

FMH606 Master's Thesis 2022
Industrial IT and Automation

IoT based data logger system for NOMAD sewage units with focus on efficiency and performance



**NOMAD vacuum unit
with pumps and necessary piping**

Edith Mari Flaten

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Course: FMH606 Master's Thesis, 2022

Title: IoT based data logger system for NOMAD sewage units with focus on efficiency and performance

Number of pages: 52

Keywords: IoT, Vacuum unit, Azure, Cloud platform, SQL database, User interface, Frontend, Backend, RevPi

Student: Edith Mari Flaten

Supervisor: Ru Yan, Saba Mylvaganam

External partner: Jets Vacuum AS – Tor Rønnestad

Summary:

NOMAD unit is a sanitary infrastructure released by Jets™ in 2022. The aim of this thesis is to look at the feasibility of data logging from a mobile site of the NOMAD unit to a cloud platform or external location.

This thesis is the preliminary work for looking at what possibilities there are in the world today and starts by looking at existing hardware, software, data platforms, and cybersecurity.

Got the RevPi setup and successfully communicated with the PLC.

In the backend solution, the RevPi succeeded in communicating with the PLC system and got a database setup. In the front-end solution, it managed to connect the GUI and the SQL database, which is at the backend. In addition, it was used integrated solutions in a cloud platform on Azure. Cybersecurity needs to be taken into consideration in any future solution.

The solution is technically viable. However, more work is needed to be taken out to ensure cybersecurity.

Further, there is a need to discover if it is an option to take it further for Jets™ on an economical and practical level.

Preface

I am an online master student, at the University of South-Eastern Norway, studying Industrial IT and Automation part time while working at Jets Vacuum AS. When the time came for selecting a master thesis topic, I got the opportunity to do a work-related topic.

I have been a part of the creation of the NOMAD units which this thesis is built upon, as a product manager for electro and being a part in the creation of the electrical drawings.

It is assumed that the reader has some technical knowledge, to get the most out of reading the thesis.

I want to give thanks to Ru Yan and Saba Mylvaganam for their support during the semester, to my good colleagues Ole Christian Marti, Alper Duran and Tor Rønnestad for technical support, and to Jets Vacuum giving me the possibility of a work-related master topic.

Programs used in the thesis are: STARuml, Microsoft Word; Excel; Teams; SQL Server Management Studio; Visio Professional; Azure SQL database; Power Apps and Project, Zotero, erwin Data Modeler Academic Edition, Notepad++.

The image on the frontpage is from the NOMAD vacuum unit's datasheet, and shows the vacuum unit with pumps, piping and required components.

Ålesund, 18.05.2022

Edith Mari Flaten

Contents

1 Introduction	6
1.1 Background	6
1.1.1 <i>Problem Description</i>	6
1.1.2 <i>New Developments in the Project</i>	6
1.2 Objectives	6
1.3 Report Structure	7
2 System Description – NOMAD Units	8
2.1 System diagram	8
2.2 Vacuum Unit Functionality	12
2.3 Communication System for NOMAD units	17
2.4 Data Collection from Vacuum Unit	17
2.5 Hardware and Software	17
3 Methods	18
3.1 Data collection and communication system	18
3.1.1 <i>Hardware for data transformation and data transfer</i>	18
3.1.2 <i>Software for Backend and Frontend Solutions</i>	20
3.1.3 <i>Azure IoT hub</i>	21
3.2 Cloud and Edge Service for IIoT Solutions	22
3.3 Feasibility of Having Wired and/or Wireless Communication	23
3.4 Categories/select process data to transfer	23
3.5 Database design	25
3.6 Cybersecurity Risk Analysis for IoT Systems	26
4 System configuration and Model Development	28
4.1 Implement hardware for data transformation and data transfer	28
4.1.1 <i>Experimental/testing setup</i>	28
4.1.2 <i>Hardware/software setup RevPi</i>	29
4.2 Develop a cloud-based platform	35
4.2.1 <i>Database Setup and Implementation</i>	35
4.2.2 <i>Frontend for Data Visualization, using Graphical User Interface (GUI)</i>	42
4.3 Cybersecurity for IoT systems	47
4.4 System Integration and Testing its Efficiency and Performance	47
5 Results and Discussion	48
5.1 Hardware Implementation	48
5.2 Developed platform	48
5.3 Cybersecurity	48
5.4 System Integration	48
5.5 Are results as expected?	48
6 Conclusion	49
6.1 Possible Future Work	49

Nomenclature

AI – Artificial Intelligence

bar – *bar*, used here for relative pressure, 1 bar is $100kPa$

DTU – Database Transaction Unit

EEA – European Economic Area

GA – General Arrangement

GDPR – General Data Protection Regulation

GUI – Graphical User Interface

HMI – Human-Machine Interface

MQTT – MQ Telemetry Transport

IoT – Internet of Things

IIoT – Industrial IoT

NA – Not Applicable

OEM – Original Equipment Manufacturer

OOAD – Object Oriented Analysis and Design

OPC UA – Open Platform Communications United Architecture

PAN – Personal Area Network

Pascal – Pa , SI unit of pressure, newton per square meter N/m^2 , $Pa = kg * m^{-1} * s^{-2}$

P&ID – Piping and Instrumentation Diagram

SCADA – Supervisory Control and Data Acquisition

SSMS – Microsoft SQL Server Management Studio

TCP – Transmission Control Protocol

UI – User Interface

UL – Underwriters' Laboratories

USN – University of South-Eastern Norway

1 Introduction

The introduction looks at the background, the objectives, and the report structure of the thesis.

1.1 Background

This chapter looks at the problem description, literature survey, previous work, and new developments in the project.

1.1.1 Problem Description

In February 2022, Jets Vacuum released Jets™ NOMAD units, a scalable mobile infrastructure meant for events, festivals, and others.

Looking to the future, it can be relevant with an IoT solution for the transfer of process data and alarms, for predictive maintenance and to lower the response time in case of alarms. This can give customers the option of upgraded units with an IoT solution and a service package. The thesis investigates the possibility of real-time data transmission from Jets™ NOMAD units. Together these units create a scalable mobile sanitation system, that be quickly established on any site [1].

For the thesis, the focus will be on the vacuum unit and what options lay there. The full task description of the thesis can be found in **Appendix A**.

1.1.2 New Developments in the Project

This report is the beginning works of looking at connecting the NOMAD units to a remote connection. The project can give the possibility of getting process data from a mobile structure location (a site) to a centralized location.

1.2 Objectives

The project description has a general view of NOMAD units, and this report narrows it down to focus on the vacuum unit.

Here are some of the main bullet points from the project description, found in **Appendix A**, with a focus on the vacuum unit:

- Get an overview of the existing data collection and communication system in the NOMAD units, with a focus on the vacuum unit
- Investigate the feasibility of having wired or wireless external/remote communication with the vacuum unit
- Categories process data to be transferred and implement hardware for data transformation and data transfer, including IoT solutions.
 - o Take a selection of variables from the vacuum unit for test/check of a functioning dataflow/communication
- Develop an “on-premises” and/or cloud-based platform, including:
 - o Backend for data transfer, storage, and necessary processing

- Frontend for data visualization using a graphical user interface (GUI).

The objective of the thesis is to look at the feasibility, and any selected setup will reflect this. For a final commercial product, the setup will need further optimization.

1.3 Report Structure

The report is structured as follows:

Chapter 1 introduces the background, objectives, and report structure.

Chapter 2 includes system diagram, functionality explanation and describes the existing communication system, data collection, hardware, and software.

Chapter 3 evaluating and selecting suitable hardware, study the feasibility of wired/wireless communication protocols, what data should be extracted from the NOMAD unit(s), database design and security risk analysis for IoT systems.

Chapter 4 implementing/applying hardware for data transformation and transfer, development of a cloud-based platform, cybersecurity for IoT systems, and system integration and testing the system's efficiency and performance

Chapter 5 present the outcome of hardware implementation, the developed platform (backend and frontend/software), cybersecurity and system integration. It debates the results and if they are as expected and discusses models and cybersecurity.

Chapter 6 is the conclusion of the report

After a list of references, the report has some documents of relevance related to programs developed and the thesis description with the tasks assigned.

2 System Description – NOMAD Units

This chapter gives a description of the NOMAD unit system, how it functions and the existing hardware and software.

2.1 System diagram

The NOMAD water and wastewater infrastructure have a low water consumption, can connect to the main sewage system, and lower the logistical cost [1].

Figure 2.1 shows the system overview diagram over the different NOMAD units, which are numbered, showing possible connections. The water supply unit (1) supplies the system with fresh water and gives protection against pollution of the water supply source, the catcher unit (2) is to separate out foreign objects in the wastewater, and the vacuum unit (3) creates the vacuum needed to transport the wastewater and macerates the sewage/wastewater. The flow control unit (4) feeds the wastewater to local sewage line by gravity and has a vent to reduce any excess pressure. A transfer unit (5) can be used in combination with a flow control unit when there is an increased distance to a local sewage line connection point [1].

The units are connected by piping and have the possibility of connecting via hardwired cables and/or ethernet.

Figure 2.2 illustrates a simplified piping and instrumentation diagram (P&ID) for the vacuum unit, showing the relevant components. This unit is the focus of the thesis, and we will not look at the connection options to other NOMAD units at this stage.

2 System Description – NOMAD Units

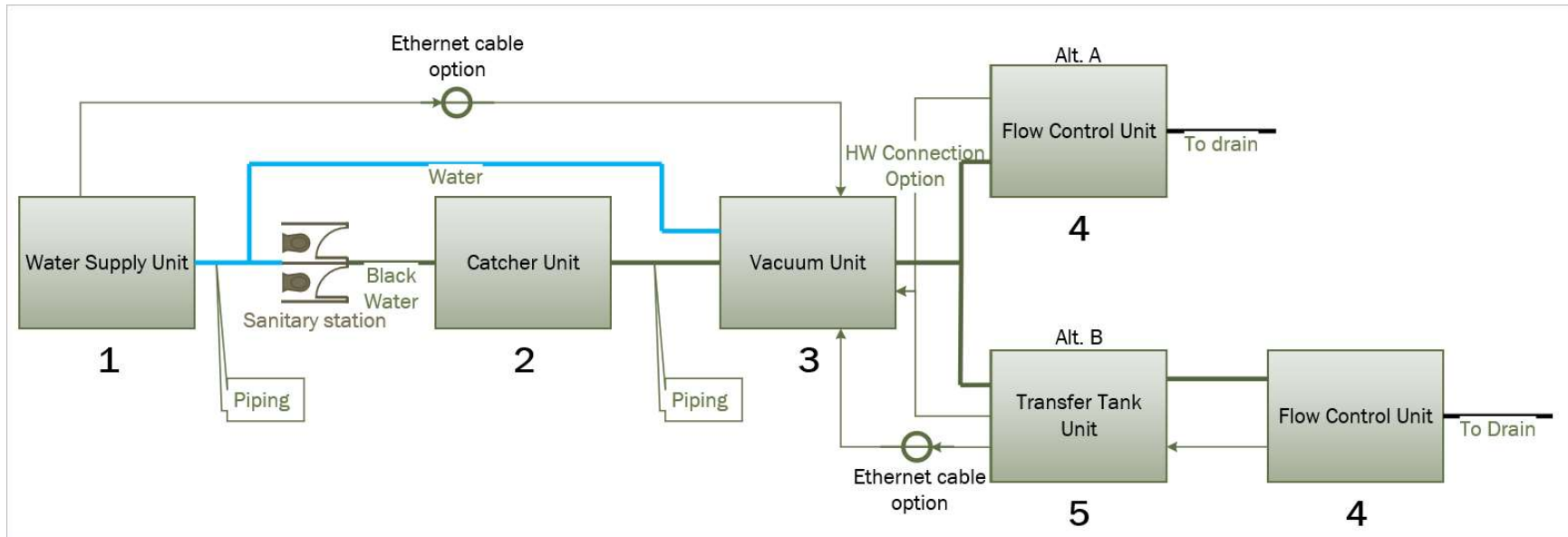


Figure 2.1: System overview of the diverse types of NOMAD units. 1: Water supply unit; 2: Catcher unit; 3: Vacuum unit; 4: Flow control unit; 5: Transfer tank unit

2 System Description – NOMAD Units

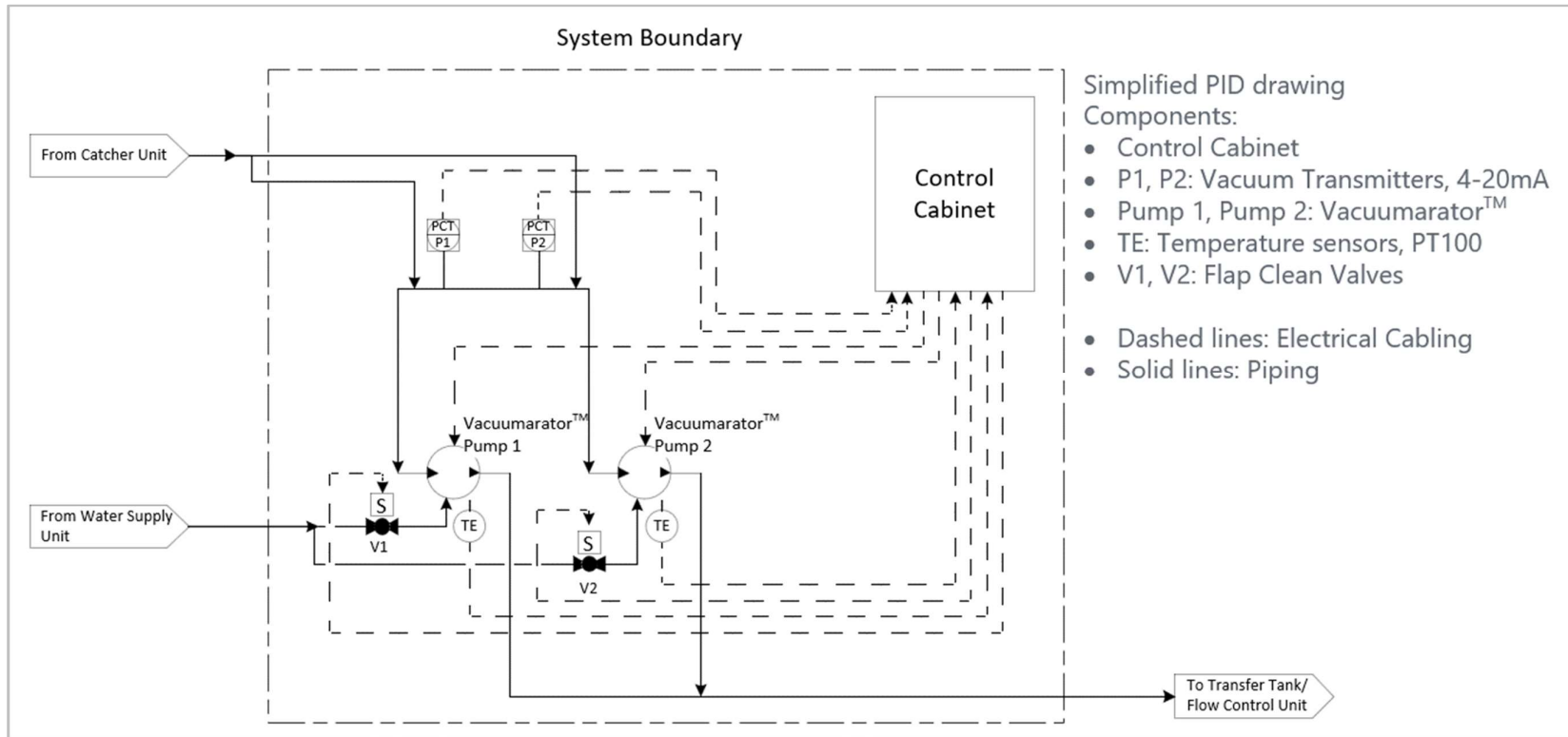


Figure 2.2: P&ID diagram for NOMAD vacuum unit with details of electrical cabling and piping

2 System Description – NOMAD Units

The relevant components in Figure 2.2, are vacuum transmitters, pumps, temperature sensors, solenoid valves and the control cabinet. The symbols of other components, such as vacuum switch, different gauges, and connections to other units, have been omitted on purpose.

The two vacuum transmitters, P1 and P2, give analog signals (4-20mA), where the raw value of 4mA indicates 100% vacuum, 20mA indicates 0% vacuum, and it is assumed to be a linear proportion between these two points [2]. Further, 0% vacuum is relative 0bar, and 100% vacuum is relative -1bar. Figure 2.3 shows the symbol used in the P&ID diagram for vacuum transmitters. Figure 2.4 shows the location of the vacuum transmitters on the manifold. A vacuum transmitter is the same as a pressure transmitter, but vacuum is used to differentiate between sensors measuring below (vacuum) and above (pressure) 0bar.

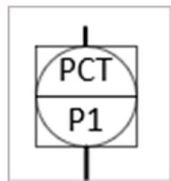


Figure 2.3: Vacuum transmitter symbol in the P&ID

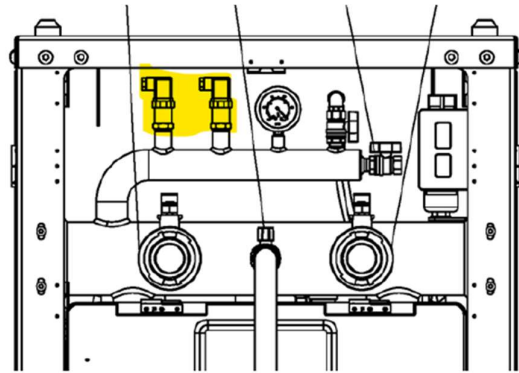


Figure 2.4: The locations of the two vacuum transmitters on the manifold is marked with yellow [2]

The Vacuumarator™ pumps create vacuum in the piping system and macerates the wastewater and discharge [3]. Figure 2.5 shows the pump symbol used in the P&ID diagram. Figure 2.6 shows the large version of a Vacuumarator™ pump, where the inlet is on the top on the left side and the outlet is on the top of the right/middle side, and the motor is on the far right.

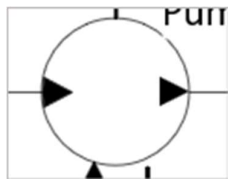


Figure 2.5: Pump symbol in the P&ID

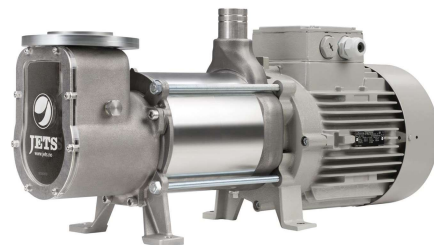


Figure 2.6: Vacuumarator™ Edge L pump [3]

Temperature sensors are PT100 elements and are connected to the programmable logic controller (PLC) in the control cabinet. They are part of the system to avoid overheating the pumps and allow for longer runtime on the pumps. The temperature sensor is represented by the symbol shown in Figure 2.7.

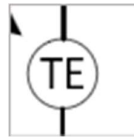


Figure 2.7: Temperature sensor symbol in the P&ID

The solenoid valves (flap cleaning valves) are connected to freshwater. They are there to flush the inlet flap of the Vacuumator™ pump of anything clogging it and to be able to cool down the pump in case of temperatures above the threshold. Figure 2.8 shows the solenoid valve symbol.

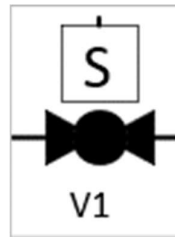


Figure 2.8: Solenoid valve in the P&ID

The control cabinet is manned via a Human-Machine Interface (HMI) panel and contains the needed components for controlling the pumps. Figure 2.9 shows the symbol used in the P&ID diagram, while Figure 2.10 shows the control cabinets outside look and the placement on the vacuum unit.



Figure 2.9: Control cabinet symbol in the P&ID



Figure 2.10: Control cabinet placement on the NOMAD vacuum unit, as shown in the general arrangement (GA) drawing [2]

2.2 Vacuum Unit Functionality

A vacuum unit controls the pressure level in the piping system of a sanitary system, for removal of sewage by vacuum, by using Vacuumator™ pumps to create vacuum and macerate the sewage. These pumps are normally set to start at 42/40% vacuum and stop at 50% vacuum. The vacuum is read by vacuum transmitters, or vacuum switches depending on vacuum unit

2 System Description – NOMAD Units

type, and should the vacuum level go below 25% for more than 10 seconds, an alarm is sounded. A low vacuum alarm indicates an issue that needs fixing, be it a leak in the piping, a pump not functioning as it should, etc.

For continued uptime of the system, the unit will start up automatically after a power loss or similar and not need a manual restart. This is if the switches/buttons are set to auto.

The NOMAD vacuum unit is normally operated via an HMI panel. The pumps can be operated manually in case of maintenance, emergency, or other needs. They can be manually operated from the HMI or by bypassing the PLC. When the PLC is bypassed, the pumps are run parallel and controlled by a vacuum switch and two potentiometers, one potentiometer per pump.

The HMI system has four (4) different access levels, OPERATOR, SUPERVISOR, SERVICE ENGINEER and ADMINISTRATOR. This aids in avoiding “unauthorized” changes to setpoint values, while still having the pumps properly operated. Actions are limited according to the settings of the access level.

Access level 0 does not require any login information that the other three levels do. Each subsequent level has access to the same functions as the one below and has some additional functionality. OPERATOR is access level 0 and is for normal operation of pumps, and there is no login required. Operators can access start/stop and manual/auto function for pumps and manual flap clean. SUPERVISOR is access level 1 and can change set points and alarm levels for the daily operation of pumps. SERVICE ENGINEER is access level 2 and has extensive control of all alarm adjustments, etc. ADMINISTRATOR is access level 3 and can create new users and passwords.

In Auto mode, the standard setup for the NOMAD vacuum unit is for the PLC to read the vacuum level from a vacuum transmitter and start and stop pumps according to the vacuum level in the system and, if needed, according to the master pump running time. Figure 2.11 for the flow chart.

Listed below are the factory setting levels [2]:

- Vacuum setpoint: 50%
- Master Pump Start Point: 42%
- Standby Pump start point: 40%
- Master Pump Stop Offset: 2%
- Standby Pump Stop Offset: 2%

If the master pump is running after a certain time without reaching the vacuum setpoint, and vacuum level does not go down below the start point of the standby pump, the standby pump will start. This ensures that the master pump will not run endlessly.

If the vacuum gets below the low vacuum set point, factory setting 25%, for a set amount of time, the factory setting is 10 seconds, then the low vacuum alarm is activated. No impact on the operation of vacuum pumps. See Figure 2.12 for the flow chart of low vacuum alarm.

If a pump runs for a certain amount of time, or the temperature gets above 50°C, the flap cleaning valve is activated. This is done to prevent any clogging and/or dry running. See Figure 2.13 for the temperature control flow chart for the flap cleaning valves.

2 System Description – NOMAD Units

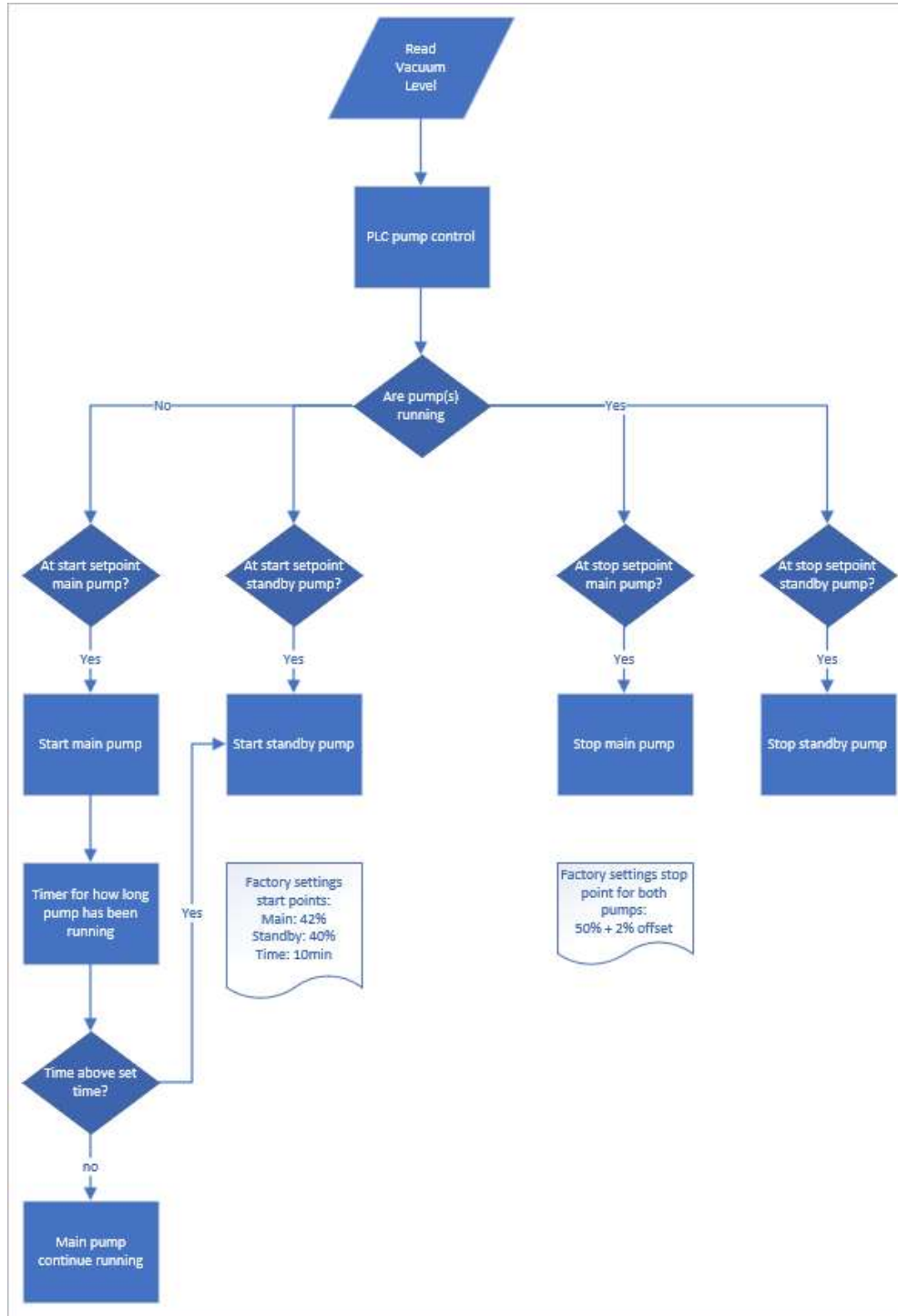


Figure 2.11: Pump control flow chart with sensors (vacuum transmitters) and actuators (pumps)

2 System Description – NOMAD Units

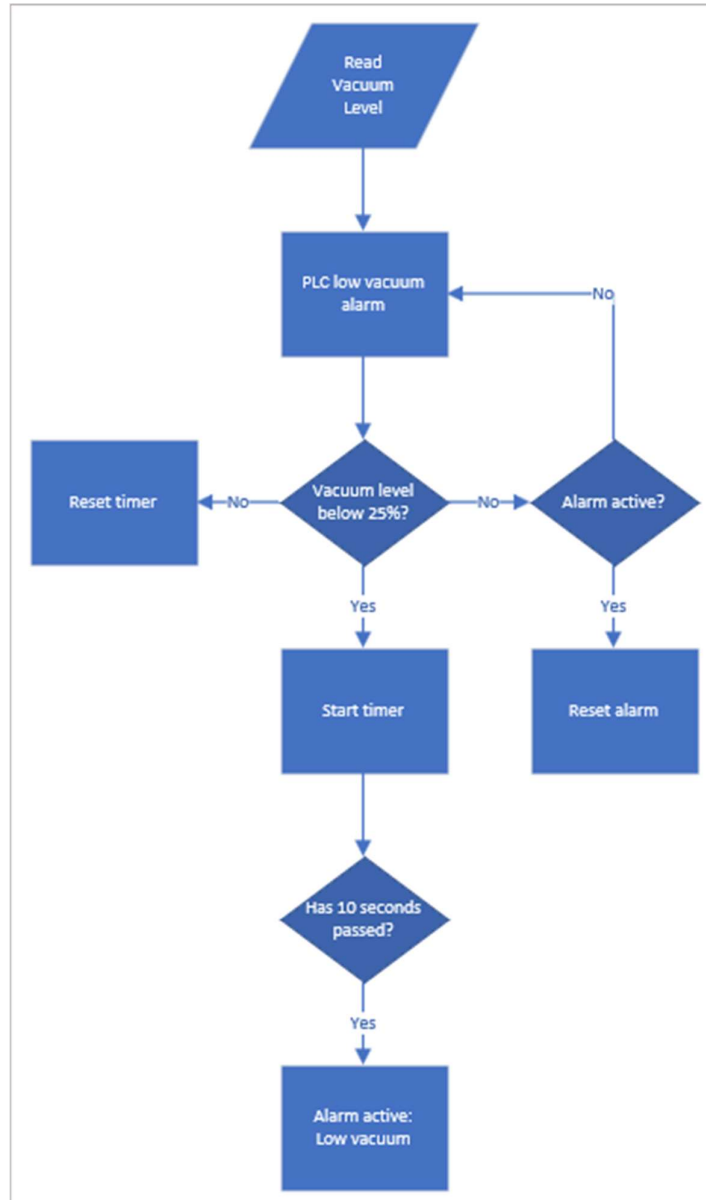


Figure 2.12: Low vacuum alarm flowchart based on vacuum level measurements

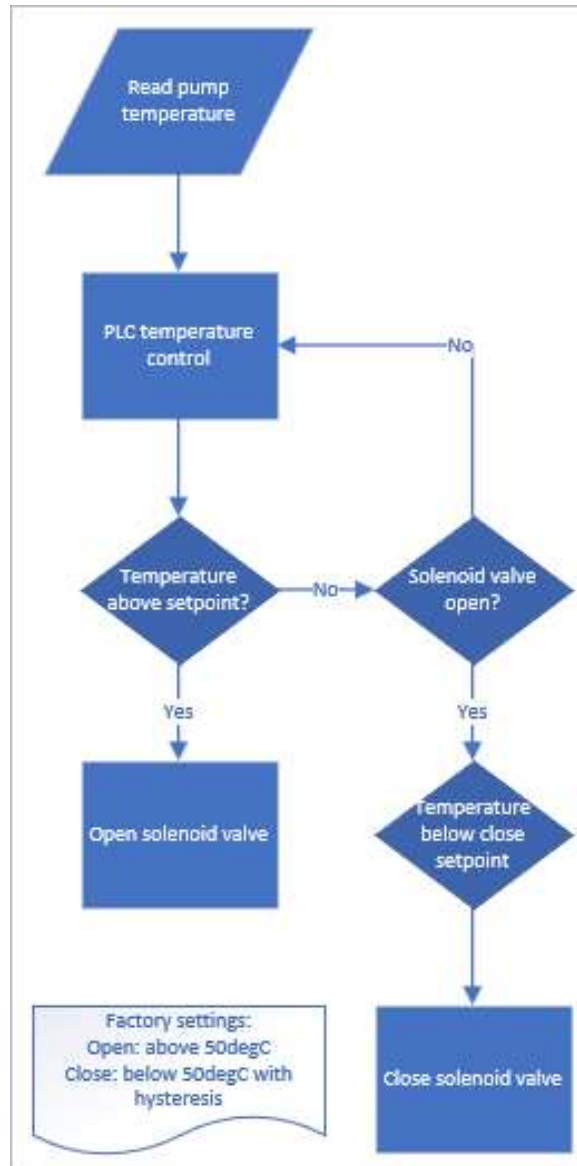


Figure 2.13: Flow chart for flap cleaning control based on temperature sensors

2.3 Communication System for NOMAD units

In the existing setup, it is possible to connect units for local communication, and there is an option for connection of an external system.

The units can be interconnected locally by hardwired cables, ethernet cables or both. This allows for one unit to give an alarm for another unit, and/or stop depending on the alarm/signal.

There is an option for an external system to remotely connect to the vacuum unit. This is mainly for reading data, but the units have been set up for the possibility of changing threshold values in a future connection to a Jets™ supervisory control and data acquisition (SCADA) system [2].

The water supply unit and the transfer unit have the option of connecting to the vacuum unit with an ethernet cable and/or hardwired cable.

The water supply connection allows for an option for having the vacuum unit to give an alarm about low pressure (below 1bar) on the outlet of the water supply unit. This is in addition to the alarm on the water supply unit.

The transfer unit connection allows for an option for having the vacuum unit to give an alarm signal for high high level in tank and a control signal to check if the cable is connected and functioning. This is in addition to the alarm on the transfer unit.

The flow control units have an option for wired connections. The wired connection adds an alarm signal for high high level in tank and a control signal to check if the cable is connected and functioning.

2.4 Data Collection from Vacuum Unit

In the existing setup data can be collected manually when the unit is setup for it. This method applies not only to the vacuum unit, but also the water supply unit and any transfer units. Each day gets a separate/new log file for the occurring and disappearing events, with timestamps, taking place for the day. Due to security risk, no more details will be given.

2.5 Hardware and Software

The NOMAD units are built using Schneider Electric hardware and software where possible.

The units use a combination of Modbus TCP, CANopen and TCP IP for communication between the PLC, HMI, and frequency drives.

PLC's, HMI's, and frequency drives are all Schneider Electric components, and they are programmed by Jets™ personnel using Schneider Electric software. The PLC program was created using EcoStructure Machine Expert and the HMI program was created using Vijeo Designer. The frequency drives get changes implemented by service engineers on the HMI.

3 Methods

This chapter is about selecting software and hardware components, methods and related background used for the different elements. In general, it is about what is needed to solve the different tasks found in the thesis description.

3.1 Data collection and communication system

This chapter will first look at hardware components available at JetsTM and software programs are available through JetsTM or USN or is a standard.

JetsTM have invested in RevPi Connect and bought Acksys Airbox Router for testing purposes for other internal projects. The idea for another project was to see if MQ telemetry transport (MQTT) can be used as the messaging protocol and Node-RED as the programming language on the RevPi.

3.1.1 Hardware for data transformation and data transfer

The hardware that can be used to transform and transfer data, available at JetsTM, is RevPi Connect and Acksys Airbox Router. Supplementary hardware that can be used for/during testing is power supply, PLC, HMI, simple ethernet switch, personal computer, and necessary cabling.

3.1.1.1 RevPi capabilities

The RevPi Connect is an industrial version of a Raspberry Pi, and has different types of interfaces, including 2xRJ45 10/100 ethernet ports. These ethernet ports can be used in a Modbus TCP network, as the RevPi have master/slave capabilities using Modbus network protocol. The RevPi supports MQTT, open platform communications united architecture (OPC UA), common Industrial Internet of Things (IIoT) protocols for direct transfer to the cloud, and applications can be programmed in different ways, e.g. Node-RED and Python [4].

The RevPi Connect+ is an Azure certified product that ensures easy integration with the Azure cloud platform [4]. Azure information can be seen in Figure 3.1, and Figure 3.2 shows the main differences. The device available at JetsTM is the RevPi Connect. However, there should be a possibility to connect to the Azure cloud platform relatively easily as the basis is the same for both. This should be taken into consideration in future work.

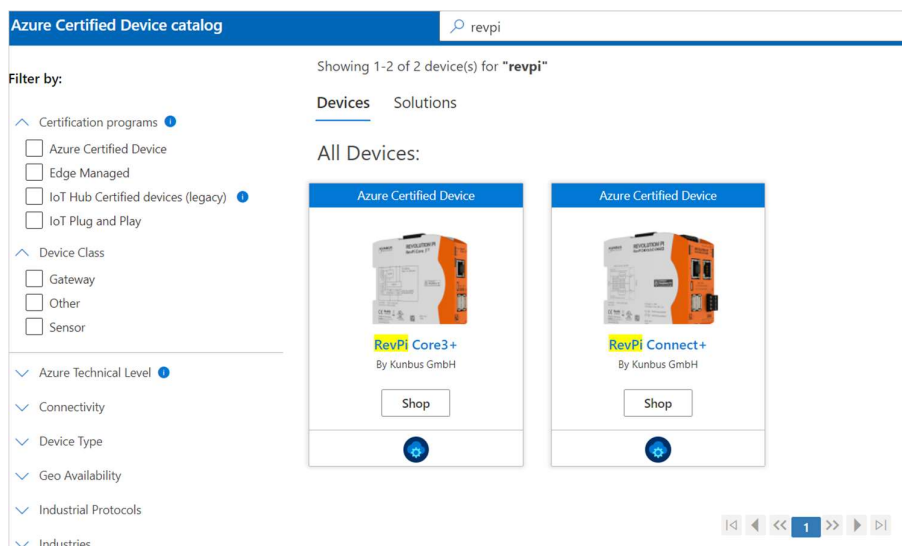


Figure 3.1: RevPi devices that are Azure certified [5]

Specifications	
Processor	Broadcom BCM2837B0 with Quad-Core ARM Cortex-A53 (RevPi Connect+) Broadcom BCM2837 with Quad-Core ARM Cortex-A53 (RevPi Connect)
Clock rate	1.2 GHz
RAM	1 GB
Storage (eMMC)	8 – 32 GB (RevPi Connect+) 4 GB (RevPi Connect)

Figure 3.2: The main differences between RevPi Connect and Connect+[4]

The PLC in the NOMAD vacuum has Modbus TCP network protocol enabled for a list of parameters/variables.

A possible expansion, should the RevPi be able to use the Schneider DataTransferTool command line from Vijeo Data Manager, is to extract the data logs stored on a unit and transfer them to the cloud. The data log files can then be used, e.g., as a backup option or for further data analysis if units have been out of reach for continuous transmissions.

3.1.1.2 Acksys AirBox router capabilities

The technology in the Acksys Airbox router means that it can be used worldwide, either by use of Wi-Fi and/or SIM card [6]. Figure 3.3 presents most of the communication options. This does not take into consideration different requirements in various regions and countries. For example, UL certification that may be needed for use of routers on the North American market has not been found. Should be investigated if it is needed.

Technical characteristics overview	
Ethernet interface	2-port Gigabit Ethernet 10/100/1000 auto-sensing, Base TX, auto MDI/MDIX, RJ45 Ethernet interface
Cellular interfaces + navigation	1 LTE radio category 4, 3GPP E-UTRA release 10, MIMO DL with Rx diversity
	Dual SIM
	LTE, UMTS/HSPA+, GSM/GPRS/EDGE (worldwide) Multi-constellation GNSS (GPS, Galileo, GLONASS, Beidou). Requires an active antenna.
Cellular radio data rate	150 Mbps ↓ & 50 Mbps ↑ (maximum radio data rate)

Figure 3.3: Communication options for Acksys Airbox router [6]

The Acksys Airbox router [6] is an industrial cellular router (2G/3G/4G) + WiFi (802.11n) that is an option for transmission of data from the RevPi and to the cloud.

3.1.2 Software for Backend and Frontend Solutions

A definition of backend and frontend [7]:

“Front-end development focuses on the visual aspects of a website - the part that users see and interact with. Back-end development comprises a site's structure, system, data, and logic. Together, front-end and back-end development combine to create interactive, visually pleasing websites.”

A combined backend and frontend solution should be able to handle storage, scaling, different geographical locations and be secure.

Elements that are a part of the backend solution is RevPi software, data transformation and transfer, and storage. The programs in the PLC and HMI will not be touched, and these are only to be read from to get data/variables and check if the communication is working with successful reading of data. A solution should also have the possibility, in the future, to add data analysis to the solution, e.g., machine learning or AI (Artificial Intelligence).

The frontend solution should also be relatively easy to use, as it can be used by people all over the globe. It should include the interface to the backend and the UI solution for people to interact with the system.

The backend solution in this thesis is all parts regarding gathering data from the PLC using RevPi, and the transfer from the RevPi and to storage. The frontend solution is the application/GUI displaying the stored data and any results from data analysis. Figure 3.4 shows the split between the backend solution elements and the frontend solution elements.

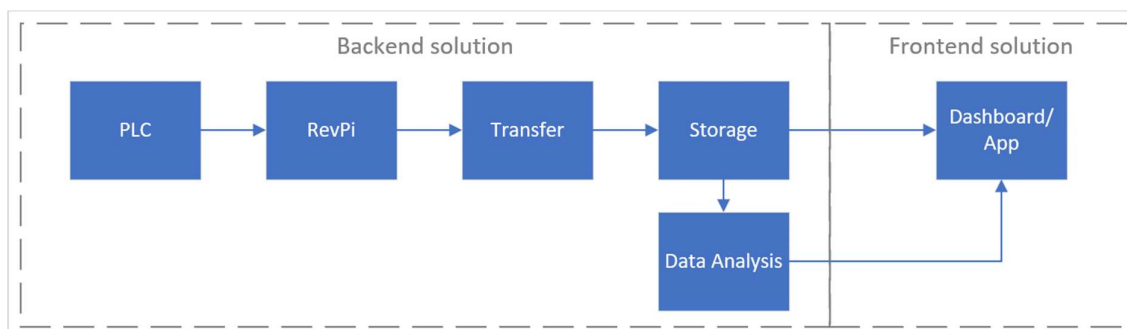


Figure 3.4: The separation between backend solution and the frontend solution

3 Methods

Starting with backend solution and the RevPi. The RevPi comes with different programming methods ready to use, e.g. Node-RED, Python or directly in C [4].

In this work, the Node-Red is selected. Node-RED is a browser-based flow editor that can be used for event-driven applications, where one does not have to understand every line of code. It gives a visual representation of the code and is accessible to a wider range of users [8]. The Node-RED program should be able to read data from the PLC, transform it to the required messaging protocol and transfer it to the cloud via a router

MQTT is a messaging protocol that can be used to transfer data from the RevPi to the cloud. Some of its advantages are that it does not use hardware identification and has low energy/lightweight needs for data transfer [9].

For cloud options, one can select to use a proprietary setup for complete control or a “standard” setup by, e.g., option for a cloud platform provider such as Azure. A proprietary setup has not been chosen due to looking at the feasibility of the project.

Azure is an established cloud platform provider with learning material available, can be scaled to need, got analytics possibility, has inbuild security features and RevPi Connect+ is an Azure certified product for easier connection/setup.

One storage option is the use of a structured query language (SQL) database with relationship connections, which Azure has support for and works when looking at feasibility. Future work will need to involve checking what option is best suited for the company and a final product

For a frontend solution, there is a need an interface to access data stored in the backend. The interface needs to be able to display data based on user login and data stored in a database. A quick connection to the database is advantageous, and the frontend software should also be able to cover security requirements.

For the frontend solution one can also look at Azure which was selected for part of the backend solution, and at Visual Studio. Azure has integrated options for creating apps/solutions that can utilize backend elements/setup, and Visual Studio can be integrated into Azure.

Microsoft Visual Studio is an option if there is a need to create from scratch and/or have a need for more complex programming needs.

Under the Microsoft umbrella, alongside with Azure and Visual Studio, we also find Power Apps where everyone can create low-code apps to share [10]. Power Apps is set up to easily connect to different data sources, e.g., Azure SQL database and Excel.

3.1.3 Azure IoT hub

The IoT Hub is included in the free trial with some restrictions, a max of 8000 messages per day and a max message size of 0.5kB, see Figure 3.5. Based on this, it can be possible to set up the RevPi to send messages five times per minute (or every 12 seconds) without getting into trouble. If data is sent continuously for 24 hours the limit of 8000 messages per day results in 333.33 messages per hour or 5.556 per minute.

Figure 3.6 shows that the RevPi connect+ is Azure qualified and ensures an easy as possible linking between RevPi and Azure.

Azure service	Description	Type	Free monthly amount	Free period
 IoT Edge	Extend cloud intelligence and analytics to IoT edge devices.	Internet of Things	Free, open-source edge runtime	Always
 IoT Hub	Connect, monitor, and manage IoT assets with a scalable platform.	Internet of Things	8,000 messages per day and .5 KB message meter size of Free edition	Always

Figure 3.5: IoT Hub limits for free trial version [11]



Figure 3.6: Azure certified device [4]

3.2 Cloud and Edge Service for IIoT Solutions

The RevPi and the Acksys router are industrial components and can be used, in a IIoT solution, for transmission of data variables selected in chapter 3.4

Advantages to cloud and edge services according to Lea [12]:

“The cloud layer provides the services of ingestion, long-term data storage, stream analytics, and patient-monitoring dashboards. It provides the interface to the healthcare providers to manage hundreds of edge systems securely through a common interface. It also is the method to quickly provide alerts to health situations, error conditions, and system failures, and provide device upgrades securely. The partitioning of cloud services versus edge services are as follows:

- *Cloud services*
 - o *Data ingestion and management for multiple edge patients and systems*
 - o *Almost unlimited storage capacity*
 - o *A controlled software deployment and updates to edge*
- *Edge services*
 - o *Low-latency and real-time reactions to events*
 - o *PAN communication to sensors*
 - o *Minimum connectivity requirements”*

According to this the vacuum unit can be seen as an edge device should it get connected to e.g., a cloud. It operates without the need for external control in order to ensure swift response time to changes in the NOMAD setup/system. A cloud connection can give the benefit of long-time storage of data and the possibility of analyzing the historic data to look for deviating trends e.g., a pump is heating up faster than the other pump or in relation to earlier data. Then this pump gets checked in a downtime period and avoid having a failure in a period with high load.

3.3 Feasibility of Having Wired and/or Wireless Communication

Looking at chapter 3.1 it seems feasible to have wired and/or wireless communication from a NOMAD vacuum unit and to a cloud.

By connecting RevPi to the internet, one can get access to its data from a remote site. The internet connection can be wired (via an ethernet switch) or wireless (also via an ethernet switch but most likely with SIM card).

By the fact that the NOMAD units are to be mobile and not placed in a fixed location, the better solution is to go the wireless route. A solution should be able to handle disruptions such as offline status (planned or unplanned), or poor cell phone coverage. Other matters to be handled are security topics and being able to handle national and international rules. For example, Norway is following the European Economic Area (EEA) regulations for production, importing and/or selling equipment [13].

With components available today it should be quite possible to send data off into the cloud/remote location using wireless communication. It is more of having suitable hardware and software to handle the communication.

Wired communication is not an ideal solution, as the NOMAD units are made to be movable and used on different sites. One cannot rely upon there being wired connections available at all possible locations, as sites will be subject to demand.

Figure 3.7 shows a possible dataflow with available hardware. The data (analog and digital signals) goes from the sensors and actuators to (and from) the PLC in the control cabinet. From the PLC a selection of data variables can go to a RevPi and, through the Acksys router, into a cloud/remote storage location. A frontend solution can give access to the data variables to those requesting it from the storage location.

Chapter 4 will check if the theory can function in practice.

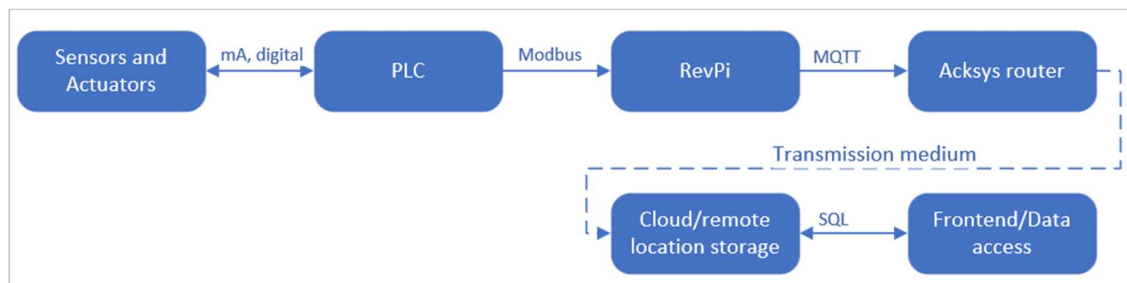


Figure 3.7: Dataflow from the field, sensors and actuators, and to the frontend solution. The dashed line is the transmission medium which can be, e.g., wireless communication

3.4 Categories/select process data to transfer

The focus will be on selecting data variables from the vacuum unit, for this stage of the project.

What variables are selected are a view of what variables can be more important to get transferred, such as the low vacuum alarm as the system is not operational without vacuum in the pipes. Similar with actual vacuum data, working hours, and temperature, this data can be

3 Methods

made into a trend/graph. Operators/monitoring models/others can look for anomalies that indicate the beginnings of a vacuum leak and/or other possible troubles.

Key data to be transferred from the vacuum unit are listed below [2]:

- Actual vacuum:
 - o Vacuum level read on the transmitter and scaled according to the service setting [2]
- Working hours for pumps
 - o Relevant to see that motors/pumps have a similar length of running time. It could also be used to look for abnormalities not otherwise detected, e.g., by using a trending graph and looking at the history of the working hours for a pump.
- Alarms
 - o Frequency drive faults
 - Check if frequency drives are OK and if related motor protection works
 - If this fault appears, the pumps will not run
 - Does not send error codes, but gives information on where to look for faults
 - o Alarm low vacuum
 - Possible leakage in the piping system
 - If there is no vacuum, the toilets/flushing will not function properly
- System OK
 - o Indicating that the PLC has power and that there are no active alarms
- Pump temperatures
 - o Temperature read at the pumps

In addition to the listed key data, other measures can be created in RevPi and forwarded to the cloud. This can be reading the time the pump starts, calculating the number of pump starts, and counting the times the solenoid valves are opened.

The selection of a transmission frequency depends on different variables, such as cell phone coverage, price of transmission and storage, how often a transmission is needed, and the size of a data package. Some reasons that the cost may fluctuate due to data transfer are:

- Global site location and network operator
- The selected plan with Azure or other platform operators
- The transmission frequency
- The size of the data package

In this work, a transmission needs at least 12 seconds as a sending interval due to the limitation in the Azure trial version. The selected sending interval is based on the information gathered in chapter 3.1.3. However, in the future, the minimum sending interval can be set pending on the user subscription and needs in the production environment. Table 1 shows a signal list of selected data variables based on the previously listed key data, and how often they can be sent.

Table 1: Signal list for the selected data variables for vacuum units and the signals generated at the RevPi

Signal name/type	Tag number	Data source	Sending interval
Actual vacuum	AV1	PLC, Modbus	In every transmission
Pump 1 Running Hours	HP1	PLC, Modbus	Every hour
Pump 2 Running Hours	HP2	PLC, Modbus	Every hour
Pump 1 Temperature	TP1	PLC, Modbus	In every transmission
Pump 2 Temperature	TP2	PLC, Modbus	In every transmission
System ok	SOK	PLC, Modbus	In every transmission
Alarm Low Vacuum	ALV	PLC, Modbus	In every transmission
Drive 1 Fault	FD1	PLC, Modbus	In every transmission
Drive 2 Fault	FD2	PLC, Modbus	In every transmission
Number of starts for pump 1	SP1	RevPi	In every transmission
Number of starts for pump 2	SP2	RevPi	In every transmission
Number of flap cleaning valve 1 opening	NC1	RevPi	In every transmission
Number of flap cleaning valve 2 opening	NC2	RevPi	In every transmission

3.5 Database design

When designing a database, one should look at what needs to be included in the database and what does not need to be included.

The database should handle the storage of received signals, signal types, timestamps, relevant unit data, and perhaps some way to connect units to owners/operators (so that owners/operators can get information about their units). In addition, it also should save the information about the firmware and software versions used on the hardware components, such as PLC, HMI, RevPi, router, and frequency drives.

Security should be taken into consideration, and requirements may differ depending on site location and content, e.g., general data protection regulation (GDPR) in Europe, for the protection of personal data [14].

3 Methods

Information that one should be more careful with is customer data. Should it be included an assessment is needed to see if a higher security is required. It may not be necessary in the existing NOMAD database, as Jets™ already has a setup for storing customer data. A second storage location can give a duplication of data, another location that needs upkeep and it can be an extra security risk.

What may be included is some way of linking NOMAD units that have a remote connection option, to customer data in Jets™ by using the unit identification numbers

Some elements are future work-related. Such has for situations where the user of NOMAD units is not a direct customer of Jets™, but has bought NOMAD units from a Jets™ customer, e.g., an original equipment manufacturer (OEM) supplier. A future work element is a need to look at whether the existing setup used by Jets™ can function for that the situation, or if an update needs to be implemented. There is a need for procedures for how to handle changes, and for analyses to see how units are being handled. The system should be able to take into consideration that units can have a change of ownership.

The database design was created using erwin Data Modeler (academic edition) [15] and the design is presented in Figure 3.8.

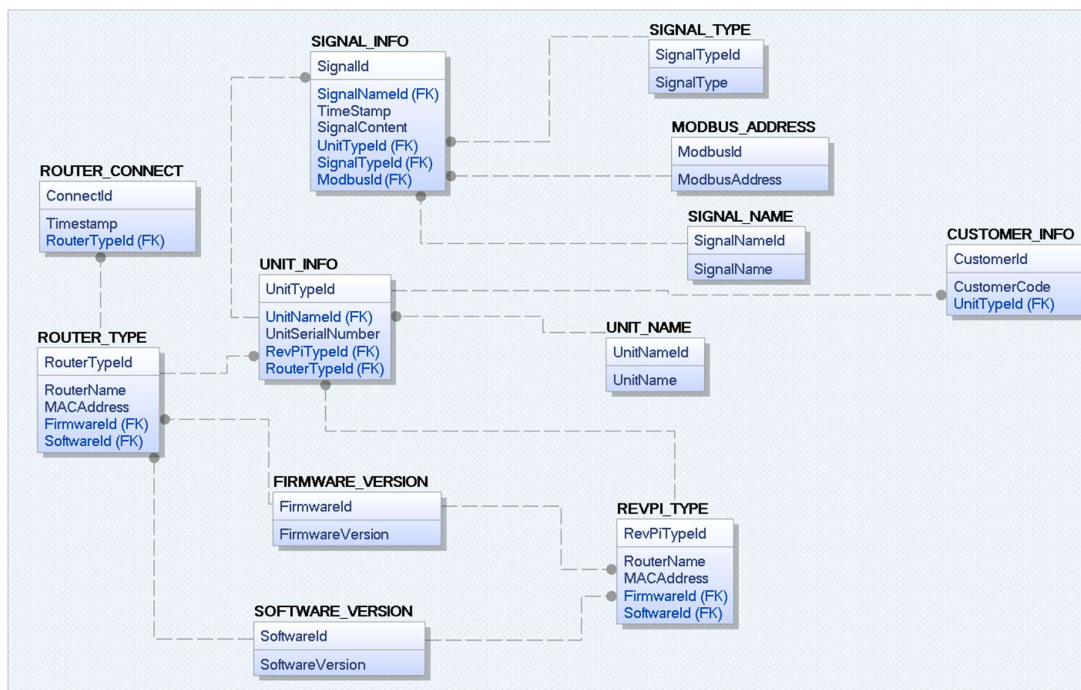


Figure 3.8: Database design showing tables and the relationships between tables

3.6 Cybersecurity Risk Analysis for IoT Systems

Possible security risk elements:

- Transmission of data
- Storage of data
- Interruptions at events

3 Methods

Any hardware and software selected for transmission of data needs to have security incorporated.

The use of Azure as a cloud platform allows continuous security updates on cloud software.

The use of cloud storage can be seen as a security buffer for Jets™, if set up properly. As NOMAD units can be used worldwide and Jets™ have little to no control over who has access to the units.

The Acksys router has the possibility of remote monitoring, either directly via a web browser or via software, WaveManager, provided by ACKSYS [6]. WaveManager can use GPS for geolocation and has inbuilt security, such as user management and password access, and HTTPS certificate management [16]. Remote monitoring can be a security risk should unauthorized people get access to the data.

A new IoT system for NOMAD units needs to take into consideration that interruptions can happen at events, and it could be internet connection issues or somebody with malicious intent.

4 System configuration and Model Development

Development

This chapter is about implementing hardware to transform and transfer data collected from the PLC, the development of the cloud-based platform, cybersecurity configuration(s), and system integration and testing.

4.1 Implement hardware for data transformation and data transfer

This chapter deals with the test setup, software and hardware configuration of the RevPi.

4.1.1 Experimental/testing setup

The first step involved was being able to make the RevPi read data from the PLC. The RevPi, PLC, and HMI needed to be connected for this to happen. The HMI was connected so that we could change a variable and see if the change registered on the RevPi. The change could be seen by connecting a computer to the RevPi, via ethernet. This also made it possible to control/check if the communication was working.

Figure 4.1 shows the data transformation and transfer test setup, including the main equipment with the ethernet and power connection. Figure 4.2 is a picture of the corresponding physical test setup.

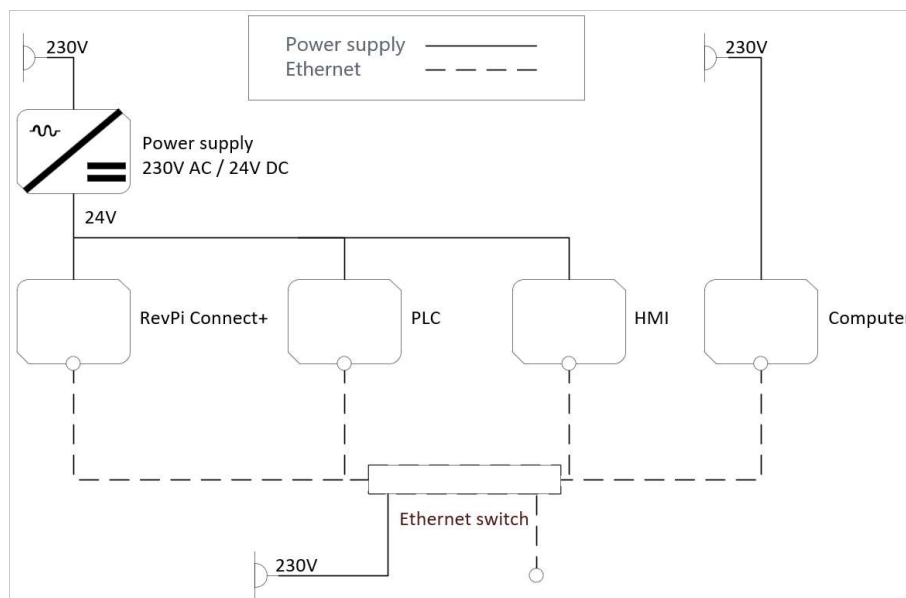


Figure 4.1: Test setup with ethernet switch, power supply, RevPi Connect, PLC, HMI, and computer (the solid lines indicate electrical wiring, the dashed lines indicate data flow)

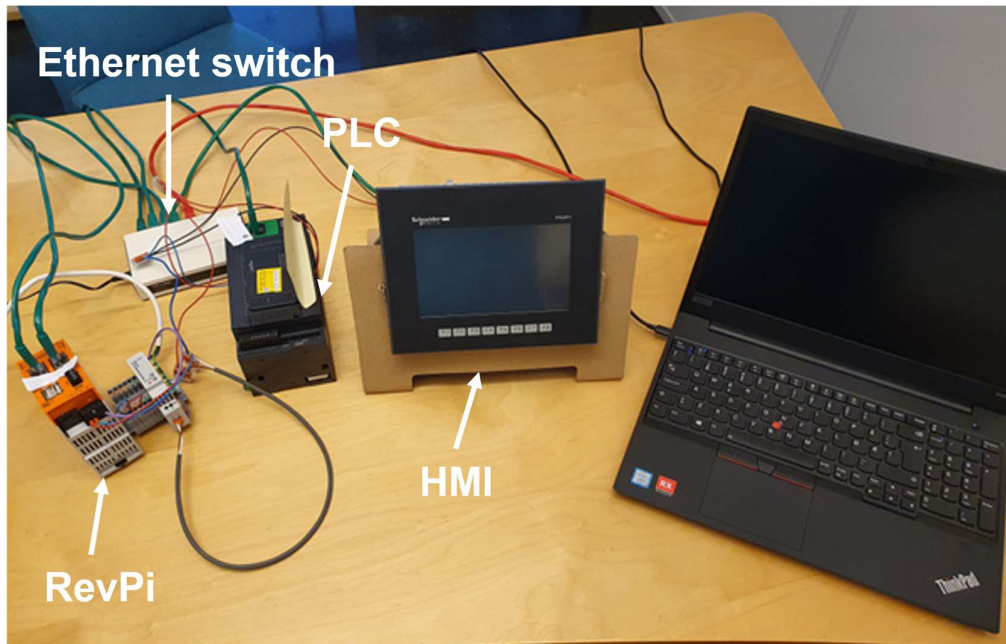


Figure 4.2: Test setup in the office

4.1.2 Hardware/software setup RevPi

When power has been connected to the RevPi, the work for setting up the software configuration, and the programming of the RevPi can start.

An internal Jets™ document was used for software configuration, together with documentation found on RevPi [4]. One of the ethernet ports on the RevPi was set to a static IP address, so that a functioning communication connection could be established with the PLC.

Object Oriented Analysis and Design (OOAD) was used for planning and documenting the programming work. Chapter 4.1.2.1 illustrates the specification, chapter 4.1.2.2 the requirement document in the form of FURPS+, chapter 4.1.2.3 shows the use case document and chapter 0 shows one use case analysis in the form of Fully Dressed Use Case Document (FDUCD).

4.1.2.1 Specification:

The aim is to get data transferred from the NOMAD vacuum unit, from PLC via RevPi to Azure cloud storage, and to make the data available in a Power BI application.

The PLC uses Modbus TCP for the communication of data.

The RevPi is to read and send data using MQTT messaging protocol and Node-RED programming. A sending frequency needs to be determined

Data is to be stored in Azure cloud using SQL database, and trending, values, alarms, etc., is to be displayed in a frontend solution.

Data variables from PLC:

- Actual vacuum level

4 System configuration and Model Development

- Working hours for pumps, 2 variables
- 3 alarm variables: two drive faults and low vacuum alarm
- System ok
- Pump temperature, 2 variables

Data variables calculated in RevPi:

- Number of pump starts, 2 variables
 - o Pump running signals to be read from the PLC every 10 seconds
 - Frequency drives have 5 seconds ramp-up time and 5 seconds ramp downtime
- Number of solenoid valves opening, 2 variables
 - o Solenoid valve opening signals need to be read from the PLC

In total 13 variables to transfer and store in the cloud.

4.1.2.2 Requirements

Using FURPS+ setup to make a set of requirements that is testable.

F: Functional

To read data from PLC, the RevPi needs to create new variables and calculate, send the data via the internet, store the data in an SQL database in Azure and display the data in the frontend solution.

Read data:

- RevPi needs to be able to read at least 9 variables from the PLC

New variables and calculation:

- The RevPi needs to create 4 new variables
- The new variables are based on data from the PLC
 - o Pump start: count the number of positive flank (changes from 0/negative to 1/positive) of pump running signal
 - o Valve opening: count the number of positive flanks of valve opening

Send data:

- The data needs to be sent from the RevPi to Azure Cloud
- Using Acksys Airbox router

Store data:

- Data is to be stored in Azure cloud SQL database

Display:

- See data in the frontend solution
- Create trends
- 13 variables

U: Usability

- Able to display the 13 variables

4 System configuration and Model Development

R: Reliability

To run when vacuum unit is in use, no information on when vacuum unit is in use

P: Performance

S: Supportability

Node-RED programming, Azure cloud, SQL database.

Use of MQTT protocol

+: design challenges/limitations

- Legal
- Cybersecurity

4.1.2.3 Use Case Document

Figure 4.3 shows the use case diagram for the datalogging system.

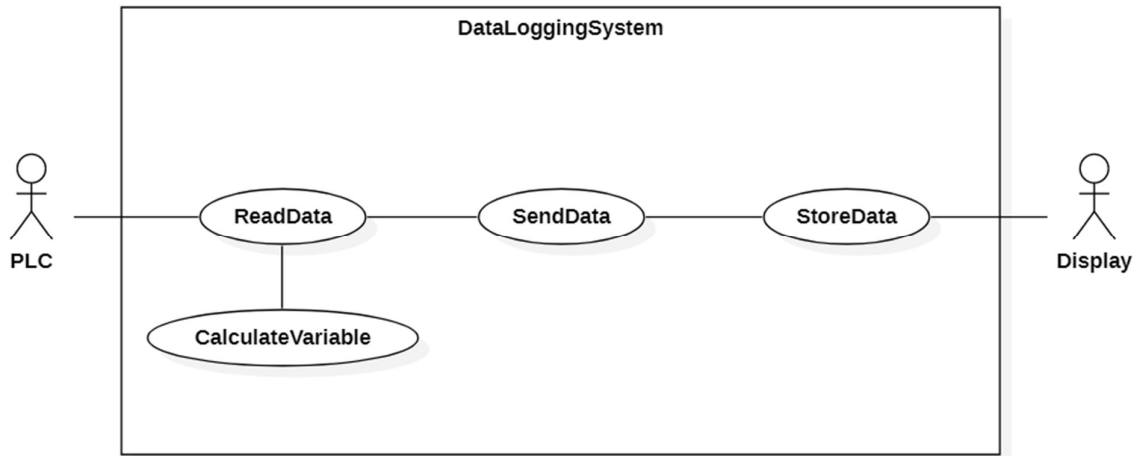


Figure 4.3: Use case diagram for datalogging system for NOMAD vacuum unit. Contains two actors, PLC and Display, and four use cases, ReadData, CalculateVariable, SendData and StoreData

4 System configuration and Model Development

4.1.2.4 Use Case Analysis

Table 4.1 is based on Figure 4.3 and the use case ReadData from this figure. This use case is selected because without the ability to read data, there is no data available to send, calculate or store.

Table 4.1: FDUCD for the use case ReadData, for reading variables from PLC

1	Use Case Name	ReadData
2	Scope	Read data from PLC with the use of RevPi Connect
3	Level	
4	Primary Actor	
5	Stakeholders	
6	Precondition	Unit is on
7	Success Gar.	
8	Main Success Scenario	<ol style="list-style-type: none"> 1. Startup of unit/RevPi 2. Connect to PLC/check if connection is ok 3. Read values on Modbus addresses <ol style="list-style-type: none"> a. Check System ok variable 4. Repeat 2-3 until shut down of unit
9	Extensions	<p>2a: no connection to PLC</p> <p>2b: send connection error information</p> <p>3a: error while reading data</p> <p>3b: error, time limit expired</p>
10	Special Req.	
11	Technology List	RevPi, Node-RED programming
12	Frequency	Every 10 seconds
13	Misc	

Figure 4.4 shows the first successful communication between the RevPi and the PLC.

4.2 Develop a cloud-based platform

This chapter is about creating an SQL database as a part of the backend solution, and the creation of a GUI for the frontend solution.

4.2.1 Database Setup and Implementation

The SQL database is created by using Azure services, as illustrated in Figure 4.9. Figure 4.10 demonstrates the database configuration, while Figure 4.11 explains the setup for creating the database server, and Figure 4.12 is about the storage setup.

Figure 4.13 presents the page *Configure database*, where we can select what type and computation and storage, we want the database to have. The recommendation is to use the vCore-based purchasing model, which works for customers who want a flexible solution with control and transparency. The database transaction unit (DTU)-based version got selected for this thesis, using the free trial version. It gives pre-configured options and simplicity[17].

Pressing *Review + Create* finishes the configuration of the database and creates the database. After the database has been created, the firewall settings need to be adjusted. The firewall settings were used to add the IP address used by the personal computer, by using “Add client IP”, see Figure 4.14. IP addresses can also be added manually. This makes it possible to login from Microsoft SQL Server Management Studio (SSMS) [18], for the implementation of the database design that can be found in chapter 3.5, using Admin login user and password. Figure 4.15 presents the look of the database in Azure portal per 19 April 2022.

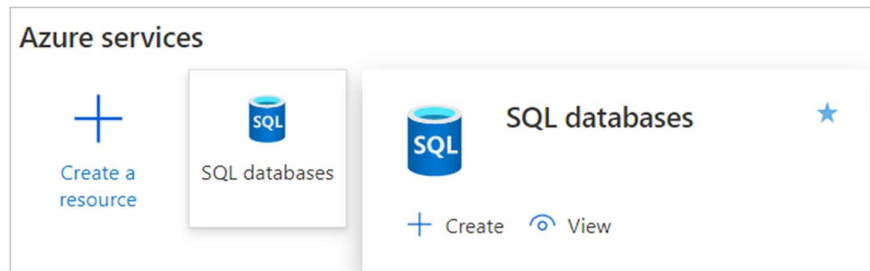


Figure 4.9: SQL database creation

Create SQL Database

Microsoft

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Free Trial ▼

Resource group * ⓘ [Redacted] ▼
[Create new](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name * Enter database name

Server * ⓘ [Redacted] (Norway East) ▼
[Create new](#)

Want to use SQL elastic pool? * ⓘ Yes No

Compute + storage * ⓘ **Standard S0**
10 DTUs, 250 GB storage

[Review + create](#) [Next : Networking >](#)

Figure 4.10: Database configuration: type of subscription (Free Trial), resource group, database name, and server with an available location (Norway East). (The specified information is highlighted in the screenshot)

SQL databases > Create SQL Database >

Create SQL Database Server

Microsoft

Server details

Enter required settings for this server, including providing a name and location. This server will be created in the same subscription and resource group as your database.

Server name * .database.windows.net

Location *

Authentication

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Azure AD authentication [Learn more &](#) using an existing Azure AD user, group, or application as Azure AD admin [Learn more &](#), or select both SQL and Azure AD authentication.

Authentication method

- Use SQL authentication
- Use only Azure Active Directory (Azure AD) authentication
- Use both SQL and Azure AD authentication

Server admin login *

Password *

Confirm password *

OK

Figure 4.11: Creating database server, including naming a server, selecting a location/region (Norway East), and creating a server admin login account

Want to use SQL elastic pool? * ⓘ Yes No

Compute + storage * ⓘ

Standard S0
10 DTUs, 250 GB storage
[Configure database](#)

Backup storage redundancy

Choose how your PITR and LTR backups are replicated. Geo restore or ability to recover from regional outage is only available when geo-redundant storage is selected.

Backup storage redundancy ⓘ Locally-redundant backup storage
 Zone-redundant backup storage
 Geo-redundant backup storage

⚠ Selected value for backup storage redundancy is Geo-redundant backup storage. Note that database backups will be geo-replicated to the paired region. [Learn more](#)

[Review + create](#) [Next : Networking >](#)

Figure 4.12: Configuration of backup storage redundancy and storage (see Figure 4.13 for the content in Configuration database)

Home > SQL databases > Create SQL Database >

Configure

[Feedback](#)


Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. [Learn more](#)

Service tier: [Compare service tiers](#)

DTUs [What is a DTU?](#)

Data max size (GB)



Cost summary

Cost per DTU (in NOK)	14.55
DTUs selected	x 10
ESTIMATED COST / MONTH	145.46 NOK

[Apply](#)

Figure 4.13: The selected setup is based on thesis needs. For the final commercial product, the setup should be optimized further.

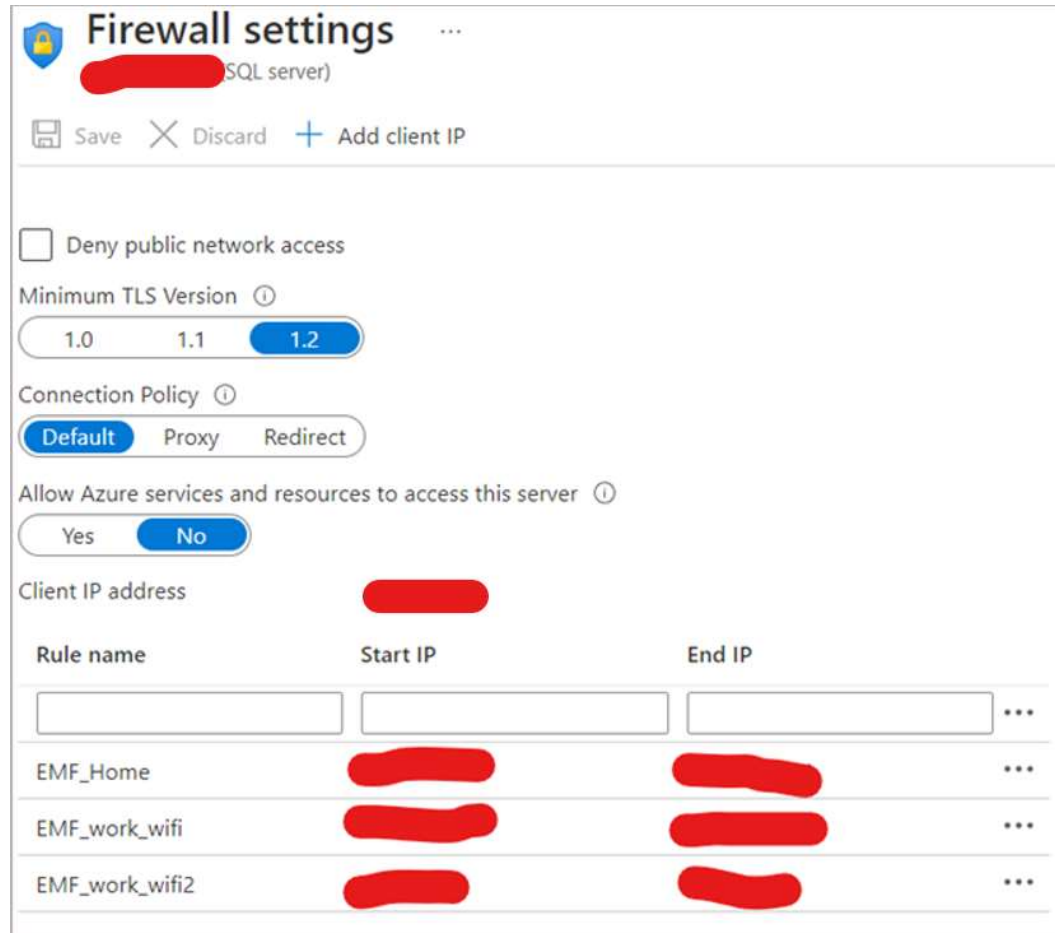


Figure 4.14: Adding IP addresses in firewall settings

4 System configuration and Model Development

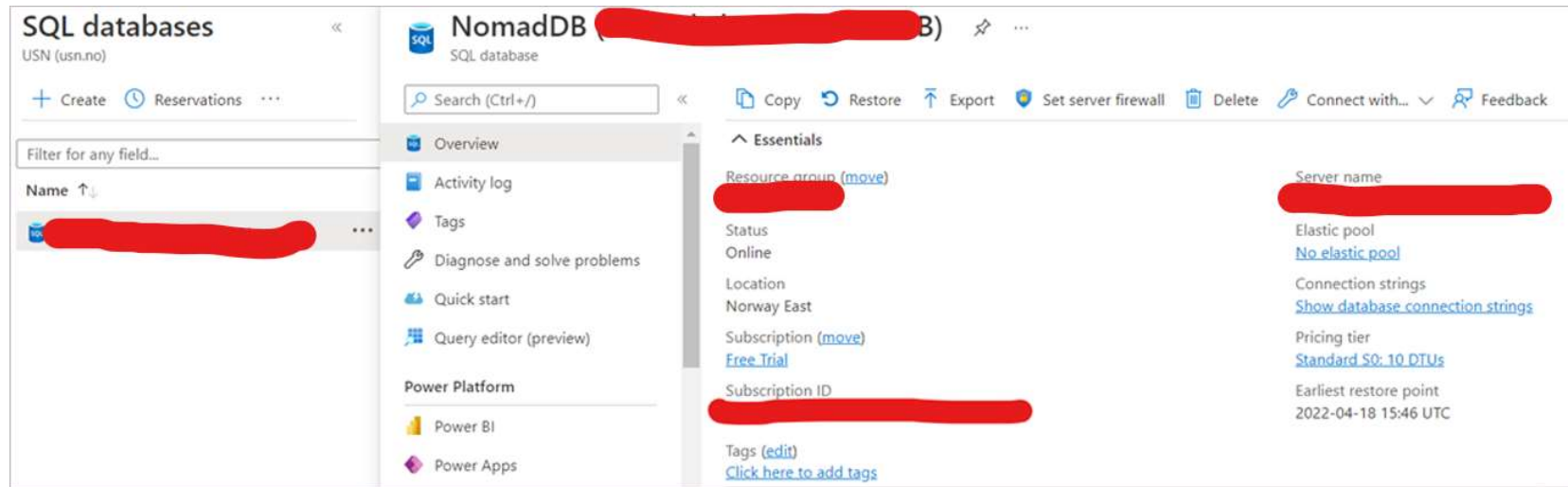


Figure 4.15: The database shown in Azure portal per April 19, 2022

4 System configuration and Model Development

Now that the database is created in the Azure portal, the database design from chapter 3.5 can be implemented. The design can be exported from Erwin Data Modeler by using Forward Engineer Schema Generation and selecting/deselecting relevant options. **Appendix C** shows the result of the export, and which can be imported into the database. This data is used to create the tables and the relationships in the SQL database by running the SQL script in SSMS.

Figure 4.16 shows the login needed to connect to the database server from SSMS. After a successful login, the exported data was imported by running the exported SQL script in a query.

Another script was created to add data into one row in each table. This was done for a more effective fill in, and because of encountering errors when trying to add data in tables with relationship to another table that was empty. **Appendix D** shows the script created, and it starts with tables that are not depending on other tables being filled in and goes on to tables depending on other tables.

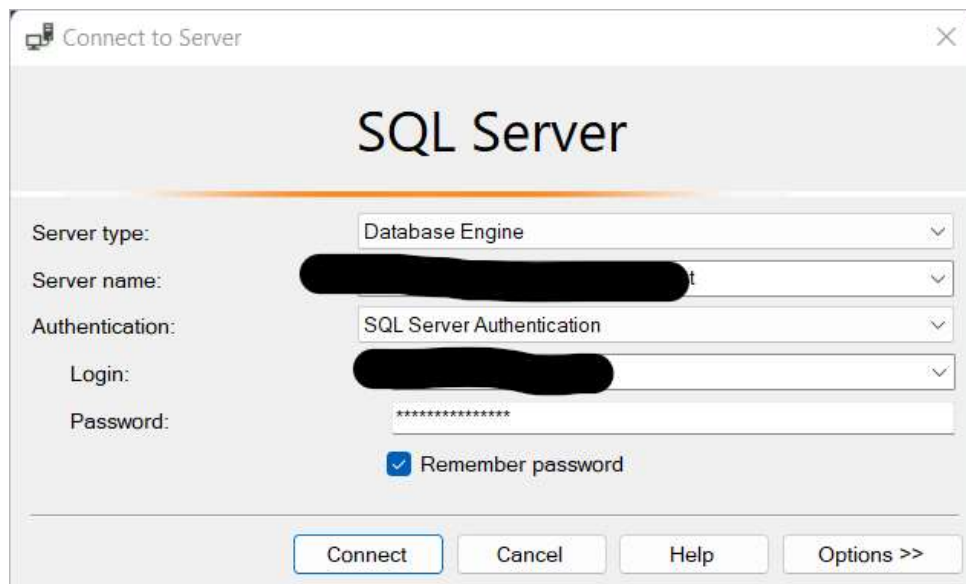


Figure 4.16: Connection to the database in the cloud from a personal computer using SSMS

Future work can be to do a market analysis to see where NOMAD units will be sold and used, what units are needed, and how many units need to be connected remotely to a database. It can also look at how much control JetsTM wants to have and manage. This is some of the elements impacting a database solution and cost.

4.2.2 Frontend for Data Visualization, using Graphical User Interface (GUI)

One solution of what a frontend solution can look like is shown in Figure 4.17, Figure 4.18, Figure 4.19, Figure 4.20, Figure 4.21, Figure 4.22, and Figure 4.24. Figure 4.23 is the background image used in the frontend solution and is from **Appendix E**. This appendix is a quick guide sample for NOMAD units [1].

The example was made in Power Apps as a canvas app for a tablet, and was connected to a SQL database and got dummy data imported from excel documents. See **Appendix F** for temperature and solenoid valve activation, and **Appendix G** for vacuum data and motor activation.

4 System configuration and Model Development

The front page, exemplified by Figure 4.17, is set up as the place to login, to get access to settings and have contact information for Jets™ without needing to login. Figure 4.18 shows the settings page where it is possible to change temperature designations and pressure designations. Figure 4.24 shows the alternative designations, Fahrenheit, and PSI.

The information page, shown in Figure 4.19, is available from multiple pages and is for easy contact to Jets™ for service requirements.

The overview page, Figure 4.20, is the place to see available units and any active alarms in relation to the available units. By selecting a unit, one can then press on one of the trending buttons so see the trend of the selected unit.

Examples of trending are shown in Figure 4.21 and Figure 4.22.

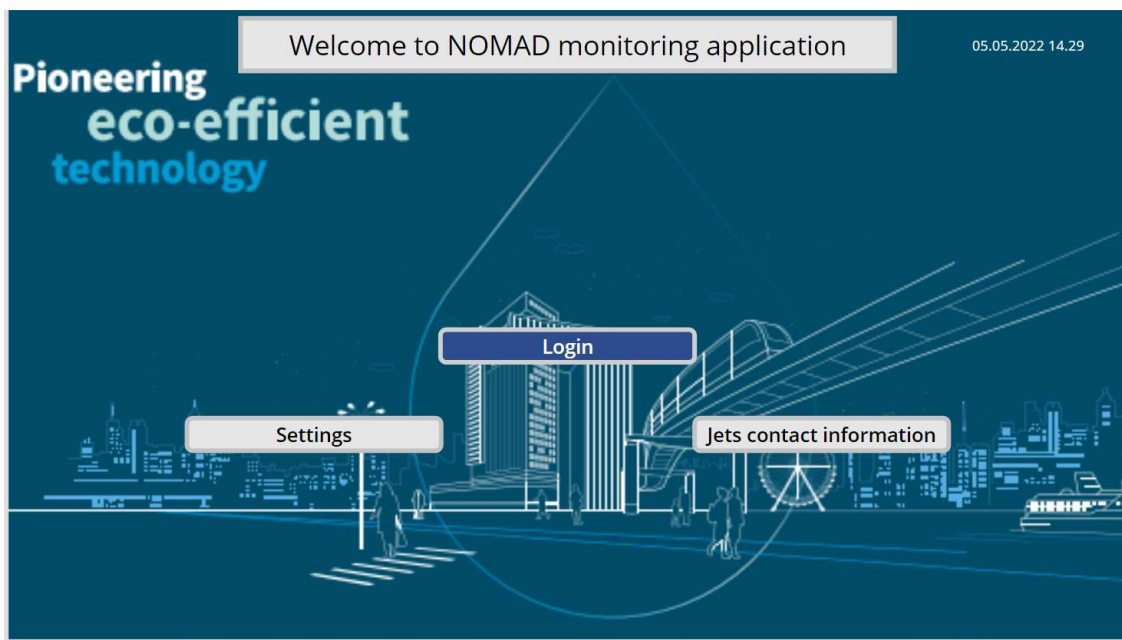


Figure 4.17: Frontpage of the tablet application

4 System configuration and Model Development

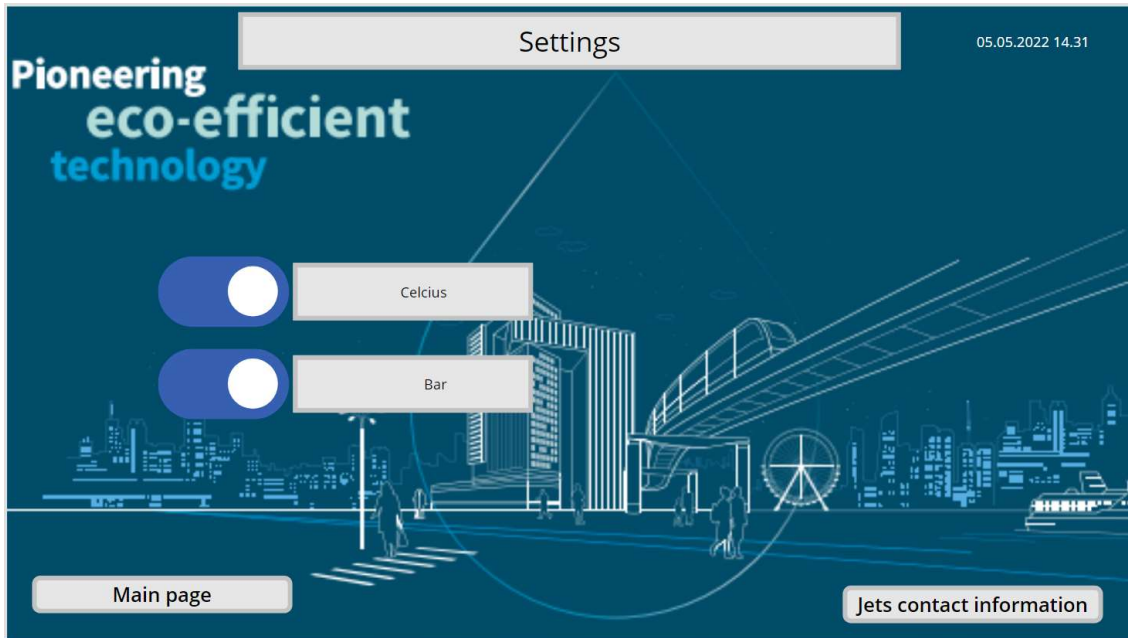


Figure 4.18: Settings page where it's possible to change between temperature and pressure

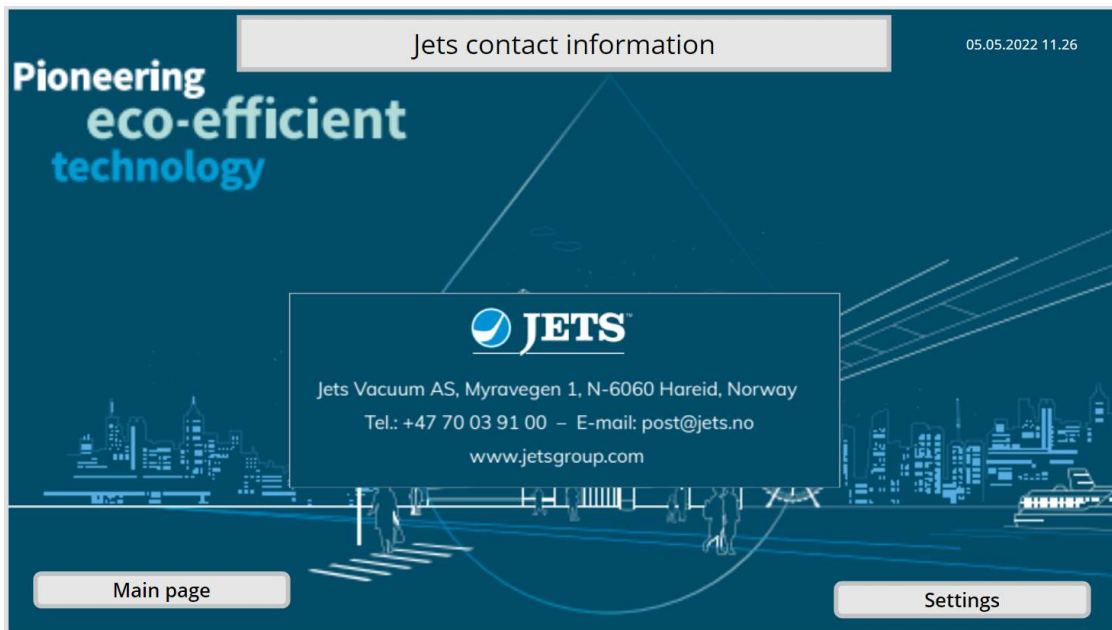


Figure 4.19: Contact information page

4 System configuration and Model Development

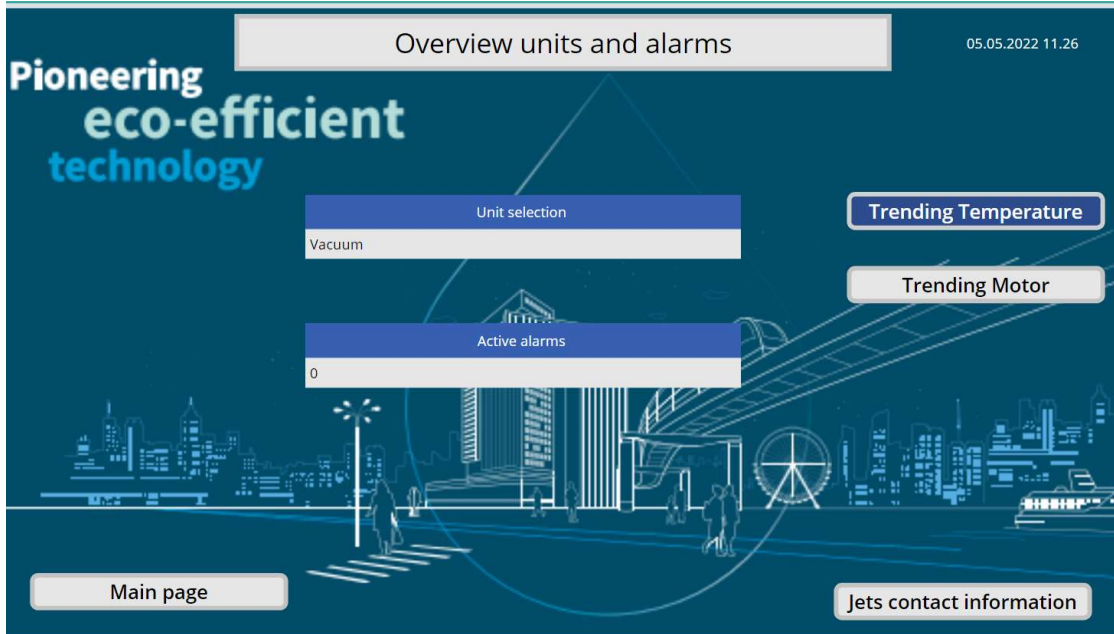


Figure 4.20: Overview page after login screen

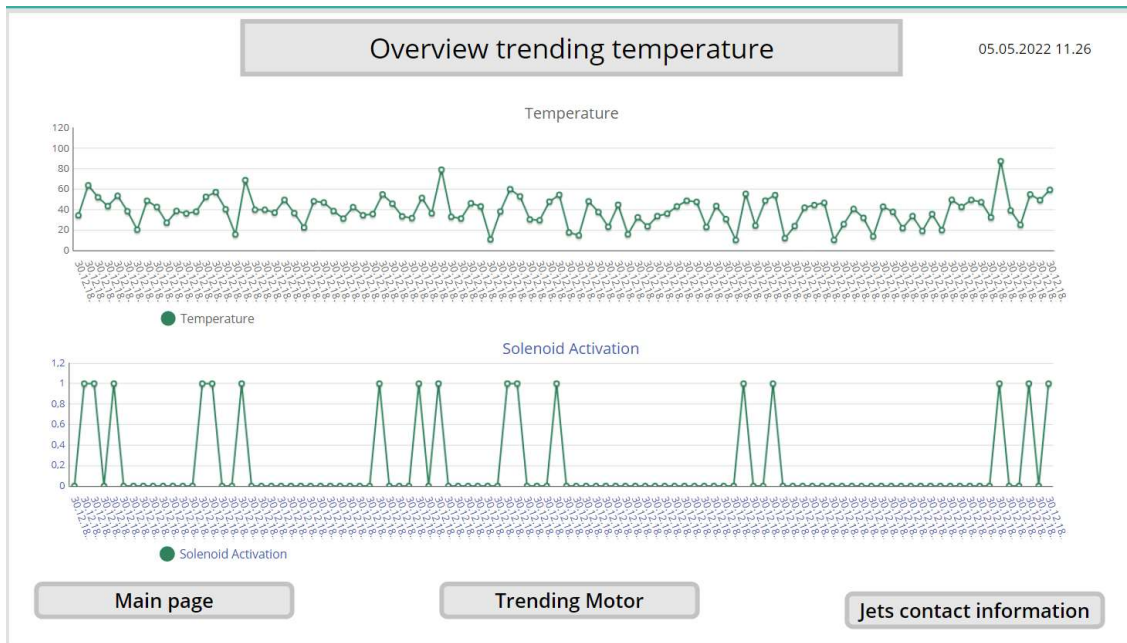


Figure 4.21: Trending page for temperature and solenoid valve

4 System configuration and Model Development

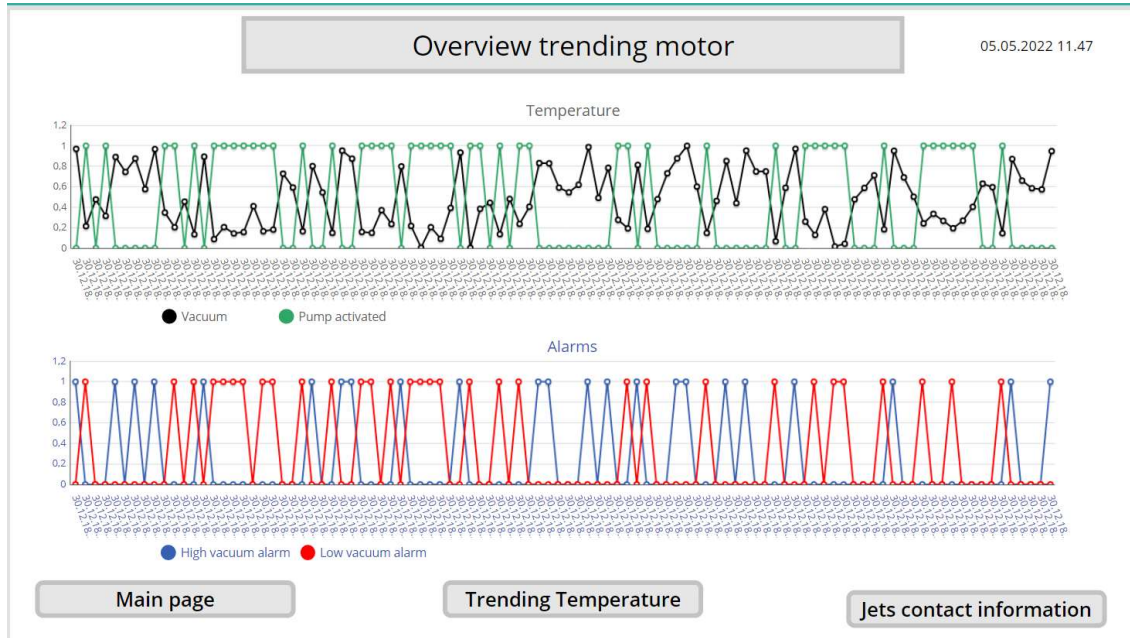


Figure 4.22: Trending page for motor and high and low vacuum alarms



Figure 4.23: Background image in application copied from **Appendix E**, found here [1], and cropped

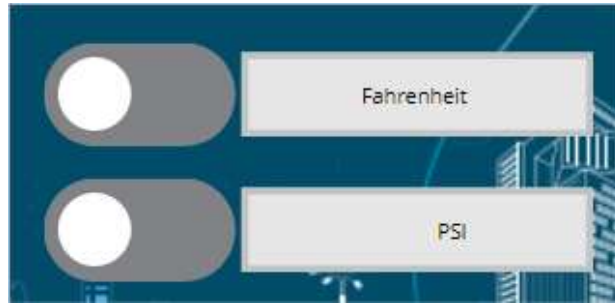


Figure 4.24: Shows the other options in Figure 4.18 for temperature and pressure

4.3 Cybersecurity for IoT systems

The existing setup uses built-in security measures available at Azure, and hardware testing has been done on a closed system.

As no transfer of data has been done between the RevPi and the SQL database, no cybersecurity work has been implemented for remote transmission.

4.4 System Integration and Testing its Efficiency and Performance

Integration has been done between the RevPi and the PLC, and between the SQL database and the frontend application. The frontend application is not yet working outside of the testing and editing environment.

5 Results and Discussion

5.1 Hardware Implementation

Got the RevPi set up and successfully communicating with the PLC.

5.2 Developed platform

The solutions selected for backend and frontend development were impacted by ease of use, time limitations and by looking at feasibility and not a finished, commercial product.

The backend solution the RevPi succeeded in communicating with the PLC, and got a database setup, and the frontend solution managed to get connection between the GUI and to the SQL database in the backend.

5.3 Cybersecurity

In this thesis integrated solutions in Azure, and a closed setup for hardware testing, were used. Cybersecurity needs to be taken into consideration in any future solution.

5.4 System Integration

Not much done, but it is important to do more in depth work and analysis in the future.

5.5 Are results as expected?

Yes, by being able to get communication between different components, and no, in the sense that I thought more would get done. This was partly due to scope, time limitations and focus points.

Elements that had a bigger, or was unexpected, scope than expected:

- Market analysis:
 - o Defining what may be required on an IoT system based on if, and how, units may change hands.
- Selection of appropriate storage solution:
 - o What is best for a possible IoT solution and a good option for Jets™
- Cybersecurity expertise:
 - o Inhouse or outsource

A solution looks technically practical, but more work is needed to make sure that cybersecurity.

Even if a solution is possible, there is a need to discover if it is an option to take it further for Jets™, on an economical and practical level.

6 Conclusion

This thesis gives a start on what work one should expect to do should the project be continued in the future.

More study needs to be done, but a remote solution looks viable. On the technical side, a solution is possible, but on the security side more research needs to be performed.

6.1 Possible Future Work

The suggestions for the future work are listed below, which should be investigated and implemented further:

- Connect the RevPi to Azure using Azure IoT hub
- Connect RevPi to database
- Try using MQTT as the communication protocol and check if it can function as intended.
- Selecting which software, hardware, and platform to use for a commercial product
- Use of data analysis
- Market analyzations

References

- [1] “Jets NOMAD - the sanitation infrastructure for all types of events - Jets™,” *Jets™ Group*. <https://jetsgroup.com/jets-nomad> (accessed Feb. 01, 2022).
- [2] “Jets.DS.VUL201-NOM.Vacuum Unit Jets L duo NOMAD.Level 3-Post-Sales Data Sheet.” *Jets™ Group*, Jan. 24, 2022.
- [3] “The most compact and reliable vacuum generator available. Jets Vacuumator pump - Jets™,” *Jets™ Group*. <https://jetsgroup.com/vacuumator> (accessed Feb. 17, 2022).
- [4] “RevPi Connect base module - Industrial Raspberry Pi,” *Revolution Pi*, May 24, 2018. <https://revolutionpi.com/revpi-connect/> (accessed Jan. 28, 2022).
- [5] “Buy IoT Devices & IoT Hardware | Azure Certified Device Catalog,” *Azure Certified Device catalog*. <https://devicecatalog.azure.com/devices/81670f07-48df-4ca0-9802-3e99a0504753> (accessed Apr. 20, 2022).
- [6] “Acksys - AirBox LTE,” *Acksys*. <https://www.acksys.fr/en/product/55-airbox-lte/> (accessed Jan. 26, 2022).
- [7] L. Simmons, “The Difference Between Front-End vs. Back-End | ComputerScience.org,” *ComputerScience.org*, Jan. 07, 2022. <https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/> (accessed May 15, 2022).
- [8] OpenJS Foundation & Contributors, “Node-RED,” *Node-RED*. <https://nodered.org/> (accessed Apr. 28, 2022).
- [9] “MQTT - The Standard for IoT Messaging,” *MQTT*. <https://mqtt.org/> (accessed Apr. 21, 2022).
- [10] “Business Apps | Microsoft Power Apps,” *Microsoft | Power Apps*. <https://powerapps.microsoft.com/en-us/> (accessed May 02, 2022).
- [11] “Create Your Azure Free Account Today | Microsoft Azure,” *Azure*. <https://azure.microsoft.com/en-us/free/> (accessed Apr. 20, 2022).
- [12] P. Lea, *IoT and Edge Computing for Architects*, 2nd ed. Packt Publishing, 2020. Accessed: Jan. 18, 2022. [Online]. Available: <https://learning.oreilly.com/library/view/iot-and-edge/9781839214806/>
- [13] “Produksjon, import og salg av utstyr,” *Nkom*. <https://www.nkom.no/frekvenser-og-elektronisk-utstyr/import/produksjon-import-og-salg-av-utstyr> (accessed May 16, 2022).
- [14] “General Data Protection Regulation (GDPR) – Official Legal Text,” *General Data Protection Regulation (GDPR)*. <https://gdpr-info.eu/> (accessed May 13, 2022).
- [15] “erwin Data Modeler | Industry-Leading Data Modeling Tool | erwin, Inc.,” *erwin by Quest*. <https://www.erwin.com/products/erwin-data-modeler/> (accessed May 13, 2022).
- [16] “Acksys - WaveManager : Network Administration tool for WaveOS-based products,” *Acksys*. <https://www.acksys.fr/en/products/software/wavemanager-waveos-products/> (accessed May 06, 2022).

References

- [17] “Purchasing models - Azure SQL Database,” *Microsoft | Docs*. <https://docs.microsoft.com/en-us/azure/azure-sql/database/purchasing-models> (accessed Apr. 19, 2022).
- [18] “SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS),” *Microsoft | Docs*. <https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms> (accessed Apr. 19, 2022).

Appendices

Appendix A – Signed Project Description

Appendix B – NOMAD Formatted Flow

Appendix C – SQL Script NOMAD

Appendix D – Dummy Starting Data

Appendix E – Jets Quick Guide NOMAD Short Version

Appendix F – Dummy Data NOMAD Temperature

Appendix G – Dummy Data NOMAD Vacuum

FMH606 Master's Thesis

Title: IoT based data logger system for NOMAD sewage units with focus on efficiency and performance

USN supervisor: Ru Yan, Saba Mylvaganam

External partner: Tor Rønnestad / JETS VACUUM AS

Task background:

Jets Vacuum, Jets™ (<https://jetsgroup.com/about-jets>) is a global company that develops, manufactures and sells vacuum toilets, vacuum systems and treatment plants. It provides eco-efficient vacuum and waste treatment technology to help companies worldwide minimize their environmental footprint and increase their profitability.

The NOMAD units are meant for festivals and events. They are new units that are soon to be released. Vacuum units, one or more transfer units, and possible one pressure relief unit are connected for sewage transfer. A separate unit handles the freshwater supply to the sewage system.

These NOMAD units presently have hardware with process data but without any real-time transmission of data. Instead, the data are now acquired manually.

Hence there is a need for automatic transfer and storage of these process data.

Internet of things (IoT) is an intelligent technology that connects sensors and devices from different locations enabling automatic transfer of real-time data. Therefore, the implementation of IoT on these NOMAD units can improve product efficiency and enable the implementation of predictive maintenance of these units. With IoT and automatic real-time transfer of data to process control centers, alarms can be picked up faster, and acquired process data can be used to schedule maintenance and repair.

Task description:

- Give an overview of the existing data collection and communication system, including hardware and software, in the NOMAD units.
- Investigate the feasibility of having wired or wireless external/remote communication
- Categories process data to be transferred and implement hardware for data transformation and data transfer, including IoT solutions.
- System integration and testing its efficiency and b performance
- Develop an "on-premises" and/or cloud-based platform including:
 - Backend for data transfer, storage, and necessary processing
 - Frontend for data visualization using a graphical user interface (GUI).
- Discuss / Implement cybersecurity risk analysis for IoT systems
- Discuss possible necessary cybersecurity measures and implement them in wired/ wireless communication system to be developed.

- Software documentation with the necessary hardware details
- Submit the thesis following the guidelines and template provided by USN

Student category: IIA

The task is suitable for online students (not present at the campus): Reserved for Edith Mari Flaten.

Practical arrangements:

Hardware and software will be available from Jets Vacuum AS with necessary support.


Supervision:

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc.).

Signatures:

Supervisor (date and signature): 28/01/2022 

Student (write clearly in all capitalized letters): EDITH MARI FLATEN

Student (date and signature): 28/01/22 

Appendix B

```
1  [
2    {
3      "id": "70ca3376.d22b5c",
4      "type": "tab",
5      "label": "Flow NOMAD Modbus Read",
6      "disabled": false,
7      "info": ""
8    },
9    {
10     "id": "d947b46e.fal418",
11     "type": "ui_text",
12     "z": "70ca3376.d22b5c",
13     "group": "91fe9b8a.7f27e8",
14     "order": 0,
15     "width": 0,
16     "height": 0,
17     "name": "",
18     "label": "modbusgetter: data",
19     "format": "{{msg.payload}}",
20     "layout": "row-left",
21     "className": "",
22     "x": 770,
23     "y": 180,
24     "wires": []
25   },
26   {
27     "id": "b2b9e8ed.fd6088",
28     "type": "ui_text",
29     "z": "70ca3376.d22b5c",
30     "group": "91fe9b8a.7f27e8",
31     "order": 1,
32     "width": 0,
33     "height": 0,
34     "name": "",
35     "label": "Time and date",
36     "format": "{{msg.payload}}",
37     "layout": "row-left",
38     "className": "",
39     "x": 760,
40     "y": 240,
41     "wires": []
42   },
43   {
44     "id": "3904a0a6.bc8f4",
45     "type": "inject",
46     "z": "70ca3376.d22b5c",
47     "name": "",
48     "props": [
49       {
50         "p": "payload"
51       }
52     ],
53     "repeat": "1",
54     "crontab": "",
55     "once": true,
56     "onceDelay": 0.1,
57     "topic": "",
58     "payload": "",
59     "payloadType": "date",
60     "x": 170,
61     "y": 220,
62     "wires": [
63       [
64         "15c7a7ca.2cc3c8",
65         "38883b67.31e5b4"
66       ]
67     ]
68   },
69   {
```

```

70     "id": "38883b67.31e5b4",
71     "type": "modbus-getter",
72     "z": "70ca3376.d22b5c",
73     "name": "PLC_modbus_getter",
74     "showStatusActivities": false,
75     "showErrors": true,
76     "logIOActivities": false,
77     "unitid": "",
78     "dataType": "HoldingRegister",
79     "adr": "***",
80     "quantity": "15",
81     "server": "76a416fb.bd8888",
82     "useIOFile": false,
83     "ioFile": "",
84     "useIOForPayload": false,
85     "emptyMsgOnFail": false,
86     "keepMsgProperties": false,
87     "x": 500,
88     "y": 160,
89     "wires": [
90         [],
91         [
92             "ede691cc.771e4",
93             "d947b46e.fal418"
94         ]
95     ]
96 },
97 {
98     "id": "c9eea190.fa869",
99     "type": "modbus-read",
100    "z": "70ca3376.d22b5c",
101    "name": "Modbus read address 19",
102    "topic": "",
103    "showStatusActivities": false,
104    "logIOActivities": false,
105    "showErrors": true,
106    "unitid": "",
107    "dataType": "HoldingRegister",
108    "adr": "***",
109    "quantity": "1",
110    "rate": "1",
111    "rateUnit": "s",
112    "delayOnStart": false,
113    "startDelayTime": "",
114    "server": "76a416fb.bd8888",
115    "useIOFile": false,
116    "ioFile": "",
117    "useIOForPayload": false,
118    "emptyMsgOnFail": false,
119    "x": 210,
120    "y": 340,
121    "wires": [
122        [],
123        [
124            "1c7808ba.558b77",
125            "afd0391.412cdc8"
126        ]
127    ]
128 },
129 {
130     "id": "15c7a7ca.2cc3c8",
131     "type": "function",
132     "z": "70ca3376.d22b5c",
133     "name": "",
134     "func": "// Copied from https://nodered.org/docs/tutorials/first-flow on April 6
2022\n// Create a Date object from the payload\nvar date = new
Date(msg.payload);\n// Change the payload to be a formatted Date
string\nmsg.payload = date.toString();\n// Return the message so it can be sent
on\nreturn msg;",

```



```
135     "outputs": 1,
136     "noerr": 0,
137     "initialize": "",
138     "finalize": "",
139     "libs": [],
140     "x": 460,
141     "y": 240,
142     "wires": [
143       [
144         "b2b9e8ed.fd6088"
145       ]
146     ]
147   },
148   {
149     "id": "ede691cc.771e4",
150     "type": "debug",
151     "z": "70ca3376.d22b5c",
152     "name": "",
153     "active": false,
154     "tosidebar": true,
155     "console": false,
156     "tostatus": false,
157     "complete": "false",
158     "statusVal": "",
159     "statusType": "auto",
160     "x": 750,
161     "y": 120,
162     "wires": []
163   },
164   {
165     "id": "1c7808ba.558b77",
166     "type": "ui_text",
167     "z": "70ca3376.d22b5c",
168     "group": "91fe9b8a.7f27e8",
169     "order": 0,
170     "width": "0",
171     "height": "0",
172     "name": "",
173     "label": "modbus-read: set address",
174     "format": "{{msg.payload}}",
175     "layout": "row-left",
176     "className": "2",
177     "x": 510,
178     "y": 320,
179     "wires": []
180   },
181   {
182     "id": "afd0391.412cdc8",
183     "type": "debug",
184     "z": "70ca3376.d22b5c",
185     "name": "",
186     "active": false,
187     "tosidebar": true,
188     "console": false,
189     "tostatus": false,
190     "complete": "false",
191     "statusVal": "",
192     "statusType": "auto",
193     "x": 470,
194     "y": 380,
195     "wires": []
196   },
197   {
198     "id": "85f85b13.9e6f88",
199     "type": "modbus-read",
200     "z": "70ca3376.d22b5c",
201     "name": "",
202     "topic": "",
203     "showStatusActivities": false,
```

```

204     "logIOActivities": false,
205     "showErrors": true,
206     "unitid": "",
207     "dataType": "HoldingRegister",
208     "adr": "***",
209     "quantity": "15",
210     "rate": "1",
211     "rateUnit": "s",
212     "delayOnStart": false,
213     "startDelayTime": "",
214     "server": "76a416fb.bd8888",
215     "useIOFile": false,
216     "ioFile": "",
217     "useIOForPayload": false,
218     "emptyMsgOnFail": false,
219     "x": 170,
220     "y": 480,
221     "wires": [
222         [],
223         [
224             "94908d68.6abb4",
225             "b31e035.a74bc"
226         ]
227     ]
228 },
229 {
230     "id": "94908d68.6abb4",
231     "type": "ui_text",
232     "z": "70ca3376.d22b5c",
233     "group": "91fe9b8a.7f27e8",
234     "order": 0,
235     "width": "0",
236     "height": "0",
237     "name": "",
238     "label": "modbus-read: all data",
239     "format": "{{msg.payload}}",
240     "layout": "row-left",
241     "className": "2",
242     "x": 500,
243     "y": 460,
244     "wires": []
245 },
246 {
247     "id": "b31e035.a74bc",
248     "type": "debug",
249     "z": "70ca3376.d22b5c",
250     "name": "",
251     "active": false,
252     "tosidebar": true,
253     "console": false,
254     "tostatus": false,
255     "complete": "false",
256     "statusVal": "",
257     "statusType": "auto",
258     "x": 470,
259     "y": 520,
260     "wires": []
261 },
262 {
263     "id": "91fe9b8a.7f27e8",
264     "type": "ui_group",
265     "name": "RevPi NOMAD",
266     "tab": "ba074780.869718",
267     "order": 1,
268     "disp": true,
269     "width": "14",
270     "collapse": false,
271     "className": ""
272 },

```

```
273 {
274     "id": "76a416fb.bd8888",
275     "type": "modbus-client",
276     "name": "PLC",
277     "clienttype": "tcp",
278     "bufferCommands": true,
279     "stateLogEnabled": false,
280     "queueLogEnabled": false,
281     "tcpHost": "****.***.***.***",
282     "tcpPort": "****",
283     "tcpType": "DEFAULT",
284     "serialPort": "/dev/*****",
285     "serialType": "RTU-BUFFERD",
286     "serialBaudrate": "9600",
287     "serialDatabits": "8",
288     "serialStopbits": "1",
289     "serialParity": "none",
290     "serialConnectionDelay": "100",
291     "serialAsciiResponseStartDelimiter": "0x3A",
292     "unit_id": 1,
293     "commandDelay": 1,
294     "clientTimeout": 1000,
295     "reconnectOnTimeout": true,
296     "reconnectTimeout": 2000,
297     "parallelUnitIdsAllowed": true
298 },
299 {
300     "id": "ba074780.869718",
301     "type": "ui_tab",
302     "name": "Time",
303     "icon": "dashboard",
304     "disabled": false,
305     "hidden": false
306 }
307 ]
```

Appendix C

```
1
2 CREATE TABLE [CUSTOMER_INFO]
3 (
4     [CustomerId]          int NOT NULL ,
5     [CustomerCode]       integer NULL ,
6     [UnitTypeId]         int NOT NULL
7 )
8 go
9
10 CREATE TABLE [FIRMWARE_VERSION]
11 (
12     [FirmwareId]         int NOT NULL ,
13     [FirmwareVersion]    char(18) NULL
14 )
15 go
16
17 CREATE TABLE [MODBUS_ADDRESS]
18 (
19     [ModbusId]           int NOT NULL ,
20     [ModbusAddress]      char(18) NULL
21 )
22 go
23
24 CREATE TABLE [REVPI_TYPE]
25 (
26     [RevPiTypeId]        int NOT NULL ,
27     [RouterName]         char(18) NULL ,
28     [MACAddress]         char(18) NULL ,
29     [FirmwareId]         int NOT NULL ,
30     [SoftwareId]         int NOT NULL
31 )
32 go
33
34 CREATE TABLE [ROUTER_CONNECT]
35 (
36     [ConnectId]          int NOT NULL ,
37     [TimeSignature]      timestamp NULL ,
38     [RouterTypeId]       int NOT NULL
39 )
40 go
41
42 CREATE TABLE [ROUTER_TYPE]
43 (
44     [RouterTypeId]        int NOT NULL ,
45     [RouterName]         varchar(50) NULL ,
46     [MACAddress]         varchar(50) NULL ,
47     [FirmwareId]         int NOT NULL ,
48     [SoftwareId]         int NOT NULL
49 )
50 go
51
52 CREATE TABLE [SIGNAL_INFO]
53 (
54     [SignalId]           int NOT NULL ,
55     [SignalContent]      varchar(50) NULL ,
56     [TimeSignature]      timestamp NULL ,
57     [SignalTypeId]       int NOT NULL ,
58     [ModbusId]           int NOT NULL ,
59     [UnitTypeId]         int NOT NULL ,
60     [SignalNameId]       int NOT NULL
61 )
62 go
63
64 CREATE TABLE [SIGNAL_NAME]
65 (
66     [SignalNameId]       int NOT NULL ,
67     [SignalName]         varchar(50) NULL
68 )
69 go
70
71 CREATE TABLE [SIGNAL_TYPE]
72 (
73     [SignalTypeId]       int NOT NULL ,
```

```

74     [SignalType]          varchar(50) NULL
75 )
76 go
77
78 CREATE TABLE [SOFTWARE_VERSION]
79 (
80     [SoftwareId]          int NOT NULL ,
81     [SoftwareVersion]    varchar(50) NULL
82 )
83 go
84
85 CREATE TABLE [UNIT_INFO]
86 (
87     [UnitTypeId]         int NOT NULL ,
88     [UnitSerialNumber]   varchar(50) NULL ,
89     [RouterTypeId]       int NOT NULL ,
90     [RevPiTypeId]        int NOT NULL ,
91     [UnitNameId]         int NOT NULL
92 )
93 go
94
95 CREATE TABLE [UNIT_NAME]
96 (
97     [UnitNameId]         int NOT NULL ,
98     [UnitName]           varchar(50) NULL
99 )
100 go
101
102 ALTER TABLE [CUSTOMER_INFO]
103     ADD CONSTRAINT [XPKCUSTOMER_INFO] PRIMARY KEY CLUSTERED ([CustomerId] ASC)
104 go
105
106 ALTER TABLE [FIRMWARE_VERSION]
107     ADD CONSTRAINT [XPKFIRMWARE_VERSION] PRIMARY KEY CLUSTERED ([FirmwareId] ASC)
108 go
109
110 ALTER TABLE [MODBUS_ADDRESS]
111     ADD CONSTRAINT [XPKMODBUS_ADDRESS] PRIMARY KEY CLUSTERED ([ModbusId] ASC)
112 go
113
114 ALTER TABLE [REVPI_TYPE]
115     ADD CONSTRAINT [XPKREVPI_TYPE] PRIMARY KEY CLUSTERED ([RevPiTypeId] ASC)
116 go
117
118 ALTER TABLE [ROUTER_CONNECT]
119     ADD CONSTRAINT [XPKROUTER_CONNECT] PRIMARY KEY CLUSTERED ([ConnectId] ASC)
120 go
121
122 ALTER TABLE [ROUTER_TYPE]
123     ADD CONSTRAINT [XPKROUTER_TYPE] PRIMARY KEY CLUSTERED ([RouterTypeId] ASC)
124 go
125
126 ALTER TABLE [SIGNAL_INFO]
127     ADD CONSTRAINT [XPKSIGNAL_INFO] PRIMARY KEY CLUSTERED ([SignalId] ASC)
128 go
129
130 ALTER TABLE [SIGNAL_NAME]
131     ADD CONSTRAINT [XPKSIGNAL_NAME] PRIMARY KEY CLUSTERED ([SignalNameId] ASC)
132 go
133
134 ALTER TABLE [SIGNAL_TYPE]
135     ADD CONSTRAINT [XPKSIGNAL_TYPE] PRIMARY KEY CLUSTERED ([SignalTypeId] ASC)
136 go
137
138 ALTER TABLE [SOFTWARE_VERSION]
139     ADD CONSTRAINT [XPKSOFTWARE_VERSION] PRIMARY KEY CLUSTERED ([SoftwareId] ASC)
140 go
141
142 ALTER TABLE [UNIT_INFO]
143     ADD CONSTRAINT [XPKUNIT_INFO] PRIMARY KEY CLUSTERED ([UnitTypeId] ASC)
144 go
145
146 ALTER TABLE [UNIT_NAME]

```

```

147     ADD CONSTRAINT [XPKUNIT_NAME] PRIMARY KEY CLUSTERED ([UnitNameId] ASC)
148 go
149
150
151 ALTER TABLE [CUSTOMER_INFO]
152     ADD CONSTRAINT [R_18] FOREIGN KEY ([UnitTypeId]) REFERENCES [UNIT_INFO] ([
UnitTypeId])
153     ON DELETE NO ACTION
154     ON UPDATE NO ACTION
155 go
156
157
158 ALTER TABLE [REVPI_TYPE]
159     ADD CONSTRAINT [R_5] FOREIGN KEY ([FirmwareId]) REFERENCES [FIRMWARE_VERSION] ([
FirmwareId])
160     ON DELETE NO ACTION
161     ON UPDATE NO ACTION
162 go
163
164 ALTER TABLE [REVPI_TYPE]
165     ADD CONSTRAINT [R_7] FOREIGN KEY ([SoftwareId]) REFERENCES [SOFTWARE_VERSION] ([
SoftwareId])
166     ON DELETE NO ACTION
167     ON UPDATE NO ACTION
168 go
169
170
171 ALTER TABLE [ROUTER_CONNECT]
172     ADD CONSTRAINT [R_15] FOREIGN KEY ([RouterTypeId]) REFERENCES [ROUTER_TYPE] ([
RouterTypeId])
173     ON DELETE NO ACTION
174     ON UPDATE NO ACTION
175 go
176
177
178 ALTER TABLE [ROUTER_TYPE]
179     ADD CONSTRAINT [R_11] FOREIGN KEY ([FirmwareId]) REFERENCES [FIRMWARE_VERSION] ([
FirmwareId])
180     ON DELETE NO ACTION
181     ON UPDATE NO ACTION
182 go
183
184 ALTER TABLE [ROUTER_TYPE]
185     ADD CONSTRAINT [R_12] FOREIGN KEY ([SoftwareId]) REFERENCES [SOFTWARE_VERSION] ([
SoftwareId])
186     ON DELETE NO ACTION
187     ON UPDATE NO ACTION
188 go
189
190
191 ALTER TABLE [SIGNAL_INFO]
192     ADD CONSTRAINT [R_8] FOREIGN KEY ([SignalTypeId]) REFERENCES [SIGNAL_TYPE] ([
SignalTypeId])
193     ON DELETE NO ACTION
194     ON UPDATE NO ACTION
195 go
196
197 ALTER TABLE [SIGNAL_INFO]
198     ADD CONSTRAINT [R_9] FOREIGN KEY ([ModbusId]) REFERENCES [MODBUS_ADDRESS] ([
ModbusId])
199     ON DELETE NO ACTION
200     ON UPDATE NO ACTION
201 go
202
203 ALTER TABLE [SIGNAL_INFO]
204     ADD CONSTRAINT [R_10] FOREIGN KEY ([UnitTypeId]) REFERENCES [UNIT_INFO] ([
UnitTypeId])
205     ON DELETE NO ACTION
206     ON UPDATE NO ACTION
207 go
208
209 ALTER TABLE [SIGNAL_INFO]
210     ADD CONSTRAINT [R_16] FOREIGN KEY ([SignalNameId]) REFERENCES [SIGNAL_NAME] ([

```

```
SignalNameId])
211         ON DELETE NO ACTION
212         ON UPDATE NO ACTION
213 go
214
215
216 ALTER TABLE [UNIT_INFO]
217     ADD CONSTRAINT [R_13] FOREIGN KEY ([RouterTypeId]) REFERENCES [ROUTER_TYPE] ([
RouterTypeId])
218         ON DELETE NO ACTION
219         ON UPDATE NO ACTION
220 go
221
222 ALTER TABLE [UNIT_INFO]
223     ADD CONSTRAINT [R_14] FOREIGN KEY ([RevPiTypeId]) REFERENCES [REVPI_TYPE] ([
RevPiTypeId])
224         ON DELETE NO ACTION
225         ON UPDATE NO ACTION
226 go
227
228 ALTER TABLE [UNIT_INFO]
229     ADD CONSTRAINT [R_17] FOREIGN KEY ([UnitNameId]) REFERENCES [UNIT_NAME] ([
UnitNameId])
230         ON DELETE NO ACTION
231         ON UPDATE NO ACTION
232 go
233
```

Appendix D

```
1
2  /* query for inserting dummy data into database tables*/
3  insert into SOFTWARE_VERSION (SoftwareId,SoftwareVersion)
4  values (1,'123abc456')
5
6  insert into FIRMWARE_VERSION (FirmwareId,FirmwareVersion)
7  values (1,'abc123def')
8
9  insert into REVPI_TYPE (RevPiTypeId,RouterName,MACAddress,FirmwareId,SoftwareId)
10 values (1,'RevPi','HEX123',1,1)
11
12 insert into ROUTER_TYPE (RouterTypeId,RouterName,MACAddress,FirmwareId,SoftwareId)
13 values (1,'Router','123HEX',1,1)
14
15 insert into ROUTER_CONNECT (ConnectId,RouterTypeId)
16 values (1,1)
17
18 insert into UNIT_NAME (UnitNameId,UnitName)
19 values (1,'Vacuum')
20
21 insert into UNIT_INFO (UnitTypeId,UnitNameId,UnitSerialNumber,RevPiTypeId,RouterTypeId)
22 values (1,1,'987321',1,1)
23
24 insert into SIGNAL_NAME (SignalNameId,SignalName)
25 values (1,'Solenoid activation')
26
27 insert into MODBUS_ADDRESS (ModbusId,ModbusAddress)
28 values (1,77)
29
30 insert into SIGNAL_TYPE (SignalTypeId,SignalType)
31 values (1,'Binary')
32
33 insert into SIGNAL_INFO (SignalId,SignalContent,SignalTypeId,ModbusId,UnitTypeId,
SignalNameId)
34 values (1,0,1,1,1,1)
35
36 insert into CUSTOMER_INFO (CustomerId,CustomerCode,UnitTypeId)
37 values (1,123,1)
```




JETTS™ NOMAD

The mobile sanitation infrastructure

Quick guide

Quick introduction to the JETS™ NOMAD



Vacuum Unit
80 x 120 x 130 cm



Catcher Unit
60 x 80 x 130 cm



Transfer Tank Unit
60 x 80 x 130 cm



Flow Control Unit
60 x 80 x 130 cm

The new JETS™ NOMAD handles the entire infrastructure of water and waste water throughout the event site.

From a category 5 water supply system to the discharge point, the Nomad system covers all your needs when it comes to temporary infrastructure.

This scalable system may be connected to any

type of trailer, container or pod solution utilizing CVS™ toilets.

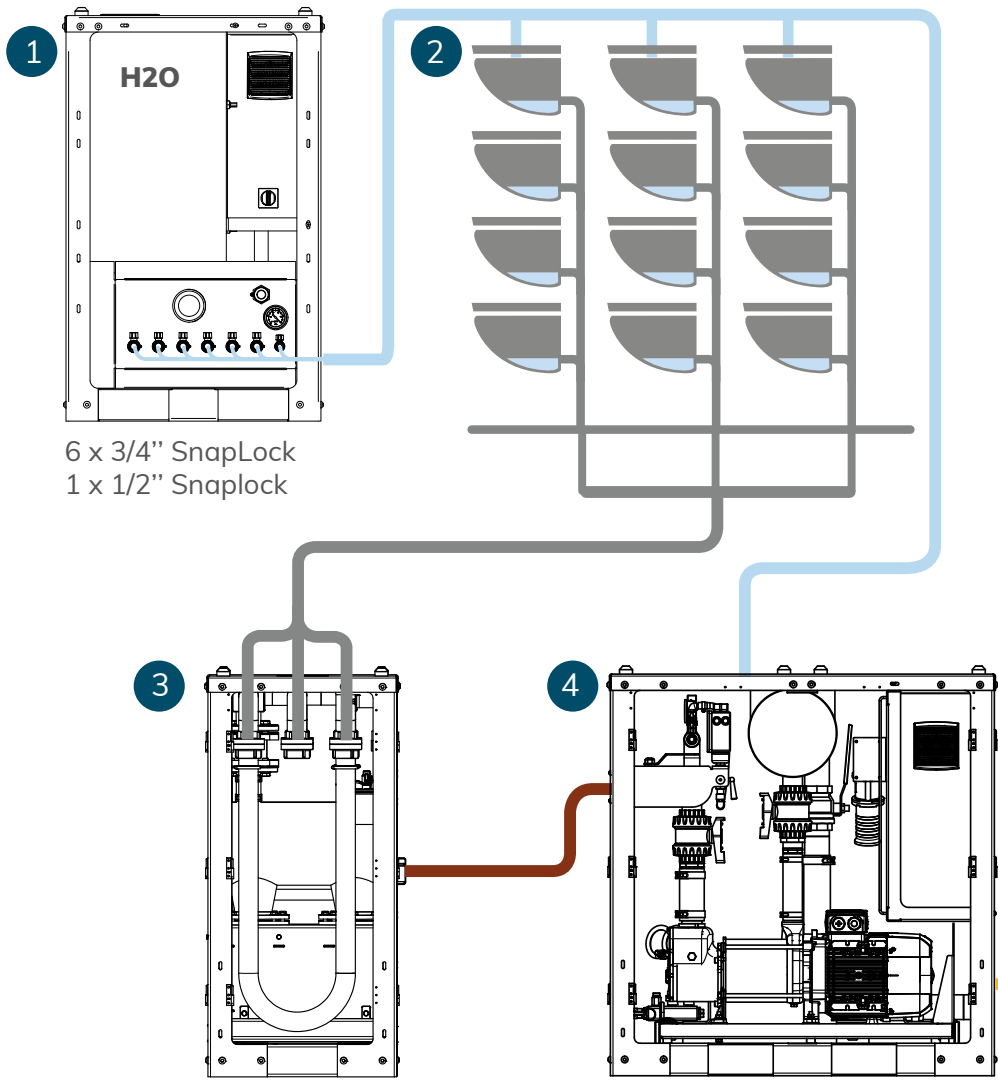
Combine modules to your specific needs at any site anywhere, with a discharge distance to mains sewer or collecting point up to 1,6 km / 1 mile!

Catcher from JETS™ removes items not supposed to be flushed down the toilets with minimal disturbance to the operation of the system.

All this in a system ensuring an even flow at 3,75 L / sec or 1 Gall / sec) by gravity to mains sewer.



Water Supply Unit
60 x 80 x 130 cm

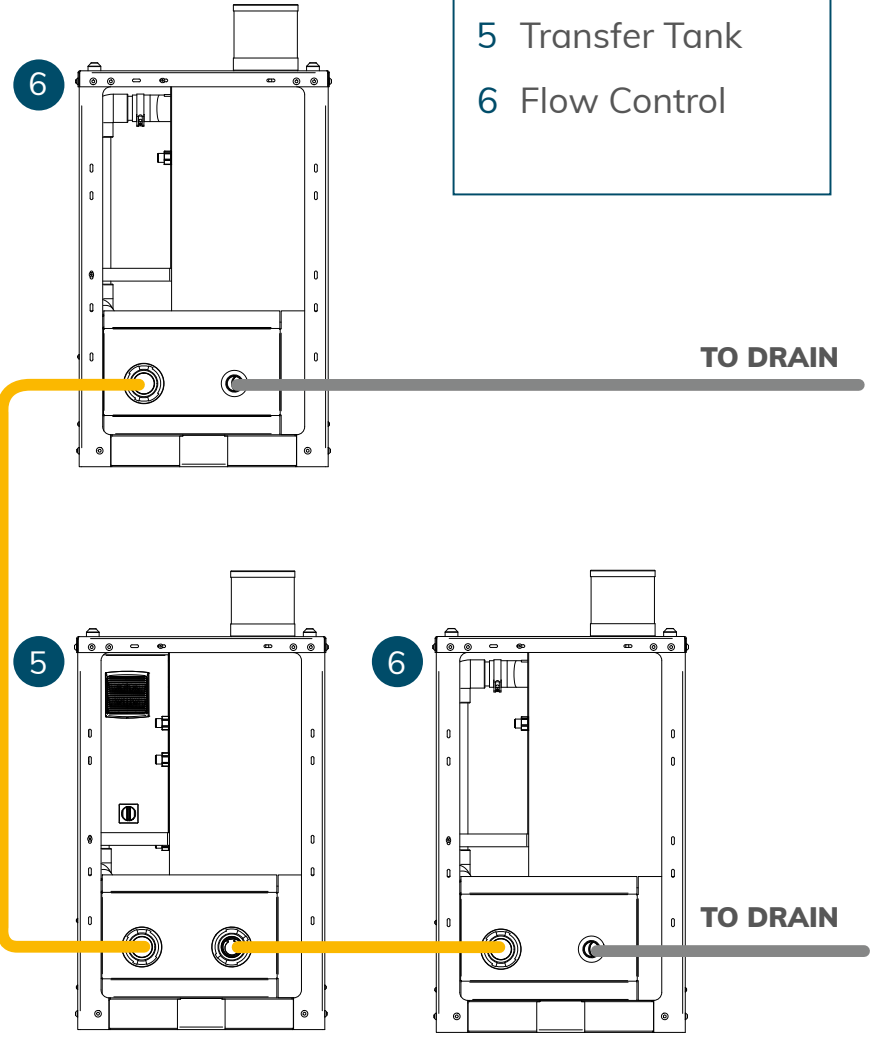
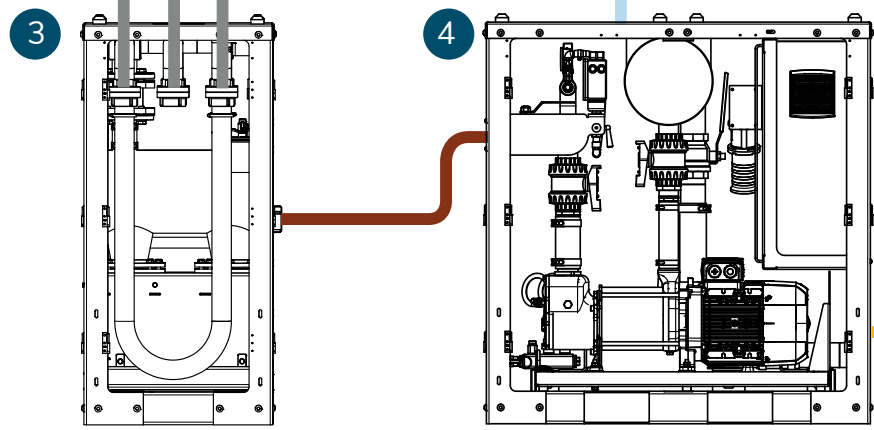


6 x 3/4" SnapLock
1 x 1/2" Snaplock

JETS™ NOMAD

System overview

- 1 Water Supply
- 2 Toilets
- 3 Catcher
- 4 Vacuum Unit
- 5 Transfer Tank
- 6 Flow Control



Up to 100 m

Up to 1,6 km / 1 mile

- WATER PIPE 50 mm 3/4"
- VACUUM HOSE (2 x 1,95 meters included)
- PRESSURE HOSE 2 1/2" (2 x 10 meters included)
- PIPE TO DRAIN 50 mm

Pioneering
eco-efficient
technology



Jets Vacuum AS, Myravegen 1, N-6060 Hareid, Norway

Tel.: +47 70 03 91 00 – E-mail: post@jets.no

www.jetsgroup.com

Appendix F

Timestamp to use	Temperature	Solenoid Activation
12:24 PM	34.32	0
12:24 PM	63.79	1
12:24 PM	52.05	1
12:24 PM	43.36	0
12:24 PM	53.55	1
12:24 PM	38.30	0
12:24 PM	20.24	0
12:24 PM	48.64	0
12:24 PM	42.53	0
12:24 PM	27.13	0
12:24 PM	38.57	0
12:24 PM	36.08	0
12:25 PM	38.03	0
12:25 PM	52.45	1
12:25 PM	57.08	1
12:25 PM	40.21	0
12:25 PM	15.65	0
12:25 PM	68.81	1
12:25 PM	39.78	0
12:25 PM	39.82	0
12:25 PM	36.94	0
12:25 PM	49.46	0
12:25 PM	36.48	0
12:25 PM	22.35	0
12:26 PM	48.16	0
12:26 PM	46.93	0
12:26 PM	38.55	0
12:26 PM	31.15	0
12:26 PM	42.34	0
12:26 PM	34.45	0
12:26 PM	35.62	0
12:26 PM	54.91	1
12:26 PM	45.76	0
12:26 PM	33.29	0
12:26 PM	31.64	0
12:26 PM	51.44	1
12:27 PM	36.30	0
12:27 PM	78.98	1
12:27 PM	32.89	0
12:27 PM	31.20	0
12:27 PM	46.19	0
12:27 PM	43.07	0
12:27 PM	10.72	0
12:27 PM	38.14	0
12:27 PM	59.95	1
12:27 PM	52.86	1
12:27 PM	30.31	0
12:27 PM	29.54	0
12:28 PM	47.68	0
12:28 PM	54.45	1
12:28 PM	17.54	0

12:28 PM	14.75	0
12:28 PM	48.08	0
12:28 PM	37.57	0
12:28 PM	23.10	0
12:28 PM	44.62	0
12:28 PM	15.78	0
12:28 PM	32.42	0
12:28 PM	23.49	0
12:28 PM	33.55	0
12:29 PM	36.02	0
12:29 PM	43.07	0
12:29 PM	48.68	0
12:29 PM	47.54	0
12:29 PM	22.88	0
12:29 PM	43.39	0
12:29 PM	30.64	0
12:29 PM	10.10	0
12:29 PM	55.44	1
12:29 PM	24.33	0
12:29 PM	48.75	0
12:29 PM	54.20	1
12:30 PM	11.90	0
12:30 PM	23.80	0
12:30 PM	41.88	0
12:30 PM	44.38	0
12:30 PM	46.68	0
12:30 PM	10.16	0
12:30 PM	25.73	0
12:30 PM	40.66	0
12:30 PM	31.87	0
12:30 PM	13.71	0
12:30 PM	42.82	0
12:30 PM	37.77	0
12:31 PM	21.86	0
12:31 PM	33.61	0
12:31 PM	18.99	0
12:31 PM	35.52	0
12:31 PM	19.82	0
12:31 PM	49.59	0
12:31 PM	42.57	0
12:31 PM	49.47	0
12:31 PM	47.32	0
12:31 PM	32.27	0
12:31 PM	87.39	1
12:31 PM	39.10	0
12:32 PM	25.07	0
12:32 PM	54.87	1
12:32 PM	49.27	0
12:32 PM	59.34	1

Appendix G

Timestamp to use	Vacuum	Pump activated	Low vacuum alarm	High vacuum alarm
12:24 PM	0.97	0	0	1
12:24 PM	0.21	1	1	0
12:24 PM	0.48	0	0	0
12:24 PM	0.31	1	0	0
12:24 PM	0.89	0	0	1
12:24 PM	0.74	0	0	0
12:24 PM	0.87	0	0	1
12:24 PM	0.58	0	0	0
12:24 PM	0.97	0	0	1
12:24 PM	0.35	1	0	0
12:24 PM	0.20	1	1	0
12:24 PM	0.45	0	0	0
12:25 PM	0.13	1	1	0
12:25 PM	0.89	0	0	1
12:25 PM	0.09	1	1	0
12:25 PM	0.21	1	1	0
12:25 PM	0.14	1	1	0
12:25 PM	0.16	1	1	0
12:25 PM	0.41	1	0	0
12:25 PM	0.16	1	1	0
12:25 PM	0.18	1	1	0
12:25 PM	0.73	0	0	0
12:25 PM	0.59	0	0	0
12:25 PM	0.16	1	1	0
12:26 PM	0.80	0	0	1
12:26 PM	0.54	0	0	0
12:26 PM	0.15	1	1	0
12:26 PM	0.95	0	0	1
12:26 PM	0.87	0	0	1
12:26 PM	0.16	1	1	0
12:26 PM	0.15	1	1	0
12:26 PM	0.37	1	0	0
12:26 PM	0.23	1	1	0
12:26 PM	0.80	0	0	1
12:26 PM	0.22	1	1	0
12:26 PM	0.00	1	1	0
12:27 PM	0.20	1	1	0
12:27 PM	0.09	1	1	0
12:27 PM	0.39	1	0	0
12:27 PM	0.93	0	0	1
12:27 PM	0.00	1	1	0
12:27 PM	0.38	1	0	0
12:27 PM	0.44	0	0	0
12:27 PM	0.14	1	1	0
12:27 PM	0.48	0	0	0
12:27 PM	0.24	1	1	0
12:27 PM	0.40	1	0	0
12:27 PM	0.83	0	0	1
12:28 PM	0.83	0	0	1
12:28 PM	0.59	0	0	0
12:28 PM	0.55	0	0	0
12:28 PM	0.62	0	0	0
12:28 PM	0.99	0	0	1
12:28 PM	0.49	0	0	0

12:28 PM	0.78	0	0	1
12:28 PM	0.28	1	0	0
12:28 PM	0.19	1	1	0
12:28 PM	0.81	0	0	1
12:28 PM	0.19	1	1	0
12:28 PM	0.48	0	0	0
12:29 PM	0.73	0	0	0
12:29 PM	0.88	0	0	1
12:29 PM	1.00	0	0	1
12:29 PM	0.60	0	0	0
12:29 PM	0.15	1	1	0
12:29 PM	0.46	0	0	0
12:29 PM	0.85	0	0	1
12:29 PM	0.44	0	0	0
12:29 PM	0.95	0	0	1
12:29 PM	0.75	0	0	0
12:29 PM	0.75	0	0	0
12:29 PM	0.07	1	1	0
12:30 PM	0.59	0	0	0
12:30 PM	0.97	0	0	1
12:30 PM	0.26	1	0	0
12:30 PM	0.13	1	1	0
12:30 PM	0.38	1	0	0
12:30 PM	0.02	1	1	0
12:30 PM	0.04	1	1	0
12:30 PM	0.48	0	0	0
12:30 PM	0.59	0	0	0
12:30 PM	0.71	0	0	0
12:30 PM	0.18	1	1	0
12:30 PM	0.95	0	0	1
12:31 PM	0.69	0	0	0
12:31 PM	0.50	0	0	0
12:31 PM	0.24	1	1	0
12:31 PM	0.33	1	0	0
12:31 PM	0.27	1	0	0
12:31 PM	0.19	1	1	0
12:31 PM	0.27	1	0	0
12:31 PM	0.40	1	0	0
12:31 PM	0.63	0	0	0
12:31 PM	0.60	0	0	0
12:31 PM	0.15	1	1	0
12:31 PM	0.87	0	0	1
12:32 PM	0.66	0	0	0
12:32 PM	0.59	0	0	0
12:32 PM	0.57	0	0	0
12:32 PM	0.95	0	0	1