**实验报告**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 学 号 | 21030031009 | | 姓 名 | 惠欣宇 | 专业班级 | | 计算机4班 | |
| 课程名称 | **物联网系统设计与开发** | | | | 学期 | | **2024年春季学期** | |
| 任课教师 | 郭忠文 | | 完成日期 | 2024年4月3日 | | 实验课时间 | | 10:25~12:15 |
| 实验名称 | | KL-W6000 采集器数据解析 | | | | | | |
| **一、实验目的及要求**  目的：通过利用 KL-W6000 通信协议对获取的传感数据进行解析，掌握字节型通信协议传感数据的解析方法，为后续的传感设备数据采集软件编程奠定基础。  要求：预习 KL-W6000 传感设备的通信协议，通过实验，熟练掌握根据 KL-W6000 传感设备通信协议进行数据解析的方法，实验过程认真记录，实验完成后撰写实验报告。  **二、实验内容及步骤**  内容：根据提供的 KL-W6000 传感设备通信协议，编写数据解析代码，解析出返回的数据。  步骤：  （1）学习 KL-W6000 传感设备的通信协议。  （2）打开仿真器桌面上的“琏雾试验系统\实验 2.2”文件夹，利用 Python 编写程序，读取 KL-W6000.txt 文件中的数据，根据 KL-W6000 通信协议对文件中的数据进行解析，将解析后的数据显示在屏幕上，KL-W6000 数据显示格式示例如图1所示。    **图1** KL-W6000 数据显示格式示例  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 代码实现思路 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  首先我们先来导入代码实现所需要的包，如下图2。其中 tkinter 用于 GUI 界面，datetime 用于获取当前时间，filedialog 和 ttk 是 tkinter 的子模块，用于文件对话框和其他 GUI 组件的创建，PIL（Pillow）用于图像处理。    **图2** 导入相关包  **代码主要分为三个部分：数据解析、数据显示和数据保存。**  **数据解析详细步骤**：  这里我们主要基于KL-W6000通信协议中的相应字段解释来实现，返回数据解析表如下图3、4。    **图3** 数据解析表1    **图4** 数据解析表2  下面依据实验手册对各个字段作以解释：  ①地址：设置的网由的地址；  ②功能码：命令的类型，查询命令的功能码为 04；  ③数据字节数：返回寄存器数据的字节数，一个寄存器占两个字节，读取 4 个寄存器，返回 8 个字节的数据；  ④寄存器 1 数据：此处返回的是烟感、火感、红外线、水浸四路开关量传 感器的状态，其中第二个字节的前 4 位代表每一个的开关状态，如 30 转换为二进制为 00110000，前四位 0011，第一位是烟感状态，第二位是火感状态，都是 “关”状态，无报警；第三位是红外线状态，第四位是水浸的状态，都是“开”状态，有报警，说明机房有人出入和漏水发生；  ⑤寄存器 2 数据：在此无实际意义；  ⑥寄存器 3 数据：返回温度传感器的数据，十六进制 00DB 转换为十进制 219，并一位小数点，温度为 21.9℃；  ⑦寄存器 4 数据：返回湿度传感器的数据，十六进制 01C5 转换为十进制 453，并一位小数点，湿度为 45.3%；  ⑧CRC 校验：从开始到 CRC 校验前的字节的 CRC16 校验码。  据此我们来进行代码实现（**parse\_data**方法）：  我们此时假定已经将数据读入为data（数据显示部分会介绍数据读入部分），并且这个data是一个二进制序列的列表，从传入的data列表中取出索引为4的元素作为switch\_status\_byte，这个元素是一个字符串，表示开关状态的字节。   * 从switch\_status\_byte字符串的第一个字符中取出一个整数，用于表示烟感状态。这里使用int()将字符转换为整数。 * 从switch\_status\_byte字符串的第二个字符中取出一个整数，用于表示火感状态。这里使用int()将字符转换为整数。 * 从switch\_status\_byte字符串的第三个字符中取出一个整数，用于表示红外状态。这里使用int()将字符转换为整数。 * 从switch\_status\_byte字符串的第四个字符中取出一个整数，用于表示水浸状态。这里使用int()将字符转换为整数。   然后将data列表中索引为7和8的两个元素拼接成一个二进制字符串，使用int()将其转换为整数。这个整数表示环境温度，除以10得到实际的温度值。同理，将data列表中索引为9和10的两个元素拼接成一个二进制字符串，然后转换为整数。这个整数表示环境湿度，除以10得到实际的湿度值。  最后创建一个格式化的字符串result，包含当前的日期和时间，以及解析出的各种状态和环境数据，然后将其返回。  **parse\_data**方法如下图5：    **图5** **parse\_data**方法具体实现  **数据显示详细步骤**：   * **Part 1**   首先是**show\_result**方法的实现，这里主要处理了选择文件以及数据插入的交互逻辑，这里我们效仿上一个实验的流程：   * 首先打开一个文件对话框，选择一个文本文件（.txt），并返回选择的文件路径 file\_path。如果file\_path不为空，则使用with语句打开选中的文件，使用只读模式（'r'），并将文件对象赋值给file变量。 * 接下来读取文件的内容，并使用split()方法以空格作为切分单元将内容分割成一个列表，存储在data变量中。 * 清空result\_text（解析数据文本框）文本框的内容，同时将result\_text文本框的状态设置为可编辑，这里是为了解决在已经解析了一份数据文件之后还需要解析下一份数据时文本框复用的问题。 * 清空source\_data\_text（源数据文本框）文本框的内容，同时将source\_data\_text文本框的状态设置为可编辑，这里是为了解决在已经解析了一份数据文件之后还需要解析下一份数据时文本框复用的问题。   下面就可以进行本次数据解析内容的插入了：   * 首先在source\_data\_text文本框插入一行标题，表示这是源数据。将整个data列表（源文件内容）转换成一个字符串，并插入到source\_data\_text文本框中。将source\_data\_text文本框的状态设置为只读，修改源数据。 * 然后遍历data列表中的每个元素，将其转换为二进制格式，并确保每个二进制字符串都是8位（前面补零）。 * 调用parse\_data方法，依据协议进行数据解析，并将结果存储在result变量中。清空result\_text文本框的内容，同时将result\_text文本框的状态设置为可编辑。随后将解析后的结果字符串插入到result\_text文本框中，再将result\_text文本框的状态设置为只读。 * 最后为了配合保存数据部分的逻辑，这里将save\_button（另存为）按钮的状态设置为可用，允许用户保存解析后的数据。   **show\_result**方法如下图6：    **图6** **show\_result**方法具体实现   * **Part 2**   接下来我们对对话框部分进行完善，初始化一个Tkinter窗口，并为这个对话框添加了一张背景图片。Tkinter窗口中包含两个按钮（选择文件和另存为），分别用于打开文件选择对话框和保存数据，窗口下方还有两个文本框，左侧为文本框用于显示源数据，右侧文本框用于显示解析后的数据，同时这两个文本框分别配有一个滚动条。代码如下图7、8、9、10、11所示。    **图7** Tkinter窗口初始化    **图8** 添加Tkinter窗口背景    **图9** Tkinter窗口按钮部分    **图10** 源数据文本框和滚动条部分    **图11** 解析数据文本框和滚动条部分  **数据保存详细步骤**：   * 点击“另存为”按钮时，调用save\_data函数，使用filedialog.asksaveasfilename()打开保存文件对话框，可以选择保存位置和文件名。 * 选择后，将解析数据文本框中的内容（即解析后的数据）写入到指定的文件中。   **save\_result函数**代码如下图12所示。    **图12** **save\_result**函数具体实现  **这里需要注意一点**：“另存为”按钮起初需要设置为不可点击状态，因为最开始没有数据解析内容可以存储，只有在选择文件并解析之后，“另存为”按钮才进入可点击状态。  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 运行效果展示 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  如图13为未选择数据源的运行效果，图14为选择则数据源后的数据解析测试结果。    **图13** KL-W6000数据解析器未选择数据源的运行效果    **图14** KL-W6000数据解析器解析测试结果  在实验的最后我使用Python的pyinstaller工具将本次KL-W6000数据解析器项目打包成一个.exe可执行文件，同时使用另存为功能将所给的测试数据的解析结果输出到一个.txt文件中。  如下图15所示，图15中的第1个文件为实验所需要的背景图片文件，第2个文件为实验所给的源数据文件。第3个文件为KL-W6000数据解析器项目打包成的.exe可执行文件，双击即可展示图13的效果。第4个文件为经解析后并将解析结果另存的.txt文件，我们可以看看这个文件的内容，如图16。    **图15** 项目打包及数据文件    **图16** 使用另存功能存储的解析结果文件  **三、心得总结**  本次实验主要实现了KL-W6000采集器数据解析，我不仅加深了对物联网系统中数据采集和解析的理解，而且提升了我的编程能力和问题解决技巧。在实验过程中，我学习了KL-W6000传感设备的通信协议，并成功实现了一个数据解析器，用于读取和解析传感器数据，通过编写代码，不仅巩固了课上学习的理论知识，还提高了自己的动手能力和问题解决能力，通过实际操作，我对物联网系统中数据采集和处理的流程有了更加清晰的认识，这对于我未来在该领域的深入学习和研究具有重要意义。  在进行数据解析的过程中，我们首先要对通信协议要有深入的理解。KL-W6000传感设备的通信协议规定了数据的格式和解析规则，需要熟悉并理解解析规则，才能正确地从原始数据中提取有用信息。对于任何物联网设备，理解其通信协议是进行有效数据交互的前提。  数据解析过程中，需要将十六进制数据转换为实际的物理量（如温度和湿度），因此需要进行进制转换。结合Python中的字符串处理方法，最终从字符串中提取有效数值。  在GUI界面的设计上，我尝试了多种布局和组件，以实现用户友好的交互体验。通过不断调整和优化，最终得到了一个简洁明了且功能齐全的界面。  最后，我还使用Python的pyinstaller工具将本次KL-W6000数据解析器项目打包成一个.exe可执行文件，便于使用。  通过完成本次KL-W6000数据解析实验，提高了我的在数据处理方面的编程能力，也锻炼了我的逻辑思维和问题解决能力。通过这次实验，我更有兴趣面对物联网系统中的数据解析任务，并且为未来物联网课程的学习打下一定的基础。  **遇到的问题及解决方法总结：**  1. 数据解析  **问题：**在解析数据时，最初没有考虑到二进制字符串的补零问题，导致数据解析不准确。parse\_data方法需要的data列表的每一个元素都为8位，但转换为二进制时有些数据缺少高位的0。如下图17调试器输出内容所示：    **图17** 二进制转换缺少高位0的问题  **解决方法：**通过添加一个补零函数，确保每个二进制字符串都是8位，从而保证了数据解析的准确性，如下图18。    **图18** 解决数据解析问题的代码块  解决后的调试器输出内容为如下图19所示，说明成功解决该问题，并且数据格式可以输入进parse\_data方法。    **图19** 二进制转换高位缺零问题已解决  **四、通信协议设计**  最早系统学习通信协议是在上个学期计算机网络课程中，根据上学期所学以及本课程上课所讲内容一个有效的通信协议需要确保数据的可靠传输、错误检测和纠正、安全性以及效率。下面我总结了几个设计通信协议时需要注意的几个关键方面：   * 数据格式：确定如何表示和编码数据，例如使用二进制、JSON、XML等。 * 数据传输：确定数据如何被发送和接收，包括帧格式、分组和流控制。 * 错误检测和纠正：设计用于检测和修复传输中出现的错误的机制，如奇偶校验、CRC（循环冗余校验）等。 * 安全性：考虑数据加密、身份验证和授权机制。 * 连接管理：处理连接的建立、维护和关闭。 * 性能和效率：优化数据传输的速度和带宽使用。 * 版本管理：为将来的协议升级留出空间。   **下面是我个人的一个设计方案：**   * **数据格式**   使用UTF-8编码的文本数据，支持多语言和特殊字符。   * **帧格式** * 起始标志（1字节）：固定为0x01。 * 版本号（1字节）：用于协议的版本管理。 * 帧类型（1字节）：定义帧的类型，如SYN、ACK、FIN、DATA等。 * 数据长度（2字节）：表示后续数据的长度。 * 数据（变长）：文本数据。 * 校验和（2字节）：使用更强大的CRC-32进行错误检测。 * 结束标志（1字节）：固定为0x04。 * **错误检测和纠正**   使用CRC-32或其他检错纠正校验码进行错误检测和纠正。   * **安全性**   加密：使用AES-256对数据进行加密。  身份验证：使用公钥加密和数字签名进行身份验证。   * **连接管理**   连接建立：使用三次握手机制。   * 客户端发送SYN帧。 * 服务器回应SYN-ACK帧。 * 客户端发送ACK帧确认连接。   连接关闭：使用四次挥手机制。   * 客户端发送FIN帧。 * 服务器回应ACK帧。 * 服务器发送FIN帧。 * 客户端回应ACK帧。   心跳机制：定期发送心跳帧以维持连接状态。  超时处理：设定超时时间并处理超时情况，如重新发送或关闭连接。   * **性能和效率**   流控制：使用滑动窗口机制进行流控制。  拥塞控制：使用TCP的拥塞控制算法进行拥塞管理。（慢开始、拥塞避免、快重传）  优化：使用数据压缩算法（如LZ77）减少数据传输量。   * **版本管理**   在帧格式中添加版本字段以支持未来的协议升级，并提供版本兼容性处理机制。 | | | | | | | | |