

# A prototype home automation project using the Project54 system



Florent RIVREAU  


## **ACKNOWLEDGEMENTS**

I would like to thank Dr. Kun not only for giving me the opportunity to work on this project but also for his guidance and the advice he gave to me.

I would also like to thank the other members of Project54 for taking the time to help me each time I have some problems with hardware, software, or Shakespeare language.

Finally I would like to thank my friend Julien Nouhant who helped me to find this job.

## TABLE OF CONTENTS

<b>1. Introduction</b>	<b>4</b>
<b>2. Project preview</b>	<b>5</b>
<b>2.1. Introduction</b>	<b>5</b>
<b>2.2. Project functionalities</b>	<b>5</b>
<b>2.3. Architectures for Home Automation</b>	<b>6</b>
2.3.1. The Multi-Agent Architecture of Peter A. Heeman	6
2.3.2. The prototype using Project54	6
<b>3. Project status overview</b>	<b>9</b>
<b>3.1. Introduction</b>	<b>9</b>
<b>3.2. User Manual</b>	<b>10</b>
3.2.1. Main Screen:	10
3.2.2. Buttons used in all windows:	11
3.2.3. House Main Window:	12
3.2.4. <i>Define Devices</i> window:	13
3.2.5. <i>House Control</i> window:	14
3.2.6. <i>House Status</i> window:	15
3.2.7. How to use the interface:	18
3.2.8. Conclusion:	18
<b>3.3. The CM11A interface</b>	<b>19</b>
3.3.1. Introduction	19
3.3.2. How to control X10 devices	19
3.3.3. How to process the feedback	22
3.3.4. How to react when the CM11A sends a message	23
<b>3.4. The code</b>	<b>24</b>
3.4.1. Introduction	24
3.4.2. Naming conventions:	24
3.4.3. Files	27
<b>4. Encountered problems</b>	<b>28</b>
<b>5. Project future improvements</b>	<b>28</b>
<b>6. Conclusion</b>	<b>29</b>
<b>7. References</b>	<b>30</b>

# 1. Introduction

Who has never dreamt to control the TV set with the voice instead of using a remote control? Who has never dreamt to have a cup of coffee already prepared for breakfast? Who has never dreamt to have windows shades that close automatically in the evening? Nowadays, home automation is not a dream anymore and in the home of the future, there will be multiple devices that can be controlled by a computer. That is why the subject of this project is the conception of a Prototype for Home Automation.

Of course, a lot of people have already worked on such a project and many systems were designed. However they all present some disadvantages and do not fulfill the high expectations of the dreamers we are. During this project, a prototype that will give some solutions to these problems, will be designed.

To do so, the first step is naturally to find documentation dealing with the systems that have already been designed and to find their drawbacks in order to determine what we should do to make them cleverer or more user friendly or easier to install, etc. The second step is to determine what, in the Projet54 software and hardware could be helpful to bring a maximum of solutions to these problems.

You will find in the following pages what was done, how, why and what should be done next. With the understanding that this project will continue, the code has been written as readable as possible. That was time consuming, but necessary in order to make it very easy to understand. Also if you would like to work on this project, this document should be read carefully.

## **2. Project preview**

### **2.1. Introduction**

The subject of this project, being relatively vast and especially unbounded, some time was devoted to imagine what could be feasible and to determine the functionalities which would be most useful for future users.

A survey of ten persons taking part in Project54 was carried out to try to determine the most significant functions to implement, those that should be possible to realize later. Insofar as the current goal is to produce a prototype, this should be sufficient. However, before including new functionalities, it would be essential to do a new survey on a larger and more representative sample of the future users. What will follow is the result of this study.

### **2.2. Project functionalities**

The system will have to implement functionalities which will be controlled by a vocal and manual user interface. It will be possible to program the various devices behaviors from the computer.

The functionalities which will have to be implemented in the prototype are:

- The control of lamps
- The control of the heating system
- The control of the burglar alarm

The interface will not only have to allow the control of various devices, but also to visualize their state in real time. The vocal control device will use feedback to let the user know what is happening.

The computer interface will be able to:

- Allow the user to control all systems via the computer in real time
- Allow the user to program the system behavior in advance
- Show the current status of the system

## **2.3. Architectures for Home Automation**

### **2.3.1. The Multi-Agent Architecture of Peter A. Heeman**

The Multi-Agent Architecture for Home Automation designed by Peter A. Heeman has been the principal source of information for this project [1]. This is the system which was studied before creating a prototype.

To summarize all the capacities of this system, please see Figure 1.

As you can see the system allows using three types of devices:

- X10 devices to control lamps and other devices [2] [3] [4].
- Weeder Technology Devices to control light sensors, hard wired motion detector and a sprinkler system.
- iButton devices to detect temperature.

There are many computers able to distribute the information between each other via a network and so react in a clever way. It does not matter if they are running different operating systems.

In many ways, this system is really good. However it presents one drawback: each type of devices uses the RS232 serial port which means that the system requires one serial port per peripheral type. In fact it uses three COM ports for three types of devices.

### **2.3.2. The prototype using Project54**

For communication the Project54 uses the “Intelligent Transportation Data Bus” (IDB). The IDB allows using only one RS232 serial port to control many peripherals.

An IDB bus is composed of IDB boxes. IDB boxes are connected to each other through CAT5 network cable and each one has a RS232 port. One of the IDB boxes is connected to the computer [5] [6] [7].

By using IDB it is possible to solve one of the problems of the system described previously.

As you will see, the first prototype designed is extremely simple (see Figure 2). It is only able to control X10 devices. However it can be easily improved to allow using new devices. The system is centralized on one computer, but this also can be improved later (see Figure 3).

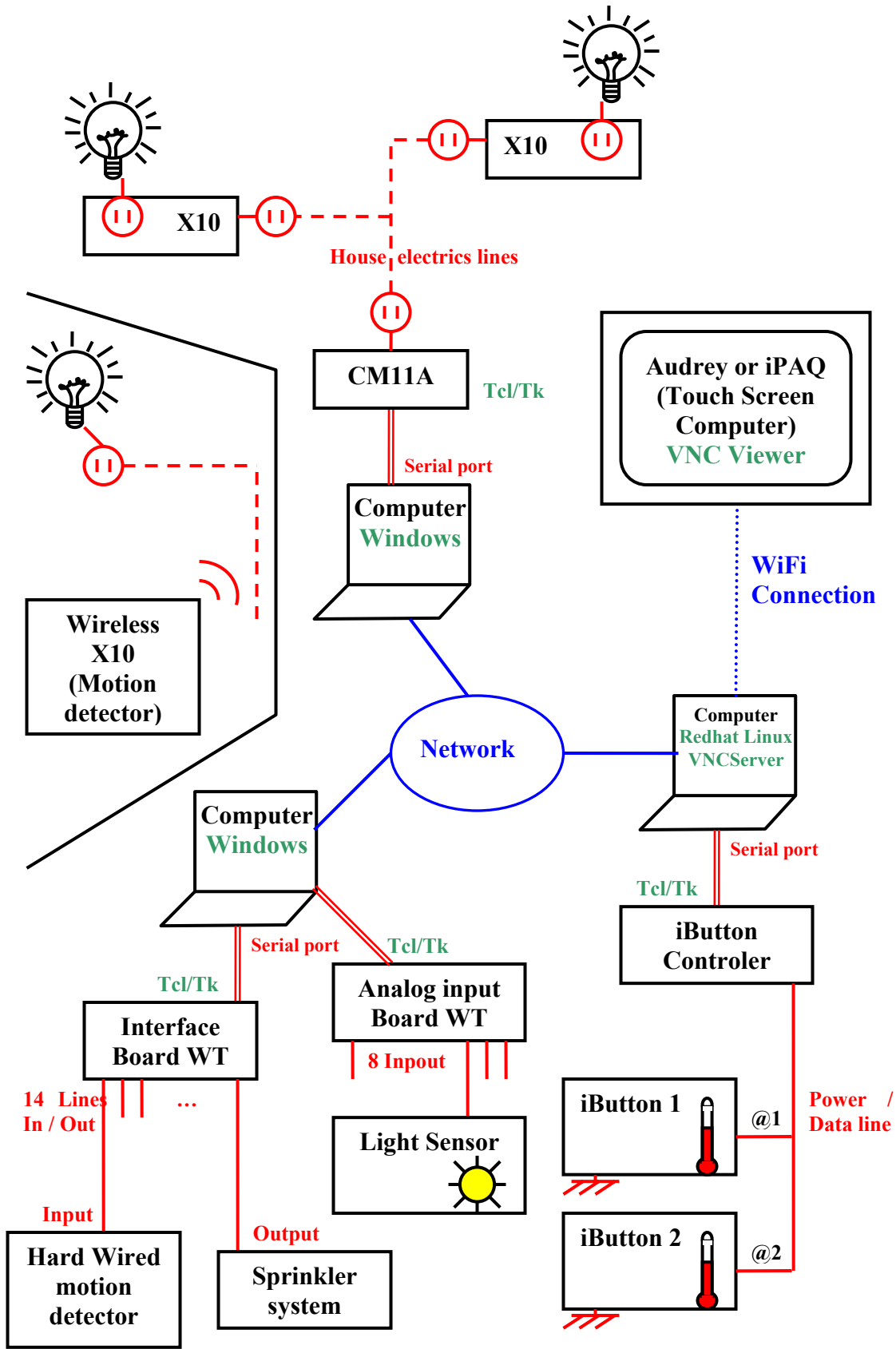
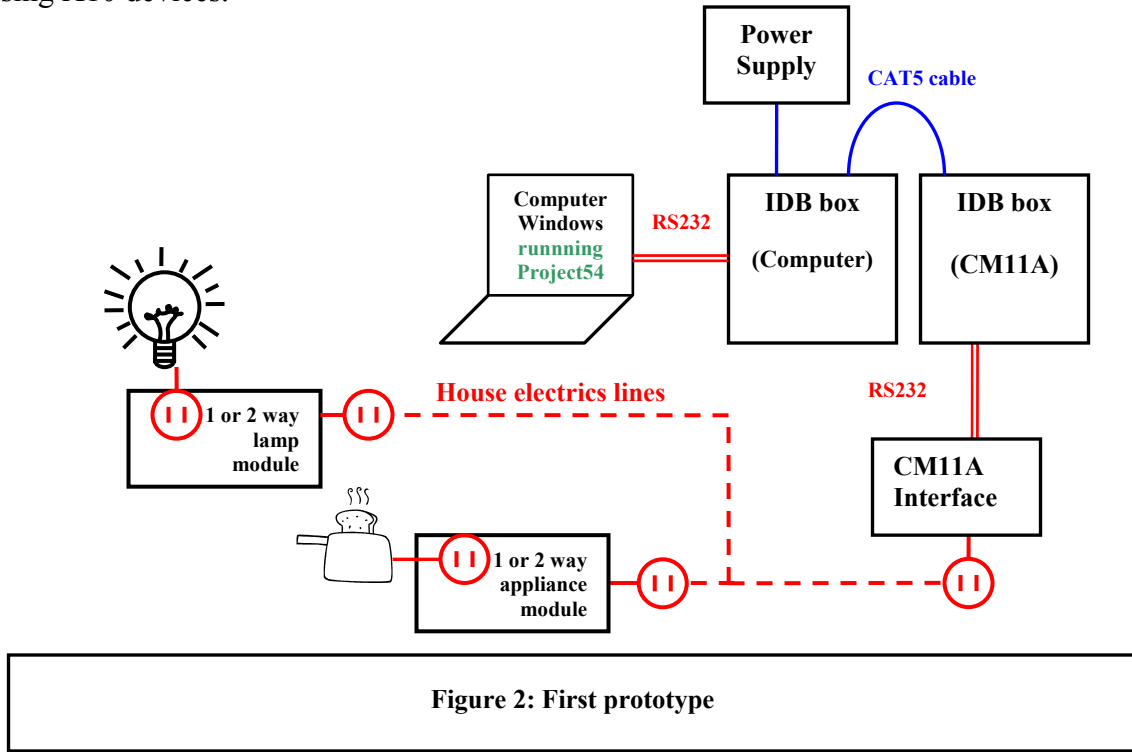
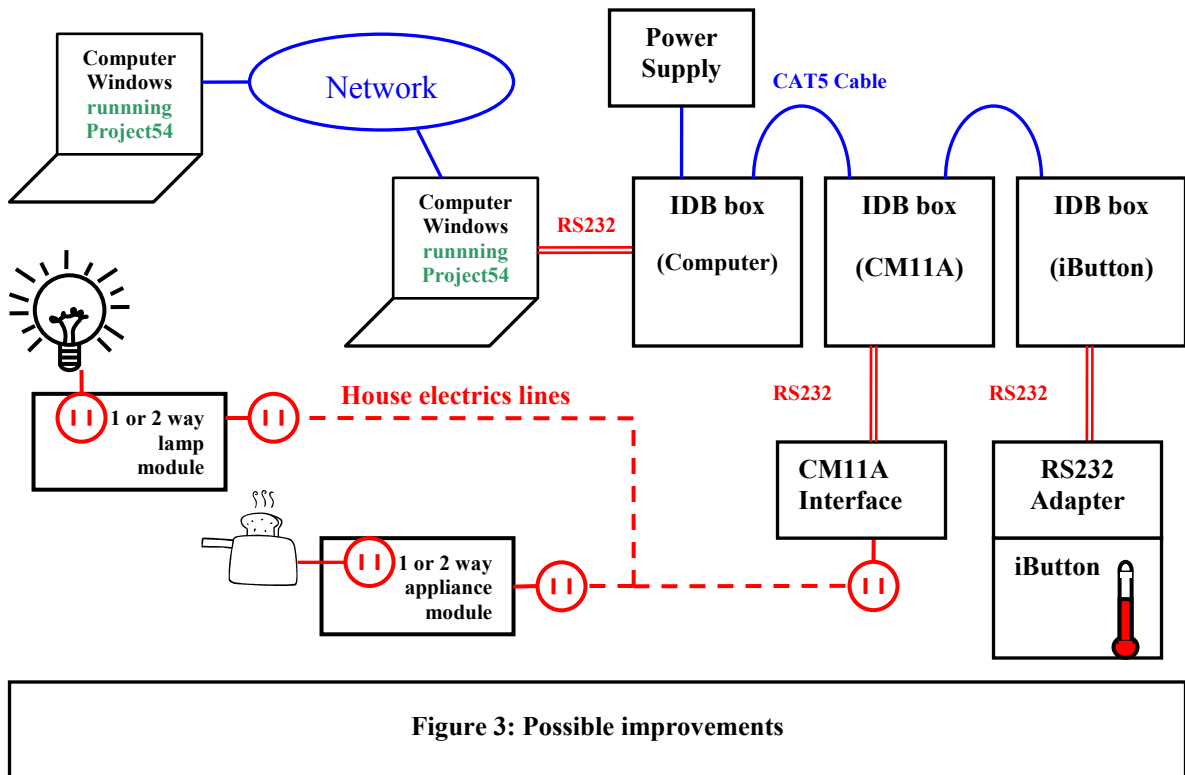


Figure 1: Multi-Agent Architecture for Home Automation (System used by Peter A. Heeman)

Below is the first prototype I designed. As you can see it is very simple and only allows using X10 devices.



Below is an example of how to improve the system. As you can see, to use another peripheral (like an iButton) the only thing to do is to have one more IDB box. It is also possible to control the devices from another computer communicating with the first one by using the Proxy Application of Project54.





## **3. Project status overview**

### **3.1. Introduction**

The system hardware corresponds to the first prototype designed (see section 2.3.2) and is currently composed of:

- A computer running Project54
- Two IDB boxes
- A CM11A interface that allow controlling the X10 devices
- 3 X10 2 way appliance modules
- 2 X10 2 way lamp modules

The computer interface is currently able to:

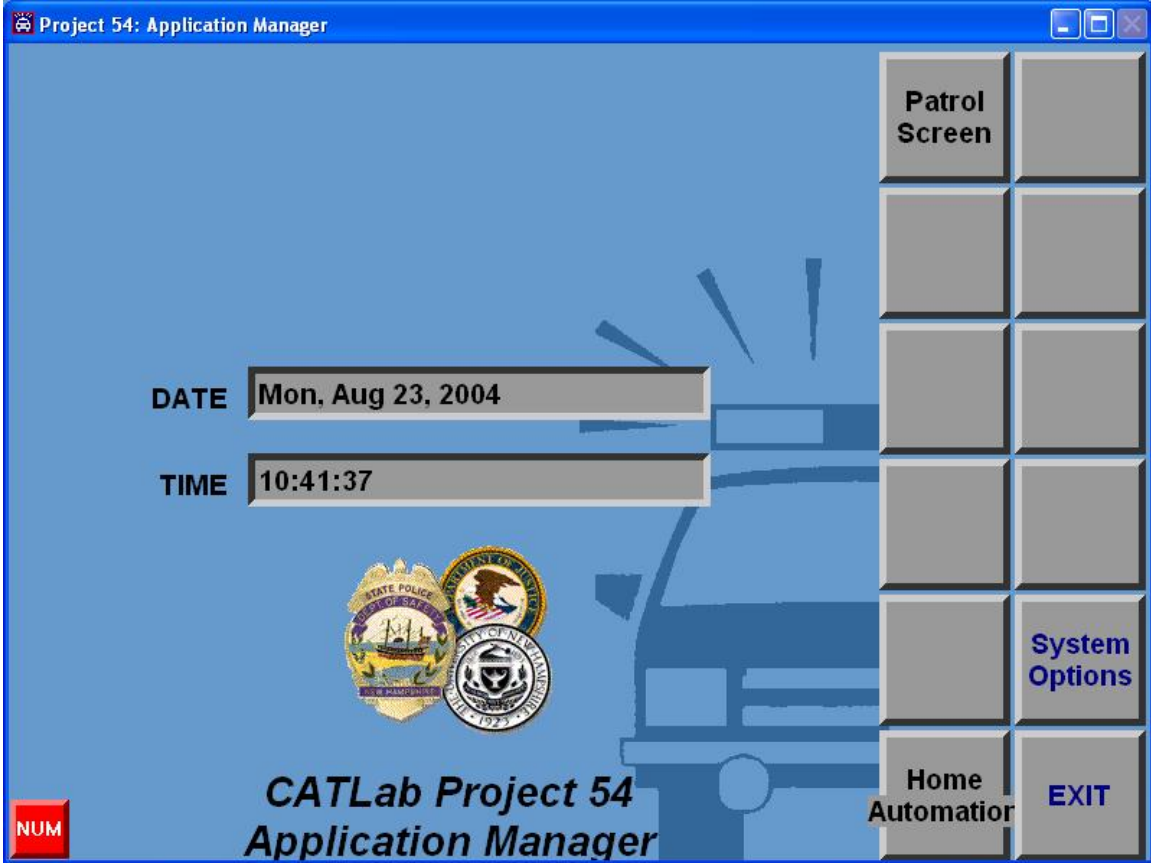
- Allow the user to control all systems via the computer in real time
- Show the current status of the system

You will find in the following more details dealing with the current status of the project. First the User manual will help you to discover the prototype's capacities. Then the part dealing with the code and the communication will help you to understand what has been done and how to improve the project by adding new functionalities.

## 3.2. User Manual

### 3.2.1. Main Screen:

As you can see on the picture below, this is the default main screen of Project54.



From this window, click on the button:

- ▶ **Home Automation:** to access to the Home Automation main window.
- ▶ **Exit:** to access to the menu you can see below:



Then click on:

- ▶ **Power Off:** to shutdown the computer
- ▶ **Exit:** to quit.
- ▶ **Cancel:** to return to the Main screen.

### 3.2.2. Buttons used in all windows:

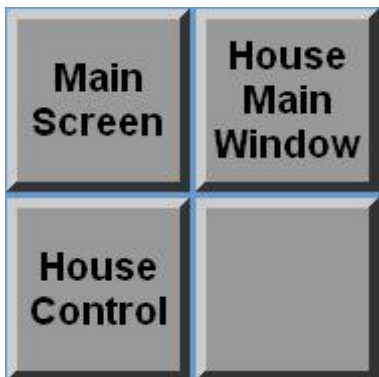
When you access the Home Automation application, you will always find these 4 buttons on the top right corner.



The user can click on:

- ▶ **Main Screen:** to access the “Main Screen” window.
- ▶ **House Main Window:** to access the “House Main Window” window.
- ▶ **House Control:** to access the “House Control” window.
- ▶ **House Status:** to access the “House Status” window.

In most of the cases you will be in one of these windows, which implies that the button corresponding to it will not be present in the interface. In that case the interface will look like this:



The user can click on:

- ▶ **Main Screen**
- ▶ **House Main Window**
- ▶ **House Control**

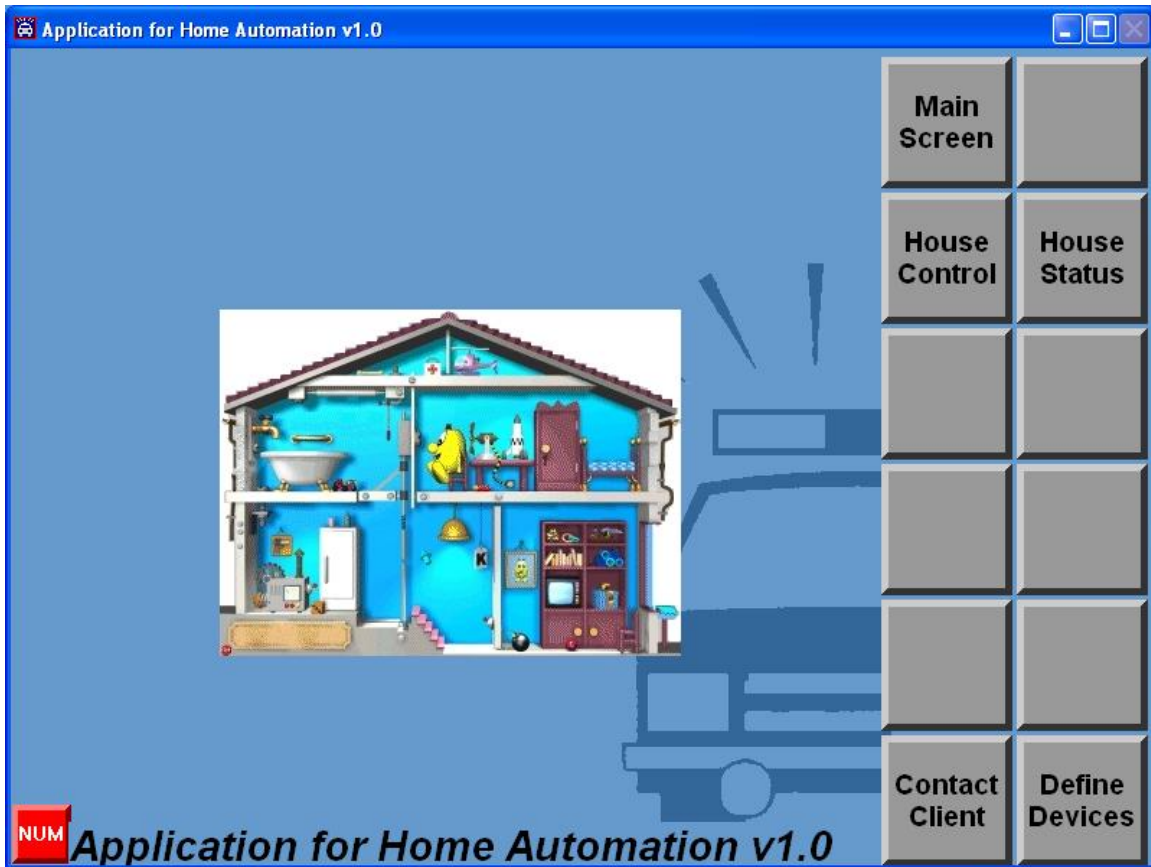
But cannot click on:

- ▶ **House Status**

This is because he is already in the House Status window.

### 3.2.3. House Main Window:

This window allows the user to navigate from window to window.



From this window, click on the button:

► **Contact Client:** to test the communications with another computer. If the communication works properly, the label “Application for Home Automation v1.0” will change to “The client has answered”. (This is just a test of the communication using the Project54 Proxy Application).

► **Define Devices:** to access the “Define Devices” window.

*Note1: if at the beginning there is a problem of communication with the IDB box, the label “Application for Home Automation v1.0” will change to “IDB didn’t startup properly”.*

### 3.2.4. Define Devices window:

It is only possible to access this window from the House Main Window. It allows the user to declare all the devices he or she is going to control. Users should be very careful as they declare the devices, because if it is not done correctly, the feedback processing will be incorrect although the user will still be able to control the devices.

**Define Devices**

**MODULE'S ADDRESS**

House Code (from A to P) :

Device Code (from 1 to 16) :

**MODULE'S CHARACTERISTICS**

Room name :

Device name :

Provide feedback (Y/N) :

Lamp module (Y/N) :

**NUM**

**Define Devices**

Main Screen

House Main Window

House Control

House Status

Edit Name File

Delete All Names

From this window, click on the button:

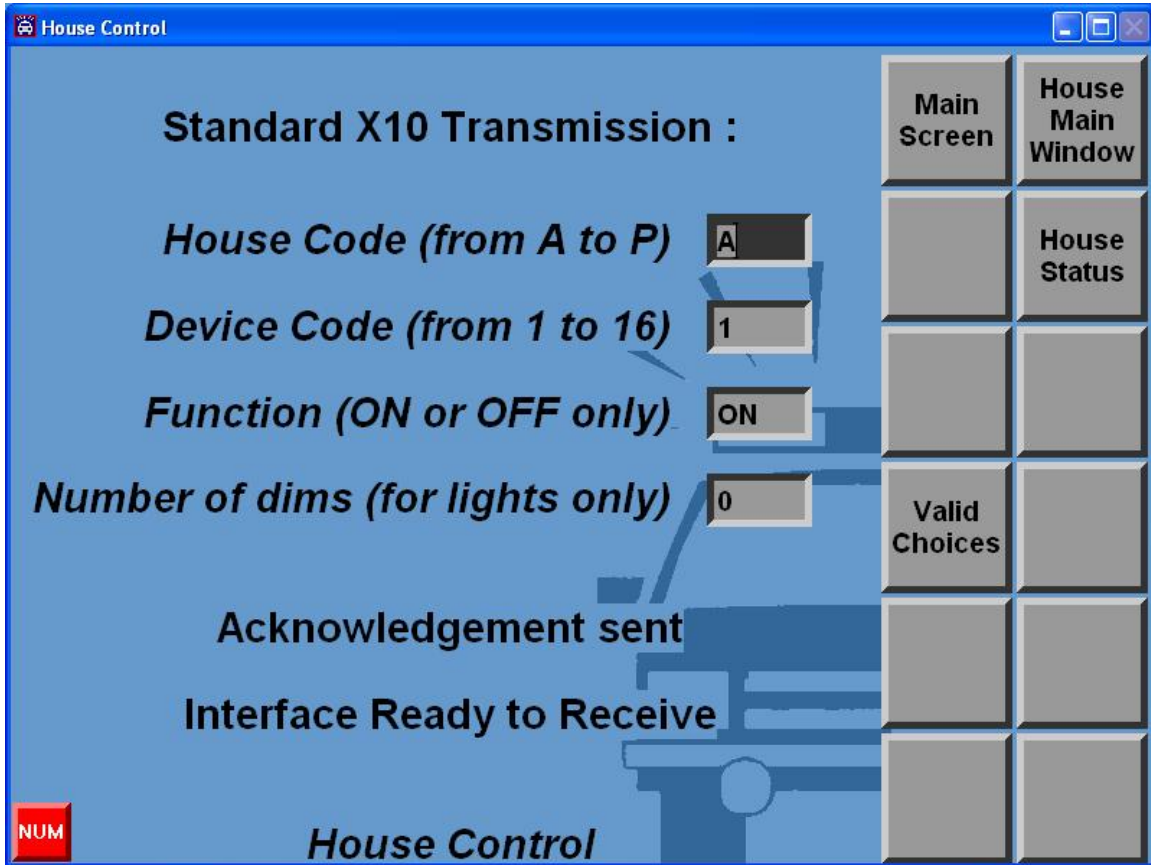
► **Edit Name File:** to replace the room and device name by default of the selected device with the text of the text fields (10 characters maximum) in the “names.txt” file (see section 3.4.3.2 for more information about this file). If the user forgets to write something in the text field, the name will be set to the address. If the device is controlled by a 2-way module, the “Provide feedback (Y/N):” text field must be set to “Y” otherwise “N”. If the module is a lamp module, the “Lamp module (Y/N):” text field must be set to “Y” otherwise “N”.

► **Delete All Names:** to reset the “names.txt” file to the default names (i.e.: the room name is the House code and the device name is the device code).

*Note2: The feedback will not work until all the devices are defined properly (i.e.: each address of the defined devices must correspond to the module address of the physical system and the “Provide feedback (Y/N):” and “Lamp module (Y/N):” text fields must correspond to the type of the module).*

### 3.2.5. *House Control window:*

This window allows the user to control the devices in real time.



From this window, click on the button:

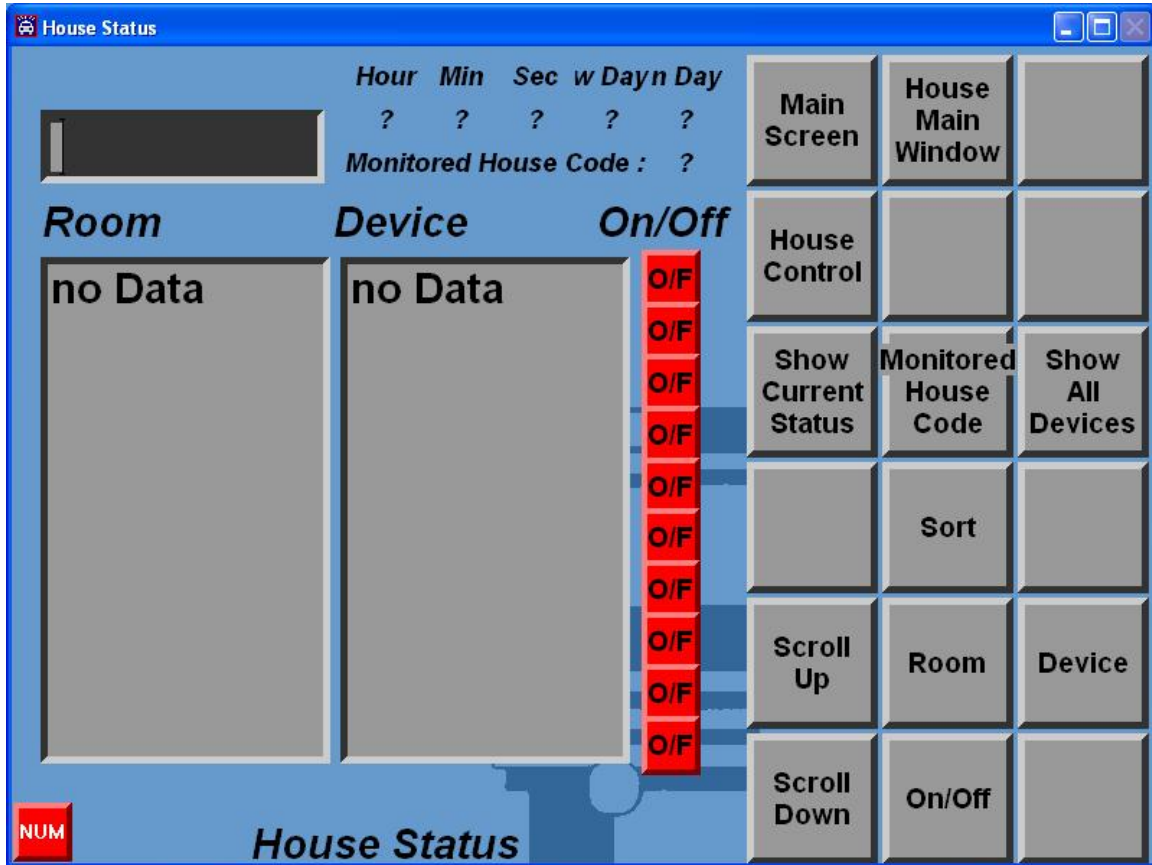
► **Valid Choices:** to send the command corresponding to what is in the text fields to the CM11A interface via the IDB box. Currently it allows the user to use these functions: On (ON), Off (OFF), All Units Off (AUOF), All Lights On (ALON), All Lights Off (ALOF), Dim (DIM) and Bright (BRI)

There are two labels that inform the user of the message exchanged between the computer and the CM11a:

- The first one can take the following values:  
Monitored House Code changed; Standard command sent; Acknowledgement sent.
  - The second one can take these values:  
Checksum correct; Interface ready to Receive.
- (See section 3.3.2 for more information about these messages)

### 3.2.6. *House Status window:*

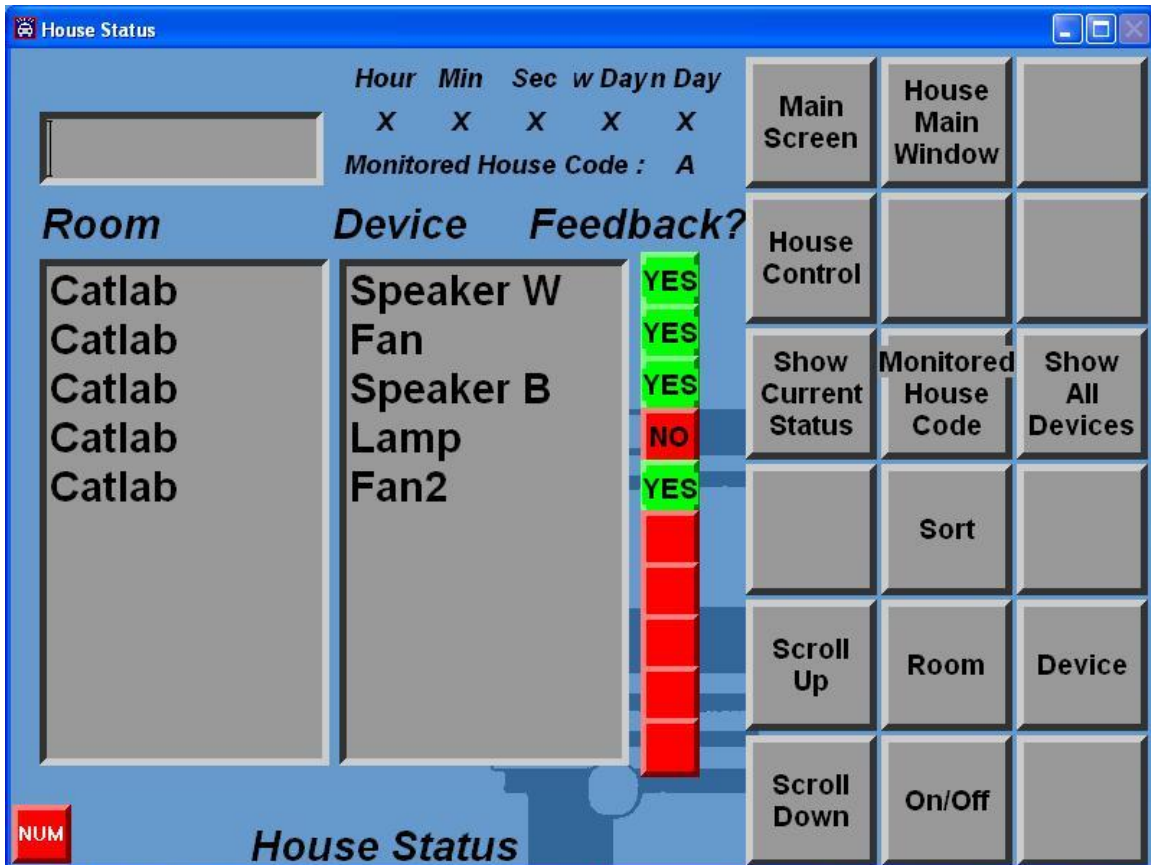
This window allows the user to check the devices status in real time.



From this window, click on the button:

- ▶ **Show Current Status:** to show the date and the status of the devices of the currently monitored house code.
- ▶ **Monitored House Code:** to allow the user to change the currently monitored house code and so to see the status of other devices.
- ▶ **Show All Devices:** to show all the devices that have been defined in the system.
- ▶ **Scroll Up** and **Scroll Down:** to allow seeing all the devices even if there are more than 10.
- ▶ **Other buttons:** allow to sort the devices and make the interface more user friendly. These functions are currently buggy.

Below is an example of the House Status window when the user clicks on “Show All Devices”.

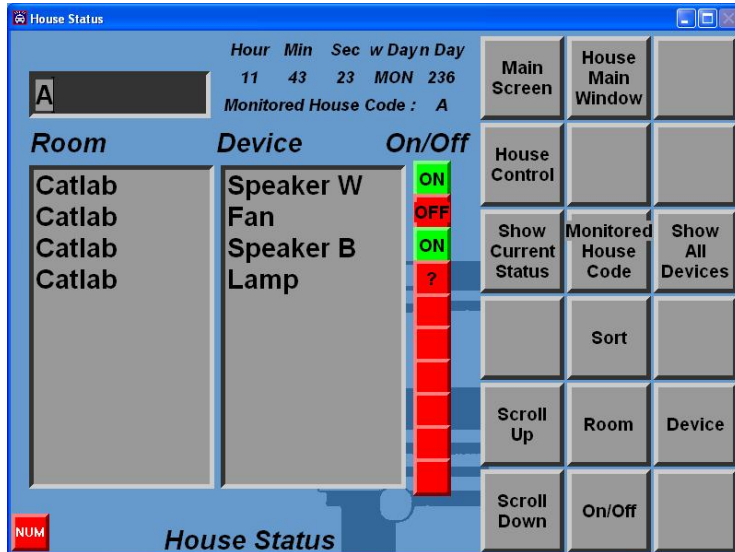


On the left text area we can see the room name which the device is in. (In that example, all the devices are in the same room, but in general that is not the case). On the right text area we can see the device’s names. The status light indicates if the module that controls the device provides feedback or not.

*Note3: the interface does not indicate the address related to the devices. It would be much better if that was the case.*

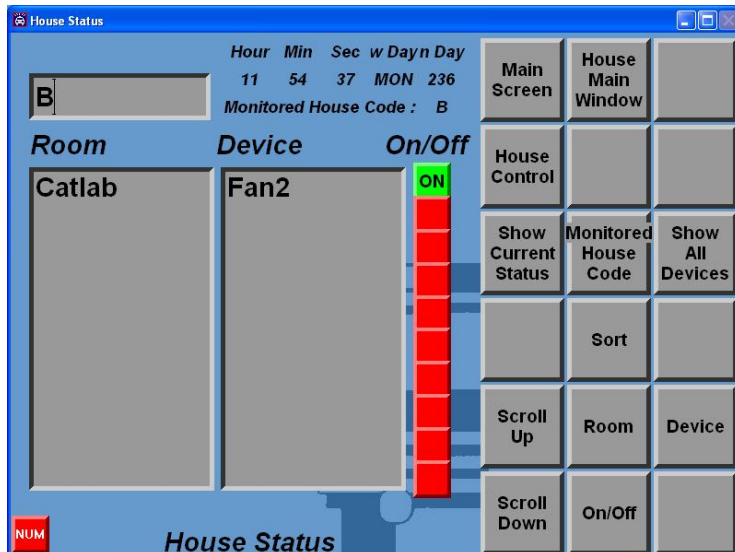


Below is an example of the House Status window when the user clicks on “Show Current Status”:



As you can see, the interface shows the current time stored in the interface. It also shows the currently Monitored House Code. Finally it shows the status of the devices corresponding to this monitored house code. It writes “ON” or “OFF” if the device is controlled by a two way module, and “?”, if it is controlled by a one way module.

The user can of course see the status of other devices. To do so, he or she must enter the character corresponding to the house code in the text field and then click on “Monitored House Code”. Then he or she just needs to click on “Show Current Status” again. Below is an example:



### **3.2.7. How to use the interface:**

The best way to use this interface is to:

- Go to the “Define Devices” window, and define all the devices one by one.
- Go to the “House Status” window and use “Show All Devices” to verify that all the devices have been defined properly.
- Go to the “House Control” window and use the function ON, OFF or AUOF (All Unit Off) in order to control the devices.
- Go back to the “House Status” window and use “Show Current Status” and “Monitored House Code” to see the status of all the devices of a particular house code.

Once the user has tried all of this, he or she knows everything about the interface.

### **3.2.8. Conclusion:**

The interface allows the user to control a device (i.e.: at least to turn it “on” or “off”.) If all the devices have been correctly defined in the define devices window, it allows the user to see the current status of the devices (i.e.: to know if they are “On” or “Off”, or to remind the user that they do not allow feedback.)

### **3.3. The CM11A interface**

#### **3.3.1. Introduction**

The CM11A interface is an interface using the X10 technology. The term X10 refers to a protocol invented by a Scottish company, Pico Electronics, in 1978. This X10 protocol provides a method of sending signals across existing electrical wiring to control compatible modules. This allows X10 modules to be installed easily and cheaply. These signals do not interfere with the normal operation of the electrical appliances [2] [3].

A typical X10 set-up consists of a transmitter module, and several receiver modules. The CM11A transmitter was chosen because it allows to send status requests and also to get feedback.

To create the functions that allow communication with the interface, I used both the official documentation of the CM11A interface [2] and sniffed the messages exchanged between the computer and these interfaces while the ActiveHome software was running. The problem is that it is not so easy to understand what should be done by just sniffing what happened. Moreover, another difficulty came from the existence of some mistakes in the official documentation. I found another document in which most of these mistakes are corrected which is very helpful [3].

I will now explain how I communicate with the interface and why I am doing it like this.

#### **3.3.2. How to control X10 devices**

To control the X10 devices, the software uses the “standard transmission” described in details in the “Interface Communication Protocol” (section 3).

Here is what happens when the user clicks on “Valid Choices” in the “House Control” window:

- 1) The parameters entered by the user in the interface are converted into integer values that are used by the `void idbcom_standardCommand(int numberOfDim, int houseCode, int deviceCode, int function)` function.
- 2) This function is called.
- 3) It checks if “houseCode” is equal to the global variable “i\_currentlyMonitoredHouseCode”.(see Note4) If that is not the case, it calls the function `void idbcom_sendCurrentTime_and_MonitoredHouseCode( int i_monitoredHousecode );` with houseCode as parameter.
- 4) It checks if “function” corresponds to a function that should be addressed to all the devices of a monitored house code (functions like All Units Off) (see Note5). If that is

not the case, it uses `bool idbcom_standardX10Transmission(int header, int code)` to select a particular device.

5) It uses `bool idbcom_standardX10Transmission(int header, int code)` to send the function.

Please have a look at the schematic that summarizes this (see Figure 4).

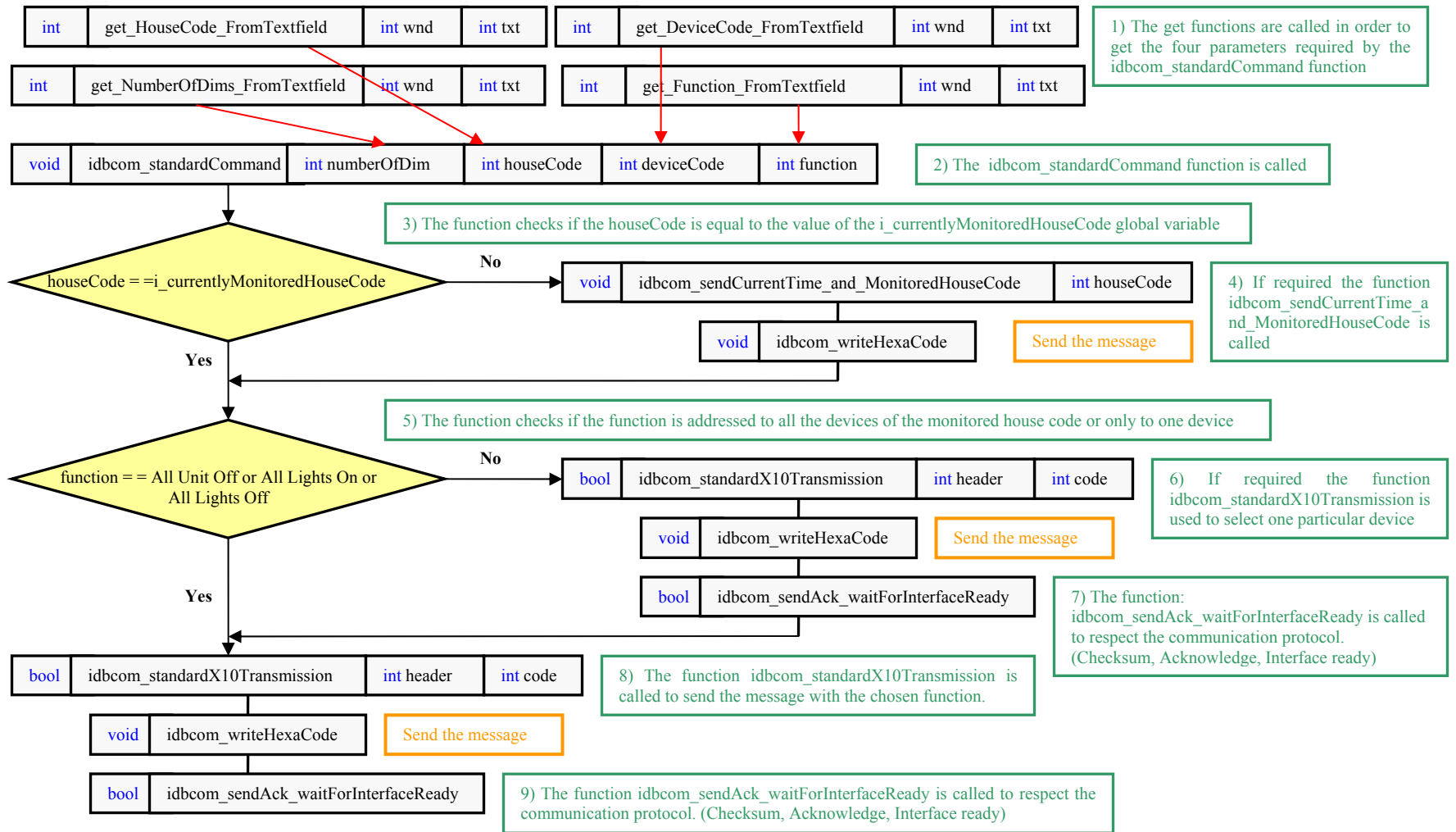
*Note4: It first checks if the house code corresponds to the monitored house code to ensure that the feedback will be correct. The device will be controlled properly no matter what the currently monitored house code is, but the interface will not know that the status of the device recently changed.*

*For example:*

*If the user sends the function On to the device A1 while the currently monitored house code is B and then asks for a status request on house code A, the interface will not remember that the user sent the function On to the device A1 before. This means that the status will be wrong.*

*That is why I decided to change the house code each time we use another device. In fact, it could be better to change the monitored house code only if the module corresponding to the address provides feedback. (Otherwise it is not useful).*

*Note5: If the function must be sent to all the devices of the monitored house code, it is nonsense to select one particular device before sending the function. (This does not mean that it does not work this way, but it is not useful).*



**Figure 4: The Standard X10 Transmission**

### 3.3.3. How to process the feedback

In order to know the status of the devices, the computer can send a status request to the interface (0x8b). The status request is immediately followed by:

Bit range	Description	Byte
111 to 96	Battery timer (set to 0xffff on reset)	(0-1)
95 to 88	Current time (seconds)	(2)
87 to 80	Current time (minutes ranging from 0 to 119)	(3)
79 to 72	Current time (hours/2, ranging from 0 to 11)	(4)
71 to 63	Current year day (MSB bit 63)	(5+)
62 to 56	Day mask (SMTWTFS)	(6-)
55 to 52	Monitored house code	(7 lo)
51 to 48	Firmware revision level 0 to 15	(7 hi)
47 to 32	currently addressed monitored devices	(8-9)
31 to 16	on / off status of the monitored devices	(10-11)
15 to 0	Dim status of the monitored devices	(12-13)

The current version of the Home Automation application is able to process all of these bytes except: Battery timer, Firmware revision level, currently addressed monitored devices and Dim status of the monitored devices. (I do not really know what the Battery timer and the Firmware revision level are useful for and I did not test my program with lamp modules a lot. I know that the currently addressed monitored device provides the address of the last used device and I guess it is useful for getting the Dim status which I think should only give the dim level of one device.)

The bytes 2 to 6 allow getting the date from the interface. As it is well explained in the official documentation, it is not a big deal to process this. (However if you still have some problems to do it, just have a look at what is done in the function: `void presentation_showDate(int i_hour, int i_min, int i_sec, int i_wday, int i_yday)` of `HomeAutomation.cpp`)

Getting the On / Off status is much more complicated.

- First you must be sure that the currently monitored house code is the same as that of the monitored house code of the device you would like to control. If that is not the case, the feedback will be wrong. If that is the case, the feedback should be correct and you can process it.
- Then, you should get the monitored house code (bit 55 to 52). The hexadecimal value corresponds to a particular house code. (See section 1.1 of the Interface Communication Protocol)
- Then, you should have a look at the 16 bits 31 to 16. The value of each bit corresponds to the status of a particular device (see section 1.1 of the Interface Communication Protocol). For example the device code 1 corresponds to the hexadecimal value 6, so to know the status of this device, you should have a look at the bit 22 (16+6).

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Device Code	10	2	8	16	12	4	6	14	9	1	7	15	11	3	5	13

For example:

If the value of the bit 22 is 1 this means that the device 1 is on. Otherwise it means that the device 1 is off or does not exist. To make the difference the computer must know exactly what the system is and all the devices that it can control. That is one of the reasons why the devices must be defined properly in the “Define Devices” window (See: User manual)

### 3.3.4. How to react when the CM11A sends a message

The interface can send different messages. As you can see in the `void MessagesFromIDB( char msg[], int count )` function, the program reacts to these messages as follows.

- If the message is a “poll from the interface (0x5a)”, the computer will immediately send the message PC ready (0xc3) to make it stop.

*Note6: I have never had any problem with this until now, but maybe it would be better to have a way to verify that it is effectively a poll from the interface by waiting this message to be sent twice in a row because this value 0x5a could also be something else (For example: a checksum).*

- If the message is a “time request (0xa5)”, the computer will immediately send the current time with the help of the `void idbcom_sendCurrentTime_and_MonitoredHouseCode(int i_monitoredHousecode)` function.

*Note7: It could also be better to verify that we receive this message twice in a row before reacting so.*

*Note8: The house code sent is “house code A”. (I could have chosen another house code it does not really matter).*

- If the last message sent by the PC is a “status request”, the answer from the interface should be a feedback detailing the status of the system. That is why in that case, the `void file_edit_CurrentStatusFile(char * msg, int count)` is called to store the status in the `currentStatus.txt` file.

*Note9: I decided to create a file to store the current status thinking that it will maybe be useful later to have a file with all of the status requests. This way, it will allow the user to do some particular actions like asking what the status of the system was 2 hours ago.*

- If the last message sent by the interface is “interface ready (0x55)”, then the global Boolean variable `b_interfaceRTR` is set to true.

*Note10: if the message is something else, for example a “checksum equal to 0x55”, the variable will be set to true even if it should not. This does not matter at all until the variable is set to true when it should. (The value of `b_interfaceRTR` is only tested when it must be true.)*

• Whatever the message sent by the interface is, until it is only a one byte message the function `bool idbcom_testChecksum(char * msg, int beginning = 0)` is called. It returns true if the message is equal to the checksum of the message sent by the computer and false otherwise. The global Boolean variable `b_checksum` is set to this value.

*Note11: sometimes the header of the message belongs to the checksum and sometimes not. To solve this problem, the parameter beginning of the idbcom\_testChecksum function should be set to 0 (default value) when the header must be added to the checksum, and 1 otherwise. Be careful because the documentation does not deal with this issue at all. As a result you will have to find out by yourself if the header must be added or not.*

## **3.4. The code**

### **3.4.1. Introduction**

The program has been written knowing that someone else would very probably have to look at it in order to understand how the communication with the CM11A via IDB works and also to modify it in order to improve the prototype. That's why you will find in the following all the naming convention created for the code of the Home Automation application. You will also find in the appendix the HomeAutomation.h file.

### **3.4.2. Naming conventions:**

In order to make the program more understandable and easier to code, I created the naming conventions described below. I highly recommend reading this before reading the code itself as it will help to understand it quickly.

#### **3.4.2.1. ComponentIDs.h**

The name of a component is always as follows:

`TYPEOFTHECOMPONENT_NAMEOFTHEWINDOW_NAMEOFTHECOMPONENT`

Except for windows for which the name is as follows:

`TYPEOFTHECOMPONENT_NAMEOFTHEWINDOW`

As you can see, only capital letters and underscores are used for the component's names.

`TYPEOFTHECOMPONENT` is:

WND: for the windows.

BTN: for the buttons.

SL: for the status lights.

LBL: for the labels.

TXT: for the text fields.

TXTA: for the text areas.



IMG: for the images.

If a component belongs to many windows NAMEOFTHEWINDOW is replaced by ALL. As few examples are better than a long explanation, I advise you to have a look at the examples below:

The window House Status is named:  
WND\_HOUSESTATUS

The button SHOWALLDEVICES of this window is named:  
BTN\_HOUSESTATUS\_SHOWALLDEVICES

The button HOUSEMAINWINDOW that belongs to many windows is named:  
BTN\_ALL\_HOUSEMAINWINDOW

Etc.

### **3.4.2.2. X10messages.h**

An X10 message is a byte that may take any value between 0 and 0xFF. For example, if you want to send a status request, you need to send 0x8b. If the interface is ready to receive it will send 0x55, and if the user needs to send the function OFF, he will have to send 0x3. As you can see it is not very convenient to use these kinds of bytes. That is why I created a naming convention for X10 message.

In most of the cases X10 messages are written as follows:  
X10\_TYPEOFMESSAGE\_MESSAGE

As you can see, I only use capital letters and underscores for the X10 message's names.

TYPEOFMESSAGE is:

HOUSE: if the message is a house code.

DEVICE: if the message is a device code.

FUNCTION: if the message is a function.

For some particular messages like “acknowledgement” or “PC ready” no type of message is specified.

Below are a few examples:

The message house code A is: X10\_HOUSE\_A

The message device code 9 is: X10\_DEVICE\_9

The message function OFF is: X10\_FUNCTION\_OFF

The message PC ready is: X10\_PC\_READY

Etc.

### **3.4.2.3. Methods**

As you can see below the methods used in Home Automation class have some particular prefix and format.

`get_particularInformation_FromWhat`

These are functions that will get particular information from something.

`file_edit_nameOfTheFile`

These are functions that will edit the file known as `nameOfTheFile`.

`file_clear_nameOfTheFile`

These are functions that will either delete all the information in the file known as `nameOfTheFile` or initialize it with default settings.

`idbcom_nameOfTheFunction`

These are the functions that are used to communicate with an IDB box.

`convert_currentTypeOfInformation_newTypeOfInformation`

These are the functions that will allow the user to convert a type of information into another type of information.

`edit_variableType_variableName`

These are the functions that will edit the value of some global variable.

### **3.4.2.4. Variables**

While there are no particular rules to give a name to a variable, frequently a variable's name contains a prefix that indicates its type. Here are few examples:

`wc_` means that the variable is of type wide character.

`c_` means that the variable is of type character.

`i_` means that the variable is of type integer.

`b_` means that the variable is of type Boolean.

`ifp_` means that the variable is of type ifstream pointer.

`ofp_` means that the variable is of type ofstream pointer.

### **3.4.3. Files**

#### **3.4.3.1. The currentStatus.txt file**

The currentStatus.txt file is a file where the answer of the CM11A interface to a status request is stored. The content of this file is then processed as described previously (see section 3.3.3).

Here is the format of the file:

The answer to a status request is a 14 bytes string.

The currentStatus.txt File simply contains 14 lines of one byte. Each time a new status request is sent the previous content of the file is deleted before writing the new bytes so that the content of the file is always of 14 lines of one byte.

#### **3.4.3.2. The names.txt file**

The names.txt file is a file where the names of the system's devices are stored. It is edited when the "Define Devices" window is used. This file is used to process the feedback properly (see section 3.3.3).

Here is the format of the file:

There are 256 lines (one line per possible module address).

Each line is as follows: RoomName ;DeviceName ;Module type '\n'

RoomName and DeviceName are always 10 characters strings chosen by the user.

Module type is a single character that can take the following values:

- X: if there is no module corresponding to this address.
- N: if there is a module that provides no feedback.
- Y: if there is an appliance module that provides feedback.
- Z: if there is a lamp module that provides feedback.

#### **3.4.3.3. Other files**

There are three other files char\_room.txt, char\_onOff.txt and char\_device.txt that are used to initialize the wchar\_t\_room, wchar\_t\_onOff and wchar\_t\_device global variables. It allows to modify these three variables and then being able reinitialize them with the content of the files.

*Note12: I am not so sure that it is a good idea to use these files. I just did it this way because it was my first idea.*

## 4. Encountered problems

The encountered problems are listed below:

- There are some mistakes in the official documentation of the X10 communication protocol. A document with some correction is available in the appendix. It is very possible that there are still some mistakes in it. So if there is any problem of communication with the interface it should be a good idea to use a sniffer to verify what is going on. Spying what happens when ActiveHome is in use could be a great help for this kind of problems.
- While using ActiveHome if the communication does not work it is very probably that the IDB services are started. To fix this problem go on “Control Panel” select “Administrative Tools” and then “Services”. There modify the option of IDB services by selecting “Manual”. You should be all set. (If you have problems to communicate while using the Project54 software, you should verify that the IDB services are started).
- If the CM11A interface is stuck the best is to use ActiveHome to reset its memory. The HomeAutomation software is currently not able to do so.
- For some reason the program is not able to communicate with the IDB boxes when it is running on some computers. It is not a hardware problem. The software seems to work well in some cases but, not at all in others. It is maybe a problem coming from the initialization, because when the configuration is changed the software works.

## 5. Project future improvements

There are many improvements that we can imagine to make a better system:

- Other devices can be added to make the system able to do more things. For this an IDB box per device must be added and a new application must be created. For this, it would be a good idea to look at what kind of devices Peter A. Heeman uses on his project. There are other good sources of information on the web.
- It could be very interesting to distribute the information in several computers. It is not very difficult to do so. There are documents dealing with the Proxy Application on the CATLab website [8] [9]. (In fact a client/server has already been created and it is possible to control X10 devices using the Proxy Application of Project54. The problem is that it has been done with an old version of the code so it should be modified).
- The wish of most of the users is to control the devices with voice. It would be a good idea to see what could be possible to do. For example we could imagine a system in which the user has a button on him or her and there is a microphone somewhere in the room. The user just has to press the button and talk to give some orders. (This would be really great. Let's see if it possible and if such a system is not too sensitive to noise).

- As the GUI has been created for a car it is not really adapted to a house system. It is not easy to make a user friendly interface with this GUI. The user has to type data in forms to control devices. For now the system does not allow to define macro. One of the reasons for this is that with the current GUI the user would have to fill in lengthy forms to create one macro. To imagine a new GUI, it would be a good idea to have a look at the ActiveHome interface which is object oriented and quite easy to use. It is also possible to download some freeware like HouseBot that allows using X10 devices [10] [11].

## **6. Conclusion**

The system is currently small but can really be improved in many ways.

As there are multiple ways to conduct the project, I cannot pretend that the way I choose is the best, but I always tried to concentrate my efforts on this objective: to obtain a prototype that works well and to make the code very easy to understand so that someone else could use everything that I have done.

## 7. References

1. Peter A. Heeman, “Multi-Agents Architecture for Home Automation”, personal communication.
2. The official version of the “*Interface Communication Protocol*”, manual.  
Available from: [ftp://ftp.x10.com/pub/manuals/cm11a\\_protocol.txt](ftp://ftp.x10.com/pub/manuals/cm11a_protocol.txt)
3. Charles W. Sullivan, corrected version of the “Interface Communication Protocol”, corrected manual.  
Available from: <http://tanj.com/heyu/protocol.txt>
4. Description of “*The X10 Powerline Carrier (PLC) Technology*”, customer support.  
Available from: <http://www.x10.com/support/technology1.htm>
5. Michael E Martin, “Progress report on the Project54 Common IDB Interface”, Technical report ECE.P54.2002.9, 6/28/2002  
Available from: <http://www.project54.unh.edu/Documents/TechnicalReports>
6. Michael E Martin, “Common IDB Interface Software Update,” Technical report ECE.P54.2002.1, 01/09/2002  
Available from: <http://www.project54.unh.edu/Documents/TechnicalReports>
7. Michael E Martin, “The Project54 Common Interface for the Intelligent Transportation System Data Bus Version4,” Technical report ECE.P54.2001.3, 10/17/2001  
Available from: <http://www.project54.unh.edu/Documents/TechnicalReports>
8. W. Thomas Miller, III, “Project54 client/server application messaging”, Technical report ECE.P54.2003.3, 03/24/03  
Available from: <http://www.project54.unh.edu/Documents/TechnicalReports>
9. W. Thomas Miller, III, “Remote Project54 application messaging via the Proxy Application”, Technical report ECE.P54.2003.4, 03/25/03  
Available from: <http://www.project54.unh.edu/Documents/TechnicalReports>
10. “HouseBot”. A freeware running on windows that allows using X10 and other devices.  
Available at: <http://www.download.com/3120-20-0.html?qt=HouseBot&tg=dl-20>
11. “Heyu”. A freeware running on Linux that allows using X10.  
Available from: <http://tanj.com/heyu/>