1.实验名称及目的

1.1 实验名称

PX4环境部署与固件下载编译实验

1.2 实验目的

主要讲解如何部署PX4环境以及固件下载编译的方法

1.3 关键知识点

无

2.实验效果

能够正确判断环境是否安装正确

3.文件目录

例程目录：[安装目录]\RflySimAPIs\2.RflySimUsage\0.ApiExps\e6_PX4Build

文件夹/文件名称	说明
Setup	环境配置脚本

4.运行环境

4.1 软件要求

Windows 10及以上版本；RflySim工具链；飞控。

> ①：若使用Pixhawk 6X飞控，平台安装时的编译命令为：px4_fmu-v6x_default，推荐PX4固件版本为：1.12.3。其他配套飞控及编译命令请见：https://rflysim.com/doc/zh/1/Hardware.html

4.2 硬件要求

笔记本/台式电脑① 1台。

> ①：推荐配置请见：https://rflysim.com/doc/zh/HowToInstall.pdf

5.实验步骤

PX4开发环境的部署方法

我们已经配置有PX4环境的虚拟机压缩包，如有需要可从百度网盘中进行下载。链接:https://pan.baidu.com/s/1jwAHfk379qOLVB2BCiXYHQ?pwd=c8y5 提取码:c8y5，已完成配置的虚拟机账号及密码均为nvidia。

0.ApiExps\e6_PX4Build\setup中的文件，是从PX4源码中提取出来的环境配置脚本，由于PX4和APM的gcc_arm编译器版本有区别，本平台对脚本进行了拆分。

1）将setup文件夹拷贝到Ubuntu系统中。

2）在拷贝到Ubuntu系统的setup文件夹中打开终端，运行ubuntu.sh就能完成依赖环境的配置，运行命令为./ubuntu.sh。

3）等待上一条命令运行完毕后，同样在文件夹中打开终端，运行gcc_arm.sh，就能安装gcc_arm-10版编译器，并注册环境，运行命令为./gcc_arm.sh。

注：如果虚拟机安装配置有困难，也可以直接使用平台的WinWSL编译器来进行后续实验。

克隆官方源码

PX4 源代码存储在 Github 的 PX4/PX4-Autopilot 存储库中。

要将最新版本（“main”）获取到您的计算机上，请在终端中输入以下命令：

git clone https://github.com/PX4/PX4-Autopilot.git Firmware

之后，导航到Firmware目录

cd Firmware

切换到1.12.3版本源代码中：

git checkout v1.12.3

之后，在输入下列代码，就能自动下载源码和子模块。

git submodule update --init --recursive

注：后续会在百度网盘中上传，如果有网络问题，可以直接进行下载，之后在进行编译。

在下载完成后，在Ubuntu中进入Firmware目录，在当前目录中，打开终端，运行以下两行代码，就能够编译固件。

make px4_sitl

make px4_fmu-v6x_default

如果编译过程没有报错，说明环境配置正确。

成功的运行将以类似的输出结束：

-- Build files have been written to:
/home/youruser/src/PX4-Autopilot/build/px4_fmu-v4_default

[954/954] Creating
/home/youruser/src/PX4-Autopilot/build/px4_fmu-v4_default/px4_fmu-v4_default.px4

注意：现在编译最新源码，可能出现网络问题，因此这里选择编译1.12.3版固件。

固件装机使用的简单步骤

附加到 make 命令，通过 USB 将编译后的二进制文件上传到 autopilot 硬件。例如：

make px4_fmu-v4_default upload

成功的运行将以以下输出结束：

Erase : [====================] 100.0%

Program: [====================] 100.0%

Verify : [====================] 100.0%

Rebooting.

[100%] Built target upload

6.参考资料

源码文件夹的布局含义

PX4源码文件目录架构分析

文件夹名称	描述
boards	boards文件夹中是各个品牌、版本的飞控板的编译脚本， 其中px4文件夹装的是pixhawk的原生固件的编译脚本。
build	build文件夹是编译目标目录， 只有经过至少一次编译才会生成此文件夹， 内部存放的是源码经过编译生成的编译选项、中间文件、 目标文件等。编译不同版本的固件会生成不同的文件夹。
cmake	cmake文件夹是cmake编译工具的相关文件， 在开发中这个文件夹中的文件不需要修改。
Documentation	开发者文档目录，包括代码说明等。
launch	launch文件夹是仿真环境用到的文件，在gazebo中生成世界， 配置ros节点等。
msg	msg文件夹是存放uORB消息主题的地方， 实现PX4各进程之间通信就是用这些message。 我们可以在这个文件夹内定义自己的message。
ROMFS	ROM file system的缩写，是文件系统文件夹， 里面存放的飞控系统的启动脚本，负责系统初始化。内部的 px4fmu_common 文件夹中的 init.d 是关于 px4 系统初始上电启动的启动脚本，即一系列的启动过程和系统配置。 其中较为重要的部分在如下目录下： Firmware\ROMFS\px4fmu_common\init.d 如文件rcS、 rc.logging、rc.mc_apps、rc.sensors等。 rcS： 最先启动的脚本，负责挂载SD卡、启动uORB、配置系统参数等。 rc.logging：日志配置和启动代码。 rc.sensors：

文件夹名称	描述
	sensors驱动启动代码。 rc.mc_apps：启动上层应用（src/modules中的模块均在此启动），如 attitude_estimate、attitude_control、position_estimate和position_control等。
platforms	系统平台实现的文件，包括PX4采用的Nuttx操作系统的源代码。
src	src是源代码目录，是PX4的核心，非常重要。
drivers	pixhawk 硬件系统中使用的所有传感器的驱动代码，包括陀螺仪、加速度计、磁力计、气压计、GPS、光流等，也包括pwm输出、px4io控制、不同遥控器通信协议等功能的驱动。
examples	examples文件夹是官方给出的简单例程， 帮助新手开发者入门进行二次开发。
include	include文件夹是其他代码需要用到的头文件和库。
lib	标准库，有矩阵运算、PID控制、传感器校准等。
modules	modules文件夹是功能模块文件夹，也是二次开发主要的文件夹，包括姿态解算，姿态控制，位置估计，位置控制，指令控制，落地检测，传感器初始化等。 attitude_estimator_q：使用 mahony 互补滤波算法实现姿态结算。 commander： 整个系统的过程实现，包括起飞前各传感器的校准、 安全开关是否使能、飞行模式切换、 pixhawk硬件上指示灯颜色定义等。 ekf2： 使用扩展卡尔曼滤波器算法实现姿态和位置结算。 fw_att_control：固定翼飞机的姿态控制。 fw_pos_control_l1： 固定翼飞机的位置控制器。 land_detector： 使用land飞行模式降落的落地监测部分。 local_position_estimator：LPE 算法实现位置解算。 logger： 关于 log 日志的读写函数。 mavlink：和地面站通信的通信协议。 mc_att_control：姿态控制的算法实现，主要就是姿态的内外环 PID 控制，外环角度控制、内环角速度控制。 mc_pos_control： 位置控制的算法实现，主要就是位置的内外环 PID 控制， 外环速度控制、内环加速度控制。 navigator：任务， 失效保护和RTL导航仪。 sensors：关于各种传感器的相关函数。 vtol_att_control：垂直起降姿态控制器。
systemcmds	systemcmds文件夹是系统指令文件夹， 都是飞控终端支持的命令的源码，如top命令，reboot命令等。

文件夹名称	描述
templates	templates文件夹中是存放功能模块的代码模板，可以用于参考。
Tools	Tools文件夹下是一些工具，比如下载工具、仿真环境等。

PX4的核心架构

PX4 由两个主要部分组成：一是飞行控制栈（flight stack）
，该部分主要包括状态估计和飞行控制系统；另一个是中间件
，该部分是一个通用的机器人应用层，可支持任意类型的自主机器人，主要负责机器人的内部/外部通讯和硬件整合。

所有的 PX4 支持的 无人机机型
（包括其他诸如无人船、无人车、无人水下航行器等平台）均共用同一个代码库。
整个系统采用了 响应式（reactive） 设计，这意味着：

所有的功能都可以被分割成若干可替换、可重复使用的部件。

通过异步消息传递进行通信。

系统可以应对不同的工作负载。

VS Code下的C++编程开发环境

为了在VS Code下进行C++编程，你需要安装以下扩展和工具：

Visual Studio Code

C++扩展（例如：C/C++，Microsoft的C/C++扩展或者其他你选择的扩展）

C++编译器（例如：GCC或者Clang）

以下是一个简单的步骤指导和配置示例：

安装Visual Studio Code。

安装C/C++扩展。

安装C++编译器。

Windows: 安装MinGW或者Cygwin，并将其bin目录添加到系统环境变量PATH中。

Linux: 通常GCC/Clang已经安装。

macOS: 安装Xcode Command Line Tools，或者使用Homebrew安装Clang。

在VS
Code中，创建一个新的文件夹作为工作区，并添加你的C++源文件（例如：main.cpp）。

在VS Code中打开终端（通过Terminal > New
Terminal），并输入编译命令（例如，对于GCC，g++ -o output main.cpp）。

7.常见问题

Q:在进行px4环境部署以及获取源码失败?

N:检查网络是否正常，在可能的情况下，尝试使用代理再次运行。