Komponent integračného frameworku pre prácu s teplotou

Matej Perejda

4Ib, 2016 - 2017

Abstrakt. V súčasnej dobe IT priemysel nedisponuje komplexným balíkom vývojových nástrojov pre prácu s Internet of Things (IoT). Túto absenciu na trhu by dokázal v budúcnosti vyriešiť projekt Silverspoon.io postavený na komponentoch integračného frameworku Apache Camel. Cieľom tejto práce je vyvinúť Camel komponent, ktorý dokáže načítavať a spracovávať dáta z tepelného senzora pripojeného k zariadeniu Raspberry Pi. Súčasťou práce je tiež analýza dostupných knižníc pre nízkoúrovňovú prácu s periférami a analýza existujúcich IoT platforiem.

Kľúčové slová: komponent, tepelný senzor, Internet of Things, Apache Camel

1 Úvod

Narastajúca popularita Internet of Things (Internet vecí) vytvára dopyt po stále väčšom počte aplikácií a systémov postavených na tomto koncepte. Realizácia takýchto riešení nie je v súčasnej dobe jednoduchá a dostupnosť jednotných, otestovaných vývojárskych nástrojov je nedostačujúca. Open-source projekt Silverspoon.io do ktorého vývoja sme sa zapojili, má za cieľ prispieť k vývoju komunity IoT a napomôcť pri ľahšej implementácii a navrhovaní takýchto riešení. Na dosiahnutie tohto cieľa je potrebné vyvinúť základné stavebné bloky, Camel komponenty, z ktorých sa platforma Silverspoon.io skladá. Integračný framework Apache Camel umožňuje jednoduché budovanie komplexného systému a integruje schopnosti embedded (vstavaných) zariadení s externými aplikáciami, ktoré využívajú viacero komunikačných protokolov a technológií.

Hlavné ciele tejto práce sme vymedzili nasledovne:

1. Analyzovať dostupné platformy pre komplexnú prácu s Internet of Things
2. Analyzovať existujúce knižnice pre nízkoúrovňovú prácu so senzormi
3. Navrhnúť a implementovať nový Apache Camel komponent
4. Otestovať vytvorený komponent

1.1 Apache Camel

Budovanie komplexných počítačových systémov (aplikácií) od úplného počiatku je veľmi náročný a zdĺhavý proces, ktorý kladie veľké požiadavky na ľudské zdroje. Tento proces je možné uľahčiť a zefektívniť spôsobom, ktorý by sme mohli prirovnať ku skladaniu puzzle. K dispozícii máme komponenty (jednotlivé časti), ktoré navzájom spájame, čím postupne po častiach vytvárame kompaktný systém.

Apache Camel[1] je integračný framework vytvorený v roku 2007 za účelom zjednodušenia integrácie softvéru. Integrácia softvéru znamená vytváranie jednotného systému zloženého z viacerých podsystémov. Ďalšou jeho významnou myšlienkou je efektívne minimalizovať množstvo napísaného zdrojového kódu potrebného pri implementovaní rôznych integračných scenárov a využiť potenciál teórie *Enteprise Integration Patterns*[2]. Napriek tomu, že je z veľkej časti napísaný v Jave, podporuje písanie kódu vo viacerých programovacích jazykoch súčasne. Umožňuje to jeho používateľom využívať silné stránky každého jazyka a ich kombináciou riešiť úlohy efektívnejšie. Camel nekladie takmer žiadne nároky na prostredie v ktorom je nasadzovaný a je ho možné využiť napríklad pri budovaní samostatných spustiteľných programov, ako aj webových aplikácií. Na tomto rámci je vybudovaných viacero open-source projektov ako je napr. *Apache ServiceMix* a *ActiveMQ*.

V jeho jadre sa nachádza *routing-engine builder*, ktorý umožňuje definovať vlastnú smerovaciu konfiguráciu, rozhodnúť z akých zdrojov budú správy prijímané, do ktorých častí sa budú posielať a akým spôsobom sa budú spracovávať. Tieto smerovacie pravidlá sú zapísané pomocou doménového špecifického jazyka DSL (*Domain-Specific Language*), ktorý Camel naplno využíva. Najčastejšie sú využívané implementácie DSL v jazyku Java a XML. Formát DSL nie je viazaný iba na jazyk Java a XML, ale taktiež aj na jazyky Groovy, Ruby alebo Scala.

Jeho veľkou výhodou je možnosť spravovania chýb a automatického testovania pomocou Camel Extension alebo JUnit. Vývojárom ponúka vyššiu úroveň abstrakcie, ktorá umožňuje pracovať s rôznymi systémami pomocou rovnakého rozhrania API nezávisle na protokole alebo dátovom formáte, ktoré systém používa.

Camel pracuje na princípe opakovaného využitia komponentov, čím vlastne redukuje veľké množstvo kódu. Aktuálne poskytuje rozsiahlu knižnicu pozostávajúcu z viac ako 120 oficiálne vytvorených komponentov[3] pre použitie akejkoľvek technológie. Tie poskytujú konkrétne implementácie rozhrania API zameriavajúce sa na odlišné protokoly a dátové typy. Rozšíriteľná a modulárna architektúra Camelu umožňuje implementovať a bez problémov pridať podporu pre akýkoľvek iný nový komponent, ktorý ešte nie je k dispozícii.

---

[1] https://camel.apache.org
[2] http://www.enterpriseintegrationpatterns.com/
[3] http://camel.apache.org/components.html

1.2 Bulldog

Bulldog[4] je voľne dostupná a ľahko použiteľná Java knižnica, ktorá poskytuje IoT vývojárom podporu prístupu k rôznym zberniciam vstavaných linuxových platforiem ako BeagleBone Black, Raspberry Pi a CubieBoard. Jej výhodou je prenositeľnosť napísaného kódu medzi platformami bez nutnosti vykonávania väčších zmien. Aktuálne podporuje prácu so vstupno-výstupnými funkciami rozhrania GPIO, native PWM, ADC, I^2C a SPI. Okrem iného podporuje všetky typy sériovej komunikácie UART a možností pripojenia periférnych zariadení LCD, LED, rôznych servomotorov a senzorov. Výhodou tejto knižnice je priama práca s pamäťou, vysoká modularita a jednoduchosť pridania podpory pre novú platformu.

1.3 Internet of Things

Internet of Things (IoT) je sieť fyzických objektov, zariadení, vozidiel, budov a iných „vecí“ pripojených k internetu drôtovo alebo bezdrôtovo. IoT umožňuje zasielať dáta (príkazy) k aktuátorom, zbierať dáta vzdialených senzorov, prenášať ich a spracovávať pomocou siete. Tieto senzory a aktuátory používajú rôzne typy pripojení ako RFID, NFC, Wi-Fi, Bluetooth, Z-Wave a iné. Zariadenia sú pripájané do siete s cieľom pristupovať k internetovým službám, komunikovať s ľuďmi a medzi sebou.

Pre užívateľov má IoT potenciál priniesť riešenia, ktoré môžu výrazne zlepšiť bezpečnosť, zdravie, vzdelanie a mnoho ďalších aspektov. Podnikom môže napomôcť pri podpore riešení, ktoré zlepšia rozhodovaciu schopnosť strojov a produktivitu v oblasti výroby. Napríklad budova využívajúca senzory k automatickému nastaveniu kúrenia a osvetlenia alebo výrobné zariadenie, ktoré upozorní personál údržby na hroziace zlyhanie.

2 Aktuálne riešenia

2.1 Platformy pre komplexnú prácu s Internet of Things

2.1.1 Základné typy platforiem

M2M (Machine-to-Machine) je spôsob komunikácie, ktorý prebieha medzi zariadeniami so schopnosťou komunikovať samostatne, bez ľudského zásahu. M2M využíva tieto zariadenia na monitorovanie určitých udalostí pomocou senzorov, alebo na posielanie pokynov k ich ovládaniu. Zachytené udalosti (napr. rýchlosť, teplota) sú prenášané cez káblové, alebo bezdrôtové siete na centrálny server, ktorý tieto informácie zbiera, spracováva a automaticky ovláda ostatné stroje. Odpoveďou servera môže byť správa zariadeniu k zmene jeho stavu (napr. vypnúť, spomaliť).

[4] https://github.com/SilverThings/bulldog

PaaS (Platform as a Service) je model výpočtového mračna (cloud computingu), ktorý poskytuje platformu a prostredie umožňujúce vytvárať internetové aplikácie a služby. Cloud provider poskytuje vývojárom rôzne hardvérové a softvérové nástroje (programovacie jazyky, knižnice...). Služby PaaS sa nachádzajú v cloude a sú prístupné používateľom jednoducho, prostredníctvom webového rozhrania. Známym typom PaaS platforiem sú *IBM BlueMix*, *Microsoft Azure* a *Red Hat OpenShift*.

SaaS (Software as a Service) opisuje ďalšiu z kategórií výpočtového mračna, tzv. cloud computingu. Tento model poskytuje aplikácií cez internet vo forme vzdialených služieb. Užívateľ pristupuje k službe, ktorá je umiestnená v cloude, pomocou siete. Nie je potrebná žiadna lokálna inštalácia softvéru ani hardvéru. Benefity používania tohto modelu predstavujú jednoduché spravovanie, automatickú aktualizáciu, kompatibilitu a globálnu dostupnosť. Príkladom SaaS služieb sú napríklad *Microsoft Office 365* a všetky Google aplikácie (Gmail, Disk,...).

2.1.2 Súhrnný prehľad vlastností analyzovaných IoT platforiem

Tabuľka 1. uvádza skúmané platformy podľa analýzy [3] a sumarizuje niektoré základné charakteristiky – použitá architektúra, typ platformy, otvorenosť zdrojového kódu a podpora REST-ového API. Tento prehľad predstavuje rýchle vizuálne informácie analyzovaných platforiem.

Tabuľka 1. Analýza dostupných IoT platforiem

#	Platforma	Architektúra	Typ	Open source	REST
1	AirVantage™	cloud	M2M PaaS	iba knižnice	áno
2	Arkessa	cloud	M2M PaaS	nie	
3	ARM mbed	centralizovaná/cloud	M2M PaaS	nie	
4	Carriots®	cloud	PaaS	nie	áno
5	DeviceCloud	cloud	PaaS	nie	áno
6	EveryAware	centralizovaná	server	nie	áno
7	Everyware	cloud	PaaS	nie	áno
8	EvryThng	centralizovaná	M2M SaaS	nie	áno
9	Exosite	cloud	PaaS	iba knižnice	áno
10	Fosstrack	centralizovaná	server	nie	nie
11	GroveStreams	cloud	PaaS	nie	áno
12	H.A.T.	decentralizovaná	PaaS	áno	áno
13	IoT-framework	centralizovaná	server	Apache license 2.0	áno
14	IFTTT	centralizovaná	SaaS	nie	nie
15	Kahvihub	centralizovaná	server	Apache license 2.0	áno
16	LinkSmart™	decentralizovaná	P2P	LGPLv3	nie
17	MyRobots	cloud	roboty PaaS	nie	áno
18	Niagara^AX	distribuovaná	M2M SaaS	nie	
19	Nimbits	centralizovaná/cloud	server	Apache license 2.0	áno
20	NinjaPlatform	cloud	PaaS	hardware, operačný systém	áno
21	Node-RED	centralizovaná	server	Apache license 2.0	nie
22	OpenIoT	decentralizovaná	hub	LGPLv3	nie
23	OpenMTC	centralizovaná/cloud	M2M client server	nie	áno
24	OpenRemote	centralizovaná	server	Affero GNU	áno
25	Open.Sen.se	cloud	PaaS /SaaS	nie	áno
26	realTime.io	cloud	PaaS	nie	áno
27	SensorCloud™	cloud	PaaS	nie	áno
28	SkySpark	centralizovaná/cloud	SaaS	nie	áno
29	Swarm	cloud	PaaS	klient	áno
30	TempoDB	cloud	PaaS	nie	áno
31	TerraSwarm	decentralizovaná	OS	-	
32	The thing system	centralizovaná	server	M.I.T.	áno
33	Thing Broker	centralizovaná	server	áno	áno
34	ThingSpeak	centralizovaná/cloud	server	GNU GPLv3	áno
35	ThingSquare	cloud	mesh	gateway firmware	áno
36	ThingWorx	cloud	M2M PaaS	nie	áno
37	WoTkit	cloud	PaaS	nie	áno
38	Xively	cloud	PaaS	iba knižnice	áno

2.2 Knižnice pre nízkoúrovňovú prácu s periférami vstavaných zariadení

Počas prvotného prieskumu sme zistili, že nízkoúrovňové knižnice pre prácu s periférami nie sú predmetom žiadnych výskumov alebo obšírnejších analýz. Zdroje vo vedeckých prácach zaoberajúce sa touto tématikou sú obmedzené, preto sme sa snažili čerpať informácie z najdôveryhodnejších zdrojov a open-source projektov dostupných na internete, nachádzajúcich sa najmä vo verejných repozitároch. Mnohé z týchto knižníc majú vytvorené oficiálne webové stránky na ktorých je prístupná aj podrobná dokumentácia.

2.2.1 Súhrnný prehľad vlastností analyzovaných knižníc

Tabuľka 2. Prehľad vlastností analyzovaných knižníc

#	Názov	Jazyk	GPIO I/O	SPI	I^2C	PWM	UART
1	Bcm2835	C, C++	✓	✓	✓	✓	✓
2	Bulldog	C, Java	✓	✓	✓	✓	✓
3	Framboos	Java, Scala	✓	x	x	x	✓
4	Johnny-Five	JavaScript	✓	✓	✓*	✓	✓*
5	Libmraa	C, C++	✓	✓	✓	✓	✓
6	LibsensorPy	Python	✓	✓	✓	x	x
7	Pi4J	C	✓	✓	✓	✓	✓
8	Pigpio	Python	✓	✓	X	✓	x
9	PiLowLib	C#	✓	x	x	x	x
10	RaspberryGPIOManager	C#	✓	✓	?	✓	?
11	RaspberryPi.Net	C#	✓	✓	✓	x	x
12	Raspberry# IO	C++, JavaScript, Python	✓	x	✓	✓	✓
13	Raspi.js	C, Java	✓	✓	✓	✓	✓
14	Xively	C, C++	✓	✓	✓	✓	✓

✓ - vlastnosť je podporovaná
✓* - podpora vlastnosti sa líši vzhľadom na použitú platformu
x - vlastnosť nie je podporovaná
? - vlastnosť nie je známa

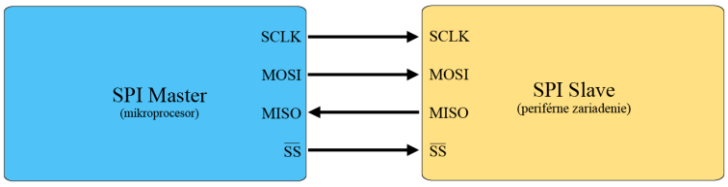
3 Návrh riešenia

3.1 Implementácia tepelného senzora Texas Instruments LM74

3.1.1 Serial Peripheral Interface

SPI je sériová komunikačná zbernica navrhnutá spoločnosťou Motorola pracujúca vo full-duplexovom móde, ktorej zariadenia komunikujú spôsobom master-slave (obrázok. 1). Toto sériové rozhranie vyžaduje štyri signály:

- SCLK: hodinový signál (clock)
- MOSI: master výstup/slave vstup (master output/slave input)
- MISO: master vstup/slave výstup (master input/slave output)
- SS: signal slave zariadenia (slave select)



Obrázok 1. Náčrt SPI komunikácie medzi master slave zariadením

Na sériovej zbernici sú signály SCLK, MOSI a MISO zdieľané všetkými pripojenými zariadeniam. Prenos dát je zosynchronizovaný s hodinovým impulzom master zariadenia, ktorý ho generuje vlastnými sériovými hodinami SCLK. Slave zariadenie vôbec neovplyvňuje tento impulz a dokáže iba prijímať a zasielať zosynchronizované údaje. Full-duplexový mód umožňuje prijímať a odosielať dáta obom stranám SPI komunikácie nezávisle v rovnakom čase. Na prenos dát slúžia MOSI a MISO dátové linky, kde smer prenosu indikujú ich názvy. MOSI pin na master zariadení - prostredníctvom ktorého sú zasielané bity z master zariadenia - sa pripája k MOSI pinu slave zariadenia. Analogicky, pomocou MISO pinu na slave zariadení sú bity odosielané smerom k master zariadeniu a sú prijímané jeho MISO pinom.

Adresovanie dát sa realizuje pomocou SS liniek. Master zariadenie najprv uskutoční výber slave zariadenia cez príslušný SS pin a až potom začne vysielať dáta smerom k slave zariadeniu, ktoré tieto dáta akceptuje. SPI je veľmi jednoduchý komunikačný protokol, ktorého použitie je vhodné najmä na prenos dát vo vstavaných systémoch. Použitie SPI v prípadoch, kedy je k jednému master zariadeniu pripojených viacero slave zariadení je možné iba vtedy, ak master zariadenie disponuje potrebným počtom SS výstupov.

3.1.2 Tepelný senzor a prepojenie zariadenia RPi

Prepojenie tepelného senzora so zariadením RPi bolo veľmi jednoduché. Postupovali sme podľa priloženej schémy zapojenia, nachádzajúcej sa v dokumentácii senzora. Využili sme pri tom stredne veľké nepájivé kontaktné pole (tzv. breadboard), keramický kondenzátor s kapacitou 0.1µF, rezistor s odporom 10kΩ, prepojovacie vodiče rôznych farieb (pre lepšiu orientáciu v zapojení) a tepelný senzor Texas Instruments LM74 s 3-wire SPI komunikačným rozhraním. Tabuľka 3. ilustruje obojstranné prepojenie pinov medzi zariadením Raspberry Pi a senzorom LM74, uvádza čísla použitých pinov a názvy signálov.

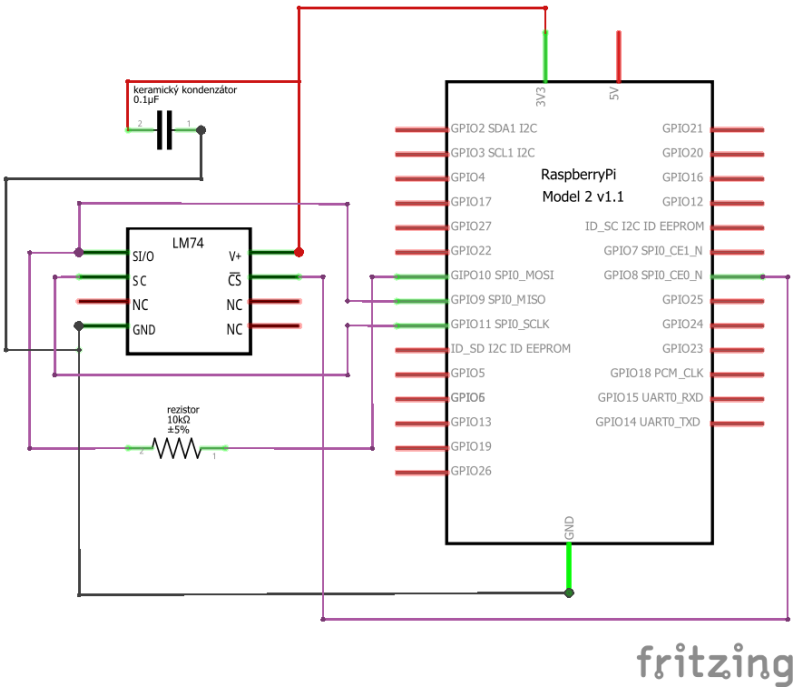
Tabuľka 3. Obojstranné prepojenie pinov medzi zariadením RPi a senzorom LM74

LM74 pin	LM74 signál	RPi pin	RPi signál
1	SI/O	19,21	SPI_MOSI, SPI_MISO
2	SC	23	SPI_CLK
3	NC (nepripojený)		
4	GND	20	Ground
5	NC		
6	NC		
7	CS	24	SPI_CE0
8	V^+	17	DC Power 3.3V

Tzv. oddeľovací kondenzátor funguje ako lokálne úložisko energie pre dané zariadenie a vkladá sa do obvodu medzi linku napájania a uzemnenia. Jeho úlohou je potlačiť vyskofrekvenčný šum signálu sieťového napájania, ktorý by inak mohol poškodiť integrovaný obvod. V prípade, ak je dočasne znížené napätie napájania senzora, kondenzátor môže krátko pôsobiť ako zdroj napätia a dodávať potrebnú energiu. Schéma typickej aplikácie senzora obsiahnutá v jeho dokumentácii si vyžadovala umiestnenie takéhoto kondenzátora medzi pin sieťového napájania (ozn. V^+) a uzemnenia senzora (ozn. GND). Potreba zapojenia rezistora do obvodu spočíva v oddelení oboch dátových liniek MISO a MOSI, čím sa zabráni vzájomným vplyvom týchto liniek a predíde sa skresleniu dát počas master-slave komunikácie.

Schému zapojenia obvodu sme vytvorili pomocou open-source nástroja *Fritzing*[5] a je znázornená na obrázku 2.

[5] http://fritzing.org/



Obrázok 2. Schéma prepojenia zariadenia Raspberry Pi so senzorom TI LM74

3.1.3 Implementácia senzora do Bulldog knižnice

Bulldog knižnica poskytuje veľké množstvo metód, ktoré dokážu jednoducho a elegantne pracovať s rozhraním SPI. Vývojárom teda zjednodušujú implementovanie SPI komunikácie. Keďže celú komunikáciu medzi master a slave zariadením dokážu zabezpečiť metódy Bulldog knižnice, našou úlohou bolo tieto metódy efektívne využiť, prenesenú zakódovanú, dvojbajtovú informáciu od tepelného senzora zachytiť, spätne ju pomocou metódy dvojkového doplnku[6] odkódovať a teplotu vrátiť.

[6] https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html

3.2 Implementácia Apache Camel komponentu pre prácu s teplotou

Najjednoduchším spôsobom vytvárania Apache Camel projektov je využitie nástroja *Apache Maven*[7], ktorý dokáže dômyselne spravovať závislosti a napomáhať tak pri automatickom zostavovaní softvéru. Camel disponuje viacerými predpripravenými šablónami (Maven archetypes) pomocou ktorých je vytváranie rôznych typov integračných projektov pohodlnejšie. Jedna z takýchto šablón, ktorú sme využili, je *camel-archetype-component*, ktorá nakonfiguruje dôležité závislosti v POM súbore a vygeneruje štandardnú kostru nového Camel komponentu. Pre generovanie môžeme využiť nižšie uvedený príkaz volaný z príkazového riadka nad adresárom nášho projektu, ktorý vytvorí nový komponent v adresári *camel-temperature* so štyrmi dôležitými triedami v balíčku *io.silverspoon.*

```
mvn archetype:create \
-DarchetypeGroupId=org.apache.camel.archetypes \
-DarchetypeArtifactId=camel-archetype-component \
-DarchetypeVersion=2.9.0 \
-DgroupId=io.silverspoon \
-DartifactId=camel-temperature
```

Štruktúra novovytvoreného komponentu vyzerá nasledovne:

```
camel-temperature
|-- pom.xml
|-- ReadMe.txt
`-- src
|-- main
    | |-- java
    | | `-- io
    | |   `-- silverspoon
    | |    |-- TemperatureComponent.java
    | |    |-- TemperatureConsumer.java
    | |    |-- TemperatureEndpoint.java
    | |    `-- TemperatureProducer.java
    | `-- resources
    |    |-- log4j.properties
    |   `-- META-INF
    |      `-- services
    |       `-- org
    |        `-- apache
    |         `-- camel
    |          `-- component
    |           `-- temperature
  `-- test
   `-- java
    `-- io
     `-- silverspoon
      `-- TemperatureComponentTest.java
```

[7] https://maven.apache.org/

Takto vygenerovaný komponent je plne funkčný a pripravený na jeho budúci vývoj. Obsahuje triedu `TemperatureConsumer`, ktorá aktuálne reprezentuje jednoduchého konzumenta generujúceho primitívne správy v pravidelných časových intervaloch. Trieda `TemperatureProducer` predstavuje producenta, ktorý prijaté správy vypisuje priamo do konzoly. Tzv. továrňou na koncové body - endpointy - je trieda `TemperatureComponent`. Úlohou triedy `TemperatureEndpoint` je modelovanie správ a vytváranie objektov `TemperatureProducer` a `TemperatureConsumer`.

Akonáhle sme vygenerovali základnú štruktúru komponentu, naším cieľom bolo upraviť triedu `TemperatureEndpoint` tak, aby zabezpečovala inicializáciu senzora LM74 a definovala MOSI pin SPI komunikácie, na ktorý sa predvolene senzor bude pripájať. Triedu `TemperatureConsumer` sme jemne upravili, aby namiesto generovania primitívnych správ, generovala správy, ktoré nesú údaje o načítanej teplote. Nakoniec trieda `TemperatureProducer` prijaté správy od konzument nevypisuje do konzoly, ale posiela ich ďalej na spracovanie.

4 Záver

Počas tvorby práce sme si osvojili základné vymoženosti nízkoúrovňových rozhraní ARM zariadení a naučili sme sa ich programovo riadiť. Oboznámili sme sa s konceptom Internet of Things a open-source projektom Silverspoon.io, pričom sme sa naučili vytvárať komponenty Apache Camel frameworku. Prispeli sme rozšírením knižnice Bulldog o vlastnú implementáciu senzora Texas Instruments LM74, ktorý komunikuje pomocou neštandardného komunikačného protokolu 3-Wire SPI. Do projektu Silverspoon.io sme pridali vlastný Camel komponent, ktorý využíva už spomínanú implementáciu senzora z Bulldog knižnice. V našej práci sme sa tiež zaoberali analýzou existujúcich knižníc, pomocou ktorých je možné komunikovať s periférami vstavaných zariadení a výsledky analýzy sme zhrnuli v tabuľke, ktorá predstavuje súhrnný prehľad najčastejšie využívaných komunikačných protokolov pre dané knižnice. V práci taktiež nechýba analýza už existujúcich IoT platforiem a tabuľka, ktorá sumarizuje ich niektoré základné charakteristiky.

Poďakovanie. Moje poďakovanie patrí vedúcemu práce Mgr. Marošovi Andrejkovi a taktiež konzultantovi Mgr. Štefanovi Bunčiakovi z firmy Red Hat Czech, s.r.o. za ich pomoc, čas a cenné rady, ktoré mi venovali pri tvorbe bakalárskej práce.

Literatúra

1. Ibsen, C., Anstey, J.: Camel in Action. Greenwich, Conn.: Manning, 2011, xxxi, 516 p. ISBN 1935182366.
2. Mcewen, A., Cassimally, H.: Designing the internet of things. Chichester, England: Wiley, 2014, 1 online resource (338 pages). ISBN 9781118430651.
3. Mineraud, J., Mazhelis, O., Su, X., Tarkoma, S.: A gap analysis of Internet-of-Things platforms. arXiv preprint arXiv:1502.01181, 2015.

4. Kim, J., Lee, J., Kim, J., Yun, J.: M2M service platforms: Survey, issues, and enabling technologies, IEEE Communications Surveys Tutorials 16 (1) (2014)