MASTER OF SCIENCE
IN ENGINEERING

Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Software Engineering

Development of a unified interface for monitoring building heating systems

Author
Guillaume-Auguste Riondet

Under the direction of

Prof. Nabil Abdennadher
HES-SO, Genève

Lausanne, HES-SO Master, June 15, 2022

Acknowledgments

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to my supervisor, Mr Nabil Abdennadher, whose contribution, stimulating suggestions and encouragement, helped me all along of this project.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the staff of SG-Energies company lead by Christian De Preux, who gave me the company's vision for this project to complete my task. A special thanks goes to Mr Pierre-Alain De Preux, the IT Manager of SG-Energies, for the work we did together.

Last but not least, many thanks go to my coworkers, friends and family who encourage me all along of this project and provide precious reviews on this report.

Abstract

Buildings' heating installation needs a periodic check-up and maintenance to guarantee their full operability. Today caretakers and technicians work together to prevent any issue that might affect the comfort of the residents. Beside that, weather is different from year to year and some adjustments to the settings of the heating stations are necessary. To perform all of this, technicians come to check every one or two weeks the integrity of the installation and perform on site the changes needed. To avoid this, SG-Energies, a Swiss heating management company, launched a project: STEP. The goal of this project is to make a step forward on the monitoring and maintainability of heating rooms by deploying an "edge" device, namely SG-Box.

This report will present the proof of concept to perform this step. The SG-Box purpose is to reduce the frequency and duration of breakdown by communicating with the main control unit in charge of the heating installation. This in-depth project goal will prove that SG-Box can retrieve data from the heating installation and can inject data as well to perform adjustments remotely. Many brands exist on the market and don't have the same communication protocol. This proof of concept will present three pilot deployments of SG-Box on different sites regrouping two communication protocols: ModBUS and BACnet.

Key words: "Edge" device, Communication protocol, Remote monitoring

Résumé

Les chaufferies des bâtiments nécessitent un contrôle et une maintenance périodique pour garantir leur bon fonctionnement. Aujourd'hui concierges et techniciens travaillent ensemble pour prévenir tout problème qui pourrait nuire au confort des résidents. D'un autre côté, chaque année la météo étant différente, des ajustements de programmation de la chaudière sont nécessaires. Pour réaliser toutes ces tâches, des techniciens contrôlent en personne périodiquement l'intégrité de l'installation et effectuent les changements nécessaires. Pour limiter ceci, SG-Énergies, une société suisse de gestion d'installation de chauffage, a initié un projet : STEP. Le but de ce projet est de faire un pas en avant dans le domaine de la surveillance et de la maintenance de chaufferie en développant un dispositif "edge", nommé SG-Box.

Ce rapport présentera la preuve du concept pour réaliser cet objectif. Le but de la SG-Box est de, à terme, réduire la fréquence et la durée des pannes en exploitant la communication avec la régulation automatique qui contrôle la chaufferie. Plusieurs marques existent aujourd'hui et n'utilisent pas toutes le même protocole de communication. Cette preuve de concept présente trois déploiements pilotes de la SG-Box sur différents sites utilisant deux protocoles de communication : ModBUS et BACnet. Ce projet d'approfondissement prouvera que la SG-Box peut récupérer des données de l'installation de chauffage et saisir des données pour réaliser des ajustements à distance.

Mots clés : Protocole de communication, Surveillance à distance, Dispositif "edge"

Contents

Abbreviations

CSV Comma-separated values. 11, 19, 21, 25

DHCP Dynamic Host Configuration Protocol. 16

IoT Internet of Things. 1, 4

IP Internet Protocol. 6, 8, 9, 11, 16, 17, 19, 23

JSON JavaScript Object Notation. xiii, 11, 12, 13, 16, 22

LoRa Long Range. 1, 3

MQTT Message Queuing Telemetry Transport. xii, 4, 10, 11, 12, 13, 14

PLCs programmable controllers. 4, 7

RTU Remote Terminal Unit. 8, 15

SCADA Supervisory control and data acquisition. 3

SMS Short Message Service. 4

SQL Structured Query Language. 4

TCP Transmission Control Protocol. 8, 9, 19

TTN The Things Network. 3

VPN Virtual Private Network. 25

List of Figures

List of Listings

Introduction

Nowadays, buildings need to have their own heating installation to provide heat for apartments' heaters and domestic hot water. For comfort and sanitary reasons, those installations must fully work all the time. This implies that if an issue appears, it needs to be detected, handled and solved as quickly as possible to avoid any disturbance to the building's comfort. Managing heating rooms consists in doing periodic check-ups to monitor or adjust values like temperatures and performing visual check-up of the components. If a check-up reveals an issue, this one will be fixed by repairing the material directly or by placing an order to replace the broken piece. An experienced caretaker can perform a quick visual check of the installation and report any issue they might detect. In general, the company, mandated to perform this kind of maintenance, visits the installation regularly depending on the contract established with the building owner. By those facts, the delay of resolving an issue is mostly impacted by the amount of time elapsed between the issue's appearance and the moment of detection and handling.

SG-Energies[1] is a Swiss heating management company that is developing a project, named STEP. STEP aims to develop an Internet of Things (IoT)-based system to make a "step" forward by developing the capability to remotely monitor and control heating rooms. The main objective is to (1) reduce the frequency and duration of breakdowns. and (2) limit the unnecessary on-site visit of the technicians. To this purpose, SG-Energies has developed an "edge" device, namely SG-Box, to interface the heating installation with the IT infrastructure. SG-Box is charged to retrieve data from the heating installation and temperature sensors (placed in apartments communicating through Long Range (LoRa) protocol).

After focusing on the development of the SG-Box device, the STEP project is focusing on the pilot deployments step. Three places in Geneva have been selected by type of installation (gas, "remote heating", wood pellets), by brands (Sauter, Viessmann, Froeling) and by size (building of 30-40 apartments, building of 4 apartments, group of houses) to experiment with the deployment of the SG-Box. In each heating room, a device will be installed to implement and test the solution.

The goal of this in-depth project is to configure SG-Box and interface it with the main control unit of three different brands which support three different protocols of communication: ModBUS and BACnet. For each brand, firstly, the communication protocol needs to be discovered and handled. Secondly, read and write abilities from SG-Box on the main command unit needs to be developed to control and monitor the heating installation. The idea is to develop a generic workflow which supports different brands. The goal is to have the same type of data on the central server regardless of the communication protocol and the brand data structure.

This report is organised as follows: the first chapter will expose the technologies used, the communication protocols involved and the main deployment idea. The second and third chapters will deal respectively with the deployment carried out for two communication protocols, namely ModBUS and BACnet protocols. Finally, the fourth chapter will discuss about these two types of deployments.

1 | Technologies used in deployments

This chapter will expose the main deployment idea of SG-Box in a heating station and define the technologies and communication protocols used for the development of this project.

Before deploying a SG-Box in a heating room, SG-Energies had already deployed and setup the Ignition server and the setup of the SG-Box. The role of this server is to centralise data retrieval from any source. The setup of SG-Box consists of installing the 4G module to have internet connection through it, installing LoRa[2] module to have an operational gateway and installing NodeRED for programming the data points retrieval. The main deployment idea described below doesn't take in consideration the details of deployment depending on the communication protocol.

To achieve the goal of the project, any type of installation should correspond to the Figure 1.1.

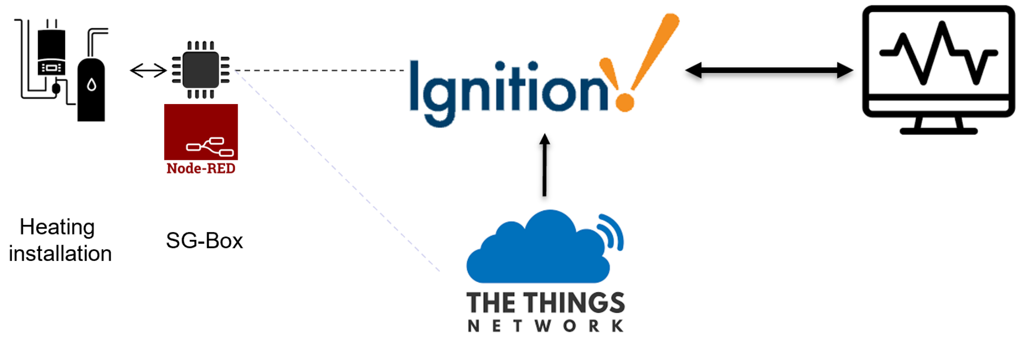


Figure 1.1 Schema of all used technologies in a deployment

The SG-Box is connected by wire to the automated control unit of the heating installation. The SG-Box needs to have an internet connection either by 4G or Ethernet to be able, on one side, to send data retrieval to the Ignition server and, on another side, to send LoRa packets by the gateway to The Things Network (TTN)[3] server. The TTN server will send back the LoRa data to Ignition.

1.1 Technologies

1.1.1 SG-Box

The SG-Box device is composed of a Raspberry Compute Module 4 and a LoRa module mounted on it (see Figure 1.2). A 4G dongle is plugged into the main board. Two antennas are connected to the box, one for the 4G network, the other for the LoRa network.

1.1.2 Ignition

Ignition is a software released by Inductive Automation in January 2010[4]. This is an integrated software platform for Supervisory control and data acquisition (SCADA) systems. The main features of this software are the capability to centralise data,

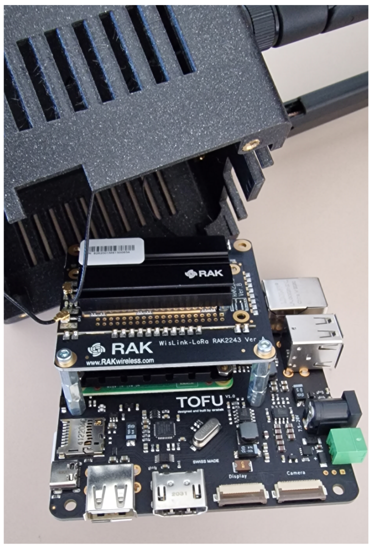


Figure 1.2 Photo of the intern composition of the SG-Box device

store them in Structured Query Language (SQL) databases and provide tools to build interfaces to visualise collected data. To achieve this, Ignition is composed of several modules. The main ones are SQL Bridge to connect programmable controllers (PLCs) and SQL databases (see Figure 1.3), Vision to design and run specific interfaces for data visualisation, Reporting to create dynamic PDF reports to have a regular overview of the managed data and Short Message Service (SMS) Notification Module to send alarm notifications directly through mobile network.

Ignition takes a central place in the STEP project. This software is responsible for receiving collected data from the different SG-Boxes and displaying them on graphical interfaces, alerting by SMS and creating operating reports for technicians and building managers.

1.1.3 NodeRED

Node-RED[5] is a programming tool based on NodeJS and offers a flow-based development tool for visual programming (see Figure 1.4). Its main goal is to connect hardware devices to online services as part of the IoT. All the development through NodeJS is based on the event-driven model. Each block is executed only if it receives an input: the payload. A flow is the addition of all blocks linked to perform a runtime at the execution. The interface provides predefined blocks to simplify the programming of the flow. JavaScript function blocks offer more flexibility to develop our own logic.

Node-RED is deployed on SG-Boxes and is responsible for interacting with the main control unit of a heating room and querying it in its own communication protocol to retrieve data. The second role of Node-RED is to format data to be sent to the Ignition server through a MQTT Broker.

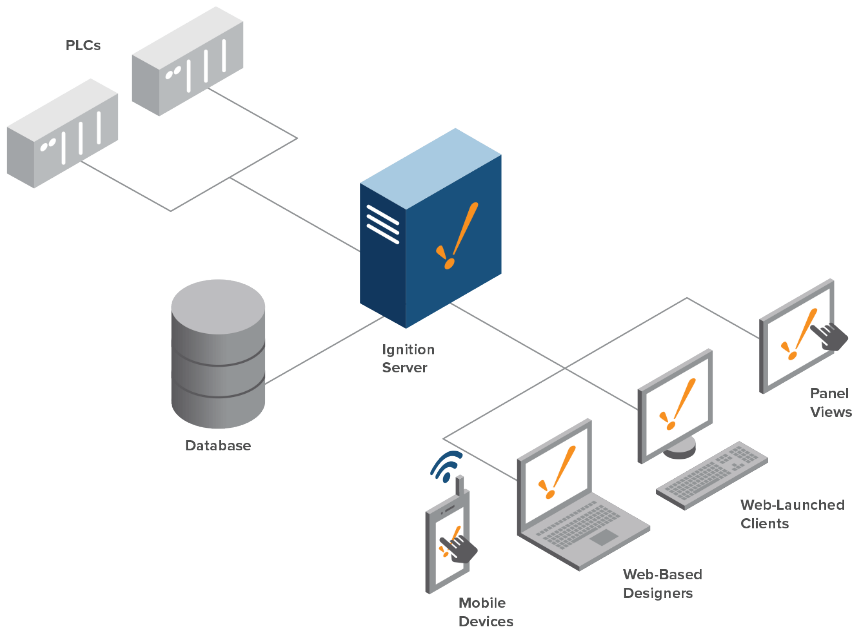
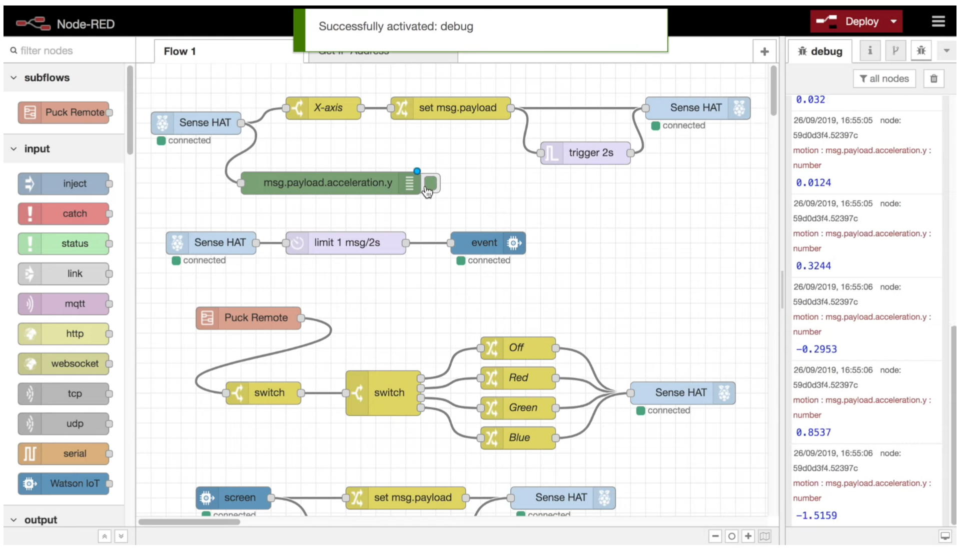


Figure 1.3 *Schema of Ignition server's place and inter-connections*
Source: *`www.inductiveautomation.com/scada-software/`*



```
0.032
26/09/2019, 16:55:05   node:
59d0d3f4.52397c
motion : msg.payload.acceleration.y :
number
0.0124
26/09/2019, 16:55:05   node:
59d0d3f4.52397c
motion : msg.payload.acceleration.y :
number
0.3244
26/09/2019, 16:55:06   node:
59d0d3f4.52397c
motion : msg.payload.acceleration.y :
number
-0.2953
26/09/2019, 16:55:06   node:
59d0d3f4.52397c
motion : msg.payload.acceleration.y :
number
0.8537
26/09/2019, 16:55:06   node:
59d0d3f4.52397c
motion : msg.payload.acceleration.y :
number
-1.5159
```

Figure 1.4 *Screenshot of NodeRED's interface*
Source: *`www.nodered.org`*

1.2 Communication protocols

1.2.1 BACnet

BACnet[6] is a communication protocol for Building Automation and Control (BAC) networks created by American Society of Heating, Refrigerating and Air-Conditioning (ASHRAE[7]). This protocol is used to control building systems such as heating, ventilating and air-conditioning. To manage those systems, a set of rules is set and the automated control unit has to manage the system by applying those rules when needed with the help of sensors (temperature, pression. . .). The fiability of this protocol is given by standards it meets (ANSI[8] and ISO 16484-5[9]). Those standards specifies 60 standard object types. Each of the 60 different types are not necessarily needed in every system. Each data point can be accessed by a unique couple of integers :

- the ID of the "object type"
- the ID of the "object identifier" in the given type (called also "object instance").

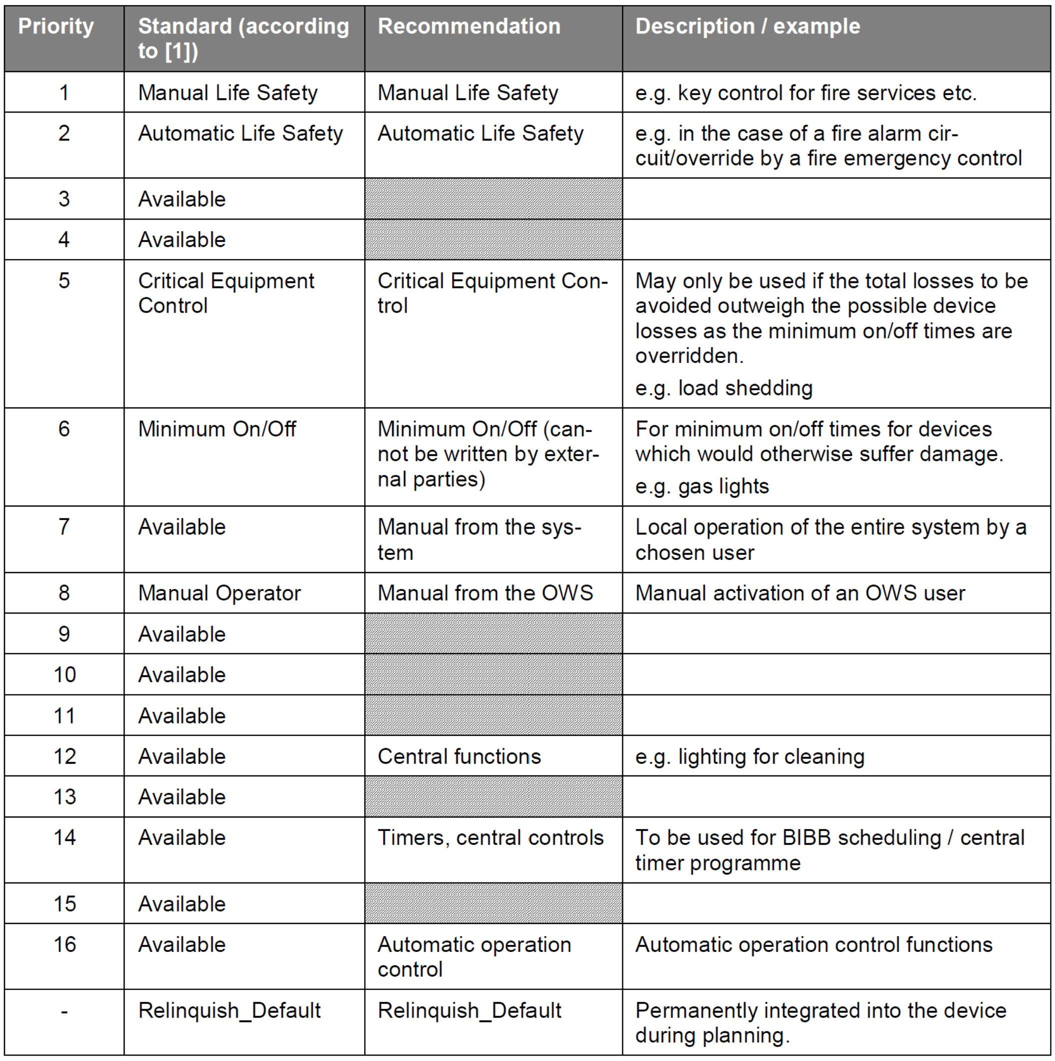
Other fields in a data point are mandatory in addition to the previous ones :

- The "object-name", to allow human recognition of the data point more simply. A unique string ID can be used.
- The "device ID", to identify to what automated control unit the data point is attached to.

All of these mandatory fields do not provide the core information, the "Present-Value" optional field is the field with the actual value measured for a sensor (readable only) or the value set for a rule of runtime (readable and writable).

This protocol is based on rules priority. Each writable data point is described by an array of 16 inputs that can be different (see Figure 1.5). The index of each command in the "Priority Array" is important, the one with the higher priority will be taken as valid for the automated control unit and set in the runtime of the system. Priority 0 is the higher priority used by fire services for example to override any other previous input in case of emergency, Priority 8 is used when a command is set manually by an operator and Priority 16, the lower priority, is the rule set to run the installation normally.

The BACnet protocol can be deployed on two communication materials (Layer 2 of the OSI model) different. The first one uses Ethernet and inter-connects the devices through Internet Protocol (IP): BACnet/IP implementation. The second one uses serial communication through serial port: MS/TP Protocol[11]. The first one offers more security, performance and accessibility to the BACnet installation between different BACnet devices all around the building installation than the second one.



Priority	Standard (according to [1])	Recommendation	Description / example
1	Manual Life Safety	Manual Life Safety	e.g. key control for fire services etc.
2	Automatic Life Safety	Automatic Life Safety	e.g. in the case of a fire alarm circuit/override by a fire emergency control
3	Available	▨	
4	Available	▨	
5	Critical Equipment Control	Critical Equipment Control	May only be used if the total losses to be avoided outweigh the possible device losses as the minimum on/off times are overridden. e.g. load shedding
6	Minimum On/Off	Minimum On/Off (cannot be written by external parties)	For minimum on/off times for devices which would otherwise suffer damage. e.g. gas lights
7	Available	Manual from the system	Local operation of the entire system by a chosen user
8	Manual Operator	Manual from the OWS	Manual activation of an OWS user
9	Available	▨	
10	Available	▨	
11	Available	▨	
12	Available	Central functions	e.g. lighting for cleaning
13	Available	▨	
14	Available	Timers, central controls	To be used for BIBB scheduling / central timer programme
15	Available	▨	
16	Available	Automatic operation control	Automatic operation control functions
-	Relinquish_Default	Relinquish_Default	Permanently integrated into the device during planning.

Figure 1.5 BACnet priority array and description[10]

1.2.2 ModBUS

ModBUS, named after the creator company name, is a data communication protocol published in 1979 by Modicon for use with its PLCs. With time, this protocol came rapidly to a standard and was used for connecting industrial electronic devices because it was openly published and royalty-free[12].

ModBUS protocol separates the data object into four different types addressable through, at maximum, 50000 different addresses. One address corresponds to one data point with one value. To read each type of data ModBUS protocol proceeds through different function code which accesses a specific split of addresses where a certain type of data is stored. This is the same concept for writing value into data objects, but not all the registers can be written, some of them are only giving information from a sensor and it will be nonsens to try to change their value.

No description is provided when a data point is read. A database needs to be present beside to identify the correspondence of each data point.

The ModBUS protocol can communicate through different format :

- Remote Terminal Unit (RTU)
- ASCII
- Transmission Control Protocol (TCP)

The first two formats communicate with serial communication on RS-282 or RS-485. The last one is based on Ethernet and IP protocol.

However, today the protocol shows its limitations, by its age the number of data types is limited and thus large binary objects are not supported. There is no standard way to design and describe a data object. ModBUS protocol does not guarantee any security against interception of data or unauthorised commands.

2 | ModBUS Deployments

This chapter will present two deployments of the SG-Box working with the ModBUS protocol. It will show chronologically the events of the work done and the issues encountered for this protocol with SG-Box.

2.1 Viessmann : Vernier city

The heating station in Vernier is responsible for providing heat for 4 apartments and is composed of a boiler working with gas and a hot water tank.

The brand of the components of the heating installation is Viessmann[13] and provides a device called Vitogate[14] to open a gateway for accessing the data through BACnet over TCP/IP or ModBUS over TCP/IP. For testing purposes for the future of STEP project it had been chosen to work with ModBUS over IP.

The Figure 2.1 describes the installation and the hardware installation of the SG-Box.

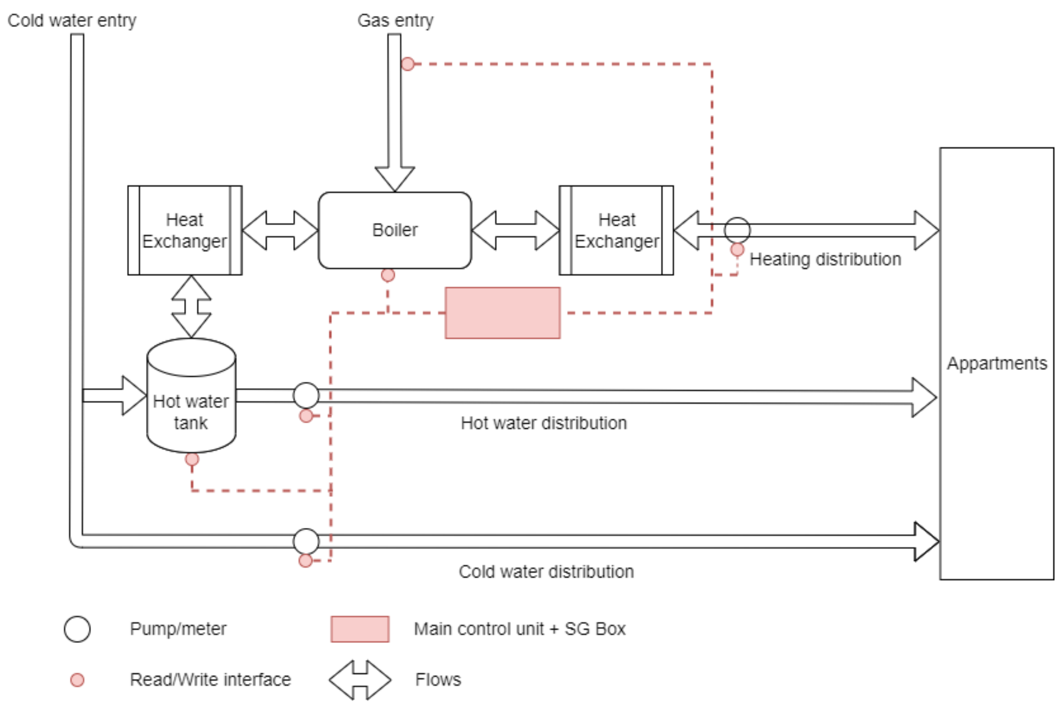
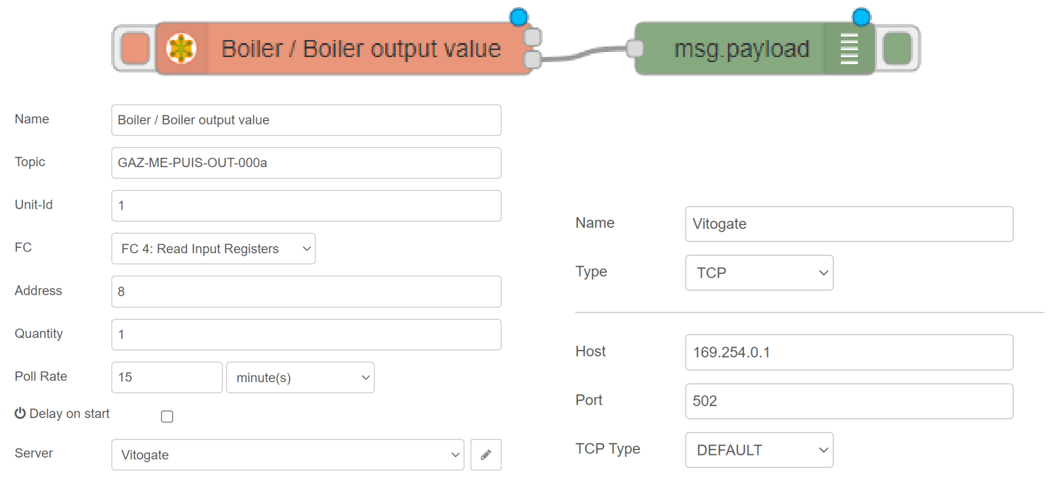


Figure 2.1 Schema of the heating installation of Vernier city with the SG-box

The Vitogate device provides a server-interface to configure correctly which data points from the device that needs to be accessed, on which port depending on the communication protocol (port 502 for ModBUS).

As this is the first installation to be set up on ModBUS protocol, it was decided to proceed in a step by step process to check and learn all the read and write capability of NodeRED on the Vitogate device. The first step consists of testing the part of the workflow from the Vitogate to NodeRED for one point of data. To do this, the following flow on the Figure 2.2 represents the visual programming of this test. The orange block[15] is responsible for creating the query with the given parameters listed below, sending it to the Vitogate and sending the response to the next block, corresponding here to the green block for printing the message in the NodeRED's debug section.



Name	Boiler / Boiler output value
Topic	GAZ-ME-PUIS-OUT-000a
Unit-Id	1
FC	FC 4: Read Input Registers
Address	8
Quantity	1
Poll Rate	15 minute(s)
Delay on start	☐
Server	Vitogate

Name	Vitogate
Type	TCP
Host	169.254.0.1
Port	502
TCP Type	DEFAULT

Figure 2.2 Single data point read flow : visual-programming representation and parameters of the test on the Vitogate device

After some adjustments on how to correctly setup the Vitogate connection in the Server field (Figure 2.2), read commands can be performed. From this, information retrieved needs to be inserted in a format for sending through MQTT to the Ignition server. To perform this step, the message from the orange ModBUS block needs to be processed by a JavaScript function to select the value needed in the raw message (Figure 2.3), in order to add a timestamp and link it to a name for automatic classification in Ignition (see Listing 2.1).

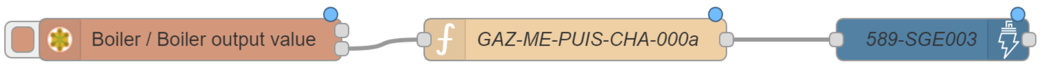
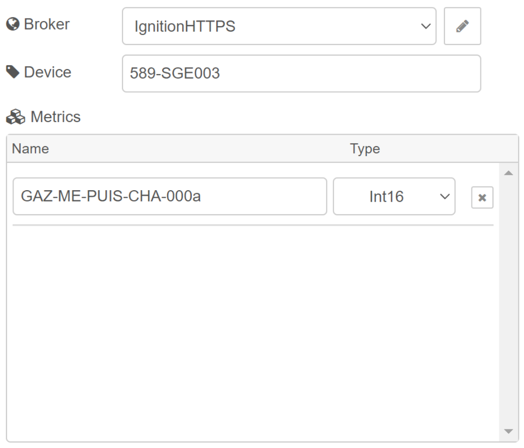


Figure 2.3 Single data point read flow : full workflow

```
 msg.payload = {
            "metrics": [
                {
                    "name":"GAZ-ME-PUIS-CHA-000a",
                    "timestamp" : new Date(),
                    "value": msg.payload.data[0]
                }]};
return msg;
```

Listing 2.1 JavaScript code for processing and formatting raw data for sending to Ignition

The final blue block of this flow is a SparkPlug MQTT block[16]. Each data referenced in the metrics array in the Listing 2.1 needs to be referenced in the configuration of the block to be correctly sent to Ignition (Figure 2.4).


Broker: IgnitionHTTPS
Device: 589-SGE003
Metrics

Name	Type
GAZ-ME-PUIS-CHA-000a	Int16

Figure 2.4 MQTT Spark block configuration

As the workflow works for one data point, a solution needs to be conceptualised, to handle a flow that recovers not just one data point but about fifty data points. The first solution produced is the naive one, to see how the Vitogate and NodeRED handle it. This solution works by replicating the same flow 50 times to see how it performs, with one flow per data point (Figure 2.5 shows the first data points retrieval).

From this naive solution two main issues can be seen. The first one is how complicated it is to maintain a visual-code like this and how big the work will be for each installation that will work with ModBUS over IP. The second one is about optimization. This implementation launches all commands at the same time and amplifies the risk of timeout by the Vitogate, NodeRED runtime and especially the Spark Plug MQTT push. It has been observed, by testing, that Ignition doesn't receive all data at every pull rate, the amount of data received is different between each pull.

As a goal to gain time on a new installation and to ensure that all data is correctly retrieved and sent to Ignition, time was taken to find a solution that allies both of criteria.

At this time, it is known that data points can be read one by block. By going through NodeRED's modbus library, there is a block that can query and retrieve multiple values from a starting address, the "ModBUS Flex Getter" block[15][17]. As mentioned before, ModBUS protocol doesn't provide any object description when a query is made. To know the correspondence between the value and its name, a table containing name and address is stored in an Excel sheet. The purpose here is to transform the data of the Excel sheet to take only relevant columns for retrieve data and link the obtained value with the name corresponding to send it properly to Ignition after. To avoid doing all table entries by hand, a Python script was created to read the Excel sheet, export it in Comma-separated values (CSV), and create a JSON file per each function code to read data (FC 1 to FC 4).

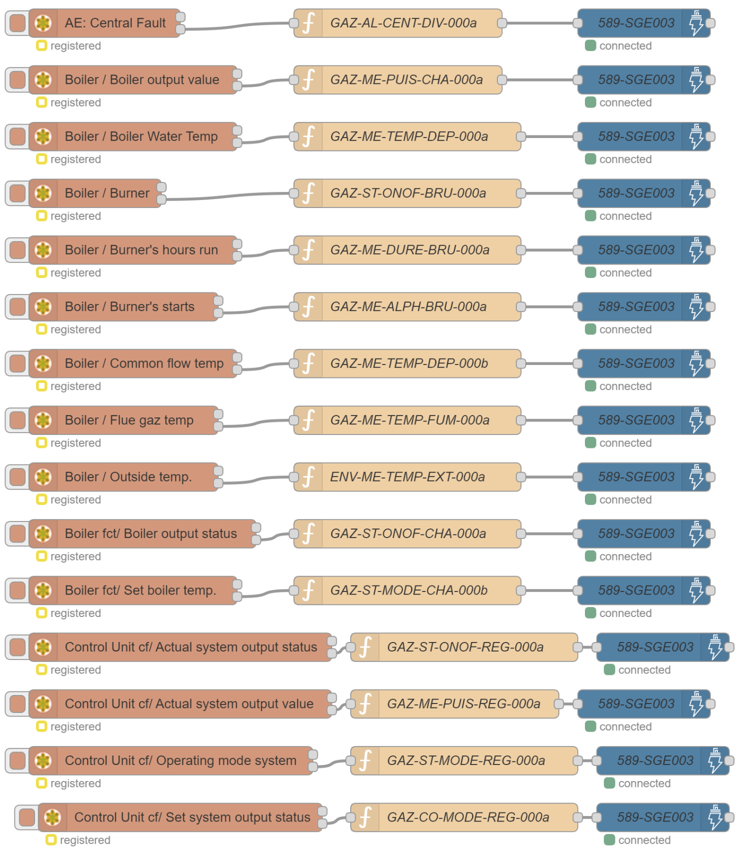


Figure 2.5 Naive solution : Multiple single reads visual-programming representation

Each JSON file has the following format (example in the Listing 2.2) :

- id : Function code
- items : array of object containing an address and the corresponding name
- max : Higher address in the file

```
{
    "id": 1,
    "items": [
        { "address": "8", "name": "SEC-CO-ONOF-SEC-100a" },
        { "address": "9", "name": "SEC-CO-ONOF-SEC-200a" }
    ],
    "max": 9
}
```

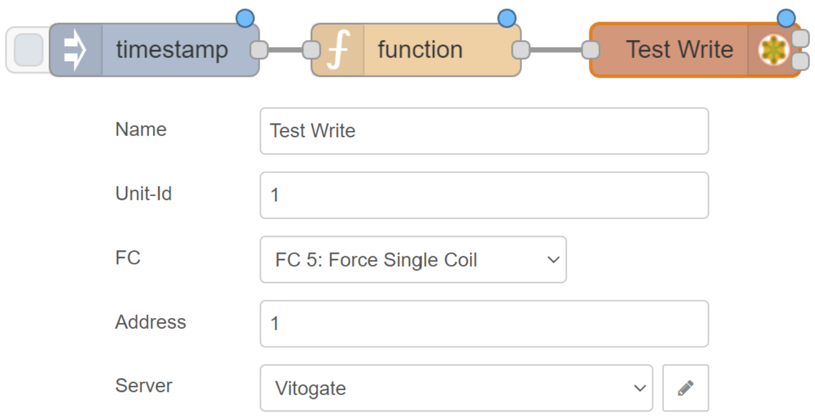
Listing 2.2 JSON data points file for ModBUS protocol on function command 1

The “max” field is present to avoid a full path through the file to find the higher address on each pull of data. The NodeRED’s flow will have in response to the query all the values between the start address and the “max” field value and it will select only data objects listed in the JSON data points file. However the SparkPlug MQTT block is not dynamically configurable, so the same scripting idea is applied to this block. In NodeRED, every item is stored in JSON format. Knowing this fact, another Python script is charged to fill the block with the name and the type of each value. Then, the JSON block file generated needs to be imported manually.

To summarize, the steps of the generic workflow are :

- Set up phase
  - Generate JSON files with data points names and addresses
  - Generate JSON file for the Spark Plug MQTT block
- Running phase at each pull
  - The four function code queries are done in parallel
    - Open the JSON data points file
    - Retrieve "max" field
    - Cache the file content
    - Send the corresponding query to the Vitogate
    - Merge raw data with data points file
    - Format data for Spark Plug MQTT block for sending
  - Merge the four messages
  - Format data for Spark Plug MQTT block for sending
  - Send it through the MQTT block

As the data point files are not supposed to be modified by an user and are only generated by the Python Script, only content errors from the Excel sheet, such as wrong address number or wrong name, are possible in this scenario. The reading/pull feature being complete, the writing/push feature needs to be tested as well. It had been chosen to proceed with the same step by step technique. First, tests between NodeRED and Vitogate are performed. To achieve this, another block is used : "ModBUS - Write" block. To trigger this block, a timestamp needs to be injected in it to send a value (set in the function block) at the block entry point. The block is correctly configured with a function command to order a writing operation on the specified address (see Figure 2.6).



Name	Test Write
Unit-Id	1
FC	FC 5: Force Single Coil
Address	1
Server	Vitogate

Figure 2.6 *Single data point write flow : visual-programming representation and parameters of the test on the Vitogate device*

The writing operation's correct execution is verified by reading the value at the mentioned address in the writing block.

Next, the Ignition to NodeRED communication is tested (Figure 2.7). Ignition will send MQTT messages on a topic and NodeRED will subscribe to the same topic with the Sparkplug MQTT Block. When a message is published on the topic, this block is triggered and will send the content of the message to the next block connected. A simple print in the debug section of NodeRED helps verifying the feasibility of the if the concept.

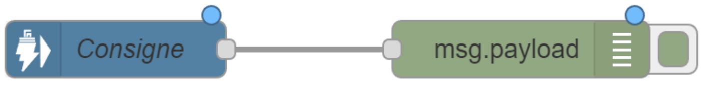


Figure 2.7 *Visual-programming flow for testing the Ignition to NodeRED communication*

Finally, the full writing workflow (Figure 2.8) is tested and works.

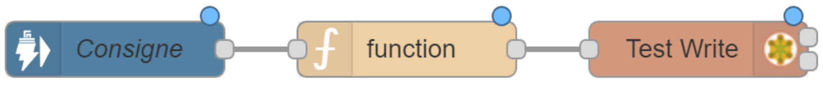


Figure 2.8 *Visual-programming flow for testing the full writing workflow communication*

Some issues were encountered during this deployment and some are common with the next one, all of them are listed after the next subchapter.

This deployment is 90% operational. All the reading features are functional and the data is retrieved by Ignition correctly. Only the writing side needs to be optimised for multiple writes in a single command but it will be tested later in the STEP project.

2.2 Froeling : Bellegarde road (Chancy city

The heating station in Chancy is responsible for providing heat for a dozen allotments and is composed of a boiler working with wood pellets and two hot water tanks.

The brand of the components of the heating installation is Froeling[18] and provides an entry point with a serial connector RS-282. The communication uses RTU Buffered frame and is supported by NodeRED. The RS-282 connector is plugged as a USB-type cable in the SG-Box.

The Figure 2.9 describes the installation and the hardware installation of the SG-Box.

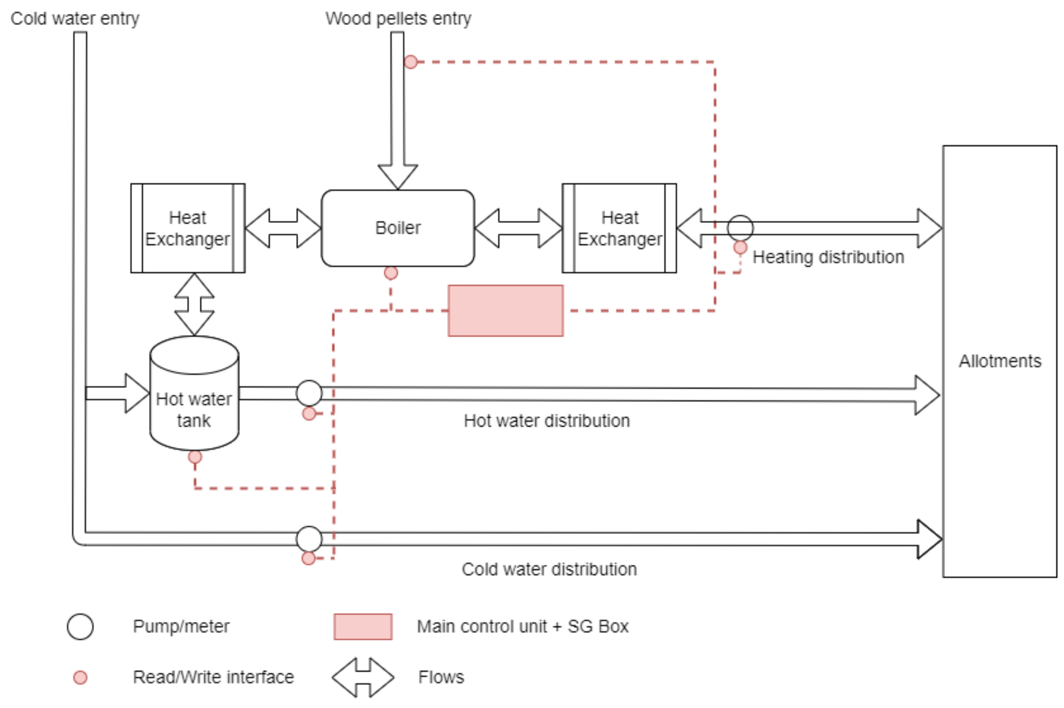
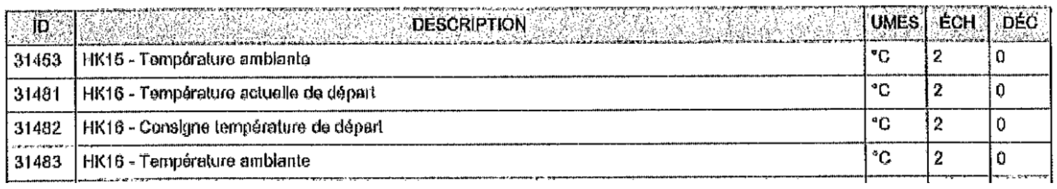


Figure 2.9 Schema of the heating installation of Bellegarde road with the SG-box

This deployment starts shortly after Vernier's one but it's very dependent on it. In fact, the first step was to make a "first-contact" with the main control unit. To achieve this, a simple flow to retrieve one data point is produced. The documentation says that RTU frame protocol is supported, during the first attempts, sometimes data was retrieved from the query and some times, a buffer error appeared. After some research, NodeRed's roll-up menu of the ModBUS client configuration provides two options for RTU usage : RTU and RTU-BUFFERD. After switching to this last option, the data recovery is stable and the next step can be started.

As it is another brand, a verification of when a data point is addressed needs to be done to ensure the veracity of the value. The verification consists of reading precise data points which can be read on the console of the main control unit directly. This test reveals two points where attention is needed within this installation. The first one is that some values have a multiplicator referenced in the documentation and need to be taken into account when the value will be processed later in Ignition. The second one is the conversion of the address between the documentation and the NodeRED configuration. The documentation shows us raw addresses (Figure 2.10) that need to be transformed in the correct function code and beside that the theoretical address must be subtracted from 1 to match the real one.



ID	DESCRIPTION	UMES	ÉCH	DÉC
31453	HK15 - Température ambiante	°C	2	0
31481	HK16 - Température actuelle de départ	°C	2	0
31482	HK16 - Consigne température de départ	°C	2	0
31483	HK16 - Température ambiante	°C	2	0

Figure 2.10 *Extract from the Frohling main control unit documentation : Data points addresses*

At this time the generic flow for a ModBUS installation was in development. To avoid any loss of time, the Chancy development was paused until the end of the development of the generic flow. Once the flow works on the Vernier's deployment, tests on Chancy development in order to prove the genericity of the flow, are made. To prove this, JSON data points files corresponding to this installation are generated. The generic flow works perfectly without any change in it, except the ModBUS Client configuration, which is mandatory to correctly communicate with the main control unit.

As the Ignition to NodeRED communication, in the writing workflow, already works, only the NodeRED to main control unit needs to be tested. Due to issues listed below, this test still needs to be performed with the same steps as the other ModBUS deployment.

This deployment is 85% operational. All the reading features are functional and the data is retrieved by Ignition correctly. Only the writing side has to be tested for single writes.

2.3 Issues encountered

The biggest problem encountered that impacted the most the project is the 4G network. At the beginning of the deployments, sometimes at the boot phase of SG-boxes, the software in charge to get a connection through the dongle was not able to register itself on the mobile network. In this failure state, only a manual action could try to restart the connection. Either a reboot of the box or a network's research restart directly on the software. At some point, even a reboot or a manual action on the software did not connect the SG-Box to the internet through 4G. Even sometimes a dozen reboots were needed to finally make the connection work.

Another impact of the malfunction of the 4G network is the local IP address given by the software when 4G was not connected. When this case occurred, the software gave an IP address of this type to the device: 169.254.x.x /16. This IP address can mean that the Dynamic Host Configuration Protocol (DHCP) server couldn't give a valid local IP address[19]. This is not the central problem. Even if the internet were not accessible, tests could be done between the SG-Box and the main control unit but not with Vitogate in this case. It had been discovered late in the development of the project that the Vitogate IP address had the same type: 169.254.x.1/16. This case creates a routing conflict, and the SG-Box cannot address the traffic in the right direction. To be able to work locally just between the Vitogate and the SG-Box, the 4G network interface was shut down to resolve the IP address conflict.

Besides the issues encountered and the debug time allowed to them, those ModBUS deployments are mostly successful and need to be completed to be fully operational.

3 | BACnet Deployment

This chapter will present one deployment of the SG-Box working with the BACnet protocol. It will show chronologically the events of the work done and the issues encountered for this protocol with SG-Box.

3.1 Sauter : Yvoi boulevard

The heating station at Yvoi Boulevard in Geneva is responsible for providing heat for more than fifty apartments and is composed of "remote heating" installation[20][21][22], two hot water tanks and a heat pump. This deployment was done the same period as the Vernier's one.

The components' brand of the components of the main control unit is Sauter and provides a device called Sauter modulo[23] to manage and open a gateway for accessing the data through BACnet over TCP/IP.

The Figure 3.1 describes the installation and the hardware installation of the SG-Box.

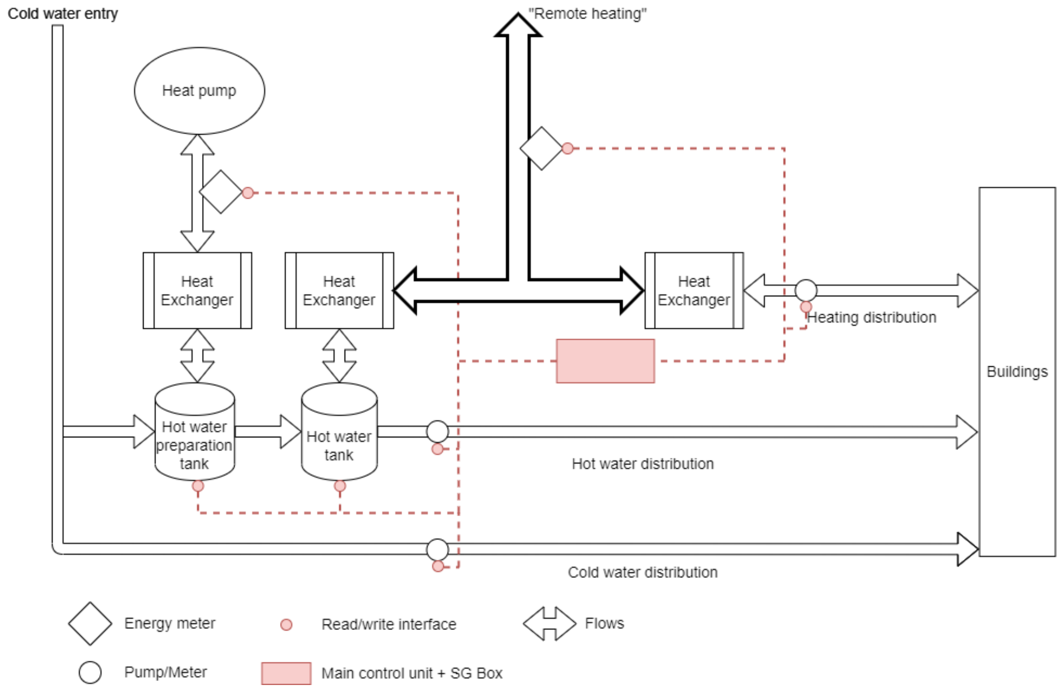
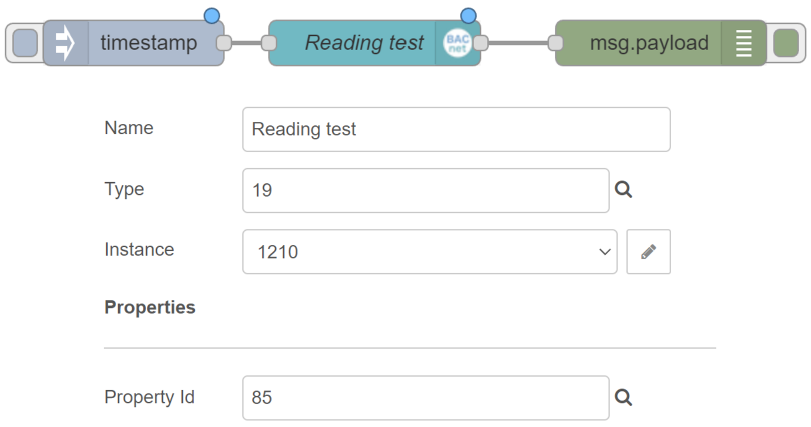


Figure 3.1 Schema of the heating installation of Yvoi boulevard with the SG-box

As this is the first installation to be set up on the BACnet protocol, it was decided to proceed step by step to check and learn all the read and write capabilities of NodeRED on the Sauter modulo device. Like the Vernier deployment, the first step consists of testing how to read a data point from the Sauter Modulo device. To perform this test, the data points' mapping in the main control unit needs to be known. With a CSV file provided by a Sauter technician responsible for the Yvoi Boulevard installation, reading tests are performed. The central blue block ("BACnet - Read" block[24]) on Figure 3.2, when receiving a message, a timestamp here, send a query to the Sauter Modulo device to ask about the value corresponding to the “Property ID” (85 means “Present value”, the field containing the actual value in the system) of the instance 1210 of type 19 (see Figure 3.2).

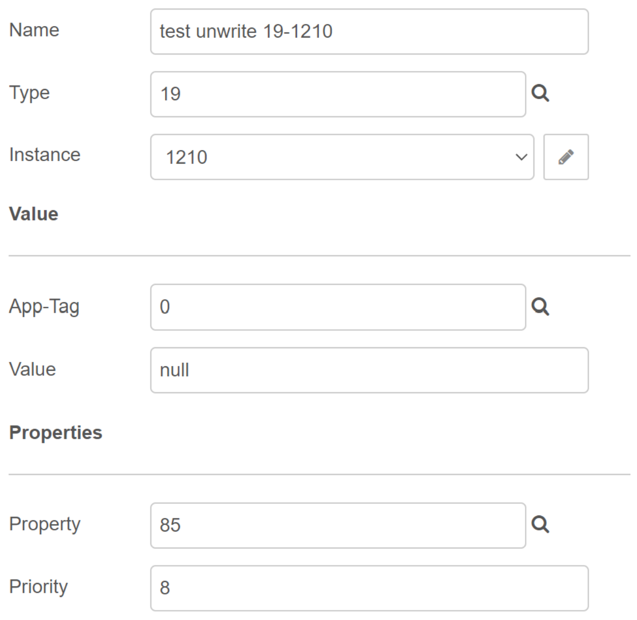


Name: Reading test
Type: 19
Instance: 1210
Properties
Property Id: 85

Figure 3.2 Single data point read flow : visual-programming representation and parameters of the test on the Sauter device

A few days later, SG-Energies organised a meeting between the development team and the Sauter technician to offer the opportunity to have a direct explanation of how the Sauter Modulo works and to be sure that all data points can be accessible by the SG-Box device. During this meeting, some reading tests and writing tests were made. It lead to discover important aspects of writing. A writing instruction can only be made by setting the priority of the new command to place the command in the priority array. This important aspect that needs to be taken care of, is how to revert and erase this writing. As mentioned before, most of the time a manual input will be set at a level priority of 8 out of 16 and will probably be temporary. So later, the previous state needs to be recovered, to keep the priority system viable. An official document of the Swiss Confederation[10], describes the main rules to use BACnet protocol to have the similar architecture between different buildings around the country. The description of our issue is clearly mentioned and says to write to the desired object a new value of type NULL (App Tag ID = 0) with a null value in it in the “Present-Value” (ID = 85) field at the priority concerned by the deletion. The Figure 3.3 shows the configuration to do this operation in NodeRED.

The problematic of the generic flow had already started in the Vernier deployment by this time. As it is not the same NodeRED’s blocks and thus not the same library, a new implementation of the same generic flow must be designed with the BACnet’s blocks library. The same logical idea is preserved, only the query part is adapted. After some research, it appears that the library is simply accessible directly by JavaScript code and not only through blocks of visual programming. A feature of multiple reads in a single instruction is accessible only by this way and needs the address of the BACnet server device (Sauter Modulo) and an array regrouping which field of which object type and instance needs to be retrieved (see Listing 3.1).



Name	test unwrite 19-1210
Type	19
Instance	1210

Value

App-Tag	0
Value	null

Properties

Property	85
Priority	8

Figure 3.3 Parameters of the BACnet write block for "unwrite" operation

```javascript
client.readPropertyMultiple('10.11.72.1', requestArray, (err, value) => {
    if(err){
        msg.payload = err;
    } else {
        msg.payload = value.values
    }
    node.send(msg)
});
```

Listing 3.1 JavaScript code for multiple BACnet readings

To do this, a Python Script adapted to the CSV file given and to the “requestArray” variable format is created to avoid copying about 150 data points information by hand. The fields to retrieve are the name (ID = 77) , description (ID = 28) and present value (ID = 85) of the data point. The Listing 3.2 show a data points file for BACnet multiple read command.

```json
[
    {
        "objectId": { "type": "5", "instance": "2021" },
        "properties": [{ "id": 28 }, { "id": 77 }, { "id": 85 }]
    },
    {
        "objectId": { "type": "2", "instance": "1173" },
        "properties": [{ "id": 28 }, { "id": 77 }, { "id": 85 }]
    }
]
```

Listing 3.2 JSON data points file for BACnet protocol

The first try of multiple reads is launched. A timeout error is returned by the Sauter device. To check that the reason is the presence of too many element in a query, the size is successively divided by 2 until reach the size of 10 data points for one query that is returning a result. Listing 3.3 shows an example of an element returned in the query response.

```
"objectId": {
    "type": 5,
    "instance": 2021
    },
    "values": [
        {
            "id": 28,
            "value": {"value": "INTER VIRTUEL COMMANDE PAC"}
        },
        {
            "id": 77,
            "value":{"value": "1_CL301_ECS_INT02_LI"}
        },
        {
            "id": 85,
            "value": {"value": 1}
        }
    ]
}
```

Listing 3.3 Sauter JSON device query response to read operation

This limitation can be due either to the device which can be quickly overloaded or due to the library itself.

To make the entire "requestArray" work, the array is split into the smallest array of size lower or equal to 10 and each array is sent in a query every 3 seconds to avoid overloading on the main control unit. All the queries' responses are merged together to be properly formatted and sent to Ignition.

The writing workflow doesn't need any more tests at this moment because it is known that the Ignition to NodeRED part and NodeRED to Sauter device part are both working well.

This deployment is 90% operational. All the reading features are working and the data is retrieved by Ignition correctly. Only the writing side needs to be optimised for multiple writes in a single command but it will be tested later in the STEP project.

3.2 Issues encountered

Besides the 4G network issue mentioned before and also heavily present in this deployment, another network problem took time to be resolved. In this deployment (and in the Vernier's one too) the SG-Box uses two network interfaces at the same time : the Ethernet one, connected to the main control unit and the 4G one, connected to the internet network (see Figure 3.4). Internet only being accessible through the 4G interface, automated computed network metrics[25] of the device send all the traffic for the internet into the Ethernet interface and can't go any further.

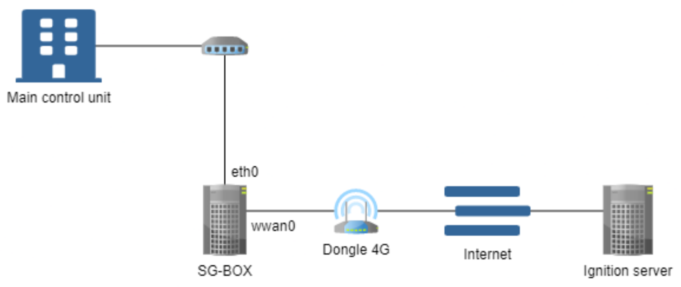


Figure 3.4 Network schema of SG-Box using two interfaces : 4G and Ethernet interfaces

To ensure this issue is resolved and won't be a problem in the future with the 4G network, the Ethernet interface metric is set at 9999 and guarantees that the 4G interface will be prioritised over Ethernet one because it will have a lower metric (lower than 1000 after 20 tests). This patch is essential because access to the Ethernet interface is still needed to reach the main control unit by addressing it directly with the IP in the JavaScript function. This patch will be reverted when an Ethernet connection to the internet will be accessible in the heating room because all devices (SG-Box and main control unit) will be in the same subnetwork and connected to the same switch with an internet gateway directly connected to it (see Figure 3.5).

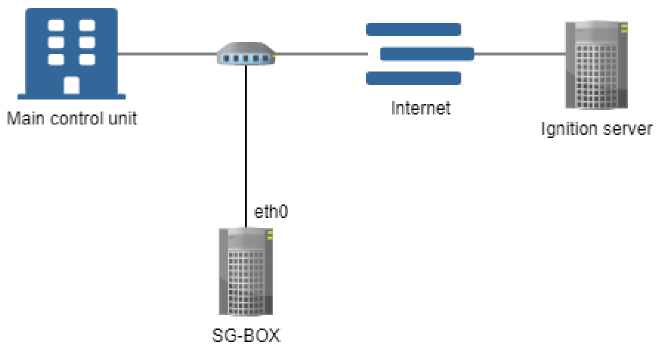


Figure 3.5 Network schema of SG-Box using only one interface : Ethernet interface

The other main issue of this deployment, as mentioned above, is the pool rate limitation on the Sauter device when doing a multiple field reading query. An optimised solution had to be found to avoid having to switch to multiple single field methods and to imply a big loss of time by reducing the generecity of the flow.

4 | Discussion

This chapter will expose the common solutions implemented in every deployments and will discuss some topics related to those ones.

4.1 Common solutions implemented

At the beginning of the project, SG-Energies presented the SG-Box and how they were working on a deployment. Every time a change was made in a flow, they went to the heating installation to update the SG-Box. No remote solution existed. A Virtual Private Network (VPN) solution has been quickly proposed for its many advantages. With this solution modifications and updates on the SG-Box could be made remotely and avoid many trips to heating stations. The OpenVPN[26] solution has been installed and tested. The VPN server was installed on the same machine as the Ignition server, and each SG-Box has a unique credential to quickly identify which box is connected. Each person who has to interact with it also has a unique credential too. A procedure was made on how to install and connect a new SG-Box to the VPN server, to be used in future SG-Box deployments. The only issue caused by this VPN installation was on the name resolving. When the VPN was running, the internet was accessible only through IP addresses and not standard URL. It was solved by setting DNS manually in the device network configuration.

4.2 Deployments discussion

The 4G network issues were detrimental to keep a good rhythm during the whole project. There was no guaranty that the network would work as planned. This fact implies that many trips to heating rooms were useless because no work could be done. After some trial and error, SG-Energies chose to install a wired internet connection in the heating station to get an Ethernet connection on the SG-Box, thus ensuring connection stability. Right before the end of the project, Chancy's deployment was running with a wired connection and was online every day of the week. Due to the delay to make this switch, this issue involved making very little progress each time a trip was made to a heating room. The work was prepared beforehand with caution to maximise the chance of success once on-site.

In all deployments, data points files need to be generated from a CSV through a Python script. During this project, a Python script was written for each type of CSV document to avoid selecting by hand relevant information to put in data points files. For future usage, the data points table should be in the Ignition server so that NodeRED could request it, or Ignition could generate and send a new file to the SG-Box when a change is detected with the Python scripting feature available in the software. However, the rest of the flows will stay unchanged.

The idea of generic flow based on the same logic is functional on all the deployments. Only code library and data points files needs to be changed to match the correct protocol to communicate with. When a comparison between the two protocols is made, it reveals that ModBUS is clearly more older than BACnet but it still works. The difference is in the wealth of information provided by the main control unit when a request is made. On this point BACnet is more developed and only an abstract list of data points IDs is sufficient to retrieve the name and the value of each point. However, with this implementation on the Sauter brand, it reveal a disadvantage: the number of points retrieved by request. On the other side ModBUS doesn't show this limitation. This BACnet query limitations need to be confirmed on other brand device using this protocol.

Conclusion

The main challenge of this project was to prove that, with an external device, the SG-Box, data retrieval and injection from a heating installation could be possible. To overcome this challenge, three brands of main control units on three different sites were selected that use two different communication protocols : ModBUS and BACnet. Once the protocols theory was discovered, each deployment used a step by step approach to validate each read and write capacity and group them in a second time to make a global workflow.

The initial objectives of the projects was to be able to interface the SG-Box between a main control unit and the Ignition server and interact with it. This main objective is fully achieved. On all deployments, read capabilities are operational and pulling every hour, while write capabilities have been tested (except on Chancy deployment) and are working well for a single write. Some of the work went beyond the initial objective by deploying a remote access on the SG-Box to have more flexibility to develop on them.

Some important issues have been encountered during the project, essentially network ones. These problems did not stop the project, they only slowed it down and forced it to always search for an alternative to the first idea. Finally, those issues result by having stronger results.

This proof of concept on two different communication protocols now opens the field to make more deployments on brands using one of them and reuse the work of this project. To go further, the collected data needs to have their interface to be visualised to offer the possibility to monitor and maintain heating stations by technicians. Adding external information to it like the weather data to observe behavioural patterns of the buildings for improving the added value of this data retrieval made by the SG-Box.

References

[1] *SG-énergies SA - Genève*. URL: https://www.sg-energies.ch/.

[2] *LoRaWAN*. fr. Page Version ID: 191845413. Mar. 2022. URL: https://fr.wikipedia.org/w/index.php?title=LoRaWAN&oldid=191845413.

[3] The Things Network. *The Things Network*. en. URL: https://www.thethingsnetwork.org/.

[4] *Powerful Control & Data-Visualization Software | Ignition SCADA*. URL: https://inductiveautomation.com/scada-software/.

[5] *Node-RED*. URL: https://nodered.org/.

[6] *BACnet*. fr. Page Version ID: 176989275. Nov. 2020. URL: https://fr.wikipedia.org/w/index.php?title=BACnet&oldid=176989275.

[7] *Home | ashrae.org*. URL: https://www.ashrae.org/.

[8] *American National Standards Institute - ANSI Home*. URL: https://www.ansi.org/.

[9] 14:00-17:00. *ISO 16484-5:2017*. fr. URL: https://www.iso.org/cms/render/live/fr/sites/isoorg/contents/data/standard/07/19/71935.html.

[10] Conférence de coordination des services de la construction et des immeubles des maîtres d'ouvrage publics KBOB. *Using BACnet Recommendation*. Anglais. Sept. 2017. URL: https://www.kbob.admin.ch/dam/kbob/de/dokumente/KBOB/KBOB_Empfehlung_BACnet_e_2017_v1.1.pdf.download.pdf/KBOB_Empfehlung_BACnet_e_2017_v1.1.pdf.

[11] *What are BACnet Ethernet, IP, and MS/TP? | Optigo Networks*. URL: https://optigo.net/blog/what-are-bacnet-ethernet-ip-and-mstp.

[12] *Modbus*. en. Page Version ID: 1092284871. June 2022. URL: https://en.wikipedia.org/w/index.php?title=Modbus&oldid=1092284871.

[13] *Systèmes de chauffage | Viessmann Suisse*. Feb. 2022. URL: https://www.viessmann.ch/fr.html.

[14] *Vitogate 300 BN/MB - Solutions de Prescription*. URL: https://solutionsdeprescription.viessmann-france.com/produits/vitogate-300-bn-mb/.

[15] *node-red-contrib-modbus*. en. URL: http://flows.nodered.org/node/node-red-contrib-modbus.

[16] *node-red-contrib-sparkplug*. en. URL: http://flows.nodered.org/node/node-red-contrib-sparkplug.

[17] steve. *How to Use Node-Red with Modbus*. en-US. Nov. 2020. URL: https://stevesnoderedguide.com/node-red-modbus.

[18] *Fabricant autrichien de système de chauffage à biomasse - Fröling*. URL: https://www.froeling.com/fr.html.

[19] *How to Fix a 169 IP Address Error*. en. Section: Lifewire. URL: https://www.lifewire.com/how-to-fix-a-169-ip-address-error-4582802.

[20] Geneva swissinfo.ch. *How to get heat from the bottom of a lake*. en. URL: https://www.swissinfo.ch/eng/sci-tech/renewable-energy_how-to-get-heat-from-the-bottom-of-a-lake/41700430.

[21] *Heating and cooling sytem*. en-GB. URL: https://www.epfl.ch/about/sustainability/energy/central-heating-unit/.

[22] *CADIOM - Chauffage à distance par l'incinération des ordures ménagères*. URL: https://www.cadiom.ch/.

[23] *Unité modulaire de gestion locale, modu524/525*. fr-FR. URL: https://www.sauter-controls.com/fr/produits/unite-modulaire-de-gestion-locale-modu524-525/.

[24] *node-red-contrib-bacnet*. en. URL: http://flows.nodered.org/node/node-red-contrib-bacnet.

[25] *Metrics (networking)*. en. Page Version ID: 1061777007. Dec. 2021. URL: https://en.wikipedia.org/w/index.php?title=Metrics_(networking)&oldid=1061777007.

[26] *Business VPN | Next-Gen VPN*. en. URL: https://openvpn.net/.