DEVELOPMENT OF A DOMOTIC SYSTEM PROTOTYPE

Francisco Daniel Matos Pinto

Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, Portugal
E-mail: francisco.pinto@ist.utl.pt

ABSTRACT
This work was developed in the context of the domotic system DomoBus which intends to be a flexible, integrable and low cost solution for home automation, and to provide mechanisms that support interoperation with other technologies. The main objective of this dissertation is to enhance the DomoBus system, developing a new version of the Control Modules that are implemented by embedded systems with very low memory and processing resources. The goal of this work is to design and implement a new software architecture that provides multitask and real-time capabilities and reduce the complexity of applications, simplifying their development and enhancing their reliability. Additionally it was also developed a new communication software module that ensures robust communications that support retransmissions, optimizes bandwidth usage and avoids congestions. The fulfillment of these features was very challenging due to the extremely low memory resources available on the microcontroller used. The developed solution proved to be flexible and robust, and easily portable to other embedded platforms due to its generality and independence of specific hardware features offered by the microcontroller, namely at the communication level.

Index Terms— Domotics, DomoBus, Embedded systems, domotic applications, communication protocols, real-time systems

1. INTRODUCTION
Home automation - domotics - is a subject on which the population in general is interested and is being emphasized over the last years. The existence of systems with the possibility of performing routine chores, help disabled and elder people and, at the same time, of making resources saving, such as electrical energy, attracts a typical consumer. A domotic system aggregates a group of technologies and mechanisms that work together and so, should be easily integrable to better serve the user. The solutions that currently exist have some restrictions namely the incapacity to interact with other technologies, difficulties to upgrade and to adapt to the new needs of users and are expensive [1]. The described scenario served as a starting point for developing a domotic system called DomoBus, in which this work is based on, aiming to be integrable with other systems, be capable of expansion and with very low cost.

This work attempts to demonstrate that using microcontrollers with very low resources, linked in a sensors and actuators network, it is possible to create a domotic system with a very low cost, high reliability and robustness. It is also possible to connect to other networks to enable monitoring and local or remote commands supporting interoperation mechanisms. The availability of very low cost devices interconnected in a network, makes the implementation of large scale systems possible and enables super-automated solutions with a rich set of functionalities and offering support to the creation of smart environments.

The intentional usage of a microprocessor with very low resources (AT90S8515) – especially in terms of memory capacity – challenges the design of a new software platform that simplifies the applications and their complexity while enables code reduction and memory saving. In line with this objective it is intended to implement a new reliable and robust communication software module with retransmission support and collision control that optimizes the usage of available bandwidth.

The developments should follow the DomoBus model, intending to offer to the applications a set of services that simplifies the read and modification of the properties associated to each domotic device. Given the generality of the model, many operations should be automatically performed (for example reading the value of a property or notifying its change), without the application having to worry about these details. The solution should also be easily portable to other platforms and microcontrollers so, only basic resources, and available in most microcontrollers (a time and a UART) will be used, avoiding the usage of specific hardware such as communication controllers.
2. RELATED WORK

2.1. Domotic Protocols

There are several communication protocols – specific for domotics or not – that can be used on a home automation solution. The text below describes the main topics of some protocols.

**KNX** – this standard (EN 50090, ISO/EIC 14543) results from the convergence of three previous protocols: European Installation Bus (EIB), European Home System Protocol (EHS) and Batibus. Due to the fact that it has five modes of transmission (electrical network, twisted pair, radiofrequency, infrared and Ethernet), and there is the possibility of including gateways to systems, it is a flexible solution. The communication throughput depends on the media used: 1200 bps over electrical wires, 16 kbps using radiofrequency and 9600 for twisted pair usage. This system is very reliable but has the disadvantage that its implementation normally has a high cost [1].

**X10** – this protocol uses the existing electrical installation to communicate with devices permitting its use in existing homes, without the need for additional cables and making it one of the systems with more acceptance and affordability. Each device has an identifier and reacts only to commands that are addressed to it. The “1” bit corresponds to a burst of 120 kHz and bit “0” corresponds to the absence of this signal. In order to improve the signal to noise ratio and to decrease the probability of electrical noise be mistaken for a valid signal, it was decided to send the bit always followed by its complement. Between commands, there should always be a pause of three cycles meaning that it generates a latency which becomes noticeable and critical for interaction with users [2].

**LonWorks** – created by Echelon Corporation this protocol is specifically directed to the performance and reliability of control applications. It supports various physical means of communication such as twisted pair, optical fiber and radiofrequency. This technology has several characteristics to improve the communication reliability such as detection of duplicate messages, collision prevention, automatic retransmission and error correction. In addition to these features, LonWorks accesses the physical media according to Predictive p-Persistent CSMA dividing the random period of time to access the media in equal parts. Therefore, when a device has to send a message, makes it with a probability $p$ adjusted according to the level of network traffic – calculated by the number of confirmations received after sending a message. It is also provided a mechanism for improving critical packages transmission, assigning the media access to priority devices, Figure 1.

LonWorks has the following communication rates: 78.13 kbps for twisted pair, 5.4 kbps for 60 Hz and 3.6 kbps for electrical networks. It has, however, the great disadvantage of being required to use a specific processor, the Neuron® [3].

**CAN BUS** – originally developed in the 80s to provide communication between the various electronic devices that constitute an automobile, without using a PC. CAN BUS has currently very diverse applications such as industrial automation, medical equipment and home automation. Each node can send or receive messages (not simultaneously) and is provided a method of arbitrating priorities: if the bus is free, any node can transmit but, if two or more devices tries to send the message at the same time, the message containing the dominant identifier (with more dominant bits – “0”), will have access to the bus and the other devices (with recessive bit value - “1”) enter the receiving mode. This scenario is depicted in Figure 2.

In the figure above, the eighth bit of node number two has a recessive bit and changes to reception mode because the other nodes have a dominant bit. The tenth bit of node number one has a recessive value so, node number three gains access to the bus because it has a dominant bit. Each node requires a processor, where sensors and actuators are connected and where the messages are processed, a CAN controller to deal with bit reception and transmission and a transceiver (integrated or not in the controller) that makes the adaptation between the signal levels of the bus and the CAN controller. For distances below 40 m, the maximum communication speed is 1 Mbps and reduces to 40 kbps for a maximum distance of 1000 m. CAN Bus has the disadvantage, as mentioned, of having to recourse to a proper controller so that the communication is possibly, raising the cost of purchasing the hardware [4].
PROFIBUS – this protocol belongs to the Fieldbus family – computer networks for distributed real-time control and used mainly in industrial applications – and featuring three main variants:

- PROFIBUS-DP (Descentralised Periphery) - high speed solution, geared primarily for critical-time control systems with time-critical and resorting to the physical media RS-485 or optical fiber. This variant is suitable for the floor of a factory, where exists a large volume of information;
- PROFIBUS FMS (Fieldbus Message Specification) - evolution of the previous variant and used for networks of programmable controllers - PLCs - and PCs;
- PROFIBUS-PA (Process Automation): latest version of PROFIBUS and its main feature is the ability to transfer data over the power wiring.

Resorting to RS-485, the communication rate is situated between 9.6 kbps (up to 1200 m) to 12 Mbps (up to 100 m) and allows 32 devices per segment up to a total of 127 devices [5].

2.2. Operating Systems

Among the various operating systems (of free usage) available for embedded systems, the following are focused:

- TinyOS – this operating system was developed for wireless sensor networks applications and is geared for limited resources microcontrollers. TinyOS language is based on C (nesC) and users can replace the task scheduler in order to experiment new methods for scheduling [6].
- eCOS – develop to take into account the needs of systems with real-time execution and reduce RAM, this system is implemented in C/C++ and has two modes of preemptive scheduling. It also has a hardware abstraction layer, memory management methods, TCP/IP protocol stack and support for PCMCIA, PCI and USB. This operating system allows users to choose the features they really need, making the system very close to the intended applicability [7].
- Contiki – also geared for wireless sensor networks and microprocessors with reduced memory capacity, Contiki is highly portable, supports multitasking and preemptive scheduling. It has a graphical system for monitoring (using internet) and supports TCP/IP. Programs for this operating system are written in C [8].
- FreeRTOS – this operating system was developed to be simple and to save memory. It supports 27 processor architectures and includes a method of preemptive scheduling and the ability to create processes and semaphores to improve the communication between tasks. This system is programmed in C language [9].

3. DOMOBUS SYSTEM

3.1. Overview

DomoBus is an academic project intended to provide a reliable and integrable solution at a very low cost and offering support to the creation of smart environments. Home automation devices are, in this solution, abstract entities described by a set of properties and their value can be read or modified by a set of operations – described below in Table 1– and whose messages are generated from events.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Reads the value of a property</td>
</tr>
<tr>
<td>SET</td>
<td>Changes the value of a property</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>Notifies the change of the value of a property</td>
</tr>
<tr>
<td>FORWARD</td>
<td>Forwards messages to other devices (usually supervisory)</td>
</tr>
</tbody>
</table>

This approach allows a very efficient usage of memory and, as opposed to other systems that have queues to store the pending messages to be processed, is used the EEPROM from the microcontroller to store the messages structure, which will then be generated. It is also possible to define a DomoBus system through XML [10].

DomoBus applications exchange messages among them (locally or remotely) in an efficient and robust way and there is also a communication module with support for retransmissions and network congestion control to avoid and minimize collisions between nodes and adjust the communication rate to use efficiently the available bandwidth.

3.2. System Components

The DomoBus system consists of Control Modules, Supervision Modules and devices for interconnection: bridge or gateway allowing the creation of an easy expandable and flexible architecture [11].

- Control Modules: programmed in C language, these modules are those who actually interact with the physical environment and are responsible for reading the sensors and commanding the actuators. For communications, it is used an EIA-485 transceiver and it is possible to communicate directly with other modules connected to the same bus or, through the bridges, with the rest of the system. Each module can have up to eight different applications and as such, perform different tasks. This feature distinguishes DomoBus from other technologies, since in these a module is connected
to only one device, and reduces the cost of a domotic system. These modules should be as autonomous as possible to improve the system robustness and reduce network traffic;

- Supervision Modules: these modules are in charge for the supervision and the management of the system. Upon the reception of data from the Control Modules, it is processed according to the expected behavior and the appropriate command is returned. In a low complexity system it is possible to use only one module but, in more complex systems, each segment can have a Supervision Module. Thus, the events generated by a Control Module connected to a network segment can be received and processed directly by the Supervision Module associated with it, enabling a distributed monitoring and increasing the network reliability. These modules can also contact with any Control Module in the system and can be connected to a different network with higher bandwidth (e.g. Ethernet). The user interface is improved by the existence of a PC (it can display a graphical interface) but, as the actions of supervision and management are embedded in the modules, it can be eliminated without the loss of the system reliability;

- Interconnection modules: perform an extremely important role in the system: the bridge modules interconnect different segments and must have the ability to recognize and route messages. The gateway allows all modules to send and receive messages via PC.

3.2. Network and media access

To access the media, and since it is shared with several devices, the CSMA/CD protocol is used. Thus, the media is continuously observed to detect the silence period required for the start of transmission. This period includes a fixed interval of time related to the physical media and transmission characteristics and a random period of time to avoid simultaneous transmissions. When a collision is detected, the devices stop the transmission and wait for a new period of silence.

As seen on Figure 3, DomoBus frames consist of a header (5 bytes), 0 up to 121 data bytes and one byte for CRC to verify the integrity of the frame.

![DomoBus Frame](image)

Figure 3 - DomoBus Frame

The header of DomoBus frames is formed by the following fields:

- 1st byte: gives the indication of the frame type (if it is a DomoBus frame or not), the following seven bits indicate the frame size;
- 2nd byte: the first four bits indicates the segment of destination and the other bits contain the segment that originated the frame;
- 3rd byte: the first four bits indicates the node of destination and the other bits contain the application to which the frame is for;
- 4th byte: the first five bits indicate the sending node and the remaining bits, the application that originated the frame;
- 5th byte: the first bit is not used, the three following bits are used to control the frame (it contains the priority and states if the frame is a retransmission or an answer), the last four bits indicate the operation to be performed;

3.3. Control Modules software

When this work was started, the DomoBus system already had several working versions and implemented applications. These applications had, at least, two state machines (one for receive, send and process messages transmitted between the applications and other to implement the desired functionality for the application) leading to code repetition and memory waste. Due to the microcontroller – AT90S8515 – low-resources, this situation could compromise or invalidate the system improvement. The existent communications module, despite having implemented a method to minimize collision among nodes, had not any method to perceive the status of the network and to act in accordance, to ensure the reliability of the communication. It is considered therefore that there is room for improvement, which is the purpose of this dissertation. The next chapter presents the proposals for this system improvement.
4. SOFTWARE ARCHITECTURE OF THE CONTROL MODULES

4.1. Real-time kernel

Given the limited resources of the microcontroller, it was decided not to use any of the real-time operating systems already existent because it would not allow great flexibility for application development. The following aspects were considered important to the application domain [12]:

- Preemption is not required;
- Co-operative environment;
- Consideration of all tasks of equal importance, not justifying the existence of priorities;
- Interrupts usage only when indispensable;
- Flexibility to manage time, allowing a simultaneous account of multiple values, from the millisecond range up to seconds or even more.

Considering all the aspects mentioned above, it was decided to use to perform the tasks in a cyclic manner and using a Round-Robin policy – every task gets a time slice of CPU utilization and when this time expires, the process is sent to the last position of the execution queue. It is therefore important to ensure, when transitioning to other task, the safeguarding of the execution context so that when the task is resumed it continues from where it was at the time the task was commuted.

4.2. Communication

The “NET” task implements the Domobus communication network through a state machine (it has the function of the logical and network layers) and at a higher level, the exchange of information between applications is ensured through the use of a buffer, mailbox, which is used both to receive and for transmit Domobus frames. The usage of only one mailbox per application means that the communication must be made in half-duplex mode because it is not possible to send and receive messages simultaneously. Domobus frames are sent via an asynchronous serial communication of 8 bits at 19200 bps. It is usually used the microcontroller’s internal UART but, an external UART can be used. It also uses an EIA-485 transceiver (which has the role of physical network layer) to access the media. The activation of the Interruption 0 (INT0) denounces the start of a communication because the start bit corresponds to a transition from 1 to 0. Similarly, a transition from 0 to 1 indicates the end of the data bits transmission, Figure 4.

The initial state of “NET” task verifies if there is any message to be received and if so, starts receiving the first byte. If there is nothing to receive, it searches for possible messages to be sent. Throughout the communication process, it is necessary to use timers to synchronize the actions of sending or receiving frames and, in case of errors, this initial state is resumed.

When receiving a frame, after the reception of the first three bytes the destination of the frame is verified. If it is local and the destination application’s mailbox is free, it is then reserved and the frame is stored. If the mailbox is busy, a jamming byte is sent to the network to generate a collision and to avoid the unnecessary sending of the frame. Upon the complete reception of the frame, and if it does not contain any error (verified by the CRC calculation), a byte is sent to confirm the success (ACK). If an error is detected a byte reporting the failure is sent (NACK).

When sending a frame, each sent byte is verified to detect any collision occurred and the need to retransmit it. When all the bytes have been sent, the machine evolves to the state of waiting for confirmation. If the confirmation (ACK) is received, the initial state is resumed otherwise the message retransmission is performed (with a limited number of tries).

In this protocol there are two different confirmations of the transmission success: in the application level the answer consists on the repetition of the frame but with the activation of the answer bit and in the lowest level, with the ACK as described above.

The increase in the number of messages can cause network congestion, leading to delays. A home automation system must be reliable and have a reduced operating time so this is a situation that must be avoided. When Domobus detects a collision, it increases the random time interval to access the media and so the number of collision should decrease. Analogously, if network amelioration is detected, the random period of time must decrease, improving the throughput, Figure 5.
It is also available a method to perceive the network status, by counting the messages sent or received in a certain sample time. The number of messages is then used to determine the network traffic level and if necessary, a pause time is attributed, originating an intentional delay on the sending process. These approaches together with retransmission capability lead to a reliable communication avoiding and minimizing collisions and optimizing the available bandwidth usage.

4.3. DomoBus applications support

With the current implementation of DomoBus, resulting from the contribution of this work, the applications contain code related solely to their functionality and to process non-standard messages. Standard messages are now treated through a task common to all applications (MSG), which in turn accesses the functions of a library that provides the structure for managing properties (PROP). These modules together with “NET” support the DomoBus applications, Figure 6.

A typical DomoBus application is intended to perform operations related to the reading of sensors and the commanding of actuators (for example, a set of applications that by reading a sensor that measures the light intensity of temperature, order a window to close or open). A Control Module can execute up to 8 applications, a maximum of 32 devices can be connected and an application must contain properties (of the type byte, word or array) that describe the devices connected. In order to make a system as autonomous as possible, the applications should not have a too specific operation so it can be used for various purposes. Hence, an application that measures a room temperature can be used (together with other applications) to trigger an alarm when a certain value is reached and to order a window to close or open.

The events, generated by applications, enable the creation of messages and therefore a value from a property can be read or set. An event is activated with an associated value to indicate if it has been already confirmed and the number of its messages retransmissions in case of error.

The following two sub-sections describe how properties and messages are handled.

4.3.1. Properties management module - PROP

This library contains the functions and data structures needed to manipulate properties and events. Despite the existence of three types of properties (byte, word and array) they are mapped into a single array, Figure 7. To one property it is assigned a single event and message.
In addition the existence of functions to get/set values and set events, there are also functions to clear the events, set a property value as valid or invalid, search an event by its sequence code and a function to be used by applications when a new value is read, to store it if valid.

4.3.2. Messages treatment module - MSG

All the operations associated with sending and receiving standard messages are handled by MSG. The responses to NOTIFY and SET commands are automatically processed and the value of property affected. This module also generates confirmations/answers to messages with GET and SET operations.

Applications triggers events that in turn will initialize the sending of messages and the procedure is done by MSG (mailboxes filling, transmission and possible retransmissions). This module was implemented through a state machine, described below.

The initial state checks for incoming messages for being handled. If existing, it searches for an application with messages to receive and the machine evolves to the first state of reception, where a standard message is processed (according to the operation) and a non-standard message is sent to the application. A message reply is performed by a proper state of the machine.

If there isn’t any message to be received, the existence of an event to be sent or retransmitted is verified and if so, the message is composed and sent to the destination application mailbox.

In order to save memory space, the messages structure is stored in the microcontroller’s EEPROM and is accessed by the event index (the missing fields of the message are then filled and the message is ready to be sent), Figure 9.

The following table contains the procedure to be performed for the different operations (GET, SET and NOTIFY).

<table>
<thead>
<tr>
<th>Operation</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>When a node sends this command, it receives the property value as answer and passes it to the application. If the node receives a GET command, reads the property value and sends it.</td>
</tr>
<tr>
<td>SET</td>
<td>When sending this command, the node receives a confirmation answer and clears the event. If the node receives this command, sets the property value and sends the answer.</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>The node clears the event after sending this command and sends the command to the application when receiving a NOTIFY command</td>
</tr>
</tbody>
</table>

4.4. DomBus implemented applications

The applications described below follow the new methodology, not dealing with standard messages. Upon reading the sensor and obtaining a new value, applications must activate the correspondent event – to trigger command messages – if the value is valid.

4.4.1. Node management application – SYS

All nodes must contain this application because it has various management roles like controlling LEDs, activating the watchdog, verifying errors occurred on other applications, receiving broadcast messages and obtaining an address if it has not been assigned.

4.4.2. Application to command relays – Relay

This application enables to control relays, activating or deactivating the devices connected to the system. Upon the reception of the message containing the command, the operation is performed and a NOTIFY message is generated in order to notify the system modification.

4.4.3. Application to read switches – Switch

This application reads the position of pushbuttons and switches, to command a group of lamps in accordance. In order to improve the efficiency, only one switch is treated at once. Switches have two states, generating a command message to modify the lamp state, when changed. If a pushbutton, normally inactive, is short pressed, the lamp should toggle. If the pushbutton if long pressed, the lamp intensity should vary. To do so, when pressed, the time is counted: if lower than a predefined time constant, the lamp changes it state; if greater, a light intensity adjustment is made, generating command messages periodically.
5. SIMULATIONS AND RESULTS

5.1. Sequence byte usage impact

This work changed DomoBus frame format to indicate beyond the device, property and value, a sequence number. The reasons that led to this implementation are related to the proper pairing of the response to the message and this section aims to analyze the impact that this approach had in the communication throughput – number of messages per second. All property types were simulated (byte, word and array with 4 elements).

Sequence byte addition decreases the network rate by 4.3% to 6.0%. This approach allows however to increase communication robustness because the singleness of sequence number eliminates situations where was the possibility – due to transmission failure – to confuse message answers, clearing an event with an old message.

5.2. Short answers usage impact

DomoBus frames contains a bit indicating if the message is an answer of not. For messages concerning a SET or NOTIFY operation, the answer is sent exactly as the original message, except for the answer bit that should be active. If the message contains a GET operation, the answer message will obviously have to contain the value of the property to which it relates, in addition to the activation of the answer bit.

The following simulation intends to verify the impact of introducing frames containing only the sequence number to answer frames with sequence number, device, property and value. The network rate is compared with frames and answer without sequence number.

The usage of answers containing only the sequence number increases the communication rate up to 15%.

5.3. Super-priority answer messages usage impact

DomoBus allows the usage of frames with different priority (normal, with priority and super-priority) – differing in the guard time to access the media. The present simulation aims to verify the impact of using a super-priority frame (with only the sequence number) as an answer to a normal priority frame. The results compare the network rate of this approach with the usage of a frame and its answer without sequence number and with normal priority.

The usage of super-priority frames enhances significantly the network rate. When combined with answers with only the sequence number, network rate improvement is greater.

5.4. Benefits from adjusting media access random time

The communication traffic is supposed to be variable during execution time. If the number of transmitting nodes increases, the average number of collisions will increase to and it is necessary to take measures to improve communication efficiency and to optimize the available
bandwidth utilization. These measures have to be readjusted later, if the transmission conditions ameliorate.

A node, before starting its transmission, has to wait a fixed and a random time to access the media to avoid colliding with another node that is also trying to transmit. In the current implementation, the random access time corresponds to a maximum of interval of 3, 7 or 15 units of 200 µs depending on the traffic level – low, average or high. This time is increased when a collision is detected and is reduce when it was noticed an improvement in the fluidity of the network.

Although it is not possible to accurately measure the improvement that this implementation has introduced just resorting to simulations, the number of collisions is expected to decrease and the available bandwidth usage to be optimized because the number of available time slots to transmit will be automatically adjusted in accordance with the network conditions.

5.5. Analysis of results

The introduction of sequence number was done in order to support a more reliable communication, eliminating duplicated messages or to prevent errors pairing delayed messages. The following example can be given: an event generated and sent a message (M1). The answer (A1) was also generated but, it couldn’t be sent. If a new event generates and sends a new message (M2), in the case of receiving the answer A1, it can be wrongly paired as a M2 answer. Sequence number usage is intended to cease these situations due to its oneness, and also possibilities the future implementation of an encryption algorithm, being used as a cipher.

As a disadvantage, the utilization of sequence number decreases the network rate from 4.3% to 6.0% but considering that all messages have to be answered, the introduction of answer messages sending only the sequence number increases the rate from 2.8% to 14.5% (message and its answer). Additionally, the usage of super-priority answers increases the number of possible messages per second even more – from 8.8% to 21.2%.

6. CONCLUSIONS

This dissertation analyzed in detail the DomoBus system which intends to be a reliable, robust and low cost solution for home automation. The main contribution of this work focused on designing and developing a new version of DomoBus’s Control Modules, which are implemented by embedded systems with very limited resources (memory and processing power).

The thesis followed the DomoBus model in which the different devices are described through a set of properties whose value is changed by standard messages triggered by the activation of events by applications. A new software architecture was developed for the Control Modules to simplify the development of domotic applications and to offer multitask capabilities and reliable communications support. This new approach has proved to be more robust and compact in terms of code than the previous version because it centralized the handling of messages, which before had to be performed application by application. In addition, the MSG software module also performs the construction of messages from their structure present in EEPROM and deals with the retransmissions and messages answers. Hence, the typical approach (and unacceptable due to the low memory capacity) of saving in memory pending answers is avoided. Now, it is also possible to detect and reject invalid values resulting from sensor reading errors or breakdown and all the property types supported by DomoBus were contemplated (byte, word and array).

Together with these new developments it was also implemented a new communication module to increase the network reliability and introducing retransmissions, minimizing collisions and controlling communication throughput in order to optimize the available bandwidth usage. It is now possibly to perceive the network status acting in accordance. The proposed objectives were fulfilled and mechanisms to enhance messages answering were also implemented – usage of sequence number, short answers, super-priority messages, adjustable media access time (to avoid collisions) and network throughput control to avoid congestion.

These contributions were simulated by calculation formulas, graphs e tables and by a simulator due to the difficulty to quantify these improvements in practice.

The sequence number usage lowers the network rate but, in contrast, increases the messages/answer pairing in case of errors occurrence. Additionally the introduction of short answer messages – containing only the sequence number – and super-priority answers benefits the network.

The developed solution is portable and provides multitasking features with real-time and reliable communication, using a minimal amount of data memory (512 bytes) and reduced code memory (8 Kbytes). The software implemented requires only basic hardware resources of the microcontroller, namely a timer and a UART for communication and therefore can be adapted to most microcontrollers available, including those with a low cost.

6.2. Future work

After this work, there is still room to improve DomoBus system. The security, in particular, can be improved with the introduction of encryption on frame’s data field. The introduction of the field ‘S’ on the messages (sequence number), opens the possibility to be used as the seed of a cipher algorithm. Thus, it would be assured that external elements to the system would not have access to the state of the devices nor the ability to change them.
The modification of the frame’s format made difficult to test in practice and quantify this thesis benefits, because it meant that the existing gateway – to connect the modules to a PC – could not be used. Without this infrastructure, to easily send messages to the modules and monitor the messages generated by them, was not possible to quantify such aspects as the number of collisions, the number of retransmissions, the number of messages confirmed, the effective throughput achieved and the impact of adjustments of some parameters such as the random value increase due to the occurrence of a collision. The necessary adaptation of the gateway and PC software is very complex and would imply significant prolongation of the present work. Additionally there was still a restriction due to the low number of available Control Modules and the impossibility to build new modules within a short time, without whom it is impossible to effectively test the mechanisms for congestion control. In future it will be desirable to test exhaustively the system to evaluate the mechanisms and allow its adjustment. Thus, results would be more accurate and reliable than with simulations.