马威廉 新手教程（翻译）

Mavlink 到底是个什么鬼？它是一种通讯协议。自从它问世的那一天起人们就被震惊到了。这个教程可以让你消除你大脑中对于 Mavlink 的神秘感并且搞懂它究竟是什么，而且更重要的是还可以搞懂这个该死的鬼东西到底是怎么运作的！ 我将要试图解释 Mision Planner（一种地面控制站） 是怎么与 APM/PX4（飞控） 相互通讯的。这将是一个很好的拓展学习，并且帮助您爆发出您体内积蓄已久的程序员小宇宙（如果还没有爆发的话）。

这个教程会假定您：

1 很菜鸡🙃 我之前也和您一样，但现在不是了

2 你拥有基础的 c 语言编程能力（其实 c/c++/c#/java 都差不多啦）。如果你是个老程序员了，那就假装自己很基础好了 XD

3 你在很认真地学习并准备因此而丢失一些睡眠

Mavink，我要进来了！

Mavlink 的信息（message），我们就叫它“msg”好了，其实就是你的地面站（比如 Mission Planner）编码的一段数据流，它会被通过无线电遥测或一系列 usb 接口的途径被发送给飞控（例如 APM/PX4）。（但是要注意的是，无线与有线方式不能在同时使用。当他们同时被使用时，USB 接口会作为首选项而无线就会被忽略）。
我们在这里说的“编码”并不是一种特别晦涩的含义，它只是把你要发送的数据以某种格式来打包，并且加上一些用于纠正的信息，然后通过某个频道一字节一字节地传送。

Msg 的结构

每一个 Mavlink 的数据包都有 17 字节的长度，它们的结构如下：

讯息长度=17 字节=6 字节的抬头 + 9 字节的数据 + 2 字节的纠错码

六字节的抬头
0.通讯开始标志，在 1.0 版本中通常是 0xFE
1. 信息的长度（9 是最高）
2.顺序标号——一个从 255 到 0 的次序
3.系统 ID——哪一个系统在发送这个讯息
4.组件 ID——这个系统的那个组件在发送这个讯息
5.讯息 ID——这个讯息是什么（那种类型，要用它干啥之类的）

自定义长度的有效载荷
（是以 1 字节（byte）等于 8 位（bit）为标准的，范围是 0-255）
这就是我们真正打算关注的讯息

错误校验码
用于诊断在数据的传输过程中有没有出错

软件系统要做的事是确认这段信息是否有效（通过确定纠错码来确定这段讯息是否有损坏，如果是的话就丢弃这段讯息）。这就是为什么无线传输的波特率要设定在 57600 而不是 115200。波特率越低，信号在传输过程中越不容易遭受干扰和出错，尽管地面站的刷新频率会低一些。如果您想用 Mavlink 来实现一个很远距离的传输，进一步地调低您的波特率也许是一个不错的方法。但是请注意，理论上，经过测试的 57600bps 的波特率在使用 3DR 无线遥测电雷达时已经足够完成一英里（约 1.6 公里）范围的电波覆盖。请回想一下我们在中学学过的信噪比的概念。

现在，阅读了以上章节后，我们要了解这些内容：

A）**系统 ID**（就是讯息的来源处）这是发送给飞控的讯息来源处标识。软件会进行一个常规检查来确定消息是不是给它的。
B）**组件 ID** （就是在某一个系统中的某一个组件或子系统）在一个系统中的某一个次级系统。如果目前没有次级系统的话，就可以不用它。
C）**讯息 ID** 说明这段讯息到底是什么类型的内容。在这个教程里，我们叫它“主要讯息“。
D）**有效载荷**（实际上的数据） 这个就是我们真正要发送到讯息！这个就是你真正想要的！

Mavlink 究竟如何工作？

Mavlink 就是一段信息。马威廉 (Mavlink) 的全称是“微型空中载具通讯链路 (Micro Aerial Vehicle Link）“，虽然名字说的很清楚，但光看名称并不能反映全部，因为它也能被用在智能车上。它叫这个名字是因为它最开始是为飞行器开发的（如果原作者猜的对）。

这个信息其实就是一段包含“固定字节数“的数据包。飞控系统可以获得这个数据流，它从硬件接口获取（可以是串口通讯也可以是无线电），并使用软件程序解码。这个信息包含了”有效载荷“，就是我们真正要从中提取的部分。

我们当然主要关注“有效载荷“的部分，但是，和”有效载荷“一起被发送的”讯息 ID“记载了讯息的用处。但是在此之前，我们下了解一下代码是如何编译 Mavlink 的数据流的：

1） 我们有一个模式叫做 “handlemessage“。这是我们需要了解的。我们可以从 Mavlink 的配置文件中找到它，
它基本上就是在询问数据包：“嘿，你是哪来的包？是要传给我的讯息还是要骇进我系统的？在我接受你之前，我们先检查一下你的**系统 ID** 和**组件 ID。**“ 所有使用 Mavlink 协议发送的讯息都一定会包含**系统 ID** 和**组件 ID**
例子：你的地面站、你的四轴通常会被算作一个“系统“，有相同的**系统 ID**。 **组件 ID** 是分配给”子系统“的，比如说你四轴上面搭载的飞控算作一个”子系统“。
提示：在目前，系统 ID 和组件 ID 在硬编码（可以理解为固定不变的版本代号）上是相同的。

所以，如果你有一个遥感雷达，一个无人机以及飞控，这些都是一个 ID 编号，别想那么多，飞就完事了。这个**系统 ID** 和**组件 ID** 其实是是为无人机机群准备的，不同编队的无人机使用不同的**系统 ID**——这就是这个通讯协议的未来！

2） 实际有效的数据会从信息中被抽取出来然后放在一个数据包（packet）里面。一个数据包就是基于一种“信息类型“而决定的数据结构来打包的。接下来我们就不再使用“信息“这个词了。这个信息已经被处理掉了。现在我们关心的是”包“里的东西，基本上就是被打包好的”原始数据“。

3） 这个数据包会被放在一个“恰当的数据结构“里面。数据结构的有很多中，比如有记录航向、翻滚等信息的数据、有 GPS 的数据、有网络频道传输的数据，数据结构是把相似的东西结合在一起，让它们变得“模块化”，容易被（机器）所理解。他们在发送端和接收端都是“100%完美相似”的。如果不是，你的无人机就要出问题。

这就是为什么说我们 Mavlink 的图形化编译窗口（GUI）可以来拯救你。为了生成这些数据结构，您不必挣扎在一行行晦涩难懂的代码中。当然，虽然不能一点代码都不沾，但是已经减少了好多了。

原作者希望以上的内容对于大家来说比较好理解。如果还没理解的话，官方建议是再看一遍。没关系，大家都是新手过来的嘛。(*^_^*)

好了，下面让我们来搞搞真意思。我们通过 Mavlink 进行双向的数据收发吧。

→ 从地面站（Ground Control Station，GCS）到 APM/PX4 （两种常见飞控），或者，从 APM/PX4 到地面站。提示：当我说“地面站”时，指的是 Mission Planner、QGroundControl，DroidPlanner 或者你自己个性化的用来与四轴相互沟通的工具。

从地面站到你的四旋翼

目前，我们知道每段信息（我们把它叫做“包”，那里面有我们要的信息），有一个“**信息 ID**”，和“有效载荷”被合适地安放在了一个数据结构中。我们会打开“main message”类（即 Mavlink_MSG_ID，“信息 ID”），一旦检测到该消息，我们就会在这个消息上施展魔法，比如将接收到的信息存储到永久内存（EEPROM）中，或是我们进行希望对其进行的任何操作。

截至 2013 年 11 月，在 Arducopter 的最新 3.0.1（发布可选版本）中，您可以找到这些参数，我尝试列出所有可能的 MavLink 消息的“main message”ID。

请注意，在每个“主信息”类别中（如下面粗体部分），您会发现属于这个目录的“子信息“，它们基本上属于与有效载荷信息（真正的内容）密切从属的内容，以及关于如何处理有效载荷信息的提示。就像”自行车“目录下面会有”铃木“、”雅马哈“等等。原作者只列出来了全部的”主信息“目录以及一部分”子信息“目录。并且鼓励大家自己探索更多相关的细节以锻炼自己的能力🙃。

MAVLINK_MSG_ID (主消息块)

1） MAVLINK_MSG_ID_HEARTBEAT: //0

a. 这是坠重要的信息！地面站会持续向你的飞控发送这个信息来确定是否能连接上（每一秒发一次）。这是为了确保当您更新某些参数时，MP 与 APM 同步。如果错过了许多心跳（HEARTBEAT），就会（可以）触发故障保护，无人机会着陆、继续执行任务或返回发射（也称为 RTL）。可以在 MP 中的 配置/设置 中的故障保护选项下选择 启用/禁用 故障保护。故障保护的设置取决于您自己。但是我们都不能停止心跳吧，所以这个名字取得还是很形象的。

2） MAVLINK_MSG_ID_REQUEST_DATA_STREAM: //66

a. 传感器、无线电频道、GPS 位置信息、高度，额外的信息通路。

3） MAVLINK_MSG_ID_COMMAND_LONG: //76

a. 循环徘徊、RTL、着陆、开始任务、装备/解除装备、重启

4） SET_MODE: //11

a. 实例：set_mode(packet.custom_mode)

5) MAVLINK_MSG_ID_MISSION_REQUEST_LIST://43

a. 总航点：command_total 参数变量。它将保存存在的路径点总数（对于多无人机系统来说，原点位置除外）

6）MAVLINK_MSG_ID_MISSION_REQUEST: //40

a．一系列 MAV_CMD 值枚举成员的值，比如: (MAV_CMD_)CHANGE_ALT, SET_HOME, CONDITION_YAW, TAKE_OFF, NAV LOITER_TIME

7）MAVLINK_MSG_ID_MISSION_ACK: //47

a.关闭路径点的发送

8）MAVLINK_MSG_ID_PARAM_REQUEST LIST: //21

a. count_parameters (Count the total parameters)

9) MAVLINK_MSG_ID_PARAM_REQUEST_READ: //20

接收和解码参数（有关参数名称和 ID）

10) MAVLINK_MSG_ID_MISSION_CLEAR_ALL：//45

a. 当您使用 MP 的飞行数据屏幕（flight data screen），并使用鼠标菜单说“清除任务”时，便可以使用它。 它会清除 APM / PX4 中的 EEPROM(可编程存储器)。

11) MAVLINK_MSG_ID_SET_CURRENT //41

a. 用于在任务执行中更改活动命令。例如，您点击 MP 的谷歌地图页面的某个点，然后点击“飞到此处”时。

12) MAVLINK_MSG_ID_MISSION_COUNT: // 44

a. 保存路径点的总数（对于多无人机系统，不包括原点）

13) MAVLINK_MSG_ID_MISSION_WRITE_PARTIAL_LIST: //

a. 只需保留一个全局变量，说明 APM 现在正在接收命令。 这是为了避免在设置重要参数时执行其他 MavLink 操作。

14) MAVLINK_MSG_ID_SET_MAG_OFFSETS: //151

a. 将指南针校准后的 mag_ofs_x，mag_ofs_y，mag_ofs_z 数据存入到 APM / PX4 的 EEPROM 中。 Mission Planner（MP）会自动执行此操作，或者您也可以通过转到“软件配置”下的“完整参数列表”来执行此操作。

15) MAVLINK_MSG_ID_MISSION_ITEM: //39

这是一个有趣的部分。 该消息包含用于执行实时操作的子消息。 就像设置航路点和高级功能一样。这是这样的，如下所示：

a. 接收一个从地面站发来的航点数据并且储存在 APM/PX4 的 EEPROM 中

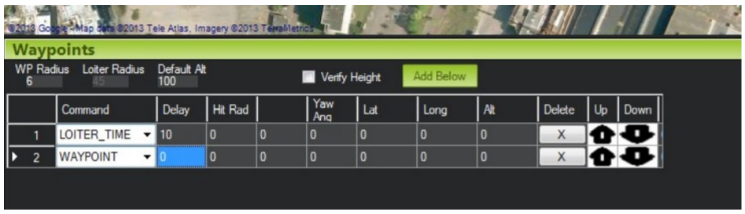
b. 发送 4 个参数（例如 Delay，HitRad，-和 Yaw Angle）伴随 LOITER TIME（作为动作 ID）+（Lat（经度），Long（纬度），Alt（高度）：定义对象在空间中的 3D 位置）。 这些参数在代码+选项中定义为 枚举数组（注：高度是以原始的高度为参照的）。 每个命令（或 ID）可能具有不同的参数。如果没有为此 ID 定义任何参数，MP 会在列表中显示“空白”。作为总结，随每个动作发送的有趣的参数是： 4 个参数+ ID（要执行的动作）+（纬度，经度，高度）。 请注意，这 4 个参数可以是某种自定义操作，例如相机设置，相机触发，游荡时间等。

c. 如下图中 Mission Planner 所示，每个 ID 定义了一个航点（AFAIK）。

LOITER_TIME， LOITER_UNLIMITED， WAYPOINT 都是与其他参数（LATITUDE，LONGITUDE 和 ALTITUDE）一起发送的航路点数据，因为每个参数的数据模式都保存为 APM / PX4 中的航路点形式。

d. 注意：在目前的版本中，“高度”总是相对于原始高度而言的（总是!!!）

e. 您可以在 Common.xml 文件中定义”这些操作”，并使用 Python GUI 编译器生成 APM / PX4 使用的代码。 我稍后再讲，或在论坛上问我如何做。 我提到过，您可以为“4 个参数“添加自己的感兴趣的参数。



f. 当 APM 收到此”主要”命令（MAVLINK_MSG_ID_MISSION_ITEM）时，它将从 MavLink 数据包中读取 ID，并根据 ID 进行 switch（case），即选择与执行，操作。 例如：

i. Loiter turns, Set home, Loiter time, Repeat servo. Set servo,等

16) MAVLINK_MSG_ID_PARAM_SET: //23

a. 设置参数。 请记住，我们可以在 MP 中为参数设置一个值（比如，“Full Parameter List”）； 这就是我们要做这件事时发生改动的地方。 APM 发送数据值集以进行确认。 您是否看过 MP 提示您“设置参数失败”？ 此外，APM / PX4 会同时记录此值，以供我们离线分析。

17) MAVLINK_MSG_ID_RC_CHANNELS_OVERRIDE: //70

a. 覆盖 HIL（Hardware-In-Loop-Simulation，硬件在环仿真）的 RC Chanel 值，或者通过 GUI 完成地面站对开关位置的控制（不过我还没有尝试过!）!

18) MAVLINK_MSG_ID_HIL_STATE: //90

a. 用于 HIL 仿真。 这是您的旋翼/固定翼的虚拟实现。

19) MAVLINK_MSG_ID_DIGICAM_CONFIGURE: //

20) MAVLINK_MSG_ID_MOUNT_CONFIGURE: //

21) MAVLINK_MSG_ID_MOUNT_CONTROL: //

22) MAVLINK_MSG_ID_MOUNT_STATUS://

a. 到目前为止，顾名思义，它可以配置由用户设置的适当命令设置。

23) MAVLINK_MSG_ID_RADIO, MAVLINK_MSG_ID_RADIO_STATUS: //

a. 查看 无线电遥测/USB 的数据包发送频率，并在信号强度低于预期或错误率越来越高时自动调整发送与接收数据包之间的延迟时间。 就像自适应软件流控制一样。 查看 C ++中的 at_mavlink_radio_t（APM 代码）或 mavlink_radio_t（C＃，Mission Planner）代码。 两者都定义为数据结构。

从四旋翼到地面站再到四旋翼

好吧，我觉得，这更加有趣，但容易得多。 事实是，地面站只是您和无人机之间的中间商，它取得无人机反馈的数据，并显示在地面站上。

如果您能找到 Arducopter.pde 文件，请查看代码的这一部分：

```
static const AP_Scheduler::Task scheduler_tasks[] PROGMEM = {
. . .
. . .
{ gcs_send_heartbeat,  100,     150 },
{ update_notify,         2,     100 },
{ one_hz_loop,         100,     420 },
{ gcs_check_input,       2,     550 },
{ gcs_send_heartbeat,  100,     150 },
{ gcs_send_deferred,     2,     720 },
{ gcs_data_stream_send,  2,     950 },
. . .
. . .
. . .
```

别害怕这易如反掌。 这是实时系统的概念（real-time systems concept）起作用的地方。 我们希望某些任务只花费特定的的时间，如果到那时还没有完成，那么我们就不要继续进行。

第一个参数是函数（功能）名称，
第二个是``应该花费的时间''（以 10 毫秒为单位，即 2 表示 20 毫秒，即 50Hz，即此功能每秒运行 50 次）。
第三个参数是“该函数（功能）不应超过的最长运行时间”。

我认为这些道理都是不言而喻的！ 您在那里看到的每个函数（功能），其命运都是注定的，它正好运行规定的那么长时间。 这就是为什么在这些淘气的机器上使用实时系统（Real-timesystems）是安全的，使其行为是可预测的而非不可预测的！

所有这些函数（功能）都是为您精心挑选的，以便让您知道它们与地面站刷新有关。 简单地，找寻每个函数（功能）的定义，您将被带到 GCS_Mavlink.pde，地面站的通讯实际在这里发生！

最有趣和最重要的是下面这个:

```
/*
 *  send data streams in the given rate range on both links
 */
static void gcs_data_stream_send(void)
{
    gcs0.data_stream_send();
    if (gcs3.initialised) {
        gcs3.data_stream_send();
    }
}
```

它做的就是通过网路连接来发送数据（gcs0 是指 USB，gcs3 是指无线电遥测）。

如果进一步深入，您会知道我们会将数据结构发送回地面站进行显示。

例如： 当您用手移动无人机并查看 Mission Planner 的 HUD 屏幕时会发生什么？ 您会看到无人机在屏幕上移动。 我们在每个单位时间内都获得"姿态"数据（俯仰，滚动和偏航）。 同样，我们会有 IMU 数据，GPS 数据，电池数据等。

因此，如果您有任何问题，请在论坛上向我发送消息，我将尽我所能回答。这个协议给了我们一个新的可能性，它开创了一个通向无限的新起点。 不断发展，不断更新!!

Shyam Balasubramanian
(Embedded UAV Researcher and Developer),
Netherlands
Shyambs85@gmail.com

（原作者通讯地址）

译者：Endless

业余渣翻，不喜勿喷 2020-09-27

支持开源，开放版权

合肥工业大学航模队

HBSF 字幕组