IT University of Copenhagen

Software Development BSc

Bachelor project

ReRide IoT platform

Author:
Anders Edelbo Lillie

Supervisor:
Tomas Sokoler

March 20, 2017

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

1 Introduction

This paper investigates and reports the development of a Cloud based Internet of Things (IoT) system extending self-monitoring for the ReRide project. The Bachelor project is carried out on the sixth semester of the "Software Development" BSc line on the IT-University of Copenhagen. The motivation for the project is an individual curiosity for IoT and its growing popularity and influence in IT and the everyday life. Exploring IoT can be done in many ways, but investigating it through an actual case provides a deeper relevance, thus making it a more attractive project.

The following problem definition describes the scope of the project and the theoretical background needed for performing this study.

1.1 Problem definition

How can a Cloud based IoT system extend self-monitoring as part of the ReRide project?

- What is "the Internet of Things"?
- What is a "Cloud based IoT system"?

1.2 Case

The study is based on a specific case introduced by the supervisor, who himself is part of the project. This section describes the case in order to give an overview of the context for this Bachelor project.

The ReRide project[Bagalkot et al., 2016] investigates how *digital technology for self-monitoring can be designed to help integrate physical rehabilitation with everyday activities*. The project explores this by designing self-monitoring in the context of motorbike riding. By using a belt with embedded flex sensor, back posture can be perceived by a "microprocessor controlled mechanically moving display unit", which can be mounted on the bike allowing the rider to adjust the back posture through feedback from the unit. The focus is on human-computer interaction (HCI) and the notion of *embodied perception* in the design for rapid coupling between the rider's adjustments and the feedback provided.

The design experiment was carried out in Bangalore, India, with a small group of bike riders suffering from back injuries. By integrating the above mentioned components to the motorbike and the rider, the data collected from the flex sensor would enable a servo motor to move the display unit placed on the motorbike's speedometer based on the posture of the rider. The idea was that when a rider is sitting in a correct posture the speedometer would become visible, but would block certain parts of the speedometer if the rider would change posture to a wrong position. The experiment concluded that the notion

of rapid coupling mentioned above was promising, but the form of the display needed more work.

2 Solution

3 Background

3.1 Internet of Things

3.1.1 Description

The term "Internet of Things" was coined by Peter T. Lewis in September 1985 and covers the concept of connected physical objects enabled to exchange telemetry. The Internet, wireless communication and especially the Smart phone have made it possible to realize this, and is a growing and popular technology.

3.1.2 Concepts

Characteristics to the Internet of Things is discussed by Bruce Sterling in his book "Shaping Things"[Sterling, 2005], where he uses the word *Spime* for objects which can be tracked through space and time (Sp-ime). The term was coined by Sterling and is a neologism for a "futuristic object" with these properties. He describes Spimes as something that can be created through six technologies: small inexpensive remote control, location awareness, data mining, virtual construction, rapid prototyping, and effective recycling. His idea of a Spime is an interesting concept in regard to the Internet of Things, as the concept very much can be realized in today's technological world through the Internet of Things.

Another neologism in the IoT language, which resembles Sterling's idea of a way to virtually construct an object, is the device "shadow". This is a concept heavily used to represent an object in the Cloud, and is used in all the IoT platform discussed in section 3.2, but with different names. Having a virtual mirror of an object makes it possible to not depend solely on e.g. an object's Internet connection, as it can be synced with its shadow, when reestablished.

3.1.3 Cloud-based computing

What really makes IoT efficient and beneficial for a consumer is the property of a device to be always connected. This is realized through Cloud computing. Instead of relying on a storage device physically close to the object, having it connected to the Internet enables scalable real-time Cloud computing.

The main characteristics and forces of Cloud computing is the resource usage and on-demand self-service. This has also been labeled "autonomic computing", while the "Cloud computing" refers to a distributed system that is distinguished from other systems through interconnection. In autonomic computing human intervention is reduced in the deployment of services, and the influence of the

location of resources is diminishing compared to previous emerged distributed systems. In [Catlett et al., 2013, p. 4-5] the characteristics Cloud Computing are more formally described as follows:

1. *involves distributed computing resources and software services which instance number varies by adapting to unpredictable changes*

2. *performs a contextual behavior through methods of self-management, self-tuning, self-configuration, self-diagnosis, and self-healing*

3. *hides intrinsic complexity to operators and users, as implementing techniques for designing, building, deploying and managing computing systems with minimal human involvement and presents itself as a robust, fault tolerant and easy to manage and operate architecture and deployment*

From point 1 it translates to a system being able to auto-scale, which is a very important feature in a large business with even thousands of customers. It also involves load-balancing, scheduling, and adaptive resource provisioning, which is the hardware aspect of it. Point 2 stands for actual self-maintenance and the ability to detect and act according to the environment and pre-defined policies. Point 3 addresses the importance of reliability, together with the property of easy deployment and minimal human involvement. One approach to deal with these applications is a platform-as-a-service (PaaS). Informally its definition states that it is a *cloud computing model that delivers applications over the Internet* REF(http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS). What it is supposed to deliver is *a Cloud-based approach that provides enterprises with all the functionalities for developing, deploying, and administering services, without the burden of installing, configuring, and managing the underlying middleware, operating system, and hardware* [Catlett et al., 2013, p. 9]. Examples of these in regards to IoT is further investigated in section 3.2.

[Bessis and Dobre, 2014]

3.2 Cloud based IoT platform

This section will investigate some of the big platforms for IoT cloud-based computing in order to be able to tell what they actually offer and how/if they differ in their product.

3.2.1 Google Cloud Platform

Overview *Cloud Platform brings scale of infrastructure, networking, and a range of storage and analytics products you can use to make the most of device generated data.*[Google, 2017b] This is Google's own description of their IoT Cloud solution, and the references article is used for analysis in this section.

The platform can be divided into three basic components, the device, gateway, and cloud, each of which are characterized by the following: A *device* is

defined as hardware and software that directly interacts with the world, and are connected to a network and the Internet. A *gateway* lets devices *not* directly connected to the Internet reach a cloud service. The *Cloud* is the *endpoint* of device data, where it can be processed.

Several terms are used to describe what devices constitute, but the most important seems to be metadata and state information. Device metadata contains information such as ID and the date manufactured. The state information is basically the device status, such as "on"/"off". Furthermore devices must be associated with some security credentials for authentication. *Telemetry* is the term for device data gathering, and is described as the *eyes-and-ears data* the device collects in its environment. Once a device has been defined, telemetry can be sent to the Cloud Platform. As mentioned, a gateway can provide connection to the cloud, when e.g. protocol translation is needed. The Cloud Platform is the infrastructure for data management, which is depicted in figure 1.

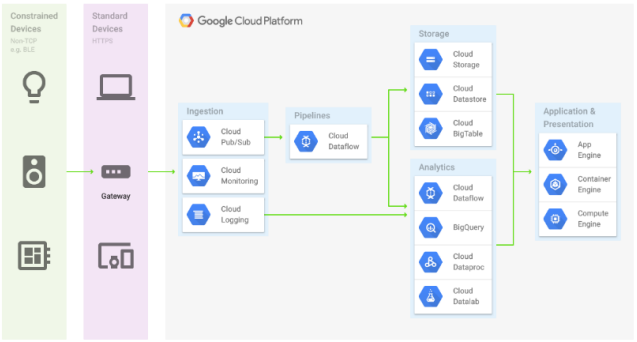

Figure 1: Diagram of the Google Cloud Platform IoT infrastructure

Services Google calls each of these components *services*, and are categorized by their functionality and role in the infrastructure. Communication between the services are called *pipelines*, which includes transforming, computing, combining, and moving data. The Cloud Platform also provides means for storing data in Google Firebase[Google, 2017a] and is especially used for continuously mirroring device states and exposing them to clients (See figure 2).

The most interesting services are the "rule processing" and "streaming analytics", which allows for programmers to write custom logic for handling data triggered by device events. What you can do is to essentially trigger alerts, filter data, and invoke other APIs. This property of quick data processing is described as *key* in the Cloud Platform.

Google Cloud Platform is not just for IoT, but provides a plethora of products. Some of these can be utilized through integration with the Cloud Platform and include different storage services like Cloud SQL, data warehouse for ana-

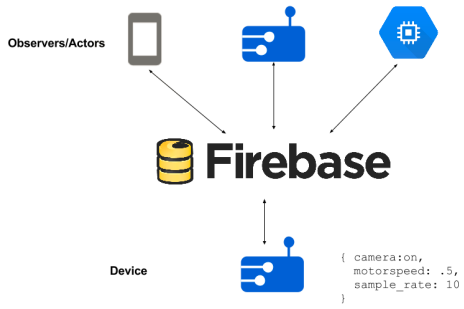

Figure 2: Abstract functionality of Google Firebase in the IoT Cloud Platform

lytics in BigQuery, data processing in Cloud Dataflow, as well as identity and access management in Cloud IAM.

Apart from the Cloud Platform, Google has created an Android-based IoT platform called "Android Things" (formerly known as "Brillo"), as well as a communication platform for IoT devices called "Weave". This has however not had much success and will not be investigated here.

3.2.2 Amazon Web Services

Overview *AWS IoT is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices.*[Amazon, 2017]. Aside from the huge and popular web shop Amazon hosts, they have developed a huge library of web services, called "Amazon Web Services" (AWS). Above citation is from their website and describes one of their products; The "AWS IoT Platform".

AWS IoT Platform constitutes of several components, labeled "AWS IoT Device SDK", "Device Gateway", "Authentication and Authorization", "Registry", "Device Shadows", and "Rules Engine". The infrastructure is depicted in figure 3.

The *AWS IoT Device SDK* provides software development kits to connect hardware devices or mobile applications, and supports C, JavaScript, and Arduino. Protocols like MQTT, HTTP, and WebSockets are supported for connectivity and data exchange. Furthermore Amazon has partnered up to offer a wider variety of SDKs, including Google's Android and Apple's iOS. The SDKs are meant for making integration with AWS easier for the programmer by encapsulating operations on the IoT Platform in library function calls.

The *Device Gateway* provides before mentioned protocols for communication between devices and the IoT Platform. It can be configured to work as a publication/subscription model for one-to-one and one-to-many communication, and it even scales automatically with *over a billion* devices connected.

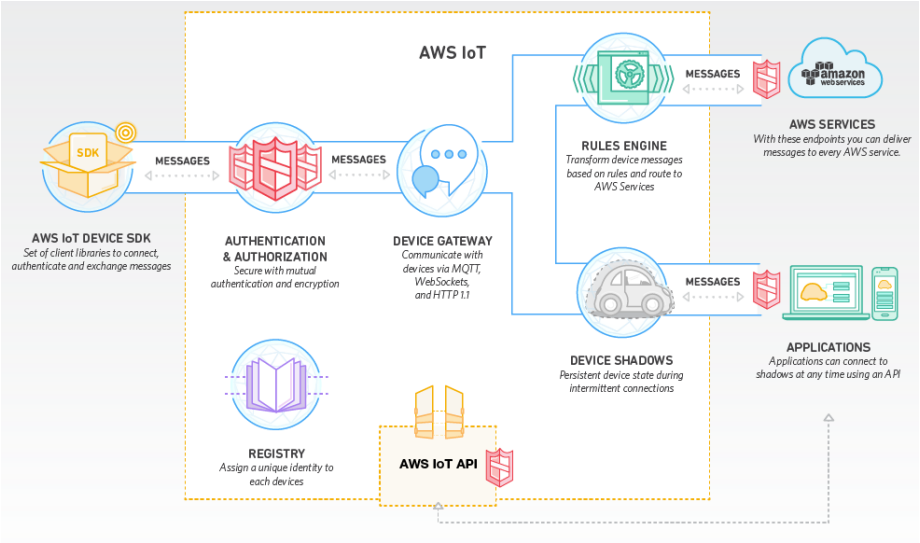

Figure 3: Components of AWS IoT Platform and the infrastructure

Security is taken seriously by Amazon, and this is what *Authentication and Authorization* takes care of. Data between device and IoT Platform is never exchanged without proven identity, by use of "SigV4" and X.509 certificates. It's possible to fully manage mapping of authentication policies for connected devices, as well as revoking access rights at any time. Another feature is to use Amazon Cognito, which basically takes care of authentication for mobile applications.

The *Registry* tracks device metadata by establishing unique identities for connected devices.

Device Shadows are persistent, virtual versions of connected devices and represent their sates. This allows for other devices to interact with other devices through a REST API. In other words, it is a convenient way of updates devices, as well as getting information about them.

The data processing happens in the *Rules Engine*, which evaluates messages passed to IoT Platform by user defined rules. The data can simply be transformed and passed to another device, or routed to other AWS cloud services for further processing. This includes several of the AWS services like AWS Lambda, S3, and DynamoDB.

Figure 4 is an example of the IoT Platform and illustrates how a light bulb can controlled through above mentioned services. A mobile application sends instructions to a controller, which communicates with the IoT Platform in the cloud. Two rules are defined to act upon triggered events. One rule routes the data to other AWS services for evaluation, and the other changes the light color of the bulb. In either case the device shadow is updated with a new state. If

the light bulb is turned off, requested device states are still saved in the device shadow.

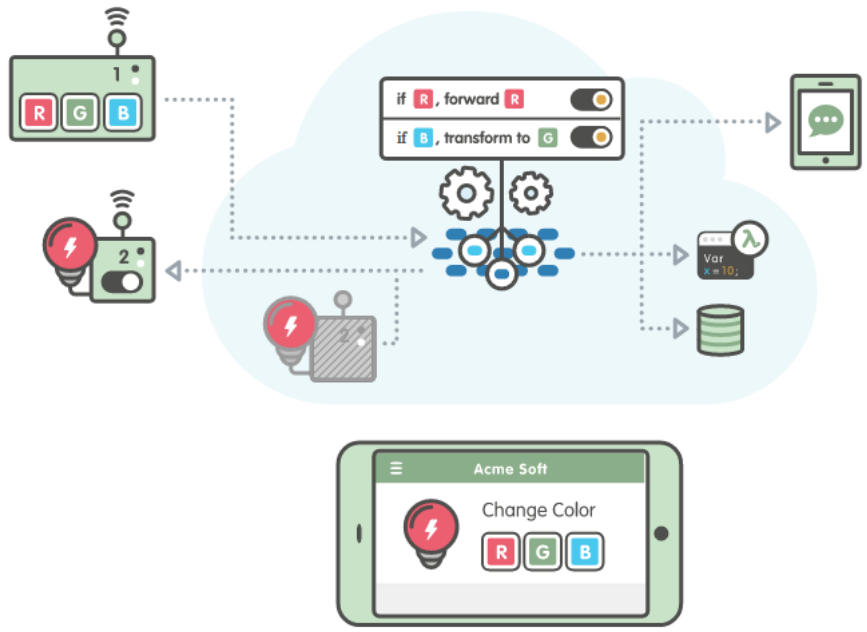

Figure 4: An AWS IoT Platform example illustrating the workings of the services.

Services As mentioned in the overview, the Rules Engine allows for inbound data to be routed to other services. One worth mentioned is AWS Lambda, which executes computations in the cloud. It is even possible to write code in e.g. C# and upload code snippets to the project. This service has the property of removing the need for a serer, as well as automatically scaling with the size of the workload. Other services like "DynamoDB" offers a NoSQL cloud database, and "Kinesis" for data analytics.

3.2.3 Microsoft Azure

Azure IoT Hub is a fully managed service that helps enable reliable and secure bi-directional communications between millions of devices and a solution back end. REF(https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide)

Microsoft is another big player in the field of IoT Cloud computing, and offers their own IoT platform. Much confusingly they offer an IoT *Suite*, as well as an IoT *Hub*. The difference is that the IoT Suite is *a collection of pre-configured solutions* REF(https://docs.microsoft.com/en-us/azure/iot-hub/iot-

hub-what-is-azure-iot). This means that they have actually made customizable solutions based on common IoT scenarios, such as remote monitoring and asset management. The IoT Suite is however also the platform including the IoT Hub, which is the most important service in the IoT infrastructure, as it takes care of bidirectional connection with devices and the cloud. Figure 5 illustrates the infrastructure of the platform from a device to the back end services.

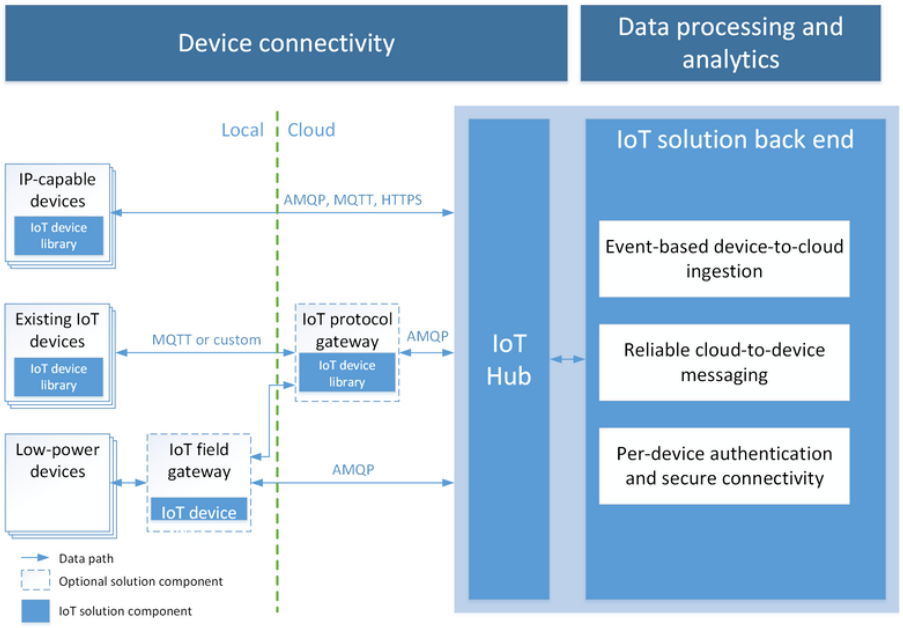

Figure 5: The Azure IoT Hub internal infrastructure.

The IoT Hub keeps records of so called "Device twins", which are JSON documents containing device state and metadata information, for all connected devices. An identity registry enables devices to authenticate securely to the IoT Hub, which allows for full control of connecting devices. Devices communicate with the cloud though a device SDK or *gateway*. The gateway is meant to be used when the offered SDKs do not support the device. Two types of gateways are offered, one being the *protocol* gateway and the other being the *field* gateway. The former is deployed in the cloud and performs protocol translation for MQTT, AMQP, and HTTP. The latter is deployed locally, i.e. physically close to the device(s), and enables also protocol translation, but also has the property of being able to actively manage devices, and the information mediated to the cloud.

Services Returning to the IoT Suite, a handful of services like Azure Stream Analytics, Storage, DocumentDB, Web Apps, and Power BI are available for

integration with the IoT Hub. Together they provide means for data analytics, storage, visualization, and integration with back-office systems. REF(https://docs.microsoft.com/da-dk/azure/iot-suite/iot-suite-overview). Telemetry can be routed to an Azure service by user-defined rules in the IoT Hub, and requires no code. As mentioned the IoT Suite contains preconfigured solutions, which are labeled "Remote monitoring" and "Predictive maintenance". These solutions are the way to start building an IoT platform, and are customizable and extensible.

3.2.4 IBM Watson IoT

Watson IoT Platform is designed to simplify cognitive IoT development so you can harness the full potential of the Internet of Things REF(https://www.ibm.com/internet-of-things/).

IBM has a lot of different domain specific products and solutions in regards to IoT. They offer different solutions based on the business application, e.g. "Asset management", "Facilities management", and "Product development" REF(https://www.ibm.com/internet-of-things/iot-solutions/). They are all hosted on the IBM Bluemix cloud platform, which essentially is a big collection of offered products and services for cloud computing. One of them is the Watson IoT Platform, which provides a "powerful application access to IoT devices and data to help you rapidly create analytics applications, visualization dashboards, and mobile IoT apps" REF(https://console.ng.bluemix.net/docs/starters/IoT/iot500.html). The infrastructure of the platform is depicted in figure 6.

The Watson IoT Platform is described through six different concepts: organizations, devices, gateways, applications, events, and commands REF(https://console.ng.bluemix.net/docs/servi
An organization is a way of separating systems and ensure data security through a unique identifier. This way, devices and applications are logically bound to a single organization and communication with other organizations can only happen with an explicit application for this purpose. Devices are divided into two classes; *managed* and *unmanaged* devices. The former contains a device management agent, which allows interactions with the Watson IoT Platform Device Management service. Such devices can perform operations which unmanaged devices cannot. This includes device management request and device management operations such as updates, downloads and reboots. Devices that cannot connect directly to the Watson IoT Platform can do so through a gateway. A gateway is both a device and an application that serves as an access point for other devices. Watson IoT Platform uses the HTTP and MQTT protocols for all communication. All devices and gateways must be registered in the Watson IoT Platform before they can connect to a service. Device manipulation and data interaction is controlled through applications, which do so with an API key and a unique application ID, all tied to a specific organization. Any application can subscribe to an event, which is triggered by a device and published to the Watson IoT Platform. Conversely, applications can communicate with devices through commands. The overall architecture of the Watson IoT is depicted in figure 7, which resembles figure 6, but is more elaborate

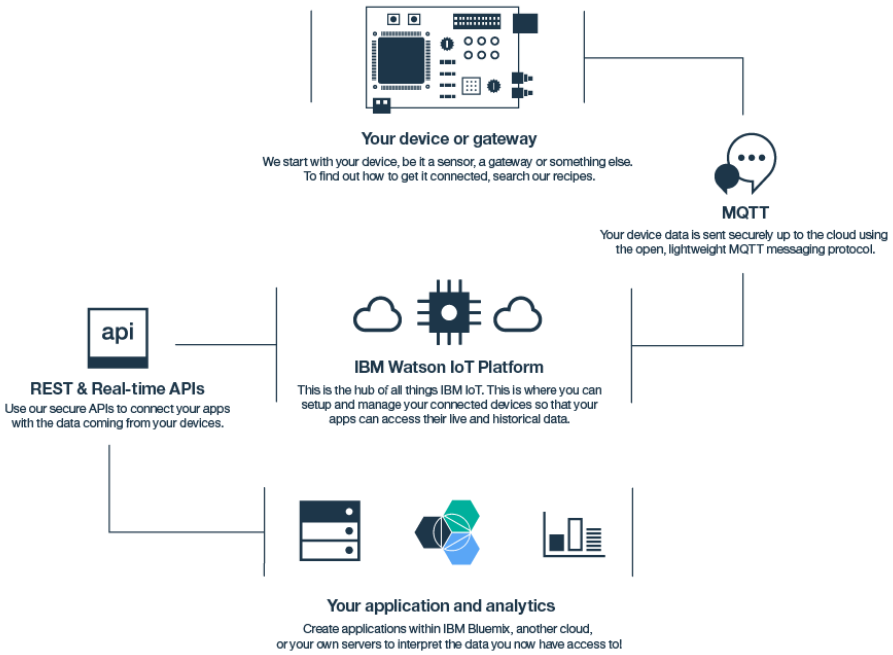

Figure 6: Infrastructure of the Bluemix service "Watson IoT Platform"

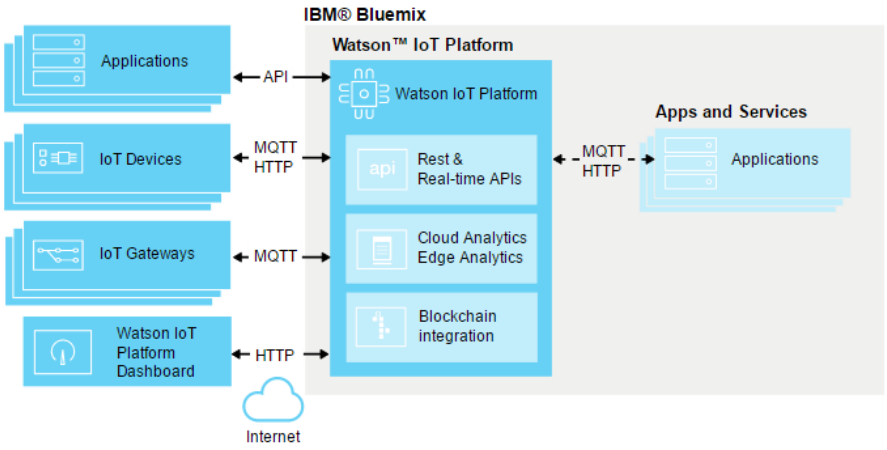

Figure 7: Architecture of the Watson IoT platform

Services Watson IoT Platform provides a way of representing data set values of different devices in the GUI for a quick overview, through "boards and cards". Analytics rules are defines to specify the conditions that trigger actions.

Cloud rules trigger rules for devices connected directly to Watson IoT Platform, while *edge* rules trigger rules for devices connected through gateways. Rules can trigger actions, which can be either of the following types: "Send email", "IFTTT"(Mentioned later), "Node-RED"(see below), "Webhook"(a means for altering a web page or application through callbacks).

Developing for Watson IoT Platform is supported by several SDKs and languages, such as C, C++, C#, Node.js, and Java. Furthermore "Node-RED" is supported for the platform, which is an integrated visual tool to develop applications, devices, and gateways.

As Watson IoT Platform is one of the IBM Bluemix services, it can be integrated with other *compatible* services that are hosted on Bluemix, e.g. for storage with "Cloudant NoSQL DB for Bluemix" or data to blockchain transactions with "Bluemix Blockchain"[IBM, 2017].

3.2.5 External IoT services

Apart from the big and popular IoT platforms investigated above, a lot of different IoT services exist, and some even effortlessly integrate with one another. Below, some are mentioned, and what they offer.

PubNub offers a "Data Stream Network" for serverless device connections and message relaying.

IFTTT stand for "If this then that", and is a rules engine for triggering actions through a variety of compatible systems, such as Philips Hue, and Weather Underground. REF(https://ifttt.com/about).

Cisco Jasper provides an automated connectivity management with the "Control Center" to link devices to global networks and back-end systems.

Oracle IoT Cloud Service offers fast connection and integration with real time data analysis.

3.2.6 Comparison

This section will compare the four IoT platforms from Google, Amazon, Microsoft, and IBM, analyzed in the above sections. It is interesting to find out the differences and what they actually offer and for what purpose. First off a technical comparison can quickly give an overview of the specifications along some of the key features for an IoT cloud platform including the following:

Device management: Keeping track of devices, their status and operations.

Integration: Access to operations and data that needs to be exposed from the platform.

Protocols: How data communication is transmitted. Important in regards to scalability.

Analytics: The data collected from devices needs to be analyzed in some way. The specifications to these parameters are illustrated in figure 3.2.6.

IoT Platform	Device management	Integration	Protocols	Analytics
Google Cloud Platform	Firebase	REST API	HTTP, gRPC, and gateway	Cloud Dataflow
AWS IoT Platform	Device Shadow	REST API	MQTT, HTTP, Web-Socket, and gateway	Amazon Kinesis, AWS Lambda
Microsoft Azure IoT Hub	Device Twin	REST API	MQTT, AMQP, HTTP, and gateway	Azure Stream Analytics
IBM Watson IoT	Dashboard	REST API	MQTT, HTTP, and gateway	IBM IoT Real-Time Insights

REF(https://dzone.com/articles/iot-software-platform-comparison) REF(https://blogs.endjin.com/2016/08/a
vs-azure-vs-google-cloud-platform-internet-of-things/)

An important property of data analytics in IoT cloud computing is real-time behavior, which all of the above platforms offer. Apart from the different analytics services they offer, they differ in some of the communication protocols. Amazon, Microsoft, and IBM have adopted the popular MQTT protocol, while Google offers the "general RPC framework" (gRPC) with their Cloud Pub/Sub service. REF(https://cloud.google.com/pubsub/grpc-overview). All of them support the HTTP protocol through a REST API.

When it comes to device management they all have some way of keeping statistics of registered devices. AWS uses "Device Shadow" as a virtual mirror of the device similar to Azure's "Device Twin". IBM uses the same concept and keeps a list of connected devices in the platform dashboard. Google on the other hand has no "real" integration of this concept, but uses the stand-alone Firebase service to maintain records of devices.

With this said, the four platforms seem very identical, at least in regards to technical features. Using the platforms and reading their documentation however, they seem to have primary focus on different application domains.

Azure for instance comes with two preconfigured IoT solutions directly aimed a business analytics and remote monitoring. AWS highlights the Lambda service together with the Rule Engine as one of the key features, making it possible to completely customize the cloud computed behavior. Watson IoT Platform focuses on application development and its integration with IoT, but offers different platforms for different needs, which is a design choice to not keep it all under same roof. Google seems to be a bit behind the others in regards to a full cloud IoT platform solution, but offers a real-time analytics service as the main feature together with the Cloud Platform.

3.2.7 Platform and services

The words "platform" and "services" have thus far been used interchangeably, but what is the actual difference in regards to this topic? TechTarget (REF(http://searchservervirtualization.techtarget.com/definition/platform)) defines it as *any hardware or software used to host an application or service.* With this definition a platform could be anything from an operating system to a vending machine. What is actually offered from the cloud platforms are services for e.g. cloud computing, tracking, storing, and messaging, and the ability to integrate these with one another through the Internet. Another term "Platform as a service" (Paas) seems to fit quite well on this concept, as its definition states that it is a *cloud computing model that delivers applications over the Internet* REF(http://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS). This means that it is a platform, but as it is *delivered* to the user, it is a service. Instead of having a in-house platform with an internal infrastructure offering different services, a PaaS provides this on the Internet as a single service.

4 Method

The method for carrying out this project is based on the "Step Wise" plan introduced in [Hughes and Cotterell, 2009] and can be seen in figure REF. The approach consists of several logically ordered steps for managing and carrying out a software project. This section will present the findings for step 3, *Analyse project characteristics*, which objective is to determine a development methodology.

4.1 Project characteristics

Projects may be distinguished by whether their aim is to produce a product or to meet certain objectives. The objective for this project has already been set, as this is the motivation for it to be carried out, which means that this project is product-driven. The system to be developed is both data and process-oriented, as collected data must be streamed and in a way be processed this way. It will be application specific, and not a general tool, as the processing will expose an

API for monitoring back posture. The system, however, could easily be used for similar purposes with few changes. The system requires a distributed real-time analytics tool, which is provided through the cloud platform. The system will ultimately be used as an expert system, meaning that the evaluation of data relies on some parameters in order to perform analytics. The nature of the environment for the system only depend on a flex sensor and some sort of microprocessor able to transmit data over Bluetooth. The actual computing will be distributed autonomous computing.

4.2 Project risks

The high-level project risks can be identified through three parameters:

Product uncertainty The project with ReRide is an ongoing work, and the requirements for the system is unspecified. The system requirements is just to expose some sort of API to allow self-monitoring, but the actual analytics is undefined and may come underway, but is otherwise in a test state.

Process uncertainty The product will be developed through prototyping in an agile environment. Uncertainties like wasted time and, deadlines and increments might be an obstacle and rise confusion.

Resource uncertainty The system development might require extensive knowledge of JavaScript and the Bluetooth Low Energy protocol, which is deemed uncertain how great a toll this might take in regards to available resource and time constraints.

4.3 Development methodology

Due to imprecise user requirements and since the system consist of the connection between multiple hardware devices as well as a distributed cloud platform, evolutionary prototyping is to be used as the life-cycle approach. Prototyping is one way for buying knowledge and reducing uncertainty. It is a working model of one or more aspects of the projected system. It is constructed and tested quickly and inexpensively in order to test out assumptions. An evolutionary prototype is modified until it is in a final form, in contrast to the throw-away prototype, which discards tested implementations and may be based on alternative resources.

One of the strengths of this approach is learning by doing, which will benefit this project, as there is some uncertainty, as described above. The context could be seen as a pilot project, and the development technique as a rather new one, which is another reason why a prototype is well suited. A problem to prototyping is changes during development, but as discussed above, this will be minimal.

5 Process

The project is carried out following the Step Wise plan introduced in section 4, in which the next step (following step 3 for analyzing the project), is to *Identify the products and activities*. This section will present the project plan including activities. An activity network[1] has been used to define the logical ordering of the system development, which is a product of the identified components required for implementing the system(REF Appendix). This is only seen as a guideline for where to begin and end for every iteration of the prototype development.

Step 5-8 in Step Wise are supposed to be carried out for every activity planned for a project, and finally when this is done the plan is ready to be executed. However, for this project these steps have not been done in-depth as their effect doesn't apply for this scope, as described in the following.

- The effort for carrying out each activity in the software development is difficult to asses for this project, as there is no previous work to base this on.
- The risks tied to each activity share the general risks identified as project characteristics in section 4.1.
- The resources needed to complete each activity does not make sense to define, as the development life cycle is based on evolutionary prototyping.

This does not mean a plan for carrying out the activities is not needed, as there is a strict deadline for the projected to be handed in. For this, "Baseline planning"[2] is used as the technique for creating a time line and defining requirements for deliverables and deadlines which can be seen in appendix REF.

6 Technical description

7 Evaluation

8 Discussion

9 Conclusion and reflection

[1] A model for scheduling activities and their relationships as a network, where time flows from left to right.

[2] Explain this technique, with REF to book

References

[Amazon, 2017] Amazon (2017). AWS IoT Features. https://aws.amazon.com/iot-platform/how-it-works/.

[Bagalkot et al., 2016] Bagalkot, N. L., Sokoler, T., and Baadkar, S. (2016). ReRide: Performing Lower Back Rehabilitation While Riding Your Motorbike in Traffic. *PervasiveHealth*.

[Bessis and Dobre, 2014] Bessis, N. and Dobre, C., editors (2014). *Big Data and Internet of Things: A Roadmap for Smart Environments*, volume 546 of *Studies in Computational Intelligence*. Springer.

[Catlett et al., 2013] Catlett, C., Gentzsch, W., Grandinetti, L., Joubert, G., and Vazquez-Poletti, J. L., editors (2013). *Cloud Computing and Big Data*. IOS Press BV.

[Google, 2017a] Google (2017a). Firebase. https://firebase.google.com/.

[Google, 2017b] Google (2017b). Overview of Internet of Things. https://cloud.google.com/solutions/iot-overview.

[Hughes and Cotterell, 2009] Hughes, B. and Cotterell, M. (2009). *Software project management*. McGraw-Hill Education, fifth edition.

[IBM, 2017] IBM (2017). Exernal Service Integration. https://console.ng.bluemix.net/docs/services/IoT/reference/extensions/index.html.

[Sterling, 2005] Sterling, B. (2005). *Shaping things*. Mediawork.

A Code