第56卷 第2期 天 文 学 报 Vol.56 No.2
2015年3月 ACTA ASTRONOMICA SINICA Mar., 2015

doi: 10.15940/j.cnki.0001-5245.2015.02.009

近地天体望远镜的控制软件系统*

王 歆[1,2†] 赵海斌[1,3] 夏 炎[1,3] 陆 昊[1,3] 李 彬[1]

(1 中国科学院紫金山天文台 南京 210008)
(2 中国科学院空间目标与碎片重点实验室 南京 210008)
(3 中国科学院行星科学重点实验室 南京 210008)

摘要 配合CCD相机更新, 2013年近地天体望远镜(CNEOST, China Near Earth Object Survey Telescope)对硬件系统进行了改造, 在此基础上重新设计并实现了控制软件系统, 软件采用了基于WebSocket协议的消息机制, 具有良好的扩展性; 基于Web的用户界面, 实现了近地天体望远镜在不同用户终端下的远程观测. 自投入使用以来, 系统运行稳定, 大幅度提高了设备运行效率, 降低了操作复杂度; 同时也为未来类似的望远镜或望远镜云的系统设计提供了一次有益尝试.

关键词 望远镜, 方法: 实测

中图分类号: P111; **文献标识码**: A

1 引言

近地天体望远镜位于中国科学院紫金山天文台盱眙观测站, 是我国最大的折射式施密特望远镜, 主要开展近地天体巡天观测. 由于其出色的探测能力, 投入运行以来已开展了多个学科方向的实测工作.

近地天体望远镜设计于上世纪90年代, 那时互联网在我国刚刚开始普及, 主要还是作为资讯获取的一种新手段, 远没有今天这样应用在生产生活的各个方面. 近地天体望远镜原有的构架中没有使用网络, 各个分系统分别按照事先制定的运行计划执行, 分别记录运行状态, 观测完成后将数据人工拷贝集中后, 按时间标记将各分系统数据逐一对应后形成最终的观测数据, 不能进行实时数据融合.

这种方式由于必须依赖事后处理, 无法实时生成完整的观测数据, 对数据处理时效产生一定影响; 另一方面各个分系统只能按照计划执行, 无法灵活调整, 特别是由于各分系统之间是相关联的, 一个分系统的运行故障会导致另一个分系统计划失效, 故障排除后需要对所有相关分系统进行重置.

2013年为了提高探测能力, 近地天体望远镜将4 K × 4 K的CCD相机SI600S更换为目前最大的单芯片(10 K × 10 K) CCD相机STA1600LN. 为了配合相机升级, 镜筒机械

2014-06-27收到原稿, 2014-09-11收到修改稿

*国家自然科学基金项目(11273067)、江苏省自然科学基金项目(BK2011890) 和小行星基金会资助

†wangxin@pmo.ac.cn

部分需要进行相应改造. 借此契机, 对镜筒分系统和时钟分系统进行相应的调整, 以支持全系统的网络化改造. 为了提高近地天体望远镜的运行效率, 在硬件改造的基础上, 对近地天体望远镜的控制软件系统进行了重新设计, 实现了一套完全基于网络的控制软件系统, 系统实现了各分系统之间的实时联动, 可通过互联网在统一用户界面下对各个分系统进行操作, 实现了近地天体望远镜的远程化和程控化观测.

2　硬件系统

2.1　原有系统构架

近地天体望远镜包括4个分系统: (1)伺服分系统, 控制望远镜的定位、运动以及圆顶的随动; (2) 图像采集分系统, 控制CCD相机用于采集观测图像; (3)镜筒分系统, 承担望远镜镜盖开合、调焦和滤光片选择; (4)时间分系统, 为所有分系统提供统一时间. 系统构架如图1.

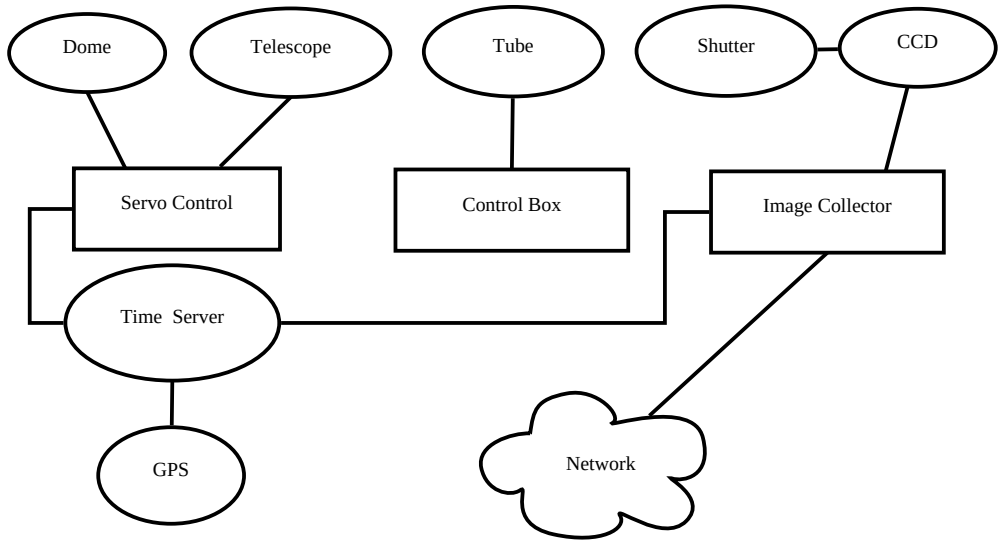

图 1　原有系统构架

Fig. 1　The original system architecture

伺服控制计算机通过专用的伺服通讯卡与望远镜的伺服系统连接, 随动圆顶通过RS232串口与计算机串口连接. 控制计算机根据望远镜指向信息通过串口向圆顶发送运动指令实现随动, 伺服控制计算机没有网络接口因此无法连入网络.

图像采集计算机通过专用连接线与CCD相机连接, 同时CCD相机向快门输出TTL(Transistor Transistor Logic)信号, 实现快门的同步开合. 为了传输图像和获取数据方便, 图像采集计算机是唯一连入网络的计算机.

镜筒控制通过控制盒内的物理开关实现, 不能进行编程控制.

时间分系统由一台时钟分配器构成,通过GPS信号获取标准时间, 通过数据线将时间信号同步给图像采集计算机和伺服控制计算机.

由于各个分系统之间没有任何通信,观测过程中需要分别和各分系统交互, 其中两

个分系统通过2台计算机进行操作, 而对于镜筒的控制需要通过控制盒内的物理开关进行. 实际操作中往往需要同时操作3个分系统并且根据各自反馈不断调整, 因此十分不便.

2.2 系统改造

2.2.1 串口的网络化

为了实现所有分系统之间的通信, 必须从硬件上实现分系统的互联, 根据原有系统结构, 主要是伺服控制计算机和镜筒控制两个部分急需实现网络化.镜筒控制原来没有通信接口只能通过物理开关操作, 这次改造利用串口控制卡给镜筒控制模块增加的串口接口, 可通过串口发送控制指令和接收反馈信息. 而伺服控制计算机由于驱动的原因, 无法通过内置或者外置方式增加网络接口, 只能通过其原生的串口进行通信.

为了实现串口转网口的功能, 我们选择通过串口服务器来实现串口到网口的转换, 这样减少了计算机的投入, 同时也避免了自行编程实现两种协议转化的风险. 串口服务器是一种体积小巧的嵌入式通用硬件设备, 能够将串口转换成TCP/IP网络接口, 实现串口与TCP/IP网络接口的数据双向透明传输, 使得串口设备能够立即具备TCP/IP网络接口功能，连接网络进行数据通信.

我们选用了MOXA公司的NPort 5210串口服务器[1], 该服务器具备两个RS232接口, 我们将其中一个用于伺服控制计算机的对外通信, 另一个则用于镜筒控制. 由于网络的互通性, 不再需要专用的计算机, 网络上的任何计算机都可实现这两个串口的访问.

2.2.2 快门监控系统

近地天体望远镜升级相机后, 配套的相机控制软件无法记录曝光开始时刻, 只能记录图像文件生成时刻, 这与实际曝光时刻相差20多秒, 在后续数据处理中是不能接受的. 经与厂商沟通, 问题定位在相机控制器, 无法通过修改控制软件来解决.

根据这种情况, 我们采用捕获CCD相机发送给快门的TTL电平信号的方式对快门进行监控. 考虑到近地天体望远镜对时间记录要求为几十ms量级, 选择了通用的树莓派(Raspberry PI)微型电脑[2]来捕获TTL电平信号. 树莓派是一款基于ARM构架处理器的微型电脑, 由慈善组织“Raspberry Pi 基金会”开发. 外形只有信用卡大小, 却具有计算机的所有基本功能. 树莓派采用Linux系统, 开放了通用输入输出(GPIO)接口, 可以直接获取TTL电平信号, 树莓派体积小, 很容易安装在镜筒上, 几乎不占用空间. 树莓派本身具有网口, 和普通计算机一样接入网络.

2.2.3 时钟服务器

时间分系统将时间分配器更换为一台时间服务器, 时间服务器通过GPS获取标准时间, 然后使用标准NTP(Network Time Protocol)协议提供对时服务, 所有连网计算机均可通过网络实现时间同步.

2.3 升级后的硬件结构

经过上述改造后, 所有分系统都实现了网络化, 可通过TCP/IP协议访问. 结构如图2.

[1] http://www.moxa.comproduct/NPort_5200A.htm
[2] http://www.raspberrypi.org

所有分系统之间也实现了解耦, 成为扁平的星形结构, 各分系统直接连接到网络, 结构上不受其它分系统的影响. 同时由于所有信息实现了网络共享, 观测系统和数据库与数据处理分系统也通过网络直接连接, 为数据的实时发布和实时同步处理奠定了基础.

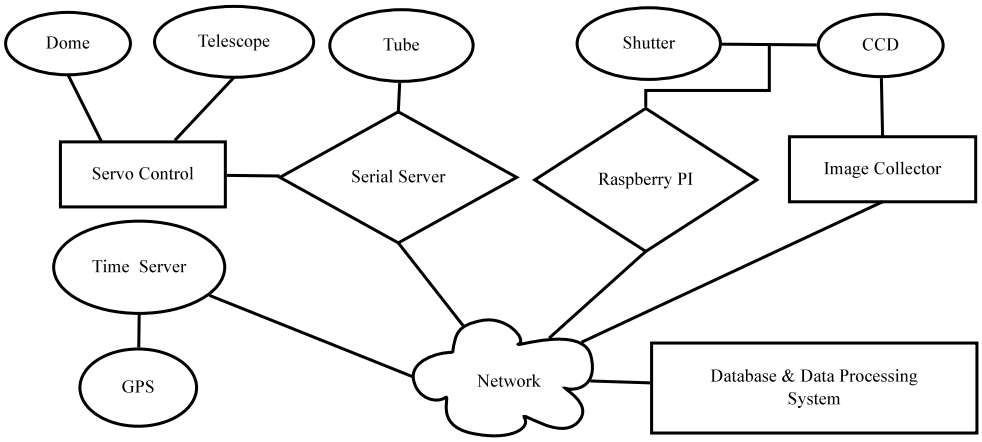

图 2　改造后的硬件系统结构

Fig. 2　The system architecture after reform

3　软件构架设计

望远镜的观测软件通常被称为“主控”软件, 因为它提供各个分系统控制的统一界面, 根据需求协调各分系统的协同工作以及和观测员的交互, 起到观测要求和观测结果数据的输入和输出作用. 观测软件被称为“主控”软件的另一层次原因是现有观测软件多是单一程序形式, 主控软件是一个独立运行的程序, 承担所有分系统的控制、反馈以及之间的协调, 人机交互和输入输出也都汇集在其中. 这种模式比较便于各分系统之间的协同调度, 但软件规模往往较大,维护比较困难, 例如任何一个分系统的变化都需要对整个软件进行修改. 这种方式对软件编写要求很高, 需要对各个分系统都比较熟悉, 具备多个分系统的编程能力.

针对上述问题, 我们采用了一种全新的设计— 基于TCP/IP消息传递的星形结构, 软件结构见图3.

星形的中心为中心服务, 中心服务不具备任何实际的设备控制功能, 仅仅起到各分系统之间信息传递的功能以及用户管理的功能. 所有实际功能由功能模块提供, 用户界面也作为一项独立功能实现. 由于和具体控制无关, 中心服务将会比较稳定, 不会随分系统升级改造而需要不断升级. 中心服务和功能模块通过TCP/IP网络实现通信. 通信过程中采用非阻塞方式, 使得中心服务可以并行地和各个模块实现消息收发. 而对于近地天体望远镜的各分系统之间无需高精度的实时同步, 网络时延等对时序并不会产生实质性影响.

各个分系统的实际控制由功能模块提供, 每个功能模块负责一个分系统, 由独立程序实现. 每个功能模块仅仅与中心服务进行双向消息传递, 各功能模块之间不直接通信, 通过中心服务接收和转发系统运行状态与参数, 根据接收到信息具体实现自身功能, 分

系统状态更新也通过中心服务反馈给其它模块.

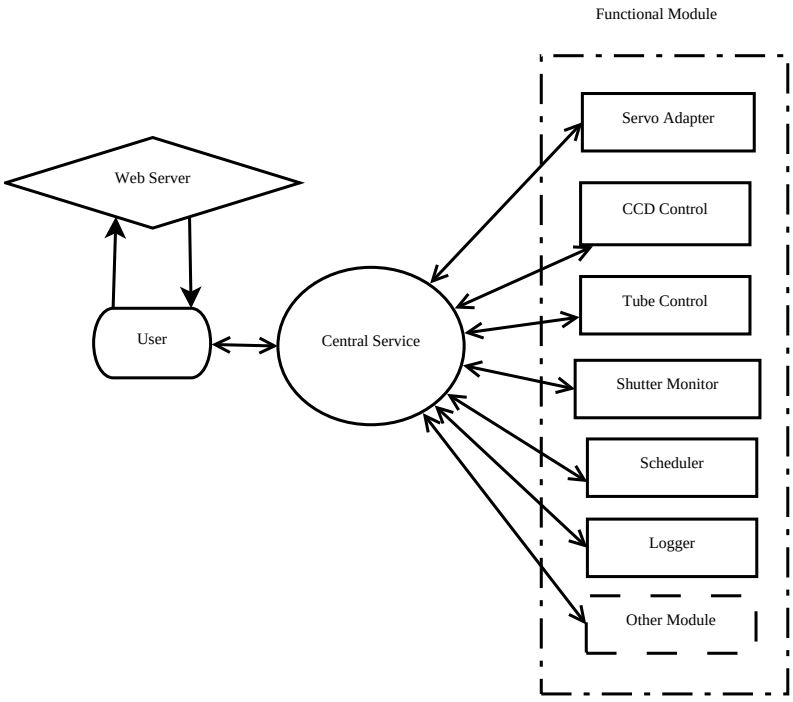

图 3 软件构架

Fig. 3 The software architecture

功能模块并不局限于硬件控制, 分系统协同工作的调度等也都可作为功能模块接入系统发挥作用, 目前状态记录和观测调度策略都作为功能模块在系统中运行.

这种结构使得各部分相对独立, 不受其它模块变化影响, 系统的任何调整都会限制在最小范围, 每个模块只要保持和中心服务的通信协议不变, 则系统其它部分不需要任何调整.

这种结构特别易于维护, 每个程序只做单一的任务, 代码量小, 易于实现和维护, 稳定性高. 同时具有很好的扩展性, 新的功能随时可以独立模块方式增加到系统中. 由于各个模块和中心服务通过网络实现通信, 对网络环境没有特殊要求并不局限于局域网, 特别对于非硬件控制类的功能模块, 完全可以根据需要运行在任何联入互联网的计算机上. 这样特别有利于远程联动或触发实时数据获取与处理.

用户界面也作为一个功能模块, 将各分系统状态汇总后以可视化的形式反馈给用户, 同时将用户操作转换为控制指令. 不同于其它功能模块, 用户界面以Web方式向用户提供, 用户通过Web方式获取页面后, 在浏览器中运行其一个实例(Instance), 该实例连接中心服务后实现与控制系统的交互, 实时反馈系统状态. 这种方式使得用户在任何地方只要能够接入互联网就可通过浏览器实现观测操作, 无需任何客户端软件.

和现有观测软件相比, 这种设计使得观测软件可以像传统主控软件那样实现对所有分系统的统一界面集中操作, 同时各个分系统独立实现, 只要保持通信协议与格式一致即可, 不耦合在一个软件内. 通过网络的松耦合, 功能的扩展不影响已有系统, 扩展简便,

支持动态扩展. 同时原生支持远程观测, 采用Web方式可在各种平台下直接进行操作, 不受操作系统限制.

4 后台软件的设计与实现

4.1 中心服务与通信协议

中心服务在整个系统中是核心部分, 虽然它不承担任何控制功能, 但所有通信都通过中心服务进行转发. 在程序实现中采用了观察者模式(Observer pattern)[1]293-304, 其它功能模块和访问用户都作为观察者通过注册方式向中心服务进行消息订阅. 一旦观察者状态发生了变化, 立即通知中心服务, 中心服务根据注册信息向订阅者发送变化的信息, 从而实现了动态的一对多的消息共享, 功能模块和用户可以动态增减.

中心服务和各个功能模块的通信协议我们没有选择常用的Socket协议,而是选用了WebSocket协议[3]. WebSocket协议是HTML5定义的一种新的协议, 实现了浏览器与服务器全双工通信(full-duplex). WebSocket协议和Socket协议非常类似, 具有相近的功能和编程方式. WebSocket最大特点是被现代浏览器支持, 可以直接建立浏览器和服务器之间的链路, 使得基于Web的操作界面可以直接和中心服务通信, 不再需要通过Web服务过渡, 大幅度提高消息传递时效, 克服了长期以来Web系统实时性不足的问题. 同时这种消息传递方式由于跳过了Web服务器, 大大降低了Web服务器负荷.

在通信格式中, 我们选择了JSON (JavaScript Object Notation)格式. 相比较于传统的XML格式, JSON是一种轻量级的数据交换格式, 在网络传输中占用带宽小; JSON采用了完全独立于编程语言的文本格式, 易于阅读和编写, 相比XML格式, 机器解析和生成更为快速和便捷. JSON采用键值对作为数据结构, 具有较好的扩展性, 随着系统运行的变化只需在原消息基础上增加键值对即可实现信息扩充, 而解析方式无需更改. 已有功能模块收到扩充后的消息仍可正常解析获取要求的键值, 采用JSON格式使得消息具有极好的扩展性和向下兼容性.

中心服务采用Node.js平台[4]以Javascript语言实现, Node.js 是一个基于Google Chrome JavaScript引擎的平台, Node.js提供了非阻塞I/O模型, 非常适合分布式的实时应用. 目前近地天体望远镜观测软件中的多数功能模块都采用了Node.js平台在Linux操作系统下实现.

4.2 CCD控制

CCD控制程序是由CCD相机厂商提供的, 同时提供了源代码, 程序采用C++语言开发, 使用了Linux下的Qt框架. 我们对源代码进行了修改, 为其增加了WebSocket协议的双向接口, 使得控制程序可通过WebSocket协议接收中心服务发来的信息, 包括其它分系统的状态和控制指令. 同时CCD状态参数发生变化或者通过软件触发操作时, 我们为相应消息信号增加了关联的槽(SLOT), 将变动参数通过WebSocket发送给中心服务, 实现状态与参数的全网络同步.

由于CCD控制软件能够获取其它分系统的信息, 在CCD控制软件中记录FITS (Flexible Image Transport System)文件同时将这些信息直接记录在头数据单元(HDU)中,

[3]http://www.w3.org/TR/websockets/

[4]http://www.nodejs.org

实现了观测原始图像的实时完整记录.

4.3 伺服控制

伺服系统由于受到伺服通信卡驱动的影响, 无法进行完全的移植. 我们采用适配器模式(Adapter pattern)[1]139-150, 为伺服控制程序重新编写了一个适配程序. 原伺服程序仍然采用标准的串口通信方式, 增加了通过串口反馈伺服和圆顶随动信息以及接收控制指令的功能. 适配程序则和串口服务器通过Socket协议实现通信, 将接收到的伺服信息转化为JSON格式并以WebSocket协议发送给中心服务, 同时将从中心服务接收控制指令转化为串口数据发送给串口服务器. 串口服务器自动完成网络数据到串口数据的双向转发.

4.4 镜筒控制

用于镜筒控制的串口已通过串口服务器接入网络, 根据通信协议编写了镜筒控制程序, 程序直接以Socket协议发送控制指令给串口服务器, 串口服务器将TCP/IP信息转换为串口数据给镜筒控制系统. 同时串口服务器将接收到的镜筒参数通过Socket发送给镜筒控制程序, 镜筒控制程序将其转换为JSON格式, 通过WebSocket发送给中心服务实现共享.

4.5 状态记录

中心服务采用观察者模式后, 整个系统状态在网络上是共享的. 将运行状态记录作为一个单独功能模块, 这个模块并不实现任何控制功能, 模块通过中心服务实时获取系统状态. 根据状态参数的不同性质进行不同方式的记录, 例如对于望远镜指向进行了逐一记录, 而对于CCD温度等进行了统计记录, 只记录一段时间内的平均值.

运行状态的记录从各分系统控制程序脱离出来, 很容易进行统一修改. 目前状态数据均记录在数据库中, 这种方式避免了各个分系统控制程序都要访问数据库, 简化了程序复杂度,提高了控制稳定性. 由于系统是开放式的, 状态量的内容、类型以及数量是不断变化的. 根据这种特点采用了非关系数据库MongoDB作为后台数据库. 状态数据以无模式的文档(Document)形式进行记录.

此外, 由于所有信息都汇总在这个功能模块, 在这个模块中实现了状态的检查. 当分系统出现不匹配状态时, 例如相机开始曝光时伺服还在摆位中, 模块会发出报警, 便于观测员及时处理, 减少观测资源的浪费. 由于以较高频率记录下系统运行状态, 通过数据分析便于发现设备运行中的隐患, 而不必依赖观测图像的异常来发现设备运行中的问题, 例如通过对于伺服数据的分析, 发现了伺服运行中的冗余迭代等.

4.6 观测调度

观测调度是现代望远镜运行中的重要内容, 面对不同的观测任务与需求, 调度策略必须随之改变. 以独立功能模块形式实现调度功能, 这样可以根据不同策略分别编写独立的调度模块, 根据观测需求随时启动相应的调度策略模块, 也就是实现望远镜的运行策略的变换.

目前观测调度模块实现了基础巡天任务的调度, 根据预先输入的观测计划自动发出望远镜摆位指令, 摆到位后发出CCD相机曝光指令, 随着CCD相机曝光完成, 调度模块自动按照计划发送指令将望远镜摆位到下一个观测天区继续观测. 调度模块实现了近地

天体望远镜的全程控观测.

4.7 快门监控

快门监控模块运行于“树莓派”, 程序持续检测高低电平, 侦测到高低转化时记录下对应时刻, 即为快门开和关的时刻. 快门监控模块采用Python语言开发, 运行于“树莓派”专用的Linux系统. Python为树莓派的标准开发语言, 提供了访问GPIO接口的编程接口.

5 用户界面的设计与实现

5.1 布局设计

用户界面以Web页面形式提供给用户, 采用了单页设计, 所有操作和信息获取无需离开页面, 减少了页面切换和刷新, 提高操作效率. 整体页面布局见图4.

页面采用了经典的栅格布局(Grid Layout), 顶栏(Top bar) 采用3栏(column) 模式, 分别为消息、时间和配置, 用于显示系统消息图标、当前时间和系统配置触发按钮. 顶栏下方采用6个仪表盘(Dashboard)来直观显示望远镜各个分系统状态和运行状态. 紧接着是命令区域和短消息区域. 为了保持页面简洁, 所有操作命令隐藏在一个弹出框中, 由消息框左侧按钮触发. 而短消息区域用于协同工作中的即时通讯. 主区域为两栏设计, 左区为CCD图像预览, 右区为观测计划列表. 底栏显示了在线的功能模块和用户情况.

Web界面采用Bootstrap框架[5], 由HTML 5实现. 界面和中心服务的通信以及网页数据的实时更新通过Javascript实现. 界面通过WebSocket和中心服务连接, 该链路的消息传递不再经过Web服务器, 因此当页面载入完成后, 几乎都是由用户计算机资源来完成界面的运行, 相比传统方式可支持更多用户同时访问.

5.2 响应式设计

随着科技发展, 移动设备的屏幕越来越大, 性能越来越强, 使用也越来越广泛. 因此在界面设计中我们做了移动设备兼容处理, 进行了响应式设计, 智能地根据用户设备的屏幕尺寸进行布局调整. 在平板电脑和手机上, 布局会变为图5所示.

布局中保留了顶栏和仪表盘, 仪表盘从单行6分栏转换为2行3分栏, 而CCD图像和观测计划列表已隐藏了, 可通过顶栏右侧的配置区域按钮呼出. 这样通过手机或者平板电脑也可以获得较好的用户访问体验.

除外观外, 操作也为手持设备做了优化, 所有命令均可通过触屏的滑动(Swipe)和点击(Tap)方式操作. 桌面用户除按钮方式外, 也可通过鼠标手势方式模拟触屏操作触发.

5.3 图像显示

现在CCD图像尺寸都非常大, 远远超过屏幕尺寸,观测软件一般只显示缩略图, 便于操作人员及时直观了解设备运行状态. 近地天体望远镜的CCD图像尺寸达到了10560 pixel × 10560 pixel, 通过网络实时显示200 MB图像是不可行的. 考虑到原始图像读出最多为2 × 8通道, 为了避免跨通道的像素合并, 对原图进行20 pixel × 20 pixel的合并, 得到528 pixel × 528 pixel尺寸的缩略图.

[5] http://getbootstrap.com/

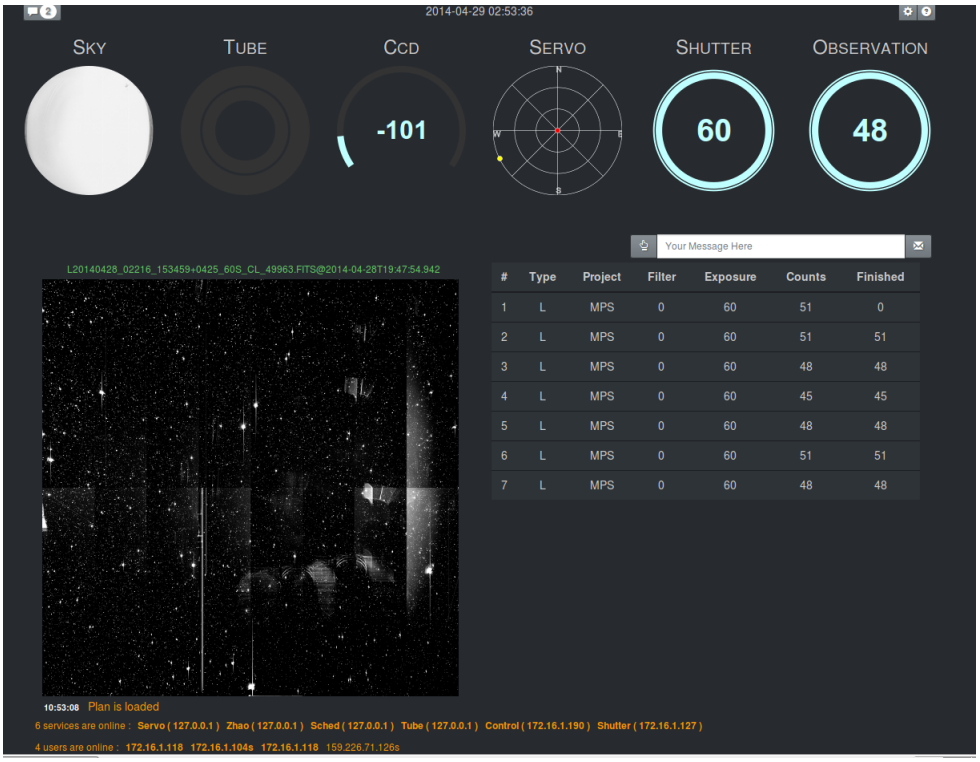

图 4 用户界面

Fig. 4 The user interface

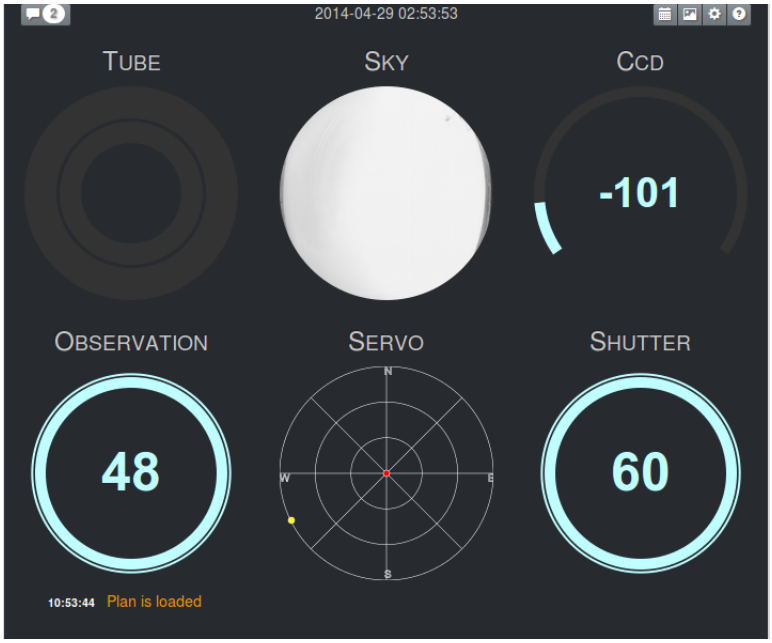

图 5 小屏幕用户界面

Fig. 5 The user interface on small screen

将由哈佛-斯密松天体物理中心开发的JS9[6]模块嵌入到用户界面用于图像显示. JS9是哈佛-斯密松天体物理中心最为著名的天文图像与数据可视化分析工具DS9的Javascript实现版本, 操作方式上和DS9基本一致. JS9能以网页形式直接显示以URL表达的FITS图像. 近地天体望远镜观测软件升级也受到了JS9的启发, 将过去只能在桌面上完成的工作转移到远程来完成.

图像显示流程为: CCD控制程序在生成原始FITS文件的同时生成缩略图, 存入近地天体望远镜图像数据库. CCD控制程序通知中心服务有新图像后, 中心服务会广播给所有在线用户, 用户界面收到通知后直接调用JS9显示URL表达的缩略图. Web服务器会根据URL从数据库中取出图像数据发送给客户端. 由于缩略图尺寸很小而且Web缓存使得不同用户获取相同图像的服务器开销不大, 该缩略图同时也作为近地天体望远镜图像数据库的预览缩略图[2-3].

5.4 即时通讯

在观测过程中, 多人的协同工作是非常常见的, 例如观测申请人与操作者、设备维护人员和现场操作员以及本地和远程的观测者之间, 协同工作中不可缺少的就是即时沟通. 由于我们采用了星形结构的网络消息机制, 很容易提供即时通讯功能, 在用户界面上设置了短消息发送区域, 短消息会以广播的形式发送给所有在线用户. 单页模式使得用户不必离开界面即可完成即时通讯与设备操控, 避免了通过其它第3方软件进行沟通带来的繁琐过程以及界面的来回切换, 大幅度提高沟通和操作效率.

6 总结与展望

近地天体望远镜控制软件系统经过重新设计与实现后, 被称为C3系统(CNEOST Control Center). 它实现了各个分系统在统一界面下的操作, 也实现了分系统之间的协同工作, 在此基础上实现了近地天体望远镜的程控观测和远程观测. 投入使用以来运行稳定, 大大简化了日常巡天观测的操作负荷, 提高了观测运行效率.

系统基于网络消息机制, 具有很好的扩展能力, 随着近地天体望远镜的持续运行以及观测工作的深化, 将会开发更多的功能模块, 丰富观测软件的内涵与外延, 更好地为科学研究服务.

随着观测设备的增加, 这种方式也很容易扩展后服务于多台望远镜的统一操控与协同工作, 并且不受地域限制, 为未来“望远镜云”的组织方式和软件构架提供有益参考.

致谢 感谢中国科学院紫金山天文台盱眙观测站的照日格图、洪仁全、胡龙飞和王旻在系统测试和运行中的帮助.

参 考 文 献

[1] Gamma E, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object-Oriented Software. Boston: Addison-Wesley, 1995

[2] 王歆. 天文学报, 2013, 54: 382

[3] Wang X. ChA&A, 2014, 38: 211

[6] http://js9.si.edu

CNEOST Control Software System

WANG Xin[1,2] ZHAO Hai-bin[1,3] XIA Yan[1,3] LU Hao[1,3] LI Bin[1]

(1 Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210008)
(2 Key Laboratory for Space Object and Debris Observation, Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210008)
(3 Key Laboratory for Planetary Sciences, Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210008)

Abstract In 2013, CNEOST (China Near Earth Object Survey Telescope) adapted its hardware system for the new CCD camera. Based on the new system architecture, the control software is re-designed and implemented. The software system adopts the message passing mechanism via WebSocket protocol, and improves its flexibility, expansibility, and scalability. The user interface with responsive web design realizes the remote operating under both desktop and mobile devices. The stable operating of software system has greatly enhanced the operation efficiency while reducing the complexity, and has also made a successful attempt for the future system design of telescope and telescope cloud.

Key words telescopes, methods: observational