HAMK
Häme University
of Applied Sciences

Optimising the Energy Efficiency of a Building Using Smart Control Systems

Designing a Customised Smart Heating System with EnOcean Hardware for Hydronic Heating Systems to Conserve Energy.

Bachelor's thesis
Degree Programme in Electrical and Automation Engineering
Autumn 2023
Animesh Joshi

HAMK
Häme University
of Applied Sciences

Electrical and Automation Engineering Abstract
Author Animesh Joshi Year 2023
Subject Optimising the Energy Efficiency of a Building Using Smart Control Systems: Designing a Customised Smart Heating Systems with EnOcean Hardware for Hydronic Heating Systems to Conserve Energy.

Supervisor Juhani Henttonen

This thesis, commissioned by the HAMK Smart Research Unit, aims at exploring solutions to optimising the energy efficiency of a building by using a customised IoT based system to control the thermostats that are installed on hydronic radiators.

This thesis discusses the features of building automation, different types of control systems, the EnOcean technology and its communication protocols. Further it details what PLCs are and informs the reader about how PLCs and EnOcean devices can be connected to aid in bidirectional communication between the actuator and the controller.

The implementation part details how the system was designed and commissioned. The aim was to create a system that integrates EnOcean temperature sensors and HORA Smart Drive MX smart thermostats to a PLC based controller. Further, a functional description of the system is provided for better clarity about the intended functionality. The implementation part then details how the connections were made, specifications of the hardware used and the control logic.

The project was successfully implemented and is running in Room A154 of the Häme University of Applies Sciences' (HAMK) Valkeakoski campus. After an analysis it was concluded that the system can save up to 70% of the energy consumed by the radiators on weekdays and 90% during the weekends.

Keywords Energy Conservation, smart home automation, EnOcean, PLC
Pages 45 pages and appendices 6 pages

Content

List of figures

List of tables

List of equations

Appendices

1 Introduction

Buildings account for one of the largest consumers of energy globally. As global challenges such as climate change and energy scarcity are rapidly gaining relevance, as is the need to come up with grassroot level solutions to solve these challenges, one of them being the operation of buildings.

Smart Control systems are being incorporated into buildings at a rapid pace. As the need for energy efficiency is growing and so is the need for all processes to be more efficient. This thesis targets the heating system of a room and explores how energy can be conserved using a smart control system on hydronic radiators that are already installed in a building.

By leveraging the power of Internet of Things (IoT) and Artificial Intelligence, smart controls prove to be an apt solution which make a difference at a micro level in order to solve the challenges of climate change and non-efficient energy usage.

Enhancing energy efficiency reduces environmental harm while simultaneously cutting down on operational costs. It promotes more comfortable and healthy indoor environments, lowers carbon footprints, and enhances occupant well-being in support of global sustainability goals. Further, building automation has become of utmost importance as it not only helps with the energy-efficiency of workplaces, homes and public spaces but also because it adds to the convenience factor of a building and can play a big role in ensuring a healthy and appropriate environment for humans to be in.

This thesis explores the role of smart controls in buildings and more specifically what steps can be taken to optimise the energy efficiency of a building with smart control systems already installed. The thesis will delve into the functioning of a hydronic heating system along with the implementation of a control system in order to come up with suggestions to ensure that buildings with the said infrastructure are operating at the most energy efficient level possible.

The room A154 in the Häme University of Applied Sciences' (HAMK) Valkeakoski campus will be used as a case study where experiments will be conducted. It is based on these experiments that the thesis will conclude on what can be improved on and what kind of an impact it could have on the energy efficiency of the room and in turn, the building.

The thesis will be divided into four parts, theory, implementation, results and conclusion. The four parts will answer the research questions that have been mentioned below.

The theory part will provide context for the various terminologies and background knowledge for the thesis, and this will enable for better understanding of chapters that follow. The theory part details smart controls, control systems, usage of data, what EnOcean is and its communication protocol, how control systems can affect the energy efficiency and what Programmable Logic Controllers (PLC) are and specific background knowledge about the implementation.

In the Implementation part, a system will be implemented where array of temperature sensors along with smart thermostats, which can be controlled using a Beckhoff PLC, will be used. The aim of this part will be to successfully establish communication between the devices and the PLC as well as to create a control system that can control the actuators based on various metrics, making it dynamic. Metrics such as the timeframes when people are present in the room, radiator temperatures and motion detection will be used to implement the logic. The results will be further analysed to verify if the findings are plausible or not.

Further, in the results part, evidence of the system working will be presented along with an analysis on what the magnitude of energy savings is vis-à-vis the current system in place. Finally, in the conclusion part will explain the challenges faced and the author's views on the system that has been developed, challenges faced while implementing it and on the ease of implementing the system.

The research questions that this thesis looks to delve into are -

1. How can digital thermostats be used for heating and what kind of difference can they create at a room level?
2. What is the magnitude of the energy savings that can be achieved in the target space?

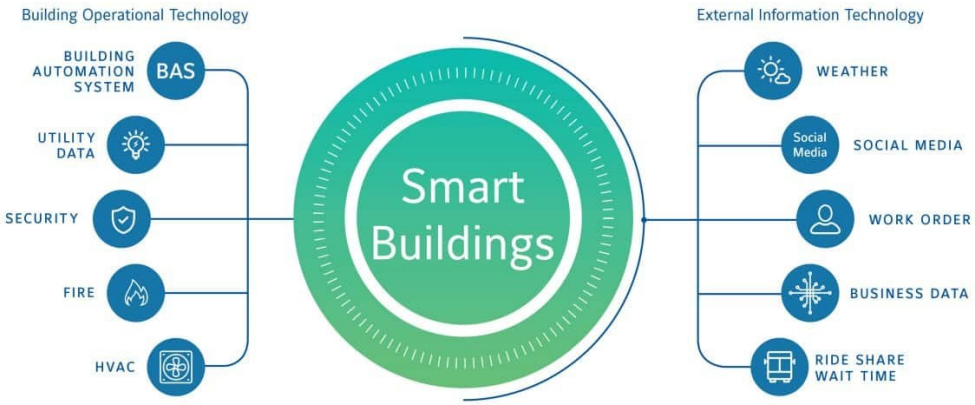
2 Building automation technologies

This chapter will provide with all the necessary information, terminologies and context that will be required to be known in subsequent chapters.

2.1 Smart controls

Smart controls refer to the use of IoT systems in buildings that are used to monitor and control building processes. Smart controls use a computerized, intelligent network of electronic devices designed to monitor and control the mechanical, electrical, lighting and other systems in a building (Sinopoli, 2016). Figure 1 presents different types of smart controls that can be used in buildings.

Figure 1. Types of smart controls in buildings (Johnsoncontrols, n.d.).



2.1.1 HVAC

HVAC, short for Heating, Ventilation and Air Conditioning is an integral building process that is often utilises smart controls. It is the system responsible for the heating and cooling of an indoor space and influences the air quality in the space.

2.1.2 Lighting

Lighting of a space can be controlled using smart controls. Often, motion sensors are coupled with lights to ensure that lights are turned on when they are needed and off when they are not. This helps in saving on energy used by the bulbs. Lighting also includes bulbs that dim as this has also proven to save energy.

2.1.3 Security systems

Security systems are also an integral application of smart control systems in buildings. Security systems can be integrated with IoT enabled hardware to enhance the monitoring.

2.2 Control Systems and Types

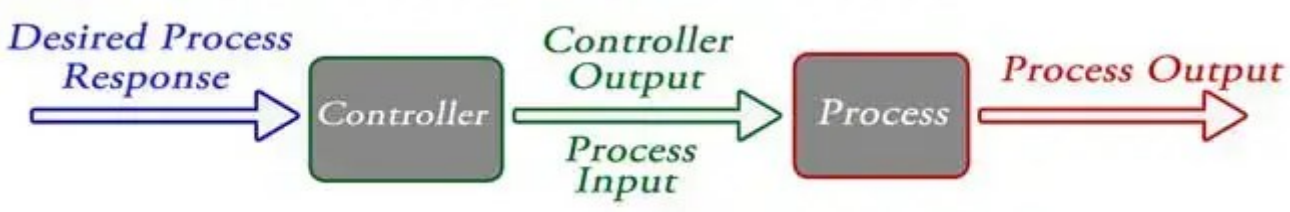
A control system is a system which uses sensor feedback from the real environment in order to alter key variables of the process.

It is system of devices that manages, commands, directs, or regulates the behaviour of other devices or systems to achieve a desired result. A control system achieves this through control loops, which are a process designed to maintain a process variable at a desired set point (Electrical4U, 2020 -a).

2.2.1 Open and closed loop control system

According to Electrical4U (2020 -a), an open loop control system is defined as a system where the control action is totally independent of the output of the system. This kind of system does not take into account the real output variable in order to control the input variable. The input variable could oftentimes just be constant and lead to a constant output variable value. Figure 2 visually represents the functioning of an open loop control system.

Figure 2. Open loop control system (Electrical4U, 2020 -a).

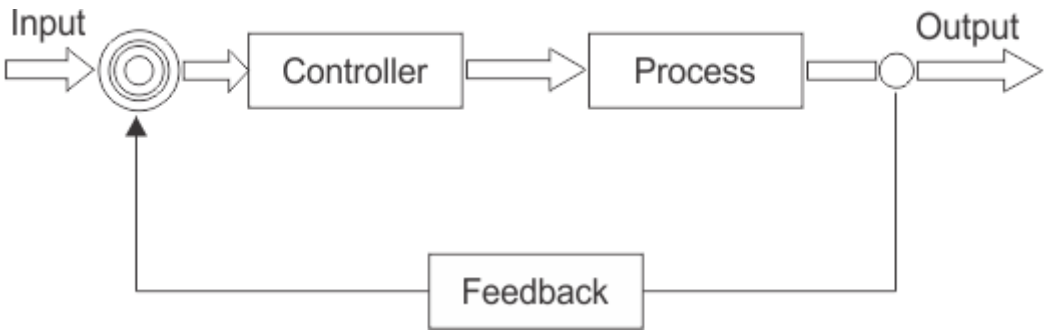


A closed loop control system is one in which the output has an effect on the input quantity in such a manner such that the input will adjust itself based on the output generated (Electrical4U, 2020 -a).

In the case of building automation, a closed loop control system is the best fit as live feedback can be gathered from the building using which the processes can be adjusted as

required. This can be done automatically by using a controller, or manually for example using a thermostat. Figure 3 shows the functioning of a closed loop control system.

Figure 3. Closed loop control system (Electrical4U, 2020 -a).



2.2.2 Linear and non-linear control systems

Linear control systems are part of control theory and are defined as the type of control systems following the principles of homogeneity and additivity (Electrical4U, 2020 -b). Collectively, they are known as the principle of Superposition.

Homogeneity is the property where, if, the input variable is multiplied by a constant, the output variable will also be multiplied by the same constant. Equation 1 depicts this property in a mathematical form.

Equation 1. Equation representing homogeneity.

$$y = s[ku] = ks[u]$$

Additivity is the property which states that the total value of the sum will be the same as the sum of those values individually. Equation 2 depicts additivity in a mathematical form.

Equation 2. Equation representing additivity.

$$S[U_1 + U_2] = S[U_1] + S[U_2]$$

A non-linear control system is defined as a control system that does not follow the principle of superposition (Electrical4U, 2020 -b). What this implies is that the input and the output are not proportional linearly, they can exhibit oscillations or curves as their relationship. Linear

control systems can also exhibit similar oscillations however for non-linear control systems, linear control algorithms cannot give a performance that is good enough.

2.2.3 Continuous and discrete control systems

Control systems are further subdivided into two categories called continuous and discrete control systems based on their application.

A continuous control system is one in which the feedback loop tried to maintain the process variable at a setpoint by subtracting the latest process variable's measurement from the setpoint to generate an error signal, it is then, after taking into account the magnitude and duration of the error signal that the value of the controlled variable is determined which then dictates the corrective action undertaken by the controller (VanDoren, 2006). For example, in the case of a thermostat, the value of the room's temperature is considered in order to determine if the heating of the room needs to be increased or decreased.

A discrete control system is one where the controller's output is determined by a triggering event, following which a measure-decide-actuate sequence is executed which results in a modification of the output variable (VanDoren, 2006). An example of this would be a motion-based control system for lighting. When motion is detected, the variable controlling the state of the light is changed resulting in the light being turned on and subsequently when no motion is detected the controlled variable is switched back again resulting in the light being turned off.

2.3 Usage of Data in Control Systems

Data is an important part of control systems. It is by using sensor data that closed loop control systems can influence the input and subsequently the output value. Data is gathered in processes using sensors. A process can have an array of sensors giving feedback to the controller in order to adjust the inputs so that the final product is satisfactory.

2.3.1 Data Collection

Data for control systems acquired by using sensors. It is possible to tune controllers using sensor signals however data collection is key in order to analyse the long-term performance of the control system. In building control systems, the system typically uses data acquisition

equipment, sensors and computerized control systems to gather and analyse data. (Abachy, n.d.)

Data collection is of utmost importance as without it, analysis is not possible and hence adjustments cannot be made to the control system in order to optimize the systems performance.

2.3.2 Data Analysis tools

Some tools that can be used for data analysis of buildings are –

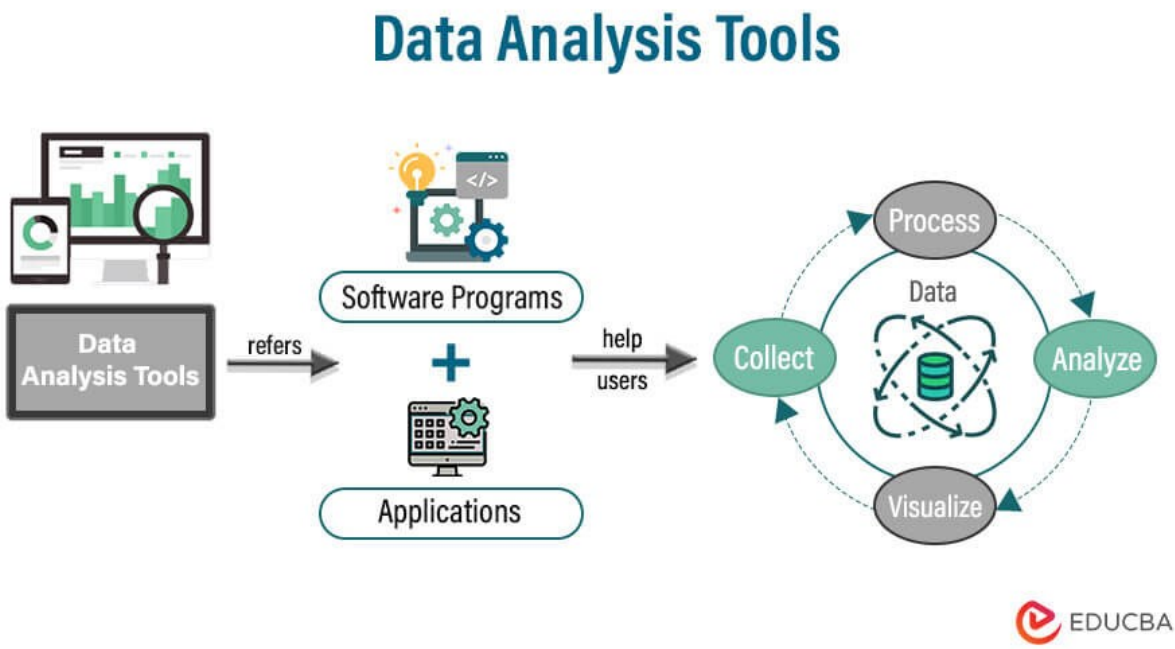
1. Specialised platforms for building automation
    a. Siemens Xcelerator
    b. BuildingsIOT's onPoint
    c. Talotohtori 2.0 by Enermix

2. Non-building automation specific tools
    a. Pandas (Python Library)
    b. Grafana
    c. InfluxDB
    d. AWS Cloud Analytics
    e. PowerBI

Using these tools, building data can be accurately analysed. The building automation specific platforms provide a fairly code-less user experience whereas the non-building automation

specific tools might require coding knowledge in order to retrieve, store and visualize data. Figure 4 shows a step-by-step process of how data analysis tools function.

Figure 4. Functions of Data Analysis tools (Tawde, 2023).



2.4 Control systems and their influence on energy consumption

Control systems play a prominent role in the energy consumption of a building. It is through the feedback loop that the system is informed about when it needs to increase its output and when not. For example, if a building is equipped with smart controls, the lighting of the building can be scheduled to operate during the times when people are present in the building. This alleviates the risk of human error where people might forget to turn off the lights when they leave leading to the lights being on during the night, which is when the building is not being utilized. Another way to control the lighting is by using motion sensors which turn the lights on for a stipulated amount of time when motion is detected and turn them off when there is no motion detected.

Well-designed and properly implemented building automation and control systems (BACS) can contribute to a reduction of the energy consumption in buildings, while increasing comfort and convenience for the occupants (Van Thillo et al., 2022).

Furthermore, Less et al. (2019) concluded that the best controls averaged about one-third of ventilation-related energy savings that increased to about 48–55% when weighted. Annual average savings were about 650–700 kWh/year for the best controllers. Also, using smart controls with lighting has shown a reduction in the average energy consumption of electric lighting of up to 23% (Campano, 2022). This reiterates that smart controls are a very effective solution to ensure the energy efficiency of a building.

Figure 5 shows the magnitude of energy savings that are achieved by using smart controls in various building processes.

Figure 5. Energy savings in processes due to smart controls (Smart Touch, 2022).

2.5 Process Control Systems and Models in Building Automation

BPCS stands for basic process control system. This system handles the process controls and monitoring for the process. It takes the input from process sensors processes it according to control and monitoring strategy fixed at the design stage to produce output for output devices/final control element, so that the process behaves according to design (Basu, 2016).

Process control systems in building are mainly appropriate for the HVAC system. Kusiac (2020) claims that by using a multiple-linear perceptron (MLP), a neural network model, demonstrated that the total energy consumed by the HVAC system is reduced by 7%.

Process control can be further enhanced in building automation by implementing and creating new data-based models which can find relationships between variables such as energy consumption, supply of air, air temperature, pressure and room air conditions.

3 EnOcean

For the implementation of this thesis, an EnOcean ecosystem is used. This means, that all the devices used are compatible with the EnOcean system. EnOcean is a wireless technology that is widely used in many IoT applications, specifically, in smart homes. It is a wireless energy efficient technology which is powered by the usage of mechanical motion and other sources of energy from the surroundings such as temperature differences and lighting. This is done by converting these fluctuations into electrical energy which operates the hardware unit. What this implies is that EnOcean hardware seldom requires power sources such as batteries or a connection to an electrical point in order for it to operate.

3.1 EnOcean’s Wireless communication protocol

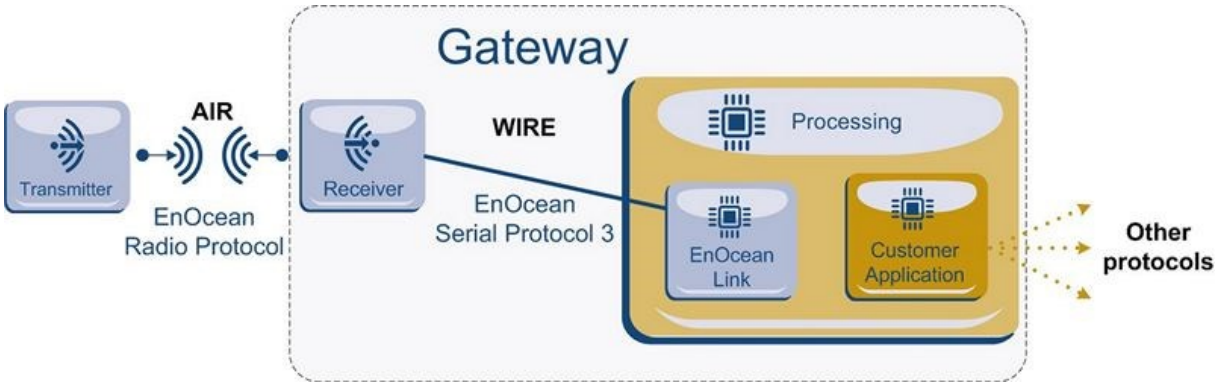
The EnOcean technology, being a wireless communication protocol that allows various IoT-friendly devices to communicate amongst themselves has a range of 300 metres in open areas and 10-30 metres in buildings (Devi, 2023).

EnOcean operates on different frequencies in different regions, in Europe, it operates at the 868 MHz band. EnOcean uses security measures such as rolling-code mechanisms, encryption and authentication procedures that ensure the protection of data that is being transmitted between devices. EnOcean devices also employ a Listen before talk (LBT) mechanism that ensures that there is no collision in wireless communications. LBT is a method in which the device listens to the channel in order to detect ongoing communication, if the channel is found to be clear, the device transmits the information and if not, it attempts again (Devi, 2023).

The data transmission rate of an EnOcean Communication is generally around 125 kbps (Devi, 2023). Cyclic Redundancy Checks (CRC) are used in order to check the data that is being transmitted; this is done in order to ensure that the data has been transferred correctly. EnOcean also uses acknowledgement mechanisms where, after a device sends a command, the receiver can send back an acknowledgement (Devi, 2023).

Gateways can be used to convert the EnOcean communication to internet protocols. This feature helps greatly in integrating EnOcean with IoT solutions across the board. These gateways also enable EnOcean devices to communicate using technologies such as Bluetooth and WiFi. Figure 6 shows how gateways convert EnOcean's radio protocol to other protocols.

Figure 6. EnOcean's conversion to other protocols (EnOcean, 2013).



3.2 Data rate and Transmission

EnOcean devices use the EnOcean Serial Protocol 3 (ESP3). The ESP3 protocol specifies how the module communicates with its host Microcontroller Unit (MCU).

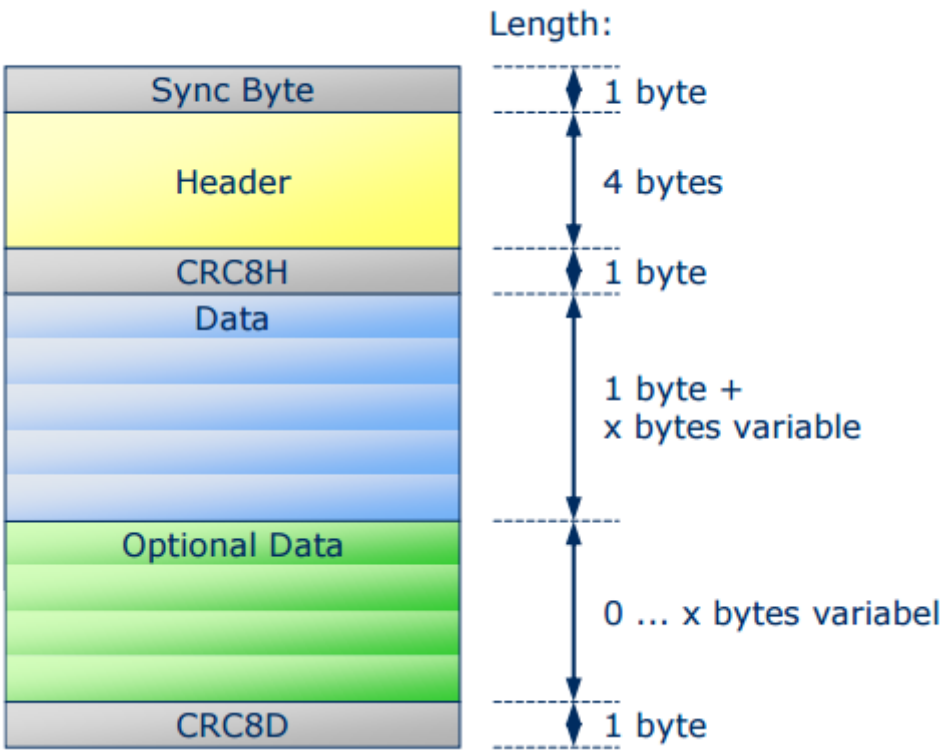
3.2.1 Data Packets

EnOcean uses data packets in order to communicate between the devices and the MCU. A packet is transferred using the Universal Asynchronous Receiver/Transmitter (UART) technology.

An EnOcean Data Packet consists of a header, data and optional data. It also contains a synchronization byte, a CRC byte for the header and CRC byte for the data (Mikroe, 2016). There are two types of data packets -

The Transmitted Packet– This type of packet is shown in figure 7, the header field of this kind of packet consists of data length, optional data length and packet type this is usually 4 bytes long. The data field contains the data and hence has a variable length; the same applies for the optional data field. The packet type field contains the type of packet that is being received or sent (Mikroe, 2016)

Figure 7. A visual representation of an EnOcean data packet (Mikroe, 2016).



As seen in figure 7. The packet is of minimum eight bytes and increases based on the 'x bytes variable' in the data byte and optional data byte changing. Figure 8 describes each byte of the data packet.

Figure 8. Description of an EnOcean data packet (Mikroe, 2016).

Group	Offset	Size	Field	Value hex	Description
-	0	1	Sync. Byte	0x55	Serial synchronization byte; always set to 0x55
Header	1	2	Data Length	0xnnnn	Specifies how many bytes in DATA must be interpreted
	3	1	Optional Length	0xnn	Specifies how many bytes in OPTIONAL_DATA must be interpreted
	4	1	Packet Type	0xnn	Specifies the packet type of DATA, respectively OPTIONAL_DATA
-	5	1	CRC8H	0xnn	CRC8 Header byte; calculated checksum for bytes: DATA_LENGTH, OPTIONAL_LENGTH and TYPE
Data	6	x	... ...	... ...	Contains the actual data payload with topics: - RawData (e.g. 1:1 radio telegram) - Function codes + optional parameters - Return codes + optional parameters - Event codes x = variable length of DATA / byte number
Optional Data	6+x	y	... ...	... ...	Contains additional data that extends the field DATA; y = variable length of OPTIONAL_DATA
-	6+x+y	1	CRC8D	0xnn	CRC8 Data byte; calculated checksum for whole byte groups: DATA and OPTIONAL_DATA

The Response Packet – This is the second type of packet which is shown in Figure 9. This packet has a structure similar to the transmitted packet but instead of the data field, it contains the return code in addition there is no field for optional data and instead all data is transmitted using the response data field. Figure 10 shows a list of return codes and their descriptions.

Figure 9. Structure of a response data packet (Mikroe, 2016).

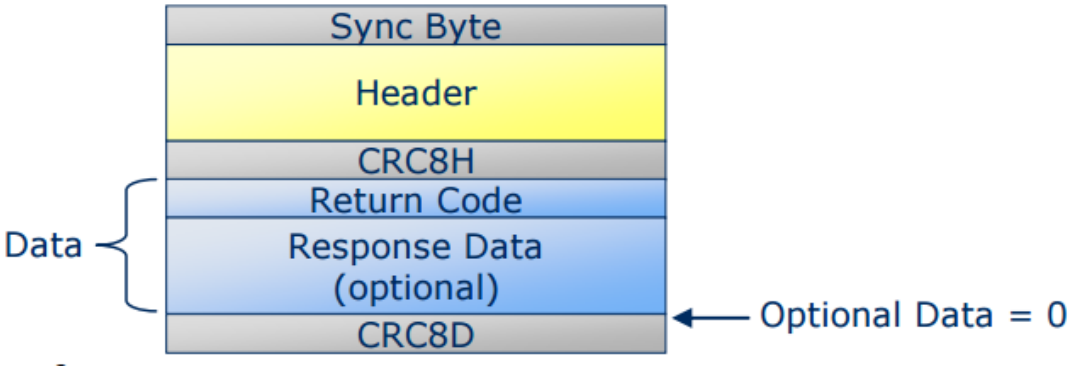


Figure 10. List of return codes (Mikroe, 2016).

Code	Name	Description
00	RET_OK	OK ... command is understood and triggered
01	RET_ERROR	There is an error occurred
02	RET_NOT_SUPPORTED	The functionality is not supported by that implementation
03	RET_WRONG_PARAM	There was a wrong parameter in the command
04	RET_OPERATION_DENIED	Example: memory access denied (code-protected)
05	RET_LOCK_SET	Duty cycle lock
06	RET_BUFFER_TO_SMALL	The internal ESP3 buffer of the device is too small, to handle this telegram
07	RET_NO_FREE_BUFFER	Currently all internal buffers are used.
> 128	---	Return codes greater than 0x80 are used for commands with special return information, not commonly useable.

4 Programmable Logic Controllers (PLC)

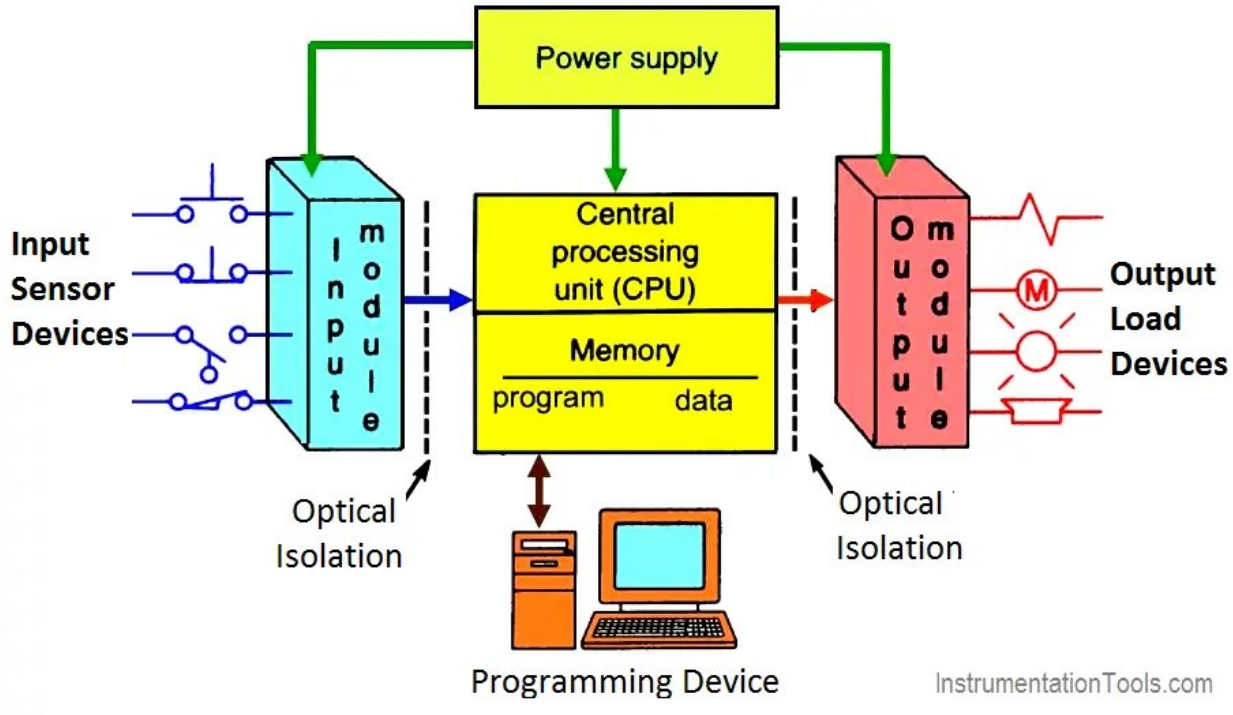
A PLC is a computer used for automation. It consists of Input and Output modules (I/Os) which allow it to receive feedback and actuate devices. It is most commonly used in industrial automation but due to its computer-like nature, it can be used for a host of other applications.

A PLC needs to be programmed to trigger outputs based on connected inputs. It can also be used for monitoring as well as error handling of the process it is controlling. A PLC traditionally consists of six essential systems presented below.

1. Central Processing Unit (CPU) is the unit where the commands that have been given to the PLC are executed. It is also popularly called the “brain” of the PLC.
2. Racks are used to hold into place the various components such as the CPU, I/O’s and the power supply.
3. The Input Assembly is used to receive signals from input devices such as sensors and also to show the status of the inputs.
4. The output assembly controls the output in accordance with the commands that have been provided to the PLC after it has been processed by the CPU. Outputs can be Analog or Digital depending on what is being controlled.
5. PLCs require a power supply in order for the electronic components to function. The power supply provides the system with 24VDC or 48VDC.
6. A programming unit is the device that is used to load the program onto the PLC. This could be an external PC or could be a software that is loaded onto the PLC itself. (Plctechnician, 2018)

Figure 11 shows various components of a PLC and the flow in which PLCs function from getting input values, to processing them and then finally computing output values.

Figure 11. Components of a PLC (instrumentationtools, n.d.).



4.1 Programming for PLCs

PLCs require commands in order to know how to relate inputs to outputs. The following languages are used for PLC programming.

1. Structures Text (ST) is a language that has close relations to C. The code is executed sequentially, which means that commands are executed line-by-line. Due to its close relation to C the syntax is mostly similar, and it can be used by people who have basic programming knowledge in traditional programming languages such as Java, C++ and C.
2. Sequential Function Chart (SFC) is more graphical than ST as in this the programmer creates a chart. The chart consists of blocks and transitions. Each block represents a step and before moving onto the next step, a condition needs to be fulfilled by utilising the transition. The blocks can have ST integrated code which allows for complex processes to be programmed with ease and enable an advanced logic flow.
3. Ladder Logic Diagram (LD) is a visual programming language used in industrial automation. It illustrates control systems with symbols like as connections and coils arranged in rungs. In contrast to contacts, which indicate input conditions, coils

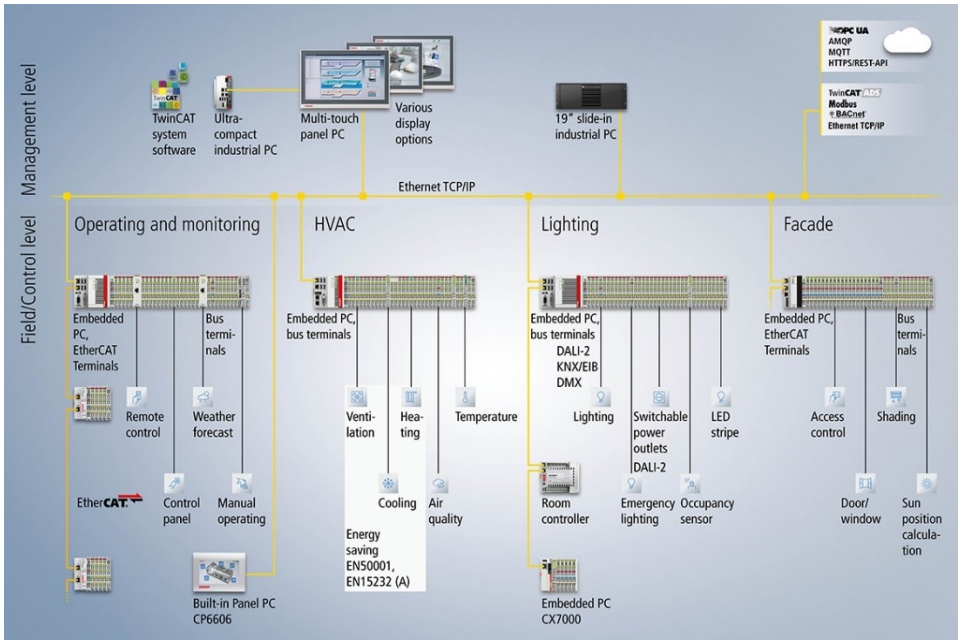
represent outputs like motors or relays. Rungs connect and specify the logical connections between these components. Power rails on each side stand in for the voltage supply and ground. Timer and counter functions enable complex timing and sequencing operations. Because ladder logic is widely used in the manufacturing industry and simplifies the design of control logic for PLCs it is a crucial tool for optimizing industrial processes and equipment control.

4. Function Block Diagram (FBD) is a graphical programming language used by PLCs for industrial automation. To visualize control logic, it depicts operations and functions as a network of interconnected function blocks. Function blocks provide for the encapsulation of complex functions, facilitating the reuse and modularization of code. Signal routes are made simple to grasp by connecting blocks with lines that depict data flow. FBD simplifies the design of complex control systems, enabling efficient automation and process control in industries where reliability and modularity are critical, such as manufacturing.

4.2 Beckhoff Automation System

Beckhoff Automation provides high performance technology for automation. It is one of the leading companies when it comes to automation which is due to its reliability and innovation in the field of automation. They mainly manufacture embedded PCs, Input/Output cards in addition to software and technologies for motor control.

Figure 12. Beckhoff's components for building automation (Beckhoff, n.d.).



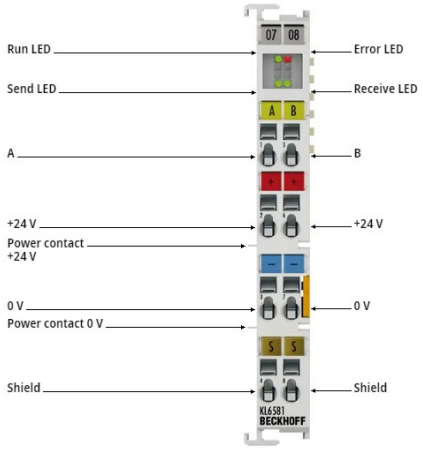
A Beckhoff Automation system refers to an automation system that uses Beckhoff hardware and software. A Beckhoff system allows for integration with many technologies and is highly customisable. Their software, TwinCat is widely used due to its ease of use and widespread functionality. The IEC 61131-3 standard being the programming protocol of TwinCat allows for easy interoperability between automation devices of different kinds. Further, TwinCat allows for both, configuration as well as programming, making it ideal for most automation applications. Lastly, it has a vast offering of libraries which makes programming more convenient and allows for an array of functionalities to be implemented.

After some research, it was decided that a Beckhoff system was to be used for the project as the IoT integration was found to be seamless and there are specific I/O cards and libraries for EnOcean which is the technology that will be used for the IoT devices.

4.3 The KL6581 Card

The KL6581 is a card that is used to integrate EnOcean with the PLC. It is the link between the KL6583 EnOcean transmission and reception terminal and the EnOcean devices. The KL6581 can support up to 8 EnOcean Terminals. It operates on 24 VDC. Figure 13 represents the appearance and configuration of a KL6581 EnOcean card.

Figure 13. The KL6581 EnOcean Card (Beckhoff, n.d.).



The KL6581 card supports telegrams in the form of Repeated Switch Communication (RPS), 1 Byte Communication (1BS) and 4 Byte Communication (4BS). Figure 14 shows the technical specifications of the KL6581 card.

Figure 14. Technical Specifications of the KL6581 card (Beckhoff, n.d. -b).

Technical data	KL6581
Technology	EnOcean
Number of channels	1
Supported telegrams	RPS (Repeated Switch Communication), RORG: F6, ORG: 05 1BS (1 Byte Communication), RORG: D5, ORG: 06 4BS (4 Byte Communication), RORG: A5, ORG: 07
Data transfer standard	–
Connection	2 x 2-wires directly at the KL6583 EnOcean module (connection of max. 8 KL6583)
Data transfer rates	125 kbaud
Cable length	max. 500 m
Power supply	via the K-bus
Connecting cable	up to 500 m
Current consumption K-bus	typ. 60 mA
Current consumption power contacts	typ. 20 mA + load
Configuration	not required
Nominal voltage	24 V DC (-15 %/+20 %)
Special features	up to 8 KL6583 EnOcean transmitter and receiver modules
Weight	approx. 85 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Protect. rating/installation pos.	IP20/variable
Approvals/markings	CE, UL

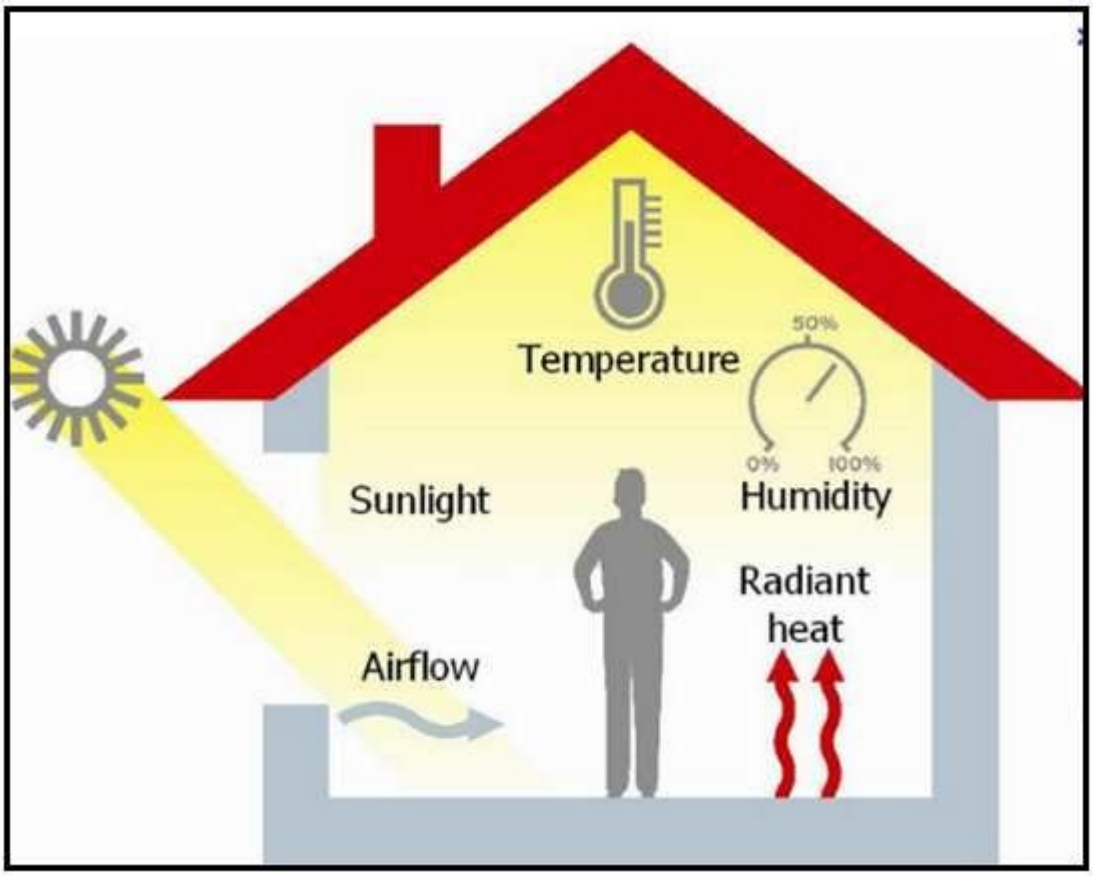
5 Evaluation of factors to consider for optimising the energy efficiency of the heating system.

The heating system of a building consists of the ventilation system, the heating elements such as radiators and the source which in the case of this thesis is the district heating. The factors that usually need be taken into account in order to optimise the heating system are:

- Outside temperature refers to the climatic conditions and the current temperature outside. It is a very important factor which can be used to determine when the heating system must be activated and what intensity it needs to have.
- Inside temperature is the temperature indoors that must be suitable for people to be comfortable and healthy in. Inside temperature is also used as a direct factor while designing and controlling a heating system.
- Flow temperature is the temperature that the boiler heats the water to. For energy efficiency purposes, this is a very important aspect as it directly correlates to how much energy is being consumed by the system.
- The return temperature is the temperature of the water after it has gone through the radiators and has expelled the heat in it. This factor is typically used by the boiler system to ascertain what the flow temperature must be set at.
- Sunlight is a factor to consider as it can alter the outside temperature as well as the temperature inside the room.
- Ventilation is used to expel air and bring in fresh air. The fresh air is usually processed in the ventilation system where it might be heated or cooled, hence ventilation is a factor to be considered too.

Figure 15 visually depicts the factors that generally affect the indoor heat in a house.

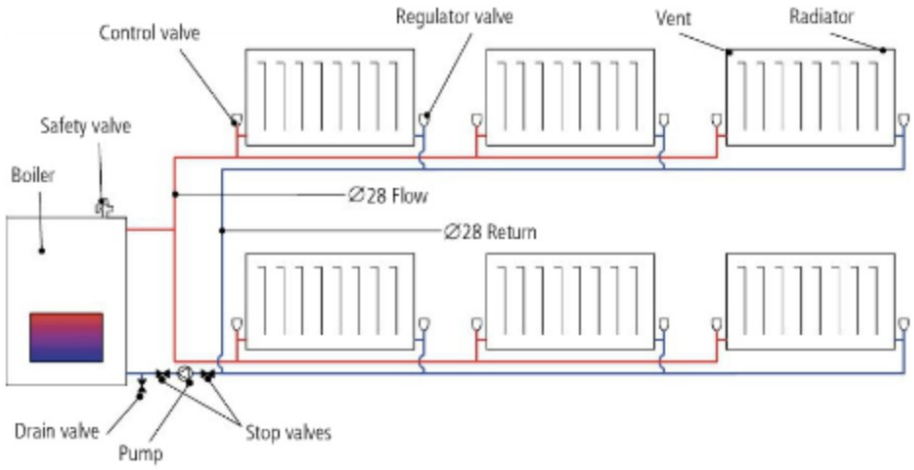
Figure 15. Factors affecting indoor heat (Alwetaishi, 2016).



5.1 The working mechanism of a hydronic heating system

Electric boilers use an electrically heated element through which the supplied water passes and is heated. This water is then, in some cases, stored in a tank and in the other cases supplied to the radiators. The boiler's energy consumption is dependent on the temperature of water which is determined by the required room temperature setpoint. The energy consumption of a boiler is directly proportional to the required temperature. Figure 16 depicts how a hydronic heating system function.

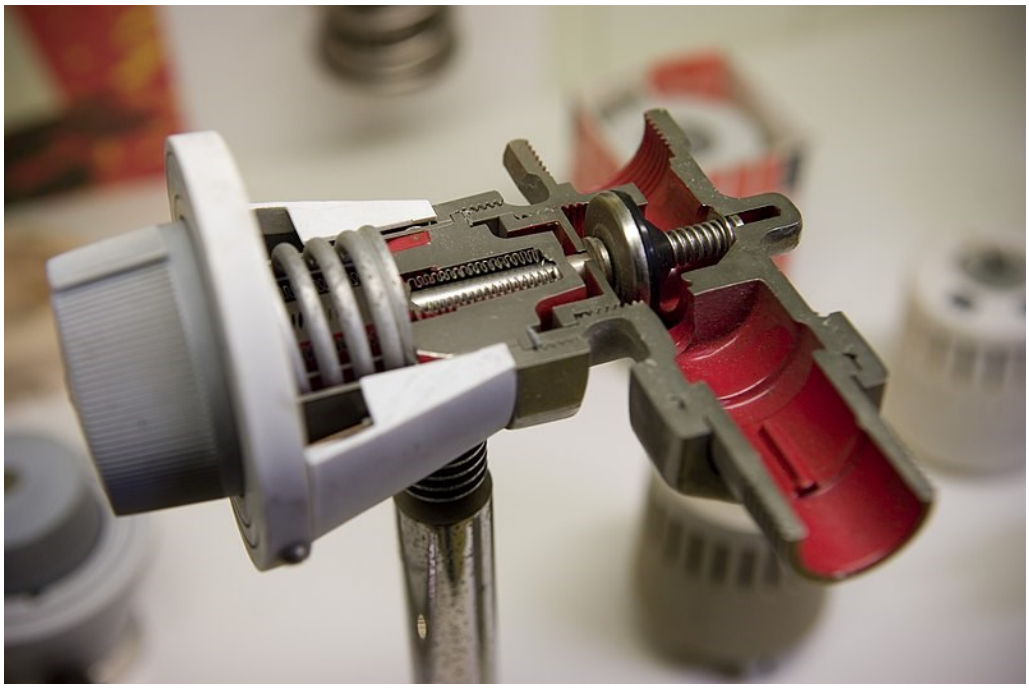
Figure 16. A hydronic heating system (Parkstone-Yorkshire, 2020).



Boilers ascertain the need to heat up water based on the setpoint for the flow temperature for the radiators. Additionally, the return temperature is directly proportional to the flow temperature, hence by decreasing the setpoint for the flow temperature the return temperature is decreased, and the energy used by the boiler is lesser. Most often flow temperature is set to a point, this causes the boiler to work even when it is not needed, for example, in the case of office buildings where there are no people during the night.

The radiators lose heat to the air when hot water is passed through them, resulting in an increase in the temperature of the surrounding area. The thermostat on the radiators utilises a Thermostatic Radiator Valve (TRV) as shown in Figure 17 which has a pin that moves inwards or outwards depending on the temperature setpoint requirements. This allows water to either flow through the radiator or causes the flow to stop, resulting in a disruption of heat supply and finally to the radiator losing all the heat contained in it.

Figure 17. A cross sectional representation of a TRV (Christensen, 2007).



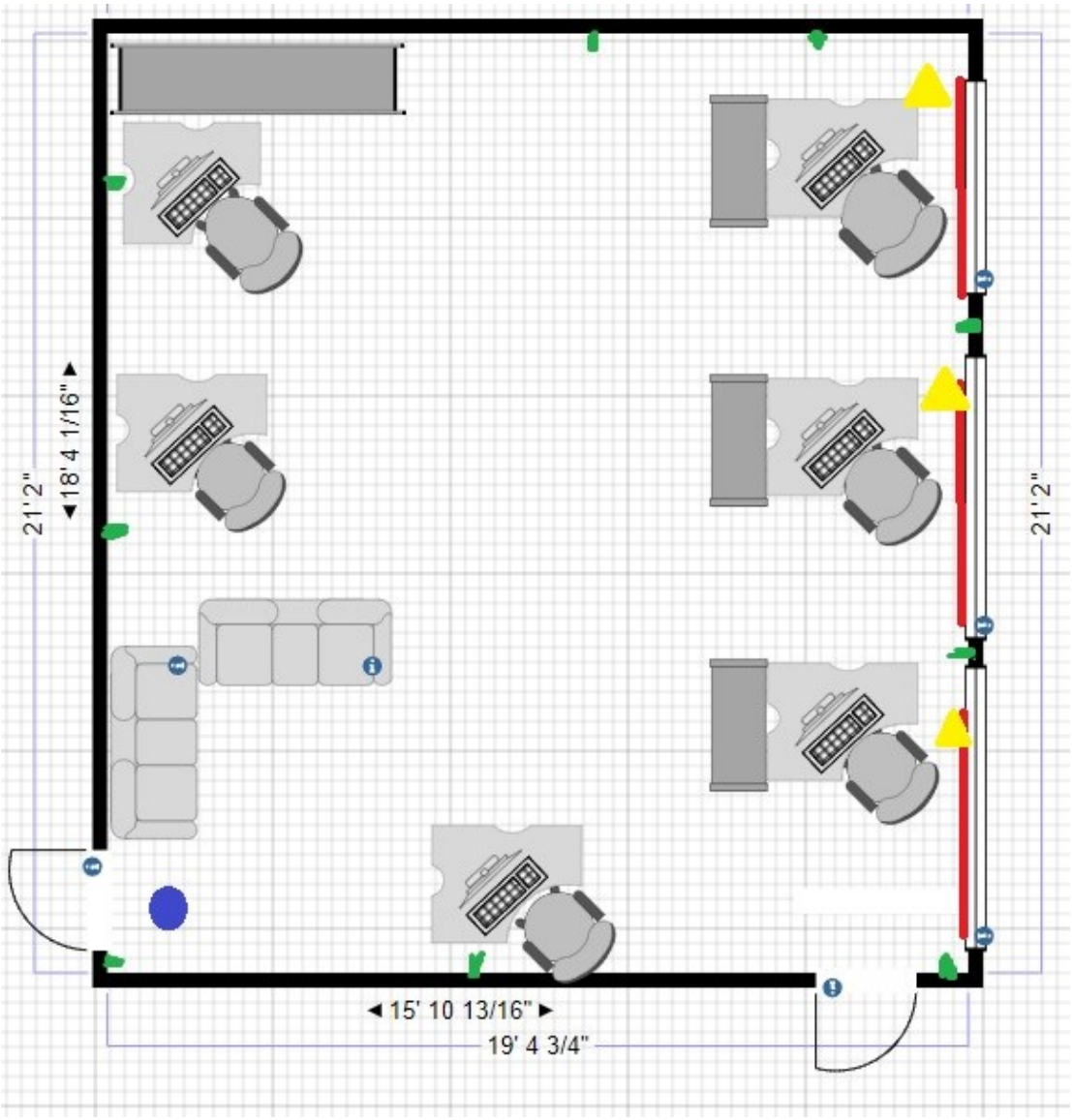
6 Implementation

The implementation of the project for this thesis was conducted in the room A154 in the Valkeakoski campus of HAMK. The room is used as a workplace for HAMK Smart and Tech employees and hence was an appropriate place to implement the solution. Figure 17 shows the floorplan of the room with the positioning of the work desks. The implementation mainly dealt with creating a system that can dynamically control the radiators installed in the aforementioned room. This was implemented by using a Beckhoff PLC, smart thermostats, an array of temperature sensors as well as presence sensors.

6.1 Case

The room A154 consists of three radiators that have been installed along the three windows shown using red lines in figure 18. These radiators are cast iron and use hot water that flows through them to facilitate heat exchange. They are connected to the district heating system and the hot water is provided directly from there. In figure 18, the green dots represent the sensor positions, the yellow triangles represent the smart thermostats, and the blue circle is the position of the motion detector in the room.

Figure 18. Floor plan and device positions in the target room (not to scale).



Many houses use the same technology, however, instead of being connected to the district heating system, they have their own boilers on-site which run on electricity. The control system that is implemented uses a Night and morning boost system which means that during the night the valves in the radiators are closed resulting in a lower setpoint for the flow temperature. During the mornings when the temperature of the room is required to be at a standardised set point, the flow temperature set-point is set to the maximum in order to heat up the room as quickly as possible. After this, the set point is set to a user defined value for the entire day. Motion sensors are connected to the system too. This is done so that, in case, during the times when the radiator flow temperature is at the minimum causing the room temperature to be low, the heating system can be turned on based on the presence of a person, ensuring that the room temperature is appropriate for human usage.

6.2 Hardware

The hardware used is divided into two parts, the IoT infrastructure and the embedded PC system. The IoT infrastructure consists of the EnOcean compatible sensors and actuators, and the embedded PC part consists of the PLC, the cards used and reception technology as detailed in the subsequent sections. The IoT infrastructure communicates with the PLC, also known as Embedded PC System, using a profile. After this, it is processed, and the logic is applied for the system to function. In order to enable the readers to understand the technology better, A brief description of the HORA smart Drive MX profile is also provided. Tables 1 and 2 specify the hardware used, its specifications and its functionality.

6.2.1 IoT Devices

Table 1. List of IoT devices used.

Device	Name	Quantity	Telegram Type	EnOcean Telegram Profile	Function
Thermostat	HORA Smart Drive MX	3	4BS	A5-20-04	Thermostat to control flow temperature to the radiator
Temperature Sensor	STM-330	9	4BS	A5-02-05	Temperature Sensors with range 0-40 °C
Motion Sensor	SR-MDS Solar	1	4BS	A5-08-01	Motion sensor

6.2.2 HORA Smart Drive MX

A HORA Smart Drive MX is used as the thermostat. For this implementation, three of these devices were used and attached to the three radiators in the room. A control logic that took into account parameters such as the time, room temperature and the presence of people was established. All these three factors played a part in actuating the device and subsequentially, the implementation of the proposed system. Figure 19 displays the HORA Smart Drive MX.

Figure 19. The HORA Smart Drive MX smart thermostat (EnOcean Alliance, n.d.).

The HORA Smart Drive MX is a thermostat that supports bi-directional communication. It uses the A5-20-04 EnOcean telegram profile. Being bi-directional, it has two telegram directions. Direction 1, which is for the transmission from the device to the controller, and direction 2, which is for transmission from the controller to the device.

Below, the byte wise data that is transmitted in direction 1 is explained:

1. Current Position (CP) is the actual position of the valve; it has a range between 0 and 100. Bitrange is DB3.7 to DB3.0

2. Temperature Set-point or Feed Temperature (FTS) is byte whose parameter is shared. This means that it either transmits the current temperature of the feed or the

temperature that has been set by the user. The information that is to be sent is defined by the TS bit. Bitrange is DB2.7 to DB2.0

3. Room Temperature (TMP) or Failure Code (FC) is another byte that is shared. It transmits either Room Temperature or the Failure Code. The default value is the Room Temperature; however, the FL bit determines if a Failure code is to be sent. Bitrange is DB1.7 to DB 1.0

4. The Measurement Status (MST) byte permits the deactivation of the temperature readings. This could be done to conserve energy during times when the radiator is not being used. The bit that transmits MST is DB0.7

5. The Status Request (SRT) bit is used to ascertain the status of the controller. If it does not respond, then the actuator initiates its own control system based on the parameters that have been set previously. Bit DB0.6 is used for the transmission of SRT.

6. The Teach-in bit (LRNB) byte is used to establish a link between the controller and the actuator.

7. The Temperature Selection (TS) bit defines which temperature value is transmitted in FTS. The bit responsible for this is DB0.1

8. The Failure (FL) bit transfers the failure status. If the bit value is 1 then there is a failure, the details of which are transmitted in Byte 2 and if it is 0 then the Temperature measurements are transmitted in Byte 2.

The information that is transmitted from the controller to the actuator (Direction 2) is in the following form:

1. Valve Position (POS) is the set point for the position of the valve. It ranges between 0 and 100%.

2. Temperature Set Point (TSP) is the user defined set point for the temperature. It is shown on the display but has no effect on the room temperature regulation.

3. Measurement Control (MC) is used to activate or deactivate the temperature measurements in order to conserve energy when the actuator is not in use.

4. Wake-up Cycle (WUC) defines when the actuator has to wake up from deep sleep-in order to communicate with the controller.

5. Service Command (SER) is to order the controller to perform functions that will aid in the installation or service of it. The valve can be completely opened or closed using this command.

6.2.3 Controller

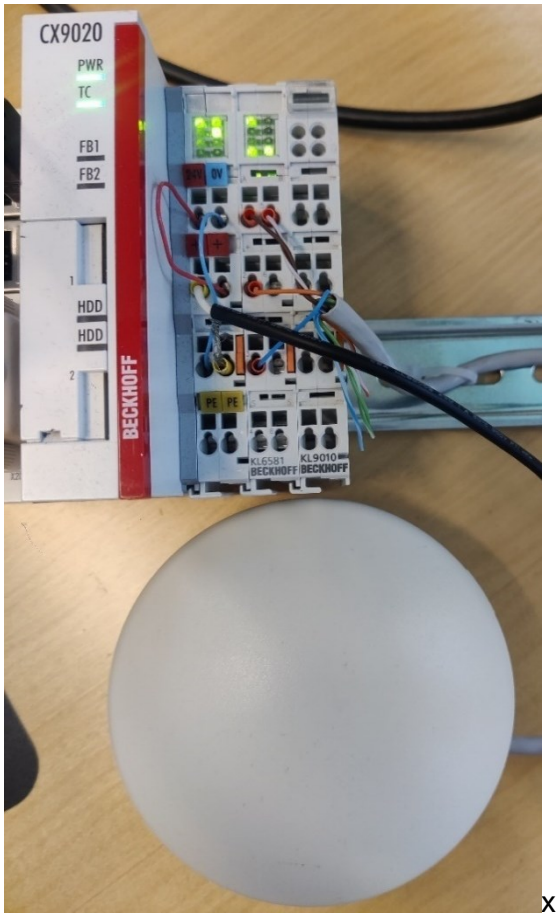
The controller used is a Beckhoff PLC. It is paired with I/O cards and an EnOcean antenna to facilitate communication between the devices and the controller. Figure 20 shows the entire controller inclusive of the PLC, I/O cards and the EnOcean antenna.

Table 2. Hardware configuration of the controller.

Name	Type	Function
Beckhoff CX9020	PLC	Embedded controller to process I/O’s
Beckhoff KL6581	I/O card	Integrate EnOcean to the PLC and process EnOcean Signals.
Beckhoff KL9010	End Terminal	To facilitate data exchange between Bus coupler and terminals

Beckhoff KL6583	EnOcean, Radio Transceiver	Antenna

Figure 20. The controller system.



x

6.3 Functional Description of the system

The system aims at actuating the HORA SmartDrive MX to change the flow temperature coming into the radiators up when required and keep it at the minimum when not required. The system logic is described below as:

1. Timer based – If the time is 7 A.M. on a working day, the actuator opens the valve completely and the thermostat set point is set to 30°C until the time that all sensors read a value that is within an error range of 5% of the user defined temperature set

point. When the values are at the setpoint value, the thermostat changes valve position to 50% to maintain the temperature. If the time is post 5 P.M. and NO motion is detected, the valves close completely and set the flow temperature set point to be 10°C.

2. Motion Based – If motion is detected during the time when the flow temperature set point is 10°C, the system set point is changed to 30°C till the room temperature reaches the set point of 22°C. After this, if no motion is detected for two hours the flow temperature is set back to 10°C.
3. On/off control – On/off control is an essential part of the system. The system uses ambient room temperature to ascertain if the radiator valve needs to be open or closed. If the temperature reaches the user defined setpoint, the smart thermostat changes valve position to 50%. Following this, if the temperature drops below the setpoint, the valves are opened to 100% till the time the ambient room temperature is at the setpoint again.

The user can change the temperature set points using the interface that is made as part of this implementation if desired. Other parameters such as time range can also be edited using the same. The interface will also display error notifications as required.

6.4 Software Implementation

The software has been developed in Beckhoff's TwinCat as the controller is a Beckhoff PLC. It uses a combination of Function Block Diagram and Structured Text for the implementation.

The software has 4 components.

1. MAIN
2. Fb_send_thermostat (Function Block)
3. EnOceanCom (PRG)
4. Visualization

The software encompasses the reception of telegrams from the devices, application of the control logic and communication with the actuators.

6.4.1 Reception of Device messages

A POU called EnOceanCom was created in order to get the information from the devices. This POU uses FBD and functions from the EnOcean library to establish a connection with the devices and the controller.

The KL6581 card's connection is established using the FB_KL6581 function which is called in MAIN as shown in figure 21. It takes inputs from the input and output addresses that have been linked to the KL6581 card along with an initialisation variable and an index variable.

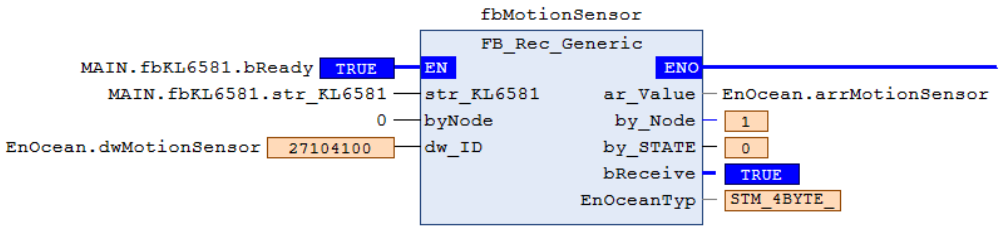
The output of this block gives error information if there is an error or makes the variable bReady true if not and provides the str_KL6581 with information in a custom data type which consists of Bytes, strings and arrays which also needs to be an input when reading and accessing data from particular devices.

Figure 21. The FB_KL6581 block.

```
1  fbKL6581(
2      bInit TRUE          := TRUE,
3      nIdx 1          := 1,
4      stKL6581_in     := stKL6581_Input,
5      stKL6581_out    := stKL6581_Output,
6      bReady          => ,
7      bBusy           => ,
8      bError          => ,
9      iErrorID        => ,
10     str_KL6581      => stKL6581);
11
12 IF (NOT fbKL6581.bReady TRUE ) THEN
13     RETURN;
14 END_IF
15
```

The sensor specific data is read using the FB_Rec_Generic function block as shown in figure 22. The outputs of the FB_KL6581 blocks such as the bComReady variable and the str_KL6581 variable are used as inputs. In addition to this the dw_ID takes the EnOcean ID of the device that is being connected to in DWORD format. The output of this block gives the state and the sensor values in array of BYTE type of size 3. This array consists of information such as the telegram type and the unscaled measuring values from the sensor. The FB_Rec_Generic block can be used for all the sensors that send telegrams regardless of transmission type.

Figure 22. The FB_Rec_Generic function block for the motion sensor.



6.4.2 Communication with the Actuator

For communication with the thermostat, the FB_Send_4BS function is used, which is part of TwinCat's EnOcean library, it is used as the thermostat supports bidirectional communication with a 4BS telegram type.

As seen in figure 23, the block has 5 inputs, the most important of them being the pt_SendData which accepts a pointer of PVOID type that points to the data structure containing the data for the actuator, in this case, a 4Byte datatype which this case is a Hexadecimal number. The nEnOceanID input which is of BYTE type and requires the EnOcean ID of the device in order to establish connection with it and send the data as required.

The block also shows if there is an error by turning the bError output to true and details the error using the iErrorID variable. When a telegram is sent, the bBusy output turns to true for a cycle and during that cycle the block does not transmit any additional information.

It was found during the implementation of this project was that the FB_Send_4BS function block should only be called once per KL6581 card. Not doing so results in a breakdown of the communication between the actuator and the controller. Hence, it was imperative to create a separate function block which could be instantiated for each actuator in order to allow the FB_Send_4BS function block to only be called once. The parameters for this instantiation are defined as input variables in the thermostat_send function block which is then instantiated three times in order to enable communication with all three actuators separately.

Figure 23.The FB_Send_4BS block for communication with the thermostat.

```
fbSend_4BS(
    bStart FALSE      := TRUE,
    by_Node 1         := byNode 1 ,
    pt_SendData 3649661488 := ADR(dwSendData 1694433544 ),
    nEnOceanID 1      := nEnoceanID 1 ,
    str_KL6581  := MAIN.stKL6581,
    bBusy       => ,
    bError      => ,
    iErrorID    => );

IF (NOT fbSend_4BS.bBusy FALSE ) THEN //TO Check if data has been sent, if it is sending bBusy will be true
    fbSend_4BS(bStart FALSE  := FALSE);
    iStep 0 :=0;
END_IF
```

6.4.3 Logic Program

The logic program deals with processing the inputs and providing outputs for the system. It has been designed keeping in mind the functionality as defined in the functional description of the implementation.

The logic of the control system is in a function block called “thermostat_send”. As demonstrated by figures 24 and 25, this function block takes the virtual EnOcean ID – “nEnoceanID”, the permanent EnOcean ID of the thermostat, start time, stop time and the temperature setpoint value as inputs.

Figure 24. List of variables for the “send_thermostat” function block.

```
FUNCTION_BLOCK thermostat_send
VAR_INPUT
    nEnoceanID: BYTE;
    dwID: DWORD;
    StartTime: DWORD;
    StopTime: DWORD;
    sPoint: REAL;
END_VAR
VAR_OUTPUT
    bStatfbsend: BOOL;
END_VAR

VAR
    fbSend_4BS  : FB_Send_4BS;
    fbRec_Generic: FB_Rec_Generic;
    fbtimestamp : FB_LocalSystemTime := (bEnable:=TRUE, dwCycle:=1);
    arrValues: ARRAY[0..3] OF BYTE;
    rTempRoom: REAL;
    dwSendData: DWORD;
    byNode: BYTE := 1;
    iStep: INT;
    bMotion: BOOL;
END_VAR
```

The thermostat_send function block first calls the function block for the timestamp after which a case structure is implemented. This case consists of two cases, case 0 and 1 which are shown in figures 26 and 28.

Figure 25. Input and output variables of the function block.

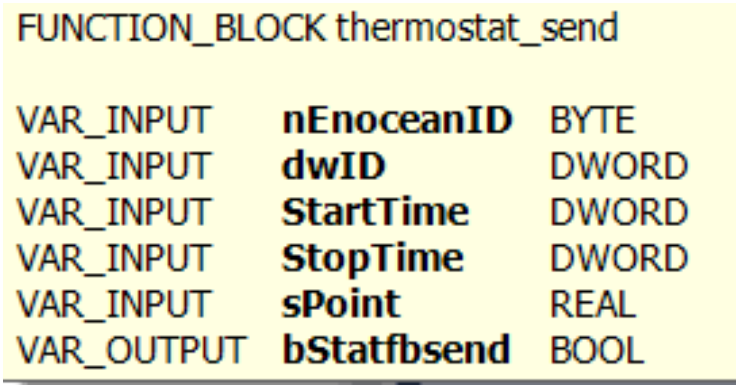


Figure 26. Case 0.

```
CASE iStep OF

    0:
        (* Calling the general reception function block. *)
        fbRec_Generic(
            str_KL6581  := MAIN.stKL6581,
            byNode      := 1,
            dw_ID       := dwID,
            ar_Value    => arrValues,
            by_Node     => ,
            by_STATE    => ,
            bReceive    => ,
            EnOceanTyp  => );

        IF(NOT fbRec_Generic.bReceive) THEN
            rTempRoom := (BYTE_TO_REAL(arrValues[1]) / 12.75) + 10; //Scaling room temperature value
            EnOcean.RoomTemp := rTempRoom+1;
            bMotion := EnOcean.arrMotionSensor[0].1; //storing Motion status

            IF (((fbtimestamp.systemTime.wHour>=(StartTime-1) AND fbtimestamp.systemTime.wHour<=StopTime)//To boost temperature up
                    AND rTempRoom<sPoint AND(fbtimestamp.systemTime.wDayOfWeek>0
                    AND fbtimestamp.systemTime.wDayOfWeek<6) OR bMotion)) THEN
                dwSendData := 16#64FF0108; //Valve position = 100% Setpoint for thermostat=30c
            ELSIF(fbtimestamp.systemTime.wHour>=StopTime AND NOT bMotion)THEN//To Boost down temperature
                dwSendData :=16#00000108; //Valve Position = 0%,Setpoint for thermostat=10C

            ELSIF(rTempRoom >= sPoint) THEN //To maintain the temperature at user defined setpoint
                dwSendData := 16#32A50108; //Valve Position=50%
            END_IF
            IF (MAIN.stKL6581.ar_DB[0].7 AND (NOT MAIN.stKL6581.ar_DB[0].4)
                AND (NOT MAIN.stKL6581.ar_DB[0].3)) THEN//Teach in for when required
            dwSendData := 16#000001F0;
            END_IF
            iStep := 1;
        END_IF
```

In Case 0, first the general reception function block – FB_Rec_Generic is called in order to record the values that are being sent by the thermostat. After this the main control logic, as shown in figure 27 is applied. During testing of the program, it was found that the program should only flow further if the general reception block has received all the values in order to

ensure that the correct and most recent values for temperature are used in the logic. Keeping this in mind, an if statement is created which checks if the “bReceive” output value of the function block is false. This value turns to false for one cycle after an EnOcean telegram is received allowing the program to continue further.

Figure 27. Control logic.

```
IF(NOT fbRec_Generic.bReceive) THEN
    rTempRoom := (BYTE_TO_REAL(arrValues[1]) / 12.75) + 10; //Scaling room temperature value
    EnOcean.RoomTemp := rTempRoom;
    bMotion := EnOcean.arrMotionSensor[0].1; //storing Motion status

    IF (((fbtimestamp.systemTime.wHour>=(StartTime-1) AND fbtimestamp.systemTime.wHour<=StopTime)//To boost temperature up
          AND rTempRoom<sPoint AND(fbtimestamp.systemTime.wDayOfWeek>0 AND fbtimestamp.systemTime.wDayOfWeek<6))OR bMotion) THEN
        dwSendData := 16#64FF0108; //Valve position = 100% Setpoint for thermostat=30c
    ELSIF(fbtimestamp.systemTime.wHour>=StopTime AND NOT bMotion) THEN//To Boost down temperature
        dwSendData :=16#00000108; //Valve Position = 0%,Setpoint for thermostat=10C

    ELSIF(rTempRoom >= sPoint) THEN //To maintain the temperature at user defined setpoint
        dwSendData := 16#32A50108; //Valve Position=50%
    END_IF
    IF (MAIN.stKL6581.ar_DB[0].7 AND (NOT MAIN.stKL6581.ar_DB[0].4) AND (NOT MAIN.stKL6581.ar_DB[0].3)) THEN//Teach in for when required
    dwSendData := 16#000001F0;
    END_IF
    iStep := 1;
END_IF
```

Further in case 0, as shown in figure 26, the room temperature that is obtained from the thermostats is recorded for further use. In the logic part which follows, it is seen from figure 27 that parameters such as the start time, stop time, the day of the week and room temperature setpoint are used to ascertain if a boost in temperature is needed. The logic essentially checks for the fulfilment of the defined conditions and then accordingly picks the correct value for dwSendData which contains the information to be sent to the actuator.

Figure 28. Case 1.

```
    1://Calling the 4BS sending function
        fbSend_4BS(
            bStart      := TRUE,
            by_Node     := byNode,
            pt_SendData := ADR(dwSendData),
            nEnOceanID  := nEnoceanID,
            str_KL6581  := MAIN.stKL6581,
            bBusy       => ,
            bError      => ,
            iErrorID    => );

        IF (NOT fbSend_4BS.bBusy) THEN //TO Check if data has been sent, if it is sending bBusy will be true
            fbSend_4BS(bStart := FALSE);
            iStep:=0;
        END_IF


END_CASE
```

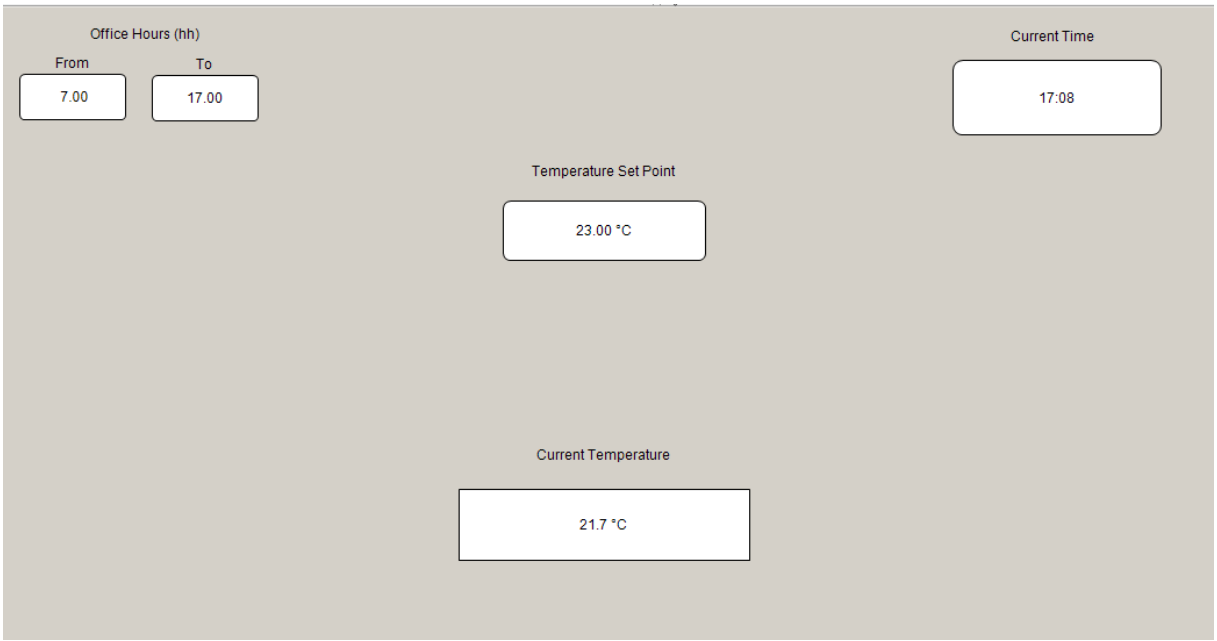
Case 1 is exclusively for sending the data using the FB_Send_4BS function block. After the value of dwSendData variable has been set in Step 0, in Step 1 it is fed to the FB_Send_4BS function block to be sent to the target thermostat. Once that has been done, the program is

sent back to step 0 to check the conditions and update the value of the data that is to be sent if necessary.

6.4.4 Visualization

The visualization has been created using components from the various inbuilt visualization libraries that are available when initialising a PLC project on TwinCat. Figure 29 shows the visualisation that has been created for the system.

Figure 29. Visualisation for the system.



As shown in Figure 29, the visualisation has features that allow the users to input the office timings or the timings when the room must be heated. In addition to this the user can input the desired temperature setpoint for the system to attain. The visualisation also displays the current temperature and time.

The office hours timings need to be input in the 24-hour format and it only accepts hour wise inputs, hence inputting a value such as 17:30 will give an error. The temperature setpoint can be input using decimals to the nearest tenths as the system can achieve that accuracy.

6.5 Results

The implementation is set up and is fully functional. The sensors and actuators have been connected to the system and are working according to the functional description. Figure 30 shows the system working during the weekdays in a graphical format.

Figure 30. Graphical representation of the radiator temperature and room temperature with the implemented system (weekdays).

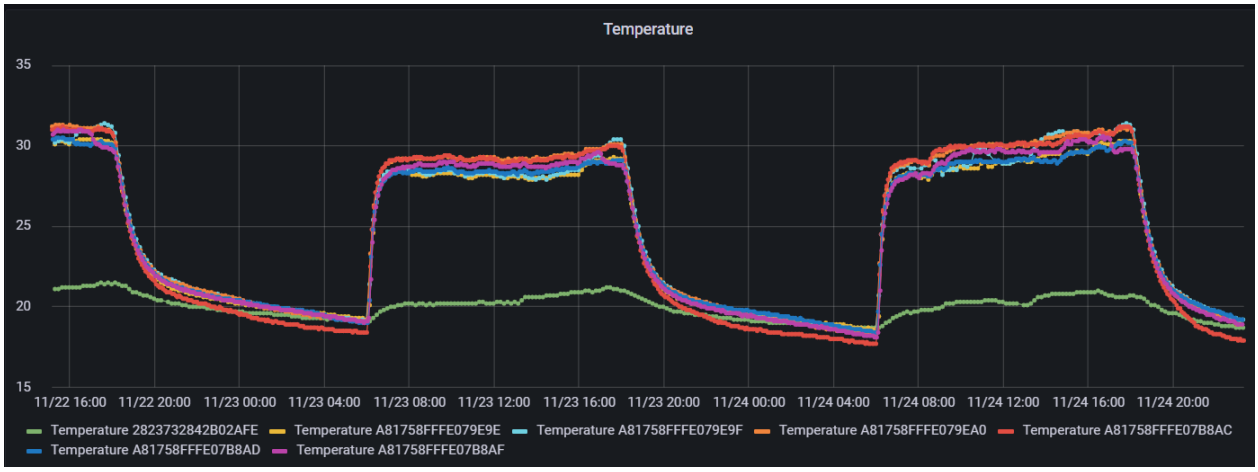


Figure 30 shows the rise and fall in the radiator and room temperature with the system implemented. The green line represents the room temperature, and all the other lines represent the radiator temperatures. Six temperature sensors were placed on either ends of all three radiators to ascertain what the radiator temperatures are. As can be seen from the graph, which captures the data over a two-day period, the temperatures start falling at 5 P.M. and then start rising at 7 A.M. every day. The rise is followed by the temperatures being stable which is in accordance with the planned implementation. Figure 31 is a graphical representation of the system functioning during the weekends.

Figure 31. Graphical representation of temperature control (weekend).

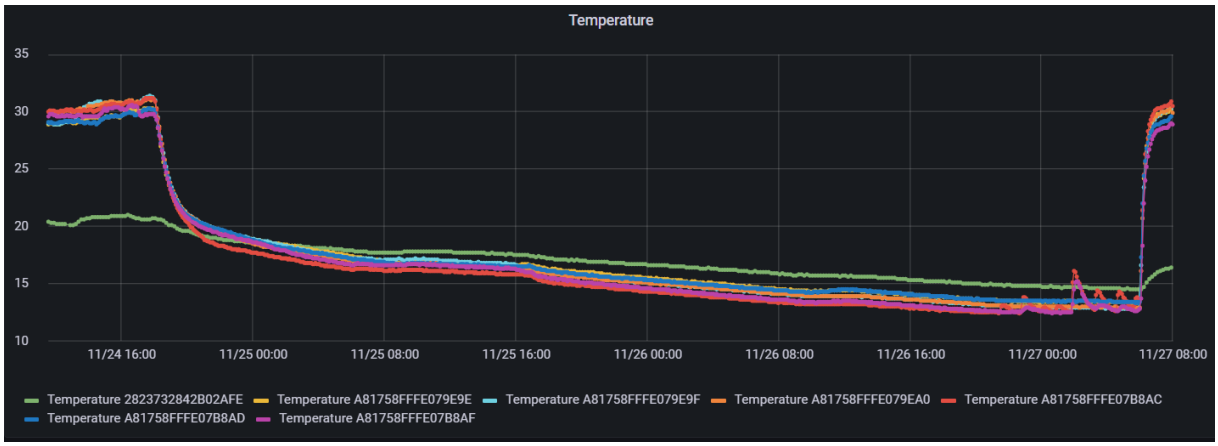


Figure 31 shows the control system in action during the weekend. A steep drop in radiator temperature can be seen on Friday 24 November in the evening at 5 P.M. after which the radiators continually lose heat indicating that the water supply to them has been turned off till Monday 27 November at 7 A.M. when the radiator temperature spikes again. The temperature setpoint not reaching 10°C is attributed to the ventilation system in the room being on and heating continually maintaining the temperature to be above 14°C.

The analysis of energy savings is being done using the energy consumption of the radiators. There are 3 radiators in the room, each measuring 197cmx10cmx45cm. The radiators use approximately 1.7 kW and in total for 3 radiators this value stands at 5.1 kW at the peak when the temperature setpoint is at the highest. Equation 3 is the general formula used to compute the energy consumed, in kWh, by the systems.

Equation 3. Equation to calculate the energy usage in kWh.

$$E = Pt$$

Where E is the energy consumption, P is the power in Kilowatts and t is the time in hours. Equation 4 computes the total energy consumed in 24 hours when the pre-existing system is in place.

Equation 4. Energy consumed in 24 hours (Traditional system).

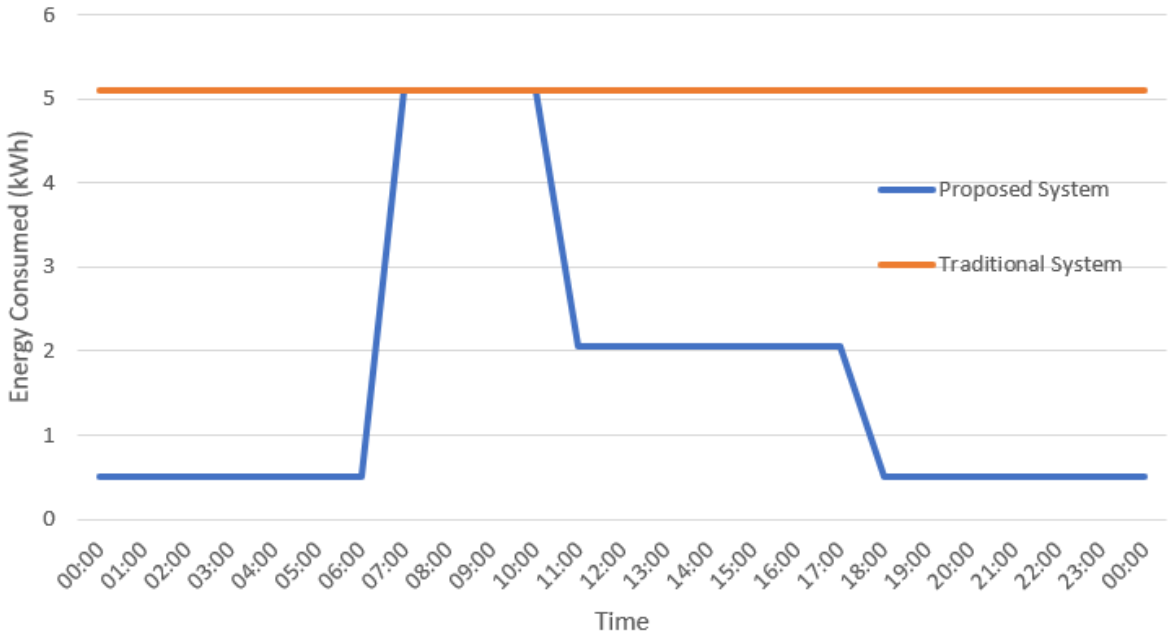
$$5.1\ kW * 24\ h = 122.4\ kWh$$

Assuming that with the proposed system it takes three hours to get to the set point during which period the radiators are at 100% energy consumption, then the radiators are at 50% capacity for seven hours to maintain the temperature and for fourteen hours during the downtime the energy consumption of the radiators is at 10%. Equation 5 shows the computation of the energy consumed using the proposed system and figure 32 depicts this case graphically using the blue line.

Equation 5. Energy Consumed in 24 hours (Proposed System).

$$(5.1\ kW * 3\ h) + (2.05\ kW * 7\ h) + (0.51\ kW * 14\ h) = 36.79\ kWh$$

Figure 32. Graphical depiction of difference in energy consumption of the two systems.



Equation 6. Energy consumed using the proposed system when the space is used for 16 hours (Worst case).

$$(5.1\ kW * 3\ h) + (2.05\ kW * 13\ h) + (0.51\ kW * 8\ h) = 46.03\ kWh$$

As can be seen by the calculation, there is a reduction of 69.9% in the energy consumption in a perfect scenario and a reduction of 62.3% in the worst case where the space is being used for 16 hours from 6 A.M. till 10 P.M.

During weekends when the space is not being used from Friday end of day at 5 P.M. to Monday start of day at 6 A.M. (61 hours). Equations 7 and 8 show the energy consumptions by the pre-existing and the proposed systems during the weekend.

Equation 7. Energy consumed by the traditional heating system during the weekend.

$$5.1\ kW * 61\ h = 311.1\ kWh$$

Equation 8. Energy consumed by the proposed system during the weekend.

$$0.51\ kW * 61\ h = 31.11\ kWh$$

This represents energy savings of 90%

As can be seen from the results, implementation of this system can be quite beneficial to conserving energy at the radiator level.

7 Conclusion

The thesis was a great learning experience which allowed the author to apply his knowledgebase and at the same time build on it. The implementation of the project was a success and after analysis it was found that the implemented system can have a significant impact on the energy consumption, specifically, in the heating system of a building.

However, the thesis process faced some challenges. A significant challenge was the usage of the HORA Smart Drive MX actuators which had very little documentation and the TwinCat EnOcean Library that had some unclear points, for example, about the utilisation of the virtual EnOcean ID. This led to many failed attempts at creating a connection between the three actuators and the controller. Finally, with some clarification from the Beckhoff Support team and through the process of trial and error, the author was able to implement a code that could bidirectionally communicate with the actuators. It was learnt during this that only one “FB_Send_4BS” function block is supported per KL6581 card and that the same block must be parameterized for each device and then called sequentially.

This led to the software being redesigned, where previously all the communication was being done in the MAIN function, it was decided to create a separate function block to communicate with the actuator and apply the control logic. This function block was then parameterized according to the virtual EnOcean ID of each actuator and called in the MAIN function.

Another challenge was the utilisation of the STM-330 temperature sensors. Initially, all the sensors were connected to the system and were giving values, however, it was discovered that after office hours when the lights in the room were turned off, the sensors could not function due to low power. This resulted in errors in the transmission of temperature values which were seen to affect the system. A solution to this problem was the usage of the HORA Smart Drive MX’s inbuilt temperature sensors which has an inbuilt correction algorithm to filter out other temperature influences and return the ambient temperature of the room.

Lastly, as modifications could not be made to the ventilation system present in the room without affecting other rooms in the building, the temperature does not always reach the desired setpoint. However, as can be seen from the graphs in the results section, a significant rise and fall of the radiator temperature can be noted which would be the basis of the temperature in the room if the ventilation system could be controlled.

The thesis was a great opportunity that helped the author gain a deep insight into the integration of IoT devices with PLCs. It also helped the author gain an understanding of wireless technology and radio communication.

The results of the analysis showing that the system can result in a reduction of close to 70% of the energy consumed by the heating system further shows that the system is a viable alternative to using analogue control based hydronic heating systems.

References

Abachy. (n.d.). Data Collection; Building control systems. Abachy. Retrieved from https://abachy.com/catalog/factory-monitoring-control-systems-fmcs/data-collection-building-control-systems

Alwetaishi, M. (2016). Impact of Building Function on Thermal Comfort: A Review Paper. *American Journal of Engineering and Applied Sciences, 9(4)*, 928-945. https://doi.org/10.3844/ajeassp.2016.928.945

Beckhoff (n.d.). KL6581 | Bus Terminal, 1-channel communication interface, EnOcean, master. Beckhoff. Retrieved on 27 October 2023, from https://www.beckhoff.com/en-en/products/i-o/bus-terminals/kl6xxx-communication/kl6581.html

Campano, M. Á., Acosta, I., Domínguez, S., & López-Lovillo, R. (2022). Dynamic analysis of office lighting smart controls management based on user requirements. *Automation in construction, 133, 104021*. https://doi.org/10.1016/j.autcon.2021.104021

Christensen, E. (2007, August 30). File: Termostat – Fabrikken 002.jpg. Wikimedia Commons. Retrieved on 24 November 2023, from https://commons.wikimedia.org/w/index.php?curid=2653531

Devi, P. (2023, August 14). EnOcean Wireless Communication Protocol. Automation Community. Retrieved from https://automationcommunity.com/enocean-wireless-communication/

Electrical4U. (2020, December 27 - a). Control Systems: What Are They? (Open-Loop & Closed-Loop Control System Examples). Electrical4U. Retrieved from https://www.electrical4u.com/control-system-closed-loop-open-loop-control-system/?utm_content=cmp-true

Electrical4U. (2020, October 23 - b). Types of Control Systems | Linear and Non-Linear Control Systems. Electrical4U. Retrieved from https://www.electrical4u.com/types-of-systems-linear-and-non-linear-system/

EnOcean Alliance (n.d.). HORA SMARTDRIVE MX. EnOcean GmbH. Retrieved on 7 October 2023, from https://www.enocean-alliance.org/fr/product/hora_smartdrive_mx/

EnOcean GmbH. (2013, April 11). New EnOcean software opens door to world of energy harvesting wireless technology. News.Cision. Retrieved from https://news.cision.com/enocean-gmbh/r/new-enocean-software-opens-door-to-world-of-energy-harvesting-wireless-technology,c9399354

Instrumentation Tools (n.d.). Components of PLC. Instrumentation Tools. Retrieved on 15 October 2023, from https://instrumentationtools.com/components-of-plc/?utm_content=cmp-true

Kusiak, A., Li, M., & Tang, F. (2010). Modeling and optimization of HVAC energy consumption. *Applied energy, 87(10)*, 3092-3102. https://doi.org/10.1016/j.apenergy.2010.04.008

Less, Brennan D., Dutton, Spencer M., Walker, Iain S., Sherman, Max H., & Clark, Jordan D. (2019, April 24). Energy savings with outdoor temperature-based smart ventilation control strategies in advanced California homes. *Energy and Buildings, 194*, 317-327. https://doi.org/10.1016/j.enbuild.2019.04.028

Mikroe. (2016, May 6). EnOcean – Energy Harvesting Wireless Solution. Mikroe. Retrieved from https://www.mikroe.com/blog/enocean-energy-harvesting-wireless-solution

Parkstone Yorkshire (2020, October 12). One pipe system used in older central heating systems. Parkstone-Yorkshire. Retrieved on 17 October 2023, from https://www.parkstoneyorkshire.co.uk/one-pipe-system-used-in-older-central-heating-systems/

PLCtechnician (2018, November 12). What are the essential elements of a PLC System? Plctechnician. Retrieved on 15 October 2023, from https://www.plctechnician.com/news-blog/what-are-essential-elements-plc-system

Sinopoli, J. (2016, August 15). Smart Buildings. Whole Building Design Guide. Retrieved from https://www.wbdg.org/resources/smart-controls

Smart Touch (2022, November 2). Energy Efficient Building. Smart Touch. Retrieved from https://www.smart-touch.hr/energy-efficient-building/

Van Thillo, L., Verbeke, S., & Audenaert, A. (2022). The potential of building automation and control systems to lower the energy demand in residential buildings: A review of their performance and influencing parameters. *Renewable & Sustainable Energy Reviews, 158*, 112099. https://doi.org/10.1016/j.rser.2022.112099

VanDoren, V. (2006, July 1). Feedback loops control discrete, continuous processes. Control Engineering. Retrieved on 28 November 2023, from https://www.controleng.com/articles/feedback-loops-control-discrete-continuous-processes/

Appendix 1. Code for the system

MAIN (PRG)

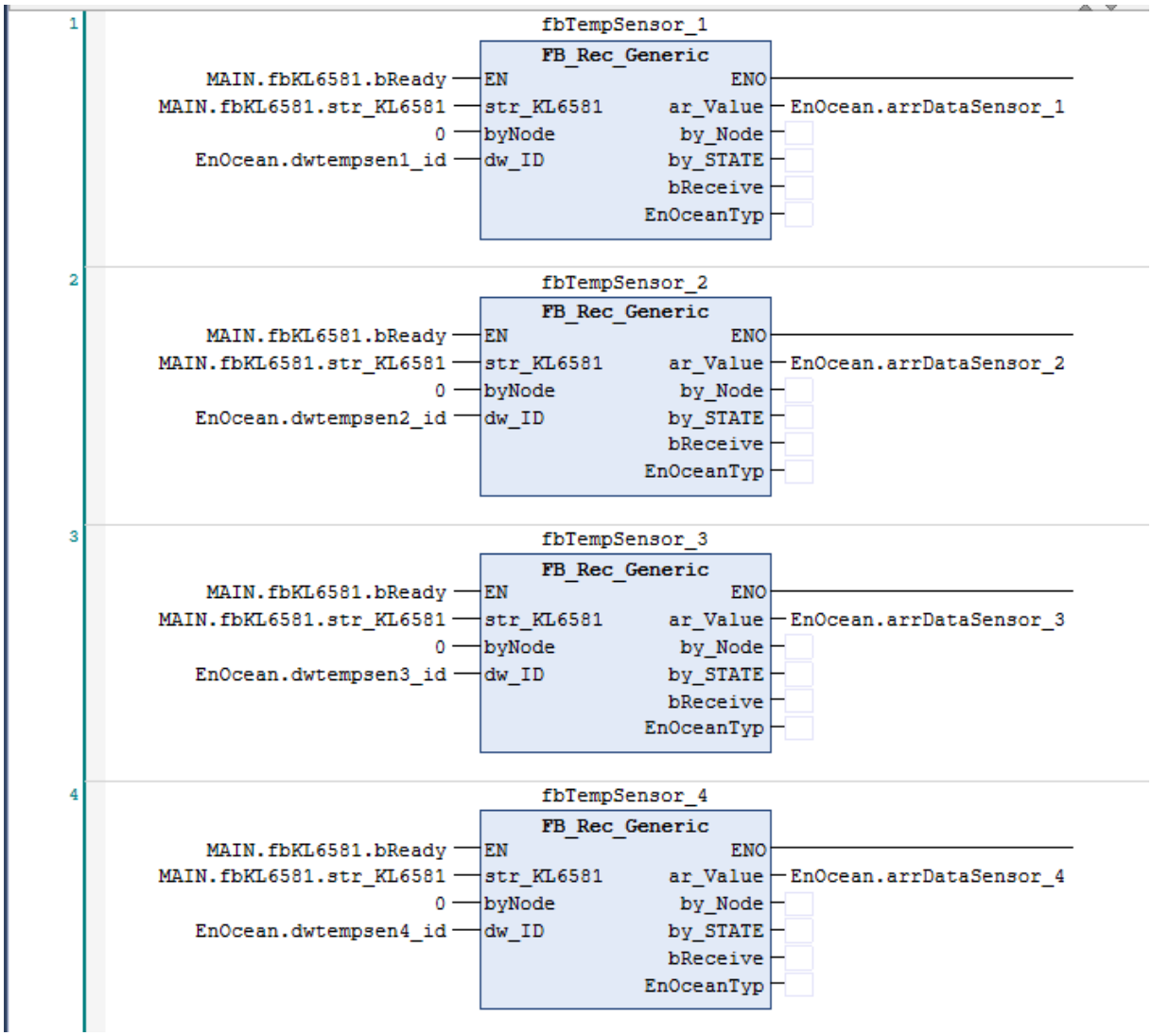
```
PROGRAM MAIN
fbKL6581(
    bInit           := TRUE,
    nIdx            := 1,
    stKL6581_in     := stKL6581_Input,
    stKL6581_out    := stKL6581_Output,
    bReady          => ,
    bBusy           => ,
    bError          => ,
    iErrorID        => ,
    str_KL6581      => stKL6581);

IF (NOT fbKL6581.bReady) THEN
    RETURN;
END_IF


EnOceanCom();
therm1Send(nEnoceanID := 1, dwID := dwId,StartTime:= EnOcean.iFromTime, StopTime:=EnOcean.iToTime,
            sPoint:=EnOcean.rSetPoint);//Calling for thermostat 1
therm2Send(nEnoceanID := 2, dwID := dwId2,StartTime:= EnOcean.iFromTime, StopTime:=EnOcean.iToTime,
            sPoint:=EnOcean.rSetPoint);//Calling for thermostat 2
therm3Send(nEnoceanID := 3, dwID := dwId3,StartTime:= EnOcean.iFromTime, StopTime:=EnOcean.iToTime,
            sPoint:=EnOcean.rSetPoint);//Calling for thermostat 3
```

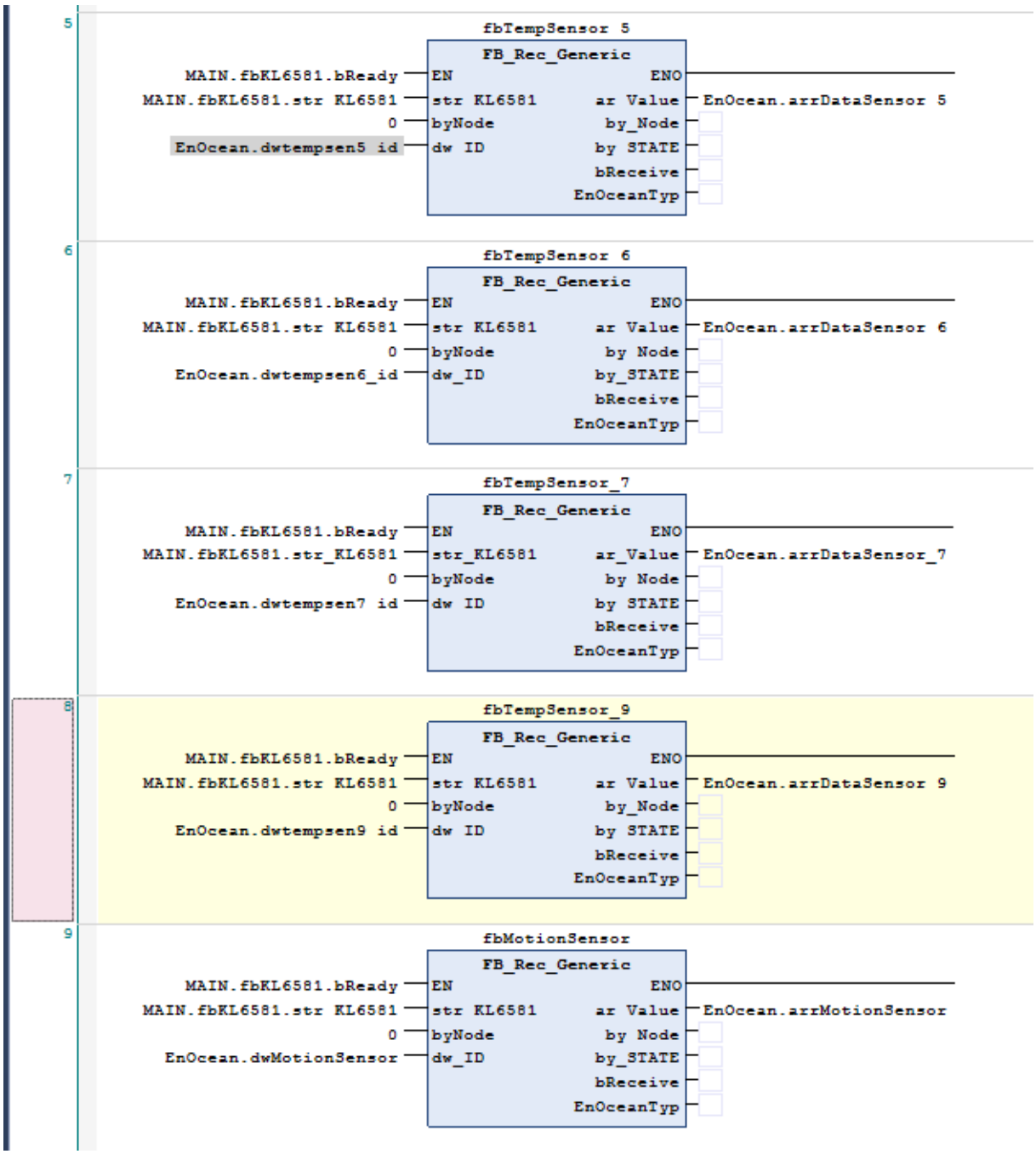
Thermostat_send (FB)

```
fbtimestamp();
CASE iStep OF

    0:
        (* Calling the general reception function block. *)
        fbRec_Generic(
            str_KL6581  := MAIN.stKL6581,
            byNode      := 1,
            dw_ID       := dwID,
            ar_Value    => arrValues,
            by_Node     => ,
            by_STATE    => ,
            bReceive    => ,
            EnOceanTyp  => );

        IF(NOT fbRec_Generic.bReceive) THEN
            rTempRoom := (BYTE_TO_REAL(arrValues[1]) / 12.75) + 10; //Scaling room temperature value
            EnOcean.RoomTemp := rTempRoom+1;
            //bMotion := EnOcean.arrMotionSensor[0].1; //storing Motion status

            IF (((fbtimestamp.systemTime.wHour>=(StartTime-1) AND fbtimestamp.systemTime.wHour<=StopTime)//To boost temperature up
                    AND rTempRoom<sPoint AND(fbtimestamp.systemTime.wDayOfWeek>0
                    AND fbtimestamp.systemTime.wDayOfWeek<6))(*OR bMotion*)) THEN
                dwSendData := 16#64FF0108; //Valve position = 100% Setpoint for thermostat=30c
            ELSIF(fbtimestamp.systemTime.wHour>=StopTime (*AND NOT bMotion)*)) THEN//To Boost down temperature
                dwSendData :=16#00000108; //Valve Position = 0%,Setpoint for thermostat=10C

            ELSIF(rTempRoom >= sPoint) THEN //To maintain the temperature at user defined setpoint
                dwSendData := 16#32A50108; //Valve Position=50%
            END_IF
            IF (MAIN.stKL6581.ar_DB[0].7 AND (NOT MAIN.stKL6581.ar_DB[0].4)
                AND (NOT MAIN.stKL6581.ar_DB[0].3)) THEN//Teach in for when required
            dwSendData := 16#000001F0;
            END_IF
            iStep := 1;
        END_IF
    1://Calling the 4BS sending function
        fbSend_4BS(
            bStart      := TRUE,
            by_Node     := byNode,
```

```
        END_IF
    1://Calling the 4BS sending function
        fbSend_4BS(
            bStart      := TRUE,
            by_Node     := byNode,
            pt_SendData := ADR(dwSendData),
            nEnOceanID  := nEnoceanID,
            str_KL6581  := MAIN.stKL6581,
            bBusy       => ,
            bError      => ,
            iErrorID    => );

        IF (NOT fbSend_4BS.bBusy) THEN //TO Check if data has been sent, if it is sending bBusy will be true
            fbSend_4BS(bStart := FALSE);
            iStep:=0;
        END_IF


END_CASE
```

EnOceanCom(PRG)




5
fbTempSensor_5
FB_Rec_Generic
MAIN.fbKL6581.bReady
MAIN.fbKL6581.str_KL6581
0
EnOcean.dwtempsen5_id
EN
str_KL6581
byNode
dw_ID
ENO
ar_Value
by_Node
by_STATE
bReceive
EnOceanTyp
EnOcean.arrDataSensor_5
6
fbTempSensor_6
FB_Rec_Generic
MAIN.fbKL6581.bReady
MAIN.fbKL6581.str_KL6581
0
EnOcean.dwtempsen6_id
EN
str_KL6581
byNode
dw_ID
ENO
ar_Value
by_Node
by_STATE
bReceive
EnOceanTyp
EnOcean.arrDataSensor_6
7
fbTempSensor_7
FB_Rec_Generic
MAIN.fbKL6581.bReady
MAIN.fbKL6581.str_KL6581
0
EnOcean.dwtempsen7_id
EN
str_KL6581
byNode
dw_ID
ENO
ar_Value
by_Node
by_STATE
bReceive
EnOceanTyp
EnOcean.arrDataSensor_7
8
fbTempSensor_9
FB_Rec_Generic
MAIN.fbKL6581.bReady
MAIN.fbKL6581.str_KL6581
0
EnOcean.dwtempsen9_id
EN
str_KL6581
byNode
dw_ID
ENO
ar_Value
by_Node
by_STATE
bReceive
EnOceanTyp
EnOcean.arrDataSensor_9
9
fbMotionSensor
FB_Rec_Generic
MAIN.fbKL6581.bReady
MAIN.fbKL6581.str_KL6581
0
EnOcean.dwMotionSensor
EN
str_KL6581
byNode
dw_ID
ENO
ar_Value
by_Node
by_STATE
bReceive
EnOceanTyp
EnOcean.arrMotionSensor

Appendix 2. List of Variables

EnOcean (GVL)

```
{attribute 'qualified_only'}
VAR_GLOBAL
    bInit: BOOL := TRUE;
    nIndxKL6581: USINT;

    bsendinfo: BOOL;
    pvToSend: PVOID;
    arrDataToSend: ARRAY [0..3] OF BYTE;

    dwtempsen1_id: DWORD := 16#05028547;
    dwtempsen2_id: DWORD := 16#0502854C;
    dwtempsen3_id: DWORD := 16#050284E4;
    dwtempsen4_id: DWORD := 16#0502854C;
    dwtempsen5_id: DWORD := 16#0502850E;
    dwtempsen6_id: DWORD := 16#01A3B1CE;
    dwtempsen7_id: DWORD := 16#050C5206;
    dwtempsen8_id: DWORD := 16#050282E0;
    dwtempsen9_id: DWORD := 16#050C5589;
    dwMotionSensor:DWORD := 16#019D9364;

    arrMotionSensor: ARRAY[0..3] OF BYTE;

    dwthermostat1_id: DWORD := 26910074;
    dwthermostat2_id: DWORD := 26803667;
    dwthermostat3_id: DWORD := 16#0196C8B0;

    arrDatathermostat1: ARRAY[0..3] OF BYTE;
    arrDatathermostat2: ARRAY[0..3] OF BYTE;
    arrDatathermostat3: ARRAY[0..3] OF BYTE;



    arrDataSensor_1: ARRAY[0..3] OF BYTE;
    arrDataSensor_2: ARRAY[0..3] OF BYTE;
    arrDataSensor_3: ARRAY[0..3] OF BYTE;
    arrDataSensor_4: ARRAY[0..3] OF BYTE;
    arrDataSensor_5: ARRAY[0..3] OF BYTE;
    arrDataSensor_6: ARRAY[0..3] OF BYTE;
    arrDataSensor_7: ARRAY[0..3] OF BYTE;
    arrDataSensor_8: ARRAY [0..3] OF BYTE;
```

```
41      arrDataSensor_9: ARRAY [0..3] OF BYTE;
42
43      RoomTemp: REAL;
44      iFromTime: DWORD;
45      iToTime: DWORD;
46      rSetPoint: REAL;
47
48
49  END VAR
```

Variables of MAIN

```
1   PROGRAM MAIN
2   VAR
3       fbKL6581                        : FB_KL6581;
4       stKL6581_Input          AT%I*   : KL6581_INPUT;
5       stKL6581_Output         AT%Q*   : KL6581_OUTPUT;
6       stKL6581                        : STR_KL6581;
7
8       fbRec_Generic                   : FB_Rec_Generic;
9       fbRec_Generic_1                 : FB_Rec_Generic;
10      fbRec_Generic_2                 : FB_Rec_Generic;
11      byNode                          : BYTE := 1;
12      dwId                            : DWORD := 26803667;
13      dwId2                           : DWORD := 26910074;
14      dwId3                           : DWORD := 16#0196C8B0;
15      arrValues                       : ARRAY[0..3] OF BYTE;
16      arrValues_1                     : ARRAY[0..3] OF BYTE;
17      arrValues_2                     : ARRAY[0..3] OF BYTE;
18
19      fbSend_4BS                      : FB_Send_4BS;
20      dwSendData                      : DWORD;
21      binSenddata                     : STRING := dword_to_binstr(dwSendData,32);
22      fbtimestamp : FB_LocalSystemTime := (bEnable:=TRUE, dwCycle:=1);
23      bTeachIn                        : BOOL := TRUE;
24      byPosition                      : BYTE;
25      iStep                           : INT;
26      iSendCount                      : INT;
27      rTempRoom                       : REAL;
28      rTempSetpoint                   : REAL;
29      dwId_t2: DWORD;
30      dwId_t3: DWORD;
31      bMotion: BOOL;
32
33      therm1Send: thermostat_send;
34      therm2Send: thermostat_send;
35      therm3Send: thermostat_send;
36  END_VAR
```

Variables of thermostat_send (FB)

```
FUNCTION_BLOCK thermostat_send
VAR_INPUT
    nEnoceanID: BYTE;
    dwID: DWORD;
    StartTime: DWORD;
    StopTime: DWORD;
    sPoint: REAL;
END_VAR
VAR_OUTPUT
    bStatfbsend: BOOL;
END_VAR

VAR
    fbSend_4BS  : FB_Send_4BS;
    fbRec_Generic: FB_Rec_Generic;
    fbtimestamp : FB_LocalSystemTime := (bEnable:=TRUE, dwCycle:=1);
    arrValues: ARRAY[0..3] OF BYTE;
    rTempRoom: REAL;
    dwSendData: DWORD;
    byNode: BYTE := 1;
    iStep: INT;
    bMotion: BOOL;
END_VAR

```

Variables of EnOceanCom(PRG)

```
PROGRAM EnOceanCom
VAR
    fbTempSensor_1: FB_Rec_Generic;
    fbTempSensor_2: FB_Rec_Generic;
    fbTempSensor_3: FB_Rec_Generic;
    fbTempSensor_4: FB_Rec_Generic;
    fbTempSensor_5: FB_Rec_Generic;
    fbTempSensor_6: FB_Rec_Generic;
    fbTempSensor_7: FB_Rec_Generic;
    fbThermostat1: FB_Rec_Generic;
    fbthermostat2: FB_Rec_Generic;

    fbTeachingPressed: FB_Rec_Teach_In_Ex;


    fbTempSensor_9: FB_Rec_Generic;
    fbMotionSensor: FB_Rec_Generic;
END_VAR

```