示教器二次开发文档

一、开发环境

1.1 开发环境简介

1.1.1 系统：ubuntu14.04 （ubuntu-14.04.5-desktop-i386.iso）

1.1.2 开发工具 IDE：QtCreator + Qt5.2.1 （使用 sudo apt-get install qtcreator 命令安装）

1.1.3 交叉编译工具配置详见 1.2.1 开发环境配置及使用方法

1.2 开发环境配置及使用方法

1.2.1 交叉编译工具配置

交叉编译工具目录是 nextpLib 文件夹中的 T30-gcc 文件夹中

配置交叉编译

第一步：

将文件 gcc.tar.gz 和 qtenv.tar.gz 解压到系统的/opt/目录下

第二步：

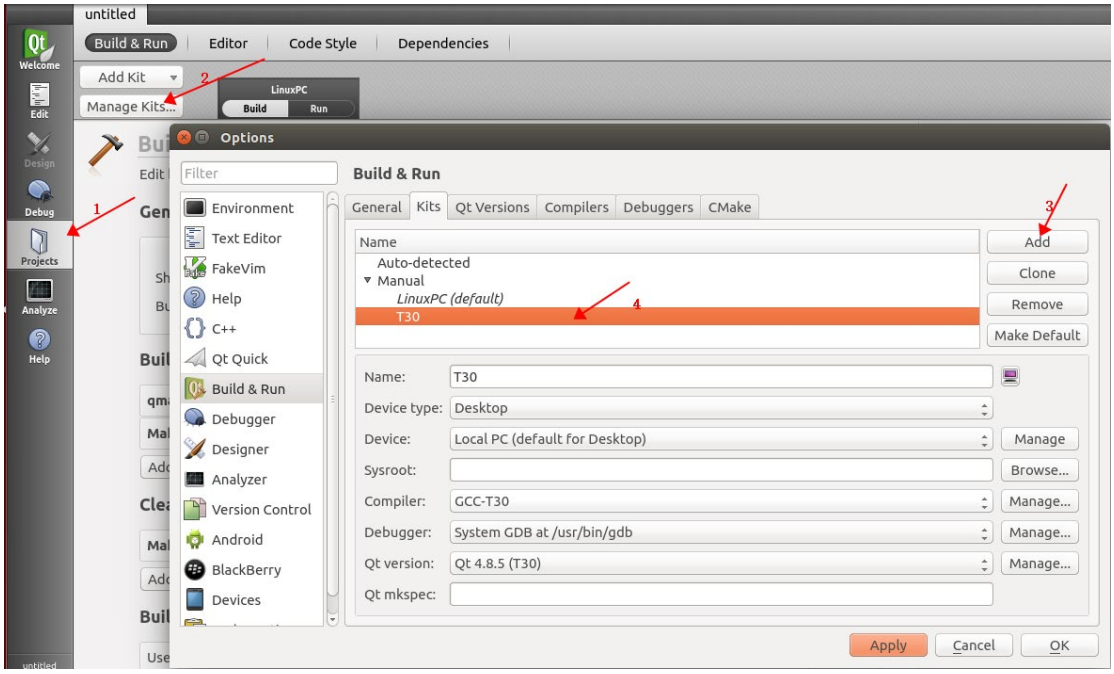
将 tslib.tar.gz 文件解压到 ubuntu 系统的/usr/local 目录下

第三步：

在 ubuntu14.04 终端中运行 qtcreator 新建一个 Qt 界面程序

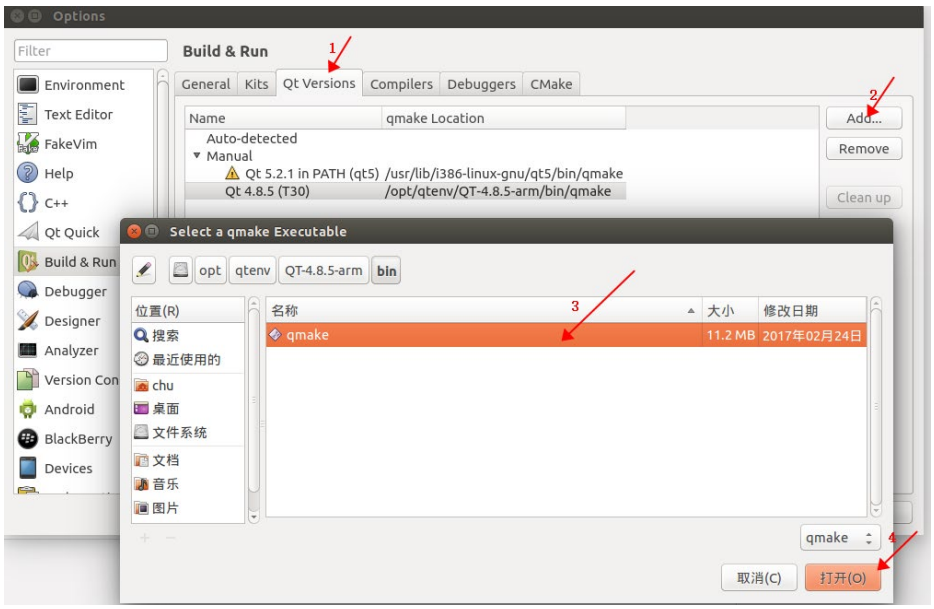
在 qtcreator 中左侧点击 Projects 按钮---->Manage Kit...

在 Options 中单击右上角的按钮 Add 添加两个编译套件 如下图



其中 T30 交叉编译 点击 Qt Versions 栏----->Add 按钮 （添加 qmake)

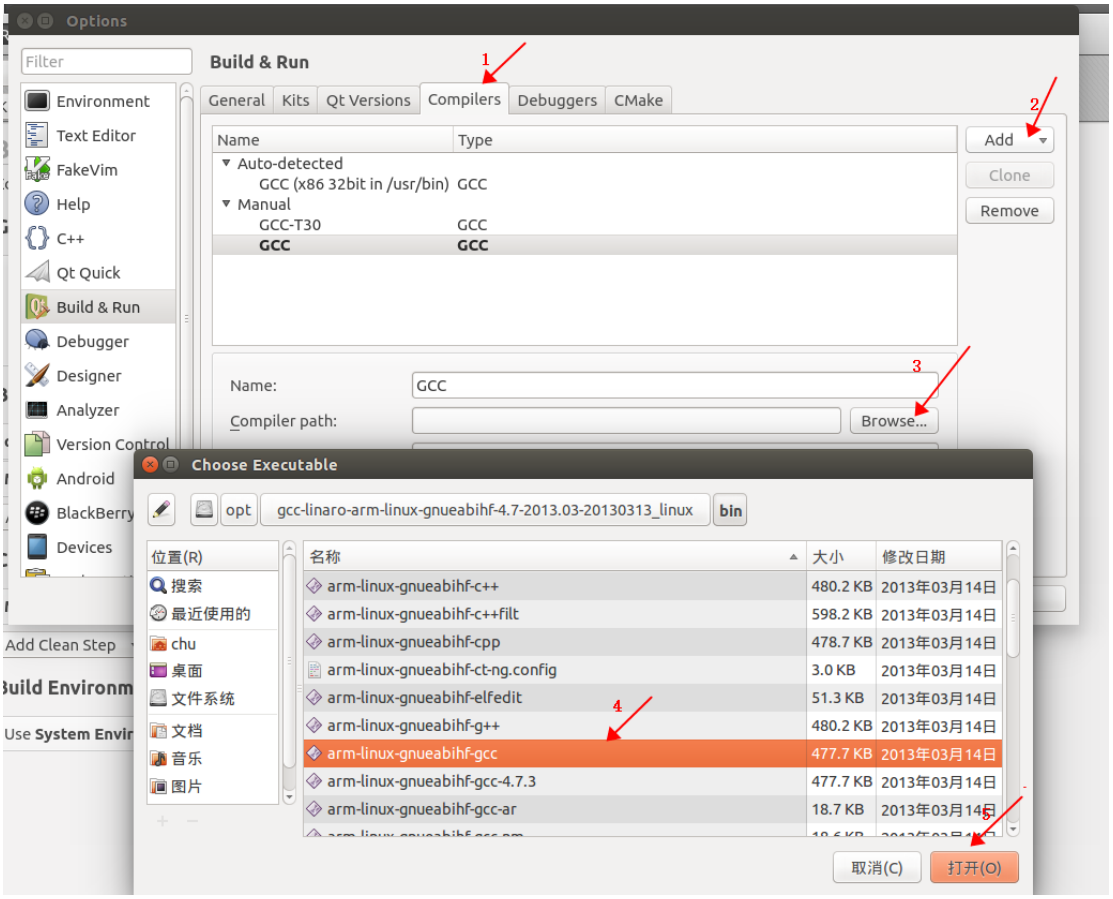
在弹出的对话框中选择 qmake(路径：/opt/qtenv/QT-4.8.5-arm/bin/qmake)



修改工具链名称为 GCC-T30　点击 Browse...按钮添加交叉编译工具链

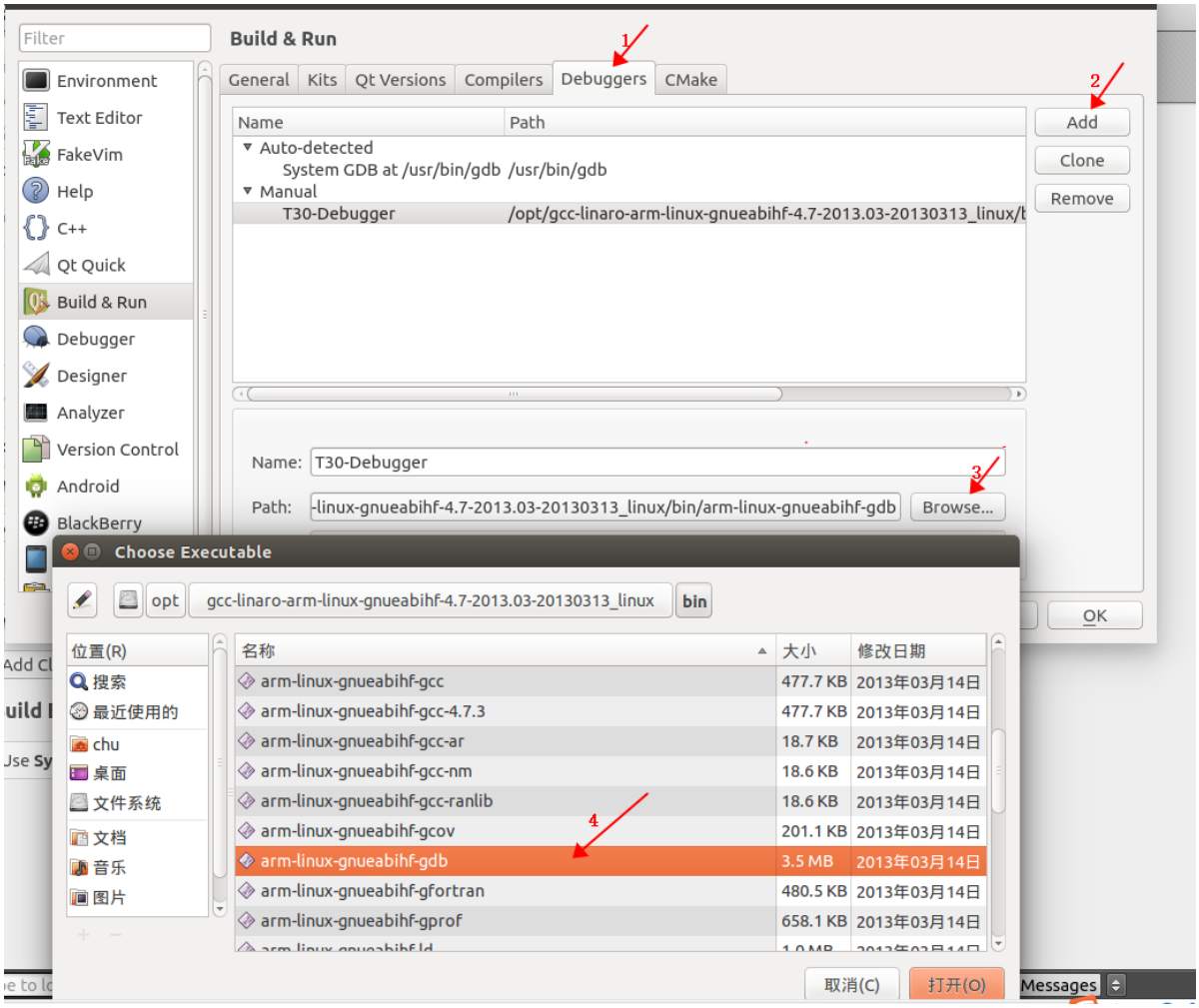
工　　具　　链　　路　　径

（/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313_linux/bin/armlinux-gnueabihf-gcc）



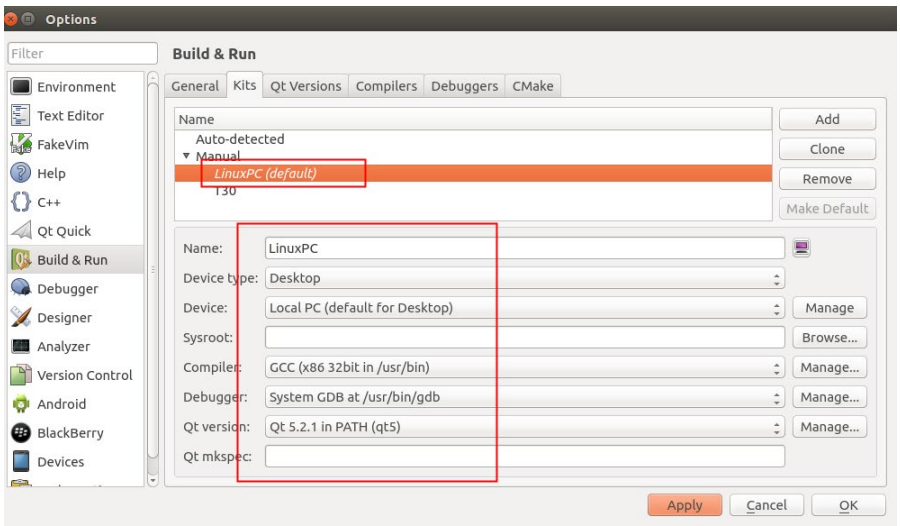
添加 Debugger:点击 Debuggers 按钮----->Add　修改 debugger 名称为 T30-Debugger

添加 debugger：点击 Browse... 按钮　debugger 路径（/opt/gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313_linux/bin/arm-linux-gnueabihf-gdb）

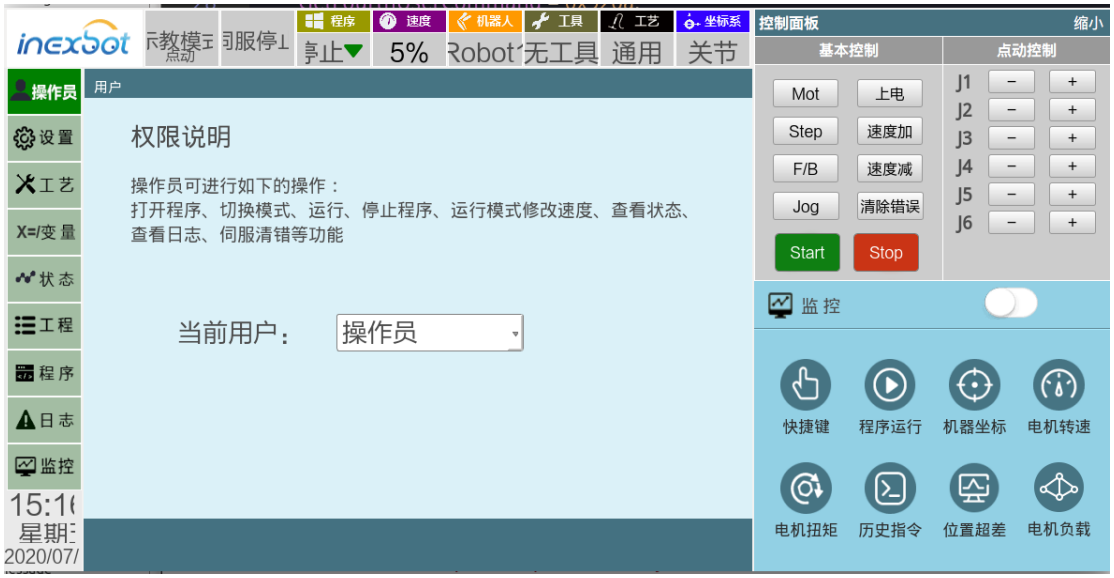


以上配置是 T30 示教器程序编译工具，该编译工具编译的程序仅适用于纳博特公司的 T30 示教器

如果仅使用及测试 Demo 可以使用 QtCreator 自带的编译器



使用 LinuxPC 编译的程序可以在 Ubuntu14.04 系统中运行，下图为运行在 ubuntu 系统的 Demo 程序



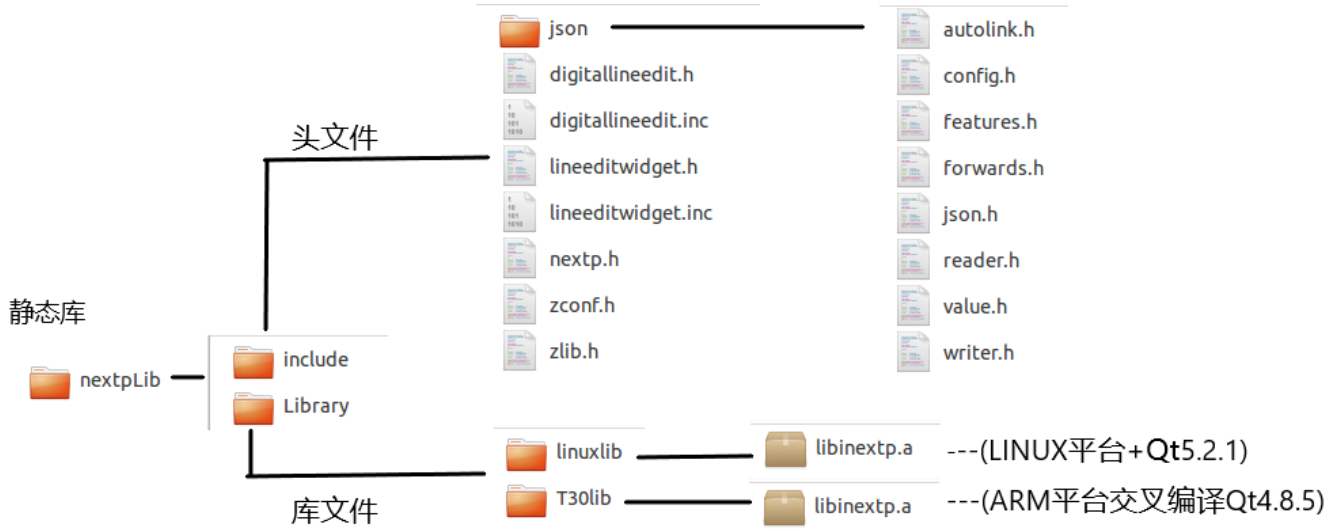
二、示教器软件二次开发静态库

2.1 二次开发静态库简介

2.1.1 功能介绍：

提供完整的示教器功能，支持添加用户自定义界面，支持特定字段的通信与控制系统通信。

2.1.2 静态库目录结构：



nextpLib 为总文件夹

Include 文件夹中为头文件

Library 文件夹中为静态库文件包括 linux 平台和 ARM 平台的库文件

nrc 文件夹中的 nrc.out 是与当前 demo 程序配套的控制器程序

其中使用（ARM 平台交叉编译的程序只适用于纳博特公司的 T30 示教器使用）

2.1.3 静态库结构说明

1.nextp.h 头文件接口说明：

//创建 Nextp 类对象

static QPointer<Nextp> getInstance();

//获取系统字体

QString getSystemFont();

```
//用户自定义窗体传输到主程序
void setWidgetParentLocation(QPointer <QWidget> widget);
//向控制器发送消息
  void sendMessage(const quint16 &command,const QByteArray &data);
//通知示教程序自定义窗体已经打开
widgetShowFinish()
//接收控制器消息信号
void signal_receiveMessage(const quint16 &command,const QByteArray
&data);
//打开自定义窗体信号
void signal_openWidget();
//关闭自定义窗体信号
void signal_closeWidget();
//隐藏工艺主界面中除了自定义按钮的其他按钮
void hideTechnologyToolbuttons();


//静态库支持与控制器通信命令字
        enum CommandList
        {
                SetFirstUserParaCommand = 0x9200,
                GetFirstUserCommand = 0x9201,
                ReceivedFirstUserCommand = 0x9202,
```

```
        SetSecondUserParaCommand = 0x9203,
        GetSecondUserCommand = 0x9204,
        ReceivedSecondUserCommand = 0x9205,

        SetThirdUserParaCommand = 0x9206,
        GetThirdUserCommand = 0x9207,
        ReceivedThirdUserCommand = 0x92,
        ReceivedThirdUserCommand = 0x9208,

        SetFourthUserParaCommand = 0x9209,
        GetFourthUserCommand = 0x920a,
        ReceivedFourthUserCommand = 0x920b,

        SetFifthUserParaCommand = 0x920c,
        GetFifthUserCommand = 0x920d,
        ReceivedFifthUserCommand = 0x920e,
    };
```

2. json/json.h 头文件提供 JSON 数据格式的组装和解析

2.1 组装 json 数据示例:

```
Json::Value root;
Json::FastWriter wt;
```

```
root["robot"] =1;

root["booldata"] =true;

root["data"] = 1.1;

root["name"] =”nihao”;
```

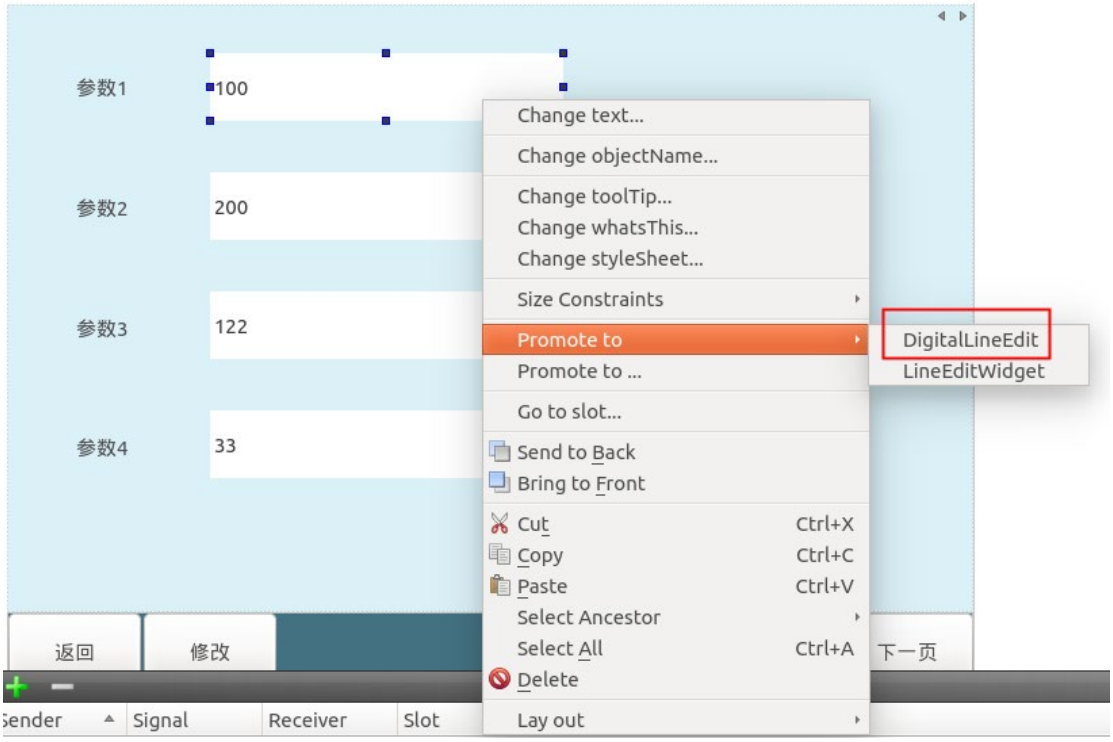
2.2 解析 Json 数据示例

```
QByteArray jsonData    //控制器发送的数据

Json::Value root;

Json::Reader reader;

QString jsonData = param.data();

if(reader.parse(jsonData.toStdString(), root))

{

int robot = root["robot"].asInt();

bool booldata=    root["booldata"].asBool();

Int data= root["data"].asDouble();

Std::string name =    root["name "].asString();

}
```

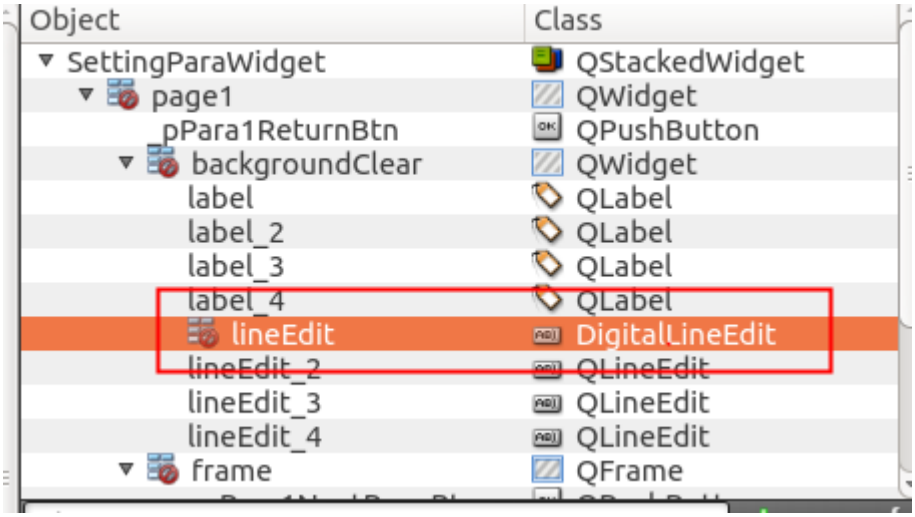
3. digitallineedit.h 提供数字输入框

支持将 QLineEdit 控件提升为数字输入框

提升方法：右键选择一个 QLineEdit 控件--->Promote to --->DigitalLineEdit



右侧树形结构中可以看到该控件 Classs 属性变为 DigitLineEdit

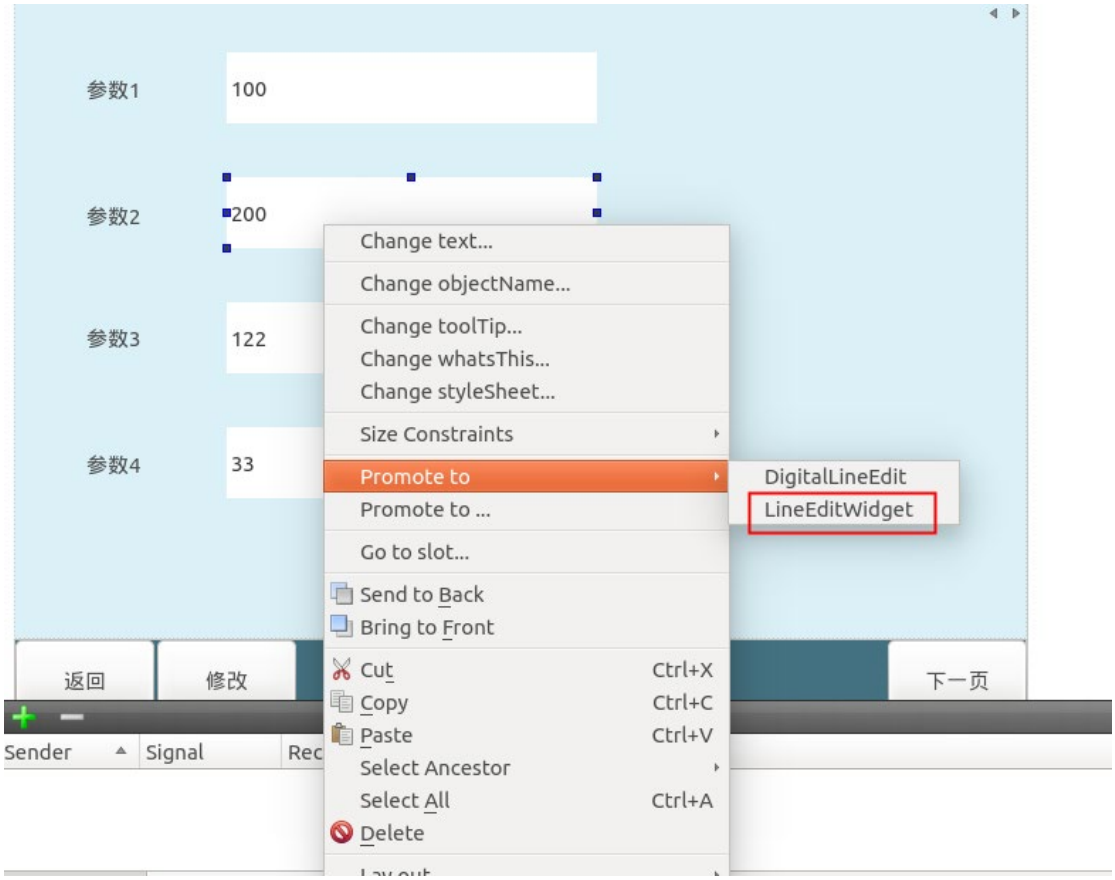


程序运行后控件效果，单击控件会弹出数字键盘：

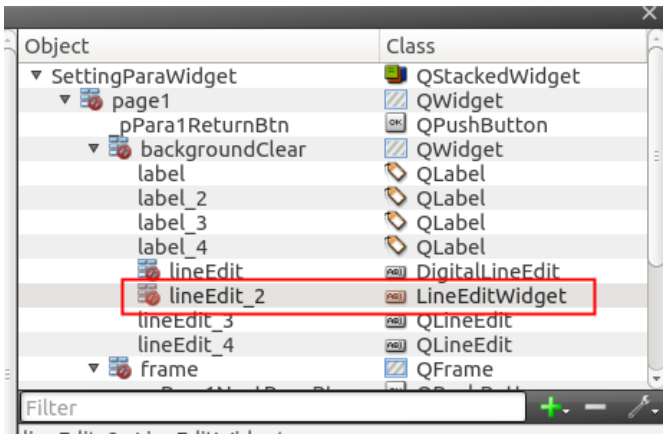
4. lineeditwidget.h 提供数字和字符输入框

支持将 QLineEdit 控件提升为数字与字符输入框

提升方法：右键选择一个 QLineEdit 控件--->Promote to --->lineEditWidget



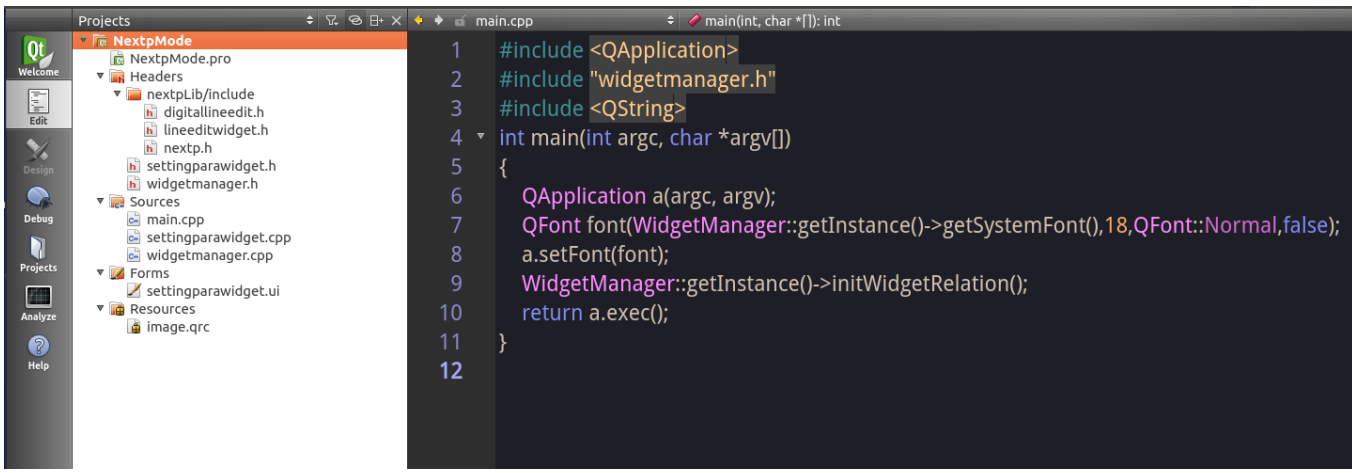
右侧树形结构中可以看到该控件 Classs 属性变为 lineEditWidget



程序运行后控件效果，单击控件会弹出数字与字母键盘：

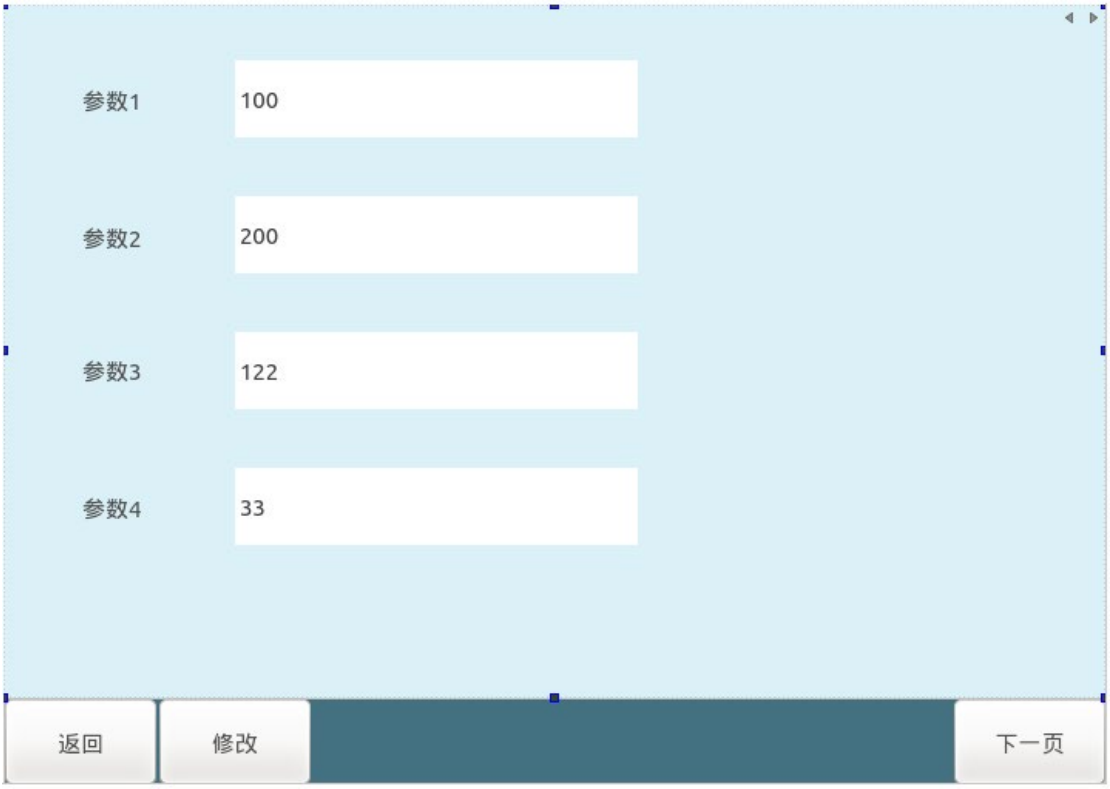
2.1.4 Demo 说明

1. Demo 结构图 demo 文件夹名称：NextpMode



2. Demo 文件类说明

2.1 settingparawidget.h settingparawidget.cpp settingparawidget.ui 三个文件为用户自定义窗体



2.2　widgetmanager.h widgetmanager.cpp 为管理类连接用户自定义窗体和静态库

2.2 静态库文件夹 nextplib 需要放置在 demod 的 NextpMode 文件夹下

2.3 运行 Demo （使用 QtCreator 直接打开 NextpMode 文件夹下的 NextpMode.pro 文件 ）

运 Demo 程序点击【操作员】> 选择管理员>输入密码 123456 登录

点击左侧【工艺】按钮> 用户 进入自定义窗体

点击修改按钮 可以修改参数 点击保存将发送参数到控制器

QtCreator 控制台会打印发送到控制器的数据

```
70    else
71    {
72       Json::Value root;
73       Json::FastWriter wt;
74       root["para1"] =ui->lineEdit->text().toInt();
75       root["para2"] =ui->lineEdit_2->text().toStdString();
76       root["para3"] =ui->lineEdit_4->text().toInt();
77       root["para4"] =ui->lineEdit_3->text().toStdString();
78
79       root["robot"] =1;
80       root["booldata"] =true;
81       root["data"] = 1.1;
82       root["name"] ="nihao";
83       Nextp::getInstance()->sendMessage(Nextp::GetFirstUserCommand,QByteArray().append(QString::fromStdString(wt.write(root))));
84
85
86       ui->_pParamodifyBtn->setText("修改");
```

```
Find:
Replace with:
Replace  Replace & Find  Replace All  Advanced...
Application Output
NextpMode
删除用户窗体
创建 用户自定义页面
MaskWidget::show
MaskWidget::hide
"发送指令-->控制器：[0x9201],{"booldata":true,"data":1.10,"name":"nihao","para1":100,"para2":"200","para3":122,"para4":"33","robot":1}
"
```

```
47
48  void WidgetManager::slot_receiveMessage(const quint16 &command,const QByteArray &data)
49  {
50     QString datastr = QString("接收控制器指令-->>：[0x")+QString::number(command,16).toUpper()+QString("
51     qDebug()<<datastr;
52  }
53
```

```
Application Output
NextpMode
创建 用户自定义页面
"发送指令-->控制器：[0x9200],{"booldata":true,"data":1.10,"name":"nihao","para1":100,"para2":"200","para3":
122,"para4":"33","robot":1}
"
"接收控制器指令-->>：[0x9201],{"booldata":true,"data":1.10,"name":"nihao","para1":100,"para2":"200","para3":
122,"para4":"33","robot":1}
"
```

如果调用函数 void hideTechnologyToolbuttons();会隐藏除工艺在主界面上自定义按钮以外的其他工艺按钮