VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# BAKALÁŘSKÁ PRÁCE

Brno, 2016 Matej Semančík


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# VYUŽITÍ PLATFORMY INTEL GALILEO V INTERNET OF THINGS
INTEL GALILEO PLATFORM USAGE IN THE INTERNET OF THINGS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS


AUTOR PRÁCE
AUTHOR

Matej Semančík

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Ondřej Krajsa, Ph.D.



BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

***Student:*** Matej Semančík ***ID:*** 164394
***Ročník:*** 3 ***Akademický rok:*** 2015/16

**NÁZEV TÉMATU:**

## Využití platformy Intel Galileo v Internet of Things

**POKYNY PRO VYPRACOVÁNÍ:**

Analyzujte možnosti platformy Intel Galileo, její praktické využití a porovnejte s Platformou Arduino.
Navrhněte a realizujte testovací senzorovu síť pro porovnání obou platforem.

**DOPORUČENÁ LITERATURA:**

[1] WAHER, Peter. Learning Internet of Things. Olton Birmingham: Packt Publishing, 2015. ISBN 9781783553532.

[2] DE SOUSA, Miguel. Internet of Things with Intel Galileo. Packt Publishing, 2015. ISBN 9781782174585.

***Termín zadání:*** 1.2.2016 ***Termín odevzdání:*** 1.6.2016

***Vedoucí práce:*** Ing. Ondřej Krajsa, Ph.D.
***Konzultant bakalářské práce:***

**doc. Ing. Jiří Mišurec, CSc.**, *předseda oborové rady*

## ABSTRAKT

Táto práca sa zaoberá využitím platformy Intel Galileo v IoT (Internet of Things) a jej porovnaním s ostatnými platformami, spôsobmi komunikácie v IoT, senzormi a senzorickými sieťami, spôsobmi implementácie tejto platformy, návrhom a realizáciou testovacej topológie.

## KĽÚČOVÉ SLOVÁ

Internet vecí, IoT, Intel Galileo, Arduino, senzorické siete, senzory

## ABSTRACT

This thesis is about practical usage of Intel Galileo board in Internet of Things and it's comparison to other development platforms, communication types in IoT, sensors and sensor networks, implementation examples, proposal and implementation of testing topology.

## KEYWORDS

Internet of Things, IoT, Intel Galileo, Arduino, sensor networks, sensors

SEMANČÍK, Matej *Využití platformy Intel Galileo v Internet of Things*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 53 s. Vedúci práce bol Ing. Ondřej Krajsa, Ph.D.

# PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „Využití platformy Intel Galileo v Internet of Things“ vypracoval(a) samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ..... .....

podpis autora(-ky)

# POĎAKOVANIE

Rád by som poďakoval vedúcemu práce pánovi Ing. Ondřejovi Krajsovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.


Brno . . . . . .

. . . . . .

podpis autora(-ky)


Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

## POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)


EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

# OBSAH

# ZOZNAM OBRÁZKOV

# ZOZNAM TABULIEK

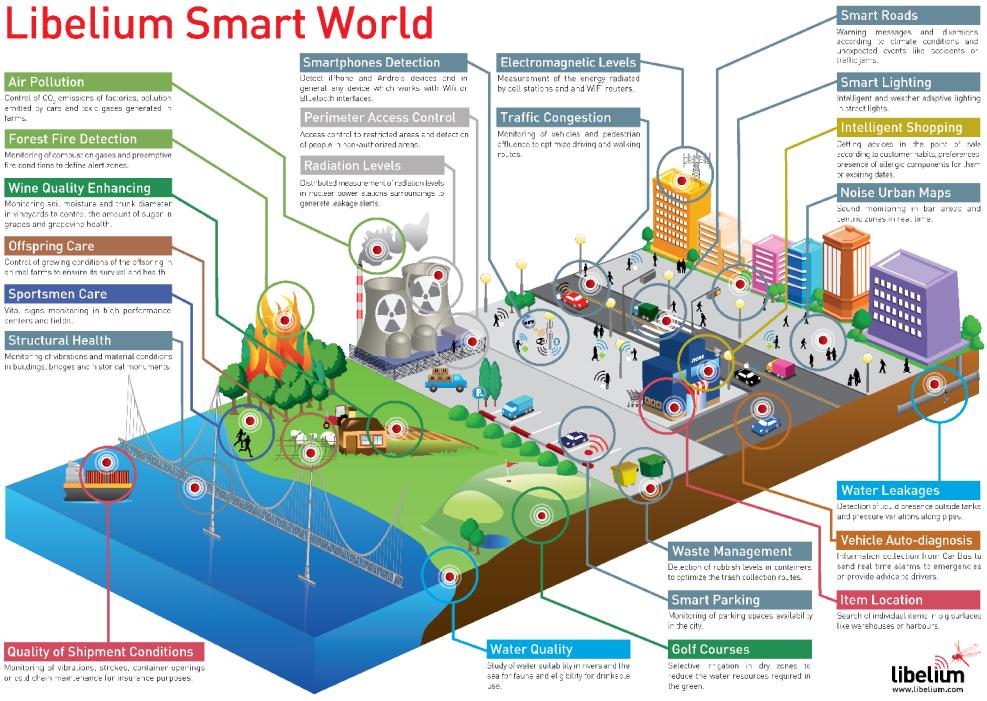
# ÚVOD

Táto práca sa venuje pojmu IoT (Internet of Things - Internet vecí) a návrhu využitia vývojovej platformy Intel Galileo v IoT. Budem v nej popisovať túto platformu a uvediem aj ostatné populárne platformy pre vývoj v IoT. Venujem sa aj spôsobom komunikácie medzi zariadeniami a protokolmi využívanými v IoT, rozoberiem implementačné detaily Galilea a spôsob komunikácie tejto dosky s internetovými službami na spracovanie a vizualizáciu dát. Uvediem tiež príklady využitia tejto dosky v IoT (priemysel a pod.) a navrhnem testovaciu schému, ktorú by som chcel realizovať v nadväzujúcej bakalárskej práci.

# 1 INTERNET OF THINGS

Internet of Things, alebo v preklade *Internet vecí* je pojem, o ktorom sa v poslednom čase vraví stále častejšie. Je to koncept, ktorý má veľký potenciál ovplyvniť to, ako žijeme a pracujeme. Ide v podstate o to, že sa snaží pripojiť akékoľvek zariadenia do Internetu – telefóny, chladničky, kávovary, svetlá, „wearables“ (zariadenia na oblečení), ale aj rôzne senzory na monitorovanie teploty a vlhkosti, meteorologické stanice, dokonca aj motory lietadiel či automobily so vstavanými senzormi alebo senzory ktoré by pomáhali hasičom lokalizovať dôležité miesta pri záchranách – to všetko by sa malo stať súčasťou tejto veľkej siete a čo je hlavné – mali by vedieť medzi sebou komunikovať, vymieňať si informácie a na základe toho nám uľahčiť život, či prácu. Na začiatok uvediem jeden zaujímavý príklad.

Jeden z populárnych konceptov založených na IoT sú inteligentné mestá, ktoré by nám pomáhali znížiť mieru odpadu, zvýšiť efektivitu elektrickej spotreby a celkovo, pomôcť nám porozumieť tomu ako v mestách žijeme. Príklad môžeme vidieť na obrázku 1.1 od firmy Libelium, ktorá sa práve touto tématikou zaoberá a vyvíja IoT zariadenia pre tento účel, napr. na monitorovanie kvality vody, alebo hodnoty hluku v mestských častiach.




Obr. 1.1: Libelium Smart City

## 1.1 Využitie IoT v reálnom svete

IoT zažíva veľkú expanziu vo svete. V dnešnej dobe má výborné uplatnenie v priemysle a domácnostiach. V inteligentných domácnostiach existuje veľa možností čo ovládať a monitorovať. Vieme automatizovať rôzne aspekty domácnosti, napríklad žalúzie, osvetlenie, monitorovať teplotu a vlhkosť, ovládať termostaty a pod. Keďže podstata IoT spočíva v tom, že zariadenia vedia medzi sebou komunikovať, vznikajú tu rôzne zaujímavé možnosti uplatnenia. Napríklad, inteligentný dom, ktorý by vedel regulovať svoju teplotu podľa tej vonkajšej, alebo podľa predpovede počasia. Na diaľku by sme vedeli ovládať svetlá, alebo monitorovať elektrickú spotrebu.

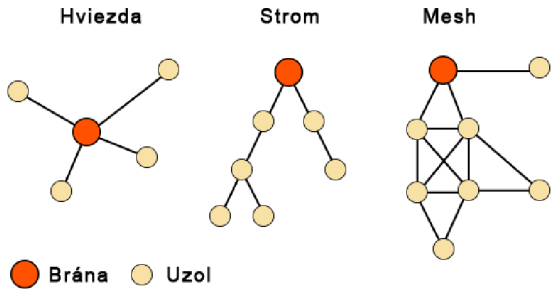
Ďalšie využitie je v priemysle. V poslednom čase sa vraví o takzvanej 4. priemyselnej revolúcii, alebo tiež názve Priemysel 4.0. Spočíva v tom, že prácu, ktorú vedia robiť stroje by mali robiť práve stroje. Vzniká koncept inteligentných tovární, ktoré prevezmú jednoduché opakjúce sa činnosti, ktoré doteraz vykonávali ľudia. Budú vedieť informovať o stave skladu, stroje a produkty dostanú čipy, vďaka ktorým ich budeme môcť vedieť ovládať cez internet.[14] Táto problematika je však tiež trochu kontroverzná a dosť sa spája s tým, že veľa menej kvalifikovaných ľudí by mohlo prísť o prácu.

Zaujímavým konceptom sú inteligentné mestá. Predstavme si napríklad čip v každej lampe, ktorý by hlásil, či je lampa pokazená, treba jej vymeniť žiarovku, alebo by ju vedel zapnú/vypnúť. Na cestách by boli senzory monitorujúce dopravu a na základe ich dát by sa jej vedeli semafory prispôsobovať za účelom zvýšenia efektivity, dostávali by sme do mobilov správy o stave premávky. Myslím si, že je nespočetne veľa možností využitia s takýmto potenciálom, keďže všetky zariadenia by medzi sebou vedeli komunikovať.

## 1.2 IoT a senzorické siete

Senzorické siete sa skladajú z menších uzlov, ktoré slúžia na zbieranie dát z rôznych senzorov. Tieto uzly ich ďalej posielajú hlavnému uzlu (tzv. brána), ktorý ich všetky zozbiera a ďalej spracuje. Existuje viacero zapojení takýchto sietí, napríklad stromové zapojenie, zapojenie do hviezdy alebo mesh.[9] V našom prípade môže byť brána práve vývojová doska s procesorom komunikujúca s jednotlivými uzlami. Existuje určitá spojitosť medzi IoT a senzorickými sieťami. IoT je postavené práve na senzorických sieťach, ktoré sú pripojené do internetu, čím sa rozširuje možnosť ich použitia v inteligentných zariadeniach. Dobrý príklad využitia sú inteligentné domácnosti, kde si vieme na diaľku zistiť stav teploty v izbe, aktuálnu spotrebu elektriny alebo vlhkosť pôdy v záhrade. Senzory môžu byť medzi sebou prepojené po kábli, no populárnejšia a lepšie škálovateľná metóda je bezdrôtová komunikácia.

Vtedy sa už vraví o bezdrôtových senzorických sieťach, pri ktorých stojí za zmienku spomenúť zapojenie typu *mesh*. Toto zapojenie spočíva v tom, že jednotlivé uzly sú medzi sebou vzájomne poprepájané, čím vzniká redundantná sieť s mnohými cestami. Všetky uzly nemusia byť priamo pripojené na bránu, ale vedia si vytvoriť cestu k bráne pomocou ostatných uzlov s ktorými komunikujú. Týmto spôsobom môžeme umiestniť senzory aj na miesta, v ktorých by ináč nemali dosah na hlavný uzol.




Obr. 1.2: Typy zapojení senzorických sietí

# 2 DOSTUPNÉ VÝVOJOVÉ PLATFORMY

Ak chceme začať vyvíjať zariadenie pre Internet of Things, je dobré použiť jednu z dostupných vývojových platforiem. Existuje viacero vývojových platforiem pre IoT, medzi najznámejšie patria **Arduino, Intel Galileo, Raspberry Pi, či BeagleBone**. Hlavnou podmienkou samozrejme je, aby malo zariadenie možnosť pripojenia do internetu či už ethernet rozhraním alebo WiFi kartou.

## 2.1 BeagleBone

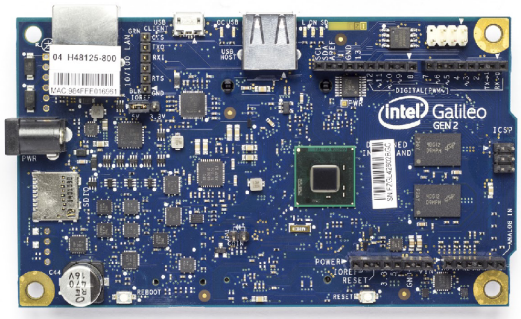
BeagleBone je malý počítač veľkosťou približne zhodný s kreditnou kartou. Na doske má osadený 32bitový procesor ARM Cortex-A8 s dvoma jadrami bežiacimi na frekvencii 720 MHz. Názov procesora nám už napovedá, že sa jedná o ARM architektúru, ktorá je špecifická svojou rozšírenou inštrukčnou sadou (RISC). Dokážeme na ňom spustiť rôzne distibúcie Linuxu a dokonca aj Android. Operačný systém sa bootuje z SD karty. BeagleBone ďalej ponúka 256 MB DDR2 RAM, má 2 USB porty (host a client port), Ethernet rozhranie a dokopy až 92 pinov slúžiacich na pripojenie rôznych perif0ri9 ako sú senzory, displeje a pod.

## 2.2 Arduino

Arduino je open-source prototypovacia platforma založená na ľahko použiteľnom hardvéri a softvéri. Arduino dosky vedia čítať vstupy – svetlo na senzore, prst na tlačidle, alebo Twitter správu – a previesť ich na výstup – aktivácia motora, zapnutie LED diódy, zverejnenie niečoho online.[5] Arduino sa delí na viacero modelov – Uno, Yún (tento model bootuje aj Linux), Mega, Nano a pod., ktoré sa líšia rozmermi, výkonom a osadenými komponentami a podporuje pripojenie rôzných doplnkov, tzv. shieldov alebo štítov, ktoré sa nasadia na dosku a plnia určitú funkciu – napríklad Ethernet shield ktorý umožňuje používať ethernetové rozhranie, alebo rôzne senzorické shieldy na pripojenie rôznych typov senzorov. Arduino je bežne dostupné v ČR od rôznych predajcov za približne 600–1500,- CZK podľa modelu. Keďže Arduino je open-source, môže si ho vyrobiť a upraviť každý. Týmto spôsobom vznikajú aj lacnejšie modely dostupné napr. z Číny. V tejto semestrálnej práci sa zameriam na použitie nižšie popísanej vývojovej dosky Intel Galileo druhej generácie, ktorá je práve kompatibilná s Arduinom.

## 2.3 Intel Galileo

Galileo je pomerne nový produkt od spoločnosti Intel, ktorý má veľa spoločného s Arduinom. Je hlavne hardvérovo kompatibilné, čo znamená, že má rovnaké rozloženie pinov. To mu umožňuje pripájať Arduino shieldy a taktiež na ňom dokáže bežať Arduino kód. Hlavným rozdielom je však architektúra. Galileo je postavené na x86 architektúre s 32bitovým procesorom Intel Quark SoC X1000 oproti Arduinu, na ktorých je poväčšine osadený AVR, prípadne ARM procesor.




Obr. 2.1: Vývojová doska Intel Galileo Gen2

Hlavný rozdiel medzi Galileom a Arduinom Yún je ten, že Yún má dva procesory. Na jednom beží Linux a druhý od Atmelu sa stará o beh Arduino kódu, kým Galileo má len jeden procesor na ktorom beží Linux a Arduino kód. Linux má nespočetne mnoho výhod a univerzálne rozširuje možnosti využitia tejto dosky. Môžme sa naň pripájať cez SSH (Secure Shell), využívať Python, webový server, vyvíjať softvér, ktorý je lepšie prepojený s hardvérom a mnoho ďalšieho. Staršia verzia Galilea nemala na doske regulátor napätia, takže zdroj napätia musel mať presne 5 V. Intel Galileo Gen2 má vstavaný regulátor napätia, takže môže byť napájané vhodným zdrojom s napätím 7 – 15 VDC.[4] Na doske nájdeme IOREF piny, ktorými vieme prepínať operačné napätie Galilea medzi 5 V a 3.3 V, čo zaručuje kompatibilitu aj so shieldmi, ktoré pracujú na 3.3 V.

Doska v sebe integruje mini-PCI Express rozhranie, 10/100 megabitový Ethernet či porty USB 2.0. Vďaka novej platforme si môžu rôzni „výmyselníci“ postaviť prototyp vlastných zariadení od displejov s LED technológiou až po komplexné automatizované zariadenia všetkého druhu. Ide teda o akýsi odrazový mostík, ktorý má pomôcť zrealizovať množstvo zaujímavých projektov. Veľmi podobnou platformou v tomto smere je aj miniatúrny počítač Raspberry Pi. Ten ale využíva procesor založený na báze ARM.[3] Galileo Gen2 je v ČR dostupné u rôznych resellerov za približne 1500,- CZK Podrobnejšie špecifikácie sú uvedené v tabuľke 2.1.

Tab. 2.1: Špecifikácie vývojovej dosky Intel Galileo Gen2

| | |
|---|---|
| **Mikrokontrolér** | Soc Quark X1000 |
| **Operačné napätie** | 3.3 V / 5 V |
| **Vstupné napätie** | 7 – 15 V |
| **Digitálne I/O piny** | 14 (6 z nich podporuje 8/12-bitový PWM výstup) |
| **Analógové vstupné piny** | 6 |
| **Flash pamäť** | 512 kB |
| **RAM** | 256 MB DDR3 |
| **SRAM** | 512 kB |
| **Flash úložisko** | 8 MB |
| **EEPROM** | 8 kB |
| **Takt procesora** | 400 MHz |
| **Power over Ethernet** | Kompatibilné |
| **Dĺžka** | 124 mm |
| **Šírka** | 72 mm |

## 2.4 PocketLab

Pre zaujímavosť uvediem ešte jednu platformu komerčnejšieho typu – PocketLab. Je to bezdrôtové zariadenie, ktore obsahuje akcelerometer, gyroskop, teplotný a tlakový senzor a megnetometer. PocketLab sa líši od vyššie uvedených platforiem tým, že sa nedá programovať. Je to už hotový produkt pripravený na použitie – IoT platforma s data hostingom, mobilnou a cloudovou aplikáciou pre vizualizáciu a analýzu týchto dát. Samotné zariadenie funguje tak, že sa nepripája priamo do internetu, ale páruje sa so smartfónom či tabletom pomocou Bluetooth 4.0. Smartfón s príslušnou aplikáciou teda slúži ako hlavný uzol (brána) pre tento tento senzor. Keďže je to pomerne nová platforma, PocketLab je dostupný zatiaľ len z oficiálneho obchodu za približne 100 USD.

# 3 IOT PROTOKOLY A KOMUNIKÁCIA MEDZI ZARIADENIAMI

Zariadenia musia medzi sebou vedieť komunikovať. Dáta zo zariadení sa potom musia zozbierať, poslať na servery a serverová infraštruktúra si tieto dáta musí vedieť vymieňať a poskytovať ich späť zariadeniam, programom na analýzu, alebo ľuďom. V IoT sa využíva viacero špecifických protokolov a spôsobov bezdrôtovej komunikácie.

## 3.1 Komunikácia

### 3.1.1 Bluetooth

Bluetooth je významnou technológiou pre komunikáciu na malú vzdialenosť, ktorá je už na trhu veľmi dlho. Nový Bluetooth Low-Energy (BLE) alebo ináč nazývaný aj Bluetooth Smart sa v IoT sa nepoužíva na prenos veľkých obnosov dát, ale skôr ho používajú menšie zariadenia, ako napríklad tzv. „wearables“ ktoré sú umiestnené na oblečení. Tie sa zväčša párujú pomocou Bluetooth so smartfónom. Takéto zariadenia si vymieňajú relatívne málo dát a Bluetooth im pre tento účel celkom vyhovuje, keďže v dnešnej dobe ho vieme nájsť už skoro na každom smartfóne či inom inteligentnom zariadení. Taktiež má nízku spotrebu, čo je len ďalšou výhodou.

- Štandard: Bluetooth 4.2 core
- Frekvencia: 2.4 GHz
- Dosah: 50 – 150 m (Smart/BLE)
- Rýchosť: 1 Mbps (Smart/BLE)

### 3.1.2 ZigBee

ZigBee je podobný Bluetooth, no viac sa využíva v priemyselnej sfére. Je založený na protokole IEEE802.15.4, čo je bezdrôtová technológia priemyselného šdandardu pracujúca na frekvencii 2.4 GHz. Zigbee má viacero dôležitých výhod v komplexných systémoch. Poskytuje prevádzku s nízkou spotrebou, bezpečnosť a škálovateľnosť s veľkým počtom uzlov.

- Štandard: ZigBee 3.0 založený na IEEE802.15.4
- Frekvencia: 2.4 Ghz
- Dosah: 10 – 100 m
- Rýchosť: 250 kbps

### 3.1.3 WiFi

WiFi pozná asi každý, je to bezdrôtová technológia rozšírená všade vo svete, poskytuje vysokú rýchlosť a schopnosť preniesť veľké objemy dát. V súčasnosti sa najviac používa štandard 802.11n, ktorý poskytuje veľkú šírku pásma, no pre použitie v niektorých IoT projektoch má trochu väčšiu spotrebu ako je požadované.

- Štandard: 802.11b/g/n/ac
- Frekvencia: 2.4 Ghz a 5 GHz pásma
- Dosah: cca 50 m
- Rýchlosť: max. 600 Mbps, typicky 150 – 200 Mbps (záleží od kanálu, použitého štandardu a frekvencie, najnovší štandard 802.11ac zvláda rýchlosti až do 1 Gbps)

### 3.1.4 NFC

NFC je skratka pre Near Field Communication. Ako už z názvu vyplýva, jedná sa o technológiu s malým dosahom. Používa sa vo smartfónoch či kartách na realizáciu bezkontaktných platieb, autorizáciu užívateľov pri elektronických zámkoch, či zdieľanie informácií medzi smartfónmi.

- Štandard: ISO/IEC 18000-3
- Frekvencia: 13.56 MHz
- Dosah: 10 cm
- Rýchlosť: 100-420 kbps

## 3.2 IoT protokoly

Tak ako v normálnom internete, aj v IoT potrebujeme rôzne protokoly na rôzne typy komunikácie. Všeobecne takéto protokoly rozdeľujeme na D2D (Device to Device), D2S (Device to Server) a S2S (Server to Server) protokoly.[11]
Najznámejšie z nich sú:

- MQTT (D2S protokol na zber dát a ich posielanie na server)
- XMPP (špeciálny prípad D2S protokolu, najlepší na komunikáciu s človekom a zariadením)
- DDS (D2D, rýchla zbernica na integráciu zariadení)
- AMQP (S2S, frontovací systém pre komunikáciu medzi servermi)

### 3.2.1 MQTT

MQTT je skratka pre Message Queue Telemetry Transport. Tento protokol sa zameriava na monitoring a ovládanie veľkého počtu zariadení vo veľkých sieťach. Jeho

úlohou je zozbierať dáta zo všetkých zariadení a poslať ich na server, kde sa ďalej spracujú. Funguje nad TCP protokolom, takže je spoľahlivý.[11]

### 3.2.2 XMPP

XMPP je skratka pre Extemsible Messaging and Presence Protocol. Tento protokol bol pôvodne pomenovaný Jabber – IM (instant messaging) služba. XMPP používa XML formát a tiež ako MQTT, funguje nad TCP protokolom. Má adresovaciu schému v tvare *meno@doména.com*, čo mu v IoT kontexte umožňuje adresovať priamo konkrétne zariadenie. Jednoduchý príklad použitia je domáci termostat pripojený na server, cez ktorý sa k nemu vieme dostať na diaľku zo smartfónu.[11]

### 3.2.3 DDS

Tento protokol slúži na prenos dát priamo medzi zariadeniami. Ponúka QoS (Quality of Service) a v podstate sa správa ako dátova zbernica medzi zariadeniami – kontroluje prístup jednotlivých zariadení k dátam a ich aktualizácie tak, aby každé zariadenie dostalo len tie, ktoré potrebuje. Je to realtime protokol s veľmi nízkou latenciou.[11]

### 3.2.4 AMQP

Je to transakčný protokol s vysokou spoľahlivosťou. Je prispôsobený na výmenu veľkého počtu transakčných správ medzi servermi. Má pôvod v bankovníctve, no má využitie aj v IoT. Funguje nad TCP protokolom, softvér ktorý ho využíva sa zameriava sledovanie transakcií a uistenie, že sa doručia. V IoT má využitie pri analýze dát na serveroch.[11]

# 4 SENZORY A ICH MOŽNOSTI VYUŽITIA

Pomocou Galilea a iných dosiek vieme snímať rôzne veličiny zo senzorov, ktoré pripojíme na jednotlivé vstupné piny. Galileo má 14 digitálnych 6 analógových vstupných pinov. Digitálne piny vedia snímať TTL úroveň signálu – HIGH a LOW, resp. 1 a 0. Na analógových pinoch je AD prevodník s 12bitovým rozlíšením (rozsah hodnôt 0 - 4095), čo nám umožňuje snímať napätie s presnosťou približne 1,22 mV/bit.

Výstup zo senzorov býva práve buď analógový alebo digitálny, záleží od typu senzora. Napríklad pri teplote je napätie na výstupe senzora úmerné teplote. Ak na vstup pripojíme tlačidlo, jeho hodnota bude digitálna – 1 alebo 0 – stlačené alebo pustené. Existuje veľmi veľa typov senzorov, ktoré vedia merať rôzne veličiny ako napr. teplota, vlhkosť, tlak ... Ako som už spomínal, senzor môže mať analógový či digitálny výstup. Ak si kúpime senzory jednotlivo, je dosť pravdepodobné, že budú mať analógový výstup. Tento výstup pripojíme na jeden zo vstupov vývojovej dosky a v programe ho ošetríme. Dajú sa však kúpiť aj senzorové dosky, ktoré sú schopné komunikovať digitálne pomocou I2C alebo SPI zberníc čo uľahčuje prácu so senzorom – získame priamo hodnoty bez toho aby sme ich museli prepočítavať z výstupného napätia. Tieto zbernice slúžia na komunikáciu medzi mikrokontrolérmi, senzormi, posuvnými registrami a pod.

## 4.1 Teplota, tlak a vlhkosť

Medzi základné typy senzorov patria senzory teploty, tlaku a vlhkosti. Používajú sa hlavne na získanie týchto vlastností v exteriéroch a interiéroch. Majú využitie v inteligentnej domácnosti alebo meteorologických staniciach. Bežné teplotné senzory dokážu merať teplotu v rozmedzí od -50 do 150 °C. Tlakové senzory sa dajú naviac využiť aj pri meraní nadmorskej výšky, pretože tlak sa s výškou mení. U senzorov vlhkosti sa dá zistiť vlhkosť prostredia, no existujú aj také, ktoré sa zameriavajú na detekciu kontaktu s vodou.

## 4.2 Orientácia, akcelerácia

Ďalšími užitočnými senzormi sú akcelerometry a gyroskopy, ktoré umožňujú zistiť presnú orientáciu alebo zrýchlenie. Dnes tieto senzory obsahuje väčšina smartfónov, čo im umožňuje zlepšovať UX (tzv. User Experience) napríklad zmenou orientácie užívateľského rozhrania. Dajú sa tiež využiť na detekciu pádu zariadenia.

## 4.3 Vzdialenosť, detekcia pohybu

Na určenie vzdialenosti sa dajú použiť ultrazvukové senzory, ktoré pracujú na princípe odrazu zvuku od snímaneho objektu. Ďalej existujú senzory na detekciu pohybu, ktoré vieme využiť v miestnosti na zapínanie svetiel, spustenie alarmu alebo napríklad aj analýzu „premávky“ na chodbách či v miestnostiach.

# 5 IMPLEMENTÁCIA SOFTWARE

Ako už vieme, Galileo beží na Linuxe. V základe obsahuje okresané linuxové jadro, ktoré je schopné spúšťať kód z Arduina, no po odpojení napájania sa kód stratí a je ho potrebné znova nahrať. Týmto spôsobom sa dá Galileo využívať ako klasické Arduino, no bola by škoda nevyužiť jeho plný potenciál. Preto máme na doske k dispozícii micro SD slot, do ktorého vieme vložiť SD kartu s bootovateľným linuxovým obrazom. Intel má k dispozícii špeciálnu linuxovú distrubúciu prispôsobenú práve pre Galileo, ktorá rozširuje možosti využitia – pripojenie po sieti cez SSH, spúšťanie rôznych Bash a Python skriptov, serverové služby, C/C++, dokonca môžme čítať a zapisovať na I/O piny, tzv. GPIO (General-purpose input/output). V podstate sa jedná plnohodnotný Linux. V tejto práci používame distribúciu s názvom *Linux 3.8.7-yocto-standard i586*, ktorá je súčasťou vývojového kitu od Intelu pre využitie v IoT. Čo sa týka balíkov, táto distibúcia je trochu okresaná oproti desktopovým distribúciam (predsa len, jedná sa o vývojovú dosku), no sú tam predinštalované užitočné balíky, ako je webový server *lighttpd* alebo *git* na správu repozitárov. Táto distribúcia používa na správu balíkov *opkg*, čo je odľahčený správca balíkov pre Linux. Pre porovnanie, Debian distribúcie používajú *apt*, Fedora a RedHat majú *yum*.[10] *Opkg* však nemusí niekomu stačiť, pretože v jeho repozitároch pre Intel Galileo sa nenachádzajú všetky známe balíky (napr. populárny textový editor *vim*). Práve preto si programy vieme skompilovať zo zdrojových kódov voľne dostupných na internete pomocou *gcc* kompilátora, ktorý býva snáď na každej linuxovej distribúcii. Celý software ako celok sa dá deliť na dve časti – backend a frontend. Frontend symbolizuje rozhranie medzi človekom a programom – grafické rozhranie, spôsob interakcie. Backend vyjadruje vnútornú logiku programu – to čo sa deje na pozadí, ako sa spracuvávajú dáta, komunikáciu medzi zariadeniami a pod.

## 5.1 Backend

Nakoľko máme k dispozícii Linux, nie sme obmedzení iba na jeden programovací jazyk ako pri Arduine. Môžeme použiť C/C++, Python, napísať si Bash skript... Možností je veľa. Treba však prihliadať na to, čo chceme realizovať a tomu sa prispôsobiť. Bash skriptmi vieme vytvoriť napríklad podporné skripty na správu logov – ich preposielanie na iný server a následné premazávanie. Dajú sa nakonfigurovať tak, aby sa spúšťali v určitom časovom intervale pomocou cron-u. Python ďalej ponúka množstvo knižníc, výborné možnosti pre prácu s textovými reťazcami, je ľahký na naučenie. Výhodou je, že preňho existuje užitočná knižnica na komunikáciu s webovou platformou umožňujúcou výmenu informácií medzi IoT zariadeniami pomerne

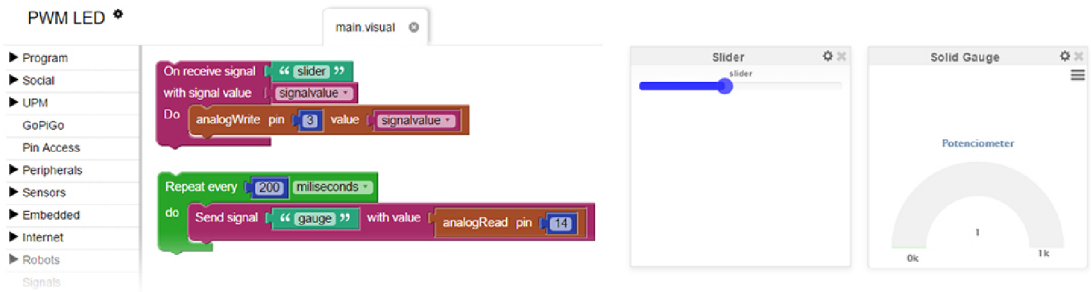
ľahkým a zaujímavým spôsobom. Popíšem ju v ďalších kapitolách.

### 5.1.1 Wyliodrin

Ďalej tu máme aj webové platformy slúžiace práve na programovanie vývojových dosiek na diaľku a následnú vizualizáciu dát. Jednou z nich je aj Wyliodrin. Wyliodrin má podporu pre Intel Galileo, Raspberry Pi, Arduino a mnoho ďalších dosiek. V princípe funguje tak, že si na stránke vytvoríme účet a zaregistrujeme naň vývojovú dosku. Ak sa jedná o dosku a ARM procesorom, Wyliodrin nám ponúkne na stiahnutie linuxový image, ktorý nahrajeme na SD kartu pomocou špeciálnej utility. Ďalej stránka užívateľovi vygeneruje unikátny JSON súbor. Ten sa uloží na SD kartu, Linux z neho pri bootovaní prečíta informácie a spustí wyliodrin službu, ktorá komunikuje s Wyliodrin serverom. Takto sa doska a server prepoja, čím nám vznikne možnosť programovať zariadenie na ďiaľku, pristupovať k shellu, monitorovať dáta a ovládať vstupné premenné pomocou rôznych ovládacích prvkov v ovládacom paneli, ako sú virtuálne posuvné potenciometre alebo tlačidlá. Špeciálnou vlastnosťou Wyliodrinu je, že podporuje vizuálne programovanie, ktoré spočíva v spájaní rôznych funkčných blokov ovládajúcich program a dosku. Od ovládania pinov až po vytváranie webových požiadaviek, či zobrazovanie informácií na LCD displeji. Wyliodrin nepodporuje programovanie 8 bitových AVR architektúr (staršie Arduino modely), pri ich použití si musíme napísať program klasickým spôsobom v Arduino IDE popísanom v ďalšej sekcii.

Ako ukážku tejto platformy uvediem program na obrázku 5.1. Tento program pri každej zmene virtuálneho potenciometra nastaví na 3. pine PWM výstup na hodnotu tohto potenciometra, čím sa zmení intenzita svetla LED diódy pripojenej na tento pin. Ďalej program každých 200 ms číta analógovú hodnotu na 14. pine (na doske označený ako A0) a nastaví ju na virtuálnom indikátore.

Jedinou nevýhodou, ktorú má táto platforma je, že počet virtuálnych prvkov na




Obr. 5.1: Ukážkový program vo vizuálnom programovaní

ovládanie dosky a zobrazovanie dát je obmedzený pre účet zdarma. Tiež je nedostatočne zdokumentovaná, čo sa týka virtuálnych prvkov v ovládacom paneli a ostatných jazykov. Pri programovaní v Pythone alebo C prakticky nevieme tieto prvky ovládať, pretože k tomu neexistuje dokumentácia.

### 5.1.2 Arduino IDE

Klasický spôsob ako naprogramovať Galileo je cez Arduino IDE. Programy sa píšu v C/C++. Výhodou je, že týmto spôsobom môžme priamo používať Arudino knižnicu, čo nám umožní prístup k pinom – ich čítaniu a zápisu. Existuje taktiež veľa knižníc na ovládanie Ethernet rozhrania, alebo rôznych zariadení ako sú servo motory, LCD displeje a podobne. Na Galileu funguje prakticky každá knižnica pre Arduino. Štruktúra klasického Arduino programu sa skladá z dvoch funkcií. Prvá funkcia *setup()* sa spustí iba raz na začiatku programu. Zvyčajne sa používa na nastavenie pinov, inicializáciu knižníc a podobne. Ďalej nasleduje funkcia *loop()* ktorá beží až do prerušenia programu, čím môžme chápať odpojenie Galilea od napájania, alebo stlačenie jedného z reset/reboot tlačidiel. Dôležité je dodať, že ak chceme Galileo programovať cez toto IDE, musíme Galileo pripojiť k počítaču USB káblom.

### 5.1.3 Komunikácia v sieti

S webovými platformami komunikujeme pomocou tzv. API (Application Program Interface). API je skupina rutín, protokolov a nástrojov slúžiacich na vytváranie aplikácii.[16]. My budeme využívať rôzne webové API, cez ktoré budeme komunikovať so servermi pomocou HTTP požiadaviek (tzv. HTTP requestov). Existuje viacero typov týchto požiadaviek. Na odoslanie hodnoty na server sa zvyčajne používa typ POST s dátami a na získanie informácií zo servera typ GET s argumentmi.

Spôsob komunikácie záleží na type zariadenia, z ktorého budeme posielať dáta do internetu. V našom prípade – Intel Galileo – je to celkom jednoduché vzhľadom na to, že na ňom beží Linux a máme možnosť spúšťať linuxové príkazy priamo z Arduino kódu pomocou funkcie `system()`, ktorá si ako jediný parameter berie reťazec znakov reprezentujúci daný príkaz. Na Linuxovej úrovni sa nám naskytá viacero možností, medzi ktorými je aj Python skript, ktorý nám dané dáta odošle na cloud, alebo použijeme ešte lepšie riešenie - príkaz `curl`, ktorý sa nachádza snáď vo všetkých distribúciach Linuxu. Jedná sa o internetový nástroj slúžiaci na posielanie/príjimanie súborov zo servera [15], ktorý vie komunikovať pomocou rôznych protokolov, medzi ktorými je aj HTTP/HTTPS. Toto riešenie je výhodnejšie najmä z hľadiska rýchlosti. Kým spustenie Python skriptu na Galileu trvalo aj dve sekundy, príkaz `curl` sa vykonal takmer okamžite.

Toto je praktická ukážka, ako poslať HTTP GET požiadavku na ThingSpeak server pomocou príkazu `curl`. Zisťujeme, či na zariadenie nečaká príkaz na vykonanie, ak áno, server nám vráti textový reťazec s príkazom:

```
curl --request GET 'https://api.thingspeak.com/talkbacks/8292/
    commands/execute?api_key=GMY5EUTQZF1KTX70'
```

Takto zas odošleme hodnotu zo senzora na ThingSpeak server pomocou POST požiadavky:

```
curl --data "api_key=S14ZOZ0CB8OF927Q&field1=73" https://api.
    thingspeak.com/update.json
```

Presný formát požiadaviek (aké argumenty a dáta posielať) špecifikuje práve API, ktoré je rozdielne pre každú platformu. Každá platforma by mala mať svoje API zdokumentované na svojich stránkach.

Ak je potreba komunikovať so serverom pomocou Arduina, budeme musieť použiť Ethernet shield. Staršie Arduino modely nemali Ethernet rozhranie, a preto bolo potrebné použiť takéto riešenie. Pri návrhu novších modelov Arduina, no aj iných vývojových platforiem, už však výrobcovia čoraz častejšie berú na vedomie rýchly nárast IoT a preto umiestňujú sieťové rozhrania už priamo na svoje dosky. Po inicializácii Ethernet shieldu na Arduine môžme použiť takýto kód na odoslanie HTTP požiadavky:

```
EthernetClient client;

.
.
.

if (client.connect(server, 80)) {
  Serial.println("pripojene na server");
  //Vytvorime HTTP request
  client.println("GET /talkbacks/8292/commands/execute?api_key=
   GMY5EUTQZF1KTX70 HTTP/1.1");
  client.println("Connection: close");
  client.println();
} else {
  Serial.println("pripojenie zlyhalo");
}
```

## 5.2 Frontend

V IoT svete sú práve webové platformy hlavným užíteľským rozhraním. Slúžia na vizualizáciu dát zo senzorov, vzdialené ovládanie spotrebičov a celkovú správu zariadení. Nemenej populárnou náhradou za webové platformy sú mobilné aplikácie do smartfónov – tento spôsob interakcie so zariadeniami je čoraz polulárnejší.

Ako som už spomínal, Wyliodrin nám nevyhovoval z hľadiska obmedzeného počtu ovládacích prvkov. Preto sme siahli po zaujímavejších riešeniach, resp. kombinácii rôznych webových platforiem. Prvé z nich je kombinácia Dweet.io a Freeboard.io, ďalším je GO+ Beta a posledné sa volá ThingSpeak.

### 5.2.1 Dweet.io

Webová platforma Dweet.io sa zameriava práve na Internet of Things a umožňuje posielanie správ a ich následný odber z jej servera pomocou HTTP GET požiadaviek. Je to dobrý príklad D2D komunikácie, kde zariadenie, alebo „vec“, keďže sa bavíme o IoT, vie poslať správu na server a ostatné zariadenia si túto správu vedia prečítať. Spôsob ako pošleme správu do internetu je jednoduchý a efektívny, keďže o spracovanie dát a uloženie sa už sám postará server. Stačí jednoducho takto zavolať URL:

```
https://dweet.io/dweet/for/mojaVec?idSenzora=0&teplota=25
```

Takto pošleme tzv. „dweet“ – správu zo zariadenia, s dvoma premennými *id* a *teplota*. Dweet.io si pamätá posledných 500 správ za posledných 24 hodín pre jedno zariadenie, takže ak nepošleme v dlhšej dobe update, dáta sa zo servera zmažú. Dáta sú na serveri voľne prístupné, stačí len vedieť meno zariadenia. Dá sa predplatiť možnosť uzamknutia zariadení tak aby sme k ním vedeli pristupovať iba pomocou kľúča, no to pre náš projekt nebudeme potrebovať. Ak by sme ale potrebovali aspoň nejakú bezpečnosť a chceli trochu utajiť zariadenie, navrhoval by som použiť ako jeho meno náhodne generovaný hash, ktorý nie je už také ľahké uhádnuť. Dweet.io funguje tak, ako aj zvyšok zmienených platforiem, cez HTTP aj HTTPS.[12] Poslednú správu od daného zariadenia prečítame tak, že zavoláme URL:

```
https://dweet.io/get/dweets/for/mojaVec
```

Server vráti odpoveď v JSON formáte, ktorá vyzerá takto:

```
{
  "this":"succeeded",
  "by":"getting",
  "the":"dweets",
  "with":
```

```
  [
    {
      "thing":"mojaVec",
      "created":"2015-12-12T16:20:57.376Z",
      "content":{"idSenzora":0,"teplota":25}
    }
  ]
}
```

Obsahuje návratové hodnoty zo servera o úspešnosti požiadavky, názov zariadenia, čas vytvorenia správy a jednotlivé premenné, ktoré sme odoslali zo zariadenia. Takto však vieme prečítať iba poslednú správu od zariadenia. Dweet.io však vie poskytnúť aj všetky zapamätané správy zo zariadenia:

```
https://dweet.io/get/dweets/for/mojaVec
```

alebo dokonca real-time stream zo zariadenia pomocou tzv. „rozkúskovanej“ HTTP požiadavky. Ten udržiava otvorené spojenie so serverom, ktorý posiela klientovi živé aktualizácie. Nefunguje to na bežných webových prehliadačoch, musíme použiť linuxový klient *curl* s parametrom *-i*, príklad:

```
curl -i https://dweet.io/listen/for/dweets/from/mojaVec
```

Dweet API je dobre zdokumentované na webe www.dweet.io/play. Veľkou výhodou tejto platformy je, že sú na ňu vyvinuté knižnice pre rôzne skriptovacie jazyky, my využijeme **dweepy**, knižnicu pre Python. Teraz, keď máme vyriešenú komunikáciu s internetom, musíme dáta vizualizovať. Na to použijeme Freeboard.io.

### 5.2.2 Freeboard.io

Freeboard je webová platforma na vizualizáciu dát z IoT.[13] Pochádza od tvorcov platformy Dweet.io, takže sa dá predpokladať že sú úzko prepojené, čo je aj pravda. Freeboard dokáže vizualizovať dáta z Dweet-u, stačí mu len názov zariadenia. S prijatými dátami vieme ešte pred samotnou vizualizáciou pracovať v JavaScript-e, je to užitočné napríklad ak potrebujeme prepočítať prijatú hodnotu zo senzora na teplotu a pod. Samotné dáta sa vizualizujú pomocou tzv. „widgetov“, podobajú sa na ovládacie prvky z Wyliodrinu. Môžme zobraziť textové hodnoty, rôzne indikátory v podobe stavových LED diód alebo tachometra, Freeboard vie tiež vykresľovať grafy hodnôt meniacich sa v čase, či HTML kód prijatý od zariadenia. Nevýhodou však je, že zariadenia nevieme na diaľku ovládať.

### 5.2.3 GO+ Beta

GO+ Beta je ďalšia z webových IoT platforiem. Umožňuje vizualizovať dáta v rôznych widgetoch ako napr. grafy, či prepínače. Naše dáta môžu byť verejné, alebo privátne a môžu mať rôzne podoby, napr. číselnú hodnotu, reťazec znakov alebo geo lokáciu. GO+ funguje na princípe zariadení. Každé zariadenie má priradené tzv. „DID“ (Device ID), ktorým sa identifikuje v API volaniach a môže niesť iba jeden typ hodnoty. To znamená, že ak máme senzor, ktorý meria teplotu, tlak a vlhkosť, musíme preňho vytvoriť 3 zariadenia na GO+, pre každú meranú veličinu. API volania sa tu vytvárajú iba pomocou GET požiadaviek. Napríklad, na zapísanie hodnoty do zariadenia použijeme takúto URL

```
http://168.63.82.20/server/income?did=mojeDID&action=put&value=65
```

Sú tu 3 povinné argumenty: `did` označuje ID zariadenia, ktoré nám bude vygenerované pri vytváraní zariadenia (alebo si ho zvolíme sami), `action` označuje typ akcie, ktorú vykonávame, v tomto prípade je to *put*, na zapísanie hodnoty. Existujú ešte možnosti ako *ack* na získanie hodnoty, alebo *switch* na ovládanie prepínačov. Posledný argument `value` značí zapisovanú hodnotu, v tomto prípade je to 65. IP adresa platformy, tak isto ako ostatné príklady API volaní sa dá nájsť na pomocnej stránke po prihlásení na beta.goplusplatform.com/help. Jedinou nevýhodou tejto platformy je, že ešte nie je plne funkčná (je to beta verzia). Nefungujú správne niektoré API volania. Napríklad zisťovanie hodnoty zariadenia pomocou *ack* hodnoty parametra `action` sa správa úplne rovnako ako zapisovanie hodnoty, ďalej, nemôžme ovládať zariadenia na diaľku. V nastaveniach zariadenia sa nachádza pole pre URL výstupných dát *Output data URL*, na ktorú by sa mali posielať požiadavky zo servera po zmene hodnoty. Napríklad, ak používame widget, ktorý má dve tlačídlá na ovládanie, po stlačení tlačidla by sa mala odoslať požiadavka na danú URL, no nestane sa tak.

### 5.2.4 ThingSpeak

ThingSpeak je akási kombinácia prechádzajúcich platforiem. Tiež poskytuje hosting pre IoT zariadenia, dáta sa na server odosielaju rovnako ako v predchádzajúcich prípadoch, len cez iné API - ThingSpeak Channels and Charts API. Dáta zo zariadení sú verejne prístupné iba ak to povolíme v nastaveniach. Táto platforma funguje na princípe kánálov. Dáta sú zapisované do kanálov, každý kanál môže obsahovať max. 8 polí pre dáta hociakého typu (čísla, reťazce znakov a pod.). Do kanálu môžme pridávať rôzne vizualizácie týchto dát (Grafy, Matlab vizualizácie, dokonca si môžme napísať vlastné widgety cez HTML, CSS a JavaScript). Najväčšou výhodou tejto platformy je možnosť ovládať zariadenia na diaľku cez tzv. TalkBack.

TalkBack je špeciálna aplikácia bežiaca na ThingSpeak-u, s ktorou môžeme komunikovať pomocou API - zasielať jej rôzne príkazy, ktoré sa postupne pridávajú na jej zásobník. Zásobník môže obsahovať max. 255 záznamov. Princíp spočíva v tom, že HTTP požiadavkou pošleme na TalkBack reťazec znakov reprezentujúci príkaz pre zariadenie. Príkaz sa uloží do zásobníka na posledné miesto. Zariadenie potom necháme periodicky sa dotazovať cez API na príkazy uložené v TalkBack-u. TalkBack nám vždy vráti príkaz na poslednom mieste v zásobníku (najnovší) a následne ho odtiaľ vymaže. Zariadenie si prečíta príkaz a podľa neho vykoná operáciu. Ak TalkBack neobsahuje žiadne príkazy, server vracia prázdny reťazec znakov.

Tu je pár ukážok API volaní pre túto platformu, napr. ako dostať posledných 5 hodnôt z prvého poľa kanálu s ID 113662

```
curl --request GET 'https://api.thingspeak.com/channels/113662/
    fields/1.json?results=5'
```

Ako poslať hodnotu 73 do prvého poľa kanálu

```
curl --data "api_key=S14ZOZOCB8OF927Q&field1=73" https://api.
    thingspeak.com/update.json
```

Ako dostať všetky príkazy čakajúce na vykonanie v TalkBack-u s ID 8292

```
curl --request GET 'https://api.thingspeak.com/talkbacks/8292/
    commands.json?api_key=GMY5EUTQZF1KTX70'
```

Treba si však dávať pozor, v akých intervaloch a akým spôsobom zapisujeme dáta do kanálu. ThingSpeak má limit 15 sekúnd na jednu aktualizáciu kanálu. To znamená, že môžme posielať dáta iba každých 15 sekúnd. Ináč server odpovedá chybovým kódom (0). Ak potrebujeme zapísať do kanálu viac polí, musíme tak urobiť naraz. V POST požiadavke jednoducho špecifikujeme dáta pre viac kanálov naraz. Príklad:

```
curl --data "api_key=S14ZOZOCB8OF927Q&field1=73&field2=45&field5=
    Sprava&field7=3.14" https://api.thingspeak.com/update.json
```

Všimnime si pole `api_key` v požiadavkách. Špecifikuje API kľúč, ktorý je generovaný pre každý kanál, alebo aplikáciu na ThingSpeak-u. Pomocou neho môžme zapisovať do kanálov, meniť alebo zisťovať citlivé dáta na serveri, ku ktorým by nemala mať verejnosť prístup, a podobne. Je to v podstate forma autentifikácie medzi serverom a klientom. API kľúč by sme nemali zverejnovať. Za zmienku tiež stojí, že server nám vracia dáta v JSON formáte, tak ako Dweet.io. Pri takýchto webových API je to štandardná forma odpovede.

## 5.3 Testovacia implemenácia pre Intel Galileo

Na ukážku tejto komunikácie a vizualizácie dát som napísal pre Galileo skript v Pythone (aj napriek tomu že je to nevýhodnejšie, ukážeme si aj tento spôsob), ktorý prečíta zo vstavaného tepelného senzora teplotu procesora a pošle ju na Dweet každých 5 sekúnd. Potom na Freeboarde vykreslím graf tejto teploty. Tiež som na ukážku ostatných widgetov zobrazil dáta z môjho smartfónu. Na Galileu sa dá prečítať hodnota z teplotného senzora zo súboru `/sys/class/thermal/thermal_zone0/temp`. Hodnota zo senzora je v jednotkách °C a je násobená 1000. Kód vyzerá nasledovne:

```
import threading
import dweepy

def sendData():
  threading.Timer(5, sendData).start()
  fo = open("/sys/class/thermal/thermal_zone0/temp", "r")
  str = fo.read()
  temp = int(str)/1000
  fo.close()
  dweepy.dweet_for('xseman03_galileo', {'teplota_CPU': temp})

sendData();
```

Najprv importujeme potrebné knižnice, medzi ktorými je aj **dweepy**, potom vytvoríme funkciu, ktorá bude každých 5 sekúnd čítať teplotu zo senzora a posielať ich na Dweet. Potom na Freeboarde pridáme nový dátový zdroj s menom Galilea (v našom prípade `xseman03_galileo`) a pridáme widget, ktorý bude vykresľovať graf premennej `teplota_CPU`. Výslednú vizualizáciu môžme vidieť na Obr.5.2


Obr. 5.2: Vizualizácia dát cez Freeboard

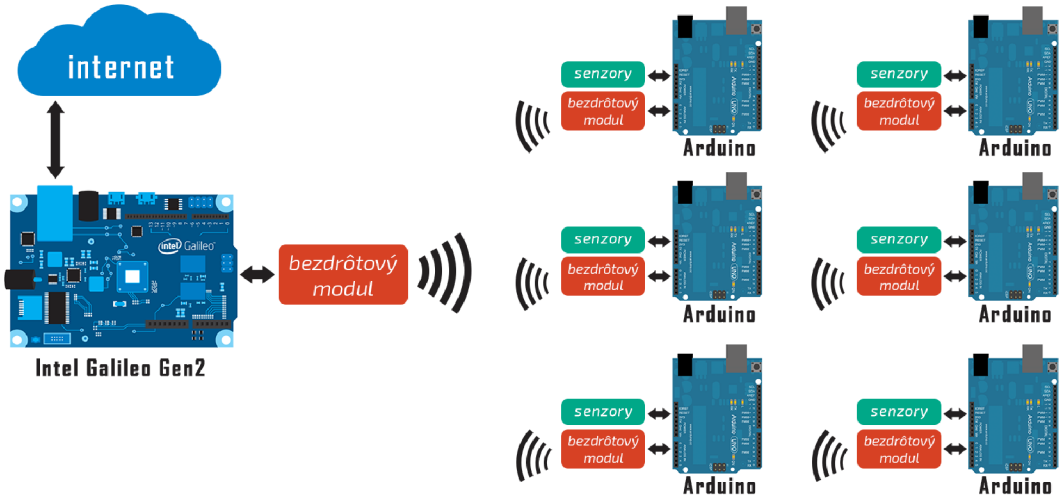
# 6 TESTOVACIA SIEŤ

## 6.1 Návrh

Základom tejto siete bude doska Intel Galileo Gen2, ktorá bude pripojená do internetu. K dispozícii budeme mať 4 vývojové dosky Arduino Mega, ku ktorým pripojíme rôzne senzory. Dosky budú s Galileom komunikovať pomocou bezdrôtových XBee modulov (ZigBee). Jednotlivé senzorické uzly (Arduino dosky) budú pripojené ku Galileu vo hviezdicovej topológii. Celá komunikácia s internetom bude postavená na už spomínanej webovej platforme ThingSpeak, keďže sa osvedčila asi najviac zo všetkých spomínaných.

Zo senzorov použijeme teplotné, tlakové a svetelné senzory. Na demonštráciu ovládania z internetu budeme ovládať LED diódy na Arduino doskách v troch režimoch – zapnutá, vypnutá, alebo budú automaticky reagovať na intenzitu osvetlenia a tak simulovať automatické rozsvecovanie svetiel pri zníženej intenzite svetla.

Celá sieť bude demonštrovať využitie v inteligentných mestách. Na každom uzle bude LED dióda reprezentujúca verejné osvetlenie a svetelný senzor, podľa ktorého sa bude toto osvetlenie regulovať. Návrh siete je na Obr. 6.1. Počet jednotlivých uzlov je orientačný.




Obr. 6.1: Návrh testovacej siete

## 6.2 Realizácia siete

### 6.2.1 Topológia siete

Sieť sme zapojili do hviezdicovej topológie, v jej strede je doska Intel Galileo Gen2 pripojená do internetu. Je osadená bezdrôtovým XBee modulom 1. série. Tieto moduly majú tú výhodu, že pre nich existuje Arduino knižnica, tým pádom sme ju ľahko mohli použiť či už na Galileu, tak na Arduino doskách so senzormi, s ktorými Galileo komunikuje. Galileo figuruje v sieti ako centrálny bod, ktorý ovláda a komunikuje s Arduino doskami, ku ktorým sú pripojené rôzne senzory. Na účel komunikácie sme vymysleli ľahký protokol, ktorý beží nad XBee vrstvou. XBee umožňuje v jednej správe poslať 100 bajtov dát, práve tento priestor sme využili na protokol a dáta.

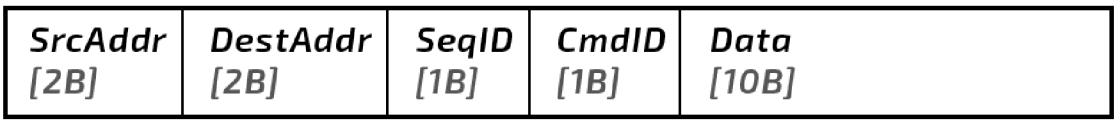
### 6.2.2 Komunikačný protokol

Aj keď máme dostupných až 100 bajtov na jednu správu, efektívne ich využívame len 16. Režijné dáta zaberajú 6 B a na informácie sme vyhradili 10 B. Protokol vyzerá nasledovne:

- 2 B - Zdrojová adresa
- 2 B - Cieľová adresa
- 1 B - Číslo sekvencie
- 1 B - Číslo príkazu
- 10 B - Dáta

Zdrojová a cieľová adresa označujú adresy XBee modulov, každá XBee adresa je 16-bitové číslo, preto sme použili takto veľkú premennú. Ďalej nasleduje číslo sekvencie – v prípade, že by sme chceli rozdeliť dáta na menšie časti. Číslo príkazu je dôležité – označuje, čo znamenajú dáta, ktoré nasledujú v dátovej časti paketu. Čísla príkazov:

- 1 - Hľadanie susedných uzlov
- 11 - Odpoveď na hľadanie suseda
- 5 - Dátový paket
- 6 - Požiadavka na dáta




Obr. 6.2: Štruktúra protokolu

### Hľadanie susedných uzlov

Hľadanie susedných senzorických uzlov funguje tak, že Galileo na začiatku vyšle tento typ paketu ako broadcast. V dátovej časti je jeho adresa. Broadcast sa vysiela na adresu 0xFFFF (255). Tento paket dostanú všetky zariadenia v XBee sieti. Následne každé zariadenie odošle späť Galileu už unicast s číslom príkazu 11 a svojou adresou. Takto si Galileo vytvorí tabuľku senzorov v sieti.

| SrcAddr | DestAddr | SeqID | CmdID | Data |
|---|---|---|---|---|
| 7 | 255 | 0 | 1 | 7 |

Obr. 6.3: Paket - hľadanie suseda

### Dátový paket

Dátový paket obsahuje v prvých dvoch bajtoch svojej dátovej časti dve čísla identifikujúce typ hodnoty, ktorá sa posiela (teplota, svetelná intenzita a pod.) a typ premennej (*int*, *float*, *string* a podobne). Potom nasledujú samotné dáta, pre ktoré zostáva 8 bajtov. Najväčšia premenná, akú dokážeme poslať je *long* s veľkosťou 32 bitov = 4 bajty. Zaujímavosťou je, že typ *double* má na Galileu veľkosť 64 bitov, no na Arduino paltforme je to 32 bitov. Preto sme od tohto dátoveho typu upustili. V konečnom dôsledku je to po konverzii na 32-bitový formát obyčajný float. Dátový paket sa posiela buď na požiadanie od zariadenia, ktoré pošle paket s príkazom č. 6 (požiadavka na dáta) alebo navyžiadane (napr. ak chceme ovládať LED diódu na Arduine). Tabuľka 6.1 obsahuje čísla premenných a typov hodnôt, ktoré využívame v komunikácii. Na obrázku 6.4 je znázornený paket, v ktorom posielame zo zaria-

Tab. 6.1: Tabuľka typov hodnôt a premenných

| **Hodnoty** | | **Premenné** | |
|---|---|---|---|
| **Teplota** | 1 | **byte** | 1 |
| **Tlak** | 2 | **char** | 2 |
| **Intenzita svetla** | 3 | **int** | 3 |
| **LED dióda** | 4 | **float** | 4 |
| | | **long** | 5 |

denia s adresou 10 hodnotu z teplotného senzora (typ *float*) do Galilea s adresou 7. Desatinné číslo bude samozrejme rozložené na ďalšie 4 bajty. Dátová časť tak zaberie celkovo 6 bajtov.

| SrcAddr | DestAddr | SeqID | CmdID | Data |
|---|---|---|---|---|
| 10 | 7 | 0 | 5 | 1, 4, 25.6 |

Obr. 6.4: Paket - posielanie hodnoty

#### Požiadavka na dáta

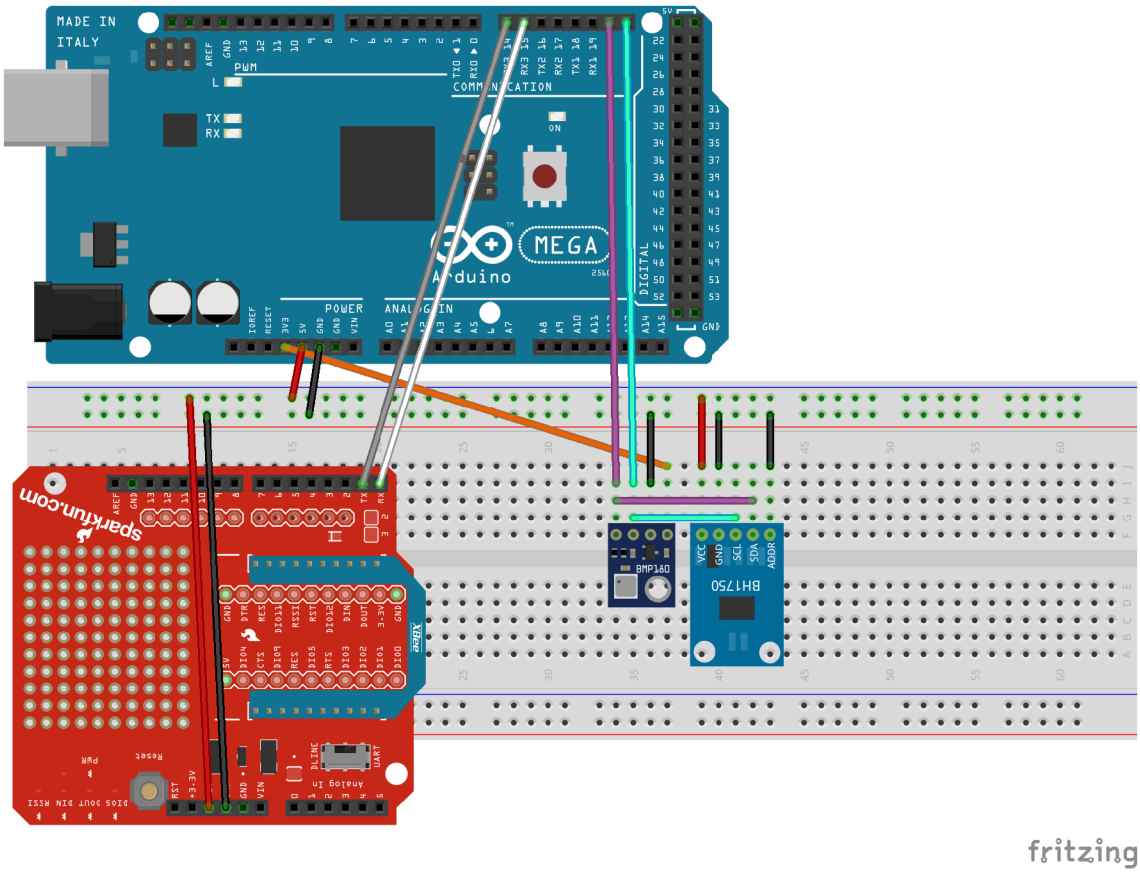
Požiadavka na dáta obsahuje len jedno číslo symbolizujúce dáta, ktoré si pýtame od zenzorického uzlu. Číslo korešponduje s tým, ktoré je použité v dátovom pakete, viď tabuľka 6.1. Na obrázku 6.5 je znázornený paket s požiadavkou na dáta zo svetelného senzora.

| SrcAddr | DestAddr | SeqID | CmdID | Data |
|---|---|---|---|---|
| 10 | 7 | 0 | 5 | 3 |

Obr. 6.5: Paket - požiadavka na dáta

### 6.2.3 Senzory a Bezdrôtové moduly

Ku každej Arduino doske sú pripojené 2 senzory. Prvý senzor sa volá BMP180 a dokáže merať teplotu a barometrický tlak. Druhý senzor sa volá BH1750 a meria svetelnú intenzitu. Senzory sú pripojené k Arduinu cez $I^2C$ zbernicu. $I^2C$ zbernica je založená na master-slave komunikácii. To znamená, že Arduino sa správa ako master a senzory ako slave. Arduino môže takto komunikovať s viacerými senzormi naraz. $I^2C$ zbernica vyžaduje na komunikáciu iba dve linky (SDA a SCL). SDA je dátová linka a SCL nesie synchronizačný hodinový signál. $I^2C$ protokol podporuje komunikáciu až medzi 1008 slave zariadeniami. Na komunikáciu so senzormi využívame voľne dostupné knižnice z internetu, ktoré nám uľahčujú inicializáciu senzorov a čítanie hodnôt zo senzorov po $I^2C$ zbernici. Teplotný senzor BMP180 je napájaný 3,3 V, svetelný senzor BH1750 je napájaný 5 V. Senzory pripojíme do zbernice spojením ich SDA a SCL pinov s korešpondujúcimi SDA a SCL pinmi na Arduine. Na obrázku 6.6 je vidieť zapojenie senzorov a XBee shieldu k Arduinu. XBee shield je pripojený k Arduinu po sériovej linke. Arduino Mega má 3 sériové rozhrania a XBee shield sme pripojili práve k tretiemu preto, aby sme mohli používať prvé rozhranie na ladiace výpisy do konzoly pri programovaní dosky.




Obr. 6.6: Zapojenie Arduina so senzormi a XBee shieldom

### 6.2.4 Algoritmus mikrokontrolérov

Vzhľadom na topológiu, používané platformy a ich úlohu v sieti, sme napísali dva algoritmy. Prvý je pre Intel Galileo a druhý pre senzorické uzly na Arduino platforme.

#### Intel Galileo

Galileo slúži ako centrálny bod v sieti, preto vyžaduje odlišný algoritmus ako ostatné zariadenia. Je v ňom uložená tabuľka susedov, API kľúče od ThingSpeak kanálov, stará sa o komunikáciu s internetom. Za zmienku stoja tieto dve dátové štruktúry: `xbee_packet` a `neighbor_list`:

```
struct xbee_packet
{
  uint16_t srcAddr;
  uint16_t dstAddr;
  uint8_t seq_ID;
  uint8_t command_ID;
  uint8_t dataapp[10];
};
```

```
struct neighbor_list
{
  String name;
  uint16_t addr;
  uint8_t rssi;
  String api_key;
  String channel_data[8];
  boolean isOnline = false;
  unsigned long lastOnline;
};
```

Štruktúra pre paket obsahuje podľa protokolu zdrojovú adresu, cieľovú adresu, číslo sekvencie, číslo príkazu a 10 bajtov dlhý zásobník na dáta. Veľkosti premenných sme použili podľa toho, akú veľkosť majú mať pri prenášaní. (`uint16_t` má veľkosť 16 bitov, `uint8_t` má 8 bitov). Tabuľka susedov má polia pre názov senzorického uzlu, jeho adresu, RSSI – intenzitu signálu, jeho API kľúč od kanálu, dáta kanálu, indentifikátor online stavu a čas, kedy bol uzol naposledy online (online – dokáže komunikovať s Galileom).

Na začiatku inicializujeme XBee knižnicu, Ethernet, časovače a odošleme broadcast na hľadanie susedov.

```
byte mac[] = {0x98, 0x4F, 0xEE, 0x01, 0xEE, 0x63}; // Galileo MAC
IPAddress server(54, 88, 155, 198); // Thingspeak API
EthernetClient client;
XBee xbee = XBee();
unsigned long timer1;
unsigned long timer2;

void setup() {
  Serial1.begin(9600);
  xbee.setSerial(Serial1);
  Ethernet.begin(mac);

  //Init timers
  timer1 = millis();
  timer2 = millis();

  discover_neighbors();
}
```

Potom nasleduje slučka *void* `loop()`, v ktorej prebieha celá logika programu. Na začiatku sa vykonáva prvý časovač. Ten každých 5 sekúnd odošle HTTP GET požiadavku na náš ThingSpeak TalkBack. Ak v ňom čakajú príkazy na vykonanie, server odpovie s najnovším príkazom. Zároveň v tomto časovači pošle broadcast na

hľadanie susedných senzorických uzlov – robíme tak periodicky každých 5 sekúnd preto, aby sme obnovovali ich online status. Taktiež každých 5 sekúnd posielame senzorickým uzlom požiadavky na dáta zo senzorov a stav LED. Robíme to len pre jednu z dosiek za jedno spustenie časovača. Dôvodom je to, že ak pošleme naraz požiadavky všetkým uzlom, tie v jednom momente odpovedia a XBee modul nestihne prečítať toľko odpovedí od rôznych uzlov zároveň - strácajú sa nám dáta a komunikácia začína byť nespoľahlivá. Ďalej nasleduje čítanie XBee paketov. Keď Galileo dostane XBee paket, nasleduje kontrola ID príkazu. Ak je to 11, program pracuje so svojou tabuľkou susedov - pridáva a aktualizuje stav susedných uzlov. Ak je to 5, znamená to, že nám prišli dáta a program ich nasledovne spracuje.

**Obsluha tabuľky susedov:** Program najprv prejde celú tabuľku susedov a zistí, či sa už daný uzol v tabuľke nenachádza (porovnáva adresy). Ak áno, od suseda zistí iba nové RSSI, nastaví mu stav online a tiež mu nastaví posledný čas, kedy bol online (funkcia `millis()` ktorá vracia čas od spustenia programu v milisekundách), ak sa v tabuľke ešte nenachádza, pridá ho do tabuľky a okrem vyššie spomínaných krokov mu ešte pridá meno, adresu a API kľúč, pod ktorým bude komunikovať s ThingSpeak kanálom.

**Spracovanie dát:** Program skontoluje, či dáta prišli od jedného z uzlov v tabuľke susedov, ak áno, prečíta od ktorého z nich to bolo, zistí aký typ hodnoty prišiel (teplota, tlak, svetlo, LED) a o aký typ premennej sa jedná. Následne podľa toho dáta spracuje. Typ premennej potrebujeme vedieť preto, aby sme správne prečítali „surové“ dáta z paketu - každá premenná má inú veľkosť, tým pádom je stále potreba prečítať iný počet bajtov. Následne zapíšeme tieto dáta susedovi do poľa s dátami pre ThingSpeak kanál na správnu pozíciu. Svetelná intenzita je na prvej pozícii, teplota je na druhej, tlak na tretej a stav LED na ôsmej. Tieto pozície korešpondujú s poradím kanálov na ThingSpeak-u.

Programová slučka nasleduje ďalej. Každých 30 sekúnd pre každý susedný uzol ktorý je online, Galileo pošle aktualizáciu na ThingSpeak kanál s jeho dátami zo senzorov. Ďalej sa kontroluje, či neprišla odpoveď na GET požiadavku z TalkBack-u. V odpovedi sa hľadá preddefinovaný reťazec - príkaz, ktorým budeme ovládať LED na senzorických uzloch. Príkazy majú nasledujúci tvar:

- `LED_ON_n` - zapnutie LED
- `LED_OFF_n` - vypnutie LED
- `LED_AUTO_n` - automatické ovládanie LED

Kde $n$ značí číslo uzlu. Po prijatí takéhoto príkazu Galileo odošle danému uzlu dáta so stavom LED, keď uzol tieto dáta príjime, nastaví LED na požadovanú hodnotu.

Ak sa jedná o príkaz na automatické ovládanie, uzol prečíta hodnotu zo svetelného senzora a ak zistí, že je v jeho okolí tma, zapne LED diódu. Naopak, ak je v jeho okolí svetlo, diódu vypne. Medznú hodnotu sme zvolili 300 LUX. Na konci programovej slučky sa kontroluje online stav senzorických uzlov. Ak už 10 sekúnd Galileo nedostalo aktualizáciu od senzora (porovnávame aktuálny čas a posledný čas online stavu), nastavíme uzol na stav offline. Takto predídeme zbytočnému pokusu odoslať dáta z uzlu na server.

### Arduino Mega

Na začiatku klasicky inicializujeme časovač a všetky knižnice pre XBee a senzory BMP180 a BH1750.

```
XBee xbee = XBee();
BH1750 lightmeter;
SFE_BMP180 pressureSensor;
unsigned long timer1;

void setup() {
  Serial.begin(9600);
  Serial3.begin(9600);
  xbee.setSerial(Serial3);

  timer1 = millis();
  lightmeter.begin();

  if (pressureSensor.begin()) {
    Serial.println("Pressure sensor init success");
  } else {
    Serial.println("Pressure init fail");
    while (1);
  }
}
```

Ďalej nasleduje slučka programu, ktorá len opakovane čaká na prichádzajúce dáta z XBee modulu, zapisuje hodnoty zo senzorov do globálnych premenných, ku ktorým neskôr pristupujeme pri odosielaní paketov a tiež každých 5 sekúnd mení stav osvetlenia podľa okolitého svetla v prípade, že je nastavené automatické ovládanie LED. Takto čítame dáta zo senzorov:

```
lux = lightmeter.readLightLevel();

char status;
double T, P;
```

```
status = pressureSensor.startTemperature();
if (status != 0) {
  // Wait for measurement
  delay(status);
  // Store temperature in T
  status = pressureSensor.getTemperature(T);
  if (status != 0) {
    temperature = T;
    // Start pressure reading, oversampling = 2, quality (1-3)
    status = pressureSensor.startPressure(2);
    if (status != 0) {
      delay(status);
      // Store absolute pressure in P
      status = pressureSensor.getPressure(P, T);
      if (status != 0) {
        pressure = P;
      }
    }
  }
}
```

Čítanie zo svetelného senzora je ľahké – je to príkaz na jeden riadok. Čítanie tlaku a teploty je však trochu komplikovanejšie. Je založené na tom, že senzor nám vždy vráti čas ktorý musíme počkať, kým sa odmeria daná hodnota. Najprv meriame teplotu do premennej `T`, čakáme kým sa dokončí meranie a túto teplotu potom zapíšeme do globálnej premennej `temperature`. Rovnakým spôsobom nasleduje aj meranie tlaku. Teplotu musíme odmerať ako prvú preto, že meranie tlaku je založené na teplote prostredia.

Nasleduje čítanie XBee paketov. Môžme dostať LED dáta (príkaz 5), alebo požiadavku na dáta zo senzorov (príkaz 6). Čo sa stane pri LED dátach sme si už popísali vyššie. Pri požiadavke na dáta vyberieme z obsahu paketu typ dát, ktoré si Galileo požiadalo a odošleme mu hodnotu. Napríklad takto odosielame hodnotu zo svetelného senzora:

```
if (dataapp[0] == 3) {
  //3 - Dostali sme poziadavku na data zo svetelneho senzora
  uint8_t dataToSend[10] = {};
  dataToSend[0] = 3; //Typ senzoru (3 - svetelny senzor)
  dataToSend[1] = 3; //Typ premennej (3 - Integer)
  memcpy((dataToSend + valueStart), &lux, 2);
  xbee_packet paket = make_paket(mojeA, destA, (uint8_t)random(100,
    60000), (uint8_t)5, dataToSend);
  xBeeSendUnicast(destA, paket);
}
```

Najprv si vytvoríme prázdny buffer pre dáta. Doň zapíšeme na prvú pozíciu číslo 3 – to znamená že odosielame hodnotu zo svetelného senzora. Na ďalšiu pozíciu zapíšeme tiež číslo 3 - znamená to, že posielame premennú typu `int`. Následne skopírujeme do buffera od druhej pozície obsah premennej `lux`. (premenná `valueStart` obsahuje číslo 2 - od ktorej pozície zapisujeme do buffera, predchadzajúce dve pozície sú vyhradené pre typ dát a premennej). Následne vytvoríme XBee paket pomocu funkcie `make_paket()`, ktorá nám vráti dátovú štruktúru `xbee_packet`.

```
xbee_packet make_paket(uint16_t srcAddr, uint16_t dstAddr, uint8_t
    seq_ID, uint8_t command_ID, uint8_t dataapp[10]) {
  xbee_packet paket;
  paket.srcAddr = srcAddr;
  paket.dstAddr = dstAddr;
  paket.seq_ID = seq_ID;
  paket.command_ID = command_ID;
  memcpy(paket.dataapp, dataapp, 10);
  return (paket);
}
```

Robíme tak z toho dôvodu, že túto štruktúru predávame funkcii `xBeeSendUnicast()` ktorá si tento paket zoberie a vytvorí z neho `Tx16Request` - dátový typ z XBee knižnice, ktorý tento paket prevedie na postupnosť bajtov – payload, v takej forme, akú sme si ukázali v sekcii 6.2.2. Ďalej si táto funkcia zoberie cieľovú adresu XBee modulu, teda adresu senzorického uzlu, na ktorý posielame dáta a pomocou funkcie `send()` volanej na náš `XBee` objekt tento paket odošle.

```
void xBeeSendUnicast(uint16_t destAddr, xbee_packet paket) {
  Tx16Request tx = Tx16Request(destAddr, (uint8_t *)&paket, sizeof(
    paket));
  xbee.send(tx);
}
```

Takýmto spôsobom funguje komunikácia aj pri Galileu.

### 6.2.5 Webové rozhranie

Keď máme vyriešenú komunikáciu medzi zariadeniami a internetom, musíme previesť dáta do rozumnej formy v podobe grafov. Na to vytvoríme na ThingSpeak-u 4 kanály pre naše štyri senzorické uzly. Každý kanál bude mať vytvorené 4 polia pre jednotlivé dáta, s takýmito číslammi:

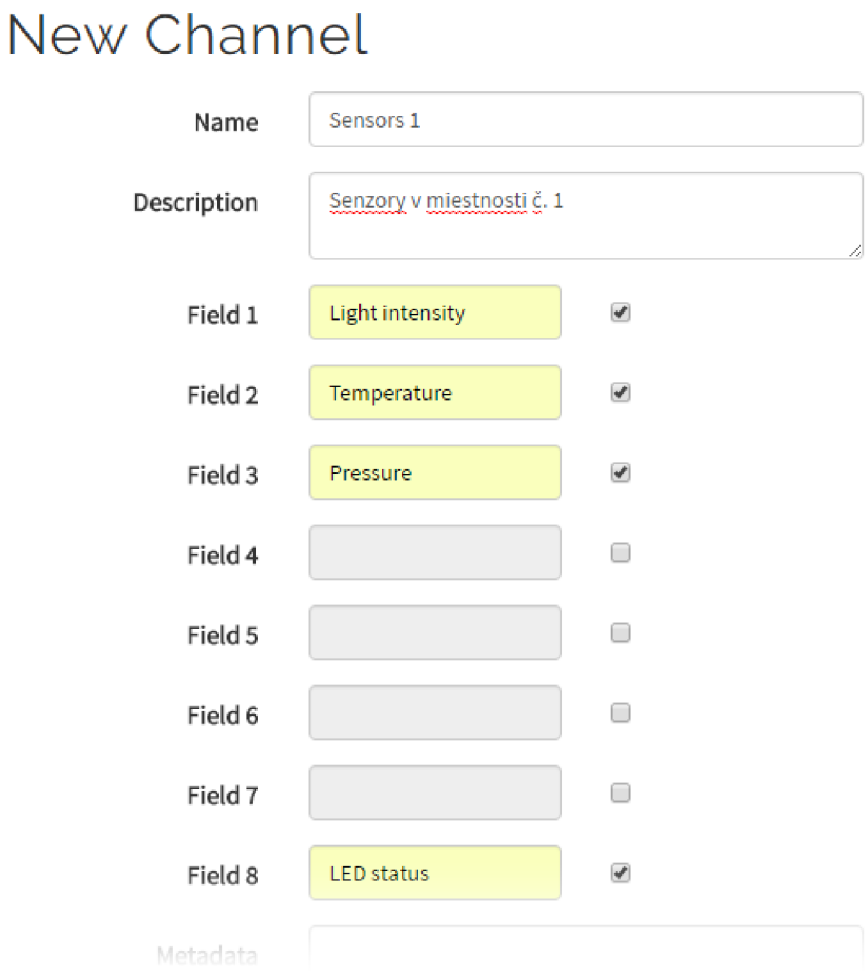
- 1. pole - Svetelná intenzita
- 2. pole - Teplota
- 3. pole - Tlak

- 8. pole - LED dáta

Posledné pole sme schválne umiestnili na poslednú 8. pozíciu, pretože sa jedná len o kontrolné dáta pre LED diódy, nemá zmysel ich vizualizovať. Toto posledné pole využíva skript, ktorý zobrazuje stav LED diódy.

### Vytvorenie kanálov dátových grafov

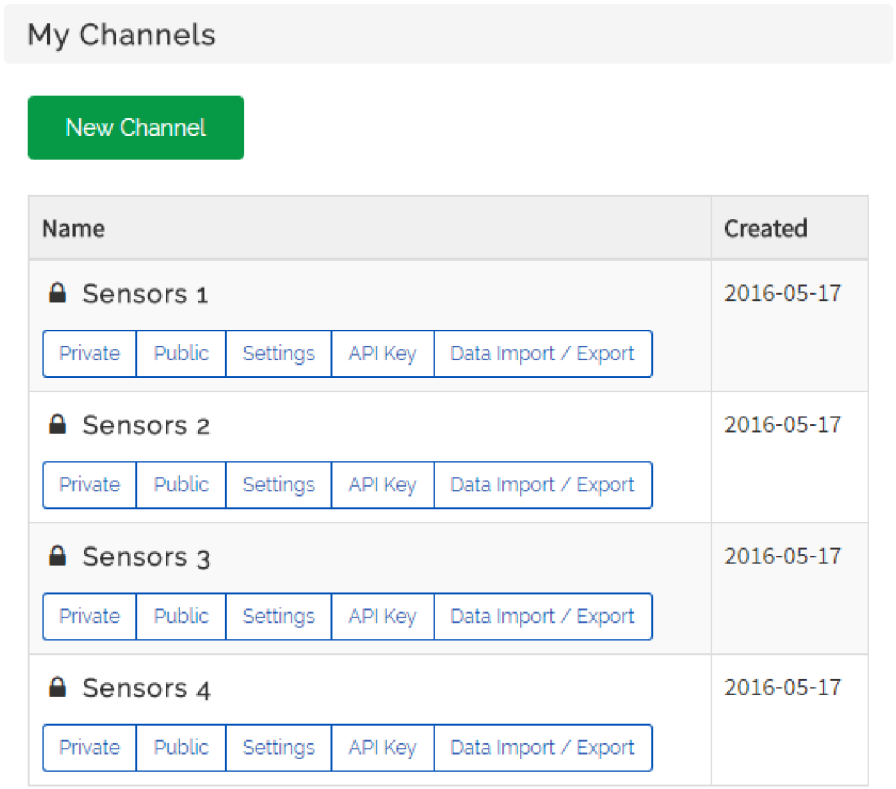
Kanál vytvoríme kliknutím v hlavnom menu stránky na `Channels > My Channels` a následne na tlačidlo `New channel` nad zoznamom kanálov. Viď obr. 6.9. Každý kanál má svoje meno, popis, môžme mu nastaviť ktoré polia budú aktívne, priradiť im názvy a dokonca môžme kanálu nastaviť GEO lokáciu, ktorá bude následne zobrazená na mape pri vizualizáciach dát. Na obr. 6.9 vidíme zoznam už vytvorených




Obr. 6.7: Webové rozhranie - vytváranie kanálu

kanálov. Každý kanál má dva typy zobrazení – súkromné a verejné. Do každého z nich môžme umiestniť iné vizualizácie. Ďalej nám stránka ponúka link na API

kľúče kanálu, nastavenia a import/export dát. ThingSpeak podporuje importovanie a exportovanie dát z/do CSV formátu. Po zobrazení kanálu sa nám naskytne




Obr. 6.8: Webové rozhranie - Výber kanálov

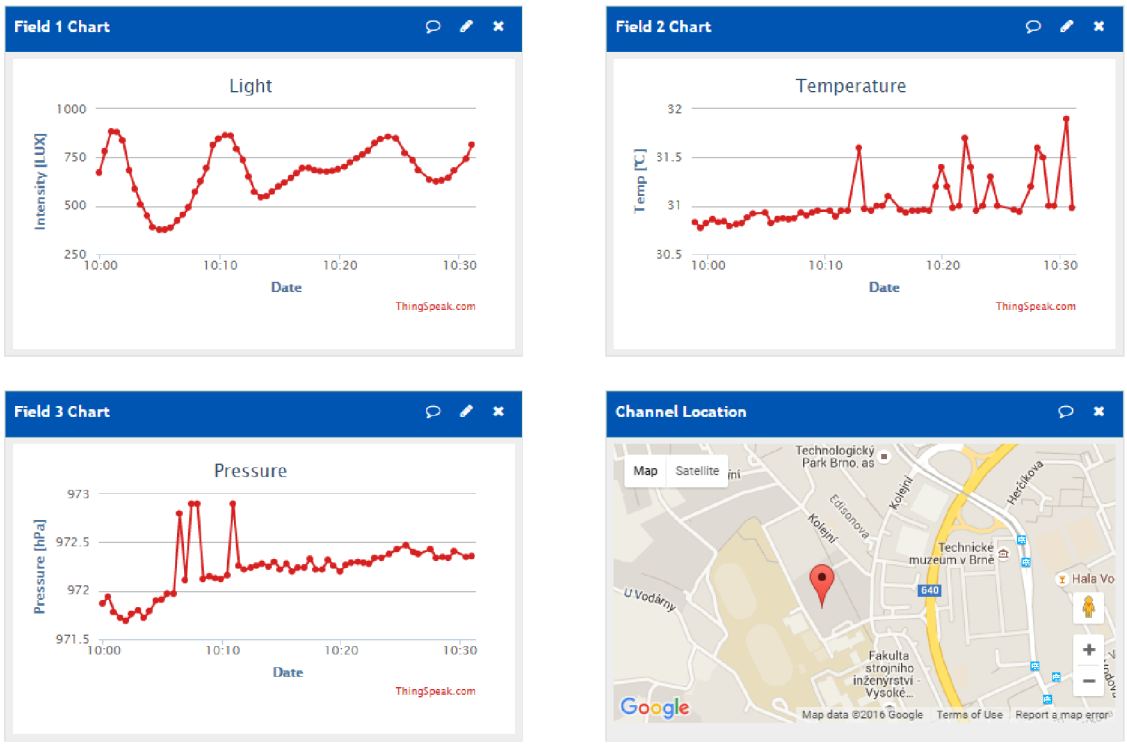
podobný pohľad ako na obr. 6.9. Automaticky nám boli vygenerované grafy dát z jednotlivých polí kanálu. Grafy sa dajú upravovať - zmenili sme im názvy a popisky ôs. Do kanálu môžme pridávať ďalšie vizualizácie a pluginy, ktoré si môžme vytvoriť z už predpripravených kódov (napr. Matlab vizualizácie), alebo si môžme vlastné napísať pomocou HTML, CSS a JavaScriptu.

### Ovládanie osvetlenia z webového rozhrania

Na ovládanie LED diód sme si práve vybrali možnosť napísať si vlastný plugin, ktorý sa zobrazí v kanáli. To pozostáva z dvoch krokov. Prvým bude vytvoriť TalkBack, ktorý bude zhromažďovať príkazy z pluginov, druhým bude samotné programovanie pluginu.

TalkBack vytvoríme zvolením položky `TalkBack` v sekcií `Apps`. Vytvorenie je jednoduché, stačí zvoliť `New TalkBack` a priradiť mu meno. Následne dostaneme API kľúč, ktorým naň budeme zapisovať príkazy.

Rovnakým spôsobom vytvoríme nový plugin - v menu `Apps` zvolíme `Plugin`. Pri vytváraní si zvolíme `Custom (no starter code)`, čo nám umožní napísať plugin




Obr. 6.9: Webové rozhranie - Detail kanálu

od základov. Prechádzame na editácu, kde nám stránka ponúka 3 textové polia obsahujúce HTML kód, CSS kód a JavaScript kód. V HTML časti sme vytvorili 3 tlačidlá a stavový text. Tlačidlami vieme ovládať stav osvetlenia (zapnuté, vypnuté, auto), text zobrazuje stav osvetlenia.

```
<body>
  <p>
    <button class="led-on">Zapni LED</button>
    <button class="led-off">Vypni LED</button>
    <button class="led-auto">Automaticke osvetlenie</button>
  </p>
  <p>
    <span class="led-status"></span>
  </p>
</body>
```

V CSS sme len nastavili biele pozadie.

```
<style type="text/css">
  body {
    background-color: #fff;
  }
</style>
```

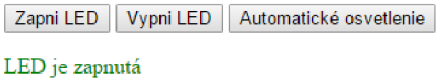
Celé jadro tohto pluginu (backend) tvorí JavaScript kód. Na vytvorenie algoritmu sme použili JavaScriptovú knižnicu jQuery, ktorá uľahčuje prácu s HTML prvkami (akcie po stlačení tlačidiel, zmena stavového textu a jeho farby, posielanie HTTP požiadaviek). Kód funguje tak, že po stlačení ovládacieho tlačidla pridá na náš TalkBack pomocou HTTP požiadavky nový príkaz, podľa toho, aké tlačidlo sme stlačili. Algoritmus tiež každých 5 sekúnd číta stav 8. poľa v príslušnom kanáli, ktoré obsahuje stav LED a podľa neho mení stavový text a jeho farbu. Príklad: Nasledujúca funkcia pošle po stlačení zapínacieho tlačidla príkaz na TalkBack:

```
$(".led-on")
  .click(function( event ) {
    event.preventDefault();
    $.post("https://api.thingspeak.com/talkbacks/8292/commands",
    {
      api_key: "GMY5EUTQZF1KTX70",
      command_string: "LED_ON_1"
    });
  });
```

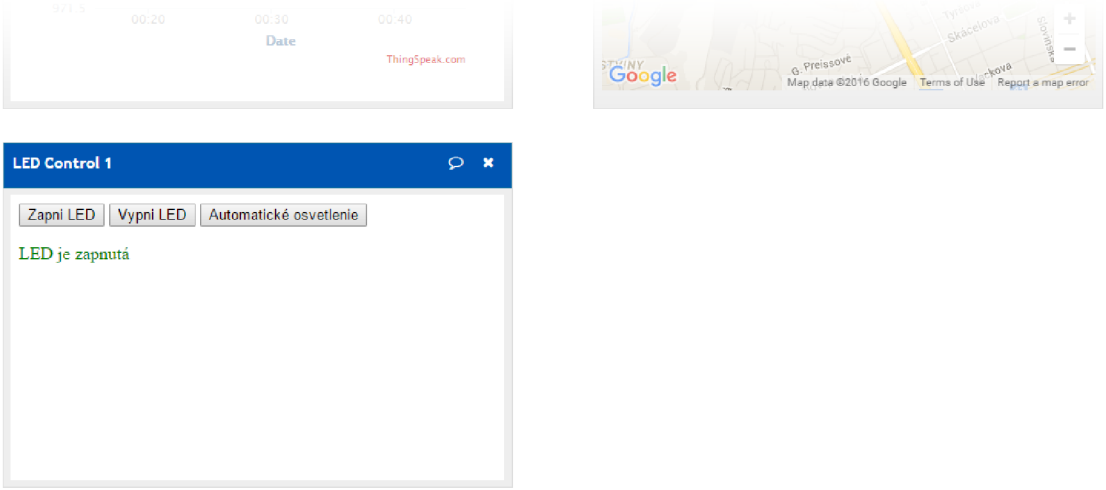
a táto zas prečíta dáta z kanálu a na ich základe zmení stavový text LED diódy:

```
$.getJSON("https://api.thingspeak.com/channels/117088/fields/8.json
   ?results=1&api_key=1V1XNKMK6O2GAJDT",
function(data) {
  console.log(data);
  $.each(data.feeds, function() {
    var channelValue = parseInt(this["field8"]);
    console.log("Channel value is: " + channelValue);
    if(channelValue == 0) {
      $(".led-status")
        .html("LED je vypnuta")
        .css('color', 'black');
    }
    if(channelValue == 1) {
      $(".led-status")
        .html("LED je zapnuta")
        .css('color', 'green');
    }
  });
});
```

Výsledný widget je vidno na obrázku 6.10 Pre každý kanál vytvoríme jeden plugin, ktorý bude čítať dáta zo svojho príslušného kanálu. Teraz už len stačí pridať tento plugin k ďalším vizualizáciam v kanáli. Viď obr. 6.11. Po stlačení ovládacieho tlačidla sa odošle príslušný príkaz pre príslušný senzorický uzol na TalkBack. Galielo si tento




Obr. 6.10: Webové rozhranie - Ovládací plugin




Obr. 6.11: Webové rozhranie - Ovládací plugin pridaný k ostatným vizualizáciam

príkaz prečíta a odošle hodnotu uzlu, ktorý si podľa nej nastaví stav osvetlenia. Neskôr si Galileo vypýta od uzlu stav osvetlenia, ktorý pošle na ThingSpeak kanál do 8. poľa, z ktorého si tento plugin prečíta hodnotu o stave LED a na jej základe nám oznámi, či je zapnutá alebo vypnutá.

Výsledkom práce je webové rozhranie, ktoré zobrazuje hodnoty zo senzorov a ovládacie prvky pre osvetlenie. Toto rozhranie komunikuje s vývojovou doskou Intel Galileo Gen2, ktorá zbiera dáta a ovláda 4 vývojové dosky Arduino Mega so senzormi. Zdrojové kódy sme navrhli tak, aby sa táto sieť dala ľahko škálovať.

# 7 ZÁVER

V tejto bakalárskej práci som sa zameral na Internet vecí a jeho využitie v reálnom svete. Venoval som sa vývojovým platformám a ich porovnaniu, spôsobmi komunikácie medzi IoT zariadeniami a protokolmi používanými v IoT, senzorom a senzorickým sieťam a ich vzťahu k IoT. Spracoval som návrh softvérovej a hardvérovej implementácie senzorických sietí v IoT a spracoval som možnosti programovania vývojových platforiem, komunikácie s internetom a vizualizácie dát. Realizoval som navhrhnutú senzorickú sieť postavenú na platforme Intel Galileo s bezdrôtovou komunikáciou medzi senzorickými doskami postavenými na Arduino platforme. Sieť komunikuje s internetom a na webovej platforme ThingSpeak môžme vizualizovať jednotlivé dáta zo senzorov v sieti, alebo ovládať osvetlenie pri senzoroch.

# LITERATÚRA


[1] Internet of Things. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-11-28]. Dostupné z: https://en.wikipedia.org/wiki/Internet_of_Things

[2] A Simple Explanation Of 'The Internet Of Things'. Forbes [online]. 2014, 2014-05-13 [cit. 2015-11-28]. Dostupné z: http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/

[3] Intel má dosku pre svoj miniatúrny procesor Quark. Mieri do škôl. Zive.sk [online]. 2013, 2013-10-04 [cit. 2015-11-28]. Dostupné z: http://www.zive.sk/clanok/69081/intel-ma-dosku-pre-svoj-miniaturny-procesor-quark-mieri-do-skol

[4] Intel Galileo Gen2. Arduino.cc [online]. 2014 [cit. 2015-11-28]. Dostupné z: https://www.arduino.cc/en/ArduinoCertified/IntelGalileoGen2 (Preložené z angličtiny)

[5] What is Arduino? Arduino.cc [online]. 2014 [cit. 2015-11-28]. Dostupné z: https://www.arduino.cc/en/guide/introduction (Preložené z angličtiny)

[6] What is BeagleBone? BeagleBone [online]. 2014, 2014-11-17 [cit. 2015-12-08]. Dostupné z: http://beagleboard.org/bone

[7] PocketLab [online]. 2015 [cit. 2015-12-09]. Dostupné z: http://thepocketlab.com/

[8] 11 Internet of Things (IoT) Protocols You Need to Know About. DesignSpark [online]. 2015 [cit. 2015-12-09]. Dostupné z: http://www.rs-online.com/designspark/electronics/knowledge-item/eleven-internet-of-things-iot-protocols-you-need-to-know-about

[9] What Is a Wireless Sensor Network? National Instruments [online]. 2012 [cit. 2015-12-10]. Dostupné z: http://www.ni.com/white-paper/7142/en/

[10] Comparison of major Linux package management systems. Linuxcareer.com [online]. [cit. 2015-12-10]. Dostupné z: http://how-to.linuxcareer.com/comparison-of-major-linux-package-management-systems

[11] SCHNEIDER, Stan. Understanding The Protocols Behind The Internet Of Things [online]. 2013-10-09 [cit. 2015-12-11]. Dostupné z: http://electronicdesign.com/iot/understanding-protocols-behind-internet-things

[12] Dweet.io [online]. [cit. 2015-12-12]. Dostupné z: https://dweet.io/

[13] Freeboard.io [online]. [cit. 2015-12-12]. Dostupné z: https://freeboard.io/

[14] Industry 4.0. Wikipedia [online]. [cit. 2015-12-12]. Dostupné z: https://en.wikipedia.org/wiki/Industry_4.0

[15] Linux.about.com. Linux / Unix Command: curl [online]. [cit. 2016-05-01]. Dostupné z: http://linux.about.com/od/commands/l/blcmdl1_curl.htm

[16] API - application program interface [online]. [cit. 2016-05-07]. Dostupné z: http://www.webopedia.com/TERM/A/API.html

# ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

| | |
|---|---|
| IoT | Internet vecí – Internet of Things |
| GPIO | General-purpose input/output |
| IDE | integrated development environment |
| JSON | JavaScript Object Notation |
| SSH | Secure Shell |
| HTTP | Hypertext Transfer Protocol |
| API | Application Program Interface |

# ZOZNAM PRÍLOH

# A OBSAH PRILOŽENÉHO CD

Priložené CD obsahuje elektronickú verziu tejto práce, zdrojové kódy pre Intel Galileo (brána) a Arduino Mega (senzorický uzol), Arduino knižnice pre XBee modul a senzory a zdrojový kód webového pluginu na ovládanie osvetlenia.

```
/
├── source
│   ├── galileo-code
│   │   └── galileo-code.ino (Zdrojový súbor pre Intel Galileo)
│   ├── libs (Arduino knižnice)
│   │   ├── BH1750.zip
│   │   ├── BMP180.zip
│   │   └── xbee-arduino.zip
│   └── sensor-node-code
│       └── sensor-node-code.ino (Zdrojový súbor pre senzorické uzly)
├── thingspeak-plugin (Ovládací plugin pre ThingSpeak)
│   ├── css.html (CSS časť kódu)
│   ├── html.html (HTML časť kódu)
│   └── javascript.html (JavaScript časť kódu)
└── praca.pdf (Elektronická verzia práce)
```