

SIAULIAI UNIVERSITY  
COMPUTER SCIENCE DEPARTMENT

**Lukas Sapiega**

Computer science department

**COMPANY AIR HANDLING UNIT ONLINE MANAGEMENT  
SYSTEM**

**Įmonės vėdinimo įrenginių valdymo sistema internetu**

MASTER THESIS

Advisor:

Prof. Dr. Sigita Turskienė

Reviewer:

Prof. G. Kulvietis

**Šiauliai, 2018**

Šiauliai

2018m.

*„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta, naudojant literatūros sąrašą pateiktus informacinius šaltinius bei savo tyrimų duomenis“*

Darbo autorius \_\_\_\_\_

(vardas, pavardė, parašas)

## Table of Contents

1. Introduction .....	12
2. Important Terms and Definitions (background information).....	14
3. ANALYTICAL PART .....	15
3.1. Current air handling unit management situation.....	15
3.1.1. The company and its technologies .....	15
3.1.2. Company device control system.....	16
3.1.3. Company mobile application analysis .....	17
3.1.4. MB-gateway analysis .....	18
3.1.5. Modbus protocol analysis.....	19
3.1.6. The project requirements .....	20
3.2. Analysis of the area of work .....	20
3.2.7. Internet of things analysis.....	20
3.2.8. Google IoT system .....	21
3.2.9. “IBM Watson” solution .....	22
3.2.10. Cognito Networks solution .....	23
3.2.11. “OJ AIR CLOUD” ahu cloud solution .....	24
3.2.12. “SystemAir” company analysis .....	25
3.3. Analysis of tools for cloud development .....	25
3.3.13. Programming language analysis .....	25
3.3.14. Programming framework/library analysis (UI development).....	27
3.3.15. Client-Server communication protocol analysis.....	28
3.3.16. MySQL and NoSQL database comparison for cloud usage .....	28
4. SOFTWARE DESIGN AND REQUIREMENTS .....	30
4.1. Plan for the implementation of the project .....	30
4.1.1. Project hardware requirements .....	30

4.2.	Initial description of the project.....	31
4.2.2.	MB-gateway data transfer protocol description .....	31
4.2.3.	Execute Modbus command description.....	32
4.2.4.	Get slave list commands description .....	34
4.2.5.	Get params per & command description .....	34
4.2.6.	Write slaves command description.....	35
4.2.7.	User interface use case diagram .....	36
4.2.8.	Registration activity diagram.....	36
4.2.9.	Login activity diagram.....	37
4.2.10.	MB-gateway – server communication sequence diagram .....	38
4.2.11.	Cloud control database schema.....	39
4.2.12.	Cloud data collection and analytics database schema.....	40
4.2.13.	Cloud security requirements description .....	40
4.2.14.	Analytics tools .....	41
4.3.	Choice of tools and techniques .....	41
5.	PROJECT DEVELOPMENT, VERIFICATION AND VALIDATION .....	42
5.1.	Project workflow and schedule.....	42
5.1.1.	Database implementation .....	42
5.1.2.	Extended Modbus protocol implementation.....	42
5.1.3.	Cloud interface development.....	42
5.2.	Description of problems and their solutions .....	43
5.2.4.	Protocol improvements.....	43
5.2.5.	Data management improvements .....	43
5.3.	Software testing .....	44
5.3.6.	Software testing rules .....	44
5.3.7.	Cloud data integrity and command testing.....	44

5.3.8. Temperature history testing.....	45
5.4. Description of the final stage of the project.....	46
6. ANALYSIS OF THE RESULTS.....	47
6.1. Extended Modbus protocol validation and verification.....	47
6.2. Collected data validation and verification.....	47
7. CONCLUSIONS AND FUTURE WORK.....	48
8. REFERENCES AND LITERATURE.....	49
ANNEX 1.....	54
ANNEX 2.....	56
ANNEX 3.....	58

## Table of Figures

Figure 1. Remote Control Options [1].....	15
Figure 2. Current air handling unit control user interface .....	16
Figure 3. “SALDA AIR” mobile application user interface .....	17
Figure 4. MB-gateway [3] .....	18
Figure 5. Mb-gateway connections [3].....	19
Figure 6. Google IoT system [8] .....	22
Figure 7. IOT platform example according to IBM [9].....	22
Figure 8. Cognito IoT solution [10] .....	24
Figure 9. MySQL mongoDB side by side comparison [18].....	29
Figure 10. MB-gateway connection manual [19].....	31
Figure 11. Cloud control use case diagram. ....	36
Figure 12. Account registration activity diagram.....	37
Figure 13. Login activity diagram .....	37
Figure 14. Gateway - Server communication sequence diagram .....	38
Figure 15. Full cloud devices communication diagram .....	38
Figure 16. Cloud control database schema.....	39
Figure 17. Cloud data collection database schema.....	40
Figure 20. Collected temperature data example (data from 25 days).....	45

## List of tables

Table 1. Important terms and their definitions .....	14
Table 2. Programming language scores and types [13].....	26
Table 3. JavaScript framework analysis [15, 16] .....	27
Table 4. Read command example .....	32
Table 5. Write single register example.....	33
Table 6. Write multiple coils command example.....	33
Table 7. Modbus read input and holding registers response .....	34
Table 8. Write slaves explanation part 1. ....	35
Table 9. Write slaves explanation part 2. ....	35
Table 10. Cloud data integrity testing .....	44
Table 11. Air handling units mode control in cloud environment testing.....	45

## **SUMMARY**

### **Company Air Handling Unit Online Management System**

**L. Sapięga**

Currently company air handling units are used for ventilation of buildings and they can only be controlled in local area network (LAN). This does not allow the company customers to control the microclimate of their home in a convenient way (e.g. when they are away from home). Therefore, there is a need for software that allows remote control of air handling units via the Internet.

The review of scientific research and the Modbus documentation found that, without direct access to the “Modbus Master“ device, the remote server cannot communicate via the Internet when the “Modbus Master” device is behind a router.

The software was created to enable users to manage their air handling units in real time from any location that has Internet access. The system collects the data of the air handling unit registers, which can be viewed and analyzed by the customer or the manufacturer. The results of the data analysis can be used to determine air handling unit malfunctions or air temperature changes over time.

After the analysis of the company hardware and software, the requirements for the software were established. From the analysis of programming languages and their frameworks, tools were selected that are suitable for realization of the final software.

A UML model has been created to describe the software being created. A protocol to support air handling unit - server communication when the Modbus master is behind a router was created, which was designed by expanding the Modbus protocol.

From the comparison of developed software and company air handling unit management system data, it was found that the collected register data is sufficiently precise. However, data collection is delayed by 2 to 30 seconds depending on the speed of the internet and the configurable device data transfer frequency.

The developed software for remote control of air handling units was installed in the company for testing (the installation document is enclosed).



The developed software will make it possible to adjust the accuracy of data [20] from different air handling units by collecting data from a large number of customer air handling units.

# SANTRAUKA

## Įmonės vėdinimo įrenginių valdymo sistema internetu

### L. Sapiega

Šiuo metu įmonės oro vėdinimo įrenginiai, naudojami pastatams vėdinti, yra valdomi tik vidiniame kompiuterių tinkle (LAN). Tai neleidžia įmonės klientams patogiai (pvz., išvykus iš namų) kontroliuoti savo namų mikroklimatą. Todėl reikalinga programinė įranga, kuri leistų nuotoliniu būdu, naudojantis internetu, valdyti oro vėdinimo įrenginius.

Iš mokslo tiriamųjų darbų apžvalgos ir „Modbus“ dokumentacijos nustatyta, kad be tiesioginės prieigos prie „Modbus Master“ įrenginio nutolęs serveris negali bendrauti internetu, kai „Modbus Master“ įrenginys yra už maršrutizatoriaus.

Darbe sukurta programinė įranga, kurios reikalavimai nustatyti bendradarbiaujant su SALDA UAB, įgalina vartotojus valdyti savo oro vėdinimo įrenginius realiu laiku iš bet kurios vietos, kurioje yra interneto prieiga. Sistema renka oro vėdinimo įrenginių valdymo registrų duomenis, juos gali peržiūrėti ir analizuoti klientas arba gamintojas. Duomenų analizės rezultatai gali būti naudojami įrenginių gedimams arba orų temperatūrai nustatyti.

Atlikus įmonės turimos techninės bei programinės įrangos analizę, suformuluoti konkretūs reikalavimai kuriamai įrangai. Atlikus programavimo kalbų ir jų karkasų analizę, nustatyti įrankiai, tinkami galutiniam produktui realizuoti.

Sukurtas UML modelis kuriamai programinei įrangai realizuoti. Pateiktas oro vėdinimo įrenginių nuotolinio bendravimo su serveriu protokolas, kuris suprojektuotas praplečiant „Modbus“ protokolą.

Iš atliktų oro vėdinimo įrenginių registrų duomenų vientisumo testų matyti, kad surinkti duomenys yra gana tikslūs, lyginant juos su įmonėje įdiegtos oro vėdinimo sistemos rezultatais. Tačiau duomenų rinkimas vėluoja nuo 2 iki 30 sek. Vėlavimas priklauso nuo interneto spartos ir įrenginio duomenų siuntimo dažnio, kuris yra konfigūruojamas.

Sukurta programinė įranga, leidžianti valdyti oro vėdinimo įrenginius nuotoliniu būdu, įdiegta įmonėje testavimui (pridedamas įdiegimo dokumentas).

Sukurta programinė įranga leis patikslinti skirtingų oro vėdinimo įrenginių duomenų tikslumą [20], surinkus duomenis iš daug klientų, turimų oro vėdinimo įrenginių.

# 1. Introduction

Air handling units are devices used to ventilate and supply clean air for buildings.

Firstly, the objective of the of air handling unit manufacturer is to collect data about efficient performance of air handling units. Currently it is an impossible task. Collecting air handling unit data would allow the air handling unit manufacturer to identify problems with reliable operation of air handling units before they occur and, consequently, provide better customer support.

Secondly, customers are not able to control air handling units over the Internet as an air handling unit can only be controlled from home LAN (local area network). Ensuring remote access to the control of air handling units via the Internet would provide customers with better accessibility to their devices while at work or out traveling.

To sum up, the objective of this master thesis is to create a system that allows clients to control air handling units, collect, manage and analyze the collected data in a cloud environment, as well as enables customers or company employees to perform air handling unit diagnostics over the Internet.

**Project objective:** Using computer system technologies create a system that allows users to connect, gather data and control their air handling units using the Internet. Accessing the devices has to be possible from any location where the Internet connection is available. The connection has to be secure while collected air handling unit data must have previewing possibility for analysis purposes.

**Project name:** Company air handling unit online management system.

## **Project tasks:**

1. Analyze the topic and establish primary air handling unit control and data management system requirements;
2. Analyze of how controlling remote devices work (using IoT) and what other companies have to offer;
3. Prepare the requirements for hardware, software and programming languages needed to be able to meet the system requirements.
4. Plan and design the software, establish device-server communication diagram, basic user interface implementation and database schemas.

5. Develop the software and describe the problems encountered in the course of the project development and specify improvements that have been achieved.
6. Validate and verify the final project.
7. Analyze the results indicating how the collected data can be useful in air handling unit manufacturing and maintenance.
8. Write basic software user interface manual.

**Project scientific methods:** research, observation, experiments, result analysis.

## 2. Important Terms and Definitions (background information)

*Table 1. Important terms and their definitions*

<b>Term</b>	<b>Definition</b>
IoT	Internet of things
LAN	Local area network
MAC address	Media access control address
API	Application programming interface
M2M	Machine to machine – same function as IoT
MQTT	MQ Telemetry transport protocol. It Is designed for M2M or IoT
HTTP	Hyper link transfer protocol
HTTPS	Hyper link transfer protocol secure
SSE	Server-Sent Events
XHR	XML http request
API	Application programming interface
DOM	Document object model
UML	Unified modeling language
BPMN	Business project model notation
AES	Advanced Encryption Standard

### 3. ANALYTICAL PART

#### 3.1.Current air handling unit management situation

##### 3.1.1. The company and its technologies

The company that manufactures air handling units is “SALDA UAB”.

Air handling units are controlled by two control boards (MCB and PRV), while separate devices use the control boards to control the air handling unit. A potential workable device list to control air handling units [1] is as follows:

1. “MB-gateway” – a network module;
2. “Ptouch” – controller for air handling unit;
3. “FLEX” – controller for air handling unit;
4. “Stouch” – controller for air handling unit;
5. “SALDA AIR” – a mobile application.

All of the above air handling unit control devices and software can only control air handling units in local area network, unless the client forwards ports of his router to be able to access MB-gateway remotely.

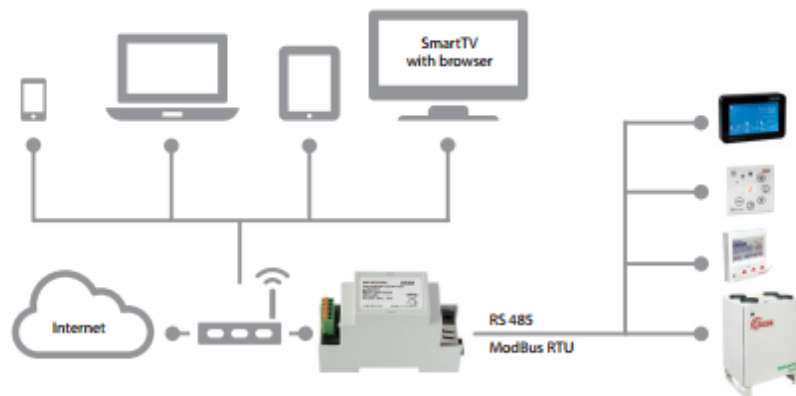


Figure 1. Remote Control Options [1]

In summary, there are many ways to control the air handling unit but they cannot be controlled over the Internet without port forwarding. This problem will have to be solved, allowing clients to easily control and monitor their air handling units over the Internet.

### 3.1.2. Company device control system

Air handling unit control is maintained by using the browser in LAN (local area network). A client can access MB-gateway local area network address which will load the Company's air handling unit control interface supporting the following functions:

1. Control air handling units;
2. View current air handling unit temperature, humidity and pressure values;
3. Setup scheduler for when to change air handling units mode;
4. View alarms and warnings.

The air handling unit web control system works by sending XMLHttpRequests to MB-gateway from the customer's browser. The requests contain Modbus commands which are then passed on to air handling units control board and are executed.

The air handling unit web control interface is developed with angularJS and its back-end is developed with C programming languages.



Figure 2. Current air handling unit control user interface

In short, the current air handling unit control interface functionality will have to be implemented in the cloud environment and the collected data will have to be analyzed and reviewed.



### 3.1.3. Company mobile application analysis

The Company has a mobile application called “SALDA AIR” which enables the control of SALDA air handling units from a smartphone or tablet. All air handling units under control must be connected to a wireless local area network or local area network [2].

The application supports the same functions as an air handling unit web user interface [2]:

1. Easy adjustment of all AHU settings: system mode, supply air temperature, calendar events, humidity, CO2 level, etc.;
2. Monitoring of current air parameters: temperature, humidity level, etc.;
3. Display and Cancellation of Alarms and Warnings;
4. Service and Adjuster level settings: remote unit configuration and monitoring.

“SALDA AIR” mobile applications user interface is similar to an air handling unit web user interface (figure below).

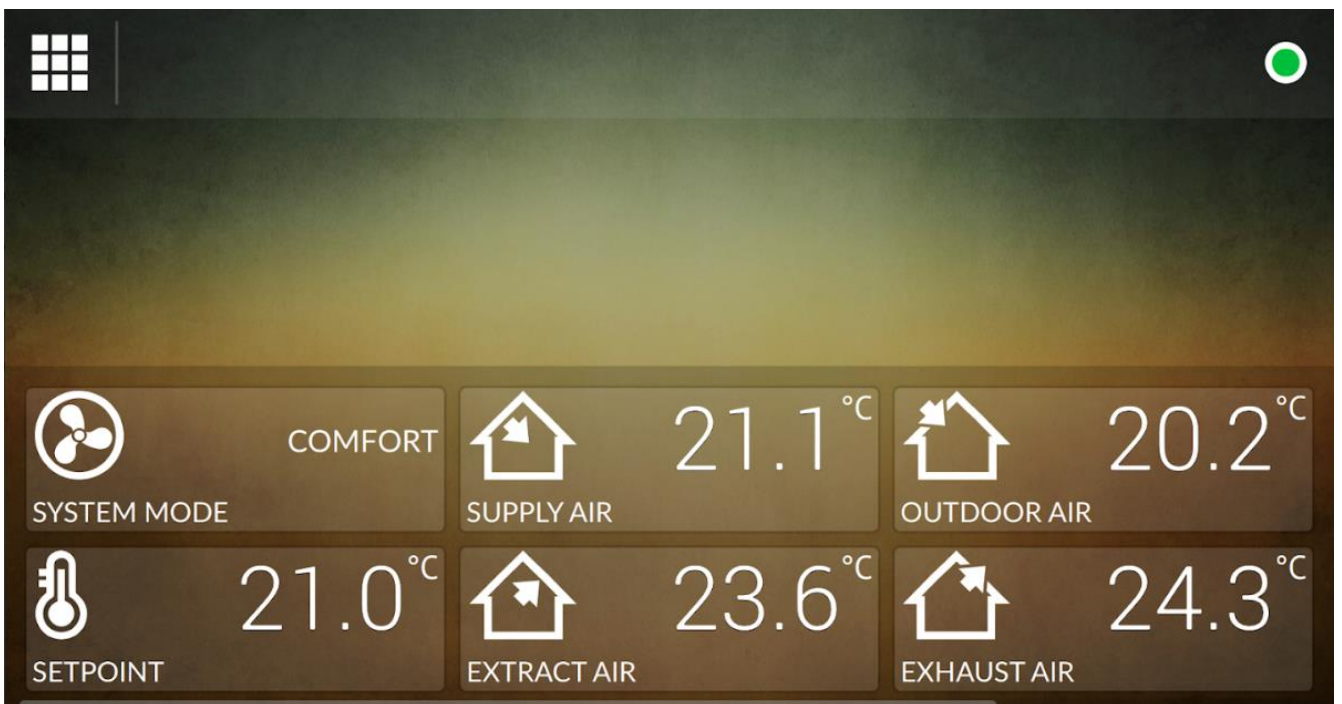


Figure 3. “SALDA AIR” mobile application user interface

To sum up, “SALDA AIR” mobile application has similar features and issues as the air handling unit control interface. One of the requirements for the developed system is for it to be able to have a service for communication with “SALDA AIR” mobile application.

### 3.1.4. MB-gateway analysis

MB-gateway uses Modbus protocol to communicate with the devices.

The information about MB-gateway was collected in an interview with the Company employees and MB-gateway specification sheet. MB-gateway has the following features [3]:

1. Supports TCP/IP Modbus connection;
2. 32 MHz processing unit;
3. 4 MB ram;
4. IPV4 protocol;
5. WEB server;
6. FTP server;
7. Building of Modbus commands using HTTP requests;
8. Authorized connection;
9. Automatic data communication between Modbus units.

MB-gateway limitations are as follows:

1. Low clock speed;
2. Low memory;
3. Only one processor core and thread;
4. Limited to only 768 bytes of data over http at a time.



*Figure 4. MB-gateway [3]*



Figure 5. Mb-gateway connections [3]

To sum up, MB-gateway supports IPV4 protocol which allows it to communicate with a server over a network. However, its hardware and amount of data it can send is limited.

### 3.1.5. Modbus protocol analysis

Commands to air handling units control board are sent using the Modbus protocol. Modbus protocol is a serial communication protocol developed by Modicon published by Modicon® in 1979 for use with its programmable logic controllers (PLCs). In simple terms, it is a method used for transmitting information over serial lines between electronic devices. The device requesting the information is called the Modbus Master and the devices supplying information are Modbus Slaves [4].

Modbus protocol supports different commands for different data addresses, each command is used to read different data and sends a different response [4]:

1. Read Coil Status (FC=01) – reads status of discrete coils;
2. Read Input Status (FC=02) – reads status of discrete inputs;
3. Read Holding Registers (FC=03) – requesting the content of analog output holding registers;
4. Read Input Registers (FC=04) – requesting the content of analog input register;
5. Force Single Coil (FC=05) – write content of discrete coil;
6. Preset Single Register (FC=06) – write content of analog output holding register;
7. Force Multiple Coils (FC=15) – write content of multiple discrete coil;
8. Preset Multiple Registers (FC=16) – write content of multiple holding registers.

In summary, Modbus protocol can be used to read and write data to air handling units. However, it cannot work over the Internet if devices are behind a router without port forwarding. This problem will have to be solved in order to be able to control air handling units remotely.

### 3.1.6. The project requirements

After analyzing the Company's technologies and problems, the project requirements were established. The project must have the following:

1. Cloud website in order to:
  - a. register air handling units (multiple devices per user);
  - b. control registered air handling units;
  - c. collect data of registered air handling units (customize which data is collected);
  - d. support multiple languages.
2. Support real time data updates when controlling air handling units (e.g. temperature changes in air handling unit should be updated real time);
3. Cloud API service which will collect and analyze air handling unit data and it must support the following features:
  - a. air handling unit data API for SALDA AIR web application and cloud web service;
  - b. command management system for air handling unit;
  - c. data parsing service received from air handling unit;
  - d. data analytics system to order and present data.

## 3.2. Analysis of the area of work

### 3.2.7. Internet of things analysis

The Internet of Things (IoT) is one of the most exciting technological developments in the world today and the global technical community is coalescing around the thought-leading content [5].

The Internet of Things, or IoT, which you probably have heard about with increasing frequency, is not a second Internet. Rather, it is a network of items — each embedded with sensors — which are connected to the Internet [6].

In 2012, about 3.7 million things were connected to the Internet via sensors, according to a report issued at the Trillion Sensors Summit, held last October at Stanford University and attended by representatives of more than 100 organizations from 14 countries. The goal of the meeting was to think up sensor-based applications likely to enter the market in the coming decade. The result was a startling prediction: The number of connected machines and devices will grow to 1 trillion by 2022 [7].

To sum up, the Internet of Things is a network of items connected to the Internet, which allows the owner of the items to control and manage their items connected to the Internet. The Internet of things also allows the device manufacturer to collect data from their devices, analyze it and provide better customer support and products in the future.

### 3.2.8. Google IoT system

Cloud IoT Core is a fully managed service that allows you to easily and securely connect, manage, and ingest data from millions of globally dispersed devices. Cloud IoT Core, in combination with other services on Google Cloud IoT platform, provides a complete solution for collecting, processing, analyzing, and visualizing IoT data in real time to support improved operational efficiency.

Google supports many device hardware interfaces such as [8]:

1. USB (Universal serial bus).
2. GPIO General - purpose input/output pins are connected directly to the processor. As their name implies, these pins are provided by the manufacturer to enable custom usage scenarios that the manufacturer didn't design for. GPIO pins can be designed to carry digital or analog signals, and digital pins have only two states: HIGH or LOW.
3. I2C Inter-Integrated Circuit serial bus uses a protocol that enables multiple modules to be assigned a discrete address on the bus.
4. SPI Serial Peripheral Interface Bus devices employ a master-slave architecture, with a single master and full-duplex communication. SPI specifies four logic signals.
5. UART Universal Asynchronous Receiver/Transmitter devices translate data between serial and parallel forms at the point where the data is acted on by the processor. UART is required when serial data must be laid out in memory in a parallel fashion.

Google IoT platform explanation:

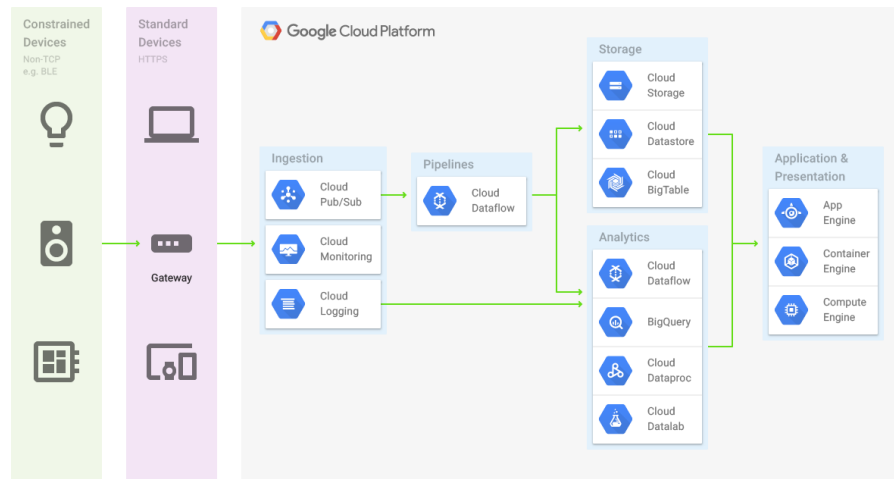


Figure 6. Google IoT system [8]

In summary, Google IoT supports data logging and analysis services, but it does not support Modbus protocol or remote device control services.

### 3.2.9. “IBM Watson” solution

IBM offers help for companies to build and deploy IoT end to end solutions.

Outcomes from IoT system [9]:

1. Increase visibility of the product, allows users to track the data and monitor products lifecycle;
2. Enhance scalability – allows the producers to update the software of the devices.
3. Enhanced user experience – users are a lot more satisfied with their products because of better customer service.

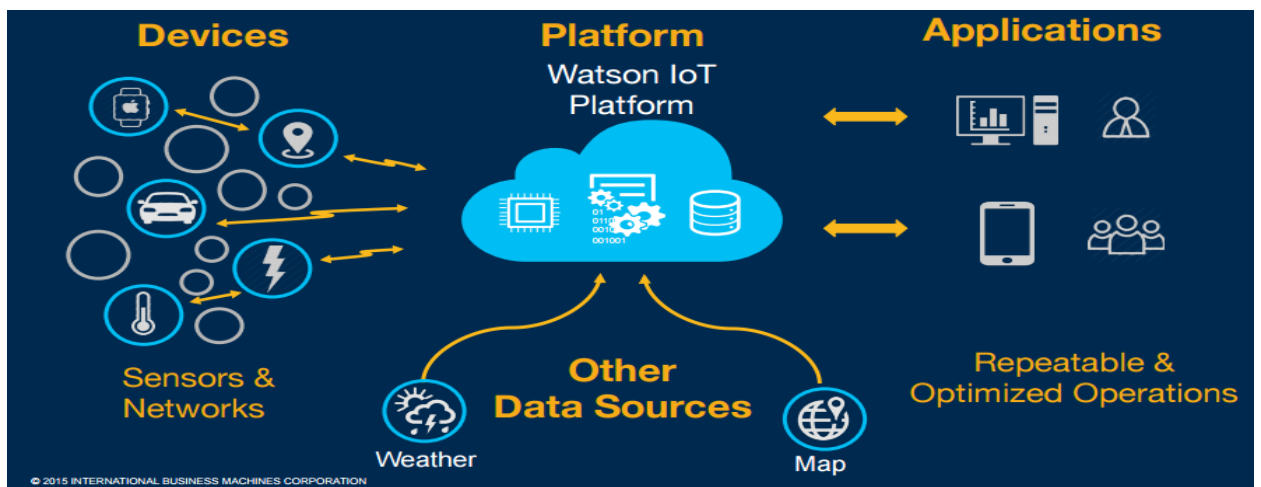


Figure 7. IOT platform example according to IBM [9]

“IBM Watson” system offers insights on the collected data from your device using artificial intelligence, by using [9]:

1. Natural language processing;
2. Machine learning;
3. Text analytics;
4. Video/Image/Audio analytics.

“IBM Watson” also has a risk management system which offers [9]:

1. End to end core security features;
2. System wide confidence via Security Analytics and Device Management;
3. Support for secure decentralized systems with Block chain.

In summary “IBM Watson” is a great system for cloud development. However, the system is not free and can be costly.

#### **3.2.10. Cognito Networks solution**

Cognito is an enterprise IoT solution for energy management and workflow automation, the framework supports multiple use cases and base applications. It can be deployed on a private or public cloud and it delivers multivendor IoT management, policy and closed-loop system as well as modeling capabilities to validate the solution before deployment [10].

Cognitos’ current status is that it is available for deployment to various customers, and is currently being used by:

1. NTT Innovation Institute. Several use cases in pipeline. Completed deployment in Mine Ventilation.
2. Assessment phase and power meter integration to deal from a leading Retail Supermarket. Additional use cases are Energy Management, Surveillance, POS integration, and Workflow Automation.

Cognito is 2016 TiE50 Winner, “Top 5 IoT startups to watch for in 2016” [10].

Cognito IoT solution is made of [10]:

1. Gateway;
  - a. Cognito gateway;
  - b. Custom gateway;
2. Cloud Service:
  - a. Eliminate IoT and sensor application silos;
  - b. Rapid IoT application development and customization;
  - c. Supports private and public cloud;
  - d. Multi-vendor interoperability and management;
  - e. Policy and framework for closed-loop systems.
3. Cognito API:
  - a. 3<sup>rd</sup> party gateway, cloud, and sensor integration;
  - b. Supports distributed architecture;
  - c. Ease of customization and extensibility.

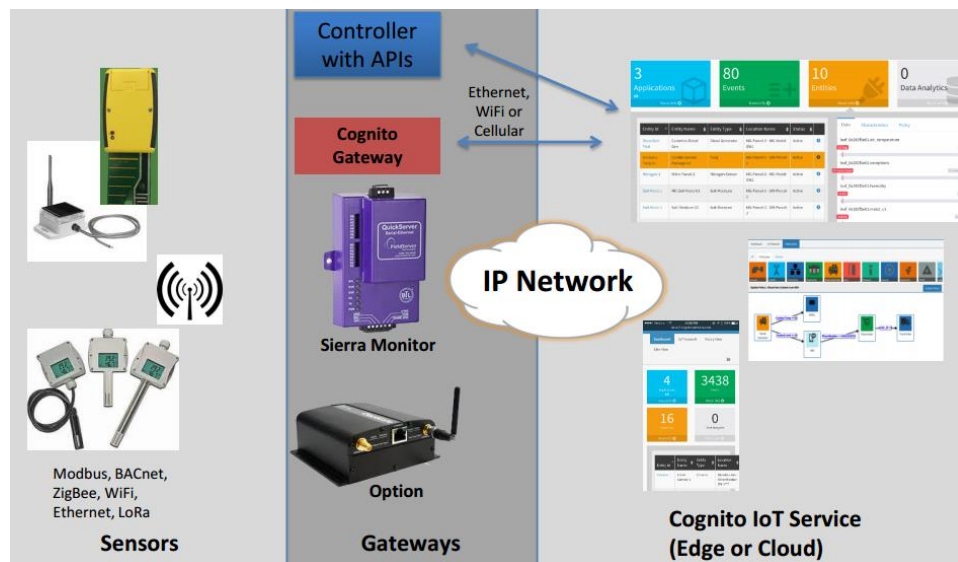


Figure 8. Cognito IoT solution [10]

In summary, Cognito IoT solution is a flexible choice for developing IoT projects and services. However, it has similar problems as “IBM Watson” Internet of things system.

### 3.2.11. “OJ AIR CLOUD” ahu cloud solution

OJ Air Cloud® is a SCADA system was developed specifically for OJ Electronic’s HVAC controllers. The users, who may include facility managers, service partners and AHU manufacturers, can all invite each other to be part of OJ Air Cloud®, and as you would expect, the cloud-based SCADA system



provides a total overview of all the systems connected to the cloud and their current status. This enables rapid identification and diagnostics of any problems – allowing you and your partners to arrive at the right solution quickly [11].

In short, there are other air handling unit cloud control systems, but they only work for OJ Electronic's HVAC air handling units, so their system does not fit our project.

#### 3.2.12. "SystemAir" company analysis

SystemAir is a Lithuanian company that provides various ventilation solutions to its clients. It sells different products, such as:

1. Fans;
2. Air ventilation units;
3. Air handling units;
4. Electric heaters;
5. Water heaters;
6. Water coolers;
7. Chillers.

The company sells air handling units that can be controlled locally using LED display; however, it does not offer clients remote cloud air handling unit control or data collection services, so currently there is no Lithuanian company that provides a remote air handling unit controlling solution [12].

### 3.3. Analysis of tools for cloud development

#### 3.3.13. Programming language analysis

In this section a cloud development programming language will be selected. Programming languages are measured by the following conditions [13]:

1. Google search – establishes how popular the programming language is and the number of online information online about the programming language;
2. Google trends – this indicates the demands for information about a particular language, this is a valid point for upcoming popular languages;
3. Twitter – measures the amount of chatter on social media about the programming language;
4. GitHub – measures fresh activity around the specific language, looking at active repositories;

5. Stack Overflow – measures the demand of information about the specific programming language;
6. Reddit – measures the demand of information about the specific programming language;
7. Hacker news – measures the demand of information about the specific programming language;
8. Dice – measure the number of fresh job postings that involve the specific programming language;
9. IEEE Xplore Digital Library – this metric captures the prevalence of difference programming languages as used and references in scholarship.

The languages are also split into four types [13]:

1. Web – used for developing websites and applications;
2. Mobile – used for developing applications for mobile devices;
3. Enterprise – used for enterprise, desktop and scientific applications;
4. Embedded – used to program device controllers.

The table below is based on the above measurements rules of programming languages [13]:

*Table 2. Programming language scores and types [13]*

<b>Programming Language</b>	<b>Score(min – 0; Max - 100)</b>	<b>Language Types</b>
<b>C</b>	100.0	Mobile, Enterprise, Embedded
<b>Java</b>	98.1	Web, Mobile, Enterprise
<b>Python</b>	98.0	Web, Enterprise
<b>C++</b>	95.9	Mobile, Enterprise, Embedded
<b>C#</b>	86.7	Web, Mobile, Enterprise
<b>PHP</b>	82.8	Web
<b>JavaScript</b>	82.2	Web, Mobile
<b>Ruby</b>	74.5	Web, Enterprise
<b>Assembly</b>	68.6	Embedded
<b>Html</b>	65.6	Web
<b>Objective C</b>	53.4	Mobile, Enterprise

In summary, there are a lot of programming languages for different applications, e.g. Java, Php, JavaScript are good for web applications, and languages C, C++, C are good for enterprise applications. For this project we are going to use C programming language to work with the hardware and WEB language such as PHP/HTML/CSS/JS for working with the user interface and server side code.

### 3.3.14. Programming framework/library analysis (UI development)

The developed system user interface will have to be able to update in real time with constantly flowing data changes. The following three programming languages were compared for cloud user interface development [14]:

1. AngularJS;
2. Angular 5;
3. ReactJS;
4. Ember.js;

The language comparison results are to be found in the table below.

*Table 3. JavaScript framework analysis [15, 16]*

<b>Framework</b>	<b>Cons</b>	<b>Pros</b>
<b>Angular 5</b>	<ul style="list-style-type: none"> <li>Directives as the Way to Add Functionality.</li> <li>Flexible when using Filters.</li> <li>In-place DOM manipulations.</li> <li>Relevant Service Providers.</li> <li>Prepared Unit Testing. Angular 5 possesses easier APIs, lazy loading, simpler/faster debugging.</li> <li>Deployment: you can build you app and deploy on a linux hosting for example, to have NodeJS on your server can be great but it is not necessary to release your final app.</li> <li>Angular 5 provides nested components.</li> <li>Angular 5 makes it possible to execute more than 5 systems together.</li> </ul>	<ul style="list-style-type: none"> <li>Difficulty in learning. If you have not used typescript before, you will need to learn it and spend time doing it.</li> <li>Regular DOM. Angular manipulates actual DOM directly, which makes it much slower and inefficient in comparison with React.</li> <li>Lagging UI. If there are more than 2000 watchers, it can get the UI to severely lag. This means that the possible complexity of Angular Forms is limited. This includes big data grids and lists.</li> </ul>
<b>ReactJS</b>	<ul style="list-style-type: none"> <li>React.js works great for teams, strongly enforcing UI and workflow patterns.</li> <li>Allows the reuse of code components.</li> <li>Data flows in one direction.</li> </ul>	<ul style="list-style-type: none"> <li>Poor documentation of React</li> <li>React.js is only a view layer.</li> <li>There is a learning curve for beginners who are new to web development.</li> </ul>
<b>Ember.js</b>	<ul style="list-style-type: none"> <li>Excellent routing.</li> <li>Increased performance,because similar tasks are processed in one go.</li> <li>Easy to reuse components.</li> <li>Server side rendering.</li> <li>One-way data flow.</li> <li>Helps with writing simple and modular code by using Promises.</li> <li>Uses a relatively clean and easy to understand templating engine.</li> <li>Useful bindings.</li> <li>Auto-updating templates.</li> </ul>	<ul style="list-style-type: none"> <li>Too large for small projects.</li> <li>Steep learning curve.</li> <li>More Ruby than JavaScript.</li> <li>Heavy on memory.</li> <li>Not a complete solution.</li> <li>Large file size.</li> <li>A new syntax has to be learnt.</li> </ul>

In short, angular 5 typescript framework (or just Angular) was chosen for cloud user interface development. It is advantageous because it supports DOM manipulation by data objects and allows for near instant user interface updates.

### 3.3.15. Client-Server communication protocol analysis

Cloud user interface will be updated on data retrieval from the server, which can be done with protocols such as XHR polling, SSE (server side events) or WebSockets.

XHR polling – is built on HTTP and uses a request-response design. They can send a request to the server and the server can respond to this request. This means that the client will have to keep sending requests to the server to get updates without reloading the page [17].

Server side events (SSE) – the communication between the client and the server is initiated by the client setting up a TCP connection and sending a HTTP request to the server. After parsing the request, the server sends a HTTP response to the client. The response is not considered complete after the first payload is received. After receiving the headers, the client then expects a chunked response, where every chunk is generated by the server and not sent at once [17].

WebSockets – The client sends a HTTP request over TCP to try to establish WebSocket connection. If the server supports WebSocket connection, it will inform the client about the established connection via HTTP request. Sending data via WebSockets is duplex which means that neither the server, nor the client has to wait for a response, both of them can send data to each other simultaneously as long as a connection is established [17].

In summary, the client user interface will use XHR polling protocol when polling device data, because it is a simple and easy way to implement the protocol for communication with the server. If the HTTP overhead is too large, the user interface and cloud data API can then be expanded to support WebSockets or SSE.

### 3.3.16. MySQL and NoSQL database comparison for cloud usage

MySQL (Structured Query Language) – databases have been a primary data storage mechanism for more than four decades. Usage exploded in the late 1990s with the rise of web applications and open-source options such as MySQL, PostgreSQL and SQLite. NoSQL databases which have existed since the 1960s, but have been recently gaining traction with popular options such as MongoDB, CouchDB, Redis and Apache Cassandra [18].

MySQL – mongoDB side by side comparison is given in figure below [18].

Date	MySQL	MongoDB
<b>Key features</b>	<ul style="list-style-type: none"> <li>- Full-text searching and indexing</li> <li>- Integrated replication support</li> <li>- Triggers</li> <li>- SubSELECTs</li> <li>- Query caching</li> <li>- SSL support</li> <li>- Unicode support</li> <li>- Different storage engines with various performance characteristics</li> </ul>	<ul style="list-style-type: none"> <li>- Auto-sharding</li> <li>- Native replication</li> <li>- In-memory speed</li> <li>- Embedded data models support</li> <li>- Comprehensive secondary indexes</li> <li>- Rich query language support</li> <li>- Various storage engines support</li> </ul>
<b>Best used for</b>	<ul style="list-style-type: none"> <li>- Data structure fits for tables and rows</li> <li>- Strong dependence on multi-row transactions</li> <li>- Frequent updates and modifications of large volume of records</li> <li>- Relatively small datasets</li> </ul>	<ul style="list-style-type: none"> <li>- High write loads</li> <li>- Unstable schema</li> <li>- Your DB is set to grow big</li> <li>- Data is location based</li> <li>- HA (high availability) in unstable environment is required</li> <li>- No database administrators (DBAs)</li> </ul>
<b>Examples</b>	NASA, US Navy, Bank of Finland, UCR, Walmart, Sony, S2 Security Corporation, Telenor, Italtel, iStock, Uber, Zappos, Booking.com, Twitter, Facebook, others.	Expedia, Bosch, Otto, eBay, Gap, Forbes, Foursquare, Adobe, Intuit, Metlife, BuzzFeed, Crittercism, CitiGroup, the City of Chicago, others.

Figure 9. MySQL mongoDB side by side comparison [18].

In summary, mongoDB is more efficient for managing the Internet of things data, because it supports high write loads. MySQL is more efficient when it has specific data structuring and not many inserts are made.

## **4. SOFTWARE DESIGN AND REQUIREMENTS**

### **4.1. Plan for the implementation of the project**

There are various tools for software design and planning, for example:

1. UML – unified modeling language, which allows to:
  - a. Create data models;
  - b. Create class diagrams;
  - c. Create data movement diagrams;
2. BPMN – business project model notation, which allows to:
  - a. Draw realistic data flows;
  - b. Allows to simulate activities and processes;
  - c. Allows human resource management;

UML was chosen for software planning and modeling, because it suits our project better in making it possible to describe software development process. BPMN was not chosen because it is preferable for company resource management and resource simulations.

#### **4.1.1. Project hardware requirements**

The literature review indicates that the project will need a MB-gateway which will be used to communicate with air handling units. MB-gateway connects to the air handling units control board in order to control the air handling unit and the router for access over the internet (Figure below).

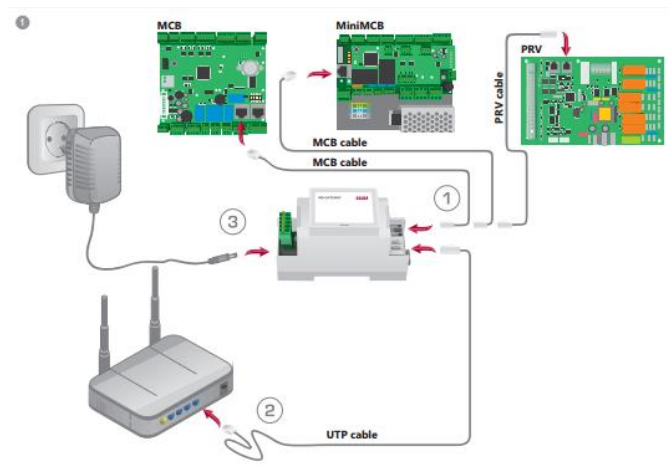


Figure 10. MB-gateway connection manual [19]

In summary, the established hardware requirements are (the hardware is provided by the company that the system is being built for):

1. MB-gateway;
2. Router;
3. Control board of the air handling unit (PRV or MCB look at Figure 7.);
4. Cloud server to handle the data.

## 4.2. Initial description of the project

### 4.2.2. MB-gateway data transfer protocol description

Data transfer protocol was analyzed during an interview with the Company IT Department developers;

MB-gateway will have to send the data to the server safely. The transferred data will be encrypted using AES-128 encryption algorithm. As MB-gateway already supports such encryption, no extra development needs to be done.

The data will be sent in time intervals with HTTP protocol using POST requests. Each time a request is sent from MB-gateway to cloud it can have the following data:

1. ID1&MACXXX – here id is unique request identifier and XXX is gateway mac address;
2. ID1&MACXXXDATAYYY – here id is unique request identifier and XXX is gateway mac address and YYY is the result air handling unit data;

The cloud service that handles the data will have to return responses to MB-gateway with specifically structured data, so the MB-gateway knows how to handle it.

MB-gateway will support these commands (written in decimal):

1. 0 – execute Modbus command (executes MB-gateway Modbus protocol command);
2. 1 – get slave list (returns all MB-gateways controller air handling unit list);
3. 2 – get params per & (returns requested parameter list);
4. 3-100 – these values are reserved for future protocol improvements;
5. 101 – writes slaves (allows to change air handling unit names and their addresses);
6. 102 – write RT (read time) writes new delay settings value for how often commands are requested from cloud in milliseconds;
7. 103-254 – reserved for future protocol improvements.

In summary, data reading and writing commands will be modified Modbus protocol commands to support MAC and command identification. Other commands will be used to manage gateway settings.

#### 4.2.3. Execute Modbus command description

This command is used to manage Modbus commands of the device. If the service sends a command with id of 0, it means that the following data will contain read or write Modbus commands.

Read single or multiple registers extended Modbus protocol command example:

1. Red – unique command identifier;
2. Blue – gateway command;
3. Green – slave id;
4. Orange – Modbus function code;
5. Brown – start register address;
6. Purple – how many registers to read.

*Table 4. Read command example*



Write single register extended Modbus protocol command example:

1. Red – unique command identifier;
2. Blue – gateway command;
3. Green – slave id;
4. Orange – Modbus function code;
5. Brown – address of the register to write;



6. Purple – value to write to the register.

*Table 5. Write single register example*



Write multiple coil registers extended Modbus protocol command example:

1. Red – unique command identifier;
2. Blue – gateway command;
3. Green – slave id;
4. Orange – Modbus function code;
5. Brown – address of the first coil;
6. Purple – number of coils written (10 coils / 8 coils per byte = 2 bytes next, in example we show 5 coils);
7. Gray – Data to write (for example if writing 5 coils, the first 5 bytes will have meaning and other 3 will be auto filled, e.g. 00011111, the three 0's are fillers to fill the full byte);

*Table 6. Write multiple coils command example*



Write multiple holding registers extended Modbus protocol command example:

1. Red – unique command identifier;
2. Blue – gateway command;
3. Green – slave id;
4. Orange – Modbus function code;
5. Brown – address of the first register to write;
6. Purple – number of registers to write;
7. Gray – number of data bytes to follow;
8. White – values of registers to write (each value is 2 bytes).

Multiple commands can be stacked up, e.g. the cloud service can send 2 read commands and 2 write commands to gateway, which then will execute all 4 commands and send the proper response for the commands.

Gateway has to send this type of responses:

1. ID – unique request ID (helps us know which data gateway is returning);
2. MAC address – unique address (helps us know which gateway is sending data);
3. Data – Modbus data read (data received from gateway depending on command).

Read input or holding register commands response looks like this – ID1000&MACXXXXXXXX&DATA110306AE4156524340, it is clear that gateway always sends commands identifier ID1000, its own MAC address MACXXXXXXXX and data read DATA110306AE4156524340.

Data is in hex and contains the following data:

1. Green – slave id;
2. Orange – Modbus function code;
3. Brown – how many bytes to follow;
4. Yellow – values of registers (each value is 2 bytes).

*Table 7. Modbus read input and holding registers response*

11	03	06	AE	41	56	52	43	40
----	----	----	----	----	----	----	----	----

#### 4.2.4. Get slave list commands description

This command will be used to read MB-gateway parameters data, such as:

1. Slave data;
2. Request time;
3. Active theme;
4. Active language.

This command will be mainly used when configuring cloud website and registering a gateway. It will allow the cloud system to import the gateway data.

#### 4.2.5. Get params per & command description

This command is used to read MB-gateways specific parameters by a given tag. Requested tags are sent to MB-gateway, e.g. “RT&Language“, MB-gateway then parses the commands and returns the given tags values, e.g. a response for this command would look like: “RT=2000&Language=EN“.

#### 4.2.6. Write slaves command description

This command writes new slaves (air handling unit) addresses and names to a given MB-gateway which is made of:

1. ID – position in file (1 byte);
2. Address – slave address (2 bytes);
3. Name – ASCII name up to 12 characters (12 bytes).

It is possible to send multiple write slaves commands; the explanation follows below (each block is 1 byte):

1. Red – unique command identifier;
2. Blue – gateway command ;
3. Yellow – file position id;
4. Green – slave address;
5. Orange – ASCII values for name of given slave;
6. When writing more than one slave, the writing continues by writing id, slave address and name of a given slave. If a slave name is shorter than 12 characters, extra empty characters (0x00) should be appended to finish up 12 bytes (tables below);

*Table 8. Write slaves explanation part 1.*



*Table 9. Write slaves explanation part 2.*



#### 4.2.7. User interface use case diagram

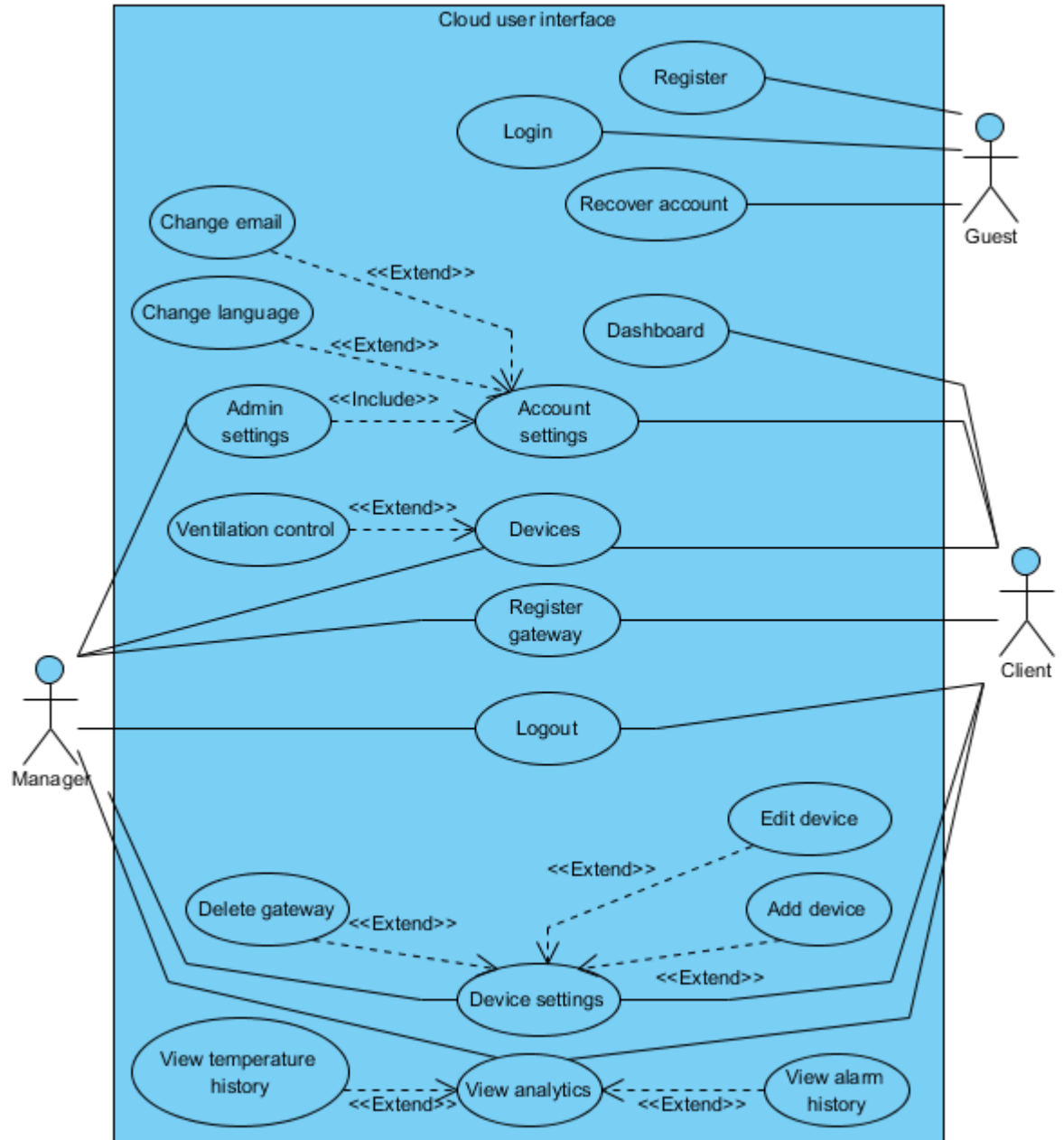


Figure 11. Cloud control use case diagram.

#### 4.2.8. Registration activity diagram

To register an account user has to enter these credentials (\* means required):

1. Username\*;
2. Password\*;
3. Re-enter password\*;

#### 4. Email\*.

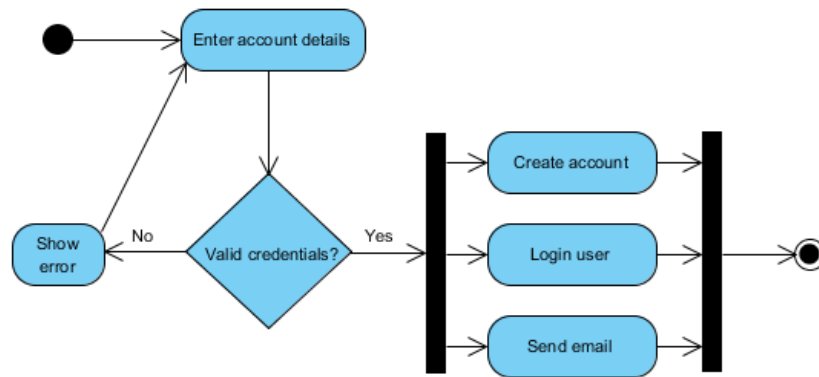


Figure 12. Account registration activity diagram

#### 4.2.9. Login activity diagram

Users who have registered on the system can login by following these steps:

1. User enters login credentials;
2. If login credentials are invalid, check login attempts;
3. If less than 3 failed login attempts in the last 15 minutes, increment login attempts by 1;
4. Go to step one;
5. If login credentials are valid retrieve user information and then show control panel;
6. If 3 or more failed login attempts have happened, send warning to client and lock the account for 15 minutes.

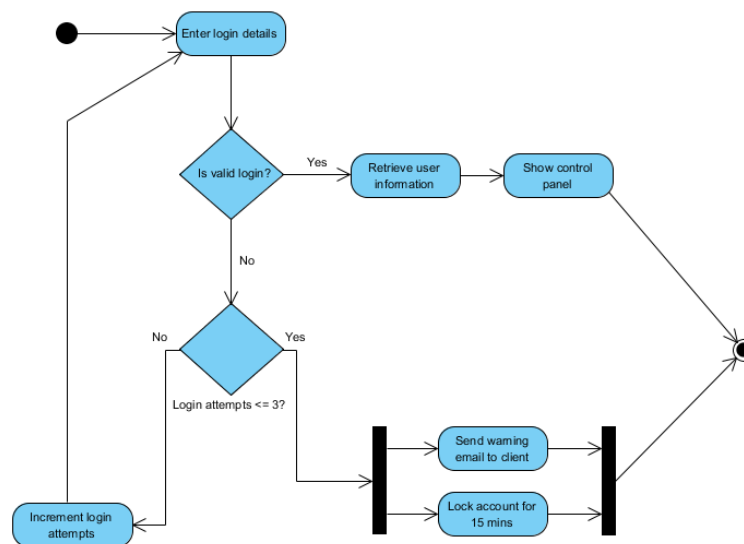


Figure 13. Login activity diagram

#### 4.2.10. MB-gateway – server communication sequence diagram

MB-gateway will communicate with the server by sending requests and receiving commands in the following sequence (figures below):

1. MB-Gateways sends a request to the server, with required data (MAC and request ID);
2. Server collects extended Modbus protocol commands from the database by MB-gateway credentials and returns them in the response body;
3. MB-gateway then executes the received commands;
4. MB-gateway then sends a new POST request to the server with its MAC address, unique request identifier and command result data;
5. Server then processes the received data from MB-gateway and saves it in the database;
6. End of cycle, back to step 1.

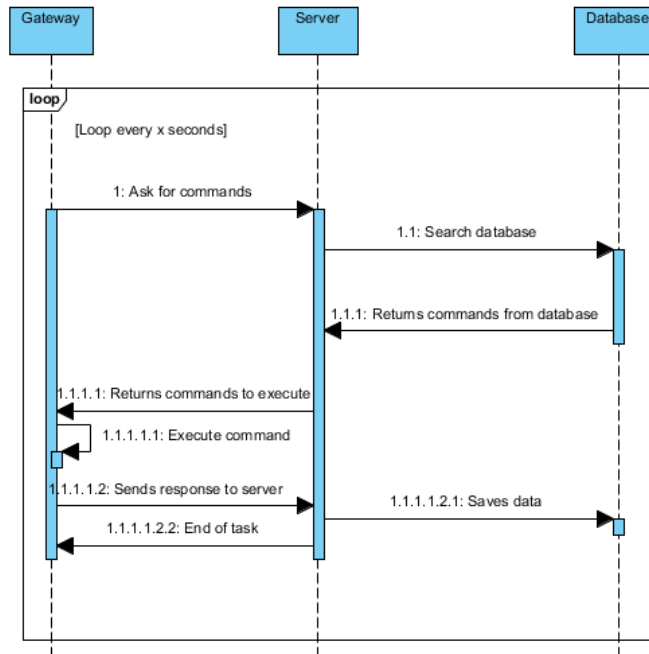


Figure 14. Gateway - Server communication sequence diagram

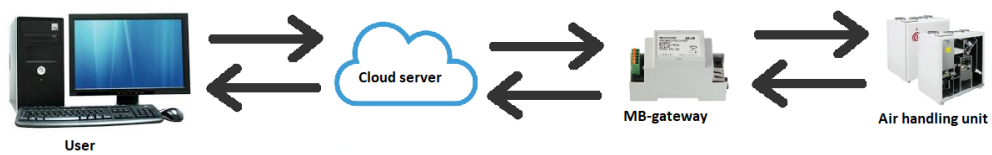


Figure 15. Full cloud devices communication diagram

#### 4.2.11. Cloud control database schema

Cloud control database schema is used to control users, commands, registers and gateway data.

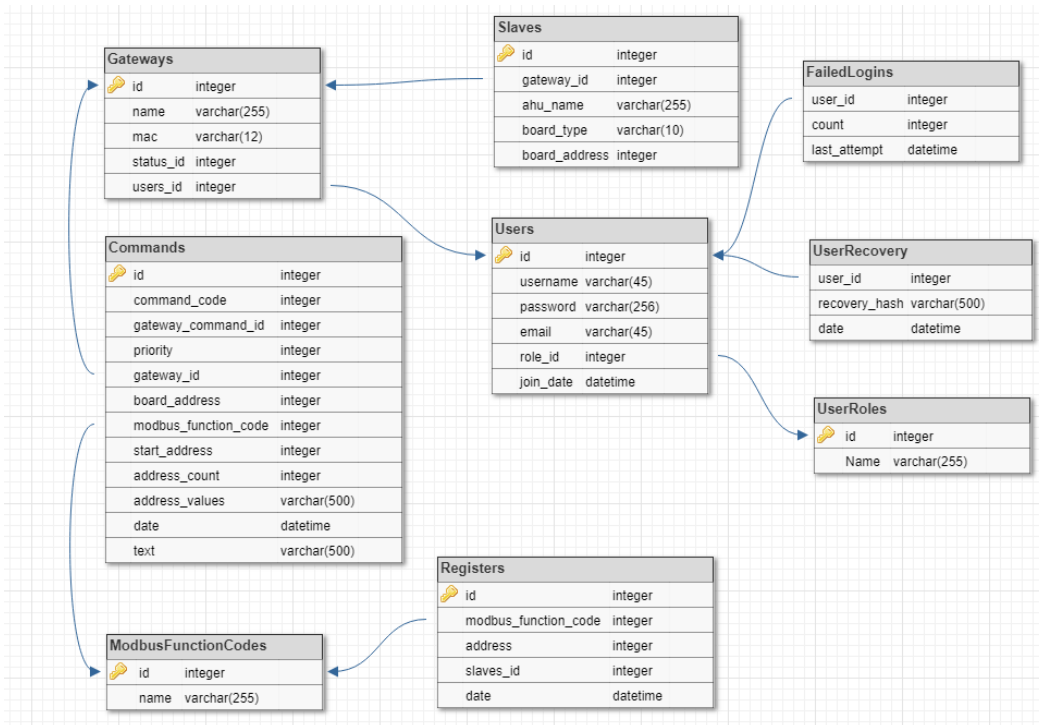


Figure 16. Cloud control database schema

Explanation of cloud control database schema is in Annex 1.

#### 4.2.12. Cloud data collection and analytics database schema

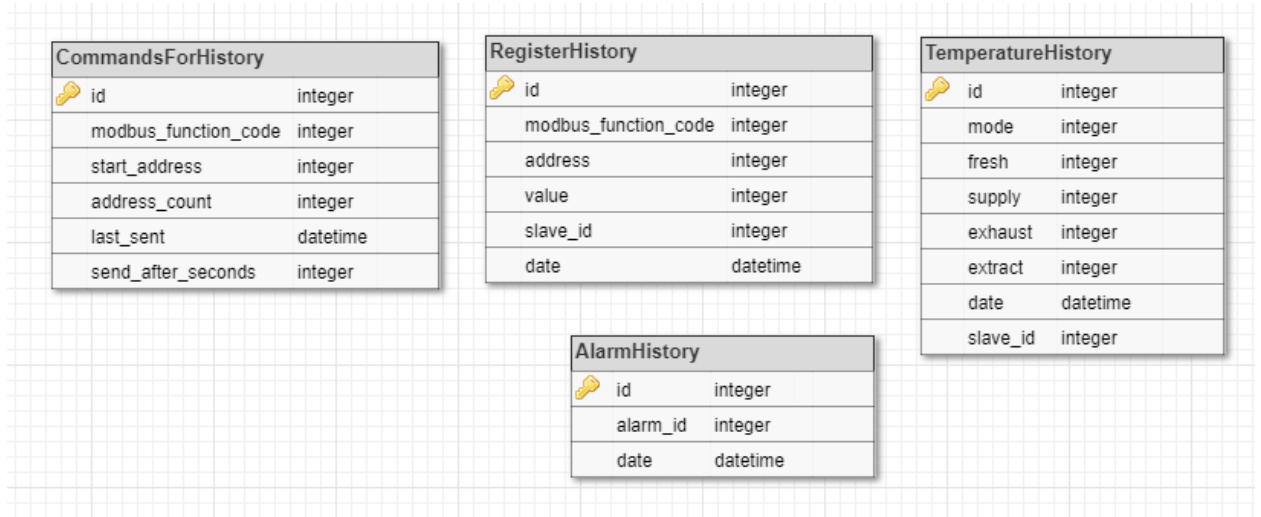


Figure 17. Cloud data collection database schema

Explanation of cloud data collection database schema is in Annex 2.

#### 4.2.13. Cloud security requirements description

Cloud web user interface must have these security features:

1. SQL injection protection – solved by escaping user inputs;
2. XSS (cross site scripting) protection – solved by cleaning all javascript/html tags from user inputs;
3. CXRF (cross site request forgery) protection – solved by implementing tokens into user inputs and only accepting inputs that have valid tokens;
4. Brute force account security – accounts should only be allowed 3 invalid login attempts every 15 minutes to prevent brute force attacks.

Device security features:

1. Communication with cloud can only be activated using the LAN (local area network) device control system;
2. Encryption key of device communication protocol should be editable in case of server leaking encryption keys.



#### 4.2.14. Analytics tools

Cloud analytics tools will allow clients to preview the collected air handling unit data. To allow for custom preview settings, analytics tools will have to support the following functions:

1. Select air handling unit;
2. Show temperature changes of:
  - a. Last 24 hours;
  - b. Last week;
  - c. Last month;
  - d. Last quarter;
  - e. Last year;
3. Support picking of start-end dates.

All data shown in analytics tools will have to be visualized in charts.

#### 4.3.Choice of tools and techniques

According to the project functional requirements, tools can be selected for project development. The following tools were selected:

1. NetBeans development IDE – tool for developing the user interface and cloud services;
2. Chart.js – a library for developing analytics tools;
3. Typescript – programming language to develop user interface;
4. Angular 5 – a typescript framework to make application development easier and code more reusable, because the project will be developed later on;
5. Bootstrap – a css library for developing the cloud web user interface.

The system will need two databases, one - to manage commands, users and other rarely changing data and the other - to manage the device air handling unit data. Both databases will be MySQL databases. If the device data collection database will be too slow, it can be migrated to mongoDB, which is suitable for such a task.

## **5. PROJECT DEVELOPMENT, VERIFICATION AND VALIDATION**

### **5.1. Project workflow and schedule**

After the analysis of the topic and the required system design the project workflow is clear. The project workflow consists of the following steps:

1. Implement the database;
2. Implement extended Modbus protocol in MB-gateway (was done by the Company IT department);
3. Implement extended Modbus protocol in server side;
4. Develop user interface of the cloud;
5. Develop data management algorithm, to handle data received from the devices;
6. Test the protocol and user interface;
7. Improve the user interface;
8. Improve data management of devices.

#### **5.1.1. Database implementation**

Database implementation was done on the Company's server. Two databases were implemented according to the project design; one database for managing cloud user interface and the other one for managing collected air handling unit data.

#### **5.1.2. Extended Modbus protocol implementation**

Extended Modbus protocol implementation was done using PHP and C programming languages. The protocol had to be implemented on the server and on the device.

The protocol was not implemented fully due to the fact that the Company employees failed to successfully ensure the device side implementation.

#### **5.1.3. Cloud interface development**

Cloud user interface was developed by using angular, chart.js, i18n.js, bootstrap libraries and frameworks. The user interface is less than complete and still needs a lot of work on dashboard, command management and custom settings.

## 5.2. Description of problems and their solutions

### 5.2.4. Protocol improvements

Initially, extended Modbus protocol had the following structure:

1. 0x00 – read/write Modbus command;
2. 0x01 – read parameters data;
3. 0x02 – write parameters data.

The problem with this approach is that some commands do too many things and sometimes are too long, e.g., read parameters data, reading 30 rows (one row taking 255 bytes) would take a long time with the limit of 768 bytes of data per single HTTP request.

This particular problem was solved by improving on protocols command structure by adding new commands. The fixed protocol is described in MB-gateway protocol description section; here is a short description of possible commands (written in decimal):

1. 0 – read or write Modbus command as in initial description;
2. 1 – get slave list (returns all MB-gateways controller air handling unit list);
3. 2 – get params per & (returns requested parameter list);
4. 3-100 – these values are reserved for future protocol improvements;
5. 101 – writes slaves (allows to change air handling unit names, their addresses);
6. 102 – write RT (read time) writes new delay value for how often commands are requested from cloud in milliseconds;
7. 103-254 – reserved for future protocol improvements.

### 5.2.5. Data management improvements

Initially it was decided to collect all register data in a single register table which would contain all data required for analysis. The table contains the following columns:

1. ID – unique id;
2. Modbus\_function\_code – which Modbus type it is;
3. Address – address of given register;
4. Slave\_id – air handling unit which data was collected from;
5. Date – time when data was collected;

The problem with this approach is that the queries to select data for charts would have to be complicated and would take too much time, knowing there are about 1000 registers currently in use.

The solution to this problem is to create multiple tables that would collect similar data depending on the table, e.g. temperature table would collect temperature data, alarm table would collect alarm data. This improvement allows for a lot simpler queries and data management.

### 5.3. Software testing

#### 5.3.6. Software testing rules

Cloud air handling unit management system data and functions have to be tested. Testing involves the following steps:

1. Feature to test – what the test is checking;
2. Expected result – expected result on successful test;
3. Result – result of the test.

Testing was performed to check the following features:

1. Data integrity of air handling units in cloud web page;
2. Whether changes in cloud user interface also change MB-gateway web user interface data (e.g. changing mode has to change registers in the device which would then be represented in MB-gateway web user interface);

#### 5.3.7. Cloud data integrity and command testing

Cloud data integrity testing:

*Table 10. Cloud data integrity testing*

<b>Testing feature</b>	Check data integrity in cloud of air handling unit.
<b>Expected result</b>	Cloud user interface should represent the same data as the data in air handling units registers.
<b>Result</b>	There was a minor data difference between the air handling unit and the data in registers. For example, the temperature data had a difference of about 0.2 C due to the fact that the cloud server did not yet receive updated air handling unit data. It takes 2-30 seconds for the data to be updated depending on the device settings and location.

Cloud commands testing:

Table 11. Air handling units mode control in cloud environment testing

<b>Testing feature</b>	Air handling unit mode and desired temperature set point selection.
<b>Expected result</b>	Cloud user interface should be able to change air handling units mode and desired output air temperature.
<b>Result</b>	Changing the mode and the air handling unit output temperature worked properly, although there was a minor delay between the pressing of the button and the air handling unit actually working under a different mode. The difference occurred because it took some time for the command to be sent to the device over the Internet.

To sum up, reading and sending data to the air handling unit works properly. This is done by using extended Modbus protocol. There is a short delay of the cloud air handling unit control system compared to LAN because the commands have to travel over HTTP. The length of the delay depends on the Internet connection and device settings.

### 5.3.8. Temperature history testing

Temperature history analytics tools allow clients to view air temperature changes over time. In the given example (picture below) there are about 25 days of collected data for a single air handling unit, which is about 100000 rows of temperature data points. This data enables customers to determine the air handling unit component failures, e.g. a sudden unexplained temperature change could show a rotor or coil error.

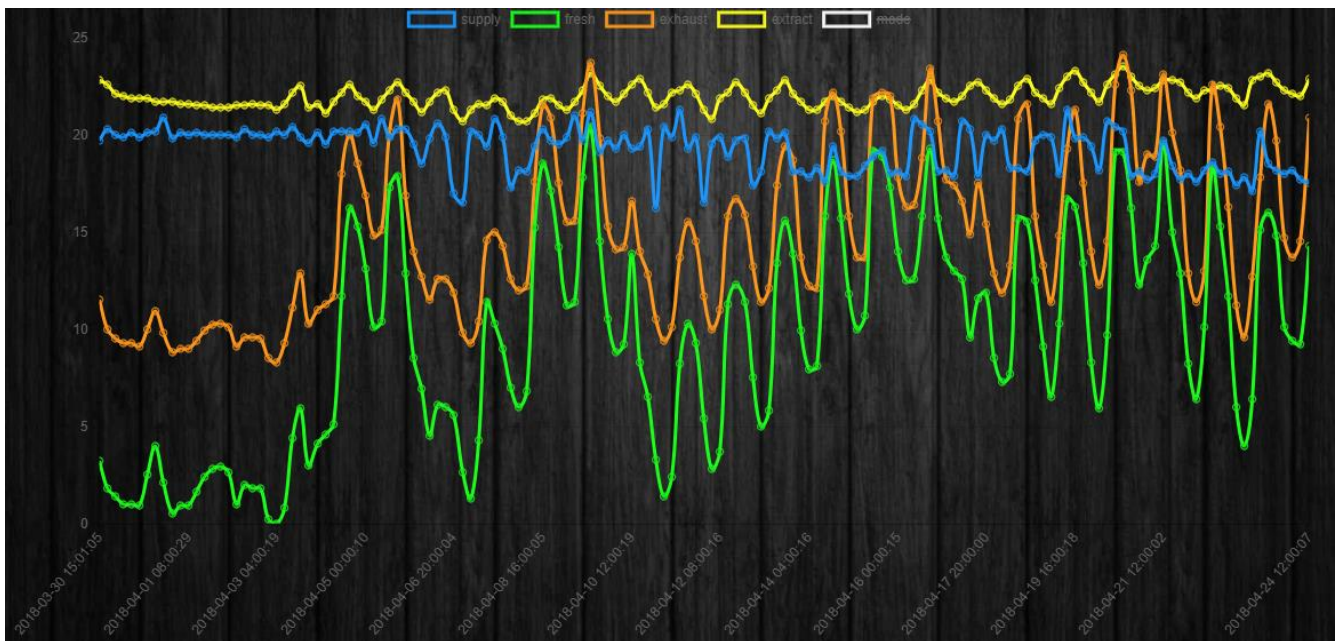


Figure 18. Collected temperature data example (data from 25 days)

In brief, temperature history analytics tool works properly. It collects all air handling units temperature data, which allows clients and company employees to monitor air handling units and detect errors.

#### **5.4. Description of the final stage of the project**

This is an ongoing project and its development will continue after the thesis completion. It will be used by the Company for remote air handling unit control and data collection. Subsequently, the system will be offered to clients as an alternative to air handling unit web interface.

Alpha version of the software has been implemented on the Company server. A single live air handling unit was connected to test Modbus protocol and user interface as well as collect temperature data and active air handling unit mode.

The cloud user interface main features that were implemented are as follows:

1. Account creation and login;
2. Device registration and management;
3. Device control pages for MCB control board;
4. Analytics tools for temperature history monitoring;
5. Alarm history monitoring.

The following cloud features that still need to be implemented:

1. Dashboard – allowing user to configure what he can see on home page;
2. Custom data logging fields – allows user to configure specifically which data he wants to log;
3. Custom data logging charts – allows user to view custom selected data;
4. Device control page for PRGC board;
5. “SALDA AIR” application service.

To conclude, the project has been a success and air handling unit data can now be collected and previewed while air handling units can be controlled over the Internet. However, it is not quite complete and needs further development.

## **6. ANALYSIS OF THE RESULTS**

### **6.1. Extended Modbus protocol validation and verification**

The problems of collecting air handling unit data and sending air handling unit commands over the Internet were resolved by extending the Modbus protocol. The extended Modbus protocol has been designed and tested to successfully support the following commands:

1. Reading single register;
2. Reading multiple register;
3. Reading single registers from multiple commands;
4. Reading multiple registers from multiple command;
5. Writing single register;
6. Writing single registers from multiple commands.

To sum it up, extended Modbus protocol supports basic Modbus protocol commands and provides extra functionality to allow remote control and data logging of devices which are behind a router, which was impossible to do with basic Modbus protocol.

### **6.2. Collected data validation and verification**

The cloud implementation was added to a live air handling unit on 30-03-2018, so about a month worth of data points were collected, which amounts to about 100000 points of temperature data. During the test period no air handling unit errors were detected, consequently, it is difficult to determine to what extent the collected data facilitates the detection of errors. However, the collected data proves definitely applicable to the following:

1. Air handling unit air filter replacement detection;
2. Warning clients about on-going air handling unit failures;
3. Monitoring air handling unit usage over time, allowing to determine what air handling units the client needs and how powerful they have to be;
4. Monitoring weather changes over time, over different locations.

## 7. CONCLUSIONS AND FUTURE WORK

Analysis of the Company completed, a problem regarding the inability of air handling units to be controlled or monitored over the Internet was identified.

1. Analysis of scientific and practical work was performed and it was decided that the most efficient way to implement the project is to develop a small internet of things platform, which would allow for communication over the Internet.
2. Analysis of the tools, techniques and programming languages for cloud development was carried out and it was established that the following technologies will be used for the project development: angular framework for user interface development and PHP for server side development.
3. The project was designed and the extended Modbus protocol was prototyped allowing air handling unit control over the Internet.
4. Development of the system was completed, the extended Modbus protocol and the user interface were implemented and improved, but with some shortcomings: some extended Modbus protocol commands and user interface functions are missing and need further development.
5. Testing and validation were performed to check data integrity and command execution correctness. The tests were successful, with a minor setback of commands taking longer to execute over the Internet, which is to be expected comparing it to commands sent on the local area network.

Basic user interface manual was written and is available in Annex 3.

In conclusion, the collected data is useful for air handling unit maintenance, customer support and monitoring weather patterns over time.

The developed software will make it possible to adjust the accuracy of data [20] from different air handling units by collecting data from a large number of customer air handling units.



## 8. REFERENCES AND LITERATURE

[1] Author: SALDA UAB

Title: Technical manual [EN]

Date of Access: 2016 12 29

Internet access:

[http://www.salda.lt/rest/techpdf/techPDFdown/Smarty%203X%20P\\_P0094\\_AZ\\_0002.pdf](http://www.salda.lt/rest/techpdf/techPDFdown/Smarty%203X%20P_P0094_AZ_0002.pdf)

[2]: Author: SALDA UAB

Title: SALDA AIR

Date of access: 2018-04-23

Internet Access: <https://play.google.com/store/apps/details?id=lt.salda.air>

[3] Author: SALDA UAB

Title: MB-gateway specification

Date of Access: 2017 04 05

Internet Access: <http://salda.lt/content/download/MB%20gateway.pdf>

[4] Author: Simply modbus website.

Title: Modbus FAQ

Date of Access: 2017 12 21

Internet Access: <http://www.simplymodbus.ca/FAQ.htm#Modbus>

[5] Author: IEEE

Title: About the IEEE Internet of Things (IoT) Initiative

Date of Access: 2018 04 14

Internet Access: <https://iot.ieee.org/about.html>

[6]: Author: IEEE

Title: Special Report: The Internet of Things

Date of access: 2018 04 14

Internet Access: <http://theinstitute.ieee.org/static/special-report-the-internet-of-things>

[7]: Author: KATHY PRETZ

Title: Smarter Sensors

Date of access: 2018 04 14

Internet Access: <http://theinstitute.ieee.org/technology-topics/internet-of-things/smarter-sensors>

[8] Author: Google

Title: Overview of Internet of Things

Date of Access 2016 01 18

Internet Access: <https://cloud.google.com/solutions/iot-overview>

[9] Author: Jim Caldwell Director, IBM Internet of Things, Continuous Engineering Solutions Development

Title: IBM Internet of Things Point of View and Strategy

Date of Access: 2016 12 30

Internet Access: <http://toronto.ieee.ca/files/2016/02/TOR-IEEE-IBM-IoT-Jan-28-2016-Final.pdf>

[10] Author: Kittur Nagesh

Title: Cognito networks Powering IoT for a smarter enterprise

Date of Access: 2016 12 30

Internet Access: <http://iot-2016-winter.ieeesa-events.org/wp-content/uploads/sites/21/2016/11/cognito-presentation.pdf>

[11]: Author: OJ Air Cloud

Title: OJ Air Cloud

Date of access: 2018-04-23

Internet Access: <https://www.ojelectronics.com/Business-Areas/HVAC-Controls-Power/Products/OJ-Air-Cloud-PROD415.aspx>

[12] Author: Systemair

Title: SyScrew 440-1550 Water EVO NEW

Date of Access: 2018 02 25

Internet Access:

[https://www.systemair.com/globalassets/documentation/43719.pdf?filename=CWWE1130RC-EDM\\_SyScrew\\_440-1550\\_Water\\_EVO\\_GB.pdf](https://www.systemair.com/globalassets/documentation/43719.pdf?filename=CWWE1130RC-EDM_SyScrew_440-1550_Water_EVO_GB.pdf)

[13] Author: Nick Diakopoulos and Stephen Cass

Title: The top programming languages 2016

Date of access: 2016 01 08

Internet Access: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016> AND [http://spectrum.ieee.org/ns/IEEE\\_TPL\\_2016/methods.html](http://spectrum.ieee.org/ns/IEEE_TPL_2016/methods.html)

[14] Author: Eugeniya Korotya

Title: 5 best javascript frameworks in 2017

Date of Access: 2017 12 20

Internet Access: <https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282>

[15] Author: Robert Mora

Title: Angular 2 pros and cons

Date of Access: 2017 12 20

Internet Access: <http://www.pro-tekconsulting.com/blog/advantages-disadvantages-of-react-js/>

[16] Author: -

Title: Ember.js vs reactJS

Date of Access: 2017 12 20

Internet Access: [https://www.slant.co/versus/34/10513/~ember-js\\_vs\\_react](https://www.slant.co/versus/34/10513/~ember-js_vs_react)

[17]: Author: Oliver Örnmyr

Title: Performance comparison of XHR polling, Long polling, Server sent events and Websockets

Date of access: 2018 04 15

Internet Access: <http://www.diva-portal.org/smash/get/diva2:1133465/FULLTEXT01.pdf>

[18]: Author: Craig Buckler

Title: SQL vs NoSQL: The Differences

Date of access: 2018-04-15

Internet Access: <https://www.sitepoint.com/sql-vs-nosql-differences/>

[19] Author: SALDA UAB

Title: MB-gateway technical manual

Date of Access: 2017 04 05

Internet Access: [http://salda.lt/content/download/MB-Gateway\\_quick%20guide\\_P0114\\_AZ\\_0002.pdf](http://salda.lt/content/download/MB-Gateway_quick%20guide_P0114_AZ_0002.pdf)

[20]: Author: Andrii Diachenko

Title: Automation of handling of the ventilation chamber data, the challenges of the received data inaccuracies, 2017, Šiauliai, pp. 74.

## Explanation of cloud web database:

1. Users – main account database table, this table consists of:
  - a. Id – unique user account identifier;
  - b. Username – account login name;
  - c. Password – account password;
  - d. Email – accounts email (used to contact account owner or recover account);
  - e. Role\_id – accounts role (used to determinate account permissions);
  - f. Join\_date – account creation date;
2. FailedLogins – table used to track failed logins on given account, table column explanations:
  - a. User\_id – given unique user identifier for which logins are counted;
  - b. Count – how many failed logins have occurred so far (max 3 logins allowed in 15 minutes);
  - c. Last\_attempt – date on last failed login attempt;
3. UserRecovery – table used to allow users to recover their accounts, it contains:
  - a. User\_id – unique user account identifier;
  - b. Recovery\_hash – hash used to go account recovery page;
  - c. Date – date of recovery hash creation (hash is only valid for 15 mins.);
4. UserRoles – this table contains possible user roles, table structure:
  - a. Id – role identifier;
  - b. Name – role name;
5. Gateways – table where all gateways are registered and managed, its structure:
  - a. Id – unique gateway identifier;
  - b. Name – gateway name that is set during registration;
  - c. MAC – unique gateway mac address;
  - d. Status\_id – current gateway status (online, offline);
  - e. Users\_id – id of owner of given gateway;
6. Slaves – table where all gateways slaves (air handling units) addresses are held, table structure:
  - a. Id – unique slave identifier;
  - b. Gateway\_id – slaves owner identifier;
  - c. Ahu\_name – slave name (air handling unit name);

- d. Board\_type – which control board for given slave (PRV or MCB);
  - e. Board\_address – which board address slave is in;
7. Commands – table used to hold commands used to request data from gateways, table structure:
- a. Id – unique command identifier;
  - b. Command\_code – command code received from gateway (generated code used for extended modbus protocol);
  - c. Gateway\_command\_id – which command to execute (command from extended Modbus protocol);
  - d. Priority – command priority, higher priority, sent first;
  - e. Gateway\_id – id of gateway to send the commands to;
  - f. Board\_address – address of slave (air handling unit) to send the command to;
  - g. Modbus\_function\_code – Modbus function of Modbus protocol;
  - h. Start\_address – start address of register to read/write (null when executing other commands);
  - i. Address\_count – how many registers to read/write (null when executing other commands);
  - j. Address\_values – values to write (null when executing other commands);
  - k. Date – time when the command was added to table;
  - l. Text – text to send (sometimes need to send text for settings);

Explanation of air handling unit data collection database:

1. CommandsForHistory – table is used to manage commands to send to gateways, table structure:
  - a. Id – unique command id;
  - b. Modbus\_function\_code – Modbus function code of Modbus protocol;
  - c. Start\_address – address of first register to read;
  - d. Address\_count – how many registers to read;
  - e. Last\_sent – date when given command was sent;
  - f. Send\_after\_seconds – time in seconds, on how often to send given command;
2. RegisterHistory – this table is used to collect register data received from gateway, data is collected here and then processed into different tables, here is the table structure:
  - a. Id – unique table identifier;
  - b. Modbus\_function\_code – Modbus function code of Modbus protocol;
  - c. Address – address of register;
  - d. Value – value of register;
  - e. Salve\_id – slave (air handling unit) that given register value is read from;
  - f. Date – time register was read;
3. TemperatureHistory – this table contains processed Registerhistory table data, here is its structure:
  - a. Id – unique history id;
  - b. Mode – currently activated air handling unit mode (0 – off, 1 – building protection, 2 – economy, 3 – comfort mode);
  - c. Fresh – temperature from outside entering air handling unit;
  - d. Supply – temperature from outside entering building;
  - e. Extract – temperature from room entering air handling unit;
  - f. Exhaust – temperature from room exiting air handling unit to outside;
  - g. Date – time data was read;
  - h. Salve\_id – slave (air handling unit) that data was read from;
4. AlarmHistory – this table contains alarm history values that are processed from RegisterHistory, it only contains active alarms at given time, here is table structure:
  - a. Id – unique history identifier;

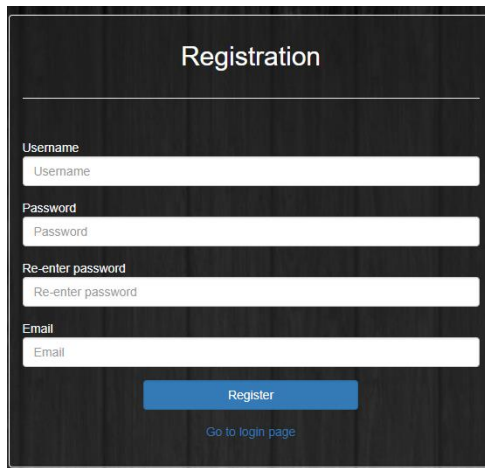


- b. Alarm\_id – active alarm id (each alarm has its unique id and explanation);
- c. Date – time when alarm is on;

Below is the basic cloud user interface manual, it explains most important cloud concepts.

Cloud registration is done in these steps:

1. Go to login page, click register button;
2. Fill in the registration form (figure below);
3. Click register, you will registered and redirected to home page;



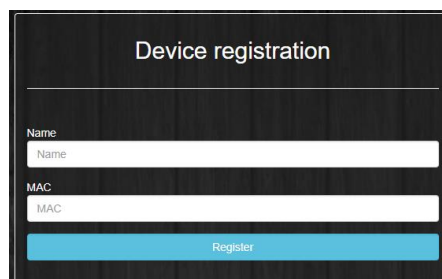
The image shows a registration form titled "Registration" on a dark background. It contains four input fields: "Username", "Password", "Re-enter password", and "Email". Below the fields is a blue "Register" button and a link that says "Go to login page".

Cloud navigation menu explanation (figure below):

1. Dashboard – home page;
2. Devices – devices list, where you can go to ventilation control;
3. Analytics tools – collected data charts;
4. Account – change account settings, register devices and their slaves;



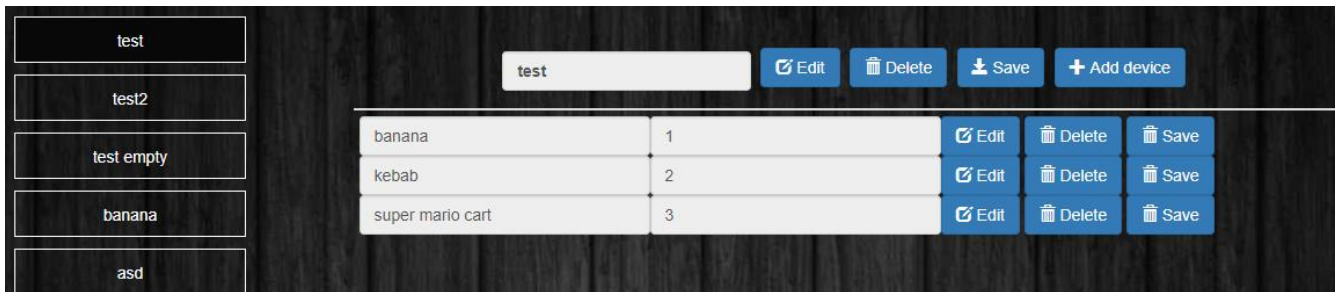
MB-gateway registration page, this allows client to register new device by its MAC address:



The image shows a device registration form titled "Device registration" on a dark background. It contains two input fields: "Name" and "MAC". Below the fields is a blue "Register" button.

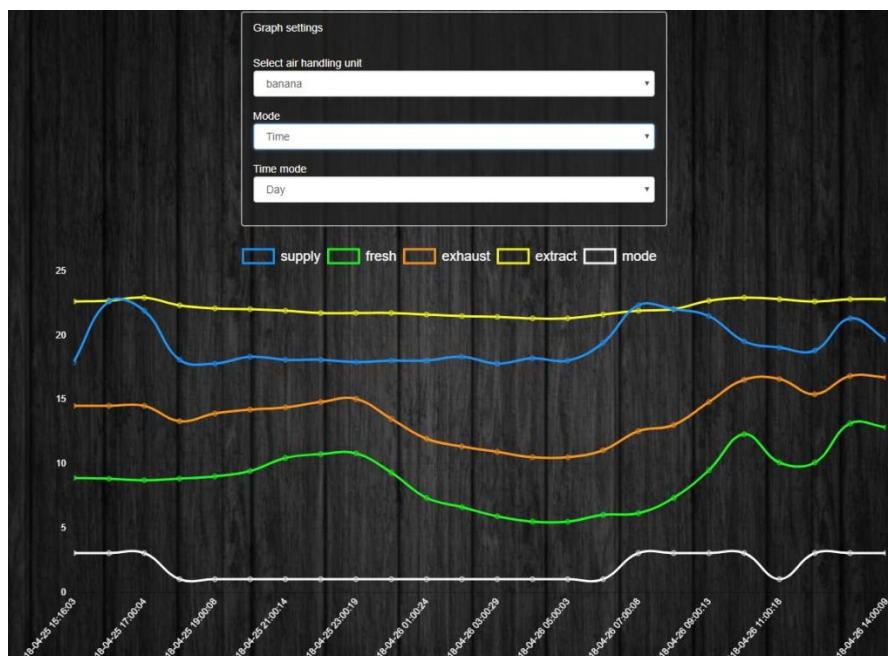
Device management page, allows you to (figure below):

1. View all your MB-gateways and their air handling units;
2. Delete air handling units from list;
3. Add air handling units to list;
4. Change air handling unit addresses and settings;

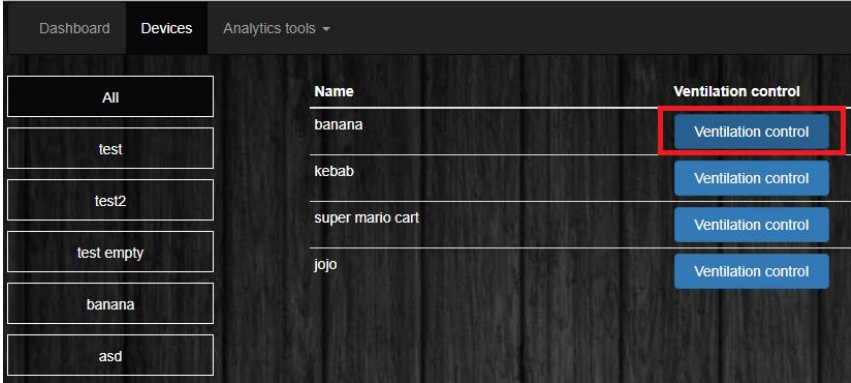


Analytics tools temperature history page allows you to view collected temperature data graphically by these rules (figure below):

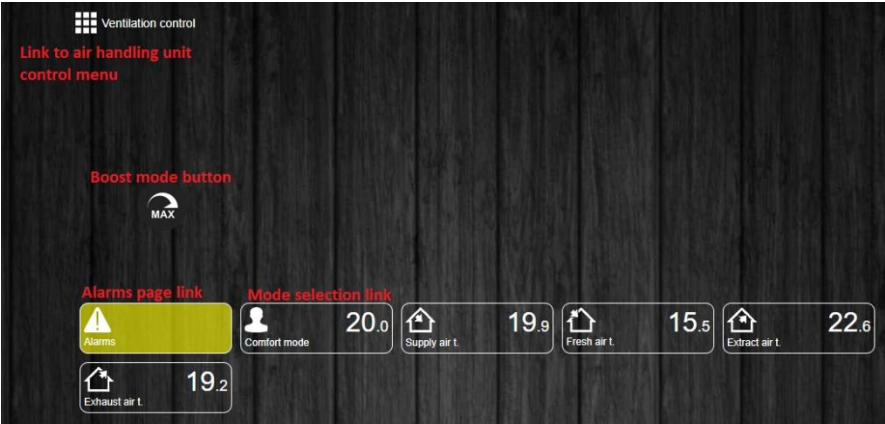
1. Selected air handling unit;
2. Selected mode;
3. Last selected interval;
4. Selected date range;



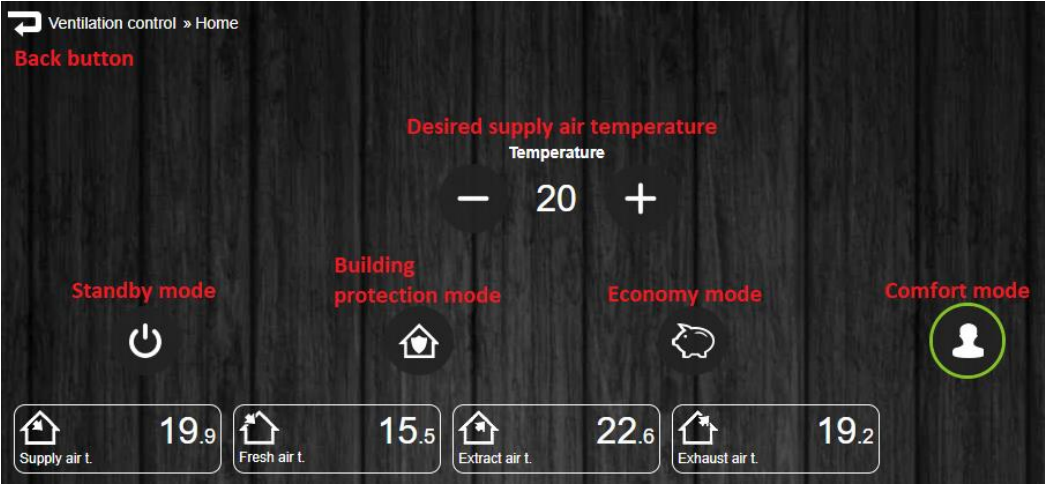
Air handling unit control page can be accessed by clicking on ventilation control button at devices page (picture below).



After entering ventilation control, air handling unit control panel home page is accessed (figure below);



Air handling unit mode selection, can be done by clicking mode selection link in the home page, which will redirect you to mode selection page (figure below);



Clicking the menu button in the home page, you will be redirected to menu page, where you can navigate to:

1. Boost max timer setup – allows to set default boost activated tower;
2. Night cooling – night cooling settings;
3. Date time – date time settings of air handling unit;
4. Filters timer – filters settings, which allows to know how your filters are doing;
5. Economy mode – default economy mode temperature set point;
6. Building protection – default economy mode temperature set point;
7. Heating season – winter/summer mode settings;
8. CO2 level – co2 settings if co2 sensor is available;
9. Humidity – humidity settings;
10. Status – current air handling unit software/firmware settings;
11. Schedule – custom air handling unit control schedule;
12. Other – other parameters of air handling unit;

