开发板QT界面代码:

main.cpp

#include "mainwindow.h"

#include <QApplication>

#include "frminput.h"

#include "ui\_frminput.h"

#include "login.h"

#include "ui\_login.h"

#include <QCoreApplication>

#include <QFile>

#include <QDateTime>

#include <QThread>

#include <QDebug>

QString nowname = "NULL";

void MessageOutput(QtMsgType type, const QMessageLogContext &context, const QString &msg)

{

QByteArray localMsg = msg.toLocal8Bit();

QString strMsg("");

switch(type)

{

case QtDebugMsg:

strMsg = QString("Debug:");

break;

case QtWarningMsg:

strMsg = QString("Warning:");

break;

case QtCriticalMsg:

strMsg = QString("Critical:");

break;

case QtFatalMsg:

strMsg = QString("Fatal:");

break;

}

QLocale locale = QLocale::English;

QString strDateTime = locale.toString(QDateTime::currentDateTime(),"yyyy-MM-dd hh:mm:ss ddd");

QString strMessage = strMsg + QString("File:%1 Line:%2 Function:%3 Message:%4 DateTime:%5").arg(context.file).arg(context.line).arg(context.function).arg(localMsg.constData()).arg(strDateTime);

static QMutex mutex;

mutex.lock();

QFile file1("/workspace/qtlog.txt");

QFile file2("/workspace/qtlog1.txt");

file1.open(QIODevice::WriteOnly | QIODevice::Append);

if(file1.size() >= 20000000 )

{

file1.close();

file2.remove();

QFile::copy("/workspace/qtlog.txt","/workspace/qtlog1.txt");

file1.remove();

QFile file3("/workspace/qtlog.txt");

file3.open(QIODevice::WriteOnly | QIODevice::Append);

QTextStream stream(&file3);

stream << strMessage << endl;

}

else

{

QTextStream stream(&file1);

stream << strMessage << endl;

}

mutex.unlock();

}

int main(int argc, char \*argv[])

{

qInstallMessageHandler(MessageOutput);

QApplication a(argc, argv);

qApp->setFont(QFont("Microsoft YaHei", 10));

frmInput::Instance()->Init("control", "lightgray", 10, 10);

MainWindow w;

w.show();

return a.exec();

}

虚拟键盘：

void frmInput::InitForm()

{

this->mousePressed = false;

this->setWindowFlags(Qt::Tool | Qt::WindowStaysOnTopHint | Qt::FramelessWindowHint);

QDesktopWidget w;

deskWidth = w.availableGeometry().width();

deskHeight = w.availableGeometry().height();

frmWidth = this->width();

frmHeight = this->height();

if (QSqlDatabase::contains("qt\_sql\_default\_connection"))

{

DbConn = QSqlDatabase::database("qt\_sql\_default\_connection");

}

else

{

DbConn = QSqlDatabase::addDatabase("QSQLITE","front");

qDebug()<<"applicationDirPath:"<<qApp->applicationDirPath();

DbConn.setDatabaseName(qApp->applicationDirPath() + "/pinyin.db");

}

bool isSuccess = DbConn.open();

if(!isSuccess)

{

qDebug() << "fonts:" <<DbConn.lastError().text();

qDebug("DbConn.open failed");

}

else

{

qDebug() << "open db for chinese";

}

isFirst = true;

isPress = false;

timerPress = new QTimer(this);

connect(timerPress, SIGNAL(timeout()), this, SLOT(reClicked()));

currentWidget = 0;

currentLineEdit = 0;

currentTextEdit = 0;

currentPlain = 0;

currentBrowser = 0;

currentEditType = "";

currentPosition = "";

currentStyle = "";

btnFontSize = 10;

labFontSize = 10;

currentType = "min";

changeType(currentType);

QList<QPushButton \*> btn = this->findChildren<QPushButton \*>();

foreach (QPushButton \* b, btn) {

connect(b, SIGNAL(clicked()), this, SLOT(btn\_clicked()));

}

connect(qApp, SIGNAL(focusChanged(QWidget \*, QWidget \*)),

this, SLOT(focusChanged(QWidget \*, QWidget \*)));

qApp->installEventFilter(this);

}

主界面

void MainWindow::on\_btn\_login\_clicked()

{

this->close();

connect(&ln,SIGNAL(sendsignal()),this,SLOT(reshow()));

qDebug() << "The user login.";

this->ln.show();

}

void MainWindow::on\_btn\_offline\_clicked()

{

this->close();

connect(&ou,SIGNAL(sesend()),this,SLOT(reshow()));

qDebug() << "Users use the offline mode";

this->ou.show();

}

void MainWindow::on\_btn\_exit\_clicked()

{

this->close();

}

void MainWindow::reshow()

{

this->show();

}

登录界面

void Login::on\_btn\_login\_clicked()

{

QString username;

QString password,strmd5;

QByteArray md5;

QString mac;

username = this->ui->username->text();

password = this->ui->password->text();

nowname = username;

/\*进行md5加密\*/

md5 = QCryptographicHash::hash(password.toLatin1(),QCryptographicHash::Md5);

strmd5.append(md5.toHex());

qDebug() << strmd5;

mac = getHostMacAddress();

QJsonObject json;

json.insert("name",username);

json.insert("password",strmd5);

json.insert("mac",mac);

QJsonDocument document;

document.setObject(json);

QByteArray byteArray = document.toJson(QJsonDocument::Compact);

qDebug() << byteArray;

QUrl url("http://192.168.100.61:8080/dev/login");

QNetworkRequest request;

request.setUrl(url);

request.setHeader(QNetworkRequest::ContentTypeHeader,QVariant("application/json"));

manager = new QNetworkAccessManager(this);

connect(manager,SIGNAL(finished(QNetworkReply\*)),this,SLOT(replyfinish(QNetworkReply\*)));

QNetworkReply \*reply = manager->post(request,byteArray);

connect(&sw,SIGNAL(sendsignal()),this,SLOT(on\_btn\_cancel\_clicked()));

}

void Login::on\_btn\_cancel\_clicked()

{

emit sendsignal();

this->close();

}

QString Login::getHostMacAddress()

{

QList<QNetworkInterface> nets = QNetworkInterface::allInterfaces();

int nCnt = nets.count();

QString strMacAddr = "";

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

{

if(nets[i].flags().testFlag(QNetworkInterface::IsUp) && nets[i].flags().testFlag(QNetworkInterface::IsRunning) && !nets[i].flags().testFlag(QNetworkInterface::IsLoopBack))

{

strMacAddr = nets[i].hardwareAddress();

break;

}

}

return strMacAddr;

}

void Login::replyfinish(QNetworkReply \* reply)

{

QString strJsonText = reply->readAll();

qDebug()<<"The information returned: " <<strJsonText;

if (QString::compare(strJsonText,"success") == 0)

{

this->ui->password->clear();

this->ui->username->clear();

this->ui->lab\_status->clear();

qDebug() << "login success";

this->close();

sw.show();

}

else if(QString::compare(strJsonText,"name\_error") == 0)

{

this->ui->password->clear();

this->ui->username->clear();

this->ui->lab\_status->setText("Can not find this username!");

qWarning() << "Can not find this username!";

}

else

{

this->ui->password->clear();

this->ui->lab\_status->setText("Password is wrong!");

qWarning() << "Password is wrong!";

}

reply->deleteLater();

}

离线界面

void Offlinemenu::on\_btn\_camera\_clicked()

{

this->close();

connect(&sm,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "Users use the video monitoring.";

sm.show();

}

void Offlinemenu::on\_btn\_environ\_clicked()

{

this->close();

connect(&mv,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "Users use the environmental monitoring.";

this->mv.show();

}

void Offlinemenu::on\_btn\_back\_clicked()

{

emit sesend();

this->close();

}

void Offlinemenu::reshow()

{

this->show();

}

在线主菜单

void Showmenu::on\_btn\_back\_clicked()

{

emit sendsignal();

this->close();

}

void Showmenu::on\_btn\_environ\_clicked()

{

this->close();

connect(&mv,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "Users use the environmental monitoring.";

mv.show();

}

void Showmenu::reshow()

{

this->show();

}

void Showmenu::on\_btn\_monitor\_clicked()

{

this->close();

connect(&sm,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "Users use the video monitoring.";

sm.show();

}

离线主菜单

void Offlinemenu::on\_btn\_camera\_clicked()

{

this->close();

connect(&sm,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "Users use the video monitoring.";

sm.show();

}

void Offlinemenu::on\_btn\_environ\_clicked()

{

this->close();

connect(&mv,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "Users use the environmental monitoring.";

this->mv.show();

}

void Offlinemenu::on\_btn\_back\_clicked()

{

emit sesend();

this->close();

}

void Offlinemenu::reshow()

{

this->show();

}

环境菜单

void Menuenv::on\_btn\_realtime\_clicked()

{

this->close();

Env \*ev = new Env;

connect(ev,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "The user to see the real time environmental monitoring";

ev->show();

}

void Menuenv::on\_btn\_history\_clicked()

{

this->close();

connect(&hy,SIGNAL(sesignal()),this,SLOT(reshow()));

qDebug() << "The user to see the environmental history";

hy.show();

}

void Menuenv::on\_btn\_back\_clicked()

{

emit sesignal();

this->close();

}

void Menuenv::reshow()

{

this->show();

}

实时环境监测

Env::Env(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Env)

{

ui->setupUi(this);

setAttribute(Qt::WA\_StyledBackground);

setAttribute(Qt::WA\_DeleteOnClose,0);

setWindowTitle("Environment");

if (QSqlDatabase::contains("qt\_sql\_default\_connection"))

{

db = QSqlDatabase::database("qt\_sql\_default\_connection");

}

else

{

db = QSqlDatabase::addDatabase("QSQLITE");

db.setDatabaseName("/workspace/AirDataBase/AirData.db");

}

if (!db.open())

{

qDebug() << "can not open db" << endl;

return;

}

else

{

qDebug() << "can open db" << endl;

}

QTimer \*timer = new QTimer(this);

connect(timer, SIGNAL(timeout()), this, SLOT(queryTable()));

timer->start(500); // 每隔0.5s

tmin = 18.0;

tmax = 26.0;

hmin = 30.0;

hmax = 65.0;

amin = 0.0;

amax = 0.3;

QString name = nowname;

qDebug() << name;

QJsonObject json;

json.insert("name",name);

QJsonDocument document;

document.setObject(json);

QByteArray byteArray = document.toJson(QJsonDocument::Compact);

qDebug() << byteArray;

QUrl url("http://192.168.100.61:8080/dev/getthd");

QNetworkRequest request;

request.setUrl(url);

request.setHeader(QNetworkRequest::ContentTypeHeader,QVariant("application/json"));

manager = new QNetworkAccessManager(this);

connect(manager,SIGNAL(finished(QNetworkReply\*)),this,SLOT(replyfinish(QNetworkReply \*)));

manager->post(request,byteArray);

}

void Env::queryTable()

{

QDateTime mytime = QDateTime::currentDateTime();

QLocale locale = QLocale::English;

QString strFormat = "yyyy.MM.dd hh:mm:ss ap";

QString strDateTime = locale.toString(mytime, strFormat);

ui->lcd\_time->display(strDateTime);

QSqlQuery query(db);

query.prepare("SELECT \* FROM Data WHERE time = (SELECT MAX(time) FROM Data)");

if(!query.exec())

{

qDebug() << query.lastError();

}

if(query.next())

{

if (query.value(1).toDouble() < tmin || query.value(1).toDouble() > tmax)

{

ui->lcd\_temp->setStyleSheet("QLCDNumber{color:red}");

}

else

{

ui->lcd\_temp->setStyleSheet("QLCDNumber{color: rgb(255, 248, 0)}");

}

if (query.value(3).toDouble() < amin || query.value(3).toDouble() > amax)

{

ui->lcd\_amm->setStyleSheet("QLCDNumber{color:red}");

}

else

{

ui->lcd\_amm->setStyleSheet("QLCDNumber{color: rgb(255, 248, 0)}");

}

if (query.value(2).toDouble() < hmin || query.value(2).toDouble() > hmax)

{

ui->lcd\_hum->setStyleSheet("QLCDNumber{color:red}");

}

else

{

ui->lcd\_hum->setStyleSheet("QLCDNumber{color: rgb(255, 248, 0)}");

}

QString temp = QString::number(query.value(1).toDouble(),'f',1);

QString hum = QString::number(query.value(2).toDouble(),'f',1);

QString amm = QString::number(query.value(3).toDouble(),'f',1);

ui->lcd\_temp->display(temp);

ui->lcd\_hum->display(hum);

ui->lcd\_amm->display(amm);

}

}

void Env::replyfinish(QNetworkReply \*reply)

{

QString strJsonText = reply->readAll();

qDebug()<<"The information returned: " <<strJsonText;

QJsonDocument jsonDocument = QJsonDocument::fromJson(strJsonText.toLocal8Bit().data());

if(jsonDocument.isNull())

{

qWarning()<< "String NULL"<< strJsonText;

}

QJsonObject object = jsonDocument.object();

if (object.contains("temp\_min"))

{

QJsonValue value = object.value("temp\_min");

qDebug() << "value" << value;

tmin = value.toDouble();

qDebug() << "tmin: " << tmin;

}

if (object.contains("temp\_max"))

{

QJsonValue value = object.value("temp\_max");

qDebug() << "value" << value;

tmax = value.toDouble();

qDebug() << "tmax: " << tmax;

}

if (object.contains("hum\_min"))

{

QJsonValue value = object.value("hum\_min");

qDebug() << "value" << value;

hmin = value.toDouble();

qDebug() << "hmin: " << hmin;

}

if (object.contains("hum\_max"))

{

QJsonValue value = object.value("hum\_max");

qDebug() << "value" << value;

hmax = value.toDouble();

qDebug() << "hmax: " << hmax;

}

if (object.contains("amm\_min"))

{

QJsonValue value = object.value("amm\_min");

qDebug() << "value" << value;

amin = value.toDouble();

qDebug() << "amin: " << amin;

}

if (object.contains("amm\_max"))

{

QJsonValue value = object.value("amm\_max");

qDebug() << "value" << value;

amax = value.toDouble();

qDebug() << "amax: " << amax;

}

reply->deleteLater();

}

void Env::on\_btn\_th\_clicked()

{

this->close();

connect(&td,SIGNAL(stsend()),this,SLOT(reshow()));

td.show();

}

void Env::reshow()

{

this->show();

}

历史环境信息查看

void History::search(int count,int need)

{

ui->qtw->clearContents();

QSqlQuery query(db);

QDateTime now\_time = QDateTime::currentDateTime();

int nowtime = now\_time.toTime\_t();

int historytime = nowtime - need\*3600;

int historynext = historytime + count - historytime%count;

int nownext = nowtime + count - nowtime%count - 1;

int tmp,i;

int t1,t2;

int row = 23;

for(tmp=historynext; tmp<nownext; tmp=tmp+count)

{

query.clear();

t1 = tmp;

t2 = tmp + count -1;

QString t3 = QString::number(t1);

QString t4 = QString::number(t2);

QString sql= "SELECT \* FROM Data WHERE time BETWEEN ";

sql.append(t3);

sql.append(" and ");

sql.append(t4);

sql.append(" LIMIT 1 ");

query.prepare(sql);

if(!query.exec())

{

qDebug() << "error:" <<query.lastError();

}

if(query.first())

{

QDateTime mytime = QDateTime::fromTime\_t(query.value(0).toInt());

QLocale locale = QLocale::English;

QString strFormat = "yyyy.MM.dd ddd,hh:mm:ss ap";

QString strDateTime = locale.toString(mytime, strFormat);

ui->qtw->setItem(row,0,new QTableWidgetItem(strDateTime));

for(i = 1; i < 4; i++)

{

QTableWidgetItem \*myitem = new QTableWidgetItem(QString::number(query.value(i).toFloat(),'f',1));

myitem->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);

ui->qtw->setItem(row,i,myitem);

}

}

row--;

}

query.clear();

}

void History::on\_btn\_si\_clicked()

{

qDebug() << "Data on the user's query for 6 hours";

dbopen();

search(900,6);

dbclose();

}

void History::on\_btn\_tw\_clicked()

{

qDebug() << "Data on the user's query for 12 hours";

dbopen();

search(1800,12);

dbclose();

}

void History::on\_btn\_tt\_clicked()

{

qDebug() << "Data on the user's query for 24 hours";

dbopen();

search(3600,24);

dbclose();

}

void History::on\_btn\_qte\_clicked()

{

this->close();

connect(&qte,SIGNAL(qtsend()),this,SLOT(reshow()));

qte.show();

qDebug() << "User uses the selection of query on time!";

}

void History::reshow()

{

this->show();

}

用户自定义时段查看

Querytime::Querytime(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Querytime)

{

ui->setupUi(this);

setWindowTitle("Query on time");

setAttribute(Qt::WA\_StyledBackground);

QDateTime nowtime = QDateTime::currentDateTime();

int valtime = nowtime.toTime\_t();

valtime = valtime - 999;

QDateTime vtime = QDateTime::fromTime\_t(valtime);

ui->stime->setDateTime(vtime);

ui->etime->setDateTime(nowtime);

}

void Querytime::dbopen()

{

if (QSqlDatabase::contains("qt\_sql\_default\_connection"))

{

db = QSqlDatabase::database("qt\_sql\_default\_connection");

}

else

{

db = QSqlDatabase::addDatabase("QSQLITE","querytime");

db.setDatabaseName("/workspace/AirDataBase/AirData.db");

}

if (!db.open())

{

qDebug() << "can not open db history" << endl;

return;

}

else

{

qDebug() << "can open db his";

}

}

void Querytime::dbclose()

{

QString connection;

connection=db.connectionName();

db.close();

//qDebug() << connection;

QSqlDatabase::removeDatabase(connection);

}

void Querytime::on\_btn\_back\_clicked()

{

emit qtsend();

ui->tableWidget->clearContents();

this->close();

}

void Querytime::on\_btn\_query\_clicked()

{

dbopen();

ui->tableWidget->clearContents();

QDateTime starttime = ui->stime->dateTime();

QDateTime endtime = ui->etime->dateTime();

int st = starttime.toTime\_t();

int et = endtime.toTime\_t();

QSqlQuery query(db);

QString t1 = QString::number(st);

QString t2 = QString::number(et);

QString sql = "SELECT \* FROM Data WHERE time BETWEEN ";

sql.append(t1);

sql.append(" and ");

sql.append(t2);

query.prepare(sql);

if(!query.exec())

{

qDebug() << "error:" <<query.lastError();

}

int initialPos = query.at();

int pos = 0;

if (query.last())

{

pos = query.at() + 1;

}

else

{

pos = 0;

}

query.seek(initialPos);

ui->tableWidget->setRowCount(pos);

pos = pos - 1;

int i;

while(query.next())

{

QDateTime mytime = QDateTime::fromTime\_t(query.value(0).toInt());

QLocale locale = QLocale::English;

QString strFormat = "yyyy.MM.dd ddd,hh:mm:ss ap";

QString strDateTime = locale.toString(mytime, strFormat);

ui->tableWidget->setItem(pos,0,new QTableWidgetItem(strDateTime));

for(i = 1; i < 4; i++)

{

QTableWidgetItem \*myitem = new QTableWidgetItem(QString::number(query.value(i).toFloat(),'f',1));

myitem->setTextAlignment(Qt::AlignHCenter|Qt::AlignVCenter);

ui->tableWidget->setItem(pos,i,myitem);

}

--pos;

}

dbclose();

}

设置阈值

void Threshold::on\_btn\_sub\_clicked()

{

QString name = nowname;

qDebug() << "threshold " << name;

if (QString::compare(name,"NULL") == 0)

{

qWarning() << "set failed!";

QMessageBox::information(this,tr("set"),tr("Offline mode can't use this feature"));

return;

}

QString tmax = ui->tmax->text();

QString tmin = ui->tmin->text();

QString hmax = ui->hmax->text();

QString hmin = ui->hmin->text();

QString amax = ui->amax->text();

QString amin = ui->amin->text();

QJsonObject json;

json.insert("name",name);

json.insert("tmax",tmax);

json.insert("tmin",tmin);

json.insert("hmax",hmax);

json.insert("hmin",hmin);

json.insert("amax",amax);

json.insert("amin",amin);

QJsonDocument document;

document.setObject(json);

QByteArray byteArray = document.toJson(QJsonDocument::Compact);

qDebug() << byteArray;

QUrl url("http://192.168.100.61:8080/dev/setthd");

QNetworkRequest request;

request.setUrl(url);

request.setHeader(QNetworkRequest::ContentTypeHeader,QVariant("application/json"));

manager = new QNetworkAccessManager(this);

connect(manager,SIGNAL(finished(QNetworkReply\*)),this,SLOT(replyfinish(QNetworkReply \*)));

manager->post(request,byteArray);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

}

void Threshold::on\_btn\_back\_clicked()

{

emit stsend();

this->close();

}

void Threshold::replyfinish(QNetworkReply \* reply)

{

QString strJsonText = reply->readAll();

qDebug()<<"The information returned: " <<strJsonText;

if (QString::compare(strJsonText,"success") == 0)

{

qDebug() << "set success!";

QMessageBox::information(this,tr("set"),tr("set success"));

}

else

{

qWarning() << "set failed!";

QMessageBox::information(this,tr("set"),tr("set failed"));

}

reply->deleteLater();

}

接收本地视频推流

Stream::Stream(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Stream)

{

setWindowTitle("Camera");

setAttribute(Qt::WA\_StyledBackground);

ui->setupUi(this);

btn\_startisclick = false;

img = new QImage;

scaledimg = new QImage;

showTime=new QTimer;

showTime->setInterval(42);

connect(showTime, SIGNAL(timeout()), this, SLOT(flushPic()));

manager = new QNetworkAccessManager(this);

connect(manager, SIGNAL(finished(QNetworkReply\*)), this, SLOT(replyFinished(QNetworkReply\*)));

openps = new QProcess();

closeps = new QProcess();

}

void Stream::on\_btn\_start\_clicked()

{

pu = "http://127.0.0.1:8080/?action=snapshot/";

if(btn\_startisclick == false)

{

ui->lab\_camera->clear();

btn\_startisclick = true;

openps->start("/workspace/VideoSwitch/open.sh");

qDebug() << "Now start camera!";

ui->btn\_start->setText("Stop");

showTime->start();

}

else

{

ui->lab\_camera->clear();

btn\_startisclick = false;

ui->btn\_start->setText("Start");

showTime->stop();

qDebug() << "Now stop camera";

}

}

void Stream::on\_btn\_shot\_clicked()

{

if(!QDir("photo").exists())

{

QDir dir;

dir.mkpath("photo");

qDebug() << "mkdir photo";

}

QString str\_time,path;

QDateTime time = QDateTime::currentDateTime();

str\_time = time.toString("yyyy-MM-dd\_hh-mm-ss");

path="./photo/"+str\_time+".jpg";

QPixmap::fromImage(\*img).save(path, "JPG", 100);

\*scaledimg=img->scaled(160,120,Qt::KeepAspectRatio);

ui->lab\_pic->setPixmap(QPixmap::fromImage(\*scaledimg));

QMessageBox::information(this,tr("save"),tr("PICTURE has been saved to folder 'photo!'"));

qDebug() << "PICTURE has been saved to folder 'photo!'";

}

void Stream::on\_btn\_quit\_clicked()

{

showTime->stop();

ui->lab\_camera->clear();

ui->lab\_pic->clear();

emit sesignal();

this->close();

}

void Stream::flushPic()

{

request.setUrl(QUrl(pu));

manager->get(request);

}

void Stream::replyFinished(QNetworkReply\* reply)

{ //save the picture

QPixmap pix;

QByteArray data = reply->readAll();

pix.loadFromData(data, "JPG");

if(!QDir("temp").exists())

{

QDir dir;

dir.mkpath("temp");

qDebug() << "mkdir temp.";

}

pix.save("temp/temp1.jpg", "JPG", 50);

if(! ( img->load("temp/temp1.jpg") ) )

{

showTime->stop();

QMessageBox::information(this,tr("warning!"),tr("The image loaded failure!"));

qWarning() << "The image loaded failure!";

btn\_startisclick = false;

return;

}

\*scaledimg=img->scaled(640,480,Qt::KeepAspectRatio);

ui->lab\_camera->setPixmap(QPixmap::fromImage(\*scaledimg));

}

void Stream::on\_Streamclose\_clicked()

{

closeps->start("/workspace/VideoSwitch/close.sh");

QMessageBox::information(this,tr("close"),tr("stream closed!"));

}

设备端环境数据获取代码:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<strings.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<signal.h>

#include<time.h>

#include<errno.h>

#include<arpa/inet.h>

#include<semaphore.h>

#include<pthread.h>

#include<sqlite3.h>

#define GotDATALEN 11

//#define DESIP "192.168.100.136"

#define DESIP "47.99.195.202"

//#define DESIP "192.168.100.61"

#define DESPORT 8880

#define VIDEOIP "0.0.0.0"

#define VIDEOPORT 8881

#define HEARTTIME 5

int fd, fdw, sfd, videofd, firstflag=1, connectflag;

FILE \*logfp;

sem\_t sem; //when sem is not 0, need connect

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

char MAC[] = "F4:5E:AB:57:F0:FF";

//char DESIP[20] = "";

sqlite3 \*db;

char \*zErrMsg = 0;

void printlog(int level, time\_t time, char type[], char detail[])

{

char s[100], temp[100], \*c;

logfp = fopen("/workspace/log", "a");

strcpy(s, ctime(&time));

s[strlen(s)-1] = '\0';

fprintf(logfp, "[LEVEL%d] [%s] %s: %s\n", level, s, type, detail);

fclose(logfp);

}

void sigfun(int flag)

{

printlog(3, time(NULL), "Program", "program end with sigfun");

close(fd);

close(fdw);

close(sfd);

close(videofd);

//fclose(logfp);

sqlite3\_close(db);

if(flag == 1)

{

system("reboot");

}

exit(0);

}

void \*VideoSwitch()

{

int cfd, ret, len;

struct sockaddr\_in seraddr;

char buff[256];

pid\_t pid;

videofd = socket(AF\_INET, SOCK\_STREAM, 0);

if(videofd < 0)

{

perror("VideoSwitch:socket error\n");

return NULL;

}

seraddr.sin\_family = AF\_INET;

seraddr.sin\_port = htons(VIDEOPORT);

seraddr.sin\_addr.s\_addr = inet\_addr(VIDEOIP);

ret = -1;

while(ret < 0)

{

ret = bind(videofd, (struct sockaddr\*)&seraddr, sizeof(struct sockaddr));

if(ret < 0)

{

perror("VideoSwitch:bind error\n");

close(videofd);

return NULL;

}

sleep(5);

}

printf("VideoSwitch:bind ok\n");

listen(videofd, 5);

len = sizeof(struct sockaddr);

while(1)

{

cfd = accept(videofd, (struct sockaddr\*)&seraddr, &len);

if(cfd < 0)

{

perror("VideoSwitch:accept error\n");

continue;

}

printf("VideoSwitch:link ok\n");

bzero(buff, sizeof(buff));

ret = recv(cfd, buff, sizeof(buff), 0);

if(ret <= 0)

{

continue;

}

printf("VideoSwitch: said:%s\n",buff);

if(strcmp(buff, "on") == 0)

{

printlog(2, time(NULL), "Camera", "got camera open request");

pid = fork();

if(pid == 0)

{

system("bash /workspace/VideoSwitch/open.sh");

printf("VideoSwitch's child die with 0\n");

exit(0);

}

}

else if(strcmp(buff, "off") == 0)

{

printlog(2, time(NULL), "Camera", "got camera close request");

system(". /workspace/VideoSwitch/close.sh&");

}

else if(strcmp(buff, "reboot") == 0)

{

sigfun(1);

system("reboot");

}

}

return NULL;

}

void \*connecttest()

{

int ret;

struct sockaddr\_in seraddr;

pid\_t pid;

while(1)

{

sem\_wait(&sem);

//socket

sfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

if(sfd < 0)

{

perror("socket error\n");

close(fd);

close(fdw);

exit(0);

}

seraddr.sin\_family = AF\_INET;

seraddr.sin\_port = htons(DESPORT);

seraddr.sin\_addr.s\_addr = inet\_addr(DESIP);

ret = connect(sfd, (struct sockaddr\*)&seraddr, sizeof(struct sockaddr));

if(ret < 0)

{

sem\_post(&sem);

if(firstflag == 1)

{

perror("Connect error!"\

" The program will run without network\n");

printlog(3, time(NULL), "NetConnect", "net connect failed");

firstflag = 0;

}

else

{

perror("Reconnect failed!\n");

printlog(3, time(NULL), "NetConnect", "net reconnect failed");

}

}

else

{

pthread\_mutex\_lock(&mutex);

connectflag = 1;

pthread\_mutex\_unlock(&mutex);

if(firstflag == 0)

{

printf("Reconnect success!\n");

printlog(3, time(NULL), "NetConnect", "net reconnect success");

sem\_init(&sem, 0, 0);

}

else

{

printf("Connect success!\n");

printlog(3, time(NULL), "NetConnect", "net connect success");

sem\_init(&sem, 0, 0);

firstflag = 0;

}

pid = fork();

if(pid == 0)

{

system("bash /workspace/SetTime/settime.sh >/dev/null 2>&1 &");

exit(0);

}

}

sleep(5);

}

}

void \*heartbeat()

{

int count = 0, flag, ret, connecttemp;

fd\_set fdset;

char heartbeat[] = "heartbeat", buff[50]={0};

struct timeval tv;

pthread\_mutex\_lock(&mutex);

connecttemp = connectflag;

pthread\_mutex\_unlock(&mutex);

while(1)

{

while(connecttemp == 0)

{

pthread\_mutex\_lock(&mutex);

connecttemp = connectflag;

pthread\_mutex\_unlock(&mutex);

sleep(1);

}

ret = send(sfd, heartbeat, strlen(heartbeat), 0);

if(ret < 0)

{

printf("send ret < 0\n");

sleep(HEARTTIME);

continue;

}

else if(ret == 0)

{

printf("send ret == 0\n");

sleep(HEARTTIME);

continue;

}

bzero(&buff, sizeof(buff));

count = 0;

while(count < 5)

{

FD\_ZERO(&fdset);

FD\_SET(sfd, &fdset);

tv.tv\_sec = 1;

tv.tv\_usec = 0;

ret = select(sfd+1, &fdset, NULL, NULL, &tv);

if(ret <= 0)

{

//printf("heartbeat select ret <= 0\n");

}

if(FD\_ISSET(sfd, &fdset))

{

ret = recv(sfd, buff, sizeof(buff), 0);

if(ret < 0)

{

//printf("heartbeat recv < 0\n");

}

if(ret == 0)

{

//printf("heartbeat recv = 0\n");

printlog(3, time(NULL), "NetConnect", "net connection broken");

}

if(strcmp(buff, "heartbeat") == 0)

{

//link is alive

//printf("heartbeat ok\n");

break;

}

else

{

printf("got wrong msg when heartbeat\n");

printlog(3, time(NULL), "NetConnect", "got wrong msg when heartbeat");

}

}

count++;

sleep(1);

}

if(count == 5)

{

// connect broken

printf("heart beat failed, reconnect\n");

printlog(3, time(NULL), "NetConnect", "heartbeat failed");

pthread\_mutex\_lock(&mutex);

connectflag = 0;

pthread\_mutex\_unlock(&mutex);

close(sfd);

sem\_post(&sem);

}

else

{

sleep(HEARTTIME);

}

}

}

static int callback(void \*NotUsed,int argc, char \*\*argv, char \*\*azColName)

{

int i;

for(i = 0;i<argc;i++)

{

printf("%s = %s\n",azColName[i],argv[i]?argv[i]:"NULL");

}

printf("\n");

return 0;

}

void linkDB(char DBfilename[])

{

int ret;

ret = sqlite3\_open(DBfilename, &db);

if(ret)

{

printf("open error, %s\n",sqlite3\_errmsg(db));

printlog(3, time(NULL), "DataBase", "database open error");

exit(0);

}

else

{

printf("open database successfully\n");

printlog(3, time(NULL), "DataBase", "database open successfully");

}

//create sql statement

char sql[] = "CREATE TABLE Data(\n"\

"time INTEGER PRIMARY KEY NOT NULL,\n"\

"temperature REAL NOT NULL,\n"\

"humidity REAL NOT NULL,\n"\

"ammonia REAL NOT NULL);";

//execute sql statement

ret = sqlite3\_exec(db,sql,callback,0,&zErrMsg);

if(ret != SQLITE\_OK)

{

printf("SQL:%s\n",zErrMsg);

sqlite3\_free(zErrMsg);

}

else

{

printf("table create success\n");

}

}

int main(int argc, char \*argv[])

{

char request[8]={0}, get[50];

char GOT[GotDATALEN];

char buff[100];

char DBfilename[100];

char sqltemp[100];

int temp, i, ret, count, timeoutflag;

pthread\_t tid1, tid2, tid3;

struct sockaddr\_in seraddr;

struct timeval tv;

time\_t oldt, newt;

float temperature, ammonia, humidity;

fd\_set fdset, netfdset;

if(argc < 3)

{

printf("Need two arguments!\n");

exit(0);

}

printlog(3, time(NULL), "Program", "program start");

//logfp = fopen("/workspace/log", "a");

//link to database

linkDB(argv[1]);

//get MAC

puts(argv[2]);

strcpy(MAC, argv[2]);

sem\_init(&sem, 0, 1);

//"request" is a Request Message

request[0] = 0x01;

request[1] = 0x03;

request[2] = 0x00;

request[3] = 0x20;

request[4] = 0x00;

request[5] = 0x03;

request[6] = 0x04;

request[7] = 0x01;

fd = open("/dev/ttyO1",O\_RDONLY,0777);

fdw = open("/dev/ttyO1",O\_WRONLY,0777);

//for test

/\*GOT[0] = 0x01;

GOT[1] = 0x03;

GOT[2] = 0x06;

GOT[3] = 0x00;

GOT[4] = 0x01;

GOT[5] = 0x00;

GOT[6] = 0x01;

GOT[7] = 0x00;

GOT[8] = 0x01;

GOT[9] = 0xdd;

GOT[10] = 0xdd;

//for test

\*/

pthread\_create(&tid1, NULL, connecttest, NULL);

pthread\_create(&tid2, NULL, heartbeat, NULL);

pthread\_create(&tid3, NULL, VideoSwitch, NULL);

signal(SIGINT, sigfun);

signal(SIGPIPE, SIG\_IGN);

count = 0;

while(1)

{

write(fdw, request, sizeof(request));

bzero(GOT,sizeof(GOT));

count = 0;

timeoutflag = 0;

while(count < GotDATALEN)

{

FD\_ZERO(&fdset);

FD\_SET(fd, &fdset);

tv.tv\_sec = 5;

tv.tv\_usec = 0;

ret = select(fd+1, &fdset, NULL, NULL, &tv);

if(ret < 0)

{

printf("get date select < 0,"\

" please check the sensor\n");

printlog(3, time(NULL), "Sensor", "sensor select error");

}

else if(ret == 0)

{

timeoutflag = 1;

break;

}

if(FD\_ISSET(fd, &fdset))

{

bzero(get, sizeof(get));

ret = read(fd,get,sizeof(get));

if(ret > 0)

{

for(i = 0; i < ret; i++)

{

//printf("%02x ",(int)get[i]);

GOT[count+i]=get[i];

}

//printf("\n");

count += ret;

if(count == GotDATALEN)break;

}

else

{

//printf("read error\n");

}

}

}

//printf("\n");

//got count > correct length

if(count > GotDATALEN)

{

printf("Got too many data!\n");

printlog(3, time(NULL), "Sensor", "sensor select got too many data");

continue;

}

if(timeoutflag == 1)

{

printf("get date select time out, please check the sensor\n");

printlog(3, time(NULL), "Sensor", "sensor select time out");

continue;

}

//got wrong data

if((int)GOT[0]!=0x01)

{

printf("error: GOT[0]!=0x01, continue\n");

continue;

}

if((int)GOT[1]!=0x03)

{

printf("error: GOT[1]!=0x03, continue\n");

continue;

}

if((int)GOT[2]!=0x06)

{

printf("error: GOT[2]!=0x06, continue\n");

continue;

}

//get time

oldt = time(NULL);

ammonia = ((int)GOT[3]\*256+(int)GOT[4])/10.0;

temperature = ((int)GOT[5]\*256+(int)GOT[6])/10.0,

humidity = ((int)GOT[7]\*256+(int)GOT[8])/10.0;

printf("%s Temperature:%.1f Humidity:%.1f ammonia:%.1f\n",

ctime(&oldt), temperature, humidity, ammonia);

//insert database

sprintf(sqltemp,

"INSERT INTO Data values(%d,%.1f,%.1f,%.1f);",

(int)oldt, temperature, humidity, ammonia);

ret = sqlite3\_exec(db,sqltemp,callback,0,&zErrMsg);

if(ret != SQLITE\_OK)

{

printf("SQL:%s\n",zErrMsg);

printlog(3, time(NULL), "DataBase", zErrMsg);

sqlite3\_free(zErrMsg);

}

else

{

//printf("insert success\n");

}

//if connected, send to server

pthread\_mutex\_lock(&mutex);

temp = connectflag;

pthread\_mutex\_unlock(&mutex);

if(temp == 1)

{

FD\_ZERO(&netfdset);

FD\_SET(sfd, &netfdset);

tv.tv\_sec = 0;

tv.tv\_usec = 0;

ret = select(sfd+1, NULL, &netfdset, NULL, &tv);

if(ret < 0)

{

printf("net select < 0\n");

pthread\_mutex\_lock(&mutex);

connectflag = 0;

pthread\_mutex\_unlock(&mutex);

close(sfd);

sem\_post(&sem);

}

else if(ret == 0)

{

printf("net select == 0\n");

printlog(3, time(NULL), "NetConnect", "net connection Broken");

pthread\_mutex\_lock(&mutex);

connectflag = 0;

pthread\_mutex\_unlock(&mutex);

close(sfd);

sem\_post(&sem);

}

else

{

pthread\_mutex\_lock(&mutex);

connectflag = 1;

pthread\_mutex\_unlock(&mutex);

if(FD\_ISSET(sfd, &netfdset))

{

sprintf(buff,"%s %d %.1f %.1f %.1f",

MAC, (int)oldt, temperature, humidity, ammonia);

ret = send(sfd, buff, strlen(buff), 0);

if(ret < 0)

{

printf("send error\n");

printlog(3, time(NULL), "NetConnect", "send error");

pthread\_mutex\_lock(&mutex);

connectflag = 0;

pthread\_mutex\_unlock(&mutex);

close(sfd);

sem\_post(&sem);

}

}

}

}

newt = oldt;

while(newt == oldt)

{

newt = time(NULL);

usleep(1000);

}

}

sqlite3\_close(db);

}

服务器后端代码:

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

Import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import org.springframework.web.filter.CorsFilter;

@Configuration

public class CorsConfig {

private CorsConfiguration buildConfig() {

CorsConfiguration corsConfiguration = new CorsConfiguration();

corsConfiguration.addAllowedOrigin("\*");

corsConfiguration.addAllowedHeader("\*");

corsConfiguration.addAllowedMethod("\*");

return corsConfiguration;

}

@Bean

public CorsFilter corsFilter() {

UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

source.registerCorsConfiguration("/\*\*", buildConfig());

return new CorsFilter(source);

}

}

import com.lkc.service.DeviceService;

import com.lkc.service.UcompDService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class DeviceController {

@Autowired

private DeviceService deviceService;

@Autowired

private UcompDService ucompDService;

@PostMapping("/dev/find")

public String findDevice(@RequestParam("name") String name){

return deviceService.findDevice(name);

}

@PostMapping("/dev/url")

public String findUrl(@RequestParam("mac") String mac){

return deviceService.findUrlByMac(mac);

}

@PostMapping("/dev/video")

public void videoControll (@RequestParam("mac") String mac,

@RequestParam("controll") String controll){

deviceService.videoControll(mac,controll);

}

@PostMapping("/dev/reboot")

public String rebootControll (@RequestParam("mac") String mac,

@RequestParam("controll") String controll){

return deviceService.rebootControll(mac,controll);

}

}

import com.lkc.entity.Sensor;

import com.lkc.repository.SensorRepository;

import com.lkc.service.SensorService;

import net.sf.json.JSONArray;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

public class SensorController {

@Autowired

private SensorService sensorService;

@Autowired

private SensorRepository sensorRepository;

@PostMapping("/find/custom")

public String findByCustom(@RequestParam("sec") Integer second,

@RequestParam("start") Integer start,

@RequestParam("mac") String mac){

List<Sensor> list = sensorService.findByCustom(second,start,mac);

JSONArray array = JSONArray.fromObject(list);

String jsonstr = array.toString();

return jsonstr;

}

@PostMapping("/find/default")

public String findByDefault(@RequestParam("needtime") Integer needtime,

@RequestParam("nowtime") Integer nowtime,

@RequestParam("mac") String mac){

List<Sensor> list = sensorService.findByDefault(needtime,nowtime,mac);

JSONArray array = JSONArray.fromObject(list);

String jsonstr = array.toString();

return jsonstr;

}

@PostMapping("/find/bigdata")

public String findBigData(@RequestParam("lasttime") Integer lasttime,

@RequestParam("mac") String mac){

List<Sensor> list = sensorService.findAllByMacAndTime(mac,lasttime);

JSONArray array = JSONArray.fromObject(list);

String jsonstr = array.toString();

return jsonstr;

}

}

import com.lkc.entity.DevName;

import com.lkc.entity.DevSetth;

import com.lkc.entity.Threshold;

import com.lkc.entity.UserMac;

import com.lkc.service.ThresholdService;

import com.lkc.service.UcompDService;

import net.sf.json.JSONObject;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.repository.query.Param;

import org.springframework.web.bind.annotation.\*;

@RestController

public class UcompDController {

@Autowired

private UcompDService ucompDService;

@Autowired

private ThresholdService thresholdService;

@PostMapping(value = "/dev/login")

public String login(@RequestBody UserMac usermac) {

return ucompDService.login(usermac);

}

@PostMapping(value = "/dev/delete")

public String deleteDevice(@Param("name")String name,@Param("mac")String mac){

return ucompDService.deleteByNameAndMac(name,mac);

}

@PostMapping("/dev/getthd")

public String findThreshold(@RequestBody DevName devName){

Threshold threshold = thresholdService.findByName(devName.getName());

JSONObject jsonObject = JSONObject.fromObject(threshold);

String jsonstr = jsonObject.toString();

return jsonstr;

}

@PostMapping("/dev/setthd")

public String setThreshold(@RequestBody DevSetth devSetth){

return thresholdService.setThreshold(devSetth.getTmax(),devSetth.getTmin(),

devSetth.getHmax(),devSetth.getHmin(),

devSetth.getAmax(),devSetth.getAmin(),

devSetth.getName());

}

}

import com.lkc.entity.Email;

import com.lkc.entity.Threshold;

import com.lkc.service.MailService;

import com.lkc.service.ThresholdService;

import com.lkc.service.UserService;

import net.sf.json.JSONObject;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

@RestController

public class UserController {

@Autowired

private UserService userService;

@Autowired

private ThresholdService thresholdService;

@Autowired

private MailService mailService;

@PostMapping("/user")

public String addUser(@RequestParam("name") String name,

@RequestParam("password") String password,

@RequestParam("email") String email){

return userService.addUser(name,password,email);

}

@PostMapping("/user/login")

public String login(@RequestParam("name") String name,

@RequestParam("password") String password){

return userService.login(name,password);

}

@PostMapping("/user/chg")

public String changePassword(@RequestParam("name") String name,

@RequestParam("newpassword") String password){

return userService.changePassword(name,password);

}

@GetMapping("/user/regtime")

public String findRegtime(@RequestParam("name") String name){

return userService.findRegtime(name);

}

@PostMapping("/user/getthd")

public String findThreshold(@RequestParam("name")String name){

Threshold threshold = thresholdService.findByName(name);

JSONObject jsonObject = JSONObject.fromObject(threshold);

String jsonstr = jsonObject.toString();

return jsonstr;

}

@PostMapping("/user/setthd")

public String setThreshold(@RequestParam("tmax")float tmax,@RequestParam("tmin")float tmin,@RequestParam("hmax")float hmax,@RequestParam("hmin")float hmin,

@RequestParam("amax")float amax,@RequestParam("amin")float amin,

@RequestParam("name")String name){

return thresholdService.setThreshold(tmax,tmin,hmax,hmin,amax,amin,name);

}

@PostMapping("/user/setmailfunc")

public String setEmailFunction(@RequestParam("name")String name,

@RequestParam("control")String control){

return mailService.setEmailFunction(name,control);

}

@PostMapping("/user/getmailfunc")

public String getEmailFunction(@RequestParam("name")String name){

return mailService.findByName(name).getControl();

}

@PostMapping("/user/remember")

public String setCode(@RequestParam("name")String name,

@RequestParam("email")String email){

return mailService.setCode(name,email);

}

@PostMapping("/user/code")

public String checkcodeAndChg(@RequestParam("name")String name,

@RequestParam("code")String code,

@RequestParam("password")String password){

return mailService.checkcodeAndChg(name,code,password);

}

}

import javax.persistence.\*;

@Entity

@Table(name = "device\_info")

public class Device {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

private Integer id;

@Column(name = "mac")

private String mac;

@Column(name = "status")

private Integer status;

@Column(name = "lasttime")

private Integer lasttime;

@Column(name = "url")

private String url;

public String getMac() {

return mac;

}

public void setMac(String mac) {

this.mac = mac;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public Integer getStatus() { return status; }

public void setStatus(Integer status) { this.status = status; }

public Integer getLasttime() {

return lasttime;

}

public void setLasttime(Integer lasttime) {

this.lasttime = lasttime;

}

public String getUrl() {

return url;

}

public void setUrl(String url) {

this.url = url;

}

}

public class DevName {

private String name;

public String getName() {

return name;

}

}

public class DevSetth {

private String name;

private float tmax;

private float tmin;

private float hmax;

private float hmin;

private float amax;

private float amin;

public String getName() {

return name;

}

public float getAmax() {

return amax;

}

public float getAmin() {

return amin;

}

public float getHmax() {

return hmax;

}

public float getHmin() {

return hmin;

}

public float getTmax() {

return tmax;

}

public float getTmin() {

return tmin;

}

public void setName(String name) {

this.name = name;

}

public void setAmax(float amax) {

this.amax = amax;

}

public void setAmin(float amin) {

this.amin = amin;

}

public void setHmax(float hmax) {

this.hmax = hmax;

}

public void setHmin(float hmin) {

this.hmin = hmin;

}

public void setTmax(float tmax) {

this.tmax = tmax;

}

public void setTmin(float tmin) {

this.tmin = tmin;

}

}

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "email\_info")

public class Email {

@Id

@Column(name = "name")

private String name;

@Column(name = "email")

private String email;

@Column(name = "code")

private String code;

@Column(name = "control")

private String control;

public String getControl() {

return control;

}

public void setControl(String control) {

this.control = control;

}

public String getName() {

return name;

}

public String getCode() {

return code;

}

public String getEmail() {

return email;

}

public void setName(String name) {

this.name = name;

}

public void setCode(String code) {

this.code = code;

}

public void setEmail(String email) {

this.email = email;

}

}

package com.lkc.entity;

public class RetDevice {

private Integer number;

private String mac;

private Integer status;

private String lasttime;

public String getMac() {

return mac;

}

public void setMac(String mac) {

this.mac = mac;

}

public Integer getNumber() {

return number;

}

public void setNumber(Integer number) {

this.number = number;

}

public Integer getStatus() {

return status;

}

public void setStatus(Integer status) {

this.status = status;

}

public void setLasttime(String lasttime) {

this.lasttime = lasttime;

}

public String getLasttime() {

return lasttime;

}

}

import javax.persistence.\*;

@Entity

@Table(name = "sensor\_info")

public class Sensor {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

@Column(name = "id")

private Integer id;

@Column(name = "time")

private Integer time;

@Column(name = "temperature")

private float temperature;

@Column(name = "humidity")

private float humidity;

@Column(name = "ammonia")

private float ammonia;

@Column(name = "mac")

private String mac;

public float getAmmonia() {

return ammonia;

}

public void setAmmonia(float ammonia) {

this.ammonia = ammonia;

}

public Integer getTime() {

return time;

}

public void setTime(Integer time) {

this.time = time;

}

public float getHumidity() {

return humidity;

}

public void setHumidity(float humidity) {

this.humidity = humidity;

}

public float getTemperature() {

return temperature;

}

public void setTemperature(float temperature) {

this.temperature = temperature;

}

public String getMac() {

return mac;

}

public void setMac(String mac) {

this.mac = mac;

}

}

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "threshold\_info")

public class Threshold {

@Id

@Column(name = "name")

private String name;

@Column(name = "temp\_max")

private float temp\_max;

@Column(name = "temp\_min")

private float temp\_min;

@Column(name = "hum\_max")

private float hum\_max;

@Column(name = "hum\_min")

private float hum\_min;

@Column(name = "amm\_max")

private float amm\_max;

@Column(name = "amm\_min")

private float amm\_min;

public String getName() {

return name;

}

public float getAmm\_max() {

return amm\_max;

}

public float getAmm\_min() {

return amm\_min;

}

public float getHum\_max() {

return hum\_max;

}

public float getHum\_min() {

return hum\_min;

}

public float getTemp\_max() {

return temp\_max;

}

public float getTemp\_min() {

return temp\_min;

}

public void setName(String name) {

this.name = name;

}

public void setAmm\_max(float amm\_max) {

this.amm\_max = amm\_max;

}

public void setAmm\_min(float amm\_min) {

this.amm\_min = amm\_min;

}

public void setHum\_max(float hum\_max) {

this.hum\_max = hum\_max;

}

public void setHum\_min(float hum\_min) {

this.hum\_min = hum\_min;

}

public void setTemp\_max(float temp\_max) {

this.temp\_max = temp\_max;

}

public void setTemp\_min(float temp\_min) {

this.temp\_min = temp\_min;

}

}

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "user\_info")

public class User {

@Id

@Column(name = "name")

private String name;

@Column(name = "password")

private String password;

@Column(name = "regtime")

private Integer regtime;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Integer getRegtime() {

return regtime;

}

public void setRegtime(Integer regtime) {

this.regtime = regtime;

}

}

import javax.persistence.\*;

@Entity

@Table(name = "usercompdev\_info")

public class UserCompDev {

@Id

@GeneratedValue(strategy= GenerationType.IDENTITY)

private Integer id;

@Column(name = "name")

private String name;

@Column(name = "mac")

private String mac;

@Column(name = "number")

private Integer number;

public Integer getNumber() {

return number;

}

public void setNumber(Integer number) {

this.number = number;

}

public String getMac() {

return mac;

}

public void setMac(String mac) {

this.mac = mac;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

public class UserMac {

private String name;

private String password;

private String mac;

public String getMac() {

return mac;

}

public void setMac(String mac) {

this.mac = mac;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

import com.lkc.socket.SocketService;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

@Component

public class MyCommandRunner implements CommandLineRunner {

@Override

public void run(String... args) {

Thread socketThread = new Thread(){

@Override

public void run() {

SocketService socketService = new SocketService(8880);

try{

socketService.SocketStart();

}catch (Exception e){

socketService = null;

System.out.println("socket service 启动失败！"+e);

}

}

};

socketThread.start();

}

}

import com.lkc.entity.Device;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.web.bind.annotation.PostMapping;

public interface DeviceRepository extends JpaRepository<Device,Integer> {

@Transactional

@Modifying

@Query("update Device as c set c.status = :status where c.mac= :mac")

Integer updateByMac(@Param("mac") String mac, @Param("status") Integer status);

Device findByMac(String mac);

@Query("select url from Device device where mac = :mac")

String findUrlByMac(@Param("mac") String mac);

@Transactional

@Modifying

@Query("update Device as c set c.url = :url where c.mac= :mac")

void updateUrl(@Param("url")String url, @Param("mac")String mac);

}

import com.lkc.entity.Email;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.web.bind.annotation.PostMapping;

public interface EmailRepository extends JpaRepository<Email,String> {

Email findByName(String name);

@Transactional

@Modifying

@Query("update Email set control=:control where name=:name")

int setEmailFunction(@Param("control")String control, @Param("name")String name);

Email findByNameAndAndEmail(String name,String email);

@Transactional

@Modifying

@Query("update Email set code=:code where name=:name")

int setCode(@Param("name")String name,@Param("code")String code);

Email findByNameAndCode(String name,String code);

}

import com.lkc.entity.Sensor;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface SensorRepository extends JpaRepository<Sensor,Integer> {

@Query(value="select \* from sensor\_info where mac = :mac and time between :t1 and :t2 limit 1",nativeQuery = true)

Sensor findByTimeAndMac(Integer t1,Integer t2,String mac);

@Query("select sensor from Sensor sensor where time = (select max(time) from Sensor where mac = :mac)")

Sensor findLastTimeByMac(@Param("mac") String mac);

@Query("select sensor from Sensor sensor where mac = :mac and time between :t1 and :t2")

List<Sensor> findAllByMacAndTime(@Param("mac") String mac, @Param("t1") Integer lasttime, @Param("t2") Integer nowtime);

}

import com.lkc.entity.Threshold;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.transaction.annotation.Transactional;

public interface ThresholdRepository extends JpaRepository<Threshold,String> {

Threshold findByName(String name);

@Transactional

@Modifying

@Query("update Threshold set temp\_min=:tmin,temp\_max=:tmax,hum\_max=:hmax,hum\_min=:hmin,amm\_max=:amax,amm\_min=:amin where name=:name")

int setThreshold(@Param("tmax")float tmax,@Param("tmin")float tmin,

@Param("hmax")float hmax,@Param("hmin")float hmin,

@Param("amax")float amax,@Param("amin")float amin,

@Param("name")String name);

}

import com.lkc.entity.Device;

import com.lkc.entity.UserCompDev;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

public interface UcompDRepository extends JpaRepository<UserCompDev,Integer> {

List<UserCompDev> findByName(String name);

List<UserCompDev> findByMac(String mac);

@Transactional

@Modifying

@Query("delete from UserCompDev where name=:name and mac=:mac")

int deleteByNameAndMac(@Param("name") String name, @Param("mac") String mac);

}

import com.lkc.entity.User;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.query.Param;

import org.springframework.transaction.annotation.Transactional;

public interface UserRepository extends JpaRepository<User,String> {

User findByName(String name);

@Query("select regtime from User user where name = :name")

Integer findRegtimeByName(@Param("name") String name);

@Transactional

@Modifying

@Query("update User as c set c.password = :password where c.name= :name")

int updatePasswordByName(@Param("name") String name, @Param("password") String password);

void delete(User user);

}

import com.lkc.entity.Device;

import com.lkc.entity.RetDevice;

import com.lkc.entity.UserCompDev;

import com.lkc.repository.DeviceRepository;

import com.lkc.repository.UcompDRepository;

import com.lkc.service.DeviceService;

import com.lkc.tool.TimeFormat;

import net.sf.json.JSONArray;

import net.sf.json.JSONObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.io.IOException;

import java.io.OutputStream;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;

@Service

public class DeviceServiceImpl implements DeviceService {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private DeviceRepository deviceRepository;

@Autowired

private UcompDRepository ucompDRepository;

@Override

public void save(Device device){

deviceRepository.save(device);

}

@Override

public Integer updateByMac(String mac, Integer status){

return deviceRepository.updateByMac(mac,status);

}

@Override

public Device findByMac(String mac) {

return deviceRepository.findByMac(mac);

}

@Override

public String findUrlByMac(String mac) {

return deviceRepository.findUrlByMac(mac);

}

@Override

public void updateUrl(String url, String mac) {

deviceRepository.updateUrl(url,mac);

}

@Override

public void videoControll(String mac,String controll) {

logger.trace("视频开关：" + controll);

String url = deviceRepository.findUrlByMac(mac);

logger.trace("mac:" + mac + "controll:" + controll );

try{

Socket socket = new Socket(url,8881);

OutputStream outputStream = socket.getOutputStream();

outputStream.write(controll.getBytes());

socket.close();

logger.debug("视频开关信息：" + controll + " 发送成功！");

} catch (IOException e){

logger.error("视频socket连接错误！");

e.printStackTrace();

}

}

@Override

public String rebootControll(String mac,String controll){

logger.trace("重启设备：" + mac);

String url = deviceRepository.findUrlByMac(mac);

try{

Socket socket = new Socket(url,8881);

OutputStream outputStream = socket.getOutputStream();

outputStream.write(controll.getBytes());

socket.close();

logger.debug("重启信号发送成功！");

return "success";

} catch (IOException e){

logger.error("重启socket连接错误！");

e.printStackTrace();

return "failed";

}

}

@Override

public String findDevice(String name) {

List<UserCompDev> list = ucompDRepository.findByName(name);

List<RetDevice> ret = new ArrayList<RetDevice>();

if (list.isEmpty()){

logger.warn(name + " 该用户没有设备！");

return "null";

}

RetDevice retDevice;

for(UserCompDev ucd : list){

String MAC = ucd.getMac();

Device dev = deviceRepository.findByMac(MAC);

retDevice = new RetDevice();

retDevice.setNumber(ucd.getNumber());

retDevice.setMac(MAC);

retDevice.setStatus(dev.getStatus());

if (null == dev.getLasttime()) {

retDevice.setLasttime("0");

}else{

retDevice.setLasttime(TimeFormat.secondToTime(String.valueOf(dev.getLasttime())));

}

ret.add(retDevice);

}

retDevice = null;

list = null;

JSONArray array = JSONArray.fromObject(ret);

String jsonstr = array.toString();

return jsonstr;

}

}

import com.lkc.entity.Email;

import com.lkc.repository.EmailRepository;

import com.lkc.repository.UserRepository;

import com.lkc.service.MailService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.stereotype.Service;

import java.util.Random;

@Service

public class MailServiceImpl implements MailService {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private JavaMailSender mailSender;

@Autowired

private EmailRepository emailRepository;

@Autowired

private UserRepository userRepository;

@Value("860990180@qq.com")

private String from;

@Override

public void sendMail(String title, String content, String email) {

SimpleMailMessage message = new SimpleMailMessage();

message.setFrom(from);

message.setSubject(title);

message.setTo(email);

message.setText(content);

mailSender.send(message);

}

@Override

public void addMail(String name, String email) {

Email mail = new Email();

mail.setName(name);

mail.setEmail(email);

emailRepository.save(mail);

}

@Override

public Email findByName(String name) {

Email mail = emailRepository.findByName(name);

return mail;

}

@Override

public String setCode(String name, String email) {

Email user = emailRepository.findByNameAndAndEmail(name,email);

if (user != null){

String code = "";

int tmp;

Random random = new Random();

for (int i=0; i<6; ++i) {

switch (random.nextInt(3)) {

case 0:

tmp = random.nextInt(26) + 65;

code = code + (char) tmp;

break;

case 1:

tmp = random.nextInt(26) + 97;

code = code + (char) tmp;

default:

tmp = random.nextInt(10);

code = code + String.valueOf(tmp);

}

}

int i = emailRepository.setCode(name,code);

logger.debug(name + "的验证码生成成功！");

final String Code = code;

code = null;

new Thread(){

@Override

public void run() {

String title = "沃基环境监测系统密码找回";

String content = "验证码为：" + Code;

sendMail(title,content,email);

}

}.start();

return "success";

}

return "error";

}

@Override

public String checkcodeAndChg(String name, String code, String password) {

Email email = emailRepository.findByNameAndCode(name,code);

if (email != null){

userRepository.updatePasswordByName(name,password);

logger.info(name + "成功找回密码！");

return "success";

}

return "error";

}

@Override

public String setEmailFunction(String name, String control) {

int i = emailRepository.setEmailFunction(control,name);

if ( i > 0){

return "success";

}else{

return "failed";

}

}

}

import com.lkc.entity.Sensor;

import com.lkc.repository.SensorRepository;

import com.lkc.service.SensorService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

public class SensorServiceImpl implements SensorService {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private SensorRepository sensorRepository;

@Override

public void save(Sensor sensor){

sensorRepository.save(sensor);

}

@Override

public List<Sensor> findByCustom(Integer second,Integer start,String mac) {

List<Sensor> retList = new ArrayList<Sensor>();

int historytime = start;

int count = second;

Sensor sensor;

for(int i=0; i<24; i++){

sensor = sensorRepository.findByTimeAndMac(historytime,historytime+count,mac);

if (null != sensor) {

retList.add(sensor);

}

historytime = historytime + count;

}

sensor = null;

return retList;

}

@Override

public List<Sensor> findByDefault(Integer needtime,Integer nowtime,String mac){

logger.trace("查询过去" + needtime + "小时的数据");

List<Sensor> retList = new ArrayList<Sensor>();

int t1,t2;

int count = 900;

if (needtime == 6){

count = 900;

}else if (needtime == 12){

count = 1800;

}else if (needtime == 24){

count = 3600;

}

int historytime = nowtime - needtime\*3600;

int historynext = historytime + count - historytime % count;

int nownext = nowtime + count - nowtime%count - 1;

Sensor sensor;

for(int tmp=historynext; tmp<nownext; tmp=tmp+count){

t1 = tmp;

t2 = tmp + count -1;

sensor = sensorRepository.findByTimeAndMac(t1,t2,mac);

if (null != sensor) {

retList.add(sensor);

}

}

sensor = null;

return retList;

}

@Override

public Sensor findLastTimeByMac(String mac){

return sensorRepository.findLastTimeByMac(mac);

}

@Override

public List<Sensor> findAllByMacAndTime(String mac, Integer lasttime) {

logger.trace("查询" + "mac号为 "+mac + " 的过去3小时的数据");

return sensorRepository.findAllByMacAndTime(mac,lasttime,lasttime+10800);

}

}

import com.lkc.entity.Threshold;

import com.lkc.repository.ThresholdRepository;

import com.lkc.service.ThresholdService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class ThresholdServiceImpl implements ThresholdService {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private ThresholdRepository thresholdRepository;

@Override

public Threshold findByName(String name) {

logger.trace("用户："+name+" 获取了阀值");

Threshold threshold = thresholdRepository.findByName(name);

return threshold;

}

@Override

public String setThreshold(float tmax, float tmin, float hmax, float hmin, float amax, float amin, String name) {

logger.trace("用户："+name+" 设置了阀值");

int i = thresholdRepository.setThreshold(tmax,tmin,hmax,hmin,amax,amin,name);

if ( i > 0){

return "success";

}else{

return "error";

}

}

}

import com.lkc.entity.Device;

import com.lkc.entity.User;

import com.lkc.entity.UserCompDev;

import com.lkc.entity.UserMac;

import com.lkc.repository.UcompDRepository;

import com.lkc.repository.UserRepository;

import com.lkc.service.DeviceService;

import com.lkc.service.UcompDService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class UcompDServiceImpl implements UcompDService {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private UcompDRepository ucompDRepository;

@Autowired

private UserRepository userRepository;

@Autowired

private DeviceService deviceService;

@Override

public String login(UserMac usermac) {

String name = usermac.getName();

String password = usermac.getPassword();

String mac = usermac.getMac();

User user = new User();

user = userRepository.findByName(name);

boolean flg = true;

if (user == null){

logger.trace("开发板登录： " + name + " 用户不存在！");

return "name\_error";

} else if (user.getPassword().equals(password)){

List<UserCompDev> ucdlist = ucompDRepository.findByName(name);

for(UserCompDev ucd : ucdlist){

if (ucd.getMac().equals(mac)){

flg = false;

break;

}

}

if (flg){

UserCompDev userCompDev = new UserCompDev();

userCompDev.setName(name);

userCompDev.setMac(mac);

ucompDRepository.save(userCompDev);

userCompDev = null;

Device device = deviceService.findByMac(mac);

if (device == null){

Device device1 = new Device();

device1.setMac(mac);

device1.setStatus(0);

deviceService.save(device1);

device1 = null;

logger.trace("开发板登录： " + name + " mac号：" + mac + " 设备添加成功！");

}

}

user = null;

logger.trace("开发板登录： " + name + " 登录成功！");

return "success";

}

user = null;

logger.trace("开发板登录： " + name + " 密码错误！");

return "password\_error";

}

@Override

public List<UserCompDev> findByName(String name) {

return ucompDRepository.findByName(name);

}

@Override

public List<UserCompDev> findByMac(String mac) {

return ucompDRepository.findByMac(mac);

}

@Override

public String deleteByNameAndMac(String name, String mac) {

int row = ucompDRepository.deleteByNameAndMac(name,mac);

if (row > 0){

logger.trace("用户"+name+" 移除设备"+mac);

return "success";

}else{

return "error";

}

}

}

import com.lkc.entity.User;

import com.lkc.repository.UserRepository;

import com.lkc.service.MailService;

import com.lkc.service.UserService;

import com.lkc.tool.TimeFormat;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.Date;

@Service

public class UserServiceImpl implements UserService {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private UserRepository userRepository;

@Autowired

private MailService mailService;

@Override

public String addUser(String name,String password, String email){

User user = userRepository.findByName(name);

if (user != null){

logger.trace("注册： " + name + " 已存在！");

return "Has";

}else{

User us = new User();

long currentTime = new Date().getTime();

String timestr = String.valueOf(currentTime/1000);

Integer time = Integer.valueOf(timestr);

us.setName(name);

us.setPassword(password);

us.setRegtime(time);

userRepository.save(us);

us = null;

mailService.addMail(name,email);

mailService.setEmailFunction(name,"off");

logger.trace("注册： " + name + " 注册成功！");

Thread emailThread = new Thread(){

@Override

public void run() {

String title = "注册了沃基环境监测系统账号";

String setText = "成功注册！";

mailService.sendMail(title,setText,email);

logger.trace("给"+email+"发送注册邮件成功！");

}

};

emailThread.start();

return "success";

}

}

@Override

public String login(String name,String password){

User user = userRepository.findByName(name);

if (user == null){

logger.trace("登录： " + name + " 用户名不存在！");

user = null;

return "name\_error"; //用户名不存在

}else if (user.getPassword().equals(password)){

logger.trace("登录： " + name + " 登录成功！");

user = null;

return "success"; //密码符合

}else{

logger.trace("登录：" + name + " 密码错误！");

user = null;

return "password\_error"; //密码错误

}

}

@Override

public String findRegtime(String name) {

Integer time = userRepository.findRegtimeByName(name);

return TimeFormat.secondToTime(String.valueOf(time));

}

@Override

public String changePassword(String name, String newpassword){

int i = userRepository.updatePasswordByName(name,newpassword);

if (i != 0){

logger.trace("修改密码： " + name + " 修改密码成功！");

return "success";

}else{

logger.trace("修改密码： " + name + " 修改密码失败！");

return "update error";

}

}

}

import com.lkc.entity.Device;

public interface DeviceService {

void save(Device device);

Integer updateByMac(String mac, Integer status);

Device findByMac(String mac);

String findUrlByMac(String mac);

void updateUrl(String url,String mac);

void videoControll(String mac,String controll);

String findDevice(String name);

String rebootControll(String mac,String controll);

}

import com.lkc.entity.Email;

public interface MailService {

void sendMail(String title,String content,String email);

void addMail(String name,String email);

Email findByName(String name);

String setCode(String name, String email);

String checkcodeAndChg(String name, String code, String password);

String setEmailFunction(String name,String control);

}

import com.lkc.entity.Sensor;

import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

public interface SensorService {

void save(Sensor sensor);

List<Sensor> findByCustom(Integer second,Integer start,String mac);

List<Sensor> findByDefault(Integer needtime,Integer nowtime,String mac);

Sensor findLastTimeByMac(String mac);

List<Sensor> findAllByMacAndTime(String mac,Integer lasttime);

}

import com.lkc.entity.Threshold;

import org.springframework.data.repository.query.Param;

public interface ThresholdService {

Threshold findByName(String name);

String setThreshold(float tmax, float tmin, float hmax, float hmin, float amax, float amin, String name);

}

import com.lkc.entity.UserCompDev;

import com.lkc.entity.UserMac;

import java.util.List;

public interface UcompDService {

String login(UserMac usermac);

List<UserCompDev> findByName(String name);

List<UserCompDev> findByMac(String mac);

String deleteByNameAndMac(String name,String mac);

}

public interface UserService {

String addUser(String name,String password,String email);

String login(String name,String password);

String findRegtime(String name);

String changePassword(String name, String password);

}

import com.lkc.entity.Device;

import com.lkc.entity.Sensor;

import com.lkc.entity.Threshold;

import com.lkc.entity.UserCompDev;

import com.lkc.service.\*;

import com.lkc.tool.SendWarnMail;

import com.lkc.tool.TimeFormat;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class SocketService {

Logger logger = LoggerFactory.getLogger(getClass());

private static Integer port;

public SocketService (Integer port){

this.port = port;

}

public void SocketStart () throws IOException{

ServerSocket serverSocket = new ServerSocket(port);

logger.debug("服务器信息: "+serverSocket.getInetAddress()+" 端口: "+serverSocket.getLocalPort());

while (true){

Socket socket = serverSocket.accept();

ReadThread readThread = new ReadThread(socket);

readThread.start();

try{

readThread.join();

logger.warn("接收线程结束");

}catch (Exception e){

e.printStackTrace();

}

}

}

@Component

public static class ReadThread extends Thread{

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private SensorService sensorService;

@Autowired

private DeviceService deviceService;

private static ReadThread readThread;

@PostConstruct

public void init(){

readThread = this;

readThread.sensorService = this.sensorService;

readThread.deviceService = this.deviceService;

}

private Socket socket;

private int heartcount;

private String MAC;

private boolean heartflg = false;

private boolean flg = true;

private static final int RUNING = 1;

private static final int CLOSE = 0;

ReadThread(){}

ReadThread(Socket socket){

this.socket = socket;

}

Thread heartThread = new Thread(){

@Override

public void run() {

while(!isInterrupted()){

while(!heartflg){

try{

sleep(10);

}catch (InterruptedException e){

e.printStackTrace();

}

}

heartcount = 0;

while(heartflg && heartcount < 50){

try{

sleep(100);

heartcount++;

}catch (InterruptedException e){

e.printStackTrace();

}

}

if (heartcount >= 50){

logger.warn("socket连接超时！连接断开！");

try{

socket.close();

}catch (Exception e){

e.printStackTrace();

}

readThread.deviceService.updateByMac(MAC,CLOSE);

interrupt();

}else {

readThread.deviceService.updateByMac(MAC, RUNING);

}

}

}

};

@Override

public void run() {

super.run();

heartThread.start();

logger.info("客户端信息: "+socket.getInetAddress() + " 端口: "+socket.getLocalPort());

int warncount = 10;

try{

socket.setSoTimeout(5000);

InputStream inputStream = socket.getInputStream();

OutputStream outputStream = socket.getOutputStream();

byte[] bytes;

Sensor sensor;

String[] strarray;

String str;

while(!isInterrupted()){

bytes = new byte[1024];

heartcount = 0;

heartflg = true;

if (inputStream.read(bytes,0,1024) != -1){

heartflg = false;

str = new String(bytes);

str = str.trim();

strarray = str.split(" ");

if (str.contains("heartbeat")){

logger.trace("收到 " + socket.getInetAddress() + " 发来的心跳包！");

outputStream.write("heartbeat".getBytes());

continue;

} else if (strarray.length != 5){

continue;

} else if (strarray[4].length() > 17){

continue;

} else if (strarray[0].length() > 17){

continue;

}

if (flg){

MAC = strarray[0];

Device device = readThread.deviceService.findByMac(MAC);

if (device != null){

InetAddress address = socket.getInetAddress();

String url = String.valueOf(address).replace("/","");

readThread.deviceService.updateUrl(url,MAC);

readThread.deviceService.updateByMac(MAC, RUNING);

}else{

InetAddress address = socket.getInetAddress();

String url = String.valueOf(address);

Device device1 = new Device();

device1.setMac(MAC);

device1.setStatus(RUNING);

device1.setLasttime(Integer.valueOf(strarray[1]));

device1.setUrl(url.replace("/",""));

readThread.deviceService.save(device1);

}

flg = false;

}

sensor = new Sensor();

sensor.setMac(MAC);

sensor.setTime(Integer.valueOf(strarray[1]));

sensor.setTemperature(Float.valueOf(strarray[2]));

sensor.setHumidity(Float.valueOf(strarray[3]));

sensor.setAmmonia(Float.valueOf(strarray[4]));

readThread.sensorService.save(sensor);

warncount--;

if (0 == warncount){

InetAddress addr = InetAddress.getLocalHost();

String ip=addr.getHostAddress().toString();

new SendWarnMail(sensor,ip).start();

warncount = 10;

}

}else{

break;

}

}

}catch (Exception e){

e.printStackTrace();

readThread.deviceService.updateByMac(MAC,CLOSE);

logger.error("与" + socket.getInetAddress() + " 连接异常断开!");

heartThread.interrupt();

this.heartThread = null;

}

}

}

}

import com.lkc.entity.Email;

import com.lkc.entity.Sensor;

import com.lkc.entity.Threshold;

import com.lkc.entity.UserCompDev;

import com.lkc.service.MailService;

import com.lkc.service.ThresholdService;

import com.lkc.service.UcompDService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import java.util.List;

@Component

public class SendWarnMail extends Thread {

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private UcompDService ucompDService;

@Autowired

private ThresholdService thresholdService;

@Autowired

private MailService mailService;

private Sensor sensor;

private String ip;

private static SendWarnMail sendWarnMail;

@PostConstruct

public void init(){

sendWarnMail = this;

sendWarnMail.mailService = this.mailService;

sendWarnMail.ucompDService = this.ucompDService;

sendWarnMail.thresholdService = this.thresholdService;

}

SendWarnMail(){};

public SendWarnMail(Sensor sensor,String ip){

this.sensor = sensor;

this.ip = ip;

}

@Override

public void run() {

int time = sensor.getTime();

float temp = sensor.getTemperature();

float amm = sensor.getAmmonia();

float hum = sensor.getHumidity();

String mac = sensor.getMac();

String title = "预警邮件";

String email;

String content="";

boolean email\_flg;

String name;

List<UserCompDev> ucdList = sendWarnMail.ucompDService.findByMac(mac);

if (ucdList != null){

for(UserCompDev ucd : ucdList){

email\_flg = false;

name = ucd.getName();

Email usermail = sendWarnMail.mailService.findByName(name);

if (usermail.getControl().equals("on")){

Threshold threshold = sendWarnMail.thresholdService.findByName(name);

email = usermail.getEmail();

if(threshold.getAmm\_max() < amm){

content = content + " 氨气值为"+ amm + " 超过上限！\n";

email\_flg = true;

}else if(threshold.getAmm\_min() > amm){

content = content + " 氨气值为"+ amm + " 低于下限！\n";

email\_flg = true;

}

if(threshold.getHum\_max() < hum){

content = content + " 湿度值为"+ hum + " 超过上限！\n";

email\_flg = true;

}else if(threshold.getHum\_min() > hum){

content = content + " 湿度值为"+ hum + " 低于下限！\n";

email\_flg = true;

}

if(threshold.getTemp\_max() < temp){

content = content + " 温度值为"+ temp + " 超过上限！\n";

email\_flg = true;

}else if(threshold.getTemp\_min() > temp){

content = content + " 温度值为"+ temp + " 低于下限！\n";

email\_flg = true;

}

if (email\_flg){

content = "您好，"+ name +":\n"+

" 您的"+ucd.getNumber()+"号设备"+ mac +" 于 "+TimeFormat.secondToTime(String.valueOf(time)) +" 发出报警：\n" +

content + "详情请查看该设备历史数据！\n" +

"请登录http://"+ip.replace("/","")+":8080/WLZY/login\_web/login.html";

sendWarnMail.mailService.sendMail(title,content,email);

logger.info("预警邮件发送成功！ to："+email);

}

content = null;

}

}

}

}

}

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class TimeFormat {

public static String secondToTime(String time){

String dateStr="1970-1-1 08:00:00";

SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

if (time.equals("0")) {

return "";

}

Date miDate;

String returnstr="";

try {

miDate = sdf.parse(dateStr);

Object t1=miDate.getTime();

long h1=Long.parseLong(time)\*1000+Long.parseLong(t1.toString());

returnstr=sdf.format(h1);

} catch (ParseException e) {

e.printStackTrace();

}

return returnstr;

}

}

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.socket.server.standard.ServerEndpointExporter;

@Configuration

public class WebSocketConfig {

@Bean

public ServerEndpointExporter serverEndpointExporter() {

return new ServerEndpointExporter();

}

}

import com.lkc.entity.Device;

import com.lkc.entity.Sensor;

import com.lkc.service.DeviceService;

import com.lkc.service.SensorService;

import net.sf.json.JSONObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import javax.websocket.\*;

import javax.websocket.server.ServerEndpoint;

import java.io.IOException;

import java.util.concurrent.CopyOnWriteArraySet;

@ServerEndpoint("/websocket")

@Component

public class WebSocketService {

Logger logger = LoggerFactory.getLogger(getClass());

private static CopyOnWriteArraySet<WebSocketService> webSocketSet = new CopyOnWriteArraySet<WebSocketService>();

private static int onlineCount = 0;

private Session session;

SendThread sendThread;

@OnOpen

public void OnOpen(Session session){

this.session = session;

webSocketSet.add(this);

addOnlineCount();

logger.info("websocket: " + " sessionid=" + session.getId() + " 连接成功！人数为："+getOnlineCount());

}

@OnError

public void OnError(Session session, Throwable error){

webSocketSet.remove(this);

logger.warn("websocket: " + session.getId() + " 连接错误！");

error.printStackTrace();

}

@OnClose

public void OnClose(Session session){

webSocketSet.remove(this);

subOnlineCount();

logger.info("websocket: " + session.getId() + " 连接关闭！剩余人数为："+getOnlineCount());

sendThread.flg = false;

}

@OnMessage

public synchronized void OnMessage(String msg, Session session){

String[] array = msg.split(" ");

if (array[0].equals("mac")){

sendThread = new SendThread();

sendThread.session = session;

sendThread.mac = array[1];

sendThread.start();

}else if (msg.equals("close")){

if(sendThread != null){

sendThread.flg = false;

try{

sendThread.join();

}catch (Exception e){

e.printStackTrace();

logger.error("websocket: " + session.getId() + " 发送线程终止失败！");

}

}

}

}

public static synchronized void addOnlineCount(){

WebSocketService.onlineCount++;

}

public static synchronized void subOnlineCount(){ WebSocketService.onlineCount--; }

public static synchronized int getOnlineCount(){

return onlineCount;

}

@Component

public static class SendThread extends Thread{

Logger logger = LoggerFactory.getLogger(getClass());

@Autowired

private SensorService sensorService;

@Autowired

private DeviceService deviceService;

private static SendThread sendThread;

@PostConstruct

public void init(){

sendThread = this;

sendThread.sensorService = this.sensorService;

sendThread.deviceService = this.deviceService;

}

private Session session;

private String mac;

private boolean flg = true;

SendThread(){}

@Override

public synchronized void run() {

int devstatus;

boolean statusflg = true;

while(flg){

Sensor sensor = sendThread.sensorService.findLastTimeByMac(mac);

Device device = sendThread.deviceService.findByMac(mac);

if (null == device){

statusflg = false;

} else {

devstatus = device.getStatus();

if (devstatus == 0){

statusflg = false;

}else{

statusflg = true;

}

}

if (statusflg){

JSONObject object = JSONObject.fromObject(sensor);

String jsonstr = object.toString();

sensor = null;

device = null;

try{

session.getBasicRemote().sendText(jsonstr);

sleep(500);

}catch (Exception e){

e.printStackTrace();

logger.error("实时发送失败！");

}

}else{

sensor = null;

device = null;

try{

session.getBasicRemote().sendText("false");

logger.warn("设备：" + mac + " 已断开！");

}catch (IOException e){

e.printStackTrace();

}

}

}

logger.debug("发送结束");

}

}

}

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

@SpringBootApplication

public class XmApplication extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(this.getClass());

}

public static void main(String[] args) {

SpringApplication.run(XmApplication.class, args);

}

}