TELICS—A Telescope Instrument Control System for Small/Medium Sized Astronomical Observatories

Author(s): Mudit K. Srivastava, A. N. Ramaprakash, Mahesh P. Burse, Pravin A. Chordia, Kalpesh S. Chillal, Vilas B. Mestry, Hillol K. Das, and Abhay A. Kohok

Reviewed work(s):
Source: *Publications of the Astronomical Society of the Pacific,* Vol. 121, No. 884 (October 2009), pp. 1112-1119
Published by: The University of Chicago Press on behalf of the Astronomical Society of the Pacific
Stable URL: http://www.jstor.org/stable/10.1086/644648

PUBLICATIONS OF THE ASTRONOMICAL SOCIETY OF THE PACIFIC, **121**:1112–1119, 2009 October

TELICS—A Telescope Instrument Control System for Small/Medium Sized Astronomical Observatories

MUDIT K. SRIVASTAVA, A. N. RAMAPRAKASH, MAHESH P. BURSE, PRAVIN A. CHORDIA, KALPESH S. CHILLAL, VILAS B. MESTRY, HILLOL K. DAS, AND ABHAY A. KOHOK

Inter-University Centre for Astronomy and Astrophysics (IUCAA), Pune, India; mudit@iucaa.ernet.in, anr@iucaa.ernet.in

Received 2009 April 20; accepted 2009 August 17; published 2009 September 17

ABSTRACT. For any modern astronomical observatory, it is essential to have an efficient interface between the telescope and its back-end instruments. However, for small and medium-sized observatories, this requirement is often limited by tight financial constraints. Therefore a simple yet versatile and low-cost control system is required for such observatories to minimize cost and effort. Here we report the development of a modern, multipurpose instrument control system TELICS (Telescope Instrument Control System) to integrate the controls of various instruments and devices mounted on the telescope. TELICS consists of an embedded hardware unit known as a common control unit (CCU) in combination with Linux-based data acquisition and user interface. The hardware of the CCU is built around the ATmega 128 microcontroller (Atmel Corp.) and is designed with a backplane, master-slave architecture. A Qt-based graphical user interface (GUI) has been developed and the back-end application software is based on C/C++. TELICS provides feedback mechanisms that give the operator good visibility and a quick-look display of the status and modes of instruments as well as data. TELICS has been used for regular science observations since 2008 March on the 2 m, f/10 IUCAA Telescope located at Girawali in Pune, India.

Online material: color figures

1. INTRODUCTION

The emergence of large optical telescopes along with state of the art back-end instruments have greatly benefited the astronomical community through their powerful capabilities to see faint objects and deep into space. However, the relevance of small and medium-sized optical telescopes continues to grow, as the large facilities are few in number, and due to relatively good availability of telescope time on small telescopes. In order to minimize the time overheads during observations in the setup of various instruments and units (filters, calibration lamps, etc.) and in data acquisition, it is essential to include modern technologies in the conception of new instruments and in control and data acquisition systems. However, unlike large optical telescopes where focal plane instrumentation as well as control systems have reached a fairly high level of complexity and are more costly in nature, the control systems for small and medium-sized observatories are required to be simple.

Here we describe the design and development of such a control system, the TELICS—TELescope Instrument Control System, developed for the IUCAA Girawali Observatory (IGO) which houses a modern 2 m, f/10 telescope located at Girawali near Pune, India (Tandon 1998; Gupta et al. 2002). The TELICS system fulfills the requirements of any small and medium size (i.e., less than 4 m diameter telescope) observatory. The structure of TELICS can be thought of as consisting of two separate environments: (1) A Linux-based graphical user interface (GUI) and application software, and (2) a microcontroller-based embedded control system known as a common control unit (CCU), and its embedded software. These two parts of TELICS communicate with each other through a serial link or an Ethernet-based communication link for remote operation.

In this article, we present the various approaches that have been taken at different levels in the implementation of TELICS. In particular the embedded CCU, application software, and graphical user interface (GUI) are discussed here. Section 2 provides the design and implementation plan of TELICS. The basic design requirements and alternatives are discussed here. Section 3 is the description of the CCU. Its architecture and various software issues (interprocessor communication, protocols, etc.) are described here. Section 4 consists of the description of the GUI and Application software. Section 5 provides a summary and discusses the performance of TELICS.

2. TELICS REQUIREMENTS, DESIGN, AND IMPLEMENTATION

2.1. TELICS: Scientific and Technical Objectives

The basic technical requirements of TELICS originated from the idea to integrate high-level controls of various instruments,

detectors, and data acquisition in a single framework to reduce complexity and to provide a single interface to the user. For example, the current instruments on the IUCAA Telescope include the IUCAA Faint Object Spectrometer and Camera (IFOSC) with a polarimetric mode (Gupta et al. 2002; Ramaprakash 2002), and a Princeton Instrument CCD Imaging System (PI-CCD). In addition, a Near-Infrared PICNIC Imager (NIPI, an infrared imaging system) (Ramaprakash et al. 2003) is due to be commissioned on the telescope. Further, there are some support systems, e.g., a calibration unit (consisting of various lamps which are used for spectral calibration purpose) and a power supply unit to control the basic power requirement of these systems. Many of these systems have movable parts (that are controlled through motor drives), switching circuitry, and feedback arrangements being used in various ways for astronomical observations. These processes were required to be integrated in such a way so that the end user can make the observations through a GUI without knowing the system in depth. Also to utilize the full potential of the telescope, a variety of other instruments are being planned (e.g., an integral field unit, an imaging Fabry-Pérot spectrometer, etc.), spanning the optical and near-infrared regime of the spectrum. Hence the issue of incorporating the controls of any other external instruments within the existing control system and GUI has been given careful thought. Further, the requirement to provide automation of the setups (e.g., in setting up various modes of the instruments, in the calibration process, etc.) during the observations was critical, in order to minimize the operations overheads. So the option to define and execute user-defined macros was also given to TELICS. Additionally, the requirement that TELICS be seamlessly integrated with the telescope control system and any auxiliary control system (e.g., environment monitoring) has also been considered. Therefore TELICS is designed to:

1. Control all the movable parts of the instruments and support system at the telescope,
2. Operate all the switches and shutters,
3. Take feedback from various sensors mounted on the telescope, and
4. Talk to the control system of any other external instrument/system.

Of these requirements, the last is the most critical, as it provides the means to add new instruments in the existing setup. In the past, control of various systems on the IUCAA telescope were implemented by six different stepper motors, while IFOSC has its own controller to drive various motors in it. Therefore the hardware requirements for the TELICS included inbuilt circuitry to drive all the motors, shutter control drivers for all the shutters on the telescope, switching relays to control various power supply lines, and the ability to gather the status of a number of digital and analog feedback devices. Communication with external devices and controllers (e.g., IFOSC's controller) required some hardware arrangements. These hardware requirements are combined in form of the embedded control unit, the CCU (to be discussed in § 3).

For the user end, it was necessary to develop an application software that could take various instructions from the users through a GUI and pass it to the CCU in a standard way, as well as to provide a standard platform to include the controls of various other instruments and devices. Though the CCD data acquisition system on the IUCAA telescope has its own hardware (independent of CCU), its software controls were also required to be integrated within the same GUI. Further due to past experiences and to facilitate calibration of the system during the engineering tests, debugging, fault diagnosis, etc., an engineering GUI interface has been provided that would execute all the basic commands/tasks at an atomic level to allow debugging and fault diagnosis, and where all the required system parameters can be checked and set.

2.2. TELICS: Other General Requirements and Considerations

The factors described in § 2.1 determined the wide scope of TELICS, while there were other general considerations that played a key role in the conception of the system. TELICS was designed to provide access to various instruments and devices mounted on the telescope to the end user in the control room, so that they can be monitored and controlled as required during observations. However, remote accessibility to TELICS was also incorporated through an Ethernet network link. TELICS was also required to be flexible and scalable in nature to allow the integration of newer devices and instruments. The option of generalized control frameworks from various commercial solution providers were considered, but discarded due to the large initial investments needed for those which makes them often not affordable for small/medium size astronomical observatories. However, these frameworks could often be configured with relative ease and short turnaround times. Hence the emphasis was given to develop an in-house framework that could be replicated and expanded with little extra effort and cost. Also during the conception of TELICS the focus was to use inexpensive platforms (both in software and hardware), as well as to adopt the accepted industry standards, so that it could be easily maintained in the future; TELICS software was developed using open license components. Further, to reduce the complexity of the system, the number of different hardware and software components was minimized by using only single architecture in design of the hardware and software of TELICS. This approach will be discussed in §§_3–4. Lastly, a very important objective was to develop a user-friendly environment via a GUI, which would provide an efficient and consistent control of the facility.

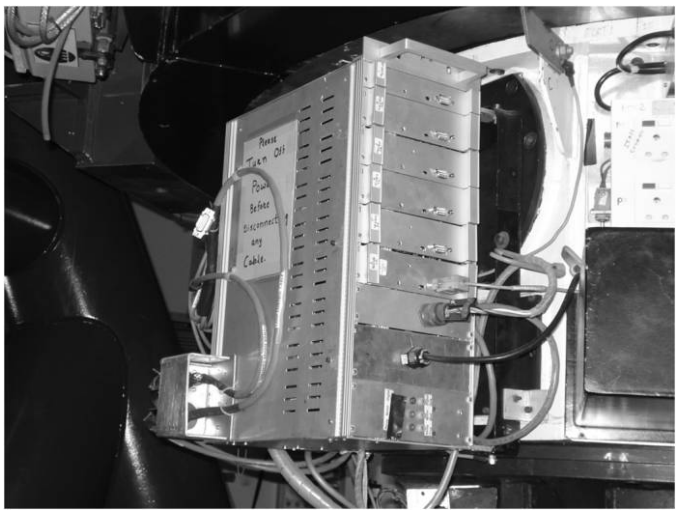

FIG. 1.—Common control unit (CCU) on the IUCAA telescope: CCU is the hardware part of TELICS, consisting of 6 identical cards on a backplane card using master-slave architecture. It also contains three SMPS power supplies. RS-232 based serial communication as well as Ethernet protocol has been used to communicate with CCU. See the electronic edition of the *PASP* for a color version of this figure.

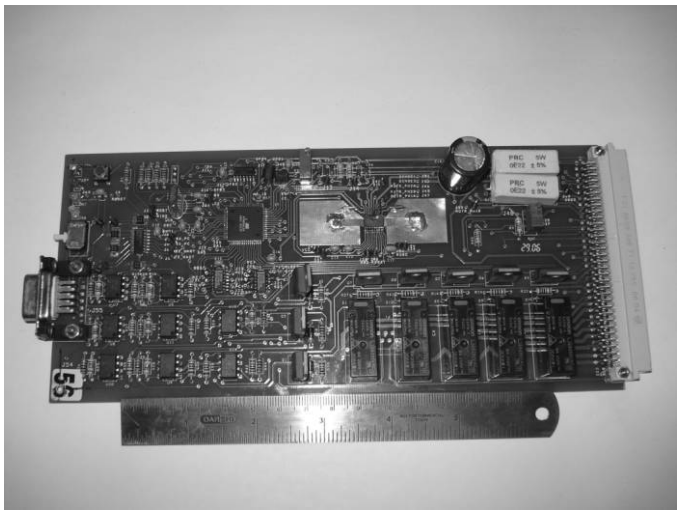

FIG. 2.—PCB card of the common control unit (CCU): This card can be configured as either master card or slave card depending on the setting of few jumpers on the backplane card. See the electronic edition of the *PASP* for a color version of this figure.

3. COMMON CONTROL UNIT (CCU)

The common control unit (CCU) is the embedded control unit of TELICS. It is designed and developed to accept various commands and instructions from the user through the GUI and execute them. The basic architecture of the CCU is built around the Atmega 128 microcontroller and the design is based on the master-slave backplane architecture. The CCU has been developed following the relevant industry standards (in design of the PCB and the enclosure box) for ease in long-term maintenance. The industry standard 3U size is used for the PCB design, while the enclosure box is of the dimensions $4\mathrm{U} \times 84\mathrm{T}$ (Fig. 1). Further to reduce maintenance efforts, only one type of card was developed. This CCU card contains all the required circuitry to fulfill a set of identified functional requirements. The CCU consists of six such identical cards, and to work in master or slave mode any one of the cards can be configured as the master card and the rest as slave cards by changing the settings of few jumpers on the backplane card. As all the CCU cards are identical, any card can be replaced by any other card without making any change in the architecture. Three switch-mode power supply units (SMPS) built into the CCU box provide the required power for all cards.

3.1. CCU: Layout and Hardware Architecture

The basic layout of a CCU PCB card is shown in Figure 2. At the heart of the CCU is an ATmega 128 microcontroller chip from ATMEL Corporation.[3] It is an 8 bit microcontroller based on AVR-enhanced RISC architecture. ATmega 128 offers various features that make it ideal for our application, e.g., an inbuilt serial peripheral interface (SPI) for interprocessor communication, dual programmable serial universal asynchronous receiver transmitters (USART), 8 channel 10 bit ADC, etc. Also, the availability of 53 programmable I/O lines make it well suited for the applications where a large number of devices and components are to be controlled or monitored. Further, the inbuilt architecture for the SPI and USART communication systems have provided the intended simplicity in the hardware and software system design. The CCU uses both USARTs: USART-1 is used for the host server to master card communication, while the USART-2 is used to communicate with any other external system. An in-system programming feature is used to load programs into the microcontroller. However, at the output, only one 9 pin connector is used for the in-system programming as well as for USART-2. Using a DPDT switch, any one of these modes can be selected. SPI is being used for the master-slave communication process.

Apart from the microcontroller, another important component in CCU is the A3984 stepper motor driver IC from Allegro MicroSystems.[4] It has a built-in translator and is designed to operate bipolar stepper motors in full, half, quarter, and 16 microstep mode with an output drive capacity up to 35 V and ± 2 A, thus allowing a variety of effective gear ratio flexibility. The A3984 also has the ability to operate in slow or mixed decay current modes. A CCU card consists of an ATmega 128 microcontroller, an A3984 stepper motor driver IC, a RS232 driver/receiver communication IC to convert the microcontroller

[3] See http://www.atmel.com/dyn/products/productcard.asp?partid=2018.

[4] See http://www.allegromicro.com/en/Products/PartNumbers/3984.

signal to the RS232 standard, and 5 relays from the OMRON G6RN series and their driver circuits. Further, an external shutter control circuit can also be added to this card using the connections of 2 of the 5 relays. These relays are kept isolated from the microcontroller using opto-isolators, and three different SMPS power supplies are used for digital power (5 V–5 A), relay power (5 V–5 A), and power for the stepper motor driver IC (24 V–9 A).

The schematic of one card and its external connections on the backplane card is shown in Figure 3. A card could be configured as master or slave by changing the logic level of a *master-slave identification pin* on the backplane card andthe execution of an embedded software program. A set of three lines (known as address lines) is used for the purpose of slave addressing, and the master card would use them to select a particular slave card. Communication between master and slave cards utilizes the built-in SPI communication protocol, and uses three lines (i.e., master in slave out [MISO], master out slave in [MOSI], and clock signal) for this purpose. Thus all six cards of the CCU are connected with each other via these six lines on the backplane card. Only one of the cards (the master card) can communicate with the host server PC. Moreover, the same embedded software is loaded into each of the cards. The CCU is finally mounted on the telescope and fetches power from the mains supply available at the focal station of the telescope. It is connected to the main server in the control room via an optical fiber cable link working on RS232 protocol. The controls of all the instruments/units on the telescope are routed through the CCU (Fig. 4). Currently the CCU controls the operations of calibration unit, PI-CCD camera, and centralized power supply controller. Access to IFOSC's controller has also been given

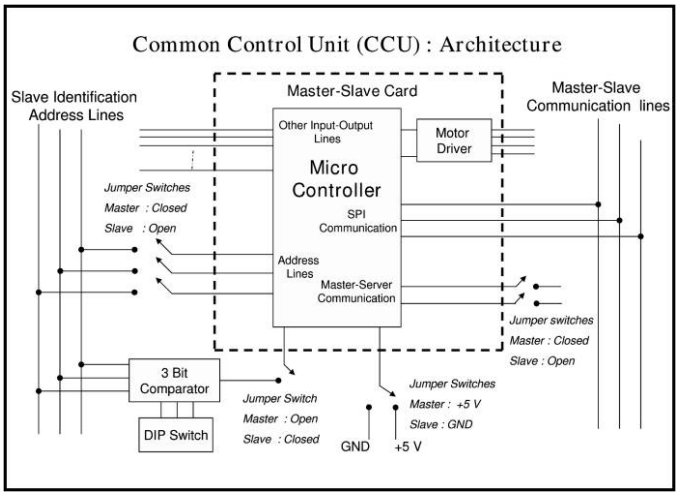

FIG. 3.—Hardware architecture of the CCU: The architecture of a CCU card and its interfaces are shown on the backplane card. There are 6 identical setups are used in the CCU connected by 3 slave address lines and 3 master-slave communication lines. The settings of 8 jumper switches on the backplane card can be changed to configure a CCU card in master mode or in slave mode.

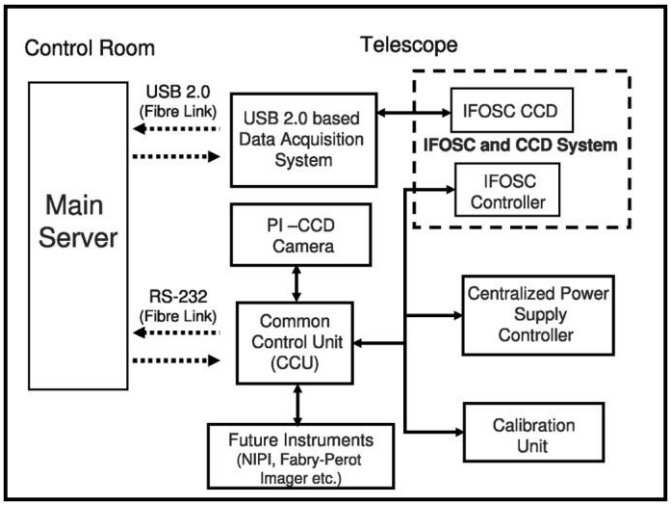

FIG. 4.—Block diagram showing the instruments/units integrated to TELICS. CCU controls the operations of calibration unit, power supply controller, and PI-CCD camera. IFOSC controls are also routed through CCU. Provisions to operate several future instruments/units (e.g., NIPI, an imaging Fabry-Pérot spectrometer, an integral field unit, etc.) have also been made to the CCU. See the electronic edition of the *PASP* for a color version of this figure.

through the CCU. In addition, controls of some of the planned instruments on the telescope are also given to CCU. Further, the potential to use several CCUs in a daisy-chain configuration to control additional instruments and devices also exists.

3.2. CCU: Software Architecture and Communication Protocols

The embedded software architecture of the CCU is responsible for the execution of tasks. Like the hardware architecture, emphasis is given to reducing the complexity of the system and therefore a similar strategy is followed in the development of embedded software for CCU. Only one kind of software is developed, which is loaded on all the cards of the CCU, making all cards identical to each other. This helps greatly in the maintenance of the cards as well as in the further development of the software. The embedded software architecture of the CCU can be categorized in four parts:

1. Command structure that defines all of the basic tasks to be executed and the structure of each command at the engineering level,
2. Communication protocol between the user's server PC and CCU,
3. Communication protocol between master and slave cards,
4. Algorithms for the execution of various commands.

The command structure for the CCU is the basic framework on which the whole TELICS software architecture (both the CCU's embedded software and the GUI support) is based. There are in all 28 basic commands in TELICS to control the various parameters of the stepper motors, to monitor and change the

status of various switches and sensors, and to transfer/receive the data to/from an external instrument or device. Each of the basic commands is designed to perform a specific function. Details of these commands and their functionality are given in Table 1. Various combinations of these basic commands are constructed at the user's end with appropriate parameters through GUI to execute a task on the telescope (e.g., bringing a particular filter into the path of the beam, turning ON/OFF spectral lamps for calibration purposes, etc.) and hence providing automation to the astronomical observations. A command is made up of three sections: the *Header* carries the address of the card (system ID) for which the command is meant, a command code which is specific to each basic command (this will indicate what function is to be performed), and the parameter size which specifies the total number of bytes in the parameter space of the command structure; the *Parameter space* contains the values of the parameters required for the execution of the task; and the *Trailor* contains one byte which is the checksum of all bytes present in the header and parameter sections of the command structure. It is used to check the success of the transmission of the command structure to the other side. As an example, the basic structure of the MOVE command (to rotate the stepper motor by 1000 steps in clockwise direction) is

```
//Header:
'system ID' = 3 (1 byte)
'Command Code' = 2 (1 byte)
'Parameter Size' = 5 (1 byte)
//Parameters:
'Number of steps' = 1000 (4 bytes)
'Motion Direction' = +1 (Clockwise)
  (1 byte)
//Trailor:
'Checksum' = 1 byte
```

Appropriate byte packets are generated by the application software based on the user's input via the GUI and sent to the relevant card through the master card. The algorithms for server to master communication, master to slave card communication, and for the execution of all the basic commands have been developed. As TELICS is the single node that processes the control of all the instruments on the telescope, multithread processing of the commands has been incorporated during the development of embedded software. This means that any given card (whether it is master or slave) can execute multiple jobs at the same time. This is in particular helpful in case of the master, which should be able to talk to the server all the time but should also perform some tasks for the hardware

TABLE 1

BASIC COMMANDS OF TELICS

Number	Command	Parameters	Function
1	Init	Address IDs of all the cards	To keep track of the active cards in the CCU
2	Move-profile	Number of steps and direction	To move a motor with acceleration profile
3	Move-constant	Number of steps and direction	To move a motor with constant speed
4	Stop	None	To stop a motor with profile
5	Halt	None	To stop a motor immediately
6	Goto reference	Direction	To move a motor up to a reference
7	Sleep motor	On/off	To enable/disable sleep mode of a motor
8	Delay	Delay time	To provide a delay between two executions
9	Switch	Switch-ID, On/off	To On/off a switch
10	Period blink	Switch-ID, On/off durations, n	To blink a switch n times
11	Halt blink	Switch ID	To On/Off a period switch
12	Define macro	Macro-ID, commands	To define a macro
13	Execute macro	Macro-ID	To execute a macro
14	Transfer data[a]	Data (in bytes)	To send a byte-packet to an instrument
15	Set motion profile	Speed, acceleration	To set motion profile of a motor
16	Set current position	Position data	To set current position of a motor
17	Set microsteps	Microsteps	To set microsteps of a motor
18	Set constant speed	Speed	To set a constant speed for a motor
19	Get motion status	None	To get back status of a motor's axis
20	Get motion profile	None	To get back motion profile of a motor
21	Get microsteps	None	To get back microstep setting of a motor
22	Get constant speed	None	To get back constant speed of a motor
23	Get current position	None	To get back current position of a motor
24	Get limit status	None	To get back status of a motor's axis
25	Get switch status	Switch-ID	To get back status of a switch
26	Get blink status	Switch-ID	To get back blink profile of a switch
27	Get ADC output	Sensor-ID	To get back output of a sensor
28	Get external response[a]	None	To get back response of an instrument

[a] Command-14 will also receive the response from the instrument in the same sequence. This response can be fetched by command-28.

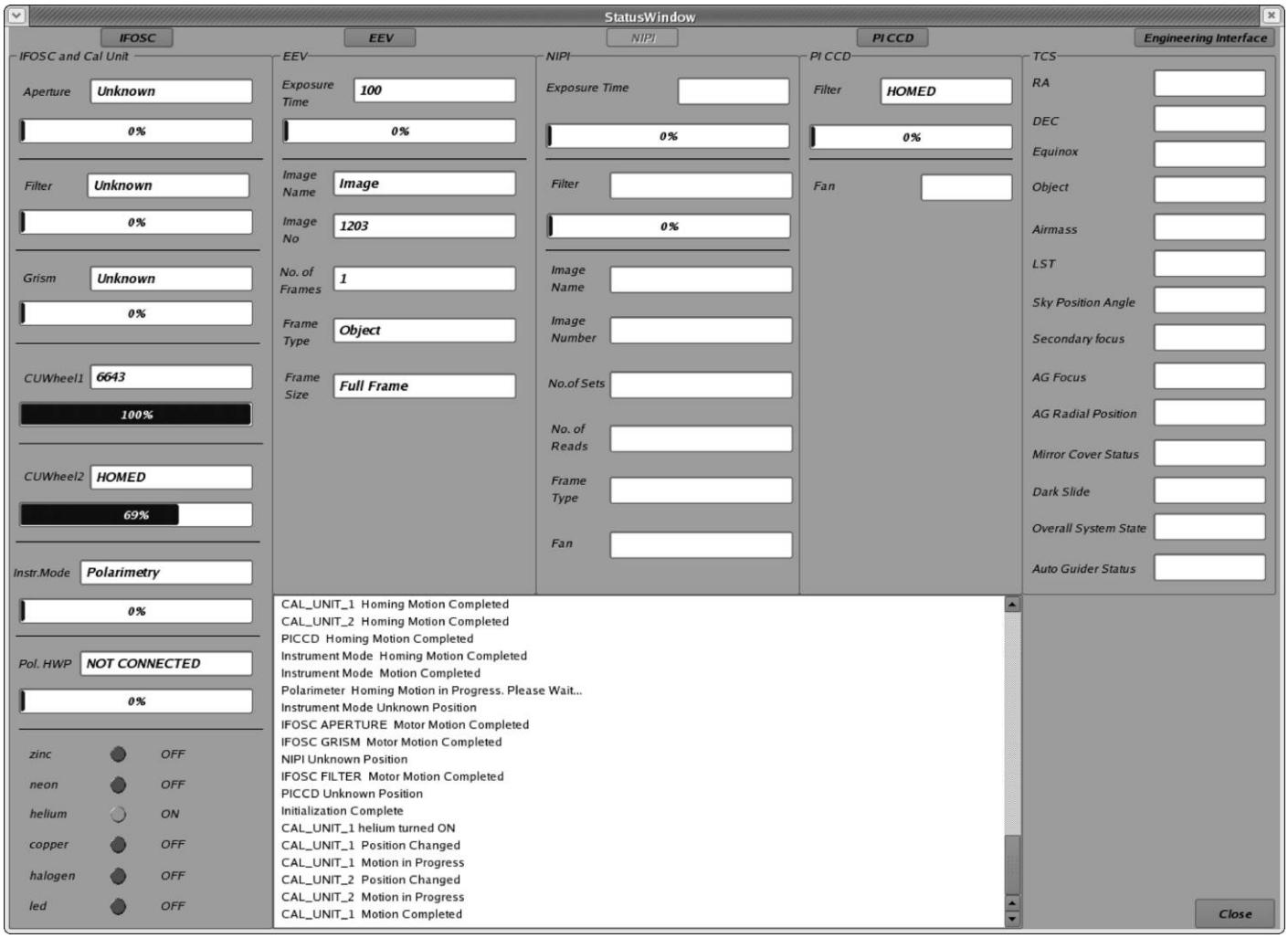

FIG. 5.—Main window of the graphical user's interface of TELICS. See the electronic edition of the *PASP* for a color version of this figure.

associated with it. Further, all the cards can perform their respective jobs simultaneously. The possibility to define macros for the CCU has also been incorporated in to the application software as well as in the embedded software of the CCU. With the macro facility, various commands with their respective parameters can be set and stored in a sequence in memory of the microcontroller and could be executed at any other time with a single command issued by the host. This would allow users and the engineers to make their own sequence of commands during the observations and routine maintenance of the system.

4. APPLICATION SOFTWARE AND GRAPHICAL USER INTERFACE (GUI)

At the front end of the TELICS, a GUI is provided to take user instructions, and provide feedback regarding the state of the system and/or actions being currently executed (Fig. 5). As mentioned earlier (§ 2.1), the GUI not only provides an interface to TELICS but also integrates the software controls of a data acquisition system. This data acquisition system has its own separate hardware (external to CCU) and is not discussed in this article. Thus GUI combines the controls of the following two separate hardware system in one graphical window (Fig. 6): (1) the TELICS application software, and (2) the application software for the data acquisition system.

The GUI is developed in Qt[5] and provides the real-time visualization of various functions and system status. The status of various processes can be monitored on the GUI screen as they progress. It provides separate interface windows for different instruments and subsystems mounted on the telescope. The control interface for IFOSC, PI-CCD camera, NIPI, and CCD data acquisition system are given through the main window of GUI.

[5] See http://trolltech.com/products/qt.

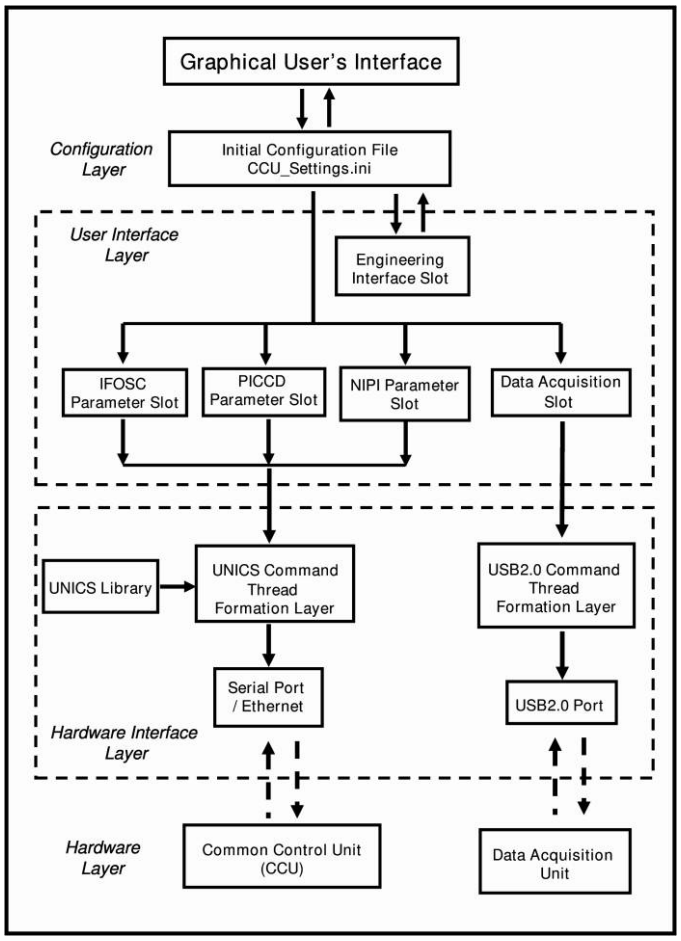

FIG. 6.—Software architecture of TELICS application software and GUI: The GUI combines the software controls of TELICS application programs and a USB 2.0-based data-acquisition system.

While working with any of these systems, the user can set all of the required parameters through this interface and the tasks can be executed. An interface to SAO-DS9 image visualization package[6] is given to GUI through the XPA messaging system[7] for quick-look inspection of acquired images. The option of selecting different image acquisition modes (e.g., fast, binned, and bias-subtracted mode) in the GUI helps the user to choose appropriate modes while observing, thus minimizing the overhead time.

The TELICS Application Software converts the user's instructions to a sequence of commands as per the TELICS command structure that are sent to CCU through a serial link or Ethernet network. The application software was developed in C/C++. Along with the embedded software (§ 3.2), it supports multithreaded execution of tasks. This gives the flexibility to control a number of independent processes simultaneously. For example, selected motors in the systems could be operated simultaneously using CCU, or spectral lamps in the calibration unit could be operated while moving the filter wheel in IFOSC. All operations of the TELICS application software (i.e., interaction of the GUI with CCU and data acquisition system) are controlled through entries in a configuration file named *CCU_settings.ini*. This configuration file is stored in ASCII text format and contains all the information describing the state and configuration details of each subsystem of TELICS. The inclusion of any new subsystem to TELICS would require addition of corresponding entries in this configuration file, and a corresponding interface window in GUI along with the underlying software of the subsystem. This file also contains all the input parameters for the serial communication protocol and Ethernet protocol. During the run time, the entries in the configuration file keep on updating with the new values corresponding to change in any of the parameters of the subsystems. A library of TELICS commands has also been developed. Any task executed via GUI would create proper command packets using the data of *CCU_settings.ini* and the TELICS library. These command packets would then be sent to CCU through a serial link. The built-in Linux libraries are used for serial communication purposes.

The GUI also provides access to the application software of a data acquisition system. It is an FPGA-based system and communicates with the server through a USB 2.0 interface. The same configuration file, *CCU_settings.ini*, is used to interact with data acquisition system as is being used for the CCU. However, the data acquisition system has a different library of its own and does not use TELICS libraries. The GUI also acts as an interface between the telescope control system (TCS) and the application software. During observations, various observation parameters from the TCS can be read and displayed to the screen and can be included in the header of the output images.

An engineering interface section of the GUI has been developed for the engineering-level checks of the subsystems. It provides the ability to configure basic parameters of all hardware and transfer each of them to CCU in any desired sequence. Also, this can be used to set the input parameters for serial communication protocol and TCP/IP. This feature has been proved to be of great utility while testing and debugging the system. Access to the engineering GUI is restricted through authentication due to safety concerns.

5. CONCLUSION

TELICS has proven to be a leap forward as compared to the earlier back-end instrument control environment at IGO, which had several different controllers for different functionality; e.g., a motion controller to control the motion of all the motors, a power supply controller, IFOSC controller, and calibration unit controller. All those controllers had their own different hardware and software architectures. In addition, different

[6] At http://hea-www.harvard.edu/RD/ds9.
[7] At http://hea-www.harvard.edu/RD/xpa/index.html.

communication protocols were being used by each of them to communicate to the server, therefore requiring several communication links between the server and those controllers. This scenario made them inconvenient to operate and maintain as well as to provide a unified user interface. TELICS incorporates the functionality of motion controller, power supply controller, and calibration unit controller in the form of the common control unit (CCU) and hence reduces the system volume and complexity by a large factor. This also gives an advantage to CCU on power consumption. Further, IFOSC's commands are routed to the IFOSC controller through CCU. Thus CCU is the single node on the telescope for instrument control, with only one communication link to the server. Along the same lines, provisions have been made in TELICS to integrate the controls of future instruments/units. The operational overheads for setup, initialization, shut down etc. have been reduced significantly by the multithreaded executions of the application control code, which is a distinct improvement over the earlier situation based on different communication protocols and software architectures of the various controllers. Also some of the observation setups (e.g., polarimetric mode of IFOSC) that were earlier done manually have now been automated within the framework of TELICS. On the maintenance side, having only one kind of hardware card and embedded software in the CCU proved to be an extremely efficient way to reduce the downtime of the system. In case of malfunctioning of any card, the system can be restored by a simple exchange of that card with another one.

To summarize, the TELescope Instrument Control System (TELICS) has been developed keeping the general requirements of smaller and medium size observatories in the picture. It provides a cost-effective solution with simple hardware and software architecture to control and monitor the various instruments mounted on the telescope. The hardware architecture of TELICS is based on the ATmega 128 microcontroller and is designed as a slave to a master host computer, while the user space software architecture runs on the Linux operating system. The user-friendly GUI based on the Qt platform on Linux was developed for the software control of TELICS; it provides intuitive control both at the operational level as well as the engineering level, and a good visibility of the current status of the system. The engineering mode of the GUI has been proved very useful in debugging hardware problems. Further, considering the need for remote accessibility of TELICS, an option of replacing the serial communication link between the server and the embedded common control unit (CCU) on the telescope by an Ethernet link has also been given to TELICS. TELICS hardware offers adequate capability for expansion in future due to its generic architecture and interface approach. It greatly simplifies the issue of the control of current instruments and provides an easy solution for adding additional instruments and devices to the telescope. Overall, TELICS is now a mature instrument control system that can handle existing and planned instruments, and has the potential for future extensions. TELICS has been in use on the IUCAA 2 m telescope since 2008 March.

We would like to thank Mr. Sunu Engineer, Mr. Ashwin Kumar, Ms. Ashwini Kadam, Mr. Moin Shaikh, and Mr. Sujit Punnadi for their support in the development of TELICS software. We are also thankful to other members of the Instrumentation Laboratory at IUCAA for their help and support during the development of TELICS. M. K. S. thanks the Council for Scientific and Industrial Research (CSIR), India, for the research grant award No. 9/545(25)/2005-EMR-I.

REFERENCES

Tandon, S. N. 1998, Bull. Astron. Soc. India, 26, 377
Gupta, R., et al. 2002, Bull. Astron. Soc. India, 30, 785
Ramaprakash, A. N. 2002, Bull. Astron. Soc. India, 30, 249
Ramaprakash, A. N., et al. 2003, Bull. Astron. Soc. India, 31, 465