The Integration of Embedded Devices with Advanced Features in an OPC UA Architecture

Vasile Gheorghiţă Găitan and Alexandru Goloca

***Abstract*—The aim of this paper is to present an advanced embedded system that was used in the past to build a paperless recorder and that will soon be integrated in new OPC UA architecture. More and more embedded devices used in process monitoring and control are integrated in advanced OPC UA architectures. The proposed system will achieve better performances after the integration with the OPC UA architecture since its features will be better exposed.**

***Index Terms*—OPC UA, EMBEDDED LINUX, ARM, PROCESS MONITORING AND CONTROL**

I. Introduction

PROCESS monitoring is a large field of activity and there is a large number of devices that perform several monitoring tasks. There are single channel and multi channel indicators, intelligent sensors and paperless recorders.

A data logger or recorder is an advanced device used for data logging and display. Old recorders used paper as informational support and were heavy machines, with lots of mechanical components that were subject to periodic failure. In order to view the recorded data one had to go through meters of paper. A major disadvantage of such devices was that the user got little information about statistical data and had to manually extract it.

Nowadays recorders are a far cry from that: they use liquid crystal displays with touch screen capability in order to display the acquisitioned data and powerful embedded systems with flash memory to store the data. These devices offer advanced features like network connectivity through which data can be remotely downloaded and advanced web interfaces that allow configuration and control of the device.

Since paperless recorders can have both digital inputs and outputs, one can use such a device for both process monitoring and control.

II. The IDMC04 Paperless Recorder

The IDMC04 (Inregistrator Digital Multi Canal - Multi Channel Digital Recorder) is a paperless recorder developed by the GenPro Company. It offers advanced process monitoring and control features and is build using two interconnected embedded devices:

- A liquid crystal display with touch screen used as an Operator Panel (OP);

- A Linux embedded device that carried out all the data logging and communication with the outside world;

These two embedded devices were connected using Ethernet cable and Sockets were used in order to establish communication.

The way these components interacted is suggested in Fig. 1:

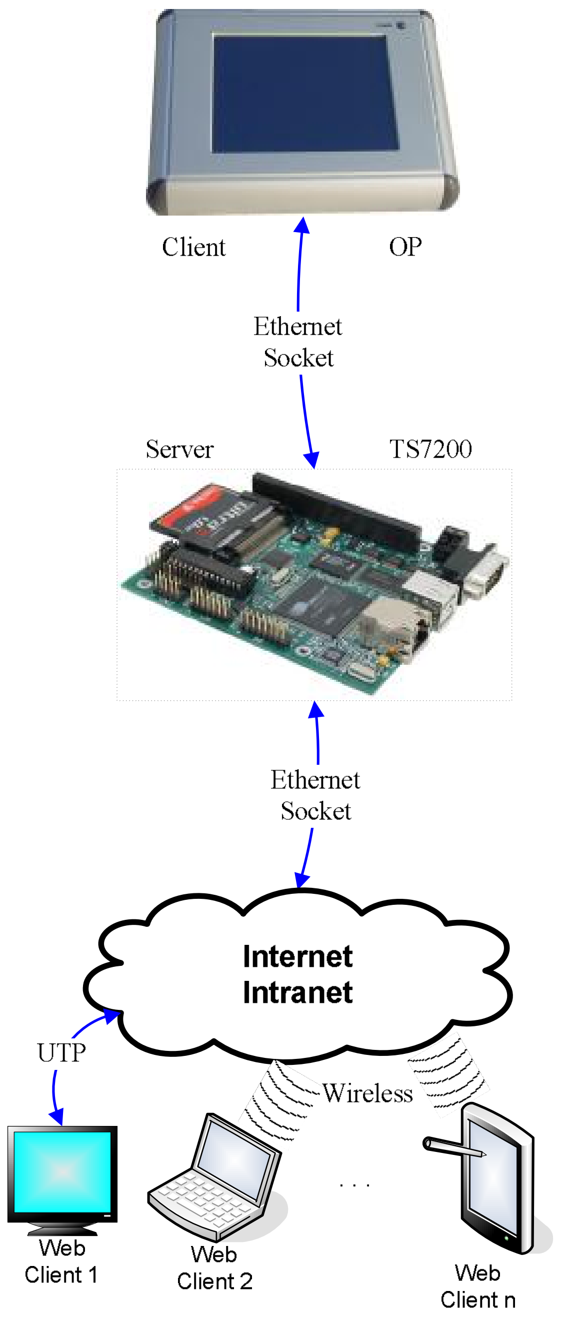

Fig. 1. Connectivity.

The Operator Panel (or simply OP) was developed on a VGT-001 LCD terminal with touch screen capabilities. It offers 640x480 resolutions and features a Windows CE® 4.20 operating system. The graphic interface is very rich and easy to use, allowing the user to customize the way that graphic controls look like and how they act using nothing but a pen.

The most important part of the IDMC04 is based on an embedded device named TS7200 [1], produced by Technological Systems. The main features of this embedded device lead to plentiful computational power and a large number of connections with the outside world.

- 200MHz ARM9 CPU (EP9302 developed by Cirrus Logic);
- PC/104 expansion;
- 32MB SDRAM;
- 8MB NOR Flash;
- 1 10/100 Ethernet;
- 2 USB 2.0 (12 Mbit/s max);
- 1 Compact Flash socket;
- 2 COM ports;
- 20 DIO lines;
- 2 12-bit ADC;
- Watchdog timer, SPI bus;
- Optional 8 12-bit ADC and RS-485
- Low-power (400mA @ 5V)
- Fanless -40° to +70°C, +85°C 166Mhz
- Small size: 3.8 x 4.5 inches
- Redboot bootloader, Linux out-of-the-box

The EP9302 processor offers more than sufficient computational power, thus allowing the development of many concurrent applications on the same device. This leaves room for many future developments and integrating the device in an OPC UA [7] architecture is such a planned activity.

There are some major drawbacks of the current architecture when it comes to operation and connectivity:

- There is a single Operator Panel and it is located in close proximity of the TS7200 module – they stand inside the same enclosure;
- The web interface does not allow any advanced configurations of the device;
- For each new web client that connects to the device there is a need for a new instance of the web server and this leads to an overload of the embedded system.

A solution to this problem is to integrate the entire device in a new and innovative architecture – the OPC UA architecture.

III. A Closer Look to the Software Architecture

The TS7200 module comes with an embedded ARM Linux [8] and allows the development of complex multithreading and multi process software applications.

The software residing on the TS7200 module was designed and written as more Linux processes that use several mechanisms to communicate with each other: message queues, shared memory, sockets.

Some processes running on the TS7200 communicate with the graphic application running on the Operator Panel using sockets as a middleware, providing online data and events, configuration information and history data.

Other processes from the TS7200 module are used to communicate with the outside world, providing online data and events to the client applets or providing history files.

The processes structure was suggested in Fig. 2:

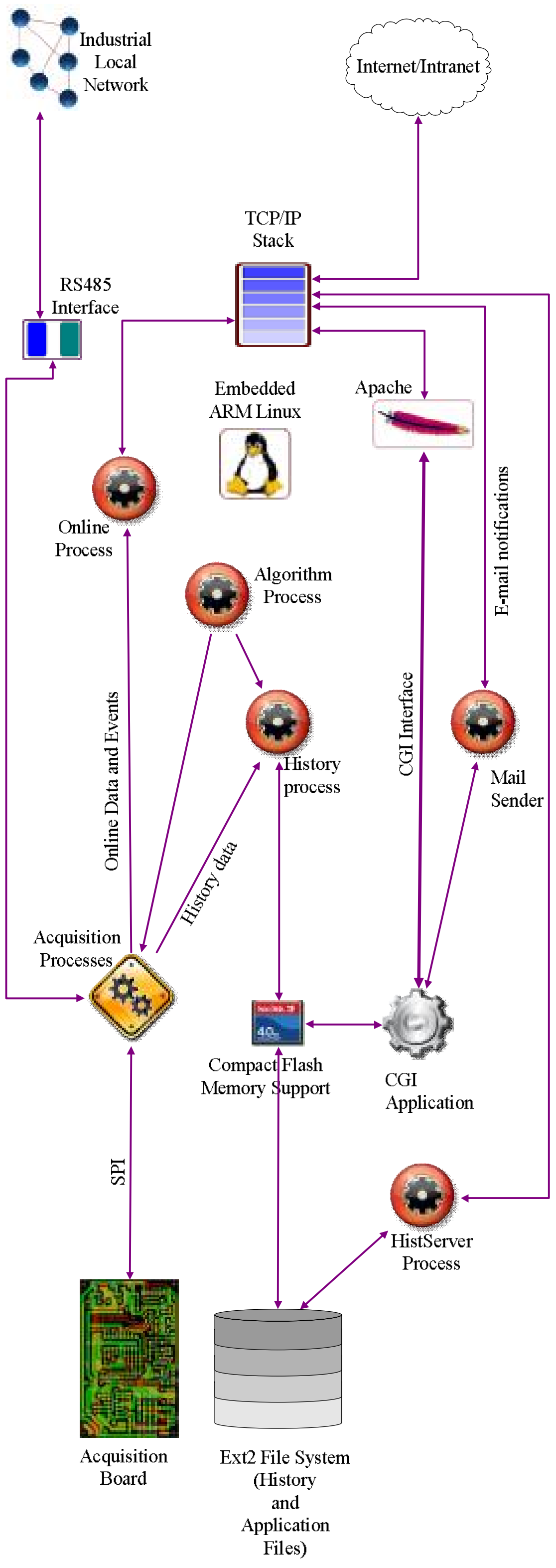

Fig. 2. Existing Software Architecture.

A. *Software model and features*

The software running on the IDMC04 device follows the hardware architecture, modeling the acquisition components. It is 100% object oriented and all the C++ and Java classes have been designed to offer the maximum amount of portability.

Acquisitioned data comes from acquisition channels. These acquisition channels are regarded as data streams, providing the device with a constant flow of information at a given rate (this rate is user-defined). Acquisition channels are organized in groups called modules. There are a number of real acquisition modules and also a virtual module, discussed later.

There are some actions that can be performed on the acquisitioned data:

- Online sending to the Operator Panel in order to be viewed by the human user or to one or more remote web clients;
- History logging for later download and use in statistical jobs or just viewing;
- Virtual data manipulation – acquisitioned data can be manipulated in mathematical operations right on the device, providing composite data.

History data can be downloaded using one of the two different ways:

- One of the two USB ports of the TS7200 board and an USB mass storage device (the device can be formatted with FAT32 file system and there is no need for a special USB stick);
- The web interface – one or more web clients can connect to the device and remotely download the files using a very simple, intuitive yet powerful interface which allows choosing a very precise time interval.

The main components of the software are divided into permanent running processes and processes launched on-demand. The structure is as it follows:

- The Acquisition processes;
- The Online process;
- The Algorithm process;
- The History process;
- The HistServer process;
- The WebServerOnline process;
- The MailSender process;
- The CGI Application;

B. *The acquisition processes*

The acquisition processes gather data from different sources:

- External Data Acquisition Modules (MAD – Modul de Achizitie de Date) that use a RS485 network as communication backbone and an ASCII protocol. These devices are managed by the External Data Acquisition Process;
- An internal acquisition board connected to the TS7200 module via a SPI interface, providing very good resolution (up to a quarter of a second) and also a CJC channel. This board is managed by The Internal Data Acquisition Process.

Acquisition data is organized in modules, a module containing several channels. Acquisition rate can be set for a module (all channels belonging to that module will provide data at the given rate) or for a single channel (the specified channel will be sampled with the given rate).

This process provides online data to the online process and to the History process.

C. *The Online processes*

This process provides online data coming from either one of the external acquisition modules or the internal acquisition board. Online data comes from the Acquisition Processes via message queues, events come from all running processes using message queues too.

This process uses a proprietary communication protocol over an Ethernet connection using Sockets as a middleware.

In future developments this process will be replaced by a ModBus [6] communication component, allowing online data and events to be passed over to an OPC server. This extension will allow for more than just one graphical client to extract online data from the device.

D. *The Algorithm processes*

This process manages the concept of data virtualization and virtual channels.

The device is equipped with a virtual module (one that has a software mode but no hardware exists for it) that provides user-configured virtual channels. These channels offer the possibility to implement a large number of mathematical operations, using acquisitioned data or software-generated values as operands. The concept of virtual channel is a strong tool for data manipulation, providing statistical and composite data straight from the device.

A special type of virtual channels is called Alarm. An alarm can be triggered when a user-defined condition is met or a certain event occurred, thus enabling the detection of a critical situation as soon as it appears. An alarm can cause data logging to be performed at a higher rate when it’s active, thus enabling a better resolution for the history data through the critical period, which is good for further analysis of the critical situation.

Alarm virtual channels can be directly associated with output one or more of the relays of the device, enabling the performing of a certain action when the alarm is triggered: e.g. ringing a bell, turning on a distress light or even shutting down the troubled device.

Triggering an alarm also has the effect of sending a notification message to registered web clients, with a detailed report on the situation that caused the triggering.

E. *The History processes*

This process gathers data coming from the Acquisition Processes through message queues and events coming from all running processes. Events are always saved in the history log since they represent important notices, while data logging can be enabled or disabled, for a single channel or for a group of channels.

There are also a very large number of parameters that can be adjusted in order to configure data logging and there are several logging algorithms implemented, such as conditional logging, threshold level logging (data is being saved only if one ore more threshold conditions are met), differential logging and so on.

Virtual channels can be saved into history memory as well as real acquisitioned data, thus enforcing the capability to provide statistical data. A virtual channel can be created using mathematical operations that go from simple additions to sophisticated formulas using data coming from other channels as inputs.

The history process implements a special selective erasure mechanism providing circularity for the history memory: old data logs are being deleted when space is needed on the memory, allowing the logging of new data to continue when the memory card gets full as shown in Fig. 3:

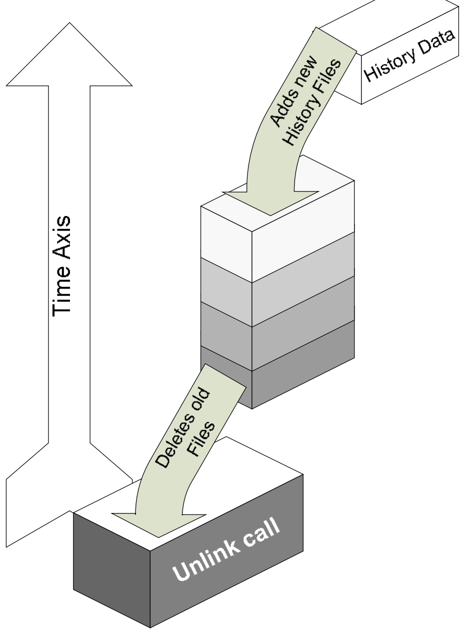

Fig. 3. Circularity of the History Memory.

The acquisition rate used for a channel can be the same as the logging rate or they can be different, in this case there are several options for which data to save. These options are as it follows:

- Maximum value of an interval – the maximum value acquisitioned during an interval is logged;
- Minimum value of an interval – the minimum value acquisitioned during an interval is logged;
- Both minimum and maximum values for the given interval – a logging point containing two values is created;
- The average value for the interval – the average value acquisitioned during an interval is logged;
- The first or the last value from the interval – this creates a logging point containing only a snapshot: the first or the last value, with no respect to whether it's maximum, minimum or average.

There are some observations to be made related to the History process. One of them is that solving the circularity problem for the history data brought a major drawback: old data might be deleted if there is no explicit action from a user to copy that data on an external memory support via USB flash memory or the web application. This drawback will easily removed when the device will be integrated in an OPC UA [7] architecture and there will be an OPC Server permanently requesting data and recording it on a remote memory support.

F. *The HistServer processes*

This process acts as a server and provides history data to the graphical interface. History data (saved by the History process) is recorded in so-called data points (a record that contains a time stamp and the associated values). The HistServer extracts the needed data points from the data memory and sends them to the client.

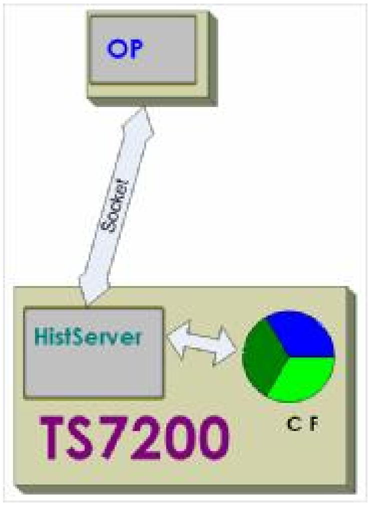

Fig. 4. Connections of the HistServer process.

There is one major drawback of the current architecture: there can be just one graphical client that can connect to the server and extract history data. This drawback can be easily eliminated when the device will be integrated in an OPC UA [7] architecture.

G. *The WebserverOnlineProcess*

This process provides online data to the online web clients represented by Java Applets running inside the web interface.

The process uses several communication mechanisms, as it follows:

- Shared memory zones: this mechanism is used to communicate with the acquisition process in order to retrieve acquisitioned data.
- Semaphores: this mechanism is used in order to implement inter-process synchronization between the webserveronline process and other processes using the same shared memory zone.

- Message queues: this mechanism is used to retrieve Events from the History process.
- Sockets: this mechanism is used to communicate with web clients found on the web and transfer data and events.
- Child processes: this mechanism is used to allow the communication with more than just one client at a time: for each new client trying to communicate there will be a new child instance of the process. After creation, this instance will deal only with its client.

There are some limitations that this process has and some of them are that the more client applets try to connect to the device, the more child processes are created. Currently, there is a limit of 16 child processes and thus 16 clients that are able to see the online data using the applets. This is a limitation that can easily solved by the integration in OPC UA architecture: the online data and events could be provided then by an OPC UA server to several clients.

H. *The MailSender Process*

This process acts as a daemon that connects to pre-configured SMTP [5] servers and delivers messages containing details about the alarms and events that occurred since the last sending. The amount of time between two mail sending can be configured by the web interface administrator as a global parameter but can also be overwritten by web interface common users, depending on their needs. Thus the interval can vary from a couple of minutes (if a very high resolution is needed) to a couple of hours (if the user is not concerned about very quick response but wishes to receive some statistics).

The process is implemented using the Linux pthread library for developing multi-thread applications: there is a separated thread running for each user that wants to receive E-mail notifications. This way there is a guarantee that the entire process will not halt if one sending cannot be done due to some reasons like an invalid E-mail address.

If one of the web interface users does not want to receive notifications, one can simply disable the feature via the web interface.

Mail sending is done using a temporization algorithm, as shown in Fig. 5.

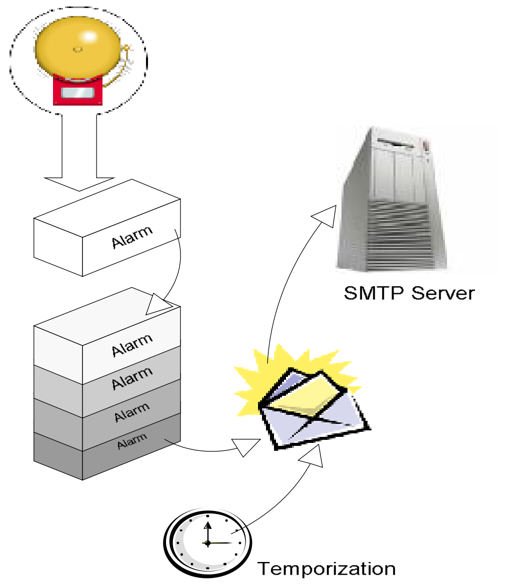

Fig. 5. The temporization of mail sending.

I. *The CGI Application*

This application was designed an implemented with respect to the CGI [2] specification and it has the purpose to offer an opened interface for web users.

The application uses Apache Basic Authentication [4] to filter users trying to access the interface: only authorized users can have access to the interface and non-registered users are rejected.

There are two types of users:

- The Administrator: a special user with extended rights;
- Regular users with limited rights.

After login, a user can interact with an upper control panel made of several links.

Using the web interface generated by the CGI application one can have access to online data and events and can also download history files from the device, both of this features being made available through Java applets that are part of the project.

Online data and events are provided by the WebserverOnline process while downloadable history files are provided by the Apache server [3] itself – the server was configured to include the history directory as a source for data available on the web.

There are some drawbacks of the current approach: the web interface is not very customizable – there is a degree of freedom but users don't have the possibility to have bar graph representations or trend graphs. This situation can be improved by integrating the device into an OPC UA [7] architecture where remote clients can offer the features that the current applets lack.

IV. Integrating the Device into an OPC UA Architecture

OPC UA [7] stands for Ole for Process Control Unified Architecture. This is a very modern and complex platform-independent standard that allows various kinds of systems and devices to communicate by sending Messages between Clients and Servers over many types of networks.

This newly developed standard supports robust, secure communication that assures the identity of Clients and Servers and is able to resist almost any kind of attack. OPC UA defines standard sets of Services that Servers may provide, and individual Servers specify to Clients what Service sets they support.

The next stage of the project is to integrate the device in a complex OPC UA Architecture.

By taking this step, the device will open its gates to a new world of communication and interoperability with other similar devices.

A. Benefits

As mention until now, the current software architecture might benefit a lot from this step. From the possible benefits one could mention:

- The possibility to have more than just one complex graphical client connected to the device and showing online data (the current implementation allows for a single complex graphical client and a maximum number of 16 simplified web clients);
- The possibility to obtain complex statistics for very long periods of time without the need to manually copy the history data thanks to the OPC UA Server [7];
- Very secure data transfer over various types of networks;
- The possibility to act as a central dispatcher, controlling complex systems.

B. Integration possibilities

There are several ways in which the current device can become part of a complex OPC UA Architecture [7]:

1. The first step is to transform it into an OPC Client by creating a process that is able to communicate using an industrial communication protocol such as ModBUS [6], Profibus [10] or CanOpen [9]. This will also allow the device to communicate with existing older versions of OPC Servers.
2. The second (and most ambitious) step is to transform the device itself from a simple data source for an OPC Server (as already suggested) to a central dispatcher by creating an OPC UA Server application based on the latest standards. This application will run on the TS7200 [1] module itself and, thanks to the new UA Specifications, it is no longer needed to use COM/DCOM as a middleware and can take advantages of the flexibility offered by the embedded Linux Operating System (the ARM Linux [8]).

A possible integration of the device might look as it is shown in Fig. 6.

V. Conclusions

OPC UA technology promises to become the leading actor in process monitoring and control. Built on the experience gained from existing OPC versions, UA comes to offer a complex cross-platform integration allowing complex applications to interoperate in a secure way.

By integrating the existing device into a Unified Architecture it will be transformed into a complex dispatcher, possible part of a larger system but also a system itself.

The resulting device will be a modern control center, with low power consumption but a very good integration within the industrial environment, thus being able to fulfill complex monitoring and control tasks.

By using CanOpen [9] as a communication protocol, the device will be able to interoperate with a large number of existing industrial devices using that protocol.

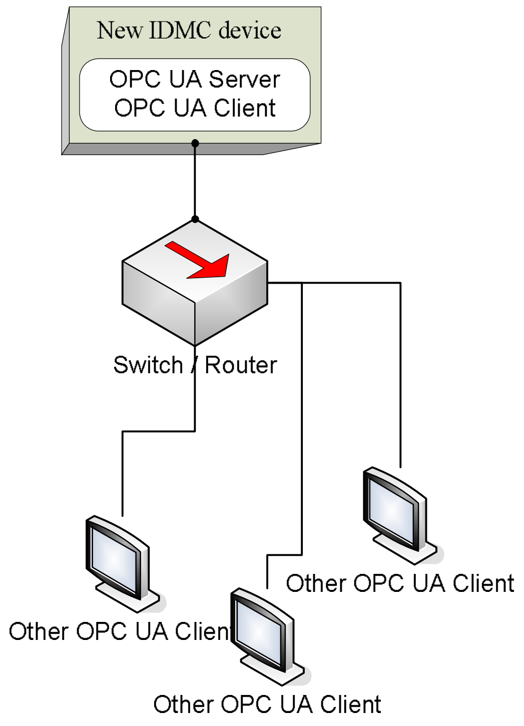

Fig. 6. The future integration of the device.

References

[1] http://www.embeddedarm.com/products/board-detail.php?product=TS-7200
[2] http://www.w3.org/CGI/
[3] http://httpd.apache.org/docs/
[4] http://httpd.apache.org/docs/1.3/mod/mod_auth.html
[5] http://james.apache.org/server/rfclist/smtp/rfc0821.txt
[6] http://www.modbus.org/
[7] www.opcfoundation.org/UA/
[8] http://www.arm.linux.org.uk/
[9] http://www.can-cia.org/
[10] http://www.profibus.com/