实验代码Readme

一、Python

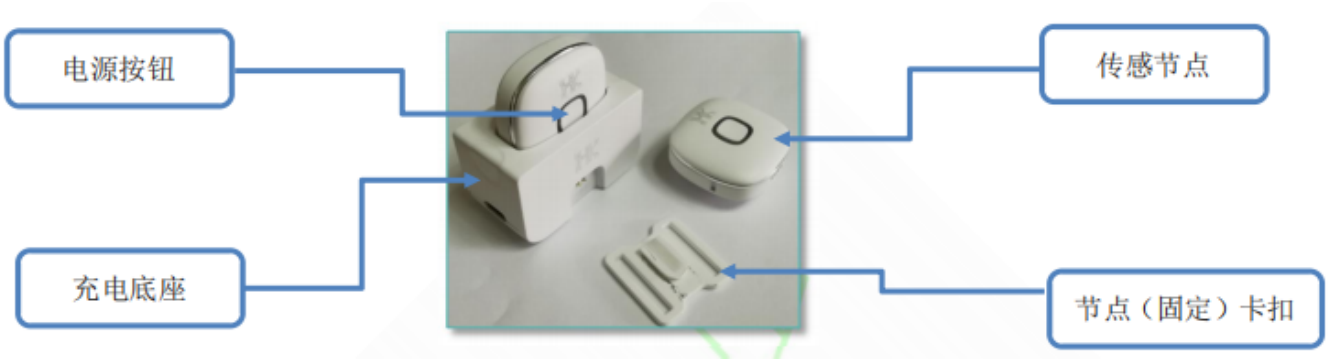
默认已经安装好python环境，没有安装好的同学可以自行在billbill上搜索安装教程。推荐编译器为pycharm

参考：

【【官方正版】pycharm 安装方法】https://www.bilibili.com/video/BV12y4y1E7Zf?vd_source=8d291ecc956e61be0fa5900d0611fbe0

二、IMU产品说明

产品构成：



产品使用：

(1) 长按节点“电源按钮”3 秒开机，节点电源指示灯**蓝灯闪烁**；

(2) 将节点放置桌面进行陀螺仪校准，1~2 秒校准结束后节点电源指示灯**蓝灯常亮**；

(3) 将节点拿起左右晃动进行地磁校准，2~3 秒地磁校准后节点电源指示灯**绿灯闪烁**；

(4) 打开电脑/手机蓝牙，查找节点的蓝牙名字（在节点背面的二维码和文字标识，每 个节点的蓝牙名都唯一），并连接节点蓝牙，连接成功后节点电源指示灯**绿色常亮**；

(5) 通过电脑/手机向节点发送通讯协议命令操作；

三、节点输出数据协议

命令 AB01FFFFFF，获取所有节点输出数据，内容如下

帧头 (1byte)	帧头 (1byte)	包大小 (1byte)	电量 (1byte)	四元数 X (2byte)	四元数 Y (2byte)	四元数 Z (2byte)	四元数 W (2byte)
0xBA	0xC0	核心数据包输出数据总字节数 (0-180)	电池电量 (0-100)	低位+高位使用需缩小 10000 倍	低位+高位使用需缩小 10000 倍	低位+高位使用需缩小 10000 倍	低位+高位使用需缩小 10000 倍
欧拉角 X (2byte)		欧拉角 Y (2byte)		欧拉角 Z (2byte)	加速度 X (2byte)	加速度 Y (2byte)	加速度 Z (2byte)
低位+高位使用需缩小 100 倍		低位+高位使用需缩小 100 倍		低位+高位使用需缩小 100 倍	低位+高位	低位+高位	低位+高位

陀螺仪 X (2byte)	陀螺仪 Y (2byte)	陀螺仪 Z (2byte)	地磁 X (2byte)	地磁 Y (2byte)	地磁 Z (2byte)
低位+高位 使用需缩小 10 倍	低位+高位 使用需缩小 10 倍	低 位 + 高位 使用需缩小 10 倍	低位+高位 使用需放大 100 倍	低位+高位 使用需放大 100 倍	低位+高位 使用需放大 100 倍
位移 X (2byte)	位移 Y (2byte)	位移 Z (2byte)			
低位+高位	低位+高位	低 位 + 高位			

例如输出数据：

ba c0 28 27 1e 15 8e 00 7f 01 2e df 77 01 e7 fe 3f d3 2c ff 8d fd 82 40 e2 ff 28 00 ec ff 22 00 ee ff 31 00 fc ff ff ff 00 00

ba c0 为帧头

28 代表包大小，此处为 16 进制，转换成 10 进制为 40，即核心包数据总大 小为 40 字节

27 代表电量，此处为 16 进制，转换成 10 进制为 39，即还有 39%的电量

1e 15 8e 00 7f 01 2e df 分别是四元数的 XYZX，需要高低位转换即 le 15 变成 15 1e 后转换成十进制 5406，然后缩小 10000 倍即四元数 X 为 0.5406

77 01 e7 fe 3f d3 分别是欧拉角的 XYZ，同样需要高低位交换后转换成十 进制，然后缩小 100 倍，欧拉角的范围为-180°~180°

2c ff 8d fd 82 40 分别是加速度的 XYZ

e2 ff 28 00 ec ff 分别是陀螺仪的 XYZ

22 00 ee ff 31 00 分别是地磁的 XYZ

fc ff ff ff 00 00 分别是位移的 XYZ

PS：四元数，欧拉角，加速度，陀螺仪（角速度），地磁，位移都需要按双字节转换，不要按四字节转换

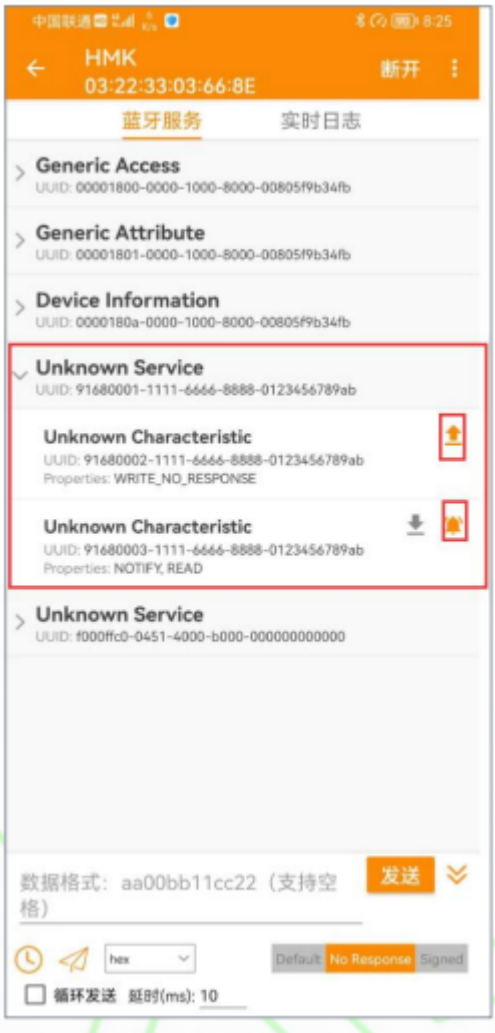
四、蓝牙连接测试

1、imu正常开启并进入蓝牙可搜索状态（节点电源指示灯绿灯闪烁）

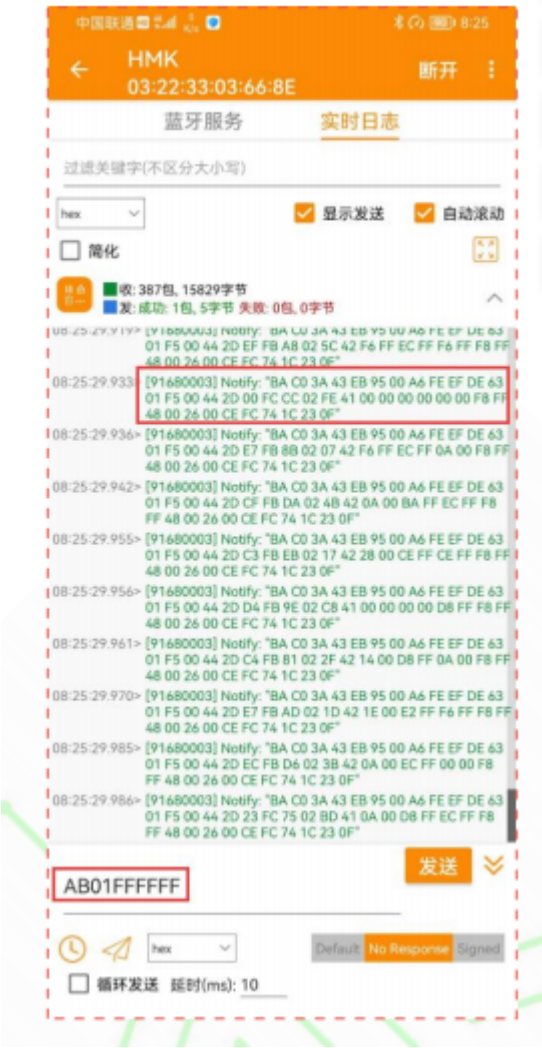
2、安装BLE蓝牙工具助手

3、打开工具，开启蓝牙搜索节点名字，连接节点

4、选择UUID：91680002-1111-6666-8888-0123456789ab的项，并开启如图的订阅设置



5、发送命令AB01FFFFFF获取节点数据



五、代码详解

1、在pycharm中打开对应的代码文件，并且安装相对应的包

```
#一些必要的安装包
import binascii
import asyncio
import torch
from pygame.time import Clock
from bleak import BleakClient
from threading import Thread
```

2、获取自己的IMU的地址和设备名称（步骤四），并改写代码（已经有示例）

```
imu_devices = [
    # {"address": "03:85:14:03:1F:B6", "name": "0722"},
    # #{"address": "03:85:14:03:1D:1A", "name": "0276"},
    # #{"address": "03:85:14:03:1C:E3", "name": "0273"},
    # {"address": "03:85:14:03:1C:CC", "name": "1006"},
    # {"address": "03:85:14:03:1F:25", "name": "0556"},
    {"address": "03:85:14:03:1C:99", "name": "0505"},
    # {"address": "03:85:14:03:94:25", "name": "0184"},
    #imu的蓝牙地址和名称
```

3、处理得到自己想要的数据：

在ble_server类中的data_transition函数里已经对实验数据进行了示例处理，注：放大缩小的倍数根据自己所需要的单位进行处理。处理完之后可以对其进行再操作。代码示例举了一个输出四元数的操作。

```
2 用法
def data_transition(self):
    data = self.get_latest_data()
    head = data[0:4]
    size = int(data[4:6], 16)
    battery = int(data[6:8], 16)
    quat_x = (HexSting2decimal(data[10:12] + data[8:10])) / 10000
    quat_y = (HexSting2decimal(data[14:16] + data[12:14])) / 10000
    quat_z = (HexSting2decimal(data[18:20] + data[16:18])) / 10000
    quat_w = (HexSting2decimal(data[22:24] + data[20:22])) / 10000
    angle_x = (HexSting2decimal(data[26:28] + data[24:26])) / 100
    angle_y = (HexSting2decimal(data[30:32] + data[28:30])) / 100
    angle_z = (HexSting2decimal(data[34:36] + data[32:34])) / 100
    acc_x = (HexSting2decimal(data[38:40] + data[36:38])) / 100
    acc_y = (HexSting2decimal(data[42:44] + data[40:42])) / 100
    acc_z = (HexSting2decimal(data[46:48] + data[44:46])) / 100
    angvel_x = (HexSting2decimal(data[50:52] + data[48:50])) / 80
    angvel_y = (HexSting2decimal(data[54:56] + data[52:54])) / 80
    angvel_z = (HexSting2decimal(data[58:60] + data[56:58])) / 80
    geomag_x = (HexSting2decimal(data[62:64] + data[60:62])) * 100
    geomag_y = (HexSting2decimal(data[66:68] + data[64:66])) * 100
    geomag_z = (HexSting2decimal(data[70:72] + data[68:70])) * 100
    tran_x = (HexSting2decimal(data[74:76] + data[72:74])) / 1000
    tran_y = (HexSting2decimal(data[78:80] + data[76:78])) / 1000
    tran_z = (HexSting2decimal(data[82:84] + data[80:82])) / 1000

    q = torch.tensor([quat_x, quat_y, quat_z, quat_w])
    print(q)
1 个用法
```

4、点击运行，运行之后输出结果，得到实时传输的IMU的四元数数据。

```
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
tensor([ 0.6941,  0.0530, -0.0403, -0.7167])
```