1. 实验名称及目的

传感器数据读取实验：通过 RflySim 的底层开发接口，可获取的传感器数据包含磁力计、加速度计、陀螺仪、气压计和时间戳以及 GPS 数据等信息。本实验将进行上述传感器部分数据的获取，以此思路可订阅更加多样的传感器数据。

2. 实验原理

飞控内部使用 NuttX 实时 ARM 系统， 而 uORB 对于 NuttX 而言，它仅仅是一个普通的文件设备对象，这个设备支持 Open、Close、Read、Write、Ioctl 以及 Poll 机制。通过这些接口的实现，uORB 提供了一套“点对多”的跨进程广播通讯机制， “点”指的是通讯消息的“源”，“多”指的是一个源可以有多个用户来接收、处理。而“源”与“用户”的关系在于，源不需要去考虑用户是否可以收到某条被广播的消息或什么时候收到这条消息。它只需要单纯的把要广播的数据推送到 uORB 的消息“总线”上。对于用户而言，源推送了多少次的消息也不重要，重要的是取回最新的这条消息。也就是说在通讯过程中发送者只负责发送数据，而并不关心数据由谁接收，也不关心接收者是否能将所有的数据都接收到；而对于接收者来说并不关心数据是由谁发送的，也不关心在接收过程中是否将所有数据都接收到。

3. 实验效果

实现磁力计、加速度计、陀螺仪、气压计和时间戳以及 GPS 部分数据读取。

4. 文件目录

文件夹/文件名称	说明
SenorDataGet.slx	传感器数据读取文件。
PX4ExtMsgSender.slx	飞控 uORB 消息监听程序

5. 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 平台免费版及以上版本	Pixhawk 6C 或 Pixhawk 6C mini②	1
		遥控器③	1
		遥控器接收器	1
		数据线、杜邦线等	若干

①：推荐配置请见：https://doc.rflysim.com

②：须保证平台安装时的编译命令为：px4_fmu-v6c_default，固件版本为：1.13.3。其他配套飞控请见：http://doc.rflysim.com

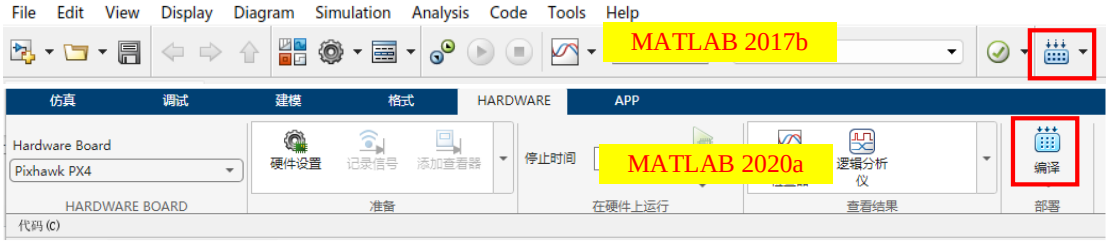
③：本实验演示所使用的遥控器为：天地飞 ET10、配套接收器为：WFLY RF209S。遥控

器相关配置见：..\e11_RC-Config\Readme.pdf

6. 实验步骤

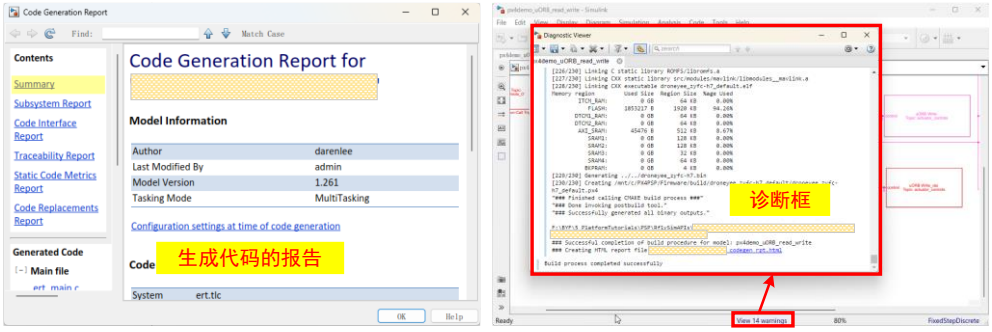
Step 1：

打开 MATLAB 软件，在 MATLAB 中打开 SenorDataGet.slx 文件，在 Simulink 中，点击编译命令。



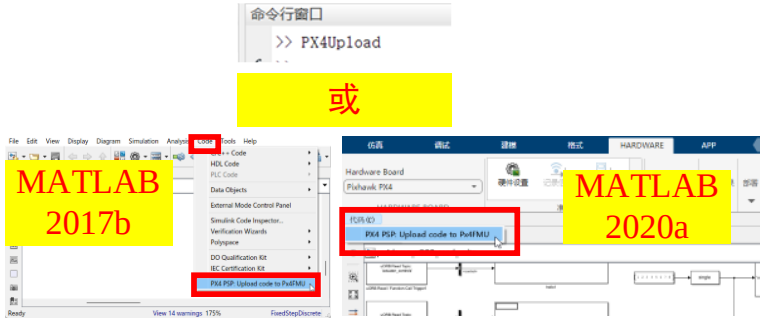
Step 2：

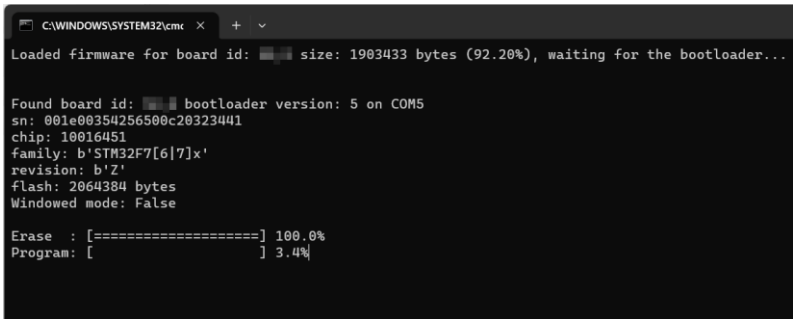
在 Simulink 的下方点击 View diagnostics 指令，即可弹出诊断对话框，可查看编译过程。在诊断框中弹出 Build process completed successfully，即可表示编译成功，左图为生成的编译报告。



Step 3：

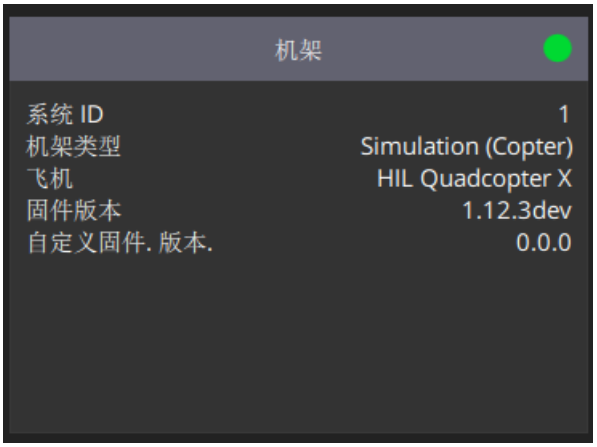
用 USB 数据线链接飞控与电脑。在 MATLAB 命令行窗口输入：PX4Upload 并运行或点击 PX4 PSP：Upload code to Px4FMU，弹出 CMD 对话框，显示正在上传固件至飞控中，等待上传成功。





Step 4：

打开 QGrounControl 软件，设置机架为“HIL Quadcopter X”之后，关闭 QGC。



Step 5：

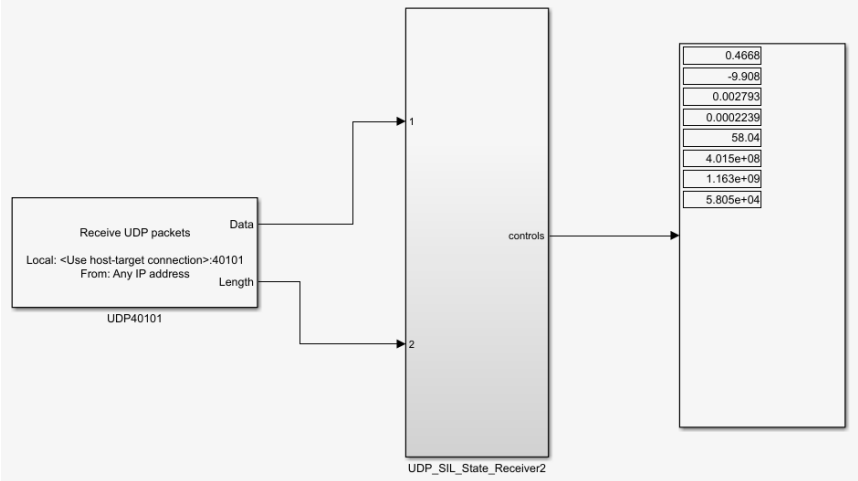
打开 CopterSim 软件，进行如下图所示设置之后，其中飞控选择一栏需要选择本电脑所识别到的飞控 COM 端口号。设置完成之后，点击开始仿真。

等待 CopterSim 显示初始化完成。

Step 6：

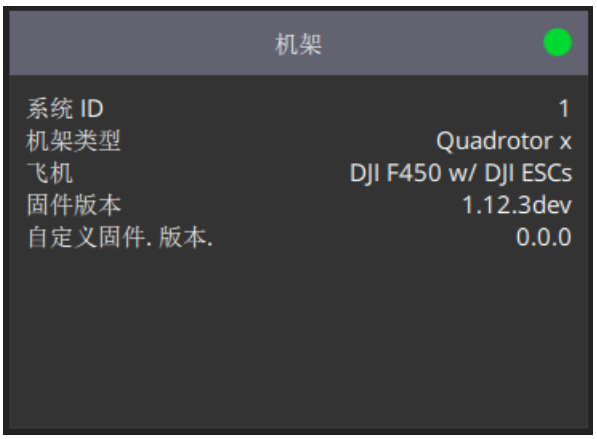
在 CopterSim 运行的运行过程中，在 MATLAB 中打开 PX4ExtMsgSender.slx 监听程序，点击运行即可监听到写入的 uORB 消息，如下图所示：

注：此时获取到的为 CopterSim 软件中生成的传感器数据。若想获取到飞控中传感器数据请见 Step 7~8。



Step 7：

上述 Step4~6 为在硬件在环仿真时，获取各种传感器的操作步骤；而在实飞时，需要打开 QGrounControl 软件，切换机架为“DJI 450 w/DJI ESCs”如下图所示。

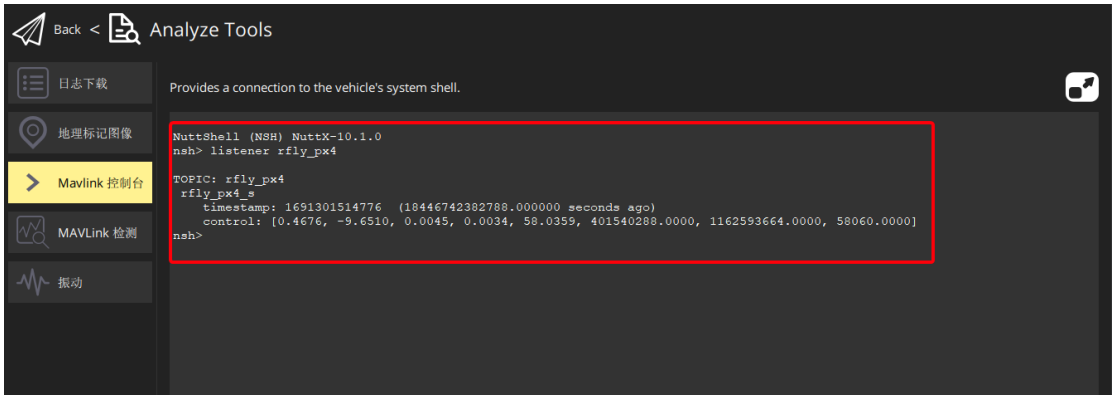


Step 8：

在 QGroundControl 软件初始界面下，点击左上角 Logo 在弹出的对话框中，选中 Analyze Tools，在 Mavlink 控制台中输入：

```
listener rfly_px4
```

即可得出所订阅的 uORB 消息，如下图所示：



6、参考资料

[1]. **本实验硬件在环仿真数据流：**Simulink 控制器算法生成代码下载到飞控中，然后用 USB 实体信号线替代 Simulink 中的虚拟信号线。CopterSim 将传感器数据（例如，加速度计、气压计、磁力计等）通过 USB 数据线发送给飞控系统；飞控系统中的 PX4 自驾仪软件将收到传感器数据进行滤波和状态估计，将估计的状态信息通过内部的 uORB 消息总线发送给控制器；控制器再通过 USB 数据线将“rfly_px4”的 uORB 消息回给 CopterSim，CopterSim 通过 UDP 的方式以 40100 系列端口将该数据转发出去，启动 Simulink 的 PX4ExtMsgSender.slx 即可实时监听到所订阅的各种传感器信息，从而形成一个硬件在环仿真闭环。

[2]. **本实验实飞数据流：**在 Simulink 中搭建各种传感器数据的获取文件，该文件内部均是一 uORB 消息订阅的方式订阅各种传感器的数据；自动代码生成的过程中会在在 PX4 软件系统中生成名为 px4_simulink_app 的模块应用，通过烧录到飞控中与其他模块形成并行运行。在 QGC 中通过监听“rfly_px4”的 uORB 消息即可接收所自定义的消息。

[3]. **本实验所涉及到的各种传感器的 msg 文件路径及详解：**

磁力计—路径：*\PX4PSP\Firmware\msg\sensor_mag.msg

```
    uint64 timestamp          # time since system start (microseconds)
    uint64 timestamp_sample

    uint32 device_id          # unique device ID for the sensor that does not change between
power cycles

    float32 x                 # magnetic field in the FRD board frame X-axis in Gauss
    float32 y                 # magnetic field in the FRD board frame Y-axis in Gauss
    float32 z                 # magnetic field in the FRD board frame Z-axis in Gauss
```

```
    float32 temperature       # temperature in degrees Celsius

    uint32 error_count

    bool is_external   # if true the mag is external (i.e. not built into the board)

    uint8 ORB_QUEUE_LENGTH = 4
```

加速度计—*\PX4PSP\Firmware\msg\sensor_accel.msg

```
    uint64 timestamp          # time since system start (microseconds)
    uint64 timestamp_sample

    uint32 device_id          # unique device ID for the sensor that does not change between
power cycles

    float32 x                 # acceleration in the FRD board frame X-axis in m/s^2
    float32 y                 # acceleration in the FRD board frame Y-axis in m/s^2
    float32 z                 # acceleration in the FRD board frame Z-axis in m/s^2

    float32 temperature       # temperature in degrees Celsius

    uint32 error_count

    uint8[3] clip_counter     # clip count per axis in the sample period

    uint8 samples             # number of raw samples that went into this message

    uint8 ORB_QUEUE_LENGTH = 8
```

陀螺仪—*\PX4PSP\Firmware\msg\sensor_gyro.msg

```
    uint64 timestamp          # time since system start (microseconds)
    uint64 timestamp_sample

    uint32 device_id          # unique device ID for the sensor that does not change between
power cycles

    float32 x                 # angular velocity in the FRD board frame X-axis in rad/s
    float32 y                 # angular velocity in the FRD board frame Y-axis in rad/s
    float32 z                 # angular velocity in the FRD board frame Z-axis in rad/s

    float32 temperature       # temperature in degrees Celsius

    uint32 error_count

    uint8 samples             # number of raw samples that went into this message

    uint8 ORB_QUEUE_LENGTH = 8
```

气压计—*\PX4PSP\Firmware\msg\sensor_baro.msg

```
    uint64 timestamp          # time since system start (microseconds)
    uint64 timestamp_sample

    uint32 device_id          # unique device ID for the sensor that does not change between
power cycles

    uint32 error_count
```

```
    float32 pressure   # static pressure measurement in millibar

    float32 temperature# static temperature measurement in deg Celsius
```

GPS—*\PX4PSP\Firmware\msg\sensor_gps.msg

```
    # GPS position in WGS84 coordinates.
    # the field 'timestamp' is for the position & velocity (microseconds)
    uint64 timestamp        # time since system start (microseconds)

    uint32 device_id                # unique device ID for the sensor that does not change be
tween power cycles

    int32 lat          # Latitude in 1E-7 degrees
    int32 lon          # Longitude in 1E-7 degrees
    int32 alt          # Altitude in 1E-3 meters above MSL, (millimetres)
    int32 alt_ellipsoid          # Altitude in 1E-3 meters bove Ellipsoid, (millimetres)

    float32 s_variance_m_s       # GPS speed accuracy estimate, (metres/sec)
    float32 c_variance_rad       # GPS course accuracy estimate, (radians)
    uint8 fix_type # 0-1: no fix, 2: 2D fix, 3: 3D fix, 4: RTCM code differential, 5: Real-Ti
me Kinematic, float, 6: Real-Time Kinematic, fixed, 8: Extrapolated. Some applications will no
t use the value of this field unless it is at least two, so always correctly fill in the fix.

    float32 eph             # GPS horizontal position accuracy (metres)
    float32 epv             # GPS vertical position accuracy (metres)

    float32 hdop            # Horizontal dilution of precision
    float32 vdop            # Vertical dilution of precision

    int32 noise_per_ms      # GPS noise per millisecond
    uint16 automatic_gain_control   # Automatic gain control monitor
    int32 jamming_indicator      # indicates jamming is occurring
    uint8 jamming_state     # indicates whether jamming has been detected or suspected by the
 receivers. 0: Unknown, 1: OK, 2: Warning, 3: Critical

    float32 vel_m_s             # GPS ground speed, (metres/sec)
    float32 vel_n_m_s       # GPS North velocity, (metres/sec)
    float32 vel_e_m_s       # GPS East velocity, (metres/sec)
    float32 vel_d_m_s       # GPS Down velocity, (metres/sec)
    float32 cog_rad             # Course over ground (NOT heading, but direction of movemen
t), -PI..PI, (radians)
    bool vel_ned_valid      # True if NED velocity is valid

    int32 timestamp_time_relative    # timestamp + timestamp_time_relative = Time of the UTC
timestamp since system start, (microseconds)
    uint64 time_utc_usec        # Timestamp (microseconds, UTC), this is the timestamp which
comes from the gps module. It might be unavailable right after cold start, indicated by a valu
e of 0

    uint8 satellites_used       # Number of satellites used

    float32 heading             # heading angle of XYZ body frame rel to NED. Set to NaN if n
ot available and updated (used for dual antenna GPS), (rad, [-PI, PI])
    float32 heading_offset      # heading offset of dual antenna array in body frame. Set to
NaN if not applicable. (rad, [-PI, PI])
```

7. 常见问题

Q1：****

A1：****