Domain Specific Software Engineering: Internet of Things

Seminar Domain Specific Software Engineering

Version 0.1

Zürcher Hochschule für Angewandte Wissenschaften

Daniel Brun

xx. Juni 2015

Eigenständigkeitserklärung

Hiermit bestätigt ich, dass vorliegende Seminararbeit zum Thema „Evaluation einer Mini ERP Lösung für einen Verein“ gemäss freigegebener Aufgabenstellung ohne jede fremde Hilfe und unter Benutzung der angegebenen Quellen im Rahmen der gültigen Reglemente selbständig verfasst wurde.

Thalwil, 11. Februar 2015 Daniel Brun

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 Hintergrund

Im Rahmen meines Bachelor-Studiums in Informatik an der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) muss im 6. Semester eine Seminararbeit zu einem vorgegebenen Themenbereich erarbeitet werden. Ich habe mich für den Themenbereich „Domain Specific Software Engineering“ entschieden.

Aus einem Themenkatalog konnte ein spezifisches Thema im Bereich „Domain Specific Software Engineering“ ausgewählt werden. Ich habe mich für das Thema „Internet of Things“ entschieden.

Für die Arbeit sollen circa 50 Arbeitsstunden aufgewendet werden. Dies entspricht etwa einem Umfang von 15 bis 20 Seiten. Zusätzlich gelten die Rahmenbedingungen gemäss dem Reglement zur Verfassung einer Seminararbeit ([**ZHAW:2012:Seminararbeit:Reglemente**])

1.2 Aufgabenstellung

Es soll ein Dokument zum Thema Domain Specific Software Engineering im Bereich Internet of Things erstellt werden. Das Papier soll die Schwierigkeiten der Software-Entwicklung in diesem Bereich aufzeigen und einen groben Überblick über das Thema eben.

1.3 Abgrenzung

Abgrenzung

1.4 Motivation

In den letzten Monaten bin ich immer wieder auf das Thema „Internet of Things“ aufmerksam geworden. Seit meiner Teilnahme an der Microsoft Konferenz in Barcelona Ende 2014 (TechEd Europe) war meine Neugier definitiv geweckt und ich hatte mir bereits einige Projekte überlegt, welche ich allenfalls als Semesterarbeit, Seminararbeit oder in einem anderen Schulprojekt realisieren könnte. Mit dem Seminar „Domain Specific Software Engineering“ hat sich mir nun eine erste Gelegenheit geboten, um mich in dieses Thema zu vertiefen. An dem Thema fasziniert mich vor allem das Breite Spektrum an Innovations- und Kombinationsmöglichkeiten.

1.5 Struktur

Diese Arbeit gliedert sich in folgende Hauptteile:

- Ausgangslage
- Die Domäne „Internet of Things“
- Software Engineering in der Domäne „Internet of Tings“
- Case Study „iotivity“

Struktur abschliessend verifizieren

Im ersten Kapitel werden die Details zur Ausgangslage und die Hintergründe der Arbeit aufgezeigt. Im zweiten Kapitel wird die Ausgangslage in Bezug auf die betrachteten Themen „Internet of Things“, „Domain Specific Software Engineering“ und „Domain Specific Languages“ kurz erläutert. im darauffolgenden Kapitel wird die Domäne „Internet of Things“ im Detail beschrieben und aufgezeigt, was für Anforderungen an das Software Engineering bestehen. Im Kapitel 4 werden nachher die in der Domäne eingesetzten Standards, Protokolle und Frameworks beschrieben. Im Kapitel 5 wird anschliessend eine kleine Fallstudio, beziehungsweise ein Praxistest mit dem Framework “iotivity“ durchgeführt. Im letzten Kapitel wird ein Fazit gezogen, eine Empfehlung an den abgegeben und über die gesamte Arbeit reflektiert.

KAPITEL 2

Ausgangslage

Mit dem laufenden Fortschritt in der Computer- und Kommunikationstechnik und der damit einhergehenden Miniaturisierung und Mobilisierung eröffnen sich immer wieder neue Bereiche in der Informatik. Eines dieser neuen Gebiete wird als „Internet of Things“ bezeichnet.

2.1 Internet of Things

Nach dem Aufkommen der Personal Computer, der Etablierung des Internets und dem Mobile-Trend der letzten Jahre, stellt das IoT oder „Internet der Dinge“ die nächste Etappe in der Entwicklung des Internets dar. Diese Etappe wird wiederum die Art und Weise wie Leute und vor allem Dinge miteinander kommunizieren und interagieren grundlegend verändern.

Der IoT-Markt wird grosses Entwicklungspotenzial zugeschrieben, ein grösses sogar als vor einigen Jahren dem Smartphone Markt.

Prognose / Zahlen –> Aus PDF von LIEBH

Mit der Etablierung des IoT wird eine Transformation von isolierten Geräten hin zu vernetzten Geräten statt finden. In Zukunft wird vermehrt die Kommunikation von Geräten untereinander und mit dem Internet im Vordergrund stehen. Im Privatbereich wird sich vieles, wenn nicht sogar alles, um die Heimautomation drehen. Wird von der Wetterstation ein Sturm gemeldet, können automatisch die Sonnenstoren eingefahren, die Fenster geschlossen und die Fensterläden heruntergelassen werden. Ein anders Beispiel wäre, dass wenn man nach Hause kommt automatisch beim Aufschliessen der Türe das Licht angeht und sich der Radio einschaltet. Die Heimautomation kann auch als Einbruchsschutz während längeren Abwesenheiten genutzt wurden. Durch den Einsatz von Machine-Learning kann das System die Gewohnheiten der Bewohner erlernen und das Verhalten bei Abwesenheit zu simulieren. Kommt es dennoch zu einem Einbruch oder einem Einbruchsversuch kann das System selbstständig reagieren und zum Beispiel die Fenster und Türen schliessen, die Fensterläden herunterlassen, die Polizei und die Bewohner alarmieren und allenfalls Videos und Fotos anfertigen.

Im Industriebereich kann das IoT zum Beispiel zur Überwachung, Kontrolle und Steuerung von Produktionsanalgen oder Fertigungsprozessen eingesetzt werden.

Durch das IoT ergeben sich viele Möglichkeiten und Chancen. Es ist jedoch nicht ausser acht zu lassen, dass auch neue Risiken und Gefahren entstehen. So müssen die entstehenden Netzwerke gegen Angriffe und Manipulationan abgesichert und die transportierten Daten vor fremden Augen geschützt werden. Kann ein Heimautomationssystem oder ein Überwachungssystem von aussen manipuliert werden, kann dies schwerwiegende folgen haben. Unter Umständen kann der Angreifer die Wohnungstüre öffnen und anschliessend so umprogrammieren, dass sie der Bewohner nicht mehr öffnen kann.

2.2 Domain Specific Software Engineering

Domain Specific Software Engineering, beziehungsweise das domänenspezifische entwickeln von Software, beschreibt den Entwicklungsprozess in einem bestimmten Kontext, beziehungsweise Problembereich. Der Entwicklungsprozess für medizinische Geräten und Software unterscheidet sich an vielen Stellen vom Entwicklungsprozess einer Unternehmung, welche Web-Seiten designt. Auch unterscheidet sich die Entwicklung von Web-Seiten von der Entwicklung von Mobile-Apps oder Fat-Client-Anwendungen.

2.3 Domain Specific Languages

Eine Domain Specific Language (DSL) bezeichnet eine Programmier-, Modellierungs- oder Metasprache, welche für eine spezifische Problemdomäne entworfen und entwickelt wurde. Eine DSL addressiert dabei spezifische Schwächen und Problemstellungen der angesprochenen Domäne.

Eine gute DSL zeichnet sich durch ihre Einfachheit aus. Die Sprache sollte sich so nah als möglich am Problembereich befinden und auch die entsprechenden Ausdrücke und Begriffe verwenden. Dadurch wird die Eintrittshürde zur Verwendung der DSL heruntergesetzt und der Einarbeitungsaufwand reduziert. Auch sollte die Semantik und Syntax so gewählt werden, das diese im Kontext Sinn ergibt und einfach lesbar und verständlich ist.

Es können zwei Arten von DSL's unterschieden werden. Eine externe, oder auch unabhängige, DSL ist so ausgelegt, dass diese nicht von einer bestimmten Sprache abhängig ist. Die Festlegung der Syntax und Grammatik liegt komplett in der Verantwortung und Entscheidungsfreiheit des Autors. Für die Umsetzung kann eine beliebige Programmiersprache verwendet werden.

Im Gegensatz dazu basiert die interne oder eingebettete DSL auf einer spezifischen Sprache (Wirtssprache). Diese Sprache gibt dabei die Einschränkungen für die zu implementierende DSL vor. Im Gegenzug muss sich der Autor nicht mehr um die Grammatik, Parser und Tools kümmern, da dies bereits von der Sprache zur Verfügung gestellt wird.

Externe DSL's sind flexibler, erfordern aber einen viel höheren Implementierungsaufwands als eine interne DSL

Beispiele

- Externe DSL
  - SQL
  - Reguläre Ausdrücke
  - CSS
  - Sass
- Interne DSL
  - Rake (Ruby)

KAPITEL 3

Die Domäne „Internet of Things“

In diesem Kapitel wird die Domäne „Internet of Things“ näher beschrieben und die Eigen- und Besonderheiten im Bereich Software Engineering aufgezeigt.

3.1 Ziel

Das primäre Ziel des IoT ist die Vernetzung von Dingen mit der realen Welt und mit anderen Dingen. Die Dinge sollen intelligent miteinander kommunizieren können um so einen Mehrwert zu schaffen.

3.2 Anwendungsbereich

Das IoT könnte grundsätzlich überall eingesetzt werden. Sei es in der Landwirtschaft, in der Pflege, in der Medizin oder in der Industrie. Das IoT ist mehr ein Konzept, beziehungsweise eine Architektur, als ein konkretes Produkt, Dadurch lässt es sich auf die jeweiligen Bedürfnisse der verschiedenen Unternehmensbereiche individuell anpassen. Neben der Anwendung im geschäftlichen Umfeld, findet das IoT auch im privaten Umfeld Anwendung. Das primäre Ziel ist dabei die Heimautomation, das heisst die intelligente Vernetzung verschiedener Geräte und Gegenstände im Haushalt. Nachfolgend einige Beispiele für konkrete Anwendungen des IoT

- Smart City
- ÖL-Pipeline
- Pflege
- Medizin
- Energie
- Verteidigung

- Kommunikation
- Industrie

Auch im Bereich Big Data stellt IoT ein weiterer Meilenstein dar. Durch die Vernetzung von Millionen von Geräten können riesige Datenmengen gesammelt, analysiert und ausgewertet werden.

3.3 Architektur & Aufbau

3.3.1 Das „Thing“

Die Definition eines „Thing“, beziehungsweise eines „Dinges“, ist nicht ganz einfach. Grundsätzlich handelt es sich um ein „Embedded System“ welches Informationen über ein Netzwerk versendet und empfängt. Ein „Embedded System“ basiert auf einem Mikrocontroller, verfügt relativ gesehen nur über einen geringe Ressourcen (Prozessor und Arbeitsspeicher) und ist in der Regel auf Energieeffizienz ausgelegt. Das „Embedded System“ verfügt über einen Kommunikationsstack mit 1 bis n Kommunikationsprotokollen welcher entweder direkt oder über ein Gateway mit einem Netzwerk kommuniziert. Die gesammelten Daten werden entweder auf dem Gerät oder auf einem Gateway gefiltert.

Solche „Embedded Systems“ sind bereits heute überall gegenwärtig. Sei es in Autos, Wearables, Kühlschränken, Kaffeemschinen oder Smartphones.

3.3.2 Gateway

Filtern, managedn Daten, Verbindungshub

Gemäss Intel WhitePaper: 85% der Geräte nicht für Internet-Kommunikation ausgelegt (Requirements Stack) Intermediär Netzwerk und Legacy Things

Netzwerk und Cloud: Datanalyse der Rohdaten Lokales Netzwerk Back-End Services (Enterprise Data System, PC, mobile Device)

3.4 Unterschied zum Konzept des klassischen Internet

Im Gegensatz zum Paradigmenwechsel: Früher Server stellen Daten zur Verfügung. Das Konzept verändert sich auch dahingehend, dass nicht mehr nur Server miteinander kommunizieren, sondern auch die Endgeräte.

3.5 Effekte & Auswirkungen

Durch IoT kann Mehrwert auf verschiedenen Ebenen geschaffen werden. Zum Beispiel in Industriebetrieben können Produkte schneller und besser entwickelt und produziert werden, was sich am Ende auf die Kosten auswirken wird.

-Besseres Shopping vergnügen -Optimierung Energienutzung / -konsum

3.6 Subdomänen

3.6.1 Wireless Sensor Network

Wireless Sensor Network (WSN) sind Netzwerke von verteilten Sensoren, welche gewisse physikalische Zustände oder Zustände in der Umgebung und Umwelt überwachen. Ein WSN-Node ist ein billig produzierbares Gerät, welches nur sehr wenig Strom benötigt. Idealerweise wird dieses über eine Batterie oder eine autonome Energiequelle (zum Beispiel ein Solar-Panel) betrieben. Ein WSN-Node besitzt nur eine einzelne Funktion.

Ein WSN-Edge-Nodes verbindet mehrere WSN mit einem Netzwerk, Gateway

Vorwiegend industrie

image: http://micrium.c

WSN-Technologien: Wi-Fi (hohe Verbreitung, hoher Stromverbrauch), Low-Power-Solutions (Low-Power and efficient radios, energy harvesting, mesh networking, long term operation, new protocols and data formats), IEEE 802.15.4: Radio Standard, base low power system, power reduction, 6LoWPAN: As small messages as possible, IPv6 over Low power Wireless Personal Area Networks, Kapselung und Kompressionsmechnismen für kürzere Übermittlungszeiten

3.6.2 Kommerzielle und Heimnutzung

Bluetooth, Ethernet (wired or not wired), nur lokale services

– Anforderungen allgemein: -Geschwindigkeit, Skalierung, Fähigkeiten -Datensicherheit und Datenschutz, Management von Geräten, Data analytics -Grosses Ökosystem: Regulatorische / Rechtliche Schwierigkeiten -Unterschiedlich je Subdomäne: http://micrium.com/iot/iot-rtos/

3.7 Anforderungen an Software

Stark von Einsatzgebiet abhängig (Kühlschrank vs. Low Energy ohne Stromversorgung)

3.8 Anforderungen an die Softwarearchitektur

3.9 Anforderungen an die Programmiersprache

—-

3.10 Anforderungen / Unterschiede zur „Standard-Domäne“

In diesem Kapitel werden für die einzelnen Abschnitte des Software Engineering Prozesses die Anforderungen, beziehungsweise die Unterschiede, zum Software Engeineering Prozess in der „Standard-Domäne“ aufgezeigt. Mit „Standard-Domäne“ wird in dieser Arbeit Software Engineering im Bereich von Enterprise-Anwendungen bezeichnet.

http://ercim-news.ercim.eu/en98/special/internet-of-things-a-challenge-for-software-engineering
http://link.springer.com/chapter/10.1007

Das der Begriff „Internet of Things“ eigentlich nur ein Oberbegriff ist.... deckt dieser Begriff auch ein sehr groses Spektrum an verschiedensten Anwendungsmöglichkeiten ab.

3.10.1 Software Requirements

eindeutig, Komplexität schwierig, nachhaltig (schwieriger Update)

non-consumer-market: regulatorisches Schwierigkeiten, Remote Tracking, Kabellose Implantate, –> Requirements-Phase: Verfolgbarkeit und Prüfbarkeit, Geschickte Verisonierung, Reviews –> Sicherstellung Compliance und Erfolg Wetter, Physische Einflüsse, Durch das Gerät erzeugte Hitze, Lange Lebensdauer

SW Muss: Skalierbar für breite Palette an verschiedenen Gerätekategorien Modular sein –> nur Auswahl (RAM-Footprint) Verbunden (Daten rein/raus) Verlässlich:Zertifizierung für kritische Applikationen

3.10.2 Software Design

Schlank, Energieeffizient, einfach wartbar?

Berücksichtigung: Protokolle, Standards, Last-Anforderungen

Machine To Machine Connectivity, Wireless, API-Evolution in MInd

3.10.3 Software Construction

Je nach Anwendungsbereich: Hardwarenahe, Optimierung Für die Entwicklung von Anwendungen für IoT-Geräte können https://www.rti.com/company/careers/software-engineer.html grundsätzlich

Firmware für Spezifische Hardware, Adressierung Netzwerk- und Verbindungsprobleme, Security

3.10.4 Software Testing

Wenn bei Consumer: Update evtl. schwierig, Fehlfunktion schwerwiegend,

Test: Nachbildung physische Umgebung, komplexe Szenarien, Netzwerk-Anforderungen,

3.10.5 Software Maintenance

Schwierig,... Immer connected –> regular updates, wenn nicht regelmässig upgedatet –> Verlust kritischer Funktionalität, Continuous Delivery

Nicht überall möglich, Low-Bandwith, Hardware-Near Devices

3.10.6 Software Configuration Management

3.10.7 Software Engineering Management

3.10.8 Software Engineering Process

Defect Tracking, small scale projects, fast product turnaroudn,

3.10.9 Software Engineering Tools and Methods

3.10.10 Software Quality

Höhere Qualität notwendig

Layer: Transport-Layer: TCP zum Teil overkill für IOT-Device –> UDP UDP: beser geeignet für relatime-data, TCPS Acknowledgment and retransmission unnützer overhad für solche Anwendungen (Stück sprache nicht rechtzeitig übermittelt -> Retransmission sinnlos),

Klassische Ansätze nicht alle geeignet für IoT,

Service-based Application: composition and orchestration of Services

KAPITEL 4

Software Engineering in der Domäne „Internet of Things“

Evtl. Struktur nach kategorie: Connectivity, Management, Security, API, ...

Real-Time schwierig, trade off datenverlust, oft TCP als Datenbasis -> messung mit Quality of service besser

http://postscapes.com/internet-of-things-software-guide https://www.silabs.com/Supporthttp://www.datamati
source/35-open-source-tools-for-the-internet-of-things-1.html https://blog.profitbricks.com/top-49-tools-internet-of-things/ http://www.businesscloudnews.com/2015/05/14/samsung-announces-open-internet-of-things-platform/ https://www.rti.com/company/news/iot-connectivity-webinar.html http://www.iotsworldcongress.com/documents/4643185/4c3cc80c-03b0-41be-baf0-6a1a0fa7db07

SWE: https://hal.inria.fr/hal-01064075/document http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7
http://www.cio.com/article/2843814/developer/how-to-develop-applications-for-the-internet-of-things.html http://www.appdevelopersalliance.org/internet-of-things/

http://link.springer.com/chapter/10.1007

Protokolle / Standards für verschieden Phasen: Collect Data, Access, Process, Collect, Control / distribute

4.1 Programmiersprachen

C, C++, Zum Teil: Java –> OS benötigt, von daher C,C++ und Java Java benötigt nicht zu vernachlässigendd Ressourcen, z.B. Oracle Java ME Embedded: Memory: 130 KB - 700 KB, ROM: 350 KB - 2000 KB, + Netzwerkstack, Kernel, nicht mehr ein embedded system, Java nur auf fähigeren, teurerern Systemen

4.1.1 Assembler

Eine Implementation in Assembler kann unter bestimmten Voraussetzungen die beste Lösung für ein bestimmtes Problem sein. In der Regel wird der Assembler-Ansatz gewählt, wenn das Programm so effizient und sparsam wie möglich ablaufen und mit so wenig Ressourcen als möglich auskommen soll.

4.1.2 C / C++

4.1.3 Java / .NET

Für die Implementation kann es durchaus sinnvoll sein eine Hochsprache, wie Java oder .NET C# einzusetzen.

4.2 Protokolle und Standards

Grafik: http://electronicdesign.com/embedded/understanding-protocols-behind-internet-things

Viele Implementationen: MQTT: Collect device data, send to server, D2S CoAP XBEE XMPP: Access device data, connect device to people, D2S DDS: Distribute Device Data, fast bus for integrating intelligent machines (D2D) HTTP AMQP (IOT-Client-Seitig?), Queuing system to connect servers to servers (S2S)

https://www.sparkfun.com/news/1705 http://micrium.com/iot/internet-protocols/

4.2.1 MQTT

Message Queue Telemetry Transport, Device Data Collection, Haupfaufgabe: Fernmessung, Remote Monitoring, Datensammeln und an Infrastruktur ausliefern, Anwendungszweck: Grosse Netzwerke von kleinen Geräte, welche überwacht und kontrolliert werden müssen. Basis: TCP (kein Datenverlust)

Publish / Subscribe, central Server, Subscribe to Topics, MQTT-Broker, Notify all connected devices, Hub and Spoke System: Geräte –> Data Verbindung zu Server, System ist designt, um daten an enterprise technologien weiter zu geben (z.B. ESB)

Kein Protokoll für D2D, nicht mehrere Empfänger der Daten, wenige Kontrollmöglichkeiten, muss nicht schnell sein –> kein rela time (sekunden)

Ziel: Conserve Power and Management Bsp: ÖL Pipeline, Energieverbrauch Überwachung, Licht-Kontrolle

Lightweight packet structure, dokumentation auf einer Seite

Wann: Thing of it as a Collection? Wenig Device2Device Kommunikation? Sehr viele Geräte?, Sehr kleine Geräte?

4.2.2 XMPP

Ursprünglich: Jabber, entwickelt für Instant messaging Extensible Messaging and Presence Protocol

Text-Kommunikation zwischen Punkten

XML, über TCP (oder HTTP over TcP?)

Stärke: Adressierungs-Schema: name@domain.com, Security, Skalierbarkeit –> Ideal für Consumer-Oriented applications

Einfacher Weg Gerät zu adressieren, nicht schnell –> Polling oder Check for Updates on demand,

Einsatz: z.B. Heimautoation –> Thermostat mit Web verbinden

Wann: Use the word „my“?, Wenige Verbindungspunkte in grossem Netz? Speed / CPU nicht wichtig? Immer Verbunden?

4.2.3 DDS

Data Distribution Service, Geräte welche direkt Geräte-Daten verwenden, Verteilung zu anderen Geräten, Interaktion mit Infrastruktur unterstützt, Daten-zentrierter Middleware-Standard, Wurzeln: High-Performance Verteidigung, Industrie, Embeddedd Applications, effektiv: millionen von nachrichten pro sekunde zu mehreren gleichzeitigen empfängern

Publish, Subscribe Architecture

Unterschied: Daten an Infrastruktur oder an anderes Gerät, Geräte sind schnell, mit vielen Geräten kommunizieren, TCP Point to Point to restriktiv, DDS: Detailierte qualitiy of service conrol, multicast, konfigurierbare verfügbarkeit, Redundanz

Filter und Selektion von Daten, bzw. Bestimmung was wohin geht

Direct Device-to-device bus, relational data model, ähnlich Datenbank,

Z.B.: Militär, Windparks, Asset-Tracking, Fahrzeug Test und Sicherheit

Wann: Chaos / Desaster wenn für 5min offline?, Performance sehr wichtig (ms / us) / mehr als 100 Anwendungen / mehr als 10’000 Datenwerte, code mehr als 3 Jahre entwickelt?

4.2.4 AMQP

Advanced Message Queuing Protocol, ab und zu: IOT-Protokoll, transaktionsbasierte Nachrichen zwischen Servern, message-centric-middleware, proces thousands of reliable queued transactions, Fokus: kein Nachrichtenverlust, Publishers –> Exchanges, queues to subscribers: TCP, acknowledge acceptance of message, optional transaction mode with formal multiphase commit sequence

Hauptsächlich: Business messaging, device - back-office data centers

IOT: appropriate for control plane oder server-basierte analyse funktionen

Wann: Verteilung von Arbeit / nicht information?, Nur a zu B senden? Speed / CPU nicht wichtig? Nichts darf verloren gehen?

4.2.5 HTTP

Basis für Client-Server-Model im Web, Sichere Methode: auf Device nur Client implementiersen

4.2.6 WebSocket

Protokoll, Full-Duplex Communication über einzelne TCP-Verbindung zwischen Client und Server, Zeilt HTML5 Spec

4.2.7 CoAP

Web-Protokolle z.T. zu schwer für IOT-Geräte, Constrained Application Protocol for low-power and constrained networks, RESTful protocol, semantisch an HTTP ausgerichtet, 1:1 Mapping von / zu HTTP

UDP, + einige nachgebildete TCP-Funktionen, Request / Responses: Asynchron via CoAP Messages, Header, Method, Status Codes binary encoded, Reduktion Overhead, Caching abhängig von Response-Code (HTTP: Request Method)

Comparison of Protocol-Stacks: http://micrium.com/iot/internet-protocols/

Leichtgewichtige Alternative zu HTTP, CoAP Packets: all around bitmapping

4.2.8 Thread

Protokolle: Thread, Netzwerk-Protokoll, Fokus: Security, Low Energy, notwendiger Chip, schon in vielen Geräten vorhanden, gestützt auf 6LoWPAN, IPv6 over Low power Wireless Personal Area Network,

4.2.9 AllJoyn

Qualcomm entwickelt, anschliessend: Linux Foundation, AllSeen Alliance (Cisco, Microsoft, LG, HTC, ...) Verbindung, Wartung Geräte in WLAN-Netzwerk, Kontrolle, Benachrichtungs Service,

4.3 Frameworks

In diesem Kapitel werden einige Frameworks vorgestellt, welche ....

Frameworks –> mobile / web –> Vereinfachung Entwiclung, Abstraktion Implementations-Details, apache ISIS? http://iot.eclipse.org/java/open-iot-stack-for-java.html

4.3.1 Kommunikation zwischen IoT-Geräten

AllSeen IOTIVITY Z-Wave

6LoWPAN, ANT, Bluetooth, DASH7, ISA100, Wireless HART, Wireless M-Bus, Z-Wave, Zigbee / Zigbee IP, EnOcean, EtherCAT, Modbus, Profinet, HomePlug, HomeGrid

Voraussetzung Zugriff Internet: IPv6 Pflicht, Non-IP Geräte; Gateway

Middleware / Platforms / OS: http://postscapes.com/internet-of-things-software-guide

Weitere-Protokolle: http://postscapes.com/internet-of-things-protocols Tools: http://www.datamation.com/open-source/35-open-source-tools-for-the-internet-of-things-1.html

4.4 middleware

Arten: Point-To-Point, Client/Server, Publish/Subscribe, Queuing, Data-Centric

KAPITEL 5

Case-Study „iotivity

5.1 Aufsetzen der Entwicklungsumgebung

5.2 Entwicklung

5.3 Deployment und Betrieb

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der noch zu erledigenden Punkte