Design and implementation of an embedded system for particle detectors

A. Almela 1, A. Sedoski Croce 1, D. Alonso 1,2, A. Fuster 1, M. Hampel 1, S. Garavano 1, G. Pierri 1, M. Platino 1, O. Wainberg 1, M. Videla 1, A. Lucero 1, A. Etchegoyen 1, F. Suarez 1, A. Cancio 1, D. Yelos 1, G. De La Vega 1.

1 Instituto de Tecnologías en Detección y Astroparticulas (CNEA, CONICET, UNSAM).
2 now at: Institut für Industrielle Informationstechnik (KIT).

alejandro.almela@iteda.cnea.gov.ar ; adrian.sedoski@iteda.cnea.gov.ar

Abstract: This work describes the design and implementation of an embedded system that provides centralized control, data acquisition and environment monitoring to particle detectors. The system was designed for low power consumption, allowing the use of renewable energies as power sources. The system runs an open source operating system, allowing to write applications that are independent of the particular hardware implementation used on the detectors. Communications are provided using an Ethernet network within the detector, and 802.11n wireless radio links for communications with a centralized data storage and control server. An implementation of this system is being deployed as part of the AMIGA muon counters. This implementation and its performance verification, which were carried out by ITeDA, are also discussed in this work.

Keywords: embedded system, particle detector

1 Introduction

Particle shower experiments using detector arrays usually have many detectors covering a wide area. Each detector needs communications with a data storage system and a power supply used to power the electronics of the detector. These experiments can be located in hard to reach areas, and are expected to work for many years. These characteristics require systems with a long lifetime, and able to have remotely accessible diagnostics and early fault detection. In order to ease hardware upgrades, applications running on the embedded system must be written in a way that minimizes the dependencies on the hardware used in the modules. A portable operating system was chosen as an abstraction layer between software and hardware.

2 Station overview

The proposed design divides the array into detection stations. These stations continuously send lists of timestamps which indicate the presence of a first trigger condition for a possible event. These lists are analyzed by a central server, which sends data requests to the corresponding stations when an event is found. Each station has the following components: one or more detection modules acquiring and sending data to a storage system, a trigger source synchronizing data acquisition, a communications system connecting the modules with the storage system, and a power source powering all the electronics within the station (figure 1).

The electronics for each module consist of three stages: Front-End, Acquisition, and Control and Communication. The Front-End stage acts as an interface between the particle detector and the Acquisition stage. The Acquisition stage continuously samples the output of the Front-End, and temporarily stores the sampled data when a trigger condition is found. Only a small number of these data points are required for further study. Finally, the Control and Communication stage is in charge of the environment monitor-
Design and implementation of an embedded system for particles detectors
33rd International Cosmic Ray Conference, Rio de Janeiro 2013

microcontroller running an operating system and applications used for data transmission and module control.

2.2 Operating system
The Operating System used for the embedded system is Linux, with Busybox providing the user environment and system tools. This combination gives a familiar interface for most users, since the user interface and the programming environment provided by the embedded system is very similar to the one found in a common Linux workstation.

Early hardware initialization, and booting the operating system is handled by a boot loader (uBoot). This boot loader allows loading of operating system images stored in the router, allowing embedded system designs without external flash memory. Booting from LAN also eases the upgrade process, sharply reducing the risk of inutilizing the modules due to badly applied upgrades.

System configuration is done using an automatic configuration protocol (DHCP), with a central server storing the configuration for each module, and the router acting as a relay. The DHCP server always sends the same IP address to a given module, using its MAC address to identify each module. This identification also allows sending different software images to different modules, or using a limited amount of modules for testing.

The operating system runs Telnet and SSH daemons, allowing remote access and easy file transfer.

2.3 Applications
The embedded system uses custom applications to configure the module and to handle data requests (section 2.3.1), send the environment monitoring information (section 2.3.2), and send the background detection rate (section 2.3.3).

Figure 2 shows the communication channels used by these applications. All these applications are run by the Busybox init daemon in respawn mode, guaranteeing a prompt application restart in case of abnormal exit, and write status messages to the system log (syslog). All the programs use an error correcting protocol (TCP) for communications outside the station.

2.3.1 Data Client
This program configures the module and responds to data requests. On program startup, the module configuration is requested from the server which returns the data read from the configuration file for the specific module. After the configuration data is applied, the module enters into request mode. When a data request is received, the program sends the incoming event identifier to the acquisition stage, and forwards the data returned by the acquisition stage to the external event repository. The module can be remotely reconfigured at any time using a program within the external event repository.

2.3.2 Monitoring
This application periodically acquires the values of all the monitred environmental variables, and sends this information to the monitoring server. The acquisition and data transfer rates can be configured by the user.

2.3.3 Background Detection Monitoring
This application reads the background detection information from the acquisition stage (considered as the amount of positive samples over a fixed period of time, regardless of trigger status) and sends it to the central server. It can also adjust the module configuration in order to maintain a predefined background detection rate.

2.3.4 FPGA programming
FPGAs are volatile devices and need to be programmed every time they are switched on. A modified version of JRunner, originally written by Altera [1], is provided for the embedded system.

3 Embedded system implementation: AMIGA
3.1 Introduction
AMIGA (Auger Muons and Infill for the Ground Array) [2] is an enhancement of Pierre Auger Observatory [3] which aims to lower the energy detection of cosmic rays down to $\sim 5 \times 10^{17}$ eV and measure the muonic component of the extended air showers. This enhancement consists of 61 stations placed in a graded infield of 750 m triangular grids, covering an area of $23.5 \text{Km}^2$. Each station will consist of an Auger Surface Detector (SD), and an additional detection surface of $\sim 30 \text{m}^2$ provided by plastic scintillators buried $\sim 2.3 \text{m}$ underground. An engineering array of muon counters consisting of a 750 m regular hexagon of six stations and one in the center is being deployed in the Pierre Auger Observatory, in order to validate the counters. Five of these stations are going to have four modules, two with a detection surface of 10m$^2$ and two with a detection surface of 5m$^2$. The other two stations are going to have a detection surface of $60 \text{m}^2$ [4]. Currently, fourteen modules are fully operational. Six stations have a 10m$^2$ module, and one station has a twin consisting of eight modules, all of them fully operational since March 18th, 2013 [5].

Each module consists of 64 plastic scintillator strips, with an optical fiber inside each strip [6]. These fibers are connected to an optical connector matched to a 64 multi-anode photomultiplier tube (PM) from the Hamamatsu H8804 series.

The first muon counters deployed had an earlier electronics system [7], which could support only one module.
per station. Furthermore, additional lab testing and simulations showed the need to upgrade the acquisition system [8]. With this upgrade, an embedded system which follows the design described above and reuses many components from the earlier design was developed and installed. This system is described in the following sections.

### 3.2 Trigger source and Data Request

AMIGA modules were designed to use an external trigger in order to acquire event data during a possible particle shower. This trigger and the data request system are provided by the SD of the corresponding station. The SD trigger briefly works in the following way [9]: Each SD has a self-trigger (T1) with an average rate of $\sim 100$ Hz. When a T1 happens, a signal is sent to the acquisition electronics. Upon reception of this signal, the acquisition electronics stores the event data in memory with a GPS timestamp (GTS), used to identify the event. Once per second, the SD sends a list with the GTS that corresponds to possible events to the Central Data Acquisition System (CDAS). Upon reception of a data request (T3) from CDAS, the SD sends the event data corresponding to the GTS received in the request.

The SDs belonging to AMIGA were upgraded with the ability to output a signal each time a T1 is produced, and additional electronics called Trigger and Data Request (TDR). When a T1 trigger occurs, the SD sends a signal to the AMIGA modules containing a Local Timestamp (LTS), which is a 24 bit number generated by a free running counter. Taking into account the average T1 rate, each LTS value is guaranteed to be unique for $\sim 26.8$s, enough time to ensure that requested events won’t have data collision with newer events. After the T1 has a GTS associated, a signal containing the LTS and a reduced version of the GTS (48 bits total) is sent using a SPI protocol to the TDR, which stores this pair in a buffer capable of holding 2048 pairs. The TDR also has a sniffer that detects T3 requests aimed at the station. When a T3 is found, the TDR looks for the received GTS in the table. If the GTS is found, its corresponding LTS is broadcasted to all the modules via UDP. Otherwise, a “LTS not found” message is broadcasted to the modules. The TDR also provides access to the SD serial console via SSH or TELNET protocols.

### 3.3 Energy system

Due to the characteristics of the AMIGA site, a battery-backed solar power system was chosen to supply power to all the AMIGA electronics. The power system for each position of the UC is composed of: two 170 W solar panels, two 12 V 150 Ah deep-discharge batteries connected in series (24V), a 40A solar regulator, and a power distributor which delivers power to each module using passive power over Ethernet. The Twin position has two power systems as described above, each handling four modules and either the radio or the data-request sniffer. The power budget is $\sim 7.5$ W for each AMIGA module, $\sim 4.8$ W for the router/switch (for twin positions) and $\sim 3$ W for the TDR.

### 3.4 Communications

Communications are provided by a Mikrotik RB493 router, which includes an Ethernet switch used to connect the modules and the TDR, and a 802.11n Wi-Fi radio [10].

### 3.5 Module electronics

The AMIGA Front-End conditions and digitalizes the signals coming from the PM. It consists of a Mother Board, providing the power supply (Hamamatsu C4900-1) and the socket for the multi-channel PM, and distributing the PM signals to eight Daughter Boards, handling eight PM channels each. Once inside the Daughter Board, each signal is amplified using a current feedback transimpedance amplifier and digitalized by a comparator with a configurable voltage threshold [11]. The comparator outputs a digital ‘0’ if the signal is above the configured voltage threshold, or a digital ‘1’ otherwise. The threshold is meant to be configured in order to filter noise signals generated by the PM.

The Acquisition stage is done by a Digital Board using a FPGA, which continuously samples the output from the Front-End at 320 Msps. The Digital Board also drives the DACs used to set the threshold level of the discriminators used in the Daughter Boards.

Control and Communication services are provided by the Control Board, which is the only new hardware part of the new module design. This board uses a microcontroller running an operating system in order to provide the necessary services. The services provided by the microcontroller could be provided by a soft-core running on the FPGA which would have lowered the power consumption of each module, but time constraints and a requirement to integrate with existing hardware made the external microcontroller a better option.

![Figure 3: Block diagram of the AMIGA module electronics.](image)

### 3.6 Embedded system - Hardware

#### 3.6.1 FPGA

The current FPGA implementation is divided into six main blocks, as shown in figure [3]. The EMC block drives 16 Mbytes of external SRAM memory, used to temporarily store event data. This data has a fixed size (1024 samples for each PM channel, equivalent to 3.2\mu s per event), and is stored contiguously. The trigger block receives the T1 signal and notifies the data block. This block has a lookup table with 2048 entries, accessed with a LTS of an event and returning the memory position on which the data for the event with said LTS begins. The data acquisition block samples each channel from the Front End at 320 Msps, using two circular buffers able to store 1024 samples per channel. When a trigger arrives, this block waits until the current buffer is filled, switches the buffer used to store samples, and saves the buffer with the event data on external SRAM. The DAC programming block configures the threshold values of each comparator individually with the values received from the
Control Board. The microcontroller interface block handles communications with the microcontroller. This is implemented by emulating a Flash memory. Finally, the coordinator block manages the rest of the blocks.

### 3.6.2 Microcontroller

The microcontroller used is the NXP LPC2468, which has an ARM7TDMI architecture. This was selected due to its low power consumption compared to other alternatives, and the wealth of peripherals included on chip. The peripherals used within the module are:

- EMC (External Memory Controller): drives the external SDRAM (32 MB, 133MHz) and handles communications with the FPGA of the Digital Board.
- Ethernet: provides communications for the running applications and remote access.
- SPI: handles the DAC used to configure the PM High Voltage (HV) source.
- ADCs: employed in environment monitoring.
- UART: enables communications with the microcontroller though a terminal for testing purposes.
- Watchdog: Monitors abnormal system statuses and reboots the system when it reaches these status.
- GPIO: Four GPIO ports implement the JTAG interface used to program the FPGA of the Digital Board.

The FPGA interacts with the microcontroller as a 64 Kbyte external flash memory with a 16 bit data bus. Its address space is divided into configuration registers, signal rate counters, and event data transfer.

### 3.6.3 Monitoring circuits

The Control Board measures the output voltages of the power supplies within the Mother Board, the voltages used to power the FPGA, the PM HV input (up to 1000 V), the voltages of the input and control lines of the PM HV power supply, and the temperature inside the module. Two multiplexers were installed on two ADCs on the Control Board in order to expand the amount of environmental variables that can be sensed.

### 3.6.4 PM HV control

The Control Board uses an external 12-bit DAC (AD5660) to control the HV power supply output. The output of the DAC is fed to an amplifier which increases its output voltage and provides ground isolation.

### 3.7 Embedded system - Software

The microcontroller runs a modified version of Linux 2.6.24, with support for its peripherals.

Module configuration is done in two stages. During the Linux boot procedure, the DHCP server sends the module hostname (unique for each module in the array), SD ID (LSID), and Module ID (MID) within the station. Afterwards, the application in charge of data acquisition receives and sets the threshold values for the comparators and the PM HV. Acquired data is sent with a header containing the trigger information sent by the TDR, plus an array of `<index, value>` pairs, where index is a relative time position within the event and value holds the acquired data for the 64 PM channels at that moment. This structure allows the system to send only the time bins with interesting data (those containing at least one PM channel with a '0'), which is usually a small amount of time bins.

A web interface is being developed with the aim of providing easy access to current and historical monitoring information, and hardware characteristics of the AMIGA modules.

### 4 Verification

The verification of the module electronics consisted of three tests in a laboratory setting, one for the environment monitoring, one for background muon and data acquisition, and another one for data acquisition. The monitoring values were contrasted with data measured by a voltmeter. The background muon monitoring and the data acquisition were verified by injecting a signal from an Arbitrary Waveform Generator (1.25 Gsps), similar to the output of a PM into the Front End at a known rate, and contrasting the information obtained from the electronics to the predefined rate. For the data acquisition test, the trigger was emulated by a fully controllable system, and data requests were generated using a special software running on a PC. The last test for data acquisition was done using a PM fed by a programmable light source [12] and the entire AMIGA electronics.

### 5 Conclusions

A complete detector electronics design was shown, incorporating, whenever possible, standard interchangeable components and decoupling the embedded software from the hardware used in the detector. An implementation of this design, commissioned and built by ITeDA, is currently being used in AMIGA. Laboratory and field testing of the system shows that it performs as expected.

### References