NTNU

Norwegian University of
Science and Technology

Introducing the BDX Platform - bringing IoT to the Fitness Sector

Jonas Westmo Strømsodd

Master of Science in Communication Technology
Submission date: June 2016
Supervisor: Peter Herrmann, ITEM

Norwegian University of Science and Technology
Department of Telematics

Title:	Introducing the BDX platform - bringing IoT to the fitness sector
Student:	Jonas Strømsodd

Problem description:

The concept of *Internet of Things* is gaining momentum in the health and fitness industry. Equipment from vendors such as FitBit, Strava and Under Armour allow users to measure and view data from their workout sessions. However, few systems provide the possibility of capturing data from several fitness apparatuses simultaneously.

The thesis will investigate the possibility of introducing a platform that will make available data originating from a network of fitness apparatuses. Doing so consists of assessing the hardware components, software development and the necessary cloud services required.

The goal of the project is to develop and test a prototype based on the results of the investigation. The thesis should conclude with a discussion of areas of future improvements of the system.

Responsible professor:	Peter Herrmann, ITEM
Supervisor:	Karls Aiwerioba, Stellarman

Abstract

The purpose of this thesis is to describe a system allowing users to record their workout data without acquiring custom gear, such as an activity wrist band.

Given this description, the necessary assessments required to develop and produce a prototype of said system is performed. After assessing the available technologies and system architecture options, the efforts would be made to develop a prototype featuring the described components of system.

The assessments were made, and recommendations on applicable technologies were made. Combined with simplifications made to the system topology to make it conform to the scope of this thesis, resulted in the decision to develop a prototype consisting of a hub, gateway and backend service. The hub and gateway components were going to be powered by node.js, communicating with sensors over Bluetooth Smart, while the backend service, powered by node.js and MongoDB, was going to be deployed to Amazon Web Services.

The three components have been developed as suggested, and the process of doing so is described in this thesis, including measures being made with respect to securing the application. Part of this thesis is also recommendations on further work that needs to be done to bring the system from a state of prototype to a state of being production-ready.

Sammendrag

Formålet med denne rapporten er å beskrive et system som lar brukere lagre data fra treningsøkter uten å måtte gå til innkjøp av nytt utstyr, som et aktivitetsarmbånd.

Basert på den gitte beskrivelsen av systemet, har de nødvendige vurderinger blitt gjort, med tanke på tilgjengelige teknologier og systemarkitekturer. Etter å ha gjort disse vurderingene, vil prototypen utvikles og testes.

Resultatet av vurderingene og arbeidet med prototypen er et system bestående av tre komponenter: en *hub*, en *gateway* og en *backend*-tjeneste (i mangel på gode norske ord for disse typer komponenter).

De tre komponentene har blitt utviklet og testet som foreslått. Prosessen det var å utvikle disse er beskrevet i rapporten. Inkludert er de hensyn som er gjort med tanke på sikkerhet i applikasjonen. Til slutt i oppgaven er en anbefaling av fremtidige forbedringer som er nødvendig for at det foreslåtte systemet skal bli klart for produksjon, da forenklinger har måttet blitt gjort for å begrense omfanget til denne oppgaven

Preface

This thesis concludes my Master's degree over five years at the Institute of Telematics at NTNU.

Before diving into the main contents of my thesis, I would like to make special thanks to some people that have assisted me during my work on this thesis.

I would like to thank Karls Aiwerioba, my supervisor at Stellarman, whose idea it was to propose the development of a prototype of the BDX system as a possible thesis, and for his assistance and advice throughout the semester.

I would also like to thank professor Peter Herrmann at the Institute of Telematics, allowing the development of the BDX system to be a possible thesis this semester, and for his advice and feedback while working on my thesis.

The work on this thesis has allowed me to do extended work with technologies and platforms with which I had not previously been able to. Assessing, modeling, and implementing the prototype has provided with useful feedback and experience that I am certain I will benefit from during later projects.

Finally, I would like to thank the administration at the Institute of Telematics, for their assistance and advice during my work on this thesis.

Contents

List of Acronyms

AES Advanced Encryption Standard.

AWS Amazon Web Services.

BLE Bluetooth Low Energy.

CA Certificate Authority.

GPIO General Purpose Input/Output.

IoT the Internet of Things.

JSON Javascript Object Notation.

LED Light Emitting Diod.

SD Secure Digital.

SDN Software-defined networking.

TLS Transport Layer Security.

USB Universal Serial Bus.

WLAN wireless local area network.

WPA2-PSK Wi-Fi Protected Access 2 - Pre-Shared Key.

Chapter 1
System description

This chapter will serve as the introduction of the thesis. It will describe the motivations behind the development of the BDX system, as well as introducing its various components. This chapter will focus on describing the high-level interaction between the components, and not delve into specifics related to the hardware and software development efforts, as these are described in later chapters.

Along with an introduction to the system, this chapter will address any simplifications made while developing the prototype. The reasoning behind the simplifications made was to reduce the complexity of the prototype, in order for it to better conform with the scope of this thesis.

1.1 Motivations

The motivations behind the development of the BDX system is to take advantage of the emergence of low-cost hardware, combined with cloud computing, in order to create a platform where users of fitness centers may record and store data from their workouts, without requiring the presence of custom hardware such as an activity tracking bracelet. While select vendors of such products already, to some extent, provide platforms of this kind to their users, these are vendor-specific. A key feature (or rather, composition of multiple features) of the BDX system is for it to no be vendor-specific, not excluding users depending on whose vendors equipment they should choose to acquire.

Portability and compatibility are other motivations behind the system. In order to more fluidly be able to install the platform at different fitness chains, the system will, rather than integrating directly with the software running in the fitness apparatuses from which one wants to record data, the platform will provide hardware solutions that may be mounted on these in a “plug-and-play” fashion.

1.2 Introduction to the system topology

Before later chapters of this thesis delve into detail about the components of the system, the system topology will be introduced and described from a high-level perspective.

An overview of the topology is shown in figure 1.1. In the diagram, only one group of hubs and gateway is shown, however multiple such groups will exist in a real system, each group representing one fitness center. It would indeed be interesting to investigate the requirement of having multiple gateways servicing one fitness center, as a form of load-balancing strategy, but for this thesis, the assumption is made that there is a one-to-one correlation between the number of gateways and the number of fitness centers having the platform installed.

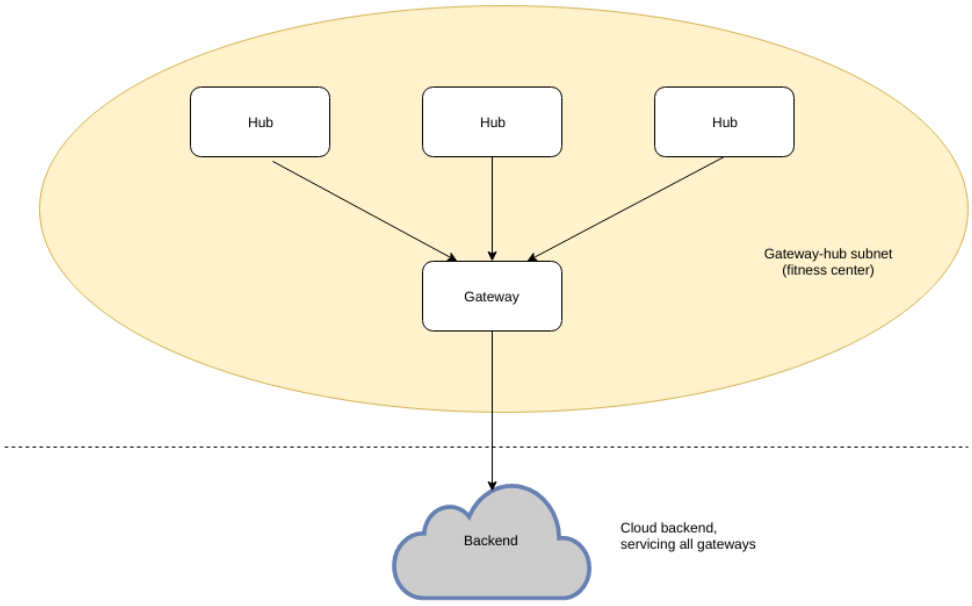

Figure 1.1: A high level overview of the system topology, showing the grouping hubs and gateway and the relation between the gatewy and backend service

1.3 User interaction

The topology depicted in figure 1.1 is limited in scope, in that it only displays the interaction between the components necessary for providing a platform storing and serving fitness data. In addition to the relations described, there is the user interaction subject to be discussed. In order to use the system for their workouts, a

user must download an application to their smartphone and subsequently complete a signup process. That way, information about registered users may be conveyed to the particular hub attached to the apparatus they wish to use at a given time, furthermore enabling the system to properly associate the resulting data from the workout with the identity of the user.

Though being a necessary component of the system, the aspect of user interaction has been simplified to a minimum during the development of the prototype, as its main focus has been the development of the software running on the components shown in figure 1.1, and the assessments that process requires. What this means more specifically is that the prototype will not include the development of a mobile application, as the effort associated with such development would make the combined effort of the prototype exceed that associated with this thesis.

Instead, for the prototype and its testing, an "example user" will be created and used when authenticating data transmission between the components of the prototype. In a production environment, a mobile application would serve as the entry point for such actions, but since the development of such application does not fall under the scope of this thesis, this matter had to be simplified.

1.4 Fitness data visualization

Part of the system is also a web application serving the collection of recorded workout data to registered users. A minimal such application will be developed, but its features will be limited to those required for a functional prototype to be presented. The features that fall outside the scope of the prototype will be touched upon in chapter 7, where future work on the system is described.

Chapter 2
Computer board assessment

Given the topology of the system that is to be developed as part of this thesis, the use of a controller board is required. The term *controller board* in this context refers to a small-sized computer that fulfills a set of requirements that will be presented in this chapter. Following the presentation of these requirements, a comparison between two candidate products will be shown.

2.1 Requirements

In order to make an informed decision on the choice of computer to use for the *hub* and *gateway* components, the requirements related to which functions they will fulfill is presented here.

Note that the requirement of compatibility with external components to allow for example radio connectivity, is implicit in the ensuing presentations and discussions.

User identification The sensor hub must support the technologies associated with being able to identify a user, in order to properly associate the fitness data recorded with the correct entity.

Sensor data processing The sensor hub must be able to receive and process readings from its connected sensors and perform any necessary preprocessing, before relaying the result of this process to the gateway.

Radio communication The sensor hub must support radio communication technologies in order to allow the hubs to communicate with the gateway. This includes supporting Bluetooth connectivity in order to allow the streaming of sensor data to a Bluetooth-enabled personal device, such as a smartphone.

Internet connectivity The gateway is responsible for transmitting data recorded by the sensors to the cloud service. In order to do so, the gateway must be able to connect to the Internet, either through Ethernet or Wi-Fi.

2.2 Candidate products

Prior to the work described in this thesis, two computers have been selected to potentially serve as the main component when building the sensor hub and gateway. These are the Raspberry Pi and the Arduino Yun. In this section a brief description of the two is given, followed by a comparison of the two with respect to how they would fit the requirements previously described in this chapter.

2.2.1 Raspberry Pi

The Raspberry Pi computer is a credit-card sized computer. It comes in a variety of sizes and specifications and has been chosen as a candidate for the project due to its

- low price
- compatibility with hardware
- compatibility with software
- wide developer community

The Raspberry Pi 3, released in February 2016, features compatibility with the most commonly used radio protocols (Wi-Fi, Bluetooth Low Energy, Blueooth 4.1) as well as the ability to interface with peripherals through its Universal Serial Bus (USB) connectors and General Purpose Input/Output (GPIO)-pins. [rpi] While it does not feature any form of built-in persistent storage, it supports the use of an Secure Digital (SD) card to store the required application code, including the operating system, which may be any flavor of Linux. The most common distribution being used is Raspbian, which is a special flavor of Debian. [ras]

2.2.2 Arduino Yún

The Arduino Yún is a computer with capabilities similar to those of the already described Raspberry Pi. It has built-in Ethernet and Wi-Fi support, USB connectivity, as well as the ability to run Linux, even though the distributions available are limited compared to those of the Raspberry Pi. In addition, the Yún distinguishes itself from the Raspberry Pi in that it is compatible with a series of peripherals intended to be used with Arduino products. Such perpherals would not be compatible “out-of-the-box” with the Raspberry Pi, which would in some cases increase the complexity

of the development process in the event that the Raspberry Pi became the candidate of choice.

2.3 Choice of computer

Based on the presented requirements and the presentation of the two candidate computers, none of the products need necessarily emerge as the superior choice to be used in the development of the signal hub and gateway components.

The author has more experience in working with the Raspberry Pi. Even though previously stated that this need not make the process of development any easier, it counts in the favor of choosing the Pi in the event of an even assessment. Due to this “tie-breaker”, the Raspberry Pi will be used to develop the components needed, namely the sensor hub and gateway.

Chapter 3
Radio communication assessment

One of the challenges related to projects utilizing Internet of Things, is deciding upon a means of radio communication. A set of viable options exist, each sporting their own advantages and disadvantages when compared to competing standards. In this chapter, the requirements dictating the eventual choice of the radio communications protocols for the sensor communications required for the prototype, are presented.

Following the presentation of the requirements, a set of available communication protocols and technologies will be presented. In the end, a protocol will be chosen to serve as the means of communications between the sensors and their corresponding sensor hub.

3.1 Requirements

The requirements of the application with respect to the radio communication taking place between the hub and its servicing sensors may be summarized as the following:

- low power consumption
- operating at a radio frequency at 2.5 GHz or below
- providing sufficient bandwidth to transmit sensor data

In order for radio communication technology to be suitable to be used for the prototype, it would need to fulfill these requirements.

It is worth noting that the amount of data that the selected protocol must accommodate (throughput) is equivalent to that of three signed floating point numbers per transmission. Depending on the configured interval of data emission, their combined amount of data is close to that of the bytes required to represent the three signed floating point numbers, which would be either twelve or twenty-four,

depending on the precision of the sensor equipment that ends up being used. For example, if the sensors are configured to transmit data every 500 milliseconds, the data transferred per second would be a little over 48 bytes per second.

3.2 Candidate technologies

There are a number of technologies adhering to the presented requirements, and these will be presented here.

3.2.1 Bluetooth Low Energy

Bluetooth Low Energy, often denoted simply as BLE, is an implementation of Bluetooth focused on reducing the energy required to transmit data between nodes. It operates at a 2.4 GHz radio frequency. While being designed to accomodate devices requiring low power consumption, the official specification does not include information on this matter, as this varies with what combination of hardware is utilized. The supported throughput of data is 0.27 Mbit/s. [blu]

3.2.2 ZigBee

ZigBee is an implementation of the 802.15.4 standard, with intentions similar to that of Bluetooth Smart of. It operates at a 2.4 GHz radio frequency and exhibits a data transfer rate of 250 kbit/s when running over this frequency band.

3.3 Decision

Both of the described technologies in this chapter adhere to the presented requirements for sensor data transfer, thus they are both fit to be used for the prototype.

The decision on which of the two to choose is largely influenced by compatibility. In order to minimize the efforts required to transmit data between sensors and the hub component, this is paramount. The sensor that the prototype will use is the SimpleLink™ SensorTag, which will be described in more detail in later chapters. It supports both Bluetooth Low Energy (BLE) and ZigBee (though after a firmware upgrade). As the Raspberry Pi 3 supports Bluetooth without the need of any accessories, Bluetooth LE will be used to carry the sensor data in the prototype system.

Alternatives to using the SensorTag and BLE, with respect to how to convert this stack into a production environment, will be briefly touched upon in chapter 7, addressing future work that may be performed on the system proposed.

Chapter 4
Backend requirements

In this chapter, the backend application requirements will be discussed. The requirements related to the process form the basis for a number of decisions, specifically, the choice of programming language to use, the persistence layer, and the cloud service provider at which the application will be deployed.

4.1 Programming language assessment

There exists a plethora of programming languages to use when developing the backend application. A selection of candidates will be presented. They have been chosen as candidates due to them being

- known to run on the Raspberry Pi
- well-documented, with a wide developer community
- scripted languages, increasing speed of development

The first point is indeed the most important one. Even though the backend application will run in the cloud, using the same, or similar, technologies wherever possible across the components of the system, is beneficiary. This is the case as it reduces the overall complexity of the software components related to the system, simplifying the process in the event of another group of developers taking over the project in the future.

4.1.1 Python

Python is a widely used interpreted programming language. It can be used for anything from simple web applications to controlling a Light Emitting Diod (LED)-panel on a Raspberry Pi. A key feature of the Python platform is the large number of available third-party software packages. Packages allowing the developer to perform

network calls, control serial ports, communicating over protocols such as Bluetooth or ZigBee makes the language a good fit for the Internet of Things (IoT) projects. It also features a simple syntax, which allows developers to implement features quickly, even though this comes at the cost of type safety.

4.1.2 node.js

node.js brings the JavaScript language, originally created for frontend development, to the server stack. Running as a server implementation of the Chrome V8 engine, it has become one of the most used languages for web application development, especially when it comes to applications needing to be lightweight, such as event-based servers. Event-based server applications are encountered at large when working with IoT.

4.1.3 Ruby

Another option when discussing the choice of programming language, is Ruby. Ruby is another popular, interpreted programming language with capabilities similar to those of Python. It will run on a Raspberry Pi and features a wide developer community along with an array of available packages that work in a "plug-and-play" fashion.

4.1.4 Language selection

Since all of the programming languages presented run on the Raspberry Pi, features a wide community (making it easier to find examples during the development process), none of the candidates stand out as a clear selection as the language to use when developing the prototype. Therefore, the authors experience prior to the work described in this thesis comes into play. The author has experience in working with the Python and node.js languages. Since the node.js lanugage has known support (in third party packages available) for interfacing with the sensor equipment that will be used for the prototype, described in chapter 5, it will be the language of choice.

4.2 Persistence technology assessment

The core functionality of the backend application is the ability to associate recorded sensor data with information on the user responsible for producing the data. In order to do so, the application will need to utilize some form of persistence technology in order make possible the retrieval of the data at a later stage. For the sake of completeness of this report, a brief assessment of whether using a relational or non-relational database best suits the requirements of the application.

4.2.1 Relational databases

One of the most commonly used persistence technologies used, besides working directly with files, is relational databases. Such databases allows data to be connected in intuitive ways, through the use of *foreign keys*, which allows *joining* mechanisms to be performed in order to combine data from different tables. If such queries, combining data not always stored together, are frequented by the application, a relational database may be a preferred choice over other technologies. However, using a relational database also adds to the complexity of the software, as special care needs to be taken in order to make sure that that data that is to be stored adheres to the specification of the tables. If table layouts are altered after the application is deployed, migrations are needed to alter the schemas, again increasing the complexity.

4.2.2 Non-relational databases

A non-relational database differs from relational database in that it does not rely relations (thus the name). This gives it more flexibility in a number of cases. The prevailing case is in the event of a traffic projecting a large amount of traffic causing a high number of read-operations. In this case a non-relational database may provide better performance than a relational database. This may indeed be the case for the BDX system. However, if the traffic is great, as it will not be for the prototype described in this thesis. Thus an eventual choice of using a non-relational database would be a matter of preference, rather than it being based on a deep theoretical discussion.

4.2.3 Selection of persistence technology

As previously stated, for the prototype, the amount of traffic expected does not dictate whether one should choose to use a relational database or a non-relational database for the persistence layer of the backend application. The proposed database architecture, as it will be described in detail in chapter 5, is of low enough complexity that it may be developed with an equally small effort given both technologies. For the protoype, a non-relational database was used. Specifically, MongoDB was used. [mon]

4.3 Cloud service providers

The backend application is intended to run in the cloud. This dictates the need of choosing which cloud service providers from which to buy the required services. Although there exists a large selection of providers out there, the choice will be made between using Amazon Web Services and Microsoft Azure.

4.3.1 Microsoft Azure

Microsoft Azure offers an array of cloud technologies, allowing developers and system engineers to realize their cloud applications, regardless of complexity of topology. In its simplest form, this involves creating virtual machines running Windows Server, or some flavor of Linux. Access control, monitoring and provisioning tools are also included in their software suite.

4.3.2 Amazon Web Services

The largest cloud provider of its kind, Amazon Web Services offers the same set of services that Microsoft Azure does. Virtual machines, simplifed application deployment, cold storage, and Software-defined networking (SDN) capabilities are available. It is well-documented, and features a wide community of supporters, making examples easy to find during the development and deployment processes of the prototype.

4.3.3 Selection

None of the two candidate providers stand out in any particular way. The needs of the backend application in terms of cloud deployment may be met by simply creating virtual machines, a service offered by both providers. Thus, Amazon Web Services (AWS) is chosen, due to the author having slightly more experience with working with its services prior to this thesis.

Chapter 5
Software development

This chapter will describe, in detail, the software components that compose the proposed system. While the components of the system and their interactions are described at a high-level in chapter 1, it makes sense to also discuss the software of the three components. This chapter focuses on the purpose and requirements of the software components, whereas thee assessments of the supporting technologies available are made in chapter 4.

Note that the specifics of securing the transport of the application data between components are described in chapter **??**, rather than in this chapter.

5.1 The hub

As depicted in the system description in chapter 1, the hub is responsible for reading data from sensors attached to the training apparatus. Specifically, its purpose may be broken down into the following features:

- Communicate with the sensor to receive data
- Perform any necessary preprocessing
- Relay processed data to the gateway

5.1.1 The SimpleLink™ SensorTag

The sensor used for the prototyping is the SimpleLink™ SensorTag, which contains a number of sensors and allows for communication over the Bluetooth Smart®, Zigbee®and 6LoWPAN protocols. For the prototype being subject to the work discussed in this thesis, Bluetooth Smart®will be used to transmit data between the tag and the hub. Note that the use of this sensor is a choice made for practical reasons and availability at the time of development. More information on this matter and further relevant comparisons of wireless protocols may be found in chapter 3.

5.1.2 Encapsulating Bluetooth communication

In its simplest form, communication with the SensorTag may be broken down into three phases:

1. Enable sensor
2. Configure the notification period
3. Notify the sensor to start recording

These actions are achieved in the software by sending specific messages to the SensorTag. The software developed for the prototype utilizes an open-source library which encapsulates the exchanging of messages and wraps them as callbacks and event emitters [noda]. This greatly simplified the development process, in that the actual Bluetooth communication software did not need to be "re-developed" for the purpose of the prototype system.

5.1.3 Sensor data preprocessing

One of the core features of the hub is to perform any necessary preprocessing of data received from its attached sensors, prior to submitting this data to the backend, via the gateway.

In the case of the prototype, this involves converting changes in magnetic fields over time to a rotational frequency of a bicycle wheel. By processing the data points streamed from the SensorTag, the strategy for this approach involved the attempt to detect the period of a sine curve, in order to determine the rotational frequency at a given point in time.

Whenever the user would end the workout, a summary of the data recorded would be submitted to the backend, via the gateway. This summary would include the total number of rotations, the mean frequency recorded, as well as the minimum and maximum values calculated during the course of th workout.

5.2 The gateway

The gateway is responsible for receiving data from the hubs at a particular site, and transmitting this data to the backend.

From a topology perspective, the gateway acts as a proxy between the hub(s) and the outside world, specifically, the backend running at AWS. The specific actions involved are as following:

– Authenticating hubs

– submitting data received from hubs to the backend.

5.2.1 Hub authentication

The gateway is responsible for performing the necessary authentication whenever a hub attempts to submit data to it. The way this is done in the prototype system is by using Transport Layer Security (TLS). More specifically, upon creation and installation of a new subnet, an RSA key pair is created for each hub, and in turn used to issue a Certificate Signing Request. This certificate in turn is signed by the gateway, acting as a Certificate Authority (CA). The built-in TLS module of node.js is used to achieve this when developing the necessary software.

The security measures employed in the prototype system are more thoroughly described in chapter 6.

5.2.2 Two network interfaces

To properly isolate the subnet of hubs and gateway, the proposed solution of realizing the network architecture between them is to setup a separate wireless local area network (WLAN), where the hubs connect to the gateway using either a static IP address assigned to the gateway, or a hostname advertised to the network. This is the gateway-hub concern being addressed. The second concern with respect to networking between the components, is between gateway and the backend service. Since the backend service is intended to run in the cloud, the gateway must have access to the Internet. How this is intended to be implemented in the proposed system is to have the gateway communicate on two network interfaces, ne connected to the local gatway-hubs WLAN, and one connected to the Internet. The Raspberry Pi, described in 2.2.1, is capable of this, by offering both Wi-Fi functionality, as well as Ethernet compatibility.

5.2.3 Installation process

As the proposed system involves the deployment of a large number of gateway-hub networks, attention must indeed be made to the matter of installing the equipment and software. This is true even though the prototype only involves one such grouping.

When discussing the installation process with respect to the gateway, there are a number of clear-cut concerns that need to be addressed, including

– Configuring of network interfaces

– Install the application

- Register the gateway with the backend, in order for the gateway to perform authenticated requests
- Create a private key and TLS certificate in order to sign certificates for hubs in the network

During the prototype development, these actions have been performed manually. However, in order for the installation process to be performed smoothly in the event of a production environment, they would benefit from being automated in some fashion. A common way to perform such automation is to maintain a collection of shell scripts that perform the necessary work needed to run the software, and run these scripts on the Raspberry Pi designated as the gateway prior to installation at a particular fitness site.

5.3 The backend

One of the key features of the proposed system is the ability to persist recorded sensor data associated with a session, as a representation of a completed workout. In order to achieve this, a backend application needs to be developed. A multi-purpose application, its responsibilities include

- Receive recorded sensor data from a gateway at a particular site.
- Persist the data
- Displaying recorded data session to authenticated users,

which dictate the minimum requirements of this software component in a prototype context.

5.3.1 Data reception

The data path taken when the user completes their workout, is shown in figure 5.1.

Figure 5.1: A highly simplified illustration of the system topology

The data is first sent to the gateway, who in turn relays the data to the gateway. This is done by issuing HTTP requests, more specifically, the gateway issues a POST

request, whose body describe the details of the workout just completed. An example body payload may look like the following:

```
{
  "user": "aUsername",
  "hubId": "somehubid",
  "recordedData": {
    "totalRotations": 150
  },
  "auth" : {
    "gatewayname": "name",
    "password": "gatewaypassword"
  }
}
```

The results of the workout are put in the *recordedData* field. Some metadata is also supplied, including the username and password assigned to the gateway that will be used to confirm the authenticity of the gateway presenting data to the backend. The *user* field contains the username of the user wanting to submit their workout data. During the prototype phase, this was hard coded (an example user) into the hubs configuration, but ideally this is obtained by authenticating user prior to starting the workout. More information on this may be found in **??**, where possible improvements and future work on the system are described.

The payload of the request in turn is parsed as Javascript Object Notation (JSON) and stored in the database. As a sign of a successful process of persistence, the JSON presentation of the created workout document is sent as the body in the reponse to the request, along with a HTTP status code of *200 OK*, indicating a successful request. Should the persistence action fail due to a malformed request body, a *400 Bad Request* is issued as the response.

5.3.2 Database architecture

The backend makes use of MongoDB, a NoSQL-database technology, to achieve data persistence at the backend layer. The database architecture present in the backend system developed as part of the prototype is shown in figure 5.2.

Note that this is the minimum architecture required for the prototype to function as desired. Additional database fields describing metadata have not been included, to better illustrate the interactions between the components.

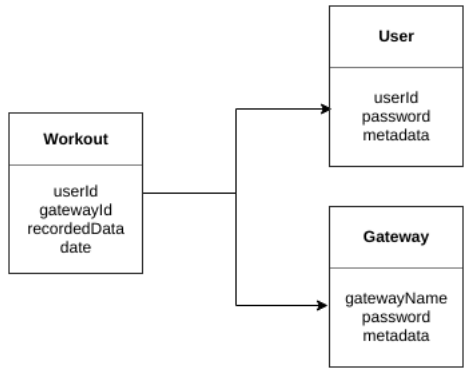

Figure 5.2: The database architecture used in the prototype backend

Chapter 6
Security

This chapter will discuss the security measures made while developing the prototype. The different components that together compose the proposed system topology are subject to different security requirements. These will be presented in turn, and possible ways to meet these requirements will be discussed.

6.1 Gateway network

When deployed at a particular fitness site, BDX hardware will be present in the form of a number of *hubs* and one or more *gateways*, responsible for servicing the installed hubs and relaying the fitness data recorded to the backend service. The transportation of data between a hub and its responsible gateway may be achieved in a number of ways, depending on where in the protocol stack the security functions are implemented.

6.1.1 Link layer security

One way of connecting the hubs to the gateway at a given fitness site, is to connect them to the same WLAN, and secure this network using Wi-Fi Protected Access 2 - Pre-Shared Key (WPA2-PSK). By allowing only the hubs and gateway to connect to the specified network, one reduces the risk of an adversary being able to sniff data packets transmitted between hub and gateway, as the complexity of such an attack is equal to breaking Advanced Encryption Standard (AES).

6.1.2 Transport layer security

Even though the possibility of an adversary penetrating the hub-gateway subnet at a particular site is reduced when using WPA2-PSK, as previously described, it makes sense to secure the transportation of data at the transport layer as well. One way to achieve this, which fits into the suggested network topology, is to employ TLS certificate-based authentication between the hubs and their servicing gateway, where

the client needs to authenticate itself with the server in order for the request to be accepted. Doing so would require the following to be achieved, prior to installing a network of hubs and gateway at a fitness site:

- Installing a CA at each gateway
- Issuing certificates to all hubs, signed by gateway CA
- Configuring the gateway application software to verify the identity of the client attempting to transmit data

6.2 Backend security

At the core of the BDX ecosystem one finds the backend server application. The application is, as previously described, responsible for persisting workout data presented to it by the gateways at the various fitness locations, and presenting this data to authenticating users. These two requirements impose two sets of security requirements, which will be discussed in turn:

- Authenticating a gateway when submitting data from a workout
- Authenticating a user visiting their personal dashboard to view a collection of their workouts

6.2.1 Gateway authentication

There are a number of ways to have the gateway authenticate itself with the backend application in order to submit data recorded from workouts. One way to achieve this, which is used when developing prototype, is to employ *HTTP Basic Authentication*, which is the most basic authentication leveraging a combination of username and password to allow the gateway to submit its data. This approach, however, relies on transferring the username and password as a base64-encoded string [htt] To allow the credentials to be securely transferred between the gateway application, running at a fitness site, and the backend application, the cloud application must allow the gateway to connect using TLS, which imposes a new requirement on the web server responsible for servicing the backend application. Further details regarding the deployment of the backend application at Amazon Web Services may be found in appendix A.

6.2.2 User authentication

A common use case of the system is for a registered user to be able to view a list of their completed workouts at supported fitness sites. Since such data may be

considered sensitive by a number of users, a user must need authenticate oneself in order to view the workout data. There are a number of ways to hide the data from unauthorized users, and thus achieve this requirement of *confidentiality*. As the backend application is meant to be ran over TLS, employing *HTTP Basic Authentication* remains secure in this case, and its security is determined by the cipher suite employed when running TLS. The backend application uses support for TLS built into the node.js platform, which, according to [nodb], employs a cipher suite which accords with "best practices" with respect to known attacks on the protocol.

Chapter 7
Future work

The purpose of this chapter is to address future improvements that are either required by the system in order for it to be production ready, or that have been discovered during the development process, but have not made it into the final prototype.

7.1 Backend domain name and signed TLS certificate

At the time of writing, the backend service is deployed at AWS under an automatically generated domain name, related to the IP address assigned to it during process of deployment. Ideally, this backend service should be reachable by a domain name registered to the owner of the application, namely the company Stellarman. Another consequence of the backend service not being reachable behind a registed domain, is the inability to aquire a signed TLS certificate. The result of this is that due to the unsigned nature of the current TLS certificate, users trying to access the application will be met by a warning from their browser, which may lead them to think that there is an error with the running application, or that it has been compromised.

7.2 Sensor data improvement

The goal of this thesis was to develop a prototype system. The main emphasis was to produce a prototype featuring all the components advertised to ber part of topology. This came at the expense of some hindrances along the road. While developing the software for the hub, it was discovered that the magnet and magnetometer used, intended to be processed into rotation frequencies, did not produce the desired results. However, the software for the hub modeled in a way to make the recording subcomponent easily replaceable. In order for a future iteration of the system to produce useful results, efforts would have to be made to improve the use of sensors to record the desired data. One such effort may be to use Hall effect sensor, which may be more suitable to the needs of the application than the combination of the the magnet and magnetometer used.

Chapter 8
Conclusions

This thesis has introduced and described a platform for recording and persisting fitness data. The goal of this thesis was to conduct the necessary assessments in order develop a prototype within the allocated time frame (semester).

The result of the assessment of programming languages was to use node.js to power the applications required. The hub and gateway components were going to run on Raspberry Pi computer boards, as was the result of the hardware assessments. The SimpleLink™ SensorTag was used as the sensor component. The backend service was to be deployed to AWS.

The result of these efforts was a prototype adhering to the requirements presented and discussed throughout this thesis.

In terms of security, TLS was employed to secure the transportation of data throughout the application, while HTTP Basic authentication was used to authenticate "human users" logging in through their web browsers.

Throughout the process of developing the prototype, issues were discovered that eventually led to simplifications being made. While not jeopardizing the functionality of the prototype as it was defined at the beginning of the thesis, it paved the way for documenting any future efforts required to enhance the prototype and bring it closer to a production state.

References

[blu] Bluetooth 4.2 core specification. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439. Accessed: 2016-06-16.

[htt] HTTP authentication: Basic and digest access authentication. https://tools.ietf.org/html/rfc2617. Accessed 2016-06-02.

[mon] The mongodb 3.2 manual. https://docs.mongodb.com/manual/. Accessed: 2016-06-18.

[noda] node-sensortag. https://github.com/sandeepmistry/node-sensortag.

[nodb] Node.js v6.2.1 documentation. https://nodejs.org/api/tls.html#tls_tls_createserver_options_secureconnectionlistener. Accessed 2016-06-04.

[ras] About Raspbian. http://www.raspbian.org/RaspbianAbout. Accessed: 2016-03-21.

[rpi] Raspberry Pi 3 model b. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed: 2016-03-21.

Appendix A
Deployment

This appendix will describe the technologies and strategies associated with the deployment of the prototype on AWS.

A.1 Docker

Besides the technologies related to the implementation of hub, gateway and backend components, which have been described in the thesis, Docker was used while deploying instances of the developed application to AWS.

Docker is a virtualization technology which allows applications to be run as containers, defined as a set of requirements run upon a Linux image. The advantage of using this technology is that of portability. While developing the applications, the author of the thesis made us of Docker to ensure an isolated environment for the applications to run in. When deploying the application, all that was required was to build and run the same containers as had been used while developing locally. This also allows for simple emulation of the hub and gateway application, which are are intended tobe run on the Raspberry Pi, but may now be deployed to the cloud for demonstration purposes.

A.2 Amazon Web Services

The cloud vendor selected to host the prototype system was Amazon Web Services. In a production environment, only the backend service would run at AWS. However, to demonstrate the interaction between all the components of the system (hub, gateway and backend). Emulated version of the hub and gateway was also deployed to the cloud. Additionally, a minimal GUI application was developed, intended to emulate the display of the hub, or the mobile application. This was also deployed to AWS.

To minimize costs associated with the prototype development and its deployment, the smallest available instance types from the Amazon Elastic Cloud 2 set of services were selected, namely the t2.micro instance type.