参赛队员姓名：卞思婉 孟盛 刘德华

中学：成都市华阳中学

省份：四川省

国家/地区：中国

指导教师姓名：宋岷生

论文题目：共享单车非接触型智能车锁系统

本参赛团队声明所提交的论文是在指导老师指导下进行的研究工作和取得的研究成果。尽本团队所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果。若有不实之处，本人愿意承担一切相关责任。

参赛队员：_____ 指导老师：_____

2018 年 **7** 月 **18** 日

论文题目：共享单车非接触型智能车锁系统

作者：卞思婉 孟盛 刘德华

论文摘要：共享单车作为共享经济的一种新形态出现在公共区域，给市民带来了极大的便利，但是共享单车被违规占有、恶意使用的现象较多，部分用户停车常常忘关锁。

本系统由客户端模块、服务器模块、电子锁模块组成，主要优化了共享单车的车锁系统。手机客户端通过蓝牙与单车建立连接，获得单车蓝牙 MAC 地址并将其发送至服务器。服务器通过 GPS 定位用户与单车位置，确定两者的对应关系后，向单车电子锁发送指令开锁；改进单车内部结构，采用齿轮传动原理，通过收放脚架完成单车锁的开关；将电子锁置于单车的车架中，利用舵机转动将插销插入单车转向龙头转轴和脚踏板转轴预置好的凹槽内，实现单车的关锁功能。

多次模拟实验证明，本系统利用蓝牙有效、安全地解锁共享单车，规避了二维码、车牌号解锁存在的问题，并能缩短用户开锁时间；通过收放脚架巧妙地完成单车锁的开关，使用户使用起来更加便捷、放心；将电子锁隐藏于单车的车架中，增强了单车的防盗能力。

关键词：共享单车；GPS 定位；非接触式开关锁；蓝牙识别；齿轮传动原理；车锁隐藏

目　录

1 前言

1.1 选题背景

随着无线技术和信息技术的快速发展，“互联网+”在日常生活中扮演着越来越重要的角色，成为我们生活中不可割离的部分。蓝牙凭借其成本低、功耗低及操作简单的优势，成为了一种广泛使用的短距离无线传输工具[1]。

目前市面上最常见的共享单车开锁方式为扫描二维码和输入车牌号。

扫描二维码开锁方式虽然能让用户较方便地使用共享单车，但如果二维码被涂抹、刮花，就很容易出现“锁死”现象，使得用户无法开锁及使用共享单车。另一方面，用户扫描包含广告、诈骗等错误信息的二维码后，可能会遇到不必要的麻烦甚至导致用户手机信息被盗窃，这严重地危及了用户的隐私安全和财产安全。同样，车牌号也容易被涂抹、刮花，导致用户无法辨认，进而影响正常开锁。

图 1 被人为损坏的二维码

同时共享单车的计时收费的机制对用户存在着许多不便。例如，当用户忘记关锁后，后台将持续计费，且在此期间易被他人骑走，不利于共享单车的管理。

图 2 忘关锁造成的经济损失

共享单车车锁被恶意破坏的现象频频发生，浔阳晚报 2017 年 5 月 18 日报道：“男生恶意破坏车锁 将共享单车当二手车卖给修车行老板”。映象网快讯：2018 年 4 月 15 日早上，有市民发现在郑州经开区航海东路与第十大街交汇处一公园内，有 7 辆共享单车车锁遭到人为破坏，有的还私自加了链子锁占为私用，有市民指责这种把“共享”变“独享”的行为太自私。

图 3 单车电子锁被人为破坏

1.2 当下共享单车运营方案

目前在共享单车中使用率较高的是 ofo、摩拜和青桔。

摩拜的蓝牙智能锁里有内置 GSM/GPRS、GPS 定位模块和蓝牙模块，用户手机扫码后读取单车蓝牙信息，尝试与单车配对。并发送该二维码信息到服务器端，经处理后，服务器端同时向车辆和手机发送开锁指令，如果车辆先接受到指令直接开锁。如果手机先接受到指令，则通过蓝牙把指令发送给车辆进行开锁。但是其为了确定单车与用户的对应关系，对二维码和单车车牌号等易失效的外部

信息依赖性太大。

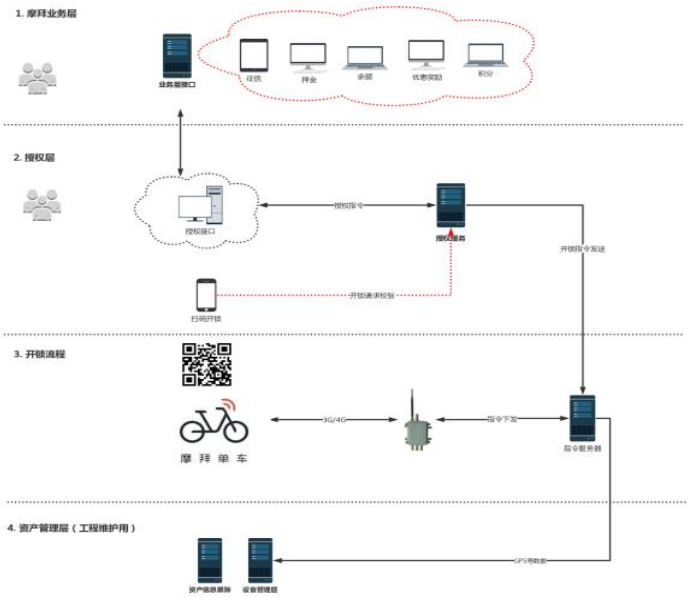

图 4 摩拜单车运营方案

ofo 第二代同样需要扫描单车二维码或输入单车 ID，随后程序给出一个开锁密码，但需在单车锁上输入开锁密码，方能开锁。其内部的开锁原理类似银行的 U 盾密码卡，其密码每分钟变换一次，输入单车 ID 后，程序后台会计算其密码。但在实际使用中会碰到旋转密码失效的问题，因为单车锁内部的时钟出现误差，导致密码无效。紧接着 ofo 单车又推出了第三代，但是都没有规避二维码这一根本性问题[2]。

近日华为公司推出的内置 NFC 的 Huawei Pay 与 ofo 公司合作后，又萌生了一开锁方式——NFC 智能开锁。用户只需将手机贴近智能锁感应区域便可直接开锁，但由于目前国内 NFC 普及程度并不高，对大部分用户的手机来说无 NFC 硬件支持，导致该开锁方式不适用于大众用户且短期内无法普及[3]。

在收费机制上，上述三种主流共享单车都采用计时的收费方式，且服务器将用户手动关闭电子锁时视为使用结束，以致无法解决用户忘关锁造成经济损失的问题。

在上述三种共享单车中，其电子锁都置于单车座垫与后轮之间的座架上，极易被小偷破坏并将单车盗走。

1.3 研究目标

（1）探索有效、安全的开锁方式，解决共享单车的二维码、车牌号易被损坏所导致的问题。

（2）探索用户离车就关锁的方法，解决用户使用后因忘记关锁造成的经济损失问题。

（3）探索将电子锁由显形变为隐形的方法，解决电子锁被恶意破坏、违规占有的问题。

2 设计思路

2.1 总体设计原理及思路

本系统分为三个模块，分别为客户端模块、服务器模块、电子锁模块，通过非接触的方式完成共享单车智能开关电子锁、放脚架巧妙关锁、将电子锁置于单车的车架中等功能，解决目前共享单车电子锁出现的问题。系统框架图如下：

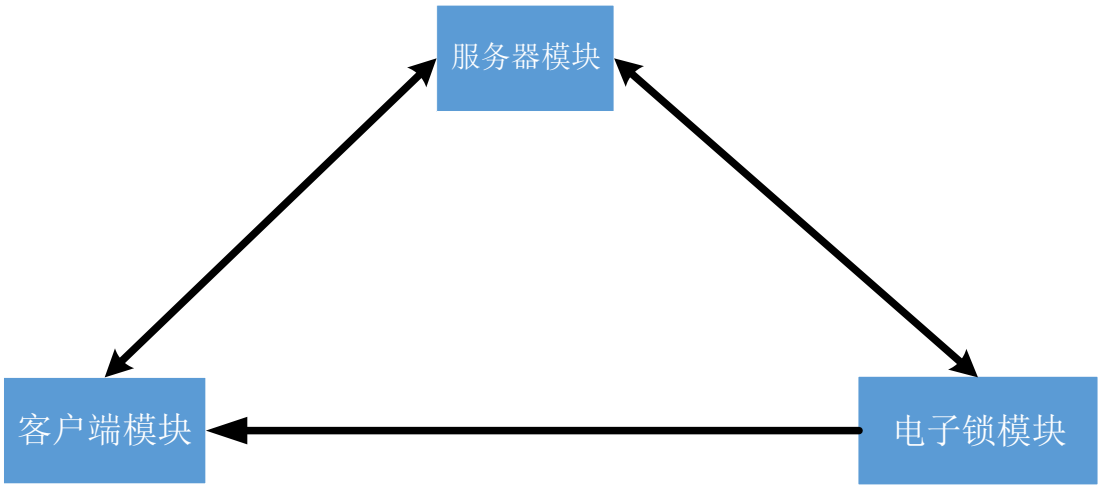

图 5 系统框架图

2.1.1 非接触式开锁模式设计

首先，用户在单车附近，打开手机 APP，通过蓝牙扫描获取用户附近十 m 内的单车位置信息；然后，用户选择 APP 界面上将要使用的单车，与之建立蓝牙连接，获取单车蓝牙 MAC 地址；第三步，用户通过单车发出的声光提示找到该单车，点击“立即用车”，并将蓝牙 MAC 地址发送至服务器，服务器确定单车与用户的对应关系，向单车电子锁发送开锁指令完成开锁。开锁流程图如下所示：

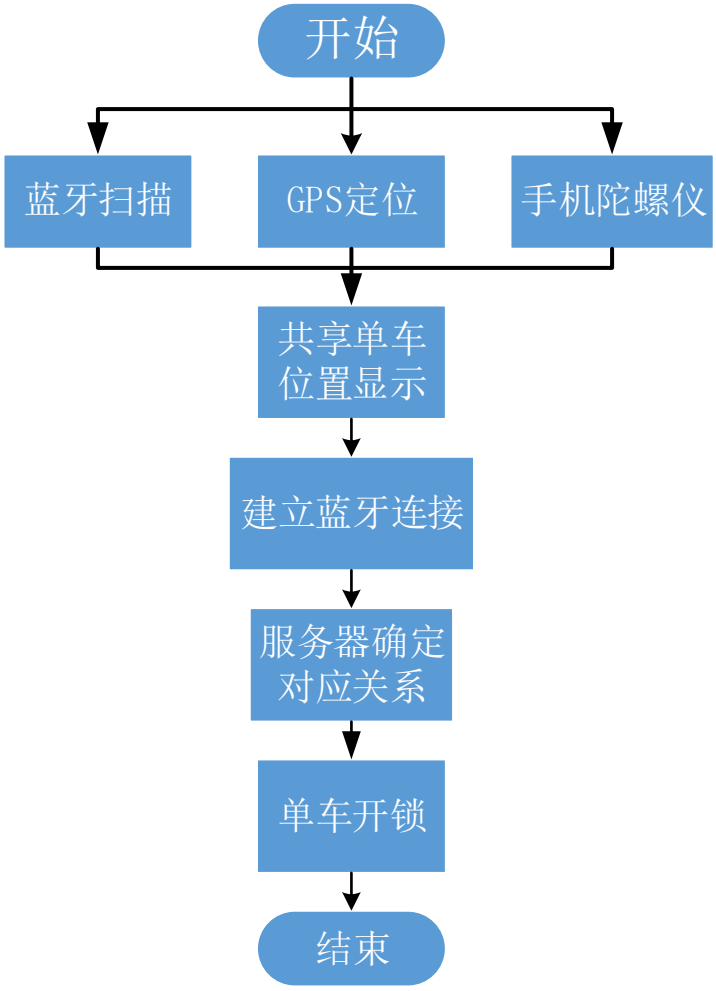

图 6 开锁流程图

2.1.2 共享单车巧妙关锁功能设计

使用齿轮传动原理，改进单车内部结构，通过放下脚架巧妙地完成电子锁的关锁功能。当用户暂停骑行放下脚架的同时，单车内部运用齿轮传动的原理实现关锁的功能并进入待机状态，且在 5min 内不会接受服务器的开锁指令。若用户在 5min 内将继续使用此单车，可点击手机上的“继续使用”使单车开锁继续供此用户使用，若用户不再使用，5min 后单车将自动解除与此用户的绑定关系，等待下一位用户的用车请求。在用户骑行完毕后，放下脚架，单车关锁并再次进入待机状态，循环上述操作。

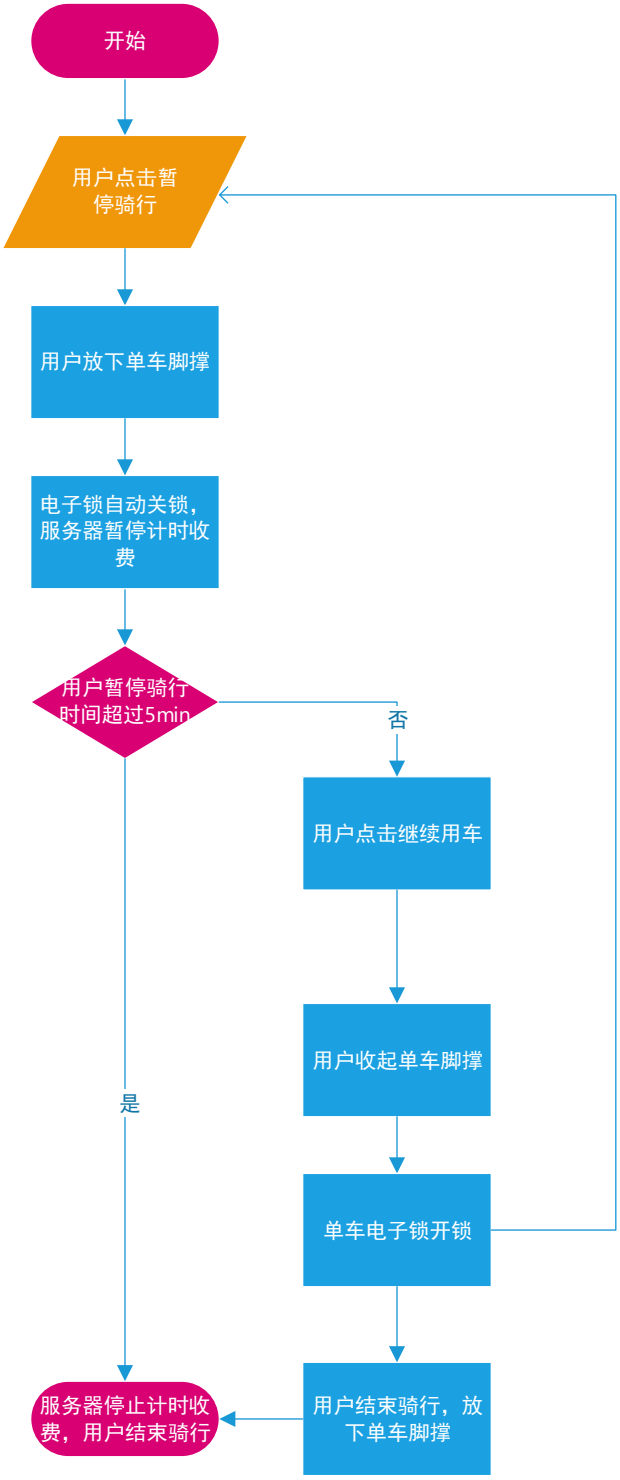

图 7 关锁控制流程图

2.1.3 共享单车隐藏式防盗锁设计

我们将电子锁置于单车的车架中，利用机械传动插销伸出插入单车转向龙头转轴上和脚踏板转轴上预置好的凹槽内，将单车转向龙头转轴和脚踏板转轴锁定，实现单车的关锁。避免了小偷与共享单车电子锁的直接接触，增强了单车的防盗能力。

2.2 系统模块设计

2.2.1 服务器模块

模拟服务器模块为一块树莓派（Raspberrypi 3B），本模块主要实现开锁流程中的数据处理和传输。在进行共享单车的开锁过程中，当用户点击手机 APP 界面上的“立即用车”后，服务器会接收到来自客户端所发出的单车蓝牙 MAC 地址，这时服务器会将此蓝牙的 MAC 地址与其数据库中的单车蓝牙 MAC 地址相配对，完成单车与用户的绑定，进而发送开锁指令至用户将要使用的共享单车，完成共享单车的开锁。在单车完成智能开关电子锁过程中，若用户暂停使用单车一段时间后，再次点击“继续使用”，服务器将读取来自客户端的继续用车的请求，并再次发送开锁指令，完成电子锁的开启，以及对此用户继续进行收费。

2.2.2 客户端模块

客户端模块为手机 APP。客户端主要是完成手机与单车建立蓝牙连接，以及蓝牙 MAC 地址的发送的功能。在进行共享单车的开锁过程中，当用户打开 APP 后，开启手机蓝牙、GPS，进行蓝牙扫描和 GPS 定位，并将共享单车的位置信息显示在手机 APP 界面的地图上供用户进行选择。待用户选择一辆单车后，与其蓝牙建立连接，完成对单车蓝牙 MAC 地址的读取。用户点击“立即用车”后，将读取到的单车蓝牙 MAC 地址通过网络发送至服务器完成单车与用户的绑定。在单车使用过程中，若用户暂停使用单车，在一段时间内，用户若要继续使用此单车，只需点击“继续使用”，手机将会再次发送请求到服务器，完成锁的开启，继续供用户使用。

2.2.3 电子锁模块

本电子锁由 Arduino nano、HC-05 蓝牙、UBLOX NEO-6M、碰撞传感器、声光模块、舵机和电源共七个部分构成。

在电子锁开锁的智能方面，本电子锁模块主要实现与用户手机之间建立蓝牙连接。待服务器完成用户与单车的绑定后，电子锁会接收到来自服务器发送的开锁指令，使单车开锁。在单车使用过程中，用户暂停使用单车并放下脚架，使电子锁关锁并进入待机状态。在待机期间服务器仅接受来自原用户的开锁请求。若用户点击“继续使用”，电子锁将接收到服务器再次开锁的指令并完成开锁；若用户在规定时间内未点击“继续使用”，电子锁将结束待机状态并反馈信息给服务器，解除单车与用户的绑定，等待下一位用户的使用。

在电子锁开锁的机械方面，我们利用电子锁的控制中心，控制舵机转动插销锁，使在单车脚踏板转轴和方向轴转轴的插销缩回，实现单车前部锁的解锁；用户收起脚架，带动单车脚架处的齿轮转动，使带齿滑条往后缩回，实现单车后部锁的解锁，从而达到解锁共享单车的目的。

在电子锁关锁的机械方面，用户放下脚架，带动单车脚架处的齿轮转动，使带齿滑条前伸，将脚踏板转轴卡住，且触发碰撞传感器，使前部电子锁控制模块驱动舵机，控制插销锁的插销伸出，将单车的脚踏板转轴和方向轴转轴卡住，达到共享单车关锁的目的。

2.3 开发环境

2.3.1 Arduino

Arduino 能通过各种各样的传感器来感知环境，通过控制灯光、蜂鸣器等其它的装置来反馈、影响环境，通过 Arduino 的高度封装的编程语言来编写程序。

本系统以 Arduino 作为电子锁模块的控制中心，完成数据的传输、处理和各个模块的控制。本系统选用的 Arduino 型号是 Nano。

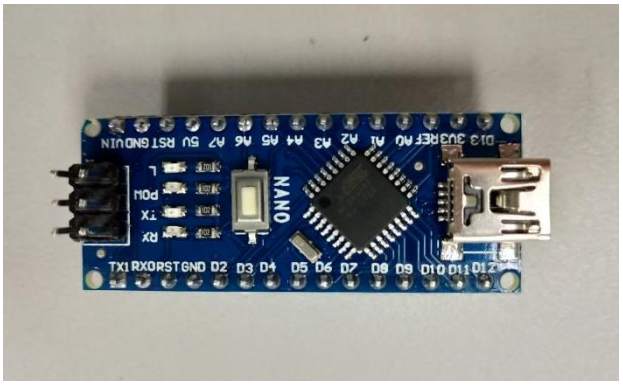

图 8 Arduino Nano

2.3.2 蓝牙

蓝牙具有短距离数据传输的功能，且功耗低、数据传输出错率低、成本少而安全性高、效率高。本系统中我们选用 HC-05 蓝牙模块完成单车与用户手机的绑定和蓝牙 MAC 地址的传输。

图 9　HC-05 蓝牙

2.3.3 舵机

舵机是一种位置伺服的驱动器，适用于需要角度不断变化并可以保持的控制系统。我们采用舵机来作为转动插销锁的动力装置，通过正转与反转完成单车前部锁的开锁与上锁。

图 10　舵机

2.3.4 GPS

GPS 能为全球用户提供低成本、高精度的三维位置、速度和时间等信息。本系统采用 UBLOX NEO-6M。

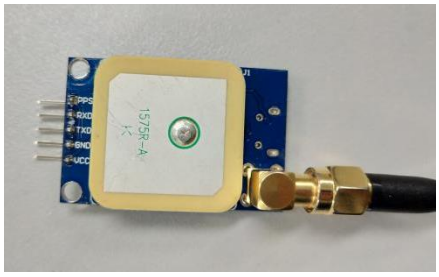

图 11 UBLOX NEO-6M

2.3.5 服务器

本系统的服务器选用 Raspberry3 B 和 3.5 寸 LCD 液晶显示屏来模拟，实现数据的处理和传输。

图 12　LCD 显示屏

图 13 Raspberry3 B

2.3.6 碰撞传感器

我们在单车后轮与脚踏板之间的一段骨架的内部下方安装碰撞传感器，在关锁过程中，单车脚架放下，带动齿条前伸并触发碰撞传感器。进而使单车前部电子锁获得单车后部锁上锁情况，并控制舵机完成单车前部锁的上锁。

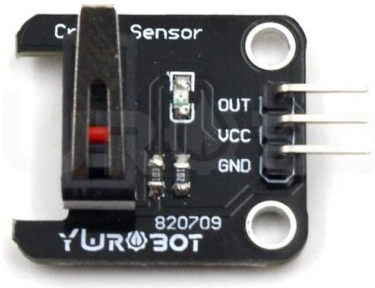

图 14 碰撞传感器

2.3.7 声光模块

为了能使用户在 APP 上选中单车后，能更便捷地寻找到目标单车，我们在单车上安装了声光模块，通过使有源蜂鸣器发声和 LED 发光来提示用户。本声光模块由一个有源蜂鸣器和三个 LED 组成，其中红灯亮表示单车锁为关闭状态，绿灯亮表示单车锁为开启状态，蓝灯亮表示单车 GPS 和蓝牙都处于正常工作状态。

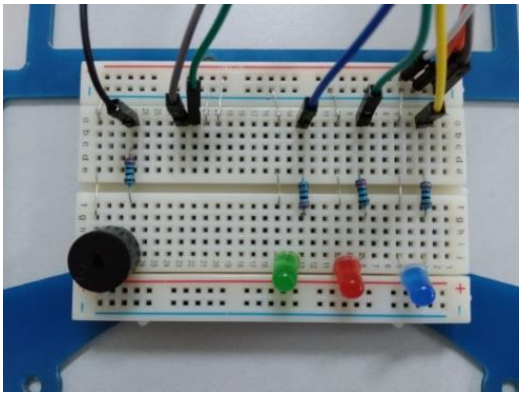

图 15 声光模块

2.3.8 插销锁

为了实现单车锁的内置，我们采用一种双向式插销锁。

图 16 双向式插销锁

3 系统软件的设计与实现

3.1 系统软件设计

3.1.1 电子锁程序设计框架

在开关单车电子锁的设计上，主要由服务器通过网络向单车电子锁中的单片机发送指令来完成对共享单车电子锁的控制。

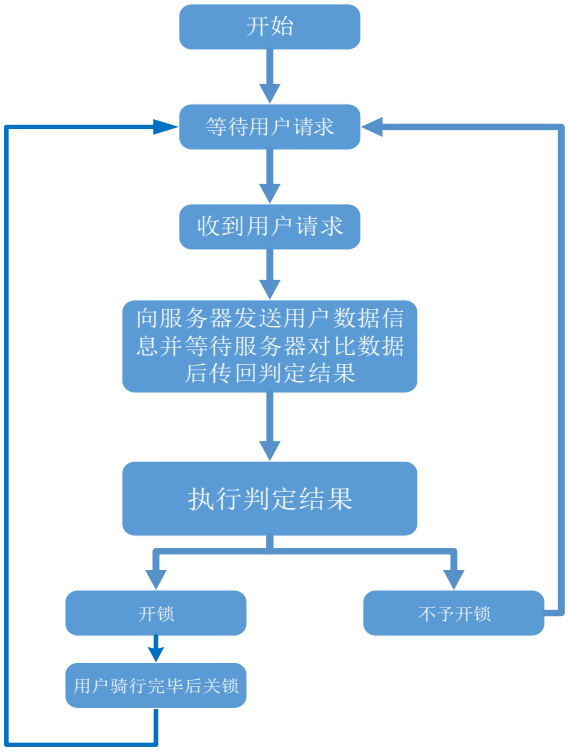

图 17 电子锁程序设计【开锁】

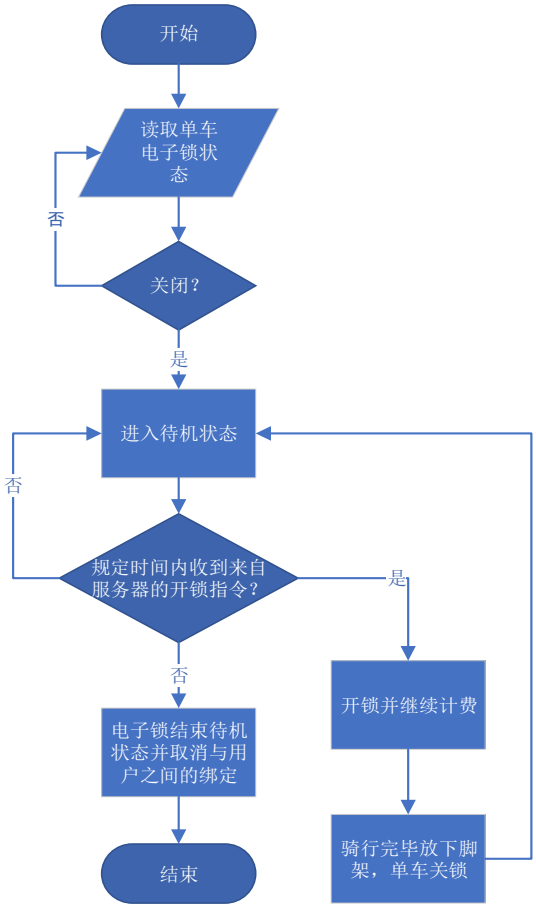

图 18 电子锁程序设计【关锁】

3.1.2 服务器程序设计框架

服务器由树莓派模拟，该模块的程序设计包括了信息传输、显示屏、数据储存和数据处理等一系列自动化操作。

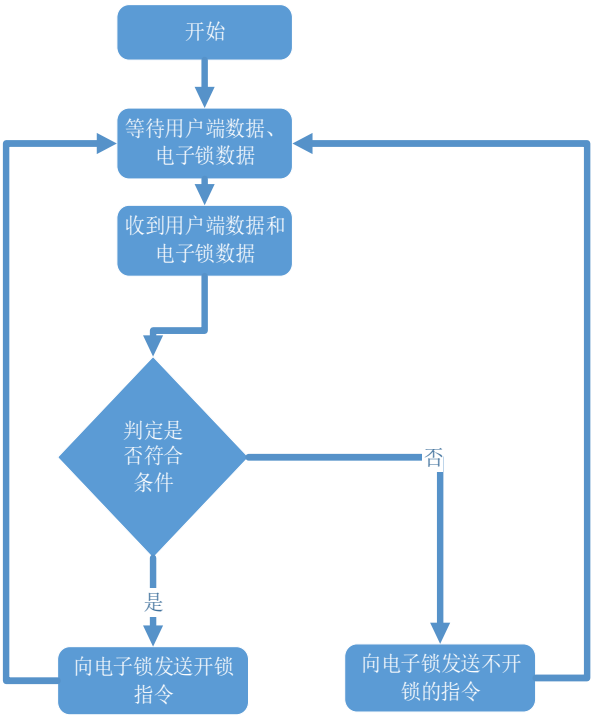

图 19 服务器端程序设计【开锁】

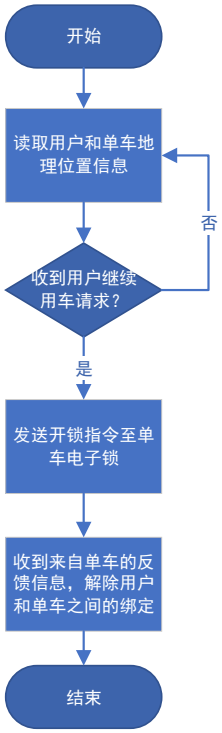

图 20 服务器端程序设计【关锁】

3.1.3 客户端程序

客户端程序主要是实现用户手机与共享单车蓝牙的连接和 GPS 定位，以及读取并发送单车的蓝牙 MAC 地址。

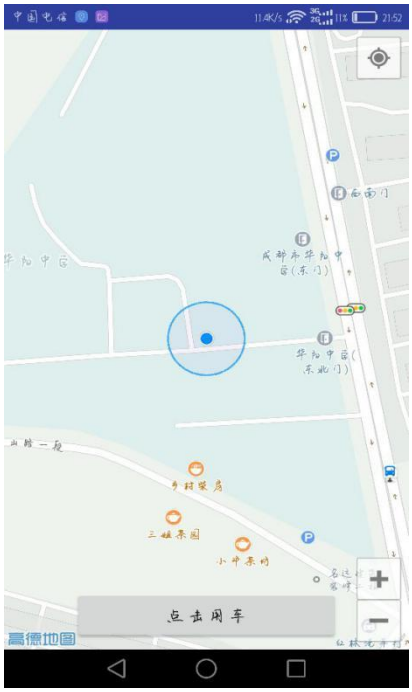

图 21 安卓程序

4 硬件设计与搭建

4.1 硬件电路图设计

在本方案中我们设计的电子锁的电路图如下：

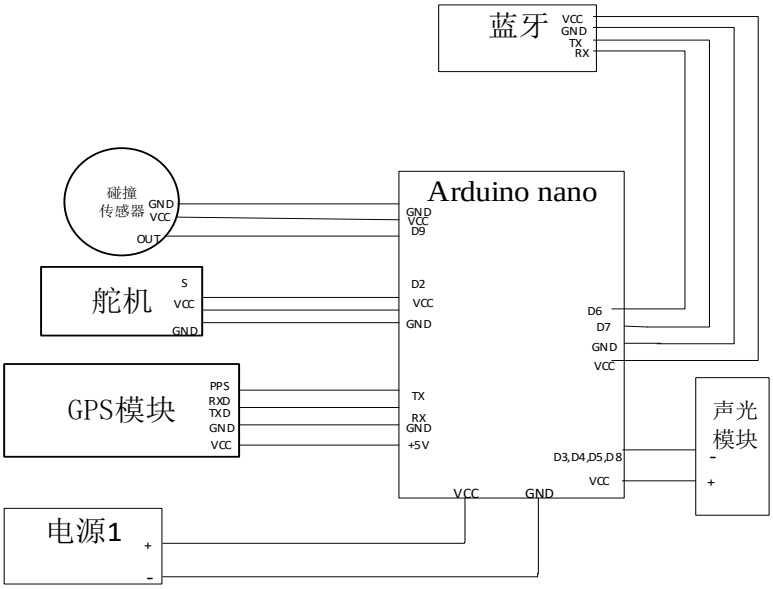

图 22 电子锁硬件电路图

4.2 电子锁的结构设计

在电子锁的机械设计上，我们采用一种双向式插销锁，通过其伸出和缩回插销来完成前部电子锁的上锁与开锁。用户选中单车后，服务器发送开锁指令至共享单车电子锁中，通过单片机控制舵机正向转动，使前部两条插销向前向后缩回，离开单车方向轴转轴与脚踏板转轴，完成前部锁的开锁；当用户将要开始骑行时，用户收起脚架，脚架带动齿轮 2 逆时针转动，齿轮 2 带动齿轮 1 顺时针转动，间接使齿条向后运动，离开单车脚踏板转轴，完成后部锁的开锁，解除单车的上锁状态。

在用户暂停或结束使用单车时，用户停下单车，将单车脚架放下，脚架带动齿轮 2 顺时针转动，齿轮 2 带动齿轮 1 逆时针转动，间接使齿条向前运动，伸入单车脚踏板转轴中，将单车脚踏板转轴卡住。在齿条前伸过程中触发在单车后轮与脚踏板之间的一段骨架的内部下方的碰撞传感器，间接使单车前部电子锁控制舵机反向转动，使前部两条插销向前向后伸出，将单车方向轴与脚踏板转轴卡住，使单车前部电子锁关锁。若用户点击“继续使用”，客户端向服务器发出请求。服务器会再次向单车电子锁发送开锁指令，使单车前部电子锁开锁。

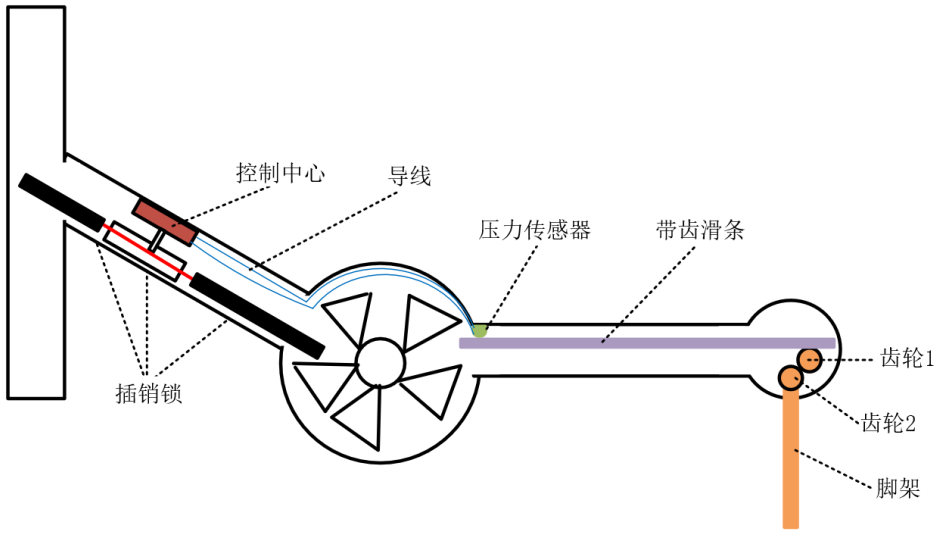

图 23 电子锁机械设计

4.3 硬件框架搭建

本模型前部锁采用直径为 5cm 的亚克力管作为锁的外壳，将 Arduino nano、HC-05 蓝牙、声光模块集成到一块电路板上，并将其与双向式插销锁、UBLOX NEO-6M、舵机和电源固定在亚克力管中。

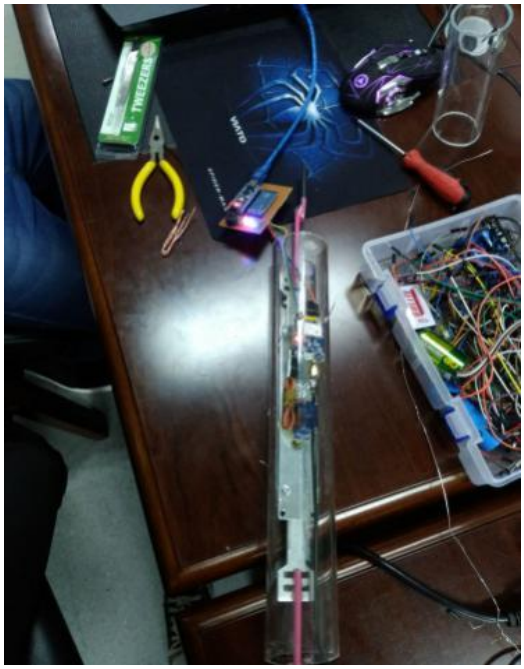

图 24 前部电子锁

本模型后部锁使用直径为 5cm 的亚克力管作为锁的外壳，在其中置入碰撞触感器。将传统脚架改为齿轮传动结构，通过收放脚架带动插销运动。

图 25 后部插销结构

图 26 机械模型组装过程

图 27 前部锁主体

5 系统测试及数据分析

5.1 系统测试方法及结果

5.1.1 系统测试方法

在本系统装置制作完成后，我们进行了模拟实验。首先我们在手机上安装 APP，打开并给予 APP 以蓝牙和 GPS 的调用权限，然后测试开锁、关锁和继续使用等功能。

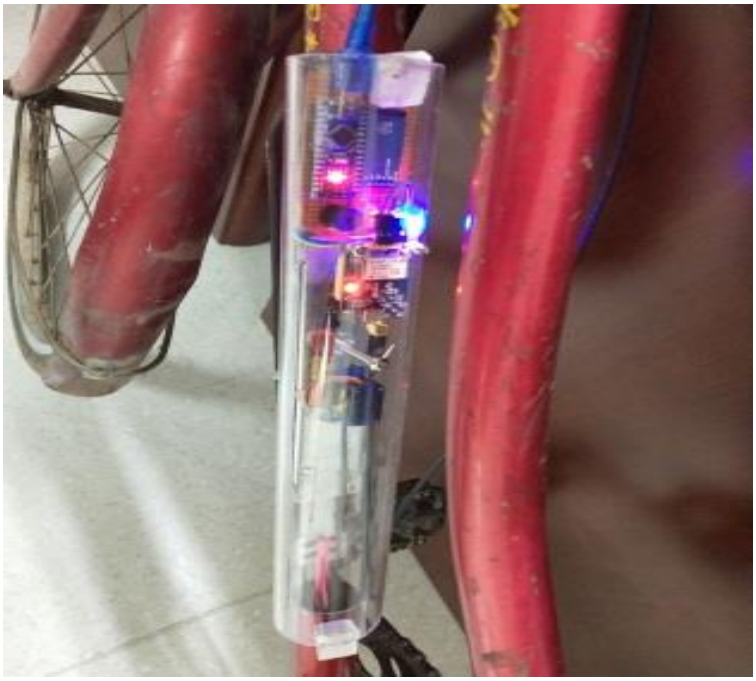

图 28 系统测试

5.1.2 系统测试过程及现象

（1）使用时用户打开 APP，手机蓝牙和 GPS 在后台自动打开。

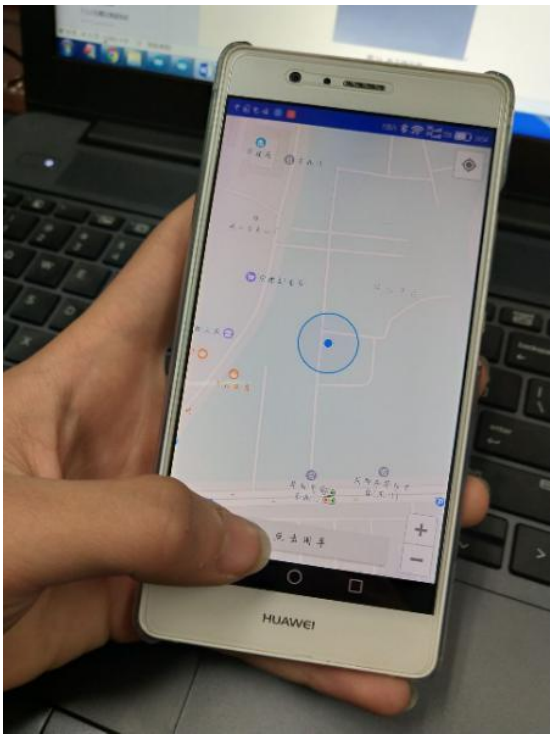

图 29 客户端操作演示

（2）进入 APP 界面后，用户可根据单车的位置显示，选择将要使用的单车。（蓝色圈为用户所在位置）

图 30 位置显示结果

（3）用户走到单车附近，点击“立即用车”。

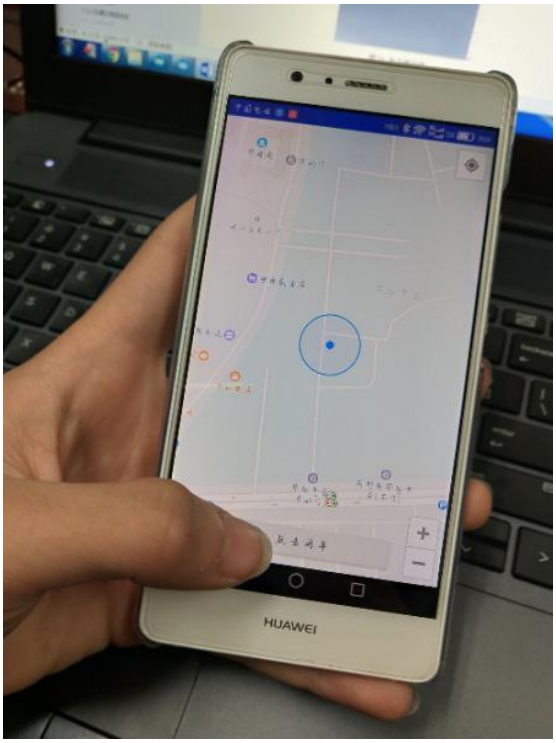

图 31 点击请求

（4）单车开锁。

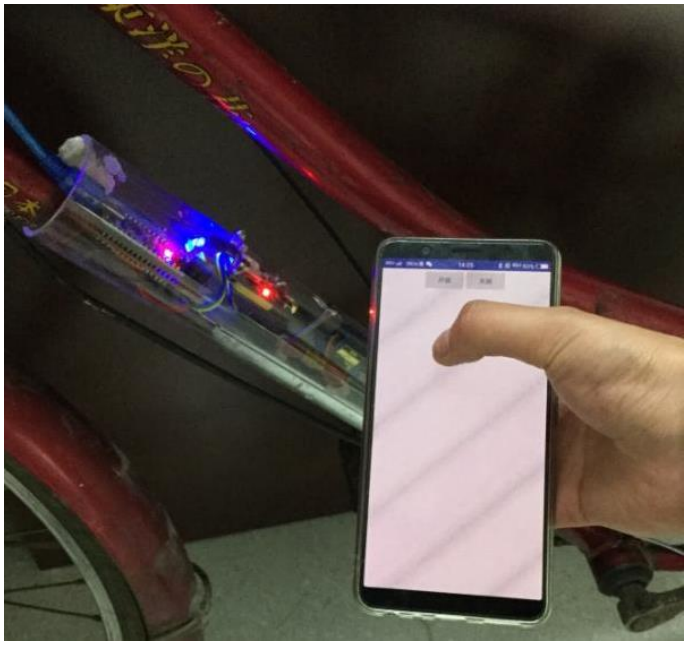

图 32 单车开锁

（5）用户在骑行前，收起脚架，单车后部锁解锁。

图 33 收起脚架后开锁

（6）用户暂停骑行或结束骑行，放下脚架，使后部锁关锁，插销触碰碰撞传感器，进而使单车前部锁关锁。

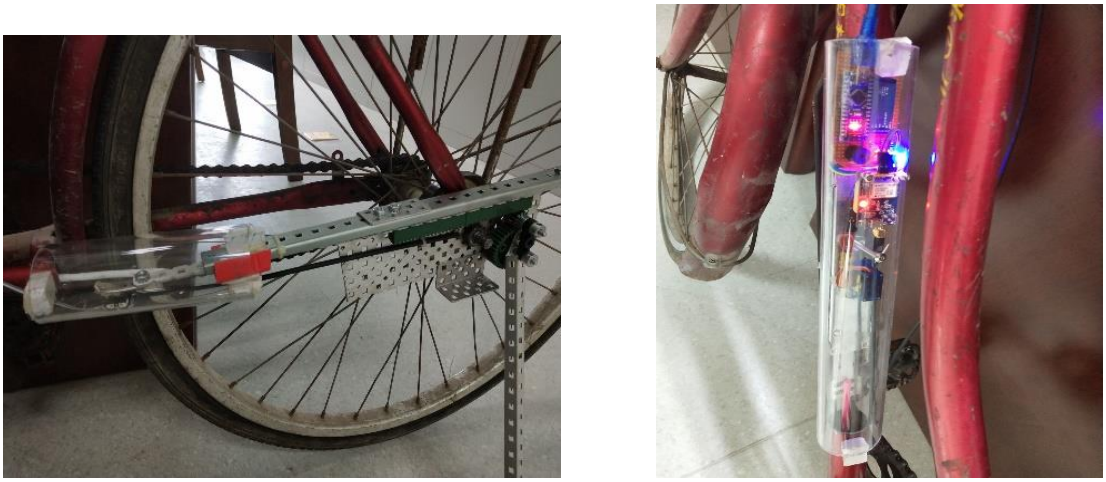

图 34 放下脚架后关锁

5.1.3 结果讨论

在 APP 界面点击将要使用的共享单车，走到单车电子锁附近，点击“立即用车”，单车开锁。暂停使用，在规定时间内点击“继续使用”，单车再次开锁，收放单车脚架完成后部的开锁与上锁。这种解锁、关锁方式，规避了二维码、车牌号带来的问题和用户停车后忘记关锁的现象，用户使用起来更加便捷、安全。

5.2 模块测试及结果

5.2.1 蓝牙扫描测试

用户选择单车后，手机蓝牙与电子锁内 HC-05 蓝牙（测试中我们选取手机蓝牙为对象进行扫描）建立连接，并读取其蓝牙 MAC 地址。

```
AT+BTSCAN=1,10

OK

+BTSCAN: 0,1,"IKON",38:6e:a2:69:40:93,-87

+BTSCAN: 0,2,"HUAWEI G9 Youth",20:a6:80:be:de:93,-91

+BTSCAN: 0,3,"OPPO",88:6a:b1:1c:eb:5f,-83

+BTSCAN: 0,4,"Xiaomi Redmi 3S",38:a4:ed:80:62:69,-89

+BTSCAN: 1
```

图 35 蓝牙扫描结果

结果：图中数据显示，本系统中蓝牙扫描精度达到了预期要求，能够准确地扫描出附近的蓝牙并读取其 MAC 地址。

5.2.2 GPS 定位测试

用户打开 APP 后，手机 GPS 开启，服务器对用户进行定位，并同时定位此用户附近 10m 单车。

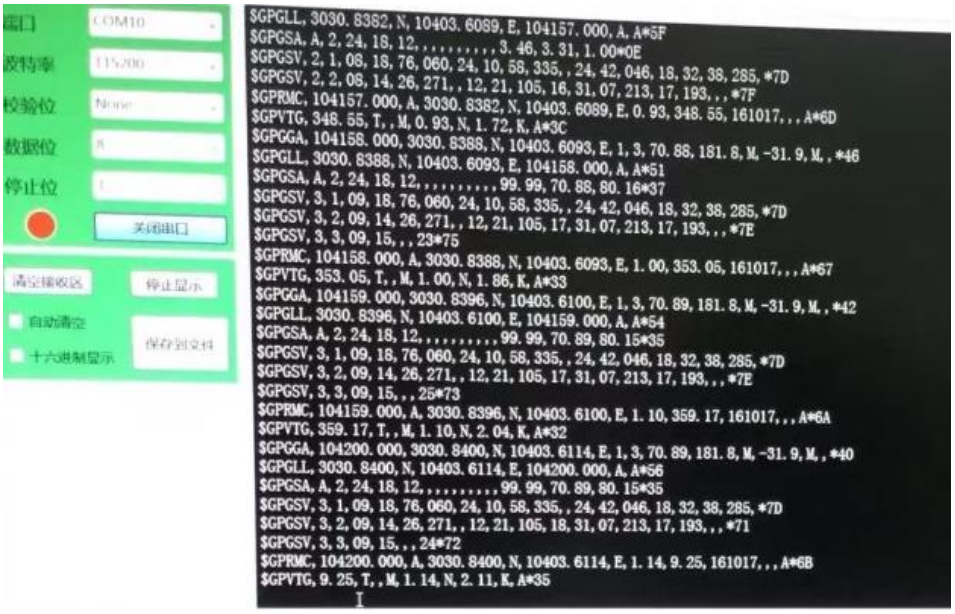

图 36 GPS 数据

结果：图中数据显示 GPS 定位数据稳定、准确，达到了预期目的。

5.2.3 手机陀螺仪测试

通过手机自带陀螺仪测定用户方向，将陀螺仪数据反馈至 APP，通过绘制地图来直观表示用户方位。

图 37 手机陀螺仪数据

结果：由手机陀螺仪数据可以看出用户端模块中的陀螺仪能够准确感知用户手机状态，帮助用户更容易寻找到单车位置。

5.2.4 开锁用时测试

在整个开锁过程中，为了测试本开锁方案的操作用时，我们在模拟开锁的同时进行计时，并且将计时结果同市面上现有的开锁方案的用时进行比较。

表 1 开锁用时测试

开锁时间/s / 开锁方案	测试 1	测试 2	测试 3	测试 4	测试 5	平均时间
摩拜一代	34.51	28.72	29.31	23.48	26.03	28.41
摩拜二代	13.19	8.82	9.79	12.28	7.32	10.28
ofo 一代	17.51	18.98	15.34	17.53	12.64	16.40
ofo 二代	16.32	14.71	18.54	16.19	13.55	15.86
青桔	5.81	3.08	4.32	5.99	6.08	5.06
本方案	5.42	6.44	5.12	6.07	6.02	5.81

结果：由收集到的数据分析可得，相对于现大多数共享单车开锁方案，本开锁方案用时更少，加快了开锁过程，能让用户有效快捷地开锁共享单车。

5.3 模拟实验数据分析

5.3.1 蓝牙扫描数据及其误差分析

我们使用不同的品牌和型号的手机，同时开启蓝牙扫描，并记录下蓝牙扫描的用时、范围等数据。测试时，我们依次使用十个不同的手机在同一点扫描其余九部手机的蓝牙，其余手机分布于以这部手机为顶点的射线上，每两部手机之间相距 1.5m（橙色圆点代表手机）。

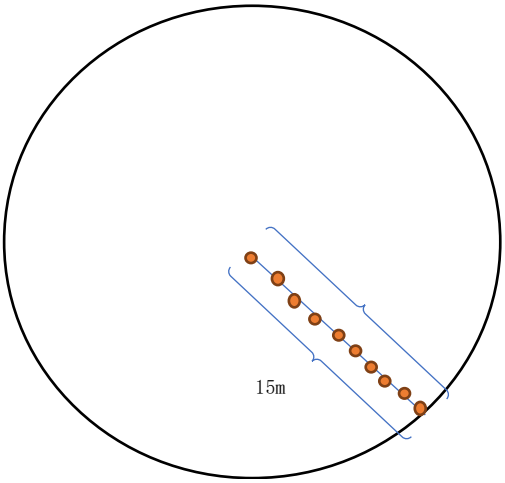

图 38 测试方案图

表 2 蓝牙扫描数据

测试类别 / 手机序号	扫描用时\s	扫描手机蓝牙数量\个
1	2.1	6
2	2.1	6
3	2.3	7
4	2.5	7
5	2.7	6
6	2.0	5
7	1.9	7
8	1.9	7
9	2.2	6
10	2.3	5

结果：经过多次测试，我们发现手机蓝牙扫描用时短，扫描范围在 10m 左右，为 APP 显示的单车位置范围提供了依据。

5.3.2 GPS 定位误差分析

为了测量 GPS 定位的误差范围，我们通过使用 GPS 卫星定位学校操场来进行模拟测试，学校操场的真实地理坐标为 104.0379556N, 30.3086606E.

表 3　GPS 测试数据

序号	经度	经度误差	纬度	纬度误差
000001	104.0379485,E	0.0000071	30.3086531,N	0.0000075
000002	104.0379494,E	0.0000062	30.3086545,N	0.0000061
000003	104.0379490,E	0.0000066	30.3086554,N	0.0000052
000004	104.0379489,E	0.0000067	30.3086580,N	0.0000026
000005	104.0379491,E	0.0000065	30.3086601,N	0.0000005
000006	104.0379488,E	0.0000063	30.3086606,N	0.0000000
000007	104.0379493,E	0.0000052	30.3086612,N	0.0000006
000008	104.0379504,E	0.0000039	30.3086621,N	0.0000015
000009	104.0379517,E	0.0000029	30.3086626,N	0.0000020
000010	104.0379527,E	0.0000024	30.3086620,N	0.0000014
000011	104.0379532,E	0.0000021	30.3086613,N	0.0000007
000012	104.0379535,E	0.0000016	30.3086609,N	0.0000003
000013	104.0379540,E	0.0000010	30.3086606,N	0.0000000
000014	104.0379546,E	0.0000003	30.3086609,N	0.0000003
000015	104.0379553,E	0.0000003	30.3086613,N	0.0000007
000016	104.0379559,E	0.0000010	30.3086618,N	0.0000012
000017	104.0379566,E	0.0000020	30.3086624,N	0.0000018
000018	104.0379576,E	0.0000020	30.3086629,N	0.0000023
000019	104.0379587,E	0.0000041	30.3086629,N	0.0000023
000020	104.0379597,E	0.0000041	30.3086629,N	0.0000023

结果：经过多次测量，我们发现 SIM808 的 GPS 定位误差范围控制在 10^{-6} 左右，既符合实际情况，又减少了用户与单车确立对应关系的错误率。

5.3.3 手机陀螺仪误差分析

为了测试手机陀螺仪的测量精度，我们首先在纸上画出三个角度，分别为 0 度、26.6 度、45 度，将纸固定在桌子上，且将手机旋转到对应的位置，进行多次实验（实验结果如表 4 所示）。

为了减小误差，我们决定使用手机输出的角速度和加速度的原始数据，用算法对其进行姿态融合，得到角度。综合利用陀螺仪和加速度计的特点，优势互补

获得准确的姿态角度，方法就是用卡尔曼滤波做数据融合。大致的思路是将模块的姿态用四元素表示，作为系统的状态量，模块的姿态运动学方程作为滤波的状态转移方程，加速度信息作为滤波的观察量信息，然后利用卡尔曼滤波的计算方法迭代计算更新 [4]。

表 4 陀螺仪测试数据

	位置 0（0 度）			位置 1（26.6 度）			位置 2（45 度）		
	tx	ty	tz	tx	ty	tz	tx	ty	tz
循环 1 次	-1.4	-2.89	0	-1.33	-3.03	28.8	-1.3	-3.13	60.56
循环 2 次	-1.49	-2.85	3.43	-1.45	-2.95	30.59	-1.34	-3.1	67.77
循环 3 次	-1.45	-2.90	5.51	-1.45	-2.93	31.79	-1.47	-2.98	70.54
循环 1 次	-1.5	-3.02	0	-1.47	-3.11	25.29	-1.4	-3.21	77.78
循环 2 次	-1.54	-2.99	0.52	-1.47	-3.04	24.78	-1.39	-3.17	56.89
循环 3 次	-1.44	-2.92	10	-1.46	-3.05	26.07	-1.4	-3.16	69.34
循环 1 次	-1.58	-3.04	0	-1.47	-3.12	30.45	-1.43	-3.19	63.66
循环 2 次	-1.57	-2.99	6.51	-1.48	-3.08	32.76	-1.42	-3.17	70.65
循环 3 次	-1.54	-2.92	7.64	-1.47	-3.08	32.79	-1.42	-3.19	71.54

结果：从数据可以看出，陀螺仪在精度上已经达到了预期要求，能够准确地反映用户手机所在的状态。

6 结果与讨论

（1）本系统使用 GPS 定位用户地理位置，并使用蓝牙进行用户与单车之间的通信以及身份确认，此方案在实际操作过程中可以有效的节省用户开锁时间，能让用户不借助二维码、车牌号就能有效、安全地解锁共享单车。

（2）运用齿轮传动原理，改进单车内部结构，通过收放脚架巧妙地完成电子锁的开关，使用户使用起来更加便捷、放心。

（3）将电子锁隐藏单车的车架中，通过舵机转动插销锁完成电子锁的开关，避免了小偷与共享单车电子锁的直接接触，增强了单车的防盗能力。

（4）本系统成本测算大约为 200 元左右，不仅廉价且实用。

7 后续研究的设想

7.1 装置简化

我们将在后期简化硬件设计部分，深化软件设计部分，强化后台处理数据的能力，从而给用户带来最佳用车体验。

7.2 双向加密认证通信机制

为了使用户的用车体验有更大的提升，我们将使用双向加密认证通信机制，使用户、服务器、单车三者之间的通信以及身份确认的安全性和快捷性在现有基础上有更大提升。

7.3 时间+路程综合收费

为了使共享单车的计费方式更加科学，在后期我们将会采用一种时间+路程综合收费机制，避免了传统的纯时间收费机制对一些使用共享单车时间较短的用户带来高额的收费现象。

8 致谢

我们的作品一共有三代，每一代都是我们自己设计，亲手制作出来的，我们需要学习大量的硬件搭建、软件编程的知识，这花费了我们大量的时间，期间我们也仿徨过。但指导老师的热心关怀和帮助、学校给予的研究资金支持，让我们坚持到现在并坚定继续研究下去。在我们的作品提交参赛之前，我们诚挚地在此对在本课题研究中给予我们帮助的专家教授们、老师们、学长们和家长同学们表示衷心的感谢。

最后对各位对本课题研究进行评审以及提出自己宝贵意见的教授专家们表示衷心的感谢！

附录

1 电子锁程序

```
#include <Servo.h>
#include <SoftwareSerial.h>

#define GpsSerial Serial//RX,TX GPS 串口
#define DebugSerial Serial//RX,TX  调试串口
```

```
Servo lock;     //舵机
SoftwareSerial BT(6, 7); //RX,TX  蓝牙

int BL = 3;
int RL = 4;
int GL = 5;
int BEEP = 8;
int CS = 9; //Crash Sensor,碰撞传感器
int senserState = 2;

int val = -1;
int pos = -180;
int L = 13; //LED 指示灯引脚
int frequency = 500; //beep 的频率

struct
{
   char GPS_Buffer[80];
   bool isGetData;       //是否获取到 GPS 数据
   bool isParseData; //是否解析完成
   char UTCTime[11];     //UTC 时间
   char latitude[11];       //纬度
   char N_S[2];       //N/S
   char longitude[12];     //经度
   char E_W[2];       //E/W
   bool isUsefull;     //定位信息是否有效
}
Save_Data;

const unsigned int gpsRxBufferLength = 600;
char gpsRxBuffer[gpsRxBufferLength];
```

```
unsigned int ii = 0;

void setup() {
  pinMode(BL, OUTPUT);
  pinMode(RL, OUTPUT);
  pinMode(GL, OUTPUT);
  pinMode(CS, INPUT);
  lock.attach(2);

  BT.begin(38400);
  // HC-05 默认，38400
  Serial.println("BT is ready!");
  digitalWrite(RL, LOW);
  GpsSerial.begin(9600);
  //定义波特率 9600，和 GPS 模块输出的波特率一致
  //   DebugSerial.begin(9600);
  DebugSerial.println("BT is ready!");
  DebugSerial.println("GPS Initializing");
  DebugSerial.println("Wating...");

  Save_Data.isGetData = false;
  Save_Data.isParseData = false;
  Save_Data.isUsefull = false;
}

void loop()
{
  gpsRead();  //获取 GPS 数据
  parseGpsBuffer();//解析 GPS 数据
  printGpsBuffer();//输出解析后的数据
  // DebugSerial.println("\r\n\r\nloop\r\n\r\n");

  if (BT.available() && GpsSerial.available())
```

```
{
  digitalWrite(BL, HIGH);
}
while (BT.available())
{
  val = BT.read();
  Serial.println(val);
  if (val == 0)
  {
    lock.write(pos);              // 让舵机转到变量 pos 所指的位置
    //delay(1000);
    digitalWrite(GL, HIGH);
    digitalWrite(RL, LOW);
    tone(BEEP, frequency);
    delay(800);
    noTone(BEEP);
    Serial.println("unlock");
  }
  while (digitalRead(CS) > 0)
  {
    if (senserState == LOW)
    {
      lock.write(-pos);
    }
  }
  if (val == 248)
  {
    lock.write(-pos);                  // 让舵机转到变量 pos 所指的位置
    //delay(1000);
    digitalWrite(GL, LOW);
    digitalWrite(RL, HIGH);
    Serial.println("locked");
  }
```

```
    }
}


void errorLog(int num)
{
    DebugSerial.print("ERROR");
    DebugSerial.println(num);
    while (1)
    {
        digitalWrite(L, HIGH);
        delay(300);
        digitalWrite(L, LOW);
        delay(300);
    }
}

void printGpsBuffer()
{
    if (Save_Data.isParseData)
    {
        Save_Data.isParseData = false;

        DebugSerial.print("Save_Data.UTCTime = ");
        DebugSerial.println(Save_Data.UTCTime);

        if (Save_Data.isUsefull)
        {
            Save_Data.isUsefull = false;
            DebugSerial.print("Save_Data.latitude = ");
            DebugSerial.println(Save_Data.latitude);
            DebugSerial.print("Save_Data.N_S = ");
            DebugSerial.println(Save_Data.N_S);
            DebugSerial.print("Save_Data.longitude = ");
```

```
            DebugSerial.println(Save_Data.longitude);
            DebugSerial.print("Save_Data.E_W = ");
            DebugSerial.println(Save_Data.E_W);
        }
        else
        {
            DebugSerial.println("GPS DATA is not usefull!");
        }

    }
}

void parseGpsBuffer()
{
    char *subString;
    char *subStringNext;
    if (Save_Data.isGetData)
    {
        Save_Data.isGetData = false;
        DebugSerial.println("***************");
        DebugSerial.println(Save_Data.GPS_Buffer);


        for (int i = 0 ; i <= 6 ; i++)
        {
            if (i == 0)
            {
                if ((subString = strstr(Save_Data.GPS_Buffer, ",")) == NULL)
                    errorLog(1);    //解析错误
            }
            else
            {
                subString++;
```

```
            if ((subStringNext = strstr(subString, ",")) != NULL)
            {
                char usefullBuffer[2];
                switch (i)
                {
                    case 1:
                        memcpy(Save_Data.UTCTime, subString, subStringNext -
subString);
                        break;   //获取 UTC 时间
                    case 2:
                        memcpy(usefullBuffer, subString, subStringNext - subString);
                        break;   //获取 UTC 时间
                    case 3:
                        memcpy(Save_Data.latitude, subString, subStringNext - subString);
                        break;   //获取纬度信息
                    case 4:
                        memcpy(Save_Data.N_S, subString, subStringNext - subString);
                        break;   //获取 N/S
                    case 5:
                        memcpy(Save_Data.longitude, subString, subStringNext -
subString);
                        break;   //获取纬度信息
                    case 6:
                        memcpy(Save_Data.E_W, subString, subStringNext - subString);
                        break;   //获取 E/W

                    default:
                        break;
                }

                subString = subStringNext;
                Save_Data.isParseData = true;
                if (usefullBuffer[0] == 'A')
```

```
                Save_Data.isUsefull = true;
            else if (usefullBuffer[0] == 'V')
                Save_Data.isUsefull = false;

        }
        else
        {
            errorLog(2);   //解析错误
        }
      }


    }
  }
}


void gpsRead()
{
  while (GpsSerial.available())
  {
    gpsRxBuffer[ii++] = GpsSerial.read();
    if (ii == gpsRxBufferLength)clrGpsRxBuffer();
  }

  char* GPS_BufferHead;
  char* GPS_BufferTail;
  if ((GPS_BufferHead = strstr(gpsRxBuffer, "$GPRMC,")) != NULL || (GPS_BufferHead = strstr(gpsRxBuffer, "$GNRMC,")) != NULL )
  {
    if (((GPS_BufferTail = strstr(GPS_BufferHead, "\r\n")) != NULL) && (GPS_BufferTail > GPS_BufferHead))
    {
```

```
            memcpy(Save_Data.GPS_Buffer,    GPS_BufferHead,    GPS_BufferTail    -
GPS_BufferHead);
            Save_Data.isGetData = true;

            clrGpsRxBuffer();
        }
    }
}

void clrGpsRxBuffer(void)
{
    memset(gpsRxBuffer, 0, gpsRxBufferLength);          //清空
    ii = 0;
}
```

2 客户端程序

***注.因 Android 软件结构较为复杂，在此处只列出活动（Activity）的 Java 程序源码，其余部分如布局（layout）等在此不予列出。**

2.1 BikeDemo

2.1.1 MainActivity.Java

```
package com.example.jamie.bikedemo;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.graphics.Point;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.amap.api.location.AMapLocation;
import com.amap.api.location.AMapLocationClient;
import com.amap.api.location.AMapLocationClientOption;
import com.amap.api.location.AMapLocationListener;
import com.amap.api.maps.AMap;
import com.amap.api.maps.CameraUpdateFactory;
import com.amap.api.maps.LocationSource;
import com.amap.api.maps.MapView;
```

```
import com.amap.api.maps.Projection;
import com.amap.api.maps.model.BitmapDescriptorFactory;
import com.amap.api.maps.model.LatLng;
import com.amap.api.maps.model.Marker;
import com.amap.api.maps.model.MarkerOptions;
import com.amap.api.maps.model.MyLocationStyle;

public class MainActivity extends Activity implements LocationSource,
        AMapLocationListener {
    private AMap aMap;//实例化 AMap 对象
    private MapView mapView;
    private OnLocationChangedListener mListener;
    private AMapLocationClient mlocationClient;
    private AMapLocationClientOption mLocationOption;

    private static final int STROKE_COLOR = Color.argb(180, 3, 145, 255);
    private static final int FILL_COLOR = Color.argb(10, 0, 0, 180);

    //自定义定位小蓝点的 Marker
    //Marker locationMarker;

    //坐标和经纬度转换工具
    //Projection projection;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);//  不显示程序的标题栏
        setContentView(R.layout.activity_main);
```

```
        mapView = (MapView) findViewById(R.id.map);
        mapView.onCreate(savedInstanceState);// 此方法必须重写
        init();
    }
    /**
     * 初始化
     */
    private void init() {
        if (aMap == null) {
            aMap = mapView.getMap();
            setUpMap();
        }
    }

    /**
     * 设置一些 amap 的属性
     */
    private void setUpMap() {
        aMap.setLocationSource(this);// 设置定位监听
        aMap.getUiSettings().setMyLocationButtonEnabled(true);// 设置默认定位按钮是否显
示
        aMap.setMyLocationEnabled(true);// 设置为 true 表示显示定位层并可触发定位，
false 表示隐藏定位层并不可触发定位，默认是 false
        setupLocationStyle();
    }
    /**
     * 设置自定义定位蓝点
     */
    private void setupLocationStyle(){
        // 自定义系统定位蓝点
```

```
        MyLocationStyle myLocationStyle = new MyLocationStyle();
        // 自定义定位蓝点图标
        myLocationStyle.myLocationIcon(BitmapDescriptorFactory.
                fromResource(R.drawable.gps_point));
        // 自定义精度范围的圆形边框颜色
        myLocationStyle.strokeColor(STROKE_COLOR);
        //自定义精度范围的圆形边框宽度
        myLocationStyle.strokeWidth(5);
        // 设置圆形的填充颜色
        myLocationStyle.radiusFillColor(FILL_COLOR);
        // 将自定义的 myLocationStyle 对象添加到地图上
        aMap.setMyLocationStyle(myLocationStyle);
    }

    @Override
    protected void onResume() {
        super.onResume();
        //在 activity 执行 onResume 时执行 mMapView.onResume ()，重新绘制加载地图
        mapView.onResume();
    }
    @Override
    protected void onPause() {
        super.onPause();
        //在 activity 执行 onPause 时执行 mMapView.onPause ()，暂停地图的绘制
        mapView.onPause();

    }
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
```

```
        //在 activity 执行 onSaveInstanceState 时执行 mMapView.onSaveInstanceState
(outState)，保存地图当前的状态
        mapView.onSaveInstanceState(outState);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        mapView.onDestroy();
        if(null != mlocationClient){
            mlocationClient.onDestroy();
        }
    }

    /**
     * 定位成功后回调函数
     */
    @Override
    public void onLocationChanged(AMapLocation amapLocation) {
        if (mListener != null && amapLocation != null) {
            if (amapLocation != null
                    && amapLocation.getErrorCode() == 0) {
                mListener.onLocationChanged(amapLocation);// 显示系统小蓝点
                aMap.moveCamera(CameraUpdateFactory.zoomTo(18));
            } else {
                String errText = "定位失败," + amapLocation.getErrorCode()+ ": " +
amapLocation.getErrorInfo();
                Log.e("AmapErr",errText);
            }
        }
```

```
    }
    /**
     * 激活定位
     */
    @Override
    public void activate(OnLocationChangedListener listener) {
        mListener = listener;
        if (mlocationClient == null) {
            mlocationClient = new AMapLocationClient(this);
            mLocationOption = new AMapLocationClientOption();
            //设置定位监听
            mlocationClient.setLocationListener(this);
            //设置为高精度定位模式

mLocationOption.setLocationMode(AMapLocationClientOption.AMapLocationMode.Hight_Acc
uracy);
            //是指定位间隔
            mLocationOption.setInterval(2000);
            //设置定位参数
            mlocationClient.setLocationOption(mLocationOption);
            // 此方法为每隔固定时间会发起一次定位请求，为了减少电量消耗或网络流
量消耗，
            // 注意设置合适的定位时间的间隔（最小间隔支持为 2000ms），并且在合适时
间调用 stopLocation()方法来取消定位请求
            // 在定位结束后，在合适的生命周期调用 onDestroy()方法
            // 在单次定位情况下，定位无论成功与否，都无需调用 stopLocation()方法移
除请求，定位 sdk 内部会移除
            mlocationClient.startLocation();
        }
    }
```

```
/**
 * 停止定位
 */
@Override
public void deactivate() {
    mListener = null;
    if (mlocationClient != null) {
        mlocationClient.stopLocation();
        mlocationClient.onDestroy();
    }
    mlocationClient = null;
}

String packname = "com.example.hefugui.blue_control";
public void BTactivity(View view) {
    PackageManager packageManager = getPackageManager();
    if (checkPackInfo(packname)) {
        Intent intent = packageManager.getLaunchIntentForPackage(packname);
        startActivity(intent);
    } else {
        Toast.makeText(MainActivity.this, "没有安装" + packname, 1).show();
    }


}

private boolean checkPackInfo(String packname) {
    PackageInfo packageInfo = null;
    try {
```

```
            packageInfo = getPackageManager().getPackageInfo(packname, 0);
        } catch (PackageManager.NameNotFoundException e) {
            e.printStackTrace();
        }
        return packageInfo != null;
    }


}
```

2.2 Blue_Control

2.2.1 MainActivity.Java

```
package com.example.hefugui.blue_control;

import android.bluetooth.BluetoothAdapter;
        import android.bluetooth.BluetoothDevice;
        import android.content.BroadcastReceiver;
        import android.content.Context;
        import android.content.Intent;
        import android.content.IntentFilter;
        import android.content.SharedPreferences;
        import android.support.v7.app.AppCompatActivity;
        import android.os.Bundle;
        import android.util.Log;
```

```
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;

public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemClickListener
{
    private static final String TAG = "Main";
    private ListView lvDevices;
    Button b1;
    private BluetoothAdapter bluetoothAdapter;
    private List<String> bluetoothDevices = new ArrayList<String>();
    private ArrayAdapter<String> arrayAdapter;
    private BluetoothDevice device;
    private SharedPreferences sp;
    public ArrayList<String> list =new ArrayList<String>();
    Set<BluetoothDevice> bondDevices ;
    public static final String PROTOCOL_SCHEME_L2CAP = "btl2cap";
    public static final String PROTOCOL_SCHEME_RFCOMM = "btspp";
    public static final String PROTOCOL_SCHEME_BT_OBEX = "btgoep";
    public static final String PROTOCOL_SCHEME_TCP_OBEX = "tcpobex";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

lvDevices = (ListView) findViewById(R.id.list_search);

sp = getSharedPreferences("config", MODE_PRIVATE);

Set<BluetoothDevice> pairedDevices = bluetoothAdapter
        .getBondedDevices();

if (pairedDevices.size() > 0)
{
    for (BluetoothDevice device : pairedDevices)
    {
        bluetoothDevices.add(device.getName() + ":"
                + device.getAddress() + "\n");
    }
}
arrayAdapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, android.R.id.text1,
        bluetoothDevices);
b1 = (Button) findViewById(R.id.button1);
b1.setOnClickListener(new View.OnClickListener()
{

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        setProgressBarIndeterminateVisibility(true);
```

```
                setTitle("正在扫描...");

                if(bluetoothAdapter.isDiscovering()){
                    bluetoothAdapter.cancelDiscovery();
                }
                list.clear();
                bondDevices = bluetoothAdapter.getBondedDevices();

                for(BluetoothDevice device : bondDevices) {
                    String str = "   已配对完成   " + device.getName() +" "
                            + device.getAddress();
                    list.add(str);
                    arrayAdapter.notifyDataSetChanged();
                }
                bluetoothAdapter.startDiscovery();


            }
        });
        lvDevices.setAdapter(arrayAdapter);
        lvDevices.setOnItemClickListener(this);
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(receiver, filter);

        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(receiver, filter);
    }
    public void onClick_Search(View view)
    {
        setProgressBarIndeterminateVisibility(true);
```

```
        setTitle("正在扫描...");

        if (bluetoothAdapter.isDiscovering())
        {
            bluetoothAdapter.cancelDiscovery();
        }
        bluetoothAdapter.startDiscovery();
}

@Override
public void onItemClick(AdapterView<?> parent, View view, int position,long id)
{
        String s = arrayAdapter.getItem(position);
        String address = s.substring(s.indexOf(":") + 1).trim();

        Log.v(TAG, "---------->"+address);
        Toast.makeText(MainActivity.this, address, 1).show();
        SharedPreferences.Editor editor = sp.edit();
        editor.putString("address", address);
        editor.commit();
        Intent intent = new Intent(MainActivity.this,Lanyakongzhi.class);
        startActivity(intent);
        finish();

}

private final BroadcastReceiver receiver = new BroadcastReceiver()
{
        @Override
```

```
        public void onReceive(Context context, Intent intent)
        {
            String action = intent.getAction();

            if (BluetoothDevice.ACTION_FOUND.equals(action))
            {

                BluetoothDevice device = intent
                        .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

                if (device.getBondState() != BluetoothDevice.BOND_BONDED)
                {
                    bluetoothDevices.add(device.getName() + ":"
                            + device.getAddress() + "\n");
                    arrayAdapter.notifyDataSetChanged();
                }

            }
            else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action))
            {
                setProgressBarIndeterminateVisibility(false);
                setTitle("连接蓝牙设备");

            }
        }
    };


}
```

2.2.2 Lanyakongzhi.java

```
package com.example.hefugui.blue_control;

import android.app.Activity;

import android.graphics.pdf.PdfDocument;

import android.os.Bundle;

import java.io.IOException;

import java.io.OutputStream;

import java.util.UUID;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.content.Intent;

import android.content.SharedPreferences;

import android.text.TextUtils;

import android.util.Log;

import android.view.MotionEvent;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import java.io.*;

import java.util.*;


public class Lanyakongzhi extends Activity {

    private static final String TAG = "BLUEZ_CAR";

    private static final boolean D = true;

    private BluetoothAdapter mBluetoothAdapter = null;

    private BluetoothSocket btSocket = null;

    private OutputStream outStream = null;

    private SharedPreferences sp;
```

```
    private String address;



    Button mButtonL;
    Button mButtonR;

    private static final UUID MY_UUID =
UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
//      Intent intent = new Intent();
////      String result = intent.getStringExtra("textViewLabel");
////      String address = result;
    // private static String address = "result" ; // <==要连接的蓝牙设备 MAC 地址
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.kongzhi);


        sp = getSharedPreferences("config", MODE_PRIVATE);
        address =   sp.getString("address", null);
        if(TextUtils.isEmpty(address)){
            Toast.makeText(Lanyakongzhi.this, "蓝牙地址为空", 1).show();
        }
//      //获取蓝牙数据
//      Intent intent = new Intent();
//      address = intent.getStringExtra("BTname");
//      Toast.makeText(Lanyakongzhi.this, address, Toast.LENGTH_SHORT).show();


        //开锁
```

```
mButtonL=(Button)findViewById(R.id.btnL);
mButtonL.setOnClickListener(new Button.OnClickListener() {


    @Override
    public void onClick(View v) {

        try {
            outStream = btSocket.getOutputStream();
        } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Output stream creation failed.", e);
        }

        int msgBuffer = 0;
        try {
            outStream.write(msgBuffer);
        } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Output stream creation failed.", e);
        }
    }
});


//关锁
mButtonR=(Button)findViewById(R.id.btnR);
mButtonR.setOnClickListener(new Button.OnClickListener() {

    @Override
    public void onClick(View v) {
```

```
                try {
                    outStream = btSocket.getOutputStream();
                } catch (IOException e) {
                    Log.e(TAG, "ON RESUME: Output stream creation failed.", e);
                }
                Byte msgBuffer = -1;
                try {
                    outStream.write(msgBuffer);
                } catch (IOException e) {
                    Log.e(TAG, "ON RESUME: Output stream creation failed.", e);
                }
            }
        });

        if (D)
            Log.e(TAG, "+++ ON CREATE +++");
        mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        if (mBluetoothAdapter == null) {
            Toast.makeText(this, "Bluetooth is not available.",
Toast.LENGTH_LONG).show();
            finish();
            return;
        }
        if (!mBluetoothAdapter.isEnabled()) {
            Toast.makeText(this,  "Please enable your Bluetooth and re-run this program.",
Toast.LENGTH_LONG).show();
            finish();
            return;
        }
```

```
        if (D)
            Log.e(TAG, "+++ DONE IN ON CREATE, GOT LOCAL BT ADAPTER +++");
    }
    @Override
    public void onStart() {
        super.onStart();
        if (D) Log.e(TAG, "++ ON START ++");
    }
    @Override
    public void onResume() {
        super.onResume();
        if (D) {
            Log.e(TAG, "+ ON RESUME +");
            Log.e(TAG, "+ ABOUT TO ATTEMPT CLIENT CONNECT +");

        }
        BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
        try {
            btSocket = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Socket creation failed.", e);
        }
        mBluetoothAdapter.cancelDiscovery();
        try {
            btSocket.connect();
            Log.e(TAG, "ON RESUME: BT connection established, data transfer link open.");
        } catch (IOException e) {
            try {
                btSocket.close();
```

```
            } catch (IOException e2) {

                Log .e(TAG,"ON RESUME: Unable to close socket during connection failure",
e2);
            }
        }
        // Create a data stream so we can talk to server.
        if (D)
            Log.e(TAG, "+ ABOUT TO SAY SOMETHING TO SERVER +");
        try {
            outStream = btSocket.getOutputStream();
        } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Output stream creation failed.", e);
        }
        String message = "1";
        byte[] msgBuffer = message.getBytes();
        try {
            outStream.write(msgBuffer);
        } catch (IOException e) {
            Log.e(TAG, "ON RESUME: Exception during write.", e);
        }
    }
    @Override
    public void onPause() {
        super.onPause();
        if (D)
            Log.e(TAG, "- ON PAUSE -");
        if (outStream != null) {
            try {
                outStream.flush();
```

```
                } catch (IOException e) {
                    Log.e(TAG, "ON PAUSE: Couldn't flush output stream.", e);
                }
            }
            try {
                btSocket.close();
            } catch (IOException e2) {
                Log.e(TAG, "ON PAUSE: Unable to close socket.", e2);
            }
        }
        @Override
        public void onStop() {
            super.onStop();
            if (D)Log.e(TAG, "-- ON STOP --");
        }
        @Override
        public void onDestroy() {
            super.onDestroy();
            if (D) Log.e(TAG, "--- ON DESTROY ---");

        }

}
```

参考文献：

[1]刘少军，王瑜瑜. 手机蓝牙技术在智能电子锁控制系统中的研究[J]. 机械与电子, 2016 年第 4 期. 2016-04-24

[2]郑拴宁. 一种共享单车二维码配对管理系统 [P]. 电声技术，第 11 期 67-70 页,共 4 页. 2017-06-23

[3] 任元华. 共享单车自动开锁系统 [P]. 中国专利:CN107203926A,2017-09-26

[4] 晏勇,雷航,周相兵,梁潘. [J]. 基于三轴加速度传感器的自适应计步器的实现东北师大学报(自然科学版). 2016-03-15