PUBLICATIONS OF THE ASTRONOMICAL SOCIETY OF THE PACIFIC, **119**: 661–668, 2007 June

Control and Communications System for Remote Operation of an Infrared Radiometer

IAN S. SCHOFIELD AND DAVID A. NAYLOR

University of Lethbridge, Lethbridge, Canada; ian.schofield@uleth.ca, naylor@uleth.ca

Received 2007 April 4; accepted 2007 May 8; published 2007 June 7

ABSTRACT. An infrared radiometer (the Infrared Radiometer for Millimetre Astronomy, IRMA) has been developed to measure the amount of water vapor in the atmosphere through its emission at 20 μm. Water vapor is the major contributor to signal phase error in submillimeter interferometric arrays and the principal source of opacity for telescopes operating at infrared wavelengths. While earlier versions of IRMA required hands-on operation, the desire to operate at remote and often hostile sites necessitated the development of sophisticated and robust software. IRMA is a distributed, real-time control and data acquisition system spread across three different computers, and can be controlled remotely over the network by command scripts or a graphical user interface client program.

1. INTRODUCTION

Infrared spectroscopic measurements of the atmosphere above Mauna Kea, obtained using a high-resolution Fourier transform spectrometer, show that virtually all of the atmospheric opacity in a band around 20 μm arises from rotational transitions of water vapor (Naylor et al. 1984). Since no other common atmospheric molecules exhibit transitions in this band, it is an ideal indicator of atmospheric water vapor content. We have developed IRMA, a remotely controlled, fully steerable, infrared water vapor monitor. IRMA is designed to perform accurate radiometry in the 20 μm band, and by the use of an atmospheric model, to retrieve the atmospheric water vapor column abundance, expressed in terms of precipitable water vapor (PWV). IRMA has the potential to be a useful tool in the correction of water vapor–induced phase errors in submillimeter interferometry and also in a site testing, observation planning, and correction role for infrared and submillimeter telescopes.

The optical design of IRMA is presented elsewhere (Naylor et al. 2003). In this paper, we summarize the key features of the radiometer and discuss the technical details surrounding the control and communication system, which has been developed to allow operation at remote and often hostile environments, including Chile and Antarctica. Designed as a compact, low-power, semiautonomous instrument capable of receiving and executing batch instructions over the Internet, IRMA required a sophisticated, fine-grained control system that provided maximum operational flexibility. Furthermore, the software had to be sufficiently robust to allow the instrument to extricate itself from error conditions, such as the failure of any one of its subcomponents. This paper discusses the development, structure, and operation of the IRMA control system, as well as the unique challenges of developing astronomical instrumentation software for an embedded platform, instead of the more conventional PC workstation.

2. BACKGROUND

The first IRMA was installed and tested at the James Clerk Maxwell Telescope (JCMT) atop Mauna Kea, Hawaii, in 1999 December (Smith et al. 1999). The encouraging results of these early measurements spurred further refinements in the design and resulted in a more detailed comparison of methods of measuring water vapor above Mauna Kea (Chapman et al. 2004). It quickly became evident that IRMA had the potential to be a useful tool in site testing potential locations for future large ground-based telescopes, such as the northern Chilean Atacama Desert some 5000 m above sea level, or Antarctica. These remote locations are among the driest regions on Earth, making them ideal for infrared and submillimeter wavelength astronomical observations. However, unlike the well-resourced JCMT facility, the demands of operating in a harsh climate, at a remote site, and with minimal power and communication resources made it necessary to reconsider the control and communication requirements of IRMA.

3. IRMA OVERVIEW

In order to place the IRMA control and communication in context, it is first necessary to present a brief overview of the radiometer and a summary of the key subcomponents of IRMA that interface with the computers.

In its site-testing role, IRMA is designed to operate at a high altitude and remote location, where it will encounter extremes of temperature and hostile weather conditions, both potentially damaging to electronic components and circuitry. Figure 1 shows the compact and lightweight IRMA radiometer deployed near the Las Campanas Observatory in Chile. Also shown is

FIG. 1.—IRMA radiometer and altitude-azimuth mount stationed at the Las Campanas Observatory, Chile.

the altitude-azimuth (alt-az) mount that allows IRMA to point to any part of the sky. This compact format was designed to accommodate direct mounting of the radiometer itself (i.e., without its alt-az mount) to a telescope dish in interferometric applications. The optical system, shown in Figure 2, consumes most of the available volume. The remainder of the space is occupied by the various electronic subsystems.

The early versions of IRMA, which were proof-of-concept instruments and thus not designed for remote operation, contained hard-coded instruction sequences that the operator could call on demand. In the latest version, IRMA III (hereafter, simply referred to as IRMA), the design philosophy was changed to allow the operator access and control of all hardware components in order to permit operation at remote locations. In the new design, the operator can modify the behavior of the unit if hardware components are changed or upgraded (e.g., installation of a new Stirling cooler or blackbody), install workarounds in the event of component failure (e.g., inability of the Stirling cooler to hold a certain base temperature due to partial loss of vacuum in the cryostat or failure of a temperature diode), or simply have the option to do something novel with the instrument. Fine-grained control of each of the subcomponents of IRMA, and the ability to do so in any sequence, required the development of a custom scripting language: IRMAscript.

3.1. IRMA Hardware

Figure 3 gives a schematic overview of various IRMA hardware components and their means of interface with the IRMA computers. The IRMA chassis consists of an 28 kg aluminum box (roughly the size of a shoe box) suspended between the forks of an alt-az mount. The alt-az mount allows IRMA to rotate 360° in azimuth and 180° in altitude (i.e., elevation) to access any sky position. The extended range in elevation allows the device to be tipped upside down to protect it during severe weather conditions. Directly behind the entrance aperture is a motorized sliding shutter. The shutter has a dual role: it seals the IRMA unit, preventing moisture and dust from entering the device when not observing, and it contains a heated blackbody target that serves as a calibration source for the radiometer.

3.1.1. Optical Path

Inside the radiometer, light reflects from a 100 mm diameter, f/1, 90° off-axis parabolic mirror and is brought to a focus on an infrared photoconductive detector. The detector is attached to the tip of a cold finger by a mechanical clamp and is cooled to its operating temperature by means of a Stirling-cycle cryocooler. The incoming optical beam passes through a five-blade reflective chopper wheel, an antireflection-coated ZnSe window, and a

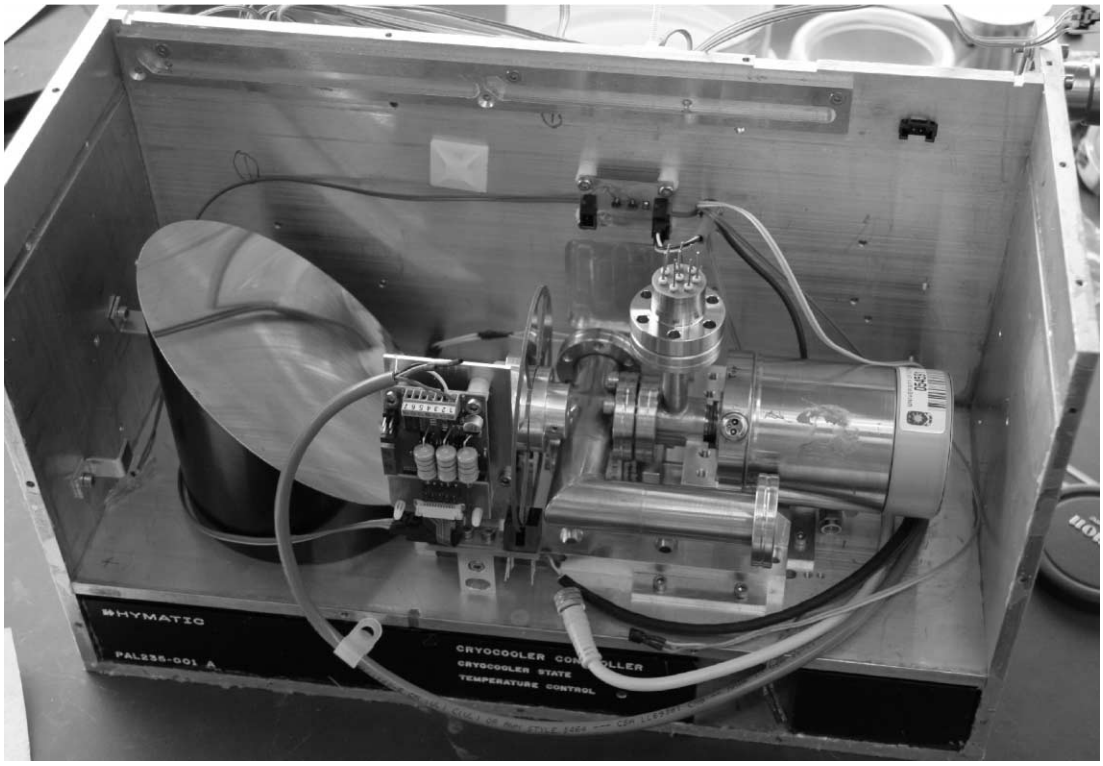

FIG. 2.—Cutaway view of the IRMA radiometer. *Left to right*: Parabolic mirror, reflective chopper and associated electronics, vacuum vessel, Stirling cooler. *Lower shelf from left to right*: Stirling cooler controller, IRMA power supply. Electronics board and control computer are attached to the reverse side of the rear metal plate.

narrowband infrared filter before reaching the detector. A single notch located on the circumference of the chopper wheel ensures samples are triggered on the same blade, thus eliminating uncertainties associated with blade-to-blade variations in reflectance and emittance.

3.1.2. Shutter/Calibration Source

The shutter/calibration source consists of a hollow aluminum block that is mounted in a track and driven by a lead screw. At opposite ends of the track are two slotted optical switches, both of which are mapped to two digital input-output (DIO) lines of the computer subsystem. Metal tabs that actuate the optical switches are placed at opposite sides of the shutter. The control computer polls these lines in order to determine when the shutter movement has completed.

The calibration source attached to the underside of the shutter consists of a carbon-black, enamel-textured epoxy coating deposited on a thin, metallic film heater. Embedded beneath the epoxy layer are two silicon diodes that act as temperature sensors and provide thermometry with an accuracy of ± 0.1 K. The coating has a high emissivity at infrared wavelengths. The blackbody is heated by passing an electrical current through the film. When the shutter is closed, the blackbody completely covers the optical aperture and is used to calibrate the radiometer. Typically, a two-point temperature calibration is performed: ambient and ~50°C. It is also possible to perform calibrations at temperatures between these extremes.

3.1.3. Stirling Cooler/Vacuum Vessel

The detector is cooled to its operating temperature of ~70 K using a Stirling-cycle cryocooler (Honeywell-Hymatic NAX025-001). A custom-designed vacuum vessel surrounds the cold finger of the cryocooler. The vacuum chamber has four inputs. Radiation enters through an antireflection-coated ZnSe window seen at the top of Figure 2. The input immediately to the right carries the detector electrical signal. The input to the lower right connects to the vacuum pinch-off tube, which provides a permanent vacuum seal.

The infrared detector is attached to the tip of the cold finger, and its temperature is servo-controlled by means of a collocated temperature diode. In order to reach its operating temperature, the internal pressure must remain below 10^{-4} mbar. Given the large surface area–to-volume ratio of the vacuum vessel, this is one of the most challenging aspects of the design. Individual

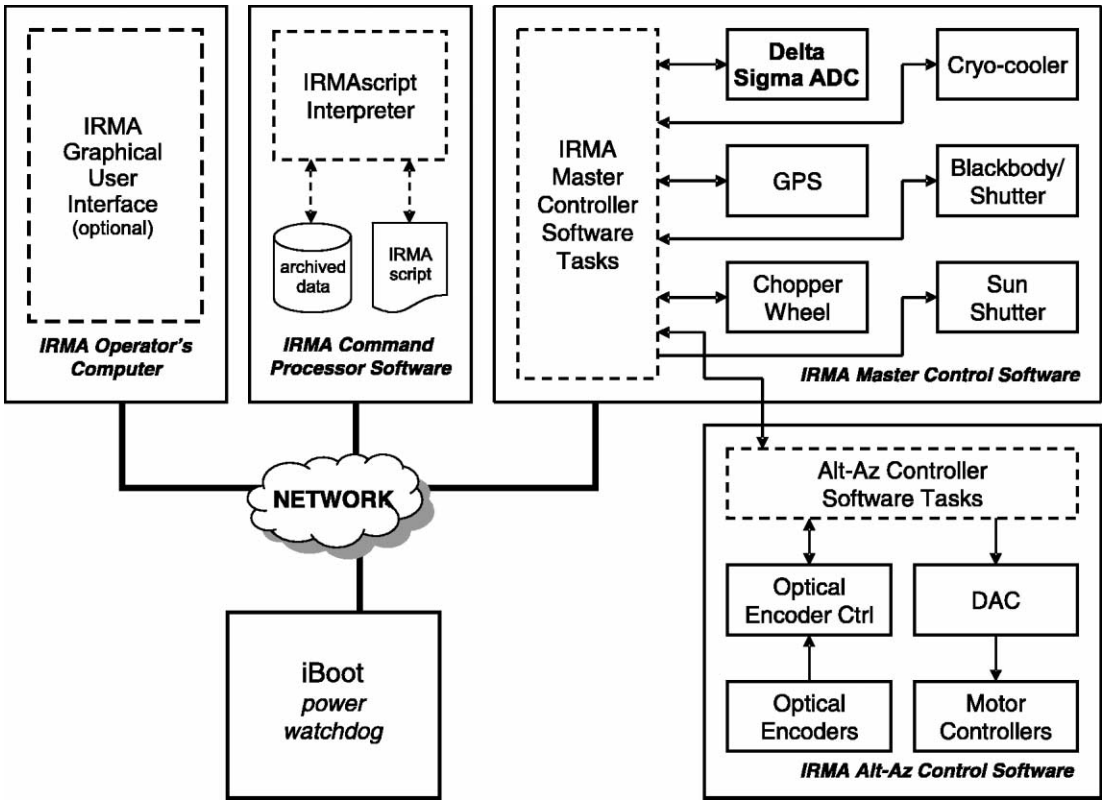

FIG. 3.—IRMA hardware/software structure showing three primary subsystems: the command processor (CP), the master controller (MC), and the altitude-azimuth controller (AAC), each of which are hosted on their own computer. Scripted command sequences are transmitted to the CP over the network either manually or by means of an optional graphical client program. The CP translates each scripted command statement into lower level IRMA command packets and sends them to the IRMA MC over a network link. Altitude-azimuth commands are dispatched by the MC to the AAC over a serial link. System functionality is divided between the MC and AAC; the MC is responsible for controlling and reading data from sensors, while the AAC looks after motion control of the alt-az mount.

components are carefully cleaned, and once assembled, the unit is heated and pumped to achieve pressures below 2×10^{-8} mbar. In order to reach its target lifetime of 5 years, the leak rate can be no greater than 1×10^{-15} mbar cm^{-2} s^{-1}.

3.1.4. Detector

Infrared radiation is detected by a mercury cadmium telluride (MCT) photoconductive detector (Kolmar Technologies, Inc.). The detector is optimized for observations at ~20 μm. A narrowband filter (Lee et al. 1996) placed immediately in front of the detector limits the spectral response of the system to a bandwidth of ~2 μm (~50 cm^{-1}), carefully chosen to match the emission spectrum of water vapor. The detector conductance is a function of the incident radiant flux and is converted to a voltage using a conventional constant bias current circuit. A lock-in amplifier converts the modulated signal voltage to a DC voltage, which is digitized by an analog-to-digital converter (ADC).

3.1.5. Analog-to-Digital Converter

The heart of the IRMA data acquisition system is a four-channel, 24 bit delta-sigma ADC (Cirrus Logic CS5534). In addition to the infrared signal, the ADC also samples, by means of an analog multiplexer (and thus at lower rates), the atmospheric pressure, relative humidity, and eight temperature channels. The ADC relies on massive oversampling, noise shaping, and digital filtering to achieve near 24 bit sample resolution at the expense of longer sample integration times. Digitization of the radiometer signal at a 22 bit resolution requires an integration time of 535 ms. For a wind speed of 10 m s^{-1}, the atmosphere passing across a 10 m diameter telescope varies on a timescale of about 1 s, sufficient for the ADC to acquire stable readings.

3.1.6. GPS

Accurate time and positional information is provided by a GPS receiver (GlobalSat DK-ER101). The GPS board continuously emits a formatted serial string containing positional and

time data. The master controller, when queried for the current GPS time or commanded to synchronize its onboard real-time clock (RTC), parses the appropriate data transmitted by the GPS board.

3.1.7. Control Computers

Limited space and electrical power, and anticipated harsh operational conditions, led to the selection of the Rabbit family of 8 bit microcontroller modules (Rabbit Semiconductor, Inc.) over conventional PCs. The non-networked RCM2010 core module was selected to control the alt-az mount, while the networked RCM2100 module was chosen to perform instrument control and data acquisition. The alt-az microcontroller controls motion on the elevation and azimuth axes by means of a digital motor controller (Maxon Motor 1QEC50V) driving twin brushless DC motors (Maxon EC167129). Alt-az motion control was off-loaded onto a separate microcontroller to reduce computational load in the main controller and reduce the number of control lines passing through the umbilical cable connecting the detector unit to the alt-az base.

Both controller modules are based on the Rabbit 2000 8 bit microprocessor and provide 40 lines of TTL-level DIO, of which eight can be reassigned to four serial IO channels. Like other small, embedded processor boards, the compiled binary executable resides in the controller module's flash memory and must be compiled using a Windows-based cross-compiler, Dynamic C.[1]

The Rabbit 2000 is based on the Zilog Z-80/Z-180 architecture. Consequently, the Rabbit 2000 shares a similar register layout, memory-addressing modes, and machine instructions with the Zilog processor. The primary difference between the two processors is that the Rabbit 2000's register layout is optimized for 16 bit arithmetic and memory manipulation, unlike the original Z-80 architecture. This feature makes the Rabbit 2000 more compatible with C language compilers, which are typically biased toward 16/32 bit arithmetic and memory access.

4. CONTROL SYSTEM SOFTWARE

Recent astronomical instrumentation control systems (e.g., automated observatories) have relied on distributed processing in order to share the computational load among multiple computers, and on real-time scheduling for hardware control (Spillar et al. 1993; Anderson et al. 1999). Typically, this has involved x86 PC hardware running under DOS or UNIX-like operating systems, such as LynxOS, or more recently, GNU/Linux. Furthermore, some of these systems are controlled through custom control scripting languages (Stark et al. 2001). The IRMA control system follows this heritage by the nature of its distributed design, scripting support, and real-time performance. Use of embedded microcontroller modules in place of PCs sets IRMA apart.

IRMA's control software is divided into three software modules: the command processor (CP), master controller (MC), and alt-az controller (AAC). The CP runs on a Linux-equipped PC, while the MC and AAC run on separate Rabbit 2000 microcontroller modules. An optional graphical user interface (GUI) front-end client and a stand-alone watchdog device are also included.

The CP translates high-level scripted command sequences into low-level IRMA instructions and passes them to the MC. The MC decodes these packets and executes them in the hardware. The CP also logs status and error messages received from the MC, which the MC generates after the execution of each low-level instruction packet. Communication between the CP and MC is performed over the observation site's Ethernet network. Where necessary, the MC passes alt-az commands along to the AAC for execution, using a serial channel interconnecting the two modules. The AAC handles all functionality relating to positioning of the alt-az mount.

The IRMA GUI client software provides the operator with an easy-to-use interface that features the most commonly used features of the radiometer. The need for a GUI was identified early on, because writing command scripts is a tedious and error-prone task. Fundamentally, the GUI is a script generator that calls from a collection of generalized scripts and customizes them with the operator's requested parameters.

The CP and its Linux-based host PC constitute a communications gateway to IRMA. All commands sent to and received from the instrument pass through the CP host. The CP host provides ample disk space for data storage, as well as sufficient computational power to drive script language interpretation. Moreover, it provides a litany of network services that allow versatile and secure access to IRMA over the Internet. Offloading script interpretation and data archiving onto the PC leaves the MC and AAC free to control and acquire data from the radiometer.

Invoking command scripts to control IRMA while logged into the CP host over a secure shell (SSH) connection is one of the most simple and direct ways to control the instrument. However, it is also the least intuitive, because it involves command-line commands. Running the GUI remotely via a remote X-session was seen as the preferred solution but has proven difficult to achieve, due to firewall restrictions and limited bandwidth at test site installations. To alleviate these problems, the GUI can be run on the operator's computer. The command scripts generated by the GUI are forwarded to the remote CP host via secure copy (SCP). This scheme requires that additional software run on the remote CP host alongside the CP software in order to dispatch scripts to the CP as they arrive and return feedback to the GUI. Large time latencies when sending commands from the remote GUI has been the primary weakness in this arrangement.

[1] Dynamic C Users Manual 2006, Rabbit Semiconductor, Inc.

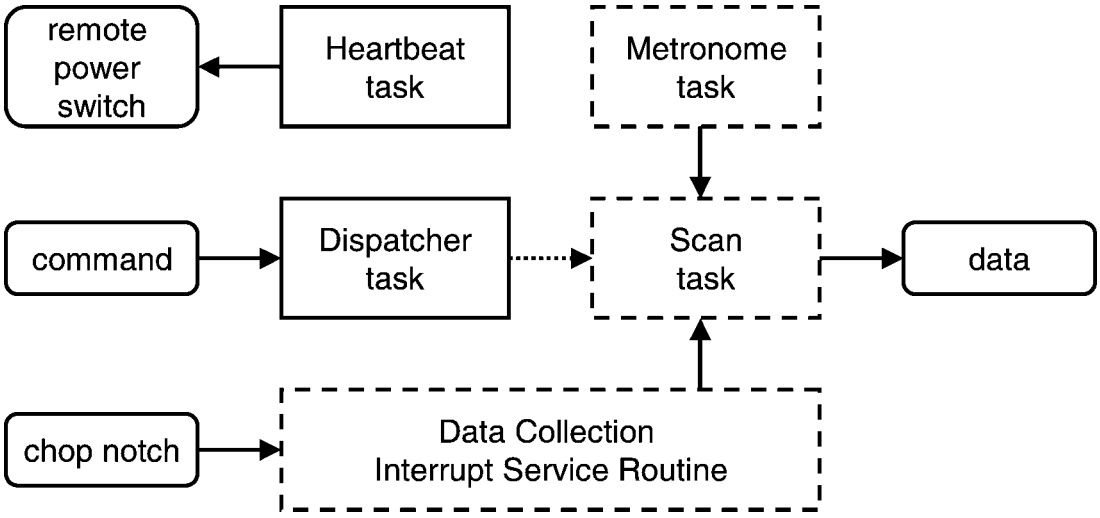

FIG. 4.—IRMA software task structure. The heartbeat and dispatcher task (*solid boxes*) run continually. When called on to perform a scan, the dispatcher task enables the scan task, metronome task, and data collection interrupt service routine (*dotted boxes*). The scan task sleeps unless woken up by the metronome task at a fixed interval, upon which the scan task sends a data packet to the CP to be archived. A notch on the reflective chopper triggers the interrupt service routine that performs the actual analog-to-digital conversion.

4.1. Real-Time Multitasking

Both the MC and AAC use the μC/OS-II (Micrium, Inc.) real-time kernel as the means to achieve real-time preemptive multitasking on the Rabbit 2000 hardware. μC/OS-II provides a small, efficient kernel in which to run software tasks that are similar to processes used in Linux or UNIX. Priority-based scheduling, coupled with a minimal operating system, allows μC/OS-II to guarantee that each task gets serviced in a predictable amount of time, thus providing real-time performance.

Both the MC and AAC are designed as a set of multiple, independently running tasks. In order for the IRMA system to operate responsively, the MC and AAC must process command requests expeditiously, even if they are in the midst of executing a job. Therefore, in both the MC and AAC software, a separate task runs in the background, listening for incoming commands. When a command is received, the task listening for incoming commands dispatches the request to another task waiting dormant in the background and then returns to listening.

Figure 4 shows the software task structure of the MC software; each block represents its respective process. On startup, the dispatcher task and heartbeat task run concurrently. The heartbeat task, the highest priority task of all the tasks, sends a network packet every 30 s to the remote power switch (Dataprobe IBoot), thus preventing the device from cycling the power to MC and AAC. Likewise, the dispatcher task runs continuously in the background, listening for incoming command packets and executing them when they arrive. Most short-duration requests (e.g., reading the internal temperature sensors) are handled within the dispatcher task.

Scanning, which involves sampling the IR detector on receipt of a notch interrupt signal from the chopper wheel, is a long-duration function and must be run within a separate task. The scan task, normally dormant, is awoken by the dispatcher task, as indicated by the arrow in Figure 4. The scan task in turn calls a timing task (called the metronome task) that monitors the number of data points collected by an interrupt service routine (ISR). When the number of data points required to populate a data packet has been reached, the metronome task signals the scan task to bundle the data in a packet and pass it to the CP software. During scans, the CP forks a separate process that collects and stores data packets generated by the MC.

The reflective chop wheel triggers the data collection ISR, which in turn commands the ADC to sample the infrared signal. These samples are stored in shared memory that the scan task can access. The ISR is written primarily in Rabbit 2000 assembly language in order to maximize execution speed. The sum total of the time spent instructing the ADC to sample the signal channel is 690 μs. No more than 310 μs at any one time is spent within the ISR, because it has multiple entry points and is structured as a state machine.

Communication between tasks is performed using global variables and flag signals, a mechanism similar to POSIX signals (Stephens 1992). Unlike conventional multitasking operating systems that time-slice among the competing running processes, priority-based multitasking used in μC/OS-II must be explicitly performed by strategically placing context-switching mechanisms inside the program. These mechanisms involve using nonblocking timed sleeps, signaling sleeping tasks when a requested shared resource becomes available (e.g., sema-

phores), and waiting on flags raised by other tasks. In all cases, the waiting task is made dormant by the scheduler, so it consumes no CPU cycles, and is awoken by the scheduler when the appropriate conditions are met.

4.2. Implementation Languages

The MC and AAC are both implemented in Dynamic C, a vendor-supplied version of C similar to ANSI C (Kernighan & Ritchie 1988) but tailored to the capabilities of Rabbit 8 bit processor modules. Numerous libraries, including their source code, are provided by the vendor to provide system functions to access onboard hardware. By replacing the vendor-supplied parallel port digital IO functions with Rabbit 2000 assembly-language macros, the speed of execution of DIO operations are increased by a factor of 4. Dynamic C has a simple mechanism for embedding assembly code within C, or vice versa. Other than the library definition/inclusion mechanism, which is markedly different from that found in ANSI C, both dialects share many similarities.

The script language interpreter, the CP, is implemented in Perl (Wall et al. 2000), a popular interpreted scripting language that allows for rapid application development, powerful string and regular expression handling, and easy-to-use modules that encapsulate advanced functionality. This makes maintaining the system's code base relatively easy; an important factor when considering software maintenance by less experienced programmers, many of whom do not have computing science backgrounds.

4.3. System Control Scripting Language

IRMA is primarily controlled by a custom-designed interpreted language called IRMAscript. A graphical interface was later implemented to provide a simple point-and-click interface that encapsulated the most commonly used functions of the instrument. Custom control languages for scripting complex control sequences of complex instrumentation (especially remotely controlled observing instruments like IRMA) have a long history. Remote automated observatories at Antarctica, such as the Antarctic Submillimeter Telescope and Remote Observatory (AST/RO; Stark et al. 2001) and the earlier Infrared Photometer Spectrometer (IRPS; Ashley et al. 1996), both saw the need for complex control systems driven by custom interpreted scripting languages.

The language itself is very simple, consisting of primitive instrument commands and control structures (e.g., iterative loops and if-then branching constructs). IRMAscript also supports simple arithmetic, time/date operations, console/file IO, global variables, and time delays. Care was taken to make it easy to read and familiar to those with rudimentary programming experience. Syntactically, it is similar to PASCAL and Perl. The following snippet of IRMAscript code illustrates the use and form of command scripts used within IRMA. This example commands IRMA to open the blackbody shutter, while setting a 40 s time-out period in which to abort the process if the shutter is jammed.

```
assign $moving 3
assign $sOpen 2
assign $sClose 3

$currTime = rtc read epoch_time
eval $timeout = $currTime + 40

shutter state open
do
    $x = shutter read limit
    print $x
    $currTime = rtc read epoch_time
    print 'CURR_TIME:, \s, $currTime, \n'

# if shutter hasn't opened within 40 secs,
# assume that it's jammed or disconnected.
# Reverse shutter direction and exit

    if $currTime > $timeout
      shutter state close
      goto DONE
    endif

  wait 2
while $x != $sOpen

label DONE
```

The entire functionality of IRMA is contained in a library of scripts. These short scripts enable precise control of each of the instrument subsystems (e.g., reading date/time from the GPS or slewing the alt-az mount to an R.A.-Decl. coordinate). The script library can change and grow as hardware components are modified and new instrument operation modes are developed. The GUI draws from this library, associating graphical controls with respective scripts and customizing them according to the operator's input parameters.

5. CONCLUSION

The IRMA control system employs a modular approach, distributing the processing among three processors in its original design. Rabbit 2000 8 bit microcontrollers, chosen for their compact size, low cost, and low power consumption, are embedded within the instrument and are responsible for system control and data acquisition. Moreover, the Rabbit microcontrollers are tolerant of low temperatures, having been successfully operated at -80°C. An IRMA unit is currently being

prepared for deployment in Antarctica in 2008. The MC module handles data acquisition and network communication, while the alt-az controller performs positioning and motion control of the altitude-azimuth mount. A PC hosts the command processor software module that acts as the human-machine interface, interpreting human-readable statements into machine-readable binary packets.

6. ACKNOWLEDGEMENTS

The authors would like to thank the following individuals for their contributions to the development of the IRMA project: P. A. R. Ade, B. G. Gom, R. R. Phillips, G. J. Smith, G. J. Tompkins, and C. Tucker. The authors also acknowledge financial support from NSERC, ASRA, and the University of Lethbridge.

REFERENCES

Anderson, J. M., et al. 1999, PASP, 111, 737
Ashley, M. C., Brooks, P. W., & Lloyd, J. P. 1996, Pub. Astron. Soc. Australia, 13, 17
Chapman, I. M., Naylor, D. A., & Phillips, R. R. 2004, MNRAS, 354, 621
Kernighan, B. W., & Ritchie, D. M. 1988, The C Programming Language (2nd ed.; Englewood Cliffs: Prentice Hall)
Lee, C., Ade, P. A., & Haynes, C. V. 1996, in Proc. 30th ESLAB Symp., Submillimetre and Far-Infrared Space Instrumentation, ed. E. J. Rolfe & G. Pilbratt (ESA SP-388; Paris: ESA), 81
Naylor, D. A., et al. 1984, PASP, 96, 167
———. 2003, Proc. SPIE, 4820, 908
Smith, G. J., Naylor, D. A., & Feldman, P. A. 1999, Int. J. Infrared Millimeter Waves, 22, 661
Spillar, E. J., et al. 1993, PASP, 105, 616
Stark, A. A., et al. 2001, PASP, 113, 567
Stephens, W. R. 1992, Advanced Programming in the UNIX Environment (Reading: Addison-Wesley)
Wall, L., Christiansen, T., & Orwant, J. 2000, Programming Perl (3rd. ed.; Beijing: O'Reilly)