

1. 运动传感器/心率：可以上传频率到服务器，由服务器查看；
2. 网络延迟：使用时间戳，服务器端将两次接收到的数据中时间戳相减，得出两次收到数据的间隔，再对比程序中的时间间隔来得到大致延迟；数据包使用异或校验，来检测是否丢包。
3. 带宽/数据包容量测试：手表向服务器发送文件，记录不同文件大小和所用时间来得到带宽和容量；
4. 多环境：室内、室外进行测量；
5. 续航影响：同表格

2025.08.07 13：25

第一次测试，**网络延迟丢包测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：200ms
4. 手表端上传数据包间隔：100ms
5. 电量：51%
6. 网络：联通4G
7. 地点：实验室
8. 时间：3min

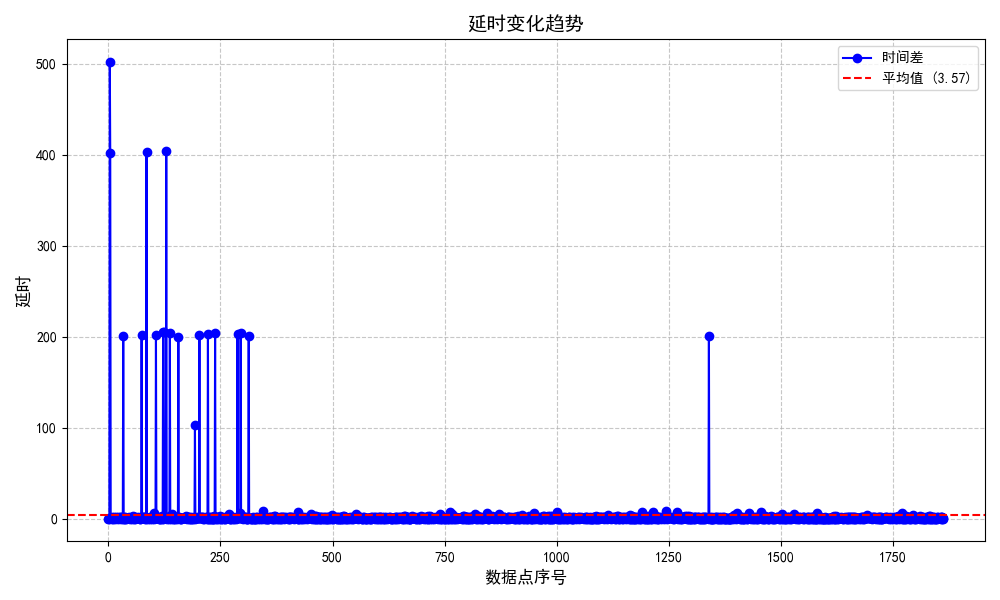
结果：

在30s开始可以读取到用户心率，共计测试188s，发送1864个包

平均9.91包/s

**丢包：**37/1864，丢包率1.98%

**延时统计信息：**平均值: 3.57ms 最小值:0ms 最大值: 502ms



2025.08.07 13：59

第二次测试，**网络延迟丢包测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：200ms
4. 手表端上传数据包间隔：100ms
5. 电量：49%
6. 网络：蓝牙连接手机，手机网络使用广电5G
7. 地点：实验室
8. 时间：3min

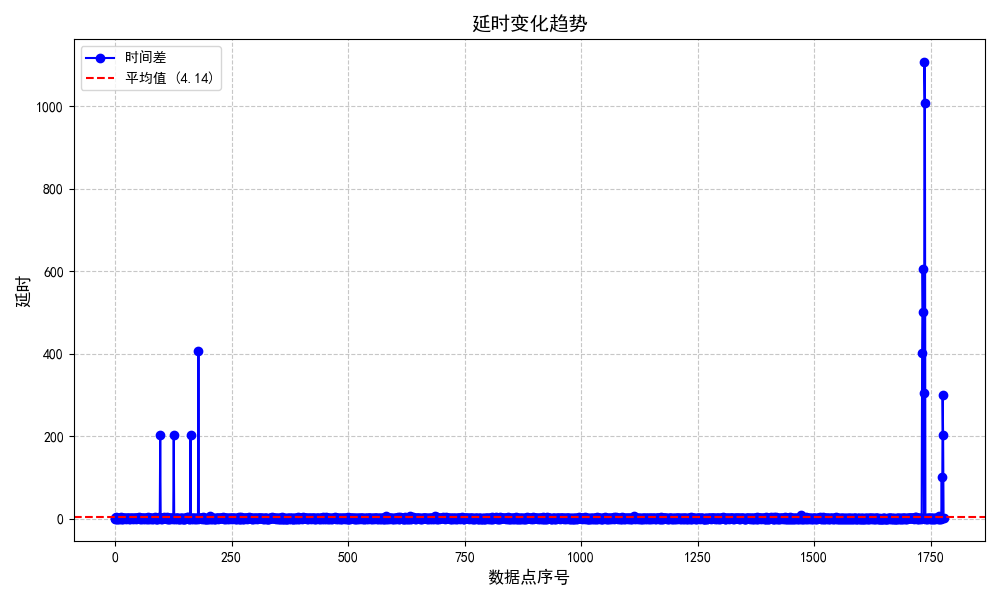
结果：

共计测试181s，共发送1779包

平均9.82包/s

**丢包：**10/1779，丢包率0.5%

**延时统计信息：**平均值: 4.14ms 最小值: 0ms 最大值: 1108ms



2025.08.07 14：37

第三次测试，**网络延迟丢包测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：200ms
4. 手表端上传数据包间隔：100ms
5. 电量：42%
6. 网络：联通4G
7. 地点：第四教学楼门口**（室外环境）**
8. 时间：3min

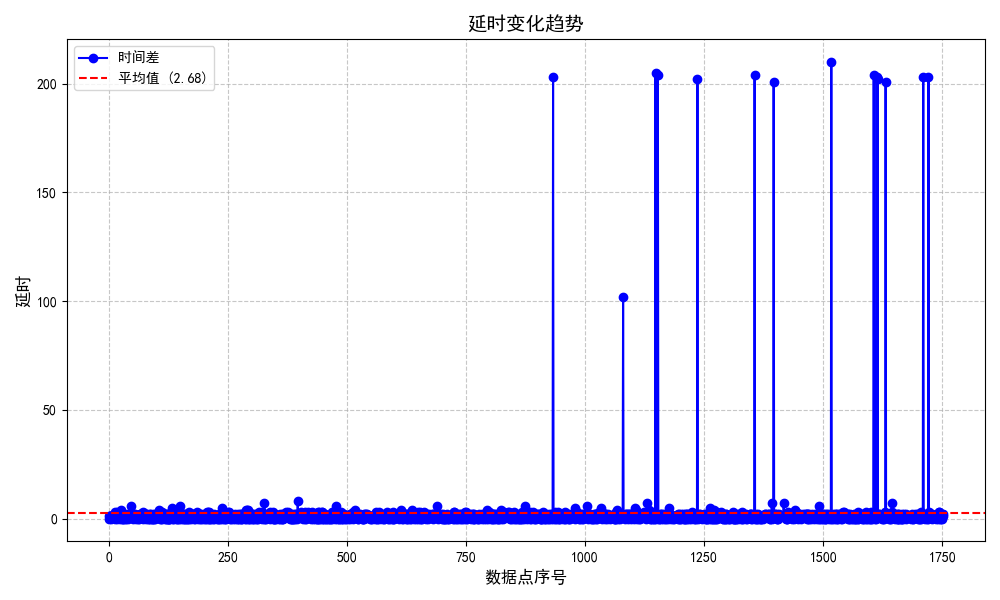
结果：

共计测试180s，共发送1753包

平均9.73包/s

**丢包：**27/1753，丢包率1.54%

**延时统计信息：**平均值: 2.68ms 最小值: 0ms 最大值: 210ms



2025.08.07

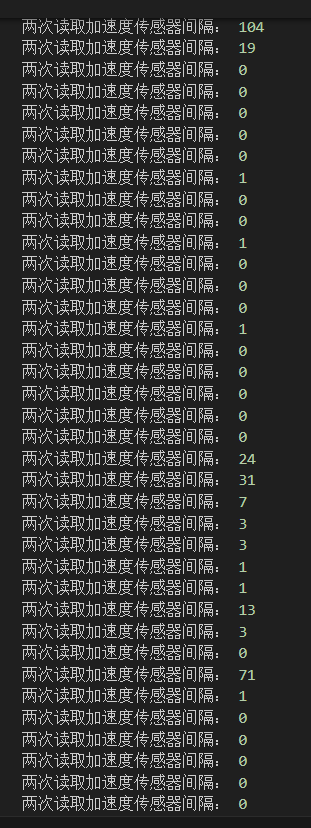
第一次测试，**加速度传感器频率测试**

**（测试方法：用两个全局变量now\_time和last\_time 记录每一次读取传感器数据的时间和上一次读取传感器数据的时间，相减得出相邻两次读取的时间间隔，并记录在系统日志中）**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：10ms
4. 手表端上传数据包间隔：50ms
5. 电量：28%
6. 网络：Cselab
7. 地点：实验室

结果：



部分时间出现大量无效数据，说明以100hz的频率读取信息会出现问题

2025.08.07

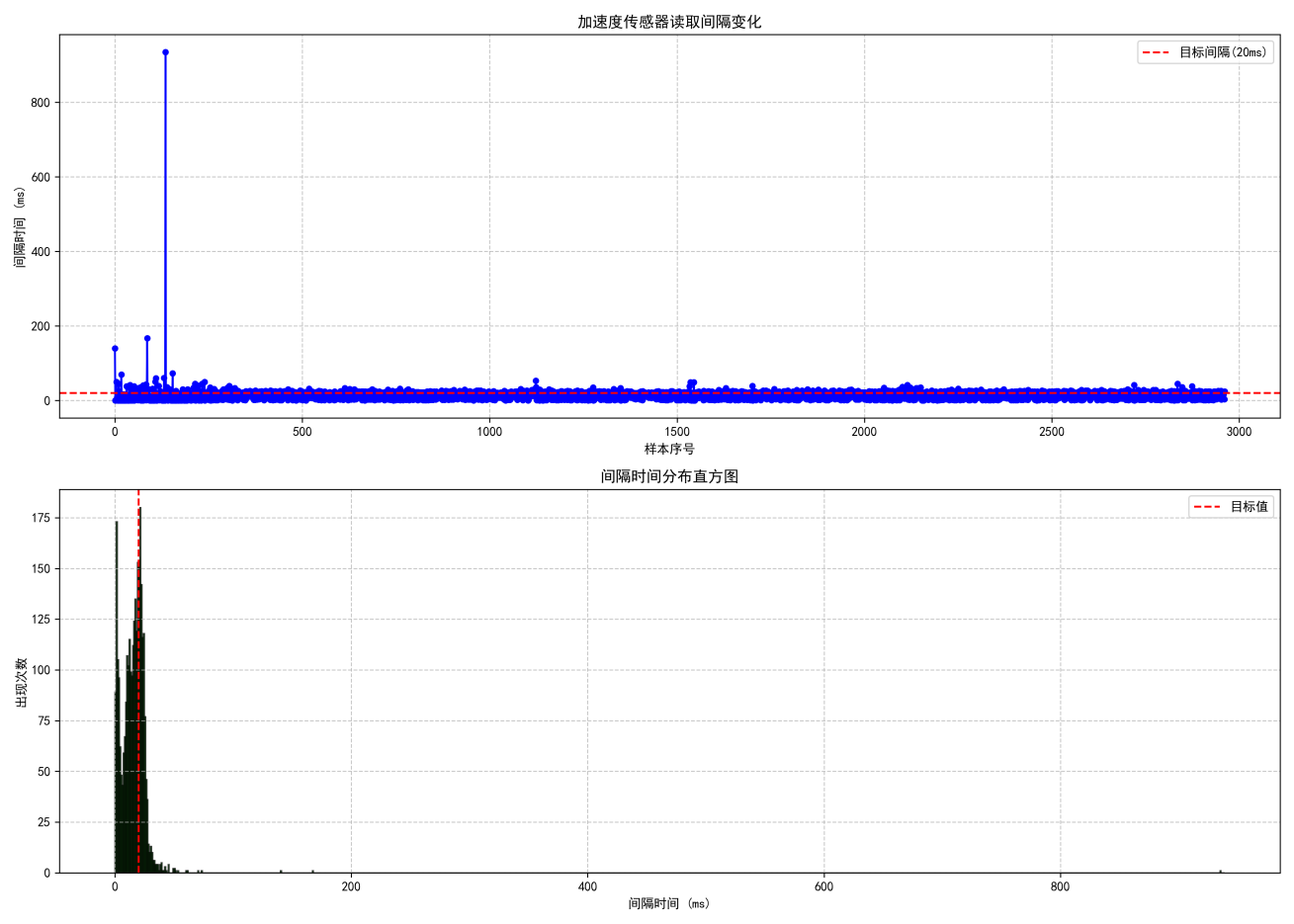
第二次测试，**加速度传感器频率测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：20ms
4. 手表端上传数据包间隔：50ms
5. 电量：28%
6. 网络：Cselab
7. 地点：实验室

结果：

数据点数: 2962 平均值: 15.10 ms 中位数: 16.00 ms 最小值: 0.0 ms 最大值: 935.0 ms 标准差: 19.35 ms



可以看到，当设置间隔时间为20ms时，仍有大量数据间隔小于20ms

在测试的18:38:38.468-18:39:36.012时间内

共58s，读取了2962个数据，平均每个数据间间隔19.58ms，证明读取数据的频率确实在20ms左右，可能是记录日志时无法高速记录

2025.08.07

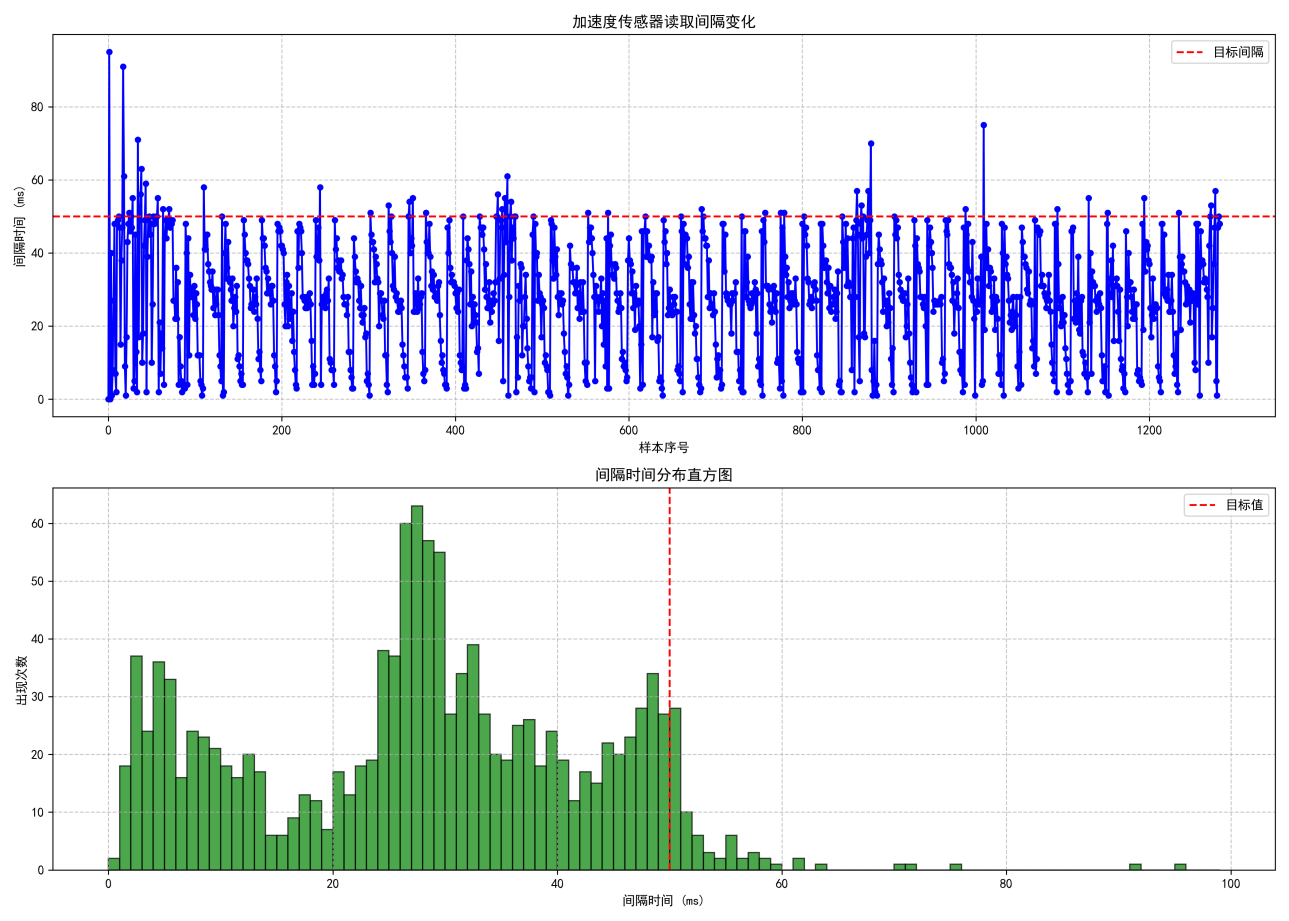
第三次测试，**加速度传感器频率测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 手表端上传数据包间隔：50ms
5. 电量：24%
6. 网络：Cselab
7. 地点：实验室

结果：

数据点数: 1282 平均值: 27.34 ms 中位数: 28.00 ms 最小值: 0.0 ms 最大值: 95.0 ms 标准差: 15.05 ms



同样可以看到，当设置间隔时间为50ms时，仍有大量数据间隔小于50ms

19:08:03.670 - 19:09:05.995

共约62s 记录1282个数据，平均间隔48.36ms

2025.08.07

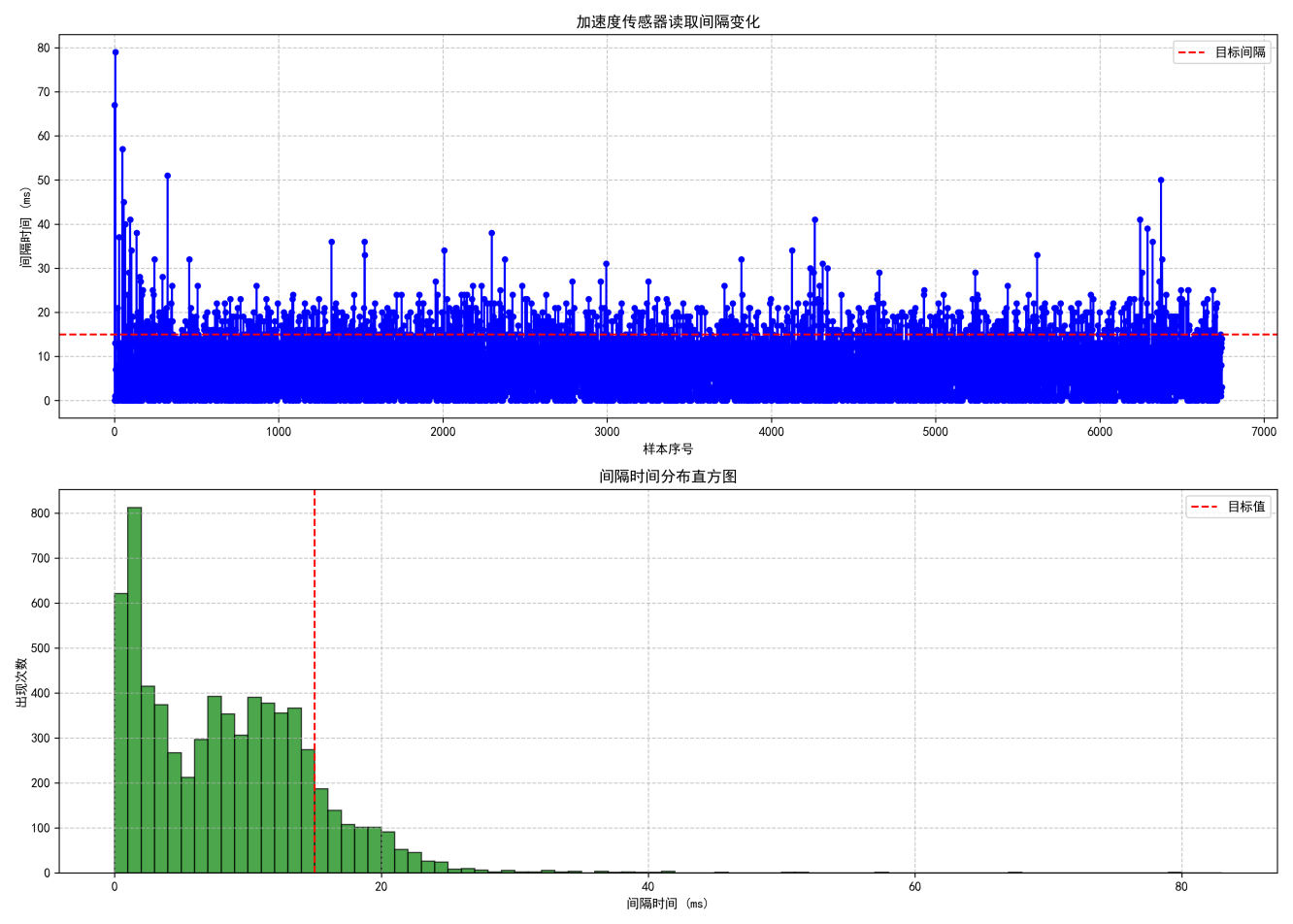
第四次测试，**加速度传感器频率测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：15ms
4. 手表端上传数据包间隔：50ms
5. 电量：19%
6. 网络：Cselab
7. 地点：实验室

结果：

数据点数: 6744 平均值: 7.94 ms 中位数: 7.00 ms 最小值: 0.0 ms 最大值: 79.0 ms 标准差: 6.42 ms



19:34:31.272 - 19:35:36.862

约65.5s，共记录6744个数据，平均9.7ms记录一个？？？估计66hz（15ms）的频率存在问题

在上传过程中，发现除了丢包外，服务器端接收到的数据包解析常常出现格式错误，因此引入网络状态检测和重试机制，建立连接时，会先尝试状态检测，确认和服务器通信通过后，才会进行数据包的发送。同时在屏幕显示中加入一个状态指示，用户可以清晰的看到与服务器的连接状态。

try {  
 // 重试机制（最多3次）  
 let retryCount = 0;  
 while (retryCount < 3) {  
 const result = await this.tryUpload(httpRequest);  
 if (result.success) {  
 this.isConnected = true;  
 this.connectionStatusColor = Color.Green;  
 return;  
 }  
 retryCount++;  
 await new Promise<void>(resolve => setTimeout(resolve, 1000 \* retryCount)); // 指数退避  
 }



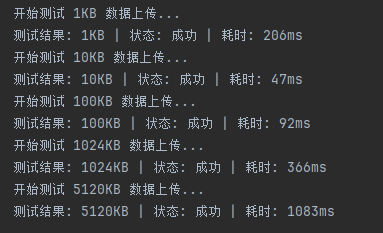
在服务器端，通过临时文件替换原文件的保存方式，降低了文件被多次读写的可能，数据保存出错率大大下降

2025.08.08

第一次测试，**数据包上传容量测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 电量：83%
5. 网络：Cselab
6. 地点：实验室

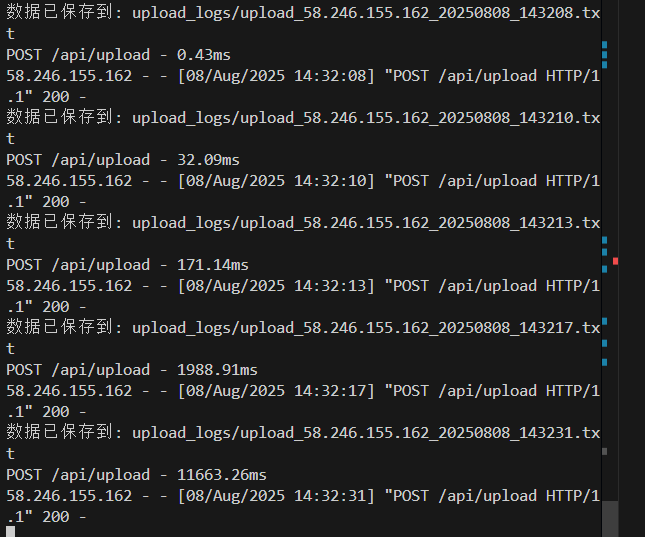


2025.08.08

第二次测试，**数据包上传容量测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 电量：83%
5. 网络：联通4G
6. 地点：实验室



图为服务器接收数据所用时间，并非上传时间，仅做展示，证明手表端上传5m以内的数据包都没问题

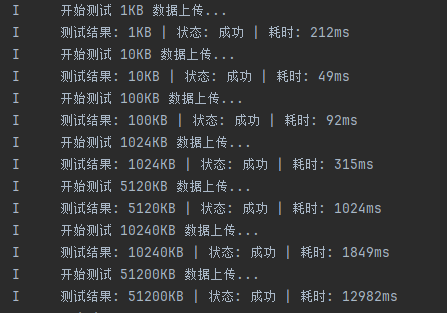
2025.08.08

第三次测试，**数据包上传容量测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 电量：80%
5. 网络：Cselab
6. 地点：实验室

结果：



当数据达到50MB时，手表也能成功上传，我认为更高的数据量可能也可以，但是在本项目中没必要再测试更大的数据包了，50MB应该足够使用

在网上的地图中，存在地图坐标偏移的问题，根据我国相关法律，在国内发行的任何民用[中国地图](https://so.csdn.net/so/search?q=%E4%B8%AD%E5%9B%BD%E5%9C%B0%E5%9B%BE&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/i_likechard/article/details/_blank)产品不准使用真实的坐标，必须加入一定的偏移量(大约600米)。

测试思路：

1. 在空旷、开阔的地方静止保持3分钟，记录经纬度位置，看是否有漂移较大情况出现。
2. 在学校操场或有固定位置信息可知的地方进行走动，记录数据，分析定位是否会出现明显错误，并测试数据的更新频率。
3. 根据测试得到的坐标点绘制运动轨迹，对于水上运动中，只需知道相对的位置变化，即记录运动开始后的轨迹即可，对于精确到地图上某一点的位置要求没有这么高

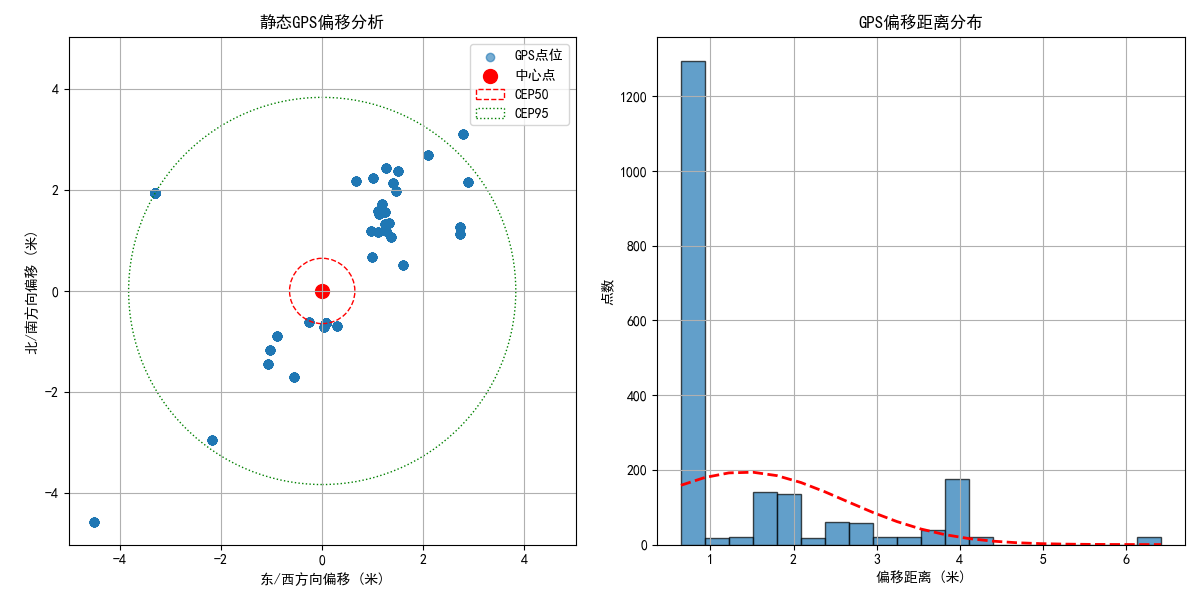
2025.08.11

第一次测试，**GPS测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 手表端上传数据包间隔：50ms
5. 电量：57%
6. 网络：联通4G
7. 地点：第四教学楼旁边

结果：



在静态时，数据分布较大，南北方向约2m，东西方向约2m内分布

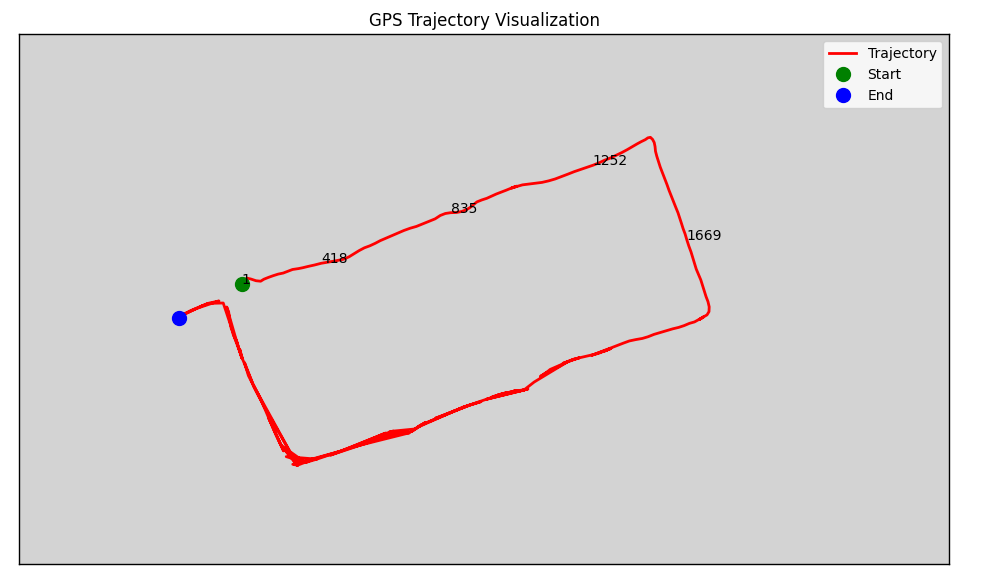
2025.08.11

第二次测试，**GPS动态测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 手表端上传数据包间隔：50ms
5. ~~电量：57%~~
6. 网络：联通4G
7. 地点：校内骑行一周

结果：



对比手机轨迹



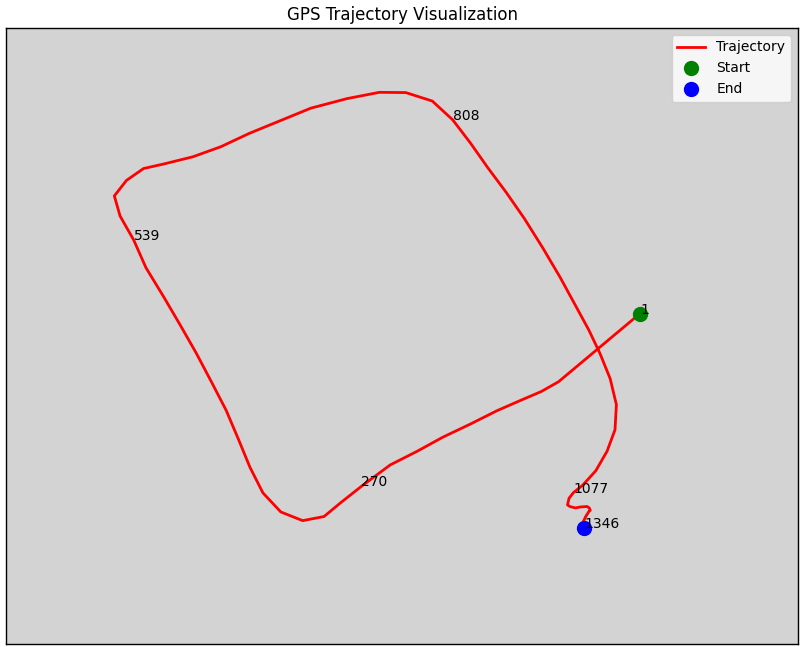
2025.08.11

第三次测试，**GPS动态测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 手表端上传数据包间隔：50ms
5. ~~电量：57%~~
6. 网络：联通4G
7. 地点：校内骑行一周

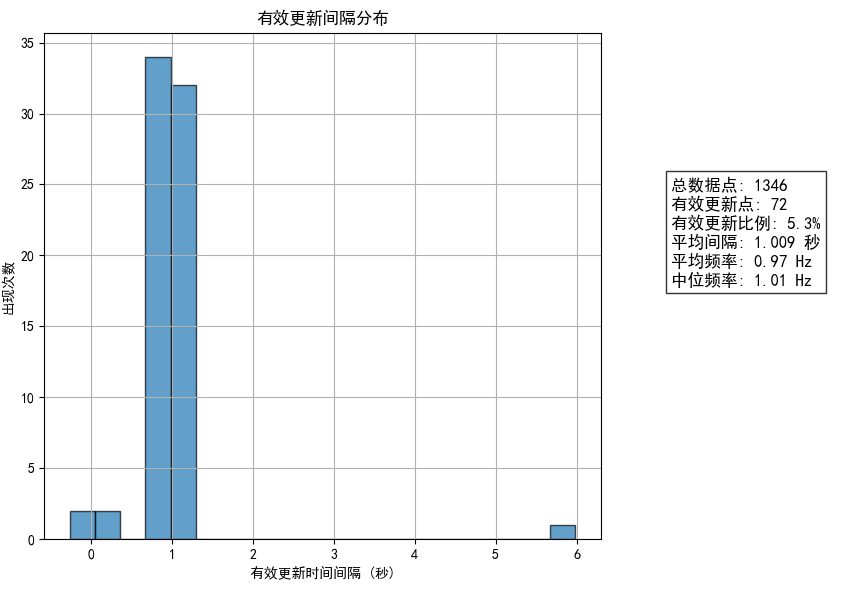
结果：



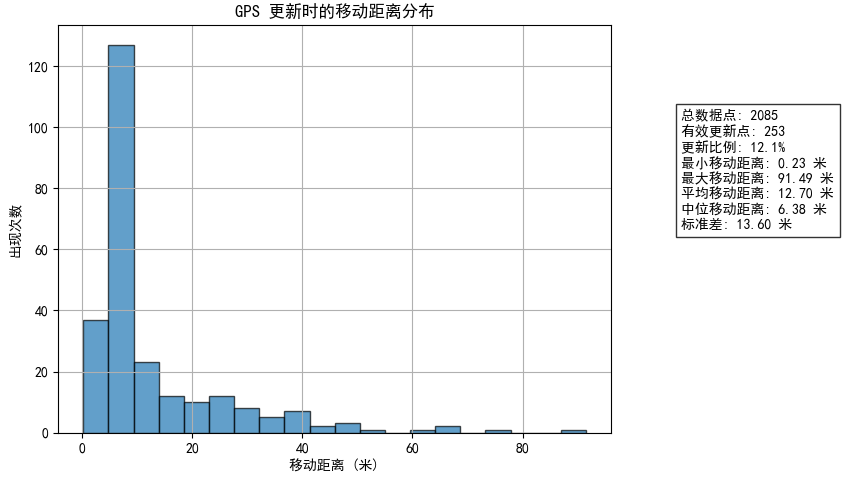
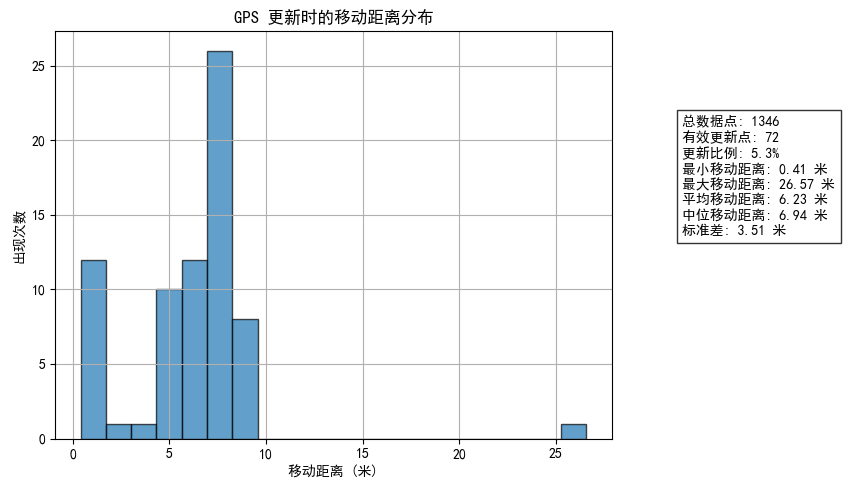
对比手机轨迹：



GPS数据更新频率：



不同点间移动距离分析：



（第二张图中，由于我测试时笔记本休眠，ssh连接断开，导致中间有一段路程有问题）

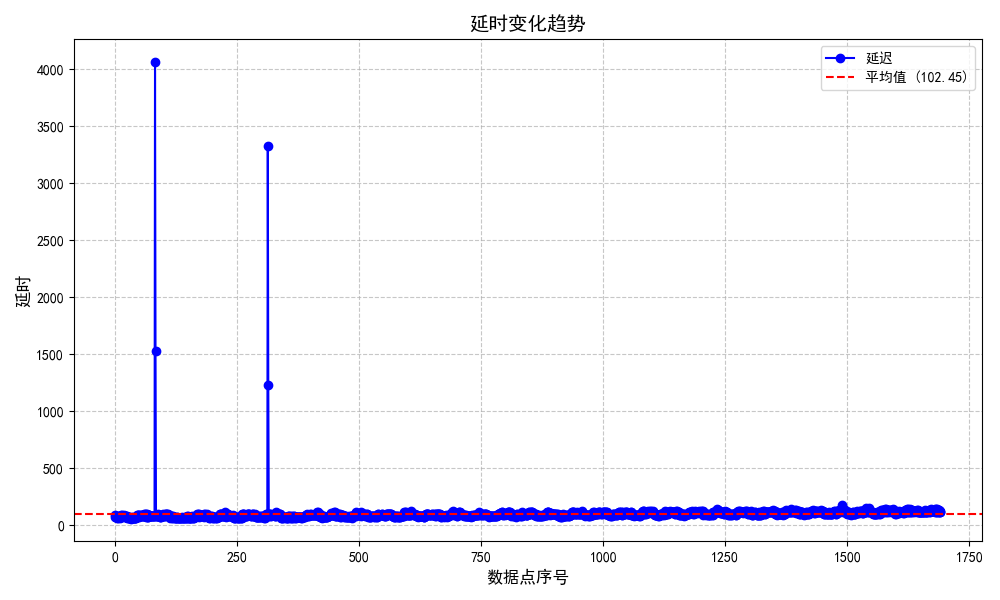
2025.08.11

第一次测试，**网络延迟测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 手表端上传数据包间隔：100ms
5. 电量：38%
6. 网络：联通4G
7. 地点：实验室

结果：



延时统计信息：

数据点数量: 1688

平均值: 96.68ms

最小值: 59ms

最大值: 4100ms

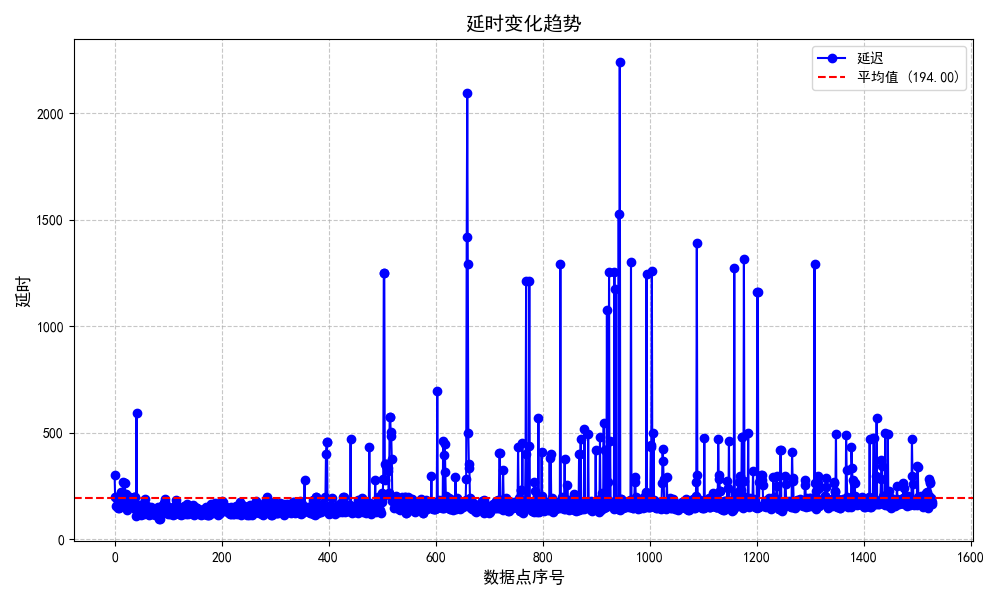
2025.08.12

第二次测试，**网络延迟测试**

设备参数：

1. 心率：1000ms
2. GPS：位置变化更新模式，高功耗定位
3. 加速度传感器：50ms
4. 手表端上传数据包间隔：100ms
5. 电量：95%
6. 网络：联通4G
7. 地点：第四教学楼门口

结果：



延时统计信息：

数据点数量: 1529

平均值: 194.00ms

最小值: 96ms

最大值: 2240ms

**GPS频率问题**

查询文档得知GPS读取位置的信息中有一参数ContinuousLocationRequest 用于定义GPS数据上报的时间间隔，单位是秒。默认值为1，取值范围为大于等于0。等于0时对位置上报时间间隔无限制。发帖求助，有官方技术支持回复：



解决方案：后续可尝试通过蓝牙连接外置GPS设备来读取信息。

**水下网络测试：**

在1.8米深泳池底部测试，网络连接状态时好时坏，基本是不可用状态

**本地缓存问题：**

当手表丢失网络连接，或服务器无法连接时，手表应在本地具有缓存数据的功能，以防数据丢失。

1. 创建一个缓冲队列，当手表开始采集数据后，先将数据存放在队列中。随后开始尝试网络连接。若网络连接不成功，则手表将数据继续保存；若网络连接成功，且与服务器通信测试成功，则开始从队头发送数据。

定义一个dataUploader类来完成数据入队、上传

class dataUploader {  
 private dataQueue: Queue<UploadData\_check> = new Queue(); // 缓冲队列  
 public isNetworkAvailable: boolean = false;  
 private isUploading: boolean = false;  
 private readonly MAX\_QUEUE\_SIZE = 1000; // 队列最大容量

//将数据添加到队列  
 public addDataToQueue(sensorData: UploadData): void   
 // 网络状态检测方法  
 private async checkNetwork(): Promise<boolean>

// 上传控制逻辑  
 private async tryStartUpload(): Promise<void>

// 从队列上传数据  
 private async startUploadFromQueue(): Promise<void>

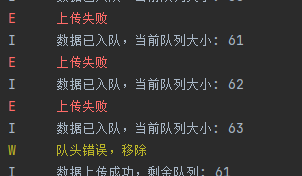
// 单包上传逻辑  
 private async uploadSinglePacket(packet: UploadData\_check): Promise<boolean>   
 // 在类初始化时启动定时器  
 private setupRetryTimer(timeInterval:number): void

}



测试中，发现服务器有时会连续返回上百个异或校验错误，然后手表端不在继续上传数据，而只会在队列中添加。

原因：队头的数据出现校验错误，导致一直无法发送，没有出队。



修改后加入一个上传重试上限次数，当队头上传3次还不成功后，直接移除这个队头，进行下一个数据的发送，成功解决了该问题。

在之前的设计中，手表打开应用即开始上传数据，若刚开始未连接网络，很容易造成队列过长溢出，因此添加了一个按钮，按下按钮可以开始上传或停止上传，同时创建一个专门的定时器（1s）用于监测服务器连接状态。



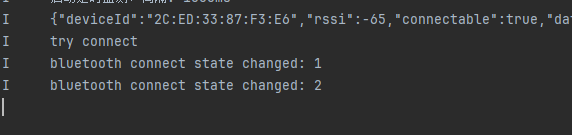
蓝牙ble测试：

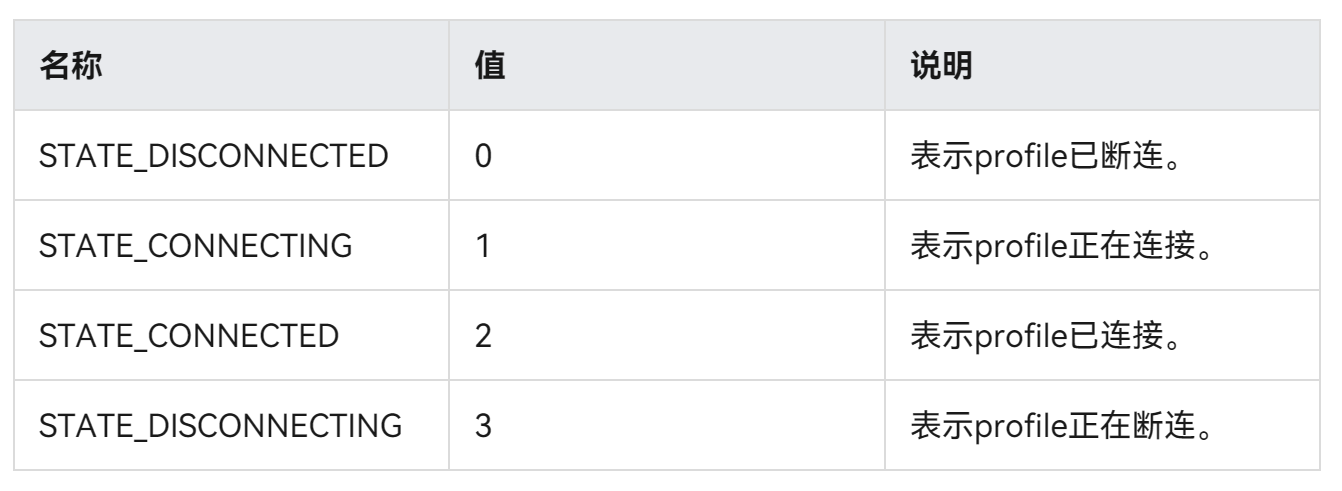
通过调用ble.startBLEScan([scanFilter])，可以使手表开启对周围BLE设备的扫描，并通过ble.on('BLEDeviceFind', this.printBLE);进入回调函数。在打印的信息中查看，可以看到手表成功搜索到WT901BLE67姿态传感器设备。

[{"deviceId":"2C:ED:33:87:F3:E6","rssi":-60,"connectable":true,"data":{"0":2,"1":1,"2":5,"3":11,"4":9,"5":87,"6":84,"7":57,"8":48,"9":49,"10":66,"11":76,"12":69,"13":54,"14":55,"15":17,"16":7,"17":251,"18":52,"19":154,"20":95,"21":128,"22":0,"23":0,"24":128,"25":0,"26":16,"27":0,"28":0,"29":229,"30":255,"31":0,"32":0},"deviceName":"WT901BLE67"}]

调用接口创建Gatt客户端设备，只需要输入找到的设备ID即可建立连接

let device: ble.GattClientDevice = ble.createGattClientDevice(deviceID);  
 device.on('BLEConnectionStateChange',this.confirmConnect)  
 device.connect();





成功与蓝牙设备进行连接