环境监测系统技术文档

1.概述

环境监测系统的中心服务器是整套系统中最关键的部分，整合了通讯、逻辑运算、存储数据、提供接口等功能。它向下接收单片机（下位机）传来的数据，存入数据库；向上根据请求把数据发给浏览器（上位机）进行展示（图1）。同时还设计了强大方便的管理系统，包括用户管理、设备管理与数据管理三大板块，可实现权限分级，不同级别的人拥有不同的操作权限。

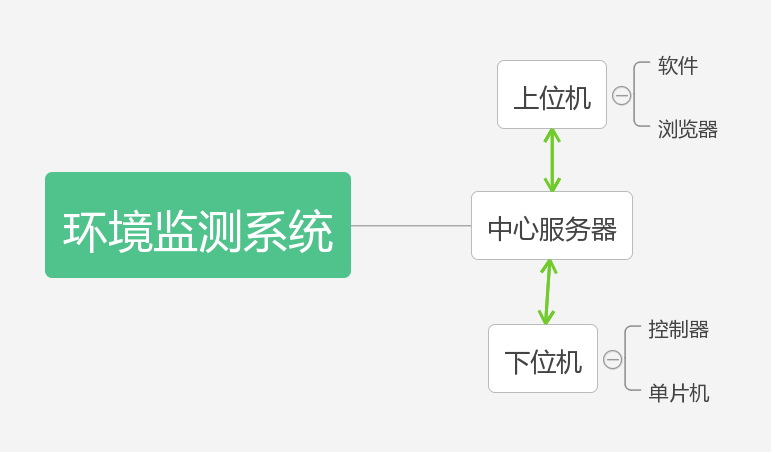


图1

2.中心服务器

中心服务器根据对接的类型由三部分组成，分别是TCP服务模块，HTTP服务模块，SMTP服务模块。（图2）

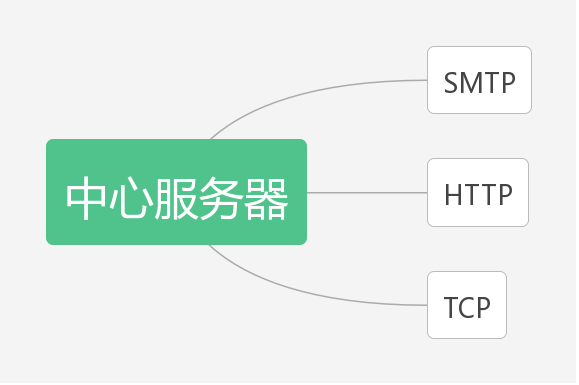


图2

2.1TCP服务模块

2.1.1服务对象

TCP服务主要用于与下位机，通过建立socket套接字链接进行通讯。用于接收单片机数据，向单片机发送指令，向控制器发送指令，以及控制器对单片机的控制枢纽。

其中单片机集成了传感器模块，并使用GPRS连接到TCP服务；控制器是在本机服务器上运行的虚拟客户端，其他进程通过调用控制器（全双工）达到向任意单片机发送指令的功能。

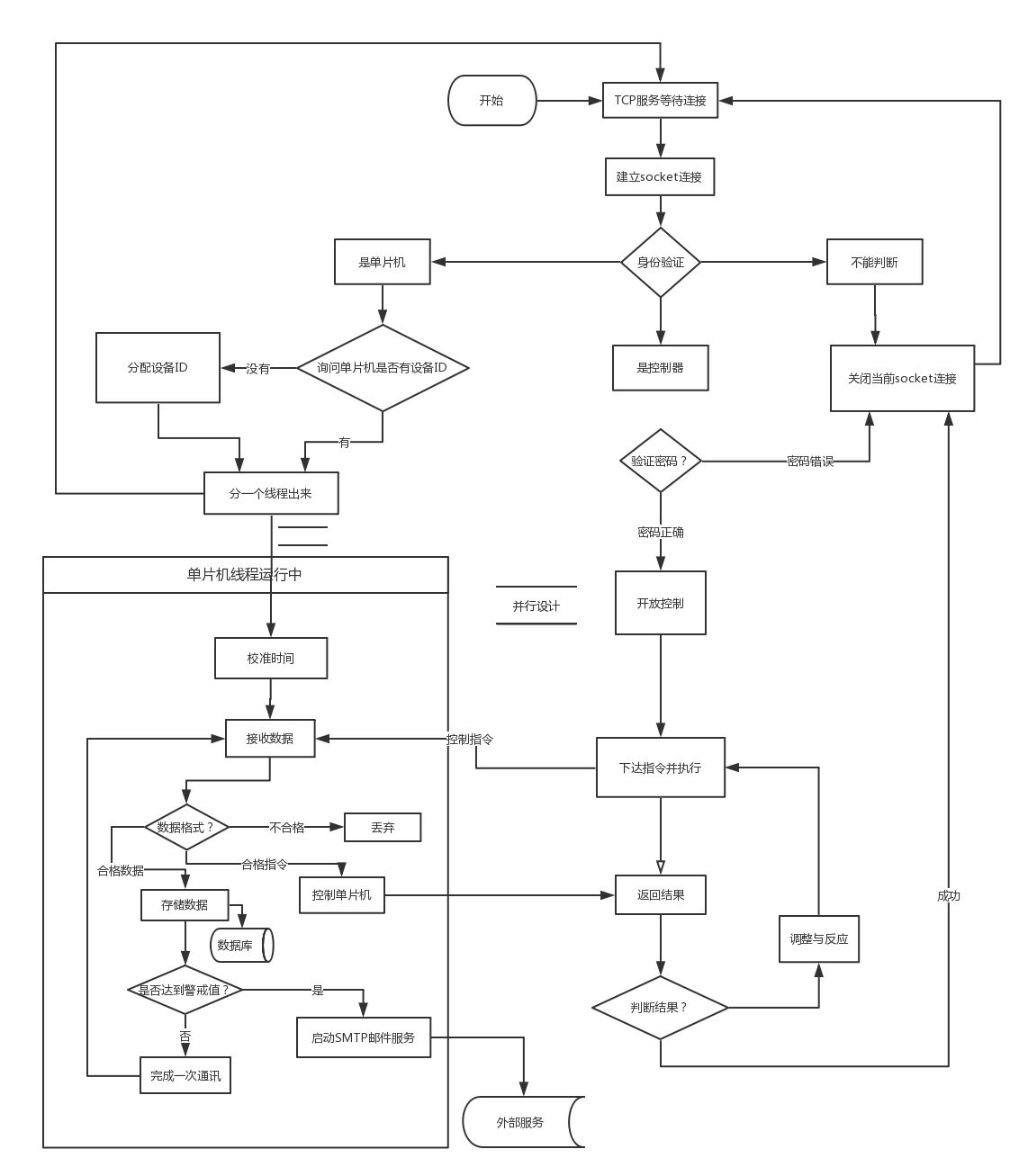
2.1.2协议

TCP协议。

2.1.3语言

使用python3写成。使用包：socket

2.1.4工作流程



2.1.5核心源码

tcpServerSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

tcpServerSocket.bind(ADDR)

tcpServerSocket.listen(5)

linking\_dev = {}

prepare = []

'''服务器端主体部分：

单片机在第一次连接后会确认身份，在此过程中出现问题会关闭连接，以免堵塞其他首次请求。

单片机可以根据返回的错误信息适当调整后再次请求连接，

身份确认通过后，服务器会发ok回去，此时才会单开一个线程供单片机使用。

在此线程中，服务器接收请求并作出回应。

控制器是用来控制单片机的客户端类型，首次连接时需确认身份。

身份确认格式为：‘con’+密码，格式正确且密码正确才会单开一个线程给控制器，

控制器可以接收到 '允许控制' 作为信号。

密码错误则直接关闭连接并在命令行内打印提醒注意。

可以操作的控制有：

指令：'ADDANID'+一设备ID 在设备添加完成后调用

要求添加设备时先在网页上添加设备，

然后新加的设备迅速连接服务器并发 DEVMYNUM 方可配对成功

指令：'STOP'+一设备ID

指令：'START'+一设备ID

再有控制单片机的需求直接重写 handle\_cont 函数即可。

'''

while True:

print('--------等待连接--------')

tcpCliSocket, addr = tcpServerSocket.accept()

print('--------与{0}建立了连接--------'.format(addr))

try:

check\_data = tcpCliSocket.recv(BUFSIZ)

except ConnectionResetError:

print('--------该连接被迫下线--------')

# 这条信息为区分单片机客户端与控制器客户端所用

if (check\_data[0] == 10) & (check\_data[1] == 13) & (data[-2] == 13)& (data[-1] == 10):

if len(check\_data) == 12:

ask\_data = (check\_data[2:10]).decode()

if ask\_data == 'DEVMYNUM':

# 新设备

if len(prepare) != 0:

the\_id = prepare[0]

if the\_id in linking\_dev:

# 应该不会出现。出现说明 prepare 列表出现了错误

print('出现了问题，请检查 prepare 不应该有已经分配出去的ID')

linking\_dev[the\_id] = tcpCliSocket

del prepare[0]

tcpCliSocket.send('ok{0}'.format(the\_id).encode())

data\_rev = threading.Thread(target=handle\_data, args=(tcpCliSocket, the\_id))

data\_rev.start()

else:

# 先去添加设备页添加设备再操作

tcpCliSocket.send('err1'.encode())

tcpCliSocket.close()

elif ask\_data[0:6] == 'DEVNUM':

# 原有设备

the\_dev\_id = int(ask\_data[6:8])

if the\_dev\_id in linking\_dev:

# 之前得连接没有正确关闭

linking\_dev[the\_dev\_id].close()

linking\_dev[the\_dev\_id] = tcpCliSocket

tcpCliSocket.send('ok{0}'.format(the\_dev\_id).encode())

data\_rev = threading.Thread(target=handle\_data, args=(tcpCliSocket, the\_dev\_id))

data\_rev.start()

else:

# 数据格式不正确

tcpCliSocket.send('err2'.encode())

tcpCliSocket.close()

else:

# 数据长度有误

tcpCliSocket.send('err3'.encode())

tcpCliSocket.close()

else:

# 有可能是控制器客户端

if check\_data[:3] == 'con':

# 就是控制器客户端

if check\_data[3:] == password:

# 密码正确，允许控制

if 'control' in linking\_dev:

linking\_dev['control'].close()

del linking\_dev['control']

tcpCliSocket.send('允许控制'.encode())

linking\_dev['control'] = tcpCliSocket

con\_thread = threading.Thread(target=handle\_cont, args=(tcpCliSocket,))

con\_thread.start()

else:

# 密码错误

print('--------有控制器试图连接，因密码错误被拒绝。请关注。--------')

print('--------地址与端口是：{0}--------'.format(tcpCliSocket.getpeername()))

tcpCliSocket.close()

else:

# 是单片机客户端，且数据开头和结尾不符规定

tcpCliSocket.send('err4'.encode())

tcpCliSocket.close()

tcpServerSocket.close()

2.2HTTP服务模块

2.2.1服务对象

这一部分是服务上位机部分。上位机主体是浏览器，即一个WEB应用，只要有账号密码，则管理人员随时随地都可以掌握数据并进行管理。同时网络部分做了非常简单的api接口，任何公司或个人可以轻松地使用我们的接口进行软件开发。我们同时也利用接口做了一款windows的桌面版程序。这里着重介绍WEB应用。

2.2.2协议

HTTP协议。

2.2.3语言

使用python3写成。使用包：django（版本：1.10）

2.2.4WEB应用介绍

整个WEB应用分为数据展示与内容管理两部分。

2.2.4.1数据展示

2.2.4.1.1服务描述

提供实时数据展示，并自动绘制成折线图。为游客权限，即任何登陆网站的个人都可以查看到实时数据。考虑到当今人们越来越关注身边环境，可以查看实时数据是大势所趋。

为提供极大方便，人们在左侧面板可以选择数据优先或地点优先。数据优先按数据类型分图，地点分线；地点优先则按地点分图，数据类型分线，横坐标一直都是时间。以此提供时间流逝比较——各项指标变化趋势与预测、同一数据的地点间横向对比——区域分块治理，以及同一地点各项指标参看——关心身边环境与各项指标间关系的研究。是本系统一大特色。

2.2.4.1.2技术手段

AJAX前后端通讯获取数据，纯前端JavaScript调整曲线。

2.2.4.2内容管理部分

内容管理部分分为四个区，分别是登录区、设备管理区、用户管理区以及数据管理区。

2.2.4.2.1登录区

只有登录的用户才有更高级的权限，未登录都是游客权限。

2.2.4.2.2设备管理区

设备管理员的权限区。可以进行添加设备、编辑设备等操作。

2.2.4.2.3用户管理区

根据登录者的等级不同，用户管理区给出的权限也不同。

在本系统中，所有用户可以随意分为若干个组别，共分为三个等级，分别是超级管理员、组管理员与普通用户。每个用户都有编辑个人信息的权限、以及在创建时被赋予的权限（包括能否管理设备，能够管理哪一项数据等）。除此之外，组管理员有管理本组组员的最高权力；超级管理员有管理所有人的一切权力。

最初的设计建议可以分一些诸如“A地管理员”、“B地管理员”、“温度专家”、“湿度专家”这样的组别，各个小组各司其职，共同维护好设备并管理好数据。

2.2.4.2.4数据管理区

数据管理主要提供了历史数据查询与警报设置两个功能模块。历史数据查询是拥有当前数据项目权限者的功能区，在这里可以查询到之前任意一天的所管辖项目的数据。这里的数据供分析、研究、预测以及治理使用，故以表格的形式呈现。

警报设置区可以设置任意项目的上下限以及负责人的联系邮箱。一旦接收到的数据不在正常范围内，便会触发邮件服务，方便了管理员。

2.2.4.3技术实现

均使用django的模型-视图-模板三层关系处理请求。

权限判断处使用了装饰器。

2.3SMTP服务模块

2.3.1介绍

该模块只供内部使用，达到数据超过警戒线时发送邮件提醒的目的。

2.3.2协议

SMTP协议。

2.3.3语言

使用python3写成。使用包：smtplib

2.3.4核心源码

def send\_mail(to\_list,sub,content):

me="17865191779"+"<"+mail\_user+"@"+mail\_postfix+">"

msg = MIMEText(content,\_subtype='plain')

msg['Subject'] = sub

msg['From'] = me

msg['To'] = ";".join(to\_list) #将收件人列表以‘；’分隔

try:

server = smtplib.SMTP()

server.connect(mail\_host) #连接服务器

server.login(mail\_user,mail\_pass) #登录操作

server.sendmail(me, to\_list, msg.as\_string())

server.close()

return True

except Exception as e:

print(str(e))

return False

3.编程风格与亮点

均使用python3写成，代码简单易读。面对对象设计，使得在功能拓展与代码维护上十分方便。很多封装出来的类都可以作为除了我们直接提供的接口以外更加底层的接口，给更多设计师与程序员自由，相信本系统经过简单的优化会变得更强大、更高效。

3.1封装出来的类接口

3.1.1写入数据库部分

每天实例化一次，添加设备后实例化一次。使用实例化后的对象实现写入与读出数据。

写入数据方法：

Data\_db.write\_in(msg, data)

msg(list) = [time, dev]

time(str) 数据时间 示例：'20161010 00:00'

dev(int) 设备id

data(list) = [tem, hum, pm25, pm10]

无返回值

读取数据方法：

Data\_db.read\_out(choose, start, end)

choose(list) = [dev'n'item\_1, dev'n'item\_2]

dev'n'item(str) 设备号与数据项目 示例：'12' 设备1的PM2.5

对应关系：0-温度 1-湿度 2-PM2.5 3-PM10

start(str) 开始时间 示例：'20161010 00:00'

end(str) 截止时间 示例：'20161010 00:00' 为空则取当前时间为默认值

return data(dic) 数据字典 键遍历 choose 中每一项，值为相应时间范围内数据 示例：

{

'00':[10.10, 10.20, 10.30, 10.40, 10.50],

'23':[1111, 2222, 3333, 'null', 5555],

}

3.1.2TCP部分

服务器端主体部分：

单片机在第一次连接后会确认身份，在此过程中出现问题会关闭连接，以免堵塞其他首次请求。

单片机可以根据返回的错误信息适当调整后再次请求连接，

身份确认通过后，服务器会发ok回去，此时才会单开一个线程供单片机使用。

在此线程中，服务器接收请求并作出回应。

控制器是用来控制单片机的客户端类型，首次连接时需确认身份。

身份确认格式为：‘con’+密码，格式正确且密码正确才会单开一个线程给控制器，

控制器可以接收到 '允许控制' 作为信号。

密码错误则直接关闭连接并在命令行内打印提醒注意。

可以操作的控制有：

指令：'ADDANID'+一设备ID 在设备添加完成后调用

要求添加设备时先在网页上添加设备，

然后新加的设备迅速连接服务器并发 DEVMYNUM 方可配对成功

指令：'STOP'+一设备ID

指令：'START'+一设备ID

再有控制单片机的需求直接重写 handle\_cont 函数即可。

3.1.3控制单片机部分

def control(ask\_sent, password):

sock = socket.socket()

sock.connect(ADDR)

sock.send(password.encode())

perm = sock.recv(BUFSIZ)

if perm.decode() == '允许控制':

sock.send(ask\_sent.encode())

stat = sock.recv(BUFSIZ)

if stat.decode() == '成功':

sock.send('exit'.encode())

sure = sock.recv(BUFSIZ)

if sure.decode() == '退出':

sock.close()

return 0

else:

sock.close()

return 1

else:

sock.close()

return 2

else:

sock.close()

return 3