miliardy nepatrných senzorů.[2]

Práce se zaměřuje na moderní mikrokontroler Photon od výrobce Particle, tento poměrně malý prvek je pro svou dostupnost a snadné použití vhodný pro navrhování prototypů nových chytrých zařízení začínajícími i pokročilými vývojáři.

V současné době jednou z důležitých otázek, kterou si při práci s IoT zařízeními a jejich aplikacemi vývojáři kladou je, jak zařídit co největší rozšíření produktů na trh. To je velice úzce spjato s mobilní platformou, na kterou se vývojáři rozhodnou zaměřit. Tam si konkurují známé systémy iOS, Android, Windows Phone a další.

---

[1] *An Introduction to the Internet of Things (IoT)* [online]. 1. San Francisco, California: Lopez Research, 2013 [cit. 2015-11-21].

[2] EVANS, Dave. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything* [online]. 2011, 2015-11-22, s. 1

Dochází k problémům s náklady na vývoj a s časovou náročností, kterou se snaží každá firma co nejvíce snížit. Možností, jak vyvíjet na více systémů zároveň, je využitím webových šablon (frameworků). Je to způsob vývoje aplikací zapouzdřených jako dynamické webové stránky, které mohou využít nativních funkcí telefonu (geolokaci, kontakty, fotoaparát, souborový systém, multitouch a jiné). Tím se použitelnost a rozšíření aplikací značně zvyšuje.

Na trhu je celá řada frameworků. Práce se bude zabývat Apache Cordovou, která byla vybrána díky svým výhodám nad konkurencí. Těmi nejdůležitějšími je podpora všech dostupných rozšířených operačních systémů (OS) iOS, Android, Windows Phone, Ubuntu. Cordova podporuje i systémy v chytrých televizích jako třeba Amazon FireOS nebo Lg webOS. Dále má již zmíněnou širokou podporu nativních funkcí mobilních zařízení.

Framework je navíc pod licencí Open Source, díky které po dodržení podmínek umožňuje nahlédnout do volně dostupného zdrojového kódu a dále jej upravovat podle svých potřeb.

Cíle Práce

Cílem je popsání metod ke zprovoznění osobního cloudu komunikujícím se zařízením Photon. Dále popsáním metodik a možností tvorby aplikací pro chytré telefony v Apache Cordova využitím frameworku Ionic 2. Pomocí ukázkové aplikace demonstrovat ovládání Photonu připojeného ke cloudu. Práce je rozdělena na 2 části, teoretickou a praktickou.

V teoretické části se práce bude zabývat pojmovým aparátem a bude tvořit nutný teoretický základ týkající se problematiky Internet of Things. Dále budou popsány funkce mikrokontroleru Photon, Apache Cordova a další technologie potřebné k pochopení a zvládnutí problematiky.

V praktické části bude vysvětlen postup instalace, nastavení a programování Photonu. Dále nastavení služeb Particle Cloud a jeho spuštění lokálně. Celé bude demonstrováno na ukázkovém řešení pomocí aplikace vytvořené v Ionic 2, která bude ovládat Photon.

Teoretická východiska

1. Internet of Things

1.1 Pojmy

Podle oficiálního doporučení ITU-T Y.2060 (06/2012) IoT (Internet of Things) říká, že je to globální infrastruktura pro informační společnost, která umožňuje pokročilé služby propojením (fyzické i virtuální) věci, na základě stávajících a vyvíjejících se vzájemně spolupracujících informačních a komunikačních technologii, prostřednictvím sběru dat, zpracováním a komunikačním schopnostem. Internet věcí plně využívá „věcí" pro nabízení aplikací a dohlíží, aby byly splněny požadavky na bezpečnost a soukromí. Zavedený je také termín věci (thing). S ohledem na internet věcí, je to předmět fyzického světa (fyzikální věci) nebo informačního světa (virtuální věci), který je možné rozpoznat a integrovat do komunikační sítě.[3]

Zjednodušeně by se dalo říci, že se jedná o nové odvětví informačních technologií nabízející služby využívající sběru dat a jejich zpracování k řízení činností automaticky nebo na dálku pomocí chytrých telefonů, kde je kladen důraz na bezpečnost a soukromí. Integrace s internetem znamená, že zařízení bude používat IP adresy jako unikátní identifikátor UID. Kvůli omezenému adresnímu prostoru IPv4, který dovoluje 4.3 miliardy unikátních adres, které jsou již vyčerpané, budou zařízení muset využívat IPv6 adres, díky své velké spotřebě.[4]

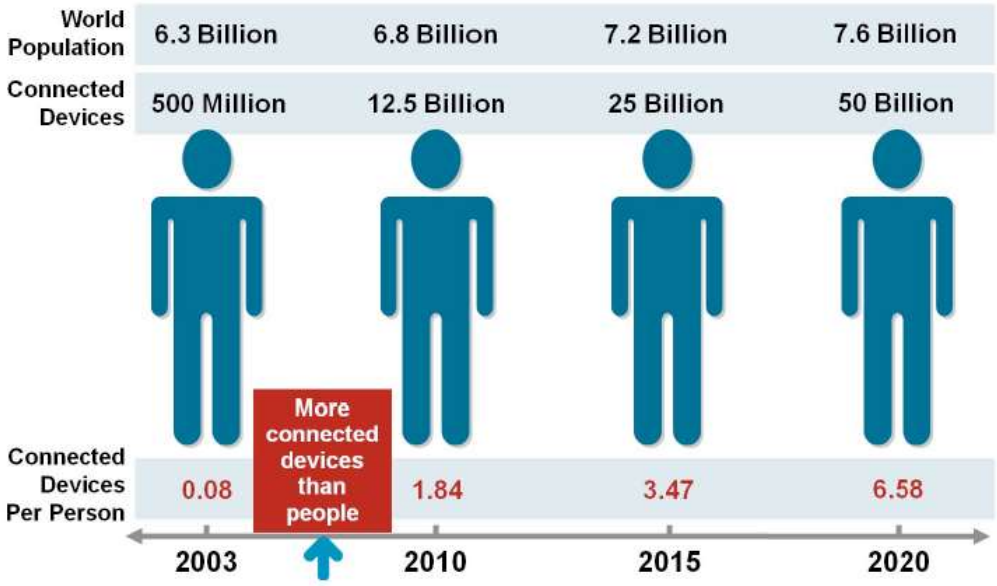
[3] ITU-T Y.2060. *SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS AND NEXT-GENERATION NETWORKS: Next Generation Networks – Frameworks and functional architecture models* [online]. 06/2012. s. 1 [cit. 2016-02-04].

[4] *Stop Using Internet Protocol Version 4!* [online]. CXO Media Inc. a subsidiary of IDG Enterprise, 2014 [cit. 2016-03-10].

1.2 Historie

Podle Cisco Internet Business Solutions Group (IBSG) myšlenka internetu věcí není nová, její původ lze dohledat zpátky v Massachusetts Institute of Technology (MIT) při práci v Auto-ID Center založeném v roce 1999. Tato skupina pracovala v oblasti identifikace na rádiové frekvenci (RFID). Laboratoř se skládala ze 7 vědeckých univerzit napříč čtyřmi kontinenty. Kde spoluzakladatel Auto-ID Center Kevin Ashton upřednostňoval termín Internet věcí.[5]

K zavedení termínu IoT dochází v době, kdy je připojeno více věcí, objektů k internetu než lidí. Z učiněných měření IBSG na obrázku (obr. 1)[5], nastalo mezi roky 2008 a 2009. Na základě toho odhadují, že IoT bude mít téměř 50 miliard objektů do roku 2020.[5]



Obrázek 1 - Vývoj počtu chytrých zařízení

[5] EVANS, Dave. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything* [online]. 2011, s. 1 [cit. 2015-11-22].

1.3 Předpokládaný rozvoj IoT

Internet věcí je další fází informační revoluce. Z historie od vzniku prvních lokálních sítí až po globální internetové sítě. Následný vývoj bezdrátových připojení na lokální a externí úrovni, který umožnil rozmach chytrých telefonů (smartphonů) a tabletů. Na řadu přichází IoT, kdy se k propojení dostanou i objekty dříve hloupé a odpojené. Internet věcí není jen o chytré ledničce, jde o evoluci technologického vývoje internetu, elektroniky a propojení objektů obecně.[6]

Díky propojení chytrých objektů (zařízení) se očekává, že IoT nastolí automatizaci téměř ve všech oblastech a zároveň umožní pokročilé využití, jako jsou chytrá města, které budou schopny automaticky řídit dopravu pomocí chytrých světel končících až u autonomních vozů, využívat chytré parkování, elektronické mýtné a starat se o bezpečnost. Dále také využití v lékařství, jako implantáty k monitorování srdečních činností nebo zařízení k analýze DNA prostředí/jídla/patogenů a následné bezpečností notifikační a informační systémy. IoT se bude používat i při výrobě, kde sníží náklady a umožní rychlou výrobu nových výrobků s dynamickou odezvou na požadavky produktů a v reálném čase bude moci optimalizovat výrobní produkci. Chytré domy budou schopny šetřit energie díky chytrým termostatům a automatizovat denní činnosti, díky schopnosti pracovat na síti s omezeným procesorem, pamětí a zdrojem energie.[7]

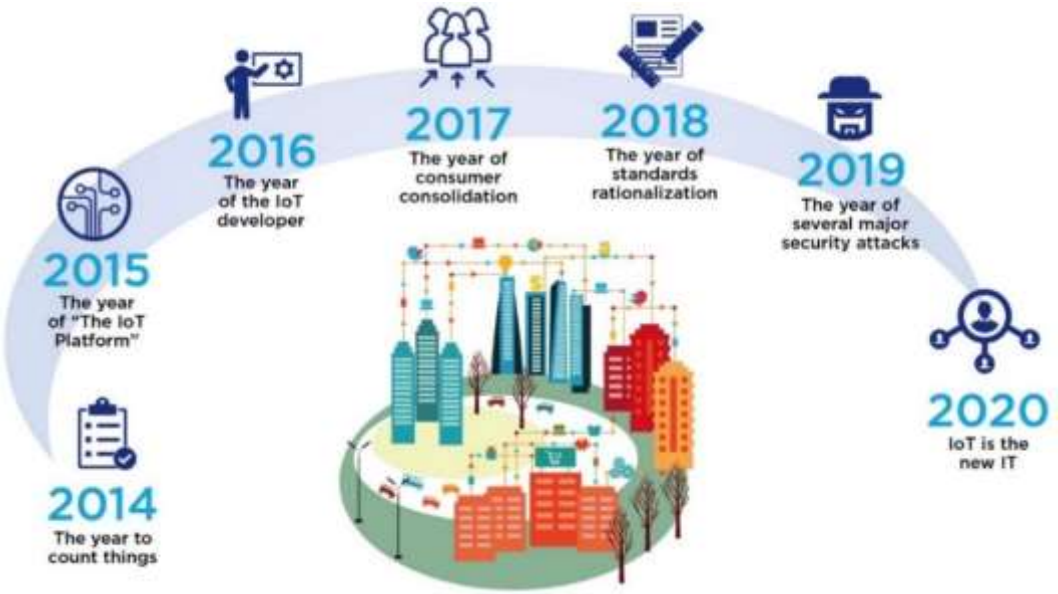
S expanzí automatizace do těchto a dalších sfér se čeká, že bude IoT generovat ohromné množství dat tzv. Big data z různých míst s následnou nutností pro rychlou agregaci dat a rostoucí potřebou data efektněji indexovat, ukládat a zpracovávat v reálném čase.[8]

---

[6] JAVŮREK, Karel. *IDC: internet věcí bude větší trh než smartphony a tablety dohromady* [online]. 1 [cit. 2016-03-28].

[7] M. Ersue, Ed., D. Romascanu, J. Schoenwaelder a A. Sehgal. *Management of Networks with Constrained Devices: Use Cases draft-ietf-opsawg-coman-use-cases-02* [online]. -, 2014 [cit. 2016-03-29].

[8] KELLY, Jeff a David FLOYER. *The Industrial Internet and Big Data Analytics: Opportunities and Challenges.* In: *Wikibon* [online]. -: Wikibon, 2013-9-16 [cit. 2016-11-11].



Obrázek 2 - Odhad vývoje IoT dle IDC

Podle infografiky (obr. 2)[9], kterou zveřejnila společnost IDC už IoT prošel dvěma fázemi, vznikem nových platforem a nyní je v bodě, kdy se objevuje spoustu nových startupů, firem a vývojářů, kteří chtějí na tomto nadějném trhu chytnout správnou vlnu a přicházet s inovativními přístupy, kde všude by se IoT dal využít a jeho realizací.

Rok 2017 a 2018 by měl být rokem ustálení standardů. V roce 2019, se očekává, že už budou chytré věci součástí běžného života, dojde k růstu útoků na zařízení a hlavní otázkou bude, jak zvýšit jejich zabezpečení (již teď velmi aktuální).

V roce 2020 by IoT měl představovat nový směr celé informační technologie a mělo by dojít k očekávanému využití ve všech denních situacích.

[9] Odhad vývoje IoT. In: IDC [online]. IDC Research, Inc.: IDC, 2016 [cit. 2016-03-28].

2. Particle Photon

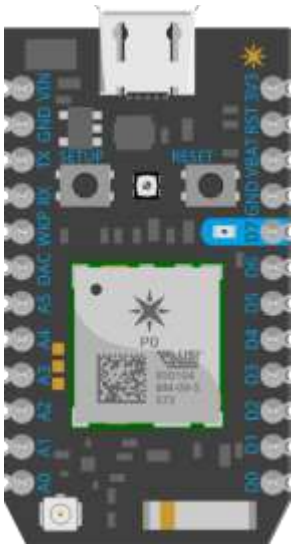
V této části bude popsán mikrokontroler Photon, který je ukázkovým příkladem nové generace zařízení internetu věcí.

K jeho výhodám patří možnost rychlého navrhování chytrých zařízení. Kód lze psát i z webového prostředí (IDE), a nahrát jej na Photon přes Wi-Fi s připojeným Photonem k internetu. Výhodou je relativně snadný začátek práce se zařízením, které je vhodné i pro méně zkušené vývojáře.

2.1 Základní vlastnosti

Photon (obr. 3)[10] je založený na ARM Cortex M3 mikrokontroleru s vestavěným Broadcom Wi-Fi čipem v modulu nazvaném P0 (P-zero).

Rádiová Frekvence (RF) a komponenty pro uživatelské rozhraní k P0 se nachází na malém plošném spoji (PCB) nazvaném Photon. Návrh je dostupný veřejnosti (open source), která jej může dále prohlížet a upravovat.[10]



Obrázek 3 - Photon

[10] Photon Datasheet. Docs.particle.io [online]. San Francisco: Particle, 2016 [cit. 2016-04-05].

Velkou výhodou oproti konkurenci je velikost zařízení (2 cm šířka a 3,5 cm výška), Photon lze koupit ve variantě s piny i bez. Je možné koupit i samostatný modul P0 (obr. 4) se stejnými parametry.

Obrázek 4 - P0 modul

Parametry Particle Photon

- Particle PØ Wi-Fi modul
- Broadcom BCM43362 Wi-Fi čip
- 802.11b/g/n Wi-Fi
- STM32F205RGY6 120Mhz ARM Cortex M3
- 1MB Flash, 128KB RAM
- 18 Mixed-signal GPIO a rozhraní
- Real-time operating system (FreeRTOS)

Particle je vhodná platforma pro full-stack (starající se o celý proces) IoT vývojáře umožňující vše, co je potřeba k vytvoření a uvedení nového produktu na trh. Začínající od zmíněného zařízení fungujícím na bezpečném Particle cloudu a aplikacemi pro veškerou správu.

2.2 Architektura

V této části je popsána architektura a možnosti Photonu. Jedná se o doplnění informací k zařízení. V rámci práce se dále probírat nebudou.

Zdroj energie

Přívod elektřiny je dodáván desce přes USB Micro B konektor nebo přes VIN pin určený pro externí energetický zdroj. Pokud je elektřina přivedena přímo přes VIN pin, napětí by se mělo pohybovat mezi 3,6VDC a 5,5VDC. Pokud je Photon napájen přes USB port, z VINu jde výstupní

napětí přibližně 4,8VDC díky ochraně proti přepólování pomocí Schottkyho diodě mezi V+ USB a VIN. Pokud je využit, jako výstup, nejvyšší zatížení na VIN je 1A. Běžný odběr proudu je 80mA s 5V vstupem. V klidovém režimu je proud 160uA.[11]

Radiová frekvence

Přívod rádiové frekvence (RF) vede z P0 modulu do SPDT RF switche. Řídící linka na P0 modulu vybere, který ze dvou portů RF switche je připojen na přívodné vedení RF. 100pF oddělovací kapacitor se nachází na každém z ovládacího vedení. Jeden port je připojen k PCB keramické anténě a druhý je připojen k u.FL konektoru pro vnější přizpůsobení antény. Defaultní port bude nastaven na čipu antény.

Uživatelské rozhraní nabízí přepnutí mezi vnitřním, vnější a automatickým módem, který nepřetržitě přepíná mezi oběma anténami a vybere nejlepší signál. Všechny 3 RF porty na RF switchi mají v sérii 10pF blokovací kondenzátor stejnosměrného proudu. Efektivně přenáší 2,4GHz frekvence, zatímco blokuje nežádoucí stejnosměrné napětí, které by mohlo poškodit RF-switch. Všechny RF trasy jsou navrženy jako malá přenosová vedení, která mají 50 ohmovou impedanci. Čipová anténa odpovídá impedanci 50 ohmů na přívodném vedení RF přes Pi síť, skládá se ze tří RF cívek (1 series, 2 shunt). Tyto hodnoty jsou zcela specifické pro Photon vzhledem k PCB konstrukci a uspořádání RF částí. [11]

[11] Photon Datasheet. *Docs.particle.io* [online]. San Francisco: Particle, 2016 [cit. 2016-04-05].

Periferie

V tabulce 1 lze vidět, které typy periferií Photon obsahuje, jejich počet, napětí a jestli je pin vstupně výstupní.

Typ periferie	**Počet**	**Vstup / Výstup**	**FT[12]/ 3V3[13]**
Digital	18	I/O	FT/3V3
Analog (ADC)	8	I	3V3
Analog (DAC)	2	O	3V3
SPI	2	I/O	3V3
I2S	1	I/O	3V3
I2C	1	I/O	FT
CAN	1	I/O	FT
USB	1	I/O	3V3
PWM	9[3]	O	3V3

Tabulka 1 - Typy periferií Photonu

Všechny piny kromě A3 a DAC tolerují 5V (pokud nejsou v analogovém módu). Pokud jsou použity jako 5V vstup pull-up/pull-down rezistor musí být vypnutý.

PWM[14] je dostupné na D0, D1, D2, D3, A4, A5, WKP, RX, TX s varováním: PWM časovač je duplikován na 2 piny (A5/D2) a (A4/D3) pro 7 nezávislých PWM výstupů. Například: PWM může být použito na A5, zatímco D2 se použije jako GPIO, nebo D2 jako PWM, zatímco A5 se použije jako analogový vstup. A5 a D2 nemohou být zároveň použity jako nezávisle kontrolované PWM výstupy. [15]

[12] FT = 5V tolerantní piny

[13] 3V3 = 3,3V max

[14] Pulzně šířková modulace

[15] Photon Datasheet. Docs.particle.io [online]. San Francisco: Particle, 2016 [cit. 2016-04-05].

JTAG

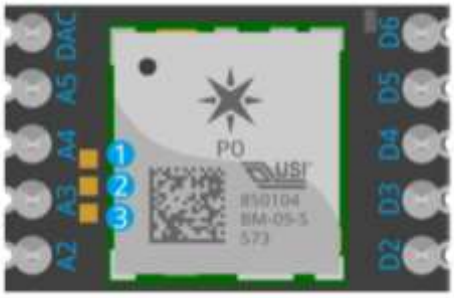
Pin D3 až D7 jsou rozhraní pinů JTAG (tab. 2). Mohou být využity k přeprogramování bootloaderu nebo uživatelského firmwaru se standardními JTAG nástroji jako je ST-Link v2, J-Link, R-Link, OLIMEX ARM-USB-TINI-H a také FTDI JTAG.

Defaultní stav – po resetu na krátkou dobu před tím, než se tyto piny vrátí do GPIO (pokud JTAG debugging není vyžadován tzn. USE_SWD_JTAG=y není specifikováno v příkazové řádce.)

Photon Pin	**Popis**	**STM32 Pin**	**PØ Pin**	**PØ Název Pinu**	**Defaultní stav**
D7	JTAG_TMS	PA13	44	MICRO_JTAG_TMS	~40k pull-up
D6	JTAG_TCK	PA14	40	MICRO_JTAG_TCK	~40k pull-down
D5	JTAG_TDI	PA15	43	MICRO_JTAG_TDI	~40k pull-up
D4	JTAG_TDO	PB3	41	MICRO_JTAG_TDO	Floating
D3	JTAG_TRST	PB4	42	MICRO_JTAG_TRSTN	~40k pull-up
3V3	Napájení				
GND	Uzemnění				
RST	Reset				

Tabulka 2 - Rozhraní JTAG pinů

Photon podporuje spolupráci s Bluetooth a dalšími externími rádii pomocí tří zlatých destiček na horní straně PCB v blízkosti pinu A3. Tyto destičky (obr. 5)[16] jsou 0,09mm^2 velké, rozmístěné 0,12mm od sebe. Tento odstup podporuje možnost připínání na malém 1,25mm - 1,27mm pitch 3-pin male header, aby bylo o něco jednodušší se propojit.



Obrázek 5 - Umístění destiček na desce

Když jsou dvě rádia ve stejném frekvenčním pásmu používají se ve stejném systému, jako je Wi-Fi a Bluetooth, společné rozhraní lze použít ke koordinaci vysílací činnosti, k zajištění optimálního výkonu a v řízení konfliktů mezi těmito rádii.

Když jsou tyto destičky (tab. 3) naprogramovány ke spolupráci s Bluetooth, jsou nastaveny na vysokou impedanci při zapnutí a resetu. Alternativně mohou být individuálně naprogramovány tak, aby šli použít jako GPIO pomocí softwarového ovládání. Také mohou být naprogramovány tak, aby měli vnitřní pull-up nebo pull-down odpor.[16]

Destička#	**PØ Pin Název**	**PØ Pin#**	**I/O**	**Popis**
1	BTCX_RF_ACTIVE	9	I	Bluetooth je aktivní
2	BTCX_STATUS	10	I	Signalizuje prioritu Bluetooth a TX/RX
3	BTCX_TXCONF	11	O	Výstup povolující Bluetooth TX

Tabulka 3 - Popis destiček Photonu

[16] Photon Datasheet. Docs.particle.io [online]. San Francisco: Particle, 2016 [cit. 2016-04-05].

2.2.1 Popis pinů a tlačítek

V tabulce je možné přečíst si popis všech pinů nacházejících se na Photonu (tab. 4).

Pin	Popis
VIN	Tento pin může být použit jako vstup nebo výstup. Jako vstup vyžaduje 3,6 - 5,5VDC k napájení Photonu. Pokud je použit jako výstup, maximální zatížení VIN je 1A.
RST	Resetovací vstup. Obvod na desce obsahuje 1k ohm pull-up rezistor mezi RST a 3v3 a 0,1uF a kapacitor mezi RST a GND.
VBAT	Energie do vnitřního RTC, zálohování registrů a SRAM, když 3V3 není přítomna (1,65 - 3,6VDC).
3V3	Pin je výstup regulátoru a je vnitřně připojen k VDD WiFi modulu. Při napájení Photonu přes VIN nebo USB port, pin bude mít 3,3VDC výstupní napětí. Může být také použit k napájení (max. vstupní 3,3VDC). Při použití jako výstup, maximální zatížení na 3V3 je 100mA. Poznámka: Při napájení Photonu přes tento pin, musí být energie odpojena od VIN a USB.
WKP	Active-high wakeup pin, probudí modul ze spánku/pohotovostního režimu. Není-li použit jako WAKEUP tento pin lze použít také jako digitální GPIO, ADC vstup nebo PWM.
D0~D7	Pouze digitální GPIO piny.
A0~A7	12-bit Analog-to-Digital (A/D) vstupy (0-4095), a také digitální GPIO. A6 a A7 piny ve skutečnosti nejsou označeny jako takové, ale může se použít kód, jako je analogRead(A7). A6 namapovat k pinu DAC a namapovat A7 na pin WKP.
DAC	12-bit Digital-to-Analog (D/A) výstup (0 až 4095), a také digitální GPIO. DAC se používá jako DAC nebo DAC1 v softwaru, a A3 je druhý DAC výstup. Používá se jako DAC2 v softwaru.
RX	Primárně používán jako UART RX, ale může být také použit jako digitální GPIO nebo PWM.

TX	Primárně používán jako UART TX, ale může být také použit jako digitální GPIO nebo PWM.

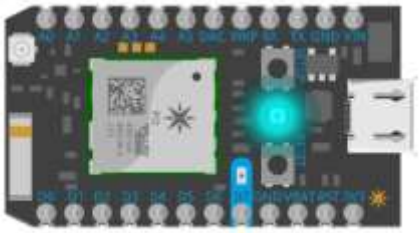
Tabulka 4 - Popis pinů

Dále je důležité zmínit nastavení dvou režimů zařízení, které se budou v práci používat.

2.3 Režimy Photonu

2.3.1 Připojený k internetu

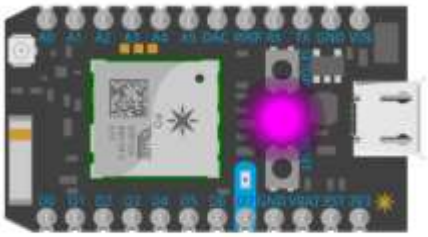
Pokud Photon pomalu pulzuje tyrkysovou barvou, značí tím, že je připojený k internetu a může komunikovat s cloudem. Na Photonu je možné volat funkce nebo do něj nahrát zdrojový kód (obr. 6).



Obrázek 6 - Připojený Photon

2.3.2 Aktualizace firmwaru

Pokud Photon světélkuje purpurově, v daném momentě nahrává zdrojový kód nebo aktualizaci zařízení (obr. 7).



Obrázek 7 - Aktualizace

2.3.3 Hledání internetového připojení

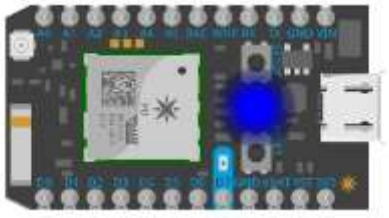
V případě, že Photon rychle světélkuje zeleně, snaží se připojit pomocí zadaných údajů k internetu.

2.3.4 Připojování k Particle cloud

Když je Photon v průběhu připojování k cloudu, zrychleně poblikává tyrkysově. Běžně je tento režim vidět při spouštění zařízení.

2.3.5 Uvedení do Listening mode

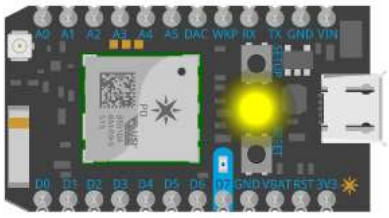
Režim poslechu (Listening mode) se používá pro konfiguraci zařízení. Podržením tlačítka SETUP přibližně 4 vteřiny začne Photon blikat modře signalizuje tím Listening mode (obr. 8).



Obrázek 8 - Režim poslechu

2.3.6 Uvedení do DFU

K nahrání souborů na zařízení, je nejprve nutné uvézt Photon do DFU režimu. Obě tlačítka (SETUP, RESET) je potřeba zároveň stisknou. Po uběhnutí přibližně 4 vteřin se uvolní tlačítko RESET a stále se drží tlačítko SETUP. Photon začne blikat žlutě, čímž signalizuje DFU režim (obr. 9)



Obrázek 9 - DFU režim

2.4 Particle Cli

V této kapitole jsou představeny příkazy programu particle. Dále budou využívány k ovládání a práci s Photonem.

```
particle setup
```

Slouží k přihlášení do vybraného cloudu a nastavení zařízení přes USB nebo Wifi.

```
particle setup wifi
```

Slouží k přidání další Wi-Fi sítě do zařízení připojeného přes USB. Je třeba se ujistit, že je zařízení v listening modu blikající modré.

```
particle login
```

Přihlášení a uložení přístupového tokenu k interakci účtu na cloudu.

```
particle logout
```

Odhlášení a následné zrušení tokenu pro klientskou session.

```
particle list
```

Generuje seznam zařízení, které jsou vlastněny, a zobrazuje informace o jejich stavu, včetně toho, jaké obsahují proměnné a funkce.

```
particle device add ID
```

Přidání nového zařízení. Zařízení je potřeba mít připojené k internetu.

```
particle device remove ID
```

Odstranění zařízení z účtu, aby bylo možné jej přihlásit na jiný.

```
particle flash ID soubor
```

Nahraje binární firmware, zdrojový soubor, adresář zdrojových souborů nebo aplikaci na zařízení.

```
particle flash --target 0.5.0 ID soubor
```

Možné upravit verzi firmwaru pomocí flagu --target, pokud není zařízení upgradované na nejnovější firmware.

```
particle compile photon app.ino library1.cpp library1.h --saveTo firmware.bin

particle compile photon adresář --saveTo firmware.bin

particle flash --usb firmware.bin
```

Zkompiluje jeden nebo více souborů a stáhne jako binární firmware. Je třeba uvést argument Photon nebo p pro upřesnění kompilování na Photon. Při kompilování adresáře je možné vytvořit soubor particle.include nebo particle.ignore, k upřesnění, které soubory má využít, a které ignorovat. Povolené je jen jedno jméno souboru na řádek.

```
particle call id funkce parametr

particle call 0123456789ABCDEFGHI digitalWrite "D7,HIGH"
```

Zavolá funkci zařízení.

```
particle get id promenna

particle get 0123456789ABCDEFGHI temperature

72.1
```

Získá hodnotu proměnné.

```
particle monitor id promenna cas

particle monitor 0123456789ABCDEFGHI temperature 5000
```

Získává hodnotu v daném intervalu. Nejmenší interval je 500.

```
particle identify
```

Zobrazí id připojeného zařízení v listening modu

```
particle keys doctor id
```

Pomáhá aktualizovat klíče nebo obnovit zařízení v případě, že nejsou klíče na serveru synchronizované s klíči na zařízení. Vyžaduje mít nainstalované DFU-util a OpenSSL. Je třeba uvést zařízení do DFU módu (viz. Uvedení do DFU). Po spuštění nahradí šifrovací klíče na zařízení. Automaticky se pokusí poslat nový veřejný klíč na cloud.

```
particle keys new
```

Generuje nový privátní a veřejný klíč.

```
particle keys load soubor.der
```

Zkopíruje DER privátní klíč na zařízení.

```
particle keys save soubor.der
```

Zkopíruje DER privátní klíč do počítače.

```
particle keys send soubor.pub.pem
```

Odesílá veřejný klíč zařízení na cloudu. Je třeba se ujistit, že zařízení má odpovídající soukromý klíč.

```
particle keys server soubor.der
```

Změní veřejný klíč serveru uložený na zařízení.

3. Cloud

Cloud patří mezi druh výpočetní techniky na internetové bázi, která poskytuje sdílené prostředky a data ke zpracování počítačům a dalším zařízením na zakázku. Umožňuje všudypřítomný přístup na požádání (on-demand) ke sdílenému fondu konfigurovatelných výpočetních zdrojů (například sítě, servery, úložné prostory, aplikace a služby).[17]

V práci bude ukázáno využití oficiálního Particle cloudu nebo upraveného privátního cloudu, obě možnosti mají své výhody a nevýhody.

3.1 Particle Cloud

Particle cloud spojuje produkty vytvořené na Particle platformě přes web, poskytuje bezpečnou bránu k interakci s IoT zařízeními prostřednictvím rozhraní (API) a webových služeb.

Zařízení využívají mikrokontrolery minimalizující útoky, které jsou často nevědomě vystaveny systémem na čipu (SoC), které spouští celý OS. Všechny zprávy mezi zařízeními a Particle Cloudem jsou chráněny silným šifrováním a detekcí anomálií. Časté kontroly totožnosti a rychle se měnící klíče relací pomáhají zmírnit útoky typu man-in-the-middle. API inteligentně vyrovnává zatížení ke snížení útoků hrubou silou (brute-force). Každý Particle server je umístěn za přísně nastavenými firewally privátní sítě s detekcí narušení, aby se zabránilo neoprávněnému přístupu.[18]

Cloud využívá REST API, „zdrojem“ je zařízení Photon. Každý Photon má vlastní URL, které může být použito k získání proměnných GET, k volání funkcí POST nebo nasazení nového firmwaru PUT.

Díky tomu je možné bezproblémově odeslat aktualizaci firmwaru pro přístroje „vzduchem“. Opravovat nalezené chyby už nasazených zařízení nebo nabízet novou sadu funkcí pro stávající zařízení, zcela bezdrátově. Particle Cloud využívá integrací s mnoha službami,

[17] MELL, Peter a Timothy GRANCE. *The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology* [online]. 1. -: U.S. Department of Commerce, 2011 [cit. 2016-05-13].

[18] The Particle Cloud. *Particle* [online]. San Francisco: Particle, 2016 [cit. 2016-05-13].

umožňující sdílet data. Využít analýz strojového učení na Google Cloud Platform. Posílat notifikace podle chování zařízení v Microsoft Azure nebo využívat webhooks, k ukládání diagnóz zařízení ve vlastních databázích.[19]

Po prvním připojení k Particle Cloudu bude Photon asociovaný k přihlášenému účtu, tím se vygeneruje přístupový token (access token), klíč, kterým je možné ovládat zařízení. Existuje více možností, jak posílat access token. Nejdůležitější v HTTP hlavičce, která funguje vždy pro všechny možnosti. V URL řetězci, funguje jen v případě dotazu GET. V těle dotazu, funguje pouze s POST a PUT, pokud je tělo šifrované.

Obrázek 10 - Komunikace

Aby Photon mohl komunikovat s Particle Cloud (obr. 10) otevře trvalý TCP socket využitím šifrované verze CoAP skrz odchozí port 5683 na síti. CoAP „Constrained Application Protocol“ je v podstatě efektivní verze protokolu HTTP používající se při omezených podmínkách jako na malých zařízení internetu věcí.[20]

Odesílaná data jsou chráněná pomocí šifrování AES s klíčem relace generovaným na cloudu a odeslaným zařízení pomocí RSA veřejný/privátní klíč, které jsou nastavené při výrobě. Každé zařízení má unikátní privátní klíč ve flash paměti. Veřejný klíč je uložený na cloudu, zařízení mají veřejný klíč Particle Cloudu. Při každém připojení obdrží zařízení nový relační AES klíč. Tím je útok limitovaný jen pokud má útočník fyzický přístup k zařízení (čipu nebo USB), došlo by jen k získání jednoho zařízení ostatní by nebyly kompromitovány. Komunikace a ovládání na cloudu jsou spravováno skrz OAuth2.[21]

---

[19] The Particle Cloud. *Particle* [online]. San Francisco: Particle, 2016 [cit. 2016-05-13].

[20] SHELBY, Z., K. HARTKE a C. BORMANN. The Constrained Application Protocol (CoAP). *The Internet Engineering Task Force* [online]. Universitaet Bremen TZI: Universitaet Bremen TZI, 2014 [cit. 2016-12-19].

[21] Particle docs. Particle [online]. San Francisco: Particle, 2016 [cit. 2016-02-20].

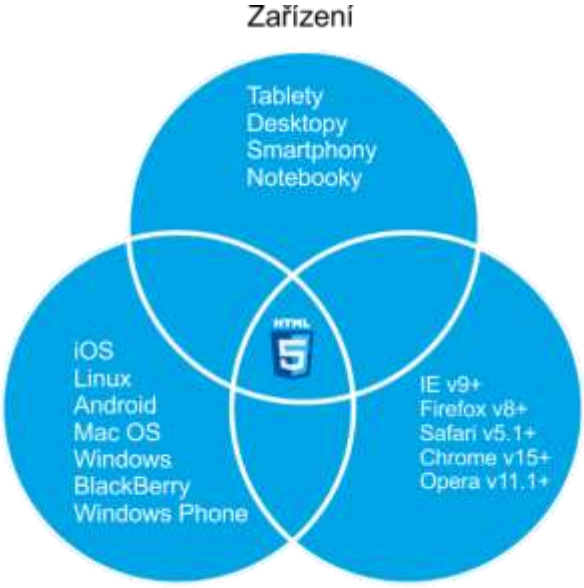
Privátní cloud – Oproti oficiálnímu cloudu je při využití privátního cloudu větší kontrola nad vlastnictvím posílaných dat. Privátním cloudem je myšlen server, který je spuštěn přímo v místě používání zařízení například na lokální síti. Tím je zamezen přístup z internetu. Privátní cloud, není delší dobu udržovaný společností Particle, jeho funkce jsou oproti oficiálnímu cloudu omezené.

4. Apache Cordova

Apache Cordova je webová šablona (framework) pro tvorbu aplikací vytvořená firmou Nitobi a následně koupená firmou Adobe a přejmenovaná na PhoneGap. Adobe později vypustila open source verzi softwaru pojmenovanou Apache Cordova.[22]

4.1 Technologie

Apache Cordova umožňuje vytvářet aplikace pro multimediální platformy pomocí zapouzdření HTML5, JavaScript a CSS3, které mají velké zastoupení napříč všemi zařízeními (obr. 11). Namísto práce s nativními technologiemi od Android, iOS a Windows Phone.



Obrázek 11 - Průnik technologií

[22] Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap. Adobe [online]. Los Angeles: Adobe, 2011, s. 1 [cit. 2016-04-15].

Výsledná aplikace se nazývá hybridní. Hybridní z důvodu, že všechno vykreslování je zajištěné pomocí Webového rozhraní (WebView), namísto nativních rozhraní (UI).[23] Zároveň to není zcela webové rozhraní, díky možnosti přístupu k nativním funkcím zařízení pomocí pluginů. Pluginy umožňují nativní funkce jako mikrofon, souborový systém, akcelerometr, kamera a další, díky velké podpoře komunity. Funkce se volají pomocí JavaScriptu, komunikujícího mezi nativní a webovou vrstvou. Přes mnoho výhod použití této technologie se zde objevuje problém s pomalejším během aplikací oproti nativním.[24]

Díky svým vlastnostem se stává hybridní aplikací možnou dále distribuovat na další systémy. Aplikace bude spustitelná na širokém spektru zařízení tzv. multiplatformní, a nevyžaduje mít oddělené týmy zabývající se jednou platformou. Na Apache Cordově je postaveno mnoho vývojových frameworků například Ionic, Adobe PhoneGap nebo Intel XDK.

4.2 Architektura aplikace

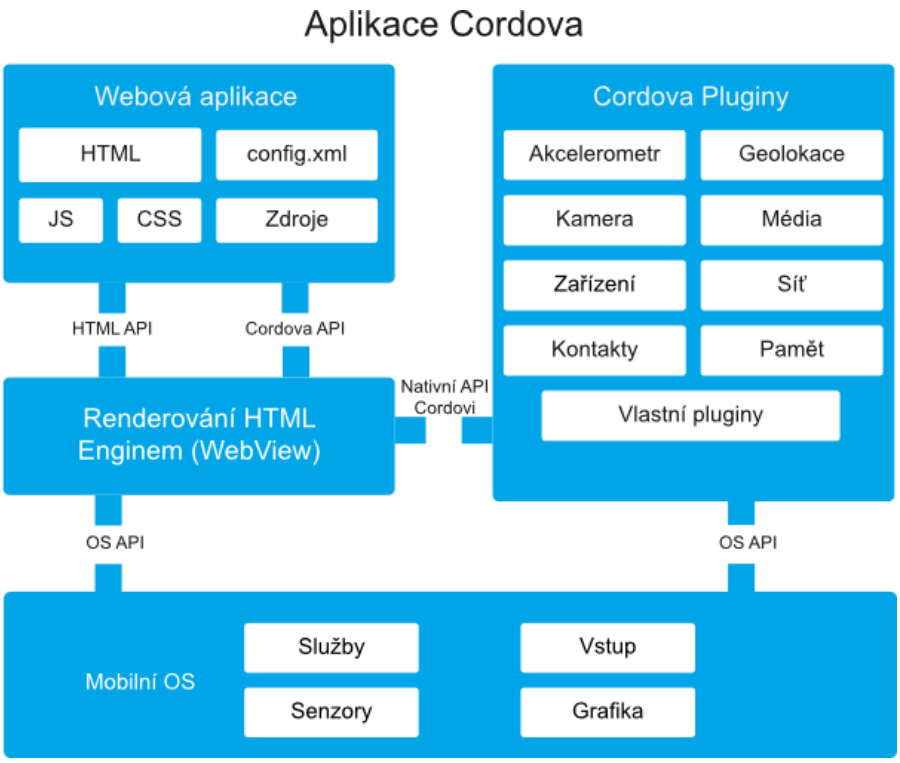
Cordova je složena z několika komponent (obr. 12). Obrázek poskytuje pohled na strukturu aplikace. Velmi důležitou částí je komponenta webové aplikace, která obsahuje stránku ***index.html*** ze které se volají všechny skripty, styly a další stránky. Komponenta obsahuje i velmi důležitý soubor ***config.xml*** obsahující všechny upřesňující parametry ovlivňující funkčnost aplikace.

Další z hlavních komponent jsou pluginy Cordova, které umožňují rozhrání mezi Cordovou a nativními komponenty komunikovat a volat jejich kód pomocí JavaScriptu. Apache Cordova spravuje skupinu pluginů Core Plugins, které umožňují práci se základními funkcemi zařízení jako je baterie, kamera, kontakty a další. Také existuje veliké množství pluginů třetích stran umožňující dodatečné funkce. Mohou být staženy pomocí plugin search nebo programem

[23] FERMOSO, Jose. PhoneGap Seeks to Bridge the Gap Between Mobile App Platforms. In: Gigaom [online]. -: Knowingly, 2009 [cit. 2016-04-15].

[24] *The Development of Mobile Applications using HTML5 and PhoneGap* on Intel® Architecture-Based Platforms. Intel Developer Zone* [online]. 2012 [cit. 2016-04-15].

npm. Při vytvoření nového projektu nebývají přítomny ani Core Plugins a musejí být ručně přidány. [25]



Obrázek 12 - Schéma Cordova aplikace

4.3 Možnosti vývoje

Při vývoji aplikace v Cordově se využívají dva základní postupy. I když je možné využít obě možnosti ke splnění stejného cíle, každá nabízí jiné výhody. Začátečníkům je doporučován jednodušší cross-platform postup.

Cross-platform (CLI) workflow

Používá se, pokud je důležité, aby aplikace fungovala na co nejvíce systémech. Bez specifických podrobností konkrétních platforem. Pracovní postup se zaměřuje na Cordova CLI. CLI je nástroj umožňující práci na vyšších vrstvách aplikace, který umožňuje vytvářet projekty pro mnoho platforem najednou, odděluje pryč většinu funkcí shell skriptů nižší úrovně. CLI kopíruje

[25] Apache Cordova: Overview. In: *Cordova* [online]. -: The Apache Software Foundation, 2016 [cit. 2016-12-03].

společné balíky webových doplňků do podsložek pro jednotlivé mobilní platformy a upraví potřebné změny. Následně spouští skripty k vytvoření binárních souborů.

Platform-centered workflow

Je vhodný při vytváření aplikace zaměřenou na jedinou platformu nad ostatními a je třeba, aby ji bylo možné upravit na nižší úrovni. Například, když se kombinují nativní komponenty s webovými Cordova komponenty. Tento postup spoléhá na nižší úroveň shell skriptů, které jsou přizpůsobeny pro každou podporovanou platformu a samostatnou utilitu Plugman, která umožňuje přidávat pluginy. Způsob je možné použít k cross-platform aplikacím, obecně je těžší z důvodu absence nástrojů vyšších vrstev, kde je potřeba mít odděleně sestavené aplikace (build) a úpravu pluginů.

5. Ionic

Ionic představuje skvělou volbu při programování aplikací postavených na Cordově, oproti PhoneGap nebo klasické Cordově, využívá technologii AngularJS namísto jQuery, tím dosahuje vyššího výkonu a plynulosti. Dovoluje se spoléhat na hardwarovou akceleraci oproti rozšířené DOM manipulaci, těží z animací CSS, ke snížení zátěže GPU, a zvyšuje dostupnost procesoru.[26]

Vývojáři nezačínají při každém projektu znovu s prázdnou složkou a neřeší problémy jednotlivých prohlížečů, dal by se přirovnat například k Twitter Bootstrap nebo Zurb Foundation. Díky šablonám se získá rychlejší start k samotné práci a nemusí se řešit design a cross-browser testování. Ionic má stejné cíle jako zmíněný Bootstrap, praxí osvědčené postupy, doporučený vzhled a je open-source, aby komunita mohla sdílet své poznatky.[27]

Při psaní bakalářské práce byla vydána stabilní verze nového Ionic 2, který těží z nejmodernějších technologií a dosahuje větších výkonů totožných s nativními aplikacemi. Pro větší užitečnost se bakalářská práce bude věnovat novějšímu frameworku, který má slibnou budoucnost.

5.1 Změny Ionic 2

V Ionic 2 došlo k přechodu na novější verzi Angular 2, zaměřující se na vývoj mobilních aplikací. Dochází k velkému nárůstu výkonu, zjednodušení a zpřehlednění syntaxe.

Ionic 2 je nově psaný pomocí TypeScript namísto klasických ES6 transpilerů, překladačů, které přeloží zdrojový kód z jednoho programovacího jazyka do jiného.[28] V Ionic 2 se nachází nové a přepracované komponenty s větší efektivitou, kopírující nativní vzhled jednotlivých systému nazvaný Platform Continuity, díky kterému se převede na odpovídající styl OS. Dále přidává nadstavbu Ionic Native a rozšiřuje dosavadní pluginy o nativní pluginy obsahující specifikace nejnovějšího EcmaScriptu 6. Aplikace obsahují standard stylů pro zařízení, který je

[26] Ionic [online]. -: Ionic, MIT, 2016 [cit. 2016-12-03].

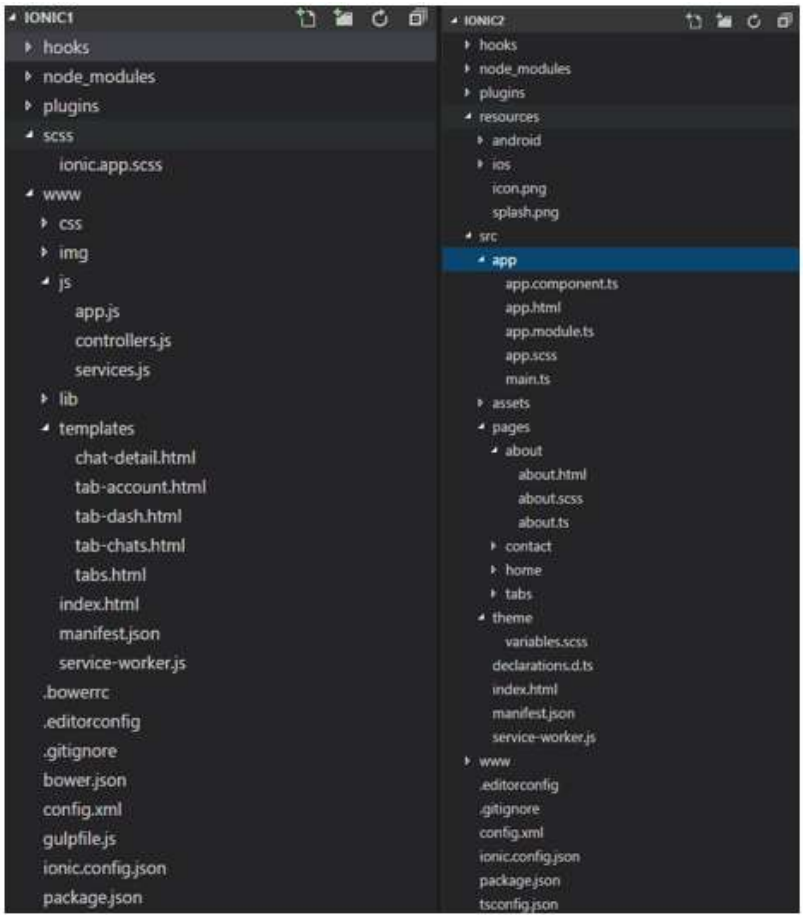
[27] BRADLEY, Adam. Where does the Ionic Framework fit in? - [online]. 2013, 2013(1), 1 [cit. 2016-07-31].

[28] Compiler. Compilers.net [online]. -: GNU Free Documentation License, 2005 [cit. 2017-02-10].

možné snadno modifikovat upravením proměnných v souboru sass (rozšířené stylování css, umožňující vrstvit styly a vytvářet proměnné). Jedna z nejdůležitějších funkcí je zvýšení výkonu přechodem na Virtual Scroll z původního JavaScript scrolling.[29]

5.2 Organizování a struktura

V Ionic 2, je každá stránka nebo komponenta umístěna ve vlastní složce se svou šablonou (template), stylem (scss) a třídou určenou souborem .ts (TypesScript) (obr. 13). Vše udržuje velmi organizované a umožňuje mít vytvořené funkce velmi modulární (možné znovupoužít). Ionic 2 nutí dělat věci správným způsobem, nelze snadno porušit osvědčené postupy. Je zde vidět velmi podobná struktura jako u Angular 2.[30]



Obrázek 13 - Porovnání struktur

[29] LYNCH, Max. Announcing Ionic 2.0.0 Final. Blog Ionic [online]. San Francisco: Ionic, 2017 [cit. 2017-02-10].

[30] MORONY, Joshua. 7 Reasons Why Ionic 2 Is Better Than Ionic 1. Josh Morony [online]. 2016 [cit. 2017-02-13].

Praktická část

6. Photon

Photon je možné nastavit více způsoby. Důležitý způsob pro práci je nastavení přes USB port a druhou rychlejší možností je nastavení pomocí aplikace v chytrém telefonu, k okamžitému používání.

6.1 Nastavení pomocí USB

Aby se Photon mohl propojit s Windows je potřeba nainstalovat pár základních programů. Důležité je využít 32-bit verzi pro následující nastavování privátního cloudu v další kapitole. Odzkoušená funkční verze Node.js dále node, použitá v bakalářské práci je **node-v0.12.0-x86**. K instalaci je doporučeno používat manažér verzí **nvm** ze stránek nacházející se na CD.

```
https://github.com/creationix/nvm //Linux, MacOS

https://github.com/coreybutler/nvm-windows //Windows

https://s3.amazonaws.com/spark-website/Particle.zip //Particle driver
```

Po stažení a nainstalování programu se použije příkazová řádka k nainstalování požadované verze **node** a následně se aplikuje. Tím je postaráno o správnou funkčnost node.

```
nvm install 0.12.0 32 //instalace

nvm use 0.12.0 32 //použití
```

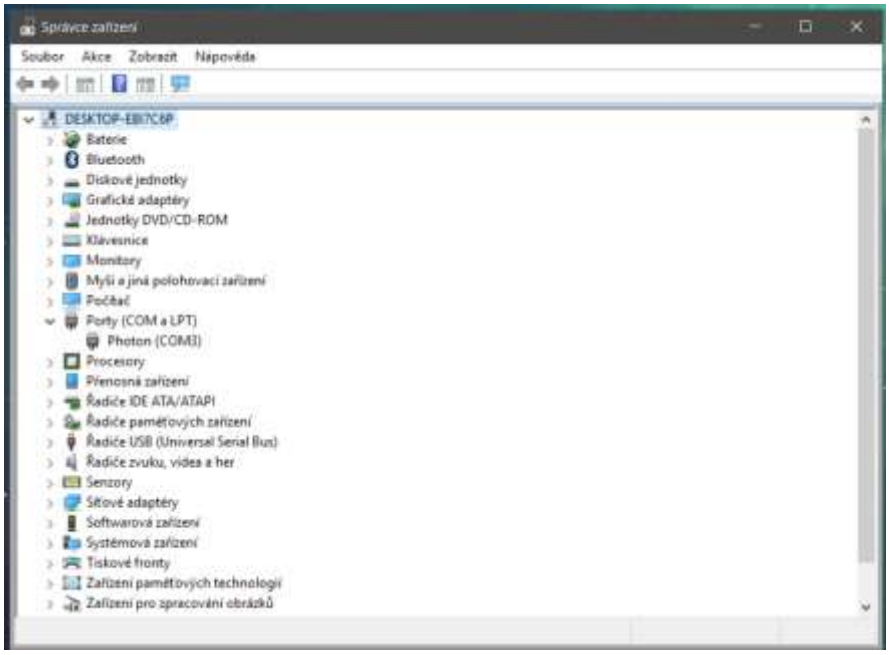
Následuje zpřístupnění Photonu nainstalováním potřebných ovladačů ze odkazu nebo z přiloženého CD: *2-Particle_Photon/Instalace/Particle-drivers.zip*
Nejprve se připojí Photon do USB portu a uvede se do Listening mode (blikající modré) podržením tlačítka **SETUP** po 3 vteřiny.

1. Ve **Správci zařízení**. By se Photon měl objevit v **Dalších Zařízení** na OS Windows 10 může být zobrazen pod složkou **Porty** nebo také **Řadiče USB**.
2. Kliknutím na **Aktualizovat software ovladače** se dostane do výběrového menu.
3. **Vyhledat ovladač v počítači**. Nastaví se umístění stažených ovladačů.
4. **Vybrat ovladač ze seznamu** následně **Z disku** a nastaví se zdrojový soubor **photon.inf**.
5. Potvrdí se instalace ovladačů. Windows by měl zařízení správně detekovat. (obr. 14). K dalšímu postupu se využije klasického příkazového řádku nebo programu Powershell.
6. Pomocí node se nainstaluje Particle CLI

```
7. npm install -g particle-cli
```

8. Po nainstalování klienta lze s Photonem začít pracovat.
9. Photon musí být zapojený do USB. Pokud není v Listening mode uvede se do něj podržením tlačítka **SETUP** po 3 vteřiny.
10. V příkazovém řádku se zahájí nastavení Photonu zadáním příkazu setup a následně postupováním podle pokynů programu.

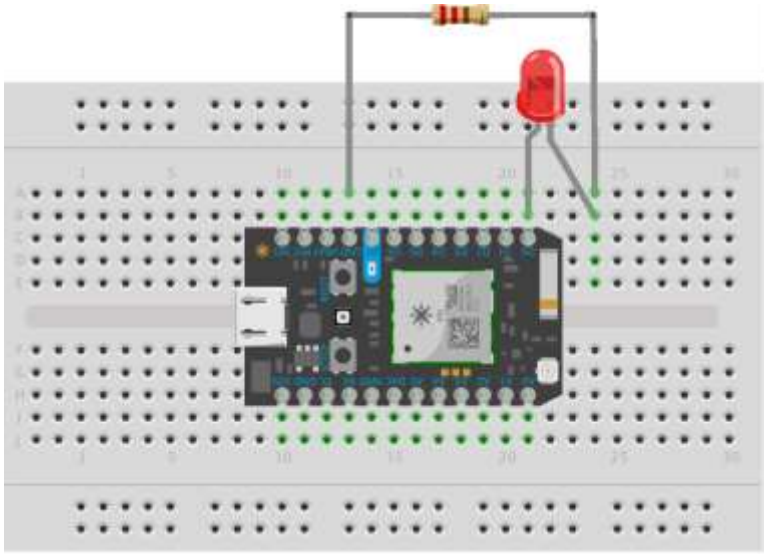
```
11. particle setup
```



Obrázek 14 - Nainstalovaný Photon

6.2 Příklad zapojení

Na ukázce je vysvětlena základní práce a postupy, „Hello World“ světa malých zařízení. Photon bude rozsvěcet 2 diody (obr. 15).[28]



Obrázek 15 - Zapojení na nepájivém poli

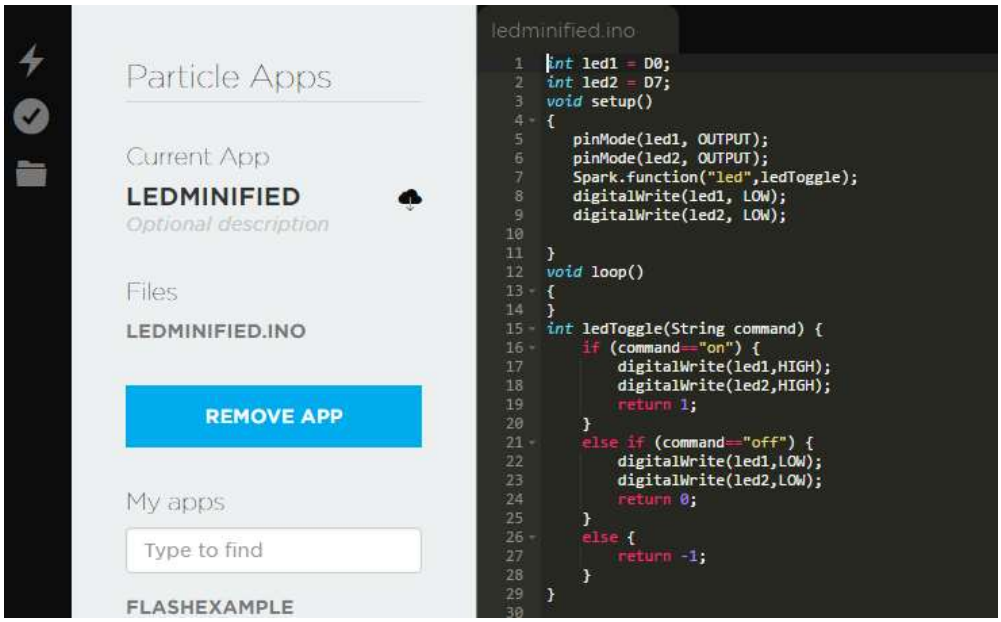
Photon se zapojí do nepájivého pole a na výstup D0 mu bude přivedena LED anoda. Katoda se uzemní pomocí rezistoru, druhá nožička rezistoru se zapojí do GND.[31] Než začne Photon provádět požadované příkazy, je nezbytné nahrát do paměti kód, který se má vykonat. Napsat a nahrát kód je možné více způsoby, první možnost je pomocí webového klienta.

Webový klient

```
https://build.particle.io/
```

Po přihlášení uživatelskými údaji se zadá název aplikace, po potvrzení je možné psát. Editor také obsahuje předpřipravené kusy kódu z ukázkových příkladů. Když je kód napsaný, použitím ikonky blesku, se nahraje na Photon (obr. 16).

[31] *Blink an LED*. Particle Docs [online]. San Francisco: Particle, 2016 [cit. 2016-04-08].



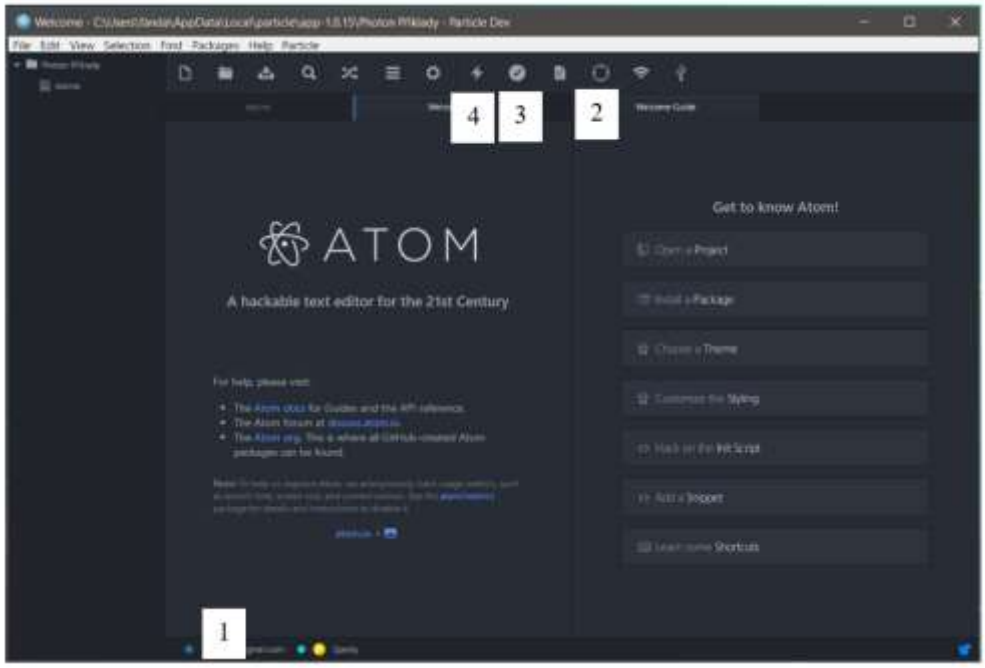
Obrázek 16 - Ukázka webového rozhraní

Druhou možností je pomocí aplikace Particle Dev, která lze stáhnout ze stránek výrobce.

Aplikace

```
https://www.particle.io/dev
```

1. Po spuštění aplikace je dobré se přihlásit na Particle Cloud (obr. 17.1). Kliknutím do dolního rohu aplikace, jinak by nebylo možné zařízení spravovat.
2. Následně se vybere, s jakým zařízením se má pracovat (obr. 17.2).
3. Kód, který má Photon vykonat, se napíše do záložky **untitled**. Soubor musí být před kompilací uložen, je třeba za jméno souboru umístit koncovka **.ino**.
4. Poté je možné soubor zkompilovat a nahrát pomocí cloudu na Photon (obr. 17.3 a obr. 17.4), během kompilace dojde k vytvoření souboru photon_firmware_číslo.bin.



Obrázek 17 - Popis programu

Příkazová řádka

Třetí možností je nahrátí pomocí příkazové řádky, kód se napíše v libovolném editoru a pomocí příkazů se zkompiluje a nahraje.

```
particle compile photon app.ino --saveTo firmware.bin

particle flash --usb firmware.bin
```

Výhodou prvních dvou metod je, že Photon nepotřebuje být fyzicky v dosahu, stačí, když bude mít přístup k internetovému připojení, díky kterému nezáleží, jestli je Photon vedle vývojáře, anebo kdekoliv jinde na světě. Okomentovaný kód použitý ze stránek výrobce[32] je možné nalézt na přiloženém CD této práce *2-Particle_Photon/Zdrojovy_kod/Priloha1-LED*.

Po nahrátí zkompilovaného souboru lze ovládat svícení led diody připojené na nepájivém poli více způsoby, pomocí příkazové řádky nebo vytvořením webové rozhraní využívající REST API.

[32] *Blink an LED*. Particle Docs [online]. San Francisco: Particle, 2016 [cit. 2016-04-08].

Nejprve se přihlášením ověří, zda má uživatel oprávnění k práci s Photonem a následně se využije příkazu k rozsvícení diod. Pomocí led off k zhasnutí.

```
particle login

particle call Sparky led on
```

Webový klient

Pro využití webového rozhraní je zapotřebí znát ID zařízení a jeho přístupový token (access token) Ty je možné zobrazit ze stránky

```
https://build.particle.io/
```

Složka **Devices** obsahuje ID zařízení a složka **Settings** přístupový token. Dále se vytvoří html stránka (*2-Particle_Photon/Zdrojovy_kod/Priloha1-LED*) a umístí se hodnoty namísto *device-ID* a *access-token*. V prohlížeči lze rozsvítit a zhasnout LED.

7. Cloud

Zprovoznění privátního cloudu není na Windows snadnou záležitostí, není podpora u Particle, problémy byly nakonec vyřešeny. Následují postupy, jak je možné zprovoznit privátní cloud na Windows a Linux.

7.1 Instalace Windows

V této části je popsán postup k instalaci Cloudu na Windows. Při ukázce byl využit čistě nainstalovaný virtualizovaný počítač **Windows 7 x86** ve VirtualBox, také se podařilo spustit na aktuálním **Windows 10 x64**.

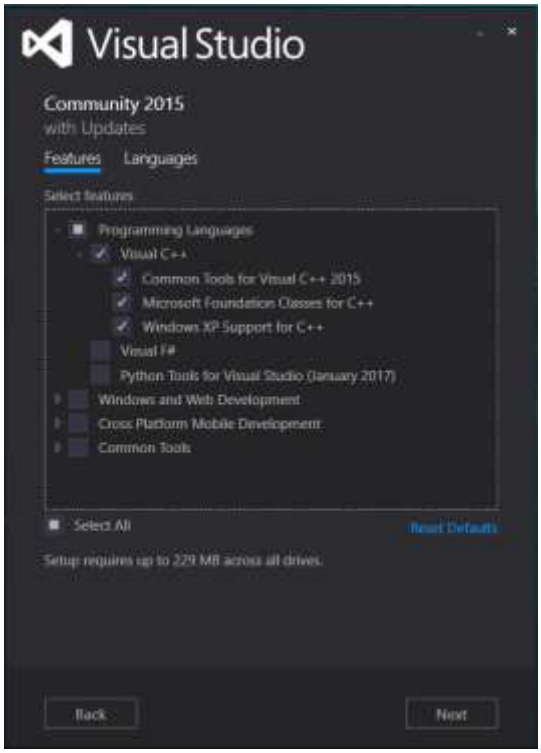
Potřebné programy

Všechny uvedené programy jsou součástí přiloženého CD: *3-Privatni_Cloud*.

- Git
- OpenSSL 1.0.1h 32-bitový
- Node.js 32-bitový (nvm) Předchozí kapitola
- DFU-util
- Potřeba Microsoft Express 2012 nebo Microsoft Visual Community 2015
- particle-cli (viz. předchozí kapitola)
- Python 2.7.3 (nové verze problémy s kompatibilitou)

Instalace Microsoft Visual Community 2015/Express 2012

V pokročilém výběru částí, které se mají nainstalovat, se zaškrtne balíček Visual C++, který bude využit pro zkompilování částí cloudu. Bez něj by proces neproběhl správně. Pokud je Visual v počítači nainstalovaný bez balíčku, je možné jej doinstalovat. V *Ovládacích panelech, Programy a funkce* se nalezne Microsoft Visual Studio a pomocí *změnit* se vyvolá instalační okno. V nabídce se vybere *Modify* a zaškrtne balíček Visual C++ (obr. 18).



Obrázek 18 - Instalace balíčku

Přidání OpenSSL, DFU-util, Python

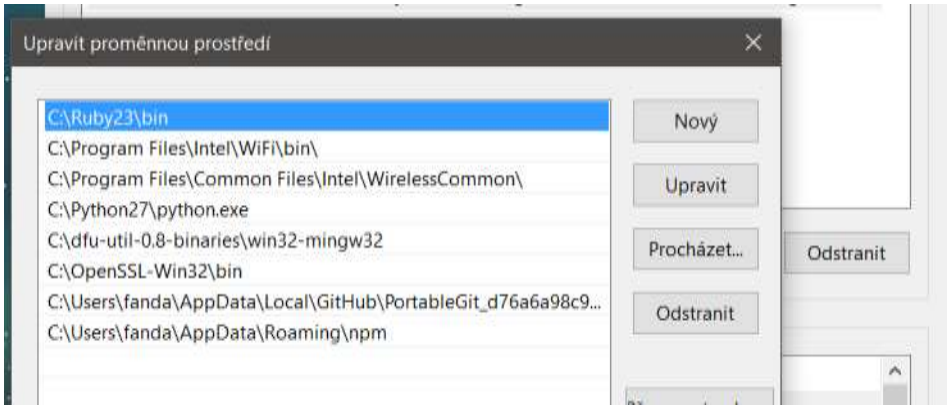
Po nainstalování programů, je důležité přidat cesty programů do proměnné *Path* v *Proměnném prostředí Windows*. Cesta se zkopíruje do proměnného prostředí (obr. 19), je vhodné celý počítač restartovat pro potvrzení změn.

```
C:\OpenSSL-Win32\bin

C:\Python27\python.exe[33]

C:\dfu-util-0.8-binaries\win32-mingw32
```

! Než je možné instalovat privátní cloud, musí být Photon připojen alespoň jednou na Particle Cloud.

Důvodem je vygenerování veřejného klíče při prvním spuštění. Po splnění potřebných prerekvizit se pomocí Git Shell nebo příkazové řádky stáhnou repozitáře ze severu GitHub.

[33] Důležité je zadat přímo cestu exe soboru, jinak se prostředí snaží spustit složku.



Obrázek 19 - Ukázka umístění souborů

Zřídka nastává problém s cestou Git, kdy skončí npm chybou, v takovém případě je potřeba spustit příkazy v Git Shellu.

V průběhu práce se docílilo zjednodušení postupů podle verzí Visual a node, budou uvedeny obě metody, jak je možné zprovoznit vlastní cloud. Metody nejdou kombinovat a je nutné si vybrat jednu z možností. Program node je velmi závislý na verzích, velmi často se stává, že pokud se cokoliv změní od požadované verze, program skončí chybou.

Verze Microsoft Visual Express 2012

```
nvm use 0.12.0 32 //Využití starší verze Node

git clone https://github.com/spark/spark-server.git

git clone https://github.com/spark/spark-protocol.git

git clone https://github.com/morkai/h5.coap.git

cd spark-protocol

mkdir js

cd js

npm install ..\..\h5.coap

npm install --msvs_version=2012 //Instalace verzí 2012

npm install
```

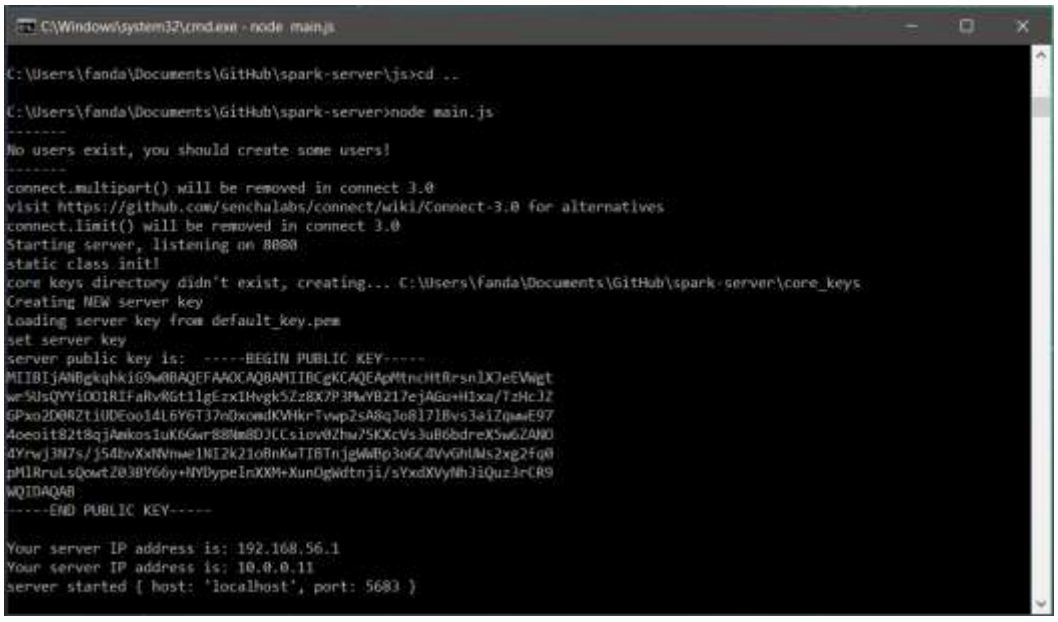
```
cd ..\..\spark-server\js

npm install --msvs_version=2012

npm install ..\..\spark-protocol\js

npm install

cd ..

node main.js
```

Verze Microsoft Visual Community 2015

U verze Community je možné využít nejnovější verzi node. Ta při ukázce byla ve verzi 7.4.0x32. 64bitová verze nelze použít kvůli nekompatibilitě s knihovnou OpenSSL.

```
nvm install 7.4.0 32

nvm use 7.4.0 32

git clone https://github.com/spark/spark-server.git

cd spark-server

npm install

node main.js
```

Dojde ke spuštění serveru a vygenerování veřejného klíče (obr. 20).



Obrázek 20 - Funkční server

7.2 Instalace Ubuntu (Linux)

V této části je uveden postup k instalaci Particle Cloud na systém Ubuntu. Při ukázce byl využit čistě nainstalovaný virtualizovaný počítač **Ubuntu 16.04** ve VirtualBox. Potřebný kód ke spuštění vykonává stejné postupy jako v případě Windows. Je doporučováno použít správce nodejs nvm.

```
wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.33.1/install.sh | bash

sudo nvm install 7.4.0

nvm use 7.4.0

sudo apt-get install git

sudo apt-get install dfu-util

sudo apt-get install python2.7
```

Pro další postup je nutné mít nainstalovaný particle-cli.

```
npm install -g particle-cli

particle cloud login
```

K instalaci cloudu se zadají tyto příkazy.

```
git clone https://github.com/spark/spark-server.git

cd spark-server

npm install

node main.js
```

Oproti instalaci cloudu na Windows je instalace Ubuntu kratší.

7.3 Zprovoznění DFU util (Windows)

K práci je nutné mít stáhnutý program **Zadig** přiložený na CD.

1. Nejprve se Photon uvede do DFU režimu a spustí se program Zadig.
2. V menu **Options** se vybere možnost **List all devices**.
3. Z nabídky se vybere **Photon DFU** a driver **WinUSB / libusbK**
4. Po výběru se spustí instalace. Když je instalace hotova, zobrazí se zpráva.
5. K ověření správné funkčnosti nainstalovaných ovladačů se v příkazové řádce zkontroluje.
6. `dfu-util -l`
7. Zařízení by mělo být zobrazeno v DFU režimu (obr. 21).

Obrázek 21 - Ukázka funkčního DFU util

7.4 Nastavení Photonu

Tato část je společná pro oba systémy. Vytvoří se nový profil lokálního serveru pomocí Particle-CLI. Pro lokální cloud je potřeba přidat číslo portu 8080 nakonec.

```
particle config jmeno_profilu apiUrl http://doména_nebo_IPadresa:8080
```

Příkazem particle config identify se zobrazí, na jaké místo client odkazuje. Odkáže se na lokální cloud.

```
particle config identify

particle config jmeno_profilu
```

Následně je možné provést vytvoření nového lokálního účtu. Po úspěšném registraci, se přeruší proces připojení Photonu CTRL+C.

```
particle setup
```

Dále je nutné opravit CryptoStream chybu, která může nastat nekompatibilitou verze Node.[34]

```
spark-server\node_modules\spark-protocol\lib\CryptoStream.js

Nahradit řádku 72:

if(chunk){

  if (!ciphertext) {

    ciphertext = chunk;

  } else {
```

[34] https://github.com/spark/spark-protocol/pull/14

```
    ciphertext = Buffer.concat([ciphertext, chunk], ciphertext.length +
chunk.length)}}
```

Následně je možné přejít do složky particle-server. Photon se uvede do DFU režimu a upraví se klíč serveru na klíč lokálního cloudu + IP adresu (bez http://). Po vykonání se umístí veřejný klíč do složky core_keys.

```
particle keys server default_key.pub.pem IP_Adresa

cd core_keys

particle keys save IdZařízení
```

Je nezbytné použít správné Id zařízení při ukládání klíčů. Manuálně se resetuje zařízení přes tlačítko **RESET**. Ke kontrole spojení musí běžet **particle-server**. Pomocí příkazové řádky se spustí node ve složce particle-server.

```
node main.js
```

V tomto bodě je server správně nastavený. Je možné zkontrolovat správnou funkčnost restartováním zařízení, pokud bude zobrazena aktivita v příkazové řádce stejně jako v příkladu, nastavení proběhlo v pořádku. Příklad:

```
Connection from: 192.168.1.159, connId: 1

on ready { coreID: '48ff6a065067555008342387',

  ip: '192.168.1.159',

  product_id: 65535,

  firmware_version: 65535,

  cache_key: undefined }
```

7.5 Příklad lokálního cloudu

Pokud je na Photonu stále nahraný kód z **kapitoly 7.2**. Je možné malou úpravou zpřístupnit pro lokální server. Celý kód na CD *2-Particle_Photon/Zdrojovy_kod/Priloha1-LED/*. Nahradí se IP adresa místo názvu domény a protokol http. Cloud v lokální verzi https nepodporuje, uživatel si jej musí zpřístupnit sám. Dále je potřeba nahradit access token, ten je možné zjistit, pokud v příkazové řádce ukazuje particle na privátní server a příkazem se zobrazí.

```
particle config identify
```

7.6 Návrat k výchozímu Particle cloud

Aby Photon komunikoval opět s oficiální cloudem, musí se nahrát veřejný klíč dostupný ze stránek níže. Následně se uvede Photon do DFU režimu a nahraje se veřejný klíč na Photon. Poslední úpravou je odkázání klientem zpět na Particle. Tím se docílí navrácení zpět na oficiální Particle Cloud.

`https://s3.amazonaws.com/spark-website/cloud_public.der // stránky klíče`

```
particle keys server cloud_public.der

particle config particle
```

8. Apache Cordova (Ionic 2)

8.1 Instalace

V této části je popsán postup vývoje aplikace pomocí Cordova a jejího frameworku Ionic 2.

Potřebné programy

- Node.js 6 a novější

Aby bylo možné v Ionic pracovat, je vyžadována novější verze node. Tu je možné stáhnout pomocí manažéru verzí nvm. Při psaní práce došlo k problému, kdy den předem vyšla nová verze 6.8.1, která zatím neměla novou verzi npm a novou verzi NODE_MODULE. V běžném případě by příkaz s parametrem *install latest* měl být funkční. Přepne se na nejnovější verzi node. Následně se může pokračovat s instalací Cordova a Ionic SDK. Instalace Ionic trvá déle oproti Cordově. Není se třeba obávat zamrznutí instalace.

```
nvm install latest //nainstalování nejnovější

nvm install 6.8.0 //použité v práci

nvm use 6.8.0

npm install -g minimatch@3.0.2

npm install -g cordova ionic
```

Pro vývoj na konkrétní platformy a jejich požadavky více v odkazu:

`https://cordova.apache.org/docs/en/latest/guide/support/index.html`

Požadavky

- Android
    - Java Development Kit (JDK)
    - Apache Ant

    - Android SDK
- Windows
    - Visual Studio 2015+ (Windows 8.1+)
- iOS
    - Xcode (Mac OS)

8.2 Požadavky Android

K využití testování a finálního exportování aplikace na Google Play je nutné nainstalovat a nastavit cesty k programům ručně. V práci bude uveden postup nastavení aplikace platformy Android. Více informací ohledně požadavků na platformy lze najít na stránkách:

`http://cordova.apache.org/docs/en/latest/guide/platforms/android/index.html`

Potřebné programy:

- Java JDK 7+
- Apache Ant
- Android SDK (je součástí Android Studio)

Potřebné programy se nacházejí na CD: *4-Cordova*. Po nainstalování je důležité vytvořit proměnné v prostředí Windows. Dále se umístí proměnná do cesty Path.

```
Proměnná

JAVA_HOME    C:\Program Files\Java\jdk1.8.0_121

ANDROID_HOME C:\Users\Uživatel\AppData\Local\Android\sdk



PATH         C:\apache-ant-1.9.2\bin

             %JAVA_HOME%\bin

             %ANDROID_HOME%\tools

             %ANDROID_HOME%\platform-tools
```

8.3 Vytvoření úvodní aplikace

Po stažení všech částí se přesune do místa, kde bude vytvořena aplikace. Příkazem start se vygeneruje struktura aplikace a doplní se o všechny potřebné soubory.

```
ionic start testAplikace tutorial --v2
```

--v2 specifikuje novou verzi Ionic 2. V případě vynechání se připraví struktura Ionic 1. Pokud není v úvodu specifikovaný vzhled, standardně se vytvoří se záložkami podobné iOS (tabs).

Vzhled rozhraní

- tabs: jednoduché troj záložkové rozvržení
- sidemenu: návrh s postranním menu
- blank: čistý start s jednou prázdnou stránkou
- super: projekt s více než 14 stránkami připraveného obsahu
- tutorial: úvodní projekt určený k vysvětlení základů

Po vytváření aplikace je dobrým zvykem přidat platformy, na které se následně aplikace bude vyvýjet, obvykle *android* nebo *ios*, také slouží k testování.

```
ionic platform add android
```

8.4 Spuštění aplikace

Do vytvořené složky se zanoří. Příkazem *serve* se spustí aplikace a zobrazí v prohlížeči. Příkazový řádek se nechá běžet na pozadí, po úpravě a uložení souborů dělá automatické kompilování a live-reloading stránky. Přidáním *-lab/-l* se spustí aplikace v emulovaném prostředí, kde je možné sledovat rozdíly mezi jednotlivými systémy a usnadňuje ladění detailů a odlišností platforem.

```
cd testAplikace
```

```
ionic serve –lab
```

8.5 Struktura aplikace

- **index.html** - je hlavní stránkou, která nastavuje skripty, CSS a spouští celou aplikaci. Pro funkční aplikaci Ionic hledá párovou značku <ion-app> a následující skripty.
- **build/main.js** - obsahuje Ionic, Angular a potřebný JavaScript
- **cordova.js** - vyhazuje 404 chybu během lokálního vývoje, následně se injektuje do projektu během fáze vytváření Cordovi
- **./src/**

Uvnitř adresáře najdeme nezkompilovaný kód. Je to místo, kde se vykonává většina práce. Po spuštění *ionic serve* je kód uvnitř src transpilován (přeložen) do správné verze JavaScript, které prohlížeč rozumí (v současné době, ES5). To znamená, že je možné pracovat na vyšší úrovni s využitím TypeScript, ale je přeložen na starší verzi Javascript podporující prohlížeče.

src/app/app.module.ts

Je vstupním bodem pro aplikaci. Nachází se zde deklarace celé aplikace. Každá aplikace má kořenový modul, který v podstatě řídí zbytek aplikace.
Kořenová komponenta je nastavena na třídu *MyApp* umístěnou v **src/app/app.component.ts**. Je to první komponenta, která bude nahrána do aplikace, a typicky slouží jako místo pro další komponenty, které se do ní nahrávají. V **app.component.ts**, je nastavena šablona (templateUrl) na **src/app/app.html** viz. dále.

```
@NgModule({
  declarations: [ MyApp,HelloIonicPage,ItemDetailsPage,ListPage],
  imports: [IonicModule.forRoot(MyApp)],
  bootstrap: [IonicApp],
  entryComponents: [MyApp,HelloIonicPage,ItemDetailsPage,ListPage],
  providers: [{provide: ErrorHandler, useClass: IonicErrorHandler}]
})
export class AppModule {}
```

/src/app/app.html

Je hlavní šablona pro aplikaci. V šabloně je nastaveno *ion-nav* jako postranní menu a *ion-menu*, jako prostor pro obsah nabídky. Vlastnost [content] je vázána na lokální proměnnou *nav* z *ion-nav* a tím získá informaci o kolo jakého místa se má animovat. Vlastnost [root] je vázaná s první „kořenovou“ stránkou *ion-nav* komponenty. Když se načte, komponenta referencovaná proměnnou *rootPage* bude hlavní stránkou.

```
<ion-nav [root]="rootPage" #content swipeBackEnabled="false"></ion-nav>

<ion-menu [content]="content">
  <ion-header>
    <ion-toolbar>
      <ion-title>Pages</ion-title>
    </ion-toolbar>
  </ion-header>

  <ion-content>
    <ion-list>
      <button ion-item *ngFor="let p of pages" (click)="openPage(p)">
        {{p.title}}
      </button>
    </ion-list>
  </ion-content>
</ion-menu>
```

V souboru **app.component.ts** se určuje tato vlastnost (rootPage).

```
...

import { HelloIonicPage } from '../pages/hello-ionic/hello-ionic';

export class MyApp {
  @ViewChild(Nav) nav: Nav;

  // určení stránky HelloIonicPage hlavní root stránkou.
  rootPage: any = HelloIonicPage;
```

```
  pages: Array<{title: string, component: any}>;

  constructor(
    public platform: Platform,
    public menu: MenuController
  ) {
    this.initializeApp();

// nastavení stránek aplikace
    this.pages = [
      { title: 'Hello Ionic', component: HelloIonicPage },
      { title: 'My First List', component: ListPage }
    ];
  }
```

Při procházení složky **hello-ionic** je možné si všimnout struktury zmíněné výše v teorii, kde každá stránka má vlastní složku s šablonou html, soubory stylů scss a svou třídu (class) ts, které jsou také kompilovány.

Níže lze vidět třídu **HelloIonicPage**. Ta vytváří stránku, komponentu Angular se všemi směrnicemi (directives), připravené k načtení navigačním systémem. Stránky jsou určeny k dynamickému načítání, nepotřebují mít *selector*. I tak je užitečné mít *selector* k přepsaní výchozích stylů u konkrétní stránky.

```
import { Component } from '@angular/core';
@Component({
  selector: 'page-hello-ionic',
  templateUrl: 'hello-ionic.html'
})
export class HelloIonicPage {
  constructor() {}
}
```

ion-navbar slouží k navigaci této stránky. Tlačítko a titul je součástí navigačního řádku. Zbytek šablony je standardní kód využívající Ionic, který nastavuje prostor pro zobrazení ***ion-content*** a zobrazí uvítací zprávu.

```
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Hello Ionic</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <h3>Welcome to your first Ionic app!</h3>
  <p>
    This starter project is our way of helping you get a functional app running in
record time.
  </p>
  <p>
    Follow along on the tutorial section of the Ionic docs!
  </p>
  <p>
    <button ion-button color="primary" menuToggle>Toggle Menu</button>
  </p>
</ion-content>
```

Při pohledu na další stránku **list.ts** je vidět, že stránka obsahuje pole s počtem prvků. V soboru se nachází také funkce *itemTapped*, která se odkazuje na další stránku **ItemDetailsPage**. Tu je nutné na začátku kódu importovat.

```
import { ItemDetailsPage } from '../item-details/item-details';

export class ListPage {
  selectedItem: any;
  icons: string[];
  items: Array<{title: string, note: string, icon: string}>;
  constructor(public navCtrl: NavController, public navParams: NavParams) {
    // If we navigated to this page, we will have an item available as a nav param
    this.selectedItem = navParams.get('item');
    this.icons = ['flask', 'wifi', 'beer', 'football', 'basketball', 'paper-plane',
```

```
    'american-football', 'boat', 'bluetooth', 'build'];
    this.items = [];
    for(let i = 1; i < 11; i++) {
      this.items.push({
        title: 'Item ' + i,
        note: 'This is item #' + i,
        icon: this.icons[Math.floor(Math.random() * this.icons.length)]
      });
    }
  }}

itemTapped(event, item) {
  this.navCtrl.push(ItemDetailsPage, {
    item: item
  });
}
```

Navigace

navigace funguje jako jednoduchý zásobník, nová stránka je funkcí *push* z komponenty NavController umístěna na vrchol zásobníku, aplikaci přejde na další stránku a zobrazí tlačítko zpět. Při návratu pomocí funkce pop odstraní stránku. Jelikož byl tento *navCtrl* nastaven v konstruktoru je možné volat *this.navCtrl.push()* a předat stránce na kterou se naviguje. Také je možné předat objekt obsahující data stránce, na kterou se naviguje. Použití *push* k navigaci na novou stránku je jednoduché. Více o navigaci je možné najít v dokumentaci Ionic a jeho příkladech.

Při využívání url funguje Ionic 2 rozdílně oproti Ionic 1. Místo využívání url k navigaci, se používá k ujištění, že je vždy možné vrátit se na stránku (například při spuštění aplikace). To znamená, že není limitováno používání href jen k navigaci. Pokud je to nutné je tu tato možnost.

8.5.1 Přidání stránky

První možnost přidání nové stránky je manuálně. Ukázáno na stránce jménem **hello-copy**, kdy se vytvoří složka stejného jména. U jména se používají pomlčky místo mezer tzv. kebab-casing konvence. Následně se přejde k jednotlivým souborům.

hello-copy.ts

Prvním řádkem je importovaná knihovna Angular. Dále je nutné vytvořit komponentu a té dát selektor a externí šablonu. Následuje vytvoření třídy, u ní se zápis více slovných názvů nepíše s pomlčkou, ale pomocí konvence PascalCasing (psaním prvního a následujícího slova velkým písmenem s koncovkou Page).

```
import { Component } from '@angular/core';

@Component({
  selector: 'page-hello-copy',
  templateUrl: 'hello-copy.html'
})
export class HelloCopyPage {
  constructor() {}
}
```

hello-copy.html

U šablony je nejdůležitější párovým tagem *ion-content*, který vykresluje umístěné prvky.

```
<ion-header>
  <ion-navbar>
    <button ion-button menuToggle>
      <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Hello Copy</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <h3>Test</h3>
</ion-content>
```

hello-copy.scss

Slouží pro přepsání výchozích stylů.

Po vytvoření stránky je nutné ji naimportovat v souboru **app.module.ts** její umístění, dále přidat třídu do deklarací (declarations) a vstupních komponent (entryComponent).

```
import { HelloCopyPage } from '../pages/hello-copy/hello-copy';
@NgModule({
  declarations: [ MyApp,HelloIonicPage,ItemDetailsPage,ListPage, HelloCopyPage],
  imports: [IonicModule.forRoot(MyApp)],
  bootstrap: [IonicApp],
  entryComponents: [MyApp,HelloIonicPage,ItemDetailsPage,ListPage, HelloCopyPage],
  providers: [{provide: ErrorHandler, useClass: IonicErrorHandler}]
})
export class AppModule {}
```

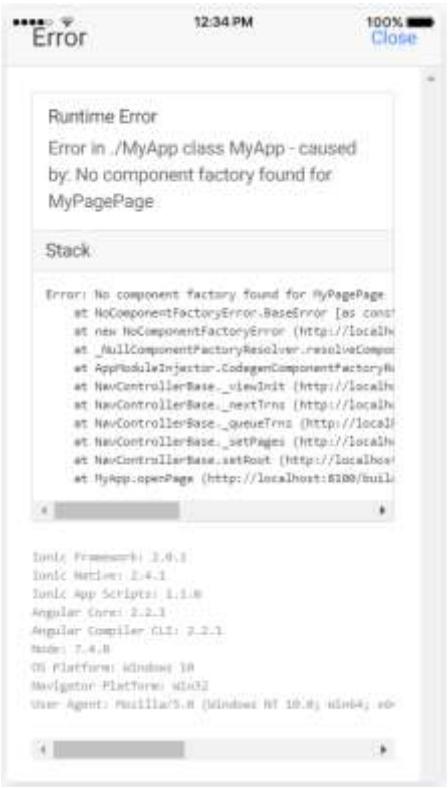
Následně je možné ji využít v souboru **app.component.ts**, kde se znovu importuje a přidá ke stránkám do postranního menu.

```
  this.pages = [
    { title: 'Hello Ionic', component: HelloIonicPage },
    { title: 'My First List', component: ListPage },
    { title: 'Hello Copy', component: HelloCopyPage}
  ];
}
```

Po obnovení aplikace se přidaná stránka zobrazí. Tento proces je poněkud zdlouhavý, přejatý z Angular. V Ionic existuje zjednodušená možnost k vygenerování všech potřebných souborů příkazem:

```
ionic g page nazev
```

dojde k vygenerování všech 3 souborů a zbývá jen přiřazení do **component** a **module**. V případě, opomenutí přidání do **module** a přidání jen do **component**, bude stránka sice zobrazena v menu, ale po rozkliknutí dojde k chybě (obr. 22).



Obrázek 22 - Zobrazení chyby

8.5.2 Přidání Cordova pluginu

Po opravení chyby stránka opět funguje. V této části bude ukázáno, na jakém principu se používají Cordova pluginy umožňující nativní funkce telefonu. Jednou z těchto funkcí může být použití fotoaparátu zařízení.

K prohledání Cordova pluginů existuje mnoho katalogů, které lze najít i na stránkách Ionic. V případě Ionic 2 není nutné stahovat jiné balíčky než Cordova. V Ionic 1, by bylo nutné stáhnout ještě Ionic Native a importovat do **index.html**. Příkazem níže se stáhne Cordova plugin zpřístupňující kameru.

```
ionic plugin add cordova-plugin-camera
```

Následně se přejde do soboru např. **hello-copy.ts** vytvořeného výše. Přidá se import stáhnutého pluginu.

```
import {Camera} from 'ionic-native';
```

V příkladu se pro jednoduchost bude uvažovat, že po vyfocení obrázku se zobrazí na konkrétním místě. Pokud by uživatel chtěl vyfotit další obrázek, přepíše původní. Vytvoří se veřejná proměnná typu *string*, do které se obrázek uloží. K vytvoření obrázku se napíše funkce například *takePhoto()*. Vložením připraveného kódu ze stránky se získá kostra, kde je třeba doplnit vyžadované parametry na místo *options*. Pomocí *destionationType* se určí kódování, následuje určení velikosti obrázku. A ošetří se chyba.

hello-copy.ts

```
public base64Image: string;

constructor() {}

takePhoto() {
  Camera.getPicture({
    destinationType: Camera.DestinationType.DATA_URL,
    targetHeight: 500,
    targetWidth: 500,
  }).then((imageData) => {
    this.base64Image = '' + imageData;
  }, (err) => {
    console.log(err);
    // Handle error
  });
}
```

hello-copy.html

V šabloně se umístí tlačítko, které po kliknutí vyvolá funkci focení. Dále se umístí karta (prostor) pro obrázek. [src] značí, že zdrojem obrázku je proměnná deklarovaná v předchozím souboru.

ngif je velice mocný nástroj k zobrazení nebo skrytí, kontrolující, zda je splněna podmínka a poté zobrazí obsah. Tím je postaráno o správný chod aplikace.

```
<ion-content padding>
  <h3>Test Camera</h3>
  <button (click)="takePhoto()">Camera</button>
  <ion-card>
    <img [src]="base64Image" *ngIf="base64Image">
  </ion-card>
```

8.5.3 Sestavení hotové aplikace

K testování aplikace slouží již zmíněný *ionic serve*, ale co v případě, kdy je potřeba testovat funkce jako fotoaparát. K tomu slouží varianta s emulací zařízení přímo v systému, program se zkompiluje a nahraje do emulátoru spuštěného z Android Studia příkazem.

```
ionic emulate android
```

V případě, že je k dispozici zařízení, je možné kód nahrát rovnou na něj.

```
ionic run android --device
```

Při práci na systému Windows, nelze kód přímo nahrát na zařízení iOS. To je možné obejít pomocí Ionic cloud. Po stažení aplikace **ionic view**, z **App Store** do zařízení, lze stáhnout hotový program z Ionic.

```
ionic upload
```

Předposledním krokem k hotové aplikaci je její sestavení, zmenšení a odstranění přebytečných funkcí, které se používají například u debugování aplikace. Tato verze se používá ke schválení aplikace do Google Play. Výsledek je uložený ve formě soboru apk.

`/platforms/android/build/outputs/apk/android-release-unsigned.apk`

```
ionic build android --prod --release
```

Poslední věcí je podepsání aplikace svým privátním klíčem. Který se vygeneruje příkazem kde se nahradí *alias_name* požadovaným jménem. Dále se dodržují instrukce programu.

```
keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

Když je privátní klíč vygenerovaný, uloží se do složky, kde byl spuštěn kód. Klíč je nutné umístit na bezpečné místo proti ztracení nebo krádeži. Jinak nebude možné dále aktualizovat aplikaci. Podepsání apk se provede v umístění souboru **android-release-unsigned.apk** a doplní se vytvořený *alias_name*.

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore android-release-unsigned.apk alias_name
```

Finální soubor se vytvoří programem *zipalign*, který se nachází v sdk složky Android, v případě bakalářské práce se nainstalovala do původní cesty.

```
C:\Users\User1\AppData\Local\Android\sdk\build-tools\25.0.2
```

```
zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk
```

Tím je dosaženo vytvoření hotového apk připraveného pro Play store. K nahrání je nutné mít vývojářský účet Google.

```
https://play.google.com/apps/publish/signup/
```

9. Aplikace

V této části se bude práce zabývat vytvořením aplikace, která bude komunikovat s libovolným cloudem (privátní, oficiální). Přes vybraný cloud se budou řídit příkazy do zařízení. Postup práce na aplikaci bude simulovat postupy a rozhodování, které mohou nastat během tvorby produktů.

9.1 Návrh a požadavky

Nejprve je důležité stanovit čeho má produkt dosáhnout. K tomu je důležité pokládat správné otázky, které by měl produkt splňovat.[35]

1. **Uživatelé**: kdo je ideální uživatel?

Ideální uživatel je mírně pokročilý umí upravit základní nastavení, týkající se zařízení. Viz. kapitola 7.

2. **Cíl**: co je jeho cílem, když si stáhl aplikaci?

Uživatel by si rád postavil chytrý domov a spravoval jej pomocí jediné aplikace.

3. **Úkoly**: jaké jsou jeho denní úkoly, pokud se přihlásí do aplikace?

Aplikace by se měla stát jeho nástrojem k automatizování činností, které uživatel chce mít rychle vyřízené a pod kontrolou.

Díky otázkám bylo možné upřesnit si požadavky a vlastnosti uživatele na správné cílení aplikace. Dále je dobré ujasnit si kolik a jaké stránky aplikace jsou nejdůležitější.

Jednou z nejdůležitějších stránek je navigace, aby se mohl uživatel pohybovat po aplikaci. Dále je doporučené mít velké nadpisy, uživatelé velmi rychle ztrácejí přehled, jaký odkaz je dostal na aktuální místo. Výraznější nadpis pomáhá v podobných případech. Mělo by být jasné, jak se dostat zpět, odkud se přišlo. Správně umístěná výrazná šipka by měla být samozřejmostí.

Když je vyřešená navigace, pokračuje se s pravidlem používat primárně jednu důležitou činnost na stránku. Nadbytečné činnosti by měli být ořezány a zpřístupněny na samostatných

[35] PORTMAN, Jane. THE ESSENTIAL USABILITY CHECKLIST FOR WEB APPS. In: InVision Blog [online]. San Francisco: InVision, 2016, 20 [cit. 2017-03-28].

stránkách. Úspěchem je ponechání nejdůležitější činnosti, tím zkrácení uživateli cestu o jeden klik/dotek.

Jsou případy, kdy je třeba mít více než jednu činnost na stránce současně, v takovém případě by důležitější činnost měla mít větší vliv při rozhodování.

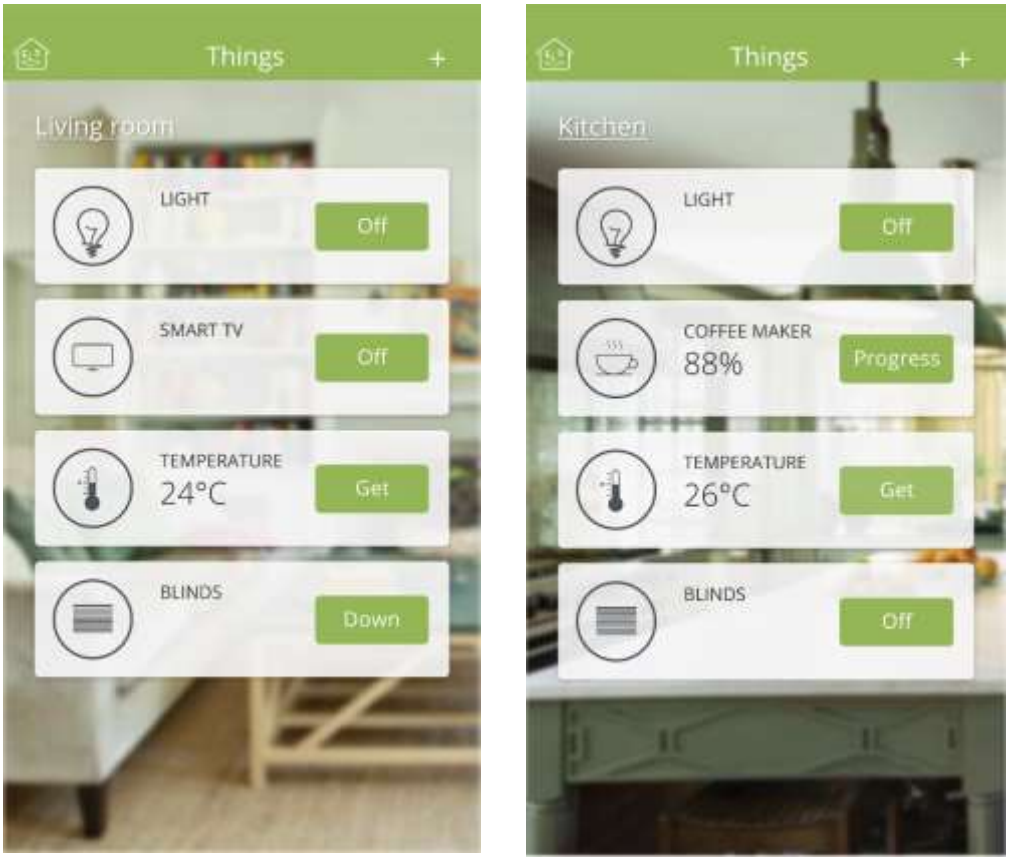
Dále se pokládají otázky například. Kam se uživatel dostane, když spustí aplikaci? Kam obvykle pokračuje? Níže je vypsáno, jak se při navrhování jednotlivých stránek postupovalo. Některé stránky byly vypuštěny.

Zobrazení zařízení - Things (hlavní stránka)

Tato stránka bude sloužit k zobrazení chytrých zařízení. Pravé tlačítko plus v navigační liště bude sloužit k přidání nového zařízení. Nalevo od titulu se bude nacházet tlačítko pro přechod do menu.

Je zde nastíněna možnost zobrazení zařízení ve „scénách“, kde by zařízení mohli být uspořádány podle místností v nichž se nacházejí. Došlo by k zjednodušení hledání a ovládání ve zcela zautomatizované domácnosti.

Každé zařízení bude mít reprezentační obrázek pro snazší orientaci, jednoduchý název a tlačítko k akci CTA (call to action). V případě, že zařízení posílá proměnné (teploměr), bude pod názvem zobrazena textová informace. Celý návrh je možné vidět na obrázku (obr. 23).



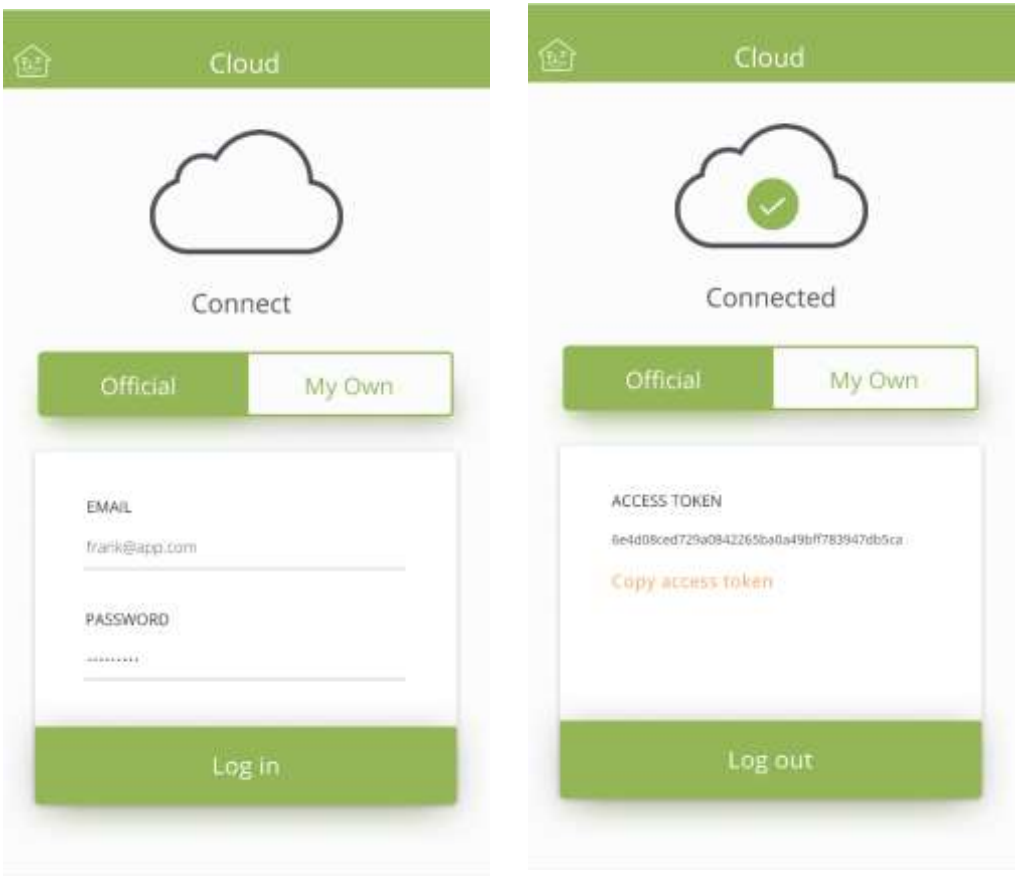
Obrázek 23 - Návrh zobrazení zařízení

Přidání zařízení

Tato stránka bude sloužit k přihlášení, zadání názvu, čísla zařízení a nastavení jeho funkce. Postupně by se funkce měli přidávat další funkce.

Výběr a přihlášení do Cloudu

Než je možné pracovat se zařízením, je potřeba se přihlásit na cloudu a tím vygenerovat token, který je vyžadován pro komunikaci se zařízením. Uživatel je upozorněn, pokud není přihlášený. Stránka nabízí výběr mezi oficiální a soukromým cloudem. Oficiální cloud vyžaduje email a heslo



Obrázek 24 - Přihlášení cloudu

pro zaslání požadavku o nový token. V případě vlastního cloudu je nabídka širší. Je potřeba vybrat protokol, IP adresu nebo doménu a přidat token ručně. Pokud uživatel nechce pracovat na aktuálním cloudu, může zadané informace bezpečně vymazat ze zařízení.

Z návrhů byl vytvořený interaktivní prototyp, kde se odzkoušely postupy a průchody aplikací.

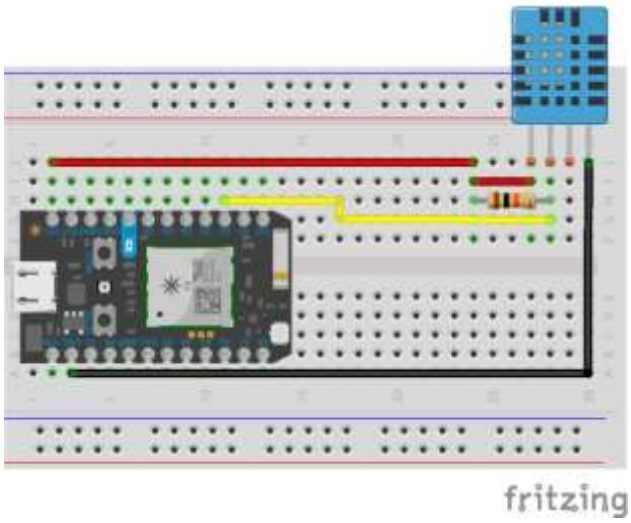
9.2 Zapojení

V této podkapitole budou zobrazeny zapojení prvků, které bude možné ovládat aplikací, včetně led popsané v *kapitole 7.2*.

9.2.1 Senzor teploty a vlhkosti vzduchu DHT11

DHT11 je základní, levný digitální senzor teploty a vlhkosti. Používá kapacitní čidlo vlhkosti a termistor pro měření okolního vzduchu. Poměrně snadno se ovládá. Senzor je schopný získat nová data až po 2 vteřinách od posledního měření, může být nevýhodou.[36] Zapojení Photonu (obr. 25)[37] Na CD *2-Particle_Photon/Zdrojovy_kod* se nachází zdrojový kód k ovládání součástky.

- 3 až 5V a I/O
- 2,5 mA maximální proud použití při konverzi (při vyžádání dat)
- Vhodné do prostředí 20% - 80% vlhkosti s 5% přesností
- Teploty 0-50 °C s přesností ± 2 °C
- Ne více než 1Hz rychlost vzorkování (jednou za sekundu)
- velikost 15,5mm x 12mm x 5,5mm
- 4 piny s 0,1" odstupem



Obrázek 25 - Zapojení DHT11

[36] DHT11 BASIC TEMPERATURE-HUMIDITY SENSOR. *Adafruit* [online]. New York City: adafruit, - [cit. 2017-04-10].

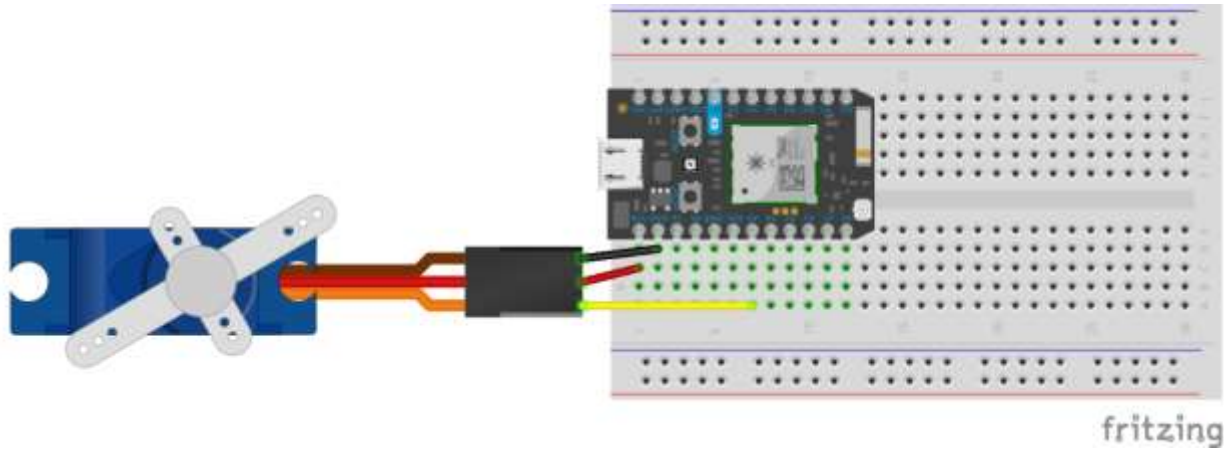
[37] Particle Weather Station Workshop. *Hackster.io* [online]. hackster.io: hackster.io, 2016 [cit. 2017-04-10].

9.2.2 Servomotor SG90

V druhém případě bude použit servomotor sloužící například k natočení rolet. Díky své vlastnosti natočení přesné polohy v limitu 180°.[38]

- Hmotnost: 9 g
- Rozměry: 23mm x 12,2mm x 29mm
- Provozní rychlost: 0,12sek / 60° (4,8 V)
- Provozní napětí: 4.8V
- Teplotní rozsah: 0 °C - 55 °C
- Napájení: přes externí adaptér
- Délka drátu: 25cm

Zapojení servo motoru (obr. 26). Na CD *2-Particle_Photon/Zdrojovy_kod* se nachází zdrojový kód k ovládání motorku.

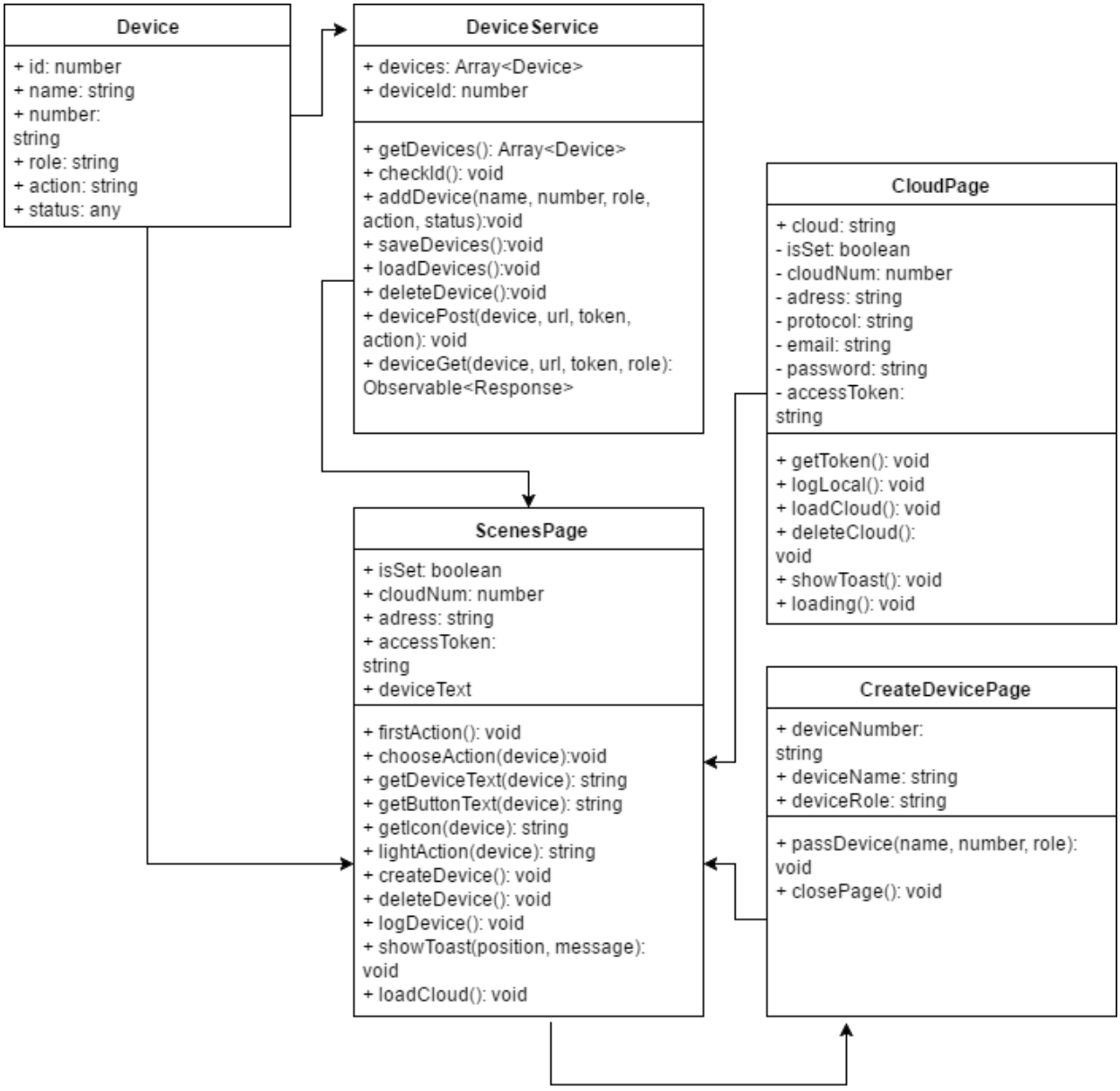


Obrázek 26 - Zapojení servo motoru

[38] SG90 Analog. *TowerPro* [online]. Torq Pro & Tower Pro: Torq Pro & Tower Pro, 2014 [cit. 2017-04-10].

9.3 Popis programu

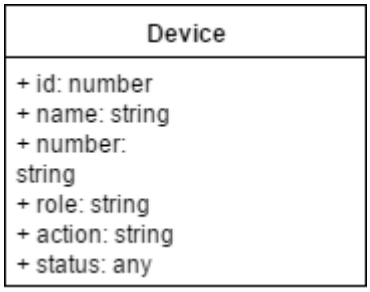
V této části bude popsán vývojový diagram aplikace (obr. 27) a vysvětleny použité funkce. Celý okomentovaný zdrojový kód aplikace se nachází na CD: *5-Aplikace.*



Obrázek 27 - Vývojový diagram

Device

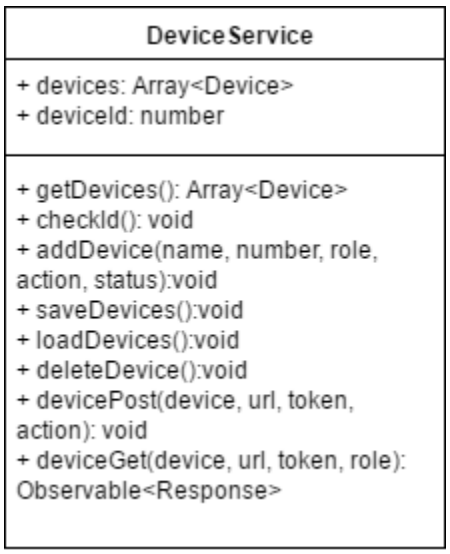
Třída *Device* (obr. 28) slouží jako reálná entita Photon, obsahuje unikátní číslo *id*, *name* (název) Photonu, *number* – sériové číslo potřebné k volání požadavků na cloud. *Role* je název proměnné, jenž rozhoduje, o jaké zařízení se jedná, váže se s produktem. V případě, že by se zařízení mělo chovat jako světlo, obsahovalo by označení „light/led“ a pomocnou proměnnou *status*, která udržuje poslední známý stav.



Obrázek 28 – Třída zařízení

DeviceService

Tato třída (obr. 29) obsahuje velice důležitou část celého programu. Jedná se o službu (service), která obstarává veškerou komunikaci mezi stránkami a jednotlivými zařízeními. Služba udržuje informace o všech zařízení, zde také vznikají. Obsahuje proměnnou *devices*, jedná se o pole zařízení. Mezi jeho metody patří *getDevices()* sloužící k vypsaní a debuggování třídy. Pomocí

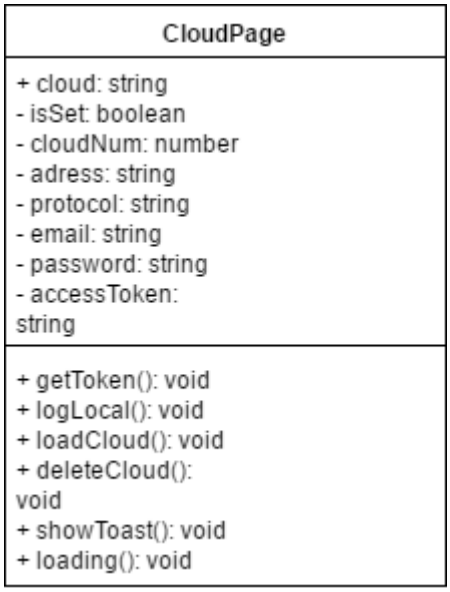


Obrázek 29 – DeviceService diagram

třídy *addDevice* se převezmou parametry a vytvoří se dočasná proměnná let, uchovávající zařízení, než je pomocí metody *push* přidána jako objekt *Device* do pole. Jelikož se jedná o webovou aplikaci, samotné html stránky nejsou schopny udržet informaci a zařízení by se po přechodu na jinou stránku ztratilo. Pomocí metody *saveDevices()* je schopné tuto informaci udržet. V metodě se zavolá funkce *set* na uložiště, které Ionic vybere jako nejlepší řešení (SqLite, indexovanou databázi, WebSQL, lokální uložiště) v tomto pořadí a uloží jej. Při spuštění programu a každém přechodu mimo hlavní stránku je potřeba znovu načíst uložená zařízení pomocí *loadDevices()*. Metody *devicePost* a *deviceGet* mají za parametry jednotlivé zařízení, url uloženého cloudu, vygenerovaný přístupový token a rozdílné parametry, určující, jaký argument se má vykonat.

CloudPage

Třída *CloudPage* slouží k výběru, získání a uložení informací na jaký cloud se má aplikace připojit. Proměnná *cloud*, slouží k výběru mezi oficiálním a lokálním serverem a využívá se hlavně pro nastavení html šablony, proměnná *cloudNum* pak ukládá stav. *isSet* značí, zda je server uložený. U oficiálního cloudu stačí pouze *email* a *password*. V případě lokálního cloudu je zapotřebí mít *protocol*, *address* a napsaný *accessToken*, jelikož lokální server zatím neumožňuje generování dotazem. Mezi metody patří získání tokenu z oficiálního cloudu *getToken(),* v případě lokálního uložení *logLocal()*. Jako u předchozí třídy *device* je nutné tyto informace uložit pomocí



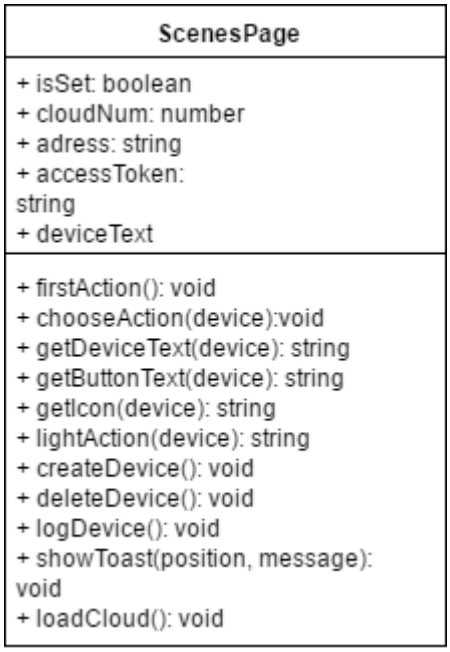
Obrázek 30 – Cloud diagram

permanentní paměti *storage* a znovu je načíst metodou *loadCloud()*. Pokud se uživatel rozhodne odhlásit, vše je bezpečně smazáno *deleteCloud()*.

ScenesPage

Třída *ScenesPage* slouží jako vstup do aplikace, stará se o zobrazení přidaných zařízení a volání funkcí po stisknutí tlačítka. Obsahuje podobné proměnné jako *CloudPage* pro uchování informace o připojení, které pomocí metody *loadCloud()* načítá z uloženého místa. Do třídy je importována služba *DeviceService* starající se o chod zařízení. Služba je následně volána v metodách třídy *ScenesPage*. Pro vytvoření nového zařízení je možné službou *ModalController* zobrazit okno s novou stránkou *CreateDevicePage*, přes kterou uživatel zadá požadavek na přidání zařízení, metodou *createDevice()*. Možností je i smazat zařízení *deleteDevice()*.

Při spuštění stránky proběhne jako první načtení zařízení službou *loadDevice(),* během načítání se třída *ScenesPage* postará metodami *getButtonText(device)* a *getIcon(device)* o vizuální podobu a doplní správné parametry obrázkům a nápisy tlačítkům. Následně se nahrají informace o cloudu. Po stisknutí tlačítka zařízení se vykoná metoda vyhodnocující roly zařízení *chooseAction(device)* a překontroluje se, zda má stránka potřebné informace o cloudu. Bez údajů



Obrázek 31 - Scenes diagram

nepokračuje dále. Šetří tím nápor chybných dotazů na server. Podle role a posledního známého

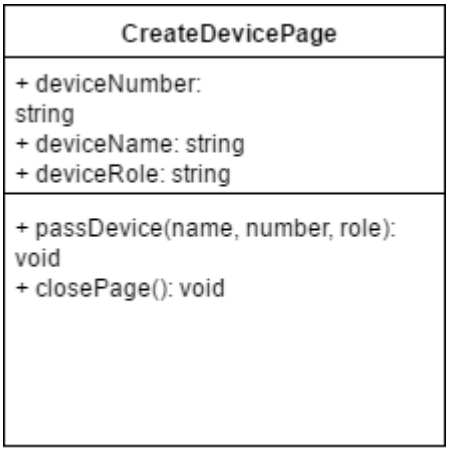
stavu *status* jsou vnitřním funkcím přidány dodatečné parametry a zavolána služba k obstarání komunikace s cloudem.

Pokud zařízení slouží jako senzor, posílá dotazy typu *get*, při spuštění stránky zkusí zavolat metodu *firstAction()* a stáhne hodnoty svých proměnných, které pak zobrazí pomocí *getDeviceText(device)* v kartičce u zařízení.

CreateDevicePage

Tato třída není volána na přímo z odkazu menu, ale slouží k zobrazení v modálním oknu ze stránky *ScenesPage*. Stránka obsahuje formulář. Po vyplnění se metodou *passDevice(..)* zkontroluje správnost obsahu (správná délka sériového čísla, zda má zařízení vyplněné jméno) a odešle se ve formátu JSON zpět na stránku *ScenesPage*, ta se postará o vytvoření objektu *Device*.

Uživatel má možnost z modálního okna odejít, k tomu slouží *closePage()*, nedochází k předání požadovaných dat, program musí být ošetřen, aby nedošlo k chybě.



Obrázek 32 – CreateDevicePage diagram

Celý okomentovaný zdrojový kód je možné nalézt na přiloženém CD.

9.4 Přechod na Ionic 3

Týden před odevzdáním práce vyšla nová stabilní verze Ionic 3. Přechod není tak výrazný, jako byl mezi verzemi 1 a 2. Hlavní novinkou je nová verze Angular 4, Google pozměnil číslování a přeskočil verzi 3.
Angular 4 se soustředí na zmenšení velikosti aplikace a tím ušetření paměti zařízení. Změny s sebou nesou i celkové zrychlení aplikace. Zmenšení bylo docíleno pomocí nového View Engine, generujícího kód. Snížení dosahuje až 60%, obecně platí, čím více komplexní šablony jsou, tím větší je zmenšení. Angular byl aktualizován na novější verzi TypeScript 2.1 kompatibilní s 2.2, urychlující překlad a lepší kontrolu typů.[39]

Souhrn animací, který byl obsažen v balíčku @angular/core byl přemístěn, došlo k zmenšení velikosti v samotném Angular. Dále byly vylepšeny syntaxe *ngIf a *ngFor, které jsou schopny pojmout if/else syntaxi a přiřadit lokální proměnnou například:

```
<div *ngIf="userList | async as users; else loading">
 <user-profile *ngFor="let user of users; count as count" [user]="user">
 </user-profile>
<div>{{count}} total users</div>
</div>
<ng-template #loading>Loading...</ng-template>
```

V Ionic 3 za zmínku stojí hlavně nový dekorátor IonicPage, který předává deep link konfigurace do hlavního root modulu aplikace. Tento způsob umožňuje využití tzv. *lazy loadingu aplikace* nastavení priority načítání stránek a úpravu individuálních stránek.[40]

```
@IonicPage({ name: 'my-about', segment: 'about-page' })
@Component({ selector: 'page-about', templateUrl: 'about.html' })
export class AboutPage { }
```

[39] Angular 4.0.0 Now Available. In: *Angular* [online]. -: -, 2017 [cit. 2017-04-11].
[40] Ionic 3.0 has Arrived!. In: *Ionic* [online]. San Francisco: Ionic, 2017 [cit. 2017-04-11].

Pro přechod na nejaktuálnější verzi stačí aktualizovat Ionic.

```
npm install -g cordova ionic
```

Smazat **node-modules**, upravit soubor **package.json** a nainstalovat.

```
npm install
```

```
"dependencies": {
    "@angular/common": "4.0.0",
    "@angular/compiler": "4.0.0",
    "@angular/compiler-cli": "4.0.0",
    "@angular/core": "4.0.0",
    "@angular/forms": "4.0.0",
    "@angular/http": "4.0.0",
    "@angular/platform-browser": "4.0.0",
    "@angular/platform-browser-dynamic": "4.0.0",
    "@ionic-native/core": "3.4.2",
    "@ionic-native/splash-screen": "3.4.2",
    "@ionic-native/status-bar": "3.4.2",
    "@ionic/storage": "2.0.1",
    "ionic-angular": "3.0.1",
    "ionicons": "3.0.0",
    "rxjs": "5.1.1",
    "sw-toolbox": "3.4.0",
    "zone.js": "^0.8.4"
},
"devDependencies": {
  "@ionic/app-scripts": "1.3.0",
  "typescript": "~2.2.1"
}
```

Poté importovat BrowserModule a Http v app/app.module.ts a přidat do importu ve stejném souboru.

```
import { BrowserModule } from '@angular/platform-browser';
import { HttpModule } from '@angular/http';
imports: [
  BrowserModule,
  HttpModule,
  IonicModule.forRoot(MyApp)
],
```

Závěr

Cílem bakalářské práce bylo popsání postupů, jak zprovoznit místní cloud. Dále sepsání metodiky k vytvoření aplikace v Ionic 2 s následnou demonstrací umožňující komunikovat s cloudem, který ovládá zařízení Photon.

V teoretické části byly formou rešerše vysvětleny potřebné základy problematiky Internet of Things, následně byl popsán mikrokontroler Photon Particle jeho výhody nad konkurencí, architektura zařízení a příkazy k ovládání programem cli v konzoli. Po ujasnění pojmu cloud byly popsány možnosti, které nabízí celá integrace produktu na platformě Particle. Nakonec bylo odůvodněno, jaké výhody s sebou nese vývoj aplikací využitím webových frameworků, z kterých byl vybrán pro svou aktuálnost framework Ionic 2. Po vysvětlení teoretického aparátu byly v praktické části sepsány kroky, k docílení funkčního řešení.

Mezi hlavní přínosy práce patří zjednodušení a sepsání funkčních postupů k zprovoznění lokálního cloudu na více systémech a verzích. Verze cloudu spustitelného na sytému Linux jde snadno využít například na zařízení Raspberry Pi, tím je možné získat svůj vlastní malý cloud s nízkou spotřebou, který může běžet v každé domácnosti. Využití oficiálního cloudu má také spoustu možností, lze integrovat s cloudy velkých hráčů jako je Google a umožnit, tak zařízením pokročilé analýzy a funkce strojového učení. Pomocí platformy IFTTT je možné získat stovky propojených webových služeb a aplikací. Dalším přínosem práce je popsání vývoje aplikace od vytvoření projektu až po dokončení a podepsání finálního balíčku připraveného pro Google Play. Vývoj ukázal spoustu výhod využití hybridních frameworků, které se s posledními verzemi blíží kvalitě nativních aplikací.

Poslední část práce se týkala spojení všech předešlých celků v hotové řešení. Prostor pro rozvoj aplikace formou přidání nových funkcí zařízení, které je schopné vykonat nebo vytvoření volného editoru funkcí v aplikaci pro neomezené možnosti, bez zasahování do zdrojového kódu. Obecně napsané dotazy ve službě zařízení jsou na tuto funkci připravené. Dalším možným rozšířením může být přidávání a řazení zařízení v jednotlivých scénách. Kosmetické části jako možnost nahrávání fotek svých míst, které by sloužily jako pozadí.

Práce byla doplněna o aktualizaci na verzi Ionic 3, která vyšla pár týdnů nazpět. Verze přináší další snížení nároků a zvýšení výkonu. Nakonec zbývá konstatovat, že všechny cíle bakalářské práce byly úspěšně splněny.

Přílohy

Přiložené CD obsahuje elektronickou verzi práce ve formátu PDF, programy k instalaci mikrokontroleru Photon. Současně obsahuje programy k instalaci privátního cloudu. Dále obsahuje okomentované zdrojové kódy k ovládání Photonu. Výsledná aplikace je přiložena ve formě Ionic projektu.

V adresáři *1-Dokument* je umístěna elektronická verze bakalářské práce.
V adresáři *2-Particle_Photon/Instalace* jsou veškeré programy k zprovoznění Photonu.
V adresáři *2-Particle_Photon/Zdrojovy_kod* jsou umístěny řídící programy ve formátu INO.
V adresáři *3-Privatni_Cloud* jsou veškeré programy k zprovoznění soukromého cloudu na systému Windows.
V adresáři *4-Cordova* se nacházejí instalační programy vyžadované pro vytvoření Android aplikace v Cordově.
V adresáři *5-Aplikace* je umístěna hotová aplikace s okomentovaným kódem.

Literatura/Reference

1. Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap. *Adobe* [online]. Los Angeles: Adobe, 2011, s. 1 [cit. 2016-04-15]. Dostupné z: http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html

2. *An Introduction to the Internet of Things (IoT)* [online]. 1. San Francisco, California: Lopez Research, 2013 [cit. 2015-11-21]. Dostupné z: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/introduction_to_IoT_november.pdf. S. 2-3.

3. Angular 4.0.0 Now Available. In: Angular [online]. -: -, 2017 [cit. 2017-04-11]. Dostupné z: http://angularjs.blogspot.cz/2017/03/angular-400-now-available.html

4. Apache Cordova: Overview. In: *Cordova* [online]. -: The Apache Software Foundation, 2016 [cit. 2016-12-03]. Dostupné z: http://cordova.apache.org/docs/en/latest/guide/overview/index.html

5. *Blink an LED*. Particle Docs [online]. San Francisco: Particle, 2016 [cit. 2016-04-08]. Dostupné z: https://docs.particle.io/guide/getting-started/examples/photon/

6. BRADLEY, Adam. Where does the Ionic Framework fit in? - [online]. 2013, 2013(1), 1 [cit. 2016-07-31]. Dostupné z: http://blog.ionic.io/where-does-the-ionic-framework-fit-in/

7. Compiler. Compilers.net [online]. -: GNU Free Documentation License, 2005 [cit. 2017-02-10]. Dostupné z: http://www.compilers.net/paedia/compiler/index.htm

8. DHT11 BASIC TEMPERATURE-HUMIDITY SENSOR. *Adafruit* [online]. New York City: adafruit, - [cit. 2017-04-10]. Dostupné z: https://www.adafruit.com/product/386

9. EVANS, Dave. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything* [online]. 2011, s. 1 [cit. 2015-11-22]. Dostupné z: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

10. EVANS, Dave. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything [online]. 2011, 2015-11-22, 1. Dostupné z: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

11. FERMOSO, Jose. *PhoneGap Seeks to Bridge the Gap Between Mobile App Platforms. In: Gigaom* [online]. -: Knowingly, 2009 [cit. 2016-04-15]. Dostupné z: https://gigaom.com/2009/04/05/phonegap-seeks-to-bridge-the-gap-between-mobile-app-platforms/

12. Ionic [online]. -: Ionic, MIT, 2016 [cit. 2016-12-03]. Dostupné z: http://ionicframework.com/?_ga=1.211686805.1614954307.1467623137

13. Ionic 3.0 has Arrived!. In: *Ionic* [online]. San Francisco: Ionic, 2017 [cit. 2017-04-11]. Dostupné z: http://blog.ionic.io/ionic-3-0-has-arrived/

14. ITU-T Y.2060. SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS AND NEXT-GENERATION NETWORKS: Next Generation Networks – Frameworks and functional architecture models [online]. 06/2012. s. 1 [cit. 2016-02-04]. Dostupné z: https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=y.2060

15. JAVŮREK, Karel. *IDC: internet věcí bude větší trh než smartphony a tablety dohromady* [online]. , 1 [cit. 2016-03-28]. Dostupné z: http://connect.zive.cz/clanky/idc-internet-veci-bude-vetsi-trh-nez-smartphony-a-tablety-dohromady/sc-320-a-181851/default.aspx

16. KELLY, Jeff a David FLOYER. The Industrial Internet and Big Data Analytics: Opportunities and Challenges. In: Wikibon [online]. -: Wikibon, 2013-9-16 [cit. 2016-11-11]. Dostupné z: http://wikibon.org/wiki/v/The_Industrial_Internet_and_Big_Data_Analytics:_Opportunities_and_Challenges

17. LYNCH, Max. Announcing Ionic 2.0.0 Final. Blog Ionic [online]. San Francisco: Ionic, 2017 [cit. 2017-02-10]. Dostupné z: https://blog.ionic.io/announcing-ionic-2-0-0-final/?_ga=1.133678280.718266716.1485785318

18. M. Ersue, Ed., D. Romascanu, J. Schoenwaelder a A. Sehgal. *Management of Networks with Constrained Devices: Use Cases draft-ietf-opsawg-coman-use-cases-02* [online]. -, 2014 [cit. 2016-03-29]. Dostupné z: https://tools.ietf.org/html/draft-ietf-opsawg-coman-use-cases-02

19. MELL, Peter a Timothy GRANCE. *The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology* [online]. 1. -: U.S. Department of Commerce, 2011 [cit. 2016-05-13]. Dostupné z: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

20. MORONY, Joshua. 7 Reasons Why Ionic 2 Is Better Than Ionic 1. Josh Morony [online]. 2016 [cit. 2017-02-13]. Dostupné z: https://www.joshmorony.com/7-reasons-why-ionic-2-is-better-than-ionic-1/

21. Odhad vývoje IoT. In: IDC [online]. IDC Research, Inc.: IDC, 2016 [cit. 2016-03-28]. Dostupné z: http://www.idc.com/infographics/

22. Particle docs. Particle [online]. San Francisco: Particle, 2016 [cit. 2016-02-20]. Dostupné z: https://docs.particle.io/reference/api/

23. Particle Weather Station Workshop. *Hackster.io* [online]. hackster.io: hackster.io, 2016 [cit. 2017-04-10]. Dostupné z: https://www.hackster.io/pjdecarlo/hackster-live-event-particle-weather-station-workshop-452572

24. PORTMAN, Jane. THE ESSENTIAL USABILITY CHECKLIST FOR WEB APPS. In: InVision Blog [online]. San Francisco: InVision, 2016, 20 [cit. 2017-03-28]. Dostupné z: http://blog.invisionapp.com/usability-checklist-for-web-apps/

25. SG90 Analog. *TowerPro* [online]. Torq Pro & Tower Pro: Torq Pro & Tower Pro, 2014 [cit. 2017-04-10]. Dostupné z: http://www.towerpro.com.tw/product/sg90-7/

26. SHELBY, Z., K. HARTKE a C. BORMANN. The Constrained Application Protocol (CoAP). *The Internet Engineering Task Force* [online]. Universitaet Bremen TZI: Universitaet Bremen TZI, 2014 [cit. 2016-12-19]. Dostupné z: https://tools.ietf.org/html/rfc7252

27. *Stop Using Internet Protocol Version 4!* [online]. CXO Media Inc. a subsidiary of IDG Enterprise, 2014 [cit. 2016-03-10]. Dostupné z: http://www.cio.com/article/2376632/internet/stop-using-internet-protocol-version-4-.html

28. *The Development of Mobile Applications using HTML5 and PhoneGap* on Intel® Architecture-Based Platforms. Intel Developer Zone* [online]. 2012 [cit. 2016-04-15]. Dostupné z: https://software.intel.com/en-us/articles/the-development-of-mobile-applications-using-html5-and-phonegap-on-intel-architecture-based

29. The Particle Cloud. *Particle* [online]. San Francisco: Particle, 2016 [cit. 2016-05-13]. Dostupné z: https://www.particle.io/products/platform/particle-cloud

Seznam obrázků

Seznam tabulek