Design of remote control software of near infrared Sky Brightness Monitor in Antarctica

Zhi-yue Wang, Ya-qi Chen, Ming-hao Jia, Guang-yu Zhang, Jun Zhang, Yi-hao Zhang, Jin-ting Chen, Hong-fei Zhang, Peng Jiang, Tuo Ji,Jian Wang*

State Key Laboratory of Technologies of Particle Detection and Electronics, Modern Physics Department, University of Science and Technology of China

*Email: wangjian@ustc.edu.cn, IEEE Senior Member

1.Introduction

A lot of astronomical observation devices will be delivered to the Antacrtica due to the superior observe condition, which makes a demand of long-duration, accurate measurements of sites condition. We developed Near-Inrared Sky-brightness monitor(NIRBM) to measure the near-infrared radiation of sky at astronomy observatories, the monitor is shown in Fig.1. However, the harsh environment of observatory make it important to develop the automatically remote control software for NIRBM.

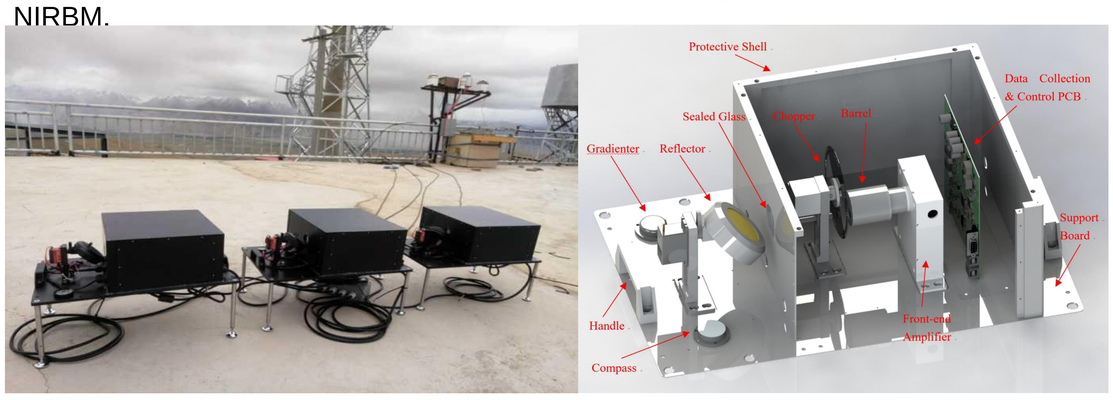

Fig.1. NIRBM at Ngari Test Spot

The NIRBM mainly consists of an InGaAs detector with TEC cooling, a chopper and a reflector. The InGaAs detector converts infrared radiation to electrical signals. The reflector can be controlled to rotate to scan the sky from 0° to 180°. By using the method of chopper modulation and digital lock-in amplifier processing, the Signal Noise Ratio(SNR), detectivity and the data acquisition speed of the device is greatly improved. Electromechanical control and weak signal readout functions are accomplished by the same circuit system, in which the core chip is a STM32F407VG microcontroller.

In order to control NIRBM remotely, a software control system based on EPICS(Experimental Physics and Industrial Control System) and the Tornado web framework has been designed. For the purpose of reusability and scalability, our software control system is divided into three levels: hardware driver, integrated control and user interface. The main architecture is shown as Fig.1. The whole control system is developed and implemented under Linux.

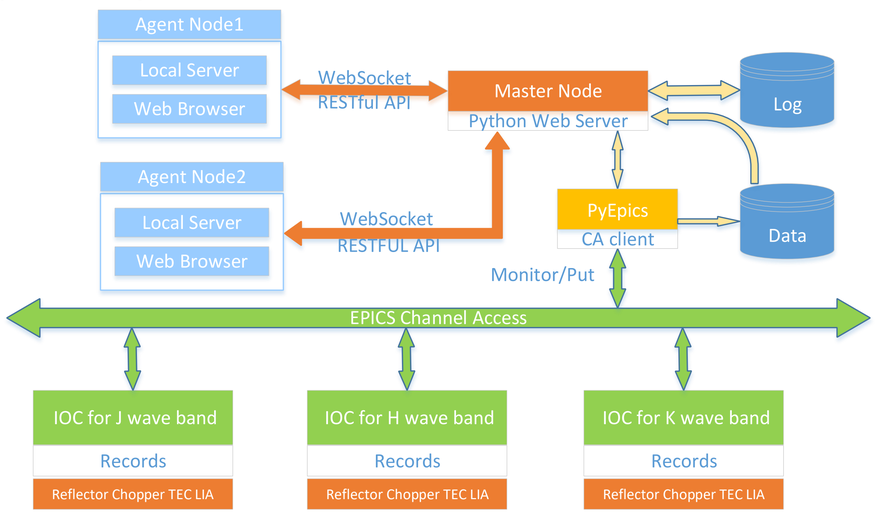

Fig.2. Diagram of Software Structure

2. System Structure

In the device driver layer, each hardware module is controlled independently by the corresponding EPICS IOC (Input Output Controller). The properties of the devices were abstracted and described as several Epics Process Variables. To describe the overall status of our devices, each device also has a status code and a more detailed status string that can be monitored by the control software. The IOC program communicates with the STM32 microcontroller via serial port and implements the basic functions of our device, including updating device status information and sending commands to perform specific operation. In this layer, our software system can control the components of the device separately. In order to cope with emergencies, remote upgrade of the STM32 microcontroller firmware is also implemented to this layer.

In the integrated control layer, Tornado, the web server framework is combined together with PyEpics and psycopg for development. As a client of the EPICS framework, PyEpics is capable of monitoring or changing the Process Variables which are used for describing the status of the device or for sending instructions to the device. Psycopg is the python client library for PostgreSQL, which is used to store the observation and temperature data. The Tornado server operates the EPICS IOC , communicates with the database system as well as exposes interfaces to the User Interface(UI) layer to make calls, we can regard the Tornado server as a connector between IOC, database and user interface.

In order to implement automatic observation, we designed a finite state machine involving seven states shown in Figure 2. The attributes of the observation schedule include start and end time, range of angle to scan, and the time constant of the amplifier which is stored in a series of variables. Users can configure the observation schedule by changing these variables. The finite state machine was driven by an additional thread in tornado so that HTTP request handlers will not be blocked.

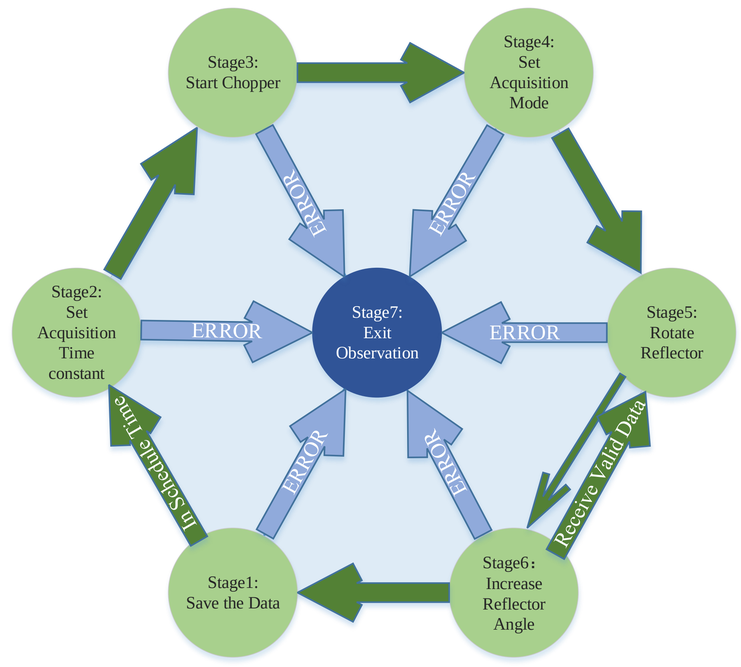

Fig.2. Diagram of Auto Observe Finite State Machine

3.Master-Agent Mode

Due to the high latency and low bandwidth of the Antarctic network environment, the tornado back-end is designed as a kind of master-and-agent architecture, as shown in Fig.3. The master and agent nodes communicate with each other through the WebSocket protocol. The master node is deployed in the Antarctic, controlling our device directly, and the users at Antarctic can visit the front-end websites directly from master nodes. Agent nodes will be deployed in the domestic servers for domestic users. Under such condition, the front-end webpages will be served directly by the agent nodes, while other dynamic HTTP requests which needs to operate on the device will be sent forward to the master node in Antarctic. All this procedure largely reduced the bandwidth usage of networking, therefore user experience for domestic users in China is improved.

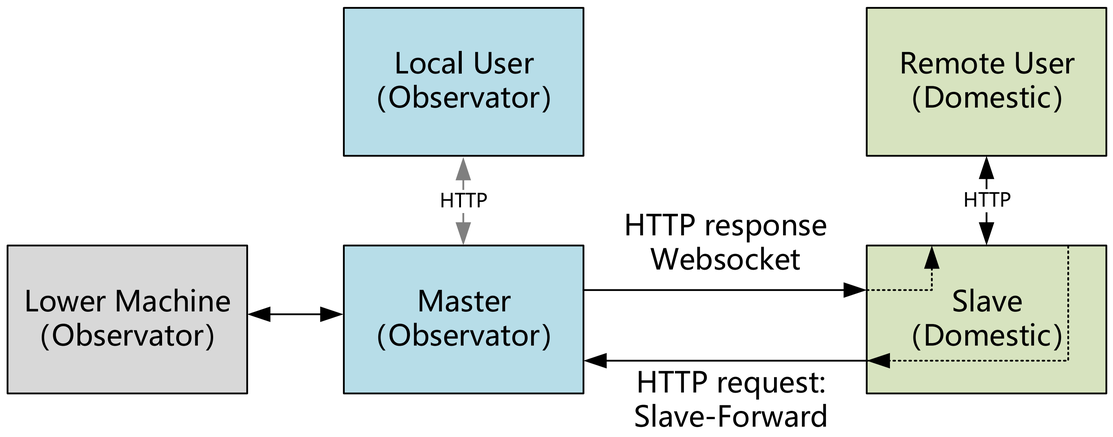

Fig.3. Block diagram of Master-Agent mode

User interface is implemented in the form of single-page application, with a front end MVVM framework called Vue used. The web page shows the data pushed through WebSocket in real time, and send control commands to the Tornado backend via AJAX requests. The web page integrates device control, data curve drawing, alarm display, auto observation and other functions together.

Bibliography

1. Jia M, Chen Y, Zhang G, et al. A web service framework for astronomical remote observation in Antarctica using satellite link[J]. Astronomy and Computing (2018).
2. Dong S, Wang J, Tang Q, et al. Design of a multiband near-infrared sky brightness monitor using an InSb detector[J]. Review of Scientific Instruments, 89(2): 023107 (2018).
3. Tian Q, Jiang P, Du F, et al. The bright star survey telescope for the planetary transit survey in Antarctica[J]. Science bulletin, 61(5): 383-390(2016).
4. Kraimer M R, Anderson J B, Johnson A N, et al. EPICS application developer's guide[J]. February 2010, http://www. aps. anl. gov/epics (2009).
5. Dory M, Parrish A, Berg B. Introduction to Tornado: Modern Web Applications with Python[M]. " O'Reilly Media, Inc." (2012).
6. [6] Momjian B. PostgreSQL: introduction and concepts[M]. New York: Addison-Wesley (2001)