本文档对TinyROS的使用方法进行说明，仅供小组内部参考，如有不够专业的地方还请见谅。如有任何问题欢迎及时反馈。

1. 什么是TinyROS

TinyROS是仿照ROS的节点-话题，订阅-发布的通信机制编写的一个任务调度系统。虽然叫TinyROS，但其实和机器人没啥关系，就是一个进程间通信的工具，可以在局域网内跨设备、跨平台运行。

1. 通信机制

弄清几个关键词：节点、话题、消息、订阅、发布。一个进程就是一个节点，是实现某个特定功能的程序。节点可以向话题发布消息，也可订阅一个话题。当节点向话题发布消息时，所有订阅了这个话题的节点就会收到这条消息。从而实现了进程间通信。

一个节点可以订阅任意多个话题，也可以向任意多个话题发布消息。一个话题可以被多个节点订阅，也可以有多个节点向其中发布消息。

节点和话题均以名称作为标识，一个局域网内，节点的名称不能重复，话题的名称不能重复。

消息具有类型，即具有自己的数据格式、内容。一个话题只拥有一种消息类型，只能向该话题发布这种类型的消息。发布和订阅时都需要指定话题和消息类型，当话题还不存在时，就会创建话题，并且具有该消息类型，而当话题已经存在时，应确保指定的消息类型与话题实际对应的消息类型匹配。一种类型的消息，可以存在多个不同的话题都属于这种类型，这些话题之间相互独立，发布和订阅的目标是一个话题，而不是一种消息类型。

虽然实现方法与ROS不同，但是功能基本类似，如有疑问也可以看一看ROS的话题通信机制。

此外，还存在一个Master，是用来管理节点，牵线搭桥，帮助节点建立通信用的。运行任何节点之前，应该先运行Master。Master的程序已经写好，无需改动，只需每次运行的时候修改json文件里的参数即可。详细的信息见后文。

1. 使用方法
2. 准备工作

引入TinyROS，对项目进行一些必要的配置：

右击自己的项目下的“引用”，选择添加引用：

图形用户界面, 文本, 应用程序

描述已自动生成

在“共享的项目”——“解决方案”里选择Share项目

图形用户界面, 文本, 应用程序

描述已自动生成

在自己的项目上右击，选择“属性“

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

在“C/C++“——”预处理器“里根据平台添加一条宏定义，如果是在Windows下运行，就添加TINYROS\_ON\_WINDOWS，如果是在Linux下运行，就添加TINYROS\_ON\_LINUX

图形用户界面, 文本, 电子邮件

描述已自动生成

在“C/C++“——”语言“里选择C++标准为14或者更高

图形用户界面, 文本, 电子邮件

描述已自动生成

如果是Linux，则在“C/C++“——”命令行“里添加一条参数”-pthread”，如果是Windows则无需进行操作

图形用户界面, 文本, 应用程序

描述已自动生成

在自己的项目右击，选择“添加”——“现有项”

图形用户界面, 应用程序

描述已自动生成

在Share/OpenSSL/lib下，找到目标平台对应的静态库，添加到自己的项目中

图形用户界面, 应用程序

描述已自动生成

注意分清不同的目标平台，包括操作系统和芯片架构。

至此，准备工作完成。

1. 使用TinyROS

包含头文件



之后就可以使用了。

TinyROS的所有内容位于TinyROS命名空间下。

1. 关于Master

Master的程序一般来说不需要改动，直接生成并运行就好了。但是运行之前，需要修改json配置文件中的“ListenIP”

图形用户界面, 文本, 应用程序

描述已自动生成

将它改为本机的局域网IP。注意，一台机器可能有多个IP，它们属于不同的网络，请确保填的是局域网的那一个。

另外，该json文件需要在应用程序的运行目录下，这里需要注意的是，如果从Visual Studio启动，那么运行目录是源代码所在的目录，如果是去文件夹里找exe文件运行，那么运行目录就是exe的目录，因此根据启动方式的不同，需要修改的json文件的位置也不同。

一个局域网内运行一个Master即可。

1. 关于Node

只需要一句，即可将当前进程初始化为一个Node：

图示, 示意图

描述已自动生成

必须在使用Publisher或Subscriber之前完成节点的初始化。建议在程序的一开始就调用。

1. 关于Publisher

Publisher是用来发布消息的类。当你需要向一个话题发布消息，就创建一个Publisher的对象。每个话题只需要一个Publisher就可以了；每个Publisher也只发布到一个话题。

Publisher的构造函数被禁用，不能直接声明变量，而是需要声明一个指针，然后，使用NewPublisher函数创建一个Publisher的对象并且获取指向它的指针：





NewPublisher具有一个模板参数，是消息的类型。函数参数是话题的名称。

之后的任何时候，都可以调用该Publisher的Publish方法，向这个话题中发布消息。

文本

描述已自动生成

1. 关于Subscriber

Subscriber是用来订阅消息的类，当年需要订阅一个话题，就创建一个对象。与Publisher类似，不能直接使用构造函数，而是使用NewSubcriber：







除了话题名称，还有一个参数就是回调函数，见后文。

之后，Subscriber就开始在后台运行了，不会阻塞。每当有节点向Subscriber所订阅的话题中发布消息，Subscriber就会收到，然后以该消息作为参数，调用callback所指定的回调函数。

1. 关于MessageCallback

这是一个类，封装了一些方法，用于实现回调函数。在创建Subscriber的对象时，先需要创建一个MessageCallback的对象，传递给Subscriber。每当收到消息，就会调用该MessageCallback对象里的所有回调函数。

该类是一个模板，需要指定消息类型，并且需要和Subscriber一致。构造函数有一个参数，是最高可以注册的回调函数的数目，默认为5，根据需要填写即可。



创建了MessageCallback之后的任何时候，不论该callback对象是否传递给了Subscriber，都可以向callback对象注册新的回调函数，也可以取消一个已经注册的函数。

注册和取消注册使用的是Register和Unregister函数，具体的操作可以看Demo。一共有4种函数可以注册，分别是普通的函数、类的静态函数、类（对象）的函数、函数对象。

不管是哪种函数，都具有相似的声明：无返回值，只有一个参数，类型就是消息的类型



1. 关于Message

Message是TinyROS的消息类型，是一个抽象类，也是其他所有具体的Message的基类。上述的Publisher、Subscriber和MessageCallback的相关的模板参数和函数参数，都具有Message类型。也就是说，想要使用它们，就需要有一个消息类型。

各Message类的相关定义在Message.h和Message.cpp文件中，这可能是唯一需要你们稍微看一下的源文件。

既可以定义自己的消息类型，也可以使用一些已经实现了的消息类型。首先介绍一下已经实现的消息类型：

SimpleObjectMessage：  
图片包含 网站

描述已自动生成这是一个可以用来传输任何简单类型的消息类型，这是一个模板类，模板参数TValue就是类型名。该消息的工作原理是直接使用TValue的对象的内存进行传输，因此，TValue必须是一个简单的类型。所谓简单的类型，就是TValue的对象直接包含自己的值，比如struct {int x;int y;}是可以的，因为这个结构就是8个字节，包含两个整数，把这8个字节复制到这个结构的另一个变量，就可以实现结构的复制，甚至是跨越进程、设备的。但是，如果类型包含指针，那直接复制显然是没有意义的。因此，TValue只能是简单的类型，不能包含或者嵌套地包含指针、静态成员等。

该SimpleObjectMessage具有如下字段和方法：

文本

描述已自动生成

即该SimpleObjectMessage可以使用一个TValue类型的值进行构造，也可以直接访问到它所携带的TValue类型的值Value。

SimplaObjectArrayMessage：



与SimpleObjectMessage类似，是一个模板类，要求与SimpleObjectMessage一样是简单的类型。该类型用于传输简单类型的数组。具有如下字段和方法：

文本

描述已自动生成

该消息可以使用一个TValue指针（数组名）和一个整数（标识数组的长度）来构造。可以通过GetCount()获知元素的个数，通过[]运算符获取指定索引的元素的值，也可以通过GetValuePointer()直接获取指向内部元素的指针。

StringMessage：



这个类比较简单，就是用来传输字符串的，可以通过如下方法进行构造，或者获取所携带的字符串值。

文本, 信件

描述已自动生成

定义自己的Message类型：  
想要定义自己的消息类型当然也是可以的，只需要继承自Message类，然后实现指定的方法，就可以使用，使用方法和已有的类型完全一致。在此基础上，可以任意添加自己想要的功能。

需要实现的方法如下：

文本, 信件

描述已自动生成

GetTypeID()用来返回一个标识本类型的HashCode，写法比较固定，可以参考一下已有的类是怎么实现的。

Serialize()用来把自己编码成字符串，Deserialize()则正好相反，将一个字符串解码回自己的值，该方法会直接修改自身的值。

NewDeserialize()也是解码，但它不改变自己的值，而是返回一个新的对象。

Serialize和Deserialize可以任意实现，只要能自洽即可，即先序列化之后再反序列化，能够正确还原出所需要的值。

实现了Deserialize之后，NewDeserialize也就很简单了，只需new一个对象，然后调用该对象的Deserialize，最后返回该对象即可。

(g) 关于异常

Node、Publisher、Subscriber会抛出一些异常。TinyROS自定义了一些异常，所有这些异常都具有同一个基类TinyROSException，TinyROSExecption又是继承自exception

。

虽然具体的异常类型有很多，但是目前也没有什么处理办法，因此也不用太在意。可以catch一下基类，甚至可以不catch，反正抛了异常基本上就得重启了。

文本

描述已自动生成

但是异常会携带一些信息，可以把他们打印出来，有助于发现具体问题所在。这时，请注意使用引用来捕获异常，这样，才会打印原始的信息。否则只会打印基类的信息。