GNURadio 从入门到入土

TerayTech 2020.11.25

（2020.11.16 内部预览版）本文档不代表最终文档，仅为内容核对更正使用。

目录

前言:

欢迎来到 GNU Radio 系列教程。本教程作者 Bilibili: TerayTech，18 级通信工程本科生。

因为最近关注软件无线电 SDR 的人越来越多，粉丝私信询问相关内容也越来越多，为了方便更多的人更系统地学习 GNU 无线电软件，因此作本篇教程，希望能够尽自己微薄的努力帮助更多的人。

在本教程中，可能会出现部分软件无线电平台。在此特别感谢 Digilent 公司的大力支持，他们所提供的 USRP 软件无线电平台作为实际操作的硬件平台，对于本系列教程的制作起到至关重要的作用。如果大家有兴趣购买 ETTUS 出品的软件无线电设备，欢迎到 Digilent 官方淘宝店铺选择购买。

第一章 GNURadio 和软件无线电概述

1.1 什么是 GNU Radio

GNU Radio 是一个软件框架，使用户能够设计、模拟和部署功能强大的软件无线电系统。它是一个高度模块化的，面向“流程图”的框架，带有一个全面的处理模块库，可以轻松组合并构成复杂的信号处理系统的应用程序。

GNU Radio 已用于大量的无线电应用程序。包括音频处理，移动通信，跟踪卫星，雷达系统，GSM，数字无线电等等，所有这些都在计算机软件中使用。

1.2 为什么我们要使用 GNU Radio

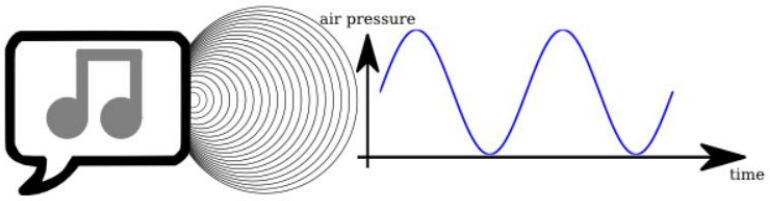
以前，在开发无线电通信设备时，工程师必须先开发用于接收并处理特定信号的接收机，来对特定信号传输进行解码或编码。随着数字信号处理与其算法越来越复杂，这些信号处理的平台也变得越来越复杂，通常需要较为高速的 ADC、FPGA 以及能将实时数据串流到计算机平台的连接芯片等，每个系统所对应的硬件平台不一定是一样的，这就带来了巨大的开发成本。通过使用软件无线电（SDR）设备进行模拟信号处理，在相同的硬件平台上可以同时兼容运行各种不同的软件程序，不仅节约了开发成本，也提高了开发新系统的效率。

1.3 关于数字信号处理

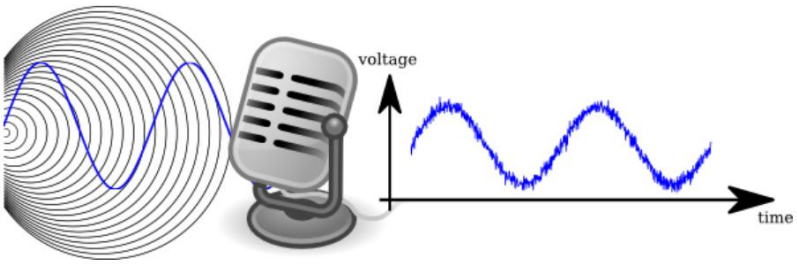
作为一种软件框架，GNU Radio 通过硬件平台串流的比特数据流输入到计算机中，并在操作系统中运行相应的应用程序以此达到对特性信号进行处理的目的。

我们都知道计算机只能处理数字信号。如何去理解数字信号呢?

简单举个例子：当你想要录制一段人声的时候，说话的人会产生声音信号，该信号由震动导致周围气压发生变化而产生。这样一个时变的物理量就是一种信号。

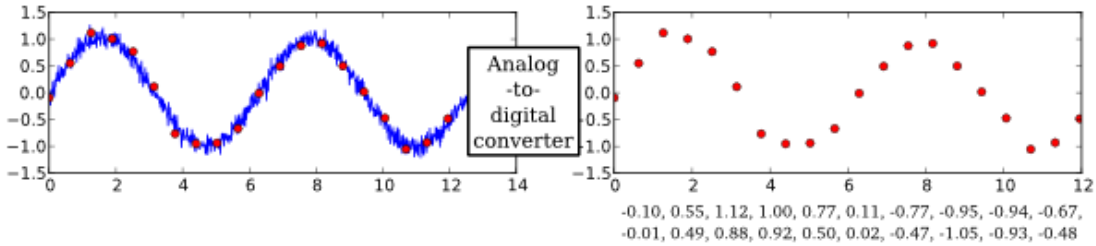


当空气波到达麦克风时，麦克风将变化的压力转换为电信号，即可变电压



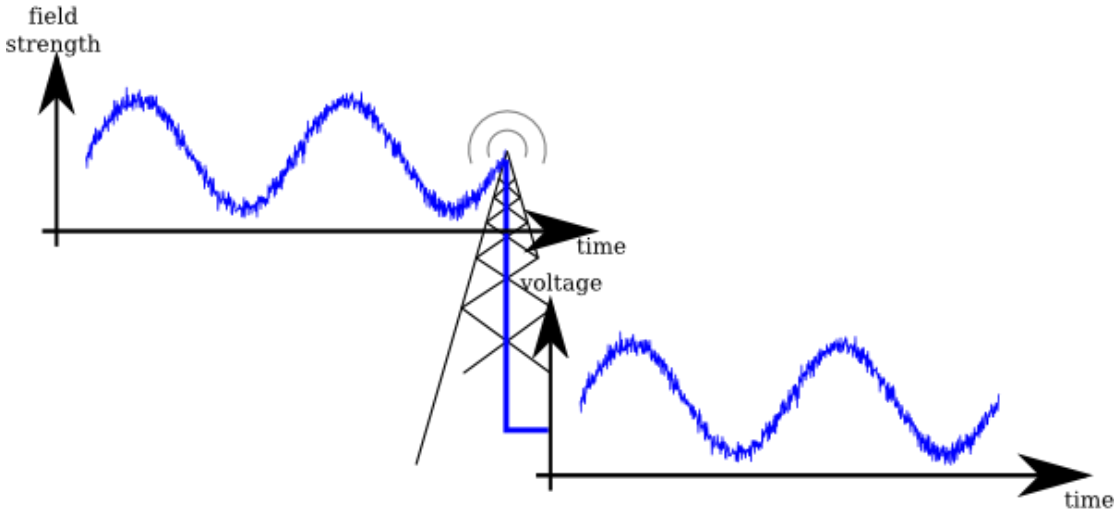
现在我们已经将信号转化为了电信号，在一些模拟系统中，已经可以开始对信号进行处

理。但是对于我们的计算机系统，一个数字的系统，这还远远不够。为了使计算机能够处理这样的数据，我们还需要满足两个条条件：1.是有限点数的 2.是在有限时间之内的



因此，该数字信号可以由称为样本的数字序列表示。采样之间的固定时间间隔直接影响到采样率。提取物理量（电压）并将其转换为数字样本的过程由模数转换器（ADC）完成。相反，我们还有数模转换器（DAC），可从将计算机中提取数字序列转换为模拟信号。

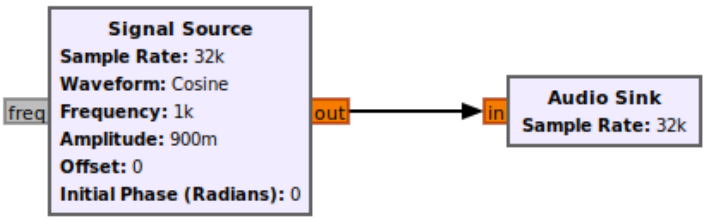
现在我们已经有了一个数字序列，我们的计算机就可以使用它进行各种操作。



同样，电磁波显然也是一种波，它跟声波有许多相同的性质。我们可以用天线将变化的电信号发射出去，这个电信号一般位于一个较高的频率上，可以是数百 KHz 到 GHz。通过使用软件无线电接收机，我们可以接收并对这些信号进行处理，以此进行我们想要的操作。

1.4 GNU Radio 是如何工作的

在 GNU Radio 中，为了处理数字信号，我们可以使用简单的流程指示箭头将其连接：



在上图中，Signal Source 即为信号源，左边的输入接口可以输入频率参数，右边的输出接口可以输出音频数据流。右边的 Audio Sink 为音频接收器，允许通过扬声器或其他音频设备播放出输入的信号。这就构成了一个十分简单的流程图，点击软件中的运行按钮即可非常简单快捷的编译流程图并运行。

GNU Radio 是一个框架，用于开发这些处理模块并创建流程图。软件自带大量的处理模

块，在这里简单举例一些：

- Waveform Generators 信号发生器
    - Constant Source 常数源（可以理解成直流分量）
    - Noise Source 噪声源
    - Signal Source (e.g. Sine, Square, Saw Tooth) 信号源

- Modulators
    - AM Demod AM 解调
    - Continuous Phase Modulation 连续相位调制
    - PSK Mod / Demod PSK 调制/解调
    - GFSK Mod / Demod GFSK 调制/解调
    - GMSK Mod / Demod GMSK 调制/解调
    - QAM Mod / Demod QAM 调制/解调
    - WBFM Receive 宽带 FM 接收机
    - NBFM Receive 窄带 FM 接收机

使用这些模块，我们只需要进行相应的连接操作，就可以快速搭建数字信号处理系统。另外，当然你也可以自己开发新的 block，或者将现有的块与其他软件结合在一起，开发出新的功能。

因此，GNU Radio 主要是用于开发信号处理模块及其交互的软件框架。它带有广泛的标准块库，开发人员可以在其中构建许多系统，是十分方便的软件无线电开发工具。

第二章 GNU Radio 软件安装与配置

GNURadio 的官方 GitHub 页面为 https://github.com/gnuradio/gnuradio。其首页中也明确说明了对于不同操作系统的不同安装方式。

2.1 操作系统的选择

我个人最推荐使用 Ubuntu18.04 我在这个系统版本上搭建过很多次所需要的环境，没怎么出过问题，使用一直很稳定。19 版本或许可以，我没有尝试过，但是 20 版本一定不可以，因为有接到过软件报错的情况报告。

2.2 Linux 环境下的直接安装

对于 GNU Radio，如果只是简单轻度使用我就建议大家直接使用 Linux 的二进制软件包安装。最快捷方便而且最重要不容易出错。根据 GNURadio 官方 GitHub 界面，首先的安装方式也是直接使用 apt 安装。
以下命令适用于 Debian，Ubuntu 及其衍生版本。它将使用 Python2 安装 GNURadio 3.7 版

```
sudo apt install gnuradio
```

对于以上操作系统，直接执行这条命令即可安装完成。如果遇到报错建议自行查询报错信息解决。对于其他 Linux 发行版，请查阅：
https://wiki.gnuradio.org/index.php/InstallingGR#From_Binaries

2.3 Linux 下使用 PyBOMBS 辅助自动从源码构建

PyBOMBS 是安装 GNURadio 以及相关软件工具的一个快捷工具。你可以使用它来安装各种 SDR 设备所依赖的支持库，绝大部分操作都是全自动的。

PyBOMBS 是方便用来从源代码构建 GNU Radio，UHD 和各种 Out of Tree（OOT）模块，然后将其安装到指定的用户目录中的工具。在使用之前，PyBOMBS 会检测用户的操作系统并在构建的第一阶段加载所有先决条件（可能会出现各种花式报错）。如果你对于自己解决 Linux 环境配置问题不是很有信心，我不建议你使用这种方法安 gnuradio。

注意!!：GitHub 中详细描述了安装的步骤，请自行参阅：
项目地址：https://github.com/gnuradio/pybombs

因为它是从源代码安装 GNU Radio，所以第五步可能需要一些时间，要进行更快的安装，请参阅 https://wiki.gnuradio.org/index.php/InstallingGR#Ubuntu_PPA_Installation

2.4 Linux 下手动从源码编译构建请参阅：

https://wiki.gnuradio.org/index.php/InstallingGR#From_Binaries

2.5 Windows 环境下的安装

在 Windows 环境下，官方提供了非正式版的 GNU Radio 3.7 和 3.8 的安装文件，虽然我也不推荐你真的在 Windows 平台运行这个软件，但是它在 Win 平台是真的可以使用的。不管是 USRP 还是 PlutoSDR，有驱动程序的话就可以使用。对于 USRP，可能存在固件版本的问题，按照教程后面的解决办法是可以解决的。

相关的安装软件包在这里下载：http://www.gcndevelopment.com/gnuradio/index.htm

2.6 Mac OS X 环境下的安装

你是认真的？
请参阅：https://wiki.gnuradio.org/index.php/MacInstall

第三章 教程初阶

3.1 熟悉使用 GNU Radio Companion

学习目的:

- 使用标准块库创建流程图
- 了解如何使用检测模块 Sink 调试流程图
- 了解 GNU Radio 中的采样和调节功能
- 了解如何使用文档找出模块的功能

在本教程中，我们将从简单框图开始，探讨如何使用 GNU Radio 的图形工具 GNU Radio Companion（GRC）来创建不同的框图。GRC 是为了简化 GNU Radio 而诞生的，有了它，我们可以以图形化编程的方式创建 python 脚本，替代了传统的复杂代码编写，进而降低软件无线电编程的入门门槛。

那么我们开始。首先打开终端，输入以下指令。

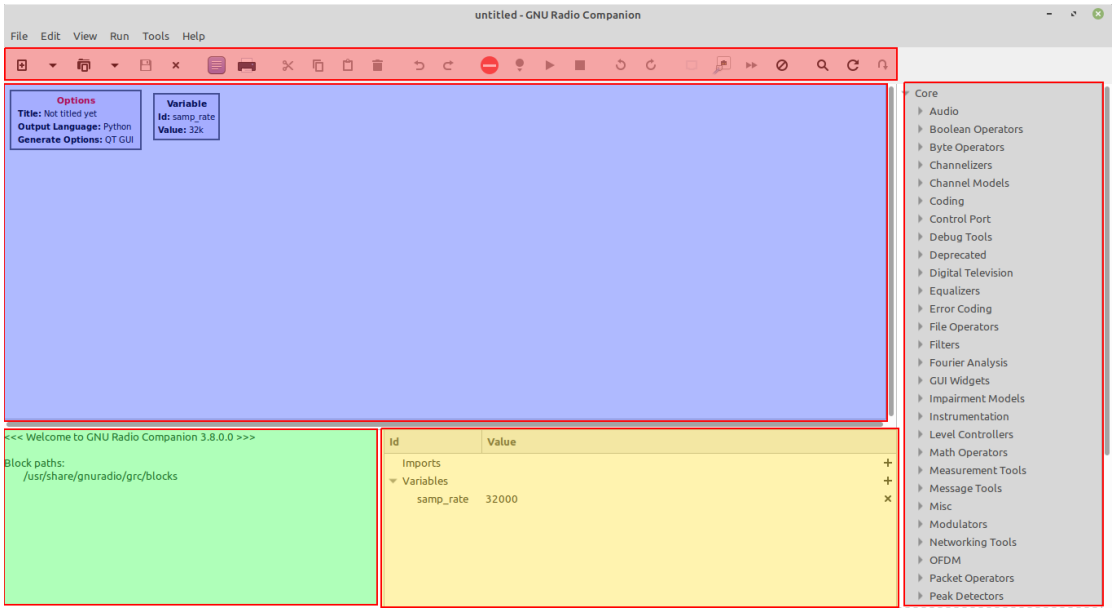
```
$ sudo gnuradio-companion
```

或者直接单击软件图标，也是可以运行软件的。

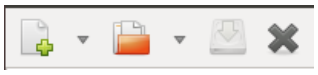
如果你发现不仅应用程序中没有出现软件图标，而且终端也不能打开这个软件，那么你的安装很有可能出现了问题。请检查安装是否存在问题。

这里有一点区别。当你通过终端运行 GRC 时，下图绿色部分的终端会同时在系统终端里显示。而如果直接通过点击软件图标运行则只能在 GRC 的终端面板中观察信息。

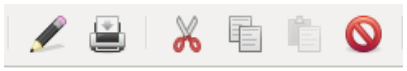
首先我们来介绍软件界面。总共分为五个部分：库，工具栏，终端，工作区和变量。



红色区域为工具栏部分，放置了平时最常用的工具，比如运行、停止、编译等重要功能按键。

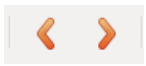


新建、打开、保存、关闭

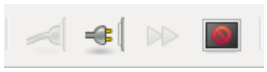


打开/关闭变量编辑器、截图、剪切、复制、粘贴、删除选中模块

查看错误信息、编译流程图、执行流程图、停止运行流程图



撤销、重做



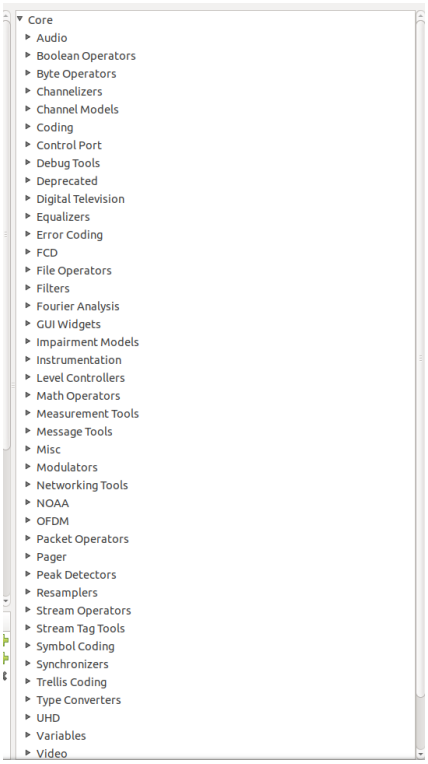
启用选中模块、禁用选中模块、绕过选中模块、反转禁用连接/模块的状态

查找模块、重置模块、打开选中阶梯模块源码

蓝色区域即为我们绘制具体流程图的地方。我们可以将右边灰色部分库中的模块拖入蓝色区域，并且将他们通过箭头连接起来，这样就可以构成一个真正的信号处理系统。

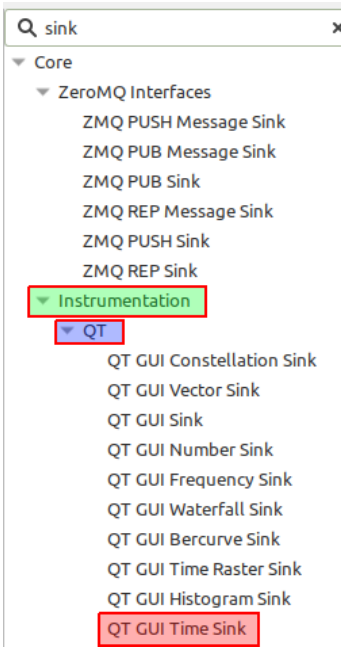
黄色部分显示的是当前框图中所使用到的变量。在蓝色部分的左上角可以看到两个方框，分别是 Options 与 Variable，这两个是创建工程时就会自动创建的。

在界面的右边灰色区域中，存放了大量可以用于拖拽到流程图中的模块。其中有很大一部分是软件安装时就自带的，如果你安装了其他 gnuradio 附属的插件脚本，也会一并显示在框中，通常自行安装的会显示在最后面。

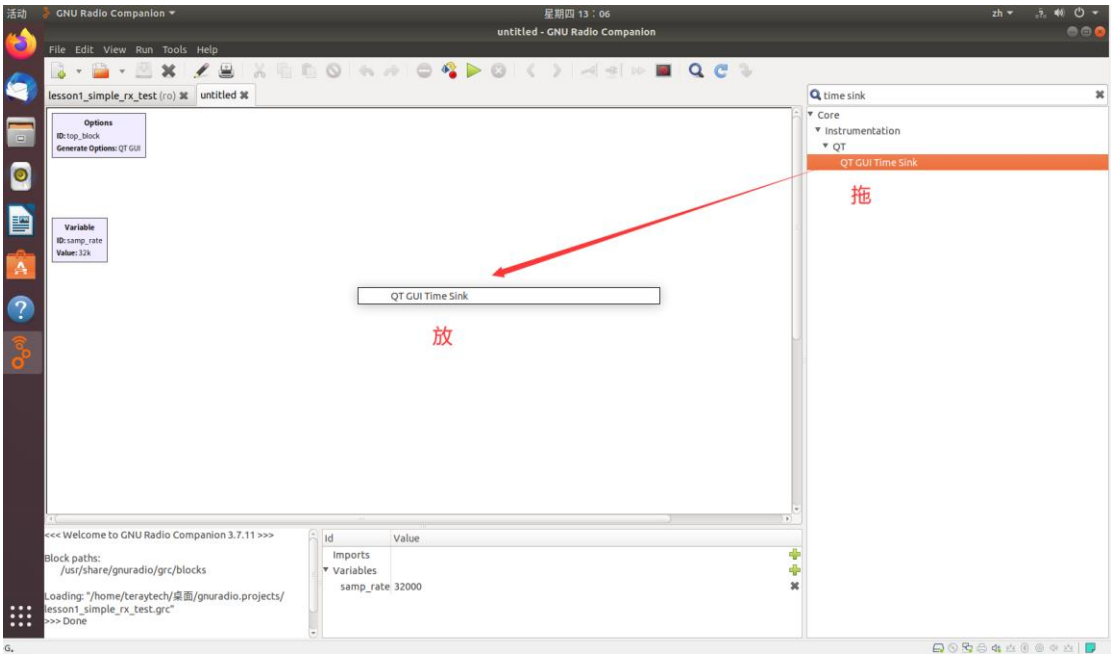


因为模块非常多，因此平时寻找想要的模块时一个一个手动翻找会非常麻烦。此时可以点击工具架上的放大镜图标，或是输入 Ctrl + f 输入该块的关键字进行检索，就可以更容易的找到这个 block。

例如，这里我们输入 sink（接收器）,就可以看到包含单词“接收器”的所有块以及将在其中找到每个块的类别。

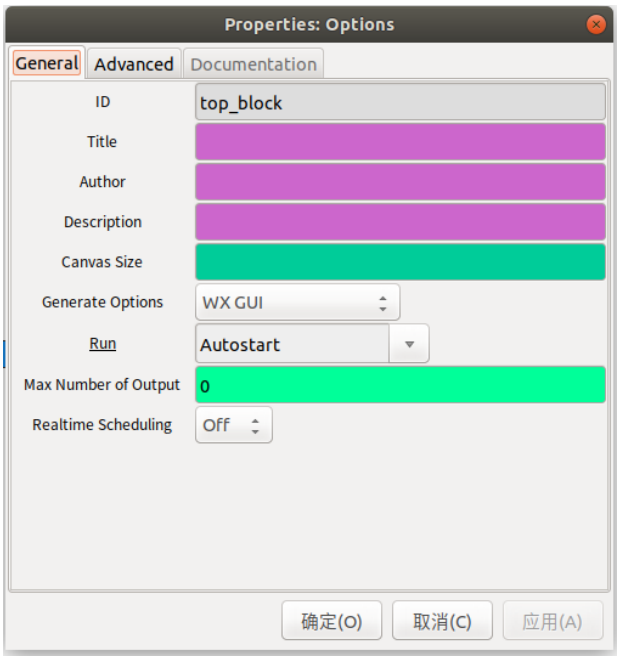


现在，我们来添加一个名为 **QT GUI Time Sink** 的块，方法是单击其名称并将其拖动到工作区中，或者双击其名称以将其自动放置在工作区中。



工作区包含构成流程图的所有块，在每个块内部都有不同的块参数，但是，每个新流程图都需要有一个特殊的块，称为**“选项块”**。让我们双击选项块以检查其属性。

双击 opthions 模块可以看到它的具体内容。Options 中包含了工程的特殊参数设置，每个流程图仅允许存在一个这样的选项模块。



上面的 ID，title，author，description，分别表示这个流程图的 ID，标题以及作者和简介。该块的 ID 决定了生成文件的名称和类的名称。例如，一个 ID 为 top_block 的文件将生成文件 top_block.py 和 top_block 类。

Cavans Size 窗口大小控制流程图编辑器的尺寸。窗口大小（宽度，高度）必须介于（300，300）和（4096，4096）之间。

Generate options 生成选项控制生成的代码的类型。非 GUI 流程图应避免使用带有 GUI 的组件或图形变量控件。

Run：流程图的运行可由变量控制，以在需要时启动和停止流程图。

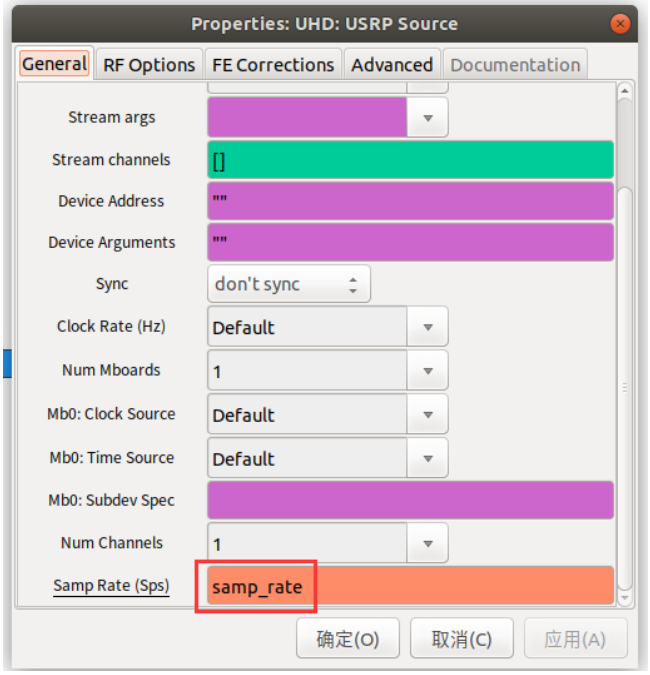
Max number of output 最大输出数是流程图中任何方框所允许的最大输出项数；要禁用此功能，将 max_nouts 设置为 0 即可。使用此功能可以调整流程图可以显示的最大延迟。

可以注意到另一个关键的东西。我们可以输入信息的字段中存在的不同颜色。这些实际上对应于不同的数据类型，我们将在本教程的后面部分介绍这些数据类型。

GRC 将我们在编辑器中创建的流程图转换为 Python 脚本。因此当我们执行流程图时，实际上是在运行编译好的 Python 程序。ID 用于命名该 Python 文件，该文件与.grc 文件保存在同一文件夹内。默认情况下，ID 是默认值，因此它将创建一个名为 default.py 的文件。更改 ID 可让我们更改保存的文件名，以便更好地管理文件。在 GNUradio 3.8 中，如果不更改默认 ID，则会收到错误消息，因此需要更改此 ID 才能运行流程图。

Variable 即变量，它的 ID 是 samp_rate，你可以在框图中的其他地方调用这个变量。

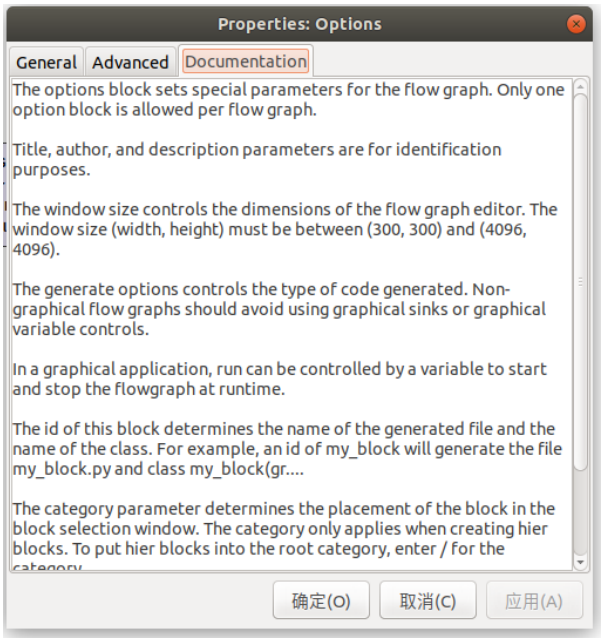
例如：



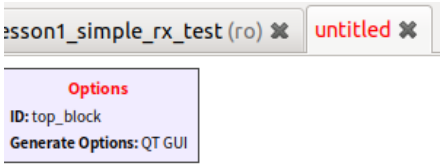
这样这里的数值就会随着该变量的变化而变化。

如果你点进了设置面板的第三个选项卡，就能看到有关这个 block 的文档。通常情况下正规的 block 都是会写使用文档的，当然少数自定义的模块可能是没有的。

虽然这些说明是英文的，但是我十分建议大家自己去用谷歌等工具翻译一下这些文档，因为教程不可能每个详细的点都能讲到，有时还是得靠自己查一查的。

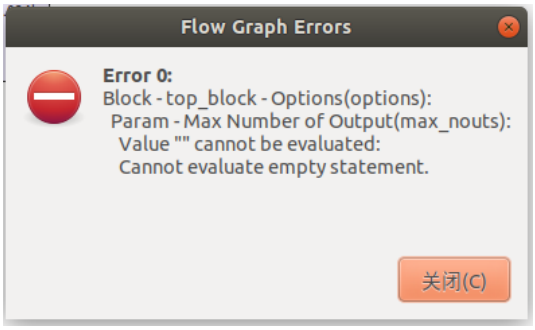


如果我们删除了一个重要的参数，或是填入了什么不正确的参数，以至于我们的框图无法正常运行，那么此时你会看到执行按钮变成灰色不可点击的状态。此时报错信息按钮亮起，并且在出现错误的 block 上，它的名称出现了红色的高亮显示。

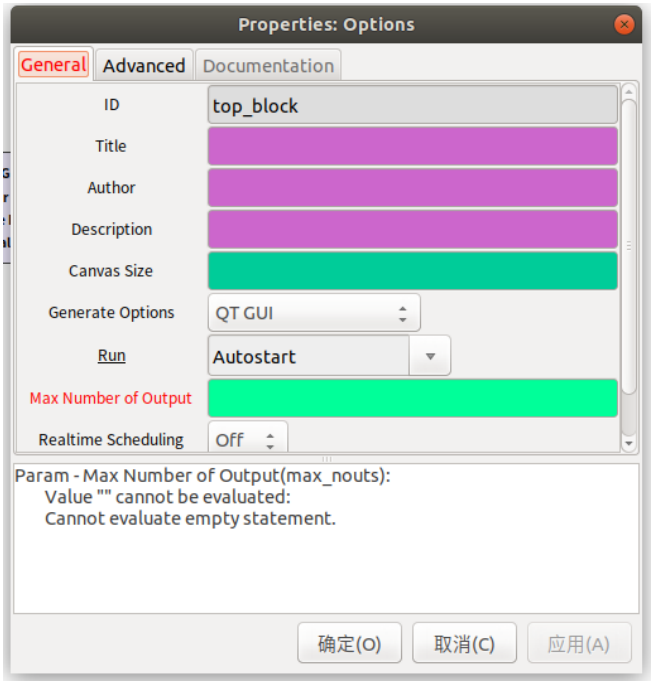


你可以点击这个按钮，就可以看到存在问题的错误信息。

在错误信息中，详细指出了错误出现的位置（如果看不懂就用翻译工具翻译一下，不过英语这么差我建议你直接放弃，这玩意高中生都能看懂。#日常劝退）我们只需要按照报错信息所提示的位置：模块-top block-选项中的一个参数：max_nouts



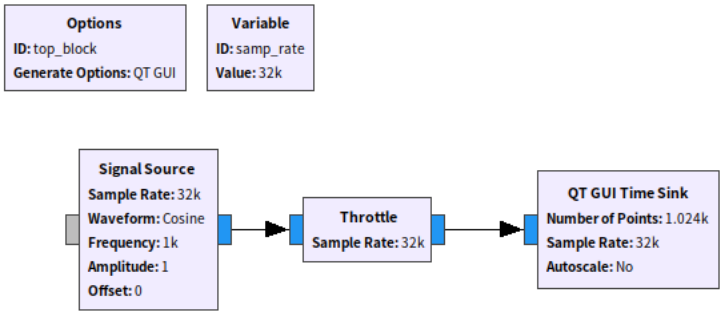
双击打开这个模块，就可以看到在模块中也存在同样的错误信息提示位于正下方。



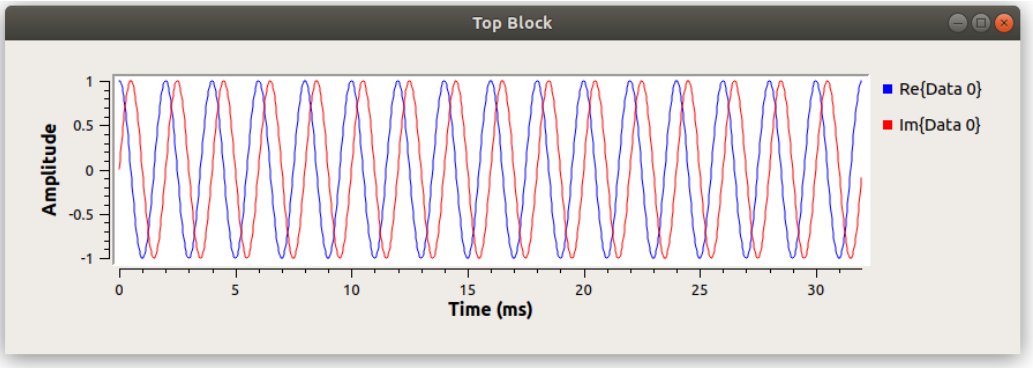
错误明确指出，在这个输入框中数值“”不能被接受，因为这里必须填写的是一个数字，我们填写数字 0 进去后点击确定，即可发现错误信息已经消失。执行按钮也亮起，说明框图无明显错误，可以正常运行。

现在，我们对**如何找到块**，**如何将它们添加到工作区**以及**如何编辑块属性**有了更好的了解，下面我们随意以几个 block 组成一个框图来进行简单的演示。

刚才我们拖入了 QT GUI Time Sink 这个模块，这是个图形接收器，可以同时显示多个信号。接下来我们搜索并向流程图中添加 Signal Source（信号源）模块，和 Throttle（**节气门**）模块，有关这几个模块详细的说明将在之后的教程中详细讲解，现在只需知道此块会限制流程图的某些数据即可，以确保它不会占用 100％CPU 资源导致电脑直接卡到裂开。



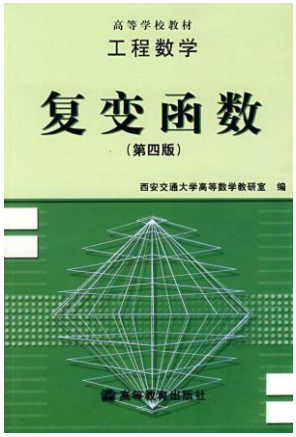
“**生成**流程图”，“ **执行**流程图”和“**终止**流程图”的快捷键分别为 F5，F6 和 F7。你可以在我们刚刚提到的工具架上点击这些按钮，或者直接按快捷键来进行相关的操作。当你按下生成流程图按钮之后，软件就会自动将你刚才绘制的流程图转化为一个 python 脚本文件。单击执行流程图按钮之后，就可以看到以下运行结果。



如果你不想运行了，只要点击终止流程图即可停止当前运行的程序。这样我们的第一个流程图就成功运行了。这是一个从信号源产生信号，经过限流器限制后输出到 time sink 进行接收并显示到屏幕上的操作。

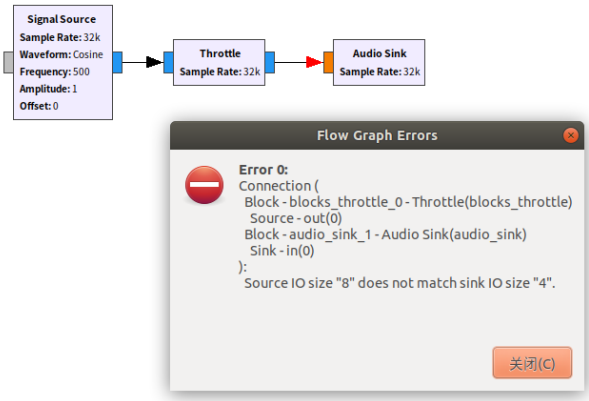
你可以注意到这里有两根数据曲线被绘制出来，他们都来自于 Data 0，蓝色的曲线为 Re（实部），红色部分为 Im（虚部）。

如果你根本不知道 Re 和 Im 是什么个玩意儿，那么我建议你先学习下我们电子通信类专业的一门必修课程《复变函数》，这将会对你的系统性学习产生很大的帮助。



（有意思的是这两个信号的相位差正好为$\frac{\pi}{2}$，这对于我们的零中频（Zero-IF）接收/发射机有至关重要的意义，不过这个咱们以后有机会再提。）

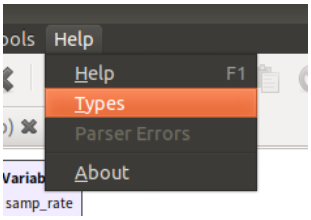
在这个流程图中，我们很轻松的就把所有的 block 连起来了，轻松的离谱你不觉得吗？没有出现任何头疼的问题或是错误。那么有没有会出现错误的情形呢？当然有，而且经常会有。

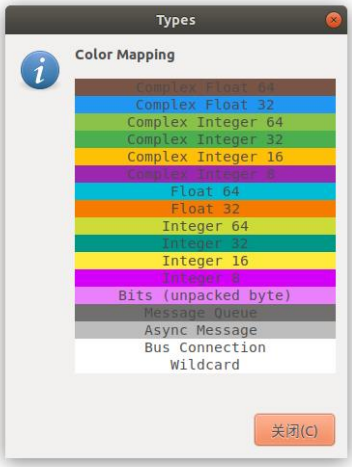


Source IO size "8" does not match sink IO size "4".

源 IO 大小“ 8”与接收器 IO 大小“ 4”不匹配。

这似乎是一个和数据类型有关的报错。既然出现了这个错误，那么就说明我们还没有搞懂框图输入输出的数据类型到底是个什么玩意儿。那么现在就让我们点击软件上方的 help，这里面有对于数据类型的说明。

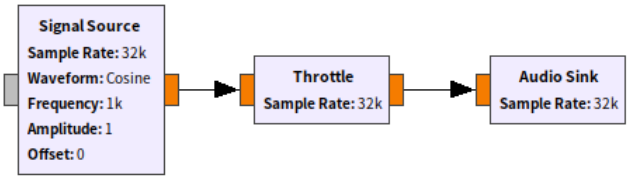




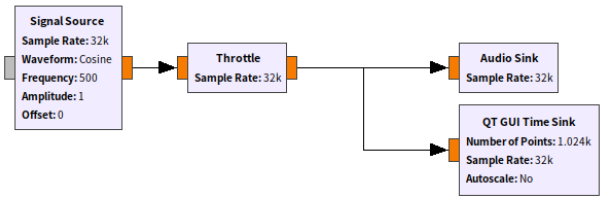
（最上面那个棕色的看的不是很清楚，不过这问题不大，你用鼠标把它选中高亮就能看清了。）

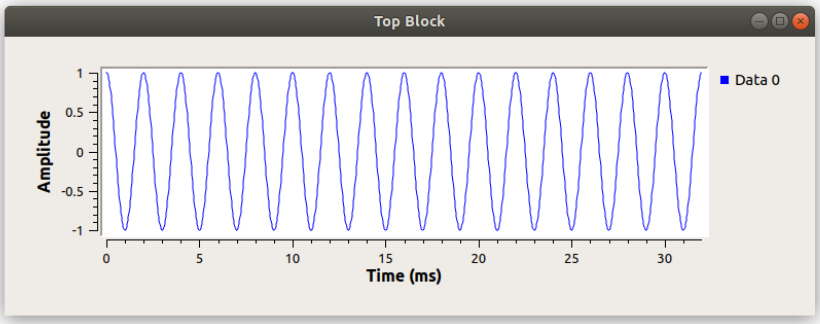
我们可以看到在许多编程语言中都可以看到的常见数据类型。在我们刚才搭建的流程图中，你可以注意到所有连接的模块端口均是蓝色的，这代表当前所传输的数据为 Complex Float 32 类型，这意味着它们同时包含实部和虚部，并且每一个都是 Float 32 类型。我们可以推断出，当“Time Sink 时间接收器”采集到这样一个 Complex 的数据类型时，它将在两个不同的通道上同时输出实部和虚部的图像，也就是我们刚才看到的红蓝两种颜色的图像了。

现在进入其 Signal Source 的属性面板，并更改**“输出类型”**参数，将**信号源**更改为**浮点型输出**。此时我们传输的数据流是一个普通的 32 位浮点数。



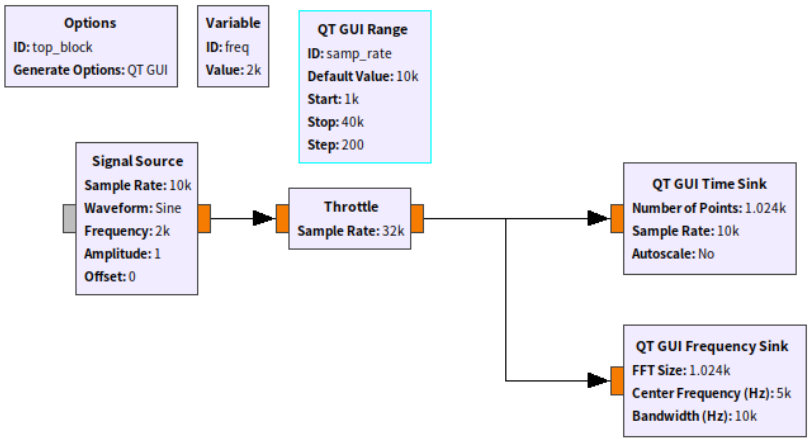
可以看到现在我们所有连接的点均变成了橘色，（当然 throttle 也要调整，别问我为什么它还是蓝色的），这也就说明了目前数据类型均匹配，当然刚才出现的报错也就消失了。





有同学发现 throttle 的输出连接了两个 block。不同的节点之间是可以支持多条同样的数据链路的，这是非常方便的一点，也是绝大部分图形化编程界面都具有的功能。可以注意到刚才的两条线此时变成了只有一条线，这是因为我们刚刚修改了数据类型。

现在让我们来尝试一些更复杂的框图吧。



运行结果如下：