

前言：

开发基于树莓派 4G 模块，可以使用 4G 模块的串口进行 AT 指令程序开发，也可以使用 4G 模块的 USB 口给树莓派实现上网功能。这里主要给用户介绍使用 python 语言的代码开发。实现使用树莓派控制 4G 模块进行功能应用。

Python 有其独特的优势。关于树莓派的相关介绍，我相信使用树莓派的用户会比我更熟悉一些，这里就不做树莓派做扩展说明。主要是给用户介绍使用 4G 的扩展板，以及使用 Python 开发的便捷性。

目录

1. 环境搭建	3
1.1 Thonny.....	3
2. 4G 扩展板硬件介绍.....	4
2.1 EC200U 原理图.....	4
2.2 电源电路.....	4
2.3 SIM 卡电路	5
2.4 耳机口电路	5
2.5 串口电路.....	6
2.6 树莓派接口电路	6
3. 软件开发	6
3.1 树莓派串口配置.....	7
3.2 EC200U.py 介绍.....	9
3.3 TCP 传输数据	11
3.4 UDP 数据发送.....	14
3.5 GPS 测试解析	15
3.6 打电话	17
3.7 发英文短信	17
3.8 发数据到阿里云	19
3.9 发数据到 ONENET 平台	21
3.9.1 注册 ONENET 平台	21
3.9.2 发数据到 ONENET 平台	24

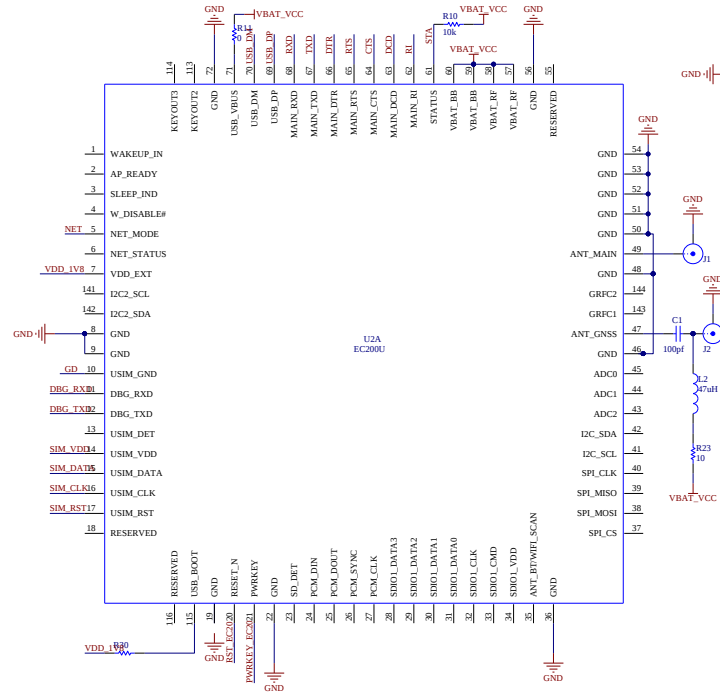
1. 环境搭建

1.1 Thonny

对于树莓派的软件开发是需要有开发环境作为搭配使用的。这里使用 Thonny , 这个主要是用来开发 Python 语句的, 非常的方便。不树莓派的系统一般都是自带了这个 Thonny 软件。并且也支持 Python。所以使用这个软件来开发非常的方便, 当然用户如果使用其他软件开发也是支持的, 只要支持 python 开发即可。

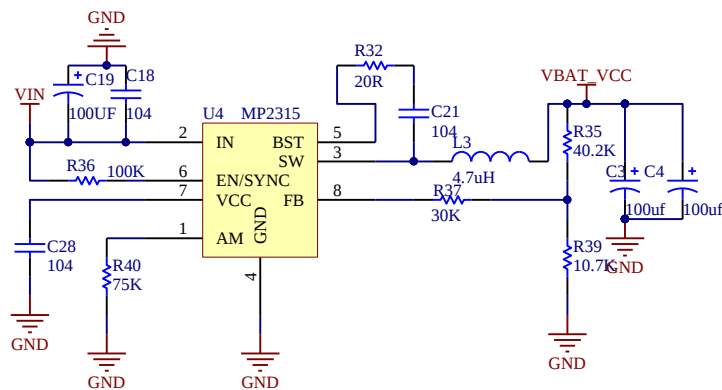
2. 4G 扩展板硬件介绍

2.1 EC200U 原理图



对于 EC200U 而言，根据硬件设计手册要求，将电源，天线，SIM 卡以及其他外设接口引出，串口重要的也要引出，因为需要通过串口来与板子进行通讯。

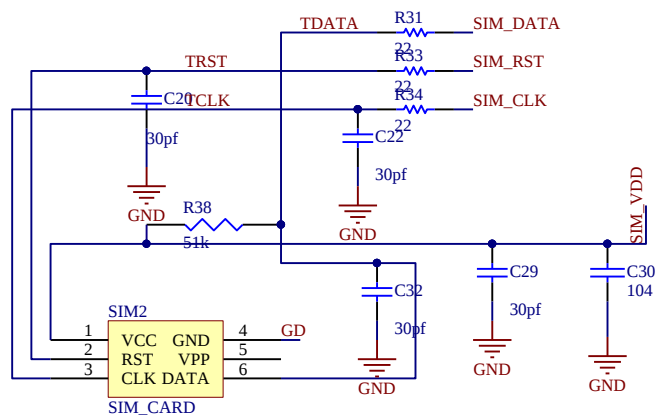
2.2 电源电路



DCDC电源

电源电路采用的是 MPS 的 MP2315 芯片，实现了 5V 降压到 3.8V，提供的电流可以保证 2A，所以用户使用起来没有问题。

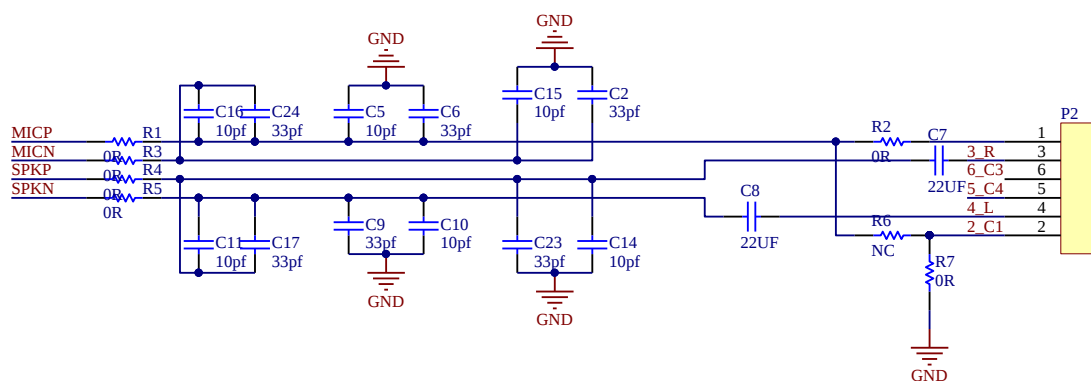
2.3 SIM 卡电路



SIM卡电路

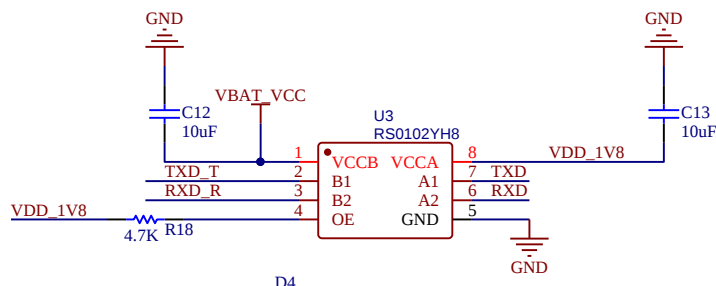
SIM 卡电路按照硬件设计要求，正常设计即可，注意走线需要包地并尽量短。

2.4 耳机口电路



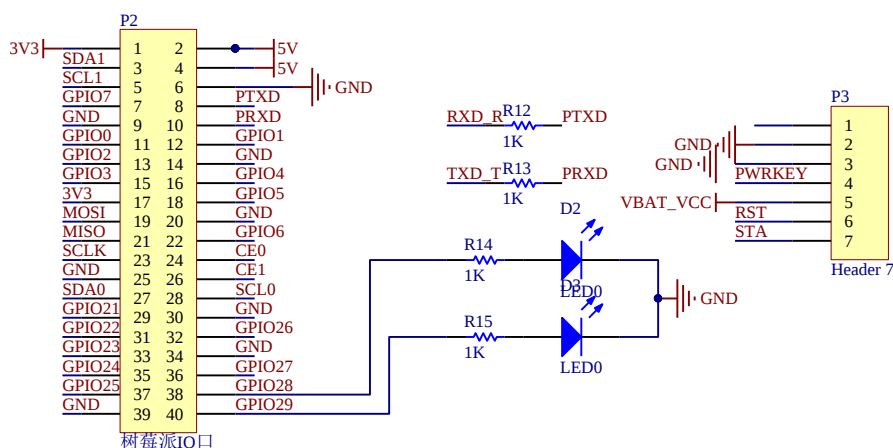
耳机口电路可以实现通过耳机进行接打电话，用户插入耳机后，可以实现通话功能。

2.5 串口电路



这个地方实现电平转换芯片，将 EC200U 的串口 1.8V 电平转换成 3.3V 给树莓派进行识别。这个串口需要接到树莓派的硬件串口脚上，对应的就是 `ttyS0`。

2.6 树莓派接口电路



树莓派提供了 40 个 IO 口，这里将端口引出并焊接引出，可以方便用户在其他需要其他外设接入的时候，还可以正常使用。

3. 软件开发

这一节将重点给用户介绍下 Python 的开发，对于使用 Python 用户需要掌握一定的 Python 语言基础，比如基本的变量定义，列表，字符串，函数，类。

Python 本身是个简单的语言，但是也极为强大，学习 Python 需要循序渐进。下面就将对 Python 的开发代码进行简单的描述。

Python 是代码写好，立即编译与下载执行，比起 C 语言运行起来简单很多。

3.1 树莓派串口配置

系统装好之后，利用 Putty 或者是其他工具登录，我们这里采用 FinalShell 来进行登录，使用 SSH 方式，需要在根目录下面创建一个 ssh 文件。

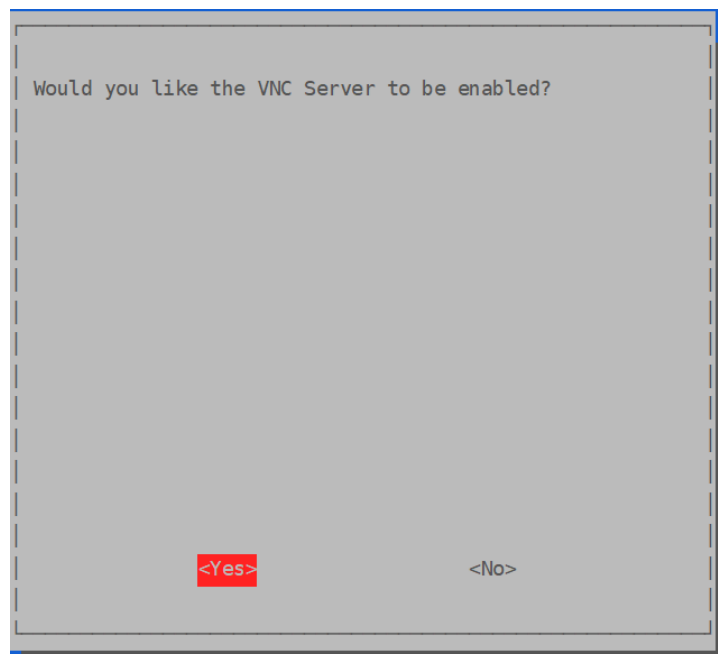
需要先配置显示器，不然没法进入到界面当中查看显示。

`sudo raspi-config`

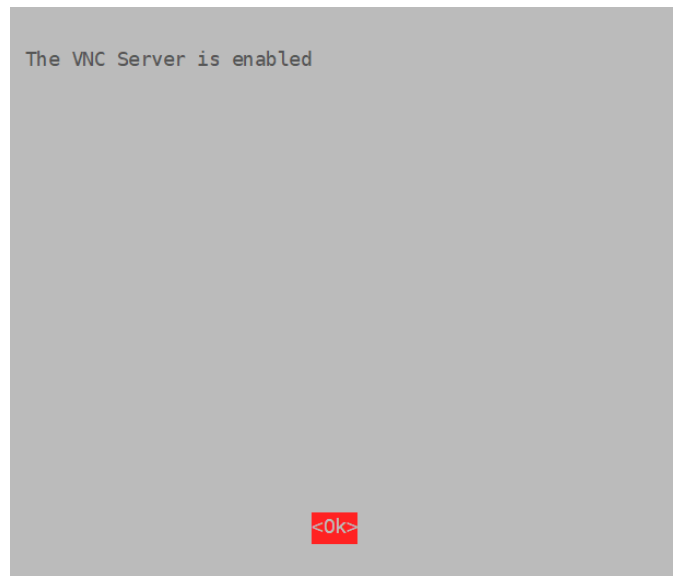
这个是配置显示屏

```
P1 Camera      Enable/disable connection to the Raspberry Pi Camera
P2 SSH         Enable/disable remote command line access using SSH
P3 VNC         Enable/disable graphical remote access using RealVNC
P4 SPI         Enable/disable automatic loading of SPI kernel module
P5 I2C         Enable/disable automatic loading of I2C kernel module
P6 Serial Port Enable/disable shell messages on the serial connection
P7 1-Wire      Enable/disable one-wire interface
P8 Remote GPIO Enable/disable remote access to GPIO pins
```

配置 VNC 开启，初次的时候，应该要装一些驱动，如果没有会提示安装的。



点击 yes 进入安装



被我们打开了



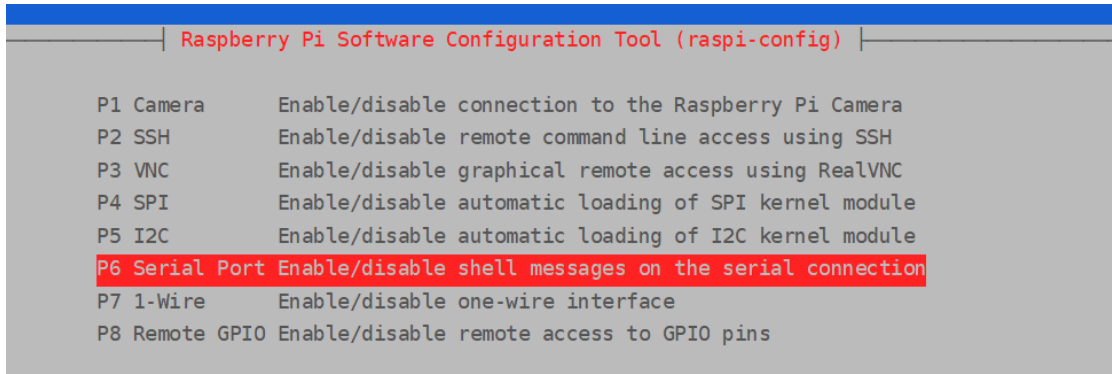
提示重启，表明 OK, 分辨率一定要设置，否则不能用

ls /dev/ttyS*

下面查看 port 端口情况，默认 ttyS0 是 minicom。就是板子上的 GPIO 的串口

```
pi@raspberrypi:~$ ls /dev/ttyS*
ls: cannot access '/dev/ttyS*': No such file or directory
```

没有串口，我们要使能串口



设置好，重启生效的，这个是个坑，否则一直不行

```
pi@raspberrypi:~$ ls -l /dev |grep ttyUSB
pi@raspberrypi:~$ ls /dev/ttyS*
/dev/ttyS0
```

查到设备端口了。

3.2 EC200U.py 介绍

为了方便用户移植程序，这里将对 EC200U 的串口 AT 操作单独写成一个 py 文件，方便用户调用。这样用户在添加其他功能的时候，可以不受影响。下面将介绍下 EC200U.py 的相关功能代码介绍。

对于 python 而言，创建类并调用类是非常必要且可行的，下面将创建 EC200U 类给用户进行使用。

```
import RPi.GPIO as GPIO
import serial
import time
```

因为需要用到串口，所以需要加入 serial 库，方便用户来做串口使用。

```
class EC200U:
    def __init__(self, port, baudrate):
        self.uart0 = serial.Serial(port, baudrate)
        if self.uart0.isOpen == True:
            self.uart0.open()
            self.uart0.flushInput()
```

初始化程序里面先对串口进行初始化，并开启，这里保留对应串口和波特率参数给用户进行调用使用。

```
def ATinit(self, ATcmd, OKcmd):#发AT指令给模块
    self.uart0.write(ATcmd.encode())
    #self.uart0.write(b'\r\n')
    time.sleep(0.3)#单位是s
    self.rxData=''
    self.rxData = self.uart0.read(self.uart0.inWaiting())
    #print(rec_buff)
    #while self.uart0.any() > 0:
        #self.rxData += self.uart0.read(1).decode()#读取里面的内容
    print(self.rxData)
    if OKcmd not in self.rxData:#返回的数据里面不在，表明读取异常
        return 0
    else:#读取正确
        return 1
```

定义了一个 ATinit 函数，这个函数主要是操作一些 AT 指令给模块，比如发 AT 确保回 OK，查询卡的状态，查询信号，注册网络状态等。这个一般用于在实际连接业务之前的判断，这个是必要的，因为需要确保模块一切准备就绪才可以进行下一步操作。主要是通过串口先发 AT 指令，然后等待 300ms，读取串口缓存里面是否有数据，如果存在数据，就与 OKcmd 指令里面进行判断内容是否相符，然后返回 0 或者 1 给调用者。

```
def ATSignle(self, ATcmd, OKcmd):#发AT指令给模块,但每次只发一次,用户连接服务器
    self.uart0.write(ATcmd.encode())
    time.sleep(2)#单位是s
    self.rxData=''
    self.rxData = self.uart0.read(self.uart0.inWaiting())#读取里面的内容
    print(self.rxData)
    if OKcmd not in self.rxData:#返回的数据里面不在，表明读取异常
        return 0
    else:#读取正确
        return 1
```

ATSignle 这个指令是主要用来做一次指令发送的，比如连接服务器指令。这个指令一般只需要发一次就可以，不能发多次，否则就会报错，这里的延时时间来到了 2S，或者也可以更长都是可以用的。

```
def TCPSend(self, ATcmd, OKcmd):#发TCP数据到服务器
    self.uart0.write(ATcmd.encode())
    time.sleep(0.5)#单位是s
    #self.rxData=''
    self.rxData = self.uart0.read(self.uart0.inWaiting())#读取里面的内容
    print(self.rxData)
    if OKcmd not in self.rxData:#返回的数据里面不在，表明读取异常
        return 0
    else:#读取正确
        return 1
```

其实这里的 TCPSend 函数和前面的函数使用没什么区别，都是 AT 的操作，这样操作的好处在于方便用户分辨指令。比如这个函数当前就是用来做 TCP 数据接收的。这样比较好容易分辨。封装成类之后方便其他应用案例进行调用。

3.3 TCP 传输数据

```

3 import RPi.GPIO as GPIO
4 import serial
5 import time
6 from threading import Timer
7 from EC200U import*

```

TCP 代码里面需要加载 EC200U 的库，所以使用 from 这个关键词来使用，因为需要使用 EC200U 的全部函数所以使用*来代替所有的类和函数被调用使用。

使用 TCP 用户需要参考 EC200U 的 TCP 相关指令，这样使用指令的时候，就不会感到陌生。

参考“Quectel_EC200x&EC600x&EG912Y 系列_TCP(IP)_应用指导_V1.3 “使用里面的指令来实现 TCP 或者 UDP 的连接。

```

14 class EC200Uctr:#发AT指令控制模块
15     def EC200U_AT(self):
16         while(ec200u.ATinit('ATE0\r\n','OK')==0):#判断模块是否存在
17             pass
18         while(ec200u.ATinit('AT+CIIM\r\n','460')==0):#判断卡是否存在
19             pass
20         while(ec200u.ATinit('AT+CGATT?\r\n','+CGATT: 1')==0):#判断是否注册网络成功
21             pass
22         ec200u.ATSignle('AT+QIAC=1\r\n','OK')
23         ec200u.ATSignle('AT+QIAC?\r\n','+QIAC:')#GET IP address
24         ec200u.ATSignle('at+qiclose=0\r\n','OK')#对上一次连接断开，不需要判断是否关闭成功
25         if ec200u.ATSignle('AT+QIOPEN=1,0,\"TCP\",'+ServerIP+'\",'+Port+',0,1\r\n','+QIOPEN: 0,0')
26             self.tcpconok=1#连接上服务器了
27         else:
28             self.tcpconok=0#fail

```

在 TCP.py 里面创建 EC200Uctr 控制类，实现 AT 指令控制模块，从上面的程序程序可以看到，首先发 AT 指令判断模块是否存在，然后发送判断 SIM 卡是否存在指令，这个是必然的，没有 SIM 卡是没法操作的。然后发送判断注册网络状态，注册上网络之后，才可以进行场景激活与 IP 获取。可以看到获取卡以及判断网络状态都是需要判断状态值是否与需要的值保持一样，如果不一样会一直在 while 循环里面进行判断，直到返回的值与实际一样，表明模块正常才会进入下一步操作。

然后进入 TCP 连接函数，在手册里面有介绍，用户根据指令的协议来操作即可。

3.2. TCP 客户端在缓存模式下工作

3.2.1. 创建 TCP 客户端连接并进入缓存模式

```
AT+QIOPEN=1,0,"TCP","220.180.239.212",8009,0,0 //场景是 1，<connectID> 为 0。在执行
AT+QIOPEN 之前，Host 需要使用 AT+QIACT
激活场景。
OK
```

上面是官方手册的说明，用户根据说明来操作即可。这里因为我们提供的函数已经可以直接使用，如果用户需要修改接入自己的服务器，只要在宏定义的地方改一下自己的服务器参数即可。

```
10 ServerIP = '47.92.146.210'
11 Port = '8888'
```

上面的 IP 地址和端口号是免费开放给用户使用的，用户可以直接使用上面的服务器来测试模块的数据收发，但是用户没法登陆服务器进行数据查看，只能测试数据收发。用户如果需要改动，就改动这个地方的定义参数即可。

```
25         if ec200u.ATSignle('AT+QIOPEN=1,0,\"TCP\",\\\"'+ServerIP+'\\',
26             self.tcpconok=1#连接上服务器了
27         else:
28             self.tcpconok=0#fail
```

上面是判断服务器是否连接成功，如果连接上，设置的变量会置为 1，否则为 0。根据这个参数来确认是否可以往服务器进行数据的发送。

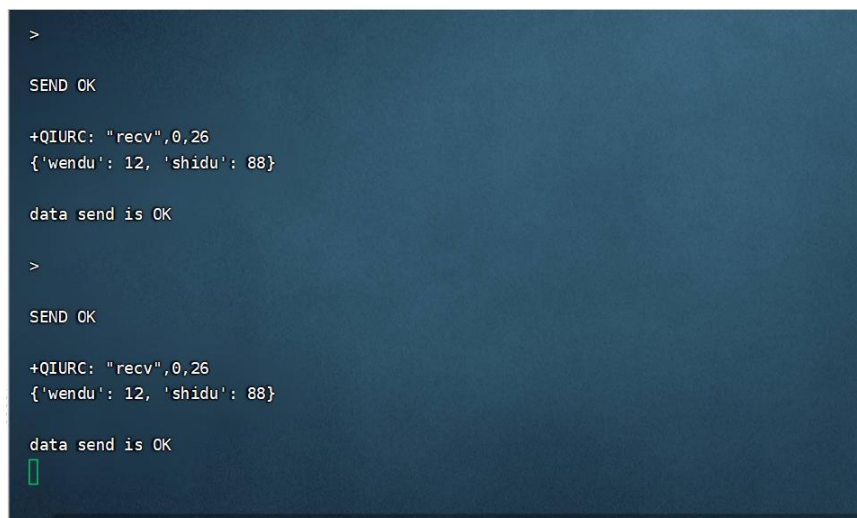
```
48 ec200u=EC200U('/dev/ttyS0',115200)
49 ec200uctr=EC200Uctr()
50 ec200uctr.EC200U_AT()
51 GPIO.setmode(GPIO.BCM)
52 GPIO.setup(20,GPIO.OUT)#LED1
53 GPIO.setup(21,GPIO.OUT)#LED2
54 wenshidu['wendu']=12#key zidian
55 wenshidu['shidu']=88#key zidian
```

上面的代码其实可以称之为初始化函数，因为 python 并不需要所谓的 main 打头的函数名称才可以为首先执行的代码，python 语言执行非常简洁，使用起来也随心所欲。

代码首先调用了 EC200U 的类并调用，配置了 ttyS0 为硬件端口，设置波特率为 115200。然后定义了一个 EC200U 控制类引用。初始化 LED 灯端口。这里模拟了 2 个参数，分别是温度和湿度值。温湿度值被存在了字典里，利用字典的 KEY 关键词来赋值，非常的方便。Python 里面的字典功能还是值得用户来使用的。

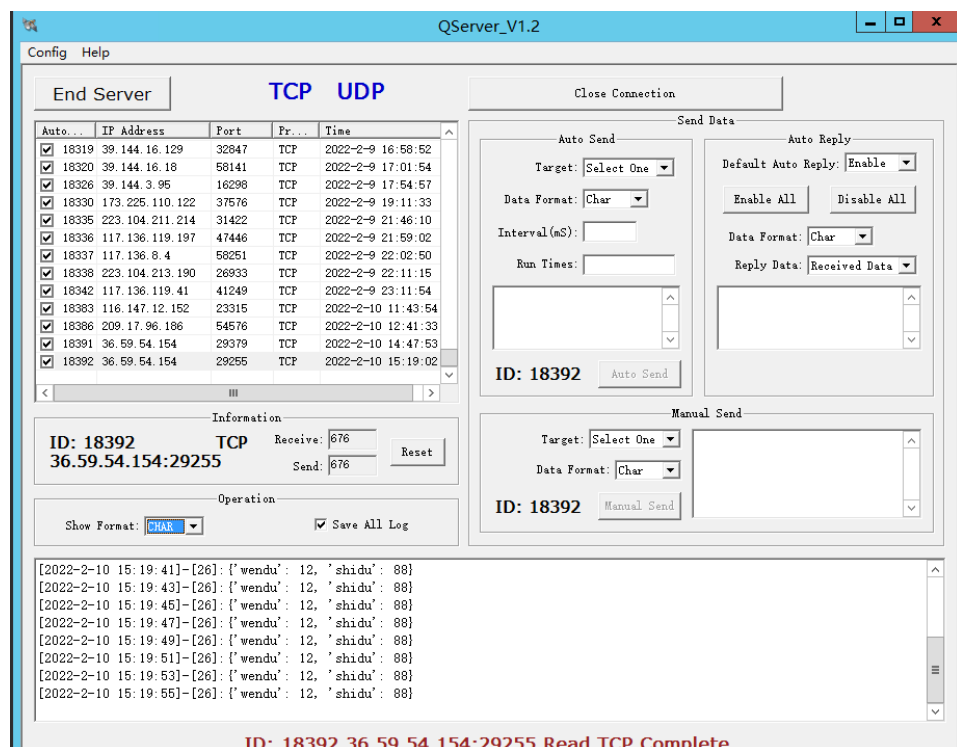
```
56 while True:
57     if ec200uctr.tcpconok==1:#conn
58         if(ec200u.TCPSend(b'AT+QISEND=0,'+str(len(str(wenshidu)))+'\r\n','>')==1):#发送数据长度
59             if(ec200u.TCPSend(str(wenshidu),'SEND OK')==1):#发送数据成功
60                 print("data send is OK")
61                 time.sleep(1)#单位是s
62                 ec200uctr.EC200U_REC()
```

上面的 while 函数和 C 语言的 while 功能是一样，是一直循环使用的。在程序里面先判断 tcp 连接标志位为 1 之后，就调用 ec200u 的 TCPSend 函数进行数据发送。如果数据发送成功，会在屏幕上打印数据发送成功。程序也支持服务器下发数据控制板子的外设，不过当前的 TCP 接收函数经常容易丢失数据，这个与程序的代码有关系，目前还不是特别完善，用户可以参考接收函数使用。



The image shows a terminal window with a dark blue background and white text. It displays two identical sequences of network-related messages. Each sequence starts with a prompt '>', followed by 'SEND OK', then a received data packet '+QIURC: "recv",0,26' and its JSON content {'wendu': 12, 'shidu': 88}. This is followed by 'data send is OK' and another prompt '>'. The messages are repeated twice, indicating a successful round-trip of data transmission.

上图是树莓派对外打印的信息。可以看到数据发送成功，并且打印了 QIURC 这个关键词，这个主要是服务器下发的相关信息并对外打印出来。



这个是服务器接收到的数据信息。可以看到服务器收到的数据内容和树莓派发送的数据内容是一样的。这样就实现了 TCP 的数据发送。

3.4 UDP 数据发送

UDP 和 TCP 使用逻辑是一样的，代码上只需要将连接服务器的 TCP 指令改成 UDP 即可，其他的与 TCP 保持一致。

```

25     if ec200u.ATSignle('AT+QIOPEN=1,0,\"UDP\",'+ServerIP+'\",'+Port+',0,1\r\n',
26         self.tcpconok=1#连接上服务器了
27     else:
28         self.tcpconok=0#fail

```

这个里面的 TCP 连接的关键词改成 UDP 即可，UDP 是无连接模式，即便服务器不存在都不出现连接失败的情况，除非指令错误。

```

10 ServerIP = '47.92.146.210'
11 Port = '9999'

```

此外需要注意 TCP 服务器和 UDP 服务器的端口，我们这里使用的是不一样的，用户如果需要改动到自己服务器，也是根据自己的需要改成自己的服务器端口才可以使用。

3.5 GPS 测试解析

EC200U 是支持 GPS 北斗定位的，用户使用 GNSS 功能的时候，可以参考 EC200U 的 GNSS 手册，根据指令来获取 GNSS 定位数据，因为 GNSS 获取到的数据都是原始数据，需要经过简单的计算将数据处理成可以在地图中使用的数据内容。

参考“Quectel_EC200U-CN_GNSS_应用指导_V1.0.0_Preliminary_20201217”来对 GNSS 开机并获取对应的 GNSS 数据。

3.1. 打开与关闭 GNSS

该示例使用默认参数来打开 GNSS。打开 GNSS 后，NMEA 语句默认从“usbmodem”端口输出。通过 AT+QGPSEND 可关闭 GNSS。

```
AT+QGPS=1           //打开 GNSS
OK
//打开 GNSS 后，NMEA 语句默认从“usbmodem”端口输出
AT+QGPSLOC=0        //获取定位信息
+QGPSLOC: 061951.0,3150.7223N,11711.9293E,0.7,62.2,2,000.00,0.0,0.0,110513,09
OK
AT+QGPSEND          //关闭 GNSS
OK
```

上面是对 GNSS 开机与关机指令。用户时候 GNSS 的时候，首先需要先开启 GNSS 功能之后，才可以获取到 GPS 相关数据。

```
AT+QGPSGNMEA="GGA"  //获取 GGA 语句
+QGPSGNMEA: $GPGGA,103647.0,3150.721154,N,11711.925873,E,1,02,4.7,59.8,M,-2.0,M,,*77
OK
```

上面是获取 GGA 语句，从 GGA 获取到的参数可以看到，数据内容包含了纬度和经度数据。这样从获取到的经纬度原始数据之后进行解析出真实有效数据。这里判断的是 GGA 实际代码用的是 RMC，其实也是一样可以获取到经纬度数据的。

GNSS 使用是不需要配合 SIM 卡的，如果没有 SIM 卡，GNSS 功能依然是可以用的，但是其他的比如电话短信数据应用是离不开 SIM 卡的。

```
8 ec200u=EC200U('/dev/ttyS0',115200)
9 ec200u.ATSignle('ATE0\r\n','OK')# set gps power on
10 ec200u.ATSignle('AT+QGPS=1\r\n','OK')# set gps power on
11 time.sleep(0.5)#单位是s
```

程序写的很简洁，首先是初始化端口，因为这里用不到 SIM 卡的相关功能，所以就不做网络判断初始化等功能，主要是对 GNSS 进行开机。然后等待进入下面的操作。

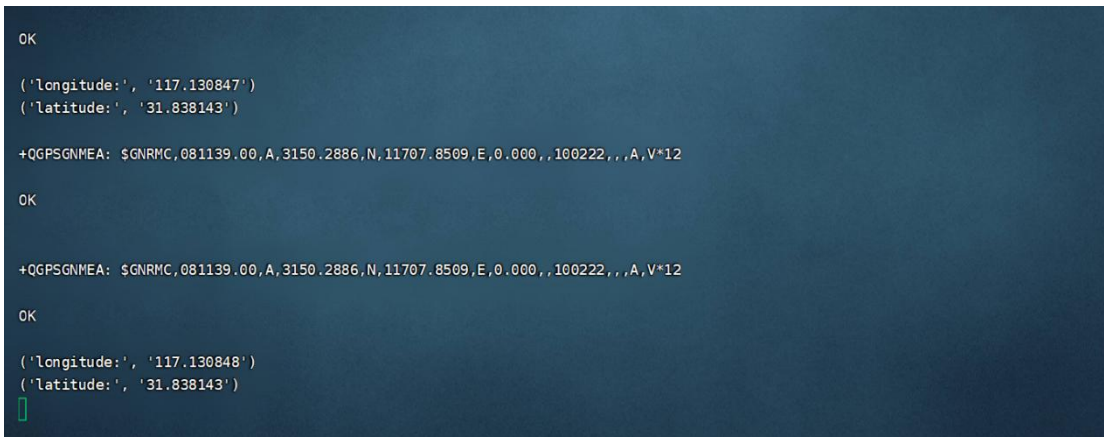
```

12 while True:
13     ec200u.ATSignle('AT+QGPSGNMEA="\rmc"\r\n','OK')# get gps data
14     print(ec200u.rxData)
15     if ec200u.rxData.find('$GNRMC')!=-1:
16         line = str(ec200u.rxData).split(',') # 将line以“,”为分隔符
17         if line[4]=='N':#dingwei is ok
18             weidu = float(line[3][:2]) + float(line[3][2:])/60
19             # 读取第5个字符串信息，从0-2为经度，即经度为117，再加上后面的一串除60将分转化为度
20             jingdu = float(line[5][:3]) + float(line[5][3:])/60
21             # 纬度同理
22             print("longitude:",'{:.6f}'.format(jingdu))
23             print("latitude:",'{:.6f}'.format(weidu))
24     time.sleep(0.5)#单位是s

```

在主循环函数里面做 RMC 指令的获取，得到 GPS 数据之后打印在屏幕上，然后判断程序里面是否有关键词 GNRMC 返回，因为只有这个参数返回才能代表 GNSS 工作正常，如果 GNSS 没有开机，发这个指令会报错的。所以用户需要注意。此外实际使用 GNSS 的时候，需要插好 GNSS 天线，并将天线放到室外才可以正常定位。

程序中将获取到的一串数据通过 split 这个关键词分成多个字段并存在一个列表中。程序通过判断列表的第四位是否为 N 代表定位成功，如果是就将经纬度数据获取出来，并根据算法将原始数据换算成实际值，然后打印出来。Python 的算法着实比 C 语言简单很多。然后利用 format 格式将经度定格在小数点 6 位。



```

OK

('longitude:', '117.130847')
('latitude:', '31.838143')

+QGPSGNMEA: $GNRMC,081139.00,A,3150.2886,N,11707.8509,E,0.000,,100222,,,A,V*12

OK

+QGPSGNMEA: $GNRMC,081139.00,A,3150.2886,N,11707.8509,E,0.000,,100222,,,A,V*12

OK

('longitude:', '117.130848')
('latitude:', '31.838143')

```

上图就是通过 AT 指令获取 RMC 值，然后通过简单的 list 获取数据处理获取到了最终的真实的经纬度数据，有了这个 GNSS 解析代码在后面的代码当中可以将这样的 GPS 数据发到云平台进行使用。

3.6 打电话

EC200U 支持电话功能，模块板上也引出了耳机孔，用户可以插入耳机进行接打电话功能，不过需要注意的是如果需要接打电话使用的 SIM 卡必须是实名制手机卡，如果使用物联网卡是不可以用的。

打电话其实很简单，只需要简单的 AT 指令即可，在 EC200 AT 指令里面有介绍。” Quectel_EC200x&EC600x&EG912Y 系列_AT 命令手册_V1.0”

7.2. ATD 发起呼叫

该命令用于建立语音或数据主叫，还可以用于控制补充业务。

ATD 发起呼叫

执行命令

ATD<n>[<mgs>][:]

响应

若无拨号音且设置 ATX2 或者 ATX4:
NO DIAL TONE

举例

```
ATD10086;           //拨号
OK
```

ATD10086;注意有分号，这样输入之后就可以进行拨号到被叫手机端了。

3.7 发英文短信

发短信和打电话一样是需要实名制手机卡的，做数据通讯是不需要的，这一点用户需要注意。

9.8. AT+CMGS 发送短消息

该命令用于将短消息（SMS-SUBMIT）从 TE 发送到网络层。调用设置命令后，返回>后输入待发数据，然后按 **Ctrl+Z** 表示 PDU 结束并发送短消息。可按 **ESC** 取消发送，取消成功也会返回 **OK** 表示停止发送。发送成功后，将返回短消息参考值<mr>到 TE。<mr>可用于根据未经请求的状态结果码识别消息。

AT+CMGS 发送短消息	
测试命令 AT+CMGS=?	响应 OK
设置命令 1) 文本模式 (AT+CMGF=1): AT+CMGS=<da>[,<tda>]<CR> 输入文本 Ctrl+Z 发送/ ESC 取消发送	响应 1) 文本模式 (AT+CMGF=1) 且发送成功: +CMGS: <mr> OK
2) PDU 模式 (AT+CMGF=0): AT+CMGS=<length><CR> 指定 PDU Ctrl+Z 发送/ ESC 取消发送	2) PDU 模式 (AT+CMGF=0) 且发送成功: +CMGS: <mr>

在 EC200 AT 手册里面有介绍关于如何进行短信发送的说明，这里发送短信采用的是文本模式，所以设置 AT+CMGF=1，因为是英文短信，所以相对比较简单。设置文本模式后，设置好短信中心号码，填入对应接收的手机号码，输入短信内容即可进行短信的发送。

```
21      ec200u.ATSignle('AT+CMGF=1\r\n','OK')#
22      ec200u.ATSignle('AT+CSCS=\"GSM\"\r\n','OK')#
```

配置短信文本模式。

```
29 ec200u.ATSignle('AT+CMGS=\"'+iphone+'\"'\r\n','>')#set call number
30 time.sleep(0.5)#单位是s
31 ec200u.Writedata(Message)
32 if ec200u.ATSignle(b'\x1a','+CMGS')==1:#set call number
33     print('SMS send ok!')
34 else:
35     print('SMS send fail')
36 while True:
37     pass
```

输入短信电话号码，这个 iphone 就是对应的手机号码，用户根据自己的需要输入自己的手机号码即可。然后输入短信内容，这里需要注意数据发送结束采用的是 0X1A 十六进制代替结束，就如手册说的是使用 CTRL+Z 来实现短信的结束发送。如果发送成功会回复 CMGS 这个关键词，用户也就可以在手机端查看到对应的短信内容了。

3.8 发数据到阿里云

阿里云物联网平台在于齐全规范，深受用户的使用和喜欢。对于 4G 模块 EC200U 而言，是可以很轻松的接入阿里云平台的，对于接入阿里云平台，首先我们提供了用户接入学习入门教程，建议用户先学习，再去考虑进行二次开发。
<https://mp.weixin.qq.com/s/4m9ActHy1lnqpa4BeYN7Qg> 这个是阿里云入门开发教程，用户可以先学习掌握之后再行 Python 语句的开发和学习。

对于 EC200U 而言使用的是 MQTT 协议接入到阿里云的，所以用户可以参考 EC200 MQTT 手册。



EC200x&EC600S&EG912Y 系列 MQTT 应用指导

LTE Standard 模块系列

版本：1.0

日期：2020-08-12

状态：受控文件

用户在平台端注册好设备之后，就可以改一下程序中的代码来进行操作了，基本的工作做完后，就可以进入程序开发了。首先看一下代码。

在阿里云主函数代码中，可以看到如下：

```
10 ServerIP = '139.196.135.135'  
11 Port = '1883'  
12 username='clientExample'  
13 ProductKey='a1qmGxDM8cd'  
14 DeviceName='mzhtest001'  
15 DeviceSecret='e47dc3c78971c3b3d00bb9ea0c5f554c'  
16 pubtopic='/sys/a1qmGxDM8cd/mzhtest001/thing/event/property/post'
```

这些都是标准参数，比如 IP 和端口号都是固定的，相对阿里云而言，对于三元素用户根据自己注册的阿里云设备获取到设备的三元素来进行对应填充，上面的参数是我们创建的设备，如果用户需要设备在线，一定要改成自己的设备才可以使用。此外还有发布主题也是需要修改为自己的才可以，否则上报的数据没法在自己的设备上显示。

上面是将三元素设置到 EC200U 里，然后进行 MQTT 连接与登录，这些流程都是固定的，用户可以不用管，但是需要注意登录到阿里云服务器之后，再进行连接。正常上面的流程跑完后，阿里云设备应该就会在线了。

Shell 打印出关键的连接消息后，查看下阿里云设备在线情况。

设备显示在线了。这样数据正常上报就没问题了。

上面是做数据上报的，可以看到在主循环不断的上报数据信息，先采集温湿度数据，然后通过服务器判断接入方式来发送数据到阿里云平台，这里采用的数据都是标准的阿里云 json 格式，上报数据也是采用 MQTT 协议进行上报的。

在物模型的地方可以看到上报了温湿度数据，数据被正常上报。这样上报阿里云就实现了。

3.9 发数据到 ONENET 平台

3.9.1 注册 ONENET 平台

在百度当中输入 ONENET 然后点击进入 ONENET 平台后，就可以进入到平台端了。第一次使用需要注册，现在一般都是实名制要求，用户注册的时候，根据注册需要，来进行操作即可。



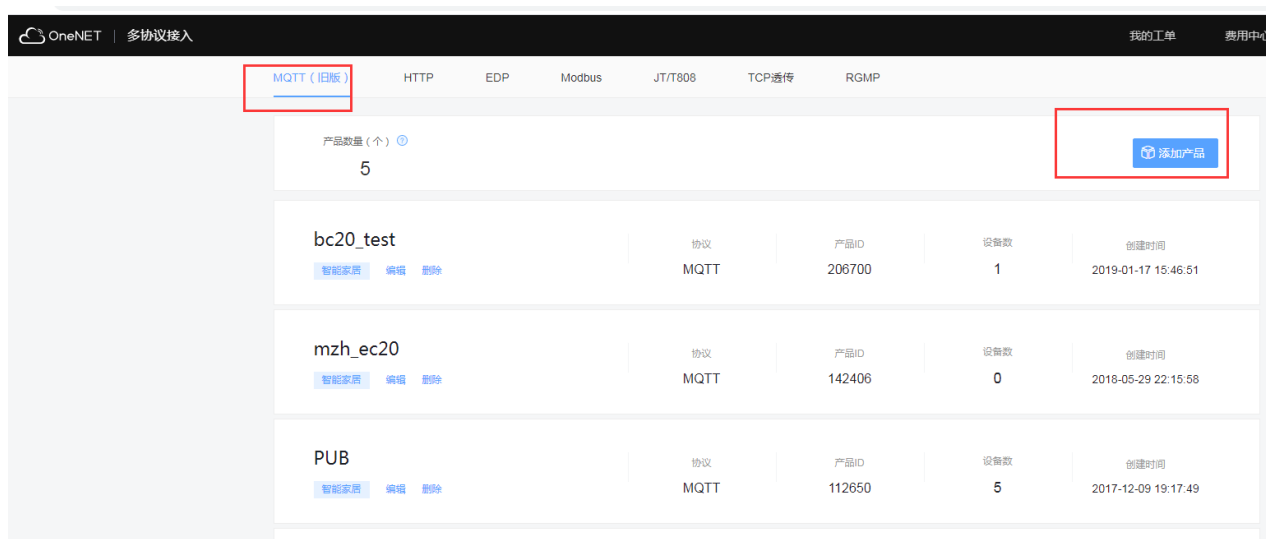
登录进入之后，就可以进行设备产品添加了，注意这里的登录协议采用 MQTT 协议，不是 MQTTS，因为现在 ONENET 改版，推荐的是 MQTTS，但是 MQTTS 目前不适用我们应用。所以用户在使用的时候，**请务必选择 MQTT 这个功能的类目接入平台。**



在产品服务里面，找到 MQTT 物联网套件，然后点击立即使用。进入到 MQTT 产品界面端。



进入到界面之后，因为这个界面我们有添加过很多设备，新注册的用户打开这个界面不一定一样，不过可以看到下面有一行原 MQTT 产品接入迁移的说明，告知用户需要使用 MQTT 协议可以点击前往旧版的标签，这样注册的产品就是 MQTT 协议了。**一定要选择多协议接入才可以进入 MQTT。**



点击进入之后，可以看到他有很多的协议类型接入，根据不同的产品来实现。对于我们的用法来说，将采用 MQTT 协议接入，其他的协议是用不到的，有兴趣的用户可以选择尝试使用其他的方式进行数据接入。

添加产品

×

产品信息

• 产品名称：

1-16个字符

• 产品行业：

请选择

• 产品类别：

请选择

请选择

请选择

产品介绍：

1-200个字符

技术参数

• 联网方式：

☐ wifi

☐ 移动蜂窝网络

• 设备接入协议：

MQTT(旧版)

若要创建其他协议套件的产品请前往相应协议套件下创建

• 操作系统：

☐ Linux

☐ Android

☐ VxWorks

☐ uC/OS

☐ 无

然后用户就可以添加产品了，在添加产品的地方，红色星号都是要求用户必须填写的，一般最好都是用英文字符数字代替，尽量不要出现中文名称。联网方式选择移动蜂窝网络，操作系统选择无即可。



添加好产品之后，就可以添加设备了。点击立即添加设备，因为产品下面可以包含多个设备，并且这样的设备就是对应的硬件。所以一定要添加设备方可使用。

添加新设备

×

* 设备名称 :

mzh001

* 鉴权信息 :

1234567891

* 数据保密性 :

☒ 私有 ☐ 公开

添加设备里面的有个鉴权信息，这个非常重要，后面在设备登录的时候，需要用到，显示应该是可以自己定义的数据，具体鉴权信息用户可以自己添加，使用数字与符号，尽量不要用一些特殊字符代替。

设备ID	设备名称	设备状态	最后在线时间	操作
580515635	mzh001	离线	-	详情 数据流 更多操作 ✓

这样就注册好了，然后设备会显示离线，这时只要单片机端控制模块发送数据到 ONENET 平台进行激活就可以了。

3.9.2 发数据到 ONENET 平台

在平台端注册好设备之后，下面开始发数据到 ONENET 平台。对于 ONENET 操作而言也是有所谓的三元素，只不过对应的是用户名账户密码这些名词和阿里云有一些区别。

```
10 ServerIP = '183.230.40.39'
11 Port = '6002'
12 ProductID = '358963'#chanpin ID
13 DeviceID = '608442797'#shebei ID
14 AutherID='12345678'#jianquan
15 pubtopic='$dp'
```

上面是 ONENET 的接入 IP 和端口，注意端口号是 6002 不是常用的 1883，用户需要注意。此外分别有产品 ID，设备 ID 和鉴权信息，这些信息在 ONENET 平台上都可以查询到的。

```

25 ec200u.ATSignle('AT+QMTDISC=0\r\n','OK')#对上一次连接断开，不需要判断是否关闭成功
26 ec200u.ATSignle('AT+QMTCFG="version",0,4\r\n','OK')
27 if ec200u.ATSignle('AT+QMTOPEN=0,"'+ServerIP+'",'+Port+'\r\n','+QMTOPEN: 0,0')==1:#连接MQTT服务器，并确认连接状态
28     self.tcpconok=1#连接上服务器了
29 else:
30     self.tcpconok=0#fail
31 if self.tcpconok==1:
32     if ec200u.ATSignle('AT+QMTCONN=0,"'+DeviceID+'",'+ProductID+'",'+AutherID+'"\r\n','+QMTCONN: 0,0,0')==1:
33         self.mqttconok=1#mqtt is ok
34     else:
35         self.mqttconok=0#mqtt is fail

```

上面是 ONENET 登录代码，是哟的也是标准的 MQTT 协议，用户需要注意的就是 ONENET 使用的 MQTT 版本是 3.1.1 版本。与其他的平台 MQTT 协议不一样，所以用户需要配置 MQTT 版本。这样才可以正常使用，否则即便其他参数正确，如果没有设置版本，那么依然会登录失败的。

```

63 while True:
64     if ec200u.mqttconok==1:#conn
65         Message='{\"datastreams\": [{\"id\": \"temp\", \"datapoints\": [{\"value\": '+str(wenshidu['wendu'])+'}], {\"id\": \"humi\", \"datapoints\":
66         MessageLength=str(len(Message))#get data length
67         totalLength=str(int(MessageLength)+3)
68         head0=chr(1)
69         head1=chr(int(MessageLength)/256)
70         head2=chr(int(MessageLength)&255)
71         head=head0+head1+head2
72         ec200u.ATSignle('AT+QMT PUBEX=0,0,0,0,\"'+pubtopic+'\",'+totalLength+'\r\n', '>')
73         ec200u.Writedata(head)#send headdata
74         if ec200u.ATSignle(Message, '+QMT PUBEX: 0,0,0')==1:#发送数据成功
75             print("data send is OK")
76             time.sleep(1)#单位是s

```

上面是发数据到 ONENET 平台端，ONENET 发数据注意他有数据头的要求。所以用户需要注意直接发一段 JSON 格式数据是发送不了到服务器的，必须加上帧头一起发送才可以使用。

数据类型 1(type == 1)格式说明：

Byte 1	数据类型值: 1 //1: JSON 格式 1 字符串	0	0	0	0	0	0	0	1
Byte 2	//指示后面 json 字符串长度								
Byte 3	固定两字节长度高位字节，值为 0x00								
Byte 4	固定两字节长度低位字节，值为 0x41								
Byte 5	{								
...	“datastreams”:[// 可以同时传递多个数据流								
...	{								
...	“id”:“temperature”,								
...	“datapoints”:[//数据流可以有多个数据点								
...	{								
...	“at”:"2013-04-22 22:22:22"//可选								
...	“value”:36.5//用户自定义								
...	}								
...	]								
...	},								
...	{								
...	“id”:“location”								
...	“datapoints”:[...]								
...	},{ ... }								
...	}								
Byte n	}								

这个是在 ONENET 的 MQTT 手册里面有这样的介绍，发数据格式是需要先发数据点类型以及 json 数据长度，这里有 3 个字节必须使用的是十六进制。发完

之后才能跟着 json 数据。相对阿里云而言复杂一些。大家发数据的时候一定要注意这个地方。

```
63 while True:
64     if ec200u.AT+MQTTCONOK==1:#conn
65         Message="{\"datastreams\": [{\"id\": \"temp\", \"datapoints\": [{\"value\": '+str(wenshidu['wendu'])+'}]}, {\"id\": \"humi\", \"datapoints\": [{\"value\":
66         MessageLength=str(len(Message))#get data length
67         totalLength=str(int(MessageLength)+3)
68         head0=chr(1)
69         head1=chr(int(MessageLength)/256)
70         head2=chr(int(MessageLength)&255)
71         head=head0+head1+head2
72         ec200u.AT+SIGNLE('AT+QMQTTPUBEX=0,0,0,0',''+pubtopic+'\\', '''+totalLength+'\\r\\n', '>')
73         ec200u.Writedata(head)#send headdata
74         if ec200u.AT+SIGNLE(Message, '+QMQTTPUBEX: 0,0,0')==1:#发送数据成功
75             print("data send is OK")
76             time.sleep(1)#单位是s..
```

在发送数据的地方，加了 3 个十六进制数据，分别代表数据点和数据长度，然后单独调用发送指令将数据发送出去，然后将后面的 json 内容进行发送。这样在 ONENET 平台就可以显示对应的数据了。用户可以按照这个说明测试下即可。