前言

客户端软件主要运行于树莓派单片电脑上，操作系统为Raspbian OS（主要基于Linux Debian发行版，安装新软件可以使用Debian的apt-get，比较方便），软件使用python3编写，使用Qt 5软件库制作GUI图形界面。因为Qt 5现已支持python3编写，使用了第三方库pyqt5来编写图形界面，编写或修改过程会更加简便。

蓝牙部分使用了第三方库bluepy，在最终决定使用bleupy前已尝试过bleak和pygatt，存在蓝牙数据传输会在心电数据采样率提高时中断的问题，而本人尚未来得及深入研究将其解决，遂选择bluepy。bluepy有其完善的说明文档和github仓库，其中不乏使用案例，如需要修改蓝牙部分的代码，可以参考这些案例。

为实现调用摄像头拍照或录制视频的功能，使用了opencv的python3支持库cv2。Opencv开源，使用方便，但也存在一些bug，此处仅起到告知作用，希望有能力的同学为本软件添加更好的摄像头操作库或代码。

网络通信方面，由于与C语言系的软件进行通信，使用了struct库，用以将字节串解读为打包的二进制数据，形成给定的C结构体，确保正确的对齐。

对于诸如Python与C++或是Python与Java等语言之间的性能差异问题，客观来说是存在的，但是通过代码的优化可以缩小差距，本人水平有限，目前只能说是能够正常运行软件，还用不上上流畅或是丝滑这类表述。此外本人代码能力一般，对于Python的熟练度不高，在代码编写过程中，对于前后端分离、各个类的继承、实现、使用，各个功能模块的实现等方面有许多不足，但代码格式、函数和变量命名尽量做到规范，所需的一些注释、各个类和其成员函数的说明都给出了简要说明，希望接手这份代码的同学轻喷。

界面说明

界面主要包括登录界面、注册界面、主界面、蓝牙搜索连接界面、蓝牙数据接收界面、用户数据管理界面、IP地址及端口号填写界面、未同步数据管理界面、诊断报告管理界面、诊断报告查阅界面。

界面代码说明：

对于Qt 5的GUI代码编写，详细的请自行找教程学习，其中的各类图形元素，如：button、label、editor、text等，以及各类layout布局方式、操作时的event、timer、thread、信号与槽函数机制等需要重点学习。

每个界面的代码均包含在构造函数\_\_init\_\_()、set\_ui()、set\_label()、set\_line\_edit()、add\_button()等函数中。如需修改界面，请阅读并修改上述部分，此外会有少部分代码在其他位置出现，如：修改蓝牙连接状态的语句出现在BLE.py中change\_BLE\_state()中，类似的情况请仔细阅读后修改。

后台功能代码说明

主界面部分：

主界面是全部功能模块的入口。主界面类MainWindow有许多成员变量，包括：

User\_account\_name 用户账号名

User\_name 用户姓名

User\_id 用户的身份证号码

User\_gender 用户性别

User\_room 用户所住房间

User\_enter\_date 用户的入园日期

User\_unique 用户的唯一标识编号

Profile 用户的照片

Ip\_address 用于后续网络通信的ip地址

Port 端口号

主界面在创建时会获取登录界面传递过来的用户信息，\_get\_user\_info()就是将这些信息准备好，然后显示。

\_get\_user\_profile()，通过用户的唯一标识编号，连接数据库获取用户的照片，然后显示。

\_slot\_acquire()，蓝牙操作的入口，BLE类的实例在此创建，并将BLE的信号函数于其槽函数\_update\_data绑定，当BLE类的信号发送时，\_update\_data会被调用。

\_update\_data()，将最近一次测量的数据显示在主界面。

\_slot\_data\_manager()，用户历史数据管理的入口，DataManager类的实例创建于此。

\_slot\_data\_sync()，用户数据同步的最终入口，DataManager类的实例创建于此。

\_slot\_change\_ip()，修改已填写的ip地址和端口号。

\_slot\_get\_ip()，用户数据同步的最初入口，如果ip\_address和port没有被定义就会创建GetIP类的实例，即ip和端口号的填写界面，如果已填写则将直接打开数据同步界面。

\_slot\_set\_ip()，当ip和端口号未定义时，在\_slot\_get\_ip()函数中调用。

\_slot\_log\_check()，诊断报告管理界面的入口。

\_slot\_video\_saving()，保存拍摄的视频。

\_slot\_video\_start()，开始录制视频。此处用了另一个线程处理视频录制，以防止主程序阻塞死。

CameraHandler类：

用于处理摄像头预览和视频拍摄，后续的视频处理算法可以考虑加入到这里，或完全重构这个类。代码参考opencv的python操作示例修改，修改前可以多阅读相关示例代码。

DataManager类与LogManager类：

管理用户历史数据和诊断报告，主要包括数据库的增、删、查操作，以及数据显示和刷新的操作，需要修改时找到对应代码修改即可。

蓝牙部分：

蓝牙的搜索、连接以及notify开启均参考了bleupy示例代码，封装进了数个函数中。其中扫面与连接使用了封装后的Util.Scanner类、蓝牙数据接收使用了DataAcquire类。

BLE.py说明：

Slot\_search\_or\_refresh()，实现蓝牙设备的扫描和扫描结果刷新。扫描结果通过TableWidget来存放，仅显示了名称和MAC地址，此处可修改显示方式或显示内容。

Select\_item()，当点击某个扫描结果时，获取被选中的蓝牙MAC地址，并返回，此处可修改或优化扫描结果显示方式。

Slot\_connect()，此处实现蓝牙连接，语句调用有些许不规范，可选择是否修改实现方式。

Slot\_acquire\_data()，开启数据接收，此处仅为跳转。

Change\_BLE\_state()，提示蓝牙连接状态，当状态改变时改变显示色块的颜色，此处实现有一点bug，当断开连接后颜色不能改变。

Slot\_disconnect()，用于断开蓝牙连接。

DataAcquire.py说明：

接收蓝牙数据并实时显示，这部分代码写的比较复杂冗长，实时数据显示的部分性能和效果有瑕疵，主要问题可能在pyqtgraph的底层实现上。如果需要优化，考虑使用更好的绘图库。

实时数据显示的实现使用了多线程，并使用了队列（Queue）数据结构存储中间数据。线程setter接收蓝牙发送的数据，通过signal形式同步给线程getter，线程getter对蓝牙数据进行解析，并存储在队列中，定时器timer已某个时间间隔取出队列中的数据，然后绘制。

\_peripheral 传入的蓝牙设备

User\_unique 用户身份编码，用于后续数据存储

Data\_type 用于标识连接的蓝牙设备能够采集的数据类型

Channel1\_type 通道一的类型

Channel2\_type 通道二的类型

Start\_time 开始接收的时间

Two\_channel 标明是否两个通道同时使用，单通道时，只有通道二使用，测量心电。双通道时，如果data\_type为ENR时，通道一为呼吸，通道二为心电，data\_type为ECG时，双通道测量心电。

Result1 数组，存放通道一结果，十进制

Result2 数组，存放通道二结果，十进制

Display1 队列，存放结果与result1一样，用于显示

Display2 队列，存放结果与result2一样，用于显示

Original1 数组，存放通道一结果，十六进制补码

Original2 数组，存放通道二结果，十六进制补码

Display\_channel1 数组，暂时存放通道一显示内容，用以实现窗体滑动

Display\_channel2 数组，暂时存放通道二显示内容，用以实现窗体滑动

Curve\_channel1 数组，真正看到的窗体，显示通道一

Curve\_channel2 数组，真正看到的窗体，显示通道二

Pointer\_channel1 窗体下标

Pointer\_channel2 窗体下标

Type\_determine()，判断连接的蓝牙设备的采集数据类型，如果有更好的判断方式，这个函数可以修改或直接不要。

Slot\_receive()，开启notification，开启线程和定时器，得到开始接收的时刻。

Show\_data()，此处函数命名有误，用以同步setter与getter线程，将setter获得的数据发送给getter线程。该函数为槽函数，由setter发送的signal触发。

Slot\_clear()，清空显示界面。

Slot\_save()，保存接收到的数据，录入数据库，并存进硬盘。保存数据前的预处理中，需要连接数据库，通过user\_unique获取用户信息（用户形名和账号）。默认情况下，预先处理通道二的数据，如果two\_channel为True时再处理通道一的数据，并根据一系列信息设置数据的存储路径名，这些信息包括用户姓名、唯一标识编号、数据类型、数据是否被处理（即十六进制数据、十进制数据以及滤波后的数据）、数据获取时间。目前的信号处理代码仅调用心电数据处理，尚未编写根据数据类型调用相对应算子的代码，此处之后需要添加。

Slot\_display()，实时显示数据，参照pyqtgraph示例代码编写并封装，如需修改可先参考示例代码。由timer定时触发。

Setter类用于接收蓝牙发送的特征值，notification等，并通过信号函数触发槽函数，以实现与Getter的线程同步。Setter需要Util.MyDelegate类的协助，Util.MyDelegate类同样根据bluepy的示例代码修改并封装。

Setter类在构造时传入已连接的蓝牙设备，并与Util.MyDelegate绑定，同时需要往Util.MyDelegate里传入信号函数。在线程创建并开启时，线程开启notify，进入无线循环等待notification。当捕获一个notification时，绑定的Util.MyDelegate中，handleNotification()函数会被调用。前三个notification会被丢弃，目的是丢弃不稳定的数据，因为蓝牙数据刚开始传输的头两三个notification常出现不期望看到的乱码，不便于进行解析。当notification中的信息被接收到时，handleNotification()会将其通过信号函数发送。

Getter类用于接收Setter类的线程实例同步过来的数据，当setter线程中的信号函数将数据发送后，其相连接的槽函数show\_data()会触发，在show\_data()中，getter类的get()方法会被调用，将信号函数发过来的数据放进getter类的成员变量q中。q是一个队列，用于保存从setter得到的数据。在getter的线程循环中，线程会不断将q中的数据从头取出，并通过Util.Splitter的帮助将蓝牙数据解析成补码、十进制数，并最终通过信号滤波得到结果。