

CREATING A USER INTERFACE FOR HOME AUTOMATION



Bachelor's thesis

Automation Engineering

Valkeakoski, autumn 2017

Duy Hung Tran

Automation Engineering
Valkeakoski

Author	Duy Hung Tran	Year 2017
Subject	Creating a user interface for home automation	
Supervisor(s)	Timo Viitala	

ABSTRACT

In this thesis, the author provides a brief theoretical background to home automation and examines how the user interface of a home automation system should be designed. The thesis was inspired by Jukka Aarnio (Dr. Tech.) and carried out under his supervision. The topic of this thesis is both practical and controversial in the field of engineering in general as home automation is exponentially developing. The topic was suitable for a Bachelor's thesis because the depth of knowledge needed was reasonable. This was also an advantageous project for the author for gaining more experience, particularly, in the field of home automation.

The final objective in the project was to provide an example of a web-based user interface application which would include some basic features of a home automation system: control of lighting, energy management and status indication. Moreover, there was a mode feature which sets household appliances into pre-set modes. Additionally, the application was programmed to have a security layer to protect the interface from unauthorized users. Finally, the interface was to be controlled remotely through WLAN to give users comfortable experience while manipulating all the electronic devices in the house.

In order to achieve the goal, Indusoft Web Studio software was chosen as a beneficial application for creating the user interface. Therefore, most of the practical work for this thesis was performed on Indusoft Web Studio v8.0.

Keywords Home automation, user interface, Beckhoff PLC, Indusoft Web Studio.

Pages 41 pages including appendices 6 pages

CONTENTS

1	INTRODUCTION.....	1
2	THEORETICAL BACKGROUND	2
2.1	History of home automation	2
2.2	Introduction to Programmable Logic Controller	3
2.3	Design of user interface	5
2.3.1	Placing users in control of the interface	6
2.3.2	Reducing memory load of users	8
2.3.3	Making the interface consistent.....	9
3	IMPLEMENTATION METHOD	11
3.1	Selection of software	11
3.2	Description of user interface	13
4	CREATION OF USER INTERFACE	14
4.1	Interface design	15
4.1.1	Header.....	15
4.1.2	Navigator	17
4.1.3	Main area	21
4.2	Studio Mobile access.....	29
5	INTEGRATION BETWEEN TWINCAT AND INDUSOFT WEB STUDIO	32
6	CONCLUSION	34
	REFERENCES.....	35

Appendices

Appendix 1 TWINCAT PROJECT CODE

1 INTRODUCTION

Dr. Tech. Jukka Aarnio, the commissioner of the thesis, has a plan of building a home automation system for his own needs. He is developing an automation system to his house where he can control the lighting, energy management and security of the house by Beckhoff PLC. Aiming to make the control activity of the system easier and more convenient, a user interface for the system was needed and it was the main topic of this thesis project. The user interface runs on a web-based application and by that application, the user can control electronic devices in the house remotely through a wireless network.

The thesis is divided into two main parts. In the first part, the author provides a theoretical background to the thesis topic including the knowledge of home automation, PLC and user interface design. In the second part, an example of the user interface is described as a practical work of the thesis. In the user interface, there are some main features which include: lights control and room temperature indication. There are also some pre-set modes for the house in which the lights and temperature are set to a certain value. For safety reasons, the interface also has a security system which requires a username and password for the user to get access to the interface.

In the software market, there are several options, such as Altia Design from Altia, Indusoft Web Studio from Wonderware by Schneider Electric, GUI Design Studio from Caretta Software, etc. from which the author chose the best option for creating a user interface. These are all programs used for designing automation systems. However, Indusoft Web Studio was chosen as the implementation method of the thesis. In chapter 3.1, there is a comparison between Indusoft Web Studio and other software and an explanation on why Indusoft Web Studio was chosen.

After the user interface had been created, the next task was creating an integration between the interface and the PLC system. When the integration part was finished, the interface could remotely control the status of real objects in the house through the PLC system.

Now that the thesis project is completed, the commissioner will apply the results of this practical work to his own home automation system.

2 THEORETICAL BACKGROUND

2.1 History of home automation

Home automation or smart home is a high-developing branch of building automation. It is basically a house where lighting, HVAC (heat, ventilation and air conditioning), security and fire systems are connected and controlled by using BMS (building management system) central point.

(Smart home 2016.)

At the beginning, home automation was just an abstract idea. There was no concrete structure or construction explicitly called “home automation”. The time line which provides the development history of home automation is given as follows:

- 1901-1920 – The invention of home appliances: The first achievement of mankind in home appliances was an engine-powered vacuum cleaner in 1901. After that, many other appliances were invented, such as refrigerator, washing machine, iron throughout twenty years.
- 1966-1967 – ECHO IV and Kitchen Computer: These inventions were the very first smart devices which can do the housework. ECHO IV can compute shopping lists, control the room temperature and turn on or off home appliances while Kitchen Computer can store cooking recipes. However, because of poor technology at the time being, they were not commercially sold.
- 1991 – Gerontechnology: This is a combination of gerontology and technology with the aim of making the lives of elderly people easier by the help of modern devices.
- 1998-Early 2000s – Smart Homes: Home automation started to become more and more popular in the society. The emergence and exponential growth of different technology lead to a sudden financial decreasing of smart home. Therefore, home automation became a viable technology for consumers and many of its application started to appear on the market.
- Today’s Smart Homes: As the living-standard has reached to a high level, solution for home automation focuses more on security and energy saving. A smart home must be sustainable so it helps us to make sure that we don’t use energy unnecessarily and it also has a strong security system.

(Hendricks 2014.)

Technology has never stopped developing. Therefore, home automation is still extremely potential in the future.

2.2 Introduction to Programmable Logic Controller

Basically, PLC is a computer-like controller which is used to control the manufacturing process. A PLC system provides high reliability control, simplicity for programming and maintenance. A PLC can work with a wide range of task including analog and digital input/output interfaces, data conversion, signal processing, etc. However, despite the importance of PLCs in industry nowadays, it is surprisingly quite a young invention. Before the 1960s, relays were the only way to control machinery. Relays use coils which are energized and produce a magnetic force to change a switch's modes which are in ON and OFF positions.

Unfortunately, at that time many control engineers had encountered some disadvantages of using relays. The worst disadvantage was reported in "History of PLC" as: "All these relays had to be hardwired in a very specific order for the machine to work properly, and heaven forbid if one relay would have an issue, the system as a whole would not work. These machines had to follow a strict maintenance schedule and they took up a lot of space. Then what if you wanted to change something? You would basically have to redo the entire system" (History of PLC n.d.). Therefore, the need for a more effective solution had arisen. On New Year's Day 1968, the PLC was introduced for the first time to the whole industrial world. After that day, relays were gradually replaced by the PLC system in factories. Figure 1 is an example model of a PLC.

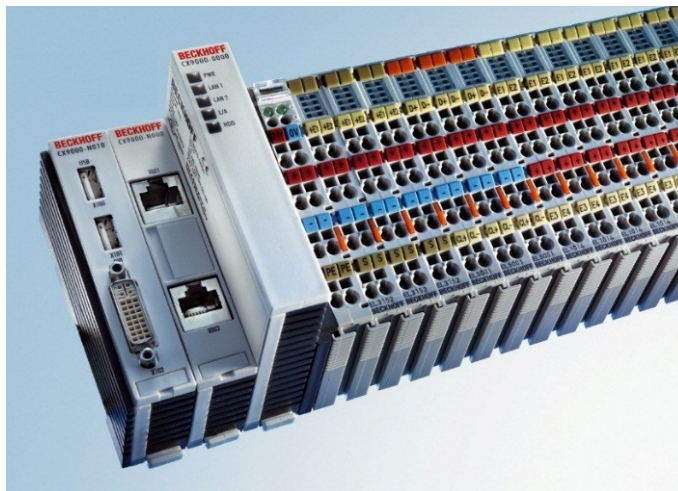


Figure 1. Beckhoff CX90x0: Product overview (Beckhoff n.d.).

Some of the advantages which are contributing to PLCs replacing complicated relay systems are listed as the following:

- Flexibility.
- Faster response time
- Less and simpler wiring
- Solid-state (no moving parts)
- Modular design – easy to repair and expand
- Able to handle much more complicated systems
- Sophisticated instruction sets available

- Allows for diagnostics “easy to trouble shooting”
- Lower expenses.

(Introduction to PLC's 2006.)

Generally, the hardware of a PLC as seen in Figure 2, includes seven different parts: inputs, outputs, a central processing unit, a programming device, a programming memory, a power supply and two layers of optical isolation. Inputs are terminals which receive information from sensors, buttons, switches, etc. On the other hand, outputs are terminals which send signals from the PLC to actuators like motors, indicating lights, cylinders, etc. Input and output modules can convert voltage to any appropriate signal used in the interface. The central processing unit (CPU) is usually a microprocessor which is used to execute logic and arithmetic operations in a program written in the PLC. It is also able to command the PLC to perform internal diagnostics. The program is written by a programming device. The programming device is a digital device, usually a computer, where there is programming software compatible with the PLC system. For example, the programming software for Beckhoff and Siemens PLC are TwinCAT and SIMATIC STEP 7 respectively. After being written by a programming device, the program is stored in the program memory and the CPU reads the program from there. Optical isolation or opto isolation is a layer which transfer electrical signals between the CPU and the input/output modules by using light and this can prevent high voltage damage to the CPU.

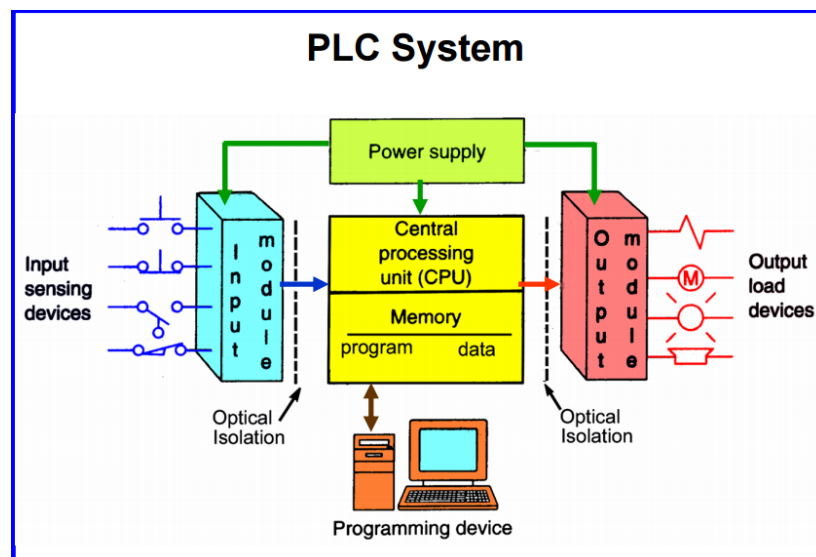


Figure 2. PLC system (Introduction to PLCs 2006).

According to IEC 61131-3, the international standard for the programming languages of programmable controllers, there are five languages which have been approved as official programming languages for PLCs:

- Ladder diagram (LD)
- Sequential function charts (SFC)
- Function block diagram (FBD)
- Structured text (ST)

- Instruction list (IL).

(PLC languages n.d.)

One of the benefits which the standard provides is that one PLC can use multiple languages. Hence, the designer can decide which language is the most suitable one for each particular task.

Last but not least, scanning operation in a PLC software is a critical feature which also makes PLCs popular nowadays. A scanning operation is a loop of actions executed repeatedly. The loop starts from reading inputs proceeding to executing the program then diagnostics and communications and lastly updating outputs.

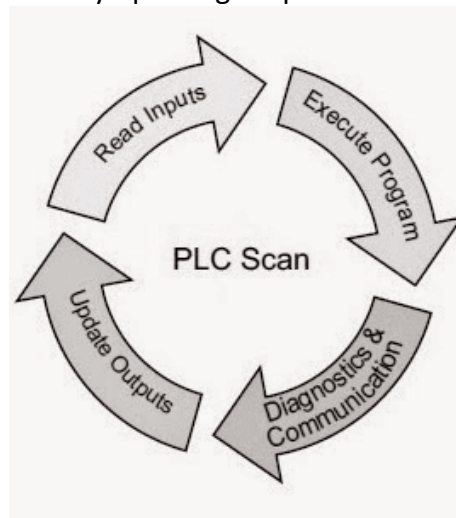


Figure 3. PLC scanning operation (ACC Automation n.d.).

This loop as illustrated in Figure 3 is executed with a high frequency. Therefore, many independent tasks can be executed in parallel and this leads to time saving which is an important factor in control.

2.3 Design of user interface

A graphical user interface (GUI) or shortly, a user interface (UI) is a method which provides means of interaction between human and computer. It helps users understand briefly about a system's functions and manipulate the system by a screen of visualization.

When a UI is designed, there is a set of principles which designers should follow to make the UI become an efficient tool for users to control a system. According to chapter 5 of "The Elements of User Interface Design", Theo Mandel (1997) pointed out three areas in the design principles of a of user interface:

- Place users in control of the interface
- Reduce users' memory load
- Make the user interface consistent.

(Mandel 1997, 5-2.)

Each area contains different principles which help to achieve the area's idea:

- Place users in control of the interface:
 1. Use modes judiciously (modeless)
 2. Allow users to use either the keyboard or mouse (flexible)
 3. Allow users to change focus (interruptible)
 4. Display descriptive messages and text (helpful)
 5. Provide immediate and reversible actions, and feedback (forgiving)
 6. Provide meaningful paths and exits (navigable)
 7. Accommodate users with different skill levels (accessible)
 8. Make the user interface transparent (facilitative)
 9. Allow users to customize the interface (preferences)
 10. Allow users to directly manipulate interface objects (interactive).

(Mandel 1997, 5-5.)

- Reduce users' memory load:
 1. Relieve short-term memory (remember)
 2. Rely on recognition, not recall (recognition)
 3. Provide visual cues (inform)
 4. Provide defaults, undo, redo (forgiving)
 5. Provide interface shortcuts (frequency)
 6. Promote an object-action syntax (intuitive)
 7. User real-world metaphors (transfer)
 8. User progressive disclosure (context)
 9. Promote visual clarity (organize).

(Mandel 1997, 5-14.)

- Make the user interface consistent:
 1. Sustain the context of users' tasks (Continuity)
 2. Maintain consistency within and across products (experience)
 3. Keep interaction results the same (expectations)
 4. Provide aesthetic appeal and integrity (attitude)
 5. Encourage exploration (predictable).

(Mandel 1997, 5-23.)

2.3.1 Placing users in control of the interface

The first area "**Place users in control of the interface**" is a set of principles which guide a designer to create an interface in perspective so that the designer allows users to do their work by themselves rather than try to resolve what they want. Mandel (1997) mentions that, there is a real-life example which helps understand better the key idea of this area: An architect designed a group of buildings without creating walkways between them. A few months later after the buildings were brought into use, the architect came back to build the walkways. At that time, he saw

many worn paths where people had walked between buildings and then he knew where he should put the walkways.

The above example shows that it is wise to give users the control right, observe their behaviour and then create an interface as a tool to help them do efficiently what they want. The principles included in this area are listed below:

- **Use modes carefully (modeless):** Letting users choose a mode which they want rather than forcing them into a mode. By doing this, users feel comfortable and pleasant when using the interface. In order to successfully follow this principle, immediate visual feedback should be taken into account. There should be some sort of indication in the interface to address which mode users are in.
- **Allow users to use either the keyboard or mouse (flexible):** The more means of access the interface has, the more convenient and flexible it is. Many ways of access to the interface does not mean that they will make the interface complicated to use but it provides alternative way to work with the interface if users do not like a specific one.
- **Allow users to change focus (interruptible):** Do not place users into a sequence of premade tasks. It is always beneficial to create a set of options so that users can choose where they want to focus on firstly and they can decide what task they want to do next.
- **Display descriptive messages and text (helpful):** There should always be indicative text on the interface to let users know what is happening. In order to users to understand, this message must be written informatively and as easy-to-understand as possible.
- **Provide immediate and reversible actions, and feedback (forgiving):** Undo and redo actions are features which every interface must have. Besides, it is also wise to inform users about an action which cannot be undone so they have to be careful about the decisions and offer alternative actions. After an action is done, an indication should appear on the interface so that users know that the action has been successfully executed.
- **Provide meaningful paths and exits (navigable):** By letting users navigate effortlessly through the interface and providing ability to move forward or backward, the process of exploring the interface will be less stressful. Therefore, users will feel relaxed when they try all the buttons or functions provided by the interface and they will get used to the interface faster.
- **Accommodate users with different skill levels (accessible):** It is beneficial to create many ways of interaction for different levels of users. Beginners need a clear and simple way to interact with the interface while experienced ones need fast paths to finish work.
- **Make the user interface transparent (facilitative):** The idea of transparent interface is the synchronization between the interface and users' mental model. When the interface achieves this idea, users can be flexible to focus on what they want to perform and

they do not have to translate the tasks into the interface's language which is functions provided by the interface.

- **Allow users to customize the interface (preference):** Allow users to customize and personalize the interface as they like. By doing this, users will feel comfortable while working and it leads to increasing in productivity.
- **Allow users to directly manipulate interface objects (interactive):** Direct manipulation is an ability to control the object directly in the interface, for example, users can drag and drop an object. Direct manipulation is usually good for interaction between users and the interface but it has its own problem which is that it is not always visually obvious for users to know that an object can be controlled directly. Therefore, the interface should be explorable so that users can be free to pick any object and try to check if it is directly manipulated.

(Mandel 1997, 5-5 – 5-13.)

In conclusion, the general idea of this area is that a well-made interface can make users feel entertained and comfortable while the computer system is proceeding. This area helps to create an interface where users are in the centre and have the full right to control the interface or at least, make users think that they are.

2.3.2 Reducing memory load of users

The second area is “**Reduce users' memory load**”. Human memory has limitations. Therefore, the interface should also help users to remember information while using the computer. In order to successfully accomplish this, there are nine principles which designers should follow according to Mandel (1997):

- **Relieve short term memory (remember):** Users usually do many tasks at the same time so it is difficult to remember all information while switching between tasks. Therefore, the interface should have functions, for instance cut, copy, paste, etc. so that the system can retrieve the previous information and users don't have to remember and retype all the information again.
- **Rely on recognition, not recall (recognition):** It is absolutely easier to choose an item which users want from a list than remember the name of the item and type it into a blank field. Hence, whenever it is possible, it is a plus to provide a menu or a list containing suitable item for selection instead of giving an empty field which user must type in.
- **Provide visual cues (inform):** There must be an indication on the interface to show users where they are and what they are doing. For example, when users are using mouse, an arrow on the screen illustrates where the mouse is and the arrow will change its shape when users do an action.

- **Provide default, undo and redo (forgiving):** Computer has great ability to remember and retrieve information from users. Therefore, designers should exploit efficiently this ability of computer by allowing users to store current work or save and name different work. It is considered wise to create many levels of undo and redo so users can explore the program without fear of negative consequences.
- **Provide interface shortcuts (frequency):** When users have enough experience on the interface, they will need a faster and more efficient way to work and this is when shortcuts are needed. There are two ways to create shortcuts: mnemonics (or access key) and accelerator keys. An example for interface shortcut is that the combination of keys Ctrl + N is for creating a new text editing file in Microsoft Word.
- **Promote an object-action syntax (intuitive):** Object-action syntax provides users information about relationships between objects and actions on the interface. Users can select an object and see what actions are available to apply on the object.
- **Use real-world metaphors (transfer):** Real-world metaphor is an effective way to demonstrate visually what a function or operation does. However, once designers have chosen a metaphor, they should try to stick with it throughout the whole program.
- **Use progressive disclosure (context):** The concept of progressive disclosure is to show what users need and then when and where they want it. It is great to provide shortcut or easy access to frequently-used operations and hide less common ones but still show users how to navigate them. Additionally, it is a minus to put all information on one window. Hence, designers should exploit effectively extra windows for displaying less crucial information.
- **Promote visual clarity (organize):** Information displayed on the interface should be put into some order and priority. In order to do so, information should be divided into groups based on its characteristics and then these groups are put on a menu or list. In addition, items should be numbered after it has been classified into some order. Using headings and prompt text also helps to make the interface easy to perceive.

(Mandel 1997, 5-13 – 5-22.)

2.3.3 Making the interface consistent

Consistency is one of the key characteristics of an interface because it determines whether the interface is usable or not. However, the consistency factor depends on the environment so designers do not have to follow strictly these principles if they are not compatible with their environment. Therefore, this factor somehow has less priority than the other factors. The most useful advantage of consistency is that users can apply their old knowledge on a new program if it is consistent with programs which they have used before.

This third area “**Make the interface consistent**” consists of six principles:

- **Sustain the context of the users’ tasks (continuity):** It is useful to provide users indication when they explore an interface so they know where they are. Users should be able to complete the task by the same way as they started it. For example, if users start a task by mouse, the mouse should be the main interaction to complete the task. Additionally, cues to predict the result of an action should also be provided to users so that they can decide if they really need to start the action or they can think about a plan for it. Last but not least, information on how items work should be provided in the same window so that users do not have to open a new window to find supplemental information.
- **Maintain consistency within and across products (experience):** An interface which achieves consistency is the one which enables the users to apply the same knowledge about this interface to other similar interfaces. Consistency applies in three aspects:
 - **Presentation:** Information and objects should be displayed in the same visual, logical and physical way. For example, if an error message is displayed in red on one screen, error message should appear in red throughout the whole interface.
 - **Behaviour:** A kind of object, for instance a button, switch, list, etc. should behave the same way everywhere.
 - **Interaction:** Mouse technique, shortcut keys, etc. should work the same throughout the whole interface and they should also be the same with other similar programs.
- **Interface enhancement and consistency:** When an interface is enhanced, only few behaviours or techniques should be changed at a time. Therefore, users will not be put into a situation where they struggle with many changes and they will adapt quickly with the enhancement.
- **Keep interaction results the same (expectations):** One action can have many results so it is professional to inform the users about a potential future result of an action and then let them decide if they want to perform it. Moreover, it is also good to give them the right to cancel the process and offer alternative actions to be performed.
- **Provide aesthetic appeal and integrity (attitude):** All elements in an interface should fit to each other in many aspects such as font, size, icon, colour, etc. One interface project is usually done by many designers. Therefore, they should make an agreement amongst themselves about how the interface should look like so that the final product as a whole should look harmonic. And a nice-looking interface will absolutely create pleasing experiences to the users.
- **Encourage exploration (predictability):** Besides aiming at a functional interface, designers should also aim at creating a user-friendly interface. The user-friendly factor encourages users to explore the interface freely without being afraid of negative results. To accomplish this factor, designers should provide enough

information, guidance and even entertainment for the users while they experience the product interface.

(Mandel 1997, 5-22 – 5-27.)

3 IMPLEMENTATION METHOD

3.1 Selection of software

Nowadays, there are various pools of options in the software market for graphical user interface design in the field of automation, for example, the TwinCAT integrated visualization editor, Kaseya VSA, Indusoft Web Studio, etc. Each piece of software has its own advantages and disadvantages and in this thesis, Indusoft Web Studio and TwinCAT integrated visualization editor are taken as examples for comparison. Studying to the comparison, readers can discover general features offered in a user interface development software. Moreover, advantages and disadvantages of these two pieces of software are also pointed out.

Wonderware Indusoft Web Studio, shortly Indusoft Web Studio, is an object-oriented designing software product from Schneider Electric for building HMI (Human-Machine Interface), SCADA (Supervisory Control And Data Acquisition) and embedded instrumentation solutions. The software contains a vast collection of automation tools and Indusoft integrated web technologies. These integrated web technologies take advantage of internet/intranet connectivity to provide designers an ease to check their projects anytime and anywhere through a standard web browser which supports XML (Extensible Markup Language). Moreover, Indusoft Web Studio supports UNICODE, which is an international standard for encoding. Hence, quick troubleshooting ability is enabled since alarms and errors can be interpreted effortlessly. Indusoft Web Studio has 14 main features for an interface:

- Design tools
- Thin clients
- Alarms
- Redundancy
- Trends
- ActiveX and .NET
- Events
- Scripting
- Security
- Recipes and reports
- Diagnostics
- Database
- Drivers/OPC
- Built-in download tools.

(Indusoft Web Studio Product Features n.d.)

When an interface is constructed by Indusoft Web Studio, each object is named with a tag. Afterwards, when the integration process between the interface and the system takes place, the tag is connected to a variable in the system and then the object's behaviour will affect the value of the variable. As a result, there is a tag list including all the objects in the interface for designers to edit and manipulate. Furthermore, scripting is a decent feature furnished in Indusoft Web Studio. The scripting feature allows designers to invent their own objects which the software does not have. In other words, these objects behave suitably as designers want to but differently to all those offered by the software. Hence, designers can exceedingly improve the innovativeness of the interface.

Unlike Indusoft Web Studio, a visualization editor is combined directly within TwinCAT, which is a programming software for the Beckhoff PLC system. Therefore, users do not have to install any other interface development software but they can still program their system and design an interface for it in parallel on the same software. Another advantage of this combination is that there is no tag list as users can access to variables in the controlling system directly. In addition, OPC, which is an interoperability standard for secure and reliable exchange of data and usually complicated to configure, can also be omitted because both the interface and the system are in the same program so there is no need for data transfer. The TwinCAT integrated visualization editor provides some ready-made functions:

- Elements:
 - Rectangle, Ellipse, Rounded rectangle.
 - Line, Polygon, Polyline, Curve.
 - Bitmap, WMF-file.
 - Button, Table, Histogram, Bar Display, Meter.
 - Reference to another visualization.
- Animations (depending on element type):
 - Text display.
 - Colour changes.
 - Visible/Invisible.
 - Shift.
 - Rotation.
 - Scaling.
 - Offset on the particular edges of an object (for Bar Display).
 - Button active/inactive.
- Input possibilities:
 - Toggle/tap Boolean value.
 - Text input.
 - Change of visualization.
 - Special actions (Leave visualization, Read/Write receipts, Switch language, call external EXE, etc.).
 - Choose line (only text display).
- Further properties:
 - Switching language.

- Tooltips for all elements.
- ASCII Import/Export.
- Background Bitmap.
- Automatic Scaling.
- Drawing operations: Alignment, Order, Grouping.
- Placeholder concept creating objects with complex graphic elements.
- Programmed visualization expressions.

(TwinCAT PLC Control Visualization n.d.).

All the functions above are basic offers in mostly all interface development software including Indusoft Web Studio. However, TwinCAT integrated visualization editor does not have scripting feature, which is a vital drawback for the software as users cannot create their own elements. Consequently, creativeness in designing the interface is limited in some way.

In conclusion, below is a table which illustrates briefly basic differences between two software:

Table 1. Comparison of software.

Indusoft Web Studio 8.0	TwinCAT 3.1 integrated visualization editor
<ul style="list-style-type: none"> - Used mainly for designing GUI, SCADA. - Interface design software must be installed separately. - Tag list for connecting to system's variables. - Scripting feature is provided. - Integration process and OPC are needed for data transfer. - Thin clients to access project from different platforms. 	<ul style="list-style-type: none"> - Used mainly for programming PLC. - Only TwinCAT is needed for both system programming and interface designing. - Direct accessibility with system's variables. - Scripting feature is not available. - No integration and OPC is not necessary. - No thin client.

It can be seen from the comparison in Table 1 that, although the integration process between the interface and the system causes difficulties, Indusoft Web Studio supports creativity and flexibility in designing the interface. Therefore, Indusoft Web Studio was chosen to implement the practical work of this thesis.

3.2 Description of user interface

According to the commissioner's requirements, the interface should be able to control fundamental household appliances such as lights and window blinds. The interface was also to manage energy consumption by

controlling the heating system. Besides monitoring devices, the interface also needed to provide users general information about the house, for example the temperature and the status of the devices. Additionally, there was to be a feature which allowed the users to apply a specific pre-set mode to the house. And a mode is a set of manipulation to all the devices so that the users do not need to control each device individually. Finally, there was also to be a security system to protect the interface from strange users.

4 CREATION OF USER INTERFACE

After having acquired knowledge about PLC and GUI, the author here wants to illustrate to the reader how the process of creating the interface was conducted. The functions of the software might be modified in updated versions in the future. This project was conducted by Indusoft Web Studio version 8.0 and one must bear in mind that all the descriptions given here may be compatible with version 8.0 only.

First of all, the interface layout needs to be sketched out and divided into small parts so that designers could work with each part at a time. By this way, the project will be simplified and the designers will have a clear picture of where in the process they are when they are working. In this project, the interface layout resolution was decided to be 1024 x 768 pixels which is the common size nowadays and had three parts: the header, the navigator and the main area with a resolution of 1024 x 150, 150 x 678 and 874 x 618 respectively. Figure 4 depicts the author's idea as to the layout.

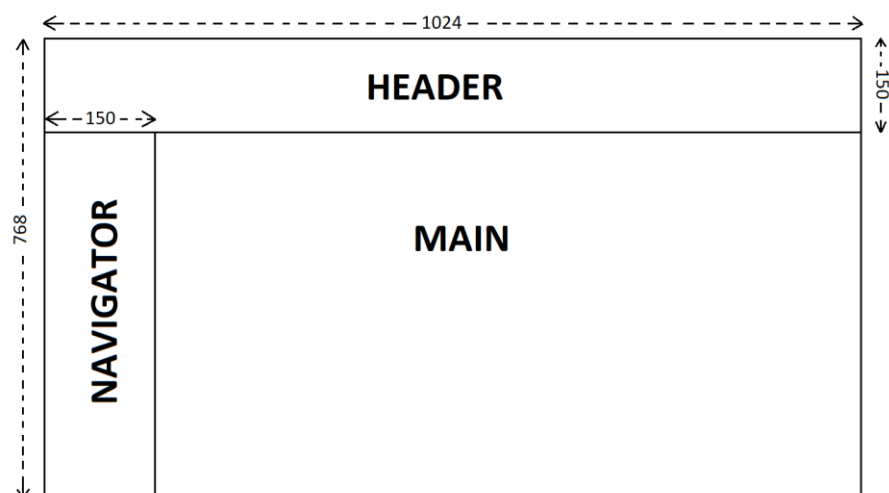


Figure 4. Interface layout.


4.1 Interface design


4.1.1 Header

Firstly, a template was created. Using a template decreases the workload when creating each screen of the interface. The reader will see the efficiency of using a template later in this thesis. As mentioned earlier, each screen has three parts: the header, the navigator and the main area. The header and the navigator are the same in each screen. Hence, the header and the navigator were designed initially for the template.

The header is a place where users will have a look at the very beginning when the interface is opened. Therefore, basic information about the interface should be shown in the header. After reading all the information here, the users should know what the interface is about and they are able to guess what kind of functions it offers. Moreover, means of access to the interface should be placed here in the header, for example, “log in” for the security system and “program exit”.

To begin with, a 1024 x 768 screen was created in Indusoft Web Studio with a name of Template. The header size was 1024 x 150 so a rectangle with the same size and light blue background was drawn and placed on the top of the screen. A title “Home Automation User Interface” was inserted here as the header with the intention to give the users an idea about the purpose of the interface. Additionally, a security system was created in order to protect the interface from strangers. For this purpose,

a “Log on” button  Log On was created so that when this button is pressed, a log on window will pop up and only users with right usernames and passwords can access to the interface. This security system is a ready-made function provided by the software so the log on button with set configuration can be obtained from Project Explorer → Graphics tab → Symbols → Buttons or in tool bar → Graphics tab → Libraries → Symbols

→ Buttons. Moreover, an “Exit” button  Exit can be picked up from the same place as the log on button. This button allows users to close the interface. These two buttons were placed on the top right corner of the header. Finally, on the left top corner of the header, a small date and time display was formed. This display was a rectangle whose caption was written by built-in scripting language of the software. The caption can be modified in the Object Properties window. By writing {Date} and {Time} like in Figure 5, these commands will automatically retrieve the date and time information from the operating system and display them on the screen.

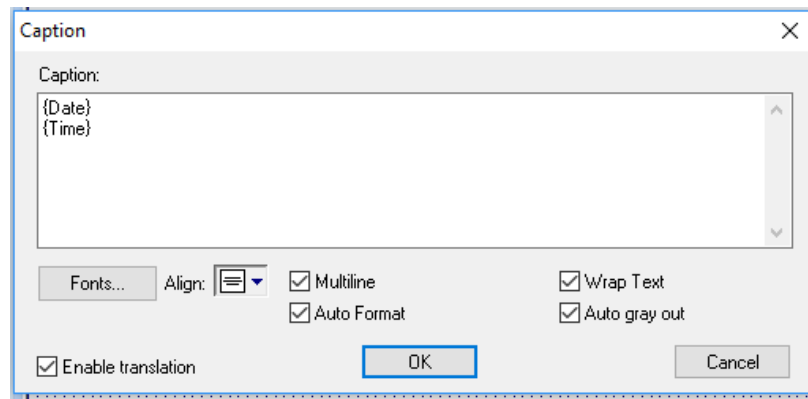


Figure 5. Date and Time commands.

As a result, the header looks like in Figure 6 below.

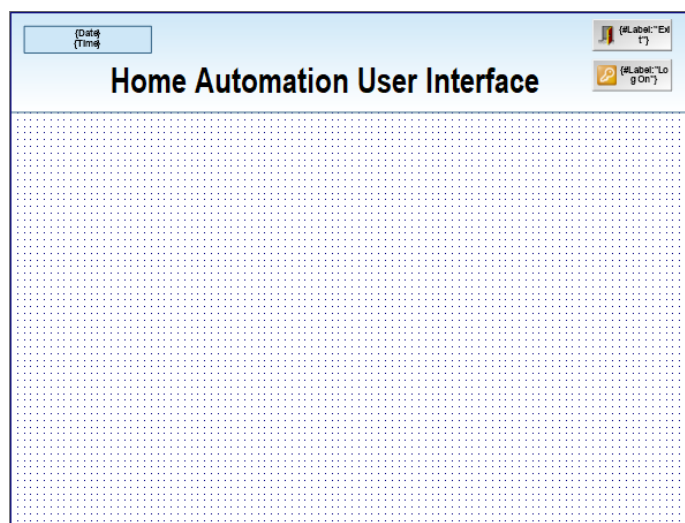


Figure 6. Header in design mode.

It is worth noticing that when in the design mode, all the captions and labels are shown in the form of scripting language. However, when the project is started and in the run mode, the captions and labels are displayed correctly as they should be when users work with the interface (Figure 7). From now on in this thesis, most of the demonstrations are shown only in the design mode.

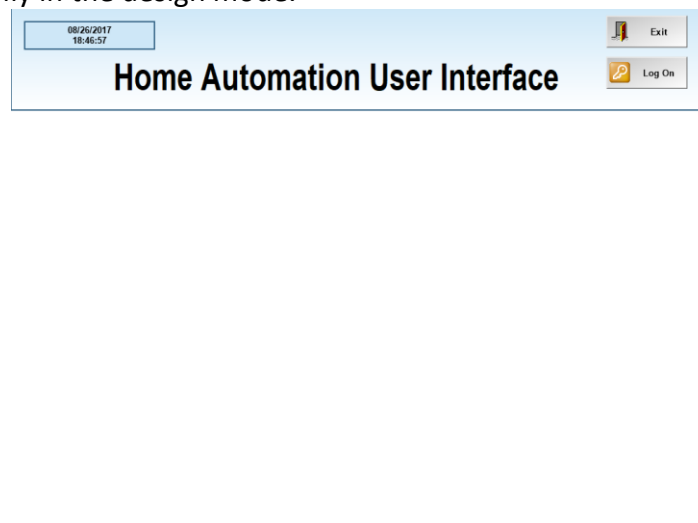



Figure 7. Header in run mode.

4.1.2 Navigator

A navigator is a tool which is used to move between the screens of the interface. In this case, the screens were rooms of a house where there were electronic devices controlled by a PLC. The interface was planned to be implemented for four rooms: the living room, bedroom, kitchen and garage. Furthermore, two additional screens were needed for changing the modes and the main screen of the interface. Hence, the navigator could be pictured to have six buttons to move between six screens.

Firstly, a 150 x 678 rectangle was placed on the left side of the interface and under the header. This rectangle was the area for the navigator's buttons. Next, the buttons were obtained from Indusoft Web Studio tool bar, in the Active Objects of Graphics tab. In Objects Properties, the names of the rooms were written into the caption of the buttons so that each button represented a room. In order to provide the buttons an ability to move to another screen, the buttons were selected and mounted to

Command  Command in the Graphics tab. After this step, in the Command window of Object Properties, the Configuration window was opened and modified. Figure 8 illustrates the configuration of the "Living room" button. As can be seen from Figure 8, command type was set to "Built-in Language" and a list of actions were written. When the "Living room" button is pressed, this list will be executed: "livingRoom.scr" which is the screen for the living room will be opened, a Boolean tag LR_Screen for the living room screen will be true and all other screens will be false.

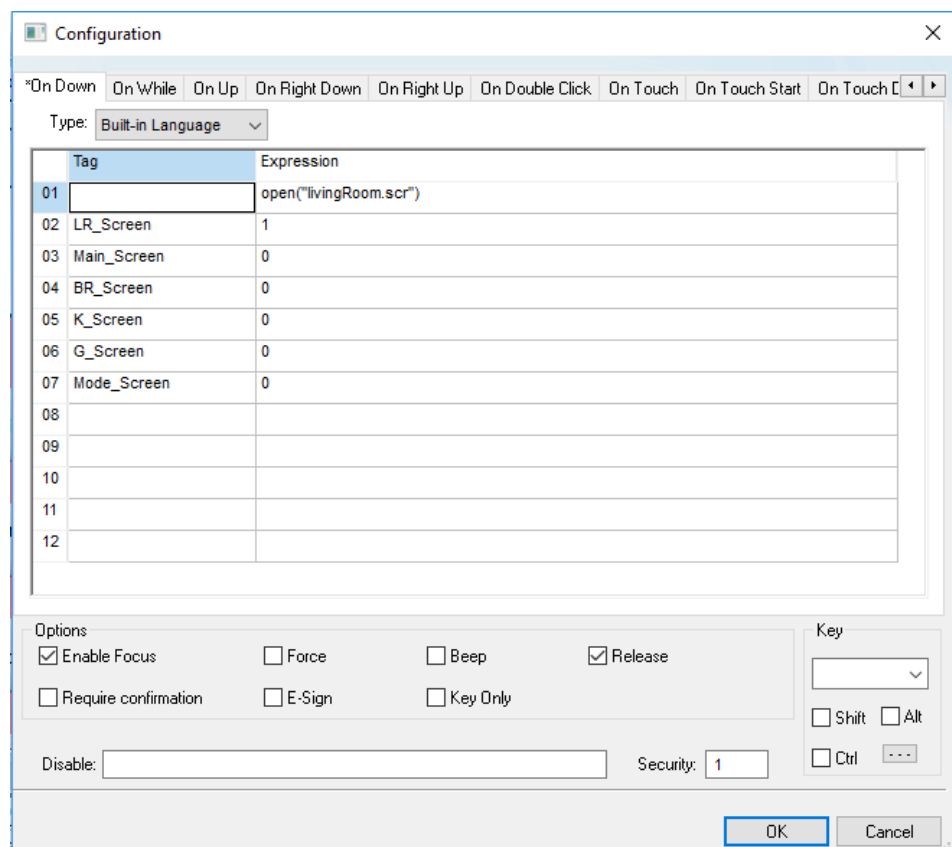



Figure 8. The "Living room" button configuration.

Secondly, Color  was applied to all buttons with the aim of illustrating to the users which screen they were in. For example, when the users were in the living room screen, the “Living room” button had distinct colour from the other buttons. It can be seen from Figures 9 and 10 that when the tag LR_Screen is 1, the button is green and it is otherwise grey.

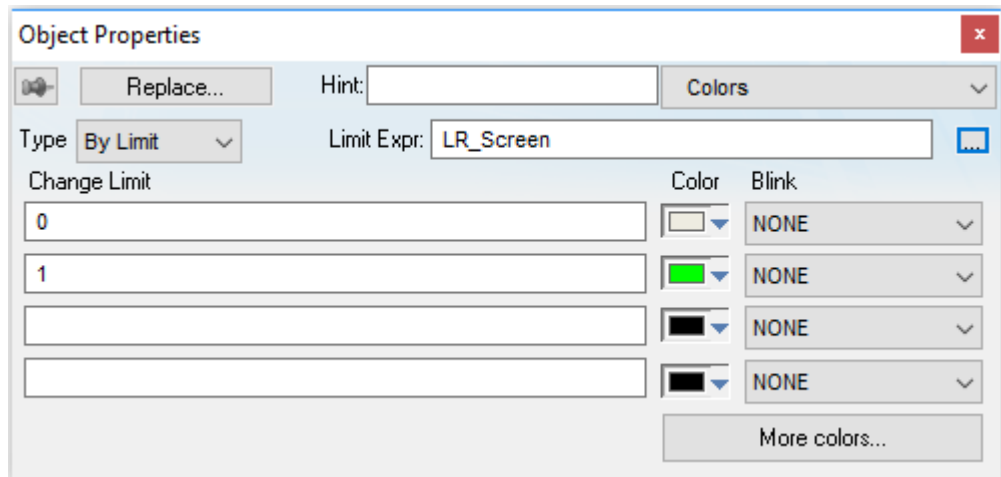


Figure 9. Colors configuration for "Living room" button.

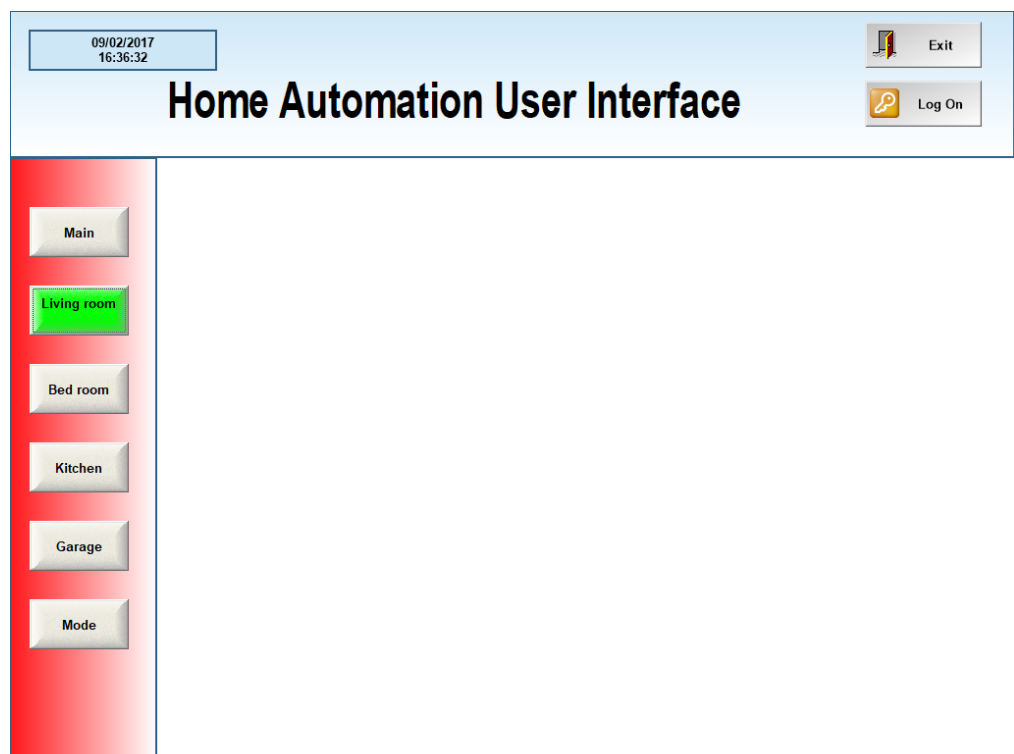


Figure 10. "Living room" button is selected in run mode.

Next, for simplification, it was assumed that there were only two groups of users: owners and guests. Thus, two levels of security 0 to 1 were established (Indusoft Web Studio offers security level from 0 to 250). This meant that guest users can access security level 0 and owner users can

access both security levels 0 and 1. This security setting was configured in Project Explorer → Global tab → Security. Figures 11 and 12 show the security configurations for guest and owner users respectively.

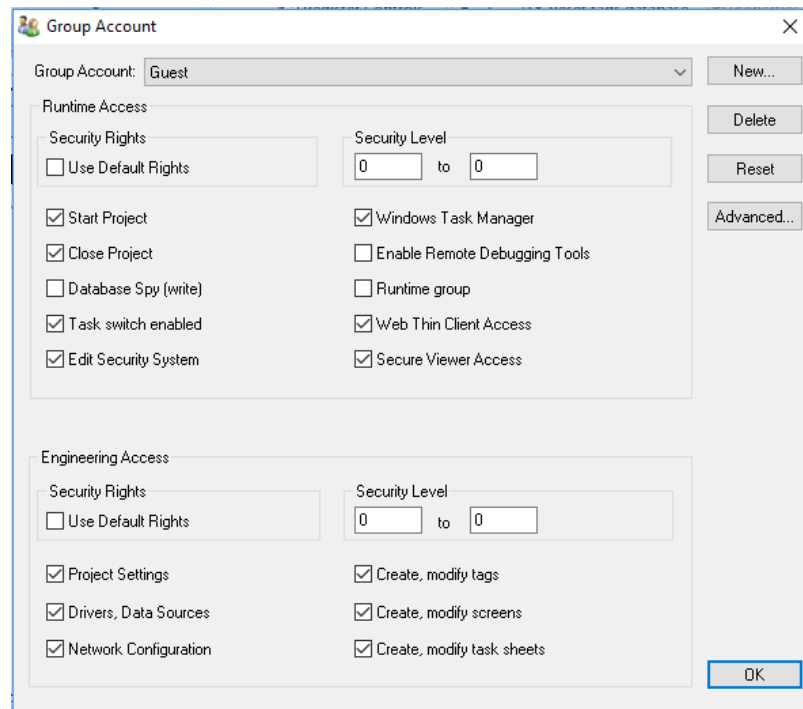


Figure 11. Guest security group configuration.

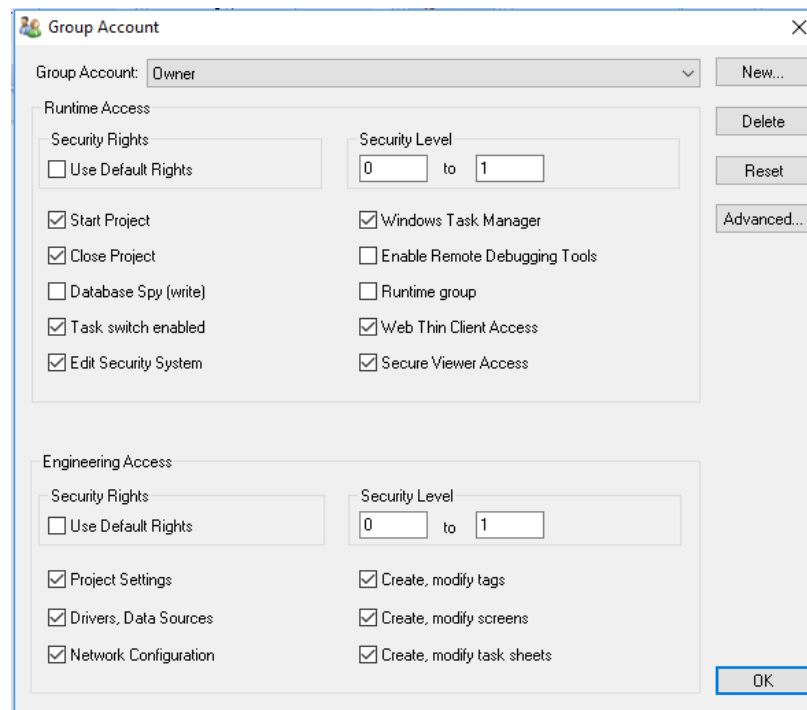


Figure 12. Owner security group configuration.

Guest users can only access the main screen while owner users have permission to access the whole interface in the run time. Therefore, all the buttons were set to 1 in security (“Living room” button in Figure 8 is an example) except that the “Main” button was set to 0. Furthermore,

“guest” was set as the default user so that whenever the interface is opened, the user needs to log on as “owner” to gain full rights to control the interface. Last but not least, after using the interface, owner users must remember to log off before closing the interface, otherwise other users can have full rights just like owner users to control the interface when they access it.

After the buttons had been created, the background of the navigator was set to red with the intention to help users distinguish the navigator from the header and the main area and to make it more attractive. Figures 13 and 14 illustrate the interface in the run mode so that the reader can see the difference in the interface between the two groups of users and the current user can be checked from the “log on” window.

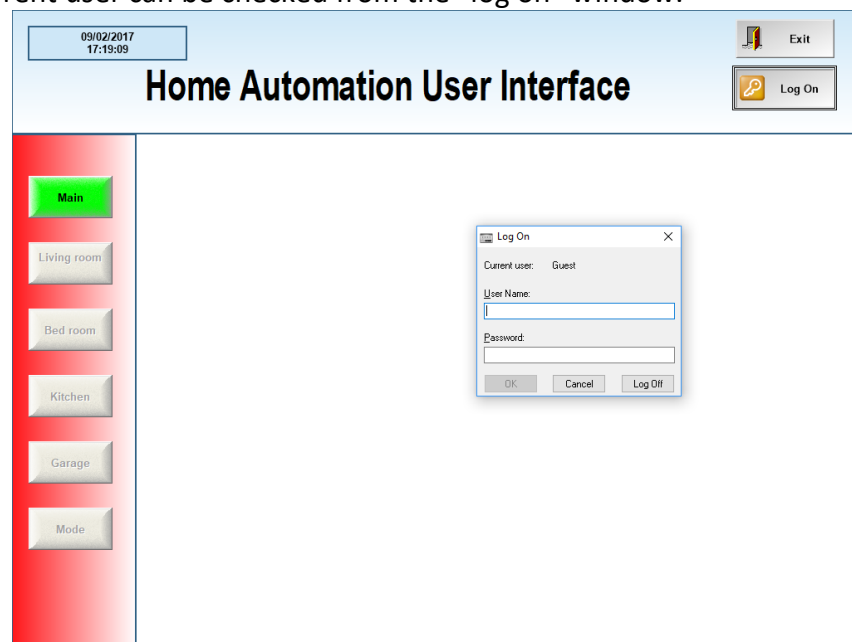


Figure 13. Navigator as guest users.

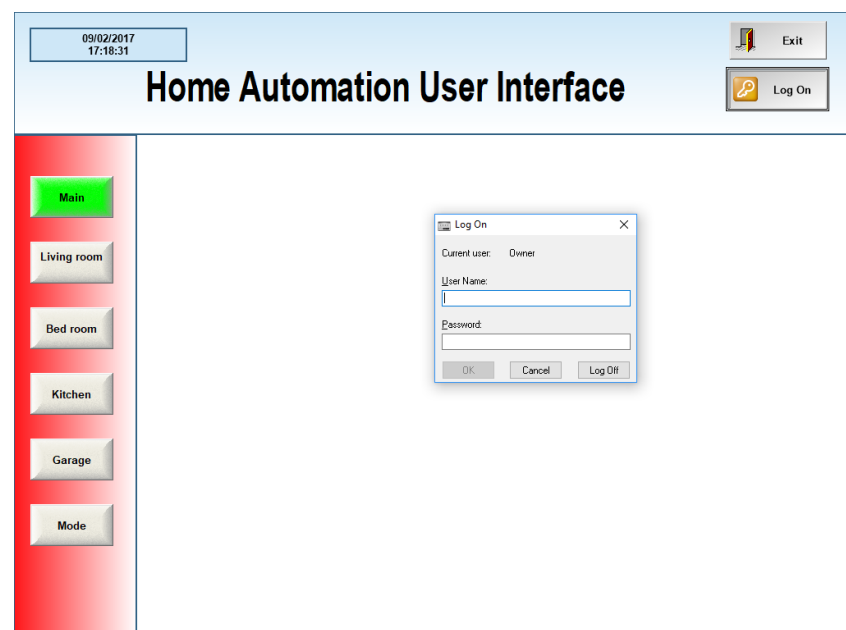


Figure 14. Navigator as owner users.

Finally, the navigator and the header, which are mandatory parts of each screen, were designed and saved as a template for all the other screens. This meant that whenever a new screen was created, they would automatically have the same navigator and the same header. Consequently, designers would not waste time for creating the header and the navigator all over again for every screen. For that reason, the workload was considerably reduced.

4.1.3 Main area

After the template had been built, six screens were created based on the template: the main screen, the living room, bedroom, kitchen, garage and the mode. In each screen, a picture depicting a room was set as the background, for instance, a picture of the living room is the background to the living room screen's background. As a result, users can easily recognize where the electronic device is which they are controlling.

The main screen is the first contact between the user and the interface. However, there is not any function which can be performed on this screen because the "Main" button which opens the main screen has security level 0 and the main screen is the start-up screen which is opened automatically whenever the interface is started. Consequently, any type of user can have access to this screen and for security reason, no function is offered from this screen. In other words, when the interface is started, the main screen is opened firstly and the user cannot do anything else than exit the interface or log on as an owner user to get a higher access to the interface from the header. To conclude, the main screen is the place to verify if the user is authorized to utilize the interface.

The background of the main screen is a picture of the house from the outside (the background picture was obtained from an external source mentioned in references). Additionally, a textbox was created to provide information on the outside temperature to make the main screen slightly more useful for the users.

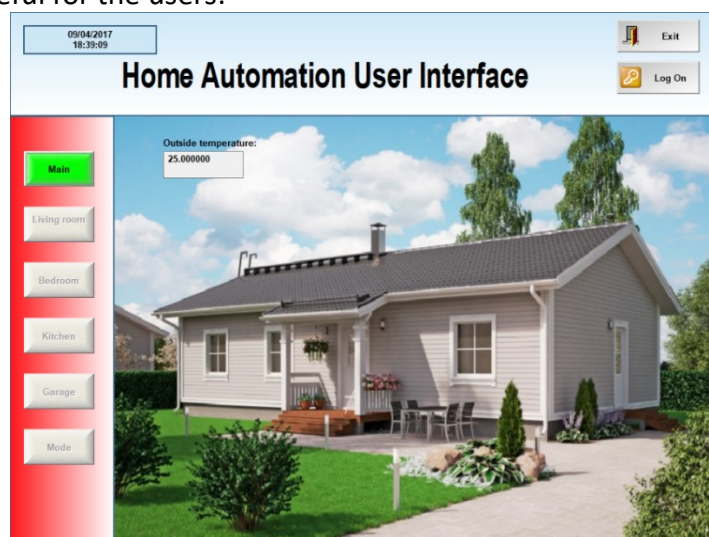


Figure 15. Main screen in run mode (Jukkatalo n.d.).

After this, the living room, bedroom, kitchen and garage screens were designed by the same technique. The creation of the living room screen was taken as example to demonstrate how these screens were designed. First of all, a picture of the living room was set as the background for the living room screen. Next, the all electronic devices in the picture were mounted to functions which made them interactable on the interface.

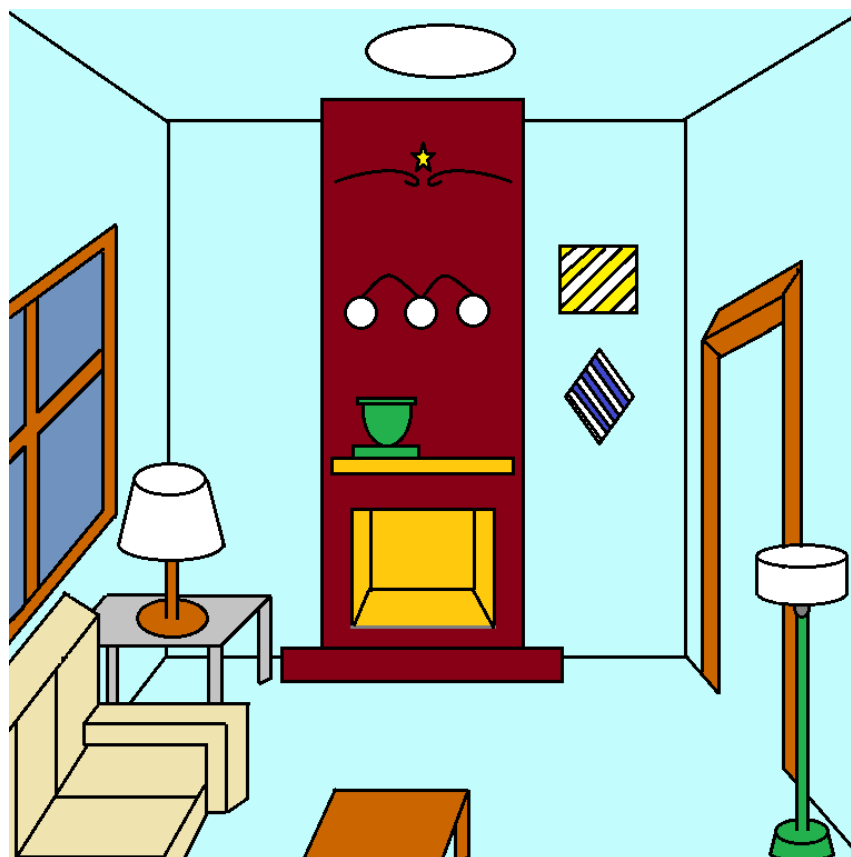




Figure 16. Background of living room screen.

As can be seen from Figure 16, the electronic devices which can be controlled remotely by the interface are the ceiling light, the dim light, the table lamp, the standing lamp and the window blind. These devices can be divided into two groups based on type of controlling input. The first group is controlled by a digital input and it includes the ceiling light, the table lamp, the standing lamp and the window blind. They have only two stages: ON or OFF, while there is only dim light in the second group and it is controlled by an analog input.

The ceiling light, the table lamp and the standing lamp were created by the same technique. Three buttons could have been created as switches in order to control these lights. However, with the aim of constructing a user-friendly and professional interface, the buttons were omitted and integrated into the lights itself. Hence, the interface looked clearer without any button. Firstly, three white ellipses were drawn to illustrate light bulbs. Next, these ellipses were equipped with Command  and Color  properties. The Command property acts as a switch to turn the lights on or off and Color property indicates if the

lights are on ON or OFF stage. In the Command configuration window, command type was set to “Toggle tag”, name of a Boolean tag which controls the light was typed in and “On down” was chosen so the tag will be toggled when the light is clicked or touched. In Colors configuration window, the same tag name was written in Limit Expr so the tag will trigger the light to change its color. This tag would be connected to a PLC so that the PLC could control the tag remotely. When the tag value is 0, the light’s color is white which means it is in OFF stage and when the tag value is 1, the light is in ON stage and its color is yellow. When all the configurations were done, the ellipses were moved to the real light’s position on the background.

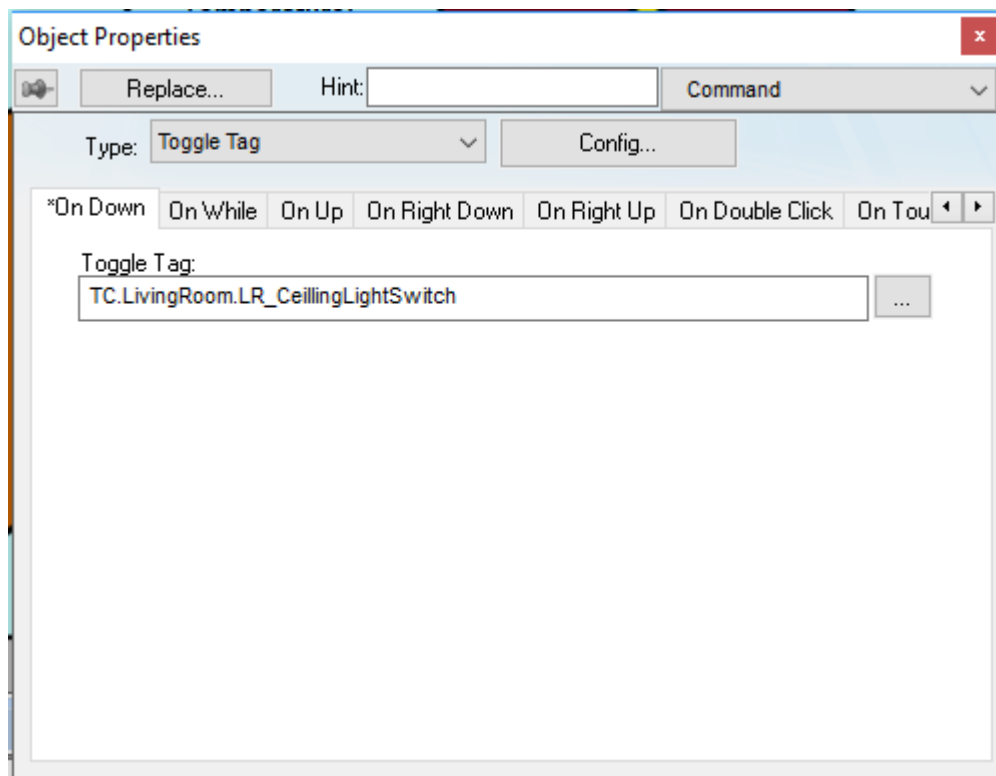


Figure 17. Ceiling light’s command configuration.

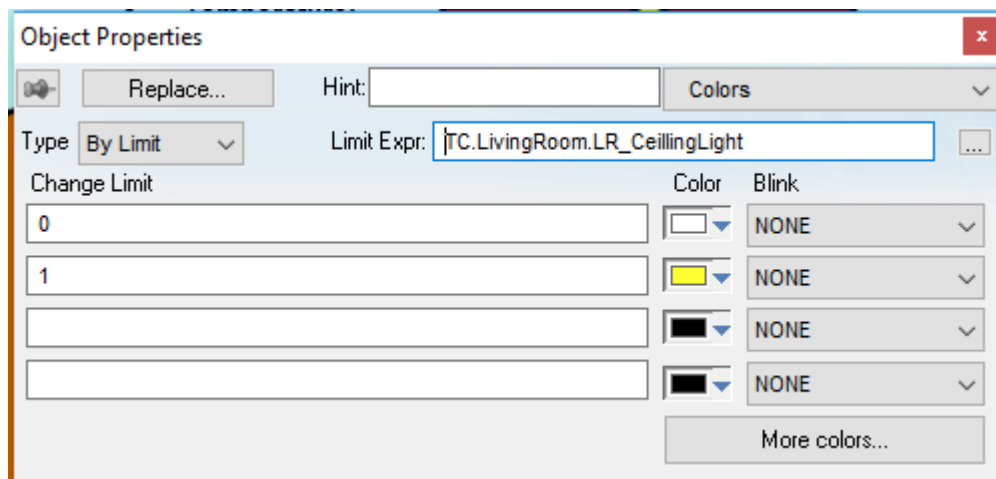




Figure 18. Ceiling light’s colors configuration.

Figures 17 and 18 are example of ceiling light's configurations. The standing lamp and the table lamp had the same configurations except the tag names. There was still the window blind in the digital group. Being different from the lights, the blind was represented by a quadrilateral and it was equipped by only the Visibility/Position  Visibility/Position property. This property provides the blind an ability to be visible or invisible on the screen when its tag is triggered. However, when the blind is invisible, it cannot be selected on the screen to toggle the tag to make it visible so it is not feasible to integrate the blind and its switch together. Therefore, an additional button with Command  Command property was created to control the blind's visibility. This command property was configured the same way as the lights' command property as illustrated in Figure 19 and 20.

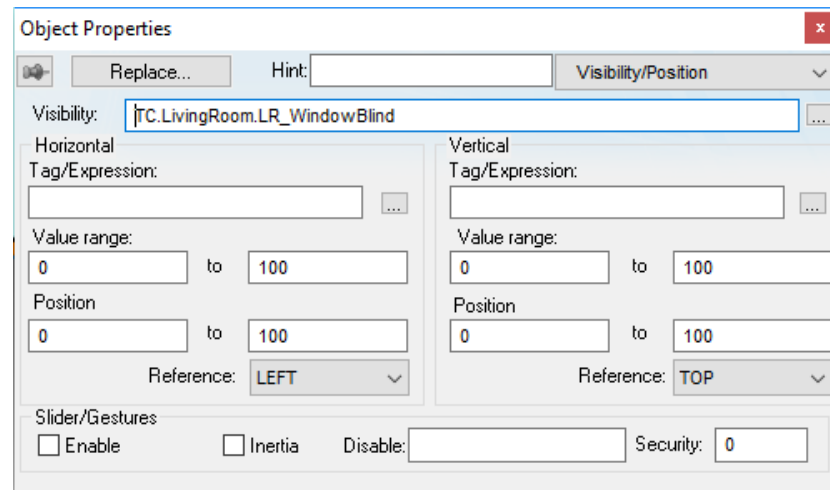


Figure 19. Window blind's visibility configuration.

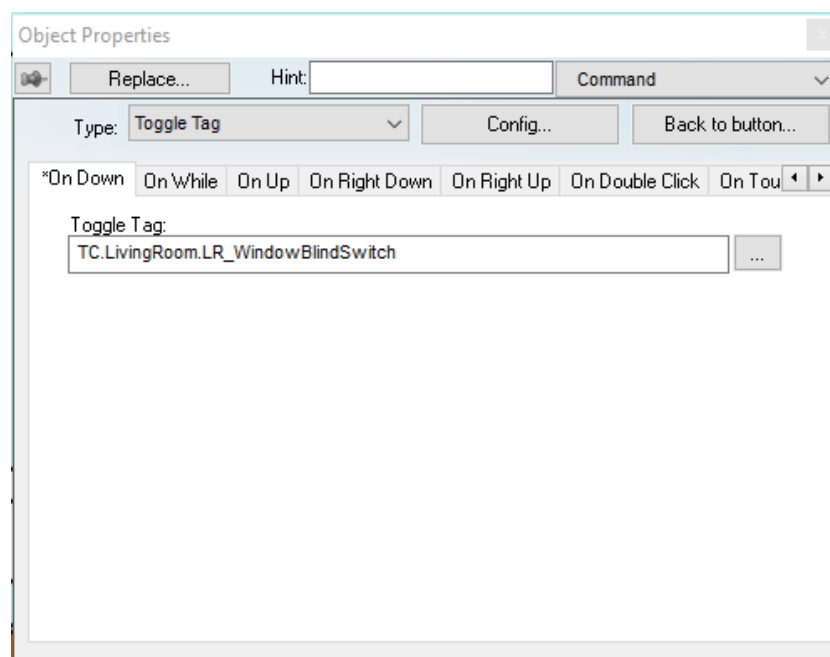



Figure 20. Window blind's control button's command configuration.

Afterwards, the dim light in the second group was handled. Since the dim light is controlled by an analog input, a slider was inserted to the screen to control it. The slider was obtained from software's ready-made symbol library (Project Explorer → Graphics → Symbols → Sliders). As can be seen from the background picture, the dim light system consisted of three light bulbs so three circles were drawn to represent them. These circles were

equipped with Color  Color property. In the configuration window, a name of an integer tag controlling the dim light was typed in Limit Expr. For simplification, the dim light had only four limits and a color for each limit was chosen as shown in Figure 21 so that the lowest limit 0 showed that the light was off and the highest limit showed that the light was in its brightest stage. For slider configuration (Figure 22), the same tag was typed in TagName, Min and Max values of slider were also provided.

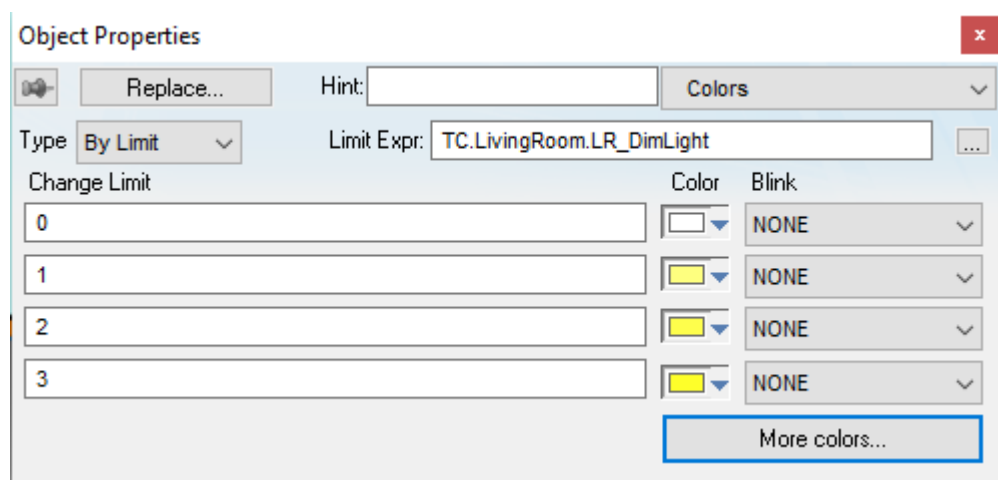


Figure 21. Dim light's color configuration.

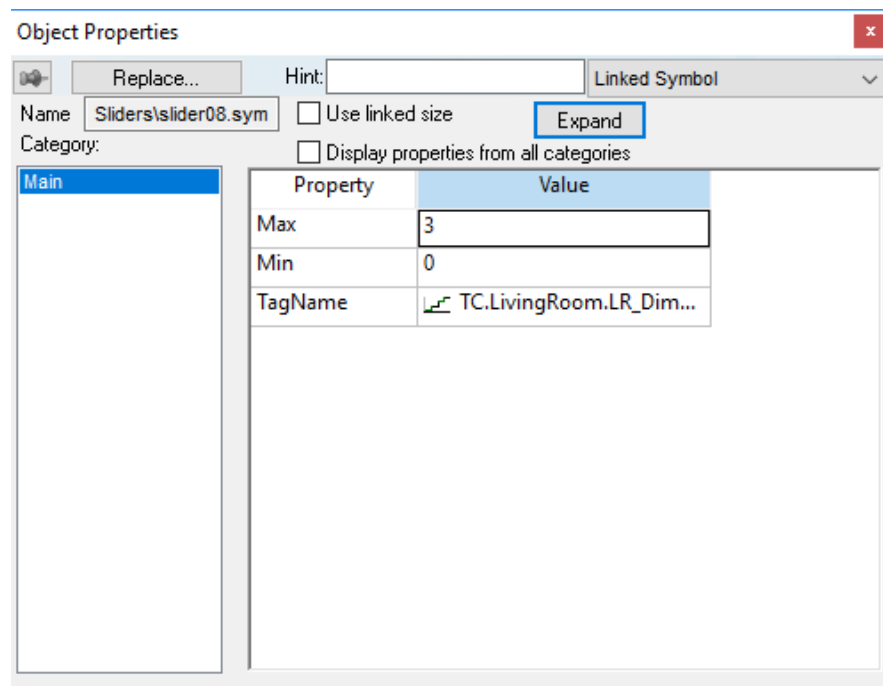


Figure 22. Slider's configuration.

Then, the energy management was the last step in the screen. In order to provide the users an ability to manage energy consumption, room temperature control feature was added to the interface. This feature includes showing the room's recent temperature and changing it. Two text boxes were created for the feature. One box is for showing the temperature. The purpose of this box was only for providing the users information on the room's temperature so the users were not allowed to enter any input to this box, therefore, ability to receive input of this box was disabled (Figure 23). The other box took temperature input from the users and sent it to the PLC. This box had an ability to receive input from users and the input was about room's temperature so max and min values for input were set as 35 and 0 respectively so users cannot enter accidentally input outside this range (Figure 24).

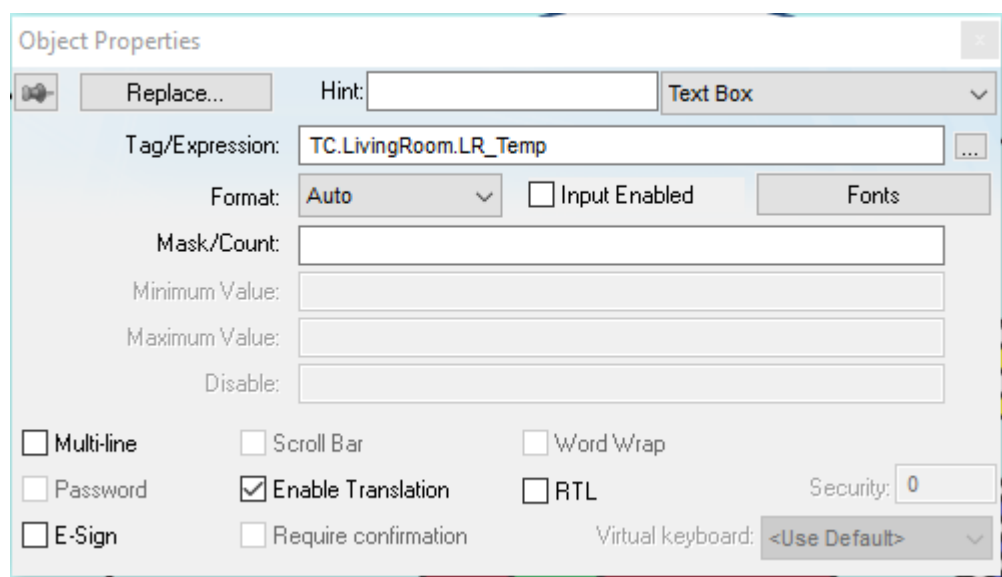


Figure 23. Showing temperature configuration.

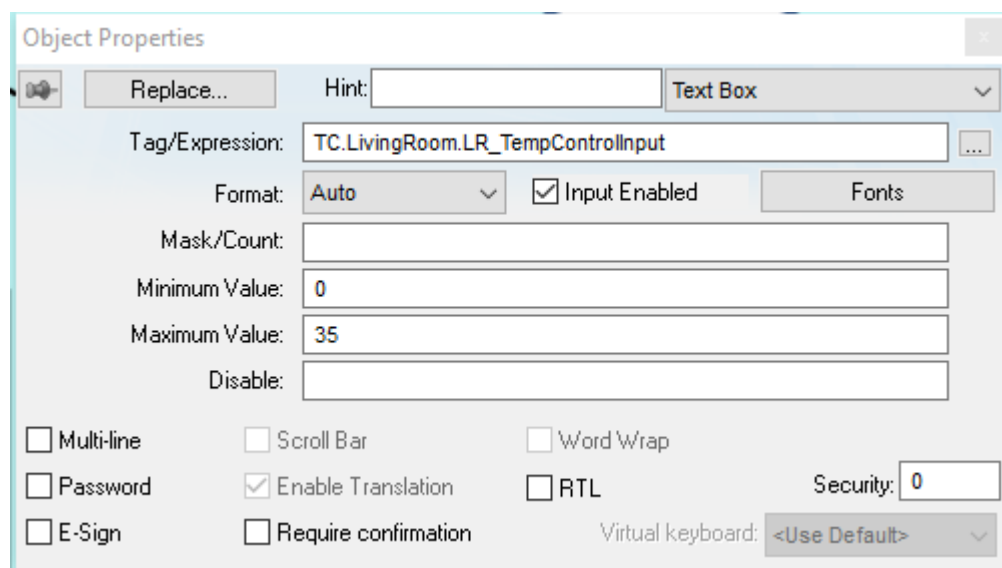


Figure 24. Controlling temperature configuration.

Finally, the living room screen was finished and Figure 25 illustrates the screen in the run mode. The bed room, kitchen and garage were constructed by the same instruction except that the garage screen did not have energy management feature because there is no need to do so.

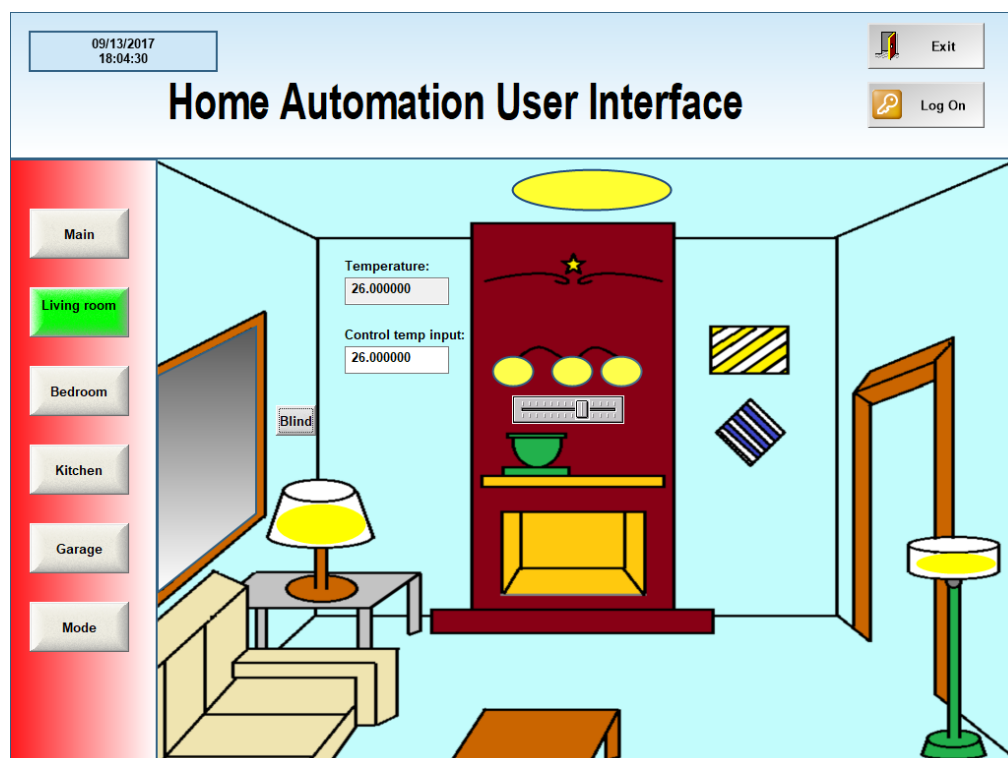


Figure 25. Living room in run mode.

There was still the mode screen which is different than the other screens. This screen contains modes setting the whole interface to some desired stages. A mode consists of a series of operations on the house's electronic devices. In this project, there were three modes: Sleep/Away, Day and Night mode. Hence, three buttons were formed and math sheets which is a feature of Indusoft Web Studio were needed to build these modes. Additionally, brief descriptions about the modes were written next to the buttons, for instance "Sleep/Away mode turns off all lights, curtains and doors". These descriptions should be short, however, informative so that they do not make the screen complicated but the user can still understand the features of the buttons. As mentioned above, math sheets were needed to build the modes and they were inserted from Project Explorer → Tasks → Math. A math sheet includes 3 parts: a description about the sheet, a tag to execute the sheet and a built-in language list to set up a series of operations. When the tag is set to TRUE by pressing the button, the series of operations is executed. However, the stage of the tag is always TRUE after the button is pressed, as a result, the series of operations is executed repetitively. This leads to a problem that users cannot change the stage of any devices after a mode is set. For example, when Sleep/Away mode button is pressed, all the lights are turned off

then a user press a light on the interface to turn it on but the light still stays off. A solution for this problem is that the last operation in the series is to set the tag back to FALSE. After this step, the whole interface is ready to use.

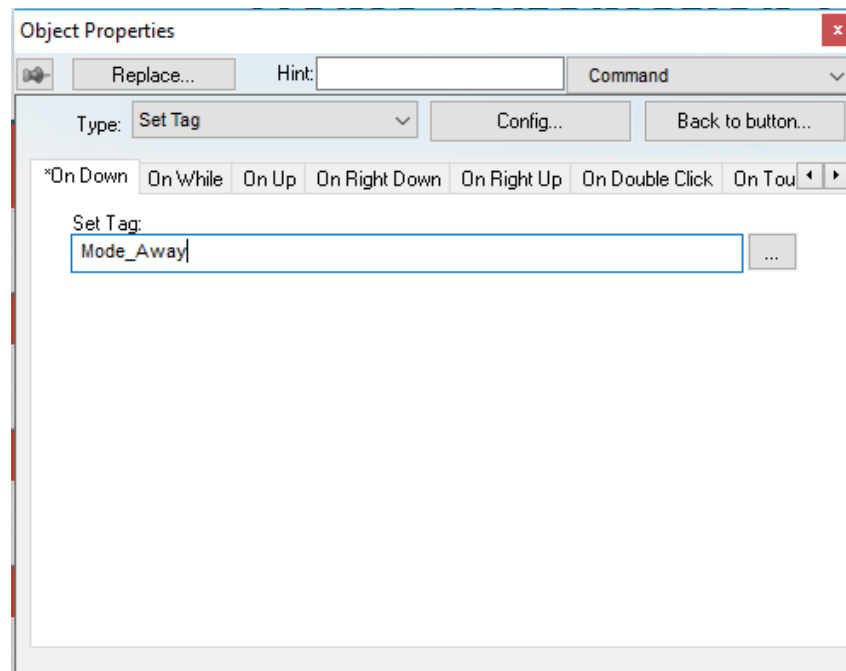


Figure 26. Away/Sleep mode button's configuration.

Description:

Execution:

	Tag Name	Expression
	<input type="text" value="Filter text"/>	<input type="text" value="Filter text"/>
1	TC.BedRoom.BR_CeilingLightSwitch	0
2	TC.BedRoom.BR_NightLampSwitch	0
3	TC.BedRoom.BR_TVRemote	0
4	TC.BedRoom.BR_WindowBlindSwitch	1
5	TC.Kitchen.K_CeilingLightSwitch	0
6	TC.Kitchen.K_SinkLightSwitch	0
7	TC.Kitchen.K_StoveLightSwitch	0
8	TC.Kitchen.K_WindowBlindSwitch	1
9	TC.LivingRoom.LR_CeilingLightSwitch	0
10	TC.LivingRoom.LR_StandingLampSwitch	0
11	TC.LivingRoom.LR_TableLampSwitch	0
12	TC.LivingRoom.LR_WindowBlindSwitch	1
13	TC.BedRoom.BR_DimLightSlider	0
14	TC.LivingRoom.LR_DimLightSlider	0
15	TC.Garage.G_CeilingLightSwitch	0
16	TC.Garage.G_DoorSwitch	1
17	Mode_Away	0
*		

Figure 27. Math sheet for Away/Sleep mode.

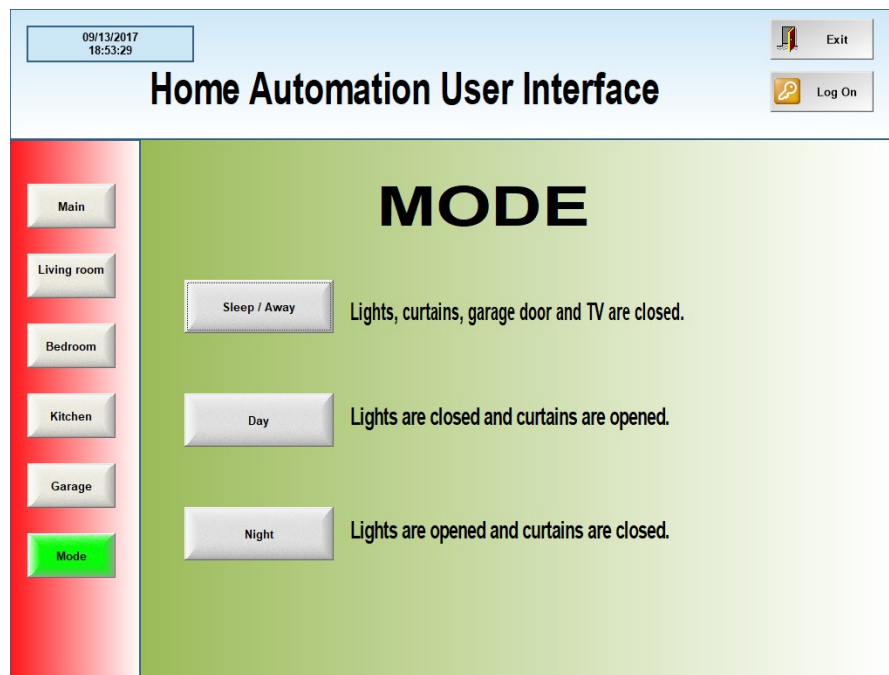


Figure 28. Mode screen in run mode.

4.2 Studio Mobile access

After the interface was created, the next goal was to provide users a convenient method to use it. Users cannot carry with them their laptops or computers just to turn on or off a light, therefore, a more feasible and remote method was required, and the web browser was chosen for this purpose. Indusoft Web Studio delivers three thin clients to access a project remotely from three different platforms:

- Secure Viewer
- Web Thin Client
- Studio Mobile Access.

Secure Viewer requires only the software installation of Secure Viewer. This software gives users permission to access remotely the interface from different computers running on the Windows operating system. Web Thin Client and Studio Mobile Access are for accessing the interface via web browsers. However, Web Thin Client works only on Internet Explorer while Studio Mobile Access works with all browsers supporting HTML5. The other difference between these two clients is that Web Thin Client is mainly for viewing and interacting the interface remotely while Studio Mobile Access is mainly for viewing and interacting with tags and alarms remotely. Even though the purpose of Studio Mobile Access is to manipulate tags and alarms, it was chosen as a remote-control method for this project because it also has the ability to access the interface by the web browser and it is compatible for many web browsers supporting HTML5, not just for Internet Explorer like Web Thin Client. The configuration process consisted of two parts: the configuration of Internet Information Services and the configuration of Mobile Access configuration.

First of all, Internet Information Services (IIS) Manager was installed for the IIS configuration. IIS is an extensible web server created by Microsoft and it was used to interact with the interface in the project. After IIS Manager had been installed, a website for the interface was added in the Connections window in IIS Manager. Figure 29 illustrates the Add Website window and as can be seen from the picture, the Physical path is the address to access the project's web folder in the computer which usually has the form of *[...]\Documents\Wonderware InduSoft Web Studio Educational v8.0 Projects\[project's name]\Web*.

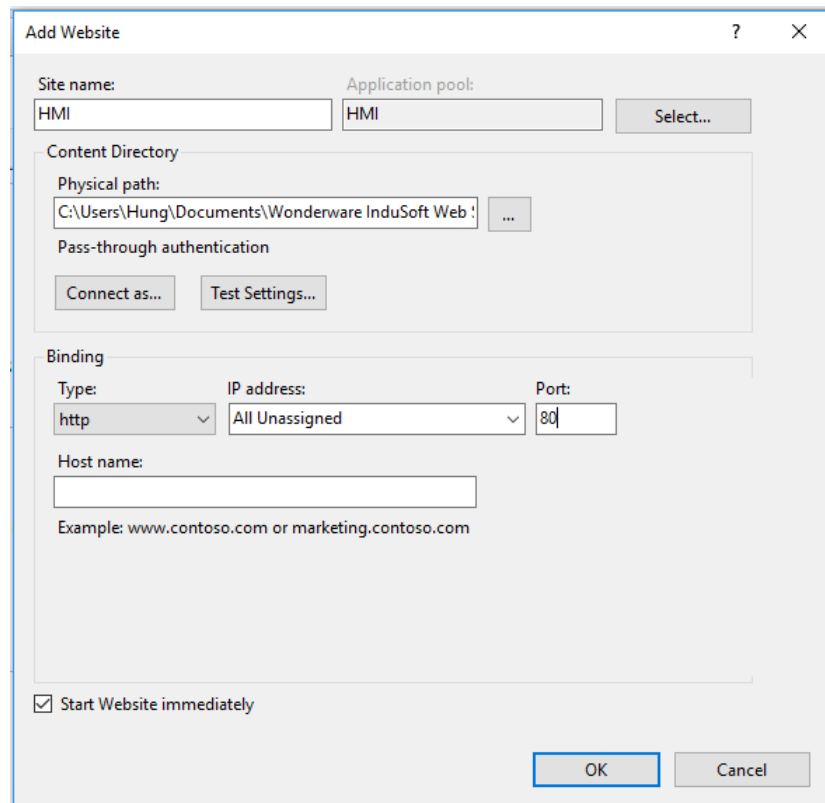


Figure 29. Add Website window.

“Test Settings...” is used to check if IIS has permission to access the project. If there is security on the project folder, the test will fail and the permission must be given to IIS in “Connect as...”. After the website had been added, the Connections window looked as shown in Figure 30.

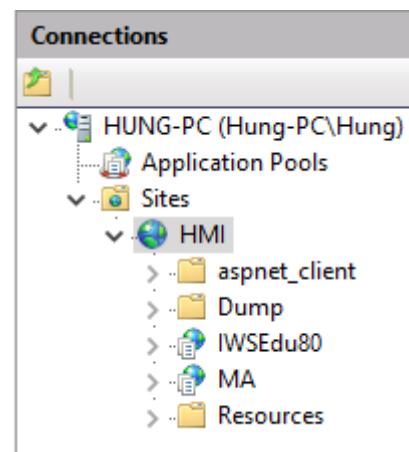


Figure 30. Connections window.



After this, MIME Types were configured in IIS Manager. MIME (Multipurpose Internet Mail Extensions) Types are extensions and related content types that are served as static files. This configuration helped the web browser to handle files received from a server. In this project, MIME Types were extensions found in the Web folder of the project. Below is the list of extensions which were added in the MIME Types configuration:

- .app
- .bin
- .csv
- .gis
- .html
- .ico
- .ini
- .lst
- .rtgis
- .scf
- .scr
- .sg
- .stmp
- .tra
- .txt.

Finally, all the configuration for IIS were conducted and the website for the interface could be started from the Manage Web Site window in the IIS Manager.

Going back to Indusoft Web Studio, before the Mobile Access configuration, all the screens were published as HTML by File → Publish → Save All As HTML (Note: All the screens needed to be closed before publishing). Next, Mobile Access configuration could be opened from Project Explorer → Graphics → Thin Clients → Mobile Access. In this configuration window, there are multiple features related to trends, alarms. However, only the screens feature which sets up for displaying the screens on web browser was taken into use. At the bottom of the configuration window, there is the screens setting. In the Screen column, the main screen was chosen because it was the start-up screen of the whole interface and all the other screens could be accessed from the Main screen. After this, all the configurations for Mobile Access were done and the interface could be accessed from any HTML5 supporting web browser via URL *http://localhost/iwsedu80* or *http://[IPv4 address]/iwsedu80*.

5 INTEGRATION BETWEEN TWINCAT AND INDUSOFT WEB STUDIO

Before going to the integration, software architecture for the project will be introduced briefly. The project included two software: TwinCat and Indusoft Web Studio and the architecture was that both Indusoft Web Studio and TwinCat were placed on same computer and run at the same time. By doing this, the integration process was simplified noticeably. When both pieces of software were installed in the same computer, all the files and connecting port are in the same computer, therefore, there was no need for additional connection. Figure 31 illustrates the architecture of the project.

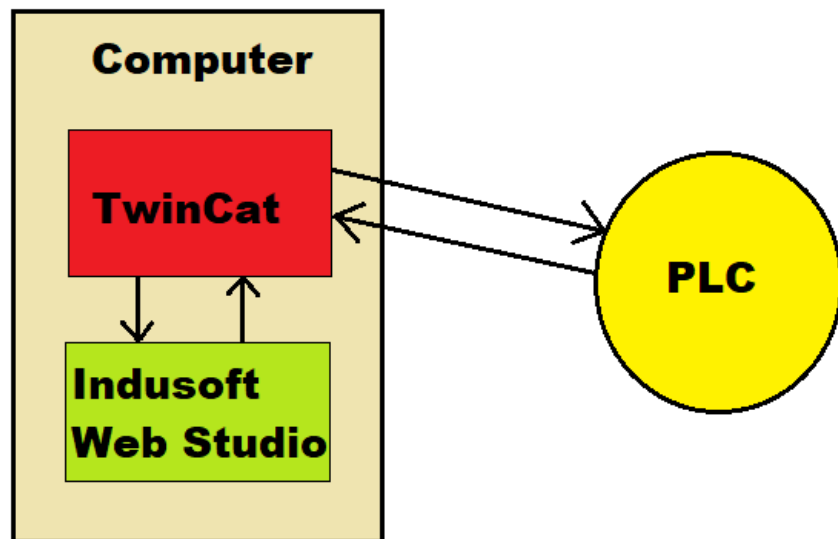


Figure 31. Project architecture.

In order to prepare for the integration between TwinCat and Indusoft Web Studio, a simple TwinCat project which had similar features as the interface, was written and the code was attached to this thesis as Appendix 1. This integration is basically a connection between the Indusoft Web Studio tags and TwinCat variables. After the connection, the stages of the tags in the interface will affect directly the variables in the PLC program.

Firstly, the Tag Integration Source window was opened from the Project Settings window by clicking the “Add...” button in Tag Integration area. The information in the window was given as Figure 32 indicates.

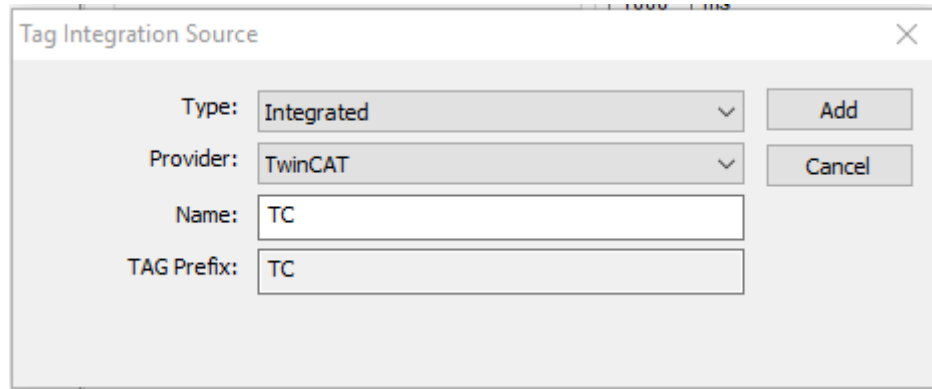


Figure 32. Tag Integration Source window.

After all the needed information had been given and the source had been added to the system, a TwinCat Interface Configuration window was displayed. This window required AMS Net ID of the TwinCat PLC, port number by which the two programs communicated and the symbol file of the TwinCat project. The AMS Net ID was obtained from the System information in the TwinCat project. Custom Port was chosen with port number 851 as TwinCat3 Runtime. Lastly, the TwinCat project's symbol file with the extension .tpy in the project's folder was given. Figure 33 illustrates these steps.

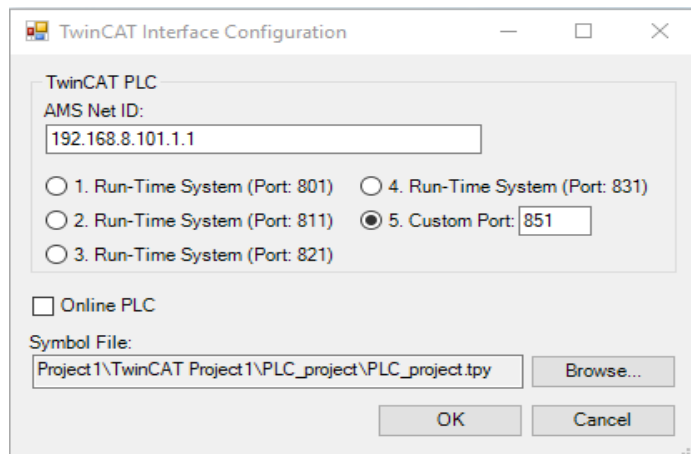


Figure 33. TwinCat Interface Configuration window.

Consequently, a list of TwinCat variables was imported to the shared database of Indusoft Web Studio. Afterwards, objects in the interface were linked to respective TwinCat variables by replacing old tag names with TwinCat variables. Finally, the integration was finished and every action in the interface would affect PLC program's variables.

6 CONCLUSION

Finally, the project was completed and it met all requirements set by the commissioner. The project had its own advantages and disadvantages.

One of the most noticeable advantages was that the interface was programmed to be able to run on a web browser thanks to the Mobile Access thin client of Indusoft Web Studio, therefore, it can be used by any device which has a browser supporting HTML5. As a result, users do not have to think of devices compatible with the interface. Additionally, users can enjoy using the interface without worrying about wiring and distance since it is controlled via WLAN. Lastly, the mode feature, which is a combination of many different control operations on household appliances, plays a significant role in the interface. The feature increases the convenience of the interface since it minimizes actions on controlling electronic devices.

On the other hand, one of the disadvantages of being controlled via a wireless network is that it causes a short delay for the signal to arrive at its destination, hence, the interface does not react immediately when an action is made. Consequently, it produces a slightly inconvenient experience for its users. Moreover, the Mobile Access thin client of Indusoft Web Studio is based on WLAN. Hence, users can access the interface only when they are in the same network as the interface. Lastly, the security system of the application is still quite simple in a sense that if a user forgets to log out, the interface will not log out automatically even though it is shut down. This leads to a problem so that when the next user turns the interface on again, he or she still has full access to the interface without entering a correct user name and password.

Finally, the author would like to suggest some possible improvements for the interface:

- The remote access feature of the interface was done via Mobile Access thin client and one of the most crucial features of this client is interacting with alarms. Therefore, an alarm feature could be developed for the interface. The idea is that users will be notified immediately if there is a problem, hence, they can take action in time so that the device will not be seriously damaged.
- The visualization of the interface in the project was just a modest demonstration and it could be improved significantly by the animation feature of Indusoft Web Studio. This improvement will help users get a better understanding of the interface and a more pleasant experience.

REFERENCES

Beckhoff CX90x0: Product overview [Image]. Retrieved from Beckhoff Automation website

https://infosys.beckhoff.com/english.php?content=../content/1033/cx9000_hw/html/cx9000_prosystem.htm&id

Hendricks, D. (April 22, 2014). The History of Smart Homes. Retrieved from <http://www.iotevolutionworld.com/m2m/articles/376816-history-smart-homes.htm>

History of PLC. (n.d.). Retrieved from

<http://library.automationdirect.com/history-of-the-plc/>

Indusoft Web Studio Product Feature (n.d.). Retrieved from Wonderware by Schneider Electric <http://www.indusoft.com/Products-Downloads/HMI-Software/InduSoft-Web-Studio>

Introduction to PLCs [Lecture notes in PFD]. (2006). Retrieved from <http://www.srmuniv.ac.in/sites/default/files/files/IC0403-ccp-2.pdf>

Main screen background [Image]. Retrieved from

<http://www.jukkatalo.fi/mallisto/huippukodit-2016/115-10k/>

Mandel, T. (1997). *The Elements of User Interface Design*. New York City: John Wiley and Sons.

PLC languages. (n.d.). Retrieved from Kronotech Instrumentation and Control <http://www.kronotech.com/PLC/Languages.htm>

PLC scanning operation [Image]. Retrieved from ACC Automation website <http://accautomation.ca/wp-content/uploads/2016/07/PLC-Program-Scan-005-min.png>

Smart Home. (October 30, 2016). Retrieved from

<http://cctvinstitute.co.uk/smart-home/>

TwinCAT PLC Control Visualization. (n.d.). Retrieved from Beckhoff Automation

https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc_control/html/tcplcvisu_intro.htm&id=23202

TWINCAT PROJECT CODE

Main program

```

PROGRAM MAIN
VAR

END_VAR

LivingRoom();
Kitchen();
BedRoom();
Garage();
Outside();

```

Bed room

```

PROGRAM BedRoom
VAR
END_VAR

VAR_INPUT
    BR_CeillingLightSwitch AT %I*: BOOL;
    BR_DimLightSlider AT %I*: INT;
    BR_NightLampSwitch AT %I*: BOOL;
    BR_TVRemote AT %I*: BOOL;
    BR_WindowBlindSwitch AT %I*: BOOL;
    BR_TempInput AT %I*: REAL;
    BR_TempControllInput AT %I*: REAL;
END_VAR
VAR_OUTPUT
    BR_CeillingLight AT %Q*: BOOL;
    BR_DimLight AT %Q*: INT;
    BR_NightLamp AT %Q*: BOOL;
    BR_TV AT %Q*: BOOL;
    BR_WindowBlind AT %Q*: BOOL;
    BR_Temp AT %Q*: REAL;
    BR_TempControl AT %Q*: REAL;
END_VAR

// Ceilling light
IF BR_CeillingLightSwitch = TRUE THEN
    BR_CeillingLight := TRUE;
ELSE
    BR_CeillingLight := FALSE;
END_IF

```

```

// Dim light
IF BR_DimLightSlider = 0 THEN
    BR_DimLight := 0;
ELSIF BR_DimLightSlider = 1 THEN
    BR_DimLight := 1;
ELSIF BR_DimLightSlider = 2 THEN
    BR_DimLight := 2;
ELSIF BR_DimLightSlider = 3 THEN
    BR_DimLight := 3;
END_IF

// Night lamp
IF BR_NightLampSwitch = TRUE THEN
    BR_NightLamp := TRUE;
ELSE
    BR_NightLamp := FALSE;
END_IF

// TV
IF BR_TVRemote = TRUE THEN
    BR_TV := TRUE;
ELSE
    BR_TV := FALSE;
END_IF

// Window blind
IF BR_WindowBlindSwitch = TRUE THEN
    BR_WindowBlind := TRUE;
ELSE
    BR_WindowBlind := FALSE;
END_IF

// Temperature
BR_Temp := BR_TempInput;
BR_TempControl := BR_TempControlInput;

```

Living room

```

PROGRAM LivingRoom

VAR_INPUT
    LR_CeillingLightSwitch AT %I*: BOOL;
    LR_DimLightSlider AT %I*: INT;
    LR_TableLampSwitch AT %I*: BOOL;
    LR_StandingLampSwitch AT %I*: BOOL;
    LR_WindowBlindSwitch AT %I*: BOOL;
    LR_TempInput AT %I*: REAL;
    LR_TempControlInput AT %I*: REAL;
END_VAR

```



```
VAR_OUTPUT
    LR_CeillingLight AT %Q*: BOOL;
    LR_DimLight AT %Q*: INT;
    LR_TableLamp AT %Q*: BOOL;
    LR_StandingLamp AT %Q*: BOOL;
    LR_WindowBlind AT %Q*: BOOL;
    LR_Temp AT %Q*: REAL;
    LR_TempControl AT %Q*: REAL;
END_VAR

// Ceilling light
IF LR_CeillingLightSwitch = TRUE THEN
    LR_CeillingLight := TRUE;
ELSE
    LR_CeillingLight := FALSE;
END_IF

//Dim light
IF LR_DimLightSlider = 0 THEN
    LR_DimLight := 0;
ELSIF LR_DimLightSlider = 1 THEN
    LR_DimLight := 1;
ELSIF LR_DimLightSlider = 2 THEN
    LR_DimLight := 2;
ELSIF LR_DimLightSlider = 3 THEN
    LR_DimLight := 3;
END_IF

// Table lamp
IF LR_TableLampSwitch = TRUE THEN
    LR_TableLamp := TRUE;
ELSE
    LR_TableLamp := FALSE;
END_IF

// Standing light
IF LR_StandingLampSwitch = TRUE THEN
    LR_StandingLamp := TRUE;
ELSE
    LR_StandingLamp := FALSE;
END_IF

// Window blind
IF LR_WindowBlindSwitch = TRUE THEN
    LR_WindowBlind := TRUE;
ELSE
    LR_WindowBlind := FALSE;
END_IF
```

```
//Temperature
LR_Temp := LR_TempInput;
LR_TempControl := LR_TempControlInput;
```

Kitchen

```
PROGRAM Kitchen
VAR
END_VAR

VAR_INPUT
    K_CeillingLightSwitch AT %I*: BOOL;
    K_WindowBlindSwitch AT %I*: BOOL;
    K_SinkLightSwitch AT %I*: BOOL;
    K_StoveLightSwitch AT %I*: BOOL;
    K_TempInput AT %I*: REAL;
    K_TempControlInput AT %I*: REAL;
END_VAR
VAR_OUTPUT
    K_CeillingLight AT %Q*: BOOL;
    K_WindowBlind AT %Q*: BOOL;
    K_SinkLight AT %Q*: BOOL;
    K_StoveLight AT %Q*: BOOL;
    K_Temp AT %Q*: REAL;
    K_TempControl AT %Q*: REAL;
END_VAR

// Ceilling light
IF K_CeillingLightSwitch = TRUE THEN
    K_CeillingLight := TRUE;
ELSE
    K_CeillingLight := FALSE;
END_IF

// Window Blind
IF K_WindowBlindSwitch = TRUE THEN
    K_WindowBlind := TRUE;
ELSE
    K_WindowBlind := FALSE;
END_IF

// Sink light
IF K_SinkLightSwitch = TRUE THEN
    K_SinkLight := TRUE;
ELSE
    K_SinkLight := FALSE;
```

```

END_IF

// Stove light
IF K_StoveLightSwitch = TRUE THEN
    K_StoveLight := TRUE;
ELSE
    K_StoveLight := FALSE;
END_IF

// Temperature
K_Temp := K_TempInput;
K_TempControl := K_TempControlInput;

```

Garage

```

PROGRAM Garage
VAR
END_VAR

VAR_INPUT
    G_CeillingLightSwitch AT %I*: BOOL;
    G_DoorSwitch AT %I*: BOOL;
END_VAR
VAR_OUTPUT
    G_CeillingLight AT %Q*: BOOL;
    G_Door AT %Q*: BOOL;
END_VAR

// Ceilling light
IF G_CeillingLightSwitch = TRUE THEN
    G_CeillingLight := TRUE;
ELSE
    G_CeillingLight := FALSE;
END_IF

// Door
IF G_DoorSwitch = TRUE THEN
    G_Door := TRUE;
ELSE
    G_Door := FALSE;
END_IF

```

Outside

```
PROGRAM Outside
```

```
VAR
END_VAR

VAR_OUTPUT
    O_Temp AT %Q*: REAL;
END_VAR
VAR_INPUT
    O_TempInput AT %I*: REAL;
END_VAR

// Temperature
O_Temp := O_TempInput;
```