#### 电机正反转状态显示功能部分实现操作代码：

protected void onCreate(Bundle savedInstanceState) {

// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

setContentView(R.layout.motordriver);

pref = SettingActivity.spf;

// 得到个控件

tv1 = (TextView) findViewById(R.id.lampStatus);// 电机的状态

lampDate = (TextView) findViewById(R.id.lampDate);// 操作时间

buttonmotorzheng = (Button) findViewById(R.id.motor1);// 电机正转按钮

buttonmotorfan = (Button) findViewById(R.id.motor2);// 电机反转按钮

buttonmotorstop = (Button) findViewById(R.id.motor0);//停止按钮

handler = new Handler() {

@Override

public void handleMessage(Message msg) {

// TODO Auto-generated method stub

if (msg.obj.equals("true") && msg.arg1 ==2) {

tv1.setText("反转");

buttonmotorzheng.setClickable(true);// 电机正转按钮设置为：不可点击

buttonmotorzheng.setVisibility(View.VISIBLE);// 正转按钮显示：可见

buttonmotorstop.setClickable(true);// 停止按钮设置为：不可点击

buttonmotorstop.setVisibility(View.VISIBLE);// 停止按钮显示：可见

buttonmotorfan.setClickable(false);// 反转按钮设置为：可点击操作

lampDate.setText(getTime());

Toast.makeText(getApplicationContext(), "成功",

Toast.LENGTH\_SHORT).show();

} else if (msg.obj.equals("true") && msg.arg1 ==1) {

tv1.setText("正转");

buttonmotorfan.setClickable(true);// 反转按钮设置为：不可点击

buttonmotorfan.setVisibility(View.VISIBLE);// 反转按钮显示：可见

buttonmotorzheng.setClickable(false);// 正转按钮设置为：可点击操作

buttonmotorstop.setClickable(true);// 停止按钮设置为：不可点击

buttonmotorstop.setVisibility(View.VISIBLE);// 停止按钮显示：可见

lampDate.setText(getTime());

}else if (msg.obj.equals("true") && msg.arg1 ==3) {

tv1.setText("停止");

buttonmotorfan.setClickable(true);// 反转按钮设置为：不可点击

buttonmotorfan.setVisibility(View.VISIBLE);// 反转按钮显示：可见

buttonmotorzheng.setVisibility(View.VISIBLE);// 正转按钮显示：可见

buttonmotorzheng.setClickable(true);// 正转按钮设置为：可点击操作

buttonmotorstop.setClickable(false);

lampDate.setText(getTime());

}

else {

Toast.makeText(getApplicationContext(), "失败",

Toast.LENGTH\_SHORT).show();

}

super.handleMessage(msg);

}

};

}

#### 风扇功能实现代码：以风扇正转为例

public void clickmotorfan(View v) {

try {

if (pref.getString("IP", "") == null

|| pref.getString("IP", "").equals("")) {

Toast.makeText(getApplicationContext(), "请先配置",

Toast.LENGTH\_SHORT).show();

return;

}

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

String deviceId = "69";

String operationContent = "2,7";

String reString = getOrSetMotorDriver(operationContent, deviceId);

Message message = handler.obtainMessage();

message.obj = reString;

message.arg1 =2;

message.sendToTarget();

}

});

thread.start();

} catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}

}

public void clickmotorzheng(View v) {

try {

if (pref.getString("IP", "") == null

|| pref.getString("IP", "").equals("")) {

Toast.makeText(getApplicationContext(), "请先配置",

Toast.LENGTH\_SHORT).show();

return;

}

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

String deviceId = "69";

String operationContent = "1,7";

String reString = getOrSetMotorDriver(operationContent, deviceId);

Message message = handler.obtainMessage();

message.obj = reString;

message.arg1 =1;

message.sendToTarget();

}

});

thread.start();

} catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}

}

### 温度采集功能部分实现代码：

##### （以温度为例：瓦斯，二氧化碳，湿度传感器的实现代码基本相同，不再赘述）

public class TemperatureActivity extends Activity {

// 变量声明

TextView textViewTemp;// 显示温度

TextView textViewTime;// 显示采集时间

EditText editTextDeviceId;// 唯一识别号

Button buttonRead, buttonTempreturn;// 温度采集按钮

SharedPreferences pref; // 声明SharedPreferences

private Intent intent;

Return retu = new Return();

private ShareDataActivity shareData;

// private String deviceId="01";

private String methodname = "getTemperature";

private Handler handler;

private Button TemButton;

// 声明网络地址

public static String serviceUrl = "";

@SuppressLint("NewApi")

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.temperature);

// ShareDataActivity.getVersion();

// 得到各控件对象

final ManageSQL manageSQL=new ManageSQL();

textViewTemp = (TextView) findViewById(R.id.TemEdit);

textViewTime = (TextView) findViewById(R.id.temDateEdit);

buttonRead = (Button) findViewById(R.id.TemReading);

//buttonTempreturn = (Button) findViewById(R.id.temReturn);

TemButton = (Button) findViewById(R.id.temManage);

// editTextDeviceId = (EditText) findViewById(R.id.TemDeviceId);

//

shareData = new ShareDataActivity();

handler = new Handler() {

@SuppressLint("HandlerLeak")

@Override

public void handleMessage(Message msg) {

// TODO Auto-generated method stu

// Bundle bundle = msg.getData();

// String data = bundle.getString("data");

retu = (Return) msg.obj;

if (retu == null) {

Toast.makeText(TemperatureActivity.this, "采集信息失败",

Toast.LENGTH\_SHORT).show();

} else {

textViewTemp.setText(retu.getValue() + "°C");

MySQLite mySQLite = MainActivity.mySQLite;

manageSQL.insertT(mySQLite.getReadableDatabase(), retu.getValue() + "°C(温度)", shareData.getTime());

textViewTime.setText(shareData.getTime());

}

super.handleMessage(msg);

}

};

TemButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

intent = new Intent(TemperatureActivity.this,

TemperatureActivityManage.class);

startActivity(intent);

}

});

// 为温度采集按钮添加监听器

buttonRead.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View arg0) {

// TODO Auto-generated method stub

final String deviceId = "01";

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

retu = shareData.GetData(deviceId, methodname);

if (retu == null) {

Message retuMessage = handler.obtainMessage();// 得到消息队列

Bundle bundle = new Bundle();

bundle.putString("data", "ERROR");// 把他想想成map

retuMessage.setData(bundle);// 把数据放进去

retuMessage.sendToTarget();// 发送到handerlaer

return;

} else {

String dateString = retu.getValue();

Message retuMessage = handler.obtainMessage();// 得到消息队列

retuMessage.obj = retu;

Bundle bundle = new Bundle();// 可以存储比较大类型 的数据

bundle.putString("data", dateString);// 把他想想成map

retuMessage.setData(bundle);// 把数据放进去

retuMessage.sendToTarget();// 发送到handerlaer

Log.e("hander", "线程开始了" + "retuMessage:"

+ retuMessage + "retu：" + dateString);

}

}

});

thread.start( );

}

});

##### 温度数据库实现部分：（数据库用SQLIte实现的）

public class TemperatureActivityManage extends Activity {

ListView listView;

MySQLite mySQLite=MainActivity.mySQLite;

String data, time;

Cursor cursor;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_mainkk);

showData();

listView.setOnItemLongClickListener(new OnItemLongClickListener() {

@Override

public boolean onItemLongClick(AdapterView<?> arg0, View arg1,

int arg2, long arg3) {

LinearLayout layout = (LinearLayout) arg1;

final TextView id = (TextView) layout.getChildAt(0);

new AlertDialog.Builder(TemperatureActivityManage.this)

.setMessage("确定删除么？")

.setPositiveButton("确定",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,

int which) {

String strRowValue = id.getText()

.toString();

new ManageSQL().DeleteARecord(

mySQLite.getReadableDatabase(),

"id", strRowValue);

showData();

}

})

.setNegativeButton("取消",

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,

int which) {

// TODO Auto-generated method stub

}

}).show();

return false;

}

});

}

public void showData() {

List<Map<String, String>> listItemsList = new ArrayList<Map<String, String>>();

listView = (ListView) findViewById(R.id.listview);

cursor = mySQLite.getReadableDatabase().rawQuery(

"select \* from temperatrue", null);

SimpleAdapter adapter = new SimpleAdapter(this, listItemsList,

R.layout.activity\_listview, new String[] { "id", "time0",

"data0" }, new int[] { R.id.id, R.id.timeview,

R.id.dataview });

while (cursor.moveToNext()) {

Map<String, String> map = new HashMap<String, String>();

map.put("id", cursor.getString(0));

map.put("time0", cursor.getString(1));// 数据库中第一列的内容显示在listview的左边

map.put("data0", cursor.getString(2));// 数据库的第三列内容显示在listview的右边

listItemsList.add(map);

}

listView.setAdapter(adapter);

}

}

### 安全矿灯（电灯）部分功能实现代码：

@SuppressLint("HandlerLeak")

public class LampActivity extends Activity {

// 声明各控件

private TextView tv1, lampDate;

private ImageView imageView;

EditText editTextDeviceId;

private Button buttonopenLamp, buttoncloseLamp, buttonlampreturn;

SharedPreferences pref; // 声明SharedPreferences

// 跳转

private Intent intent;

// 声明网络地址

public static String serviceUrl = "";

private Handler handler;

@SuppressLint("NewApi")

protected void onCreate(Bundle savedInstanceState) {

// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

setContentView(R.layout.lamp);

pref = SettingActivity.spf;

// 得到个控件

tv1 = (TextView) findViewById(R.id.lampStatus);// 灯的状态

lampDate = (TextView) findViewById(R.id.lampDate);// 操作时间

imageView = (ImageView) findViewById(R.id.lampPictrue);// 灯的显示图标片

buttonopenLamp = (Button) findViewById(R.id.openLamp);// 开灯按钮

buttoncloseLamp = (Button) findViewById(R.id.closeLamp);// 关灯操作

buttonlampreturn = (Button) findViewById(R.id.lampreturn);// 返回按钮

handler = new Handler() {

@Override

public void handleMessage(Message msg) {

// TODO Auto-generated method stub

if (msg.obj.equals("true") && msg.arg1 == 1) {

tv1.setText("开");

imageView.setImageResource(R.drawable.openedlamp);

buttonopenLamp.setClickable(false);// 开灯按钮设置为：不可点击

buttoncloseLamp.setVisibility(View.VISIBLE);// 关灯按钮显示：可见

buttoncloseLamp.setClickable(true);// 关灯按钮设置为：可点击操作

lampDate.setText(getTime());

Toast.makeText(getApplicationContext(), "成功",

Toast.LENGTH\_SHORT).show();

} else if (msg.obj.equals("true") && msg.arg1 == 0) {

tv1.setText("关");

imageView.setImageResource(R.drawable.closedlamp);

buttoncloseLamp.setClickable(false);// 关灯按钮设置为：不可点击

buttonopenLamp.setVisibility(View.VISIBLE);// 开灯按钮显示：可见

buttonopenLamp.setClickable(true);// 开灯按钮设置为：可点击操作

lampDate.setText(getTime());

}

else {

Toast.makeText(getApplicationContext(), "失败",

Toast.LENGTH\_SHORT).show();

}

super.handleMessage(msg);

}

};

// 为返回按钮添加

/\*buttonlampreturn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

// 点击返回按钮则返回到主界面

intent = new Intent(LampActivity.this, MainActivity.class);

startActivity(intent);

}

});\*/

}

/\*\*

\* 开灯事件

\*

\* @param v

\*/

public void clickOpenLamp(View v) {

try {

if (pref.getString("IP", "") == null

|| pref.getString("IP", "").equals("")) {

Toast.makeText(getApplicationContext(), "请先配置",

Toast.LENGTH\_SHORT).show();

return;

}

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

String deviceId = "65";

String operationContent = "1";

String reString = SetSwitch(operationContent, deviceId);

Message message = handler.obtainMessage();

message.obj = reString;

message.arg1 = 1;

message.sendToTarget();

}

});

thread.start();

} catch (Exception e) {

// TODO: handle exception

}

}

/\*\*

\* 关灯事件

\*

\* @param v

\*/

public void clickCloseLamp(View v) {

try {

if (pref.getString("IP", "") == null

|| pref.getString("IP", "").equals("")) {

Toast.makeText(getApplicationContext(), "请先配置",

Toast.LENGTH\_SHORT).show();

return;

}

Thread thread = new Thread(new Runnable() {

@Override

public void run() {

String deviceId = "65";

String operationContent = "2";

String reString = SetSwitch(operationContent, deviceId);

Message message = handler.obtainMessage();

message.obj = reString;

message.sendToTarget();

}

});

thread.start();

} catch (Exception e) {

// TODO: handle exception

}

}

@SuppressLint("SimpleDateFormat")

public String getTime() {

SimpleDateFormat simpleDateFormat = new SimpleDateFormat(

"yyyy/MM/dd hh:mm:ss");

String dd = simpleDateFormat.format(new Date());

return dd;

}

// 开关灯设置操作 ​

public String SetSwitch(String str, String devId) {

// 声明变量

Return retu = null;

ArrayList<Return> retuArr = null;

// 声明时间

// 参数样式

String ip = pref.getString("IP", "");

String port = pref.getString("Port", "");

String deviceId = devId;

String connectType = "1";

String connectParameter = "";

String operationType = "2";

String operationContent = str;

String paraValues = "<sensor>" + "<deviceId>" + deviceId

+ "</deviceId>" + "<connectType>" + connectType

+ "</connectType>" + "<ip>" + ip + "</ip>" + "<port>" + port

+ "</port>" + "<connectParameter>" + connectParameter

+ "</connectParameter>" + "<operationType>" + operationType

+ "</operationType>" + "<operationContent>" + operationContent

+ "</operationContent>" + "</sensor>";

try {

serviceUrl = pref.getString("Url", "");

// 命名空间

String nameSpace = "http://service.sensor.microsec.com";

// 定义调用的WebService方法名

String methodName = "getOrSetSwitch";

// 第1步：创建SoapObject对象，并指定WebService的命名空间和调用的方法名

SoapObject request = new SoapObject(nameSpace, methodName);

// 第2步：设置WebService方法的参数

request.addProperty("sensor", paraValues);

// 第3步：创建SoapSerializationEnvelope对象，并指定WebService的版本

SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(

SoapEnvelope.VER11);

// 设置bodyOut属性

envelope.bodyOut = request;

// 第4步：创建HttpTransportSE对象，并指定WSDL文档的URL

HttpTransportSE ht = new HttpTransportSE(serviceUrl);

// 第5步：调用WebService

ht.call(null, envelope);

if (envelope.getResponse() != null) {

// 第6步：使用getResponse方法获得WebService方法的返回结果

Object object = (Object) envelope.getResponse();

// 得到返回结果 ，Object类型转换成字符串

String result = object.toString();

Log.i("测试", "连接成功后返回的值：" + result.toString());

ByteArrayInputStream is = new ByteArrayInputStream(

result.getBytes("ISO-8859-1"));

// 声明变量解析器

XmlPullParser parser = Xml.newPullParser();

parser.setInput(is, "UTF-8");

Log.i("测试", "连接成功后返回的值============：" + parser.toString());

int event = parser.getEventType();

while (event != XmlPullParser.END\_DOCUMENT) {

switch (event) {

case XmlPullParser.START\_DOCUMENT:// 判断是否文档开始事件

retuArr = new ArrayList<Return>();

break;

case XmlPullParser.START\_TAG:// 判断是否标签元素开始事件

if ("return".equalsIgnoreCase(parser.getName())) {

retu = new Return();

} else if (retu != null) {

if ("success".equalsIgnoreCase(parser.getName())) {

retu.setSuccess(parser.nextText());

}

}

break;

case XmlPullParser.END\_TAG:// 判断当前事件是否是标签元素结束事件

if ("return".equalsIgnoreCase(parser.getName())) {

retuArr.add(retu);

}

break;

}

event = parser.next();// 进入下一个元素并触发相应事件

}

if (is != null) {

is.close();

}

}

} catch (Exception e) {

e.printStackTrace();

Log.e("测试", "连接URL产生异常：" + e.toString());

} finally {

}

Log.i("测试", "连接失败：" + retu.getSuccess().toString());

String returnString = retu.getSuccess().toString().trim();

return returnString;

}

}

### 登录界面功能实现代码：

public class Login extends Activity {

EditText txtlogin;// 创建EditText对象

Button btnlogin, btnclose;// 创建两个Button对象

@Override

protected void onCreate(Bundle savedInstanceState) {

// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

setContentView(R.layout.login);// 设置布局文件

txtlogin = (EditText) findViewById(R.id.txtLogin);// 获取密码文本框

btnlogin = (Button) findViewById(R.id.btnLogin);// 获取登录按钮

btnclose = (Button) findViewById(R.id.btnClose);// 获取取消按钮

btnlogin.setOnClickListener(new OnClickListener() {// 为登录按钮设置监听事件

@Override

public void onClick(View arg0) {

// TODO Auto-generated method stub

Intent intent = new Intent(Login.this, MainActivity.class);// 创建Intent对象

PwdDAO pwdDAO = new PwdDAO(Login.this);// 创建PwdDAO对象

// 判断是否有密码及是否输入了密码

if ((pwdDAO.getCount() == 0 || pwdDAO.find().getPassword()

.isEmpty())

&& txtlogin.getText().toString().isEmpty()) {

startActivity(intent);// 启动主Activity

} else {

// 判断输入的密码是否与数据库中的密码一致

if (pwdDAO.find().getPassword()

.equals(txtlogin.getText().toString())) {

startActivity(intent);// 启动主Activity

} else {

// 弹出信息提示

Toast.makeText(Login.this, "请输入正确的密码！",

Toast.LENGTH\_SHORT).show();

}

}

txtlogin.setText("");// 清空密码文本框

}

});

btnclose.setOnClickListener(new OnClickListener() {// 为取消按钮设置监听事件

@Override

public void onClick(View arg0) {

// TODO Auto-generated method stub

finish();// 退出当前程序

}

});

}

}

### 主界面功能实现部分代码：

private Intent intent;

GridView gvInfo;// 创建GridView对象

// 定义字符串数组，存储系统功能

String[] titles = new String[] { "二氧化碳", "湿度", "电灯", "风扇",

"系统设置", "温度", "瓦斯浓度","退出" };

// 定义int数组，存储功能对应的图标

int[] images = new int[] { R.drawable.ad,

R.drawable.ah, R.drawable.as,

R.drawable.af, R.drawable.ww,

R.drawable.ak, R.drawable.ax, R.drawable.a };

gvInfo = (GridView) findViewById(R.id.gvInfo);// 获取布局文件中的gvInfo组件

pictureAdapter adapter = new pictureAdapter(titles, images, this);// 创建pictureAdapter对象

gvInfo.setAdapter(adapter);// 为GridView设置数据源

gvInfo.setOnItemClickListener(new OnItemClickListener() {// 为GridView设置项单击事件

@Override

public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,

long arg3) {

Intent intent = null;// 创建Intent对象

switch (arg2) {

case 0:

intent = new Intent(MainActivity.this, CoActivity.class);// 使用AddOutaccount窗口初始化Intent

startActivity(intent);// 打开AddOutaccount

break;

case 1:

intent = new Intent(MainActivity.this, HumidityActivity.class);// 使用AddInaccount窗口初始化Intent

startActivity(intent);// 打开AddInaccount

break;

case 2:

intent = new Intent(MainActivity.this, LampActivity.class);// 使用Outaccountinfo窗口初始化Intent

startActivity(intent);// 打开Outaccountinfo

break;

case 3:

intent = new Intent(MainActivity.this,MotordriverActivity.class);// 使用Showinfo窗口初始化Intent

startActivity(intent);// 打开Showinfo

break;

case 4:

intent = new Intent(MainActivity.this, SettingActivity.class);// 使用Sysset窗口初始化Intent

startActivity(intent);// 打开Sysset

break;

case 5:

intent = new Intent(MainActivity.this, TemperatureActivity.class);// 使用Accountflag窗口初始化Intent

startActivity(intent);// 打开Accountflag

break;

case 6:

intent = new Intent(MainActivity.this, SmokeActivity.class);// 使用Accountflag窗口初始化Intent

startActivity(intent);// 打开Accountflag

break;

case 7:

finish();// 关闭当前Activity

}

}

});

}

}

class pictureAdapter extends BaseAdapter// 创建基于BaseAdapter的子类

{

private LayoutInflater inflater;// 创建LayoutInflater对象

private List<Picture> pictures;// 创建List泛型集合

// 为类创建构造函数

public pictureAdapter(String[] titles, int[] images, Context context) {

super();

pictures = new ArrayList<Picture>();// 初始化泛型集合对象

inflater = LayoutInflater.from(context);// 初始化LayoutInflater对象

for (int i = 0; i < images.length; i++)// 遍历图像数组

{

Picture picture = new Picture(titles[i], images[i]);// 使用标题和图像生成Picture对象

pictures.add(picture);// 将Picture对象添加到泛型集合中

}

}

@Override

public int getCount() {// 获取泛型集合的长度

if (null != pictures) {// 如果泛型集合不为空

return pictures.size();// 返回泛型长度

} else {

return 0;// 返回0

}

}

@Override

public Object getItem(int arg0) {

return pictures.get(arg0);// 获取泛型集合指定索引处的项

}

@Override

public long getItemId(int arg0) {

return arg0;// 返回泛型集合的索引

}

@Override

public View getView(int arg0, View arg1, ViewGroup arg2) {

ViewHolder viewHolder;// 创建ViewHolder对象

if (arg1 == null)// 判断图像标识是否为空

{

arg1 = inflater.inflate(R.layout.gvitem, null);// 设置图像标识

viewHolder = new ViewHolder();// 初始化ViewHolder对象

viewHolder.title = (TextView) arg1.findViewById(R.id.ItemTitle);// 设置图像标题

viewHolder.image = (ImageView) arg1.findViewById(R.id.ItemImage);// 设置图像的二进制值

arg1.setTag(viewHolder);// 设置提示

} else {

viewHolder = (ViewHolder) arg1.getTag();// 设置提示

}

viewHolder.title.setText(pictures.get(arg0).getTitle());// 设置图像标题

viewHolder.image.setImageResource(pictures.get(arg0).getImageId());// 设置图像的二进制值

return arg1;// 返回图像标识

}

}

class ViewHolder// 创建ViewHolder类

{

public TextView title;// 创建TextView对象

public ImageView image;// 创建ImageView对象

}

class Picture// 创建Picture类

{

private String title;// 定义字符串，表示图像标题

private int imageId;// 定义int变量，表示图像的二进制值

public Picture()// 默认构造函数

{

super();

}

public Picture(String title, int imageId)// 定义有参构造函数

{

super();

this.title = title;// 为图像标题赋值

this.imageId = imageId;// 为图像的二进制值赋值

}

public String getTitle() {// 定义图像标题的可读属性

return title;

}

public void setTitle(String title) {// 定义图像标题的可写属性

this.title = title;

}

public int getImageId() {// 定义图像二进制值的可读属性

return imageId;

}

public void setimageId(int imageId) {// 定义图像二进制值的可写属性

this.imageId = imageId;

}

}