Bachelor Thesis

Master's Programme in Computer science and engineering, 300 credits

Software development of visualizationsystem

Computer science and engineering, 15 credits

Halmstad 2021-06-06

Rohullah Khormai, Fredrik Kortetjärvi

HALMSTAD UNIVERSITY

Abstract

Today wireless technologies are increasing in the automation systems used in homes and buildings. More electrical devices are used in a house to save time, money, and energy because they are relatively inexpensive and easy to install; these devices even allow smart components such as mobile tablets and computer connectivity. To connect all these devices for data transmission purposes and easy access, the KNX is the best choice. The KNX standard is an open standard for home and building automation. KNX standard supports different communication media such as Twisted pairs, Power line, Radio Frequency, and tunneling IP. KNX system is a bus system for building control, making all electrical and smart devices in a KNX system use the same transmission method and exchange telegrams via a shared bus network.

To check and control all the electrical devices in a home or an apartment takes time; that is why there is a massive need for applications to make every room's controlling process much easier and take a much shorter time. This project is about designing and implementing a visualization application for windows and .NET for managing and comparing input data with the actual data. This application is equipped with a KNX bus driver to communicate with hardware in a building. The practical part of the application is to take some raw data and then sort them in a specific way to minimize the time of controlling the process of the KNX devices in a building.

Sammanfattning

Trådlös teknik ökar i de automatiseringssystem som används i hem och byggnader. Mer elektriska apparater används i ett hus för att spara tid, pengar och energi eftersom de är relativt billiga och enkla att installera. Dessa enheter tillåter även smarta komponenter som mobil telefoner, surfplattor och datoranslutning. För att ansluta alla dessa enheter för dataöverföringsändamål och enkel åtkomst är KNX det bästa valet. KNX-standarden är en öppen standard för hem- och byggnadsautomation. KNX-standarden stöder olika kommunikationsmedier som Twisted pair, Powerline, Radio Frequency och tunneling IP. KNX-systemet är ett bussystem för byggnadskontroll, vilket gör att alla elektriska och smarta enheter i ett KNX-system använder samma överföringsmetod och utbyter telegram via ett delat bussnätverk.
Det tar tid att kontrollera alla elektriska apparater i ett hem eller en lägenhet; det är därför ett stort behov av applikationer för att göra varje rums kontrollprocess mycket enklare och ta mycket kortare tid. Detta projekt handlar om att designa och implementera en visualiseringsapplikation för windows och .NET för att hantera och jämföra indata med den faktiska data. Denna applikation är utrustad med en KNX-bussförare för att kommunicera med hårdvara i en byggnad. Den praktiska delen av applikationen är att ta några rådata och sedan sortera dem på ett specifikt sätt för att minimera tiden för att kontrollera processen av KNX-enheterna i en byggnad eller fastighet.

Contents

1 Introduction

Wireless technologies are improving every day in the home, and it need automation systems to communicate with all the devices in the house. The KNX standard was developed by the Konnex Association for the design of home automation. KNX have evolved from the European Installation Bus (EIB) and now being the open standard. It is a building control communication system that uses information technology to connect devices such as sensors, actuators, controllers, operating terminals, and monitors. KNX's mission is to work with automated homes. KNX standard supports different communication media such as Twisted Pair (KNX TP), Powerline (KNX PL), Radio Frequency (KNX RF), and tunneling IP (KNX IP). In this paper, the KNX IP (KNXnet/IP) is used as communication media[1]. KNX system is a bus system for a building control communication system that uses information technology to connect devices such as sensors, actuators, controllers, operating terminals, and monitors. All the KNX devices are connected and sends data to each other by telegram. Many KNX devices are connected in a large property and exchanging data all the time. To control each device one by one is very time-consuming, and the risk of missing a deviation is major. There is a tremendous need for a visualization system connected to the property system KNX. To operate and test a larger property is very time-consuming and the risk of missing a deviation is big, that is why there is a need for some kind of application to solve these issues and minimize the processing time. Tekniska Byrån have a project to implement an application that works on Windows and .NET. The application is going to be connected to KNX by a driver can be implemented in any high-level programming language that works with .NET[2].

1.1 Purpose

The intended application will ensure quality testing and the time required to test the system on different real estate significantly. The purpose is to develop an application from a foundation that connects to the property system (via existing driver) and listens to the traffic to log in and display it graphically.

1.2 Requirements

The requirement from the company was how they wanted the application to work to their workflow. The requirement was that the application needed to:

- Loaded with a structured csv file from ETS5 program that's used for programming the devices.
- Automatically generate controls that can easily be modified and test the KNX system with the csv file that has been loaded.
- Can handle on / off controls, Lights with value from 0-100%, Up / down and Stop / step for shutter controls, Position from 0-100% for shutter controls.
- Has a function for connection to the KNX installation via an existing and open driver from the company.
- Generates informative status back about what has been done during writing and reading to or from the bus system.

2 Background

Tekniska Byrån is working with the KNX system and in this section, the background and a short introduction about Tekniska Byrån and KNX will be explained.

2.1 What is KNX?

KNX has evolved from the European Installation Bus (EIB) and is now the open standard. It is a building control communication system that uses information technology to connect devices such as sensors, actuators, controllers, operating terminals, and monitors. KNX was founded in May 1999 by the European installation bus association (EIBA), European home systems association (EHSA), and BatiBus club international (BCI) associations[3].

KNX is an open global standard for commercial and domestic building automation. KNX is ISO/IEC 14543-3. This standard is based on open systems interconnection(OSI)[4], ISO/IEC 14543-3 is a wireless protocol for low power devices. The protocol is made to keep switches and sensors on super low energy consumption. KNX is built upon the predecessor European installation bus (EIB) and is compatible with every EIB compatible device. KNX standard is extended from an OSI-based protocol stack with the physical layer[5].

2.1.1 Why KNX?

Smart homes technology is improving fast and KNX has already had the latest technologies such as the internet of things in their current software. KNX plays a huge role in building automation (Smart House) worldwide and here are some reasons why KNX is important to learn: KNX provides the ideal introduction to the world of bus technology a lot of new functions and features are used in KNX current bus systems for example bit rate, media access control, data frames, and protocols. Learning KNX helps to become familiar with how to solve automation-related problems. There are 10% of electronic industries that can plan and install KNX systems[6].

KNX's mission is to work with automated homes. KNX is working on automation technologies that the smart home market recognizes as KNX association, but what is an automated home?
The automated home is also known as a smart house. Automated homes incorporate common devices that control features of the home. In the beginning, the functionality of a smart home was to control environmental systems such as lighting and heating, but technology has been developed so that every electrical component within the house can be included in the system. Smart house means that all electrical devices work together and can be modified by smartphones or by an application that is connected to a network. Devices communicate with each other by using the "busline". A busline is a cable that connects all the devices and enables interconnectivity between devices in different rooms throughout the home[7].

KNX is an open standard that makes companies build their own KNX devices freely and still work with other KNX devices without problems. The standard makes KNX not linked to any brands or manufacturers to build or maintain the devices. To the end-user, it is perfect if they want to buy special devices that match the wallpapers. KNX is open for communication with other devices that are not built for KNX systems without issues, and the KNX bus can be used for other devices that can extend the automation solutions[8]. Home assistant can integrate KNX Devices into home automation solutions for more features than ETS software can bring to home automation. Home assistant is type of software that can control devices and can be integrated with other devices with other standards like Zigbee, z-wave, plex, google assistants, and much more[9]. Loxone is also a home automation solution that can compete with the KNX system, but the main difference is that Loxone is not a standard or have any universal standards, so it is locked to the Loxone company that makes it less extendable due to price, brands, and devices. Loxone is easier to set up than the KNX system due to a much more complex setup for KNX than for Loxone. KNX systems often need to be set up by experienced technicians[10].

2.1.2 KNX Bus Devices

KNX devices can be classified into four main groups like in Figure 1:

- System components(Power supply units, accumulators, USB interfaces, and more).
- Sensors(Switch sensors, motion detectors, and glass break detectors).
- Actuators(Switch and blinds).
- Logical components and control panel.

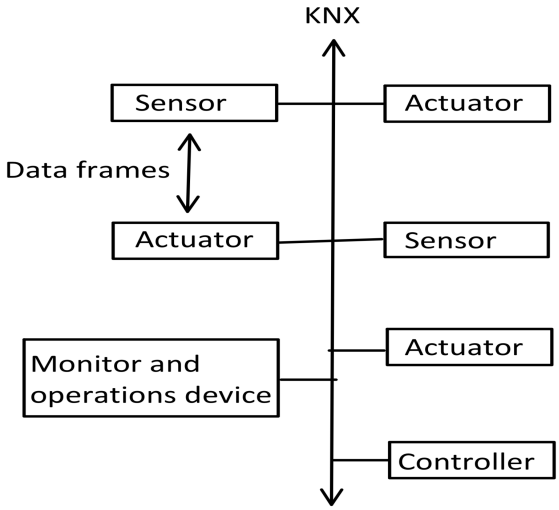

Figure 1: Devices over KNX [11]

To connect these devices the following communication media are available for a KNX system.

- KNX twisted pair(KNX TP): the twisted pair is a cable with two twisted conductors for a single circuit for enhancing electromagnetic compatibility (EMC) that will be acceptable in the electromagnetic environment
- KNX power line(KNX PL): using the electricity cables in a building as a communication medium. 230V main electricity will provide the power required by bus devices.
- KNX radio frequency(KNX RF): communication with the help of radio signals.
- KNX Ethernet(KNX IP): communication with the help of internet protocols[12].

2.1.3 KNX Telegram

KNX bus devices exchange data between each other by telegram. A telegram consists of 8 bits characters, and a field is a combination of several characters. The communication media have different telegram structures:

1. KNX Twisted pair telegram has four fields:
    - Control field: it is responsible for receiving a response from the receiver whether the telegram transmission was successful or not.
    - Address field: this field contains the address of the sender and receiver.
    - Data field: this field contains the telegram's payload.
    - Checksum: this field is for parity checks. By parity check means that this field is used for detecting errors in the communication channel.
2. KNX Power line has four fields:
    - Training field: this field arranges and sets the levels of senders and destinations.
    - Preamble field: the objective of this field is to show the start of transmission, control access to the bus, and avoid telegrams from a collision.
    - Third field: this field contains the KNX twisted-pair telegram.
    - System ID field: this field keeps an ID of signal from different KNX power line systems separately. It is because devices with the same ID can communicate with one another.
3. Radio Frequency telegram

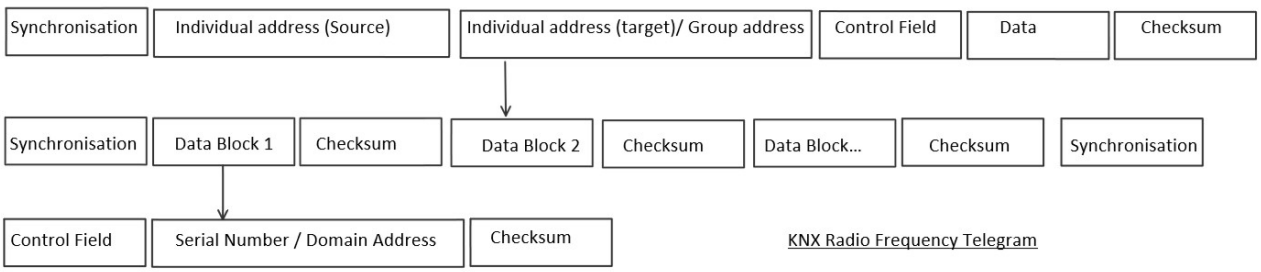

Figure 2: Radio Frequency [13]

4. KNX internet protocol telegram
    - Header Length: this field is to identify the stat of the telegram.
    - Protocol version: this field shows what version of the KNX IP applies.
    - KNX IP service type identifier: it shows the action that should be carried out.
    - Total length: the objective of this field is to show the size of the KNX IP telegram.
    - KNX IP body: this field contains telegrams payload figure 3 [14].

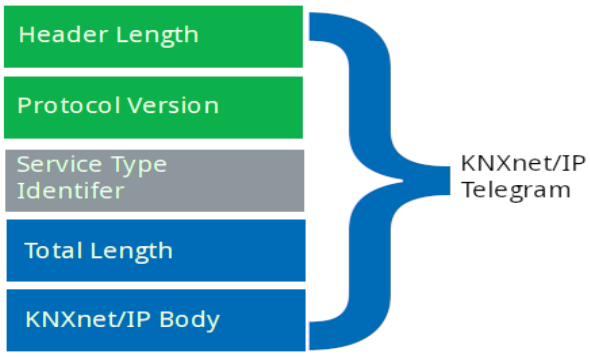

Figure 3: KNXnet/IP telegram[14]

2.1.4 KNX protocol

The KNX system uses two Ethernet communication methods. The first one is tunneling which uses the user data-gram protocol (UDP), which is operating to access the bus from a local network or the internet to program the KNX installation[15]. The second one is routing methods that provide access to exchange telegrams over an Ethernet network, such as KNX Falcon.net v2.20. The KNX ETC run-time 2.2v is a driver that allows access to the KNX bus and offers an application program interface (API) to send and receive telegrams across the KNX network, figure 4.The KNX virtual is an image of a physical KNX component. KNX virtual uses IP, IP port, and TP addresses to communicate with drivers and other applications. KNX driver uses physical address and KNXnet/IP to communicate with other devices such as KNX virtual.The KNX protocol for these methods is called KNXnet/IP[16].

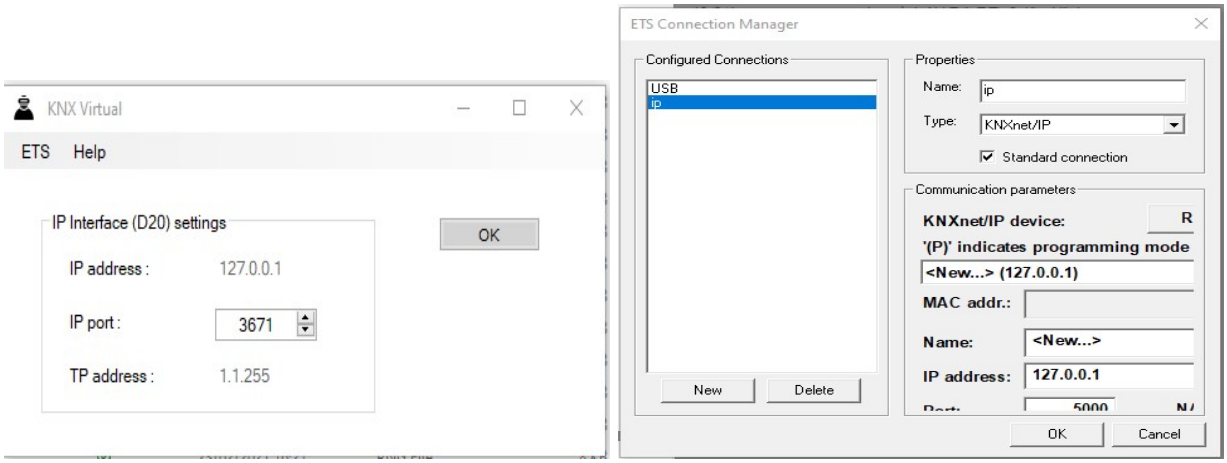

Figure 4: KNX Virtual and KNX Driver

2.1.5 KNXnet/IP

A KNXnet/IP system provides integration access to different KNX networks through an Internet protocol. Integration is realised by a specific device called KNXnet/IP router. IP communication in KNX can be explained using the OSI reference model.

Application Layer: communication takes place via this layer which generates the KNXnet/IP telegram.
Transport Layer: where the tunneling method occurs. The transport layer in the KNX standard provides four different communication modes:
1. Multi-point-to multi-point connection-less (multi-cast)
2. Point-to-all-points connection-less (broadcast)
3. Point-to-point connection-less (uni-cast)
4. Point-to-point connection-oriented
Network Layer: in this layer, internet protocol happens.
Physical Layer and Data-Link layer: Ethernet table 1 [17].

Layer	Standards and Protocols
Application	HTTP,FTP,SMTP
Presentation	JPEG,GIF,MPEG
Session	AppleTalk, WinSock
Transport	TCP, UDP
Network	IP, ICMP, IPX
Link	Ethernet, ATM
Physical	RS232, T1, 802.x

Table 1: OSI reference Model with protocols in a KNX System[18]

2.2 Teknisk Byrån

Tekniska Byrån is a consultant within the electricity industry with a focus on innovative solutions and smart technology. The company currently has 3 employees and has been active since 2000. The employees work with advice, descriptions and designs of electrical installations and as far as they can with property automation and mainly KNX. Within KNX, they conduct design, commissioning, support, training and take care of office functions for interest association with KNX Sweden. In addition to this, they have also programmed certain hardware and software against the system. Tekniska Byrån has premises in Vetlanda and Linköping and has over the years had several interns and supervised a degree project.

2.2.1 Services

Tekniska Byrån is active in the following areas:

- Electrical design: activities in the field of electrical engineering including installation/construction. They install electricity in houses, apartments, schools, offices, and more. They also work in electrical design, construction, and documentation.
- Programming: tekniska Byrån jobs have programming projects to be able to have smart functions in their property system. Tekniska Byrån performs smart property automation with programming for KNX/EIB(ETS), IHC-Smart House, Mod-bus, PLC(Programmable Logic Controller), SCADA(Supervisory Control and Data Acquisition), OPC(Open Platform Communication), and the programming languages are HTML, ASP, PHP, VB, SQL.
- Education: the technical agency has many completed education programs and also in real estate automation and CAD technology. Property automation training includes an introduction to intelligent properties, a complete certified education center for KNX and IHC-Smart House.

2.2.2 Products

Building Portal Suite creates a portal to the KNX system in the form of a simple PC application. In the application, different functions can be handled depending on which component is installed. Products that Tekniska Byrån develops that are used for smart properties are the following:

- Alarm: the alarm is a compact, easy-to-use, customizable and flexible program for Windows that displays and follows up alarms in the KNX system. The alarm takes care of 3 different alarm classes (A, B, and C) where class A has the highest priority and audio features are used to attract attention.
- Controller: the user can with the included interface configure their buttons and display the data. The Controller is easy to use and customizable at the same time. The states and values displayed are actual values loaded from the system. All fields are updated automatically. Programs for the controller are flexible and compact for windows desktops.
- Logger: logger collects measurement data from the KNX system. The data that the logger collects can be used for example debiting, analysis or visualization with the help of a diagram. All values are saved together with the date and time when they were read. The values can be easily exported to XML and imported into another application for example in Excel, for further processing.
- Meter: meter is a flexible program for PC's that collects measurement data from energy meters in the KNX system. The purpose of collecting energy measurement data is that they can be used for further processes like debiting, analysis or visualization with the help of a diagram. To save the data in this program is the same process as in the logger program[19].

2.3 Related Work

KNX is a worldwide communication standard for home automation. It's goal is to support the automation of different electric devices such as heating, ventilation, air-condition, and other intelligent devices. That is why there are many thesis and projects related to the KNX bus systems and their functionality [15], for example, KNX IPv6: Design issues and proposed architecture by Stefan Seifried, Gunther Gridling, and Wolfgang Kastner. The paper is about using IPv6 networking as a native KNX medium. The existing issues in KNXnet/IP networking and the solution to these issues are discussed in this paper. This thesis aims to increase performance and compatibility with standard information and communication technology (ICT) networking. Estimating KNXnet/IP routing congestion is about analyzing the effect of congestion of KNXnet/IP routers on the information flow between KNX components linked to different KNX networks through an internet protocol backbone[16]. These works are related to our project due to the KNXnet/IP and a brief explanation about how the data are sending and receiving work using an internet protocol in a KNX bus system[20].
Implementation of a KNX-ZigBee Gateway for Home automation is an examination about KNX-ZigBee Gateway, it's functionality and interfacing between KNX and ZigBee systems. ZigBee is a protocol used to link smart devices like lights, plugs, and smart locks to a home network, figure 5. KNX-ZigBee Gateway aims to translate the user data attributes, application level services, and translation of addresses that are received from devices using ZigBee protocols and send it to a KNX bus system. By attribute, translation means the translation of data from one form to the other using Data Points (DPT) functions. The DPTs are defined by the KNX system, which enables the user to encode data. One bit of user data can be defined as DPTSwitch("1.001"), DPTBoolean("1.002"), DptEnable("1.003") and so on[21]. This work is related to this project because, in visualization application, it's uses those function that is defined by the KNX system to convert data format(More explanation in section 4), figure 6.

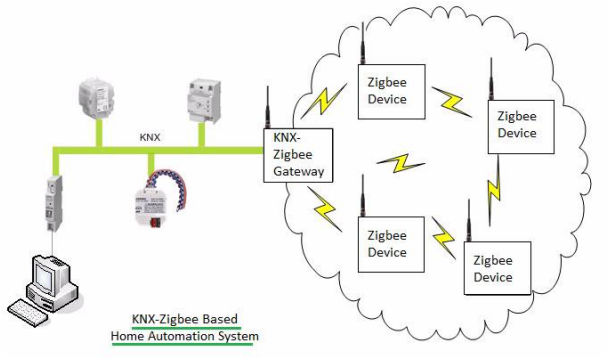

Figure 5: KNX Gateway interfacing between KNX bus system and ZigBee [21]

Name of Address	Type of Data	DPT	Bit while sending
Turn on/off switch	1 bit, 1/0 (1 = turn on, 0 = turn off)	1.001	True
Value	1 byte, 0..255 (0...100%)	5.001	False
U/N	1 bit, 1/0 (1 =Jalousie up, 0 = Jalousie down)	1.008	True
S/S	1 bit, 1/0 (1 = scroll up, 0 = scroll down)	1.009	True
Position	1 byte, 0..255 (0..100%)	5.001	False

Figure 6: Data point table

An example of a similar application would be the KNX Ultimate, which has a similar task as our application. The KNX ultimate is written in Node-red, making it more web-based oriented. This application also works with CSV files generated from an ETS program and uses an error detector mechanism, but it's use is less individual testing in KNX systems. It does not open a CSV file. Instead, the user should copy all data from a CSV file and paste it into the application to run the system. The other difference between the visualization application and the KNX ultimate's application is that it does not show the data in a tree view structure. It just deploys it in the system and runs the program to check for errors [22].

3 Method

3.1 Language and Tools

The application will be in C# to be able to connect the application with the .NET driver. It just needs to be in Windows then the company only uses Windows. Another advantage of C# is that the graphical user interface (GUI) part is "built-in". It is possible to use other languages also e.g C or C++ and these have advantages that have a fast execution time and is a lightweight language. C and C++ are cross-platform programs through C & C++ have separate GUI libraries therefore it is suitable to use C# to program the application. Visual studio 2019 and C# will be used to program the application because it is easier to launch the code and has good tools to debug and integrated with the dynamic link library (DLL) file[23]. DLL files is a library containing code and data that can be read same time from other programs. Windows uses DLL files for much of the functionality and other programs use it for data and functions for their programs to separate main code and general code that can be used for other part of the program. The application needs the DLL files to get the function to connect and disconnect and communicate with KNX systems. When sending data to the bus the KNX devices don't understand the data that the user sent so "DPT.dll" file is going to convert the data from to hex decimal[24].

3.2 Procedure

In this section, the process to achieve the goal is explained. To achieve the goal in this project, it needs to use different tools and materials to test the work. The company's materials are KNX virtual, a virtual system of KNX devices, ETS software that loads the KNX virtual and Falcon Driver.

Drivers are available from the company that will be connected to the KNX system that allows the application to communicate with KNX compatible products that will be tested. The company has created documentation of how the driver will communicate with the code. While testing the application, some files are being used by the application that were received from the company and to test the system virtually. The software that the company provided is KNX virtual and engineering tool software(ETS).

Engineering Tool Software (ETS) is a software to design, configure smart homes, buildings and control installations that build for KNX devices. ETS is used to program KNX components, but it is also used to program the KNX virtual. Another functionality of ETS is that the users can configure a KNX system virtually, write data to the system, and read the feedback from the KNX bus system. ETS are used in test phase to load the KNX devices in the KNX virtual that can be used with the application.

KNX Virtual represents a KNX system that can be configured with devices for test processes like the application built in this project like connection, disconnect, send data and listening on the KNX system. With KNX virtual the connection can be detected and the user can test if the system works as requested. The application connects to the KNX virtual using KNXnet/IP that mentioned in section 2.1.5. The application will send data to the KNX devices that can be seen in the KNX virtual when a light is on, off or dim on. ETS is used to test KNX virtual by sending data to the system and receiving correct feedback and Xunit is used to check whether the application sends valid data or not.

Xunit.net framework is a test automation tool, and it's available in many languages to efficient test with automated tests written in the language the developer will test. Writing in the same language allows the developer to focus on the crucial task and write tests on the code phase in minutes. Xunit tries to simplify and code more efficiently even in the code phase. Xunit tests are written in a separate file that can be written with many tests and then easy to test with a simple click of a button and correct the errors. The tests are made easy to encourage developers to try more frequently and reduce the cost to test the code. The "bi-product" of all the tests is overall quality code and better execution[25].

3.3 Tests

The test will ensure that the application is working like intended and follow the requirements from the company.

3.3.1 Connection test

In this section, the connection using KNXnet/IP, which was discussed in section 2.1.5 will be explained. This test will provide the connection and loading confirmation. The connection test starts with inserting the IP address in figure 7. KNXnet/IP and KNX virtual are going to use two methods to communicate, the first one is IP tunneling, and the second is IP routing (read 2.1.5). The connection test will check if the application can communicate with the KNX devices like intended. It will even check if the connection is established and can begin to communicate with the devices on the bus. Connection is checked with KNX virtual there connection with the bus can be watched in real-time. Checking if the application can send and listen on the KNX devices, KNX virtual shows what all the devices have done and which value every virtual component has. The test aims to test the connection between the application and the KNX bus system using Falcon driver, which is one of the requirements in this project. When the bus system is connected or disconnected to the application though it should send a feedback to the application in order to approve that the connection or disconnection was successful.

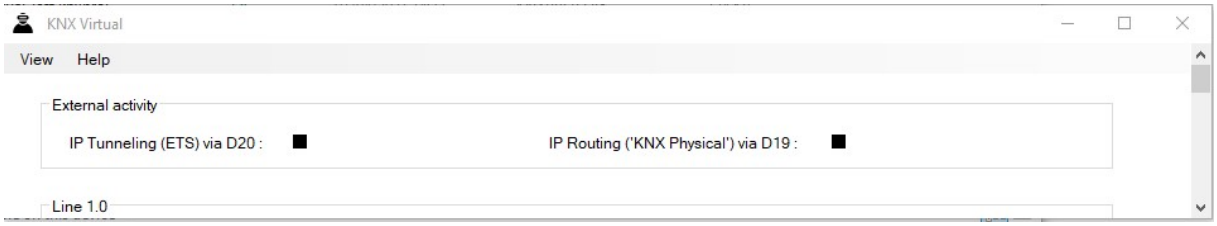

Figure 7: Tunneling

3.3.2 Design test

The design test will test the design and the requirements from the company on the application with design case techniques. The techniques can test "easy to use", how data can be handled and see if all data is correct from the start to the end with the help of Xunit. Xunit uses a separate test file that can test the application code without interfering. These techniques do a range of tests that will get a width coverage. The tests that will cover the design part are error guessing, equivalence partitioning, boundary value analysis, decision table, and state transition technique.

Error guessing is a technique to guess what the user could do to give back an error. This technique aims to find the human error from a person who has not programmed the application and knows what's right. The test makes the application more robust and makes it easy to modify the values from the loaded csv file. For example, it has a text box, and the developer knows only numbers that work in the box, but users can write whatever they feel like no one can say what's wrong or guide them. The users could write text instead of numbers.

Equivalence partitioning is a class case test where a significant interval is divided into classes. There is one class; get one test each and then see if it passes. The Equivalence partitioning tries to minimize the trials but still has good coverage over the whole interval[26]. Sending data to the bus has big intervals that need to be divided into smaller parts to speed up the whole process of testing if the interval is functional; that was a requirement to handle on/off and 0-100% values. When sending data the communication needs to be set to point-to-point connection-less to tell the device it's only one-to-one transmission over the bus at this moment.

For example, if it has an interval between 1-200, it's too many tests if the whole interval was tested one by one. Instead, divide the interval into classes like -101 → 0, 1 → 100,
101 → 200, and 201 → 300 and test every case if it passes. From every class, and choose one value to test, and if that passes, then the whole class passes.
Example:

```
            /*int data = is a value that send in and will be transfer to dpt
            conversion and send to the buss and it can only be 1 or 100 */
            // test (-50)-0
            int data = -20;
            Assert.False(senddata(address, status, data, dptstatus));
            //test 1-50
            data = 46;
            Assert.True(senddata(address, status, data, dptstatus));
            //test 51-100
            data = 79;
            Assert.True(senddata(address, status, data, dptstatus));
            //test 101-150
            data = 149;
            Assert.False(senddata(address, status, data, dptstatus));

```

Listing 1: Xunit example

The boundary value analysis technique finds weaknesses in the boundary values. It tests right before and after the value, and the value ensures that the boundary works like intended[26]. The boundary values in all sent data intervals need to be checked because it's really important to send the right data into the DPT converter that sends the data to the device.
Example:

```
            /* int data= data is a value that send in and will be transfer to dpt
            conversion and send to the buss and it can only be 1 or 0 */
            //test -1
            int data = -1;
            Assert.False(senddata(address, status, data, dptstatus));
            //test 0
            data = 0;
            Assert.True(senddata(address, status, data, dptstatus));
            //test 1
            data = 1;
            Assert.True(senddata(address, status, data, dptstatus));
            //test 2
            data = 2
            Assert.False(senddata(address, status, data, dptstatus));

```

Listing 2: Xunit example

The decision table technique checks for failures in the application's conditions and how many to test, is two power of numbers of cases. MC/DC (modified condition/decision coverage) is criterion for code coverage for decision tables where all possible outcomes of the software are tested[27]. To check if the generated information status is working, the test will take the first value from KNX virtual and compare it to the feedback value that's come out from the bus and compare them and check all of the outcomes. Before checking, the table needs to be generated to see what's needed to be tested and what's not needed(duplicate tests). For example, the application has three conditions. Then it will be two powers of 3, and that's 8. so it has a whole eight tests for three conditions.

Example:
Condition 1: person over 40 = 15%
Condition 2: person under 40 = 10%
Condition 3: person with coupon = 30%
Every column will be a test to test.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Condition 1	T	T	T	T	F	F	F	F
Condition 2	T	T	F	F	T	T	F	F
Condition 3	T	F	T	F	T	F	T	F
Result	X	X	45%	15%	40%	10%	X	X

Table 2: Example

The state transition technique ensures that every state does what it's intended to achieve. The goal here is to visualize how the system should work and test every outcome that needs to be tested[28]. Flow diagram for all the functions there it can test to load a structured csv file and follow the diagram to see if it's working correctly, and go to all states where it needs to go to an error or load the file to the application. Flow diagram shows if the connection to the KNXnet/IP is working and when could disconnect if it's connected.

4 Implementation

The implementation of the application has been divided into three parts. 1. Design implementation. 2. Data storage implementation, 3. KNX connection. As seen in figure 13.

The graphical part of the application was implemented in C# and visual studio 2019. The application has one menu bar, three data fields, and two buttons. The menu bar has two parts, one for uploading and saving, and the second one gives information about the company and the application. The first part of the menu bar includes open and save options for uploading a CSV file from the computer, and the save option is for keeping the feedback from the KNX system in text format. Data fields are for uploading and receiving data. The first field is to show the tree view rooms and plans. The second field is to open value boxes and information text boxes, and the third field is to give users feedback, and feedback comes from the KNX system. The buttons are for connection, and sending data to the KNX system also disconnects. The first click on the check button, which shows a configuration box for connection to the KNX system where the user can select the type of connection (USB or tunneling IP (KNXnet/IP)), but before that, the user should fill in the data and then click OK because of an empty input error. After sending data, the application will receive feedback that the main reason behind the application implementation. The feedback will appear in the feedback box, which is an editable text box. The disconnect button is for disconnecting the connection between the application and the KNX system.

Data is stored in a double linked list called Room, and each node or each Room has been linked to a new double linked list called RoomItems. When opening the room/section, all the items are running through linked list. They are saved when and sorted out the essential parts shown in the value section on the right. The linked list was chosen because of it's easy to increase/decrease factor. The cons for the linked list are that it takes more memory because it holds the addresses to the previous node and the next node. The linked list is created like in figure 8. They get names given DPT values depending on what type of device it is, an example in section 2.3. Section with the values is where every parameter can be set to a value sent to the device. To send the values to the application, throw every row to see if the value is set or not. If it's not set, ignore the row and go to next to speed up the processes that use multi-threading to make it faster. Multi-threading uses four cores to make it go quicker and utilizes more cores to make an application do tasks simultaneously. If not used, multi-threading the application only uses one CPU-core of the computer and can stress the CPU-core and speed up the process to load the rooms. To send to the bus all data is converted from a decimal number to a hex value and sent to a driver file called DPT.dll that will transform the hex value and send back the value and the bus can read those values. KNX devices send back what has happened and to catch all the information. The application has an interrupt to send all the information to the feedback box where the user can watch what has happened and when they did it. When the program closes or wants to change the KNX system or close the program, they need to disconnect with a button (Disconnect) Figure 13.

To connect the application to the KNX system, the user needs a driver called KNX eteC Falcon runtime v2.2. The function that opens the connection after installing the driver is called getConnectionInit(). getConnectionInit() includes a class called The FalconConnection, which contains all data of a connection, including the user-friendly name, flags, EDI, ID, and the connection parameters. FalconConnection has two parameters used in the getConnectionInit() function, the guidEdi: EDI ID to open the connection, and wszParameters which is the string to be used to open the connection.

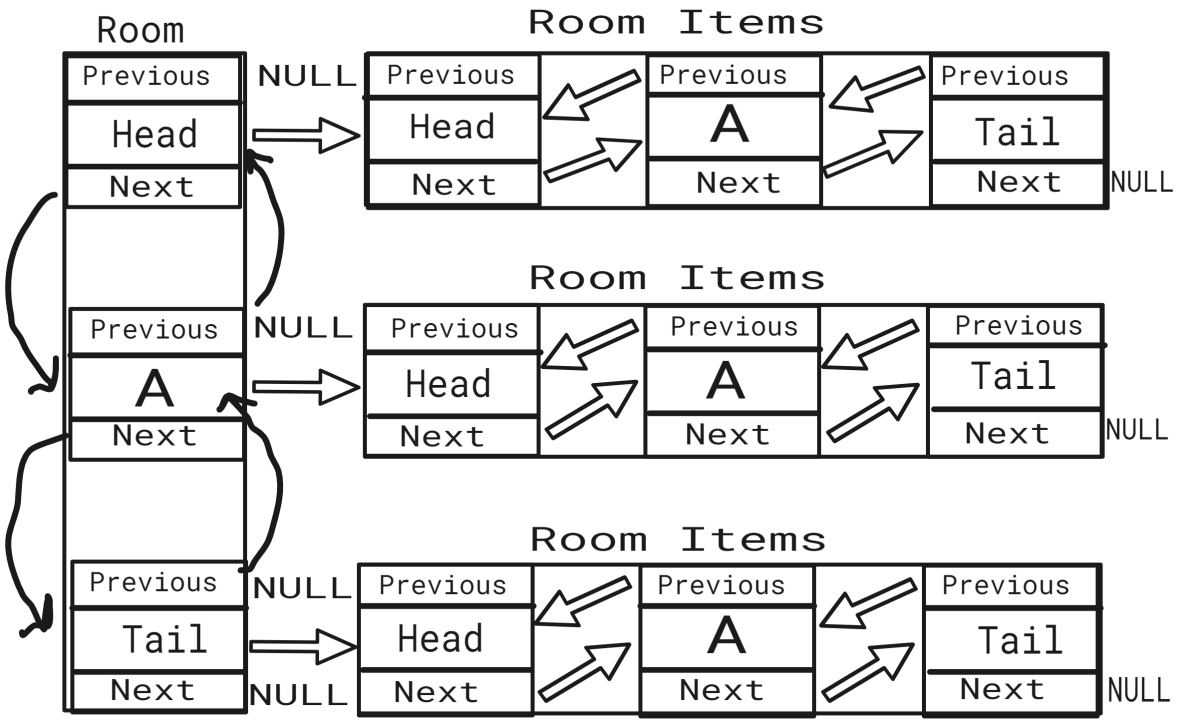

Figure 8: Linked list structure of data storage [29]

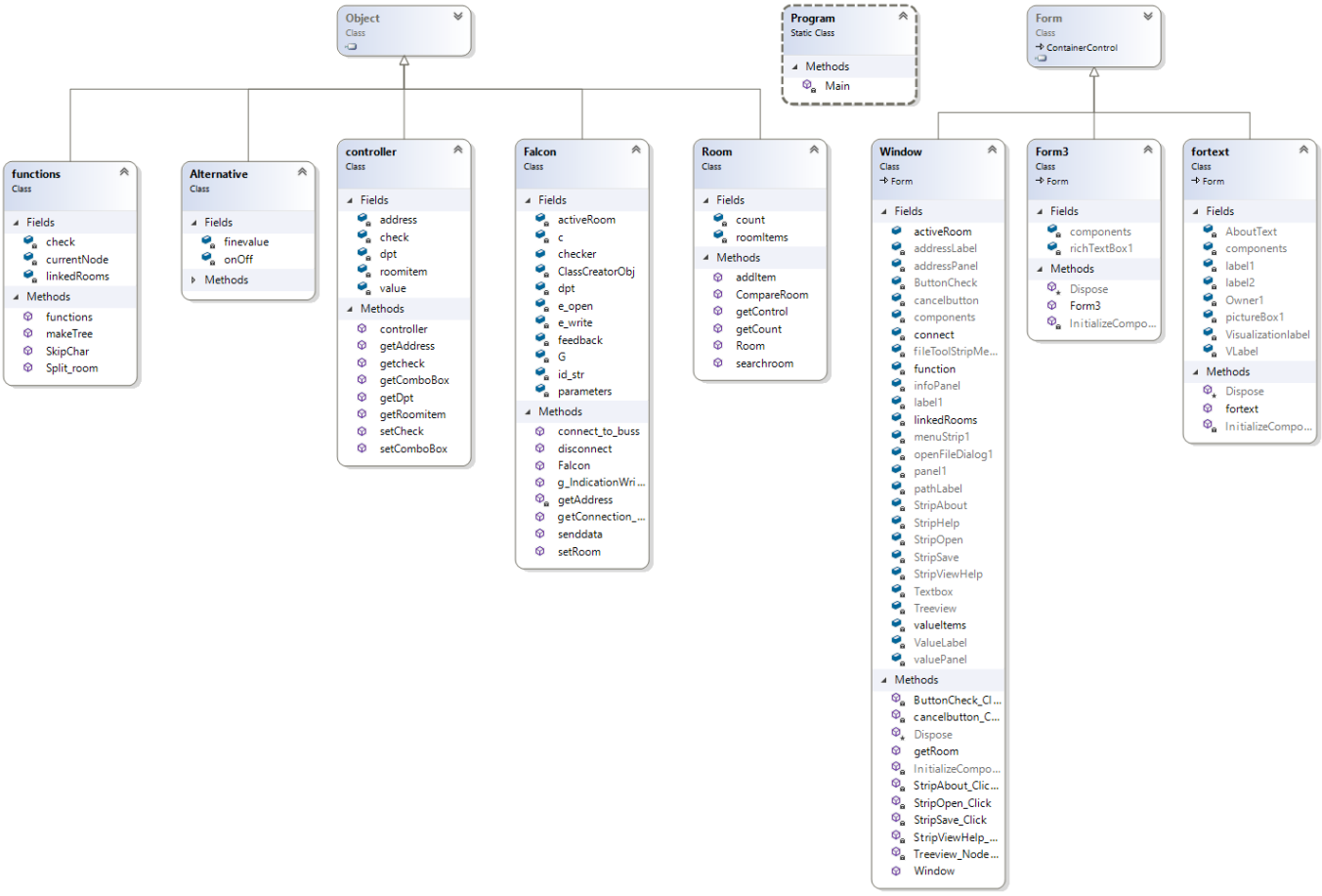

Figure 9: Unified modeling language diagram

```
            if(Guid = empty or Parameters = empty){
              do not open the connection;
              // Guid is the EDI ID to be used to open the connection.
              // wszParameter is the parameter string to be used to open the connection
             }

```

Listing 3: If statement to check for open connection

If both of them are null then there would not open a connection. But if those parameters are not null, then the Edi manager can be used to get the Edi Guid which corresponds to the selected port number and Edi type and return a Guid.

```
             else
                 {
                     The Guid ID (gu_id) = Guid EDI ID; //

                     parameters = Empty string
                     for (looping through the wszParameter)
                     {
                         Parameter and wszParameters is going to be converted to characters;

                     }
                     id_str = Convert the Guid to string again;
                 }
                 return return id_str and Parameters;
                 // id_str = "13bec8e3-969d-4604-83f4-5ebf418e54f6";
                 // parameters = "Name='<New...>';IpAddr='127.0.0.1';Port='5000';NAT='off
    '\0\0\0\0\0\0\0\0\0\..."

```

Listing 4: How guid is handled in connection function

Id_str is the address of KNX virtual, all KNX systems that communicate using KNX tunneling have a unique address.
Parameters is for storing the information about connection. The code are for opening a connection between an application and a KNX bus system. To open this connection, it needs one Guid ID or a key which is a string. If we do not have the ID which the Falcon driver will generate, the connection will not happen, but if we have just one of them, it will take the string and convert it to the character, and then stop it in other variables and return it as feedback.

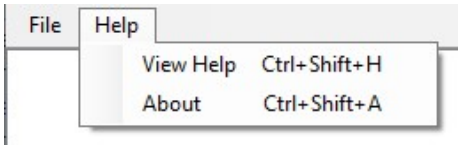

Figure 10: Help

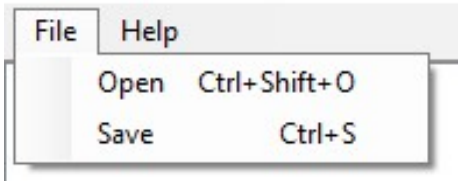

Figure 11: File

5 Result

The result in this paper has been divided into three results, design result, which describes the result of the graphical part with the company's requirements, connection result describes the connection between the application and KNX bus result, and the final result is the addition of design and connection result.

5.1 Design Result

The company asked for an application with a complimentary design choice that can open a CSV file in an organized way. As a result, the application includes a menu bar with two options: open a CSV file and save the feedback in text format figure 11. Two to help the user with some given information about the company and how to use the application step by step figure 10. The application has three fields. The first one is the file field, where users can see the opened file in the tree view structure with the select-able capability. The second field is the data field, where users can see the information about the selected section, the address of the section, and a combo box for each section to insert values to send those values to the KNX bus system. The last field is the feedback field, where the application receives feedback from the KNX system and prints it in a text box where the users can save their feedback in text format. The design has two buttons, but they are used to connect and disconnect to the KNX bus system see figure 13.

5.2 Connection Result

The connection result starts with the check button. The check button starts the connection and sends data to the KNX bus system, see figure 12. The application connects to KNX eteC Falcon runtime 2.2v, which is the driver, and then the driver connects with the KNX bus system; in this case, it communicates with the KNX virtual and starts sending data. After sending data, the bus system sends feedback to the application, as mentioned in the previous section. The disconnect button cancels or removes the connection line between the application and the driver. After connecting and disconnecting the system returns a message which confirms the connection process.

5.3 Final Result

The final result is about the addition of the design result, communication result, and the company's requirements for application. The final result of this project is that an application has been implemented with the following functionality.

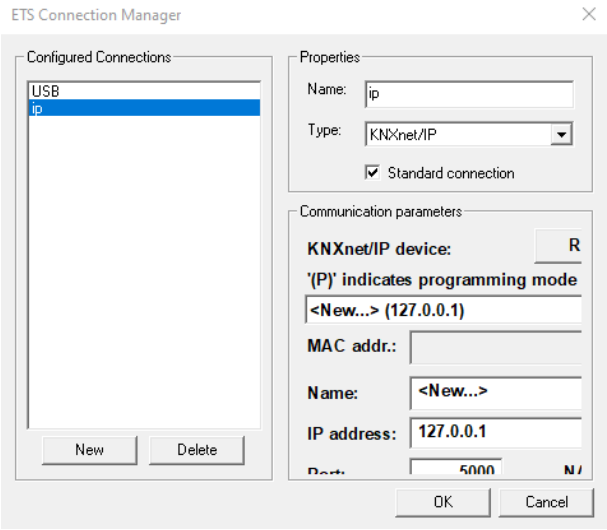

Figure 12: Start of connection using IP address and port 5000

- It loads a structured list generated from ETS.
- It creates a tree structure over constituent groupings.
- It lists controls and inputs for commands by the selected group.
- It creates a connection to the KNX bus and sends specified commands.
- It generates informative status back about what has been done during writing and reading to or from the bus system.

The Application is presented in figure 13.

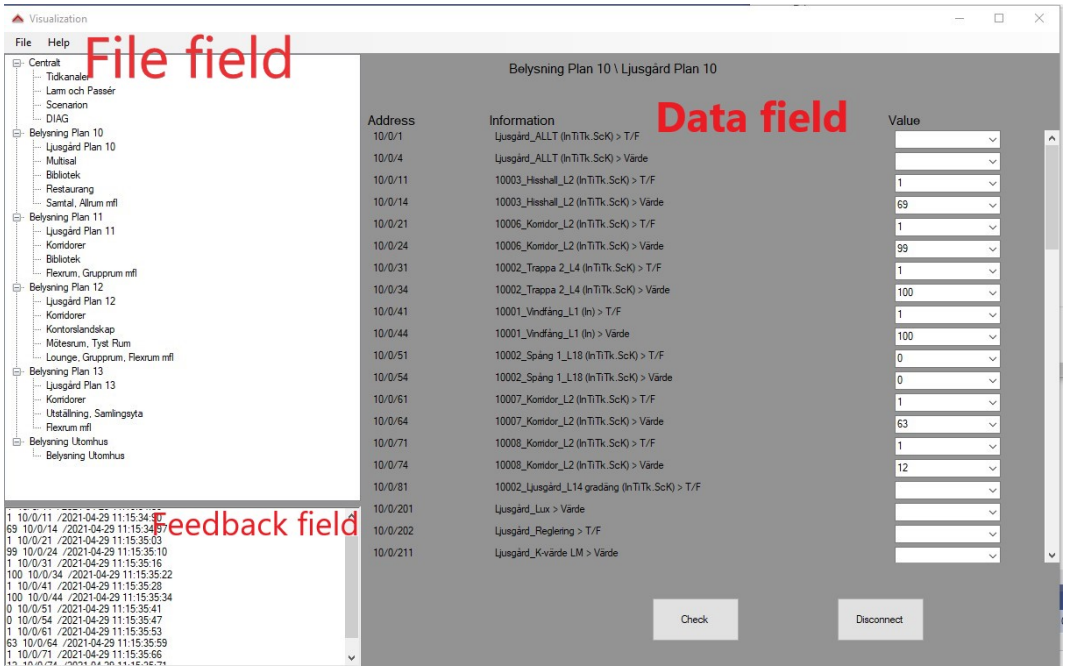

Figure 13: Application

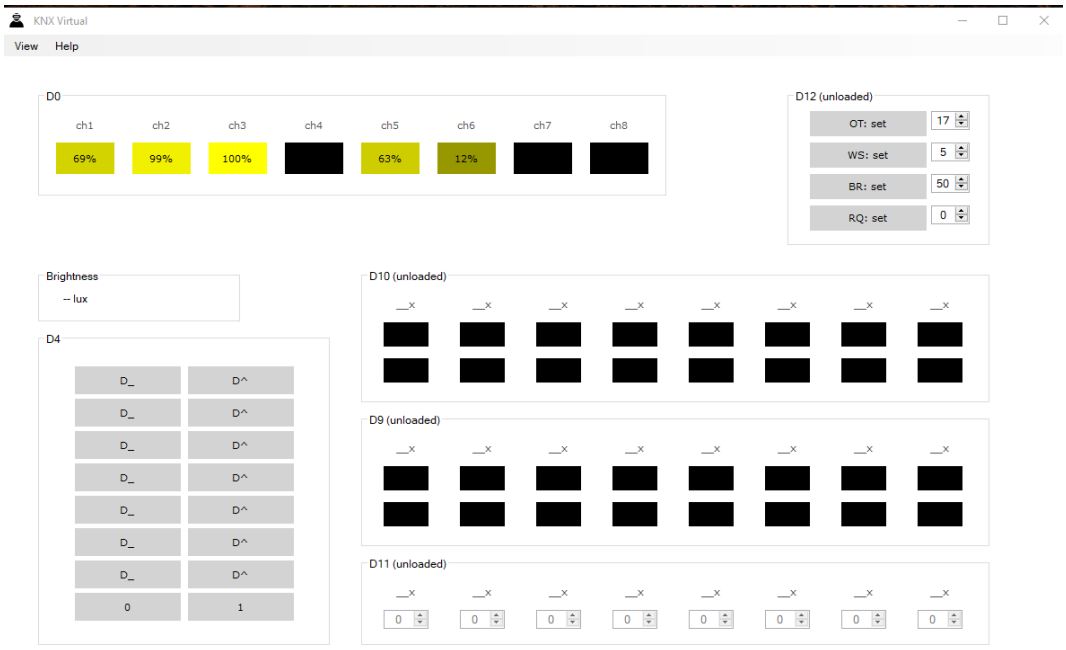

Figure 14: KNX Virtual

5.4 Result analysis

In this section, the effect of the tests mentioned in the method on the result will be explained. The result is fulfilled as the company requested, but many errors were in the application that could disturb the process. The design test (3.3.2) was used to test all errors that users could find, for example, to turn on and off a dim light. The value for this operation is going to be just 1 or 0. Still, if the user inserts a value bigger than 1 or smaller than 0, the application must give an error message back to remind users of the boundary in the application. Error guessing tests help us to find all possible errors caused by the users.

To connect the application to the KNX bus system, the user needs to connect it manually using a check button. This is not an issue for the company that connects the application to the bus system manually, but it is still a bug in this application. The application should be connected to the bus when the user opens the application instead of connecting it manually. This is an issue that needs attention because it makes the result less perfect.
The company tested the application, but unfortunately, we did not get this opportunity to test it ourselves. The only tests that we did on our application on a building system were on the dim light button in different levels of a building using KNX virtual. In figure 13 and 14, the data are sent from the application to the KNX virtual (3.1) using KNXnet/IP and the KNX virtual receives the data, and the dim lights have been arranged using the inserted data. Then the KNX virtual sends feedback with the date and time of operation. The problem is that we got only one loaded task to test, which is the dim light. We did not receive a full version of tasks to test all possible components such as sensors, lights, and other electrical components in KNX virtual. The mentioned problem makes the result less complete because we do not know how the application will work with other components.
The issues that are mentioned above affect the result. If we had more options and more tools to test applications ourselves on electrical components in a building, the product would be more complete.

6 Discussion

In this section, we discuss the final result, how it helps the company with economic aspects, and the future plans for the application. The result in this work depended on knowledge and information about the KNX system and how its bus system works. The result could be much better, and the application could be more functional with several advanced features if we took the course about the KNX system or found more information and more details about the driver and how an application should be connected to a bus system. The KNX system is not that popular, and there is a leak of an information about the system and a leak of application that works with the ETS software, which made our work much harder to find and collect knowledge. If there were more similar projects that would be easier to compare our application with others for improvement purposes.

To achieve this result, we had a timetable that divided the application process into sub-tasks and worked one at a time. Sometimes we had to work with two parts simultaneously, for example, working with the written part and collecting information about the KNXnet/IP. It took a shorter time to implement the graphical part, but the connection part was challenging. The timetables were arranged from the beginning of the course, but the end dates depended on meeting with the company and feedback from the supervisor. After the first seminar, we had to change it because we needed to divide the project into much smaller parts to achieve the goal. The goal was to finish the course before May 2021, but we could not complete the process as we planned due to a delay from the company; instead, we were finished with the project on the 2nd of May 2021.

6.1 Economy

One of the requirements is to minimize the test time for the company and then result in lower cost for the companies that hire them to do work. With lower cost makes the company more attractive on the market so then do more jobs, the competitors will have to push forward with their technology to lower their prices. This chain reaction will create development to go forward and speed up testing, throw more digitized tools to test home automation with KNX system that is now slow and makes it more prone to human error that can drag more cost in the end. To speed up and with the same or higher standards, throw testing to minimize human errors.

6.2 Future works

The application connects and sends data to the KNX bus, but only the bus sends data back that the device has done. In the future, will all data be saved in a log file so the program can check and make a green light to the log file and make it easier to see if it's good or bad news about the KNX system. With the log file, the user can see what have been wrong, and then they can decide how to solve it. This solution can decrease the testing much more after this implementation we have done so far. The solution will be more precise and more beneficial for the client that uses the program. The solution to connecting to the KNX system happens every time the user clicks on the check button. This faulty solution could be solved when starting the program in the beginning. You connected already, so you don't need to connect under the run-time of the program.

6.3 Technical dependency

In this part of the discussion, the weaknesses of the result are discussed. The issues are re-connection, taking time to load a bigger CSV file, and resizing problems. The application has a connection where the user needs to connect the application KNX virtual or to a KNX bus system by a button at the beginning of the process. It takes time to open the connection window every time the user starts the application. The application should be connected to the KNX system automatically when the user starts the application. Another issue with the application is that it takes a few seconds more to load a bigger CSV file because of while and for loops used in the code. The last issue is that the application window is fixed in size. The user can not adjust the size of the application by minimizing and maximizing options.

References

[1] KNX basic PDF [Internet]. KNX.org; 2020 [cited 2021 Feb 1]. Available from: www.knx-professionals.de.

[2] Kraus N, Viertel M, Burgert O. Control of KNX devices over IEEE 11073 service-oriented device connectivity. In: 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS). vol. 1; 2020. p. 421–424.

[3] Hochschule Mannheim Fak. Building Automation,2009. Paul-Wittsack-Straße 10 68163 Mannheim(Germany): Hermann Merz , Thomas Hansemann,Christof Hübner; 2009.

[4] Tanenbaum AS. Computer networks. 5th ed. Upper Saddle River, N.J: Prentice Hall; 2010.

[5] ISO/IEC 14543-3-10:2012 [Internet]. International Organization for Standardization; 2012 [updated 2012 Mar; cited 2021 Feb 26]. Available from: www.iso.org.

[6] The benefits of KNX [Internet]. KNX association; 2021 [cited 2021 Feb 28]. Available from: www.knx.org.

[7] Lee WS, Hong SH. KNX — ZigBee gateway for home automation. In: 2008 IEEE International Conference on Automation Science and Engineering; 2008. p. 750–755.

[8] López Aguilar AA, Navarro Tuch SA, Bustamante Bello MR, Izquierdo Reyes J, Curiel Ramirez LA. Interpretation and Emulation for Telegrams of the KNX Standard on MATLAB Simulink. In: 2018 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE); 2018. p. 129–133.

[9] Home Assistant [internet]. Home Assistant; 2021 [updated 2021; cited 2021 May 1]. Available from: www.home-assistant.io.

[10] loxone [internet]. loxone; 2021 [updated 2021; cited 2021 May 1]. Available from: www.loxone.com.

[11] Vanus J, Cerny M, Koziorek J. The proposal of the smart home care solution with KNX components. In: 2015 38th International Conference on Telecommunications and Signal Processing (TSP); 2015. p. 1–5.

[12] Springer. Building Automation, Communication systems with EIB/KNX, LON,and BACnet. Paul-Wittsack-Straße 10 68163 Mannheim Germany: H.Merz, T.Hansemann, C. Hubner; 2009.

[13] KNX.org. KNX System argument. Paul-Wittsack-Straße 10 68163 Mannheim Germany: KNX association; 2000.

[14] López-Aguilar A, Bustamante-Bello R, Navarro Tuch S, Ramirez-Mendoza RA. Communication system development for emotional domotics interactive space ESCI-Journal. International Journal for Interactive Design and Manufacturing (IJIDeM). 2020 06:1–10.

[15] Seifried S, Gridling G, Kastner W. KNX IPv6: Design issues and proposed architecture. In: 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS); 2017. p. 1–10.

[16] Cavalieri S. Estimating KNXnet/IP routing congestion. In: 2011 IEEE International Symposium on Industrial Electronics; 2011. p. 1200–1205.

[17] Cavalieri S. Analysing congestion in KNXnet/IP communication system. In: 2011 IEEE International Conference on Industrial Technology; 2011. p. 244–249.

[18] Nagayama T, Moinzadeh P, Mechitov K, Ushita M, Makihata N, Ieiri M, et al. Reliable multi-hop communication for structural health monitoring. Smart Structures and Systems. 2010 07;6.

[19] Elprojektering [Internet]. Tekniska byrån; 2020 [updated 2020 Aug 12; cited 2021 Feb 26].

[20] Seifried S, Gridling G, Kastner W. KNX IPv6: Design issues and proposed architecture. In: 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS); 2017. p. 1–10.

[21] Lee WS, Hong SH. Implementation of a KNX-ZigBee gateway for home automation. In: 2009 IEEE 13th International Symposium on Consumer Electronics; 2009. p. 545–549.

[22] KNX ultimate [Internet]. Supergiovane; 2021 [cited 2021 Mars 03]. Available from: `https://github.com/Supergiovane/node-red-contrib-knx-ultimate`.

[23] How to create GUI in C programming using GTK Toolkit [Internet]. geeksforgeeks; 2021 [cited 2021 Feb 24]. Available from: `http://geeksforgeeks.com`.

[24] Dynamic link library(DLL) [Internet]. microsoft; 2021 [updated 2021 May 25; cited 2021 Feb 26]. Available from: `www.microsoft.com`.

[25] Chaczko Z, Braun R, Carrion L, Dagher J. Design of unit testing using xUnit.net. In: 2014 Information Technology Based Higher Education and Training (ITHET); 2014. p. 1–9.

[26] Bhat A, Quadri SMK. Equivalence class partitioning and boundary value analysis - A review. In: 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom); 2015. p. 1557–1562.

[27] Hemmati H, Arefin SS, Loewen HW. Evaluating Specification-level MC/DC Criterion in Model-Based Testing of Safety Critical Systems. In: 2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP); 2018. p. 256–265.

[28] Gupta A, Mishra N, Kushwaha DS. Rule based test case reduction technique using decision table. In: 2014 IEEE International Advance Computing Conference (IACC); 2014. p. 1398–1405.

[29] Prentice hall. Data structures With Java. Upper saddle River, New Jersey: William R. Topp, William H. Ford; 2005.

My name is Fredrik Kortetjärvi, and I study computer science and engineering for my third year.

My name is Rohullha Khorami, and I study computer science and engineering for my third year.

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se