

A Communication Infrastructure for Home Automation

Renato J. NUNES

IST - Technical University of Lisbon / INESC-ID

Lisbon, Portugal

Email: Renato.Nunes@inesc-id.pt

ABSTRACT

This paper describes a home automation system - the DomoBus - that was developed as an academic project. One of the reasons that led to its development was the intent to address *super-automated* homes in an effective and economical way. We start by presenting that concept and then we describe the DomoBus architecture and its communication infrastructure.

We describe the main components of the DomoBus system - the Control Modules - which are based on very simple and inexpensive microcontrollers, with very small data memory. In spite of that, we managed to implement a robust protocol and we describe the options taken.

We describe also a serial line communication protocol used to interconnect the DomoBus network with high-level supervision modules or backend systems such as PCs. Finally, we briefly describe an IP-based communication service, with a socket interface, that seamlessly support the interaction between high-level applications and the DomoBus network.

Keywords: Home Automation, Domotics, Communication Network, System Architecture, Interoperability.

1. INTRODUCTION

Homes are places for living and resting, and also to relax and have some entertainment. At home people want comfort, security and not to bother with repetitive tasks, such as turning on lights at dusk, watering the garden, controlling the temperature in a room and turning off the lights in vacant places. People also want to manage effectively the electric energy, gas and water consumption to reduce expenditures and, at the same time, to be friendlier to the environment.

Home automation can offer these capabilities and also offer support to aging people, allowing them to stay in their homes, be more autonomous and increase their quality of life.

Because of its benefits, home automation is slowly becoming more popular. Several home automation technologies are available today. Some are proprietary

and some are standard or open. From the last ones we highlight X10 [1], LonWorks [2] and Konnex [3]. X10 is very popular, worldwide, due to its simplicity and low price, although it has many technical limitations. LonWorks is a powerful technology but it prevails in the building automation field and is more expensive. Konnex or KNX (formerly EIB) is having a good market acceptance, especially in Europe. It is technically sound but its price is not the cheapest and the design and configuration process of a system requires involvement of technical personnel.

We mention also CEBus [4], a very interesting technology developed specifically for home automation, although its market share is very small. For an analysis of market trends and reference to other technologies see [5]. Regarding power-line and wireless technologies, we refer to HomePlug [6], HomeRF [7] and Zigbee [8], which promise to offer new capabilities and ease installation in already built homes.

In spite of the many existing technologies, all approaches are expensive, particularly if we address homes with a very big number of control points (super-automated homes). This constitutes a challenge that motivated us to search for a different and more economical solution. Our approach is different as we use modules based on inexpensive microcontrollers that control 16 or more devices. In this way, we optimize the use of common resources such as the microcontroller itself, the power supply and communication hardware.

When compared with currently available products, where a module controls, typically, just one device, our approach may seem less modular as it forces some concentration in each module. However, if we consider a room with a large number of devices, it makes sense to use a technical cabinet in it, where modules are installed and from which cables are routed to the devices. This is perfectly adequate regarding actuators as they need power and, so, must be wired. It even offers the advantage of allowing the power electronics to be located inside the technical cabinet and, for example, use common power sockets or connect directly to motors, light bulbs, and other apparatus. This simplifies physical installation, is more economical and does not account for any significant

increase in cabling costs. Of course this applies only to new or renovated houses but, anyway, we are talking about super-automated houses and not common homes.

Regarding sensors, wireless approaches can make sense but we must not forget that they need batteries and that the communication elements are more expensive and less reliable than wired solutions (more susceptible to electromagnetic interference, for example). In the context of new or renovated homes it compensates to invest in a cabled solution. And it is important to note that multi-wire cables can be used to connect to switches, sensors and other devices that are located in the same vicinity, allowing to reduce the number of cable raceways in walls, ceilings or floors.

A situation in which a wireless approach has obvious advantages is in the domain of user interaction for monitoring or commanding purposes. A common PDA with wireless communications, or an equivalent portable system, can be the ideal means for a user to access the entire home and, using an adequate high-level graphical interface, get status information, issue direct commands or define time schedules and program repeating actions.

The current prototypes of the DomoBus home automation system use wired communications at the device level (sensors and actuators). At a higher level, standard IP communication can be used (wired and wireless, such as WIFI). However, we intent to pursue the study of wireless solutions and adequate gateways, in recognition of the flexibility offered by this type of approach. Developments such as reported in [9] are very interesting and should be studied regarding application in the home automation field.

In the next section we detail the notion of super-automated homes and exemplify the control points one could expect in a typical house's room. Then, we describe the DomoBus architecture and the communication protocol it uses, and we detail some of the options taken to implement a robust protocol using a microcontroller with very low data memory. Finally, we describe a serial line communication protocol and present a communication service that eases the development of DomoBus applications in computing platforms that support IP communication.

2. SUPER-AUTOMATED HOMES

Current home automation solutions involve, typically, a small number of control points. In a home with one living room, one kitchen, three bedrooms and two bathrooms (seven divisions in total), one may expect around 40 control points. These may correspond, for example, to 10 lights, 12 switches, 5 temperature sensors, 5 heater controls, 8 presence detectors (for lighting control and

intrusion detection), and one RF remote control. Although this may be considered a high number, we want to address much bigger solutions that offer automation levels an order of magnitude greater. Given our aim, we propose the concept of super-automated homes that apply to homes with a very big number of control points and a rich set of integrated services.

To illustrate the level of automation envisaged, we give examples of the control points that could be expected in a single room:

- 3 lights (ceiling/lamps, all with intensity control);
- 3 switches to control the lights (short pulse: on/off; long pulse: control intensity);
- 2 outputs for moving up and down the window blinds;
- 2 switches to control the window's blinds (up/down);
- 2 on/off sensors to monitor the open/close status of the window's blinds;
- 3 outputs to control electric power sockets;
- 1 presence sensor;
- 2 on/off sensors to monitor the open/close status of the room's door and window;
- 1 temperature sensor;
- 1 smoke sensor;
- 1 light intensity sensor;
- 3 switches to control the ambience sound (one for channel and two for volume);
- 1 infrared remote control receiver (to issue commands to the automation system);
- 1 infrared remote control emitter (to command a TV, HI-FI or other equipment);
- 4 switches to select usage scenarios;
- 8 low power outputs to signal specific status/operation conditions (using, for example, bicolor LEDs);
- 2 low power outputs to issue visual and sound feedback to commands or to signal alert or emergency situations.

This example accounts for 40 control points. Considering that a house has simpler divisions (e.g., bathrooms), but the kitchen and the living room are more complex, we can estimate that, in a house with seven divisions, we will have around 300 control points. This number may increase significantly for a detached house, which will be bigger and have, probably, a garden and outside devices. This clearly represents a different approach to home automation and the DomoBus offers a solution to it.

3. DOMOBUS ARCHITECTURE

The DomoBus architecture is centered on the use of Control Modules (CM). These are simple microcontroller based boards that interconnect with each other using a

communication network. For performance reasons, the DomoBus network may be divided into segments. In this way, the communications local to one segment do not affect the traffic in another segments. Figure 1 illustrates the DomoBus architecture and represents a network segment. The interconnection of different segments can be achieved using a backbone segment and Routing Modules (R).

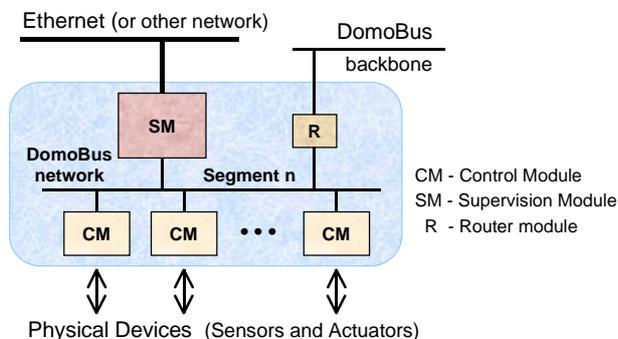


Figure 1. Architecture of the DomoBus system

The Control Modules connect directly to switches, temperature sensors, infrared receivers and other input devices. And they can control power electronics used, for example, to adjust the intensity of lights or to turn on/off small motors, pumps, lights, electric heaters or air conditioners. They can also execute other types of specialized actions, such as the generation of infrared signals used to command TVs, HI-FIs or other consumer electronics equipment.

Each Control Module (CM) is able to run different applications that perform autonomous tasks. For example, an application can read switches and control the intensity of lights, another one can read temperatures and test if predefined values are surpassed and another can simply actuate power relays upon reception of adequate commands. However, due to memory restrictions, the level of interaction with other CMs is limited to accepting and executing orders, answering requests and sending messages reporting the occurrence of some events. It is possible to configure a CM and specify the destination address and the type of message to be sent for each event it detects. But, due to the limited resources available, it is not possible to program a more complex behavior such as, for example, performing multiple actions, system-wide, in response to a given event.

To circumvent the limitations of CMs mentioned above, the DomoBus system includes another type of modules - the Supervision Modules (SM). These are responsible for system management and supervision. They receive information from the CMs, process it accordingly to programmed rules and required behavior, and issue the appropriate commands to the CMs. A system may have as many SMs as needed. In a small system we can have

just one SM, while in a big or complex system we may have, for example, one SM for each DomoBus network segment, as illustrated in figure 1. In this way, events generated by CMs connected to a given network segment can be received and processed directly by the associated SM. This approach allows a distributed supervision, offering benefits regarding response time and reliability (no single points of failure).

Note that a SM can control any CM in the system and that the SMs can interact with each other in order to share information or coordinate actions. To that end, the SMs can use a different network (for example, Ethernet or a wireless LAN) with more bandwidth and support for multimedia services. In this way, the interaction with other systems also becomes easier allowing interoperation and achievement of integrated solutions.

In the proposed architecture we consider the existence of a Home Server and Gateway (PC), that offers a powerful graphical interface with the user for system monitoring and programming. The Home Server and Gateway allows access to the Internet and may interact with other systems if required. More details can be found in [10].

The system supervision actions are defined and managed in the Home Server and Gateway, and downloaded to the Supervision Modules, which carry on the programmed tasks. In reference [11] a web-based application is proposed for the specification and programming of the required behavior for a home.

4. SOFTWARE ARCHITECTURE

We mentioned earlier that the Control Modules (CM) are very simple, as we want them to be inexpensive. A CM is constituted essentially by a microcontroller with a built-in UART, a transceiver EIA-485 to interface with the communication medium and the required electronics to connect with switches, sensors and power electronics.

We chose the microcontroller AT90S8515 from ATMEL, which has 8KB of FLASH program memory, 0.5KB of RAM and 0.5KB of EEPROM. This was one of the first microcontrollers having FLASH program memory, which eases the development phase. Nowadays there are many microcontrollers with FLASH program memory and with much more capacity (for example the ATMEGA128 has 128MB of program memory and 4KB of data RAM).

The limited memory of the microcontroller chosen was a challenge and a motivation to pursue a software approach as simple and modular as possible. Due to the lack of resources we decided to implement what can be called a very basic cooperative multitask kernel. In practice, this is little more than a cyclic loop that calls the existent tasks one at a time (round-robin policy).

The tasks are implemented as state machines. They are responsible for executing their actions and memorizing their state in global variables. A task must never block and should execute its actions as fast as possible to free the processor and allow other tasks to run. If an action is lengthy, it has to be decomposed into smaller parts that are executed one at a time. Although this approach requires some additional care from the programmer, the model is simple to understand and, after some practice, it becomes easy to develop tasks that follow this logic. As an advantage, little care is needed regarding concurrency as, at any time, there is only one task running. Access to common resources (when actions are lengthy) can be synchronized using global variables.

Our approach proved to be very effective, offering good performance in terms of execution time and memory usage. It also allows good control over time, which is essential for real-time applications.

It was clear from the beginning that communication was a central aspect of the Control Modules. We developed a task, named NET, that allows the exchange of messages with other modules and also supports communication within a module itself. Due to memory limitations, each task has just one mailbox that is used both for transmitting and receiving. Mailboxes are managed by the NET task that, itself, has no mailbox. When a new message arrives, the NET task analyses the destination address and determines the target mailbox. If it is full, receiving is aborted. If the mailbox is free, it is marked as BUSY_RECEIVING and reception continues with direct writing in it. At completion, the NET task signals there is a new message in the mailbox, for the corresponding task to process it. When a task wants to send a message, it starts by testing if its mailbox is free and marking it as BUSY. It then composes the message and, when done, signals the NET task that there is a new message to be sent. Upon transmission of the message, the NET task marks the mailbox as free.

5. THE DOMOBUS PROTOCOL

Frame Format

The protocol frames are constituted by a variable number of bytes. These are sent serially using half-duplex asynchronous communication. Sending and receiving is accomplished using the microcontroller's built-in UART that is connect to the EIA-485 transceiver. A bit-rate of 19200 bps was chosen.

Figure 2 describes the structure of a frame, which contains the following fields: dimension, destination and source addresses, control information, data and a CRC (Cyclic Redundancy Check). The dimension uses 7 bits allowing frames to have up to 127 bytes, although typical size is 9 bytes. The most significant bit of the first byte

must be zero. This bit can be used later to expand the frame's format. The destination and source addresses occupy 3 bytes (12 bits for each address) and were organized as shown in figure 2. It is important to note that each address reflects the DomoBus architecture, being composed by three fields that identify a network segment, a module within a segment and an application (task) in that module. This organization of an address simplifies drastically the implementation of the routing modules illustrated in figure 1. Each module is associated with a network segment making it trivial to decide if a packet should be forwarded from a network segment to the backbone and vice versa. An address with all bits equal to 1 is the broadcast address.

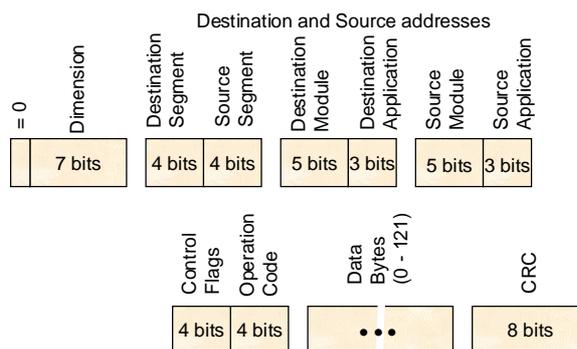


Figure 2. Structure of a DomoBus frame

Following the addresses in the frame is a control byte that contains an operation code (4 bits) and 3 control flags (there is a fourth bit that is unused). The control flags specify the frame's priority (normal/ high), type of frame (request/answer) and the frame's number (0/1). This last flag is provided to offer support for detection of receiving repeated messages. Currently this flag is not used as we keep all interactions idempotent. Commands explicit in an absolute manner the actions to execute and do not refer to a previous state.

The data field has a variable length (0 to 121 bytes) and contains the data sent to an application. This data will be interpreted accordingly to the value of the operation code. The frame ends with a CRC byte used to check the integrity of the message.

Typically a frame has 9 bytes since it uses 3 bytes in the data field: one to identify a device within the application (task), another to specify a property (for example, "light-intensity", "temperature", "on-off-status") and a third byte with the property's value. For more detail about application interaction refer to [11].

Access to Transmission Medium

The access to the transmission medium uses a variant of CSMA/CD. Every module listens to the medium and waits for a specific time of silence (inactivity). After that

period of time a new frame can start. A module that wants to transmit must wait at least that fixed period of silence and then, typically, will wait some more time. This second period of silence is variable in length and depends on two factors: the priority of the message (high priority means a shorter value) and a random value (to minimize the probability of collision).

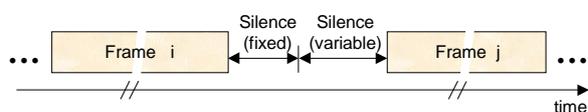


Figure 3. Access to the transmission medium

While transmitting, a module listens to the medium in order to detect a collision. If that happens, transmission is aborted and a new access to the medium starts.

To improve the communication reliability, every frame is acknowledged. If an acknowledgement is not received, the frame is automatically retransmitted. This process is repeated a certain number of times, after what the frame is discarded. To simplify the protocol and its implementation, the acknowledgement is constituted by just one byte that is sent shortly after the end of the frame. It violates the fixed silence period and, as such, it is not confused with the start of a new frame. It is assumed that only the valid recipient of a frame will send the acknowledge byte and only the sender will be waiting for it. The other modules just ignore this byte. This approach, besides the simplification it introduces, offers an increase in performance since it avoids the transmission of a complete acknowledge frame and avoids waiting the fixed silence period to access the transmission medium.

Reliable Communication

The retransmission mechanism that exists at the link layer is important to overcome frame losses that can occur due to collisions, because the destination mailbox is full or another reason. However, that mechanism does not assure that the messages are actually delivered as retransmission is canceled after some attempts. A way of guaranteeing that a message really reaches its destination (at least once) is to implement an acknowledgement and retransmission mechanism at the application level.

This type of behavior is usually accomplished using a buffer that stores the messages that were sent. When an acknowledgment is received, the corresponding message is removed from the buffer. Messages not acknowledged are retransmitted after some time. Unfortunately, we had not enough memory available and were forced to devise another approach.

Our solution uses a data structure to signal that an event has occurred and that a corresponding message must be

sent. In practice that data structure is simply a byte array where the index identifies the event (and corresponding message) and the byte's content specifies the following information (at the bit level):

- Send immediately (bit 7);
- Fast repeat counter (bits 6 and 5);
- Slow repeat counter (bits 4 to 0).

When an application detects some occurrence (for example, temperature above a predefined value) it activates the corresponding event, setting the "send immediately" bit and specifying values for the repeat counters. As a consequence, the adequate message is generated and sent immediately. If an acknowledgment is received the event is cleared. If that does not happen, the message is generated again after some time and re-sent (and the counter is decremented). As a protection measure for the network, we allow only a small amount of fast repeats, i.e., repeats with a small interval between them. After three fast repeats (at most), further sending will occur with a long interval in between. This greatly reduces the network traffic in the event of a loss of connectivity with a module or a network segment, while offering a "keep trying" mechanism. If the slow repeat counter equals 31 (all bits set) the message repeats forever if an acknowledgment is not received.

When the counters reach zero, the event becomes inactive and no more messages are sent.

6. SERIAL LINE COMMUNICATION

Serial-line asynchronous communication is a simple way of connecting the DomoBus network with common PCs and other systems. It can also be used to interface with supervision modules implemented by embedded PCs or single board computers that cannot connect directly to the DomoBus network. From our experience we defined the following requirements for this type of communication:

- Use only the transmit and receive lines;
- Be robust and allow connectivity to be broken and re-established at any time;
- Be independent of specific timings, as they can be difficult to enforce in certain platforms and operating systems.

Our implementation consists of a framing protocol that provides synchronization, error detection and flow control mechanisms. We defined six special bytes: END, CTS, ESC, ESC_END, ESC_CTS and ESC_ESC. Each frame starts and ends with END and this octet may never appear within the frame itself (it delimits the frame). Sequences of two or more END bytes are equivalent to just one END.

The byte CTS (Clear To Send) is used for flow control and signals that the sender can receive a new frame. We

use a "stop-and-wait" policy to conserve memory and allow the use of a Control Module (with an additional UART) to implement the DomoBus bridge to serial line.

In case there is no traffic, a CTS byte is sent periodically. This assures the restart of communication in case a CTS is lost or in case the interlocutor is disconnected and reconnected later on.

We use escape sequences to represent END and CTS if they appear in the data to transmit. Those sequences are, respectively, ESC + ESC_END and ESC + ESC_CTS. Because of this, the byte ESC must itself be represented by ESC + ESC_ESC. The byte stream is analyzed byte by byte, in search for END, CTS or ESC. If ESC is found, the next byte is analyzed (which must be ESC_END, ESC_CTS or ESC_ESC) and the sequence is converted to the corresponding byte.

To detect communication errors a checksum byte is appended to each frame's data. We use this simple scheme because the occurrence of errors is uncommon. The data sent consists of DomoBus frames without the CRC byte (see figure 2).

7. IP-BASED COMMUNICATION

Currently it is easy to find single board computers with Ethernet controllers and a TCP/IP protocol stack. These boards allow implementation of complex applications and supervision services, and ease interaction with other systems, locally or remotely using the Internet.

To ease development of DomoBus applications in those boards (or common PCs) and offer support to IP communication, we developed a communication service named ComServ. This service uses a socket interface and allows a seamless interaction between DomoBus applications, independently of their location, using UDP datagrams or the serial line protocol described earlier.

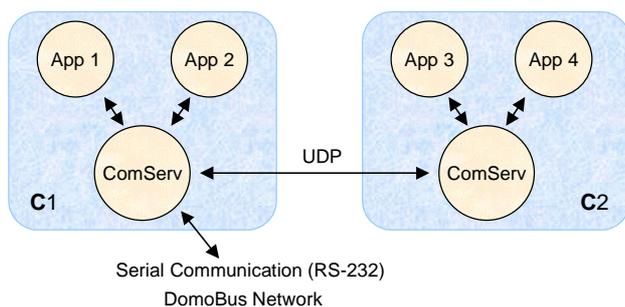


Figure 4. Communication support offered by ComServ

We chose UDP because it offers an adequate service, with low latency, for the small messages involved. However, UDP does not guarantee message delivery. We

considered this behavior acceptable within the same computer as that is improbable and the DomoBus applications offer, themselves, an acknowledgement and retransmission mechanism. However, when different computers are involved, we decided to implement that mechanism between ComServs, for optimization reasons.

ComServs are responsible for translating DomoBus addresses into IP addresses and port numbers. To that end, each application registers in its ComServ. ComServs interact with each other and propagate the relevant addressing information.

8. CONCLUSIONS

In this paper we presented the notion of super-automated homes and described the architecture of the DomoBus home automation system. We focused mainly on the communication infrastructure and the design options taken to implement a robust protocol in devices with very small data memories. We described also a serial line communication protocol and briefly presented an UDP communication service that eases the development of DomoBus applications in platforms that offer TCP/IP communication.

9. REFERENCES

- [1] X10, <http://www.x10.com>
- [2] LonWorks, <http://www.echelon.com>
- [3] KNX, Konnex Association, <http://www.konnex.org>
- [4] CEBus - Consumer Electronics Bus, <http://www.cebus.org>
- [5] Maria Panou, Evangelos Bekiaris, "Mobile/Wireless Systems for Domotic Applications: The Wireless Home", CCCT 2003 - International Conference on Computer, Communication and Control Technologies, Orlando, USA, August 2003.
- [6] HomePlug, Powerline Alliance, <http://www.homeplug.org>
- [7] HomeRF, <http://www.homerf.org>
- [8] ZigBee wireless technology, <http://www.zigbee.org>
- [9] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, John Anderson, "Wireless Sensor Networks for Habitat Monitoring", ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, September 2002.
- [10] Renato Nunes, "DomoBus - A New Approach to Home Automation", 8CLEEE - 8th International Congress on Electrical Engineering, Portugal, July 2003.
- [11] Renato Nunes, "A Web-Based Approach to the Specification and Programming of Home Automation Systems", MELECON 2004 - The 12th IEEE Mediterranean Electrotechnical Conference, Dubrovnik, Croatia, May 2004.