technische universität
dortmund

LS 12 /
TU Dortmund

Ein on-premise Task Automation Service für intelligente Wohnumgebungen basierend auf dem Eclipse IoT-Stack

Konstantin Tkachuk

DiDo-Vortrag

SS 2016

Gliederung

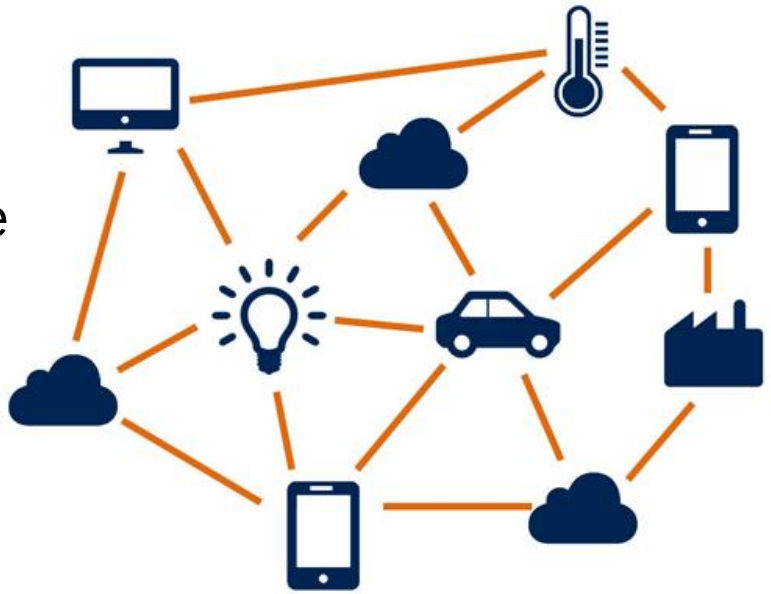
- Inhaltverzeichnis
- Grundlagen
  - Internet of Things
  - Task Automation Services
  - Smart Home
  - Eclipse SmartHome
- Ziele
- Zeitplan

Hype cycle


expectations
Advanced Analytics With Self-Service Delivery
Autonomous Vehicles
Internet of Things
Speech-to-Speech Translation
Machine Learning
Wearables
Cryptocurrencies
Consumer 3D Printing
Natural-Language Question Answering
Smart Advisors
Micro Data Centers
Digital Dexterity
Software-Defined Security
Neurobusiness
Citizen Data Science
Biochips
IoT Platform
Connected Home
Affective Computing
Smart Robots
3D Bioprinting Systems for Organ Transplant
Volumetric Displays
Human Augmentation
Brain-Computer Interface
Quantum Computing
Hybrid Cloud Computing
Augmented Reality
Enterprise 3D Printing
Gesture Control
Virtual Reality
Autonomous Field Vehicles
Cryptocurrency Exchange
Bioacoustic Sensing
People-Literate Technology
Digital Security
Virtual Personal Assistants
Smart Dust
As of July 2015
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time
Plateau will be reached in:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau

Internet der Dinge

- Steigende Anzahl von intelligenten Geräten
- Große Menge an Daten
- Unterschiedliche Unterbereiche
  - Smart Home
  - Task Automation Services
  - …

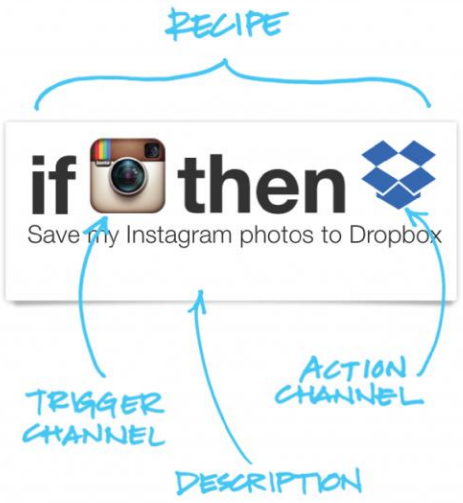


Task Automation Services (TAS)

- Automatisierung historisch eines der zentralen Ziele der Menschheit – von Mühlen bis Kaffeeautomaten
- Heute: Automatisierung von Serviceinteraktionen
- Task Automation Service:
  - Dienst, der die Orchestration von anderen Diensten ermöglicht
- Typische Merkmale
  - Device/Web Channels
  - Event-Condition-Action Regeln
  - Visuelle Regel-Editoren
  - Device-/Webdriven Rule Execution Profiles

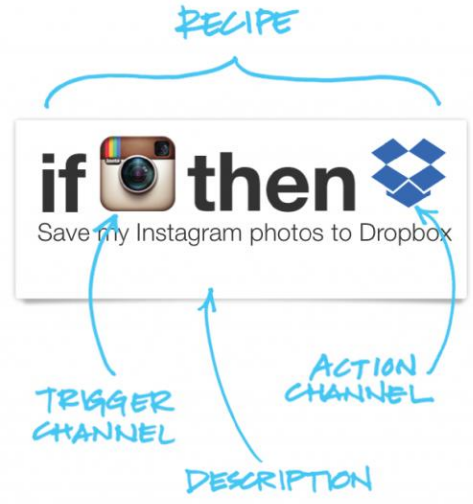
Task Automation Services: IFTTT

- If this than that:
  - Erstellen von Event-Condition-Action Regeln zum Steuern von Geräten
  - Benötigt Zugriffsdaten auf entsprechende Services
  - Bietet intuitiven visuellen Regel-Editor



Task Automation Services: IFTTT

- Vorteile:
  - Einfach, cool, bietet Automatisierung
  - In der Cloud: von überall erreichbar und konfigurierbar, minimale Anforderungen an eigenes Netzwerk
- Nachteile:
  - Keine komplexen Regeln
  - In der Cloud: Keine direkte Gerätesteuerung
  - Datenschutz



Task Automation Services – Überblick

		Web						Smartphone				Home	
		Ifttt	Zapier	CloudWork	Elastic.io	ItDuzzit	Wappwolf	On{x}	Tasker	Atooma	Automatelt	WigWag	Webee
Channels	Web channel support	✓	✓	✓	✓	✓	Few	Few	✓	✓	✓	✓	✓
	Device channel support	Few	×	×	×	×	×	✓	✓	✓	✓	✓	✓
	Smartphone resources as channels	✓	×	×	×	×	×	✓	✓	✓	✓	×	×
	Public channels support	✓	×	×	✓	✓	×	✓	×	×	×	×	×
	Pipe channel support	×	×	×	✓	×	Few	Few	×	×	×	×	×
	Group channel support	×	×	×	×	×	×	×	×	×	×	Few	×
Rules	Device channel discovery	×	×	×	×	×	×	×	×	×	×	✓	Few
	Multi-event rules	×	×	×	×	×	×	✓	✓	×	×	✓	×
	Multi-action rules	×	×	×	✓	×	×	✓	✓	×	✓	✓	✓
	Chain rules	×	×	×	✓	×	Few	Few	×	×	×	×	×
	Group rules	×	×	×	×	×	×	×	×	×	×	Few	×
	Collision handling	×	×	×	×	×	×	×	×	×	×	×	×
	Predefined common rules	×	×	✓	×	✓	✓	×	×	×	✓	✓	✓
	Rule execution profile	WD	WD	WD	WD	WD	WD	DD	DD	DD	DD	DD	DD
TAS	Visual rule editor	✓	✓	×	✓	✓	✓	×	✓	✓	✓	✓	✓
	Provides API	×	✓	×	✓	×	×	×	×	×	×	×	×
	Programming language	×	×	×	✓	×	×	✓	Few	×	✓	✓	✓

* ✓ = supported; × = not supported; Few = few support; WD = Web-driven execution profile; and DD = device-driven execution profile.

3 unterschiedliche Typen

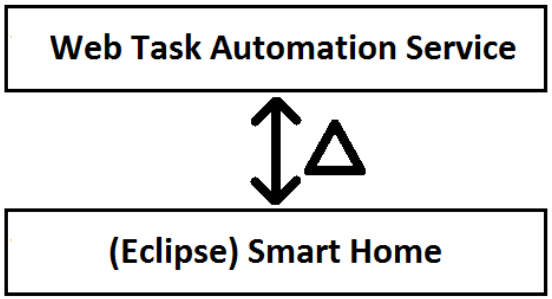
1. Web TAS
   - Automatisierung von Web-Serviceinteraktionen in der Cloud
2. TAS auf Smartphone
   - Automatisierung von App-Interaktionen
3. (Smart-)Home
   - Automatisierung von Geräten

Smart Home

- Teil von IoT
- Konzentriert sich auf die direkte Steuerung von Geräten/Services im Kontext eines Hauses
- Proprietär
  - Unterstützung von begrenzter Anzahl von Geräten
- Open Source
  - Fokus auf maximale Offenheit

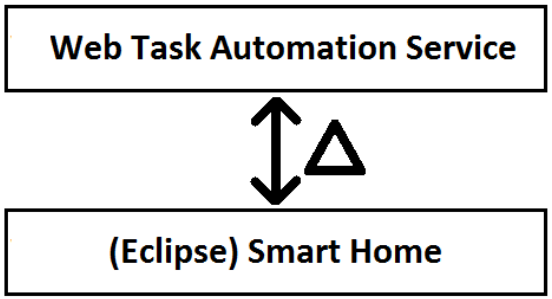
Vergleich SmartHome und TAS

- Gemeinsamkeiten von Webbasierten TAS und SmartHome
  - Automatisierung von Geräten und Services
  - Event-Condition-Action Regeln kommen zum Einsatz
  - Visuelle Regeleditoren
- Unterschiede
  - Fokus der Automatisierung
  - On-premise vs. Cloud



Vergleich SmartHome und TAS

- Gemeinsamkeiten von Webbasierten TAS und SmartHome
  - Automatisierung von Geräten und Services
  - Event-Condition-Action Regeln kommen zum Einsatz
  - Visuelle Regeleditoren
- Unterschiede
  - Fokus der Automatisierung
  - On-premise vs. Cloud



Bis dato sind webbasierte TAS und SmartHome völlig unabhängig. Nun sollen sie kombiniert werden.

Vorteile durch Kombination

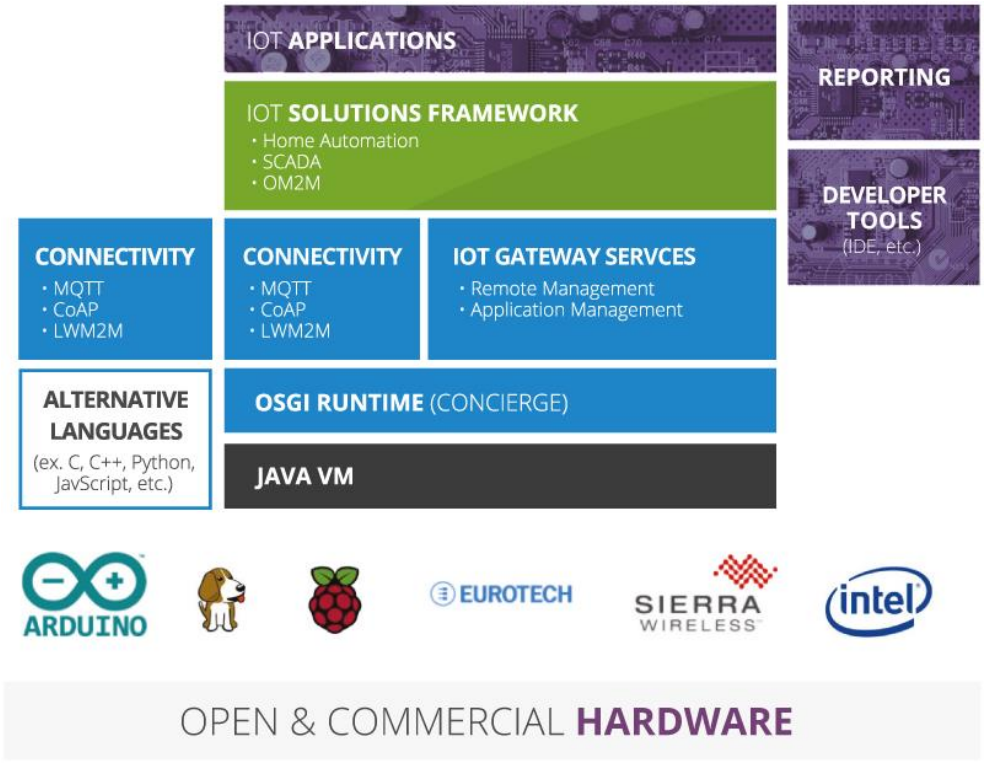
- Zugriffsdaten werden lokal gelagert
  → Datenschutz
- Geräte werden lokal gesteuert
  → Funktionalität im Haus bleibt bei Verbindungsabbruch erhalten
  → Reaktionszeiten von Geräten verbessert?
- Komplexere Regeln/Szenarien ermöglichen
  → Wiederverwendung von existierenden Funktionalitäten

Nachteile:

- Erfordert nun eine HomeBase (z.B. Raspberri Pi)

Eclipse Open IoT Stack

- Eclipse SmartHome
- Framework zur Entwicklung von eigenen SmartHome Lösungen
- Unterstützt Entwickler bei:
  - Discovery-Services
  - Bindings
  - Rule Engine
  - GUI
  - …
- Wird in bekannten Lösungen eingesetzt, z.B. openHAB, Qivicon



Eclipse SmartHome – Domain

Channel

- Informationstyp, der vom Thing kommuniziert wird

Thing

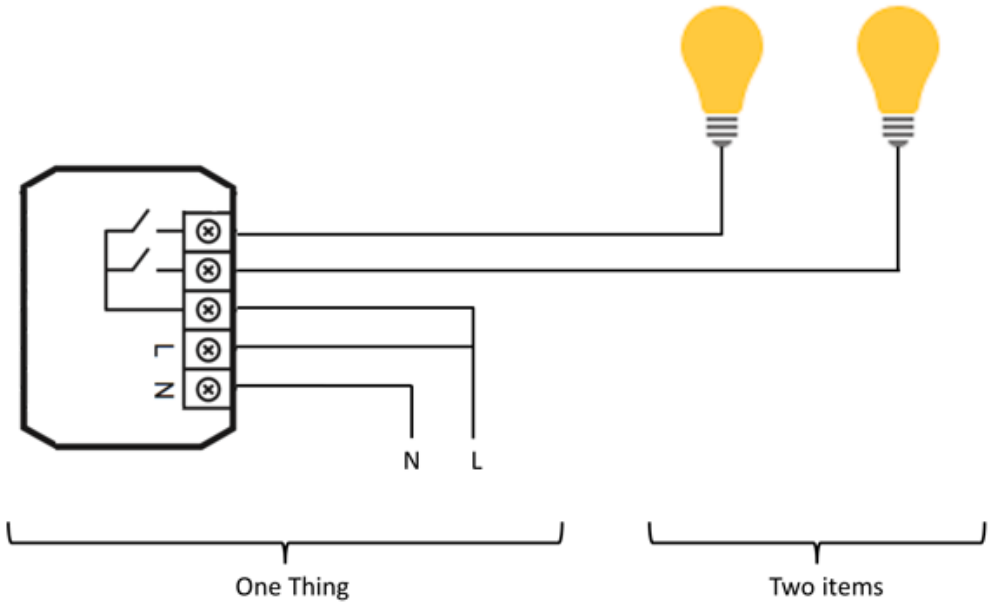
- Ein Gerät, das zum System hinzugefügt werden kann. Ein Thing kann mehrere Channels unterstützen

Link

- Verbindet Things mit Items 1-to-1

Item

- Repräsentieren konkrete Funktionalitäten von Dingen. Haben einen Zustand und können Commands empfangen



Eclipse SmartHome – Items

Itemname	Description	Command Types
Color	Color information (RGB)	OnOff, IncreaseDecrease, Percent, HSB
Contact	Item storing status of e.g. door/window contacts	OpenClose
DateTime	Stores date and time	
Dimmer	Item carrying a percentage value for dimmers	OnOff, IncreaseDecrease, Percent
Group	Item to nest other items / collect them in groups	-
Number	Stores values in number format	Decimal
Player	Allows to control players (e.g. audio players)	PlayPause, NextPrevious, RewindFastforward
Rollershutter	Typically used for blinds	UpDown, StopMove, Percent
String	Stores texts	String
Switch	Typically used for lights (on/off)	OnOff

Passende Items für Web Services werden benötigt.

Eclipse SmartHome – Bindings

ThingTypes

- XML: Id, Channels, Channel-Details (z.B. unterstützte Item-Typen, Kategorien), Parameter

ThingHandlerFactory

- Erstellt zugehörige ThingHandler-Instanzen

ThingHandler

- Bearbeitet Commands, aktualisiert Zustand

Bridge (opt.)

- Repräsentieren Gateways, über die auf weitere Things zugegriffen wird

Eclipse SmartHome – Rules

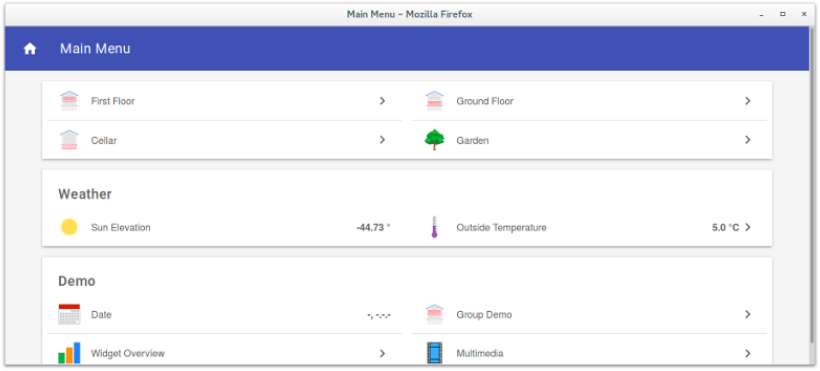
- Triggers beschreiben, wann eine Regel ausgeführt wird, z.B. Event
- Conditions funktionieren als Filter
- Actions sind Sequenz von Befehlen, die ausgeführt werden

```
{
   "uid":"sample.rule1",
   "name":"SampleRule",
   "tags":[ ... ],
   "description":"Sample Rule definition.",
   "triggers":[ ...],
   "conditions":[ ... ],
   "actions":[ ... ]
}
```

- Gegeben sind Interfaces, z.B. *GenericEventTrigger, GenericCompareCondition* → Konkrete Implementierungen benötigt
- Rule Templates sind zu definieren

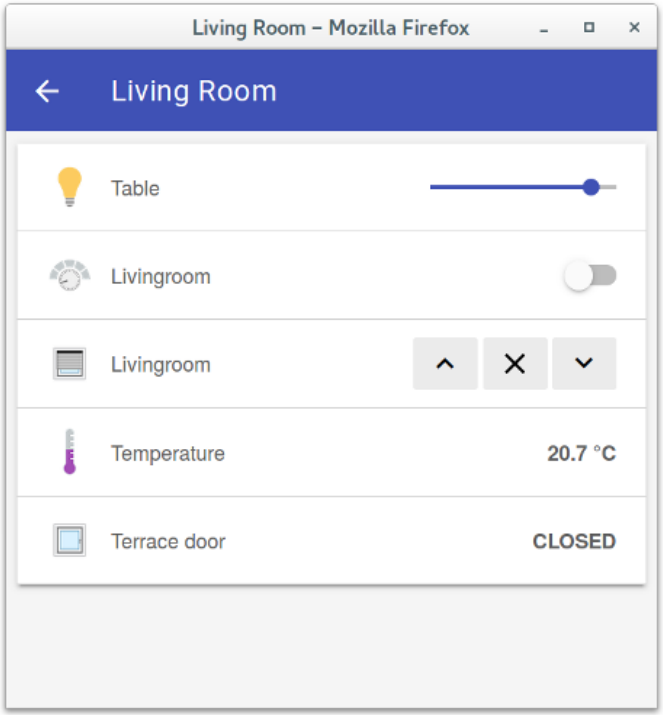
Eclipse SmartHome – GUI

- Rudimentäre GUI für SmartHome Zwecke existiert bereits



Es wird benötigt:

- Idealerweise: Visueller Regeleditor
- Minimum: JSON-Regeleditor
- Darstellung existierender Regeln



Ziele der Arbeit

- SmartHome mit webbasierten TAS vereinenen
- Demonstrator entwickeln, der
  - On-premise auf einem Raspberri Pi 3 läuft
  - Geräte über lokales WLAN bzw. Bluetooth direkt steuern kann
  - SmartHome um Funktionalitäten von webbasierten TAS erweitert
  - Für eine Reihe von Web-Services Bindings enthält (mindestens 1 FileSharing, 1 Social Media und 1 Wetterdienst, sowie weitere)
  - Funktionalität in Demo-Regeln demonstriert
  - Eine Möglichkeit bietet neue Regeln zur Laufzeit hinzuzufügen
- Evaluation des Demonstrators
  - Aspekte wie Reaktionszeiten und Netzwerklast untersuchen
  - Mit IFTTT Vergleichen

Optionale Ziele

Falls Zeit übrig bleibt:

- Implementierung weiterer Bindings
- Implementierung Offloading in die Cloud (z.B. Amazon VM)
- Detailliertere Ausarbeitung der GUI
- Konfliktindentifikation und –resolution in Szenarien
- …

Zeitplan – 26 Wochen

- xx.08.2016 Anmeldung
- 3 Wochen Architekturplanung/Implementierung generischer Dienste
- 5 Wochen Implementierung Bindings für 5 Services
- 3 Wochen Integration/Erweiterung/Configuration Rule Engine
- 3 Wochen Entwicklung GUI
- 3 Wochen Evaluation
- 6 Wochen Schreiben
- 3 Wochen Planungsspielraum für optionale Ziele, Notfälle, etc.