**中国海洋大学计算机科学与技术学院**

实验报告

**姓名：惠欣宇 年级：2021级 专业：计算机科学与技术**

**科目：物联网系统设计与开发 题目：光照度采集及控制交互编程**

**实验时间：2024年 5月 23日 实验教师：郭忠文**

**目录**

[一、实验目的及要求 2](#_Toc167496616)

[1. 实验目的 2](#_Toc167496617)

[2. 实验要求 2](#_Toc167496618)

[二、实验内容 2](#_Toc167496619)

[三、实验设备 2](#_Toc167496620)

[四、通信协议 3](#_Toc167496621)

[五、实验步骤 4](#_Toc167496622)

[六、报告要求 10](#_Toc167496623)

[1. 简要叙述实验结果 10](#_Toc167496624)

[2. 把打印采集程序和控制程序源代码和存储数据的文本文件附在实验报告中 11](#_Toc167496625)

[3. 总结编写数据采集软件及软件调试的关键步骤 12](#_Toc167496626)

[4. 写出调试过程遇到的主要问题和解决方法 13](#_Toc167496627)

[七、实验总结 14](#_Toc167496628)

一、实验目的及要求

1. 实验目的

该实验为串口通信实验，计算机通过串口采集室内光照度传感数据，进行解析处理，根据光照度大小，通过控制控制模块灯泡开关。了解真实设备与计算机互联的方式，培养学生编程能力。通过编写传感设备采集数据及控制软件，培养利用串行通信接口采集传感数据以及控制设备的编程能力，为物联网系统综合数据采集软件的设计与开发打下基础。

2. 实验要求

根据实验内容，预先编写数据采集及控制软件，实验过程中认真调试、思考 和记录，掌握利用软件自动控制设备的方法，撰写实验报告。

二、实验内容

通过笔记本电脑，利用USB-RS232转换器连接光照度传感器，利用USB-RS485 转换器连接YF-60 控制模块，YF-60 控制模块连接灯泡，编写数据采集及控制程 序，通过串行接口获取光照度传感数据，把解析后的光照度数据写入笔记本电脑 桌面文本文件，文件以姓名加学号命名，每次采集的数据在文件中占据一行，每 行的开头为采集数据的时间格式为“YYYY-MM-DD,hh:mm:ss”，如：2017-03-26， 15:28:32。不同数据项以“;”号分隔。当解析后光照度数据小于5000Lux时，通 过控制模块控制灯泡开启，当光照度大于10000Lux时，控制灯泡关闭。

三、实验设备

（1）YF-60 控制模块一个；

（2）灯泡一个；

（3）直流24V电源两个；

（4）USB-RS485 转换器一个；

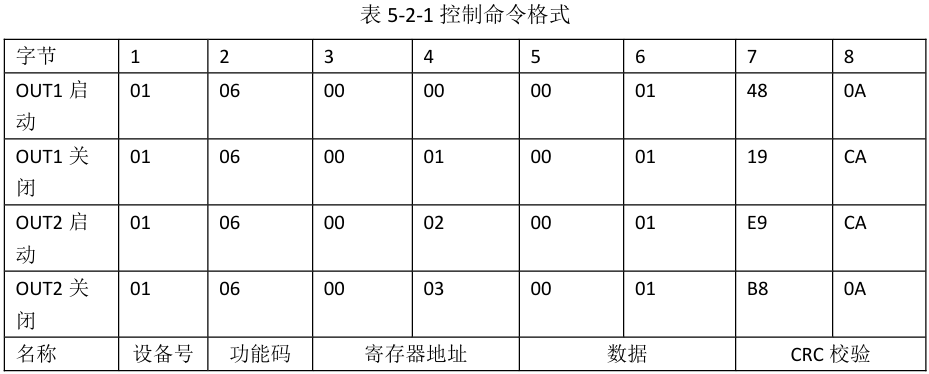
（5）USB-RS232 转换器一个；

（6）光照度传感器一个；

（7）笔记本电脑一台。

四、通信协议

（1）YF-60 控制模块控制指令见表5-2-1。



2）室内光照度传感器量程是 0-65535Lux，光照度传感设备数据获取命令 为:01 03 00 03 00 02 34 0B，接收到命令后会返回测试数据，如：01 03 04 00 09 01 F4 2A 26，具体数据解析内容见表5-2-2和续表5-2-2。

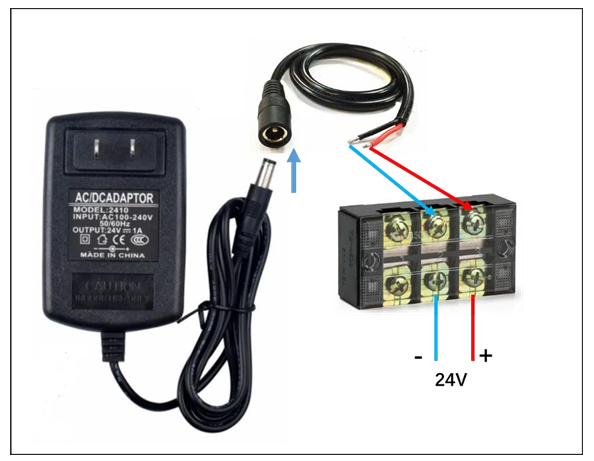
表格

描述已自动生成

传感器返回数据为0x0009 和 0x01F4 转换为十进制为 9 和 500，照度值为 500\*10+9=5009。

五、实验步骤

（1）实验系统提供了两个 24V 电源适配器，为了进行电源线连接，根据电 源线引出线路，如图5.2.1所示，建立两组电源引出电路。注意：线路全部连接 完毕，检查线路正确后再通电。



**图5.1.1** 电源线引出电路

（2）光照度传感器外观结构如图5.2.2所示，笔记本电脑与光照度传感器硬 件连线如图5.2.3 所示，从该图可以看出，光照度传感器的信号线是根据颜色区 分，其中信号地和电源地共用GND信号线。USB-RS232转换器外观结构如图5.2.4 所示。

电子设备

描述已自动生成

**图5.2.2** 光照度传感器外观结构

图示, 示意图

描述已自动生成

**图5.2.3** 笔记本电脑与数据采集模块硬件连线



**图5.2.4** USB-RS232 转换器外观结构

（3） YF-6 控制模块硬件线路连接如图5.2.5所示，利用笔记本电脑通过USB RS485 转换器连接 YF-60 控制模块，YF-60 控制模块连接被控对象灯泡及 24V 电 源。USB-RS485 转换器外观结构如图5.2.6所示，其中的5V电源在连接485接口 设备较多，电路驱动能力不足时需要连接，连接设备数量较少时可以不接5V电 源。等奥外观如图5.2.7所示。

图示

描述已自动生成

**图5.2.5** YF-60 设备硬件线路连接

图示, 工程绘图

描述已自动生成

**图5.2.6** USB-RS485 转换器外观结构



**图5.2.7** 灯泡外观

（4）打开计算机桌面上的“琏雾实验系统\工具”文件夹中的串口助手软 件进行通信调试，确保硬件通信线路正常。

（5）对根据实验内容编写的数据采集及控制程序进行调试。

（6）通过USB 转串口连接线连接光感度传感器，当解析后光照度数据小于 5000Lux 时，通过YF-60 控制模块控制灯泡开启，当光照度小于等于5000Lux 时，控制灯泡运行关闭。

我们要模拟上述温度数据采集以及设备控制的交互过程，这里我采用Python语言来实现这个模拟过程：

首先我们导入相应的包，分别有serial、time和datetime。serial用来进行串口通信的模拟，time用来进行程序与实验板的时间同步，datetime用来向目标文件写入当前时间。

1. import serial
2. import time
3. from datetime import datetime

然后设置串口参数并打开串口，这里我们需要检查设备管理器中对应的串口号是多少，这里我是COM9，并配置波特率为9600。

1. # 打开串口连接
2. ser = serial.Serial('COM9', 9600, timeout=1)

程序向实验板的发送命令函数send\_sensor\_command() 的实现。这个函数的目的是向光照度传感器发送一个获取光照度数据的命令，并读取传感器的响应数据。具体来说，函数首先定义一个代表获取光照度数据命令的十六进制字符串 '02 03 00 03 00 02 34 38'，然后使用 bytes.fromhex(hex\_command.replace(' ', '')) 方法将该字符串转换为字节数组。接下来，使用 ser.write(command\_bytes) 方法将命令字节数组发送到传感器，并使用 ser.read(100) 方法读取传感器的响应数据，读取 9 字节。最后，函数返回读取到的响应数据。

1. # 发送获取光照度数据的命令
2. def send\_sensor\_command():
3. hex\_command = '02 03 00 03 00 02 34 38'  # 定义传感器命令的十六进制字符串
4. command\_bytes = bytes.fromhex(hex\_command.replace(' ', ''))  # 转换为字节数据
5. ser.write(command\_bytes)  # 向传感器发送命令
6. response\_data = ser.read(9)  # 读取传感器返回的数据
7. return response\_data  # 返回读取的数据

解析光照数据的函数parse\_illumination()的实现。这个函数的功能是解析传感器返回的光照度数据，并将其转换为一个可读的光照度值（单位：Lux）。函数首先检查传感器返回的数据长度是否小于 9 字节，如果数据长度不合法，就抛出 ValueError 异常。接着，从数据中提取第 6 到第 10 个字节（包含边界）的十六进制字符串，并将其转换为整数，作为低位数据 data\_ge；然后提取第 10 到第 14 个字节（包含边界）的十六进制字符串，并将其转换为整数，作为高位数据 data\_ten。最后，计算最终的光照度值 parsed\_illumination = data\_ge + data\_ten \* 10，并返回该光照度值。

1. # 解析光照度数据
2. def parse\_illumination(data):
3. if len(data) < 9:  # 检查返回数据的长度是否合法
4. raise ValueError("返回数据长度不合法")  # 数据长度不合法，抛出异常
5. data\_ge = int(data[6:10], 16)  # 提取并解析低位光照度数据
6. data\_ten = int(data[10:14], 16)  # 提取并解析高位光照度数据
7. parsed\_illumination = data\_ge + data\_ten \* 10  # 计算光照度值
8. return parsed\_illumination  # 返回解析后的光照度值

控制灯泡的开关状态的函数control\_bulb() 的实现。这个函数根据传入的状态参数 ('on' 或 'off') 控制灯泡的开关状态。首先，函数检查传入的状态参数 state：如果 state 是 'on'，则定义灯泡启动命令 '03 06 00 00 00 01 49 E8'；如果 state 是 'off'，则定义灯泡关闭命令 '03 06 00 01 00 01 18 28'；如果 state 不是 'on' 或 'off'，函数抛出 ValueError 异常。接下来，使用 bytes.fromhex(command.replace(' ', '')) 方法将命令字符串转换为字节数组，然后使用 ser.write(command\_bytes) 方法将命令字节数组发送到设备，并使用 ser.read(8) 方法读取设备返回的数据，确保命令被正确处理，防止影响下一次光照度数据的解析。

1. # 控制灯泡
2. def control\_bulb(state):
3. if state == 'on':  # 如果状态是 'on'
4. command = '03 06 00 00 00 01 49 E8'  # 定义灯泡启动命令
5. elif state == 'off':  # 如果状态是 'off'
6. command = '03 06 00 01 00 01 18 28'  # 定义灯泡关闭命令
7. else:
8. raise ValueError("Invalid state")  # 状态无效，抛出异常
9. command\_bytes = bytes.fromhex(command.replace(' ', ''))  # 转换为字节数据
10. ser.write(command\_bytes)  # 向灯泡发送命令
11. ser.read(8)  # 读取设备返回的数据，防止影响下一次对光照数据的解析

将光照强度数据记录到文件中的函数save() 的实现。这个函数将当前的光照度数据记录到一个文件中。具体而言，函数使用 open('惠欣宇-21030031009.txt', 'a') 以追加模式打开一个名为 '惠欣宇-21030031009.txt' 的文件，然后获取当前的时间，并将其格式化为字符串 "%Y-%m-%d,%H:%M:%S"。接着，使用 file.write(f"{current\_time}; {result}lux\n") 方法将当前时间和光照度数据写入文件，每条记录占一行，便于数据的存储和分析。

1. # 写入数据到文件
2. def save(result):
3. with open('惠欣宇-21030031009.txt', 'a') as file:  # 以追加模式打开文件
4. current\_time = datetime.now().strftime("%Y-%m-%d,%H:%M:%S")  # 获取当前时间
5. file.write(f"{current\_time}; {result}lux\n")  # 写入时间和光照度值

下面就是主程序的循环部分，负责不断地获取光照度数据，控制灯泡状态，并记录光照度数据到文件中。首先，通过 try 块开启一个无限循环 while True，持续执行内部操作，直到发生中断条件。代码中调用 send\_sensor\_command() 函数向光照度传感器发送获取数据的命令，并接收传感器的响应数据。为了确保数据已准备好，程序会暂停 1 秒。接收到的响应数据转换为十六进制字符串格式 response.hex() 以便解析。如果有响应数据，则进入嵌套的 try 块，调用 parse\_illumination(response) 函数解析数据并计算光照度值，随后打印当前光照度值，并调用 save(illumination) 函数将数据记录到文件中。根据光照度值判断灯泡的状态：如果光照度小于 5000 Lux，调用 control\_bulb('on') 打开灯泡并打印“打开灯泡”；如果光照度大于或等于 5000 Lux，调用 control\_bulb('off') 关闭灯泡并打印“关闭灯泡”。解析过程中若抛出 ValueError 异常，则捕捉并打印错误信息。程序每秒钟采集一次数据。捕捉到键盘中断信号（如按下 Ctrl+C）时，通过 except KeyboardInterrupt 捕捉异常并打印“程序终止”。无论是否发生异常，都会执行 finally 块中的代码，调用 ser.close() 关闭串口连接，确保资源正确释放。这段代码通过循环不断获取光照度数据，根据数据控制灯泡的开关，并记录数据到文件中，确保程序在中断时能够安全地关闭串口连接。

1. # 主程序循环
2. try:
3. while True:
4. response = send\_sensor\_command()  # 发送获取光照度数据的命令
5. time.sleep(1)  # 等待设备响应 1 秒
6. response = response.hex()  # 将响应数据转换为十六进制字符串
7. if response:  # 如果有响应数据
8. try:
9. illumination = parse\_illumination(response)  # 解析光照度数据
10. print(f"当前光照度: {illumination} Lux", end=" ")  # 打印光照度
11. save(illumination)  # 保存光照度数据到文件
12. if illumination < 5000:  # 如果光照度小于 5000
13. control\_bulb('on')  # 打开灯泡
14. print("打开灯泡")  # 打印灯泡状态
15. elif illumination >= 5000:  # 如果光照度大于或等于 5000
16. control\_bulb('off')  # 关闭灯泡
17. print("关闭灯泡")  # 打印灯泡状态
18. except ValueError as e:  # 捕捉数据解析异常
19. print(f"分析错误: {e}")  # 打印错误信息
20. time.sleep(1)  # 每秒采集一次数据
21. except KeyboardInterrupt:  # 捕捉到键盘中断信号
22. print("程序终止")  # 打印程序终止信息
23. finally:
24. ser.close()  # 关闭串口连接

下面给出控制台输出的解析数据的过程截图，如图5.2.8所示。

图形用户界面, 文本, 应用程序

描述已自动生成

**图5.2.8** 光照度数据控制台运行结果

六、报告要求

1. 简要叙述实验结果

本次实验的主要目标是通过串口通信实现对室内光照度的实时监测，并根据光照度的大小自动控制灯泡的开关。实验结果显示，我们成功开发了一个数据采集和控制软件，该软件能够与光照度传感器和YF-60控制模块无缝协作。

在实验过程中，我们首先验证了数据采集模块与传感器之间的通信。通过发送特定的命令，我们能够准确地从传感器获取光照度数据。数据显示，传感器能够提供0到65535Lux范围内的精确读数，这为我们后续的数据分析和设备控制提供了可靠的基础。

我们实现了根据光照度数据自动控制灯泡的功能。当室内光照度低于预设阈值5000Lux时，我们的程序能够自动发送控制指令至YF-60模块，成功开启灯泡以补充光线。相反，当光照度超过5000Lux时，程序同样能够自动关闭灯泡，以避免过度照明。

最后我们将采集到的光照度数据以及控制指令的执行情况记录在了以学生姓名和学号命名的文本文件中。文件记录了每次数据采集的时间戳和对应的光照度值，便于后续的数据分析和验证。

2. 把打印采集程序和控制程序源代码和存储数据的文本文件附在实验报告中

* **采集并解析的函数与控制函数**

采集并解析的函数：

1. # 发送获取光照度数据的命令
2. def send\_sensor\_command():
3. hex\_command = '02 03 00 03 00 02 34 38'  # 定义传感器命令的十六进制字符串
4. command\_bytes = bytes.fromhex(hex\_command.replace(' ', ''))  # 转换为字节数据
5. ser.write(command\_bytes)  # 向传感器发送命令
6. response\_data = ser.read(9)  # 读取传感器返回的数据
7. return response\_data  # 返回读取的数据
8. # 解析光照度数据
9. def parse\_illumination(data):
10. if len(data) < 9:  # 检查返回数据的长度是否合法
11. raise ValueError("返回数据长度不合法")  # 数据长度不合法，抛出异常
12. data\_ge = int(data[6:10], 16)  # 提取并解析低位光照度数据
13. data\_ten = int(data[10:14], 16)  # 提取并解析高位光照度数据
14. parsed\_illumination = data\_ge + data\_ten \* 10  # 计算光照度值
15. return parsed\_illumination  # 返回解析后的光照度值

控制函数：

1. # 控制灯泡
2. def control\_bulb(state):
3. if state == 'on':  # 如果状态是 'on'
4. command = '03 06 00 00 00 01 49 E8'  # 定义灯泡启动命令
5. elif state == 'off':  # 如果状态是 'off'
6. command = '03 06 00 01 00 01 18 28'  # 定义灯泡关闭命令
7. else:
8. raise ValueError("Invalid state")  # 状态无效，抛出异常
9. command\_bytes = bytes.fromhex(command.replace(' ', ''))  # 转换为字节数据
10. ser.write(command\_bytes)  # 向灯泡发送命令
11. ser.read(8)  # 读取设备返回的数据，防止影响下一次对光照数据的解析

主程序中调用上面函数的代码如下，这里进行开关逻辑的实现：

1. if illumination < 5000:  # 如果光照度小于 5000
2. control\_bulb('on')  # 打开灯泡
3. print("打开灯泡")  # 打印灯泡状态
4. elif illumination >= 5000:  # 如果光照度大于或等于 5000
5. control\_bulb('off')  # 关闭灯泡
6. print("关闭灯泡")  # 打印灯泡状态

* **文本文件内容**

2024-05-23,16:26:03; 520lux

2024-05-23,16:26:07; 521lux

2024-05-23,16:26:11; 521lux

2024-05-23,16:26:15; 15757lux

2024-05-23,16:26:19; 24315lux

2024-05-23,16:26:23; 20497lux

2024-05-23,16:26:27; 517lux

2024-05-23,16:26:31; 519lux

2024-05-23,16:26:35; 519lux

2024-05-23,16:26:39; 1484lux

2024-05-23,16:26:43; 25173lux

2024-05-23,16:26:47; 28050lux

2024-05-23,16:26:51; 29983lux

2024-05-23,16:26:55; 517lux

2024-05-23,16:26:59; 517lux

2024-05-23,16:27:03; 517lux

2024-05-23,16:27:07; 1016lux

2024-05-23,16:27:11; 23913lux

2024-05-23,16:27:15; 25346lux

2024-05-23,16:27:19; 12452lux

3. 总结编写数据采集软件及软件调试的关键步骤

编写数据采集软件及调试的关键步骤包括：

* 确定通信协议和数据格式。
* 设计并实现数据采集函数，用于发送命令并接收传感器数据。
* 设计并实现数据解析函数，将传感器返回的原始数据转换为可读的光照度值。
* 设计并实现控制函数，根据光照度值控制灯泡的开关状态。
* 在下一次读取数据时需要将上一次执行操作的命令的返回数据从串口中读出。
* 编写主程序循环，实现数据的持续采集、解析、控制和记录。
* 调试过程中，需要注意检查串口通信是否正常，数据解析是否正确，以及控制命令是否有效。

4. 写出调试过程遇到的主要问题和解决方法

* **问题和解决方法1**

**问题：**在我们编辑好通信代码后，进行数据采集与解析，这个时候发现实验板并不能按照预期期望做出相应的动作响应，也就是说，当测得的温度大于10摄氏度时实验板的风扇不开启。

**解决方法：**经过研究后，我们首先在控制台输出每一次实验板的返回数据，这里发现程序只有第一次收到的返回数据是9字节的，而后续收到的返回数据都是17字节的。在这17字节中我们发现后面9个字节是有效的温度返回数据，而前面8个字节看似是无效的。这里我们想到是否向实验板发送打开或关闭灯泡的命令后实验板会有返回数据。这里我们通过串口助手来进行验证。如图5.2.9、5.2.10所示，向实验板发送灯泡打开命令后，实验板会通过串口向我们返回一个8字节的数据，这个数据会缓存在串口中，灯泡关闭命令也同理。

图形用户界面, 文本, 应用程序

描述已自动生成

**图5.2.12** 灯泡打开命令的数据返回

图形用户界面, 文本, 应用程序

描述已自动生成

**图5.2.13** 灯泡关闭命令的数据返回

那么我们的问题就迎刃而解，只需要将每一次执行开关风扇的命令之后的返回数据从串口中读出即可，解决部分的代码高亮给出。

1. # 控制灯泡
2. def control\_bulb(state):
3. if state == 'on':  # 如果状态是 'on'
4. command = '03 06 00 00 00 01 49 E8'  # 定义灯泡启动命令
5. elif state == 'off':  # 如果状态是 'off'
6. command = '03 06 00 01 00 01 18 28'  # 定义灯泡关闭命令
7. else:
8. raise ValueError("Invalid state")  # 状态无效，抛出异常
9. command\_bytes = bytes.fromhex(command.replace(' ', ''))  # 转换为字节数据
10. ser.write(command\_bytes)  # 向灯泡发送命令
11. ser.read(8)  # 读取设备返回的数据，防止影响下一次对光照数据的解析

七、实验总结

在完成“光照度采集及控制交互编程”实验的过程中，我深刻体会到了物联网系统中数据采集和设备控制的重要性，并掌握了串口通信技术的实际应用。通过编写和调试Python程序，我不仅锻炼了自己的编程能力，还提高了解决实际问题的能力。实验中，我成功实现了软件与光照度传感器及YF-60控制模块的通信，并根据光照度自动控制了灯泡的开关，这验证了软件设计的有效性。在调试过程中，我遇到了设备响应不符合预期的问题，但通过仔细分析和检查，我发现并解决了串口通信中数据读取的问题。本次实验让我认识到了理论知识与实践操作相结合的重要性。在实验前需要熟悉实验设备和通信协议，注意代码的模块化和异常处理，以提高代码的可读性和可维护性。同时，遇到问题时要耐心分析，多与同学交流，这有助于快速找到解决方案。总的来说，这次实验是一次宝贵的学习经历，对我的技术能力和问题解决能力的提升都有着重要的影响。