**2023 嵌入式系统设计实验讲义**

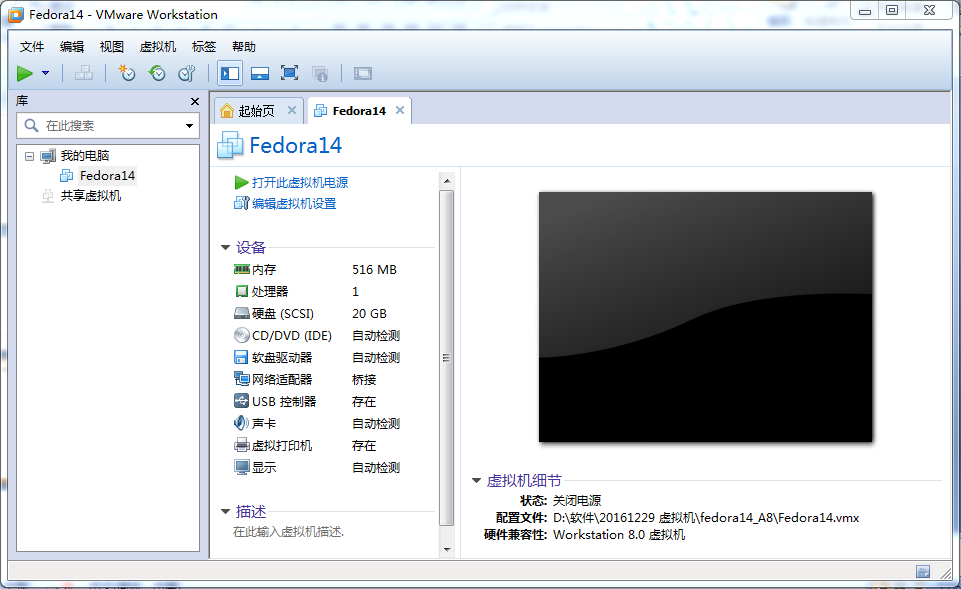
为配合《嵌入式系统设计》课程的教学，通过一系列软、硬件编程的实现，使学生能更深刻地领会嵌入式系统原理和方法，并提高程序设计能力和动手能力，特开设此实践环节。嵌入式系统设计是物联网专业核心课程，其相应实验依托硬件实验设备，相继完成如下实验

**实验一：Linux常用命令以及实验箱基本操作**

学时：2学时

**实验内容**：

1、打开虚拟机电源：用户名：root，密码：123456



2、linux基本命令

（1）ls：显示当前目录下的所有文件和目录。

ls -a：可以看到隐藏的文件，如以.开头的文件。

ls -l ：可以看到文件存取模式

（2）pwd：显示当前目录路径。

（3）ps：列举当前TTY 下所有进程 ps -A：列举所有

（4）cd 目录名：进入目录

cd - 返回进入此目录之前的目录

cd .. 回到上级目录

（5）mkdir 目录名：创建目录

（6）rmdir 目录名：删除空目录

（7）rm -rf 目录名：强行删除整个目录内容（无法恢复）；其中f 表示强制不进行提示；r 表示目录递归。

注：LINUX 下的文件和目录是区分大小写的。

（8）ifconfig : 查看ip

（9）w : Linux查看当前登录的用户运行的程序

可以查看到当前有哪些用户登录了系统以及这些用户正在运行的程序，主要查看USER和WHAT这两列的信息

（10）#代表root用户 $代表普通用户

（11）cp sourcefile destfile: 文件拷贝

（12）mv oldname newname : 重命名文件或移动文件

（13）grep 'pattern' file: 在文件内搜索字符串比如：grep 'searchstring' file.txt

（14）cat file.txt: 输出文件内容到标准输出设备（屏幕）上

（15）file somefile: 得到文件类型

（16）find: 搜索文件比如：根据文件名搜索find -name filename -print

（17）echo "some text": 将文字内容打印在屏幕上

（18）在进行shell编程时，以#开头的句子表示注释，直到这一行的结束

（19）环境变量

　　由export关键字处理过的变量叫做环境变量

（20）程序必须以下面的行开始（必须放在文件的第一行）：

　　 #!/bin/sh

　　符号#!用来告诉系统它后面的参数是用来执行该文件的程序。

在这个例子中我们使用/bin/sh来执行程序。

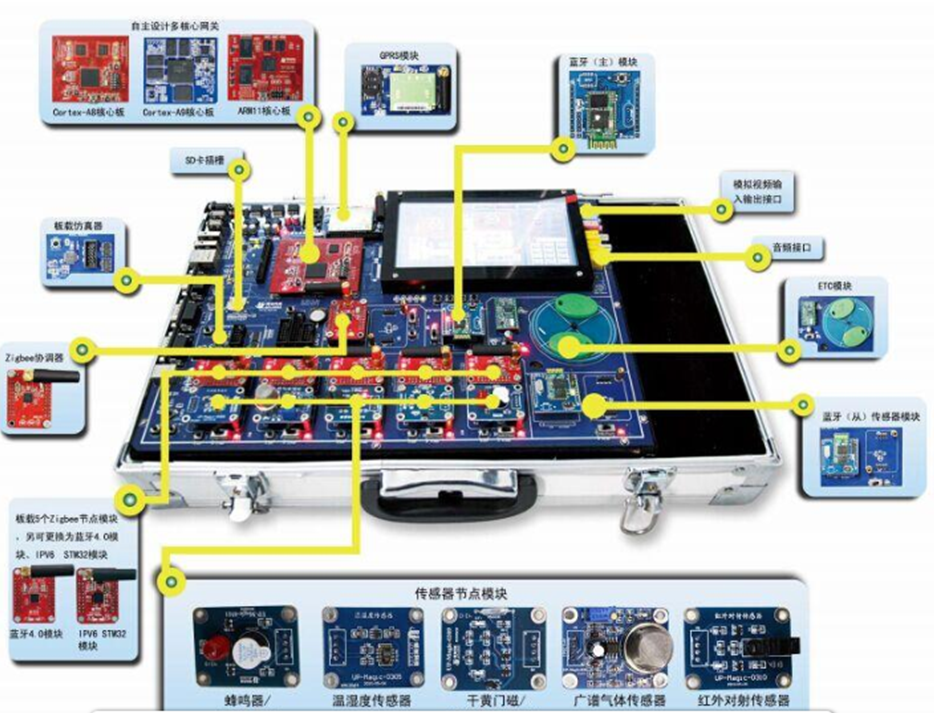
　　当编辑好脚本时，如果要执行该脚本，还必须使其可执行。

　　要使脚本可执行：

　　编译 chmod +x filename 这样才能用./filename 来运行

（21） chmod改变文件的存取模式(mode)

3、实验箱及其基本操作



**实验要求**：

1、以root身份虚拟机登录：用户名：root，密码：123456

2、熟练掌握linux基本命令

3、了解实验箱及其基本操作

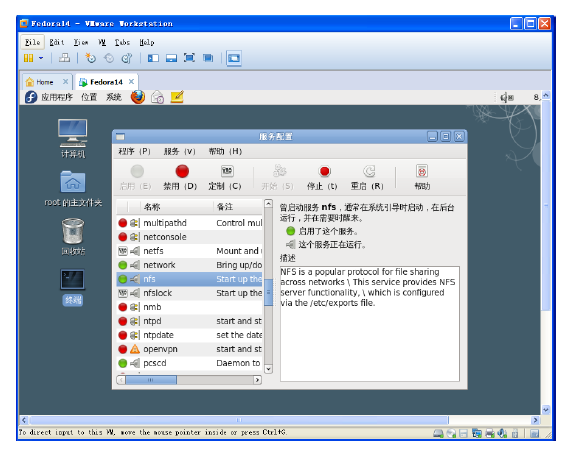
**实验二：Hello程序**

学时：2学时

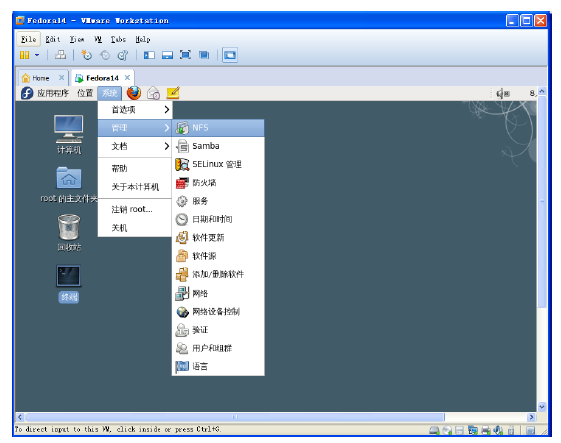
**实验内容：**

1、基于 NFS 的挂载方式**（已经设置）**

（1）建立NFS服务：启动Fedora虚拟机后，在 系统—>管理—>服务 中添加该服务。如图



（2）配置NFS共享目录 ：进入系统—>管理—>NFS 中设置目录：如图

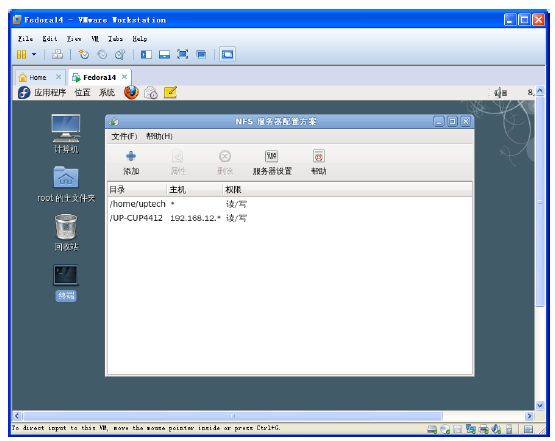


（3）目录：设置宿主机端共享目录文件夹，例如，使用产品光盘安装后产生的光盘目录：/UP-CUP4412，该NFS共享目录在后续实验都将用到，建议用户创建的NFS目录与本指导一致，方便后续实验。

主机：可以访问该共享目录的机器的IP地址，例如：192.168.12.\*

基本权限：读写权限

建立好的NFS共享目录如图



（4）挂载NFS共享目录

宿主机端NFS服务目录建立好后，即可在UP-CUP4412平台端运行mount命令挂载宿主机端NFS共享目录。启动开发板后，进入串口终端，先配置开发板IP(默认是192.168.12.199)，再挂载：

[root@UP-TECH yaffs]# ifconfig eth0 192.168.12.199

[root@UP-TECH yaffs]# mountnfs 192.168.12.133:/UP-CUP4412 /mnt/nfs/

192.168.12.199 为开发板IP地址

192.168.12.133为宿主机端IP地址。

/UP-CUP4412目录为宿主机端NFS共享目录

/mnt/nfs目录为开发板端临时挂载目录

挂载成功后即可在开发板的/mnt/nfs下访问宿主机的/UP-CUP4412目录下文件内容。

2、在宿主机端任意目录下建立工作目录 hello

[root@localhost ~]# mkdir hello

[root@localhost ~]# cd hello

3、编写程序源代码

#include <stdio.h>

main()

{

printf(“hello world \n”);

return 0;

}

编译方法1：

4、编写 Makefile

CC= arm-none-linux-gnueabi-gcc #目标机即实验箱设备编译（宿主机编译：gcc）

EXEC = hello

OBJS = hello.o

CFLAGS +=

LDFLAGS+= –static

all: $(EXEC)

$(EXEC): $(OBJS)

$(CC) -o $@ $(OBJS)

clean:

-rm -f $(EXEC) \*.elf \*.gdb \*.o

5、编译应用程序

在上面的步骤完成后，就可以在 hello 目录下运行“make”来编译程序了。

如果进行了修改，重新编译则运行:

[root@localhost hello]# make clean

[root@localhost hello]# make

6、NFS 挂载实验目录测试

7、执行程序。

执行程序用./表示执行当前目录下 hello 程序。

[root@UP-TECH hello]# ./hello ¨

实验效果

hello world

编译方法2：

4、arm-none-linux-gnueabi-gcc hello.c -o hello #目标机即实验箱设备编译（宿主机编译：gcc hello.c -o hello）

5、./hello

实验效果

hello world

**实验要求：**

1、学会使用基于 NFS 的挂载方式

2、宿主机编译程序并在宿主机运行程序

3、arm-none-linux-gnueabi-gcc 交叉编译器编译程序，并在实验箱设备上运行程序

**实验三：多线程**

学时：2学时

**实验内容**：

1、理解进程和线程

进程有自己独立的运行空间，不同进程间的信息交换和共享需要通过操作系统提供的机制来进行。

线程：进程的一条执行路径

主线程：任何进程至少要有一个线程，该线程称为主线程。

辅线程：一个进程也可以启动多个辅线程，这些线程可以执行该进程内的特殊任务。

2、线程同步

（1）当两个或多个线程要同时访问共享资源

（2）两个线程的执行有先后次序时

3、生产者-消费者（初级）

4、多线程并发执行加1操作（进阶选做）

**实验要求**：

1、了解进程与线程

2、理解线程间同步

**实验四：串口通信**

学时：2学时

**实验内容:**

1、设置串口参数

2、发送数据

3、接受数据

**实验要求：**

1、了解串行通信类型

2、了解串行通信方式

3、通过串口输入和显示字符

**实验五　数据库**

学时：2学时

**实验内容**

1、 select 语句：返回用指定的条件在一个数据库中查询的结果，返回的结果被看作记录的集合。

基本格式如下：

select [列名],[列名]...[或\*]

from [表名]

where 查询条件

group by 分组条件

order by 排序条件

2、CREATE TABLE 语句用于创建数据库中的表。

SQL CREATE TABLE 语法：

CREATE TABLE 表名称

( 列名称1 数据类型,

列名称2 数据类型,

列名称3 数据类型,

....

)

3、INSERT INTO 语句：用于向表格中插入新的行。

语法：

INSERT INTO 表名称 VALUES (值1, 值2,....)

也可以指定所要插入数据的列：

INSERT INTO table\_name (列1, 列2,...) VALUES (值1, 值2,....)

4、DELETE 语句：DELETE 语句用于删除表中的行。

语法：

DELETE FROM 表名称 WHERE 列名称 = 值

5、设计一个简单的管理系统，题目自拟，在Linux环境下实现数据库数据的增、删、改、查

**实验要求**

1、熟练掌握常用SQL语句

2、设计一个简单的管理系统

**实验六　Qt界面与控件**

学时：2学时

**实验内容**

１、按钮

（1）下压按钮QPushButton

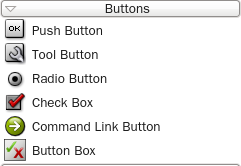
（2）工具按钮QToolButton

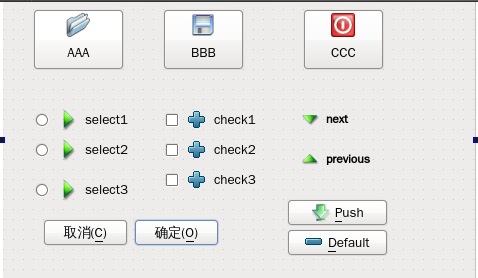
（3）选择按钮QRadioButton

（4）检查框按钮QCheckBox

（5）命令链接按钮QCommandLinkButton

（6）对话框按钮QDialogButtonBox





２、输入窗口

（1）组合框（Comb Box）

（2）字体组合框（Font Comb box）

（3）单行编辑框（Line Edit）

（4）文本编辑框（Text Edit）

（5）纯文本编辑框（Plain Text Edit）

（6）数字旋转框（Spin Box）

（7）双精度数字旋转框（Double Spin Box）

（8）时间编辑框（Time Edit）

（9）日期编辑框（Date Edit）

（10）日期时间编辑框（Date/Time Edit）

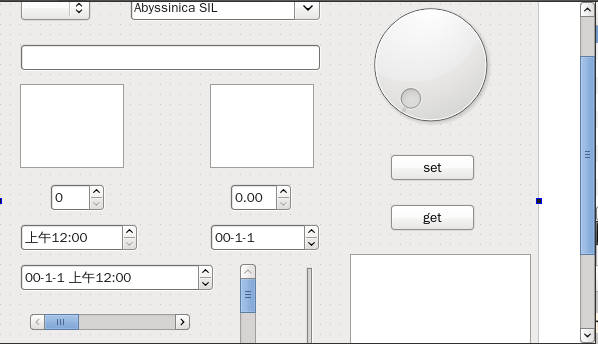
（11）数字拨盘框（Dial）

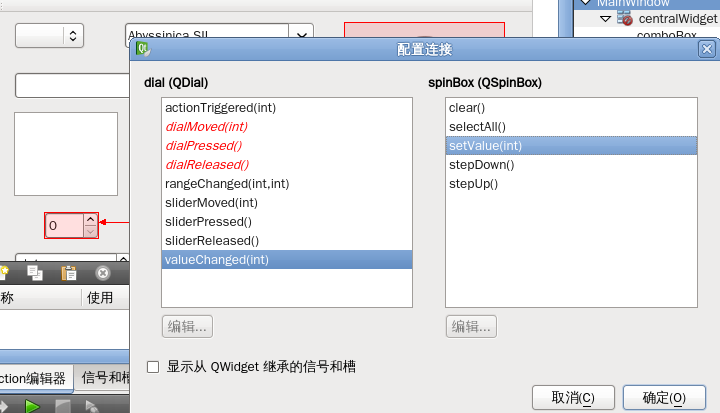
（12）水平滚动条（Horizontal Scroll Bar）

（13）垂直滚动条（Vertical Scroll Bar）

（14）水平滑动条（Horizontal Slider）

（15）垂直滑动条（Vertical Slider）





３、显示窗口部件

标签（Label）

文本浏览器（Text Browser）

日历（Calendar）

LCD数字显示（LCD Number）

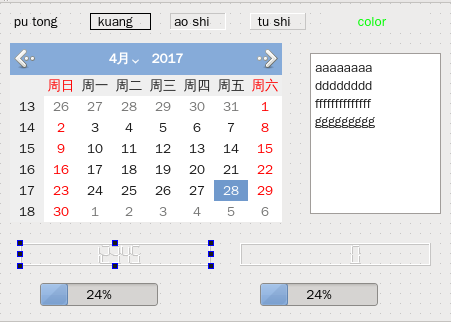
进度条（Progress Bar）

水平和垂直线

图像视图（Graphics View）

声明视图（DeclarativeView）

Web视图（QWebView）





4、实现包含菜单，工具栏，状态栏，控件，输入窗口，输出窗口的程序

5、QT界面程序在实验箱LCD屏如何显示

一般来说需要先执行一些环境语句export，然后再执行你交叉编译生成的程序。这个过程在export4arm2.sh文件中：

#!/bin/bash

cd /mnt/yaffs/up\_wsn/

insmod /mnt/yaffs/up\_wsn/8250.ko

export QTDIR=$PWD

export LD\_LIBRARY\_PATH=$PWD/lib

export TSLIB\_TSDEVICE=/dev/event1

export TSLIB\_PLUGINDIR=$PWD/lib/ts

export TSLIB\_CONSOLEDEVICE=none

export TSLIB\_CONFFILE=$PWD/etc/ts.conf

export POINTERCAL\_FILE=$PWD/etc/ts-calib.conf

export QWS\_MOUSE\_PROTO=tslib:/dev/event1

export TSLIB\_CALIBFILE=$PWD/etc/ts-calib.conf

export QT\_QWS\_FONTDIR=$PWD/lib/fonts

export QT\_PLUGIN\_PATH=$PWD/plugins/

#export QWS\_DISPLAY="LinuxFb:mmWidth260:mmHeight245:0"

export LANG=zh\_CN

# for tslib

if [ ! -f /mnt/yaffs/up\_wsn/etc/ts-calib.conf ];then

/mnt/yaffs/up\_wsn/bin/ts\_calibrate

fi

ifconfig eth0 192.168.12.199

/usr/sbin/telnetd&

/mnt/yaffs/up\_wsn/ev\_server 7838 9&

usleep 10000

/mnt/yaffs/up\_wsn/wsn -qws -font wenquanyi

export4arm2.sh文件最后一行就是要执行的程序，/mnt/yaffs/up\_wsn/wsn就是每次开实验箱时在LCD屏幕上显示的示例程序。/mnt/yaffs/up\_wsn/ev\_server 7838 9&就是导致滚屏的程序。

（1）为防止误操作，首先把export4arm2.sh文件拷贝到挂载目录/mnt/nfs，也就是最终拷到你的虚拟机和/mnt/nfs对应的那个目录（一般为/home/uptech）

cd /mnt/yaffs/up\_wsn

cp export4arm2.sh /mnt/nfs

（2）进入新的export4arm2.sh，把最后一行换成你的程序

cd /mnt/nfs

vi export4arm2.sh

即新的export4arm2.sh内容为：（注意：除了红色字，其他内容不要修改；注释掉滚屏程序，使得运行你程序时不会滚屏。）

#!/bin/bash

cd /mnt/yaffs/up\_wsn/

insmod /mnt/yaffs/up\_wsn/8250.ko

export QTDIR=$PWD

export LD\_LIBRARY\_PATH=$PWD/lib

export TSLIB\_TSDEVICE=/dev/event1

export TSLIB\_PLUGINDIR=$PWD/lib/ts

export TSLIB\_CONSOLEDEVICE=none

export TSLIB\_CONFFILE=$PWD/etc/ts.conf

export POINTERCAL\_FILE=$PWD/etc/ts-calib.conf

export QWS\_MOUSE\_PROTO=tslib:/dev/event1

export TSLIB\_CALIBFILE=$PWD/etc/ts-calib.conf

export QT\_QWS\_FONTDIR=$PWD/lib/fonts

export QT\_PLUGIN\_PATH=$PWD/plugins/

#export QWS\_DISPLAY="LinuxFb:mmWidth260:mmHeight245:0"

export LANG=zh\_CN

# for tslib

if [ ! -f /mnt/yaffs/up\_wsn/etc/ts-calib.conf ];then

/mnt/yaffs/up\_wsn/bin/ts\_calibrate

fi

ifconfig eth0 192.168.12.199

#/usr/sbin/telnetd&

#/mnt/yaffs/up\_wsn/ev\_server 7838 9&

usleep 10000

/mnt/nfs/myQT\_X11/myQT\_X11 -qws -font wenquanyi

（3）在虚拟机修改export4arm2.sh后，在XShell中运行程序

./export4arm2.sh

**实验要求：**

1、掌握菜单、工具栏、状态栏以及各种控件使用方法

2、掌握输入窗口的用法

3、掌握显示窗口的用法

4、在宿主机运行程序

5、在实验箱设备上运行程序

**实验七 Qt自定义对话框**

学时：2学时

**实验内容**：

1、第一个界面的编辑框输入一个字符串a，按“确定”按钮后弹出第二个对话框

2、在第二个对话框的编辑框中显示字符串a，然后修改字符串a为字符串b，按“关闭”后返回第一个对话框

3、在第一个对话框的编辑框中显示第二个对话框中输入的字符串b。

**实验要求：**

1、掌握自定义对话框

2、在宿主机运行程序

3、在实验箱设备上运行程序

**实验八 Qt程序获取传感器数据**

学时：4学时

**实验内容**：

１、获取传感器数据，如温湿度或者红外对射数据等，可以参考实验指导用例通过网络传输得到相关数据。

2、将获得的传感器数据显示到某个QT显示控件中

3、控制灯的亮灭

例如：获取温度传感器中的温度数据

void Widget::show\_TempHum(unsigned long x){

char temp[5],humi[5],data[4];

float num = 0, humnu = 0;

char \*pTemp,\*pHumi;

pTemp = temp;

pHumi = humi;

int return\_t=0;

data[0]=BREAK\_UINT32( x,3 );

data[1]=BREAK\_UINT32( x,2 );

data[2]=BREAK\_UINT32( x,1 );

data[3]=BREAK\_UINT32( x,0 );

return\_t=getsht11(data, pTemp, pHumi);

QDateTime time = QDateTime::currentDateTime();

QDate dataa = QDate::currentDate();

dataa.setYMD(2017, 6, 27);

time.setDate(dataa);

ui->dateTimeEdit->setDateTime(time);

if(return\_t){

QPixmap img1 (":/images/node\_warning.png");

ui->staus->setPixmap(img1);

QPixmap img (":/images/node\_smog.png");

ui->smog->setPixmap(img);

ui->label\_temp\_value->setStyleSheet("color:rgb(35,235,185)");

ui->label\_hum\_value->setStyleSheet("color:rgb(35,235,185)");

ui->label\_temp\_value->setText(pTemp);

ui->label\_hum\_value->setText(pHumi);

num = atof(pTemp);

humnu = atof(pHumi);

QTime time;

time =QTime::currentTime();

qsrand(time.msec() + time.second()\*1000);

int srand = qrand()%20;

int tosave = (int)num + srand;

list<<tosave;

if(num > temmax)

{

showlight();

ui->staus->setText(QString::fromUtf8("temperature WARING"));

QPixmap img2 (":/images/node\_warning\_selected.png");

ui->staus->setPixmap(img2);

}

if(humnu\*2.5 > hummax)

{

QPixmap img (":/images/node\_smog\_selected.png");

ui->smog->setPixmap(img);

showlight();

ui->staus->setText(QString::fromUtf8("humidity WARING"));

QPixmap img3 (":/images/node\_warning\_selected.png");

ui->staus->setPixmap(img3);

//ui->smog->setText(QString::fromUtf8("WARING!!!"));

}

ui->label\_temp\_value->update();

ui->label\_hum\_value->update();

}

}

int Widget::calc\_sth11(float \*p\_humidity ,float \*p\_temperature)

{

int ret= 0;

const float C1=-4.0; // for 12 Bit

const float C2= 0.0405; // for 12 Bit

const float C3=-0.0000028; // for 12 Bit

const float T1=0.01; // for 14 Bit @ 5V

const float T2=0.00008; // for 14 Bit @ 5V

float rh=\*p\_humidity; // rh: Humidity [Ticks] 12 Bit

float t=\*p\_temperature; // t: Temperature [Ticks] 14 Bit

float rh\_lin; // rh\_lin: Humidity linear

float rh\_true; // rh\_true: Temperature compensated humidity

float t\_C; // t\_C : Temperature [­C]

t\_C=t\*0.01 - 42; //calc. Temperature from ticks to [­C]

rh\_lin=C3\*rh\*rh + C2\*rh + C1; //calc. Humidity from ticks to [%RH]

rh\_true=(t\_C-25)\*(T1+T2\*rh)+rh\_lin; //calc. Temperature compensated humidity [%RH]

if(rh\_true>100)rh\_true=100; //cut if the value is outside of

if(rh\_true<0.1)rh\_true=0.1; //the physical possible range

\*p\_temperature=t\_C; //return temperature [­C]

\*p\_humidity=rh\_true; //ret ss = read();

if (t\_C>0)return ret = 1;

return ret;

}

int Widget:: getsht11(char \*rData, char \*&pTemp, char \*&pHumi)

{

int i=0,j=0;

float temp,humi;

uint8 tlen,hlen;

int calcret=0;

char Temp\_t[16];

i = rData[1] \* 256 + rData[0];

j = rData[3] \* 256 + rData[2];

temp = (float)j;

humi = (float)i;

calcret = calc\_sth11(&humi,&temp);

//qDebug("\rtemp=%f\t humi=%f\r",temp,humi);

if( ((int)temp) > 0 && 0 <= humi <= 100 && calcret)

{

tlen = sprintf(pTemp, "%d.%d",(int)temp,(int)((temp-(int)temp)\*10));

hlen = sprintf(pHumi, "%d.%d",(int)humi,(int)((humi-(int)humi)\*10));

//qDebug("\rTEMP=%s\tHUMI=%s\r",pTemp,pHumi);

return 1;

}

return 0;

**实验要求：**

1．获取传感器数据

2．将获取的传感器数据显示到可视化界面

3、单击按钮控制灯亮灭