Поскольку вы переходите на **Electron + Vite**, вам нужно реорганизовать этот файл. Вот что нужно сделать:

1. Разделение кода на модули

Создайте следующую структуру файлов:

```
src/
  main/          # Electron main process
    index.js     # Главный процесс Electron
    udp-server.js # UDP сервер (вынесен отдельно)
  renderer/      # Electron renderer process (Vite)
    main.js      # Клиентский код
    components/  # Vue/React компоненты
  shared/        # Общие утилиты
    mavlink-parser.js
```

2. UDP сервер (main process)

src/main/udp-server.js:

```
const dgram = require('dgram');
const { ipcMain } = require('electron');

class UDPServer {
  constructor(mainWindow) {
    this.udpServer = dgram.createSocket('udp4');
    this.mainWindow = mainWindow;
    this.UDP_PORT = 14550;
  }

  start() {
    this.udpServer.bind(this.UDP_PORT, () => {
      console.log(`UDP server listening on port ${this.UDP_PORT}`);
    });

    this.udpServer.on('message', (msg, rinfo) => {
      try {
        const data = this.parseMAVLink(msg);
        if (data && this.mainWindow) {
          // Отправляем данные в renderer process
          this.mainWindow.webContents.send('mavlink-data', data);
        }
      } catch (e) {
        console.error('Error processing MAVLink message:', e);
      }
    });
  }
```

```
  parseMAVLink(buffer) {
    // Перенесите сюда вашу функцию parseMAVLink из исходного файла
    // ... (код функции parseMAVLink)
  }

  stop() {
    this.udpServer.close();
  }
}

module.exports = UDPServer;
```

3. Главный процесс Electron

src/main/index.js:

```
const { app, BrowserWindow, ipcMain } = require('electron');
const path = require('path');
const UDPServer = require('./udp-server');

let mainWindow;
let udpServer;

function createWindow() {
  mainWindow = new BrowserWindow({
    width: 1200,
    height: 800,
    webPreferences: {
      nodeIntegration: false,
      contextIsolation: true,
      preload: path.join(__dirname, 'preload.js')
    }
  });

  // Загружаем Vite dev server в development или файл в production
  if (process.env.NODE_ENV === 'development') {
    mainWindow.loadURL('http://localhost:5173');
    mainWindow.webContents.openDevTools();
  } else {
    mainWindow.loadFile(path.join(__dirname, '../renderer/index.html'));
  }

  // Инициализируем UDP сервер после загрузки окна
  mainWindow.webContents.on('did-finish-load', () => {
    udpServer = new UDPServer(mainWindow);
    udpServer.start();
  });

  mainWindow.on('closed', () => {
    mainWindow = null;
    if (udpServer) {
      udpServer.stop();
    }
  });
```

```
}

app.whenReady().then(createWindow);

app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') app.quit();
});

app.on('activate', () => {
  if (BrowserWindow.getAllWindows().length === 0) createWindow();
});

// IPC handlers
ipcMain.handle('get-udp-port', () => {
  return 14550;
});
```

4. Preload script

src/main/preload.js:

```
const { contextBridge, ipcRenderer } = require('electron');

contextBridge.exposeInMainWorld('electronAPI', {
  onMavlinkData: (callback) => {
    ipcRenderer.on('mavlink-data', (event, data) => callback(data));
  },
  getUDPPort: () => ipcRenderer.invoke('get-udp-port')
});
```

5. Клиентский код (renderer process)

src/renderer/main.js:

```
import { createApp } from 'vue'; // или React, в зависимости от выбора
import App from './App.vue';

// Обработка MAVLink данных
window.electronAPI.onMavlinkData((data) => {
  // Обновляем состояние вашего приложения
  console.log('MAVLink data received:', data);

  // Здесь можно диспатчить события в store (Vuex/Pinia/Redux)
  // или обновлять компоненты напрямую
});

const app = createApp(App);
app.mount('#app');
```

6. Настройка Vite

vite.config.js:

```
import { defineConfig } from 'vite';
import vue from '@vitejs/plugin-vue'; // или react

export default defineConfig({
  plugins: [vue()],
  root: 'src/renderer',
  build: {
    outDir: '../../dist/renderer'
  },
  server: {
    port: 5173
  }
});
```

7. Package.json скрипты

```
{
  "scripts": {
    "dev": "NODE_ENV=development electron src/main/index.js",
    "build": "vite build && electron-builder",
    "build:renderer": "vite build",
    "build:main": "electron-builder"
  }
}
```

Ключевые изменения:

1. **Разделение процессов**: UDP сервер работает в main process Electron
2. **IPC коммуникация**: Данные передаются через IPC между процессами
3. **Безопасность**: Используется contextIsolation и preload скрипты
4. **Vite интеграция**: Renderer process использует Vite для разработки
5. **Модульная структура**: Код разделен на логические модули

Такой подход обеспечит лучшую производительность, безопасность и удобство разработки в Electron + Vite среде.