唤梦科技.数据采集系统.设计说明书

------深圳市唤梦科技有限公司.编制

版本修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 动作 | 日期 | 备注 |
| 1.0 | 创建 | 2016年10月28日 |  |
| 2.0 | 修改 | 2016年11月02日 |  |
| 3.0 | 修改 | 2016年11月18日 |  |
| 4.0 | 修改 | 2016年11月27日 | 支持网络，支持数据库 |
| 5.0 | 修改 | 2017年01月7日 | 界面皮肤 |
| 6.0 | 定制 | 2017年01月19日 | 更改主界面显示内容和显示方式，  增加设备管理模块，  增加对设备在线状态的维护，  对多语言版本的支持 |

**一 系统物理拓扑结构图：**



**二 采集系统描述：**

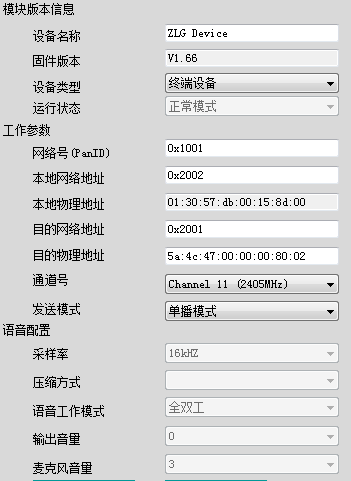
唤梦科技工业缝纫机数据采集系统，是应用于大批量工业缝纫机企业的自动化信息管理系统的子系统，具有对系统内工业缝纫机进行数据采集，参数配置，远程控制，和调测的功能。

采集系统采用传感器，串口，zigbee，以太网等多种数据通讯技术，在终端设备（工业缝纫机），采集器，转换网关，和服务器主机之间传递指令，上传下载数据，实现对终端数据的采集和对终端的配置，对终端的控制 ，在服务器主机上实现数据的存储和展示等。

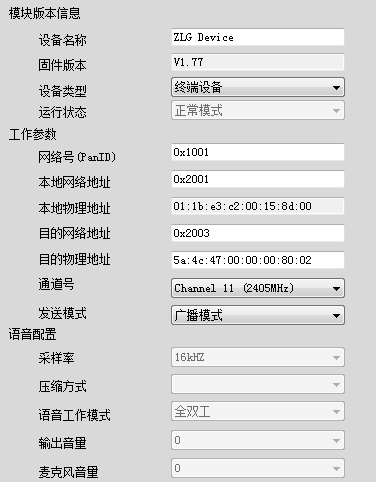
企业ERP系统可以直接访问库数据或者通过编程接口间接访问数据库。

**三 技术构架概述：**

* 采集器集成zigbee模块，采集器拥有串口，以便和电脑连接，对zigbee进行配置
* 每个采集器一般对应一个终端设备（工业缝纫机），终端设备通过传感器或者串口，和采集器通讯。
* 每个采集器在zigbee网络内有唯一的id, 以及通讯频道
* 转换网关集成zigbee和以太网转换模块，每个转换网关在zigbee网络内有唯一的id
* 每个转换网关和从属于这个网关的终端设备拥有相同的zigbee网络号，相同的通讯频道。
* 所有转换网关，通过JR45网线连接到路由器，这些转换网关在同一个子网内， 在以太网内分配唯一的IP地址。 转换网关先通过网线和电脑连接，设置子网掩码，默认网关，IP地址，tcp/IP工作模式，工作端口，之后再连接到路由器。
* 采集器和zigbee-ethernet转换网关之间通过zigbee协议通讯，每个转换网关对应N个 采集器。
* zigbee-ethernet转换网关和服务器主机之间通过ethernet网连接，通过TCP/IP协议通讯，服务器主机可以对应M个转换网关。
* 服务器主机和每个转换网关之间通过TCP/IP协议，构成C/S结构的模型，服务器主机为server端，转换网关为client端，转换网关配置为上电即自动连接服务器主机。
* 一个转换网关设置一个IP地址，一个zigbee网络id ， 二者构成一一对应关系。
* 缝纫机数据通过传感器，采集器，转换网关，中央服务器，以及数据库服务器，上传并保存到指定的数据库---数据中心，企业ERP系统直接访问数据中心，或者通过中央服务器提供的编程接口间接访问数据中心。 数据库选型待客户进一步确认。
* 非阻塞异步SOCKET通讯，IOCP模型+ 多线程池， 采用主线程加辅助线程的多线程编程技术，可以认为每个连接对应一个独立的工作线程，该部分是系统通讯的核心, 该模型单台服务器支持高达65535个并发用户。



采集器 zigbee配置示意图



转换网关的zigbee配置示意图



转换网关tcp/IP设置

**四 通讯协议：**

* tcp 和 串口 传输数据 类似的， 属于流式的，一方发送一个数据包，对方可能一次就刚好接收到一个完整的包，也可能分多次才接收完毕一个完整的数据包。当数据包尺寸比较小，通讯线路比较空闲的时候，可能对方一次性收到一个完整的包，当数据包比较大，或者线路比较忙的时候， 往往会出现一个包被分成了几个部分 ，所以接受方，需要组包，来还原数据包 ，所以组包是个关键算法。

UDP虽然是可以保证每次收发的都是一个完整的数据包，通过实测， 本转换网关，因为牵扯到串口通讯，所以，即使使用UDP的通讯方式，依然会出现分包的情形。

本方案我们采用tcp 模式， 不考虑udp。

* 上下位机之间的数据通讯采用的是二进制格式， 而不是ASCII码字符， 这个一定要注意，这个是因为我们传输的是二进制数据， 不是普通的聊天程序那样传输的文本信息 。
* 数据包格式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2 byte** | **2 byte** | **2 byte** | **2 byte** | **Data length byte** | **2 byte** |
| **0xAABB** | **command** | **Serial No** | **Data length** | **data** | **Check sum** |
| **unsigned short** | **unsigned short** | **unsigned short** | **unsigned short** | **char** | **unsigned short** |

* 校验码为数据包中除校验码之外的所有字段的累加和，对65535求余后的余数， 该字段为可选的，根据数据可靠性的要求，决定是否启用校验码，不启用校验码的，该字段可以填入任意数值， 对于下位机上报的普通的信息, 对于不需要对方确认的命令，都可以不做和校验 ，以便提高程序效率。

校验码是可选的，即使不不做校验， 那个校验码位置一定要填充2字节的数据。为了安全因素，上位机下发给下位机的 命令，建议要带校验码， 比如上位机发送的控制命令， 下发的更新系统的命令 等。

校验是需要花费一定的计算工作的，请权衡利弊。

* cmmd 定义 ：

#define CMMD\_NORMAL\_DATA 0 //下位机上传正常的数据

#define CMMD\_CONTROL\_UPGRADE 1 //上位机请求更新指定设备的系统，上位机给下位机下发文件

#define CMMD\_CONTROL\_ON 2 //上位机请求开启指定的设备

#define CMMD\_CONTROL\_OFF 3 //上位机请求关闭指定的设备

#define CMMD\_CONTROL\_SYNC\_TIME 4 //上位机请求同步下位机时间，用于矫正下位机时间

#define CMMD\_HEART\_BEAT 5 //上报设备状态， 设备心跳包，上报数据为上报时间（格式字符串 YYYY-MM-DD HH-MM-SS），zigebee网络的panid（十六进制数值） 和 zigbee网络本机地址（十六进制数值）； 定时上报，时间暂定为一分钟； 本命令不需要上位机回应； 上位机将心跳数据存放与数据库，上位机程序定时或者手动检查设备的状态

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度 byte | 字段说明 |
| **起始码** | unsigned short | 2 | 固定为2字节的 0xAABB |
| **command** | unsigned short | 2 | 0 --下位机 向上位机发送的普通数据（netId,netAddr,event,steps），这个指令不需要对方回应,  1--上位机请求下位机更新系统，下位机需要回应此命令， 该命令的data字段为配置数据，可以从本地电脑文件中读取  2--上位机请求控制下位机设备的请求，开机， 下位机需要回应命令  3--上位机请求控制下位机设备的请求，关机， 下位机需要回应命令  4--... |
| **Serial No** | unsigned short | 2 | 上下位机需要对命令进行回应确认的，需要设置序列号， 其他的情形，该字段忽略，可以任意设置没有实际意义  请求包和回应包，使用同一个序列号 |
| **Data length** | unsigned short | 2 | 数据长度 ，即data字段的长度 |
| **data** | char | data length | 数据 |
| **Check sum** | unsigned short | 2 | 起始码**0xAABB,** 不参与校验 !  校验码， 从起始码之后command字段，到data的 所有字段的 校验和，算法为，单字节求和再除以65535的余数，，该项为选项，建议上位机下发到下位机的命令使用校验，不使用校验的场合该字段可以填充该字段范围内的任意值 |

**下位机上报的普通数据**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度  byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |
| **event** | unsigned short | 2 | 事件类型  1 开机，  2 关机，  3剪线，  4 倒缝，  5 断线 |
| **time** | Char[19] | 19 | 时间字符串：2016-11-29 10:50:20 |
| **steps** | unsigned short | 2 | 针数 |
| **EmployeeNo.** | Char[10] | 10 | 员工工号：0100101001 |
| **ProcessNo.** | Char[24] | 24 | 工序号：301302303304305306307308 |
| **ModelNo.** | Char[8] | 8 | 款号：3902.601 |

示例数据 ：

unsigned short start = 0xaabb;

unsigned short cmmd = CMMD\_NORMAL\_DATA; // 0x0000;

unsigned short serialNo = 1;

unsigned short dataLen = 69;

cd.netId = 0x1001;

cd.netAddr = 0x2002;

cd.event = 0x0000;

memcpy(cd.time, "2016-11-30 11:06:18", 19);

cd.steps = 0x0100;

memcpy(cd.employeeNo, "0100101001" , 10); //[10]; //员工编号

memcpy(cd.processNo, "301302303304305306307308", 24); //[24]; //工序号

memcpy(cd.modelNo, "3902.601" ,8); //[8]; 型号

checksum = 0000;

aabb000000010045100120020000323031362d31312d33302031313a30363a3138000130313030313031303031333031333032333033333034333035333036333037333038333930322e3630310000

* **上位机对上传普通报文的回应包的data部分的格式**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度  byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |

* 更新系统文件指令的数据结构

struct TagUpgradeDataStruct

{

unsigned short netId;

unsigned short netAddr;

//////////////////////////////

unsigned short totalPack; //总的pack数量

unsigned short packNo; //pack编号

unsigned short packLen; // current pack length , 此处为当前pack的长度，标准pack的长度为MAX\_FILE\_PACK\_LEN 1024byte，

char fileName[30]; // 不足30的 空余地方填充0

// above sum up to 36 bytes

char \* packData; //packData[ current pack Len];

};

struct TagUpgradeResDataStruct //下位机到上位机

{

unsigned short netId;

unsigned short netAddr;

char fileName[30]; // 不足30的 空余地方填充0

};

* **更新系统文件指令的数据格式（上位机到下位机）**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |
| totalPack | unsigned short | 2 | pack总的数量 |
| packNo | unsigned short | 2 | pack编号 |
| packLen | unsigned short | 2 | 此处为当前pack的长度，标准pack的长度为MAX\_FILE\_PACK\_LEN 1024byte， |
| fileName | char | 30 | 不足30的 空余地方填充0 |
| packData | char | packLen | pack数据 |

* **更新系统文件响应指令的数据格式（下位机到上位机）**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |
| fileName | char | 30 | 不足30的 空余地方填充0 |

/\* test data : 回应数据示例

sample:

起始两个字节 0xAA 0xBB

cmmd = 0x0001

serialNo = 0x0006

dataLength =0x0022 //34

data指向一个结构体 TagUpgradeResDataStruct， 赋值如下 ：

netId=0x1001

netAddr=0x2002

fileName[30]="system" //0x73 0x79 0x73 0x74 0x65 0x6d 0x0000.......

checkSum = 0x0000 //这里可以任意写一个2字节的校验码

该完整的数据包是

0xaa 0xbb 0x0001 0x0006 0x0022 0x1001 0x2002 0x73 0x79 0x73 0x74 0x65 0x6d ....填充24个 0x0000..... 校验码 0x0000

去掉前缀0x 后 拷贝到串口调试助手，按十六进制 发送 即可

aa bb 0001 0006 0022 1001 2002 73 79 73 74 65 6d 0000.... 0000

aabb0001000600221001200273797374656d0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

\*/

* **控制开关机的指令数据部分格式**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |

同步下位机时间的指令 ， 数据是上位机当前本机的时间

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |
| Time | char[19] | 19 | 同步时间 |

示例 数据

AA BB 00 04 00 01 00 17 00 00 00 00 32 30 31 36 2D 31 32 2D 30 31 20 31 31 3A 33 36 3A 33 39 03 CE

* 校验和算法示例， 根据具体情况选用自己的算法：

注意网络字节顺序和本机字节顺序， 高低位字节顺序是相反的

unsigned short CProtocol::CheckSum(char \* buffer, int len)

{

unsigned short sum = 0;

for (int i = 0; i < len; i++)

{

sum = (sum+buffer[i]) % 65535;

}

return sum;

}

// 注意, 起始位不参与校验

unsigned short CProtocol::CheckSum( unsigned short cmmd, unsigned short serial,unsigned short length, char\* data)

{

char head[6];

char \* pHead = head;

unsigned short sum = 0;

\*((int\*)pHead) = cmmd;

pHead += sizeof(unsigned short);

\*((int\*)pHead) = serial;

pHead += sizeof(unsigned short);

\*((int\*)pHead) = length;

for(int i=0; i<sizeof(head); i++)

sum = (sum+ head[i] ) % 65535;;

for(int i = 0; i<length; i++)

sum = (sum+data[i]) % 65535;

return sum;

}

unsigned short CProtocol::CheckSum()

{

unsigned short sum = 0;

for (int i = 0; i<sizeof(m\_packHead); i++)

sum =(sum+ m\_packHead[i]) % 65535;

for (int i = 0; i<m\_PacketStruct.dataLength; i++)

sum = (sum+ m\_PacketStruct.data[i]) % 65535;

return sum;

}

* 示例

start= 0xAABB

cmmd = 0x0000

serialNo = 0x0006

dataLength =0x0008

data指向一个结构体 TagClientDataStruct， 赋值如下 ：

netId=0x1001

netAddr=0x2002

event =0x0000 ;

steps =0x0100

该完整的数据包是

0xaa 0xbb 0x0000 0x0006 0x0008 0 x1001 0x2002 0x0000 0x0100

去掉前缀0x 后 拷贝到串口调试助手，按十六进制 发送 即可

aa bb 0000 0006 0008 1001 2002 0000 0100

================checksum 计算方法=================================

TagClientDataStruct dataSt ;

dataSt .netId=0x1001

dataSt .netAddr=0x2002

dataSt .event =0x0000 ;

dataSt .steps =0x0100

data = (char\*) & dataSt;

checksum= CheckSum( cmmd, serial, length, data) = 0x0042

* 生成序列号算法

static unsigned short GetSerialNo()

{

if (false)

{

static unsigned short num = 0;

num = (num + 1) % 65535;

return num;

}

else

return 1; //for test purpose，， 测试初期暂时先固定为1 ， 以方便调试报文,

}

* **上位机生成控制命令缓冲区buffer的示例算法代码：**

WSABUF wbuf;

//OVERLAPPED ol;

//ZeroMemory(&ol, sizeof(OVERLAPPED));

WSABUF \*p\_wbuf = &wbuf;

//OVERLAPPED \*p\_ol = &ol;

DWORD dwBytes = 0, dwFlags = 0;

char buffer[512];

char \* pbuf = p\_wbuf->buf = buffer;

p\_wbuf->len = 0;

\*(unsigned short\*)pbuf = htons(0xAABB);

p\_wbuf->len += 2;

pbuf += 2;

controlCode = pcs->controlCode == 1 ? CMMD\_CONTROL\_ON : CMMD\_CONTROL\_OFF;

\*(unsigned short\*)pbuf = htons(controlCode); //cmmd

p\_wbuf->len += 2;

pbuf += 2;

\*(unsigned short\*)pbuf = htons(CProtocol::GetSerialNo());

// GetSerialNo初始测试期间，暂定为1 ，

p\_wbuf->len += 2; pbuf += 2;

\*(unsigned short\*)pbuf = htons(4); // data length

p\_wbuf->len += 2; pbuf += 2;

\*(unsigned short\*)pbuf = htons(pcs->netId);//0x2001

p\_wbuf->len += 2; pbuf += 2;

\*(unsigned short\*)pbuf = htons(pcs->netAddr); //0x2002

p\_wbuf->len += 2; pbuf += 2;

//unsigned int check = htons(pSocketOlp->m\_cp.CheckSum(pfs.cmmd, pfs.serialNo, pfs.dataLength, pfs.data));

unsigned int check = htons(pSocketOlp->m\_cp.CheckSum(p\_wbuf->buf+2, p\_wbuf->len-2));

\*(unsigned short\*)pbuf = check;

p\_wbuf->len += 2; //pbuf += 2;

// on---- AA BB 00 02 00 01 00 04 20 01 20 02 00 4A

//off--- AA BB 00 03 00 01 00 04 20 01 20 02 00 4B

//int nBytesSent = Send(pSocketOlp->m\_socket, p\_wbuf, 1, &dwBytes, dwFlags, p\_ol, NULL);

**下位机上报的心跳数据包**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 长度  byte | 说明 |
| **netId** | unsigned short | 2 | zigbee网络号 |
| **netAddr** | unsigned short | 2 | zigbee网址 |
| **time** | Char[19] | 19 | 时间字符串：2016-11-29 10:50:20 |

**五 数据库表定义：**

table\_log

|  |  |  |
| --- | --- | --- |
| Column name | Definition | Memo |
| table\_id | int NOT NULL | 主表记录id，单向递增整型 |
| event\_id | int | 事件id |
| pan\_id | nchar (10), | Zigbee 网络号 ，存放的是16进制的数字的 ascii字符串 |
| zigbee\_address | nchar (10), | Zigbee 地址，存放的是16进制的数字的 ascii字符串 |
| Ip | nchar (13), | 以太网ip |
| steps | int, | 行走步长 |
| employee\_no | nchar(10) |  |
| process\_no | nchar(24) |  |
| model\_no | nchar(8) |  |
| create\_time | datetime | 事件生成日期 |

table\_device

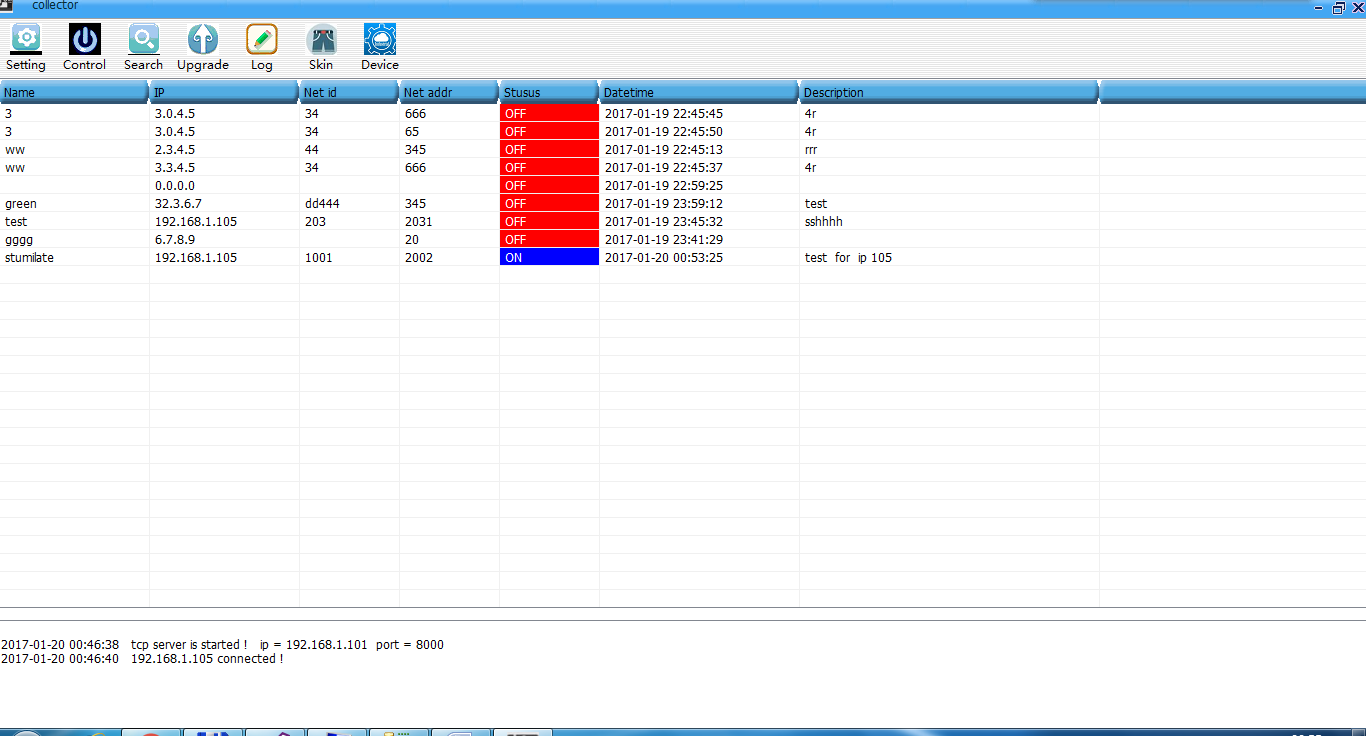
|  |  |  |
| --- | --- | --- |
| Column name | Definition | Memo |
| table\_id | int NOT NULL | 主表记录id，单向递增整型 |
| name | varchar(20) | 设备名称 |
| description | varchar(20) | 描述 |
| type | Tinyint | 设备类型，网关 or 终端 |
| Ip | nchar (16), | 以太网ip |
| pan\_id | nchar(10) | 网关的Zigbee 主节点id， 16进制 ，不带前缀0X |
| zigbee\_address | nchar(10) | Zigbee 普通终端 地址，16进制 ，不带前缀0X |
| device\_status | varchar(15) | 设备状态，on ，off ， |
| time | datetime | 设备生成的时间，或者最时间近收到心跳包的 |
| other | varchar(20) | 其它 |

table\_status

|  |  |  |
| --- | --- | --- |
| Column name | Definition | Memo |
| table\_id | int NOT NULL | 表记录id，单向递增整型 |
| message | nchar (50) | 消息 |

**六 软件主要界面截图 和功能说明**：

* 1 主界面



上半部列表显示实时的采集数据，

下半部显示以太网设备的连接和连接关闭状态。

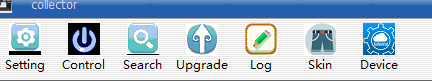
上半部列表可以点击头部排序

在主界面 设备列表中 用鼠标双击 一条记录， 则打开该设备的日志记录

设备状态刷新周期 在 设备管理（Device）的定时器（Timer）参数设置 ，

认定设备离线的时间参数在 setting 中配置

* 2工具栏，



点击工具栏的按钮（setting ,control,search,upgrade）, 切换到对应的对话框页面，进行相应的操作

要对某个页面进行操作，请从工具栏点击按钮进入,,否则可能导致apply按钮失效。

Log 显示日志文件

Skin 切换软件界面的皮肤， 一共有4种皮肤可以切换

Device 设备管理



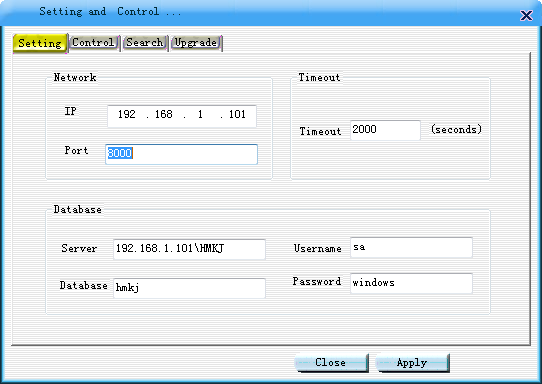
上图是 串口调试工具的截图

串口工具测试样本的时候，请先勾选按十六进制发送， 然后再粘贴要发送的十六进制测试数据

* 3 Setting -服务器ip地址和端口设置 ，数据库 参数设置

设置栏目做修改，并apply 以后，参数会在重启软件后生效。

配置参数存放在本地文件config.ini中



Timeout 用于设置 终端多长时间没有上报心跳包，就认为是离线的

Ip 和 port 用于设置tcp server 的 ip和端口参数， 本软件为tcp server

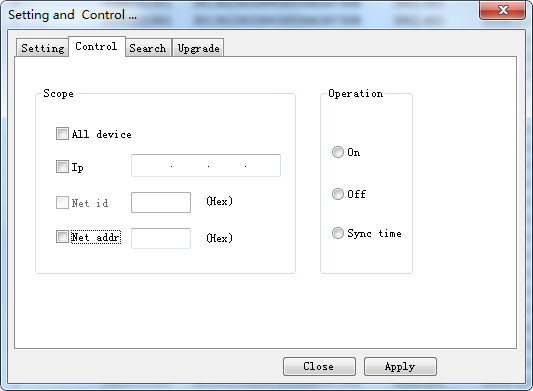
Database 用于设置 数据库服务器的参数

* 4 Control - 上位机向下位机发送指令， 控制下位机 ，

Operation 是控制命令，比如停机off，开机on， 同步设备日期和时间

设备范围scope可以选择， 全部设备， 指定ip 下的所有设备， （或者某个zigbee id 下的所有设备）， 某个zigbee设备 ，

选择设备的范围 hex 表示输入的是十六进制数据



如果想要控制所有的设备，那么勾选“all device ”就可以了，其他的复选框不用再勾选 ,

如果想要控制某个net id 下所有的设备，那么勾选“Ip ” 或者 “Net id”就可以了，其他的复选框不用再勾选。

如果想要控制某个net id 下某个的设备，那么勾选“Ip ” （或者 “Net id” ）， “Net addr” 。

本质上，Ip 和一个 Net id 是具备一一对应的关系， 所以界面上把net id 屏蔽了，不让用户选择 。

**通讯协议约定：**

勾选ip， 但不勾选Net addr，也就是指定ip或者 net id，但是不指定Net addr，则Net addr置为0

Net addr置为0，表示组内广播，操作对该Net id下的所有 zigbee终端有效。

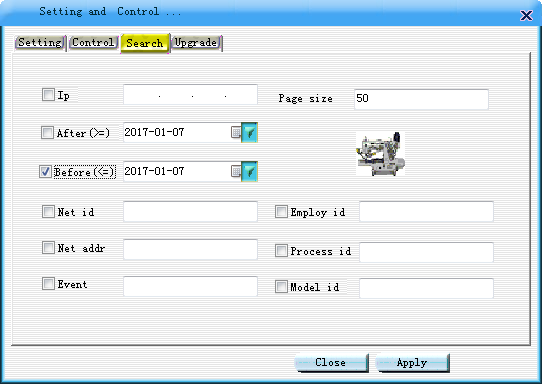
===============================================================================

* 5 Serach --- 用来检索数据库中的数据，并显示， 注意过滤条件的组合使用，各个条件之间是 逻辑与 （AND）关系

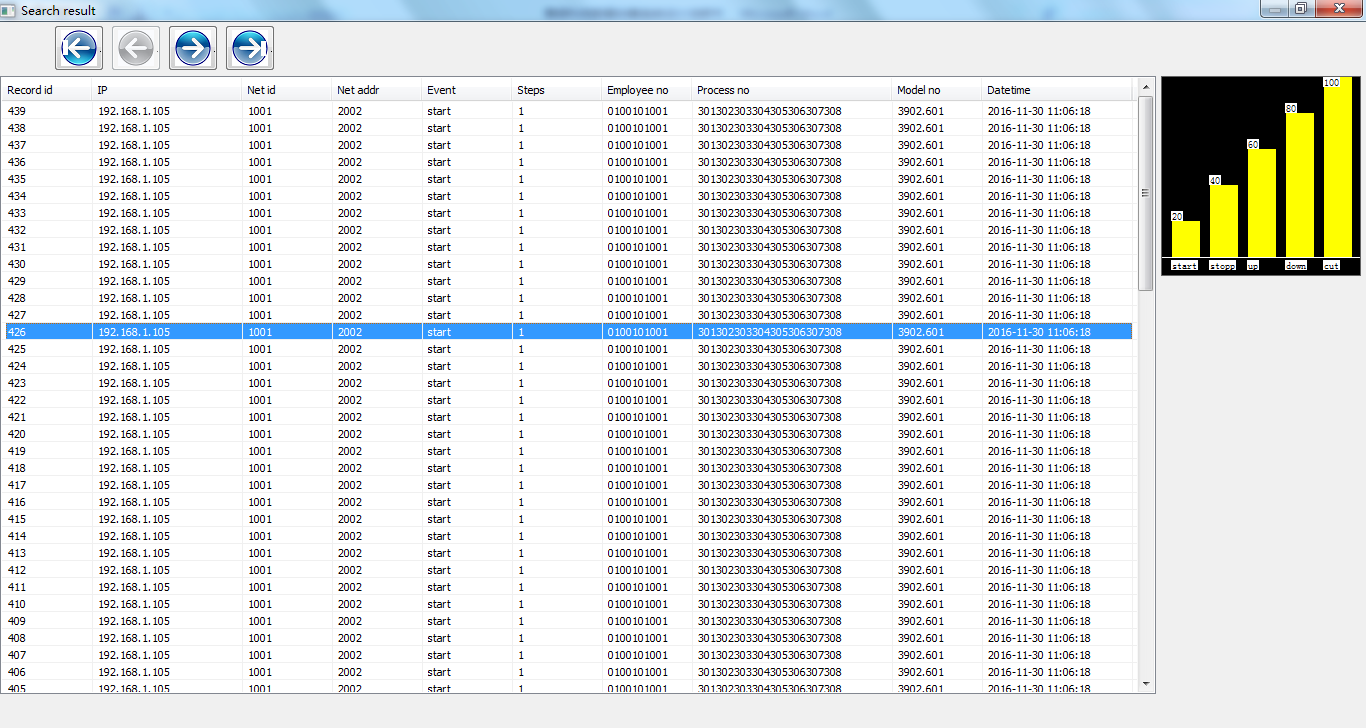
时间条件是都大于等于 ，或者小于等于， 日期为选择的日期，时间为当前时间

如果没有勾选任何过滤条件，那么默认的搜索条件是 ，搜索 当前日期和时间 到前一天当前时间， 这段时间内的记录。

Page count 指定检索界面每个页面显示记录的数目 ，该配置参数会保存在本地配置文件config.ini中， 最大数值因为系统内存的原因和速度的考虑，每页面最大记录数目限定为不超过2000

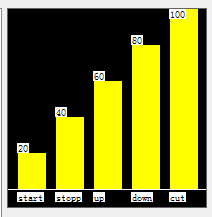


上图的搜索条件 搜索结果 示例, 点击列表头部列名，可以按指定的列重新排序





四个按钮是翻页按键，从左到右分别是 first, previous , next ,last

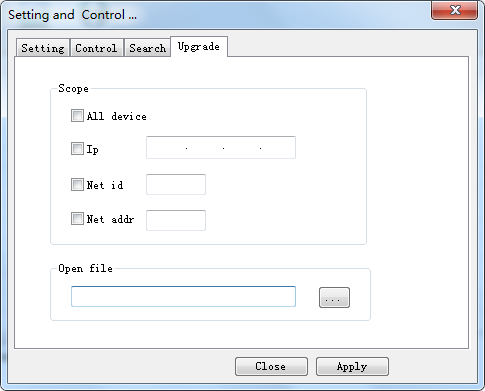


右上角的图形显示该搜索条件下的 event出现次数的统计

* 6 Upgrade ---用来对指定范围的设备进行升级，

设备范围的语义和 control 页面的一致

Open File 打开本地电脑上的二进制文件， 下载到设备更新系统



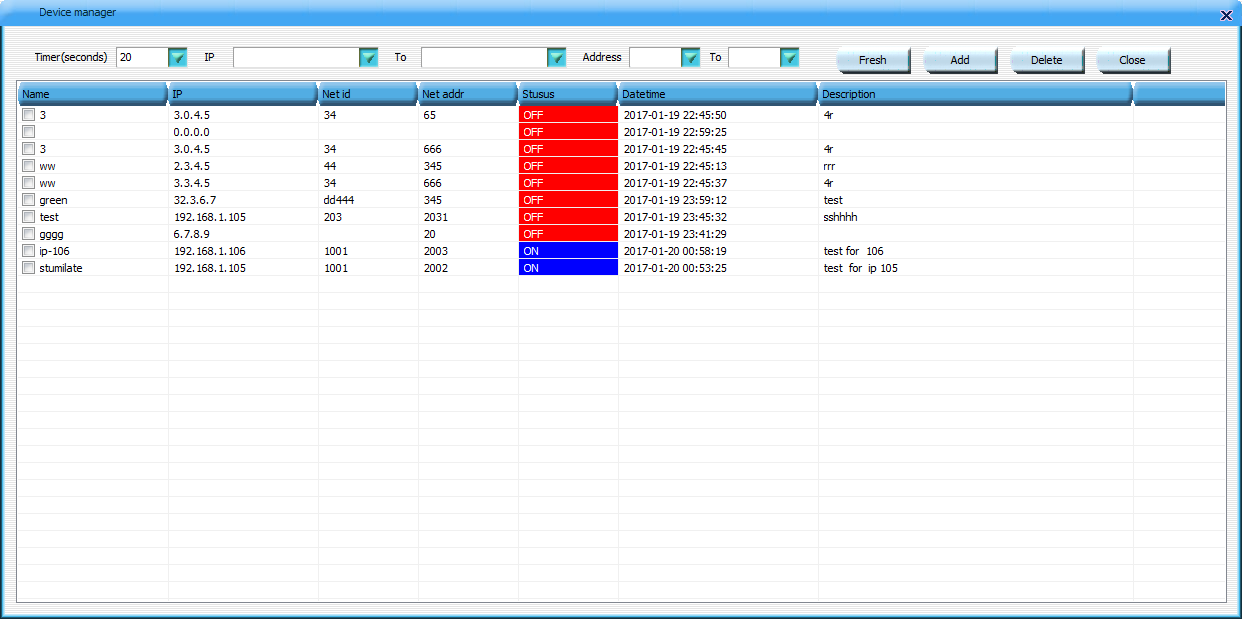
* 7 点击Log ， 打开日志文本文件

日志文件主要记录 软件启动，关闭信息，zigbee网关tcp连接信息， 下位机对上位机发出的控制命令的回馈信息 ， 数据库信息 等 。

日志文件log.txt存储在本地电脑



* 8 点击Skin 按钮 ，切换软件界面的皮肤 ，循环切换
* 9 device 设备管理



定时器timer ，设置设备列表刷新周期， 时间以秒为单位

增加按钮Add ，用于添加一个终端 到数据库，添加后，点击刷新按钮，或者等待自动刷新时间到， 设备列表会跟新新添加的设备

删除按钮 Delete， 用于删除一个终端设备， 数据库和 设备列表 立即更新

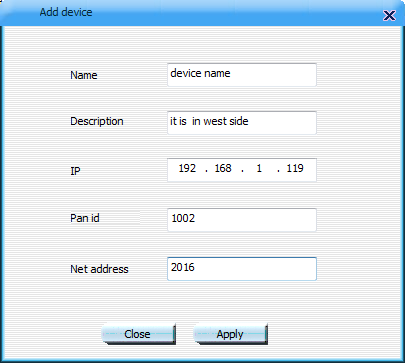
刷新按钮Fresh ， 用于刷新设备列表

组合列表组件 Ip 和Address 用于筛选需要列表显示的设备，

刷新周期 和过滤器 的配置参数， 会长期保存在电脑中， 软件主界面的刷新周期 和过滤器 的配置参数和本 界面中的配置参数是一致的。

鼠标双击列表中的一条记录， 则打开该设备对应的日志记录

* 10 增加一个zigbee 终端设备



设备名称name ， 输入长度20个字符

描述 description ， 输入长度 限制20个字符

Ip 和 pan id 是一一对应关系，

pan id 和 net address 请输入十六进制数值， 数值输入范围0 ~ ffffffff

* 软件界面图标，尺寸 32x 32 像素，8位256色， 格式为bmp位图

**七 软件对不同操作系统的支持**

支持32位和 64 windows系统， 支持xp ， Vista， win7 ，win8 ，win10

32位和 64 位 需要做成2个不同版本的软件，

Xp 系统的和 其它的系统的需要做成2个不同版本的软件

所以软件一共需要做4个版本

**八 软件对多语言的支持**

目前语言版本统一为英文版

**九 系统性能保障**：

1 系统响应及时性：

中央服务器，数据中心服务器，数据库服务器，三者根据用户负载决定是否分离或者合并，或者做负载平衡；

采用高效的数据采集算法，多线程异步处理高并发；

数据库查询优化。

2 系统安全性：开启防火墙

3 系统可靠性：可以通过数据库库备份实现

4 系统的可扩展性：目前的推荐配置可以支持数万的终端设备同时工作，如果终端数量大量增加，只需要配置更高性能的硬件即可，软件不需要任何改动，系统具备基本的可靠性的同时，具有灵活的可扩展性。

**十 开发工具**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 主程序IDE | 数据库 | 代码仓库 | 项目管理工具 | 软件测试管理平台 | 文档编写工具 |
| VS studio 2015 C++ | MS Sql server 2012 | SVN | MS project  MS excel | Bugzilla | MS Word，  MS Viso |

**十一 管理系统软件运行环境，推荐配置**

1 服务器主机配置：和系统设备的数量相关，网内终端设备多的话，可以相应的提高配置，以下为1000个终端设备的时候的推荐配置

|  |  |
| --- | --- |
| 操作系统 | win server 2012 |
| CPU | Intel Xeon E5系列/ i7系列处理器 |
| 内存 | 16G |
| 硬盘 | 500M |
| 网速 | 100M |

2 数据库服务器配置：和系统设备的数量相关，网内终端设备多的话，可以相应的提高配置，以下为1000个终端设备的时候的推荐配置。

应用服务器和数据库服务器可以独立，也可以合并。 数据库也可以部署在客户原有的ERP系统的数据库服务器上。这个跟客户沟通后决定

|  |  |
| --- | --- |
| 操作系统 | windows server2012 |
| CPU | Intel Xeon E5系列/ i7系列处理器 |
| 内存 | 16G |
| 硬盘 | 1 T |
| 网速 | 100M |
| 数据库 | sql server 2012  如果需要和客户的数据库一致的话，则用客户推荐的数据库 |

**十二 项目开发计划**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 大类 | 小类 | 功能说明 |
| 1 | 需求分析 | 需求调研 | 客户对接，需求沟通，需求确认  需求沟通将会花费大量的时间 |
| 需求分析 | 需求功能分解 |
| 需求方案设计 | 编写需求方案说明书 |
| 2 | 系统设计 | 构架设计 | 系统构架设计和评审 |
| 概要设计 | 概要设计和评审 |
| 详细设计 | 详细设计和评审 |
| 数据库设计 | 数据模型和评审 |
| 3 | 系统测试 | 测试方案 |  |
| 单元测试 |  |
| 接口测试 |  |
| 集成测试 | 试运行 |
| 4 | 系统部署 | 系统部署 | 正式部署产品 |
| 系统初始化（包括数据迁移） |  |
| 部署文档编写 |  |
| 5 | 系统运维 | 运行维护 |  |
| 使用说明书 |  |