Implementing IEC 60870-5 data link layer for an Open and Flexible Remote Unit

Enrique Dorronzoro *, Isabel Gómez*, Ana Verónica Medina*,
Jaime Benjumea*, Gemma Sánchez*, Sergio Martín* and David Oviedo*
*Departamento de Tecnología Electrónica
Universidad de sevilla, Spain
Email: enriquedz@dte.us.es

***Abstract*—This paper presents an open source implementation for a data-link layer protocol specified in IEC 60870, protocol specification for telecontrol networks. It has been tested over LEON an embedded system with a Linux based operating system.**

Protocol engineering methods have been used in order to implement the protocol. The standard is in natural language so a formal language is needed to describe its behavior. A prototype has also been created to simulate the protocol behavior.

The protocol has been tested on a real environment, using PCs and LEON as primary and secondary stations, and different physical layers, serial cable, radio frequency and GSM.

I. Introduction

Nowadays, communications in telecontrol networks are designed for workstations that are rich in resources, high performance microprocessors and numerous kinds of interfaces.

Although there are some implementations for embedded systems, the main problem with these implementations are that they are not open source. It implies two problems, the user is tied to a vendor, and there is no access to the source code, so no adaptation can be made.

Telecontrol protocol stack usually implements the specification provided by the International Electrotechnical Commission (IEC) called IEC-60870 [1]. This document, which specifies a suite of protocols, is divided into six parts and specifies an application-layer protocol and a data-link layer protocol. This standard also defines a combination of the application layer using TCP/IP transport services.

The Open Flexible Unit project (called OFU), funded by Junta de Andalucía and Multimedia Operatives Techniques applied to Supply Electric Networks project (called TOMARES), funded by the Ministry of Education and Science of Spain, seek the development of a remote unit using other alternatives, distinct from those usually found in current networks with these kinds of devices.

OFU is focused on open software and hardware. It has been specified by LEON [2], a SPARC-based processor, which can be implemented on a XILINX Spartan-3 FPGA. As its operating system is Linux based (Linux Debian for Sparc has been installed in the system [3]), it is possible to develop the software using PC developing tools. On the same way, testing and debugging are possible using PCs and embedded systems. For this project a gcc compiler has been used .

OFU is designed to be flexible even on the physical layer. As it is designed to work on mobile units, Global System for Mobile communications (GSM) and Radio Frequency (RF) have been chosen to evaluate the protocol.

GSM infrastructure has established technical support bases. In metropolitan areas, GSM service provides high reliability and transmitting power and coverage are constantly increasing.

RF is offered as an alternative to GSM, it reduces cost to none and can work on areas where GSM has no coverage. Speed transmission is lower than using GSM, but the main problem is that RF only works in half duplex mode.

As the hardware in OFU has been designed to be very portable, the interface of communication to the physical layer is via the RS-232 port as it is the only I/O interface. Data are taken from this port by a modem, walkie... keeping transparent for the data-link layer.

Data-link layer also has some parameters with variable values, like number of retries. It is important to study the impact of these values in a communication, in order to get a criteria to set them to a specific number in the real implementation.

Point-to-point data-link layer protocol (PPP [4]) has been designed to operate in full duplex. Due to this limitation, the development of the IEC 870-5 data-link layer has been necessary.

This paper is organized as follows; first, it describes the IEC 60870 -5 protocol suite (section 2) followed by an introduction to the IEC 60870-5 data-link layer protocol (section 3), once the protocol characteristics are described, implementation steps are explained (section 4). Testing has been made using RF and GSM over PCs and LEON (section 5). Finally, some conclusions are presented (section 6).

II. IEC 60870-5 Series

This series follows the EPA (Enhanced Protocol Architecture) model, where there is only a stack of three layers, application, data link and physical, compared to the OSI model that has seven layers.

This stack defines the standards to allow real time telecontrol applications to take place. It is mostly used to communicate a primary station with some secondary stations.

It is not a closed standard but there is a set of options. In order to implement this standard, the first step is decided from this set the needed functions.

Physical layer is based on ITU-T. It provides binary communication without memory.

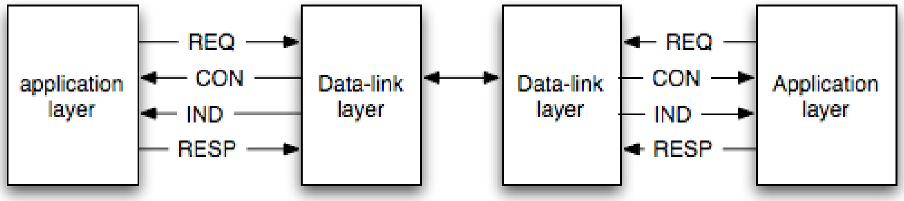


Fig. 1. Data-link layer service primitives

Data-link layer is based on a serial transmission bit oriented. It operates in full duplex or half duplex mode, window size is equal to one and it is possible balanced or unbalanced transmission. In unbalanced model stations do not have the same role. There is a primary station in charge of querying the secondary ones. In balanced mode, any station can send data even though the primary station has not polled them.

There are three kinds of services, described in section 3 of this paper, specified in this layer.

Application layer offers two kinds of services: unconfirmed and confirmed. It also defines application functions that describe the behavior of the standard procedures between primary and secondary stations.

Although IEC 60870-5 specifies a transmission profile for messages exchange between a primary and secondary station, in networks with more than one segment data-link layer does not work . Because of this, apart from the EPA, IEC also specifies how to operate with TCP/IP(Internet Protocols). IEC 60870-5-104 define the similarity between application functions described in IEC 60870-5-5 and services offered by TCP/IP.

III. IEC 60870-5 PROTOCOL CHARACTERISTICS

The IEC 60870-5-2 protocol is a specification for a data link layer. It is used for telecontrol in serial transmissions, for monitoring and controlling process in wide geographical areas.

There is a primary station that coordinates the communication. This station can survey the secondary stations asking them for information. The windows size of the transmission is set to 1, this means that the primary station only accepts a new message, when previous communication is finished, either with succeed or error.

The IEC 60870-5 can be applied with balanced and unbalanced transmissions. Our environment requires an unbalanced transmission.

As the protocol works on half-duplex mode, it is adecuated for RF.

There are three different kinds of services and four kinds of primitives.

The services are:

1) SEND/NO REPLY: Primary station sends a message to a secondary station. There is no error or success notification from the secondary station.
2) SEND/CONFIRM: Primary station sends a message to a secondary station. Primary station waits for a positive or negative acknowledgment from the secondary station. In case of error or no notification, an error message is sent to the application layer.
3) REQUEST/RESPOND: Primary station makes a survey to the secondary station. If this station has data to transmit, it sends them to the primary station.

SEND/CONFIRM and REQUEST/RESPOND services imply a dialog between the primary and secondary station. In this dialog the window size is one.

The primitives (Fig. 1) are:

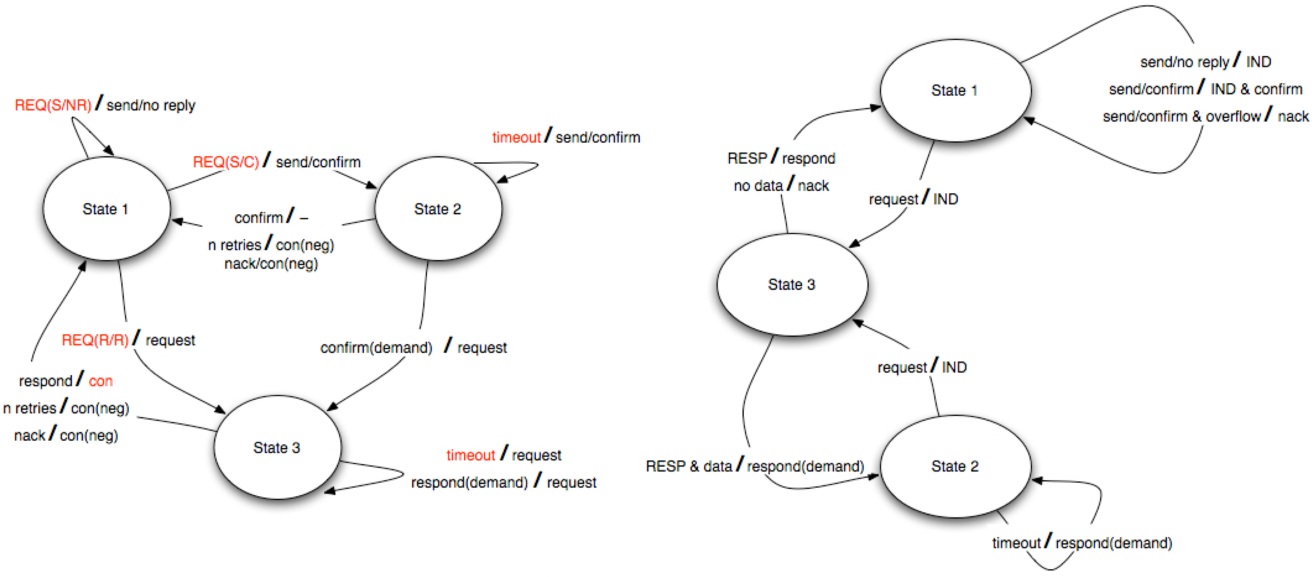
1) Request Primitive (REQ): a REQ is sent by the application layer to request a certain process to the data link layer.
2) Confirmation Primitive (CON): a CON is sent by the data link layer to the application layer. It is the response to a REQUEST primitive.
3) Indication Primitive (IND): an IND is sent by the data-link layer to inform the application layer that a message has arrived.
4) Response Primitive (RESP): a RESP is sent by the application layer as an answer to a previous indication.

When a primitive is sent by the application layer to the data-link layer, data-link layer generates a service. In case of a service that requires an acknowledgment the service is sent a number of times, called retries, until it gets response or it exceeds the number of maximum retries. This maximum retries is sent by the application layer in the primitive parameters.

IV. IMPLEMENTATION OF IEC PROTOCOL

In order to develop IEC protocol it is necessary to use protocol engineering techniques. Using natural language to describe it creates problems like ambiguity, and it would be difficult to be sure about its functionality, actually, there is a extended agreement about using formal description techniques as an adequate way of describing the behavior of protocols. Formal Description Techniques (FDT) is used to name any technique or method that allows to define completely the behavior of a system (hardware or software) using a language with formal syntax and semantic. Because of that, it is ideal to describe the protocol behavior.

The Finite State Machine (FSM) was designed from the IEC 60870-5 standard after reading and studying this standard.



Primary FSM	
REQ(S/NR)	REQ primitive using SEND/NO REPLY service
REQ(S/C)	REQ primitive using SEND/ CONFIRM service
REQ(R/R)	REQ primitive using REQUEST/RESPOND service
n retries	Number of retries exceeded
nack	negative ack message received (secondary station cannot accept more data or has no data to send)
CON	CON primitive generated
CON (negative)	negative CON primitive generated
confirm	confirm message received
respond	respond message received (data sent by the secondary station)
respond(demand)	respond message received (secondary station sent data and ask for more data to send)
timeout	no response form the secondary station
confirm	confirm message received

Secondary FSM	
RESP	RESP primitive
RESP & data	RESP primitive and demand request for sending more data.
send/no reply	send/no reply message received
send/confirm	send/confirm message received
send/confirm & overflow	send/confirm message received and overflow on the receiving data queue
no data	no data to send
IND	IND primitive
IND & confirm	IND primitive generated and confirm message sent.
timeout	no response from the primary station

Primary station FSM

Secondary station FSM

Fig. 2. Primary and secondary FSM

From FSMs is easy to use a formal language like Estelle. Estelle is a FDT defined by the International Organization for Standardization (ISO) for protocol specifications [5].

As shown in Fig 2 primary state machine is composed by three states. At state 1, the initial state, the station can use any of the services. Only a service with confirm or response requirement would change the state.

If a REQ primitive is sent by the application layer, a send/confirm is generated and a transition to state 2 is triggered. In this state the station waits for an acknowledgment to return to the initial state and the conversation is over. But if a confirm (demand) is received a transition to state 3 takes place. At this state the station continues polling the secondary station until a respond without demand is received or the connection with the secondary station is lost.

In secondary state machine, Fig. 2, state 1 is the initial state. It waits until a service is received. Only the state is changed when a request has arrived. It generates an IND and waits until the application layer sends a RESP primitive. If there is more data to send at the application layer buffer, a transition to the state 2 is triggered.

Once the protocol has been described without any ambiguity next step is to proceed to implement a prototype that simulates the protocol behavior described by the FSM using a programming language, in our case C.

As the prototype is going to work on a computer, it is

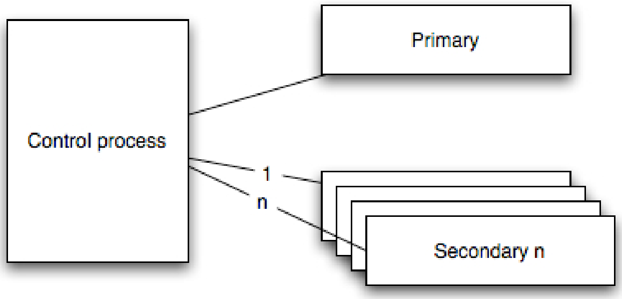


Fig. 3. Process structure

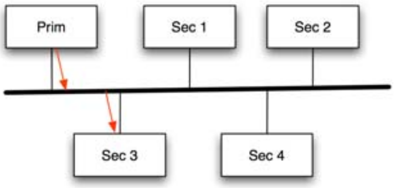


Fig. 4. Bus access example

needed to simulate the primary and the secondary FSM. This prototype launches different process for each station involved in the communication, so there is a process for every secondary station. Each process can communicate with each other using UNIX message queues; these queues simulate the physical channel.

The prototype is composed by a main process that launches and controls the other processes Fig. 3. This process also creates the message queues. This main process keeps a track on every process, and also creates a log file useful for debugging the behavior of the prototype.

In UNIX message queues, a station can write data in it. But if any station read from the queue, the message is removed from it, in this case the simulated channel. Because a bus topology is trying to be simulated, at the moment a station writes on the channel, any of the others can read and remove the data of the queue.

On Fig. 4 the message is written by the primary station, and read and removed by the station 3, so the other stations cannot read this message. To prevent this from happening, it is needed to tag the messages with a number, called address. So a station only reads a message in the queue if it is tagged with its address.

It is used the message type on UNIX message queuing to specify the destination station. Only the destination station reads and removes the message from the queue. As there are no broadcast frames, it is possibly to assure that no station will remove a message destinated to another one.

In real implementation the frames are not tagged. The physical layer is not in charge of coming to a resolution about which of the station is the destination of the frame. The shared medium assures that all stations receive the frames and only the destination station processes it. Only the prototype works this way.

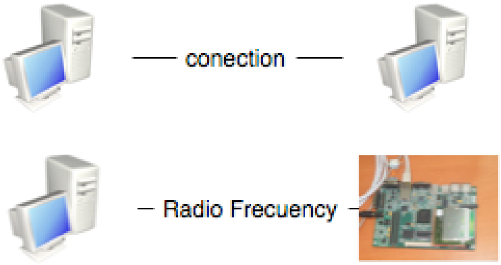


Fig. 6. Testing scenarios

However, it is not a problem. As the primary station only keeps one conversation with one secondary station and there is no broadcast frame, the behavior of the protocol is the same.

Another important problem when implementing FSMs is that when a FSM is waiting an event it consumes all time processor. Signals are used to avoid it, each process running a FSM remains sleep until an event is triggered. The main process captures the event and wakes up the appropriated station.

V. FIELD TESTING WITH GSM AND RF

Once the prototype has been debugged and everything is working on the simulation environment, the next step is implementing the protocol on a real system.

For testing the protocol the next scenarios have been developed: two computers connected via a serial cable, RF modem and GSM modem, and a computer connected via RF modem to the LEON embedded system.

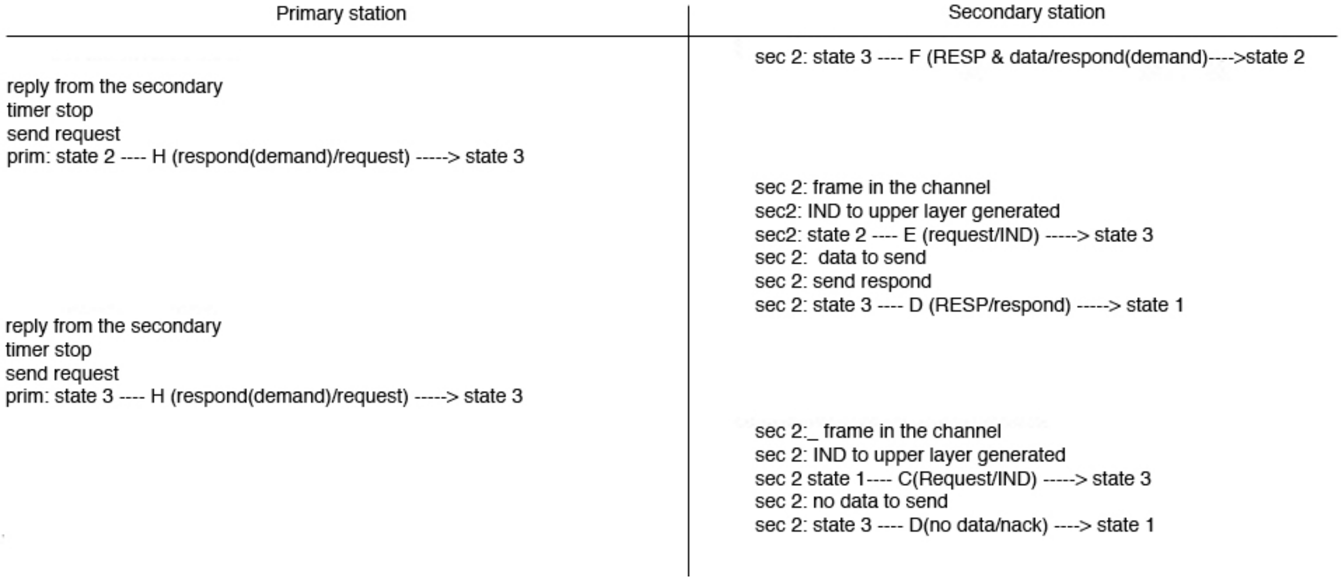
As not all the stations play the same role on the communications, unbalanced mode is used, there is different source code for a primary station and a secondary station.

In the development there is a process for each station, and the process code is the one that implements the FSM of that particular station. Separating the code is easy using this model, because the code of primary and secondary station are in different files instead of mixed in just one file.

As the application layer is not implemented yet, it has been simulated. Primary station queries a secondary station and this station responds with data: there is 80 primitives to send by the secondary station, so while there is data to send, the secondary station requests on demand the primary. The primary queries the secondary station until there is no demand response.

For the tests with RF and GSM the next devices has been used.

- RF equipment: An ICOM IC-V82 (VHF transceiver) with the digital unit UT-118. This device is D-star [6] capable, and can be connected to any RS232 device for data transmission, with a data transmission speed of 1200bps. The specifications for this equipment can be found at [7]



```
                Primary station                         |                 Secondary station
                                                        |   sec 2: state 3 ---- F (RESP & data/respond(demand)---->state 2
reply from the secondary                                |
timer stop                                              |
send request                                            |
prim: state 2 ---- H (respond(demand)/request) -----> state 3
                                                        |   sec 2: frame in the channel
                                                        |   sec2: IND to upper layer generated
                                                        |   sec2: state 2 ---- E (request/IND) -----> state 3
                                                        |   sec 2:  data to send
                                                        |   sec 2: send respond
                                                        |   sec 2: state 3 ---- D (RESP/respond) -----> state 1
reply from the secondary                                |
timer stop                                              |
send request                                            |
prim: state 3 ---- H (respond(demand)/request) -----> state 3
                                                        |   sec 2:_ frame in the channel
                                                        |   sec 2: IND to upper layer generated
                                                        |   sec 2 state 1---- C(Request/IND) -----> state 3
                                                        |   sec 2: no data to send
                                                        |   sec 2: state 3 ---- D(no data/nack) ----> state 1
```

Fig. 5. Simulated example log

- GSM equipment: Wavecom Fastrack M1306B GSM Modem. It is a device behaving as a standard AT-command modem via a RS232 port. According to devices specification, it allows a data transmission up to 14.400bps [8] but this feature is dependent on the GSM operator used, so it might not be available (in fact, our tests run at 9600bps).

As it is mentioned previously on this paper, LEON I/O interface is a serial interface. In testing field, LEON acts as a secondary station and communication with the PC, primary station, is by RF.

A. *Testing on serial cable*

Channel is error free, so no frames are lost. No timeout is expired on this test.

A trace from the test is in Fig. 5, the secondary station (SEC 2) reply to a previous query sending data, and because it has more data to send the reply is a respond (demand). While the secondary station has data to send, primary station continues requesting the data.

At the primary station Fig. 5, tabbed to the left in the trace, is at state 3. A respond (demand) is received, because of this, a request is generated and sent to the secondary station. Primary station FSM remains at the same state waiting for the requested data.

Secondary station, receives the request at state 2 and generates an IND. Its state changes to state 3 where it waits for the RESP from the application layer. When the RESP is received a respond is sent to the primary station and its state changes to state 1.

When all the data has been transferred, the dialog finishes.

B. *Testing on RF*

It has been tested RF communication connecting at the serial ports of two PCs two ICOM IC-V82, and every service has been tested getting the right results.

As the number of retries is an option available at the application layer, it is interesting trying to get the optimal number for best performance.

Channel errors have been simulated in order to control the quality of the communication. By modifying the source code it is possible to change an error parameter. This parameter expresses the percent of frames that are incorrect, both they are lost or they are wrong.

Changing these parameters (number of retries and error rate) it is possible to get some statistics in order to decide how many number of retries are produced for a given error rate. Each time the number of retries is exceeded data-link layer generates an indication to the application layer that can handle this error sending the information again or deciding not to do it.

Number of retries cannot be a very high number because while the data-link layer is trying to send the frame, communications are stuck. In an unbalanced mode only one transmission can be working at one moment.

A low number of retries generates a big number of indications. If the application layer decides to resend the data, the delay will be increased because application layer would have to process the data again and send it back to the data-link layer.

A big amount of traffic is generated for testing purpose, that is why at the application layer of the secondary station a big quantity of primitives have been generated. When the secondary station is polled it sends the information given by the application layer but if it has more data to send, it also demand the primary station for being asked for more information. Once all the data are transmitted, the polling is finished.

At this point it is recorded how many times the number of retries is exceeded. Based on this number and the penalty for an indication it is possible to choose the right limit.

TABLE I
PERCENT OF INDICATIONS GENERATED

Error rate	10%	20%	30%	40%	50%
Retries 0	9	22	29	42	52
Retries 1	1	3	5	12	15
Retries 2	0	2	3	4	7
Retries 3	0	0	0,8	1,3	2
Retries 4	0	0	0,8	1,3	2

Table I has been built from experimental simulation. It shows the number of indication generated depending on the percent of errors in the channel.

For example, with a error rate of 20% and a number of retries set to 1, it gets a 10% of indications. Because there is no real application layer, it is not possible to measure the penalty for each indication. But there is important information on Table I, it is a reference to check the estimated indications that a transmission would generate. When operating in the complete protocol stack with the application layer it will be possible to estimate the appropriate maximum retries.

Choosing a number of retires set to 1, instead of 0 in a medium with an error percent of 30, the number of indications will be reduced from a 25% to a 5%. But when trying more times to get an answer from a station it increases the time to poll another station. When dealing with important events stopping too much time on a station could be critical.

C. Testing on GSM

The protocol behavior has been tested over GSM. For the test two computers have been connected via modem over a GSM network. The modem is connected through the serial port and no change has to be made in the protocol implementation as it follows RS232 standards.

Protocol behavior has been equal than previous test as there is no application layer, or an important amount of information to be sent. The only difference is that over RF faster speeds are appreciated .

D. Testing on LEON(RF)

The protocol has been tested over LEON. As this embedded system works over a SPARC processor the PC compiled code it is not executable. A compilation on LEON has been needed.

LEON has a compact flash card, so it is possible to write the source code of the protocol on it using a card writer. The access to LEON is via a hyperterminal program, minicom, and the source is compiled using gcc.

The protocol works the same way as in a PC. As RF and GSM have the same I/O interface, the protocol behavior is similar.

VI. CONCLUSSIONS

On this paper an approach to telecontrol network communications is made. The IEC 60870 standard has been studied in order to get an open source implementation of a telecontrol protocol.

Developing an implementation for the IEC 60870-5-2 data-link layer is the first step in order to develop an application layer. The protocol implementation of the data-link layer has been made using FDT. A prototype has been developed in order to confirm the right protocol behavior.

As it is showed in section V, number of retries is not a standard value that can be configured the same in each implementation. It depends on the application layer, and it is a future line of research. However the protocol works correctly with different types of communications channels, as point to point, GSM and RF, and in an embedded systems like LEON (same PC source code compiled and executed in this platform).

Other future lines of research:

The IEC 60870-5-5 specifies an application layer for the IEC-60870 protocol suite. It defines basic application functions that perform procedures for telecontrol systems. The defined application procedures use standard services of the application layer. No all services have to be implemented so the required functions will be chosen depending on where the developed system is applied.

Future testing implies getting an optimal number of retries based on the channel error rate, that can be get by SNR and BER, and based on the application requirements. With those results it is possible to set minimum retries value in an automatic mode.

In order to work with other technologies and testing other options it is also been developed an application layer working with TCP/IP using transport services instead of data link layer.

ACKNOWLEDGMENT

This work has been undertaken in the framework of two research projects: OFU (EXC-2005-TIC-1023) - Open Flexible Unit funded by Junta de Andalucía and TOMARES (TEC2006-08430) -Multimedia Operatives Techniques applied to Supply Electric Networks funded by the Ministry of Education and Science of Spain..

REFERENCES

[1] International Electrotechnical comission, *International Stardard IEC-607870-5 (6 parts)*

[2] http://www.gaisler.com., *GRLIB/LEON3 manual"*

[3] A. Muñoz, E. Ostúa, P. Ruiz, M. J. Bellido, J. Viejo, A. Millan; J. Juan, D. Guerrero, *Un ejemplo de implementación de una distribución Linux en un SoC basado en hardware Linux* Actas de las IV jornadas de computación reconfigurable y aplicaciones (JCRA07), pp. 85-92, Sep-2007.

[4] http://www.networksorcery.com/enp/default0903.htm, *"PPP RFC."*

[5] Verónica Medina, Isabel Gómez, Joaquín Luque, Sergio Martín, *ESTELLE: A Method to Analyze Automatically the Performance of Telecontrol Protocol in SCADA Systems*

[6] http://www.arrl.org/FandES/field/regulations/techchar/D-STAR.pdf, *"Dstar system"*

[7] http://www.icom.co.jp/ world/products/pdf/IC-V82 U82 LM.pdf, *"ICOM IC-V82 brochure"*

[8] http://wavecom/media/files//support/Hard platforms/Modems/Fastrack M1306/User manual/Fastrack M1306B Userguide rev003.pdf , *Wavecom Fasttrack 1306M User Manual*