

**基于ESP 8266的**

**列车信息查询-时钟-天气预报系统**

院 系 人工智能与自动化学院

专业班级 自动化校际交流1802班

姓 名 罗 荣 胜

学 号 U201812856

2021年4月26日

**目录**

[1.个人任务说明 1](#_Toc70172701)

[2.列车信息查询系统概述 1](#_Toc70172702)

[2.1 设计背景 1](#_Toc70172703)

[2.2 系统模块图 2](#_Toc70172704)

[2.3 系统流程图 2](#_Toc70172705)

[2.4 元器件清单 3](#_Toc70172706)

[3.列车信息查询系统功能展示 3](#_Toc70172707)

[4.列车信息查询系统硬件设计 4](#_Toc70172708)

[5.列车信息查询系统软件设计 5](#_Toc70172709)

[5.1 开发环境 5](#_Toc70172710)

[5.2 主程序设计 5](#_Toc70172711)

[5.3 WiFi 标志显示的子程序 6](#_Toc70172712)

[5.4访问服务器，获取列车信息的子程序设计 6](#_Toc70172713)

[5.5 列车信息显示的子程序设计 7](#_Toc70172714)

[5.6 列车信息更新的子程序设计 8](#_Toc70172715)

[5.7 HTML网页页面显示和数据发送的子程序设计 8](#_Toc70172716)

[5.8 显示用户输入的列车信息的子程序设计 9](#_Toc70172717)

[5.9 URL解码的子程序设计 10](#_Toc70172718)

[6.调试过程及遇到的问题 10](#_Toc70172719)

[6.1 OLED屏幕驱动的调试 10](#_Toc70172720)

[6.2 图片显示的调试 11](#_Toc70172721)

[6.3 接受网页数据的调试 11](#_Toc70172722)

[6.4汉字解码的调试 12](#_Toc70172723)

[6.5 遇到的问题 12](#_Toc70172724)

[7.总结与收获 14](#_Toc70172725)

[7.1课设历程回顾： 14](#_Toc70172726)

[7.2软件和硬件设计知识收获巨大 14](#_Toc70172727)

[7.3团队合作的经历： 15](#_Toc70172728)

[参考文章及博客 16](#_Toc70172729)

[附录：列车信息查询系统部分关键代码 17](#_Toc70172730)

# 1.个人任务说明

本次电子技术课程设计我们小组三人选定的题目是“列车信息查询-时钟-天气预报系统”。我们将任务分成了三部分，其中我的任务是设计并完成列车信息查询系统，将获取到的信息显示在OLED屏幕上面(128×64)。这篇个人报告我也将就这一部分展开谈谈我的调试过程和收获。

# 2.列车信息查询系统概述

## 2.1 设计背景

随着社会科技的迅速发展，火车，动车使我们的出行更加方便，但是铁路营运市场的管理和规范问题一直是一个难题，虽然政府一直关注但是依旧未能完全解决处理，因此铁路运营成为了政府管理中的一个难点。火车票的售票业务是铁路营运中的一个最基本的业务，从表面上来看，它只是铁路营运中的一个很普通的部分，但是它在铁路管理和用户服务上都有着一定的影响，消费者在购买车票之前需要了解自己出行的时候有哪些车次是空闲的，列车出发的时间和到达的时间是否方便自己换乘，乘坐列车的当天天气如何，所以，以前那种传统的车票查询方式——需要到售票点才能查询到车票信息，已经不能够满足现代铁路运营业务流量暴涨现象的客观需求,这时候就需要一种全新的车票查询方式一网上查询，因此这种网上查询车票的方式顺应着社会的需求而崛起，这种方式可以缓解人流量高峰时段的压力，也能为消费者提供便利。  
 中国现在的铁路营运系统已经非常先进了,售票制度一直处在逐步改进的阶段。但是一到高峰期的时候，总是会出现订票难的现象。例如在春节期间，买不到票的现象一直出现，新闻中也是一直播报，因此订票制度的改革还是非常需要的,虽然电话订票一些方式的产生能够帮助广大用户更好的了解到火车票的信息，但是依旧是有漏洞的，现在是网络的时代，我们需要符合社会的发展而实现火车票的网上信息查询，这无疑方便了用户，对于铁路售票的管理也有很大的帮助。

## 2.2 系统模块图



图2-1 列车信息查询系统模块图

## 2.3 系统流程图



图2-2 列车信息查询系统流程图

## 2.4 元器件清单

表2-1 元器件清单

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 用途 |
| miniUSB | 1 | 给ESP芯片供电 |
| OLED显示屏 | 1 | 显示获取到的列车信息 |
| ESP 8266芯片 | 1 | 联网，访问智行火车票的服务器 |

# 3.列车信息查询系统功能展示

列车信息查询系统实物图如下图所示：

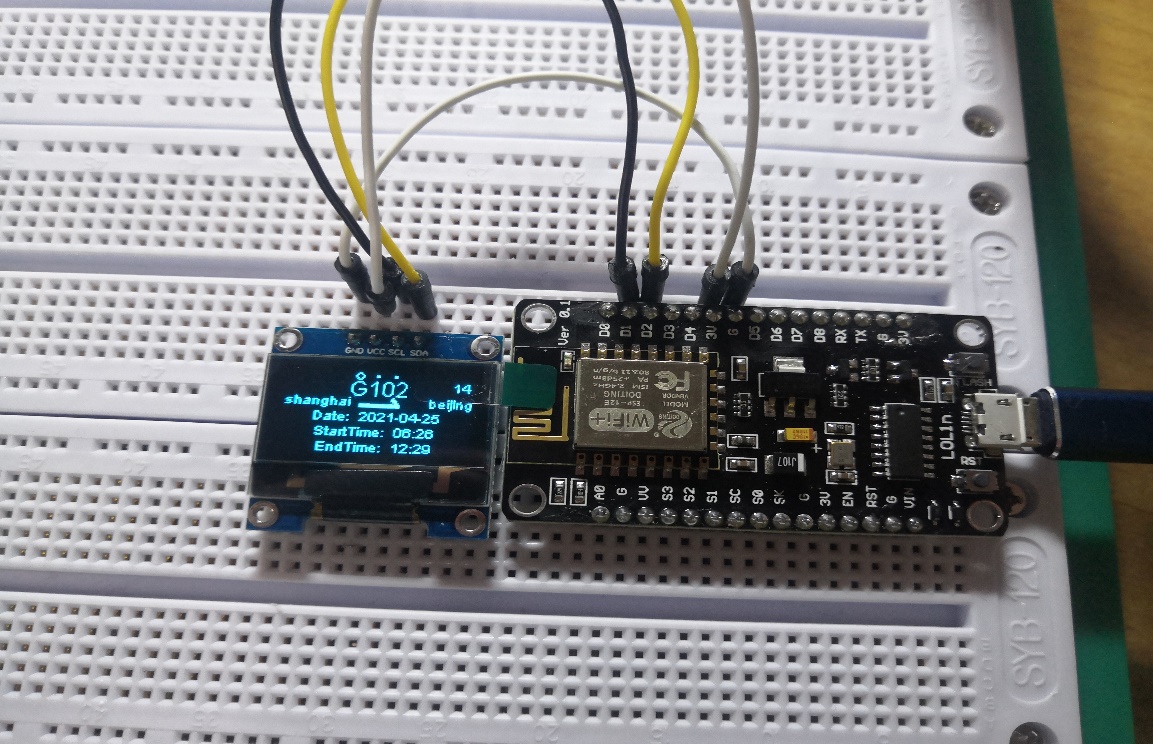


图3-1列车信息查询系统实物图

如图所示，右边是ESP芯片，用于访问智行火车票的服务器，OLED屏幕显示ESP芯片获取到的信息，ESP通过四个引脚给OLED屏幕供电的同时，使用Arduino IDE中的OLEDDisplayUI库驱动OLED屏幕的显示(引脚的连接如下表所示)：

表 3-1引脚连接表

|  |  |
| --- | --- |
| ESP 8266芯片 | OLED屏幕 |
| VCC | VCC |
| GND | GND |
| D1 | SCL |
| D2 | SDA |

OLED屏幕上显示的数据从上到下依次为列车的车次号，列车的起始站和终点站，列车出发的日期，列车出发的时间，列车到达的时间，右上角为该次列车预售票的时间。

列车信息查询系统网页如下图所示：

图3-2 列车信息查询系统网页



使用HTML语言写出可供用户输入和查询的网页接口，第一行是用户想要查询的起始站，第二行是用户想要查询的终点站，第三行是用户想要查询的出发日期。用户输入完成后点击submit按钮，ESP芯片就会根据用户输入的信息获取指定的列车信息并将更新后的信息显示在OLED屏幕上面。

# 4.列车信息查询系统硬件设计

本设计采用ESP 8266作为主控芯片，OLED屏幕显示ESP 8266芯片获取到的列车信息，ESP 8266芯片的端口连接OLED屏幕的时钟信号(SCL),给OLED屏幕提供一个时间基准，ESP 8266芯片的端口连接OLED屏幕的数据信号(SDA)，传输需要OLED屏幕显示的数据，5V给OLED提供正常运行需要的电压，OLED屏幕工作在IIC模式。

# 5.列车信息查询系统软件设计

## 5.1 开发环境

本次列车信息查询系统采用Arduino的IDE，配合Arduino自带的库和从GitHub上获取的第三方库完成整个过程的开发，开发过程中使用到的库如下表所示：

5-1 开发过程使用的库

|  |  |
| --- | --- |
| 库的名称 | 作用简述 |
| SSD1306Wire.h  OLEDDisplayUI | 驱动OLED屏幕 |
| ESP8266WiFi | 软件驱动ESP芯片连网 |
| ESPHTTPClient | ESP芯片切换为服务器模式 |
| Train  images | 自定义，用来保存访问服务器的函数和需要显示的图片的信息。 |

## 5.2 主程序设计

void setup():整个程序的入口函数，ESP 8266芯片上电后即进入setup的程序内部，完成对OELD屏幕的初始化，该函数内部的代码只会执行一次。

void loop():void setup()函数执行完毕后就会执行void loop()函数，他相当于一个while(1)的循环，不断监听是否到达信息更新的时间或者是否有用户输入要查询的信息，该函数内部的代码会不断的循环。

## 5.3 WiFi 标志显示的子程序

在ESP 8266上电并完成初始化之后，会显示WiFi图标(如下图所示)，在显示WiFi图标的过程中，ESP会不断尝试连接指定的网络，直到连接成功。

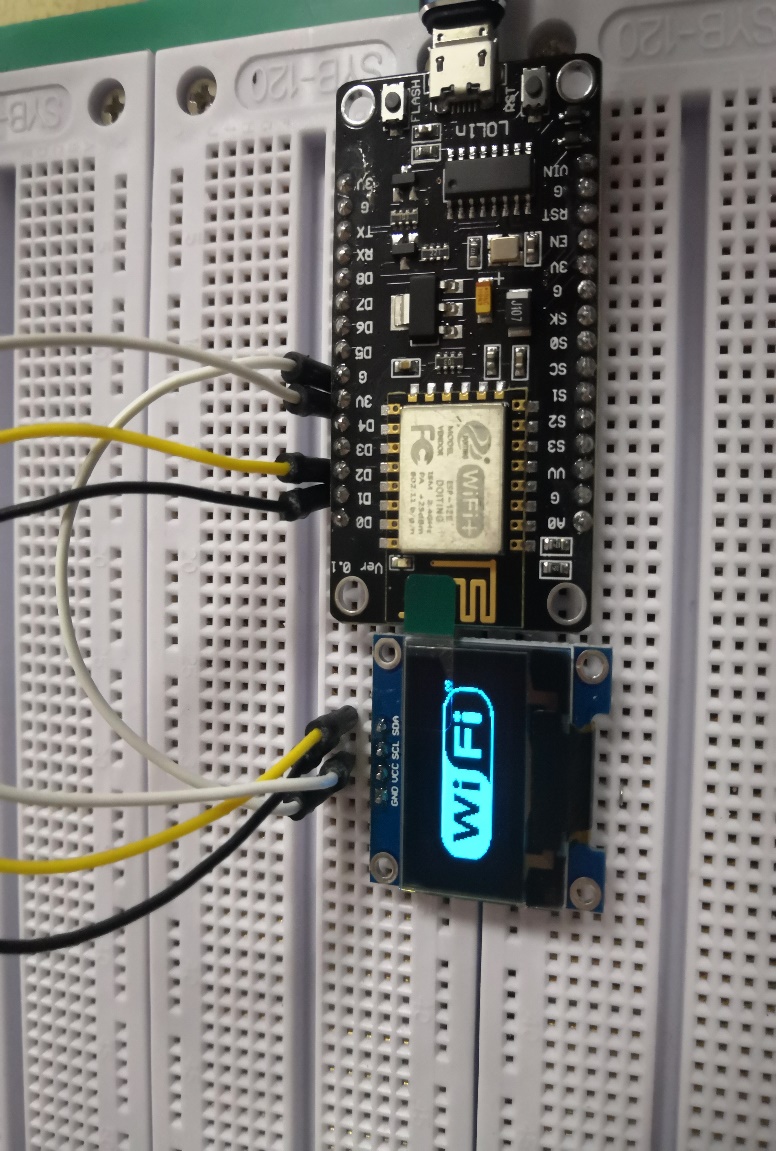


图5-1 WiFi图标

## 5.4访问服务器，获取列车信息的子程序设计

在Train.h中定义了一个Train类，它具有以下属性：TrainName(用来保存从服务器获取到的列车号)，DepartureSearch(用来保存从服务器获取到的出发站)，ArrivalSearch(用来保存从服务器获取到的终点站)，StartTime(用来保存从服务器获取到的列车出发的时间)，EndTime(用来保存从服务器获取到的列车达到的时间),Time(用来保存从服务器获取到的列车出发的日期),PreSaleDay(用来保存从服务器获取到的提前开始售票的时间)。

Train类有config()方法:用来把输入的信息重构成能用来访问服务器的字符串。

Train类有request()方法:用来请求智行火车票的服务器并获得请求数据，步骤如下所示：

第一步：定义需要访问的服务器的URL，用带有参数的POST请求来访问服务器，得到服务器的返回状态和数据。

String url = <https://m.suanya.cn/restapi/soa2/14666/json/GetBookingByStationV3ForPC>

String paramter="{\"ArriveStation\":"+train\_requestEndLocation+",\"ChannelName\":"+train\_ChannelName+",\"DepartDate\":"+train\_requestTime+",\"DepartStation\":"+train\_requestStartLocation+"}";

https.begin(\*client, url)

int httpCode=https.POST(paramter)

第二步：根据返回的值来判断是否请求成功，的值为200则表示成功获取服务器的数据，的值为403，表示服务器拒绝访问，404表示页面丢失，当的值不为200，则不断用POST请求服务器，直到数据成功获取。

while(httpCode!=200){

            httpCode=https.POST(paramter);

        }

第三步：对得到的数据进行解析。(这里仅用部分代码进行说明)

首先分配足够大的内存来存放解析后的数据：

DynamicJsonDocument doc(2048);

将整个得到的数据转换为Json对象：

JsonObject ResponseBody  = doc["ResponseBody"].as<JsonObject>();

将Json对象中的数据转换成Json数据串，方便之后的数据获取

JsonArray trainItems=ResponseBody["TrainItems"].as<JsonArray>();

将Json数据串中的数据提取出来，赋值给Train对象对应的属性

this->TrainName=trainItems[0]["TrainName"].as<String>();

## 5.5 列车信息显示的子程序设计

该系统一共获取了三趟列车的有关信息，对于每趟列车的信息，可以使用我在images.h这个头文件中定义的几个字体来显示(网上有对应的字库可以下载)，对于获取到的每个信息，首先设置显示的方式，从左往右，从右往左或者居中，然后再设置字体的大小，最后让字符串显示在给定的位置。以下以列车号的显示为例：

display->setTextAlignment(TEXT\_ALIGN\_CENTER);

display->setFont(ArialMT\_Plain\_16);

display->drawString(x+64, y+5, train[2]->TrainName);

display->setTextAlignment()用来设置字符显示的方式；

display->setFont()用来设置显示的字体的大小；

display.drawString()用来在指定位置显示字符串。

然后用Frameback函数设置三个页面的轮播：

FrameCallback frames[]= {showInfoOled0, showInfoOled1, showInfoOled2};

 int frameCount = 3;

其中showInfoOled0，showInfoOled1，showInfoOled2分别为指向需要轮播的函数入口地址的指针。

## 5.6 列车信息更新的子程序设计

代码中设置了列车信息更新的时间周期，如下图所示：

const int trainUpdateSpan = 10 \* 60;

在loop()循环中，不断使用millis()函数得到从ESP 8266芯片开机启动到当前的时间(单位为ms)，与上次更新的时间相减得到更新的时间间隔，如果时间间隔大于trainUpdateSpan，则调用更新列车信息的子程序，重新访问服务器。

if (millis() - LastTrainUpdate > (1000L \* trainUpdateSpan)) {

       LastTrainUpdate = millis();

fisrtUpdate(&display);

       readyTrainUpdate=false;

}

## 5.7 HTML网页页面显示和数据发送的子程序设计

首先将我写好的页面通过client.print()函数发送到连接在同一个局域网下的客户端，并且设置网页数据发送的格式为UTF-8

accept-charset=\"utf-8\"

设置请求信息的方式为get()

formmethod=\"get\"

这样就ESP就可以接受来子网页的信息。

## 5.8 显示用户输入的列车信息的子程序设计

当ESP 8266芯片连接上互联网之后，会通过串口发送自己的IP地址，通过Arduino IDE的串口监视器可以看到，在电脑端连接和ESP相同的网络，使得电脑和ESP芯片处在同一个局域网下面，然后在谷歌浏览器中输入串口发进来的IP地址就可以进入我写好的一个页面，用户可以根据自己的需要输入起始站，终点站和想要查询的日期，ESP会重新请求服务器，并将用户想要查询的信息显示在第三个轮播的页面。如图所示：输入起始站为杭州，终点站为武汉，出发日期为2021-04-25。



图5-2 网页输入示例

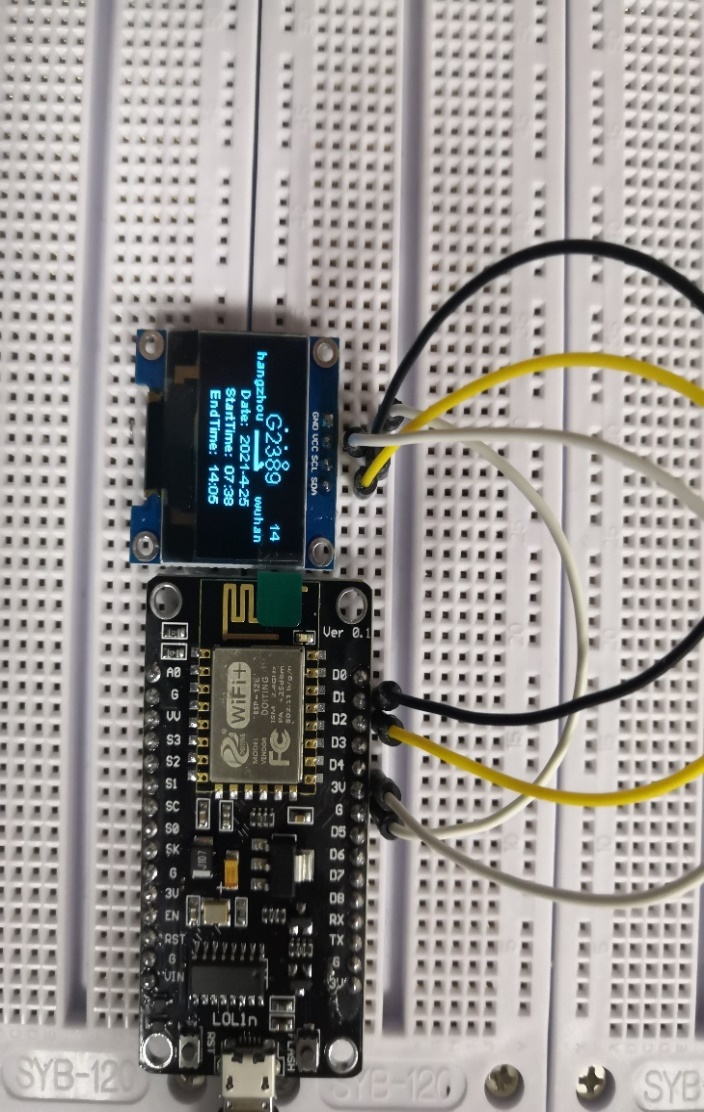


图5-3 更新后的信息显示

ESP会更新信息，然后重新访问服务器，获取用户需要的列车信息。

## 5.9 URL解码的子程序设计

由于用户输入的起始站和终点站是汉字，因此ESP 8266收到的信息是经过URL编码的，如图所示：得到的数据是经过编码的。

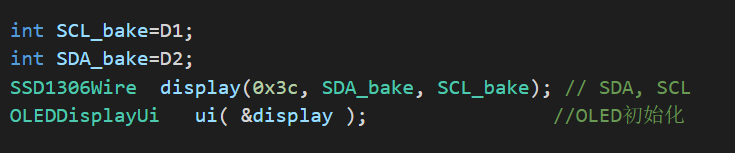


图5-4 经过编码的汉字

因此在ESP接受端需要对编码后的数据进行解码，用解码后得数据才能重新访问服务器，这里我在HTML中规定URL进行编码的方式为UTF-8，然后在程序中用函数对编码后的信息解码，得到正确的汉字信息。

# 6.调试过程及遇到的问题

## 6.1 OLED屏幕驱动的调试

一开始我购买的是7pins的OLED屏幕，可以工作在SPI或者IIC的模式，但是我通过查看<OLEDDisplayUI>头文件中对于类的定义，发现这个库只能驱动IIC工作模式下面的OELD，于是我尝试去让7pins的OLED屏幕工作在IIC的模式，最后通过CSDN查到，我使用的库函数只能驱动4pins且工作在IIC模式下面的OELD屏幕，不能驱动7pins且工作在IIC模式下面的OLED屏幕。

## 6.2 图片显示的调试

由于OELD屏幕是大小为128×64的单色点阵屏幕，因此图片不能够直接在屏幕上显示出来，需要将图片转换为BMP格式，然后用字模工具把图片二值化，最后把二值化后的矩阵显示出来才能得到对应的图片。

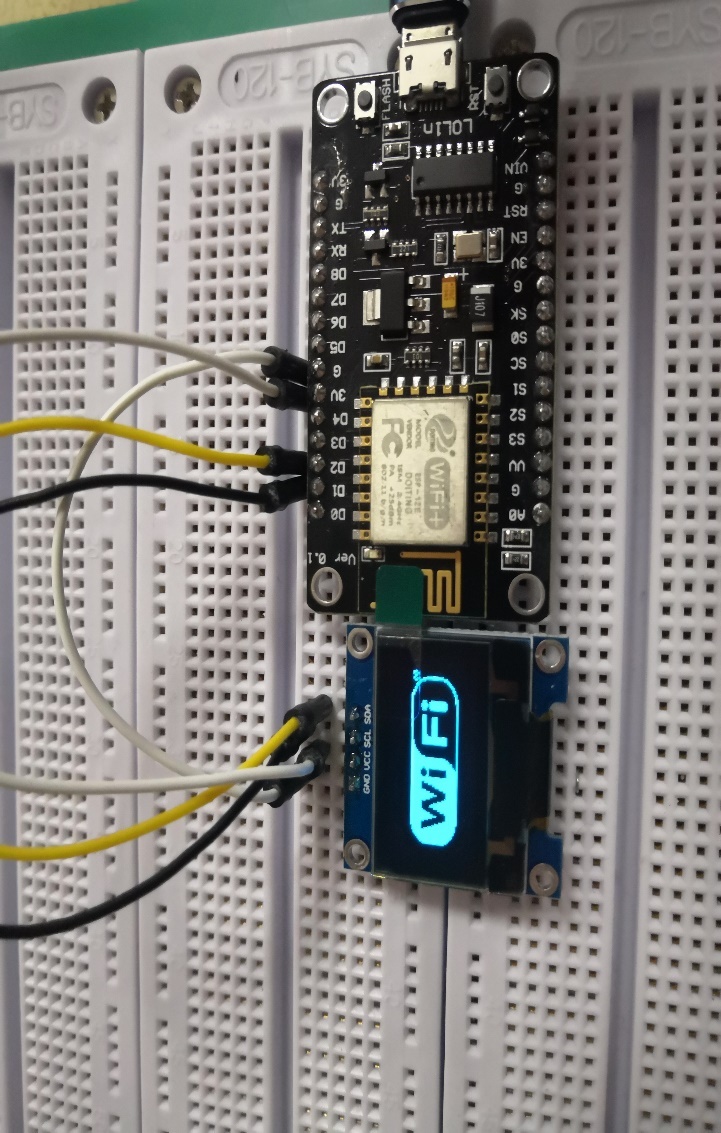


图6-1 WiFi图标的显示

## 6.3 接受网页数据的调试

最开始网页发送到ESP的经过编码的汉字使用的编码方式很难看出来，每一部分编码的长度都是不确定的，有的用2位编码，有的用3位编码，因此对汉字的解析是一项很大的挑战，为此我在网上找到了一种可以通过设置HTML中的参数从而规定URL编码的方式的属性，设置为UTF-8编码，这样每个部分都是2位编码，大大减小了解码的工作难度和工作量。

## 6.4汉字解码的调试

由于汉字是不能直接通过网页发送的，需要按照我指定的编码方式(UTF-8)对汉字进行编码，才能发送数据到ESP 8266，供其解析出汉字，对于UTF-8对汉字的编码方式，Github上面有详细的解释，所以我自己写了三个子函数，专门用于对汉字的解码。

char Dec2HexChar(short int n)；

short int HexChar2Dec(char c)；

String DecodeURL(String URL)；

最后一个函数内部会调用前两个函数对编码后的字符进行解码，返回解码后的结果。

## 6.5 遇到的问题

我在调试中遇到的第一个问题对获取到的数据进行解析，拿到我需要的数据。拿到的数据是Json格式，不能直接将其显示在OLED屏幕上，因此需要对其进行解析。而Arduino的IDE并没有提供很好的调试功能，因此我只能先写，然后烧录，再通过串口把获取状态和获取到的数据打印出来，观察是否解析成功，这种方法耗时耗力。因此我决定先用Python把Json格式先解析一遍，因为Python提供了可以直接解析Json格式为字典的包，同时还可以看到中间变量的类型，调试起来十分的方便。

通过断点调试，我发现了中途主要用了两个数据类型，Json对象和Json数据串，在Arduino提供的库函数里面有这两个数据类型

JsonObject ResponseBody  = doc["ResponseBody"].as<JsonObject>();

JsonArray trainItems=ResponseBody["TrainItems"].as<JsonArray>();

这样以来，就方便我通过RseponseBody和trainItems来获取我想要的信息。

第二个遇到的问题是如何让ESP 8266芯片能够实时检测我是否从网页发送了数据，最开始我是把发送网页内容和检测是否有数据发送过来的功能放在void setup()函数中，但是这个函数在整个代码烧录成功之后只会执行一次，也就是说，他只会在初始化的时候发送网页的信息并且检测是否有数据传送过来，在整个界面轮播的过程中不会实时检测，即使有数据发送过来也难以检测到。所以这个方法被我否定了。然后我尝试把这个检测的功能放在void loop()函数中，这样每一次循环都能够检测数据的发送了，但是我把这个函数加进去之后，整个轮播的页面会卡在某一页，最常出现的就是卡在最后一页就不再继续执行轮播的功能了，我通过百度和CSDN了解到，原因是轮播的时候是有一定的时间间隔的，并且整个轮播的时间是固定的，在中途中有可能会出现更新而重新访问服务器，这就使得整个循环的延迟大大增大，但是把轮播时间增大会使得每次运行循环的时候延迟都会增大，同时造成很大的浪费，因此直接把检测的函数放在loop()函数中虽然可行，但是时间代价很高。我的解决方法是在loop()循环中加一个判断，观察当前循环结束的时候，是否有空余的时间让我执行检测是否有数据发送的操作：

   int remainingTimeBudget = ui.update();

  if (remainingTimeBudget > 0) {

    pushControlPanel();

    delay(remainingTimeBudget);

  }

这样以来，虽然有时候会因为时间到了而不进行数据监测的功能，但是每次循环完成的时间都是微妙，毫秒级别的，因此下次循环到来时再检测数据也不会带来延迟，经过测试，这样可以保证在不增大时间开销的基础上实时监测数据的发送。

第三个遇到的问题就是汉字的解码，最开始我以为可以直接把获取到的汉字作为值传入访问服务器需要的参数，发现OELD屏幕上显示的内容要么是乱码，要么是null，于是我通过串口把获取到的数据打印出来，发现汉字是被网页进行过URL编码的。然后我想在网上找到对应的汉字库，通过计算偏移量找到对应的汉字的位置，相当于从编码后的汉字到汉字之间有一个映射，但网上的汉字库是GBK编码，不是uft-8编码，如果指定网页传数据的编码方式是GBK编码，每一部分的长度又会不一样，这又增大了解码的难度，所以通过汉字库解码的方式被我否定了。于是我试图自己写函数对UTF-8编码的汉字进行解码，我在CSDN和Giuhub上找到了UTF-8编码的原理，并且有对应的实现例子，仿照Github上的方法，我实现了对汉字的解码。

# 7.总结与收获

本次电子课程设计中，在闵玉堂老师的指导，我们小组三人的共同努力下，完成了列车信息查询-时钟-天气预报系统，通过C/S模式，实现了ESP作为客户端获取智行火车票服务器数据的功能，然后ESP作为服务器端，我通过连接到和ESP芯片同一个网络下，作为客户端和芯片进行数据的交互，让他满足我查询的需求，同时ESP中我还设定了重新向服务器发送数据更新请求的功能，每间隔十分钟，ESP会重新向服务器请求信息，以达到实时的目的。

## 7.1课设历程回顾：

回顾整个制作历程，我们历经了坎坷。在开始课设之前，我们在硬件设计和软件设计方面经验匮乏，但我们依然充满了期待。经过小组三人的商量，我们选定了这个题目，我们也想到了这个系统在实际生活中有一定的作用，我们可以查询列车有关的信息，然后可以根据天气预报看出发的那天天气情况如何，如果天气不好可以选择一个天气晴朗的出发日期，实时的天气预报可以方便我们做好规划，如果发车的时候有可能会下雨，则我们可以选择提前一点出发，这样就可以避免天气对出行造成的不便，如果在出发之前有重要的事情害怕忘记，可以设置一个倒计时来提醒自己。我们在制作的时候把时钟天气和列车功能分开，可以单独把时钟和天气功能封装起来，单独供电后可以给喜欢晨练的人使用，可以查看实时的天气，湿度，体感温度，空气质量等。

## 7.2软件和硬件设计知识收获巨大

整个课设，对我的硬件电路设计和软件系统设计都有巨大的提升！最重要的是软件驱动硬件电路和交互设计方面，之前我只有数电模电和微机原理等理论知识，以及电子线路实验做的简单电路，硬件设计方面经验严重匮乏。但通过这次课设，我学会了使用ESP 8266和OLED屏幕的使用，以及如何与ESP之间通过串口或者网络进行交互。在软件设计方面，我也是收获颇丰，如何使得ESP实时检测发送过来的数据同时又保证不占用过多的时间资源，对于一个陌生的对象，比如汉字的解码，如何用一个工科生的思维去解决问题，即“发现问题，解决问题”的能力，如何在void setup()和void loop()两个函数中把整个工程的逻辑写清楚等等，在编写这些程序的时候，不仅仅是对自己编程能力的进一步提升，更使自己在整体的系统设计上收益颇丰。

## 7.3团队合作的经历：

这是一次难忘的团队合作经历。虽然这对于我们来说是一个全新的东西，以前从来没有接触过类似于这样的课程设计，但是面对这次挑战，我们小组没有一人退缩。无论是最开始的设计方案时，互相的头脑风暴；还是中途陷入困境时，三人通力合作才柳暗花明；还是后期，三人明确的分工合作，都给我留下了深刻的影响，让我更加意识到了团队合作的重要性，并提升了合作的能力。

三个人的能力偏向不同，有的擅长编程，对新的东西有很快的学习和接受能力，知识面比较广，使得对Arduino的编程和OELD屏幕的使用学习的比较快，HTML语言编写网页上手很快，有的比较擅长做网页和ESP之间的交互，对网页返回的请求能够快速解析出想要的信息，有的擅长对获取到的信息在OELD屏幕上进行排版，使得看起来美观，总之，这是一次历经坎坷却令人难忘的而又受益匪浅的课程设计，让我全面的提升了软件和硬件的设计和制作能力，同时也明白了团队之间相互学习，通力合作的重要性。

参考文章及博客

[1] https://www.arduino.cn/thread-1066-1-1.html

[2] https://blog.csdn.net/private\_void\_main/article/details/78947831

[3] https://blog.csdn.net/qq\_41188880/article/details/109091037

[4] https://blog.csdn.net/dpjcn1990/category\_9291479.html

[5] https://blog.csdn.net/lee198717/article/details/102147775

[6] https://blog.csdn.net/a97jh98908/article/details/102054224

[7] https://blog.csdn.net/qq\_33894122/article/details/83896747

[8] https://blog.csdn.net/jiangqin115/article/details/44751663

[9] https://blog.csdn.net/qq\_41168902/article/details/103851144

[10] https://blog.csdn.net/kim5659/article/details/114445882

# 附录：列车信息查询系统部分关键代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*头文件定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "SSD1306Wire.h"        // OLED屏幕驱动

#include "OLEDDisplayUi.h"

#include <time.h>               //时间库

#include <sys/time.h>

#include <coredecls.h>

#include <ESP8266WiFi.h> //wifi联网

#include<stdio.h>

#include<stdlib.h>

#include <ESPHTTPClient.h>

#include<WString.h>

#include"Train.h"

#include"images.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*宏定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

#define STATION\_HEIGHT  18

#define DATE\_HEIGHT     30

#define STARTTIME\_HEIGHT  42

#define ENDTIME\_HEIGHT  54

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ESP与OLED连接的引脚定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int SCL\_bake=D1;

int SDA\_bake=D2;

SSD1306Wire  display(0x3c, SDA\_bake, SCL\_bake);

OLEDDisplayUi   ui( &display );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*全局变量定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool fisrtUpdateFlag=true;

bool readyTrainUpdate = false;            // 实况列车信息更新

long LastTrainUpdate = 0;                 // 上次列车更新时间

const int trainUpdateSpan = 10 \* 60;      // 实况天气更新时间间隔

WiFiServer server(80);

String send\_info("");

String infoStartLocation("");    //用于接收读取到的出发站

String infoEndLocation("");      //用于接收读取到的终点站

String departureTime("");       //用于接受出发时间

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*函数声明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void wificonnect();

void showInfoOled0(OLEDDisplay \*display,OLEDDisplayUiState\* state, int16\_t x, int16\_t y);

void showInfoOled1(OLEDDisplay \*display,OLEDDisplayUiState\* state, int16\_t x, int16\_t y);

void showInfoOled2(OLEDDisplay \*display,OLEDDisplayUiState\* state, int16\_t x, int16\_t y);

void showInfoSerial();

void fisrtUpdate(OLEDDisplay \*display);

void pushControlPanel();

char Dec2HexChar(short int n);

short int HexChar2Dec(char c);

String DecodeURL(String URL);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:void wificonnect();

函数作用:ESP连接到指定的网络

说明:当ESP当前尝试未连接成功时，会不断地请求连接，直到连接成功才会跳转页面

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void wificonnect() {

  WiFi.mode(WIFI\_STA); //设置为接收WiFi模式

  WiFi.begin(WIFI\_SSID, WIFI\_PWD);

  display.drawXbm(0, 3, 128, 64, WiFi\_Logo);  // 显示WiFi图标

  display.display();

  delay(2000);

  while (WiFi.status() != WL\_CONNECTED)

  {

    Serial.println("connected failed!");

    delay(2000);

  }

  Serial.println("connected successfully!");

  server.begin();

  Serial.println(WiFi.localIP());

  delay(1000);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:void firstUpdate(OELDDisplay \*display)

函数作用:从服务器获取信息，保存在Train类的成员中

说明:外部可以通过Trian->character的方式来访问获取到的信息

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void fisrtUpdate(OLEDDisplay \*display){

     drawProgress(display,0,"Updating Train Info...");

     train[0]->config(requestStartLocation1,requestEndLocation1,requestTime1);

     train[0]->request();

     Serial.println("0 done");

     drawProgress(display,33,"Updating Train Info...");

     train[1]->config(requestStartLocation2,requestEndLocation2,requestTime2);

     train[1]->request();

     Serial.println("1 done");

     drawProgress(display,66,"Updating Train Info...");

     train[2]->config(requestStartLocation3,requestEndLocation3,requestTime3);

    // train[2]->config("深圳","襄阳","2021-4-21");

     train[2]->request();

     Serial.println("2 done");

     drawProgress(display, 100, "Updating Done...");

     delay(500);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:void showInfoOled0()

函数作用:在Oled屏幕上显示获取到的信息

说明:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void showInfoOled0(OLEDDisplay \*display,OLEDDisplayUiState\* state, int16\_t x, int16\_t y)

{

  //display->clear();

  //显示列车号

  display->setTextAlignment(TEXT\_ALIGN\_CENTER);

  display->setFont(ArialMT\_Plain\_16);   //10.16.24

  display->drawString(x+64,y+5, train[0]->TrainName);

  //显示列车起始站

  display->setTextAlignment(TEXT\_ALIGN\_RIGHT);

  display->setFont(ArialMT\_Plain\_10);

  display->drawString(x+46, y+STATION\_HEIGHT, train[0]->DepartureSearchPinYin);

  //箭头

  display->drawXbm(x+48,y+STATION\_HEIGHT+2, 32, 16, jiantou);

  //显示列车到达站

  display->setTextAlignment(TEXT\_ALIGN\_RIGHT);

  display->setFont(ArialMT\_Plain\_10);

  display->drawString(x+128, y+STATION\_HEIGHT, train[0]->ArrivalSearchPinYin);

  //显示列车离开日期

  display->setTextAlignment(TEXT\_ALIGN\_CENTER);

  display->setFont(ArialMT\_Plain\_10);

  display->drawString(x+64, y+DATE\_HEIGHT, "Date:  "+train[0]->Time);

  //显示列车离开时间

  display->setTextAlignment(TEXT\_ALIGN\_CENTER);

  display->setFont(ArialMT\_Plain\_10);

  display->drawString(x+64, y+STARTTIME\_HEIGHT, "StartTime:  "+train[0]->StartTime);

  // //显示列车到达时间

  display->setTextAlignment(TEXT\_ALIGN\_CENTER);

  display->setFont(ArialMT\_Plain\_10);

  display->drawString(x+64, y+ENDTIME\_HEIGHT, "EndTime:  "+train[0]->EndTime);

  //显示车票预售时间

  display->setTextAlignment(TEXT\_ALIGN\_CENTER);

  display->setFont(ArialMT\_Plain\_10);

  display->drawString(x+120, y+5, train[0]->PreSaleDay);

  //Serial.println(train[0]->TakeDays);

  //display->display();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:void Train::request()

函数作用:向服务器发送请求，并对获取到的数据进行解析

说明:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Train::request()

{

  std::unique\_ptr<BearSSL::WiFiClientSecure>client(new BearSSL::WiFiClientSecure);

  client->setInsecure(); // 不进行服务器身份认证

  HTTPClient https;

  String paramter="{\"ArriveStation\":"+train\_requestEndLocation+",\"ChannelName\":"+train\_ChannelName+",\"DepartDate\":"+train\_requestTime+",\"DepartStation\":"+train\_requestStartLocation+"}";

  String url = "https://m.suanya.cn/restapi/soa2/14666/json/GetBookingByStationV3ForPC";

   if (https.begin(\*client, url)) {

        //发送请求

       int httpCode=https.POST(paramter);

        while(httpCode!=200){

            httpCode=https.POST(paramter);

        }

        if (httpCode == 200) {

        //读取响应内容

        String payload = https.getString();

        DynamicJsonDocument doc(2048);

 deserializeJson(doc, payload,DeserializationOption::NestingLimit(20));

     JsonObject ResponseBody  = doc["ResponseBody"].as<JsonObject>();

     JsonArray trainItems=ResponseBody["TrainItems"].as<JsonArray>();

        //测试

        JsonObject ResponseStatus=doc["ResponseStatus"];

        const char\* status=ResponseStatus["Ack"];

        Serial.println(status);

        Serial.println(trainItems[0]["TrainName"].as<String>());

        char buff[4];

        this->TrainName=trainItems[0]["TrainName"].as<String>();

        this->DepartureSearchPinYin=ResponseBody["ArrivalSearchPinYin"].as<String>();

        this->ArrivalSearchPinYin=ResponseBody["DepartureSearchPinYin"].as<String>();

        this->StartTime=trainItems[0]["StartTime"].as<String>();

        this->EndTime=trainItems[0]["EndTime"].as<String>();

        this->PreSaleDay=trainItems[0]["PreSaleDay"].as<String>();

        }

   }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:String DecodeURL(String URL)

函数作用:对UTF-8编码后的汉字进行解码

说明:返回解码之后的汉字

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

String DecodeURL(String URL)

{

    String result = "";

    int len=URL.length();

    for ( unsigned int i=0; i<len; i++ )

    {

        char c = URL[i];

        if ( c != '%' )

        {

            result=result+String(c);

        }

        else

        {

            char c1 = URL[++i];

            char c0 = URL[++i];

            int num = 0;

            num += HexChar2Dec(c1) \* 16 + HexChar2Dec(c0);

            result=result+String(char(num));

        }

    }

    return result;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:void pushControlPanel()

函数作用:上传网页，并检测是否有数据发送

说明:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void pushControlPanel()//上传控制面板

{

     WiFiClient  client = server.available();

    if (!client)

    {

        return; //如果未连接就重新循环监测

    }

    client.setTimeout(1000); // default is 1000

    String req = client.readStringUntil('\r');     //从缓冲区读取数据，直到'\r'

    Serial.println("请求: ");

    Serial.println(req);

    if (req.indexOf(F("/gpio/0")) != -1)

    {

        Serial.println(F("发出了0"));

    }

    else if (req.indexOf(F("/gpio/1")) != -1)

    {

        Serial.println(F("发出了1"));

    }

    else if (req.indexOf(F("/send?")) != -1)

    {

        send\_info = req;

        int start =send\_info.indexOf("info\_box");

        int end=send\_info.indexOf("HTTP/1.1")-1;

        int firstOfAnd=send\_info.indexOf("&");

        int lastOfAnd=send\_info.lastIndexOf("&");

        infoStartLocation=send\_info.substring(start+9,firstOfAnd);

        infoEndLocation=send\_info.substring(firstOfAnd+10,lastOfAnd);

        departureTime=send\_info.substring(lastOfAnd+10,end);

        Serial.println("information shown");

        Serial.println(DecodeURL(infoStartLocation));

        Serial.println(DecodeURL(infoEndLocation));

        Serial.println(departureTime);

    }

    while (client.available())

    {

        client.read();

    }

    client.print(F(HTML));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称:void loop()

函数作用:控制整个工程中的函数的实现逻辑

说明:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop()

{

    if(fisrtUpdateFlag==true){

      fisrtUpdate(&display);

      fisrtUpdateFlag=false;

    }

    if (millis() - LastTrainUpdate > (1000L \* trainUpdateSpan)) {      // 实况列车信息刷新

      readyTrainUpdate = true;

      LastTrainUpdate = millis();

      Serial.println("Update");

  }

   if(infoStartLocation!=""&&infoEndLocation!=""&&departureTime!=""){

     readyTrainUpdate=true;

     Serial.println("send\_info");

     requestStartLocation1=requestStartLocation2;

     requestEndLocation1=requestEndLocation2;

     requestTime1=requestTime2;

     requestStartLocation2=requestStartLocation3;

     requestEndLocation2=requestEndLocation3;

     requestTime2=requestTime3;

     requestStartLocation3=DecodeURL(infoStartLocation);

     requestEndLocation3=DecodeURL(infoEndLocation);

     requestTime3=departureTime;

     infoStartLocation="";

     infoEndLocation="";

     departureTime="";

   }

  if (readyTrainUpdate/\* && ui.getUiState()->frameState == FIXED\*/) {       // 实况列车信息更新

      fisrtUpdate(&display);

      readyTrainUpdate=false;

  }

   int remainingTimeBudget = ui.update();

  if (remainingTimeBudget > 0) {

    pushControlPanel();

    delay(remainingTimeBudget);

  }

}