计算机网络课程设计报告

题目 煤矿井下安全生产监测系统协议设计与实现

院、系 计算机与控制工程学院

专　业 计算机科学与技术

班　级 计 221-2

姓　名 杨文彬

学　号 202258501209

指导教师 侯永超

2024 年　12 月　18 日

目　　录

1 项目描述

The underground production environment in coal mines is harsh, and to ensure the safety of workers, it is essential to monitor various environmental factors in real-time. The monitoring data can be classified into two categories: the first category consists of analog quantities, such as methane, gas, oxygen, carbon monoxide, carbon dioxide, and temperature (expressed in numerical values); the second category consists of binary quantities, such as ventilation fans, equipment operational status, and power supply sensors (expressed as 0 or 1, where 0 represents a disconnection and 1 represents a connection).

The system is divided into an upper-level computer (host) and a lower-level embedded system (field device) connected via network communication. The lower-level device is connected to various sensors and is equipped with multiple relays (input/output ports, capable of controlling the power supply to sensors). Sensor data is uploaded to the upper-level computer at regular intervals (every 5 seconds), and if there is a change in sensor values, it is immediately uploaded. Upon receiving the data from the lower-level system, the upper-level computer displays the values or statuses of the sensors and, based on predefined control rules, transmits control data to the lower-level device to manage the operation of the sensors.

煤矿井下生产环境恶劣，为保障井下人员的安全，需要实时对井下各种环境因素进行监控，监控数据主要分为两大类，第一类为模拟量：如甲烷，瓦斯，氧气，一氧化碳、二氧化碳、温度等（结果以数值表示），第二类为开关量：例如通风风机、设备运行状态、馈电传感器等（结果以 0\1 表示，代表两个状态，0 代表断开状态，1 代表接通状态）。

系统分为上位机（计算机）和下位机（嵌入式系统），通过网络通信。下位机连接各种传感器，并设置多个继电器（进出端口，可实现对传感器的通断电），并将传感器数据定时上传到上位机（每 5 秒上传一次；如果传感器数值发生变化，则立即上传）。上位机收到下位机数据后，要显示收到的各个传感器的数值或者状态，并且根据控制规则向下位机传输控制数据，从而对各个传感器进行控制。

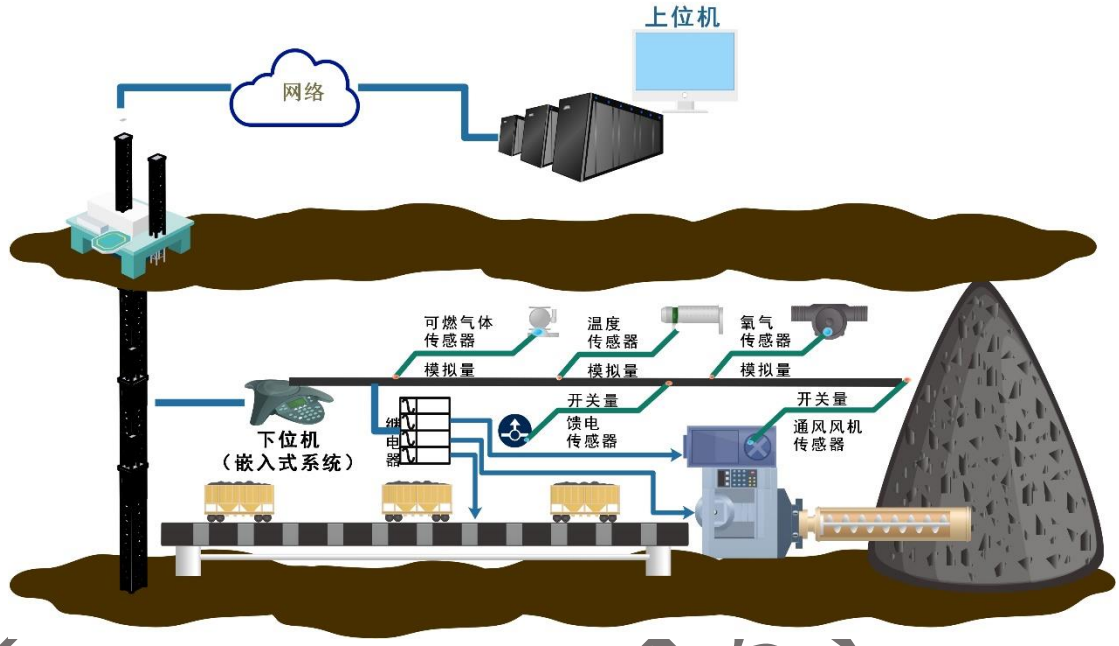

图 1 煤矿井下安全生产监测

2 任务分析

下位机可采集到甲烷检测传感器、温度测量传感器、氧气检测传感器的模拟量数据，以及主备两台风机开关状态；井下开采设备的馈电传感器开关状态的开关量。下位机连接了各继电器，可控制井下开采设备、风机设备的通电。

1) 井下通风风机分为主备，两者不能同时处于关闭状态。若上位机检测到两个通风风机同时为关闭状态，需要立即报警，不可自动解除。设置手动解除报警机制，下位机必须输入解除报警指令，才能解除报警恢复生产。

2) 甲烷浓度超过预警值时，通知下位机相应的继电器工作（开采设备断电），并检测对应的馈电传感器是否有电：如断电成功，待甲烷浓度降低到预警值以下，恢复下位机相应的继电器工作，给开采设备通电；如断电失败，上位机要立即报警，不可自动解除，设置手动解除报警机制；。

3) 井下温度过高或氧气浓度过低，需要同时开启两个通风风机，待温度和氧气浓度适宜，关闭备用风机，保持主风机开启。

4) 系统需要上位机和下位机保持时间同步。如果下位机传来的数据（其中包括时间信息）中，时间误差超过 2 秒，要给下位机下发校正时钟命令，让下位机与上位机的时间保持一致。

5) 下位机对于上位机发来的命令，要给出确认报文，并反馈上位机的指令是否成功执行，避免服务器重传。

6) 设置数据校验机制。

3 协议的设计

3.1 通信模式

上位机与下位机采用客户服务器的方式连接。客户端通过 TCP 连接与服务器进行通信。TCP 是一种面向连接的、可靠的、基于字节流的通信协议，数据在传输前要建立连接，传输完毕后还要断开连接。

图 2 TCP 数据报的结构

每当客户端启动时，它会尝试连接服务器，直到连接成功。在服务器监听客户端连接，一旦连接成功，服务器会继续接收并处理来自客户端的数据。

3.2 数据交换

数据在客户端和服务器之间以 JSON 格式进行传输。客户端将各个传感器的数据整合成 JSON 对象，并使用 MD5 校验和对其进行验证，每次发送的数据都会有一个校验和，以确保数据的完整性。客户端发送的数据的格式为：

图 3 传输数据格式

服务器在接收到数据后，首先提取数据和校验和，进行验证。如果校验和匹配，则认为数据有效，继续处理；否则请求客户端重新发送数据。

3.3 校验机制

为了确保数据传输完整，客户端和服务器都使用 MD5 校验和来确保数据的完整性。

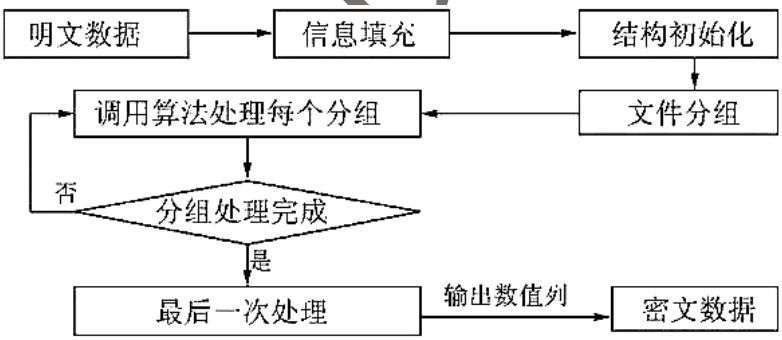

图 4 MD5 流程图

每次数据交换时，都会计算并附加校验和，并将数据和校验和一起发送给服务器。服务器接收到数据后，首先提取数据和校验和，然后计算数据的校验和，比较其与接收到的校验和。如果校验和不匹配，服务器会请求客户端重新发送数据。

3.4 指令交互

服务器根据接收到的数据分析结果，向客户端发送指令。这些指令包括：断电指令、起动风机指令、校时指令、报警指令、重新发送数据指令、数据处理成功指令等。客户端根据收到的指令执行相应的操作，包括：断电、开启风机、校时、报警等。

3.5 报警与处理机制

服务器根据传感器数据触发报警。如果甲烷浓度过高且断电失败，或者主备风机都关闭，则触发报警。触发报警后，服务器发送报警指令到客户端。客户端收到报警指令后，需要进行人工干预以解除报警，直到输入正确的密码（默认 0721）为止。客户端手动解除报警后，会发送确认信息给服务器。

3.6 时间同步

如果服务器发现客户端的时间戳与当前时间的差异超过一定阈值（2 秒），则会发送校时指令，要求客户端同步时间。客户端接收到该指令后，会执行时间同步操作，校正其系统时间。

3.7 异常处理

客户端在网络连接失败时会自动重连。在数据传输过程中，如果发生连接中断或数据校验失败，客户端会关闭当前连接并重新尝试连接。如果接收到的指令的校验和不匹配，客户端会拒绝执行该指令，并报告错误。

3.8 日志与反馈机制

服务器和客户端都有反馈日志机制。每次接收到数据或执行指令时，都会在控制台输出相应的日志消息。

4 项目主要流程

4.1 建立连接

客户端在收发数据前要使用 connect() 函数和服务器建立连接。建立连接的目的是保证 IP 地址、端口、物理链路等正确无误，为数据的传输开辟通道。TCP 建立连接时要传输三个数据包，即三次握手。

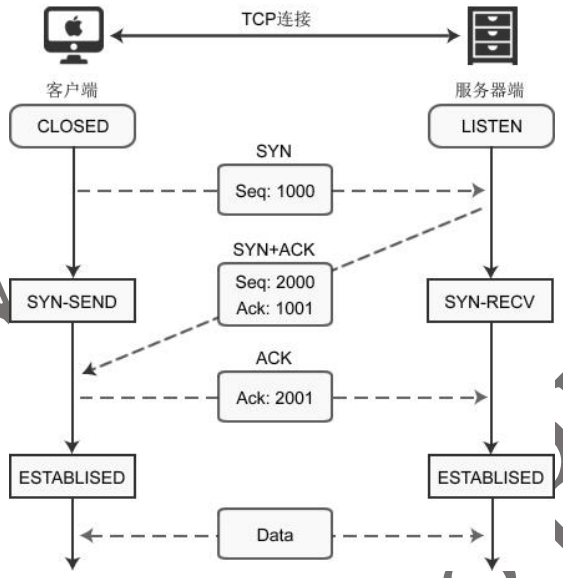

图 5 建立连接示意图

客户端调用 socket() 函数创建套接字后，因为没有建立连接，所以套接字处于 CLOSED 状态；服务器端调用 listen() 函数后，套接字进入 LISTEN 状态，开始监听客户端请求。

这个时候，客户端开始发起请求：

1) 当客户端调用 connect() 函数后，TCP 协议会组建一个数据包，并设置

SYN 标志位，表示该数据包是用来建立同步连接的。同时生成一个随机数字 1000，填充“序号（Seq）”字段，表示该数据包的序号。完成这些工作，开始向服务器端发送数据包，客户端就进入了 SYN-SEND 状态。

2) 服务器端收到数据包，检测到已经设置了 SYN 标志位，就知道这是客户端发来的建立连接的“请求包”。服务器端也会组建一个数据包，并设置 SYN 和 ACK 标志位，SYN 表示该数据包用来建立连接，ACK 用来确认收到了刚才客户端发送的数据包。服务器生成一个随机数 2000，填充“序号（Seq）”字段。2000 和客户端数据包没有关系。服务器将客户端数据包序号（1000）加 1，得到 1001，并用这个数字填充“确认号（Ack）”字段。服务器将数据包发出，进入 SYN-RECV 状态。

3) 客户端收到数据包，检测到已经设置了 SYN 和 ACK 标志位，就知道这是服务器发来的“确认包”。客户端会检测“确认号（Ack）”字段，看它的值是否为 1000+1，如果是就说明连接建立成功。接下来，客户端会继续组建数据包，并设置 ACK 标志位，表示客户端正确接收了服务器发来的“确认包”。同时，将刚才服务器发来的数据包序号（2000）加 1，得到 2001，并用这个数字来填充“确认号（Ack）”字段。客户端将数据包发出，进入 ESTABLISED 状态，表示连接已经成功建立。

4) 服务器端收到数据包，检测到已经设置了 ACK 标志位，就知道这是客户端发来的“确认包”。服务器会检测“确认号（Ack）”字段，看它的值是否为 2000+1，如果是就说明连接建立成功，服务器进入 ESTABLISED 状态。

至此，客户端和服务器都进入了 ESTABLISED 状态，连接建立成功，接下来就可以收发数据了。

4.2 数据传输

建立连接后，两台主机就可以相互传输数据了。

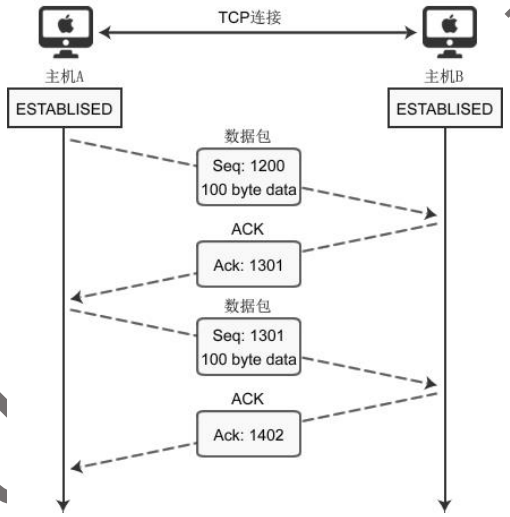

图 6 数据交换示意图 ‘

上图给出了主机 A 分 2 次（分 2 个数据包）向主机 B 传递 200 字节的过程。按如下的公式确认 Ack 号：

$$Ack\text{号} = Seq\text{号} + \textit{传递的字节数} + 1$$

主机 A 通过 1 个数据包发送 100 个字节的数据，数据包的 Seq 号设置为 1200。主机 B 为了确认这一点，向主机 A 发送 ACK 包，并将 Ack 号设置为 1301。

在数据传输过程中，数据包会遇到丢失的情况。如下图：

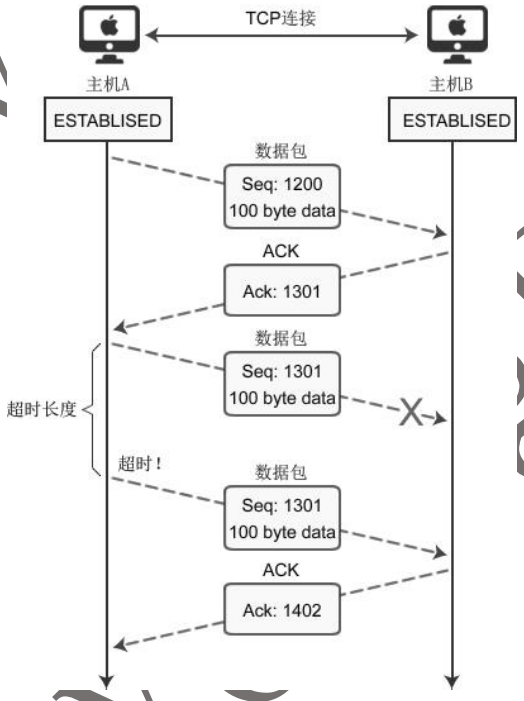

图 7 交换数据丢失示意图

上图表示通过 Seq 1301 数据包向主机 B 传递 100 字节的数据，但中间发生了错误，主机 B 未收到。经过一段时间后，主机 A 仍未收到对于 Seq 1301 的 ACK 确认，因此尝试重传数据。

为了完成数据包的重传，TCP 套接字每次发送数据包时都会启动定时器，如果在一定时间内没有收到目标机器传回的 ACK 包，那么定时器超时，数据包会重传。往返时间（RTT，Round-Trip Time）表示从发送端发送数据开始，到发送端收到来自接收端的 ACK 确认包（接收端收到数据后便立即确认），总共经历的时延。定时器采用 RTT 作为超时判定时间。

4.3 数据处理

数据处理主要分为上位机接收数据后的处理、下位机接收指令后的处理两个部分。

1）上位机接收数据后的处理

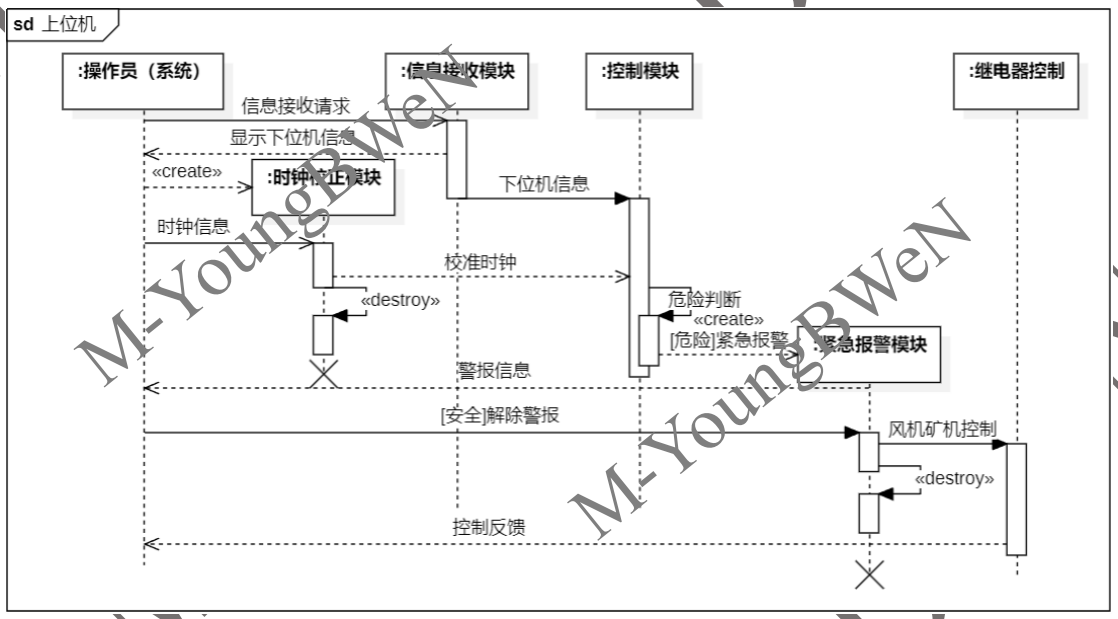

图 8 上位机数据处理时序图

上位机操作员（系统）首先向信息接收模块发送信息接收请求，请求通过后接收模块可以接收到下位机传递的数据并反馈给操作员；然后信息接收模块将下位机信息传递给控制模块进行数据处理（此时会通过时钟校正模块对时钟进行校正，并将结果反馈给控制模块），当数据为异常状态时会进行报警判断，如果达到报警条件会创建紧急报警模块；之后控制模块及紧急报警模块会将报警信息反馈给操作员和下位机（通过用户客户端的方式将指令和报警信息传递给下位机）；

最后下位机可以根据上位机的指令实现对继电器的控制。

2）下位机接收指令后的处理

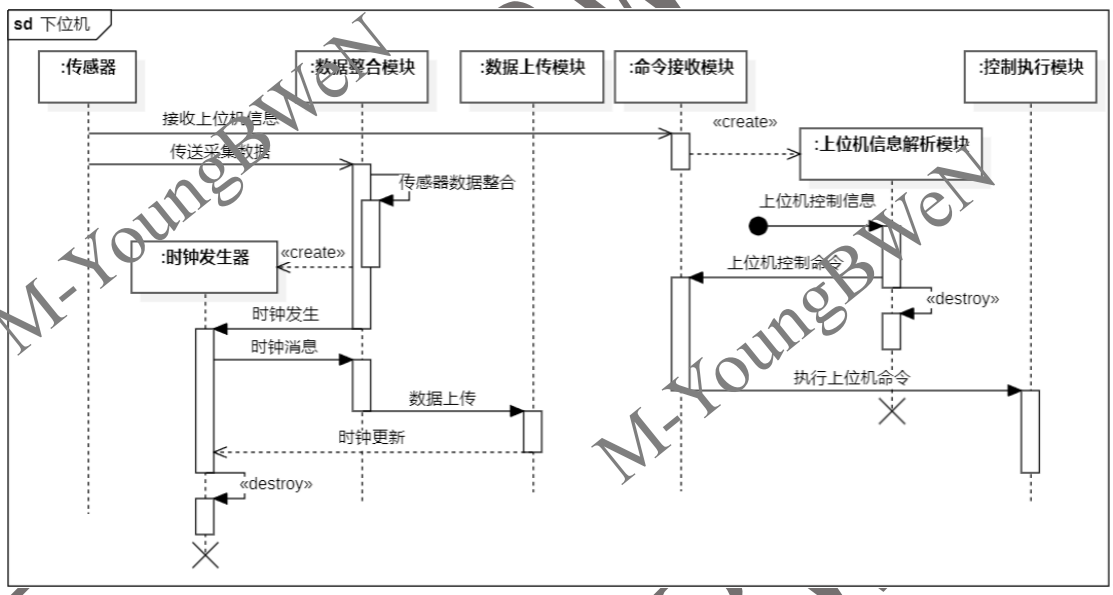

图 9 下位机数据处理时序图

下位机首先通过指令接收模块接收上位机传来的信息；然后指令接收模块会创建上位机信息接收模块解析上位机的指令是否正确，并将解析通过的指令反馈给命令接收模块；之后命令接收模块将控制命令传递给控制执行模块执行相应的指令。

4.4 断开连接

断开连接需要四次握手：

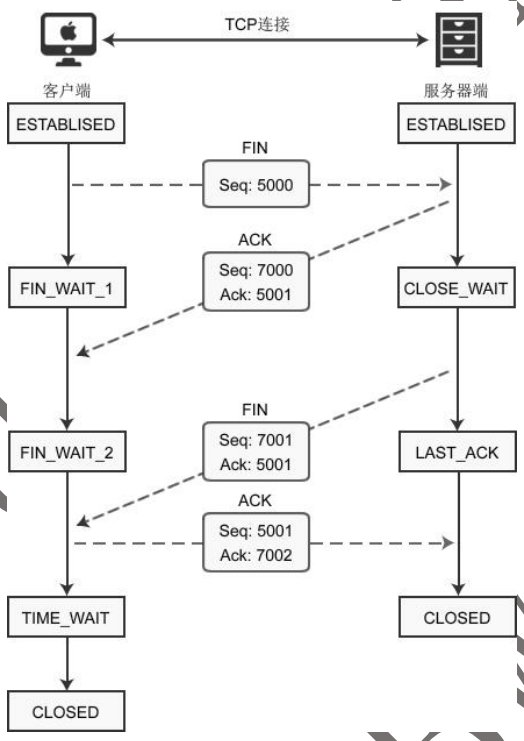

图 10 断开连接示意图

建立连接后，客户端和服务器都处于 ESTABLISED 状态。这时，客户端发起断开连接的请求：

1) 客户端调用 close() 函数后，向服务器发送 FIN 数据包，进入 FIN_WAIT_1 状态。FIN 是 Finish 的缩写，表示完成任务需要断开连接。

2) 服务器收到数据包后，检测到设置了 FIN 标志位，知道要断开连接，于是向客户端发送“确认包”，进入 CLOSE_WAIT 状态。

注意：服务器收到请求后并不是立即断开连接，而是先向客户端发送“确认包”，告诉它我知道了，我需要准备一下才能断开连接。

3) 客户端收到“确认包”后进入 FIN_WAIT_2 状态，等待服务器准备完毕后再次发送数据包。

4) 等待片刻后，服务器准备完毕，可以断开连接，于是再主动向客户端发送 FIN 包，告诉它我准备好了，断开连接吧。然后进入 LAST_ACK 状态。

5) 客户端收到服务器的 FIN 包后，再向服务器发送 ACK 包，告诉它你断开连接吧。然后进入 TIME_WAIT 状态。客户端最后一次向服务器回传 ACK 包时，有可能会因为网络问题导致服务器收不到，服务器会再次发送 FIN 包，如果这时客户端完全关闭了连接，那么服务器无论如何也收不到 ACK 包了，所以

客户端需要等待片刻、确认对方收到 ACK 包后才能进入 CLOSED 状态。TIME_WAIT 要等待 2MSL（报文最大生存时间）才会进入 CLOSED 状态。

6) 服务器收到客户端的 ACK 包后，就断开连接，关闭套接字，进入 CLOSED 状态。

5 项目的实现

5.1 通信协议的构建

使用 Python 的 socket 模块实现了对 TCP 协议的整体构建：

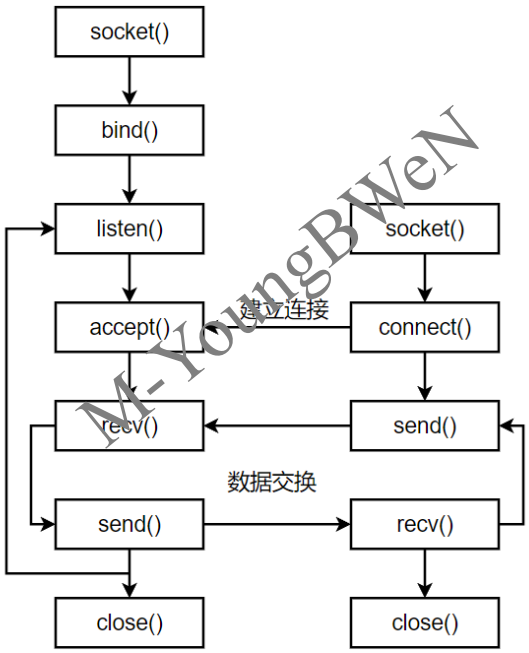

图 11 客户-服务器的实现示意图

1）服务器（上位机）

① 使用 socket.socket(socket.AF_INET, socket.SOCK_STREAM)创建 TCP 套接字（SOCK_STREAM），用于建立 TCP 连接。socket.AF_INET 用来指明使用 IPv4 地址。流格式套接字（Stream Sockets）也叫“面向连接的套接字”，在代码中使用 SOCK_STREAM 表示。SOCK_STREAM 是一种可靠的、双向的通信数据流，数据可以准确无误地到达另一台计算机，如果损坏或丢失，可以重新发送。

② 使用 self.server_socket.bind((self.host, self.port))绑定服务器的 IP 地址和端口号。

③ 使用 self.server_socket.listen(1)启动监听，监听下位机连接，表示服务器

处于被动模式，等待下位机的连接。

④ client_socket, client_address = self.server_socket.accept()：服务器通过accept()方法等待并接受客户端的连接请求。此方法会阻塞，直到有客户端连接到服务器。一旦建立连接，它返回一个新的套接字对象（client_socket）用于与该客户端进行通信。

⑤ 在数据处理完成后，服务器通过 client_socket.close()关闭与客户端的连接。

2）客户端（下位机）

① 使用 socket.socket(socket.AF_INET, socket.SOCK_STREAM)创建一个 TCP 套接字用于连接服务器。

② self.client_socket.connect((self.host, self.port))：客户端通过 connect()方法向服务器发起连接请求。如果服务器在指定地址和端口处监听，连接会成功建立。

③ 当客户端发生错误或者程序结束时，会调用 self.client_socket.close()关闭与服务器的连接。

3）数据发送

在客户端和服务器之间传输数据时，使用了 send()方法发送数据。

4）数据接收

data = self.client_socket.recv(1024)，客户端通过 recv()方法接收服务器发送的指令。TCP 协议会保证消息的完整性，如果数据传输过程中出现丢包或错误，TCP 会负责重传和数据完整性检查。

服务器端同样通过 recv()方法接收客户端发送的数据。recv()会返回一个字节对象（bytes），直到接收到的数据达到指定大小或者连接关闭。

5.2 上位机的实现

1）解析消息和校验和

接收到的数据包含传感器数据和校验和，格式为“json 数据|校验和”。服务器端会通过 extract_data_and_checksum 方法提取出数据部分和校验和部分：

```
sensor_data, received_checksum = self.extract_data_and_checksum(message)
```

此时，sensor_data 为解析后的 JSON 数据（包含传感器的各种状态），received_checksum 为客户端发送的数据的校验和。

服务器随后会计算从客户端接收到的 sensor_data 的校验和，使用

compute_checksum 方法进行计算：

```
computed_checksum = self.compute_checksum(sensor_data)
```

2）校验数据的完整性

在校验步骤中，服务器会检查接收到的校验和与计算出的校验和是否一致。如果一致，说明数据完整，服务器可以继续处理数据；否则，服务器会请求客户端重新发送数据：

```
if received_checksum != computed_checksum:
    self.dialog_server(2)  # 数据校验失败，要求重新发送
    self.send_command_to_client('resend', True)
    continue
```

如果校验通过，服务器则处理接收到的数据，执行相应的操作。

3）处理传感器数据

服务器根据传感器的数据内容来判断是否需要进行报警、是否需要开启设备，或者是否需要处理其他逻辑。例如：

① 甲烷浓度报警

```
if sensor_data['methane'] > 5.0:
    self.dialog_server(12)  # 甲烷浓度过高报警
    self.send_command_to_client('cut_power', True)  # 发送断电指令
```

② 温度和氧气浓度警告

```
elif sensor_data['temperature'] > 60 or sensor_data['oxygen'] < 18:
    self.dialog_server(13)  # 温度过高或氧气浓度过低
    self.send_command_to_client('fan', 'both_on')  # 开启风机
```

根据数据内容，服务器可以做出相应的指令，如断电、开启风机、校时等。

4）响应指令到客户端

服务器根据处理的结果生成响应指令并发送回客户端。发送指令时，服务器会重新计算校验和，以确保数据的完整性。

```
def send_command_to_client(self, command, value):
    """发送指令到下位机"""
    for client in self.clients:
```

```
command_data = json.dumps({command: value})
checksum = self.compute_checksum(command_data)
message = f"{command_data}|{checksum}"
client.send(message.encode('utf-8'))
```

如果服务器有报警等指令，会通过 send_command_to_client 函数将指令发送给客户端。此时，指令会附带校验和。

5.3 下位机的实现

1）解析消息和校验和

计算校验和与上位机类似。

如果校验和不一致，客户端会放弃该指令，并向服务器反馈指令校验失败的错误信息：

```
if checksum != computed_checksum:
    self.dialog_client('receive_command_failure')
    break  # 跳过不合法的指令
```

2）指令解析

如果校验和一致，客户端会解析指令的内容。如以下指令：

① 断电指令

```
if 'cut_power' in command_data:
    self.dialog_state = 1
    # 执行断电操作
```

② 风机控制

```
elif 'fan' in command_data:
    self.dialog_state = 2
    # 执行风机控制
```

③ 时间同步

```
elif 'sync_time' in command_data:
    self.dialog_state = 3
    # 执行时间同步
```

```
        self.sync_time()
```

④ 解除警报

```
elif 'alarm' in command_data:
        self.dialog_state = 4 #  等待用户输入密码来解除报警
        self.resolve_alarm()
```

3）执行指令

指令解析后会进行相应的执行操作。比如报警解除，需要输入密码才能解除：

```
def resolve_alarm(self):
    """手动解除报警"""
    while True:
        resolve_command = input('输入解除警报密码：')
        if resolve_command == '0721':
            confirmation_message = {"alarm_message": "False"}
            message = json.dumps(confirmation_message)
            checksum = self.compute_checksum(message)
            full_message = f"{message}|{checksum}"
            self.client_socket.send(full_message.encode('utf-8'))
            break
```

只有当下位机用户输入正确的密码后，才会发送解除报警的消息到上位机。

4）正常/异常响应

指令解析或执行异常、指令执行成功时会均会触发相应的响应机制。

① 指令发送异常

如果上位机要求下位机重新发送指令执行情况，下位机会执行重发机制：

```
elif 'resend' in command_data:
        self.dialog_state = 5
        self.resend_message()
```

② 指令执行完成

```
elif 'next_message' or 'success' in command_data:
```

```
self.dialog_client('command_execution_completed')
break  # 指令执行完成，跳出循环
```

5.4 界面设计

图 12 界面设计

6 项目实现结果

6.1 双风机关闭

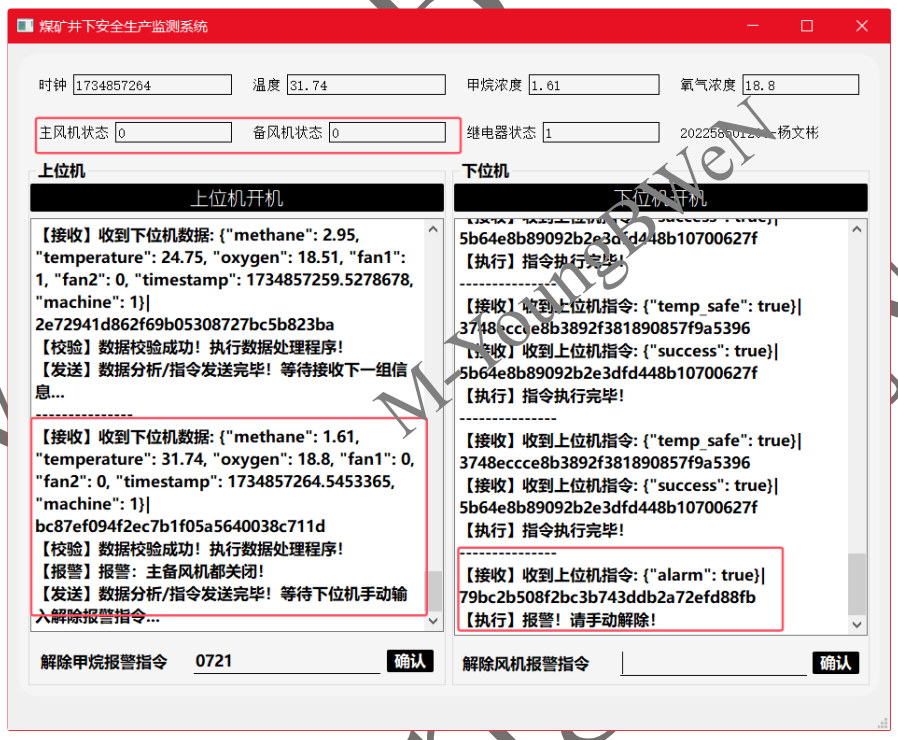

图 13 主备风机关闭报警

图 14 解除报警

6.2 甲烷浓度超过阈值

图 15 甲烷浓度过高断电成功

图 16 甲烷浓度在预警值以下

图 17 甲烷浓度过高断电失败报警

图 18 手动解除警报

6.3 温度或氧气不适宜

图 19 氧气或温度不适宜开风机

图 20 氧气浓度适宜

6.4 时间同步

图 21 校时

6.5 确认报文

图 22 命令执行结束会发送确认报文

6.6 数据校验

图 23 上位机下位机均会通过校验和校验数据

7 课程设计总结

在本次计算机网络课程设计中，我使用了 Python 的 `socket` 库实现了基于 TCP 协议的客户端-服务器通信，并结合 PyQt5 实现了图形化用户界面（GUI）。该项目的主要目的是理解并应用计算机网络通信的基本原理，同时通过 PyQt5 提供的控件和布局来实现用户友好的界面。

首先，TCP 协议作为一种可靠的面向连接的协议，它保证了数据传输的顺序性和完整性。在设计中，我首先实现了服务器端和客户端的基础框架。上位机主要负责监听指定的端口，接受来自下位机的连接请求。下位机则通过 socket 向上位机发送数据，并接收上位机的响应。在实现过程中，使用了 `socket.socket()` 来创建套接字，通过 `connect()` 方法与上位机建立连接，发送和接收数据则使用了 `send()` 和 `recv()` 方法。

在服务器端，我原本想要使用一个简单的多线程机制来处理多个客户端的请求。每当有新的客户端连接时，服务器会为其创建一个新的线程，以便同时处理多个客户端的数据。但是考虑到下位机就一台，因此默认客户机只有下位机。

为了让项目具备更好的用户体验，我使用 PyQt5 开发了图形化界面。在界面设计中，上位机和下位机各自拥有自己的线程，且各自拥有一个小窗口显示日志信息。UI 提供了文本输入框，允许用户输入解除报警的密码。通过 PyQt5 提

供的事件绑定机制，用户与界面之间的交互变得更加直观和简便。

在项目的实现过程中，我遇到了一些挑战。首先是 TCP 和 GUI 线程的通信问题。由于 PyQt5 的 GUI 操作只能在主线程中进行，因此我需要通过信号和槽机制来确保不同线程之间的数据同步和安全。其次，TCP 通信的调试也让我深刻理解了数据包的传输过程，如何处理不同类型的数据和确保传输的可靠性。

最终，项目实现了基于 TCP 协议的稳定通信，下位机和上位机能够实时地交换信息，并且用户可以通过图形化界面方便地与程序交互。通过这次课程设计，我不仅加深了对计算机网络原理的理解，也提高了自己在 Python 编程、Socket 编程以及 PyQt5 界面设计方面的能力。这个项目不仅让我掌握了实际应用中的网络通信技术，也让我体验到了将抽象的理论转化为具体应用的过程，进一步激发了我对计算机网络和软件开发的兴趣。

成绩评价表

课程目标	评价依据	评价指标及标准	满分	学生自评	教师确认
课程目标 1：针对相对复杂的具体应用问题场景与需求，能用通过查阅文献资料掌握主要解决思路【分析】，能在计算机网络体系结构层面上根据应用需求规划网络结构或者建立软件模型【创造】。	课程设计报告中“1 项目描述”和“2 网络结构、软件功能或者协议分析”部分	项目描述清晰、完整	5	5	
		用网络拓扑图或者软件功能功能能图或者交互时序图正确描述要解决问题的抽象模型。	5	5	
		难度：解决问题需要较复杂的网络功能、软件功能或者选择协议实现。	5	5	
		答辩：正确回答相关问题。连续两个问题回答错误此项不得分。	5	/	
课程目标 2：能够熟练使用仿真软件或者硬件设备组建网络并正确配置协议；或者使用软件开发工具实现功能功能完整的软件系统【应用】。	课程设计报告“3 网络设计、软件设计或者协议设计”部分	网络划分合理、设备选型恰当或者软件功能齐全或者协议三要素设计明确，并很好地论述了选用理由。	10	10	
		用拓扑图或者伪代码或流程图等形式规范、清晰地描述了主要功能或者协议三要素	5	5	
		正确分析和讨论主要设备的作用、软件系统功能或者网络通信协议的核心结构。	10	10	
		答辩：正确回答相关问题。连续两个问题回答错误此项不得分。	5	/	
课程目标 3：数量掌握不同操作系统或者不同那个网络协议之间通信所采用的机制，能够在实现中考虑到协议安全或者系统安全，了解网络安全所面临的问题以及相应解决方法【应用】。	项目验收、源程序、网络功能演示	难度：解决问题需要较复杂的网络设置、软件功能，或者选择了协议设计。	5	5	
		结构：网络结构、程序模块功能划分合理，网络设备连接接口清晰、软件功能完整、或者协议高效。	5	5	
		报告：按模板要求写作，（手写）字迹清楚，书写工整，或（打印）版面在边距、行距等方面处理恰当，整体可读性	5	5	

课程目标	评价依据	评价指标及标准	满分	学生自评	教师确认
		好。			
		答辩：正确回答相关问题。连续两个问题回答错误此项不得分。	5	/	
课程目标 4：在小组合作中，能够制定详细的工作计划，独立完成自己的任务，并提供清晰完整的文档【应用】；合理分解题目，达到小组成员任务明确，工作量均衡【评价】；建立有效的沟通协调机制和考核制度，制定文档模板【应用】	项目验收和答辩	答辩：独立完成某项功能，对应文档完善、清晰。	5	5	
	项目验收和答辩	答辩：分工合理，机制得当。	5	5	
课程目标 5：能够通过借助演示文稿等方式讲解、展示自己的工作；能够正确运用英语撰写报告摘要；合理回答老师和同学的问题，能够正确评价自己的工作的主要贡献【评价】。	课程设计报告英文摘要、工作宣讲	答辩：能够系统、清晰阐述自己的工作，	10	10	
		设计报告：能够正确使用英语撰写报告摘要	10	10	
			100	85	