An Extended Postprint of a WeChat Mini Program-Based Control Framework for RTS2

Authors: Liang Bo, Tian Zhiyan, Wang Feng, Deng Hui, Wei Shoulin

Date: 2018-05-28T00:00:00+00:00

Abstract

Autonomous telescope control is an important component of modern astronomical observation technology. Among current mainstream autonomous control systems, the open-source RTS2 features a modular and plug-and-play design philosophy, along with rapid response capabilities and stable operation characteristics, and is widely applied in autonomous control systems for astronomical telescopes. Since RTS2 is based on the Linux platform and primarily relies on CLI for remote access control, it imposes relatively high requirements on observers. Through in-depth analysis of the RTS2 system and appropriate modification of its JSONAPI, using JSON as the data transmission format and the WeChat application on mobile terminals as the carrier, cross-platform data access and function invocation for the RTS2 astronomical telescope control system are achieved. By utilizing WeChat Mini Programs, the control system is transplanted into WeChat Mini Programs, enabling astronomical technology researchers to conveniently and efficiently use mobile terminals on the WeChat platform to remotely control astronomical telescopes and monitor the status of autonomous telescope control systems in real time. Adopting this model, it can be extended to other autonomous control architectures such as ASCOM, thereby achieving a universal mobile terminal remote control based on WeChat Mini Programs.

Full Text

A Control Framework Extension for RTS2 Based on WeChat Mini-Programs

Liang Bo, Tian Zhiyan, Wang Feng, Deng Hui, Wei Shoulin
Key Laboratory of Computer Technology Application of Yunnan Province, Kunming University of Science and Technology, Kunming 650500, China

Abstract

Autonomous telescope control constitutes a crucial component of modern astronomical observation technology. Among current mainstream autonomous control systems, the open-source RTS2 (Remote Telescope System, 2nd version) stands out for its modular, plug-and-play design philosophy, rapid response capabilities, and stable operation, making it widely adopted in autonomous telescope control systems. However, since RTS2 is Linux-based and primarily accessed via command-line interface (CLI) for remote control, it imposes relatively high technical demands on observers. This paper presents a thorough analysis of the RTS2 system with moderate modifications to its JSON API, employing JSON as the data transmission format and leveraging the WeChat mobile application as a carrier to enable cross-platform data access and functional invocation of the RTS2 telescope control system. By porting the control system to a WeChat mini-program, astronomical researchers can conveniently and efficiently control telescopes remotely and monitor the status of autonomous telescope control systems in real-time using mobile terminals on the WeChat platform. This model can be extended to other autonomous control architectures such as ASCOM, thereby realizing a universal mobile terminal remote control solution based on WeChat mini-applications.

Keywords: RTS2; JSON; WeChat Mini-Program

1. Introduction

RTS2 is a remote astronomical telescope control system based on the Linux operating system, designed to achieve fully automated telescope control. Leveraging its open-source nature, astronomical researchers can easily obtain the source code from GitHub and install and configure RTS2 on Ubuntu systems following the provided documentation. While RTS2 includes CLI tools such as rts2-mon for system control, these tools can only run on Linux machines where the RTS2 software packages and dependencies are installed. The widespread use of mobile terminals and the expanding work-life radius of researchers have created demand for mobile office and remote monitoring capabilities, making RTS2' s native tools extremely inconvenient and sometimes entirely impractical.

For telescope observation and monitoring personnel, the ability to view real-time operational status, alarm information, and send control requests is essential. In recent years, domestic research on control extensions for autonomous astronomical telescope observation systems has yielded several achievements. In RTS2, researchers can access the rts2-xmlrpc service through port 8889 for network-based monitoring and status viewing [1-4]. By improving the XMP-RPC interface and using XML-formatted data, astronomical technicians have enabled browser-based operation of RTS2 systems. However, XML format is complex, consumes significant bandwidth during transmission, and requires substantial CPU resources for parsing, making it unsuitable for lightweight environments. Alternatively, some have implemented services by inheriting the Rts2 Client

Core class and developed browser pages to monitor and control RTS2 via WebSocket [3-4]. Both approaches aim to leverage the lightweight nature of browsers to overcome traditional CLI limitations, but they require users to remember and manage URLs, which is cumbersome and inconvenient for telescope monitoring and control tasks.

With the development of mobile terminals and the widespread adoption of WeChat, transplanting telescope monitoring and control interfaces into WeChat mini-programs would greatly simplify real-time telescope system control for observers. This paper presents research conducted against this background. Building upon existing studies, we employ HTTP protocol for JSON data interaction with WeChat mini-programs to achieve monitoring and control of the RTS2 autonomous control system.

1.1 WeChat Mini-Programs

WeChat is a free instant messaging application launched by Tencent in 2011 for smart terminals. In recent years, Tencent has introduced WeChat mini-programs [5]—applications that can be used without download or installation. Users can access them by scanning a QR code or searching, and simply exit when finished without needing to uninstall [6]. These programs offer convenient acquisition and dissemination within WeChat along with excellent user experience. WeChat supports various mobile operating systems, allowing mini-program developers to focus on functionality without considering whether the terminal runs Android or iOS, though users must upgrade WeChat to version 6.5.3 or above.

Mini-program development resembles HTML interface creation, comprising primarily index.wxml (interface), index.wxss (styles), and index.js (logic) files, which interact with the developer' s server through WeChat-provided APIs. The data interaction flow between WeChat mini-programs and RTS2 is illustrated in [Figure 1: see original paper].

[Figure 1: see original paper]

Fig 1. The WeChat Mini-Program access to RTS2 flow chart

Each access to RTS2 services from a mini-program requires first contacting the WeChat server. Therefore, service calls must use the encapsulated `wx.request(OBJECT)` method, which facilitates data exchange with RTS2 services through the WeChat server and returns JSON data to the mini-program. Due to mini-program interface restrictions prohibiting port numbers in URLs, accessing custom servers requires mapping and hiding the port number in observatoryserver to enable proper interaction between the mini-program and RTS2.

1.2 Data Processing

WeChat mini-programs interact with RTS2 services via HTTP requests, consistent with network-based RTS2 access. Based on various JSONAPI commands, mini-programs can access RTS2 services through HTTP addresses, with RTS2 returning corresponding data in JSON format. To avoid Chinese character encoding issues, both mini-programs and RTS2 services must use UTF-8 encoding for all data interactions.

For example, accessing the address `http://observatoryserver/apipath/jsoncommands` with appropriate parameters retrieves JSON data such as `{infotime:1502882474.90689,uptime:1502882474.052562,focuser:"F0"}`. In the mini-program, jQuery facilitates easy retrieval of this JSON data. The timestamp value can be obtained via `obj.infotime` and requires conversion using the following code:

```
var day = new Date(parseInt(time) * 1000);
var date = "." + time.split(".")[1];
time = day.toLocaleDateString().replace(/\//g, "-") + "T" + day.toTimeString().substr(0, 8)
```

Since different JSON commands return different data, a universal method was implemented in the mini-program to transform RTS2' s returned JSON data by separating original key-value pairs into a new JSON format. The core code is as follows:

```
var json = "{'list':[";
var i = 0;
for(var k in devices){
    if(i>0) json +=",";
    json +="{'name':'"+k+"','values':'"+devices[k]+"'}";
    json +="]}";
    var obj = eval('('+json+')');
}
```

The aforementioned JSON data is transformed into `{list:[{key: infotime, value: 1502882474.90689},{key: uptime, value: 1502882474.052562},{key: focuser, value: "F0"}]}` format. This JSON structure is easy to read, write, parse, and generate, while effectively improving network transmission efficiency.

2. Implementation

2.1 JSON API Commands

Analysis of the latest RTS2 source code on GitHub reveals numerous commands for retrieving device information and controlling equipment via browsers or mobile devices. Table 1 lists the available commands in the JSON API.

Table 1 The JSON API provides commands

change_constraints	cnst_alt_v	executor	night	message
change_script	cnst_time	devbytype	expose	obytid
labellist	satisfied	status	consts	deviceinfo
labels	script	sunalt	cnst_alt	create_target
devices	hasimages	runscript	lastimage	selval
taltitudes				

These commands enable remote control of autonomous astronomical telescope systems and remote viewing of device control and status information, as well as acquisition of astronomical images. The following describes the primary commands used in this work.

(1) Control Commands

a. Direct Control (cmd)
The `cmd` command sends control commands to different devices in RTS2 based on `device` and `target` parameters. For example, to observe the next target after target 8, the address `http://observatoryserver/apipath/cmd?d=EXEC&c=next%208` sends the control command (note that spaces in scripts are replaced with '%20' to comply with HTTP URL standards).

b. Script Control (runscript, killscript)
Mini-programs can run scripts using the `runscript` command and terminate running scripts using `killscript`. Sending the control information `runscript?d=C0&s=E%201` to RTS2 executes a prepared script (again with spaces replaced by '%20').

(2) devbytype
The `devbytype` method queries device lists by type. Table 2 lists the available equipment type codes.

Table 2 Equipment type

FOCUS	MIRROR	CUPOLA	AUGERSH	SENSOR
EXECUTOR	IMGPROC	SELECTOR	SCRIPTOR	MOUNT
WEATHER	ROTATOR			

(3) devices
The `devices` command retrieves a list of all devices connected to RTS2, described in JSON format.

(4) get
Combined with a device name parameter, the `get` command retrieves specific data for a connected device, such as status and position details, generating JSON format data returned from RTS2 to the mini-program for parsing and display by observers.

(5) lastimage, currentimage
The `lastimage` command retrieves the most recently captured image, while `currentimage` obtains the currently capturing image.

2.2 Implementation of RTS2' s WeChat Mini-Program Terminal Control System

Based on the aforementioned technologies, we implemented a prototype system on a mobile terminal using a WeChat mini-program and open-source RTS2. The system enables telescope system status viewing, connected device status and information monitoring, and captured image review. The prototype uses RTS2 deployed on Ubuntu 16.04 as the server, an Android 6.0 phone with WeChat version 6.5.10, with the interaction flow illustrated in [Figure 2: see original paper].

[Figure 2: see original paper]

Fig 2. The WeChat Mini-Program function diagram

After publishing the mini-program to WeChat through the development tools, the `devices` command retrieves the connected device list, allowing users to view device status and information as needed. The interface for displaying device information, status, and performance data is shown in [Figure 3: see original paper].

[Figure 3: see original paper]

Fig 3. Device information, status and performance

As shown in [Figure 3: see original paper], the mini-program sends JSON-encapsulated commands to retrieve basic information about connected devices. WeChat mini-programs' built-in debugging and performance monitoring features indicate a CPU usage of only 1% and memory consumption of just 151MB, demonstrating low hardware requirements. Additionally, due to the mini-program' s instant-use nature without requiring download or installation, it consumes no system storage space. In practical use, observers can view connected device information and captured astronomical images anytime and anywhere, while also controlling devices through JSON commands.

3. Conclusion

As an open-source astronomical telescope control system, RTS2 has been extensively studied and utilized in the astronomical community. Previous extensions have enabled real-time control and status querying via XML-RPC [1] and WebSocket [3-4] protocols through browsers. By leveraging the widespread adoption of WeChat as a carrier, this work provides a WeChat mini-program solution that enables observers or duty personnel with WeChat clients to view connected device status and obtain telescope-captured images on mobile terminals regardless

of the phone' s operating system. Although this autonomous control system targets RTS2, it can be extended to the ASCOM framework through a JSON Proxy Server, achieving a universal remote control solution for autonomous telescope control systems based on WeChat mini-programs.

Overall, the universal telescope control WeChat program design implemented in this work satisfies observational requirements for viewing and controlling connected device information. Since the current implementation is based on HTTP protocol with active retrieval mode, it lacks real-time capability compared to WebSocket. Future work will explore technical improvements to address this limitation.

References

[1] Ran Fanhui, Deng Hui, Liang Bo, et al. A study of remote access techniques for an RTS2 autonomous observation software system based on the XML-RPC[J]. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 2013, 10(4): 372-377.

[2] Chen Junyi, Chen Dong, Fan Yufeng, et al. The remote control experiment of the telescope based on the embedded Internet[J]. Astronomical Research & Technology—Publications of National Astronomical Observatories of China, 2007, 4(1): 36-41.

[3] Xu Jun, Jin Zhenyu. Remote control of astronomical terminals based on WinSockets[J]. Publications of the Yunnan Observatory, 2001(1): 40-46.

[4] Wei Shoulin, Cao Zihuang, Wang Feng, et al. A study of web control of an RTS2 system based on the WebSocket[J]. Astronomical Research & Technology —Publications of National Astronomical Observatories of China, 2014, 11(4): 404-409.

[5] Kuang Wenbo, Li Rui, Ren Zhuoru. Face view micro-program[J]. News Tribune, 2017(2): 15-18.

[6] Wang Tianni. When library encounters WeChat mini-apps[J]. Library and Information, 2016(6): 83-86.

Funding: This work was supported by the National Natural Science Foundation of China (U1631129, U1531132, 11403009, 61462053, U1231205, 11203011) and the Yunnan Provincial Basic Research Project (2017FB001).

Received: August 20, 2017; **Revised:** October 10, 2017

Author Introduction: Liang Bo, male, engineer. Research interests: computer applications, astronomical instruments and methods. Email: lb@cnlab.net.

Corresponding Author: Tian Zhiyan, male, M.S. Research interests: computer applications. Email: tianzhiyan@cnlab.net.

Note: Figure translations are in progress. See original paper for figures.