使用环境

> 主机：Ubuntu18.04
>
> ROS版本：melodic
>
> qt版本：qt-opensource-linux-x64-5.9.9.run

一、安装输入法

由于使用的镜像没有安装输入法，无法直接输入中文，因此安装搜狗输入法或者使用其他解决方案，当前使用安装搜狗输入法

使用的安装包为：sogoupinyin_4.0.1.2800_x86_64.deb

安装命令

```
sudo dpkg -i sogoupinyin_4.0.1.2800_x86_64.deb
```

报错：dpkg: 依赖关系问题使得 sogoupinyin 的配置工作不能继续，需要安装 fcitx，命令：

```
sudo apt install fcitx
```

仍然报错：E: 有未能满足的依赖关系。请尝试不指明软件包的名字来运行“apt --fix-broken install”(也可以指定一个解决办法)

```
sudo apt-get --fix-broken install
```

重新执行安装fcitx和搜狗拼音命令：

```
sudo apt install fcitx
sudo dpkg -i sogoupinyin_4.0.1.2800_x86_64.deb
```

打开***fcitx*配置**软件，点击左下角“+”，找到**sogoupinyin**添加，并在***fcitx*配置**软件界面设置**sogoupinyin**排在第二位，如下所示

安装完成后若仍是只能输入英文，可以执行以下命令

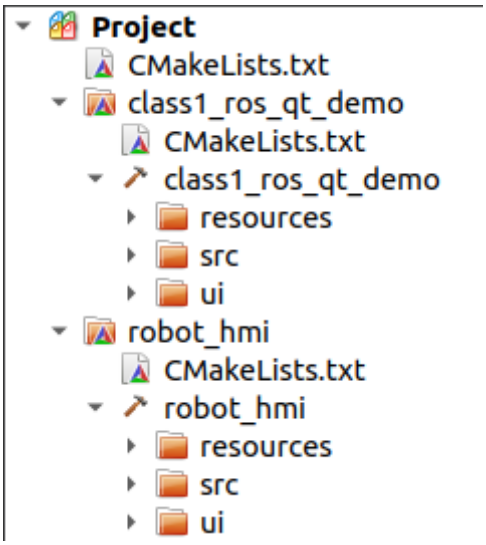
```
sudo apt-get install libqt5qml5 libqt5quick5 libqt5quickwidgets5 qml-module-qtquick2
sudo apt install libgsettings-qt1
```

重启输入法即可输入中文，使用**ctrl+空格**进行激活输入法。

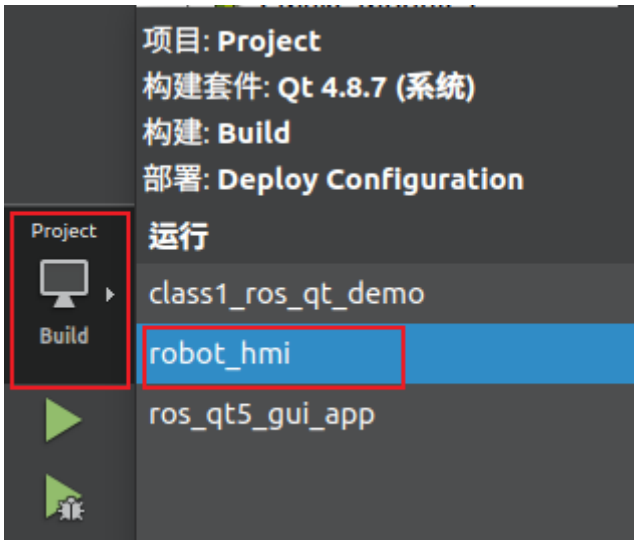
二、如何直接使用QT打开功能包修改源码并编译

首先必须确认功能包能够并且已经**使用catkin_make编译通过**，否则无法使用QT打开

编译通过后使用QT打开CMakeLists.txt文件，会自动导入项目工程所有文件展开如下：



针对不同的功能包，可以选择相应的项目名称，点击“run”即可完成使用QT重新编译并进行构建



三、kinetic如何在ROS程序中使用QT5库来开发GUI界面

ROS官方是支持QT4的，可以使用catkin_create_qt_pkg创建qt功能包，而ROS中很多著名的工具都是基于QT4。但是若是想使用QT5进行开发，**catkin_create_qt_pkg创建的功能包在QT5下会出现很多版本不兼容**的问题，修改需要消耗时间精力。此外，ROS并没有一个官方的IDE。

安装ROS qtc pluging版本qtcreator

> 如果是ros melodic版本，可以正常展开项目，但是ros kinetic版本可能不能正常展开。解决方法：新建kit，但是还是可能出现项目不能正常展开的问题，这里可以使用ros-qtc-pluging解决，针对于kinetic版本和melodic版本均适用：

1，下载安装包并安装
首先去ros-qtc官网下载系统对应版本的软件（qtcreator）
2，创建工作空间
此版本的qtcreator并不能像原生qtcreator打开项目，需要先创建工作空间：
文件->new file or project：
选中ros workspace：

> 注意，原生的qtcreator版本没有ros workspace能选择，只有ros-qtc-pluging版本的才有

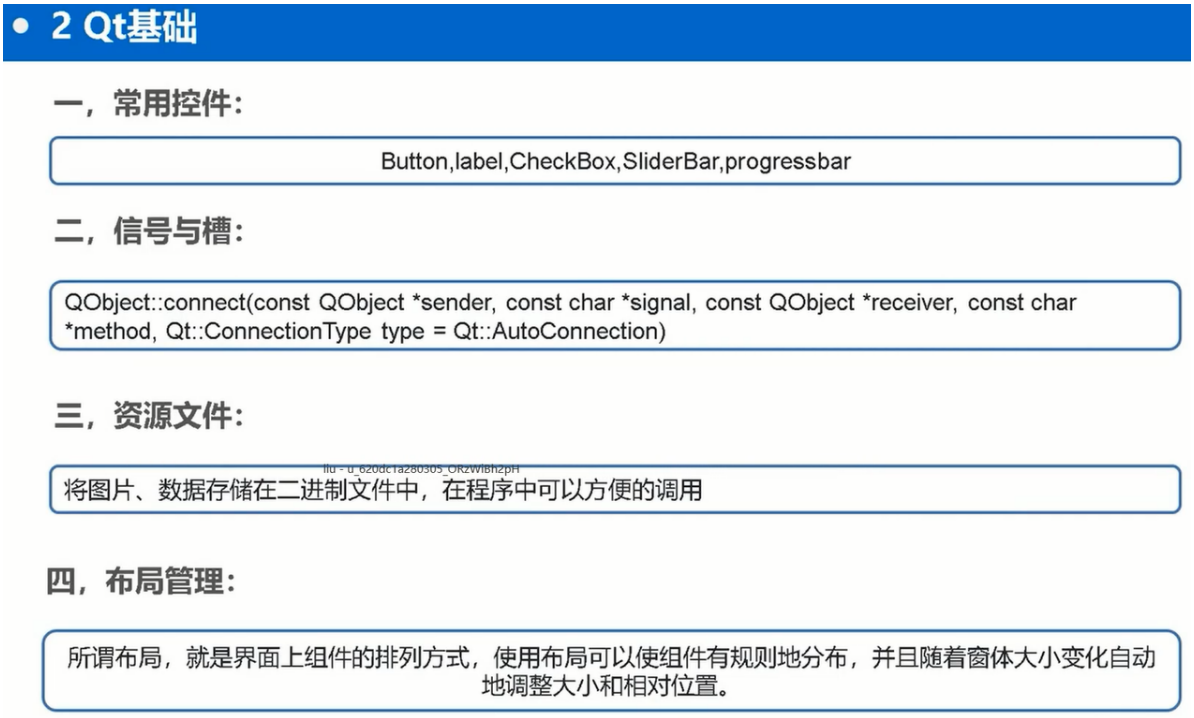
然后workspace path选中已有的工作空间目录(注意工作空间需要提前执行catkin_make)，name和工作空间名称同名即可，build system选中catkin_make

点击下一步即可自动展开

同时也会在工作空间目录下自动生成.workspace后缀的文件，打开工作空间即也通过这个文件打开（file->open file or project ）

同理也需要配置run路径(选中devel/lib目录下的可执行程序)，即可实现点击绿色三角形按钮同时构建并运行

四、QT的相关组件使用开发流程



新建工程**Qt Widgets Application**后会直接进入项目的源代码**编辑模式**：

这个模式和其他的开发工具是一样的，可以看到开发工具为生成的项目目录并且编辑源码。也可以在最左面的工具栏中点击相应的图标切换开发模式，其中设计模式的图标起初是不可用的，可以在项目的Forms目录中**双击.ui文件进入设计模式**，在设计模式下可以通过**拖动的方式**设计界面并完成布局。

在设计模式中，从左面的控件列表中拖动Label，Line Edit，PushButton，Text Edit，CheckBox，SliderBar，progressbar，Tab widgets，分别为标签，单行文本编辑框，按钮，文本编辑框，复选框，滑动条，进度条，带标签页的窗口。

进行简单使用设计

以label为例，从ui左侧的控件列表拖动Label到窗口，然后先点击**构建->构建项目**（这一步是为了刷新识别，使qt能够进行补全代码），然后转到cpp文件，由于是从ui里添加label，所以对象从ui开始，qt会自动将“.”转换成“->”，例如`ui->label`，如果是直接在cpp文件中使用new进行添加的label，则对象直接从`label`开始。然后label有很多方法，其中有setText可以设置显示文本，因此直接输入ui->label后会提示使用什么方法，然后选择setText即可，即是`ui->label->setText("mylabel");`其余同理。

以下是通过QLabel类的setText函数设置显示的内容:

```
void    setText(const QString &)
```

可以显示普通文本字符串

```
QLable *label = new QLable;
label->setText(“Hello, World!”);
```

使用connect链接信号与槽函数

1.在头文件，如mainwindow.h中的类里添加**声明**：

```
public slots:
    void slot_push_btn(bool);
```

2.在相关cpp文件，如mainwindow.cpp中的实现**定义**，slot_push_btn槽函数，设置显示文本为"clicked"（按F4可以在h文件和cpp文件快速跳转）：

```
void MainWindow::slot_push_btn(bool)
{
    ui->pushButton->setText("clicked");
}
```

3.connect源码中的定义为

```
QObject::connect(sender, SIGNAL(signal()), receiver, SLOT(slot()));
//信号发出者，处理的信号，信号接收者，处理动作方法（槽函数）。
```

其中第一个参数sender是信号的发送者；第二个参数signal为发送信号，可以选中相关组件如“pushButton”，然后按F1进入帮助快速查找相关内容；第三个帮助为信号的接受者；第四个参数为方法，即槽函数。

相关概念认知：

- 信号（Signal）就是在特定情况下被发射的事件。
- 槽（Slot）就是对信号响应的函数，槽就是一个函数。
- 信号与槽之间的关联：是用 QObject::connect() 函数实现的。
- sender 是发射信号的对象的名称。
- signal() 是信号名称。信号可以看做是特殊的函数，需要带括号，有参数时还需要指明参数。
- receiver 是接收信号的对象名称，slot() 是槽函数的名称，需要带括号，有参数时还需要指明参数。
- SIGNAL 和 SLOT 是 Qt 的宏，用于指明信号和槽，并将它们的参数转换为相应的字符串。

MainWindow::MainWindow(QWidget *parent) ：QMainWindow(parent), ui(new Ui::MainWindow)中添加**connect**函数，根据以上按要求实现如下：

```
connect(ui->pushButton,SIGNAL(clicked(bool)),this,SLOT(slot_push_btn(bool)));
```

点击构建运行即可查看实现connect链接信号与槽函数后的效果。

资源文件

添加资源文件，右键项目，添加新文件-〉**Qt Resource File**-〉命名

添加完成后在工程目录中会出现相对应的资源qrc文件，右键选择用资源管理器打开，先添加前缀，前缀可自行设置，后选择要添加的图形文件，图形文件一般放在相应工程文件夹中，可新建image文件夹并将所有工程所需图片放至文件夹。

使用时，填入相关路径path或url即可，如下：

```
setWindowIcon(QIcon(":/images/icon.png"));
```

常用的布局

QLayout类是布局管理器的基类，是一个抽象基类，继承自QObject、QLayoutItem，QLayoutItem类提供了一个供QLayout操作的抽象项目。

Vertical Layout (QVBoxLayout)	**垂直布局**	**垂直布局，让布局里面的控件按照竖直方式排列**
Horizontal Layout (QHBoxLayout)	水平布局	水平布局，让布局里面的控件按照水平方式排列
Grid Layout (QGridLayout)	网格布局	QGridLayout占用提供给它的空间，将其划分为行和列，并将其管理的每个小部件放入正确的单元格中。
From Layout (QFormLayout)	表单布局	QFormLayout 只包含 2 列，且第一列通常放置第二列控件的标签

垂直布局：

```
创建垂直布局    QVBoxLayout *VB=new QVBoxLayout;
布局中添加控件   VB->addWidget()
窗口中添加布局   Widget->setLayout(VB);
```

水平布局：

```
创建水平布局    QVBoxLayout *HB=new QVBoxLayout;
布局中添加        HB->addWidget()
窗口中添加布局   Widget->setLayout(HB);
```

网格布局：

```
void QGridLayout::addLayout(QLayout *layout, int row, int column, Qt::Alignment
alignment = Qt::Alignment())
向网格中的 (row, column) 位置处添加 layout 布局管理器。
 void QGridLayout::addLayout(QLayout *layout, int row, int column, int rowSpan,
int columnSpan, Qt::Alignment alignment = Qt::Alignment())
将 layout 布局管理器从 (row, column) 位置开始，跨 rowSpan 行和 ColumnSpan 列添加到网格
中，并且可以自定义该布局控件的对齐方式。
```

表单布局：

```
QFormLayout：：addRow（QWidget *label， QWidget *field)
    将一个新行添加到此窗体布局的底部，其中包含给定的标签和字段
addRow（const QString &labelText， QWidget *field)
   后台创建一个 QLabel，并将 labelText 作为其文本。该字段被设置为新的 QLabel 的伙伴
   将指定的 field 控件和 labelText 描述信息添加到表单控件的末尾。
addRow（const QString &labelText， QLayout *field)
   可以在表单中添加布局
```

改变布局大小：

双击ui文件进行设计模式后，在QT右下侧的可以对控件、布局等设置大小max、min等。

五、QT创建节点

六、QT添加RVIZ组件

rviz是ros的一个可视化工具，用于可视化传感器的数据和状态信息。
rviz支持丰富的数据类型，通过加载不同的Dispalys类型来可视化，每一个Dispaly都有一个独特的名字。

常见的display类型

类型	描述	消息类型
Axes	显示坐标系	–
Gamera	从相机视角显示图像	sensor_msgs/Image sensor_msgs/CameraInfo
Grid	显示网格	–
Image	显示图像	sensor_msgs/Image
LaserScan	显示激光雷达数据	sensor_msgs/LaserScan
Image	显示图像	sensor_msgs/Image
PointClode2	显示点云数据	sensor_msgs/PointCloud2
Odomerty	显示里程计数据	nav_msgs/Odometry
PointClode2	显示点云数据	sensor_msgs/PointCloud2
RobotModel	显示机器人模型	–
PointClode2	显示TF树	–