Bachelor's thesis

Information and Communication Technology

2018

Ville Kilpinen

CREATING AN INDUSTRIAL BLUETOOTH SENSOR NETWORK FOR PREDICTIVE MAINTENANCE

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

ICT

2018 | 48 pages

Ville Kilpinen

CREATING AN INDUSTRIAL BLUETOOTH SENSOR NETWORK FOR PREDICTIVE MAINTENANCE CLICK HERE TO ENTER TEXT.

Predictive and preventive maintenance is starting to grow with the promotion of IoT and machine learning. Traditionally, the large industry plants have big potential when it comes to big data applications. The purpose of this project was to create a monitoring system to improve efficency, provide a better control over resources, and provide added value to industry processes.

The objective of the thesis was to create an application for Fidera Ltd. to gather, analyze, and present the data in a useful manner.The thesis includes research, implementation, testing, and prototyping in a real environment. During the thesis, the research focused on finding on a fitting beacon manufacturer and the best wireless technology for short-range applications. The software was implemented using Python, the wireless technology of choice was Bluetooth, beacons were provided by Confidex and Ruuvi, and Raspberry Pi functioned as a relay between the beacons and the cloud. Raspberry Pi was bootstrapped by a basic Linux operating software, and Chef automation software was used to install distribution-specific packages and edit configuration files.

The software was designed using behaviour-driven development, and it includes a Bluetooth scanner, a Bluetooth packet parser, and a label printer. The software includes comprehensive documentation that describes the functionality and command-line arguments.

The prototype consisted of dozens of Bluetooth beacons from two different manufacturers, five gateway devices, and a vpn router. Four gateways were able to connect to internet, and all gateway devices back up their data to local files. The gateways with online access were able to send their measurements over the Internet. The measurement data is stored in a database using a cloud application and a user interface will present the data using graphs and gauges.

The result was a working prototype of a data gathering system which could easily be scaled and its data interpreted.

KEYWORDS:

Internet of Things, Python3, Bluetooth, Raspberry Pi, predictive maintenance

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Informaatio- ja kommunikaatioteknologia

2018 | 48 sivua

Ville Kilpinen

BLUETOOTH-SENSORIVERKON RAKENTAMINEN TEOLLISUUTEEN

Click here to enter text.

Ennaltaehkäisevät ja ennustavat huoltoprosessit ovat alkaneet kiinnostaa myös teollisuuden aloja, joilla prosessit ovat pysyneet lähes muuttumattomina vuosikymmeniä. Suuret teollisuuslaitokset tuottavat suunnattoman määrän dataa, jota usein ei hyödynnetä missään määrin. Opinnäytetyön tarkoituksena olikin luodan järjestelmä, jolla voidaan tehostaa teollisuuden prosesseja, antaa teollisuuden toimijoille paremmat työkalut laitteiston hallintaan sekä tuottaa lisäarvoa hankituille laitteille.

Opinnäytetyön aikana kehitettiin Fidera Oy:n toimesta järjestelmä, joka mittaa, analysoi sekä visualisoi teollisuuden laitteista tulevaa dataa. Opinnäytetyö sisälsi tutkimusta, ohjelmistokehitystä, testausta sekä laiteasennuksia teollisuusympäristössä. Ohjelmistokehitys tehtiin Python-ohjelmointikielellä ja sisälsi monia uusia ohjelmistokehityksen työtapoja. Ohjelmistokehityksen aikana toteutettiin käyttäytymishakuista kehitystä, ja se sisälsi kolme alaosiota: Bluetooth-skannerin, Bluetooth-pakettien käsittelyn sekä lisätyökalun luomisen laitetunnisteiden tulostamista varten. Ohjelmistoa varten luotiin myös mittava dokumentaatio, joka käsittää asetuksien asettamisen, virhetilanteet sekä yleisen toiminnallisuuden.

Järjestelmän prototyyppi koostui kymmenistä Bluetooth-majakoista kahdelta eri valmistajalta, viidestä tukiasemalaitteesta, sekä pilvipalvelusta. Bluetooth-majakat tulivat kahdelta eri valmistajalta, Ruuvilta sekä Confidexilta. Tukiasemana toimi uusin Raspberry Pi -minitietokone, johon asennettin Linux-käyttöjärjestelmä ja joka yhdistettiin 3G-reitittimellä Internetiin. Käyttöjärjestelmän lisäksi tukiasemalle asennettiin räätälöity Bluetooth-skannausohjelmisto, asetuksienhallintaohjelmisto Chef sekä viestienvälitysohjelmisto RabbitMQ.

Tuloksena oli toimiva datankeruujärjestelmä, jota pystytään helposti skaalaamaan ja jonka data on helposti tulkittavissa.

ASIASANAT:

Asioiden Internet(IoT), Python3, Bluetooth, Raspberry Pi, ennaltaehkäisevä huolto

CONTENTS

LIST OF ABBREVIATIONS (OR) SYMBOLS

AI = Artificial Intelligence

API = Application Programming Interface

AWS = Amazon Web Services

BLE = Bluetooth low energy

CSV, csv = comma separated values

FFT = Fast Fourier transform

GB = Gradient boosting

Gb = Gigabytes

gE = Enveloping Acceleration

HTML = Hypertext Markup Language

IoT = Internet of Things

JSON = JavaScript object notation

MAC = Media Access Control

MS = Microsoft

MSE or RMSE = mean squared error or root mean squared error, respectively

NFC = near field communication

pip = Python package manager

R = Software kit for statistical data analysis and visualization

RAM = Random access memory

RDS = Amazon Relational Database Service

SOC = System on chip

SSL = Secure sockets layer, encryption protocol over the Internet

SVG = Scalable Vector Graphics

UI, GUI = User interface, Graphical User Interface

UID, EID = unique identifier and encrypted identifier, respectively

VPN = Virtual Private Network

ZMQ = Zero Message Queue, asynchronous messaging library

1 INTRODUCTION

The purpose of this thesis is both to learn more about one of the most prominent areas in Information Technology and to develop a data analysis system for the company that commissioned the thesis.

The method used to achieve the purpose of the thesis is the following: Research the tools and technologies required to build a data analysis module for a cloud-based platform, form a baseline for predictions from old data, prototype the system in a controlled environment, use baseline to compare results to the prototype, and then tune the system to work with desired accuracy.

Using sensory information to detect desired events is nothing new, but to predict accurately when the machinery needs maintenance saves the user money and manpower. A user can follow the state of machinery in real time, making the maintenance more efficient and accurate, when compared to manual measurements once or thrice a day. Conditions which affect the maintenance interval can vary immensely. This system is designed to be used in an array of machines from electric motors to industrial rock crushers, and from conveyor belts to equipment in larger kitchens. The sensory data is transferred via a Bluetooth network, and currently Bluetooth mesh is emerging as a highly desired technology. The project prototype will be tested both with and without the mesh network, with the hope that the mesh-enabled product will be lower in cost and higher in efficiency. A mesh network also enables gathering of sensory data from places normally difficult to reach.

Utilizing machine learning, however, is quite modern. It is not new, but it is gaining popularity in data science and analytics. Its purpose is to learn the system, its parameters (inputs) and what the parameters mean (outputs). There are many different machine learning tools and this thesis compares a few. It is also important to form a good initial hypothesis for the machine learning model since the system outputs are dependant on it. For example, using machine learning in predictive maintenance, the hypotheses test whether or not there is a significant difference in temperature measurements from different days or time points. If such difference is found, the system updates, and outputs the predicted lifespan or maintenance interval.

For this system, the data used is the temperature of the machine and its vibration. It is important to take ambient temperature and air humidity in the account for the purpose of finetuning the machine learning algorithm. For vibration, there needs to be a filter that removes spikes, for example in electric motor's case, caused by many different sources such as motor unbalance or stator tooth resonance.

Creating a predictive model from these measurements is fairly straightforward when the system produces so much data. Whether or not the model will be accurate remains to be seen. Each piece of machinery has unique properties that must be taken in to account. The thesis project will include software development, data analytics, hardware research, and some electronics and field work in industrial environment. During the thesis, it is also planned to apply a multitude of machine learning and analytical methods to produced dataset in an attempt to accurately predict a desired outcome such as maintenance interval or lifespan of a piece of machinery. There are a number of different tools to create a predictive model and they are fairly developed and the the ones used in this thesis, are completely open-source.

Throughout the thesis, the aim also is to critically think about the viability of the source information and software, and modify them to suit the needs of the project. There is an abundance of information on the subject and finding and utilizing the relevant pieces will be a challenge.

2 KEY TECHNOLOGY OVERVIEW

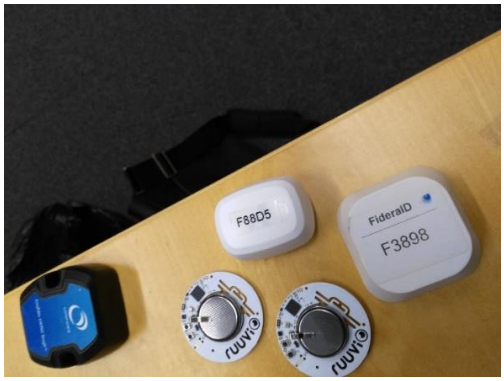
2.1 Code

Most of the code is written in Python for the client-side and analytics, and javascript or typescript for the server-side and UI part. Python is a popular high-level programming language that works well with open source operating systems such as Ubuntu. The gateway device requires three Python libraries: Beacontools to read and scan Bluetooth data, Pandas to organize the received data, and Zmq to send data over the Internet. The Beacontools library is a customized one installed within the application folder.

Python and R are great tools for the analysis part. Their vast libraries for data handling and visualization include for example Numpy, Scipy, Pandas, and Matplotlib for Python, and Tidyverse, and Ggplot for R. Both environments have their own niche and specialities, and they work together quite seamlessly. Javascript and typescript are widely popular web application languages and with the Angular 6 framework are used to build an UI with a dashboard for the customer.

2.2 Bluetooth Beacons

Beacons are devices connected to each other via Bluetooth network. They monitor desired variables, which in this project initially are temperature, vibration and humidity. They can be easily customized further. The most recent beacons utilize Bluetooth 4.x and Bluetooth Low Energy (BLE) technology. The devices that use BLE are often more energy-efficient, but backward-incompatible with previous Bluetooth versions, or “classic” Bluetooth as a trade-off. [1] The Bluetooth 4.x version does not contain full support for the mesh network. As such, prototype phase beacons installed with the Bluetooth 4.x can only send data, but not receive. Picture 1 shows examples of what the beacons might look like. Beacons send the data they gather by broadcasting, or advertising the data as formatted Bluetooth packets to their immeadiate proximity. The devices that are configured to be able to read this information are called Bluetooth receivers, or in this project, gateways.



Picture 1. Bluetooth Beacons.

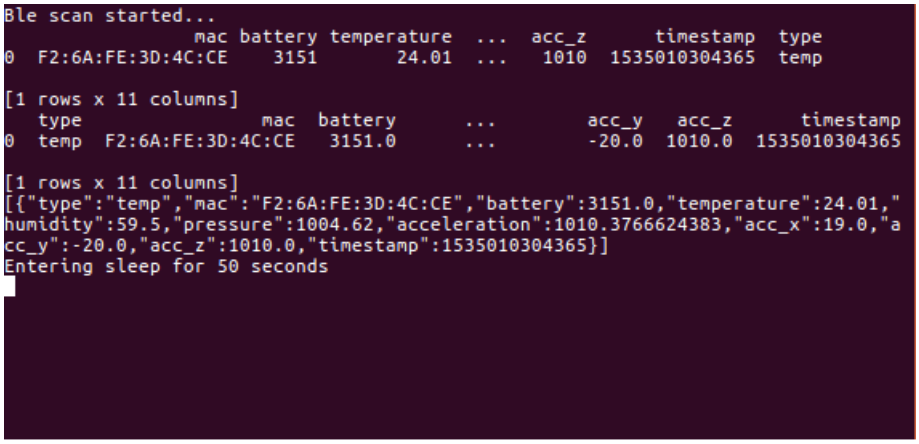
2.3 Measurements

Bluetooth beacons were used to measure temperature, humidity, acceleration, and beacon battery levels in the thesis project. The beacons can be potentially customized with any available sensor circuits to provide additional measurements which for example include atmospheric pressure, noise levels and air quality. Most of these are basic tracking of their respective sensor states, but generating vibration data proved somewhat more complex. The beacon reports three voltages from a single triaxial accelerometer, which indicates the amount of stress the sensor is under. For example voltage caused by gravity acceleration is approximately 1000 millivolts on a single axis. This value is calibrated by the component manufacturer.

Picture 2 shows an example of a scan. In this example, only a single beacon was detected after filtering out unwanted Bluetooth devices. The scan shows different variables included in the telemetry coming from the beacon, as well as their MAC addresses. Depending on the manufacturer, beacons can be identified by different methods such as MAC addresses, instance id's, namespaces, or all of them combined. In the picture below, the voltage caused by the acceleration due to gravity is reported on the z-axis. Observing this behaviour is useful in determining the orientation of the device. The total magnitude of acceleration can be acquired from the formula:

$$A = \sqrt{x^2 + y^2 + z^2}$$

More importantly, the accelerometer data can be refined into vibration data using fast Fourier transform (FFT).



Picture 2. Output of a scan.

2.4 Bluetooth Network

The devices in the client-side will be utilizing a Bluetooth low energy network. BLE technology requires at least Bluetooth 4 version, and it is very energy-efficient Bluetooth protocol that is able to handle one-way traffic of Bluetooth packets. Beacons in the BLE network will send their configured packets to a gateway device which acts as a Bluetooth receiver. The most recent Bluetooth devices are also capable of mesh networking, which is a protocol based on BLE. Mesh network refers to a topology that describes a network where connected devices are capable of limited two-way traffic. In this context, limited means that the beacons are not able to process the information sent by other similar devices, but they will forward it until the information reaches their destination, the gateway device. Both technologies are considered to be applied to the final product. [1,2,3]

2.5 Cloud Infrastructure

Cloud is responsible for receiving, handling, storing, and visualizing the data coming from the beacons. It contains modules that handle messages and packets coming via Internet. The packets contain sensor identification, sensor type, sensor data, and timestamps which is to be appended to the logs and database. Data will be then displayed in a useful fashion using a dashboard which shows the status of each individual sensor and the sensor history of each machine. Each machine will have at least one sensor which will report at least two metrics: temperature and vibration. UI will also have analytics part

where the user can observe sensor states regarding battery state, time online, and transmitting power.

2.6 Database

Data is stored in a database specialized for storing time-series data. Manufacturers encode the beacons to report their telemetry with subtle but significant differences, which requires different data stores. The database of choice also has to be capable of interacting with non-numeral time-series data. Non-numeral data in this case includes identification information for the beacon, as well as customized fields with data about software version and frame 'topic' which indicates what the endpoint of the data is.

During the prototype phase, the amount of data remains fairly small: 1 minute intervals from a maximum of 53 measurement stations results in roughly 76 thousand rows of information, potentially spread across multiple tables, and 25.2 million rows per year, estimating based on the fact that the machines are stopped for roughly 30 days during one year of operation. This approximates to a maximum of just over 16 megabytes of data each day, and can be considered small. A dataset is generally considered large when it exceeds computer's capacity to place it in a RAM, and some estimates categorize datasets with the size of over 30 Gb as large [3].

2.7 Data Analytics

The goal of analytics in the thesis is to help create model which takes in desired data and outputs accurate predictions about the lifespan or maintenance interval of a piece of industrial machinery. Accurate predictions lead to more streamlined workflow and more efficient maintenance processes. Initially analytics form a model from two different variables, the machine temperature, and its vibration where innate vibrations are excluded. Analytics also include visualization of data and build a interactive dashboard of those visualizations for easy use for those in the field of IT.

2.7.1 Jupyter Notebook and JupyterLab

For the purpose of doing data analysis with python-based applications, the Jupyter Notebook or JupyterLab is a very good tool. Both visually pleasing and powerful, the Jupyter notebook is a python interpreter running in the user's browser. The application's special features include cell-based segmentation of the code. Each cell can be run separately given dependencies are worked out. This feature is both good and bad. Less familiar users might be confused by the non-linear code of the notebooks. Normally, variables in the same program with the same name could not be used, but in Jupyter notebook it is possible, if they are in the different cells.

There are also different type of cells. There are cells for code, text and images. Text cells can take place of comments inside code, while having the added benefit of being more readable. For the purpose of this project, Jupyter notebook was used with Ipython kernel, but it supports kernels for many different programming languages. [4]

JupyterLab is recently released development environment with a file browser, terminals, and a text editor. It supports over 100 programming languages and is widely used in GitHub repositories. [5]

2.7.2 R and RStudio

More popular tools for data analysis include programming language R and its development environment RStudio. Both are built for statistics and graphics, and have wide variety of packages to help user perform the analysis. Similarly to Jupyter Notebook, RStudio also has cell-like structure available where the user can run the desired parts of the code. In RStudio, the user can import their own datasets or they can leverage the R language to mine and manipulate data on their computer. For RStudio to work with databases, the programs only need to import a relevant package. For plots and graphs, R offers a wide range of libraries and functions to work with.

2.7.3 Machine Learning

Machine learning is a modern method of data analysis, where an algorithm makes projections based on the input metrics or features. In the thesis project, machine learning is applied to produce predictions, or labels, which in this case are maintenance intervals and/or lifespan of a machine. Additionally, it might be possible to predict faults from past data to trigger maintenance before the machine becomes irreparable. With the use of labeled examples from the pre-existing test data, the system is trained to create a model to represent how a change in feature affets the predicted label. Later it is planned to take ambient temperature and humidity into account and use them to make more accurate predictions if they are deemed to have an effect.

Regression vs Classification

When creating a machine learning model, it is important to define what it is the user is looking for. Classification problems require a model that predicts discrete values, such as classifying objects in an image or a video as a human, a dog, or a table.

Regression models provide continuous values such as population sizes or housing prices given a set of features.

In the machinery example, the machine wear parameters need to be clearly defined in order to find out which category fits the best. Classification can be used to ask the question: “Does the machine needs maintenance?” and regression can be used to define a continuous metric that can predict values from trends.

Supervised vs Unsupervised Learning

Supervised learning is typically used in classification problems when the developer wants to map known features to known labels, such as low or high amount of smoke particles in air representing safe or unsafe environment, respectively. [6]

Unsupervised learning is used to provide additional insight and explore patterns in the data. For example, clustering algorithms can process population data and find patterns

regarding feature relationships which are not obvious to human observer, not unlike correlation between age and income. [6]

2.7.4 Data Visualization

Sensor data needs to be visualized in order to draw conclusions about correlations and trends. It is also a better method to view data by people not familiar with statistics. There are several tools worth looking for and a number of them are tested with the data gathered during the thesis. There exists a sizeable gap between visualization tools written for either web-based, more lightweight analysis, and more mathematical and statistical analysis tools traditionally used in offline environments. Web-based tools written in javascript tend to work great in the client side, but are unable to do much outside of interactive dashboards or just presenting the information.

Plotly

Plotly is a library that helps the user create graphs, both using their web interface and from code. Plotly library can be used in both python and R. In the browser app, the user can import data and create dozens of different graphs. If the user is keen on creating the graphs in code, it is also possible. After creating functioning code, the app sends data to the plotly web application which draws the graph. The use of the library requires authorization which is given when the user logs into their website. If the user wants to have more than 25 graphs simultaneously, paying the annual subscription fee is required. Example data about rent price per square meter over time in different regions of Finland is shown in the Figure 1.

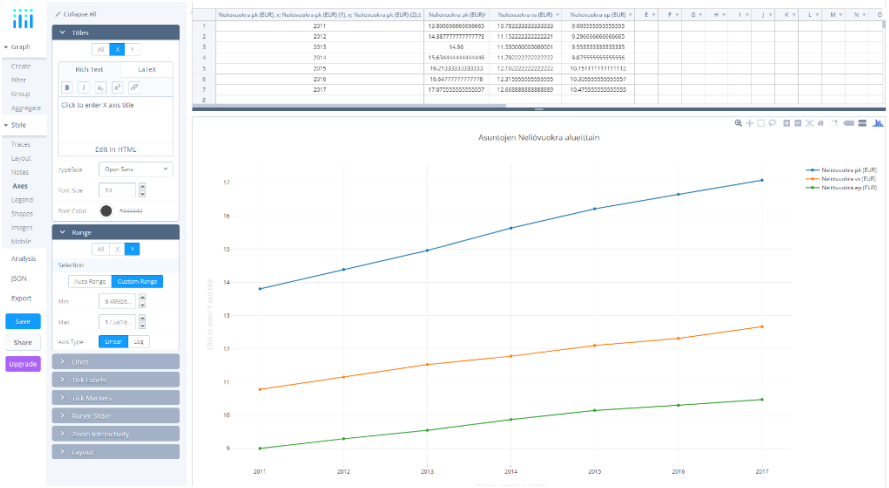


Figure 1. User Interface and drawing of a plot in Plot.ly.

QuickSight

QuickSight is a tool from Amazon that is fairly simple to use. It takes in a number of file formats and outputs from other systems (mostly systems within AWS framework). The user can perform several data manipulations such as mathematical operations, type changes and other more specialized operations provided by the application. After manipulating data, the user can visualize it easily choosing from an array of different plots and graphs. There exist normal graph modifying features such as axis ranges, axis labels and legend. The user can also filter datums from appearing in the graph, perform analyses on the data. Figure 2 shows temperature and power consumption of a machine over time, as lines and bars, respectively.

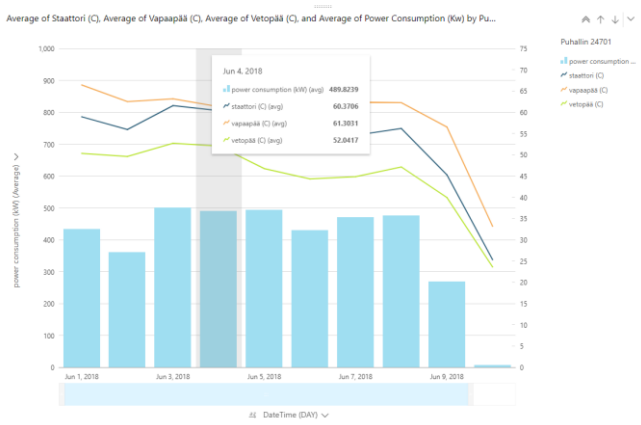


Figure 2. Graph drawn in QuickSight.

Grafana

Grafana is an open source tool for analytics and monitoring. It is used to quickly prototype dashboards. Several different datasources can be used with the grafana platform, such as postgreSQL, mySQL, and influxDB. Grafana has multiple different graph types available, such as line and bar plots, and gauges. One of the key features include integration of different platforms such as Elasticsearch, Cloudwatch, and Prometheus. Other notable features are inclusion of ogranization structure, authentication, alerts, notifications, mixed data sources, annotations, and ad-hoc filters. [7]

Grafana interface is simple to use, the user can find their multiple dashboards easily, manage users, and even has a number of plugins to use. Moreover, the user can choose import dashboards made by other people, or make their own and share or embed their graphs on their own websites or platforms. Picture 3 shows an example of a dashboard created with Grafana platform. The elements in the example are freely customizable in size, colors, their placement, and their representation of data. [7]

Picture 3. Grafana interface.

2.7.5 Data Analytics Summary

After a great deal of research and testing, all of the tested tools and frameworks showed promise that they could be well utilized in presenting the data analysis. However, their use must be examined case by case basis.

Three popular programming languages are currently in large-scale use in the field of analytics: Python, R, and Javascript/Typescript. Each has their own strengths and weaknesses as presented in the Table 1.

Table 1. Comparsion of analytics programming languages.

Python	The most readable syntax, powerful math operations, good performance with large datasets, enables machine learning technology, rapid prototyping, large community
R	Vast analytics libraries, easy use of statistical tests, powerful (but simplistic) graphics
JS/TS	Pretty user interfaces, low performance for large datasets, fast, lightweight, lots of frameworks, asynchronous

There also exists a multitude of visualization tools. The user can choose the most comfortable option depending on their skillset without giving away too much functionality. There exist tools for both presentation of data, and presentation of analysis. The user also has to determine which features are required on a case by case basis. For the thesis project, the choice was to use a web application built on using node.js and grafana for the visualization. Datasets from the customer were received as excel and .csv documents.

3 THE COMPARISON OF DATASETS

3.1 Differences

The first datasets used for the baseline analysis had a number of disparities. The most sparse dataset, which was measured by a technician using a hand-held device had long intervals but proved to be short enough for the machines to perform their tasks.

The next dataset consisted of temperature data from an automated system during the period of ten days. This dataset was easy to visualize and showed how the measurements differ in certain conditions such as normal function vs malfunction or shutdown. However, the data was from such a short interval it was not useful to make accurate conclusions.

The third dataset was the most accurate representation of what the system is going to look like. It shows many different variables and variations in their measurements. However, even this dataset is not without issues. The dataset spans almost three weeks. There are a few machine stops visible, and a single spike in variables before what looks like a shutdown (all sensors drop to zero and temperature drops over time).

3.2 Handling, Cleaning, and Wrangling of Data

3.2.1 Data from the Old Measurement System

The data used in the exploratory analysis was from a few different types of datasets. One dataset consisted of manual measurements done every eight weeks, and other datasets were data from an automated system with the measurement interval of one minute. The latter datasources would still be consireded small, consisting from up to approximately 15 000 rows from a single machine. Few techniques to clean the dataset was required. Handling missing values and isolating the outliers was required to perform further analysis on the data. After some cleaning, it was feasible to do some data wrangling. Wrangling consisted of aggregating the data into smaller subsets, visualizing it, and building statistical model of it for later use in machine learning algorithms.

3.2.2 Data from the Prototype System

The data coming in from the Python application requires some cleaning aswell. There are some extra characters that need to be removed, and the Bluetooth packets have numerous data fields, most of which the prototype will not use or utilize. Therefore, some parsing will be required before Bluetooth packet is sent to the server. After some cleanup operations, the data is sent to a cloud application in the json format, and stored in a database that is specifically tailored to handle time-series data.

As per customer's request, the prototype will utilize two different beacons from two manufacturers. Different beacons are required due to different environments where the data is gathered from. The beacon suitable for less harsh environments only have to deal with temperature and humidity, whereas some beacons may have to reside in areas with dust with metallic particles. Such dust may have an effect on the transmitting capacity of the beacons.

The customer also requested that two types of gateways should be tested: online, and offline. The online gateway will report it's data over the Internet and save it into a database, whereas the offline gateways will save the data into a csv file.

There are at least two types of data to be integrated for the use of the system. The data from an automated measurement system the customer has in place, and the data from the raspberry pi gateways, both online and offline. Data has to be made uniform for the purpose of the analysis, and offline and online gateway data has to be in the same format.

Data coming from the prototype system includes multiple fields, consisting of identifiers, version data, timestamp, and measurements. Picture 2 presented the sending format. The amount and content of the fields depends on the beacon manufacturer and type. All the fields coming from the gateway ideally serve a purpose. Some fields are used to direct the incoming message into its correct destination, and the others are important measurement data to be preserved in a database. Data from the automated system is available as a file and requires a trivial piece of code that parses the information. The beacon data coming to the cloud service in the JSON format will be stored in the database as is and only parsed in the event the data is queried by the user interface component. The user interface will handle the correct visualizations, and the interactive

components related to them. The interactions include things like hovering over the graph for information and specifying the time range.

3.3 Results

Altough the datasets have severe differences, the prototype will test the feasibility of integrating the different technologies in order to refine the use of the measurement information. One of the features of immature big data applications is lack of use cases. The applications gather all kinds of data, but their uses are very limited. Limited use usually comes from the lack of knowledge as to how to take advantage of all the possibilities. As an example, the prototype device will try to create a proof of concept for the industry where things have been done the same way for dozens of years. Only during the emergence of the Internet of Things (IoT) the decision-makers have realized this as a way to update their processes to a more modern state of manufacturing.

Previously, the data was gathered, but only used during the measurement to make somewhat informed evaluations about the state of the machines. With the modern systems, the evaluations can be made remotely, more often, more accurately, and the maintenance crew becomes vastly more efficient as the result.

Other use cases for the data include complete fault prevention and more efficient resource planning. The people monitoring the software can gain information about how much wear each machine is accruing, and order replacement parts before the real issue rises. Data also includes power consumption, and once dataset has matured enough, it can be used to plan more efficient use of electricity.

4 THE EXPLORATORY ANALYSIS

4.1 Tools

AWS QuickSight

Using QuickSight it was fairly quick and easy to draw sensible graphs that contained temperature of various parts and speed of the stator as a function of time in days. Plotting in QuickSight is painless and user can modify their data and see changes to the plot in real time. Data from one of the machines can be seen in Figure 3. Lines are temperatures in different parts and blue bar is the power consumption over the period of ten days.

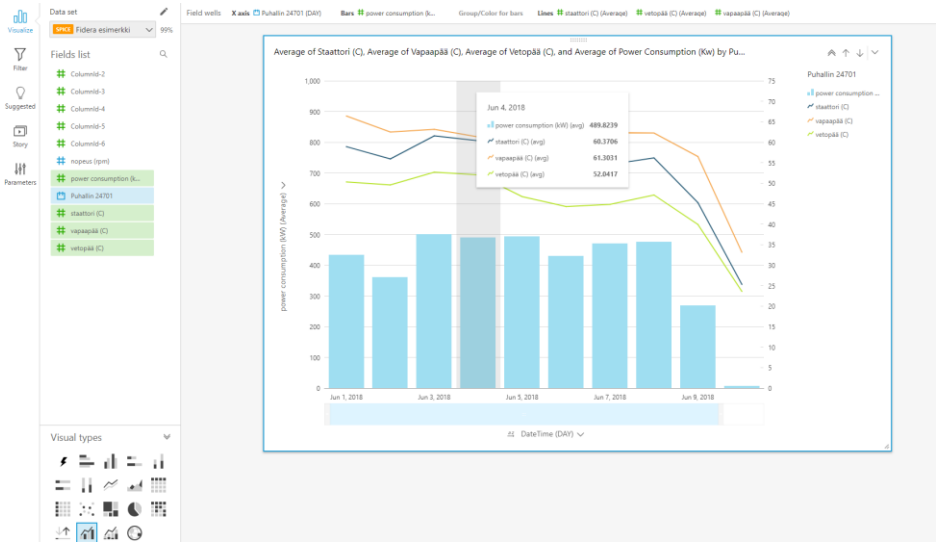


Figure 3. Creating and modifying a graph in QuickSight.

MS Excel

Vibration data was visualized using Excel. There are plenty of data from different pieces of machinery containing the frequency (mm/s) and amplitude (gE) of vibrations. Excel can be used to create pivot tables and charts, making it easy yet powerful tool to manipulate axes and data in the graph. Example graphs drawn in Excel are shown in the Figure 7 and Figure 8.

Python Libraries

When the data is first acquired, a good tool to use to manipulate it for future use is Pandas, a python library. Pandas converts read files into DataFrame type which easily customized to suit needs of the data analyst. After arranging the data to suitable format, it needs to be visualized. Perhaps the oldest, “the grandfather” of plotting in python, the library matplotlib, offers the greatest number of functions and drawing options. It can perform many things, but often plots produced by it are not visually appealing or take too much work due to the library’s extensive API. Fortunately, there are more specialized tools to visualize data you manipulated with matplotlib or some other tool. Libraries such as ggplot (or ggplot2 in R) or Seaborn offer much more compact functionality while still retaining complexity of matplotlib [8]. Figure 4 shows a figure of temperature measurements drawn using matplotlib.

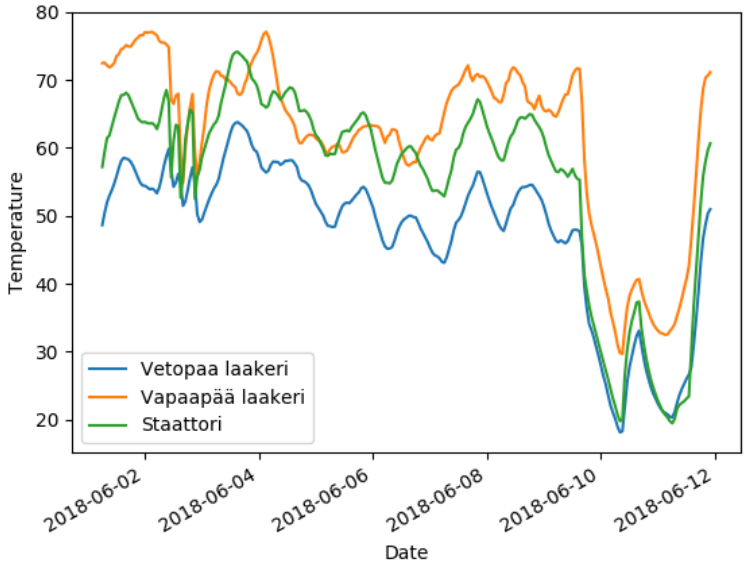


Figure 4. Temperatures in matplotlib.

Then there are a few good tools that are not based on matplotlib, Bokeh, pygal, and Plot.ly. Bokeh is meant for creating web visualizations for modern browsers. Figure 5 is showing a graph about temperature measurements created using Jupyter Notebook and Bokeh.

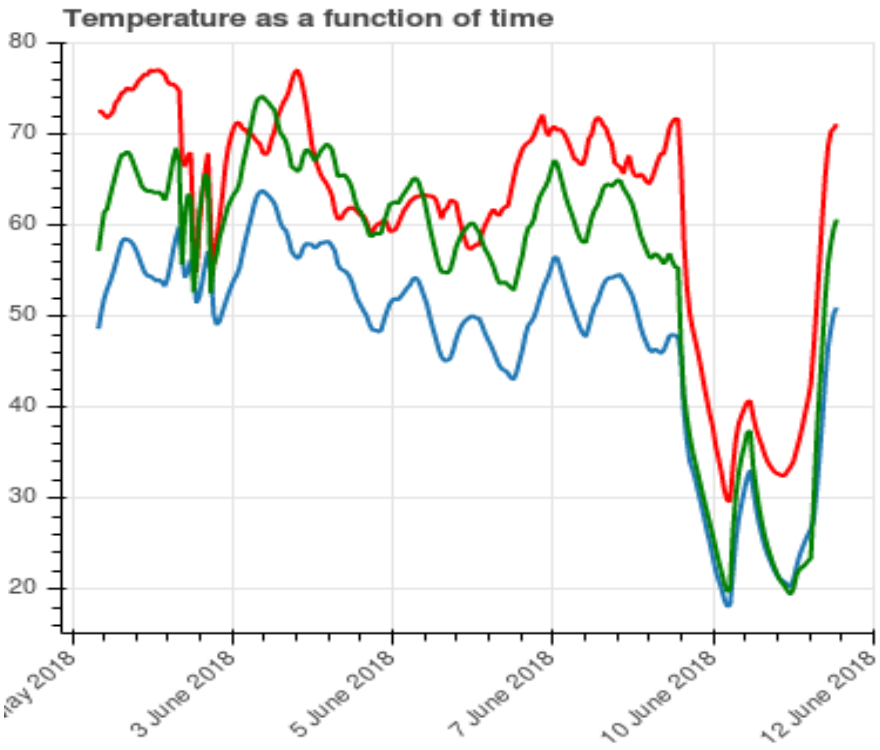


Figure 5. Temperatures using Bokeh.

Pygal is used to create interactive SVG charts. Plot.ly has specialized more into the field of data analysis and has comprehensive online tool for both plotting and analysis. It also should be noted that more advanced features are behind a paid service. One of the newer tools is the one called Altair which creates visualization using Vega-Lite graphics library with minimal amount of code [9]. Altair is a python library that produces a plot in JSON form that is visualized using D3.js in the front-end. It can be leveraged to create interactive charts very easily. These charts can be embedded as html on almost any website.

R and RStudio

RStudio is developed solely for statistics and graphics. Compared to Python tools, cleaning the data in R requires more work. Therefore, a good option would be to work your data to a good format with python and then visualize it using RStudio. Figure 6 shows temperatures in one of the machines drawn in RStudio.

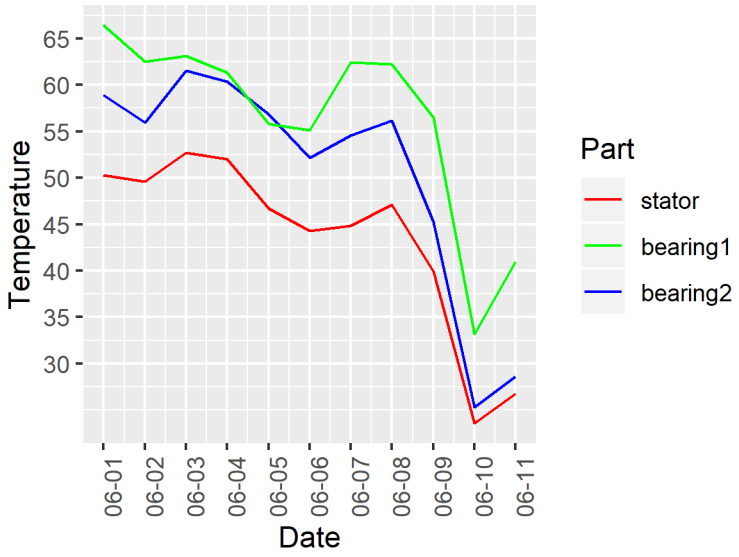


Figure 6. Temperatures drawn in RStudio.

4.2 Process

There are two sets of data to work with. The first is manually measured data with long intervals, but which spans a longer time. The other is data from an automatic system with 1 minute intervals. The latter only has been installed recently and does not span most of the maintenances done on the system. Both datasets require cleaning before they can be visualized. Cleaning the data required removing some cells and transposing before it could be used for a graph. After that only the axes had to be defined correctly to succesfully show peaks in the data. The results are shown in the Figure 7 and Figure 8.

Defining wear of machine from baseline data ends up being inaccurate, thanks to measuring intervals being both inconsistent and long. Measurements took place every ca 1.5 to 4 months and measured values varied a lot on occasion. Vibration measurement consists of two metrics: velocity, measured in millimeter per second, and

acceleration or enveloping acceleration, gE, where 1 gE equals to free fall acceleration. Figure 7 shows velocity measurement data and Figure 8 shows acceleration measurement data from a piece of machinery, the most recent datapoint on the left. Both graphs are made from data measured by manual devices. These measurements had high variance depending on the machine. In such cases, the person responsible has no choice but base their evaluation to the maximum measurement. Lines for M1 and M2 correpsond to motors, and lines for L1 and L2 correspond for their respective bearings.

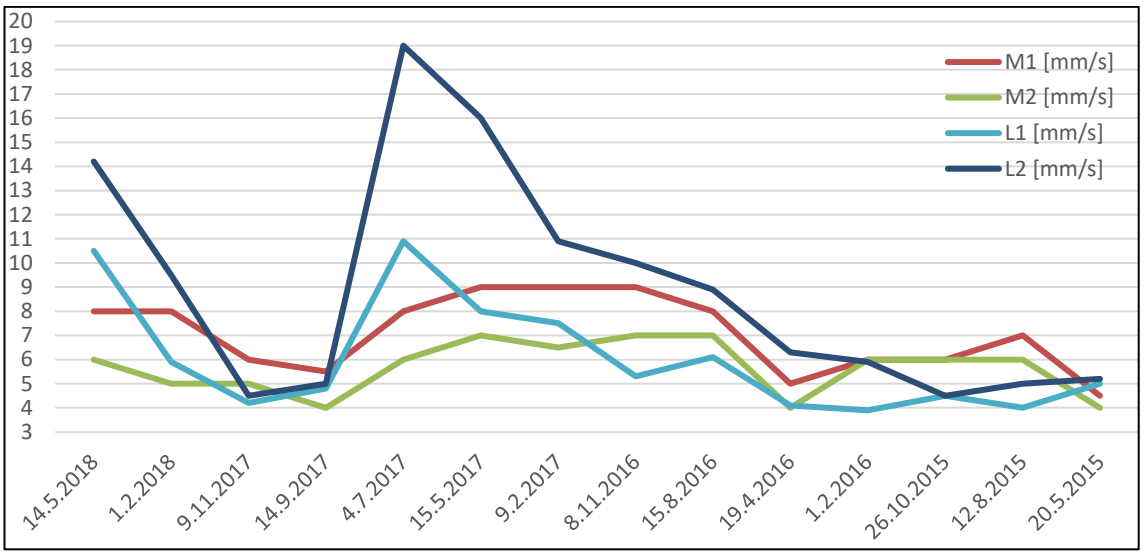


Figure 7. Velocity measurements.

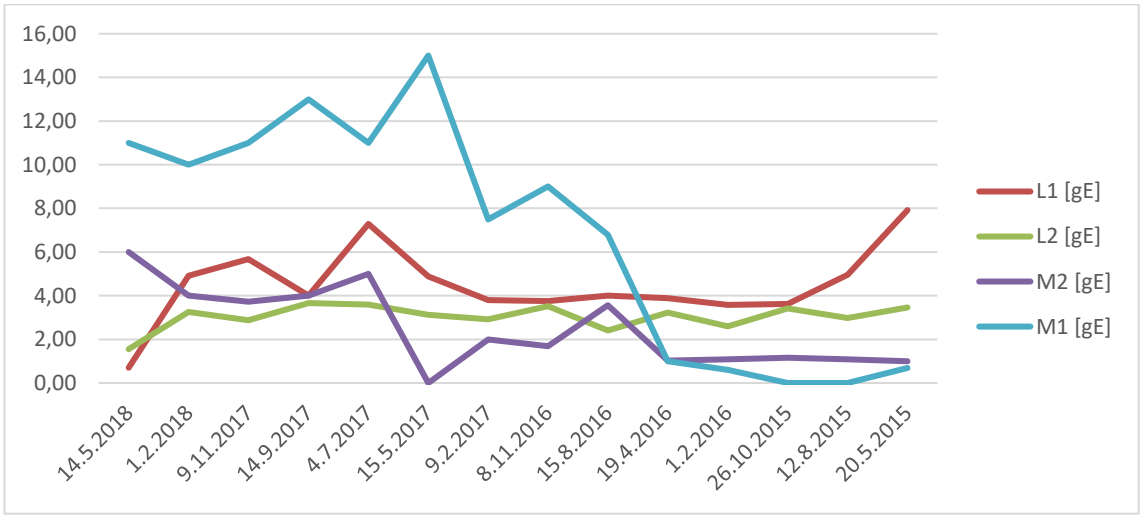


Figure 8. Acceleration measurements.

4.3 Results

From the figures 9 and 10, it is clearly visible when machine starts approaching danger levels. This data was acquired from manual measurements with high intervals, and as such, remains unrealiable. For each machine, each measured part has their own danger levels. For this particular machine, motors 1 and 2 have danger levels of 12 and 6 gE, respectively. Machine bearings can be seen to pass danger levels on both velocity and acceleration. Dangerous velocity levels for the bearings 1 and 2 are 8 mm/s and 15 mm/s, respectively, and dangerous acceleration levels are 4.5 gE and 3.5 gE, respectively.

The other dataset from the automated measurement system is far more optimal for visualization. The data still required cleaning and normalization to be fit into the same graph. After plotting all of them, it resulted in a chaotic Figure 9 where most of the spikes were hidden in the noise. Graph can be made more clear by combining the temperatures and vibrations, and grouping them by their categories.

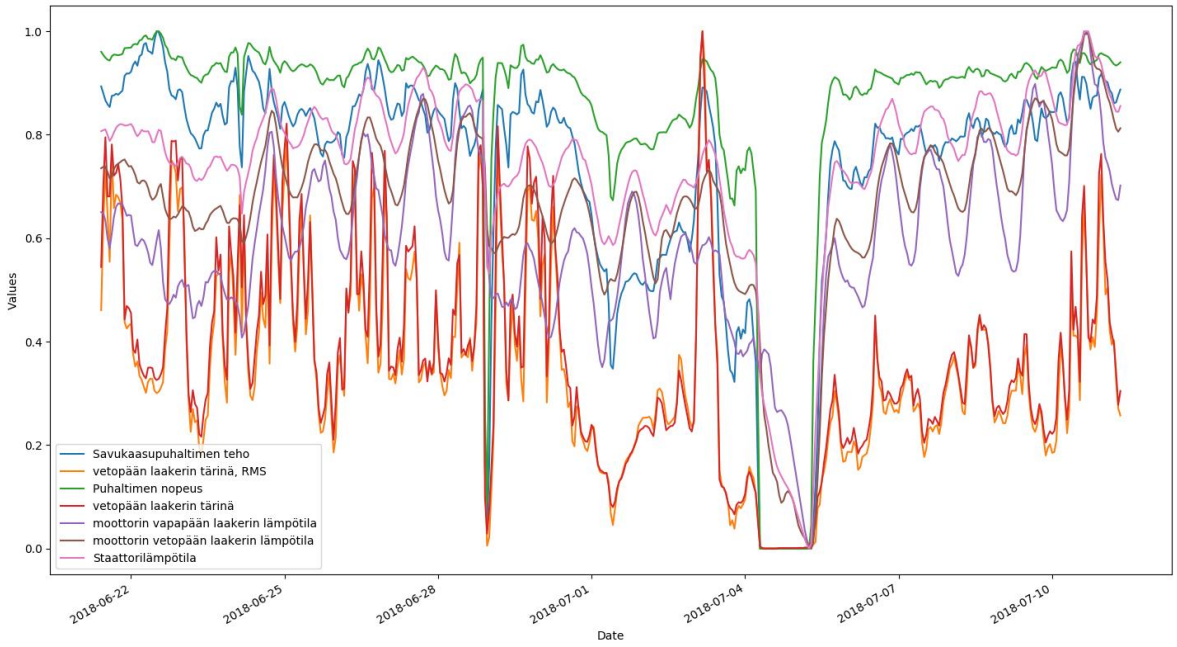


Figure 9. Plot with all the variables.

Even though machine stops were already visible in the Figure 9, they are more clear in the cleaned version, Figure 10. Same as the previous figure, this one is also normalized.

Spikes in the vibration, power consumption and speed of the motor are visible, and are likely the cause for the shutdown resulting soon after. Interestingly, there are also two other shorter stops, with causes unknown. Figure 10 also shows a clear day-night-cycle as a variation in temperature.

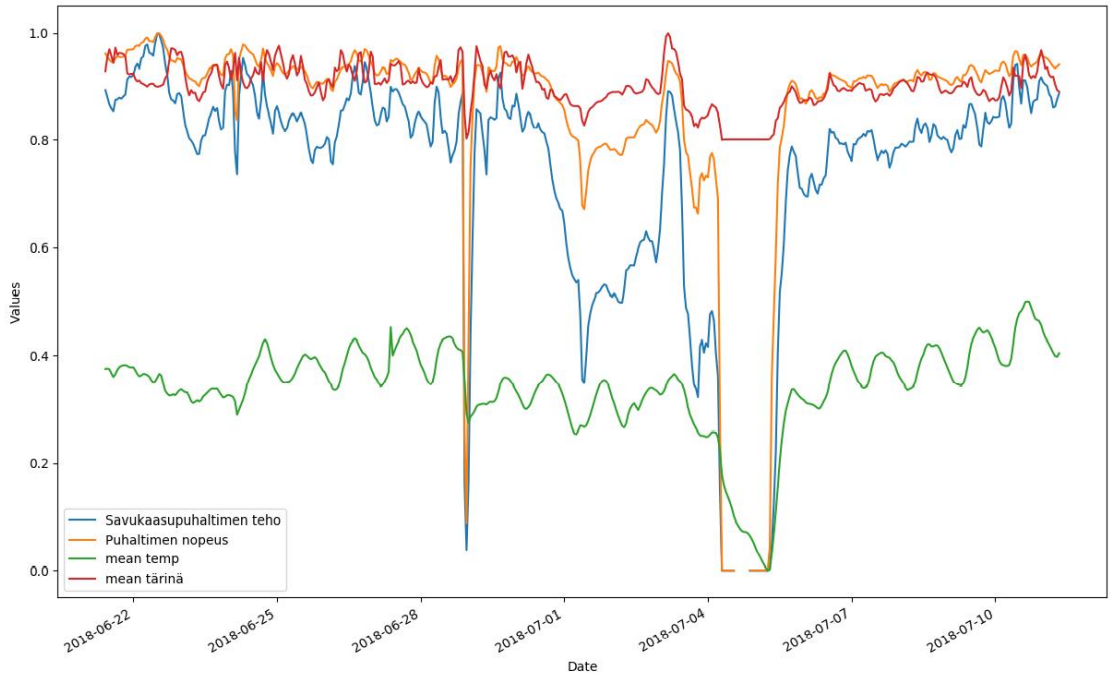


Figure 10. Cleaned data plot.

4.4 Conclusions

It is visible from both temperature and vibration data sets when the machine requires maintenance, but their relation remains elusive. Metric values of the measurements keep increasing to warning and danger levels, and after replacing or servicing certain parts, the levels drop to safe values. In theory, the person responsible for maintaining the machine can view the measurements and make decisions based on the observed metrics. The data also disregards external variables such as humidity and ambient temperature. In the future, when implementing the system, it will more likely get more clear how temperature and vibration relates to one another. After observing and analyzing temperature and vibration data separately, from different machines during different periods of time, the predictive model would be too inaccurate to be of any use.

The dataset gathered by the automated system yields better results. The dataset shows clear correlations between power consumption, motor speed, and motor vibration. All

three can be seen spiking before the shutdown of the machine, and can be concluded to have risen past the limits. Temperature varies over the time of day and correlates with the ambient temperature. The shutdown of the machine visualized in the graph is not likely to have occurred due to temperature alert, altough there is a slight aberration at the of the power consumption/speed/vibration spikes. Usual variation in temperature results in a single peak during a day, and at the time of the malfunction, there are three visible peaks in the graph.

All the different tools to create figures to visualize the same data looks similar, as expected. Python tools are flexible and powerful, but can get complex with more complicated graphs. Excel is also a powerful tool with easy customization using pivot tables. It is also worth noting that using Excel tools require you to have data somewhere, import it as a csv-file for example, and then you are able to visualize it. Similarly, QuickSight can be a useful tool if the data is ready to use, or otherwise exists inside Amazon services such as Amazon RDS. After visualizing data in your database, it is easy to share your analysis with other people within QuickSight. There is no option to export figures but you can export your manipulated data as a csv-file. Learning to use python tools is more useful in the long-term if the aim is to visualize data straight from a database. Especially if desire is to maintain some sort of independency from Amazon's services.

Machine Learning and Making Predictions

One of the more popular methods of machine learning is a method called Gradient Descent. It is used to optimize the model by minimizing the loss function, essentially coming up with the best description of the observed values. One of these gradient descent algorithms is called XGBoost. The algorithm is exremely simple to use and implement. The user imports two datasets: one with labels (the variable user is trying to predict) and one without, a train set and a test set, respectively. The algorithm then makes predictions based on a handful of user-chosen parameters, sometimes called hyperparameters. Finetuning the model usually means trying out different combinations of the hyperparameters and observing the output through different statistical metrics such as root mean squared error, or RMSE.

The acquired datasets can not with enough accuracy describe the system as a whole. Vibration measurements can be predicted using the speed of the motor and its power

consumption with some accuracy. However, predicted values assume that the machine will stay in working condition. This in mind, a case can be made to compare predicted value to observed to detect faults with some accuracy. The best model, based on the sparse dataset of roughly 500 datapoints, still had RMSE of 0.25 which means that the margin of error for predictions in the case of this particular machine is roughly 6-12%. The size ratio between the training and testing sets were roughly 4:1, a popular ratio used in many cases. The other model from a dataset of almost 3000 datapoints resulted in a larger RMSE of 0.50 but it more accurately described the faulty conditions. The resulting vibration predictions are visualized with observed values in the Figure 11. The figure also shows how the choice of the model affects sensitivity to changes in the values. A line plotted from larger dataset does not easily show small changes but exaggerates larger ones. However, these exaggerations can be useful when trying to model the faulty state of the machine. It would appear that the person performing the analysis has to make a concious choice about which case to model, and then depending on the model chosen, either measure difference or similarity to observed values.

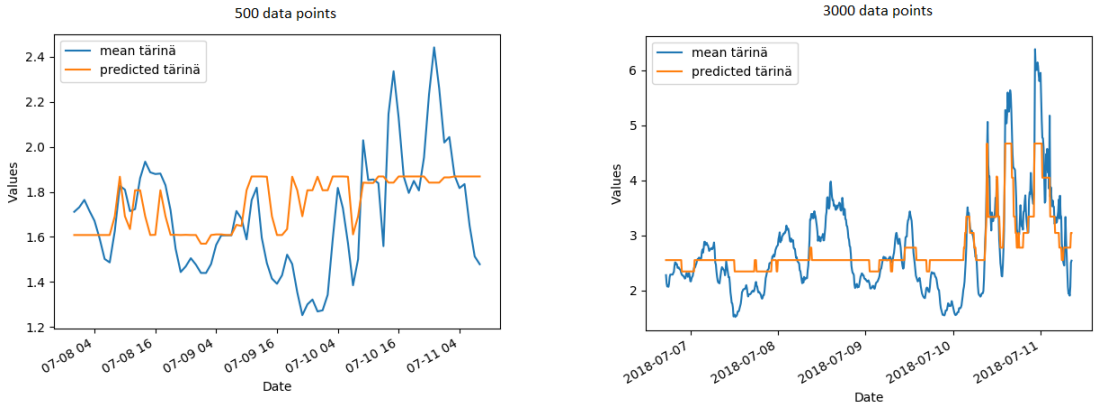


Figure 11. Predictions from different datasets.

5 THE SYSTEM

5.1 System Overview

The planned system has five layers: sensors, sensor gateway, the internet or local network, the server with analytics, and the UI. In the Figure 12, the layers are visualized. It is possible to connect sensor nodes to each other using the recent Bluetooth Mesh specification [2], but for the moment it is neither planned nor required. The gateways will be built to withstand network outages by storing their measurements in a file. Gateways can potentially be connected to each other using a wi-fi mesh network, which is commercially available in most stores. Gateways then pass the data received from the sensors and send it to a cloud application to store it in a database via a virtual private network, or VPN. The cloud application will process the data and performs specified analytics on it. Lastly, the UI component receives the processed data and visualizes it in a useful manner.

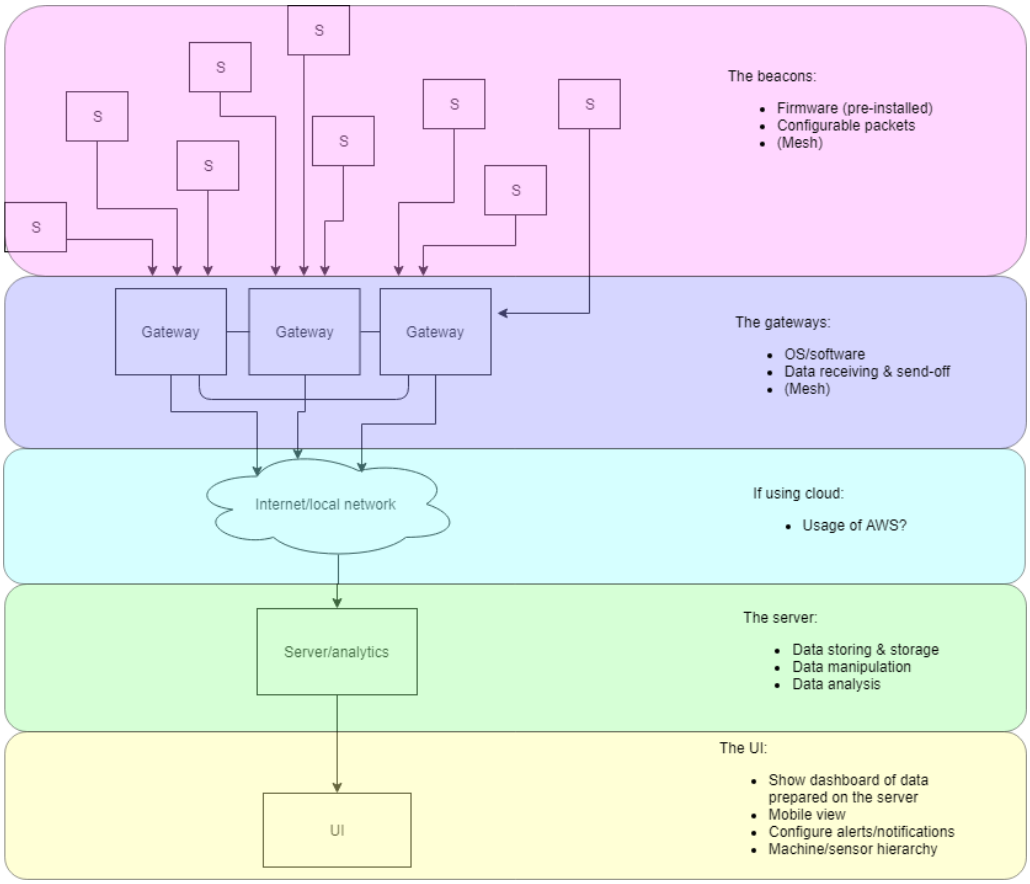


Figure 12. The system overview.

5.2 Software

5.2.1 Bluetooth Beacons

Bluetooth beacons are capable of broadcasting on either Eddystone or iBeacon protocols. The former is developed by google and has no implementation restrictions. Eddystone protocol also contains telemetry frame which can be used to report beacon health and other sensor data. The latter, iBeacon, is developed by Apple. Aforementioned protocol is more focused on location-based data and tracking.

Sent packets can be observed using a smartphone application as shown in the Picture 4. This application can only be used to observe sent packets, it can not be used to configure the beacons in any way. Most beacon manufacturers do not allow changes to the firmware of the beacons. Consequently, configuring the beacons requires a smartphone application provided by the product developer or vendor. Configuration applications often also require NFC connection to the device. After moving the smartphone near the device, these applications can be used to configure a few options such as packet types sent by the beacon, their advertising interval, and the packet contents.

In Picture 4, the packets' contents are also visible. User can view information such as packet type, protocol, device MAC address, IDs, signal strength, battery life, and telemetry information. Each device can send different combinations of packets, and they show up as separate instances in the scanner application or device. Similarly, when different protocol is used, packets are shown In the different instances. In the iBeacon protocol packets, 'major' and 'minor' values are used. These values are required by the IBeacon standard but they do not have to be used necessarily. They can be used to describe the position of of the device in the hierarchy.

Picture 4. Beacon Scanner Smartphone Application.

5.2.2 Bluetooth Network and the Gateway

The operating system in a gateway device is Raspbian Stretch Lite. Stretch is a debian-based operating system and lite is a distribution without the graphical user interface. After succesfully installing the OS, the prototype will require installation of a suitable python distribution and its libraries. Python 2 support will end in 2020, and thus python 3 is chosen.

Initially, Bluetooth connection exists only between a sensor and the gateway the sensor is connected to. The network for the prototype product will be represented using the star toplogy. In the future revisions of the product, it is planned to have every Bluetooth-capable sensor device to be connected in a mesh network to reduce the number of required gateways, increase redundancy against network outages and enable placing of

beacons in hard-to-reach locations. Both topologies are visualized in the Figure 13. In the figure, S represents a sensor beacons and G represents the gateway.

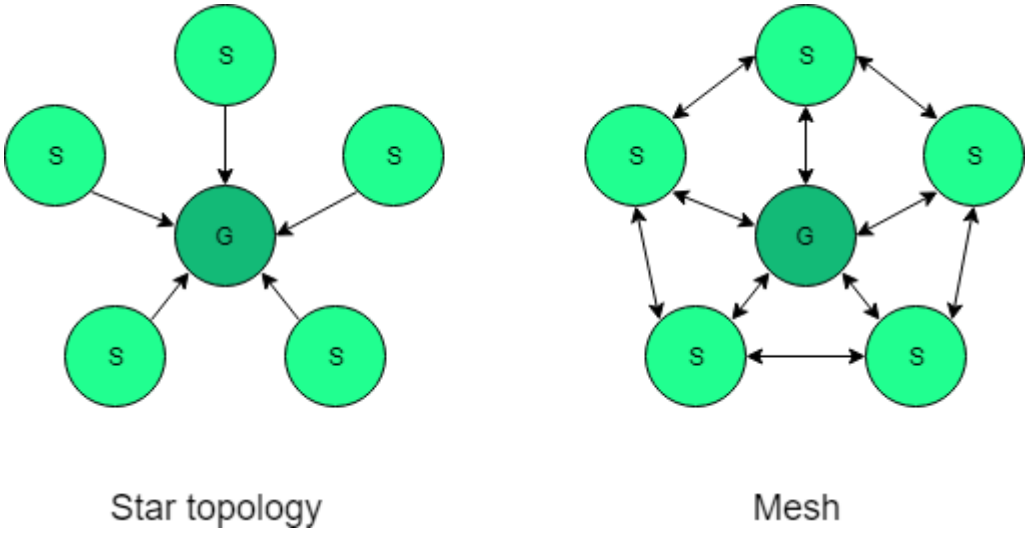


Figure 13. Network topologies.

The gateway device, a Raspberry Pi during the thesis project, will be responsible for sending the data over either a local network or the Internet. Whichever the case, the Bluetooth packet should not be sent forward in the state it arrives in, containing all sorts of unnecessary information. Therefore, the gateway has software that will clean up the data and prepare a right format to send it to the receiver.

5.2.3 Cloud Handling

After the data has been procesed by the gateway, it is sent via an asynchronous messaging service, ZeroMQ, to the cloud service which saves the incoming data into a a database. The cloud application will consist of a number of ‘consumers’. Each consumer is responsible for handling different types of incoming messages, and performing operations on them in order to integrate the data from all types in a single database.

Other cloud functionalities include services that visualize the data in the UI, as well as analytics. Visualizations are done in a web application written in typescript and javascript. Analytics are also visualized, displaying machine states and predicting the lifetime of the machines. Machine states have three different levels. The first level is machine operating

within normal parameters, usually marked as green. The second level is achieved by the machine when it periodically or sporadically reaches unsafe levels, and usually requires further investigaion from the maintenance crew, marked as yellow. Third level is typically when the machine reports abnormal readings from the sensors and can be concluded to be faulty. This level is usually marked with red, and often requires immeadite action or shutdown to prevent wider system faults or damage.

5.2.4 Database

Data coming from the beacons is stored in a database. The database of choice is TimescaleDB, a more specialized version of PostgreSQL, apt at storing time-series data. There are options how to model tables, for example, each table can store certain data types or each table can store user-defined types. It all depends on what the user desires the data to achieve.

In the thesis project, however, there exist a few possible alternatives. One solution would be a database that will have a table for each beacon type, since each manufacturer uses different type of reporting. The other would be to store each incoming message in a table holding JSON objects, and when the data is requested by the UI component, the required fields are read from the JSON. Even the data that might seem redundant at first might have uses later, and is stored in the database.

5.2.5 Data Visualization and Analysis

Data is visualized in an *angular* web application using typescript or javascript, and their related libraries for visualization such as D3.js or Chart.js. Other frameworks include for example Grafana, which can be used for rapid prototyping. Visualizations consist of line, and bar plots, as well as circular gauges. Machine learning components can be embedded into the graphs or reported as numerical values. Machine learning components report predicted lifetime of a machine from each sensor. The person doing the maintenance can then use the UI to display data from other sensor of the same machine and decided whether or not an action is to be taken. Each machine also has specific temperature and vibration thresholds at which the maintenance crew must be notified or alerted. These values have to be customizable from the UI.

5.2.6 Miscellaneous

Other modules include printing software, which listens to ZMQ messages sent from the main code in JSON format. The module is mainly used to print Dymo labels for the beacons. The code is written in a way that each unique beacon gets their label printed only once. This piece of software also creates a file which contains the SensorID, or the mac address, of every scanned beacon in vicinity. The output file is stored as a csv and can be used to, for example, view all beacons of the system or print their labels using other programs, such as DYMO Label. Examples of beacons fitted with their respective hardware identification in the Picture 5.

Picture 5. Beacons with labels.

5.3 Hardware

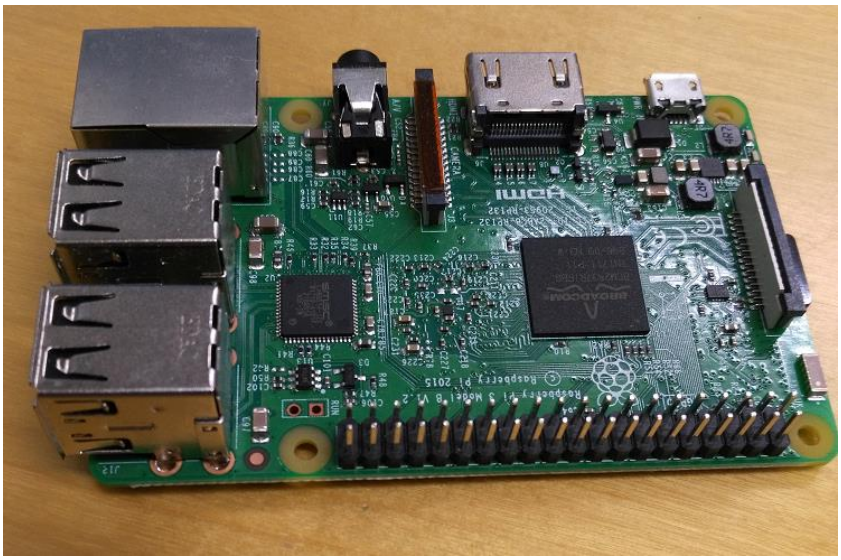
5.3.1 Bluetooth Beacons

There are a number of beacons suitable for the prototype. The most important factors are Bluetooth 4.x capability, and temperature and vibration sensors. Other optional features include Bluetooth 5 capability and extra sensors. The casing might also be important, as the environment is quite warm, humid, and full of chalk dust. Newer beacons are fitted with nRF52832 SOC, which enables Bluetooth 5. Both beacon candidates are also equipped with STMicroelectronics LIS2DH12 accelerometer which is used to report the 3-axial magnitude of acceleration.

The beacon advertisement packet type can be configured using smartphone application. Application can be used to enable beacon to advertise 4 different frame types. Frame types defined in the Eddystone protocol are UID, EID, TLM, and URL [11]. UID frames contains code that allows apps to retrieve information from app servers, such as location, object identification, and app interaction. EID frames contain encrypted identifier to increase security. TLM frames contain telemetry data of the beacon. URL frames contain a redirection to a website secured using SSL.

5.3.2 Beacon Gateway

A gateway is the device that connects beacons to the Internet. The most suitable gateway device is Raspberry Pi 3 B or B+, the former shown in the Picture 6. It has on-board Bluetooth adapter which by default works with Bluetooth 4-enabled beacons. However, there are usb-dongles on the market which can give RasPi Bluetooth 5 capabilities. Using Bluetooth 5 would greatly enhance the functionality, enabling the use of mesh networking. Gateway also collects the data from the beacons, performs small cleaning operations, and converts it to a format that can easily be handled by the cloud application. Gateway device will be highly configurable and will store the sensor data for a period of time as a protection against network outages.



Picture 6. Raspberry Pi 3 Model B.

5.4 Environment

Most of the machinery are located indoors, in a tall, multi-store, cylinder-shaped building. The devices will be located in several floors, separated by both metal and cement. Temperature varies greatly depending on height and proximity to the machinery. Possible variations in conditions such as seasonal changes should be taken into account when creating the model. The machinery can be very sensitive if ambient temperature rises above certain levels. In order to account for that, it is planned to place a few extra beacons somewhere in the vicinity of the measured machinery in the sole purpose of measuring ambient temperature, and use visualization to observe possible correlation, if one exists.

5.5 Testing

Each of the components should be tested prior to the release. Most of the testing is related to the message forwarding. Connection between the beacons and the gateway relies on the strength of the Bluetooth connection and it is the strongest when the devices are within each other's line-of-sight. Each manufacturer's reporting of data over Bluetooth might vary slightly. Therefore, the gateway has to be configured to receive the correct type of packets to know how to process them appropriately. In the optimal case, the system will be operational as long as possible. One of the limitations is the beacon battery life, approximately 3 to 5 years. Power outages are expected, and the devices should restart themselves and related processes when the power resumes.

As the gateway should be able to handle multiple beacons reporting their data continuously without pauses, a proper stress testing is highly important. During initial stress testing, it was discovered that beacons with unexpected firmware and therefore unexpected format of Bluetooth packets caused the system to crash. Consequently, a feature had to be added for the program to be able to handle also unexpected formats in order to keep the system operational.

The real challenge is to test the beacon in the real environment. The temperatures in the field might rise to near 80 °C, and the beacons might experience a loss in in battery life. Other challenges include the Bluetooth connection strength. In the target area, the factory, wireless beacons have to be installed in various sides of the machinery. This can

create difficulties altough the distance between a beacon and a gateway should remain rather short. These characteristics will be heavily examined during the prototype phase.

The next subset of testing is the message queue. When the gateway sends the data over the Internet to the cloud, it is placed in a queue. The testing scenarios include a one where the data is received in a desirable format, and verifying that the data was not changed during the transmission, also known as data integrity testing. After testing the queue, the tests have to include what happens if the cloud receives wrong types of messages.

After testing the messages and their handlers, the testing moves to the database. When the data is stored in a database, it should keep its integrity while it is being fragmented into the correct rows and columns. The database will have different tables depending on the data received. The data packets have different amount of fields depending on the manufacturer, and therefore a different amount of columns are formed from the packets. If the user wants to store every data field, and reserve some for future use, the packets have to be stored in multiple tables.

Last part of the system is the user interface. The testing has to be exhausting here to ensure not only the correct functionality, but also the optimal user experience. The interface will receive data from a database table, and use it in ways that make sense to the user. The data can be in form of graphs, gauges, lists, or a number of other things. It has to be exportable, and it has to make sense. Most importantly, however, the user interface has to make the work of the user more efficient.

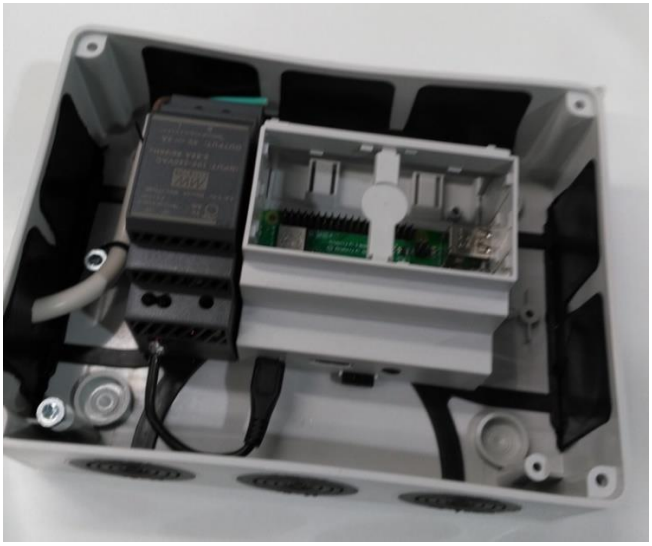
6 THE PROTOTYPE

6.1 Parts

6.1.1 The Beacons and the Gateways

The prototype on site will include the beacons, the gateways, and a 3G-router. The initial version will function without mesh networking, and potentially require a non-trivial amount of extra gateways if the system is adopted as a fully-fledged product and has to be scaled up. If mesh technology was enabled, each beacon could relay their information packet until the packet arrives to a beacon that is connected to a gateway, which would eventually send it to the cloud service. Prototype will also test two different beacon types simultaneously. Beacon type depends on the conditions the beacon has to reside in. Beacons are fitted with labels containing the mac address of a beacon. Labels are printed either using the Dymo Label printer python script or the Dymo Label software. Dymo Label software can take a csv file as an input, and print a single label for each row.

The gateway devices are protected with a casing that holds both the Raspberry Pi, and its power supply. The casing has outlets for the power cord, and the ethernet cable. Depending on the environment, the casing also has to be able to handle humid conditions. Picture 7 displays one of the gateway casings, without the lid.



Picture 7. The gateway casing

6.1.2 The Network

During the prototyping phase, the incoming data has three different sources, and two are related directly to the network architecture. One of the prototype gateways will have access to the Internet, and send the beacon data over Internet to a cloud service. The other gateway devices store the data locally and the data from them is recovered later.

The first data source is the initial data from a time before the prototype system from a different automatic measurement system as a file which is inserted into the database. The second source is the real time measurement data from the the Bluetooth beacons which is sent using a JSON format. The third source is offline gateway data from which the data has to be recovered manually. The data from the first source is in vastly different format compared to the data from the second and the third source, and the customer has raised a desire to see them integrated. In the later stages, with unifrom data, the analysis will become considerably easier as all data in the system is from a single continuum.

6.1.3 The Cloud Services

The cloud part of the system will include a piece of code that listens to incoming messages and forwards parts of it to their correct destinations. The gateway devices will be sending a number of different messages depending on which beacon the data is coming from. The messages are differentiated by adding a topic into the message and sending the message over the Internet via zero message queue protocol. Depending on the topic, the cloud application will perform some cleaning operations after which the data is stored in a database. After the data is stored, it is presented in anUI application using a visualization tool or library such as Graphana in a form of graphs and gauges.

6.1.4 The End-user Interface

The requirements for the UI are as follows: capability to display the customer’s machine hierarchy and data from each measurement station, capability to trigger alerts and notifications depending on the state of measurement beacons. The initial version does not need to include the smartphone application, but the UI could be prepared to be

responsive in order to be usable using a smartphone. If the product is restricted to a local network, a single page application used from a computer will remain usable.

Further development ideas include tracking of assets, contractors, and maintenances using the GUI. Asset tracking can also be accomplished using the Bluetooth beacons, and their ability to measure connection strength to a receiver.

6.2 Installation

6.2.1 The Software installation

The prototype includes 6 Raspberry Pi 3 model b+ single-chip computers which will work as gateways. Five Raspberries are configured to function offline, saving their measurements into a csv file. A single online gateway is planned to demonstrate the functionality of the company's online platform. Each of the gateways will forward data from approximately 10 beacons.

Gateways require some setupping before they can be considered operational. Each gateway is bootstrapped with Raspbian Stretch Lite operating system, which comes without graphical user interface. Gateways will also be installed with Chef. Chef is a configuration manament tool, and it is used to keep software and package versions synchronized with the the server, as well as provide remote capability to write configurations. 11]

The main application is built into a pip package, a special format which can be handled easily by the python package manager, pip. The company has a dedicated server which holds all internal pip packages. Depending on the commit configuration, the package goes to one of two indices: staging, or stable. The staging index is used to hold all the packages currently in development that might receive changes at any time. The stable index houses all the released packages.

Installing the application on a Raspberry Pi automatically converts it into a service, allowing the script not only to run in the background, but also to be stopped, started, and restarted depending on the needs of the user.

6.2.2 The site

The installation of the prototype consists of a few offline gateways and one cloud gateway. Offline gateways are not expected to connect to the VPN, but nevertheless have a capability to send their measurements over the Internet if the connection strength is good enough.

Each gateway has its own group of sensors gathering data from different areas of the plant and from different machines. To help identify each of the beacons, they are outfitted labels describing their hardware identification: the MAC address of the beacon. Each MAC address is tied to one of the machines and each machine might have multiple measurement areas, such as bearings, motors, or gear boxes, and each area has to be measured individually. The beacons do not possess the ability to rotate or randomize their MAC addresses, and therefore is a reasonable way to identify them with. Before installation, each beacon's mac address is ascertained and their place in the machine hierarchy established. The user can display the machine hierarchy in the UI.

The gateway is located in the same general area with the beacons, and is placed in a protective casing. Beacon casing prevents high temperatures and humidity from damaging the electronics, and some similar casing has to be considered for the gateways. Furthermore, there is a large number of metallic pipes and machinery between each beacon and gateway. After some testing, all but one gateway had online access. After installations, each gateway has the ability to discover the beacons in its vicinity, and the functionality can be used to keep track of the beacons that for some reason are not heard by the gateways.

6.3 Differences to the Original or Final System Design

The prototype will be using normal BLE networking instead of mesh networking meaning the the device topology will be similar to the star topology shown in Figure 13. This will potentially increase the number of gateways required. The customer also expressed desire to have multiple types of beacons included in the prototype phase. Each gateway can be configured to either receive packets from a single manufacturer, or receive packets without discrimination, where the latter is a slower but more comprehensive method.

Originally the design included only cloud-enabled devices, with the intention of sending the data over the Internet to a database. However, in the prototype phase, the customer has voiced their desire to also test a version functioning offline. Presenting its own set of challenges, the offline version would not only require extra care if the device required updating, but also require the stored data to be gathered manually. The offline devices also have to have large enough storage to store measurement data.

The prototype will include a label printing software as an extra module. The same module also includes an event listener which forwards ZMQ messages. The online gateway will also use Ngrok software to provide secure tunneling over the Internet.

6.4 Conditions

The prototype will function in a cylinder-like structure tens of meters high. The beacons will be scattered all over the structure with enough gateways to cover the required areas. The temperature in the structure will be higher than the normal ambient temperature. Potential effects on the lifetime of the devices are unknown at this stage. Some coarse dust particles may exist in the environment the prototype resides in, and thus, beacon casing needs to be considered or protective measures applied. The environment is also full of chalk dust which accumulates on horizontal surfaces. Many of the installed beacons fit this category but it remains unknown whether or not the dust actually affects the measurements.

6.5 The Results

Installing the beacons was quite simple. Each of the beacons was attached to their corresponding piece of machinery using epoxy glue. The prototype also included five gateways, and despite all of them not having access to the Internet, gather data. The ones without Internet access save the data to a file, and are ready to send the data over the internet if they ever come to contact with a properly configured wi-fi router. After installing the beacons and placing the gateways into source of electricity, the data started to flow into the system, ready to be displayed. Surprisingly, a total of five gateways were able to gain online access. Further work will determine how many beacons are heard by the gateways.

Python turned out to be an excellent choice to write maintainable code, and due to its readable syntax, other developers can easily continue or branch the development. Thanks to Python's natural modularity, it would be easy to continue to refactor the code into smaller pieces, and expand the functionality to other wireless technologies, and tailor similar systems to needs of different fields of industry.

Raspberry Pi was a good choice to develop the system on. It is very cost-effective at what it does, even though it might be too powerful for this kind of system. As Raspberry Pi has room for the program to expand, some AI or analytics can be developed as future features.

Beacons form the nervous system of the prototype, and are the most interesting part to test. Their temperature data is simple to gather, but vibration data would need some refinement in order to be useful. Ruuvitags have no such functionality built-in, but their firmware is open source and can be modified freely. Confidex beacon are shipped with some vibration calculations in the firmware, but it remains to be seen if the data they report is useful in the eyes of the customer. Pricewise, Ruuvi is the clear winner, Confidex Vikings, while more durable, cost twice the money when compared to Ruuvitags.

In summary, the prototype was a very good test run for this kind of environmental tracking. The system may be more useful in monitoring more generic conditions such as temperature and humidity, as well as tracking whether or not the beacon has people in its proximity. The industrial plant has very specific needs, and the commercial beacons might need more specialized boards and more customized cases to report more accurate and useful data.

7 CONCLUSION

The goal of thesis was to apply and deepen the knowledge gained throughout the studies, as well as to gain an introduction to industrial IoT technology and data analytics. During the studies, the experience gained from areas such as math, physics, programming, electronics, project management helped to progress the project immensely. Altough the programming languge used during the project, Python, was not familiar from the studies, it was easy to learn and debug thanks to its clear syntax.

An important factor for the prototype was to determine several things. Important things include but not limited to: suitability for the industrial environment, suitability of the beacons and accuracy of their measurements, internet connectivity within the plant, and beacon connectivity to gateway devices via Bluetooth. All of these items were tested over the course of several days after the installations, and longer following will try to determine lifetime of different parts of the system. The prototype includes 6 gateway devices, of which 1 was expected to gain internet access, and surprisingly, 5 were in the range of the VPN-router. Positively surprised, the cloud application was flooded with great amount of data, but fortunately the amount of data remained small enough to be easy to work with. At the time of the installations it was unclear whether or not all of the installed beacons were heard by the gateways, and it was to be determined by the future work. Measurements were accurate enough, altough vibration measurements required refinement from acceleration into actual vibration. Data refining would be done in the cloud application.

Another goal of thesis was to plan and implement a working prototype of an embedded data gathering system using Bluetooth and Raspberry Pi. The prototype will act as a proof of concept, and will be utilized by both the company which commissioned the thesis and its customer. The prototype was installed in the one of customer's manufacturing plants towards the end of the thesis project. After the project ends the prototype devices will be left to the plant to gather data for future use. The installation of the prototype system was a quick process. A three-person team installed 50 beacons and 5 gateways during a single workday. Nevertheless, this is a small amount when compared to a plant with 1500 pieces of machinery, and every machine is a potential datasource if the prototype is deemed fit and the customer wants to scale it upwards.

There are many areas which could utilize a similar system, and monitor the conditions of certain interior or exterior areas, personnel, machinery, or other resources. Thanks to its simple installation process, the system would be very fast to implement and the potential customer would gain the use of the system in a very short time.

Many IoT applications in the business-to-business world are about making the process more efficient and providing added value to different processes. The prototype developed during the thesis is no different. The aim of the prototype was to provide the customer with improved efficiency and gain a better control over the resources at their disposal. Towards the end of the thesis project, the UI elements for displaying the data were unfinished, but the data gathering was implemented, tested, and installed. The customer was positively surprised at how fast the installation process was and mentioned how easy the system would be to scale up if the project was continued after the proof of concept phase.

All in all, the project revealed many software development and business world work methods and techniques that would not be found in the student world. During the project, good programming practices, version control, testing, documentation, and communication skills proved to be invaluable. Software development also requires a person who is fast to adapt to new desires of a customer, and a mindset to efficiently find answers to problems that rise throughout the development process. Throughout the project, the gained work experience may prove invaluable in the future, and it will be interesting to observe the development of the IoT applications, industrial or otherwise.

REFERENCES

[1] Bluetooth SIG. (2016). *Bluetooth Core Specification 5.0.* [pdf] Available at: https://www.bluetooth.com/specifications/bluetooth-core-specification [Accessed 29 Oct. 2018].

[2] Bluetooth SIG. (2017). *Bluetooth Mesh Specification 1.0*. [Online] Available at: https://www.bluetooth.com/specifications/mesh-specifications [Accessed 29 October 2018]

[3] Bluetooth.com, (2018) *Bluetooth Topology Options*. [Online] Available at: https://www.Bluetooth.com/Bluetooth-technology/topology-options [Accessed 22 October 2018]

[4] Pafka, S.(2015). *Big RAM is eating big data – Size of datasets used for analytics.* [Blog] datascience.la Available at: http://datascience.la/big-ram-is-eating-big-data-size-of-datasets-used-for-analytics/ [Accessed 7 September 2018]

[5] Jupyter/IPython Notebook Quick Start Guide, (2015). *What is the Jupyter Notebook?* [Online] Available at: http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html [Accessed 4 July 2018]

[6] Project Jupyter, (2018). *JupyterLab is ready for users.* [Online] Available at: https://blog.jupyter.org/jupyterlab-is-ready-for-users-5a6f039b8906 [Accessed 31 July 2018]

[7] Soni, D. (2018). *Supervised vs Unsupervised Learning* [Blog] Towardsdatascience Available at: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d [Accessed 27 June 2018]

[8] Grafana.com (2018). *Grafana – The open platform for analytics and monitoring.* [Online] Available at: https://grafana.com/ [Accessed 30 September 2018]

[9] Moffitt, C. (2015). *Overview of Python Visualization tools.* [Blog] Available at: http://pbpython.com/visualization-tools-1.html [Accessed 29 June 2018]

[10] Moffitt, C. (2016) *Introduction to Data Visualization with Altair.* [Blog] Available at: http://pbpython.com/altair-intro.html [Accessed 29 June 2018]

[11] Amadeo, R. (2018). Meet Google's "Eddystone" - a flexible, open source iBeacon fighter, *Ars Technica* [Online] Available at: https://en.wikipedia.org/wiki/Eddystone_(Google) [Accessed 10 July 2018]