中南大學
CENTRAL SOUTH UNIVERSITY

定位技术课程设计

实 验 报 告

指导老师：李刚

专　　业：物联网工程

班　　级：物联网 1802

学　　号：8213180228

姓　　名：王云鹏

目录

一、前言介绍

图 1 2021 苹果春季新品发布会中出现的 AirTags 追踪标签

目前，对于没有 WiFi 通信功能的设备（如手写笔、低功耗传感器、智能移动设备、部分智能家居产品等）而言，由于其缺少 WiFi 通信模块，所以无法利用 WiFi 的 RSSI 对其进行定位。而通信系统是物联网设备的必要组成部分，相对于 WiFi 通信，蓝牙通信模块更容易部署在各种物联网设备上。所以，相对于利用 WiFi 信号强度定位而言，利用蓝牙信号强度可以使被定位设备的种类增多。可以通过利用蓝牙信号强度，根据三边定位等方法也可以实现定位。

现如今，许多互联网和智能制造大厂也开始利用蓝牙信号对设备或物体进行定位。图 1 是 2021 苹果春季新品发布会中出现的 AirTags 追踪标签，该标签可以附着在某些设备甚至是小物体上，AirTags 本

身是低功耗蓝牙信号发生器，于是苹果终端的 App 可以通过搜索其信号及 MAC 地址，根据蓝牙信号强度，为 AirTags 定位。

于是本次物联网定位课程设计的应用场景为：在部署了蓝牙信号基站的区域里，通过对信号强度和 MAC 地址的收集和分析，通过三边定位法，对蓝牙设备进行定位。从而对区域里的物联网设备（如低功耗传感器、智能家居设备和智能移动设备）进行室内或室外定位。

二、实验目的

1）回顾物联网定位技术课程的讲授内容，深化对于物联网定位相关技术和理论的理解和掌握；

2）在回顾所学知识的基础上，综合利用物联网定位技术、网络应用开发技术和嵌入式开发技术，开发和实现基于物联网定位的位置服务应用（LBS)；

3）在应用开发过程中提高对物联网定位技术的掌握和综合解决问题的能力。

知识：

掌握物联网定位的基本概念，基本的测距技术原理和实现方法，常用的位置计算方法和策略，理解全球卫星定位系统的基本原理和组成，移动通信中的定位技术，室内无线定位技术的基本原理，基于 RFID 的定位技术，无线传感器网络定位技术，基于位置的服务概念及应用

等知识。学会从应用需求出发选择合适的定位技术，建立实现定位系统的基本思维方式，从理论和实验上分析定位系统的性能，形成定位原理-定位技术及实现-定位系统应用的基本知识结构。

能力：

从应用的角度出发掌握如何选择合适的定位技术，具备将具体定位技术进行实现并应用于具体工程的能力，培养基于移动设备开发实现定位系统的能力，能够对所开发的定位系统的性能从进行分析并在真实环境中进行测评优化；掌握基本的定位知识，并具备针对具体问题提出新的定位技术和方法的能力；在与物联网应用和基于位置服务的交叉知识的讨论中培养创新意识，提高分析、发现、研究和解决问题的能力。

素质：

通过课程中的分析、讨论、辩论培养分析沟通交流的素质，建立应用驱动的技术研发与实现的问题解决方式和思维模式，提升理解工程管理实现和技术决策的基本素质。通过课外导学的模式，提升自主学习和终身学习的意识，形成通过不断学习来提升、发展自身素质以适应经济、社会发展的能力。

三、背景知识

3.1 蓝牙与低功耗蓝牙

从蓝牙 4.0 开始，蓝牙就包括两个标准：传统蓝牙部分（Classic Bluetooth）和低功耗蓝牙部分（Bluetooth Low Energy）这两个部分适用于不同的应用或者应用条件。传统蓝牙是在之前的 1.0.1.2，2.0+EDR,2.1+EDR,3.0+EDR 等基础上发展和完善起来的，低功耗蓝牙是 Nokia 的 Wibree 标准上发展起来的。

传统蓝牙可以用与数据量比较大的传输，如语音，音乐，较高数据量传输等，低功耗蓝牙这样应用于实时性要求比较高，但是数据速率比较低的产品，如遥控类的，如鼠标，键盘，遥控鼠标(Air Mouse)，传感设备的数据发送，如心跳带，血压计，温度传感器等。

传统蓝牙有 3 个功率级别，Class1,Class2,Class3,分别支持 100m,10m,1m 的传输距离，而低功耗蓝牙无功率级别，一般发送功率在 7dBm，一般在空旷距离，达到 20m 应该是没有问题的。

所以蓝牙 4.0 是集成了传统蓝牙和低功耗蓝牙两个标准的，并不只是低功耗蓝牙。

关于低功耗蓝牙，BLE 是蓝牙低能耗的简称（Bluetooh Low Energy）。蓝牙低能耗(BLE)技术是低成本、短距离、可互操作的鲁棒性无线技术，工作在免许可的 2.4GHz ISM 射频频段。它从一开始就设计为超低功耗(ULP)无线技术。它利用许多智能手段最大限度地降低功耗。蓝牙低能耗架构共有两种芯片构成：单模芯片和双模芯片。

蓝牙单模芯片可以和其它单模芯片及双模芯片通信，此时后者需要使用自身架构中的蓝牙低能耗技术部分进行收发数据。双模芯片也能与标准蓝牙技术及使用传统蓝牙架构的其它双模芯片通信。双模芯片可以在目前使用标准蓝牙芯片的任何场合使用。这样安装有双模芯片的手机、PC、个人导航设备(PND)或其它应用就可以和市场上已经在用的所有传统标准蓝牙设备以及所有未来的蓝牙低能耗设备通信。然而，由于这些设备要求执行标准蓝牙和蓝牙低能耗任务，因此双模芯片针对 ULP 操作的优化程度没有像单模芯片那么高。单模芯片可以用单节钮扣电池(如 3V、220mAh 的 CR2032)工作很长时间(几个月甚至几年)。相反，标准蓝牙技术(和蓝牙低能耗双模器件)通常要求使用至少两节 AAA 电池(电量是钮扣电池的 10 至 12 倍，可以容忍高得多的峰值电流)，并且更多情况下最多只能工作几天或几周的时间(取决于具体应用)。

3.2 自由空间传播模型

自由空间传播模型（Free space propagation Model）用于预测接收机和发射机之间是完全无阻挡的视距路径时接收信号的场强，属于大尺度路径损耗的无线电波传播的模型。自由空间模型预测接收功率的衰减为发射机与接收机之间（T-R）距离的函数。

理想的无线传播条件是不存在的，一般认为只要地面上空的大气层是各向同性的均匀媒质，其相对介电常数和相对导磁率都等于 1，传播路径上没有障碍物阻挡，到达接收天线的地面反射信号场强也可以忽略不计，在这样的情况下，电波的传播方式就被认为是在自由空

间传播。信号能量在自由空间传播了一定距离后，信号能量也会发生衰减。通常卫星通信系统和微波视距无线链路是典型的自由空间传播。

在自由空间中，设发射功率为P_t，接收功率为P_r，那么有

$$P_r = \frac{A_r}{4\pi d^2} P_t G_t \qquad \text{* MERGEFORMAT (1)}$$

其中，$A_r = \frac{\lambda^2 G_r}{4\pi}$，$\lambda$为工作波长，$G_r$和$G_t$分别为发射天线和接收天线增益，$d$为发射天线与接收天线之间的距离。

当$G_r = G_t = 1$时，自由空间传播损耗可以写为

$$L = \left(\frac{4\pi d}{\lambda}\right)^2 \qquad \text{* MERGEFORMAT (2)}$$

如果用分贝表示，则可以写作

$$L = 32.45 + 20\lg f + 20\lg d$$

* MERGEFORMAT (3)

其中，$f(MHz)$为工作频率，$d(km)$为收发天线之间的距离。

3.3 树莓派的连接（SSH 协议）

SSH 是 Secure SHell protocol 的简写，安全外壳协议（SSH）是一种在不安全网络上提供安全远程登录及其它安全网络服务的协议。

OpenSSH 是 SSH （Secure SHell）协议的免费开源实现。SSH 协议族可以用来进行远程控制，或在计算机之间传送文件。而实现此

功能的传统方式，如 telnet(终端仿真协议)、rlogin、rsh 都是极为不安全的，并且会使用明文传送密码。OpenSSH 提供了服务端后台程序和客户端工具，用来加密远程控件和文件传输过程的中的数据，并由此来代替原来的类似服务。

在过去我们使用的 rsh 和 telnet，因为包括登录时的 ID 和密码数据没有加密就传到网络上，存在安全上的问题。即使在内部网上，也有在因特网上的窃取和篡改等危险性。SSH 将包括密码在内的所有数据都已进行了加密处理，可以进行更安全的远程操作。在 SSH 中，由于协议标准的不同而存在 SSH1 和 SSH2 两个不同的版本。SSH2 是为了回避 SSH1 所使用的加密算法的许可证问题而开发的。TLES 8 中作为安装 SSH 协议的应用程序采用了开放源码的 OpenSSH。OpenSSH 与 SSH1 和 SSH2 的任何一个协议都能对应，但默认使用 SSH2。

其工作原理是：

（1）服务器建立公钥：每一次启动 sshd 服务时，该服务会主动去找/etc/ssh/ssh_host*的文件，若系统刚刚安装完成时，由于没有这些公钥，因此 sshd 会主动去计算出这些需要的公钥，同时也会计算出服务器自己需要的私钥。

（2）客户端主动联机请求：若客户端想要联机到 ssh 服务器，则需要使用适当的客户端程序来联机，包括 ssh,putty 等客户端程序连接。

（3）服务器传送公钥给客户端：接收到客户端的要求后，服务器便将第一个步骤取得的公钥传送给客户端使用。

（4）客户端记录并比对服务器的公钥数据及随机计算自己的公私钥： 若客户端第一次连接到此服务器，则会将服务器的公钥记录到客户端的用户家目录内的~/.ssh/known_hosts。若是已经记录过该服务器的公钥，则客户端会去比对此次接收到的与之前的记录是否有差异。若接受此公钥，则开始计算客户端自己的公私钥。

（5）回传客户端的公钥到服务器端：用户将自己的公钥传送给服务器。此时服务器：具有服务器的私钥与客户端的公钥，而客户端则是：具有服务器的公钥以及客户端自己的私钥，你会看到，在此次联机的服务器与客户端的密钥系统 (公钥+私钥) 并不一样，所以才称为非对称加密系统。

（6）开始双向加解密：服务器到客户端：服务器传送数据时，拿用户的公钥加密后送出。客户端接收后，用自己的私钥解密。客户端到服务器：客户端传送数据时，拿服务器的公钥加密后送出。服务器接收后，用服务器的私钥解密，这样就能保证通信安全。

3.4 Raspbian 系统

本次实验树莓派上的系统为 Raspbian，是 linux 的发行版之一，为树莓派官方推荐系统。

Raspberry Pi OS（原为 Raspbian）是为树莓派基于 Debian 开发的操作系统。从 2015 年起，树莓派基金会正式将其作为树莓派的官方

操作系统。Raspbian 是由 Mike Thompson 和 Peter Green 创建的一个独立项目。第一个版本于 2012 年 6 月发布，至今仍在更新中。Raspbian 对树莓派系列的低性能 ARM 架构 CPU 进行了高度优化。

Raspbian 在其最新版本中使用新的桌面环境：PIXEL（Pi Improved X-Window Environment, Lightweight），PIXEL 由一个修改过的 LXDE 桌面环境和 Openbox 窗口管理器组成。Raspbian 发行版还附带了 Wolfram Mathematica 软件、《我的世界》的 Raspbian 版本 Minecraft Pi 和精简版的 Chromium 浏览器。

2015 年，来自 DistroWatch 的 Jesse Smith 评论 Raspbian：虽然我不打算将 Raspberry Pi 作为桌面电脑运行，但 Raspbian 操作系统确实为用户提供了 LXDE 桌面环境。树莓派没有很快的处理器或很大的内存，但它有足够的资源来运行 LXDE 和其他一些应用程序。只要用户不在同一时间执行大量计算，Raspbian 桌面界面的还是响应相当快的。我可能不会在树莓派上运行像 LibreOffice 或者 Firefox 这样比较大的程序，但是 Raspbian 确实提供了 Epiphany 网页浏览器和一些其他的桌面程序。

四、实验平台

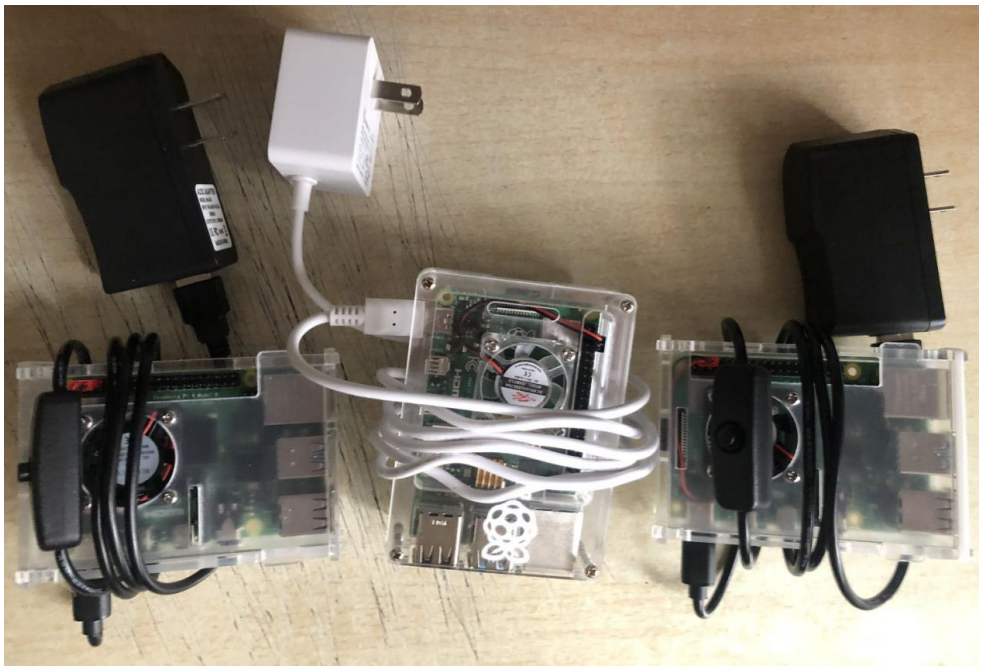

图 2 物联网定位课程设计实验平台-树莓派

设备清单包括：

- 树莓派 Raspberry Pi 4B (4G/8G) ×3
- 笔记本计算机（服务器） ×1
- 移动终端 ×1

软件平台：

服务器：windows10

树莓派：Raspberry Pi OS

移动终端：iOS

语言与包：

python3.7+bluepy+pybluez

五、实验原理

三台树莓派固定在某些位置作为基站，搜索周围的蓝牙信号并获取设备的 MAC 地址和蓝牙信号的 RSSI。利用 SSH 协议将信息上传给终端，终端将信息汇总后，根据蓝牙设备的 MAC 地址和 RSSI，利用自由空间传播模型，计算特定的蓝牙设备的位置，从而实现蓝牙定位。

六、实验内容

实验演示视频：https://www.bilibili.com/video/bv1rh411Q7HE

如下图（图 3）所示，我们将三个树莓派按一定的位置进行摆放。这三个位置分别是（0，0）、（1，0）、（1，2），它们将在定位中使用到。在确定位置之后，我们测定了自由空间传播模型的参数——系统损耗因子。自由空间传播模型被用于将信号强度（RSSI）转化为距离，然后利用距离和三边定位算法确定待定位设备的位置。

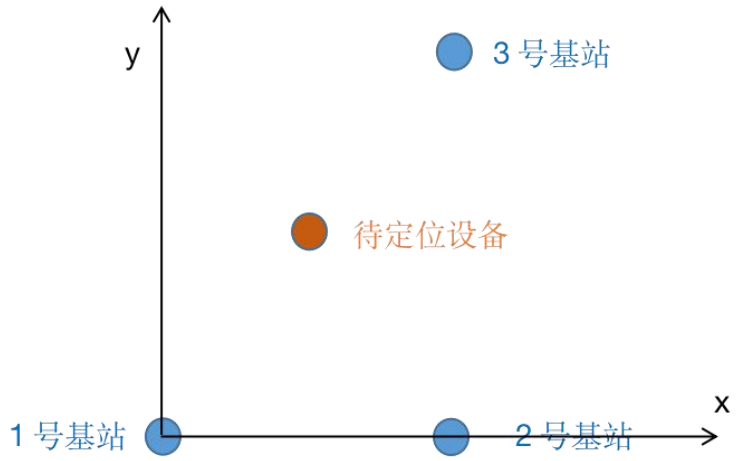

图 3 物联网定位示意图

树莓派的作用是检测环境中的蓝牙信号，这个信号的内容包括信号源的 MAC 地址和信号强度（RSSI），然后将这些数据发送给服务器，服务器将三个基站发来的信号数据中的 MAC 地址与待定位设备的 MAC 地址进行比较，当找到待定位设备的信号数据后，利用自由空间传播模型计算三个基站分别离待定位设备的距离，最后利用这些距离和三边定位算法计算待定位设备的位置。同时对于用户来说，其只需要与服务器进行交流即可在任意时间获得自己的位置信息。

在不同的环境中应用自由空间传播模型时需要测定对应的参数，这个参数需要手动测定，即拿一个蓝牙设备站在离基站一米的位置测定此时的 RSSI，然后计算该环境下的系统损耗因子。在计算系统损耗因子之后，从 RSSI 转化为距离的公式就已经确定。

七、实验结果

下图是三个基站与待定位设备的摆放位置：

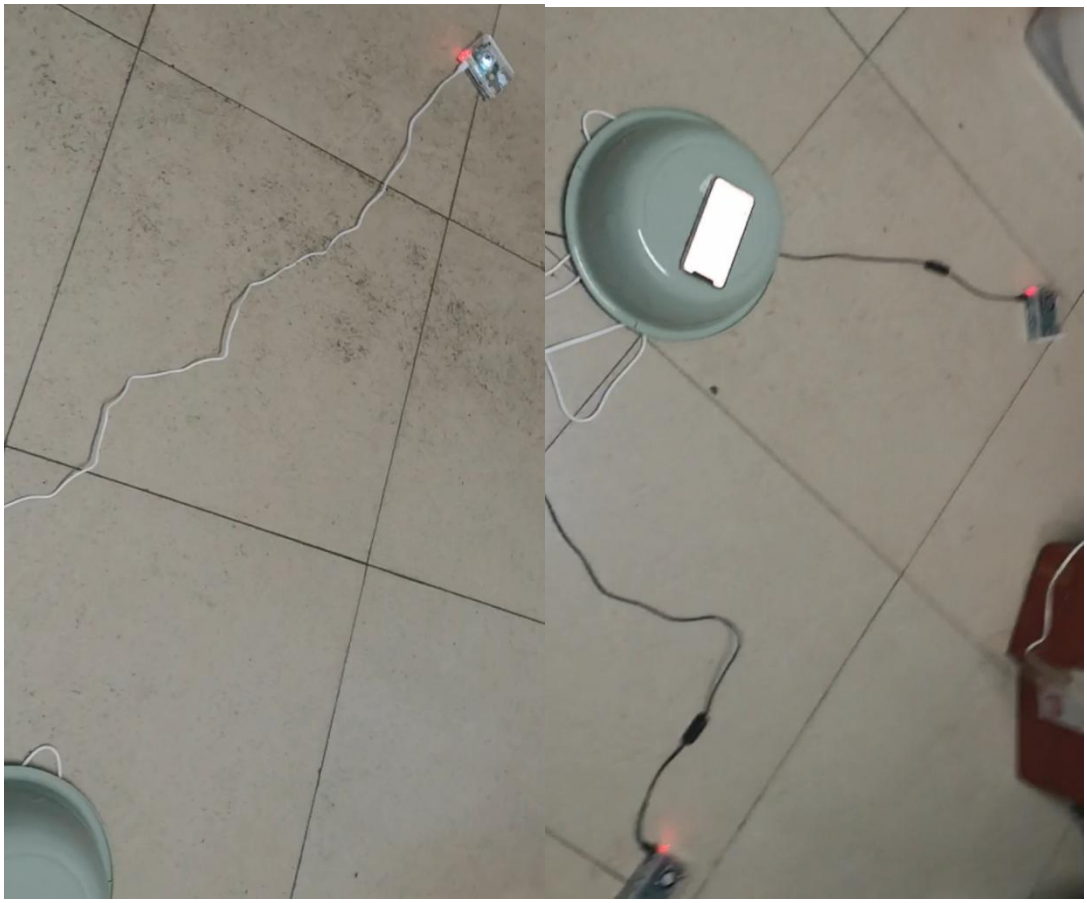

图 4 物联网定位实验基站与待测设备的位置

下图是在服务器上显示的定位结果，其中红色代表 1 号基站，绿色代表 2 号基站，蓝色代表 3 号基站。它们的位置分别是（0，0）、（1，0）、（1，2）。这三个圈的半径分别代表基站到带定位设备的距离，这也就是说这三个圈的重合部分就是带定位设备所在的位置。

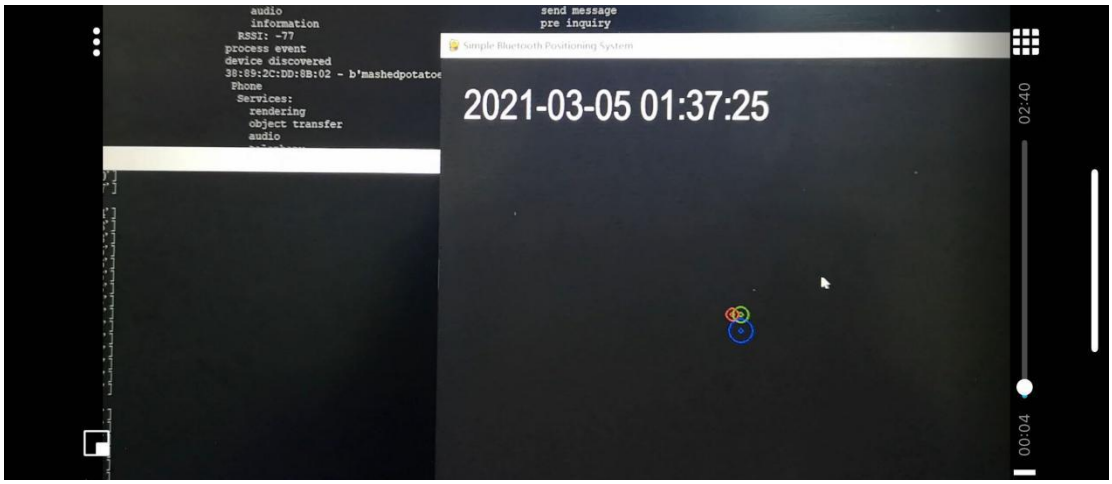

图 5 定位结果

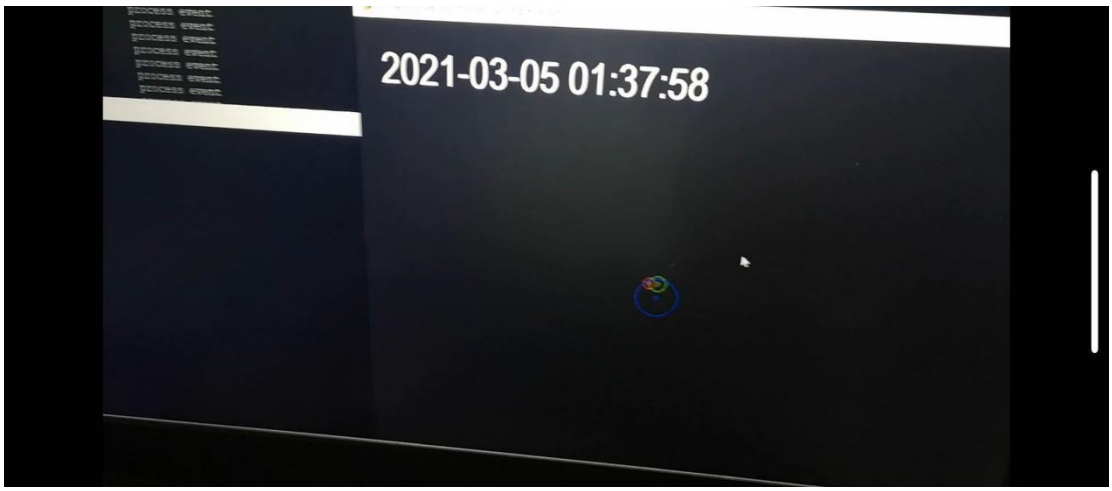

图 6 定位结果

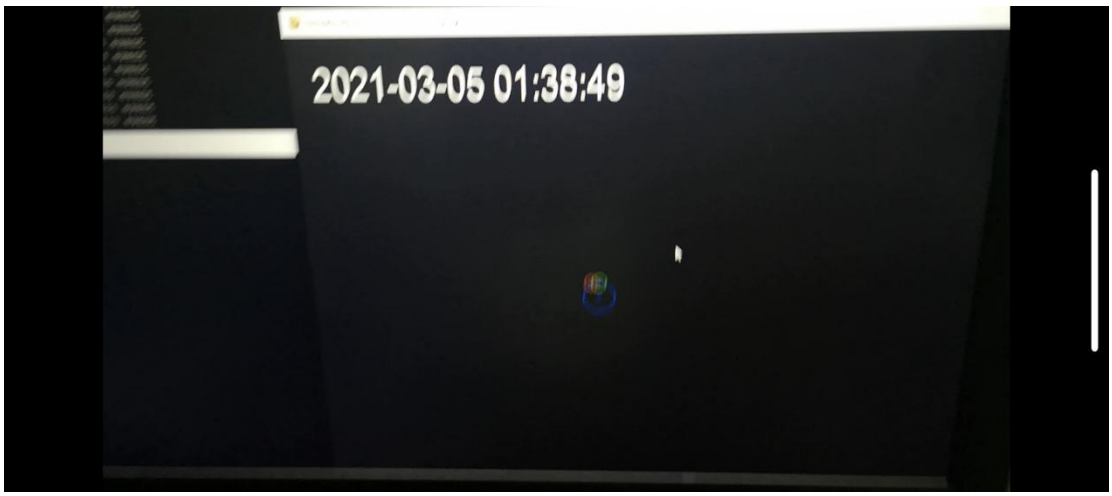

图 7 定位结果

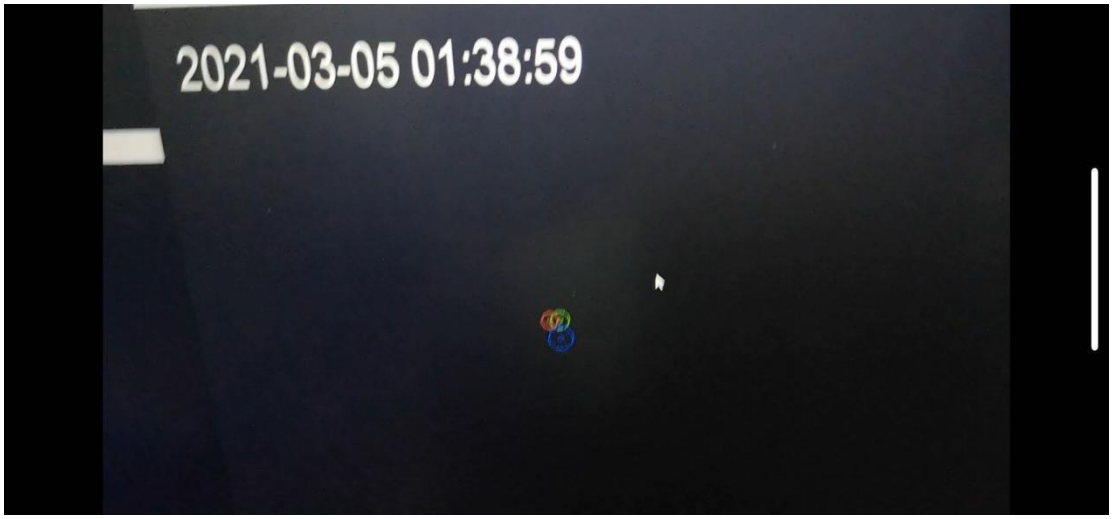

图 8 定位结果

下图是树莓派终端上的部分输出显示，树莓派上的程序在检测到蓝牙信号后，会输出这个设备的 MAC 地址、名称、设备类型、设备提供的服务、RSSI 值等信息。

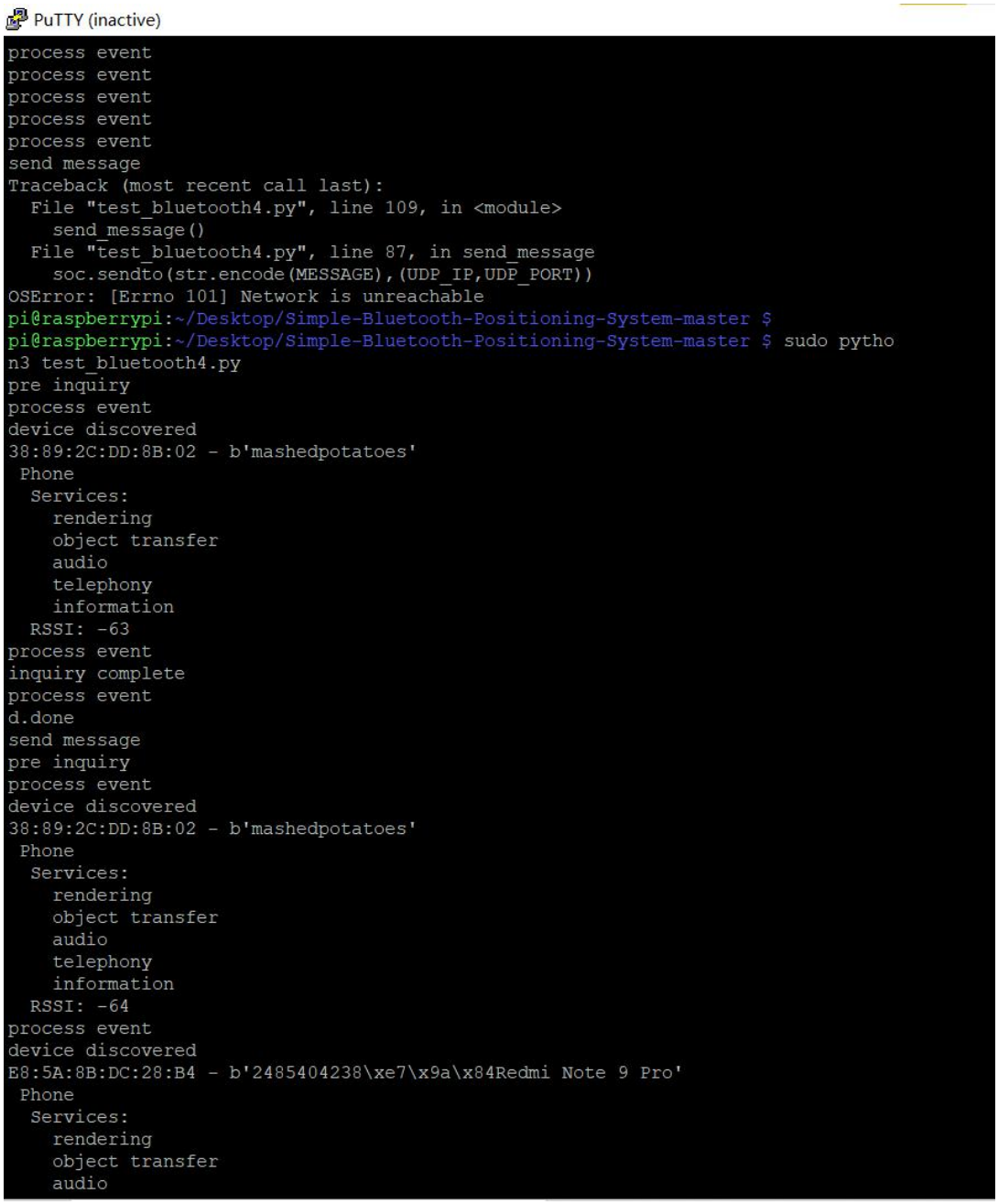
PuTTY (inactive)
process event
process event
process event
process event
process event
send message
Traceback (most recent call last):
File "test_bluetooth4.py", line 109, in <module>
send_message()
File "test_bluetooth4.py", line 87, in send_message
soc.sendto(str.encode(MESSAGE),(UDP_IP,UDP_PORT))
OSError: [Errno 101] Network is unreachable
pi@raspberrypi:~/Desktop/Simple-Bluetooth-Positioning-System-master $
pi@raspberrypi:~/Desktop/Simple-Bluetooth-Positioning-System-master $ sudo pytho
n3 test_bluetooth4.py
pre inquiry
process event
device discovered
38:89:2C:DD:8B:02 - b'mashedpotatoes'
Phone
Services:
rendering
object transfer
audio
telephony
information
RSSI: -63
process event
inquiry complete
process event
d.done
send message
pre inquiry
process event
device discovered
38:89:2C:DD:8B:02 - b'mashedpotatoes'
Phone
Services:
rendering
object transfer
audio
telephony
information
RSSI: -64
process event
device discovered
E8:5A:8B:DC:28:B4 - b'2485404238\xe7\x9a\x84Redmi Note 9 Pro'
Phone
Services:
rendering
object transfer
audio

图 9 基站接收到的信息

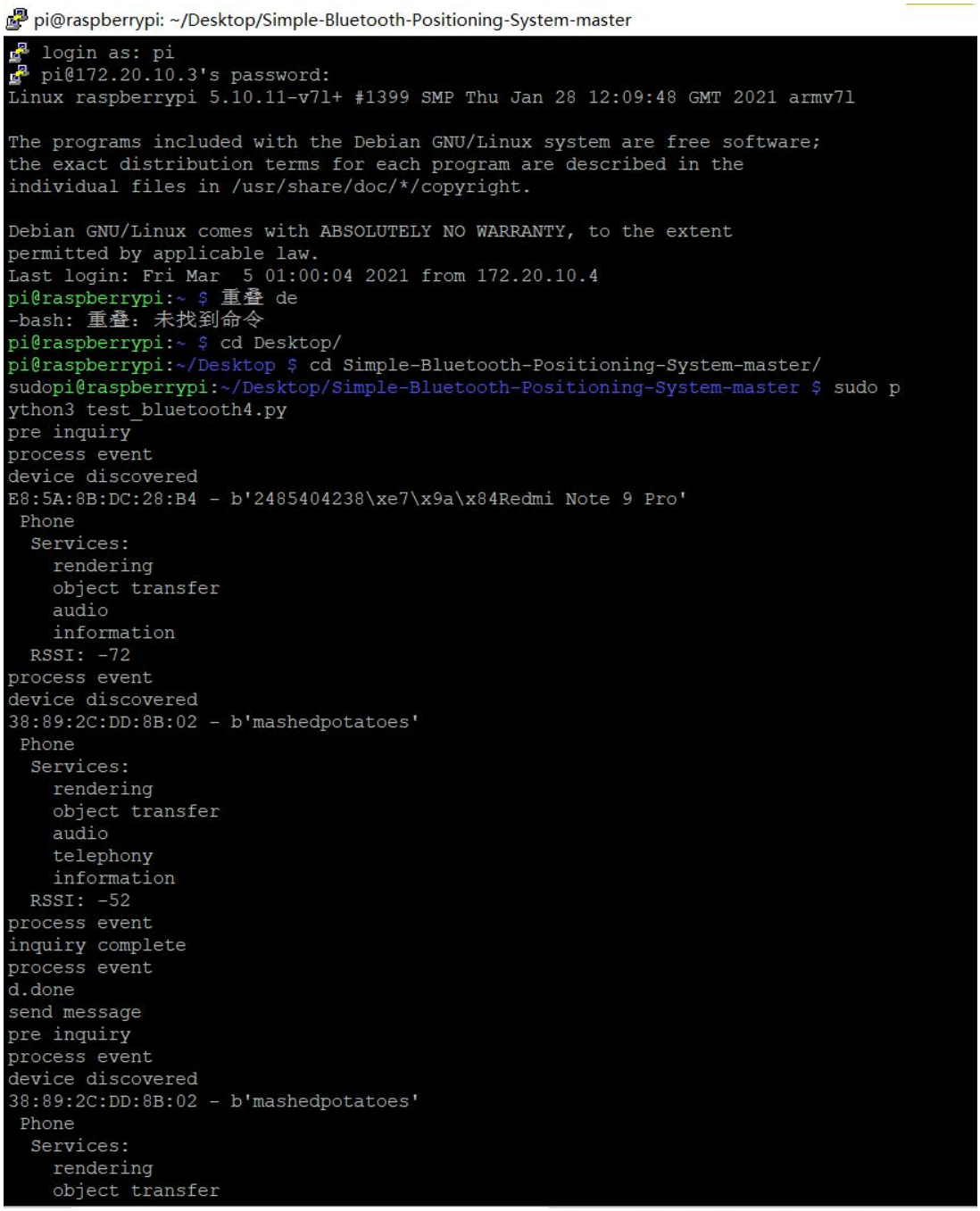

图 10 基站接收到的信息

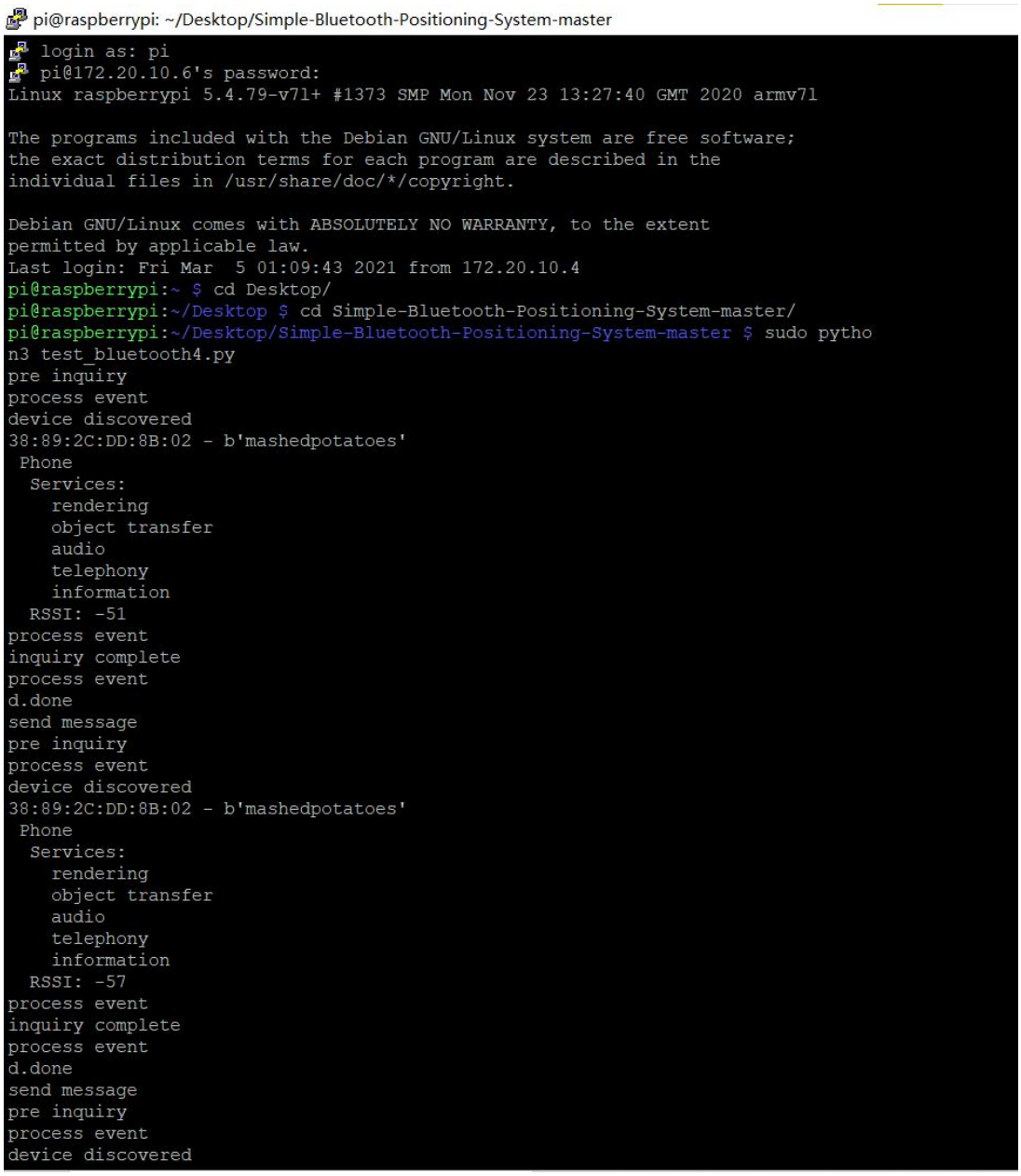

图 11 基站接收到的信息

八、应用场景

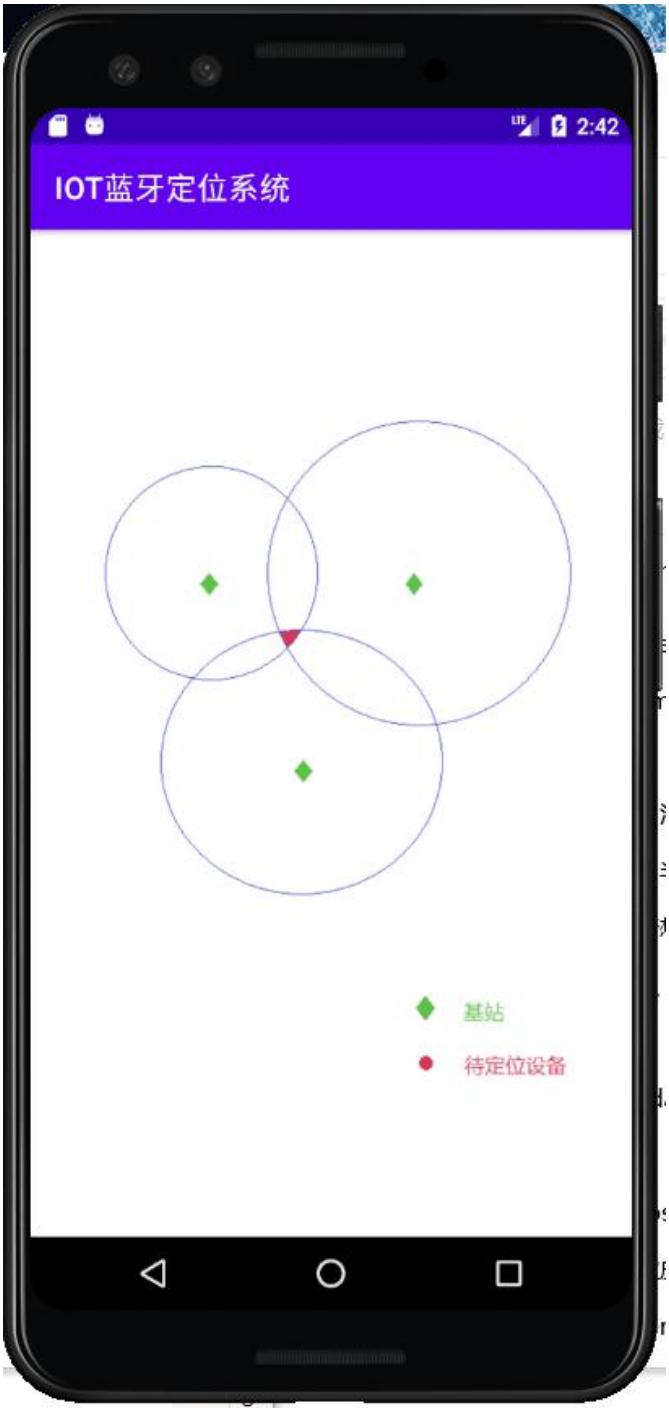

图 12 移动端的 App

蓝牙定位服务是蓝牙技能范畴增加最快的解决计划，现在现已被越来越多的运用于寻物、室内定位、室内导航以及资产追踪等范畴。

蓝牙联盟也在不断的迭代蓝牙技能，使得定位技能的精度越来越高，从最早的纯粹根据信号强度的 Beacon 定位，到蓝牙 5.1 支持的多天线视点定位等。蓝牙定位方面进行了早期的投入和深化的研制，现在现已有多款芯片支持 Beacon 定位，Mesh+Beacon，蓝牙 5.1 视点定位等技能，一起也即将推出根据蓝牙技能的厘米级定位芯片。

从计划视点来说，现在市面上的蓝牙定位体系计划大致能够分为三种，分别是主动式蓝牙定位体系，被迫式蓝牙定位体系，以及主被迫一体化室内定位体系。

主动式蓝牙定位体系是经过手持设备（手机、平板等）扫描 iBeacon 基站，进行定位的方式。采用高性能蓝牙定位算法，支持多种移动终端室内高精度定位导航，具有实时定位、途径规划、反向寻车等功能，可集成到微信大众号、小程序、APP 之中。此蓝牙定位体系偏重于室内途径规划和导航运用，可广泛运用于医院、停车场、旅游景区、办公大楼、展览馆、博物馆、火车站、游乐场、商超等场所。

被迫式蓝牙定位体系依托于定位算法、蓝牙网关和蓝牙定位标签（胸卡、手环、标签等），网关接收到蓝牙设备宣布的信息后经过服务器的运算后，在展现平台实时了解人员和物品方位的一种定位方式。此定位体系多用于后台定位监控前台人或物的运用，广泛运用于工厂资产定位、单位访客管理、养老院等场景。

主被迫一体化室内定位体系集以上体系的优势于一身。体系集成了蓝牙定位算法、蓝牙网关、蓝牙定位标签和 ibeacon 等一系列硬件，

并最大极限地减少蓝牙网关的运用。相对来说，主被迫一体化定位体系计划可大大下降布置定位硬件的成本，一起可用作主动定位和被迫定位，性价比超高。体系采用蓝牙 5.0 技能完成数据回传，具有覆盖间隔远、实时性高、并发大等技能特点，广泛适用于博物馆、化工厂、展览馆、物流仓储、养老院等场所。

蓝牙定位技能不乏许多实用的应用，例如 GPS，在世界各地被广泛运用。可惜的是，GPS 在室内运作不够完善，实务上需要更精确的室内定位技能。我们的目的是运用外部追寻体系测量单个对象的方位(或视点)，或追寻设备在室内环境中的方位。这种定位体系可应用于仓库的财物追寻或商场顾客追寻，或者人们可以用于定位寻路。

九、遇到的问题与解决方案

在开发程序过程中，我们选用 python 来实现功能。python 关于蓝牙的包主要有两个：bluepy 和 bluetooth。其中 bluepy 主要偏向于低功耗蓝牙。bluetooth 主要偏向于经典蓝牙，同时也提供对于低功耗蓝牙的支持。而网上关于这两个包的信息其实很少，同时我们刚开始时并没有注意到蓝牙和蓝牙低功耗的差别。这导致了我们在调用了低功耗蓝牙的包实现程序后，发现了一堆蓝牙信号，但就是发现不了我们的待定位设备——手机的蓝牙信号。这让我们百思不得其解，最终才在一个博客和官方文档中了解到了这一个区别。蓝牙低功耗是不向后兼容蓝牙的，因此最终我们开发的程序有两个版本：一个是蓝牙低功耗版本，一个是经典蓝牙版本。

我们利用 SSH 远程控制树莓派，将树莓派连接到个人热点组成的局域网，并用 putty 远程终端控制树莓派上的 linux 系统。同时，服务器和待定位设备也是连接在这个局域网上。由于个人热点创造的局域网不够稳定、带宽受限，因此在树莓派（基站）向服务器传送数据时，树莓派时常断连。程序也会因为无法获取网络而报错中断。如果有良好的局域网使得树莓派与服务器通信更好的话，系统的稳定性与实用性会大大提高。如果让我重来一次，我一定购买一个显示器用来操作树莓派。

十、心得体会

这次实验中的成功与失败都给了我们丰富的体验，让我们体会到开发的不易。实际应用中的 bug 往往出乎意料，所以我们只有将知识掌握的更加牢固，将能力提升到更高的水平，才能够在实际应用中披荆斩棘，化腐朽为神奇。而成功的体验更是让我们体会到从知识到实践的喜悦，这种将自己所学化为现实的感觉无可替代，给了我们更加充分的自信心。我相信，这次实验对于别人来说是一小步，但对我个人来说是一大步。

这次实验遇到的 bug 大多莫名其妙，难以明白为何产生，也不明白如何解决。后来我发现 debug 同时需要两个因素才能成功：一是相关知识，有了相关知识才能明白为什么出错，而明白为何出错后离解决 bug 也不远了；二是灵感，长时间坐在电脑前漫无目的的四处查询往往并无结果，而暂时放开这个问题，散散步或者处理一些生活上的事情，做这些看似不相关的事情之后却能获得灵感，往往能从一个新的角度看待问题，或是能够想通问题的本质，这能解决之前看似毫无头绪的问题。关于灵感的想法，我发现其实我对于问题的解决往往也是在散步过程中解决的，坐在电脑前使劲敲击键盘只是在将脑海中的想法付诸实施。而对于写代码来说，想法或者说解决问题的算法才是最重要的，抛开这个方法，其实无论是什么样的语言，或是代码风格都是不重要的，语言只是工具而已，关键是对于问题的想法，解决办法等。

附录

代码片 1 基站代码 monitor.py

```
import socket,math,time,pygame
from datetime import datetime

obj_MAC = "38:89:2C:DD:8B:02"#
obj_dis=[0,0,0]
# stations = [[3,0.4,1.5],[5,3,1.5],[0.35,1.4,1.5]]
stations = [ [0 , 0], [1, 0], [1, 2] ]

UDP_IP = "172.20.10.4" #ip
UDP_PORT0 = 5004
UDP_PORT1 = 5005
UDP_PORT2 = 5006

socket0 = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
socket1 = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
socket2 = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
socket0.bind((UDP_IP, UDP_PORT0))
socket1.bind((UDP_IP, UDP_PORT1))
socket2.bind((UDP_IP, UDP_PORT2))

pygame.init()

FPS = 40
fpsClock = pygame.time.Clock()

size = width,height = 900,700
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Simple    Bluetooth    Positioning
System")
screen.fill((44,44,44))
font = pygame.font.SysFont("arial", 50)

check0 = False
check1 = False
check2 = False

while True:
   data0, addr0 = socket0.recvfrom(1024)
   tmp0 = str(data0)
```

```
    message0 = tmp0.split('\'')[1]
    tmp0 = message0.split('|')
    devices_num0 = len(tmp0)-2
    devices0 = [["","",""] for i in range(devices_num0)]
    for i in range(devices_num0):
        s = tmp0[i+1].split(',')
        devices0[i][0] = s[0]#station_id
        devices0[i][1] = s[1]#dev_MAC
        devices0[i][2] = s[2]#RSSI
        print(devices0[i])
    print()
    id_station0 = devices0[0][0]
    for i in range(devices_num0):
        if devices0[i][1] == obj_MAC:
            check0 = True
            rssi0 = int(devices0[i][2])
            distance0 = math.pow(10,((abs(rssi0)-53)/(10*5)))#这个
公式可能需要测试参数

    data1, addr1 = socket1.recvfrom(1024)
    tmp1 = str(data1)
    message1 = tmp1.split('\'')[1]
    tmp1 = message1.split('|')
    devices_num1 = len(tmp1)-2
    devices1 = [["","",""] for i in range(devices_num1)]
    for i in range(devices_num1):
        s = tmp1[i+1].split(',')
        devices1[i][0] = s[0]#station_id
        devices1[i][1] = s[1]#dev_MAC
        devices1[i][2] = s[2]#RSSI
        print(devices1[i])
    print()
    id_station1 = devices1[0][0]
    for i in range(devices_num1):
        if devices1[i][1] == obj_MAC:
            check1 = True
            rssi1 = int(devices1[i][2])
            distance1 = math.pow(10,((abs(rssi1)-53)/(10*5)))

    data2, addr2 = socket2.recvfrom(1024)
    tmp2 = str(data2)
    message2 = tmp2.split('\'')[1]
    tmp2 = message2.split('|')
    devices_num2 = len(tmp2)-2
```

```
    devices2 = [["","",""] for i in range(devices_num2)]
    for i in range(devices_num2):
        s = tmp2[i+1].split(',')
        devices2[i][0] = s[0]#station_id
        devices2[i][1] = s[1]#dev_MAC
        devices2[i][2] = s[2]#RSSI
        print(devices2[i])
    print()
    id_station2 = devices2[0][0]
    for i in range(devices_num2):
        if devices2[i][1] == obj_MAC:
            check2 = True
            rssi2 = int(devices2[i][2])
            distance2 = math.pow(10,((abs(rssi2)-53)/(10*5)))

    if check0 and check1 and check2:
        screen.fill((44,44,44))
        #高度(z)相同
        sta0                                                    =
pygame.draw.circle(screen,(255,0,0),(370+int(10*float(stations
[0][0])),320+int(10*float(stations[0][1]))),3,1)
        sta1                                                    =
pygame.draw.circle(screen,(0,255,0),(370+int(10*float(stations
[1][0])),320+int(10*float(stations[1][1]))),3,1)
        sta2                                                    =
pygame.draw.circle(screen,(0,0,255),(370+int(10*float(stations
[2][0])),320+int(10*float(stations[2][1]))),3,1)



    pygame.draw.circle(screen,(255,0,0),(370+int(10*float(stati
ons[0][0])),320+int(10*float(stations[0][1]))),int(10*(distanc
e0)),1)

    pygame.draw.circle(screen,(0,255,0),(370+int(10*float(stati
ons[1][0])),320+int(10*float(stations[1][1]))),int(10*(distanc
e1)),1)

    pygame.draw.circle(screen,(0,0,255),(370+int(10*float(stati
ons[2][0])),320+int(10*float(stations[2][1]))),int(10*(distanc
e2)),1)

        t = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        text = font.render(t,True,(255,255,255))
        screen.blit(text, (30, 30))
```

```
        pygame.display.update()
        fpsClock.tick(FPS)
        check0 = False
        check1 = False
        check2 = False

    for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
```

代码片 2 终端代码 monitor.py

```
import select

import bluetooth

from bluepy.btle import Scanner, DefaultDelegate
import socket,time

station_id = "1"
UDP_IP = "172.20.10.4" #receiver's ip
UDP_PORT = 5004
soc = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)

flag_send=0
num_find=0
devices_info=[]


class MyDiscoverer(bluetooth.DeviceDiscoverer):

    def pre_inquiry(self):
        print('pre inquiry')
        self.done = False

    def  device_discovered(self,  address,  device_class,  rssi,
name):
        print('device discovered')
        global flag_send
        flag_send=flag_send-1
        global num_find
        num_find=num_find+1
        print("{} - {}".format(address, name))
```

```
        # get some information out of the device class and display
it.
        # voodoo magic specified at:
        #
https://www.bluetooth.org/foundry/assignnumb/document/baseband
        major_classes = ("Miscellaneous",
                         "Computer",
                         "Phone",
                         "LAN/Network Access Point",
                         "Audio/Video",
                         "Peripheral",
                         "Imaging")
        major_class = (device_class >> 8) & 0xf
        if major_class < 7:
            print(" " + major_classes[major_class])
        else:
            print("  Uncategorized")

        print("  Services:")
        service_classes = ((16, "positioning"),
                           (17, "networking"),
                           (18, "rendering"),
                           (19, "capturing"),
                           (20, "object transfer"),
                           (21, "audio"),
                           (22, "telephony"),
                           (23, "information"))

        for bitpos, classname in service_classes:
            if device_class & (1 << (bitpos-1)):
                print("   ", classname)
        print("  RSSI:", rssi)

       # device_info=[]
        global station_id
       # device_info.append(station_id)
       # device_info.append(address)
       # device_info.append(rssi)

devices_info.append(station_id+","+address+","+str(rssi))
       # print(device_info)
       # print(devices_info)

    def inquiry_complete(self):
```

```
        self.done = True
        print('inquiry complete')


def send_message():
    print('send message')
    global devices_info
    global soc
    global UDP_IP
    global UDP_PORT
    MESSAGE="|"
    for i in range(len(devices_info)):
        MESSAGE+=devices_info[i]+"|"
    #print("----------------"+MESSAGE)
    soc.sendto(str.encode(MESSAGE),(UDP_IP,UDP_PORT))
    time.sleep(0.2)
    global flag_send
    flag_send=0



while True:
    d = MyDiscoverer()
    d.find_devices(lookup_names=True,duration=8)

    readfiles = [d, ]

    while True:
        rfds = select.select(readfiles, [], [])[0]
        #print(rfds)

        if d in rfds:
            d.process_event()
            print('process event')
            flag_send=flag_send+1
            if flag_send>10:
                send_message()
                break

        if d.done:
            print('d.done')
            send_message()
            break
```

代码片 3 终端代码 monitor.py（低功耗蓝牙版本）

```
from bluepy.btle import Scanner, DefaultDelegate

import socket,time



station_id = "1"

UDP_IP = "" #receiver's ip

UDP_PORT = 5004

soc = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)



class ScanDelegate(DefaultDelegate):

    def __init__(self):

        DefaultDelegate.__init__(self)



    def handleDiscovery(self, dev, isNewDev, isNewData):

        if isNewDev:

            print "Discovered device", dev.addr

        elif isNewData:
```

```
                print "Received new data from", dev.addr



scanner = Scanner().withDelegate(ScanDelegate())
while True:
    devices = scanner.scan(2.0)
    DevicesInfo = ["" for i in range(len(devices))]
    i = 0
    for dev in devices:
        print "Device %s , RSSI=%d dB" % (dev.addr , dev.rssi)
        DevicesInfo[i] = station_id+","+dev.addr+","+str(dev.rssi)
        i = i+1
    MESSAGE = "|"
    for i in range(len(DevicesInfo)):
        MESSAGE += DevicesInfo[i]+"|"
    soc.sendto(MESSAGE, (UDP_IP,UDP_PORT))
    time.sleep(0.2)
```