1 远程管理 APP 开发

1.1 概述

远程管理 APP 是 MineSafe 系统的重要组成部分，旨在为矿井环境提供实时监控、设备控制和日志管理的功能。该 APP 基于 HTML5、CSS3 和 JavaScript 开发，并通过 Apache Cordova 框架打包为 Android APK，以实现跨平台部署和便捷的移动端访问。APP 通过 MQTT 协议与矿井头盔设备通信，实现数据的实时上传与指令下发。本节将详细介绍 APP 的架构设计、功能实现以及消息格式。

1.2 系统架构

远程管理 APP 采用前后端分离的架构，前端基于 Web 技术栈（HTML、CSS、JavaScript），后端通过 MQTT 协议与矿井头盔设备交互。整体架构如图 **??**所示。

- **前端界面**：采用响应式设计，基于 HTML5 和 CSS3 构建用户界面，支持 PC 端和移动端显示。核心库包括 MQTT.js，用于与 MQTT Broker 建立 WebSocket 连接。
- **通信层**：通过 WebSocket 协议连接至 EMQX Broker（wss://broker.emqx.io:8084/mqtt），实现低延迟、双向通信。APP 订阅 “helmet/status” 主题以接收头盔数据，并发布 “helmet/cmd” 主题以发送控制指令。
- **打包与部署**：使用 Apache Cordova 将 Web 应用打包为 Android APK，适配 Android 设备。打包流程包括环境配置（Node.js、JDK、Android SDK）、项目初始化、资源整合和 APK 构建。

1.3 功能实现

远程管理 APP 主要包括用户认证、实时监控、设备控制和状态日志四大功能模块，以下逐一阐述其实现细节。

1.3.1 用户认证

用户认证模块通过登录页面实现，确保仅授权用户可访问系统。登录页面采用 HTML 表单设计，包含用户名和密码输入框。CSS 样式使用 Flexbox 布局，确保界面在不同屏幕尺寸下居中显示。认证逻辑在 JavaScript 中实现，代码如下：

```
function attemptLogin() {
    const username = document.getElementById('username').value;
    const password = document.getElementById('password').value;
    if (username === 'chan' && password === 'chan') {
        document.getElementById('loginPage').style.display = 'none';
        document.body.classList.remove('login-active');
        document.getElementById('appContainer').style.display = '
           block';
        connectMQTT();
    } else {
        alert('Invalid username or password');
    }
}
```

登录成功后，隐藏登录页面，显示主应用界面，并调用 connectMQTT() 函数建立 MQTT 连接。为增强安全性，实际部署中可集成后端认证服务（如 OAuth2）。

1.3.2 实时监控

实时监控模块展示头盔传感器数据，包括温度、湿度、烟雾值、照明状态、地理位置和危险状态。数据通过订阅 “helmet/status” 主题获取，消息以 JSON 格式解析并动态更新到界面。界面采用网格布局（CSS Grid），每个数据项显示在独立卡片中，危险状态卡片支持动态闪烁动画以提醒用户。核心更新逻辑如下:

```
client.on('message', (topic, message) => {
    if (topic === 'helmet/status') {
        try {
            const data = JSON.parse(message.toString());
            updateMonitoringData(data);
        } catch (e) {
            console.log('Error parsing status data:', e);
        }
    }
});

function updateMonitoringData(data) {
    if (data.danger !== undefined) {
        dangerValue.textContent = data.danger ? '是' : '否';
        if (data.danger) {
            dangerItem.classList.add('blink');
            monitorPanel.classList.add('warning-panel');
            alertBar.style.display = 'block';
            addLogEntry('危险状态触发');
        } else {
            dangerItem.classList.remove('blink');
            monitorPanel.classList.remove('warning-panel');
            alertBar.style.display = 'none';
        }
    }
    // 更新温度、湿度、烟雾值等
}
```

地理位置数据通过 convertToDMS 函数将十进制度数转换为度分秒（DMS）格式，并显示为东经/西经、北纬/南纬，增强可读性:

```
function convertToDMS(dec) {
    const deg = Math.floor(dec);
    const minFloat = (dec - deg) * 60;
    const min = Math.floor(minFloat);
    const sec = (minFloat - min) * 60;
    return `\${deg}^\circ\${min}'\${sec.toFixed(3)}"`;
}
```

当检测到危险状态（data.danger === true）时，界面显示警告条并触发日志记录，警告条支持点击关闭。

1.3.3 设备控制

设备控制模块允许用户远程控制头盔的照明开关、触发或取消预警，并设置温度、湿度、烟雾的阈值。控制指令通过发布 “helmet/cmd” 主题发送，消息为 JSON 格式。照明控制界面提供“开启”和“关闭”按钮，点击后调用 `setLight` 函数：

```
function setLight(state) {
    const cmdObj = { "light_switch": state };
    client.publish(CMD_TOPIC, JSON.stringify(cmdObj), (err) => {
        if (err) {
            alert('灯光控制命令发送失败');
        } else {
            addLogEntry(`灯光已\${state === 'on' ? '开启' : '关闭
               '}`);
        }
    });
}
```

阈值设置通过输入框获取用户输入，验证后发送至头盔。例如，温度阈值设置逻辑如下：

```
function sendTempThreshold() {
    const tempTh = document.getElementById('tempThreshold').value.
       trim();
    if (!tempTh) {
        alert('请输入温度阈值');
        return;
    }
    const thresholdObj = { "temperature_threshold": parseFloat(
       tempTh) };
    sendThreshold(thresholdObj, '温度');
}
```

远程预警功能通过 `triggerWarning` 函数实现，支持启动或取消预警，增强系统对紧急情况的响应能力。

1.3.4 状态日志

状态日志模块记录系统操作和事件（如危险触发、灯光开关、阈值更新），日志条目按时间倒序显示。日志数据存储在 `logRecords` 数组中，并动态渲染到界面：

```
function addLogEntry(message) {
    const now = new Date();
    const logEntry = document.createElement('div');
    logEntry.className = 'log-entry';
    logEntry.textContent = `\${now.toLocaleString()} - \${message}`;
    logContainer.prepend(logEntry);
    logRecords.push({ time: now, message });
}
```

日志界面使用 CSS 样式美化，确保条目清晰可读，支持滚动查看历史记录。

1.4 MQTT 消息格式

通信采用 MQTT 协议，消息格式为 JSON，分为上行（头盔上传）和下行（APP 下发）两类。

1.4.1 上行消息

上行消息由头盔发布至 “helmet/status” 主题，包含传感器数据和配置信息，格式如下：

```
{
    "temperature": 25.3,
    "humidity": 60.5,
    "smoke": 150.7,
    "latitude": 39.9042,
    "latitudeHem": "N",
    "longitude": 116.4074,
    "longitudeHem": "E",
    "light": "off",
    "danger": false,
    "temperatureThreshold": 30,
    "humidityThreshold": 80,
    "smokeThreshold": 2000.0
}
```

各字段含义：

- `temperature`：温度值（℃）。
- `humidity`：湿度值（%）。
- `smoke`：烟雾浓度（ppm）。
- `latitude`，`longitude`：地理坐标（十进制度）。
- `latitudeHem`，`longitudeHem`：半球标识（N/S, E/W）。
- `light`：照明状态（“on” 或 “off”）。
- `danger`：危险状态（true/false）。
- `temperatureThreshold`，`humidityThreshold`，`smokeThreshold`：阈值设置。

1.4.2 下行消息

下行消息由 APP 发布至 “helmet/cmd” 主题，用于发送控制指令，格式如下：

- 灯光控制：{"light$_s witch$" : "on"} {"light$_s$witch" : "off"}
- - 温度：{"temperature$_t hreshold$" : 30} {"humidity$_t$hreshold" : 80}

- 烟雾：{"smoke$_t hreshold$" : 2000.0}

远程预警：{"remote$_w arning$" : "$activate$"} {"remote$_w$arning" : "deactivate"}

1.5 打包与部署

为实现移动端部署，APP 通过 Apache Cordova 打包为 Android APK。打包流程包括：

1. 安装 Node.js、Cordova CLI、JDK（8 和 17）、Android SDK，并配置环境变量。
2. 创建 Cordova 项目（`cordova create myApp`），添加 Android 平台（`cordova platform add android`）。
3. 将 HTML、CSS、JS 文件复制到 `www` 目录，确保入口文件为 `index.html`。
4. 修改 `config.xml`，设置应用名称和图标。
5. 执行 `cordova build android` 生成 APK，解决 Gradle 下载、JDK 版本等常见问题。
6. 使用 `cordova run android` 或 `adb install` 将 APK 部署至设备。

常见问题（如 Gradle 下载缓慢）通过配置阿里云 Maven 镜像和手动指定 Gradle 包解决，确保构建成功。

1.6 总结

远程管理 APP 通过 Web 技术和 MQTT 协议实现了矿井环境的实时监控与远程控制。用户认证确保系统安全性，实时监控提供直观数据展示，设备控制支持灵活操作，状态日志记录关键事件。MQTT 消息格式清晰高效，保证了数据通信的可靠性。Cordova 打包技术使 APP 适配 Android 设备，满足移动化需求，为 MineSafe 系统提供了便捷的管理入口。