局域网设备控制管理系统

摘　要

以潞安矿务局五阳矿南峰扩区为例，探讨了利用速度法预测矿井新盘（采）区瓦斯涌出量的方法，对制定瓦斯防治方案，进而根治矿井瓦斯具有重要的实际意义。

关键词 局域网；控制；管理

**A Study of Forecasting Gas Emission Rate at New District of**

**Coal Mine with the Initial Velocity Method**

#### Abstract

Based on the measurement data of Nanfeng District of Wuyang Coal Mine, Luan Coal Mining Administration, the gas emission forecast method of the initial velocity with the initial velocity method is introduced, and the application of this method has important practical significance of working out the plan and further prevention and control of mine gas.

**Key words:** initial velocity method; gas emission rate; mine panel(district)

1　绪论

起初做这个系统只是为了解决自己日常生活中的不方便，手机和电脑传文件还需要数据线，觉得太麻烦了。于是自己便随手做了一个手机上的文件服务器APP，发布到应用市场上之后，没想到尽然收到了大量的评论。评论之中，不乏批评我做的不好的地方，但绝大多数的评论都是在给我提供一些建设性的意见，哪些地方做的不好，哪些地方可以这样这样改进，是时受宠若惊。带着不能辜负大众的一片热情的心情，我开始花大量的时间去优化，完善这个系统。

时至今日（2018年４月），此系统在酷安应用市场上已经有５.7万此下载量，6542个关注者，549条评论。同时，此系统的Android版如今已经上架到各大应用商店了：小米，百度，腾讯。其他应用商店下载量虽然不如酷安平台多，但也算是给使用其他应用平台的用户多一种下载渠道吧。虽然这样的一个成绩对于真正的开发者来说并不算多，但是，于我个人而言，是对我自身专业能力的莫大的鼓励。

要知道，在此之前，我尽历了大三下学期的求职阶段。以一个计算机相关专业的本科生的身份来说，大学四年的经历，与全国本科求职者比较，我的简历很普通。难以同其他来自全国各地的本科计算机学生争求同一职位。

个人性格不喜与他人竞争，四年未参加任何全国性的竞赛，也未获得任何奖学金，成绩也不过及格而已，毫无闪光点。要说我大学四年有何收获，我想应该是想明白了一些人或事，找到了自己奋斗的理由，给自己的的人生定义了价值取向。我想我以后不会再为“人一辈子活着有什么意义”之类的问题所困扰了。私以为这才是我在大学四年得到的最大收获。毛主席也说过：看待事物要优先抓住主要矛盾。相比漫无目的的奋斗，先找到正确的方向才是我大学阶段亟待解决的主要矛盾。

当然我在大学四年也不是什么都没干，相比课堂和书本上的学习，我更喜欢自己动手在互联网上学习。从高中开始我就有了这样一个念头，如果没有老师教我了，我该怎么学习？这在高中的同学眼中是一个看起来杞人忧天的想法，现状也证明了我当时的想法的前瞻性。从大一开始逼迫自己去自学，锻炼独立思考问题的能力，独立寻找解决办法的能力。授人以鱼不如授人以渔。我认为一个真正的好老师，应该把学生教到不需要老师的程度才行！

本项目便是在我的自学过程之中诞生的一个点子！待此项目完善至终，也正好到了毕业设计的年份了。选了一个适合自己的题目，便顺势将其作为自己大学本科的毕业设计了。

2 课题背景

3 应用场景

PPT遥控器

平时大家在做PPT演讲的时候，都是用电脑插上投影仪，然后使用鼠标来切换PPT的下一页。这样很麻烦，因为演讲者不能离开电脑，必须在切换下一页的时候，回到电脑旁边，用鼠标点击才能切换。那有了本系统之后，用户就可以实现用手机遥控电脑，只要手机在手里，就可以不需要回到电脑旁边，一直演讲下去，对演讲者的站位会更加自由。

可移动摄像头

因为本系统可以实现局域网内的摄像头直播，所以，只要让手机和电脑通过WiFi连接在同一个局域网下，然后将手机的摄像头画面直播到局域网，电脑上就可以看到摄像头的画面。因为电脑本身携带不方便，很难移动，这样做可以实现用电脑观看隔壁房间的摄像头画面。因为是WiFi连接，所以不需要任何线连接就可以实现手机变成移动摄像头这一梦幻功能。

文件传输和剪切板共享

在我们的日常生活中，手机和电脑传文件一直是非常不方便的。有了我们做的这个系统之后，手机和电脑可以轻松的通过WiFi来传输文件，而且稳定高效，不用担心数据线中途断开等问题。

另外，剪切板在电脑和手机之间的传输也是我们日常生活中的一大难题。如果电脑不安装QQ等软件很难传输，所以我们完全实现了在Web端的剪切板共享，从此，如果你在手机上看到什么网址，文字之类的，在手机上复制了之后，立马可以在电脑上接收到了，十分方便。

4 本文主要研究内容

4 原理

5 设计

5.1 整体架构

整个系统是又两个部分组成：网络核心库、UI层。

为什么要这样设计呢？由于本系统的目标是要做到跨多个平台(Windows,Linux,Android,Mac等)，为了减轻开发和维护的负担，整个系统必须最大化地将通用的部分剥离出来，尽量做到一次编写，到处运行。

因为移动端(Android)和桌面端(Windows,Linux,Mac)的UI差异太大，所以无法实现UI层的共享。但是，网络核心库在各个操作系统中是通用的，可以共享。因为本系统对于UI界面没有太大的要求，所以网络核心库是整个系统编写过程中的重中之重！在完成网络核心库之后，再将网络核心库集成到各大操作系统的UI程序里面，通过一个事件回调接口(后面会在系统初始化里面详细讲解)来实现网络核心库与UI层的通信。这样就实现了系统的跨平台。

其中，网络核心库主要分为两大模块：HTTP服务器模块、UDP守护进程模块

5.2 HTTP服务层

整个系统在启动的时候，第一步就是启动一个HTTP服务器。

这个HTTP服务器承载着文件传输、剪切板复制、摄像头直播和控制指令发送等功能。也就是说，基本上所有的模块都是基于这个HTTP服务器来传输信息和数据的。

HTTP服务器初始化

HTTP服务器启动之后第一件事情就是进行系统初始化操作。系统在初始化的时候，需要传入几个参数：事件回调接口、HTTP服务器所要监听的端口号、临时文件夹的路径还有文件接收路径。如果所传入的参数为空的话，则使用默认的值。

(1) 事件回调接口(EventHandler)

事件回调接口(EventHandler)，是核心网络库向UI层通信的媒介。

为什么要设计这样一个接口？

在GUI编程中，不管是桌面端，还是移动端。整个UI的操作、显示和修改，都是在同一个线程里面的，我们把这个线程叫做UI线程。在UI线程之中，是不允许进行一些耗时的操作的，比如说：Socket监听，网络请求等。否则会出现程序的图形界面卡死的情况。

所以，在本系统中，因为需要启动一个HTTP服务器，这个HTTP服务器是绝对不能在UI线程中启动的，必须要另起一个新的线程。那么问题就来了，如果这个HTTP服务器有什么状态变化的情况（比如：HTTP服务器收到一个上传过来的文件，收到控制指令等），如何去通知UI线程，并在图形界面之中显示出来呢？

我们只能设计这样一个接口，在UI层实现接口中定义好的方法，然后当HTTP服务器出现状态变化的时候，只需要调用该接口对应的方法即可。

事件回调接口是一个接口(Interface)，定义在网络核心库之中。里面定义了一些事件的方法，但是没有实现他们。这些方法是需要由UI层来实现的。因为UI层各有不同，比如收到的消息如何显示？在Android上是显示一个AlertDialog，在Windows上则是弹出一个AlertWindow。所以这些事件回调的方法，必须是由不同的UI层自己去实现，无法共享。

事件回调接口中包含一下事件方法：

- 当收到剪切板的时候

- 当收到Web上传的文件的时候

- 当收到其他客户端上传来的文件夹的时候

- 当新的设备上线或者下线的时候

- 当收到远程控制指令的时候

- 存储键值对

- 获取键值对

为什么事件回调接口中还需要包含存储键值对、获取键值对两个方法？

系统在初始化的时候需要读取配置文件，这些配置文件是以键值对的形式存储在设备之中的。然而，在不同的操作系统之中，存储配置信息的方式各有不同。比如，Windows是存储在AppData目录下的，Linux是存储在用户目录下的隐藏文件夹里面的，而在Android系统中，配置信息是存储在SharedPreferences之中的。所以，我们需要抽象出一个存储键值对方法，和一个获取键值对的方法，来让各大平台各自实现自己的存储过程即可。

(2) HTTP服务器监听的端口号

HTTP服务器要监听的端口号(Port)，是在启动HTTP服务器时的必要参数。他有自己的默认值，用户可以在设置界面里面更改端口号。但是，因为服务器一旦启动了，端口号就不能修改了，所以当用户在设置界面里面修改了端口号之后，必须重启整个系统才能使修改生效。

(3) 临时文件夹路径

临时文件夹路径(Temp Path)，该参数是为了防止Temp目录空间不足的情况出现(Linux下默认Temp目录存储空间大小为4G)，于是让用户可以自定义Temp目录位置。

(4) 文件接收路径

文件接收路径，是该参数是在启动HTTP服务器时的必要参数，他有自己的默认值，用户也可以对其进行修改。在整个系统之中，该参数是以一个共有变量的形式存储在内存之中的。也就是说，如果用户在设置界面里面修改了该参数，则不需要重启系统，即可使之生效。

5.3 UDP发现层

守护进程(Deamon)，是整个网络核心库的两大主要模块之一。主要的作用是进行同一局域网下的在线设备相互发现(后面会在“设备发现模型”部分详细讲解)。

5.3.1 守护进程初始化

守护进程初始化，是在HTTP服务器初始化之后进行的。初始化的时候，只需要一个参数即可：操作系统的标识。

操作系统的标识，主要作用不过是让用户能够快速识别其他在线设备。

5.3.2 设备发现模型

为什么需要“在线设备互相发现”这一功能？

在同一局域网之中，当有多台设备同时开启了本系统时。用户可以进行一些更加高级的操作，比如：远程控制，向指定设备发送文件夹等。如果没有在线设备相互发现这一功能的话，用户需要自行查询每台机器的局域网IP地址，同时还要自己一个字一个字填写IP地址和端口号，况且IP地址和端口号的输入过程十分麻烦且耗时。

为了解决这一问题，为了让用户能够无需输入在线设备的IP地址，用户唯一需要做的事情就是在所有在线设备列表里面选择自己想要的设备即可。所以我们增加了UDP守护进程，主要用于同一局域网下的在线设备互相发现。

在线设备发现模型的要求

- 当设备上线的时候，要广播通知所有局域网下的已有设备

- 当设备下线的时候，所有局域网下的其他设备也要实时更新在线设备列表

- 设备需要让所以同一局域网下的其他设备知道自己的状态(如：是否开启了接收控制指令的选项)

- 当设备的状态改变时，要实时地在其他设备的在线设备列表里面更新

设备发现模型

【图】

如图，本系统的设备发现模型包含三个部分：广播发现、心跳连接和一个状态获取HTTP接口。

(1).广播发现

广播发现部分，是整个设备发现模型的基础，也是实现整个模型的过程的第一步。当设备启动本系统的时候，UDP守护进程初始化，同时会向所在的局域网广播一条上线提醒消息。但是，消息之中并不会包含”上线“等复杂表示，该消息仅仅只包含一个信息：那就是本机的HTTP服务器端口号。也就是说，所谓的广播发现，不过是一个向局域网下的全部设备广播一下自己的HTTP服务器端口号的简单操作罢了。

为什么只包含一个HTTP服务器端口号？

广播发现这一行为是整个设备模型的基础，但也仅仅只是基础而已，他不会包含太多的功能或者作用。在系统启动的时候，向局域网的所有设备广播自己的HTTP服务器端口号，当其他在线设备收到这一个UDP广播消息的时候，仅仅意味着：”当前局域网下的在线设备列表产生变化了，需要更新一下在线设备列表了！“。

虽然仅仅只包含一个简单的HTTP服务器端口号，但它却像一个跳板，告诉其他设备：我的状态改变了(这里的状态不仅仅包括上线、下线，还有在用户开启或关闭了”接收控制指令“选项的时候，也属于状态变化的范畴，这个时候也会向局域网下的所有设备广播自己的HTTP服务器端口号。)，快利用你们收到的HTTP服务器端口号，在我的HTTP服务器上获取我的具体状态信息。

(2).心跳连接

在设备发现模型之中，为了满足“当设备下线的时候，所有局域网下的其他设备要实时更新自己的在线设备列表”这一要求，我们必须要让每一台设备直接都建立一个心跳连接，这个心跳连接不会进行任何数据的传输，他的唯一作用就是让其他的设备知道自己是在线的。如果这个心跳连接断开的话，就说明此设备已经离线了。通过这样一种方式，就实现了“当设备下线的时候，所有局域网下的其他设备要实时更新自己的在线设备列表”。

(3).状态获取HTTP接口

每一台设备都需要设置这样的一个状态获取HTTP接口。因为HTTP协议是无状态的，这个接口的作用很简单，不管是谁来访问这个接口，都会向访问者返回本设备的实时状态信息，返回之后则立即断开连接。

每当收到在线设备列表变更的消息的时候，每台设备都会访问所有设备的状态获取HTTP接口。如果无法访问，则说明该设备已经离线，如果能访问，则将返回来的设备实时状态信息覆盖原有的设备列表中的信息。

5.3.3 总结

在本系统中设计的这个设备发现模型，在设计模式上，完美的实现了分而治之的思想。将设备实时状态更新这一功能按照类型分给了三个部分：广播发现，心跳连接和状态获取HTTP接口。他们分别代表了三个角色：通知者，监视者和展示者。每一个角色自身的功能极其简单，但是三个角色合作起来，却又能正好完成一项复杂的工作。这也是我个人在Go语言的学习过程之中领悟到的：解耦。这样一来，就不会像其他的设备发现模型那样，出现无限的广播反弹的现象，而且又简单。正所谓：把一件简单的事情做复杂很容易，但是能把一件复杂的事情简化的人才是大师。

5.4 文件传输模块

整个文件传输模块由两部分组成：文件部分和剪切板部分

5.4.1 文件部分

在文件部分的数据结构设计上，每一个文件个体包含三个信息：文件名，文件的绝对路径和文件的ID。

为什么要设计文件ID这一信息？

由于文件传输模块是建立在HTTP服务器之上的，所以，为了防止不同的文件在的URL下载地址出现冲突，URL不能以文件名作为标识，又因为文件路径包含的“/”符号与URL的分隔符冲突，所以也不能以文件路径作为文件URL的唯一表示。只能在每一个文件添加了之后，用一个随机生成的数字作为每一个分享出去的文件的唯一标识，这就是文件ID的由来。

文件部分包含四个HTTP页面或接口，分别是：

- 文件列表页面

- 文件下载接口

- 文件预览接口

- 文件上传接口

文件部分是文件传输模块的重要组成部分，其未来的发展方向还有很多，改进的方法也有很多，比如使用并行下载来提升文件传输的速度等。

5.4.2 剪切板部分

剪切板部分比较简单，同样是给予HTTP服务器的。主要包含两个共有变量，一个用于存储剪切板的内容，另一个用于存储剪切板开启的状态。

为什么需要剪切板部分？

剪切板是人们日常使用手机的过程中最常用到的功能之一。比如平时大家在手机上复制一段文字，复制一个网址，想要传给电脑怎么办呢？以前的办法是通过QQ或者微信发送给电脑端，然后电脑端再复制一下。这样有两个缺点：一个是电脑必须联网，另一个是电脑还必须安装QQ或者微信的客户端，如果说还有什么缺点的话，我认为应该是二次操作。所谓的二次操作是指手机复制了一次剪切板，发送到电脑上之后，电脑还得再复制一次。很不方便。所以我决定在本系统之中加入剪切板功能，就是为了方便大家在家具智能设备之间共享剪切板。

5.5 远程控制模块

远程控制模块包括控制者和被控制者。目前的版本，控制者只能是手机，被控制者只能是电脑。

5.5.1 控制者

控制者通过发送将控制指令以HTTP请求的方式发送给被控制者

5.5.2 被控制者

被控制者通过在HTTP服务器上新增一个接口，专门用于接收HTTP控制指令。当收到HTTP控制指令的时候，再又UI层去解析并执行控制指令

5.5.3 权限问题

被控制者默认是不接受任何控制指令的，这个时候如果有控制者想发送控制指令过来，会返回无效信息。只有当被控制者开启受控端选项的时候，才会接收控制指令。在“受控端”开启的时候，有两种选项：第一种是接收任何人的控制指令，第二种是接收指定设备的控制指令，第三种是接收除了指定设备以外的所有控制指令。

5.5.4 申请控制权限的过程

如果控制者是在受控端的可控设备列表之外的话，需要先向受控端申请控制权限，受控端同意了之后，才能进行远程控制。

5.6 摄像头直播模块

摄像头直播模块是基于HTTP服务器的，主要由广播，WebSocket长连接，直播者页面和观看者页面组成。

5.6.1 .WebSocket长连接

WebSocket长连接是将所有观看者和直播者通过HTTP服务器连接起来的媒介。之所以选择WebSocket是因为这是目前最新的Web技术————HTML5推荐的保持长连接的方法。摄像头直播时的广播就是依靠这一长连接来实现的。

5.6.2 广播

广播是在HTTP服务器收到直播者在Web浏览器上发过来的摄像头数据的时候，将这一数据由服务器向所有在线的观看者发送过去的行为。

没当有新的用户连接到摄像头直播页面，就会与服务器建立一个WebSocket连接，同时，将该连接加入所有已连接的数组里面。这样一来，当直播者需要广播数据的时候，只需要遍历一下这个数组，然后逐一发送即可。

5.6.3 直播者页面和观看者页面

直播者页面和观看者页面是由HTML5技术实现的，通过调用HTML5浏览器接口，获取摄像头数据，然后通过WebSocket发送给服务器，在观看者收到摄像头数据之后，利用Canvas画布显示出来即可。

6 实现

6.1 本地服务器客户端开发现状

6.2 关于跨平台网络库

6.3 为什么选择Go语言

Go语言介绍：

Go语言是谷歌开发的一款全新的编程语言，可以在不损失应用程序性能的同事，大大降低代码的复杂度。

他的宗旨是让开发者更加容易地开发出简单，稳定，高效的软件！

为什么选择Go语言？

1.性能(并发)

2.跨平台

3.网络库丰富

4.语法简单

6.3.1 性能(并发)

Go语言针对多处理器系统应用程序专门进行了优化，在使用了Go编译之后，程序可以媲美C或C++代码的速度，而且更加安全，而且还支持并行进程。

编译之后的Go语言代码的运行速度和C语言非常接近，而且编译速度非常快，就好像在使用一个动态的交互式编程语言一样。

目前，现有的各大流行的编程语言都没有对多核处理器进行专门的优化。而Go语言就是为了解决这一问题而诞生的。

6.3.2 并发编程

时至今日，并发编程已经成为了程序员的基本技能了，在各大技术社区都可以看到诸多与之相关的讨论主题。到底那种方式是最佳的并发编程体验？或许会一直争论下去。但是Go语言却一反常态，从底层就将一切都并发化了！运行时使用了Goroutine运行所有的一切，当然也包括main.main入口函数。

可以说，Goroutine已经成为了Go语言的标志性特征了。他使用类协程的方式来灵活地处理并发单元，同时却又在运行时层面做了更加深度的优化处理。这样，使得Go语言在语法上的并发编程变得极为简单！无需处理回调，无需关注执行时的切换，一切仅仅需要一个go关键字，简单而又自然！

搭配Go语言的channel，可以让新手也能轻松实现CSP并发模型。将并发单元间的数据耦合拆解开来，各司其职。这对所有纠结于内存共享、锁粒度的开发人员来说都是一个可以期盼的大解脱！如果真的要说有什么不足，那就应该是要有一个更大的计划，将通信从进程内拓展到进程之外，实现真正意义上的分布式！

Goroutine和Channel使得编写高并发的服务器软件变得相当容易，很多情况下完全不需要考虑锁的机制和由此带来的一切问题。比如，单个的Go应用也能有效的利用多个CPU核心，并且执行的性能特别好。这个和Python相比有天壤之别！多线程的Python程序其实并不能有效地利用多核的优势，只能用多进程的方式来部署。如果使用标准库里面的multiprocessing包又会对监控和管理造成很多不必要的挑战。所以，部署一个python语言的时候，通常是每个CPU核心来部署一个应用，这样就会造成很多不必要的资源浪费。比如说假设某个Python应用启动之后需要占用100MB内存，而服务器有32个CPU核心，那么留下一个核心给系统，而运行31个应用副本需要浪费3GB的内存资源，太糟糕了！

6.3.3 良好的语言设计

这也是我个人非常喜欢Go的很大的一个原因。虽然，从学术的角度来讲，Go语言是十分平庸的，不支持许多高级语言的特性。但是，从工程学的角度来讲，Go语言的设计是十分优秀的！规范足够灵活，有其他语言基础的程序员都可以通过快速学习来上手。更加重要的是Go语言自带的完善的工具链，大大提高了团队协作的一致性。比如gofmt自动对Go语言的代码排版，很大程度上杜绝了不同的人写的代码排版风格不一样的问题。吧编辑器配置成在编辑存档的时候自动运行gofmt，这样在编写代码的时候可以随意摆放位置，当你保存的时候自动变成正确的排版代码。是不是很方便！此外还有gofix,govet等非常有用的工具。

6.4 Go语言的现状与前景

其实我在做出这个选择之前已经花了大量时间做过详尽调研。 国外如Google、AWS、Cloudflare、CoreOS等，国内如七牛、阿里等都已经开始大规模使用Golang开发其云计算相关产品。 跟着世界级巨人的脚步应该不至于走错方向，而且在学习Golang的过程中，我也渐渐被其背后的设计哲学所折服。

另外，云风博客中曾说过这样一句话：

我发现我花了四年时间锤炼自己用C 语言构建系统的能力，试图找到一个规范，可以更好的编写软件。结果发现只是对 Go 的模仿。缺乏语言层面的支持，只能是一个拙劣的模仿。

6.5 桌面GUI框架

因为本系统需要做到跨平台，所以在系统实现的时候需要寻找合适的桌面GUI库。

桌面GUI开发有很多选择，主要分为跨平台的GUI框架和非跨平台的GUI框架。其中我们重点关注跨平台的GUI框架。

- Qt。比较知名的跨平台框架，但是因为导出的安装包体积太大，所以放弃。

- Sciter。非常轻量级的GUI框架，使用HTML来写界面，导出的安装包体积也小。

- UI。来自github上的一个第三方GUI框架，跨平台，原生支持，Go语言。唯一的缺点是GUI组件太少。

综合考虑之后，最终选择了Sciter框架。

6.6 Android开发现状

一开始在实现的时候就是在Android平台实现的，等到用户量多了起来的时候，才开始扩展到其他平台。

目前Android应用程序开发的现状就是本地化要求比较高的App就使用Native开发，也就是用Java来开发；那些以内容为主的，对本地功能要求不高的，完全可以用网页替代的App（如：百度贴吧，微博等），就是用一些Web技术来开发，也就是用HTML,CSS和JavaScript来开发。对于本系统而言，JavaScript的功能自然是不足以满足我们的要求的啦，所以肯定是Native方式。但是依然存在一个问题。

因为我们是需要在本地建立一个HTTP服务器的，由于HTTP服务器本身结构复杂，Java标准库里面又没有已经实现好的HTTP服务器，所以，HTTP服务器底层实现部分我选择了使用Go语言来编写。

Go语言官方团队构建了一个GoMobile库，这个库是可以帮助你将Go语言的代码运行在Android上面的。虽然不能用Go语言来写Android的UI，但是可以将Go语言代码以第三方库的形式嵌入到你的Android项目里面，然后使用Java来调用。有了这个库，我们就可以把让Go语言来写我们的HTTP服务器变成现实啦！因为Go语言本身就是为了写服务器而生的。所以使用Go语言写起来会非常顺手。

6.7 摄像头直播功能实现

摄像头直播功能主要是参考了网上的一个第三方摄像头直播示例代码，然后有了灵感。

为什么在Web端实现？

整个摄像头直播功能主要是在Web端实现的，也就是说需要借助浏览器来调用系统的摄像头接口。这和大家以往想象的摄像头直播软件不同。之所以这么实现，一个是因为Go语言本身就有很多这种类型的例子；其二是因为在Web端实现的话，另一台设备不需要安装客户端了，只要有一个浏览器即可实现直播者或者观看者的角色。我觉得这样会方便很多。

首先直播者在Web端点击一个按钮，JavaScript便通过WebSocket连接到HTTP服务器，然后用JavaScript通过HTML5的摄像头API————navigator.getUserMedia()函数，来申请摄像头画面数据。拿到摄像头画面数据之后呢，把这些数据通过刚才连接上的WebSocket发送给服务器，服务器收到数据之后，则对所有的WebSocket连接进行广播。至于观看者这边，同样，也是用WebSocket与服务器进行连接，然后监听数据。如果服务器发送数据过来了，观看者这边再用JavaScript把画面显示在HTML中的Img标签里面即可。

6.8 Robotgo自动化库

为了实现键盘和鼠标的控制，我们找到了一个第三方库：https://github.com/go-vgo/robotgo

这个第三方库实现了跨平台的键盘鼠标输入控制，也就是说，可以在Go语言代码里面控制键盘和鼠标的输入。比如执行指令"enter"，就相当于按了键盘上的回车按键。另外一个，关于组合按键，他是这样解决的。例如：输入"control alt t"这样的命令，以空格来隔开，就可以实现组合按键的功能。

本系统在实现的时候也是原封不动的使用了该第三方库的指令系统，在控制端输入指令之后，通过HTTP协议发送给受控端，然后受控端执行指令即可。

7 外文翻译

8 结论

9 致谢

10 参考文献