



The Baker Observatory Robotic Autonomous Telescope

L. L. Hicks*, M. D. Reed, M. A. Thompson, J. T. Gilker

*Baker Observatory and Missouri State University, 901 S. National Ave., Springfield, MO,
United States*

Abstract. We describe the Baker Observatory Robotic Autonomous Telescope project. The hardware includes a 16 inch Meade LX-200 telescope, an AstroHaven 7 feet dome, an Apogee U47 CCD camera and filter wheel, a Boltwood Cloud Sensor II, and various other minor hardware. We are implementing RTS2 for the Telescope Control System and incorporating custom drivers for ancillary systems.

Keywords : BORAT, autonomous, telescope, robotic, rts2

1. Introduction

With the availability of precision commercial telescopes at a low cost, the opportunity for low cost scientific observatories is within reach of many astronomy departments. Combined with affordable components and inexpensive computers with the power to control many systems, a complete autonomous robotic telescope can be achieved on a small budget, under \$50,000. This paper covers the implementation of the Baker Observatory Robotic Autonomous Telescope (BORAT).

BORAT was designed to meet a set of simple goals. BORAT had to be on a small budget (we were initially thinking < \$25,000). To keep BORAT within budget and for simplicity, most, if not all components had to be available commercially. During component selection, it was a goal to ensure the system could be autonomous. While some devices do not require personnel during operation some devices do require intervention during start up and shutdown. In this process, the last and most important goal is reliability. This is fostered with a quality component selection and simple design. Only time will tell if we have met this goal.

BORAT's purpose will mainly be the observation of variable stars. Observations

*email: lee.lorenzo@gmail.com

of variable stars introduce several requirements. Variable stars are observed by taking many short exposures over an extended period of time. Since pulsating stars can vary over short timescales, it is important to have a CCD that has very short (< 2 s) read-out times. The next requirement is precision tracking. We wish to monitor a star for at least four hours up to all night. The reduction and analysis pipeline requires tracking accuracy with minimal drifting to remain efficient.

Good site selection is important to reduce the effect of sky variations and bad weather to obtain as much data as possible. To keep within our budget a site needed to be within short driving distance, as we could not afford out-of-state trips for set-up and maintenance. So, it was decided that BORAT be located in southern Missouri, USA at our pre-existing observatory. The site encounters all four seasons with extreme temperatures of 0°F to 110°F . A drawback of the site is high humidity levels during summer months. Currently at the site there exists commercially provided Internet via microwave tower. So far this has proven to be slow and unreliable at times. In §4.2 we discuss about required additional hardware and methods to ward off possible offenders.

Even though BORAT's scientific purpose is very specific, it is sufficiently flexible to perform other types of observations. We intend to use it for public outreach, which will likely not be targeting variable stars. Since this extensibility has been coupled with inexpensive off-the-shelf components and using open source control software with good documentation, we hope that BORAT can be used as a template for building a low cost autonomous robotic observatory.

2. Hardware

Each component of BORAT was selected to meet two primary goals. First, each component had to be compatible with Linux (our expertise). This could be in the form of an open specification or driver produced. The second goal is it had to be extendable for the programmers. This meant an open source driver, documented API, or again an open specification. The hardware discussed below covers all the hardware utilized at BORAT excluding the alarm system (see figure 1). This list includes more than just performing observations.

2.1 Telescope hardware

A 16" Meade LX200-GPS with fork mount was selected for several reasons. The LX200-GPS met our primary goals of being Linux compatible with an open specification. The LX200-GPS has a RS-232 communication port to allow computer control. The protocol specification is available including the extended LX200-GPS feature set. The LX200-GPS is also attractive because of its low cost. The LX200-GPS was the cheapest 16" optical tube available. While many others have used German Equatorial

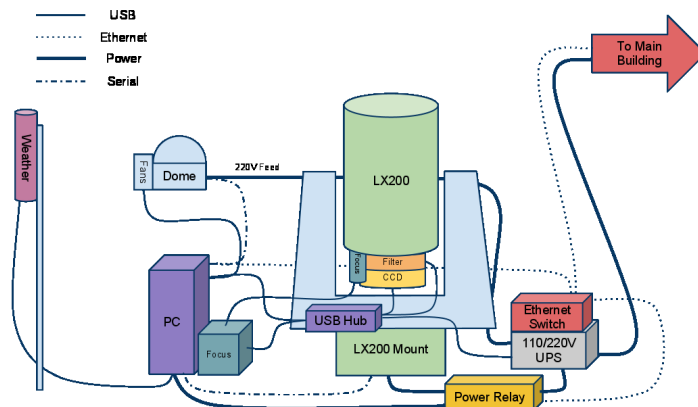


Figure 1. BORAT, Component Layout

mounts, we opted to use the fork mount from Meade. This was largely necessitated by cost, but the compactness of a fork mount and the Equatorial mount's need to switch sides during the night are also advantages. The LX200-GPS has a larger feature set than the classic LX200. One of the important additions is the GPS unit. The GPS unit serves two primary purposes. The GPS unit is reported to give the LX200-GPS very accurate pointing. Since our set-up does not include a tracking camera, initial pointing is very important. A second feature of the GPS unit is obtaining accurate timing. GPS timing combined with Network Timing Protocol (NTP) service should provide accurate timing to within several milliseconds. This is essential for sharing and collaborating data on a secure time frame.

The LX200-GPS optical tube has manual focuser adjustment knobs. To keep project costs down it was decided to power operate the focuser knobs instead of purchasing an automatic (probably in-line) focusing unit. The focuser setup consists of two stepper motors connected to an EZ4Axis logic board. The EZ4Axis was selected because it has a fully documented instruction set and USB connections.

The CCD we will use is the Apogee U47 with an AFW50-9R filter wheel. Both units are connected via USB and have open source drivers written by Random Factory (<http://www.randomfactory.com>).

2.2 Site hardware

During dome selection a clam shell design was sought-after based on ease of use and fewer moving components. The clam shell Astrohaven Enterprises (www.astrohaven.com) 7' was selected. The clam shell design allows the dome to be open or closed independent of telescope position. With the clam-shell fully open it provides the tele-

scope with an unobstructed view of the entire night sky and maximizes air exchange for thermalization. The clam shell design does not require rotation coordination between the dome and telescope pointing as traditional slit domes do. The dome connectivity is done with a single RS-232 with a very simple ASCII protocol. Our original dome had nice simple motor design with a four-button panel to manually open or close the dome via a lockable hatch. However a motor burnt out and Astrohaven stopped supporting the motor of our one year old dome. As such, we had to upgrade to a newer motor system, which is now installed. Unfortunately the new system is horribly over-engineered and the simple four-button system (for manual operation in case of computer failure) has been replaced with three touch-pads which are neither well documented nor readily accessible. We now have to cover three new and unnecessary light sources.

The weather in Missouri can change very quickly from clear sky to rain. An affordable all in one sensor is the Boltwood Cloud Sensor II. It contains all the instruments for humidity, wind, dew point, cloud detection, visual light, and rain detection. The sensor also includes contacts to wire the dome directly for full dome operation without the computer for emergencies. The Boltwood is connected via USB and comes with open source drivers. Our hope is that this sensor will provide all of our needs, though future plans will be to implement two for redundancy.

To keep costs down our design does not include an HVAC system, but a simple ventilation system that balances the internal dome temperature with ambient air temperature. This was not only a low cost solution, but with balanced temperature no excess heat could alter the seeing while observing. This was implemented with two 120 mm computer fans, with HVAC filters, driven by a T-Balancer BigNG (www.t-balancer.com) controller that included temperature sensors. The BigNG is connected by USB and comes with open source drivers.

In the event of a failure, devices need to be power cycled to recover. Because the server is the heart of BORAT and in the event of a hard lock we would like the ability to remotely power cycle the server. A solution was found with a product called Web Switch III by Digital Loggers. It is a network accessible remote power relay that includes scripting and watchdog features. The Web Switch also has an additional role in being part of our alarm system. Additional details are described in following sections.

The nature of a remote site is unstable with the possibly of dirty power supply provided by the utility company. Two UPS units were purchased for the dome and electronic systems. Both UPSs have computer interfaces via USB or RS-232. Both UPS units are also supported by NUT (www.networkupstools.org/). NUT will be discussed in section 3.3.

One major component we have failed to detail is the server. We are currently using a standard desktop computer. During development, new requirements have come

to light, including needing multiple PCI slots and additional RS-232 ports. At the end of BORAT development, when we know port needs, a weather-hardened server, will replace the current one.

3. Control system

3.1 RTS2

There exists many commercial Telescope Control systems for various hardware configurations. By nature of being closed source, proprietary software, these systems did not support all of our devices and are not flexible enough to add these devices by a local programmer. In the spirit of low cost and open development, an open source control system called RTS2 was selected. RTS2, written by Petr Kubánek, is primarily designed for UNIX based systems. Originally designed for GRB observations RTS2 can support seemingly any sort of observation desired (including our program of long-duration, short integration observations).

The software RTS2 was selected for several reasons. As mentioned above, it is open source code which allows additions and modifications when necessary. This is extremely important for adding support for devices such as the Boltwood Cloud Sensor or the BigNG fan controller. These components do their functions very well, but can possibly be overlooked by commercial platforms due to low popularity. Since RTS2 is open source once we added support in RTS2 we committed the code, thus extending the supported hardware RTS2 comes with for other users.

Another reason RTS2 is so attractive is the architecture and message passing of the different processes. RTS2 is designed with a central server and client processes that performs tasks for each device such as a telescope and/or cloud sensor. This architecture separates server and client processes that prevent a failure from propagating upward crashing the entire system. This fault tolerance is very important for autonomous systems that can detect and recover from errors. This became very important during the development of additional hardware and testing.

Last primary reason RTS2 was selected is all the features built in. Part of the hardware utilized by BORAT is already supported in RTS2. RTS2 also has features for advanced observation scheduling, multiple interfaces for communication, scripting, tracking corrections via Astrometry.net (astrometry.net), and support for other services such as NUT.

3.2 Nagios

Nagios is an open source monitoring suite that is used in network environments. It is primarily used for monitoring critical network appliances such as switches, routers,

and servers. Nagios also has an advanced notification system that can alert personnel via SMS, email, and SNMP.

For BORAT, Nagios is implemented on a server located at Missouri State University. It currently monitors the switch and server at the remote site via PING and SSH. Additional network devices such as IP cameras and WebSwitch III will be supported.

The power of Nagios is the ability to write custom tests. Nagios includes support for monitoring internal properties not publicly available on the network. Such statistics as number of logged in users, CPU load, monitoring processes or daemons, and available disk space. With this ability, plans are in place for writing Nagios tests to monitor RTS2 logs and processes. If any critical errors and required processes are not found, Nagios will generate an alert thus generating a notification sent to personnel. This monitoring process is crucial in ensuring observational up-time and protecting equipment from damage.

3.3 NUT

NUT is another open source suite that supports many models of UPSs. It gives an interface to communicate and poll the UPS. This is used by RTS2 to be notified if power failure occurs. Once RTS2 has been notified by NUT that power has been disconnected, RTS2 will commence shutdown procedures. This includes parking the telescope and closing the dome. While simple in explanation, it is an important service because it allows RTS2 to make necessary shutdown steps to protect the equipment faster than personnel can respond to the Nagios notification.

4. Still to do

The integration section covers challenges and difficulties in constructing and implementing BORAT. BORAT is not 100% functional at the time of writing so this is not an exhaustive list.

4.1 Custom drivers

As mentioned earlier, RTS2 did not support some of our components but allows a programmer to add support. This is done by determining which category the device falls under. Most of our components are information based and require simple polling. This falls under the sensor category. Component support can be added by inheriting a device class, in this case the sensor class, and redefine a set of required functions. Since all the components had open source drivers or simple protocols the implementation of unsupported components is an easy task.

4.2 Alarm system

BORAT requires an alarm system to ward off vandalism or theft. An alarm package to cover our requirements would be expensive. This led us to assemble an alarm package with multiple components. The dome will be surrounded by a 6' privacy fence, thereby isolating our external environment. The alarm system then only needs to monitor within the fence. If activity is detected, a loud audible warning will be played and bright flood lamps will be activated to ward off persons, scare away animals, and provide lighting for our cameras. This setup is broken into two smaller systems.

The primary method for detecting unwanted activity is an array of motion sensors placed around the dome. With a range of 10 feet it strikes a balance of detecting positive matches such as a person near the dome while minimizing false detections such as wild life. The motion sensors are wired into a series and placed around the dome. An Arduino (<http://www.arduino.cc/>) board is programmed to detect a voltage drop and notify the server via USB.

Our video surveillance is a combination of CCTV cameras combined with IP cameras. This serves two purposes. CCTV cameras are primarily external of the dome and for security surveillance. Our IP cameras serve a more general role as monitoring telescope operations but are also connected with the CCTV cameras to aid in recording inside of the dome during an alarm. The CCTV cameras are driven by a capture card inside the server.

An open source camera suite called ZoneMinder (www.zoneminder.com) is the heart of the surveillance system. ZoneMinder suite includes primitive operation such as monitoring and recording. The power of ZoneMinder is the detection of activity on any CCTV live feeds. ZoneMinder is keyed to detect small amounts of pixel change in cameras to determine activity. Any activity will put ZoneMinder in an ALARM state starting frame by frame recording and noting activity in logs. Since Nagios is already implemented it will generate a notification via a custom Nagios plugin detailing which zone detected activity.

During nighttime observations, the CCTV camera is dependent of the motion sensors. The procedure of detection follows

1. The Arduino will detect a voltage drop in the motion detectors.
2. The Arduino will send a detection message to the server.
3. The server will respond by notifying RTS2 of the alarm.
4. RTS2 will stop any observations, noting bad exposures in FITS header and close dome immediately.
5. The server will send a ON message to the WebSwitch powering the flood lamps.

6. ZoneMinder will detect the sudden extra light and begin monitoring CCTV feeds for any activity and begin recording.
7. Nagios will detect alarm by monitoring ZoneMinder logs thus generating a notification.

This system provides a complete monitoring solution during observations and idle night periods. The alarm system will also be active during the day by a much simpler method. When the dome is closed during daytime a magneto switch, placed where two leafs meet, will be in a OPEN state. Any opening of a leaf by force or unauthorized opening of the dome will set the switch in a CLOSED state notifying the server via Android. Nagios notifications will be generated along with ZoneMinder recordings.

5. Anticipated outcomes

At the time of writing, BORAT had not taken the first light. If testing succeeds, BORAT will be a fully autonomous telescope capturing up to 2,000+ images per night without any human intervention. To date BORAT has cost $\approx$ \$60,000 independent of labor. Our intention is to provide a template for constructing an autonomous observatory for $<$ \$50,000 which may be used by other institutions. Over time, we will also test the durability and reliability of the Meade fork mount and other ancillary systems. BORAT includes some unique low-cost solutions and we have yet to determine if they are functional.

5.1 Data pipeline

An autonomous telescope system can capture vast amounts of data. All this raw data has to be processed before scientific analysis can be performed. Normally this processing is done manually or using a pipeline developed to reduce processing times. Since BORAT is an autonomous replacement for an existing scientific program, a reduction pipeline is already being utilized, written by Lee Hicks and Matthew Thompson.

The pipeline is divided into two parts, reduction and analysis processing. The reduction portion still has several steps which require human interaction, but it is possible to make more improvements. Better image header information will alleviate several of these interventions. Several additions can still be made to further streamline the entire reduction pipeline. The analysis pipeline does not require manual steps (Thompson et al. 2011).

6. Conclusions

BORAT has been planned and built to be a low cost autonomous robotic observatory. This has proven to be challenging in different ways. The first challenge has been component selection. Each component that interfaces with the server has to be compatible with Linux. The second requirement is an open protocol or documented API. This greatly reduces the number of available components, thus limiting options. Once the components arrived the next challenge became integrating all the components to work with RTS2. For some components it was fairly straight forward and for others not quite as simple.

The next challenge is receiving notifications of certain events. Without notifications, the autonomous system can crash or malfunction posing unnecessary risk such as bad weather or vandalism. This issue may be solved by implementing Nagios. Nagios should provide a notification transport system that can be customized for BORAT.

The last major challenge is designing an alarm system to protect the entire system. A standard alarm system was not applicable due to certain constraints (especially cost). By combining several components a two step alarm system can protect and play defensive and offensive roles. The extensibility of the alarm system also allows remote viewing and instant email or SMS notifications via Nagios.

Once these challenges are overcome, BORAT will become a fully autonomous telescope gathering data on clear nights over several months. BORAT combined with an optimized data pipeline will provide astronomers with a wealth of data.

Acknowledgments

This project is a partnership between Missouri State University and Krakow Pedagogical University. Financial support has come from the Missouri Space Grant Consortium, funded by NASA, the College of Natural and Applied Sciences, and Krakow Pedagogical University.

References

Thompson M. A., Reed M. D., Gilker J. T., et al., 2011, Searching for Variable Events in High-Cadence Data, http://tfa.cfht.hawaii.edu/papers/thompson_tfa_paper.pdf