1. 实验名称及目的

1.1. 实验名称

MAVLink 数据发接实验

1.2. 实验目的

MAVLink （Micro Air Vehicle Link）是一种用于小型无人载具的通信协议，于 2009 年首次发布。该协议广泛应用于地面站（Ground Control Station，GCS）与无人载具（Unmanned vehicles）之间的通信，同时也应用在载具上机载计算机与 Pixhawk 之间的内部通信中，协议以消息库的形式定义了参数传输的规则。MAVLink 协议支持无人固定翼飞行器、无人旋翼飞行器、无人车辆等多种载具。本实验将基于“*\PX4PSP\RflySimAPIs\7.RflySimExtCtrl\1.BasicExps\e0_ExtAPIUsage\1.MavLinkPackSimulink”实验中建立的两个模块，模拟发送 MAVLINK_MSG_ID_HIL_ACTUATOR_CONTROLS 消息并进行接收消息。

1.3. 关键知识点

无

2. 实验效果

实现模拟的 MAVLINK_MSG_ID_HIL_ACTUATOR_CONTROLS 消息发送与接收。

3. 文件目录

例程目录：[安装目录]\RflySimAPIs\6.RflySimExtCtrl\1.BasicExps\e0_ExtAPIUsage\2.MavlinkCodeDecode\

文件夹/文件名称	说明
mavlink	MAVLink 的 C++源代码。
MavlinkCodeDecode.slx	MAVLink 消息发接模块模型文件。

4. 运行环境

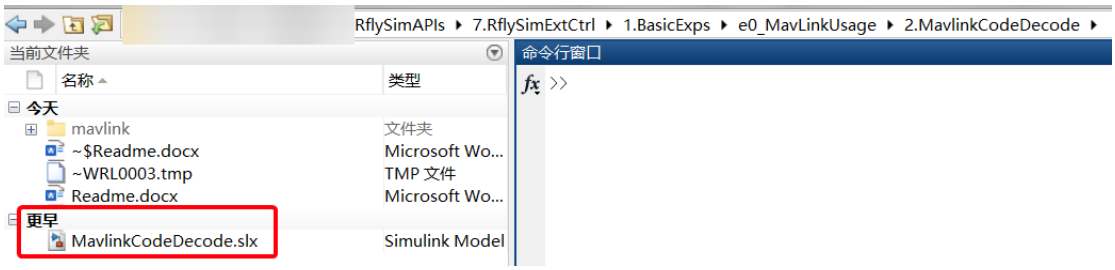
序号	软件要求	硬件要求	
		名称	数量
1	Windows 10 及以上版本	笔记本/台式电脑①	1
2	RflySim 工具链		
3	MATLAB 2022b 及以上版本		

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

5. 实验步骤

Step 1：

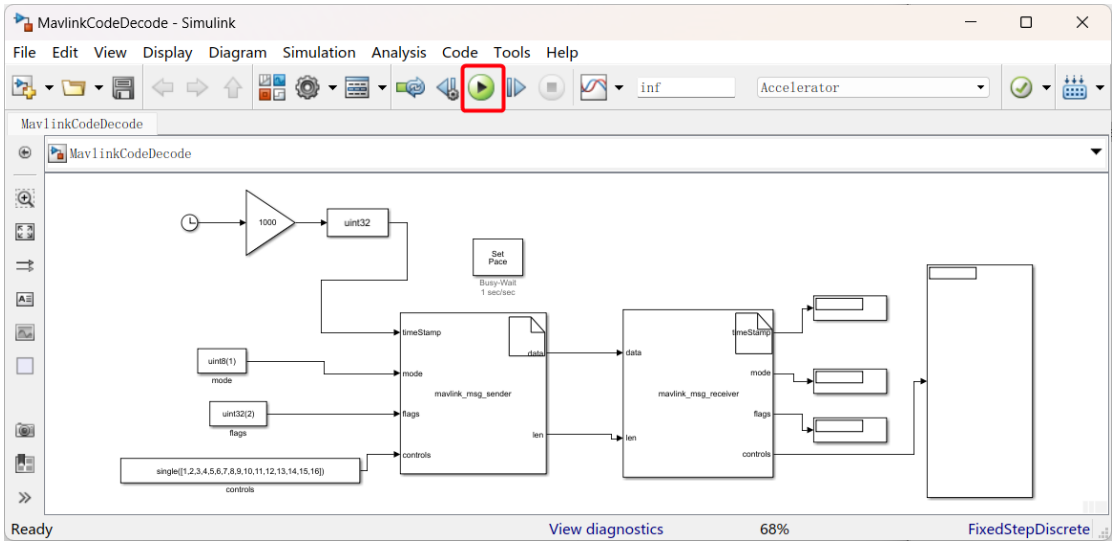
打开 MATLAB 软件，打开 MavCmdDefines.m 并运行，进行设置。再双击运行 MavlinkCodeDecode.slx 文件，如下图所示。



如果出现 mavlink.h 的相关报错，检查 mavlink 文件夹是否解压到该文件夹下的正确路径。

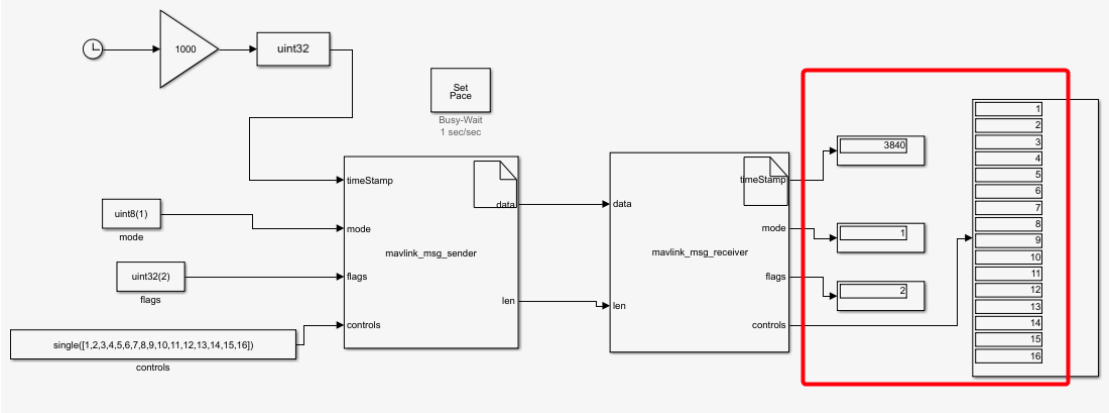
Step 2：

在 Simulink 中运行该文件。



Step 3：

运行完成后，即可在看到相关数据显示



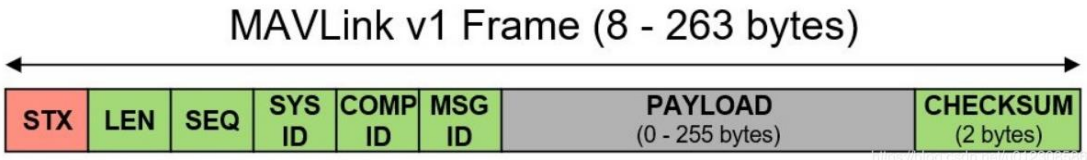
6. 参考资料

[1]. MAVLink 官方使用文档网站：https://mavlink.io/en/messages/common.html

[2]. MAVLink 源码：https://github.com/mavlink/mavlink

[3]. 基于 MAVLink 的 QGroundControl 地面站源码：https://github.com/mavlink/qgroundcontrol

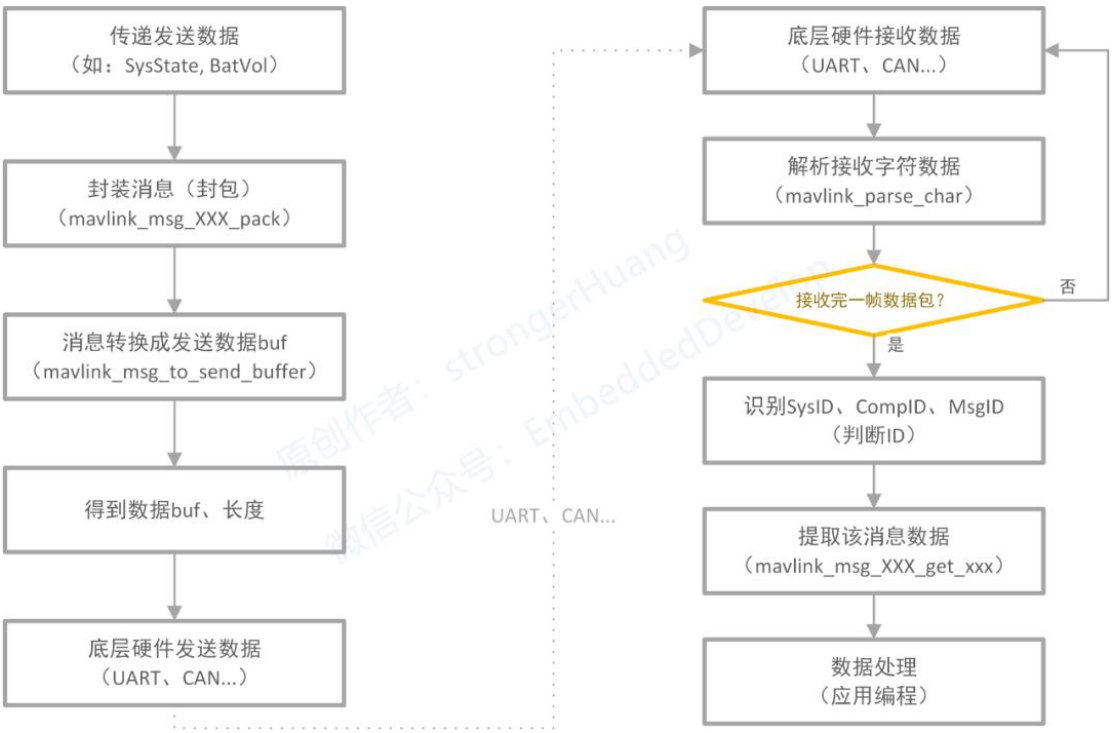
[4]. MAVLink 1 的数据包格式如下：



[5]. MAVLink 2 的数据包如下：

区域	名称	索引	长度	含义	取值
STX	起始标识	0	1	标识新消息的开始，用于消息识别、解析	254
LEN	负载长度	1	1	记录负载信息的长度	N
SEQ	序列码	2	1	消息发送序列码，用于通信可靠性检验	0-255
SYS	系统ID	3	1	发送该消息系统的系统ID	0-255
COMP	组件ID	4	1	发送该消息系统组件的组件ID	0-255
MSG	消息ID	5	1	标识该消息的种类	0-255
PAYLOAD	负载信息	6	1	消息内部负载信息	
CKA	校验位A	N+6	N	CRC校验位	自动
CKB	校验位B	N+7	1	CRC校验位	自动

[6]. 解析原理：**读取：**所有字节流存入 buffer，依次读取 buffer 中的字节数据，遇到 STX 标志位（MAVLink v1 的标志位是 0xFE，v2 的标志位为 0xFD）开始识别一条消息直到消息尾部，如果消息校验正确则将消息发送给处理程序。**发送：**按上一页 PPT 将消息转换成字节流。



[7]. 接收解析源码分析

给定一定长度的字节流 buffer，长度为 length，通过下列脚本解析，每解析出一个 mavlink 数据包就执行 onMavLinkMessage 函数

```
     for(int i = 0 ; i < length ; ++i){
            msgReceived = mavlink_parse_char(MAVLINK_COMM_1, (uint8_t)buffer[i], &message, &status);
            if(msgReceived){
         emit onMavLinkMessage(message);
            }
      }
```

其中：

```
    void onMavLinkMessage(mavlink_message_t message);
```

是得到一个 MAVLink 消息包后的处理函数，需要根据这个消息的 ID 来识别当前包的用途（心跳包，GPS 位置，姿态等），并提取出感兴趣的数据。

解析函数实现如下，根据 message.msgid 跳到对应的_decode 函数，解码出数据：

```
    void onMavLinkMessage(mavlink_message_t message){
        switch (message.msgid){
          case MAVLINK_MSG_ID_GLOBAL_POSITION_INT:{
            mavlink_global_position_int_t gp;
            mavlink_msg_global_position_int_decode(&message, &gp);
            outHilData.time_boot_ms = m_LastReceiveMavMsg;
            outHilData.GpsPos[0]=gp.lat;
            outHilData.GpsPos[1]=gp.lon;
            outHilData.GpsPos[2]=gp.alt;
            outHilData.relative_alt = gp.relative_alt;
            outHilData.GpsVel[0]=gp.vx;
            outHilData.GpsVel[1]=gp.vy;
            outHilData.GpsVel[2]=gp.vz;
            outHilData.hdg = gp.hdg;
```

```
            break;
        }
```

[8]. 发送源码解析 — 发送一条 mavlink_hil_actuator_controls 消息：

```
void sendHILCtrlMessage(uint8_t modes, uint64_t flags, float ctrl[])
{
    mavlink_hil_actuator_controls_t hilctrl;
    hilctrl.mode = modes;
    hilctrl.flags = flags;
    for(int i=0;i<16;i++){
        hilctrl.controls[i]=ctrl[i];
    }
    mavlink_message_t mess;
    mavlink_msg_hil_actuator_controls_encode(SystemID, TargetCompID, &mess, &hilctrl);
    char buffer[500];
    memset(buffer,0,500);
    unsigned int length = mavlink_msg_to_send_buffer((uint8_t*)buffer, & mess);
    udp.writeDatagram(buffer,length);//通过 UDP 或者串口将 buffer 发送出去即可
}
```

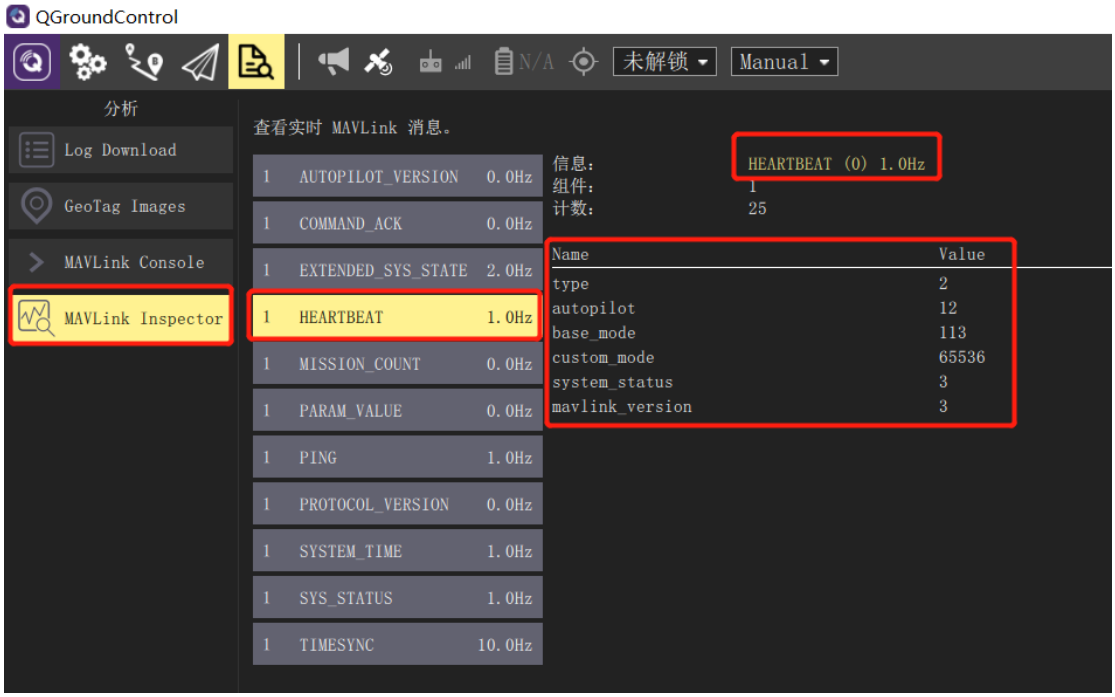
[9]. MAVLink 消息包的 ID 列表：https://mavlink.io/en/messages/common.html

HEARTBEAT (#0) 心跳包，ID=0号消息

[Message] The heartbeat message shows that a system or component is present and responding. The type and autopilot fields (along with the message component id), allow the receiving system to treat further messages from this system appropriately (e.g. by laying out the user interface based on the autopilot). This microservice is documented at https://mavlink.io/en/services/heartbeat.html

Field Name	Type	Values	Description
type	uint8_t	MAV_TYPE	Vehicle or component type. For a flight controller component the vehicle type (quadrotor, helicopter, etc.). For other components the component type (e.g. camera, gimbal, etc.). This should be used in preference to component id for identifying the component type.
autopilot	uint8_t	MAV_AUTOPILOT	Autopilot type / class. Use MAV_AUTOPILOT_INVALID for components that are not flight controllers.
base_mode	uint8_t	MAV_MODE_FLAG	System mode bitmap.
custom_mode	uint32_t		A bitfield for use for autopilot-specific flags
system_status	uint8_t	MAV_STATE	System status flag.
mavlink_version	uint8_t_mavlink_version		MAVLink version, not writable by user, gets added by protocol because of magic data type: uint8_t_mavlink_version

[10]. QGC 地面站查看 MAVLink 消息在 QGC 的 MAVLink Inspector 页面中可以浏览 Pixhawk 发送的所有 MAVLink 包，查看各个包的频率以及具体数值。


QGroundControl
未解锁
Manual
分析
Log Download
GeoTag Images
MAVLink Console
MAVLink Inspector
查看实时 MAVLink 消息。
1 AUTOPILOT_VERSION 0.0Hz
1 COMMAND_ACK 0.0Hz
1 EXTENDED_SYS_STATE 2.0Hz
1 HEARTBEAT 1.0Hz
1 MISSION_COUNT 0.0Hz
1 PARAM_VALUE 0.0Hz
1 PING 1.0Hz
1 PROTOCOL_VERSION 0.0Hz
1 SYSTEM_TIME 1.0Hz
1 SYS_STATUS 1.0Hz
1 TIMESYNC 10.0Hz
信息: HEARTBEAT (0) 1.0Hz
组件: 1
计数: 25
Name Value
type 2
autopilot 12
base_mode 113
custom_mode 65536
system_status 3
mavlink_version 3

7. 常见问题

Q1：****

A1：****