IIoT data collection for OEE measurements
Ásgeir Halldórsson

Final Report in Electrical Engineering B.Sc.
2016

Name: Ásgeir Halldórsson
Kennitala: 041283–2549
Supervisor: Baldur Þorgilsson

School of Science and Engineering
Tækni- og verkfræðideild
**Heiti verkefnis:**
IloT data collection for OEE measurements

<table>
<thead>
<tr>
<th>Námsbraut:</th>
<th>Tegund verkefnis:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Önn: Námskeið:</th>
<th>Ágrip:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016–2 RT LOK 1012</td>
<td>The aim of this project was to compile a list of requirements for connecting industrial computers and devices to collect data and send to the cloud. Furthermore design and build a prototype that fulfills all requirements. The data can be analyzed to improve Overall Equipment Effectiveness(OEE) and help companies make more efficient and productive use of their production methods. There are many types of communication protocols between devices. Tests were concluded on different methods to fulfill the requirements defined for the system. The prototypes made fulfilled all the requirements, users can connect devices to a gateway that sends the data to the cloud where data can be analyzed. The results show that by collecting data from production, companies can improve their OEE and increase profits.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Höfundur:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ásgeir Halldórsson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Umsjónarkennari:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baldur Porgilsson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Leiðbeinandi:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haraldur Porkelsson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fyrirtæki/stofnun:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sámey ehf. Lyngás 13 210 Garðabæ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dagsetning:</th>
<th>Lykilorð íslensk:</th>
<th>Lykilorð ensk:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.12.2016</td>
<td>lóntölur, Tölusamskipti, Gagnasafnskerfi</td>
<td>IloT, M2M, OEE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dreifing:</th>
</tr>
</thead>
<tbody>
<tr>
<td>opin X lokud</td>
</tr>
</tbody>
</table>
The undersigned hereby grants permission to the Reykjavík University Library to reproduce single copies of this Final Report entitled **IIoT data collection for OEE measurements** and to lend or sell such copies for private, scholarly or scientific research purposes only. The author reserves all other publication and other rights in association with the copyright in the Final Report, and except as herein before provided, neither the Final Report nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author’s prior written permission.

__________________________

date

__________________________
Ásgeir Halldórsson  
Bachelor of Science
I dedicate this to my mom.
# Contents

## Contents

List of Figures xi

List of Abbreviations xiii

1 Introduction 1

1.1 Introduction 1

1.2 Background 2

1.2.1 IIoT 2

1.2.2 OEE 3

1.3 Requirements for the system 5

1.4 Structure of thesis 6

2 The design of the solutions 7

2.1 Unitronics PLC/HMI - Cloud 8

2.2 Unitronics PLC - Red Lion HMI - FTP Server 11

2.3 Future research suggestions 12

2.3.1 Data dashboard 12

2.3.2 Arduino 12

2.3.3 dweet.io 12

2.3.4 Machine learning 12

3 Testing and Usage 13

3.1 Testing 13

3.1.1 PLC to IoT Gateway 13

3.1.2 Red Lion HMI 14

3.2 Usage 15

3.3 Use case I 16

3.4 Use Case II 17

4 Results and discussion 19

4.1 Benefits of OEE 19

4.2 Data handling 20

4.3 Collecting Data 21

4.4 Present Data 21

5 Discussion 23

5.1 Conclusion 23

5.2 Future work 23
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Internet of Things[1]</td>
</tr>
<tr>
<td>1.2</td>
<td>IIoT flow[5]</td>
</tr>
<tr>
<td>1.3</td>
<td>Example of IIoT data flow</td>
</tr>
<tr>
<td>1.4</td>
<td>Flow chart proposal to measure OEE in a production line</td>
</tr>
<tr>
<td>1.5</td>
<td>Example of visual presentation of OEE measurement results made in Google Spreadsheet</td>
</tr>
<tr>
<td>2.1</td>
<td>Flow chart of design parameter</td>
</tr>
<tr>
<td>2.2</td>
<td>The 7 layer OSI model[8]</td>
</tr>
<tr>
<td>2.3</td>
<td>Outlook of the Visilogic software application with a MODBUS configuration function block</td>
</tr>
<tr>
<td>2.4</td>
<td>Node-Red configured as a Master to read from the PLC</td>
</tr>
<tr>
<td>2.5</td>
<td>Visilogic Ladder configured to send a HTTP GET request</td>
</tr>
<tr>
<td>2.6</td>
<td>Communication protocol between PLC and HMI defined in the Crimson</td>
</tr>
<tr>
<td>2.7</td>
<td>Machine learning design flow</td>
</tr>
<tr>
<td>3.1</td>
<td>Example of the testing flow in Node-Red reading holding registers in PLC over MODBUS and send it to Google Spreadsheet</td>
</tr>
<tr>
<td>3.2</td>
<td>MODBUS communication between Master and Slave</td>
</tr>
<tr>
<td>3.3</td>
<td>Node-Red flow reading MODBUS TCP/IP and sending it to the Microsoft Azure IoT Hub</td>
</tr>
<tr>
<td>3.4</td>
<td>FTP server with data from HMI rolling in</td>
</tr>
<tr>
<td>3.5</td>
<td>Google Spreadsheet showing data imported, going from the HMI gateway device to the FTP server where the Spreadsheet imports it from</td>
</tr>
<tr>
<td>3.6</td>
<td>Flow chart proposal to measure OEE in a production line</td>
</tr>
<tr>
<td>4.1</td>
<td>OEE measurement of low wine output measured for 46 weeks</td>
</tr>
<tr>
<td>4.2</td>
<td>Flow of the prototype design idea</td>
</tr>
<tr>
<td>4.3</td>
<td>oauth 1.0a authorization diagram, source [18]</td>
</tr>
<tr>
<td>A.1</td>
<td>Set up MODBUS function block in PLC, define the start of vector</td>
</tr>
<tr>
<td>A.2</td>
<td>Define MODBUS between RPI and PLC and make sure the address of vector is same as in PLC and IP address</td>
</tr>
<tr>
<td>A.3</td>
<td>Node-Red setup of Azure IoT Hub</td>
</tr>
<tr>
<td>A.4</td>
<td>Communication protocol between PLC and HMI defined in the Crimson</td>
</tr>
<tr>
<td>A.5</td>
<td>Red Lion Crimson application for data collection</td>
</tr>
<tr>
<td>A.6</td>
<td>Crimson software sync manager connection with ftp server</td>
</tr>
<tr>
<td>A.7</td>
<td>Data logging in Crimson application</td>
</tr>
</tbody>
</table>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADU</td>
<td>Application Data Unit</td>
</tr>
<tr>
<td>AMQP</td>
<td>Advanced Message Queuing Protocol</td>
</tr>
<tr>
<td>API</td>
<td>Application programming Interface</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>CoAP</td>
<td>Constrained Application Protocol</td>
</tr>
<tr>
<td>DDS</td>
<td>Data Design System</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>HDLC</td>
<td>High level Data Link Control</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interface</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IIoT</td>
<td>Industrial Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MB</td>
<td>MODBUS Protocol</td>
</tr>
<tr>
<td>MQTT</td>
<td>MQ Telemetry Transport</td>
</tr>
<tr>
<td>OPC</td>
<td>Open Platform Communications</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable logic computer</td>
</tr>
<tr>
<td>REST</td>
<td>Representational state transfer</td>
</tr>
<tr>
<td>SDU</td>
<td>Service Data Unit</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Internet protocol suite</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>XMPP</td>
<td>Extensible Messaging and Presence Protocol</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Introduction

This B.Sc final project in Electrical Engineering at the University of Reykjavik was done in co-operation with Samey, a local company that has been selling automation solutions for 25 years.

With the increased amount of devices that are able to communicate with each other over the Internet, companies are now able to make more efficient and productive use of their production methods. By being able to receive data that has never been accessible before where they can get feedback from their production and the market instantly, offers endless possibilities in business models and companies service offerings. Today, many factories are run by computers that have mastered the art of productivity, by using data available to maximize the production using pattern recognition and machine learning, also known as artificial intelligence. However, up to 85% of devices are in isolated systems or unconnected, and replacing them all costs a lot of money and takes a long time. To be able to utilize all the benefits of smart systems, there must be a way to connect them.

The goal of this project is to find a way to collect data from industrial devices and send it to a cloud based server where the data can be analyzed and to measure the Overall Equipment Effectiveness (OEE). The collection of the data is the most difficult task in OEE measurements as there are many types of communication protocols and network constrictions that need to be solved. Furthermore, sending the data straight to the cloud is often tricky because of security reasons. Therefore, there needs to be some kind of a gateway on the edge of the network which needs to be implemented in a secure way so it won’t be vulnerable to cyber-attacks. Once the data has been collected in the cloud, it can be analyzed and used to develop concepts with the possibility to transform businesses.

The results are shown in a step-by-step manual in Appendix A for anyone who wishes to copy or get ideas on how to connect devices, collect data and send it to the cloud. Testing will be completed on a couple of different types of devices and applications and other methods are discussed.
1.2 Background

1.2.1 IIoT

Since the beginning of the industrial revolution, the manufacturing industry has been evolving rapidly. With the help of automation, companies have been able to increase their production and operational effectiveness. Samey has been doing research on the overall equipment effectiveness (OEE) with the objective to provide those measurements along with their automation systems. This application would collect data from the system, namely availability, performance and quality rate. This data would be sent to a server in the cloud where the data can be analyzed and measure the OEE factor.

Automation in manufacturing is getting more common and is often referred to as the Industry 4.0 or smart factory [3], where the physical world of the atoms is connected to the cyber-physical world of bytes. As competition in manufacturing is constantly growing, manufacturers are always trying to increase their productivity. By measuring the OEE, they would have the tool they need to increase their productivity. As OEE is not about complex mathematical equations, but rather about collecting simple data during operation and analyzing the trends. Simple analysis like comparing productivity between days, time of day or shifts can give production managers a better overview in order to make adjustments.

The "Internet of Things" (IoT) also called "Machine to Machine" (M2M) is one of the biggest trends in technology today. Microprocessors can be found in all kinds of "things", household electronics, cars, smart watches and almost anything you can imagine. Add Internet connectivity and you have a way of connecting it to an Internet service. According to McKinsey Global Institute, the estimated number of IoT smart devices will be 26-30 billion in 2020. As a result of greater efficiency, it could save 36 trillion dollars in operating costs in affected industries [4]. However, although the benefits are known, the true value is not realized until most devices have been connected. How to use that data, is then for the user to decide. There are no limitations on how to use the data as it could be used to effectively improve the quality of almost everything. Making use of the data is easy, finding a way to transfer that data is more difficult.

The "Industrial Internet of Things" (IIoT) is in a way where the IoT concept comes from. The IIoT is a true alphabet soup of technology acronyms. OPC UA, HTTP, REST, JSON, MQTT, CoAP, DDS, AMQP, and the list goes on. The bottom structure is based on connecting sensors, networks, services and applications together. Today most manufacturing companies are implementing it into their production to collect data and to increase their effectiveness. There is still many companies that have yet to realize the benefits of this. The biggest problem with IIoT today is not connecting the sensors; it is how the data is collected. The problem with industrial computers is that they are often not very accessible for communication to the outside of the factory world. There needs to be a gateway that reads the data and send it to the cloud. For visual explanation, figure1.3 shows where the gateway is used to connect the isolated control system to the cloud.

![IIoT flow diagram]

Figure 1.2: IIoT flow[5]
These data collection is something every company and household should try to get familiar with today. The benefits of collecting data are obvious. There are almost no limitations to what can be measured. The only limit perhaps being the imagination. However, when companies want to use the data, they need to know what type of information they need. This project is not going to answer that question. Every company is different, therefore, there is no one answer for each type of industry. This project aims to research and implement some of the options available to collect data from manufacturing and make the data available for analytics and future development, like pattern recognition and machine learning. For this project it is

1.2.2 OEE

In 1988, a book was published by Seiichi Nakajima about the concept of the total productive maintenance (TPM) [6] which provided the quantitative metric called overall equipment effectiveness (OEE) to measure productivity of individual equipment in a factory. The author stated that the objective is to eliminate the six big losses.

Downtime losses

1. Breakdown losses categorised as time losses when productivity is reduced, and quantity losses caused by defective products.

2. Set-up and adjustment losses result from downtime and defective products that occur when production of one item ends and the equipment is adjusted to meet the requirements of another item.

Speed losses

3. Idling and minor stoppage losses occur when production is interrupted by temporary malfunction or when a machine is idling.

4. Reduced speed losses refer to the difference between equipment design speed and actual operating speed.
CHAPTER 1. INTRODUCTION

Quality losses

5. Quality defects and rework are losses in quality caused by the malfunctioning of the production equipment.

6. Start-up losses are yield losses that occur during the early stages of production, from machine start-up to stabilisation.

The OEE calculations takes these 6 big losses and gathers them into one number that represents the effective operating rate of the equipment or a production line. It measures key factors in production, availability, performance and quality rate to find the percentage of time the equipment or line is operating effectively. This percentage shows the comparison between the produced product and what could have been produced. It is calculated as follows:

\[
OEE = A \times P \times Q
\]  

\[
\text{AvailabilityRate}(A) = \frac{\text{OperatingTime(hrs)}}{\text{LoadingTime(hrs)}} \times 100
\]  

\[
\text{PerformanceEfficiency}(P) = \frac{\text{TheoreticalCycleTime(hrs)} \times \text{ActualOutput(Units)}}{\text{OperatingTime(hrs)}}
\]  

\[
\text{QualityRate}(Q) = \frac{\text{TotalProduction} - \text{DefectAmount}}{\text{TotalProduction(units)}} \times 100
\]

The availability rate is the measured total time the system is not operating. The reason could be breakdown, adjustments or other stoppages. It calculated as the ratio between actual operating time and the planned time available. The performance rate is the ratio of actual operating speed of the equipment and the ideal speed. The quality rate only looks at quality losses. Therefore, it is perhaps the most tricky part in the OEE. There could be multiple reasons for losses which may be hard to identify. Hence, it should be kept simple in order to make calculations easier. However, what isn’t measured can’t be improved, therefore, it needs to be looked at by production managers. What data should be collected depends on the type and how many variables. When the process is really complex, it would be optimal to break down the process into groups and measure the data as illustrated in figure 3.6.

Figure 1.4: Flow chart proposal to measure OEE in a production line
OEE measurements can be as simple as ON or OFF data. Therefore, the data representation
could be as easy to display as shown in figure 1.5 where each box represents one hour. Green
for production ON, and red for OFF. Further evaluation might not be needed to measure OEE.
Hence, a simple device might be used without having to connect to the production equipment.

Figure 1.5: Example of visual presentation of OEE measurement results made in Google
Spreadsheet.

1.3 Requirements for the system

Functional Requirements (FR) and Design Parameters (DP) strategy.

FR1 Read data from device

DP1 Set up communication protocol on device to enable connection to the Internet or a Gateway device

FR2 Decide what data to collect and time interval

DP2 Set up server application that can send/receive data from IoT device

FR3 Send the data to cloud

DP3 Set up cloud database

FR4 Sort the data for analysis

DP4 Design a program to read the data and forward it to the right place
1.4 Structure of thesis

The thesis is organized as follows: Chapter 2 describes the design process for the prototypes that were made and also talks about design suggestions for future design projects. Chapter 3 presents the results of testing the prototypes and discusses the usage opportunities the solution has. Chapter 4 discusses the results of these solutions. Chapter 5 concludes this thesis and Appendix A presents programming code used, installation guide of the solutions and other related material.
Chapter 2

The design of the solutions

In this chapter the design process behind different methods of data transfer is reviewed and the different aspects which can be seen in the design flow in figure 2.1. Data is collected in a local area network and sent to a cloud based server where the data can be analyzed.

The original plan was to send data straight from the PLC to the cloud. The problem with the Unitronics PLC is lack of communication protocols. Therefore, an alternate solution was needed. The solution included an IoT gateway device. It is sometimes used to provide the connection and translation between devices and the cloud. A gateway device has many advantages, such as:

- Translate different protocols and other interoperability tasks
- Condense data so it can be sent to the cloud over a single link
• Store the data in a local database if constant connection to the cloud is not available or needed

• Provide real time clock, with battery backup, to provide accurate timestamp for devices.

As stated in Chapter 1 this project was supported by Samey which offers a wide selection of automation solution. In addition, the industrial computers from Unitronics and Red Lion were supplied. As a result, this project focuses on solutions with the use of these computers as data source. This chapter lists different types of combinations that were tested, followed by the description of the design process behind some of the prototypes that were made to meet the requirements discussed in chapter 1.3 in addition to some suggestions for future development. The setup process for these prototypes is explained in more details in Appendix A.

2.1 Unitronics PLC/HMI - Cloud

Unitronics makes HMI PLCs that are very compact and have a touchscreen, multiple I/O options, are easy to use, and are sold at affordable price. The software application for it is free and easy to use, called Visilogic. It is programmed by using Ladder logic and is used to develop the PLC/HMI application in one environment. It can configure hardware and communication over Ethernet. Ladder is based on boolean principals and follows IEC 1131-3 conventions[7].

The Unitronics V350-35-T2 model that was used for this prototype has an Ethernet port and can communicate with MODBUS over TCP/IP[9]. MODBUS is an application layer messaging protocol positioned at level 7 of the OSI model see figure 2.2.

MODBUS TCP/IP is a request/reply protocol and consists of an application data unit (ADU) and a protocol data unit (PDU). The data is stored in 2-byte registers. To request data from the device over Modbus TCP/IP, a software called Node-Red was used, a tool for wiring together hardware devices, APIs, and online services in a very creative way. It’s a browser based flow editor that is built on Node.js and therefore can run on a cloud based server. Nodes are wired together to make a flow, each node is a JavaScript function so it can be programmed to do multiple things. As seen in figure 2.4 Node-Red was configured to use the Read Holding Register function and read contiguous blocks of registers. The maximum size of a Modbus protocol data unit is 253 bytes. In the request, 3 bytes are used by a function code (1 byte) and the starting register address (2 bytes). Two bytes are used for the amount of registers to read, therefore the maximum is 125 16-bit registers. The response consists of 1 byte for the function code, 1 byte for count and the remaining for the values needed to be read out[10]. In this design feature the MODBUS information data is passed to TCP where it adds information and gives it to IP. The IP protocol places the data in a packet and transmits it. TCP must establish a connection between the master(client in MODBUS TCP/IP) and the slave(server). The slave waits for information from the master and once a connection is established, data can be transferred. One packet consists of an Ethernet...
header (14 bytes), IP header (20 bytes) and TCP header (20 bytes), which encapsulate the Modbus ADU (8 + x bytes) and the Ethernet checksum (4 bytes). Therefore, each packet has a size of 14+20+20+8+x+4=66+x bytes. For the "Read Holding Register" request, the packet size with x=4 bytes is 70 bytes. The response depends on the amount of requested values N. With x=1+2·N bytes, the response packet has a size of 67+2·N bytes. The ACK is an empty TCP packet with some activated option flags and has a size of 58 bytes for header overhead + 6 padding bytes, due to the fact, that every Ethernet frame must be at least 64 bytes. Based on these numbers, the request-response block is of size 70+2·64+67+2·N=265+2·N bytes, which is transferred each second per Modbus device[11].

Figure 2.3: Outlook of the Visilogic software application with a MODBUS configuration function block

As seen in figure 2.3, the configuration asks for Ethernet card initialization and a MODBUS function block where the Ethernet socket and port are defined. By default socket 2 communicates over TCP over port 502 which is configured for MODBUS communication.

The Node-Red application can be installed on all types of computer operating systems that can run JavaScript. There is a MODBUS TCP library for Node-Red which is easy to use. The debug node shows the registers coming from the PLC see figure 2.4.

Another option for this PLC, is to send a HTTP GET request locally. The Node-Red can also be configured for HTTP GET to request values for various time periods or request live values which are the last current values in the PLC memory. The GET URL for a request consists of some project parameters and the request parameters, such as device ID and the desired data. For each measurement type, the device ID and data are appended to the URL. And because the desired data is mapped to different string values instead of simple small numbers, the URL size fluctuates and increases depending on the data size. The response HTTP header has a fixed
size of 127 bytes and the HTTP body depends on the amount of requested N data values. The response contains data in a JSON format which includes a time stamp, device ID and requested data value. The size of the HTTP body can be approximated: \( x = 40 + N \cdot 31 + N \cdot 38 \) bytes with 40-byte JSON overhead, N·31 bytes for the measurements, and N·38 bytes for the timestamps. Therefore, the HTTP response has a size of 167+N·31+N·38 bytes. With Ethernet, IP Header, etc., the whole packet size is 225+N·31+N·38 bytes. For a full request-response block, the size is 485+28·N/3+69·N bytes\(^1\). The HTTP GET request function in Visilogic can be seen in figure 2.5 below.

When the communication is established with the PLC, the JSON string\(^1\) can be split and values appended individually and forwarded in the Node-Red application. The Node-Red offers a wide selection of data forwarding options. It is possible to send the data to Google Spreadsheet by making a HTTP request to Google Form and in the Spreadsheet make a subscription to those Form Responses. Then you have a timestamped data rolling automatically into Spreadsheet where it can be analyzed. Other possibility tested was Node-Red to send the data to a service from Microsoft called Azure IoT Hub. It stores the data in the cloud where it

---

\(^{1}\) Visilogic Ladder configured to send a HTTP GET request.
can be viewed in Microsoft Power BI, data analytics tool. Set up process of the prototype is in instruction manual in Appendix A.

2.2 Unitronics PLC - Red Lion HMI - FTP Server

Crimson software application is used to configure the HMI operator made by Red Lion. The HMI can communicate with 300 protocols[14], it is used to interpret and collect the data sent from each device over Ethernet or other protocols. For this prototype the HMI was connected to 4 Unitronics PLCs on the same LAN over Ethernet. Therefore, the HMI acts as a IoT gateway device as described in the beginning of this chapter.

The Red Lion Crimson application has a built in driver called UNITRONICS PCOM ASCII TCP/IP MASTER. It allows the Red Lion to access all the data values, in the PLCs as seen in figure A.5.

The Crimson IDE offers a "Sync Manager" where data connection to FTP Server is established and the data logs are sent to the server. In the Data Logger function, there is an option to set up a data log and set which integers from the connected devices should be logged to the data logger. There is also options to choose the rate of measurements and how many each file holds. The FTP server is running a php based web host. It reads the CSV file that is sent from the HMI, renames it and gives it a unique URL. In Google Spreadsheet offers users to make their own script that runs in the background of the Spreadsheet. So there is a script running in the background that imports the CSV files from the FTP server automatically at user defined rate.

Figure 2.6: Communication protocol between PLC and HMI defined in the Crimson

The Crimson IDE offers a "Sync Manager" where data connection to FTP Server is established and the data logs are sent to the server. In the Data Logger function, there is an option to set up a data log and set which integers from the connected devices should be logged to the data logger. There is also options to choose the rate of measurements and how many each file holds. The FTP server is running a php based web host. It reads the CSV file that is sent from the HMI, renames it and gives it a unique URL. In Google Spreadsheet offers users to make their own script that runs in the background of the Spreadsheet. So there is a script running in the background that imports the CSV files from the FTP server automatically at user defined rate.
In the Spreadsheet the data can be analyzed for OEE. See instruction manual for this method in Appendix A.

## 2.3 Future research suggestions

Other options that were tested but not implemented through the whole process. In general these are research suggestions for future development.

### 2.3.1 Data dashboard

There needs to be a user interface where they can access the data and analyze them in a simple way. Data dashboard is the key to making this into an appealing product to buy.

### 2.3.2 Arduino

Sensors can be connected to an Arduino to receive data. Furthermore, there is a possibility to attach an Ethernet Shield on the Arduino that allows it to connect to a network. It is based on the Wiznet W5100 ethernet chip [15]. The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP communication. It supports up to four simultaneous socket connections. The Arduino can be connected with sensors and it can be used to send a JSON script as an Ethernet package.

### 2.3.3 dweet.io

Dweeting is a web based RESTful API service that allows data to be sent as a HTTP query. The data source sends data as unique URL, where the slave knows the recipe of the message body and can filter out the data values. The query strings:

```
https://dweet.io/dweet/for/my-thing-name?hello=world
```

is filled out with the unique name of the device and put into the string where it says "my thing name" and after the question mark comes the data string.

### 2.3.4 Machine learning

As the design of the solution evolves and it shows that it is stable and secure. The next step would be to implement machine learning into the design and offer customers to use the data this product has been providing with the goal to make the system more automatic. Computers can learn patterns and take action based on the data to increase the OEE[16].

![Machine learning design flow](image)

Figure 2.7: Machine learning design flow
Chapter 3

Testing and Usage

3.1 Testing

This chapter contains results from testing various combinations of devices.

3.1.1 PLC to IoT Gateway

The MODBUS TCP/IP communication tests between the Unitronics PLC and Node-Red application in the gateway device were successful. Node-Red is easy to customize, it offers various ways to manipulate the data received from the PLC. For example, a JavaScript function can be programmed to do calculations or add anything needed to the data before it forwards it to the cloud. Tests were performed with the use of both a Raspberry PI 3 and a HP 850 G2 laptop PC as the IoT gateway running the Node-Red application successfully. As seen in the Node-Red flow in figure 3.1 below. The first node is the MODBUS TCP/IP communication establishing connection to the PLC and reading the holding registers. Thereafter the incoming data array is passed into a function node that splits the array and identifies each data register whereas the PLC does not. The user knows where the data starts and ends, therefore he knows how the payload is configured and uses the function node to identify the values he wants to be sent. Thereafter, an injection node takes the new payload from the previous function node and passes it into another function node that passes the URL parameters before it gets sent to Google Spreadsheet as a HTTP GET request.

Figure 3.1: Example of the testing flow in Node-Red reading holding registers in PLC over MODBUS and send it to Google Spreadsheet
Wireshark can be used to inspect the MODBUS TCP/IP query/response packets being sent and calculate the overhead latency and the bandwidth need. The size of transferred packets depend on the amount of registers to be read. This transaction can be seen in figure 3.2 where the IP address of PC is 10.2.28.19 and the PLC is 10.2.28.50. Tests were performed on the RU LAN.

![Figure 3.2: MODBUS communication between Master and Slave](image)

Tests were successfully made sending the message payload from the PLC to Node-Red and forwarding it to a cloud server from Azure, see figure 3.3.

![Figure 3.3: Node-Red flow reading MODBUS TCP/IP and sending it to the Microsoft Azure IoT Hub](image)

### 3.1.2 Red Lion HMI

Successful tests were concluded with a Red Lion HMI operator as the gateway device which gathered data from 4 Unitronics PLCs locally. The Crimson software has a built-in data logger and it can send the data as a CSV file to an ftp server.

The data logger was configured to update measurements every minute to the log and log for 60 minutes before sending the CSV file to the FTP server. Figure 3.5 shows the data in Google Spreadsheet, there is a script running that automatically imports the CSV files so they can be analyzed.

In the Google Spreadsheet users can analyze the data and create data tables with calculations and measure OEE for example using the pivot table and graph functions. The script collects all 24 files created per day and puts them into the same sheet so OEE measurements can be made easier for each day. Example of usage can also be seen in Chapter 3.4 below.
3.2 Usage

These applications can be used by anyone with knowledge of computer programming and Ethernet protocols. Hopefully with the help of the user guide in appendix A, anyone interested can use this solution and tailor it to their needs. The gateway device acts as a hub which all kinds of devices can be connected to. The basic idea is to increase efficiency. There is no limit on how to use the data available. However, further development of this application is difficult to answer. In order to get some idea of the capabilities of the application, two use cases will be discussed.
3.3 Use case I

For use case 1, a fish factory in Iceland has a production line where fish is processed, packaged according to type and class. Thereafter the packaging is distributed on a pallet by type or class for shipping. The owner of the factory wants to convert the factory into a "smart factory". With the application, there are multiple options. First, a list of possibilities that are achievable with our solution. For this case, it is assumed that we are using a Raspberry PI with Node-Red.

- Connect to the computer that knows what fresh fish is coming into the factory before it is fed to the production line. Buffer 1
- Connect to the computer at the beginning of the production line to get data about what type or class. Buffer 2
- Connect to the computer that controls the distribution of packaging. Buffer 3

We could perform various OEE measurements with the use the data read by all the computers to see where improvements can be made. For example if these numbers are visible to the workers in the factory, they could see whether there is a difference in load on the production line. Perhaps if there is too much load in the beginning where fish is processed, the workers in the loading department have no product to load, they could be relocated for a period to help out elsewhere which could even out the overall effectiveness of the production system. This system in a very simple visual can be seen in figure 3.6.

![Flow chart proposal to measure OEE in a production line](image)

Figure 3.6: Flow chart proposal to measure OEE in a production line

To measure the OEE factor these are the parameters needed to be found.

1. Availability A
   - To find the availability, see equation 1.2 as we would need operating time and loading time. Operating time is the length of each shift which is usually a constant. The loading time could be a counter sensor at the beginning of the line. Thereafter, calculate the ratio of the theoretical max load it can take

2. Performance efficiency P
3.4 Use Case II

We have a distillery that makes 3 types of wine. The distillation process has 3 phases and 4 machines. 2 machines perform the same phase. All the processes have a couple of different routines, called recipes which they go through depending on what type of product they are making. All the machines are run by a Unitronics PLC. The owner wants to be able to collect information from the PLCs about the distillation processes so that he can analyze the OEE of the process and see whether or not he can make any changes in order to improve the production. To solve this he could choose between either the Node-Red or Red Lion HMI as the gateway. If the HMI is chosen, the display could be enhanced further by allowing staff to input data related to the ongoing process they would be working on. For example, if the process takes longer time than usual, there could be a display on the HMI with a list of variables that are taken into the OEE measurements as the process is running. Hence, if the OEE drops, it would be easy to identify what happened as the input from the staff comes up next to the OEE value. The only downside of this solution is that it only offers users to send the data to an ftp server.

A Raspberry PI could also be used, the RPI could run Node-Red which would communicate with all the PLCs and all kinds of sensors or any related data. It could gather the data to a local database, connect other information to the same Node-Red flow, like sales information for instance. We can monitor the inventory of the shop by setting up function node that keeps track of the inventory of our product. That way the brewer can see from the sales numbers whether sales for some products are increasing more than other wines. With this information, he could decide what to brew at any time depending on the sales trend at each moment. The staff could have an app on their phone where they could input variables related to the OEE measurement. For example, if there were mistakes in packaging or any other data related to why the OEE factor went down. All that data can be stored locally or sent to the cloud.
Chapter 4

Results and discussion

4.1 Benefits of OEE

The real value of collecting data and doing OEE calculations was tested on the use case II in Chapter 3.4 where one sector of the production line was measured. Output from the production was collected at the end of each week. The results can be seen in figure 4.1. However the results were not analyzed until week 40. The results showed the production manager the trend of the output which helped identify the cause of the fluctuations. By using data analytics the production manager was able to make changes and the graph shows that productivity increased.

![Figure 4.1: OEE measurement of low wine output measured for 46 weeks](image)

Studying the numbers further to evaluate the real value after the data collection started can be seen in table below.

<table>
<thead>
<tr>
<th></th>
<th>Week</th>
<th>Year</th>
<th>Value</th>
<th>+Value p.year</th>
<th>+Value p.month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1.680</td>
<td>84.000</td>
<td>168,000,000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Max</td>
<td>2.708</td>
<td>135.400</td>
<td>270,800,000</td>
<td>102,800,000</td>
<td>2,056,000</td>
</tr>
<tr>
<td>Avg last 8</td>
<td>2.110</td>
<td>105.491</td>
<td>210,981.250</td>
<td>42,981.250</td>
<td>859,625</td>
</tr>
</tbody>
</table>

Where the theoretical max output is 3,000L. Average produced 1.680. Max produced is 2.708L in week 45. Average produced last 8 weeks is 2.110L. The target OEE factor is set to...
75% If the market value for each liter is 2,000 ISK,- we can find the value. Finding the added value after production has been analyzed shows an increased income of nearly 43 million ISK per year if the productivity keeps the same pace as last 8 weeks. This is only a small example of how easy it is to quickly increase productivity and profit using data analytics. This could be a great starting point for OEE measurements. Start with simple data like output quantity, work from there to build better understanding of the whole system.

### 4.2 Data handling

The original plan was to send data straight from the PLC to the cloud. Options in the PLC were limited to TCP/IP so there was only the option to send data with a HTTP GET request. However, a problem arose as those requests did not find their way through the networks firewall. In addition the PLC doesn’t support SSH or SSL standard security protocols. Therefore, the problem was solved by setting up a TCP tunnel which is used to bypass network restrictions. Allowing that kind of traffic is not a secure way[17], however, another method had to be used. The next step was to make a gateway device on the edge of the network. These gateways can run applications like Node-Red which can add the security standard to the message. The prototyping was done by using a Raspberry PI and a PC. Although the RPI is not considered to meet industrial standards in some cases, it can be considered perfect for prototyping. Hence, it is a relatively good starting platform for these kind of applications before one starts to use expensive industrial grade applications. The equipment used was able to solve parts of the problem for a fraction of what the cost of an industrial standard one would cost. Therefore this solution can be used as a start up package for companies who are interested in exploring the potential of IIoT, but are perhaps hesitant to do so.

![Figure 4.2: Flow of the prototype design idea](https://developers.google.com/identity/protocols/OAuth2)

The biggest problem with IoT from the beginning are the communication protocols. There are multiple application layer protocols proposed for IoT, such as HTTP, MQTT, AMQP, CoAP, XMPP. HTTP and CoAP are a web oriented methods based on REST. However, they are heavy with a large overhead and therefore not suitable for IoT sending messages frequently. The protocol that is trending the most, is MQTT. It has a very small footprint and needs a middleware called broker that takes care of the publish/subscribe channel. It was designed for constrained devices, like sensor, making it suitable for IoT applications. There are security concerns for the MQTT protocol[18], which causes the developer to be adding the oauth protocol to the solution. Figure 4.3 shows the oauth flow diagram Google for example offer oauth 2.0 to access Google APIs which should be taken a closer look at:

https://developers.google.com/identity/protocols/OAuth2

There are other platforms, such as HiveMQ, OcotBlu, PubNub and Pusher, that provide service that can communicate by using one or many of these protocols, or you can simulate it with
proprietary SDKs. These can be implemented in the Node-Red application. This solution needs to be able to offer secure communication for the clients if it is supposed to offer service in the cloud.

### 4.3 Collecting Data

There are multiple options for cloud services. Before selecting a service, further research is needed in order to assess whether the service would fit the needs required. There are multiple start up companies offering this service. For example:

- [https://temboo.com/](https://temboo.com/)
- [http://www.tableau.com/](http://www.tableau.com/)
- [https://thethings.io/](https://thethings.io/)
- [https://hologram.io/](https://hologram.io/)
- [https://www.tempoiq.com/](https://www.tempoiq.com/)
- [https://www.xively.com/](https://www.xively.com/)

These are just few examples of alternative choices to the big companies such as Google and Microsoft. Amazon Web Services are also offering solutions for IoT devices. If one chooses a service from a start up company, it could present a risk as the service could be canceled with a short notice. In that perspective it would be wise to choose service from the big companies. Google, Microsoft and Amazon are some of the biggest companies on the market for cloud database service.

Tests were made using both the service from Google and Microsoft with success. Once one has a gateway for the data, it is an easy task to store it in the cloud.

### 4.4 Present Data

The objective of this solution is to make the product user friendly and not requiring coding skills to analyze the data. Therefore Google Spreadsheet was primarily used for testing the data dashboard. The main reason being, that it is based on Microsoft Excel which is a program most people are familiar with. With Excel, people can do calculations based on their preferences.
Chapter 5

Discussion

5.1 Conclusion

This thesis hopefully presents some insight into the trending world of IIoT devices and their capabilities. This thesis forms the basis of future product development for an IIoT gateway device for the manufacturing industry. Hopefully this thesis can give some context on how to make something similar, and how to collect data with the objective to provide real time data analysis of OEE. Many of these parts can be used independent from each other. This project touched many different job titles and previously studied material was used to solve difficult problems. OEE is a powerful tool that should be used to drive improvements for users. The target is always changing, be it small or large increments.

5.2 Future work

The demand for this kind of product is growing rapidly as more companies are getting familiar with the term IIoT and Big Data. Further development of this project could entail countless possibilities. As data handling can be considered one of the most complicated areas for further development, a collaboration with a professional IT company could be advantageous for both parties as an IT company could be more familiar with networking protocols and security issues. Raspberry PI is a very solid IoT gateway solution for small projects. Other factors to consider include further development on the data dashboards display, getting a better understanding of the data, and security issues have to be looked at to make sure the product is safe to use.

Perhaps what this as a product lacks most importantly, it needs someone to sell the idea of OEE to potential buyers. This product needs someone to:

- Define and establish effective system for collecting, analyzing and report OEE data in real time.
- Understand the process and establish where the OEE data will be collected and how it can be used to make improvements.
- Establish effective methods to control both systems and process to make sure OEE is integrated for the production

As a future final project suggestion would be either in the field of operations analysis or Big Data analysis for IIoT data.
Bibliography


Appendix A

Instructions manual

A.1 MODBUS TCP/IP in Node-Red

This instruction manual goes through setting up a Raspberry PI and setting up MODBUS TCP/IP communication with a PLC and forward it to Microsoft Azure IoT Hub. Keep in mind that you can use any type of OS as long as it supports JavaScript Node.js. Begin by setting up the RPI OS Raspbian Jessie full version:


Next follow this installation manual for Node.js and Node-Red:

https://nodered.org/docs/hardware/raspberrypi

Type in these commands in the Terminal on the RPI to install the MODBUS node, note that all nodes have to be install into the node-red directory to be available in the flow editor:

```
sudo apt-get install npm
sudo npm install -g npm@2.x
hash -r
cd ~/.node-red
npm install node-red-contrib-modbus
ifconfig (gives you the IP address of the RPI)
```

Figure A.1: Set up MODBUS function block in PLC, define the start of vector
Find the MODBUS READ node in the node panel and put it in the node, give it a name, choose what you want to read. Here we read Holding Registers, set Address at same as start of vector in PLC and how many registers. Choose Polla rate, here it is 5 seconds. And define the PLC Server attributes.

![Figure A.2: Define MODBUS between RPI and PLC and make sure the address of vector is same as in PLC and IP address](image)

There you have established the data read between the Node-Red and PLC and can manipulate the data however you want.

**A.1.1 Node-Red to Google Spreadsheet**

Here is an example on how to send the data to Google Spreadsheet by using Google Form:

Goto drive.google.com and choose New->More->Google Form
Create a question with name of variable being sent and as a text answer
Create a new Spreadsheet for this Form or link it to existing
Answer the question once and it will generate this link: https://docs.google.com/forms/d/e/\(<Your Form ID>/viewform\?entry.\<Your text ID>\
Copy to text editor and we will use it in our Node-Red flow
Insert a HTTP request node from the function panel in Node-Red
Define Method as GET
And from use the URL you copied to text editor and insert ID and payload is data
URL: https://docs.google.com/forms/d/e/\(<Your Form ID>/formResponse\?entry.\<Your text ID key>=\{\{payload.id\}\}
If you want to add data entry just make another question and add:
?entry.\<Your 2 text ID key>=\{\{payload.id-2\}\}
Payload should be a JSON string with question and data integer

**A.1.2 Node-Red to Azure IoT Hub**

Here we send the data to Microsoft Azure IoT Hub. Set up your free account at https://azure.microsoft.com/en-us/services/iot-hub/
In the Raspberry Pi install these and decide which messaging protocol to use. In this example we use mqtt.

```shell
npm install node-red-contrib-azure-iot-hub
npm install azure-iot-device
npm install azure-iot-device-mqtt
npm install https://github.com/lcarli/NodeRedIoTHub.git
```

In your Microsoft dashboard go to your IoT Hub in

General->Shared access policies->iothubowner->Connection string primary key

```plaintext
HostName=.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=<copy-this-key>=
```

copy/paste into the Azure IoT Hub Registry node see in figure A.3

"Register Payload" node is sending the device ID like {
"deviceId": "FinalProj"

Figure A.3: Node-Red setup of Azure IoT Hub

Insert the following into "Send Payload" and select message protocol in "Azure Iot Hub" node see figure A.3.

"Send Payload" node select JSON string and insert

```json
{"deviceId": "FinalProj", "key": "<your key>", "Protocol": "mqtt", "Data": 
	{ <payload JSON string>}}
```

"Azure IoT Hub" insert protocol of choice and your Hostname, make sure your routers port allow traffic. Use:

http://www.canyouseeme.org/

To see if port is open. Then you can define the update rate of the messages to your IoT Hub, it’s free to receive 8k messages per day.

**A.2 PLC - HMI - FTP - Spreadsheet**

Crimson software application is used to configure the HMI operator made by Red Lion[14]. The HMI can communicate with 300 protocols, it is used to interpret and collect the data each device is sending over Ethernet or other protocols. Ethernet was used for this design project.

The Red Lion Crimson application has a built in driver called UNITRONICS PCOM ASCII TCP/IP MASTER, it allows the Red Lion to access all the integers that are being used by the PLC as seen in figure A.5.

The sync manager seen in figure A.6 is where data connection to ftp server is established and the data logs are sent to the server which was a FTP server running a PHP web service.

In the Data Logger function see figure A.7, is an option to set up a data log and set which memory integers should be logged to the data logger. There is also options to choose the update rate and how many measurements each file holds which will be discussed in more details in the
FTP server needs to have the following coding inside .php files in the same folder the dataloggs are going to. The code below is fetching the filename and sorting according to age. Lets call it getfile.php

```php
<?php
/*
function oldest_file($dir)
{
if(is_string($dir))
{
$dir = glob($dir);
}
$oldest = array_shift($dir);
foreach($dir as $file)
{
if(filemtime($file) < filemtime($oldest))
```
{ 
$oldest = $file;
}
return $oldest;
*/
$csv = glob('*.csv');
echo $csv[0];

Then we make another .php file that moves the file once it has renamed it into a folder called done/

<?php
error_reporting(ALL);
$filename = $_REQUEST["file"]; 
echo "Moving file $filename";
In Google Spreadsheet make a new sheet. Goto Tools->Script Editor Make a code with the following functions that will run in the background in the Spreadsheet

```javascript
function ReadFiles() {
  //Reading files until the folder is empty
  while (ReadDataLogs() != false);
}

function ReadDataLogs() {
  var sFile = getFile();
  //If web host returns a filename to work with fetch it
  if (sFile != "") {
    //Data inserted at specific sheet
    FetchCSV(sFile);
    //File is moved to folder /done
    moveFile(sFile);
    //Return true value until all data has be read
    return true;
  }
  //else folder is empty and we exit
  return false;
}

function getFile()
{
  //Calls web host getFile which returns the oldest csv file in the folder
  var options = {
    "method" : "GET",
    "muteHttpExceptions": false
  };
  return result = UrlFetchApp.fetch('http://asgeir.jt.is/eimverk/logs/brugg/getFile.php', options);
}

function moveFile(file_name)
{
  //Call web host to move file to /done
  return result = UrlFetchApp.fetch('http://asgeir.jt.is/eimverk/logs/brugg/moveFile.php?file='+file_name);
}

function FetchCSV(sFile) {
  //Reads csv file from host and appends it to rows and columns
  var stringF = String(sFile);
  var csvUrl = "http://asgeir.jt.is/eimverk/logs/brugg/" + sFile;
  var csvContent = UrlFetchApp.fetch(csvUrl).getContentText();
  var my_ss = SpreadsheetApp.getActiveSpreadsheet();
  //Var that reads date from file
  var my_sheet = "20" + stringF.substr(0,6);
  if (csvContent != "") {
    var csvData = Utilities.parseCsv(csvContent);
    //Creates new sheet for every day if it doesnt exist
    if (my_ss.getSheetByName(my_sheet) == null) {
      //append to rows
      my_ss.getSheetByName(my_sheet).appendRow(csvData);
    }
  }
}
```
my_ss.insertSheet(my_sheet);
}
var sheet = my_ss.getSheetByName(my_sheet);
//Appends values to file according to date that is in timestamp
sheet.getRange(1, 1, csvData.length, csvData[0].length).setValues(csvData);
}
}

In the toolbar is a clock symbol which users can define how frequently the script runs. For this project the trigger was set to 60 minutes. Data is now ready to be analyzed. Security factors were not investigated for this method.

The FTP server is hosted at
Tölvustoð ehf
Hlíðasmára 17
201 Kópavogur
Iceland

www.tolvustod.is
Appendix B

Hardware and software used

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unitronics</td>
<td>V350-35-T2</td>
</tr>
<tr>
<td>Red Lion</td>
<td>G306</td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td>3 Model B V1.2</td>
</tr>
<tr>
<td>HP Elitebook</td>
<td>850 G2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
<th>Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crimson</td>
<td>3.0</td>
<td>683.002</td>
</tr>
<tr>
<td>Visilogic Vision OPC</td>
<td>9.8.31</td>
<td>0</td>
</tr>
<tr>
<td>Visual Studio 2015</td>
<td>14</td>
<td>025431.01</td>
</tr>
<tr>
<td>Wireshark</td>
<td>2.2.2</td>
<td>g775fb08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Raspberry PI</th>
<th>Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linux</td>
<td>4.4.32-v7</td>
</tr>
<tr>
<td></td>
<td>Node.js</td>
<td>5.12.0</td>
</tr>
<tr>
<td></td>
<td>Node-Red</td>
<td>0.15.2</td>
</tr>
</tbody>
</table>