New Control System for the 1.5m and 0.9m Telescopes at Sierra Nevada Observatory

Luis P. Costillo, J. Luis Ramos, J. Miguel Ibáñez, Beatriz Aparicio, Miguel Herránz y Antonio J. García
Instituto de Astrofísica de Andalucía (IAA), Camino Bajo de Huétor 50, 18008 Granada, Spain

ABSTRACT

The Sierra Nevada Observatory (Granada, Spain) has a number of telescopes. Our study will focus on two Nasmyth telescopes with apertures of 1.5m and 0.9m and an equatorial mount.
The system currently installed to control these telescopes is a 1995 centralized VME module. However, given the problems which have arisen due to the number of wires and other complications, we have decided to change this control module.

We will control each telescope with a distributed control philosophy, using a serial linear communication bus between independent nodes, although all system capabilities are accessible from a central unit anywhere and at any time via internet. We have divided the tasks and have one node for alpha control, another for delta control, one for the dome, one for the focus and the central unit to interface with a pc.

The nodes for alpha, delta and the dome will be used by means of FPGA's in order to efficiently sample the encoders and the control algorithms, and to generate the output for the motors and the servo. The focus will have a microcontroller, and the system is easy to expand in the event of the inclusion of more nodes.

After having studied several fieldbus systems, we have opted for the CAN bus, because of its reliability and broadcasting possibilities. In this way, all the important information will be on the bus, and every node will be able to access the information at any time.

This document explains the new design made in the IAA for the new consoles of control whose basic characteristics are, the distributed control, the hardware simplify, the cable remove, the safety and maintenance improve and facilitating the observation improving the interface with the user, and finally to prepare the system for the remote observation.

Keywords: Telescope Control System, IAA, OSN, hardware design, software design, CAN Bus, Embedded, distributed control, servo-control.

1. INTRODUCTION

The Institute of Astrophysics of Andalusia (IAA), is a dependent centre of the Consejo Superior de Investigaciones Científicas (CSIC) in Spain. The IAA is placed in Granada (Spain), and owns an observatory in the Sierra Nevada: the Sierra Nevada Observatory (OSN). The OSN is located at the Loma de Dilar, at 2800 m above the sea level, and overlooking the Sierra Nevada Ski Resort.

This observatory contains two Nasmyth telescopes with apertures of 1.5m and 0.9m and an equatorial mount. Both telescopes are identical except by the size and the aperture. So, the control systems are twin to facilitate the design, construction and maintenance.

The control system for these telescopes was designed and made in the IAA according to the state of the art in that moment:1995. It is a centralized VME module for each telescope. The module uses a CPU (68000 microprocessor), and some of the E/S cards were designed specifically by the IAA. (Fig. 1). So, we can control the dome movement and positioning, the alpha and delta movement (point and tracking). We control also the secondary and third mirrors

Advanced Software and Control for Astronomy, edited by Hilton Lewis, Alan Bridger,
Proceedings of SPIE Vol. 6274, 62741I, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.671308

(movement and position), and the primary mirror. There are controls for the telescope safety, and all the relating and necessary for the observation.
We use a high level program to handle the observation. The program connects with the control system and the instruments, making the data acquisition and providing facilities to the observer.

At present we are having some problems with this system. In one hand, due to the hard environmental conditions, there are deteriorated connectors and wires. We have also obsolete components, and the large amount of wires on the telescopes makes the movement difficult.

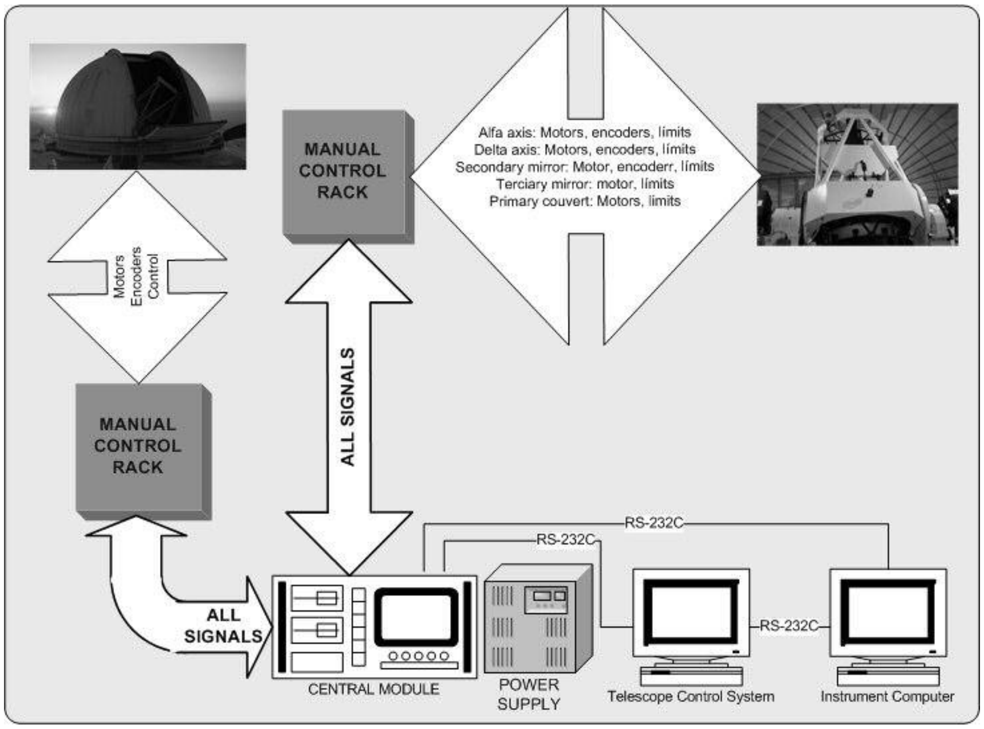
MANUAL CONTROL RACK
Alfa axis: Motors, encoders, limits
Delta axis: Motors, encoders, limits
Secondary mirror: Motor, encoderr, limits
Terciary mirror: motor, limits
Primary couvert: Motors, limits
Motors
Encoders
Control
ALL SIGNALS
MANUAL CONTROL RACK
RS-232C
RS-232C
RS-232C
ALL SIGNALS
CENTRAL MODULE
POWER SUPPLY
Telescope Control System
Instrument Computer

Figure 1: Current control system

On the other hand, nowadays is remarkable the advance in electronics components, electro mechanics and computer (hardware and software). For that reason we want to improve the features, capacities, securities, the maintenance and mainly the facilities of observation in the console.

We have designed new control systems for the 150 and 90cm telescopes. Both systems will be twin, so we will talk about a single control system.

The telescope is now operative, and working, and only stops one week per moth to do technical repairing or improvements. Therefore, it is very important to design a modular system that can be mounted without disturbing the normal operation of the telescope.

We used a distributed control system, with several independent control modules with specific functions. This kind of control provides an easier detection and repair of mistakes. It is also possible the modular implantation of the system, and to add more signals to the control.

In order to avoid the excess of wires, we use a serial bus, and due to the environmental conditions, we have chosen an industrial bus, called CAN bus.

In section 2 we present the general description of the total system, with the communication bus. In section 3 we detail each of the submodules, and in section 4 we detail the software used.

2. GENERAL DESCRIPTION

We designed a system with several modules with specific functions, located in the same place of its function to avoid the excessive number of wires. Each module has to control some of the elements, and its work is supervised by another one, called Central Module.

The Central Module (CM) is located outside the dome, in a room next to the dome, called "console". This module supervises all the processes.

2.1. Elements to control

In this section, we describe the elements we should control.

- Axis Alpha: We must control the movements on this axis, to point and do the tracking movements correctly. It can do absolute, relative and incremental movements with at least three speeds. We have also to control the angle safety (to avoid dangerous positions) and the counterweights. There is a timer receiver GPS to operate in astronomical coordinate system.
- Axis Delta: We must move the Delta Axis with absolute, relative and incremental positions, with three different speeds, and with 1arcseg of precision
- The Dome: We must control the Azimuth+ and Azimuth- movements.
- Secondary mirror: We must control reading of position and limits, and focus movements. Currently we are installing a hexapod to support the secondary mirror, and the control for this hexapod has also to be handled.
- The Third mirror: It must switch between the two Nasmyth focus, (east and west).
- Primary covers: We must open and close the covers of the primary mirror, and read their position.
- Computer instrument: Each instrument on the telescope must have a control computer. This computer is connected to the global control system, and handles the instrument with standard commands, accord to the telescope.

All these controls are made by several subsystems, called "modules". In Figure 2, we can see the different modules and the distribution of the control in all of them. As we can see, all the modules are supervised by the central module. Dome and alpha and delta axis can be controlled manually or automatically by the total system. The "others" control is referred to the mirrors, and focus.

The central module is connected by Ethernet to an instrument computer, and can also be handled remotely.
The user can operate in three different modes: manual, automatic, and remote. In manual mode, the user handles keyboards located in telescope and the dome, or at the console room. In the automatic mode, the instrument computer takes the control of the telescope, working together with the instrument. In remote mode, a remote computer controls the system via Internet.

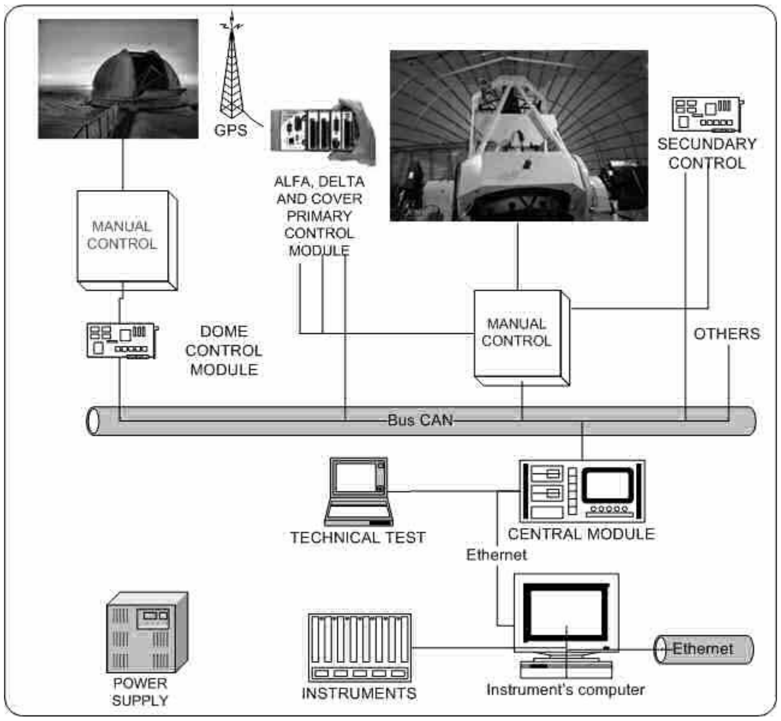

Figure 2: New distributed system scheme

2.2. Communication

To communicate all the modules and avoid the excessive wiring, we need a serial bus with linear configuration. It should be an industrial bus, due to the environmental conditions. The bus has to transmit approximately 100bits in 6ms, and must be compatible with the actual VME system, to allow the observation while installing the system. The adaptors and elements have to be easy to put, prove, and quit. The bus has also to admit the addition of new devices to provide changes on the system, and should be multimaster, providing each device the capacity of transmitting information in a moment.

After a comparison between several industrial buses[1],[2],[3], we have to choose between RS485[4] and CAN[5],[6],[7]. With CAN[8] is easy to add and subtract devices of the net, and there are standardized transmission protocols. So, we choose the CAN bus as the system bus.

3. CONTROL MODULES

In Figure 2 we can see the different control modules. In this section, we describe them in detail.

We designed a system with all the functions made by four modules, located on the telescope or in the dome. These are the Dome Control (DM), the Secondary Control (SC) the Axes Alpha Control (AC), the axes Delta Control (DC), and the focus. There will be also an additional module outside the dome, the Central Module (CM).

3.1. Dome Control

The dome does not have power slip rings, then the opening of main and dropout shutters must be done before the observation, manually using a keyboard located next to the shutter.

For the Azimuth movement, the dome has two continuous current motors, and the reading of the position is made by an absolute 12 bits encoder. It is also possible to move manually the dome using a keyboard located in the dome.

The dome control module can do several things:
1. Operate in tracking mode. The module receives the alt-azimuthal coordinates of the telescope, and using the information from the dome encoder, moves the dome to the correct position.
2. Operate in absolute mode. Optionally, the user can send the dome to a particular position
3. Operate in manual mode without the keyboard. From the Central Module, the user can rotate the dome.

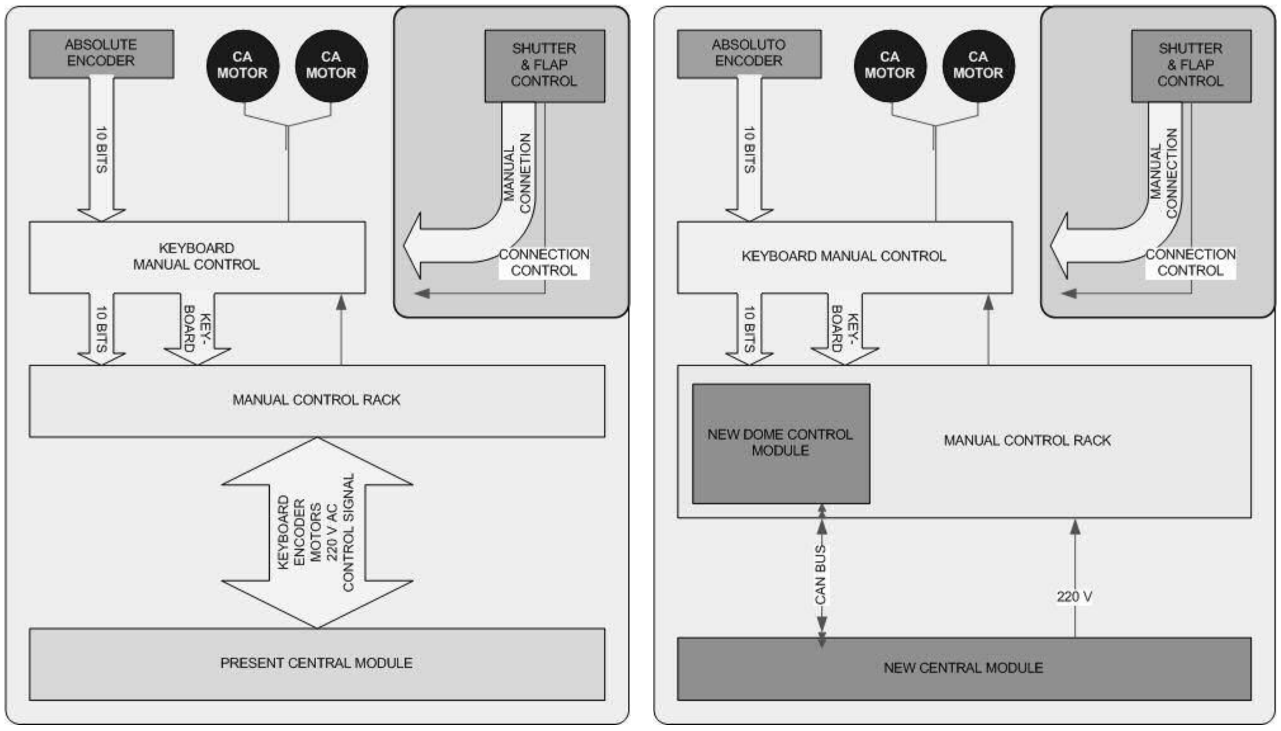

Figure 3: Current and future scheme of the dome control module

In addition, the program can make tests to prevent failures in the system, as the current sensing of the motors, that will allow to detect when one of the motors is not working properly by overheat, or when the dome is frozen and simply does not move.

This module uses two PC-104 format cards. One of them contains a PIC18F458, and a transceiver for CAN bus. The micro can be programmed in high level languages as C. The other card is for specific use, with some adaptation circuits as optocoupler or relays.

The installation and testing of the module is thought to easily return to the previous system. It is enough to connect the new system to a connector on the rack in the dome, and to change, in the console room, the control computer (Figure 3).

3.2. Secondary control

We use a stepper motor to move the secondary mirror, providing 12 microns by steps. That is, 120 microns on the focal plane (Fig 4). There are limit switches to sense the final positions of the mirror, and to stop the motor when they are reached. We read the position by a LVDT, which provides an output from -10V to +10V. This output is leading to a 12 bits A/D converter, in a card located next to the secondary mirror, on the top of the telescope. Actually, we have a twelve conductor cable to the control room. The new controller reduces wiring, replacing the existing one by the CAN bus and 220 AC. Software system is implemented for the prevention and trouble shooting, mainly the very low temperatures for the stepper.

For the accomplishment of this controller we use the modular system mentioned in the previous section, with identical cards for the CPU with PIC 18F548. The specific card, also in format PC-104, has the A/D converter, the transceiver for the stepper motor, optocoupler for reading the limit switch, and other needed circuits. The firmware for the PIC is made in assembly language and C, as the one of the dome.

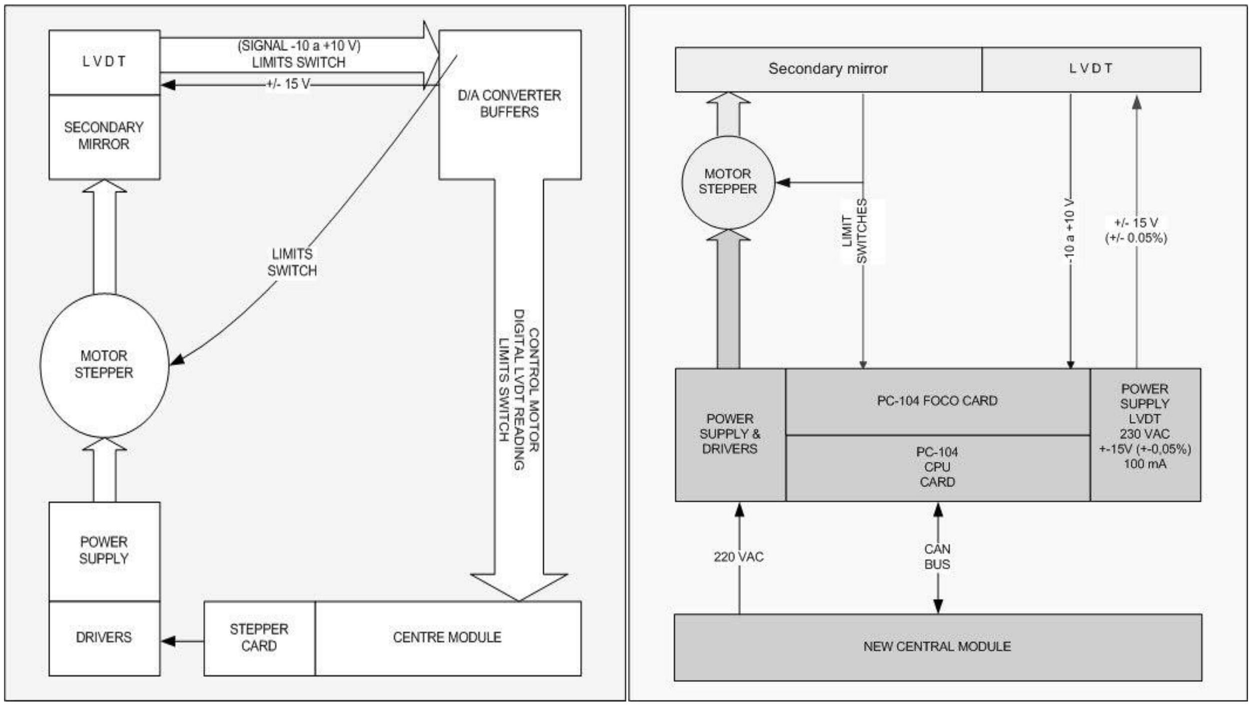

Figure 4: Current and future scheme for the secondary control

The installation and testing of the new system should not interrupt the observation, so we will do it keeping the old system until the new one is enough tested.

When installing hexapod, the control system will be the same, exchanging all the controls from one to the other.

3.3. Axes module (AM)

Both, alpha and delta axes are similar in the configuration hardware. Each one has two motors (motor and contramotor) and an emergency brake. For the first PID control cycle, there is a relative encoder, coupled directly to the motor. We read the axis position with a 10 bits absolute encoder, that provides the reading of the arcseg (21arcmin precision), whereas an Inductosyn relative encoder provides arcseg precision. We actually use a HCTL controller[9] to handle the movement. We obtain the sidereal movement in alpha axis with a quartz with the suitable frequency. Currently, the console module makes all the axes control, thus there is a big amount of wires from the sensors, and to the feedings of the motors (Figure 6). It is possible to manually move both axes of the telescope, using the keys in the manual control rack located in the dome, next to the telescope.

The new design does a lot of changes. The motors will be replaced by newer ones. The codification system will also be replaced by a 24 bits absolute encoder. The power supplies will be changed by better ones, with smaller consumption and size. The manual movement system will be changed also.

This is probably the most complex module of the total system, because of all the variables to consider, and the precision needed. The first idea was to locate the two axes (control of Alpha and Delta) in a single module that should be equipped with the suitable power. We want to use products COTS (Commercial Off The Shelf) in this module, to facilitate their design and implementation and mainly to diminish the time of development.

Therefore we have thought about the possibility of using compact or embedded computer. Embedded can offer far ruggedness, reliability, and low power consumption (25W). We will use the Reconfigurable Compact Embedded System (cRIO of NI) (Figure 5). It has a 200 MHz industrial Pentium-class processor for deterministic and reliable real-Time applications and to 3M gates FPGA for the control and handling of I/Os.

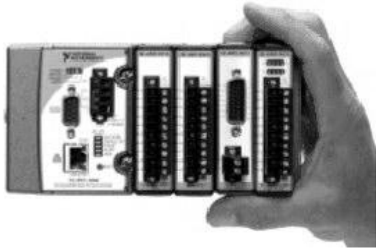

Figure 5: RCIcO is of small size and little consumption

This system gives the possibility of programming the FPGA in a high-level language (LabView). This will remarkably reduce the time of development and the number of people implied in the project. The verification time will be reduced also, which is almost the most important thing, because in the verification phase the telescope will not be able to observe.

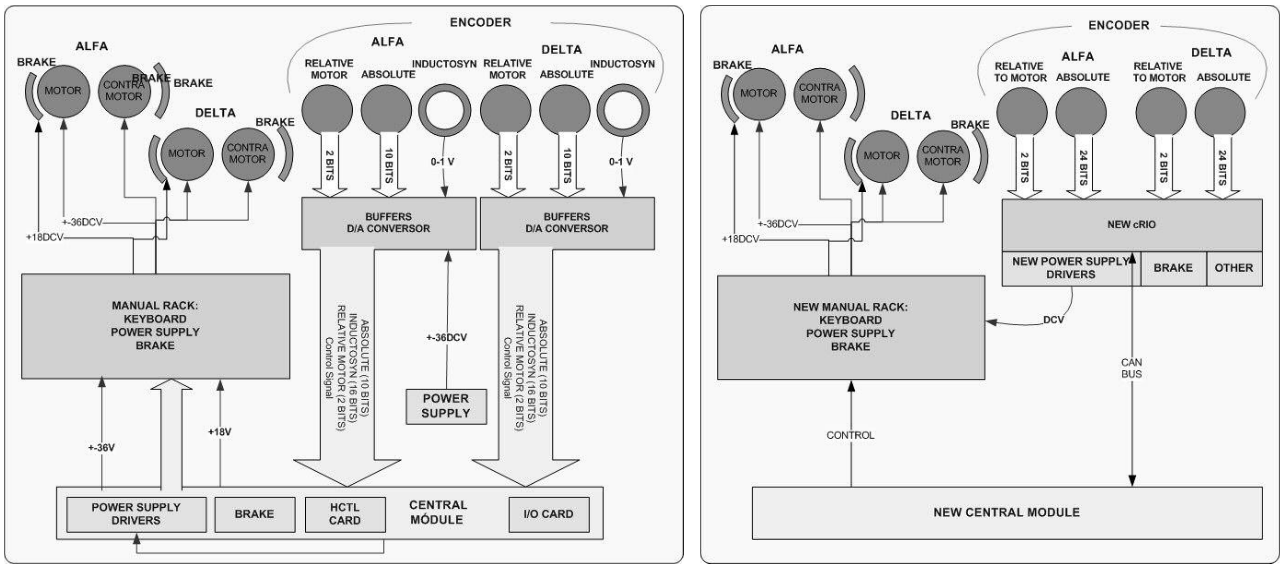

Figure 6: Scheme of the old and the new configuration.

Finally, using this system we easily implement the systems of prevention and trouble shooting and it will be faster and easier to introduce subsequent modifications and improvements. We will locate this module upon the telescope, which is not difficult by its small size and low consumption. Thus the number of wires is considerably reduced.

3.4. The other controls

It is necessary also to control the movement of the third mirror, the covers of the primary mirror, and some security limits to avoid dangerous movements. We keep all the automatism, and the Axis module will do the control. This reduces the amount of wires, because the axis module is physically located very near of the security limits, on the telescope. Now there are switches and mercury that prevent crashing of the telescope with elements of the dome, the mirror falling, the oil lacking in the traction system, etc. These limits directly work on motors supply and brakes. The central module can read them, and its reading is concentrated in a box located on the telescope. The new version improves the security increasing the number of limits and test systems for the detection and forecast of mistakes (Fig 7).

The covers of the primary mirror move with individual DC actuators. Now they are manually handled from a hand control next to the telescope and from the central module. The new system allows open/close the covers, in manual mode, automatic and remote.

The third mirror has to move to focus one of the two Nasmyth, East and West. Therefore it has only two positions, reading by switches. Now we can only do these movements from the central module using an AC motor. As before, the new control system handles the third mirror from the Axis module; we can get access to the data through the CAN bus from the CM or any other module.

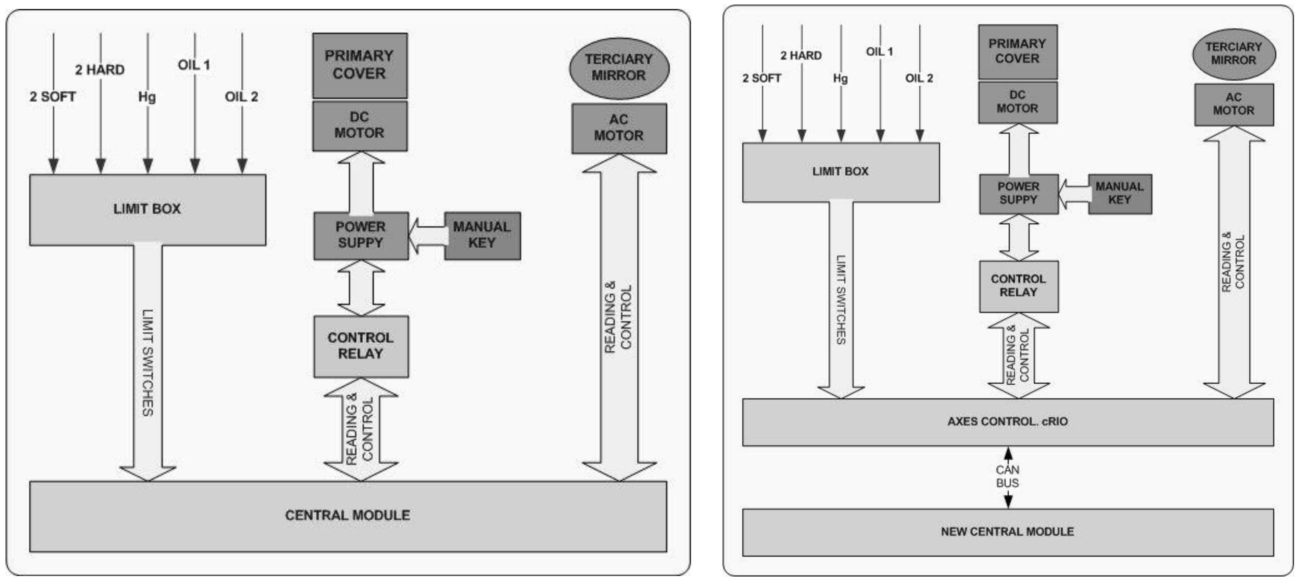

Figure 7: New and old limit control scheme

3.5. The central module

The central module is the main subsystem of the telescope control. The graphical user interface (GUI) is executed in this module. We call this GUI Telescope Control System (TCS). The central module has a private interface to interact with the rest of the control modules. It has also a public interface to interact with the rest of the systems in the observatory, such as instruments, weather station, data storage system, etc. To get scalability, reliability and availability we need to add to this module some support to the observation, supervision, monitoring and registry, remote accessibility, interface with the instruments, etc. The system implementation must be progressively done, to not disturb the observation.

It is for that reason that the two systems (old and new) will coexist temporarily. But the installation of the Alpha and Delta axes controllers, with motors and encoders, must be done without return, for which reason it will require a certain time without using the telescope.

4. CENTRAL MODULE SOFTWARE

The Central Module (CM) software will be the software package responsible for:

- Control and monitoring the telescope
- Manage interaction between CM processes and execute tasks requested
- Provide a local console GUI for user operations
- Bus CAN interface to other telescope control subsystems
- TCP public interface to other observatory subsystems
- Autoguiding

These functionalities each one within a separate process, will be done by the Central Module (CM). The Figure 8 shows this.

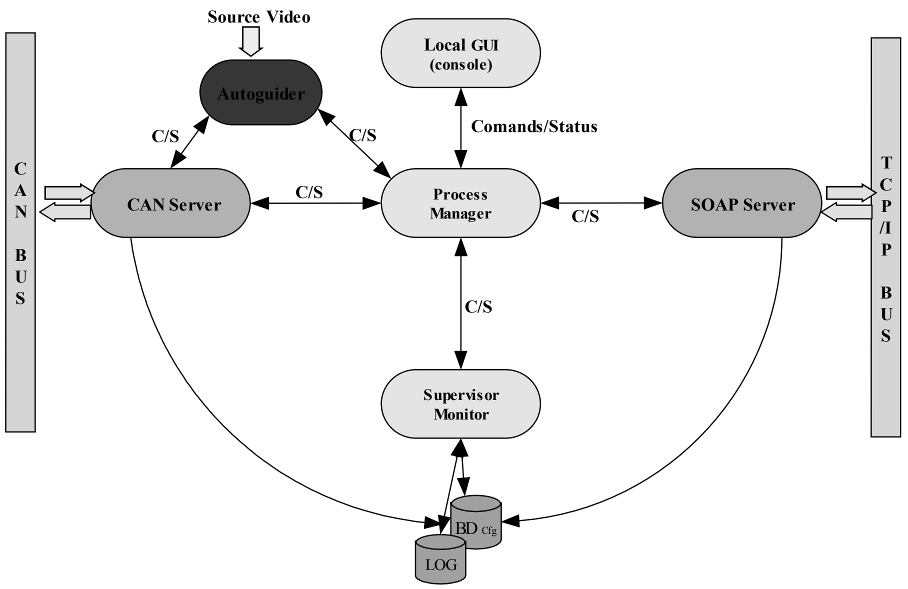

Figure 8: Central Module proceses

The development methodology used in this software will be object oriented, using as much as possible the open source software[10], Other projects already have employed this philosophy.

4.1. Bus CAN interface

As mentioned before, the control of the whole system is based on a single Linux PC and a large number of microcontroller modules that perform basic operations (like "position dome" or "focus telescope") and a main module in charge of AR and Dec axis (Axes Module). These modules and the control PC shall be interconnected - in a distributed system approach - by the popular and very robust CAN (Control Area Network) field bus. Thus they form the Telescope Control Network (TCN). The protocol layer implemented over the data link layer [which conforms to CAN 2.0A and/or 2.0B] and the physical layer [which specified in the ISO 11898 standard], is a proprietary and simple protocol. We discard to use any standard CAN bus high layer protocol, like CANOpen or DeviceNet, because we have a small number of nodes and a simple CAN bus architecture. The data link layer and the physical layer are implemented in hardware.

So, an important task that must be carried out by the CM is the bus CAN handling that supports our own CAN bus application layer. It will implement a process with the following functional requirements:

- Read all messages from the field bus
- Parse and interpret that messages
- Dispatch the message and/or pass it out to the appropriate process
- Accept output requests from other local process for writing commands into the CAN bus
- Write well-formed messages to the bus CAN
- Implementation of monitoring and debugging functions via CAN bus

Because of these critical and important functions, this process must be executed with the highest priority possible in the system.

4.2. SOAP interface

Besides the field bus communication mechanism between telescope control subsystems, the CM must have a public interface which allows communication between the external systems other than these, like remote GUI or instrument control software. This interface must satisfy the following requirements:

- fast and standard application to application communication mechanism
- a well-defined interface used by many observatory subsystems (even remote observing)
- flexible and secure protocol
- platform independent

We have chosen SOAP (Simple Object Access Protocol) based transport scheme as the base protocol for the telescope public interface (TPI). Since this protocol builds up on the XML format, in our implementation SOAP will use XML on HTTP to enable RPC calls to different systems. It is totally independent of operating systems and programming languages. So, we could use C, C++, Perl and Python with SOAP packages on Linux. In this way, remote objects can be accessed as if they were local to the application. The intrinsic of the actual remote calls are transparent to the application develop. A large number of SOAP toolkits are available for different programming languages and platforms. Toolkits range from simple APIs to elaborate SDKs for SOAP client and server development. We foresee to use the core package for the operations gSOAP. Some features attached to use gSOAP are:

- **Unique SOAP-to-C/C++ binding**: gSOAP provides an automated mapping of native C/C++ and user-defined application data types to XML data types. Data is serialized in XML "in-situ" by the pre-compiled encoding routines generated by the gSOAP compiler.
- **SOAP 1.2 compliant**: SOAP 1.1/1.2 support is fully automated (use soapcpp2 compiler options -1 and -2).
- **Platform independent**: Windows, Unix, Linux, Pocket PC, Mac OS X, TRU64, VxWorks, etc.
- **Integrated WSDL generator:** for convenient Web Service publishing.
- **WSDL parser**: for automated Web service application development.
- **Efficiency**: gSOAP has a very low memory overhead and very low SOAP RPC latencies which are important properties for **embedded** and **real-time systems**.
- **Small memory footprint**: Web service and client executables can be as small as 90K on Linux with a 150K total memory footprint.
- **Multi-threaded stand-alone SOAP servers**: automatic C and/or C++ source code generation for efficient SOAP Web services (gSOAP 2.0 and higher). Load stress tests indicated that gSOAP has the best performance and scalability compared to other Web services toolkits and implementations.
- **Security and Encryption**: supports HTTPS/SSL.
- **Flexible transport layer**: gSOAP's I/O operations callback mechanism provides customizable transport layers.
- **Compression**: supports compression (HTTP and plain SOAP/XML messages).
- **Logging**: provides request/response message logging.
- **Save and retrieve data in XML**: gSOAP generates XML (de)serializers for application data types which allow application data to be saved and retrieved as XML from e.g. files.

In practical terms SOAP works like this:

- The SOAP library on the client side is constructing an XML envelope for the call parameters.
- The standardized XML message is sent on the wire to the server listening at a specific TCP/IP port on a computer.
- The server does its task accordingly, and sends back a message to the client.

Therefore, the CM will have a process working as a SOAP server that will accept the request from authorized clients and send back the response to them.

4.3. Autoguider

The absolute RMS tracking error must be as low as possible in order to guarantee the image quality during the observations. So an important component of the CM will be the telescope autoguider. Right in the beginning we could actually start without a guider, given that the telescope tracks very well but for long integration times we will need a good autoguider mechanism that corrects the tracking errors.

We think to implement a process into the CM that periodically acquires images from the Guider TV, compute the tracking error and sends the required offsets to the telescope (axis module). Previously, the Guider TV needs to be pointed to a suitable guide start. To do this, the autoguider will query the Guide Star Catalog (GSC) to find out a guide star near the field of view of the source, transform from star equatorial coordinates to XY guideprobe coordinates and point it to them.

As result autoguider process will run a servo system in a loop with a high frequency and high priority. Because of these requirements, it is possible we designate a separate and single linux box for autoguiding purposes.

4.4. Process Manager

The Process Manager executes all the requests received by each interface of the MC. All the principal components of the MC make it function through it. This component is the agent to move the telescope, to focus, to move the dome, etc. When the Process Manager receives an instruction of the SOAP Server, Can Server, Local GUI or Autoguider, the first step that it must to do is to ask the Supervisor Monitor about the availability to do this instruction. When the Supervisor respond the Process Manager it's the responsible for making the opportune calls to the hardware through the CAN Server.

This component allow us to centralize the operation of the telescope, there are a lot of different components that can send instructions to the telescope to do movements, like the autoguider, local GUI or remote GUIs. The Process Manager must decide who can move the telescope and in which moment. This centralization provides us a safe telescope control.

4.5. Supervisor Monitor

The Supervisor Monitor component does the supervision tasks. All the telescope subsystems are monitorized by it (Alfa, Dec, dome, focus, mirrors, weather, autoguider). It asks the status to each subsystem, logging the most important events and warning the user. This component allows the user to have the most important telescope parameters. Also this component logs all the most relevant information about the operation status of the telescope, dome, focus, autoguider, etc., in a log data base. Very usually for the debugging of the possible problems that could take place.

When a special event, related with the supervision tasks, occurs, this component must generate the corresponding warning to the user.

This component is used by the Process Manager to check the availability to execute an instruction. It has the information about the status, the different clients, the priority, the telescope position, and all the related information that the Process Manager needs to check before executing an instruction.

4.6. The Graphical User Interface (GUI)

This is a control console, it represents the most important parameters related to the telescope status and permits the basic following actions: (a) Move telescope to Zenith; (b) Move telescope to Polar; (c) Dome control; (d) Change tertiary mirror East/West; (e) Telescope pointing; (f) Focus movement.

The information provided by this GUI is all the information that the telescope operator needs in the observations, we thought about a small touch screen, in which all this data will be displayed, with the minimal functionality to see the different tabs, like telescope status, dome status, focus status, engines status, log files, weather information, etc.

All the warnings and alarms are also displayed in this GUI, There are different levels of alarms, depending of the risk.

This GUI is the user console, in addition to this, we will develop a software GUI which implements all functionality, developed in the graphical programming language LabVIEW[11].

4.7. Implementation considerations

The CM itself will be written entirely in C++. We also use ACE[12] (ADAPTIVE Communication Environment), a freely available, open-source object-oriented (OO) framework that implements many core patterns for concurrent communication software. ACE provides a rich set of reusable C++ wrapper facades and framework components that perform common communication software tasks across a range of OS platforms. The communication software tasks provided by ACE include event demultiplexing and event handler dispatching, signal handling, service initialization, interprocess communication, shared memory management, message routing, dynamic (re)configuration of distributed services, concurrent execution and synchronization.

All C/C++ code will be written to support documentation using javadoc style. This means that all documentation will be available via HTML and will be fully indexed and consistent with source code.

Naturally, the design will be object-oriented and the implementation will emphasize code reuse.

ACKNOWLEDGMENTS

The authors would like to Miguel Abril for his current work in the implementation of this project.

REFERENCES

1. Klaus J., Annerose B. *Application of industrial CAN bus technology for LEO-Satellites*.
2. Mayne Grau, J. *Sistemas de comunicaciones industriales*. AVNET electronics marketing.
3. Kaschel, H., Pinto E., *Análisis del estado del arte de los buses de campo aplicados al control de procesos industriales*.
4. Davis, L. Leroy's *Engineering web site www.interfacebus.com*, 2002
5. Ruben,A. ,Köster, A., Plein, M., W-IE-NE-R Plein & Baus GmbH, *CAN-BUS based VME Crate Remote Control*
6. Esd gmbh Hannover. *Controller Area Network CAN, a Serial bus System Not just for vehicles*
7. Bosch: *CAN Specification V2.0*
8. CiA Draft Standard 102 Version 2.0 CAN Physical Layer for Industrial Applications, April 1994
9. Elizo, M.A., Gonzalez, P. *Controladores de motores integrados HCTL-1000 y LM628/629. Descripción y comparación*. Mundo Electrónico, Junio 1991
10. Graybeal, J., Brock, D., & Papke, B. *The Use of Open Source Software for SOFIA's Airborne Data System*. 2000, Proc. SPIE Vol. 4009-17
11. Ashe, M., Schumacher, G. *SOAR Telescope Control System: A Rapid Prototype and Development in LabVIEW*. 2000 Proc. SPIE Vol. 4009, Invited paper.
12. See the ACE source web page http://www.cs.wustl.edu/~schmidt/ACE.html