ZigBee

本次的硬件选型是使用的是德州仪器公司（Texas Instruments）旗下的CC2530型号Zigbee作为主控芯片。CC2530通信在2.4G HZ的频段，遵守IEEE802.15.4协议，进行无线组网通信。同时，Zigbee和RF4CE是应用的一个真正的片上系统（SoC）解决方案。CC2530结合了领先的RF收发器的优良性能，业界标准的增强型MCU 8051单片机可以作为主控芯片。系统内可编程闪存，8KB RAM和其它许多强大的功能。CC2530可以运行不同的工作模式，使得它尤其适应超低功耗要求的系统，它能够以非常低的总的材料成本建立强大的网络节点。在工业采集方面上，CC2530能够胜任小型局域网的通信要求。

Zigbee和IEEE802.15.4是基于标准的协议，为无线传感器网络应用程序提供所需的网络基础设施。802.15.4定义物理层（PHY）和媒体访问控制（MAC）层，Zigbee定义网络层（NWK）和应用层（APL）。

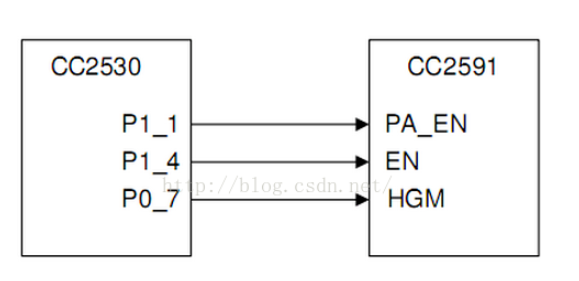
对于传感器进行无线组网应用而言，最主要的设计就是要求低成本、安全、可靠和自愈、灵活可扩展、低功耗、部署容易且低成本等问题，这样才能支持在一个应用环境中设备相互通信传输数据。Zigbee无线网状网络对于许多的应用来说是非常的适用。它能够促进楼宇自动化管理、个人健康看护、工业控制、移动商务信息服务、家居控制的智能化发展。

2.4GHz的CC253x片上系统解决方案适合于广泛的应用。它们可以很容易建立在基于IEEE802.15.4标准协议（RemoTI网络协议、TIMAC软件和用于ZigBee兼容解决方案的Z-Stack软件）上面，或是专门的SimpliciTI网络协议上面。但是它们的使用不限于这些协议。

CC2530设备系列使用的8051 CPU内核是一个单周期的8051兼容内核。它有三个不同的存储器访问总线（SFR、DATA 和CODE/XDATA），以单周期访问SFR、DATA 和主SRAM。它还包括一个调试接口和一个18个输入的扩展中断单元。USART 0和USART 1每个被配置为一个SPI主/从或一个UART。它们为RX和TX提供了双缓冲，以及硬件流控制，因此非常适合于高吞吐量的全双工应用。每个都有自己的高精度波特率发生器，因此可以使普通定时器空闲出来用作其他用途。

三个CC2530终端节点来驱动温度传感器、振动传感器、压力传感器和继电器。通过组网的形式，组建以zigbee为无线通信的局域网继而通信。节点一的Zigbee终端采集压力数据，节点二的Zigbee终端采集振动传感器数据，节点三的Zigbee终端采集温度数据。终端节点一，二通过路由器1转发到协调器上。终端节点三是通过路由器2转发到协调器上。协调器负责接收数据，并且控制继电器开关量。协调器通过与4G模组通过串口建立起连接，通过透传模式，上传数据至服务器端。这样，Zigbee无线组网的基本通信功能就实现了。至于继电器的控制则放置于协调器进行控制，当上报给到Zigbee协调器的温度数据高于等于35摄氏度以上时，继电器维持原来状态。当温度小于等于25摄氏度时，继电器自动关闭。Zigbee协调器通过4G模块传输数据至服务器端后，用户可以在页面进行按钮点击发出一个打开/关闭继电器的信号，再通过服务器端传输回给4G模组，4G模组将接收到的数据通过串口的形式再次传输给Zigbee协调器。协调器通过比对之前约定好的数据帧格式，若接收到与之前约定好的数据帧格式正确的数据帧之后，取出里面负责控制继电器的信号数据，Zigbee协调器控制与继电器相连接的引脚来控制继电器的开关状态。

CC2530+CC2590(PA功率放大芯片)

CC2590是一款经济高效的高性能RF前端，适用于低功耗和低电压2.4GHz无线应用。CC2590是德州仪器（TI）所有现有和未来2.4GHz低功耗RF收发器，发送器和偏上系统产品的范围扩展器。CC2590通过提供增加输出功率的功率放大器和低噪声系数的LNA来提高链路预算，从而提高接收器灵敏度。CC2590采用4\*4-mm QFN-16封装，提供小尺寸，高输出功率RF设计。CC2590包含PA,LNA,RF匹配和巴伦，可用于高性能无线应用的简单设计。

CC2590有三个脚和CC2530连接，如果硬件管脚连接的是其他的管脚则需要在协议栈上修改。值得注意的是PA\_EN与EN脚只能连接在P1\_0到P1\_5之间，这是由CC2530决定的。HGM可以任意定义。此次设计中，选用的是Zigbee核心底板是CC2530+PA模块的。所以，带PA模块，若没有开启PA功能的话，通信距离十分地近，不能够达到实际实用范围。所以，在Zstack协议栈中，就需要打开PA功能。首先，找到hal\_board\_cfg.h文件中的#define Xhal\_PA\_LAN行，把x去掉，就是带有PA。打开mac\_radio\_defs.c文件，可以查看到带有PA芯片的模块功率放大表。依次对应放大功率从10dBm到20dBm对应的寄存器设置参数。最后，在mac\_radio.c文件中找到设置功率放大的函数MAC\_INTERNAL\_API uint8 macRadioSetTxPower(uint8 txPower)。按照在mac\_radio\_defs.c文件中看到的功率放大表的参数来修改Zigbee需要的发射的功率大小，把需要设置的功率赋值给reqTxPower参数即可。比如，reqTxPower = 0XE5。如此一来，这样就打开了PA模块，Zigbee无限通信也可以连接上了，根据测试结果。打开PA模块之后，不加入路由器转发的情况下，终端节点和协调器互相通信的距离大概在20~30米范围左右。（该距离与模组的外接天线有关系，测试时是使用的ipx插口的外接天线。）

4G模组

还需将局域网内的Zigbee协调器的数据上传至服务器端。本次方案中采用的是4G模组SIM7600CE。芯讯通无线科技有限公司（SIMCom Wireless Solutions）是全球领先的M2M模块及解决方案供应商。自2002年成立以来，一直致力于提供GSM/GPRS/EDGE,WCDMA/HSPA/HSPA+,CDMA，1xRTT/EV-DO,FDD/TDD-LTE,eMTC(CAT-M1),NB-IoT无线蜂窝通信以及GPS/GLONASS/BEIDOU卫星定位等多种技术平台的模块或终端级解决方案。SIM7600CE是一款SMT封装的模块，支持支持LTE-TDD/LTE-FDD/HSPA+/TD-SCDMA/EVDO和GSM/GPRS/EDGE等频段, 支持LTE CAT4（下行速度为150Mbps）。其性能稳定，外观小巧，性价比高，可以低功耗实现SMS和数据信息的传输。SIM7600CE尺寸为30\*30\*2.9mm，能适用于各种紧凑型产品设计需求，能满足客户的多种需求。主要特性 TDD-LTE B38/B39/B40/B41；FDD-LTE B1/B3/B8；TD-SCDMA B34/B39；WCDMA/HSDPA/HSPA+ B1/B8；CDMA 1X/EVDO BC0；GSM/GPRS/EDGE 900/1800 MHz；通过AT命令控制；供电电压范围: 3.4V~ 4.2V；操作温度范围: -40℃ to +85℃；三维尺寸: 30\*30\*2.9 mm；重量: 5.5g ；GNSS gpsOne Gen 8B;standalone;assisted,XTRA。接口提供的有USB2.0,UART,SIM card,SPI,I2C,GPIO,ADC,PCM,SDIO接口。由于通信中每帧数据字节数小，用UART串口和Zigbee协调器进行互相通信，串口的通信速率足够能够达到要求。4G模组烧录的是AT固件，使用AT命令进行配置。

在电路板上电之前，务必先将SIM卡插在卡槽中，这样4G模组才能正常供电启动。上电之后，首先发送AT语句来检测4G模组上电是否启动成功的状态。while(!(Sim7600\_SendCmd("AT\r\n","OK",1000)))语句用来检测4G模组的AT固件是否正常运行。

Sim7600\_SendCmd("AT+COPS?\r\n","OK",1000);用来检测4G模组是否检测到SIM卡插在卡槽中。while(!(Sim7600\_SendCmd("AT+CGDCONT=1,\"IP\",\"3gNET\"\r\n","OK",1000)))语句用来配置4G模组的注网过程。Sim7600\_SendCmd("AT+CIPMODE=1\r\n","OK",1000)语句用来配置4G模组进入透传模式，使用透传模式进行上传数据，在透传模式中，上传数据过程就可以不用管理数据的格式，怎么上传的数据，在接收端就是接收到一样的数据格式。Sim7600\_SendCmd("AT+NETOPEN\r\n","OK",1000)语句用来打开4G模块的套接字。while(!(Sim7600\_SendCmd("AT+CIPOPEN=0,\"TCP\",\"120.78.173.188\",8080\r\n","CONNECT 115200",1500)))语句用来配置网络传输的网络协议为TCP，服务器的IP和端口号，与配置的串口的波特率为115200的速率。由于在联网过程中，有可能就是之前输入的远程服务器IP或者端口号错误，可能会碰到服务器端出现假连接的状态或者是未插上SIM卡的情况发生，这样，就会一直无法连接上服务器。所以在配置网络传输的循环语句中，加入了cnt作为标志位。

cnt++;

if(cnt==9)

{

Sim7600\_SendCmd("+++","OK",1500);

Sim7600\_SendCmd("AT+NETCLOSE\r\n","OK",1500);

printf("fail to connect 4G,reconnecting...\r\n");

break;

}

当循环10次时，若还是不能连接上服务器，就将使用AT指令Sim7600\_SendCmd("+++","OK",1500)语句配置首先推出透传模式，Sim7600\_SendCmd("AT+NETCLOSE\r\n","OK",1500)语句配置关闭网络，并且通过Zigbee的串口二打印出"fail to connect 4G,reconnecting..."语句，提示用户此时配置4G模组上网功能配置失败。

另外，由于在实际生产环境中，整套设备需要长时间保持联网状态。可是难免会发生网络错误情况，一旦发生该情况，4G模组与服务器端连接失败 ，此时再派人拿取设备返回，不仅采集不到数据，并且是一件很浪费人力的事情。所以，在一开始的4G模组配网中，添加上心跳包功能的配置。Sim7600\_SendCmd("AT+CTCPKA=1,3,3\r\n","OK",1500);语句就是打开了4G模组与服务器通信的心跳包功能。此AT语句配置为打开心跳包功能，以每15分钟的频率向服务器端发送一次心跳包，重复发送3次。那么什么是心跳包呢？是这样规定的，网络中的接收和发送数据都是使用SOCKET进行实现。但是如果此套接字已经断开，那发送数据和接收数据的时候就一定会有问题。可是如何判断这个套接字是否还可以使用呢？这个就需要在通信系统中创建心跳机制。其实TCP中已经为我们实现了一个叫做心跳的机制。如果你设置了心跳，那TCP就会在一定的时间（比如设置的是3秒钟）内发送你设置的次数的心跳（比如说2次），并且此信息不会影响你自己定义的协议。所谓“心跳”就是定时发送一个自定义的结构体（心跳包或心跳帧），让对方知道自己“在线”， 以确保链接的有效性。若网络出现了异常断开，此时服务器端将在规定的时间间隔内没有接收到规定好的心跳包，此时，服务器端就认为客户端已经断开了网络连接。并且客户端一段时间内也接收不到服务器端发送过来的心跳包的应答，也会判断出网络已经断开，即将与服务器重新连接。

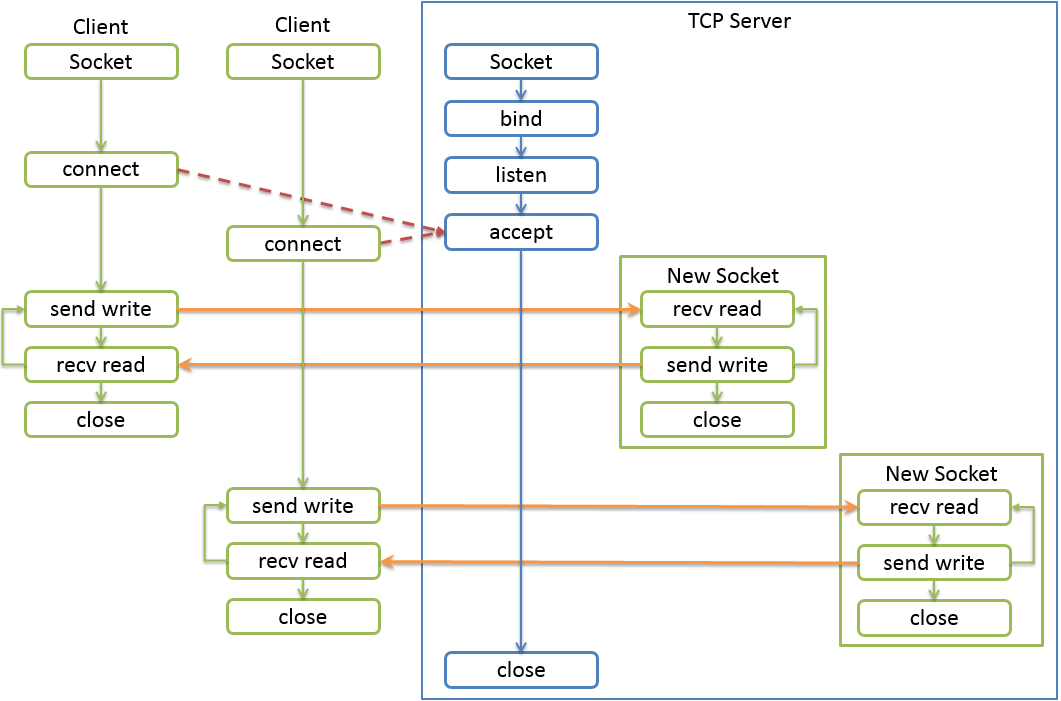
服务器开发语言

服务器和前端方向选择Python作为开发语言，是用来编写应用程序的一种高级语言。Python是由荷兰人Guido van Rossum于1989年创立，1991年初发布第一个公开发行版。Python有2.x版本，发展到了2.7就停止维护了，至今为止Python2.7还有10个月零19天将会退休。而Python3.x版本至今发行到了3.6。3.x版本始自2008年，3.5版本发布于2015年。

Python为我们提供了非常完善的基础代码库，覆盖了网络、文件、GUI、数据库、文本等大量内容，被Python开发，许多功能不必从零开始编写，直接使用现有的库进行开发即可。除了已有的内置的库功能帮助开发外，Python还有大量的第三方库，供开发人员进行直接使用学习。Python的哲学就是简单优雅，尽量和写成容易能看明白的代码，尽量和写少的代码。所以Python程序对于初学者可以说是，简单且容易入门。Python尤其适用于网络应用开发，包括网站、后台服务等等。因为Python是跨平台的，它可以运行在Windows、各种Linux/Unix和Mac系统上。在Windows上写的Python程序，放到Linux服务器上也是可以运行的。所以就不会牵扯到跨平台编译不通过的问题了。

本次开发方案决定选择Python3.7进行开发。网络接口包括后台服务器、数据库以及网站选用Python3.7进行开发。编译环境选择Pycharm。Pycharm是一个用于计算机编程的集成开发环境，主要用于Python语言开发，由捷克公司JetBrains开发，提供代码分析、图形化调试器、集成测试器、集成版本控制系统，并支持使用Flask进行网页开发。Pycharm是一个跨平台开发环境，拥有Windows,Linux,MAC版本。

服务器端网络编程



在4G模组与服务器端，采用TCP/IP通信协议来进行通信传输数据。Socket变成，需要两端，一方作为服务器端，另一方就作为客户端。服务器端称为Server，客户端称为Client。

4G模组作为客户端的角色以TCP/IP协议通过透传的方式不断地向服务器端发送采集到的传感器数据。而服务器端，则要建立起监听客户端上传的数据的模式。在客户端向服务器端发送数据之前，服务器端就应该已经做好了开启监听模式的工作，等待客户端的连接并能接收处理之前双方定义好的数据帧格式。

TCP编程

Socket是网络编程的一个抽象的概念。通常我们使用一个Socket表示打开了一个网络链接，而打开一个Socket需要知道目标计算机的IP地址和端口号，再指定通信协议类型就可以了。

1. Socket介绍

Socket套接字，Python中提供了socket.py标准库，非常底层的接口库用来进行网络编程。Socket是一种通用的网络编程接口，和网络层次没有一一对应的关系。在调用一个socket()函数中，就涉及到了两个参数。

协议族，AF表示Address Family,用于socket()函数的第一个参数。

|  |  |
| --- | --- |
| 名称 | 含义 |
| AF\_INET | IPV4 |
| AF\_INET6 | IPV6 |
| AF\_UNIX | Unix Domain Socket,windows没有 |

Socket类型

|  |  |
| --- | --- |
| 名称 | 含义 |
| SOCK\_STREAM | 面向连接的流套接字。默认值，TCP协议 |
| SOCK\_DGRAM | 无连接的数据报文套接字。UDP协议 |

1. Socket启动过程

首先，创建一个Socket对象，绑定服务器的IP地址和唯一的未被其他程序占用的端口Port。调用bind()方法，IPV4地址为一个二元组(‘IP地址字符串’，Port)。绑定好之后，服务器就开始监听，将在指定的IP的端口上监听。调用listen（）方法。之后，若有客户端通过TCP/IP协议连接服务器端，进行通信。服务器端通过调用socket.accept()方法获取用于传送数据的Socket对象。Accept方法阻塞等待客户端建立连接，返回一个新的Socket对象和客户端地址的二元组地址是远程客户端的地址，IPV4中它是一个二元组（clientaddr,port）。之后，就可以调用recv()方法来接收数据了。在测试中，可以知道accept和recv方法都是阻塞的。这样一来，主线程经常被阻塞住而不能工作了。所以，可以想到这里就需要引入多线程来处理服务器socket的监听和接收。

1. SocketServer模块编程

Socket编程过于底层，编程虽然方法步骤固定简单，但是很难写出功能贴近实际生产使用的代码，所以Python语言就对socket底层API进行了封装。在Python库中就有了socketserver模块，它是网络服务编程框架，便于企业级快速开发。

BaseServer

UnixStreamServer

TCPServer

UnixDatagramServer

UDPServer

SocketServer简化了网络服务器的编写。它有4个同步类：TCPServer,UDPServer,UnixSteamServer,UnixDatagramServer。

Socketserver模块提供了Socketserver.BaseServer(server\_address,RequestHandlerClass)编程接口，需要提供服务器绑定的地址信息，和用于处理请求的RequestHandlerClass类。RequestHandlerClass类必须是BaseRequestHandler类的子类。

BaseRequestHandler类是和用户连接的用户请求处理类的基类，定义为BaseRequestHandler(request,client\_address,server)。服务器Server实例接收到用户请求后，最后会实例化这个类。它被初始化时，就送入了3个构造函数：request,client\_address,server本身。以后就可以在BaseRequestHandler类的实例上使用以下属性：self.request是和客户端的连接的socket对象，self.server是TCPServer实例本身，self.client\_address是客户端地址。这个类在初始化的时候，就会依次调用3个方法。定义的子类可以覆盖掉这些方法。

#BaseRequestHandler要自定义的子类覆盖的方法

**def** \_\_init\_\_(self, request, client\_address, server):  
 self.request = request  
 self.client\_address = client\_address  
 self.server = server  
 self.setup()  
 **try**:  
 self.handle()  
 **finally**:  
 self.finish()  
  
**def setup**(self): #每一个连接初始化

**pass  
  
def handle**(self): #每一次请求处理  
 **pass  
  
def finish**(self): #每一个连接清理  
 **pass**

在其中的handler()方法相当于socket的recv()方法。每个不同的连接上的请求过来之后，生成这个连接的socket对象即self.request，客户端地址是self.client\_address。

由于上面说到了BaseRequestHandler是会依次调用上面的三个方法的，当调用到finish()方法时，可以看到连接之后就立即断开了。所以，在handler方法中加入**while not** self.event.is\_set():在handler里面循环。

server = socketserver.ThreadingTCPServer(('0.0.0.0', 9999), MyHandler)。这里用的是ThreadingTCPServer。ThreadingTCPServer是多线程异步的，可以同时处理多个连接。TCPServer是同步的，一个连接处理完了，即一个连接的handler方法执行完了，才能处理另一个连接，且只有主线程。同步库，多个客户端都可以accept，但是相当于进入了队列中等待，先来先处理的原则，来处理recv,处理完第一个socket的recv后，继续之后的socket操作recv方法，尽管其他socket在等待，但是数据都是被缓存起来的。

在网络通信中，客户端会有断开连接，造成网络异常断开的情况发生。如果连接的线程中handler方法中抛出异常，，例如客户端主动断开导致的异常，线程崩溃。应添加执行self.clients的pop方法，捕获客户端的各种异常。通过测试打印可以看到，客户端主动断开，会导致recv方法立即返回一个空bytes，并没有同时抛出异常。当循环回到recv这一句的时候就会抛出异常。所以，下列语句

**if not** data:  
 print("Broken pipe")  
 **break**

可以通过判断data数据是否为空来判断客户端是否断开。若断开，则break退出循环。

所以，使用SocketServer创建服务器的过程如下。

3.1 从BaseRequestHandler类派生出子类，并覆盖其handler()方法来创建请求处理程序类，此方法将处理传入请求。

3.2 实例化一个服务器类，传参服务器的地址和请求处理类。

3.3 调用服务器实例的handler\_request()或serve\_forever()方法。

3.4 调用server\_close()关闭套接字，完成网络通信。

数据库

1.数据库

按照数据结构来组织、存储、管理数据的仓库。

计算机的发生是为了做科学计算的，而科学计算需要大量的数据输入和输出。早期，可以使用打孔卡片的孔、灯泡的亮灭来表示数据输入、输出。后来，数据可以存储在磁带上，顺序的读取、写入磁带。1956年IBM发明了磁带驱动器这个革命性产品，支持随机访问。随着信息化时代的到来，有了硬件存储技术的发展，有大量的数据需要存储和管理，数据库管理系统DBMS就诞生了。不管使用什么存储介质，数据库的数据模型才是其核心和基础。

分类：按照数据模型分类：网状数据库、层次数据库、关系型数据库。

这里，我们需要着重了解关系数据库。使用行、列组成的二维表来组织数据和关系，表中行（记录）即可以描述数据实体，也可以描述实体间关系。关系模型比网状模型、层次模型更简单，不需要关系数存储的物理细节，专心于数据的逻辑构建，而且关系模型有论文的严格的数学理论基础支撑。

1970年，IBM的研究员E.F.Codd发表了名为“A Relational Model of Data for Large Shared Data Banks”的论文，提出了关系模型的概念，奠定了关系模型的理论基础。关系模型，有严格的数学基础，抽象级别较高，简单清晰，便于理解和使用。经过几十年的发展，关系数据库百花齐放，技术日臻成熟和完善。基于关系模型构建的数据库系统成为RDBMS(Relational DataBase System)。IBM DB2、Oracle的Oracle和Mysql、微软的MS SQL。以前的informix、Sybase等。

2.MySQL

MySQL是一种关系型数据库管理软件，支持网络访问，默认服务端口3306。MySQL使用mysql协议。

3.安装

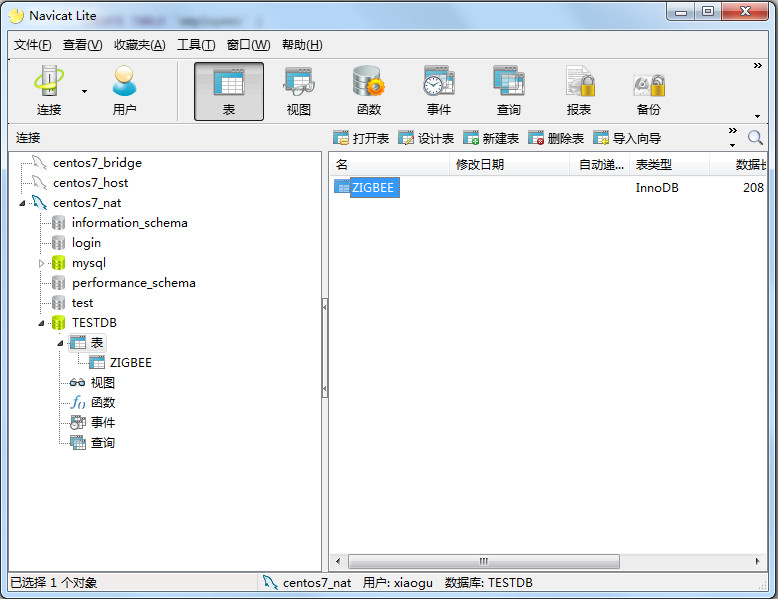
使用yum安装rpm包，推荐安装MariaDB。在服务器上安装好MariaDB，充当数据库服务器。yum install mariadb安装好数据库之后，service mysqld\_start开启服务器程序。

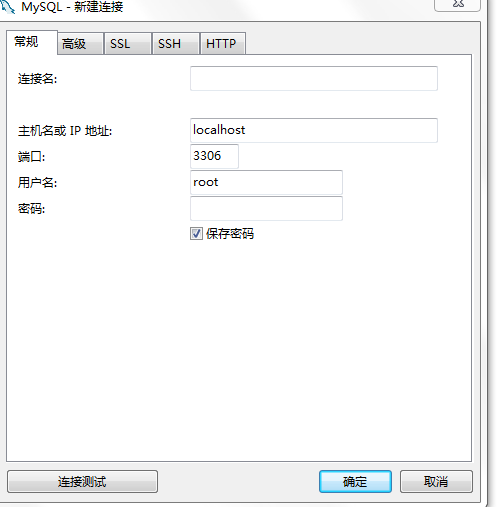
1. SQL语句

SQL是结构化查询语言Structured Query Language。1987年被ISO组织标准化。所有主流的关系型数据库多支持SQL,NoSQL也有很大一部分支持SQL。

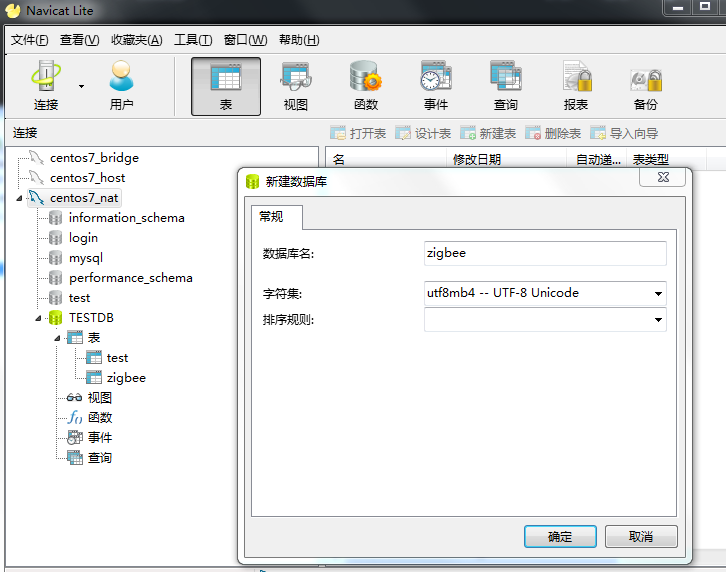
SQL语句分为：DDL数据定义语言，负责数据库定义、数据库对象定义，由CREATE、ALTER与DROP三个语法所组成。DML数据操作语言，负责对数据库对象的操作，CRUD增删改查。DCL数据控制语言，负责数据库权限访问控制，由GRANT和REVOKE两个指令组成。TCL事务控制语言，负责处理ACID事务，支持commit、rollback指令。SQL语句大小写不敏感。SQL语句末尾应该使用分号结束。

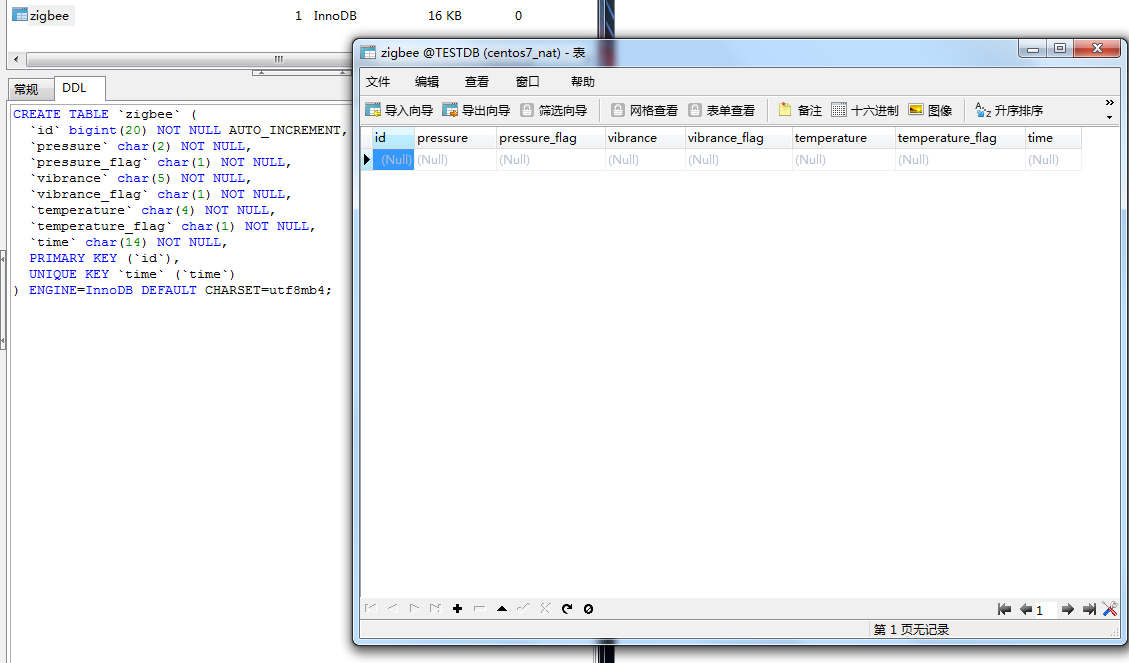
1. 数据库工具

Navicat Lite是一个多连接数据库开发工具。是一套快速、可靠的资料库管理工具。它是可以用图形界面来连接，操作自己的数据库。如上图所示，从左上方的文件，新建连接，选择MySQL之后，输入连接名，主机名或IP地址，端口号以及用户名和密码，确定之后即可连接到服务器上的数据库。



之后就可以在连接下，使用Navicat Lite工具的图形化界面来操作服务器上面的数据库了。首先新建一个数据库。如下图。



 在数据库底下，新建表，如下图。右键zigbee表，选择“对象信息”，单击DDL，可以看到生成的创建zigbee表的SQL语句。