**使用手册**

安装要求：

本数据库系统基于开源框架，需要安装如下软件：

1. redis
2. mongoDB
3. python3.4
4. python库：pip安装

minimalmodbus 0.6.0

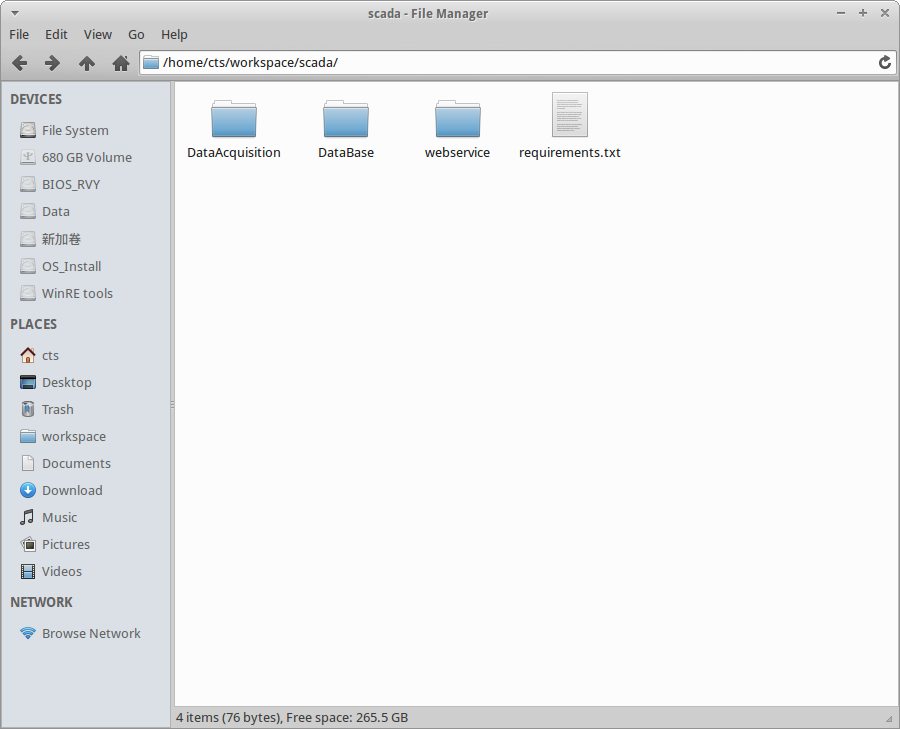
pymongo 2.7.1

pyserial 2.7

redis 2.10.1

tornado 2.7

以及相关依赖包。

Windows上没有官方的redis，可以通过自行编译方式取得，也可以不要redis直接使用mongoDB作为实时数据库。下面是本软件的文件:

DataAcquistion是客户端采集程序，DataBase是数据库接口，webservice是web服务程序。

图 系统软件全貌

服务端Ubuntu下安装：

数据库安装：

**mongoDB**:sudo apt-get mongoDB将自动下载安装mongoDB并且自动设置服务自启动。如果需要更新mongoDB版本，直接下载官网(http://www.mongodb.org/)linux 版本并覆盖到/usr/bin/下即可。

**Redis**:到http://www.redis.io/download下载源代码安装,根据readme的指示, 在解压后的路径下输入命令：

make

**32位**：

make 32bit

就可以完成安装，并可以用：

make test

进行测试。

**运行redis**:

cd src

./redis-server

可以用以下命令将**redis**安装进入 **/usr/local/bin**:

make install

你可以用"**make PREFIX=/some/other/directory install**" 装入其他目录。

为了自动启动redis服务，需要将**untils**下的**redis\_init\_script**复制到**/etc/init.d**内，重命名为**redis\_端口号**,默认端口**6379**。同时配置文件内的端口号要保持一致。

建立如下文件夹：**/etc/redis**用于存放配置文件;**/var/redis/端口号**　用于存放持久化文件。

配置文件模板在下载的**redis**根目录，目前不需要**redis**持久化功能所以配置文件里设定不需要存持久化文件，在**redis.conf**找到**save**的选项并按说明改成不存持久化文件,同时修改**daemonize**选项为**yes**,**pidfile**路径为**/var/run/redis\_端口号.pid,dir**为**/var/redis/端口号**,重命名为 **端口号.conf**（如**6379.conf**）并放入**/etc/redis/**。

最后，执行**sudo update-rc.d redis\_端口号 defaults**来使**redis**服务随系统启动。

python安装：

本系统软件全部采用python3.4开发，因此要安装python3.4。现在新版ubuntu自带python2.7与python3.4,因此必须区分，运行时要执行python3命令确保用python3.4执行。另外，在安装python库的时候也必须区分2.7与3.4,应使用如下命令安装python3的pip:

**sudo apt-get install python3-pip**

并且在安装其他库的时候用如下命令安装：

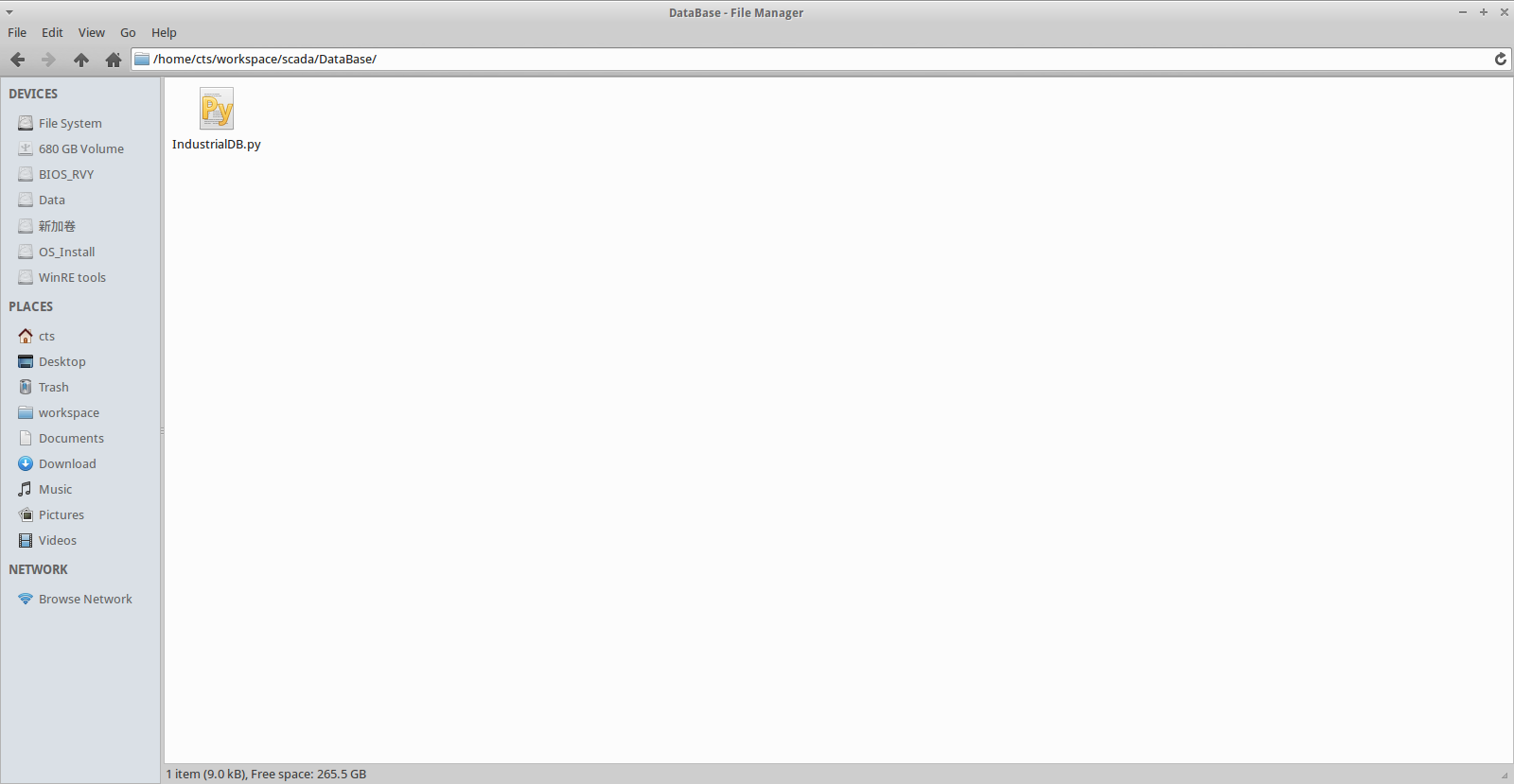
**sudo pip3 pakage\_name**

IndustrialDB api安装：

我们为实现特殊的mongoDB存储策略和文档结构对pymongo进行了封装，使用户可以通过十分简明的接口来进行数据库操作。

\*由于处于原型开发阶段，功能不是很完善，同时也不是很稳定，所以现阶段并未采用此接口，有可能以后还要进行很大调整。

将**database**文件夹下的**IndustrialDB.py**文件放入**/usr/local/lib/python3.4/dist-packages/**并赋予所有用户可读写执行的权限即可完成库的安装。



web服务安装：

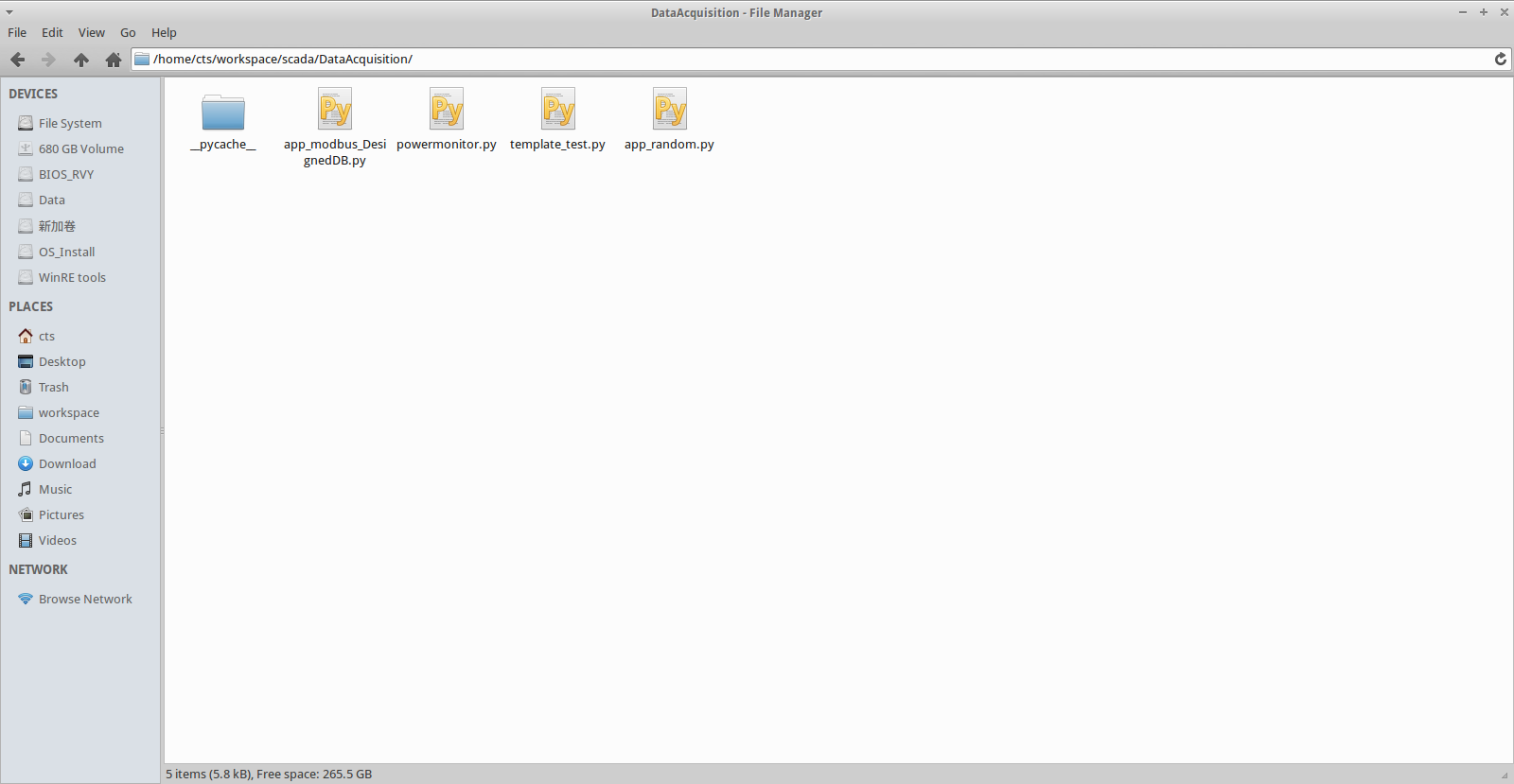
webservice文件夹包含了tornado服务所需要的网站文件。App.py是服务器主程序，用它来启动web服务，realtime.py和history.py是用来响应查询实时数据和历史数据的模块，api是当前使用的数据查询接口，static文件夹内是网站所需要的jquery、highstock等库文件；template文件夹内是静态网页模版。

只要直接启动app.py就建立起web服务器了，不需特别安装，只是需要用python3来运行。

客户端安装：

客户端采集程序采用python编写，因此可跨平台运行，前提是安装python3.4以及相关python的库，python库的安装步骤与服务端基本一致。完成后，DataAcquistion文件夹下的py程序就可以执行了。目前客户端程序支持串口、modbus RTU协议，也可自行编写采集程序。

DataAcquistion文件夹内有如下文件，app\_modbus\_DesignedDB是采用modbus协议向真实设备获取数据并采用我们开发的api存储数据的采集程序，powermonitor.py是一套电功率测量点的定义,属于点定义的范例，app\_random.py是采用简单型文档结构并用随机数测试的程序;template\_test.py是用于定义并生成IndustrialDB所采用的复杂文档结构的初始化程序。

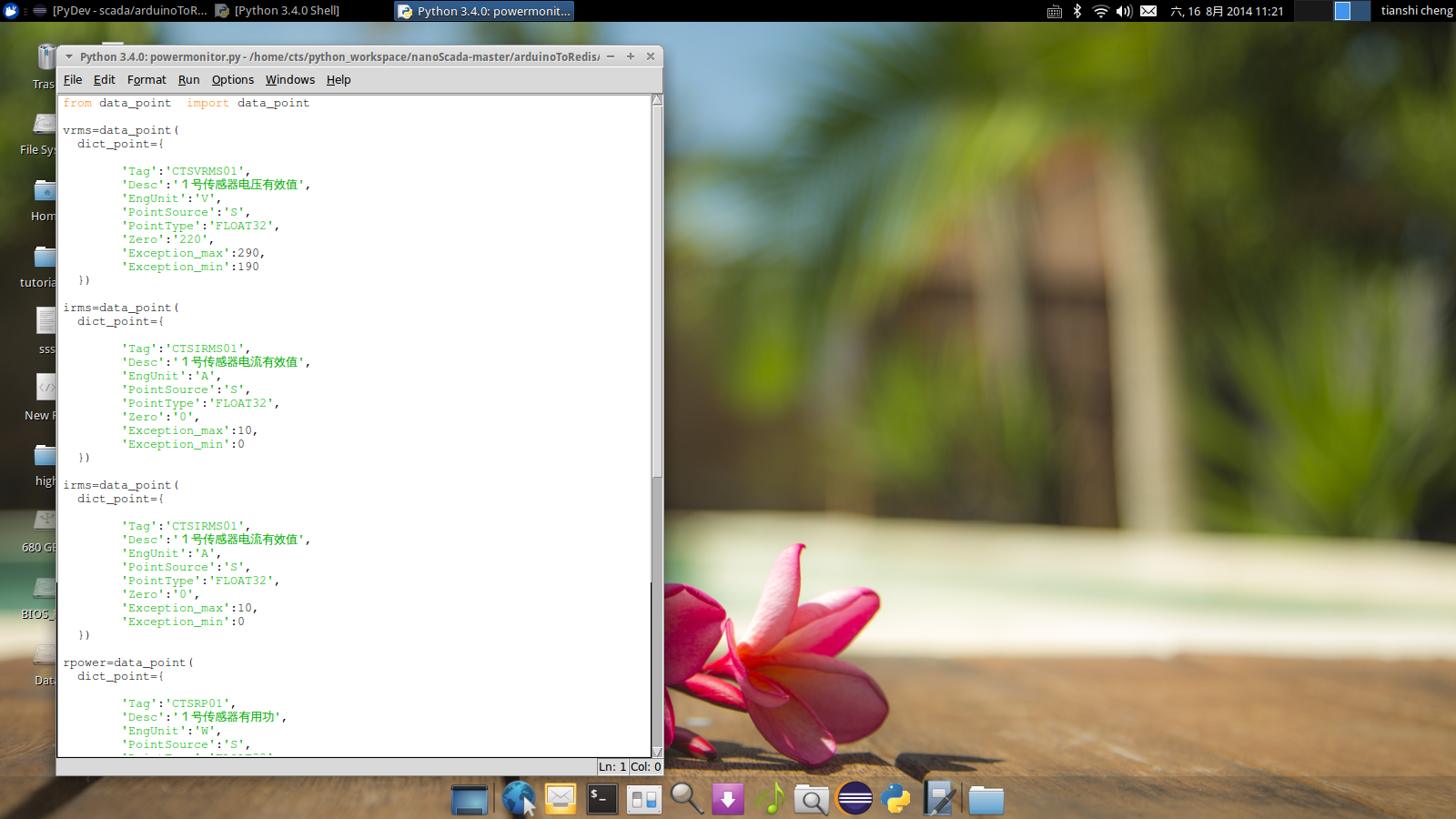


系统配置：

数据库配置：

系统运行之前必须进行相关配置工作。Redis应设定不保存本地文件。MongoDB应注意若使用IndustrialDB api，程序默认将数据存放至mydb下，并且mydb下应有至少四个集合:hourly、daily、template以及Config，分别用来存当前历史数据、每日汇总数据、存储文档模版以及测点定义信息。若不使用IndustiralDB api而采用简单文档存储，则程序默认存入mytest数据库，但仍然需要测点定义集合。打开分片功能需用mongod终端设置。(目前开发尚未完成很多东西很不完善)

测点初始化：

 在系统正式运行之前必须对测点进行初始化，原先采用类的方式配置测点信息，现在证明没有必要，将在以后改成字典方式，并通过读取excel表格来初始化测点（目前人工写python代码来组织）。下面这张图就是示例：

从图中我们看到点的基本信息有：

Tag：测点名称，应当是唯一的易辨识的名称

Desc：点的描述信息

EngUnit：单位

PointSource：点的来源类型

PointType：点数据值类型

Zero：零值

Exception\_max：值上限

Exception\_min：值下限

建成列表并调用api里的point\_init()就可以在数据库Config集合里生成Point\_definitions文档，所有点的信息将会保存在里面。这将是分析测点数据的依据,同时也是根据这个文档里的测点来预分配文档。

采集文档模版初始化：

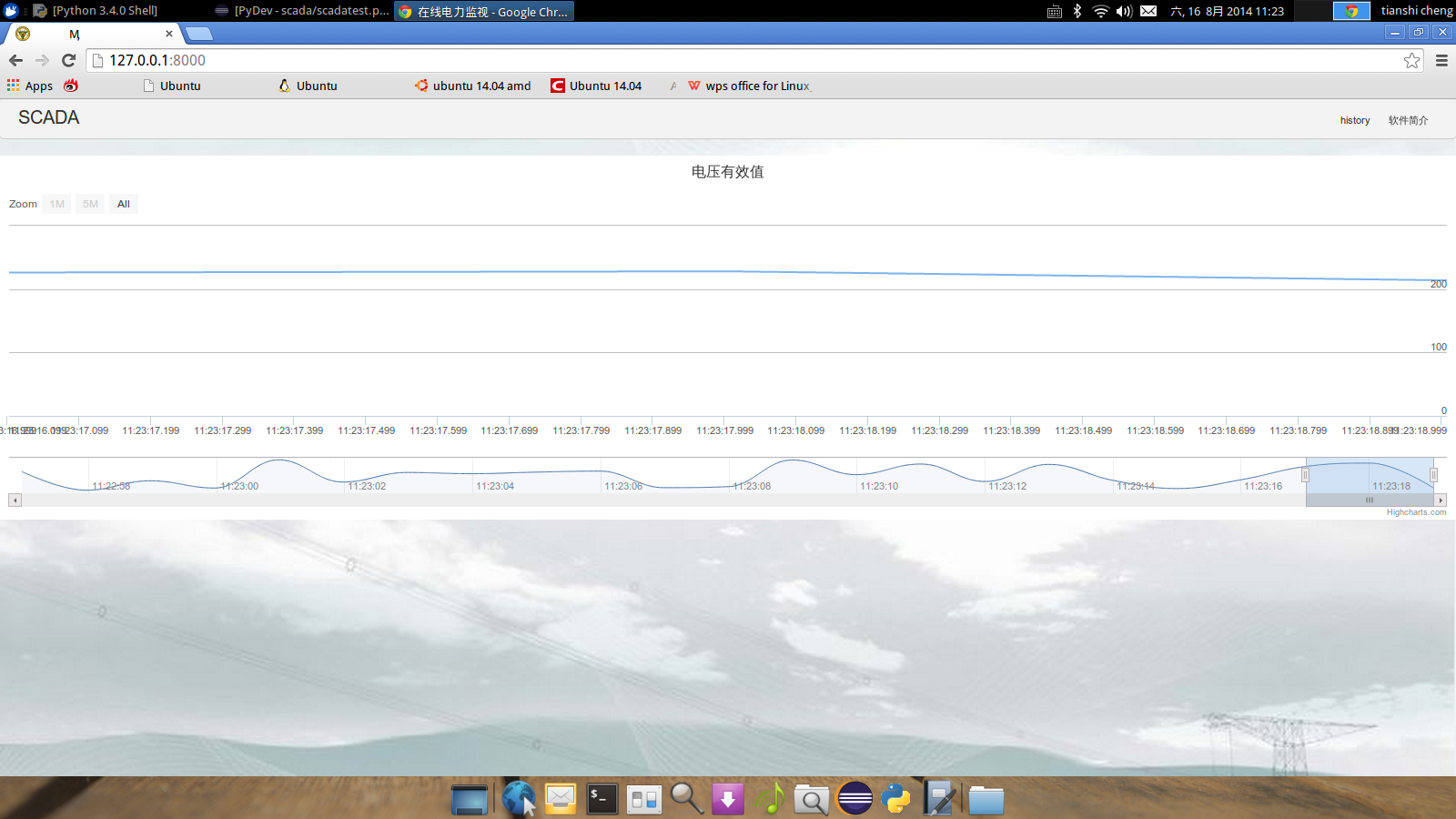
如果使用IndustrialDB api则系统会根据template集合里的文档模版分配存储数据的文档，这些文档结构可能有很多种，可以满足各种需求，目前我们只实现了一种文档模版。DataAcquistion里的template\_test就是用来生成文档的。好的文档设计可以为开发提供很多方便，并且使性能大幅提升。

数据采集程序配置：

需要配置好串口通讯和modbus协议所需要的所有参数，目前原型阶段针对不同测量设备亲自编写采集程序（因为没有标准化的接口）。数据将先送入redis缓存，当有新数据送来时再将前一数据存入历史数据库。未来将把采集程序标准化，并改成异步通讯方式。目前采集程序仅供测试。

Web服务配置：

Web服务默认侦听端口8000，若有必要请在app.py里修改端口。Web服务程序里提供了显示上一小时历史数据和实时数据的ajax模版网页，realtime.py和history.py演示了获取实时、历史数据的方式.可以通过自己编写网页和服务端程序的方式实现组态的功能。



附：系统总体框架图

