Controller Area Network domotic prototype using GPRS and Ethernet interfaces for virtual monitoring applications

Francisco Cortés[1], Sergio Gallardo[1], Federico Barrero[1], Sergio Toral[1].

[1]Electronic Engineering Department. University of Seville.
Avda. Camino de los Descubrimientos, s/n. 41092. Seville, Spain
Tel.:+34 954 48 73 68, fax:+34 954 48 73 73, e-mail: sgallardo@gte.esi.us.es, fbarrero@gte.esi.us.es, toral@gte.esi.us.es

Abstract. Nowadays there are three basic technological supports which are suffering an enormous development. First we could mention field buses applications, which are getting more and more important in domotics. The second one, Web and Internet applications, is becoming popular and accessible in the last time. Finally, the third support, based on GSM networking, make one of the most important development focus up to now. The proposed system is based on the mentioned idea, the importance of the nexus between these three technological supports, developing a domotic system using CAN, a field bus which has been used mainly in automotive, and interconnected to the Internet employing Ethernet networks and GSM nodes via GPRS protocol.

Keywords.

Field bus, domotic, DSP, Ethernet, GSM/GPRS

1. Introduction

The increasing complexity of control systems and especially the development of distributed control [1] demand communication systems suitable for their needs. This, joined to the necessity of open control systems and the evolution of the communication technologies, contributed the growing of the field buses, spreading quickly due to the advantages that they offer in contrast to traditional systems. On the other hand, Internet applications are becoming very popular in a fast and continuous way, under Ethernet applications, and due to the GPRS protocol, mobility in communications is a reality [2,3]. The third point which is one's duty nowadays refers to the new technologies, understood as a social revolution which is being incorporated in daily life quickly. These three technological supports have been joined in order to implement a domotic system using field buses and interconnected to users through the GSM telephony network, using the GPRS protocol, and via Ethernet for non-mobile applications. This prototype, designed by the Department of Electronic Engineering, focused on a home frame domotic system, being easily expandable to other fields of application, such as industry, security, automotive, sailing, etc. Particularly, it is going to be described the case of a home security system. In order to validate this system, a scale model is being designed so that it can be used as a prototype for testing purposes.

The system features will be the following ones:

1) *Expandable*. Being able to include new sensors and actuators without changing neither the protocols nor the software stored in existing elements.
2) *Open.* With clear specifications so that anyone can develop their own hardware and include it in the system.
3) *Remotely controlled.* To monitor the state of the house and to allow the intervention in the system from any PC connected to the Internet and from mobile terminals.
4) *With a model.* In order to achieve a test bench closer to reality and a place to validate the system.

2. Field buses and domotics

There are several field buses which have been used in domotics for the last years, such as X10, EHS, BatiBUS and Lonworks. In the following lines some of them are going to be explained.

X10 is a standard which takes advantage of the existing electric infrastructure in the house in order to control devices. It uses bursts at 120 KHz which are transmitted everytime the signal of the electric network reaches 0V, this allow an average transmission bit rate of 50 bit/s. In spite of being very easy to install, the speed of this system is very low and it has a reduced functionality. X10 basically allows the user to switch on/off a device and control the brightness of lights.

EHS is an open standard developed in Europe which can use the electric infrastructure or a separated pair. It sends the information using a FSK modulation and it is able to reach bit rates up to 2400 bit/s. One interesting feature of this standard is the fact that EHS devices can be configured automatically (Plug&Play technology).
BatiBUS uses a medium access technique similar to CAN (CSMA/CA, Carrier Sense Multiple Access with Collision Avoidance) but it is only able to work up to 4800 bit/s. The user has to configure manually the address of every BatiBUS device, like in X10.
Lonworks can be classified as a high performance field bus and it is intended to be used in domotic applications. It allows devices to be interconnected in a hierarchical architecture where the different subnet are integrated to make a big domotic network with connections to the Internet. It supports several transmission mediums and can work at high speed (Up to 1.25 Mbit/s). Although it is said by Echelon, the company which designed it, that Lonworks is an open standard, the reality shows that it can be only operated by using a certain kind of processors registered by Echelon and called “Neuron Chip”. Lonworks provides fast and robust networks but on the other hand, they are expensive and complex.
CAN meets many of the advantages of other field buses in a really open standard. It is simple, cheap and it gives more freedom to design protocols of the higher layers. This, joined to the features described below, made us choose CAN as a support for communications inside the domotic system.

3. CAN (Controller Area Network)

CAN [4] is a kind of field bus designed by Bosch and it is widely used in automotive. This field bus is very simple because it is made of only two wires where data is transmitted in differential mode, reaching a speed up to 1 Mbit/s. All this makes CAN easy to install and robust. Remember it was intended to be used in environments with strong electromagnetic fields.
The two main features of CAN are:

1) At the data link layer, CAN protocol does not use identifiers as node addresses but they introduce the content of the message. For example: We could assign an identifier for temperature measurements, another for speed, etc. Therefore, point to multipoint communications can be established because only the nodes interested in that message will receive it.
2) The medium access type is CSMA/CA, so it does not spend when a collision occurs due to collisions are resolved by using a bitwise arbitration that wins the node with higher priority without modifying any bit the winner node has transmitted. A similar mechanism is used in ISDN “D” channels.

CAN also have error detection mechanisms in the link layer frames such as CRC and acknowledgement fields. Its “fault confinement” mechanism can also isolate defective nodes in order not to obstruct messages from other nodes.

4. System Topology

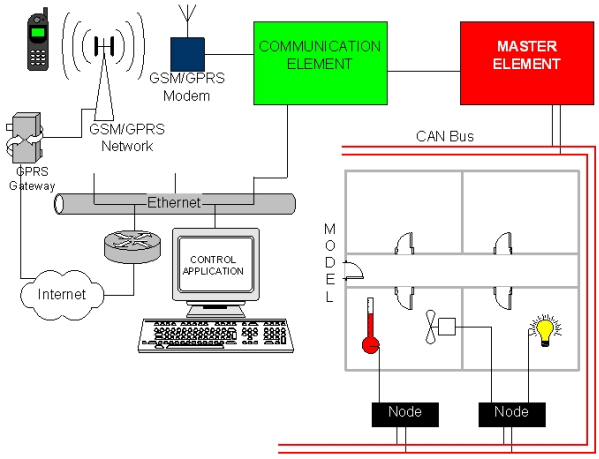

Fig. 1. System structure

The previous figure shows the distribution of the elements inside the system. We can point out the master element, which is responsible for the system control. The device which carries out the functions of the master element is a Texas Instruments® DSP, concretely a TMS320F2812. The master element performs the following functions:

1) It stores and manages the general information about the system, such as the distribution of sensors and actuators over the sectors or rooms, IP address, subnet mask, password, telephone numbers to send alarms to, etc. For this purpose it has a EEPROM memory.
2) It manages identification numbers of all elements (sensors and actuators).
3) It sends user commands to the elements and picks up all data which has been applied by the user.
4) It generates alarms if the corresponding conditions occur.

Up to now we have installed sensors and actuators in our prototype that produce a kind of information which cannot be considered as a high computational load for the DSP. In spite of this, in the future, the system might need to process some signals with complex algorithms and that is the reason why we have chosen a DSP for our prototype. In fact, we are thinking about introducing a web cam in the system and carrying out some image processing tasks.

Another of the most important elements in the system is the Communication element. It is basically a Rabbit Semiconductors® RCM2200 module and it contains a Rabbit 2000 processor. This module is intended to be connected to 10/100BaseT Ethernet networks, and it has a RJ-45 connector incorporated for this purpose. This was one of the reasons why we decided to choose this module for communication tasks. This module also

controls the GM29, a Sony-Ericsson® GSM/GPRS modem, through one of its asynchronous serial ports and it is permanently connected to the master element through other of its asynchronous serial ports. The software created for this module mainly acts as a unified server for remote access to the system. Therefore, the same server is able to give access to the system both if the user uses a PC connected to the Internet or if the user uses a GPRS mobile terminal. The RCM2200 module is placed over a base board created specifically for this project because the module cannot work by itself. This board has been designed according to the functions it is going to carry out in the system but it is also useful as a development system for the Rabbit 2000 if it is used outside the prototype.

The GSM/GPRS modem is used basically to send alarm messages to the user and the corresponding emergency service through SMS messages. We have decided to use SMS messages to send alarms because they are received almost instantaneously.

The different elements (sensors and actuators) are grouped by nodes. Each node can have one or more elements. The type of these elements can be different and they can be placed in different sectors or rooms. The nodes which have been implemented in the prototype contain a Microchip® PIC microcontroller because they have analog inputs with analog-to-digital converters and they have also a CAN module incorporated. Although these processors are not intended to make complex calculations, they meet all requirements needed to control temperature sensors, light sensors, proximity sensors, etc. They even include a little EEPROM data memory where the identifiers of the elements controlled by the node can be stored. CAN nodes have been implemented as generic boards with a group of connectors so that any sensor or actuator can be attached to the node. It can be done provided the signals are adapted previously to meet the electrical specifications for the PIC18F248 and the software has been carefully modified in order to manage the new element without change the behaviour of the elements which had already been attached.
There are three kind of sensors installed in the model:

1) Temperature sensors. Based on type K termocouples.
2) Ultrasonic proximity sensors.
3) Light sensors. Based on fotorresistors.

There are three kind of actuators installed in the model:

1) Lights.
2) Fans
3) Manual switches

All the system has been installed on a model created for this project. This model has a suitable structure for the system and can be dismantled and assembled easily.

5. Protocols

In order to control the different elements, two protocols have been designed. The first one is used in the CAN network and it has been called “internal protocol”. The other protocol is used between the server and the client application through the Internet and it has been called “external protocol”.

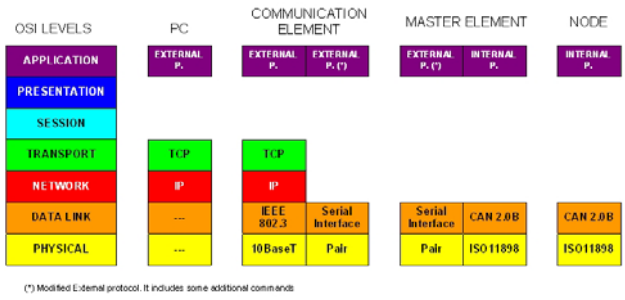

Fig. 2. Protocol scheme

The internal protocol is an application layer protocol although it is closely bound to the data link layer due to the CAN bus features. CAN uses the identifiers of the link layer to announce the content of the message. This identifier and the data field are in fact the application layer PDU. CAN 2.0 B specification has been used because it allows 29-bit identifiers. Those 29 bits have been structured in three fields:

TABLE I. – Identifier structure.

TYPE	COMMAND	NUMBER
11 bits	10 bits	8 bits

Field TYPE indicates the type of element which has originated the message or the type of element which should receive the message. Up to 2048 types of element are allowed. The lower the field TYPE is, the higher priority has the message. The reason for this can be found in the fact that dominant bits (0) prevail over recessive bits (1), according to the CAN specification, therefore, identifiers with lower values have higher priorities. Values from 00h to 1Fh have been reserved for urgent messages such as alarms, assignation channel, etc.

Any element can create a message to transmit some kind of information or to apply for others to carry out a certain action. Field COMMAND specifies which action has to be performed. This field allows having up to 1024 different commands for each type of element. Depending on the command, parameters may be included in the message. In this case, parameters would be placed in the data field of the link layer frame. For example: When a temperature sensor has to send the measured temperature, it uses command number 25 (VTEMP) and sends a byte in the data field that contains the value of the measured temperature in Celsius degrees.
There are also some common commands which have to be implemented by all the elements. They are used to get some general information about the element, like the model and the trademark, and to read and modify both operation and dependence parameters. These commands have reserved values which cannot be used for other purposes.

Field NUMBER is used to identify which element has sent the message or which element has to receive it if it was necessary. When messages are not sent to a particular element but a group of elements or when the element that sends or receive the message is necessary the only one of its type in the system, such as alarm messages, there is no point in filling this field, so its value has to be 0 in these cases. As this field is 8 bits long, there may be up to 255 elements of the same type in the system (value 0 is reserved). These numbers are assigned by the master element through a Plug & Play mechanism.
Field NUMBER together with field TYPE define unambiguously a certain element of the system.

It is evident that everyone who decided to manufacture elements for this domotic system should know and comply with the rules for the registered types of elements, so there should be an entity which published that information and assigned values to the new types of elements. This problem does not happen only in this system, but it appears in other applications such as Internet domains, MAC addresses for Ethernet boards, etc.

The external protocol is an application layer protocol and it operates over TCP. This protocol allows the user to control the system both from a PC connected to the Internet and from a GPRS mobile terminal. The application for PCs is made with Java (J2SE) and the application for mobile terminals is made with Java 2 Micro Edition (J2ME) and it does not implement the whole protocol functionality due to the limited resources these devices have, therefore, this application can only perform basic actions.
The application for PCs uses a special kind of files called "handle files". These files behave like drivers in conventional operative systems and they must be given along with every element. Since the application uses handle files, the external protocol does not need to be modified every time a new type of element is installed because these files *tell* the application how it must manage the element and which functions are available for that element. In order to perform specific functions, the application can compose frames helped by the handle files. These external protocol frames wrap an internal protocol frame, which is released at the master element. From this point, the internal protocol frame runs through the CAN network towards the target element. In this way, neither the server nor the client application have to know the meaning of the wrapped frame.
According to the nature of the interchanged information, three communication schemes have been defined:

1) Value. When the information is a value shorter than 8 bytes. For example: measured temperature
2) Block. When the information is a block of bytes (more than 8 bytes). This scheme is suitable for character strings. For example: Trademark of the element.
3) Series. When the information is made of values sent periodically. This scheme is suitable for monitoring signals. For example: Monitoring temperature.

After a reset, the master element automatically looks for new elements. If it finds a new node, it will assign numbers to the new node and the elements which belong to that node. Once the numbers have been assigned, the node stores them in its non-volatile memory. On the other hand, the master element registers the new elements in its EEPROM memory. It stores the following parameters by each new element:

1) Node which the element belongs to
2) Number
3) Type of element
4) Sector where it has been placed

Obviously, nodes cannot give the last parameter to the master by themselves. The SECTOR parameter refers to the zone of the house where the element is placed. To make it easier, we recommend to assign a sector by each room in the house. The SECTOR parameter must be given by the user through the client application. When there are elements pending on this parameter, the master notices it to the communication element so that it can notice it as soon as the user connects to the system.
In order to make the elements to behave correctly, they may require the user to specify some parameters. For example, a temperature sensor can work as a thermostat provided we have specified an upper limit and a lower limit. Nodes have to be able to keep this parameters at least in RAM memory. They might be stored in non-volatile memory also, but it is not a good option for some parameters because they may change many times and non-volatile memories like EEPROM or Flash can be programmed only a limited number of times. Furthermore, these memories might not be available for all nodes. The nodes which have been implemented for the prototype only have a 256-byte EEPROM memory.
The parameters are classified in two groups: operation parameters and dependence parameters. Both operation and dependence parameters can be read and modified by using a set of common commands.

Operation parameters usually contain values which determine the general operation of the element. In the previous example, parameters "upper limit" and "lower limit" would be operation parameters because they influence directly the behaviour of the thermostat. It is important to say that loading a value in a parameter does not imply it is going to be used, so if you modify "upper limit" parameter, the temperature sensor is not going to behave as a thermostat unless you send the "MTST" command (Activate thermostat mode) to that element.

Dependence parameters are used to specify which elements are going to influence the behaviour of a certain element. This is very useful when there are elements that depends on the information given by other elements. For example, a fan will depend on the temperature measured by a nearby sensor. If the temperature is higher than its upper limit, which has been previously determined as a

operation parameter, the fan will turn on. The node associated to the fan will have to reserve an space in its memory for the dependence parameter, where it will store the number of the temperature sensor specified by the user. When working, the fan will listen to that sensor and will turn on/off depending on the measured temperature.

Handle files contain the information that the application need to manage an element. These files are structured in five sections in order to provide an organised information easily readable by the application:

1) General Features. This section contains general information about the element such as the type of element.
2) Operation parameters. This section contains a description of each operation parameter available for the element. The number of the parameter, the mnemonic, the data format, and the parameter description are defined for each parameter.
3) Dependence parameters. This section contains a description of each dependence parameter available for the element. The number of the parameter, the mnemonic, the type of influential element and a parameter description are defined for each parameter.
4) Command parameters. This section contains a description of each command parameter available for the element. Command parameters are not readable or writable by using common commands because these parameters can be used only with certain commands. The information for each command parameter is the same than the one for operation parameters.
5) Commands. This section contains a description of each command available for the element. The command number, the mnemonic, and the list of parameters (optional) are defined for each command.
6) Actions. This section contains a description of each action available for the element. Actions are groups of related commands that perform an specific task. The description of the action, the mnemonic, the applied commands, the answer commands and the commands able to abort the action are defined for each action.

The client application shows the user the operation parameters, the dependence parameters and the actions available for the selected element. If the user wants to modify a parameter, the application gets the data format for that parameter from the handle file so that it is able to show or ask for the value in a suitable way.
If the user wants to perform an action, the application gets the information about that action so that it is able to know which commands it has to wait for and which command sequence it has to send to perform the action. If actions were not defined, the user should know the communication scheme for each command and should send and receive manually each command. It is obvious that actions simplify the use of the application.

6. The client – server architecture

At present we are building the client applications, which will allow the user to do many things. The application for PCs will be able to point the elements over a house map and draw the evolution of monitored signals. The applications are being made according to the chosen option, which is explained below:

The communication element is intended to act as a unified server and assist user requests through an specific port (5001). In order to access the server, the user must have the client application and the handle files installed in the PC. The application may generate some configuration files such as lists which link the elements with their handle files. Of course, the client application has to know the IP address of the communication element, so this element must have a public IP address in order to be accessible from somewhere. This becomes a problem when the system is attached to a LAN which uses virtual IP addresses and firewalls because the application cannot connect to such virtual addresses and firewalls usually do not allow transmitting data using that port. There are many applications which need to be installed to work, this has its advantages and disadvantages. Among the advantages we can point out the freedom and simplicity when designing and the possibility of developing specific protocols that increase the functionality. One of the disadvantages is the necessity of having the application and the handle files installed in the PC. The user should carry all the files in order to be able to access the system anywhere. The second disadvantage is the necessity of a public IP address for the communication element as we have explained before.

We also thought about other options. One of them is to implement a web server in the communication element in order to be able to control the system via a web page. Although it is possible to implement web servers in that element, functionality would be reduced because some web page formats, like HTML, do not offer all the flexibility and functionality of a specific protocol and it would be more difficult to program. This option also needs a lot of memory and the RCM2200 might not have enough memory to work properly. In spite of the ease for the user, who would not need to set up any program and carry any file, the disadvantages are bigger. This option has not been totally discarded yet because we are studying the possibility of using PHP pages, which add a higher functionality, but this would have the problem of the lack of memory too.

An option derived from the previous one is to implement a simple web server which would serve an HTML page with a Java applet. This Java applet would be the client application and it would work like the application in the chosen option. In this case, a specific protocol could be used but the communication element would have to implement two servers, the web server and the specific domotic server. It would be difficult to program and the problem of the lack of memory might appear again.

A better solution based on the previous option is being studied. The communication element would implement only the specific domotic server, like in the chosen option, but the HTML page, the Java applet and the handle files would be allocated in a conventional web server. The user would not have to set up any program and carry any file. This option would keep the functionality and would not be very difficult to design but it has two problems. The first problem occurs also in the chosen option and it has been mentioned before: The necessity of having a public IP address for the communication element and the problem with firewalls. The second problem would appear when the application tried to save some configuration information. The application might need to update some configuration files and the application should provide a mechanism to save that configuration information in the conventional web server or inside the EEPROM memory of the master element if there were enough memory.

7. Conclusions

Three important ways of development, the NNTT based on websites applications, Ethernet & GSM communication interfaces and field buses industrial protocols are joined in order to obtain a high performance system. This tool will be also used as an educative tool in training Telecommunication Engineering students. It will be versatile, open and expandable.

References

[1] Héctor Kaschel and Ernesto Pinto, “Análisis del estado del arte de los buses de campo aplicados al control de procesos industriales”.

[2] “A world of information in your pocket” Curtis, K.; Brown, K.; Pilkington, R.; Personal, Indoor and Mobile Radio Communications, 2000. PIMRC 2000. The 11th IEEE International Symposium on , Volume: 2 , 18-21 Sept. 2000 Pages:1017 - 1021 vol.2

[3] Implementation of mobile information device profile on virtual lab. Kumar, A.; Nambi, A.; Information Technology: Coding and Computing [Computers and Communications], 2003. Proceedings. ITCC 2003. International Conference on , 28-30 April 2003. Pages:612 - 616

[4] CAN Specification Version 2.0 1991, Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart