# 功能

gwdata-server的主要功能是接收并解析无线网络通过串口传递过来的slip格式的数据。并将此数据按照json格式传递给应用层程序。



# 实现

软件实现上主要分了事件循环，串口读写，网络IO，协议解析四个模块



## 协议

### 串口协议

#### 数据帧：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 设备ID  (1字节) | 设备类型  (1字节) | 传输类型  (1字节) | 设备数据  (n字节,n>=0) | 校验和  (1字节) |

校验和是数据帧内其他所有字节之和的补码。

示例：

下表是一个温度传感器的数据帧结构。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 设备ID | 设备类型 | 传输类型 | 设备数据 | 校验和 |
| 0x01 | 0x10(温度) | 0x00(代表zigbee) | 0x10 | 0xdf |

#### 组包格式

按照slip格式组包。定义了END（0x7E）和ESC（0x7D）两个特殊字符。

|  |  |
| --- | --- |
| 数据帧  （n字节,n>=0） | END  (0x7e) |

示例：

下表是一个温度传感器数据的数据包结构

|  |  |
| --- | --- |
| 数据帧 | END |
| 0x01,0x10,0x00,0x10,0xdf | 0x7E |

但是当我们发送的数据帧内含有END（0x7e），连续发送两个字符ESC（0x7D）和0x5E，当数据帧内含有ESC（0x7E）时，连续发送两个字符ESC（0x7D）和0x5D。

示例：

数据帧含有0x7e、0x7d

|  |  |
| --- | --- |
| 数据帧 | END |
| 0x7E，0x7D | 0x7E |

组包后数据

|  |  |
| --- | --- |
| 数据帧 | END |
| 0x7D，0x5E，0x7D，0x5D | 0x7E |

#### 软件接口

|  |
| --- |
| /\*传感器数据描述结构\*/  struct sensor\_data**{**  int id**;**  int type**;**  cJSON **\***value**;**  char **\***transfer\_type**;**  char asctime**[**20**];**  time\_t timestamp**;**  **};**  /\*将sensor\_data结构描述的传感器数据，转为slip数据描述\*/  int sensor\_data\_to\_slip**(**struct sensor\_data **\***sd**,**char **\***slip**,**int size**);**  /\*解析串口数据帧，并将其转为sensor\_data结构\*/  struct sensor\_data **\***slip\_to\_sensor\_data**(**const char **\***slip**,**int len**);**  /\*根据slip分包格式，取得一个数据帧\*/  int buffer\_read\_slip**(**struct buffer **\***buf**,**char **\***slip**,**int size**);** |

### 网络协议

#### 数据帧

传感器到网关

|  |  |
| --- | --- |
| device\_id | 设备ID |
| device\_type | 设备类型 |
| transfer\_type | 传输类型 |
| device\_value | 设备数据 |
| timestamp | 时间戳 |

示例

{“device\_id”:1,”device\_type”:16,”transfer\_type”:”zigbee”,”device\_value”:”16”}

网关到传感器

|  |  |
| --- | --- |
| cmd | 命令 |
| args | 参数 |

典型的命令有

{“cmd”:”set\_switch”,”args”:{“device\_id”:1,”device\_value”:”true”}}

#### 组包格式

行解析，每条数据都以’\n’为结束符。

#### 软件接口

|  |
| --- |
| int buffer\_read\_line**(**struct buffer **\***buf**,**char **\***line**,**int size**);**  /\*json解析采用了cJSON库，为标准用法\*/ |

### 串口读写

介绍

串口波特率设为115200,8位数据位，1位停止位，无校验位。

软件接口

|  |
| --- |
| struct gwseriport**;**  /\*打开串口\*/  struct gwseriport **\***gwseriport\_create**(**const char **\***ttypath**,**int uart\_speed**);**  /\*关闭串口\*/  void gwseriport\_release**(**struct gwseriport **\***s**);**  /\*根据传输协议，找到对应串口\*/  struct gwseriport **\***find\_transfer\_media**(**const char **\***transfer\_type**);**  /\*打开串口\*/  int open\_seriport**(**const char **\***ttypath**,**int uart\_speed**);**  /\*写数据到串口\*/  int write\_seriport**(**struct gwseriport **\***s**,**char **\***buf**,**int len**);** |

### 网络IO

介绍

创建了一个服务器，监听51001端口。

软件接口

|  |
| --- |
| int anetTcpConnect**(**char **\***err**,** char **\***addr**,** int port**);**  int anetTcpNonBlockConnect**(**char **\***err**,** char **\***addr**,** int port**);**  int anetTcpNonBlockBindConnect**(**char **\***err**,** char **\***addr**,** int port**,** char **\***source\_addr**);**  int anetTcpNonBlockBestEffortBindConnect**(**char **\***err**,** char **\***addr**,** int port**,** char **\***source\_addr**);**  int anetUnixConnect**(**char **\***err**,** char **\***path**);**  int anetUnixNonBlockConnect**(**char **\***err**,** char **\***path**);**  int anetRead**(**int fd**,** char **\***buf**,** int count**);**  int anetResolve**(**char **\***err**,** char **\***host**,** char **\***ipbuf**,** size\_t ipbuf\_len**);**  int anetResolveIP**(**char **\***err**,** char **\***host**,** char **\***ipbuf**,** size\_t ipbuf\_len**);**  int anetTcpServer**(**char **\***err**,** int port**,** char **\***bindaddr**,** int backlog**);**  int anetTcp6Server**(**char **\***err**,** int port**,** char **\***bindaddr**,** int backlog**);**  //int anetUnixServer(char \*err, char \*path, mode\_t perm, int backlog);  int anetTcpAccept**(**char **\***err**,** int serversock**,** char **\***ip**,** size\_t ip\_len**,** int **\***port**);**  int anetUnixAccept**(**char **\***err**,** int serversock**);**  int anetWrite**(**int fd**,** char **\***buf**,** int count**);**  int anetNonBlock**(**char **\***err**,** int fd**);**  int anetBlock**(**char **\***err**,** int fd**);**  int anetEnableTcpNoDelay**(**char **\***err**,** int fd**);**  int anetDisableTcpNoDelay**(**char **\***err**,** int fd**);**  int anetTcpKeepAlive**(**char **\***err**,** int fd**);**  int anetSendTimeout**(**char **\***err**,** int fd**,** long long ms**);**  int anetPeerToString**(**int fd**,** char **\***ip**,** size\_t ip\_len**,** int **\***port**);**  int anetKeepAlive**(**char **\***err**,** int fd**,** int interval**);**  int anetSockName**(**int fd**,** char **\***ip**,** size\_t ip\_len**,** int **\***port**);**  int anetFormatAddr**(**char **\***fmt**,** size\_t fmt\_len**,** char **\***ip**,** int port**);**  int anetFormatPeer**(**int fd**,** char **\***fmt**,** size\_t fmt\_len**);**  int anetFormatSock**(**int fd**,** char **\***fmt**,** size\_t fmt\_len**);** |

### 事件循环

介绍

采用linux的epoll模型，做的IO事件模型

软件接口

|  |
| --- |
| int anetTcpConnect**(**char **\***err**,** char **\***addr**,** int port**);**  int anetTcpNonBlockConnect**(**char **\***err**,** char **\***addr**,** int port**);**  int anetTcpNonBlockBindConnect**(**char **\***err**,** char **\***addr**,** int port**,** char **\***source\_addr**);**  int anetTcpNonBlockBestEffortBindConnect**(**char **\***err**,** char **\***addr**,** int port**,** char **\***source\_addr**);**  int anetUnixConnect**(**char **\***err**,** char **\***path**);**  int anetUnixNonBlockConnect**(**char **\***err**,** char **\***path**);**  int anetRead**(**int fd**,** char **\***buf**,** int count**);**  int anetResolve**(**char **\***err**,** char **\***host**,** char **\***ipbuf**,** size\_t ipbuf\_len**);**  int anetResolveIP**(**char **\***err**,** char **\***host**,** char **\***ipbuf**,** size\_t ipbuf\_len**);**  int anetTcpServer**(**char **\***err**,** int port**,** char **\***bindaddr**,** int backlog**);**  int anetTcp6Server**(**char **\***err**,** int port**,** char **\***bindaddr**,** int backlog**);**  //int anetUnixServer(char \*err, char \*path, mode\_t perm, int backlog);  int anetTcpAccept**(**char **\***err**,** int serversock**,** char **\***ip**,** size\_t ip\_len**,** int **\***port**);**  int anetUnixAccept**(**char **\***err**,** int serversock**);**  int anetWrite**(**int fd**,** char **\***buf**,** int count**);**  int anetNonBlock**(**char **\***err**,** int fd**);**  int anetBlock**(**char **\***err**,** int fd**);**  int anetEnableTcpNoDelay**(**char **\***err**,** int fd**);**  int anetDisableTcpNoDelay**(**char **\***err**,** int fd**);**  int anetTcpKeepAlive**(**char **\***err**,** int fd**);**  int anetSendTimeout**(**char **\***err**,** int fd**,** long long ms**);**  int anetPeerToString**(**int fd**,** char **\***ip**,** size\_t ip\_len**,** int **\***port**);**  int anetKeepAlive**(**char **\***err**,** int fd**,** int interval**);**  int anetSockName**(**int fd**,** char **\***ip**,** size\_t ip\_len**,** int **\***port**);**  int anetFormatAddr**(**char **\***fmt**,** size\_t fmt\_len**,** char **\***ip**,** int port**);**  int anetFormatPeer**(**int fd**,** char **\***fmt**,** size\_t fmt\_len**);**  int anetFormatSock**(**int fd**,** char **\***fmt**,** size\_t fmt\_len**);** |

#### 开发指导

这个软件功能相对单一，目的明确，不会出现设计上重来的问题。前面我们主要介绍了软件的编写思路。这里我们介绍下当此软件不满足我们的需要，比如系统中需要增加传感器时,我们应该怎么做

增加传感器(以led灯为例)

1 修改 device.c 文件，添加如下两个函数

|  |
| --- |
| /\*slip转json\*/  static cJSON **\***led\_v2json**(**int id**,**const char **\***data**,**int len**)**  **{**  int v **=** data**[**0**];**  **if(**v **==** 1**){**  **return** cJSON\_CreateString**(**"true"**);**  **}else{**  **return** cJSON\_CreateString**(**"false"**);**  **}**  **}**  /\*json转slip\*/  static int led\_v2chararray**(**int id**,**cJSON **\***value**,**char **\***buf**,**int len**)**  **{**  **switch(**value**->**type**){**  **case** cJSON\_False**:**  buf**[**0**]** **=** 3**;**  **break;**  **case** cJSON\_True**:**  buf**[**0**]** **=** 1**;**  **break;**  **case** cJSON\_String**:**  **if(**strcasecmp**(**value**->**valuestring**,**"true"**)** **==** 0**)**  buf**[**0**]** **=** 1**;**  **else**  buf**[**0**]** **=** 3**;**  **break;**  **default:**  **break;**  **}**  **return** 8**;**  **}** |

2 device.c文件中找到 devices数组，在其中添加一条

|  |
| --- |
| **{**0x18**,**led\_v2json**,**led\_v2chararray**,NULL},** |