**分类号：TP311 U D C：D10621-408-(2022)1083-0**

**密 级：公 开 编 号：2018022196**

**成都信息工程大学**

**学位论文**

**一种基于物联网的疫情播报系统**

|  |  |
| --- | --- |
| **论文作者姓名：** | **陈宇航** |
| **申请学位专业：** | **电子信息工程** |
| **申请学位类别：** | **工学学士** |
| **指导教师姓名（职称）：** | **朱静（讲师）** |
| **论文提交日期：** | **2021年05月18日** |

**一种基于物联网的疫情播报系统**

**摘要：**当前全球新冠疫情形势严峻，我国面临着较大的疫情压力。境外回国的人员变多，境内的人员流动增多，疫情传播风险变大，所以实时了解疫情情况是非常有必要的。基于物联网技术，实现对疫情数据的收集和处理，并且选择合适的嵌入式平台完成疫情的数据展示和播报，实现对当前疫情的实时了解掌控。

系统分为硬件和软件部分，硬件主要是由串口屏，eps32芯片，SYN6288语音模块，软件部分为服务器，网页和arduinoIDE上面编写esp32的代码。利用esp32的WiFi模块，向node服务器发送请求，然后node服务器向外部接口发送请求，拿到数据之后在node服务器处理数据，因为数据量太大，node服务器再返给esp32处理之后的数据，并且将这个数据利用串口通信发送到串口屏和语言模块，因为数据太多，所以搭建了一个网页来展示更多的数据。搭建的网页主要采用的技术为vue3+typescript+element-plus+echarts。在这个网页里主要的内容为五个部分，第一个部分为疫情日历，页面展示一个日历，点击日历的某一天会展示出当天的疫情情况。第二个部分为接种情况，主要展示的是国内外疫苗接种数据。第三个部分为数据可视化，将中国地图每个省划分为一个区域，每个区域根据当天的新增感染数来分级，感染数目越多颜色越深。第四个部分为国外疫情情况展示，主要是展示国外的国家疫情情况。第五个部分为国内三十四个省份的疫情情况总览和对应的每个省份的市县区感染情况如何。

**关键词：**疫情数据；esp32；node服务器；SYN6288语音模块；网页前端

**An epidemic reporting system based on Internet of Things**

**Abstract:**The current global COVID-19 situation is grim, and China is facing great pressure from the epidemic.  As the number of people returning to China increases, the risk of the spread of the epidemic increases, so it is necessary to know the situation in real time.  Based on the Internet of Things technology, the epidemic data can be collected and processed, and the appropriate embedded platform can be selected to display and broadcast the epidemic data, so as to realize the real-time understanding and control of the current epidemic.

The system is divided into hardware and software parts. The hardware is mainly composed of serial screen, EPS32 chip, SYN6288 voice module, and the software part is composed of esp32 code for server, web page and arduinoIDE.  The WiFi module of ESP32 is used to send a request to the Node server, and then the Node server sends a request to the external interface. After receiving the data, the node server processes the data. Because of the large amount of data, the Node server returns the data processed by ESP32.  And the data is sent to the serial port screen and language module by using serial port communication. Because there is too much data, a web page is built to show more data.  Vue3 +typescript+ Element-Plus + Echarts are the main technologies used to build web pages.  Some data visualization has been done in this page.

**Key words：**Epidemic data; Esp32; Node server; SYN6288 voice module; Web front end

目 录

论文总页数：43页

[1 引 言 1](#_Toc15460)

[1.1 研究背景及意义 1](#_Toc19868)

[1.2 国内外研究现状 1](#_Toc5142)

[1.3 主要研究内容 1](#_Toc9073)

[2 系统总体设计 3](#_Toc6521)

[3 硬件选择 4](#_Toc25983)

[3.1 网络连接 4](#_Toc10380)

[3.2 数据展示 4](#_Toc2150)

[3.3 语音播报 4](#_Toc8071)

[4 软件设计 5](#_Toc24841)

[4.1 esp32主程序 5](#_Toc18178)

[4.1.1 WiFi模块 5](#_Toc6156)

[4.1.2 Json解析数据以及HttpClient网络请求 6](#_Toc21692)

[4.1.3 软串口 7](#_Toc9526)

[4.1.4 串口屏 7](#_Toc7768)

[4.1.5 SYN6288语音模块 7](#_Toc14507)

[4.2 node服务器 8](#_Toc18491)

[4.2.1 node简单介绍 8](#_Toc28389)

[4.2.2 原生node和框架比较 8](#_Toc26632)

[4.2.3 postman 10](#_Toc11686)

[4.2.4 服务器搭建 10](#_Toc8299)

[4.3 前端网页 12](#_Toc14639)

[4.3.1 Vue框架 12](#_Toc10196)

[4.3.2 ElementUI 12](#_Toc19590)

[4.3.3 Echarts 12](#_Toc31296)

[4.3.4 网页搭建 12](#_Toc5264)

[4.4 串口屏 18](#_Toc13913)

[4.5 SYN6288语音模块 19](#_Toc11586)

[5 测试部分 21](#_Toc26728)

[5.1 软件测试 21](#_Toc31047)

[5.2 硬件连接 22](#_Toc23015)

[5.3 遇到问题 23](#_Toc12222)

[5.3.1 Wifi连接不成功 23](#_Toc29381)

[5.3.2 Json数据展示不完全 23](#_Toc6399)

[5.3.3 语音模块 23](#_Toc24976)

[5.3.4 请求失败 23](#_Toc31377)

[5.3.5 跨域问题 23](#_Toc10784)

[5.4 联调 23](#_Toc29614)

[结 论 30](#_Toc30335)

[参考文献 31](#_Toc22283)

[致 谢 32](#_Toc9244)

[声 明 33](#_Toc6028)

[作者签名： 33](#_Toc3703)

[附录一 服务器简单构建 34](#_Toc29544)

[附录二 express框架 35](#_Toc5483)

[附录三 文件处理 36](#_Toc29453)

[附录四 获取接种数据 37](#_Toc11421)

[附录五 路由相关 38](#_Toc28729)

# 引 言

## 研究背景及意义

在当下，新冠疫情对我国甚至是全球人民的生活都产生了巨大的影响，并且在可以预见未来的短期内是难以结束的，新冠疫情深刻地说明，全世界人类是休戚与共命运共同体。疫情带来的影响远远超过了疫情本身，对此曾经有人说过，新冠疫情是自第二次世界大战来世界面临的最大挑战。所以，作为这个休戚与共的命运共同体的一员，应当每时每刻关注疫情的发展情况如何，并且根据最新的疫情数据来调整自己的日程安排，疫情物联网播报系统旨在帮助人们实时地获取最新的疫情数据，将疫情的最新数据用一块屏幕展示出来，这个系统可以作为一个桌面的装饰物，一眼就可以看到当前疫情的最新情况，并且这个系统还搭载了语音模块，可以用将疫情情况用语音的形式播报出来。最后还搭建了一个网页来查看更多的数据，比如国外的疫情情况如何，全球疫苗的接种情况如何，今日全国各地新增确诊多少个，或者是国内各省甚至各市的疫情情况如何，都可以在这个网页查询到一目了然。

## 国内外研究现状

一场很突然的疫情防控战役在全球打响，值得庆贺的是，现在的时代已经和2003年完全不同，科技的力量正在赋予这次疫情防控战更多的可能性。在疫情防控中。新一代信息技术在这方面起到了至关重要的作用，例如大数据和物联网。

自新冠以来，物联网大数据在帮助疫情监测等，做出了极大的贡献，例如一些实时更新疫情数据的小程序，不但能够让全国人民实时了解疫情数据的变化，还能为治愈率和发病率等等提供数据方面的支持。

物联网还能用在医疗领域，物联网医疗设备能通过许多传感器持续地收集数据，例如人的体温，血压，还有其他医疗参数，并且能够不间断跟踪数据，对检测疾病发作的迹象发挥了很大的作用。不仅如此，医护人员还可以根据实时采集回来的数据进行远程医疗指导。

疫情下面，前线医生的人数不足或者是医生的水平差异，这时候远程医疗就成为了解决问题的手段。既能快速解决问题，也避免了线下接触，减少了医疗人员的感染风险。

针对某些重点疫区相关接触史人员的居家管理，也是防疫中一个重要的环节，有某些比较粗暴的做法是直接使用物理手段将门进行封锁，但是这种做法引起了很多被封锁烦人的不满，再者甚至引起了接触式的冲突。所以物联网也能方便安全的做到居家的监控。在被居家隔离的入户门安装感应门磁，当门磁感应到有移动的时候，向监管人员发送警告，监管人员采取行动到现场检查确认。NB-IOT智能门磁系统以NB-IOT物联网网络作为基础，通过安装NB-IOT智能门磁硬件，实现对门的开关状态监控，当有人开门触发门磁发生警报，这个警报会实时地发送给防疫隔离中心。

## 主要研究内容

本文用esp32的WiFi模块实现网络连接，node搭建服务器实现接受esp32发送的请求，并且在收到esp32发送的请求之后，向外部的接口发送请求获取未经处理的大量数据，收到数据之后在node服务器进行数据的处理，过滤掉不需要的数据，并且做一些数据的校准。将最终的数据返回给esp32，最后在esp32利用串口通信将这个数据发送给串口屏实现数据的展示，并且也利用串口通信发送给语音模块，实现疫情数据的播报功能，由于esp32处理不了太多数据，更多的数据搭建一个网页作为更多数据的展示，并且在这个网页做一些数据可视化。最终实现了疫情概况在串口屏展示和疫情情况的播报以及在网页上展示疫情数据的功能。

# 硬件设计与实现

整个系统需要用到的功能有网络部分，数据的展示部分以及实现语音播报功能的部分。以下是各部分功能硬件的选择。

## 系统总体设计

物联网疫情播报系统硬件不分采用ESP32作为主控芯片，使用USART串口屏，SYN6288语音模块等外设，软件部分使用node作为后端服务器支持，vue作为前端网页支持。下图2.1为整个系统的流程图。其中esp32利用wifi模块发起网路请求，获取数据并且将数据通过串口通信发送到串口屏和SYN6288语音模块。串口屏拿到esp32发送过来的数据，予以展示。SYN6288模块拿到数据之后进行语音播报疫情数据。esp32的代码主要是用arduino框架开发，串口屏主要是用上位机软件开发，SYN6288主要是利用了厂家提供的在arduino开发下的函数生成器。前端部分主要是向服务器发送请求，拿到数据之后进行数据的展示，分为几个部分，疫情日历，疫苗接种，每日新增病例数据可视化，国外疫情，国内三十四省疫情的展示。主要是用vscode编辑器开发，用到的技术有axios一个网络请求的第三方库，elementui一个UI界面的框架，echarts一个数据可视化第三方库。之后是后端node服务器的开发，软件也是用vscode编辑器开发，用到的技术有node.js和基于node.js的框架express.js。

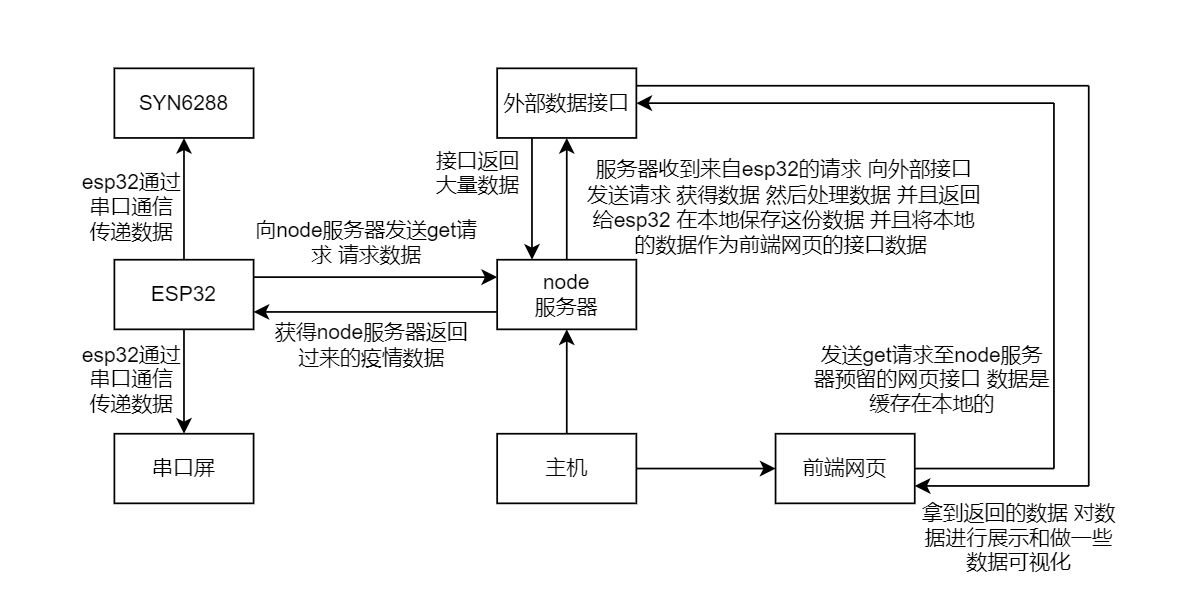


图2.1 系统流程框图

## 网络连接

这个系统中一个很重要的部分就是网络连接部分，整个系统需要连接网络并且发送网络请求，然后拿到数据，将数据发送到需要的地方。

### 芯片选型

这个功能部分选择了esp32。虽然stm32也能实现相同的功能，但是stm32在价格上就比esp32贵了不少，并且esp32体积小巧，速度超快。ESP32是一款通用型WiFi-BT-BLE MCU模组，集成了传统蓝牙、低功耗蓝牙和Wi-Fi，具有广泛的用途[1]。可以说是专为物联网而生，它的应用领域非常广，像智能家居和可穿戴电子设备。并且功耗很低，价格便宜，esp32的性能十分稳定，工作温度的范围能够适应很多环境，低到零下40摄氏度高到125摄氏度 。

### 管脚图

下图2.2是这个模组的管脚图，在这个模组中VIN为串口屏和语音模块提供+5v，IO16和IO17用作和串口屏进行串口通信，IO22和IO23作为软串口和语音模块进行串口通信。

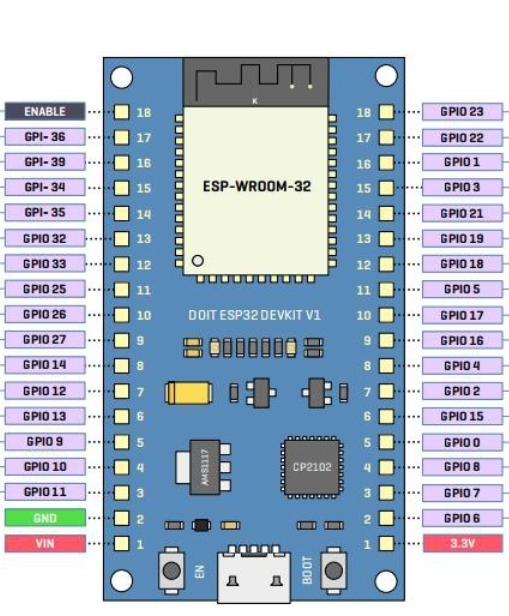


图2.2 esp32模组管脚图

## 数据展示

整个系统还需要有个展示数据的部分，主要是展示一些疫情数据，所以需要用到屏幕来展示数据。

### 屏幕选型

数据展示部分选择了串口屏，因为串口屏操作简单，串口屏厂家为用户封装了所有显示的底层功能，抽象出一系列的通用UI空间和对应的事件逻辑，大大缩短开发周期。简单的4根杜邦线就可以控制串口屏，只需要用对应的上位机软件对样式进行配置。通过图形化的方式就可以对人机界面的布局和大多数的逻辑（比如界面背景，按钮效果，文本显示等）进行设置，十分便捷和高效[2]。编译无误后将工程下载到串口屏内部存储器中。

### 串口屏实物展示

如下图2.3，在使用这个串口屏的时候，首先需要将用上位机写好的工程下载到串口屏。这里下载工程到串口屏的时候需要用到usb转ttl模块，如图2.4。



图2.3 串口屏实物



图2.4 usb转ttl

## 语音播报

syn6288是一个用于播报中文语音的模块，文本识别更智能，语音合成更自然，硬件接口简单，极高的性价比，通过串口通信方式，接受待合成的文本数据实现文本刀语音的转换。SYN6288支持GB2312、GBK等内码格式的文本, 支持16级音量调整、6级词语语速调整等多种功能[3]。

# 软件设计

软件设计部分主要有esp32的主程序，node服务器，前端网页搭建，串口屏，SYN6288语音模块。

## esp32主程序

用arduino框架开发了esp32主程序的代码，在这个主程序中主要用了WiFi，HttpClient，ArduinoJson，SoftwareSerial等库，利用esp32自带的WiFi模块，连接上wifi，发送网络请求然后拿到返回的数据，用json格式化处理返回的数据。JSON是一种独立于编程语言的通用数据交换格式，通用性强，可读性好，可满足嵌入式设备数据量少的通信需求[4]。下图3.1为esp32主程序的流程图。

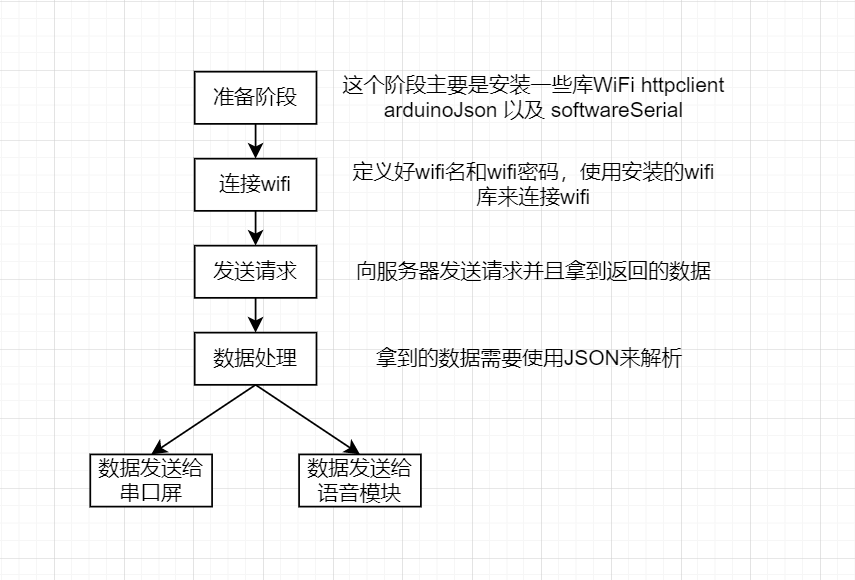


图3.1 esp32主程序的流程图

### WiFi模块

需要用到WiFi的库，然后定义wifi名和wifi密码，串口初始化设置波特率为115200，方便打印相关信息，方便看wifi的连接状态如何，然后调用wifi.begin方法，参数为账号和密码开始连接wifi，wifi.status的返回值有几种，比如WL\_CONNECT\_FAUKED表示密码不正确，当这个值为WL\_CONNECTED时候就表示wifi连接成功了。当wifi还未连接成功的时候就一直在这个循环里面打印.....来表示wifi正在连接中。当wifi连接成功之后就会跳出这个循环进入下一条语句打印WiFi connect success！这里就表示wifi连接成功了，并且这里用串口监视器输出了一下ip地址，因为后面要发送网络请求的地方也会用到这个ip地址。获取这个ip地址也可以直接在电脑命令行ipconfig查看。

### Json解析数据以及HttpClient网络请求

这一步主要做的就是发起网络请求，然后拿到服务器那边返回过来的已经JSON序列化之后的字符串，然后再JSON反序列化拿到数据。首先需要调用两个库，分别是ArduinoJson和HTTPClient，声明HTTPClient对象，将接口地址填写入http.begin方法，此处的接口地址就是上面wifi中的ip地址加上开启在本地的node服务器的端口号，准备启用连接，然后发送get请求，如果请求过程无异常，状态码应该是200，这里判断一下返回的状态码如果等于200或者介于200到300之间，请求无异常进行下一步。这里用一个payload变量来存储拿到的数据，这里的数据应该是一个json序列化之后的字符串，创建一个对象doc，用于存放被解析之后的JSON信息，在创建这个对象的时候还需要传入一个参数，这个参数的作用是告诉esp32所创建的这个对象应该占用多大内存空间。然后进行JSON文件解析，解析完之后就可以根据对应的字段来获取对应的数据了，对应的字段图在下方图3.2，并且将需要用到的数据都取出来，待用。

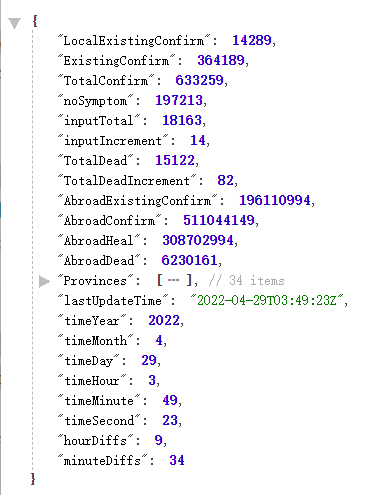


图3.2 返回给esp32的数据

### 软串口

由于串口屏和语音模块还有串口打印都要用到串口，串口不够用所以用到了这个软串口，需要调用SoftWareSerial库，然后直接用定义一个串口，名字自己定义，再传入两个参数分别代表RX，TX引脚，然后就有更多的串口可以用于串口通信。

### 串口屏

将串口屏收发串口定义为串口2，初始化串口波特率115200，开始连接串口屏的时候打印一下等待串口屏连接，提示马上连接串口屏，向串口屏发送数据的操作本文参考了这款串口屏官网的教程。首先定义一个空的char型变量来存放要发送的指令，然后将要发送的指令和要发送的数据用sprintf拼接起来放入这个char型变量中，再用串口2将这个指令发送到串口屏，这样数据就被传递到了串口屏，每发送完一个数据，都要发送结束符，所以这里把发送结束符这个步骤写成了一个函数sendEnd，这个函数没有参数，也没有返回值，官方操作文档说明了发送完一个数据需要发送结束符，如下图3.3。每次发送完一个数据，只需要调用这个函数就行。

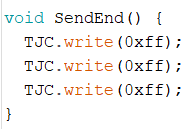


图3.3 sendEnd函数

### SYN6288语音模块

语音模块用的就是软串口，定义一个软串口，初始化串口波特率为9600，因为这个语音模块支持的通信波特率只有三种，分别是9600，19200，38400。然后判断语音模块初始化有没有成功，写一个判断，打印不同的信息，方便调试错误。在arduino框架里面要使用这个语音模块，需要将对应的需要播报的内容生成一个函数，并且这个函数是需要用厂家给的一个函数生成器来生成，函数生成器在下方图3.4，比如想播报一个“你好”，就需要提前将“你好”放在这个函数生成器，生成一个函数，再将这个函数复制粘贴到程序中，然后将这个函数中的串口，改为语音模块的串口，需要播报“你好”的时候，直接调用这个函数就可以了，这么做就会造成一个很大的问题，播报的东西都必须提前写好，播报的内容是死数据，但是这个疫情播报系统的数据都是实时的，不能写死，这个问题在之后的篇章里面会提到如何解，这里就是大概说明一下这个语音模块是如何工作的。

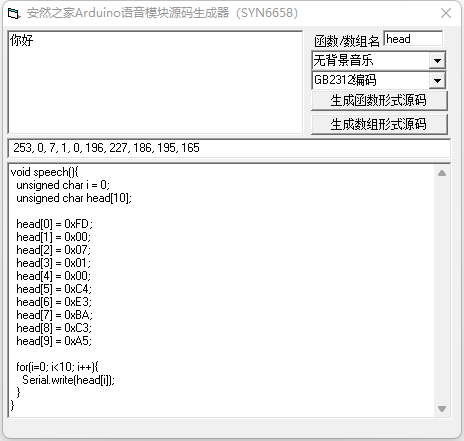


图3.4 函数生成器

## node服务器

使用node.js搭建一个服务器，用于接收esp32发送过来的网络请求，然后node服务器向外部接口发送网络请求，拿到数据，在服务器内部处理数据，将处理后的数据返回给esp32，同时要监听esp32发送过来的请求和前端网页发送过来的请求，需要写两个接口分别处理，由于esp32发送的请求可以设置一个延时，用来限制访问的间隔时间，不然对同一个接口短时间内访问次数过多，当前ip访问可能会受到限制，所以在提供给网页的接口的数据并不会直接来自向外部的接口，设计在对应esp32的接口里面将数据保存一份在本地，再用这份本地数据作为前端数据接口的数据来源，每次esp32发送请求的时候，本地的数据就会更新一次，所以也能保证数据是最新的。

### node简单介绍

Node.js本身运行于V8 JavaScript引擎[5]。运行时就是运行环境，node没有出现的时候，js通常是被当做客户端的程序设计语言使用的，一般是运行在用户的浏览器，node出现以后让js也能用在服务端编程。然而node现在的应用场景还不止有服务器，还可以被应用在开发脚本上例如webpack，electron等等。v8引擎本身的源码非常复杂，大概有超过100w行C++代码，但是这里可以简单了解一下它执行JavaScript代码的原理，如图3.5。

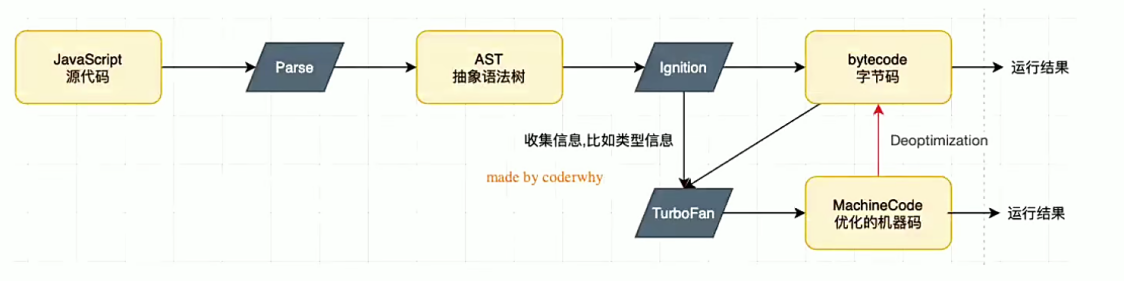


图3.5 v8引擎

图中的parse解析模块会把js代码变成抽象语法树，这是因为ignition并不认识js代码，ignition是一个解释器，会将AST抽象语法树转换为字节码，turboFan是一个编译器，可以将字节码编译为CPU可以直接执行的机器码，假如有某个函数被调用的次数很多，那么它就会被标记为热点函数。这个函数就会经过turboFan转换成优化的机器码，提高代码的执行性能。上面提到node.js基于v8引擎的JavaScript运行时环境，但是node.js不仅仅只有v8引擎，v8引擎可以嵌入到任意C++应用程序中，无论是谷歌浏览器还是node.js实际上都嵌入了v8引擎来执行JavaScript代码。另外在node.js中还需要进行一些额外的操作，比如文件系统读写，网络IO，加密，压缩解压文件等等操作，如图3.6是node.js的架构。

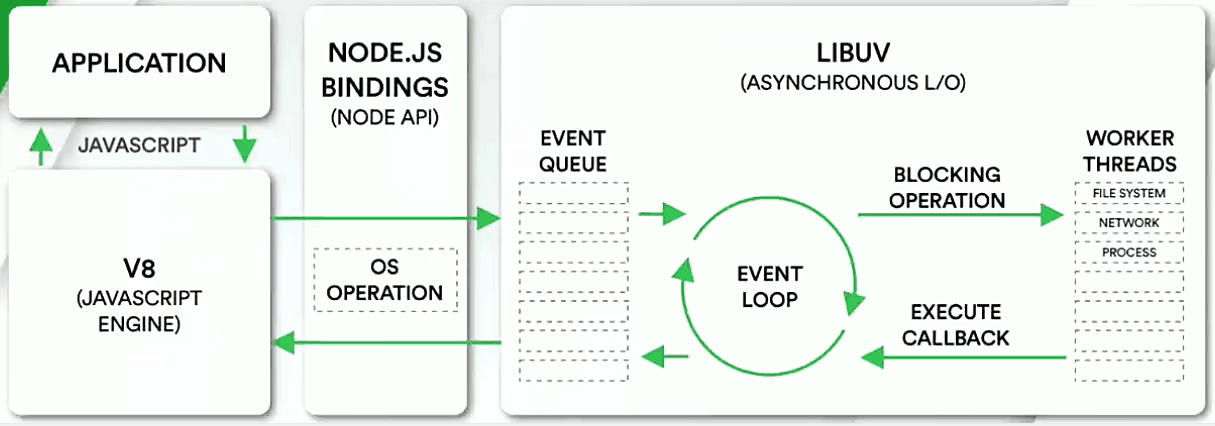


图3.6 node.js架构图

编写的JavaScript代码会经过v8引擎，再通过node.js的bindings将任务放到libuv的事件循环中，libuv是使用C语言编写的库，但他提供的功能不仅仅是io包括进程，线程等等。Libuv提供了时间循环，文件系统读写，网络IO，线程池等等内容。

### 原生node和框架比较

如果直接用原生node开发一个服务器，操作会比框架繁琐很多，就单单从http模块来说，先需要引入http模块，然后在http上面有一个函数叫做createServer，给这个函数传入一个回调函数，这个回调函数的参数有两个，一个是request另外一个是response，在这个回调函数中可以拿到这两个参数，并且做一系列的操作，比如响应一个字符串，数据响应完之后还必须调用一下end函数，不然客户端会认为数据还没有发送完，直观的感受就是网页的小图标一直在转圈，最后还需要启动服务器，绑定端口号和主机，这里的主机的参数可以不用传，默认就是0.0.0.0，然后还需要传入一个回调函数，当服务启动成功可以在这个回调函数打印一些标志信息提示服务器已经启动成功。这里就是一个非常简单的服务，如附录一。

在访问本机地址的3000端口的时候，就会返回数据。但是这个服务器很简陋，甚至分辨不出get请求，delete请求，put请求或者是post请求，为了分辨出此次请求的请求方式是什么，就需要用到reuqest这个参数，此次请求的请求信息会包含在这个request里面。为了看一下这个request里面到底有什么东西，这里模拟了一次请求，并且打印一下这个request。

模拟请求的工具为postman，之后会介绍到这个工具，request中包含的东西非常多，图中就包含了url请求路径和method请求方式。因为不同的请求方式有不同的作用，get请求一般是向服务器请求获取数据，post请求主要是进行一些表单提交或者是身份验证，比如登录一个网站需要账号密码，这里的账号密码就会放在post请求的请求体里面，后端解析到了账号密码再到数据库里面去查询，账号密码是否匹配。又或者是delete请求，一般是用在删除某些数据的，这类请求通常直接就在url后面带上一个需要删除某条数据的id，这样就实现了删除的功能。所以原生node需要做处理来能够分辨请求类型，不仅仅需要分辨类型，还需要分辨请求的url，因为同一个url对应的不同的请求类型，对应的功能也是不一样的，例如/user，当这个请求类型是get的时候，可以写为获取某个用户信息，当请求类型是post的时候，可以是注册用户，delete可以是删除当前用户，所以不仅仅要分辨请求类型，还需要同时分辨请求的url，这样就会写非常多的判断来解决这种问题。

但是如果用框架的话这种问题就非常好解决了，node的框架有很多，如果只是用作服务器开发的框架的话，主流就是三种，express.js，koa.js，egg.js，然而这三个框架中有两个框架都是由一个人开发的，TJ Holowaychuk，不过当前他已经不再维护express框架了，将express交给了他的团队维护，他自己的精力转向了koa的开发和维护，因为两个框架都是由他开发，所以API的风格非常的类似，但是koa旨在更轻量级。这里本系统用了express，下面是用express解决路径和请求方式的问题，如附录二，这样很见名知意，比用原生node还要写一大堆的逻辑来判断方便多了。node.js+express框架技术已经发展为比较成熟的基于web的软件开发应用技术[6]。

原生node来开发服务器还会有一种比较典型的问题，就是文件上传。当用户选择了一个文件需要上传到服务器，在reuqest里面是不能直接拿到数据的，但是request有一个方法on，第一个参数写data，第二个参数是回调函数，这个回调函数就能拿到data，但是由于数据是二进制的，并且在这个二进制数据里面不止包含这个文件本身，还有描述这个文件的一些信息也会被放在这个data里面，比如这个文件的名字叫xxx.png，那么在data里面必然有描述这个文件的名字的数据，正常情况下应该是拿到纯文件的数据，就是这个data里面应该只包含此文件的二进制数据，然后将这个二进制数据转为想要的形式，所以必须要去除很多在data里面不属于文件本身的内容，这里要做的操作就很复杂，就不一一描述，如附录三。

如果用框架来处理的话会非常方便，只需要用一个中间件，需要安装一下multer，这个中间件是express官方提供的，专门用来解决这个问题。然后在request.body中就能够直接拿到想要的数据，再对数据做相应的处理就可以了。

### postman

Postman软件为专业接口测试工具[7]。如果不用这个工具，测试接口的话需要自己写发送请求的代码，挺麻烦的。这个工具可以直接发送请求，可以设置请求类型，请求路径，还能携带请求体，文件之类的也支持，在下面还能查看响应内容之类的，是一个很方便的工具。如下图3.7。

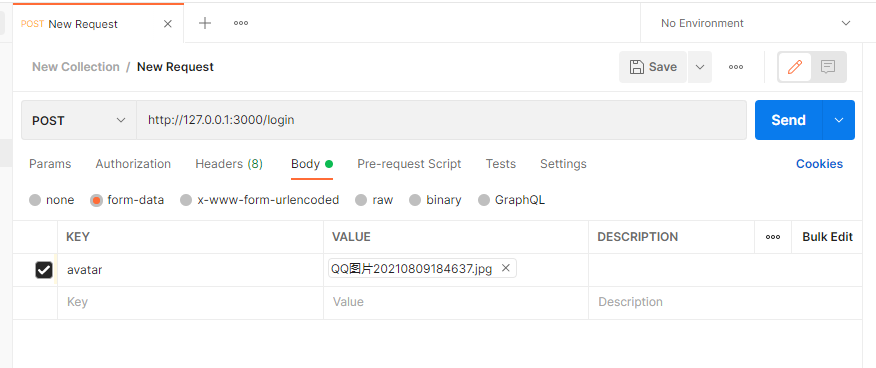


图3.7 postman工具

### 服务器搭建

服务器主要的作用为当收到esp32发送请求的时候，向外部数据接口发送数据，并且将这个数据处理，拿出esp32需要用到的一部分，返回给esp32，数据的格式为对象，对象里面用一个个键值对来存储数据。之后将这个外部接口返回的完整数据用JSON格式保存在本地一份，写另外一个接口为前端提供服务，这个前端的接口就用到了保存在本地的JSON数据，因为前端网页和esp32不一样，需要发送请求的次数会很多，为了避免外部接口因短时间内同一个ip地址发起多次网络请求导致当前ip地址被屏幕，所以将前端接口的数据源用本地JSON数据做代替，不过这个数据也是最新的，只要esp32发送了一次请求，本地的JSON数据就会更新一次。服务器用到了上面提到的express框架，可以快速搭建一个服务器，先引入express，然后引入axios。Axios是基于promise的HTTP库，用在浏览器和node.js中[8]。主要就是用于发送网络请求的。还要引入fs，用于文件的操作。fs模块为node.js的核心模块之一，能够做到文件读写，复制，删除等等操作，使用的时候需要先导入fs。需要把请求下来的数据利用fs同步保存在本地一份，然后引入一个utils工具函数，这个函数是自己封装的，根据接口数据的结构来封装的一个工具函数，这里用到了js的模块化，因为要是将所有的东西全部都放在一个文件里面，最后这个文件里的代码会很多，不利于维护，上面提到的内容如图3.8。

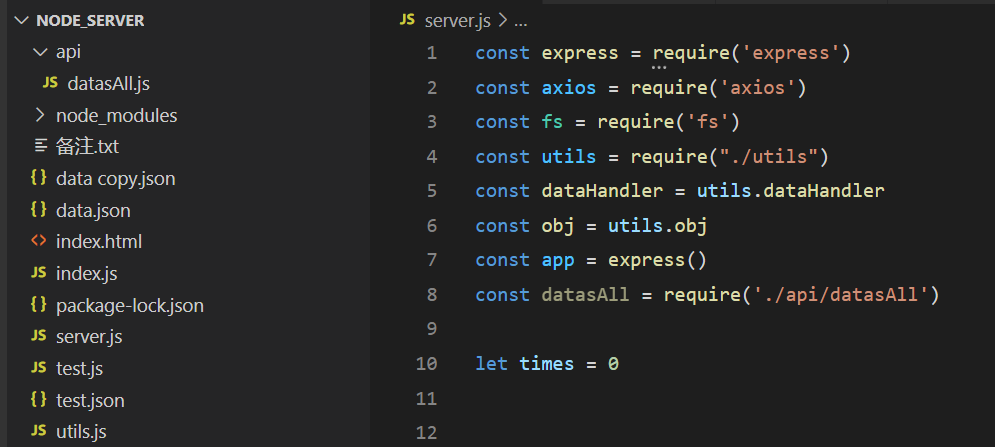


图3.8 js模块化

然后将请求类型设置为get请求，请求路径为/，这个接口就对应了esp32发送的请求，然后发送网络请求向外部的接口，当拿到了数据之后，就同步地将数据写入到data.json文件里面，如果没有这个文件的话，就会在当前路径创建一个data.json文件。然后将这个文件里面的数据再用fs同步读取出来，因为将文件写入的时候是用了JSON序列化的，所以使用这个数据的时候应该JSON反序列化，然后将这个数据当做参数传入函数dataHandler，这个函数是在utils.js文件里面封装好的，因为utils.js里面不止一个方法。

这个工具函数的主要的内容就是将外部接口数据中想要展示在串口屏的数据选取出来，还有时间戳需要格式化，最后用正则表达式取出数字存放在数组中，根据数组的索引值将对应的时分秒作为变量保存起来发送给esp32。

之后就是网页部分的接口实现，网页部分的接口实现就简单多了，直接读取一下本地的json文件，然后将这个文件返回过去就行了，因为本系统在前端部分可以自由操作这里的数据，所以数据不用做处理，只需要在前端处理数据就可以了。

## 前端网页

网页主要用了下面的技术，vue框架，elmentui，echarts，下面是这三个技术的简单介绍。Vue.js框架是一种面向前端界面开发的轻量化框架，此种开发框架的最本质的特点是具有响应式编程和组态化的特点[9]。Element UI是基于Vue2.0开发的桌面端组件库，由于它组件丰富，样式较多，风格统一，容易上手，代码稳定，且提供多种组件属性和触发事件列表，所以对于使用者使用该组件库进行再开发比较友好[10]。ECharts是一款基于JavaScript的数据可视化图表库，可以提供直观、生动、可交互、可个性化定制的数据可视化图表，其中拥有大量开源的可视化图表文件，可以提供用户下载使用[11]。

### Vue框架

在过去的十年内我们的网页变得更加强大和动态化，多亏有了JS，但是传统的网页形式是js连接了各种各样的html，css缺乏正规的组织形式，像结构很庞大的网页中，数据和视图部分全部混在一起要处理和维护代码就会非常费劲，并且如果某些结构之前存在某些联系，就会造成牵一发而动全身的效果，vue的官网说它是一款友好的，多用途且高性能的js框架，vue允许将一个网页分割成可以复用的组件，其中每个组件都有独立的html，css和js，用来渲染网页中对应的地方，并且vue是响应式的，只要数据发生了改变，页面中数据也会随之改变。

### ElementUI

ElementUI就是基于vue的一个UI框架，该框架基于vue开发了很多组件，主要是为了方便开发人员快速开发页面，这些页面写原生css实现起来会比较麻烦，所以都被封装成了组件。

### Echarts

Echarts是一个使用js实现的开源可视化库，可以流畅运行在PC和移动设备上，绝大部分浏览器都是兼容的，提供了常规的折线图，柱状图等等，还有很多地理数据可视化的地图。

### 网页搭建

先用vue脚手架搭建一下项目，然后再安装上面提到的elementui，echarts。前端开放框架和环境都是需要node.js的，为此首先需要先安装node.js，并且node服务器的开发也是需要node.js。首先去官网下载node.js，域名是nodejs.org。安装完成之后可以打开cmd查看一下是否安装成功，输入以下指令node --version，如果出现了该命令不是内部或外部命令的字眼，可以先将命令行关闭，重新打开并且用管理员模式运行，如果还是不行的话，就是环境变量没有配置好，配置一下环境变量就可以了。安装完node之后就可以使用npm包管理工具了，npm在安装node的时候就会自动安装，因为项目要用到vue框架，所以可以直接npm install vue来安装vue框架，也可以用这种方式安装echarts，elementiui或者是axios。当安装完vue的时候还不算完，如果想快速搭建一个项目需要用到vue的脚手架，vue-cli，这个脚手架可以帮助快速搭建一个项目，只需要用npm安装脚手架就可以了，现在脚手架搭建项目一般都是用3版本，这个版本新建项目的指令为在cmd中输入vue create再加上一个你的项目名字，之后就可以快速帮助搭建一个项目，之后会弹出很多选项，比如vue框架选择2版本还是3版本，或者是需不需要用router还是vuex，这些东西可以选择，选择之后就会进行项目初始化了，使用脚手架的目的主要是不用手动去配置很多东西，例如webpack一个打包工具，package.json主要用于管理一些第三方库的版本，或者是一些文件夹用于存放静态文件，这样就可以专注于业务，不必为了一些环境的配置分心。这样就简单搭建了一个vue项目了，下面就是这个项目中各个模块的实现。

网页分为几个部分，第一个是疫情日历，点击对应的日期可以显示当天的疫情数据，具体的当日确诊，累计确诊，累计死亡，累计输入和今日输入，如下图3.9。

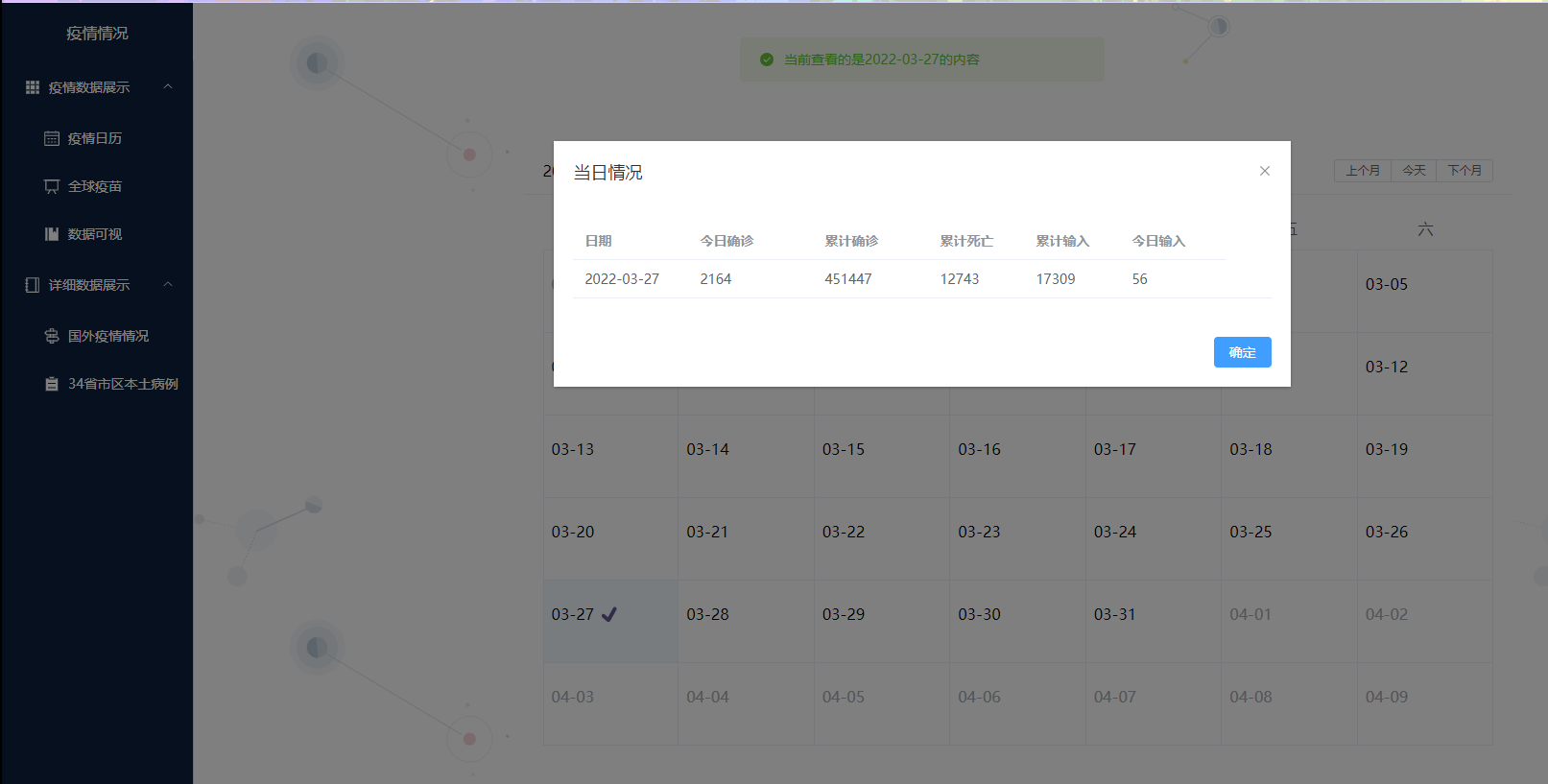


图3.9 疫情日历界面

这部分页面的代码在view文件夹中的Home.vue文件，这个页面主要是用了elemenui的一个日历组件，然后找到对应的每一天的组件位置绑定一个点击事件并且触发一个回调函数，并且将当天的日历的当天信息传入这个回调函数中，例如我点击了日历中的某一天，然后将这个传递过去的参数打印一遍，里面有一个day的属性，如图下3.10。



图3.10 参数打印

这样拿到这个日期之后，再到请求下来的那个数据中去查找对应天数的数据，然后发现很巧数据中的日期和这个day是同一种格式的，不需要转换再对比，直接就可以对比这个日期是否相同，如果相同将当天的数据放在当天的表中。数据请求的位置在这个组件挂载完成的时候就发起了，这里用到了一个vue中的东西叫做vuex，这是一个数据管理的地方，大概的意思就是在内存中有一块地方存放着你的数据，然后你随便哪个组件的位置想用这个数据都可以获取到，在组件挂载完成的时候发起了发起了数据请求，并且将需要的数据都存在了vuex里面，然后在当前页面组件里面用一个变量chinaDayList保存一下从vuex里面获取的数据，在点击对应的日期显示的数据就是来自这个chinaDayList变量。

之后一个部分是全球疫苗接种数，如图3.11。

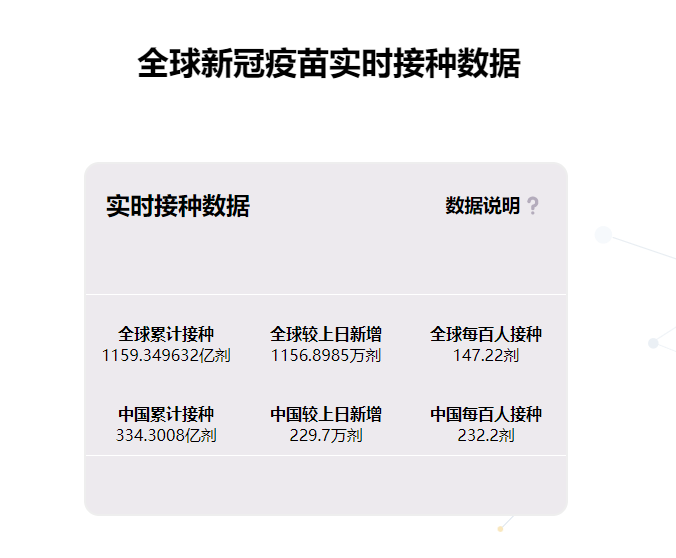


图3.11 疫苗接种数据

这个数据的接口是在腾讯疫情的一个网页里面找的，这个疫情展示网页为https://news.qq.com/zt2020/page/feiyan.htm#/，接口是在这个网页的network里面找到的，如图3.12。

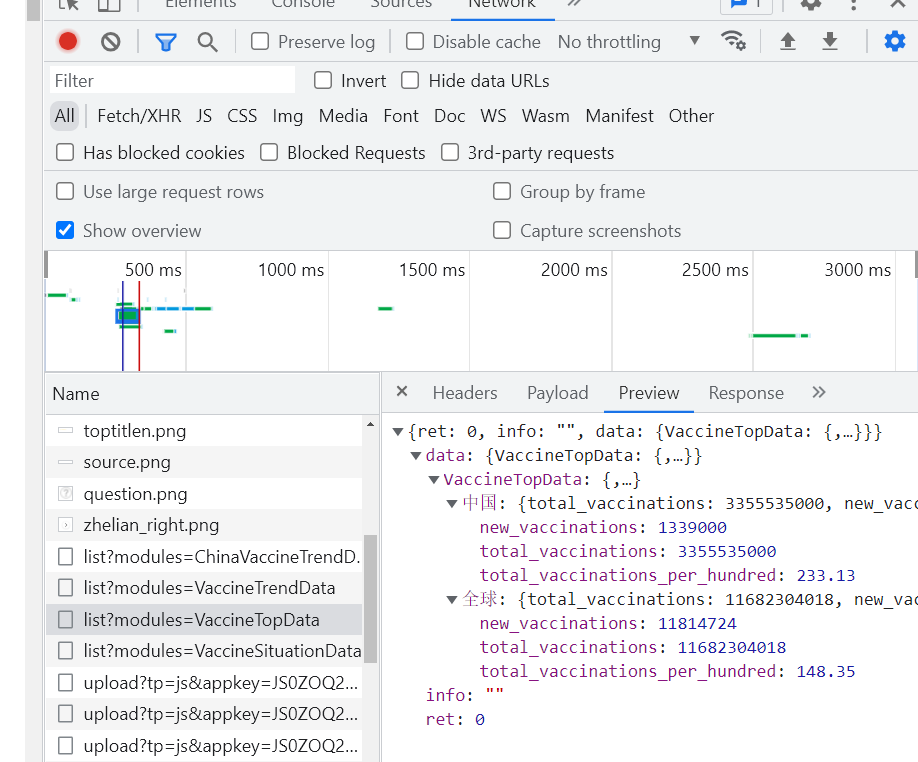


图3.12 疫苗数据接口来源

数据是实时的。这里的数据来源直接就是从外部的接口获取的，下面是获取获取数据的代码，如附录四。

这里的getVaccines函数是一个发送网络请求的函数，async和await是将异步函数用同步的方式写，它封装在network文件夹下面的request.js中，类似于之前提到的utils.js，里面也封装些一些函数，主要是发送网络请求的函数都封装在这里面如附录四。

这里的属性是中文，因为外部接口的数据字段就是中文，下图是这个疫苗接种数据接口，图3.13。

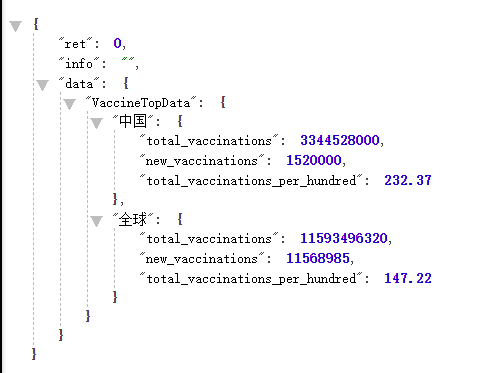


图3.13 疫苗接口数据

下面一个部分是数据可视化，主要是以当日新增确诊人数的数量来分阶段标记的地图，如下图3.14。

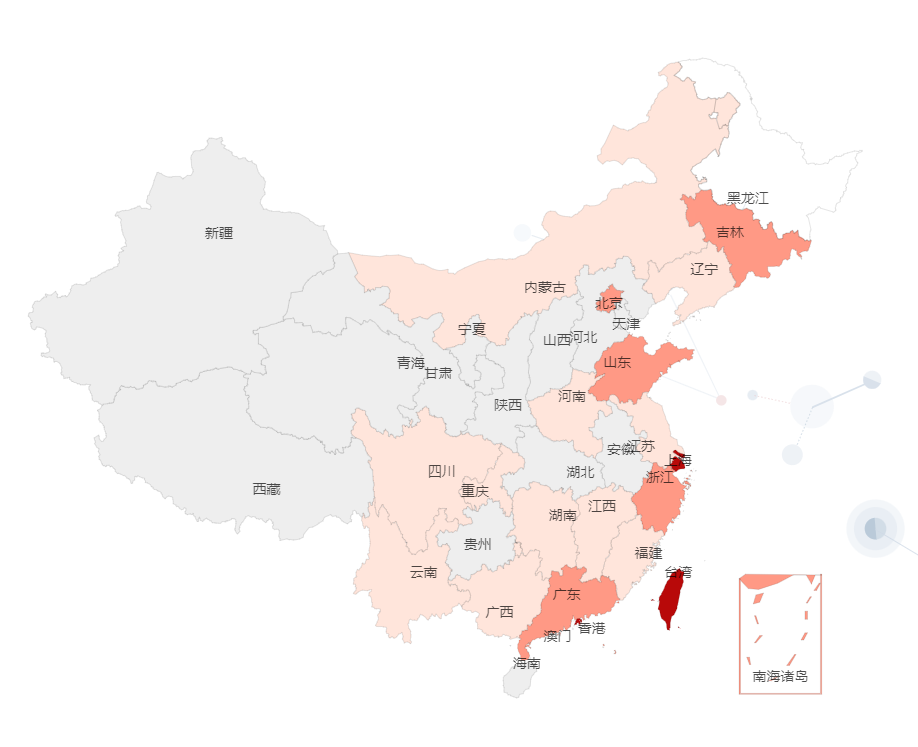


图3.14 当日新增可视化

这个部分主要是用了echarts中的地图插件，这个地图中的数据获取的方式稍微有点不一样，这个数据是在浏览器的localstorage中获取的，localstorage是浏览器中用来存储客户端临时信息的对象，在之前获取数据的时候，获取了对应的数据，主要是34个省份的数据作为一个数组存放到localstorage中，然后要用到数据的时候再取出来就行了。因为做这个网页的时候发现有些时候页面刷新了，vuex中的内容会丢失，所以数据会丢失，所以将数据备份了在localstorage中，既可以通过vuex获取也可以通过localstorage获取数据。localstorage在浏览器的这个位置，如下图3.15。

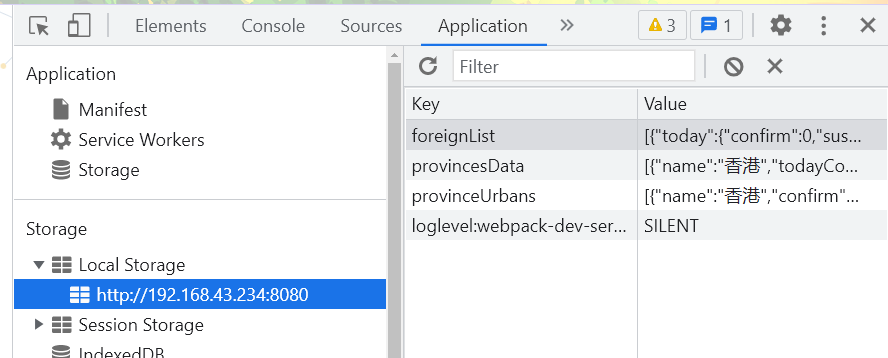


图3.15 浏览器中的localstorage

然后是国外疫情情况，国外的情况主要就是做了一个列表来展示数据，只能看到对应国家的大致情况如下图3.16。



图3.16 国外疫情

这里数据来源也是来自vuex，由于考虑到刷新的问题，所以还做了一个判断，如果当前的数据数组的长度为0，那么就在localstorage中获取数据。

最后是34省市区本土病例，如图3.17。然后还做了一个详情页，每个省份的详细情况也可以查询，如图3.18。

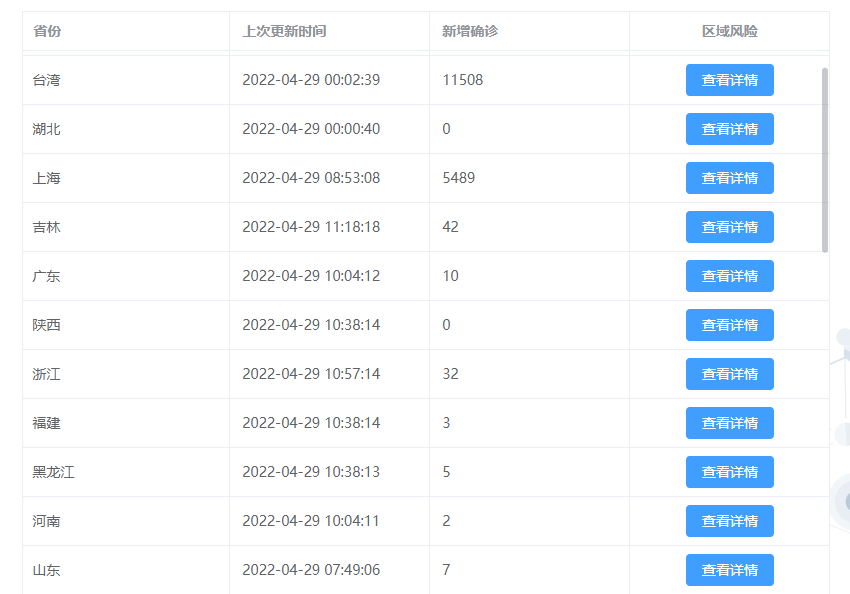


图3.17 国内疫情



图3.18 各省疫情详情

这个部分用到了两个组件，一个用来展示省份数据一个用来展示省份内具体数据。数据的处理和上面的类似，只不过这里用到了一个东西叫做路由，当点击图3.17中的查看详情时，会跳转到另一个组件图3.18，这里用到了vue中的router。首先需要将写一个router配置的文件，里面写上路由映射关系，如附录五，一个路径对应着一个组件。

然后在需要跳转路由的时候需要先引入router，这个是vue框架里面已经封装好了的，然后在这个查看详情按钮绑定一个点击事件，当点击的时候触发一个回调函数，这个函数里面进行路由跳转操作，router有一个方法是router.push，将需要跳转到的路径写入，就可以跳转到对应的组件，如附录五。整个页面的跳转也是用的这个路由。

## 串口屏

串口屏的上位机软件为USART HMI，打开这个软件之后进入页面如下图3.19。

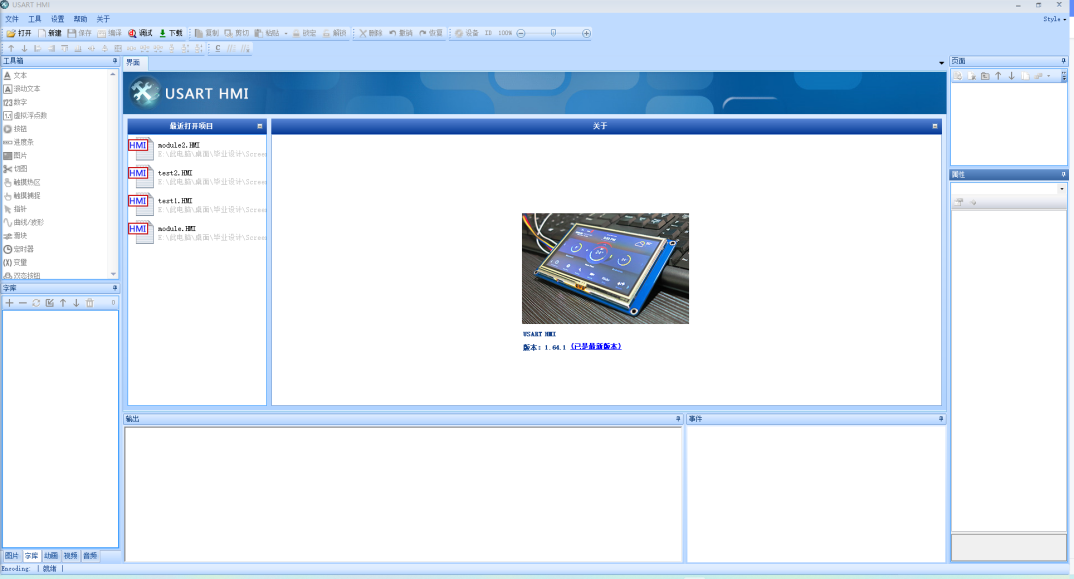


图3.19 上位机软件

然后新建一个项目，新的项目里面需要给这个项目添加字库，由于添加很多字也用不到所以这里我选择了指定字符，如下图3.20。在这个字库里面将需要用到的汉字都添加进去。



图3.20 字库制作

添加了字库之后就是页面的绘制。主要用到了图3.21中的数字组件和文本组件。数字组件一般用于变量，文字组件一般用于一些固定的内容，例如当天确诊的人数就需要用数字组件，而当天确诊这四个字就是固定的内容。



图3.21 组件

并且点击某些按钮会触发事件，主要触发的事件类型就是页面跳转。如下图就是一个页面跳转，点击一下需要触发页面的组件，这个组件就可以写一些事件，分为按下事件和弹起事件，在按下这个组件的时候需要的逻辑就可以在按下事件里面编写，需要在组件弹起之后再执行的命令就在弹起事件里面编写。如图3.22，这里就在按下事件里面编写了一个跳转至page3的指令。当按下这个组件的时候，页面就会跳转至page3，这个页面需要是整个串口屏项目里面自己编写的页面。

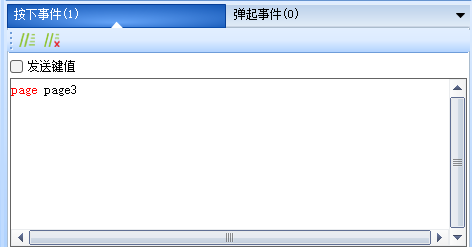


图3.22 事件编写

另外需要注意的还有一点就是作用域，每个变量的作用域需要设置为全局作用域才能够拿到数据，如下图3.23这点也是官方文档指出的。



图3.23 变量作用域

最后在这个上位机写完后的项目如下图3.24， 3.25。

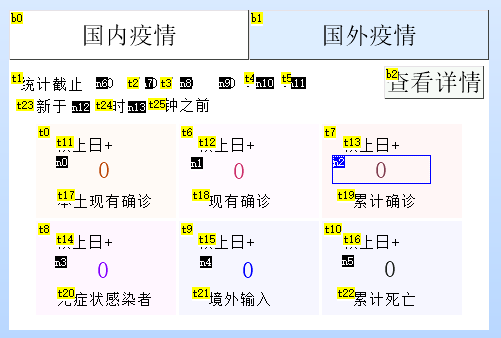


图3.24 串口屏结果



图3.25 串口屏结果

## SYN6288语音模块

上面也提到过，如果想在arduino框架使用这个语音模块，必须使用对应的函数生成器，考虑到需要播报的内容有一部分内容是死的，有部分内容是活的，比如说播报内容为，本地确诊100人，除了这个数字，其他内容都是需要变动的，所以就将文字部分先用函数生成器生成对应的函数，声明在esp32程序内，需要调用的时候直接调用一下，然后就是写一个函数，这个函数需要的参数为一个数字，然后实现的效果是播报这个数字，可以将一些数字的读音作为函数先声明出来，例如123，先把2，3，10， 100的语音函数写出来，对这个数字先%10取出个位，再用这个数字除以10再%10，取出十位，以此类推取出百位，然后对最高位开始，写很多判断，如果百位等于1，在这个if语句里面调用一下100语音函数，然后判断十位，如果十位等于2，在这个if语句里面调用一下2的语音，紧跟着调用一下10的语音函数，因为10这个数字的读音在这个语音模块中读的是一十，而不是直接的十，这样才符合数字的正确读法，然后看一下最大的数字是多少，根据这个最大的数字来写这个函数需要播报的数字范围，如图3.26。

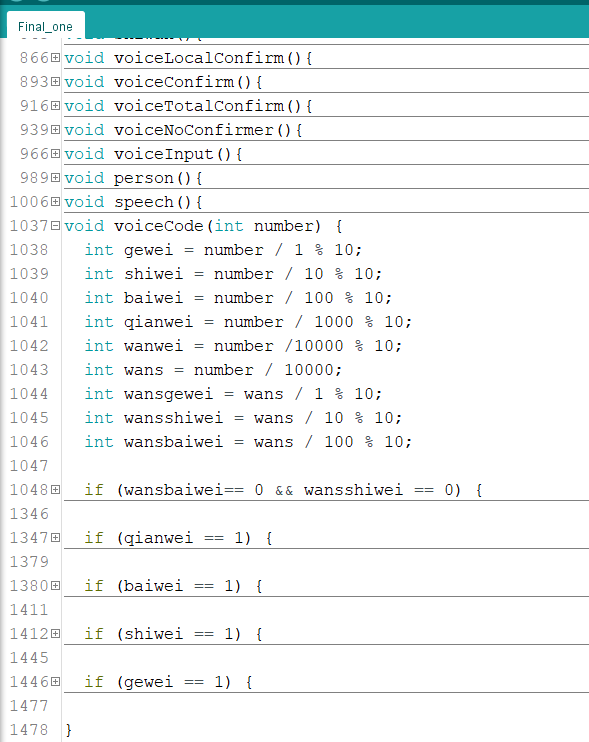


图3.26 语音模块逻辑

# 测试部分

## 软件测试

先测试一下服务器有无问题，先将服务器启动，之后用postman工具模拟发送请求，首先测试的是esp32的接口，顺利得到了数据，并且在服务器下方打印了一次信息，这个信息是用来提示的，如图4.1。

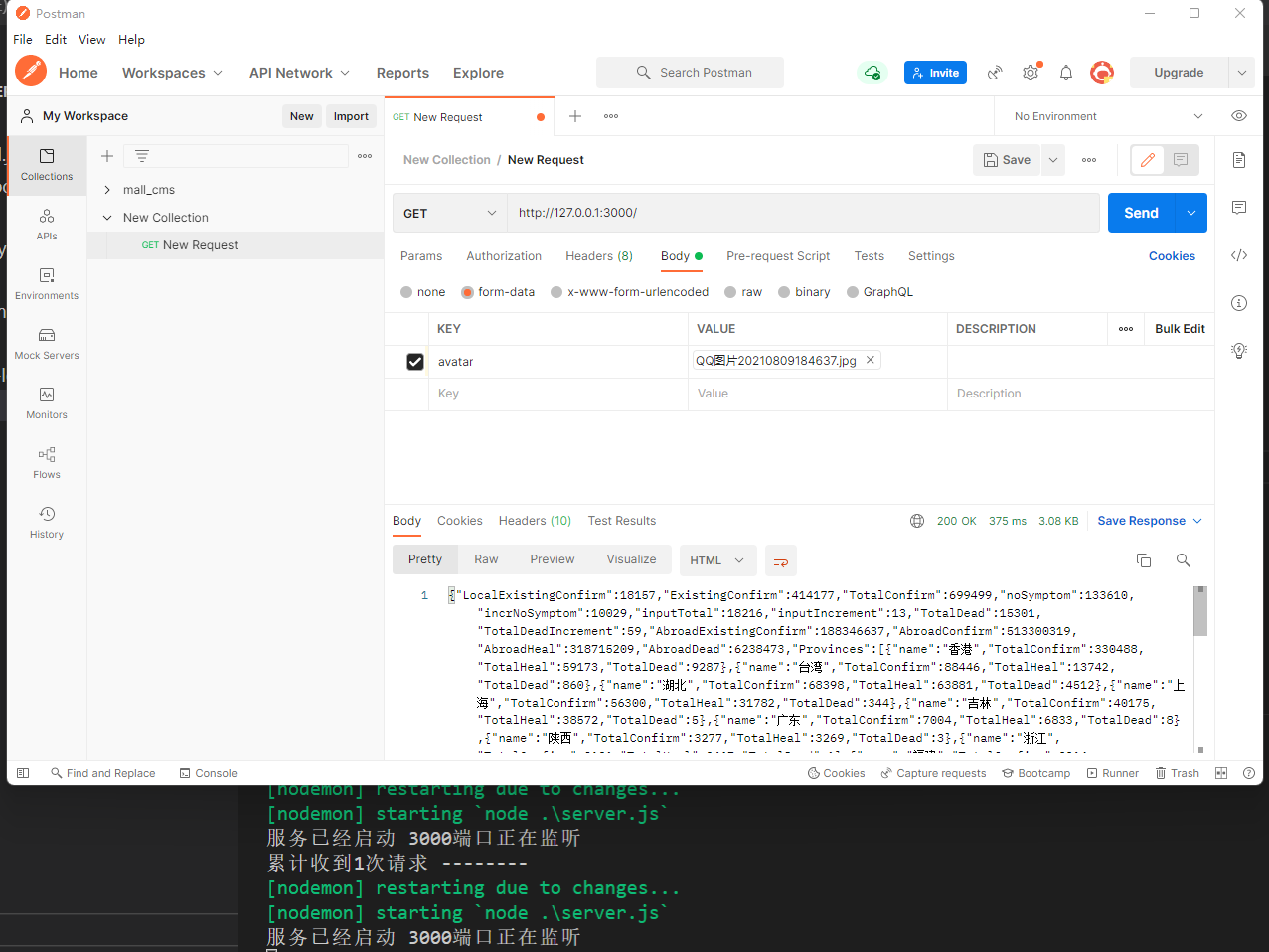


图4.1 测试esp32接口

然后测试的是前端网页接口，也顺利拿到了数据，如图4.2。

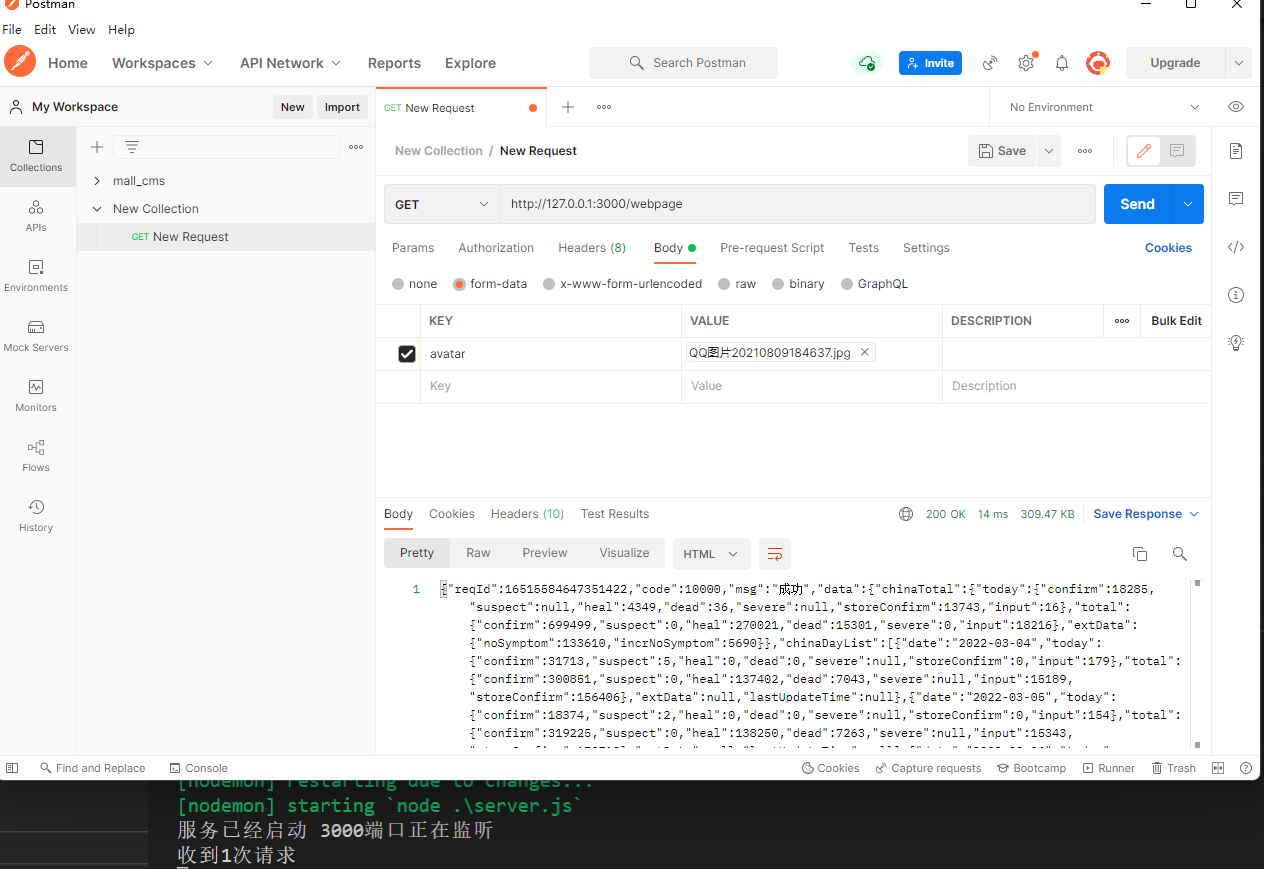


图4.2 测试网页接口

之后测试网页能不能正常发送请求。如图4.3，结果可以正常获取数据，并且能够将数据正确展示。

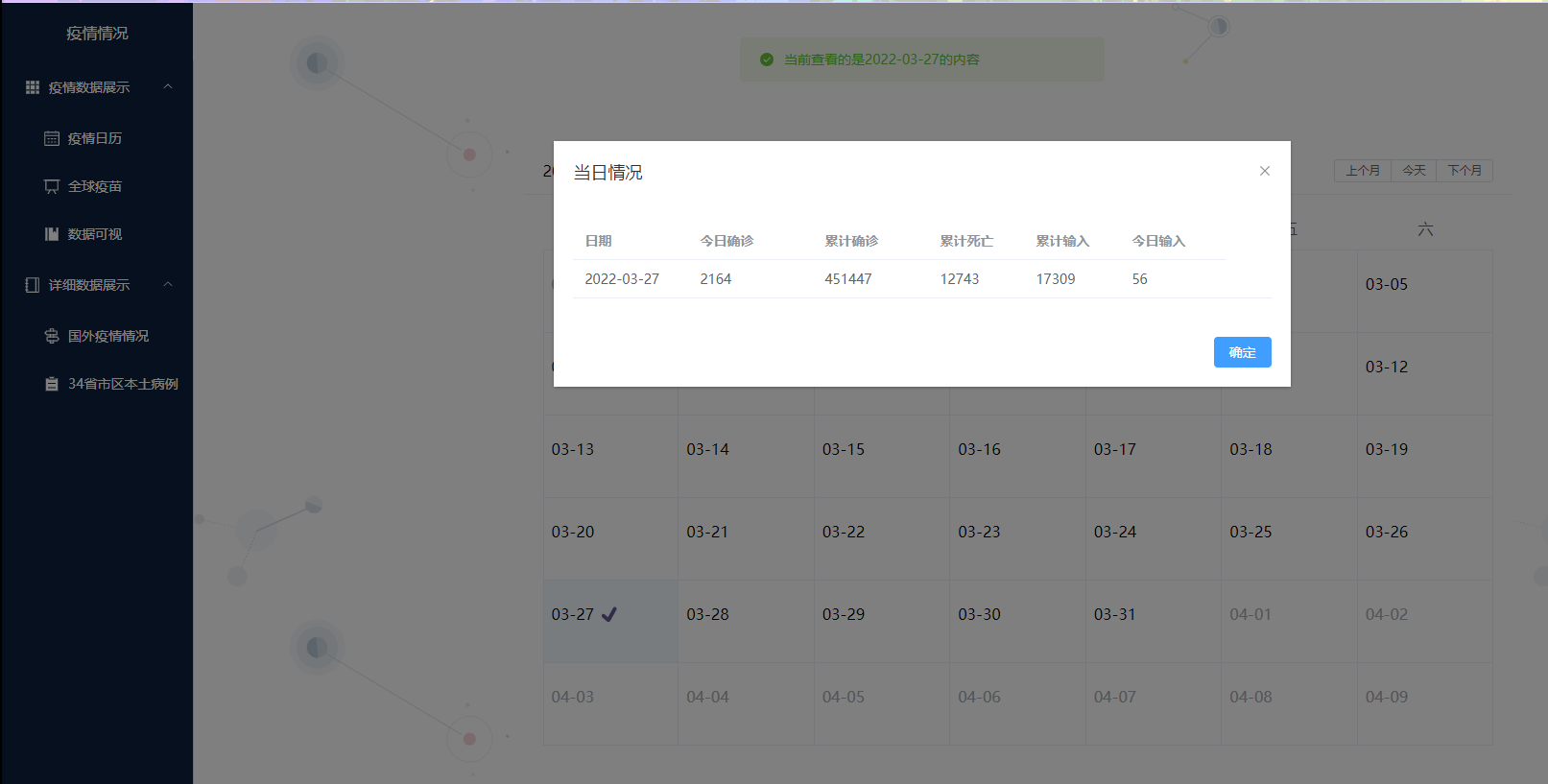


图4.3 测试网页

## 硬件连接

将esp32的+5v管脚连一根杜邦线到面包板，因为语音模块和串口屏都需要+5v供电，但是只有一个+5v管脚。串口屏和语音模块的GND接esp32的GND，串口屏RX接esp32的17号管脚，TX接16号管脚。语音模块的RX和TX接esp32的22和23管脚，是软串口。串口屏和语音模块的VCC都接在面包板上，下面是实物图4.4。

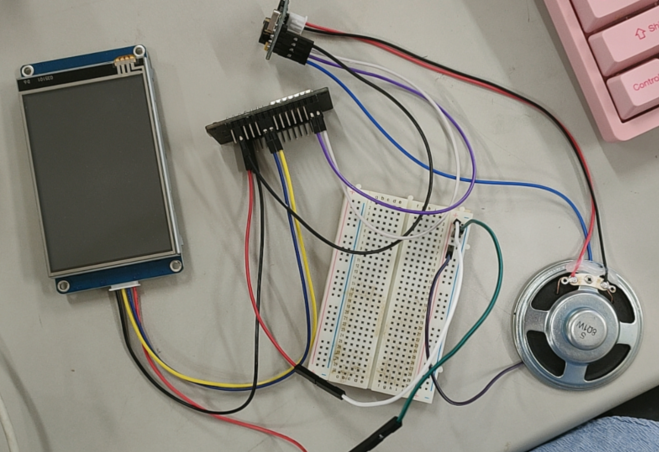


图4.4 硬件连接

## 遇到问题

### Wifi连接不成功

如图4.5，由于之前给的延时太短了，导致留给wifi连接的时间太短，还没有连接上wifi就进入下一次循环了，后面将这个延时写大了，wifi就能顺利连接了。

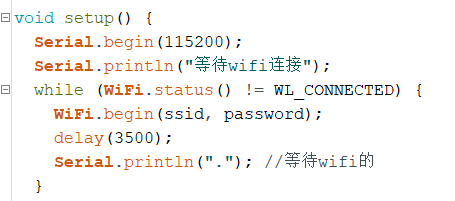


图4.5 wifi连接问题

### Json数据展示不完全

之前在创建这个doc对象的时候随便给了一个值，然后测试的时候，发现拿到的数据有一部分是null或者0，始终拿不到值如图4.6，最后才想起是创建这个对象给的值不够，最后加大了这个值，数据能够完全拿到了。

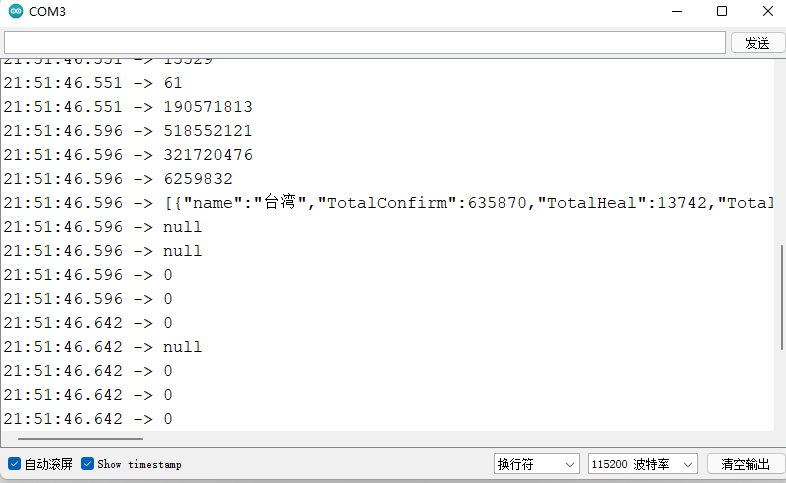


图4.6 拿到某些数据为null或者0

### 语音模块

语音模块遇到的问题就是上面提过的如何播报一个变量。

### 请求失败

导致网络请求失败的原因基本都是请求地址不对，有时候测试的时候是用手机开热点测的，有时候直接连接wifi测，所以上次测了热点下的下次在wifi下测就请求不到数据，后面才想起ip地址变了，拿不到数据。还有一种情况是手机开热点的这个ip也不是固定的，是由手机的DHCP服务自动分配的，所以有时候也会出现上次也是用的热点，下一次就请求不到数据的情况。

### 跨域问题

跨域指的是浏览器不能执行其他网站的脚本，是浏览器对js实施的安全限制。当一个请求的url的协议，域名，端口，这三者中的任何一个与当前页面的url不同就称作跨域。跨域只会在浏览器和服务器之间才会存在。由于前端项目需要占用一个端口，后端服务器也需要占用一个端口，他们不能是同一个端口，所以这里端口号不一样会造成跨域。这里解决跨域的办法用到了webpack中的devServer服务器，这里将所有的网络请求先发送给这个devServer服务器，因为这个服务器是在node本地的，所以不存在跨域问题，然后由这个服务器向node服务器发送请求，因为是服务器向服务器发送请求，所以这里不会存在跨域问题。

## 联调

先开启服务器，然后电脑usb口给esp32供电，刚开始没有拿到数据的情况如下图4.7。

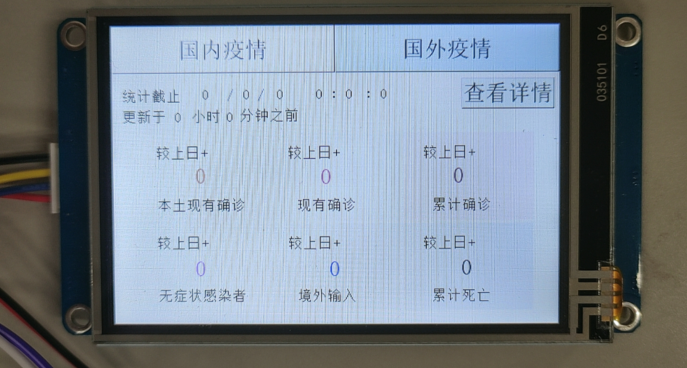


图4.8 没有拿到数据情况

然后查看一下串口打印图4.9，这里能看出wifi已经连接成功，串口屏连接成功，语音模块也连接成功了。

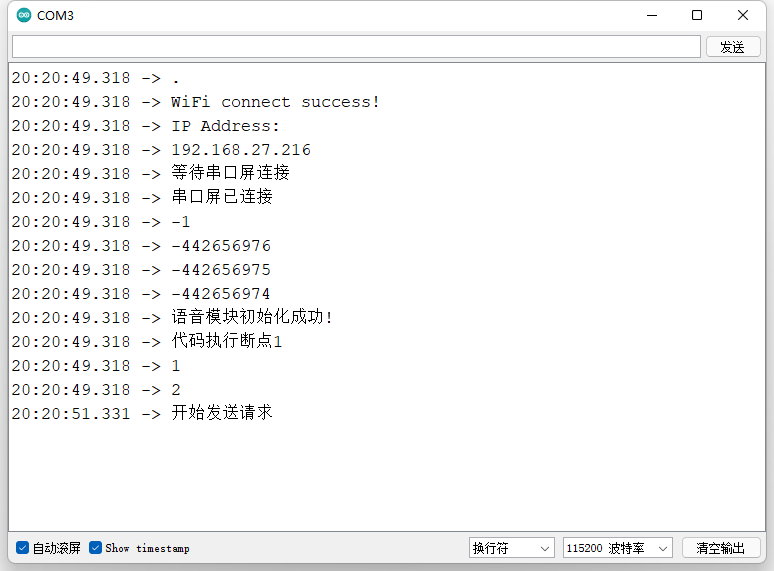


图4.9 串口打印信息

拿到数据之后，串口屏的展示是这样的，下图4.10，4.11。

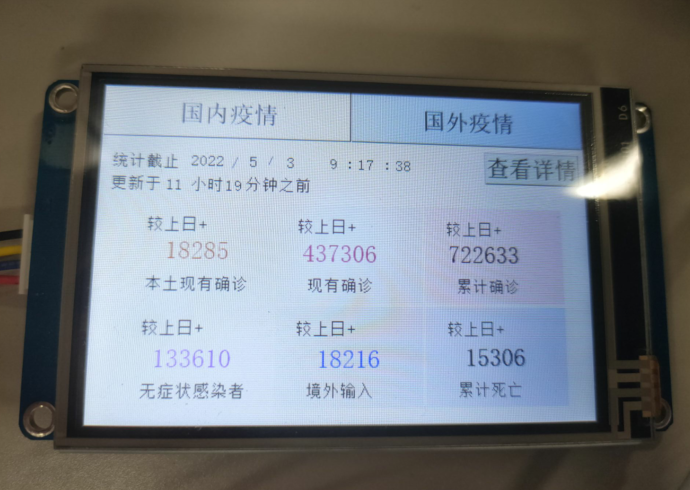


图4.10 拿到数据情况

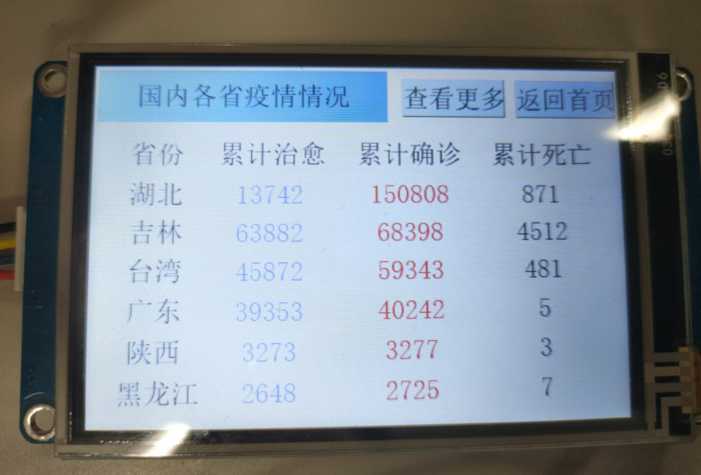


图4.11 拿到数据情况

下面是前端网页展示内容，图4.12是疫情日历的页面，4.13是点击某一天，弹出详情的页面。前端网址为http://127.0.0.1:8080, 127.0.0.1是本地地址，这个地址还可以换成当前ip地址，8080是前端网页的端口号，如果本机以外的设备想要访问这个网页需要在同一局域网下，且访问网址中的ip地址为本机网络的ip地址。

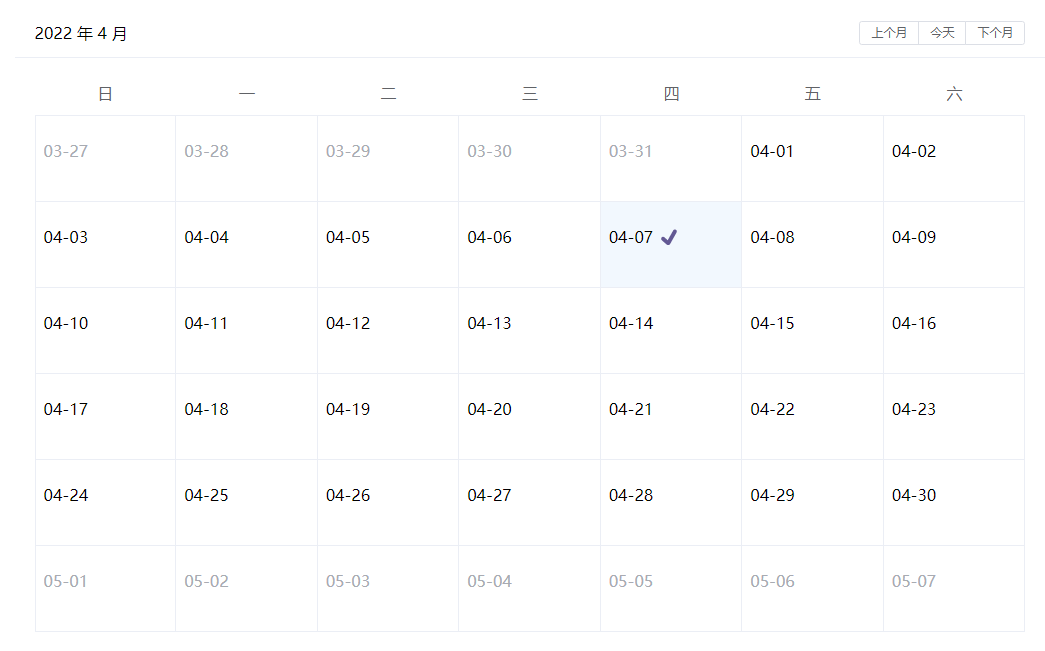


图4.12 疫情日历

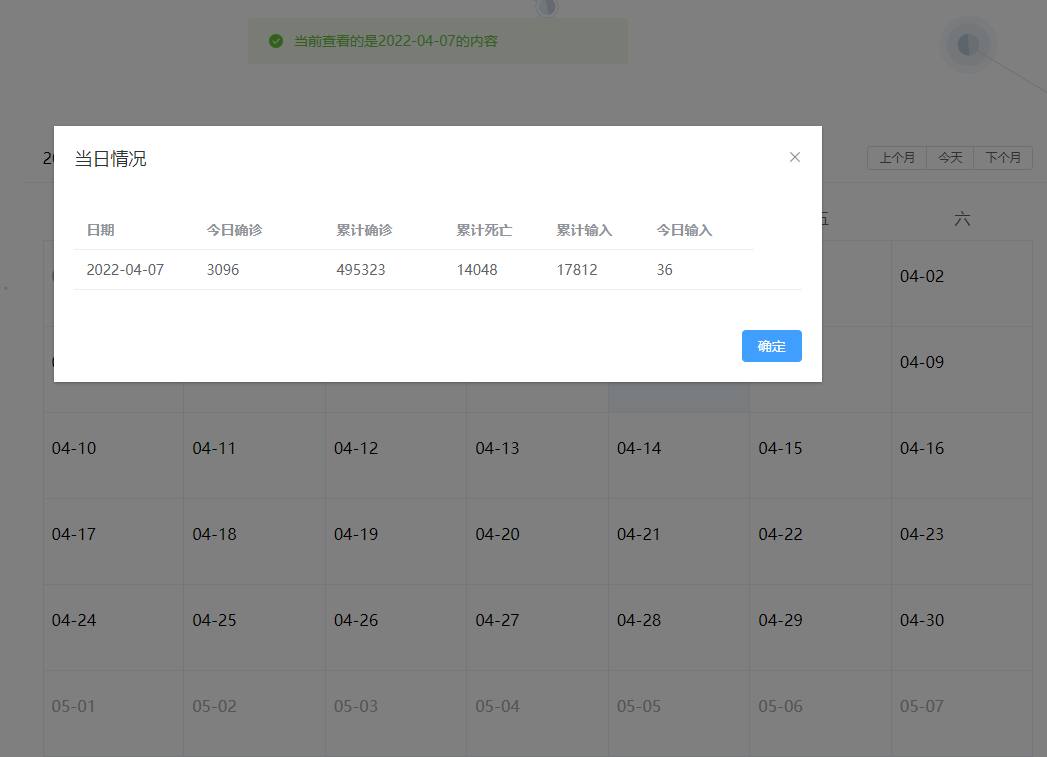


图4.13 疫情日历某日情况

下图分别是全球一面接种情况和数据可视化页面，图4.14和图4.15。



图4.14 疫苗接种数据

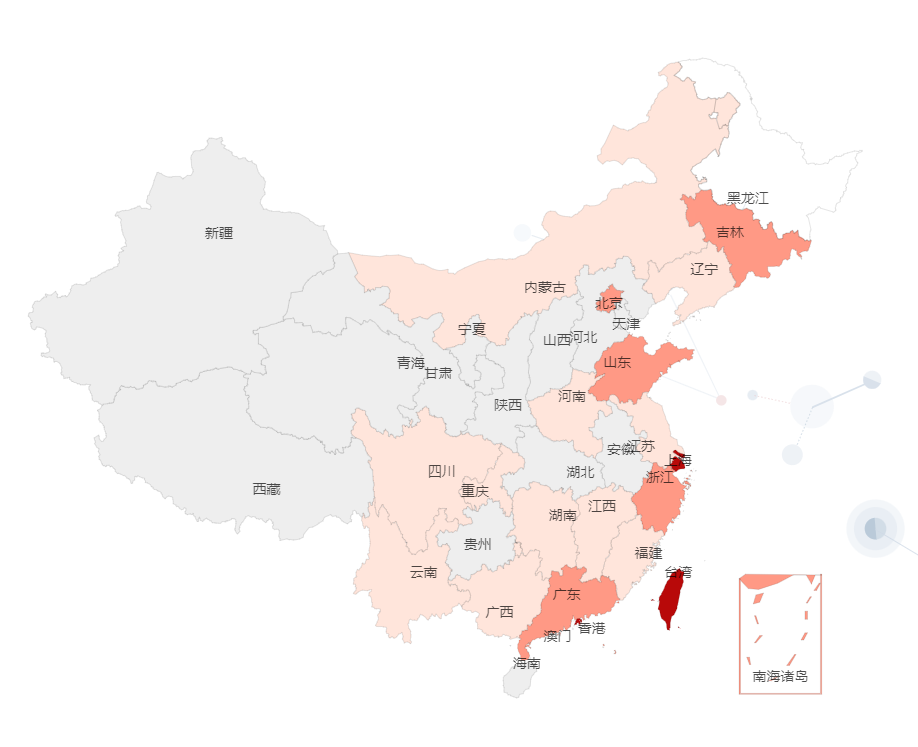


图4.15 新增确诊数据可视化

下面是国外情况展示和三十四省和市区展示。如下图4.16为国外疫情情况展示，因为国家太多就只截取了一小部分，图4.17为国内三十四省各省的疫情总体情况，图4.18为具体每一个省份中各个市的疫情情况。



图4.16 国外疫情

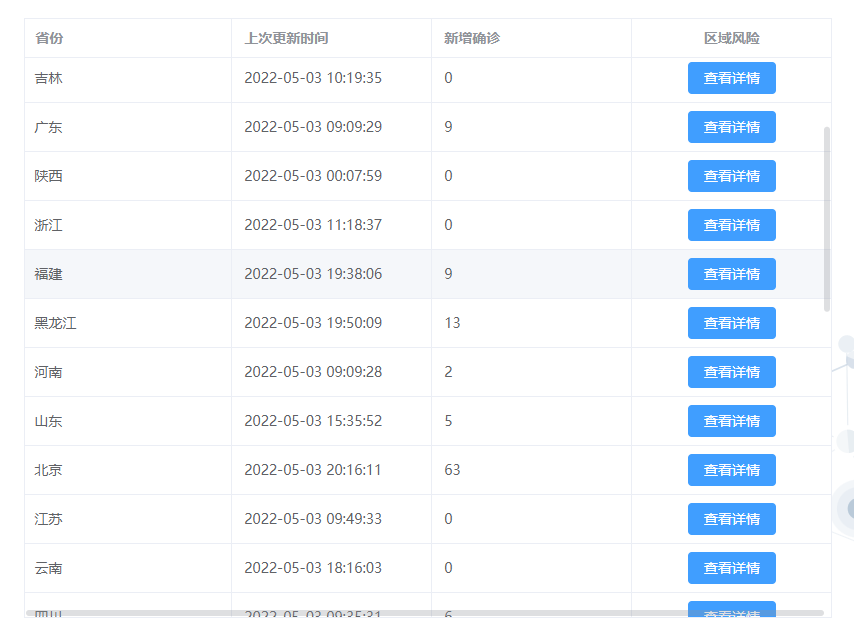


图4.17 国内三十四省疫情



图4.18 各省疫情详情

结 论

本文基于物联网技术，以esp32为主控芯片，发送网络请求获取数据。syn6288语音模块实现语音播报功能，串口屏用作数据展示。Node.js为基础搭建服务器，用vue框架实现前端网页搭建，elementui，axios，echarts为工具。经过试验，该系统可以实现获取实时的疫情数据，并且展示数据和播报数据，采集的数据也能通过物联网准确传输。

参考文献

1. 何经伟,刘玉媛.基于ESP32的办公环境信息监测系统[J].电子技术与软件工程,2019(24):153-154.
2. 李梁京,张雪芹,刘华波.基于USART-HMI智能串口屏的节能恒温控制系统设计[J].制造业自动化,2021,43(09):140-143.
3. 常国权,彭云峰.基于STC15单片机和SYN6288的机床语音报警系统设计[J].电子器件,2015,38(03):616-620.
4. 宋岸峰,曾新顺,宁占彬.基于JSON的数据采集通信协议分析与实现[J].电脑编程技巧与维护,2022(03):75-79.DOI:10.16184/j.cnki.comprg.2022.03.054.
5. 杨涛. 基于Node.js的高并发电商购物系统设计与实现[D].北京邮电大学,2021.DOI:10.26969/d.cnki.gbydu.2021.002805.
6. 李草. 基于Node.js+Express框架的智能燃气监管系统应用研究[D].中国石油大学(北京),2018.DOI:10.27643/d.cnki.gsybu.2018.001168.
7. 陈晓阳.基于Java反射自动生成Postman接口测试文件的方法[J].电脑编程技巧与维护,2021(09):27-28.DOI:10.16184/j.cnki.comprg.2021.09.011.
8. 綦慧,徐晓慧.基于Web系统的大数据搜索技术的实现和优化[J].计算技术与自动化,2021,40(01):155-163.DOI:10.16339/j.cnki.jsjsyzdh.202101030.
9. 李晓薇.vue.js前端应用技术分析[J].网络安全技术与应用,2022(04):44-45.
10. 曾晓钰,唐莹,温丰蔚,罗斌,韦通明.一种基于ElementUI的表格查询组件开发方案[J].现代工业经济和信息化,2021,11(12):50-51+56.DOI:10.16525/j.cnki.14-1362/n.2021.12.018.
11. 彭曙光,王梦梅,赵麒博,申刘宝,彭玉杰,陆雪艳.面向ECharts的疫情信息可视化系统[J].福建电脑,2022,38(04):80-83.DOI:10.16707/j.cnki.fjpc.2022.04.021.

# 致 谢

首先，将我最诚挚的谢意献给我的导师朱静。他渊博的知识，深厚的学术功底和独到的见解给了我莫大的帮助。他严谨的学风、孜孜不倦的钻研和勤奋进取的精神令我终生难忘。感谢朱老师为我创造了良好的学习和科研环境，感谢老师在我多年的求学生涯中指导我如何为人、处事、做学问。

在过去两年的生活、学习和科研实践中，老师、老师给予了我很多关怀和帮助，从他们身上，我学到了很多宝贵的品质和精神。他们不仅指导我们的学习，也关心我们的生活，教我们为人处事，使我在理论上和实践上都得到了很多锻炼的机会。在整个论文的写作期间，从选题，设计到论文的最终成稿阶段都渗透着他们巨大的心血，在他们的帮助下，我才能顺利地完成论文的撰写。

衷心感谢我实验室的同学们。感谢他们在生活中对我的帮助，感谢他们在科研实践中的合作。特别是杨海涌同学给了我巨大的支持和帮助，在此表示诚挚的谢意。

衷心感谢在百忙之中评阅论文和参加答辩的各位专家、老师、教授!

作者简介：

姓 名：陈宇航 性别：男

出生年月：2000.7.28 民族：汉族

E-mail: 1910138470@qq.com

# 声 明

本论文的工作是2020年12月至2021年6月在成都信息工程大学电子工程学院完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名：

2021年5月26日

附录一 服务器简单构建

*//创建一个简单服务器*

*const http = require(“http”);*

*const server = http.createServer((request, response) => {*

*response.write(‘123’)*

*response.end()*

*})*

*server.listen(‘3000’, ‘0.0.0.0’, () => {*

*console.log(‘服务启动成功， 3000端口正在监听’)*

*})*

附录二 express框架

*//express框架解决路径和请求方式问题*

*const express = require(‘express’);*

*const app = express();*

*app.post(‘/login’, (request, response) => {*

*console.log(‘login中间件’)；*

*response.end(‘login post’)*

*})*

*App.listen(3000, () => {*

*console.log(‘3000端口正在监听’)*

*})*

附录三 文件处理

*//原生node处理文件*

*const server = http.createServer((request, response) => {*

*if (request.url === '/upload' && request.method === 'POST') {*

*request.setEncoding('binary') //设置图片文件为二进制数据*

*let body = ''*

*const boundary = request.headers["content-type"].split(';')[1].replace(' boundary=', '')*

*request.on('data', (data) => {*

*body += data //将所有的数据用字符串拼接起来*

*})*

*request.on('end', () => {*

*//为了拿到Content-Type的值先拿到这个值的位置+这个值的长度就找到截取开始的index*

*let payload = qs.parse(body, "\r\n", ": ")*

*const type = payload["Content-Type"];*

*let datafinal = body.substring(body.indexOf(type) + type.length)*

*//将多余的空格去掉*

*datafinal = datafinal.replace(/^\s\s\*/, '')*

*//将从0到boundary之前的字符串截取*

*datafinal = datafinal.substring(0, datafinal.indexOf(`--${boundary}--`))*

*console.log(datafinal);*

*//数据类型设置为二进制*

*fs.writeFile('./foo.png', datafinal, {encoding: 'binary'}, (err) => {*

*console.log(err);*

*})*

*console.log('文件上传成功');*

*response.end('文件上传成功');*

*})*

*}*

*})*

附录四 获取接种数据

*//获取接种数据的代码*

*onMounted(async() => {*

*let res: any = await getVaccines();*

*console.log('这里是Vaccines组件挂载完成后的打印', res);*

*console.log('cccc',res.data.data.VaccineTopData['中国'].total\_vaccinations);*

*datas.china\_total\_vaccinations = res.data.data.VaccineTopData['中国'].total\_vaccinations*

*datas.china\_total\_vaccinations\_per\_hundred = res.data.data.VaccineTopData['中国'].total\_vaccinations\_per\_hundred*

*datas.china\_new\_vaccinations = res.data.data.VaccineTopData['中国'].new\_vaccinations*

*datas.global\_total\_vaccinations = res.data.data.VaccineTopData['全球'].total\_vaccinations*

*datas.global\_total\_vaccinations\_per\_hundred = res.data.data.VaccineTopData['全球'].total\_vaccinations\_per\_hundred*

*datas.global\_new\_vaccinations = res.data.data.VaccineTopData['全球'].new\_vaccinations*

*keyValue.value = keyValue.value + 1;*

*console.log('456');*

*})*

*// 发送接种数据的网络请求*

*async function getVaccines() {*

*const res = await axios({*

*url: 'https://api.inews.qq.com/newsqa/v1/automation/modules/list?modules=VaccineTopData',*

*method: 'get'*

*})*

*console.log('疫苗接种情况如下', res);*

*return res;*

*}*

附录五 路由相关

*//router 路由映射关系*

*const routes = [*

*{ path: '/test1', component: () => import('../views/vaccines.vue')}, //对应疫苗接种页面*

*{ path: '/test2', component: () => import('../views/dataViewable.vue')}, //对应可视化页面*

*{ path: '/test3', component: () => import('../views/foreignSituation.vue')}, //对应国外疫情页面*

*{ path: '/test5', component: () => import('../views/provinces.vue')}, //对应34省份页面*

*{ path: '/home', component: () => import('../views/Home.vue')}, //对应疫情日历页面*

*{ path: '/urbans', component: () => import('../views/Urbans.vue') //对应某省各市区页面*

*} ]*

*//router的push方法*

*const viewMore = (payload:any) => {*

*console.log('payload', payload);*

*router.push({*

*path: '/urbans',*

*query: {*

*array: payload*

*}*

*})*

*}*