

# 浙江大学

## 本科实验报告

课程名称：现代移动通信与物联网综合系统实验

姓 名：黄嘉欣、喻治滔

学 院：信息与工程学院

系：信息与电子工程学系

专 业：信息工程

学 号：3190102060、3190105056

指导教师：马洪庆、李培弘

2022 年 12 月 29 日

# 浙江大学实验报告

一、背景介绍

三、实验总体框架

五、实验结果与分析

二、实验目的与内容

四、实验各部分实现过程

六、总结

## 一、背景介绍

万物互联的时代，人、数据、事物通过互联网紧密地连接在一起，相互感知，无处不在。得益于庞大的感知、控制体系，小到我们的家庭，大到整个城市，我们可以实时监控各种设备的状态，也可以快捷控制它们，让每个设备成为自动化网格的一部分，真正做到智慧生活。

然而，为了实现万物互联，我们需要克服许多难题。从感知层的状态读取，到应用层的用户控制，如何完成通信，如何设计数据的流动，都需要一个严格、通用的规定或协议。作为相关专业的学子，对我们而言，了解整个系统的运作方式，自己动手去设计一个物联网系统，可以极大地提高我们的专业素养和综合能力。因此，在这次实验当中，我们将以自己的个人电脑 PC 模拟物联网网关，以 GWDataSim 软件模拟物联网沙盘设备及其服务程序，在虚拟机中安装 win7 系统作为云服务器，安装 Wamp 套件和 Mqtt 服务程序，以此完成设备状态的读取和控制任务。

## 二、实验目的与内容

- ① 写一个 python 脚本在 PC 上运行，可以订阅模拟网关中所有设备的状态并调用 Http 操作更新至数据库逻辑设备中；
- ② 写一个 python 脚本在 PC 上运行，订阅 MQTT 主题，并响应 Qt-App 发送过来的 json 数据转去控制模拟网关上面的设备；
- ③ 修改拓展老师提供的数据库触发器工程（支持数据库情况直接发布至 MQTT 服务器）或利用老师提供的数据库触发器插件，配合在虚拟机上运行的 MQTT 连接程序（上报数据库情况至 MQTT 服务器），实现数据库逻辑设备状态的发布；
- ④ 完成 Qt 程序设计，包括界面和功能，要能看到传感器数据、能通过界面操作控制控制器（主要是订阅任务 1 传导至任务三最终发布的 MQTT 主题消息并反映到界面上，同时响应界面控件的 UI 操作，向任务 2 的 MQTT 主题发布相应的消息）。

### 三、实验总体框架

本实验主要涉及 Qt 程序、模拟网关、服务器三个部分，其中，Qt 程序负责显示模拟网关中各设备的状态，并提供控制器的控制接口，服务器主要负责 MQTT 服务和数据库的运行。

系统正常工作时，PC 端将同时运行 Python 脚本和模拟网关，前者会订阅网关中各设备的状态，并通过 Http 操作调用与之对应的 PHP 服务脚本。若模拟网关中传感器和控制器的数据发生改变，Python 脚本将会接收到网关发来的更新信息，并将该信息传递给 PHP 脚本，用以更新数据库内逻辑设备的状态。由于服务器上配置了数据库触发器，当逻辑设备状态发生改变时，触发器会通过 UDP 端口输出信息，监听该端口的 Python 脚本收到后即会解析该信息并发布 MQTT 主题消息。由于订阅了该主题，此时 Qt 程序将会接收到脚本发出的更新信息，解析后即可将设备的状态显示在 UI 界面中。

当用户在 Qt 程序的 UI 界面中操作控制器时，Qt 程序将会发布相应主题的 MQTT 消息，而订阅了该主题的 Python 脚本接收到消息后，即可从中解析出用户控制的设备 id 及其状态，并向模拟网关发出指令，完成物理设备的状态控制。

### 四、实验各部分实现过程

#### ① 网关控制与订阅 Python 脚本

由于订阅网关后，设备状态一旦更新就会返回信息，而控制设备时也会有返回信息，因此，如果将读取设备状态与控制设备的 Python 脚本分成两个程序，就会导致时序上的异步，返回信息将无法被脚本解析。为了解决这个问题，我们设置了两个线程，将订阅设备和控制设备合并为一个线程，而将订阅 MQTT 消息作为第二个线程，二者通过全局变量：“设备 id，设备值”实现沟通。程序启动后，第一个线程将不断接收到模拟网关传来的设备状态更新信息，并调用 Http 通过 PHP 脚本完成数据库中逻辑设备的数据更新，注意此处控制器的设备 id 会在原值的基础上加上 100，再更新到数据库中；第二个线程将监视 MQTT 主题“qt”，其为 Qt 程序发布消息时所采用的主题，一旦接收到 json 格式的消息 {'device\_id': xx, 'device\_value': y}，即会将其转为词典并更新全局变量，进而使第一个线程向模拟网关发出指令。经过这样的处理，即可实现模拟网关输入、输出信息的有序进行。

#### ② 数据库逻辑设备状态发布 Python 脚本

通过设置数据库触发器，当数据库中逻辑设备的状态发生变化时，即会有消息通过 UDP

端口发出。此 Python 脚本运行后，将会一直监听服务器的 UDP 端口，从中读取更新后逻辑设备的状态值并转为 json 格式发布。此处发布的 MQTT 消息主题为“Trigger”，其将由 Qt 程序订阅并解析，用于 UI 界面的设备状态显示。

### ③ Qt 程序

我们的 Qt 程序是在老师提供的 libEmqtt 例程的基础上修改完成的。我们修改了程序的 UI 界面，通过 lineEdit 控件显示各个传感器的值，并使用 pushButton 作为用户控制控制器的接口。当程序开始运行后，其会自动连接服务器的 MQTT 并订阅主题“Trigger”。因此，一旦数据库逻辑设备的状态发生改变，Qt 程序就会接收到含有传感器状态值的 json 格式消息。对于此字符串，我们并未使用 json 的相关函数进行处理，而是直接从字符串中提取数字（由 qemqtttestwidget.cpp 文件中的函数 extract\_int 实现）。对于红外对射传感器和门磁传感器，由于其值不是数字，而是“true”或者“false”，我们会从字符串中检测这两个单词并解析，最后显示在 UI 界面上。当用户点击控制器的“开”/“关”按钮后，程序会自动发布对应的主题为“qt”的 MQTT 消息，由 ① 中的脚本解析并控制模拟网关中的物理设备。

## 五、实验结果与分析

如图 5.1，为 Qt 程序的初始界面，此时服务器上 MQTT 服务暂未开启，故下方状态栏显示“please connect server first!”。

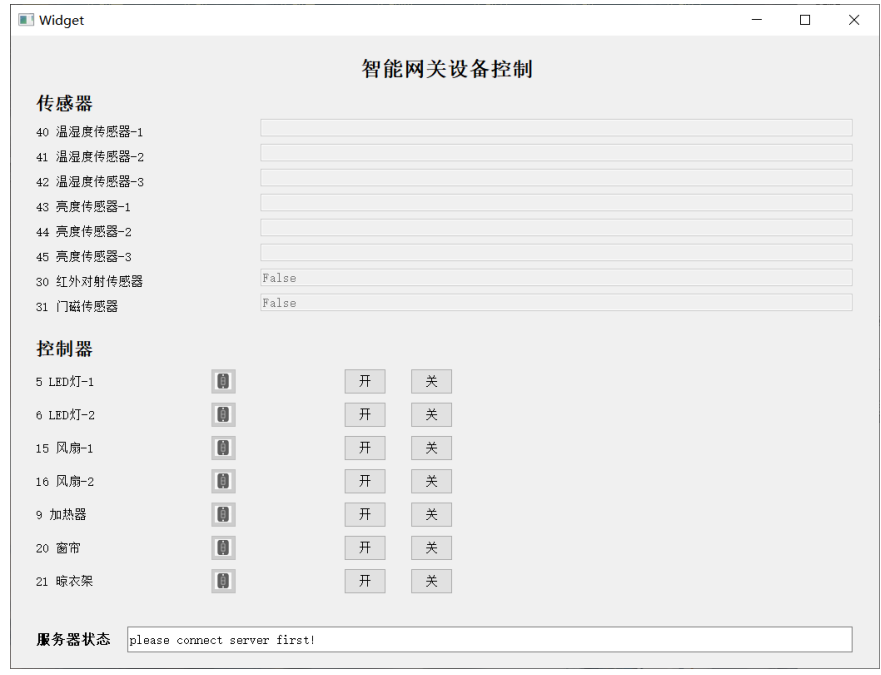


图 5.1 Qt 程序初始界面

如图 5.2，当开启服务并运行 Python 脚本后，Qt 程序将会持续接收到各个传感器的状态更新消息并显示，此时状态栏更新为“已订阅主题-Trigger”。



图 5.2 Qt 显示传感器状态

如图 5.3，当打开 LED 灯-1 时，UI 界面中相应的图标会点亮，下方状态栏将显示发布出去的 MQTT 消息主体。与此同时，数据库与模拟网关中该设备的状态也已经发生改变，如图 5.4、5.5 所示（注意逻辑设备的 id 需在原值基础上加 100）。



图 5.3 Qt 操作控制器















← T →			id	name	type	value
<input type="checkbox"/>	 编辑	 复制  删除	105	led	svController	{"value":true}
<input type="checkbox"/>	 编辑	 复制  删除	106	led	svController	{"value":false}
<input type="checkbox"/>	 编辑	 复制  删除	109	heater	svController	{"value":false}
<input type="checkbox"/>	 编辑	 复制  删除	115	fan	svController	{"value":false}
<input type="checkbox"/>	 编辑	 复制  删除	116	fan	svController	{"value":false}
<input type="checkbox"/>	 编辑	 复制  删除	120	curtain	svController	{"value":false}
<input type="checkbox"/>	 编辑	 复制  删除	121	rack	svController	{"value":false}

图 5.4 数据库控制器状态

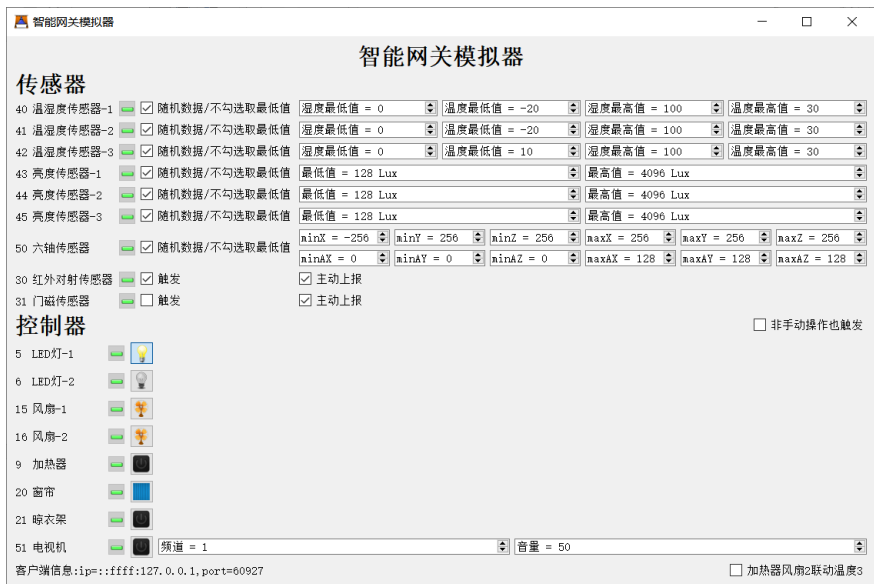


图 5.5 模拟网关控制器状态

## 六、总结

总体而言，此次实验，我们从零开始实现了一个物联网系统，利用数据库和 MQTT 服务实现了数据的流动和设备的控制，不仅对这些工具有了深入的学习，也对 json 等数据格式、Qt、Python 等更加熟练。回顾项目的完成过程，我们遇到过大大小小的困难，但所幸都成功解决，其中最让我们印象深刻的如下：

- 1、温湿度等传感器值的更新。因为在最初完成的服务器端 PHP 脚本中，我们单次只能更新一个设备的一条值属性，而温湿度等传感器的值包含有多个部分，因此在数据更新上存在一定问题。考虑到 PHP 脚本修改不存在的值属性时会自动将其添加到设备的属性中，我们最后决定采用连续更新，即对设备值的每个部分，设定一个单独的值属性，每当温湿度等传感器的状态改变时，便将其每个值都更新一次。经过这样的处理，我们就可以实现多值传感器的状态读取与显示；

2、Qt 程序 UI 界面的图片插入。根据文档，Qt 的 `pushButton` 可以设置图片作为其外观，但在代码中设置了图片的路径后其却并未正常显示。经过查阅博客，我们最后使用了 `qrc` 资源配置文件指定了需要插入的图片，终于解决了此问题。

总的来说，在整个项目过程中，我们并没有遇到太大的困难。感谢老师们的耐心指导，让我们避免了很多弯路，能够快速上手、不断适应，最后完成整个系统。当我们看到程序正确显示传感器数据、成功操控控制器时，心中所体会到的是苦尽甘来的喜悦，是收获的满足。经过这样一次练习，我们成功地将学到的知识融会贯通，并打下了该有的基础，不失为一次良好的学习体验。